

Hardware, Engineering and Functional Description

Control system XControl CPU module XC-CPU201...(-XV)

11/04 AWB2724-1491GB

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

1st published 2003, edition date 12/03
2nd published 2003, edition date 12/03
3rd published 2004, edition date 06/04
4th published 2004, edition date 08/04
5th published 2004, edition date 11/04

See revision protocol in the "About this manual" chapter

© Moeller GmbH, 53105 Bonn

Authors: Werner Albrecht
Editor: Thomas Kracht
Translator: David Long

All rights reserved, including those of the translation.

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

Subject to alteration without notice.



Warning! Dangerous electrical voltage!

Before commencing the installation

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

Contents

About this manual		5
	List of revisions	5
	Assembly	5
	Abbreviations	5
	Reading conventions	6
	Advanced documentation	6
1 Design of the XC200		7
	CPU with PSU and local inputs/outputs	7
	Power supply	8
	– Assembly	8
	Local digital inputs/outputs	8
	– Terminal assignments	9
	– LED indicators	9
	– Local bus expansion with XIOC-BP-EXT	9
	– Extended functions of the local digital inputs	9
	– Connecting the incremental value encoder	11
	– Connecting the up/down counter	12
	– Connecting interrupt inputs	12
	CPU	12
	– Function	12
	– Power stages	12
	– Use of the CPU types	13
	– Assembly	13
	– Monitoring of the system voltage	13
	– Real-time clock	13
	– Operating mode selector switch	13
	– XC200 drives “disk_sys”, “disk_usb” and “disk_mmc”	13
	– Multimedia card (MMC) and USB stick	14
	– USB interface	14
	– ETH232 programming device interface	14
	– CANopen interface	16
	– Real-time clock	18
	– Battery	18
	– CPU installation	18
	– Detaching the CPU	18
2 Engineering		19
	Control panel layout	19
	– Ventilation	19
	– Layout of units	19
	Preventing interference	19
	– Suppressor circuitry for interference sources	19
	– Shielding	19
	Lightning protection	20
	Wiring example	20
	– Connecting the power supply	20
	Power supply of the digital inputs/outputs	20

3 CPU operation	<ul style="list-style-type: none"> Start behaviour after voltage is applied 21 Switch-off behaviour 22 Start behaviour 22 <ul style="list-style-type: none"> – Start conditions in the XSoft 22 HALT 22 Cold start 22 WARMSTART 22 Test and commissioning (Debugging) 22 <ul style="list-style-type: none"> – Breakpoint/single-step mode 22 – Single-cycle mode 22 – Forcing 22 – Status indication 23 Program reset 23 <ul style="list-style-type: none"> – Warm reset 23 – Cold reset 23 – Full reset 23 Program parameterization 23 Save project and generate boot project 23 Updating the operating system 23 <ul style="list-style-type: none"> – Transferring the operating system from the PC to the PLC 24 Erasing the operating system from the Multimedia card 25
4 Program processing, multitasking and system times	<ul style="list-style-type: none"> Task configuration 27 <ul style="list-style-type: none"> – Cyclic task 27 – Event controlled task 28 – System events 28 Multitasking 30 <ul style="list-style-type: none"> – Behaviour of the CAN stack with multitasking 30 – Task monitoring and watchdog 30 – Watchdog configuration 31 – Multiple tasks with the same priority 32 System libraries, function blocks and functions 32 <ul style="list-style-type: none"> – “Lib_Common” library 32 – “Lib_CPU201” library 32 – XS40_MoellerFB.lib “library” 33 – “SysLibFile.lib” library 33 – Examples of the “SysFile...” functions 33 Direct peripheral access 34 <ul style="list-style-type: none"> – ReadBitDirect 36 – ReadWordDirect 36 – ReadDWordDirect 36 – WriteBitDirect 37 – WriteWordDirect 37 – GetSlotPtr 38 – Error code with “direct peripheral access” 38 Data remanence 38 Operating states 39 Web visualization 39 Remote services 39 CANopen bus expansion 39 Limit values for memory usage 40 Download of programs 40

5 PC – XC200 connection set-up		41
	Establishing a connection via the RS232 interface (XC200)	41
	– Programming cable	41
	– Settings in the XSoft	41
	Connection set-up via Ethernet	42
	– Settings in the XSoft	42
	Scan/Modify the IP address	43
6 Configuration and parameterization of the inputs/outputs		45
	Input/output general	45
	Incremental encoder	46
	– Functionality	48
	Counter input	49
	– Functionality	50
	Interrupt processing	52
	– DisableInterrupt	52
	– EnableInterrupt	52
7 XC200 specific functions		55
	Event functions	55
	– IEC_DeleteErrorList	55
	– IEC_DeleteEventList	55
	– IEC_GetErrorID	56
	– IEC_GetEventID	56
	– IEC_GetNrOfErrors	56
	– IEC_GetNrOfEvents	56
	– IEC_WriteError	57
	– IEC_WriteEvent	57
	CAN_Utilities.	57
	Additional functions of the XC200_UTIL2.lib	58
	– UTI2_GetIPConfig	
	Output of the IP address, subnetmask address and IPGateway address	58
	– UTI2_GetMacAddress	
	Output of the MAC address (MAC=Media Access Control)	58
	– UTI2_SetIPConfig	
	Setting of the IP address and subnetmask address	58
	– UTI2_SetIPGateway	
	Setting of the IPGateway address	59
	– UTI2_Reboot	
	Restart with registry save	59
	– UTI2_SaveRegistry	
	Saving of the registry	59

8 Browser commands	61
Communication parameter access	62
– Change baud rate (setcomconfig)	62
– Change IP address (setipconfig)	62
– Change IP gateway address (setipgateway)	63
– Change target name (settargetname)	63
– Parameterize date and time (setrtc)	63
– Display CPU loading (plcload)	63
– Display memory assignment of the “disk_sys” (memdisk_sys)	63
– Browser command “canload” for XC200	64
– Access to memory objects	65
– Error and event list after calling browser commands	65
– Additional help information for the browser commands	66
9 RS232 interface in transparent mode	67
Programming of the RS232 interface in transparent mode	67
– Function “(x)SysComClose”	68
– Function “(x)SysComOpen”	68
– Function “(x)SysComRead”	69
– Function “xSysComReadControl”	70
– Function “(x)SysComSetSettings”	71
– Function “(x)SysComWrite”	72
– Function “(x)SysComWriteControl”	73
– Automatic closing of the interface	73
Appendix	77
USB-Stick-Typen	77
Dimensions	78
– XC-CPU201...	78
– XT-FIL-1 line filter	78
– Module rack	78
Technical data	79
Index	85

About this manual

List of revisions

Edition date	Page	Subject	New	Modified
12/03 (Reprint)	38	Data remanence, 1 st paragraph		✓
04/04	40	Limit values for memory usage	✓	
	37	WriteBitDirect		✓
06/04	20, 78, 83	"External 24 V DC line filter for the XC200 power supply"	✓	✓
08/04	38	Data remanence, note	✓	
	40	Download of programs	✓	
	65	"RS232 interface of the XIOC-SER in transparent mode (COM2/3/4/5)"	✓	
	82	Electromagnetic compatibility		✓
11/04	14, 77	Multimedia card (MMC) and USB stick	✓	
	15	Splitting of the ETH232 interface of the CPU		✓
	23	Status indication	✓	
	41	Establishing a connection via the RS232 interface (XC200)	✓	
	55	XC200 specific functions	✓	
	58	Additional functions of the XC200_UTIL2.lib		✓
	67	RS232 interface in transparent mode	✓	

Assembly

The XC-CPU201-... control units – referred to as XC200 in this manual – are designed for use on machines and industrial and building control systems.

XSoft from Version 2.3 is required for programming the XC200.

These controls form the basis for the configuration of a comprehensive automation system with their interfaces for connection to programming devices (RS232/Ethernet), centralized interfacing of XI/OC signal modules and decentralized interfacing of CANopen expansions.

The manual is laid out as follows:

The chapter "Hardware and installation" tells you about installing the controllers, project engineering, and the settings that you can make on the controllers.

Communication with a controller is established by connecting it to a PC, via the programming device interface. Configuration of the XC200 is described in chapter "PC – XC200 connection set-up".

Abbreviations

The following abbreviations are used in this manual:

MWS	Menu selector switch
BAS	Operating mode switch
CPU	Central processing unit
CRC	Cyclic redundancy check
MMC	Multimedia card
BTS	Operating system
POU	Program organization unit (program, function, function element)
I/O	Inputs/outputs

► Indicates instructions on what to do

Reading conventions

Selecting «File → New» means: activate the command “New” in the “File” menu.

**Important**

Warns of a hazardous situation that could result in damage to the product or components.

**Caution!**

Indicates the risk of major damage to property, or slight injury.

**Warning!**

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

In order to provide a clear layout, the chapter title is shown in the header on left-hand pages, and the current section on right-hand pages. Exceptions are at the first pages of chapters and empty pages at the end of chapters.

Advanced documentation

At different positions 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. AWB2700-1437GB). All manuals are available in PDF format. If for some reason the manual is not supplied on the product CD, it is available for download as a PDF file. In order to find the document quickly go to <http://www.moeller.net/support>: Enter the document number as a search term.

1 Design of the XC200

The XC200 controller has a compact design, and can be fitted with either local or decentralized expansion. The control system consists of:

- Module rack
- A CPU for control or visualisation, with integral power supply and local inputs/outputs.
- XIOC signal modules.

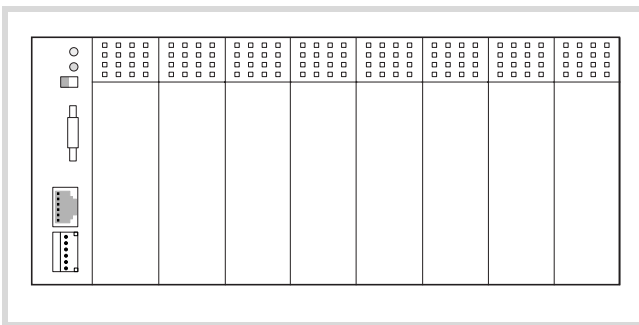


Figure 1: Layout of the XC-CPU201 with XIOC modules

→ Further details about the CPU can be found in the next section.

Detailed information about the module racks and XIOC modules can be found in the manual "Hardware and Engineering, XIOC Signal modules".

CPU with PSU and local inputs/outputs

The CPU module of the XC200 has a compact design that is divided into two functional units:

- Processor unit with interfaces
- 24 V power supply with integral digital inputs (eight) and digital outputs (six).



Figure 2: Assembly of the CPU module XC-CPU201

- ① Processor unit
- ② 24 V power supply with local inputs/outputs

Power supply

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

Assembly

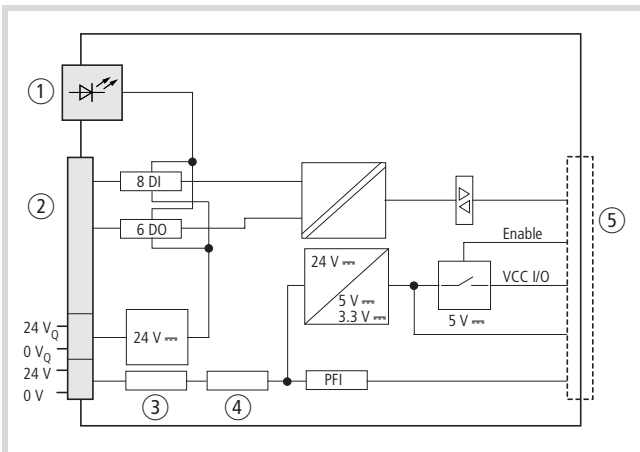


Figure 3: Block diagram: power supply unit

- ① Input/output status indication
 - ② Front connection terminals
 - ③ Internal filter
 - ④ Buffer
 - ⑤ XIOC I/O-bus, module rack
- PFI = Power Fail Interrupt

The voltage connection 0V_Q/24V_Q is only for the supply voltage to the integral local inputs (8) and outputs (6), and is electrically isolated from the bus.

The 0V/24V voltage connection is fed to a DC-DC converter which generates the necessary system voltages. The internal power supply for the 5 V system voltage is designed so that the processor unit is supplied with the required current.

Caution!
When using the XC200-CPU and the XIOC-Signal modules in an ABS plastic enclosure, the limitations stated in Table 1 apply. ABS enclosures are identified with "ABS" on the surface which faces the backplane.

Table 1: Limitations which apply when using the XC200-CPU and the XIOC-Signal modules in an ABS plastic enclosure

Fitted in:	Installation location internal temperature:	Current rating of the 5 V system voltage of the I/O bus
CI enclosure	> 40 °C	Use of the XC200 not permissible
	0 to 40 °C	max. 1.5 A ¹
Distribution fuse-board	0 to 55 °C	max. 1.5 A ¹
Control panel	> 40 °C	max. 1.5 A ¹
	0 to 40 °C	max. 3.2 A

1) On the outputs of the CPU made of ABS enclosure material, a utilization factor g of 0.5 applies

➔ Limitations in performance for the digital I/O modules with ABS enclosures are described in the documentation for the XIOC signal modules (AWB2725-1452GB).

If there is an interruption break or collapse of the 24 V supply (threshold is about 10 V) then a power-down logic switches of the 5 V supply to the signal modules (central I/O). The sequence is initiated by the PFI signal and leads to a power-down through the CPU.

Local digital inputs/outputs

The 18-pole terminal block which has the power supply to the CPU, the local I/Os and the physical connection to the local inputs/outputs is located on the right half of the CPU behind the front enclosure.

The eight digital inputs and six semiconductor outputs are designed for 24 V signals and have a common 0V_Q/24V_Q 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".

Important
Please observe the limitations of performance for the outputs with ABS enclosures in ➔ table 1.

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

Terminal assignments

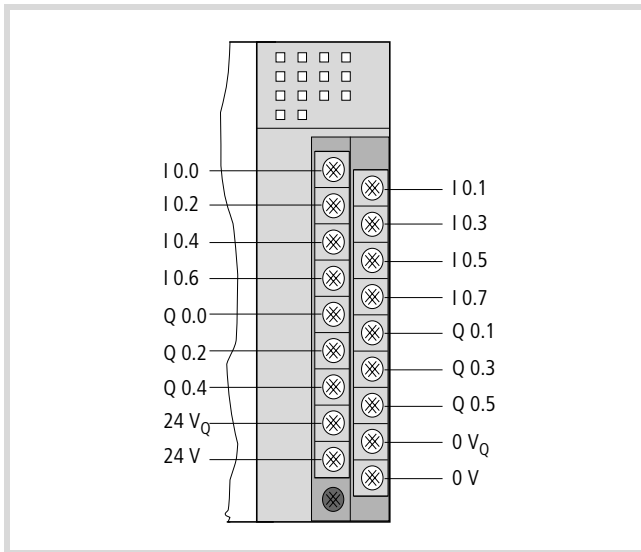


Figure 4: Power supply and local input/output connections

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

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 terminal.

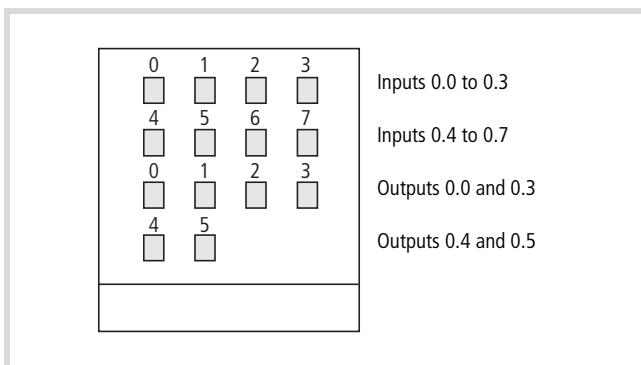


Figure 5: LEDs for the local inputs/outputs

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).

Local bus expansion with XIOC-BP-EXT

The XIOC-BP-EXT backplane enables expansion of local system busses from a max. of 7 to a max. of 15 slots.

The intelligent modules such as network and gateway modules can only be inserted into I/O slots 1 to 3. All other modules can be connected to any slot.

The possible arrangement of the backplane is described in the documentation of the XIOC signal modules (AWB2725-1452GB). Please pay attention to the current requirements, particularly the current supplied by the power supply and the current requirement of the signal modules.

Further information can be found in the "XIOC signal modules" (AWB2725-1452GB) documentation. Integration of the bus expansion via the software is explained in the "Expansion of the XIOC bus" section.

Extended functions of the local digital inputs

In addition to the function as normal 24 V inputs, one section of the inputs also possesses an extended functional range. This extendable function range can be preselected and configured in the XSoft.

The following functions are available:

Table 2: Extended input functional range

General information about functions 1 to 3		
If the inputs are parameterized as incremental encoders or counter inputs, pre-processing of the incoming signals is undertaken by the input hardware. The result is read by the CPU and processed accordingly in the user program.		
Function 1: Incremental encoder input for 32 bit data capacity		
Input assignment:	I0.0	A-Signal
	I0.1	B-Signal
	I0.2	C-Signal or marker pulse
	I0.3	Reference window; e.g. for connection of a position-switch.

The encoder signals A and B are electrically phase-shifted by 90°. The count direction is automatically derived from the phase shift. The rising and falling edges of the A and B signals are analyzed (quadruple evaluation). Max. input frequency: 50 kHz (track frequency). The quadruple evaluation therefore results in a total count frequency of 200 kHz.

Function 2:**1 x up/down counter for a data capacity of 32 bits**

Input assignment:	I0.0	Counter counting signal
	I0.1	Direction signal

Max. input frequency: 50 kHz

The counting signals are read and passed onto pre-processing via the counter count input. Incrementation/decrementation of the input signals is undertaken by the hardware here.

The "direction signal" is a static signal which must be present before the incoming counting signals.

Function 3:**2 x up/down counters for a data capacity of 16 bits each**

Input assignment:	I0.0	Counter -1- counting signal
	I0.1	Counter -1- direction signal
	I0.2	Counter -2- counting signal
	I0.3	Counter -2- direction signal

Max. input frequency: 50 kHz per channel

The demands placed on the counter count input and on the "Direction signal" are the same as those already described with "Function 2".

Function 4: Interrupt inputs

The XC200 supports up to four local user 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. The interrupt inputs I0.4 and I0.5 are complemented by 2 interrupt signals which are generated by the rotary encoder/counter functions.

Parametric programming and priority assignment of the interrupts occurs in the "PLC Configuration and Task Configuration" of the XSoft. In the Task configuration ("Task configuration → System events") each user interrupt can be assigned with an executable user program (POU). This POU is executed when the interrupt occurs.

The interrupts are enabled when changed to the RUN state and inhibited when changed to the STOP state. Interrupt sources not enabled in the configuration do not cause interrupts; the input is then a normal digital input.

**Important**

With an overload of the system caused by the frequent occurrence of interrupts during a cycle and the associated overrange or the watchdog time, the controller will STOP.

The user interrupts can be inhibited and re-enabled from the user program. The "DisableInterrupt" and "EnableInterrupt" XSoft functions are available for this purpose. A call parameter can be used to determine if the interrupts are inhibited and enabled individually or all at the same time. Enabling of an interrupt must be performed with the same parameter used to inhibit it, see → section "Direct peripheral access" on page 34.

Time constraints placed on the interrupt inputs: refer to "Technical data – input delay – fast digital input".

Programming in the "Interrupt function" is described on page 52.



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

Function 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. Time constraints placed on the "direct peripheral access": refer to "Technical data – input delay – fast digital input".

The local and central input and output signals are the input and output signals of the CPU and the centrally expanded XC200 control with the XIOC signal modules. XIOC signal modules which can be integrated via a bus system can't be accessed via the "Direct peripheral access".

Addressing is dependent on the slot number "0 to 7 (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 parameters are indicated in Table 8.

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

Connecting the incremental value encoder

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

The reference window depicts the range in which the marker signal of the rotary encoder (C or marker pulse) is to be evaluated. The marker pulse generally occurs just once per revolution with incremental encoders.

The L/H edge of the marker signal will set the current counter state with a reset pulse to "0", only when the reference window signal is active.

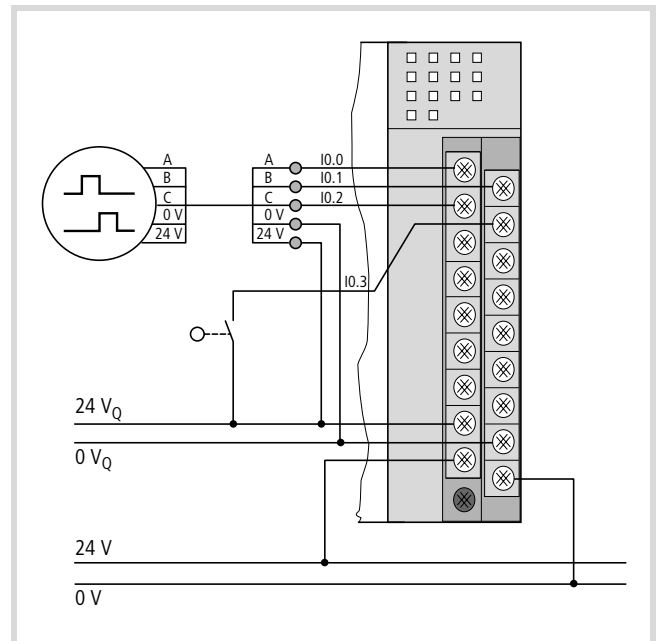


Figure 6: Connection of the incremental value encoder with a reference window position switch

→ Ensure that in the range where the reference window is active, that the marker pulse is only available once and that it is long enough to ensure that the marker pulse is evaluated safely and reliably.

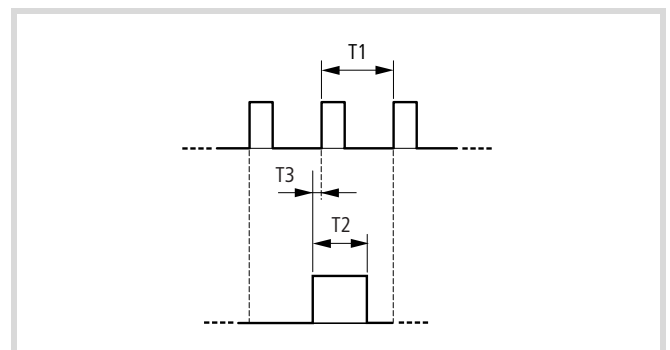


Figure 7: Relationship between reference signal and reference window

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

Connecting the up/down counter

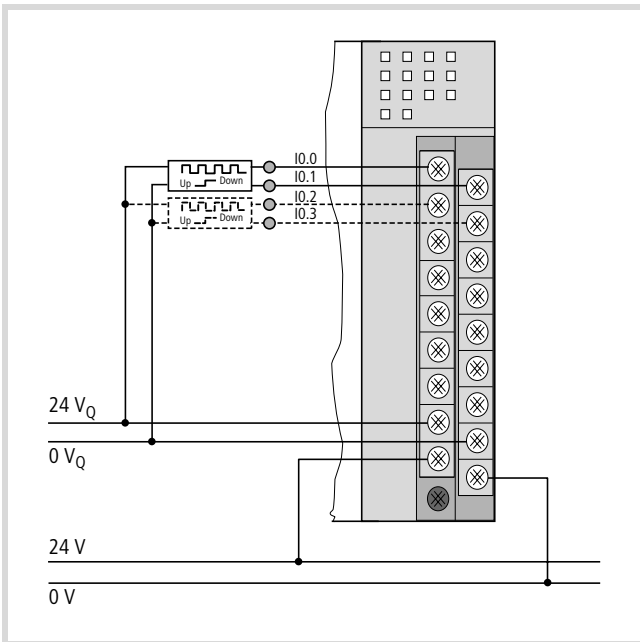


Figure 8: Pulse generator connection with preselection of the count direction

Connecting interrupt inputs

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

The interrupt inputs are effective directly and independently of the cycle time on the processor, and starting of the programmed interrupt routines. The program section which has been processed up to the arrival of the Interrupt signal is interrupted immediately. All further operations should be programmed to suit the application.

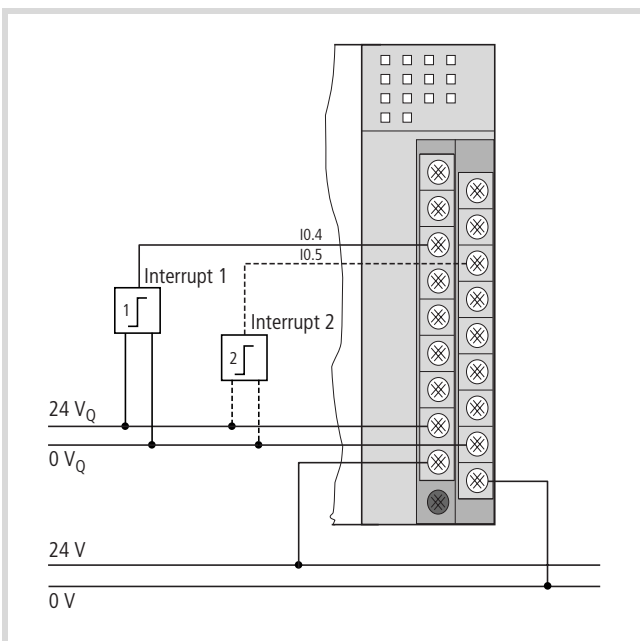


Figure 9: Interrupt input connections

Two further interrupt signals are generated by the rotary encoder/counter functions and are thus no physical inputs as such.

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

CPU

Function

The task of the CPU is to generate output signals from the incoming local and central/decentralized signal, in accordance with the application program.

Input/output signal can be, for instance:

- Digital inputs/outputs
- Analog inputs/outputs
- Communication via the MMC interface
- Communication with the programming system via RS232 and/ or Ethernet
- Communication via the USB interface
- Communication via the CANopen fieldbus interface
- Communication via fieldbus modules, if they are in the design
- Communication with intelligent XI/OC signal modules, if they are in the design.

Power stages

The CPUs for XC200 controllers are available in various different versions/performance levels:

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

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

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

	XC-CPU201-...-8DI-6DO(-XV)	
	... EC256K	... EC512K
Program code	256 kByte	512 Kbyte
Program data, of which:	256 kByte	512 Kbyte
markers	16 kByte	16 kByte
Retain data	32 Kbyte	32 Kbyte
Persistent data	32 Kbyte	32 Kbyte

Use of the CPU types

Table 3: Combination possibilities of the XC200 with display

CPU types	PLC	Web
XC-CPU201...	✓	–
XC-CPU201... (-XV)	✓	✓

Assembly

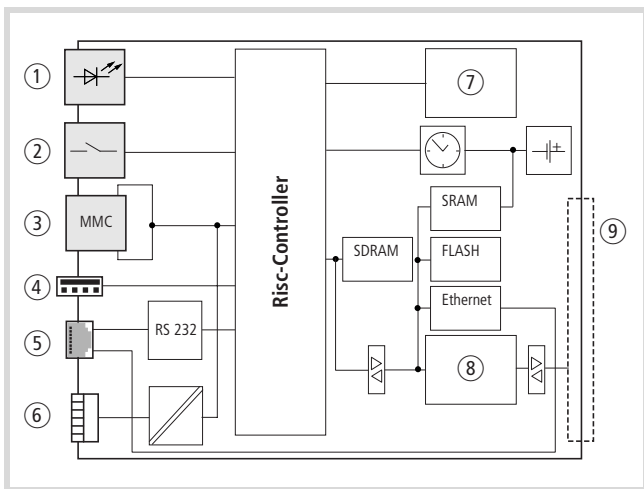


Figure 10: Block diagram of the XC-CPU200

- ① State indication RUN, STOP, SF
→ chapter "Operating states" on page 39
- ② Operating mode selector switch, → page 13
- ③ Multi Media Card (Drive: disk_mmc) → page 14
- ④ USB interface (Drive: disk_usb) → page 14
- ⑤ Programming device: RS 232/Ethernet
→ page 14
- ⑥ CANOpen interface, → page 16
- ⑦ Monitoring of the system voltage, → page 13
- ⑧ I/O bus interface
- ⑨ XIOC I/O bus (on module rack)

Monitoring of the system voltage

The monitoring of the system voltage ensures that the data-saving routine will be initiated if the voltage goes below a fixed preselected level. In order to ensure that the stored energy required for the data-saving routine is not used up by I/O activities, the system voltage for the I/O modules is switched off.

Real-time clock

The internal real-time clock facilitates time and date dependent control functions.

Operating mode selector switch

The operating modes "Stop" and "Run" are selected by a rocker switch at the front of the CPU module. Please note that the position of the operating mode selector switch sets the behaviour of the CPU. The effectiveness of the software settings depends on the position of the operating mode selector switch. If the selector switch is changed to the "Stop" position while the equipment is in the "Run" mode, then the CPU will switch from the operating mode "Run" to the "Stop" state at the end of the cycle that is running at the moment. The position of the operating mode selector switch is polled at the end of each cycle, and the controller switches to the selected state, → chapter "CPU operation".

XC200 drives "disk_sys", "disk_usb" and "disk_mmc"

The XC200 has the following drives available:

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

The boot system and the operating system are saved in compressed format and protected against failure of the power supply in the transaction safe system memory. In the operating state, the boot project and the relevant sections of operating system are "unpacked" and copied into SDRAM 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 11 indicates the interaction of the differing XC200 memory systems/drives:

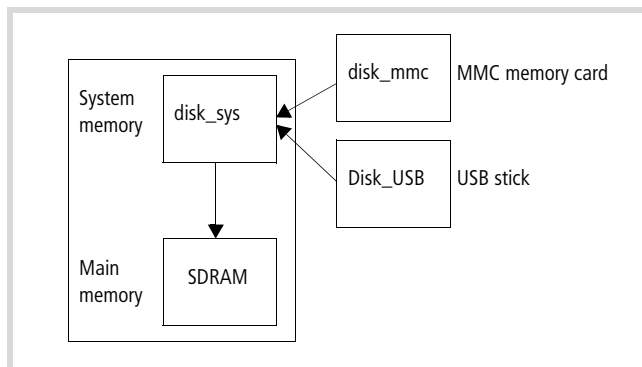


Figure 11: XC200 memory organization

Multimedia card (MMC) and USB stick

→ Use of a USB stick is possible from operating system version V01.03. The USB stick types are supported with the FAT16 file system, → chapter “USB-Stick-Typen” on page 77.

MMC and USB stick serve as bulk storage. They are written or read with the assistance of browser commands or functions. They can contain recipe data, the user program or general files may also be stored. Use the “copyprojtomm” browser command for example, to copy the user program on the MMC.

A description of the browser commands can be found from page 61. The functions are contained in the “SysLibFile.lib” library and on page 33.

Note!
The file system of the memory card is not transaction-safe. Ensure that all files have been closed by the program before your remove or insert a card, or switch off the voltage.

→ also section “Erasing the operating system from the Multimedia card” page 25

USB interface

Table 4: Assignment of the USB interface

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

ETH232 programming device interface

Communication between the controller and the programming device takes place through the ETH232 programming device interface of the CPU. The plug socket contains two interfaces: RS 232 and Ethernet.

The Ethernet interface is used for programming and debugging, as it is processed more quickly by the operating system.

The RS232 interface is a point-to-point connection, the Ethernet interface can be networked.

→ From operating system V01.03 you can also switch the RS-232 interface into the transparent mode.

Physically the programming device interface is provided as an RJ45 jack. This means that normal commercial RJ45 connectors or Ethernet patch cables can be used .

→ A crossover cable is used as the connection cable between the XC200 and the programming device.

Crossover cables have the following design features:

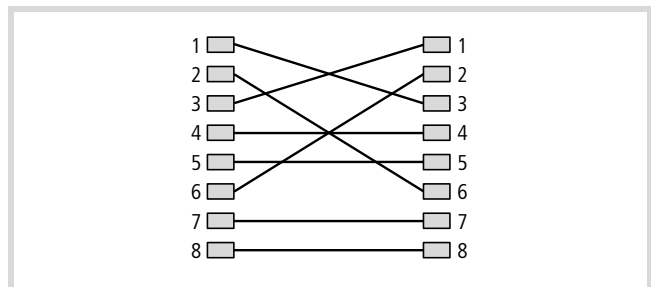


Figure 12: Connection set-up of an 8-pole crossover cable

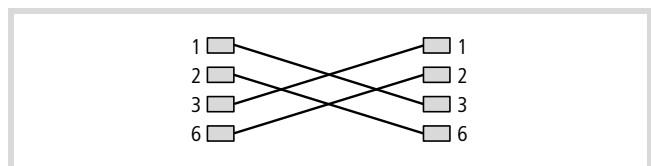


Figure 13: Connection set-up of a 4-pole crossover cable

Electrical isolation

The RS232 interface is not electrically isolated. This Ethernet interface is electrically isolated.

→ Please note that when there is a double assignment of the RJ 45 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.

Assignment of the ETH232 programming device interface

		Signal	
		RS 232	Ethernet
8	8	RxD	-
7	7	GND	1)
6	6	-	Tx-
5	5	TxD	-
4	4	GND	1)
3	3	-	Tx+
2	2	-	RD-
1	1	-	RD+

1) Pin 4 and 7 may not be used

Splitting of the ETH232 interface of the CPU

Using the XT-RJ45-ETH-RS232 cable splitter, it is possible to communicate using the RS232 and the Ethernet interface 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 connector of the CPU.

On the Ethernet interface of the cable splitter you can for example, connect the programming device and the RS232 interface (in transparent mode) of a printer. The pin assignment of the RS232 and Ethernet socket of the cable splitter correspond with the pin assignment of the ETH232 connector of the CPU, → section "Assignment of the ETH232 programming device interface" on page 15.

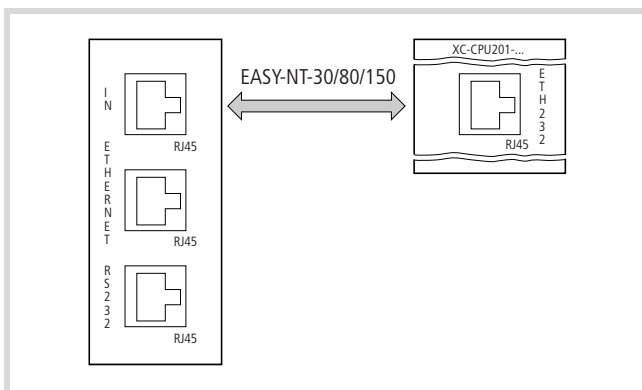


Figure 14: Connection of the XC-CPU201 with the XT-RJ45-ETH-RS232

Activating data transfer rate

- Connection for download of the operating system
The data transfer rate is set to a fixed value of 115200 Bit/s for operating system download.

- Connection for programming
The data transfer rate for programming is set by default to 38400 Bit/s. Modification of the data transfer rate is possible as follows:

- ▶ Select "PLS 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.
- ▶ Select the "Serial Communication Parameters" command in "Online" and update the baud rate as described in the following.

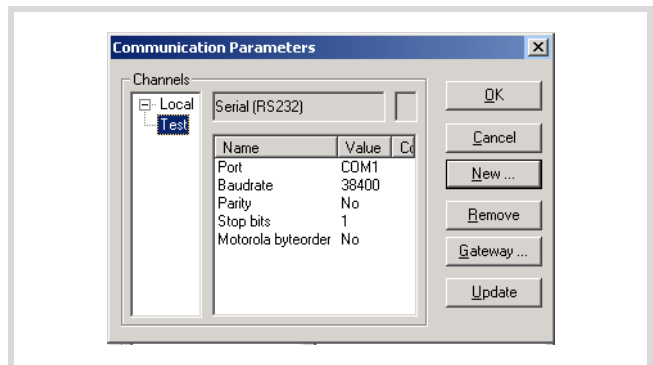


Figure 15: Communication parameters for programming

- ▶ Use a double-click to select the field with the preset baud rate.

This field now has a grey background.

- ▶ With further double-clicks in this field you can select the required baud rate, e.g. 115200 Bit/s, and confirm the selection with "OK".
- ▶ Select the menu <Online → Log-in> again.

Once again, you will see the following message:

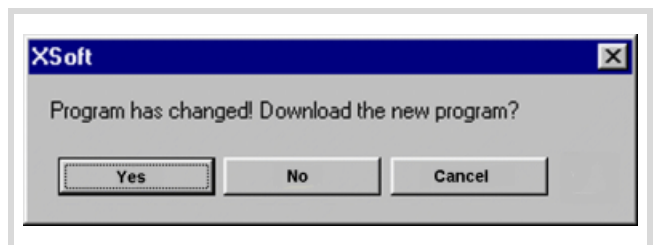


Figure 16: Program change; reload?

- ▶ Answer here with "Yes".
- ▶ Select the menu <Online → Start> . This puts the controller into the RUN mode.

The subsequent communication between the XC200 and the PC (as the programming device) will be made at the selected transmission rate.

Operating the Ethernet interface

The transmission medium used is a two-core twisted pair (10BaseT) cable with or without screen (SSTP, STP, UTP). Various network topologies such as ring, line (bus) and star designs can be used with Ethernet.

Ethernet connections

- PC – XC200 connection
The XC200 can be connected directly to the (programming) PC via a crossover Ethernet cable, → figure 12, 13.

• PC – Hub/Switch – XC200 connection

If a Hub or Switch is used for the connection between the PC – XC200 for network topology reasons, a standard Ethernet cable must be used for the connection between the PC – Hub/Switch and the Hub/Switch – XC200. This cable is connected 1 : 1.

The following Ethernet cable is available:

- Crossover cable
 - XT-CAT5-X-2 2 m long
 - XT-CAT5-X-5 5 m long
- 1:1 cable
 - CAT5-KG2.0 2 m long
 - CAT5-KG5.0 5 m long
 - CAT5-KG10.0 10 m long

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 5: Characteristic of the Ethernet cable

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

- 1) Use in industrial environments is not recommended due to poor EMC characteristics.
- 2) The conductor pairs are shrouded in a full screen. 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 meters. If the network is larger, suitable infrastructure components should be used. Transceivers, Hubs and Switches are particularly suitable.

Decisive with the selection of the cable are the environmental conditions (interference, flexibility, data transfer rate) at the location used.

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

CANopen interface

The CPUs can be connected to the CANopen bus via the electrically isolated ISO-11898 interface.

Assignment of the CANopen interface

The connector assignments are as follows:

	Terminal	Signal
	6	GND
	5	CAN_L
	4	CAN_H
	3	GND
	2	CAN_L
	1	CAN_H

Connector type: 6-pole, plug-in spring-loaded terminal block, conductor cross-section up to 0.5 mm²

The CPUs can be operated on the CAN bus either as the network (NMT) master or as the NMT slave.

The CPU can be used to send/receive CAN telegrams directly to/from the user program. An interruption on the CAN Bus will only be recognised when the respective CAN slave is monitored by the PLC (Nodeguarding function).

Power supply

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

Start/Stop behaviour

If you set the operating mode selector to the “Stop” position, all outputs of the decentralized devices will be set to the “0” level at the end of the cycle.

Bus termination resistors

The ends of the network link must be terminated with 120 Ω bus termination resistors:

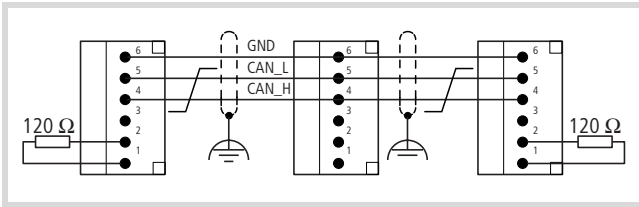


Figure 17: Example: possible configuration of a CANopen bus with bus termination resistors

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

Properties of the CANopen cable

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

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

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

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

Table 6: Standard parameters for CANopen network cable according to the ISO 11898

Bus length [m]	Loop resistance [mΩ/m]	Core cross-section [mm ²]	Bus termination resistor [Ω]	Transfer rate with cable length [kBit/s]
0 – 40	70	0,25 – 0,34	124	1000 at 40 m
40 – 300	< 60	0,34 – 0,6	150 – 300	> 500 at 100 m
300 – 600	< 40	0,5 – 0,6	150 – 300	> 100 at 500 m
600 – 1000	< 26	0,75 – 0,8	150 – 300	> 50 at 1000 m

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

Table 7: Cable cross-section, bus length and number of bus slaves conform to ISO 11898

Cable cross-section [mm]	Maximum length [m]		
	n = 32	n = 64	n = 100
0,25	200	170	150
0,5	360	310	270
0,75	550	470	410

n = number of connected bus users

If the bus length is greater than 250 m and/or are more than 64 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 "XSoft" programming software.

Cable recommendation:
LAPP-Cable
UNITRONIC-BUS LD

Real-time clock

The XC200 is equipped with a real-time clock which can be accessed by functions in the user program.

The real-time clock can be set or queried with the following browser commands:

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

Further information concerning the browser commands can be found at page 61.

Battery

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

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



Important

To avoid loss of data, the battery must be changed when the low threshold level has been reached.

Order designation of the battery: XT-CPU-BAT-1

CPU installation



Detailed information about the installation of the module racks and XI/OC modules can be found in the manual "Hardware and Engineering XI/OC Signal Modules" (AWB2725-1452GB). This manual is provided as a PDF file (h1452g.pdf) on the CD. Here you can also find further information on the various types of module rack and the individual slot assignments for the CPU and the XI/OC signal modules.

The latest versions of specific manuals can be found at <http://www.moeller.net/support>
: Search item: AWB2725-1452GB)

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

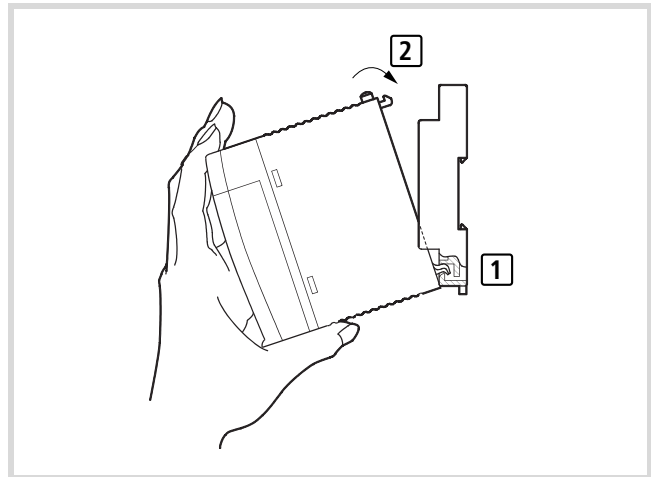


Figure 18: CPU installation

Detaching the CPU

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

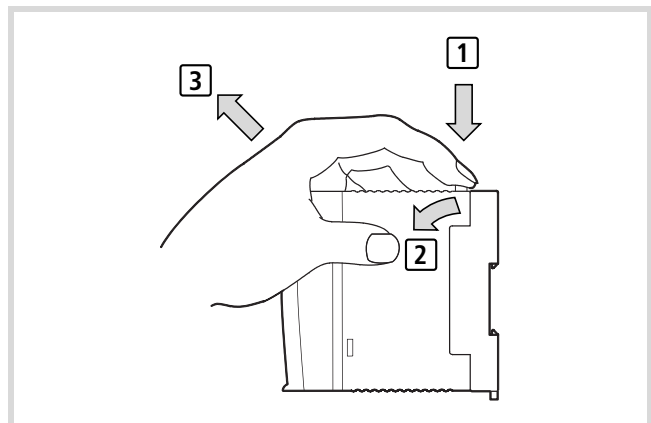


Figure 19: Detaching the modules

2 Engineering

Control panel layout

The layout of the components inside the switchgear cabinet 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/interfaces components
- Transformers
- Frequency inverters
- Converters

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

Ventilation

A clear space of at least 50 mm must be kept between passive components, to ensure adequate ventilation. If the neighbouring components are active elements, such as power supplies or transformers, then the minimum spacing should be 75 mm. The values that are given in the technical data must be observed.

Layout of units

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

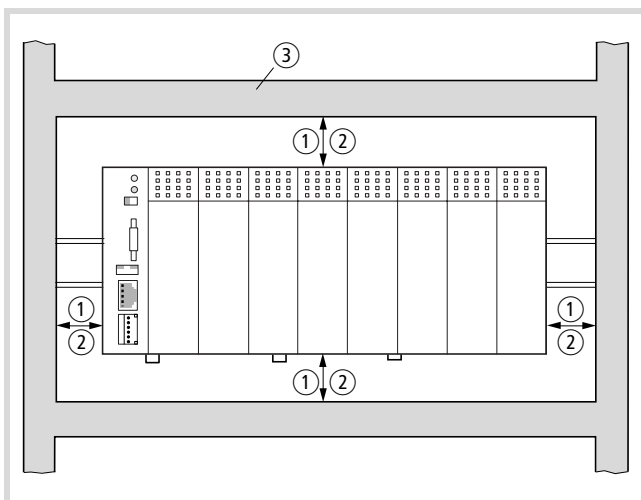


Figure 20: Cabinet layout

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

Preventing interference

Cable routing and wiring

Cables are divided into the following categories:

- Power cables (e.g. cables that carry high currents, or cables to converters, contactors, solenoids)
- Control and signal cables (e.g. for digital inputs)
- Measurement and signal cables (e.g. for fieldbus connections)

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

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

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

Suppressor circuitry for interference sources

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

→ Switched inductors should always have suppressor circuitry fitted.

Shielding

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

Lightning protection

External lightning protection

All cables that go outside buildings must be shielded. Metal conduit is the best solution to this problem. Use protective components against overvoltage for the signal cables, e.g. such as Varistors or other overvoltage surge voltage protectors. The measures should be undertaken at best where the cable enters the building, at the latest however on the control panel.

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:

- equipotential bonding/earthing
- shielding
- using overvoltage protection devices.

Please consult the following manuals for advice on cable routing and shielding measures:

- AWB27-1287 "EMC Project Engineering for Automation Systems".
- TB27-001-GB "Electromagnetic Compatibility (EMC) for Automation systems".
- TB02-022-GB "Electromagnetic Compatibility (EMC) for Machinery and Plant".

Wiring example

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

Connecting the power supply

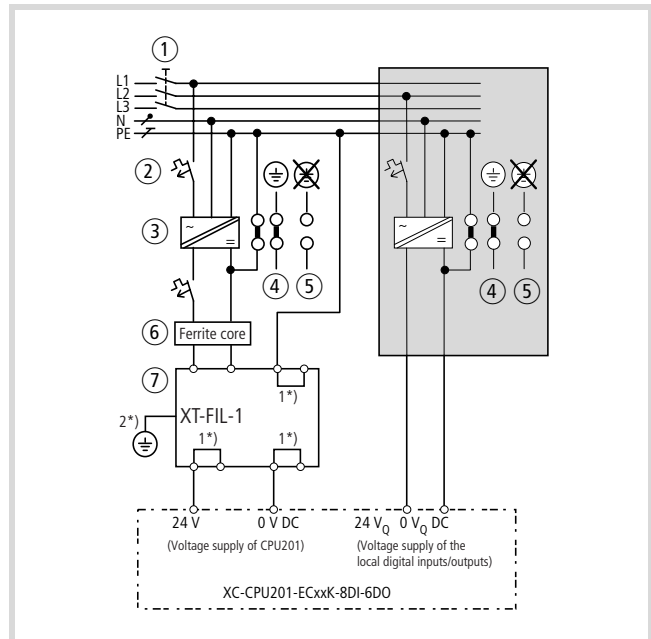


Figure 21: Wiring example for the supply section

- ① Main switch
- ② Protective cut-out
- ③ 24 V DC supply voltage
- ④ Earthed operation
- ⑤ In floating (i.e. unearthed) operation, an isolation monitor must be used (IEC 204-1, EN 60204-1, DIN EN 60204-1)
- ⑥ Ferrite ring (Type: PS416-ZBX-405, Article No.: 025519)
- ⑦ 24 V DC line filter; ensures that a current of up to 2.2 A (maximum) is available at a rated voltage of 24 V DC. Ensures that the EMC stipulations for devices are fulfilled when the filter is used. The filter is not a component of the CPU and must therefore be ordered separately:
Type: XT-FIL-1, Article no.: 285316 (Supplier: Moeller GmbH)
→ Dimensionson page 78
→ Technical data on page 83.

1*) Internal jumper

2*) Additional PE connection via contact spring on rear

Power supply of the digital inputs/outputs

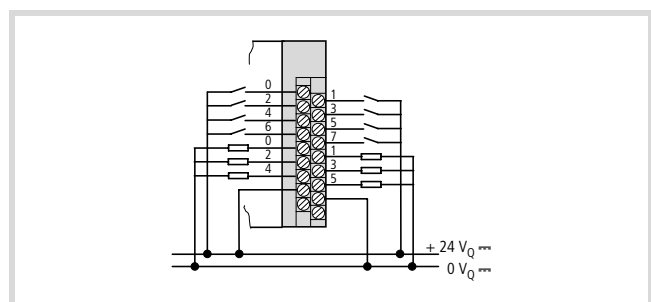


Figure 22: Wiring example for the power supply of the inputs/outputs

The wiring example indicates the wiring of a separate voltage supply for inputs/outputs.

3 CPU operation

Start behaviour after voltage is applied

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

The flow diagram (Fig. 23) indicates which program is used. The diagram shows the update of the operating system using the MMC.

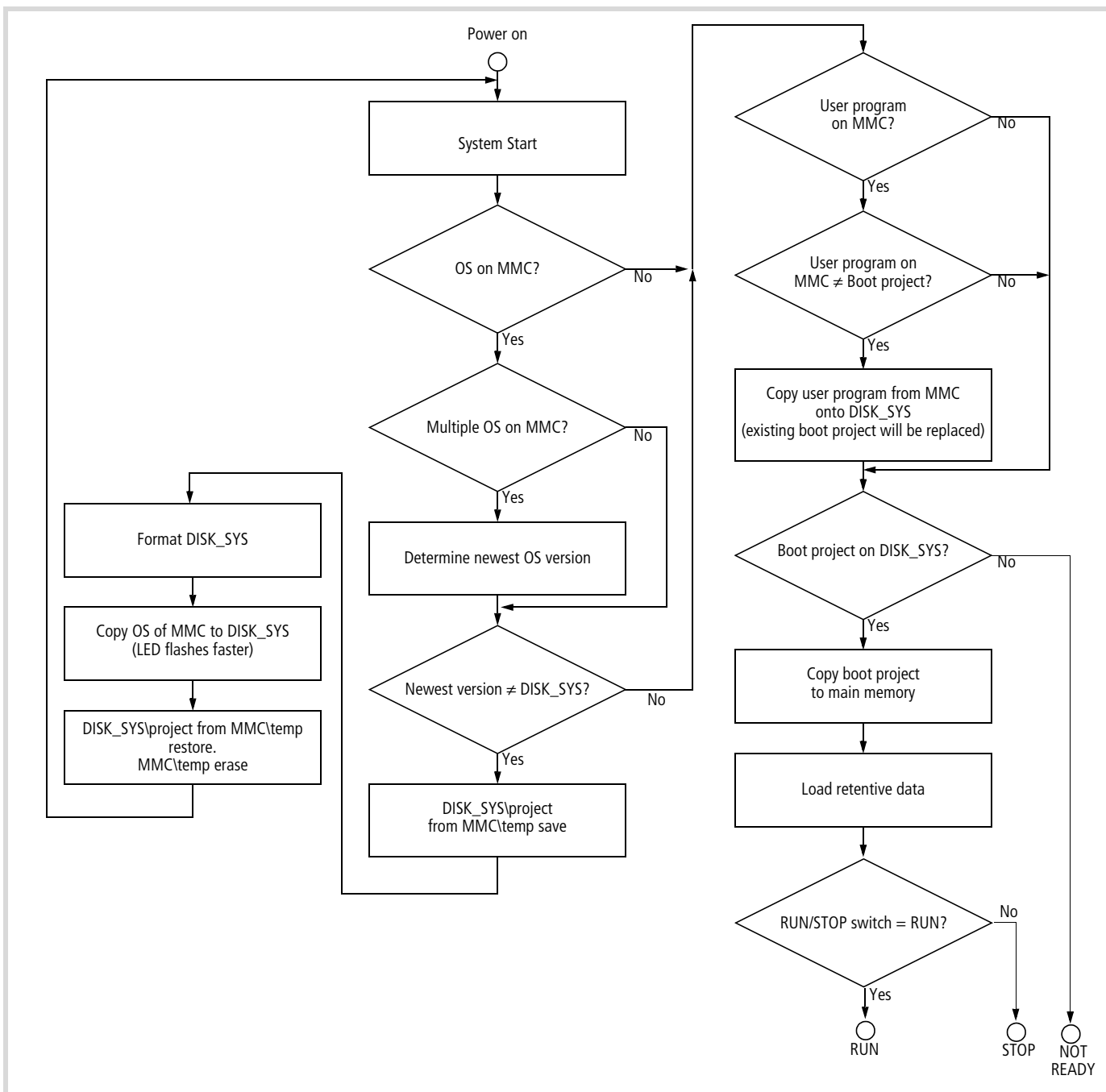


Figure 23: Switch-on behaviour

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

**Important**

During engineering and programming of an application, ensure that the relevant safety regulations which apply to the branch of industry are observed, e.g. such as the DIN/EN 60204 "Electrical Equipment of Machinery" or the relevant accident prevention regulations.

Switch-off behaviour

Switching off (Operating mode selector switch: RUN → STOP) leads to an interruption of the program execution after all active tasks have been completed. After the task has ended the outputs used by the I/O task are set to "0", → chapter "Program processing, multitasking and system times" on page 27.

During a voltage dip (recognition via the PFI signal) the execution of the program is ended immediately and the outputs are switched off. When the supply voltage recovers the controller carries out a restart.

Start behaviour

The start behaviour of the PLC when the voltage is switched on is dependent on:

- the position of the local operating mode selector switch
- the start conditions parameterized in the XSoft programming system.



When changing the operating status from STOP to RUN, the operating mode selector switch must be switched to the RUN position.

When a program starts, a check is made whether the configured inputs and outputs match those that are actually present. A check is also made to verify that the module that was parameterized is physically present, or if a different module is present. An incorrect module type prevents the start of the user program, → chapter "Configuration and parameterization of the inputs/outputs" on page 45.

Start conditions in the XSoft

This setting defines how the controller should respond after switch-on, if an application program is present and the operating mode selector switch is in the "Run" position.

The following settings are available:

- HALT
- COLDSTART
- WARMSTART (default setting)

Refer also the respective subsequent sections!

HALT

The application is not started automatically.

Cold start

A cold start is only performed when the voltage is applied after the program has been loaded into the controller. During this start, all the program variables are set to their initialisation values and the program is started (see Table on page 39).

WARMSTART

All subsequent starts of the program that has been loaded are warm starts. The variables that were declared with "RETAIN" retain their values, the other variables are set to their initialisation values. Variables which have not been explicitly assigned with an initialization value are set to the standard initialization values (see Table on page 39).

Test and commissioning (Debugging)

The PLC supports the following test and commissioning features:

- Breakpoint/single-step mode
- Single-cycle mode
- Forcing
- Online modification
- Status indication/Powerflow.

Breakpoint/single-step mode

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

**Important**

The outputs that were already set when the breakpoint occurred remain set!

Single-cycle mode

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

Forcing

All the variables in an application program can be forced to a given setting. If variables for physical outputs of the local I/Os are forced, they will only be connected through to the peripherals in the "Run" state.

Status indication

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

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

in IL:

```
LD   %IB0
ST   Defaultbyte
LD   Defaultbyte
ST   %QB0
```

Program reset

The application program can be reset to one of the following levels:

- Warm reset
- Cold reset
- Full reset

Warm reset

This correspond to the initialisation during a warm start, see section "WARMSTART" on page 22.

Cold reset

This correspond to the initialisation during a cold start, see section "Cold start" on page 22.

Full reset

This command erases all variables – even the retentive variables (VAR RETAIN, VAR PERSISTANT). The application program in the PLC as well as the boot project are erased and the PLC is reset to the initial default state. After this, the controller is in the "NOT READY" state → table on page 39.

Program parameterization

An application program has various parameters that can be set or adjusted in the programming system:

- Maximum watchdog time of the program
- Maximum program cycle time
- Start behaviour at Power-On

Save project and generate boot project

After the program has been generated, transfer the program to the control (log on). This is how to transfer the program to main memory, → figure 11 on page 13. This memory is not zero-voltage safe.

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 the XSoft programming system in order to create a boot project:

- ▶ Change over to the "Online" folder.
- ▶ Select the "Logon" command.
- ▶ Select the "Generate boot project" command.

Updating the operating system

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

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

You have two options to transfer the OS.

- Directly from the PC into the PLC (only via RS 232)
- From the PC via the PLC on the MMC into the "disk_mmcmoellerXC-CPU201" folder (only via Ethernet).

A transfer of the OS from the PC to the MMC of the PLC is possible only when the PLC has an OS from version 01.03.00 or higher.

Transferring the operating system from the PC to the PLC

If an OS is loaded into the PLC, the existing OS as well as the user program are erased.

Procedure

- ▶ Establish a serial connection via the RS232 interface of the PLC with the XC200.
- ▶ Activate the "Other Parameters" tab in the "PLC Configuration" window and click on the → figure 24 "Start" button.

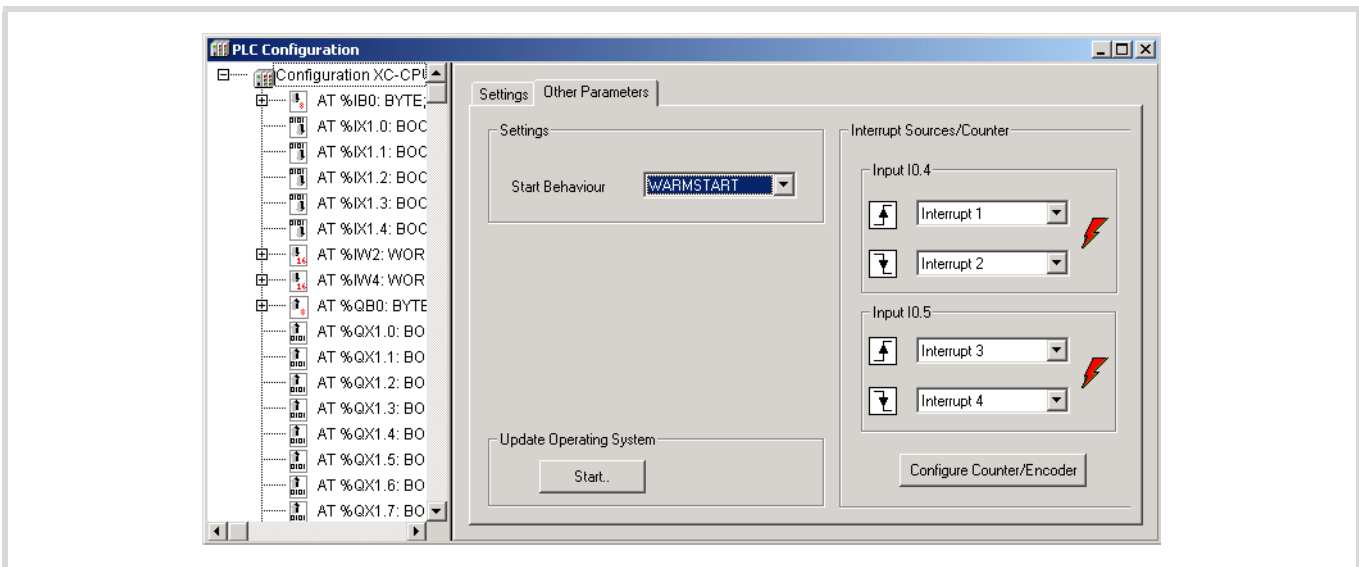


Figure 24: Updating the operating system (OS) of the XC200

The "Download Tool" window opens.

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

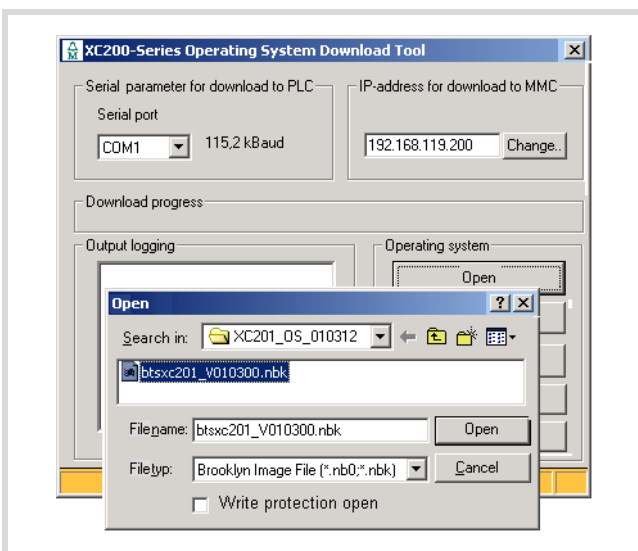


Figure 25: Operating system selection

- ▶ Opening of the operating system file to be transferred.

The following window appears:

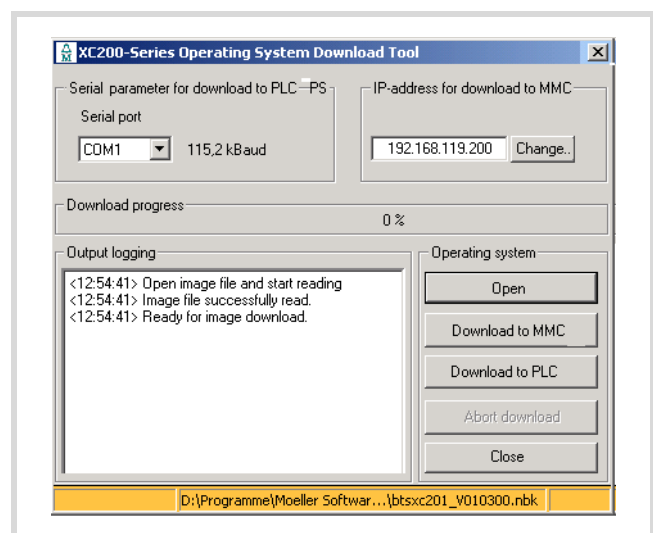


Figure 26: Download of the XC200 operating system

- ▶ Click on the "Download image" button.

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

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

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

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

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

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

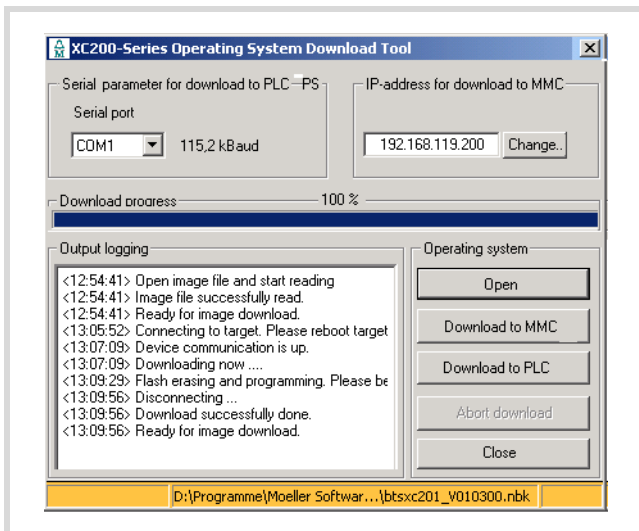


Figure 27: Download of the XC200 operating system ended

- ▶ End the download with the "Close" button.

Erasing the operating system from the Multimedia card

You can delete the operating system from the PC, e.g. with Internet Explorer:

- ▶ Establish a connection to the XC200 via the default address "ftp://192.168.119.200".
- ▶ Navigate to the folder "disk_mmc\moeller\XC-CPU201".

In this directory all the operating system files are stored and can be erased."

4 Program processing, multitasking and system times

Task configuration

The time relevant execution of an IEC program is designated with the word "task". The task is defined by a name, a priority and a type which defines under which conditions a task starts.

The conditions can be:

- cyclic
- event controlled
- system events.

Each task can be assigned with a range of programs which should be run during execution of the task. Priority and task conditions determine the sequence in which the tasks are to be processed.

The user program consists of multiple tasks of the same or differing priority, which are to be processed cyclically in a time interval which can be parameterized, or which is processed when an event occurs. 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. The tasks are processed in the order of their

priority. An output image is written to the physical outputs and the image of the inputs is read before every task call. 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 XSoft, Online changes etc. ...

All IEC tasks, even those with the highest priority are interrupted by a configured task when an event occurs. The necessary functions are available in the "XC201_Util.lib" library for direct addressing of the local inputs/outputs (without process image requirement).

Cyclic task

The following figure demonstrates the parametric programming of the task attributes. The task is initiated cyclically in accordance with the time entered for the interval.

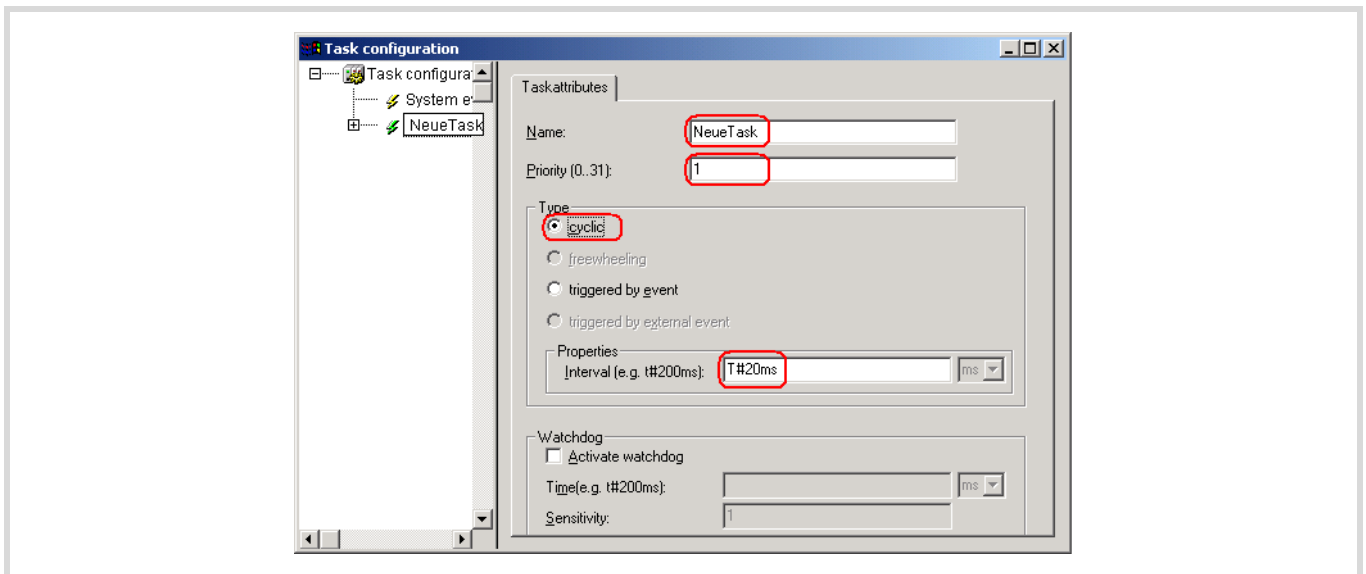


Figure 28: Cyclic task attribute

Event controlled task

The following figure demonstrates the parametric programming of the task attributes. The task is initiated as soon as the event entered for the variable receives a rising edge.

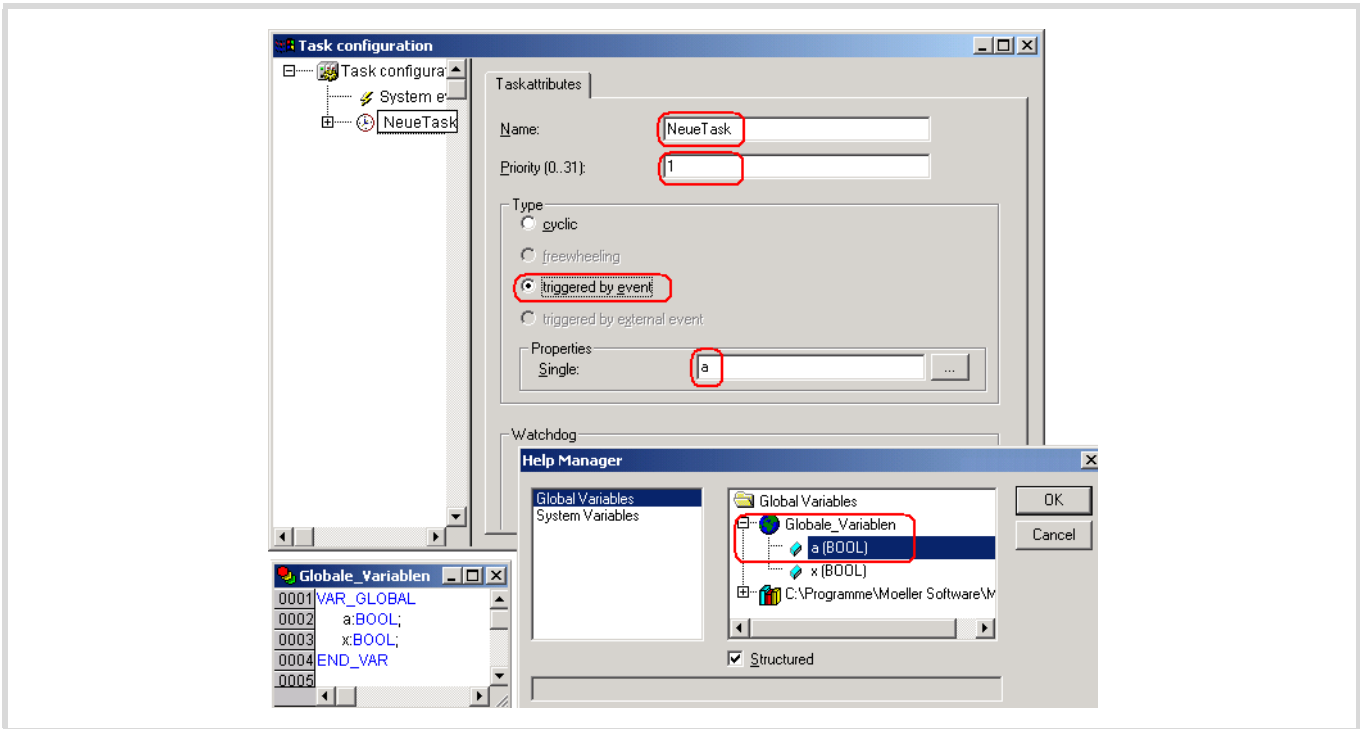


Figure 29: Event controlled task attributes

System events

The following figures indicate the programming/parametric programming of the system events. Not only can a task call up a project module for processing, a system event (event) can also call it up.

System events can be: Start, Stop, Interrupt

The XC200 offers physical interrupt inputs. These interrupt inputs are assigned to the inputs I0.4 and I0.5. Every interrupt source can be assigned to a program element to which a branch is made when an event occurs.

With every input you can select if the interrupt task is to be started by a rising and/or falling edge.

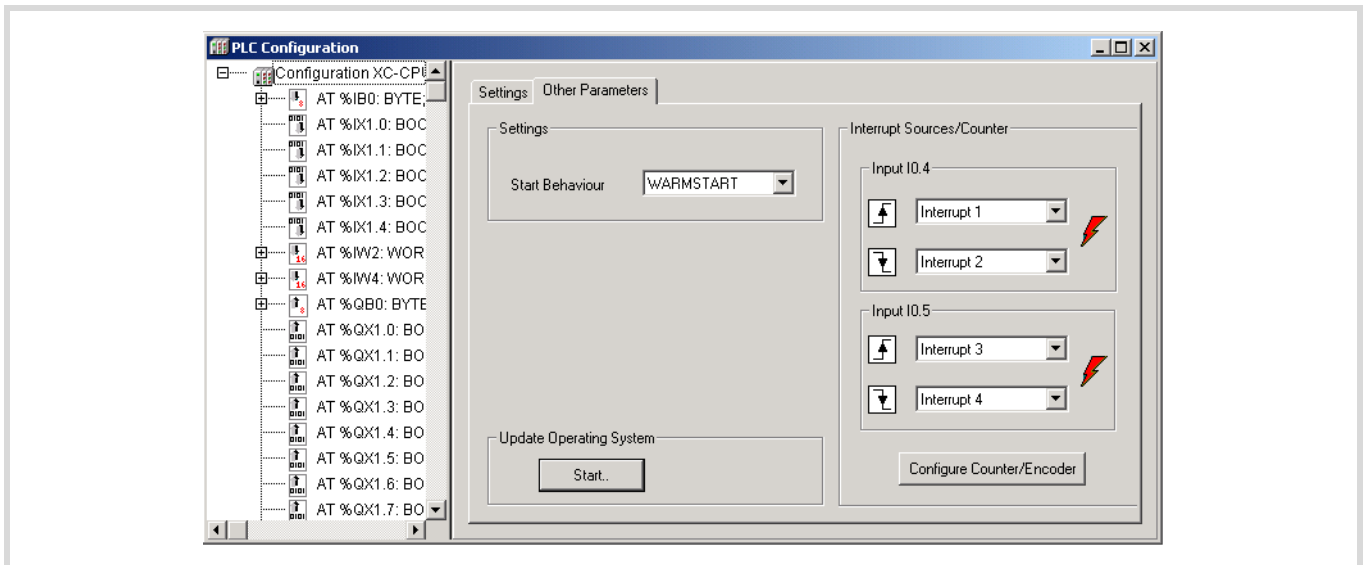


Figure 30: Assignment of the interrupt source with edge evaluation

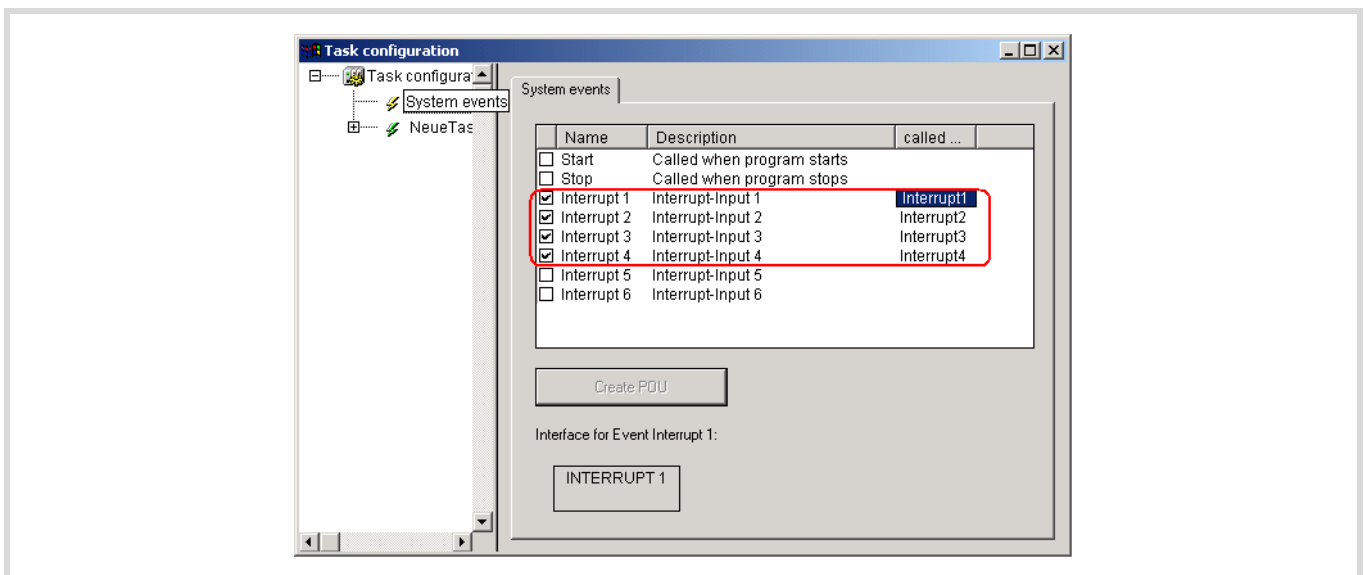


Figure 31: Assignment of interrupt source/POU (Interrupt task)

→ Further information concerning the cyclic and event controlled tasks as well as the system events can be found in the XSoft manual (AWB2700-1437GB) and in the online help of the "XSoft" programming system.

→ The sum of the time intervals of the tasks must be sufficiently smaller than the time interval of the watchdog.



Warning!

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

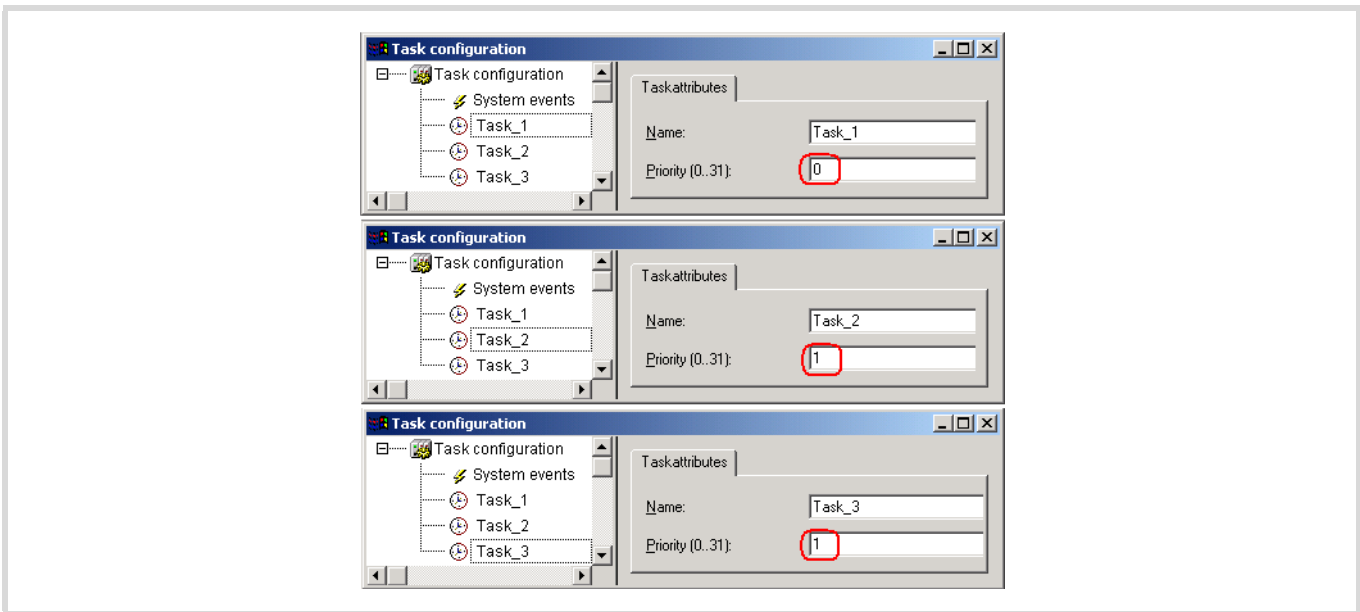


Figure 32: Task configuration with differing and equal priority

Multitasking

The XC200 run time system is a multitasking system. This means that multiple-tasks can be practically performed at the same time. As a consequence, the consistency of input/output values is only guaranteed within the task, which solely accesses the local inputs/outputs. If multiple tasks access the same inputs/outputs, consistency is only provided with the task having the highest priority and the shortest time interval.



Important

If multiple tasks access the locally configured signal modules, the consistency is only valid for the task with the highest priority and the shortest time interval.

The access to the inputs/outputs from multiple tasks only occurs with the respective task is called. Thereby it is possible, that differing signal states are detected or output during processing of a cycle for the same variable.

Access to physical outputs should be avoided in principle from multiple tasks, in order to assure a unique control sequence.

Avoid differing signal states during execution of a program: Create a task for the XIOC-I/O into which all inputs copy to global variables, and at the end of the interval, all outputs from global variables are written to the output modules (I/O update task). Within this task, all the I/Os are consistent as long as they are not used in another task. The global variables can then be used instead of the I/Os in other tasks.



A maximum of 10 tasks are allowed with the XC200 control system.

Parameterization of a task as "freerunning" is not supported.

Note with parametric programming of the watchdog time that the IEC interrupt service routines extend the task run times accordingly.

Behaviour of the CAN stack with multitasking

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



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

Task monitoring and watchdog

The user tasks are time monitored. The watchdog interrupts processing of the program when the user task exceeds a defined time and a defined frequency. In this case the outputs of the PLC are switched off and the application program is set to the HALT state. Afterwards, the user program must be reset with RESET.



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

Watchdog configuration

The following settings can be preselected in the XSoft for configuration of the watchdog:

- Watchdog on/off
- Watchdog time
- Watchdog sensitivity.

These settings apply for time controlled and event controlled tasks.

Watchdog active

The activated watchdog controls the timing of the program. The watchdog is started with the commencement of every processing cycle and ends after successful processing of the task to be completed. The time interval of the watchdog should be parameterized so that the sum of all task run times is sufficiently shorter than the parameterized watchdog time.

If the processing time is longer than the watchdog time, e.g. due to an endless loop in the user program, the watchdog will be activated. If the processing cycle is shorter than the watchdog time, the watchdog is not activated.

The triggering of the watchdog mechanism 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".

Furthermore, the watchdog is triggered with an endless loop when the execution time of the task takes longer than the result of the watchdog time \times watchdog sensitivity. This criterion provides a mechanism for recognition and reaction to endless loops with preselected watchdog functionality.

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

Example: Watchdog active

The interaction of interval time, task run time, watchdog time 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 task ⚡, as both task times are longer than the defined watchdog time and occur consecutively.

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

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

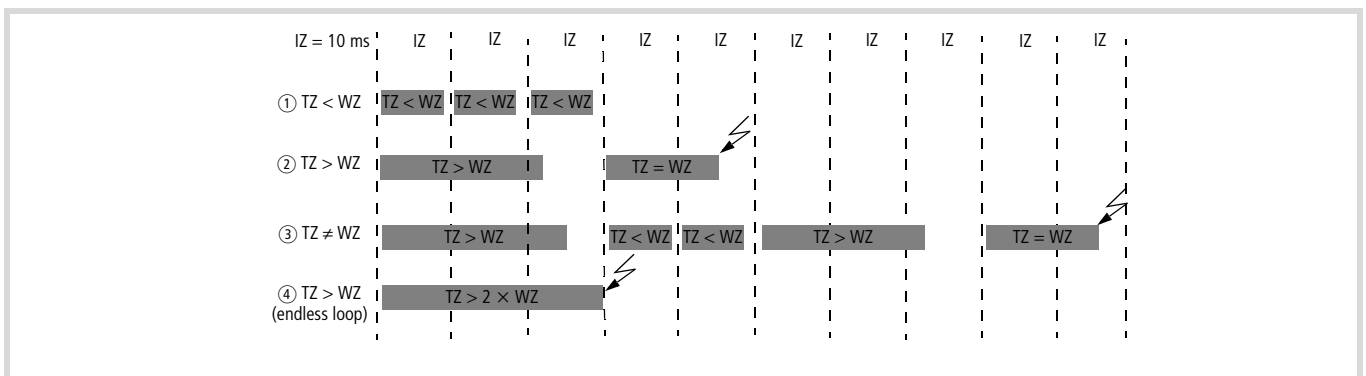


Figure 33: Watchdog active, multiple tasks with differing priority

Watchdog deactivated

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

Example: Watchdog deactivated

The interval time (IZ) is 10 ms.

Variant ①: The interval time (IZ) of a task has been set to 10 ms. The actual task run time (TZ) is 15 ms. The task was started during the initial call but only ended during 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.

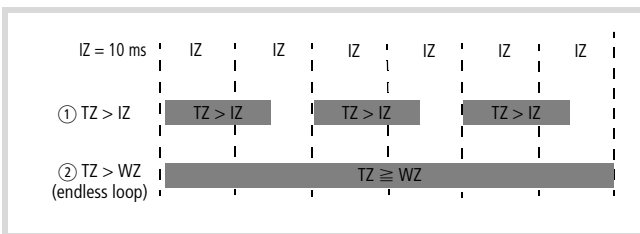


Figure 34: Watchdog deactivated, multiple tasks with differing priority

Multiple tasks with the same priority

It is also possible to create multiple tasks which have the same priority. The tasks with the same priority are executed in the sequence in which they have been programmed in the task configuration. If the task times are longer than the interval times, the tasks are split according to the "Time Slice" principle and are practically executed simultaneously as part intervals.

The following figure demonstrates processing of multiple tasks with the same priority.

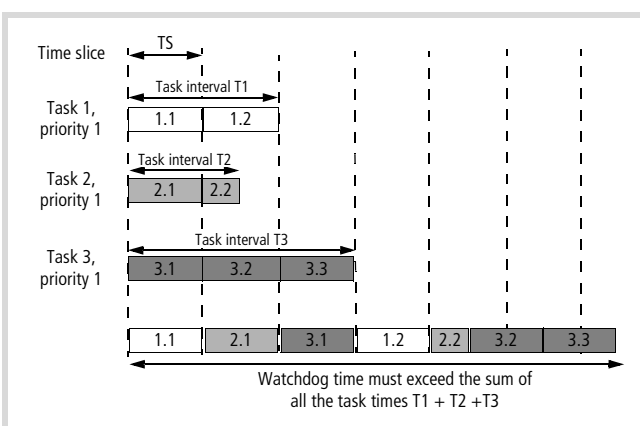


Figure 35: Multiple tasks with the same priority

System libraries, function blocks and functions

Various system libraries with the respective functions can be used for the application. Fundamentally, the standard "Standard-lib" library is available with the XSoft installation. The IEC modules and functions are contained in this library. Further libraries can be installed if required.

The description of the function blocks and functions can be found in the "XSoft" (AWB2700-1437GB) and in the Library/Online help of the "XSoft" programming system.

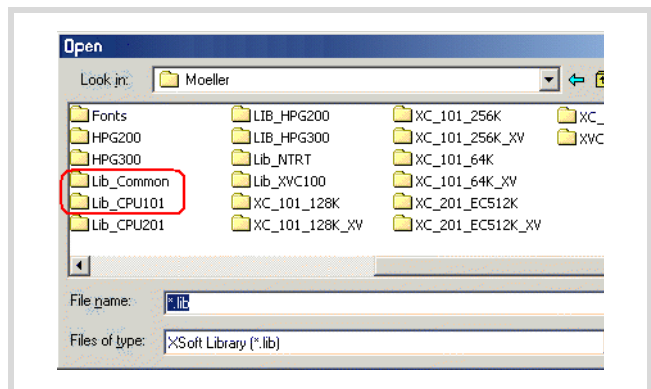


Figure 36: Overview of the libraries in the "Moeller" folder

"Lib_Common" library

→ In the "Lib_Common", further function blocks and functions are contained which are described in the "XSoft" (AWB2700-1437GB) and in the Library/Online help of the "XSoft" programming system.

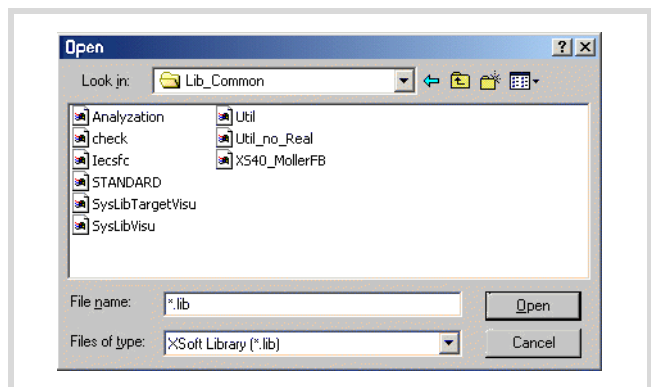


Figure 37: "Lib_Common"

"Lib_CPU201" library

The Lib_CPU201 library contains XC200 specific libraries. It assumes the XC-CPU201-EC....K-8DI-6DO (-XV) target system availability.

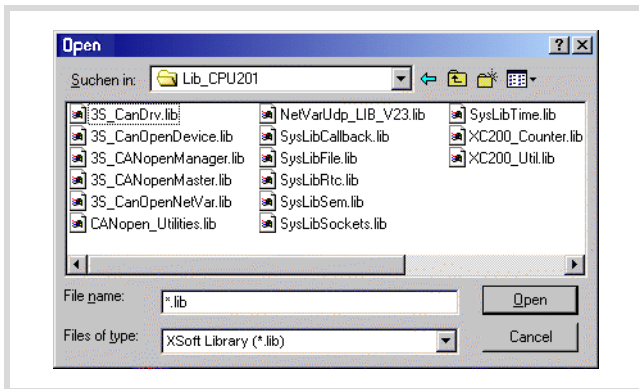


Figure 38: "Lib_CPU201"

Both the following libraries include the function blocks for scanning the incremental encoder input and the counter inputs as well as additional components and functions of the XC200:

- XC200_Counter.lib "Counter functions" and
- XC200_Util.lib "XC200 modules, functions and commands"

The modules and functions of the "XC200_Counter.lib" are described in chapter6 Configuration and parameterization of the inputs/outputs and "XC200_Util.lib" chapter7 XC200 specific functions.

XS40_MoellerFB.lib "library"

This library contains Moeller specific modules and functions which have been integrated into the XSoft from the SucoSoft S40. You can find the descriptions of "XS40_MoellerFB.lib" in the manual "Function Blocks for XSoft" (AWB2786-1456GB).

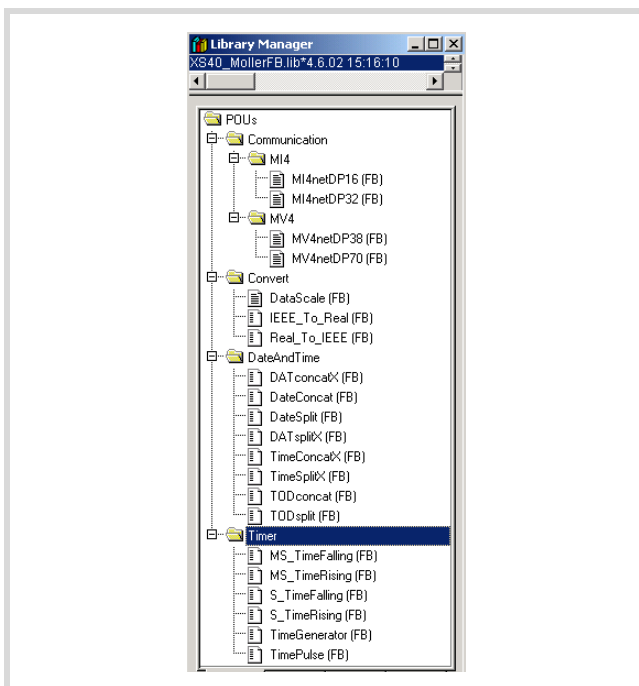


Figure 39: Overview of the "XS40_MoellerFB.lib" library

"SysLibFile.lib" library

The "SysLibFile" library allows you access to the file system of the XC200. 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 XSoft programming system under the "SysFile<Function>" search term.

Examples of the "SysFile..." functions

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

"w" mode

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

⚠ Important

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

"r" mode

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

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

"a" mode

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

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

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

Open in "r" mode

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

Open in "w" mode

```
OpenFile2 := SysFileOpen('\disk_mmc\M0ELLER\XC-CPU201-EC512k-8DI-6D0\Project \File2','w');
```

Open in "a" mode

```
OpenFile3 := SysFileOpen('\disk_usb\M0ELLER\XC-CPU201-EC512k-8DI-6D0\Project \File3','a');
```

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

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

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



Note!

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

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 are the input and output signals of the CPU and the centrally expanded XC200 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 parameters are indicated in 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

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

Programming of "Direct peripheral access" is described in section "Direct peripheral access" from page 34.

ReadBitDirect

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

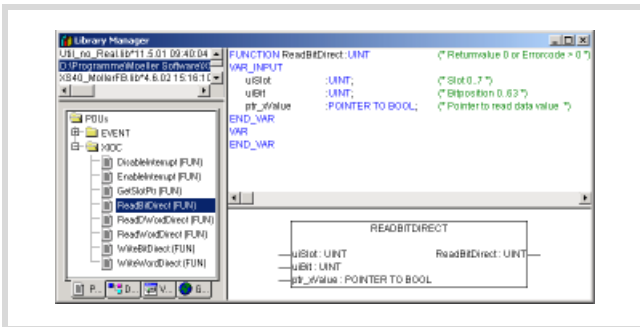


Figure 40: ReadBitDirect function

Parameters

uiSlot	Slot number of the signal module. For possible parameters see Table 8 on page 35
uiBit	Bit position within the input value of the signal module. For possible parameters see Table 8 on page 35
ptr_xValue	Pointer to the variable value
ReadBitDirect	Display of the fault code, see Table 9 on page 38

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.

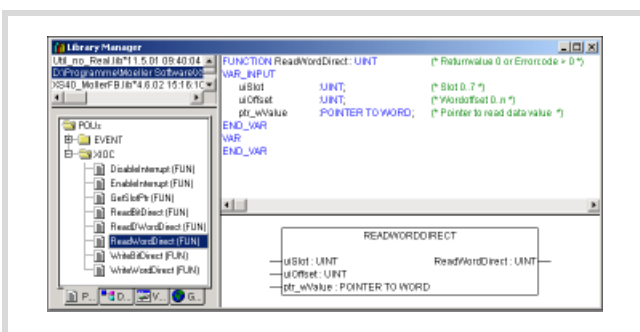


Figure 41: ReadWordDirect function

Parameters

uiSlot	Slot number of the signal module. For possible parameters see Table 8 on page 35
uiOffset	Word offset within a signal module. For possible parameters see Table 8 on page 35
ptr_wValue	Pointer to the variable value
ReadWordDirect	Display of the fault code, see Table 9 on page 38

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.

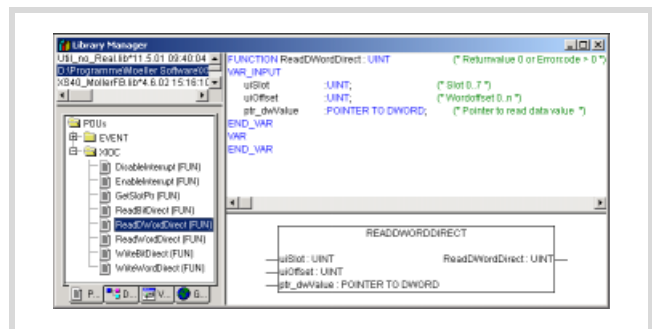


Figure 42: ReadDWordDirect function

Parameters

UiSlot	Slot number of the CPU/Signal module. For possible parameters see Table 8 on page 35
uiOffset	Word-offset within the CPU/Signal module. For possible parameters see Table 8 on page 35
ptr_dwValue	Pointer to the variable value
ReadDWordDirect	Display of the fault code, see Table 9 on page 38

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".

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

If you are working with the functions for directly writing an output bit or word in a task or an interrupt, ensure that the output bit or word in which the bit is contained is not declared or referenced in a task!

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 assigned to any other task!
The other bits of the output word may still be assigned in other tasks as the "Q-BOOL" output bit.
- If an output bit is modified for fast processing with the "WriteBitDirect" functions, and this bit is also processed at another location (task, event or interrupt), the "WriteBitDirect" function must be used at all locations (no Q-BOOL declaration and no referencing).

Example

Output variable declaration:

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

Referencing (assignment in the application):

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

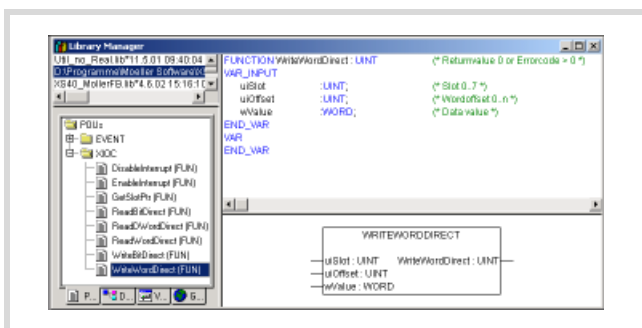


Figure 43: WriteBitDirect function

Parameters

uiSlot	Slot number of the signal module. For possible parameters see Table 8 on page 35
uiBit	Output bit within the signal module. For possible parameters see Table 8 on page 35
xValue	The pointer points to the variable in which the value for the output bit is located
WriteBitDirect	Display of the fault code, see Table 9 on page 38

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.

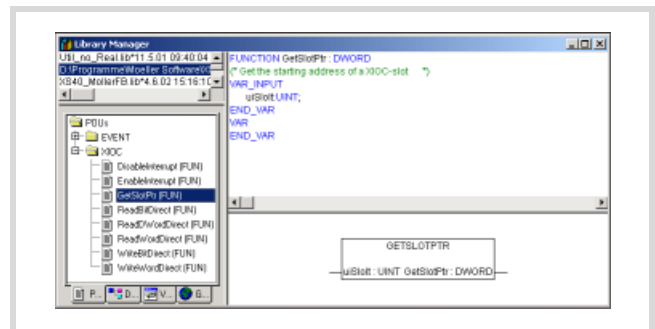


Figure 44: WriteWordDirect function

Parameters

uiSlot	Slot number of the signal module. For possible parameters see Table 8 on page 35
uiOffset	Output word within a signal module. For possible parameters see Table 8 on page 35
wValue	The pointer points to the variable in which the value for the output word is located
WriteWordDirect	Display of the fault code, see Table 9 on page 38

GetSlotPtr

→ This function is not available!

Error code with "direct peripheral access"

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

The following return values are possible:

Table 9: Error codes with direct peripheral access

IO_ACCESS_NO_ERROR:	No error
IO_ACCESS_INVALIDE_SLOTNUMBER	Slot = 0 or greater than 15
IO_ACCESS_INVALIDE_OFFSET	BitWord offset is too large
IO_ACCESS_DENIED	Invalid access, e.g. write access to input module, read access to output module or access to non-available address range (offset too large)
IO_ACCESS_NO_MODULE	No module available at the parameterized slot

IO_ACCESS_INVALIDE_Buffer	No or incorrect pointer to the output variables
IO_ACCESS_INVALIDE_Value	Event is not "0" or "1" with "WriteBitDirect"

Data remanence

The controller has a memory area for remnant data → page 12. The variables declared with "VAR_RETAIN" are saved in this area and are thus retentive during a warm start of the application program (Caution: This does not apply for I, Q and M variables!). Data that are remnant for a cold start – "VAR_RETAIN Persistent" are also supported. The data remanence also functions if the controller is switched off.

If it was not possible to finish a cycle that was being processed, because of a supply interruption, then the data will not be consistent, since the interruption could have occurred at any point of the cycle. When the voltage supply recovers, the residual-cycle will not be completed. Instead, the controller will carry out a warm start. If this is to be prevented, appropriate measures must be taken as part of the project engineering. One solution for this problem is an uninterruptible power supply with additional accumulator buffering.

→ Physical operands such as I, Q, M cannot be declared as RETAIN variables.

Table 10: Data remanence overview

Behaviour after voltage recovery and RUN/STOP operating mode selector switch in RUN with software preselection:	Not retentive	Retain	Persistent	Retain Persistent
COLDSTART	Activation of the initial values			Existing values are retained
WARMSTART	Activation of the initial values	Existing values are retained	Activation of the initial values	Existing values are retained
Full reset ¹		Activation of the initial values		
Download			Existing values are retained	Existing values are retained
Start/Stop/Start...	Existing values are retained	Existing values are retained		

1) After "Full reset" a program download must occur. The program is started with COLDSTART or WARMSTART. With WARMSTART the initial values are activated.

Operating states

The following summary provides you with the state definitions for the XC200. The LED indications for the various states are also shown.

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

State	Display		Definition
	RUN/STOP	SF	
Boot	Off (flashes 1 × 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 1 × at Start)	OFF	Windows CE system start and system test are carried out. Start of applications <ul style="list-style-type: none"> • HTTP-Server • FTP-Server • Telnet-Server • PLC-Runtime • Webserver
PLC-STOP	Flashes	OFF	PLC in "STOP" state
PLC-RUN	ON	OFF	PLC in "RUN" state
PLC-STOP/RUN with diagnostics – Error message (Error list)	Flashes/on	ON	The occurrence of a fault will be protocolled in the "Error List". (Read out with browser : "geterrorlist") occurs as a critical fault in RUN state "Cycle time exceeded". The system is set to the Halt state. A "Warm Reset" command is automatically executed.
NOTREADY (STOP with critical fault; no RUN state possible)	Flashes	ON	No start possible. A major fault prevents a start (see "Error List") e.g.: <ul style="list-style-type: none"> • No program loaded • Fieldbus error • Configuration error • Checksum error • ...
ShutDown	Flashes	Flashes	Wait for the shut down of the supply voltage (after Browser command: shutdown)

Web visualization

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

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

<http://192.168.119.200:8080/webvisu.htm>

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

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

Remote services

See the respective browser commands on page 61.

CANopen bus expansion

The description of the CANopen bus extension can be found in the following application notes:

- XC...-XION (AN2700K18GB),
- XC...-XC using network variables (AN2700K19GB),
- Coupling multiple autonomous controls (CAN-Device) via CANopen (AN2700KGB) and
- Engineering of CAN stations(AN2700K27GB).

The following baud rates are possible: 20000; 50000; 100000; 125000; 250000; 500000; 800000; 1000000 Bits/s.

Limit values for memory usage

The memory of the XC200 is divided into memory segments for data. The maximum memory volume is dependent on the segment size and the number of segments. The segment size is defined as a fixed value and cannot be modified, the number of segments is variable.

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

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

The number of segments is set to 1 by default.

The number of segments is changed as follows:

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

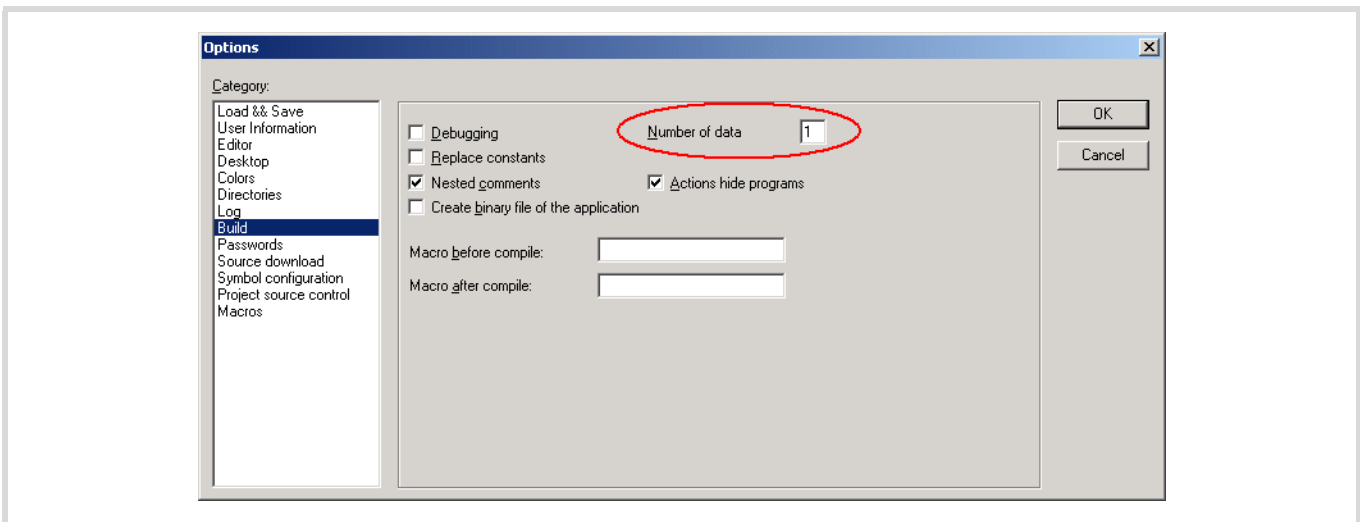


Figure 45: Memory management: Change the number of data segments

Download of programs

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

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

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

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

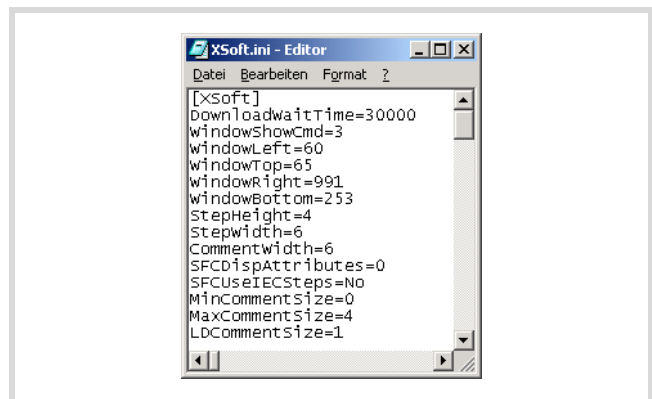


Figure 46: Addition of the DownloadWaitTime

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

The transfer time is entered in ms. A time of 30000 ms has been entered here for example.

- ➔ Ensure that the "Retain" variables are entered with a program download, the "Persistent" variables are retained.

5 PC – XC200 connection set-up

This section describes the measures that are required to link a PC to the XC200, so that the PC can be used as a programming device (hardware and software).

Establishing a connection via the RS232 interface (XC200)

Communication is implemented via the serial RS232 interface. You can use either the COM1 or the COM2 port for the PC interface. Please use the programming cable XT-SUB-D/RJ 45 to make the physical connection.

Programming cable

The programming cable is made up as shown below:

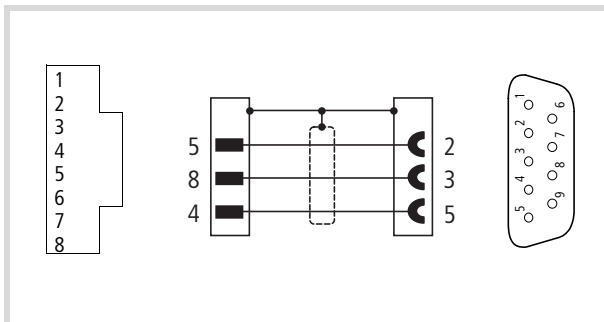


Figure 47: Core assignments for RS232 programming cable

Settings in the XSoft

Use XSoft to define the communication parameters:

- ▶ Call up the menu item <Online → Communication Parameters> in XSoft, and select the COM1 or COM2 interface.
- ▶ Enter the values as entered in Figure48.

You can alter the default values by making a double-click on the entered value.

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

From operating system version V01.03.xx 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.

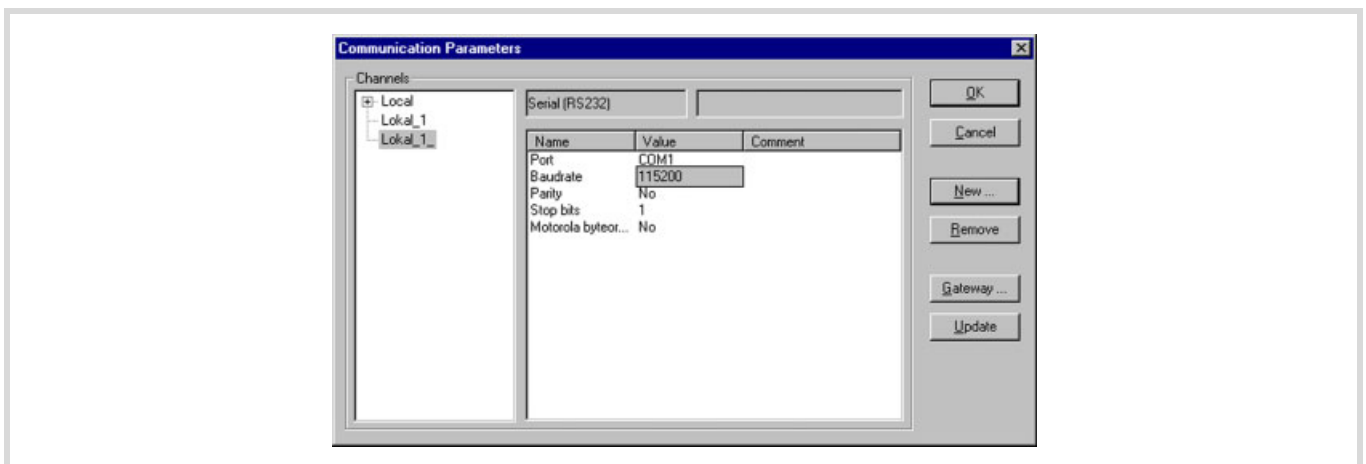


Figure 48: Setting communication parameters

Connection set-up via Ethernet

The programming device interface is designed so that it can be operated as an alternative to the Ethernet interface. An Ethernet interface card is required as a PC interface. For establishment of the direct physical connection, please use the generally commercially available Ethernet crossover cable (XT-CAT5-X-2 or XT-CAT5-X-5).

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

Assignment of the Ethernet interface

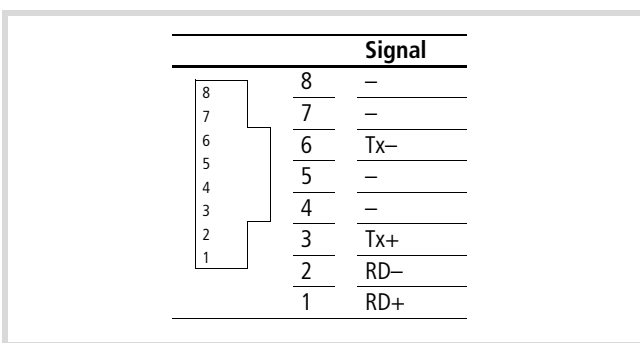


Figure 49: Assignment of the Ethernet interface

➔ A download of the operating system is not possible using the Ethernet.

Settings in the XSoft

Use XSoft to define the communication parameters:

- ▶ In the XSoft, call up the (Online → Communication Parameters) and select the "Tcp/Ip (Level 2 Route)" device with the "New" parameter. The name of the new channel could be "Ethernet-XC200-Test" to name an example.

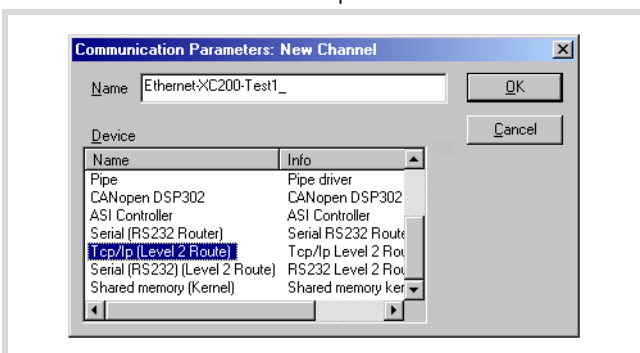


Figure 50: Communication parameters for Ethernet connection

- ▶ Confirm with OK.

The following window opens:

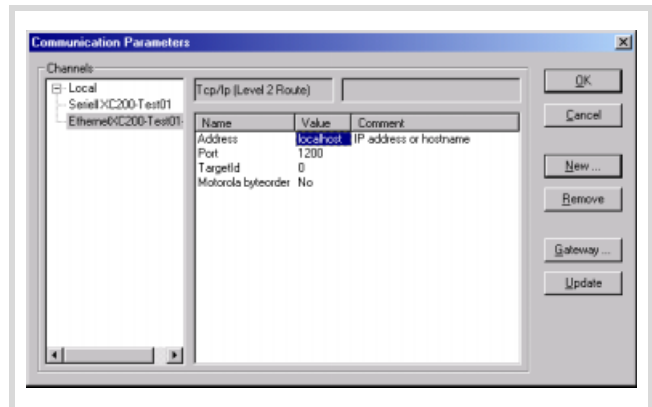


Figure 51: XC200 IP address

- ▶ Select the "localhost" field and enter the default Ethernet address "192.168.119.200".
- ▶ Confirm with OK; another field such as the field below it with the figure "1200" must be selected for this purpose.
- ▶ Generally, the IP mark: 255.255.255.0 is to be selected.

The following window appears:

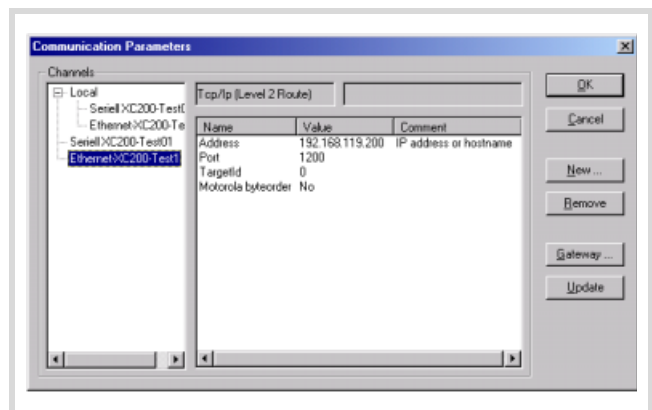


Figure 52: Communication parameters with IP address

- ▶ Save your program with the new communication parameters, compile the program and logon to the control.

Restart the XC200 before you operate the Ethernet connection. The DHCP function (Dynamic Host Configuration Protocol) is not activated.

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

Example 1

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

Example 2

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

The following conditions apply in example 1 and 2:

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

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

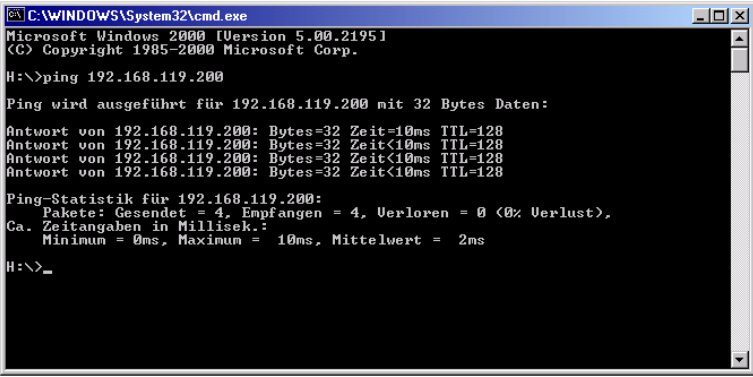
- ▶ Open the DOS window via the "Start" field and the "Run" command.
- ▶ Enter "CMD" in the white field and confirm with "OK".

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

- ▶ For the example mentioned you would enter the following text: "ping 192.168.119.200" and confirm this with "OK".

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

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



```

C:\WINDOWS\System32\cmd.exe
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
H:\>ping 192.168.119.200

Ping wird ausgeführt für 192.168.119.200 mit 32 Bytes Daten:

Antwort von 192.168.119.200: Bytes=32 Zeit=10ms TTL=128
Antwort von 192.168.119.200: Bytes=32 Zeit<10ms TTL=128
Antwort von 192.168.119.200: Bytes=32 Zeit<10ms TTL=128
Antwort von 192.168.119.200: Bytes=32 Zeit<10ms TTL=128

Ping-Statistik für 192.168.119.200:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0 (0% Verlust),
    Ca. Zeitangaben in Millisek.:
        Minimum = 0ms, Maximum = 10ms, Mittelwert = 2ms
H:\>_

```

Figure 53:PING answer with a correct Ethernet connection

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 61.

After you have changed the IP address, a reboot must be initiated.

6 Configuration and parameterization of the inputs/outputs

Input/output general

The hardware necessary for the application and the physical inputs and outputs are determined (configured) in the PLC configurator.

At the start of the CPU (change of the operating mode from HALT to RUN) the CPU verifies the conformance of the configured signal modules with their physical availability. The CPU switches to RUN if verification is positive. If verification detects non-conformance or if a configured module is not present, the PLC will stay in the HALT mode.

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

Note!

- 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 cant change over to the RUN mode.
- Updating the status indication of the input values in the configuration occurs only in the physical inputs used for the program.

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

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

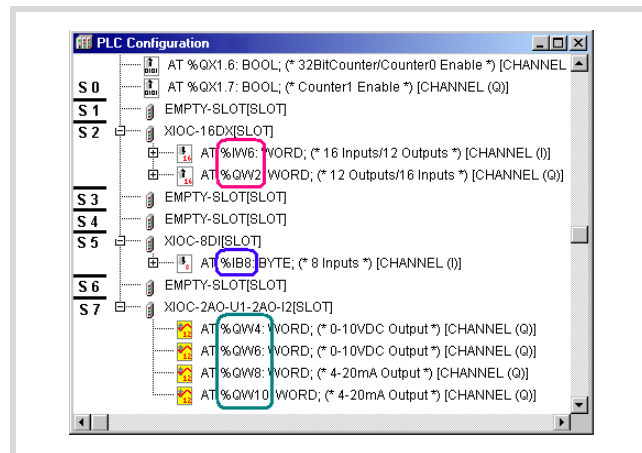


Figure 54: Current configuration

*) S0, S1, ..., S7 = slot or slot number on backplane

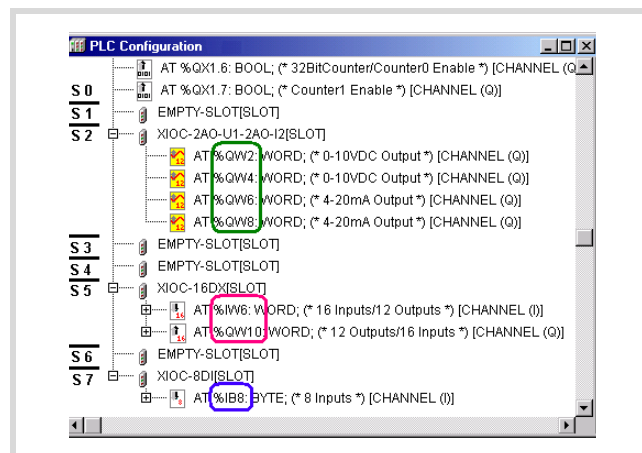


Figure 55: Configuration change by exchanging the modules

*) S0, S1, ..., S7 = slot or slot number on backplane

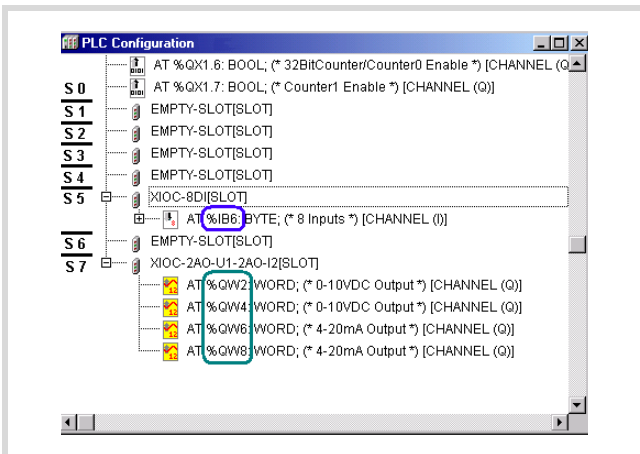


Figure 56: Configuration change by removing a module

*) S0, S1, ..., S7 = slot or slot number on backplane

Figures 54 to 56 indicate the changes to the input/output parameters of the signal modules in dependence on the slots and are compiled in Table 12.

Table 12: Input/output parameters with a change of configuration

Figure	Slot	Module type	Input parameter	Output parameter
54	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
55	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	–
56	2	Slot not used	–	–
	5	XIOC-8DI	%IB 6	–
	7	XIOC-2AO-U1-2AO-I2	–	%QW 2 %QW 4 %QW 6 %QW 8

Incremental encoder

You can use the local inputs I0.0 to I0.3 as incremental encoder inputs. 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 "Inc-Encoder" in the window which opens and click on the "Apply" button.

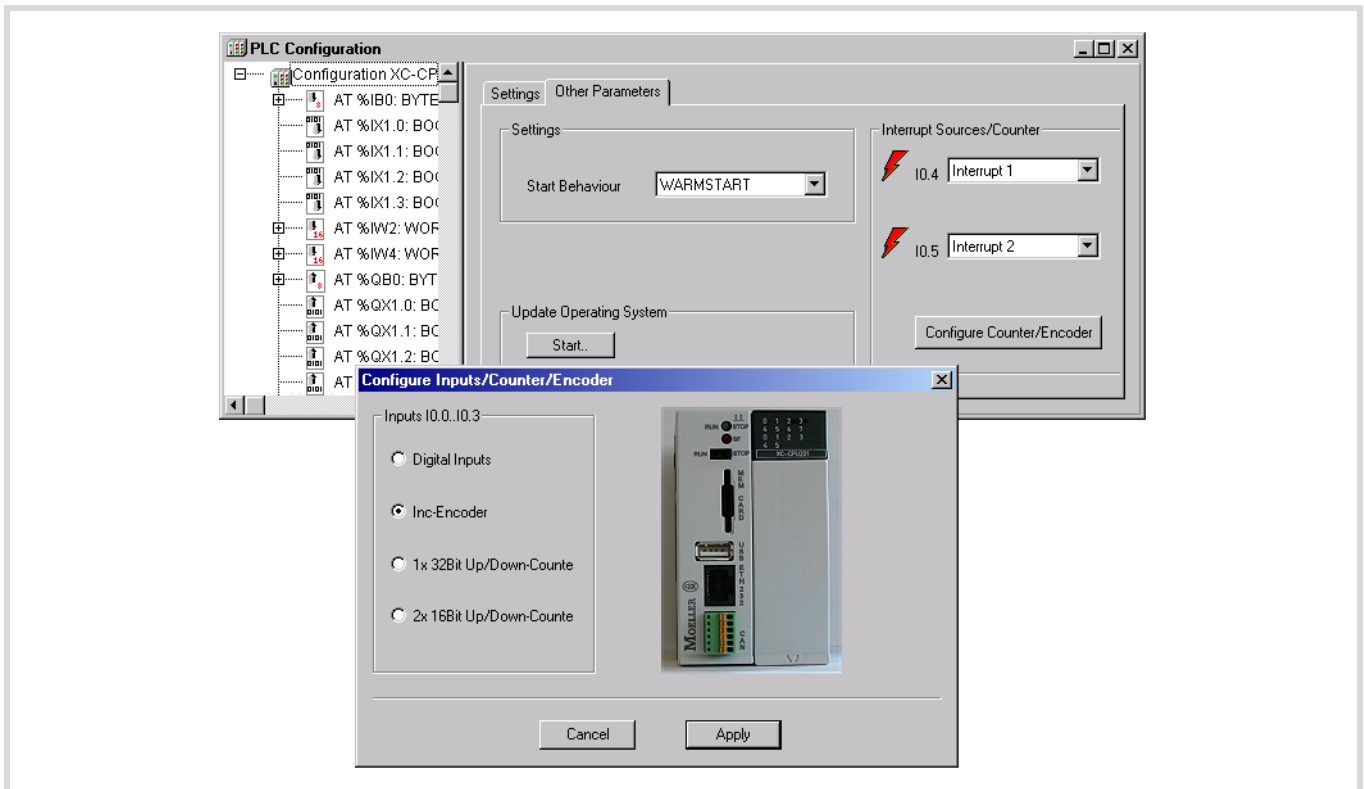


Figure 57: Incremental encoder preselection

Another window opens for the configuration.

- Enter the "Reference Marks" and the "Reference Mode" here. Input an offset value as a reference value if necessary.

The internal hardware counter is set to the reference value if an L/H edge is present on the reference input of the module and one of the following conditions is also fulfilled:

- the reference end switch has a "H" potential (referencing activated) or
- a software reset (reference window) has been performed.

After the reference end switch has switched from the "L" to "H" potential (hardware) or the bit "reference window" has been set (software), referencing occurs once or permanently with one/multiple L/H edges on the reference input. Please enter under "Reference Value" in the following window:

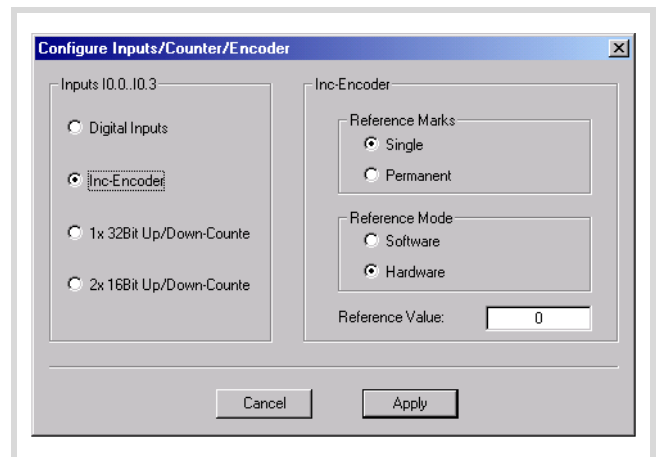


Figure 58: Incremental encoder parametric programming

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

Functionality

By scanning the inputs in the user program, you can query the information in the table:

Scan of the input (Bit, Byte, Word)	Information
AT %IB0: BYTE; (*Local Inputs/Counter*) [CHANNEL(I)]	Local input Byte 0
AT %IX0.0: BOOL; (*Bit0*)	Incremental encoder input track A
AT %IX0.1: BOOL; (*Bit1*)	Incremental encoder input track B
AT %IX0.2: BOOL; (*Bit2*)	Incremental encoder input reference track
AT %IX0.3: BOOL; (*Bit3*)	Digital input reference window
AT %IX0.4: BOOL; (*Bit4*)	
AT %IX0.5: BOOL; (*Bit5*)	
AT %IX0.6: BOOL; (*Bit6*)	
AT %IX0.7: BOOL; (*Bit7*)	
AT %IX1.0: BOOL; (*State*) [CHANNEL(I)]	H = homing implemented
AT %IX1.1: BOOL; (*NO*) [CHANNEL(I)]	L = no zero crossover; H = zero crossover
AT %IX1.2: BOOL; (*N1*) [CHANNEL(I)]	
AT %IX1.3: BOOL; (*Error*) [CHANNEL(I)]	L = no fault H = internal fault (both edges simultaneously)
AT %IW2: WORD; (*Counter-Value Low-Word*) [CHANNEL(I)]	Incremental encoder input Low-Word counter state
AT %IW4: WORD; (*Counter-Value High-Word*) [CHANNEL(I)]	Incremental encoder input High-Word counter state
AT %QB0: BYTE; (*Local Outputs*) [CHANNEL(Q)]	Local output Byte 0
AT %QX1.0: BOOL; (*Reference Window*) [CHANNEL(Q)]	Ref. signal with software reference
AT %QX1.1: BOOL; (*Reset Counter0*) [CHANNEL(Q)]	Reset to reference value
AT %QX1.2: BOOL; (*Reset Counter1*) [CHANNEL(Q)]	
AT %QX1.3: BOOL; (*NO Quit*) [CHANNEL(Q)]	Acknowledge 0-crossover for incremental encoder
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 = no counter function H = enable counter function
AT %QX1.7: BOOL; (*Counter1 Enable*) [CHANNEL(Q)]	

Counter input

You can use the local inputs I0.0 to I0.3 as counter inputs also. 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.

- ▶ Enter the "Interrupt Source" and the "Setpoint Value" at which the interrupt is to be initiated.

Parameterization of the interrupt source

If you parameterize an interrupt source, it leads to the following described functions. The count direction should be observed:

Count direction "up": If a setpoint value is achieved, the parameterized interrupt is activated. When the next count pulse occurs, the counter begins to count at "0" and then increments.

Count direction "down": If the counter value "0" is achieved, the parameterized interrupt is activated. When the next count pulse occurs, the counter begins to count at the "setpoint value" and decrements.

With the activated interrupt, it is possible to branch for example to the IEC interrupt program routines.

Setpoint value

An interrupt is activated when the setpoint is reached. This interrupt can be processed further in the user program.

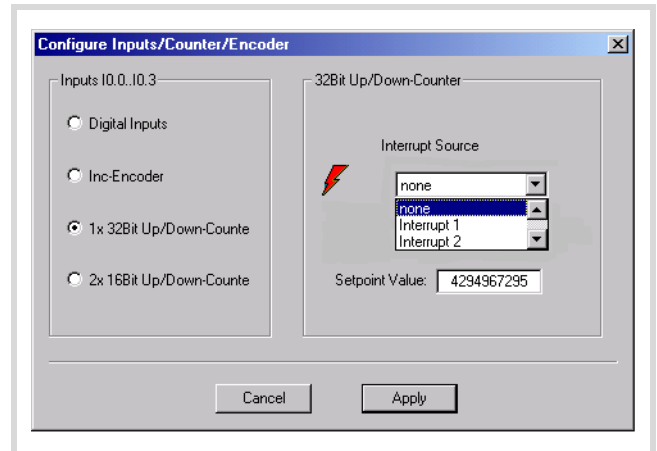


Figure 59: Parameterization of counter input 1 × 32 Bit

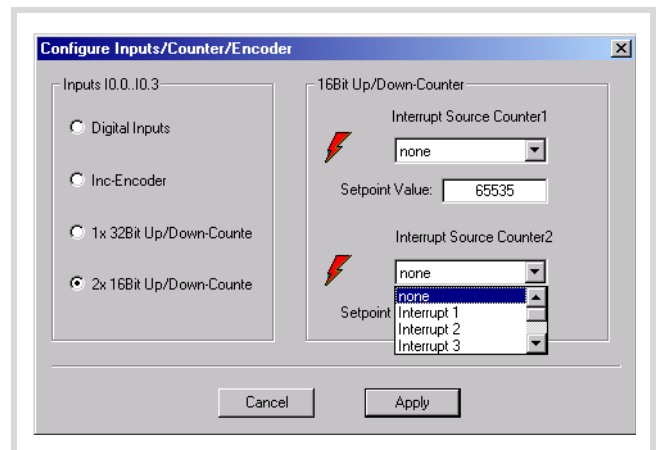


Figure 60: Parameterization of counter input 2 × 16 Bit

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

Functionality

By scanning the inputs in the user program, you can query the information in the table:

32 Bit counter

Scan of the input (Bit, Byte, Word)	Information
AT %IB0: Byte; (*Local Inputs/Counter*) [CHANNEL(I)]	Local input Byte 0
AT %IX0.0: BOOL; (*Bit0*)	Counter input for 32 Bit data capacity
AT %IX0.1: BOOL; (*Bit1*)	Counter input for 32 Bit count direction
AT %IX0.2: BOOL; (*Bit2*)	
AT %IX0.3: BOOL; (*Bit3*)	
AT %IX0.4: BOOL; (*Bit4*)	
AT %IX0.5: BOOL; (*Bit5*)	
AT %IX0.6: BOOL; (*Bit6*)	
AT %IX0.7: BOOL; (*Bit7*)	
AT %IX1.0: BOOL; (*State*) [CHANNEL(I)]	
AT %IX1.1: BOOL; (*NO*) [CHANNEL(I)]	L = no zero crossover, H = 32 Bit zero crossover
AT %IX1.2: BOOL; (*N1*) [CHANNEL(I)]	
AT %IX1.3: BOOL; (*Error*) [CHANNEL(I)]	
AT %IW2: WORD; (*Counter-Value Low-Word*) [CHANNEL(I)]	Counter state of 32 Bit Low-Word counter
AT %IW4: WORD; (*Counter-Value High-Word*) [CHANNEL(I)]	Counter state of 32 Bit High-Word counter
AT %QB0: BYTE; (*Local Outputs*) [CHANNEL(Q)]	
AT %QX1.0: BOOL; (*Reference Window*) [CHANNEL(Q)]	
AT %QX1.1: BOOL; (*Reset Counter0*) [CHANNEL(Q)]	Reset to 0
AT %QX1.2: BOOL; (*Reset Counter1*) [CHANNEL(Q)]	
AT %QX1.3: BOOL; (*NO Quit*) [CHANNEL(Q)]	Acknowledge 0-crossover for 32 Bit counter
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 = no counter function, H = enable counter function
AT %QX1.7: BOOL; (*Counter1 Enable*) [CHANNEL(Q)]	

Counter 2 x 16 Bit

Scan of the input (Bit, Byte, Word)	Information
AT %IB0: Byte; (*Local Inputs/Counter*) [CHANNEL(I)]	Local input Byte 0
AT %IX0.0: BOOL; (*Bit0*)	Counter input 0 for 16 Bit data capacity
AT %IX0.1: BOOL; (*Bit1*)	Counter input 0 for 16 Bit data capacity count direction
AT %IX0.2: BOOL; (*Bit2*)	Counter input 1 for 16 Bit data capacity
AT %IX0.3: BOOL; (*Bit3*)	Counter input 1 for 16 Bit data capacity count direction
AT %IX0.4: BOOL; (*Bit4*)	
AT %IX0.5: BOOL; (*Bit5*)	
AT %IX0.6: BOOL; (*Bit6*)	
AT %IX0.7: BOOL; (*Bit7*)	
AT %IX1.0: BOOL; (*State*) [CHANNEL(I)]	
AT %IX1.1: BOOL; (*NO*) [CHANNEL(I)]	L = no zero crossover, H = counter 0, 16 Bit zero crossover
AT %IX1.2: BOOL; (*N1*) [CHANNEL(I)]	L = no zero crossover, H = counter 1, 16 Bit zero crossover
AT %IX1.3: BOOL; (*Error*) [CHANNEL(I)]	
AT %IW2: WORD; (*Counter-Value Low-Word*) [CHANNEL(I)]	Counter state, counter 0 16 Bit
AT %IW4: WORD; (*Counter-Value High-Word*) [CHANNEL(I)]	Counter state, counter 1 16 Bit
AT %QB0: BYTE; (*Local Outputs*) [CHANNEL(Q)]	
AT %QX1.0: BOOL; (*Reference Window*) [CHANNEL(Q)]	
AT %QX1.1: BOOL; (*Reset Counter0*) [CHANNEL(Q)]	Reset to 0 counter 0
AT %QX1.2: BOOL; (*Reset Counter1*) [CHANNEL(Q)]	Reset to 0 counter 1
AT %QX1.3: BOOL; (*NO Quit*) [CHANNEL(Q)]	Acknowledge 0-crossover for counter 0 16 bit
AT %QX1.4: BOOL; (*N1 Quit *) [CHANNEL(Q)]	Acknowledge 0-crossover for counter 1 16 bit
AT %QX1.5: BOOL; (*Error Quit*) [CHANNEL(Q)]	Error acknowledge
AT %QX1.6: BOOL; (*32BitCounter/Counter0 Enable*) [CHANNEL(Q)]	L = no counter function, H = enable counter 0 function
AT %QX1.7: BOOL; (*Counter1 Enable*) [CHANNEL(Q)]	L = no counter function, H = enable counter 1 function

Interrupt processing

In the XC200 it is possible to program and parameterize up to four interrupt events. Interrupts can be activated by:

- physical input I0.4
- physical input I0.5
- 32 Bit up/down counter
- 16 Bit up/down counter 1
- 16 Bit up/down counter 2

If an interrupt occurs, the runtime module 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 are supported, whose priority assignment with simultaneous occurrence is determined by an interrupt number (the lowest number has the highest priority). Up to four interrupt sources are used from the XC200-CPU.

IEC interrupts must be completely processes and cant be interrupted by a new interrupt. A new interrupt is only carried out after the current interrupt has ended.



Important

All the outputs controlled (H signals) up to this point remain active and cant be switched off.

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

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

User interrupts can be inhibited and re-enabled from the program. The "DisableInterrupt" and "EnableInterrupt" functions are provided for this purpose. A call parameter in the XSoft 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 XSoft.

DisableInterrupt

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

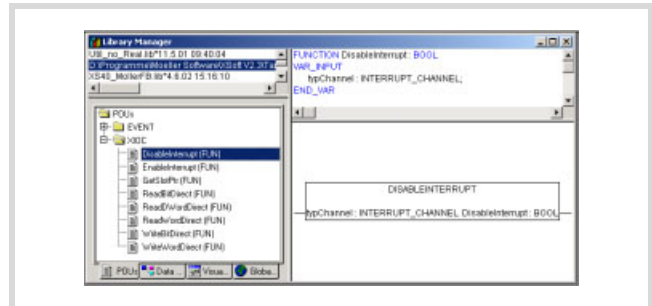


Figure 61: DisableInterrupt function

EnableInterrupt

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

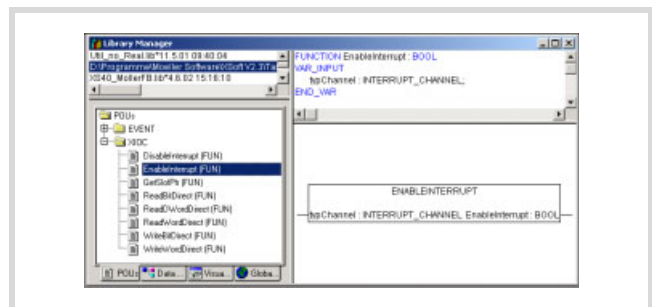


Figure 62: EnableInterrupt function

The formal procedure for the provision and integration of an interrupt function is described in individual steps in the following.

In the example, a H-signal on input I0.5 should branch into a programmed module and execute it.

- ▶ Create a program module with the component designation "Interrupt6" for this purpose.

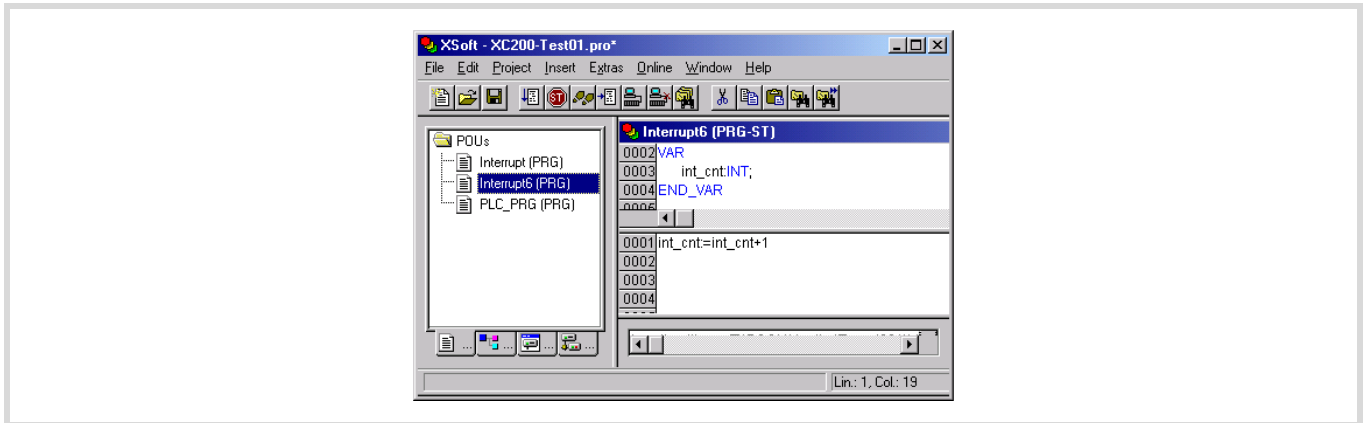


Figure 63: Interrupt module "Interrupt6"

- Changeover to the PLC Configuration and assign Interrupt 6 from the list field to input I0.5.

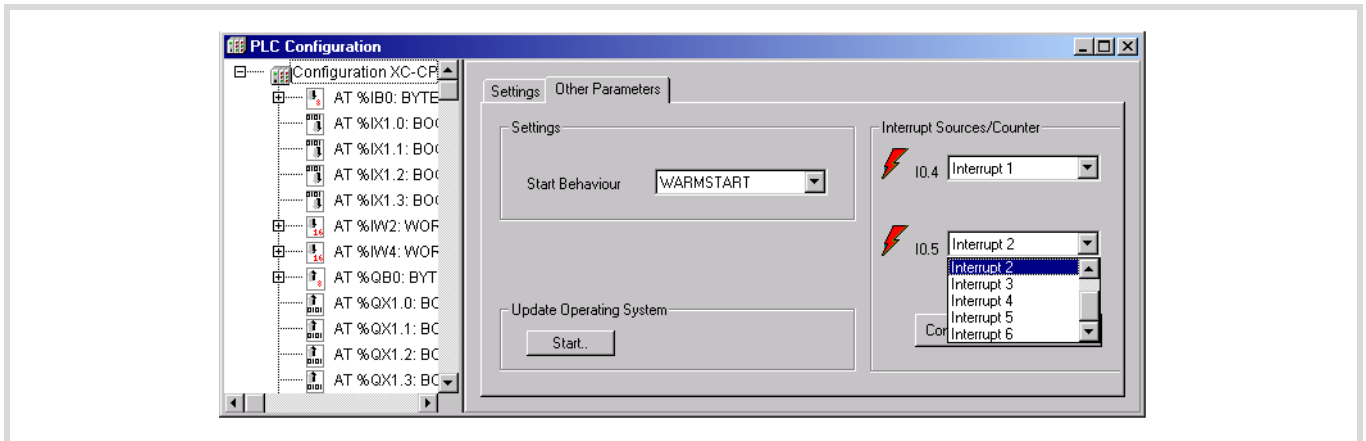


Figure 64: Assign input I0.5 with interrupt 6

- Changeover to the "Task configuration" and tick the box in the right hand field for "Interrupt6".
- Now stay on the same row and mark the "called POU" field with the left-hand mouse key and press function key "F2".

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

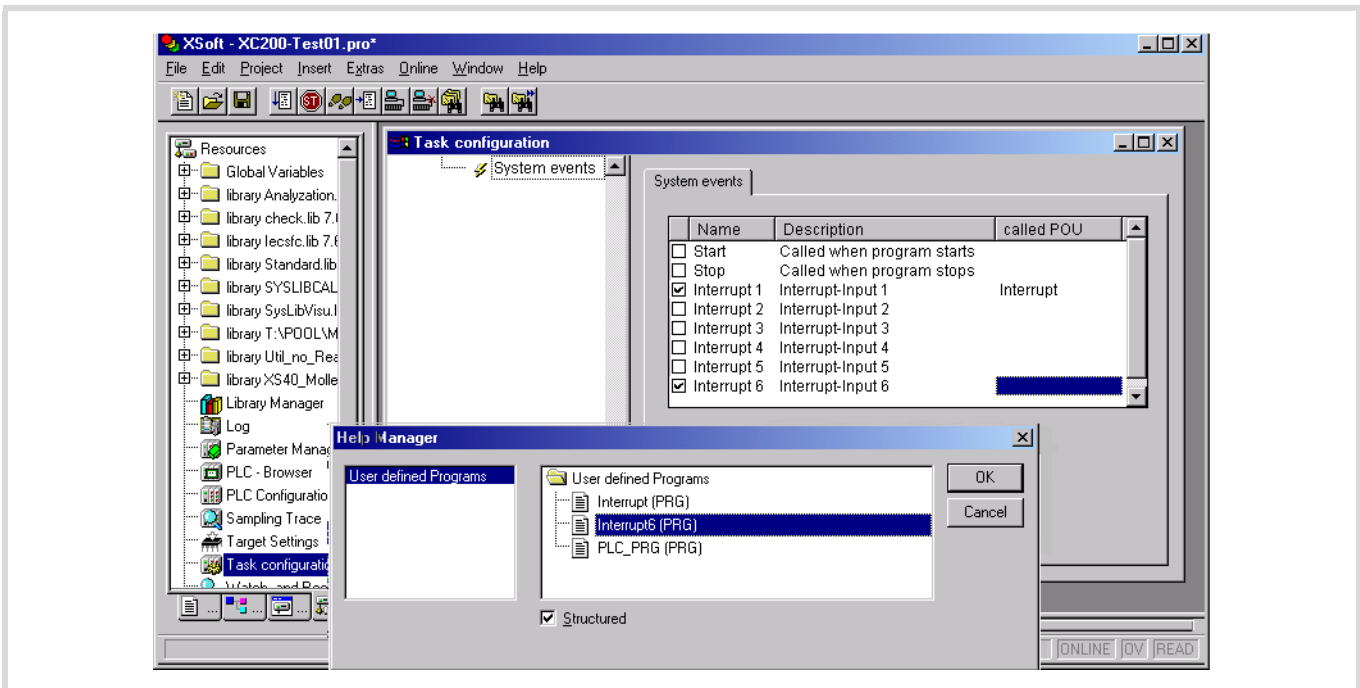


Figure 65:Assignment of an interrupt module to an interrupt event

► Select the "Interrupt6" program and confirm with OK.

The following window appears:

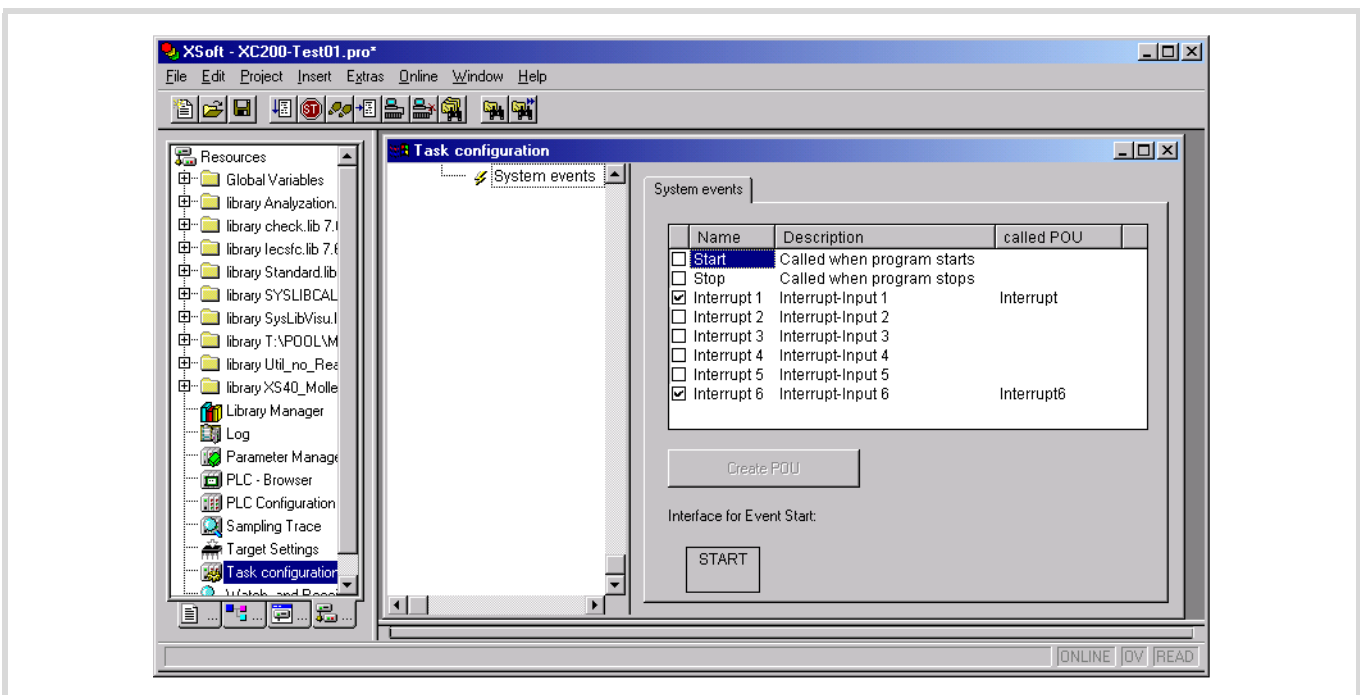


Figure 66:Interrupt module completed task configuration

► Save the program created, compile it and logon to the control and test the functions of the program modules which you have created.

7 XC200 specific functions

The XC200 specific functions are contained in the XC200_UTIL.lib library. From operating system version V01.03.xx the XC200_UTIL2.lib library with additional functions has been introduced. The additional functions are described from page 58.

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

- Event functions
- XIOC functions (direct peripheral access)
- CAN_Uilities.

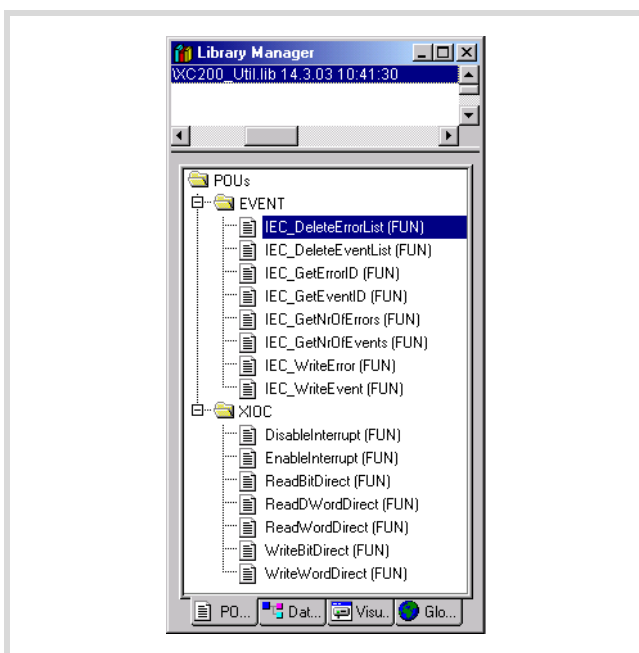


Figure 67: XC200 specific functions of the XC200-UTIL.lib library

The task of the event modules and the CAN utilities will be explained in the following. For the task of the XIOC modules see section "Direct peripheral access" on page 34.

Event functions

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

IEC_DeleteErrorList

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

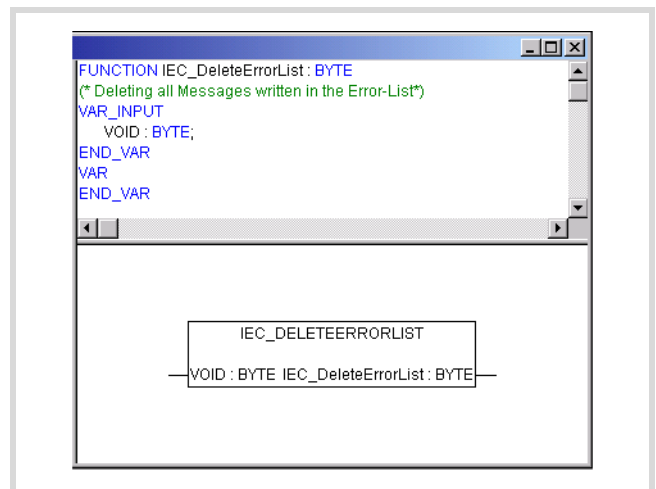


Figure 68: IEC_DeleteErrorList with declaration section function

IEC_DeleteEventList

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

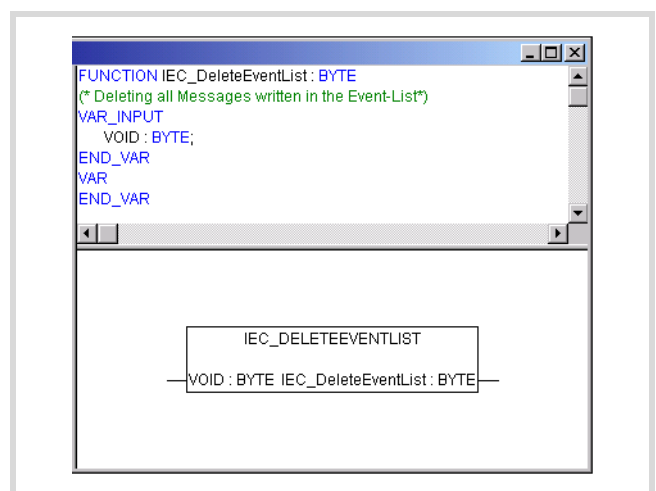


Figure 69: IEC_DeleteErrorList with declaration section function

IEC_GetErrorID

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

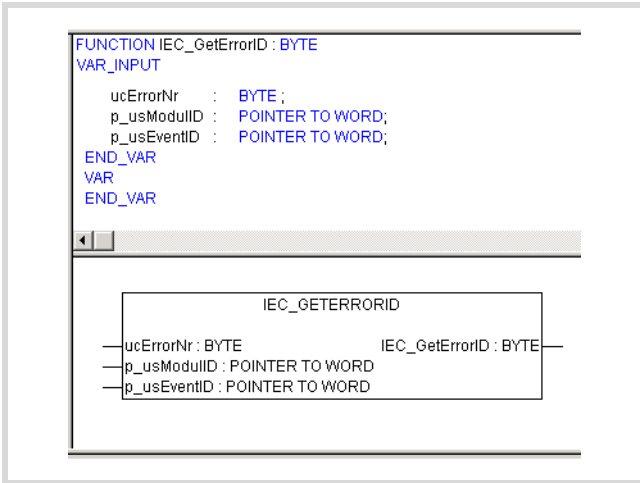


Figure 70: Function IEC_GetErrorID with declaration section

The description of the error messages and the error identity can be found in the Online documentation of the XSoft from Version V2.3 relating to function IEC_GetErrorID.

IEC_GetEventID

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

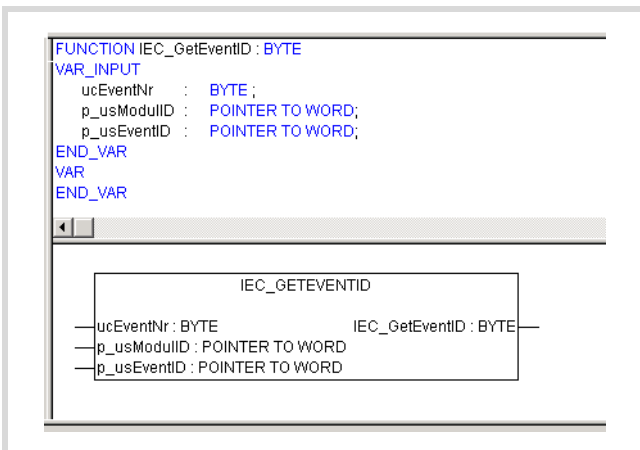


Figure 71: Function IEC_GetEventID with declaration section

The description of the event messages and the event identity can be found in the Online documentation of the XSoft from Version V2.3 relating to function IEC_GetErrorID.

IEC_GetNrOfErrors

This function returns the number of entered error messages.

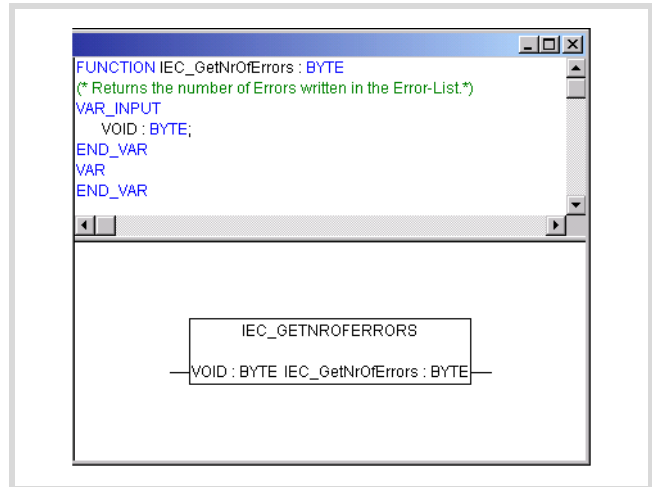


Figure 72: Function IEC_GetNrOfErrors

IEC_GetNrOfEvents

This function returns the number of entered event messages.

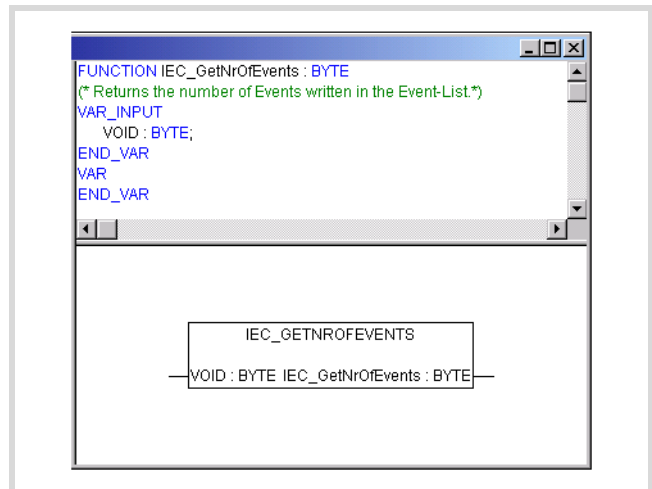


Figure 73: Function IEC_GetNrOfErrors

IEC_WriteError

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

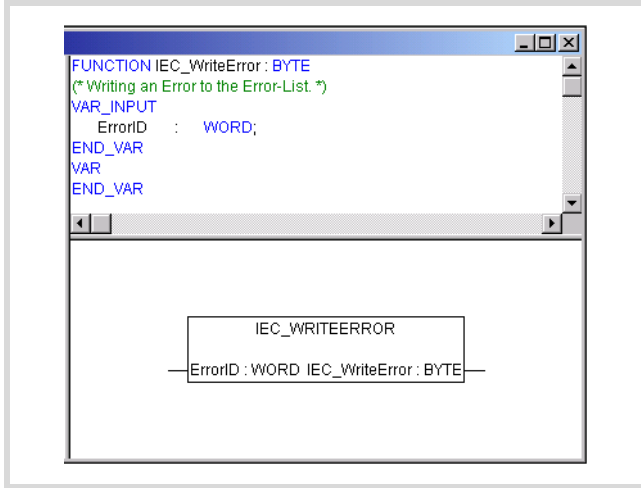


Figure 74: Function IEC_WriteError

IEC_WriteEvent

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

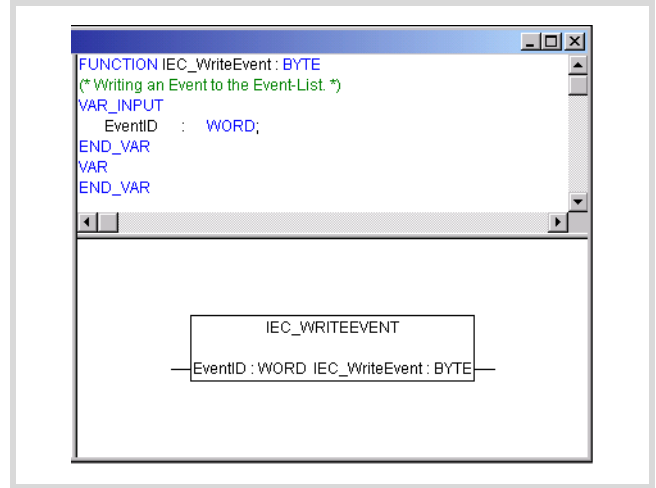


Figure 75: Function IEC_WriteEvent

CAN_Utilities.

CAN_BUSLOAD

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

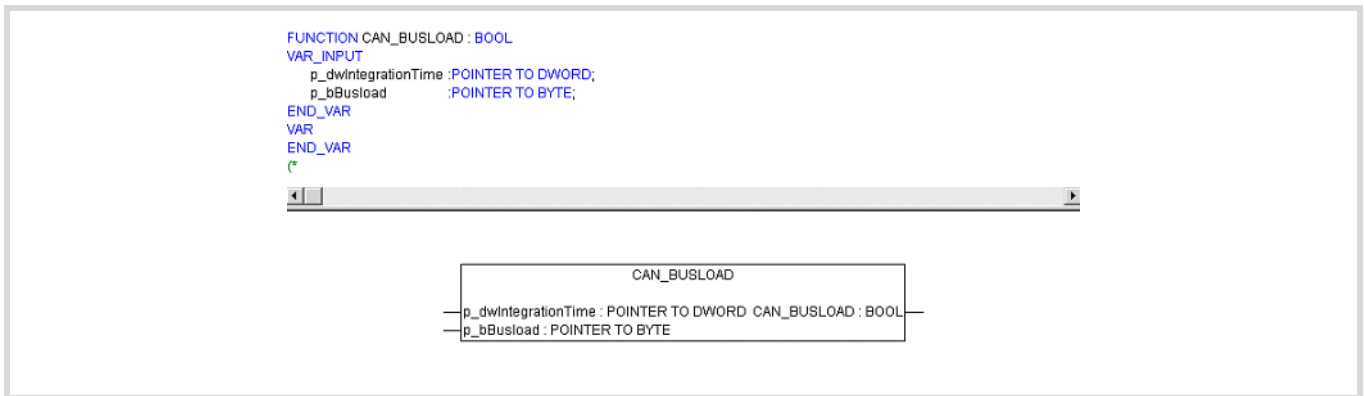


Figure 76: CAN_BUSLOAD function in the XC200_Util.lib

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

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

A read cycle is 500 ms long.

Information for evaluation of the CAN bus load can be found in section "Browser command "canload" for XC200" on page 64.

Additional functions of the XC200_UTIL2.lib

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

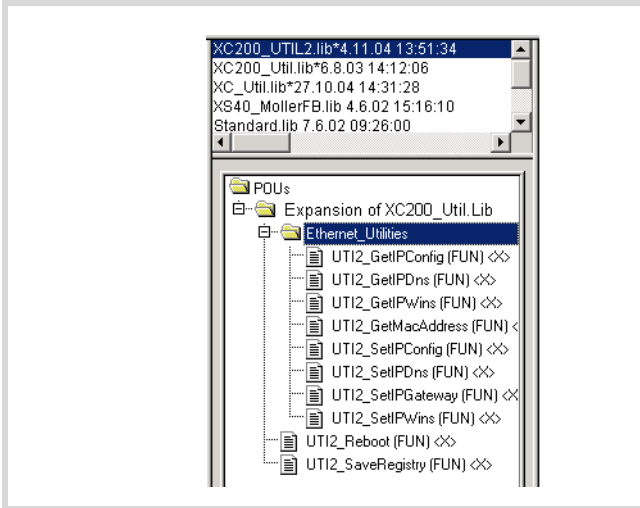


Figure 77: Overview of the XC200_UTIL2.lib

UTI2_GetIPConfig
Output of the IP address, subnetmask address and IPGateway address

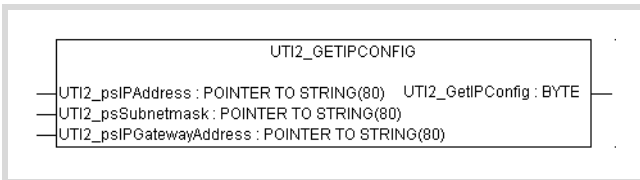


Figure 78: UTI2_GetIPConfig

Table 13: Input variables

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 subnetmask is written.
UTI2_psIPGatewayAddress:	Pointer to a string in which the read address of the standard gateway is written.

Table 14: Return value

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

UTI2_GetMacAddress
Output of the MAC address (MAC=Media Access Control)

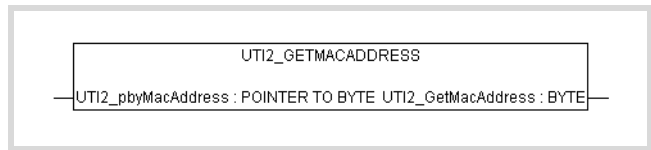


Figure 79: UTI2_GetMacAddress

Table 15: Input variables

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

Table 16: Return value

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

UTI2_SetIPConfig
Setting of the IP address and subnetmask address

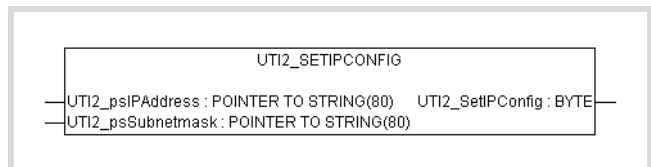


Figure 80: UTI2_SetIPConfig



Important

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 17: Input variables

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 18: Return value

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

UTI2_SetIPGateway
Setting of the IPGateway address

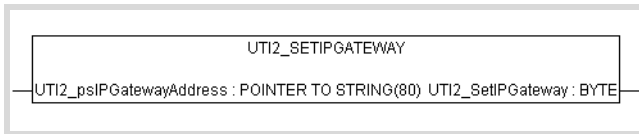


Figure 81: UTI2_SetIPGateway

Important
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 19: Input variables

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

Table 20: Return value

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

UTI2_Reboot
Restart with registry save

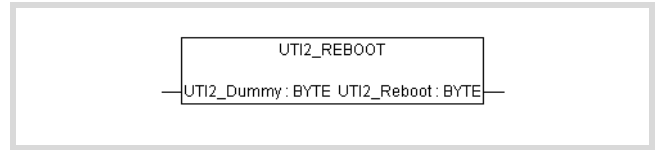


Figure 82: UTI2_Reboot

Table 21: Input variables

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

Return value

The value "1" is always returned.

UTI2_SaveRegistry
Saving of the registry

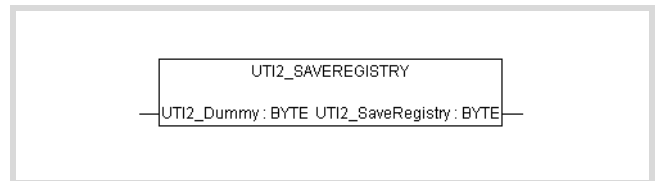


Figure 83: UTI2_SaveRegistry

Table 22: Input variables

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

Table 23: Return value

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

8 Browser commands

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

The commands available to the XC200 target platform are divided into two groups:

- Standard PLC 2.3 browser commands
- Target system specific PLC 2.3 browser commands

These commands are managed in a file and are implemented in the runtime module.

Table 24: Standard PLC 2.3 browser commands

Command	Description
?	Get a list of implemented commands.
delpwd	Erase password for online access
dpt	Output data pointer table
filecopy	Copy file
FileDelete	Erase file
filedir	Directory list [First folder in the list]
FileRename	Rename file
getprgprop	Read program information
getprgstat	Read program status
pid	Output project ID
pinf	Output project information
plcload	Display system performance: CPU usage
ppt	Output module pointer table
reflect	Mirror current command line for test purposes.
reload	Reload boot project again
resetprg	Reset user program
resetprgcold	User program cold reset
resetprgorg	Reset user program to original state
restoreretain	Restore retentive data from file [File name]"
saveretain	Save retentive data in the file [Filename]
setpwd	Activate password for online access
startprg	Start user program
stopprg	Stop user program
tsk	Output IEC task list with task information
canload	Display of the loading of the CANopen fieldbus

Table 25: Target system specific PLC 2.3 browser commands

Command	Description
clearerrorlist	Erase error list
cleareventlist	Erase event list
copyprojtommc	Copy project to Multimedia Card
copyprojtousb	Copy project to USB drive
getbattery	Display battery status
getcomconfig	Display baud rate of serial interface 1
geterrorlist	Display error list
geteventlist	Display event list
getipgateway	Display Gateway address
getipconfig	Display Ethernet address
getlanguage	Display dialog language for the error list
getmacaddress	Display MAC address [80-80-99-2-x-x]
getrtc	Display data and time [YY:MM:DD] [HH:MM:SS]
getswitchpos	Display status of the operating switch
gettargetname	Display device name
getversion	Display version information
memdisk_sys	displays the free memory at "disk_sys"
reboot	Accept changes (registry save) and restart PLC
saveregistry	Accept modifications
setcomconfig	Set the baud rate of the serial interface [setcomconfig4800,9600,19200,38400,57600,115200]
setipconfig	Set Ethernet configuration [setipconfig adr1.adr2.adr3.adr4 mask1.mask2.mask3.mask4]
setipgateway	Set Gateway address [adr1.adr2.adr3.adr4]
setlanguage	Determine dialog language for error list [deu/eng/fra/ita]
setrtc	Set data and time [YY:MM:DD] [HH:MM:SS]
settargetname	Set device name [Device name]
shutdown	Accept changes (registry save) and switch off PLC

Communication parameter access

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

- setcomconfig
- setipconfig
- setipgateway
- settargetname

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

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

The "setcomconfig", "setipconfig", "setipgateway", "settargetname" commands are represented in the following figures, and with additional expansions are entered in the command line of the PLC browser and confirmed with "Return". With "Return" you receive the respective feedback information (field with grey background).

Change baud rate (setcomconfig)

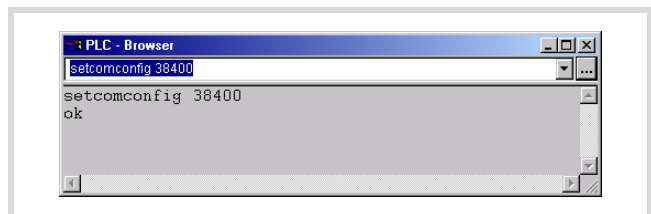


Figure 84: Browser command "setcomconfig"

Change IP address (setipconfig)

The IP net mask is also to be entered with the IP address.

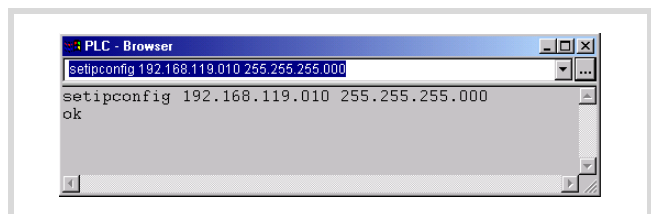


Figure 85: Browser command "setipconfig"

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. In Figure86 this is for example "Xc201_Nr010".

The target name is automatically generated in dependence on the IP address and the target system. It can be called via "gettargetname".

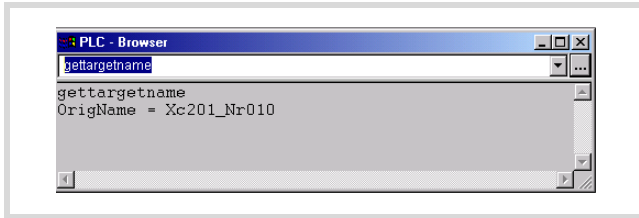


Figure 86: Browser command "gettargetname"

Change IP gateway address (setipgateway)

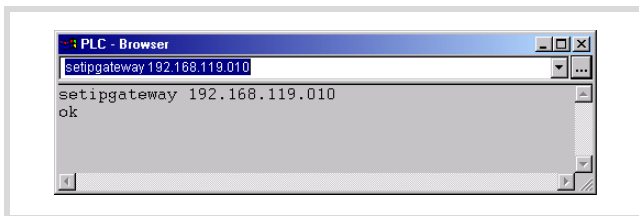


Figure 87: Browser command "setipgateway"

Change target name (settargetname)

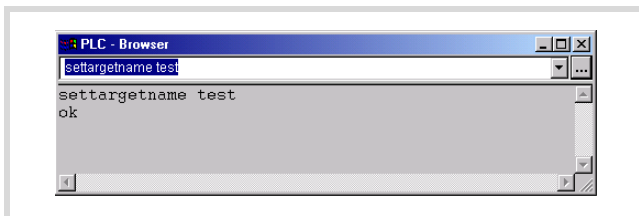


Figure 88: Browser command "settargetname"

Parameterize date and time (setrtc)

You can reprogram the date and time with the "setrtc" browser command.

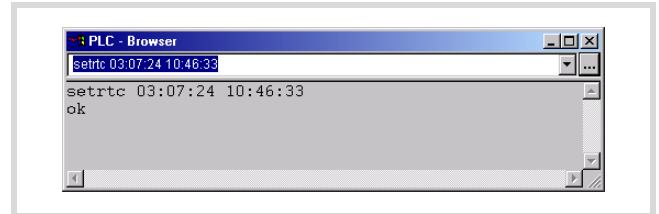


Figure 89: Browser command "setrtc"

Display CPU loading (plcload)

The "plcload" browser command informs about the current system loading of the CPU.

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

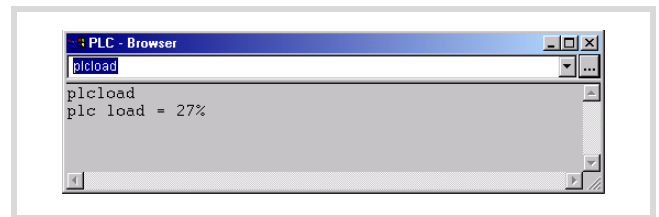


Figure 90: Browser command "plcload"

Display memory assignment of the "disk_sys" (memdisk_sys)

This command informs about the current assignment of the "disk_sys" drive of the user program. Additionally, the memory space still free is listed.

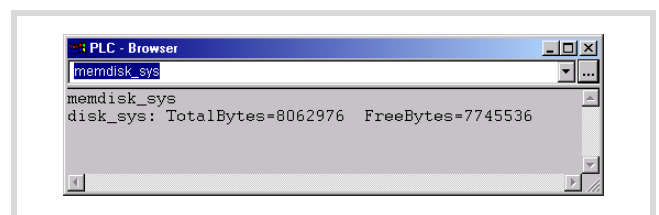


Figure 91: Browser command "memdisk_sys"

Browser command "canload" for XC200

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

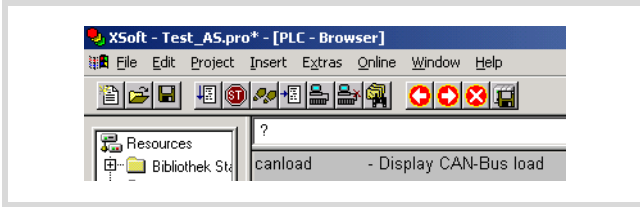


Figure 92: "canload" command for display of the CAN bus load

Examples for display:

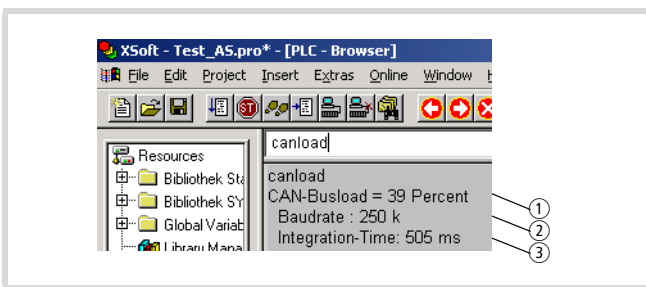


Figure 93: Loading of the CAN bus (Example 1)

- ① Loading of the CAN bus in 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 cant be changed via the browser.

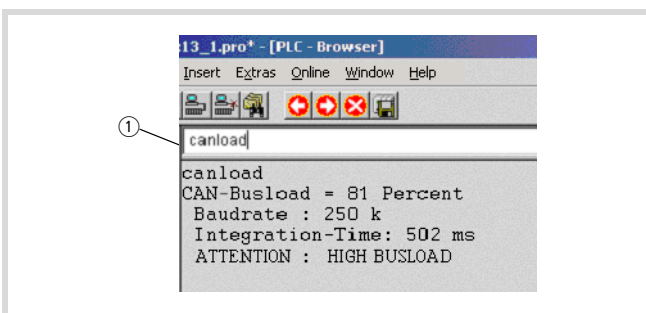


Figure 94: Loading of the CAN bus with alarm message (Example 2)

- ① Alarm message, → table 26

Table 26: Possible alarm messages

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

Access to memory objects

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

- filecopy
- filename
- FileDelete
- filedir

Examples:

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

The browser commands "copyprojtommc" and "copyprojtousb" copy the project directory onto the MMC or USB memory medium:

Example:

```
filecopy \\disk_sys\\project\\*. * \\disk_mmc\\MOELLER\\XC-CPU201-EC512K-8DI-6DO \\project\\*. *
filecopy \\disk_sys\\project\\*. * \\disk_usb\\MOELLER\\XC-CPU201-EC512K-8DI-6DO \\project\\*. *
```

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

Error and event list after calling browser commands

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

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

Examples for language conversion:

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

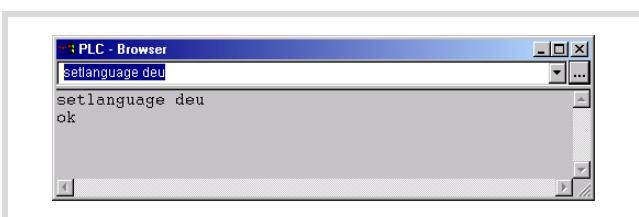


Figure 95: Browser command "setlanguage"

Additional help information for the browser commands

More detailed information is available for the following browser commands. This extended information can be called by using the following syntax in the command line of the PLC browser: <? Command>

mem: <? mem>
 reflect: <? reflect>
 setcomconfig: <? setcomconfig>
 setipconfig: <? setipconfig>
 setipgateway: <? setipgateway>
 settargetname: <? settargetname>
 filedelete: <? filedelete>
 filerename: <? filerename>
 filecopy: <? filecopy>
 filedir: <? filedir>
 copyprojtommc: <? copyprojtommc>
 copyprojtousb: <? copyprojtousb>
 plcload: <? plcload>
 setrtc: <? setrtc>

➔ Between "?" and the browser command, a space is required.

Following is an overview of the messages which can occur in the browser error and event lists:

Module ID	Abbreviation	
1	RTS	The module ID indicates which program signals the error. The Event-ID defines the fault number of the program. The error number can start at 0 for every module ID.
2	CST	
3	XIO	
4	CAN	
5	IEC	

RTS = runtime system

CST = Moeller specific device adaptation

Module 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

Module ID	Event-ID	Error message
2	8	Task monitoring
4	10	CAN controller started
4	20	CAN controller stopped
4	30	Overflow
4	31	Overflow
4	40	Overflow
4	41	Overflow
4	42	Overflow
4	50	Critical CAN fault
4	60	CAN controller in status error warning
4	70	CAN controller in status Bus-Off
1	16	Task monitoring fault
1	17	Hardware monitoring fault
1	18	Bus error
1	19	Checksum error
1	20	Fieldbus error
1	21	I/O update fault
1	22	Cycle time exceeded
1	80	Invalid instruction
1	81	Access violation
1	82	Privileged instruction
1	83	Page fault
1	84	Stack overflow
1	85	Invalid scheduling
1	86	Invalid access Identity
1	87	Access on protected page
1	256	Access to uneven address
1	257	Array limit exceeded
1	258	Division by zero
1	259	Overflow
1	260	Exception cant be overlooked
1	336	Floating decimal point: General fault
1	337	Floating decimal point: Not normalised operand
1	338	Floating decimal point: Division by zero
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

9 RS232 interface in transparent mode

In transparent mode the data transfer occurs between the XC200 and data terminals (e.g. terminals, printers, PCs, measurement devices) without interpretation of the data. For this purpose, the serial RS 232 interface of the XC-CPU201 (COM1) or the XIOC-SER modules (COM2/3/4/5) is to be switched using the user program in the transparent mode. This applies from operating system version 01.03.xx for the RS232 interface of the XC-CPU201 (COM1).

→ If the RS232 interface of the CPU is in transparent mode, programming via this interface is not possible. However, the program can be tested via the Ethernet interface.

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

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

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

The control lines of the RS232 of the XIOC-SER modules are controlled with the "SysComWriteControl" function from the "xSysCom200.lib" library and monitored with the "SysComReadControl" function.

The RS232 interface of the XC-CPU201 does not have any control lines in contrast to the RS232 interface of the XIOC-SER module.

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

xSysCom200.lib: 300,...,115200

SysLibCom.lib: 4800,...,115200

The local serial interface COM1 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 COM1 is contacted should have an interval time of at least 50 ms and be assigned with a low priority (high value) in multitasking mode, so that time critical tasks are not hindered.

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

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

Programming of the RS232 interface in transparent mode

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

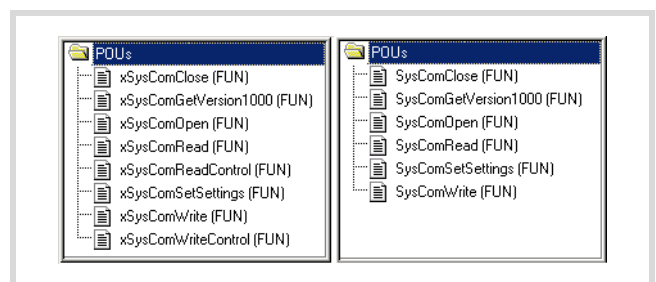


Figure 96: Overview (left: xSysCom200.lib, right: SysLibCom.lib)

The individual functions are described in the following.

Function "(x)SysComClose"

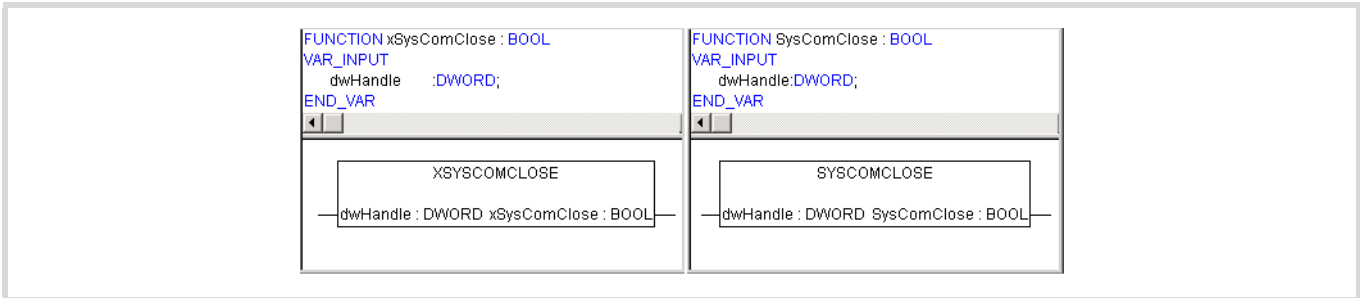


Figure 97: Function "(x)SysComClose"

Description

The function closes the RS232 interface. The communication parameters which were set last are restored during closing. The function provides TRUE as a return value if the action is completed successfully.

Parameters

dwHandle	Return value of the "(x)SysComOpen" function
(x)SysComClose	Return value TRUE: closing the RS232 interface was successful

Function "(x)SysComOpen"

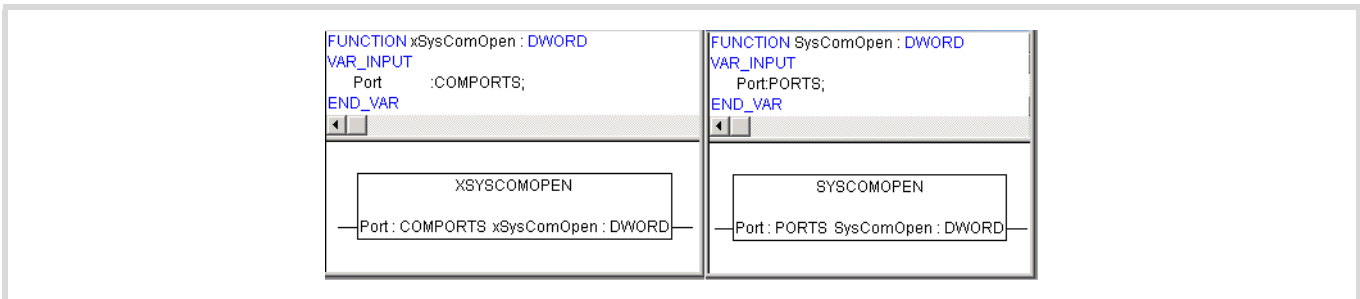


Figure 98: Function "(x)SysComOpen"

Description

The function opens the RS232 interface for transparent mode. After successful opening of the interface the function returns a value greater than "0".

- ▶ Enter this value with the following functions as the "dwHandle" parameter.

If a fault occurs, the return value is equal to "0". Transparent mode of the interface will not be enabled.

After opening of the RS232 interface the parameters can be set with the assistance of Function "(x)SysComSetSettings" (→ page 71). The values set beforehand for the XIOC-SER in the PLC configurator, and the default settings of the CPU are then ignored and are only valid again after the RS232 interface is closed.

Parameters

Port	Selection of the interface
	Parameter: Specified for the open interface.
(x)SysComOpen	Return value 0: Opening of the RS232 interface was not successful.
	Return value > 0: Opening of the RS232 interface was successful.

Data types

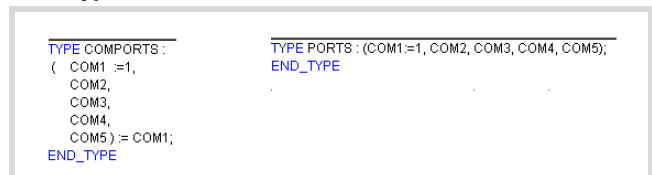


Figure 99: COMPORTS/PORTS data types

Function "(x)SysComRead"

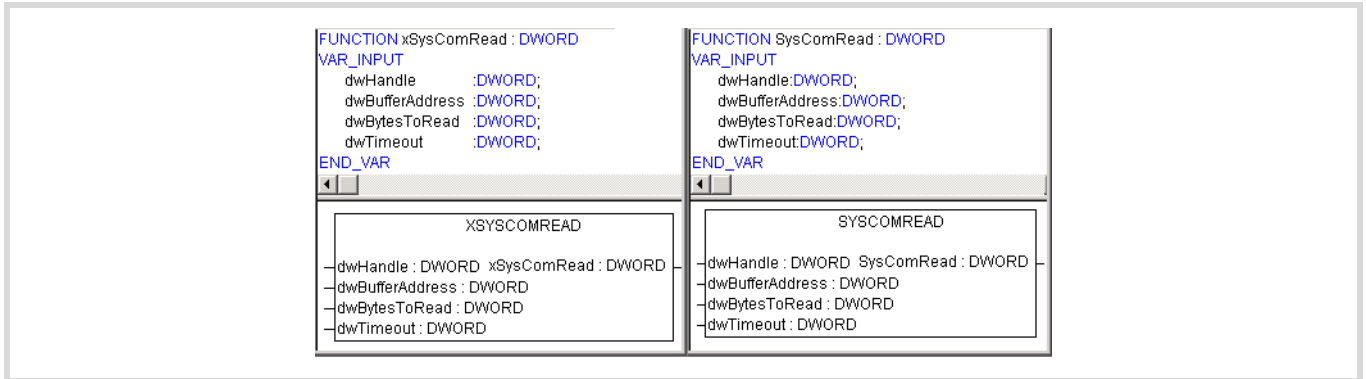


Figure 100:Function "(x)SysComRead"

Description

Data received via the RS232 interface in transparent mode can be read with this function.

Parameters

dwHandle	Return value of the "(x)SysComOpen" function
dwBufferAddress	Address under which the read data is stored
dwBytesToRead	Limitation of the max. number of data bytes (COM 2 to COM 5: max. 250 bytes)
dwTimeout	Parameters without meaning
(x)SysComRead	Return value provides information about the number of read data bytes.

Important
A test of the buffer address or buffer size does not occur!

Function "xSysComReadControl"

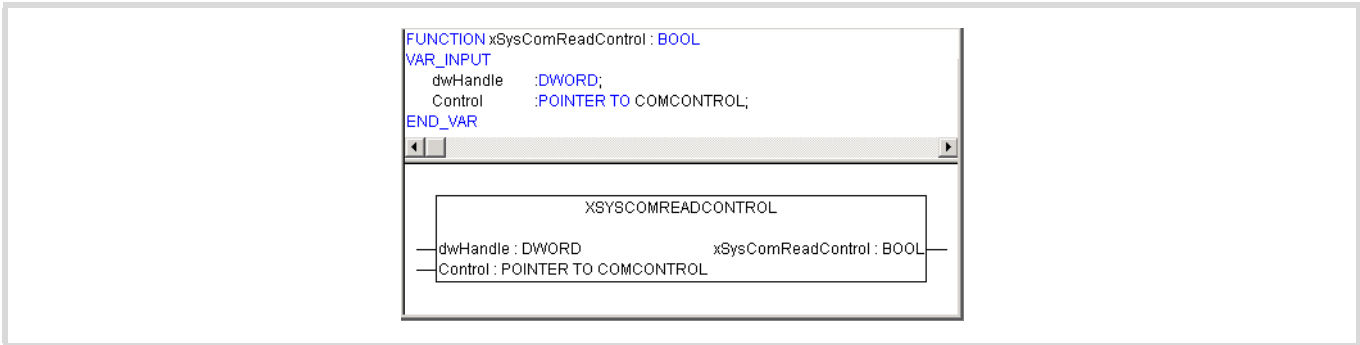


Figure 101:Function "xSysComReadControl"

Description

The XIOC-SER hardware interface module avails of control/ interface lines. It provides the "xSysComReadControl" module with read access to the control/interface lines of the COM 2 to COM 5 interfaces.

Parameters

dwHandle	Return value from the "xSysComOpen" function
Control	COM 2 to COM 5: TRUE = read command on the control lines of the hardware interface
xSysComReadControl	COM 2 to COM 5: TRUE = read command was successful; FALSE = read command was unsuccessful

Data type

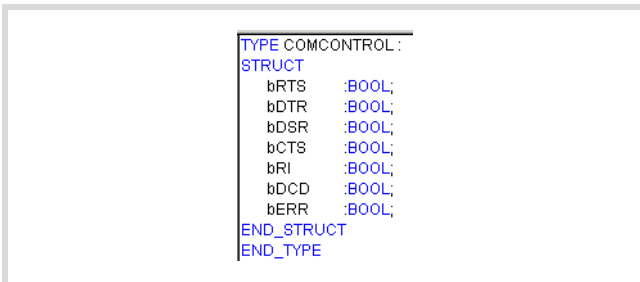


Figure 102:COMCONTROL data type

Function "(x)SysComSetSettings"

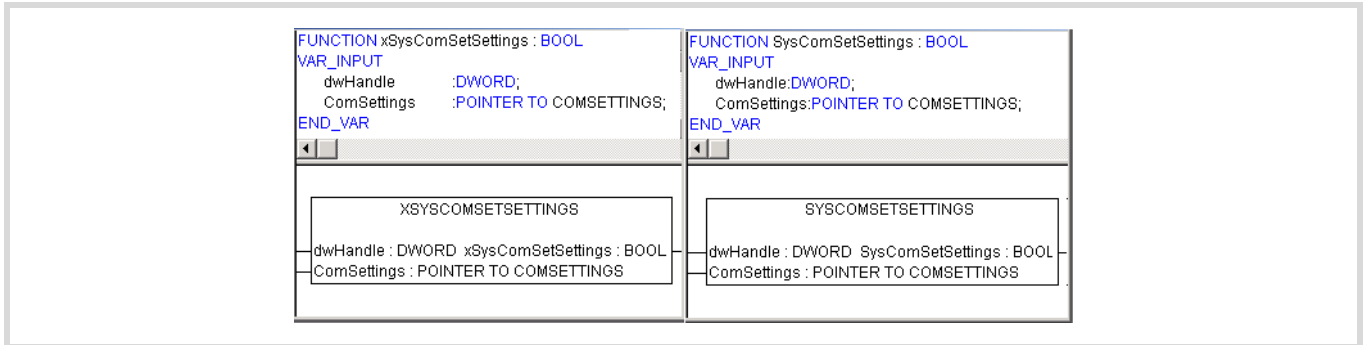


Figure 103:Function "(x)SysComSetSettings"

Description

Interface parameters of the RS232 interface for the transparent mode can be set with this function.

Parameters

dwHandle	Return value of the "(x)SysComOpen" function
ComSettings	Pointer to the memory area in which the interface parameters are stored
(x)SysComSetSettings	TRUE return value if the interface has been successfully parameterized; otherwise FALSE

Data types

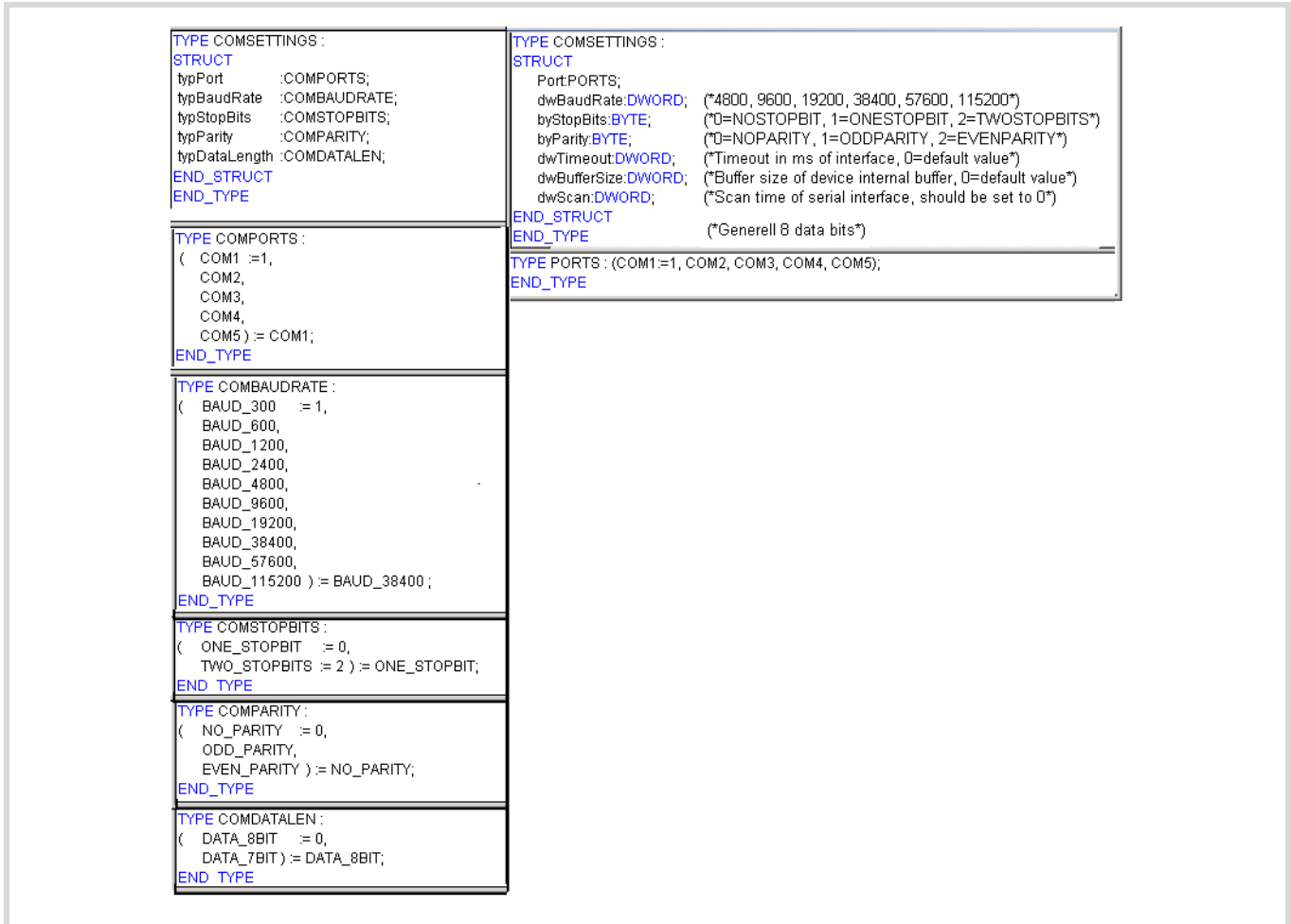


Figure 104:COMSETTINGS data type

Function "(x)SysComWrite"

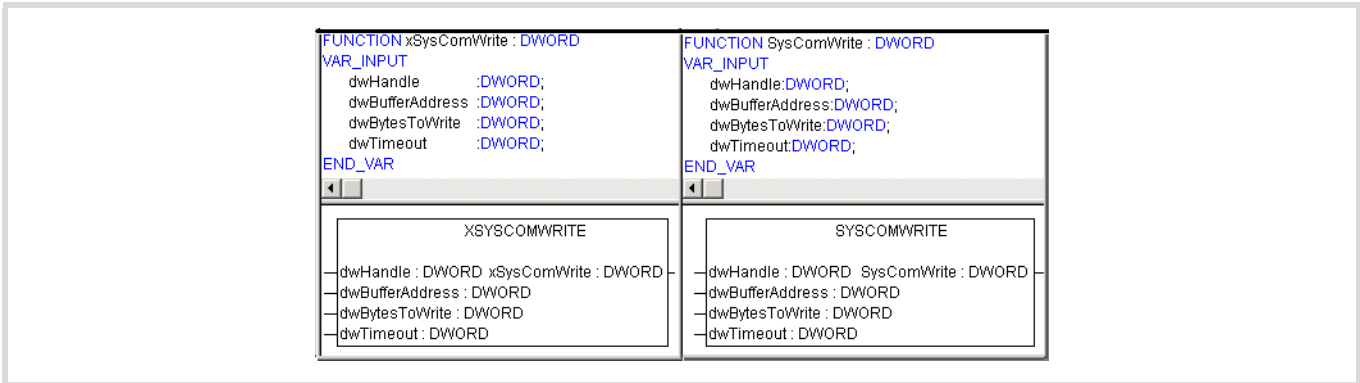


Figure 105:Function "(x)SysComWrite"

Description

This function allows output of the data via the RS232 interface.

Parameters

dwHandle	Return value of the "(x)SysComOpen" function
dwBufferAddress	Address under which the data to be output is stored
dwBytesToWrite	Number of data bytes to be sent (COM 2 to COM 5: max. 250 bytes)
dwTimeout	Parameters without meaning
(x)SysComWrite	Return value provides information about the amount of sent data.



Important

A test of the buffer address or buffer size does not occur!

Data type

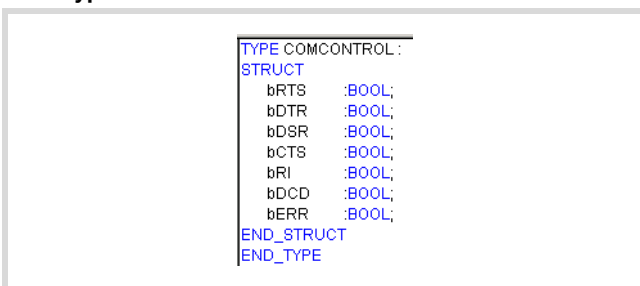


Figure 106:COMCONTROL data type

Function "(x)SysComWriteControl"

→ This function can only be used with the XIOC-SER module!

The XIOC-SER hardware interface module avails of control/interface lines. It provides the "SysComWriteControl" module with write access to the control/interface lines of the COM 2 to COM 5 interfaces.

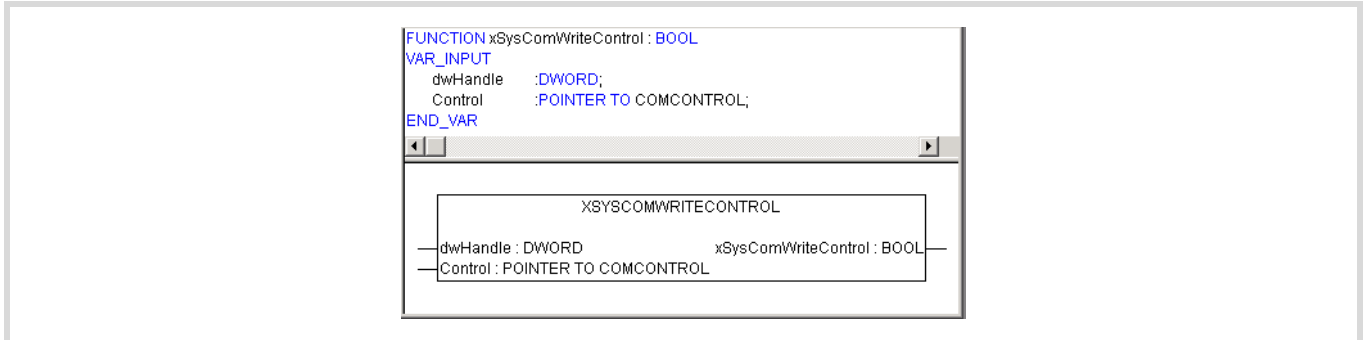


Figure 107:Write access to the control lines of the COM 2 to COM 5 interface

Parameters

dwHandle	Return value of the "(x)SysComOpen" function
Control	COM 2 to COM 5: TRUE = write command on the control lines of the hardware interface
(x)SysComReadControl	COM 2 to COM 5: TRUE = write command was successful; FALSE = write command was unsuccessful

Automatic closing of the interface

With a state change of the XC200 to STOP the transparent mode is ended automatically by the operating system. The interface is initialised again with the interface parameters last set.

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;
  Closesresult: BOOL;
  Coun: DWORD;
  RESET: BOOL;
END_VAR

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

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


Appendix

USB-Stick-Typen

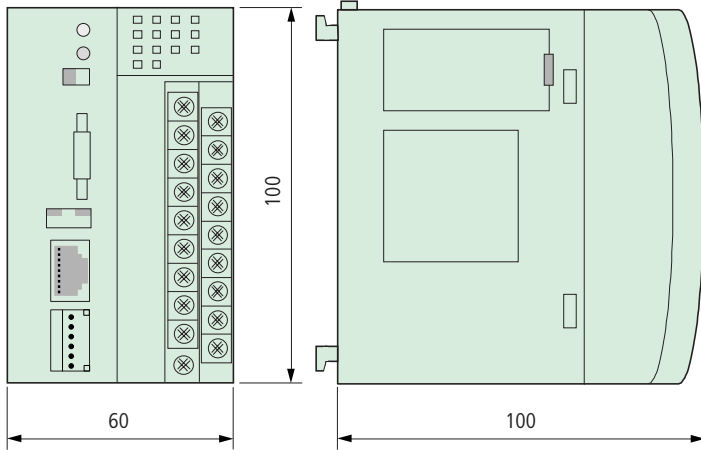
Manufacturer	Aiptec	extreme- memory	Traxdata	ScanDisk	Kingston		Fuj./Siemens	ScanDisk	ScanDisk
Type designation	MP3/WMA Player		EZ Drive 2.0	cruzer micro	Data Traveler ELITE	Pen Drive	Memorybird	Cruzer Mini	Cruzer Mini
USB specification	1.1	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
Memory size [MByte]	256	256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

1) Time which the operating system requires to mount the USB stick

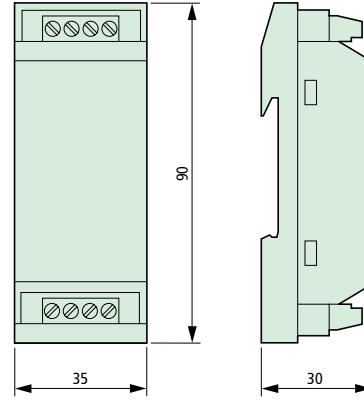
2) Control access of user programs to the USB stick with extended mount times, to ensure that the mount time must elapse before access is allowed after the stick is plugged-in or program is started.

Dimensions

XC-CPU201...



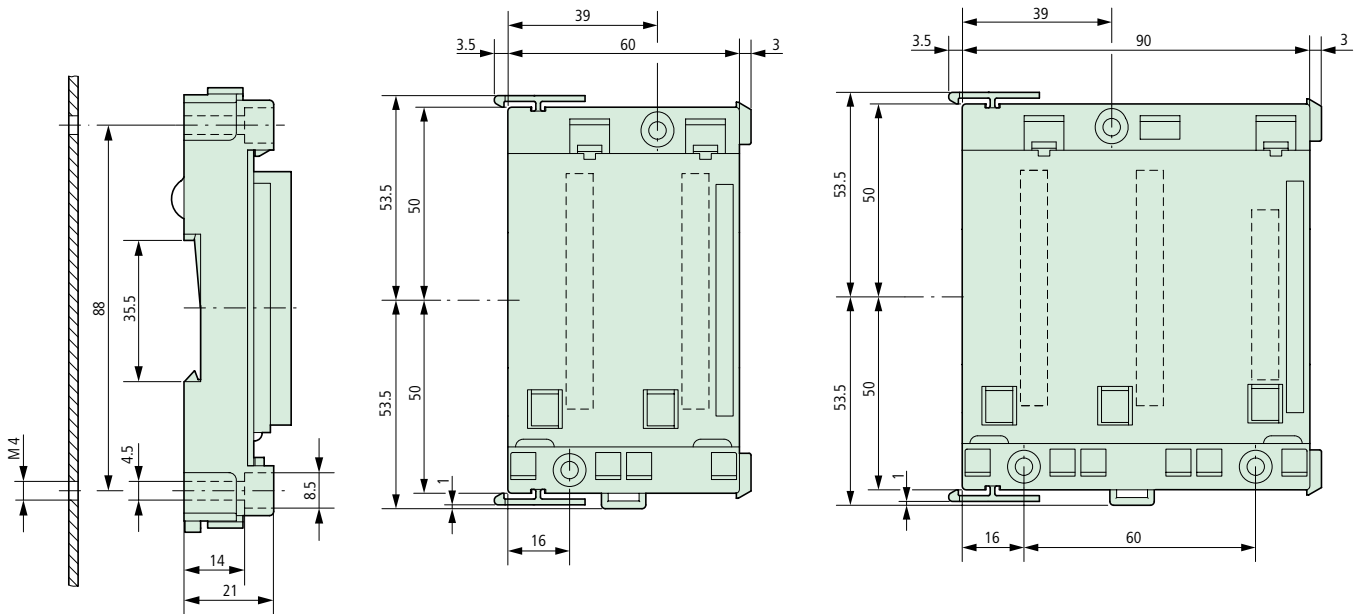
XT-FIL-1 line filter



Module rack

XIOC-BP-XC

XIOC-BP-XC1



Technical data

			XC-CPU201-EC256-8DI-6DO(-XV) XC-CPU201-EC512-8DI-6DO(-XV)
General			
Standards and regulations			IEC/EN 61131-2 EN 50178
Ambient temperature	°C		0 to +55
Storage temperature	°C		-25 to +70
Mounting position			horizontal
Relative humidity, no condensation (IEC/EN 60068-2-30)	%		10 to 95
Air pressure (in operation)	hPa		795 to 1080
Vibration resistance			10 to 57 Hz ± 0.075 mm 57 to 150 Hz ± 1.0 g
Mechanical shock resistance			15 g/11 ms
Overvoltage category			II
Pollution degree			2
Enclosure protection			IP20
Rated insulation voltage	V		500
Interference emission			EN 50081-2, Class A
Interference immunity			EN 50082-2
Battery (lifetime)			Worst case 3 years, typ. 5 years
Weight	kg		0,23
Dimensions (W × H × D)	mm		90 × 100 × 100
Connecting terminals			Plug-in terminal block
Conductor cross-section			
Screw terminals			
stranded, with bootlace ferrule	mm ²		0.5 to 1.5
solid core	mm ²		0.5 to 2.5
Spring-loaded terminal			
stranded	mm ²		0.14 to 1.0
solid core	mm ²		0.34 to 1.0
Electromagnetic Compatibility (EMC)			→ page 82
Supply voltage for the CPU (24 V/0 V)			
Mains failure bridging			
Drop-out duration	ms		10
Repeat rate	s		1
Input rated voltage	V DC		24
Permissible range	V DC		20.4 to 28.8
Current consumption	A		typ. 1.4
Residual hum and ripple	%		≤ 5
Maximum power dissipated (without local I/O)	P _V	W	6
Overvoltage protection			Yes
Polarity protection			Yes

			XC-CPU201-EC256-8DI-6DO(-XV) XC-CPU201-EC512-8DI-6DO(-XV)
External supply filter			Type: XT-FIL-1, see the technical data on Page 83
Internal supply filter			Yes
Switch-on current surge		$\times I_n$	Not limited, (limiting only by a supply-side 24 V DC PSU)
Output voltage for the signal modules			
Output rated voltage		V DC	5
Output current		A	3,2
Off-load stability			Yes
Short-circuit proof			Yes
Electrically isolated from supply voltage			No
CPU			
Microprocessor			RISC processor
Memory			
Program code (EC256K/EC512K)		kByte	256/512
Program data (EC256K/EC512K)		kByte	256/512
Marker (EC256K/EC512K)		kByte	16/16
Retain data (EC256K/EC512K)		kByte	32/32
Persistent data (EC256K/EC512K)		kByte	32/32
Interfaces			
Multimedia card			Yes, optional, 16 MB or 32 MB, to be ordered separately
Ethernet interface			
Data transmission rate		MBit/s	10/100
Connection by			RJ45
Potential isolation			No
RS232 interface (without handshake line)			
Data transmission rate		kBit/s	38,4
Connection by			RJ45
Potential isolation			No
in the "transparent mode"			
Data transfer rates			300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 Bit/s
Character formats			8E1, 8O1, 8N1, 8N2, 7E2, 7O2, 7N2, 7E1
CANopen			
Maximum data transmission rate		kBits/s	20/50/100/125/250/500/800/1000
Electrical isolation			Yes
Device profile			to DS301V4
PDO type			asyn., cyc., acyc.
Connection			Plug-in spring-loaded terminal block, 6-pole
Bus termination resistors			External
Participant		No.	max. 126
USB interface, V1.1			
Data transfer rate (Autochanging)		MBit/s	1,5/12
Electrical isolation			No

			XC-CPU201-EC256-8DI-6DO(-XV) XC-CPU201-EC512-8DI-6DO(-XV)
Power supply for connected devices:			
Rated voltage	V DC		5
max. current	A		0,5
Connection by			Downstream plug
Watchdog			Yes
RTC (Real Time Clock)			Yes
Supply voltage for the local inputs/outputs (24 V _Q /0 V _Q)			
Rated voltage	V DC		24
Voltage range	V DC		20.4 to 28.8
Current consumption	A		typ. 1
Electrical isolation			
Between supply and CPU voltage			Yes
Overvoltage protection			Yes
Polarity protection			Yes
Digital inputs			
Input rated voltage	V DC		24, observe polarity
Voltage range	V DC		19.2 to 30
Input current per channel at rated voltage			
Functionality: Normal digital input	mA		typ. 3.5
Functionality: Fast digital input	mA		typ. 7
Power dissipation per channel			
Functionality: Normal digital input	mW		typ. 85
Functionality: Fast digital input	mW		typ. 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		typ. 0.1
On → Off	ms		typ. 0.1
Functionality: Fast digital input			
Off → On	μs		typ. 7
On → Off	μs		typ. 1
Inputs	No.		8
Channels with common reference potential	No.		8
Of which can be used as			
Interrupt inputs	No.		2
Counter input 32 Bit or	No.		1
Counter input 16 Bit or	No.		2
Incremental encoder input (Track A, B, C)	No.		1
Status indicator			LED

			XC-CPU201-EC256-8DI-6DO(-XV) XC-CPU201-EC512-8DI-6DO(-XV)
Digital outputs			
Power dissipation per channel			
QX0.0 and QX0.5	W		0,08
Load circuits			
QX0.0 and QX0.5	A		0,5
Output delay			
Off → On			typ 0.1 ms
On → Off			typ 0.1 ms
Channels	No.		6
Channels with common reference potential	No.		6
Status indicator			LED
ON-time (duty cycle)	% ED		100
Simultaneity 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 coupling)		1 kV
Surge (IEC/EN 61000-4-5)	Digital I/O, unsymmetrical		0.5 kV
	Analog I/O, unsymmetrical, coupling on the screen		1 kV
	Mains DC, unsymmetrical		1 kV
	Mains DC, symmetrical		0.5 kV
	Mains AC, unsymmetrical		2 kV
	Mains AC, symmetrical		1 kV
Cable conducted interference, induced by high frequency fields (perviously: immunity to line-conducted interference) (IEC/EN 61000-4-6)			3 V

			XT-FIL-1 24 V DC filter
General			
Standards and regulations			IEC/EN 61131-2 EN 50178
Ambient temperature	°C		0 to +55
Storage	°C		-25 to +70
Mounting position			Horizontal/vertical
Relative humidity, no condensation (IEC/EN 60068-2-30)	%		10 to 95
Air pressure (in operation)	hPa		795 to 1080
Vibration resistance			10 to 57 Hz ±0.075 mm 57 to 150 Hz ±1.0 g
Mechanical shock resistance			15 g/11 ms
Shock resistance			500 g/∅ 50 mm ± 25 g
Overvoltage category			II
Pollution degree			2
Enclosure protection			IP20
Rated impulse voltage	V		850
Interference emission			EN 50081-2, Class A
Interference immunity			EN 50082-2
Weight	g		95
Dimensions (W × H × D)	mm		35 × 90 × 30
Connecting terminals			Screw terminal
Conductor cross-section			
Screw terminals			
stranded, with bootlace ferrule	mm ²		0.2 to 2.5 (AWG22-12)
solid core	mm ²		0.2 to 2.5 (AWG22-12)
Power supply			
Input voltage	V DC		24
Permissible range	V DC		20.4 to 28.8
Residual hum and ripple	%		≤ 5
Overvoltage protection			Yes
Electrical isolation			
Input voltage to PE			Yes
Input voltage to output voltage			No
Output voltage to PE			Yes
Output voltage	V DC		24
Output current	A		2,2

Index

A	Abbreviations	5		
B	Battery	18		
	Battery buffer	38		
	Battery change	18		
	Block diagram			
	CPU	13		
	Supply module	8		
	Breakpoint	22		
	Browser commands	61		
	Bus expansion, local	9		
C	Cabinet layout	19		
	Cable routing	19		
	CAN stack	30		
	CAN telegrams, receive/send from user program	16		
	CAN_BUSLOAD	57		
	CANopen bus expansion	39		
	CANopen interface	16		
	Change baud rate	62		
	Change IP address	62		
	Change IP gateway address	63		
	Change target name	63		
	Cold start	22		
	COLDSTART	22		
	Commissioning	22		
	Communication parameters	41		
	Communications fault	40		
	Configuration, inputs/outputs	45		
	Connect PC	41		
	Connect programming device	41		
	Connecting the incremental value encoder	11		
	Connecting the power supply	20		
	Connection			
	Incremental encoder	11		
	Interrupt inputs	12		
	PSU and local I/O	9		
	Up/down counter	12		
	Connection establishment, PC – XC200	41		
	Count direction	9		
	Count input, counter	10		
	CPU loading	63		
	CPU module	7		
	Create bootable project	23		
	Cyclic task	27		
D	Data remanence	38		
	Data-saving	18		
	Debugging	22, 61		
	Diagnostics	61		
	Direct peripheral access	11, 34		
	Error code	38		
	Direction signal	10		
	Display memory assignment	63		
	Documentation, online	6		
	Download of programs	40		
	Drives	13		
E	Electromagnetic contamination	19		
	Engineering	19		
	Error code with "direct peripheral access"	38		
	Error list	65		
	Event controlled task	28		
	Event list	65		
	Expansion, local bus	9		
	Extend transfer time, of programs	40		
F	Forcing	22		
	Function blocks	32		
	Functions	32		
	Functions, XIOC			
	DisableInterrupt	52		
	EnableInterrupt	52		
	GetSlotPtr	38		
	ReadBitDirect	37		
	ReadWordDirect	36		
	WriteBitDirect	37		
	WriteWordDirect	37		
H	Halt behaviour	22		
I	IEC functions			
	DeleteErrorList	55		
	DeleteEventList	55		
	GetErrorID	56		
	GetEventID	56		
	GetNrOfErrors	56		
	GetNrOfEvents	56		
	WriteError	57		
	WriteEvent	57		
	Incremental encoder input	9		
	Inductors	19		
	Input configuration and parameterization	45		
	Installation, CPU	18		
	Interface			
	CANopen	16		
	USB	14		
	Interference factors	19		
	Interrupt inputs	10		
	Connecting	12		
	Interrupt processing	52		
	IP address	58		
	IPGateway address	58		

L	Layout of units	19	S	Scan/Modify the IP address	43
	LED indicators	9		Segments	40
	Lightning protection	20		Set IP address	58
	Limit values, for memory usage	40		Set subnetmask address	58
	Local bus expansion	9		Setting IPGateway address	59
<hr/>				Shielding	19
M	Memory systems	13		Single-cycle mode	22
	Memory usage, limit values	40		Single-step mode	22
	Memory, application program	12		Start behaviour	22
	Monitoring, system voltage	13		Start behaviour at Power-On	22
	Mounting position	19		Subnetmask address	58
	Multimedia card	14		Supply interruption	38
	Multitasking, behaviour of the CAN stack	30		Suppressor circuitry for interference sources	19
<hr/>				Switch-off behaviour	22
O	Online documentation	6		System event	28
	Operating mode selector switch	13		System libraries	32
	Operating states	39		System loading, CPU	63
	Operating system		T	Task	
	Erasing, Multimedia card	25		Cyclic	27
	Transfer, from PC to the PLC	24		event controlled	28
	Update	23		Task configuration	27
	Output configuration and parameterization	45		Task monitoring	30
	Output MAC address	58		Terminal assignments	9
<hr/>				Test and commissioning	22
P	Parameter setting		U	Uninterruptible power supply	38
	Inputs/outputs	45		Up/down counter	10
	Parameterization	23		Connecting	12
	PFI signal	22		USB Card	14
	Point-to-point connection	14		USB interface	14
	Power supply, for processor unit and local inputs/outputs	8		USB stick	14
	Program date	63	<hr/>		
	Program download	13, 40	V	Ventilation	19
	Program processing	27		Versions, CPU	12
	Program run interrupted	22		Voltage dip	22
	Program time	63	<hr/>		
	Programming cable		W	Warm start	22
	for RS232 interface	41		Watchdog	30
<hr/>				Watchdog sensitivity	31
R	Real-time clock	18		Web visualization	39
	Parameter setting	63		Wiring	19
	Reference window	11		Wiring example	
	Registry save	59		Power supply of the digital inputs/outputs	20
	Reset	23		PSU	20
	Residual cycle	38	<hr/>		
	Restart, registry save	59	X	XIOC modules	7
	RJ45 interface	14			