Programmable Logic Controller
easyControl EC4-200

E·T·N

Powering Business Worldwide

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

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

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

⚠ Danger!
Dangerous electrical voltage!

Before commencing the installation

- Disconnect the power supply of the device.

- Ensure that devices cannot be accidentally restarted.

- Verify isolation from the supply.

- Earth and short circuit.

- Cover or enclose neighbouring units that are live.

- Follow the engineering instructions (AWA) 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

Contents

→ The previous Chapter 17: "The easyNet network" and
    Chapter 18: "Programming via easyNet (routing)" are
    omitted.

    You will find this information in far greater detail in the
    manual MN05006004Z-EN (previously 08/07 AWB2786-
    1593) "Data transfer between easy and IEC PLCs
    (easyNet)".

About this manual

List of revisions

The following significant amendments have been introduced since previous issues:

Edition date	Page	Keyword	new	Modification
12/06	87	EC4-200 network modules		
03/07	9	Addition of types EC4P-222-…	✓	
	14	Ethernet interface	✓	
	15	Cable connections	✓	
	59	Browser commands for Ethernet	✓	
	67	Startup.INI mit with Ethernet entries	✓	
	104	Character sets	✓	
01/08	29	Selecting or toggling between menu items		✓
	33	System menu		✓
	49	START, COLD START, WARM START, STOP		✓
	50	Interrupt inputs I1 … I4		✓
	51	Timer interrupt		✓
	87	Chapter 17: "The easyNet network" and Chapter 18: "Programming via easyNet (routing)" are omitted. You will find this information in far greater detail in the manual MN05006004Z-EN (previously 08/07 AWB2786-1593en) " Data transfer between easy and IEC PLCs (easyNet)".		✓
10/10	all	Change to Eaton notation		✓

Additional documentation

At different points in this manual, references are made to more detailed descriptions in other manuals. This documentation is stored as a PDF file when the product CD is installed on your PC.

To find documentation choose the following in the Windows Start menu:

Programs → Moeller Software → easy Soft CoDeSys → Documentation…

It is also possible to download the PDF files from the FTP server. This always provides the latest data.

ftp://ftp.moeller.net/DOCUMENTATION/AWB_MANUALS/

Concrete information regarding communication with CAN stations and their configuration can be found in the following listed documentation:

- AN27K19GB: Communication between two PLCs using network variables via CAN (AN2700K19GB.PDF)
- AN27K20GB: Coupling multiple stand-alone PLCs (CAN-Device) via CANopen (AN2700K20GB.PDF)
- Engineering of CAN stations (AN2700K27GB.PDF) (To be found in Windows start menu under Programs → Moeller Software → easy Soft CoDeSys → Application examples…)
- MN05010001Z-EN (previously AWB2786-1554GB): Library description CANUser.lib, CANUser_Master.lib. The functions of the CANUser.lib and CANUser_Master.lib libraries enable you to access CAN objects directly. (To be found in Windows start menu under Programs → Moeller Software → easy Soft CoDeSys → Documentation…)

Reading conventions

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

→ Draws your attention to interesting tips and supplementary information.

▽ **Caution!**
Warns of the risk of material damage.

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

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

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

1 Device application

The controllers of the EC4-200 series are programmable switching and control devices. They can be used in domestic applications, machine building and plant construction. An EC4-200 controller can be used as a stand-alone controller or connected to remote input/output devices via the CANopen interface. This interface also allows you to communicate with other PLCs (with a CANopen interface).

The EC4P-222-... controller types have an additional Ethernet interface.

From version 2.0 of the operating system the controllers have the following features:

• Connection of expansion devices/controllers via easyLink
• Connection of the MFD-CP4 multi-function display via the multi-function interface
• Transparent mode via the multi-function interface
• Direct access to local I/O and the high-speed counters
• Integration in the easyNet network via the easyNet/ CAN interface

Controllers from version 2.10 can be connected to the ASI, PROFIBUS-DP, CAN or DeviceNet networks with suitable network interfaces.

The controller is programmed with the easySoft CoDeSys programming software. This software should be installed on a standard PC with the Windows NT, 2000 or XP operating system. Further information on the software is provided in the manual for the programming software (MN05010003Z-EN; previously AWB2700-1437GB).

This software provides you with a simple entry in the IEC programming languages such as:

• Instruction List (IL)
• Function Block Diagram (FBD)
• Ladder Diagram (LD)
• Structured Text (ST)
• Sequential Function Chart (SFC).

This provides a large number of operators such as:

• Logic operators such as AND, OR, …
• Arithmetic operators such as ADD, MUL, …
• Comparison operators such as <,=, >

You use the programming software to create, test and document a project. Functions for analog processing, closed-loop control and function blocks such as timers, counters simplify programming.

EC4-200 part number overview

The EC4-200 series contains controllers with different displays and the type and number of inputs/outputs.

Part no.	Features				
	Keys/display	Transistor outputs	Relay outputs	Analog output	Ethernet connection
EC4P-221-MTXD1	×	8	–	–	–
EC4P-221-MTXX1	–	8	–	–	–
EC4P-221-MRXD1	×	–	6	–	–
EC4P-221-MRXX1	–	–	6	–	–
EC4P-221-MTAD1	×	8	–	×	–
EC1P1-MTAX1	–	8	–	×	–
EC4P-221-MRAD	×	–	6	×	–
EC4P-221-MRAX1	–	–	6	×	–
EC4P-222-MTXD1	×	8	–	–	×
EC4P-222-MTXX1	–	8	–	–	×
EC4P-222-MRXD1	×	–	6	–	×
EC4P-222-MRXX1	–	–	6	–	×
EC4P-222-MTAD1	×	8	–	×	×
EC1P1-MTAX1	–	8	–	×	×
EC4P-222-MRAD	×	–	6	×	×
EC4P-222-MRAX1	–	–	6	×	×

2 Setup
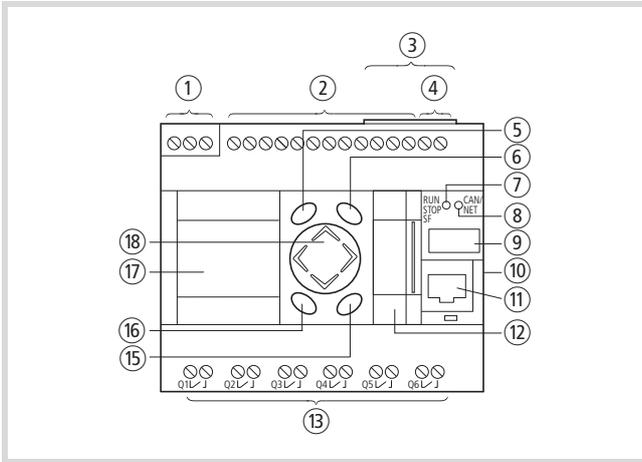
Figure 1: Front of the EC4P-221-MRAD1,
Legend → figure 2
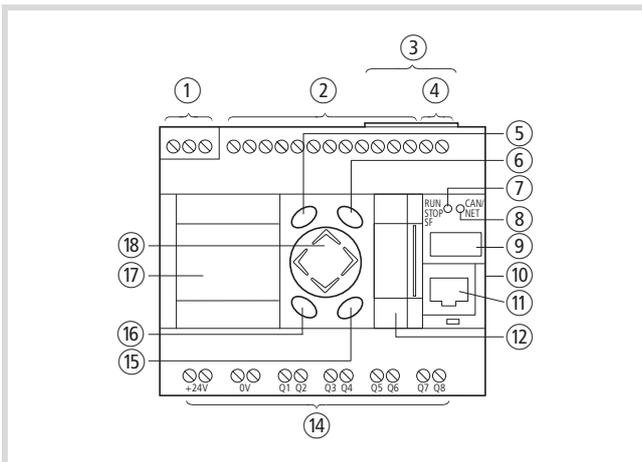
Figure 2: Front of the EC4P-221-MTAD1

① 24 V DC power supply

② Inputs

③ Interface for connecting the CAN network

④ Analog output, 0 – 10 V (not active)

⑤ DEL button

⑥ ALT button

⑦ RUN/STOP/SF LED

⑧ CAN/NET LED

⑨ Field for device labelling

⑩ easyLink interface to expansion device

⑪ Programming interface for connection to a PC

⑫ Multi-function interface

⑬ Relay outputs

⑭ Transistor outputs

⑮ OK button

⑯ ESC button

⑰ LCD display (EC4P-22x-M…D1)

⑱ Cursor buttons P1…P4 (rocker button)

Inputs

Table 1: Type and number of inputs

Digital	12 (I1…I12)	24 V DC
of which can be used as analog	4 (I7, I8, I11, I12)	24 V DC/0…10 V

Inputs I7, I8, I11, I12 can also be used as analog inputs. They are selected in the user program by means of the appropriate syntax used in the PLC configurator.
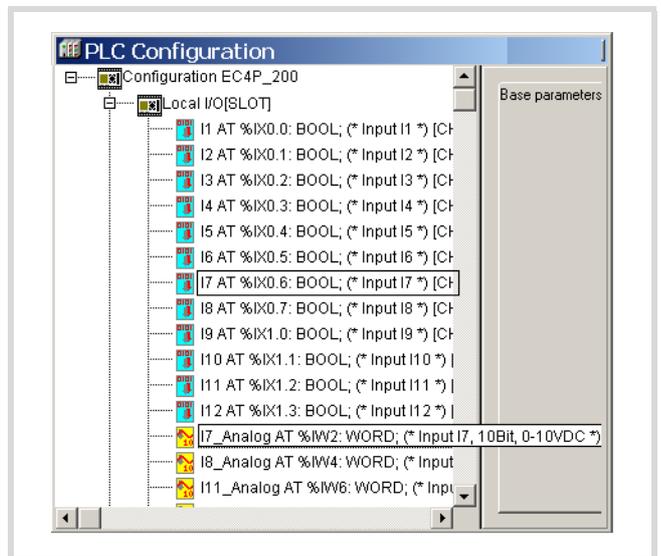
Figure 3: Selection between digital and analog input, e.g. I7

When programming the inputs as digital inputs in the user program, the input voltage of 8 V forms the limit value for the TRUE/FALSE signals.

Voltage [V]	State
$\leqq 8$	FALSE
> 8	TRUE

Technical data: → page 96

Inputs I1, I2, I3, I4 can be used for:

- generating interrupts (inputs I1, I2, I3, I4)
- controlling high-speed counters such as:
  - 16 or 32-bit counters, for counting pulses (I1, I2), up/down counting
  - Incremental counters, 32-bit, for processing the signals of an incremental encoder (I1, I2, I3, I4).

The function is selected in the PLC configuration. However, several functions cannot be used at the same time.

Example: If you are using input I1 for a high-speed counter (16-bit), I2 can be used for another high-speed counter (16-bit) but not for generating an interrupt. Inputs I3 and I4 likewise cannot be used for generating an interrupt.

Connection description → figure 22 on page 23.

Function and cursor buttons as inputs

The front plate of the device is provided with the function buttons DEL, ALT, ESC, OK which are arranged around the rocker switch. The rocker switch is divided into 4 sections with the cursor button designations P1 to P4. The function and cursor buttons are represented in the PLC configuration as inputs. These inputs are scanned in the program according to general syntax rules. Only one button can be actuated at a time, otherwise uncontrolled states may occur when these buttons are scanned.
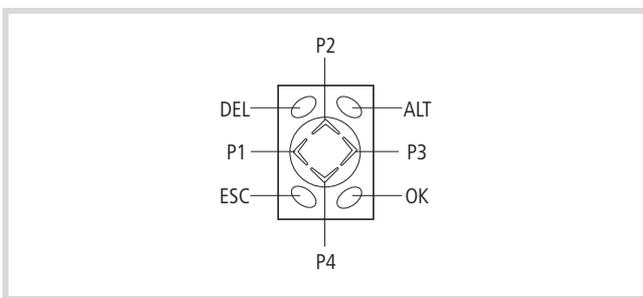
Figure 4: Function buttons and rocker switch with cursor buttons P1, P2, P3, P4
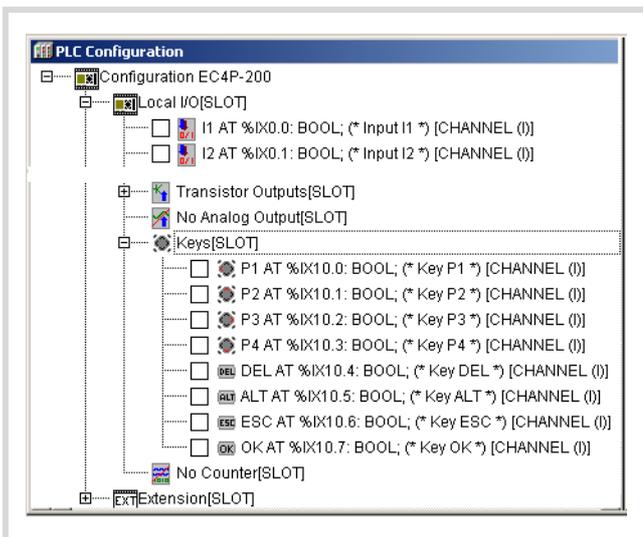
Figure 5: Inputs of the rocker and function buttons

The GetDisplayInfo function block from the EC_Visu2.lib library enables you to control the scanning of the buttons according to the active menu on the controller, → section "EC_Visu.lib/EC_Visu2.lib library", page 62.

Diagnostics inputs

The inputs I13, I14, I15, I16 provide you with additional information:

Input	Function
I13	No function
I14	Expansion device via easyLink (not yet active in the operating system version 1.x): 0: ok, 1: not ok
I15	Outputs Q1, Q2, Q3, Q4: 0: No short-circuit, 1: Short-circuit
I16	Outputs Q5, Q6, Q7, Q8: 0: No short-circuit, toggle: Short-circuit

The inputs can be scanned in the program with symbolic operands.

Inputs for high-speed counters

You can choose between several different functions:

- 1 × 32-bit counter, for counting pulses (up/down)
- 2 × 16-bit counters, for counting pulses (up/down); the count direction (up/down) can be set via the DIRECTION operand in the program.
- 1 × incremental value counter, 32-bit, for processing the signals of an incremental encoder; the count direction is set by the edge sequence of the encoder.

You can select the counter type in the PLC configuration.

The function of the high-speed counter requires the setting of inputs and the scanning of outputs in a POU, e.g. PLC_PRG. This POU must not be called by an interrupt generated by a counter.

For further information see section "High-speed counters (Counter)", page 45.

Outputs

Table 2: Type and number of outputs

EC4P-221/222-MT… transistor outputs	8 (Q1…Q8)	24 V DC/0.5 A
EC4P-221/222-MR… relay outputs	6 (Q1…Q6)	250 V AC/8 A

The transistor outputs are provided with a short-circuit monitoring function. In the event that a short-circuit occurs at one of the outputs, this is indicated via the diagnostics inputs I15/I16. I15 is set to 1 if a short-circuit occurs at the outputs Q1 to Q4. Input I16 is toggled if a short-circuit occurs on Q5 to Q6.

⚠ **Caution!**
Scan I15/I16 in the program. In the event of a short-circuit set the outputs to 0 in order to prevent the thermal overload of the output circuit.

Memory card (MCC)

The memory card is used for data storage and supports the FAT16 file system.

Memory card data

On the memory card you can save the following data:

Data	Transfer method
Boot project	Browser command: copyprojtommc
Startup.INI file	Browser command: createstartupini
Operating system (OS)	Updating the OS, → page 56
Source code of the project	Online mode/Online menu: load source code
General data	Online mode/Online menu: Write file to PLC Load file from PLC

A brief description of the browser commands is provided from page 59.

▽ **Caution!**
In order to avoid any loss of data, ensure that you have closed all files of the program before removing / inserting the memory card or switchingoff the power supply.

Data access on the memory card

Functions such as FileOpen or FileRead allow you to access the files of the memory card from the user program. These functions are provided in the library EC_File.lib and are described in the Function Blocks manual (MN05010002Z-EN; previously AWB2786-1456GB).

RUN/STOP/SF and CAN/NET LEDs

After power up, the CPU can switch to the following states, as indicated by the LEDs:

Table 3: LED status indicator

LED		Meaning/CPU status
RUN/STP/SF	CAN/NET	
red	red[1]	System test being run (up to 6 seconds after start; after 6 seconds if no user program is present). CPU in NOT READY!
orange	orange[1]	System update in progress
red	off[1]	System test finished without error
red flashing	red flashing[1]	System test found a fault
orange	off	No user program present CPU in NOT READY
green flashing	–	Load user program CPU in STOP
green	–	Load user program CPU in RUN
red	–	Cycle time exceeded CPU in STOP
orange flashing	–	Continuous loop detected in program CPU in STOP
red flashing	red flashing	Fatal error

1) LED is only relevant during startup/system test

If the CPU is in RUN status, the CAN/NET LED indicates the following states:

Table 4: LED status signals for CAN/easyNet

LED		Meaning
RUN/STP/SF	CAN/NET	
green	off	Communication not active
green	red	Bus status STOP
green	orange	Bus status PREOPERATIONAL Station can be initialised, no transfer of process data
green	green	Bus status OPERATIONAL Process data transferred

Real-time clock

The PLC is provided with a real-time clock that can be accessed in the user program via functions from the SysLibRTC library. The functions are described in the PDF file „SysLibRTC". After the software is installed, this file can be opened via <Programs → Moeller Software → easySoft CoDeSys → Documentation → Automation Manuals>.

You can read and set with the browser commands "getrtc" and "setrtc" respectively. More information is provided in section "setrtc" on page 60.

During a voltage loss the clock is backed up for at least 72 hours.

Interfaces

Programming interface for connection to a PC

Communication between PLC and the programming device is implemented via the programming interface, consisting of an RJ45 connector.

The connector is provided with an RS232 interface and an additional Ethernet interface on the EC4P- 222-... PLC types for programming.

Table 5:   Signal assignment of the programming interface

RJ45		EC4P-221… Signal RS232	EC4P-222-… Signal RS232	Ethernet
	1	–	–	Tx+
	2	–	–	Tx-
	3	–	–	Rx+
	4	GND	GND	1)
	5	TxD	TxD	–
	6	–	–	Rx-
	7	GND	GND	1)
	8	RxD	RxD	–

1) The GND signal is not required for an Ethernet connection. Therefore use a cable with unassigned terminal pins 4 and 7!

Transparent mode

In order to establish a point-to-point connection (without handshake cables) to another device, switch the RS232 interface to Transparent mode using the functions from the library EC_SysLibCom.lib. In Transparent mode, the interface is addressed as COM1.

→ chapter "RS232 interface in Transparent mode", page 73.

Splitting the RS232/Ethernet interface

Using a cable splitter XT-RJ45-ETH-RS232 you can communicate simultaneously via the RS232 and the Ethernet interface. The connection between the PLC and the cable splitter is established using the EASY-NT-30/80/130 cable. The pin assignment of the RS232 and Ethernet connector socket of the cable splitter corresponds with the pin assignment of the programming interface as shown in table 5.
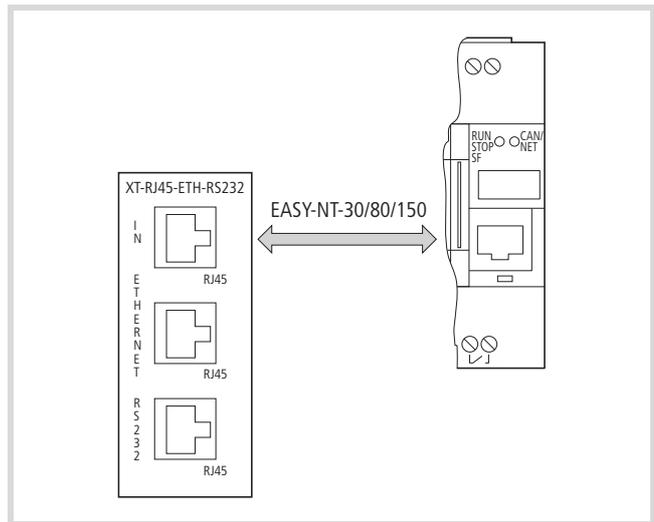
Figure 6:   Connecting the PLC with XT-RJ45-ETH-RS232

See also:

→ section "CAN/easyNet, PC connection", page 27

→ chapter 12 „Connection setup PC – EC4-200", page 63

Multi-function interface (MFI)

The controller can alternatively communicate with the following devices via this interface:

• Memory card
The memory card should be fitted in an adapter which is then fitted on this slot.

• MFD-CP4 multi-function display
The MFD is a display with HMI features that is mounted away from the PLC. It displays the content of the PLC display. Integrated buttons enable you to send signals to the controller and control the processing of the program. The MFD can be mounted in a control cabinet door up to 5 m away from the controller. The devices are connected with the cable MFD-CP4-800-CAB5.

• Terminal/printer
A terminal enables you to display and enter alphanumeric characters. A printer can also be used to output data. The terminal is connected to the PLC via an RS232 interface using the EASY800-PC-CAB cable. The cable with the components for adapting the PLC signals must be provided with a separate power supply from the terminal. The signals and pin assignment of the interface must be implemented in compliance with the RS232 specification.

In order to supply the components in the cable, the RTS signal device must be set in the (terminal) device, → section "easy800-PC-CAB connection cable" on page 95.

The RS232 interface that is addressed with COM2 must be set to Transparent mode in order to send or receive data to or from the terminal.

→ chapter "RS232 interface in Transparent mode" on page 73.

The functions for opening and closing the interface and for sending and receiving data are described in the library EC_SysLibCom.lib.
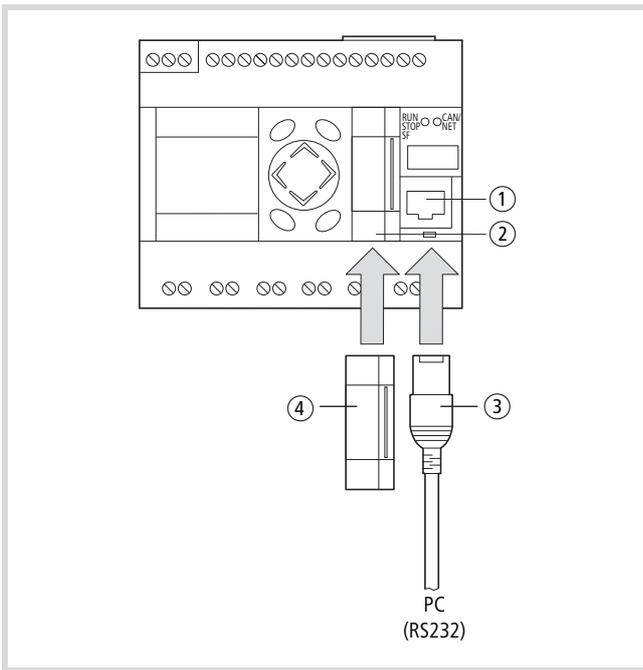
Figure 7: Interfaces

① Programming interface for connection to a PC
② Multi-function interface
③ Programming cable, e.g. EU4A-RJ45-CAB1
④ Adapter with memory card or cable connection

Cable connections

The following overview shows the cable types that can be connected to the PLC and their functions.

Interface	Cable type	Device	Function
RJ45			
RS232	EU4A-RJ45-CAB1	PC, terminal/ printer	Program, transparent mode (COM1)
Ethernet	XT-CAT5-X-2	PC	Program
MFI	MFD-CP4-800-CAB5	MFD-CP4	Display extension
	easy800-USB-CAB	PC	Program
	easy800-PC-CAB	Terminal/ printer	Transparent mode (COM2)
	easy800-MO-CAB	PC, terminal/ printer	Program, transparent mode (COM1)
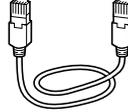
CAN/easyNet interfaces

The PLC is provided with a CAN/easyNET interface with two slots that are internally connected via terminals.
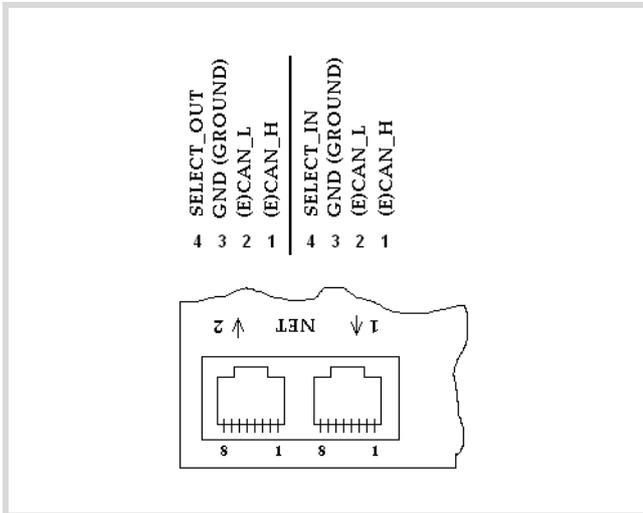
Figure 8: CAN/easyNet interfaces

CANopen
The CAN interface is designed as a CANopen interface in compliance with the CIA specification DS301V4.0. The PLC can be operated both as an NMT master as well as a CAN device on CAN networks. When used as a CAN device the PLC requires an address (= Node ID) for identification on the bus. Permissible node IDs are 1, … ,127. The configuration of the master and the device is carried out in the PLC configuration.

→ section "Network CAN/easyNet", page 93.

3 Expansion units

You connect the expansion devices directly to the PLC via the easyLink interface. The following expansion devices can be used to increase the number of PLC inputs and outputs.

Type overview of expansion devices

Part no.	Supply voltage connection	Inputs	Outputs
EASY618-AC-RE	100 … 230 V AC	12 AC	6 relays
EASY618-DC-RE	24 V DC	12 DC	6 relays
EASY620-DC-TE	24 V DC	12 DC	8 transistor
EASY202-RE	–	–	2 relay outputs with common power supply for several outputs

The EASY200-EASY coupling device enables you to connect a remote expansion device to the controller via a 30 m 2-wire or multi-core cable.

Overview of inputs/outputs
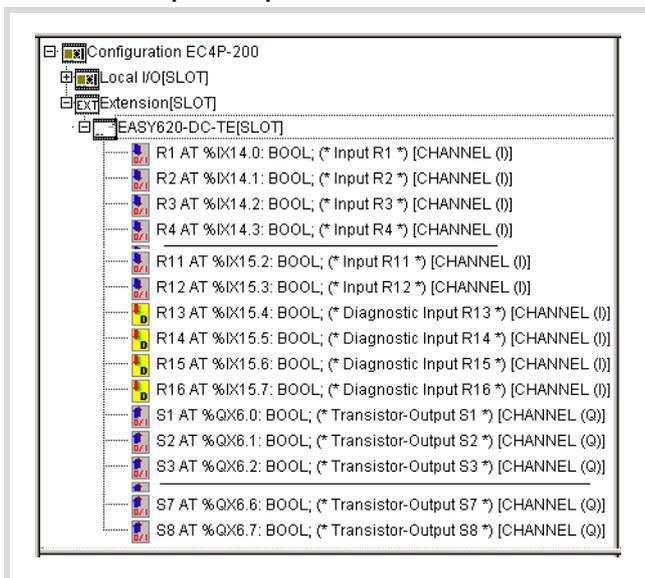
Figure 9:   I/Os of the EASY620-DC-TE

Inputs

Table 6:   Number of inputs and symbolic operands

Part no.	Number	Operand
EASY6…-…-…	12	R1, … ,R12
EASY620-DC-TE	4 (diagnostic)	R13, … ,R16

They are selected in the user program by means of the appropriate syntax used in the PLC configurator.

Diagnostics inputs

The inputs R15, R16 provide you with additional information:

Table 7:   Functions of the diagnostics inputs

Input	Function
R13,R14	No function
R15	Outputs S1, S2, S3, S4: 0: No short-circuit, toggle: Short-circuit
R16	Outputs S5, S6, S7, S8: 0: No short-circuit, toggle: Short-circuit

The inputs can be scanned in the program with symbolic operands.

Outputs

Table 8:   Number of outputs and symbolic operands

Part no.	Number	Operand
EASY618	6	S1,…, S6
EASY620	8	S1,…, S8
EASY202-RE	2	S1,S2

The transistor outputs are provided with a short-circuit monitoring function. In the event that a short-circuit occurs at one of the outputs, this is indicated via the diagnostics inputs R15/R16. R15 is set to 1 if a short-circuit occurs at the outputs S1 to S4. Input R16 is toggled if a short-circuit occurs on S5 to S6.

4 Mounting

Install the PLC in a control cabinet, a service distribution board or in an enclosure so that the power supply terminals and other terminals are protected against direct contact during operation.

The PLC can be installed vertically or horizontally on a top-hat rail in compliance with IEC/EN 60715 or on a mounting plate using fixing brackets.

Ensure that the terminal side has a clearance of at least 3 cm from the wall and from neighbouring devices in order to simplify wiring.
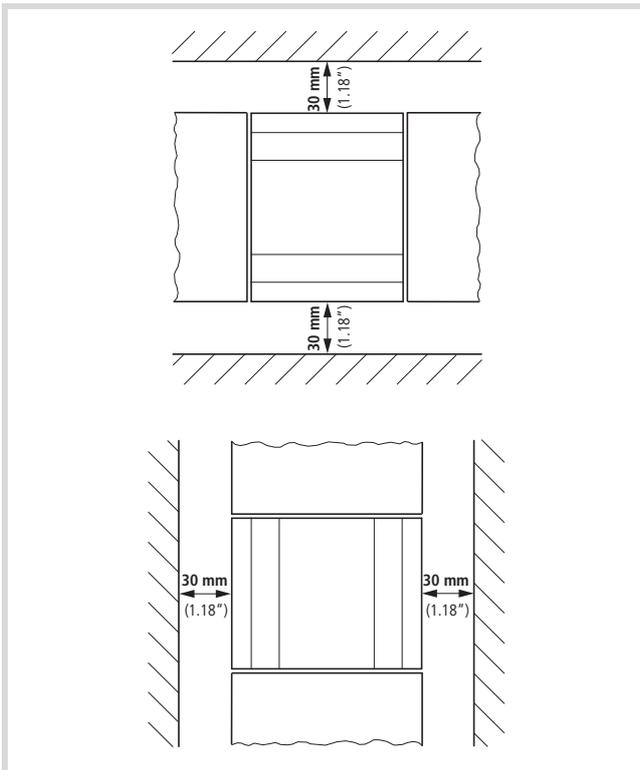
Figure 10: Observing the clearances for wiring

Mounting on top-hat rail

▶ Place the device diagonally on the upper lip of the top-hat rail. Press down lightly on both the device and the top-hat rail until the unit snaps over the lower edge of the top-hat rail. The spring mechanism should ensure that the device snaps into position automatically.
▶ Check that the device is seated firmly.
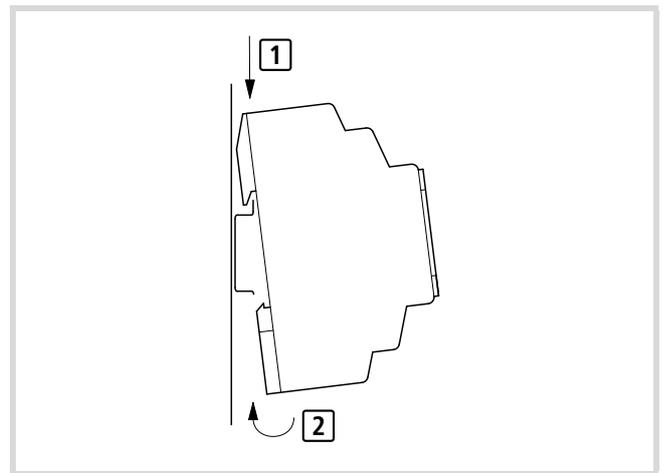
Figure 11: Mounting on top-hat rail

The device is mounted vertically on a top-hat rail in the same way.

Mounting on mounting plate

Fixing brackets that can be inserted on the rear of the device are required for screw mounting. The fixing brackets are available as an accessory.

→ Three fixing brackets are sufficient for a device with four fixing points.
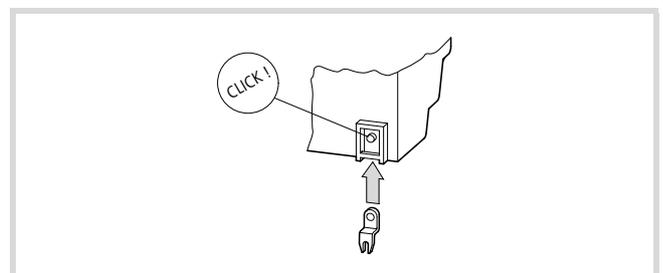
Figure 12: Inserting a fixing bracket
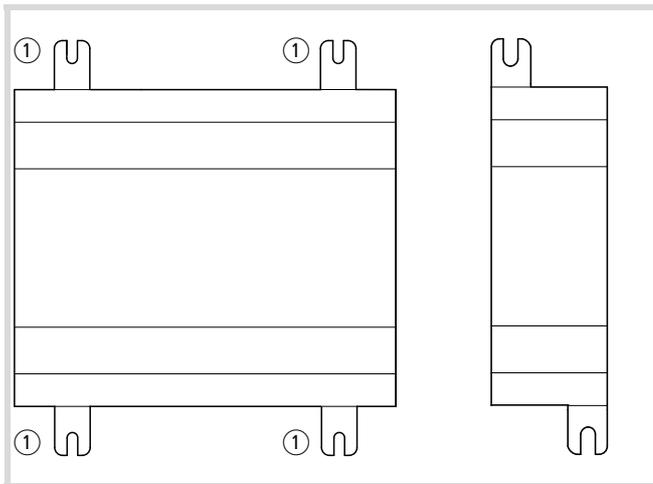
Figure 13: Screw fixing the devices

① Fixing brackets
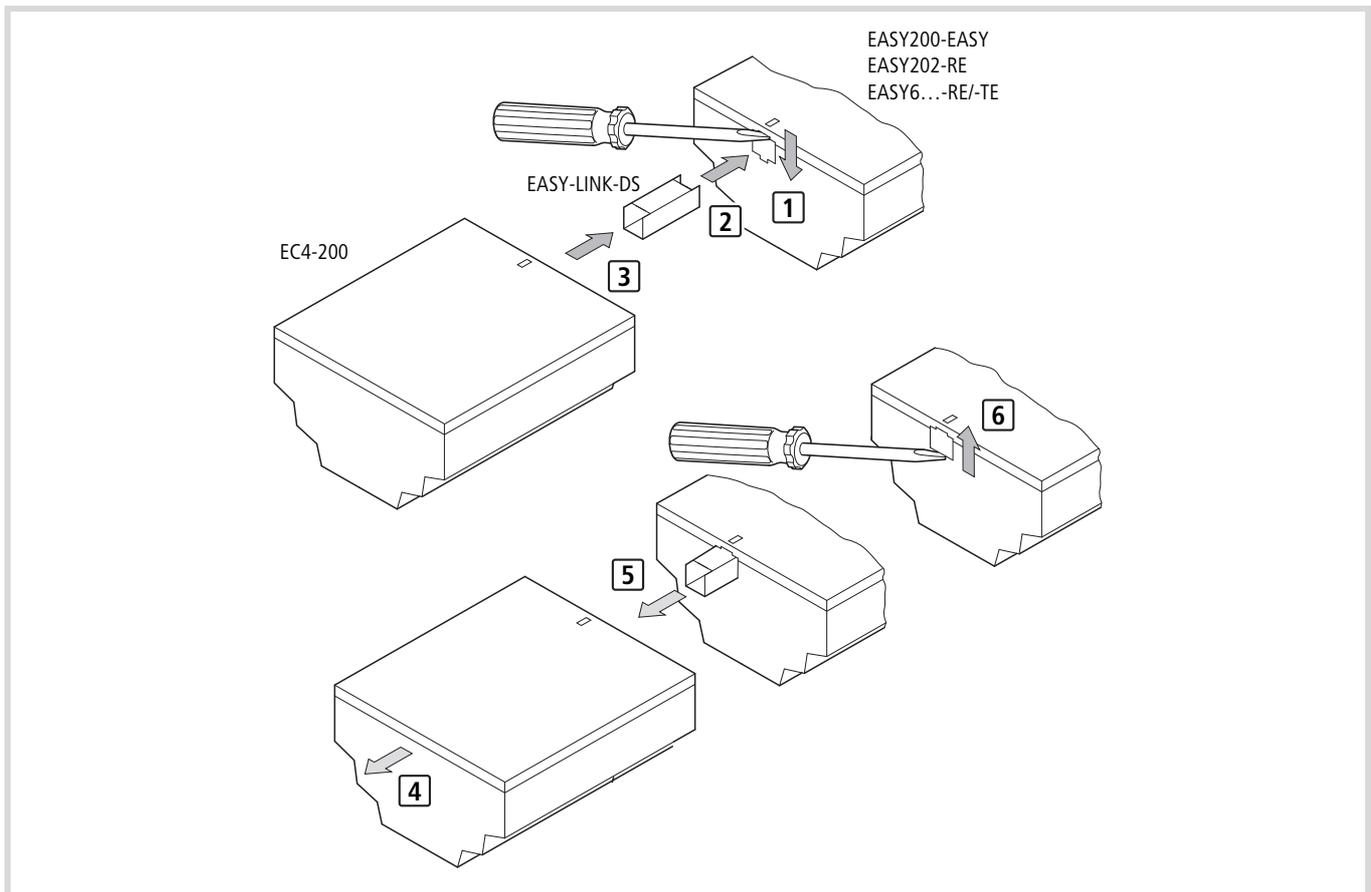
Figure 14: Connecting the expansion unit/network module to the EC4-200

5 Installation

Connecting the power supply
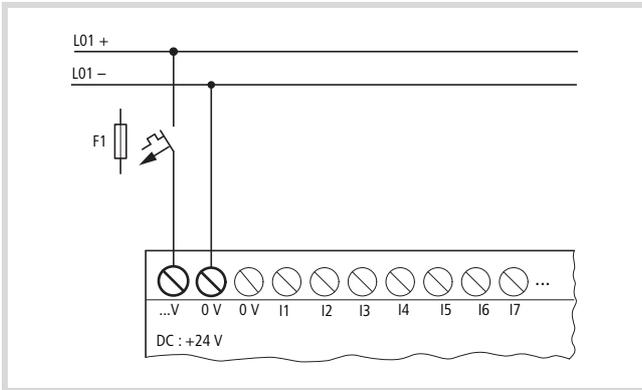
Figure 15: Connecting the power supply
The two 0 V terminals are connected internally!

→   Die EC4-200 is protected against polarity reversal.

→   The necessary connection data is provided chapter "Technical data", page 96.

Cable protection
Protect the supply cables with a miniature circuit-breaker or at least a 1A (slow blow) fuse (F1).

→   The controller behaves like a capacitor the first time it is powered up. The switching device and the supply device for switching on the power supply must be designed for this, i.e. no Reed relay contacts, no proximity switches.

Connecting digital inputs

Use input terminals I3 to I4 to connect pushbutton actuators, switches or 1 or 12-wire proximity switches. Given the high residual current, do not use 2-wire proximity switches.
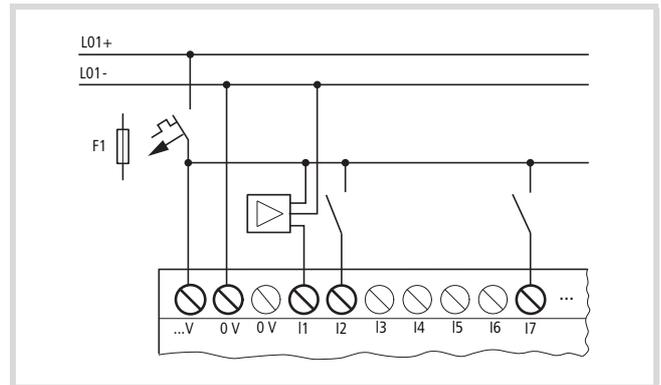
Figure 16: Connecting digital inputs

Connecting analog inputs

Inputs I7, I8, I11 and I12 can also be used to connect analog voltages ranging from 0 V to 10 V.

The resolution is 10-bit = 0 to 1 023.

⚠ **Caution!**
Observe the following when laying and connecting analog cables:

▶ Use screened, twisted pair conductors, to stop interference of the analogue signals.
▶ With short cable lengths, ground the shield at both ends using a large contact area. If the cable length is more than around 30 m, grounding at both ends can result in equalisation currents between the two grounding points and thus in the interference of analog signals. In this case only earth the conductor on one side.
▶ Don't lay the signal conductor parallel to the power conductor.
▶ Connect inductive loads that you are switching via the outputs to a separate power supply or use a suppressor circuit for motors and valves. If the controller is run with motors, solenoid valves or contactors via the same power supply, the switching may cause interference on the analog input signals.

The following circuits show examples of analog measuring applications.

→ Ensure that the reference potential is galvanically connected. Connect the 0 V of the power supply unit for the setpoint potentiometer and various sensors shown in the examples with the 0 V of the power supply.

Setpoint potentiometers connection
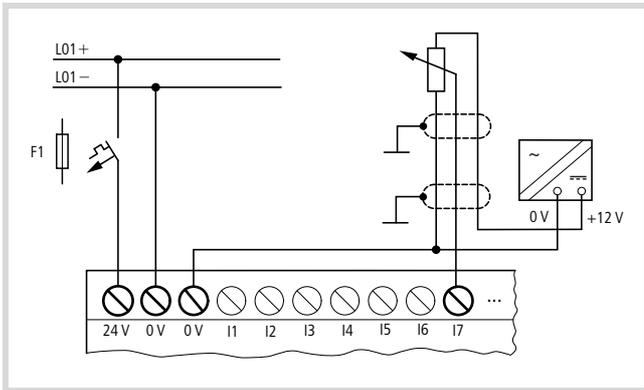
Figure 17: Setpoint potentiometer

Use a potentiometer with the resistance $\leqq$ 1 ≠Ω, e.g. 1 ≠Ω, 0.25 W.
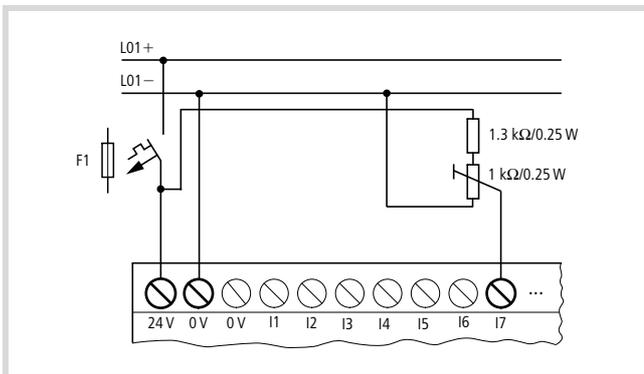
Figure 18: Setpoint potentiometer with upstream resistor
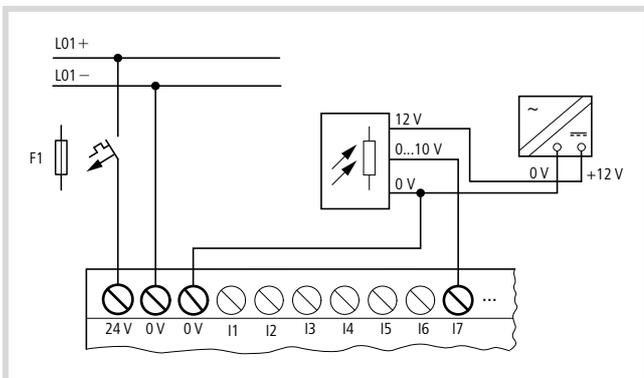
Figure 19: Brightness sensor

Temperature sensor connection
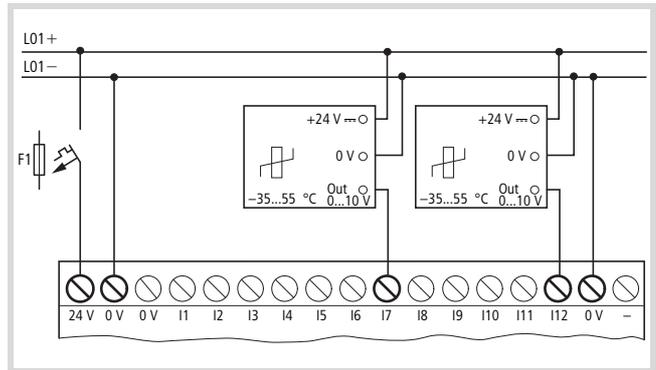
Figure 20: Temperature sensor

Connecting the 20 mA sensor

A 4…20 mA (0…20 mA) sensor can be connected easily with an external 500 Ω resistor.
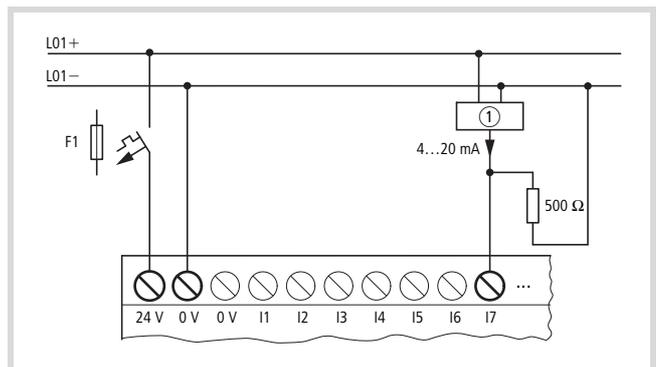
Figure 21: 20 mA sensor

① Analog sensor

The following values apply:

• 4 mA = 1.9 V
• 10 mA = 4.8 V
• 20 mA = 9.5 V
(according to $U = R \times I = 478\ \Omega \times 10\ mA \sim 4.8\ V$)

Connecting a pulse transmitter/incremental encoder

Inputs I1 to I4 are designed so that high-speed signals from pulse transmitters/incremental encoders can be counted.

The following connection options are possible:

- 1 × pulse transmitters (32-bit)
- 2 × pulse transmitters (16-bit)
- 1 × incremental encoder (32-bit).

Connecting pulse transmitter
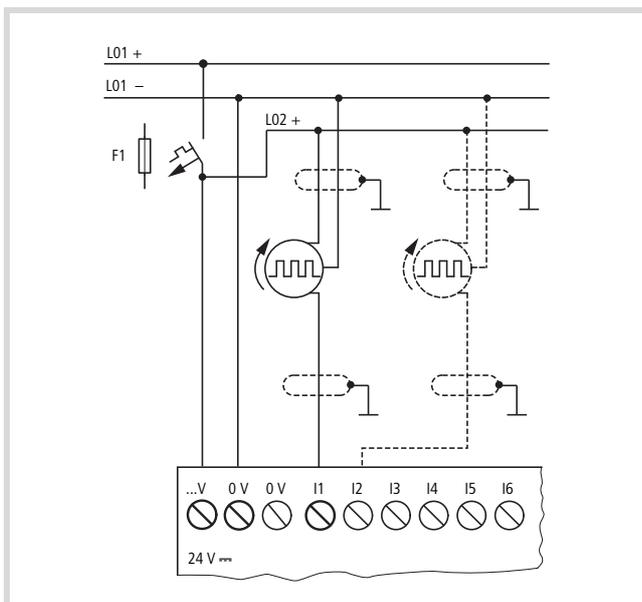
Figure 22:   Connecting pulse transmitter

The figure shows the connection of a pulse transmitter which sends pulses to input I1. An internal counter processes the pulses. You can choose between a 16-bit counter (max. 65535) and 32-bit counter (max. 4294967295). The pulse transmitter for the 32-bit counter must only be connected to I1. Only if a 16-bit counter was used at I1, can another pulse transmitter (32-bit) be connected to I2.

Connecting the incremental encoder
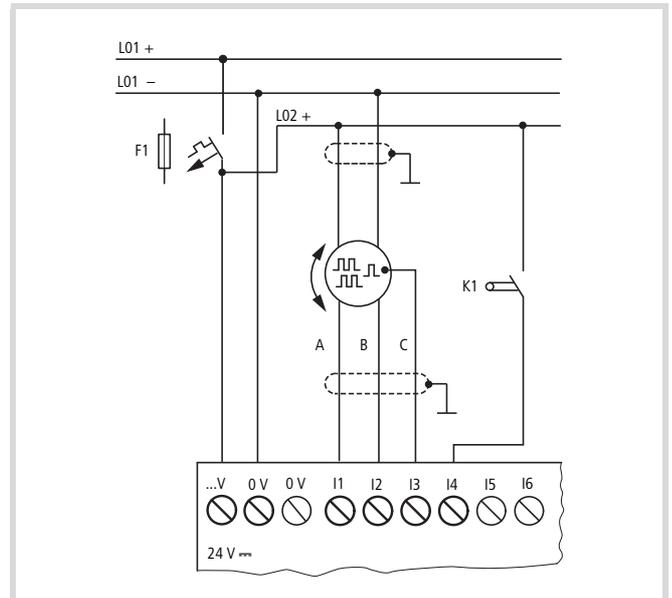
Figure 23:   Connecting the incremental value encoder

A, B: square-wave incremental signals that have a 90 degree phase shift
C: Reference signal
K1: Reference window switch

23

Connecting outputs

The relay or transistor outputs are used to switch loads such as fluorescent tubes, filament bulbs, contactors, relays or motors. Check the technical thresholds and output data before installing such devices (→ page 100, 101).

Connect relay outputs

EC4P-221/222-MR…, EASY6..-DC-RE
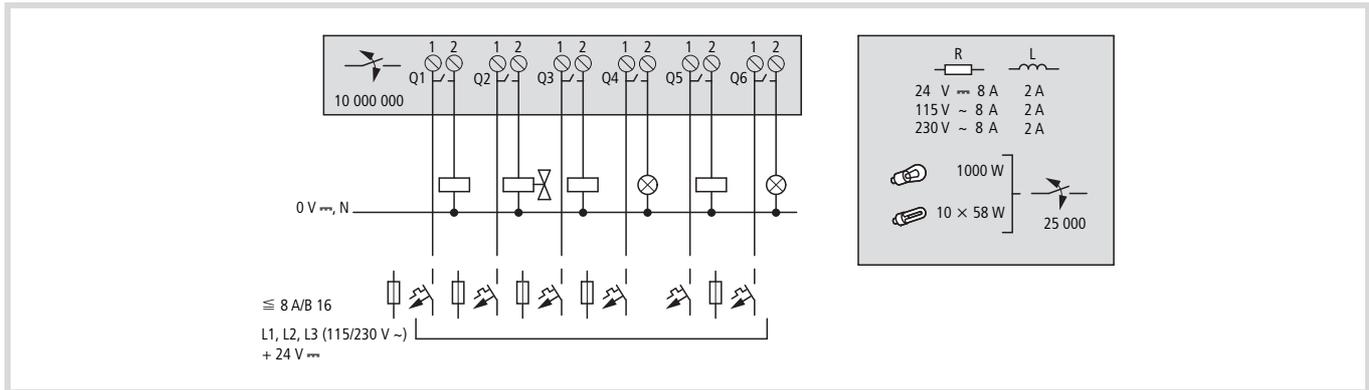
Figure 24:   Relay outputs EC4P-221/222-MR…

Unlike the inputs, you can connect the EC4P-221/222-MR…, EASY6..-..RE relay outputs to different phase conductors.

⚠ **Caution!**
Do not exceed the maximum voltage of 250 V AC on a relay contact. If the voltage exceeds this threshold, flashover may occur at the contact, resulting in damage to the device or a connected load.

Connecting transistor outputs

EC4P-221/222-MT…, EASY6…-DC-TE
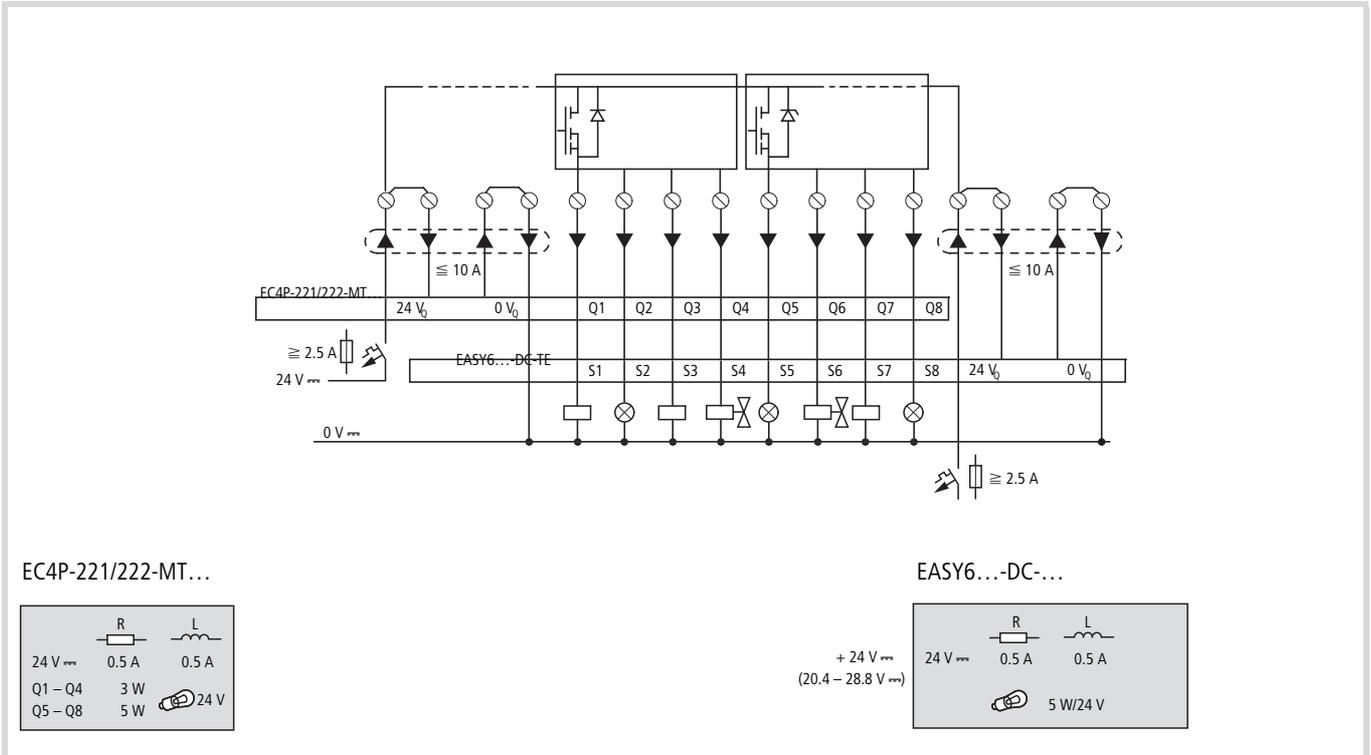
Figure 25: Transistor outpus EC4P-221/222-MT…, EASY6-DC-TE

Parallel connection:
Up to four outputs can be connected in parallel in order to increase the power. This enables a maximum output current of 2 A.

⚠ **Caution!**
Please note the following when switching off inductive loads:
Suppressed inductive loads cause less interference in the entire electrical system. It is generally recommended to connect the suppressor as close as possible to the inductance.

⚠ **Caution!**
Only outputs of the same group (Q1 to Q4 or Q5 to Q8) can be connected in parallel; e.g. Q1 and Q3 or Q5, Q7 and Q8. Outputs connected in parallel must be switched at the same time.

If inductive loads are not suppressed, the following must be observed: Several inductive loads should not be switched off simultaneously to avoid overheating the driver blocks in the worst possible case. If in the event of an Emergency-Stop the +24 V DC power supply is to be switched off by means of a contact, and if this would mean switching off more than one controlled output with an inductive load, then you must provide suppressor circuits for these loads ($\longrightarrow$ following diagrams).
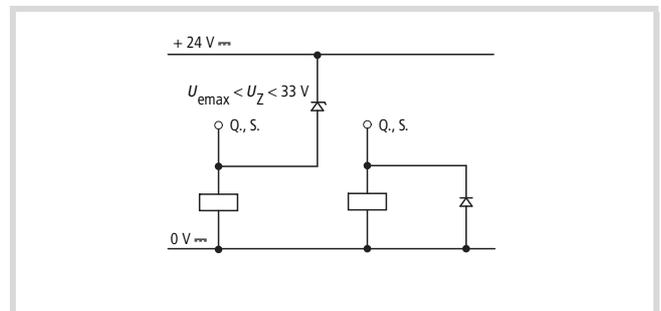
Figure 26: Inductive load with suppressor circuit

Behaviour with short-circuit/overload
A transistor output will switch off in the event of a short-circuit or overload. The output will switch back on up to the maximum temperature after a cooling time that depends on the ambient temperature and the current level. If the fault continues, the output will switch off and on until the fault is rectified or the power supply is switched off.

Connecting the analog output

The EC4-200 is provided with one analog output QA 01, 0 V up to 10 V DC, 10-bit resolution (0 to 1 023). The analog output can be used for controlling servo valves and other actuators.

Connecting servo valve
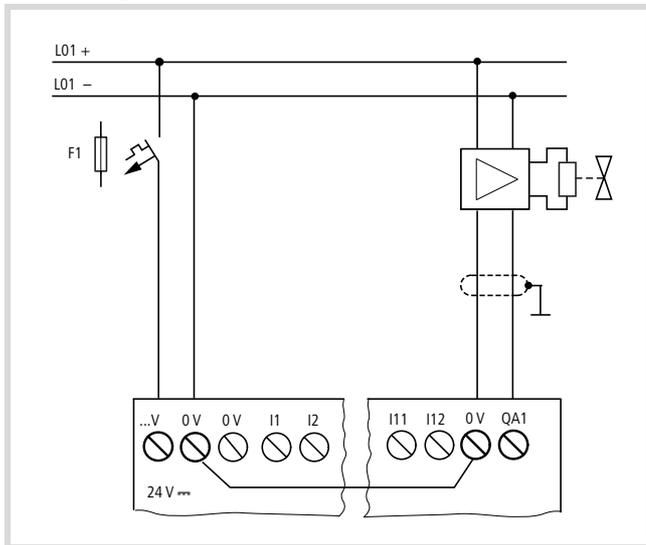
Figure 27:   Connecting servo valves

Setpoint entry for a drive
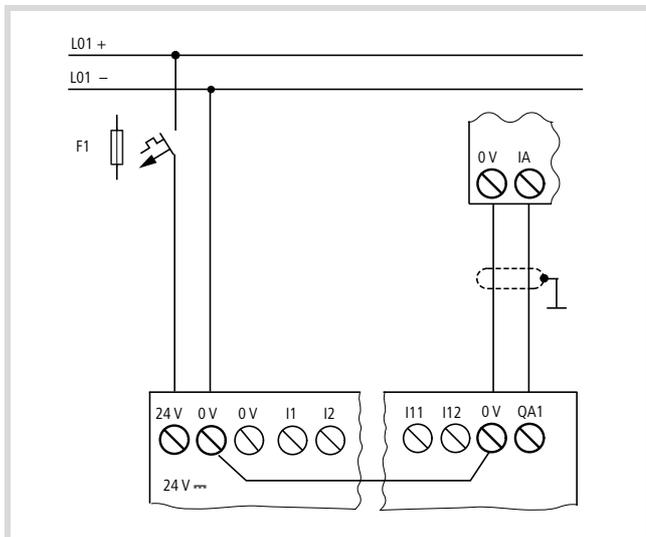
Figure 28:   Setpoint entry for a drive

⚠ **Caution!**
Analog signals are more sensitive to interference than digital signals so that more care must be taken when laying and connecting the signal cables. Incorrect switching states may occur if they are not connected correctly.

Memory card, CAN/easyNet, PC connection

To fit a memory card or establish a CAN/easyNet or PC connection, the protective cap must be removed first of all.
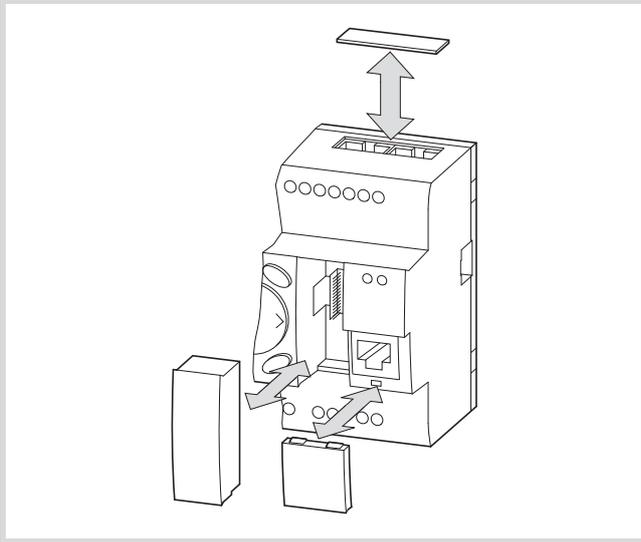
Figure 29: Removing the protective cap/adapter:
top: for CAN/easyNet connection
bottom left: adapter for memory card
bottom right: PC connection

Fitting or removing the memory card

The memory card is located in adapter ③.
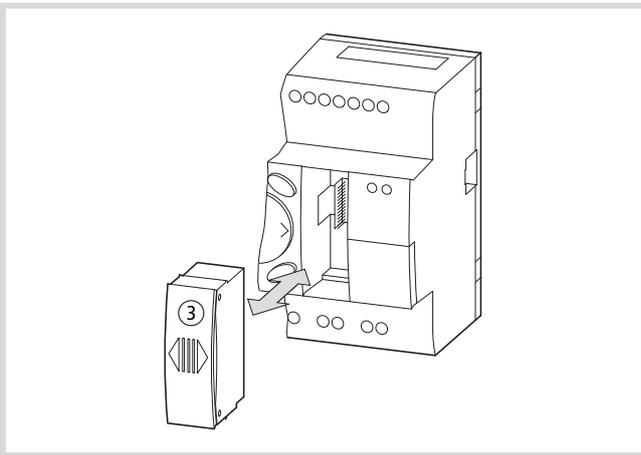
Figure 30: Adapter with memory card

▶ To fit the memory card, press it until it snaps into position.
▶ To remove the memory card, press it until it is released.
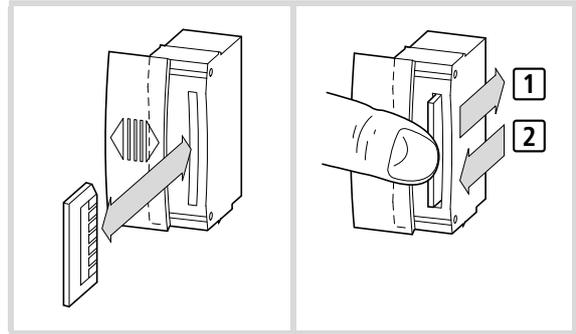
Figure 31: Fitting/removing the memory card

CAN/easyNet, PC connection

▶ Fit the plug for the CAN/easyNet connection into the opening at the top of the device ①.
▶ Fit the plug for the PC connection in the opening on the bottom right on device ②.
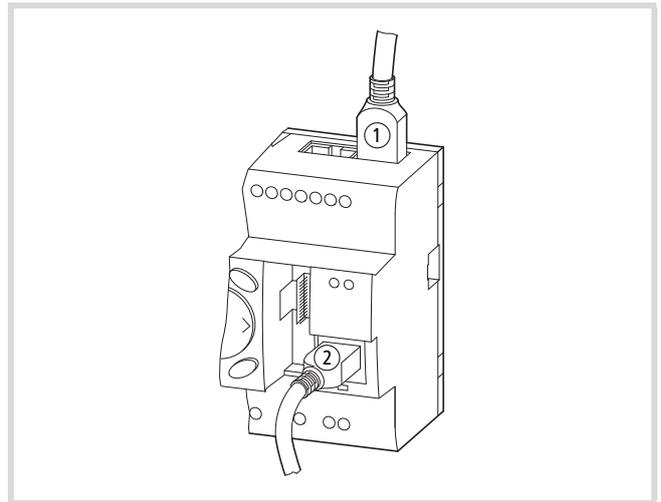
Figure 32: Plugs for CAN/easyNet connection ① and the PC connection ②

→ For further information → section "Network CAN/easyNet", page 93.

⚠ **Caution!**
Protect the EC4-200 and memory card from electrostatic discharge in the following manner: Discharge yourself of electrostatic charge by touching a grounded surface before fitting or removing the memory card.

Connecting expansion devices/network modules

Local expansion

▶ Connect the devices to the expansion or to the network module via the EASY-LINK-DS connection plug.
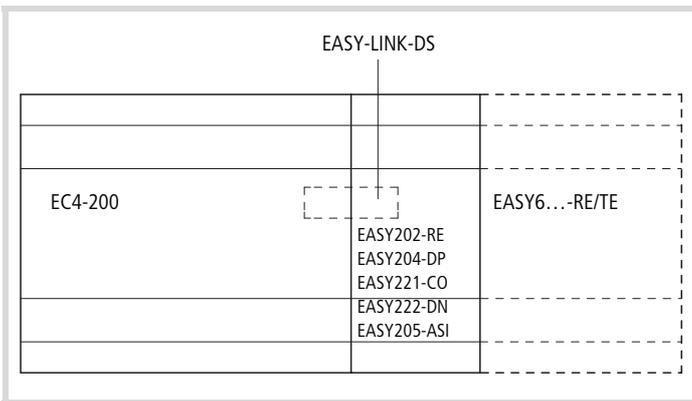
Figure 33:   Connecting expansion devices with EC4-200

⚡ The following electrical separation is implemented between the basic unit and an expansion unit (separation always in local connection of expansion unit).

• Basic isolation 400 V AC (+10 %)
• Safe isolation 240 V AC (+10 %)

Units may be destroyed if the value 400 V AC +10 % is exceeded, and may cause the malfunction of the entire system or machine!

→ Basic unit and expansion unit can be provided with different DC power supplies.

Remote expansion

Remote expansion units can be installed and run up to 30 m away from the basic unit.

⚡ **Warning!**
The two-wire or multiple-wire cable between the devices must comply with the insulation voltage requirement which is stipulated for the installation environment. Otherwise, a fault (ground fault, short-circuit) may lead to the destruction of the units or injury to persons.

A cable such as NYM-0 with a rated operational voltage of U_e = 300/500 V AC is normally sufficient.

→ Terminals E+ and E– of the EASY200-EASY are protected against short-circuits and polarity reversal. Functionality is only ensured if E+ is connected with E+ and E- with E-.

Figure 34:   Connecting remote expansion units to the EC4-200

6 Operation

The following chapter describes the operation of the buttons and the display on the front plate.

Keypad
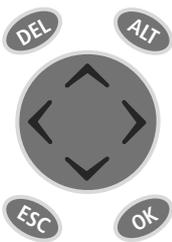
DEL: Delete

ALT: Special function, status display

Cursor buttons ⟨ ⟩ ∧ ∨:
Move cursor
Select menu items
Set numbers and values

OK: Next menu level, Save your entry

ESC: Previous menu level, Cancel

Selecting menus and entering values

DEL and ALT   Show System menu

OK   Move to next menu level Call menu item
Activate, change, store entries

ESC   Move to previous menu level
Cancel entries since last **OK**

∧   Change menu item
∨   Change value
    Change place
⟨ ⟩

P button function:

⟨   Input P1,   ∧   Input P2
⟩   Input P3,   ∨   Input P4

Selecting or toggling between menu items

```
PROGRAM...
STOP        RUN
SET CLOCK...…
INFORMATION...
```
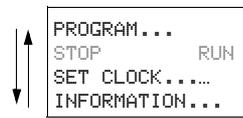
Cursor ∧ ∨

OK   Select or toggle

Cursor display

```
HH:MM   ∎ID4:I23
JJ.MM   05.05
ANNÉE    2003
```

The cursor flashes.

Full cursor ∎✓:

• Move cursor with ⟨ ⟩ ∧ ∨

```
HH:MM   14:23
JJ.MM   05.05
ANNÉE    2003
```
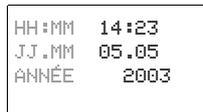
Value M/M

• Change position with ⟨ ⟩
• Change values with ∧ ∨
Flashing values/menus are shown in grey in this manual.

Setting values

```
HH:MM ┌14:23
DD.MM ┌03.10
YEAR  ┌2002
```

Values
Places
Value at place
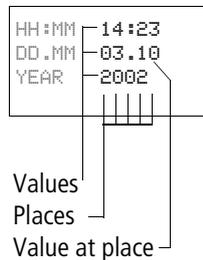
Select value ∧ ∨
Select digit ⟨ ⟩
Change value at digit ∧ ∨

OK   Store entries

ESC   Retain previous value

Choosing the main and system menu

Status display

```
I 12........
NT1SP BP MC DC
MO 02:00    WS
Q ..34…. STOP
```

DEL and ALT

Local
expansion

OK

```
R ..34......
RS SP BP MC DC
MO 02:00    WS
S....56.. STOP
```

ESC

ESC

No
password

R = Inputs.. RS =
Expansion S =
Outputs..

OK

Current
selection
flashes in
menu

```
PROGRAM…
STOP√ RUN
SET CLOCK…
INFORMATION…
```

1st menu level
Main menu

```
SECURITY...
SYSTEM…
STARTPARAMETER..
MENU LANGUAGE...
```

1st menu level
System menu

Status display with time

Inputs

```
I 12………
NT1 SP BP MC DC
MO 02:00   WS
Q ..34…. STOP
```

Weekday/Time
Outputs

On: 1, 2/Off: 3, 4…

System information #)
Startup behaviour: WS = Warm start, CS = Cold start, ST = Stop
Operating state: RUN/STOP

Status display with date

ALT

#) System information

−NT1 = Station 1 (2..8)
−SP = Startup.INI file present
−BP = Boot project present
−MC = Memory card present
−DC = DC/AC expansion present

Inputs

```
I 12………
NT1 SP BP MC DC
MO 01.04.2006 WS
Q ..34…. RUN
```

Weekday/date
Outputs

ESC

Advanced Status display

ALT

ID = Device address of missing
devices on the easyNet

```
I 12..........
ID ...45678 DC
I ..15.R…. WS
Q ..34....RUN
```

I/R = Diagnostics inputs:
I13 = No meaning
I14 = 1, if no Link expansion
I15 = 1, if short-circuit on output Q1, Q2, Q3 or Q4
I16 = toggles if short-circuit on output Q5, Q6, Q7 or Q8
R15 = toggles if short-circuit on output S1, S2, S3 or S4
R16 = toggles if short-circuit on output S5, S6, S7 or S8

Menu structure

Main menu without password protection

▶ You access the main menu by pressing **OK**.

Main menu   * Only use in STOP

```
PROGRAMM…*)  →  BOOT PROJECT      MEMORY->FLASH  →  REPLACE ?      BOOT PROJECT
STOP √ RUN      RESET…             FLASH->CARD                     IS
SET CLOCK…      DELETE             CARD->FLASH                     STORED
INFORMATION
```

```
MEMORY->FLASH  →  REPLACE ?      BOOT PROJECT
FLASH->CARD                      IS
CARD->FLASH                      STORED
```

```
MEMORY->FLASH  →  REPLACE ?      BOOT PROJECT
FLASH->CARD                      IS
CARD->FLASH                      STORED
```

```
BOOTPROJECT   RESET WARM
RESET…        RESET COLD
DELETE
```

```
BOOT          RESET HARD     →  DELETE?
PROJECT.      DELETE CARD
RESET…
DELETE
```

```
RESET HARD     →  DELETE?
DELETE CARD
```

```
PROGRAM
STOP √ RUN
SET CLOCK...
INFORMATION
```

```
PROGRAM        →  SET CLOCK      →  HH:MM  14:23
STOP √ RUN                          DD.MM  03.10
SET CLOCK                           ANNÉE   2001
INFORMATION
```

```
PROGRAM        →  EXAMPLE:
STOP √ RUN        EC4P-222-MTAD1
SET CLOCK         OS :V01.00
INFORMATION      BUILD:1002
```

Main menu with password protection

Main menu

Password entry

Incorrect entry

```
PASSWORD…
STOP RUN √
SET CLOCK…
INFORMATION
```

```
Password
```

```
PASSWORD…
RUN
```

Correct entry

Status display
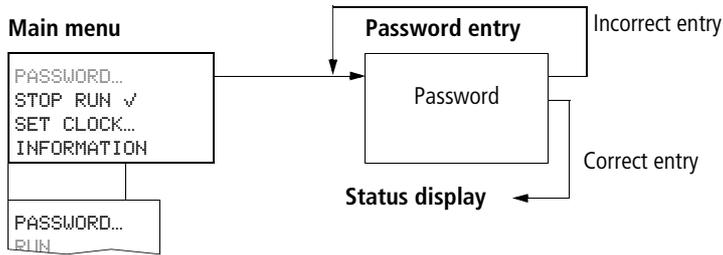
System menu

▶ The System menu is accessed by simultaneously pressing **DEL** and **ALT**.

System menu   Password setup   **Password entry**

```
SECURITY...
SYSTEM…
STARTPARAMETER
MENU LANGUAGE
```

```
PASSWORD…
RANGE…
```

```
ENTER PASSWORD
------
```

```
ENTER PASSWORD
000000
```

Change/
delete
password

```
CHANGE PW
ACTIVATE PW
```

```
ENTER PASSWORD
------
```

```
ENTER PASSWORD
000000
```

```
PASSWORD…
RANGE…
```

```
PROGRAMM √
CLOCK
OPERATING MODE
```

```
SECURITY…
SYSTEM…
STARTPARAMETER
MENU LANGUAGE..
```

```
RESTART…
DISPLAY…
FACTORY SET
```

```
HALT √
RESET WARM
RESET COLD
```

Setting is valid until the next
program download

```
RESTART
DISPLAY
FACTORY SET
```

```
CONTRAST: 0
LIGHTING √
```

```
RESTART
DISPLAY
FACTORY SET
```

```
DELETE ALL?
```

Program + boot project are deleted!
→ section "Restoring factory settings (factory set)",
page 44.

```
SECURITY…
SYSTEM…
STARTPARAMETER
MENU LANGUAGE
```

```
IN
TO BE
DEFINED
```

```
SECURITY…
SYSTEM…
STARTPARAMETER
MENU LANGUAGE
```

```
ENGLISH
DEUTSCH √
```

System menu

System menu   PLC: STOP!

```
SYSTEM…              NET...…           NET PARAMETER...    NET-ID :01
STARTPARAMETER       LINK...…          STATIONS…           BAUDRATE:125KB
MENU LANGUAGE                          CONFIGURE           BUSDELAY: 00
CONFIGURATOR                                               SEND IO : √
```

```
REMOTE RUN
```

Position within easyNet   ID of active devices
                                           0 = No device

```
NET PARAMETER...     1 1
STATIONS…            2 2
CONFIGURE            3 3
                    4 0
```

```
UP TO
8 0
```

```
NET PARAMETER...     CONFIGURE?
STATIONS…
CONFIGURE
```

(OK)

```
CONFIGURATION
IN
PROGRESS
```

```
SYSTEM…              NET...           NOT ACTIVE
STARTPARAMETER       LINK...…
MENU LANGUAGE
CONFIGURATOR
```

7 Description of settings

All settings are made using the operating elements on the controller.

Password protection

You can protect access to the main menu and the System menu, the clock setting and the operating mode (RUN/STOP) with a password. Choose <SECURITY → RANGE> to activate the individual setting options.

The System menu is always protected when a password is activated.

In this case the password consists of a value between 000001 and 999999. The number combination 000000 is used to delete a password.

Password setup

A password can be set up via the System menu in either RUN or STOP mode. You cannot change to the System menu if a password is already activated.

▶ Press **DEL** and **ALT** to call up the System menu.
▶ Select the menu option SECURITY… to enter the password.
▶ Press the **OK** button and move to the PASSWORD… menu.
▶ Press **OK** again to enter the Password entry mode.

Six dashes will appear if no password is entered: No password present.

```
ENTER PASSWORD
■-----
```

▶ Press **OK**, six zeros will appear
▶ Set the password using the cursor buttons:
  – ⟨⟩ select position in the password,
  – ∧∨ set a value between 0 to 9.

▶ Save the new password by pressing **OK**.

```
ENTER PASSWORD
  000042
```

Use **OK** to exit the password display and proceed with **ESC** and ∨ to the RANGE… menu.

The scope of the password has not yet been defined. The password is now valid but not yet activated.

Selecting the scope of the password

▶ Press the **OK** button.
▶ Select the function or the menu to be protected.
▶ Press the **OK** button in order to protect the function or menu (tick = protected).

```
PROGRAM √
CLOCK
OPERATING MODE
```

→ The password protection protects the program by default.

At least one function or menu must be protected.

• PROGRAM: The PROGRAM menu is protected.
• CLOCK: Date and time are protected with the password.
• OPERATING MODE: The toggling of the RUN or STOP operating mode is protected.

Activating the password

You can activate an existing password in three different ways:

• Automatically when the controller restarts
• Automatically after the program is loaded
• Automatically if no telegram was sent on the PC interface for 30 minutes after password entry.
• via the password menu

▶ Press **DEL** and **ALT** to call up the System menu.
▶ Open the password menu via the SECURITY… menu

The password menu is only displayed if a password is present.

```
CHANGE PW
ACTIVATE
```

The password protection protects the program by default.

→ Make a note of the password before you activate it. If the password is no longer known, it will not be possible to activate the System menu.

▶ Select ACTIVATE PW and press **OK**.
The password is now active. The status display is activated.

You must enter the password before you can activate a protected function or menu, or activate the System menu.

Access with password protection

Password protection is deactivated once the password is entered. You can reactivate password protection later via the Password menu or by switching the power supply off and on again.

▷ Press **OK** to switch to the main menu.

The PASSWORD… entry will flash.

▷ Press **OK** to enter the password entry menu.

```
PASSWORD…
STOP  RUN  √
PASSWORD…
SET CLOCK…
```

→ If the main menu shows PROGRAM… instead of PASSWORD…, this means that password protection is not activated.

The password entry field is shown.

```
ENTER PASSWORD
   XXXXXX
```

▷ Set the password using the cursor buttons.
▷ Confirm with **OK**.

If the password is correct, the Status display is reactivated.

The PROGRAM… menu item is enabled.

The System menu is also accessible.

```
PROGRAM…
STOP
PARAMETER
SET CLOCK…
```

Changing or deleting the password range

▷ Enter your password.
▷ Press **DEL** and **ALT** to call up the System menu.
▷ Open the password menu via the menu option SECURITY… and PASSWORD…

The CHANGE PW entry will flash.

This menu is only displayed if a password is present.

```
CHANGE PW
ACTIVATE PW
```

▷ Press **OK** to enter the password entry menu.
▷ Press **OK** to move to the 6-digit entry field.
▷ The current password will be displayed.
▷ Modify the six password digits using the cursor buttons.
▷ Confirm with **OK**.

```
ENTER PASSWORD
   XXXXXX
```

Press **ESC** to exit the security area.

Delete

Use number combination 000000 to delete a password.

Six dashes will appear if no password is entered.

```
ENTER PASSWORD
   100005
```

```
ENTER PASSWORD
   ------
```

Password incorrect or no longer known

Have you entered an incorrect password?

▷ Re-enter the password.

```
ENTER PASSWORD
   XXXXXX
```

This can be repeated as many times as required!

Pressing ESC returns you to the starting menu

⚠ If you have forgotten the password, you can only call the "factoryset" browser command. The password, the user program and the boot project will then be deleted, and the controller will be reinitialised with the default parameters, → section "Reset", page 44.

Changing the menu language

Two menu languages can be selected. These can be set via the System menu.

Language	Display
English	ENGLISH
Deutsch	DEUTSCH

→ The language selection is only available if the controller is not protected by a password.

▶ Press **DEL** and **ALT** to call up the System menu.
▶ Select MENU LANGUAGE… to change the menu language.

The language selection for the first entry ENGLISH is displayed.

```
ENGLISH
DEUTSCH    √
```

▶ Use ⌃ or ⌄ to select the new menu language.
▶ Confirm with **OK**. The language will be assigned a tick.
▶ Exit the menu with **ESC**.

The new menu language is active.

Press **ESC** to return to the Status display.

Setting date and time

The devices are provided with a real-time clock with date and time. Set the hour, minute, day, month and year during initial commissioning.

▶ Select SET CLOCK… from the main menu.

This will open the menu for setting the time.

```
SET CLOCK
```

▶ Select SET CLOCK.

▶ Set the values for time, day, month and year.
▶ Press the **OK** button to access the Entry mode.
  – ⟨ ⟩ Move between the parameters
  – ⌃⌄ Change the value of a parameter
  – **OK** Save day and time
  – **ESC** Retain previous setting.

```
HH:MM: 00:27
DD.MM 05.05
YEAR : 2002
```

Press **ESC** to leave the time setting display.

Startup behaviour

Setting the startup behaviour

The following start options can be set via the menu:

• HALT
• WARMSTART
• COLDSTART

▶ Switch to the System menu.

→ If the controller is password-protected, the System menu is only available after the password has been entered (→ section "Access with password protection",page 36

▶ Set the Startup behaviour.

Setting LCD contrast and backlight

The background illumination of the LCD display can be switched off. The display contrast can be set to one of five stages. The display is not required during operation. The backlight is only required during maintenance and when texts have to be displayed.

▶ Switch to the System menu.

→ If the controller is password-protected, the System menu is only available after the password has been entered (→ section ″Access with password protection″ page 36.

▶ Select the SYSTEM menu.
▶ Press the **OK** button.

```
SECURITY
SYSTEM…
STARTPARAMETER..
.
```

▶ Use the ⌄ button to select the DISPLAY menu and press **OK**.

```
RESTART…
DISPLAY…
FACTORY SET
```

The menus for setting the contrast and backlight are displayed.

```
CONTRAST      0
LIGHTING      √
```

▶ Press the **OK** button and move to the contrast entry field.

Use the ⌃ and ⌄ cursor buttons to set the contrast to a value between −2 and +2.

```
CONTRAST:     +1
LIGHTING      √
```

▶ Select your setting.

▶ Complete your setting by pressing **OK**.

```
CONTRAST:    +1
LIGHTING      √
```

The contrast setting is valid until it is changed again.

▶ Use the cursor buttons ⌃ and ⌄ to move to the LIGHTING menu.
▶ Press the **OK** button.

```
CONTRAST:     +1
LIGHTING      √
```

▶ The backlight is deactivated.

```
CONTRAST:     +1
LIGHTING
```

▶ Press **OK** if you wish to reactivate the backlight.
▶ The tick √ indicates that the backlight has been switched on.

```
CONTRAST:     +1
LIGHTING      √
```

→ The basic factory setting is as follows:
The contrast is set to 0.
The backlight is permanently switched on. Menu setting:
`LIGHTING √`

8 Configuration of the inputs/outputs (I/O)

Representation of the inputs/outputs in the configuration

The direct addresses of the inputs/outputs are assigned symbolic names beforehand in the PLC configuration.

Symbolic operand	Physical operand	Data type
I1	AT % IX0.0	BOOL

The symbolic operands can be used directly in the program.

Displaying the local inputs/outputs

▶ To display the local inputs/outputs, first click on the plus sign in front of "Configuration EC4P-200", then on the plus sign in front of "Local I/O".
▶ The following folders are displayed underneath the folder "Local I/O". The function of these folders must be adapted to the actual controller type.

- Transistor Outputs
- No Analog Outputs
- No Keys
- No Counter.

The folder function "Transistor Outputs" must be changed to "Relay Outputs" for a controller with relay outputs
→ section "Changing the folder function".

▶ To display the inputs/outputs of the individual folders use the mouse to left-click the plus sign in front a folder.
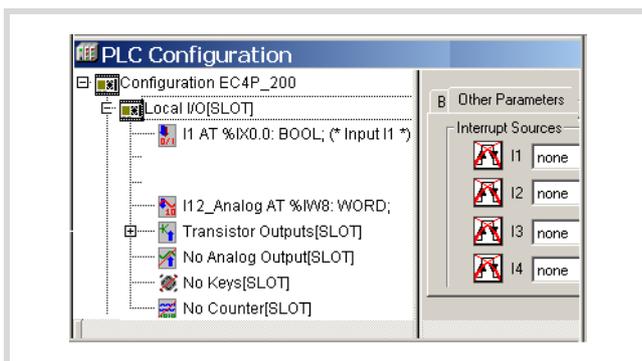
Figure 35:   Selectable inputs/outputs

Changing the folder function

Transistor Outputs ↔ Relay Outputs
The Transistor Outputs are displayed as the default PLC configuration. If you are using a controller with relays, you will have to change the Output Type:

▶ Right-click Transistor Outputs.
▶ Choose Replace Elements in the context menu and click Relay Outputs.

General principle: To display the direct and symbolic addresses of the outputs, click the xxx Output node.

No Analog Output
The default configuration "No Analog Output" can be replaced with "Analog Output" from operating system V2.0

No Keys
The term "Keys" stands for the buttons on the front of the controller such as ALT, DEL, ESC and OK, as well as the 4 buttons of the rocker switch. The buttons are represented in the PLC configuration as inputs.

You can program the direct or symbolic addresses of the inputs in order to scan the states of buttons in the program.

▶ Right-click NoKeys.
▶ Choose Replace Elements in the context menu and click Keys.
▶ Click on the plus sign in front of Keys.

No Counter
High-speed counters must be activated if they are required for your application:

▶ Right-click No Counters.
▶ Choose Replace Elements in the context menu and click on one of the 3 counter functions.
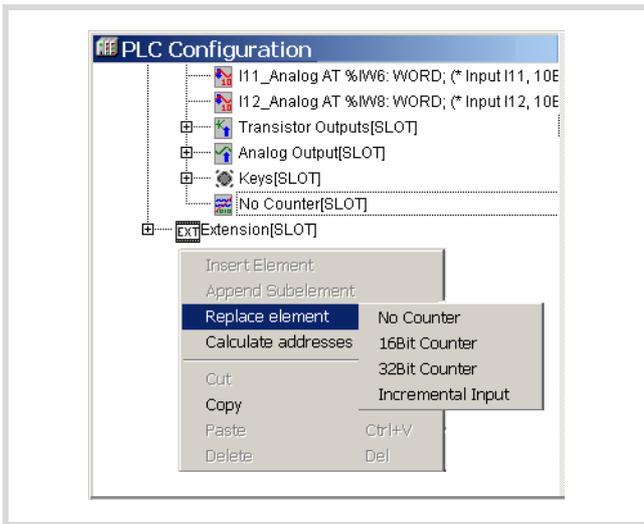
Figure 36: Selecting counters

The submenu will appear:

▶ Select a counter type, such as 32-bit counter.
▶ No Counter will then be replaced by "32 Bit Counter".
▶ Clicking the plus sign will display the inputs and outputs of the counter.
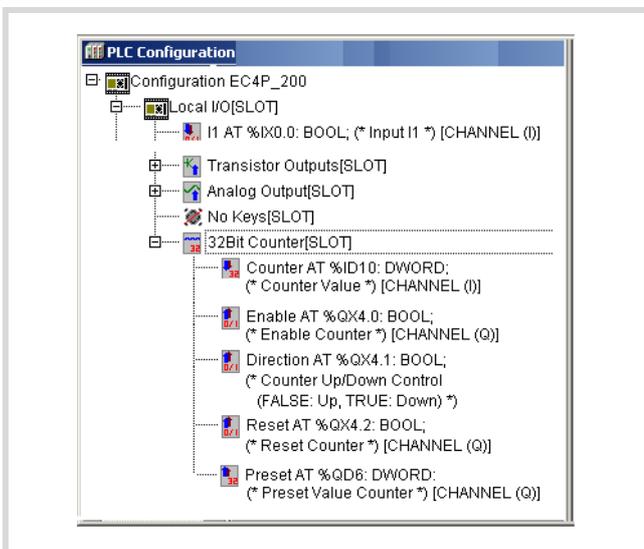
Figure 37: Configuring a counter (32-bit)

Displaying the inputs/outputs of the expansion devices

▶ Click the "+" sign in front of the folder "Extension".
▶ Right-click the "No Extension" folder
▶ Select a device from the Replace element menu.
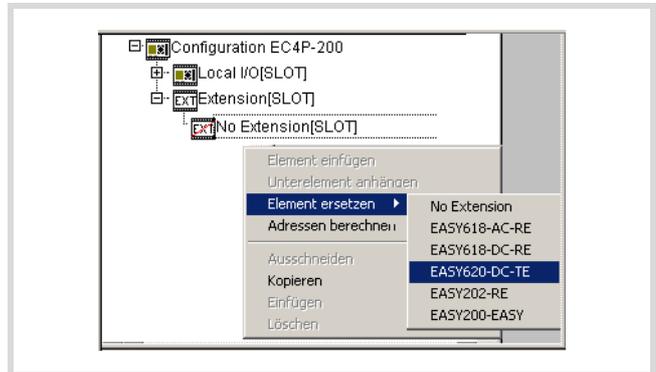
Figure 38: Selecting the expansion device

▶ Click the plus sign in front of the new device folder in order to display the inputs and outputs, including the diagnostics inputs.

9 Operation

General

Overview of memory sizes

The following maximum memory/POUs are available:

Program (Code)	256 kByte
Global variables (Global)	224 kByte
Data memory (Memory)	16 kByte
Input image (Input)	4 kByte
Output image (Output)	4 kByte
Retentive variables (Retain)	8 kByte
Max. number of POUs	Approx.2000

→　If a variable is declared as RETAIN in a function block (FB), all the variables in this function block have the RETAIN status.

Memory definition

The controller has the following memory:

- Working memory (SRAM), not retentive.
  - Content, e.g. program, data
- System memory (FLASH), retentive.
  - Content, such as boot project
- Memory card
  - Content such as boot project, operating system.

Startup behaviour

The controller does not have a battery for backing up the working memory containing the program. To save the program in the event of a power failure, you should create a boot project of this program that can be stored in the retentive system memory.

After the power supply is switched on, the CPU carries out a self-test of the system. In the event of a fault, the LEDs RUN/STOP/SF and CAN/NET LEDs will flash red. After the self-test has been completed fault free, the controller checks whether:

- an operating system update is present on the fitted memory card. In this case, it must be loaded.
- a boot project is present. In this case it is loaded into the working memory of the controller and started according to the startup behaviour set. If no boot project is present, the controller stays in the NOT READY state.

Startup behaviour with boot project on the memory card

When the controller is switched on, a boot project on the memory card has priority over a project stored in the system memory.
If both projects are different, the boot project of the memory card is copied to the system memory and then started. Due to the copy process the PLC start-up phase will be extended by a few seconds.

```
            ┌─────────────────────────┐
            │        Power on          │
            └─────────────────────────┘
                         │
                         ▼
              ╱─────────────────────╲          No
             ╱   Boot project on MMC? ╲─────────────────────────────┐
             ╲                        ╱                              │
              ╲─────────────────────╱                               │
                    │ yes                                           │
                    ▼                                               ▼
        ╱───────────────────────╲       No            ╱───────────────────────╲      No
       ╱   Boot project in the    ╲──────────┐       ╱   Boot project in the    ╲───────────┐
       ╲   system memory (Flash)? ╱          │       ╲   system memory (Flash)? ╱           │
        ╲───────────────────────╱            │        ╲───────────────────────╱            │
                    │ yes                     │                   │ yes                      │
                    ▼                         │                   │                          │
        ╱───────────────────────╲            │                   │                          │
       ╱      Boot project         ╲   No     │                   │                          │
      ╱ on MMC = Boot projet in the ╲─────┐   │                   │                          │
      ╲        main memory         ╱      │   │                   │                          │
       ╲        (Flash)?          ╱       │   │                   │                          │
        ╲───────────────────────╱        │   │                   │                          │
                    │ yes                 ▼   ▼                   ▼                          │
   ┌────────────────────────┐  ┌──────────────────────────┐  ┌──────────────────────────┐   │
   │ Load boot project from │  │ Load boot project from   │  │ Load boot project from   │   │
   │ system memory (Flash)  │  │ the MMC into the working │  │ the system memory        │   │
   │ into the main memory   │  │ memory                   │  │ (Flash) into the working │   │
   │                        │  │ Save boot project from   │  │ memory                   │   │
   │                        │  │ the MMC in system        │  │                          │   │
   │                        │  │ memory (Flash)           │  │                          │   │
   └────────────────────────┘  └──────────────────────────┘  └──────────────────────────┘   │
```

Figure 39:   Startup behaviour with boot project

Setting the startup behaviour in the programming software

With the setting of the start-up behaviour you determine the start behaviour of the PLC when the supply voltage is switched on.

The setting can be made in the PLC configurator or via the operating elements of the controller. The setting options are not prioritised. The last entry is valid.

Activate the Common Parameters tab in the PLC configurator and choose the required startup condition from the list.

- HALT
- WARMSTART
- COLDSTART.
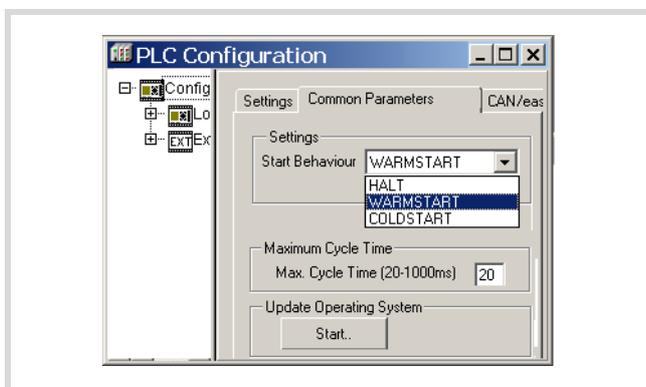
Figure 40:   Definition of start behaviour

Program START/STOP

Program start (STOP → RUN)

You can start the program in two ways:

- In online operation, issue the START command, for example after loading a program.
- Via the operating elements on the controller.
  – In the main menu choose START in the Program menu.

Behaviour after shutdown/interruption of the power supply

If the power supply is switched off or interrupted, this will immediately stop the program cycle. The program is no longer processed up to the end of the cycle. This is also not resumed after the power is restored. Processing starts at the beginning of the program. This can cause retentive data, such as variables in double word format to be no longer consistent depending where the program was aborted.

If inconsistent data is not permissible for an application, you can use an uninterruptible power supply with a battery backup.

In the event of a power failure, all outputs are set to 0 and switched off.

The behaviour of retentive variables according to the startup behaviour set is shown in table 9.

The PLC restarts as defined by the settings in the PLC Configuration window, → figure 40.

Table 9:   Behaviour of variables on startup

Start-up conditions	Variable type	
	Non-retentive	Retentive
COLDSTART	Activation of the initial values	
WARMSTART	Activation of the initial values	Values remain in memory
Program loaded and started in online operation	Activation of the initial values	
Start/Stop/Start…	Values remain in memory	

Program stop (RUN → STOP)

You can stop the program in one of two ways:

- In online operation, issue the STOP command.
- Via the controller menu.
  – In the main menu choose STOP in the Program menu.

If you activate the STOP command, the CPU will switch to STOP status as soon as the program cycle has been completed. The outputs are set to 0.

Starting/stopping the program via external switch

An external switch connected to an input can be used to start or stop the processing of the program. Some additional program instructions are required, which are shown in the example in the Appendix (→ page 94). The SysLibPlcCtrl.lib library contains the function SysStartPlcProgram required for the start, and the function SysStopPlcProgram required for the stop.

In this case, the startup behaviour of the controller must be set to WARM START in the PLC Configurator under <Other Parameters → Settings>!

It is then still possible to switch the controller to START or STOP via the PC in Online mode.

Program processing and system time

The user program is processed cyclically. The states of the inputs are read before the start of each program cycle, and the output states are written to the physical outputs at the end of the cycle.

The cycle time depends on the length and the structure of the user program. In order to ensure a fast response to events, program routines can be programmed that are started when a system event occurs. See → section "System events" on page 49.

Cycle time monitoring

The cycle time of the user program is monitored. The controller switches to STOP status and the outputs are switched off if the cycle time exceeds the set time.

You should set the maximum permissible time in Other Parameters in the PLC configurator: between 20 ms (default value) and 1 000 ms.

Reset

You can carry out a reset via the PC in online mode or via the controller menu. To do this, select the appropriate menu item in the PLC configurator or in the controller menu.

The following Reset commands are provided in the menu:

Configurator (Online menu)	PLC menu
Warm reset	Warm reset
Cold reset	Cold reset
Hard reset	DELETE-> HARD RESET

Warm reset

- The program is stopped.
- The non-retentive variables are initialised, the "Retain" variables are retained.
- The program can be restarted.

Cold reset

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

Hard Reset

- The program in the working memory and the boot project in the system memory of the controller are deleted.
- With the memory card fitted:
  – All the project files on the memory card, the operating system and the boot project are deleted.
  – All user specific files and the startup.ini file remain unchanged
- The PLC is set into the NOT READY state.

Restoring factory settings (factory set)

The browser command "factoryset" or choosing <SYSTEM → FACTORY SET> can start a Hard reset command (→ section "Hard Reset"). The startup.ini file on the memory card and the system parameters in the controller are also deleted. After a start the controller resumes operation with the STARTUP data. The interfaces are initialised with their default values.

Behaviour of variables after Reset

| Reset | Variable type | |
	Non-retentive	Retain
Warm reset	Activation of initial values	Values remain in memory
Cold reset	Activation of the initial values	
Hard reset[1]	No more variables present, program deleted	

1) After a hard reset, the program must be reloaded. In online operation, you can then restart the PLC.

Test and commissioning

The controller supports the following test and commissioning functions:

- Breakpoint/single-step mode
- Single cycle mode
- Forcing
- Online-Änderung
- Progression display (Power Flow).

→ The following applies to breakpoint/single-step mode and single cycle mode:

Do not use these commissioning functions in the program routines such as start. A malfunction may cause an undefined state in the controller.

If the commissioning functions cannot be run, activate the debugging function (default status): Choose Project → Options → Build and click the Debugging option.

Breakpoint/single-step mode

You can set breakpoints within the user program. If an instruction has a breakpoint attached, then the program will halt at this point. The program can be then executed in the single-step mode. Cycle time monitoring is deactivated.

⚠ **Caution!**
At this moment any outputs set will remain set!

Single-cycle mode

In single-cycle operation, one program cycle is performed in real time. The outputs are enabled during the cycle. The cycle-time monitoring is active.

⚠ **Caution!**
At this moment any outputs set will remain set!

Forcing variables and inputs/outputs (Forcing)

All variables of a user program can be forced into fixed values. Forced local outputs are only switched to the periphery when the controller is in RUN status.

→ The I/O connected through the CANopen fieldbus cannot be forced.

Status display in the programming software

- The signal states of the physical, Boolean inputs are displayed in both the CPU's RUN state and in STOP.
- The signal states of the physical, Boolean inputs are only displayed in RUN state; in the STOP state they are designated with FALSE.
- All other variables are displayed with the current variable value.

High-speed counters (Counter)

The controller input for pulse processing is shown in section "Connecting a pulse transmitter/incremental encoder" (page 23) for every counter function. After you have selected the counter type, such as 32 BitCounter (,), the other inputs/outputs of a counter are shown in the PLC configuration with symbols, such as Reset. (→ chapter "Configuration of the inputs/outputs (I/O)" page 39). The symbolic inputs/outputs are parameterised in the program (programming with symbols).


```
0001 PROGRAM PLC_PRG
0002 VAR
0003     Zaehlerwert: DWORD;
0004     Freigabe: BOOL;
0005     Richtung: BOOL;
0006     Urzustand: BOOL;
0007     Setzwert: DWORD;
0008 END_VAR

0001 Zaehlerwert:= Counter; (* Abfrage des Zähleristwerts *)
0002
0003 Enable:=Freigabe;(* Freigabe *)
0004
0005 Direction:=Richtung;(* vor-/rückwärts zählen *)
0006
0007 Reset:= Urzustand;(* Rücksetzen *)
0008
0022 Preset:=Setzwert;(* Sollwert setzen *)
```

Figure 41: Programming inputs/outputs of the 32-bit counter

It must be taken into account that the operations, such as Reset are processed via the program image. The output "Reset AT%QX1.2" is not active until the end of the program.

You can also achieve faster access times by bypassing the image register, such as may be required in an Interrupt routine. In this case, use the function blocks – as an alternative to programming with symbols. The library EC_Util2.lib contains a function block for every counter type. The function block for the 32-bit counter has the following parameters:

Figure 42: 32-bit counter function block

The inputs/outputs of the function blocks are essentially the same as the inputs/outputs listed in the PLC configuration.

➜ You must also program the Enable input in the PLC configuration to enable the 16-bit and for the 32-bit counter, in addition to the function block inputs.

Counter functions (inputs/outputs)

The description of the input/output functions in the following sections applies to the inputs/outputs of function blocks and those of the PLC configuration.

32-bit counter
Only one 32-bit counter is available. The pulse transmiter must be connected with the external input I1. It receives the pulses at a maximum frequency of 50 kHz. The CPU counts these pulses and provides them as an actual (= Counter) value. The actual value can then be scanned in the user program. Whether the actual value is incremented or decremented when a count pulse is received depends on the setting of the Direction output in the user program.

➜ An interrupt can be generated if the actual value is the same as the reference value. This causes a program routine to be executed. To do this, you must activate the interrupt in the task configuration and assign the program routine ➜ section "Interrupt processing", page 52.

The following counter features can be defined via the program:

- Enable:
  - TRUE: Pulses are counted.
  - FALSE: Pulses are not counted.
  - A 1 signal at the Enable input activates the counter: The incoming pulses are counted. With the next 0 → 1 edge of the Enable signal, the actual value is set to 0 and the status at the Direction input and at the Preset input are accepted. Any direction change during operation is not detected.
- Direction:
  - Incrementing (Direction = FALSE): the counter counts up to the set reference value (PRESET). Once the reference value has been reached, this activates the configured interrupt which branches to a program routine (➜ page 52). The counter continues counting from zero when the next count pulse is received.
  - Decrementing (Direction = TRUE): with the first count pulse, the actual value is set from 0 to the setpoint. If an interrupt is programmed, the associated program routine is called (➜ page 52). With each further pulse, the actual value is reduced until it reaches 0. On the next count pulse the reference value is accepted again and the program routine is called again.
- Reset:
  - A 0 → 1 edge at the Reset input will cause the actual value to be set to 0 and the direction and reference value to be accepted, irrespective of the status of the Enable signal.
- Preset
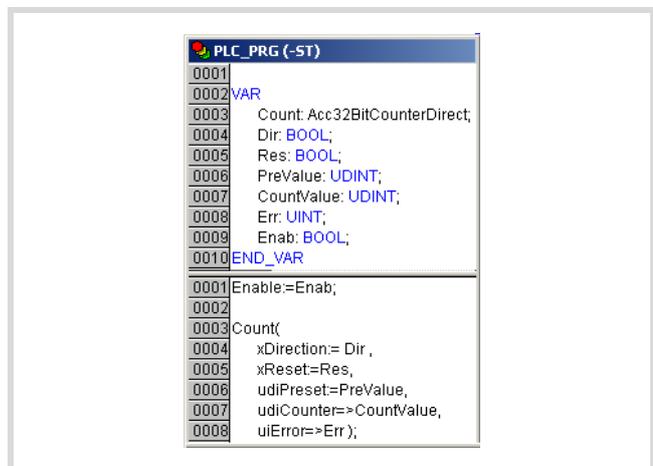
Example: Program with FB for 32-bit counter

Figure 43: Program with FB for 32-bit counter

16-bit counter

The function is available twice. The function of this counter is the same as that of the high-speed counter (32-bit). In order to identify the two 16-bit counters, the symbolic operands have a number: 0 or 1. The operands with 0 control count pulses that are present at input I1. Those with the number 1 are for the count pulses of I2.

External inputs:

Counter number	Pulse input
0	I1
1	I2

The counter number can be seen in the symbolic operands in the PLC configurator in the folder "16Bit Counter".
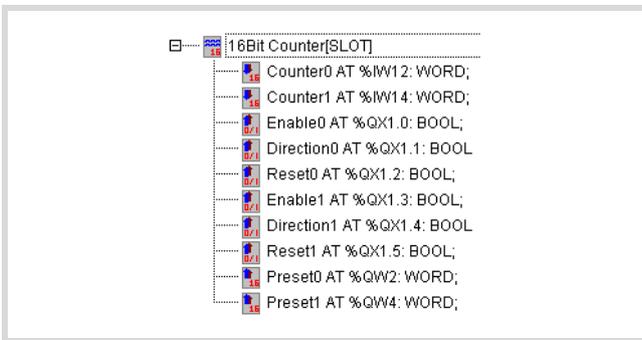

Figure 44: Inputs/outputs of the 16-bit counter 0 and 1

→ When the actual values equals the reference value, an interrupt can be generated in order to activate a program routine. To do this, you must activate the interrupt in the task configuration and assign the program routine → section "Interrupt processing", page 52.

Incremental input

One incremental input is available. The incremental signals A and B of the transmitter are sent to the external inputs I1 and I2, and the reference signal that is generated by the transmitter with every rotation is sent to input I3. The reference switch is connected to input I4. When closed this forms the reference window in which the reference signal is processed.

The incremental signals A and B are phase shifted by 90 degrees in order to indicate the count direction. The rising and falling edges are evaluated (quadrature decoding). The maximum input frequency is 40 kHz. This results in a total frequency of 160 kHz. The counter does not generate an interrupt.

You can control the counter and adapt it to the application with the following signals. The signal inputs can be scanned and the signal outputs set in the program. The signal designations are provided in the PLC configuration.
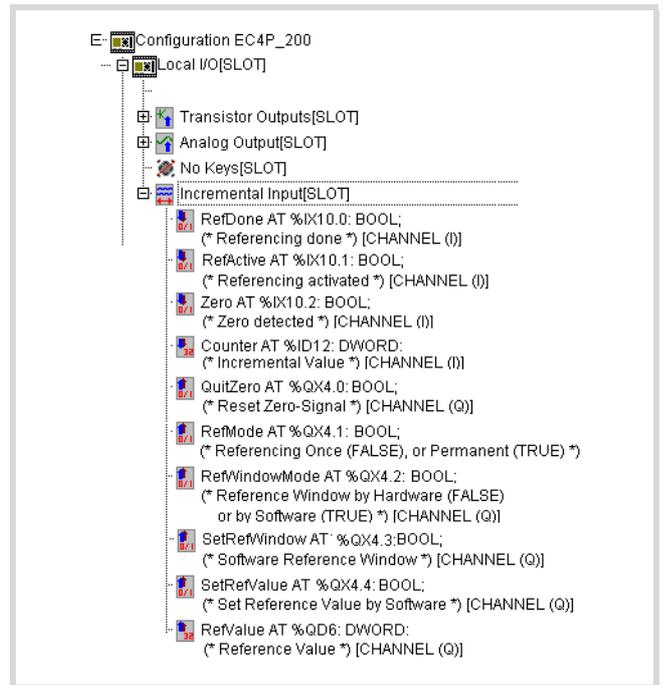
Figure 45: Input/output signals of the incremental value counter

Explanation of the input/output signals (I/Q)

| Signal	I/Q	Explanation
RefDone	I	Referencing completed (feedback signal of SetRefWindow)
RefActive	I	Referencing activated (feedback signal of SetRefWindow or I4)
Zero	I	Actual value through zero
Counter	I	Counter actual value
QuitZero	Q	Acknowledge ZERO signal
RefMode	Q	Number of reference checks 0 = once 1 = permanent
RefWindowMode	Q	Activation of reference window by 0 = external input I4 1 = with "SetRefWindow" in the program
SetRefWindow	Q	Activation of reference window when "RefWindowMode" = 1
SetRefValue	Q	Reference value overwrites actual value (Reset)
RefValue	Q	Reference value

Overview of input/output signals (I/Q)
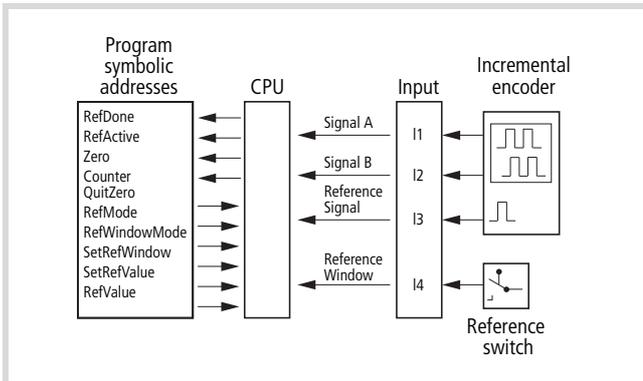
Figure 46: Input/output signals of the incremental value counter

Functions of the input/output signals

Switching the CPU from HALT → RUN enables the counter: The incoming pulses are counted.

SetRefValue (Reset)
A rising edge 0 → 1 at the input overwrites the actual value with the value present at the RefValue input.

Counter (actual value)
The counter actual value is provided at the Counter input.

Zero
If the actual value reaches the value 0, the Zero output is set. It remains set until it is acknowledged by a 0 → 1 edge at the QuitZero input.

RefWindowMode (activate reference window)
You can use this signal to define whether the signal for setting the reference window is sent via input I4 or via the user program with the SetRefWindow signal.

RefMode (type of referencing)
The signal at the input determines whether referencing is carried out once (0 at input) or permanently (1 at input). The actual value is overwritten with the reference value if the reference window is set and a reference pulse is present at input I3. This is carried out once (if the conditions are present after the controller is started) or permanently (with every reference pulse in the reference window depending on the setting for the RefMode signal).

Referencing

In many positioning controllers, a reference point is approached at the start of positioning. For example, a tool slide is moved to its home position. In this position, a reference switch is closed, thus sending a signal to input I4. This can also be done by the SetRefWindow signal which can be activated in the user program. The RefActive signal is set as a feedback signal. An incremental encoder connected to the slide generates a reference pulse to specify the tool position exactly. This is detected at input I3 if the reference switch is closed and the reference window is opened. The reference pulse causes the counter to be overwritten with the reference value that you have defined in the PLC configuration. RefActive is reset and RefDone is set until the reference window is opened again.

→ Set the reference window large enough for the reference signal to be present only once and still be evaluated reliably.
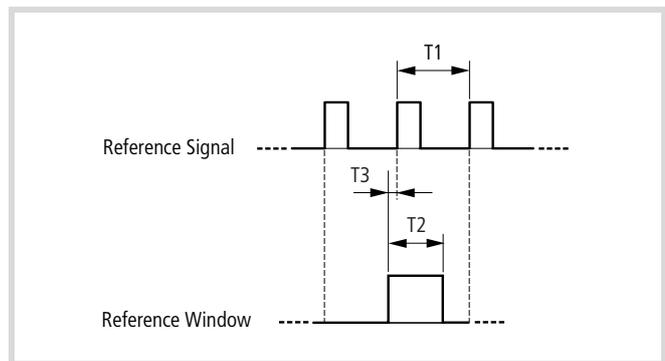
Figure 47: Relationship between reference signal and reference window

T1 Pulse repetition time of two successive reference pulses with one rotation of the incremental encoder

T2 Maximum permissible duration of the reference window. Must be sufficiently less than T1 to ensure that a second marker pulse is not detected.

T3 Must be long enough to ensure that the L/H edge of the marker pulse is safely detected.

T2 and T3 depend on the frequency of the reference pulse and must be determined for each application by trial and error.

System events

System events are:

START	START: User program start (cold and warm start)
COLDSTART	Cold start of the user program
WARMSTART	Warm start of the user program
STOP	User program stop (does not apply to cycle time timeout or hardware watchdogs)
I/O Interrupt 1, 2, 3, 4	Voltage change at inputs I1, I2, I3, I4
Counter-Interrupt1	Act = Preset on 16-bit counter 0
Counter-Interrupt2	Act = Preset on 16-bit counter 1 or 32-bit counter
TIMER INTERRUPT	A timer set by the user triggers an interrupt.

You can react to system events of the controller by creating program routines (POUs) that are run once if an event occurs. Its execution is time-monitored. The time base is the configured longest permissible cycle time.

START, COLD START, WARM START, STOP

If an event occurs, such as. such as a warm start of the controller, an interrupt is generated (→ page 52) that calls up the program routine assigned to it. This assignment is carried out in the task configuration.

Figure 48:   System events

Interrupt inputs I1 … I4

Inputs I1 to I4 can be configured as interrupt inputs. An edge at the input generates an interrupt signal (⟶ page 52) that calls the program routine assigned to it.

▶ First define the edge of the input signal in the PLC configurator.
▶ Assign the program routine to the input in the task configuration.
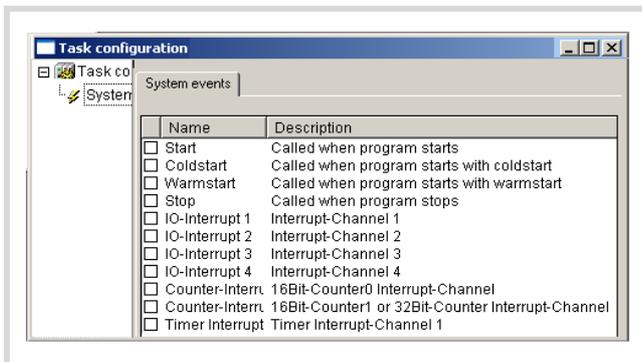
The inputs are prioritised in groups. I1 and I2 have a high priority, I3 and I4 have a low priority.

Lower priority interrupts can be interrupted by those with higher priority.

Counter interrupt

When using the High-speed counter function, the controller continuously compares the actual value with the reference preset value of the counter. If both are the same, an interrupt is generated (⟶ page 52) which calls the program routine (POU) you have created.

To do this you first have to define the counter type in the PLC configurator. You then have to assign the input receiving the count pulses to the POU in the task configuration.

Timer interrupt

You can create a program routine that is called at a fixed time interval. The TIMERINTERRUPTENABLE function is started by a Boolean variable or an external input. The program routine is assigned to the timer interrupt in the task configuration. The interval can be set from 500 – 2 500 000 microseconds. This period duration is programmed by adding the TIMERINTERRUPTENABLE function from the EC_Util.lib library to your user program.

Figure 49: TimerInterruptEnable function

Enter the interval time at the dwTimerTickUS input.

The value is accepted with the start of the timer and can not be modified for the run time. If the value assigned is outside the 500 – 2 500 000 range, the function outputs FALSE as a return value and the timer is not run.
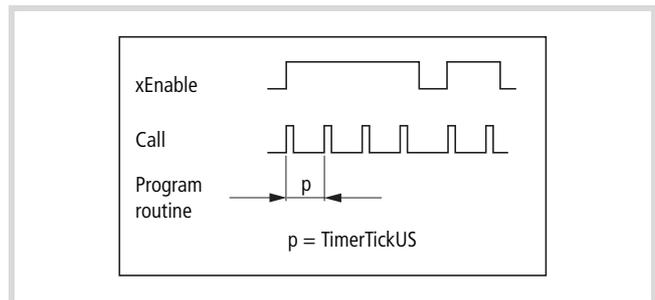
Figure 50: Periodic calling of the program routine

For example, to set an interval time of 2 seconds to be started by the external input I0.0, you must enter the following line in the user program:

```
TimerInterruptEnable(%IX0.0,2000000)
```
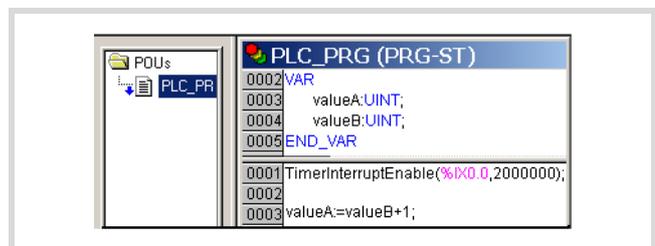

Figure 51: Including the function in the program

Example
- Create a program with a function call

Create a program with the function TIMERINTERRUPTENABLE as per figure 51.

- Creating the program routine
▷ Open the Task Configuration sub-directory with a double click in the Resources directory.
▷ Click here the System Events folder. The System events tab is active.
▷ Click the Timer Interrupt check box to activate the timer interrupt.
▷ In the Called POU column enter the name of the program routine, e. g "Time_Int".
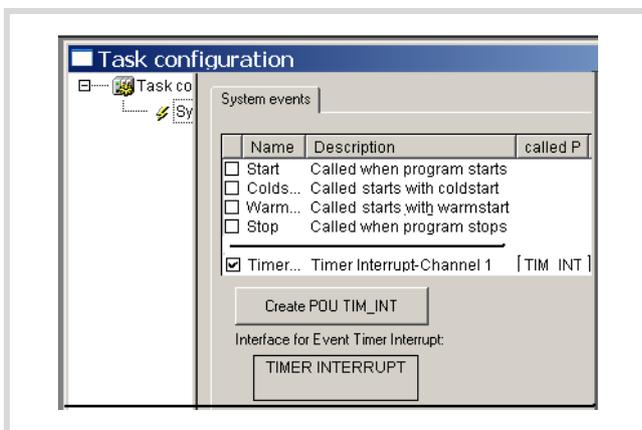
Figure 52: Creating the program routine

▷ Click again on the name "Timer Interrupt". Now the "Create POU" button becomes active and indicates the name of the POU.
▷ Click on this button. A POU with the name "Time_Int" will be added under "PLC_PRG" in the POU window.
▷ Open the POU and write your program routine:
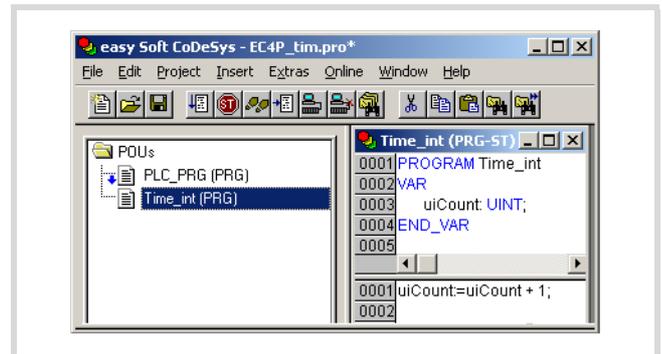
Figure 53: Writing a program routine

If input IX0.0 is activated, the "Time_Int" POU is called periodically and the variable "uiCount" is incremented.

→ The interrupt can be interrupted by higher-priority system interrupts. Cycle time monitoring is active during execution of the timer interrupt.

   If timer interrupts occur too frequently, this may cause the selected program cycle time to be exceeded. In this case the controller will switch from RUN to STOP.

   The timer interrupt can be disabled and enabled from the user program. The functions "DisableInterrup and "EnableInterrupt" are provided for this purpose in the library EC_UTIL.lib.

Interrupt processing

If an interrupt occurs, the program is interrupted and the program routine associated with the system event is processed. figure 54 shows a list of interrupt sources.

START/STOP
COLD START
WARM START
(start of controller)

TIMER INTERRUPT
(periodic)

Interrupt inputs
(voltage at I1, I2, I3, I4)

High-speed counter
(Actual value = Preset)

Program for processing the interrupt

Figure 54:   Interrupt sources

→   The execution time of the program routines is monitored.

The program routine called by the interrupt can be interrupted by a new interrupt (different channel).

If the current interrupt is followed by a new interrupt (same channel), the new interrupt is not executed until the processing of the current one has been completed.

The interrupts are enabled in the RUN state of the CPU and disabled in the STOP state. Interrupt sources which are not enabled in the configuration do not initiate an interrupt.

You can disable or enable the interrupt inputs I1....I4 and the timer interrupt from the program. The functions "DisableInterrup" and "EnableInterrupt" are provided for this purpose. A call parameter determines whether a single interrupt or all interrupts are to be disabled/enabled. A disabled interrupt must be enabled with the same parameter that was used to disable it.

The two functions DisableInterrup and EnableInterrupt are provided as part of the library EC_UTIL.lib. This library must be included if necessary in your project by the Library Manager of the programming software.

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

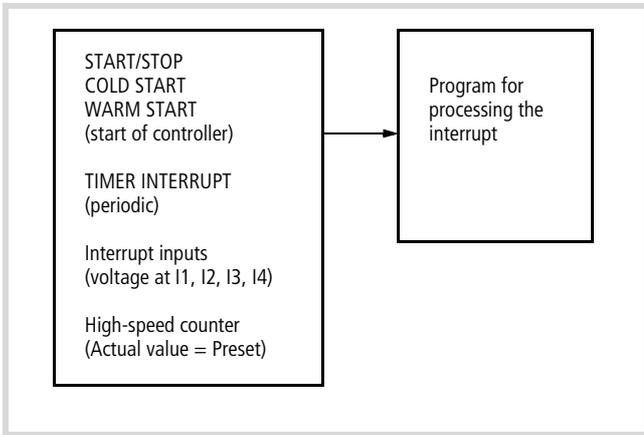EnableInterrupt: With this function, the physical interrupt which was deactivated beforehand can now be re-enabled as an active interrupt.

Steps for interrupt processing

▶ Define the interrupt properties:

Startup behaviour	Select type
TIMER INTERRUPT	Call function TIMERINTERRUPTENABLE
Interrupt inputs	Define edges
High-speed counters	Select type

▶ Create the program routine (POU)
Another program routine (POU) must be added to the existing POU PLC_PRG . This is of type PRG and calls an interrupt.

▶ Assign the program routine to an interrupt source:
  – To do this call the PLC configurator and click Task Configuration l System Events. The interrupt sources (names) are listed in the "System Events" tab with a free entry field for the name of the "Called POU".
  – Enable the interrupt by clicking the box next to the required interrupt and entering the name of the POU in the same line. Further details on this are described in the Example of interrupt processing.

Example of interrupt processing

A "PLC_PRG" POU has to be processed continuously. An additional POU "Fastprog" has to be processed when a rising edge (L → H) at input I3 generates an interrupt.

▶ Create the POUs "PLC_PRG" and "Fastprog" as shown in figure 55.
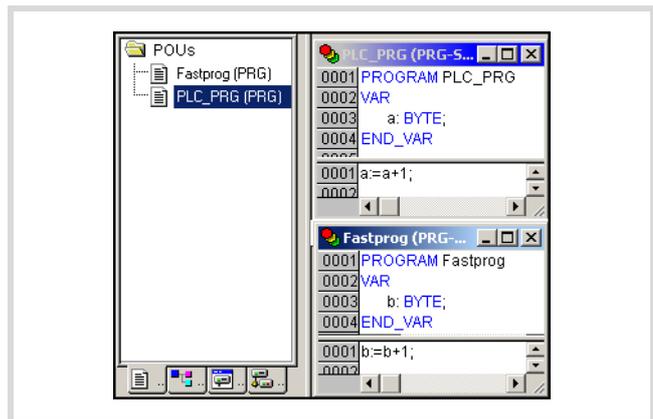
Figure 55:   Writing a program

▶ Move to the PLC configuration, click on the Local I/O[SLOT] folder and open the "Other Parameters" tab
▶ Assign the "Rising edge" type to input I3.
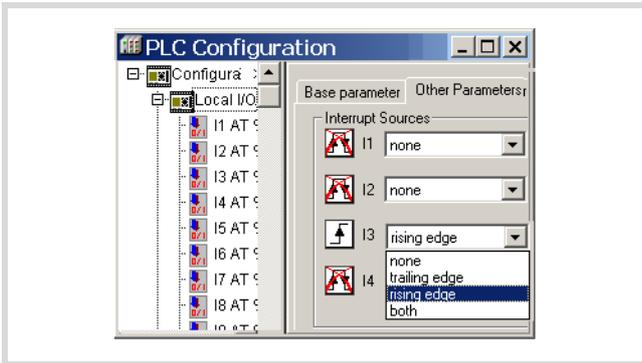
Figure 56:   Interrupt edge selection

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

Figure 57:   Enabling an interrupt

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

The Input Assistant window is opened. This lists all the predefined programs:

Figure 58:   Allocation of Interrupt source → POU

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

The variable "b" is incremented by one with every rising edge on input I3.

Direct I/O access

The functions of the library EC_Util.lib allow you direct access to the local I/O on the PLC. This is executed directly from the user program and not via the I/O image register. Direct access is not supported for the following inputs/outputs:

• Inputs/outputs of the expansion devices
• Local diagnostics bits
• Buttons of the rocker switch
• Inputs/outputs of the devices that are integrated via a bus system.

Access to the high-speed counters can be implemented using the function blocks of the EC_Util2.Lib library.

Description of functions

The function "ReadBitDirect" is described as an example for all other functions:

ReadBitDirect
This function enables you to read the status of an input bit directly. It is stored in the variable to which the parameterised pointer "ptr_xValue" is assigned. The pointer variable will not be changed when a fault occurs during processing.
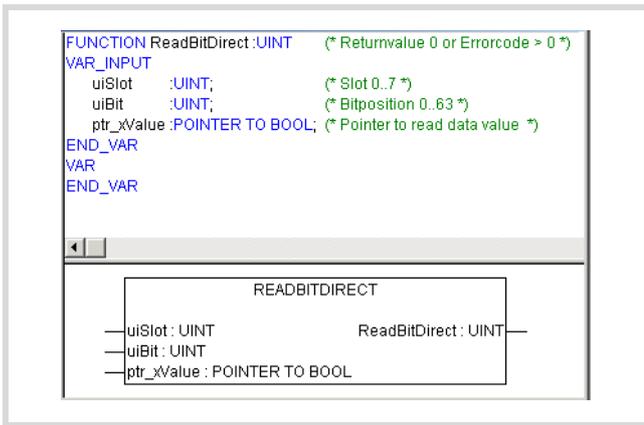
Figure 59: ReadBitDirect function

Function call: ReadBitDirect(uiSlot, uiBit, ptr_xValue)

The following tables show the access options available:

Table 10: Functions for accessing the I/Os

EC_UTIL.Lib	EC4-200-I/O			
	I1-I12	I7,I8,I11,I12	Q1-Q8	QA1
	Digital	Analog	Digital	Analog
ReadBitDirect	Bit 0-11	–	–	–
ReadByteDirect	Byte 0+1	–	–	–
ReadWordDirect	–	Offset 2,4,6,8	–	–
WriteBitDirect	–	–	Bit 0-7	–
WriteByteDirect	–	–	Byte 0	–
WriteWord Direct	–	–	–	Offset 2

Table 11: FBs for the high-speed counters

EC_UTIl2.Lib	High-speed counters		
Function block	16 Bit	32 Bit	Incremental
Acc16BitCounterDirect	Offset 0+1	–	–
Acc32BitCounterDirect	–	OK	–
AccIncremental InputDirect	–	–	OK

Error code for "direct I/O access"

Verify all functions as far as possible for the validity of the call parameters. If a fault is determined, access is not undertaken and an error code is output.

The following return values are possible:
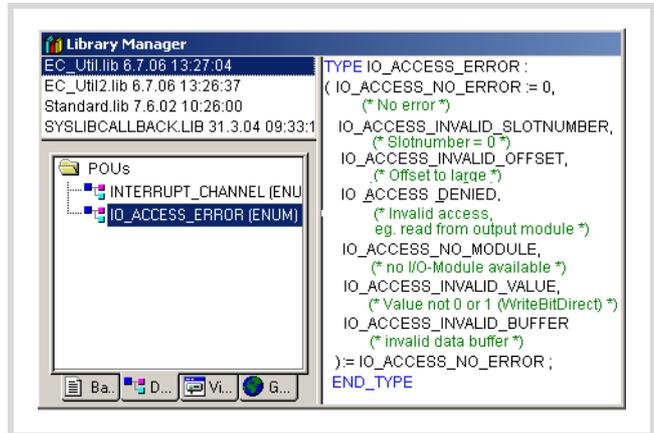
Figure 60: Return values of the EC-UTIL.Lib functions

Table 12: Overview of return values

	IO_ACCESS_INVALID_SLOTNUMBER	IO_ACCESS_INVALID_OFFSET	IO_ACCESS_DENIED	IO_ACCESS_NO_MODUL	IO_ACCESS_INVALID_BUFFER
READBITDIRECT	x	x	x		x
READBYTEDIRECT	x	x	x		x
READWORDDIRECT	x	x	x		x
WRITEBITDIRECT	x	x	x		
WRITEBYTEDIRECT	x	x	x		
WRITEWORDDIRECT	x	x	x		
ACCESS16BITCOUNTERDIRECT		x	x		
ACCESS32BITCOUNTERDIRECT			x		
ACCESSINCREMENTALINPUTDIRECT			x		

Generating and transferring a boot project

The CPU processes the user program stored in the main memory. As the working memory is not backed up, the program will be lost in the event of a power failure. You can therefore create a boot project to store the program retentively.
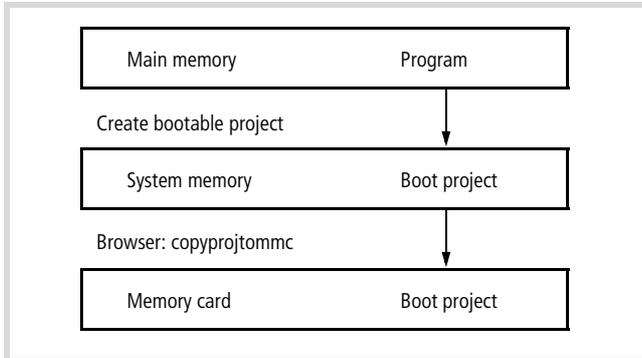
Figure 61: Saving the boot project

You can generate the boot project in online mode or via the menu of the controller. The boot project is generated with the current operating system of the controller!

In online mode the following steps are required:

▶ From the "Online" menu, select "Login".
▶ If the controller is in the RUN state, you will be requested to stop it.
▶ Select the "Create boot project" command.

The following prompt appears:
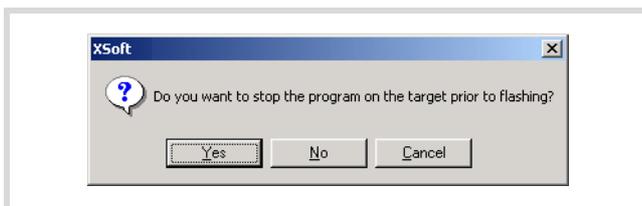
Figure 62: Create bootable project

▶ Click Yes.

The following dialog appears briefly:

Figure 63: Creating a boot project

The boot project has been created when this dialog disappears again. You can now restart the PLC.

Storing a boot project on a memory card

The boot project stored in the system memory (Flash) can also be stored on the memory card. This is done by calling the browser command "copyprojtommc" in online mode or by choosing PROGRAM → BOOT PROJECT → FLASH → CARD from the main menu of the controller using the operating buttons.

Boot project and operating system (OS) on memory card

The boot project will only run with the actual operating system (OS) on which it was created!

If you fit the memory card with an OS into the controller, the OS of the controller will be updated after startup and a boot project will be loaded into the controller. If the boot project was not created with the OS, it will not be detected by the controller. In this case, load the program and create a new boot project.

Erase boot project

The browser command "Remove" deletes both the boot project in the system memory (Flash) and also on the memory card. The browser command "removeprojfrommmc" deletes the boot project and the Startup.INI file on the memory card. The boot project on the memory card can also be deleted via the menu of the controller: PROGRAM → DELETE → DELETE CARD.

Download/update operating system

The EC4-200 enables you to replace the currently stored operating system (OS) with a more recent version. The latest OS version can be downloaded from the Eaton website:
http://www.eaton.com/moeller → **Support**
It is also included on the latest easySoft-CoDeSys CD.

▽ **Caution!**
The download is only possible in offline mode via the RS232 interface! Downloading the OS will delete all the files on the controller/memory card. The controller will then carry out a hard reset, → page 44

The OS can then be transferred in two ways:

• Directly from the PC to the PLC
• From the PC to the memory card. When the controller is started, the OS is copied from the MMC into the controller.

Transferring the operating system from the PC to the PLC

▷ Open a project, choose <Resources → PLC Configurator> and activate the Common Parameters tab.
▷ Click the Start button in the Update operating system field.

The Download operating system dialog opens.

Figure 64: Download operating system

The system reports that the COM port is not initialised.

▷ Click the OS-File button and select the required operating system file (*.hex).

→ The files last opened can be selected from the list field (dropdown menu).
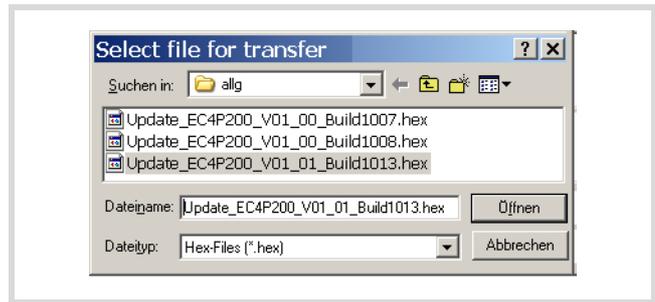
Figure 65: Operating system file selection

The target type and file version are displayed.

▷ Click on the "Transfer OS to PLC" button.
▷ Select the RS232 interface.

The transfer will start. The Flash Eprom is programmed in around 20 to 30 seconds.

→ The power supply must not be switched off if a warning symbol appears in the Programming Flash field!
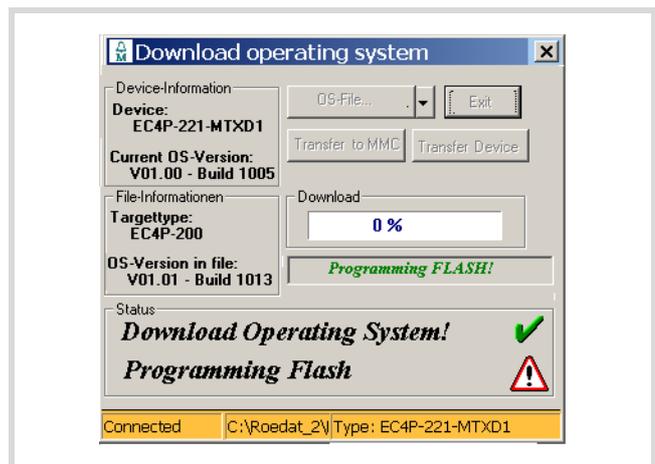
Figure 66: Warning during download

Wait for the following display.
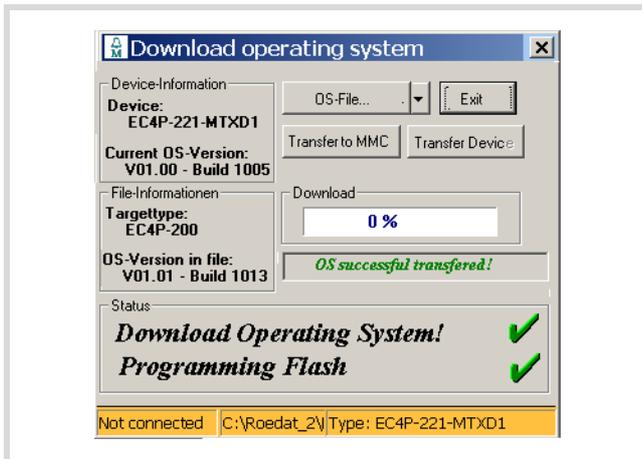
Figure 67: OS successfully transferred to the PLC

▶ In this window click the Exit button.

Transferring the OS from PC to the memory card

Loading an OS onto the memory card will delete the existing OS and the boot project on the memory card as well as the user program in the controller. This is carried out in the same way as described in section "Transferring the operating system from the PC to the PLC". However, in this case you click the Transfer to MMC button, → figure 64 on page 56.

Transferring the OS from the memory card to the controller

▶ Fit the memory card into the controller when it is switched off.
▶ Switch on the PLC.

The OS of the PLC is updated during the switch on process and a boot project is loaded into the PLC. The transfer can take more than 30 seconds as the CPU must be booted several times.

→ Do not interrupt the process, e.g. by switching off the supply voltage!

10 Browser commands

The PLC browser is a text based control monitor. This is where you enter commands in the entry line for scanning PLC status from the PLC. They are sent as strings to the PLC. A results window of the browser displays the response. This allows you to carry out diagnostics or debugging tasks.

→ The browser commands can only be used online.

To run these commands:

▶ Double-click Resources and then PLC Browser in the programming software.

A new window called PLC Browser will appear in the field on the right.

▶ Click the button ⌷ .

The selection field lists the available browser commands.

▶ Double-click the required command to select it.

The selected command now appears in the "PLC Browser" window.

▶ Press the enter button in order to view the response of the PLC to the browser command in the event window.

→ Further information on the selected Browser command can be obtained by entering a "?" followed by a space in front of the selected browser command and then pressing Enter.

Further information on the selected browser command are also described in chapter "Resources → PLC Browser" in the manual on the programming software (MN05010003Z-EN).

The controller supports the browser commands from table 13.

Setting Ethernet parameters

If you use a browser command to set the IP address/subnet mask or gateway address, this is not active until the following steps have been completed:

• Switch the device off and on
• Enter the "reboot" command
• Unplug and replug the Ethernet cable on the EC4-200.

Table 13: Browser commands

?	Get a list of implemented commends.
pinf	Output project information
cycle	Output cycle time
canload*	Display load of CAN bus
copyprojtommc	Copy the current boot project to the memory card
createstartupini	Generate the Startup.INI file on the memory card
factoryset	Activate factory settings
format	Format memory card
GetNodeId	Display the CANopen Node ID of the CAN interface
GetRoutingId	Display of the routing Node ID and the routing interface
getipgateway	Display of the set gateway address (Default address: 0.0.0.0)
getipconfig	Display of set IP address and subnet mask Display of default values: IP address: 192.168.119.60 Subnet mask: 255.255.255.0
getmacaddress	Display of the MAC address of the Ethernetcontroller. Display e. g. 00-80-99-05-11-22
getrtc	Read real-time clock
metrics	Output PLC information
reboot	Reboot of EC4-200 ▽ Execution is carried out immediately, i.e. without a prompt, after the command is confirmed.
reload	Load boot project from FLASH to PLC
remove	Delete boot project in the FLASH
removeprojfrommmc	Delete boot project and Startup.INI file on the memory card
removestartupini	Delete the Startup.INI file on the memory card
setipconfig	Setting the IP address and subnet mask Syntax (example): setipconfig 192.168.119.60 255.255.255.0 -> ok or error message
setipgateway	Setting the Gateway address Syntax (example): setipconfig 192.168.119.10 -> ok or error message
setrtc*	Set real-time clock

Further information concerning the commands marked with * can be found in the following pages.

Description of important Browser commands

canload

Indicates the capacity utilisation of the CANopen fieldbus.

Example:
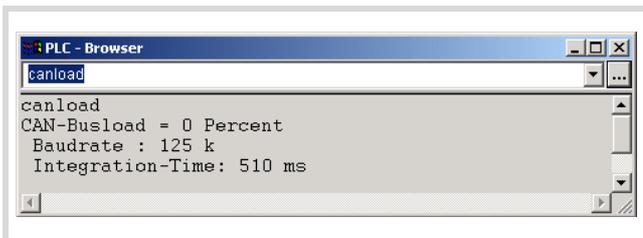
Figure 68: Browser command canload

This browser command returns, for example, the following information:

- CAN busload = 0 Percent
- Baud rate 125 Kbit/s
- Integration Time: 510 ms

▽ **Attention!**
With a bus utilization of 75 percent or higher, the warning ATTENTION: HIGH BUSLOAD also appears. Overload of the local CAN bus in conjunction with further short term load peaks can lead to CAN data loss.

→ As well as the browser command, the CAN_BUSLOAD function can also be used to determine the CAN bus load from the user program → section "CAN_BUSLOAD function" on page 62.

setrtc

Sets or changes the date and/or the time in the controller.

Syntax:

```
<setrtc_YY:MM:DD:DW_HH:MM:SS>
```

Legend:

_	Space
YY	The last two digits of the year (00 F YY F 99)
MM	Month (01 F YY F 12)
DD	Day (01 F DD F 31)
DW	Weekday (01 F DW F 07; 01 = Monday, 07 = Sunday)
HH	Hour (00 F HH F 23)
MM	Minute (00 F MM F 59)
SS	Second (00 F SS F 59)

11 Libraries, function blocks and functions

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

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

The libraries are located in the folders:

- Lib_Common for all PLCs
- Lib_EC4P_200 for the EC4-200 controller

Using libraries

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

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

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

Document	Library
MN05010003Z-EN (previously AWB2700-1437)	Standard.lib Util.lib XX_Util. Lib
Online help or PDF files (in the Windows start menu via Programs → Moeller Software → easySoft-CoDeSys → Documentation → Automation Manuals)	SysLib…pdf
MN05010002Z-EN (previously AWB2786-1456GB)	XS40_MoellerFB. Lib/Visu. Lib/…
AN2700K20	3S_CANopenDevice. Lib 3S_CANopenManager. Lib
AN2700K19	3S_CANopenNetVar. Lib
AN2700K27	SysLibCan. Lib
MN05010001Z-EN (previously AWB2786-1554GB)	CANUserLib. lib CANUser_Master. lib

Installing additional system libraries

You can install libraries manually as follows:
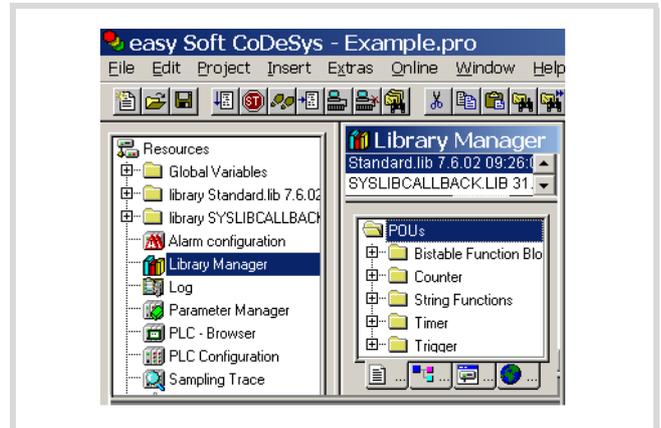
Figure 69: Libraries, installing manually

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

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

▶ Select the library to install and click Open.

The library now appears in the Library Manager.

EC4-200 specific functions

EC_Util.lib library

This library contains the functions shown in the illustration below:
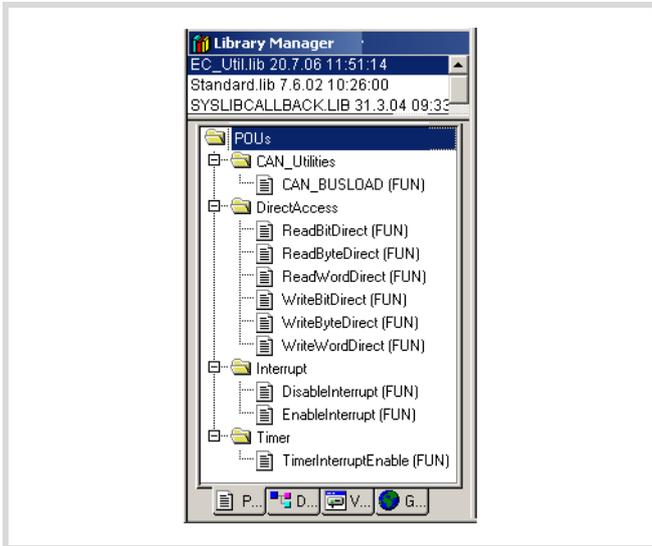
Figure 70: Functions of the EC_Util.lib library.

CAN_BUSLOAD function
This function can be called cyclically in a user program. When a read cycle was completed successfully, the function returns the TRUE value and writes the values calculated for integration time and bus load to the transferred addresses.

If the bus load calculation is not yet completed or the CAN controller has not yet been initialised, the function returns FALSE.

Information on evaluating the return value is provided in the browser command „canload" on page 60.
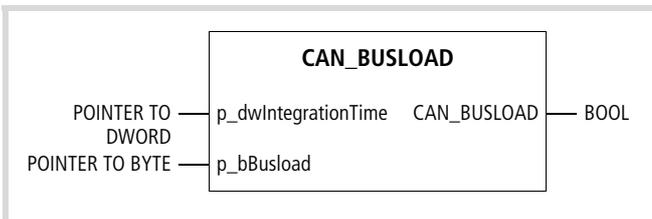
Figure 71: Function CAN_BUSLOAD

The other functions are shown on the following pages:

- Direct I/O access (DirectAccess) → page 53
- TimerInterruptEnable → page 50
- DisableInterrupt/EnableInterrupt → page 52

EC_Visu.lib/EC_Visu2.lib library

The EC_Visu2.lib library contains function blocks for controlling the LCD display.
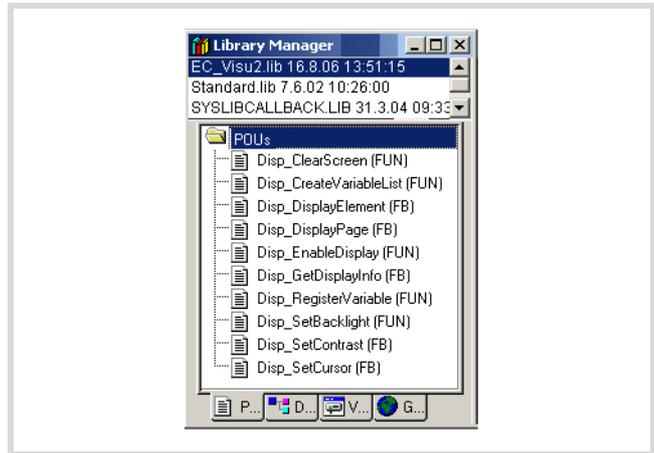
Figure 72: EC_Visu2.lib library

→ The already available functions/function blocks SetBacklight, SetContrast and GetDisplayInfo from library EC_Visu.lib can still be used. However, they have been replaced by the functions/function blocks contained in the library EC_Visu2.lib (→ table)

EC_Visu.lib	EC_Visu2.lib
SetBacklight	Disp_SetBacklight
SetContrast	Disp_SetContrast
GetDisplayInfo	Disp_GetDisplayInfo

This function is described on page 76.

12 Connection setup PC – EC4-200

The communication parameters of both the PC and the PLC must match in order to establish a connection between them. The default parameters are set as shown in figure 73 on devices that are used for the first time. Just select the COM... interface on the PC. No other settings are required.

➜   An error message means that the default CPU settings have been changed beforehand. In this case, try all other baud rates or set the factory settings.

The CPU parameters can then be defined again (➜ figure 74). always making sure that you have the same settings on the PC.

The connection between the PC and the programming interface of the PLC can be established via:

- the RS232 interface
- the Ethernet interface (with type EC4P-222-... also in addition to RS232)

Connection setup via RS232

First adjust the PC's communication parameters to the standard parameters of the PLC ➜ section "Defining/changing the PC's communication settings".

The RS232 interface of the PLC (COM1) has the following standard parameters:

Baud rate	38400 Bit/s
Parity	No
Stop bits	1
Motorola Byte	No

Defining/changing the PC's communication settings

You can use the COM1 to COMx interfaces of the PC.
In the programming software define the communication parameters of the interface.

▶ In the Online menu, select Communication → Parameters.
▶ Specify the port (COM1 or COM2, ➜ section "Changing settings"
▶ Use the remaining settings as shown in figure 73.
▶ Confirm the settings with OK.
▶ Log on to the PLC.
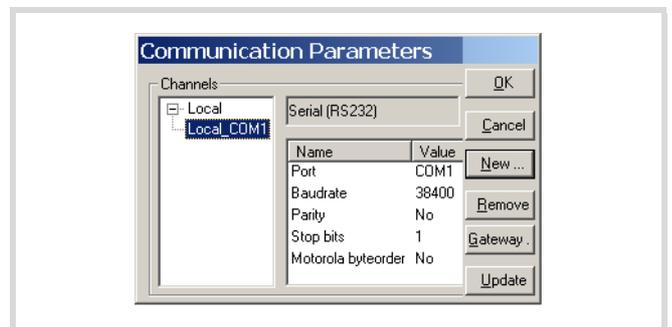
Figure 73: Defining the PC's communication settings

Changing settings
Proceed as follows in order to change parameters such as baud rate or port:

▶ Double-click the value, such as 38400.
  The field is highlighted in grey.
▶ Enter the desired value.

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

Changing the communication parameters (baud rate) of the CPU

▶ Open the PLC configuration.
▶ Click the Communication tab.
▶ In the Baud rate list box select the baud rate (e.g. 57600 Bit/s as shown in figure 74).
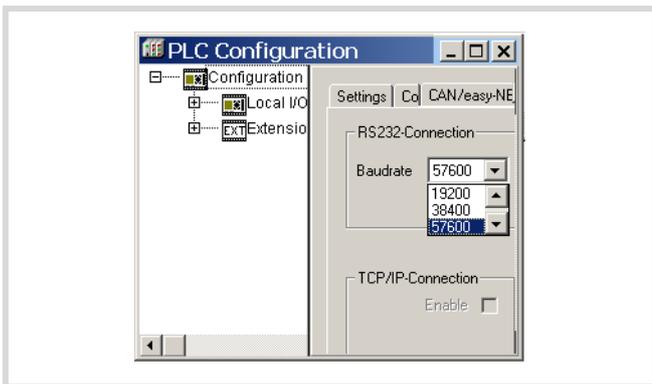
Figure 74:   Specifying the CPU's communication settings

▶ Log on to the PLC.

The following prompt appears:
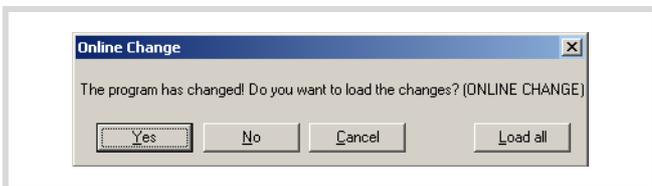
Figure 75:   Confirmation request after program change

▶ Click Yes.

The program is loaded. After a delay of approx. 2 minutes a communication fault message will be output since the baud rate of the CPU and the PC no longer match:

Figure 76:   Communication fault

▶ Acknowledge the message with OK.

In order to reconnect to the PC you must adjust the baud rate of the PC again to that of the project.

Connection setup via Ethernet

After you have connected the PC to the PLC with an Ethernet cable, select the TCP/IP communication channel in the programming software and enter the IP address of the PLC. The PLC has the default address 192.168.119.60.

Selecting communication channel and address
▶ Access the menu with ‹Online → Communication parameters›.
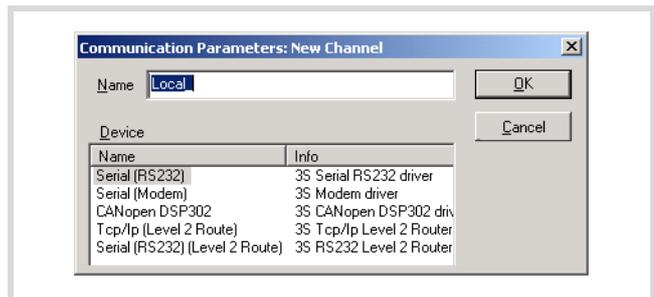
Figure 77:   Channel selection

▶ Push the "New…" button.
▶ Select the overview of the communication channel TCP/IP (Level2Route) and change the name "local"e.g. to "Ethernet-Test"
▶ Confirm with OK.
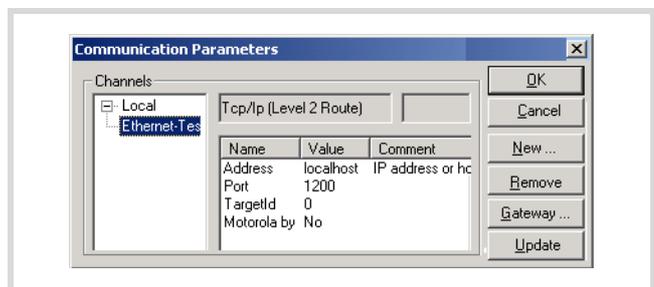
Figure 78:   Enter the IP address

▶ Perform a double click on the "localhost" field and enter the default address 192.168.119.60
▶ Confirm your details, by first pressing on another field and then on OK.
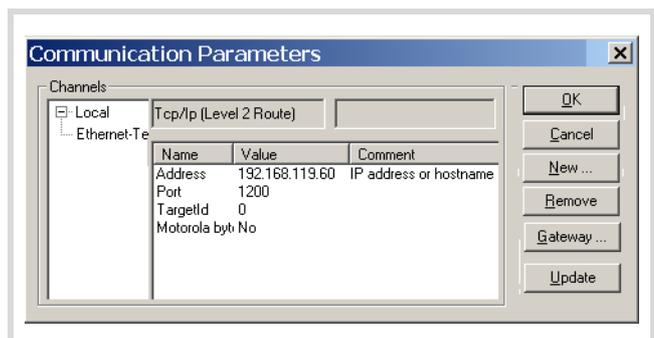
Figure 79:   Communication parameters with IP address

▶ Compile the program and log out.

Data transfer with TCP/IP (Level2Route) protocol
This data transfer is supported by PLC types XC200, MFD4 and
EC4-200 (only EC4P-222…).

The data between the PC and PLC are transferred in data blocks of
128 Kbytes (default setting of the device). The EC4P-222… has
the following restriction: if large volumes of data is transferred,
this can cause an error message due to insufficient memory. In this
case set the block size to 4 Kbytes. The setting is made in the
Communication tab in the Configuration folder:

▶ Simply click on the Adjust settings button. The 4 Kbyte block
   size is then set automatically.

If the button is dimmed such as in figure 80, the block size is
already set to 4 Kbyte.
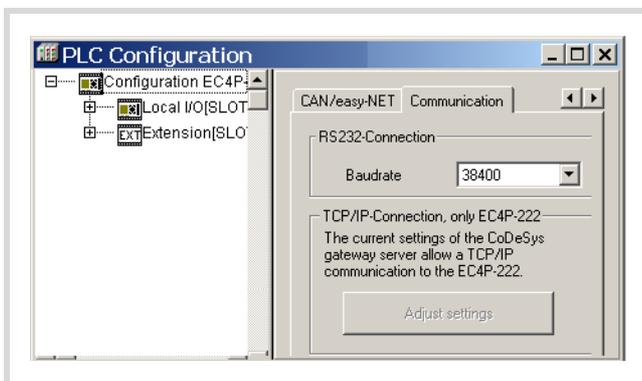
Figure 80:   Setting the block size of the TCP/IP connection

Reset the block size to 128 KByte if you are setting parameters
for a XC200 or MFD4. These PLCs have more memory available.
The setting can be modified as required using the
BlockSizeEditor.exe application (in the Windows Start menu under
Program → Moeller Software → easySoft-CoDeSys →
Communication…).

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 59.

Restart the PLC after you have changed the IP address. Ensure that the IP address of the programming device belongs to the same address family. In other words the IP address of the programming device and the PLC must correspond in the following figure groups with a subnet mask of 255.255.255.0:

Example 1
IP address PLC: 192,168,119xxx
IP address PC: 192,168,119yyy

Example 2
IP address PLC: 192,168,100xxx
IP address PC: 192,168,100yyy

The following conditions apply in example 1 and 2:

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

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

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

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

▶ For the example mentioned you would enter the following text: "ping 192.168.119.60" and confirm this with "OK".

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

The following figure indicates the result of a correct connection set-up.
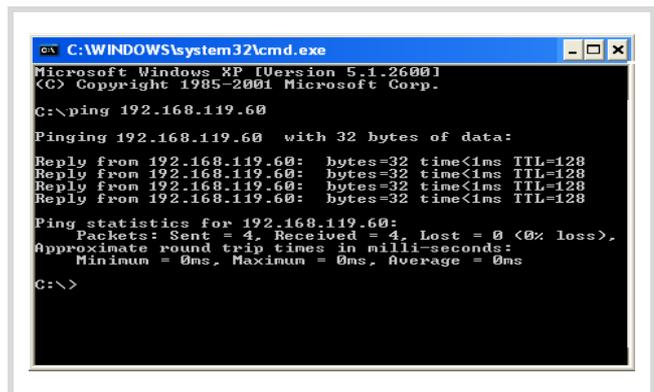
Figure 81: PING response with a correctly established Ethernet connection

13 Defining system parameters via the STARTUP.INI file

Overview

You can create project-dependent system parameters and store them on the memory card. Here they are contained in the Startup.INI file. The memory card can also be fitted in other controllers. The PLC accepts the parameters during start up. The Startup.INI file is always created with all controller parameters (→ table 14).

Table 14: Parameters in the Startup.INI file

```
Entries
COM_BAUDRATE: 4800,9600,19200,38400,57600
CAN1_BAUDRATE: 10,20,50,100,125,250,500
CAN1_NODEID: 1-127
CAN_ROUTINGID: 1-127
IP_ADDRESS=xxx.xxx.xxx.xxx
IP_SUBNETMASK=xxx.xxx.xxx.xxx
IP_GATEWAY=xxx.xxx.xxx.xxx
```

The parameters from the INI file have priority before the parameters of the PLC configuration. The parameters of the PLC configuration are not accepted after a program download or after loading the boot project.

Structure of the INI file

An INI file is a text file with a fixed data format. From a section defined with a name (in square brackets) such as [STARTUP] the system parameters are listed followed by an equals sign and their value. The line is terminated with CR/LF (Carriage/Return).

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

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

```
; CAN_NODEID = 2
```

You can change or create the parameters with a text editor if you fit the memory card in the memory card slot of the PC. First fit the memory card in the supplied adapter, and then fit this into the PC slot. The STARTUP.INI file is stored on the memory card in the folder "MOELLER/EC4P_200/PROJECT/".

Creating the Startup.INI file

When it is switched on for the first time (basic status), the controller always works with the default system parameters, i.e. the STARTUP data. When you load a project into the controller that is in the basic status, the controller starts immediately with the system parameters of the project.

With the browser command "createstartupini" you transfer the current system parameters from the PLC to the memory card. This creates the Startup.INI file that contains this data. Requirement: The memory card must be fitted and formatted, i.e. without a Startup.ini file already on it.

Table 15: Example: STARTUP.INI file for EC4-200

```
[STARTUP]
TARGET = EC4P-200
IP_ADDRESS=192.168.119.60
IP_SUBNETMASK=255.255.255.0
IP_GATEWAY=0.0.0.0
COM_Baudrate=38400
CAN1_Baudrate=125
CAN1_NODEID=2
CAN_ROUTINGID=127
```

It is not possible to overwrite or change an already existing file with the "createstartupini" browser command. If you still enter the command, a warning appears. In order to create a new file the existing file must be deleted first,→ section "Deleting the Startup.INI file" onpage 68.

Switching on the PLC with the fitted memory card containing the Startup.INI file

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

Changing settings

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

Deleting the Startup.INI file

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

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

The behaviour of the Startup.ini file with the Hard Reset and Factory Set menu commands on the controller and with the "factoryset" browser command is described in section "Reset" on page 44.

14 Programming:via a CANopen network (Routing)

Routing means to establish an online connection from a programming device (PC) to any (routing-capable) PLC in a CAN network without having to directly connect the programming device to the target PLC. The target can instead be connected to any other PLC in the network. The routing connection enables you to carry out all the operations that are possible with a direct online connection between the programming device and the controller:

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

Routing offers an advantage which makes it is possible to access all routing capable PLCs on the CAN bus from any PLC which is connected with the programming device. You select the control with which you want to communicate by the project selection. This provides an easy way of controlling remote PLCs.

However, the data transfer rate with routing connections is considerably slower than with direct connections (serial or TCP/IP). This will result in slower refresh times for visualisation elements (variables) or slower download speeds.

Prerequisites

The following requirements must be fulfilled to use routing:

• The routing PLC and the target PLC must both support routing.
• Both PLCs must be connected via the CAN bus.
• The PLCs must both have the same active CAN baud rate.
• The valid routing Node ID must be set on both PLCs.

Routing features of the controller

The controller supports routing via the CAN bus.

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

It is possible, for example, to load a program from the PC into the EC4-200 via a controller of the XC series. In this case, you assign the EC4-200 (target controller) with a routing Node ID.
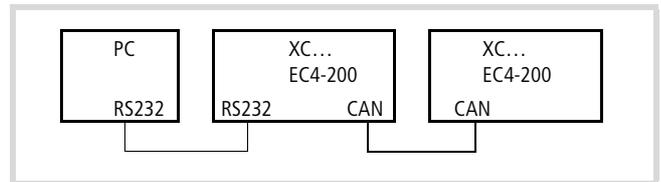
Figure 82:   Program download per Routing

Routing through XC200

To perform a program transfer or routing using TCP/IP through a connection between XC200 and PC, you must first set the block size for the transferred data. The packet size (4 KByte or 128 KByte) depends on the transfer type (program transfer or routing) and the operating system, → table 16.

Table 16:   Block size for data transfer

	Program/file transfer		Routing	
	OS < V1.03.03	OS ≧ V1.03.03	OS < V1.03.03	OS ≧ V1.03.03
Block size Default: 128 KByte	128 Kbyte	4/128 KByte	Routing not possible	4 kByte

▽ **Caution!**
The program download with a block size of 4 KByte to a PLC with an operating system version earlier than V1.03.03 will cause faulty behaviour!

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

Routing with the XC200 is possible from BTS version V1.03.03.

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

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

Changing the block size:

▶ Close all applications.
▶ Close the CoDeSys gateway server.
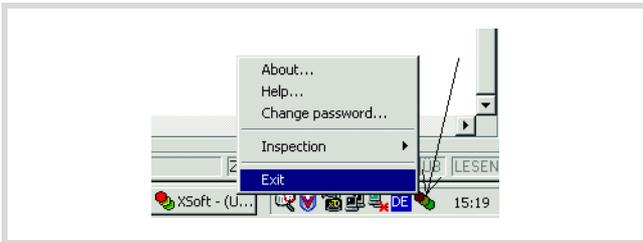
Figure 83: Closing the CoDeSys gateway server

▶ Change the block size to the required value.

Call the BlockSizeEditor.exe application in the easy Soft CoDeSys directory of the programming software and select the block size.

Alternative option:

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

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

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

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

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

Notes on routing

- If large files are written to the target PLC or read from the PLC, it is possible that the online connection will be interrupted after the transfer process has been completed. Renewed connection is possible.
- If a program with a modified routing node ID is loaded into the target PLC, the target PLC accepts the modified routing node ID; however, the communication connection will be interrupted. Reconnection with a corrected routing Node ID is possible.
- A controller cannot be connected via a routing connection if it contains a program without any valid routing parameters (Baud rate/Node ID).

- The routing is independent of the configuration (master/slave): a target PLC that has not been configured as a master or as a slave can be accessed. It must only receive the basic parameters such as Node ID and baud rate, as well as a simple program.

Setting the node ID/routing ID

PLCs on the CAN-Bus can be configured as a master or as a slave. The PLCs are assigned with a Node ID/node number (address) in order to uniquely identify them (with the basis communication). The target controller must also be assigned a (routing) ID if you wish to access it by means of the routing function. The RS232 or the Ethernet interface can be used as connection between the PC and EC4-200.
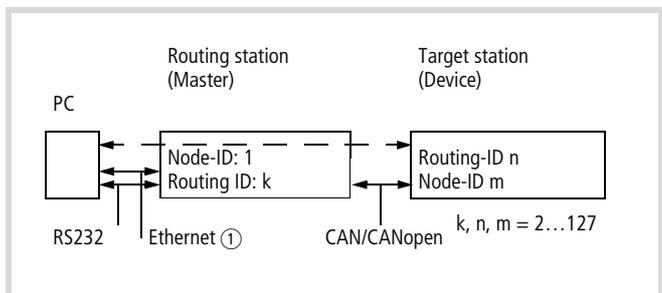
Figure 84: Routing via ID1
(XC…, EC4-200 or MFD4)

① Ethernet connection possible with XC200, MFD4 and EC4-222

Table 17: Example for setting the Node ID, routing ID, Baud rate

PLC	**Function**	**Node-ID Routing-ID**	**Baud rate**	**→ figure**
Routing controller	Master	1 (Basic) 127 (Routing)	125 KB	86 85
Target controller	Device	3 (Basic)	125 KB	87
		54 (Routing)		85

→ The following applies to device stations: The Routing-ID must not be equal to the Node-ID (Basic communication)!

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

Setting the master station

Define two node IDs in the master station:

- One ID for routing
- One ID for basic communication

for the routing function:

Set the routing ID and the CAN baud rate in the CAN/easyNet tab of the Configuration folder as shown in figure 85.
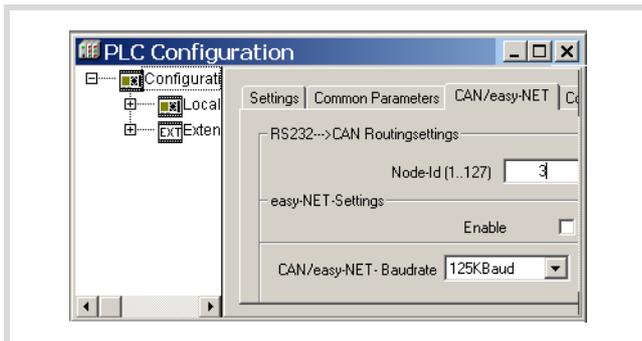
Figure 85:   CAN Master routing settings

For basic communication:

The ID for basic communication and the CAN baud rate are defined in the CanMaster folder in the CAN parameters tab in figure 86.
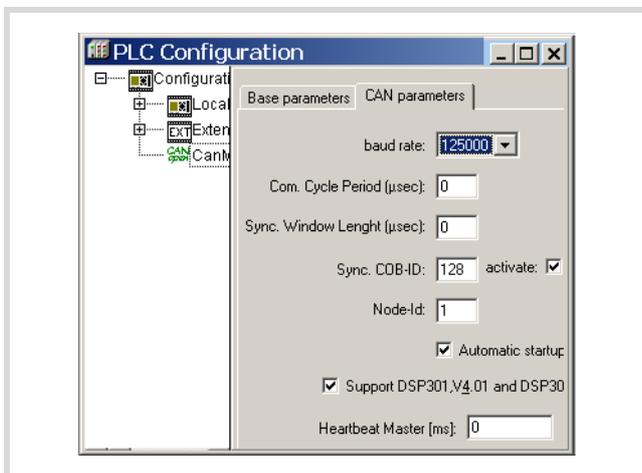
Figure 86:   CAN Master, ID for basis communication

Setting the device station

Define two node IDs in the device (target) station:

- One ID for the routing function: The routing ID and the CAN baud rates are set as shown in figure 85 in the CAN/easyNet tab. As node ID set for example "54".
- One ID for basic communication: The ID for basic communication and the CAN baud rate are defined in the Can settings folder as shown in figure 87.
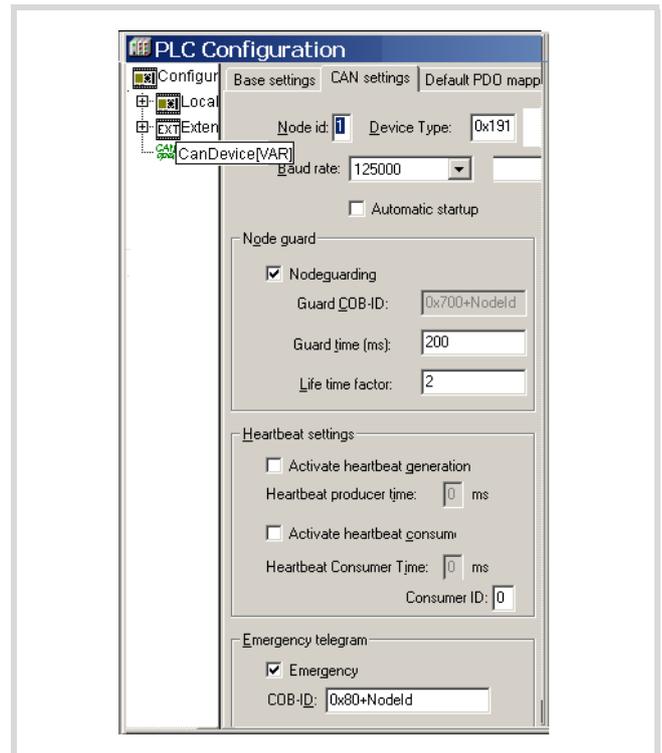
Figure 87:   CAN device parameters

Node ID and baud rate are transferred with the project download.

Example: Accessing a PLC program
The example below illustrates the procedure for accessing a PLC program.
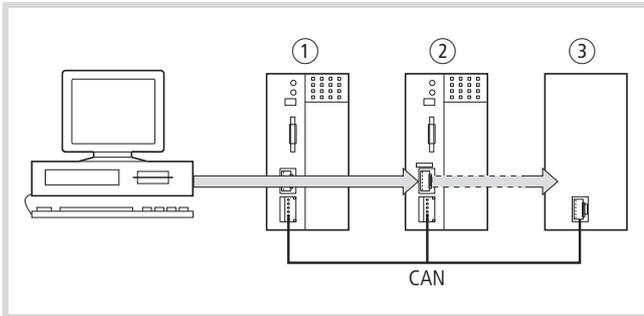
Figure 88:   Diagnostics possibilities

① XC100 with Node ID 1
② XC200 with node ID 2, routing ID 127
③ Controller with the node ID 3 and routing ID 54
   (e.g. XC100, XC200, XC121, EC4-200)

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

▶ Open the project of the target PLC (Node ID 3) whose program you wish to edit or test.
▶ First configure the parameters for the hardware connection PC ↔ PLC (Node ID 2).
▶ From the Online menu select Communication Parameters...
▶ Click the New button under "local" channels.

The "New Channel" window appears.

▶ Select the channel in the "Device" window:
   Serial [RS232] [Level 2 Route] or TCP/IP [Level 2 Route].
▶ You can assign a new name in the Name field, e.g. "Rout_232".
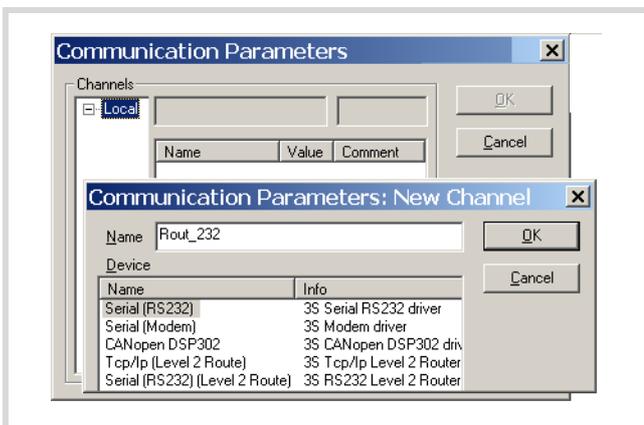
Figure 89:   Channel parameter setting

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

▶ Enter the target ID of the target station, number 54 in the example. The target ID is identical to the routing ID!
   To enter the target ID, click on the field in the Value column, to the right of the term Target ID. Enter the number 54 there and confirm with OK.
▶ Log on and carry out the action.
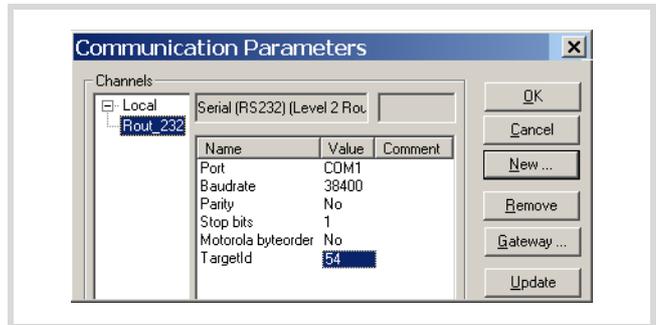
Figure 90:   Setting the target ID of the target PLC

PLC combinations for routing

The following PLC support routing:

From → To ↓	XC100	XC121	XC200	EC4-200	MFD4
XC100	×	×	×	×	×
XC121	×	×	×	×	×
XC200	×	×	×	×	×
MFD4	×	×	×	×	×
EC4-200	×	×	×	×	×

15 RS232 interface in Transparent mode

In Transparent mode, data is exchanged between the EC4-200 and data terminal devices (e.g. terminals, printers, PCs, measuring devices) without any interpretation of the data. For this the serial interface RS232 (COM1/COM2 = Multifunction interface) must be switched to Transparent mode via the user program.

For running the transparent mode there are functions available for opening and closing the interface, for sending and receiving data and for setting the interface parameters. After opening, the interface runs with the current communication parameters that you can adapt by calling the "SysComSetSettings" function.
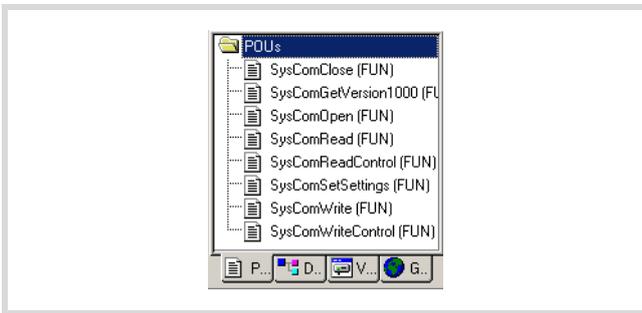
Figure 91: Function summary

The functions of Transparent mode are contained in the library EC_SysLibCom.lib. The library must therefore be included in the Library Manager. A description of the functions is provided in the manual "Function Blocks" (MN05002002Z-EN; previously AWB2786-1452GB).

→ Programming via the RS-232 interface (COM1) is not possible if it is in Transparent mode. Transparent mode must first be disabled. When transparent mode is closed, the original communication parameters are reinitialised.

COM1/COM2:

The transparent mode is forcibly deactivated when the PLC state changes to the STOP mode or when the "SysComClose" function is accessed.

16 Interactive display

The use of functions and function blocks (FBs) enables you to display variables (text/values) on the PLC display and enter values via the buttons/rocker switch. An MFD-CP4 that performs these functions in parallel can be connected to the PLC for external HMI tasks.

Display form

The display of the PLC and the MFD-CP4 has a matrix consisting of 4 lines and 16 columns. Each line can therefore display 16 characters. Three characters sets are available for use.

A maximum of 12 variables can be displayed on one screen of the display.

The display length and number of characters depends on the data type concerned. If the variable consists of a decimal value, an additional place must be allowed for the decimal point. The value of the variable can be continuously updated. A value is entered via the buttons/rocker switch.

Data type	min. / max. value	max. places	max. accuracy	Format
BYTE	255	4	2	1.23
WORD	65 535	6	4	1.2345
DWORD	4 294 967 295	11	9	1.23456789
USINT	255	4	2	1.23
UINT	65 535	6	4	1.2345
UDINT	4 294 967 295	11	9	1.23456789
SINT	−128/127	5	2	−1.23
INT	−32768/32767	7	4	−1.2345
DINT	−2 147 483 648/ +2 147 483 647	12	9	−1.23456789

Switching between Status display and Entry/output mode

In the initial state, the display shows the status of the PLC. The Entry/output mode must be activated in order to output application-specific texts/variables or to enter values/variables. For greater clarity imagine two internal displays in the PLC with displays that are being continuously refreshed. The first is for displaying the status and the PLC menus. The second is for displaying texts and variables in Entry/output mode.
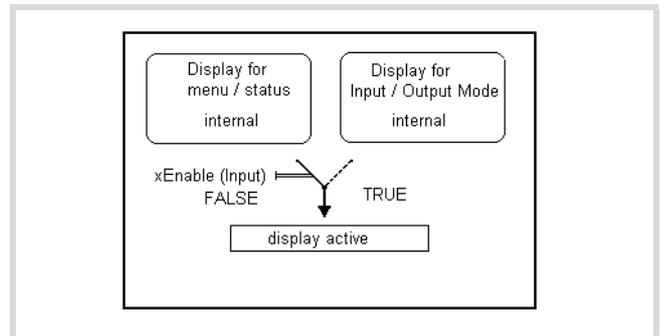
Figure 92: Changing from Status display ⟷ Entry/output mode

The program must process the function "Disp_EnableDisplay" (→ page 77) from the library EC_Visu2.lib continuously in order to activate Entry/output mode. The status at the xEnable function input determines the mode (→ figure 92):

FALSE: Status display of the PLC

TRUE: Entry/output mode

In Entry/output mode the display shows the values generated in the user program. The program continuously updates the values and accepts the entries made via the (rocker) buttons. You can use functions and function blocks to define the form of the variables, their presentation and positioning on the display:

The function block "Disp_DisplayElement" is used to show a variable on the display.

Function block "Disp_DisplayPage" is used to display a screen of 8 variables.

Function block "Disp_DisplayPage"

For each variable you define its use, i.e. text display or value entry.

The inputs of the function block must be parameterised for this purpose. Function block "Disp_DisplayPage" supports the cursor control function. If several variables requiring a value entry are shown, the first entry position is marked by a cursor. After the entry is completed, the cursor moves to the next position. An application involving several screens and the calling of one screen can be implemented in the user program. For this use the rocker buttons P1, P2, P3, P4 and the ESC, DEL, ALT and OK buttons for which the status can normally be scanned in the program.

Information on the menu display, current cursor position and button status is indicated by the outputs of the "Disp_GetDisplayInfo" function block.

The "Disp_DisplayElement" and "Disp_DisplayPage" function blocks are used for defining elements. The term "element" refers to the function blocks. An element is a variable that has additional properties such as the positioning on the display. The additional properties are defined by the parameters assigned to the function block inputs.

Other functions are shown in the function/function block overview.

Function/function block overview

The display in the Entry/output mode can be defined and controlled with the following functions/function blocks contained in the library "EC_Visu2.lib".

Function/ function block	Description
Disp_SetBacklight	Backlight of the PLC display on/off
Disp_RegisterVariable	Register variables
Disp_CreateVariableList	Define length of variables list
Disp_GetDisplayInfo	Scan display information
Disp_ClearScreen	Clear screen
Disp_SetCursor	Determine position and type of the cursor
Disp_SetContrast	Define the contrast of the local display
Disp_DisplayElement	Display of a single element / variable
Disp_DisplayPage	Display a screen with max. 12 elements for texts, values and for entering several values

The following table shows which display (PLC display or MFD-CP4) is controlled by the functions/function blocks.

Function/function block	Part no.	Display of
Disp_SetBacklight	Function	PLC
Disp_RegisterVariable	Function	–
Disp_CreateVariableList	Function	–
Disp_GetDisplayInfo	Function block	–
Disp_ClearScreen	Function	PLC, MFD[1]
Disp_SetCursor	Function block	PLC, MFD[1]
Disp_SetContrast	Function block	PLC
Disp_DisplayElement	Function block	PLC, MFD[1]
Disp_DisplayPage	Function block	PLC, MFD[1]

1) Only executable if function Disp_EnableDisplay active

Description of important functions / function blocks

FUNCTION Disp_EnableDisplay: BOOL (*Changing Status display <-> Entry/output mode*)

```
VAR_INPUT
xEnable:       (* FALSE: Status display, TRUE: Entry/output mode *)
xDisableESCKey: (*Enabling of ESC button on local display and MFD-CP4:
                   FALSE: Enable
                   TRUE: Button disabled *)
END_VAR
(* Return value: TRUE
*)
```

About xDisableESCKey:

Pressing the ESC button (requirement: ESC button must be
enabled) with Entry/output mode active will activate the Status
display. You can disable the ESC button by setting the input
xDisableESCKey to TRUE.

If the "Enable" input is set back to FALSE, the Status display of the
PLC will be displayed again.

FUNCTION Disp_RegisterVariable : BOOL (* Define an IEC variable as display variable *)

```
VAR_INPUT
  sName:     (* Symbolic name of display variable *)
  dwAddress: (* Address of associated IEC variable *)
  eVarTyp:   (* Data type of the associated IEC variable, see DISP_VARTYP*)
END_VAR
(* Return values:*)
(* TRUE: Display variable successfully registered*)
(* FALSE: Variable list full *)

TYPE DISP_VARTYP :
( DISP_TYP_USINT := 0,
  DISP_TYP_UINT,
  DISP_TYP_UDINT,
  DISP_TYP_SINT,
  DISP_TYP_INT,
  DISP_TYP_DINT,
  DISP_TYP_BYTE,
  DISP_TYP_WORD,
  DISP_TYP_DWORD,
  DISP_TYP_STRING ) := DISP_TYP_UINT;
END_TYPE
```

50 variables can be used. If you need more, this must be defined
via the "Disp_CreateVariableList" function.

FUNCTION_BLOCK Disp_GetDisplayInfo (* Actual information of the display status *)

```
VAR_OUTPUT
byMenueLevel :(*    Menu level:                                                  *)
             (*0:   Status menu                                                   *)
             (*1:   Main menu                                                     *)
             (*2:   Main menu / program                                          *)
             (*3:   Main menu / Set clock                                        *)
             (*4:   Main menu / information                                      *)
             (*5:   System menu                                                   *)
             (*6:   System menu/ Security                                        *)
             (*7:   System menu/ System                                          *)
             (*8:   System menu/ Start parameters                               *)
             (*9:   System menu / menu language                                 *)
             (*10:  System menu / Configuration                                 *)
             (*11- 14: Not used*)
             (*15:  Entry/output mode                                            *)
byActualLine:       (*Cursor position, line 1 - 4                               *)
byActualColumn:     (*Cursor position, column 1 - 16                            *)
xESCKeyDisabled:    (* FALSE: Press ESC button -> Status menu                   *)
                    (* TRUE: ESC button can be scanned in the user program      *)
xInputEnabled #)    (* TRUE: If inputs xEnable and xEnableInput of the FB Disp_DisplayPage = TRUE *)
                    (* FALSE: One input disabled                                 *)
xInputActive #)     If inputs xEnable and xEnableInput of the FB Disp_DisplayPage = TRUE and the ALT button is pressed*)
                    (* FALSE:Entry not active *)
(* #) When using FB Disp_DisplayPage *)
END_VAR
```

FUNCTION_BLOCK Disp_DisplayElement (*Display of a single element*)

```
VAR_INPUT
  xEnable:          (* Execution if input = TRUE *)
  sName         :   (* Symbolic element name *)
  byLine        :   (* Display element in line 1 - 4 *)
  byColumn      :   (* Display element in column 1 - 16 *)
  eFont         :   (* Font, only elements of type STRING ! See DISP_FONTS*)
  byDigits      :   (* Number of characters, only for numerical elements*)
  byPrecision   :   (* Number of characters after decimal point, only for numerical elements *)
  eAttribut     :   (* Element properties normal, reverse, flashing. See DISP_ATTRIBUT*)
END_VAR
VAR_OUTPUT
  eError          (* See DISP_ERROR*)
END_VAR
(* Return values:*)
(* DISP_ERROR_NO_ERROR:OK, no error*)
(* DISP_ERROR_INVALID_LINE: *)
(* DISP_ERROR_INVALID_COLUMN: outside of value range*)
(* DISP_ERROR_ELEMENT_NOT_FOUND: Element not found*)
(* DISP_ERROR_INVALID_VARIABLE_TYP:outside of value range*)

TYPE DISP_FONTS :
( DISP_FONT_LATIN1 := 0,
  DISP_FONT_LATIN2,
  DISP_FONT_CYRILLIC ) := DISP_FONT_LATIN1;
END_TYPE
...
```

```
...
TYPE DISP_ATTRIBUT :
( DISP_ATTR_NORMAL := 0,
  DISP_ATTR_REVERSE,
  DISP_ATTR_BLINK ) := DISP_ATTR_NORMAL;
END_TYPE
```

FUNCTION_BLOCK Disp_DisplayPage (* Display of a screen *)

```
VAR_INPUT
xEnable:                    (* TRUE: Activate display *)
xEnableInput:               (* TRUE: Activate Entry *)
byNoOfElements:             (* Number of elements for this screen 1 - 12*)
aElementDescription:ARRAY [1..12]   (* See DISP_ElementDescription*)
OF DISP_ElementDescription:
END_VAR
VAR_OUTPUT
byError
END_VAR
(* Return values:*)                 *
(* 0:        OK, all elements are displayed*)
(* 1 - 12: error on display of the element "n" or see DISP_ERROR_INVALID_NO_OF_ELEMENTS*)


TYPE DISP_ElementDescription :      (* Description of one display element *)
STRUCT
            xEnable         : (* TRUE Default setting: Element is displayed; FALSE: Display frozen*)
            xInputEnable    : (* FALSE: Display of element, see figure 93; TRUE: Display (initialisation) value,
                              Entry possible*)
            sName           : (* Symbolic element name *)
            byLine          : (* Display element in line 1 - 4 *)
            byColumn:         (* Display element in column 1 - 16 *)
            eFont           : (* Font, only elements of type STRING ! See DISP_FONTS*)
            byDigits        : (* Number of characters, only for numerical elements*)
            byPrecision     : (* Number of characters after decimal point, only for numerical elements *)
            diMinInputValue#:  (* Min value for entry value, only for numerical elements *)
            diMaxInputValue#:  (* Max value for entry value, only for numerical elements *)
            eAttribut:         (* Element properties normal, reverse, flashing. See DISP_ATTRIBUT*)
            xInputActiv#:      (* TRUE: If inputs xEnable and xEnableInput of the FB Disp_DisplayPage = TRUE*)
            xInputDone#:       (* TRUE: After completing value entry by pressing the
                              "OK" button. Must be reset by user to FALSE!*)
            eError          (* See DISP_ERROR*)
END_STRUCT
END_TYPE
END_VAR

                            # Active if xInputEnable = TRUE
(* Returnvalues: *)
(* DISP_ERROR_NO_ERROR,       OK, no error *)
(* DISP_ERROR_INVALID_LINE,   outside of value range: 1 - 4 *)
(* DISP_ERROR_INVALID_COLUMN, outside of value range: 1 - 16 *)
(* DISP_ERROR_ELEMENT_NOT_FOUND,  (* DISP_ERROR_ELEMENT_NOT_FOUND, Element not found*)
Element not found*)
```

Relationship between DISP_DisplayPage.xEnable/ xEnableInput and DISP_ElementDescription.xInputEnable for the value entry
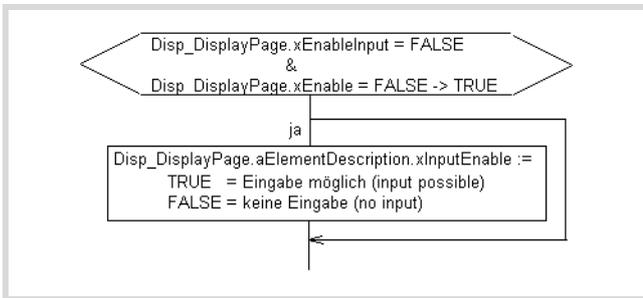
Figure 93:   Activate display/value entry

Value entry procedure
▶ Set the following in the program:

```
Disp_DisplayPage.xEnable = TRUE        (Display of
                                       values/modifications
                                       visible)

Disp_DisplayPage.xEnableInput = TRUE   (Entry enabled)
```

▶ Press the ALT button on the display.
The cursor appears on the first element "aElementDescription[1]" for which its xInputEnable is set to TRUE.

▶ Press the OK button.
The value is presented in the basic form of the data type, e.g. TYPE UINT : 00000)

▶ Use the cursor buttons to change the value:
  – Use the > or < button to select the position of the value.
  – Press the ∧"∨ button to change the value
▶ Confirm the entry with the OK button.
The cursor jumps to the next entry option, e.g. the second element.

Press the ALT button to return to Entry/output mode.

General programming procedure

• Declaration of the (display) variables to display entry in the list "Global_Variables_Display"
  → figure 95
• Program creation
  3 programs (for each example) are created:
  – Start program: generation of a start pulse (cycle 1)
  – PLC_PRG: User program with call of the program "Visualisation"
  – Visualisation: Program for presenting variables on the display

Structure of the program "Visualisation"
• In Cycle 1:
  – Define number of display variables -> Function Disp_CreateVariableList (only if more than 50 display variables are required!)
  – Registering of display variables -> Function Disp_RegisterVariable (general execution)

• In the subsequent cycle (depending on the application):
  – Clear display -> Function Disp_ClearScreen
  – Backlight of local display -> Function Disp_SetBacklight
  – Set cursor -> Function Disp_SetCursor
  – Define the contrast of the local display-> Function Disp_SetContrast
  – Define properties of the variables such as position-> FB Disp_DisplayElement or FB Disp_DisplayPage

• All cycles:
  – Start the display -> Start the FB Disp_DisplayElement or FB Disp_DisplayPage
  – Activate the display-> Start the function Disp_DisplayEnable
  – Scan the display states -> Function Disp_GetDisplayInfo

Example of text and values output

(with the Disp_DisplayElement FB) The display is required to display the values of the variables "motor1" and "motor2". The two values are changed continuously by the user program.

Figure 94: Example of text and values output

Operations via the PLC inputs
– I1 = FALSE: Status display
– I1 = TRUE: Entry/output mode
– I2 = FALSE: ESC button active
– I2 = TRUE: ESC button disabled
– I3 = TRUE: The first line is shown on the display.
– I5 = TRUE: The third line is shown on the display.

Execution
The example program consists of programs:

- STARTPROGRAM
  – The "startprogram" is called on system event "Start".
  – The auxiliary variable g_xFirstCycleAfterStartProgram is set.

- PLC_PRG
  – 2 values are incremented.
  – The program "Visualisation" is called.

- VISUALISATION
  – Registering and positioning of variables on the display in the first cycle
  – The auxiliary variable g_xFirstCycleAfterStartProgram is reset.
  – Activation of Entry/output mode (I1)
    Start display (I3,I5)

Declare variables
▶ First declare for each text element that you wish to display, such as "motor1", a variable of type "String" in the "Global_Variables_Display" list as shown in the following example (see alsofigure 95):

```
VAR GLOBAL
        g_sDisp_String1 :STRING:='Motor1';
END_VAR
```
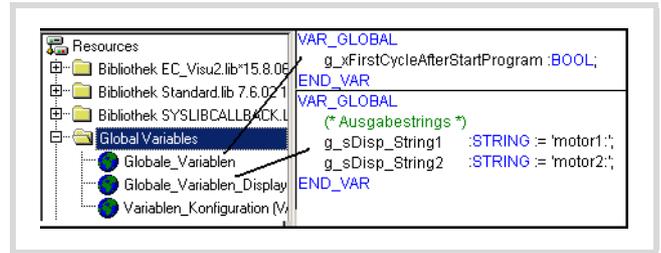

Figure 95: Declaration of display variables

Creating auxiliary variables
▶ For the first program cycle call "Startprogram" on system event "Start".
▶ Set an auxiliary variable "g_xFirstCycleAfterStartProgram" in this program that you reset after the first cycle is completed. The auxiliary variable must be declared globally ➞ figure 95.
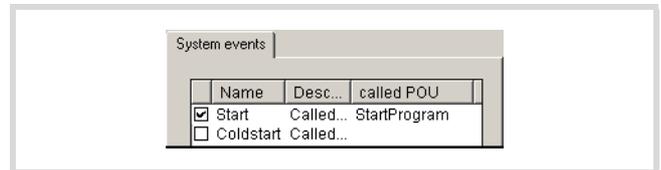
Figure 96: Defining the system event

Creating the program "StartProgram"
▶ Write the program "StartProgram" as in ➞ figure 97.
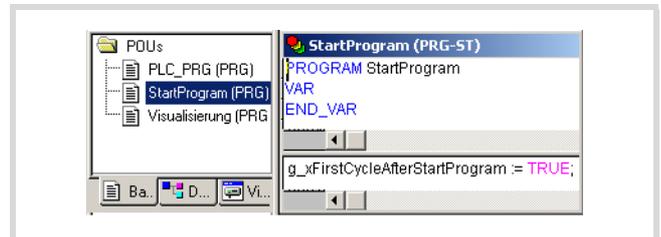
Figure 97: Creating the "startprogram"

Creating the "PLC_PRG" program

```
PROGRAM PLC_PRG
VAR
  fbTimer1                          :TON;
  (* Display values of the application
  *)
  byValue                           :BYTE;
  wValue                            :WORD;
END_VAR
--------------------------------------------------------
fbTimer1(IN:=NOT fbTimer1.Q , PT:=t#50ms );
IF fbTimer1.Q = TRUE THEN
  byValue := byValue + 1;
  wValue := wValue + 1;
END_IF


Visualisation(); (* Call visualisation *)
```

Creating the program "Visualisation"
Depending on the status of auxiliary variable
"g_xFirstCycleAfterStartProgram" register the variables for which
the text/value is to be displayed:

▶ Program the function "Disp_RegisterVariable" with the
following example parameters: Disp_RegisterVariable
(`S1',ADR(g_sDisp_String1), Disp_TYP_STRING). Here the
variable is assigned the name S1.
▶ To display a value (type Byte) program a variable with the
function call Disp_RegisterVariable ('V1', ADR(byValue),
Disp_TYP_BYTE).


```
FUNCTION Disp_RegisterVariable : BOOL
(* Register one IEC-Variable for using as display variable *)
VAR_INPUT                  (* Symbolic name for display variable *)
    sName      :STRING(16);  (* Address of corresponding IEC-variable *)
    dwAddress :DWORD;        (* Datatyp of corresponding IEC-variable *)
    eVarTyp    :DISP_VARTYP; (* Returnvalue *)
END_VAR
```

Figure 98: Function Disp_RegisterVariable

In this program section you can also define the position of the
variables on the display by specifying the Line and Column.
Call the function block (FB) "Disp_DisplayElement" and assign
parameters for the inputs sName, byLine, and byColumn, e.g.:

```
fbDisplayElement1.sName := 'S1';
fbDisplayElement1.byLine := 1;
fbDisplayElement1.byColumn := 1;
```

The element S1 with the text "motor1" would be displayed in the
first line starting from the first column.

In order to display/enter several elements call the function block
"Disp_DisplayElement" in the following program section that is
continuously processed and assign external inputs to the
"xEnable" inputs, e.g. I3.

```
VAR
    xIsDisplayEnabled: BOOL;
    fbGetDisplayInfo:             Disp_GetDisplayInfo;
    fbDisplayElement1: Disp_DisplayElement;
    fbDisplayElement2: Disp_DisplayElement;
    fbDisplayElement3: Disp_DisplayElement;
    fbDisplayElement4: Disp_DisplayElement;
    byError:                      BYTE;
    byValue:                      BYTE;
    wValue:                       WORD;
END_VAR
------------------------------------------------------------
(* Initialisation in the first cycle after program start *)
IF g_xFirstCycleAfterStartProgram = TRUE THEN
Disp_ClearScreen(xEnable:=TRUE);
Disp_RegisterVariable('S1', ADR(g_sDisp_String1),
DISP_TYP_STRING);
```

```
Disp_RegisterVariable('S2', ADR(g_sDisp_String2),
DISP_TYP_STRING);
Disp_RegisterVariable('V1', ADR(PLC_PRG.byValue),
DISP_TYP_BYTE);
Disp_RegisterVariable('V2', ADR(PLC_PRG.wValue),
DISP_TYP_WORD);

    fbDisplayElement1.sName         := 'S1';
    fbDisplayElement1.byLine        := 1;
    fbDisplayElement1.byColumn      := 1;

    fbDisplayElement2.sName         := 'S2';
    fbDisplayElement2.byLine        := 3;
    fbDisplayElement2.byColumn      := 1;

    fbDisplayElement3.sName         := 'V1';
    fbDisplayElement3.byLine        := 1;
    fbDisplayElement3.byColumn      := 8;
    fbDisplayElement3.byDigits      := 4;
    fbDisplayElement3.byPrecision   := 1;

    fbDisplayElement4.sName         := 'V2';
    fbDisplayElement4.byLine        := 3;
    fbDisplayElement4.byColumn      := 8;
    fbDisplayElement4.byDigits      := 6;
    fbDisplayElement4.byPrecision   := 1;

(* The first cycle is completed, reset flag *)

g_xFirstCycleAfterStartProgram := FALSE;
END_IF

xIsDisplayEnabled := Disp_EnableDisplay(I1, I2);
fbDisplayElement1( xEnable:= I3 );
fbDisplayElement2( xEnable:= I5 );
fbDisplayElement3( xEnable:= I3 );
fbDisplayElement4( xEnable:= I5);
```

▶ Start the programs.

Example of a screen output with texts and value entries

With the Disp_DisplayPage function block

The following display has to be implemented.

The contents of the variables MO11 and TEMP8 are changed continuously by the user program.

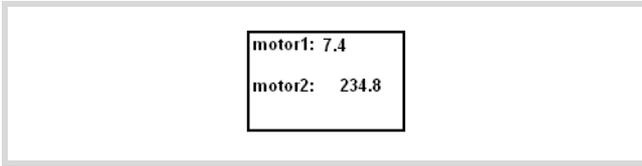| MO11	3.5
TIM14	0
MOZ14	0
TEMP8	183

Figure 99: Example of a screen for entries and outputs

Operations via the PLC inputs
- I1 = FALSE: Status display
- I1 = TRUE: Entry/output mode
- I2 = FALSE: ESC button active
- I2 = TRUE: ESC button disabled
- I3 = TRUE: The values are refreshed by the program.
- I4 = TRUE: Entry active.

Execution
The example program consists of programs:

- "Startprogram": (called on system event Start)
  - Auxiliary variable "g_xFirstCycleAfterStartProgram" is set.

- PLC_PRG:
  - 2 values are incremented.
  - The program "Visualisation" is called.

- VISUALISATION
  - Registering and positioning of variables on the display in the first cycle.
  - The auxiliary variable g_xFirstCycleAfterStartProgram is reset.
  - Activation of Entry/output mode (I1).
  - Enable ESC button (I2).
  - Start display (I3).
  - Start entry (I4).

Declaring display variables
▶ First declare for each text element that you wish to display, such as "MO11", a variable of type "String" in the "Global_Variables_Display" folder as in the following example:

```
VAR_GLOBAL
  g_sDisp_String1    :STRING := 'MO11 :';
  g_sDisp_String2    :STRING := 'TIM14 :';
  g_sDisp_String3    :STRING := 'MOZ14 :';
  g_sDisp_String4    :STRING := 'TEMP8 :' ;
END_VAR
```

▶ Create an auxiliary variable and write the program "Startprogram" as in the "Example of text and values output".
▶ Write the PLC_PRG and Visualisation programs as shown in the following example:

```
PROGRAM PLC_PRG (**********************)
VAR
    fbTimer1        :TON;
    (* Display values of the application *)
    byValue         :BYTE;
    wValue          :WORD;
    dwValue         :DWORD;
    usiValue        :USINT;
    siValue         :SINT;
END_VAR
--------------------------------------------------------
fbTimer1(IN:=NOT fbTimer1.Q , PT:=t#50ms );
IF fbTimer1.Q = TRUE THEN
    usiValue := usiValue + 1;
    byValue:=byValue+1;
END_IF

Visualisation(); (* Call visualisation *)
```

```
PROGRAM Visualization (***************)
VAR
    xIsDisplayEnabled               :BOOL;
    fbDisplayPage1                  :Disp_DisplayPage;
    byError                         :BYTE;
    siValue                         :SINT;
END_VAR
----------------------------------------------------------
(* Initialisation in the first cycle after program start *)
IF g_xFirstCycleAfterStartProgram = TRUE THEN

Disp_RegisterVariable('S1', ADR(g_sDisp_String1), DISP_TYP_STRING);
Disp_RegisterVariable('S2', ADR(g_sDisp_String2), DISP_TYP_STRING);
Disp_RegisterVariable('S3', ADR(g_sDisp_String3), DISP_TYP_STRING);
Disp_RegisterVariable('S4', ADR(g_sDisp_String4), DISP_TYP_STRING);
Disp_RegisterVariable('V1', ADR(PLC_PRG.byValue), DISP_TYP_BYTE);
Disp_RegisterVariable('V2', ADR(PLC_PRG.wValue), DISP_TYP_WORD);
Disp_RegisterVariable('V3',ADR(PLC_PRG.dwValue),
DISP_TYP_DWORD);
Disp_RegisterVariable('V4', ADR(PLC_PRG.usiValue), DISP_TYP_USINT);

fbDisplayPage1.aElementDescription[1].sName          := 'S1';
fbDisplayPage1.aElementDescription[1].byLine         := 1;
fbDisplayPage1.aElementDescription[1].byColumn       := 1;
fbDisplayPage1.aElementDescription[2].sName          := 'S2';
fbDisplayPage1.aElementDescription[2].byLine         := 2;
fbDisplayPage1.aElementDescription[2].byColumn       := 1;
fbDisplayPage1.aElementDescription[3].sName          := 'S3';
fbDisplayPage1.aElementDescription[3].byLine         := 3;
fbDisplayPage1.aElementDescription[3].byColumn       := 1;
fbDisplayPage1.aElementDescription[4].sName          := 'S4';
fbDisplayPage1.aElementDescription[4].byLine         := 4;
fbDisplayPage1.aElementDescription[4].byColumn       := 1;

fbDisplayPage1.aElementDescription[5].sName          := 'V1';
fbDisplayPage1.aElementDescription[5].byLine         := 1;
fbDisplayPage1.aElementDescription[5].byColumn       := 13;
fbDisplayPage1.aElementDescription[5].byDigits       := 4;
fbDisplayPage1.aElementDescription[5].byPrecision    := 1;
fbDisplayPage1.aElementDescription[5].xInputEnable   := FALSE;
fbDisplayPage1.aElementDescription[5].diMinInputValue := 1;
fbDisplayPage1.aElementDescription[5].diMaxInputValue := 100;
fbDisplayPage1.aElementDescription[6].sName          := 'V2';
fbDisplayPage1.aElementDescription[6].byLine         := 2;
fbDisplayPage1.aElementDescription[6].byColumn       := 12;
fbDisplayPage1.aElementDescription[6].byDigits       := 5;
fbDisplayPage1.aElementDescription[6].byPrecision    := 0;
fbDisplayPage1.aElementDescription[6].xInputEnable   := TRUE;
fbDisplayPage1.aElementDescription[6].diMinInputValue := 0;
fbDisplayPage1.aElementDescription[6].diMaxInputValue := 33333;
fbDisplayPage1.aElementDescription[7].sName          := 'V3';
```

```
fbDisplayPage1.aElementDescription[7].byLine          := 3;
fbDisplayPage1.aElementDescription[7].byColumn        := 8;
fbDisplayPage1.aElementDescription[7].byDigits        := 9;
fbDisplayPage1.aElementDescription[7].byPrecision     := 0;
fbDisplayPage1.aElementDescription[7].xInputEnable    := TRUE;
fbDisplayPage1.aElementDescription[7].diMinInputValue := 0;
fbDisplayPage1.aElementDescription[7].diMaxInputValue := 4444444;
fbDisplayPage1.aElementDescription[8].sName           := 'V4';
fbDisplayPage1.aElementDescription[8].byLine          := 4;
fbDisplayPage1.aElementDescription[8].byColumn        :=13;
fbDisplayPage1.aElementDescription[8].byDigits        := 4;
fbDisplayPage1.aElementDescription[8].byPrecision     := 0;
fbDisplayPage1.aElementDescription[8].xInputEnable    := TRUE;
fbDisplayPage1.aElementDescription[8].diMinInputValue := 4;
fbDisplayPage1.aElementDescription[8].diMaxInputValue := 400;


(* The first cycle is completed, reset flag *)
g_xFirstCycleAfterStartProgram := FALSE;
END_IF


xIsDisplayEnabled := Disp_EnableDisplay(I1, I2);
fbDisplayPage1( xEnable:= I3 , xEnableInput:= I4,
byNoOfElements:= 8, byError =>byError );


IF fbDisplayPage1.aElementDescription[7].xInputDone = TRUE THEN
siValue := PLC_PRG.siValue; (*Value for internal processing*)
fbDisplayPage1.aElementDescription[7].xInputDone     := FALSE;
END_IF
```

▶ Start the programs.

Multifunction display MFD-CP4 on the EC4-200

The multi-function display (MFD-CP4) enables you to implement externally the same display and operating functions available on the PLC.
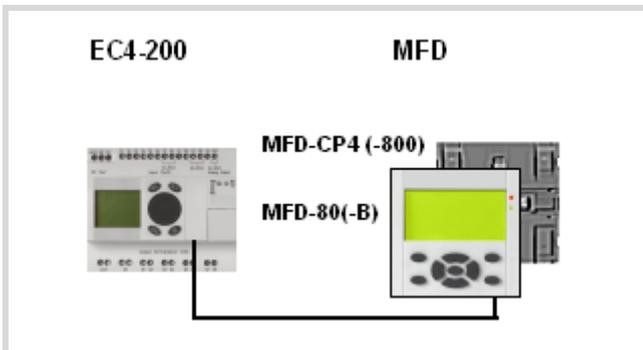

Figure 100: EC4-200 with MFD-CP4

When the power supply of the MFD-CP4 connected to the EC200-4 is switched on, it starts up in Terminal mode. In this mode it receives the information of the PLC display and shows it on the (MFD) display.

Switch the MFD-CP4 to Local mode in order to set its parameters. These parameters are as follows:

- Contrast
- Backlight
- Menu language: adaption of the parameter designations to the language
- COM interface
- ID=ID number 0, 1,..,8
  - 0: The MFD-CP4 communicates with the actual connected device.
  - 1…8: ID of the easyNet stations:
    Station selection (ID) on the easyNet
    If the EC4-200 is a station on the easyNet, the MFD-CP4 can communicate with the selected station via the EC4-200.
- Baud rate: 9600 (19200) baud

Press the "*" button to switch the MFD-CP4 between Terminal mode and Local mode (present only on MFD-CP4).

Changing to Terminal mode can only be carried out from the main menu of Local mode.

See also MFD-CP4 manual (MN05013011Z-EN; previously AWB2528-1548GB), chapter "Settings".

Main menu: COM…
MENU LANGUAGE…
LIGHTING: 80%
CONTRAST: +1
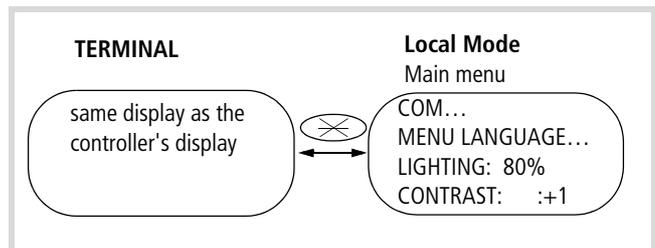
Figure 101: Toggling Terminal mode ↔ Local mode

In Local mode the MFD display buttons are active.
See MFD-CP4 power supply/communication module manual (MN05013011Z-EN; previously AWB2528-1548GB).

MFD setup

The MFD-CP4 is an assembled unit. The actual display, the HMI unit MFD-80(-B), is designed for mounting on the front of a control cabinet door. It is snap fitted onto the MFD-CP4(-800) power supply/communication module which is fastened on the back. The connection to the EC4-200 (multi-function interface) is implemented with the MFD-CP4-800-CAB5 cable.

Further information on handling, connecting and technical data of the device is provided in the operating manual of the MFD-CP4 power supply/communication module (MN05013011Z-EN; previously AWB2528-1548GB).

17 EC4-200 network modules

The EASY205-ASI, EASY221-CO, EASY204-DP, EASY222-DN network interfaces enable you to connect the EC4-200 as a slave to ASI, CAN, PROFIBUS-DP or DeviceNet (→ table 18) networks. The controller can also be integrated as a station in an easyNet network.
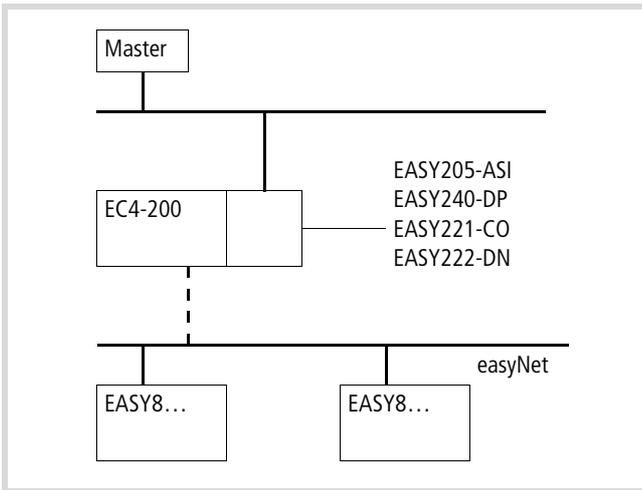
Figure 102: EC4-200 wih network connection

The type of data exchange between master and network interfaces is shown in table 18.

Table 18: Overview of network interfaces

Network connection	Network	Data exchange
EASY205-ASI	ASi	cyclic
EASY204-DP	PROFIBUS-DP	Cyclic + acyclic
EASY221-CO	CANopen	Cyclic + acyclic
EASY222-DN	DeviceNet	Cyclic + acyclic

→ The network modules in conjunction with the easy800 are described in detail in separate manuals (→ table 19). These manuals also apply to the EC4-200 connected to the network modules since this controller operates exactly like the easy800. The following sections on the individual network modules therefore only cover the differences between them and any particular procedures.

Table 19: Manuals on the network modules

Part no.	Manual (MN; previously AWB)
EASY204-DP	MN05013005Z-EN (previously AWB2528-1401GB)
EASY221-CO	MN05013008Z-EN (previously AWB2528-1479GB)
EASY222-DN	MN05013007Z-EN (previously AWB2528-1427GB)

EASY205-ASI

Cyclic data exchange

The master sends 4 bits to the EASY200-ASI network module connected to the EC205-8: 4 bits of output data and 4 parameter bits. It receives 4 bits of input data from the EC200-4.
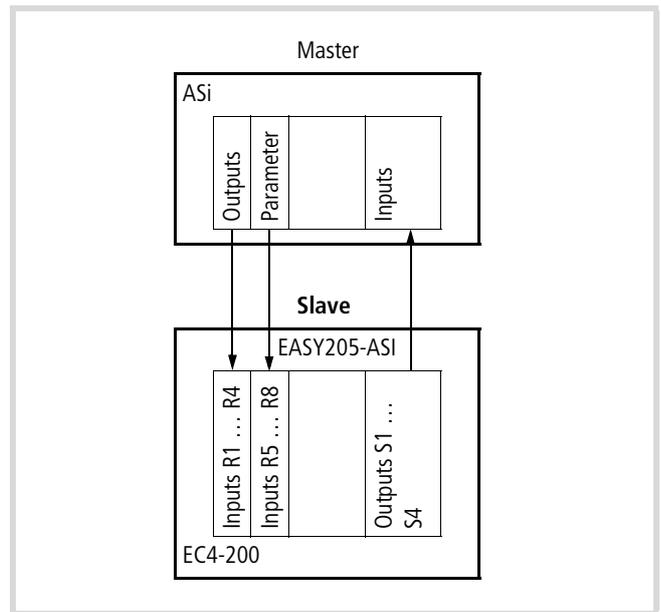
Figure 103: Cyclic data exchange of the EASY205-ASI

Table 20: Input/output data of the EC4-200

Master → EC4			EC4 → Master		
Master outputs	Q0 → R1	EC4 inputs	EC4 outputs	S1 → I0	Master inputs
	Q1 → R2			S2 → I1	
	Q2 → R3			S3 → I2	
	Q3 → R4			S4 → I3	
Master parameters	P0 → R5	EC4 inputs			
	P1 → R6				
	P2 → R7				
	P3 → R8				

Configuration

The configuration is carried out in the PLC configuration of the easy Soft CoDeSys programming software. The network module is entered as an expansion device in the configuration tree. This contains predefined input and output channels (R1…R8, S1…S4) for the cyclical transfer of data.

Figure 104: Configuring EASY205-ASI

Setting the station address

The EASY205-ASI is assigned a station address with an external programming device.

EASY221-CO, EASY204-DP, EASY222-DN

The procedure for data exchange between the EASY network modules and a master is described in detail in separate manuals, → table 19.

Cyclic data exchange

The EASY204-DP, EASY221-CO, EASY222-DN network modules have the same cyclical data exchange procedure.

The master exchanges 3 bytes of data in each direction with the network modules connected to the EC4-200.
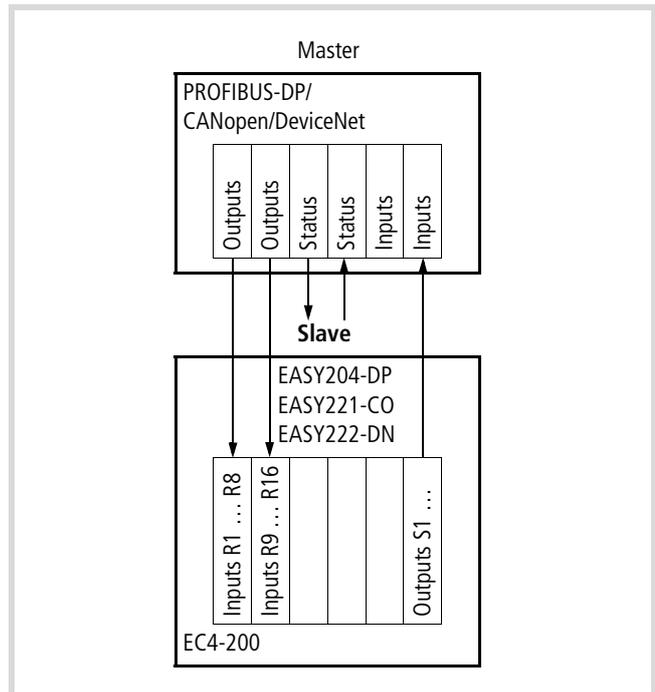
Figure 105: Cyclical data exchange between master EASY204-DP, EASY221-CO, EASY222-DN

From the point of view of the master, this data is written to the EC4-200.

Byte	Meaning[1]
0	Status (e. g. RUN/HALT)
1	R9 … R16 (Inputs)
2	R1 … R8 (Inputs)

1) The meaning of the bits, e.g. a bit of byte 0 indicates RUN/HALT status, described in separate manuals, → table 19.

From the point of view of the master, this data is read from the EC4-200.

Byte	Meaning[1]
0	Status (e. g. RUN/HALT)
1	S1 … S8 (Outputs)
2	Not used

1) The meaning of the bits, e.g. a bit of byte 0 indicates RUN/HALT status, described in separate manuals, → table 19.

Configuration

The configuration is carried out in the PLC configuration of the easy Soft CoDeSys programming software. The network module is entered as an expansion device in the configuration tree. This contains predefined input and output channels (R1…R16, S1…S8) for the cyclical transfer of data.
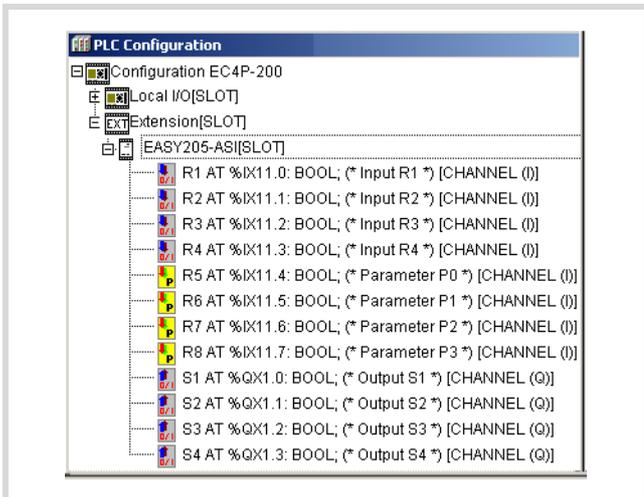
Setting the station address

The station address of the network module is set in a special parameter dialog in the PLC configuration. The address is transferred to the module when the program is downloaded and when the boot project is loaded.

The address set in the PLC configuration can be overwritten in the Startup.ini file. The associated entry in the Startup.ini file is:

`EXTENSION_SLAVE_ADDRESS = <Address>`

Figure 106 is an example of where to enter the address for the EASY204-DP.

Figure 106: Address entry

→ Note on EASY204-DP:

You can only change the bus address if communication with the master is not active.

Once a module is assigned a valid address, it saves it internally and loads it with every restart. If you set a new address in the PLC configuration and carry out a program download, this address will only be loaded if there was no communication with the master during the download, i.e. by unplugging the DP bus cable!

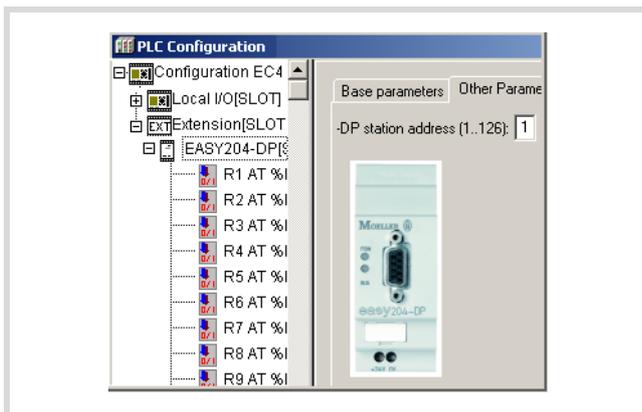If you load the program from the PC to the PLC, this will check whether the address currently used by the PLC matches the configured address. A warning message is generated if these are not the same.

Acyclic data exchange

Acyclical data exchange enables access to the defined objects of the EC4-200. These objects are a subset of the objects supported by easy800/MFD.

The objects listed in the table are supported by the EC4-200 and can be addressed by a master on the CAN, PROFIBUS-DP or DeviceNet.

Table 21: Objects of the EC4-200

Object name	Access type (R/W)
Mode[1]	R/W
Identification (only with EASY204-DP)	R
Inputs I1 … I16	R
Analog inputs I7, I8, I11, I12[2]	R
Inputs R1 … R16[1]	R
Outputs Q1 … Q8	R
Analog output QA1	R
Outputs S1 … S8[1]	R
Local diagnostics ID1-ID16[3]	R
Inputs of network stations IW1…IW8[3]	R
Inputs of network stations RW1…RW8[3]	R
Outputs of network stations QW1…QW8[3]	R
Outputs of network stations SW1…SW8[3]	R
Receive data of network stations RNW1…RNW8[3]	R
Send data of network stations SNW1…SNW8[3]	R
Bit markers M1…M96[4]	R/W
Byte markers MB1…MB96[4]	R/W
Marker words MW1 … MW96[4]	R/W
Marker double words MD1 … MD96[4]	R/W
8 bytes data (MD67 - MD68)[4]	R/W
16 bytes data (MD69…MD72)[4]	R/W
32 bytes data (MD73…MD80)[4]	R/W
64 bytes data (MD81…MD96)[4]	R/W

1) With PROFIBUS-DP only for class 2 master
2) IA1…IA4 in the operating manuals → page 87
3) Network station means the stations on the easyNET network!
4) The easy800/MFD markers are mapped to the EC4P-200 as shown in table 22.

Accessing other easy800/MFD objects causes an error message.

→ Only with EASY204-DP: The data and markers in word and double word format are transferred in Motorola format. Byte swapping does not occur!

Start addreses for inputs/outputs and markers

The start addresses for the address ranges of the inputs and outputs can be set in the PLC configuration. The marker range is shown in table 22.
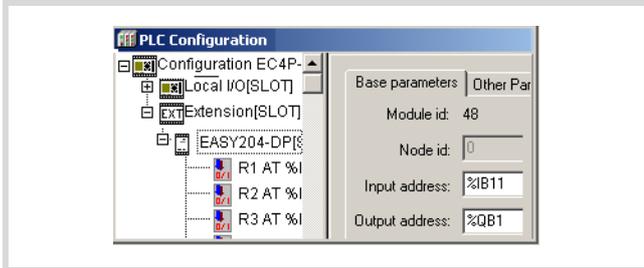

Figure 107: Setting the address ranges

The configuration and setting of the station address was already described in the section "Cyclical data exchange".

Table 22: Mapping of the EASY800 marker range to the EC4-200 (values of the EC4-200 shown in brackets)

Bit	96–89 (11.7–11.0)	88–81 (10.7–10.0)	80–73 (9.7–9.0)	72–65 (8.7–8.0)	64–57 (7.7–7.0)	56–49 (6.7–6.0)	48–41 (5.7–5.0)	40–33 (4.7–4.0)	32–25 (3.7–3.0)	24–17 (2.7–2.0)	16–9 (1.7–1.0)	8–1 (0.0–0.7)
Byte	12 (11)	11 (10)	10 (9)	9 (8)	8 (7)	7 (6)	6 (5)	5 (4)	4 (3)	3 (2)	2 (1)	1 (0)
Word	6 (10)		5 (8)		4 (6)		3 (4)		2 (2)		1 (0)	
DWord	3 (8)				2 (4)				1 (0)			
Byte	24 (23)	23 (22)	22 (21)	21 (20)	20 (19)	19 (18)	18 (17)	17 (16)	16 (15)	15 (14)	14 (13)	13 (12)
Word	12 (22)		11 (20)		10 (18)		9 (16)		8 (14)		7 (12)	
DWord	6 (20)				5 (16)				4 (12)			
Byte	36 (35)	35 (34)	34 (33)	33 (32)	32 (31)	31 (30)	30 (29)	29 (28)	28 (27)	27 (26)	26 (25)	25 (24)
Word	18 (34)		17 (32)		16 (30)		15 (28)		14 (26)		13 (24)	
DWord	9 (32)				8 (28)				7 (24)			
Byte	48 (47)	47 (46)	46 (45)	45 (44)	44 (43)	43 (42)	42 (41)	41 (40)	40 (39)	39 (38)	38 (37)	37 (36)
Word	24 (46)		23 (44)		22 (42)		21 (40)		20 (38)		19 (36)	
DWord	12 (44)				11 (40)				10 (36)			
Byte	60 (59)	59 (58)	58 (57)	57 (56)	56 (55)	55 (54)	54 (53)	53 (52)	52 (51)	51 (50)	50 (49)	49 (48)
Word	30 (58)		29 (56)		28 (54)		27 (52)		26 (50)		25 (48)	
DWord	15 (56)				14 (52)				13 (48)			
Byte	72 (71)	71 (70)	70 (69)	69 (68)	68 (67)	67 (66)	66 (65)	65 (64)	64 (63)	63 (62)	62 (61)	61 (60)
Word	36 (70)		35 (68)		34 (66)		33 (64)		32 (62)		31 (60)	
DWord	18 (68)				17 (64)				16 (60)			
Byte	84 (83)	83 (82)	82 (81)	81 (80)	80 (79)	79 (78)	78 (77)	77 (76)	76 (75)	75 (74)	74 (73)	73 (72)
Word	42 (82)		41 (80)		40 (78)		39 (76)		38 (74)		37 (72)	
DWord	21 (80)				20 (76)				19 (72)			
Byte	96 (95)	95 (94)	94 (93)	93 (92)	92 (91)	91 (90)	90 (89)	89 (88)	88 (87)	87 (86)	86 (85)	85 (84)
Word	48 (94)		47 (92)		46 (90)		45 (88)		44 (86)		43 (84)	
DWord	24 (92)				23 (88)				22 (84)			

Word	54 (106)	53 (104)	52 (102)	51 (100)	50 (98)	49 (96)
DWord	27 (104)		26 (100)		25 (96)	
Word	60 (118)	59 (116)	58 (114)	57 (112)	56 (110)	55 (108)
DWord	30 (116)		29 (112)		28 (108)	
Word	66 (130)	65 (128)	64 (126)	63 (124)	62 (122)	61 (120)
DWord	33 (128)		32 (124)		31 (120)	
Word	72 (142)	71 (140)	70 (138)	69 (136)	68 (134)	67 (132)
DWord	36 (140)		35 (136)		34 (132)	
Word	78 (154)	77 (152)	76 (150)	75 (148)	74 (146)	73 (144)
DWord	39 (152)		38 (148)		37 (144)	
Word	84 (166)	83 (164)	82 (162)	81 (160)	80 (158)	79 (156)
DWord	42 (164)		41 (160)		40 (156)	
Word	90 (178)	89 (176)	88 (174)	87 (172)	86 (170)	85 (168)
DWord	45 (176)		44 (172)		43 (168)	
Word	96 (190)	95 (188)	94 (186)	93 (184)	92 (182)	91 (180)
DWord	48 (188)		47 (184)		46 (180)	
DWord	51 (200)		50 (196)		49 (192)	
DWord	54 (212)		53 (208)		52 (204)	
DWord	57 (224)		56 (220)		55 (216)	
DWord	60 (236)		59 (232)		58 (228)	
DWord	63 (248)		62 (244)		61 (240)	
DWord	66 (260)		65 (256)		64 (252)	
DWord	69 (272)		68 (268)		67 (264)	
DWord	72 (284)		71 (280)		70 (276)	
DWord	75 (296)		74 (292)		73 (288)	
DWord	78 (308)		77 (304)		76 (300)	
DWord	81 (320)		80 (316)		79 (312)	
DWord	84 (332)		83 (328)		82 (324)	
DWord	87 (344)		86 (340)		85 (336)	
DWord	90 (356)		89 (352)		88 (348)	
DWord	93 (368)		92 (364)		91 (360)	
DWord	96 (380)		95 (376)		94 (372)	

Appendix

Network CAN/easyNet

Accessories

- RJ45 plug, Type: EASY-NT-RJ45 (8-pole)

→ Pre-assembled cables have RJ45 plugs at both ends.

Table 23: Prefabricated cables

Cable length cm	Part no.
30	EASY-NT-30
80	EASY-NT-80
150	EASY-NT-150

- User-assembled cable, part no.: EASY-NT-CAB
  (100 m 4 × 0.18 mm²)
- Crimping tool for RJ45 plug, Type: EASY-RJ45-TOOL.
- Bus terminating resistor, Type: EASY-NT-R
  RJ45 plug with integrated bus terminating resistor 120 Ω

Cable length with cross-sections

For correct operation of the network the cable lengths, cross-sections and cable resistances must match thoses listed in the following table.

Cable length m	Cable resistance mΩ/m	Cross-section mm²	AWG
up to 40	≦ 140	0.13	26
up to 175	≦ 70	0.25 to 0.34	23, 22
up to 250	≦ 60	0.34 to 0.5	22, 21, 20
up to 400	≦ 40	0.5 to 0.6	20, 19
up to 600	≦ 26	0.75 bis 0.8	18
up to 1000	≦ 16	1.5	16

The impedance of the cables used must be 120 Ω.

→ Further information on the CAN cable lengths and terminals can be obtained from the ISO standard 11898.

Calculating the cable length for a known cable resistance

If the resistance of the cable per unit of length is known (resistance per unit length R' in Ω/m), the entire cable resistance R_L must not exceed the following values. R_L depends on the selected baud rate:

Baud rate Kbaud	Cable resistance R_L Ω
10 to 125	≦ 30
250	≦ 25
500	≦ 12

l_{max} = maximum cable length in m
$R\infty$ = Total cable resistance in $_O$
R' = Cable resistance per unit length in Ω/m

$$l_{max} = \frac{R_L}{R'}$$

Calculating cross-section with known cable lengths

The minimum cross-section is determined for the known maximum extent of the network.

l = cable length in m
S_{min} = minimum cable cross-section in mm²
ρ_{cu} = specific resistance of copper, if not stated otherwise, 0.018 Ωmm²/m

$$S_{min} = \frac{l \times \rho_{cu}}{12.4}$$

→ If the calculation result does not correspond to a standard cross section, take the next higher cross section.

Calculating length with known cable cross-section

The maximum cable length for a known cable cross-section is calculated as follows:

l_{max} = cable length in m
S = cable cross-section in mm²
ρ_{cu} = specific resistance of cooper, if not stated otherwise, 0.018 Ωmm²/m

$$l_{max} = \frac{S \times 12.4}{\rho_{cu}}$$

Example program for PLC START/STOP using external switch

The SysLibPlcCtrl.lib library contains the function SysStartPlcProgram required for the start, and the function SysStopPlcProgram required for the stop.

In this case, the startup behaviour of the controller must be set to WARM START in the PLC Configurator under <Other Parameters → Settings>!

Function
The POU "StartPrg", which is called once on every PLC start is used to register the function FuncCalledWhenPlcIsInStop on the event "EVENT_TASKCODE_NOT_CALLED"-. This registration causes the function FuncCalledWhenPlcIsInStop to be called via the event "EVENT_TASKCODE_NOT_CALLED" if the PLC is in STOP state. The StartStopFunction function is used to monitor the status of the input and call the function for starting or stopping the PLC if there is a status change.

As the POU "StartPrg" is called only once, there should be no outputs or parameters set in this POU. User programs should be programmed in separate POUs.

▶ Activate the system event "Start" and name the Called POU "Startprg".

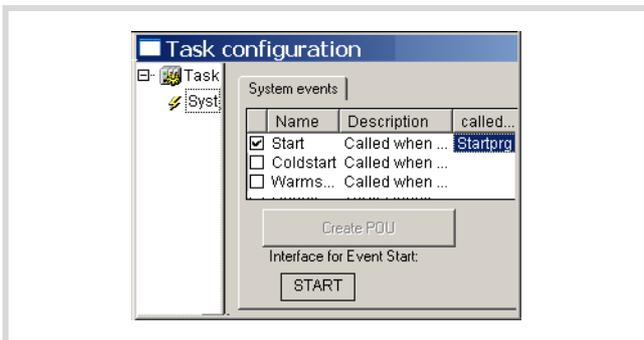Figure 108: Activating a system event

▶ Open a new POU with the name "Startprg" in the POUs folder and program the function SysCallbackRegister which "presents" the Start/Stop functions to the operating system.
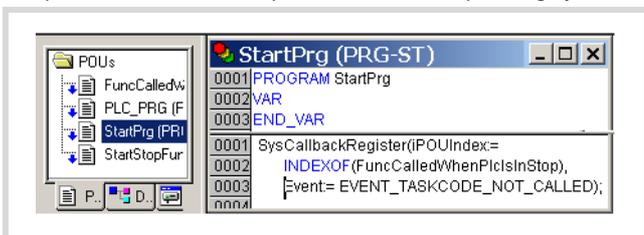

Figure 109: "Startprg" function

▶ Declare the following global variables.

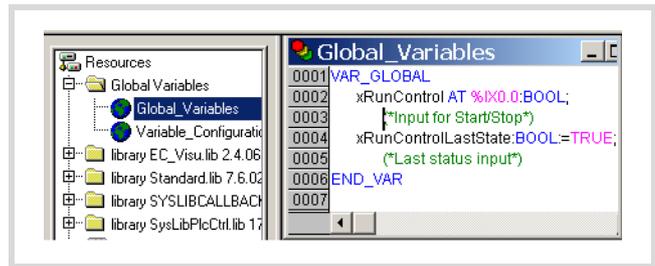Figure 110: Declaring global variables

▶ Enter the program for PLC_PRG as shown in figure 111. It is important that the user program and the POU calls are inserted as shown in figure 111.
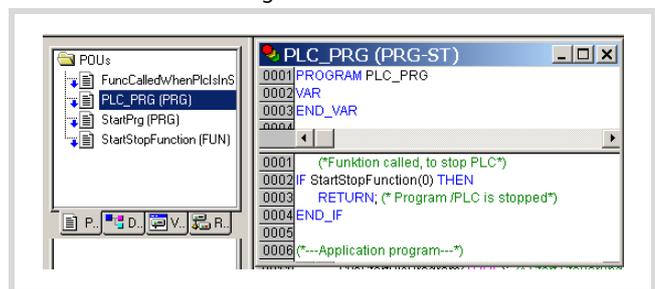

Figure 111: Scanning START/STOP

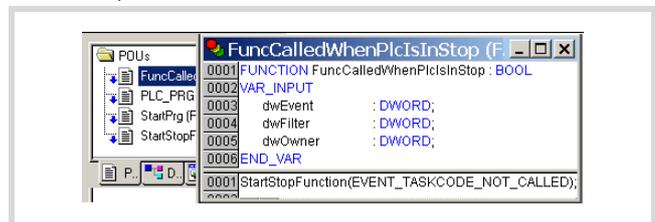▶ Enter the function FuncCalledWhenPlcInStop and StartStopFunction.

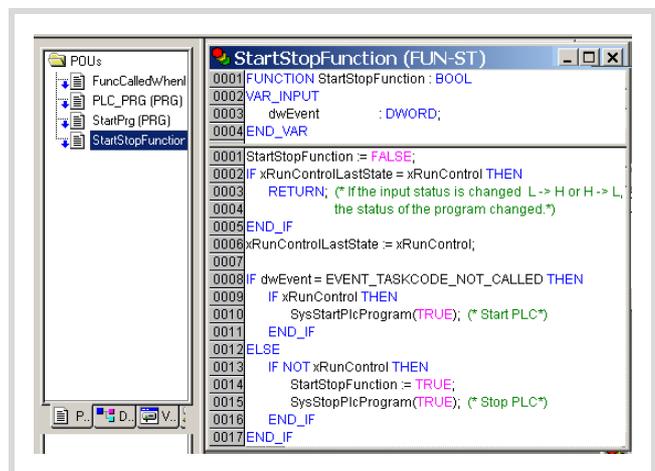
Figure 112: Call of the function FuncCalledWhenPlcIsInStop

Figure 113: Function that monitors the input

easy800-PC-CAB connection cable

9-pole socket connector on the cable (Terminal/PC plug)

Pin	Signal
2	RxD
3	TxD
4	DTR
5	GND
7	RTS
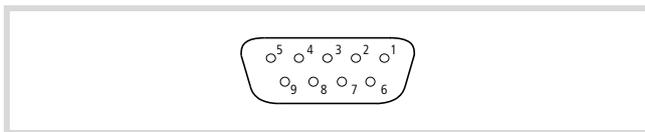
Figure 114: 9-pole socket connector

→ The RTS signal must be set for the cable to function as the voltage on the RTS cable supplies the components in the plug.

Dimensions and weight

Dimensions W × H × D	
[mm] with adapter for MMC	107.5 × 90 × 72 107.5 × 90 × 79
[inches] with adapter for MMC	4.23 × 3.54 × 2.84 4.23 × 3.54 × 3.11
Space units (SU) width	6
Weight	
[g]	320
[lb]	0.705
Mounting	Top-hat rail to DIN 50022, 35 mm or screw mounting with 3 ZB4-101-GF1 mounting feet
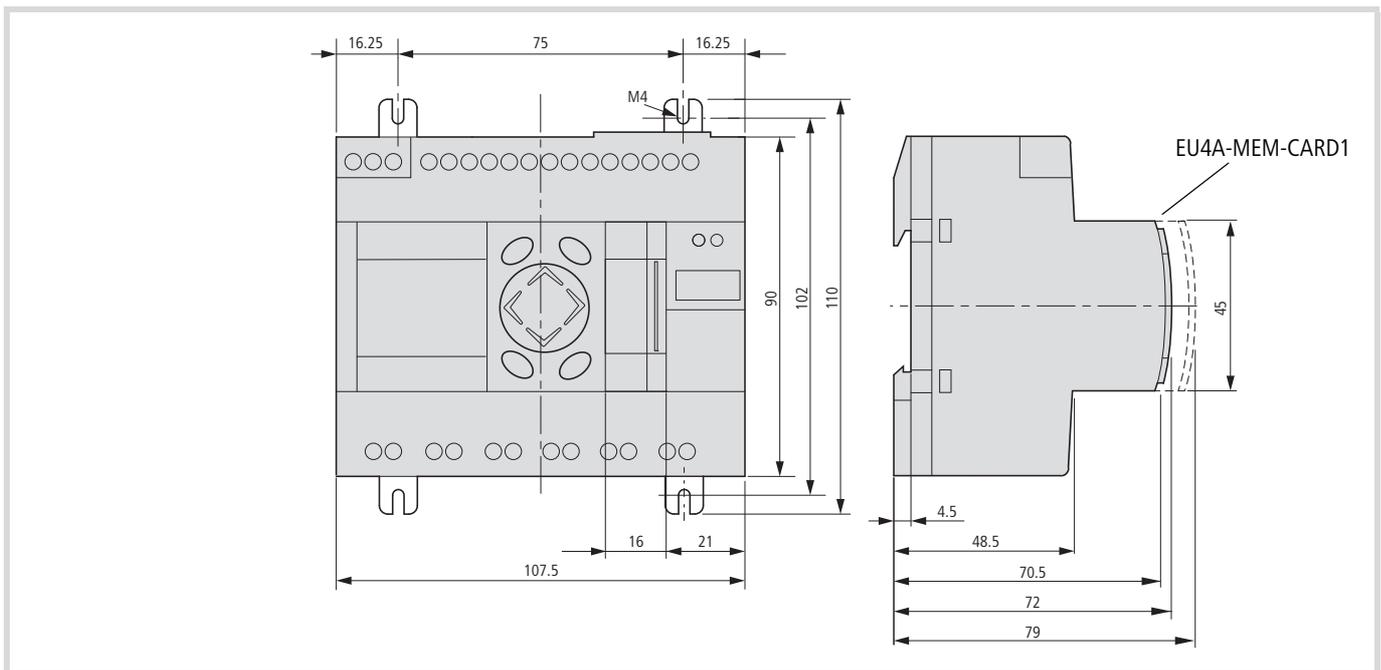
Figure 115: Dimensions in mm (specified in inches → table 24)

Table 24: Dimensions in inches

mm	inches	mm	inches
4.5	0.177	79	3.11
16.25	0.64	90	3.54
48.5	1.91	102	4.01
70.5	2.78	107.5	4.23
72	2.83	110	4.33
75	2.95		

Technical data

Climatic environmental conditions (Cold to IEC 60068-2-1, Heat to IEC 60068-2-2)		
Operational ambient temperature Installed horizontally/vertically	°C, (°F)	–25 to 55, (–13 to 131)
Condensation		Prevent condensation with suitable measures
LCD display (reliably legible)	°C, (°F)	0 to 55, (–32 to 131)
Storage/transport temperature	°C, (°F)	–40 to 70, (–40 to 158)
Relative humidity (IEC 60068-2-30), non-condensing	%	5 to 95
Air pressure (in operation)	hPa	795 to 1080
Ambient mechanical conditions		
Degree of protection (IEC/EN 60529)		IP20
Vibrations (IEC/EN 60068-2-6)		
Constant amplitude 3.5 mm	Hz	5 to 9
constant acceleration 1 g	Hz	9 to 150
Shock (IEC/EN 60068-2-27) Sinusoidal 15 g/11 ms	Shocks	18
Drop (IEC/EN 60068-2-31) Drop height	mm	50
Free fall, packaged (IEC/EN 60068-2-32)	m	1
Mounting position		horizontal,vertical
Electromagnetic compatibility (EMC)		
Electrostatic discharge (ESD), (IEC/EN 61000-4-2, severity level 3)		
Air discharge	kV	8
Contact discharge	kV	6
Electromagnetic fields (RFI), (IEC/EN 61000-4-3)	V/m	10
Radio interference suppression Limit value class		EN 55011, EN 55022 Class B
Fast transient burst (IEC/EN 61000-4-4, severity level 3)		
Power cables	kV	2
Signal cables	kV	2
Surge (IEC/EN 61000-4-5, degree of severity 2)	kV	0.5 symmetrical 1 asymmetrical
Line-conducted interference (IEC/EN 61000-4-6)	V	10
Insulation resistance		
Clearance in air and creepage distances		EN 50178
Insulation resistance		EN 50178
Overvoltage category/degree of pollution		II/2
Tools and cable cross-sections		
Solid, minimum to maximum	mm²	0.2 to 4
	AWG	22 to 12
Flexible with ferrule, minimum to maximum	mm²	0.2 to 2.5
	AWG	22 to 12
Factory wiring:	AWG	30

Slot-head screwdriver, width	mm	3.5 × 0.8
	inch	0.14 × 0.03
Tightening torque	Nm	0.6
CPU		
Memory specifications		
Program code	kByte	256
Program data	kByte	14 segments of 16 KB each
Marker/Input/Output/Retain data	kByte	16/4/4/8
Cycle time for 1 k instructions		< 0.3
Back-up/Accuracy of the real-time clock		
Back-up of the clock		 ① backup time in hours ② service life in years
Accuracy of the real-time clock		
Per day	s/day	± 5
Per year	h/year	± 0.5
Interfaces		
Programming interface		
Terminals		RJ45, 8-pole
RS232 (without control lines)		
PLC port		COM1
Potential isolation		none
Programming mode		
Transfer rate		4.8, 9.6, 19.2, 38.4, 57.6
Character format		8 data bits, no parity, 1 Stop bit
Transparent mode		
Transfer rate		0.3, 0.6, 1.2, 2.4, 4.8, 9.6, 19.2, 38.4, 57.6
Character format		8E1, 8O1, 8N1, 8N2, 7E2, 7O2, 7N2, 7E1
Number of transmission bytes in a block		190
Number of received bytes in a block		190
Ethernet		
Transfer rate	MBit/s	10
Potential isolation		yes
Multi-function interface (RS232) without control cables		
Transparent mode		
PLC port		COM2
Potential isolation		Yes, in the easy800-PC-CAB cable
Terminals		easy800-PC-CAB cable
Transfer rate	kBit/s	9.6, 19.2

CAN(open)/easyNet		
Transfer rate	kBit/s	10, 20, 50, 100, 125, 250, 500 Default: 125
Potential isolation from inputs/outputs/power supply		yes
Bus termination resistor		120 Ω or EASY-NT-R plug (incl. bus terminating resistor 120 Ω)
Terminals		2 x RJ45, 8pole
CAN(open) operating mode:		
– Station	Number	max. 126
– PDO type		Asynchronous, cyclic, acyclic
– Device profile		to DS301V4
easyNet mode:		
– Station	Number	max. 8

Power supply		
Rated voltage		
Nominal value	V DC, (%)	24, (–15, +20)
Permissible range	V DC	20.4 to 28.8
Residual ripple	%	$\leqq 5$
Input current at 24 V DC, typical	mA	140
Voltage dips, IEC/EN 61131-2	ms	10
Power loss at 24 V DC, typical	W	3.4

Inputs		
Digital inputs		
Number		12
Inputs that can be used for analog signals		I 7,8,11,12
Inputs that can be used for pulse signals (High-speed counters)		I 1,2,3,4
Inputs for interrupt generation		I 1,2,3,4
Status indication		LC display
Potential isolation		
from power supply, PC interface		No
Between each other		No
from the outputs, to CAN interfaces		Yes
Rated voltage		
Nominal value	V DC	24
At signal "0"		
I1 to I6 and I9 to I10	V DC	< 5
I7, I8, I11, I12	V DC	< 8
At signal "1"		
I1 to I6 and I9 to I10	V DC	> 15
I7, I8, I11, I12	V DC	> 8
Input current on "1" signal (at 24 V DC)		
I1 to I6, I9 to I10	mA	3.3
I7, I8, I11, I12	mA	2.2

Delay time from 0 to 1		
I1 to I4	ms	0.02
I5 to I12	ms	0.25
Delay time from 1 to 0		
I1 to I4	ms	0.02
I5 to I12	ms	0.25
Cable length (unshielded)	m	100
Additional input functions		
Inputs for analog signals		
Number		4 (I7, I8, I11, I12)
Signal range	V DC	0 to 10
Resolution, analog	V	0.01
Resolution, digital	Bit	10
	Value	0 to 1023
Input impedance	kΩ	11.2
Accuracy of actual value		
Two devices	%	± 3
Within a single device	%	± 2
Input current	mA	< 1
Cable length (shielded)	m	30
Inputs for high-speed counters		I 1, I2
Number/value range	Bit	2 × 16 bit (I1,I2) 1 × 32 bit (I1)
Max. frequency	kHz	50
Count direction selectable via software		incrementing/decrementing
Cable length (shielded)	m	20
Pulse shape		Square
Mark-to-space ratio		01:01
Incremental counter		I1, I2, I3, I4
Quantity		1
Value range	Bit	32 Bit
Max. frequency	kHz	40
Cable length (shielded)	m	20
Pulse shape		Square
Counter inputs		I1,I2 = Counter input I3 = Reference pulse I4 = Reference window
Signal offset		90°
Mark to space ratio		01:01
Inputs for interrupt generation		I1, I2, I3, I4
Max. frequency	kHz	3

Relay outputs		
Number of outputs		6
Parallel switching of outputs to increase performance		Not permissible
Protection of an output relay		
Miniature circuit-breaker B16	A	16
or fuse (slow-blow)	A	8
Electrical isolation		Yes
Safe isolation	V AC	300
Basic insulation	V AC	600
Mechanical lifespan	Switch operations	10×106
Contacts relays		
Conventional therm. current	A	8
Recommended for load at 12 V AC/DC	mA	> 500
Protected against short-circuit cos $\varphi = 1$ Characteristic B (B16) at 600 A	A	16
Protected against short-circuit cos $\varphi = 0.5$ bis 0.7 Characteristic B (B16) at 900 A	A	16
Rated impulse withstand voltage U_{imp} contact coil	kV	6
Rated insulation voltage U_i		
Rated operational voltage U_e	V AC	250
Safe isolation to EN 50178 between coil and contact	V AC	300
Safe isolation to EN 50178 between two contacts	V AC	300
Making capacity, IEC 60947		
AC-15 250 V AC, 3 A (600 Ops/h)	Operations	300000
DC-13 L/R $\leqq$ 0 ms 24 V DC, 1 A (500 Ops/h)	Switch operations	200000
Breaking capacity, IEC 60947		
AC-15 250 V AC, 3 A (600 Ops/h)	Switch operations	300000
DC-13 L/R $\leqq$ 150 ms 24 V DC, 1 A (500 ops/h)	Switch operations	200000
Filament bulb load		
1000 W at 230/240 V AC	Switch operations	25000
500 W at 115/120 V AC	Switch operations	25000
Fluorescent tube load, 10×58 W at 230/240 V AC		
Fluorescent tubes - with ballast - with conventional compensation - uncompensated	Switch operations	25000

Relay switching frequency		
Mechanical switch operations	Switch operations	10 mill. (107)
Mechanical switching frequency	Hz	10
Resistive lamp load	Hz	2
Inductive load	Hz	0.5

Transistor outputs		
Number of outputs		8
Rated voltage U_e	V DC	24
Permissible range	V DC	20.4 bis 28.8
Residual ripple	%	$\leqq 5$
Supply current		
On 0 state, typical/maximum	mA	18/32
On 1 state, typical/maximum	mA	24/44
Reverse polarity protection		Yes
▽**Caution!** Connecting the outputs to a power supply with a reverse polarity will result in a short-circuit.		
Potential isolation		Yes
Rated current I_e at state 1, maximum	A	0.5
Lamp load without R_V	W	5
Residual current on signal 0 per channel	mA	< 0.1
Maximum output voltage		
On "0" signal with external load, 10 MΩ	V	2.5
On „1", signal, $I_e = 0.5$ A		$U = U_e - 1$ V
Short-circuit protection (thermal) Group Q1 to Q4 /Group Q5 to Q8. Evaluation with Diagnostics input I16 (Q1 to Q4), I17 (Q5 to Q8)		Yes
▽**Caution!** Set the output group in the program to a "0" signal in order to prevent the output from overloading		
Short-circuit tripping current for $R_a \leqq 10$ mΩ (depending on number of active channels and their load)	A	$0.7 \leqq I_e \leqq 2$
Maximum total short-circuit current	A	16
Peak short-circuit current	A	32
Thermal cutout		Yes
Maximum switching frequency with constant resistive load $R_L = 100$ kΩ (depends on program and load)	Switch operations/h	40 000

Parallel connection of outputs with resistive load; inductive load with external suppression circuit (⟶ section "Connecting transistor outputs", page 25); combination within a group			Yes
Group 1: Q1 to Q4			
Group 2: Q5 - Q8			
Maximum number of outputs			4
total maximum current		A	2
▽ **Caution!** Outputs connected in parallel must be switched at the same time and for the same duration.			
Status display of the outputs			LC display

Inductive load **without external suppressor circuit**

General explanations:
$T_{0.95}$ = time in milliseconds until 95 % of the stationary current is reached.

$$T_3 \approx 3 \times T_3 = 3 \times \frac{L}{R}$$

Utilisation categories in groups Q1 to Q4, Q5 to Q8

T0.95 = 1 ms $R = 48\ \Omega$ $L = 16$ mH	Utilisation factor per group g =		0.25
	Relative duty factor	%	100
	Max. switching frequency $f = 0.5$ Hz Max. duty factor DF = 50 %	Switching operations/h	1 500
DC13 $T0,95 = 72$ ms $R = 48\ \Omega$ $L = 1.15$ H	Simultaneity factor g =		0.25
	Relative duty factor	%	100
	Max. switching frequency $f = 0.5$ Hz Max. duty factor DF = 50 %	Switching operations/h	1 500

Other inductive loads:

T0.95 = 15 ms $R = 48\ \Omega$ $L = 0.24$ H	Simultaneity factor g =		0.25
	Relative duty factor	%	100
	Max. switching frequency $f = 0.5$ Hz Max. duty factor DF = 50 %	Switch operations/h	1 500
Inductive loading with external suppressor circuit for each load (⟶ section "Connecting transistor outputs", page 25)			
	Simultaneity factor g =		1
	Relative duty factor	%	100
	Max. switching frequency Max. duty factor	Switch operations/h	Depending on the suppressor circuit

Analog output		
Quantity		1
Potential isolation		
to power supply		No
From the digital inputs		No
To the digital outputs		yes
from the easy-NET network		yes
Output type		DC voltage
Signal range	V DC	0 bis 10
Output current max.	mA	10
Load resistor	kΩ	1
Short-circuit and overload proof		yes
Resolution, analog	V	0.01
Resolution, digital	Bit	10
	Value	0 to 1023
Recovery time	us	100
Accuracy		
(—-25 … 55 °C), related to the range	%	2
(25 °C), related to the range	%	1
Conversion time		each CPU cycle

Character sets

Latin 1, Western European

Code	Meaning	Code	Meaning	Code	Meaning	Code	Meaning
0	Space	64	@	128	€	192	À
1	Ā	65	A	129	‰	193	Á
2	B̄	66	B	130	↕	194	Â
3	C̄	67	C	131	?	195	Ã
4	D̄	68	D	132	„	196	Ä
5	Ē	69	E	133		197	Å
6	F̄	70	F	134		198	Æ
7	Ḡ	71	G	135		199	Ç
8	H̄	72	H	136		200	È
9	Ī	73	I	137		201	É
10	J̄	74	J	138		202	Ê
11	K̄	75	K	139		203	Ë
12	L̄	76	L	140	–	204	Ì
13	M̄	77	M	141		205	Í
14	N̄	78	N	142		206	Î
15	Ō	79	O	143		207	Ï
16	P̄	80	P	144		208	Đ
17	Q̄	81	Q	145		209	Ñ
18	R̄	82	R	146		210	Ò
19	S̄	83	S	147		211	Ó
20	T̄	84	T	148		212	Ô
21	Ū	85	U	149	.	213	Õ
22	V̄	86	V	150	█	214	Ö
23	W̄	87	W	151		215	<
24	X̄	88	X	152		216	Ø
25	Ȳ	89	Y	153	Š	217	Ù
26	Z̄	90	Z	154	š	218	Ú
27	↓	91	[	155	œ	219	Û
28	↓	92	\	156	œ	220	Ü
29	→	93	]	157	ž	221	Ý
30	←	94	^	158	ž	222	
31	◤ (cursor)	95	_	159	ÿ	223	ß
32	Space	96	`	160	√	224	à
33	!	97	a	161	¡	225	á
34	"	98	b	162	¢	226	â
35	#	99	c	163	‖	227	ã
36	$	100	d	164		228	ä
37	%	101	e	165		229	å
38	&	102	f	166		230	æ
39	'	103	g	167		231	ç
40	(	104	h	168	█ (Cursor)	232	è
41	)	105	i	169		233	é
42	*	106	j	170		234	ê
43	+	107	k	171		235	ë
44	,	108	l	172	=	236	ì
45	–	109	m	173		237	í
46	.	110	n	174		238	î
47	/	111	o	175		239	ï
48	0	112	p	176	°	240	ð
49	1	113	q	177	±	241	ñ
50	2	114	r	178	²	242	ò
51	3	115	s	179	³	243	ó
52	4	116	t	180	´	244	ô
53	5	117	u	181	µ	245	õ
54	6	118	v	182	10	246	ö
55	7	119	w	183	11	247	≥
56	8	120	x	184	12	248	ø
57	9	121	y	185	13	249	ù
58	:	122	z	186	14	250	ú
59	;	123	{	187	15	251	û
60	<	124			188	16	252	ü
61	=	125	}	189	0	253	ý
62	>	126	~	190	0̄	254	þ
63	?	127	£	191	¿	255	ÿ

Character set Latin 2, "Central European" (for Polish, Hungarian and Czech)

Code	Meaning	Code	Meaning	Code	Meaning	Code	Meaning
0	Space	64	@	128	€	192	Ŕ
1	Ā	65	A	129	¢	193	Á
2	B̄	66	B	130	↕	194	Â
3	C̄	67	C	131	‚	195	Ă
4	D̄	68	D	132	„	196	Ä
5	Ē	69	E	133		197	Ĺ
6	F̄	70	F	134		198	Ć
7	Ḡ	71	G	135		199	Ç
8	H̄	72	H	136	□	200	Č
9	Ī	73	I	137	‰	201	É
10	J̄	74	J	138	Š	202	Ę
11	K̄	75	K	139	‹	203	Ë
12	L̄	76	L	140	ś	204	Ě
13	M̄	77	M	141	ẗ	205	Í
14	N̄	78	N	142	ž	206	Î
15	Ō	79	O	143	ż	207	Ď
16	P̄	80	P	144	Ţ	208	Đ
17	Q̄	81	Q	145	'	209	Ń
18	R̄	82	R	146	'	210	Ň
19	S̄	83	S	147	"	211	Ó
20	T̄	84	T	148	"	212	Ô
21	Ū	85	U	149	•	213	Ő
22	V̄	86	V	150	▮	214	Ö
23	W̄	87	W	151	—	215	≤
24	X̄	88	X	152	˛	216	Ř
25	Ȳ	89	Y	153	³	217	Ů
26	Z̄	90	Z	154	š	218	Ú
27	↑	91	[	155	›	219	Ű
28	↓	92	\	156	ś	220	Ü
29	→	93	]	157	?	221	Ý
30	←	94	^	158	ž	222	
31	◪ (cursor)	95	_	159	ź	223	ß
32	Space	96	`	160	√	224	ŕ
33	!	97	a	161		225	á
34	"	98	b	162		226	â
35	#	99	c	163	Ł	227	ă
36	$	100	d	164		228	ä
37	%	101	e	165	Ą	229	í
38	&	102	f	166	¦	230	ć
39	'	103	g	167		231	ç
40	(	104	h	168	▮ (cursor)	232	č
41	)	105	i	169	©	233	é
42	*	106	j	170	Ş	234	ę
43	+	107	k	171		235	ë
44	,	108	l	172		236	ě
45	−	109	m	173		237	í
46	.	110	n	174		238	î
47	/	111	o	175	Ż	239	ď
48	0	112	p	176	°	240	đ
49	1	113	q	177	±	241	ń
50	2	114	r	178		242	ň
51	3	115	s	179	ł	243	ó
52	4	116	t	180	´	244	ô
53	5	117	u	181	µ	245	ő
54	6	118	v	182	¶	246	ö
55	7	119	w	183	·	247	≥
56	8	120	x	184		248	ř
57	9	121	y	185	ą	249	ů
58	:	122	z	186	ş	250	ú
59	;	123	{	187		251	ű
60	<	124	|	188	Ľ	252	ü
61	=	125	}	189		253	ý
62	>	126	~	190	ľ	254	ţ
63	?	127	£	191	ż	255	

Character set "Cyrillic" (for Russian)

Code	Meaning	Code	Meaning	Code	Meaning	Code	Meaning
0	Space	64	@	128	Є	192	А
1	Ā	65	A	129	ѓ	193	Б
2	B̄	66	B	130	↕	194	В
3	C̄	67	C	131	ѓ	195	Г
4	D̄	68	D	132	„	196	Д
5	Ē	69	E	133		197	Е
6	F̄	70	F	134	2	198	Ж
7	Ḡ	71	G	135	з	199	З
8	H̄	72	H	136	□	200	И
9	Ī	73	I	137	‰	201	Й
10	J̄	74	J	138	Љ	202	К
11	K̄	75	K	139	‹	203	Л
12	L̄	76	L	140	Њ	204	М
13	M̄	77	M	141	Ќ	205	Н
14	N̄	78	N	142	ћ	206	О
15	Ō	79	O	143	џ	207	П
16	P̄	80	P	144	ђ	208	Р
17	Q̄	81	Q	145	'	209	С
18	R̄	82	R	146	'	210	Т
19	S̄	83	S	147	"	211	У
20	T̄	84	T	148	"	212	Ф
21	Ū	85	U	149	•	213	Х
22	V̄	86	V	150	▮	214	Ц
23	W̄	87	W	151	—	215	Ч
24	X̄	88	X	152	Ю	216	Ш
25	Ȳ	89	Y	153	h	217	Щ
26	Z̄	90	Z	154	ᵫ	218	Ъ
27		91	[	155	›	219	Ы
28	↓	92	\	156	њ	220	Ь
29	→	93	]	157	Ќ	221	Э
30	←	94	^	158	ћ	222	Ю
31	↙ (cursor)	95	_	159	џ	223	Я
32	Space	96	`	160	ѵ	224	а
33	!	97	a	161	Ў	225	б
34	"	98	b	162	ў	226	в
35	#	99	c	163	Ј	227	г
36	$	100	d	164	ђ	228	д
37	%	101	e	165	Ґ	229	е
38	&	102	f	166	¦	230	ж
39	'	103	g	167	Ё	231	з
40	(	104	h	168	▮ (cursor)	232	и
41	)	105	i	169	©	233	й
42	*	106	j	170	Є	234	к
43	+	107	k	171	≤	235	л
44	,	108	l	172		236	м
45	–	109	m	173		237	н
46	.	110	n	174	h	238	о
47	/	111	o	175	Ї	239	п
48	0	112	p	176	°	240	р
49	1	113	q	177	±	241	с
50	2	114	r	178	I	242	т
51	3	115	s	179	і	243	у
52	4	116	t	180	г	244	ф
53	5	117	u	181	µ	245	х
54	6	118	v	182	Њ	246	ц
55	7	119	w	183	ᵫ	247	ч
56	8	120	x	184	ё	248	ш
57	9	121	y	185	¢	249	щ
58	:	122	z	186	є	250	ъ
59	;	123	{	187	≥	251	ы
60	<	124	¦	188	ј	252	ь
61	=	125	}	189	S	253	э
62	>	126	~	190	s	254	ю
63	?	127	Ј	191	ї	255	я

Index