# Programmable Logic Controller
# easy Control
# EC4-200

## Hardware, Engineering and Function Description

12/06 AWB2724-1584GB

MOELLER

Think future. Switch to green.

1st published 2006, edition date 09/06
2nd edition 12/06
See revision protocol in the "About this manual" chapter



Author: Peter Roersch
Editor: Thomas Kracht
Translator: Terence Osborn

# Warning!
# Dangerous electrical voltage!

**Before commencing the installation**

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

# Contents

# About This Manual

## List of revisions

The following significant amendments have been introduced since previous issues:

| Edition date | Page | Description | New |
|---|---|---|---|
| 12/06 | 93 | EC4-200 network modules | ✓ |

## Additional documentation

At different points in this manual, references are made to more detailed descriptions in other manuals. These are described with their title and documentation number (e.g. AWB2700-1437GB).

All manuals are available in PDF format. If for some reason they are not supplied on the product CD, they are available for download as PDF files. Go to http://www.moeller.net/support and enter the document number in the Quick Search field.

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
- AN27K20GB: Coupling multiple stand-alone PLCs (CAN-Device) via CANopen
- AN27K27GB: Engineering of CAN stations
- AWB2786-1554: 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.

## Reading conventions

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

→ Draws your attention to interesting tips and supplementary information.

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

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

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

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).

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

- Connection of expansion devices/controllers via easy-LINK
- 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 easy-NET network via the easy-NET/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 easy Soft 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 XSoft (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.

**PLCs: Part number overview**

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

| Part no. | Features | | | |
|---|---|---|---|---|
| | Buttons/display | Transistor outputs | Relay outputs | Analog output |
| EC4P-221-MTXD1 | x | 8 | – | – |
| EC4P-221-MTXX1 | – | 8 | – | – |
| EC4P-221-MRXD1 | x | – | 6 | – |
| EC4P-221-MRXX1 | – | – | 6 | – |
| EC4P-221-MTAD1 | x | 8 | – | x |
| EC4P-221-MTAX1 | – | 8 | – | x |
| EC4P-221-MRAD | x | – | 6 | x |
| EC4P-221-MRAX1 | – | – | 6 | x |

# 2 Setup

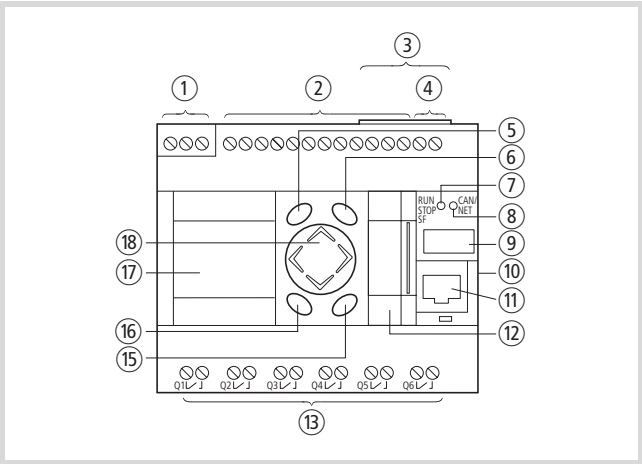



Figure: 1: Front of the EC4P-221-MRAD1,
Legend → figure 2

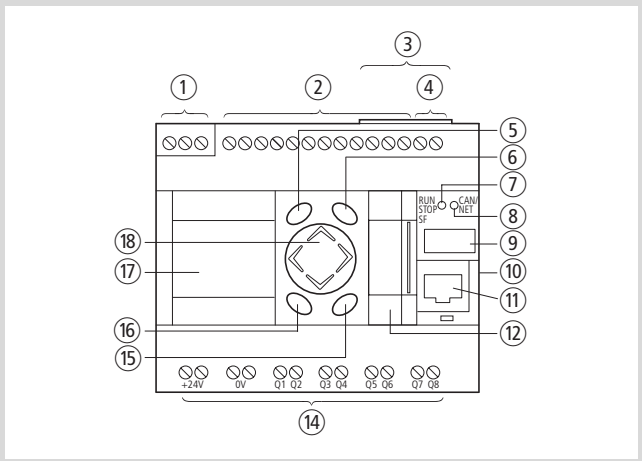



Figure: 2: Front of the EC4P-221-MTAD1

①24 V DC power supply
②Inputs
③Interface for connecting the CAN network
④Analog output, 0 – 10 V (not active)
⑤DEL key
⑥ALT key
⑦RUN/STOP/SF LED
⑧CAN/NET LED
⑨Field for device labelling
⑩easy-Link interface to expansion device
⑪Programming interface for connection to a PC
⑫Multi-function interface
⑬Relay outputs
⑭Transistor outputs
⑮OK key
⑯ESC button
⑰LCD display (EC4P-22x-M…D1)
⑱Cursor 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.

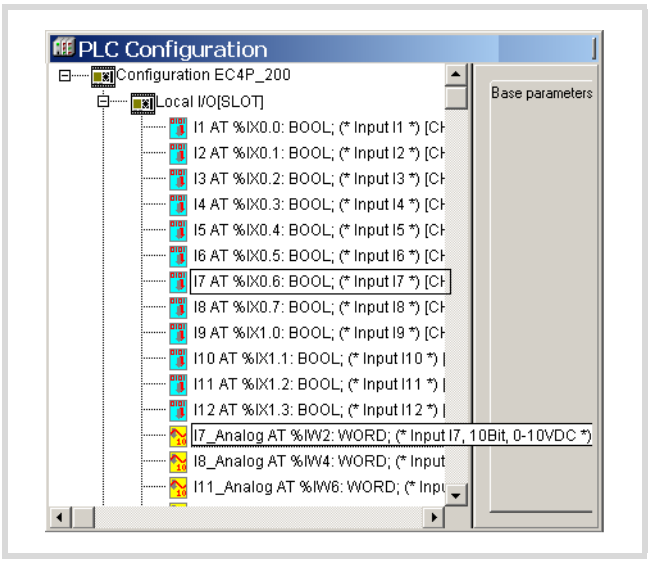



Figure: 3: Selection between digital and analog input, e.g. I7

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 |
|---|---|
| ≦ 8 | FALSE |
| > 8 | TRUE |

Technical data: → page 102

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 21 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 the general syntax rules. Only one button can be actuated at a time, otherwise uncontrolled states may occur when these buttons are scanned.

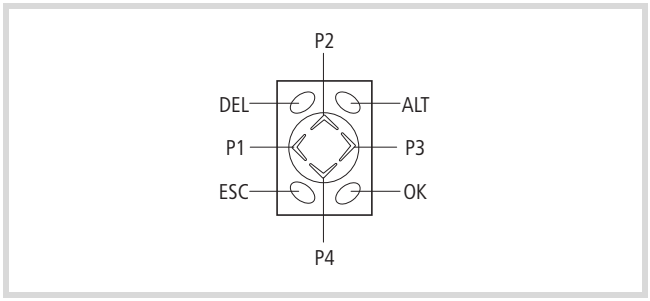



Figure: 4: Function buttons and rocker switch with cursor buttons P1, P2, P3, P4

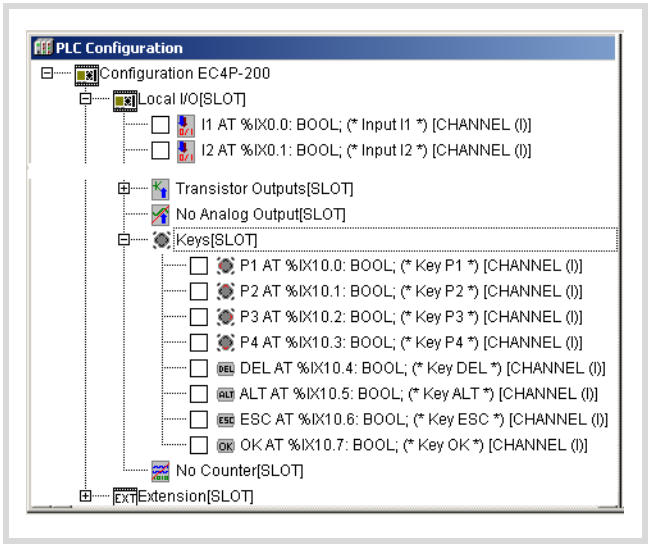



Figure: 5: Inputs of the rocker and function buttons

The GetDisplayInfo function block from the EC_Visu.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 | Connection to the expansion device via easy Link (not yet active in operating system version 1.x):<br>0: ok, 1: not ok |
| I15 | Outputs Q1, Q2, Q3, Q4:<br>0: No short-circuit, 1: Short-circuit |
| I16 | Outputs Q5, Q6, Q7, Q8:<br>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.

→ section "High-speed counters (Counter)", page 45

### Outputs

Table 2: Type and number of outputs

| | | |
|---|---|---|
| Transistor outputs EC4P-221/222-MT… | 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:<br>Write file to PLC<br>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 switching off 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 (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 signals

| LED | | Meaning/CPU status |
|---|---|---|
| RUN/STP/SF | CAN/NET | |
| Red | Red1) | System test being run (up to 6 seconds after start; after 6 seconds if no user program is present).<br>CPU in NOT READY! |
| Orange | Orange1) | System update in progress |
| Red | Off1) | System test finished without error |
| Red Flashing | Red Flashing1) | System test found a fault |
| Orange | Off | No user program present<br>CPU in NOT READY |
| Green Flashing | – | User program loaded<br>CPU in STOP |
| Green | – | User program loaded<br>CPU in RUN |
| Red | – | Cycle time exceeded<br>CPU in STOP |
| Orange Flashing | – | Continuous loop detected in program<br>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/easy-NET

| LED | | Meaning |
|---|---|---|
| RUN/STP/SF | CAN/NET | |
| Green | Off | Communication not active |
| Green | Red | Bus status STOP |
| Green | Orange | Bus status PREOPERATIONAL<br>Station can be initialised, no transfer of process data |
| Green | Green | Bus status OPERATIONAL<br>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 → easy Soft CoDeSys → Documentation → Automation Manuals>.

The clock can be read or set using the browser command "getrtc" and "setrtc". More information is provided in section "setrtc" on page 60.

In the event of a voltage loss the clock is backed up for at least 72 hours.

## Programming interface for the PC

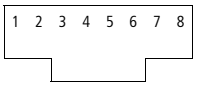
To connect the controller with a PC, use the cable EU4A-RJ45-CAB1. The cable should be plugged into the programming interface (RS232) of the controller. The interface is not electrically isolated.

The interface is initialised with the following default parameters when the PLC is started.

Table 5: Default parameters of the RS232 interface

| **Data length** | **8 bit** |
|---|---|
| Parity | None |
| Stop bits | 1 |
| Baud rate | 38400 Baud |

Table 6: Pin assignment of the RS232 programming interface




| | **Signal** |
|---|---|
| 1 | – |
| 2 | – |
| 3 | – |
| 4 | GND |
| 5 | TxD |
| 6 | – |
| 7 | GND |
| 8 | RxD |

### Transparent mode

The programming interface is addressed as COM1. It can be switched to Transparent mode using the functions of the library EC_SysLibCom.lib.

→ chapter "RS 232 interface in Transparent mode", page 71.

## Multi-function interface

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 (MFD)

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 101.

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 "RS 232 interface in Transparent mode", page 71.

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

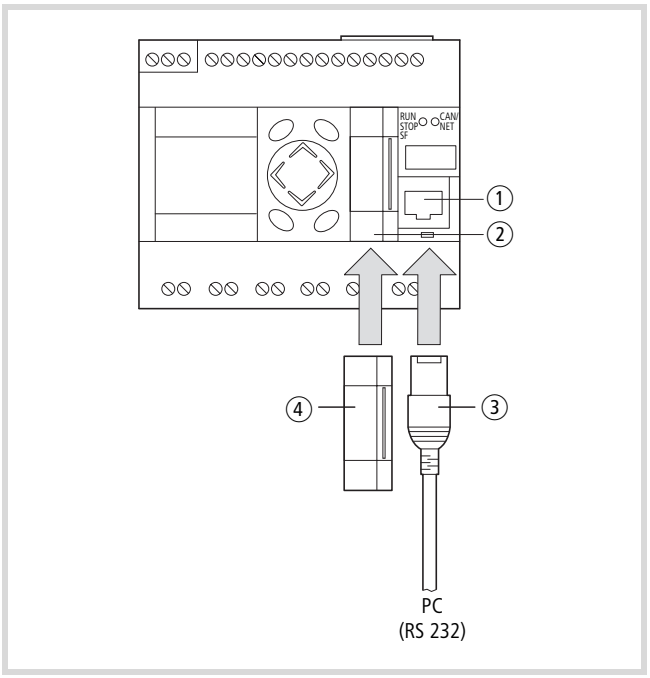



Figure: 6: Interfaces

①Programming interface for connection to a PC

②Multi-function interface

③EU4A-RJ45-CAB1 cable

④Adapter with memory card or connection of the following cables:
MFD-CP4-800-CAB5 cable for MFD-CP4
EASY800-PC-CAB cable for terminal/printer
EASY-USB-CAB for PC for programming via
USB interface

## CAN/easy-NET interfaces

The PLC is provided with a CAN/easy-NET interface with two slots that are internally connected via terminals.

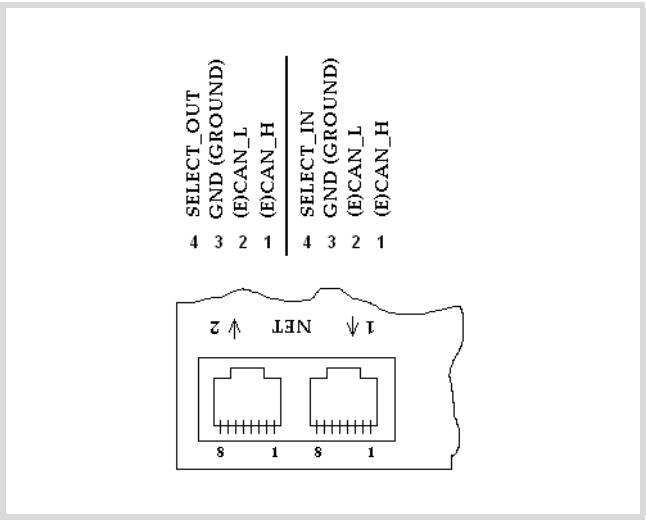



Figure: 7: CAN/easy-NET interfaces

### CANopen

The CAN interface is designed as a CANopen interface in compliance with the CIA specification DS 301V4.0. 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 "CAN/easy-NET network", page 99.

# 3 Expansion units

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

### Type overview of expansion devices

| TYPE | Power supply | Inputs | Outputs |
|---|---|---|---|
| EASY618-AC-RE | 100 … 230 V AC | 12 AC | 6 relay |
| EASY618-DC-RE | 24 V DC | 12 DC | 6 relay |
| 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

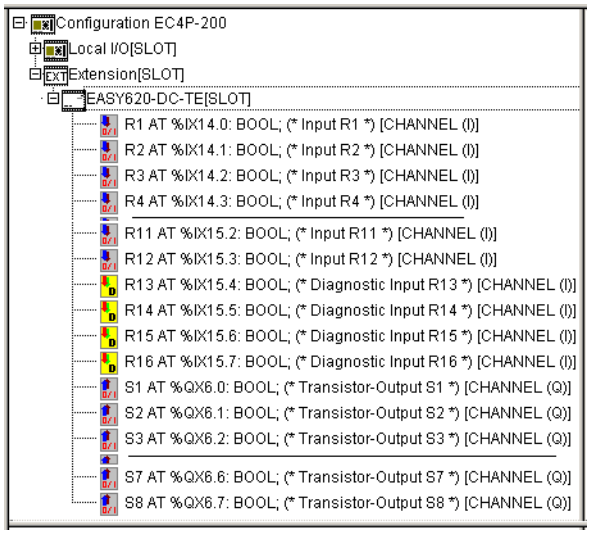



Figure: 8: I/Os of the EASY620-DC-TE

### Inputs

Table 7: Number of inputs and symbolic operands

| TYPE | 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 8: Functions of the diagnostics inputs

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

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

### Outputs

Table 9: Number of outputs and symbolic operands

| TYPE | 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.

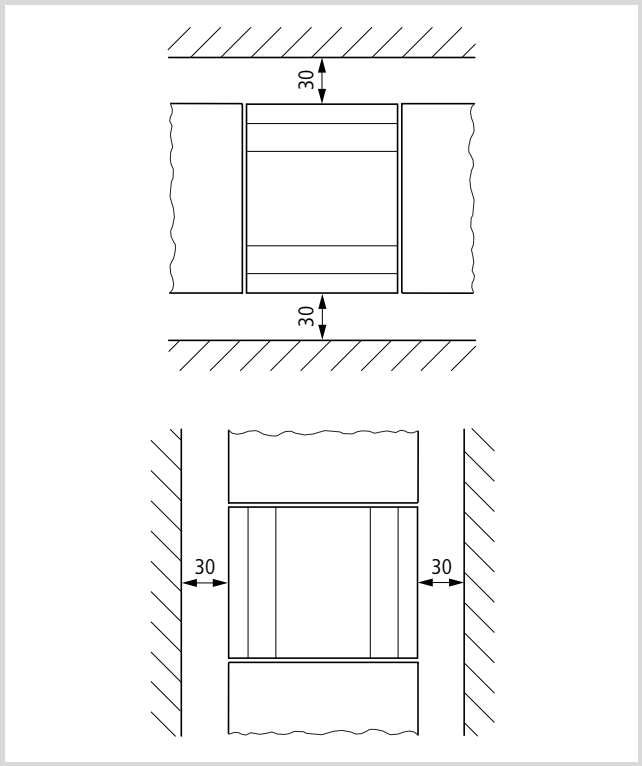



Figure: 9: Observing the clearances for wiring

**Mounting on top-hat rail**

- ▶ Place the device diagonally on the upper lip of the top-hat rail. Slightly push the device down and against the top-hat rail until it also snaps onto the bottom lip of the rail. The spring mechanism should ensure that the device snaps into position automatically.
- ▶ Check that the device is seated firmly.

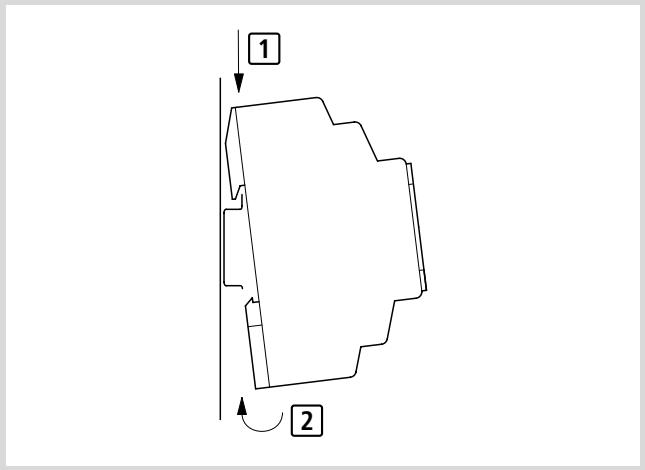



Figure: 10:

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.

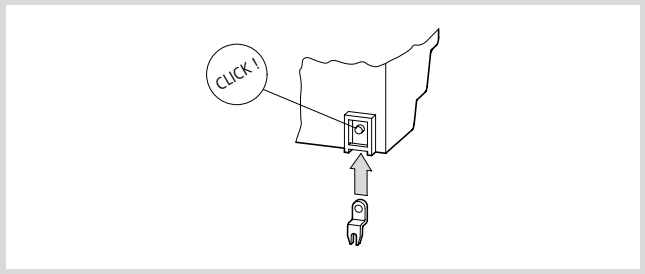



Figure: 11: Inserting a fixing bracket

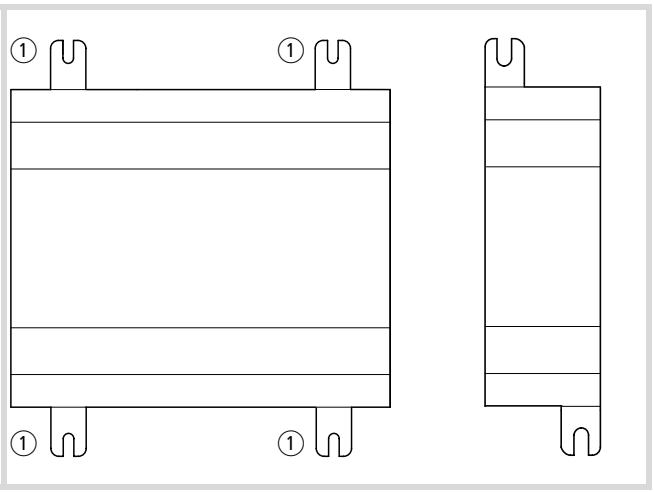



Figure: 12: Geräte anschrauben

①Fixing brackets

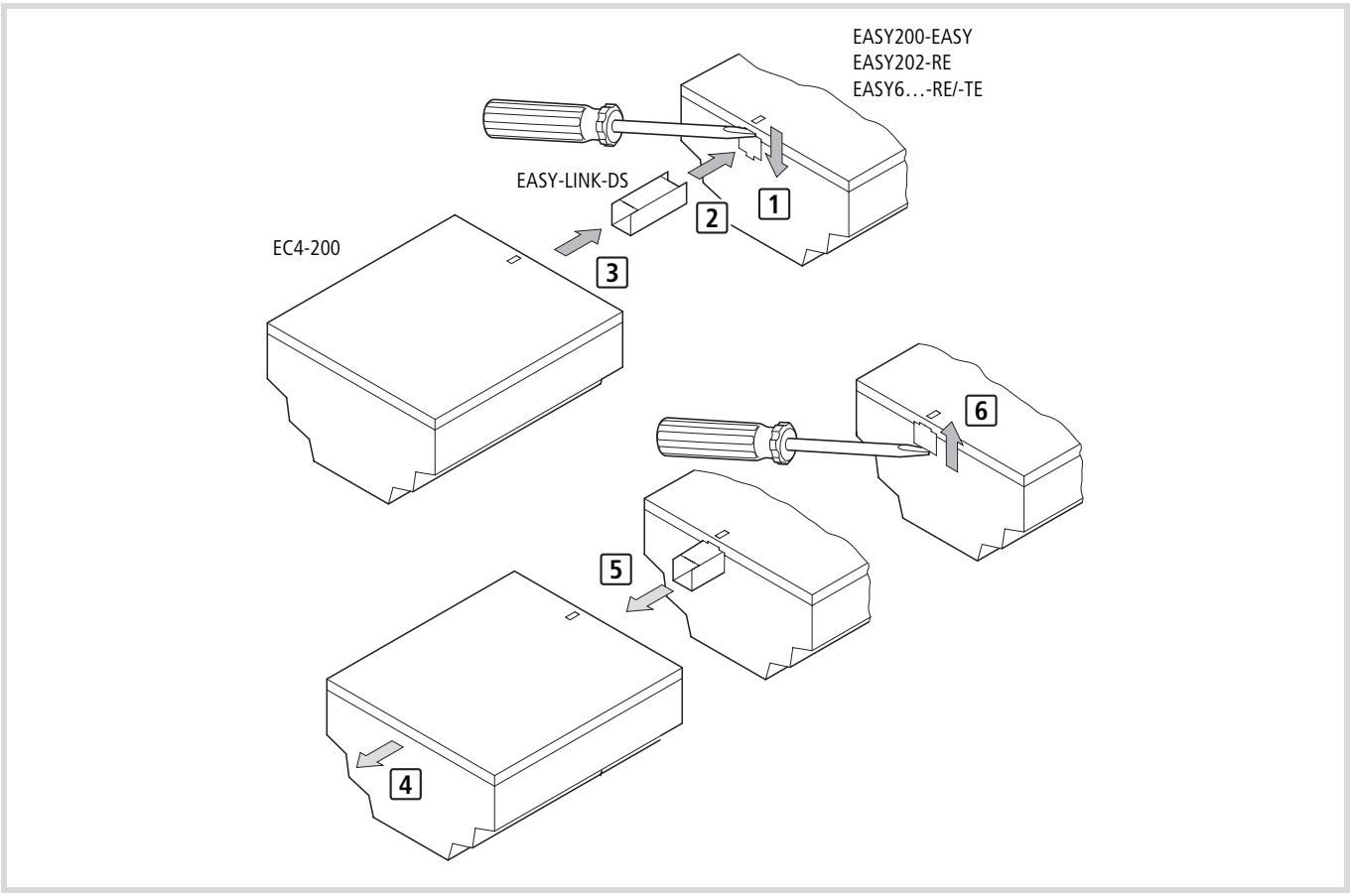



Figure: 13: Connecting the expansion unit/network module to the EC4-200

# 5 Installation

## Connecting the power supply

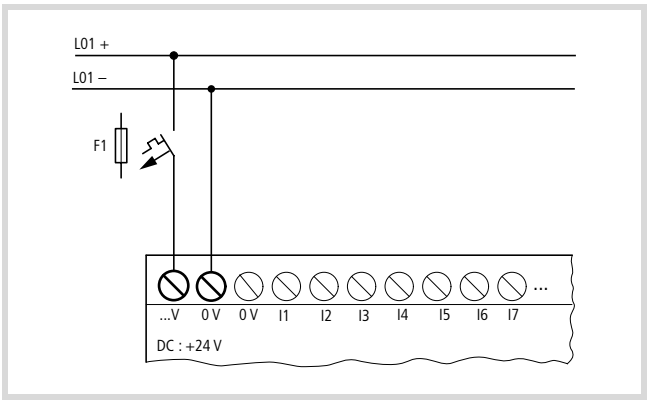



Figure: 14: Connecting the power supply
The two 0 V terminals are connected internally!

→ Die EC4-200 is protected against polarity reversal.

→ The necessary connection data is provided chapter "Technical data", page 102.

**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 I1 to I12 to connect pushbutton actuators, switches or 3 or 4-wire proximity switches. Do not use any 2-wire proximity switches due to the high residual current.

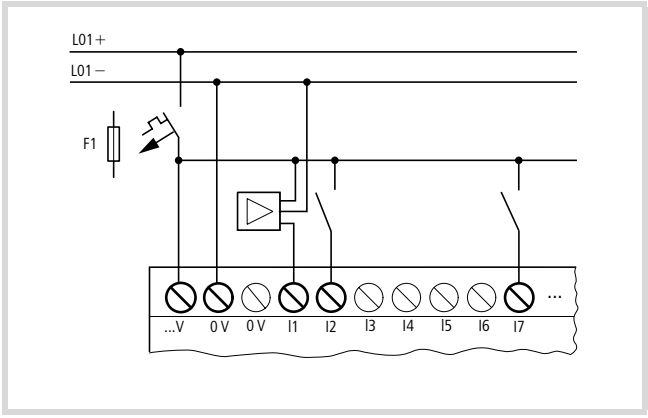



Figure: 15: Connecting digital inputs

## 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 … 1023.

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

- ▶ Use shielded twisted pair cables to prevent interference with the analog 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 ground the cable at one end.
- ▶ Do not lay signal lines parallel to power cables.
- ▶ 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 contain examples of applications for analog value processing.

→ 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.

### Connecting setpoint potentiometers

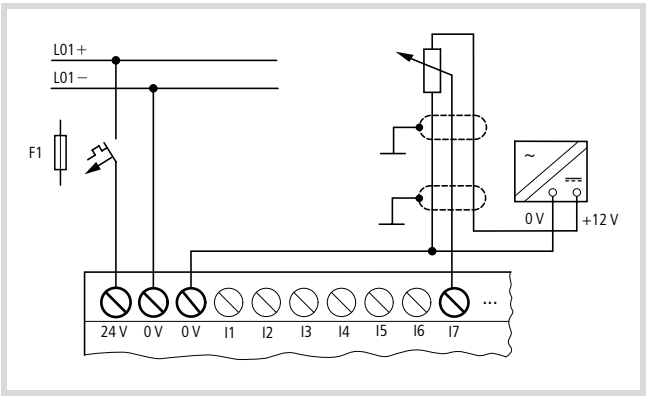



Figure: 16: Setpoint potentiometer

Use a potentiometer with the resistance ≦ 1 kΩ, e.g. 1 kΩ, 0.25 W.

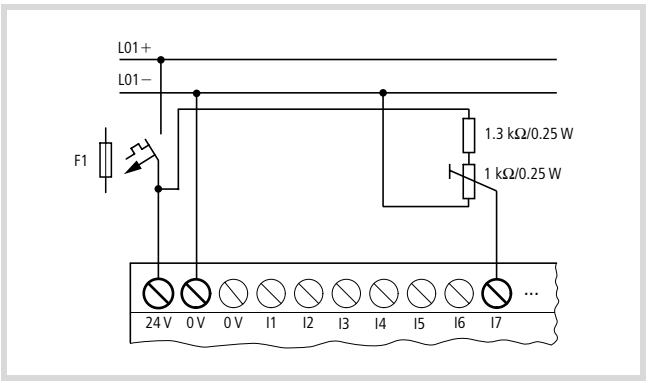



Figure: 17: Setpoint potentiometer with upstream resistor

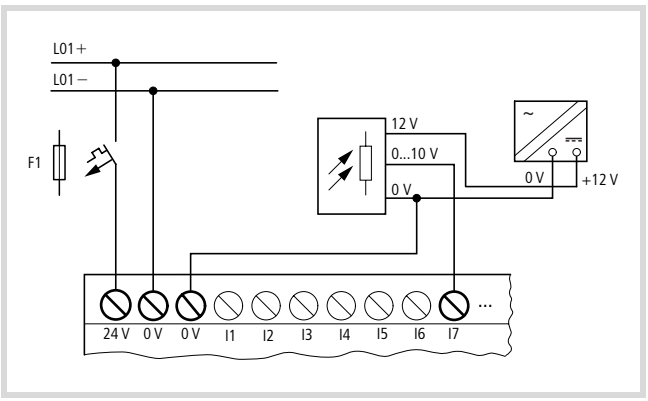



Figure: 18: Brightness sensor

### Temperature sensor connection

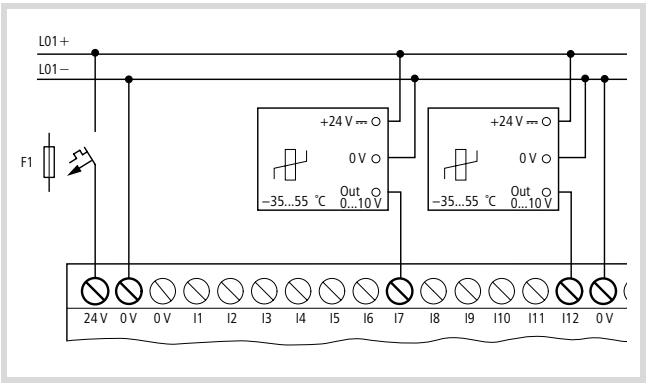



Figure: 19: Temperature sensor

### Connecting the 20 mA sensor

A 4 to 20 mA (0 to 20 mA) sensor can be connected easily with an external 500 O resistor.

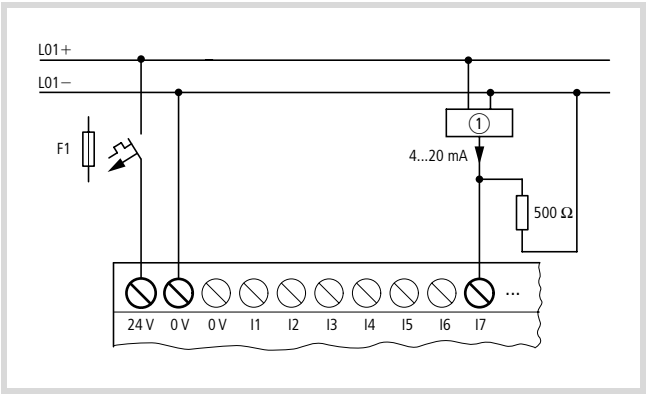



Figure: 20: 20 mA sensor

① Analog sensor

The following values apply:

- 4 mA = 1.9 V
- 4 mA = 1.9 V
- 20 mA = 9.5 V

(according to U = R x I = 478 O x 10 mA ~ 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:

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

### Connecting pulse transmitter

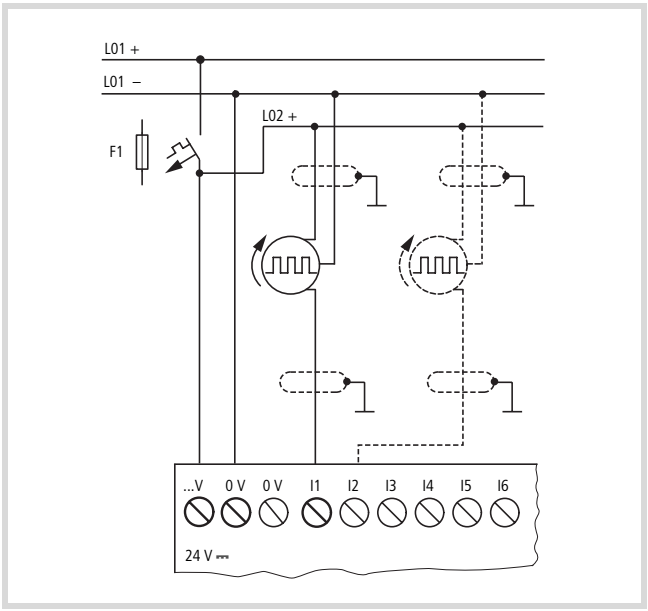



Figure: 21: Connecting pulse transmitter

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 (16-bit) be connected to I2.

### Connecting the incremental value encoder

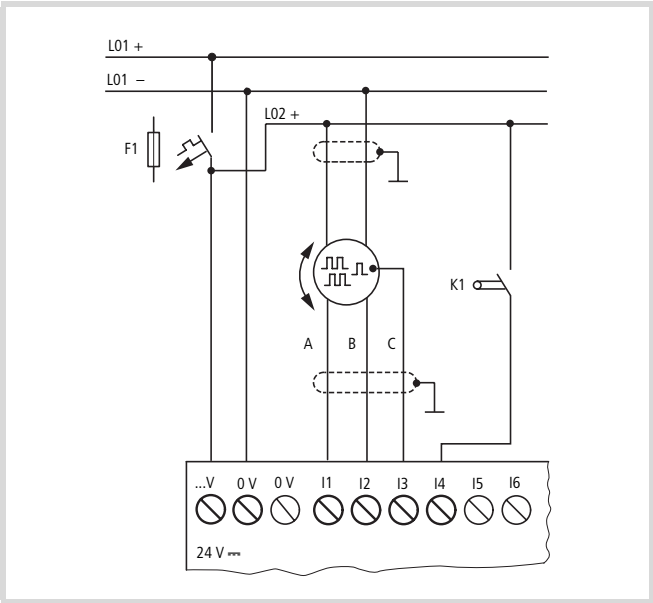



Figure: 22: Connecting the incremental encoder

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

## Connecting the outputs

The relay or transistor outputs are used to switch loads such as fluorescent tubes, filament bulbs, contactors, relays or motors. Prior to installation observe the technical limit values and data for the outputs (→ page 106, 107).

### Connecting the relay outputs

#### EC4P-221/222-MR…, EASY6..-DC-RE

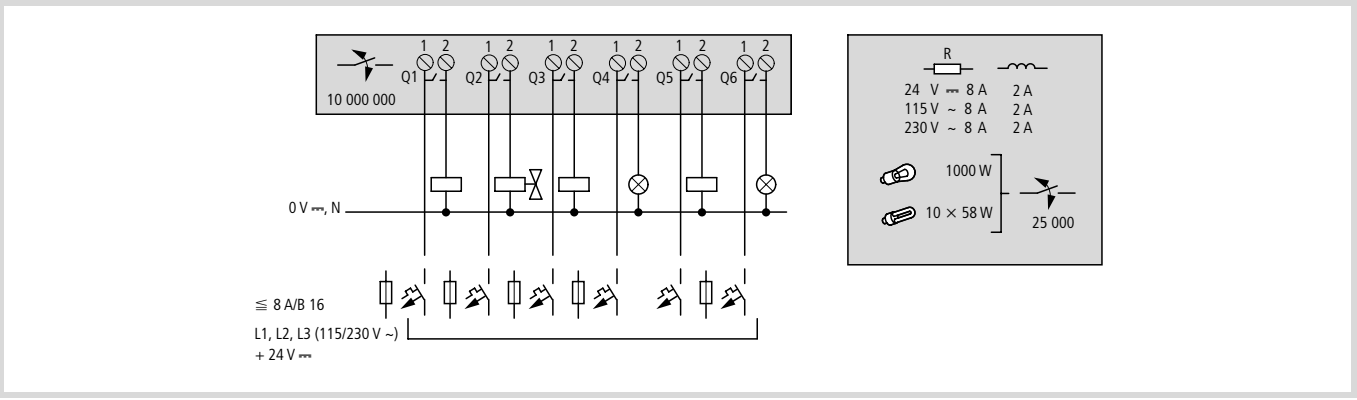



Figure: 23: Relay outputs EC4P-221/222-MR…

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

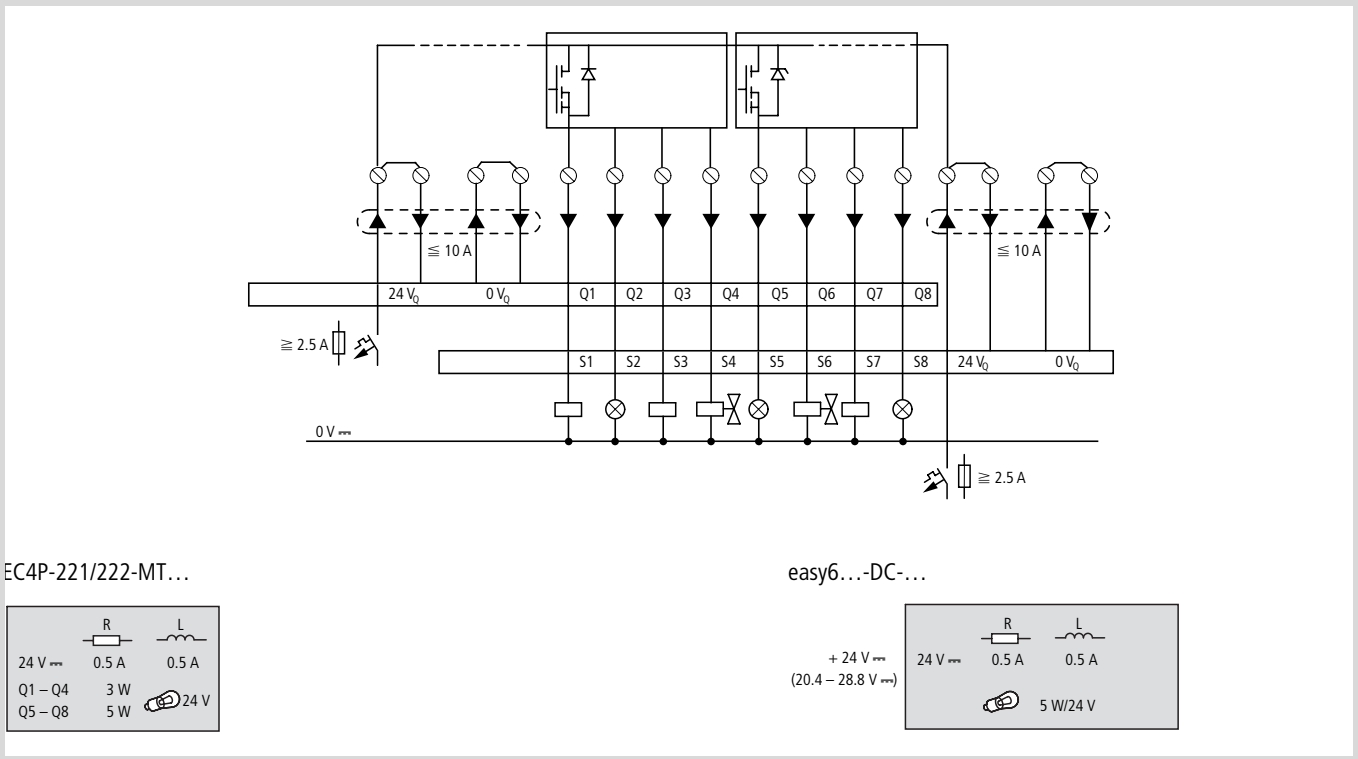



Figure: 24: Transistor outpus EC4P-221/222-MT…, EASY6..-DC-TE

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. Connecting the suppressor circuit as close to the inductance as possible is recommended.

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

If inductive loads are not suppressed, the following applies:
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 (a following diagrams).

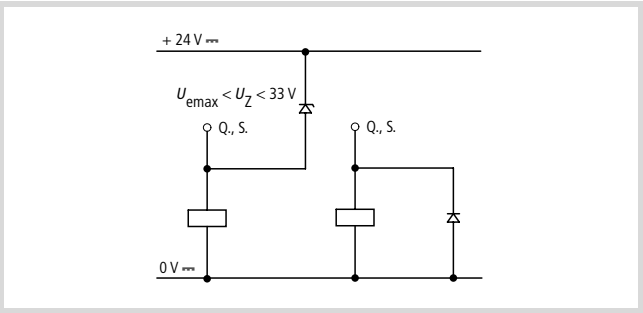



Figure: 25: Inductive load with suppressor circuit

### Behaviour in the event of a 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 analog outputs

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 valves

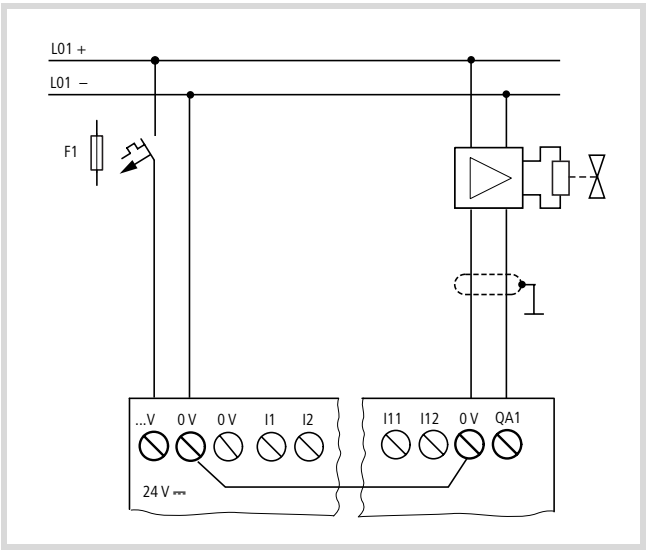



Figure: 26: Connecting servo valves

### Setpoint entry for a drive

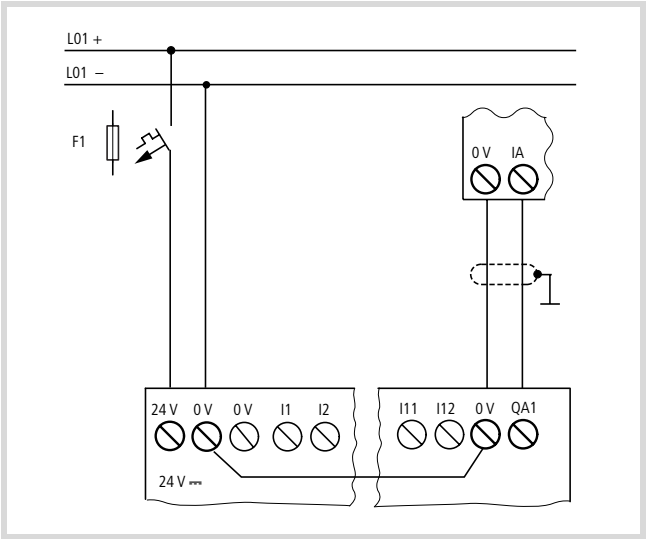



Figure: 27: Setpoint entry for a drive

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

## Memory card, CAN/easy-NET, PC connection

To fit a memory card or establish a CAN/easy-NET or PC connection the protective cap must be removed first of all.

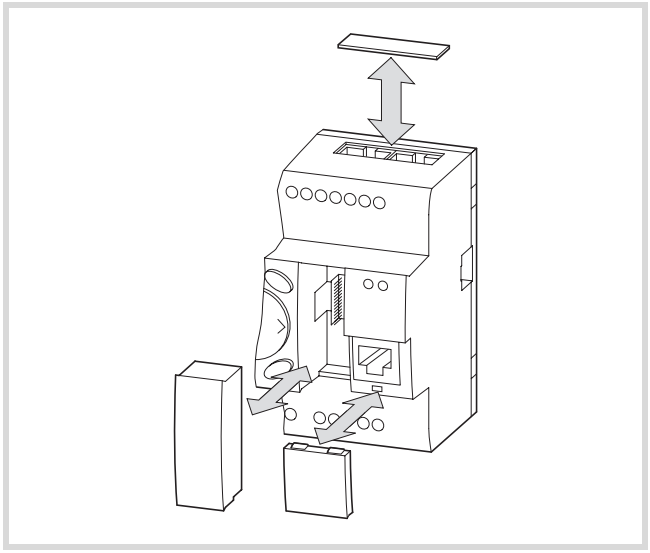


Figure: 28: Removing the protective cap/adapter: top: for CAN/easy-NET connection bottom left: adapter for memory card bottom right: PC connection

### Fitting or removing the memory card

The memory card is located in adapter c.

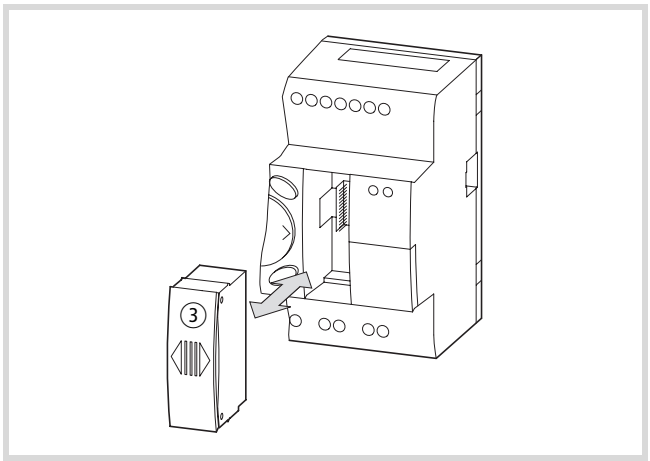



Figure: 29: Adapter with memory card

- ▶ To fit the memory card, press it until it snaps into position.
- ▶ To remove the memory card, press it until it is released.

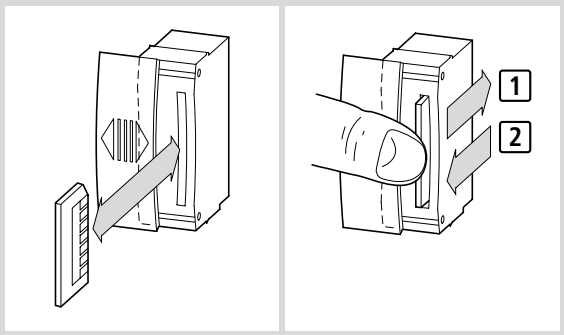



Figure: 30: Fitting/removing the memory card

### CAN/easy-NET, PC connection

- ▶ Fit the plug for the CAN/easy-NET 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 b.

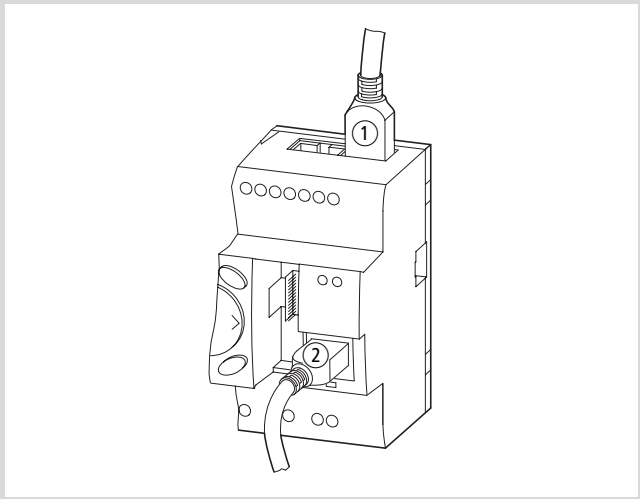



Figure: 31: Plug for the CAN/easy-NET connection ① and the PC connection ②

→ For further information see → section "CAN/easy-NET network", page 99.

▽ **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.

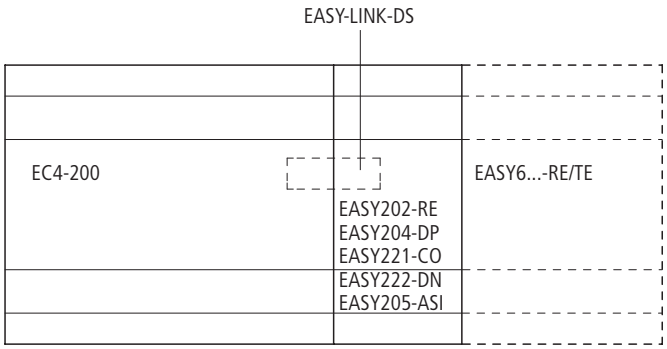



Figure: 32: Connecting expansion devices with EC4-200

The following electrical separation is implemented between the power supply/CPU module of the MFD device and the 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 adhere to 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 Ue = 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–.

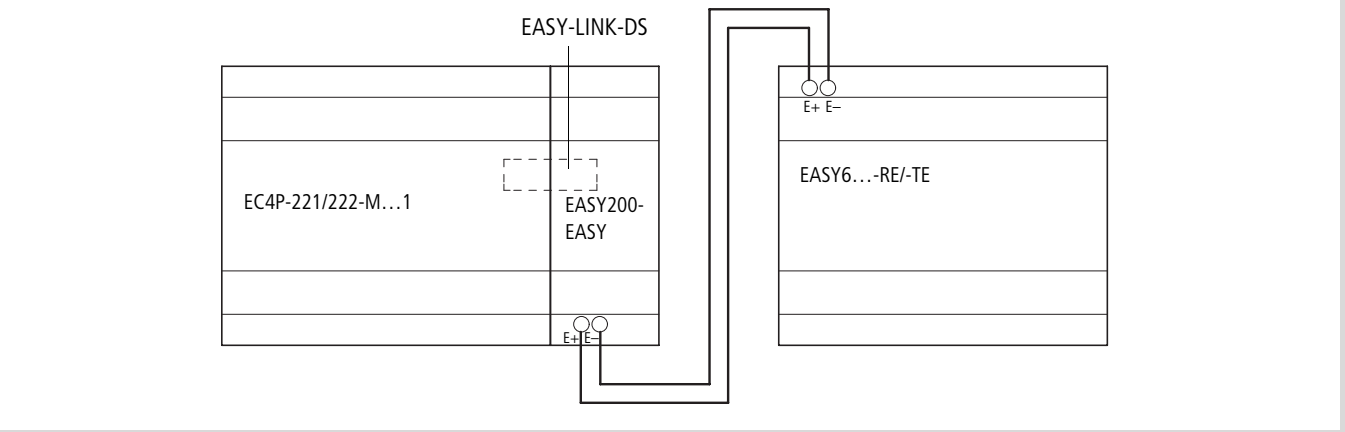
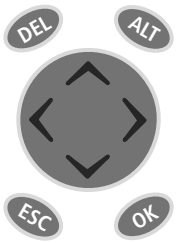



Figure: 33: Connecting remote expansion units to the EC4-200

# 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 buttonsúÌíÕÍ Ú: 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

v < Change value

> Change place

P buttons function:

< Input P1 ^ Input P2

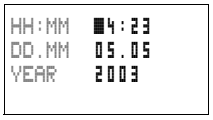
> Input P3 v Input P4

## Selecting or toggling between menu items


Cursor Í Ú

OK — Select or toggle

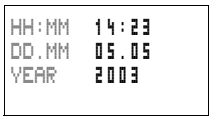
## Cursor display




The cursor flashes.

Full cursor Ê/:

- Move cursor with ú í ÍÕÚ

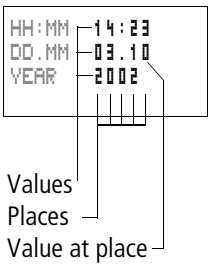



Value M/ M

- Change position with ú í
- Change values with Í Ú

Flashing values/menus are shown in grey in this manual.

## Setting values




Select value Í Ú

Select digit < >

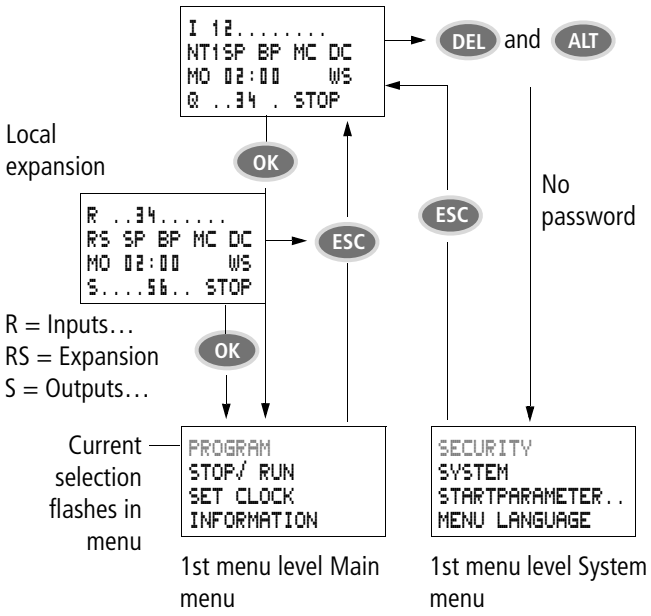
Change value at digit ^ v

OK — Store entries

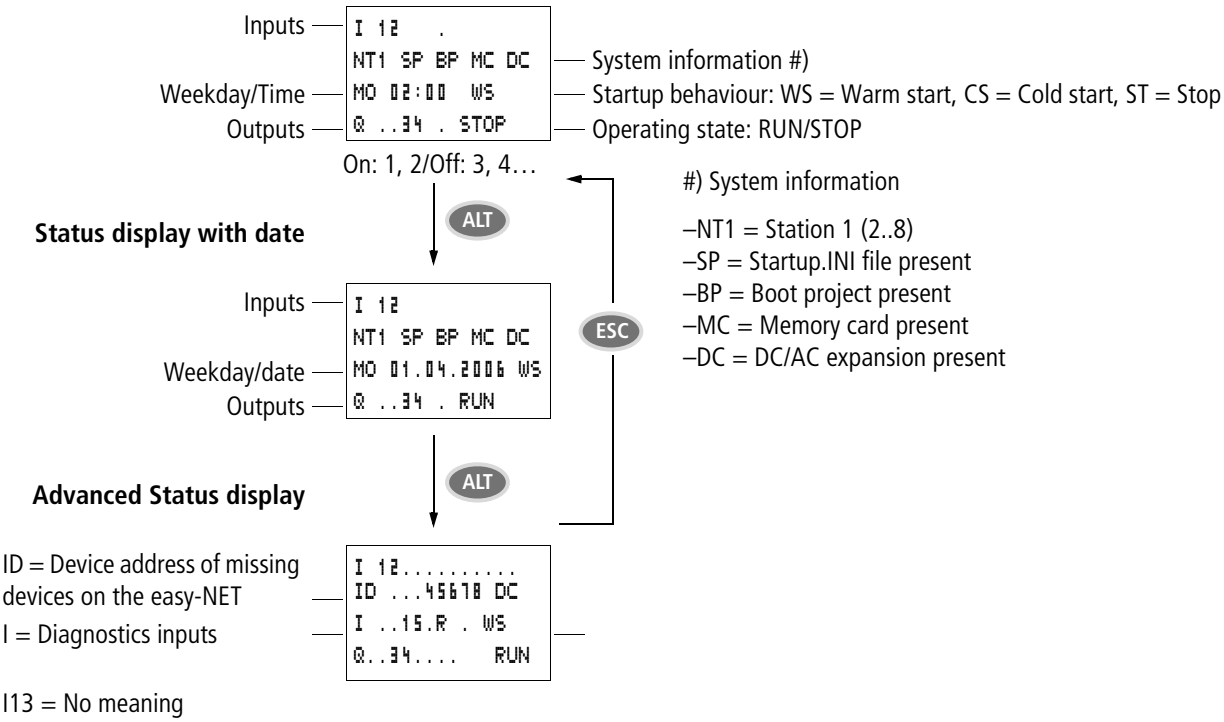
ESC — Retain previous value

## Selecting the System menu

### Status display




### Status display with time




#) System information

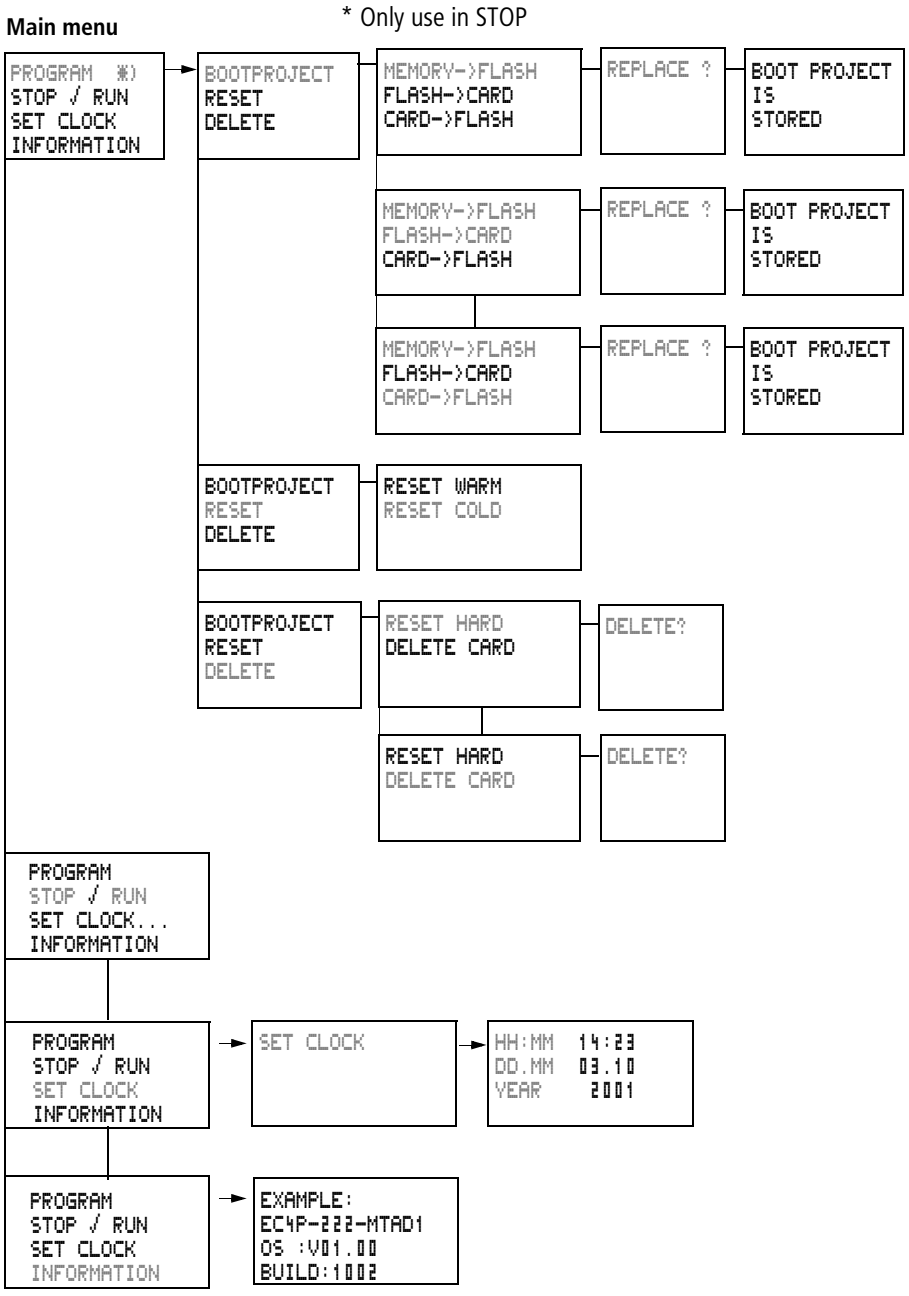
–NT1 = Station 1 (2..8)
–SP = Startup.INI file present
–BP = Boot project present
–MC = Memory card present
–DC = DC/AC expansion present

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 = toggels if short-circuit on output S1, S2, S3 or S4
R16 = toggels if short-circuit on output S5, S6, S7 or S8

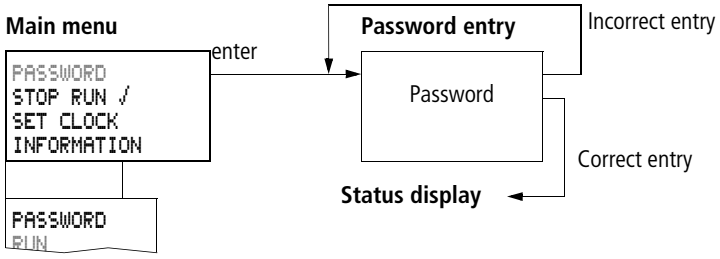
## Menu structure

### Main menu without password protection

▶ You access the main menu by pressing OK.

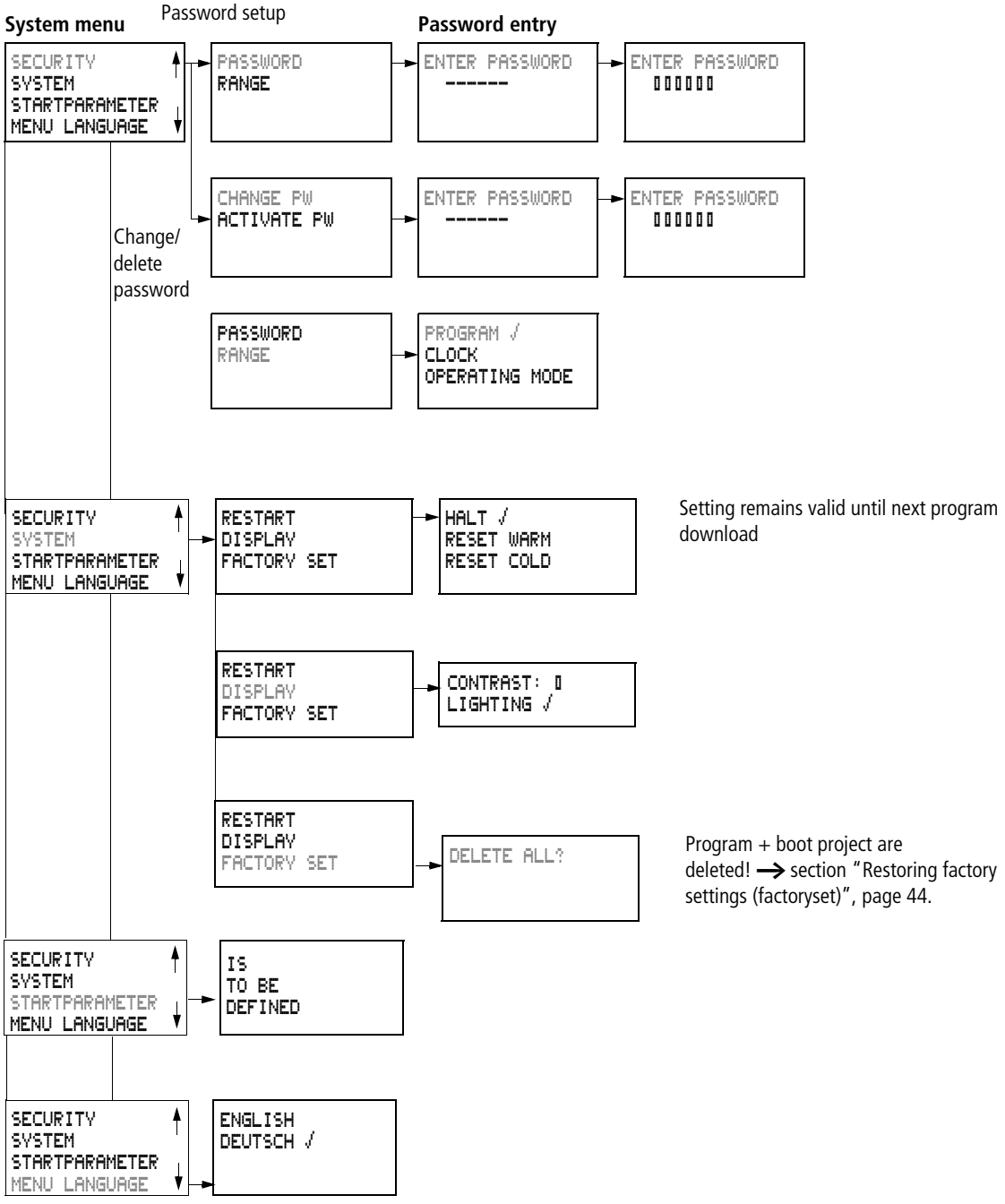


### Main menu with password protection




### System menu

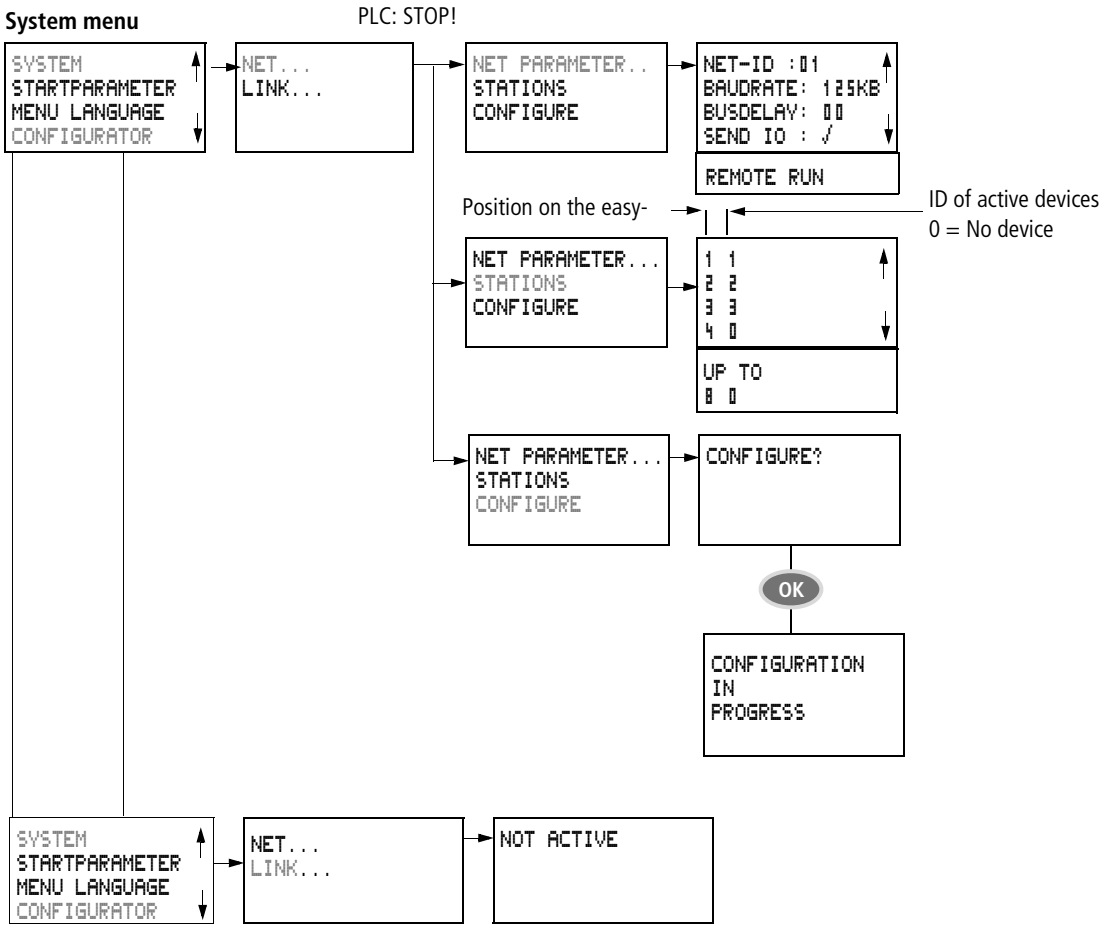
▶ The System menu is accessed by simultaneously pressing DEL and ALT.




Setting remains valid until next program download

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

## System menu


System menu
PLC: STOP!
SYSTEM
STARTPARAMETER
MENU LANGUAGE
CONFIGURATOR
NET...
LINK...
NET PARAMETER..
STATIONS
CONFIGURE
NET-ID :01
BAUDRATE: 125KB
BUSDELAY: 00
SEND IO : √
REMOTE RUN
Position on the easy-
ID of active devices
0 = No device
NET PARAMETER...
STATIONS
CONFIGURE
1 1
2 2
3 3
4 0
UP TO
8 0
NET PARAMETER...
STATIONS
CONFIGURE
CONFIGURE?
OK
CONFIGURATION
IN
PROGRESS
SYSTEM
STARTPARAMETER
MENU LANGUAGE
CONFIGURATOR
NET...
LINK...
NOT ACTIVE

# 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 l 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.


▶ Press OK, six zeros will appear
▶ Set the password using the cursor buttons:
– ú í select position in the password,
– ÍÚ set a value between 0 to 9.

▶ Save the new password by pressing OK.


Use OK to exit the password display and proceed with ESC and Ú 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).


→ 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 the password was entered.
- 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.

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.

▽ **Caution!**
If the password is unknown or lost, and the password delete function is not activated: The unit can only be reset to the factory setting by the manufacturer. The program and all data will be lost.

▶ 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.

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

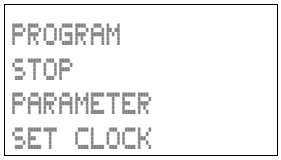
The password entry field is shown.


▶ 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.

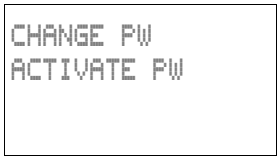



### Changing or deleting the password range

▶ 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.





▶ 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**.


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.

### Password incorrect or no longer known

Have you entered an incorrect password?


▶ Re-enter the password.

This can be repeated as many times as required!

Pressing ESC returns you to the starting menu

→ If you have lost the password, you can only change or delete it via the PC.

**Deleting the password (via PC)**
▶ Log on to the PLC.

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

▶ Call the browser command "factoryset".

## Changing the menu language

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

| Language | Display |
|---|---|
| English | ENGLISH |
| German | 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.


▶ Use Í or Ú 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.

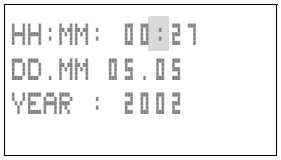
## Setting date and time

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.


▶ Select SET CLOCK.




▶ Set the values for time, day, month and year.
▶ Press the **OK** button to access the Entry mode.
  – 〈 〉 Move between the places.
  – ÍÚ Change the value of a parameter
  – OK Save day and time
  – ESC Retain previous setting.

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.


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


The menus for setting the contrast and backlight are displayed.


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

Use the Í and Ú cursor buttons to set the contrast to a value between –2 and +2.


▶ Select your setting.

▶ Complete your setting by pressing OK.


The contrast setting is valid until it is changed again.

▶ Use the cursor buttons Í and Ú to move to the LIGHTING menu.
▶ Press the OK button.


▶ The backlight is deactivated.


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


→ 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.

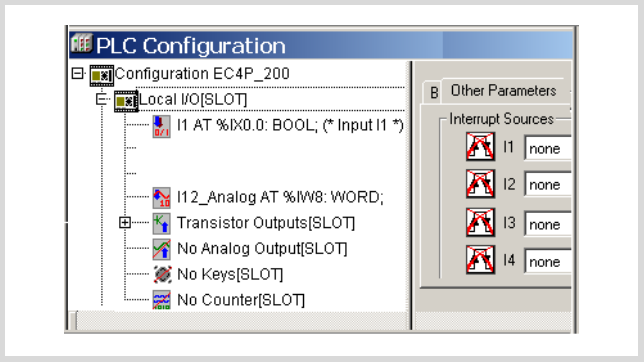



Figure: 34: Selectable inputs/outputs

## Changing the folder function

### Transistor Outputs n 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.

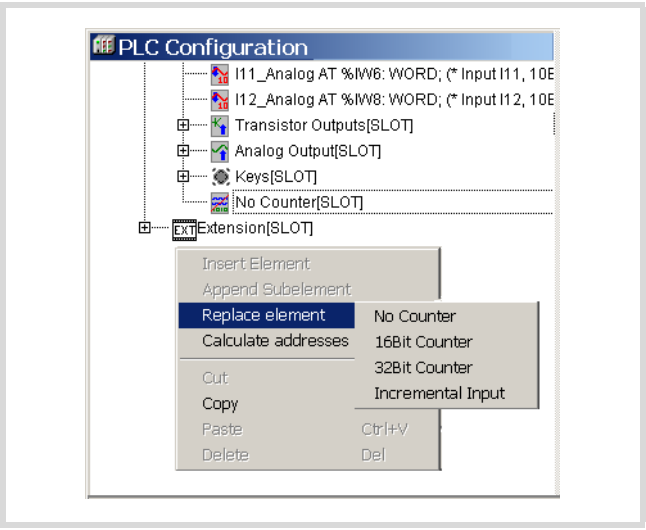



Figure: 35: Selecting counters

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.

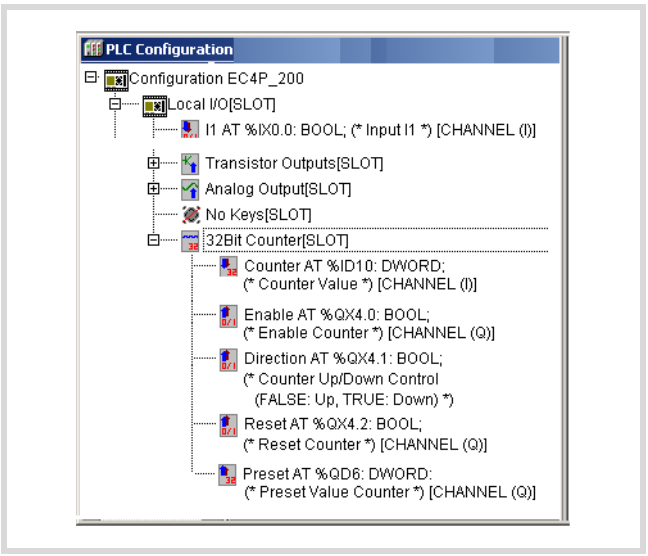



Figure: 36: Configuring a counter (32-bit)

## 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.

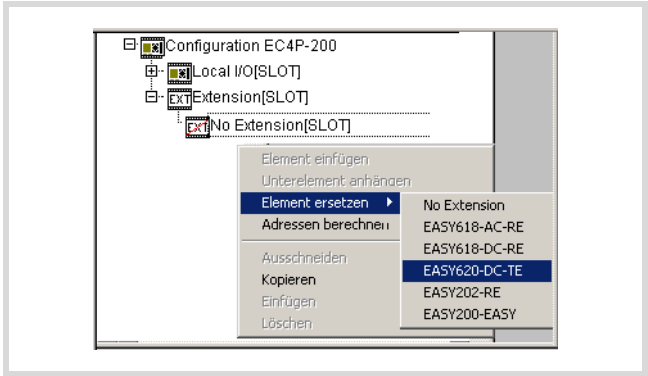



Figure: 37: Selecting the expansion device

- ▶ 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, if no libraries were included |
| 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 a 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.

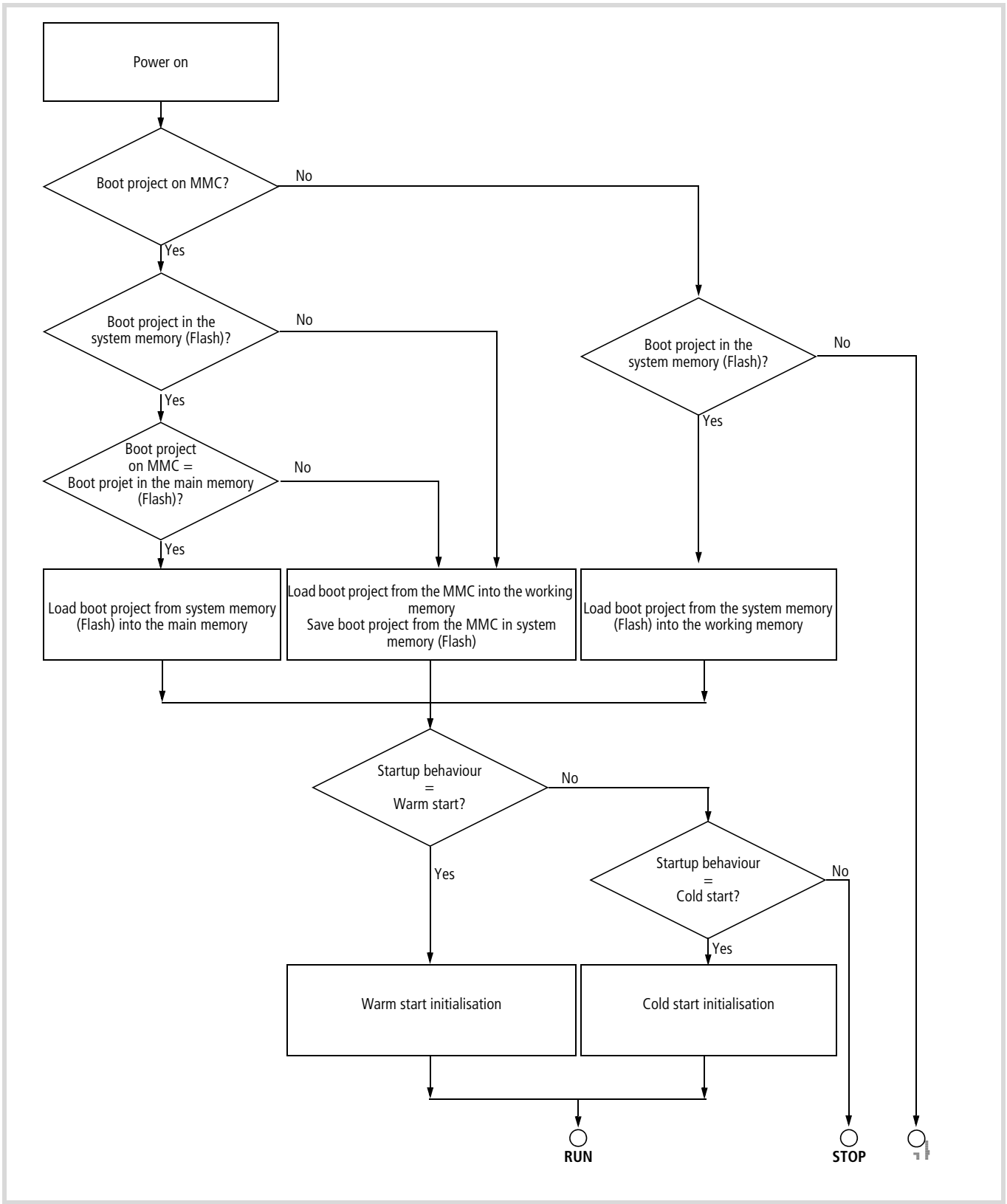



Figure: 38: Startup behaviour with boot project

## 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.

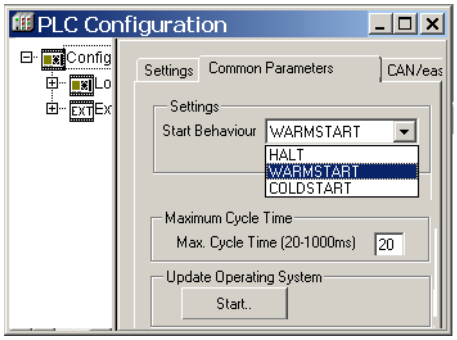



Figure: 39: Definition of start behaviour

## Program START/STOP

### Program start (STOP l RUN)

You can start the program in one of 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 power off or power interruption

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 causes 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 acceptable in your application, you can, for example, use an uninterruptible power supply (UPS) with back-up.

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 10.

The PLC restarts as defined by the settings in the PLC Configuration window, → figure 39.

Table 10: Behaviour of the variables at PLC start

| Start-up condition | Variable type | |
|---|---|---|
| | Non-retentive | Retentive |
| COLDSTART | Activation of the initial values | |
| WARMSTART | Activation of the initial values | Values remain in memory |
| Load and start program in Online mode | Activation of the initial values | |
| Start/Stop/Start… | Values remain in memory | |

### Program stop (RUN l 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 (). The SysLibPlcCtrl.lib library contains the function SysStartPlcProgram required for the start, and the function SysStopPlcProgram required for the stop.→ page 100

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 status image of the inputs is read before the start of each program cycle, and the status image of the outputs is written 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, → section "System events", 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. The shortest time is 20 ms (default value), the longest time is 1000 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 (factoryset)

The browser command "factoryset" or choosing SYSTEM I FACTORY SET can start a Hard reset command (). 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.→ section "Hard Reset"

### Behaviour of variables after Reset

| Reset | Variable type | |
|---|---|---|
| | Non-retentive | Retain |
| Warm reset | Activation of the 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 PLC supports the following test and commissioning functions:

- Breakpoint/single-step mode
- Single cycle mode
- Forcing
- Online modification
- 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 executed only in the single-step mode. Cycle time monitoring is deactivated.

⚠ **Caution!**
Any outputs already set when the program reaches the breakpoint remain set!

### Single cycle mode

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

⚠ **Caution!**
Any outputs already set when the program reaches the breakpoint 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 field bus can not 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. The other inputs/outputs of a counter such as Reset, are shown in the PLC configuration after you have selected the counter function, e.g. 32BitCounter (→ 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: 40: Programming inputs/oututs of the high-speed 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, e.g. 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:

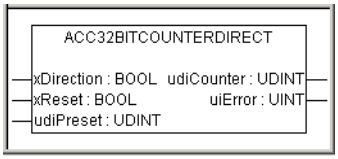



Figure: 41: 32-bit counter function block

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 l 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 l 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

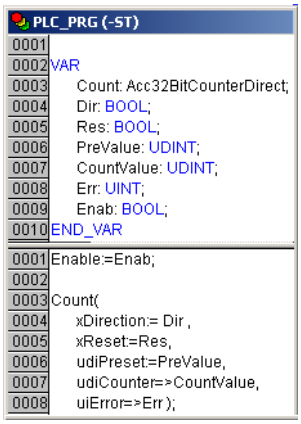



Figure: 42: Program with FB for 32-bit counter

#### 16 Bit Counter

Two of these counters are available. 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".

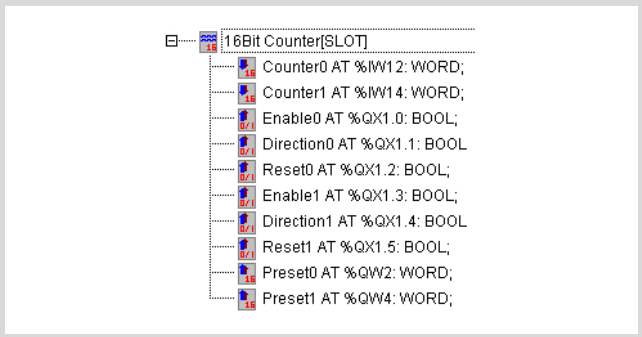



Figure: 43: Inputs/outputs of the 16-bit counter 0 and 1

→ 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.

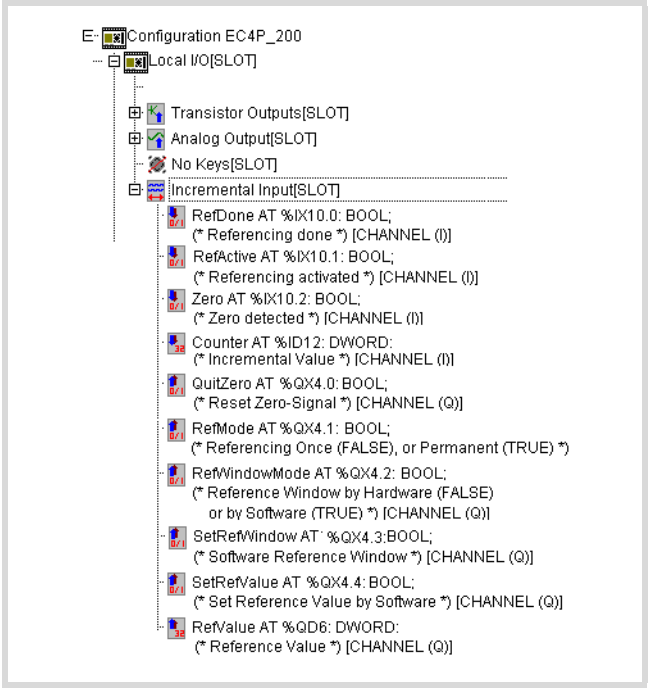



Figure: 44: Input/output signals of the incremental value counter

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

| Signal | I/Q | Explanation |
|---|---|---|
| RefDone | I | Referencing completed<br>(feedback signal of SetRefWindow) |
| RefActive | I | Referencing activated<br>(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<br>0 = once<br>1 = permanent |
| RefWindowMode | Q | Activation of reference window by<br>0 = external input I4<br>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)

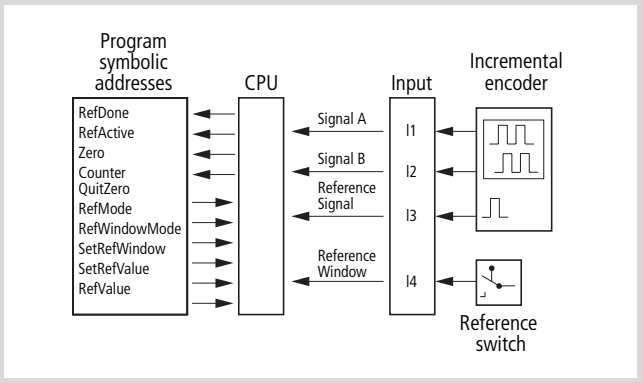



Figure: 45: Input/output signals of the incremental value counter

### 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 RefValue present at the 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 once and still be evaluated reliably.

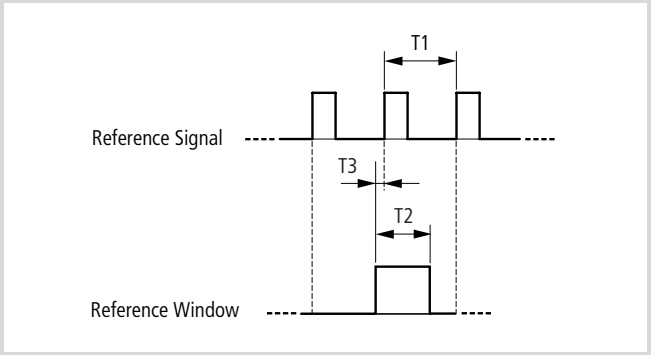



Figure: 46: Relationship between reference signal and reference window

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 reference pulse is reliably 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<br>(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 interrupt 2 | 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. The execution time is monitored. The value set as the maximum permissible cycle time is used as a basis.

### START, COLD START, WARM START, STOP

If an event occurs, 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.

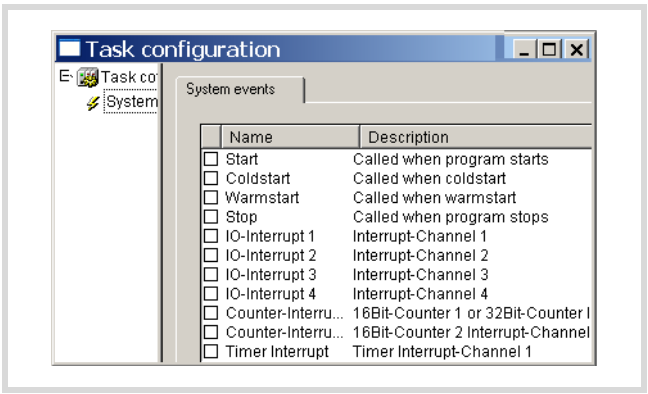



Figure: 47: System events

### 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. I1 has the highest priority, followed by I2, I3 and I4.

### 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.

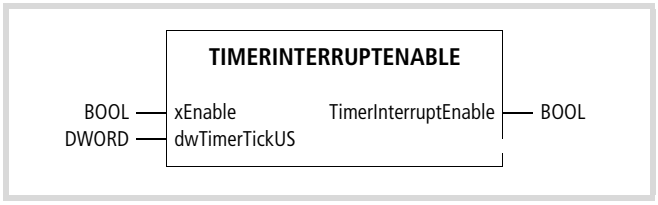



Figure: 48: The TimerInterruptEnable function

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.

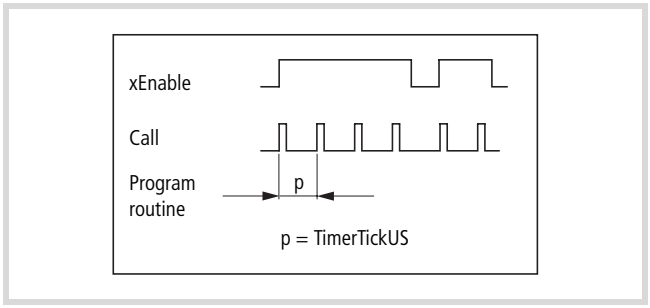



Figure: 49: Periodic calling of the program routine

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)
```

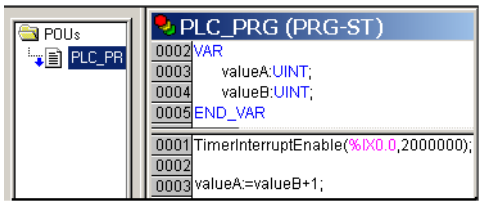



Figure: 50: Including the function in the program

**Example**

- Create a program with a function call

Create a program with the function TIMERINTERRUPTENABLE like in figure 50.

- Writing program routines

▶ 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.

▶ Activate the timer interrupt required by activating the check box on the left of "Timer-Interrupt" name.

▶ In the Called POU column enter the name of the program routine, e.g. "Time_Int".

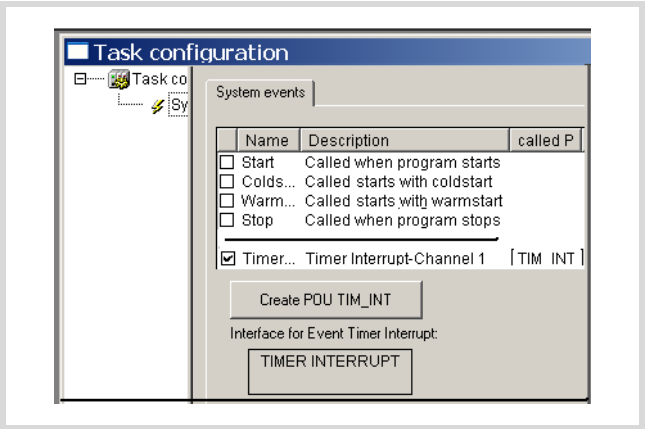



Figure: 51: Creating the program routines

▶ 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:

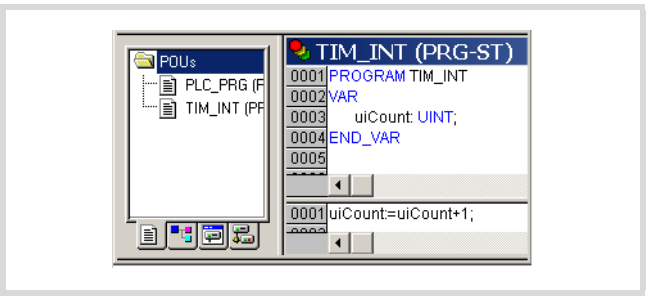



Figure: 52: Writing a program routine

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 inhibited 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 53 shows a list of interrupt sources.

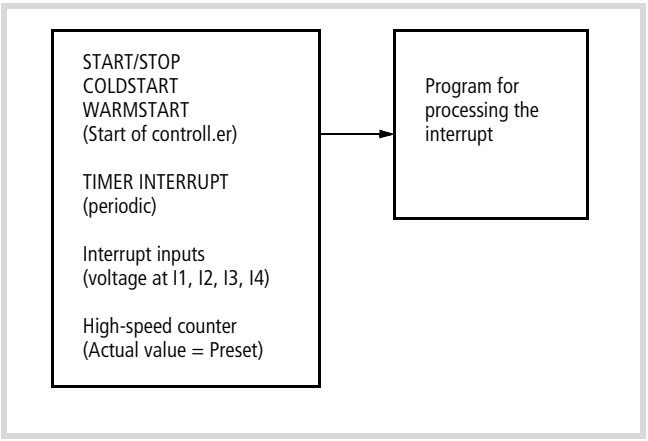



Figure: 53: Interrupt sources

→ Execution of the POU is time-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.

The timer interrupts I1...I4 can be disabled and enabled from the user 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 parameterized 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 I 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 for interrupt processing

## Example for 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 54

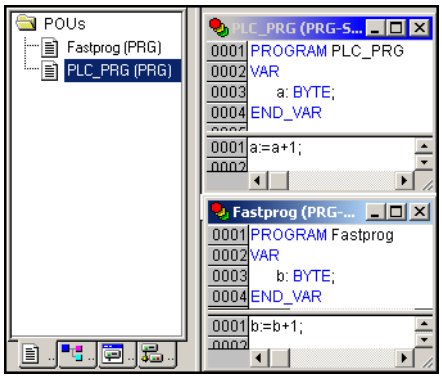



Figure: 54: Writing a program

▶ 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.

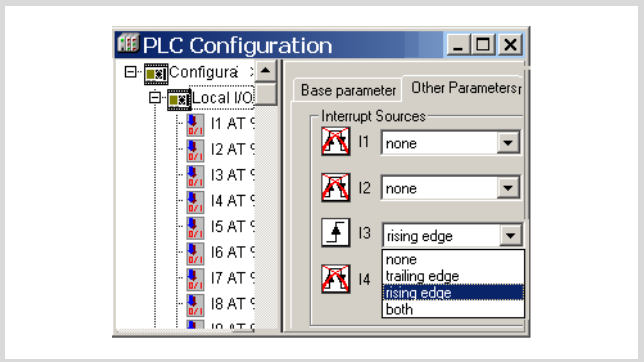



Figure: 55: Interrupt edge selection

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

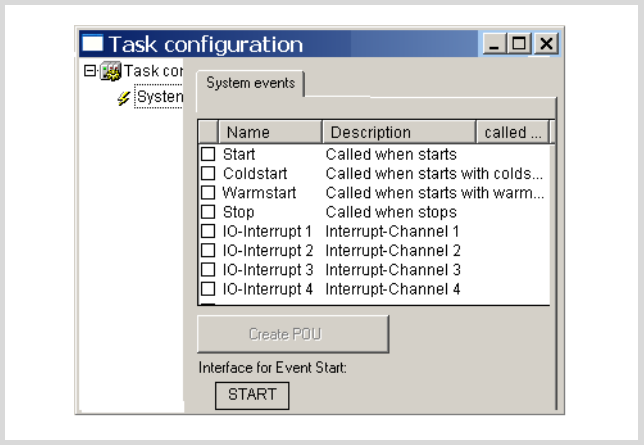



Figure: 56: Enabling an interrupt

▶ 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.
▶ Markieren Sie den Bereich von Spalte „ aufgerufene POU" und Zeile „IO-Interrupt3".
▶ 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:

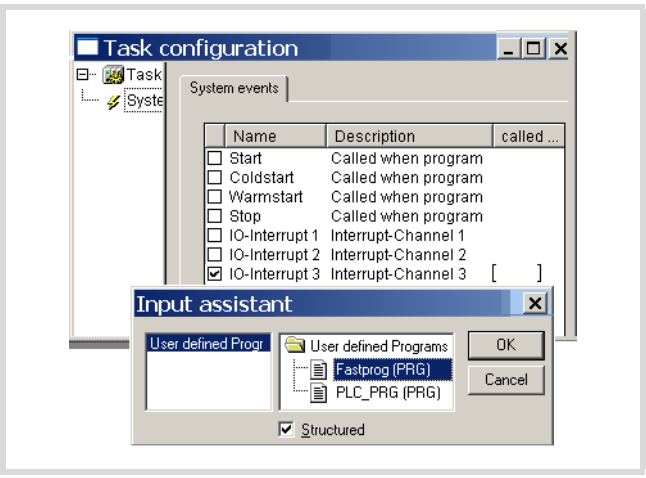



Figure: 57: Allocation of Interrupt source → POU

▶ 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 is not changed if there is an error during processing.

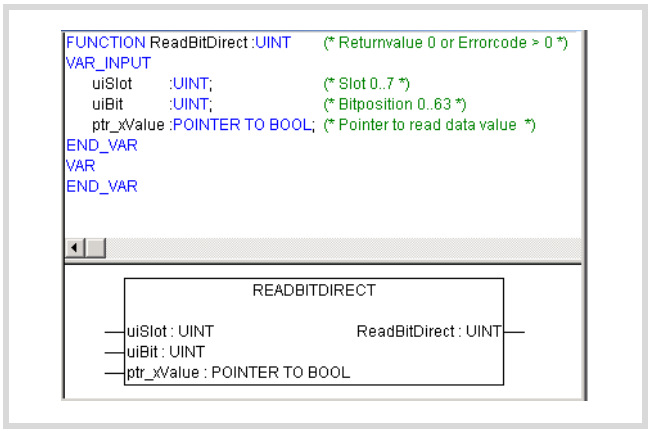



Figure: 58: ReadBitDirect function

Function call: ReadBitDirect(uiSlot, uiBit, ptr_xValue)

The following tables show the access options available:

Table 11: Functions for accessing the I/Os

| EC_UTIL.Lib | EC4-200-I/O | | | |
|---|---|---|---|---|
| | I1 … I12<br>Digital | I7, I8, I11, I12<br>Analog | Q1 … Q8<br>Digital | QA1<br>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 12: 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"

All functions verify as far as possible the validity of the call parameters. If a fault is determined, access is not executed and an error code is output.

The following return values are possible:

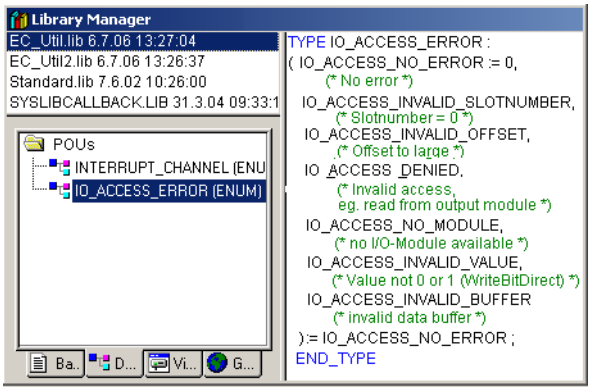



Figure: 59: Return values of the EC-UTIL.Lib functions

Table 13: 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 | | |

## Creating and transferring 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.

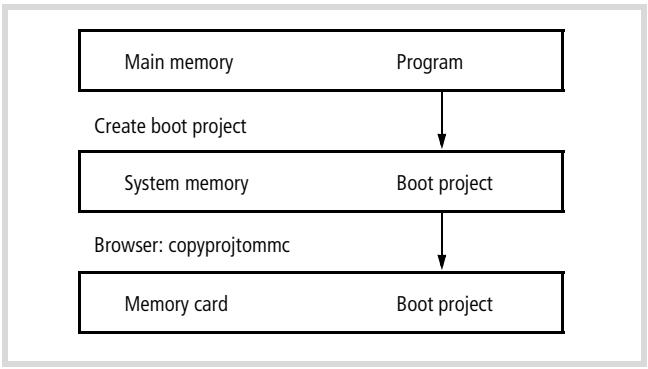



Figure: 60: Saving the boot project

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:

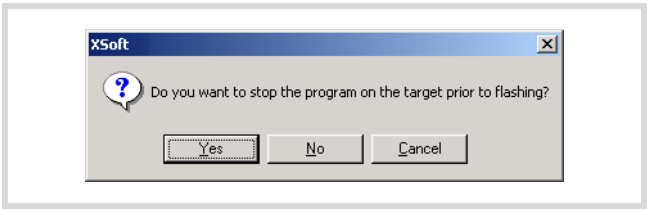



Figure: 61: Creating a boot project

- ▶ Click Yes.

The following dialog appears briefly:

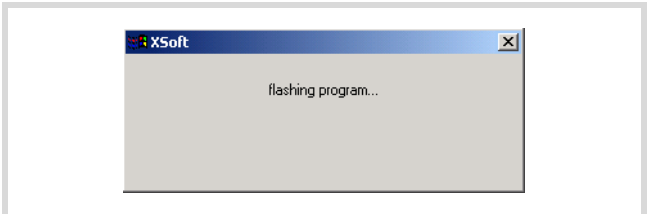



Figure: 62: Creating a boot project

The boot project has been created when this dialog automatically 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.

## Operating system, download/update

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 Moeller website (http://www.moeller.net/support). The latest OS is also supplied on the latest easy Soft CoDeSys CD.

**Caution!**
The download is only possible in offline mode! 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 l PLC Configurator and activate the Common Parameters tab.
▶ Click the Start button in the Update operating system field

The Download operating system dialog opens.

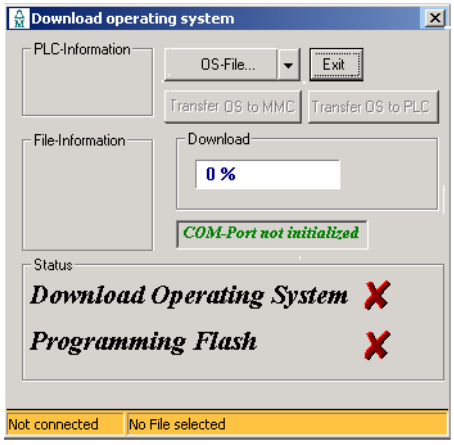



Figure: 63: Download operating system

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).

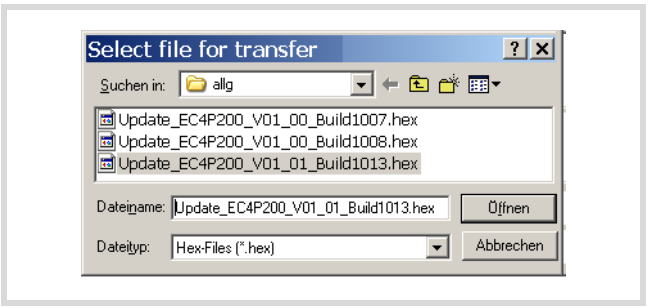



Figure: 64: Selecting the operating system file

The target type and file version are displayed once the OS file is selected.

▶ Click the Transfer Device 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!

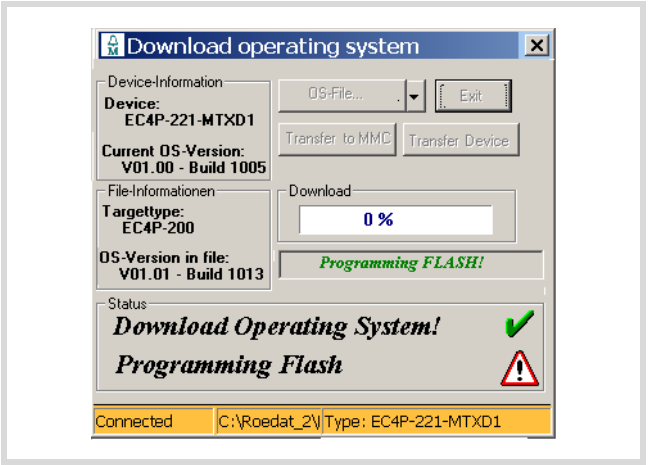



Figure: 65: Warning during download

Wait for the following display.

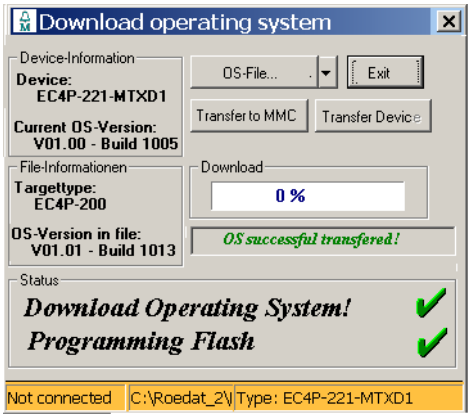



Figure: 66: OS successfully transferred to the PLC

▶ 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 63 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 controller monitor. This is where you enter commands in an entry line and send them as strings to the controller in order to access specific information from it. The response string is shown in a result window of the browser. This function can be used for diagnostics and 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 browser commands available are displayed in the selection field.

▶ 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 Result 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.

The commands are also described in chapter "Resources → PLC Browser" in the manual on the programming software (h1437g.pdf).

The controller supports the browser commands from table 14.

Table 14: Browser commands

| | |
|---|---|
| ? | Get a list of implemented commands. |
| 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 |
| metrics | Output PLC information |
| 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 |
| getrtc | Read real-time clock |
| setrtc* | Set real-time clock |

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

## canload

Displays the load of the CANopen fieldbus.

Example:

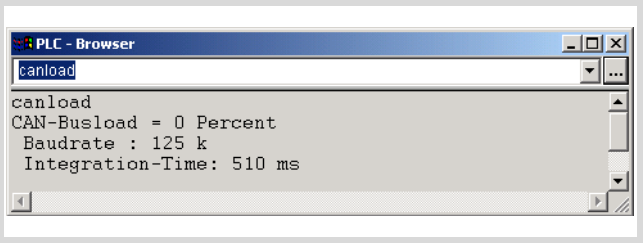



Figure: 67: "canload" browser command

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

- CAN bus load = 0 percent
- Baud rate 125 Kbit/s
- Integration time: 510 ms.

**Caution!**
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, see 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 via the CANopen bus
- Controlling the real-time clock
- Determining the 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 |
|---|---|
| AWB2700-1437 | Standard.lib<br>Util.lib<br>XX_Util. Lib |
| Online help or PDF files<br>(in the Windows start menu via Programs → Moeller Software → easy Soft CoDeSys → Documentation → Automation Manuals | SysLib…pdf |
| AWB2786-1456 | XS40_MoellerFB. Lib/Visu. Lib/… |
| AN2700K20 | 3S_CANopenDevice. Lib<br>3S_CANopenManager. Lib |
| AN2700K19 | 3S_CANopenNetVar. Lib |
| AN2700K27 | SysLibCan. Lib |
| AWB2786-1554 | CANUserLib. Lib<br>CANUser_Master. Lib |

## Installing additional system libraries

You can install libraries manually as follows:


Figure: 68: Libraries, installing manually

▶ In your project, click the Resources tab.
▶ Double-click the Library Manager folder.
▶ Click Insert I 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:

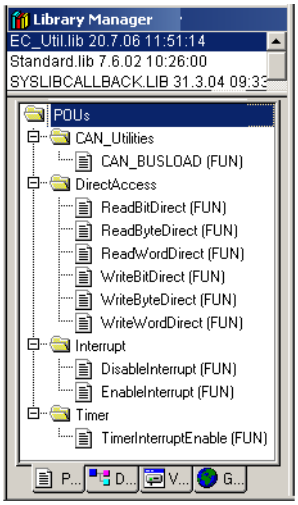



Figure: 69: Functions of the EC_Util.lib library

### CAN_BUSLOAD function

This function can be called cyclically in a user program. If a read cycle has been completed successfully, the function returns TRUE and writes the determined integration time and the bus utilization values to the passed addresses.

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

Information on evaluating the return value is provided in the browser command "canload" on page 60.

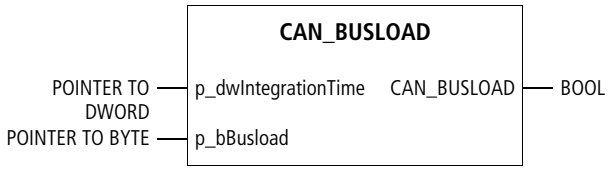



Figure: 70: CAN_BUSLOAD function

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.

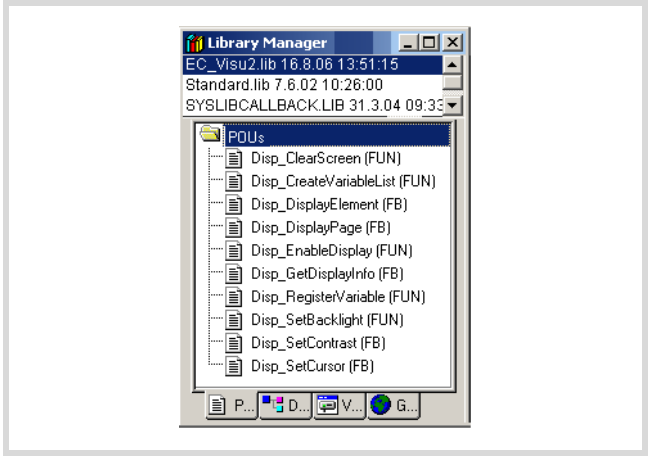



Figure: 71: EC_Visu2.lib library

→ 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 74.

# 12 PC ↔ EC4-200 connection setup

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 72 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 73). These parameter changes must then be adapted again for the PC.

## Communication settings of the PC

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.
- ▶ Define the port (COM1 or COM2 interface), → section "Changing settings"
- ▶ Use the remaining settings as shown in figure 72.
- ▶ Confirm the settings with OK.
- ▶ Log on to the PLC.

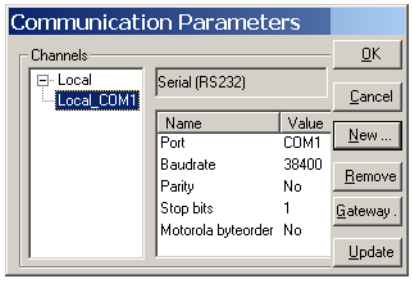



Figure: 72: Defining the PC's communication settings

### 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.

## 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. 57 600 Bit/s as shown in figure 73).

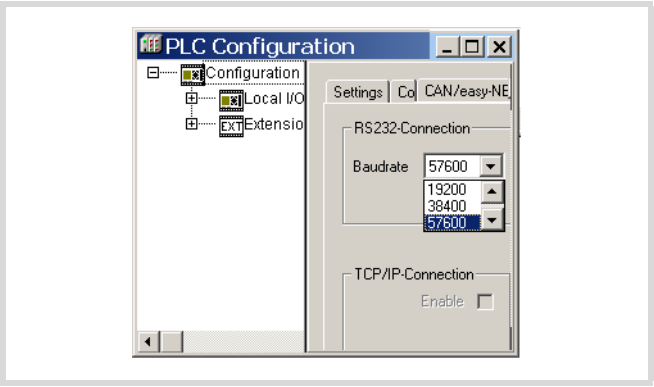



Figure: 73: Specifying the CPU's communication settings

- ▶ Log on to the PLC.

The following prompt appears:

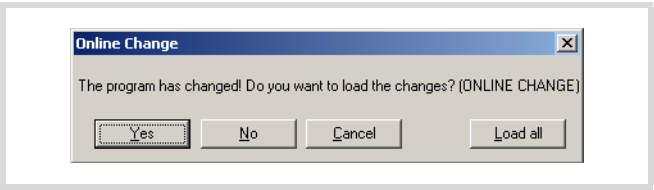



Figure: 74: Confirmation request after program change

- ▶ 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:

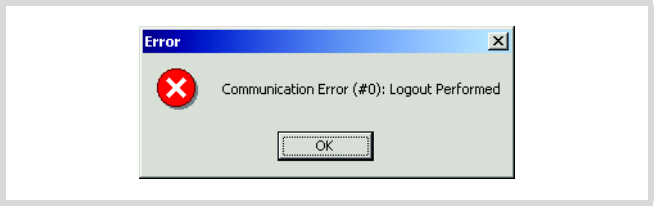



Figure: 75: Communication fault

- ▶ 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.

# 13 Defining system parameters via the STARTUP.INI file

## Overview

You can create project-neutral 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 controller will accept the parameters during the startup. The Startup.INI file is always created with all controller parameters (→ table ).

Table 15: 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 |

The parameters from the INI file have priority over parameters from 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. The system parameters are listed from a specified section (in square bracket) such as [STARTUP], followed by an equals sign and the corresponding 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/BOOTPRJ/".

## 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.

The browser command "createstartupini" can be used to transfer the current system parameters 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 16: Example: STARTUP.INI file for EC4-200

| [STARTUP] |
|---|
| TARGET = EC4P-200 |
| 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. A warning message will appear if you still enter this command. To create a new file, delete the existing one first of all.
→ section "Deleting the Startup.INI file" on page 66.

## Switching on the controller 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 parameters

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

## 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 are loaded from the project at the next startup.
- removeprojfrommmc:
  Deletes the boot project and the INI file on the memory card. The system parameters in the controller are retained.
- 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. It can be connected to another 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 to the PLC
- Reading files from the PLC

Routing offers the advantage of being able to access all routing capable PLCs on the CAN bus from any PLC which is connected with the programming device. You can determine in the project selection which controller you wish to communicate with. This makes it possible to operate remotely configured controllers easily.

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.

### Requirements

The following prerequisites must be fulfilled to use routing:

- Both the routing PLC and the target PLC must support routing.
- Both PLCs must be connected via the CAN bus.
- The PLCs must both have the same active CAN baud rate.
- A 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 carried out without the need to download a user program beforehand (Default:125 Kbaud, Node ID 127). The target controller does not have to be configured as a CAN master or CAN device.

It is possible, for example, to load a program from the PC into the EC4-200 via an XC series controller. In this case, you assign the EC4-200 (target controller) with a routing Node ID.

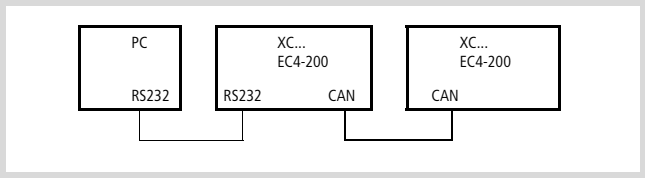



Figure: 76: Program download using routing

### Routing via XC200

In order to carry out a program transfer or routing on a connection between an XC200 and a PC via TCP/IP, you have to set the block size of the data to be transferred. The block size (4 KByte or 128 KByte) depends on the type of transfer (program transfer or routing) and the operating system, → table 17.

Table 17: Block size for the data transfer

| | **Program/file transfer** | | **Routing** | |
|---|---|---|---|---|
| | **OS < V1.03.03** | **BTS ≧ V1.03.03** | **OS < V1.03.03** | **BTS ≧ 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 screen of the programming device monitor will only change erratically (about every 10 seconds).

Routing with the XC200 is possible from OS 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 (access to registry).

Changing the block size:

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

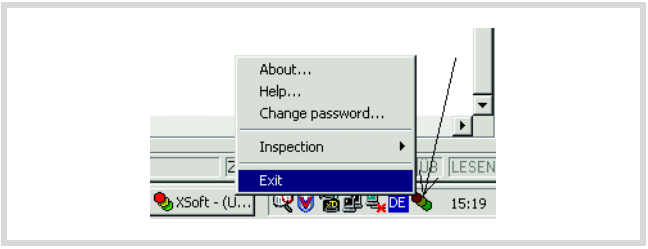



Figure: 77: Closing CoDeSys Gateway Server

▶ 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 of 20 000$_{hex}$ = 128 KByte (default value) in the Registry. |
| BlockSizeRout.reg | Enters a block size of 1 000$_{hex}$ = 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 via a routing 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 does not depend on the configuration (master/device): a target PLC that has not been configured as a master or as a device can be accessed. It only has to be assigned the basic parameters such as Node ID and baud rate, as well as a simple program.

## Addressing

Controllers on the CAN bus can be configured a master or as a device. The controllers are assigned a Node ID/node number (address) so that they can be identified uniquely. The target controller must also be assigned a (routing) Node ID if you wish to access it by means of the routing function. The RS232 or Ethernet interface can be used as a connection between the PC and XC200.

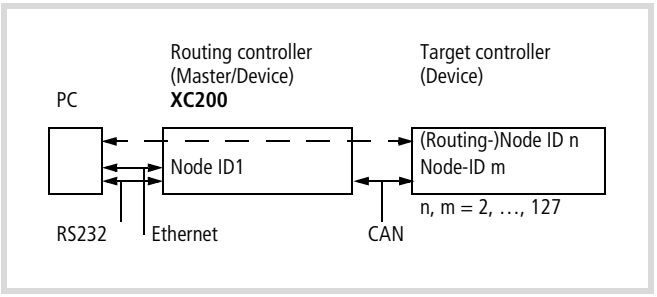



Figure: 78: Routing via the XC200

In the following diagram the routing controller is connected to the PC via the RS232 interface.

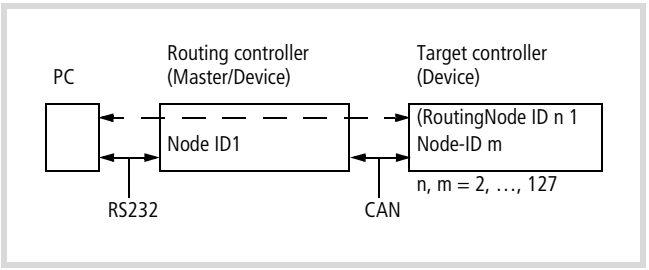



Figure: 79: Routing via XC..., EC4-200

1) Node IDs of the Device function and Node IDs of the Routing function: The (Routing) Node ID must not have the same value as the Node ID (Device)!

Table 18: Example of a Node ID setting, baud rate

| PLC | Function | Node ID | Baud rate | → |
|---|---|---|---|---|
| Routing controller | Master | 1 | 125 KB | figure 81 |
| Target controller | Device | 3 | 125 KB | figure 82 |
| | | 127 (Routing) | | figure 80 |

## Communication with the target controller

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

You can run the following functions:

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

### Note for project creation

You define 2 Node IDs in the target controller:

- One ID for basic communication
- One ID for routing

The Node ID/node number and the baud rate of the target PLC for routing functions are set in the CAN/easy-Net window of the PLC configuration, → figure 80:

- ▶ Enter the baud rate on the CANopen bus and the Node ID/node number in the RS 232 → CAN routing settings field.

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

→ Routing should only be carried out with CANopen baud rates of at least 125 Kbit/s in order to ensure rapid data transfers.

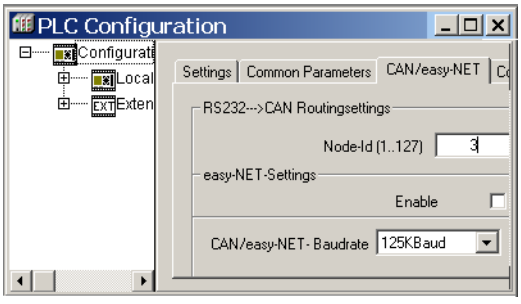



Figure: 80: Routing settings of the EC4-200 target controller

The IDs for basic communication are defined according to the Master/Device function. The settings are made in the master PLC in the CAN Parameters tab (→ figure 81) or in the device PLC in the CAN Settings tab (→ figure 82).

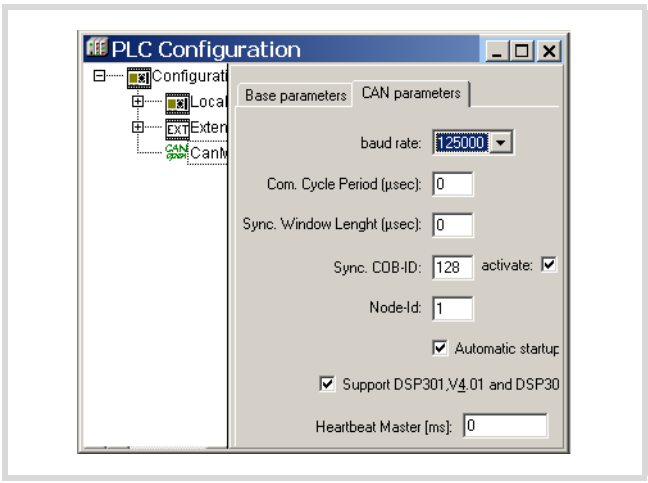



Figure: 81: CAN master parameters

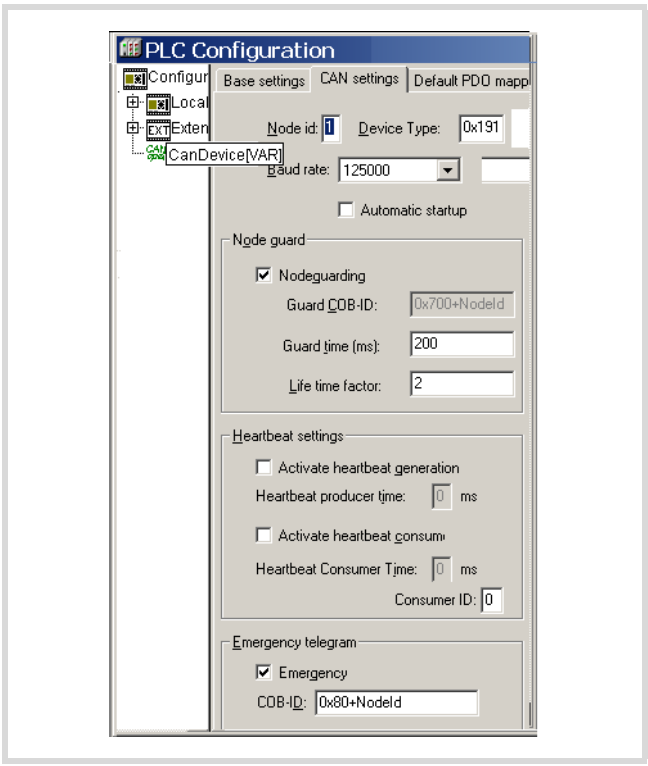



Figure: 82: CAN device parameters

### Example: Accessing a PLC program

The example below illustrates the procedure for accessing a PLC program.

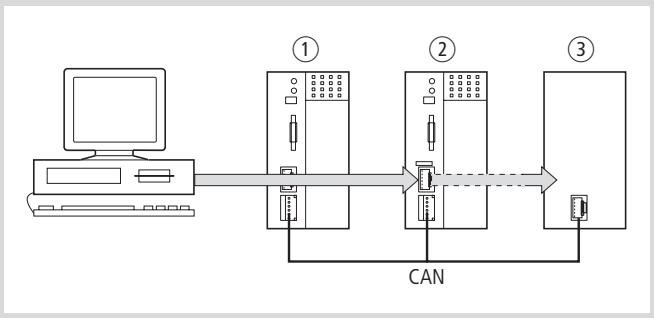



Figure: 83: Diagnostics options

①XC100 with Node ID 1

②XC100 with Node ID 1

③PLC with Node ID 3, e.g. XC100, XC200, XC121, XN-PLC, EC4-200.

You have connected the PC to the PLC with Node ID 2 and want to access the target PLC with Node ID 3.

- ▶ Open the project of the target PLC (Node ID 3) which has the program you wish to edit or test.
- ▶ First configure the parameters for the hardware connection PC n 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 field. XC200: 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".
- ▶ Click OK to confirm. You will return to the initial window.

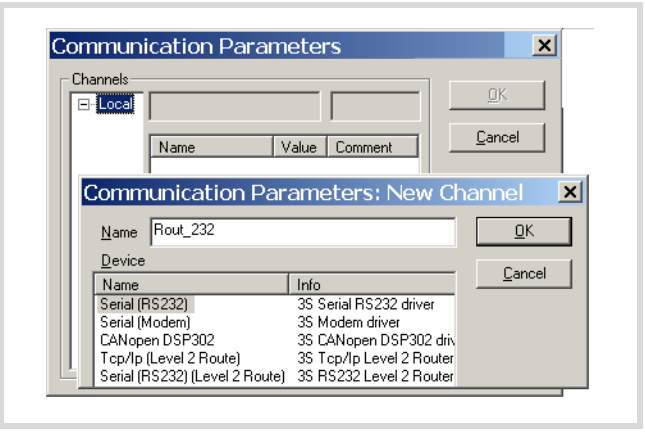



Figure: 84: Channel parameter setting

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

- ▶ Call up the communications parameters in the Online menu once again and select the PLC which you want to program/test.
- ▶ Enter the target ID, number 3 in the example. The target ID is identical to the node 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 3 there and confirm with OK.
- ▶ Log on and carry out the action.

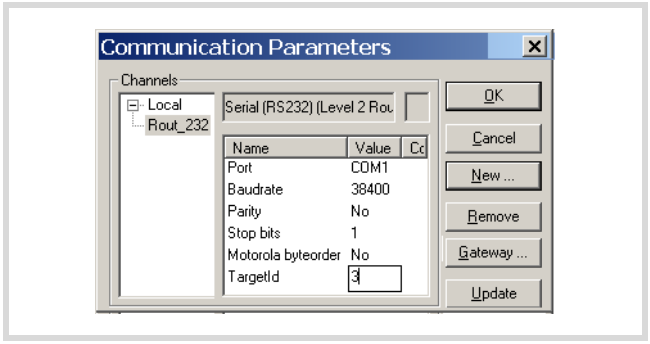



Figure: 85: Setting the target ID of the target PLC

### PLC combinations for routing

The following PLCs support routing:

| From P<br>To O | XC100 | XC100 | XC200 | EC4-200 |
|---|---|---|---|---|
| XC100 | × | × | × | × |
| XC100 | × | × | × | × |
| XC200 | × | × | × | × |
| EC4-200 | × | × | × | × |

# 15 RS 232 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 RS 232 (COM1/COM2 = Multifunction interface) must be switched to Transparent mode via the user program.

Functions for opening and closing the interface, for sending and receiving data, and for setting the interface parameters are provided for the running of Transparent mode. After opening, the interface runs with the current communication parameters that you can adapt by calling the "SysComSetSettings" function.

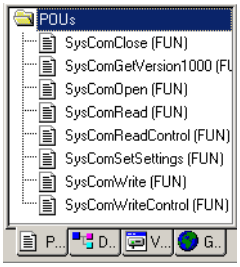



Figure: 86: Function overview

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" (AWB2786-1452GB).

→ Programming via the RS232 interface (COM1) is not possible if it is in Transparent mode. For this Transparent mode must first be deactivated. When Transparent mode is closed, the original communication parameters are reinitialised.

COM1/COM2:

The Transparent mode is automatically deactivated when the PLC state changes to 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 | 1.234568 |
| USINT | 255 | 4 | 2 | 1.23 |
| UINT | 65 535 | 6 | 4 | 1.2345 |
| UDINT | 4 294 967 295 | 11 | 9 | 1.234568 |
| 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 | –1.234568 |

### 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.

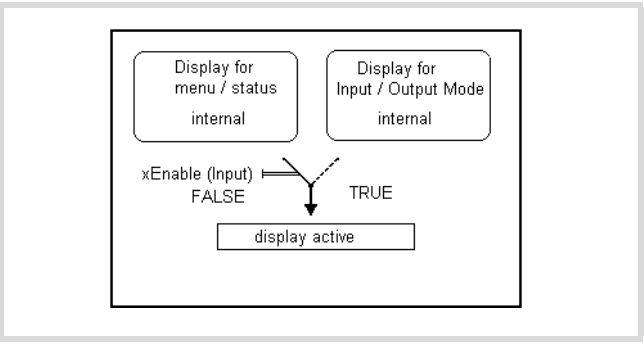



Figure: 87: Changing from Status display ⇔ Entry/output mode

The program must process the function "Disp_EnableDisplay" (→ page 75) 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 87):

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. | Representation on 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 88; 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 - 4 *)
(* 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

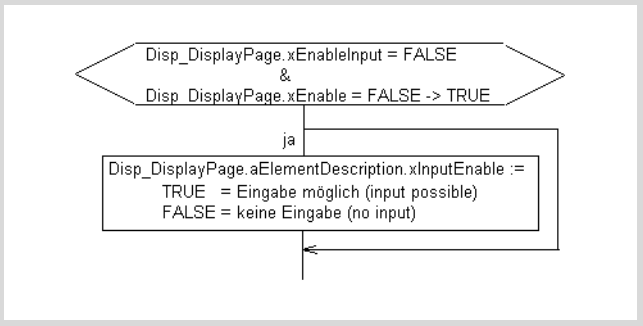



Figure: 88: Activate display/value entry

### 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 90
- 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.

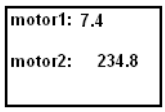



Figure: 89: Example of text and values output

### 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.

- VISUALIZATION
  - 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 type String variable in the list "Global_Variables_Display" as shown in the following example (see also figure 90):

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

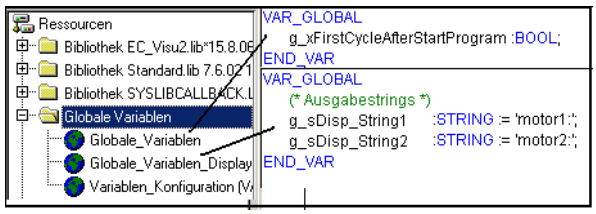



Figure: 90: Declaration of display variables

### 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 90

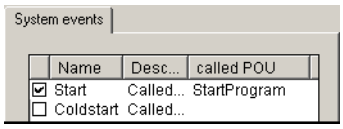



Figure: 91: Defining the system event

### Creating the program "StartProgram"

▶ Write the program "StartProgram" as in → figure 92.

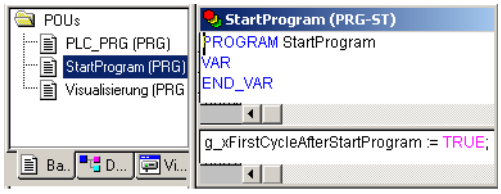



Figure: 92: Creating the "startprogram"

### 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                   (* Symolic 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: 93: 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.

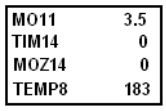



Figure: 94: Example of a page for entries and outputs

### 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.

- VISUALIZATION
  - 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 a variable of type string for each text element you wish to display such as "MO11" in the folder "Global_Variables_Display"
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 according to 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;
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;
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.

## MFD-CP4 multi-function display 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.

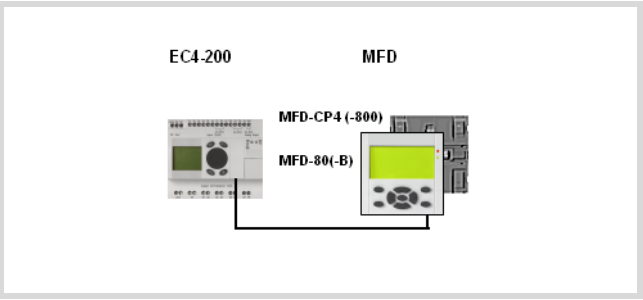



Figure: 95: EC4-200 with MFD-CP4

When the power supply of the MFD-CP4 connected to the EC4-200 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. When the PLC is in Entry/output mode, the MFD-CP4 behaves like the PLC 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 easy-NET stations: Station selection (ID on the easy-NET
    If the EC4-200 is a station on the easy-NET, 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 (AWB2528-1548), chapter Settings)

Main menu: COM…
MENU LANGUAGE…
LIGHTING: 80%
CONTRAST: +1

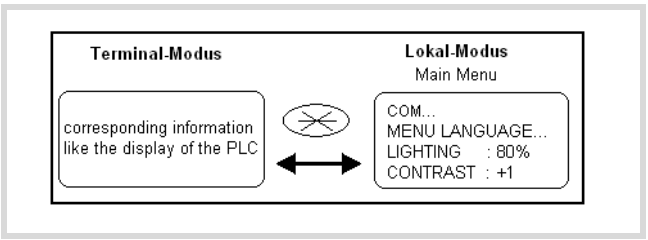



Figure: 96: Changing Terminal mode ↔ Local mode

In Local mode the MFD display buttons are active. See MFD-CP4 power supply/communication module manual (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 (AWB2528-1548G).

# 17 The easy-NET network

## Overview

easy-NET is based on the CAN network that enables the exchange of process and system data. This network is designed for 8 stations (PLCs). Each station is assigned an ID number 1…8 and can exchange data with all other stations. Bit, byte, word and double word data formats are available. The station with ID1 must always be present. This is responsible for managing the communication in the network. When used as a stand-alone station it can communicate with remote I/O stations.

Remote I/O stations are PLCs without a program such as an easy800 for which the outputs are set and the inputs are read by the station with ID1.

If the easy-NET consists of XC200-/EC4-200 and easy800 controllers, both EASY-SOFT and easy Soft CoDeSys are required for the programming and the configuration.

The manual "Description of the SysLibEasyNet.lib for connecting XC Controllers to an easy-NET Network" describes the connection of the XC200/EC4-200 to the easy-NET in detail. This particularly applies to the data transfer functions, including data types and structures.

The following data can be transferred via the easy-NET network:

- User data
  - Input and output states
  - Data from the controller (ID1) to remote I/O controllers for setting the outputs
  - Selective sending of data to a station and receiving data from a station
  - Sending of data on the network for several stations to access (Broadcast).

- System data
  - Run/Stop status display
  - Program status (program present/not present)
  - Synchronisation of device clocks
  - Data for configuring the easy-NET stations
  - Data for creating an online connection between the programming system (Easy-Soft) and any NET station.

→ The chapters on the easy-NET primarily use easy800 in the examples. This can be replaced by a networkable MFD visualisation device for the same easy-NET function.

## Sending/receiving user data

The EC4-200 accesses the data of the easy-Net stations by using the functions NET_UPDATE and NET_GET in the application program. These functions are contained in the library SysLibEasyNet.lib. The actual data transfer between the EC4-200 and the individual stations is handled by the protocol task that operates separately to the program and is not visible to the user. Calling the functions causes the data to be exchanged between the application program and the protocol task.

The protocol task also handles administrative services such as station monitoring, clock synchronisation and the enabling of program accesses.

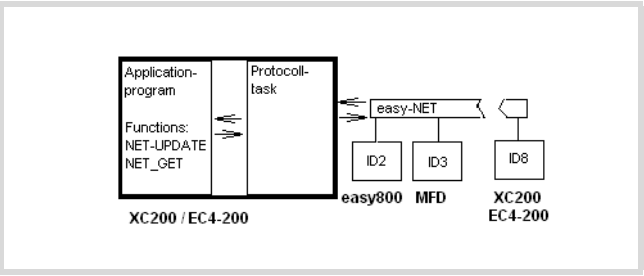



Figure: 97: Data transfer between EC4-200 and easy-NET stations

### NET_UPDATE function

The function NET_UPDATE executes data exchange between the user program and the protocol task. The following data is exchanged via the structure EASY_NET_MAIN:

From the protocol task to the user program:

- Status information (Info)
  - Own NET-ID
  - Baud rate used
  - Last error code
  - For every station:
    Availability
    Status (Start/Stop)
    Program present (Yes/No).

- Receive user data for each station (RCV[X])
  - Local inputs (wI) of the station X
  - Local outputs (byQ) of the station X
  - Inputs of the expansion device (wR) of the station X
  - Outputs of the expansion device (byS) of the station X
  - Directed data telegram (dwSN) of the station X to the EC4-200.

From the user program to the protocol task:

- Send user data via the easy-NET
  - Local and extended outputs (byQ/byS)
  - Local and extended inputs (wI/wR)
  - Send data to a selected station X (aToID[x].dwSN)
  - Set local outputs of stations without own program (aToID[x].byQ/aToID[x].byS ). This operation is only possible from NET-ID1.
  - Send data with the PUT command (Broadcast).

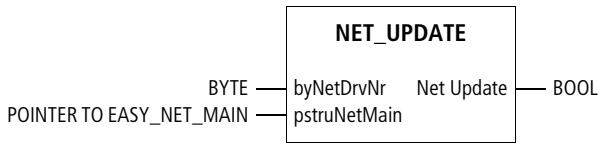



Figure: 98: NET_UPDATE function

The NET_UPDATE function must be called once every user cycle. This ensures that the current data of the easy-NET stations is always available and that the easy-NET stations are provided with the current data of the local station.

**NET_GET function**
This function enables an easy-NET station to fetch a data value that other easy-NET stations have sent to the bus with the PUT command.

A data value can be fetched with each call of the NET_GET function. The byNET_ID and byModulNumber entries of the EASY_NET_GET structure select the data value "present in the network". When the function is called, the protocol task enters the current data in the structure. Multiple calling of the NET_GET function enables any number of data values to be read.

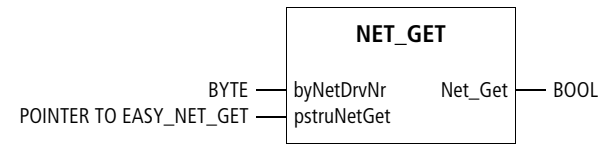



Figure: 99: NET_GET function

**Data transfer options**

There are three data transfer options:

- Data transfer between a PLC (ID1) and remote I/O device, → page 86
- Transfer of bit data blocks between several PLCs, → page 87
- Transfer of D words (32-bit) according to the PUT-GET principle between several PLCs, → page 88

**Data transfer between a PLC (ID1) and remote I/O device**
The input and output states of PLCs or remote I/O devices can be read by all other stations. If an easy800 is used as a remote I/O device, for example, the PLCs on the easy-NET can scan the inputs of the easy800. The PLC with ID=1 can also set the outputs of the easy800. This also applies to the easy800 expansion (Exp.).

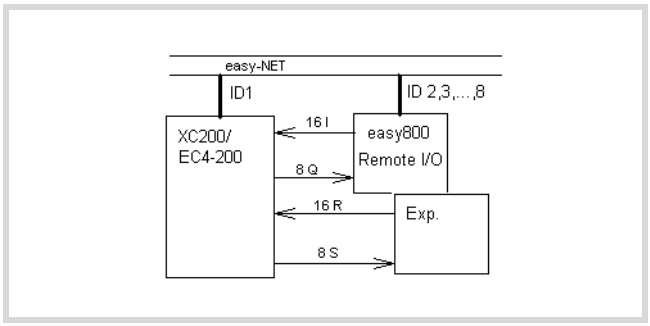



Figure: 100: Data transfer between EC4-200 and Remote I/O

To set the outputs and scan the inputs of the easy800 (ID2), call the NET_UPDATE function in the user program of the EC4-200 (ID1) as shown in the following program example.

Enter a pointer at the "pstruNetMain" input to a structure of type EASY_NET_MAIN declared in the user program. In this structure enter the transfer data → figure 101.

The value 5 (1 byte) is entered in the program via the structure variable:

```
NET_MAIN.SND.aToID[2].byQ:=5;
```

When the EC4-200 PLC is in RUN, the value is transferred to the outputs of the easy800 after every change.

The 16 inputs of the easy800 are accepted by the PLC with the structure variables NET_MAIN.RCV[2].wI.

```
PLC_PRG (PRG-ST)
0001 PROGRAM PLC_PRG (*XC200/EC4-200(ID1)<->easy800(ID2)*)
0002 VAR
0003     NET_MAIN: EASY_NET_MAIN;
0004     INPUTS: WORD;
0005 END_VAR
0001 Net_Update(byNetDrvNr :=0, PstruNetMain:=ADR(NET_MAIN));
0002
0003 NET_MAIN.SND.aToID[2].byQ:=5;
0004       (*XC200/EC4-200 -> easy800 Output*)
0005
0006 INPUTS:=NET_MAIN.RCV[2].wI;
0007       (*XC200/EC4-200 <- easy800 Input*)
0008
0009
```

Figure: 101: Program example of a data transfer between EC4-200 and remote I/O

figure 102 shows the online display of the program from which the send and receive data originates. Note that the output value 16#05 (aToID[2].byQ) sent to the easy800 is received by the easy800 and is automatically sent back to the input (RCV[2].byQ).

```
0001 NET_MAIN
0002     INFO
0003     SND
0004         byQ = 16#00
0005         byS = 16#00
0006         wI = 16#0000
0007         wR = 16#0000
0008         PUT
0009         aToID
0010             aToID[1]
0011             aToID[2]
0012                 byQ = 16#05 (* to easy800 *) ->
0013                 byS = 16#00
0014                 dwSN = 16#00000000
0015             aToID[3]
0016             aToID[4]
0017             aToID[5]
0018             aToID[6]
0019             aToID[7]
0020             aToID[8]
0021         xStartClockSynchronisation = FALSE
0022         xAcknowledgeActiveError = FALSE
0023     RCV
0024         RCV[1]
0025         RCV[2]                                  1)
0026             byQ = 16#05 <- (* to XC200 *) -
0027             byS = 16#00      (*EC4-200*)
0028             wI = 16#2007   2)
0029             wR = 16#0000
0030             dwRN = 16#00000000
0031         RCV[3]
0032         RCV[4]

1) easy800 Inputdata (16#05) return to XC200 /EC4-200
2) Input Data: Hex. 007 = 7 dez.
```

Figure: 102: Online display of the Data transfer between EC4-200 and remote I/O

### Transfer of bit data blocks between several PLCs

Each PLC can send a 32-bit data block to another specified PLC via the easy_NET. To do this, call the NET_UPDATE function in the program of the EC4-200 PLC and enter the station number X of the receiver via the structure (EASY_NET_MAIN) (SND.aToID[X].dwSN). Write the data to be sent to the variable dwSN.

To receive data from other PLCs, use the structure (EASY_NET_MAIN).RCV[x] in the EC4-200, in which x represents the NET-ID of the station from which the data is to be received.

From the figure 103 it is assumed that every PLC contains a one dimensional array for the data blocks with 8 elements for sending and receiving. There is a direct allocation between the send element of a station and the receive element of another station.

The position of the element (aToID) in the send array and (RCV) in the receive array of the EC4-200 corresponds to the ID of the receiving/sending station.

The received value is in an element of the receive array of a EC4-200 with a position that is identical to the ID of the send station.

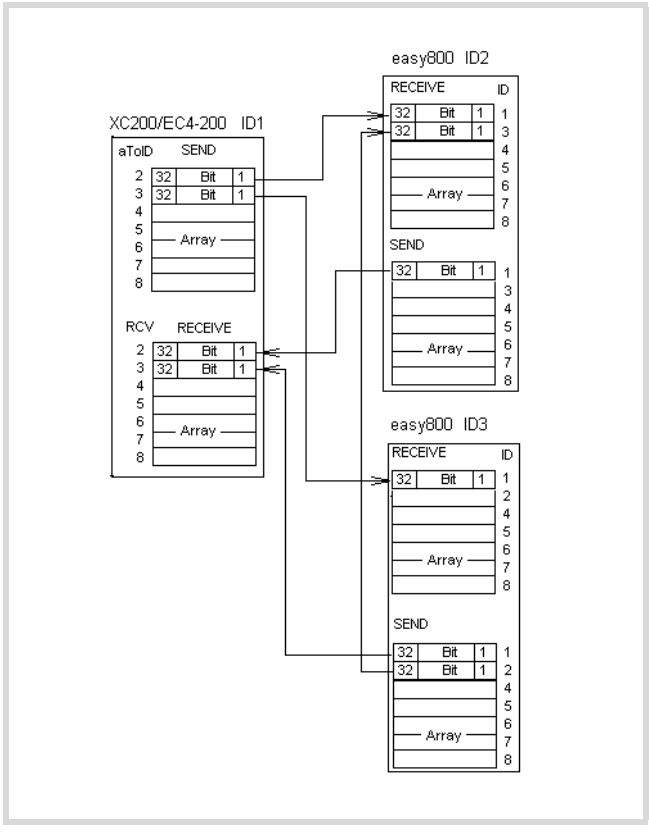



Figure: 103: Overview of data blocks

1st connection:

The value "a" (32-bit) is transferred from EC4-200 (ID1) to the easy800 (ID2).

The following structure must be programmed in the EC4-200:

```
NET_MAIN.SND.aToID[2].dwSN:=a;
```

The value can be processed in the easy800 by the scanning of the input bits 1RN1 to 1RN32.

2nd connection:

A data block with the value 7 ($3_{hex}$) has to be sent from the easy800 PLC (ID3) to the EC4-200 (ID1).

The 3 outputs 1SN1, 1SN2, 1SN3 must be programmed and set in the easy800.

The following structure variable must be programmed for scannning in the EC4-200:

```
INPVAL:=NET_MAIN.RCV[3].dwRN;
```

The variable INPVAL provides the value.

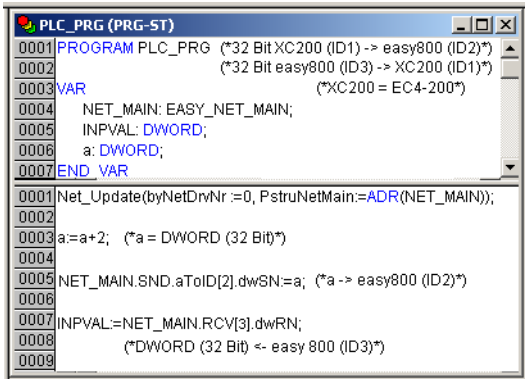



Figure: 104: Program example of the transfer of bit data blocks

figure 105 shows the online display of the program from from which the send and receive data originates.figure 104

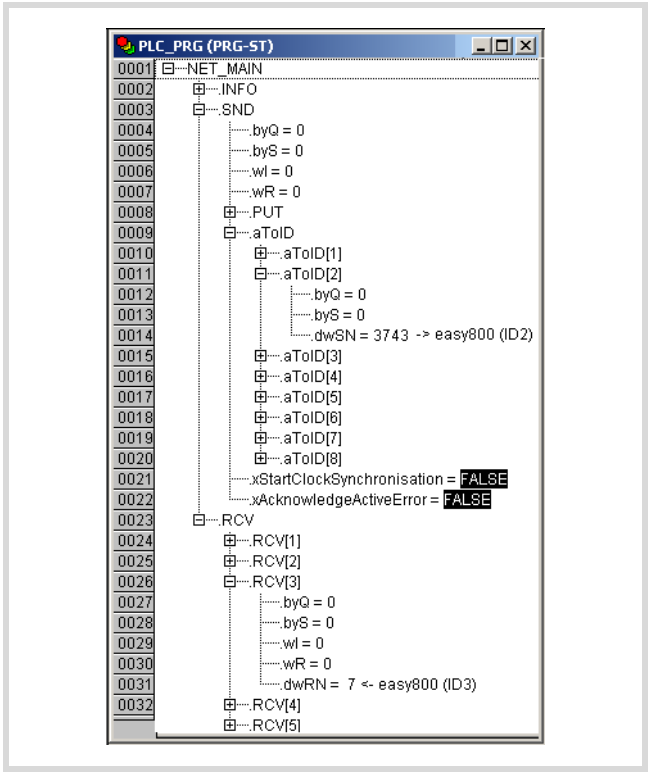



Figure: 105: Online display of the program

**Transfer of D words (32-bit) according to the PUT-GET principle between several PLCs**

PUT = Data to network; GET = Data from network

According to this principle, a PLC sends data (D word) to the network (PUT) that is identified by a module number (MN number for EC4-200, PT number for easy800). The other PLCs can scan the data by stating the ID of the send PLC and the module number (GET).

Each PLC can put data sequentially to the network (PUT) with the MN/PT numbers 1, 2, … 32.

The NET_UPDATE function must be called on the EC4-200 for the PUT operation and the elements of the structure variables EASY_NET_PUT/byModuleNumber and /dwData.

For the GET operation, the NET_GET function must be called and the elements of the structure variables EASY_NET_GET/byModuleNumber and /dwGetData defined.

The PUT and GET functions on the easy800 are implemented with function blocks.

In the following example, the EC4-200 (ID1) sends the data $3747_{hex}$ (MN1) to the network (PUT). The easy800 PLCs scan this data with a GET function block. The function block number (PT) in the easy800 PLCs (ID2 and ID3) depends on the MN number of the send data, in this case MN1 → PT1.

The easy800 (ID3) provides two data units on the network using a PUT function block for each one. The function block for the second data unit with the value $9774_{hex}$ is assigned the PT number 2.

The EC4-200 scans this data unit. For this the following structure variable entries are required:

```
(EASY_NET_GET).byModuleNumber := 2;
(EASY_NET_GET).byNET_ID := 3;
```

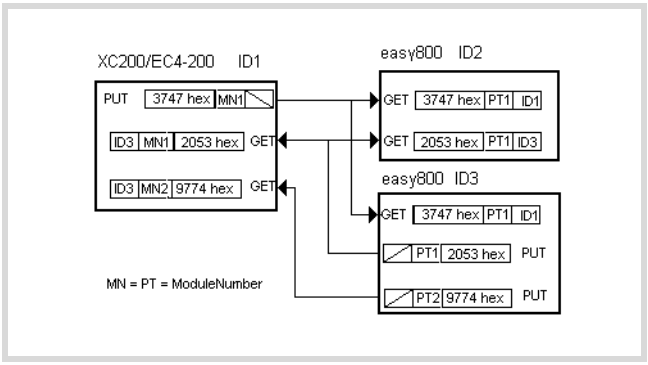



Figure: 106: Transfer of data according to the PUT-GET principle

The following program sequence can be used in the EC4-200 in order to execute a PUT operation.

If the input IX0.0 is set High (=TRUE), the program sequence is started after xTransferPending is switched to FALSE. The value of the variable "Datum" (MN1) is sent to the network.

```
PLC_PRG (PRG-ST)
0001 PROGRAM PLC_PRG  (* D-WORD XC/EC4 -> NET*)
0002 VAR
0003     NET_MAIN: EASY_NET_MAIN;
0004     Datum: DWORD;
0005     NewToTransmit: BOOL;
0006     INP AT%IX0.0:BOOL;
0007 END_VAR
0001 Net_Update(byNetDrvNr :=0, PstruNetMain:=ADR(NET_MAIN));
0002 IF NOT NET_MAIN.SND.PUT.xTransferPending AND NOT
0003         NET_MAIN.SND.PUT.xStartTransfer THEN
0004     IF INP THEN
0005         NET_MAIN.SND.PUT.byModuleNumber:=1;
0006         Datum:=Datum+3;
0007         NET_MAIN.SND.PUT.dwData:=Datum;
0008         NewToTransmit:=TRUE;
0009     END_IF
0010 END_IF
0011 IF NewToTransmit THEN
0012     NewToTransmit :=FALSE;
0013     NET_MAIN.SND.PUT.xStartTransfer:=TRUE;
0014 END_IF
0015 IF NET_MAIN.SND.PUT.xTransferPending THEN
0016     NET_MAIN.SND.PUT.xStartTransfer:=FALSE;
0017 END_IF
```

Figure: 107: Transfer of data according to the PUT-GET principle (PUT)

figure 108 shows an example how the NET_GET function is to be used. In this example, the device with ID3 scans the value of the module with the number 2.

At input "pstruNetGet", a pointer to a structure of type EASY_NET_GET declared in the user program must be entered from which you read the data that is sent to the network with a PUT operation from other stations → figure 108.

```
PLC_PRG (PRG-ST)
0001 PROGRAM PLC_PRG
0002   (*D-WORD easy800(ID3) -> XC200 EC4-200 (ID1)*)
0003 VAR
0004     Getdatastruct: EASY_NET_GET;
0005     xSuccess: BOOL;
0006     RecData: DWORD;
0007     Errorprog: BYTE;
0008 END_VAR
0001 Getdatastruct.byNET_ID:= 3;
0002 Getdatastruct.byModuleNumber:= 2;
0003
0004 xSuccess := Net_Get(
0005     byNetDrvNr:=0,
0006     pstruNetGet:=ADR(Getdatastruct));
0007 IF xSuccess THEN
0008     IF Getdatastruct.xNewDataReceived THEN
0009     RecData:=Getdatastruct.dwGET_Data;
0010     END_IF
0011 ELSE
0012     (*Error handling*)
0013     Errorprog:=Getdatastruct.byError;
0014 END_IF
```

Figure: 108: Transfer of data according to the PUT-GET principle (GET)

## Configuring EC4-200 with easy800 on the easy-NET

In easy-NET every controller must be configured. Each controller is assigned a network address, the ID and parameters like SEND IO that define the behaviour of the PLC in the network.

The configuration is carried out in the programming software of the PLC:

- EASY-SOFT for easy800
- easy Soft CoDeSys for XC200 and EC4-200

EASY-SOFT provides the support required for this configuration. As soon as you add other PLCs to a single PLC, they are connected symbolically via the network. The first PLC is assigned the ID = 1. The IDs for the other PLCs and the parameters must be set as follows:

▶ Click the "Communication Parameters" tab.
▶ Select the NET-ID via the NET-ID menu.
▶ Activate/deactivate the Sende IO and Remote RUN functions in the NET Configurator field (only for ID = 2…8).

All controllers must be integrated in the network of easy800 with XC200 or EC4-200 controllers using the EASY-SOFT Configurator. This also applies to the XC200 and EC4-200. You also have to assign the parameters for the XC200 and EC4-200 in the configurator of easy Soft CoDeSys.

▶ Create a circuit diagram for each easy800 and a program for the EC4-200.
▶ Connect the PC to a PLC and load the program/circuit diagram including the configuration in the relevant PLC.

Other configuration options are described in section "Programming via easy-NET (Routing)" on page 90.

## Configuration in the easy SOFT CoDeSys

Configure and program the EC4-200 with the easy Soft CoDeSys programming software.

- ▶ For this activate the easy-NET setting in the CAN/easy-NET tab of the configuration.
- ▶ Enter the ID number under "easy-NET-ID". This ID number must be identical to the ID number that you assign to the EC4-200 in the EASY-SOFT configurator!
- ▶ If necessary, activate the functions Remote RUN and Send I/O.

### Remote RUN

Only for controllers with ID 2 … 8: If this function is active controllers with ID 2 … 8 follow the RUN/STOP status of the master

### Send I/O

- Off: Values for local inputs/outputs byQ; byS; wI; wR are sent cyclically (cycle time depends on the CAN/easy-NET baud rate set)
- On: Local inputs/outputs are sent cyclically and in the event of a change.

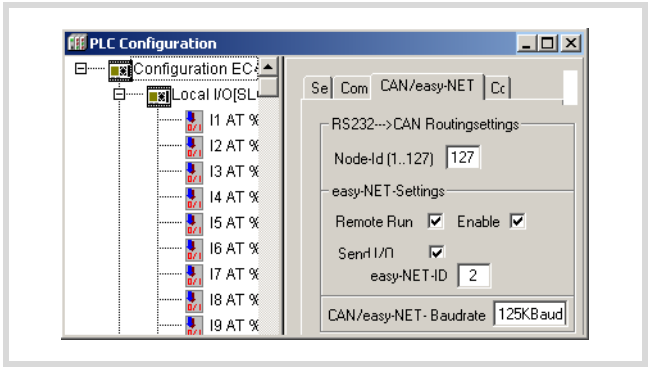



Figure: 109: Configuration of the EC4-200 as easy-NET station

### Controlled configuration via EC4-200

An EC4-200 that is configured with ID 1 can be used to configure easy800 controllers via the easy-NET:

- Via the entries on the display of the EC4-200
  - – On the PLC display enter in the System menu via CONFIGURATOR → NET → NET PARAMETER the values for the network such as baud rate and bus delay.
  - – Activate/deactivate the SEND IO and REMOTE RUN functions applicable to all stations.
  - – Assign an ID to the stations 2…8 via the menu NET → STATION (PLC).

The NET PARAMETERS and the STATION ID are transferred to the individual stations by confirming the menu selection NET → CONFIGURE.

The menus are described in detail in the section Menu structure

- Via the parameterisation of the EC4-200 in the easy Soft CoDeSys configurator

The configuration is carried out in 2 steps:

1. Parameter definition in the configurator

- Add a tick to the section Activate in the easy-NET Settings area on the CAN/easy-NET tab
- Set the NET-ID = 1.
- Activate/deactivate the functions Remote Run (ID = 2 … 8) and Send I/O.
- Assign an ID to the other controllers:
  Click the Configure easy-NET button and select in the subsequent view the IDs for the stations.
- Enter the baud rate in the CAN/easy-NET Baudrate field.

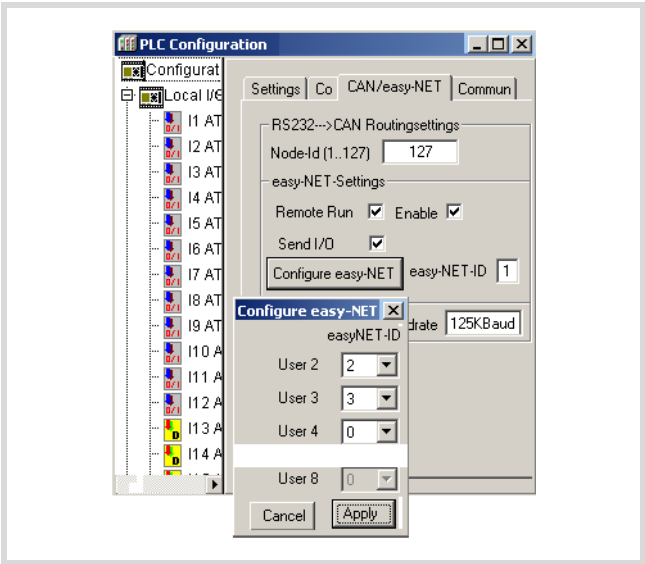



Figure: 110: Parameterisation

2. Transferring the parameters and assignment of IDs using the NET_CONFIG function.

By calling the function, the parameters that you have created in the configurator are transferred to the individual controllers. The function is contained in the library SysLibEasyNet.lib. The operation of the function and the incorporation in the user program are described in the manual "Description of the SysLibEasyNet.lib …".

## Programming via easy-NET (Routing)

Routing means to establish an online connection from a programming device (PC) to any (routing-capable) PLC in an easy-NET network without having to directly connect the programming device to the target PLC. It can be connected to another 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)

Routing offers an advantage which makes it possible to access all routing capable PLCs on the easy-NET bus from any PLC which is connected with the programming device. This makes it possible to operate remotely configured controllers easily.

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.

The following requirements must be fulfilled in order to use routing:

- Both the routing PLC and the target PLC must support routing.
- Both PLCs must be connected via the bus.
- The PLCs must have the same active bus baud rate.

### Routing via EC4-200

The routing options available are listed in figure 111 and figure 114.

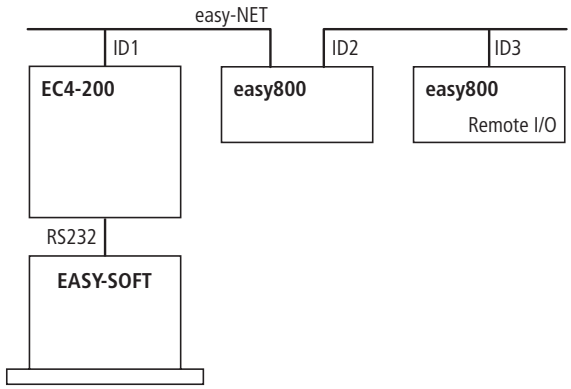



Figure: 111: Routing via EC4-200 (ID1) to easy 800

As shown in figure 111, the PC with the EASY-SOFT programming software can be connected to the EC4-200 if the configuration of the EC4-200 already contains an ID (see Configuration).

Establish a connection between the PC and the EC4-200 in the following way:

▶ In the Communication/Connection menu in EASY-SOFT select the COMx interface and the baud rate (standard baud rate of the EC4-200: 38400 Bit/s).

▶ Click the Online button. The connection between the PC and the connected PLC is established.
▶ Under Device select a target PLC with which you wish to communicate.


Figure: 112: Program download in Online mode

You can then carry out the following functions in the operating fields:

| Field | Function |
|---|---|
| Clock | Set the local time and synchronise the device times in the network. |
| Program | Change between RUN and STOP. |
| Display | View the status of the easy-NET variables and the device information. |

If you have selected an easy800 as the target PLC, you can execute the following functions in the Program field:

- Program download
- Online modifications.

▽ Attention!
Communication access via the easy-NET increases the bus load by up to 15 percent.

You can change the following settings.

| | |
|---|---|
| Baud rate | In the Project menu in the network overview |
| Bus delay | |
| REMOTE RUN | Depending on the PLC selected in the Communication parameter tab → NET Configurator |
| Send IO | |
| NET-ID | |

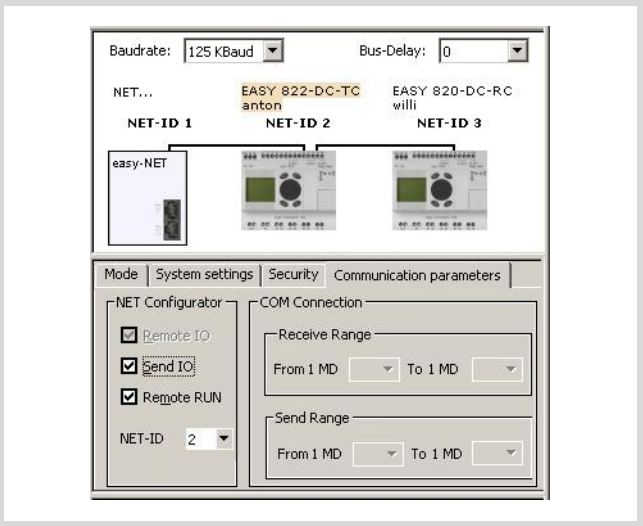



Figure: 113: Communication parameters of easy800 (ID2)

Transfer the new settings to the PLC in the following way:

- ▶ Click the PC => Device button in the menu ‹Communication → Program›.
- ▶ The settings can be read from the PLC by clicking the Device => PC button.

You can also connect the PC to another routing-enabled controller such as the easy800 with ID 3. If you then select the EC4-200 as target PLC, you can carry out the follwing operations with the EC4-200 after calling the EASY-SOFT communication menu:

- Starting/stopping
- Setting the time
- Displaying device information

The routing options are also available if you configure an easy800 with ID1 and the EC4-200 with ID2 as shown in figure 114. The PC with EASY-SOFT can be connected to any easy-NET-configured PLC. You can then carry out the functions described in the previous sections.

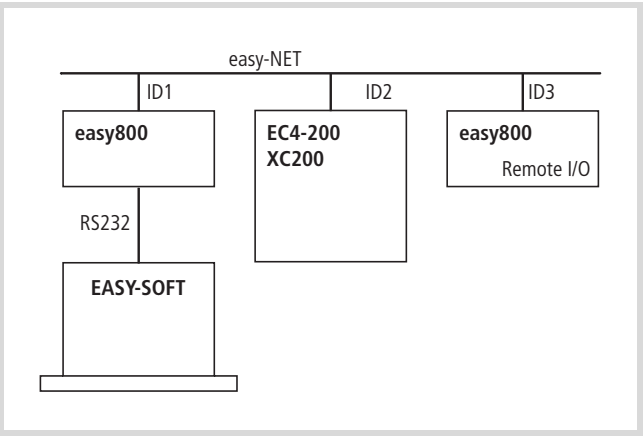



Figure: 114: Routing via easy800 (ID1) to EC4-200/easy800

## Bus topology

A bus topology can be configured as a linear topology with optional spur lines. The ends of the bus must be terminated by bus terminating resistors (120 Ohm).

figure 115 shows two connection options.

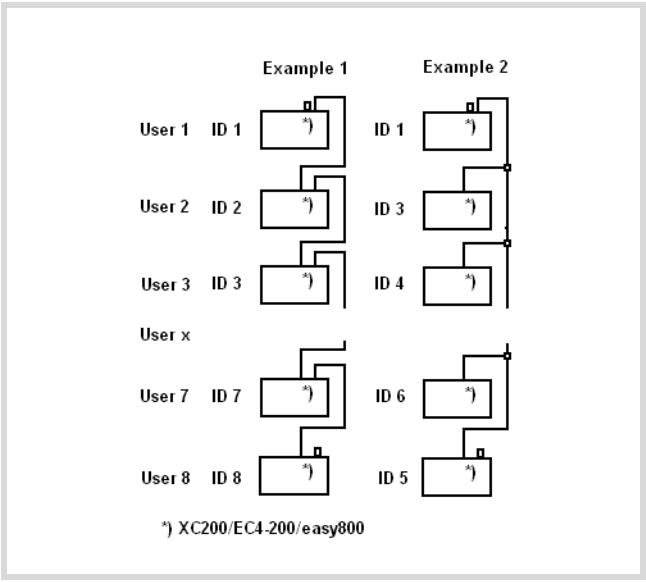



Figure: 115: PLC connection options

Example 1: Physical location matches ID Example 2: Physical location does not match ID

The PLC on the physical location 1 is always assigned ID1. The 7 other controllers are connected to this PLC via the easy-NET.

As the EC4-200, like the easy800, has two easy-NET terminals with additional signal cables (SEL_IN/OUT), they can be incorporated in the easy-NET like an easy800. This requires that they have a minimum configuration, i.e. a project with an easy-NET configuration must be loaded in the EC4-200. As long as this condition is fulfilled, all PLCs can be connected to easy-NET.

Table 19: Signal connection between the EC4-200 and easy800

| **EC4-200** ↔ | **easy800** |
|---|---|
| ECAN_H | ECAN_H |
| ECAN_L | ECAN_L |
| GND | GND |
| SEL_IN | SEL_OUT |
| SEL_OUT | SEL_OUT |

# 18 EC4-200 network modules

The EASY205-ASI, EASY221-CO, EASY204-DP, EASY222-DN network modules enable you to connect the EC4-200 as a slave to ASI, CAN, PROFIBUS-DP or DeviceNet (→ table 20) networks. The controller can also be integrated as a station in an easy-NET network.

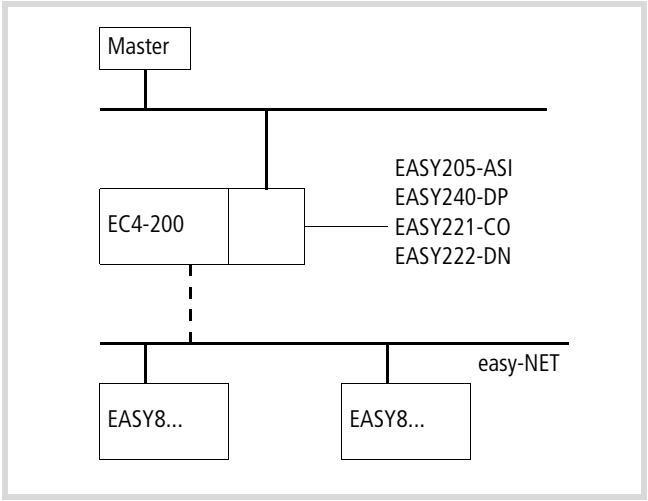



Figure: 116: EC4-200 wih network connection

The type of data exchange between master and network modules is shown in table 20.

Table 20: Overview of network modules

| 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 21). 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 21: Manuals on the network modules

| Part no. | Manual (AWB) |
|---|---|
| EASY204-DP | AWB2528-1401GB |
| EASY221-CO | AWB2528-1479GB |
| EASY222-DN | AWB2528-1427GB |

## EASY205-ASI

### Cyclic data exchange

The master sends 8 bits to the EASY205-ASI network module connected to the EC4-200: 4 bits of output data and 4 parameter bits. It receives 4 bits of input data from the EC4-200.

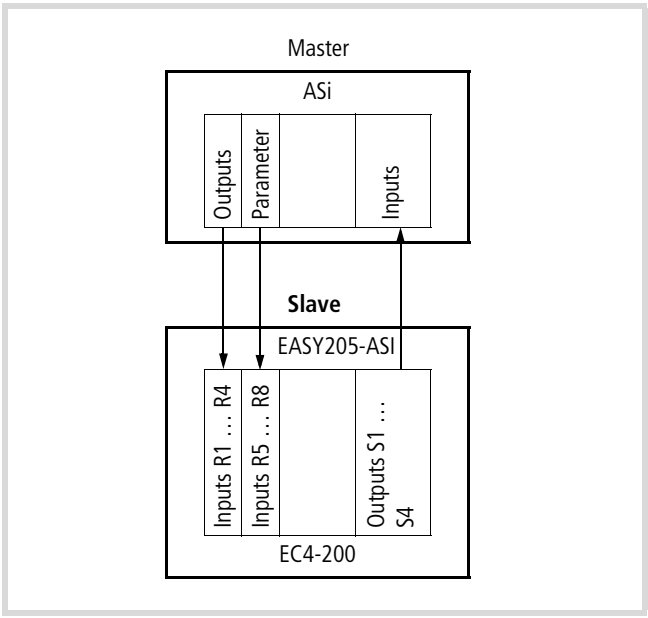



Figure: 117: Cyclic data exchange of the EASY205-ASI

Table 22: 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.

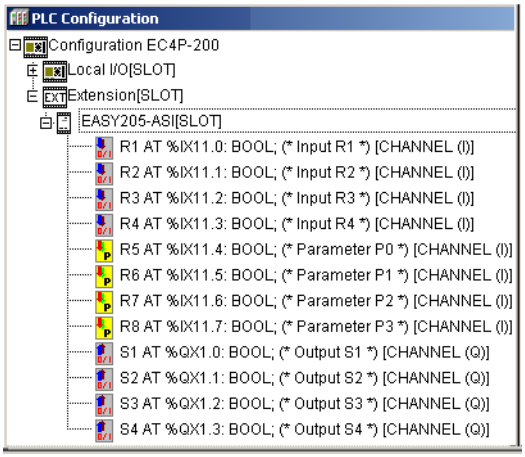



Figure: 118: Configuring EASY205-ASI

### 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 21.

### 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.

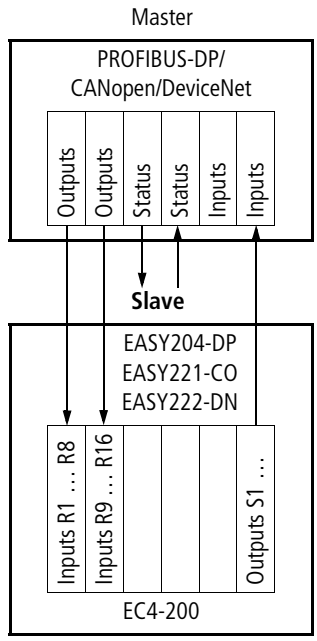



Figure: 119: Cyclical data exchange between master EASY204-DP, EASY221-CO, EASY222-DN

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 or STOP status, described in separate manuals, → table 21.

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 21.

### 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:s

```
EXTENSION_SLAVE_ADDRESS = <Address>
```

figure 120 is an example of where to enter the address for the EASY204-DP.

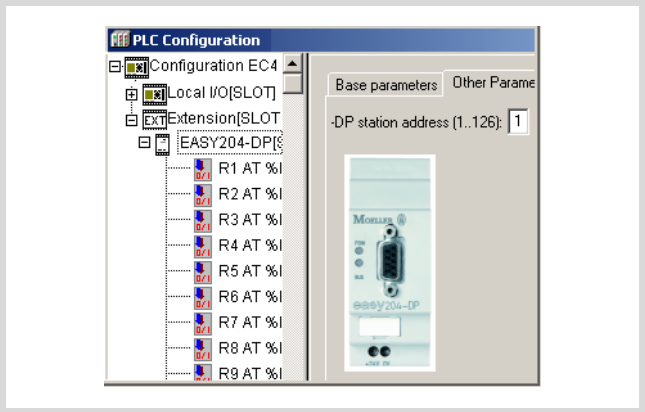



Figure: 120: Address entry

→ 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 23: 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 93
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 24.

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 24.

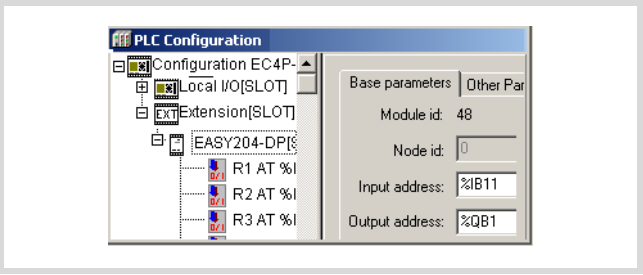



Figure: 121: Setting the address ranges

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

Table 24: 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

## CAN/easy-NET network

### Accessories

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

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

Table 25: Prefabricated cables

| Cable length<br>cm | Part no. |
|---|---|
| 30 | EASY-NT-30 |
| 80 | EASY-NT-80 |
| 150 | EASY-NT-150 |

- User-assembled cable, Type: EASY-NT-CAB (100 m 4 x 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 O

### 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<br>m | Cable resistance<br>mO/m | Cross-section<br>mm² | AWG |
|---|---|---|---|
| up to 40 | ≦ 140 | 0.13 | 26 |
| Up to 175 | ≦ 70 | 0.25 … 0.34 | 23, 22 |
| Up to 5 x 240 | ≦ 60 | 0.34 … 0.5 | 22, 21, 20 |
| Up to 400 | ≦ 40 | 0.5 … 0.6 | 20, 19 |
| Up to 600 | ≦ 26 | 0.75 … 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<br>Kbaud | Cable resistance RL<br>Ω |
|---|---|
| 10 … 125 | ≦ 30 |
| 250 | ≦ 25 |
| 500 | ≦ 12 |

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

$$lmax = \frac{RL}{R'}$$

### 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
Smin = minimum cable cross-section in mm2
rcu = specific resistance of copper if not stated otherwise 0.018 Omm2/m

$$Smin = \frac{l \times rcu}{12.4}$$

→ 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
$S$ = minimum cable cross-section in mm²
rcu = specific resistance of copper if not stated otherwise 0.018 Omm2/m

$$lmax = \frac{S \times 12.4}{rcu}$$

## 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".

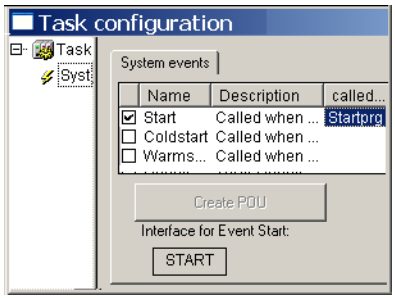



Figure: 122: Activating a system event

▶ 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.

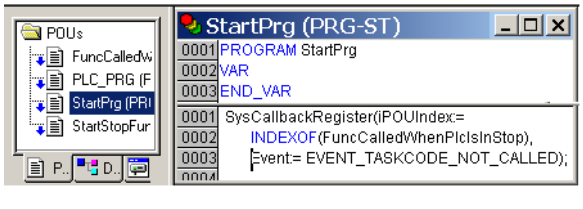



Figure: 123: "Startprg" function

▶ Declare the following global variables.

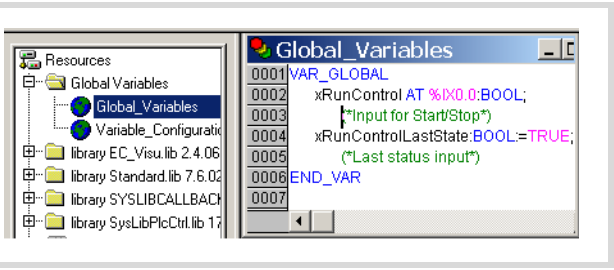



Figure: 124: Declaring global variables

▶ Enter the program for PLC_PRG as shown in figure 125. It is important that the user program and the POU calls are inserted as shown in figure 125.

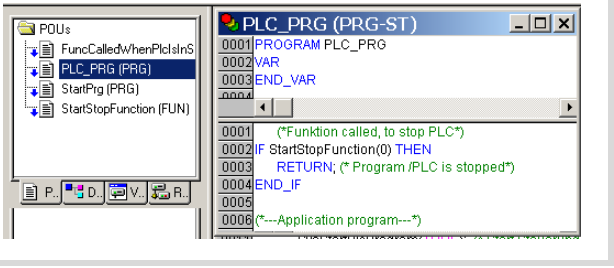



Figure: 125: Scanning START/STOP

▶ Enter the function FuncCalledWhenPlcInStop and StartStopFunction.

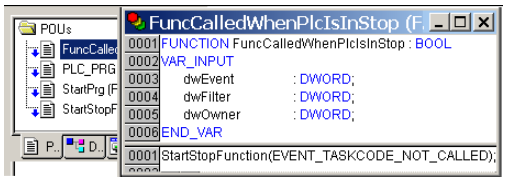



Figure: 126: Call of the function FuncCalledWhenPlcIsInStop

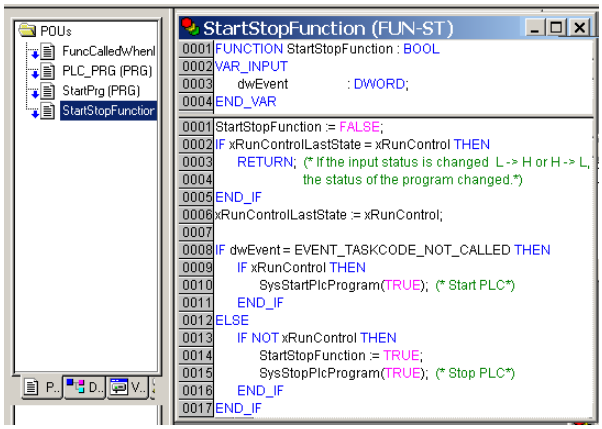



Figure: 127: Function that monitors the input

## 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 |

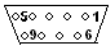



Figure: 128: 9-poliger Buchsen-Stecker

→ 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]<br>with adapter for MMC | 107.5 × 90 × 72<br>107.5 × 90 × 79 |
| [inches]<br>with adapter for MMC | 4.23 x 3.54 x 2.84<br>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 |

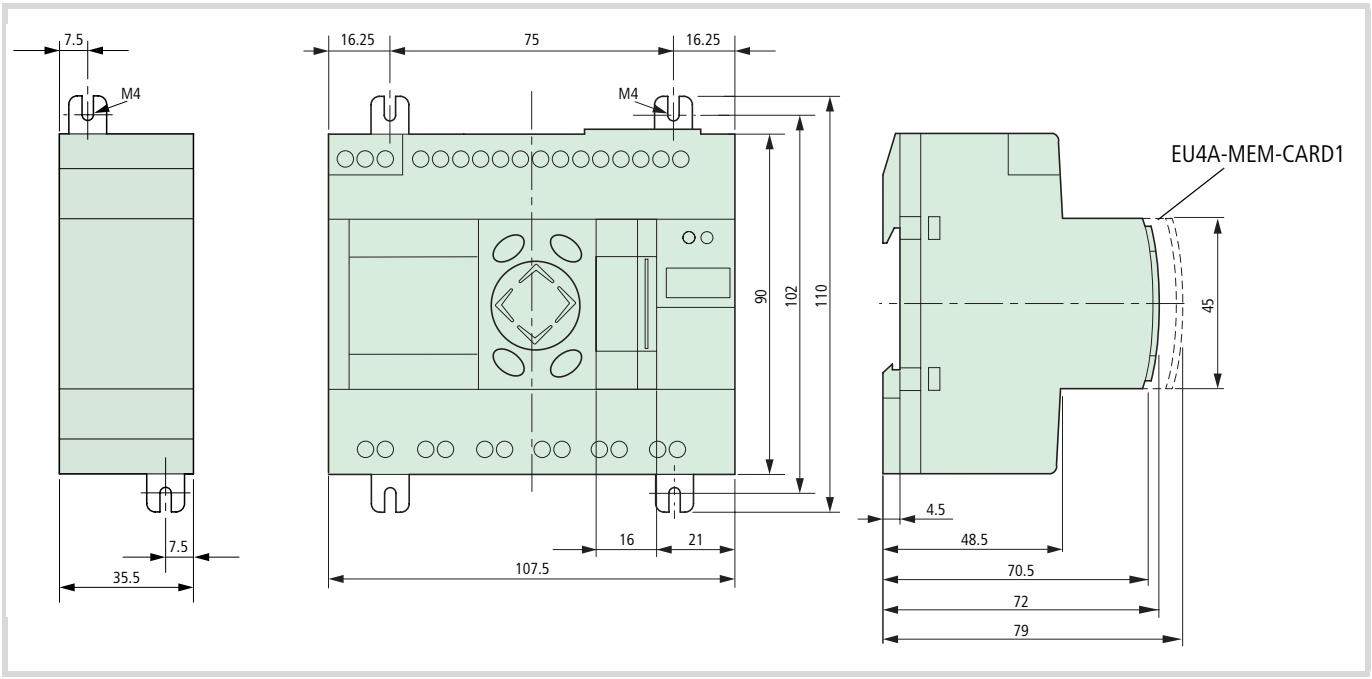


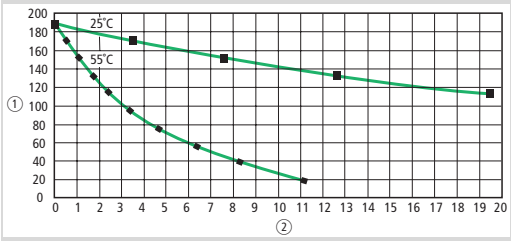

Figure: 129: Dimension in mm (specified in inches ), → table 26

Table 26: 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<br>Installed horizontally/vertically | | °C, (°F) | –25 to 55, (–13 to 131) |
| Condensation | | | Prevent condensation by means of suitable measures |
| LCD display (reliably legible) | | °C, (°F) | 25 … 55, (–13 … 131) |
| Storage/transport temperature | | °C, (°F) | –40 … 70, (–40 … 158) |
| Relative humidity (IEC 60068-2-30), non-condensing | | % | 5 to 95 |
| Atmospheric pressure (operation) | | hPa | –795 … 1080 |
| **Ambient mechanical conditions** | | | |
| Degree of protection (IEC/EN 60529) | | | IP20 |
| Vibrations (IEC/EN 60068-2-6) | | | |
| Constant amplitude 3.5 mm | | Hz | 5 … 9 |
| Constant acceleration 1 g | | Hz | 9 … 150 |
| Shock (IEC/EN 60068-2-27)<br>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),<br>(IEC/EN 61000-4-2, severity level 3) | | | |
| Air discharge | | kV | 8 |
| Contact discharge | | kV | 6 |
| Electromagnetic fields (RFI),<br>(IEC/EN 61000-4-3) | | V/m | 10 |
| Radio interference suppression<br>Limit value class | | | EN 55011, EN 55022<br>Class B |
| Fast transient burst (IEC/EN 61000-4-4, severity level 3) | | | |
| Supply cables | | kV | 2 |
| Signal cables | | kV | 2 |
| Surge<br>(IEC/EN 61000-4-5, degree of severity 2) | | kV | 0.5 symmetrical<br>1 asymmetrical |
| Line-conducted interference (IEC/EN 61000-4-6) | | V | 10 |
| **Dielectric strength** | | | |
| Clearance and creepage distances | | | EN 50178 |
| Insulation resistance | | | EN 50178 |
| Overvoltage category/pollution degree | | | II/2 |
| **Tools and cable cross-sections** | | | |
| Solid, minimum to maximum | | $mm^2$ | 0.2 to 4 |
| | | AWG | –22 … 12 |
| Flexible with ferrule, minimum to maximum | | $mm^2$ | 0.2 … 2.5 |
| | | AWG | –22 … 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** | | |
| Clock battery back-up | |  <br>1) 1 backup time in hours<br>2) 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 (RS232) without control lines | | |
| PLC port | | COM1 |
| Electrical isolation | | None |
| Connection types | | RJ45, 8-pole |
| **Programming mode** | | |
| Data transmission rate | | 4.8, 9.6, 19.2, 38.4, 57.6 |
| Character format | | 8 data bits, no parity, 1 Stop bit |
| **Transparent mode** | | |
| Data transmission 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 |
| **Multi-function interface (RS232)** without control cables | | |
| **Transparent mode** | | |
| PLC port | | COM2 |
| Electrical isolation | | Yes, in the easy800-PC-CAB cable |
| Connection types | | easy800-PC-CAB cable |
| Data transmission rate | Kbit/s | 9.6, 19.2 |

| | | |
|---|---|---|
| **CAN(open)/easy-NET** | | |
| Data transmission rate | Kbit/s | 10, 20, 50, 100, 125, 250, 500<br>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) |
| Connection type | | 2 x RJ45, 8pole |
| CAN(open) operating mode: | | |
| – Stations | Number | max. 126 |
| – PDO type | | Asynchronous, cyclic, acyclic |
| – Device profile | | to DS301V4 |
| easy-NET operating mode | | |
| – Station | Number | max. 8 |

| | | |
|---|---|---|
| **Power supply** | | |
| Rated voltage | | |
| Rated value | V DC, (%) | 24, ( –15, +20) |
| Permissible range | V DC | 0.25 to 0.34 |
| Residual ripple | % | ≦ 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 for pulse signals<br>(High-speed counters) | | I 1,2,3,4 |
| Inputs for interrupt generation | | I 1,2,3,4 |
| Status indication | | LCD display |
| Electrical isolation | | |
| From the power supply, PC interface | | No |
| Between each other | | No |
| From the outputs, to CAN interfaces | | Yes |
| Rated voltage | | |
| Rated value | V DC | 24 |
| On 0 signal | | |
| I1 to I6 and I9 to I10 | V DC | < 5 |
| I7, I8, I11, I12 | V DC | < 8 |
| On 0 signal | | |
| 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 … I4 | ms | 0.02 |
| I1 to I4 | ms | 0.25 |
| Delay time from 1 to 0 | | |
| I1 … I4 | ms | 0.02 |
| I1 to I4 | 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 … 10 |
| Resolution analog | V | 0.01 |
| Resolution digital | Bit | 10 |
| | Value | 0 … 1023 |
| Input impedance | kO | 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 x 16 bits (I1, I2)<br>1 x 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 | | 1:1 |
| **Incremental counter** | | I7, I8, I11, I12 |
| Number | | 1 |
| Value range | Bit | 32 Bit |
| Max. frequency | kHz | 40 |
| Cable length (shielded) | m | 20 |
| Pulse shape | | Square |
| Counter inputs | | I1,I2 = Counter input<br>I3 = Reference pulse<br>I4 = Reference window |
| Signal offset | | 90° |
| Mark-to-space ratio | | 01:01 |
| **Inputs for interrupt generation** | | I7, I8, I11, I12 |
| Max. frequency | kHz | 3 |

| **Relay outputs** | | |
|---|---|---|
| Number of outputs | | 6 |
| Connection of outputs in parallel to increase the output | | 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 x 106 |
| Contacts relays | | |
| Conventional therm. current | A | 8 |
| Recommended for load at 12 V AC/DC | mA | > 500 |
| Protected against short-circuit cos v = 1<br>Characteristic B (B16) at 600 A | A | 16 |
| Protected against short-circuit cos v = 0.5 to 0.7<br>Characteristic B (B16) at 900 A | A | 16 |
| Rated impulse withstand voltage Uimp contact coil | kV | 6 |
| Rated insulation voltage Ui | | |
| 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) | Switch operations | 300000 |
| DC-13 L/R F 150 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 F 150 ms 24 V DC, 1 A (500 Ops/h) | Switch operations | 200000 |
| Filament lamp 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<br>- with ballast<br>- with conventional compensation<br>- uncompensated | Switch operations | 25000 |

| | | |
|---|---|---|
| Relay switching frequency | | |
| Mechanical switch operations | Switch operations | 10 mill. ($10^7$) |
| 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 | 0.25 to 0.34 |
| Residual ripple | % | ≦ 5 |
| Supply current | | |
| On "0" signal, typical/maximum | mA | 18/32 |
| On 1 state, typical/maximum | mA | 24/44 |
| Reverse polarity protection<br>▽ **Attention!**<br>Connecting the outputs to a power supply with a reverse polarity will result in a short-circuit. | | Yes |
| Electrical isolation | | Yes |
| Rated current $I_e$ on 1 state, maximum | A | 0.5 |
| Lamp load without RV | W | 5 |
| Residual current on 0 state per channel | mA | < 0.1 |
| Max. output voltage | | |
| On 0 state with external load, 10 MO | V | 2.5 |
| On 1 state, $I_e$ = 0.5 A | | U = Ue – 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)<br>▽ **Attention!**<br>Set the output group in the program to a 0 signal in order to prevent the output from overloading | | Yes |
| Short-circuit tripping current for Kurzschlussauslösestrom für $R_a$ ≦ 10 mΩ (depending on number of active channels and their load) | A | 0.7 ≦ $I_e$ ≦ 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) | Switching operations/ h | 40000 |

| | | |
|---|---|---|
| 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 … Q4 | | |
| Group 2: Q5 … Q8 | | |
| Maximum number of outputs | | 4 |
| Total maximum current<br><br>**Attention!**<br>Outputs connected in parallel must be switched at the same time and for the same duration. | A | 2 |
| 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.

$$T0.95\ Q\ 3 \times T0.65 = 3 \times \frac{L}{R}$$

Utilisation categories in groups Q1 to Q4, Q5 to Q8

| | | | |
|---|---|---|---|
| T0.95 = 1 ms<br>R = 48 O<br>L = 16 mH | Utilisation factor per group g = | | 0.25 |
| | Relative duty factor | % | 100 |
| | Max. switching frequency<br>f = 0.5 Hz<br>Max. duty factor<br>DF = 50 % | Switching operation s/h | 1500 |
| DC13<br>$T_{0.95}$ = 72 ms<br>R = 48 O<br>L = 1.15 H | Simultaneity factor g | | 0.25 |
| | Relative duty factor | % | 100 |
| | Max. switching frequency<br>f = 0.5 Hz<br>Max. duty factor<br>DF = 50 % | Switching operation s/h | 1500 |

Other inductive loads:

| | | | |
|---|---|---|---|
| $T_{0.95}$ = 15 ms<br>R = 48 O<br>$L$ = 0.24 H | Simultaneity factor g | | 0.25 |
| | Relative duty factor | % | 100 |
| | Max. switching frequency<br>f = 0.5 Hz<br>Max. duty factor<br>DF = 50 % | Switching operation s/h | 1500 |
| 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<br>Max. duty factor | Switching operation s/h | Depending on the suppressor circuit |

| Analog output | | |
|---|---|---|
| Number | | 1 |
| Electrical 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 … 10 |
| Output current max. | mA | 10 |
| Load resistor | kO | 1 |
| Short-circuit and overload proof | | Yes |
| Resolution analog | V | 0.01 |
| Resolution digital | Bit | 10 |
| | Value | 0 … 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 |

# Index