

**easyHMI**  
**Multi-Function Display**  
MFD4

**Hardware, Engineering and  
Functional Description**

06/07 AWB2726-1594GB

**MOELLER** 

We keep power under control.

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

1<sup>st</sup> edition 2007, edition date 03/07

2<sup>nd</sup> edition 06/07

See revision protocol in the "About this manual" chapter

© Moeller GmbH, 53105 Bonn

Author: Peter Roersch

Production: Thomas Kracht

Translation: Terence Osborn

All rights reserved, including those of the translation.

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

Subject to alteration without notice.



## Warning! Dangerous electrical voltage!

---

### Before commencing the installation

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



## Contents

<b>About this manual</b>		5
	List of revisions	5
	Reading conventions	5
	Advanced documentation	5
	Designations	5
<b>1 Setup of the MFD4</b>		7
	– Performance scope	7
	– Real-time clock	7
	– Battery	7
	– Memory card (MCC)	8
	– CPU drives	8
	– Programming interfaces	9
	– CANopen/easyNet interface	9
<b>2 Mounting the MFD4</b>		11
	– Mounting/removing the device	11
<b>3 Engineering</b>		13
	Control panel layout	13
	– Ventilation	13
	Preventing interference	13
	– Suppressor circuitry for interference sources	13
	– Shielding	13
	Lighting protection	14
	– Earthing the device	14
	Power supply	14
	Interfaces	14
	– Ethernet	14
	– RS 232	15
	– CANopen/easyNet	15
<b>4 MFD4 operation</b>		17
	Setting parameters	17
	– Executing a reset	17
	– Setting parameters on the display	17
	– Starting MFD4	19
	Changing from application screen ↔ basic menu	19
	– Changing to the basic menu	19
	– Changing to the application screen	19

<b>5 Operation</b>	<ul style="list-style-type: none"> <li>Switching on MDF4 21</li> <li>– Operating mode switch 22</li> <li>– SET button 22</li> <li>– Startup behaviour after loading an operating system or reset 22</li> <li>Startup behaviour of the program 23</li> <li>– Setting the startup behaviour in the programming software 24</li> <li>– Program start (STOP → RUN) 24</li> <li>– Program stop (RUN → STOP) 24</li> <li>Power off/interruption of the power supply 24</li> <li>Test and commissioning (Debugging) 25</li> <li>– Breakpoint/single-step mode 25</li> <li>– Single-cycle mode 25</li> <li>– Forcing 25</li> <li>– Status indication 25</li> <li>Reset 25</li> <li>Programs and project 26</li> <li>– Loading the program 26</li> <li>– Creating a boot project 26</li> <li>– Storing the boot project on a memory card 26</li> <li>– Deleting a boot project on a memory card 26</li> <li>Updating the operating system 27</li> <li>– Transferring the operating system from the PC to the device 27</li> <li>– Transferring the operating system from the PC into the MMC 28</li> <li>– Erase operating system/boot project from the MMC 28</li> </ul>
<b>6 Program processing, multitasking and system times</b>	<ul style="list-style-type: none"> <li>Task configuration 29</li> <li>– Standard task of the visualisation 29</li> <li>Creating a task (example) 30</li> <li>– Creating the “PLC_PRG” program 30</li> <li>– Creating the event-triggered task “Param” and defining the program call 31</li> <li>System events 32</li> <li>Multitasking 32</li> <li>– Update CANopen variables with multitasking 32</li> <li>Task monitoring with the watchdog 33</li> <li>– Watchdog configuration 33</li> <li>– Multiple tasks with the same priority 34</li> <li>Web visualization 34</li> <li>Limit values for memory usage 35</li> <li>– Address range 35</li> <li>Addressing inputs/outputs and markers 35</li> <li>– “Automatic calculation of addresses” 35</li> <li>– Check for overlapping addresses 35</li> <li>– Free assignment or modification of addresses of input/output modules and diagnostic addresses 36</li> <li>– Run “Automatic calculation of addresses” 36</li> <li>– Uneven word addresses 36</li> </ul>

<b>7</b>	<b>Establishing a PC – MFD4 connection</b>	37
	Connection set-up via RS 232 interface	37
	– Setting the PC communication parameters	37
	– Setting the MFD4 communication parameters	38
	Connection set-up via Ethernet	38
	– Selecting communication channel and address	38
	– Scan/Modify the IP address	39
<b>8</b>	<b>Defining the system parameters via the STARTUP.INI file</b>	41
	Overview	41
	Structure of the INI file	41
	Creating the Startup.INI file	41
	Entry of the INI file: "HOST_NAME"	42
	Switching on the controller with the fitted memory card containing the Startup.INI file	42
	Changing settings	42
	Deleting the Startup.INI file	42
<b>9</b>	<b>Programming via the CANopen network (routing)</b>	43
	Requirements	43
	Routing properties of the MFD4	43
	– Optimise TCP/IP data transfer	43
	– Setting the size of the data blocks	43
	Notes	44
	Setting the node ID/routing ID	44
	Setting the master station	45
	Setting device (target) station	45
	Station combinations for routing	46
	Number of communication channels	46
<b>10</b>	<b>RS 232 interface in Transparent mode</b>	47
<b>11</b>	<b>Libraries, function blocks and functions</b>	49
	Using libraries	49
	Installing additional system libraries	49
	MFD specific functions	50
	– Display.lib	50
	– MFD57_Util.	50
	CAN utilities	50
	Ethernet utilities	51
	– UT12_GetIPConfig	
	Get IP, subnet mask and IPGateway address	51
	– UT12_GetIPDNS	
	Display IP address of the dynamic name server set on the MFD4	51
	– UT12_GetIPWINS	
	Display IP address of the Windows name server set on the MFD4	52
	– UT12_GetMacAddress	
	Get MAC address (MAC=Media Access Control)	52
	– UT12_SetIPConfig	
	Setting the IP- and subnet mask address	53
	– UT12_SetIPDNS	
	Change the IP address of the dynamic name server on the MFD4	53

	– UTI2_SetIPGateway Setting of the IP Gateway address	54
	– UTI2_SetIPWINS Change the IP address of the Windows name server on the MFD4	54
	General functions	55
	– UTI2_Reboot Restart with registry storage	55
	– UTI2_SaveRegistry Saving of the registry	55
<b>12 Browser commands</b>		57
	Communication parameter access	57
	Calling up browser commands	57
	Command overview	57
	Important browser commands	60
	– Display CPU loading (plload)	60
	– Display the loading of the CAN bus (canload)	60
	– Access to memory objects	60
	Error and event list after calling browser commands	61
<b>13 Network easyNet</b>		63
	Overview	63
	Monitoring easyNet	63
	Sending/receiving user data	64
	– NET_UPDATE function	64
	– NET_GET function	64
	Data transfer	65
	– Data transfer between a station with ID1 and remote I/O device	65
	– Transfer of bit data blocks between several stations	66
	– Transfer of D words (32-bit) according to the PUT-GET principle between several PLCs	67
	Configuring with easy800 on the easyNet	68
	– Configuration in easySoft	68
	– Configuration in easySoft-CoDeSys	68
	– Controlled configuration via MFD4	69
	Bus topology	70
<b>14 Programming via easyNet (Routing)</b>		71
	Routing via MFD4	71
<b>Appendix</b>		73
	Characteristic of the Ethernet cable	73
	Properties of the CANopen cable	73
	Transparent mode: Text output via RS232 (example)	74
	Access to the CPU drives/memory card	75
	– "SysLibFile.lib" library	75
	– Mode for opening a file	75
	– Examples of the "SysFile..." functions	75
	Dimensions	76
	Technical data	77
<b>Index</b>		81



## About this manual


### List of revisions


The following significant amendments have been introduced since previous issues:


Edition date	Page	Keyword	New	Modifi- cation	Omitted
06/07	14	Earthing the device		✓	
	17	User interface for display calibration, note	✓		
	46	figure 64	✓		
	68	Programming software (2nd listing)		✓	
	69	Warning			✓
	79	Approvals and declarations	✓		

### Reading conventions

→ Draws your attention to interesting tips and supplementary information.

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

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

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

► Indicates actions to be taken.

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

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

### Advanced documentation

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

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

Programs → Moeller Software → easy Soft CoDeSys →  
Dokumentation → Automation Manuals.

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.

This always provides the latest data.

### Designations

The MFD4-5-XRC-30 multi-function display described in this document is called the MFD4 in the following. The software sometimes also refers to the MFD 57 or MFD 5.7. The “4” stands for the performance class of the device, the “57” or “5.7” for the “5.7” display.

Regardless of whether MFD4, MFD57 or MFD5.7 is mentioned, they all refer to the same device.



# 1 Setup of the MFD4

The MFD4 multi-function display consists of a touch screen and an integrated compact PLC. It is designed for controlling, operating and monitoring machines and plants, as well as being provided with several interfaces:

- Ethernet
  - for connecting a programming device
  - for communication with other devices
- RS232
  - for connecting a programming device
  - for data transfer in Transparent mode
  - for transferring the operating system
- CANopen/easyNet
  - for the remote connection of input/output devices
  - for communication with other devices.

The PLC program and the graphic screens are created with the easySoft-CoDeSys programming software

## Performance scope

The number of configured screens, messages, languages etc. is limited by the 6 MByte memory available. The PLC program has priority in the system over the visualisation. The program must therefore not require all the CPU processing time otherwise the operation of the visualisation will be impossible or very slow.

When the files are being transferred (e.g. project download, FTP, WEB server) or when communication with the development system is active (such as for PLC debugging) delays may occur in the visualisation as the visualisation is processed with low priority.

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

Program code	2048 KByte
Program data, of which:	512 KByte
Markers	16 KByte
Retain data	32 KByte
Persistent data	32 KByte

The MFD4 features:

- Real-time clock → page 7
- Battery → page 7
- Memory card (MCC) → page 8
- CPU drives → page 8
- RS 232 interface → page 9
- Ethernet interface → page 9
- CANopen/easyNet interface → page 9
- Operating mode switch → page 22

## Real-time clock

The MFD4 features a real-time clock, which can be referenced in the user program via the functions from the "SysLibRTC" library. The following functions are possible:

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

A description of the functions can be found in the "SysLibRTC.pdf" file. Further information on this is provided on page 5.

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

- setrtc (set the real-time clock) → page 57
- getrtc (query the real-time clock) → page 57.

## Battery

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

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



### Attention!

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

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

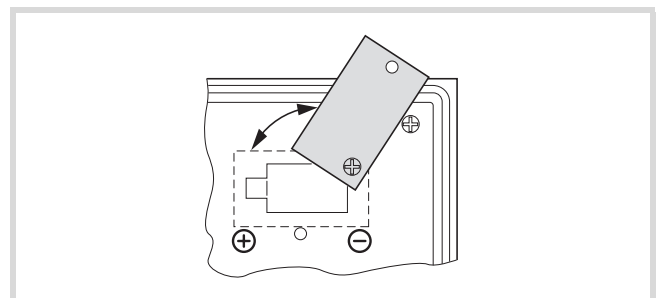


Figure 1: Battery change

## Memory card (MCC)

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

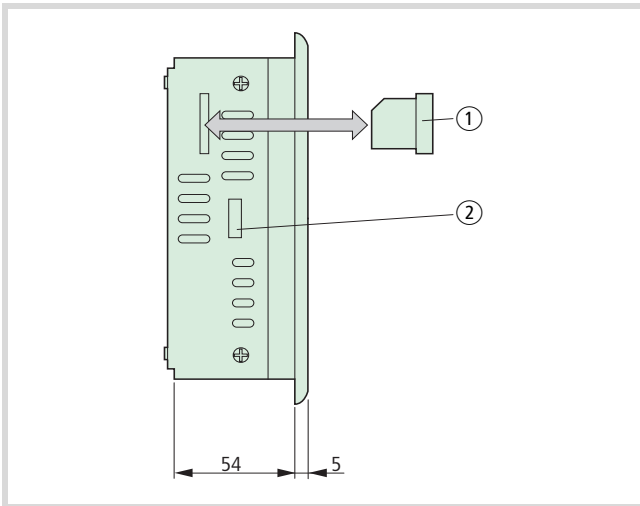


Figure 2: Inserting the memory card

- ① Memory card
- ② Tab for fixing bracket

You can transfer the operating system (OS) to the memory card in order to load it from there into other MFD4 devices (OS update).



### Attention!

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

See also:

- Updating the operating system → page 27
- Erase operating system/boot project from the MMC → page 28

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

## CPU drives

The MFD4 has the following drives available:

- Internal
  - Memory system (disk\_sys)
- external, optional
  - Memory card (disk\_mmc)

The boot system and the operating system are saved in compressed format and in the transaction safe and non-volatile system memory. In the operating state, the boot project and the relevant sections of the operating system are “unpacked” and copied into the working memory. The retentive data is stored in the battery-backed SRAM memory.

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

The figure 3 indicates the interaction of the differing memory systems:

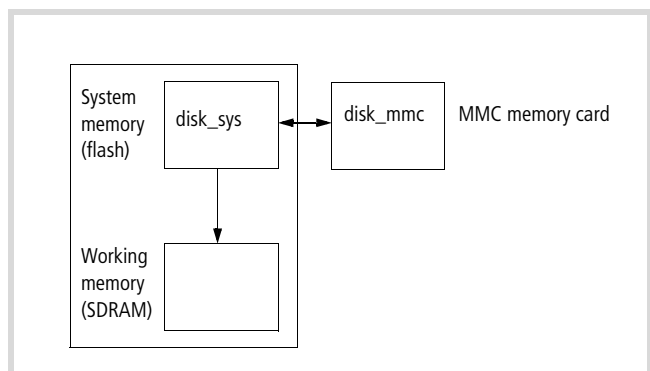


Figure 3: CPU memory organization

See also:

- Data access to the memory card with the aid of
  - browser commands such as, for example, copyprojtommc, to copy the user program onto the MMC → page 57
  - Functions such as “SysFileOpen” or “SysFileRead” → page 75
- Limit values for memory usage → page 35

## Programming interfaces

Two programming interfaces are provided

- Ethernet interface
- RS 232 interface

The operating system processes the Ethernet interface faster than the RS232. Program transfers or functions for troubleshooting should therefore be carried out via the Ethernet interface. The interfaces have the following additional functions.

### Ethernet interface

The galvanically isolated interface is used to establish a connection to the Ethernet network and exchange data with other PLCs/devices on the network.

See also:

- Interfaces → page 14
- Assignment of the Ethernet interface → page 15
- Establishing a PC – MFD4 connection → page 37.

### RS 232 interface

This interface allows you to establish a point-to-point connection to another device without handshake cables. To do this switch the interface to Transparent mode by calling the functions from the SysLibCom.lib library. In this status, the interface is addressed as COM1. The interface has no potential isolation.

See also:

- Interfaces → page 14
- Assignment of the RS 232 interface → page 15
- Establishing a PC – MFD4 connection → page 37.

### CANopen/easyNet interface

You connect the MFD4 to the CANopen or the easyNet network via the 9-pole SUB-D connector. Both networks use the same data cables; the control cables SELECT-IN, SELECT-OUT are also provided for the easyNet. The control cables enable the MFD4 to configure the other stations on the easyNet or be configured itself. The interface is potentially isolated.

The MFD4 can be run as network (NMT) master or as NMT slave (Device) on the CANopen bus. CANopen and easyNet stations can be connected to the MFD4 via the same bus.



If an MFD4 is incorporated in the easyNet network, the baud rate of the stations must not exceed 125 Kbit/s!

See also:

- easy800 control relay user guide (AWB2528-1423)
- Network easyNet → page 73

### Behaviour of the stations on the CANopen bus

Node/bus monitoring: CANopen telegrams are sent and received directly by the user program. An interruption on the CANopen bus is only detected if the appropriate CANopen nodes are being monitored by another station (Node guarding function).

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

Voltage switch on: The sequence in which the power supply of the individual CAN slaves is connected does not have an effect on the functionality of the CAN bus. Depending on the parameters set, the PLC "waits" for non-existent slaves or starts them when they are connected to the CAN network.

Communication with CANopen stations: The communication with the CANopen stations and their configuration is described in the following online documentation that was installed during the installation on your computer:

- Engineering CANopen stations (AN2700K27G.PDF)<sup>1)</sup>
- Coupling of multiple autonomous controls (CAN-Device) via CANopen (AN2700K20G.PDF)<sup>2)</sup>
- Library description: CANUser.lib/CANUser\_Master.lib (h1554g.pdf).<sup>3)</sup>

- 1) The documentation is located in the directory C:\Programs\Moeller Software\easy Soft CoDeSys V2.3.5\easy Soft CoDeSys\APPLICATION-EXAMPLES-NOTES-MODULES\XCONTROL\CAN\CAN\_XC100\_XC200\ENGLISH
- 2) The documentation is located in the directory C:\Programs\Moeller Software\easy Soft CoDeSys V2.3.5\easy Soft CoDeSys\APPLICATION-EXAMPLES-NOTES-MODULES\XCONTROL\CAN\_DEVICE\CAN\_XC100\_XC200\ENGLISH
- 3) → page 5

See also:

- CANopen/easyNet → page 15
- Properties of the CANopen cable → page 73



## 2 Mounting the MFD4

### Mounting/removing the device

The device is suitable for vertical or diagonal (up to 45 °) mounting in a control cabinet, a front panel or control desk.

The device is fastened from the rear:

- ▶ Fit the seal flat and evenly between the front panel and the front plate
- ▶ Tighten the fixing brackets for fastening the device evenly until the front plate is flat against the front panel.

The device can be mounted in a housing provided that the permissible ambient temperature is observed, → section „Technical data“.

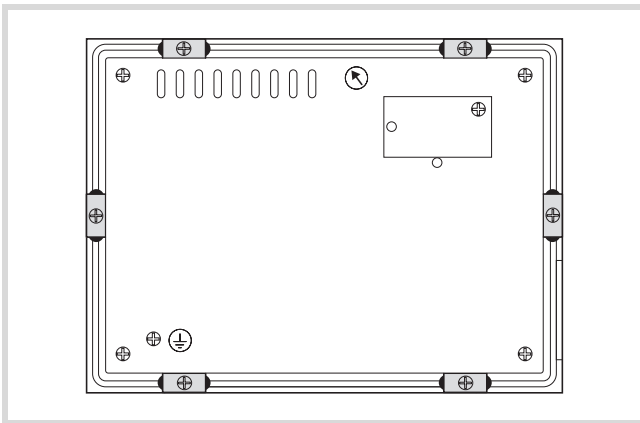


Figure 4: Arrangement of fixing brackets

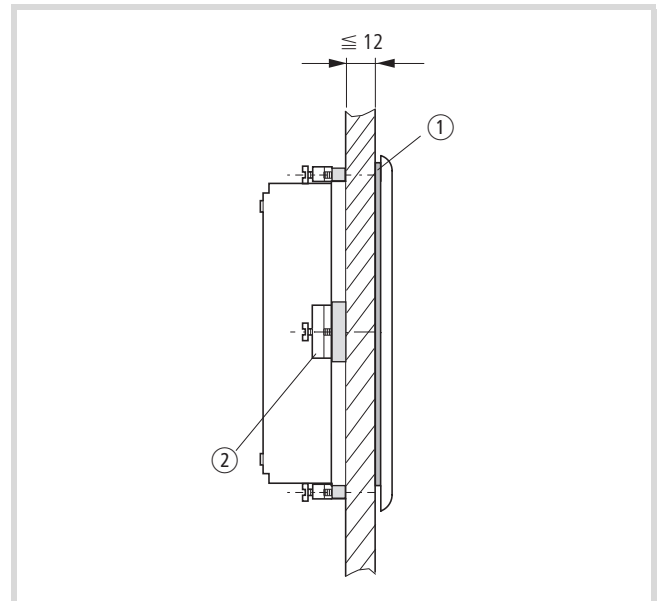


Figure 5: Mounting with the fixing brackets

- ① Seal (tightly fitted on the device)
- ② Fixing bracket

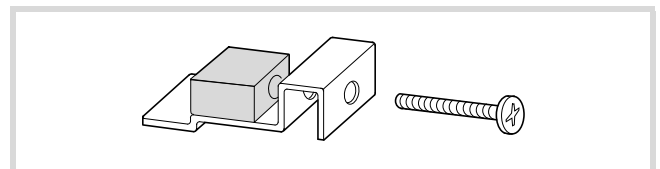


Figure 6: Fixing brackets

Mounting dimensions → page 76





## 3 Engineering

### Control panel layout

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

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

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

### Ventilation

In order to ensure sufficient ventilation a minimum clearance of 50 mm to passive components must be observed. If the adjacent components are active elements (e.g. power supplies, transformers) a minimum clearance of 75 mm must be observed. The values specified in the technical data must be observed. Ensure that the device does not overheat during operation:

- ▶ Keep the cooling slots clear to ensure the cooling of the system.
- ▶ Avoid direct sunlight on the front.
- ▶ When mounting the device vertically ensure that the angle of tilt does not exceed 45 degrees.

### Preventing interference

#### Cable routing and wiring

Cables are divided into the following categories:

- Heavy current cables (e.g. carrying large currents or cables to current converters, contactors, solenoid valves)
- Control and signal cables (e.g. digital input cables)
- Measuring and signal cables (e.g. fieldbus cables)

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

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

- ▶ Avoid parallel routing of sections of cable in different power categories.
- ▶ As a basis rule, keep AC cable separated from DC cables.
- ▶ Keep to the following minimum spacing:
  - at least 10 cm between power cables and signal cables;
  - at least 30 cm between power cables and data or analog cables.
  - When routing cables, make sure that the outgoing and return leads of a circuit pair are routed together. The opposing currents on this cable pair cause the sum of all currents to equal zero. The generated electromagnetic fields therefore cancel each other out.

#### Suppressor circuitry for interference sources

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

→ Switched inductors should always have suppressor circuitry fitted.

#### Shielding

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

**Lighting protection**

**External lightning protection**

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

**Internal lightning protection**

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

- Equipotential bonding/earthing
- Shielding
- Using overvoltage protection devices

Refer to the “EMC engineering guidelines for PS4/PS416 automation systems” (h1287g.pdf) for any questions on cabling and shielding measures.

**Earthing the device**

The device housing must always be earthed. The earth connection (PE) is located on the rear of the device. It must be marked with a sticker showing the PE earth symbol.

**Power supply**

Connect the device to the 24 VDC supply. This is fuse protected. To exchange the fuse the device must be sent to the manufacturer. There is no potential isolation on the device. Protection against reverse polarity protects the device from incorrect connection.

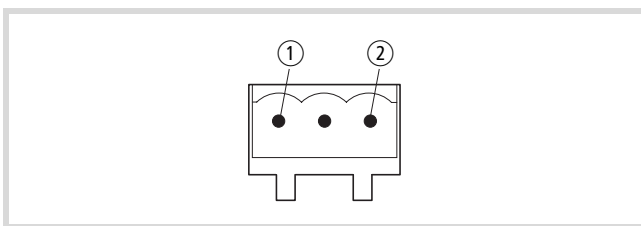


Figure 7: Plug connector to the power supply

- ① 0 V
- ② 24 V DC

**Interfaces**

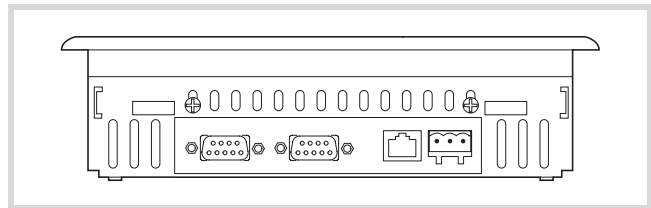


Figure 8: Interfaces (from left): CANopen/easyNet, RS232, Ethernet, 24 V DC

**Ethernet**

The programming device interface is designed physically as an RJ45 interface (socket). This means that normal commercial RJ45 connectors or Ethernet patch cables can be used.

**Direct connection PC – MFD4**

The MFD4 can be connected directly to the (programming) PC via a crossover Ethernet cable, → figure 9, 10.

Crossover cables have the following design features:

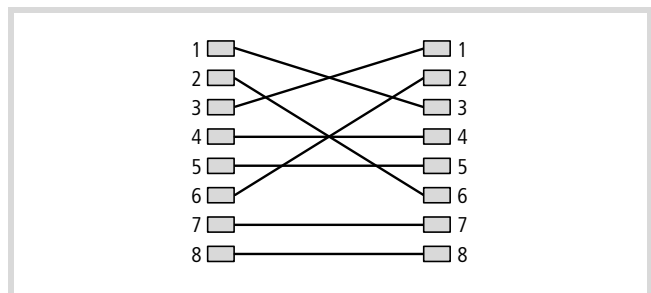


Figure 9: Connection set-up of a 8-pole crossover cable

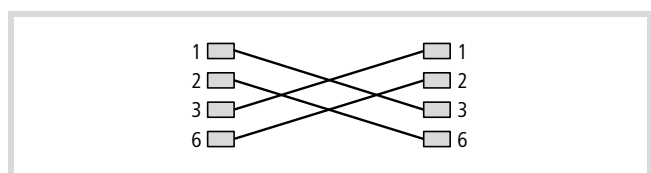


Figure 10: Connection set-up of a 4-pole crossover cable

The following cross-over cables are available:

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

**PC – MFD4 via Hub/Switch connection:**

If you use a Hub or a Switch between the PC – MFD4 connection, you must use a standard Ethernet cable which is connected 1:1 for the connection between PC – Hub/Switch and Hub/Switch – MFD4.

See also:

- Characteristic of the Ethernet cable → page 73

Table 1: Assignment of the Ethernet interface

RJ 45	PIN	Signal
	8	–
	7	–
	6	Rx–
	5	–
	4	–
	3	–
	2	Rx+
	1	Tx–
	2	Tx–
	1	Tx+

**RS 232**

Table 2: Assignment of the RS 232 interface

RS 232	PIN	Signal
	9	–
	8	CTS <sup>1)</sup>
	7	RTS <sup>1)</sup>
	6	DSR <sup>1)</sup>
	5	GND
	4	DTR <sup>1)</sup>
	3	TxD
	2	RxD
	1	DCD <sup>1)</sup>

1) The control cables are routed out but are not active.



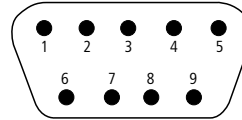
**Attention!**

- Do not connect a potential to the control cables.
- If the interface cable is longer than 4 m, connect the 24 V DC power supply via the DEHnrail DR 24 FML filter (filter manufacturer: Dehn).

**CANopen/easyNet**

Table 3: Assignment of the CANopen/easyNet interface

Pin	Signal	
	CANopen	easyNet
9	–	–
8	–	Select_Out
7	CAN_H	ECAN_H
6	GND	GND
5	–	–
4	–	Select_In
3	GND	GND
2	CAN_L	ECAN_L
1	–	–



**Bus terminating resistors**

The first or last station on the bus must be provided with 120 Ohm bus terminating resistors between the CAN\_H and CAN\_L cables.

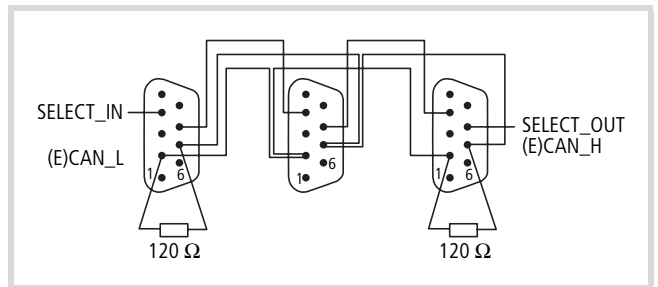


Figure 11: Cabling between 3 MFD4



## 4 MFD4 operation

### Setting parameters

The display can be used to set the following parameters and execute the following functions:

- Executing a reset
- Set parameters on the display
  - User interface for display calibration (Touch tab)
  - Entering parameters of the Ethernet interface (TCP/IP tab)
  - Setting the time (Clock tab)
  - Contrast and brightness setting (Contrast/Light tab)
- Starting MFD4.

### Executing a reset

In the Setup menu tap the Reset button (→ figure 22) in order to execute a warm Reset.

### Setting parameters on the display

Tap the Settings button to switch the display to the basic menu as shown in figure 12.

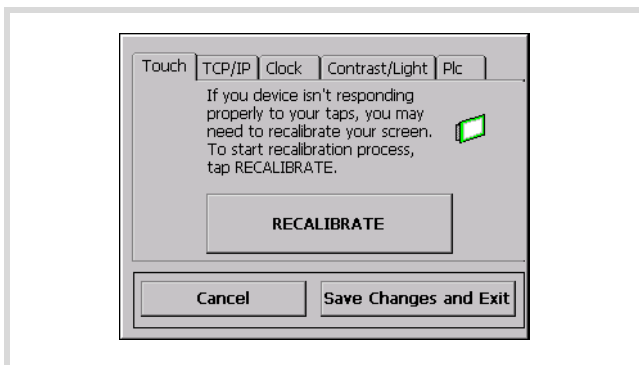


Figure 12: Display of the basic menu

You can now calibrate the MFD4 or select other tabs. In some tabs you can enter values and addresses via screens and using arrow keys. The longer you press the button, the faster the value/address changes. Save the new settings by pressing OK or Save Changes and Exit.

### User interface for display calibration

The system will initiate the calibration process automatically after loading a new operating system into the MFD4 or carrying out a reset to restore the factory settings.

If the display does not respond to touch, you should recalibrate it again. To do this move to the basic menu and tap the RECALIBRATE button on the Touch tab.

→ Calibrate the panel at the ambient temperature at which you wish to use the device.

- ▶ Follow the instructions that appear in English on the cross-hair at the centre of the display.

→ The ESC button described in the text for aborting the function is not provided. To cancel the operation switch the device off and on again.

- ▶ Tap the centre of the cross-hair. This will then move to the top left corner of the display. Touching the centre of the cross-hair again will cause it to move along the edge of the display to the top right corner. At this point two different sets of operations are possible.

The calibration was carried out correctly. In this case the cross-hair will disappear and an English text will appear. Ignore this text. The ESC and Enter buttons mentioned in this text are not provided on the MFD4.

- ▶ To return to the menu tap the display area.

Calibration was not completed correctly. In this case the cross-hair returns to the centre of the display. This occurs if the centre of the cross-hair was not always touched during calibration. The calibration procedure is restarted. You can repeat or cancel the process. To cancel the process switch the device off and on again.

### Entering parameters of the Ethernet interface

- ▶ Touch the TCP/IP tab.

This shows the IP address, subnet mask and gateway address.

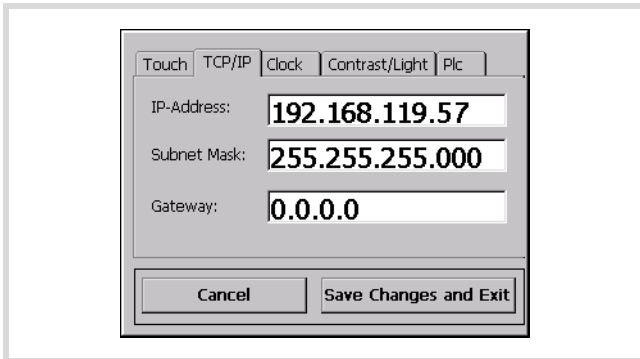


Figure 13: Entering Ethernet interface parameters

- ▶ To change address, tap the Address field. The assigned setting screen appears.
- ▶ Change the address via the arrow buttons and confirm it with OK.

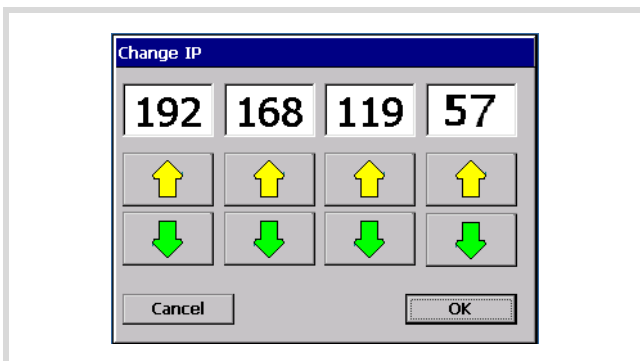


Figure 14: Screen for setting the IP address

- ▶ When you have changed all addresses, tap the Save Changes and Exit button. The display will change:

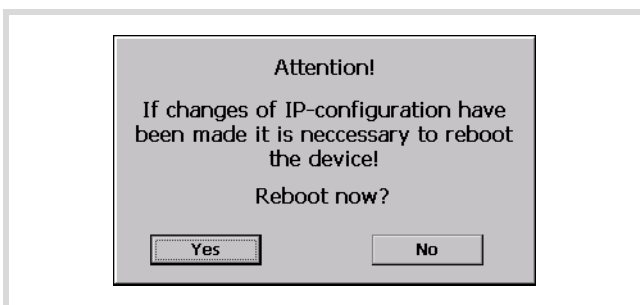


Figure 15: Reboot prompt

- ▶ Tap Yes in order to carry out a reboot. The modified addresses are not accepted until after a reboot has been completed. During a reboot, the modified data is stored in the registry and a software reset is carried out.

### Setting the time

- ▶ Tap the Clock tab.

This displays the current time that can be changed using the arrow buttons.

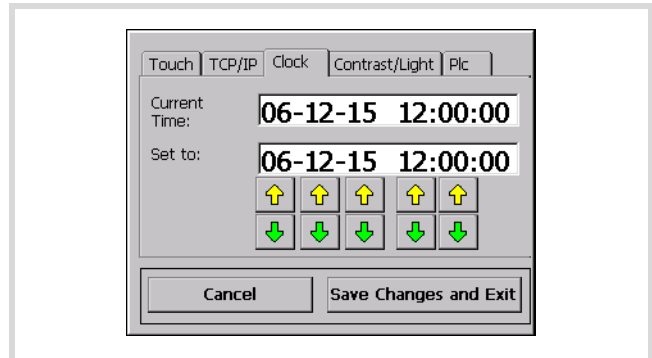


Figure 16: Setting the time

- ▶ Accept the values using the Save Changes and Exit button or abort the operation by tapping the Cancel button.

### Contrast and brightness setting

- ▶ Tap the Contrast/Light tab.

This shows the current values that you can change with the arrow buttons.

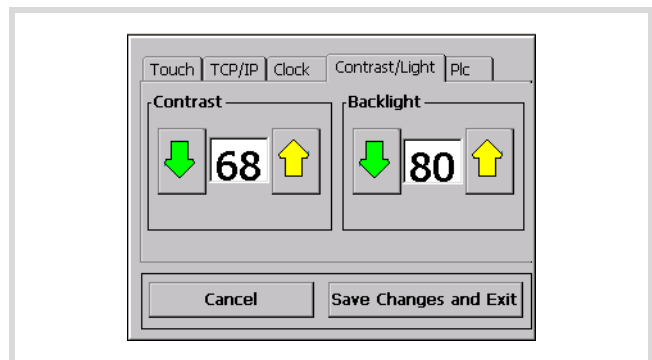


Figure 17: Screen for setting the contrast and brightness

- ▶ Accept the values with the Save Changes and Exit button or abort the operation with the Cancel button.

## Starting MFD4

- ▶ Tap the Plc tab.

This will show the following status image.

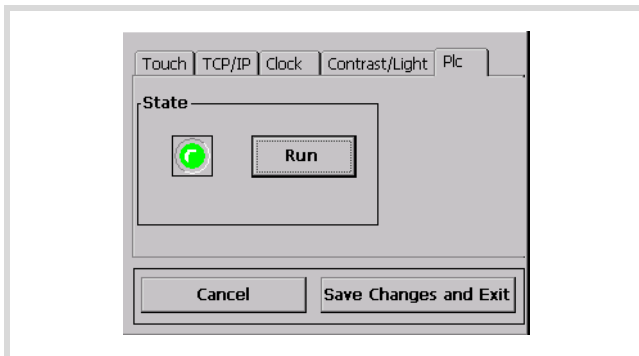


Figure 18: Status image for starting the MFD4

The green LED will flash if the device is in STOP mode. If a program is loaded you can then start it by tapping the Run button.

## Changing from application screen ↔ basic menu

### Changing to the basic menu

- ▶ To change from an application screen to the basic menu, add a button as shown in figure 19 to the application screen.

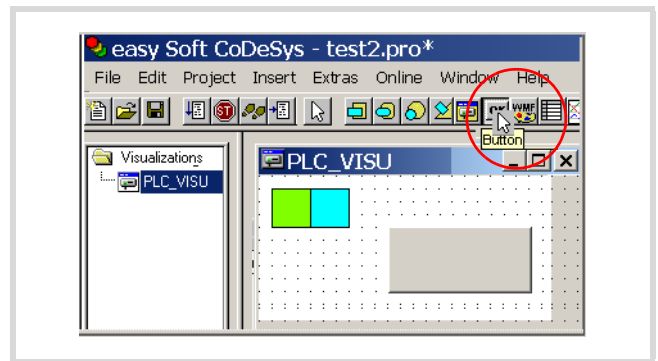


Figure 19: Application with button for changing to the basic menu

- ▶ Double-click the button with the left mouse button. The window will be displayed as shown in figure 20.
- ▶ Select the Input category.
- ▶ Activate Execute program and in the appropriate entry line write: MFDpanel.exe.
- ▶ Confirm the entry with the OK button.

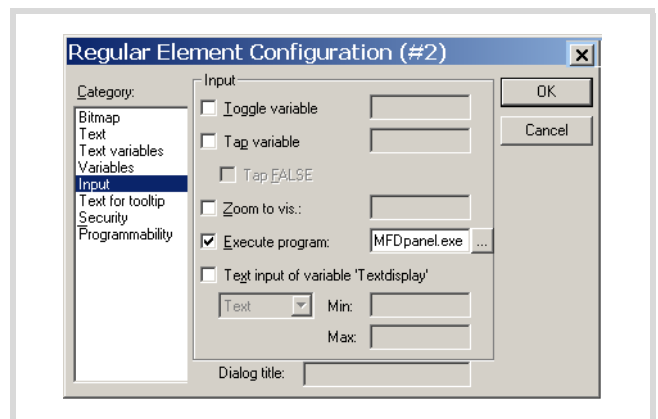


Figure 20: Configuring an element

You can touch the button after the program is started. The display with the basic menu will appear.

### Changing to the application screen

Touch the Save Changes on Exit button in the basic menu to show the Reboot screen. Touch No to return to the application screen.





## 5 Operation

### Switching on MDF4

Switching on the power supply will cause the display to show a run of information (information cycle) starting with System start.

If a boot project (with a visualisation task) is on the MFD4 and the operating mode switch is set to RUN, the first image of the project is displayed after the information cycle and the PLC program is started in the background.

→ Switching off the device during the startup/information cycle shortens the lifespan of the colour STN display

If the device does not contain a boot project and a program the display shows the Setup Menu 1:

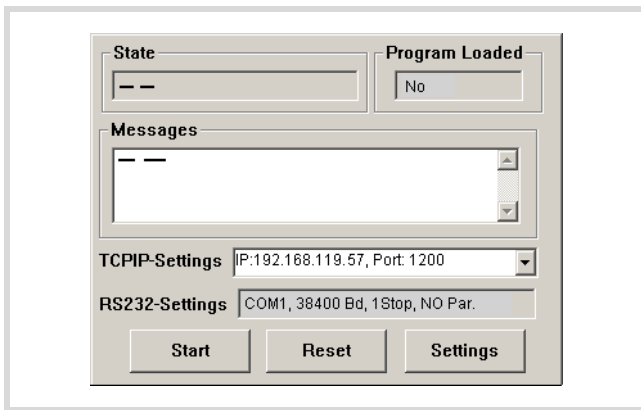


Figure 21: Setup menu 1

It contains information about the parameters of the Ethernet (TCP/IP setting) and RS232 interface. The parameters of the Ethernet interface can be changed on the display (→ section „Entering parameters of the Ethernet interface“, page 18). The baud rate of the RS232 interface can be adjusted with the browser command "setcomconfig" (→ section „Setting the MFD4 communication parameters“, page 38).

Logging on via the Ethernet interface and transferring a program to the MFD4 will refresh the display: The State field will show STOPPED and the Program Loaded field will show Yes.

The Messages field shows information about the loaded project, → figure 22.

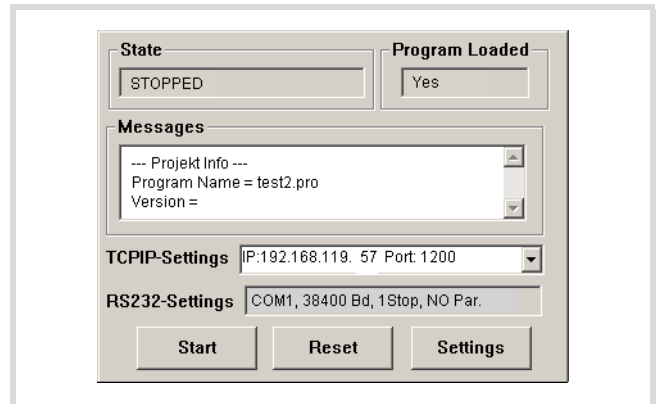


Figure 22: Setup menu 2

If the operating mode switch is set to RUN, you can start the program: In Online mode with the "Start" command or via the Start button in the Setup menu of the MFD4.

The first screen of the project is displayed and the PLC program is started in the background.

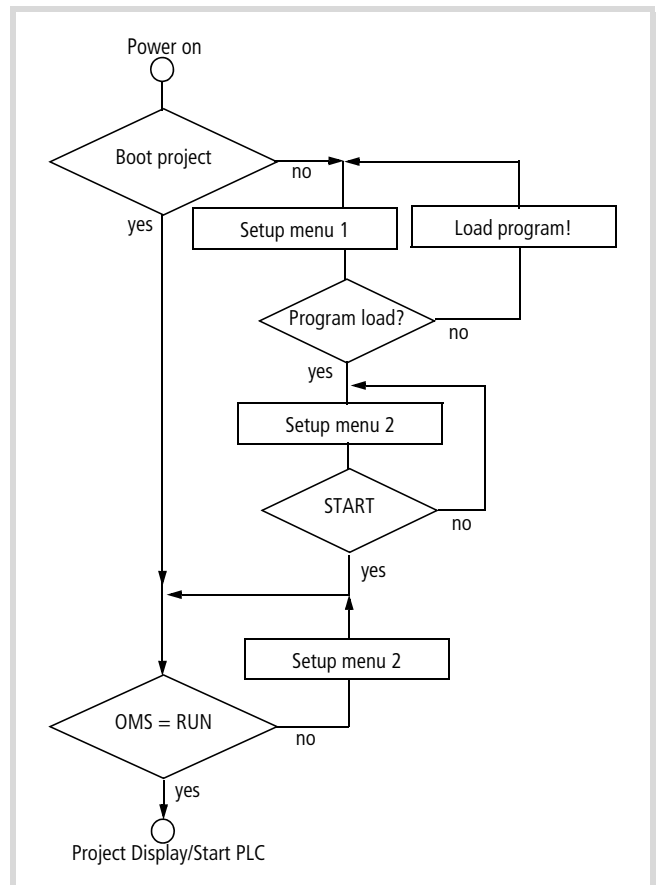


Figure 23: Startup behaviour of the MFD4  
OMS = Operating mode switch

### Operating mode switch

The operating mode switch enables you to start or stop the MFD4. You can also carry out a reset in conjunction with the SET button.

The operating mode switch is on the rear of the device. Turning the switch from STOP to RUN or from RUN to STOP will activate the required mode immediately.

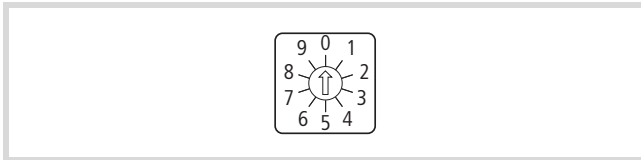


Figure 24: Three switch positions

**⚠ Caution!**  
If the operating mode is switched to position 1 (RUN) when the MFD4 contains a project, the screen on the display is refreshed and the PLC program is started immediately.

Table 4: Operating mode switch functions

Switch position	Function
0	STOP
1	RUN To start the project set the operating mode switch to position 1.
2, 3, 4	STOP
5	Restoring factory settings (factory set) → chapter „Reset for restoring the factory defaults“, page 26 → chapter „Startup behaviour after loading an operating system or reset“, page 22
6	STOP
7	STOP On pressing the SET button: RESET HARD
8	STOP On pressing the SET button: RESET COLD
9	STOP Pressing the SET button: RESET WARM

### SET button

The SET button is used to activate some functions that you have preselected with the operating mode switch, → table 4.

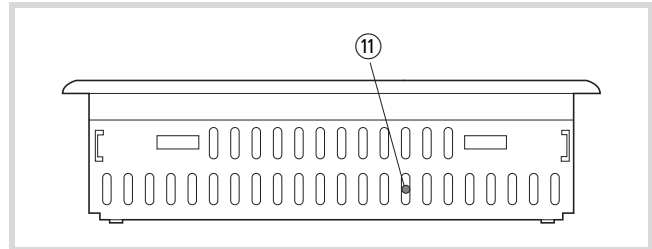


Figure 25: SET button ⑪

### Startup behaviour after loading an operating system or reset

If you have loaded a new operating system in the MFD4 or carried out a Reset for restoring the factory defaults (→ page 26), the device has the following startup behaviour: switching on the power supply will cause a bar to appear from the left of the screen that grows towards the middle. A SYSTEM START is displayed and carried out. After approx. 60 seconds, the display shows the message "FORMAT DISK in progress". The bar grows slowly towards the right. The display MOELLER Automation appears, followed by additional information until calibration is initiated, → section „User interface for display calibration“, page 17.

**Startup behaviour of the program**

Several different user programs/boot projects can be saved on the MFD4. They can be located on the MMC as well as on the DISK\_SYS system memory. However, the MFD4 simply runs a user program.

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

After voltage recovery, a boot project saved in the MFD4 will be started in accordance with the position of the operating mode switch and the programmed start conditions.

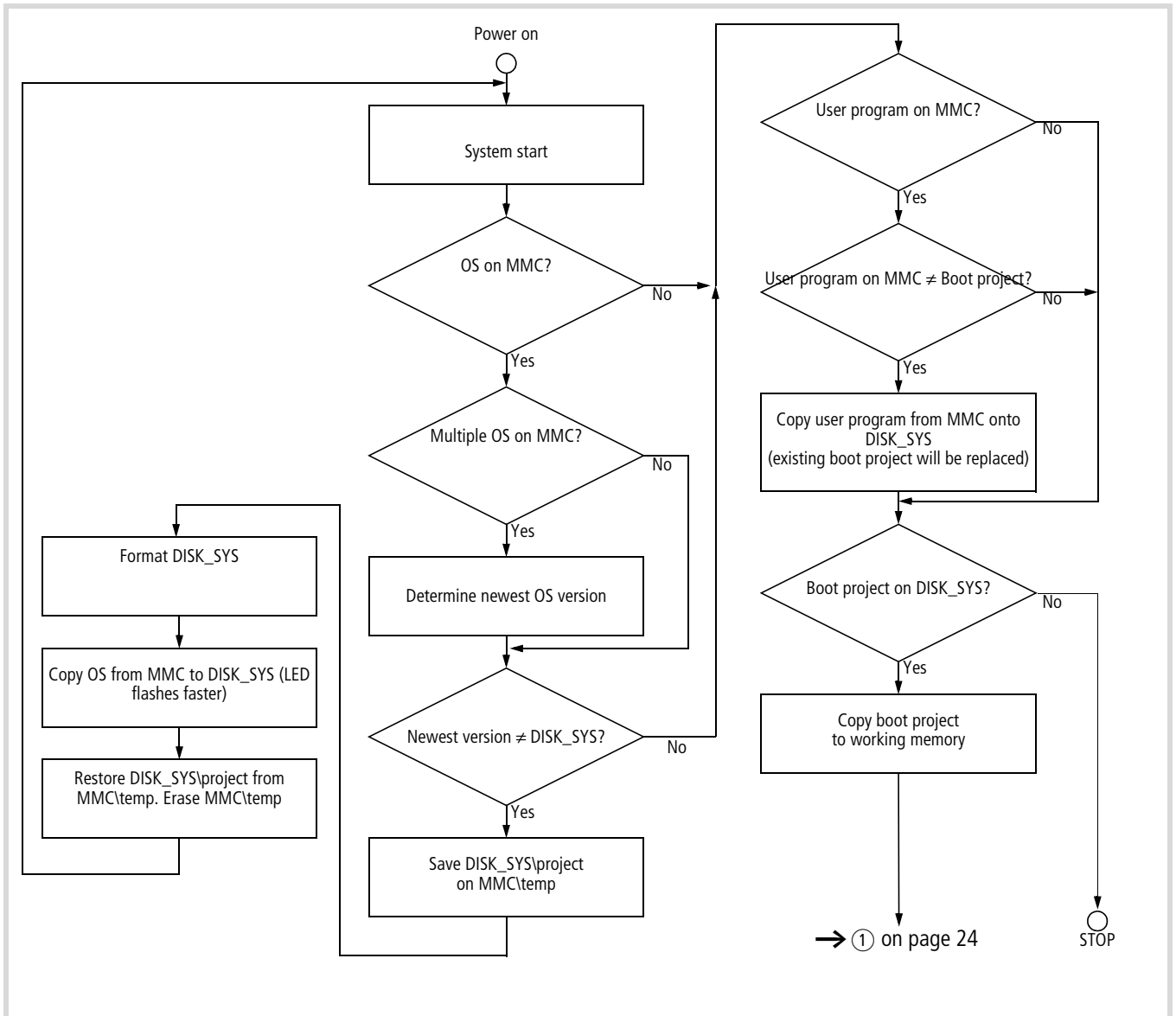
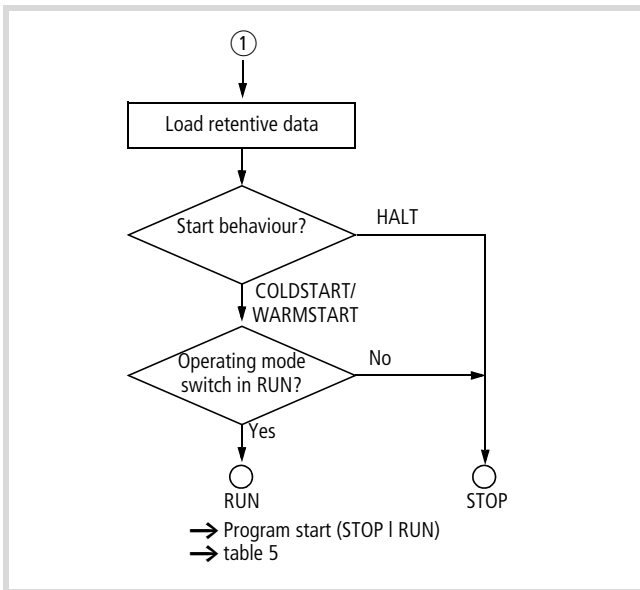


Figure 26: Start behaviour



### Program start (STOP → RUN)

You have the following possibilities to start the program:

	Program exists in main memory	Program should be loaded
<b>Pre-requisite</b>	<ul style="list-style-type: none"> <li>MFD4 in STOP</li> <li>Operating mode switch in STOP</li> </ul>	<ul style="list-style-type: none"> <li>MFD4 in STOP</li> <li>Operating mode switch in RUN</li> </ul>
<b>Action</b>	<ul style="list-style-type: none"> <li>Set operating mode switch to RUN or</li> <li>in online operation, issue the "Start" command.</li> </ul>	<ul style="list-style-type: none"> <li>Load program</li> <li>in online operation, issue the "Start" command.</li> <li>Tap the Start switch in the Setup menu of the MFD4</li> </ul>
<b>Result for all variables</b>	MFD4 in RUN Values are retained at the start	MFD4 in RUN Initial values are activated.

### Setting the startup behaviour in the programming software

The start-up behaviour setting primarily defines the handling of the retentive variables. The following settings are not activated until the power has been switched on.

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

- HALT
- COLDSTART
- WARMSTART

#### HALT

The user project is not started irrespective of the switch position of the operating mode switch.

#### COLDSTART/WARMSTART

Precondition: The operating mode switch is in RUN position.

The variables are initialised in accordance with table 5, before MFD4 starts.

Table 5: Behaviour of the variables after COLDSTART/WARMSTART

Variable type	Behaviour of the variables after ...	
	COLDSTART	WARMSTART
<b>Non-retentive</b>	Activation of the initial values	
<b>Retain<sup>1)</sup></b>	Activation of the initial values	Values remain in memory
<b>Persistent</b>	Activation of the initial values	
<b>Retain Persistent</b>	Values remain in memory	

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

### Program stop (RUN → STOP)

Changing the operating mode switch to the STOP position switches the MFD4 to STOP after the program cycle has been completed (ending of all active tasks).

After the task has ended the outputs used by the I/O task are set to 0, → chapter „Program processing, multitasking and system times“ on page 29.

You can stop the program in one of three ways:

- In online operation, issue the STOP command.
- Set the operating mode switch to its STOP position.
- Move to the basic menu (→ page 19). Open the PLC tab and tap Reset.

### Power off/interruption of the power supply

Switching off/disconnecting the power supply causes an immediate abort of the program cycle or tasks when a program is running. The data is then no longer consistent!

All outputs in which the I/O tasks are used are set to 0 or switched off, → chapter „Program processing, multitasking and system times“ on page 29. The behaviour of retentive variables in shown in can be seen in table 5.

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

If inconsistent data is not practicable for an application, additional measures must be designed such as the use of an uninterruptible power supply with a battery backup. The MFD4 is started as shown in (Start behaviour).

## Test and commissioning (Debugging)

The MFD4 supports you during test and commissioning with the following:

- Breakpoint/single-step mode
- Single-cycle mode
- Forcing
- Online modification, → PLC programming with CoDeSys manual (h1437g.pdf), Chapter Online functions
- Status indication/Powerflow.

### Breakpoint/single-step mode

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



#### Attention!

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

### Single-cycle mode

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

### Forcing

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

### Status indication

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

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

in IL:

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

## Reset

There are three different types of Reset commands:

- Warm reset
- Cold reset
- Full reset

The commands for initializing a retentive variable range are shown in table 6. The commands also affect the state of the CPU:

### Warm reset

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

### Cold reset

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

### Full reset

The program in the device and the boot project are erased. The variables are initialised. The device is switch to STOP.

Table 6: Behaviour of the variables after a Reset

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

1) After a full reset, the program must be reloaded. In online operation, the "Start" command can now be issued.

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

## Reset for restoring the factory defaults



### Attention!

Before this operation remove the MMC otherwise the boot project will be deleted.

A requirement for the reset is that the operating mode switch is in position 5.

- ▶ But switch off the supply voltage first!
- ▶ Press the SET button and switch on the power supply again with the SET button depressed.
- ▶ Hold down the SET button for at least 20 seconds, → section „Startup behaviour after loading an operating system or reset“ on page 22 .

All interfaces are initialised with default parameters. A loaded user program, all variables and the the boot project are deleted in the system memory (Flash) and on the MMC.

## Programs and project

### Loading the program

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



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

Program download is monitored.

### Creating a boot project

In order to safely store the program, a boot project must be generated from the user program. With the “Create boot project” command the program is loaded from the PC into the system memory and stored as a non-volatile boot project.

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

- ▶ Change over to the “Online” folder.
- ▶ Select the “Login” command.
- ▶ Select the “Create boot project” command.

### Storing the boot project on a memory card

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

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

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

### Deleting a boot project on a memory card

Click on the folder Resources → PLC Browser and enter, for example, the command for the MFD4.

```
filedelete \\disk_mmc\MOELLER\MFD57\project\default.prg
```

## Updating the operating system

On the MFD4 it is possible to replace the operating system (OS) by a more up-to-date operating system. Moeller offers the most recent operating system version for download on the Internet.

[ftp://ftp.moeller.net/AUTOMATION/DOWNLOAD/FIRMWARE\\_UPDATES/](ftp://ftp.moeller.net/AUTOMATION/DOWNLOAD/FIRMWARE_UPDATES/)

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

The operating system (OS) can be transferred in two ways.

- Directly from the PC to the device (only via RS232) → page 27
- From PC via the device to the MMC to the directory \disk\_mmc\moeller\MFD57\ → page 28.

The second variant is only possible for an Ethernet connection!

## Transferring the operating system from the PC to the device

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

Procedure:

- ▶ Establish a serial connection with the MFD4 via the RS232 interface of the PC. Information on this is provided in the sections "Interfaces" on page 14 and "Establishing a PC – MFD4 connection" on page 37.

→ The Baud rate is set to a fixed value of 115 200 Bit/s for loading the operating system.

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

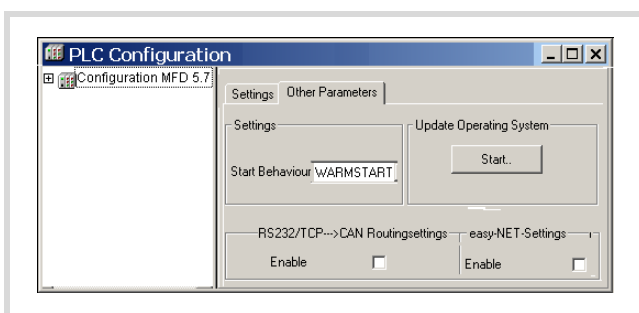


Figure 27: Updating the operating system

The "Download Tool" window opens.

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

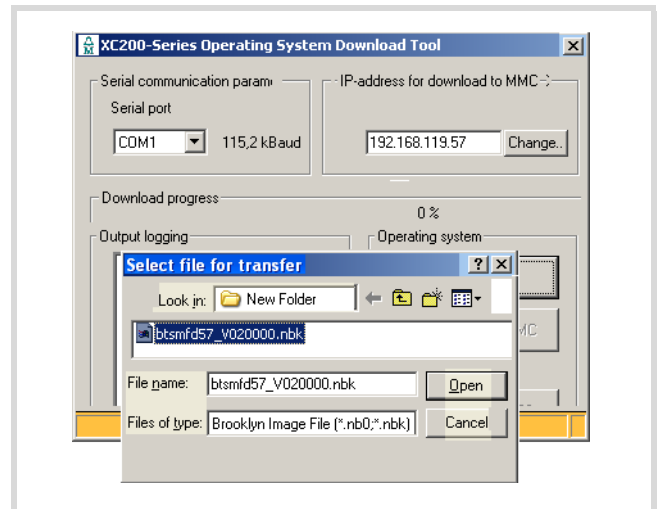


Figure 28: Operating system selection

- ▶ Open the operating system file to be transferred.

The following window appears:

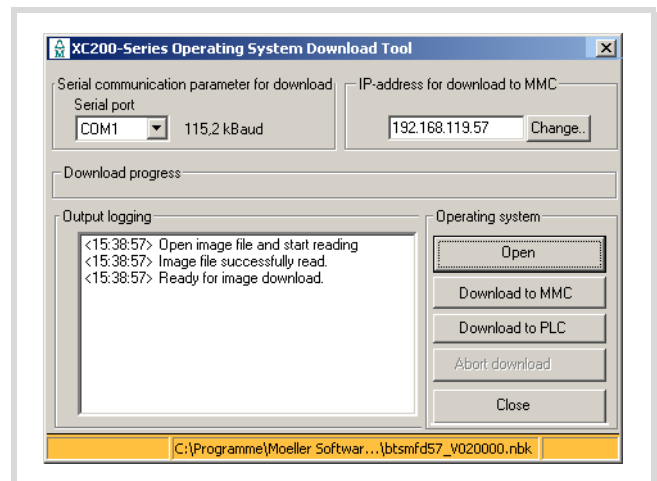


Figure 29: Download of the operating system

- ▶ Click on the "Download to PLC" button.

The protocol window shows the message "Connecting to PLC. Please reboot the MFD4."

- ▶ Switch off the control voltage of the MFD4 and wait a few seconds. This will ensure that the residual voltage is discharged.
- ▶ Switch the control voltage of the MFD4 back on.

The transfer of the operating system to the MFD4 is started. This can take several minutes. During the transfer a progress bar in the transfer field shows the volume of transferred data as a percentage.



### Attention!

Please do not engage in the download process until the "Ready for operating system transfer" appears for a second time on the download window.

When the transfer display shows 100%, the message "Flash erasing and programming. Please be patient" appears in the protocol window.

Further entries will appear in the protocol window. Only when the entry "Ready for image download" appears a second time is the download completed.

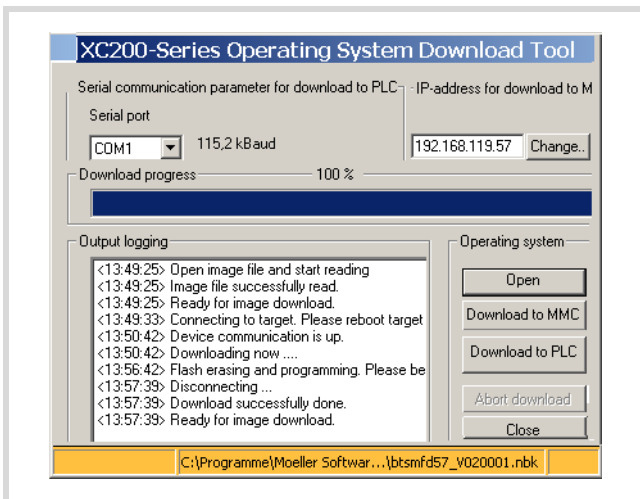


Figure 30: Download of the operating system ended

- End the download with the "Close" button.

The message "FORMAT DISK in progress" and a status bar appear on the display. Wait until the bar has covered the entire width of the screen, the system information was displayed in an image sequence and an English text appears with a cross-hair in the middle of the screen. Now calibrate the device. → chapter „User interface for display calibration“, page 17.

### Transferring the operating system from the PC into the MMC

#### PC → MMC

The process is similar to the transfer of the OS from the PC to the device. Simply click on the button "Transfer OS to MMC" (see → figure 29).

#### MMC → Device

Transfer the OS to the device. The OS is updated on power up → figure 26 on page 23.

### Erase operating system/boot project from the MMC

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

- Establish a connection to the MFD4 via the default address "ftp://192.168.119.57"
- Open the "disc\_mm\moeller\MFD57" directory.

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



## 6 Program processing, multitasking and system times

### Task configuration

The project can be controlled using several tasks. Each task can be assigned with a sequence of programs which should be run when the task is executed.

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

You can set Cyclical or Event-triggered as a task condition: a cyclical task is restarted after the set interval time. An event-triggered task is only started if the event occurs. Furthermore, you can associate system events such as Start or Stop with the execution of a program.

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

Before every task is called the output image is always written to the physical outputs and the input image is read (updating of the input/output image). The task is then executed. All system activities are also carried out before or after the task call. This includes for example, communication with the programming software, Online changes etc...

The updating of the input/output image of several tasks is described in section "Multitasking" on page 32.

All IEC tasks, including those with the highest priority can be interrupted by an interrupt or an event controlled task. Time monitoring can be activated for each task (Watchdog).

section "Creating a task (example)" on page 30 explains the PLC settings by means of an example. A detailed description of task configuration, interval times and priorities is provided in the online documentation PLC programming with CoDeSys 2.3 (h1437g.pdf) and the CoDeSys visualisation (h1528g.pdf) in the chapter Target visualisation.

### Standard task of the visualisation

After the MFD5.7 is selected, the functions Target Visualisation and VISU\_INPUT\_TASK are always activated in the Target Settings window.

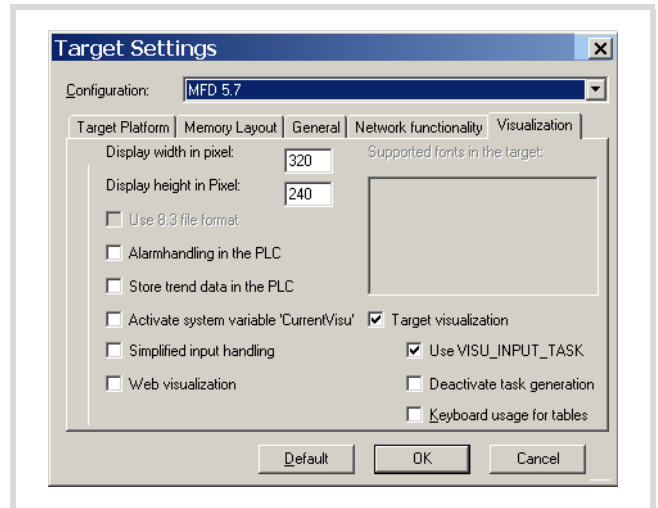


Figure 31: Target settings

The cyclical tasks are created in the task configuration shown in figure 32. Default values were entered for the interval times and priorities. For example, an interval time of 10 ms was entered for the PLC\_PRG\_TASK, which is configured as a user task.



The interval time should not be less than 10 ms otherwise there will be too much delay in the visualisation.

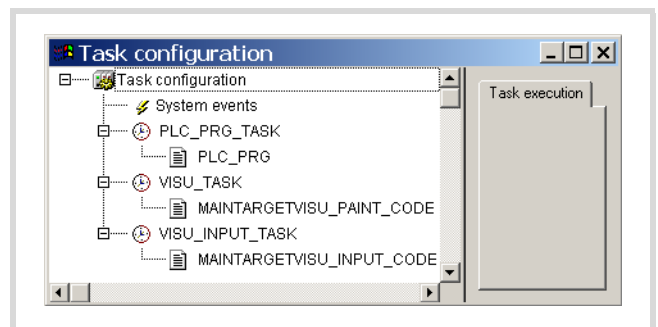


Figure 32: Task configuration

The SysLibTargetVisu.lib library is included in addition to the standard library on the tab Resources → Library Manager. The library Display.lib must be incorporated in the Library Manager if it is necessary to use the program in order to change display settings such as the contrast.

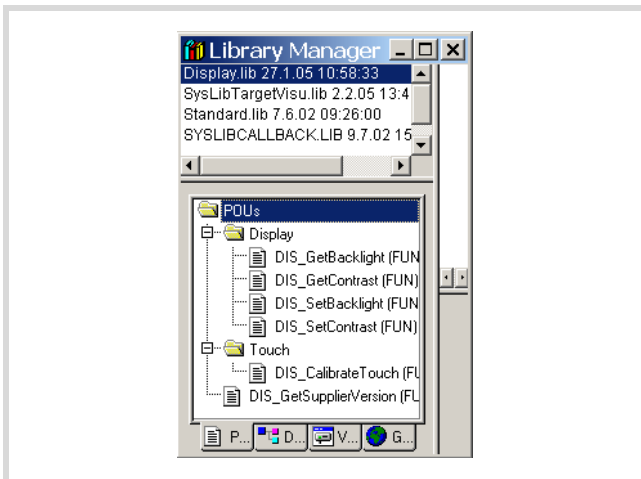


Figure 33: Library manager with display functions

You can activate alarm handling and trend data storing in the Target Settings (→ figure 31). These selections add the ALARM\_TASK and TREND\_TASK to the task configuration.

The Library Manager then contains the following libraries:

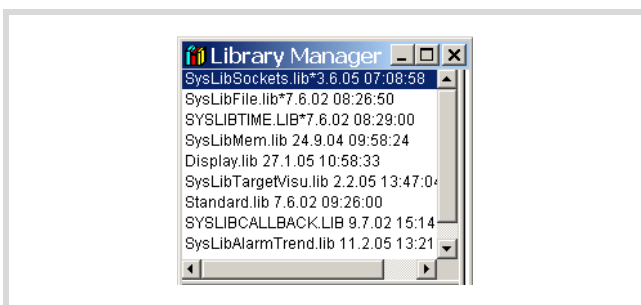


Figure 34: Libraries for alarm and trend functions

### Creating a task (example)

The section “Creating the event-triggered task “Param” and defining the program call” on page 31 describes the basic steps for creating a task.

The following example uses two tasks:

The first task PLC\_PRG\_TASK with the program call PLC\_PRG() is already added to the task configuration after the MFD4 is selected as target system (→ figure 32).

The second task “Param” with the program “Param\_prog” is event-triggered and must be created from scratch.

In the program PLC\_PRG (PLC\_PRG\_TASK) a variable “a” is set that is to call the event-triggered task “Param”.

### Creating the “PLC\_PRG” program

- ▶ Move from the task configuration to the POU tab and right-click on the default program POU PLC\_PRG. Enter the program as shown in figure 35.

Function of the program: The variable “count” is incremented. On counter status = 9, a = TRUE.

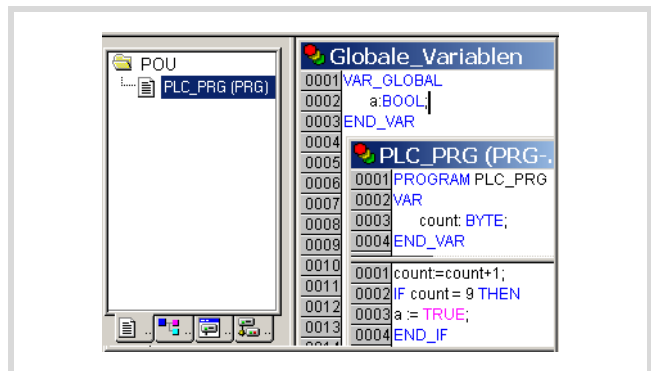


Figure 35: Creating a POU for a cyclic task

## Creating the event-triggered task "Param" and defining the program call

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

- Create POU (object) and program
- Add a task
- Define the program call

### Creating a POU and program

- ▶ Change over to the "POUs" tab and insert a program type object (POU) with the name "Param\_prog".

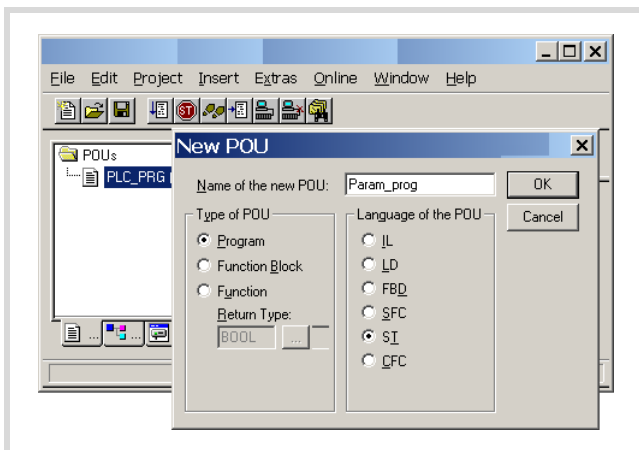


Figure 36: POU for event controlled task

- ▶ Enter the program as shown in figure 37. The program "Param\_prog" is processed if the variable a = TRUE. The variable "value" is then incremented by 1.

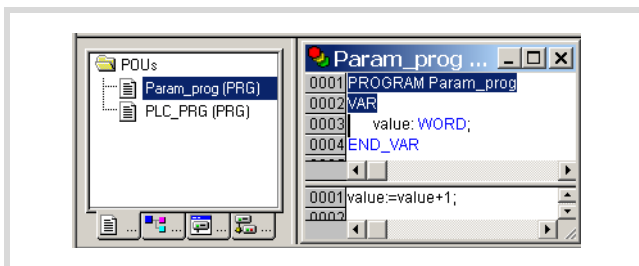


Figure 37: POU for event-triggered task

### Creating a task "Param"

- ▶ Open the "Task configuration" folder in the "Resources" tab
- ▶ Click with the right mouse button on the "Task configuration" folder and select the "Add task" command in the popup menu.
- ▶ Enter in the Name field a name such as "Param".
- ▶ Change the task type by activating the Triggered by event option.
- ▶ Define the Boolean variable "a" as the result of the event. → figure 38
- ▶ Click on the "Task configuration" folder and the configuration is accepted.

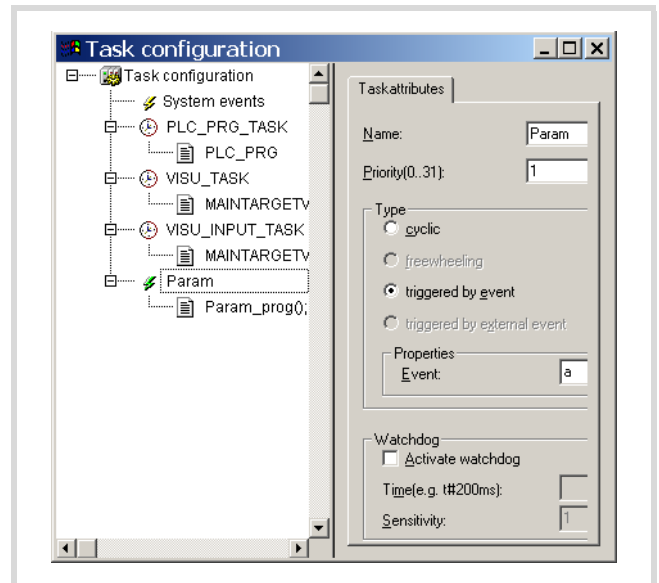


Figure 38: Parameterisation of the event-triggered task

### Defining the program call (Param\_prog)

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

- ▶ Click with the right mouse button on the lightning symbol of the "Param" task created beforehand and select the "Program call" command in the popup menu.
- ▶ Enter the name "Param\_prog" in the "Program call" window.
- ▶ Click the button at the end of the entry field.
- ▶ Select the name in the Input Assistant window and confirm the selection.

## System events

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

### Assigning a POU to a system event

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

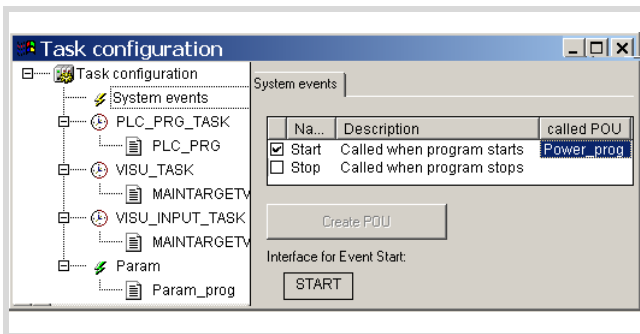


Figure 39: Assigning the POU to a system event

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

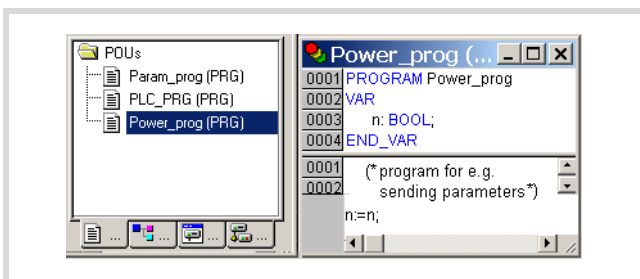


Figure 40: Programming a POU

- ➔ Further information on system events is provided in the manual "PLC programming with CoDeSys" (h1437g.pdf) and in the online help of the programming software.

## Multitasking

The MFD4 run time system is a multitasking system. This means that multiple tasks can be run at the same time (in parallel).

- ➔ Up to 10 tasks are possible. The parameterisation of a task as "free wheeling" is not supported.

### Update CANopen variables with multitasking

When using a CANopen master, the update of CAN variables is associated with a so-called update task. All CAN variables should be programmed in this task as all CAN variables are updated when the update task is called.

An existing task that has the highest priority functions as an update task. If another task is to take on the update function, the name of the new task must be entered in the Module parameters tab. The task should have a high or the highest priority.

- ➔ A multitasking system can contain individual tasks which can be interrupted as required according to their priority. If the update task has a low priority, it can be interrupted by a higher priority task. This behaviour may cause an inconsistency of the CAN variables.

When you incorporate the CAN master in the configuration, an entry with the name update task will appear in the Module parameters tab and the name Highest will appear in the Value column.

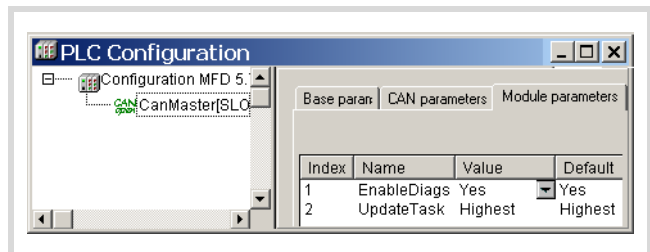


Figure 41: CANopen master update task

The name "Highest" indicates that the update of the CAN variables is associated with the task with the highest priority. This allocation can be changed: use the name of an existing task instead of "Highest" or create a new task with the name that you assign instead of Highest.

The name "Highest" was changed in PLC\_PRG in figure 42. The task should be assigned a high priority.

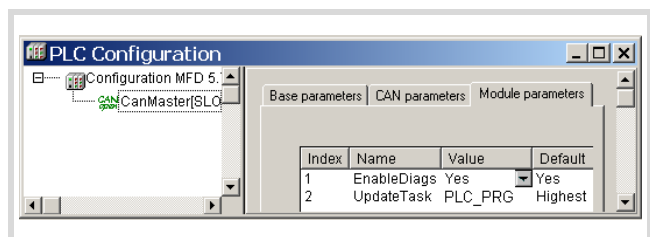


Figure 42: Changing the name of the CANopen update task

### Task monitoring with the watchdog

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

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

If the processing time exceeds the interval time, the end of the second interval time is awaited until the task is restarted (→ Watchdog deactivated).

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

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

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



#### Warning!

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

### Watchdog configuration

You can preselect the following settings in the task configuration:

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

These settings apply for time controlled and event controlled tasks.

### Watchdog active

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

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

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

The triggering of the watchdog continues to be dependent on the watchdog sensitivity. The watchdog sensitivity determines the number of successive watchdog timeouts after which the watchdog is triggered.

The watchdog is triggered:

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

For example, a task with a watchdog time of "10 ms" and a watchdog sensitivity of "5" will end at the latest after 10 ms × 5 = 50 ms.

Example:

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

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

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

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

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

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

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

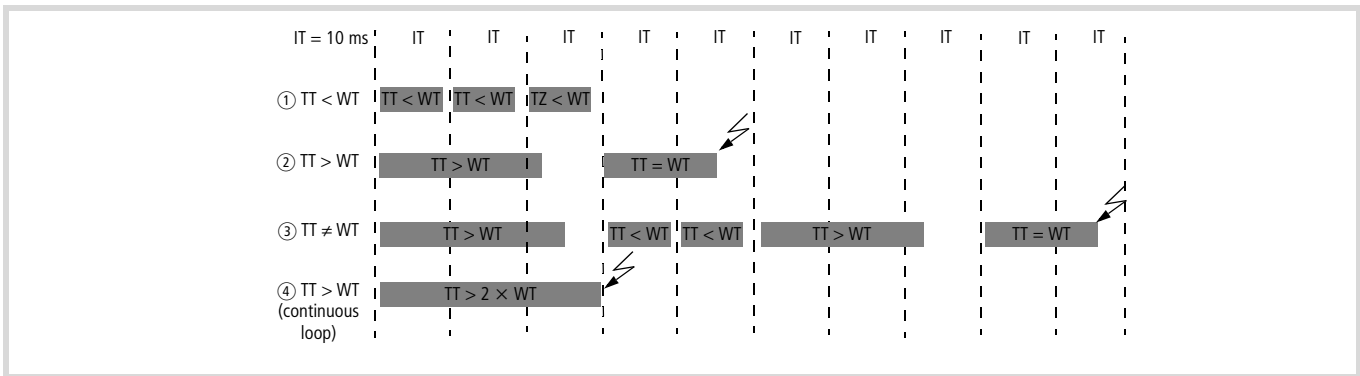


Figure 43: Watchdog active, multiple tasks with differing priority

### Watchdog deactivated

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

The interval time (IT) is 10 ms.

Variant ①: The interval time (IT) of a task was set to 10 ms. The actual task time (TT) is 15 ms. The task is started on the first call but is not terminated before the second cycle. This task is therefore not initiated in the second cycle again. Only in the third cycle – after 20 ms – can this task be restarted. The task therefore does not run every 10 ms but only in a time interval of  $2 \times 10$  ms.

Variant ②: The running cycle is not ended.

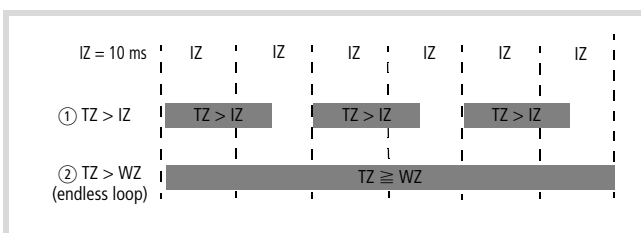


Figure 44: Watchdog deactivated

### Multiple tasks with the same priority

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

### Web visualization

The description of the web visualisation is provided in the "The CoDeSys visualisation" (h1528g).

The specific call for the web visualization is as follows:

<http://192.168.119.57:8080/webvisu.htm>

(Requirement: You have not changed the default setting of the IP address!

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



### Attention!

A max. of 10 clients may access the MFD4!

## Limit values for memory usage

The data memory is divided into memory segments. The memory size of the individual segments can be found in figure 45. The global data utilises multiple segments. The required amount can be specified to suit the size of the loaded program.

The segment size can be seen under <Resources → Target settings → Memory layout>:

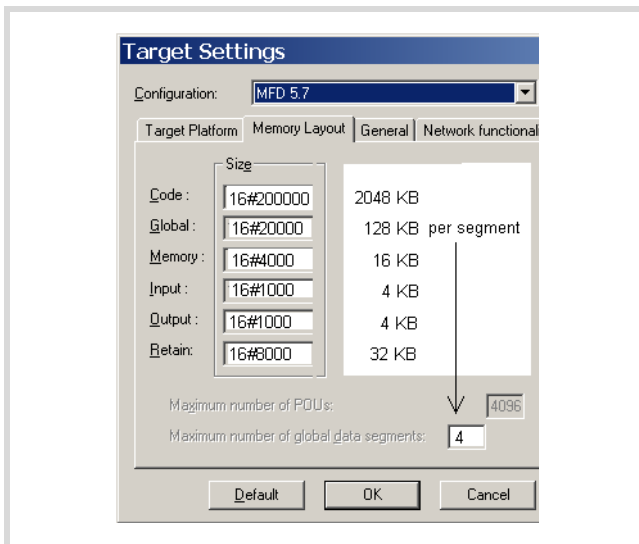


Figure 45: Size of the memory segments

To allow optimized, efficient usage of the memory range available for global data, set the number of global data segments to 4 when you create a new project.

The number of segments is set to 1 by default.

The number of segments is changed as follows:

- ▶ Select <Project → Options → Build options>; select the data segments field and enter the number of segments listed above for the respective device type.

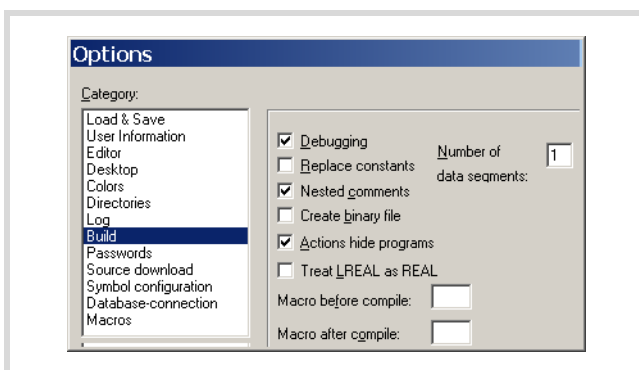


Figure 46: Memory management: Change the number of data segments

## Address range

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

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

## Addressing inputs/outputs and markers

The following functions for addressing are activated by default in the PLC configuration of a new project:

- "Activate Automatic addresses"
- Address overlaps

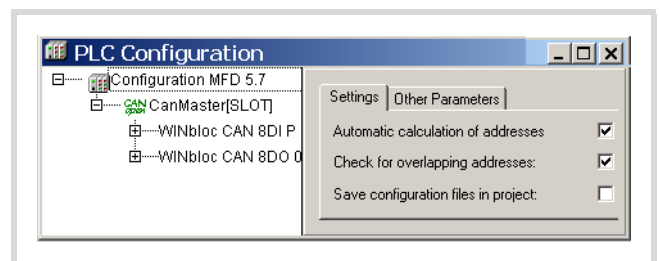


Figure 47: Default setting of the addressing

## "Automatic calculation of addresses"

In this function the addresses are automatically assigned or modified when changing or adding a module. If you add a module, the addresses of all subsequent modules (irrespective of the line) are adjusted by the address number of the added module. Modules that are located in front of the added module in the configuration are not changed. If you remove the tick at Automatic calculation of addresses, the addresses are kept in the event of changes/additions.

## Check for overlapping addresses

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



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

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

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



#### Attention!

The function Automatic calculation of addresses arranges the freely assigned addresses in ascending order.

### Run "Automatic calculation of addresses"

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

If you mark the "Configuration MFD 5.7" text and call the "Automatic calculation of addresses", all the addresses are recalculated.



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

### Uneven word addresses

(Independent of the "Check for overlapping addresses" setting)

When assigning a word-addressable module in the Input address field to an odd offset address, e.g. IB1, the subsequent even word address (IW2) appears in the PLC configurator.

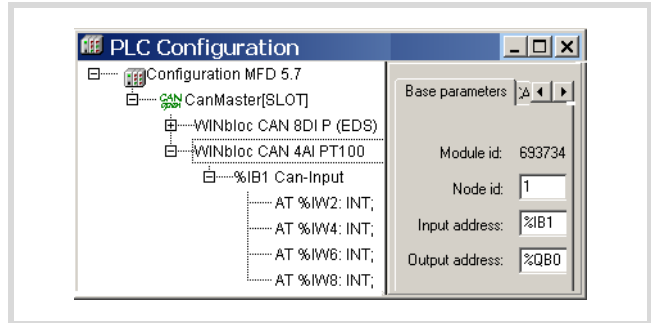


Figure 48: Uneven address



## 7 Establishing a PC – MFD4 connection

The connection between the PC and MFD4 can be established via:

- the RS 232 interface
- the Ethernet interface

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

See also:

- Interfaces → page 14

### Connection set-up via RS 232 interface

To establish a connection between PC and MFD4, the two devices' communication parameters must be the same.

- To match them, first adjust the PC's communication parameters to the MFD4's standard parameters settings → section „Setting the PC communication parameters“.

The MFD4 features the following standard parameters:

Baud rate	COM1
Parity	38400
Stop bits	1
Motorola Byte	No

→ If you get an error message during login, the MFD4's default settings have already been changed. In that case try a baud rate of 57600.

- After logging on the MFD4 parameters can be redefined (→ section „Setting the MFD4 communication parameters“, page 38).

### Setting the PC communication parameters

In the programming software define the communication parameters of the interface. You can use either the COM1 or the COM2 port of the PC.

- ▶ In the Online menu, select Communication → Parameters.
- ▶ Specify the port (COM1 or COM2), → section „Changing settings“
- ▶ Use the remaining settings as shown in figure 49.
- ▶ Confirm the settings with OK.
- ▶ Log on to the device.

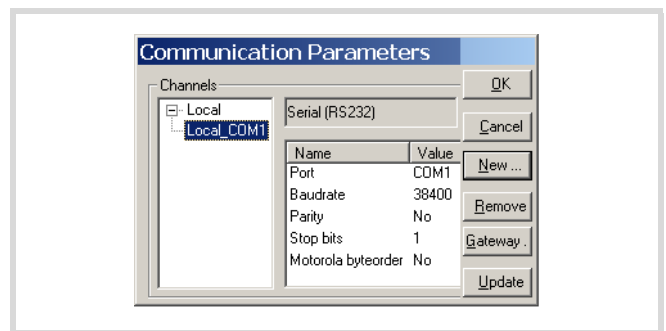


Figure 49: Setting communication parameters

→ Other information on communication parameters is provided in the manual "PLC programming with CoDeSys 2.3" (h1437g.pdf).

You can also select the communication channel "serial (RS232) (Level 2 Route)" and set a target ID. If you enter a 0 as target ID, communication is carried out with the local device.

### Changing settings

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

- ▶ Double-click the value, such as 38 400. The field will be underlined in grey.
- ▶ Enter the desired value.

You can double-click the field again to select the required baud rate, e.g. 57600 Bit/s.

### Setting the MFD4 communication parameters

- ▶ Select "PLC Browser" in the "Resources" tab.
- ▶ Select the "setcomconfig" browser command and add the required baud rate after inserting a space.
- ▶ Acknowledge the selection with RETURN.
- ▶ Select the "save registry" browser command.
- ▶ Select the "reboot" browser command. After reboot has been completed, the new baud rate is activated in the MFD4.

Now access the MFD4 (e.g. by a login), you will receive the following fault message:

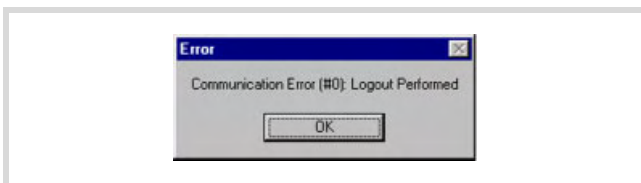


Figure 50: Communications fault

In order to communicate with the MFD4, you must adapt the communication settings of the PC, → section „Setting the PC communication parameters“.

### Connection set-up via Ethernet

After you have connected the PC to the MFD4 with a cable, select the TCP/IP communication channel in the programming software and enter the IP address of the MFD4. The MFD4 has the default address 192.168.119.57.

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

### Selecting communication channel and address

- ▶ Access the menu with <Online → Communication parameters>.
- ▶ Push the "New..." button.

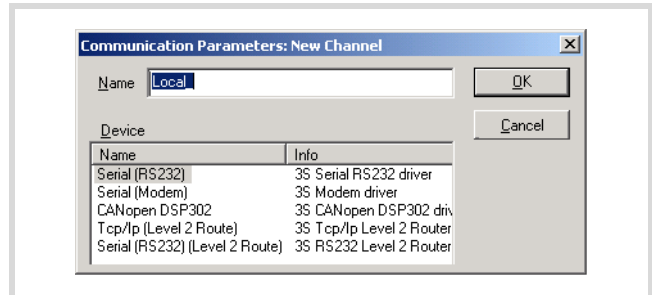


Figure 51: Channel selection

- ▶ Select the overview of the communication channel TCP/IP (Level2Route) and change the name "local" e.g. to "Ethernet-Test"
- ▶ Click OK to confirm.

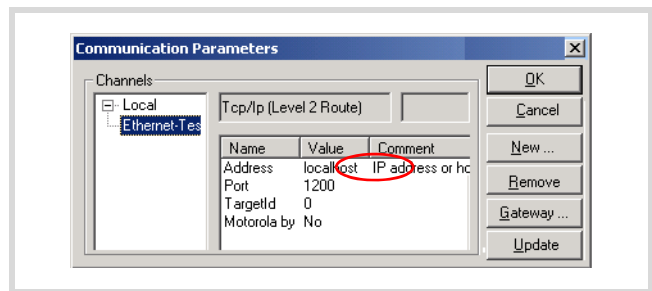


Figure 52: Enter the IP address

- ▶ Perform a double click on the "localhost" field and enter the default address 192.168.119.57

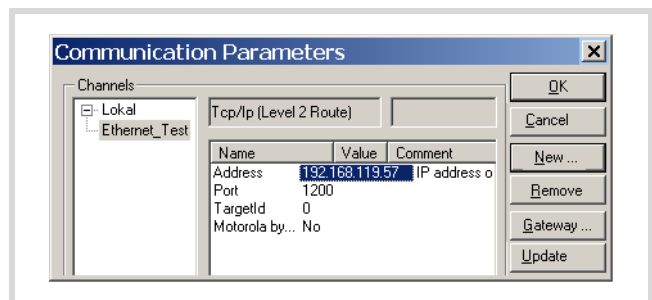


Figure 53: Communication parameters with IP address

- ▶ Confirm your details, by first pressing on another field and then on OK.
- ▶ Compile the program and log out.

### Scan/Modify the IP address

The "setipconfig" and "getipconfig" browser commands are available for modifying and scanning the IP address → section „Browser commands“ on page 57.

Restart the MFD4 after you have changed the IP address. The DHCP function (Dynamic Host Configuration Protocol) is not activated.

Ensure that the IP address of the programming device belongs to the same address family. This means, the IP address of the programming device and the MFD4 must correspond in the following number groups:

#### Example 1

IP address MFD4: 192.168.119.xxx  
IP address PC: 192.168.119.yyy

#### Example 2

IP address MFD4: 192.168.100.xxx  
IP address PC: 192.168.100.yyy

The following conditions apply in example 1 and 2:

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

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

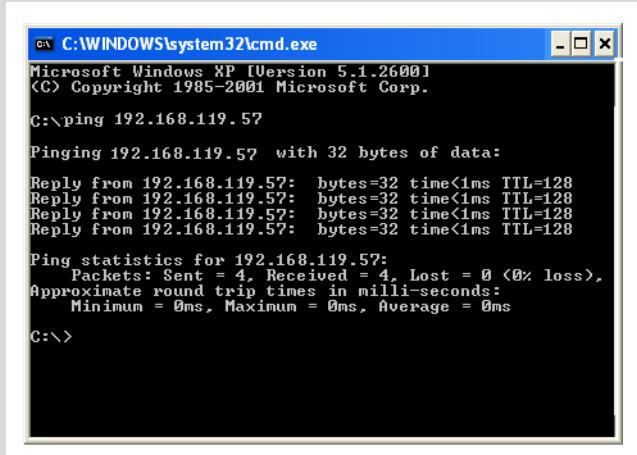
- ▶ Open the DOS window via the "Start" field and the "Run" command.
- ▶ Enter "CMD" in the input field and confirm with "OK".

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

- ▶ For the example mentioned you would enter the following text: "ping 192.168.119.57" and confirm this with "OK".

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

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



```

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

C:\>ping 192.168.119.57

Pinging 192.168.119.57 with 32 bytes of data:

Reply from 192.168.119.57: bytes=32 time<1ms TTL=128
Reply from 192.168.119.57: bytes=32 time<1ms TTL=128
Reply from 192.168.119.57: bytes=32 time<1ms TTL=128
Reply from 192.168.119.57: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.119.57:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
  
```

Figure 54: PING response with a correctly established Ethernet connection



## 8 Defining the system parameters via the STARTUP.INI file

### Overview

You can set project-independent system parameters and save them on the memory card. They are compiled there in a Startup.INI file. The memory card can also be plugged into other devices. The device accepts the parameters when started. The Startup.INI file is always created with all the device-specific settings (→ table 7).

### Parameters in the Startup.INI file

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

Table 7: Predefined default parameters in the Startup.INI

```
TARGET=MFD4-5-XRC-30
HOST_NAME=NoNameSet
IP_ADDRESS=192.168.119.57
IP_SUBNETMASK=255.255.255.0
COM_BAUDRATE=38400
CAN_ROUTING_CHANNEL=1
```

Table 8: Example: contents of the STARTUP.INI file

```
[STARTUP]
TARGET=MFD4-5-XRC-30
to the Ethernet connection:
HOST_NAME=NoNameSet
IP_ADDRESS=192.168.119.57
IP_SUBNETMASK=255.255.255.0
IP_GATEWAY=
IP_DNS=
IP_WINS=
to the programming interface RS232:
COM_BAUDRATE=4800, 9600, 19200, 38400, 57600
to the CAN interface:
CAN1_BAUDRATE: 10,20,50,100,125,250,500
CAN1_NODEID=1-127
CAN_ROUTEID=1-127
CAN_ROUTING_CHANNEL=1
```

### Structure of the INI file

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

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

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

```
; CAN_NODEID=2
```

The parameters can be changed or created with a text editor if you insert the memory card into the memory card slot of a PC. The file STARTUP.INI is saved on the memory card in the directory "MOELLER/MFD57PROJECT/".

### Creating the Startup.INI file

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

With the browser command "createstartupini" you will transfer from the device either the STARTUP data – or if it contains a project – the system parameters onto the memory card. This creates the Startup.INI file which contains this data.

Precondition: the memory card must be plugged in, formatted and empty, i.e. without Startup.ini file.

A file which already exists cannot be changed or overwritten by the browser command "createstartupini". If you still enter the command, a warning appears. In order to create a new file the existing file must be deleted first. → section „Deleting the Startup.INI file“ on page 42.

### Entry of the INI file: "HOST\_NAME"

With the help of the parameter "Device name (HOST\_NAME)" the PLC can be contacted via the Ethernet with this device name. Furthermore, it can also be contacted with its IP address. It receives the "NoNameSet" entry from the system. If it is not changed the device connected to the Ethernet can be contacted via its IP address. Using the browser command "settargetname ..." you enter a new device name. The device name must be unique for each device. It can also be used as a communication parameter (in the figure 55: alias) if it is defined in the programming software in the menu <Online → Communication parameter>. The parameters define the properties of the programming connection between PC and device.

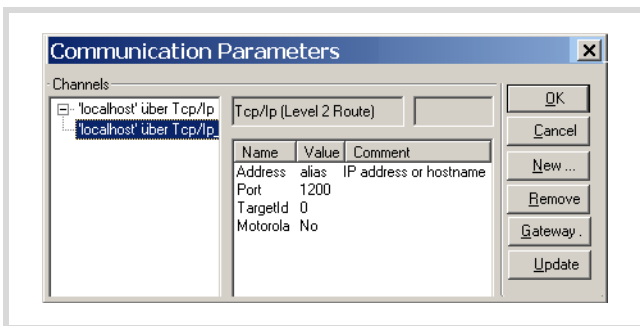


Figure 55: Communication parameters

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

### 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 settings

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

### Deleting the Startup.INI file

The following Browser commands are available for deleting.

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

The behaviour of the Startup.ini file with the Hard Reset menu command and the "factoryset" browser command is described in section "Reset" on page 25.

If you execute the "Full reset" command in online mode, the operating system and the project on the Disk\_sys are deleted. The STARTUP.INI file is retained.

## 9 Programming via the CANopen network (routing)

“Routing” is the capability to establish an Online connection from a programming device (PC) to any desired (routing capable) station in a CAN network, without having to directly connect the programming device with the target station. All actions can be implemented using the routing connection, which is available between the programming device and the station with a direct Online connection:

- Program download
- Online alterations
- Program test (Debugging)
- Generation of boot projects
- Write files to the station
- Read files from the station

Routing offers the advantage of being able to access all routing capable stations from a station connected with the programming device. You determine in the project selection in the programming software which station you wish to communicate with. This makes it possible to operate remotely configured controllers easily.

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

### Requirements

The following requirements must be fulfilled to use routing:

- Both the routing station and the target station support routing.
- Both stations are connected via the bus.
- The stations must both have the same active baud rate.
- Both stations must have a valid routing node ID.

### Routing properties of the MFD4

Routing can be carried out without the need to download a user program beforehand (Default: 125 Kbaud, Node ID 127). For this the target station does not have to be configured as a master or device.

For example, you can load a program from the PC via a station to the MFD4. For this assign a routing node ID to the MFD4 (target station).

### Optimise TCP/IP data transfer

If you select a TCP/IP connection between the PC and MFD4, you must adapt the size of the data blocks of the transfer type (program transfer or routing). The block size for fast program transfer to the MFD4 is 128 Kbyte (default setting). This is also possible with a block size of 4 Kbyte (somewhat slower). If you are routing via the MFD4 to the target station, the block size must be set to 4 Kbyte.

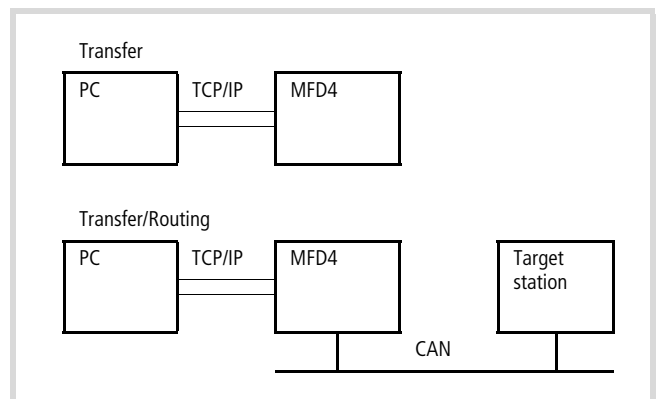


Figure 56: Optimising TCP/IP data transfer

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

### Setting the size of the data blocks

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

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

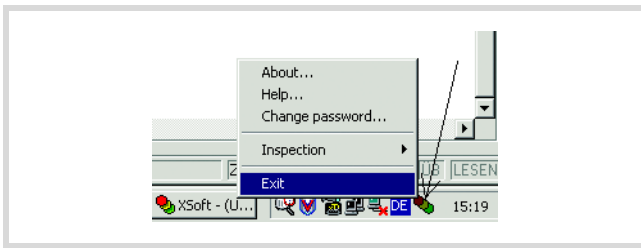


Figure 57: Closing the CoDeSys gateway server

► Change the block size to the required value.

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 20000 <sub>hex</sub> = 128 KByte (default value) in the Registry.
BlockSizeRout.reg	Enters a block size of 1000 <sub>hex</sub> = 4 KByte in the Registry.

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

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

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

The default block size is 20000<sub>hex</sub> (=128 Kbyte), the block size for routing is 1000<sub>hex</sub> (= 4 Kbyte).

**Notes**

- If larger files are written to the target station or read from the station, the online connection may be interrupted after the transfer. Renewed connection is possible.
- If a program with a modified routing Node ID is loaded into the target station via a routing station, the target station accepts the modified routing Node ID; however, the communication connection will be interrupted. Reconnection with a corrected routing Node ID is possible.
- A station cannot be accessed via a routing connection if it contains a program without any valid routing parameters (Baud rate/Node ID). The parameters may for example have been deleted by a HARD RESET, if the PC with the programming software was directly connected to the target station. The parameters are retained if the HARD RESET is carried out via the routing station.
- The routing is independent of the configuration (Master/Device): it is possible to access a target station which has not been configured as a master or as a device. It must only receive the basic parameters such as node ID and baud rate, as well as a simple program.

**Setting the node ID/routing ID**

Stations on the CANopen bus can be configured as a master or as a device. The controllers are assigned a Node ID/node number (address) so that they can be identified uniquely (for the basic communication). If you wish to use the routing function to access a target station, you must assign another routing ID to the routing and the target station. For example, the RS232 or the faster Ethernet interface can be used as connection between the PC and MFD4.

In the following example, the routing station is of type Master and the target station is of type Device.

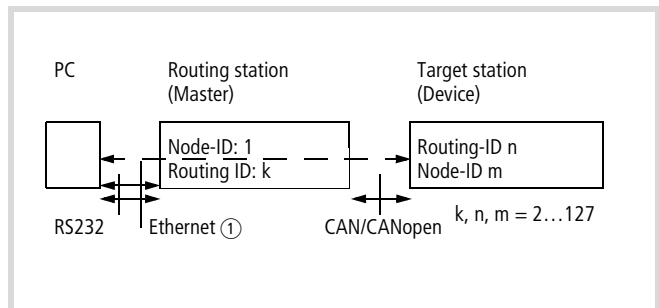


Figure 58: Routing via ID1=> XC..., EC4-200 or MFD4

① Ethernet connection only possible with XC200, MFD4 and EC4-200

Table 9: Example for setting the Node Id, routing ID, Baud rate

Station	Function	Node ID	Routing ID	Baud rate	→ Fig.
Routing station	Master	1	127	125 kB	60
Target station	Device	3	54	125 kB	61

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

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



### Setting the master station

Define two node IDs in the master station:

- One ID for routing
- One ID for basic communication

Define ID for the routing function:

- ▶ In the PLC configuration set the routing ID and the CAN baud rate in the Other Parameters tab, → figure 59.
  - First click the Enable check box in the RS232/TCP → CAN routing settings field. The activation is required so that the station can communicate with the bus.
  - Then enter the routing ID/node number and the baud rate on the bus in the appropriate entry fields.

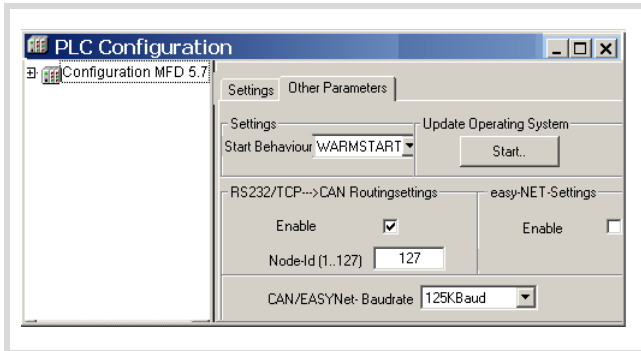


Figure 59: Routing settings for master and device

Define ID for basic communication

- ▶ The ID for basic communication is defined in the “CanMaster” folder in the “CAN parameters” tab (figure 60).

**Attention!**  
The baud rate for master and device must be identical.

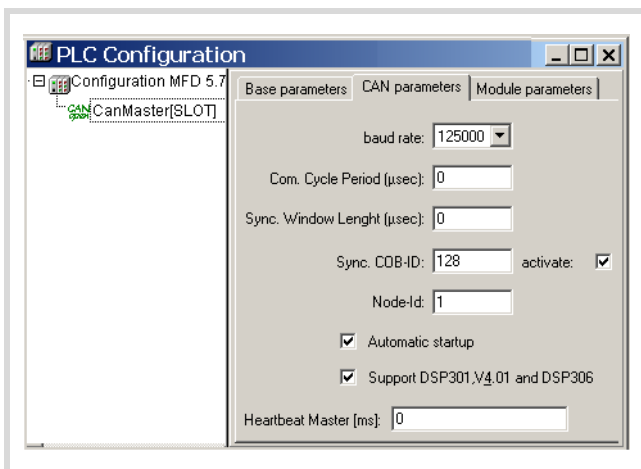


Figure 60: CAN Master: Node ID for basic communication

### Setting device (target) station

You assign two node IDs to the device (target) station:

- One ID for routing
- One ID for basic communication

Define the routing ID and the CAN baud rate of the target stations, e.g. MFD4, in the station configuration in the Other parameters window, → figure 59.

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

The ID for basic communication is defined in the “CanDevice” folder in the “CAN setting” tab, → figure 61.

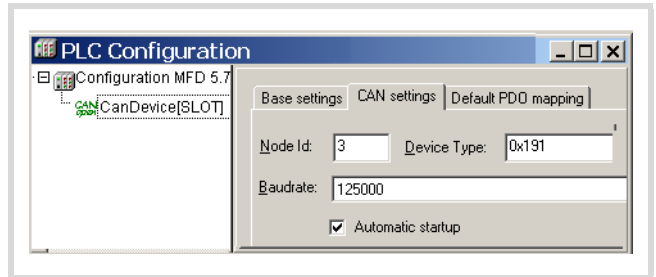


Figure 61: CAN-Deviceparameter

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

**Example**

The following example shows the access to the program of a station.

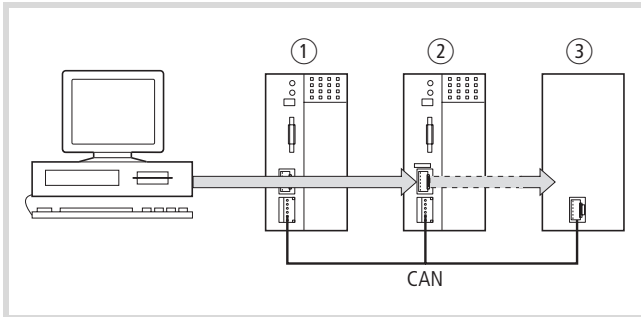


Figure 62: Diagnostics possibilities

- ① XC100 with Node ID 1
- ② XC200 (MFD4 also possible) with node ID 2, routing ID 127
- ③ Station (e.g. XC..., EC4-200, MFD4) with the node ID 3 and routing ID 54

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

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

The "New Channel" window appears.

- ▶ Select the channel in the "Device" window: Serial [RS 232] [Level 2 Route] or TCP/IP [Level 2 Route].
- ▶ You can assign a new name in the Name field, e.g. "Rout\_232."

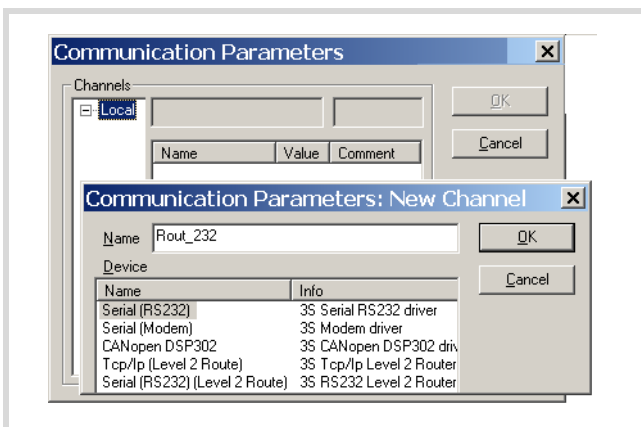


Figure 63: Channel parameter setting

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

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

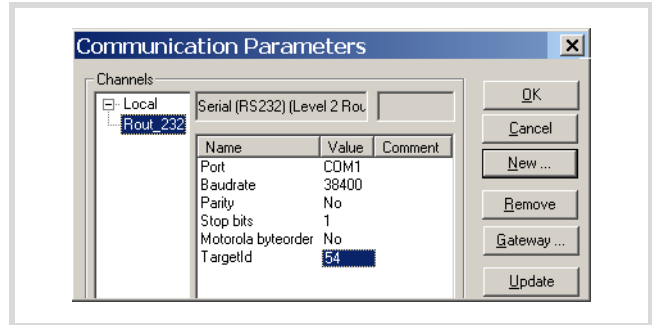


Figure 64: Setting the target ID of the target PLC

**Station combinations for routing**

The following stations support routing:

From →	XC100 XC121	XC200 MFD4	EC4-200
To ↓			
XC100, XC121	x	x	x
XC200 MFD4	x	x	x
EC4-200	x	x	x

**Number of communication channels**

Several communication channels can be opened, e.g. PC ↔ station 2, PC ↔ station 3 depending on the station (communication channel) which is connected to the PC. In this way, the status display of the stations 2 and 3 can be carried out at the same time.

Table 10: Type and number of communication channels

Communications channel	Station	Max. channel number
TCP/IP Level2Route	XC200/MFD4	5
Serial RS232 Level2Route	XC.../EC4-200/MFD4	1

## 10 RS 232 interface in Transparent mode

In Transparent mode data is exchanged between the MFD4 and the data terminals (e.g. terminals, printers, PCs, measuring devices) without interpreting the data. For this the RS232 serial interface (COM1) must be switched to Transparent mode via the user program.

→ Programming via the RS232 interface is not possible if it is in Transparent mode. However, the program can be tested via the Ethernet interface.

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

The libraries xSysCom200.lib or SysLibCom.lib provide functions for handling the data. Only one of the two libraries can be incorporated in the Library Manager. The SysLibCom.lib library was introduced in order to ensure compatibility between the MFD4 and other **xControl** devices.

Both libraries contain functions for opening and closing the interface, for sending and receiving the data and for setting the interface parameters. The data types of the libraries are not identical. The baud rate selection differs:

- xSysCom200.lib: 300, ..., 115200
- SysLibCom.lib: 4800, ..., 115200

For a better overview, the functions of the libraries are shown next to each other in figure 65.

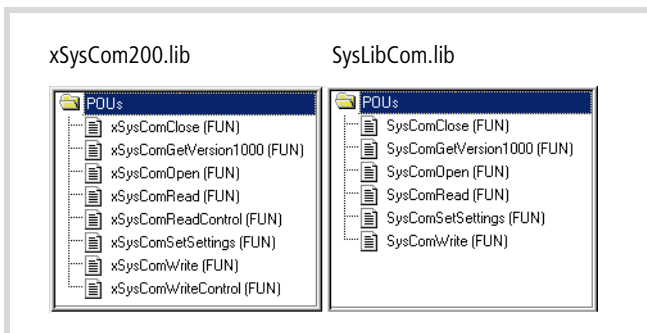


Figure 65: Function overview

The RS232 interface is addressed via the operating system. The execution of the interface functions can therefore take up to 50 ms. The task in which the RS232 interface is addressed should have an interval time of at least 50 ms and in multi-tasking operation should have a low priority (high value) so that time-critical tasks are not delayed.

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

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

The control cables of the interface are not supported by the software.

The individual functions are described in the online documentation "XSoft Function Blocks" (h1456g.pdb).

See also:

- Transparent mode: Text output via RS232 (example) → page 74
- RS 232 → page 15



## 11 Libraries, function blocks and functions

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

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

The libraries are located in the following folders:

- Lib\_Common for all devices
- Lib\_MFD57 for the MFD4.

### Using libraries

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

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.

Library	Online document <sup>1)</sup>
Standard.lib	h1437g.pdf
MFD57_Util. Lib	→ page 50
SysLib... .lib	Online help or PDF files
XS40_MoellerFB. Lib/ Visu. Lib/...	h1456g.pdf
CANUserLib. Lib CANUser_Master. Lib	h1554g.pdf

1) Further information on finding the files → page 5.

### Installing additional system libraries

You can install libraries manually as follows:

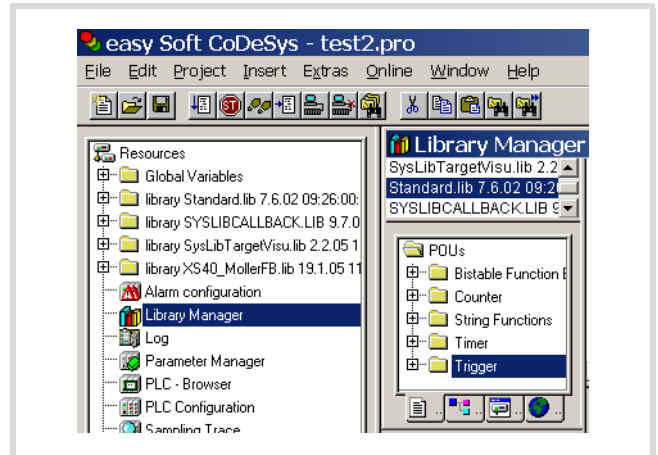


Figure 66: Libraries, installing manually

- ▶ In your project, click the Resources tab in the object organiser.
- ▶ Double-click the Library Manager element.
- ▶ Click on the menu <Insert → Additional Library... Ins>.

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

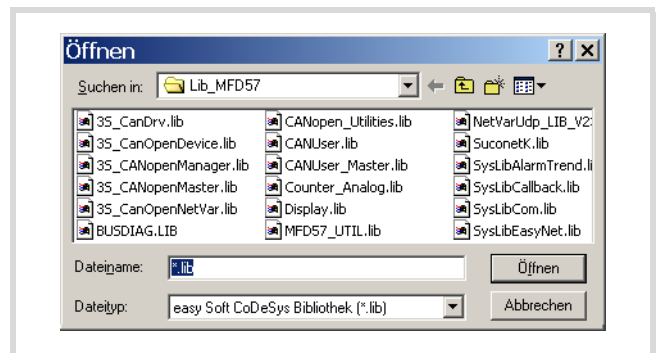


Figure 67: Selecting a library

- ▶ Select the library to install and click Open.

The library now appears in the Library Manager.

### MFD specific functions

The specific functions are contained in the libraries Display.lib and MFD57\_UTIL.lib.

#### Display.lib

The functions of the library allow you to change the properties of the display.

Function	Description
DIS_GetBackLight	Back-lighting
DIS_GetContrast	Read contrast values
SetBackLight	Set Back-lighting
DIS_SetContrast	Set contrast values
DIS_CalibrateTouch	Calibrate MFD4
DIS_GetSupplierVersion	Read library version

#### MFD57\_Util.

The functions of the library are divided into the following groups:

- CAN utilities, → page 50
- Ethernet utilities, → page 51
- General functions, → page 55.

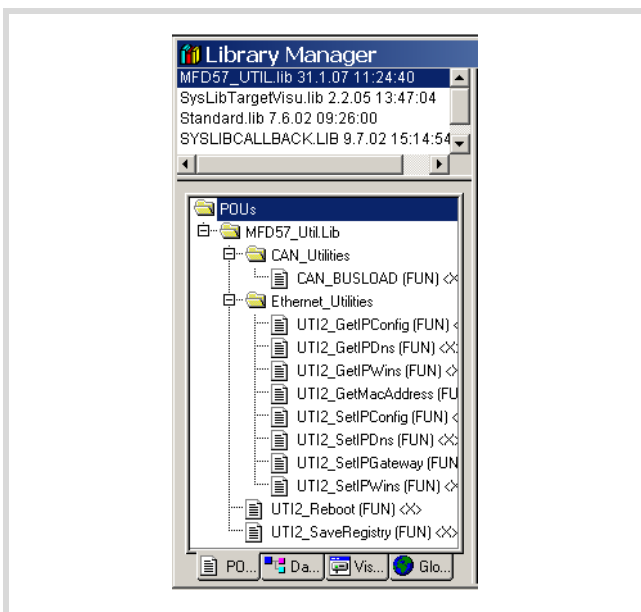


Figure 68: Specific functions of the MFD57-UTIL.lib library

### CAN utilities

The CAN\_BUSLOAD function is provided in the library MFD57\_Util.lib in the CAN\_Utilities folder.

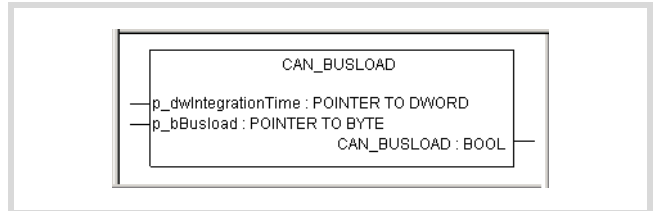


Figure 69: Function CAN\_BUSLOAD

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

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

A read cycle is 500 ms long.

See also:

- Display the loading of the CAN bus (canload) → page 60.

Ethernet utilities

**UTI2\_GetIPConfig**  
Get IP, subnet mask and IPGateway address

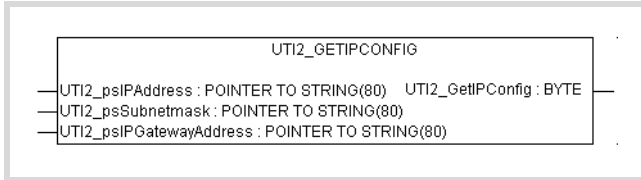


Figure 70: UTI2\_GetIPConfig

Table 11: Input variables

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

Table 12: Return value

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

**UTI2\_GetIPDNS**  
Display IP address of the dynamic name server set on the MFD4

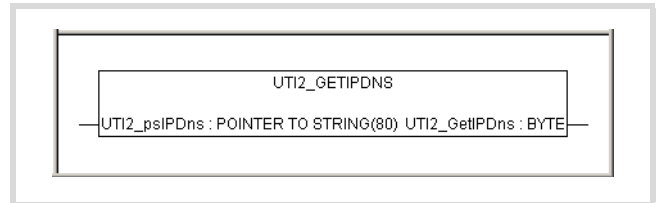


Figure 71: UTI2\_GetIPDNS

Table 13: Input variables

Input variables	Meaning
UTI2_psIPDns	Pointer to a string variable, in which the address of the dynamic name server is entered.

Table 14: Return value

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

**UT12\_GetIPWINS**  
**Display IP address of the Windows name server set on the MFD4**

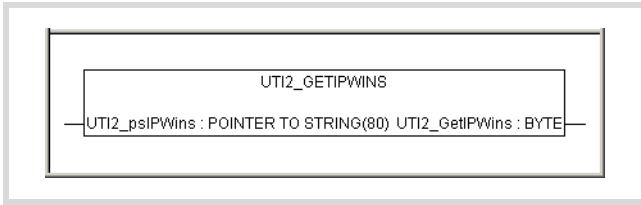


Figure 72: UT12\_GetIPWINS

Table 15: Input variables

Input variables	Meaning
UT12_pslPWins	Pointer to a string variable, in which the address of the dynamic name server is entered. (Netbios Name Server)

Table 16: Return value

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

**UT12\_GetMacAddress**  
**Get MAC address (MAC=Media Access Control)**

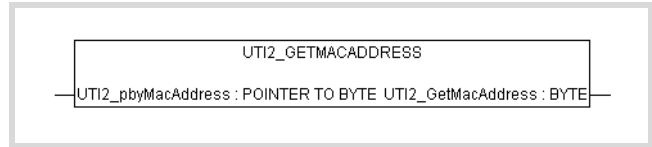


Figure 73: UT12\_GetMacAddress

Table 17: Input variables

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

Table 18: Return value

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



**UTI2\_SetIPConfig**  
**Setting the IP- and subnet mask address**

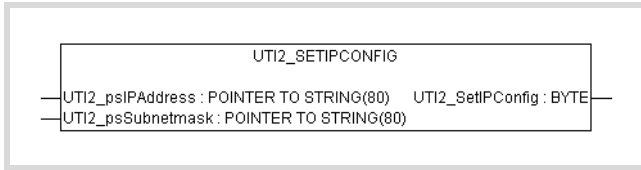


Figure 74: UTI2\_SetIPConfig

**Attention!**  
 A newly entered value must be saved as a non-volatile value by a “SaveRegistry” or a “Reboot” command. The newly entered value is accepted only after a restart of the device.

Table 19: Input variables

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

Table 20: Return value

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

**UTI2\_SetIPDNS**  
**Change the IP address of the dynamic name server on the MFD4**

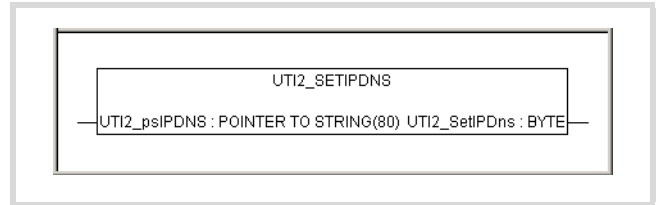


Figure 75: UTI2\_SetIPDNS

Table 21: Input variables

Input variables	Meaning
UTI2_psIPDNS	Pointer to a string variable, which contains the value to be entered from the dynamic name server.

Table 22: Return value

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

**UT12\_SetIPGateway**  
Setting of the IP Gateway address

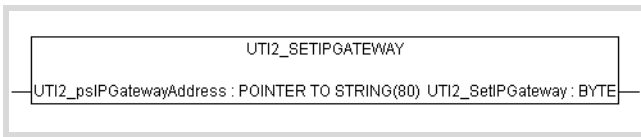


Figure 76: UT12\_SetIPGateway

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

Table 23: Input variables

Input variables	Meaning
UT12_psIPGatewayAddress	Pointer to a string variable, which contains the value to be entered for the gateway address.

Table 24: Return value

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

**UT12\_SetIPWINS**  
Change the IP address of the Windows name server on the MFD4

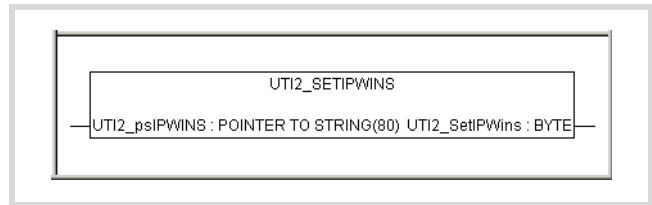


Figure 77: UT12\_SetIPWINS

Table 25: Input variables

Input variables	Meaning
UT12_psIPWINS	Pointer to a string variable, which contains the value to be entered for the Windows name server.

Table 26: Return value

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

## General functions

### UT12\_Reboot Restart with registry storage

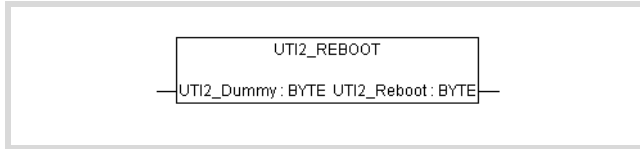


Figure 78: UT12\_Reboot

Table 27: Input variables

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

#### Return value

The value "1" is always returned.

### UT12\_SaveRegistry Saving of the registry

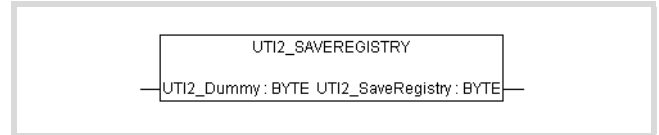


Figure 79: UT12\_SaveRegistry

Table 28: Input variables

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

Table 29: Return value

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



## 12 Browser commands

The browser is a text-based device (terminal) monitor. Browser commands can be used to query parameters and the status of the MFD4. To do this enter the command in the input line and confirm it with Enter (Return). The response appears in the result window. This function can be used for diagnostics and debugging purposes.

---

### Communication parameter access

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

- setcomconfig
- setipconfig
- setipgateway
- settargetname



#### Attention!

Delete the Startup.INI file before executing one of these commands, → chapter „Defining the system parameters via the STARTUP.INI file“ on page 41.

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

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

With the commands “setcomconfig”, “setipconfig”, “setipgateway”, “settargetname”, you must add additional parameters in the command line of the PLC browser, e.g. the Baud rate with “setcomconfig”, → table 30. Close the line by pressing RETURN. An answer will be shown in the results window.

The “setipconfig” browser command automatically generates a “settargetname”. The target name is comprised of a short description of the target system and the last numeric block of the IP address, e.g.: MFD57\_Nr010.

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

---

### Calling up browser commands

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

---

### Command overview

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

- Standard PLC 2.4 browser commands (assigned with a grey background in the table)
- Target system specific PLC 2.4 browser commands; these commands are managed in a file and are implemented in the runtime system.

Table 30: Browser commands

Command	Description	→ Page
?	Get a list of implemented commands	
canload	Display of the loading of the CANopen fieldbus	60
clearerrorlist	Erase error list	
cleareventlist	Erase event list	
copyprojtoMMC <sup>1)</sup>	Copy the (boot) project onto a Multi Media Card (incl. directory structure/project directory)	26
copyprojtoUSB <sup>1)</sup>	Copy the (boot) project onto the USB drive (incl. directory structure/project directory)	
createstartupini	Create the Startupini file on the disk_sys and disk_mmc	41
delpwd	Erase password for online access	
dpt	Output data pointer table	
filecopy <sup>1)</sup>	Copy file	60
filedelete <sup>1)</sup>	Erase file	60
filedir <sup>1)</sup>	Directory list [First folder in the list]	60
filerename <sup>1)</sup>	Rename file	60
getbattery	Display battery status	
getcomconfig	Display baud rate of serial interface 1	
geterrorlist	Display error list	
geteventlist	Display event list	
getipconfig	Display Ethernet address	39
getipdns	Display current DNS address	51
getipgateway	Display Gateway address	54
getipwins	Display current WINS address	52
getlanguage	Display dialog language for the error list	61
getmacaddress	Display MAC address [80-80-99-2-x-x]	52
getprgprop	Read program information	
getprgstat	Read program status	
getrtc	Display data and time [YY:MM:DD] [HH:MM:SS]	7
getswitchpos	Display status of the operating switch	
gettargetname	Display device name	42
getversion	Display version information	
memdisk_sys	Displays the free memory at "disk_sys"	
pid	Output project ID	
pinf	Output project information	
plcload <sup>1)</sup>	Display system performance: CPU usage	60
ppt	Output module pointer table	
reboot	Accept changes (registry save) and restart PLC	38
reflect <sup>1)</sup>	Mirror current command line for test purposes.	
reload	Reload boot project again	
removeprojfromMMC	Removes the backup project from the MMC	42
removestartupini	Erases the Startupini file from the disk_sys and disk_mmc	42
resetprg	Reset user program	
resetprgcold	User program cold reset	
resetprgorg	Reset user program to original state	

Command	Description	→ Page
restoreretain	Restore retentive data from file [File name]"	
saverestry	Accept modifications	38
saveretain	Save retentive data in the file [Filename]	
setcomconfig <sup>1)2)</sup>	Set the baud rate of the serial interface [setcomconfig4800,9600,19200,38400,57600,115200] e.g.: setcomconfig 38400	
setipconfig <sup>1)2)</sup>	Set Ethernet configuration [setipconfig adr1.adr2.adr3.adr4 mask1.mask2.mask3.mask4] z. B. setipconfig 192.168.119.010 255.255.255.000	39
setipdns	Set DNS address [setipdns adr1.adr2.adr3.adr4]	
setipgateway <sup>1)2)</sup>	Set gateway address [adr1.adr2.adr3.adr4]; e.g.: setipgateway 192.168.119.010	
setipwins	Set WINS address [setipwins adr1.adr2.adr3.adr4]	
setlanguage	Determine dialog language for error list [deu/eng/fra/ita]	61
setpwd	Activate password for online access	
setrtc <sup>1)</sup>	Set date and time [YY:MM:DD] [HH:MM:SS]; e.g.setrtc 03:07:24 10:46:33	7
settargetname <sup>1)2)</sup>	Set device name [devicename]; e.g.: settargetname test	
shutdown	Accept changes (registry save) and switch off PLC	57
startprg	Start user program	
stopprg	Stop user program	
tsk	Output IEC task list with task information	
updatefrommmc	Update windows image from /disk_mmc/MOELLER/MFD57/btsmfd57_Vxxxx.nbk	

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

2) Observe the section "Communication parameter access", page 57.

**Important browser commands**

**Display CPU loading (plcload)**

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

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

**Display the loading of the CAN bus (canload)**

The “canload” PLC browser command is part of the “MFD57\_Util.lib” library.

Examples for display:

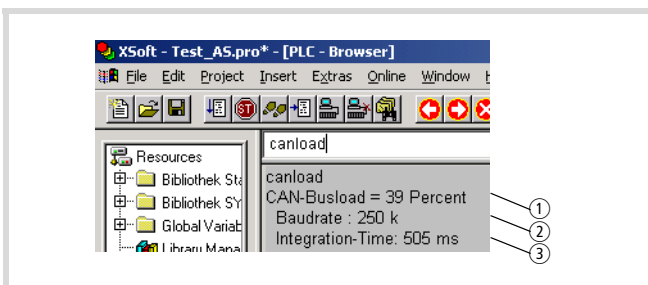


Figure 80: Loading of the CAN bus (Example 1)

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

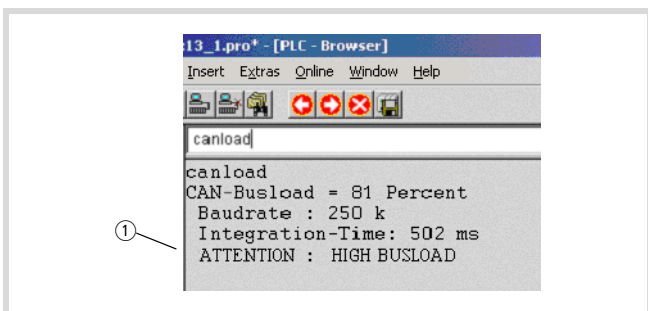


Figure 81: Loading of the CAN bus with alarm message (example 2)

- ① Alarm message, → table 31

Table 31: Possible alarm messages

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

**Access to memory objects**

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

- filecopy
- FileRename
- filedelete
- filedir.

Examples:

```
filedir (without parameter details the default setting is:
\\disk_sys\project)
filedir \\disk_sys
filedir \\disk_sys\project
filedir \\disk_mmc\MOELLER\MFD57
filedir \\disk_mmc\MOELLER\MFD57\project\aaa.prg
filecopy \\disk_sys\project\default.prg
\\disk_sys\project\yyy.prg
filerename \\disk_sys\project\yyy.prg \\disk_sys\project\
xxx.prg
filecopy \\disk_sys\project\default.prg \\disk_mmc \
MOELLER\MFD57\project\default.prg
filedelete \\disk_mmc\MOELLER\MFD57\project\default.prg
```



### Error and event list after calling browser commands

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

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

Examples for language conversion:

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

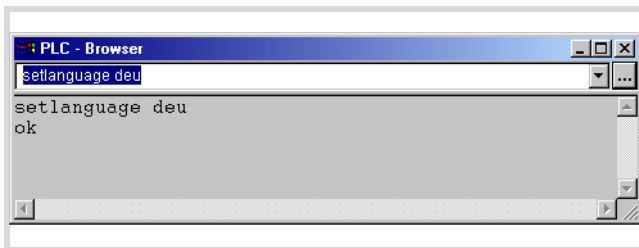


Figure 82: Browser command "setlanguage"

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

Module ID	Program type
1	RTS (runtime system)
2	CST (Moeller specific adaption)
4	CAN
5	IEC

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

Module ID	Event-ID	Error message
2	1	Stop program
2	2	Start program
2	3	Reset warm
2	4	Cold reset
2	5	Reset Hard
2	6	Battery empty
2	7	No program loaded
2	8	Task monitoring
4	10	CAN controller started
4	20	CAN controller stopped
4	30	Overflow
4	31	Overflow
4	40	Overflow

Module ID	Event-ID	Error message
4	41	Overflow
4	42	Overflow
4	50	Critical CAN fault
4	60	CAN controller in status error warning
4	70	CAN controller in status Bus-Off
1	16	Task monitoring fault
1	17	Hardware monitoring fault
1	18	Bus error
1	19	Checksum error
1	20	Fieldbus error
1	21	I/O update fault
1	22	Cycle time exceeded
1	80	Invalid instruction
1	81	Access violation
1	82	Privileged instruction
1	83	Page fault
1	84	Stack overflow
1	85	Invalid scheduling
1	86	Invalid access Identity
1	87	Access on protected page
1	256	Access to uneven address
1	257	Array limit exceeded
1	258	Division by zero
1	259	Overflow
1	260	Exception cant be overlooked
1	336	Floating decimal point: General fault
1	337	Floating decimal point: Not normalised operand
1	338	Floating decimal point: Division by zero
1	339	Floating decimal point: Inexact result
1	340	Floating decimal point: Invalid instruction
1	341	Floating decimal point: Overflow
1	342	Floating decimal point: Stack verification error
1	343	Floating decimal point: Underflow



## 13 Network easyNet

### Overview

easyNet is based on the CAN network which enables the exchange of process and system data. This network is designed for 8 stations (PLCs). Each station receives an ID number 1...8 and can exchange data with all other stations. Data formats such as Bit, Byte, Word and Double word are available. The station with ID number 1 must always be present. This station handles the management of the communication on the network. It is the only station that can communicate with remote I/O stations.

“Remote I/O” stations are PLCs without a program, such as for example an easy800, with outputs the station with ID1 can set and with inputs that station with ID1 can query.

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 following section describes stations that are programmed with easySoft-CoDeSys, which are IEC stations. Stations that are programmed with easySoft are called easy stations.

The connection of the IEC stations to the easyNet is described in detail in the online documentation “Description of the SysLibEasyNet.lib for connecting XC controllers to an easyNet network” (SysLibEasyNet-G.PDF).

The functions for data transfer, including the data types and structures are described in particular. The PDF file was stored with the installation of the product CD in the directory CAA-Targets\Moeller\Lib\_CPU201.

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 easyNET station.

→ In the sections on easyNet, the easy800 is primarily used in the examples. This can be replaced with the same easyNet function by the network-capable MFD-Titan visualisation device. The MFD4 can also be used for the XC200.

### Monitoring easyNet

#### Baud rate

Each station in easyNet must have the same baud rate. IEC stations such as MFD4 or XC200 can be run on the easyNet with a baud rate of 50 Kbits/s or 125 Kbits/s, an EC4-200 with a baud rate of 10, 20, 50, 125, 250, or 500 Kbits/s. easy stations operate with a baud rate of 10, 20, 50, 125, 250, 500 or 1000 Kbits/s. If you are operating easy and IEC stations on a network only one common baud rate is possible!

#### Bus utilisation

Ensure that the bus utilisation does not exceed 70 %. The bus utilisation can be checked using the PLC browser command “canload”. After the command is executed, the utilisation of the bus system is displayed. Carry out the command several times in order to compensate for any deviations in the bus load.

#### Receiving CAN telegrams

The receipt of CAN telegrams can also be monitored. Further information on this is provided in the online documentation “Description of the SysLibEasyNet.lib for connecting XC controllers to an easyNet network” in the chapter “Programming instructions/Troubleshooting” (SysLibEasyNetG.pdf). The PDF file was stored in the directory CAA-Targets\Moeller\Lib\_CPU201 when the product CD was installed.

### Sending/receiving user data

The access of the MFD4 to the data of the easyNet stations is implemented with the NET\_UPDATE and NET\_GET functions in the user program (application program). The functions belong to the library SysLibEasyNet.lib. The actual data transfer between an IEC station and the other stations is handled by a protocol task that is invisible to the user and which operates independently of the user program. The data exchange is initiated when the function is called.

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

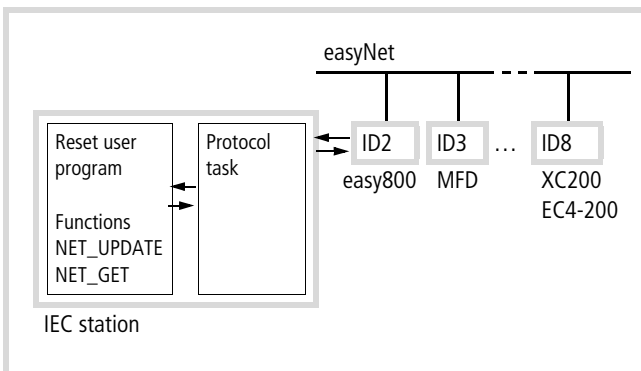


Figure 83: Data transfer of an IEC station with other stations

### NET\_UPDATE function

The function NET\_UPDATE is used to execute data exchange between the user program and the protocol task.

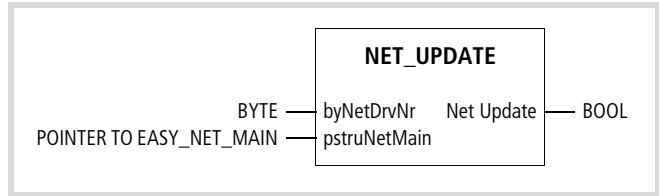


Figure 84: NET\_UPDATE function

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.

With every call of the NET\_GET function a data value can be collected. The data value on the net is selected via the entries byNET\_ID and byModulNumberof the EASY\_NET\_GET-structure. The protocol task enters the current data in the structure with the call up of the function. Any number of data values can be read by multiple calls of the NET\_GET function.

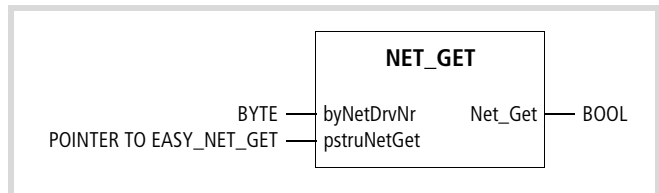


Figure 85: NET\_GET function

### Data transfer

There are 3 data transfer options:

- Data transfer between a station with ID1 and remote I/O device, → page 65
- Transfer of bit data blocks between several stations, → page 66
- Transfer of D words (32-bit) according to the PUT-GET principle between several PLCs, → page 67

### Data transfer between a station with 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 is also true for the expansion of the easy 800.

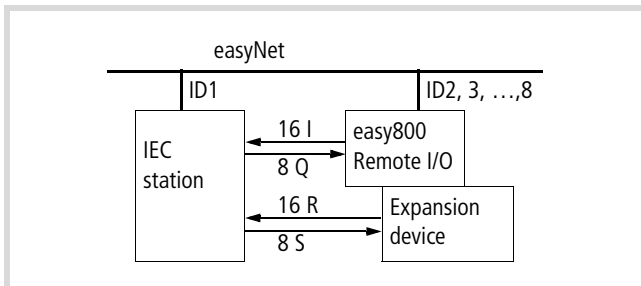


Figure 86: Data transfer between the IEC station and remote I/O

To set the outputs and scan the inputs of the easy800 (ID2), call the NET\_UPDATE function in the user program of the (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 87.

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

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

When the station is in RUN mode, the value is transferred to the outputs of the easy800 after each change.

The 16 inputs of the easy800 are accepted by the PLC with the structure variables NET\_MAIN.RCV[2].wl.

```

PLC_PRG (PRG-ST)
0001 PROGRAM PLC_PRG (* IEC-station (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 (* IEC-station -> easy800 Output *)
0005
0006 INPUTS:= NET_MAIN.RCV[2].wl;
0007 (* IEC-station <- easy800 Input *)
0008
0009
    
```

Figure 87: Program example of a data transfer between ID1 and remote I/O

The figure 88 shows the online display of the program from which the send and receive data originates. Note that the output value #1605 (aToID[05].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     wl = 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]
0026       byQ = 16#05 ← (* to XC200 *) 1)
0027       byS = 16#00 (*EC4-200*)
0028       wl = 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 88: Online display of the Data transfer between ID1 and remote I/O

### Transfer of bit data blocks between several stations

Each station can send a 32-bit long data block to a specific station via the easyNet. To do this call the function NET\_UPDATE in the program and enter the station number X of the recipient via the structure (EASY\_NET\_MAIN) (SND.aToID[X].dwSN). The data to be sent is written in the variable dwSN.

To receive data from other stations, use the structure (EASY\_NET\_MAIN).RCV[x], in which x represents the NET-ID of the station from which the data is to be received.

figure 89 shows that every station 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 MFD4 corresponds to the ID of the receiving/sending station.

The received value is in an element of the receive array of the MFD4 with a position that is identical to the ID of the send station.

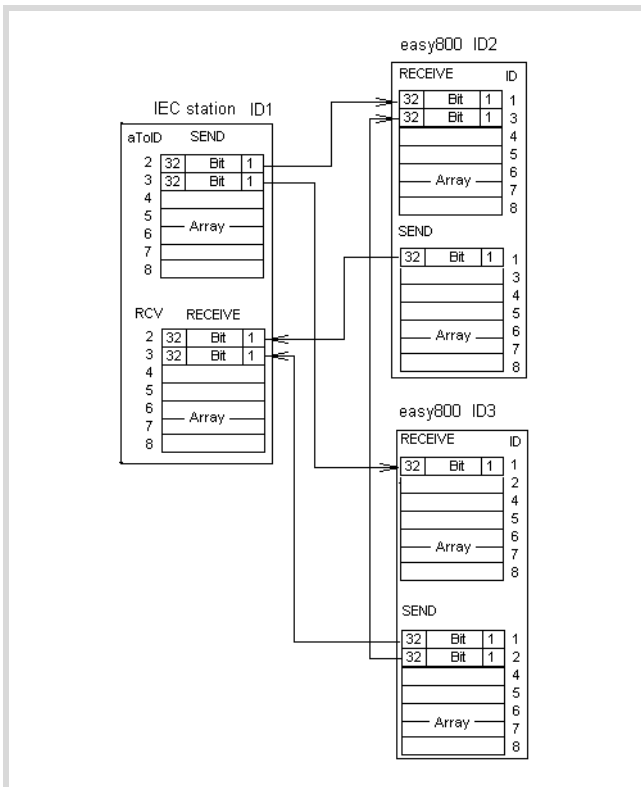


Figure 89: Overview of data blocks

1st connection:

The value "a" (32-bit) is transferred from the MFD4 (ID1) to the easy800 (ID2).

The following structure must be programmed in the MFD4:

```
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 (3hex) has to be sent from the easy800 PLC (ID3) to the (ID1). MFD4

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

The following structure variable must be programmed for scanning in the MFD4:

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

The variable INPVAL provides the value.

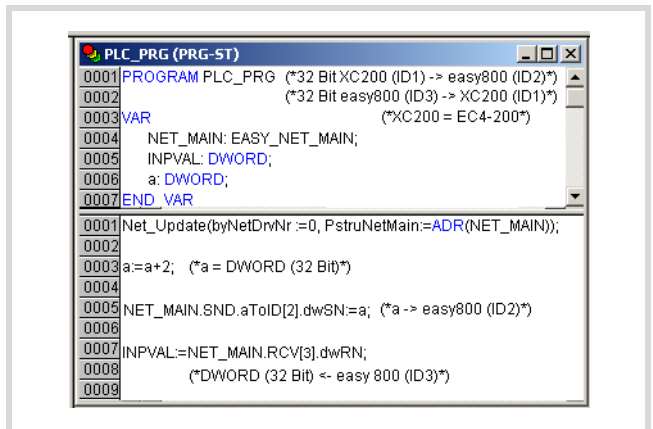


Figure 90: Program example for transferring bit data blocks (instead of XC200 (ID1) the MFD4 can also be used.)

The figure 91 shows the online display of the program from figure 90, from which the send and receive data originates.

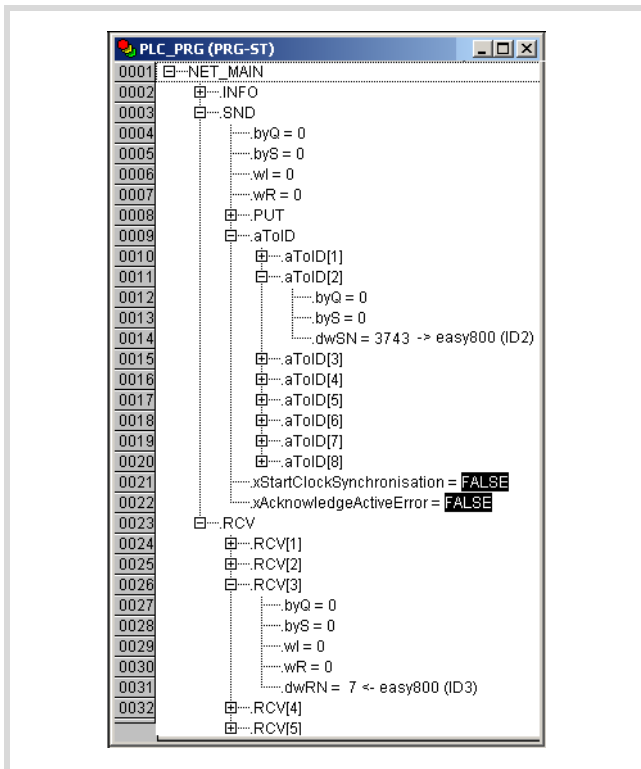


Figure 91: Online display of the program

**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 station puts a data unit (D word) on the network (PUT) which is identified with a module number (MN number for IEC stations, PT number for easy stations). The other stations can query the data unit by stating the ID of the sending station and the module number (GET).

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

On the MFD4, the NET\_UPDATE function must be called for the PUT operation and the elements of the structure variables EASY\_NET\_PUT/byModuleNumber and /dwData defined.

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.

**Example**

In the following example, the MFD4 (ID1) sends the data 3747<sub>hex</sub> (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<sub>hex</sub> is assigned the PT number 2.

The MFD4 scans this data unit. For this the following structure variable entries are required:

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

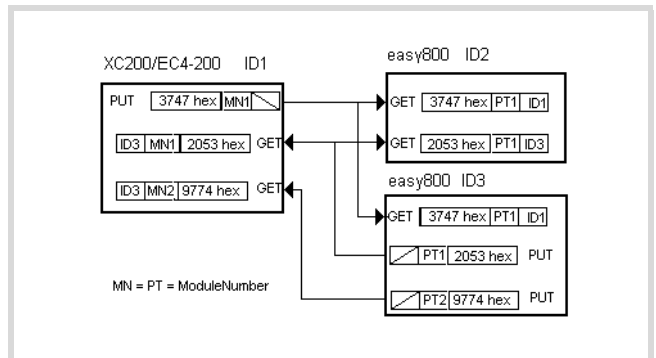


Figure 92: Transfer of data according to the PUT-GET principle (also applies to MFD4)

The following program sequence can be used in the in order to execute a PUT operation. MFD4

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.

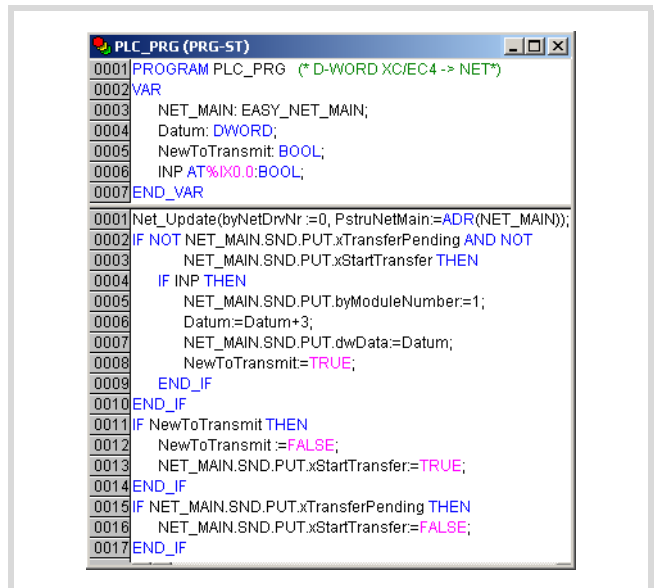


Figure 93: Transfer of data according to the PUT-GET principle (PUT)

The figure 94 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 94.

```

PLC_PRG (PRG-ST)
0001 PROGRAM PLC_PRG
0002 (*D-WORD easy800(ID3) -> XC200 (ID1)*)
0003 VAR
0004   Getdatastruct: EASY_NET_GET;
0005   xSuccess: BOOL;
0006   RecData: DWORD;
0007   Errorprog: BYTE;
0008 END_VAR
0009 Getdatastruct.byNET_ID:= 3;
0010 Getdatastruct.byModuleNumber:= 2;
0011
0012 xSuccess := Net_Get(
0013   byNetDrvNr:=0,
0014   pstruNetGet:=ADR(Getdatastruct));
0015 IF xSuccess THEN
0016   IF Getdatastruct.xNewDataReceived THEN
0017     RecData:=Getdatastruct.dwGET_Data;
0018   END_IF
0019 ELSE
0020   (*Error handling*)
0021   Errorprog:=Getdatastruct.byError;
0022 END_IF;
0023 END_PROGRAM

```

Figure 94: Data transfer using the PUT-GET principle, in this case: GET (the MFD4 can also be used instead of the XC200)

### Configuring with easy800 on the easyNet

Each station in easyNet must be configured: each station is assigned a network address, the ID, and parameters such as SEND IO which determine the behaviour of the station on the network.

This configuration is carried out in the programming software of the station concerned:

- easySoft for easy800
- easySoft-CoDeSys for XC200, MFD4 and EC4-200.

### Configuration in easySoft

easySoft supports you in the configuration of the network. As soon as you add another station to a stand-alone station, these stations are symbolically interconnected via the network. The first station is assigned the ID1. The IDs for the other stations and the parameters are set as follows:

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

All stations of the easy800 network, including XC200, MFD4 or EC4-200 controllers, i.e. all IEC stations, must be integrated in the easySoft configurator. The IEC stations are also configured in the easySoft-CoDeSys configurator.

- ▶ Create a circuit diagram for each easy800 and a program for the IEC stations.
- ▶ Connect the PC with an IEC station and download the program/circuit diagram including the configuration.

Other configuration options are described in section "Programming via easyNet (Routing)" on page 71.

### Configuration in easySoft-CoDeSys

Configure and program the MFD4 with the easySoft-CoDeSys programming software.

- ▶ For this activate the easy-NET setting in the Other Parameters tab of the configuration.
- ▶ Enter the ID number of the basic communication under the "easy-NET-ID". This ID number must be identical to the ID number that you assign to the MFD4 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.



## Controlled configuration via MFD4

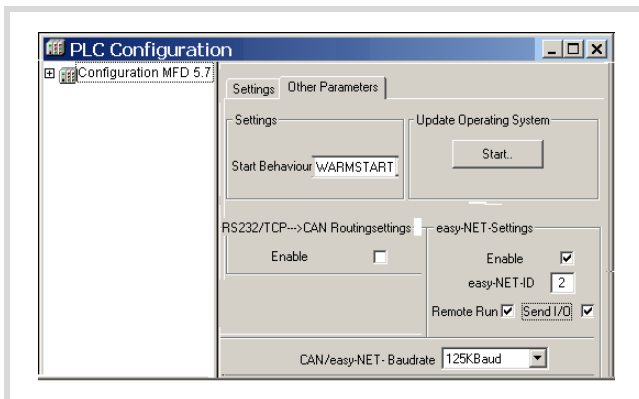


Figure 95: Configuration of MFD4 as easyNet station

You can use a MFD4 that was configured with ID 1 to configure other type easy800, EC4-200 or MFD4 controllers via the easyNet:

The configuration is carried out in 2 steps:

### 1. Parameter definition in the configurator

- ▶ Add a tick to the Enable box 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.

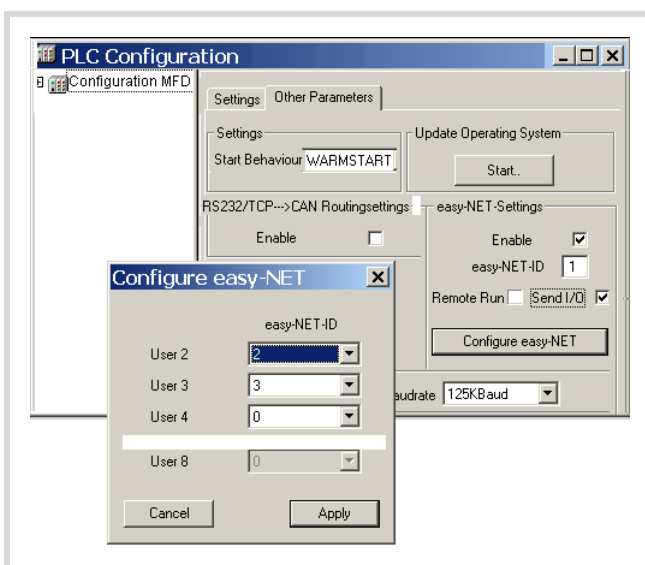


Figure 96: Parameterization

- ▶ Transfer the project to the station.

### 2. Transferring the parameters and assignment of IDs using the NET\_CONFIG function.

Calling this function will cause the parameters you have created in the configurator to be transferred to the individual PLCs. The function is contained in the library SysLibEasyNet.lib. The operation of the function and the integration into the user program are described in the manual "Description of the SysLibEasyNet.lib ...".

## Bus topology

A line topology with optional stub lines can be configured for the bus topology. The ends of the bus are terminated by bus termination resistors (120 Ohms).

If you integrate a MFD4 in the easy-Net, you can choose between two bus baud rates: 50kBit/s and 125kBit/s

The figure 97 shows two connection options.



### Attention!

The automatic configuration is not supported by the MFD4. If an MFD4 is present on the easyNet, the execution of an automatic configuration will cause an error which can block communication on the easyNet.

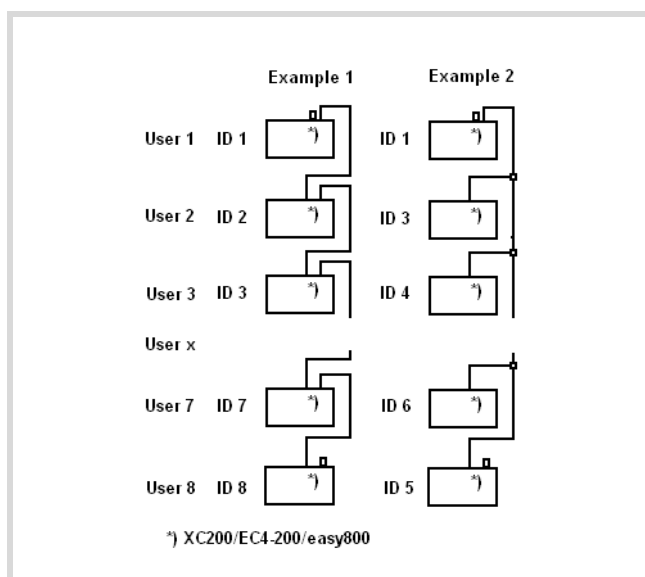


Figure 97: PLC connection options

Example 1: Physical location matches ID

Example 2: Physical location does not match ID

The station on the physical location 1 is always assigned ID1. The 7 other stations are connected to this station via the easyNet.

As the MFD4 is provided like the easy800 with two additional signal cables (SEL\_IN/OUT), it can be integrated in the easyNet like an easy800. For this it must have a minimum configuration, i.e. a project with an easyNet configuration must be loaded in the MFD4. This also enables all stations on the easyNet to be configured.

Table 32: Signal connection between the MFD4 and easy800

MFD4	↔	easy800
ECAN_H		ECAN_H
ECAN_L		ECAN_L
GND		GND
SEL_IN		SEL_OUT
SEL_OUT		SEL_IN

## 14 Programming via easyNet (Routing)

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

- Program download
- Online alterations
- Program test (Debugging)

Routing offers an advantage which makes it possible to access all routing capable stations on the easyNet from any station which is connected with the programming device. This makes it possible to operate remotely configured stations easily.

The data transfer with routing, however, is considerably slower than with a direct connection (serial or TCP/IP). This can be noticed with longer refresh times for visualisation elements (variables) or slower download speeds.

The following must be fulfilled in order to implement routing:

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

### Routing via MFD4

The routing options available are listed in figure 98 and figure 101.

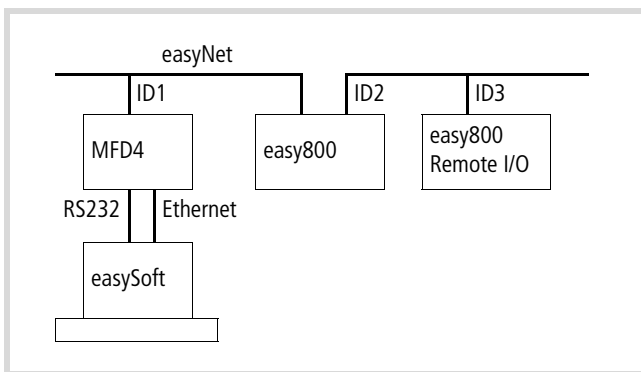


Figure 98: Routing via MFD4 (ID1) to easy 800

As shown in figure 98 you can access the easy800 (ID2) via the MFD4. For this the PC with the easySoft programming software must be connected to the MFD4 if the MFD4 already has a configuration in which it has an easyNet-ID (→ figure 95)..

This is how a connection is established between the PC and MFD4:

- ▶ Select the COMx or Ethernet interface in the EASY-SOFT «Communication → Connection» menu.

- Ethernet: Under the heading Ethernet Profiles click the Edit button and enter the IP address of the MFD4, e.g. 192.168.119.57, Port = 1200.
- COM...: Select an interface and the baud rate.

- ▶ Click the Online button.
- ▶ Under Device select an easy station with which you wish to communicate.

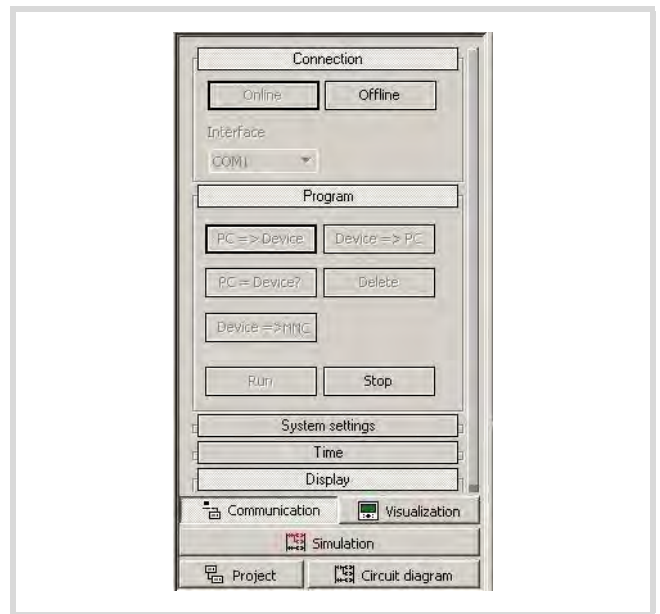


Figure 99: Program download in online mode

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 easy800 as the target station, 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	

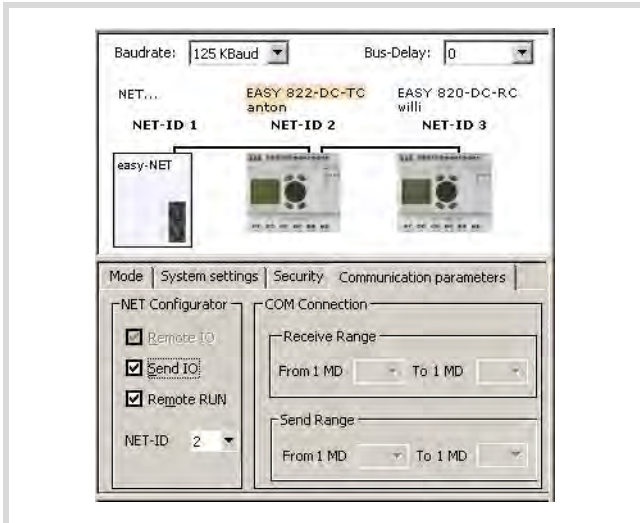


Figure 100: Communication parameters of easy800 (ID2)

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.

The routing options are also available if you configure an easy800 with the ID1 and the MFD4 with the ID2, as shown in figure 101. If you then select the MFD4 as target station, you can carry out the following operations with the station after the easySoft communication menu is called:

- Start/stop
- Set time
- Allow display of device information.

The PC with EASY-SOFT can be connected to any easy-NET-configured PLC.

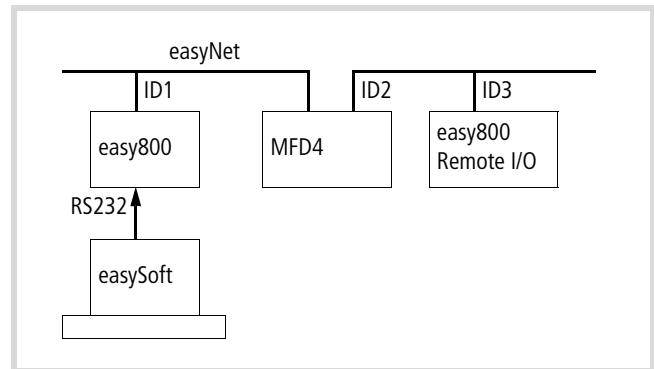


Figure 101: Routing via easy800 (ID1) to MFD4/easy800

## Appendix

### Characteristic of the Ethernet cable

Only use the intended cable type for wiring the Ethernet network. The cable must be at least category "CAT1" compatible. "CAT5" cables are suitable for data transfer rates of between 10 and 100 MBit/s.

Table 33: Characteristics of the Ethernet cable

	UTP <sup>1)</sup>	STP <sup>2)</sup>	SSTP <sup>3)</sup>
Transmission medium	Unshielded Twisted Pair	Shielded Twisted Pair	
Transmission speed	10 MBit/s, 100 MBit/s	10 MBit/s, 100 MBit/s	10 MBit/s, 100 MBit/s
Layout	Stranded every two cores,		
	without shield	with full shield	with full shield, each core pair is additionally shielded
Flexibility	Average	Average	Average
Shielding	None	Single	Double
Topology	Point-to-point	Point-to-point, line, star	
Maximum segment length	100 m	100 m	100 m

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

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

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

The installation guidelines for the (Ethernet) wiring are described in ISO/IEC 11801 and EN 50173.

### Properties of the CANopen cable

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

- Characteristic impedance 100 to 120  $\Omega$
- Capacitance < 60 pF/m

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

The table 35 lists standard parameters for the CANopen network with less than 64 CANopen stations (table complies with the stipulations of the ISO 11898).

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

Table 34: Cable cross-section, bus length and number of bus slaves conform to ISO 11898

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

n = number of connected bus users

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

Table 35: Standard parameters for CANopen network cable according to the ISO 11898

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

### Transparent mode: Text output via RS232 (example)

The example shows a text output via the RS232 interface of the CPU in transparent mode.

```

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

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

```

```

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

```

## Access to the CPU drives/memory card

### “SysLibFile.lib” library

The library SysLibFile allows you access to the file system of the MFD4 and the MMC. The library contains the functions:

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

→ Information on these functions is provided in the online documentation of the programming software under SysFile<Function>

#### Attention!

- The MFD4 must not be switched off when the files from the MultiMediaCard are opened.
- A voltage failure when a file is opened can destroy the memory card
- All the open files must be closed before switch off of the voltage.

### Mode for opening a file

#### “w” mode

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



#### Attention!

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

#### “r” mode

The “r” mode opens the file for reading. The file handle which is returned by the “SysFileOpen” function is invalid if this file does not exist. The value 0 is then shown.

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

#### “a” mode

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

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

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

### Examples of the “SysFile...” functions

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

#### Open in “r” mode

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

#### Open in “w” mode

```
OpenFile2 := SysFileOpen('\disk_mmc\MOELLER\MFD57\Project \File2','w');
```

#### Open in “a” mode

```
OpenFile3 := SysFileOpen('\disk_mmc\MOELLER\MFD57\Project \File3','a');
```

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

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

Dimensions

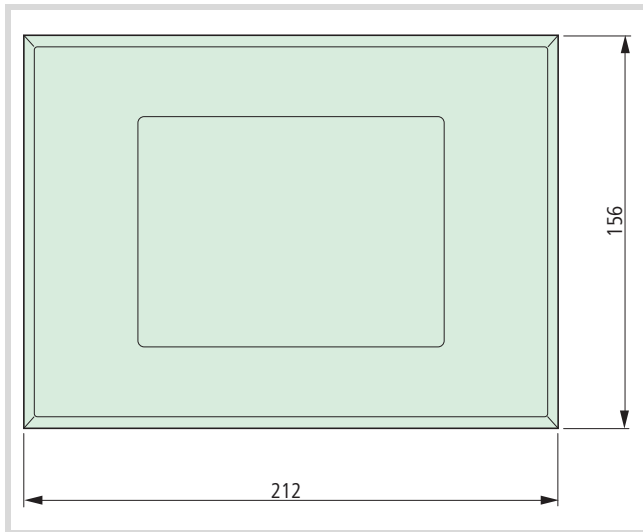


Figure 102: Front view

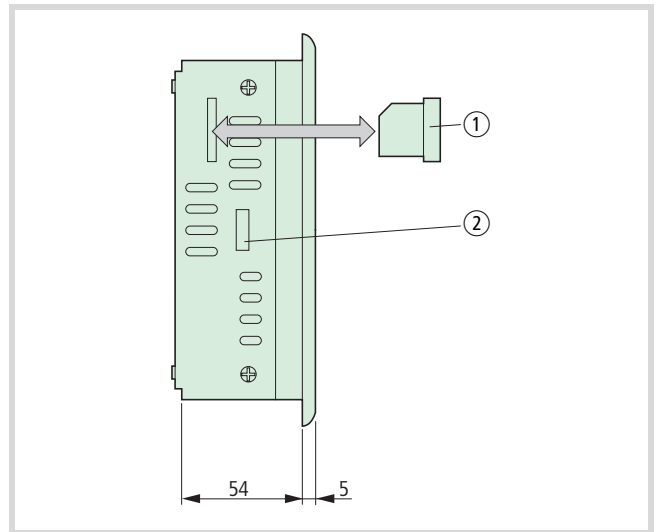


Figure 104: Side view

- ① Memory card
- ② Cutout for fixing bracket

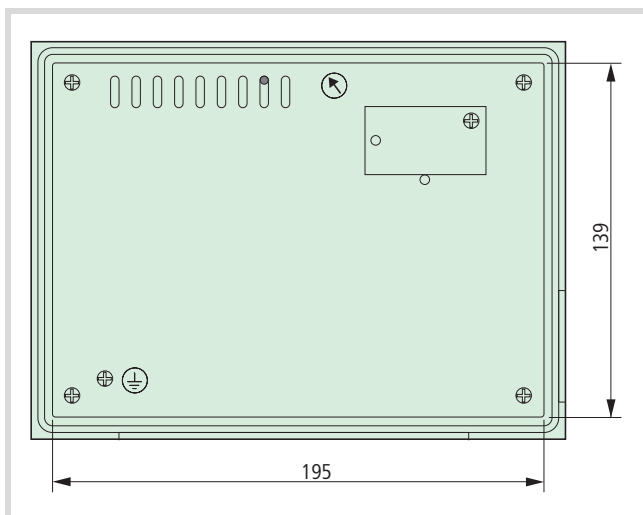


Figure 103: Rear view

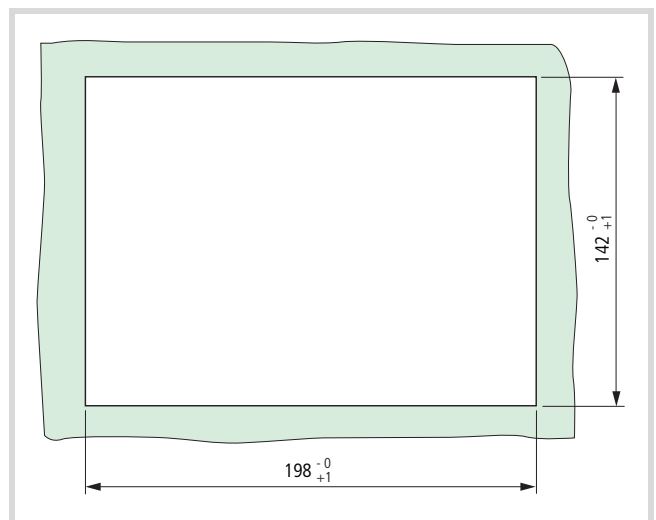


Figure 105: Mounting dimensions



## Technical data

			MFD4
<b>General</b>			
Standards and regulations			IEC/EN 61131-2 EN 50178
Ambient temperature			
with vertical mounting	°C		0 to 50
with diagonal mounting up to max. 45 °	°C		0 to 40
Storage temperature	°C		-20 to +70
Mounting position			vertical or diagonal mounting up to max. 45 °
Installation cut-out	mm		198 (-0+1) x 142 (-0+1)
Relative humidity, no condensation (IEC/EN 60068-2-30)	%		10 to 95
Air pressure (in operation)	hPa		795 to 1080
Vibration resistance			10 to 57 Hz ± 0.075 mm 57 to 150 Hz ± 1.0 g
Mechanical shock resistance			15 g/11 ms
Overvoltage category			II
Pollution degree			2
Degree of protection			
Front			IP65
Rear			IP20
Rated insulation voltage	V		500
Interference emission			EN 61000-6-4 Class A
Noise immunity			EN 61000-6-2
Battery (lifetime)			Lithium, 1/2AA(3.6V) , normally 5 years
Weight	kg		1.3
Dimensions (W x H x D)	mm		212 x 156 x 60
Terminals			Plug-in terminal block
Terminal capacity			
Screw terminals			
Flexible with ferrule	mm <sup>2</sup>		0.5 to 1.5
Solid	mm <sup>2</sup>		0.5 to 2.5
<b>Power supply</b>			
Input rated voltage	V DC		24
Admissible range	V DC		20.4 to 28.8
Mains failure bridging			
Duration of dip to IEC/EN 61131-2	ms		10
Current consumption	A		normally 0.3
Residual ripple	%		≤ 5
Overvoltage protection			Yes
Protection against polarity reversal			Yes
Inrush current	× I <sub>n</sub>		No limitation (limited only by upstream 24 V DC power supply unit)

			MFD4
<b>Display</b>			
Diagonal	Inches		5.7
Type			CSTN LCD
Resolution	Pixels		320 x 240
Display area	mm		118 x 89
Colours			256
Contrast ratio (Normally)			40 :1
Brightness (Normally)	cd/m2		160
Back-lighting			1 x CCFL
Lifespan	Operating hours		Normally 50000 h at 25°C ambient temperature (reduction of brightness to 50 %)
UV resistance			No
<b>Operating unit</b>			
Type			Touchscreen
Technology			analog resistive
<b>CPU</b>			
Microprocessor			RISC processor
Memory			
Program code	KByte		2048
Program data	KByte		512
Markers	KByte		16
Retain data	KByte		32
Persistent Data	KByte		32
Watchdog			Yes
RTC (Real-Time Clock)			Yes
Memory card			Yes, optional, order separately
<b>Interfaces</b>			
Ethernet interface			
Transfer rate	MBit/s		10/100
Connection types			RJ 45
Electrical isolation			Yes
RS 232 serial interface (without handshake line)			
Transfer rate	Bit/s		4800, 9600, 19200, 38400, 57600, 115200
Connection types			9-pole SUB-D plug (pins)
Electrical isolation			No
in Transparent mode			
Transfer rate	Bit/s		300, 600, 1 200, 2 400, 4 800, 9 600, 19 200, 38 400, 57 600, 115 200
Character formats			8E1, 8O1, 8N1, 8N2

			MFD4
CANopen/easyNet			
Electrical isolation			Yes
Bus terminating resistor			External
Connection types			9-pole SUB-D plug (pins)
CANopen operating mode			
Device profile			To DS 301 V4
PDO type			Asyn., cyc., acyc.
Stations		Number	maximum 126
Transfer rate		KBit/s	10/20/50/100/125/250/500/800/1000
Control mode easyNet			
Station		Number	max. 8
Transfer rate		KBit/s	50/125
<b>Electromagnetic compatibility</b>			
Noise immunity			
ESD (IEC/EN 61000-4-2)	Contact discharge		4 kV
	Air discharge		8 kV
RFI (IEC/EN 61000-4-3)	AM (80 %)	80 - 2000 MHz	10 V/m
GSM mobile (IEC/EN 61000-4-3)	AM (80 %)	800 - 960 MHz	10 V/m
Burst (IEC/EN 61000-4-4)	Network		2 kV
	Fieldbus (capacitive coupling)		1 kV
Surge (IEC/EN 61000-4-5)	Mains DC, unsymmetrical		1 kV
	Mains DC, symmetrical		0.5 kV
Cable conducted interference, induced by high frequency fields (previously: immunity to line-conducted interference) (IEC/EN 61000-4-6)			3 V
<b>Approvals and declarations</b>			
EMC			89/336/EEC
Explosions protection			ATEX Directive 94/9/EC Device group II, Zone 22, category 3D, T85 °C



## Index

<b>A</b>	Address overlaps	35		
	Alarm functions, libraries	30		
	Application screen	19		
<b>B</b>	Backup time, battery	7		
	Basic menu	17, 19		
	Battery	7		
	Baud rate	63		
	Baud rate, specifying/changing	37		
	Baudrate			
	Example for setting with routing	44		
	Boot project			
	Delete from memory card	28		
	Delete on memory card	26		
	Breakpoint	25		
	Brightness setting	18		
	Browser command "setlanguage"	61		
	Browser commands	57		
	Bus terminating resistors	15		
	Bus terminating resistors (easy-NET)	70		
	Bus topology (easy-NET)	70		
	Bus utilisation	63		
<b>C</b>	Cabling (engineering notes)	13		
	Calibrating, Display	17		
	CAN			
	Device parameters	45		
	Telegrams, receive/send from user program	9		
	CAN baud rate	45		
	CAN device parameters	45		
	CAN_Uilities.	50		
	CanDevice	45		
	CanMaster	45		
	CANopen			
	Bus interruption	9		
	Communication and configuration	9		
	Interface, Function	9		
	Interface, structure	15		
	Routing settings for master and device	45		
	Updae variables with multitasking	32		
	CANopen cable, properties	73		
	Changing screen, between application screen and basic menu	19		
	Character formats	47		
	CoDeSys gateway server	44		
	COLDSTART (start behaviour)	24		
	Command overview, Browser commands	57		
	Commissioning	25		
	Communication			
	Channel	37, 46		
	Channel (MFD4) setting	38		
	Error	38		
	Communication parameter setting			
	For MFD4 via browser command	38		
	For PC in programming software	37		
	STARTUP.INI	42		
	Communication, CANopen	9		
	Configuration			
	CANopen station	9		
	MFD4 on easyNET	68		
	MFD4, Visualisation task	29		
	CONFIGURE	19		
	Connecting the power supply	14		
	Connection cable	14		
	Contrast setting	18		
	Contrast/Light	18		
	Control panel layout	13		
	CPU			
	System utilisation	60		
	Creating a boot project	26		
	Cross-over cable	14		
<b>D</b>	Data definition (PUT-GET principle)	67		
	Data transfer via easy-NET	64		
	Data-saving	7		
	Debugging	25, 57		
	Delete			
	Boot project from memory card	26		
	Operating system from memory card	28		
	Deleting the Startup.INI file	42		
	Diagnostics			
	Via Browser commands	57		
	Dialog language, for errors and event lists	61		
	Dimensions	76		
	disk_mmc	8		
	disk_sys	8		
	Displaying device information (easy-NET)	72		
	Documentation, MFD4	5		
	Download of programs	26		
	Drives	75		
	Drives, CPU	8		
<b>E</b>	Earthing, MFD4	14		
	easyNet			
	Interface, Function	9		
	Interface, structure	15		
	Electromagnetic contamination	13		
	Engineering	13		
	Enter the IP address	38		
	Erase error list	58		
	Erase event list	58		
	Erase file	58		
	Erase password for online access	58		
	Error list	61		
	Ethernet cable, properties	73		

Ethernet interface	9, 14	L	Languag switchover, error and event list (browser commands)	61	
Parameter definition	18		Libraries		
Ethernet utilities	51		Display.lib	50	
Event controlled task	31		MFD57_UTIL.lib	50	
Event list	61		Library	75	
Example			Installation	49	
Transfer of bit data blocks	66		Lightning protection	14	
			Linear topology (easy-NET)	70	
<b>F</b>	Factory set	26	<b>M</b>	Manuals, MFD4	5
	Flash	8		Memory	
	Forcing, variables	25		Card	8
	Function blocks	49		Systems	8
	Functions	49		User program	7
	CAN_BUSLOAD	50		memory card	75
	NET_GET	64		MFD4	11
	NET_UPDATE	64		MMC memory card	8
	SysFile...	75		Monitoring time, task	33
	UT12_GetIPConfig	51		Mounting	11
	UT12_GetIPDNS	51		Mounting, MFD4	11
	UT12_GetIPWINS	52		Multi Media Card	8
	UT12_GetMacAddress	52		Multitasking	29, 32
	UT12_Reboot	55			
	UT12_SaveRegistry	55	<b>N</b>	NET-ID	68
	UT12_SetIPConfig	53		Network easyNet	63
	UT12_SetIPDNS	53		Node guarding function	9
	UT12_SetIPGateway	54		Node ID	44
	UT12_SetIPWINS	54		Node number	44
<b>G</b>	Gateway address setting	18	<b>O</b>	Operating system	
	getipconfig, browser command	39		Delete, from memory card	28
	getlanguage, Browser command	61		Transferring, from the PC into the PLC	27
	gettargetname, browser command	42		Operation, MFD4	21
				Output MAC address	52
<b>H</b>	HALT (start behaviour)	24		Outputs	
	Help, for browser comands	59		Referencing	25
			<b>P</b>	Parameter setting, MFD4 in easySoft-CoDeSys	69
<b>I</b>	Inductors	13		Parameters	
	Inputs			Changing	37
	Referencing	25		Setting via the display	17
	Interface	14		PC – MFD4 connection	
	Communication parameter setting	37		Setting in easySoft	71
	ETH232, assignment	9		PING response	39
	Interference factors	13		Plc	19
	Interruption, CANopen bus	9		Point-to-point connection	9
	Interval time	29		Power off/interruption of the power supply (behaviour)	24
	IP address	51		Program	
	Changing (Dynamic Name Server)	53		Call (task)	31
	Changing (Windows Name Server)	54		Download	71
	Display (Dynamic Name Server)	51		Loading	26
	Display (Windows Name Server)	52		Processing	29
	Scan/modify	39		Programming interface	9
	Setting	53		Protection against polarity reversal	14
	IP address setting	18			
	IP Gateway address				
	issue	51			

	PUT-GET principle	67			
<b>R</b>	Reboot	18	<b>T</b>	Task	
	Registry saving	55		Condition	29
	Remote I/O	63		Configuration	29
	Remote RUN	68		Creating/configuring	31
	removeprojfrommmc, Browser command	42		Monitoring	33
	removestartupini, browser command	42		POU creation, setting the program call	31
	Reset	17, 25		TCP/IP connection	43
	Restoring factory settings	26		Technical data	77
	Retentive variables	24		Test and commissioning	25
	RJ 45 interface	14		Text output, via the RS 232 interface	74
	Routing			Topology (easy-NET)	70
	via CANopen	43		Transparent mode	9, 47
	via easyNet	71		Example of text output via RS232	74
	RS232-interface	9		Trend functions, libraries	30
	Run, MFD4	19		Type (task condition)	29
<b>S</b>	Saving, Registry	55	<b>U</b>	USB stick	76
	SDRAM (working memory)	8		User program, memory values	7
	Send I/O	68		User task	29
	Sending/receiving user data	64	<b>V</b>	Ventilation	13
	setipconfig, browser command	39		Visualization	29
	Setting	18	<b>W</b>	WARMSTART (start behaviour)	24
	IPGateway address	54		Watchdog	33
	Setting the MFD4 – easyNet slave communication parameters	68		Web visualization	34
	Setting the time	18		Wiring (engineering notes)	13
	via easy-NET	72		Working memory	8
	Shielding	13			
	Single-cycle mode	25			
	Single-step mode	25			
	Starting				
	MFD4 via display	19			
	Starting/stopping via easy-NET	72			
	Start-up behaviour				
	Configuring with XSoft	24			
	STARTUP.INI file	41			
	Status display				
	Several CAN stations simultaneously (Routing)	46			
	Stop program	24			
	Subnet mask address				
	Setting	53			
	Subnet mask, Address setting	18			
	Subnetmask address	51			
	Suppressor circuitry for interference sources	13			
	Switch	15			
	System				
	Events	29, 32			
	Libraries	49			
	Memory	8			
	parameter predefined (via STARTUP.INI file)	41			
	Times	29			
	Utilisation, CPU	60			