Manual

WAGO-I/O-SYSTEM 750

WAGO-I/O-IPC-G2

758-870/000-112

CANopen Master, CODESYS 2.3

Version 2.4.0

WAGO Kontakttechnik GmbH & Co. KG

Hansastraße 27
D-32423 Minden

Phone: +49 (0) 571/8 87 – 0
Fax: +49 (0) 571/8 87 – 1 69

E-Mail: info@wago.com

Web: http://www.wago.com

Technical Support

Phone: +49 (0) 571/8 87 – 5 55
Fax: +49 (0) 571/8 87 – 85 55

E-Mail: support@wago.com

Every conceivable measure has been taken to ensure the accuracy and completeness of this documentation. However, as errors can never be fully excluded, we always appreciate any information or suggestions for improving the documentation.

E-Mail: documentation@wago.com

We wish to point out that the software and hardware terms as well as the trademarks of companies used and/or mentioned in the present manual are generally protected by trademark or patent.

Table of Contents

1 Notes about this Documentation

NOTICE

Read the operating instructions!
To avoid personal injury and property damage, only install and operating the I/O-IPC according to these operating instructions and the system description for 750-xxx. In addition, carefully follow the instructions in the "Safety" section.

NOTICE

Local regulations must be observed!
When integrating the 750 Series components in your machine or system, all currently applicable norms, regulations and guidelines shall be observed during all activities.

NOTICE

Power layout of the WAGO-I/O-SYSTEM 750!
In addition to these operating instructions, you will also need the system description "Project Planning Notes", which can be downloaded at www.wago.com. There, you can obtain important information including information on electrical isolation, system power and supply specifications.

Note

Keep this documentation!
The operating instructions are part of the product and shall be kept for the entire lifetime of the device. They shall be transferred to each subsequent owner or user of the device. Care must also be taken to ensure that any supplement to these instructions are included, if applicable.

1.1 Validity of these Operating Instructions

These operating instructions are only valid for WAGO-I/O-IPC-G2 of the WAGO-I/O-SYSTEM 750 with item number 758-870/000-112.

1.2 Copyright

1.3 Symbols

DANGER

Personal Injury!
Indicates a high-risk, imminently hazardous situation which, if not avoided, will result in death or serious injury.

DANGER

Personal Injury Caused by Electric Current!
Indicates a high-risk, imminently hazardous situation which, if not avoided, will result in death or serious injury.

WARNING

Personal Injury!
Indicates a moderate-risk, potentially hazardous situation which, if not avoided, could result in death or serious injury.

CAUTION

Personal Injury!
Indicates a low-risk, potentially hazardous situation which, if not avoided, may result in minor or moderate injury.

NOTICE

Damage to Property!
Indicates a potentially hazardous situation which, if not avoided, may result in damage to property.

NOTICE

Damage to Property Caused by Electrostatic Discharge (ESD)!
Indicates a potentially hazardous situation which, if not avoided, may result in damage to property.

Note

Important Note!
Indicates a potential malfunction which, if not avoided, however, will not result in damage to property.

Information

Additional Information:
Refers to additional information which is not an integral part of this documentation (e.g., the Internet).

1.4 Number Notation

Table 1: Number notation

Number code	Example	Note
Decimal	100	Normal notation
Hexadecimal	0x64	C notation
Binary	'100' '0110.0100'	In quotation marks, nibble separated with dots (.)

1.5 Font Conventions

Table 2: Font conventions

Font type	Indicates
italic	Names of paths and data files are marked in italic-type. e.g.: *C:\Programme\WAGO-I/O-CHECK*
Menu	Menu items are marked in bold letters. e.g.: **Save**
>	A greater-than sign between two names means the selection of a menu item from a menu. e.g.: **File** > **New**
Input	Designation of input or optional fields are marked in bold letters, e.g.: **Start of measurement range**
“Value”	Input or selective values are marked in inverted commas. e.g.: Enter the value “4 mA” under **Start of measurement range**.
[Button]	Pushbuttons in dialog boxes are marked with bold letters in square brackets. e.g.: **[Input]**
[Key]	Keys are marked with bold letters in square brackets. e.g.: **[F5]**

2 Important Notes

This section includes an overall summary of the most important safety requirements and notes that are mentioned in each individual section. To protect your health and prevent damage to devices as well, it is imperative to read and carefully follow the safety guidelines.

2.1 Legal Bases

2.1.1 Subject to Changes

WAGO Kontakttechnik GmbH & Co. KG reserves the right to provide for any alterations or modifications that serve to increase the efficiency of technical progress. WAGO Kontakttechnik GmbH & Co. KG owns all rights arising from the granting of patents or from the legal protection of utility patents. Third-party products are always mentioned without any reference to patent rights. Thus, the existence of such rights cannot be excluded.

2.1.2 Personnel Qualification

All sequences performed on the I/O-IPC shall only be carried out by electricians with appropriate knowledge in the field of automation technology. These specialists must be familiar with the current norms and guidelines for I/O-IPCs and automated environments.

All changes made to the control system shall always be performed by specialists with the appropriate knowledge of PLC programming. Detailed knowledge of Linux is required when making changes to this operating system.

2.1.3 Use in Compliance with Underlying Provisions

The I/O-IPC is used exclusively for controlling automated tasks and Linux applications. Therefore, it shall not be used for transmitting and processing security-related information; i.e., for example, emergency stops shall not be carried out on this equipment.

The I/O-IPC is a class A device and can cause radio interference in residential areas. If this is the case, you shall only use the I/O-IPC after measures have been taken to reduce emitted interference.

The I/O-IPC was developed for applications requiring IP20 protection.

The I/O-IPC shall only be used as a unit or in combination with the 750/753 Series I/O Modules.

Up to 64 750/753 series I/O modules may be connected to the I/O-IPC. The use of up to 250 I/O modules is possible with the WAGO internal data bus extension (optional). When doing so, the following system parameters shall be observed:

- The total length of the I/O modules behind the I/O-IPC, including the end module, shall not exceed a maximum of 780mm.
- The maximum size of the process image for the input and output data shall not exceed 500 bytes in each case.

Applications other than those described in these instructions are not permitted.

2.1.4 Technical Condition of Specified Devices

The devices to be supplied ex works are equipped with hardware and software configurations, which meet the individual application requirements. WAGO Kontakttechnik GmbH & Co. KG will be exempted from any liability in case of changes in hardware or software as well as to non-compliant usage of devices.

Please send your request for modified and new hardware or software configurations directly to WAGO Kontakttechnik GmbH & Co. KG.

2.2 Safety Advice (Precautions)

To avoid **personal injury**, read and observe the following safety information before using the I/O-IPC and I/O terminals of the 750/753 series.

DANGER

Electric voltage!
Only operate the I/O-IPC with 24VDC PELV- (Protective Extra Low Voltage) or SELV (Safety Extra Low Voltage) power sources. There is a danger of electric shock if this precaution is not observed.

- High temperatures may occur on the underside of the I/O-IPC during operation. If the I/O-IPC has been in operation, allow it to cool off before moving it.
- Observe the appropriate accident prevention regulations for your system during mounting, start-up, maintenance, and repairs.
- For each activity, observe the corresponding personnel qualification in Section "Personnel Qualification".
- Read and observe the operating instructions for the WAGO I/O modules that you connect to the I/O-IPC.

To avoid property damage, read and observe the following information:

- The 750 Series components shall not come in contact with substances having seeping and insulating properties. Otherwise, additional measures shall be taken for the devices, such as installation of an enclosure that is resistant to the above-mentioned substance properties.
- The 750 Series components contain electronic elements that may be destroyed during electrostatic discharge. When handling these components, make sure that everything close by is well grounded (persons, work station and packaging). Do not touch any conducting parts; e.g., data contacts and printed circuit boards.
- To achieve a high resistance to interference from electromagnetic emissions, maintain a sufficient distance from electromagnetic sources of interference (e.g., frequency converters, motors). Use shielded conductors only at the required sites. Observe the corresponding norms for EMC-compatible installations as well.
- Replace defective or damaged 750 Series components since malfunctions may otherwise occur.
- When laying any lines, take care that you do not place them within shear range of movable system parts.

2.3 Safety Equipment

All 750 Series components correspond to protection class IP 20. This includes complete protection against accidental contact with electrical voltage and currents.

2.4 Notes on Operation

When integrating the 750 Series components in your machine or system, all the currently applicable norms, regulations and guidelines shall be observed during all activities. The emergency stop equipment shall remain effective in all operating modes of the system and machine.

For protection from electromagnetic interferences

- Connect your system to protective earth (PE) and
- Ensure that the routing and installation of the supply and signal lines are correct.

The following elements for 24V supply shall be present:

- Outer lightning protection on buildings
- Inner lightning protection of supply lines and signal lines
- Safe electrical separation of 24VDC low voltage through PELV (Protective Extra Low Voltage) or SELV (Safety Extra Low Voltage) voltage sources

2.5 Special Use Conditions for ETHERNET Devices

If not otherwise specified, ETHERNET devices are intended for use on local networks. Please note the following when using ETHERNET devices in your system:

- Do not connect control components and control networks to an open network such as the Internet or an office network. WAGO recommends putting control components and control networks behind a firewall.
- Limit physical and electronic access to all automation components to authorized personnel only.
- Change the default passwords before first use! This will reduce the risk of unauthorized access to your system.
- Regularly change the passwords used! This will reduce the risk of unauthorized access to your system.

- If remote access to control components and control networks is required, use a Virtual Private Network (VPN).
- Regularly perform threat analyses. You can check whether the measures taken meet your security requirements.
- Use “defense-in-depth” mechanisms in your system's security configuration to restrict the access to and control of individual products and networks.

3 Scope of Delivery

The following components are included in the scope of delivery of the I/O-IPC:

- 750-602 Power Supply Module
  (No longer included in the scope of delivery as of HW Version 11)
- Socket for power supply connection
- protective caps

4 Device Description

The I/O-IPC automation device is an industrial PC that can handle the control tasks of a PLC. It is suitable for mounting on a DIN rail and distinguishes itself with its various interfaces.

You can connect all available I/O modules of the WAGO-I/O-SYSTEM 750/753 to the I/O-IPC. This allows any analog and digital signals from the automation environment to be internally processed or to be made available to other devices through one of the available interfaces.

Transfer rates of 10 Mbit/s or 100 Mbit/s are possible via two independent ETHERNET interfaces in both half-duplex and full-duplex operation.

For data exchange, there are implemented MODBUS/TCP, -UDP and -RTU servers available, such as a CANopen master.

Automation tasks can be executed in all IEC 61131-3-compatible languages with the programming system CODESYS 2.3 (WAGO-I/O-PRO CAA). The implementation of the CODESYS task processing is optimized with real-time extensions in order to provide maximal performance for automation tasks. For visualization, CODESYS target visualization and web visualization are also available in addition to the development environment.

The fieldbus can be configured using the PLC configuration function of CODESYS 2.3.

The I/O-IPC provides 128 MB of program and data memory and 128 kB of non-volatile memory.

Both clients and servers for TCP or UDP can be programmed via function blocks.

An internal server is available for Web-based applications. Information regarding the configuration and status of the I/O-IPC, among other things, is already stored as dynamic HTML pages in the I/O-IPC and can be read via Internet browser. In addition, you can also save your own HTML pages or call up programs directly via an implemented file system.

The firmware installed at delivery is based on Linux with special real-time extensions of the RT-Preempt patch. In addition, various user programs are already installed on the I/O-IPC. These include, for example:

- a SNMP server/client
- a Telnet server
- a FTP server (supports simultaneously two ftp connections)
- an NTP client
- a BootP and DHCP daemon

- the CODESYS runtime environment and other different help programs

Note

Memory card is not included in the scope of delivery!
Note, the I/O-IPC is delivered without memory card.
To use a memory card, you must order one separately. The I/O-IPC can also be operated without memory card expansion, the use of a memory card is optional.

Note

Only use recommended memory cards!
Use only the CF memory card available from WAGO (order no. 758-879/000-000) since it is suitable for industrial applications under difficult environmental conditions and for use in the I/O-IPC.
The compatibility to other storage media available in trade cannot be ensured.

4.1 Overview of Physical Interfaces

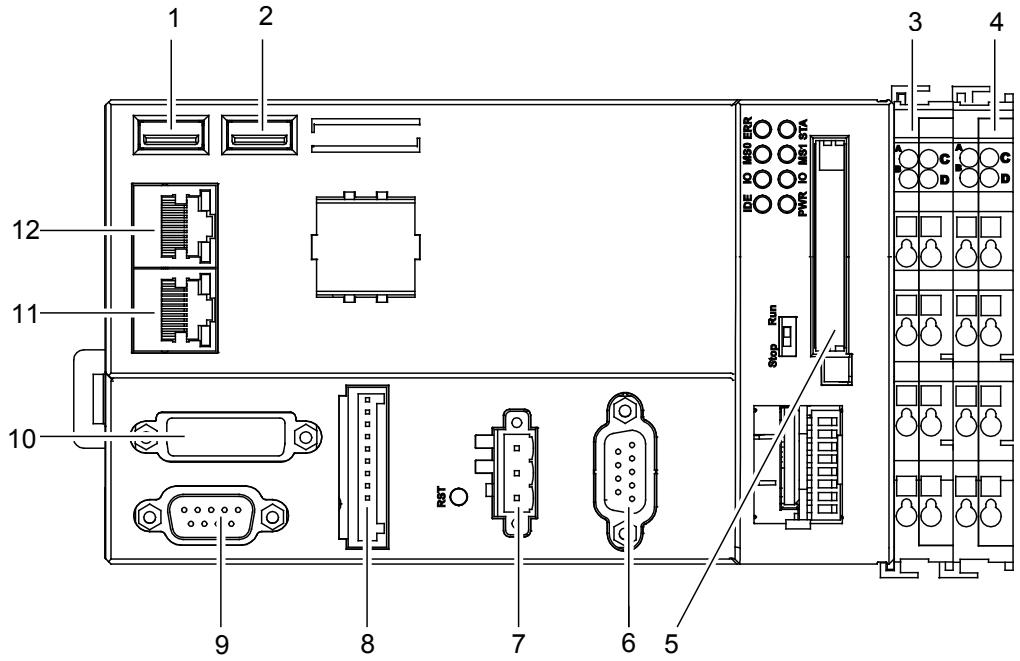


Figure 1: Overview of physical physical interfaces

NOTICE

Internal data bus interface

The internal data bus interface (item 3) may not be disassembled. This I/O module is part of the I/O IPC.

Table 3: Legend for figure "Overview of physical interfaces"

Position	Description	Function
1	USB interface (X10), Type A	To connect USB devices such as, keyboards, USB memories, etc.
2	USB interface (X11), Type A	
3	I/O module interface	Data exchange over the internal data bus.
4	750-602 Power Supply Module (No longer included in the scope of delivery as of HW Version 11)	Infeed of the field supply (sensors/actuators).
5	Slot for CF card	Slot for CF cards, type I and II.
6	CANopen interface (X3), 9-pole D-sub plug	Interface to connect the I/O-IPC to a CAN network.

7	Interface for electronic supply (X4)	Power supply for the 24 V electronic supply of the I/O-IPC. The power supply is protected against reverse polarity.
8	Integrated input and output (X5), 12-pole D-sub socket	Interface for connecting direct digital signal generators.
9	Serial RS-232 interface (X6), 9-pole D-sub plug	Physical connection to the Linux console, MODBUS/RTU, IO-Check or CODESYS. This interface can be configured via Web-based management or Linux console.
10	DVI interface (X7), D-sub socket, 24+5	To connect a digital or analog monitor. A "DVI to VGA" adapter is required to connect a monitor via a VGA cable.
11	ETHERNET interface (X8), RJ-45	Interfaces to connect the I/O-IPC to a ETHERNET Network.
12	ETHERNET interface (X9), RJ-45	

4.2 Display Elements

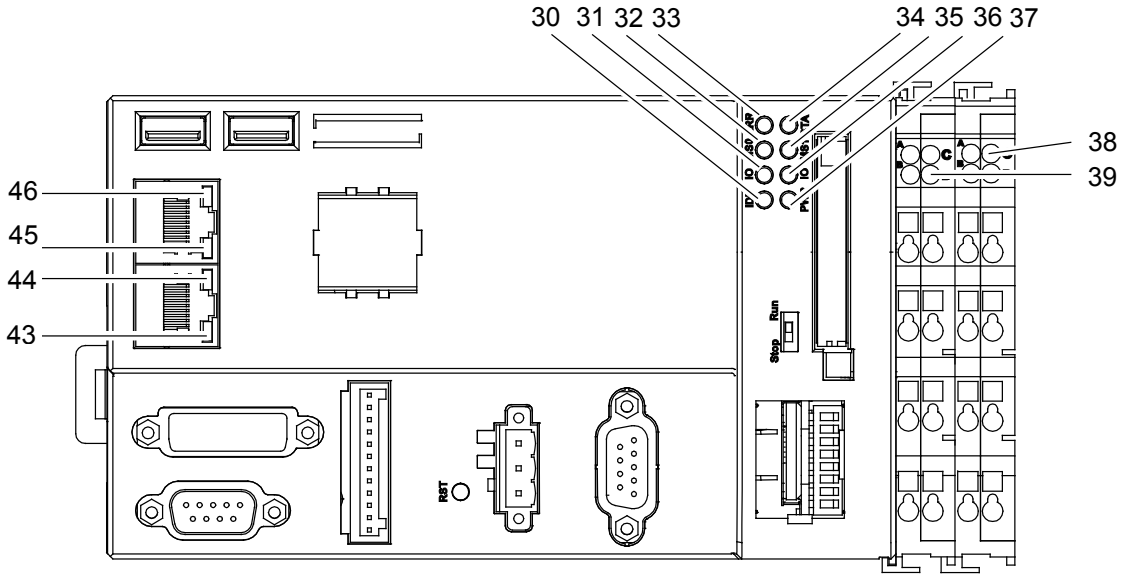


Figure 2: LEDs

Table 4: Legend for figure "Display elements"

Position	LED	Color	Explanation
30	IDE	red	Status for the activity of the internal flash memory or the inserted CF card.
31	IO	red	Display of malfunction messages by a blink code.
32	MS0	red	Module state
33	ERR	red	Fieldbus Status
34	STA	green	
35	MS1	green	Module state
36	IO	green	Internal data bus state
37	PWR	green	Status of the supply voltage.
38	LED C	green/off	24 V supply voltage available via 750-602 Power Supply Module (No longer included in the scope of delivery as of HW Version 11).
39	LED D	off	The I/O module interface LED is not used.
43	ACT	yellow/off	Status of the X8 ETHERNET interface for data communication.
44	LNK	green/off	Status of the X8 ETHERNET interface network connection
45	ACT	yellow/off	Status of the X9 ETHERNET interface for data communication.
46	Speed	green/off	Status of the X8 ETHERNET interface network connection

Detailed information regarding the LEDs can be found starting in section "LED Signaling".

4.3 Operating Elements

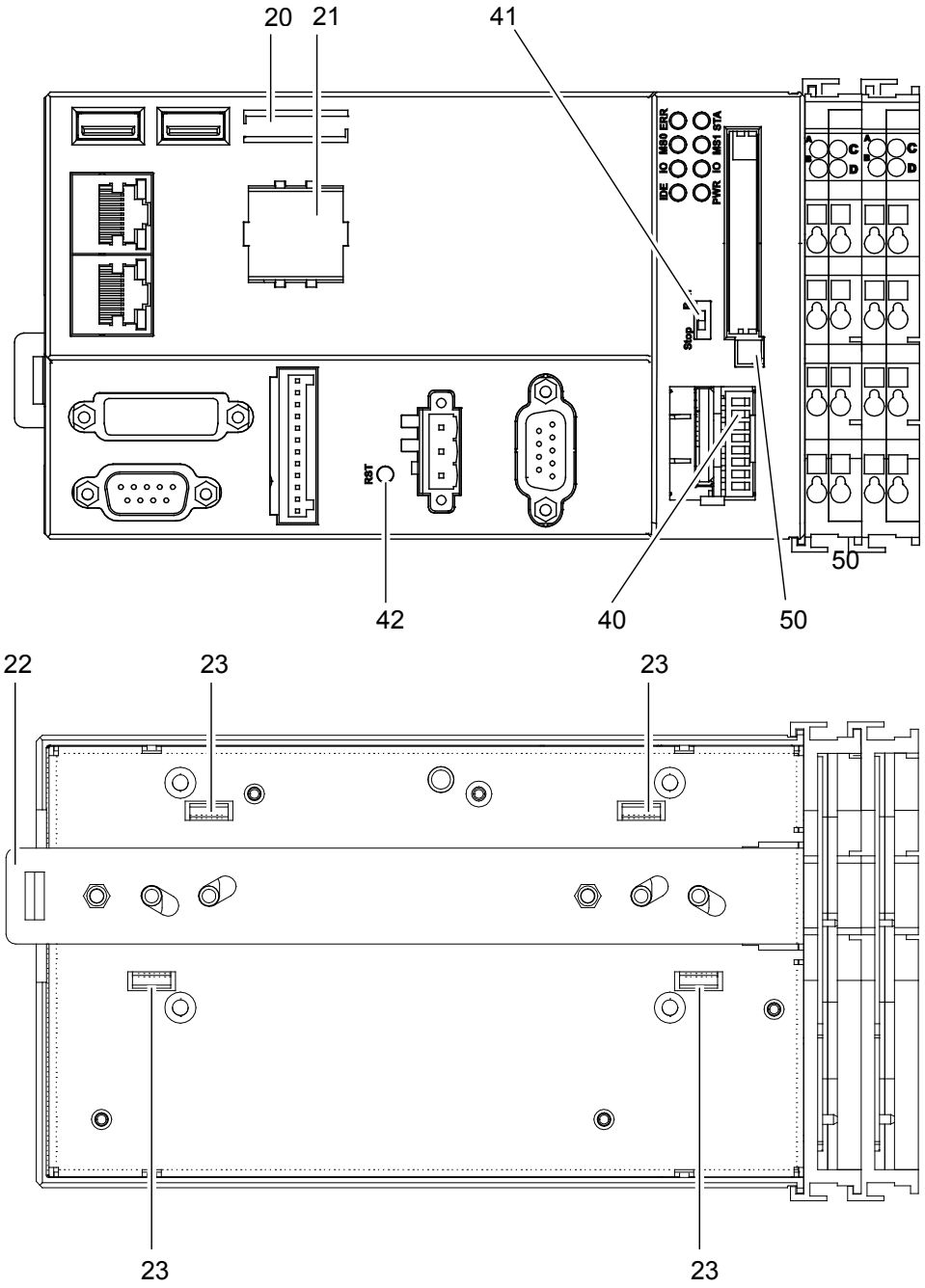


Figure 3: Operating Elements

Table 5: Legend for figure "Operating elements"

Position	Operating element	Explanation
20	Marker strips	For 4-character identification of the I/O-IPC via WAGO Miniature WSB Quick marking system.
21	Marking field	-
22	Unlocking mechanism	For loosening the I/O-IPC from a grounded mounting rail.
23	Mounting rail fastener	For securing the I/O-IPC to the mounting rail.
40	DIP switch	For setting the fieldbus address.
41	Run/Stop switch	**Run:** Automatic start of the boot project (CODESYS) when starting the I/O-IPC or when starting the PLC program. **Stop:** Stops the PLC program.
42	Reset button	For re-starting the I/O-IPC.
50	Reject button	To remove the CF card, press the reject button

4.4 Battery

The 3 V battery (52) of type CR2032 (Li/MnO_2, ca. 225 mAh) is in the battery compartment (51).

The battery continues to provide power for the real-time clock (RTC) and volatile storage (SRAM) with the CODESYS retain variables if there is a power failure. You can obtain additional information in Section "Maintenance".

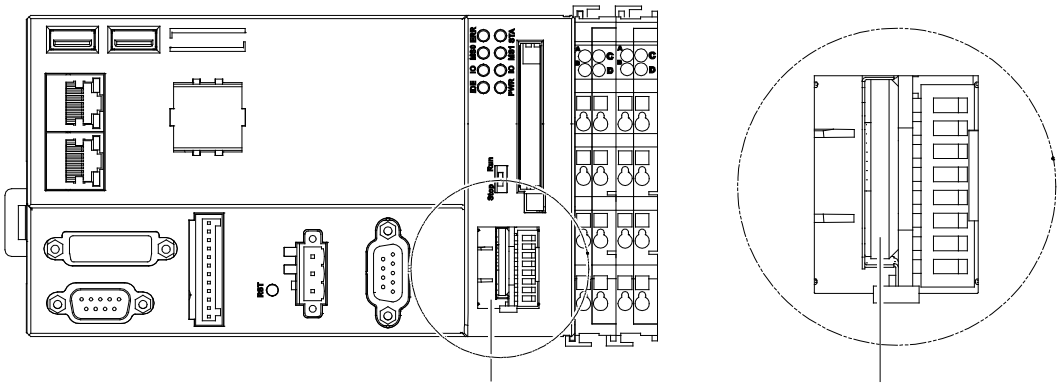


Figure 4: Battery

4.5 Labeling

There is a label on the side of the I/O-IPC with the following information:

- A: Name of the I/O-IPC
- B: Item number of the I/O-IPC
- C: Serial number of the device
- D: Hardware status at the time of delivery
- E: Firmware status at the time of delivery
- F, G: MAC addresses for X8 and X9 ETHERNET interfaces.
  The MAC addresses are used to identify and address ETHERNET devices.
  Each MAC address is unique worldwide.
- H: Licenses for the I/O-IPC
- I: Manufacturer

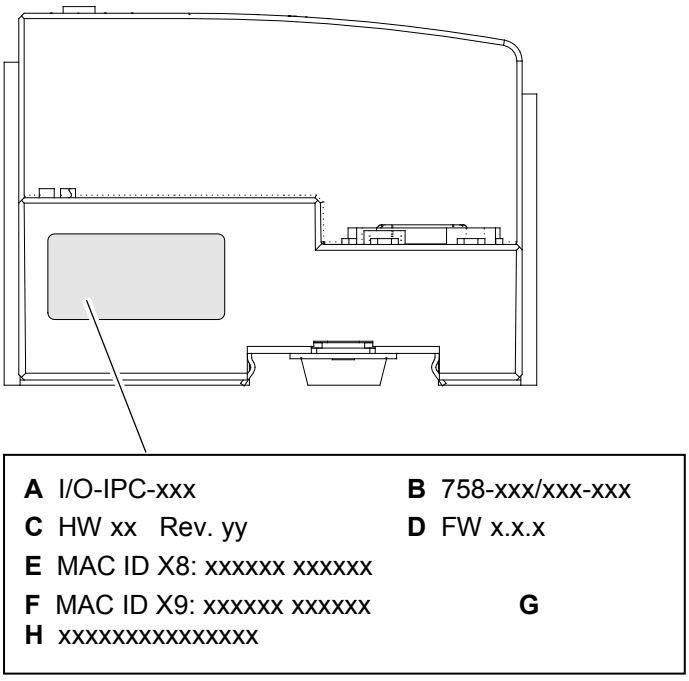


Figure 5: Lateral marking on the I/O-IPC

4.6 Technical Data

4.6.1 Device Data

Table 6: Technical data – Device Data

Width	172 mm
Height	65 mm from upper edge of DIN 35 rail
Length	100 mm
Housing material	Plastics
Montageart	TS 35
Weight	Approx. 550 g

4.6.2 System Data

Table 7: Technical data – System data

Number of I/O modules that can be connected to the I/O-IPC	64 pieces With internal data bus extension (optional) up to 250 pieces.
Input process image, max.	500 bytes
Output process image, max.	500 bytes
CPU	Geode SC 1200, 266 MHz
Bios	Insyde
Display resolution	640x480, 16 bit 800x600, 16 bit 1024x768, 16 bit
Memory expansion	CompactFlash, Type I/II
Operating system	Linux 2.6 with RT-Preempt patch
Main memory (RAM)	128 MB
Internal memory (flash)	128 MB
Non-volatile memory (retain)	127 kB
Graphic	DVI, 1024x768; LCD/Panel link
User-specific, real-time support	128 kB PLC-SRAM with battery backup; NMI timer

4.6.3 Supply

Table 8: Technical data – Supply

Voltage supply	DC 24 V (-25 % … + 30 %)
Input current	770 mA
Total current for I/O modules	1000 mA

4.6.4 Communication

Table 9: Technical Data – Communication

LAN	2 x 10Base-T/100Base-TX
I/O interface, serial	A 9-pole D-sub connector in accordance with EIA RS-232
I/O interface, USB	Two USB interfaces in accordance with specification 2.0
I/O interface, USB	Two USB interfaces in accordance with specification 1.1
Fieldbus	CANopen, Master

4.6.5 Protection and Security

Table 10: Technical data – Protection and Security

Degree of protection	IP20 in accordance with EN 60529

4.6.6 Runtime System

Table 11: Technical data – Runtime System

Programming	CODESYS 2.3 (WAGO-I/O-PRO CAA)
IEC 61131-3	IL, LD, FBD, ST, FC

4.6.7 Environmental Requirements

Table 12: Technical Data – Environmental Requirements

Vibration resistance	In accordance with IEC 60068-2-6
Humidity	5 – 95 % without condensation
Storage temperature	-10 °C … +85 °C
Operating temperature	0 °C … +55 °C

4.6.8 Wire Connection

Table 13: Technical Data – Wire Connection

Wire connection	CAGE CLAMP®
Cross section	0.08 mm^2 – 2.5 mm^2/AWG 28 – 14
Stripped lengths	8 – 9 mm/0.33 Inch

4.7 Standards and Guidelines

4.7.1 Electromagnetic Compatibility

Table 14: Technical data – Electromagnetic Compatibility

Thresholds for emitted interference and operation	In accordance with DIN EN 61000-6-4
Thresholds for interference resistance and operation	In accordance with DIN EN 61000-6-2

4.8 Approvals

Information

More Information about Approvals
Detailed references to the approvals are listed in the document "Overview Approvals **WAGO-I/O-SYSTEM 750**", which you can find via the internet under: www.wago.com → Documentation → WAGO-I/O-SYSTEM 750 → System Description.

The following approvals have been granted to 758-870/000-112 I/O-IPC:758-870/000-112

CE	Conformity Marking	
cULus	cULus	UL508

The following ship approvals have been granted to 758-870/000-112 I/O-IPC 758-870/000-112

ABS	ABS (American Bureau of Shipping)	
BV	BV (Bureau Veritas)	
DNV	DNV (Det Norske Veritas)	Class B
GL	GL (Germanischer Lloyd)	Cat. A, B, C, D (EMC 1)
RINA	RINA (Registro Italiano Navale)	

5 Description of I/O-IPC Interfaces

5.1 ETHERNET Interfaces (X8, X9)

Both ETHERNET interfaces of type RJ-45 are based on the 10/100 BASE-T transmission standard. These enable, depending on the ETHERNET network used, data exchange at a transmission rate of 10 Mbit/s or 100 Mbit/s in half-duplex and full duplex operation respectively.

Note

A bandwidth of 100 Mbit for the data transmission via ETHERNET interface X9 can only be guaranteed if the cable length does not exceed 30 m. If the cable is longer, data transmission will be limited, or may not occur at all. It is possible to use a 100 m-long cable with a bandwidth of 10 Mbit.

The "ACT" and "LNK" LEDs of the two ETHERNET interfaces indicate the current operating status:

Table 15: ACT and LNK LED

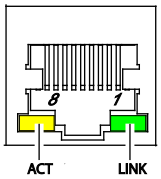
LED	Color/Status	Cause
ACT	off	No data exchange
	yellow flashing	Data exchange is taking place.
LNK	off	No link to the ETHERNET network available
	green	Link to the ETHERNET network available

You have the following possibilities for connecting the I/O-IPC to a PC using ETHERNET:

- Directly, with the aid of a crossover cable
- Using a switch or hub in connection with a patch cable

The following table provides information on the ETHERNET interface pin assignments:

Table 16: ETHERNET Interfaces: Pin Assignments

Connector	Pin	Description
 Figure 6: RJ-45 Geode	1	TD+
	2	TD-
	3	RD+
	4	Not assigned
	5	Not assigned
	6	RD-
	7	Not assigned
	8	Not assigned

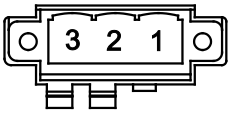
5.2 Interface for Electronic Power Supply (X4)

Feed the electronic power supply for the I/O-IPC and for the I/O modules connected to the internal data bus in through this interface.

DANGER

Electric voltage!
Only operate the I/O-IPC with 24VDC PELV- (Protective Extra Low Voltage) or SELV (Safety Extra Low Voltage) power sources. There is a danger of electric shock if this precaution is not observed.

Table 17: Interface for Electronic Power Supply: Pin Assignments

Connector	Pin	Description
 Figure 7: Electronic Power Supply (X4)	1	V_IN (+)
	2	GND (-)
	3	Shield (optional)

5.3 CANopen Interface (X3)

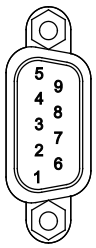
The fieldbus is used for communication between the I/O-IPC and the connected CANopen fieldbus couplers (slaves).

NOTICE

Open interface!
If this interface is not needed, it shall be closed by the protective cap provided with delivery in order to prevent any possible damages caused by electrostatic discharge.

The following table provides information on the CANopen interface pin assignments.

Table 18: CANopen Interface: Pin Assignments

Connector	Pin	Description
 Figure 8: Interface X3	1	NC
	2	CAN-
	3	CAN_GND
	4	NC
	5	NC
	6	NC
	7	CAN+
	8	NC
	9	CAN_+5V

5.4 Integrated Inputs and Outputs (X5)

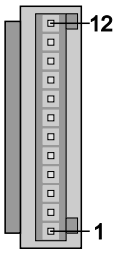
The 12-pole D-sub connector provides two integrated digital inputs and two outputs. These are used to connect sensors or actuators that are to be used independently of the internal data bus.

Note

Integrated inputs and outputs for use only shielded cables!
Remember when using the integrated inputs and outputs that these do not fulfill the requirements of the IEC-61131-2. Only the connection of screened lines is permitted.

The following table provides information on the configuration of the integrated inputs and outputs:

Table 19: Digital Inputs and Outputs: Pin Assignments

Connector	Pin	Description
 Figure 9: Connector 12-pole D-sub-socket	1	DIN0
	2	~DIN0
	3	DIN1
	4	~DIN1
	5	DOUT0
	6	~DOUT0
	7	DOUT1
	8	~DOUT1
	9	WDOG
	10	REL_NC
	11	REL_NO
	12	SHIELD

Digital Inputs

The two digital inputs are independent of the internal data bus. This means that digital signals are processed even if the internal data bus breaks down.

Voltage range	Low: -3 V … +5 V High: +11 V … +30 V (+24 V standard)
Max. current per channel	5 mA
Channels	2
Input impedance	Min. 1.5 kOhm Max. 6 kOhm at 30 V
Characteristics	Optocoupler, 2 kV Low-pass filter, 10 kHz Current limitation Surge protection Reverse voltage protection

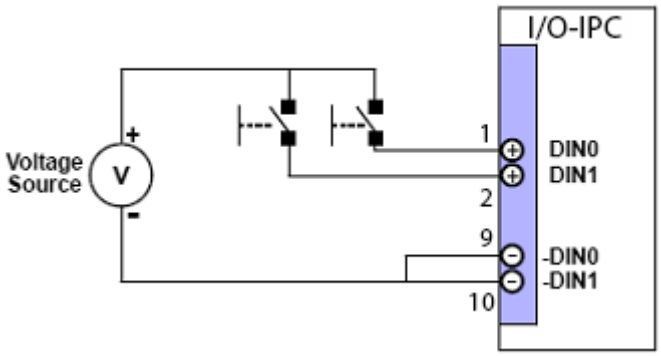


Figure 10: Connection of integrated inputs

Digital Outputs

NOTICE

Highest current carrying capacity of the output channels is 0.1A!
Always observe the maximum current carrying capacity of 0.1A for the digital output channels. An increase in the current leads to overheating of the output driver and damage to the I/O-IPC.

The two digital outputs are independent of the internal data bus. This means that digital signals are processed even if the internal data bus breaks down.

External power supply, max.	24 V DC
Voltage range	Depends on external circuit
Max. current per channel	0.1 A (typical)
Channels	2
Characteristics	Optocoupler 2 kV

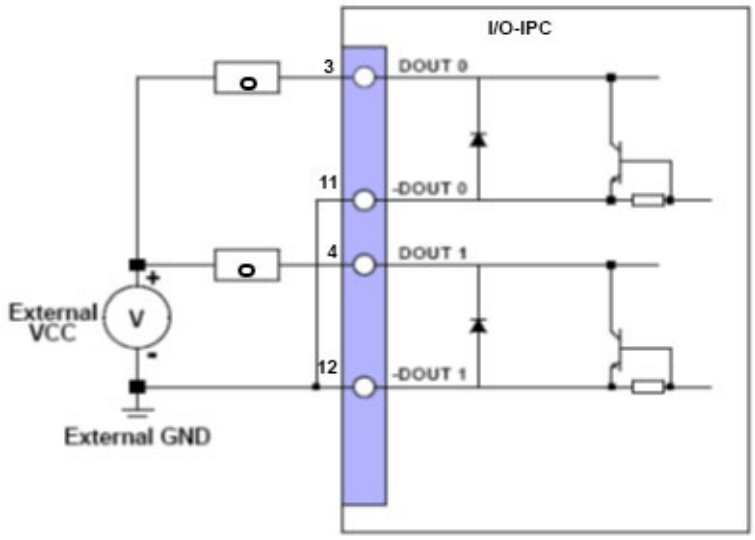


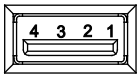
Figure 11: Connection of integrated outputs

5.5 USB Interfaces (X10, X11)

The USB interfaces are used to connect USB devices such as USB memory. If the connected USB device is not used, you can remove it at any time. If a USB memory is connected, it is important to remember to close any open files before removing the USB memory.

The following table provides information on the USB interface pin assignments:

Table 20: USB Interfaces: Pin Assignments

Connector	Pin	Description
 Figure 12: USB Interface	1	USB_VCC1
	2	USB_N1
	3	USB_P1
	4	USB_GND

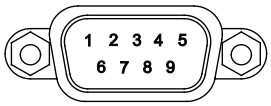
NOTICE

Operation of an external hard drive is not possible!
It is not possible to operate an external hard drive on the I/O IPC from a USB connection.
It cannot be guaranteed that the USB connection can supply the external hard drive with the required power.
Insufficient power can lead to internal device damage.

5.6 RS-232 Serial Interface (X6)

The following table provides information on the RS-232 interface pin assignments:

Table 21: RS-232 Interface: Pin Assignments

Connector	Pin	Description
 Figure 13: RS-232 Serial Interface	1	DCD1
	2	RXD1
	3	TXD1
	4	DTR1
	5	GND
	6	DSR1
	7	RTS1
	8	CTS1
	9	RI1

NOTICE

Open interface!
If this interface is not needed, it shall be closed by the protective cap provided with delivery in order to prevent any possible damages caused by electrostatic discharge.

You can use the following applications and services through this interface:

- I/O-Check
- MODBUS-RTU
- CODESYS 2.3
- Linux console
  - German keyboard layout
  - English keyboard layout

Only one of these applications or services can be accessed at one time over the RS-232 interface. This unique access can only be carried out in Web-Based Management. See section "Page Administration" for more information.

Note

Boot process
During the boot process, attached devices are not permitted to send data to the RS-232 interface because the firmware will not start otherwise. However, if this cannot be excluded, comment the "serial" parameter out in the menu.lst file (/boot/grub/menu.lst):
01 #serial --unit=0 --speed=115200
02 terminal --timeout=2 console
The boot loader then does not respond to the data input on the RS-232 interface.

5.7 DVI-I Interface (X7)

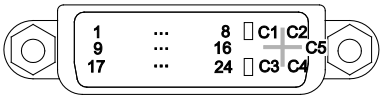
The DVI-I interface transmits analog and digital signals and is suitable for connecting to digital monitors.
This interface also transmits analog image signals, so that it is possible to connect CRT-VGA monitors while using a DVI-to-VGA adapter.

NOTICE

Open interface!
If this interface is not needed, it shall be closed by the protective cap provided with delivery in order to prevent any possible damages caused by electrostatic discharge.

The following table provides information on the DVI-I interface pin assignments:

Table 22: DVI-I Interface: Pin Assignments

Connector	Pin	Description
 Figure 14: DVI-I Interface	1	TXD2-
	2	TXD2+
	3	GND
	4	Not assigned
	5	Not assigned
	6	DDCCLK
	7	DDCDATA
	8	CRT_VSY
	C1	CRT_R
	C4	CRT_HSY
	9	TXD1-
	10	TXD1+
	11	GND
	12	Not assigned
	13	Not assigned
	14	VCC_DVI
	15	GND
	16	Not assigned
	C2	CRT_G
	C5	GND
	17	TXD0-
	18	TXD0+
	19	GND
	20	Not assigned
	21	Not assigned
	22	GND
	23	TXCP
	24	TXCN
	C3	CRT_B
	C5A	GND

6 Installing and Removing the I/O-IPC

The I/O-IPC has an elevated generation of heat. Excess heat is drawn off by passive heat dissipation (aluminum underside of the I/O-IPC and mounting rail). For this reason, the I/O-IPC shall always be mounted on a mounting rail.

6.1 Instructions for Installation/Removal

The following information shall always be observed:

NOTICE

Ventilation of the installation site
When installing the I/O-IPC, leave a free space on the sides and above of 40 mm to achieve sufficient heat dissipation. The surrounding temperature at the installation site shall not exceed +55 °C during operation.

NOTICE

Open interface!
Unneeded interfaces of the I/O-IPC shall be closed with the accompanying protective caps to prevent possible damage caused by electrostatic discharge.

- Select a sufficiently stable mounting rail and use several mounting points (every 20 cm), if applicable, to prevent the I/O-IPC from bending and twisting the mounting rail.
- When using mounting rails with a height of 7.5 mm, use flat riveting or sunken screws. Otherwise, the I/O-IPC and the connected I/O modules cannot be correctly mounted on the mounting rail.
- Take care that the physical interfaces are not soiled during installation. This can lead to damage and corrosion of the contacts.
- To avoid damage to the I/O-IPC, do not install it within the shear ranges of mobile systems or machine parts.
- Arrange for an appropriate potential equalization in your system.
- The mounting rails shall be screwed on so that they are grounded at the installation site.

6.2 Accessories Required for Installation

To install the I/O-IPC, you will need

- perforated or non-perforated mounting rails according to EN 60715 and
- a 750-600 End Module.

6.3 Acceptable Mounting Directions for the I/O-IPC

The I/O-IPC can be mounted either horizontally or vertically on a mounting rail. For vertical installation, appropriate measures shall be taken, such as using a safeguard to prevent sliding down (B) so that the I/O-IPC does not fall down due to vibrations and shock loads.

Note

Heat dissipation
In order to achieve good heat dissipation for the I/O-IPC, we recommend installing the device in mounting direction A 1 (Figure 10) on a mounting rail that has a heat dissipating connection to the fastening site.

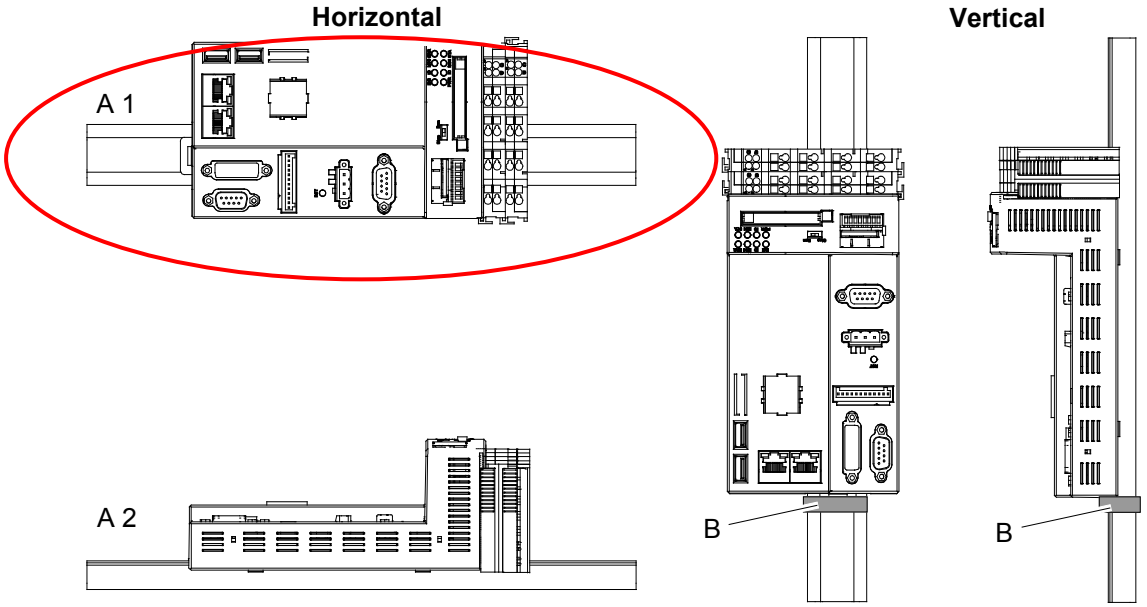


Figure 15: Mounting directions for the I/O-IPC; recommended mounting direction (A 1)

6.4 Securing the I/O-IPC to a Mounting Rail

For installing the I/O-IPC on a mounting rail, there are fasterners on the underside (23). These secure the I/O-IPC to the mounting rail.

Note

Fastening the mounting rail
If you use mounting rails with a height of 7.5 mm, then use flat riveting or sunken screws to fasten the mounting rail. Otherwise, the I/O-IPC and the connected I/O modules cannot be mounted correctly on the mounting rail.

Proceed as described below for mounting:

1. Disconnect from the power supply those parts of the system on which you wish to mount the I/O-IPC.

2. To mount on the rail, press the I/O-IPC with the underside against the mounting rail (25) until it snaps in.

3. Check that the I/O-IPC is securely placed on the mounting rail. The I/O-IPC shall not wiggle.

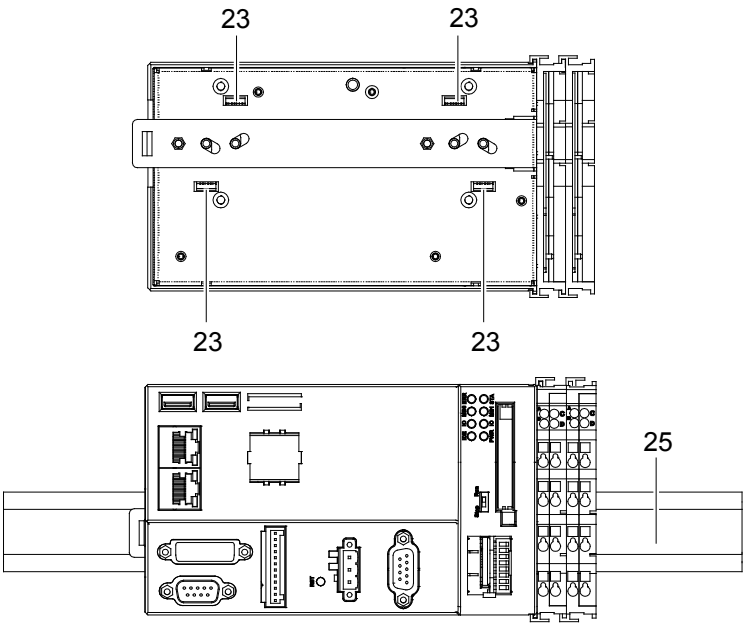


Figure 16: Securing of the I/O-IPC to a mounting rail

6.5 Connecting the I/O Module to the I/O-IPC

After mounting the I/O-IPC on a mounting rail, connect the I/O modules required for your application to the I/O-IPC. You can connect up to 64 750/753 Series I/O Modules to the I/O-IPC. The number is dependent on the total length of the connected I/O modules. This may amount to a maximum of 780 mm, including the end module, and the maximum size of the process image for the input and output data cannot exceed 500 bytes in each case.

Example of total length:
If the individual I/O modules have a width of 12 mm, 64 modules may be connected, but if they are 24 mm wide, then only 32 I/O modules may be connected.

With the optional WAGO internal data bus extension (consisting of the 750-628 Coupler Module and 750-627 End Module), it is possible to use up to 250 I/O modules. The same limitations apply in this case as for the use of 64 I/O modules.

Note

Using the WAGO internal data bus extension
Information on the use of the WAGO internal data bus extension can be found in the 750-627 and 750-628 documentation, which can be obtained from the WAGO internet site.

Note

Additional information and application instructions for use of WAGO Terminals
Additional information and instructions for use for WAGO I/O modules can be found in the WAGO-I/O-SYSTEM 750/753 system description. The respective manuals and data sheets can be found at www.wago.com.

To connect the I/O modules, proceed as described below:

1. Disconnect from the power supply those parts of the system on which you wish to mount the I/O-IPC.

2. Insert the slot (71) of each I/O module onto the key (70) of the previous module.

DANGER

Electric voltage!
When using 120/230V I/O modules, observe the safety precautions in the accompanying manual. If these precautions are not observed, there is a danger of electrical shock.

NOTICE

The highest current carrying capacity of the power contacts is 10A!
The maximum current carrying capacity of the I/O module power contacts must not exceed 10 A. An increase in current can lead to overheating of the contacts and damage to the I/O modules.

3. Connect the end module as the last.

When another I/O module is snapped on, the supply voltage for the sensors and actuators is automatically conducted over the power jumper contacts. The prerequisite is that the I/O modules used must also have power jumper contacts.

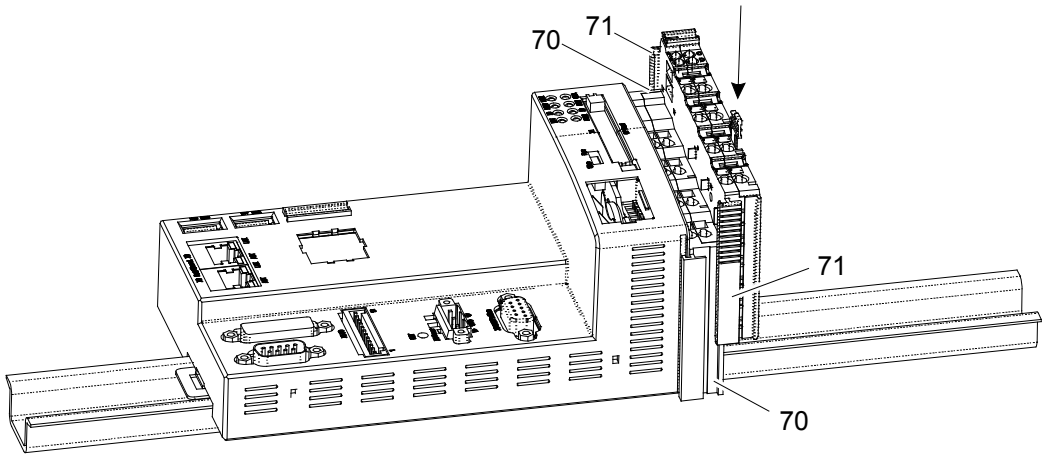


Figure 17: Connecting an I/O module to the I/O-IPC

6.6 Dismounting the I/O-IPC

To replace an I/O-IPC (e.g., in the event of version change), proceed as described in the following section.

⚠ CAUTION

Hot underside!
High temperatures may occur on the underside of the I/O-IPC during operation. If the I/O-IPC has been in operation, allow it to cool off before moving it.

6.6.1 Disconnecting Cables and Conductors

To remove the cables and conductors, proceed as follows:

1. Disconnect from the power supply those parts of the system on which you have mounted the I/O-IPC.

2. Disconnect the plugs/sockets from the I/O-IPC interfaces and remove the data cables.

3 Remove the conductors from the first 750 Series I/O Module connected to the I/O-IPC (A).

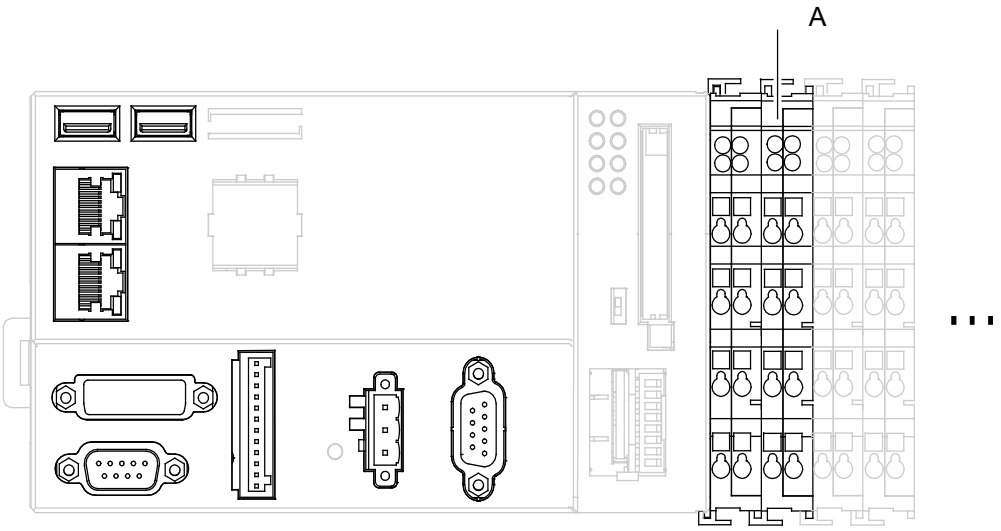


Figure 18: I/O-IPC interfaces

6.6.2 Removing the I/O-IPC from the Mounting Rail

Remove the I/O-IPC from the mounting rail as described below:

1. First, disconnect from the power supply those parts of the system on which you have mounted the I/O-IPC.

2. Then, pull the orange release tab (24) to remove the first 750 Series I/O Module connected to the I/O-IPC from the mounting rail (A and B).

3. To remove the I/O-IPC from the mounting rail, press the unlocking mechanism (22) with a suitable tool (e.g. screwdriver) in the direction of the arrow (see fig.) and pull the I/O-IPC out (C).

NOTICE

Internal data bus interface
The internal data bus interface (item 3) may not be disassembled. This I/O module is part of the I/O IPC.

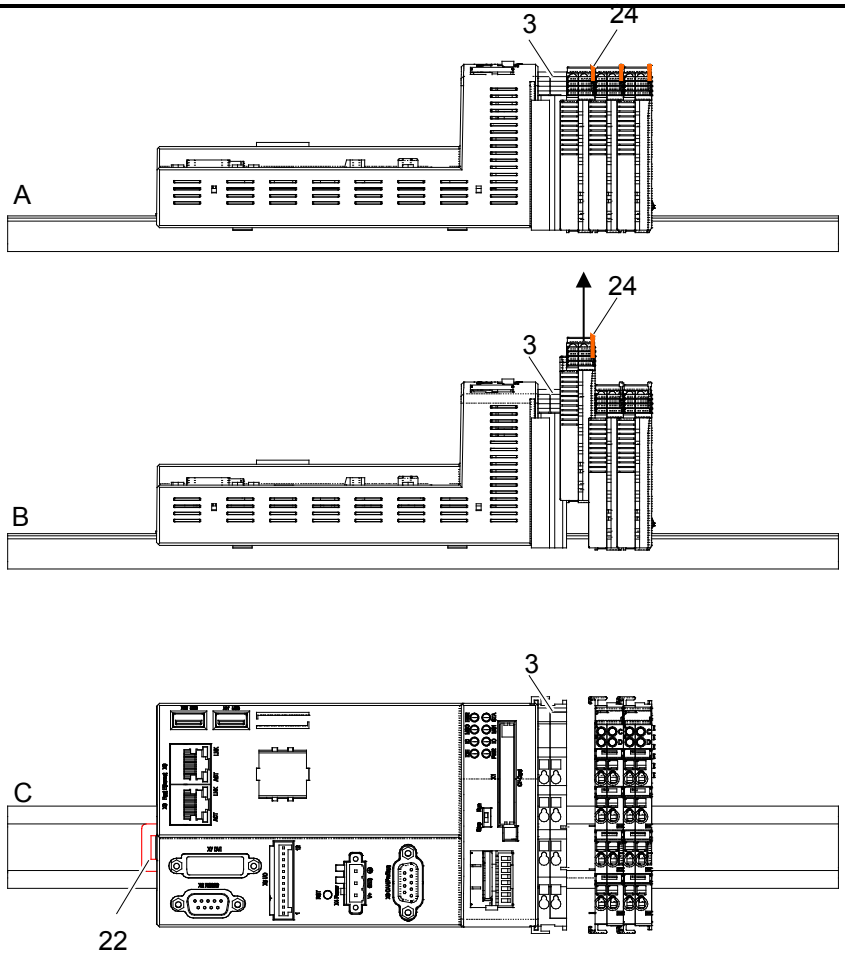


Figure 19: Removing the I/O-IPC from the mounting rail

7 Connecting the Supply Voltage

The supply voltage is connected as follows, depending on the application area of the I/O-IPC:

Note

Scope of Delivery
The 750-602 Supply Module is no longer included in the scope of delivery as of HW Version 11.

Table 23: Use of 750-602/626 depending on the application area of the I/O-IPC

Application Area	Filter Modules	Power Supply
Cable length < 3 m	750-602	See section, “Power Supply via 750-602 Supply Module”
Cable length > 3 m	750-626	See section “Power Supply via 750-626 Filter Module”

7.1 Notes

DANGER

Electric voltage!
Only operate the I/O-IPC with 24 V DC PELV (Protective Extra Low Voltage) or SELV (Safety Extra Low Voltage) power sources. There is a danger of electric shock if this precaution is not observed.

Note

Interruption of the supply voltage
If the power supply is interrupted by more than 1ms, the I/O-IPC is backed up and automatically restarted.

- To ensure electrical isolation, a power supply unit each for the electronic supply and the field supply must be used.
- Connect the power supply lines only when they are in a de-energized state.
- Keep power supply lines a sufficient distance away from electromagnetic sources of interference in order to maintain a high level of interference resistance of the 750 Series components against electromagnetic emissions.
- When laying any lines, make sure that you do not lay them within the shear range of movable machine parts.
- Observe the correct layout of the potential equalization.

7.2 Required Accessories

To connect the supply power to the I/O-IPC potentially requires the 750-626 Filter Module. It can be ordered at www.wago.com. The customer provides the accessories (e.g. individual cables) and tools required to connect the supply power.

7.3 Power supply via 750-602 Supply Module

Note

Scope of Delivery
The 750-602 Supply Module is no longer included in the scope of delivery as of HW Version 11.

Note

Length of the power supply cable
With this type of power supply, the length of the power supply cable shall not exceed 3 meters from the voltage source to the I/O-IPC X4 connection. If a longer cable is necessary, power shall only be supplied as described in section “Power Supply via 750-626 Supply Module”

NOTICE

Ensure the correct conductor cross-section!
Use conductor cross-sections from 0.08 mm² to 2.5 mm² (28 - 14 AWG) only for the X4 connection and for the CAGE CLAMP® connections of the power supply module.

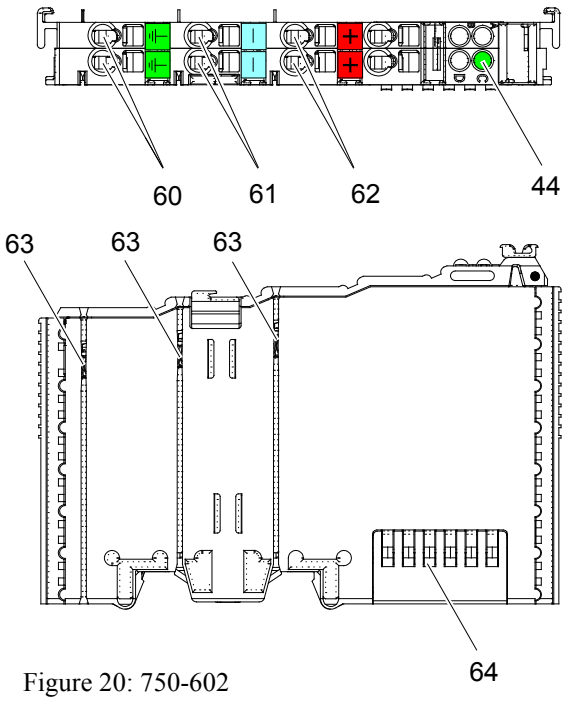


Figure 20: 750-602

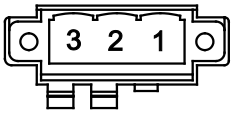
Table 24: Connections, contacts and supply module LEDs

Position	LED/Operating Element	Color/ Status	Meaning
60	Earth	-	Connection of protective earth
61	0 V DC	-	Ground (GND) of the supply voltage
62	Field supply, 24 V DC	-	24V supply voltage for the sensors/actuators. The power supply is protected against reverse polarity. For the supply of other field potentials; e. g., 230 V AC, appropriate power supply modules are available. You can find detailed information on this in the manuals for the power supply modules and in the 750-xxx system description.
63	Power jumper contacts	-	Conduction of the field-side supply voltage to the connected I/O modules.
64	Data contacts	-	These make the supply voltage (5 V, 1 A) available for the electronics in the I/O modules connected to the internal data bus. The supply voltage is provided by the I/O-IPC (connection X4).
44	LED	Green/off	The LED lights up if the field supply is available (see Pos. 62). Otherwise, the LED is off.

Requirements for connecting the power supply:

- You have connected two supply lines each to two voltage sources of +24 V DC and 0 V DC in a de-energized state.
- You have connected the accompanying socket for the X4 connection to the supply cable.

Table 25: Connection for Electronic Supply: Terminal Layout

Connector	Pin	Description
 Figure 21: Electronic Power Supply (X4)	1	V_IN (+)
	2	GND (-)
	3	Shield (optional)

To connect the supply cable for the I/O-IPC electronic supply and for feeding in the field supply for the connected I/O modules, sensors and actuators through the power supply module, proceed as follows:

1. Disconnect from the power supply those parts of the system on which you have mounted the I/O-IPC.

2. Connect the electronic supply cable to the I/O-IPC by connecting power supply cable socket to the I/O-IPC X4 plug (6). Provide this supply with a 1.6 A fuse.

3. Then, secure the socket using the screws supplied for this purpose.

4. To feed in the field supply, connect the 750-602 Supply Module (3) (No longer included in the scope of delivery as of HW Version 11) according to the illustration, with 24 V to "+" and 0 V to "-". Provide this supply with a 10 A fuse.

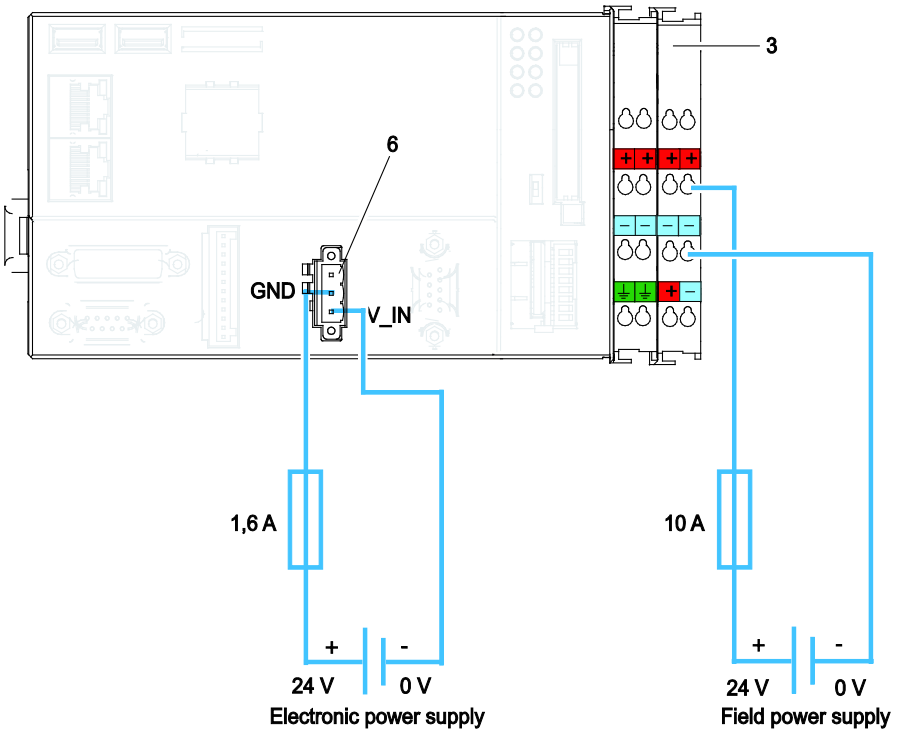


Figure 22: Power Supply via 750-602 up to HW 10 (with fieldbus)

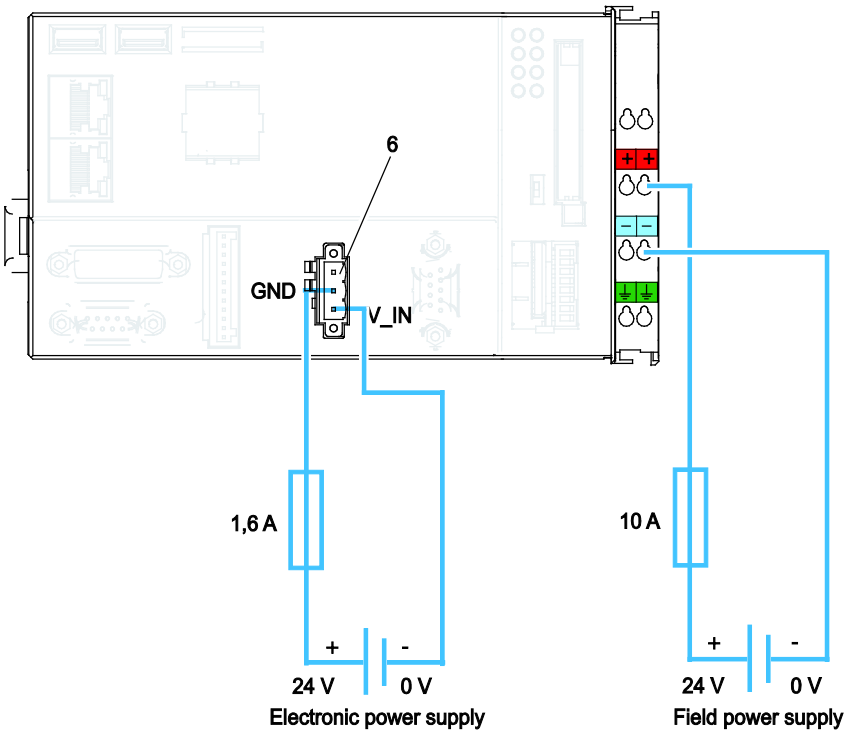


Figure 23: Power Supply via 750-602 as of HW 11 (with fieldbus)

7.4 Power supply via 750-626 Filter Module

The 750-626 Filter Module is required as soon as a supply line exceeds 3 m between the voltage source and the I/O-IPC.

NOTICE

Observe current carrying capacity!
For this power supply variant, you will need the 750-626 Filter Module **starting with hardware version 4**. This is the only filter module configured for the higher current carrying capacity. You can obtain the filter module from www.wago.com.

NOTICE

Observe the isolation voltage!
When using the 750-626 Filter Module, the isolation voltage of the field and electronic power supply is reduced to 50 V against ground.

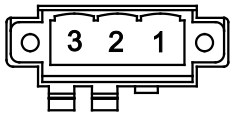
NOTICE

Ensure the correct conductor cross-sectional area!
Use conductor cross-sectionsranging from 0.08 mm² to 2.5 mm² (28 - 14 AWG) only for the CAGE-CLAMP® connections of the filter module.

Requirements for connecting the supply voltage

- You have connected the supply line to the voltage sources of +24 V DC and 0 V DC in a de-energized state.
- You have connected the accompanying socket for the X4 connection to the supply cable.

Table 26: Interface for Electronic Power Supply: Pin Assignments

Connector	Pin	Description
 Figure 24: Electronic Power Supply (X4)	1	V_IN (+)
	2	GND (-)
	3	Shield (optional)

To connect the I/O-IPC electronic supply and the field supply for the connected I/O modules, sensors and actuators, proceed as follows:

1. Disconnect from the power supply those parts of the system on which you have mounted the I/O-IPC.

2. Connect the power supply for the electronics and field supply for the I/O-IPC to the 750-626 Filter Module (4) by connecting 24 V to “+” and 0 V to “-” according to the illustration. Provide this supply with a 1.6 A fuse.

3. Feed in the power supply for the electronics and field supply from the Filter Module (4) to the I/O-IPC by connecting the supply cable socket to the X4 plug (6).

4. Connect the remaining lines with a 1.6 A fuse according to the figure.

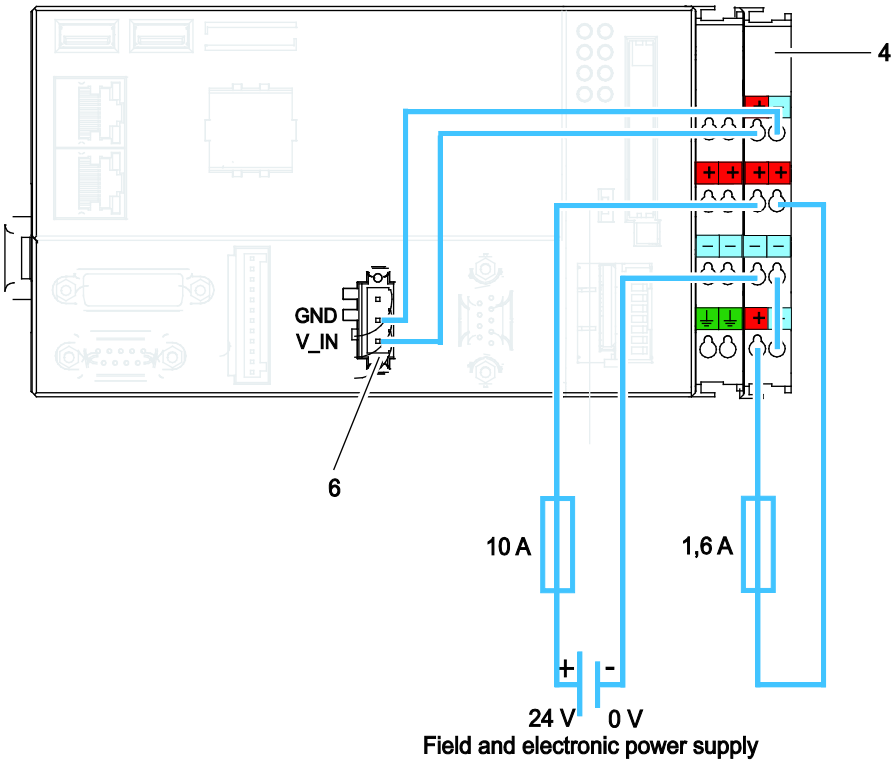


Figure 25: Power Supply via 750-626 (with fieldbus)

7.5 Connecting Sensor and Actuator Lines to I/O Modules

You can find additional information and instructions for wiring individual WAGO I/O modules in the WAGO-I/O SYSTEM 750 system description, and the respective manuals and data sheets at www.wago.com.

8 Commissioning

8.1 Turning the I/O-IPC On

Before turning the I/O-IPC on, check that you

- have installed the I/O-IPC properly (see section "Installing and Removing the I/O-IPC"),
- have connected all required data cables (see section “Description of I/O-IPC Interfaces”) to the corresponding interfaces and have secured the locking screws at the plugs/sockets,
- have connected the power supply for the electronics and field supply (see section “Connecting the Power Supply”),
- have connected the 750-600 End Module (see section "Connecting the I/O Module to the I/O-IPC"),
- have performed an appropriate potential equalization on your machine/system (see System Description for 750-xxx) and
- have performed shielding properly (see System Description for 750-xxx).

⚠ WARNING

Warning against personal injury!
When using the 758-87**4**/xxx, 758-87**5**/xxx und 758-87**6**/xxx I/O-IPC variants in potentially explosive atmospheres, carefully follow the instructions in the section “Use in potentially explosive atmospheres”.

To turn on both I/O-IPC and connected I/O modules, switch on your power supply unit. After the initialization phase, the Linux operating system starts, and then the CODESYS 2.3 programming system. After an error-free system start, the I/O-LED of the I/O-IPC lights up green.

If you update your existing firmware version, depending on the version, this can take a few minutes. Please be patient until the operating system has restarted.

Note

Do not remove I/O modules
During operation, it is forbidden to insert or remove I/O modules, as this results in an error in the I/O-IPC and/or the connected I/O module.

8.2 Determining the IP Address of a Host PC

So that the host PC (e.g. notebook) can communicate with an I/O-IPC via the ETHERNET, both must be located in the same subnet.

To determine the IP address of a host PC (with MS Windows operating system) using the MS-DOS prompt, proceed as follows:

1. Click on "Start" and select "Execute".
2. Enter the command `cmd` and press the Enter key. A prompt opens.
3. Enter the command `ipconfig` and press the Enter key.
4. The IP address, subnet mask and standard gateway, including the appropriate parameters, are displayed.

8.3 Setting Up an IP Address

As delivered, the I/O-IPC assigns the following IP addresses to the ETHERNET interfaces X8 and X9:

Table 27: Pre-set IP addresses for the ETHERNET interfaces

ETHERNET interface	Default setting
X9	Dynamic assignment of the IP address using the bootstrap protocol (BootP)
X8	Fixed IP address **192.168.2.17**.

So that a PC and the I/O-IPC can communicate with one another, use the Web-based management or the "IPC Configuration Tool" to adapt the IP addressing to your system structure (see section "Configuration").

Example for incorporating the I/O-IPC (192.168.2.17) into an existing network:
If the IP address of your host PC is 192.168.1.2, then the I/O-IPC must be on the same subnet. That is, with the net mask **255.255.255**.0, the first three digits of the I/O-IPC must match those of your PC. From this, the following address space arises for the I/O-IPC:

Table 28: Net mask 255.255.255.0

Host PC	Subnet address space for the I/O-IPC
192.168.1. 2	**192.168.1**.2 … **192.168.1**.254

Note

IP addresses
You cannot assign IP addresses for both ETHERNET interfaces in the same subnet. Neither can you do this with the BootP or DHCP server.

8.3.1 Assigning an IP Address Using BootP

The I/O-IPC can obtain its IP address dynamically (DHCP/BootP) from a server or be provided with a static IP address. In contrast to fixed IP addresses, dynamically assigned addresses are not stored permanently. Therefore, the presence of a BootP or DHCP server is necessary each time the I/O-IPC is restarted.

The assignment of the IP address using BootP is explained here using an example with the WAGO BootP Server.

Prerequisite:
The WAGO BootP Server can be installed on your PC. It can be obtained at www.wago.com.

Configuring the WAGO BootP Server

1. Make a note of the MAC address of the I/O-IPC's ETHERNET interface. You can find this on the label on the side of the I/O-IPC (see Section "Lateral Marking").

2. Turn on your PC.

3. In the "Start" menu, start the WAGO BootP Server by navigating to **Programs** >**WAGO Software** > **WAGO BOOTP Server**.

4. Open the configuration file by clicking on the **[Edit BootPtab]** button in the BootP Server. In the configuration file, assign each of the MAC addresses to an IP address in the same network.

5. To do so, click in the marked line of the configuration file:

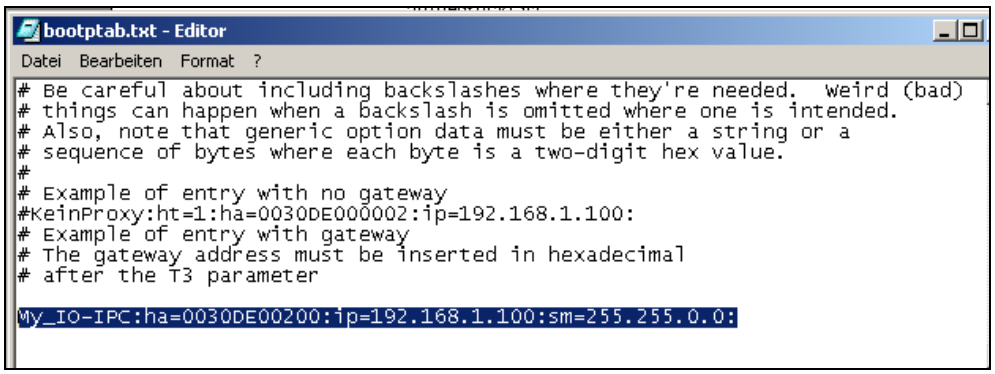


Figure 26: Configuration line in the configuration file

6. Replace the MAC address consisting of twelve characters of the first ETHERNET interface behind **:ha=** with the one printed on the label on the side of the I/O-IPC.

7. Enter an IP address after **ip=**. In this example, this is 192.168.1.100.

8. To address the second ETHERNET interface, insert another line with the corresponding assignment in the file bootpTabletxt. When doing so, repeat steps 5 through 7.

9. Save the new settings in the bootpTabletxt file. To do this, click on "File" in the menu and select "Save".

10. Close the editor.

Table 29: Explanations of the configuration line

Parameter	Description
Node_1	Name of the I/O-IPC with the I/O modules. This can be freely chosen.
ht=1	Network hardware type. For ETHERNET, this is 1.
ha=0030DE000200	MAC address of an ETHERNET interface.
ip= 192.168.1.2 ip= 192.168.1.100	IP address for an I/O-IPC located in the same network as the host PC as well.
gw=192.168.1.1	IP address for the gateway. For a local network, you do not need to specify a gateway.
Sm=255.255.255.0	Subnet mask of the subnet to which the I/O-IPC is supposed to belong.

Assigning an IP Address Using the WAGO BootP Server

1. To start the BootP Server click the button [Start] in the BootP dialog window that is opened. Various messages will be displayed in the window. The error messages indicate that some services (e. g., Port 67, Port 68) have not been defined in the operating system. You can ignore these error messages.

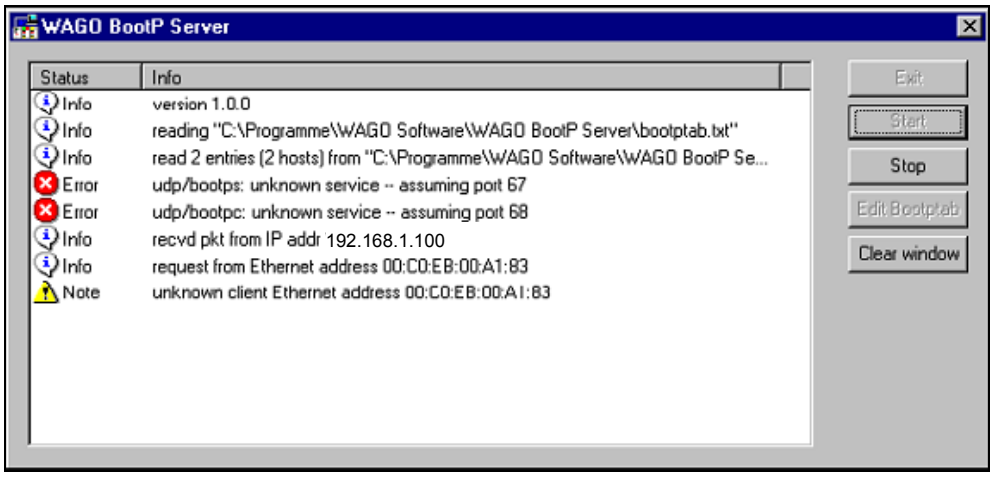


Figure 27: Dialog window of the WAGO BootP Server with messages

2. Restart the I/O-IPC by switching the supply voltage of the I/O-IPC off and then on again, or by pressing the reset button.
A request from the I/O-IPC will appear. The BootP Server responds that the IP address has been accepted (no errors). The IP address is now temporarily present in the I/O-IPC, but not stored permanently. If restarted, the I/O-IPC will try again to receive a new IP address from the BootP Server.

3. Click **[Stop]**, then **[Exit]** to close the BootP Server.

4. To save the IP address permanently in the I/O-IPC, select the option "Static IP" in Web-Based Management in the "TCP/IP" page.

8.3.2 Changing an IP Address Using the Linux Console (IPC Configuration Tool)

Using the IPC configuration tool accessible on the Linux console, you can assign the X8 and X9 ETHERNET interfaces a new IP address. For additional information about the IPC configuration tool, see section "Configuration".

Preparation:
You have connected a monitor to the DVI-I interface and a keyboard to the USB interface of the I/O-IPC. For more information, see section "Access over DVI-I and USB Interface".

1. Using the key combination **[Alt] + [F3]**, open the third console of the I/O-IPC on which the IPC configuration tool is located.

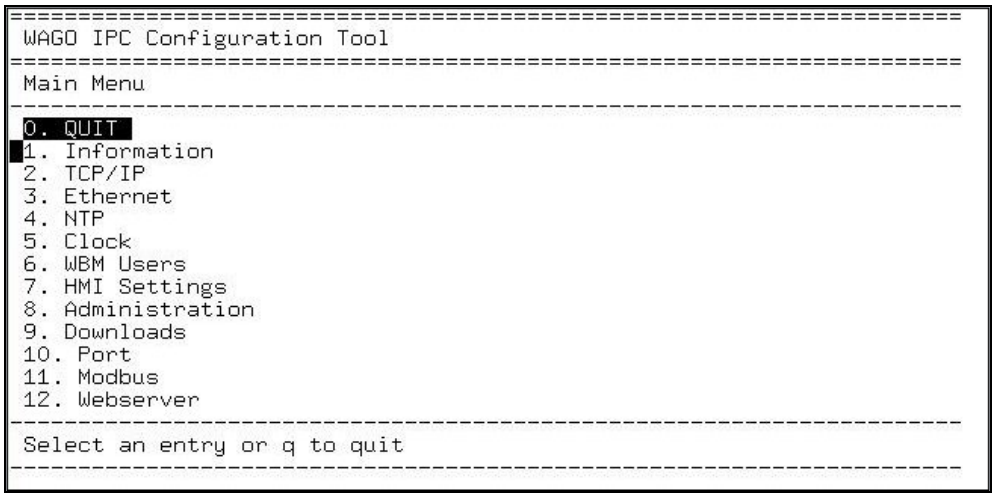


Figure 28: Start screen of the WAGO IPC configuration tool

2. Use the keyboard (arrow keys or number block) to select the **TCP/IP** entry and press the **[Enter]** key.

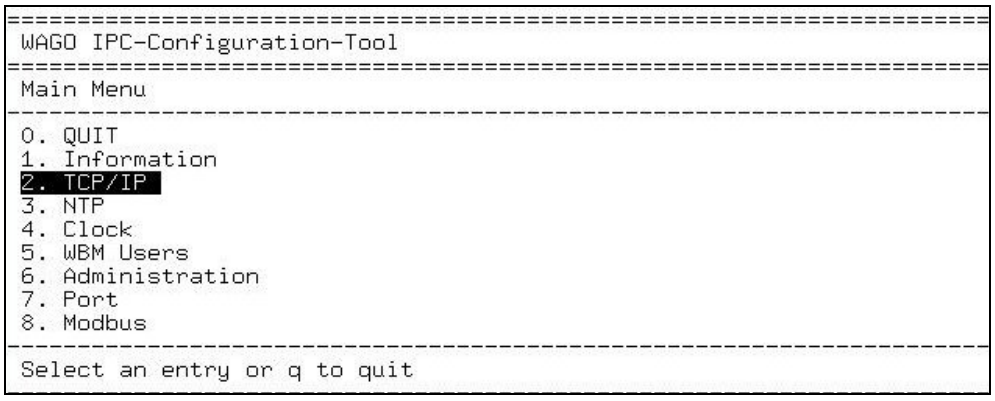


Figure 29: TCP/IP

3. To change the X9 ETHERNET interface, select **TCP/IP Configuration eth0** or **TCP/IP Configuration eth1** for the X8 ETHERNET interface. Then press the **[Enter]** key.

   In this example, the X9 ETHERNET interface is selected to change the preset IP address:

```
================================================================================
 WAGO IPC-Configuration-Tool
================================================================================
 TCP/IP
--------------------------------------------------------------------------------
 0. Back to Main Menu
 1. Hostname
 2. Default Gateway
 3. DNS Server
 4. TCP/IP Configuration eth0 (X9)
 5. TCP/IP Configuration eth1 (X8)
--------------------------------------------------------------------------------
 Select an entry or q to quit
--------------------------------------------------------------------------------
```

Figure 30: TCP/IP configuration eth0 (X9)

4. Select **IP-Address** and click the button **[Enter]**.

```
================================================================================
 WAGO IPC-Configuration-Tool
================================================================================
 TCP/IP Configuration eth0 (X9)
--------------------------------------------------------------------------------
 0. Back to TCP/IP Menu
 1. State.............................enabled
 2. Type of IP-Address-Configuration...static
 3. IP-Address.......................192.168.1.17
 4. Subnet Mask......................255.255.255.0
--------------------------------------------------------------------------------
 Select an entry or q to quit
--------------------------------------------------------------------------------
```

Figure 31: IP address

5. Enter the new IP address for the selected ETHERNET interface and confirm it with **[OK]**. If you want to return to the main menu without making changes, select **[Abort]**.

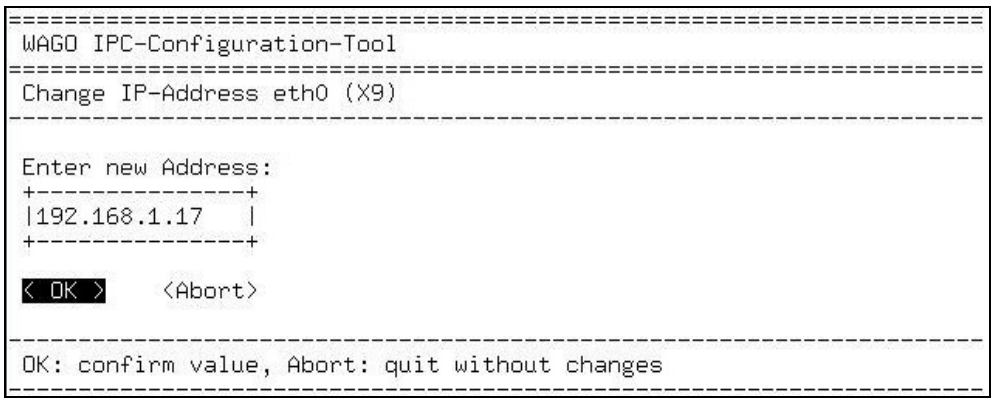


Figure 32: Enter new address

Note

IP addresses
You cannot assign IP addresses for both ETHERNET interfaces in the same subnet. Neither can you do this with the BootP or DHCP server.

8.4 Testing the Network Connection

To check whether you can reach the I/O-IPC at the IP address you have assigned in the network, carry out the ping network service. To do this, open the prompt in MS Windows by clicking on the "Start" button and selecting **Programs** > **Execute.** Enter `cmd` in the "Execute" dialog and click **[OK].**

1. Enter the ping command and the IP address of the I/O-IPC in the DOS window: Example: `ping 192.168.1.100.`

2. Press the Enter key. Your PC will send a request that will be answered by the I/O-IPC. The answer appears in the DOS window. If the error message "Timeout" appears, the I/O-IPC has not reported properly. Please check your network setting.

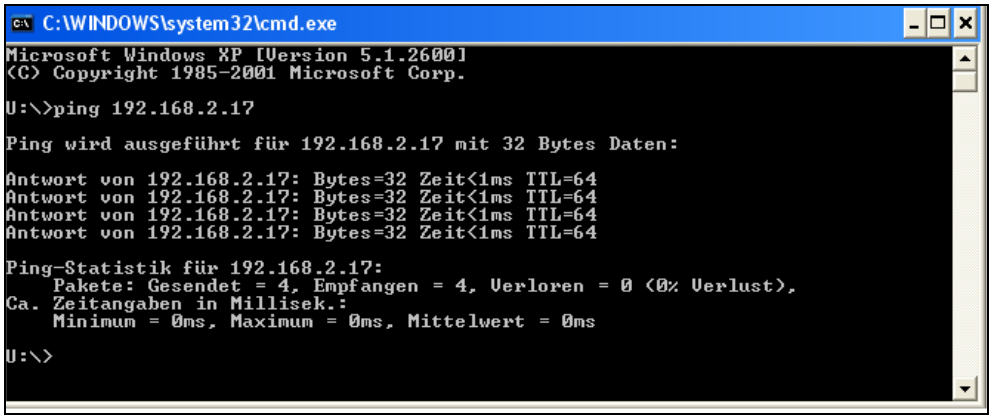


Figure 33: Example of a functional test

3. If the test is completed successfully, close the DOS window.

8.5 Switch Off/Re-start

⚠ CAUTION

Hot underside!
High temperatures may occur on the underside of the I/O-IPC during operation. If the I/O-IPC has been in operation, allow it to cool off before moving it.

To switch the I/O-IPC off, proceed as follows:

1. Close the running software and the operating system.

2. Switch the I/O-IPC off by turning off the power supply or by disconnecting the power socket from theX4 connector (7).

To restart the I/O-IPC, press the reset button (42) or switch the I/O-IPC off and then on again.

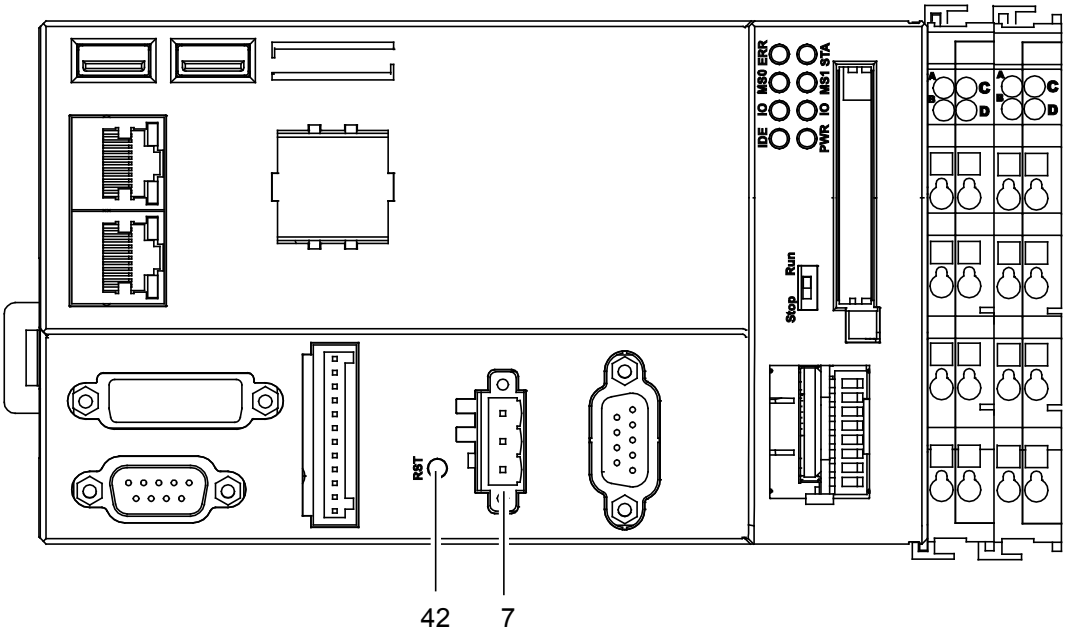


Figure 34: Switch off/restart of the I/O-IPC

9 Configuration

The following methods are available for configuring the I/O-IPC:

- Access using the PC via Internet browser to the Web-based management (section Configuration via "Web-based Management (WBM)")
- Access using the PC via a terminal program (via ETHERNET and/or RS-232 interface) to the "IPC Configuration Tool" (section "Configuration with a Terminal Program")
- Access via the I/O-IPC using the touch screen/monitor and USB keyboard to the "IPC Configuration Tool" (section "Configuration with Touchscreen/Monitor and USB Keyboard")
- Access via the PLC program CODESYS using WagoConfigToolLIB.lib (section "Calling the ' WagoConfigToolLIB.lib'")

The "IPC Configuration Tool" makes the same parameters available for the configuration of the I/O-IPC as the WBM. Descriptions of the parameters are available in the section "Information" Page.

9.1 Web-Based Management (WBM)

The implemented HTML pages (abbreviated as "pages" in the following) for Web-Based Management are used to configure the I/O-IPC. To access Web-Based Management, proceed as follows:

1. Connect the I/O-IPC via the **X8** ETHERNET interface to the ETHERNET network.

2. To access the pages, start your internet browser and enter the preset IP address **192.168.2.17** in the address bar. Note that the PC and I/O-IPC must be on the same subnet for this (see section "Setting Up an IP Address").

If you have installed a BootP server on your PC and would like to access WBM through BootP, use the other interface. You can find detailed information about this in the section "Assigning an IP Address Using BootP".

Note

Start page of the I/O-IPC
If the I/O-IPC does not display the start page, make sure that your internet browser settings allow the circumvention of the proxy server for local addresses. Also check whether your PC is located in the same subnet as the I/O-IPC.

Some WBM pages are password protected. The first time you select an item from the navigation bar, the password query appears:

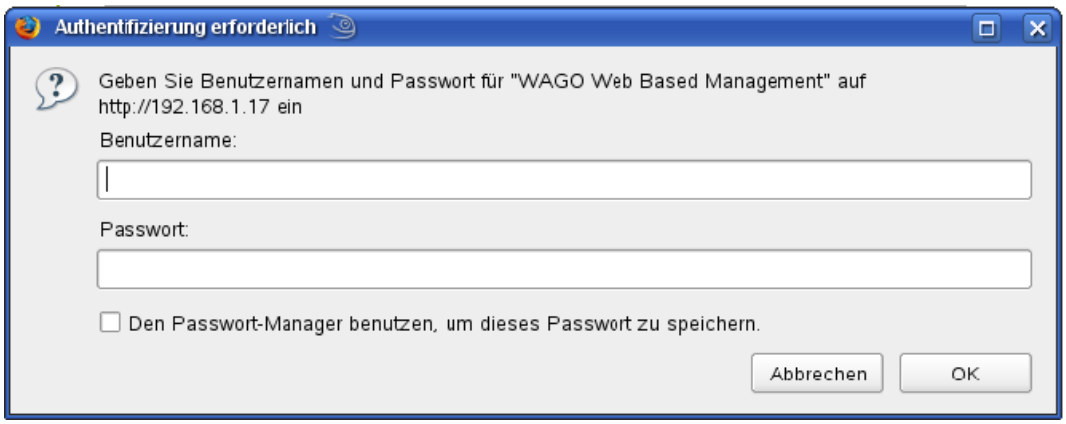


Figure 35: Entering the authentication

9.1.1 User Administration of WBM

To allow settings to be made by a select number of person only, limit access to WBM functions through User Administration.

NOTICE

Passwords
Change the passwords for your own use since the standard passwords are documented in these instructions. Therefore, the standard passwords do not provide sufficient protection (see section "Users ‘Page").

Table 30: User Settings in the Initial State

User	Password
user	user
admin	wago

Access to the WBM pages is as follows:

Table 31: Access Rights for WBM Pages

WBM Internet Site	Password
Information	-
CODESYS	
TCP/IP	user or admin
ETHERNET	user or admin
NTP	user or admin
Clock	user or admin
Users	admin
HMI Settings	
Administration	
Firmware Update	
Mass Storage	
Downloads	
Port	user or admin
MODBUS	user or admin
SNMP	user or admin
I/O Configuration	-
WebVisu	

9.1.2 "Information" Page

After entering the IP address, the start page "Information" for web-based management appears. This page provides information on the I/O-IPC and the ETHERNET network.

Figure 36: "Information" page (Example)

The following table explains the parameters listed on the page:

Table 32: Description of the Parameters of the "Information" Page

Coupler Details	
Order Number	Item number for the I/O-IPC
Processor Type	CPU-Type des I/O-IPC
Fieldbus Type	Fieldbus type des I/O-IPC
Firmware Revision	Firmware status
License Information	Notification that the runtime system CODESYS is available.
Kbus FW Revision	Firmware status of the I/O module interface
Network Details Eth0 (X9)/Eth1 (X8)	
State	Status of the ETHERNET interface (activated/deactivated).
Mac Address	MAC address used to identify and address the I/O-IPC.
IP Address	Current IP address of the I/O-IPC.
Subnet Mask	Current subnet mask of the I/O-IPC.

9.1.3 "CODESYS" Page

You will all information about the PLC program created in CODESYS on that page.

Table 33: Description of the Parameters of the "CODESYS" Page

Project Details	
Date	Display of project information that the programmer entered in the PLC program (in CODESYS under **Project** > **Project Information...**). The information only appears in a executed PLC program. Descriptive texts up to 1024 characters long are found under "Description".
Title	
Version	
Author	
Description	
CODESYS State	
State	STOP: PLC program is not executed RUN: PLC program is executed
Tasks	
When running a PLC program, the following information appears for each task: - Number of tasks - Cycle count - Cycle time (mcmsec) - Cycle time min (mcmsec) - Cycle time max (mcmsec) - Cycle time avg (mcmsec) - Status - Mode - Priority - Interval (msec) - Event - Function pointer - Function index	

9.1.4 "TCP/IP" Page

On the page for TCP/IP configuration, change the parameters for the ETHERNET configuration. To accept your entries, click on the **[SUBMIT]** button.

Table 34: Description of the Parameters of the "TCP/IP" Page

Common Configuration Data	
Host Name	If you have chosen the dynamic assignment of an IP address through DHCP, enter the host name of your PC here so that your PC can connect to the ETHERNET network.
Default Gateway	
None	This is where you select the interface that should be used as the default gateway. The default gateway is used by the I/O-IPC if the target address lies outside of the individual network.
X8	
X9	
Value	Enter the address of the default gateway here.
DNS Server	
Domain Name	Set the domain name here.
DNS-Server 1, 2, …	Set the address of the DNS server here. **[CHANGE]**: Accept the DNS address. **[DELETE]**: Deletion of the DNS address (line is removed).
Add DNS Server	Add additional DNS addresses here.
TCP/IP Configuration Eth0/Eth1	
State	
Enabled	Activate the corresponding ETHERNET interface here.
Disabled	Deactivate the corresponding ETHERNET interface here.
Type of IP Address Configuration	
Static IP	Here, choose whether you would like to use a static or dynamic IP address. Static IP: Static IP address DHCP and BootP: Dynamic IP address
DHCP	
BootP	
Configuration Data	
IP Address	Here, enter a static IP address. This is active if "Static IP" is activated in the "Type of Address Configuration" field.
Subnet Mask	Here, enter the subnet mask. This is active if "Static IP" is activated in the "Type of Address Configuration" field.

9.1.5 "ETHERNET" Page

Configure the transmission speed and communication method of the I/O-IPC ETHERNET interface on this page. To accept your entries, click on the **[SUBMIT]** button.

Table 35: Description of the Parameters of the "ETHERNET" Page

Transmission Mode Eth0 und Eth1	
Autonegotiation on	When the function is enabled, two ETHERNET interfaces (e.g. from a computer and I/O-IPC) connected to each other establish the maximum possible transmission speed and the duplex method together independently.
Settings	Select the fixed transmission speed and communication method: 10 Bit or 100 Mbit half-duplex (Half-duplex: Information can be send or received) 10 MBit or 100 Mbit full-duplex (Full-duplex: Information can be sent and received simultaneously)

9.1.6 "NTP" Page

On this page, configure the NTP parameters for setting the time clock. To accept your entries, click on the **[SUBMIT]** button.

Table 36: Description of the Parameters of the "NTP" Page

Configuration Data	
Enabled	Activate/deactivate time update here.
Disabled	
Port	Here, enter the port number for the NTP access (base setting: 123)
Time Server	Enter the IP address of the time server here.
Update Time (seconds)	Set the cycle time here to specify how often the time should be requested from the time server.

9.1.7 "Clock" Page

On this page, configure the real-time clock. To accept your entries, click on the **[CHANGE]** button.

Note

Enter the time zone
Your time zone entries are not active until after a restart/reset of the I/O-IPC.

Table 37: Description of the Parameters of the "Clock" Page

Time and Date	
Time on device, local	Set local time here.
Time on device, UTC	Set GMT time here.
Date on Device	Set the date here.
12-Hour-Format	Switching between 12-hour and 24-hour time display.
Time Zone	
Select	Select the appropriate time zone for your country here. Basic setting: AST/ADT: "Atlantic Standard Time", Halifax EST/EDT: "Eastern Standard Time", New York, Toronto CST/CDT: "Central Standard Time", Chicago, Winnipeg MST/MDT: "Mountain Standard Time", Denver, Edmonton PST/PDT: "Pacific Standard Time", Los Angeles, Whitehouse GMT/BST: "Greenwich Main Time", GB, P, IRL, IS, ... CET/CEST: "Central European Time", B, DK, D, F, I, CRO, NL, ... EET/EEST: "East European Time", BUL, FI, GR, TR, ... CST: "China Standard Time" JST: "Japan/Korea Standard Time"
Edit TZ-String	For time zones that cannot be selected using the "Select" parameter, enter the appropriate time zone for you here. You can obtain an overview of all time zones at http://home.tiscali.nl/~t876506/TZworld.html Information on how to edit the time zone string in Linux can be found at http://www.minix-vmd.org/pub/Minix-vmd/1.7.0/wwwman/man5/TZ.5.html

9.1.8 "Users" Page

On this page, change the passwords of the users **admin** and **user**. To do this, you must be logged on as the user **admin**. An overview of passwords can be found in section "User Administration of WBM". To accept your entries, click on the **[SUBMIT]** button.

NOTICE

Passwords
Change the passwords for your own use since the standard passwords are documented in these instructions. Therefore, the standard passwords do not provide sufficient protection.

Table 38: Description of the Parameters of the "Users" Page

Configuration Data	
Select User	Here, select the user (**user** or **admin**) for whom you want to assign a new password.
New Password	Here, enter the new password for the user selected under "Select User".
Confirm Password	Here, enter the new password again for confirmation.

9.1.9 "HMI Settings" Page

On the "HMI Settings" page, change the graphic resolution for the DVI-I interface, configure the touch screen or monitor, and choose between a English or German keyboard layout.

To save all settings made on the page, click the **[SUBMIT]** button. The I/O-IPC must be restarted or reset to apply the settings.

Note

Graphics resolution
To change the graphics resolution in the (758-870/000-xxx) I/O-IPC, the video configuration in BIOS must also be reset.

The following adjustments must be carried out:
BIOS > Motherboard Device Configuration > Video and Flat Panel Configuration > Panel Type: VGA / SVGA / XGA
VGA -> 640x480, 16 bit
SVGA -> 800x600, 16 bit
XGA -> 1024x768, 16 bit

Table 39: Description of the Screensaver and Cleanmode parameters on the "HMI Settings" page

Screensaver	
Display	Indicates whether the display is switched on or off. off: red cross and text "**off**" on: green check and text "**on**" The display can be switched off by the screensaver (after the corresponding wait time) or explicitly by the user. The current value is queried and indicated cyclically every two seconds on the web site (this is only queried once in Configtool instead of cyclically). The display can be switched on or off using the **[Switch on]** or **[Switch off]** buttons. These changes are effective immediately.
Screensaver	Indicates the status of the Screensaver Function. enabled: green check and text "**enabled**" disabled: red cross and text "**disabled**" When the screensaver is turned on (enabled), the touch screen display is turned off after the configured wait time and turned on again following touch screen or keyboard entry. When the screensaver is turned off (disabled), the display can only be explicitly switched on or off by the user (see section "Display"). By using the **[Enable]** or **[Disable]** buttons, the Screensaver Function can be immediately activated or deactivated. This does not effect the current status of the display – even if the screensaver is activated, the status of the display only changes after passage of the wait time or pressing on the touch screen/keyboard.

Table 39: Description of the Screensaver and Cleanmode parameters on the "HMI Settings" page

Wait time (sec)	The "Wait time" parameter is only evaluated if the "Screensaver" parameter has enabled the status. If so, the value indicates the time in seconds after which the screensaver is activated. You can change the wait time by changing the value in the input field and clicking on the **[Change]** button. The changes immediately effect the next activation of the screensaver.
Cleanmode	
Actual State	Indicates whether Cleanmode is currently active or inactive active: green check and text "**on**" inactive red cross and text "**off**" When Cleanmode is active, contacts of the touch screen are ignored. Cleanmode can be switched off by the user; at the latest, it is automatically turned off (and the touch screen is then usable again) following the set timeout time. The current value is queried and indicated cyclically every two seconds on the web site (this is only queried once in Configtool instead of cyclically). The display can be switched on or off using the **[Switch on]** or **[Switch off]** buttons. These changes are effective immediately.
Timeout (sec)	This indicates the Timeout time for the Cleanmode, i.e. the time, during which contacts of the touch screen are ignored due to activated Cleanmode. You can change the wait time by changing the value in the input field and clicking on the **[Change]** button. The changes immediately effect the next activation of Cleanmode (but not the already activated Cleanmode).
VGA-Configuration	
Select the graphic resolution for the DVI-I interface for the monitor used here. To accept your entries, click on the **[SUBMIT]** button. If a dark image is displayed on the monitor/touch screen only, the resolution selected does not match the resolution of the monitor/touch screen. Choose another resolution.	
Show mouse pointer	This is how you can hide or show the mouse pointer on the monitor.

Table 39: Description of the Screensaver and Cleanmode parameters on the "HMI Settings" page

Touchscreen Configuration	
Device-Name	This is where you select a touch screen (mouse dev) connected via USB. The I/O-IPC does not support serially connected touch screens. If a selected touch screen is no longer connected, the message "not available" will appear here.
Driver-Name	This is where you select a device driver for the previously-selected touch screen.
Execute calibration of touchscreen at next start	If you activate this option, when you restart the I/O-IPC, an interface for calibrating the touch screen will appear before executing the SPS program.  **Note** **Touchscreen** If no touch screen is connected to the I/O-IPC, this option must remain deactivated since otherwise the SPS program will not start.
Keyboard Layout	
German English	Select either an English or German keyboard layout for the Linux console here.

9.1.10 "Administration" Page

The "Administration" page is used to save all settings made to a CF card or in the internal memory of the I/O-IPC.

Table 40: Description of the Parameters of the "Administration" Page

Create bootable image from active partition	
Select destination	Select the media to which you want to copy the image, either from the CF card to the internal flash memory or vice versa. Press the **[Start Copy]** button to copy.
Configuration of Serial Interface	
CODESYS I/O-Check MODBUS/RTU Linux Console	Here, select the service that is to be executed on the serial interface. To accept your entries, click on the **[SUBMIT]** button. **⚠ WARNING** **Activating "Control Mode“ in WAGO-IO-CHECK!** Using WAGO-IO-CHECK, you can overwrite parameters and process data in "Control Mode" regardless of whether the fieldbus or PLC functionalities are enabled or disabled. By doing so, machine components may be placed in a dangerous state and personnel and machines may be at risk. Before changing parameters and process data, ensure that the machine components are in a safe and defined state and switch off the higher-level controller. Also ensure before start-up that no personnel remain in the danger area of the machine components.
Free Port (CODESYS Libs)	The serial interface is available for some applications such as CODESYS (CODESYS/SysLibCom or SerComLib).
File system Check	
Select Device	This is how you execute a check of the file system for a device selected from the list. To start the check, click the **[Start Check]** button. If during the check a problem was detected, a corresponding error message is displayed above on the page about "Configuration of Serial Interface" ("Error while filecheck. If more ..."
Start Backup System	
Start Backup System	Switch the boot loader to start the other system partition the next time the system is restarted. There is an older version of the firmware here if you previously restored the system software via the "Firmware Update" page.
Reboot IPC	
Press the **[Start Reboot]** button to restart the I/O-IPC.	

9.1.11 "Package Server" Page

From the "Package Server" page, copy the firmware as a "backup package" from the current partition of the I/O-IPC to devices attached to the I/O-IPC. An "Update Package" may contain all or the following individual components: the CODESYS project, I/O-IPC settings, and the file system.

Table 41: Description of the Parameters of the "Package Server" Page

Backup You can initiate backups from the active partition of your IPC to a selected destination. In the table, an arrow illustrates the direction of the backup – the packages are saved from the active partition to the selected destination. Creating the backup files takes some time on the IPC. A progress bar appears after clicking the button **[SUBMIT]**.	
Active Partition	The partition of your IPC currently active appears. The backups can only be performed from this partition.
Packages	Select the checkbox of the package for which you want to create a backup: **All:** Selection of all "backup packages" **CODESYS Project:** CODESYS project **Settings:** All settings that can be made e.g. via the WVM such as IP addresses, ETHERNET settings, HMI configuration, etc. **System:** The complete system partition
Destination	You can select the destination to which your backup should be saved. All media currently attached appear such as the CF card, internal flash memory and various USB storage. The media currently active (from which the backup is created) does not appear in the list. The list also contains the "Network" item. If you select this item, the backup files are downloaded to your computer via the network; only one package can be transferred per operation. If you select the "Network" menu item, you cannot select multiple packages nor select the "All" checkbox. After clicking the **[SUBMIT]** button and creating the backup file, your browser displays the typical download window to save the file to the file system of your computer. When saving the file, do not remove any text from the suggested file name and add something to the end of it (numbers, etc.). The packages can be assigned in the first step (only) by the file names. The backup of the "CODESYS" and "System" packages creates compressed memory dump of complete partitions. However, the Settings backup is created using the config tools and saved in a text file. The text files can be viewed in any normal text editor and manually edited, as required. However, you must know exactly in what format the individual config tools save or need the parameter values.

Table 41: Description of the Parameters of the "Package Server" Page

Activate auto update feature	When creating packages using this function, the packages are automatically copied from the media (CF card/USB storage) to the device when starting the device. **Note** **Security question** You are not prompted to save the packages when booting. All data on the media (CF card/USB storage) is automatically copied to the system and old data/programs are overwritten.
Restore	
When restoring, the backups previously saved on the active media of the IPC are imported. The packages are saved to the active partition from the source selected. After restoring the "System" and/or "Settings" packages, the IPC is automatically restarted. If only the "CODESYS" package is installed, only the run-time system is stopped during the backup. In contrast to the other packages, the system restore has the special feature that an extra partition is available for this purpose. The backup files are copied to the partition and only activated after the IPC is automatically restarted .with the toggled boot loader. As a result, the old version of the system is still available (as long as the "Restore" function is not executed) and can be activated again using the "Start Backup System" function on the "Administration" WBM page.	
Source	All possible memory media as the source of backup files are displayed. The currently active partition, i.e. the restore destination, does not appear in the list, rather in the table to the far right under "active partition".
Packages	You can select which package(s) you want to restore by selecting the respective checkboxes. To select all packages, mark the "All" checkbox. "Network" is also a special case during the restore process, i.e. the browser sends the backup files from your computer to the IPC as an upload. You have to determine, which files should be used. If "Network" is selected, additional input fields then appear for the names of each backup file. After clicking the **[BROWSE]** button, the file selection menu of your browser appears, allowing you to select the required file. The name is then checked to ensure it follows the naming convention specified by the IPC during the download process. If not, a warning message appears. You can still upload the file, but to increase the security at this point, as already mentioned under Backup, it makes sense to not change the actual file names when downloading. The warning message also appears if you inadvertently select a completely inappropriate file or a backup file that does not belong to another package. Please check this carefully before executing the "Restore" function anyway. For all other selection options in the "Source" menu, the website itself will determine which are available on the source media selected. This takes some time which is why (in the box for package selection) a corresponding message appears. This process is also needed when first calling up the website. The checkboxes of the unavailable packages are grayed out.
Active Partition	The currently active media appears. This is always the restore destination.

9.1.12 "Mass Storage" Page

The "Mass Storage" page provides information about the mass storage media available for the IPC and can perform different actions in this regard. A separate table is displayed for each media available.

Table 42: Description of the Parameters of the "Mass Storage" Page

Mass Storage	
Storage medium/active Partition	The media is specified in the table overview, i.e. "Internal flash", "CF card", "USB1", etc. If active, "Active Partition" appears after the media.
Device	The name of the device in the file system of the operating system.
Volume name	The name of the storage medium.
Bootflag	A graphic and corresponding text display if the memory is bootable or not. Press the **[SET/RESET BOOTFLAG]** button to set or reset the bootflag accordingly. The internal flash memory must always be bootable and therefore the button does not appear.
FAT format Medium	Press the **[START FORMATTING]** to start formating the media with (exactly) one partition in FAT32 format. **NOTICE** **Delete existing data** Any existing data is deleted when formatting the media. The currently active partition and the internal flash memory cannot be formatted. Therefore, the button does not appear.
Volume Name	The partition of your media just formatted receives this name. Linux uses this name, for example, to mount the partition later.

Note

Limited number of USB flash drives
A maximum of 8 USB flash drives connected to the I/O-IPC can be displayed and processed. All other USB flash drives are listed/displayed as "Unknown Medium" and cannot be used by the system.

9.1.13 "Downloads" Page

On this page, you can search for current fieldbus software, program licenses and update scripts using the **[Browse…]** button in the PC file system and import them into the I/O-IPC using **[Download]**. You activate the new data in the I/O-IPC by using the **[Activate]** button.

9.1.14 "Port" Page

On this page for protocol configuration, select the protocol that you would like to use for communication. You have a choice between the following protocols:

- Telnet
  When using the Linux console through ETHERNET
- CODESYS Web server
  For CODESYS Web visualization
- FTP
  For transferring files
- CODESYS
  For accessing CODESYS

To accept your entries, click on the **[SUBMIT]** button.

9.1.15 "MODBUS" Page

On this page, change the MODBUS settings. Choose whether you would like to use MODBUS/UDP and/or MODBUS/TCP as a protocol for process data exchange. You also set the MODBUS/TCP timeout on this page. To accept your entries, click on the **[SUBMIT]** button.

Table 43: Description of the Parameters of the "MODBUS" Page

MODBUS/UDP	
Enabled/Disabled	Activate or deactivate the MODBUS/UDP protocol here.
MODBUS/TCP	
Enabled/Disabled	Activate or deactivate the MODBUS/TCP protocol here.
Timeout (msec)	Here, set the time period for the MODBUS/TCP connection, after which the connection is automatically ended during a break in communication.
MODBUS/RTU	
State	Display of the current MODBUS connection that was selected on the "Administration" page.
Node ID	Selection of a MODBUS node ID in area 1 – 247.
Timeout (msec)	Here, set the time period for the MODBUS/RTU connection after which the connection is automatically ended during a break in communication.
Baud rate	Here, select the transmission speed of the serial interface.
Databit	Selection of the databits to be transferred.
Parity	Activate/deactivate transmission error recognition.
Stop Bits	Here, select the number of stop bits.
Flow Control	Here, set the flow control for hardware and software.

9.1.16 Page „SNMP“

On this page, change the parameters for the "Simple Network Management Protocol" on the SNMP configuration page. To accept your entries, click on the **[SUBMIT]** button or delete them by clicking on **[DELETE]**.

The "Community Name" must be transmitted in the SNMP package header. This assigns a type of access privilege. However, as this is transmitted in plaintext, SNMP in its first versions (v1 and v2c) is considered a very insecure protocol.

IPC supports SNMP in versions v1, v2c, and v3.
v2c offers a few additional functions in comparison with v1, and v3 introduced encryption and an improved authentication. In SNMP v3, exchanging messages is user-related.

Table 44: Description of the parameters for the "SNMP" page

General SNMP Configuration	
General SNMP information data are configured here.	
Name of device	Device name (sysName)
Description	Device description (sysDescription)
Physical location	Device location (sysLocation)
Contact	Email contact address (sysContact)
SNMP v1/v2c Manager Configuration	
Protocol enabled	Activate/deactivate the SNMP protocol for v1/v2c here.
Local Community Name	You specify here the community name for the SNMP manager configuration. The community name can be used to establish relationships between SNMP managers and agents who are respectively referred to as as community and who control identification and access between SNMP participants. The community name can be up to 32 characters long and may not include spaces. To use the SNMP protocol, a valid community name must always be specified. The default community name is "public"
SNMP v1/v2c Trap Receiver Configuration	
A list with the data for all configured "Trap Receivers" for v1/v2c is indicated here. The number of Receivers is in principle not limited in the I/O-IPC. You have the opportunity to delete individually configured Receivers via **[DELETE]**. At the end of the list, you will find a form for compiling a completely new Trap Receiver. Individual configuration data of already existing Receivers cannot be changed.	
IP Address	IP address for the Trap Receiver (Management-Station)
Community Name	Specify here the community name for the trap receiver configuration. The community name can be evaluated by the trap receiver. The community name can be up to 32 characters long and may not include spaces.
Version	SNMP version, via which the Traps are sent: v1 or v2c (Traps via v3 are configured in a different form).

Table 44: Description of the parameters for the "SNMP" page

SNMP v3 Configuration	
A list of all configured v3 users is indicated here. The number of users is in principle not limited by the I/O-IPC. You have the opportunity to delete individually configured Users via **[DELETE]**. At the end of the list, you will find a form for compiling a completely new User. Individual configuration data of already existing Users cannot be changed.	
Security Authentication Name	User Name. This must be unique; an already present User name is not accepted by the new entry. The community name can be up to 32 characters long and may not include spaces.
Authentication Type	Authentication type for SNMP v3 packages. Possible values are: - Use no authentication ("None") - Message Digest 5 ("MD5") - Secure Hash Algorithm ("SHA")
Authentication Key (min. 8 char.)	Key string for authentication. The authentication key must be min. 8 characters and max. 32 characters without blank characters.
Privacy	Encryption algorithm for SNMP Messages. Possible values are: - No encryption ("None") - Data Encryption Standard ("DES") - Advanced Encryption Standard ("AES")
Privacy Key (min. 8 char.)	Key string for encryption ("privacy") of the SNMP message. If nothing is entered here, the “Authentication Key” is automatically used. The Privacy Key must be min. 8 characters and max. 32 characters without blank characters.
Notification Receiver IP	IP address for a Trap Receiver for v3 Traps. In case no v3 Traps should be sent for this user, the field remains empty.

9.1.17 I/O Configuration

On this page, the I/O module configuration connected to the I/O-IPC is shown with the process values of the individual I/O modules.

Table 45: Description of the parameters on the "I/O Configuration" page

I/O configuration and vales	
Pos	Position of the I/O module connected to the I/O-IPC. Passive I/O modules do not appear in WBM (e.g. 750-600, -602, …).
Module	Product number of the I/O module or abbreviation.
Type	Description of which is the I/O module in question (8DI, 4AO, etc.).
Channel	Specification of the module position and channel number of the I/O module.
Values	Display of the process data at the time of the last update of the "I/O Configuration" page. To display the current process data, update the page.

9.1.18 "WebVisu" Page

On the "WebVisu" page, you select whether the WBM or CODESYS web visualization should appear when entering the IP address of the I/O-IPC.
To save all settings made on the page, click the **[SUBMIT]** button. The I/O-IPC must be restarted or reset to apply the settings. To display the CODESYS Web visualization, also update the Internet browser.

In order to reach the WBM again, in addition to the IP address, enter the port number "8080": http://**192.168.2.17:8080** (socket address).

You can obtain additional information on CODESYS web visualization in the section of the same name.

9.2 Configuration with a Terminal Program

You can configure the I/O-IPC both via ETHERNET using Telnet as well as via the Linux console using the RS-232 interface via the IPC Configuration Tool. To call up the IPC Configuration Tool for both variants, log into the Linux console and enter the command *ipcconfig*. See section "Access over Telnet" and section "Access via RS-232 Interface and Terminal Program".

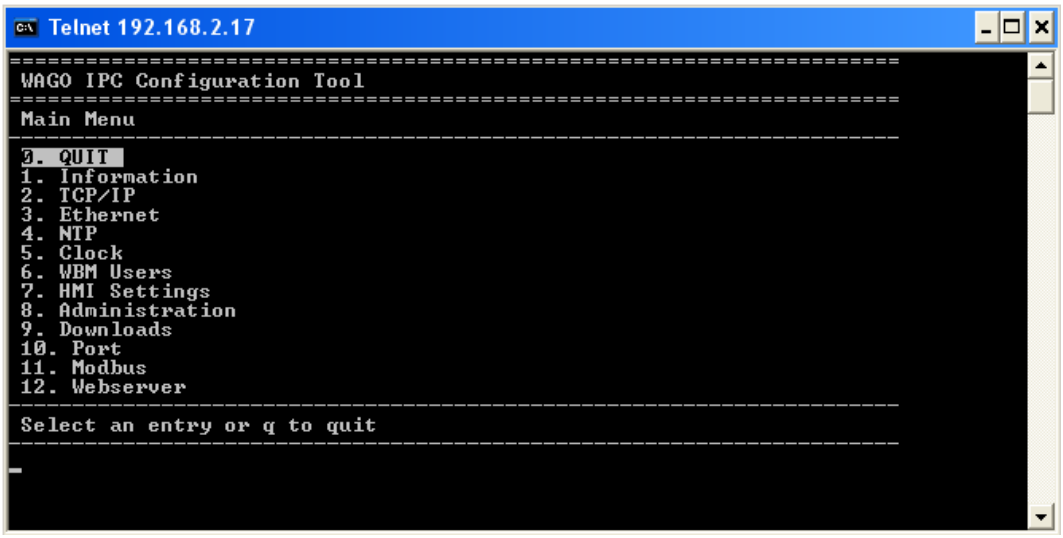


Figure 37: Access to the IPC Configuration Tool using Telnet

9.3 Configuration with Touch screen/Monitor and USB Keyboard

Preparation:
You have connected a monitor to the DVI-I interface and a keyboard to the USB interface of the I/O-IPC. For more information, see section „Access over keyboard and monitor (DVI- and USB interface)“.

Using the key combination **[Alt] + [F3]**, open the third console of the I/O-IPC on which the IPC configuration tool is located. You cannot quit this Linux console. Therefore, "QUIT" on the navigation bar has no function.

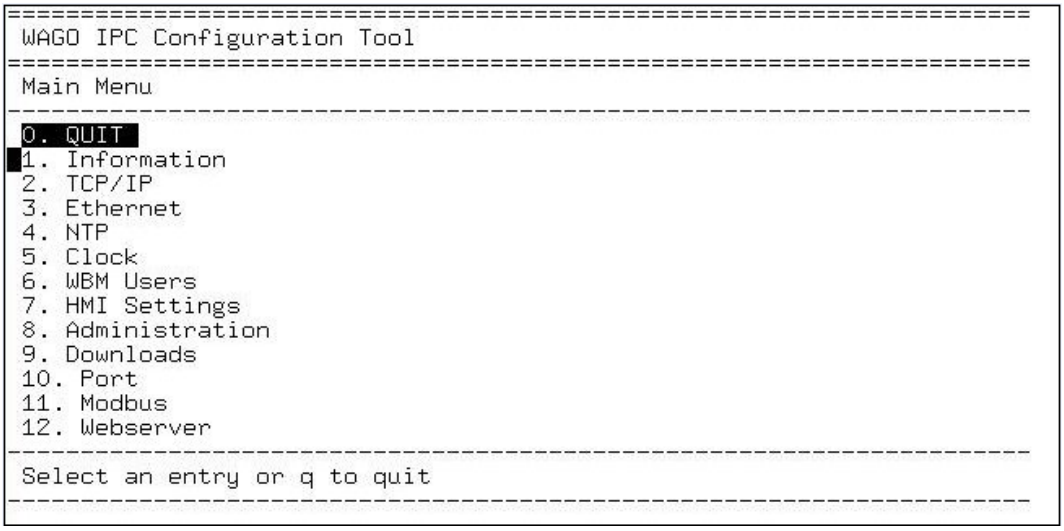


Figure 38: Start screen of the WAGO IPC configuration tool

10 MODBUS/TCP

The modular concept of the 750 Series makes it possible to connect up to 250 (via internal data bus extension) I/O modules to the I/O-IPC. This variable construction and the large number of different I/O modules prevent a static assignment of input and output data to fixed MODBUS addresses, however. The only exceptions are the "digital" MODBUS services. For these, the MODBUS address is identical to the channel number; i.e., the 47th digital input can always be found at MODBUS address "46".

By adding or removing I/O modules, the structure of the process images is changed which also changes the MODBUS addresses of individual I/O module channels.
MODBUS communication is performed via service calls, the MODBUS master (client) sending a request telegram to port 502 of the MODBUS slave (server). The MODBUS slave returns the result of the service call in a response telegram to the MODBUS master.
The most important elements of a MODBUS telegram are:

Table 46: Elements of a MODBUS telegram

Term	Description
UnitID	Identification of which device is to be activated (<FF)
FunctionCode (FC)	Service identification: read or write operation in bits or words
Address	Operation start address
Count	Number of bits or words depending on the service
[Data]	Process data

The service identification or "FunctionCode" (FC) first determines whether the service is a read or write operation. It also determines the basic data type to which the operation is to be applied. Therefore, the meaning of the parameters "Address" and "Count" is also dependent on the function code. Thus "address :=3" can stand for a bit or a word in the input or output process image.

The MODBUS/TCP protocol is largely based on the following basic data types:

Table 47: Basic data types of MODBUS

Data type:	Length	Description
Discrete Inputs	1 bits	Digital Inputs
Coils	1 bits	Digital outputs:
Input Register	16 bits	Analog input data
Holding Register	16 bits	Analog output data

One or more "FunctionCodes" are defined for each basic data type.

10.1 Process Data of the MODBUS Server

The first analog output or input or the digital (if there is no analog one) is reached via the word services of the MODBUS server.

A particularity of access via MODBUS is that with the "digital" MODBUS services on the I/O-IPC address 0, you always access the first digital output or input of the internal data bus process image, although digital and analog process data of the I/O-IPC and the I/O modules are combined into one process image. For information about the length of the process data, please see section "Process Data Architecture for MODBUS/TCP."

⚠ WARNING

Activating "Control Mode“ in WAGO-IO-CHECK!
Using WAGO-IO-CHECK, you can overwrite parameters and process data in "Control Mode" regardless of whether the fieldbus or PLC functionalities are enabled or disabled. By doing so, machine components may be placed in a dangerous state and personnel and machines may be at risk.

Before changing parameters and process data, ensure that the machine components are in a safe and defined state and switch off the higher-level controller. Also ensure before start-up that no personnel remain in the danger area of the machine components.

10.2 Accessing the Process Image via MODBUS Functions

The following table describes the MODBUS function codes with which you can access the address ranges of the process image for the inputs and outputs connected to the internal data bus and the fieldbus:

Table 48: MODBUS Function Codes

FC	Name	Description
FC1	Read coils	Several digital output values are read out
FC2	Read inputs discrete	Several digital input values are read
FC3	Read holding registers	Several analog output values are read
FC4	Read input registers	Several analog input values are read
FC5	Write coil	A single digital output value is written
FC6	Write single register	A single analog output value is written
FC15	Force multiple coils	Several digital output values are written
FC16	Write multiple registers	Several analog output values are written
FC23	Read/write multiple registers	Read and write operation on analog input and output values

10.2.1 Register Services

With the register services, determine or change the statuses of analog input and output terminals for the following address ranges:

Table 49: Reading Analog Input Terminals Using FC3, FC4, FC23

MODBUS Address	Addresses in CODESYS	Description
0x0000 – 0x00FF (0 – 255)	%IW0 … %IW255	Reading analog or digital input values. Physical address space of 256 words for the input data.
0x100 – 0x1FF (256 – 511)	%QW256 … %QW511	Read the PFC variables
0x1000 – 0x2FFF (4096 – 12287)	See MODBUS configuration register	MODBUS Configuration Register
0x3000 – 0x3FFF (12288 – 16384) or can be set to a maximum of 0xFFFF (65534)	%MW0 … %MW4095 %MW0 … %MW53247	Retain memory (8 kB) Non-volatile PLC variables Retain memory (24 kB) Non-volatile PLC variables

Table 50: Writing of Analog Output Terminals Using FC6, FC16, FC23

MODBUS Address	Addresses in CODESYS	Description
0x0000 – 0x00FF (0 – 255)	%QW0 … %QW255	Writing analog or digital output values. Physical address space of 256 words for the output data.
0x100 – 0x1FF (256 – 511)	%IW256 … %IW511	Write the PFC variables
0x1000 – 0x2FFF (4096 – 12287)	See MODBUS configuration register	MODBUS Configuration Register
0x3000 – 0x3FFF (12288 – 16384) or can be set to a maximum of 0xFFFF (65534)	%MW0 … %MW4095 %MW0 … %MW53247	Retain memory (8 kB) Non-volatile PLC variables Retain memory (24 kB) Non-volatile PLC variables

10.2.2 Bit Services

With the digital bit services, determine or change the statuses of digital input and output terminals for the following address ranges:

Table 51: Reading of Digital Input Terminals Using FC1, FC2

MODBUS Address	Addresses in CODESYS	Description
0x0000 – 0x01FF (0 – 511)	%IX 0.0 ... %IX 32.15 + offset value	Input process image Bit-based addressing in MODBUS begins with the first digital I/O module at the internal data bus. If analog I/O modules are used, the address range allocated by these modules is skipped over by the addressing function (offset value).
0x0400 – 0x0401 (1024 – 1025)	%IX2300.0 … %IX2300.1	Integrated digital inputs
0x3000 – 0x7FFF (12288 – 32750)	%MX0.0 … %MX1279.15	Retain memory (8 kB) Non-volatile PLC variables

Table 52: Writing of Digital Output Terminals Using FC5, FC15

MODBUS Address	Addresses in CODESYS	Description
0x0000 – 0x01FF (0 – 511)	%QX 0.0 ... %QX 32.15 + offset value	512 bits of digital output data. Bit-based addressing in MODBUS begins with the first digital I/O module at the internal data bus. If analog I/O modules are used, the address range allocated by these modules is skipped over by the addressing function (offset value).
0x0400 – 0x0401 (1024 – 1025)	%QX2300.0 … %QX2003.1	Integrated digital outputs
0x3000 – 0x7FFF (12288 – 32750)	%MX0.0 … %MX1023.15	Retain memory (8 kB) Non-volatile PLC variables

10.3 Configuration Tab

By using the MODBUS configuration register, you can configure the I/O-IPC and read out information through it.

Table 53: Configuration register

MODBUS-Address	Length [Word]	Access	Description
0x1031 (4145)	3	Read	MAC address of ETHERNET interface X9.
0x1034 (4148)	3	Read	MAC address of ETHERNET interface X8.
0x1030 (4144)	1	Read/write	Here, set the time period for the MODBUS connection, after which the connection is automatically ended during a break in communication.

10.4 Addressing Example

The following addressing example clarifies the access to the process image:

Table 54: Arrangement of the I/O modules for the addressing example

I/O-IPC	750-400	750-554	750-402	750-504	750-454	750-650	750-468	750-600
	1	2	3	4	5	6	7	8

Table 55: Addressing example

I/O Module		Input Data		Output Data		Description
Type	**C***	**FC3, FC4**	**FC1, FC2**	**FC6**	**FC5**	
750-400	1	0008	00000			**2DI, 24 V, 3 ms:** 1. Digital I/O module with a data width of 2 bits. Since the analog input modules already occupy the first 8 words of the input process image, the 2 bits occupy the lowest-value bits of the 8th word.
	2		00001			
750-554	1			00000		**2AO, 4 – 20 mA:** 1. Analog output module with a data width of 2 words. This module occupies the first 2 words in the output process image.
	2			00001		
750-402	1	0008	00002			**4DI, 24 V:** 2. Digital input module with a data width of 4 bits. These are added to the 2 bits of the 750-400 and stored in the 8th word of the input process image.
	2		00003			
	3		00004			
	4		00005			
750-504	1			00004	0000	**4DO, 24 V:** 1. Digital output module with a data width of 4 bits. Since the analog output module already occupies the first 4 words of the output process image, the 4 bits occupy the lowest-value bits of the 4th word.
	2				0001	
	3				0002	
	4				0003	
750-454	1	0000				**2AI, 4 – 20 mA:** 1. Analog input module with a data width of 2 words. This module occupies the first 2 words in the input process image.
	2	0001				
750-650	1	0002		00002		**RS232, C 9600/8/N/1:** The serial interface module is an analog input and output module, which displays 2 words apiece both in the input process image as well as in the output process image.
		0003		00003		
750-468	1	0004				**4AI, 0 – 10 V S.E:** 2. Analog input module with a data width of 4 words. Since the 750-454 and 750-650 analog input and output modules already occupy the first 4 words of the input process image, the 4 words of this I/O module are added behind the others.
	2	0005				
	3	0006				
	4	0007				

Table 55: Addressing example

I/O Module		Input Data		Output Data		Description
Type	**C***	**FC3, FC4**	**FC1, FC2**	**FC6**	**FC5**	
750-600						**End module** The passive 750-600 End Module does not transmit any data.

	Analog input/output modules
	Digital input/output modules

*C: Number of the input/output

11 CODESYS 2.3 Runtime Environment

11.1 Process Images

A process image is a memory area in which the process data is stored in a defined sequence. The process image consists of the I/O modules attached to the internal bus, the PFC variables, the bit memory address area and the slaves attached to the fieldbus.

Access to the process images is different with MODBUS and CODESYS. For information about access to the process data of the I/O modules connected to the I/O-IPC and slaves, please see sections “Accessing the Process Image via MODBUS/TCP Functions” and “Access to the Process Images of the Input and Output Data via CODESYS 2.3”.

11.1.1 Process Image for the I/O Modules Connected to the I/O-IPC

After the I/O-IPC is started up, it automatically detects all connected I/O modules. The I/O-IPC generates a process image from this, which is divided into an input data and output data area with a maximum of 500 bytes.
The analog input and output data is stored first word by word in the process image. Next, the bits of the digital input and output data are combined into words and stored behind the analog data in the process image.

Note

Data width of an I/O module
The data width of an I/O module is between 0 and 48 bytes. For detailed information about the respective process data width of individual I/O modules, see section "Process Data Architecture for MODBUS/TCP".

Note

Process data of the I/O modules
Check the I/O module process data whenever you add or remove the modules to/from the I/O-IPC. Changing the I/O module topology results in an adjustment of the process image since the process data addresses also change.

11.1.2 Process Image for the Slaves Connected to the I/O-IPC

Up to 126 slaves can be connected to the I/O-IPC. The I/O-IPC can receive input data up to a size of 3584 bytes from the slaves and send 3584 bytes of output data to the slaves. Observer the network guidelines of the fieldbus used. You perform the fieldbus configuration together with the CODESYS controller configuration (See section "Creating the PLC Configuration").

11.2 Syntax of Logical Addresses

Access to individual memory elements according to IEC 61131-3 is done by the use of special symbols:

Table 56: Syntax of Logical Addresses

Position	Prefix	Designation	Comments
1	%	Starts the absolute address	-
2	I	Input	
	Q	Output	
	M	Flag	
3	X	Single bit	Data width
	B	Byte (8 bits)	
	W	Word (16 bits)	
	D	Double word (32 bits)	
4		Address	

Two examples:

Addressing by word %QW27 (28th word)
Addressing by bit %IX1.9 (10th bit in word 2)

Enter the character string of the absolute address without empty spaces.

11.3 Access to the Process Images of the Input and Output Data via CODESYS 2.3

The following table describes the possibilities with which you can access the address ranges of the process image for the inputs and outputs connected to the internal data bus and the fieldbus**

Table 57: Memory Areas for the Input and Output Data of CODESYS

Memory Area	Description	Access via MODBUS-TCP	Access via PLC	Logical Address Space
Input process image	Image of the local input terminals (internal data bus, I/O module 1 through 1 through 64*).	Read	Read	**Word** %IW0 to %IW255
				Byte %IB0 to %IB511
Output process image	Image of the local output terminals (internal data bus, I/O modules 1 through 64*).	Write	Read/ Write	**Word** %QW0 to %QW255
				Byte %QB0 to %QB511
PLC input process image	Image of the PLC input variables that can be accessed using MODBUS TCP.	Read/ Write	Read	**Word** %IW256 to %IW511
				Byte %IB512 to %IB1023
PLC output process image	Image of the PLC output variables that can be accessed with MODBUS TCP.	Read	Read/ Write	**Word** %QW256 to %QW511
				Byte %QB512 to %QB1023
Integrated digital input	Image of the digital I/O-IPC input bits 0.1.	-	Read	**Bit** %IX 2300.0 to %IX 2300.1
Integrated digital output	Image of the digital I/O-IPC output bits 0.1.	-	Read/ Write	**Bit** %QX 2300.0 to %QX 2300.1
Fieldbus Input Variables**	Input variables of the configured fieldbus.	-	Read	**Word** %IW2400 to %IW31750
				Byte %IB4800 to %IB65535
Fieldbus output variables**	Output variables of the configured fieldbus.	-	Write/ Read	**Word** %QW2400 to %QW31750
				Byte %QB4800 to %QB65535

Table 57: Memory Areas for the Input and Output Data of CODESYS

Memory Area	Description	Access via MODBUS-TCP	Access via PLC	Logical Address Space
Flag variables**	8 kB residual memory in the SRAM. Declared with "AT %M<Address>". (Expandable to 24 kB)	Read/ Write	Read/ Write	**Word** %MW0 to %MW 4095 **Byte** %MB0 to %MB8190 (%MW12287)
Retain variables	Symbolic addressable retain memory in the SRAM: 127 kBytes		Read/ Write	

* The use of up to 250 I/O modules is possible with the WAGO internal data bus extension modules.
** only for I/O-IPC with fieldbus connections

The total memory size for flag and retain variables is 127 kB. Use bit-oriented addressing and observe that the base address is word-oriented. The bits are in the areas 0 to 15.

⚠ WARNING

Activating "Control Mode" in WAGO-IO-CHECK!
Using WAGO-IO-CHECK, you can overwrite parameters and process data in "Control Mode" regardless of whether the fieldbus or PLC functionalities are enabled or disabled. By doing so, machine components may be placed in a dangerous state and personnel and machines may be at risk.
Before changing parameters and process data, ensure that the machine components are in a safe and defined state and switch off the higher-level controller. Also ensure before start-up that no personnel remain in the danger area of the machine components.

Adaptation of the remanent memory area

When creating a project, a configuration window for the selection of the destination system appears (see Section "Designing a Project and Selecting the Target System").

1. To adapt the remanent memory area, click on the "Memory Layout" tab in the "Target Settings" configuration window.

2. Enter the following values in the "Memory" and "Retain" field:

- **Remanent memory area of 8 kB**
  Memory: 16#2000 (8 kB)
  Retain: 16#1DF00 (119 kB)
  Sum: 16#1FF00 (127 kB))

- **Remanent memory area of 16 kB**
  Memory: 16#4000 (16 kB)
  Retain: 16#1BF00 (111 kB)
  Sum: 16#1FF00 (127 kB)

- **Remanent memory area of 24 kB**
  Memory: 16#6000 (24 kB)
  Retain: 16#19F00 (103 kB)
  Sum: 16#1FF00 (127 kB)

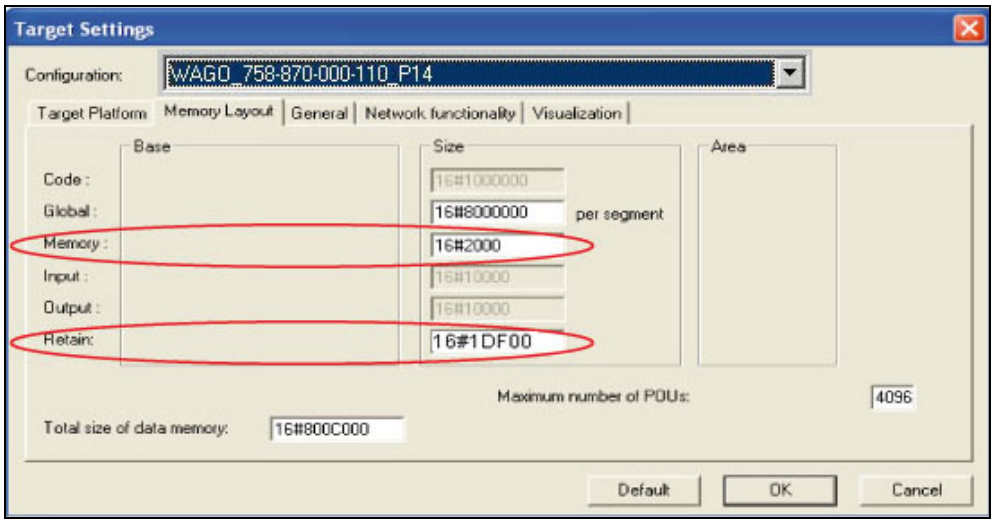


Figure 39: Adaptation of the remanent memory area

11.4 Addressing Example

The following addressing example clarifies the access to the process image:

Table 58: Arrangement of the I/O modules for the addressing example

I/O-IPC	750-400	750-554	750-402	750-504	750-454	750-650	750-468	750-600
	1	2	3	4	5	6	7	8

Table 59: Addressing example

I/O Module		Input Data		Output Data		Description
Type	C*					
750-400	1		%IX8.0			**2DI, 24 V, 3 ms:** 1. Digital I/O module with a data width of 2 bits. Since the analog input modules already occupy the first 8 words of the input process image, the 2 bits occupy the lowest-value bits of the 8th word.
	2		%IX8.1			
750-554	1			%QW0		**2AO, 4 – 20 mA:** 1. Analog output module with a data width of 2 words. This module occupies the first 2 words in the output process image.
	2			%QW1		
750-402	1		%IX8.2			**4DI, 24 V:** 2. Digital input module with a data width of 4 bits. These are added to the 2 bits of the 750-400 and stored in the 8th word of the input process image.
	2		%IX8.3			
	3		%IX8.4			
	4		%IX8.5			
750-504	1				%QX4.0	**4DO, 24 V:** 1. Digital output module with a data width of 4 bits. Since the analog output module already occupies the first 4 words of the output process image, the 4 bits occupy the lowest-value bits of the 4th word.
	2				%QX4.1	
	3				%QX4.2	
	4				%QX4.3	
750-454	1	%IW0				**2AI, 4 – 20 mA:** 1. Analog input module with a data width of 2 words. This module occupies the first 2 words in the input process image.
	2	%IW1				
750-650	1	%IW2				**RS232, C 9600/8/N/1:** The serial interface module is an analog input and output module, which displays 2 words apiece both in the input process image as well as in the output process image.
		%IW3				
				%QW2		
				%QW3		
750-468	1	%IW4				**4AI, 0 – 10 V S.E:** 2. Analog input module with a data width of 4 words. Since the 750-454 and 750-650 analog input and output modules already occupy the first 4 words of the input process image, the 4 words of this I/O module are added behind the others.
	2	%IW5				
	3	%IW6				
	4	%IW7				

Table 59: Addressing example

I/O Module		Input Data		Output Data		Description
Type	C*					
750-600						**End module** The passive 750-600 End Module does not transmit any data.

	Analog input/output modules
	Digital input/output modules

*C: Number of the input/output

11.5 Installing the Programming System CODESYS 2.3

The WAGO target files are installed during the installation of CODESYS. These contain all device-specific information for the WAGO 750/758 product series.

Proceed as described below to install the programming software CODESYS 2.3 on the I/O-IPC.

1. Insert the CD-ROM "WAGO-I/O-PRO CAA" into your computer drive.

2. To install the programming system, follow the instructions that appear on your screen. A successful installation is indicated by a CODESYS icon on your desktop.

11.6 The First Program with CODESYS 2.3

This section explains, through examples, the relevant steps required for the creation of a CODESYS project. It is intended as a set of quick start instructions and does not address the full functional range of CODESYS 2.3.

Note

Further informations
Please see the manual "Handbook for PLC Programming with CODESYS 2.3" on the CD "WAGO-I/O-PRO CAA" (759-911) for a detailed description of the full functional range.

11.6.1 Start the CODESYS Programming System

Start CODESYS by double clicking on the CODESYS pictogram on your desktop or through your operating system using your start menu. To do this, click on the "Start" button and choose **Program** >
WAGO Software > **CODESYS for Automation Alliance** > **CODESYS V2.3**.

11.6.2 Designing a Project and Selecting the Target System

1. Click on **File** in the menu bar and select **New**. The "Set Target System" window opens. Here, all available target systems that can be programmed with CODESYS 2.3 are listed.

2. Open the selection box in the "Set Target System" window and select the I/O-IPC you are using. In this example, it is an I/O-IPC of the PROFIBUS-Master "WAGO_758-876-000-111" type.

3. Click on the **[OK]** button. The "Set Target System" configuration window opens.

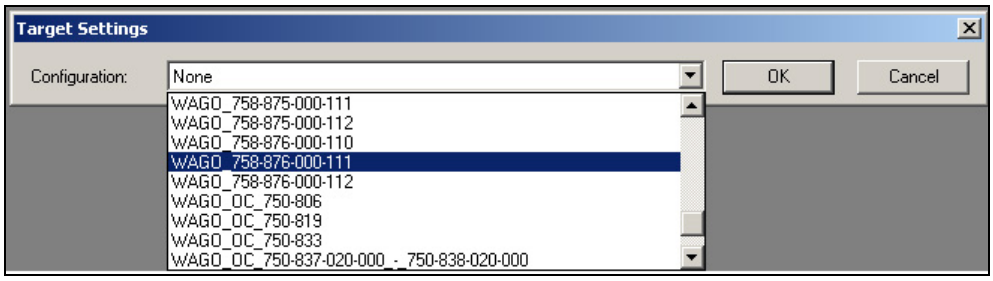


Figure 40: Target Settings (1)

4. To accept the default configuration for the I/O-IPC, click on the **[OK]** button. The "New Component" window opens.

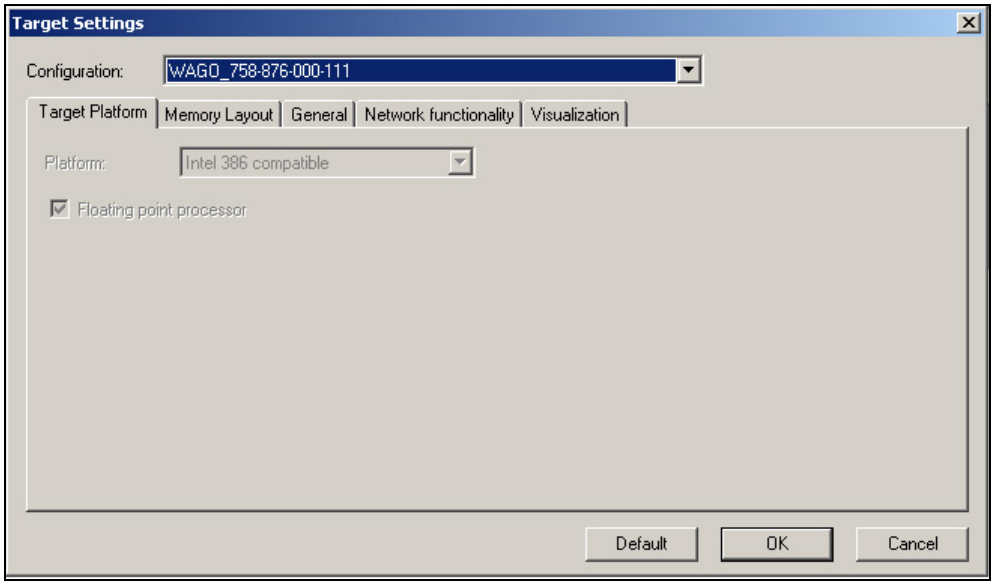


Figure 41: Target Settings (2)

5. Create a program function block in the "New POU" window. In this example, a new function block, "PLC_PRG", is created in the "ST" programming language.

6. Click on **[OK]** to create the project. The programming interface opens.

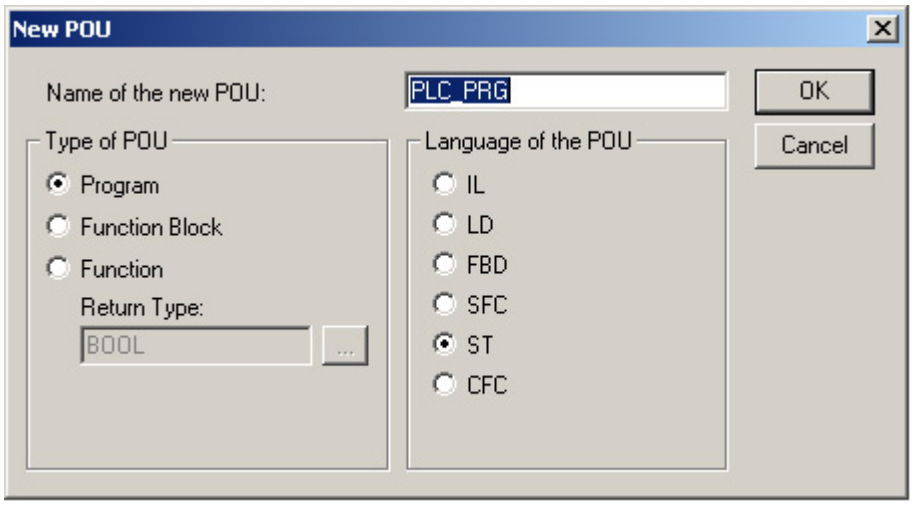


Figure 42: Designing a new function block

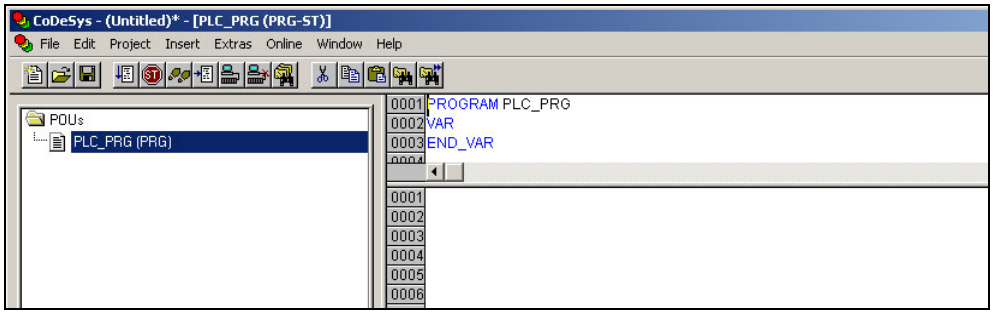


Figure 43: Programming interface with the program function block PLC_PRG

11.6.3 Creating the PLC Configuration

Note

Procedure for the creation of the PLC configuration
Use an I/O-IPC with a fieldbus connection, then proceed with the creation of the PLC configuration as described in Section "CANopen Master in CODESYS 2.3" or "PROFIBUS Master in CODESYS 2.3". For I/O-IPC without fieldbus connections proceed with the creation of the PLC configuration as described in this Section.

The PLC configuration is used to configure the I/O-IPC along with the connected I/O modules and to declare variables for accessing the inputs and outputs of the I/O modules. Proceed as follows:

1. Click on the "Resources" tab.

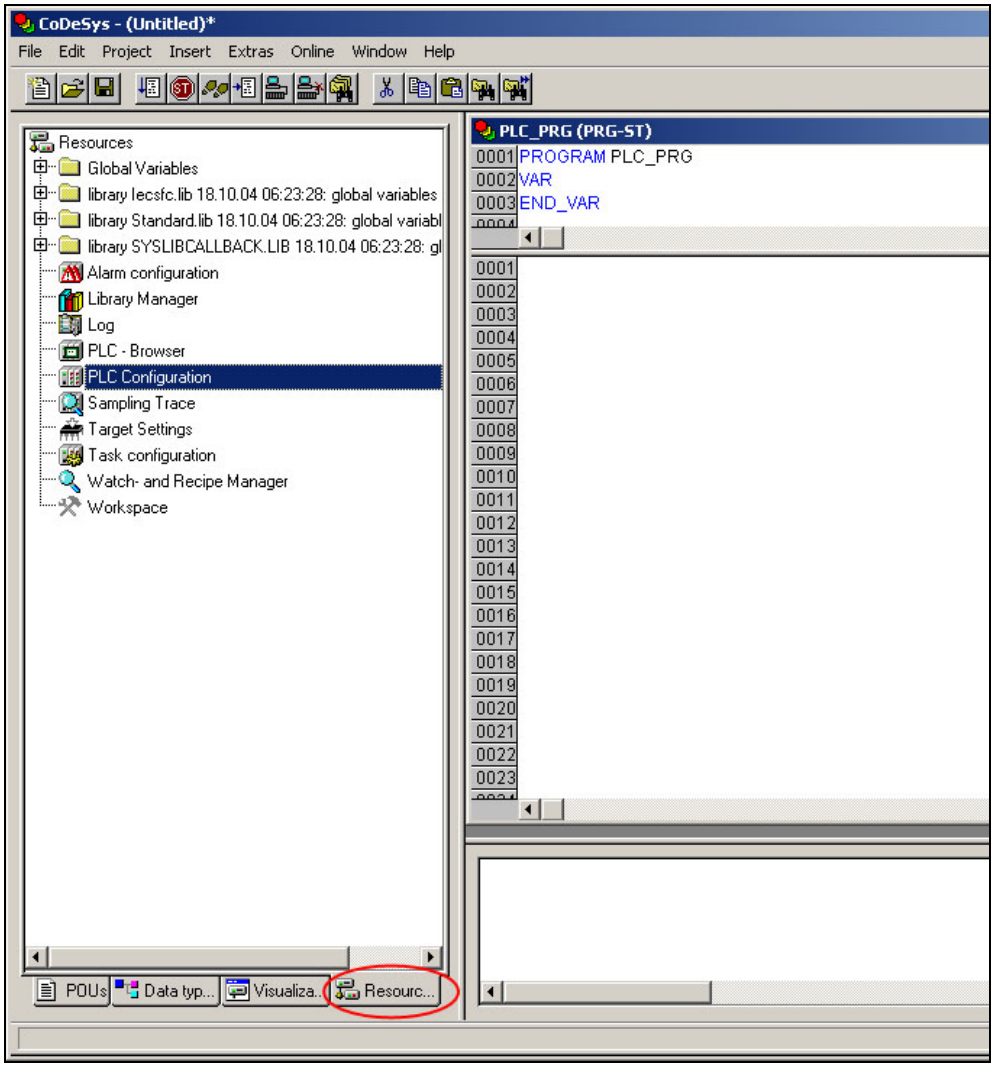


Figure 44: "Resources" tab"

2. Double click on "PLC Configuration" in the left field. This opens the PLC configuration of the I/O-IPC.

3. Right click on the entry "K-Bus[Fix]" and select "Edit" in the context menu.

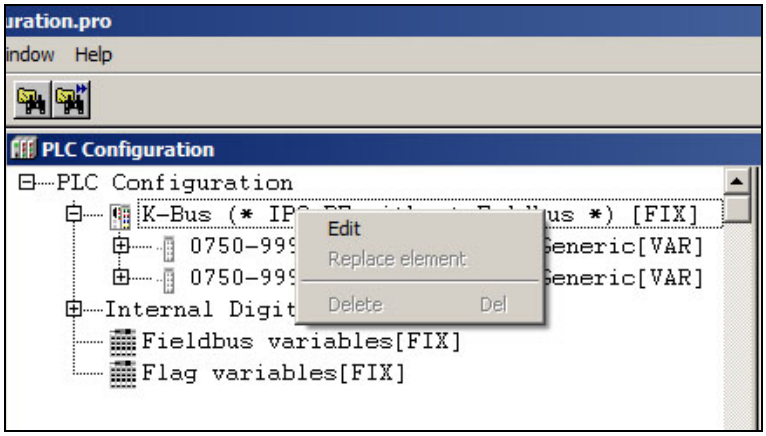


Figure 45: PLC configuration: Edit

4. The "Configuration" dialog opens.

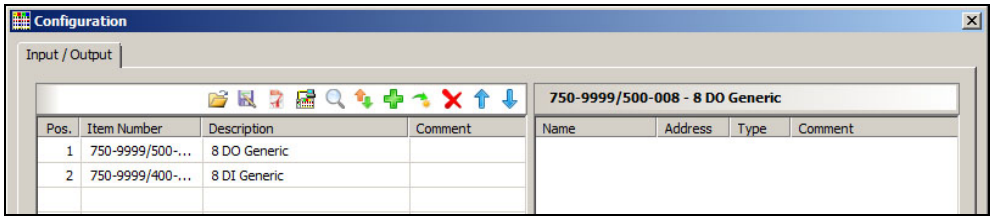


Figure 46: Configuration

5. You can use the **[Add]** button to add new I/O modules to manually define or change the configuration.

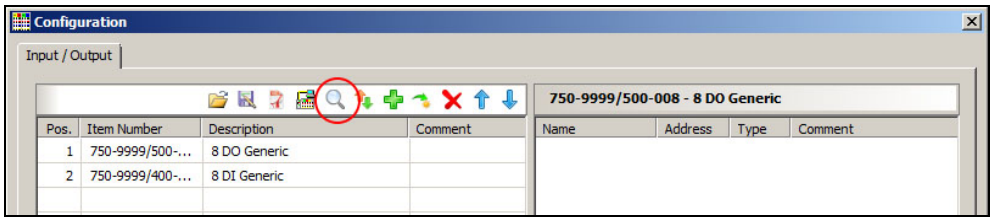


Figure 47: „Add“ button

6. You can select a module in the new “Module selection” window that then appears.

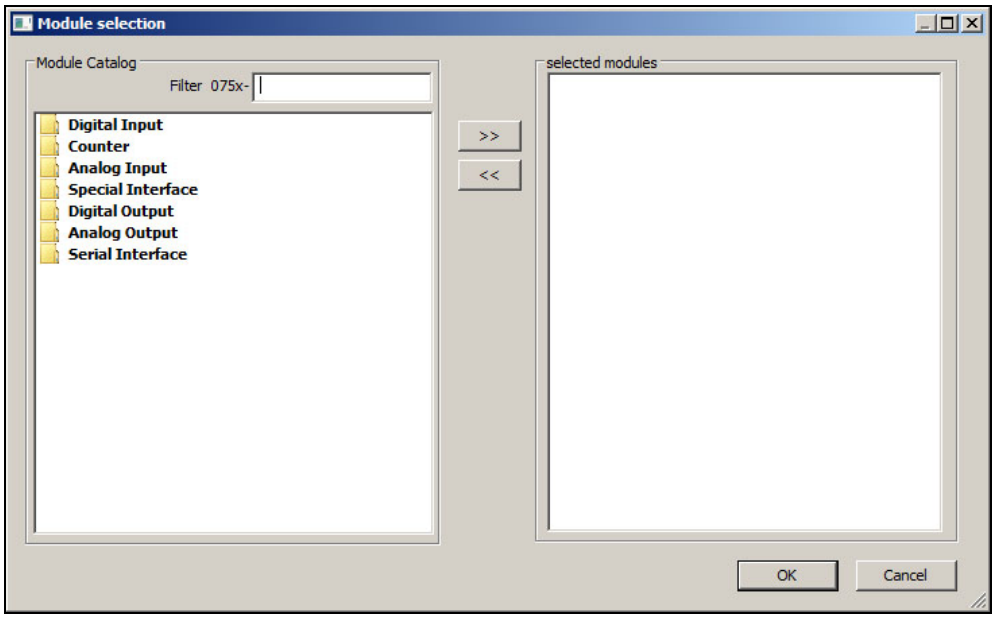


Figure 48: Window „Module selection“

7. You can change the position of an I/O module in the right window by marking it and then moving it up or down using the arrow button on the right edge of the window.

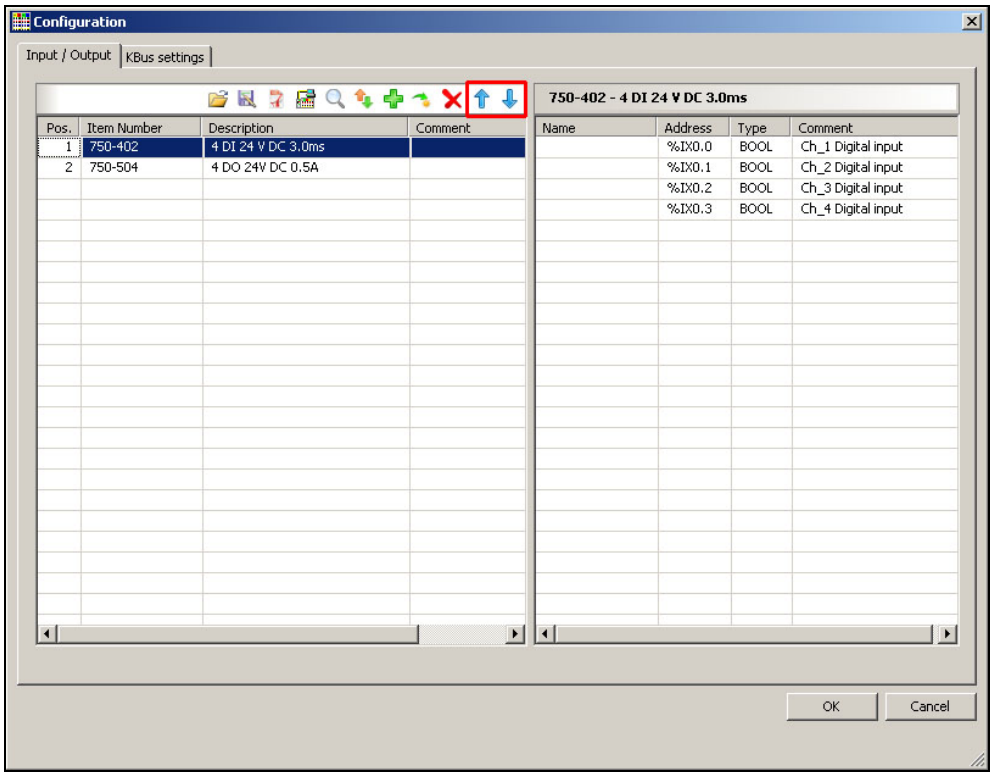


Figure 49: I/O Configurator with defined I/O modules

8. The individual inputs and outputs of the selected I/O module are displayed in the right half of the configuration window. Here, you can declare a dedicated variable in the “Name” column for each input and output, e.g., “Output_1”, “Output_2”, “Input_1”, “Input_2”.

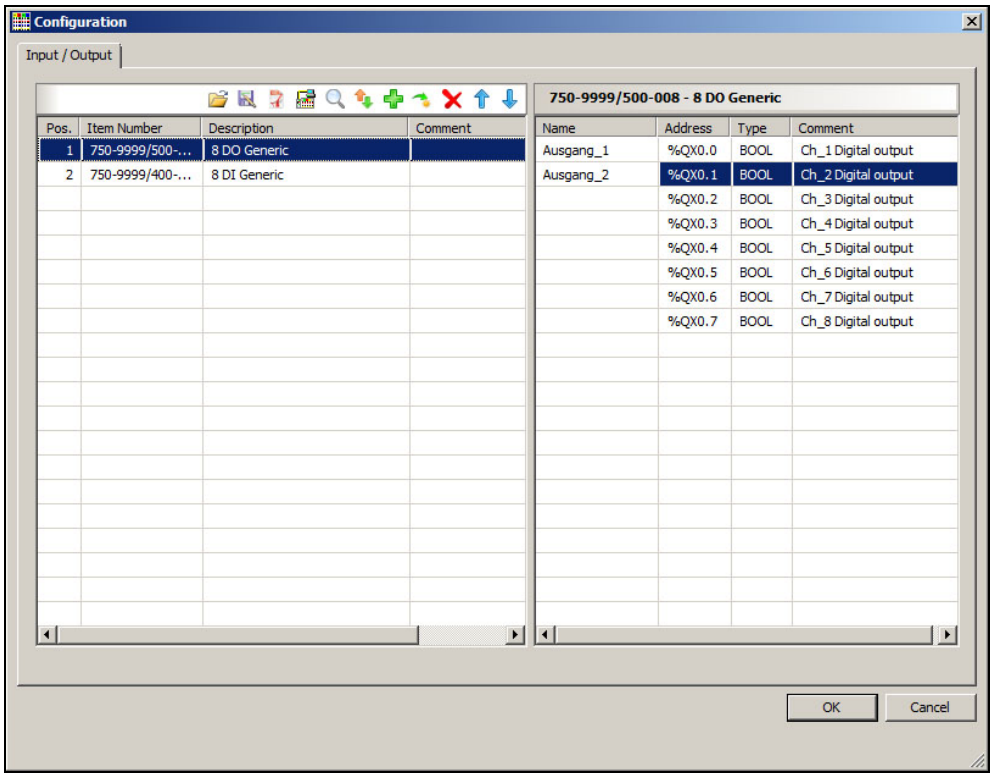


Figure 50: Variable declaration

9. To close the I/O Configurator, click **[OK]**.

10. The added I/O modules appear in the control configuration under “K-Bus[FIX]” with their associated fixed addresses and, where applicable, their previously set variable name.

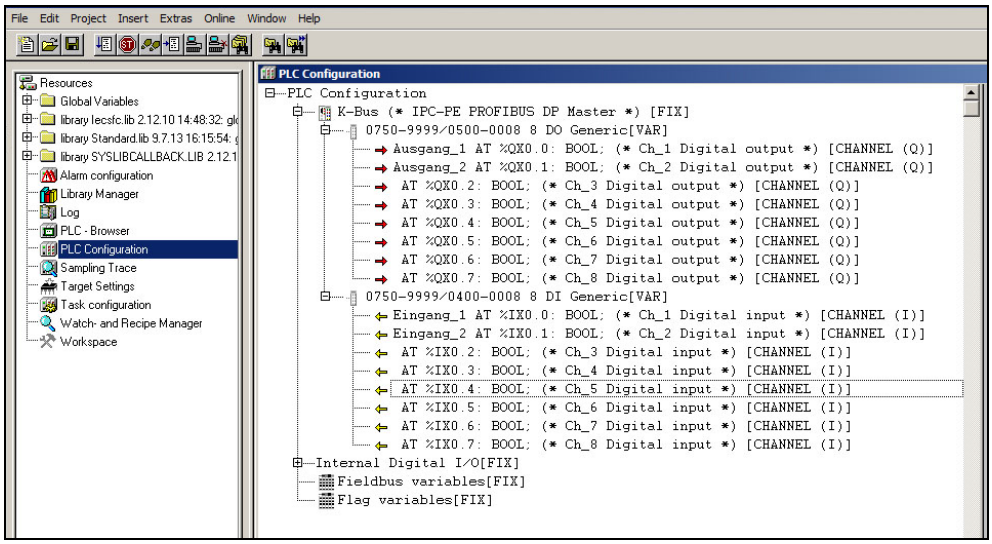


Figure 51: Control configuration: I/O modules with their associated addresses

11.6.4 Editing a Program Function Block

To edit the program component PLC_PRG, change to the "POUs" tab and double click on the program function block PLC_PRG.

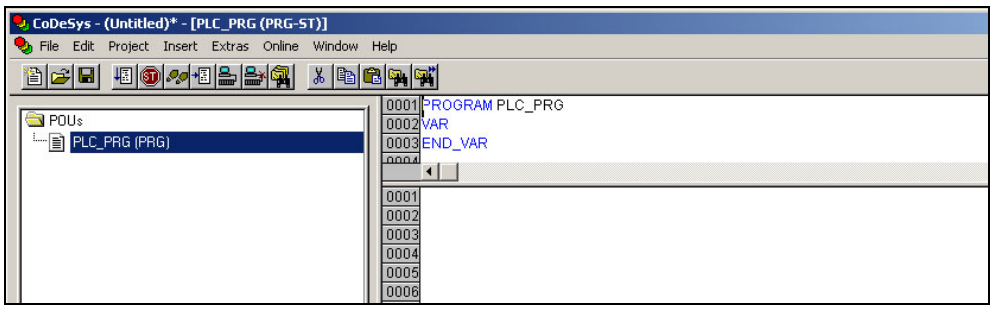


Figure 52: Program function block

The following example is intended to illustrate the editing of the program function block. To do this, an input is assigned to an output:

1. Press **[F2]** to open the input assistant or right click and select the "Input Assistant…" context menu.

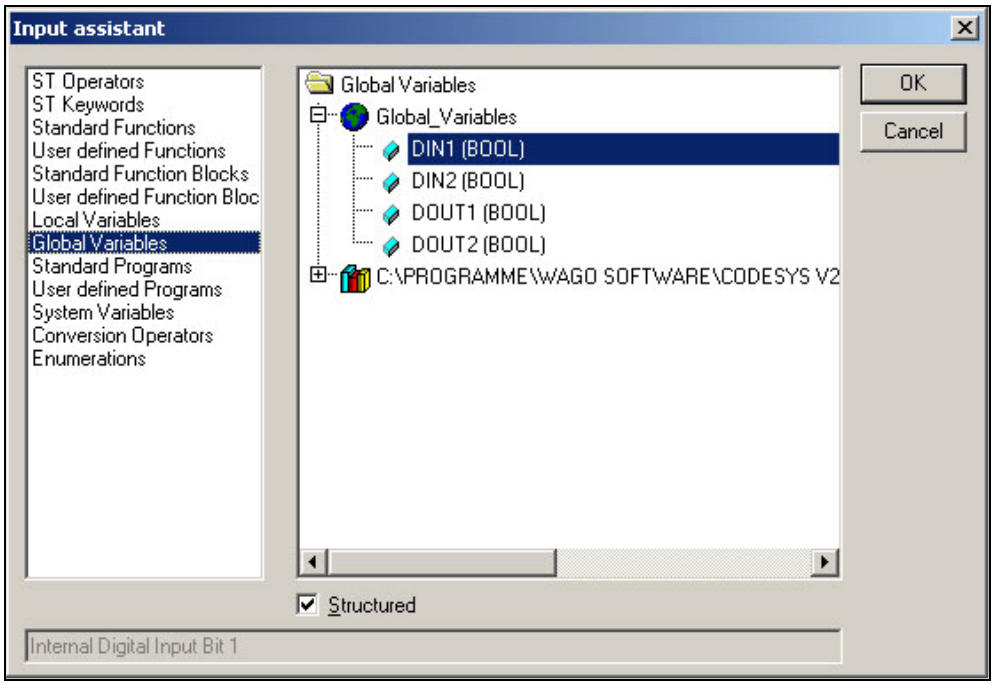


Figure 53: Input assistant for selecting variables

2. Select the previously declared variable "K-Bus_Do_01" under "Global Variables" and click to on **[OK]** to add it.

3. Enter the assignment**:=** behind the variable name.

4. Repeat step 2 for the variable "K-Bus_DI_01".

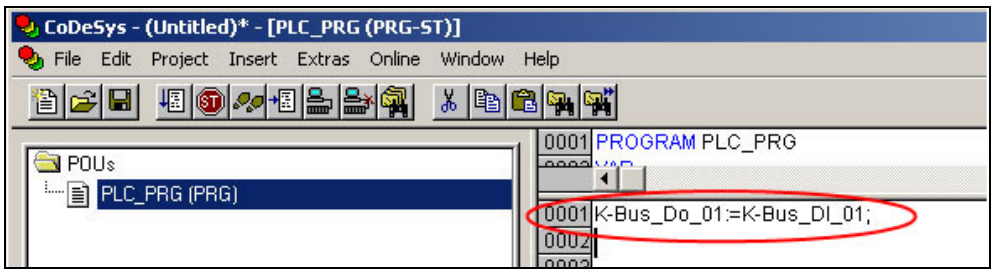


Figure 54: Example of an assignment

5. To compile, click on "**Project**" in the menu bar and select "**Rebuild all**".

11.6.5 Loading and Executing the PLC Program in Control (ETHERNET)

Prerequisite:
The simulation mode is disabled (**Online > Simulation**).
You have connected the PC to the ETHERNET interface of the I/O-IPC via an ETHERNET cable (RJ-45).

1. Click in the menu bar on **Online** and select **Communication Parameters ...**
   The "Communication Parameters" window opens.

2. To create a communication, click on **[New…]** in the "Communication Parameters" window. The window for creating a new channel opens.

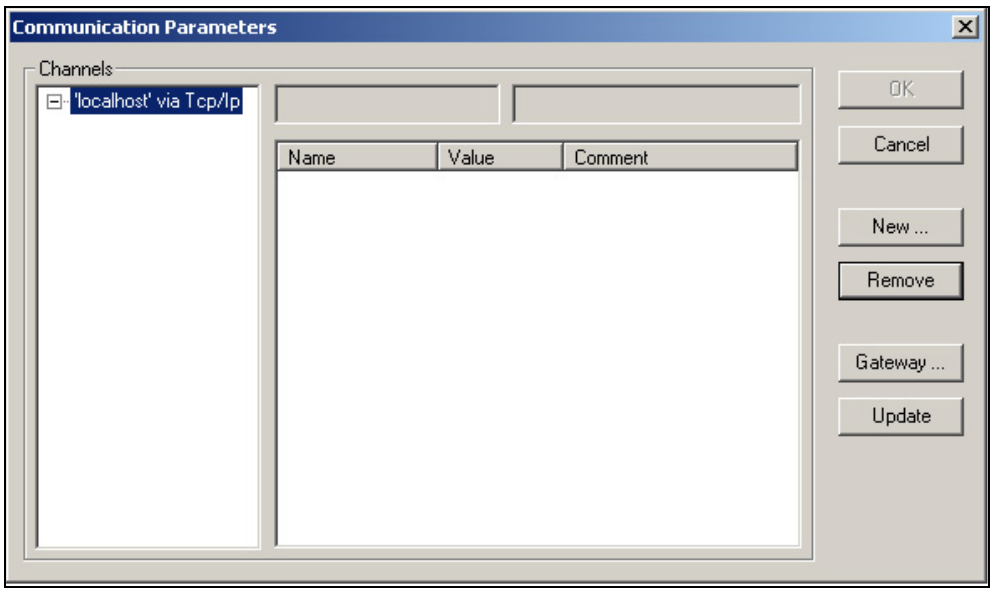


Figure 55: Creating a new communication channel 1

3. Enter any name for your I/O-IPC in the "Name" field and click "Tcp/Ip...". Then click **[OK]**.

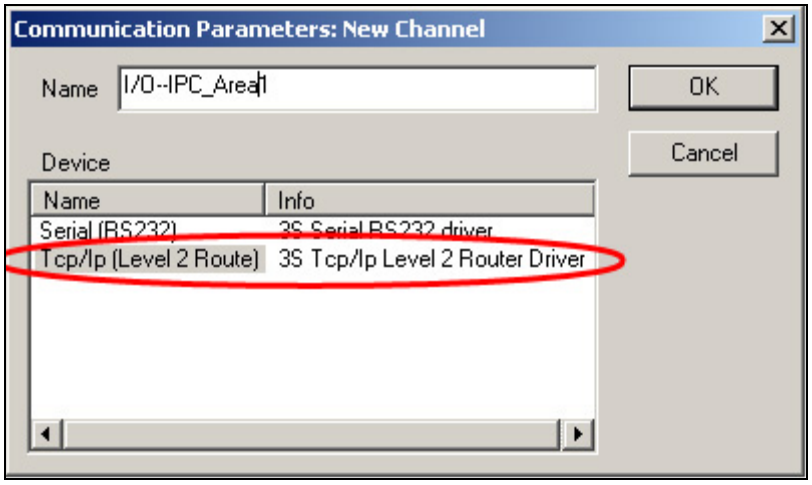


Figure 56: Creating a new communication channel 2

4. Enter **the IP address of your I/O-IPC** in the "Address" field in the "Communication Parameters" window and change the value under “Port” to **1200**.
Then press the enter key on your PC keyboard. To close the window, click in the window on **[OK]**.
To select an already created I/O-IPC, select it in the left window and then click on **[OK]**.

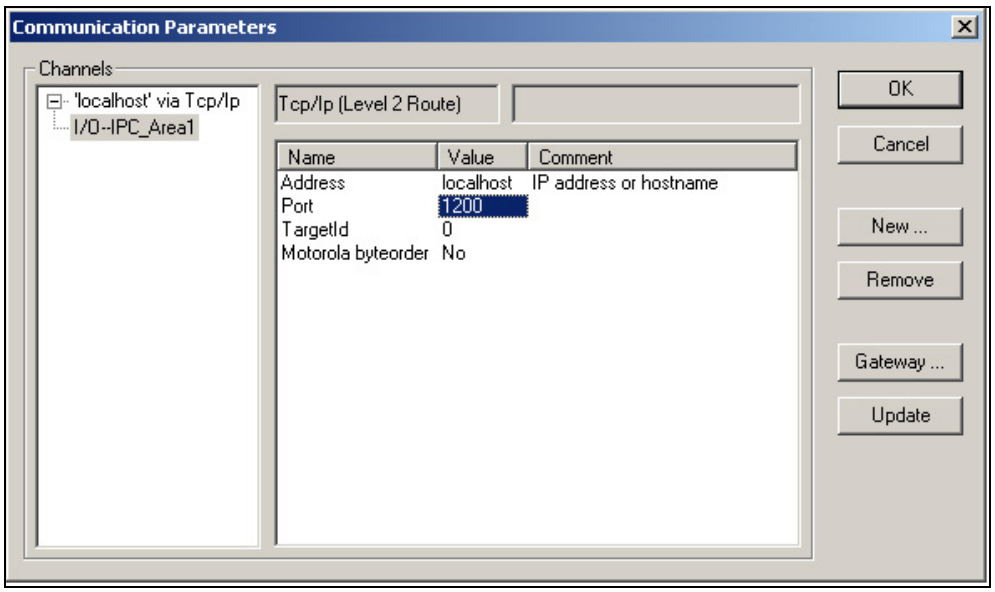


Figure 57: Creating a new communication channel

5. Transfer the PLC program by clicking on **Online** in the menu bar and select **Login**.

6. Remember that the I/O-IPC's Run/Stop switch must be in the "Run" position.

7. Start the PLC program by clicking on **Online** in the menu bar and selecting **Run**.

11.6.6 Loading and Executing the PLC Program in Control (RS 232)

Prerequisite:
The simulation mode is disabled (**Online > Simulation**).
You have connected the PC to the serial interface of the I/O-IPC via a null modem cable. See section "Access via RS-232 Interface and Terminal Program".

1. Select the WBM or the IPC configuration tool for the RS-232 Interface CODESYS. See section "Administration".

2. Click in the menu bar on **Online** and select **Communication Parameters ...** The "Communication Parameters" window opens.

3. To create a communication, click on **[New…]** in the "Communication Parameters" window. The window for creating a new channel opens.

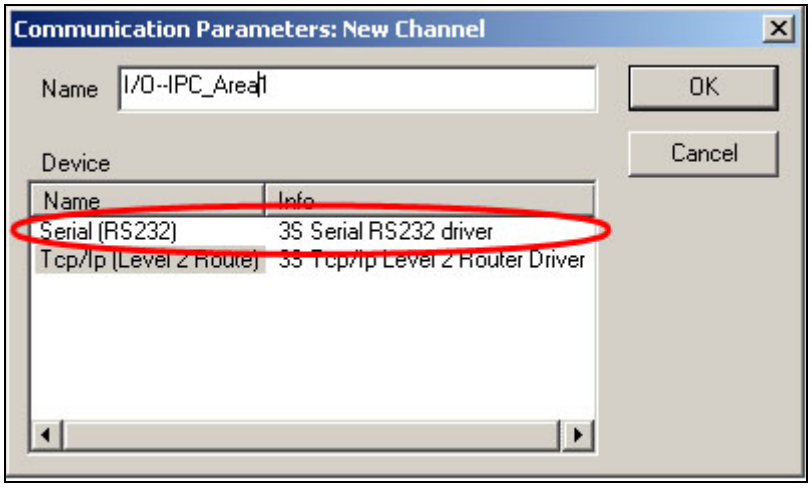


Figure 58: Creating a new communication channel (RS 232) 1

4. Enter any name for your I/O-IPC in the "Name" field and click "Serial (RS232)". Then click **[OK]**.

5. Click in the "Communication parameters" window on **[Gateway]** and select the parameters for the communication "local". To close the window, click **[OK]**.

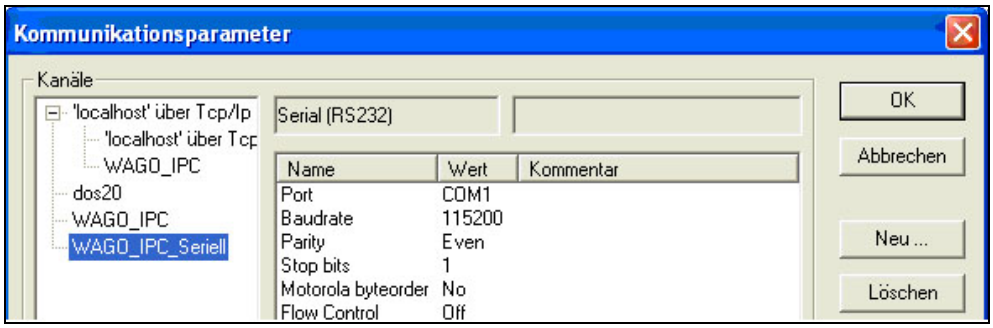


Figure 59: Creating a new communication channel (RS 232) 2

6. Enter the following communication parameters of the RS-232 interface:

   - Baudrate: 115200 bit/sec
   - Parity: Even
   - Stop Bits: 1
   - Motorolla Byteorder: No
   - Flow Control: Off

7. Transfer the PLC programs by clicking on **Online** in the menu bar and select **Login**.

8. Remember that the I/O-IPCs Run/Stop switch must be in the "Run" position.

9. Start the PLC program by clicking on **Online** in the menu bar and selecting **Run**.

11.6.7 Creating a Boot Project

So that the PLC program automatically restarts after restarting the I/O-IPC, create a boot project. To do this, select **Online > Create boot project** in the menu bar. Make sure that you are still logged in to CODESYS.

Note

Boot project automatically load
In addition, you can load the boot project automatically when starting the I/O-IPC. Click on the "Resources" tab and open the "Target settings". Select the "General" tab and then "Load boot project automatically".

If a boot project (DEFAULT.PRG and DEFAULT.CHK) is present under */home/codesys* and the "Run/Stop" switch of the I/O-IPC is set at "Run", the I/O-IPC automatically starts with the processing of the PLC program. If the switch is set at "Stop", the PLC program is not started.

If a PLC program is running in the I/O-IPC, a PLC task starts with the reading of the fieldbus data (only for I/O-IPC with fieldbus connections), the integrated input and output data and the internal data buses. The output data changed in the PLC program is updated after the PLC task is processed. A change in operating mode ("Stop/Run") is only carried out at the end of a PLC task. The cycle time includes the time from the start of the PLC program to the next start. If a larger loop is programmed within a PLC program, the PLC task time is prolonged accordingly. The inputs and outputs are updated during processing. These updates only take place at the end of a PLC task.

11.7 Creating a Task Configuration

With task configuration, you set the time response and priority of individual program function blocks.

Note

Watchdog
In an application program **without** task configuration, there is no watchdog that monitors the cycle time of the application program (PLC_PRG).

Create a task configuration in the following manner:

1. To open task configuration, double click on "Task configuration" in the "Resources" column.

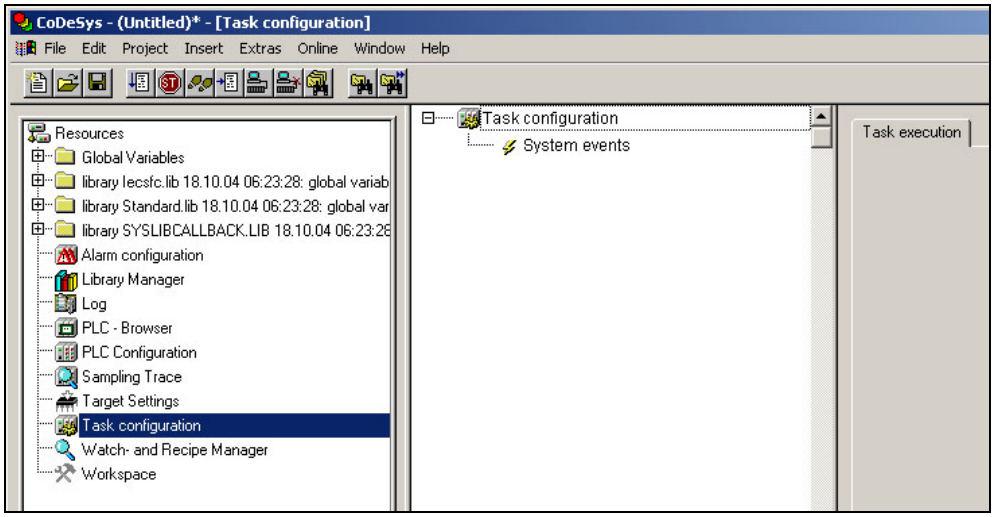


Figure 60: Task configuration

2. To create a task configuration, right click on "Task configuration" and select "Append Task" in the context menu.

3. To assign a new name to the task (e.g. PLC_Prog), click on "NewTask". Then select the type of task. In this example, this is the "cyclic" type.

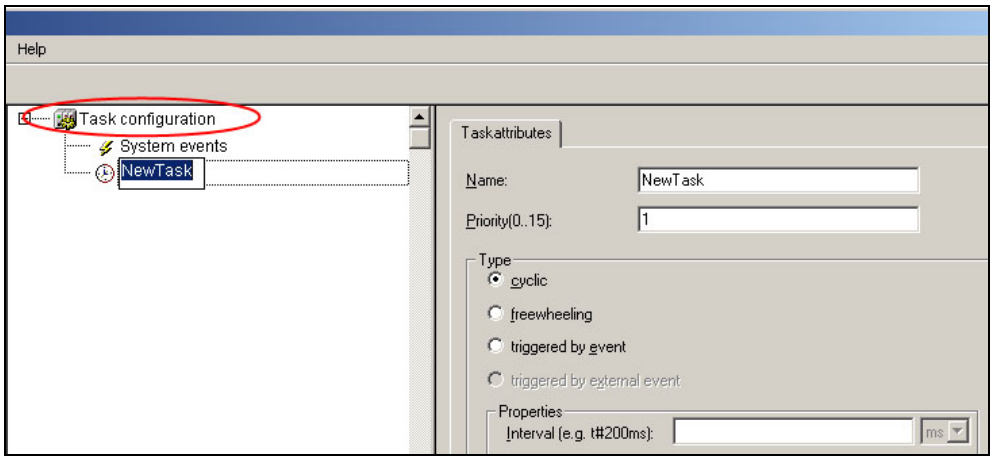


Figure 61: Changing the task name

4. Add the previously created program function block PLC_PRG (see section "Editing a Program Function Block") by right clicking on the "hour" symbol and selecting "Append Program Call" in the context menu. Then click on **[…]** button and **[OK]**.

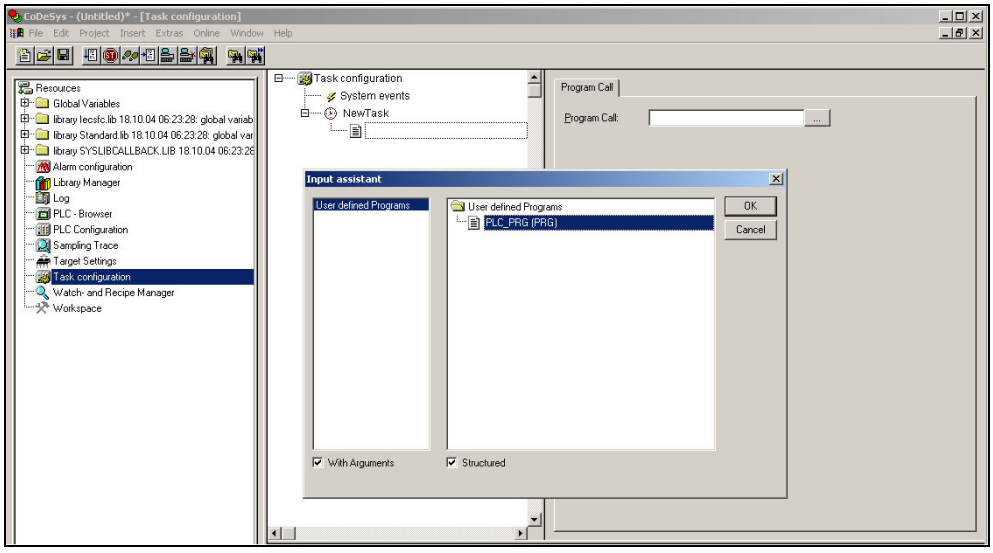


Figure 62: Call to add program function block

5. Compile the example program by selecting **Project > Rebuild all** in the context menu.

11.7.1 Cyclical Task Priorities

You can assign a priority for each task in order to establish the task processing sequence.

All tasks that access the process image of the I/O module are synchronized with it. This means that the tasks with access to the process image of the internal data bus wait until at least one correctly completed internal data bus cycle has been executed.

If there is an error on the internal data bus (e.g. defective I/O module), the tasks that access the process image of the internal data bus are no longer executed. These tasks can only be processed when there is new input data available to them.

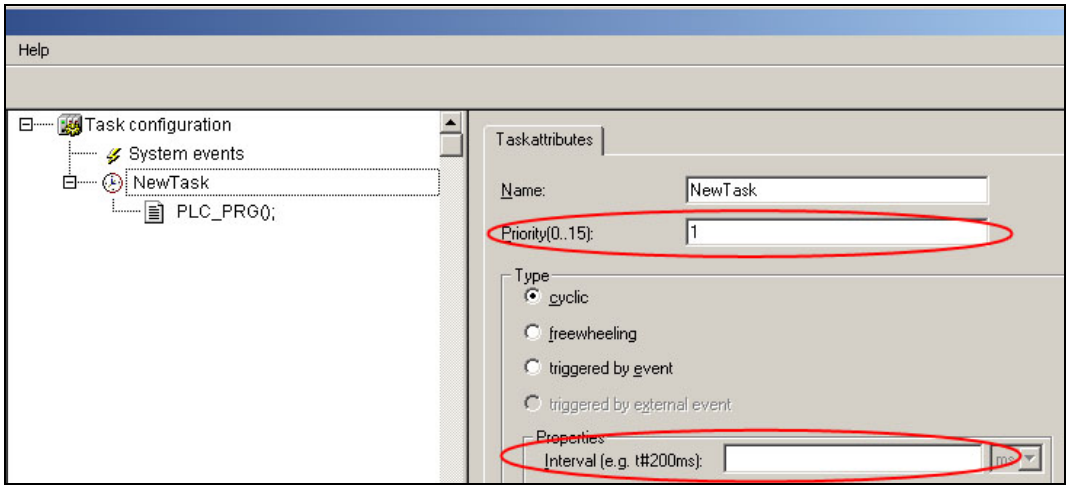


Figure 63: Changing the task name

Note

Priorities of the task processing
The priorities below do not specify the sequence of the task processing. The tasks are started in any sequence.

Priority 0 – 5:
Important arithmetic operations and highly synchronized accesses to I/O module process images should be carried out as tasks with the highest priorities 0-5. These tasks are fully processed according to priority and correspond to LinuxRT priorities -68 through -63.

Priority 6 – 10:
Real-time accesses such as accesses to the ETHERNET, file system, fieldbus (only for I/O-IPC with fieldbus connections) and RS-232 interface should be carried out as tasks with the average priorities 6-10. The tasks are fully processed according to priority and correspond to LinuxRT priorities -50 through -46.

Priority 11 – 15:
Applications such as long-lasting arithmetic operations and non-real-time relevant accesses to the internal data bus, the ETHERNET, file system, fieldbus (only for I/O-IPC with fieldbus connections) and RS-232 interface should be carried out as tasks with the lowest priorities 11-15. The programs within a priority have no difference in priority since each task is assigned the same computing time.

Note

Freewheeling Tasks
If you do not undertake any task configuration, the program PLC_PRG is carried out with the lowest priority at an interval of 10 ms. The runtime of "freewheeling tasks" is not monitored by a CODESYS watchdog.

11.7.2 Freewheeling Tasks

For the use of freewheeling tasks, the input field "Priority (0 ... 15)" in the figure below has no function since it has the lowest priority in the operating system.

For the use of several freewheeling tasks, the operating system takes over their management and assigns each one the same computing time since freewheeling tasks are not distinguished by their priority.

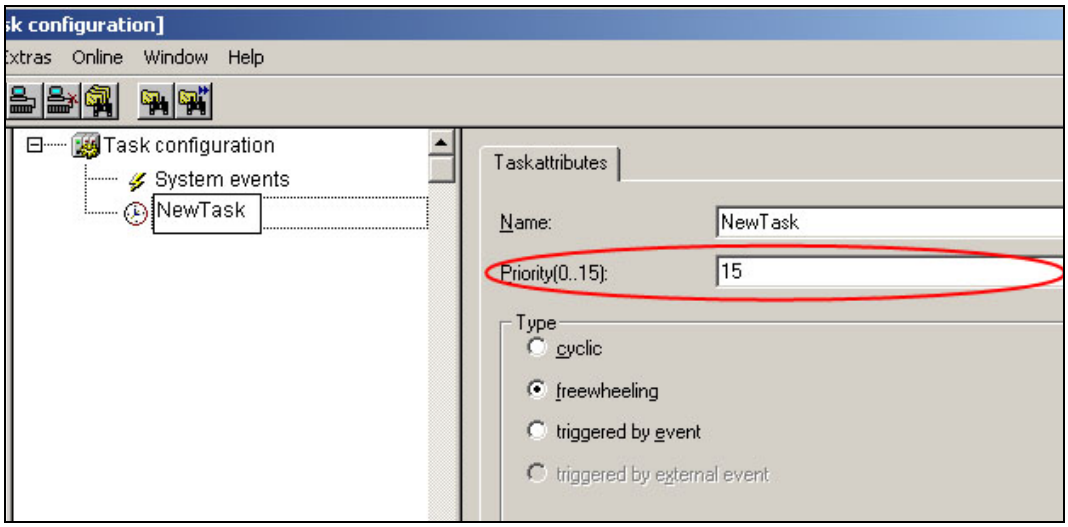


Figure 64: Freewheeling Tasks

11.8 System events

Event tasks can be used in the CODESYS task configuration in addition to cyclical tasks. Event tasks call up certain events in the device.

You can activate events in the following dialog and input a called program:

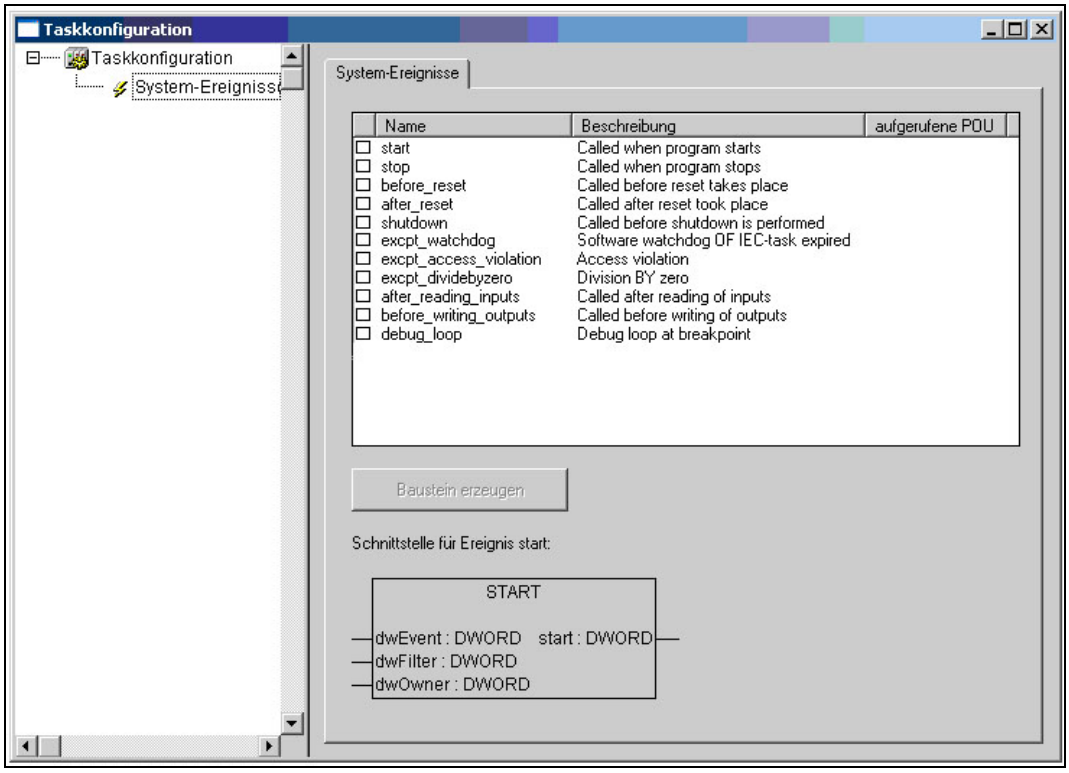


Figure 65: System events

The following events can be activated:

Table 60: Events

Name	Description
start	The event is called directly after the user program starts.
stop	The event is called directly after the user program stops.
before_reset	The event is called directly before the user program is reset.
after_reset	The event is called directly after the user program is reset.
shutdown	The event is called directly before the user program is shutdown.
excpt_watchdog	The event is called if a task watchdog is recognized.
excpt_access_violation	The event is called if a memory access error to an invalid memory area is recognized (incorrect pointer, invalid array index, invalid data descriptor).
excpt_dividebyzero	The event is called if a division by zero is recognized.
after_reading_inputs	The event is triggered independent of the user program after reading all of the inputs.
before_writing_outputs	The event is triggered before writing all of the outputs independent of the user program.
debug_loop	This event is triggered at every task call, if a breakpoint was reached in this task and the processing of this task is therefore blocked

11.9 I/O Module Synchronization

The I/O module cycle and the CODESYS task cycle are optimally automatically synchronized: This depends on the number of I/O modules connected and the fastest CODESYS task cycle set in the I/O-IPC. The synchronization cases described below can therefore take place.

In this section, CODESYS task means only tasks within CODESYS that contain an access to the I/O module. Tasks that do not access the I/O module are not synchronized in the same way as the following. Compare the section "Creating a Task Configuration".

11.9.1 Case 1: The CODESYS task interval is set as less than the I/O module cycle

The implementation of the CODESYS task is synchronized with the cycle time of the I/O module.

The CODESYS task is processed parallel to the I/O module. The CODESYS task interval is lengthened to the I/O module cycle time. This is necessary so that each CODESYS task is started with new input data from the I/O module and the output values are also set at the module after each CODESYS task.

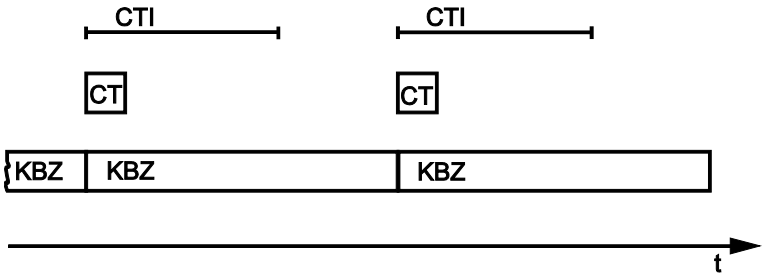


Figure 66: I/O module synchronization 01

CTI: CODESYS Task Interval
CT: CODESYS Task that accesses the I/O module of the internal data bus
KBZ: I/O Module Cycle

Example:
CODESYS Task Interval (CTI): 100 µs
I/O Module Cycle (KBZ): 350 µs
Result: Adaptation of the CODESYS Task Intervals to the I/O module cycle: 350 µs.

11.9.2 Case 2: The CODESYS task interval is less than double the I/O module cycle

The implementation of the I/O module is synchronized to the CODESYS Task Interval set.

At the end of the CODESYS task, the I/O module cycle starts, which is processed synchronously with the fastest CODESYS task. This ensures that when starting each CODESYS Task, current input data are available from the I/O module and the output values of each CODESYS task are also output to the data bus.

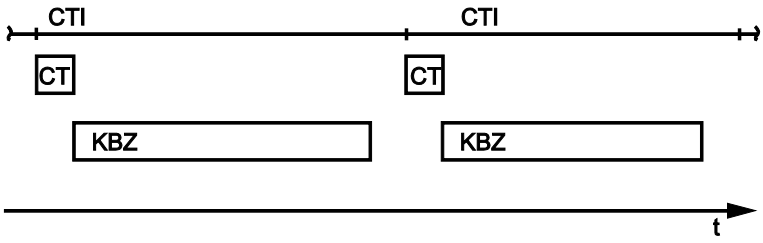


Figure 67: I/O module synchronization 02

CTI: CODESYS Task Interval
CT: CODESYS Task that accesses the I/O module of the internal data bus
KBZ: I/O Module Cycle

Example:
CODESYS-Task-Interval (CTI): 500 µs
I/O Module Cycle (KBZ): 350 µs
Result: Implementation of the I/O module cycle every 500 µs.

11.9.3 Case 3: The CODESYS task interval is greater than double the I/O module cycle

The I/O data from the internal data bus are actualized once prior to the CODESYS task and once after the CODESYS task.

Prior to processing the CODESYS task, the I/O module cycle is implemented, which provides the current input data for the CODESYS task. After implementation of the CODESYS task, an additional I/O module cycle is started, which provides the output data to the internal data bus.

This ensures that at the start of every CODESYS task, current input data are available from the internal data bus and the output data from each CODESYS task are quickly output to the internal data bus. This prevents processing of I/O module cycles that would unnecessarily use a lot of computing time on the CPU.

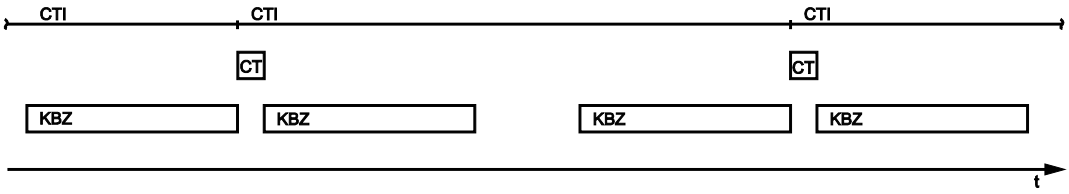


Figure 68: I/O module synchronization 03

CTI: CODESYS Task Interval
CT: CODESYS Task that accesses the I/O module of the internal data bus
KBZ: I/O Module Cycle

Example:
CODESYS-Task-Interval (CTI): 2000 µs
I/O Module Cycle (KBZ): 350 µs
Result: Implementation of the I/O module cycle 350 µs before the CODESYS Task and once immediately after the CODESYS Task.

11.9.4 Case 4: CODESYS Task interval greater than 10 ms

The synchronization takes place as in Case 3; however, the output data busses would be reset to their default state after 150 ms without an I/O module cycle. This prevents the implementation of an I/O module cycle after at least every 10 ms.

The I/O data from the internal data bus are actualized once before the CODESYS task and once after the CODESYS task and, in addition, an additional I/O module cycle is implemented every 10 ms.

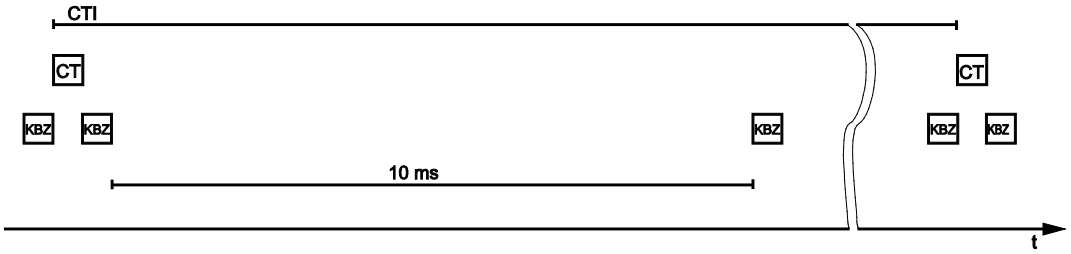


Figure 69: I/O module synchronization 04

CTI: CODESYS Task Interval
CT: CODESYS Task that accesses the I/O module of the internal data bus
KBZ: I/O Module Cycle

Example:
CODESYS-Task-Interval (CTI): 150000 µs
I/O Module Cycle (KBZ): 350 µs
Result: Implementation of the I/O module cycle 350 µs before the CODESYS Task, once immediately after the CODESYS Task, and 10 ms after the last I/O module cycle.

11.10 CODESYS Visualization

The CODESYS Web visualization is based on Java technology. All Java programs require a Java runtime environment (JRE), which must be installed on the host PC along with an Internet browser. An applet (Java program) is created in the file system of a web server and made accessible through an HTML home page.

You create all visualization variants (HMI, Web visualization, and target visualization) with the same CODESYS graphic editor. You define which visualization variants should be executed in the target system setting. For each of these pages a description file in the XML format is generated using the information. You can find these files in the subfolder *"visu"* of the CODESYS installation path. There is also the HTML start page "WebVisu.htm", the Java archive "webVisu.jar" in which the applet (webvisu.class) is stored in compressed form.

After creating a visualization, the following steps are necessary so that you can execute it:

1. Click on the "Resources" tab and open the "Target Settings." Select whether you want the visualization to be displayed as "Web visualization" via an Internet browser and/or as "target visualization" via a monitor connected to the DVI-I interface.

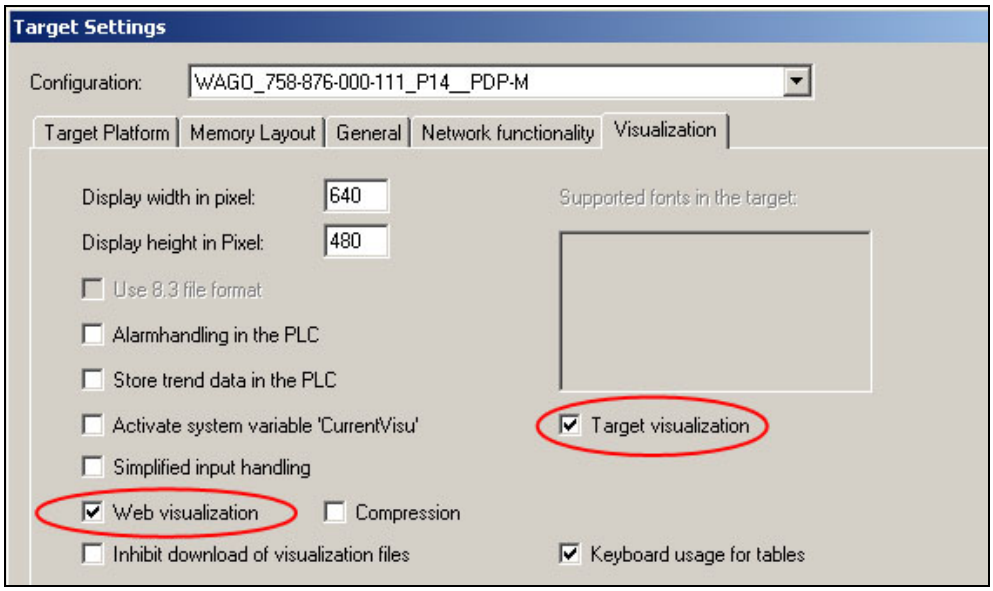


Figure 70: Selection of the visualization variants in the target setting

2. Generate a start page for the visualization. Use the right mouse button to click on the "Visualization" folder on the "Visualization" tab. Select **Add object** ... from the context menu. The "New visualization" dialog box opens.

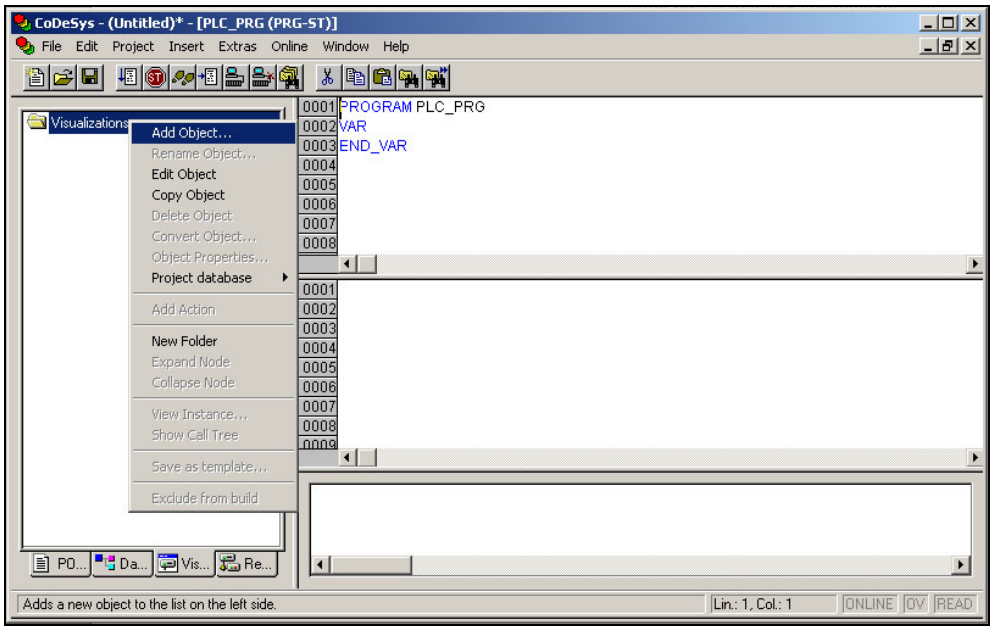


Figure 71: Generating the PLC_VISU home page

3. On the "New visualization" dialog, enter **PLC_VISU** as name for the start visualization.
   If another name should be used, the file "webvisu.htm" must be adapted accordingly.

If you transfer the PLC program into the I/O-IPC (**Online > Log in**) and you have started it (**Online > Start**), the target visualization is displayed automatically. To display the Web visualization, enter the following URL in the address line: *http://<IP>:8080/webvisu.htm*.
Furthermore, you can also display the Web visualization via the WBM. For more information, see Section "'WebVisu' Page".

Note

WAGO start screen with target visualization
At the target visualization of the WAGO start screen only appears, if no COPDESYS project is loaded (from FW09)

Note

Further Informations
You can obtain additional information (FAQ) on CODESYS Web visualization in Section "Frequently Asked Questions About CODESYS Web Visualization" and in the CODESYS 2.3 online help.

11.10.1 Incorporating Fonts

For the CODESYS target visualization, the Truetype fonts Arial and Courier are available upon delivery.

In addition, you can also incorporate any TrueType fonts (*.ttf). Here, it may be necessary to take into account the license conditions of the fonts used. Free fonts are available on the Internet free of charge at

ftp://microwindows.censoft.com/
pub/microwindows/microwindows-fonts-0.90.tar.gz.

To use these fonts, the following steps must be taken:

- The graphics library of the CODESYS target visualization accesses a directory in the file system of the WAGO-I/O-IPC in which you must store these fonts. You must create this directory. The easiest way to do this is via FTP access from the PC.
- After logging in, you will be in the folder */home* of the I/O-IPC file system. Copy the fonts into the directory */home/codesys/fontz/truetype*.
- The extension of the desired font must always be ***.ttf***.
- The font name downloaded onto the I/O-IPC must always be related directly to the name as indicated in the Windows selection box (e.g. Times new Roman.ttf). Upper and lower case letters, but also spaces should be observed here. Otherwise, a substitute fonts will be used automatically.

Table 61: Name convention for fonts (example)

CODESYS selection box	Font name for Target Visualization
Arial	Arial.ttf Arialb.ttf (Bold) Ariali.ttf (Italic) Arialz.ttf (Bold Italic)
Courier	Courier.ttf Courierb.ttf (Bold) Courieri.ttf (Italic) Courierz.ttf (Bold Italic)

11.10.2 Limitations of the CODESYS Visualization

The visualization integrated in CODESYS has the three variants "HMI", "TargetVisu" and "WebVisu", which are all supported by the I/O-IPC. Depending on the process variant, there are technological limitations.

Note

Representation of an ActiveX element in the "Target Visu"
The representation of an ActiveX element in the "Target Visualization" is not possible.

Several options of the complex visualization objects "Alarm" and "Trend" are only provided by the "HMI" version. This applies for example to the sending of emails as a response to an alarm or to the navigation through historical trend data as well as the creation of the data.

In comparison to "HMI", Web visualization on the I/O-IPC is carried out within significantly narrower physical limits. While "HMI" can access almost unlimited resources on a desktop PC, the following limitations must be observed when using Web visualization:

Adaptation to the File System

The overall size of the PLC program, visualization files, bitmaps, log files, configuration files etc. must fit into the file system.

Process Data Memory

Web visualization uses its own protocol for the exchange of process data between applet and control. In doing so, process data is transmitted ASCII coded. The pipe-symbol ("|") is used between two process values as a separating character. Therefore, the space requirement for a process data variable in the process data memory is dependent not only on the data type, but also on the process value itself. Thus, a variable of the "WORD" type occupies between one byte for the values 0 through 9 and five bytes for values from 10000.
The selected format (ASCII + |) only permits a rough estimate of the space requirement for the individual process data in the process data buffer. If the size of the ASCII coded process data is exceeded, Web visualization no longer works according to expectations.

Concurrent connections

The runtime system supports a maximum of 97 (number of CODESYS tasks in the PLC program) simultaneous TCP/IP connections on port 1200.

Computer Performance/Processor Time

The I/O-IPC is based on a real-time operating system. This means that high-priority processes (e.g., PLC program) will interrupt or block lower priority processes. The Web server responsible for Web visualization is among these lower priority processes.

Note

Processor Time
Make sure when configuring tasks that there is sufficient processor time available for all processes.

Network Load

The CPU of the I/O-IPC is responsible for the processing of both the PLC program and network traffic. ETHERNET communication demands that each received telegram is processed, regardless of whether it is intended for the I/O-IPC or not.

A significant reduction of the network load can be achieved by using switches instead of hubs.

There is no measure against broadcast telegrams that can be used on the I/O-IPC, however. These can only be curtailed by the sender, or blocked with configurable switches that have a broadcast limitation. A network monitor such as "wireshark" (www.wireshark.com) provides an overview of the current load in your network.

11.10.3 Eliminating CODESYS Web Visualization Errors

If problems occur when using CODESYS Web visualization, please try to find a solution in the following table first. If you cannot eliminate the problems, please contact WAGO support.

Table 62: Errors and its solution

Error	Solution
Internet Explorer reports the error "APPLET NOT INITIATED"	Close all Internet Explorer windows and restart it. If the error continues to occur, this indicates a missing or damaged file. Check whether the Java archive "webvisu.jar" is completely available in the folder "/PLC" of the I/O-IPC using FTP. The original file can be found in the installation path of CODESYS (usually under *C:\Programme\WAGO Software\CODESYS V2.3\Visu\webvisu.jar)*. If necessary, replace the damaged file using FTP or force the download of all files in CODESYS with **Purge All > Compile All > Log In**.
Web visualization is not displayed	Have you installed the JRE? Check the firewall settings, e.g. if port 8080 is open.
Web visualization "freezes". Web visualization stops after a longer period of time.	The call-up intervals selected in the task configuration are too small. As a result, the I/O-IPC Web server, which is executed with a low priority, receives insufficient or no computing time. If no (explicit) task configuration has been provided, the PLC_PRG is (implicitly) executed as a free running task with Priority 1. This allows the Webserver too little computing time. Always provide a task configuration when using Web visualization. In doing so, the call-up interval should not exceed three times the average execution time. When determining the execution time, ensure that the PLC program is not "steady state."
Web visualization cannot be loaded into the I/O-IPC	It may be that not all files fit into the file system of the I/O-IPC. Do not delete necessary data (e.g., via FTP).
Bitmap is not displayed	If the name of an image file contains umlauts, the Web server cannot interpret these image names.
Java console reports: "Class not found"	The JRE does not find the entry point for the class "webvisu.class" in the Java archive "WebVisu.jar. The Java archive is probably incomplete. Delete "WebVisu.jar" from the Java cache and/or deactivate the cache. In this case, the I/O-IPC requests the archive (applet) again. If the problem continues to exist, upload the project onto the I/O-IPC again.
Web visualization is static, all process values are "0"	The cause is that the process data communication has failed. If web visualization is operated over a proxy server, then a SOCKS proxy is also necessary for process data exchange in addition to the actual HTTP proxy.

11.10.4 Frequently Asked Questions About CODESYS Web Visualization

How can I optimize the applet for special screen resolutions?

In order to optimize the Web visualization for a PDA or a touch panel with a fixed resolution, you should proceed as follows:
In the "Target system settings", enter the pixel width and height in the tab "Visualization". When the visualization is created, the visible area is highlighted in grey. However, the actual pixel width and height of the Web visualization is defined by the attributes "Height" and "Width" of the HTML APPLET tag in the "webvisu.htm" file. Do not forget to also adapt these parameters to the existing resolution.

Which JRE should I use?

It is recommended that you use Java2 standard edition Version 1.5.0 (J2SE1.5.0_06) or higher. This is available free of charge at www.sun.com. Microsoft's MSJVM3810 was also tested. Furthermore, for PDAs there are runtime environments available from other manufacturers (JamaicaVM, CrEme, etc.). To be considered is that for the Web visualization, these solutions can behave differently with respect to their scope of services (e.g. stability) than those mentioned above.

Should the Java Cache be used?

Neither yes nor no. After a standard installation the cache is enabled. If the cache is enabled, the JRE uses it to store applets and Java archives. If the Web visualization is called up a second time, it requires considerably less time to start because the applet (approx. 250 kb) does not need to be reloaded via the network, but is already available in the cache. This is especially interesting when network connections are slow.

Note:
The Java archives may not be transferred into the cache completely due to network failures. In this case, the cache must be cleared manually or be disabled.

Why does the visualization element "TREND" in the Web visualization only work "Online"?

The following settings must be selected for the visualization projects: **Resources** tab **> Target system settings**.
Activate "Web visualization" and "Trend data recording within control." Otherwise, the trend data are saved on the hard drive of the CODESYS development computer. This makes a permanent connection between the I/O-IPC and the CODESYS gateway necessary. An interruption in this connection can lead to unforeseeable behavior of the I/O-IPC.

In the TREND configuration dialog, you can choose between the operating modes "Online" and "History." The I/O-IPC only supports the operating mode "Online" for visualization projects since it is not possible to configure the maximum size (quota) of the trend files (*.trd). An uncontrolled expansion of trend files can lead to unforeseeable behavior of the I/O-IPC.

In most cases, the use of the "HISTOGRAM" visualization element is the better choice, as this gives full control over the time and number of measurements and thus the amount of memory required.

What needs to be observed when the visualization element "ALARM TABLE" is used in the Web visualization?

The status of this component is best described with "Add-On", i.e. an extra that is free of charge and without warranty.

The following settings must be selected for visualization projects: **Resources** tab **> Target system settings**.
Activate "Web visualization" (checkmark) and "Alarm handling within control". Otherwise, the alarm data is processed on the CODESYS development computer. This makes a permanent connection between the I/O-IPC and the CODESYS gateway necessary. An interruption of this connection can lead to unforeseeable behavior of the I/O-IPC.

12 CANopen Master in CODESYS 2.3

12.1 CANopen I/O-IPC PLC Control Configuration

Note

Calling the addresses or symbolic names of the inputs and outputs
Addresses or symbolic names of the inputs and outputs have to be called explicitly, because otherwise the process image is not updated. Alternatively, you can also create an array of max. 240 bytes at the memory addresses IB%4800 or QB%4800. This array has to be called in the PLC program.

Before an application has access to the connected CAN network, you must configure it in CODESYS:

1. To integrate the CANopen-I/O-IPC in the control configuration, right-click on "PLC Configuration" and select "Append CANMaster."

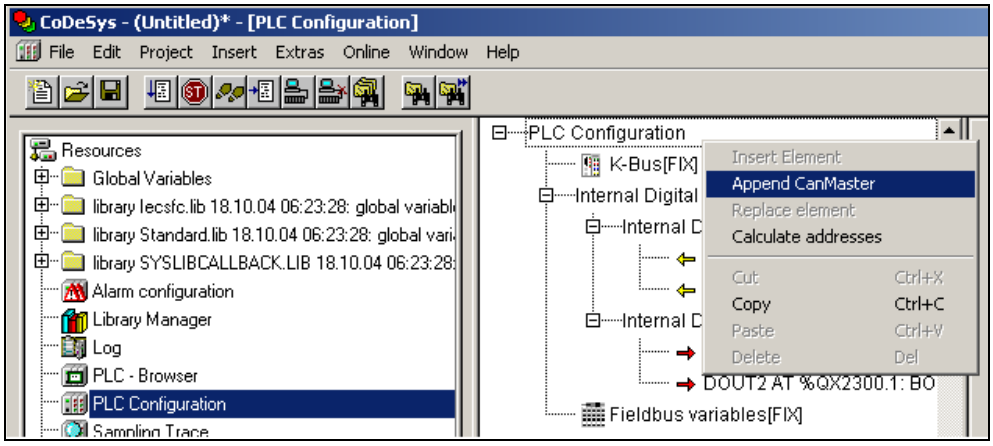


Figure 72: Adding the CANopen master

2. On the "CAN Parameters" tab, activate the "Support DSP301, V4.01 ..." checkbox so that modular slaves and some additional extensions with respect to the standards DSP301 V3.01 and DSP 306 are supported.

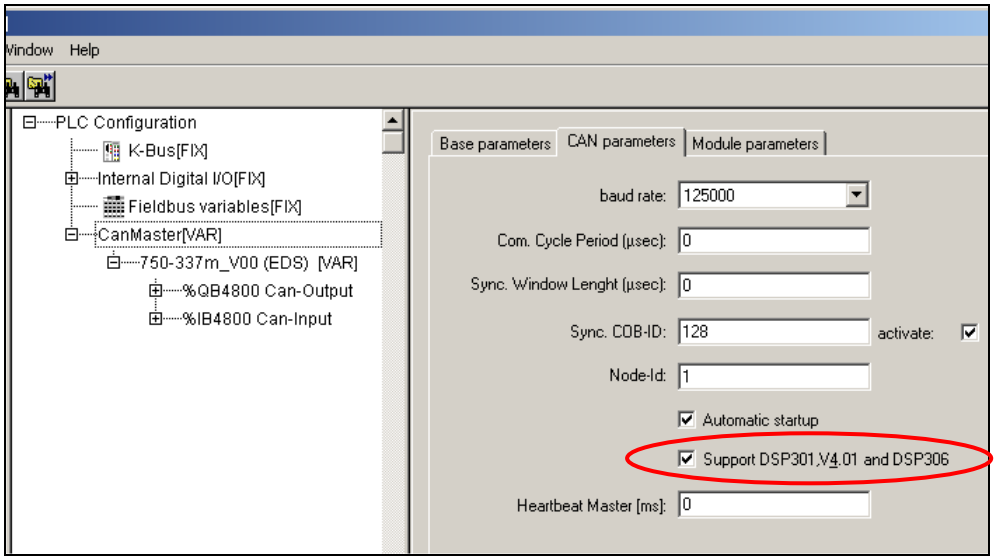


Figure 73: Activating DSP301 and DSP306

3. To select one (or several) CANopen slaves, click the right mouse button on the I/O-IPC (CANMaster[VAR]) and select "Append Subelement." In this example, the 750-337 was selected as the slave.

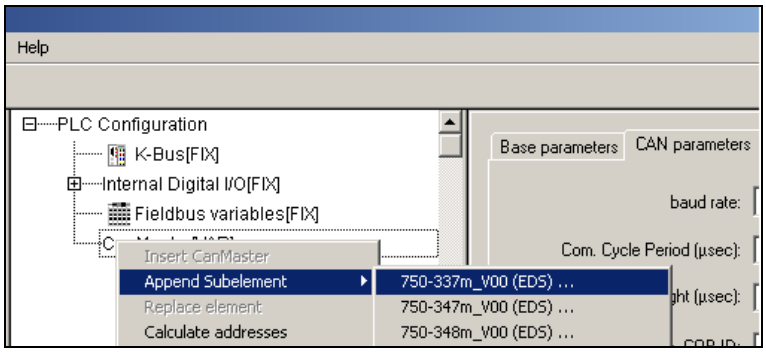


Figure 74: Appending the CANopen slaves

Note

EDS files

The EDS files for current components of the WAGO-I/O-SYSTEM are integrated in the target files for the WAGO-I/O-IPC. The accompanying EDS files must be used when connecting non-WAGO devices. To do this, click on **Extras > Add configuration file…** in the menu bar.

4. Click the "CAN Module Selection" tab. Adopt the topology of the I/O modules connected to the slave (from the end module to the coupler) into the control configuration. To do this, insert the respective I/O modules in the right window ("Selected Modules") using **[Add]**.

5. Using **[Remove],** if necessary you can delete I/O modules added incorrectly from the right window. To do this, the I/O module must be marked.

In this example, a digital input module and a digital output module are connected to the slave.

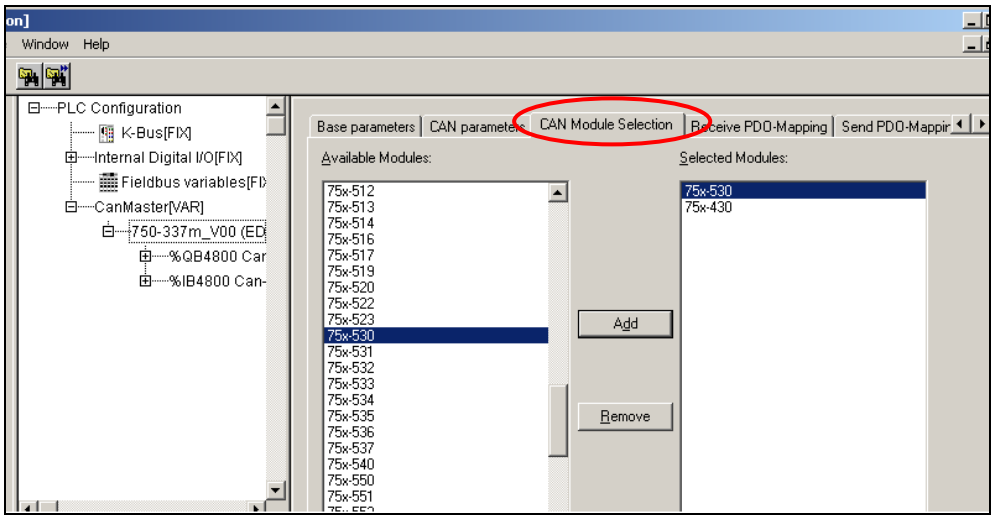


Figure 75: Selecting I/O modules

Note

I/O modules that do not provide any process data

I/O modules that do not provide any process data (e.g. power supply module, end module) are not taken into consideration during the configuration and therefore also do not appear in the selection list of the EDS file.

6. Depending on the I/O modules selected for the CANopen slave, the corresponding CANopen objects appear on the "Receive PDO Mapping" and "Send PDO Mapping" tabs. Using the tabs, you can change the "Default Mapping" described in the EDS file.

7. Click on the "CAN Parameters" tab to adapt the station address of the slaves. In the "Node ID" field, enter the station address that corresponds to that of the slave . If the two station addresses do not agree, no communication connection can be established.

8. Enter the device type in the "Device Type" field. This depends on the type (digital/analog) of I/O modules plugged into the slave. For the device profile DS401, there are 15 different device types that must be adapted to the slave topology in case of changes:

1	0x10191	Digital inputs
2	0x20191	Digital outputs
3	0x30191	Digital inputs and outputs
4	0x40191	Analog inputs
5	0x50191	Digital inputs and analog inputs
6	0x60191	Digital outputs and analog inputs
7	0x70191	Digital inputs and outputs and analog inputs
8	0x80191	Analog outputs
9	0x90191	Digital inputs and analog outputs
10	0xA0191	Digital outputs and analog outputs
11	0xB0191	Digital inputs and outputs and analog outputs
12	0xC0191	Analog inputs and outputs
13	0xD0191	Digital inputs and analog inputs and outputs
14	0xE0191	Digital outputs and analog inputs and outputs
15	0xF0191	Digital inputs and outputs and analog inputs and outputs

Information

Additional information

For additional information about the device type (object 0x1000), please see the manuals for the CANopen fieldbus coupler.

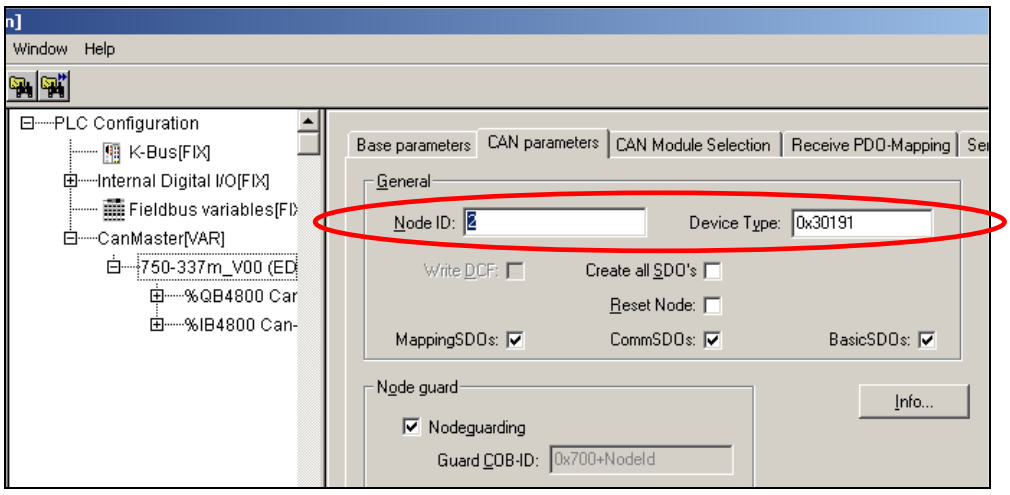


Figure 76: Setting the Node ID

12.2 PLC Configuration Setting Choices

12.2.1 CANopen Master (I/O-IPC)

Via the tab for the I/O-IPC, you define the global settings and monitoring parameters for the CANopen master.

Base Parameters

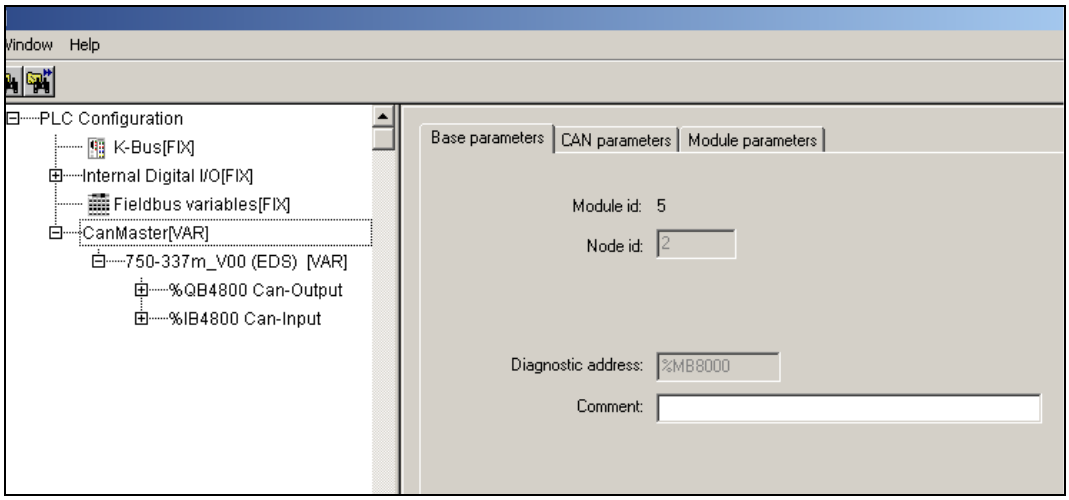


Figure 77: "Base Parameters" tab

The following table explains the parameters listed on the tab:

Table 63: Description of the basic parameters

Basic Parameters	
Module ID	Parameters that use the runtime system CODESYS.
Node ID	
Diagnostic address	
Comment	Input field for a comment.

CAN Parameter

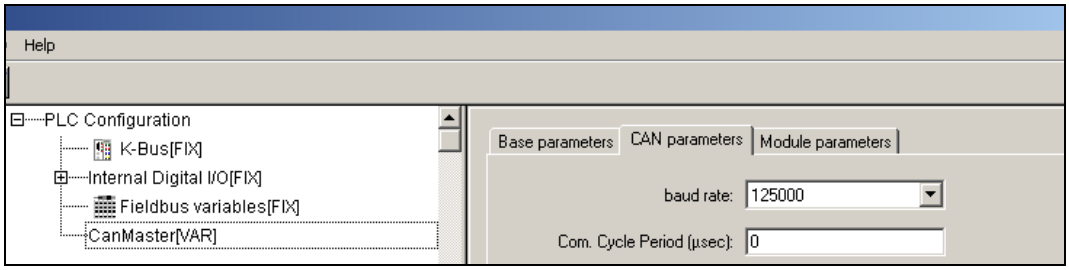


Figure 78: "CAN Parameters" tab (example)

The following table explains the parameters listed on the tab:

Table 64: Description of the CAN parameters

CAN Parameter	
Baud rate	Here you select the baud rate that should apply for the transfer on the CAN bus (default setting: 125000 baud).
Com. Cycle Period (µsec)	Here you enter the time interval (in µsec) at which the synchronization message is sent by the I/O-IPC. Smallest time interval: 1000 µsec
Sync. Window Length (µsec)	Not currently implemented.
Sync. COB-ID	Here you activate or deactivate the sending of synchronization messages of the I/O-IPC. This function is activated for the slave only if you are using the transmission type "acyclic-synchronous." Default setting: COB-ID 128 (0x80). See also table "Description of the parameters".
Node-ID	Station address (node-ID) of the I/O-IPC on the CAN bus.
Automatic startup	If you activate this checkbox, the I/O-IPC establishes the communication connection with the CANopen slaves (via broadcast message "Start remoteNode"). If you deactivate this checkbox, the broadcast message is not sent. The CANopen slaves can be started individually via the module parameter "EnableNMTStartNode" (see "Module parameters").
Support DSP301 ...	If you activate this checkbox, modular CAN slaves as well as additional extensions with respect to the standards DSP301 V3.01 and DSP 306 will be supported by the control configuration.
Heartbeat Master (ms)	If the "Heartbeat Generation" option is activated, the CAN device transmits heartbeats in ms intervals specified in "Heartbeat Producer Time". Heartbeat consumption is not currently implemented.

12.2.2 CANopen Slaves

Using the tab described below, you define the behavior of CANopen slaves.

Base Parameters

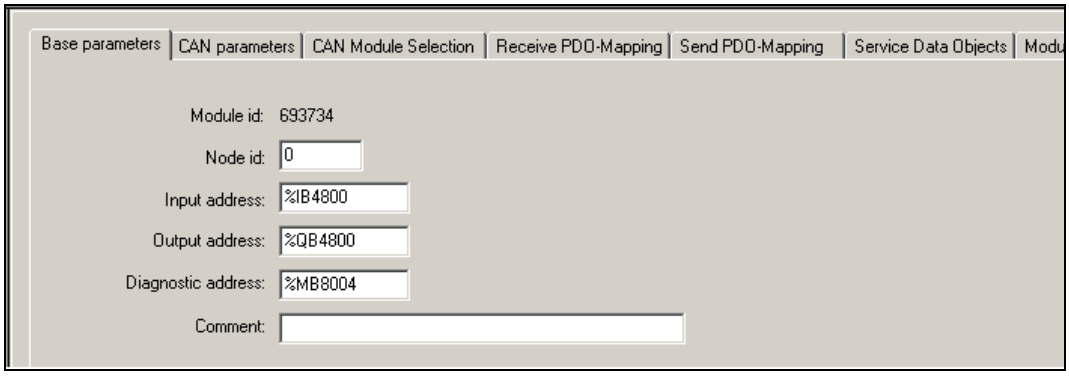


Figure 79: "Base Parameters" tab

The following table explains the parameters listed on the tab that are automatically assigned by the target system settings (see section "Designing a Project and Selecting the Target System"):

Table 65: Description of the basic parameters

Base Parameters	
Module ID:	Recognition of the slave.
Node ID:	Node number of the slave used in the CODESYS runtime environment.
Input address:	Starting address for the input data: The address space always begins at %IB 4800 and is assigned automatically. You can also increase the starting address.
Output address:	Starting address for the output data: The address space always begins at %QB 4800 and is assigned automatically. You can also increase the starting address.
Diagnostic address:	Starting address of the diagnostic data: The address space always begins with %MB 8004 and is automatically assigned and may not be changed.
Comment:	Input field for a comment.

CAN Parameters

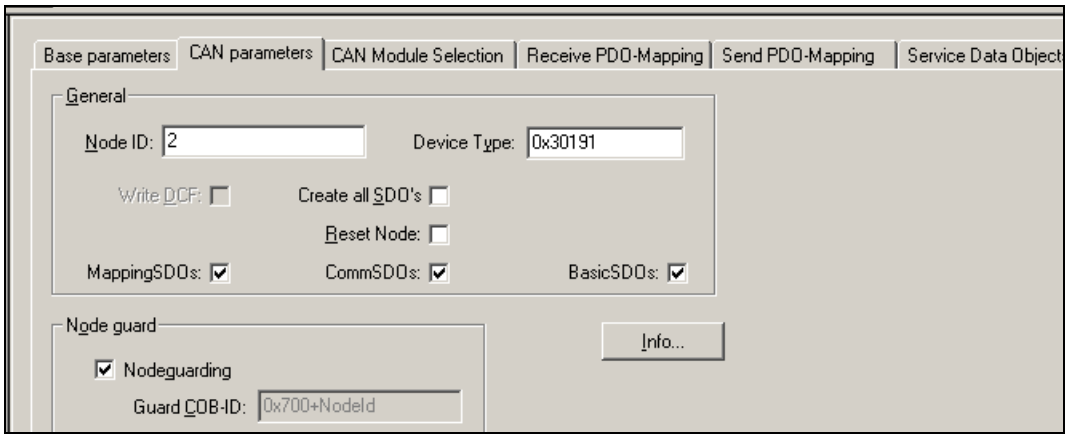


Figure 80: "CAN Parameters" tab 1

The following table explains the parameters listed on the tab:

Table 66: Description of the CAN parameters

CAN parameter	
General	
Node ID:	The node ID (1-126) is the station address under which the I/O-IPC communicates with the slave on the CAN network.
Device Type:	Enter the device type here. This depends on the type (digital/analog) of the I/O modules plugged into the slave. For the device profile DS401, there are 15 different device type values that must be adapted to the slave topology in case of changes (object 0x1000). See section "CANopen I/O-IPC PLC Control Configuration".
Write DCF	Not currently implemented.
Create all SDO´s:	With an activated checkbox, SDOs are generated for all objects in the EDS file. In addition, the corresponding options must be activated. If the node-guarding objects are to be written, for example, the checkbox for the "Node-Guarding" option also has to be activated. If the checkbox is deactivated, SDOs are only generated for the objects for which the default values deviate from the EDS file.
Reset Node	If you activate this option, the slave is reset by the I/O-IPC (receives a "reset node") before the configuration is sent to the slave. This function is not currently implemented.
MappingSDOs:	Here you activate or deactivate each of the three SDO ranges of the slave configuration. MappingSDOs: Objects 0x1600 – 1620 Objects 1A00 – 1A20 CommSDOs: Objects 0x1400 – 1420 Objects 1800 – 1820 BasicsSDOs: Objects 0x100C – 1017
CommSDOs:	
BasicSDOs:	

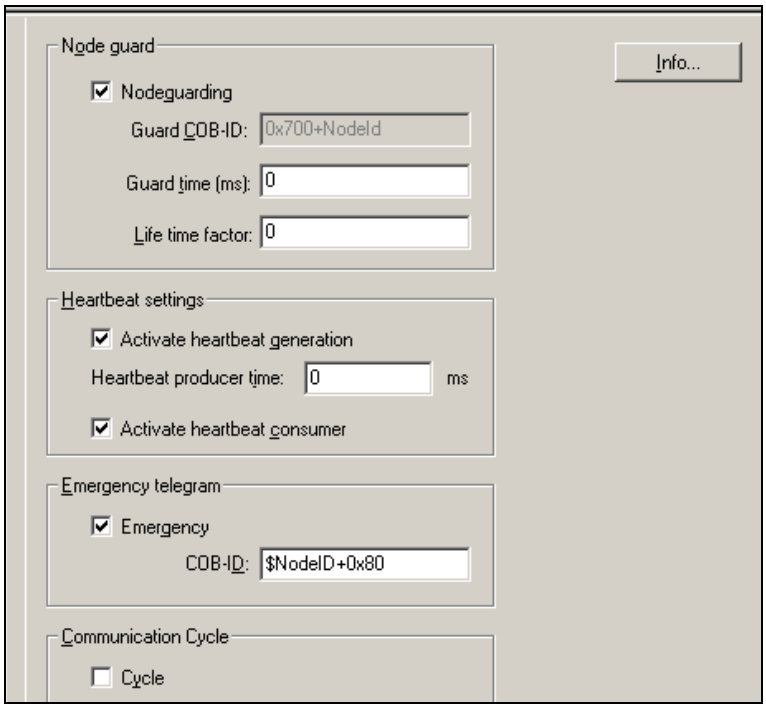


Figure 81: "CAN Parameter" tab 2

Table 67: Description of the parameters

CAN Parameter	
Nodeguard	
Nodeguarding: Guard Time (ms): Life Time Factor:	With activated nodeguarding, the slave monitors the I/O-IPC for a possible interruption of the fieldbus communication. Under "Guard Time" you specify the interval in which the I/O-IPC expects the "Confirmation" of the slave. In the "Life Time Factor" field (>= 3) you set the multiplication value for the "Guard Time." If the time arising from "Guard Time" x "Life Time Factor" ("Node Life Time") has passed, the WAGO slave is brought into the predefined state.
Heartbeat Setting	
If the "Enable Heartbeat Producer" option is enabled, the module broadcasts heartbeats in the ms intervals specified by the Heartbeat Producer Time. If the "Enable Heartbeat Consumption" option is enabled, the module listens for heartbeats coming from the master.	
Emergency-Telegram	
Emergency	If you activate this checkbox, the slave sends error and status messages that are stored as emergency messages to the diagnostic address in the flag area. You read out these error and status messages using the "BusDiag.lib." If you deactivate this checkbox, SDO 0x1014 is not transmitted to the slave. The default setting of the slave is thus still valid.
COB-ID	Default: Node ID + 0x80
Info ...	
„FileInfo“ und „DeviceInfo“ der EDS-Datei	
Communication Cycle	
Not currently implemented.	

Receiving/Sending PDO Mapping

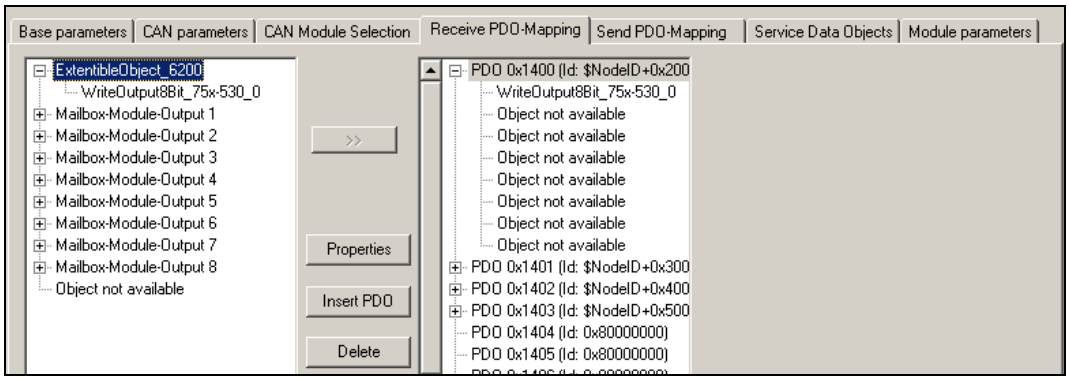


Figure 82: "Receive PDO-Mapping" tab

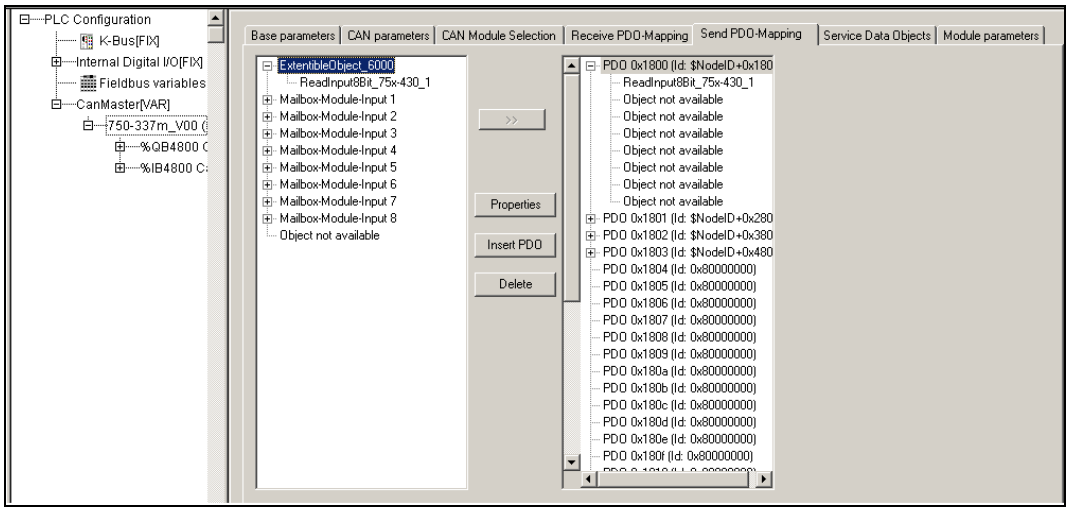


Figure 83: "Receive PDO-Mapping" tab

The following table explains the parameters listed on the tab:

Table 68: Receiving and sending description for the PDO mapping

Receiving PDO Mapping	
Insert PDO	Depending on the I/O modules selected for the CANopen slave, the corresponding CANopen objects appear on the "Receive PDO Mapping" (I/O-IPC → Slave) and "Send PDO Mapping" (Slave → I/O-IPC) tabs. Using these tabs, you can change the "Default Mapping" described in the EDS file. Using the [**Insert PDO**] button, you adapt the PDO to the I/O module topology. the PDO Properties window opens, via which you can assign the PDO particular properties. For additional information, see "Properties." In order to assign one of the 32 PDOs an object from the left window, mark both the corresponding object and the corresponding PDO and then click [>>]. Then the object will be added below the PDOs in the right window. The first 64 digital and the first 12 analog inputs and outputs are then assigned automatically to the PDOs 1 -- 4.
Delete	With the [**Delete**] button, you delete the entry that is currently marked in the right window from the configuration.
Properties	A dialog box with information about the PDO properties opens.

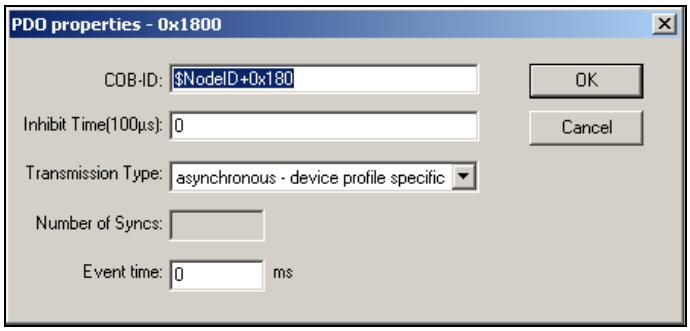


Figure 84: PDO properties window

The following table explains the parameters listed on the PDO Properties window:

Table 69: Description of the PDO Properties window

Receiving PDO Mapping	
COB-ID:	CAN-Identifier
Inhibit Time (100 µs):	Here you set the time span of a PDO for the reduction of the communication incidence; this is the time that must pass before it can be sent again.
Transmission Type:	Here you select the transmission mode for the PDO: **acyclic-synchronous:** (transmission type 0) The PDO is transmitted synchronously, but not periodically. For receive PDOs, the transmission types 0 - 240 are handled the same way. **cyclic-synchronous:** (transmission type 1 - 240) The PDO is transmitted synchronously, whereby "Number of Syncs" specifies the number of synchronization messages that lie between two transmissions of the PDO. For receive PDOs, the transmission types 0 - 240 are handled the same way (not implemented). **synchronous – RTR only:** (transmission type 252) The PDO is updated after a synchronization message, but not sent. It is only transmitted with an explicit inquiry "Remote Transmission Request" (not implemented). **asynchronous – RTR only:** (transmission type 253) The PDO is only updated and transmitted with an explicit inquiry "Remote Transmission Request" (not implemented). **asynchronous-manufacturer-specific:** (transmission type 254) The PDO is only transmitted after particular events. asynchronous-device profile-specific: (transmission type 255) The PDO is only transmitted after specific events.
Number of Syncs	Depending on the "Transmission Type," this field can be edited to enter the number of synchronization messages 1 - 240
Event time	Event time: Depending on the "transmission type," here you enter the time span (in ms) that should elapse between two transmissions of the PDO.

Service Data Object

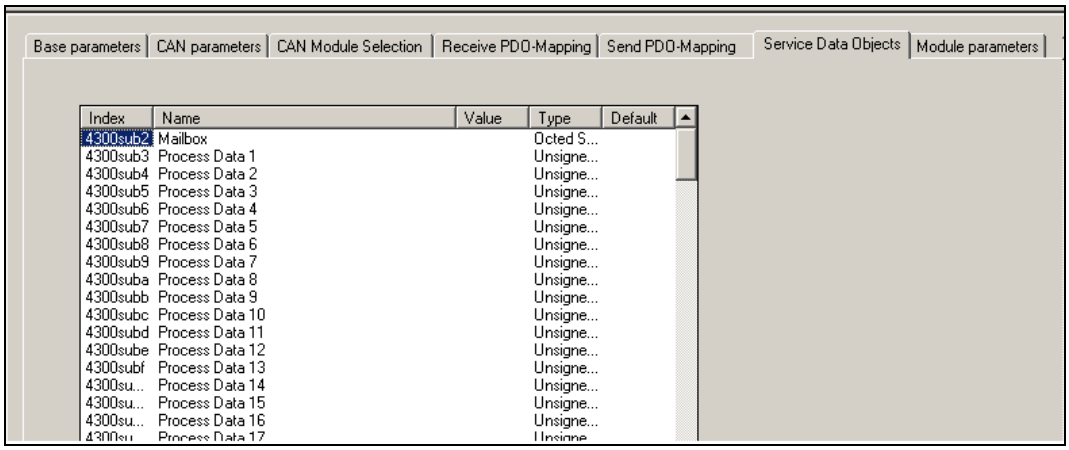


Figure 85: "Service Data Objects" tab

All objects of the EDS file are listed here, which are in the range from index 0x2000 to 0x9FFF and can be described.

For each object, the index, name, value, type, and default are specified.

The value of the objects can be changed. To do this, mark the field in question in the "Value" column and overwrite the value with your input, then press the **[Enter]** key. On initialization of the CAN bus, the set values will be transmitted to the slaves in the form of SDOs.

Module Parameters

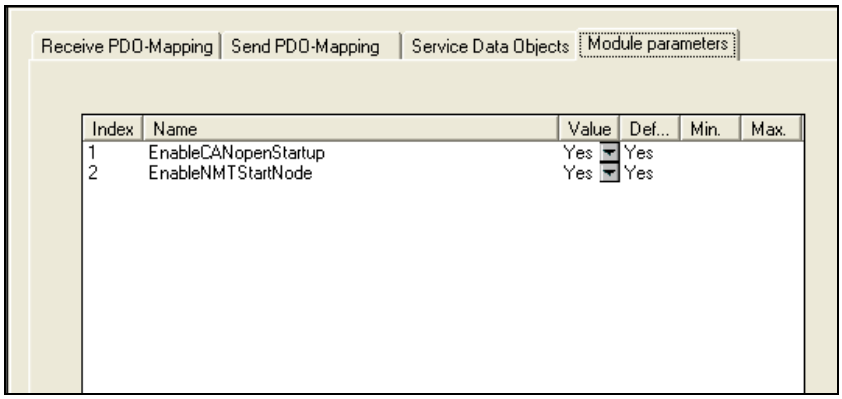


Figure 86: "Module Parameter" tab

Table 70: Description of the module parameters (slave)

Receive PDO Mapping	
EnableCANopenStartup	**Yes** (default): During the boot-up phase of the CANopen network, all basic SDO frames are sent to the selected CANopen slave. **No**: With this setting, no SDO frames are sent to the CAN-Layer-2 device.
EnableNMTStartNode	**Yes** (default): During the boot-up phase of the CANopen network, the NMT command "Start remote node" is sent to the selected CANopen slave (communication connection is established). **No**: With this setting, no start command is transmitted. The CANopen slave can be started at any time using the "Start remote node" command. **Note**: To do this, deactivate the parameter "Start automatically."

12.3 Access to the CANopen Process Data

The added I/O modules appear in the control configuration under the "WAGO 750-337" slave along with their fixed addresses (see figure). You can declare your own variables for the inputs and outputs of these I/O modules.

To declare variables, proceed as follows:

1. Click on the "Resources" tab and open the control configuration.

2. Double-click on the "AT" entry (next to the arrow) and enter a variable name. In this example, the names "CAN_Input_Byte" and "CAN_Output_Byte" are assigned.

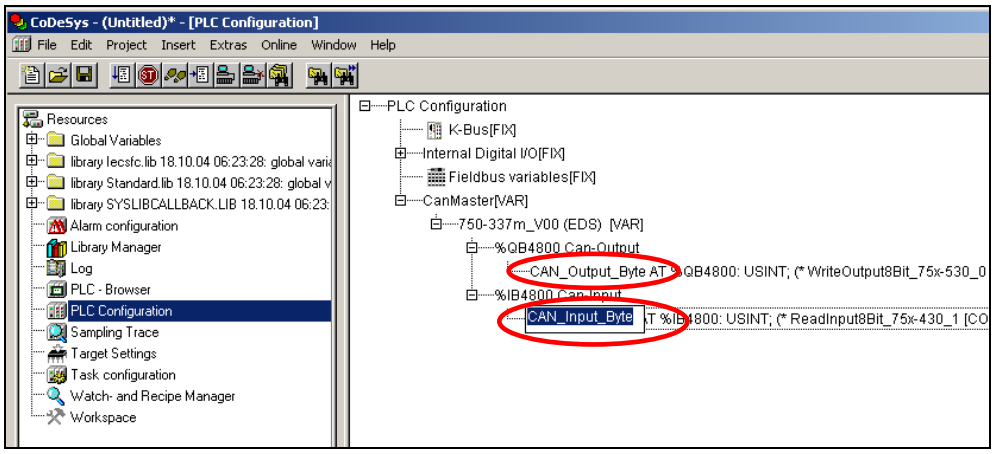


Figure 87: Control configuration: I/O modules with their associated addresses

The following example in the "Structured Text" (ST) programming language is intended to clarify access to the variables. To do this, an input is assigned to an output:

1. Change to the "POUs" tab and double-click on the program function block PLC_PRG.

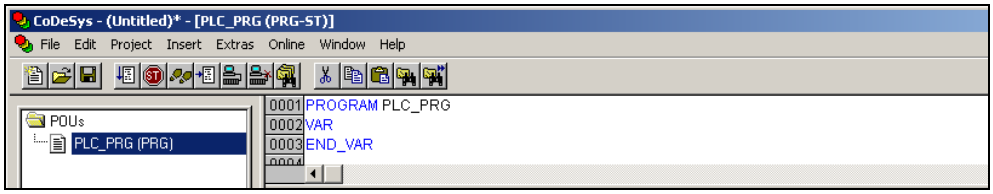


Figure 88: PLC_PRG

2. Press the **[F2]** key to open Input assistant or right-click and select **Input assistant** from the context menu.

3. Select the previously-declared variable "CAN_Output_Byte" under "Global Variables" and click on **[OK]** to add it.

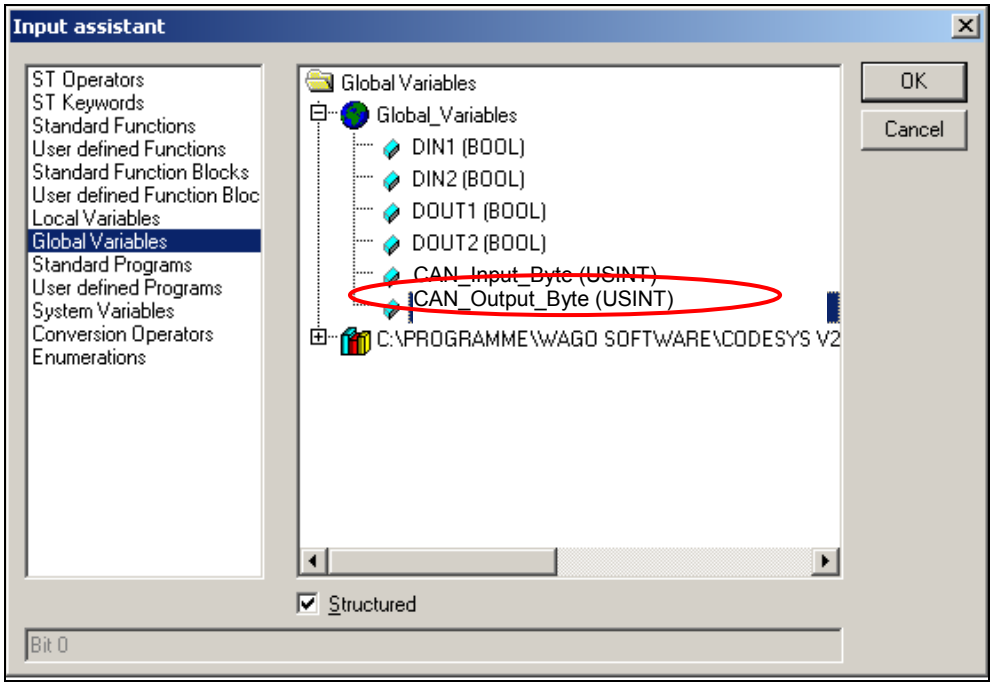


Figure 89: Input assistant for selecting variables

4. Enter the bit access **0** behind the variable name "CAN_Output_Byte."

5. Enter the assignment**:** = behind the bit access **0**.

6. Repeat step 3 for the variable "CAN_Input_Byte."

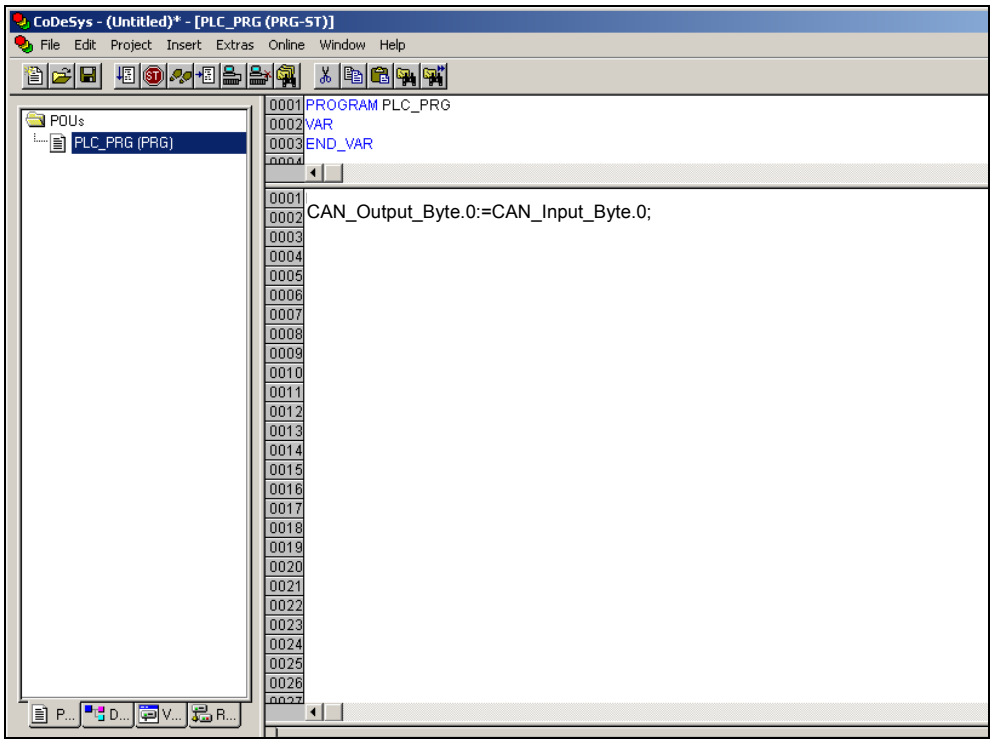


Figure 90: Example of an assignment of the previously-created variables

7. To compile, click on **Project** in the menu bar and select **Rebuild all**.

12.4 Loading the Program in the I/O-IPC

Transfer the program to the I/O-IPC by clicking **Online** in the menu bar and selecting **Log in**. The CAN bus starts automatically during the downloading of the PLC configuration into the I/O-IPC. For more information, see section "Loading and Executing the PLC Program in Control (ETHERNET)".

12.5 Diagnostics of the fieldbus coupler

This section requires a good knowledge of the CODESYS programming tool. It only describes the procedure to create diagnostics using the fieldbus master as an example.

Configured slaves (e.g., a fieldbus coupler or a fieldbus controller) are a prerequisite for diagnostics in fieldbus networks.

12.5.1 DiagGetBusState() and DiagGetState()

To evaluate the diagnostics, you will need the following function blocks from the library BusDiag.lib:

- DiagGetBusState() for the bus diagnostics
  This function block provides you with general information on each connected slave (e.g. number of slaves).

- DiagGetState() for the participant diagnostics
  This function block provides you with detailed information on each slave (e.g. information on diagnostics).

12.5.2 Creation of Diagnostic Functions in CODESYS 2.3

To conduct bus diagnostics and participant diagnostics of the slaves, the library BusDiag.lib must be contained in CODESYS. This library contains the necessary function blocks DiagGetBusState() for bus diagnostics and DiagGetState() for participant diagnostics.

Integrate the BusDiag.lib library into CODESYS as described below:

1. Click on the "Resources" tab.

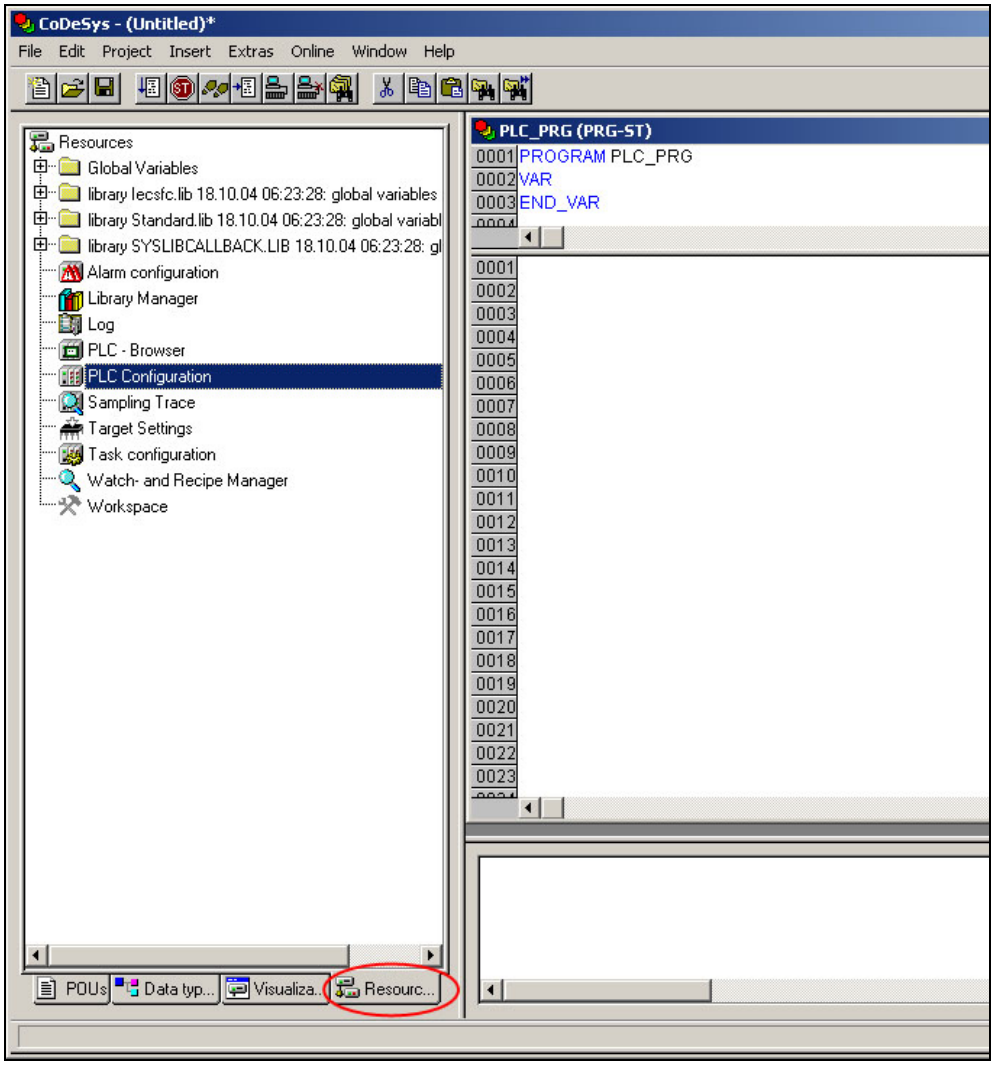


Figure 91: "Resources" tab

2. Double click in the left field on "Library Manager."

3. Click on **Insert > Additional Library…** in the menu bar. The "Open" dialog will open. Select the BusDiag.lib and click on **[Open]** to add it to this project.

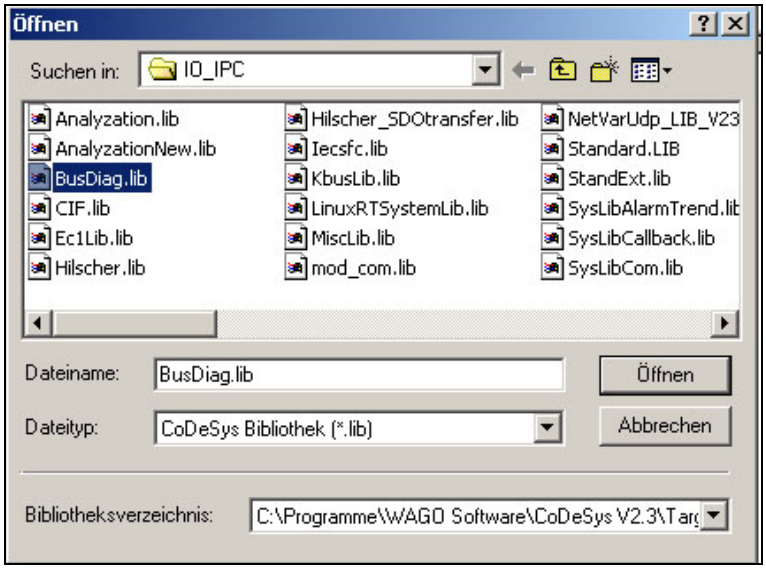


Figure 92: "Open" dialog

4. In the menu bar, click on the "Box" symbol.

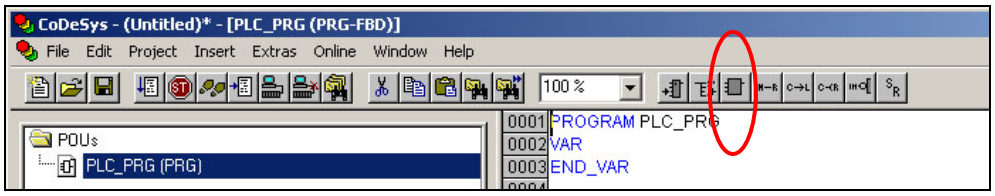


Figure 93: Box symbol in the menu list, FBD programming language

5. Press the **[F2]** key on your keyboard. The "Input Assistant" dialog will open. Click on the "Standard function block" option and select the function block DiagGetBusState().

6. Create an instance of the function block DiagGetBusState(). Here, enter a name above the function block. In this example, it is "GeneralBusInformation".

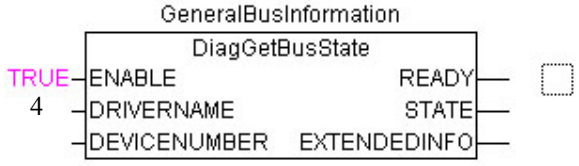


Figure 94: Instance of the function block DiagGetBusState() in FBD

7. Call up the function block DiagGetBusState() for the slave diagnostics from the library BusDiag.lib.

8. Create an instance of the DiagGetState() function block. In this example it is "DiagnosticsNode."

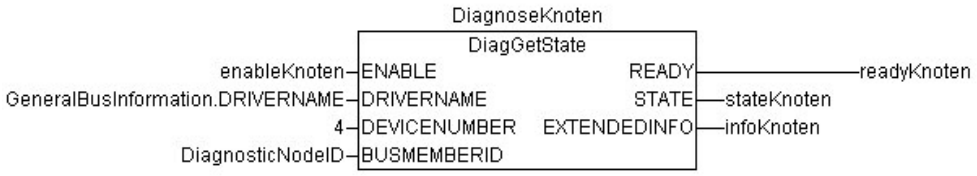


Figure 95: Function block DiagGetState() in FBD

In this example, both function blocks are called up during the program sequence. In order to not prolong the cycle times during the program sequence, do not set the input "ENABLE" of DiagGetState() to "TRUE" until you have conducted a diagnostic.

12.5.3 Calling the Diagnostic Block

Call the function block as shown in the following image.

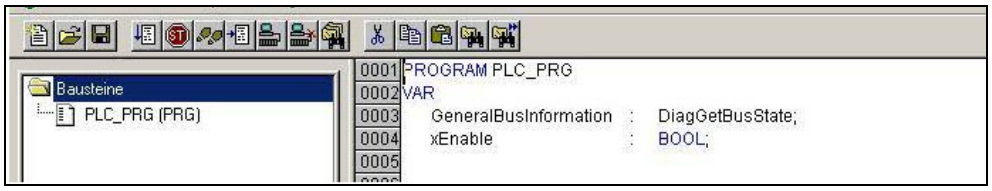


Figure 96: Offline view of the variable window in CODESYS

12.5.4 Performing Bus Diagnostics via DiagGetBusState()

To perform bus diagnostics, proceed as follows:

1. Log in to CODESYS. To do this, click in the menu bar on **Online > Login**. Now the variable window displays the information on the variables (Online View).

2. Start the PLC program by clicking in the menu bar on **Online > Run**. Starting causes the call up of the function block DiagGetBusState() and the diagnostic information is output to the array EXTENDEDINFO.

In the online view of the variable window, the array EXTENDEDINFO provides information on the status of the slaves. One entry is reserved for each slave in the array. The slave address is assigned to the array index. In this example, it is the slaves with the station addresses 2 and 5 that store the diagnostic information.

Note

Display the diagnostic information
The diagnostic information is only displayed for the duration of one program cycle. If the diagnostic information is to be available for longer, a suitable program must be written.

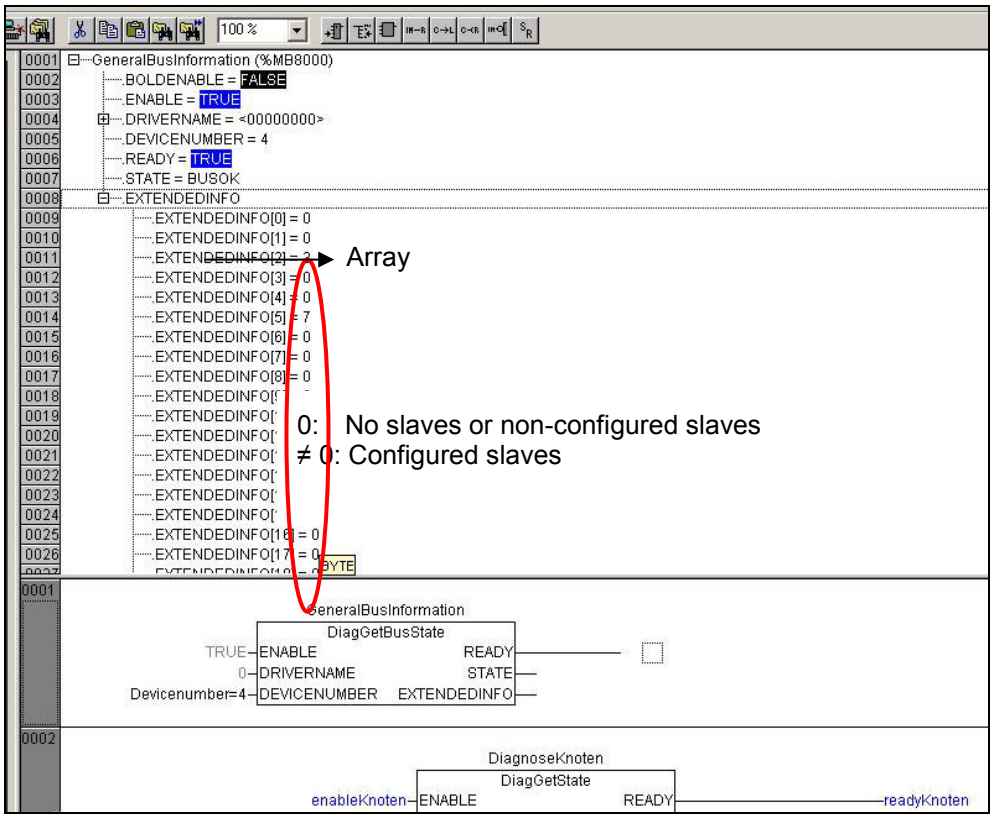


Figure 97: Online view of the variable window (upper window) in FBD

3. The binary code facilitates the evaluation of the individual diagnostic bits. You can have the diagnostic information of the array EXTENDEDINFO displayed as binary code. To do this, right click in the variable window and select **binary**.

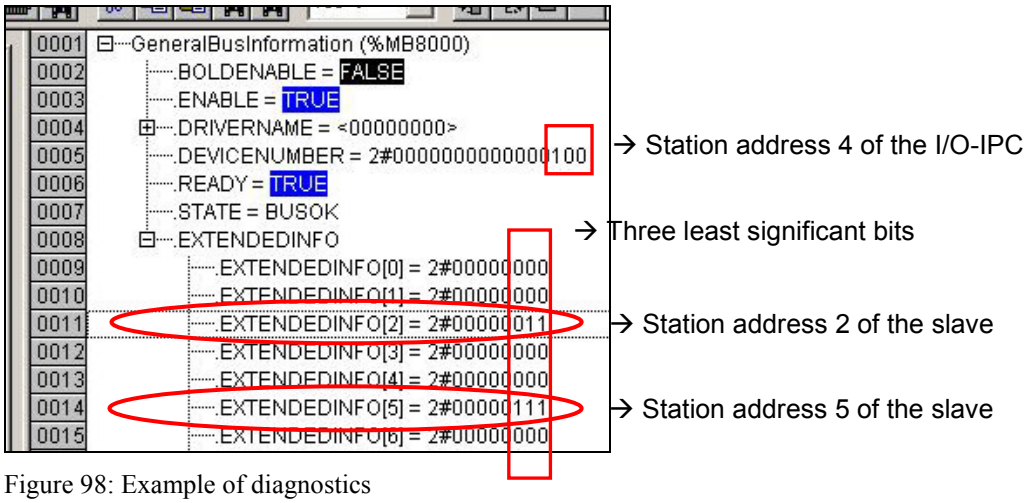


Figure 98: Example of diagnostics

4. Compare the three lowest value bits of the diagnostic information of the slaves with station addresses 2 and 5 with the bits from the following table:

Table 71: Bits of the diagnostics information

2. bit		**1. bit**		**0. bit**	
1	0	1	0	1	0
Diagnostic information is available on the slave.	No diagnostic information is available on the slave.	Slave is active.	Slave is inactive.	Slave projected.	Slave not projected.

- The slave with station address 2 delivers the value 011. This means that the slave is projected and active.
- The slave with station address 5 delivers the value 111. This means that the slave is projected and active and contains diagnostic information. To evaluate this diagnostic information, the participant diagnostics must be carried out. See Section "Performing Device Diagnostics via DiagGetState()".

Note

Diagnostic information
If READY = TRUE, then STATE indicates the current bus status by one of the following values:

BUSOK: All configured slaves are in data exchange with the DP master.

BUSFAULT: One or more configured slaves are not in data exchange with the DP master
BUSNOTCOMMUNICATION: All configured slaves are not in data exchange with the DP master.

12.5.5 Performing Device Diagnostics via DiagGetState()

If the bus diagnostics have revealed that an I/O module contains diagnostic information, then conduct participant diagnostics on the corresponding slave. To do this, proceed as follows:

1. Call up the function block DiagGetState() by setting the input ENABLE to "True."

2. Specify the slave that provides the diagnostic information at the input variable BUSMEMBERID. In this example it is the slave with the fieldbus address 5.

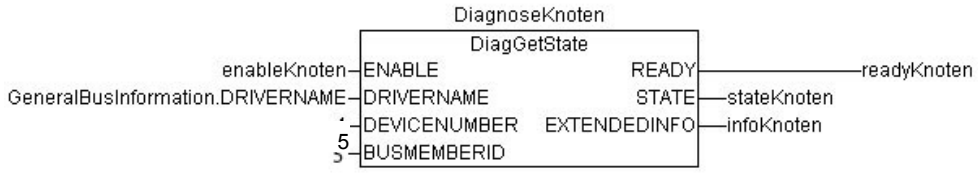


Figure 99: Diagnostics call DiagGetState()

- DRIVERNAME:
  The input parameter DRIVERNAME is specified through the instance data of the function block DiagGetBusState.

- DEVICENUMBER:
  The DEVICENUMBER for the I/O-IPC must always be 4.

12.5.6 Evaluation of the CANopen Diagnostics (Emergency Messages)

The array elements [0] to [15] listed in the illustration below are reserved for the CANopen status information in bytes. The emergency messages of the slaves are stored starting with array element [16].

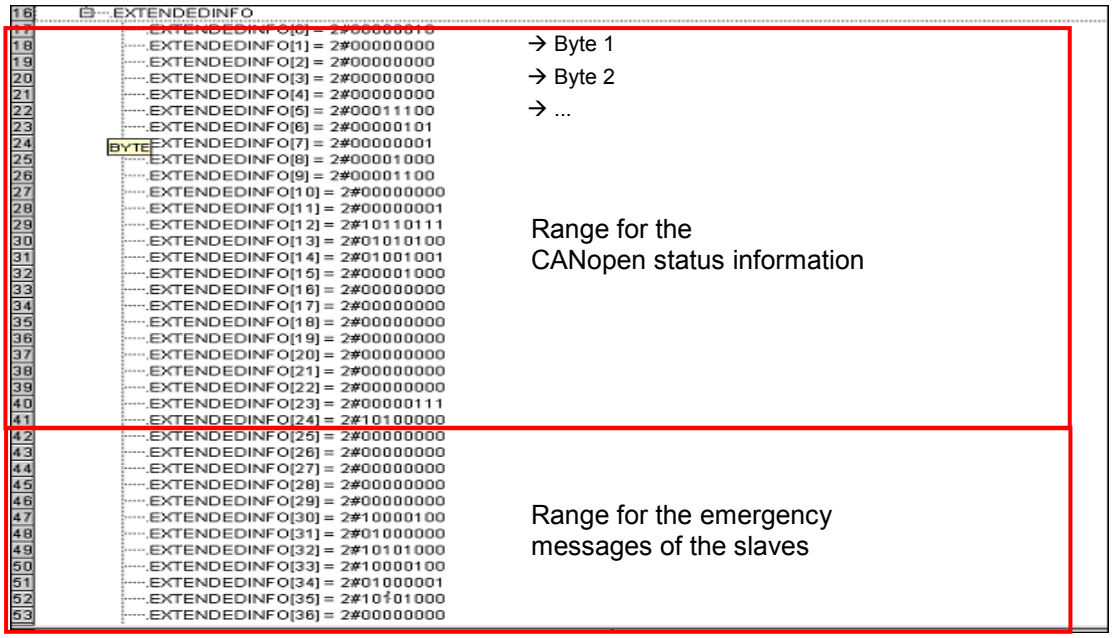


Figure 100: Online view of the array EXTENDEDINFO in the binary representation

The CANopen status information and the slaves' emergency messages are described on the following pages.

Description of the Diagnostic Information of the Function Block DiagGetState.EXTENDEDINFO for CANopen

The following table describes the content of the array EXTENDEDINFO[0-23].

EXTENDEDINFO[0]:	Slave address
EXTENDEDINFO[1]:	0 (free)
EXTENDEDINFO[2]:	0 (free)
EXTENDEDINFO[3]:	0 (free)
EXTENDEDINFO[4]:	0 (free)
EXTENDEDINFO[5]:	48, length of the diagnostic structure
EXTENDEDINFO[6]:	10 (free)
EXTENDEDINFO[7]:	1 (free)
EXTENDEDINFO[8]:	**Connection status of the slave** Bit 1: Slave does not respond. Bit 2: Error memory full. Bit 3: Slave is incorrectly parameterized. Bit 4: "Node guarding" protocol is enabled. Bit 5: Reserved Bit 6: Reserved Bit 7: Reserved Bit 8: Slave is disabled.
EXTENDEDINFO[9 – 12]:	**Device Type** Profile number and additional information for the slave. These four bytes are read via the object 0x1000 during start-up of the slave.
EXTENDEDINFO[13]:	**Station status** Feedback of the slave status from the "Node guarding" protocol.If the "Node guarding" protocol is enabled for this slave, the status is indicated in the array element EXTENDEDINFO[13]. 1 Disconnected (station failure) 2 Connecting 3 Preparing 4 Prepared 5 Operational (Operational Mode) ... 127 Pre-operational (Pre-Operationaln Mode and Stopped)
EXTENDEDINFO[14]:	**Actual error** This byte shows the online error information of the slave. See the tables on the following page.
EXTENDEDINFO[15]	**Emcy_ Entries** This byte contains the number of emergency telegrams stored in the following data range.

EXTENDEDINFO[16 – 23]	**Emcy_Data** The emergency messages are saved in EXTENDEDINFO[16 – 23]. An emergency message consists of the "error code," "error register," and "additional code": EXTENDEDINFO[16]: Error Code LSB EXTENDEDINFO[17]: Error Code MSB EXTENDEDINFO[18]: Error Register EXTENDEDINFO[19]: Additional Code EXTENDEDINFO[20]: Additional Code EXTENDEDINFO[21]: Additional Code EXTENDEDINFO[22]: Additional Code EXTENDEDINFO[23]: Additional Code For a detailed description of the emergency messages, consult the manuals for the CANopen slave (e.g. 750-337, "Emergency" section).

Actual error:

Table 72: Actual error

Err_event	Cause	Range	Solution
30	There is no response to the "node guarding" in the set time: The I/O-IPC cannot communicate with the slave.	Slave	Check the cabling of the power supply and fieldbus for damage.
			Increase the "Node guarding" time.
31	Slave is no longer operational	Slave	Execute a hardware reset. To do this, press the reset button (46) on the I/O-IPC or switch the I/O-IPC off and then on again.
			Execute the NMT "Reset_Node" service.
			Restart the SPS program.
32	Sequence error in the "guarding" protocol	Slave	Execute a hardware reset. To do this, press the reset button (46) on the I/O-IPC or switch the I/O-IPC off and then on again.
			Restart the SPS program.
33	-	-	-
34	The slave cannot respond to configuration telegrams.	Slave	Check the cabling of the power supply and fieldbus for damage.
			The slave must be operational.
35	The set profile "DSP301, 306" of the I/O-IPC does not match that of the slave.	Control configuration	On the "CAN Parameters" tab, deactivate the "DSP301, V.4.01 ..." checkbox.
36	The "Device type" of the slave does not match the I/O modules plugged in.	Control configuration	Enter the correct "Device type" on the "CAN Parameters" tab. This depends on the type (digital/analog) of the I/O modules plugged into the slave. For more information, see section "CANopen I/O-IPC PLC Control Configuration".
37	Invalid SDO response received	Slave	Slave does not match the CANopen profile.
38	SDO data length < 8 bytes	Slave	Slave does not match the CANopen profile.

12.6 Data Exchange of Simple CAN Subscribers with the I/O-IPC

The EDS file "Generic CAN device" reduced to the essential was created to simplify the control configuration when incorporating CAN-Layer-2 devices. The EDS file contains 16 send and receive PDOs, each of which have 8x1byte entries. You only have to deactivate the CANopen typical configuration and monitoring telegrams for this subscriber.

You can also execute the control configuration with any EDS file for CANopen.

1. To integrate the CANopen-I/O-IPC in the control configuration, right-click on "PLC Configuration" and select "Append CANMaster."

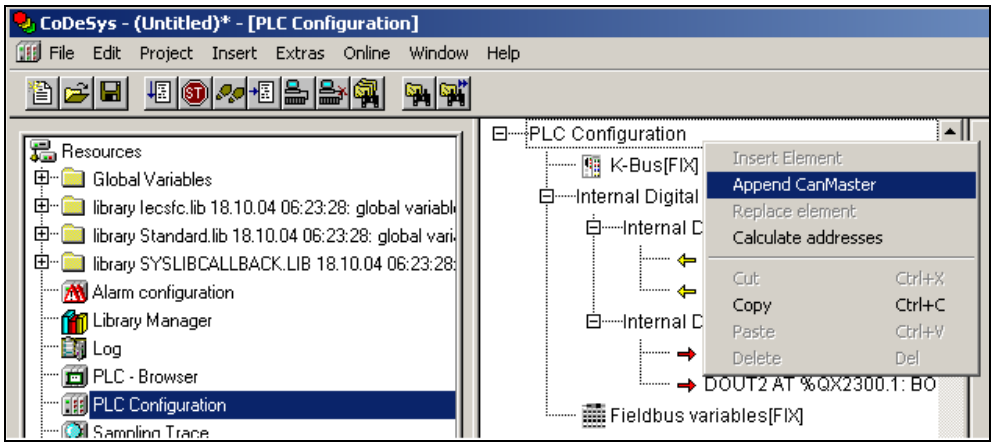


Figure 101: Attaching the CANopen master

2. On the "CAN parameters" tab, select the desired baud rate.

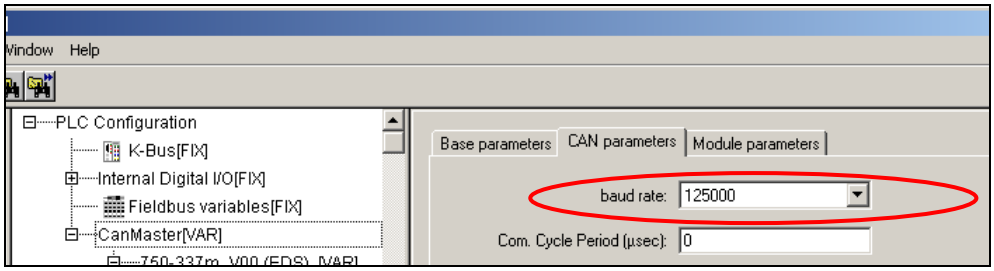


Figure 102: Setting the baud rate

3. To insert the slave, right-click on the I/O-IPC (CANMaster[VAR]) and select **Append Subelement > Generic CAN-Device.**

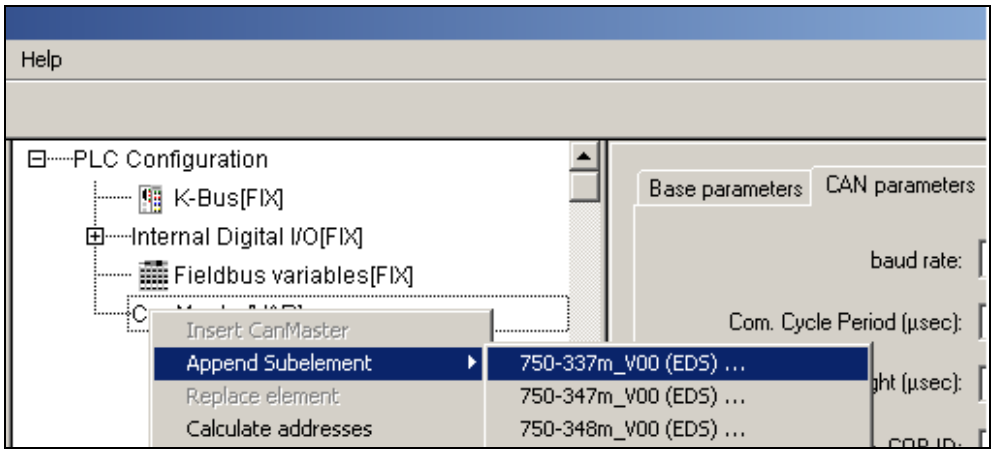


Figure 103: EDS file "Generic CAN-Device"

4. Open the "Module Parameter" tab of the slave. For communication with simple CAN-Layer-2 devices, deactivate "BasicSDO" (0x1000, 0x1005, 0x100C, 0x100D, 0x1014) via "EnableCANopenStartup" (= "No").

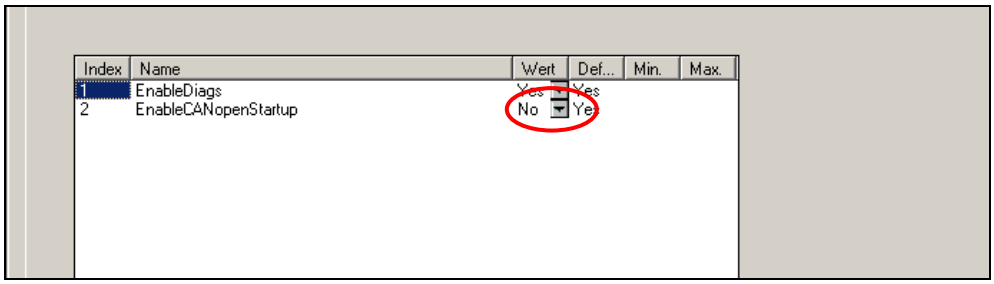


Figure 104: "Module Parameter" CAN tab

5. Open the "CAN parameters" tab of the slave. Deactivate the parameters "CommSDO" and "MappingSDO."

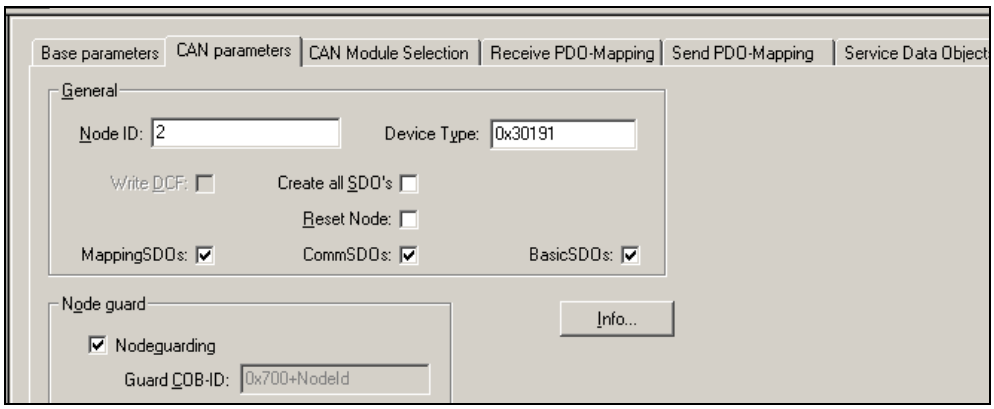


Figure 105: "CAN Parameters" CAN tab

6. To configure the CAN frames for CAN-Layer-2 devices, see section "CANopen Slaves" > Receiving/Sending PDO Mapping.

13 Incorporation of C-Functions as CODESYS Library

For the use of any C- or also Linux functions in CODESYS, the import interface described in the following is available. This can be used to load and use Linux libraries dynamically into the CODESYS runtime system.

13.1 Example for Linking a Dynamic Library

The following section will explain to you by means of an example the procedure for linking a dynamic library using a C-function in CODESYS.

13.1.1 Developing and Compiling a Linux Shared Library

Before you can use C functions within a CODESYS program, you must develop and compile a Linux shared library. Proceed in the following manner:

1. Create a file with the necessary functions. In this example, the file "libmytest.c" was created for this purpose with a "MyTestFunction" function and "unsigned short" data types.

```
#include <stdio.h>
unsigned short MyTestFunction(unsigned short value)
{
    return value+=2;
}
```

Figure 106: "libmytest.c" file

2. Compile and link the file by entering the following command in the Linux console:

```
gcc libmytest.c -Wall -shared -o libmytest.so
```

In Linux, the parameter "-shared" must be used when calling up the "gcc" compiler.

With an error-free compilation of the file, a dynamic library "libmytest.so" is generated with the parameter "-shared" that contains the "MyTestFunction" C-function.

Note

File name of the Linux libraries
The file name of the Linux libraries must begin with lib.

13.1.2 Creating a Description File for the CODESYS Runtime System

Create a description file with the fixed name "extlibs.ini" in order to make the C-functions known to the CODESYS runtime system.

1. So that the runtime system can recognize the Linux library and correctly link to it during startup, you must create an INI file that contains all library names to be loaded dynamically and their function names. The INI file for the example appears as follows:

```
[EXT_LIB_LIST]
1=mytest

[mytest]
1=MyTestFunction
```

Figure 107: "extlibs.ini" file

2. If you would like to add additional libraries under the entry EXT_LIB_LIST, add a continuous index under the corresponding library tag, followed by library names (e.g. "2 = CSV File"). It is not necessary to provide the transfer and return parameters of the functions at this point.

The correspondingly expanded file would appear as follows:

```
[EXT_LIB_LIST]
1=mytest
2=CSV-File

[mytest]
1=MyTestFunction

[CSV-File]
1=ReadCSVString
2=WriteCSVString
```

Figure 108: "extlibs.ini" file

13.1.3 Copying a Library and INI File and Restarting the CODESYS Runtime System

To link the library and the INI file to the CODESYS runtime system, proceed as follows:

1. Copy both of the newly created files (libmytest.so and extlibs.ini) onto the I/O-IPC. Use a USB memory, a CF card, FTP or NFS to do this.

2. Copy the file extlibs.ini into the directory */home/codesys*.

3. Copy the library into the directory */lib* or */usr/lib*.
   There is also the possibility of copying the new library to any place in the I/O-IPC. To do this, adjust the environment variable LD_LIBRARY_PATH in the Linux console you are using before each restart of CODESYS, e.g.:

   ```
   env LD_LIBRARY_PATH=/home/codesys ./plclinux_rt
   ```

4. Before the CODESYS runtime system can be restarted, you must enter the following command in the Linux console:

   ```
   ps A
   ```

5. Determine the PID of the program "plclinux_rt" from the displayed list.

6. Stop the CODESYS runtime system by entering the command `kill <PID>` with the previously determined PID, e.g.,

   ```
   kill 2069.
   ```

7. To restart the runtime system, enter the following command in the Linux console:

   ```
   plclinux_rt &.
   ```

If there is a discrepancy between the INI file and the library, an error message will be displayed on the Linux console when the CODESYS runtime system is restarted.

Note

CODESYS runtime system
The CODESYS runtime system must be restarted by a user with superuser rights.

Note

Alter the library
Do not alter the library while it is being used by the CODESYS runtime system since, otherwise, access violations may occur.

13.1.4 Creating an IEC Library

In order to be able to use the incorporated library within CODESYS library functions, the function prototypes must be stored in an external CODESYS library. To do this, proceed as follows:

1. Open a new CODESYS project by selecting **File > New** in the menu bar.

2. In the window "Target Settings", select "None" and click on the **[OK]** button.

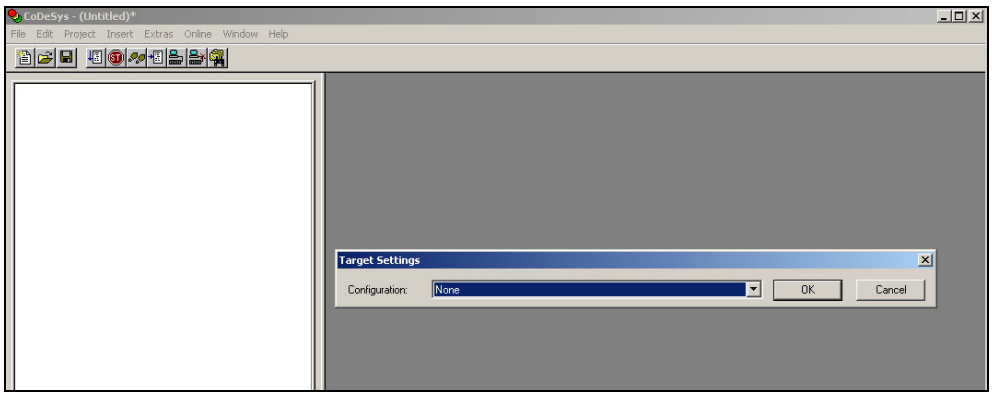


Figure 109: "Target Settings" window

3. Make the settings listed in the "New POU" window (see figure below). The name of the POU must be the same as that of the previously created C file. In doing so, pay attention to upper and lower case letters.

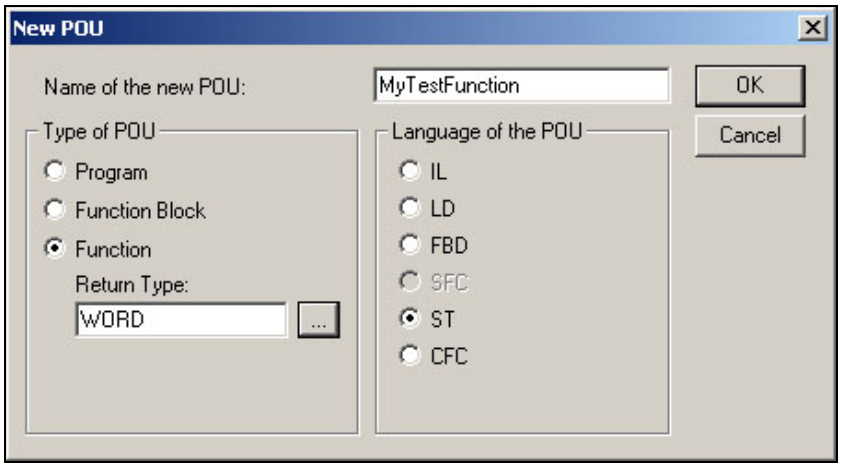


Figure 110: "New POU" window

4. Then define the input parameter with *value : WORD*; and add a semicolon in the program part of the function (lower window). Otherwise, a CODESYS error will occur.

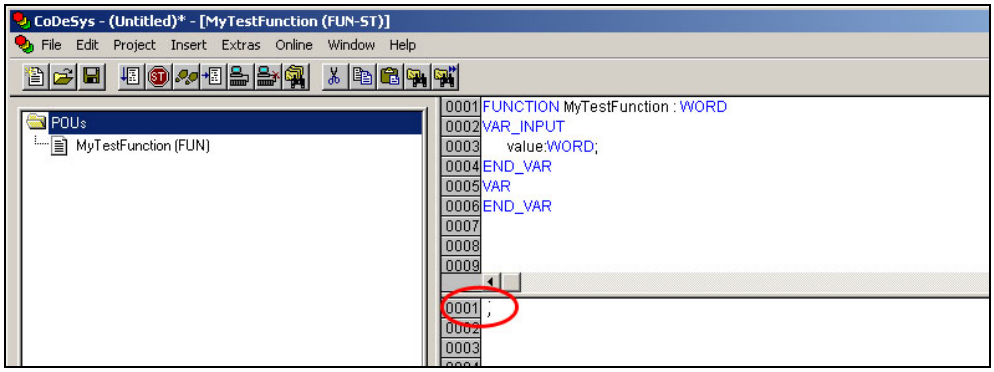


Figure 111: "MyTestFunction" window

5. Select **File > Save as…** in the menu bar. Enter "mytest.lib" as the file name, select the file type "External library" and click on **[Save].**

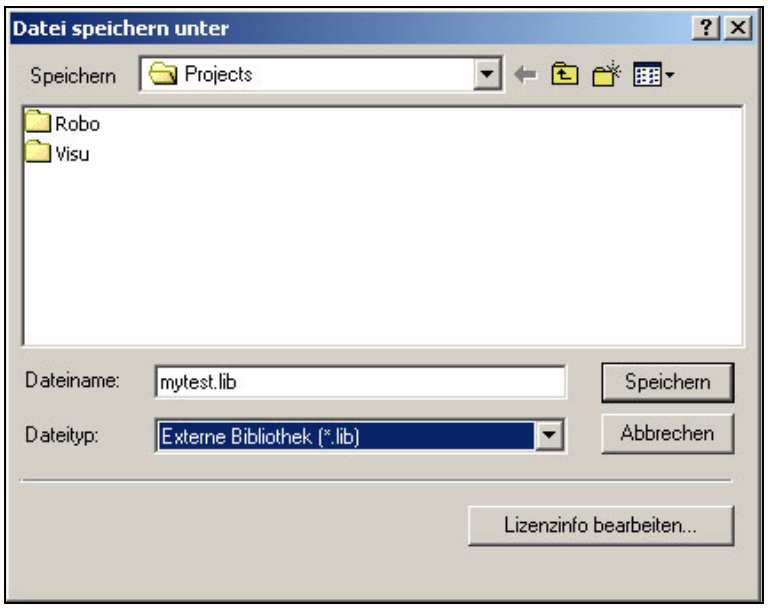


Figure 112: "Save File as" window

If the library contains several functions, allow these functions to be linked as well. In doing so, functions with several transfer parameters are also possible.

13.1.5 Linking a Library to the CODESYS Project

To link the previously created library mytest.lib to CODESYS, proceed as follows:

1. Click on **File** in the menu bar and select **New**.

2. Open the selection field for "Target Settings" and select the I/O-IPC you are using. In this example, it is the 758-876-111.

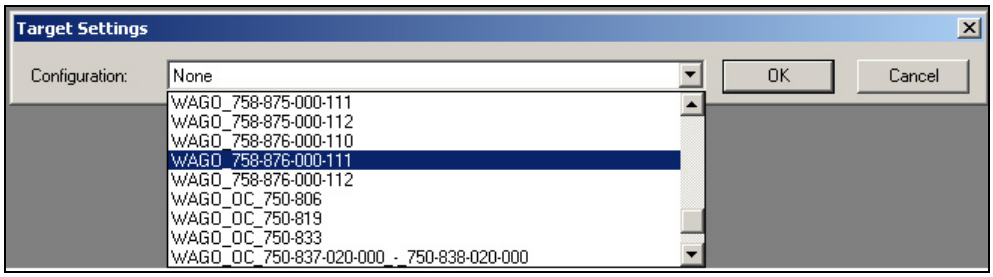


Figure 113: "Target Settings" window (1)

3. Click on the **[OK]** button. The "Target Settings" window opens.

4. Click on the **[OK]** button in the "Target Settings" window.

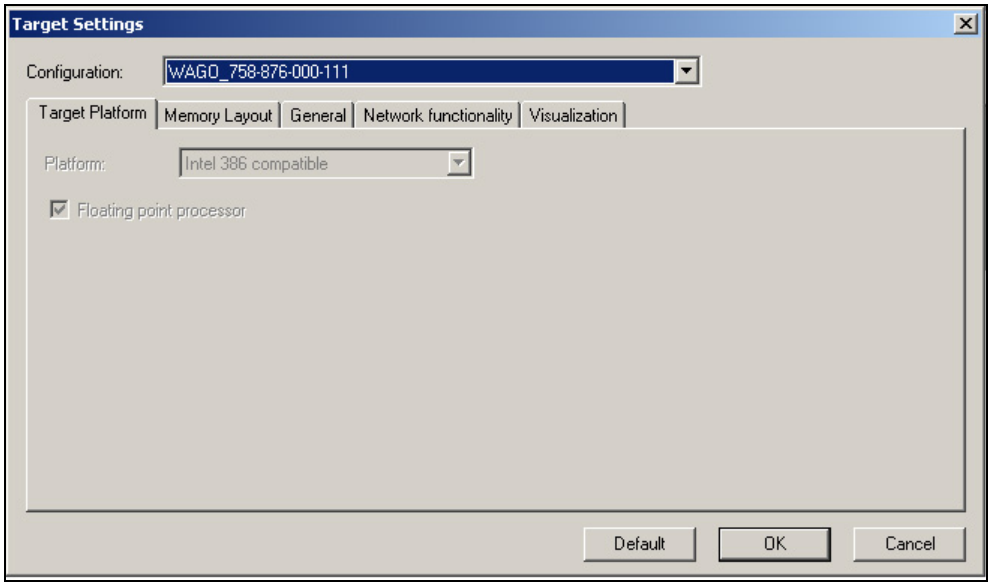


Figure 114: "Target Settings" window (2)

5. Click on **[OK]** in the "New POU" window.

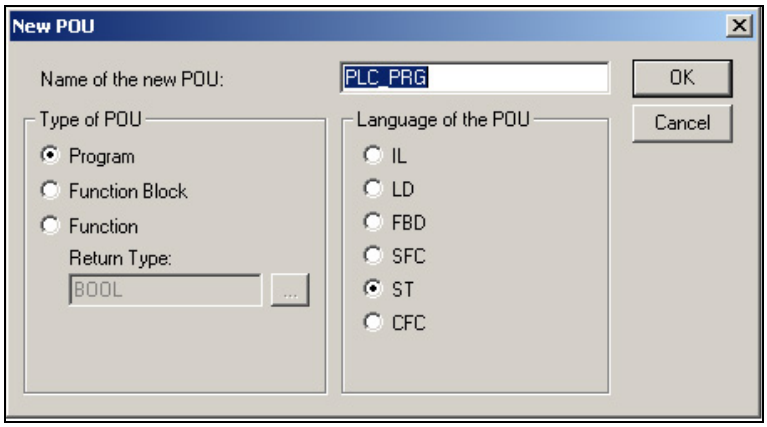


Figure 115: "New POU" window

6. Click on the "Resources" tab.

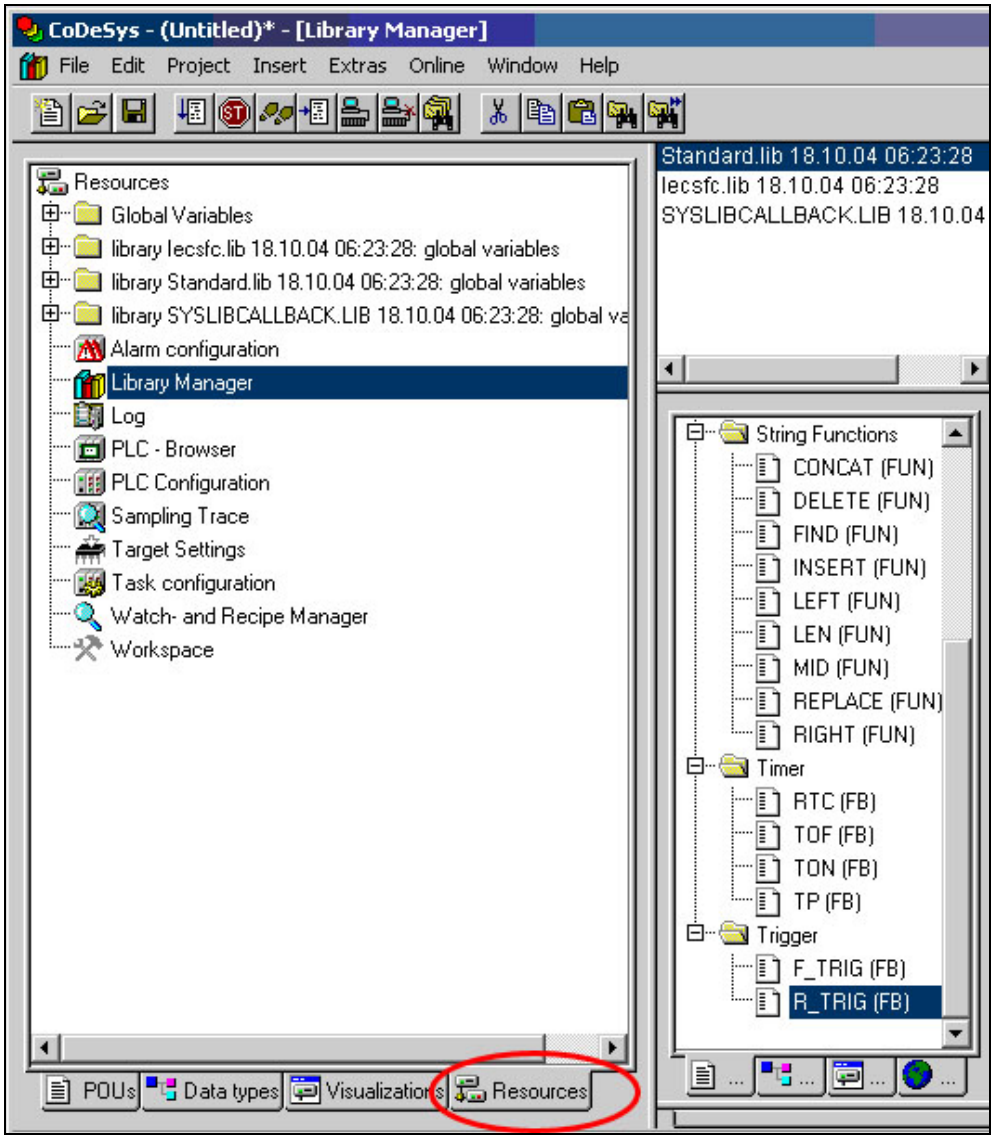


Figure 116: "Resources" tab

7. Double click in the left field on "Library Manager."

8. Click on **Insert > Additional library…** in the menu bar and select mytest.lib.

9. Click on the "POUs" tab.

10. Then call up the function in CODESYS as follows:

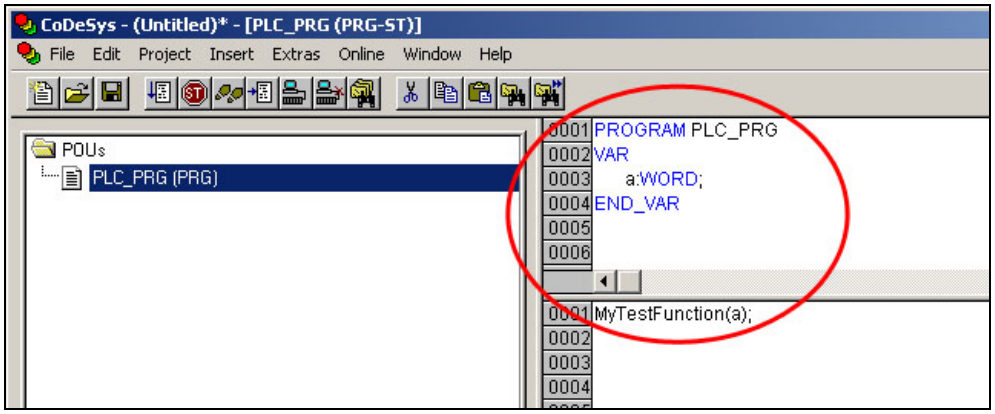


Figure 117: "PLC_PRG(PRG)" window

13.2 Special Features

13.2.1 Data Types

All CODESYS data types can be used as transfer parameters. Here, the CODESYS data types are interpreted in C as follows:

Table 73: Data Types

CODESYS	C/C++
BOOL	char
BYTE	char
WORD	unsigned short
DWORD	unsigned int
LWORD	unsigned long
SINT	signed char
USINT	unsigned char
INT	short
UINT	unsigned short
DINT	int
UDINT	unsigned int
LINT	long int
ULINT	unsigned long int
REAL	float
LREAL	double
STRING	char[]

13.2.2 Structures

Structures can also be transferred. In doing so, it is important that the data types be maintained exactly. The structures must also be defined with the "packed" attribute. Therefore, the following CODESYS structure

```
TYPE t_teststruct :
STRUCT
      a : BYTE;
      b : WORD;
      c : INT;
      d : DWORD;
      e : REAL;
      f : POINTER TO STRING;
END_STRUCT
END_TYPE
```

Figure 118: "Example.lib" file

would appear in C as follows:

```
struct t_teststruct{        // Codesys-Style:
    char a;                 //  BYTE
    unsigned short b;       //  WORD
    signed short c;         //  INT
    unsigned int d;         //  DWORD
    float e;                //  REAL
    char *f;                //  POINTER TO STRING
} __attribute__((packed));
```

Figure 119: "Example.h" file

Pay attention to the data types and the "packed" attribute in the "Beispiel.h" file.

Furthermore, an init-function must be created in the library for each structure created in CODESYS. For the file "Beispiel.h", the init-function might look like the following:

```
char t_teststructinit(struct t_teststruct *pteststruct, char
bRetain)
{
    pteststruct->a = 0;
    pteststruct->b = 0;
    pteststruct->c = 0;
    pteststruct->d = 0;
    pteststruct->e = 0;
    pteststruct->f = NULL;
    return 1;
}
```

In doing so, the name of the function must be made up of the name of the structure and the string "init" (e.g., `t_teststruct`**`init`**). The function is called up one time when CODESYS is started. It has a pointer to the structure itself as the transfer parameter and a BOOL value, which is not relevant in this case.

13.2.3 Parameter Transfer by Reference or by Value

It is possible to transfer the parameters by means of reference or by value.
In doing so, it is important that the correct sequence and the data types of the parameters are used for transferring to the function. For this purpose, the data types from the table in section "Data types" must be adhered to. If falsely declared values are accessed within the C program, memory access errors may occur. The CODESYS runtime system will then delete (suspend) the access violating task immediately.

13.3 Additional Applications

It is possible to start any Linux program or script using encapsulation in a C-function.

C-functions can also, for example, contain an init-function, which generates its own Linux threads and therefore contains independent programs. These init-functions can be called up through CODESYS system events such as PLC-Start or PLC-Stop. In this way, complete applications can also be encapsulated in their own thread.

In init-functions, pointers to common data structures, with which a convenient data interface between CODESYS and C-applications is made possible, can also be transferred.

14 Operating System

14.1 Linux Kernel Used

For the I/O-IPC, an RT-Preempt real-time kernel is used. This is a kernel that has been provided with the corresponding real-time patch. This, as well as the kernel, is available under GPL on the Internet:
http://www.kernel.org/pub/linux/kernel/projects/rt/.

This real-time extension provides the following advantages:

- Completely priority-controlled processes in the real-time area.
- Possibility for using processes in the user area with the CVS-Scheduler ("Completely Fair Scheduling").
- Prioritization of interrupt processing.
- The system timer is based on dynamic tick.
  This way, reaction times in the I/O-IPC are no longer bound to a fixed time pattern. As a result, cyclic processes can be started in µs.

14.2 Grand Unified Bootloader (GRUB)

GRUB is used as the bootloader for the I/O-IPC. To change the start settings of the Linux kernel, press one of the following keys within the waiting time you have set during the start phase of GRUB:

- A key on the keyboard connected to the I/O-IPC
- In the case of an opened terminal program, a key on the PC keyboard

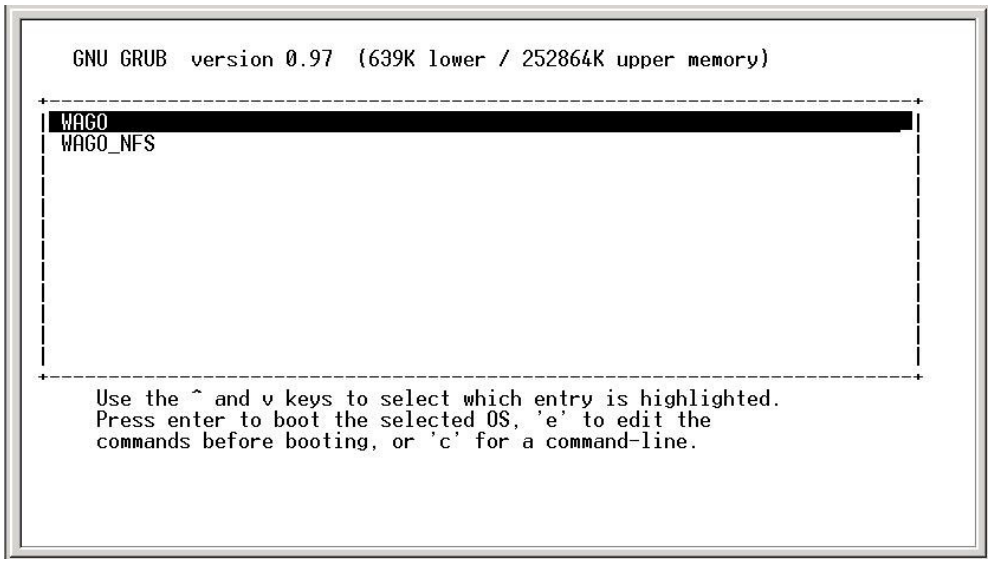


Figure 120: Serial console "hyperterminal"

In the GRUB, you have a choice of two start settings for the Linux file system:

- WAGO
  Starting of the Linux file system from the internal flash memory.
- WAGO_NFS
  Starting of the Linux file system from a remote NFS partition, to be defined.

To alter the bootstring (transfer parameters in the kernel), press the E key on the keyboard. This will enable you to determine a fixed IP address under **IP=** or to set the resolution of the monitor connected to the I/O-IPC under **VGA=**.

The change to the startup behavior is not saved permanently. If this behavior is to be permanently saved, the file */boot/grub/menu.lst* in the Linux file system must be altered.

14.3 Linux Startup Process

After switching on the I/O-IPC, the BIOS starts first. If you would like to implement the settings known to the PC there, press **[F1]** on the keyboard connected to the I/O-IPC. No other changes to the I/O-IPC are necessary at this point.

Following BIOS, the bootloader GRUB starts, which starts the kernel. During the startup phase of the kernel, all the hardware is configured.

After the hardware is configured, the kernel starts the first user space process (init). As with large distribution, init starts the start scripts in */etc/rc.d/...* in alphabetic and numeric sequence.

If additional user programs are to be started, these can also be started in this directory using a start script, and can also be automatically stopped when powering down the I/O-IPC.

CODESYS is started with the last rc.d script. The target visualization of CODESYS is displayed on the connected monitor. Using **[Alt] + [F2]** on the connected keyboard, change over to the Linux console and, using **[Alt] + [F1],** access the target visualization again.

Note

Behavior during the boot process
When booting, the file system is checked.
The check can delay the system startup time by several seconds.

14.4 Linux Console

The Linux consoles are available via keyboard as follows:

1. Linux Console	2. Linux Console	3. Linux Console
Target visualization	Linux	IPC configuration tool
[Alt] + [F1]	**[Alt] + [F2]**	**[Alt] + [F3]**

If you want to deactivate individual consoles, you must modify the /etc/inittab file on the device. You can do this using the integrated vi editor or via an FTP upload/download. The /etc/inittab file contains the following lines:

tty1:2345:respawn:/sbin/getty 38400 tty1

tty2:23:respawn:/sbin/getty 38400 tty2

tty3:23:respawn:/etc/init.d/ipcconfig start

The three command lines start the three consoles described in the table.
If you comment individual lines using the "#" character, the corresponding commented Linux console is not longer automatically executed during system start.

14.4.1 Access to the Linux Console

DANGER

Change passwords
Standard passwords are documented in these instructions and therefore do not offer adequate protection. Change these passwords to meet your particular needs.

You can access the Linux console through different paths. For one, through Telnet; for another through the RS-232 interface. The Linux console can also be accessed through a monitor on the DVI-I interface in combination with a USB keyboard.

In the condition as delivered to the customer, the I/O-IPC is equipped with the following users:

Table 74: Users for the Linux Console

Name	Password
root	wago
admin	wago
user	user
guest	guest

Change Password:

```
passwd [user]
```

You can also create and delete your own users:

```
sudo adduser [user]

sudo deluser [user] sudo
```

Note

Be careful when deleting users!
The `deluser` command can be used to delete **superusers**. This can result in you no longer having access to the device. If you want to restore access, the device must be reset using a firmware download.

14.4.1.1 Access over Telnet

In order to access the I/O-IPC over Telnet, use a terminal program such as minicom (under Linux) or hyper terminal (under Windows).

For use of the hyper terminal, the following settings in the login interface have to be adapted:

Host address: IP address of the ETHERNET interface of the I/O-IPC used
Connect via: TCP/IP

On top of that, you can also access the I/O-IPC through the textual console of Linux or MS-DOS using Telnet. This procedure is described below:

1. Connect the X8 ETHERNET interface of the I/O-IPC to your PC through an ETHERNET patch cable.

2. Open a console of your PC.

3. Enter the command `telnet <IP-ADRESSE des I/O-IPC>`.

Figure 121: DOS console 1

4. Enter your user name (See chapter "Access to the Linux console").

Figure 122: DOS console 2

5. Enter the password for your user. The Linux console of the I/O-IPC opens in the HOME directory (~) of the selected user.

14.4.1.2 Access via RS-232 Interface and Terminal Program

In order to access the Linux console via RS-232 interface using a terminal program, proceed as follows:

1. In the WBM or IPC configuration tool, assign the Linux console to the RS-232 interface. For this, see section “Administration”.

2. Connect the serial interface of the PC with the serial interface X6 (9) of the I/O-IPC using a null modem cable.

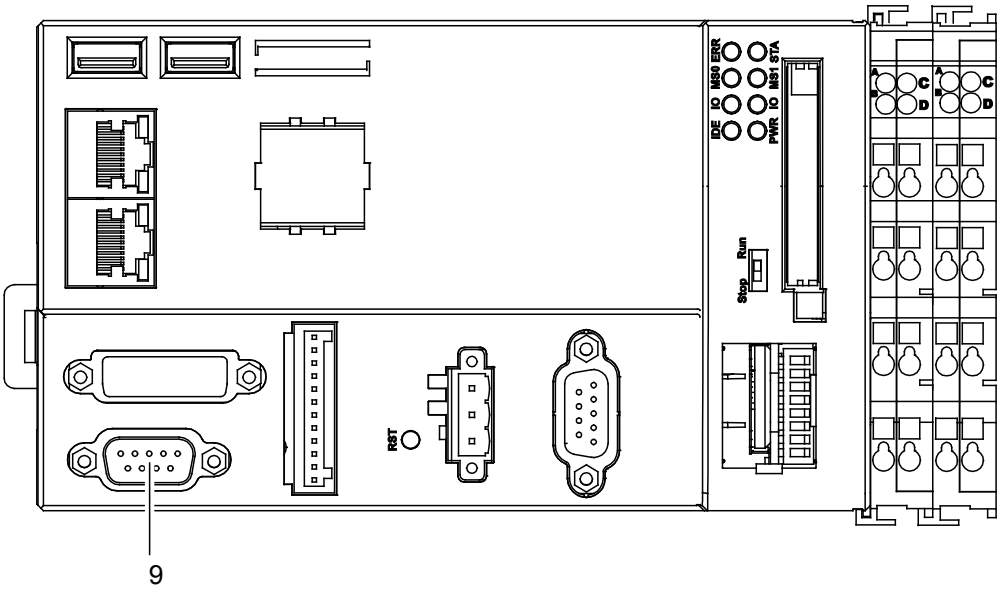


Figure 123: RS-232 interface X6

3. Open a terminal program (Linux: e.g. minicom, Windows: e.g. hyper terminal) on your PC.

4. Enter the communication parameters of the I/O-IPC serial interface set beforehand into the terminal program:

   - Speed: 115200 bit/sec
   - Data width: 8 Bit
   - Parity: none
   - Stop bits: 1 Bit
   - Flow control: none

5. The Linux console start screen appears.

6. Enter your user name (See section "Access to the Linux console")

7. Enter the password for your user name. The Linux console of the I/O-IPC opens in the HOME directory (~) of the selected user.

14.4.1.3 Access over keyboard and monitor (DVI-I and USB Interface)

In order to access the Linux console over a monitor/touch screen connected to the DVI-I interface and a USB keyboard, proceed as follows:

1. Connect a monitor to the DVI-I interface X7 (10) of the I/O-IPC.

2. Connect a USB keyboard to one of the two USB interfaces X10 (1) or X11 (2).

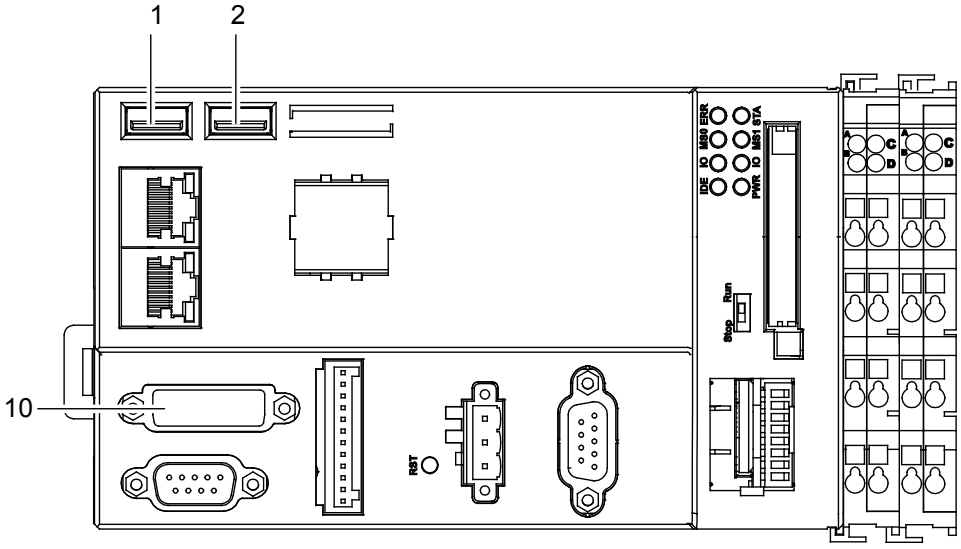


Figure 124: DVI-I interface X7 and USB interfaces X10/11

3. After starting Linux, the starting image of the target visualization appears on the monitor.

Note

Monitor/Touch screen resolution
If only a dark screen is displayed, then the resolution of the monitor/touch screen may not be right. Change the resolution with WBM (see section "'HMI Settings' Page").

4. Using **[Alt] + [F2]** on the connected keyboard, change over to the Linux console.

5. Enter your user name (See section "Access to the Linux console").

6. Enter the password for your user name. The Linux console of the I/O-IPC opens in the HOME directory (~) of the selected user.

You can display the Linux start messages again through *cat /proc/kmsg* (or *dmesg*).

14.4.2 Installed Applications

The I/O-IPC is delivered with a basic image that already contains the most important applications in the file system. The following applications, among others, are included:

- Bootloader: GRUB
- File system support for Ext2, Fat
- Console initialization: getty
- FTP client/server
- Telnet client/server
- SSH client/server
- Webserver (lighttpd)
- PHP5
- BootP/DHCP clients
- NFS client
- Event-Manager (udevd) for automatically integrating USB memories
- NTP client

14.4.3 Construction of the File System

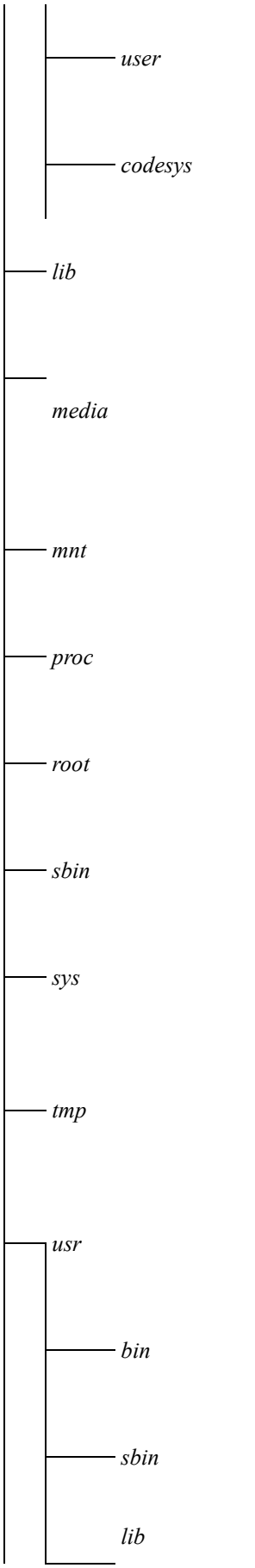
The file system of the internal flash memory is partitioned as follows at delivery:

Table 75: Construction of the file system

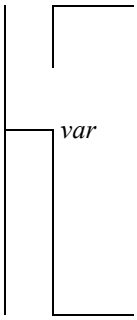
Name	Size
Master boot record and bootloader (grub)	about 1.5 MB
Linux system partition 1	about 40 MB
Linux system partition 2	about 40 MB
Home partition	about 40 MB
Remaining flash memory, not partitioned	about 6 MB

The file system also contains, as is customary for modern Linux distributions, the following directories with the established programs/files:

bin	Programs that can be executed
boot	Kernel image and configuration of the bootloader.
dev	Device driver files for the entire periphery of the I/O-IPC
etc	Global configuration files of the I/O-IPC
config-tools	WAGO-specific, executable configuration tools
network	Configuration files for the ETHERNET interfaces.
init.d	Start scripts
rc.d	Links to start scripts. All scripts named **Sxx_** are executed during booting of the I/O-IPC.
home	
guest	Home directory of a user who is logged on as a **guest**.



user	Home directory of a user who is logged on as a **user**.
codesys	CODESYS directory: This contains, if applicable, the boot application and the necessary files for target or web visualization.
lib	Directory with all dynamic libraries (shared objects).
media	Directory used as the **mountpoint** for the devices integrated through **automount** (udevd) (e.g., USB memories). Devices formatted using the FAT file system are integrated in **/media** in a subdirectory using their partition names (For further details, see chapter "SysLibFile, SysLibDir, SysLibFileAsync").
mnt	**mountpoint**, which can be used by the user. This directory has no function in the condition as supplied to the customer.
proc	Virtual directory that provides information from the kernel.
root	Home directory for the users **root** and **admin**
sbin	Executable programs that can be used by a user logged on as **superuser**.
sys	Virtual directory used as the interface to different kernel modules.
tmp	RAM disk that can be used for temporary files. Files stored in this directory are no longer available after restarting. The RAM disk does not reserve parts of the memory until files are written to it.
usr	
bin	Programs that can be executed
sbin	Executable programs that can be used by a user logged on as **superuser.**
lib	Directory with all dynamic libraries (shared objects)



Directory that the web server has accessed. Here lie the HTML/SSI pages of the web server and the CGI parser that can be executed over the web.

14.4.4 Installed Shell (BASH)

A BASH (Bourne Again Shell) containing built in commands such as `cd` is installed for the I/O-IPC. In addition, the BASH makes available the environment variables and enables navigation of the file system and the starting of programs.

14.4.5 Busybox and Other Help Programs

Busybox combines many programs from the standard Linux distributions into one program in order to minimize memory requirements in the file system. The program is called up through symbolic links only. Busybox evaluates names being called up and can, in this way, implement different functions in one program together.

The following programs, among others, are made available by Busybox:

- `mount`
  Integrating drives
  By choosing the name (PC: format > volume label) of integrated memory media (e.g. CF card, USB memory), these are indicated in the file system (/media).

- `reboot`
  Restart the I/O-IPC

- `ifconfig`
  Indicates current ETHERNET interface settings. Use WBM or IPC-Configuration-Tool to change these settings.

- `rmdir`
  Delete directory

Note

Compiling Busybox
When compiling from Busybox, you can determine which programs are to be integrated. In doing so, you adapt the program size according to the required function. In the I/O-IPC, all the functions necessary for a minimal system are implemented.

Additional programs are also installed on the I/O-IPC, such as, for example, `cp` or `ftp`. Moreover, the programs listed in section “Installed applications” are also included, such as the following help programs (examples):

- htop, top
  Program for displaying priorities and used resources individual processes.

- sed
  Help program for the simple parsing of text/configuration files.

- gdbserver
  Remote debugger

- cyclictest
  Measurement program for recording the real-time capability of the system.

- zip, unzip
  For packing and unpacking zip archives.

14.5 Drivers for Special Hardware Parts

User space I/O drivers (UIO) can be implemented through the real-time capable kernel. This enables a memory mapping functionality to be accessed directly from the user space. In this manner, you can access the process image of the connected I/O modules as well as other hardware areas.

14.6 Incorporation of a USB Printer

You can connect a printer for ASCII texts to the USB interface.

To print the words Test0, Test1 and Test2, for example, enter the following commands:

```
echo –e “\n\nTest0\n“ >/dev/lp0
echo –e “\n\nTest1\n“ >/dev/lp0
echo –e “\n\nTest2\f“ >/dev/lp0
```

The test words appear printed on a page as follows:

```
Test0
        Test1
                Test2
```

14.7 Installed Services of the ETHERNET Interface

All sorts of client/server services are activated for the ETHERNET interface in the I/O-IPC as delivered to the customer. A selection of the installed services is listed below:

- Telnet server
  The Telnet server enables the connection of several participants of a network to the Linux console of the I/O-IPC.

- Telnet client
  This is used to access the console of a remote Telnet server through a network.

- FTP server
  The FTP server enables several participants of a network to access the file system of the I/O-IPC.

- FTP client
  This enables the exchange of files with remote FTP servers.

- Web server
  Participants in the network can call up information on the settings of the I/O-IPC and configure them using an Internet browser.

- NTP client
  The NTP client allows the exact time of an NTP server to be requested.

- NFS client
  This is used to integrate released network drives from NFS servers.

- SNMP server
  The SNMP server allows you to monitor and control the I/O-IPC from a PC. You also have the option to exchange with the PLC program via SNMP data (see attachment, section "WAGO_Snmp.lib")

14.7.1 Telnet Server (telnetd)

The Telnet server of the I/O-IPC is activated when delivered to the customer. The telnetd daemon is activated over the ETHERNET with a corresponding request. This creates a new Linux console with access over Telnet. The Telnet daemon is started or stopped with the script */etc/rc.d/S07_telnetd*.

To log on to the Telnet server, enter your user name and the associated password. The Linux console of the I/O-IPC opens and the HOME directory of the selected user is displayed.

14.7.2 FTP Server (pure-ftpd)

"File Transfer Protocol" is used to exchange files between a PC and the I/O-IPC. To do this, Linux cannot be installed on the PC since Windows also provides FTP client functions.

In the I/O-IPC as delivered to the customer, the FTP server is activated. The FTP daemon is started and stopped with the script */etc/rc.d/S09_pureftd* and waits after that for corresponding requests.

Exchange of FTP Files between a PC and the I/O-IPC over ETHERNET

To exchange files between a PC and the I/O-IPC over ETHERNET using FTP, an FTP client is necessary. To do this, you can use both an Internet browser (Internet Explorer) and an FTP program (e.g., Filezilla, DOS Console, Linux Console). The FTP connection is carried out through port 21, which many FTP programs require you to enter.

Using Windows Internet Explorer as the FTP client, for example, you would enter the following address in the address line of Windows Explorer:

`ftp://username:password@hostname.`
Example: `ftp://user:user@192.168.1.17.`

Information on users and passwords for the Linux console can be found in section "Access to the Linux Console".

Using the DOS console as the FTP client, for example, you would enter the following command in the console:

```
ftp <hostname/IP>
```

Figure 125: DOS console

After you have logged on, you can call up help with `help`, which describes the available commands of the slave's FTP console. For example:

```
put file.html       // Write a file to the I/O-IPC
get file.gif        // Read a file from the I/O-IPC
cd/                 // Change to the root directory
help                // Show all available commands
```

14.7.3 NFS Server

The NFS ("Network File System") is a service that permits cross-network access of files. If, for example, the local directory */home* is made available in the network, the following lines must be added in the file */etc/exports*:

*/home *(rw,sync,all_squash,anonuid=<uid>,anongid=<gid>)*

For *<uid>* and *<gid>,* enter the Linux user number and group number through which you are logging on. These numbers can be determined as follows:

```
> id
uid=0(root) gid=0(root)
```

For this example, the lines of the file */etc/exports* are as follows:

*/home *(rw,sync,all_squash,anonuid=0,anongid=0)*

14.7.4 FTP Client

The FTP client allows files to be loaded or written from an FTP server. The FTP client is installed in the directory */bin* and can thus be used by every user from every directory. Port 21 is used for the FTP protocol.

FTP Client Operation

To use the FTP client, an FTP server with a known user must be available on a remote PC for FTP access. To start the FTP client, enter the following command:

`ftp <IP/hostname>`

Example: `ftp 192.168.1.11`

The FTP server queries the user and password. After successful login, you can execute commands on the server. You can query available server commands using `help`. The server then shows a list with all available FTP commands. You can obtain a description of a command by using `help <command>`, such as. `help cd`.

14.7.5 Web Server (lighttp)

Lighttp is a program under GPL and is especially characterized by its speed. The configuration relies on the Apache web server and can therefore be simply configured. PHP5 support is also available to the web server, which is already used for the WBM web pages.

The I/O-IPCs web server is activated when delivered. It provides a graphic interface through Web-Based Management by which you can configure the I/O-IPC. For this, see section „Configuration via Web-Based Management (WBM)“.

The already stored web pages can be found in the directory */var/www*. In the directory */var/www/cgi-bin/* is a CGI parser that enables the creation of dynamic web pages. Examples using the CGI parser can be found in the directory */var/www/wbm* and implement WBM for the configuration of the I/O-IPC.

14.7.6 NTP Client

An NTP client functionality is provided for the I/O-IPC through the program ntpclient. With NTP, the time can be requested from a remote NTP server through port 123.

If NTP servers are activated in the Internet, routing and firewalls must be suitably configured. In NTP, the time is transmitted as a 64-bit value and has a resolution of about 0.25 ns. The precision of the time transmission is given as +/-10 ms in the Internet and up to +/-200 µs in local networks.

The configuration and activation / deactivation of the NTP client is done through WBM.

14.7.7 NFS Client

An NFS client is integrated in the kernel, which enables the addition of remote drives to one's own file system. To integrate a directory from a remote system, it is assigned to the Linux directory structure like a partition of a hard drive with the command `mount`. To use the NFS service, an NFS server with a corresponding released directory must be available on the remote PC. The integration of the remote directory into the file system of the I/O-IPC is done with the following command:

```
mount -t nfs -o nolock <IP/hostname>:/<Verzeichnis>
/<lokales Verzeichnis>
```

Example: `> mount –t nfs –o nolock 192.168.1.12:/targetfs /mnt`

The drive */mnt* is present when the I/O-IPC is delivered to the customer. It is used to incorporate foreign drives. Access to the drive incorporated through NFS is conducted as if accessing a local directory. If drives are to be automatically incorporated during system startup, these can be entered in the directory */etc/rc.d* using a script.

14.7.8 SNMP Agent

The "Simple Network Management Protocol" is used to monitor and control network components. During communication via SNMP, SNMP managers (clients) and SNMP agents (servers) come into use.

The manager installed on a PC controls the agents installed on the I/O-IPC via a TCP/IP network. It can send queries to the I/O-IPC and receive responses. The agent is used to capture and transmit device data (name, status, OIDs, etc.).

The data of a device that the agent can access or modify are called SNMP objects. The SNMP objects are posted to the manager via the MIB file (Management Information Base). OIDs (Object Identifier) are responsible for unique addressing of the individual information within an MIB.

Use the Web-Based Management (WBM) to configure the SNMP agents of the I/O-IPC.

The SNMP is supported in version 1, 2c and 3. The SNMP agent is disabled in the I/O-IPC as delivered. In SNMP versions 1 and 2c, the exchange of messages is device-related. This requires the IP address of the manager to be specified. This IP address makes communication between manager and network node possible. In SNMP version 3, exchanging messages is user-related. Each device that knows the passwords set via WBM may read or write values from the I/O-IPC. With SNMPv3, the data can also be transmitted encrypted. This way, the requested values and those to be written cannot be easily decoded by other on the ETHERNET. Therefore, SNMPv3 is frequently used on security-relevant networks.

To create customer-specific variables (OID), the CODESYS library WAGO_Snmp.lib is available. Detailed information about the data packages that allow communication via SNMP is available in the attachment, Section " WagoLibNetSnmp.lib ".

15 Diagnostics

15.1 Operational Messages

All operational messages of the I/O-IPC are described in the table below. These messages are indicated by LEDs:

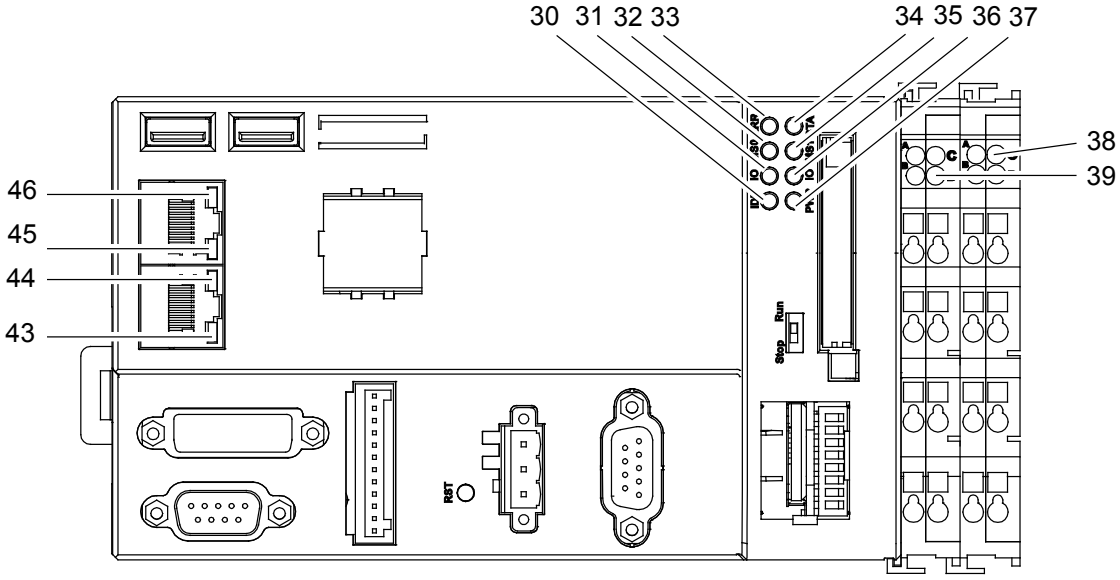


Figure 126: Identification of LEDs

Table 76: Operating Messages of IDE and PWR LED

Position	LED	Color/Status	Cause	Explanation/Remedy
30	IDE	Red flashing	The red flashing indicates that access to Flash storage (internal or CF card) is taking place.	-
37	PWR	Green	The power supply is available on the I/O-IPC.	-

Table 77: Operating Messages of „IO“-LEDs

Position	LED	Color/Status	Cause	Explanation/Remedy
31	IO	Red flashing/off	The I/O-IPC is in the startup phase.	-
		Red flashing	Internal data bus is in the initialization phase.	-
		Off	Speed indicator for updating the I/O module. If CODESYS is not used, only brief flashing is visible.	-
		Red flashing	Display of error message through blink code.	For more information, see section „Error Messages via I/O-LED“.
36	IO	Green flashing/off	The I/O-IPC is in the startup phase.	-
		Off	Internal data bus is in the initialization phase.	-
		Green flashing	Speed indicator for updating the I/O module. If CODESYS is not used, only brief flashing is visible.	-
		Off	Display of error message through blink code.	For more information, see section „Error Messages via I/O-LED“.

Table 78: Operating Messages of MS0- and MS1-LED

Position	LED	Color/Status	Cause	Explanation/Remedy
32	MS0	Off, red, red flashing	LED that can be freely programmed by the user.	-
35	MS1	Off, green, green flashing		

Table 79: Operating reports of „ERR“ aund „STA“ LEDs

Position	LED	Color/Status	Cause	Explanation/Remedy
33	ERR	Off	CANopen master is operational and can send or receive telegrams.	-
		Off	CANopen master sends a telegram.	-
		Red	Communication problem between the CANopen master and at least one CANopen slave.	-
		Off	The CANopen mater is still not configured.	
		Red	The CANopen is still not ready for use or no slave is configured.	
34	STA	Off	CANopen master is operational and can send or receive telegrams.	
		Green flashing (irregular)	CANopen master sends a telegram.	
		Green flashing (irregular)	Communication problem between the CANopen master and at least one CANopen slave.	
		Green	The CANopen mater is still not configured.	
		Green	The CANopen is still not ready for use or no slave is configured.	

Table 80: I/O-IPC Operating Reports

Position	LED	Color/Status	Cause	Explanation/Remedy
38	750-602 Supply Module, LED C	Green	24V supply voltage to the power jumper contacts available.	-
		OFF	24V supply voltage not available to the power jumper contacts.	Connect the power supply.
	Optional 750-626 Filter Module, LED A	Green	24V supply voltage available.	-
		OFF	24V supply voltage not available.	Connect the power supply.
	LED C	Green	24V supply voltage available to the power jumper contacts.	-
		OFF	24V supply voltage not available to the power jumper contacts.	Connect the power supply.
39	I/O- IPC interface	OFF	-	If the LED still lights up, you have not connected the supply voltage correctly. Use the power supply module or filter module for the power supply (see section "Power Supply").
43/45	ACT	OFF	No data is being exchanged via the ETHERNET network.	-
		Yellow flashing	Data is being exchanged via the ETHERNET network	-
44/46	LNK	Green	Link to the ETHERNET network is available.	-
		Off	I/O-IPC does not have a link to the ETHERNET network.	Check the cables for the X8/X9 ETHERNET interfaces.

15.2 Error Messages via I/O-LED

This section describes I/O LED (31) in details.

It shows diagnostic messages of the I/O-IPC as a blink code. The number of blink pulses indicates the error code and error argument. See the following sections.

If there are several error messages, the first message to occur chronologically always blinks until it is corrected. Only then will any subsequent error message be displayed.

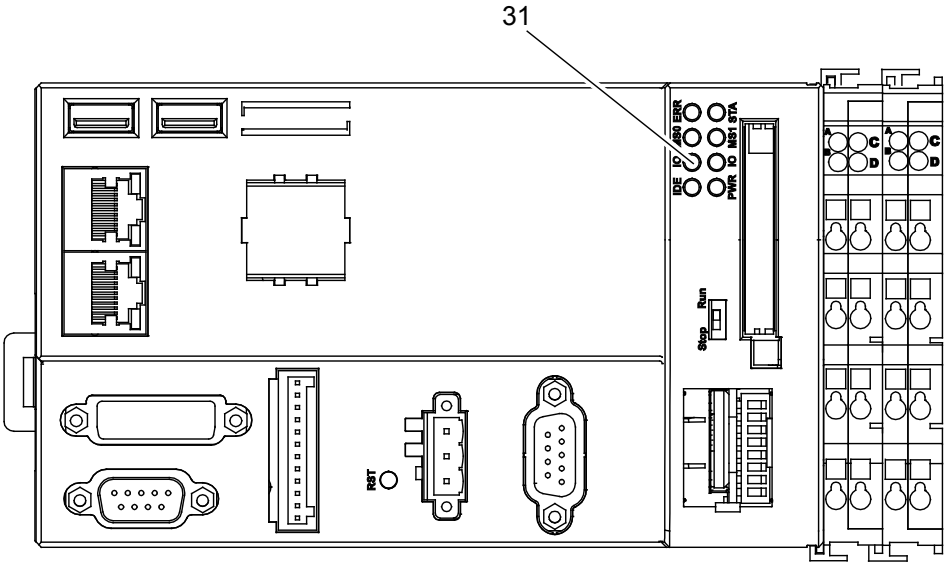


Figure 127: Display of blink codes by the I/O-LED

15.2.1 Progression of Blink Sequence

An error is always displayed as three blink sequences in a cyclic manner:

1. The first blink sequence (flickering) introduces the error message.

2. After a short break, the second blink sequence starts. The number of blink pulses indicates the exact **error code** that describes the type of error.

3. After another break, the third blink sequence starts. The number of blink pulses indicates the **error argument**, which provides additional descriptions of the error, e.g. on which of the 750 Series components connected to the I/O-IPC the error has occurred.

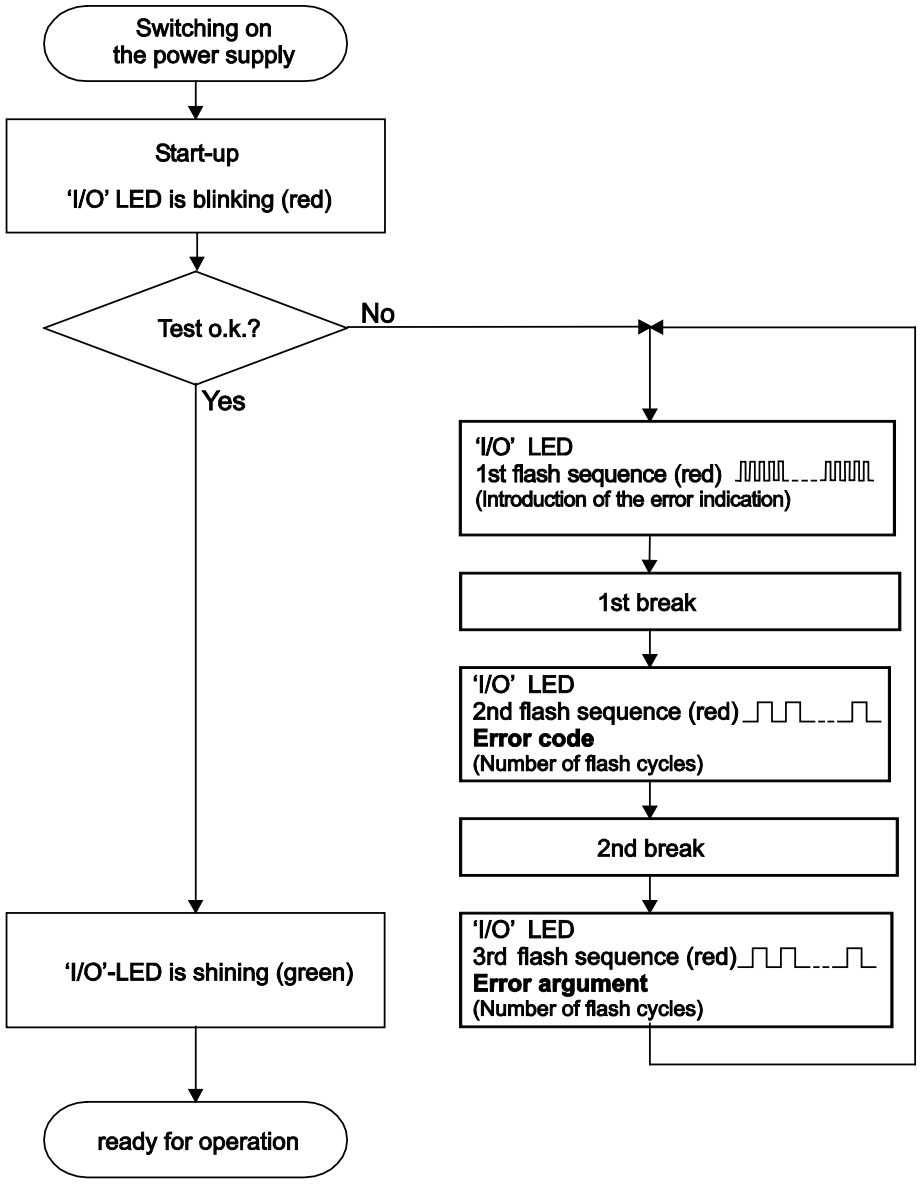


Figure 128: Blink sequence process diagram

15.2.2 Example of an Error Message via Blink Code

The following example explains the representation of an error message via blink code. A data error is displayed on the internal data bus caused by the removal of an I/O module located in the 6th position of the I/O-IPC.

Initiation of the Start Phase

1. The I/O LED begins with the initiation of the start phase: one cycle of about 10 Hz (10 blinks/second).

2. This is followed by a pause of about one second.

Error Code 4: Data Error in the Internal Data Bus

3. The I/O-LED blinks 4 cycles of about 1 Hz.

4. This is followed by a pause of about 1 second.

Error Argument 5: I/O Module at the 6th Slot

5. The I/O-LED blinks 5 cycles of 1 Hz.
   This means that an interruption has occurred on the internal data bus after the 5th I/O module.

6. The blink code starts flickering when the start phase is initiated again. If there is only one error, this process repeats.

15.2.3 Meaning of the Blink Codes and Procedures for Troubleshooting

In this section, all errors and warnings indicated by the I/O-LED are listed.

If the following errors and warnings cannot be eliminated with the indicated measures, please contact WAGO Support, and provide them with the blink code given by your I/O-IPC.

Phone: +49 571 887 555
Fax: +49 571 887 8555
E-mail: support@wago.com

Table 81: Meaning of Blink Codes and Measures to Eliminate Errors

Error Argument	Cause	Correction
Error Code 1: Hardware and Configuration Error		
-	Invalid parameter checksum of the internal data bus controller (I/O-IPC interface)	- Switch the supply voltage of the I/O-IPC off and change it. - Then switch the supply voltage on again.
1	Internal buffer overflow (max. amount of data exceeded) during inline code generation	- Switch the supply voltage of the I/O-IPC off. - Reduce the number of I/O modules. - Switch the supply voltage on again.
2	Data type of the I/O module(s) is not supported	Update the I/O-IPC firmware. If the error persists, one of the I/O modules is defective. Determine this as follows: - Switch the supply voltage off. - Place the end module in the middle of the connected I/O modules. - Switch the supply voltage on again. - If the I/O-LED is still blinking red, switch the supply voltage off again and place the end module in the middle of the first half of the I/O modules (from the I/O-IPC). - If the LED is no longer blinking, switch the supply voltage off and place the end module in the middle of the second half of the I/O modules (away from the I/O-IPC). - Switch the supply voltage on again. Repeat this procedure until you have determined the defective I/O module. Then exchange the defective module.

Table 81: Meaning of Blink Codes and Measures to Eliminate Errors

Error Argument	Cause	Correction
3	Unknown module type of the flash program memory	- Switch the supply voltage of the I/O-IPC off and change it. - Then switch the supply voltage on again.
4	Error occurred while writing to the flash memory	- Switch the supply voltage of the I/O-IPC off and exchange it. - Then switch the supply voltage on again.
5	Error occurred while erasing a flash sector	
6	The I/O module configuration after an internal bus reset differs from the one after the last I/O-IPC start-up.	Restart the I/O-IPC by - switching the supply voltage off and then on again, or - press the reset button on the I/O-IPC.
7	Error occurred while writing to the serial EEPROM	- Switch the supply voltage of the I/O-IPC off and exchange it. - Then switch the supply voltage on again.
8	Invalid hardware/firmware combination	
9	Invalid checksum in the serial EEPROM	
10	Fault when initializing the serial EEPROM.	
11	Error occurred while reading from the serial EEPROM	- Switch the supply voltage of the I/O-IPC off and reduce the number of I/O modules. - Then switch the supply voltage on again.
12	Time to access the serial EEPROM exceeded	- Switch the supply voltage of the I/O-IPC off and exchange it. - Then switch the supply voltage on again.
14	Maximum number of gateway or mailbox modules exceeded.	- Switch the supply voltage of the I/O-IPC off. - Reduce the number of gateway or mailbox modules. - Then switch the supply voltage on again.
Error Code 2: Not Used		
-	-	-

Table 81: Meaning of Blink Codes and Measures to Eliminate Errors

Error Argument	Cause	Correction
Error Code 3: Internal Data Bus Protocol Error		
-	Internal data bus communication error; defective I/O module cannot be determined.	If a power supply module (e.g., 750-602) is connected to the I/O-IPC, make sure that it is operating properly (see "LED Signaling" section). If this is the case,, then there is a defective I/O module. Determine this in the following manner: - Switch the supply voltage off. - Place the end module in the middle of the connected I/O modules. - Switch the supply voltage on again. - If the I/O-LED is still blinking red, switch the supply voltage off again and place the end module in the middle of the first half of the I/O modules (from the I/O-IPC). If only one I/O module is left over, but the LED is still blinking, then this module or the connecting clamp of the I/O-IPC is defective. Replace the I/O module or the I/O-IPC. - If the LED is no longer blinking, swich the supply voltage off and place the end module in the middle of the second half of the I/O modules (away from the I/O-IPC). - Switch the supply voltage on again. Repeat this procedure until you have determined the defective I/O module. Then replace it.

Table 81: Meaning of Blink Codes and Measures to Eliminate Errors

Error Argument	Cause	Correction
Error Code 4: Physical Error in the Internal Data Bus		
-	Internal data bus communication error or interruption of the internal data bus at the I/O-IPC	- Switch the supply voltage of the I/O-IPC off. - Connect an I/O module for process data to the I/O-IPC. - Connect the end module as the last to the I/O-IPC. If no error argument is given by the I/O-LED, the I/O-IPC interface is defective and the I/O-IPC must be replaced.
n*	Interruption of the internal data bus behind the nth process data module.	- Switch the supply voltage of the I/O-IPC off. - Replace the (n+1)th process data module. - Switch the supply voltage on again. I/O modules that do not provide any data are not considered (e.g., power supply module without diagnostics).
Error Code 5: Internal Data Bus Initialization Error		
n*	Register communication error during internal data bus initialization	- Switch the supply voltage of the I/O-IPC off. - Replace the (n+1)th process data module. - Switch the supply voltage on again. I/O modules that do not provide any data are not considered (e.g., power supply module without diagnostics).
Error Code 6: Design Error in the Node Configuration		
5	Maximum size of the process image exceeded	- Switch the supply voltage of the I/O-IPC off and reduce the number of I/O modules. - Switch the supply voltage on again.
Error Code 7: Not Used		
-	-	-
Error Code 8: Not Used		
-	-	-

Table 81: Meaning of Blink Codes and Measures to Eliminate Errors

Error Argument	Cause	Correction
Error Code 9: CPU Exception Error		
1	Invalid program statement	Malfunction of the program sequence. Contact WAGO Support.
2	Stack overflow	Malfunction of the program sequence. Contact WAGO Support.
3	Stack underflow	Malfunction of the program sequence. Contact WAGO Support.
4	Invalid event (NMI)	Malfunction of the program sequence. Contact WAGO Support.

16 Service

This section contains information on maintenance and service.

⚠ CAUTION

Hot underside!
High temperatures may occur on the underside of the I/O-IPC during operation. If the I/O-IPC has been in operation, allow it to cool off before moving it.

16.1 Replacing the Battery

If you replace the battery due to a lack of power supply, make sure that you have a new battery of type CR2032 (Li/MnO_2, 225 mAh). Please use batteries of the following manufactures: Matsushita Electric Industrial Co. LTD.

A capacitor supplies the missing power for the real-time clock for a short time. This allows the SRAM data to be retained during battery changes.

⚠ CAUTION

Danger of explosion if the battery is incorrectly installed!
Make sure that you insert the battery correctly (positive pole upwards). Otherwise, there is a danger of explosion and the possibility of injury to persons and property damage.

NOTICE

Electrostatic discharge!
Parts of the printed circuit board are accessible without the front plate. Take necessary ESD measures to prevent possible damages caused by electrostatic discharge.

Note

Charging the battery
The battery cannot be recharged.
Never open the battery, and never throw it in a fire.

Note

Life cycle of the battery
The life cycle of the battery depends on the ambient temperature. Therefore, it is recommended that the battery be replaced annually.

To replace the battery, proceed as follows:

1. Open the transparent cover of the battery compartment (51).

2. Remove the old battery (52) by pulling it out.

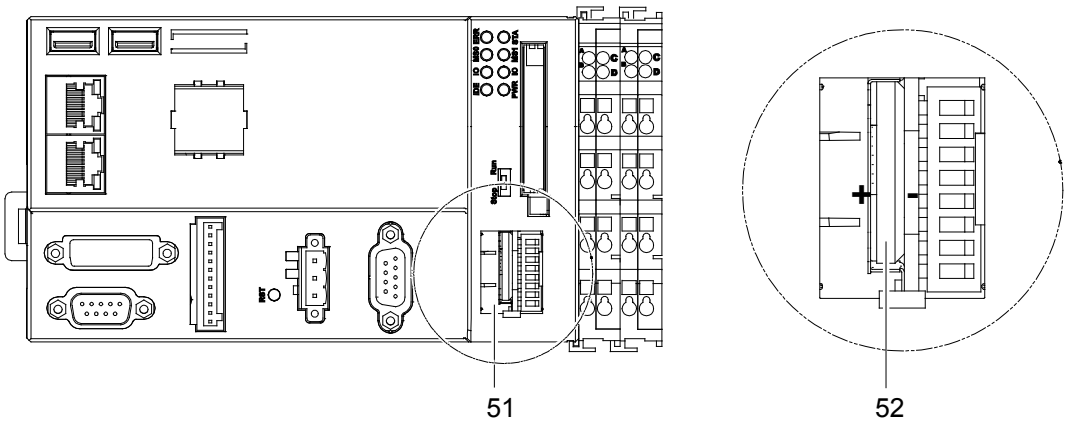


Figure 129: Changing the battery for the emergency power supply

3. Place the new battery, type CR2032, into the battery compartment as in the figure (plus pole to the left) until it snaps in palpably.

4. Close the cover.

16.2 Disposal

Dispose of the 750 Series components in accordance with the applicable laws. You can also turn to a certified disposal operation.

17 I/O Modules

17.1 Overview

For modular applications with the WAGO-I/O-SYSTEM 750/753, different types of I/O modules are available

- Digital Input Modules
- Digital Output Modules
- Analog Input Modules
- Analog Output Modules
- Special Modules
- System Modules

For detailed information on the I/O modules and the module variations, refer to the manuals for the I/O modules.

You will find these manuals on the WAGO web pages under http://www.wago.com.

Information

More Information about the WAGO-I/O-SYSTEM
Current information on the modular WAGO-I/O-SYSTEM is available in the Internet under: http://www.wago.com

17.2 Process Data Architecture for MODBUS/TCP

With some I/O modules, the structure of the process data is fieldbus specific.

MODBUS/TCP process image uses a word structure (with word alignment). The internal mapping method for data greater than one byte conforms to the Intel format.

The following section describes the process image for various WAGO-I/O-SYSTEM 750 and 753 I/O modules with MODBUS/TCP.

NOTICE

Equipment damage due to incorrect address!
Depending on the specific position of an I/O module in the fieldbus node, the process data of all previous byte or bit-oriented modules must be taken into account to determine its location in the process data map.

The structure of the process data mapping is identical for the PFC process image of the programmable fieldbus controller.

17.2.1 Digital Input Modules

Digital input modules supply one bit of data per channel to specify the signal state for the corresponding channel. These bits are mapped into the Input Process Image.

Some digital modules have an additional diagnostic bit per channel in the Input Process Image. The diagnostic bit is used for detecting faults that occur (e.g., wire breaks and/or short circuits).

When analog input modules are also present in the node, the digital data is always appended after the analog data in the Input Process Image, grouped into bytes.

17.2.1.1 1 Channel Digital Input Module with Diagnostics

750-435

Table 82: 1 Channel Digital Input Module with Diagnostics

Input Process Image							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
						Diagnostic bit S 1	Data bit DI 1

17.2.1.2 2 Channel Digital Input Modules

750-400, -401, -405, -406, -410, -411, -412, -427, -438, (and all variations),
753-400, -401, -405, -406, -410, -411, -412, -427

Table 83: 2 Channel Digital Input Modules

Input Process Image							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
						Data bit DI 2 Channel 2	Data bit DI 1 Channel 1

17.2.1.3 2 Channel Digital Input Module with Diagnostics

750-419, -421, -424, -425,
753-421, -424, -425

Table 84: 2 Channel Digital Input Module with Diagnostics

Input Process Image							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
				Diagnostic bit S 2 Channel 2	Diagnostic bit S 1 Channel 1	Data bit DI 2 Channel 2	Data bit DI 1 Channel 1

17.2.1.4 2 Channel Digital Input Module with Diagnostics and Output Process Data

750-418,
753-418

The digital input module supplies a diagnostic and acknowledge bit for each input channel. If a fault condition occurs, the diagnostic bit is set. After the fault condition is cleared, an acknowledge bit must be set to re-activate the input. The diagnostic data and input data bit is mapped in the Input Process Image, while the acknowledge bit is in the Output Process Image.

Table 85: 2 Channel Digital Input Module with Diagnostics and Output Process Data

Input Process Image							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
				Diagnostic bit S 2 Channel 2	Diagnostic bit S 1 Channel 1	Data bit DI 2 Channel 2	Data bit DI 1 Channel 1

Output Process Image							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
				Acknowledge-ment bit Q 2 Channel 2	Acknowledge-ment bit Q 1 Channel 1	0	0

17.2.1.5 4 Channel Digital Input Modules

750-402, -403, -408, -409, -414, -415, -422, -423, -428, -432, -433, -1420, -1421, -1422, -1423
753-402, -403, -408, -409, -415, -422, -423, -428, -432, -433, -440

Table 86: 4 Channel Digital Input Modules

Input Process Image							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
				Data bit DI 4 Channel 4	Data bit DI 3 Channel 3	Data bit DI 2 Channel 2	Data bit DI 1 Channel 1

17.2.1.6 8 Channel Digital Input Modules

750-430, -431, -436, -437, -1415, -1416, -1417, -1418
753-430, -431, -434

Table 87: 8 Channel Digital Input Modules

Input Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data bit DI 8 Channel 8	Data bit DI 7 Channel 7	Data bit DI 6 Channel 6	Data bit DI 5 Channel 5	Data bit DI 4 Channel 4	Data bit DI 3 Channel 3	Data bit DI 2 Channel 2	Data bit DI 1 Channel 1

17.2.1.7 8 Channel Digital Input Module PTC with Diagnostics and Output Process Data

750-1425

The digital input module PTC provides via one logical channel 2 byte for the input and output process image.

The signal state of PTC inputs DI1 … DI8 is transmitted to the fieldbus coupler/controller via input data byte D0.
The fault conditions are transmitted via input data byte D1.

The channels 1 … 8 are switched on or off via the output data byte D1. The output data byte D0 is reserved and always has the value “0“.

Table 88: 8 Channel Digital Input Module PTC with Diagnostics and Output Process Data

Input Process Image															
Input Byte D0								**Input Byte D1**							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Signal status DI 8 Chan-nel 8	Signal status DI 7 Chan-nel 7	Signal status DI 6 Chan-nel 6	Signal status DI 5 Chan-nel 5	Signal status DI 4 Chan-nel 4	Signal status DI 3 Chan-nel 3	Signal status DI 2 Chan-nel 2	Signal status DI 1 Chan-nel 1	Wire break/ short circuit DB/KS 8 Chan-nel 8	Wire break/ short circuit DB/KS 7 Chan-nel 7	Wire break/ short circuit DB/KS 6 Chan-nel 6	Wire break/ short circuit DB/KS 5 Chan-nel 5	Wire break/ short circuit DB/KS 4 Chan-nel 4	Wire break/ short circuit DB/KS 3 Chan-nel 3	Wire break/ short circuit DB/KS 2 Chan-nel 2	Wire break/ short circuit DB/KS 1 Chan-nel 1

Output Process Image															
Output Byte D0								**Output Byte D1**							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	0	DI Off 8 Chan-nel 8 0: Chan-nel ON 1: Chan-nel OFF	DI Off 7 Chan-nel 7 0: Chan-nel ON 1: Chan-nel OFF	DI Off 6 Chan-nel 6 0: Chan-nel ON 1: Chan-nel OFF	DI Off 5 Chan-nel 5 0: Chan-nel ON 1: Chan-nel OFF	DI Off 4 Chan-nel 4 0: Chan-nel ON 1: Chan-nel OFF	DI Off 3 Chan-nel 3 0: Chan-nel ON 1: Chan-nel OFF	DI Off 2 Chan-nel 2 0: Chan-nel ON 1: Chan-nel OFF	DI Off 1 Chan-nel 1 0: Chan-nel ON 1: Chan-nel OFF

17.2.2 16 Channel Digital Input Modules

750-1400, -1402, -1405, -1406, -1407

Table 89: 16 Channel Digital Input Modules

Input Process Image															
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data bit DI 16 Channel 16	Data bit DI 15 Channel 15	Data bit DI 14 Channel 14	Data bit DI 13 Channel 13	Data bit DI 12 Channel 12	Data bit DI 11 Channel 11	Data bit DI 10 Channel 10	Data bit DI 9 Channel 9	Data bit DI 8 Channel 8	Data bit DI 7 Channel 7	Data bit DI 6 Channel 6	Data bit DI 5 Channel 5	Data bit DI 4 Channel 4	Data bit DI 3 Channel 3	Data bit DI 2 Channel 2	Data bit DI 1 Channel 1

17.2.2.1 Digital Output Modules

Digital output modules use one bit of data per channel to control the output of the corresponding channel. These bits are mapped into the Output Process Image.

Some digital modules have an additional diagnostic bit per channel in the Input Process Image. The diagnostic bit is used for detecting faults that occur (e.g., wire breaks and/or short circuits). For modules with diagnostic bit is set, also the data bits have to be evaluated.

When analog output modules are also present in the node, the digital image data is always appended after the analog data in the Output Process Image, grouped into bytes.

17.2.2.2 1 Channel Digital Output Module with Input Process Data

750-523

The digital output modules deliver 1 bit via a process value Bit in the output process image, which is illustrated in the input process image. This status image shows "manual mode".

Table 90: 1 Channel Digital Output Module with Input Process Data

Input Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
						not used	Status bit “Manual Operation“

Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
						not used	controls DO 1 Channel 1

17.2.2.3 2 Channel Digital Output Modules

750-501, -502, -509, -512, -513, -514, -517, -535, (and all variations),
753-501, -502, -509, -512, -513, -514, -517

Table 91: 2 Channel Digital Output Modules

Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
						controls DO 2 Channel 2	controls DO 1 Channel 1

17.2.2.4 2 Channel Digital Input Modules with Diagnostics and Input Process Data

750-507 (-508), -522,
753-507

The digital output modules have a diagnostic bit for each output channel. When an output fault condition occurs (i.e., overload, short circuit, or broken wire), a diagnostic bit is set. The diagnostic data is mapped into the Input Process Image, while the output control bits are in the Output Process Image.

Table 92: 2 Channel Digital Input Modules with Diagnostics and Input Process Data

Input Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
						Diagnostic bit S 2 Channel 2	Diagnostic bit S 1 Channel 1

Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
						controls DO 2 Channel 2	controls DO 1 Channel 1

750-506,
753-506

The digital output module has 2-bits of diagnostic information for each output channel. The 2-bit diagnostic information can then be decoded to determine the exact fault condition of the module (i.e., overload, a short circuit, or a broken wire). The 4-bits of diagnostic data are mapped into the Input Process Image, while the output control bits are in the Output Process Image.

Table 93: 2 Channel Digital Input Modules with Diagnostics and Input Process Data 75x-506

Input Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
				Diagnostic bit S 3 Channel 2	Diagnostic bit S 2 Channel 2	Diagnostic bit S 1 Channel 1	Diagnostic bit S 0 Channel 1

Diagnostic bits S1/S0, S3/S2: = ‘00’ standard mode
Diagnostic bits S1/S0, S3/S2: = ‘01’ no connected load/short circuit against +24 V
Diagnostic bits S1/S0, S3/S2: = ‘10’ Short circuit to ground/overload

Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
				not used	not used	controls DO 2 Channel 2	controls DO 1 Channel 1

17.2.2.5 4 Channel Digital Output Modules

750-504, -516, -519, -531,
753-504, -516, -531, -540

Table 94: 4 Channel Digital Output Modules

Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
				controls DO 4 Channel 4	controls DO 3 Channel 3	controls DO 2 Channel 2	controls DO 1 Channel 1

17.2.2.6 4 Channel Digital Output Modules with Diagnostics and Input Process Data

750-532

The digital output modules have a diagnostic bit for each output channel. When an output fault condition occurs (i.e., overload, short circuit, or broken wire), a diagnostic bit is set. The diagnostic data is mapped into the Input Process Image, while the output control bits are in the Output Process Image.

Table 95: 4 Channel Digital Output Modules with Diagnostics and Input Process Data

Input Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
				Diagnostic bit S 4 Channel 4	Diagnostic bit S 3 Channel 3	Diagnostic bit S 2 Channel 2	Diagnostic bit S 1 Channel 1

Diagnostic bit S = ‘0’ no Error
Diagnostic bit S = ‘1’ overload, short circuit, or broken wire

Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
				controls DO 4 Channel 4	controls DO 3 Channel 3	controls DO 2 Channel 2	controls DO 1 Channel 1

17.2.2.7 8 Channel Digital Output Module

750-530, -536, -1515, -1516
753-530, -534

Table 96: 8 Channel Digital Output Module

Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
controls DO 8 Channel 8	controls DO 7 Channel 7	controls DO 6 Channel 6	controls DO 5 Channel 5	controls DO 4 Channel 4	controls DO 3 Channel 3	controls DO 2 Channel 2	controls DO 1 Channel 1

17.2.2.8 8 Channel Digital Output Modules with Diagnostics and Input Process Data

750-537

The digital output modules have a diagnostic bit for each output channel. When an output fault condition occurs (i.e., overload, short circuit, or broken wire), a diagnostic bit is set. The diagnostic data is mapped into the Input Process Image, while the output control bits are in the Output Process Image.

Table 97: 8 Channel Digital Output Modules with Diagnostics and Input Process Data

Input Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Diagnostic bit S 8 Channel 8	Diagnostic bit S 7 Channel 7	Diagnostic bit S 6 Channel 6	Diagnostic bit S 5 Channel 5	Diagnostic bit S 4 Channel 4	Diagnostic bit S 3 Channel 3	Diagnostic bit S 2 Channel 2	Diagnostic bit S 1 Channel 1

Diagnostic bit S = ‘0’ no Error
Diagnostic bit S = ‘1’ overload, short circuit, or broken wire

Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
controls DO 8 Channel 8	controls DO 7 Channel 7	controls DO 6 Channel 6	controls DO 5 Channel 5	controls DO 4 Channel 4	controls DO 3 Channel 3	controls DO 2 Channel 2	controls DO 1 Channel 1

17.2.2.9 16 Channel Digital Output Modules

750-1500, -1501, -1504, -1505

Table 98: 16 Channel Digital Output Modules

Output Process Image															
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
controls DO 16 Channel 16	controls DO 15 Channel 15	controls DO 14 Channel 14	controls DO 13 Channel 13	controls DO 12 Channel 12	controls DO 11 Channel 11	controls DO 10 Channel 10	controls DO 9 Channel 9	controls DO 8 Channel 8	controls DO 7 Channel 7	controls DO 6 Channel 6	controls DO 5 Channel 5	controls DO 4 Channel 4	controls DO 3 Channel 3	controls DO 2 Channel 2	controls DO 1 Channel 1

17.2.2.10 8 Channel Digital Input/Output Modules

750-1502, -1506

Table 99: 8 Channel Digital Input/Output Modules

Input Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data bit DI 8 Channel 8	Data bit DI 7 Channel 7	Data bit DI 6 Channel 6	Data bit DI 5 Channel 5	Data bit DI 4 Channel 4	Data bit DI 3 Channel 3	Data bit DI 2 Channel 2	Data bit DI 1 Channel 1

Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
controls DO 8 Channel 8	controls DO 7 Channel 7	controls DO 6 Channel 6	controls DO 5 Channel 5	controls DO 4 Channel 4	controls DO 3 Channel 3	controls DO 2 Channel 2	controls DO 1 Channel 1

17.2.3 Analog Input Modules

The hardware of an analog input module has 16 bits of measured analog data per channel and 8 bits of control/status.

However, the coupler/controller with MODBUS/TCP does not have access to the 8 control/status bits.

Therefore, the coupler/controller with MODBUS/TCP can only access the 16 bits of analog data per channel, which are grouped as words and mapped in Intel format in the Input Process Image.

When digital input modules are also present in the node, the analog input data is always mapped into the Input Process Image in front of the digital data.

Information

Information to the structure of the Control/Status byte
For detailed information about the structure of a particular module's control/status byte, please refer to that module's manual. Manuals for each module can be found on the Internet under: http://www.wago.com.

17.2.3.1 1 Channel Analog Input Modules

750-491, (and all variations)

Table 100: 1 Channel Analog Input Modules

Input Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	D1	D0	Measured Value U_D
1	D3	D2	Measured Value U_{ref}

17.2.3.2 2 Channel Analog Input Modules

750-452, -454, -456, -461, -462, -465, -466, -467, -469, -472, -474, -475, 476, -477, -478, -479, -480, -481, -483, -485, -492, (and all variations),
753-452, -454, -456, -461, -465, -466, -467, -469, -472, -474, -475, 476, -477, 478, -479, -483, -492, (and all variations)

Table 101: 2 Channel Analog Input Modules

Input Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	D1	D0	Measured Value Channel 1
1	D3	D2	Measured Value Channel 2

17.2.3.3 4 Channel Analog Input Modules

750-453, -455, -457, -459, -460, -468, (and all variations),
753-453, -455, -457, -459

Table 102: 4 Channel Analog Input Modules

Input Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	D1	D0	Measured Value Channel 1
1	D3	D2	Measured Value Channel 2
2	D5	D4	Measured Value Channel 3
3	D7	D6	Measured Value Channel 4

17.2.3.4 3-Phase Power Measurement Module

750-493

Table 103: 3-Phase Power Measurement Module

Input Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	-	S0	Status byte 0
1	D1	D0	Input data word 1
2	-	S1	Status byte 1
3	D3	D2	Input data word 2
4	-	S2	Status byte 2
5	D5	D4	Input data word 3

Output Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	-	C0	Control byte 0
1	D1	D0	Output data word 1
2	-	C1	Control byte 1
3	D3	D2	Output data word 2
4	-	C2	Control byte 2
5	D5	D4	Output data word 3

17.2.3.5 8 Channel Analog Input Modules

Table 104: 8 Channel Analog Input Modules

Input Process Image			
Offset	**Byte Destination**		**Description**
	High Byte	**Low Byte**	
0	D1	D0	Measured Value Channel 1
1	D3	D2	Measured Value Channel 2
2	D5	D4	Measured Value Channel 3
3	D7	D6	Measured Value Channel 4
4	D9	D8	Measured Value Channel 5
5	D11	D10	Measured Value Channel 6
6	D13	D12	Measured Value Channel 7
7	D15	D14	Measured Value Channel 8

17.2.4 Analog Output Modules

The hardware of an analog output module has 16 bits of measured analog data per channel and 8 bits of control/status. However, the coupler/controller with MODBUS/TCP does not have access to the 8 control/status bits. Therefore, the coupler/controller with MODBUS/TCP can only access the 16 bits of analog data per channel, which are grouped as words and mapped in Intel format in the Output Process Image.

When digital output modules are also present in the node, the analog output data is always mapped into the Output Process Image in front of the digital data.

Information

Information to the structure of the Control/Status byte
For detailed information about the structure of a particular module's control/status byte, please refer to that module's manual. Manuals for each module can be found on the Internet under: http://www.wago.com.

17.2.4.1 2 Channel Analog Output Modules

750-550, -552, -554, -556, -560, -562, 563, -585, (and all variations),
753-550, -552, -554, -556

Table 105: 2 Channel Analog Output Modules

Output Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	D1	D0	Output Value Channel 1
1	D3	D2	Output Value Channel 2

17.2.4.2 4 Channel Analog Output Modules

750-553, -555, -557, -559,
753-553, -555, -557, -559

Table 106: 4 Channel Analog Output Modules

Output Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	D1	D0	Output Value Channel 1
1	D3	D2	Output Value Channel 2
2	D5	D4	Output Value Channel 3
3	D7	D6	Output Value Channel 4

17.2.4.3 8 Channel Analog Output Modules

Table 107: 8 Channel Analog Output Modules

Output Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	D1	D0	Output Value Channel 1
1	D3	D2	Output Value Channel 2
2	D5	D4	Output Value Channel 3
3	D7	D6	Output Value Channel 4
4	D9	D8	Output Value Channel 5
5	D11	D10	Output Value Channel 6
6	D13	D12	Output Value Channel 7
7	D15	D14	Output Value Channel 8

17.2.5 Specialty Modules

WAGO has a host of Specialty I/O modules that perform various functions. With individual modules beside the data bytes also the control/status byte is mapped in the process image.

The control/status byte is required for the bidirectional data exchange of the module with the higher-ranking control system. The control byte is transmitted from the control system to the module and the status byte from the module to the control system.
This allows, for example, setting of a counter with the control byte or displaying of overshooting or undershooting of the range with the status byte.

The control/status byte always is in the process image in the Low byte.

Information

Information to the structure of the Control/Status byte
For detailed information about the structure of a particular module's control/status byte, please refer to that module's manual. Manuals for each module can be found on the Internet under: http://www.wago.com.

17.2.5.1 Counter Modules

750-404, (and all variations except of /000-005),
753-404, (and variation /000-003)

The above Counter Modules have a total of 5 bytes of user data in both the Input and Output Process Image (4 bytes of counter data and 1 byte of control/status). The counter value is supplied as 32 bits. The following tables illustrate the Input and Output Process Image, which has a total of 3 words mapped into each image. Word alignment is applied.

Table 108: Counter Modules 750-404, (and all variations except of /000-005), 753-404, (and variation /000-003)

Input Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	-	S	Status byte
1	D1	D0	Counter value
2	D3	D2	

Output Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	-	C	Control byte
1	D1	D0	Counter setting value
2	D3	D2	

750-404/000-005

The above Counter Modules have a total of 5 bytes of user data in both the Input and Output Process Image (4 bytes of counter data and 1 byte of control/ status). The two counter values are supplied as 16 bits. The following tables illustrate the Input and Output Process Image, which has a total of 3 words mapped into each image. Word alignment is applied.

Table 109: Counter Modules 750-404/000-005

Input Process Image			
Offset	**Byte Destination**		**Description**
	High Byte	**Low Byte**	
0	-	S	Status byte
1	D1	D0	Counter Value of Counter 1
2	D3	D2	Counter Value of Counter 2

Output Process Image			
Offset	**Byte Destination**		**Description**
	High Byte	**Low Byte**	
0	-	C	Control byte
1	D1	D0	Counter Setting Value of Counter 1
2	D3	D2	Counter Setting Value of Counter 2

750-638,
753-638

The above Counter Modules have a total of 6 bytes of user data in both the Input and Output Process Image (4 bytes of counter data and 2 bytes of control/status). The two counter values are supplied as 16 bits. The following tables illustrate the Input and Output Process Image, which has a total of 4 words mapped into each image. Word alignment is applied.

Table 110: Counter Modules 750-638, 753-638

Input Process Image			
Offset	**Byte Destination**		**Description**
	High Byte	**Low Byte**	
0	-	S0	Status byte von Counter 1
1	D1	D0	Counter Value von Counter 1
2	-	S1	Status byte von Counter 2
3	D3	D2	Counter Value von Counter 2

Output Process Image			
Offset	**Byte Destination**		**Description**
	High Byte	**Low Byte**	
0	-	C0	Control byte von Counter 1
1	D1	D0	Counter Setting Value von Counter 1
2	-	C1	Control byte von Counter 2
3	D3	D2	Counter Setting Value von Counter 2

17.2.5.2 Pulse Width Modules

750-511, (and all variations /xxx-xxx)

The above Pulse Width modules have a total of 6 bytes of user data in both the Input and Output Process Image (4 bytes of channel data and 2 bytes of control/ status). The two channel values are supplied as 16 bits. Each channel has its own control/status byte. The following table illustrates the Input and Output Process Image, which has a total of 4 words mapped into each image. Word alignment is applied.

Table 111: Pulse Width Modules 750-511, /xxx-xxx

Input and Output Process			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	-	C0/S0	Control/Status byte of Channel 1
1	D1	D0	Data Value of Channel 1
2	-	C1/S1	Control/Status byte of Channel 2
3	D3	D2	Data Value of Channel 2

17.2.5.3 Serial Interface Modules with alternative Data Format

750-650, (and the variations /000-002, -004, -006, -009, -010, -011, -012, -013),
750-651, (and the variations /000-001, -002, -003),
750-653, (and the variations /000-002, -007),
753-650, -653

Note

The process image of the / 003-000-variants depends on the parameterized operating mode!
With the freely parameterizable variations /003 000 of the serial interface modules, the desired operation mode can be set. Dependent on it, the process image of these modules is then the same, as from the appropriate variation.

The above Serial Interface Modules with alternative data format have a total of 4 bytes of user data in both the Input and Output Process Image (3 bytes of serial data and 1 byte of control/status). The following table illustrates the Input and Output Process Image, which have a total of 2 words mapped into each image. Word alignment is applied.

Table 112: Serial Interface Modules with alternative Data Format

Input and Output Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	D0	C/S	Data byte	Control/status byte
1	D2	D1	Data bytes	

17.2.5.4 Serial Interface Modules with Standard Data Format

750-650/000-001, -014, -015, -016
750-653/000-001, -006

The above Serial Interface Modules with Standard Data Format have a total of 6 bytes of user data in both the Input and Output Process Image (5 bytes of serial data and 1 byte of control/status). The following table illustrates the Input and Output Process Image, which have a total of 3 words mapped into each image. Word alignment is applied.

Table 113: Serial Interface Modules with Standard Data Format

Input and Output Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	D0	C/S	Data byte	Control/status byte
1	D2	D1	Data bytes	
2	D4	D3		

17.2.5.5 Data Exchange Module

750-654, (and the variation /000-001)

The Data Exchange modules have a total of 4 bytes of user data in both the Input and Output Process Image. The following tables illustrate the Input and Output Process Image, which has a total of 2 words mapped into each image.
Word alignment is applied.

Table 114: Data Exchange Module

Input and Output Process Image			
Offset	Byte Destination		Description
	High Byte	Low Byte	
0	D1	D0	Data bytes
1	D3	D2	

17.2.5.6 SSI Transmitter Interface Modules

750-630 (and all variations)

Note

The process image of the / 003-000-variants depends on the parameterized operating mode!
The operating mode of the configurable /003-000 I/O module versions can be set. Based on the operating mode, the process image of these I/O modules is then the same as that of the respective version.

The above SSI Transmitter Interface modules have a total of 4 bytes of user data in the Input Process Image, which has 2 words mapped into the image.
Word alignment is applied.

Table 115: SSI Transmitter Interface Modules

Input Process Image			
Offset	**Byte Destination**		**Description**
	High Byte	**Low Byte**	
0	D1	D0	Data bytes
1	D3	D2	

17.2.5.7 Incremental Encoder Interface Modules

750-631/000-004, -010, -011

The above Incremental Encoder Interface modules have 5 bytes of input data and 3 bytes of output data. The following tables illustrate the Input and Output Process Image, which have 4 words into each image. Word alignment is applied.

Table 116: Incremental Encoder Interface Modules 750-631/000-004, --010, -011

Input Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	-	S	not used	Status byte
1	D1	D0	Counter word	
2	-	-	not used	
3	D4	D3	Latch word	

Output Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	-	C	not used	Control byte
1	D1	D0	Counter setting word	
2	-	-	not used	
3	-	-	not used	

750-634

The above Incremental Encoder Interface module has 5 bytes of input data (6 bytes in cycle duration measurement mode) and 3 bytes of output data. The following tables illustrate the Input and Output Process Image, which has 4 words mapped into each image. Word alignment is applied.

Table 117: Incremental Encoder Interface Modules 750-634

Input Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	-	S	not used	Status byte
1	D1	D0	Counter word	
2	-	(D2) *)	not used	(Periodic time)
3	D4	D3	Latch word	

*) If cycle duration measurement mode is enabled in the control byte, the cycle duration is given as a 24-bit value that is stored in D2 together with D3/D4.

Output Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	-	C	not used	Control byte
1	D1	D0	Counter setting word	
2	-	-	not used	
3	-	-		

750-637

The above Incremental Encoder Interface Module has a total of 6 bytes of user data in both the Input and Output Process Image (4 bytes of encoder data and 2 bytes of control/status). The following table illustrates the Input and Output Process Image, which have 4 words mapped into each image. Word alignment is applied.

Table 118: Incremental Encoder Interface Modules 750-637

Input and Output Process Image			
Offset	**Byte Destination**		**Description**
	High Byte	**Low Byte**	
0	-	C0/S0	Control/Status byte of Channel 1
1	D1	D0	Data Value of Channel 1
2	-	C1/S1	Control/Status byte of Channel 2
3	D3	D2	Data Value of Channel 2

750-635,
753-635

The above Digital Pulse Interface module has a total of 4 bytes of user data in both the Input and Output Process Image (3 bytes of module data and 1 byte of control/status). The following table illustrates the Input and Output Process Image, which have 2 words mapped into each image. Word alignment is applied.

Table 119: Digital Pulse Interface Modules 750-635

Input and Output Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	D0	C0/S0	Data byte	Control/status byte
1	D2	D1	Data bytes	

17.2.5.8 DC-Drive Controller

750-636

The DC-Drive Controller maps 6 bytes into both the input and output process image. The data sent and received are stored in up to 4 input and output bytes (D0 ... D3). Two control bytes (C0, C1) and two status bytes (S0/S1) are used to control the I/O module and the drive.

In addition to the position data in the input process image (D0 … D3), it is possible to display extended status information (S2 … S5). Then the three control bytes (C1 … C3) and status bytes (S1 … S3) are used to control the data flow.

Bit 3 of control byte C1 (C1.3) is used to switch between the process data and the extended status bytes in the input process image (Extended Info_ON). Bit 3 of status byte S1 (S1.3) is used to acknowledge the switching process.

Table 120: DC-Drive Controller 750-636

Input Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	S1	S0	Status byte S1	Status byte S0
1	D1*) / S3**)	D0*) / S2**)	Actual position*) / Extended status byte S3**)	Actual position (LSB) / Extended status byte S2**)
2	D3*) / S5**)	D2*) / S4**)	Actual position (MSB) / Extended status byte S3**)	Actual position*) / Extended status byte S4**)

*) ExtendedInfo_ON = ‘0’.
**) ExtendedInfo_ON = ‘1’.

Output Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	C1	C0	Control byte C1	Control byte C0
1	D1	D0	Setpoint position	Setpoint position (LSB)
2	D3	D2	Setpoint position (MSB)	Setpoint position

17.2.5.9 Stepper Controller

750-670

The Stepper controller RS422 / 24 V / 20 mA 750-670 provides the fieldbus coupler 12 bytes input and output process image via 1 logical channel. The data to be sent and received are stored in up to 7 output bytes (D0 … D6) and 7 input bytes (D0 … D6), depending on the operating mode.

Output byte D0 and input byte D0 are reserved and have no function assigned.

One I/O module control and status byte (C0, S0) and 3 application control and status bytes (C1 ... C3, S1 ... S3) provide the control of the data flow.

Switching between the two process images is conducted through bit 5 in the control byte (C0 (C0.5). Activation of the mailbox is acknowledged by bit 5 of the status byte S0 (S0.5).

Table 121: Stepper Controller RS 422 / 24 V / 20 mA 750-670

Input Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	reserved	S0	reserved	Status byte S0
1	D1	D0	Process data*) / Mailbox**)	
2	D3	D2		
3	D5	D4		
4	S3	D6	Status byte S3	Process data*) / reserved**)
5	S1	S2	Status byte S1	Status byte S2

*) Cyclic process image (Mailbox disabled)
**) Mailbox process image (Mailbox activated)

Output Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	reserved	C0	reserved	Control byte C0
1	D1	D0	Process data*) / Mailbox**)	
2	D3	D2		
3	D5	D4		
4	C3	D6	Control byte C3	Process data*) / reserved**)
5	C1	C2	Control byte C1	Control byte C2

*) Cyclic process image (Mailbox disabled)
**) Mailbox process image (Mailbox activated)

17.2.5.10 RTC Module

750-640

The RTC Module has a total of 6 bytes of user data in both the Input and Output Process Image (4 bytes of module data and 1 byte of control/status and 1 byte ID for command). The following table illustrates the Input and Output Process Image, which have 3 words mapped into each image. Word alignment is applied.

Table 122: RTC Module 750-640

Input and Output Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	ID	C/S	Command byte	Control/status byte
1	D1	D0	Data bytes	
2	D3	D2		

17.2.5.11 DALI/DSI Master Module

750-641

The DALI/DSI Master module has a total of 6 bytes of user data in both the Input and Output Process Image (5 bytes of module data and 1 byte of control/status). The following tables illustrate the Input and Output Process Image, which have 3 words mapped into each image. Word alignment is applied.

Table 123: DALI/DSI Master module 750-641

Input Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	D0	S	DALI Response	Status byte
1	D2	D1	Message 3	DALI Address
2	D4	D3	Message 1	Message 2

Output Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	D0	C	DALI command, DSI dimming value	Control byte
1	D2	D1	Parameter 2	DALI Address
2	D4	D3	Command extension	Parameter 1

17.2.5.12 DALI Multi-Master Module

753-647

The DALI Multi-Master module occupies a total of 24 bytes in the input and output range of the process image.

The DALI Multi-Master module can be operated in "Easy" mode (default) and "Full" mode. "Easy" mode is used to transmit simply binary signals for lighting control. Configuration or programming via DALI master module is unnecessary in "Easy" mode.

Changes to individual bits of the process image are converted directly into DALI commands for a pre-configured DALI network. 22 bytes of the 24-byte process image can be used directly for switching of electronic ballasts (ECG), groups or scenes in "Easy" mode. Switching commands are transmitted via DALI and group addresses, where each DALI and each group address is represented by a 2-bit pair.

The structure of the process data is described in detail in the following tables.

Table 124: Overview of input process image in the "Easy" mode

Input process image				
Offset	**Byte designation**		**Note**	
	High byte	**Low byte**		
0	-	S	res.	Status, activate broadcast Bit 0: 1-/2-button mode Bit 2: Broadcast status ON/OFF Bit 1,3-7: -
1	DA4…DA7	DA0…DA3	Bitpaar für DALI-Adresse DA0: Bit 1: Bit set = ON Bit not set = OFF Bit 2: Bit set = Error Bit not set = No error Bit pairs DA1 … DA63 similar to DA0.	
2	DA12…DA15	DA8…DA11		
3	DA20…DA23	DA16…DA19		
4	DA28…DA31	DA24…DA27		
5	DA36…DA39	DA32…DA35		
6	DA44…DA47	DA40…DA43		
7	DA52…DA55	DA48…DA51		
8	DA60…DA63	DA56…DA59		
9	GA4…GA7	GA0…GA3	Bit pair for DALI group address GA0: Bit 1: Bit set = ON Bit not set = OFF Bit 2: Bit set = Error Bit not set = No error Bit pairs GA1 … GA15 similar to GA0.	
10	GA12…GA15	GA8…GA11		
11	-	-	Not in use	

DA = DALI address
GA = Group address

Table 125: Overview of the output process image in the "Easy" mode“

Output process image				
Offset	**Byte designation**		**Note**	
	High byte	**Low byte**		
0	-	S	res.	Broadcast ON/OFF and activate: Bit 0: Broadcast ON Bit 1: Broadcast OFF Bit 2: Broadcast ON/OFF/dimming Bit 3: Broadcast short ON/OFF Bit 4 … 7: reserved
1	DA4…DA7	DA0…DA3	Bit pair for DALI address DA0: Bit 1: short: DA switch ON long: dimming, brighter Bit 2: short: DA switch OFF long: dimming, darker Bit pairs DA1 … DA63 similar to DA0.	
2	DA12…DA15	DA8…DA11		
3	DA20…DA23	DA16…DA19		
4	DA28…DA31	DA24…DA27		
5	DA36…DA39	DA32…DA35		
6	DA44…DA47	DA40…DA43		
7	DA52…DA55	DA48…DA51		
8	DA60…DA63	DA56…DA59		
9	GA4…GA7	GA0…GA3	Bitpaar für DALI-Gruppenadresse GA0: Bit 1: short: GA switch ON long: dimming, brighter Bit 2: short: GA switch OFF long: dimming, darker Bit pairs GA1 … GA15 similar to GA0.	
10	GA12…GA15	GA8…GA11		
11	Bit 8…15	Bit 0…7	Switch scene 0…15	

DA = DALI address
GA = Group address

17.2.5.13 LON® FTT Module

753-648

The process image of the LON® FTT module consists of a control/status byte and 23 bytes of bidirectional communication data that is processed by the WAGO-I/O-*PRO* function block "LON_01.lib". This function block is essential for the function of the LON® FTT module and provides a user interface on the control side.

17.2.5.14 EnOcean Radio Receiver

750-642

The EnOcean radio receiver has a total of 4 bytes of user data in both the Input and Output Process Image (3 bytes of module data and 1 byte of control/status). The following tables illustrate the Input and Output Process Image, which have 2 words mapped into each image. Word alignment is applied.

Table 126: EnOcean Radio Receiver 750-642

Input Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	D0	S	Data byte	Status byte
1	D2	D1	Data bytes	

Output Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	-	C	not used	Control byte
1	-	-	not used	

17.2.5.15 MP Bus Master Module

750-643

The MP Bus Master Module has a total of 8 bytes of user data in both the Input and Output Process Image (6 bytes of module data and 2 bytes of control/status). The following table illustrates the Input and Output Process Image, which have 4 words mapped into each image. Word alignment is applied.

Table 127: MP Bus Master Module 750-643

Input and Output Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	C1/S1	C0/S0	extended Control/ Status byte	Control/status byte
1	D1	D0	Data bytes	
2	D3	D2		
3	D5	D4		

17.2.5.16 *Bluetooth*® RF-Transceiver

750-644

The size of the process image for the *Bluetooth*® module can be adjusted to 12, 24 or 48 bytes.
It consists of a control byte (input) or status byte (output); an empty byte; an overlay able mailbox with a size of 6, 12 or 18 bytes (mode 2); and the *Bluetooth*® process data with a size of 4 to 46 bytes.
Thus, each *Bluetooth*® module uses between 12 and 48 bytes in the process image. The sizes of the input and output process images are always the same.

The first byte contains the control/status byte; the second contains an empty byte.

Process data attach to this directly when the mailbox is hidden. When the mailbox is visible, the first 6, 12 or 18 bytes of process data are overlaid by the mailbox data, depending on their size. Bytes in the area behind the optionally visible mailbox contain basic process data. The internal structure of the *Bluetooth*® process data can be found in the documentation for the *Bluetooth*® 750-644 RF Transceiver.

The mailbox and the process image sizes are set with the startup tool WAGO-I/O-*CHECK*.

Table 128: Bluetooth® RF-Transceiver 750-644

Input and Output Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	-	C0/S0	not used	Control/status byte
1	D1	D0	Mailbox (0, 3, 6 or 9 words) and Process data (2-23 words)	
2	D3	D2		
3	D5	D4		
...	...	...		
max. 23	D45	D44		

17.2.5.17 Vibration Velocity/Bearing Condition Monitoring VIB I/O

750-645

The Vibration Velocity/Bearing Condition Monitoring VIB I/O has a total of 12 bytes of user data in both the Input and Output Process Image (8 bytes of module data and 4 bytes of control/status). The following table illustrates the Input and Output Process Image, which have 8 words mapped into each image.
Word alignment is applied.

Table 129: Vibration Velocity/Bearing Condition Monitoring VIB I/O 750-645

Input and Output Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	-	C0/S0	not used	Control/status byte (log. Channel 1, Sensor input 1)
1	D1	D0	Data bytes (log. Channel 1, Sensor input 1)	
2	-	C1/S1	not used	Control/status byte (log. Channel 2, Sensor input 2)
3	D3	D2	Data bytes (log. Channel 2, Sensor input 2)	
4	-	C2/S2	not used	Control/status byte (log. Channel 3, Sensor input 1)
5	D5	D4	Data bytes (log. Channel 3, Sensor input 3)	
6	-	C3/S3	not used	Control/status byte (log. Channel 4, Sensor input 2)
7	D7	D6	Data bytes (log. Channel 4, Sensor input 2)	

17.2.5.18 KNX/EIB/TP1 Module

753-646

The KNX/TP1 module appears in router and device mode with a total of 24-byte user data within the input and output area of the process image, 20 data bytes and 2 control/status bytes. Even though the additional bytes S1 or C1 are transferred as data bytes, they are used as extended status and control bytes. The opcode is used for the read/write command of data and the triggering of specific functions of the KNX/EIB/TP1 module. Word-alignment is used to assign 12 words in the process image. Access to the process image is not possible in router mode. Telegrams can only be tunneled.

In device mode, access to the KNX data can only be performed via special function blocks of the IEC application. Configuration using the ETS engineering tool software is required for KNX.

Table 130: KNX/EIB/TP1 Module 753-646

Input Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	-	S0	not used	Status byte
1	S1	OP	extended Status byte	Opcode
2	D1	D0	Data byte 1	Data byte 0
3	D3	D2	Data byte 3	Data byte 2
4	D5	D4	Data byte 5	Data byte 4
5	D7	D6	Data byte 7	Data byte 6
6	D9	D8	Data byte 9	Data byte 8
7	D11	D10	Data byte 11	Data byte 10
8	D13	D12	Data byte 13	Data byte 12
9	D15	D14	Data byte 15	Data byte 14
10	D17	D16	Data byte 17	Data byte 16
11	D19	D18	Data byte 19	Data byte 18

Output Process Image				
Offset	Byte Destination		Description	
	High Byte	Low Byte		
0	-	C0	not used	Control byte
1	C1	OP	extended Control byte	Opcode
2	D1	D0	Data byte 1	Data byte 0
3	D3	D2	Data byte 3	Data byte 2
4	D5	D4	Data byte 5	Data byte 4
5	D7	D6	Data byte 7	Data byte 6
6	D9	D8	Data byte 9	Data byte 8
7	D11	D10	Data byte 11	Data byte 10
8	D13	D12	Data byte 13	Data byte 12
9	D15	D14	Data byte 15	Data byte 14
10	D17	D16	Data byte 17	Data byte 16
11	D19	D18	Data byte 19	Data byte 18

17.2.5.19 AS-interface Master Module

750-655

The length of the process image of the AS-interface master module can be set to fixed sizes of 12, 20, 24, 32, 40 or 48 bytes.
It consists of a control or status byte, a mailbox with a size of 0, 6, 10, 12 or 18 bytes and the AS-interface process data, which can range from 0 to 32 bytes.

The AS-interface master module has a total of 6 to maximally 24 words data in both the Input and Output Process Image. Word alignment is applied.

The first Input and output word, which is assigned to an AS-interface master module, contains the status / control byte and one empty byte.
Subsequently the mailbox data are mapped, when the mailbox is permanently superimposed (Mode 1).

In the operating mode with suppressible mailbox (Mode 2), the mailbox and the cyclical process data are mapped next.
The following words contain the remaining process dat.

The mailbox and the process image sizes are set with the startup tool WAGO-I/O-*CHECK*.

Table 131: AS-interface Master module 750-655

Input and Output Process Image				
Offset	**Byte Destination**		**Description**	
	High Byte	**Low Byte**		
0	-	C0/S0	not used	Control/status byte
1	D1	D0	Mailbox (0, 3, 5, 6 or 9 words)/ Process data (0-16 words)	
2	D3	D2		
3	D5	D4		
...	...	...		
max. 23	D45	D44		

17.2.6 System Modules

17.2.6.1 System Modules with Diagnostics

750-610, -611

The modules provide 2 bits of diagnostics in the Input Process Image for monitoring of the internal power supply.

Table 132: System Modules with Diagnostics 750-610, -611

Input Process Image							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
						Diagnostic bit S 2 Fuse	Diagnostic bit S 1 Fuse

17.2.6.2 Binary Space Module

750-622

The Binary Space Modules behave alternatively like 2 channel digital input modules or output modules and seize depending upon the selected settings 1, 2, 3 or 4 bits per channel. According to this, 2, 4, 6 or 8 bits are occupied then either in the process input or the process output image.

Table 133: Binary Space Module 750-622 (with behavior like 2 channel digital input)

Input and Output Process Image							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
(Data bit DI 8)	(Data bit DI 7)	(Data bit DI 6)	(Data bit DI 5)	(Data bit DI 4)	(Data bit DI 3)	Data bit DI 2	Data bit DI 1

17.3 Mailbox Modules

The I/O-IPC currently supports I/O modules that work according to the mailbox principle such as, e.g., the 750-655 AS-interface master modules (process data range max. 500 byte) or the 750-670 stepper controller modules.

18 Appendix

18.1 WAGOConfigToolLIB.lib

With the function block of the CODESYS "WAGOConfigToolLIB.lib" library,, configure and parameterize the I/O-IPC like with the WBM and "IPC Configuration Tool". You need the calls from the next Section. These have no effect on the run time of the tasks because the functions are called asynchronsly.

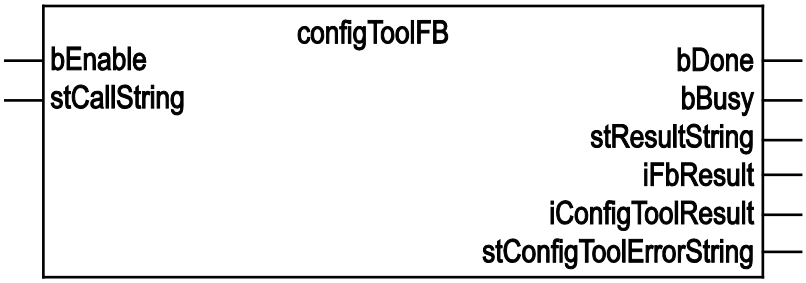


Figure 130: Graphical representation of the "ConfigTool" function block

Table 134: "ConfigTool" function block

Parameter	Name	Data Type	Description
Input	bEnable	BOOL	The function block starts processing when the input registers an ascending edge.
	stCallString	STRING (150)	Enter a call like in the Linux consol. An overview of the calls is available in section "Calls for the WAGOConfigToolLIB.lib Library" .
Output	bDone	BOOL	Indicates whether the function block is executed fully or after canceled after reporting an error code.
	bBusy	BOOL	The function block processes a call.
	stResultString	STRING (80)	Return value displayed on the Linux console.
Output	iFbResult	INT	The return values have the following meaning: 0: No error 1: Invalid input parameter, e.g. an empty string at stCallString 2: Error at implementation of the function block 3: Unknown Configtool 4: The Configtool result string is too large for the function block return parameter stResultString
	iConfigToolResult	INT	The value corresponds directly to the Configtool result parameter.
	stConfigToolError String	STRING (150)	This parameter displays the error description if the call "wConfigToolResult" returns a value ≠ 0.

Table 135: Function STRING_TO_IP

Parameter	Name	Data Type	Description
Input	stIpAddress	STRING (15)	String with the IP address in the form xxx.xxx.xxx.xxx
In-/Output	IpAddress	ARRAY [0..3] OF BYTE	Arry with the values of the individual bytes of the IP address.
Return Value	-	BYTE	Status feedback of the error code. 0 = no error 255 = invalid parameter

Table 136: Function IP_TO_STRING

Parameter	Name	Data Type	Description
Input	ipAddress	ARRAY [0..3] OF BYTE	Array with the values of the individual bytes of the IP address.
In-/Output	stIpAddress	STRING (15)	String with the IP address in the form xxx.xxx.xxx.xxx
Return Value	-	BYTE	Status feedback of the error code. Always 0 = no error

18.1.1 Calls for the "WAGOConfigToolLIB.lib" Library

The following table shows calls that allow you to configure and parameterize the I/O-IPC from the PLC program or Linux via the "ConfigTool" function block. In addition to WBM and the "IPC Configuration Tool", this is another variant to configure the I/O-IPC for operational requirements.

The configuration directory under Linux is: `/etc/config-tools/`

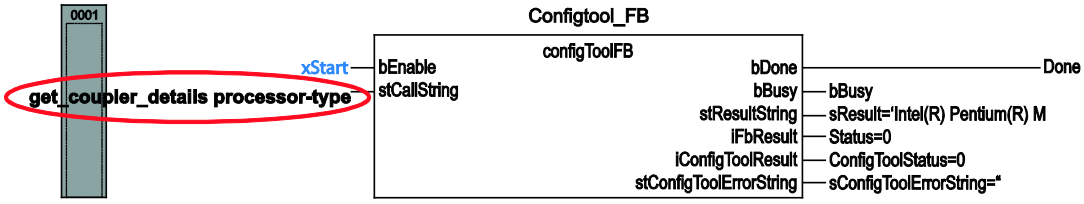


Figure 131: Graphical representation of the "ConfigTool" function block

Table 137: Description of the configuration scripts for "Information"

Parameter	Status	Call	Output/Input	Validity
Information				
Coupler Details: Identified various information of the I/O-IPC				
Order Number	read	get_coupler_details order-number	Item number of the I/O-IPC.	immediate
Processor Type	read	get_coupler_details processor-type	Processor type of the I/O-IPC	
Fieldbus Type	read	get_coupler_details fieldbus-type	Fielbus type of the I/O-IPC	
Firmware Revision	read	get_coupler_details firmware-revision	Firmware version of the I/O-IPC	
Licence Information	read	get_coupler_details license-information	CODESYS license information	
Kbus FW Revision	read	get_coupler_details kbus-fw-revision	Firmware version of the I/O module controller	
CODESYS Webserver Version	read	get_coupler_details codesys-webserver-version	CODESYS Webserver version	
Network Details Eth0: Identifies the parameters of the ETHERNET interfaces currently used				
State	read	get_actual_eth_config eth0 state	Status of the interface: **enabled** **disabled**	immediate
Mac Address	read	get_actual_eth_config eth0 mac-address	Display of the MAC address	
IP Address	read	get_actual_eth_config eth0 ip-address	Display of the current IP address	
Subnet Mask	read	get_actual_eth_config eth0 subnet-mask	Display of the current subnet mask	
Network Details Eth1				
See "Network Details Eth1". When calling, replace eth0 with eth1				

Table 138: Description of the configuration scripts for "CODESYS"

Parameter	Status	Call	Output/Input	Validity
CODESYS				
Project Details				
Date	read	get_rts_info project date	Display of the project information specified in CODESYS (Menu > Project > Project Information)	immediate
Title	read	get_rts_info project title		
Version	read	get_rts_info project version		
Author	read	get_rts_info project author		
Description	read	get_rts_info project description		
CODESYS State				
State	read	get_rts_info state	Display of the CODESYS status (RUN or STOP)	immediate

Table 139: Description of the configuration scripts for "TCP/IP"

Parameter	Status	Call	Output/Input	Validity
TCP/IP				
Common Configuration Data				
Hostname	read	get_coupler_details hostname	Display of the host name	immediate
	write	change_hostname hostname=<String>	Changing the host name. Input a host name for <String>.	after restart
Default Gateway				
"Default Gateway" Interface	read	get_coupler_details default-gateway	Display of the standard gateway set	immediate
	write	config_default_gateway interface=<value>	Select the interface you want to use as the standard gateway. **eth0** **eth1** **none** (no standard gateway selected)	
Default Gateway Value	read	get_eth_config eth0 default-gateway get_eth_config eth1 default-gateway	Display of the standard gateway address. Leads both to the same result because the value is always written for both interfaces at the same time.	immediate
	write	config_default_gateway default-gateway-value=<value>	Enter the address of the standard gateway here. The <value> is an IP address in the format **number. number. number. number.**	after restart
DNS-Server				
Domain Name	read	get_coupler_details domain-name	Display of the domain name.	immediate
	write	edit_dns_server domain-name=<String>	Changing the domain name. Input a domain name for <String>.	

Table 139: Description of the configuration scripts for "TCP/IP"

Parameter	Status	Call	Output/Input	Validity
DNS Server 1	read	get_dns_server 1	DNS server address with the consecutive number 1.	immediate
	write/ change	edit_dns_server dns-server-nr=1 change=change dns-server-name=<value>	Set the address of the DNS server with the consecutive number 1. The <value> is an IP address in the format **number.number.number. number.**	
	write/ delete	edit_dns_server dns-server-nr=1 delete=delete	Delete the DNS server with the consecutive number 1.	
DNS Server 2-n	See "DNS Server" 1. When calling, adjust the server number (count up)			immediate
Add DNS Server	write	edit_dns_server add=add dns-server-name=<value>	Add additional DNS addresses here. The <value> is an IP address in the format **number.number.number. number.**	immediate
TCP/IP Configuration Eth0				
State				
Device ID Eth0	read	get_eth_config eth0 device-id	Device ID of the ETHERNET interface: **X8** **X9**	immediate
State Eth0	read	get_eth_config eth0 state	Status of the ETHERNET interface Eth0: **enabled** **disabled**	
	write	config_interfaces interface=eth0 config-type=static state=enabled	Switch off interface: **disabled** When switching on, always specify the **config type**.	

Table 139: Description of the configuration scripts for "TCP/IP"

Parameter	Status	Call	Output/Input	Validity
Type of IP Address Configuration Eth0	read	get_eth_config eth0 config-type	Way be which the interface receives its IP address: **static** (static assignment), **dhcp** (per DHC) or **bootp** (per BootP)	immediate
	write	config_interfaces interface=eth0 config-type=<value> state=enabled	Switch on method by which the interface receives its IP address. Entries for <value> are: **static** (static assignment), **dhcp** (per DHC) or **bootp** (per BootP)	
IP Address Eth0	read	get_eth_config eth0 ip-address	Address set for using a static IP address (static IP).	
	write	config_interfaces interface=eth0 ip-address=<value>	Change IP address for static IP. The <value> must contain an IP address in the format **number.number.number.number**.	
Subnet Mask Eth0	read	get_eth_config eth0 subnet-mask	Subnet mask set for using a static IP address (static IP).	
	write	config_interfaces interface=eth0 subnet-mask=<value>	Change subnet mask for static IP. The <value> must contain an IP address in the format **number.number.number.number**.	
TCP/IP Configuration Eth1				
See "TCP/IP Configuration Eth0". When calling, replace eth0 with eth1.				

Table 140: Description of the configuration scripts for "ETHERNET"

Parameter	Status	Call	Output/Input	Validity
ETHERNET				
Transmission Mode Eth0				
Autonegotiation	read	get_eth_config eth0 autoneg	Query the status of the Autonegotiation function: **on** **off**	immediate
	write	config_interfaces eth0 autoneg=on	Switch on Autonegotiation function: **on**	
		config_interfaces interface=eth0 autoneg=off speed-duplex=<value>	Switch Autonegotiation function off: **off** **Note:** The speed and duplex value should be entered when switching the Autonegotiation off. Entries for <value> are: **10-half** **10-full** **100-half** **100-full**	
Speed and Duplex Settings	read	get_eth_config eth0 speed	Display the ETHERNET speed.	
	read	get_eth_config eth0 duplex	Display of the duplex value.	
	write	config_interfaces interface=eth0 autoneg=off speeed-duplex=<value>	Change the ETHERNET speed and the Duplex settings. Entries for <value> are: **10-half** **10-full** **100-half** **100-full**	
Transmission Mode Eth1				
See "Transmission Mode Eth0". When calling, replace eth0 with eth1.				

Table 141: Description of the configuration scripts for "NTP"

Parameter	Status	Call	Output/Input	Validity
NTP				
Configuration Data				
State	read	get_ntp_config state	Entries for <value> are: **enabled** **disabled**	immediate
	write	config_sntp state=<value>	Entries for <value> are: **enabled** **disabled**	
Port	read	get_ntp_config port	Port number of the NTP server	
	write	config_sntp port=<value>	Specify the port number for <value>.	
Time Server	read	get_ntp_config time-server	Query the IP address of the Time server.	
	write	config_sntp time-server=<value>	Enter the IP address of the Time server. The <value> can contain an IP address in the format **number.number.number.number** or a domain name as a string.	
Update Time (seconds)	read	get_ntp_config update-time	Query of the polling cycle of the Time server.	
	write	config_sntp update-time=<value>	Specify the time-server's query cycle (in s) for <value>.	

Table 142: Description of the configuration scripts for "Clock"

Parameter	Status	Call	Output/Input	Validity
Clock				
Time and Date				
Date on Device, local	read	get_clock_data date-local	Local time and date	immediate
	write	config_clock type=local date=<date>	Change date. The format for <date> is: **DD.MM.YYYY**	
Time on Device, UTC	read	get_clock_data time-utc	Time/UTC	
	write	config_clock type=utc time=<Time>	Change time, based on UTC time. The format for <time> is: **hh:mm:ss xx**	
Time on Device, local	read	get_clock_data time-local	Time/local time	
	write	config_clock type=local time=<Time>	Change time, based on local time. The format for <time> is: **hh:mm:ss xx**	
12-Hour Format	read	get_clock_data display-mode	Display format of the time in 12- or 24-hour format: 12-hour format or 24-hour format	
	write	config_clock _ display_mode display-mode=<value>	Set the display format of the time. Entries for <value> are: **12-hour-format** **24-hour-format**	
Time zone				
TZ String	read	get_clock_data tz-string	Currently set time zone – original TZ string as stored in the operating system.	after restart
	write	config_timezone tz-string=<String>	Change TZ string directly. Example of <string>: **CET-1CEST, M3.5.0/2,M10.5.0/3**	

Table 143: Description of the configuration scripts for "HMI Settings"

Parameter	Status	Call	Output/Input	Validity
HMI Settings				
Screensaver				
Display Status	read	get_touchscreen_config display-state	**on** – display is turned on (screensaver is off). **off** – display is switched off (screensaver is on).	immediate
	write	change_screen_state display-state=<value>	<value>=**on** – switches the display on immediately. <value>=**off** – turns the display off immediately. Switching the display on/off takes place independently of the screensaver activation. Whether the displace is turned off again after a wait time, or is turned on again after contact at the touch screen, depends on whether the screensaver function is activated (see next point).	

Table 143: Description of the configuration scripts for "HMI Settings"

Parameter	Status	Call	Output/Input	Validity
Screensaver Status	read	get_rts3scfg_value SCREENSAVER Enabled	**enabled** – the screensaver state is actively switched on, i.e. after the configured wait time, the display is automatically turned off, and it is turned on again via user input at the touch screen or keyboard. **disabled** – the screensaver state is not activated. The display is thus not switched off automatically after the wait time (or turned on after a keystroke); instead it remains in the state that the user explicitly set.	immediate
	write	change_rts_config area=SCREENSAVER state=<value>	<value>=**enabled** – the screensaver state is not activated, i.e. after the configured wait time, the display is automatically turned off. <value>=**disabled** – the screensaver state is turned off, i.e. the screensaver does not influence the display. However, the display can be turned on or off by the user.	
Screensaver Wait Time	read	get_rts3scfg_value SCREENSAVER WaitTime	Returns the time value in seconds, after which the display is turned off when the screensaver is activated.	
	write	change_rts_config area=SCREENSAVER WaitTime=<value>	Changes the time value, after which the display is turned off when the screensaver is activated. <value> = **integer**, time in seconds	

Table 143: Description of the configuration scripts for "HMI Settings"

Parameter	Status	Call	Output/Input	Validity
Cleanmode Status	read	get_touchscreen_config cleanmode-state	**on** – Cleanmode is currently switched on, i.e. contacts on the touch screen will be ignored for the duration of the set timeout time. A note image is displayed on the screen. **off** – Cleanmode is currently not switched on, contacts on the touch screen will be processed.	immediate
	write	change_screen_state cleanmode-state=<value>	<value>=**on**: – cleanmode is activated for the time duration entered in timeout. <value>=**off**: – if cleanmode is active at this time, it is switched off again without waiting for the runout of the timeout time.	
Cleanmode Timeout	read	get_rts3scfg_value CLEANMODE Timeout	Returns the set timeout value of the cleanmode in seconds, i.e., when cleanmode is activated, user inputs at the touch screen will be ignored for this time duration.	
	write	change_rts_config area=CLEANMODE Timeout=<value>	<value>=integer, time value in seconds. Changes the timeout value of cleanmode.	
VGA-Configuration				
Video Mode	read	show_video_mode string	Display of the configured video mode (resolution and color depth).	immediate
	write	change_video_mode video-string=<value>	Change video mode: Possible entries for <value> are e.g.: **640x480+256+color** **800x600+16+bit** **1024x768+32+bit** (according to supported resolution and color depth)	after restart
Show Mouse Pointer	read	get_touchscreen_config mouse-pointer	Mouse pointer setting visible: **yes** **no**	immediate
	write	config_mousepointer show-mouse-pointer=<value>	Change mouse pointer setting. Possible entries for <value>: **yes** **no**	after restart

Table 143: Description of the configuration scripts for "HMI Settings"

Parameter	Status	Call	Output/Input	Validity
Touchscreen Configuration				
Device Name	read	get_touchscreen_ config device-name	Read device name	immediate
Driver Name	read	get_touchscreen_ config driver-name	Read driver name	
Execute Calibration of Touchscreen at Next Start	read	get_touchscreen_config calibrate-touchscreen-flag	Returns text **checked** if calibration is set.	immediate
	write	config_touchscreen calibrate-touchscreen=<value>	Calibration of the touchscreen when the I/O-IPC is started next. Entries for <value> are: **yes** **no**	after restart
Keyboard Layout				
Keyboard Layout	read	get_coupler_details keyboard-layout	Keyboard layout: **german** **english**	immediate
	write	change_keyboard_layout keyboard-layout=<value>	Entries for <value> are: **German** **English**	

Table 144: Description of the configuration scripts for "Administration"

Parameter	Status	Call	Output/Input	Validity
Administration				
Configuration of Serial Interface				
Configuration of Serial Interface	read	get_coupler_details RS232-owner	User of the serial interface (RS-232). Possible values are: **CODESYS** **IO-Check** **MODBUS** **Linux** **None**	immediate
	write	Config_RS232 owner=<value>	RS-232 user. Entries for <value> are: **CODESYS** **IO-Check** **MODBUS** **Linux** **None**	

Table 144: Description of the configuration scripts for "Administration"

Parameter	Status	Call	Output/Input	Validity
File system Check				
Filesystem Check	write	filesystem_check device=<value>	Check the file system of the specified device names or for all devices. Entries for <value> are: **hda1** **hda2** **hda3** **hda4** **hdb1** **hdb2** **hdb3** **hdb4** **all**	immediate
Start Backup System				
Start Backup System	write	start_backup_system	Switch the boot loader, so that the other older version of the system firmware is started when rebooting next.	after restart
Reboot IPC				
-	write	start_reboot	Reboot the I/O-IPC.	immediate

Table 145: Description of the configuration scripts for "Package Server"

Parameter	Status	Call	Output/Input	Validity
Firmware Update				
Medium of the Active Partition	read	get_filesystem_data active_partition_medium	Outputs the media of the active partition cf-card, internal-flash, usb1, usb2, etc.)	immediate
Creating Firmware Backup	write	firmware_backup package-settings=<value1> package-codesys=<value2> package-system=<value3> device-medium=<value4> auto-update-feature=<value5>	Creates a backup of the selected package on the media specified. Parameter: <value1> = **1**, if Settings package should be selected. <value2> = **1**, if System package should be selected. <value3> = **1**, if System package should be selected. <value4> = Target media to save the backup (**cf-card, internal-flash, usb1, usb2**). <value5> = **1**, if the Auto Update feature should be enabled. Parameters that should not be set (**1**) can be set to **0** or completely omitted	

Table 146: Description of the configuration scripts for "Mass Storage"

Parameter	Status	Call	Output/Input	Validity
Mass Storage				
Bootflag	read	get_device_data bootflag <value>	Returns the value of the bootflag of a device (**0** = device is not bootable or **1** = device is bootable) Specify the device names for <value>, e.g. **hda**, **hdb**,	immediate
	write	change_bootflag device=<value1> bootflag=<value2>	Set or reset the bootflag of a device. <value1> = Device name (**hda, hdb**, etc.) <value2> = **0** (reset bootflag), **1** (set bootflag)	

Table 147: Description of the configuration scripts for "Port"

Parameter	Status	Call	Output/Input	Validity
Port				
Telnet				
Telnet Port	read	get_port_state telnet	Read the status of the Telnet server: **enabled** **disabled**	immediate
	write	config_port port=telnet state=<value>	Entries for <value> are: **enabled** **disabled**	
CODESYS-Webserver				
CODESYS Webserver Port	read	get_port_state codesys-webserver	Read status of the CODESYS web server. Entries for <value> are: **enabled** **disabled**	immediate
	write	config_port port=codesys-webserver state=<value>	Enable/disable the CODESYS web server. Entries for <value> are: **enabled** **disabled**	
FTP				
FTP Port	read	get_port_state ftp	Read the status of the FTP server. Entries for <value> are: **enabled** **disabled**	immediate
	write	config_port port=ftp state=<value>	Entries for <value> are: **enabled** **disabled**	
CODESYS				
CODESYS Port	read	get_rts3scfg_value PLC DisableTcpIp Programming	Query the status of the value for "DisableTcpIpProgramming" in the CODESYS configuration: **YES**: CODESYS port is not used. **NO**: CODESYS port is used.	immediate
	write	change_rts_config area=PLC disable-tcpip=<value>	Entries for <value> are: **YES**: CODESYS port is not used. **NO**: CODESYS port is used.	
CODESYS-Port Number	read	get_rts3scfg_value PLC TcpIpPort	Value set in the CODESYS configuration for the TCP/IP port.	
	write	change_rts_config area=PLC TcpIpPort=<value>	Change the value of the TCP/IP port number. Specify the TCP/IP port number for <value>.	

Table 148: Description of the configuration scripts for "MODBUS"

Parameter	Status	Call	Output/Input	Validity
MODBUS				
MODBUS/UDP				
MODBUS/UDP Status	read	get_rts3scfg_value MODBUS_UDP state	Status of MODBUS/UDP: **enabled** **disabled**	immediate
	write	change_rts_config area=MODBUS_UDP state=<value>	Enable/disable the MODBUS/UDP server. Entries for <value> are: **enabled** **disabled**	
MODBUS/TCP				
MODBUS/TCP Status	read	get_rts3scfg_value MODBUS_TCP state	Status of MODBUS/TCP: **enabled** **disabled**	immediate
	write	change_rts_config area=MODBUS_TCP state=<value>	Enable/disable the MODBUS/TCP server. Entries for <value> are: **enabled** **disabled**	
Timeout (msec)	read	get_rts3scfg_value MODBUS_TCP TCPTimeout	Timeout value for MODBUS/TCP	
	write	change_rts_config area=MODBUS_TCP timeout=<value>	Here, set the time period (timeout) for the MODBUS/TCP connection (ms), after which the connection is automatically ended during a break in communication.	
MODBUS/RTU				
State	read	get_rts3scfg_value MODBUS_RTU state	Status of MODBUS/RTU. Entries for <value> are: **enabled** **disabled**	immediate
	write	change_rts_config area=MODBUS_RTU timeout=<value>	Enable/disable the MODBUS/RTU server. Entries for <value> are: **enabled** **disabled**	
Node ID	read	get_rts3scfg_value MODBUS_RTU Node_ID	Node ID for MODBUS/RTU	
	write	change_rts_config area=MODBUS_RTU node-id=<value>	Node ID (number) for MODBUS/RTU Specify the node-ID for <value>	
Timeout (msec)	read	get_rts3scfg_value MODBUS_RTU Timeout	Timeout value for MODBUS/RTU	
	write	change_rts_config area=MODBUS_RTU Timeout=<value>	Change timeout value (ms) for MODBUS/RTU. Specify the timeout value in ms for <value>.	

Table 148: Description of the configuration scripts for "MODBUS"

Parameter	Status	Call	Output/Input	Validity
Baudrate	read	get_rts3scfg_value MODBUS_RTU Baud	Baud rate for MODBUS/RTU	immediate
	write	change_rts_config area=MODBUS_RTU Baud=<value>	Change the baud rate for MODBUS/RTU. Entries for <value> are: **2400** **4800** **9600** **19200** **38400** **57600** **115200**	
Databit	read	get_rts3scfg_value MODBUS_RTU Data_Bits	Data bit number for MODBUS/RTU	
Parity	read	get_rts3scfg_value MODBUS_RTU Parity	Parity value for MODBUS/RTU	
	write	hange_rts_config area=MODBUS_RTU Parity=<value>	Change the parity for MODBUS/RTU. Entries for <value> are: **None** **Odd** **Even**	
Stop Bits	read	get_rts3scfg_value MODBUS_RTU Stop_Bits	Stop bits for MODBUS/RTU	
	write	change_rts_config area=MODBUS_RTU Stop_Bits=<value>	Set the number of stop bits for MODBUS/RTU. Entries for <value> are: **1** **2**	
Flow Control	read	get_rts3scfg_value MODBUS_RTU Flow_control	Flow control value for MODBUS/RTU	

Table 149: Description of the configuration scripts for " General SNMP information parameters "

Parameter	Status	Call	Output/Input	Validity
General SNMP information parameters				
Name of Device	read	get_snmp_data device-name	Specify the SNMP “sysName” parameter	immediate
	write	config_snmp device-name=<value>	Change the SNMP “sysName” parameter (<value> = string). *	after restart
Description	read	get_snmp_data description	Specifies the SNMP “sysDescr” parameter.	immediate
	write	config_snmp description=<value>	Changes the SNMP "sysDescr" parameter (<value> = string). *	after restart
Physical Location	read	get_snmp_data physical-location	Specifies the SNMP "sysLocation" parameter.	immediate
	write	config_snmp physical-location=<value>	Changes the SNMP "sysLocation" parameter (<value> = string). *	after restart
Contact	read	get_snmp_data contact	Specifies the SNMP "sysContact" parameter.	immediate
	write	config_snmp contact=<value>	Changes the SNMP "sysContact" parameter (<value> = string).	after restart
* When entering values, the blank characters must be filled by either "+" or "%20". Otherwise, the input is not detected as a coherent string.				
SNMP Manager Configuration for v1 and v2c				
Protocol Status	read	get_snmp_data v1-v2c-state	Returns the SNMP protocol status for v1/v2c as a string: **enabled** **disabled**	immediate
Local Community Name	read	get_snmp_data v1-v2c-community-name	Specifies the community name set for v1/v2c/	
Protocol Status/ Community Name	write	config_snmp v1-v2c-state=<value1> v1-v2c-community-name=<value2>	Activates/deactivates the v1/v2c protocol (<**value1**> = **enabled** or **disabled**) and assigns a community name. (value2> = string without blank characters, min. 1, max. 32 characters). **Note:** No community name is required for deactivation. Activation is only possible by entering a community name. Saving the community name is only possible if the protocol is activated.	after restart
SNMP Trap Receiver Configuration for v1 and v2c Any number of trap receivers can be configured. A trap receiver that has been set up is always active; the data set must be completely deleted to deactivate it.				

Table 149: Description of the configuration scripts for " General SNMP information parameters "

Parameter	Status	Call	Output/Input	Validity
IP Address of a Trap Receiver	read	get_snmp_data v1-v2c-trap-receiver-address <number>	Specifies the IP address of the trap receiver, to which the I/O-IPC should send the v1 or v2 traps. The <number> parameter serves to enable consecutive reading of the related data from the individually configured trap receiver for a short period of time (without interim changing of the data). This is a running number that is not connected to the data. If the number is not included, the data of the first receiver are read.	immediate
Community Name	read	get_snmp_data v1-v2c-trap-receiver-community-name <number>	Specifies the community name that the SNMP agent of the IPC sends in the Trap Header. Parameter <number>, see section “IP Address of a Trap Receiver”.	
Trap Version	read	get_snmp_data v1-v2c-trap-receiver-version <number>	Specifies the SNMP version ("v1" or "v2c") via which the SNMP agent sends the traps to the associated trap receiver address. Parameter <number>, see section “IP Address of a Trap Receiver”.	
Creating/ deleting a trap receiver	write	config_snmp v1-v2c-trap-receiver-edit=<value1> v1-v2c-trap-receiver-address=<value2> v1-v2c-trap-receiver-community-name=<value3> v1-v2c-trap-receiver-version=<value4>	Create a new trap receiver (value1=**add**) or delete an already configured trap receiver (value1=**delete**). Additional parameters: <value2> = IP address (**number.number.number.number**), to which the IPC should send the traps. <value3>: community string (string), which the IPC enters into the trap header. <value4>: SNMP version, via which the traps are sent (**v1** or **v2c**). **Note:** When deleting a trap receiver, all parameters must also be entered, as this is the only means to uniquely identify the data set.	after restart

Table 149: Description of the configuration scripts for " General SNMP information parameters "

Parameter	Status	Call	Output/Input	Validity
Configuration of SNMP v3 Any number of SNMP v3 users can be created. A user that has been set up is always active; the complete data set must be deleted to deactivate a user.				
Authentication Name	read	get_snmp_data v3-auth-name <number>	Specifies the user name of the v3 user. The <number> parameter serves to enable consecutive reading of the related data from the individually configured trap receiver for a short period of time (without interim changing of the data). This is a running number that is not connected to the data. If the number is not included, the data of the first user are read.	immediate
Authentication Encryption Type	read	get_snmp_data v3-auth-type <number>	Specifies the type of encryption that the v3 user uses (**none**, **MD5**, or **SHA**). Parameter <number> see section “Authentication Name”.	
Authentication Key	read	get_snmp_data v3-auth-key <number>	Specifies the key string for authentication. Parameter <number> see section “Authentication Name”.	immediate
Privacy Encryption Type	read	get_snmp_data v3-privacy <number>	Specifies the type of privacy encryption for the v3 user (**none**, **DES**, or **AES**). Parameter <number> see section “Authentication Name”.	
Privacy Key	read	get_snmp_data v3-privacy-key <number>	Assigns the key string for privacy. If nothing is entered here, the SNMP agent will use the "Authentication Key" for this. Parameter <number> see section “Authentication Name”.	
Trap Receiver Address	read	get_snmp_data v3-notification-receiver <number>	IP address of an SNMP manager, to which the agent traps for this v3 user are sent. If nothing is entered here, no traps are sent for this user. Parameter <number> see section “Authentication Name”.	

Table 149: Description of the configuration scripts for " General SNMP information parameters "

Parameter	Status	Call	Output/Input	Validity
Add new v3-User	write	config_snmp v3-edit=add v3-auth-name=<value1> v3-auth-type=<value2> v3-auth-key=<value3> v3-privacy=<value4> v3-privacy-key=<value5> v3-notification-receiver=<value6>	Creating a new v3 user. v3-auth-name: user name, string without blank characters, maximum 32 characters. The user name may not have been previously assigned. Parameters: User name (<value1> = string) Encryption type: (<value2> = **none**, **MD5**, or **SHA**). Key string for authentication, (<value3> = string with min. 8 and max. 32 characters). Privacy encryption type (<value4> = **none**, **DES**, or **AES**). Privacy key string (<value5> = string, min. 8 and max. 32 characters), can be empty, in which case the authentication key is used. The IP address of a trap receiver is transmitted as the notification receiver (<value6> = **number.number.number.number**). If no v3 traps are to be sent, this entry is omitted.	after restart
Delete v3-User	write	config_snmp v3-edit=delete v3-auth-name=<value>	Deleting a v3 user that has been set up. Because the doubled allocation of the same user name is prevented when creating a user, the name is sufficient to uniquely identify a data set (<value> = string).	

18.2 WagoLibNetSnmp.lib

The WagoLibNetSnmp.lib library is an external CODESYS library. It serves to create customer-specific object identifiers (OIDs) and to set/query these values from the SPS program.

The following functions are therefore available:

- Create/Register: snmpRegisterCustomOID_xxxxx
- Query: snmpGetValueCustomOID_xxxxx
- Set: snmpSetValueCustomOID_xxxxx

Note

OID Variables
Already created OID variables remain in existence until the system is restarted or the program is loaded at the control. During an "Online-Change", the OIDs remain stable.

Note

Available Variable Memory
8 kB of variable memory are available. Therefore, you can create a max. 32 OIDs of the "octet string" type or 2048 "integer" or "gauge32" OIDs.

Variables

The following data types are supported:

Table 150: Data type

OID data type	CODESYS data type	Length (in bytes)
Integer	DINT	4
UInteger, Gauge32	UDINT, DWORD	4
Octet String	STRING(255)	255

Functions

Functions of registered, customer-specific OIDs:

An OID can only be registered once, a repeated call up of these functions for the same OID returns the error:
2 (RET_ERR_OID_ALREADY_REGISTERED)

18.2.1 snmpRegisterCustomOID_INT32()

Registers an OID for the data type: Integer (4 bytes)

Table 151: Parameter snmpRegisterCustomOID_INT32()

Parameter	Description
sOID : STRING(128)	numerical OID i.e.: .1.3.6.1.4.1.8072.2.4.1.1.1.0
iDefValue : DINT	default value i.e.: 2
bReadOnly : BOOL	true, if read-only variable

Table 152: Return snmpRegisterCustomOID_INT32()

Return	Description
WORD	error number: 0 = ok

```
FUNCTION snmpRegisterCustomOID_INT32 : WORD
VAR_INPUT
      sOID : STRING(128) := '';
      iDefValue : DINT := 0;
      bReadOnly : BOOL := FALSE;
END_VAR
```

18.2.2 snmpRegisterCustomOID_STRING()

Registers an OID of the data type: Octet string (255 bytes)

Table 153: Parameter snmpRegisterCustomOID_STRING()

Parameter	Description
sOID : STRING(128)	numerical OID i.e.: .1.3.6.1.4.1.8072.2.4.1.1.1.0
sDefValue : STRING(255)	default value i.e.: 'hallo world'
bReadOnly : BOOL	true, if read-only variable

Table 154: Return snmpRegisterCustomOID_STRING()

Return	Description
WORD	error number: 0 = ok

```
FUNCTION snmpRegisterCustomOID_STRING : WORD
VAR_INPUT
      sOID : STRING(128) := '';
      sDefValue : STRING(255) := '';
      bReadOnly : BOOL := FALSE;
END_VAR
```

18.2.3 snmpRegisterCustomOID_UINT32()

Registers an OID of the data type: UInteger, Gauge32 (4 bytes)

Table 155: Parameter snmpRegisterCustomOID_UINT32()

Parameter	Description
sOID : STRING(128)	numerical OID i.e.: .1.3.6.1.4.1.8072.2.4.1.1.1.0
uiDefValue : UDINT;	default value i.e.: 2
bReadOnly : BOOL	true, if read-only variable

Table 156: Return snmpRegisterCustomOID_UINT32()

Return	Description
WORD	error number: 0 = ok

```
FUNCTION snmpRegisterCustomOID_UINT32 : WORD
VAR_INPUT
      sOID : STRING(128) := '';
      uiDefValue : UDINT := 0;
      bReadOnly : BOOL := FALSE;
END_VAR
```

18.2.4 snmpGetValueCustomOID_INT32()

Queries the value of an OID of the data type: Integer (4 bytes)

Table 157: Parameter snmpGetValueCustomOID_INT32()

Parameter	Description
sOID : STRING(128)	numerical OID i.e.: .1.3.6.1.4.1.8072.2.4.1.1.1.0
iValue : DINT	OID value

Table 158: Return snmpGetValueCustomOID_INT32()

Return	Description
WORD	error number: 0 = ok
iValue : DINT	OID value

```
FUNCTION snmpGetValueCustomOID_INT32 : WORD
VAR_INPUT
      sOID : STRING(128) := '';
END_VAR
VAR_IN_OUT
      iValue : DINT;
END_VAR
```

18.2.5 snmpGetValueCustomOID_STRING()

Queries the value of an OID of the data type: Octet string (255 bytes)

Table 159: Parameter snmpGetValueCustomOID_STRING()

Parameter	Description
sOID : STRING(128)	numerical OID i.e.: .1.3.6.1.4.1.8072.2.4.1.1.1.0
sValue : STRING(255);	OID value

Table 160: Return snmpGetValueCustomOID_STRING()

Return	Description
WORD	error number: 0 = ok
sValue : STRING(255);	OID value

```
FUNCTION snmpGetValueCustomOID_STRING : WORD
VAR_INPUT
      sOID : STRING(128) := '';
END_VAR
VAR_IN_OUT
      sValue : STRING(255);
END_VAR
```

18.2.6 snmpGetValueCustomOID_UINT32()

Queries the value of an OID of the data type: UInteger, Gauge32 (4 bytes)

Table 161: Parameter snmpGetValueCustomOID_INT32()

Parameter	Description
sOID : STRING(128)	numerical OID i.e.: .1.3.6.1.4.1.8072.2.4.1.1.1.0
uiValue : UDINT;	OID value

Table 162: Return snmpGetValueCustomOID_INT32()

Return	Description
WORD	error number: 0 = ok
uiValue : UDINT;	OID value

```
FUNCTION snmpGetValueCustomOID_INT32 : WORD
VAR_INPUT
      sOID : STRING(128) := '';
END_VAR
VAR_IN_OUT
      uiValue : UDINT .= 0;
END_VAR
```

18.2.7 snmpSetValueCustomOID_INT32()

Sets the value of an OID of the data type: Integer (4 bytes)

Table 163: Parameter snmpSetValueCustomOID_INT32()

Parameter	Description
sOID : STRING(128)	numerical OID i.e.: .1.3.6.1.4.1.8072.2.4.1.1.1.0
iValue : DINT	new default value i.e.: 2

Table 164: Return snmpSetValueCustomOID_INT32()

Return	Description
WORD	error number: 0 = ok

```
FUNCTION snmpRegisterCustomOID_INT32 : WORD
VAR_INPUT
      sOID : STRING(128) := '';
      iDefValue : DINT := 0;
END_VAR
```

18.2.8 snmpSetValueCustomOID_STRING()

Sets the value of an OID of the data type: Octet string (255 bytes)

Table 165: Parameter snmpSetValueCustomOID_STRING()

Parameter	Description
sOID : STRING(128)	numerical OID i.e.: .1.3.6.1.4.1.8072.2.4.1.1.1.0
sDefValue : STRING(255)	new value i.e.: 'hallo world'

Table 166: Return snmpSetValueCustomOID_STRING()

Return	Description
WORD	error number: 0 = ok

```
FUNCTION snmpRegisterCustomOID_STRING : WORD
VAR_INPUT
      sOID : STRING(128) := '';
      sDefValue : STRING(255) := '';
END_VAR
```

18.2.9 snmpSetValueCustomOID_UINT32()

Sets the value of an OID of the data type: UInteger, Gauge32 (4 bytes)

Table 167: Parameter snmpSetValueCustomOID_UINT32()

Parameter	Description
sOID : STRING(128)	numerical OID i.e.: .1.3.6.1.4.1.8072.2.4.1.1.1.0
uiDefValue : UDINT;	new value i.e.: 2

Table 168: Return snmpSetValueCustomOID_UINT32()

Return	Description
WORD	error number: 0 = ok

```
FUNCTION snmpRegisterCustomOID_UINT32 : WORD
VAR_INPUT
      sOID : STRING(128) := '';
      uiDefValue : UDINT := 0;
END_VAR
```

18.2.10 Feedback

The following error messages can be returned by the functions:

Table 169: Error messages

Value	Definition	Description
0	RET_SUCCESS	All ok, no error
1	RET_ERR_WRONG_OID	Incorrect OID, only numeric variables are supported, e.g.: .1.3.6.1.4.1.8072.2.4.1.1.1.0 Max. 32 points permissible here. The highest numerical value may be 2^31-1 = 2147483647.
2	RET_ERR_OID_ALREADY_REGISTERED	OID is already registered
3	RET_ERR_OID_NOT_FOUND	OID is not registered -> Register OID via the snmpRegisterOID_xxx – function
4	RET_ERR_IPC_COMM_NOT_INITIALIZED	Communication between SPS runtime environment and the Net SNMP agent has been interrupted -> reboot system
5	RET_ERR_IPC_COMM_FAILED	Insufficient variable memory available -> only 8 kB of variable memory is available

18.2.11 Example Program "Test.pro"

The "Test.pro" example program illustrates registration, querying, and setting customer-specific OIDs:

Program variables

```
PROGRAM PLC_PRG
VAR

    (* Flags *)
    bRegisterOID_INT32:BOOL := FALSE;
    bRegisterOID_STRING:BOOL := FALSE;
    bRegisterOID_UINT32:BOOL := FALSE;
    bSetValueOID_INT32:BOOL := FALSE;
    bSetValueOID_STRING:BOOL := FALSE;
    bSetValueOID_UINT32:BOOL := FALSE;
    bGetValueOID_INT32:BOOL := FALSE;
    bGetValueOID_STRING:BOOL := FALSE;
    bGetValueOID_UINT32:BOOL := FALSE;
    (*CustomOIDs *)
    sCustomOID1:STRING(128) := '.1.3.6.1.4.1.8072.2.4.1.1.1.0'; (* Integer32 *)
    sCustomOID2:STRING(128) := '.1.3.6.1.4.1.8072.2.4.1.1.2.0'; (* OctetString *)
    sCustomOID3:STRING(128) := '.1.3.6.1.4.1.8072.2.4.1.1.3.0'; (* UInteger32 *)
    (* Values *)
    iValue:DINT := 11;
    sValue:STRING(255) := 'test';
    uiValue:UDINT := 33;
    (* Error *)
    wError:WORD := 0;
END_VAR
```

Program block

```
(* Register new OID with Integer value  *)
IF bRegisterOID_INT32 = TRUE THEN
        wError := snmpRegisterCustomOID_INT32(sOID1, iValue, FALSE);
        bRegisterOID_INT32 := FALSE;
END_IF;
(* Register new OID with OctetString value*)
IF bRegisterOID_STRING = TRUE THEN
        wError := snmpRegisterCustomOID_STRING(sOID2, sValue, FALSE);
        bRegisterOID_STRING := FALSE;
END_IF;
(* Register new OID with UInteger value *)
IF bRegisterOID_UINT32 = TRUE THEN
        wError := snmpRegisterCustomOID_UINT32(sOID3, uiValue, FALSE);
        bRegisterOID_UINT32 := FALSE;
END_IF;
(* Set Integer value *)
IF bSetValueOID_INT32 = TRUE THEN
        wError := snmpSetValueCustomOID_INT32(sOID1, iValue+1);
        bSetValueOID_INT32 := FALSE;
END_IF;
(* Set OctetString value *)
IF bSetValueOID_STRING = TRUE THEN
        sValue := 'hello wolrd';
        wError := snmpSetValueCustomOID_STRING(sOID2, sValue);
        bSetValueOID_STRING := FALSE;
END_IF;
(* Set UInteger value *)
IF bSetValueOID_UINT32 = TRUE THEN
        wError := snmpSetValueCustomOID_UINT32(sOID3, uiValue+1);
        bSetValueOID_UINT32 := FALSE;
END_IF;
(* Get Integer value *)
IF bGetValueOID_INT32 = TRUE THEN
        wError := snmpGetValueCustomOID_INT32(sCustomOID1, iValue);
        bGetValueOID_INT32 := FALSE;
END_IF;
(* Get OctetString value *)
IF bGetValueOID_STRING = TRUE THEN
        wError := snmpGetValueCustomOID_STRING(sOID2, sValue);
        bGetValueOID_STRING := FALSE;
END_IF;
(* Get UInteger value *)
```

```
IF bGetValueOID_UINT32 = TRUE THEN
        wError := snmpGetValueCustomOID_UINT32(sOID3, uiValue);
        bGetValueOID_UINT32 := FALSE;
END_IF;
```

The custom OIDs can be registered, queried, and set via the “TEST” visualization form:

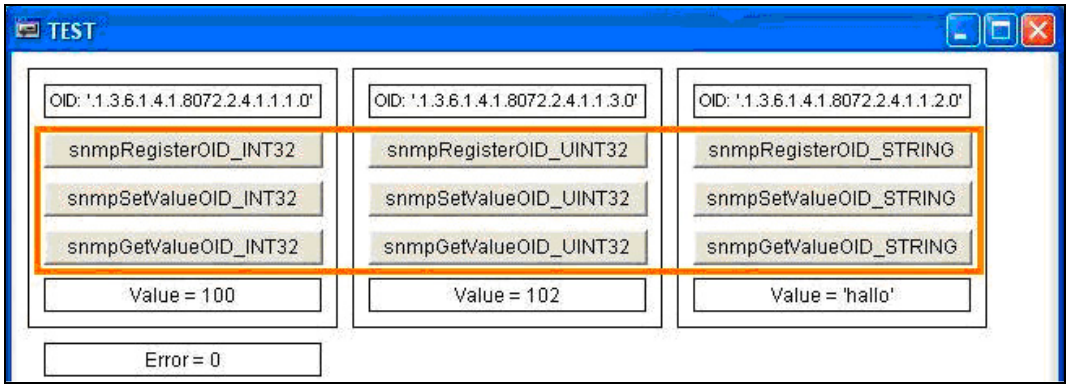


Figure 1: "TEST" visualization form

18.3 WAGO_CANopen_02.lib

This library contains function blocks specified according to CiA Draft Standard 405.

18.3.1 CIA405_GET_KERNEL_STATE

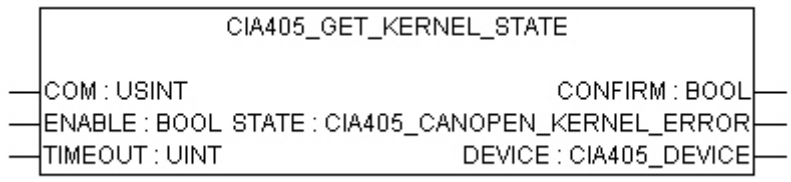
The CIA405_GET_CANOPEN_STATE function block outputs the state of the CANopen kernel.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_ KERNEL _STATE
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

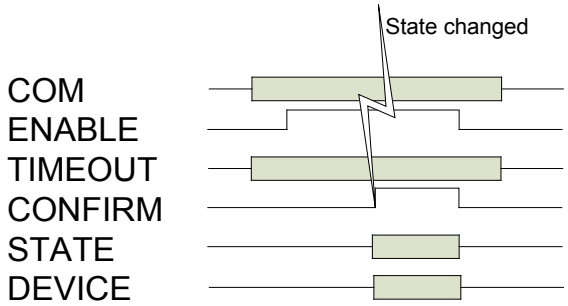
Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
ENABLE	BOOL	TRUE, FALSE	Activation of the function
TIMEOUT	UINT	0..65535d	Implemented for reason of compatibility. Input is not captured.

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	Execution confirmation
STATE	CIA405_ CANOPEN_ KERNEL_ ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ ERROR
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE

Graphical illustration



Time referenced behavior



Description

The status "Other Error" of the CIA405_CANOPEN_KERNEL_ERROR structure is not output to this module because the module with the output "CIA405_SDO_ERROR" that provides more detailed information is not assigned.

This module can output the following error statuses of the CANOPEN_KERNEL_ERROR structure:

0010h CAN controller in the "Bus off" state
0011h CAN controller has exceeded the "Error warning limit"
0021h No response received from the node
0026h Internal error.
0027h Node-Guarding error
0028h Device not in Operational mode

If the *ENABLE* input switches to *TRUE*, the monitoring function is enabled. Upon each PLC cycle, the CIA405_GET_CANOPEN_STATE is checked. *CONFIRM* outputs *TRUE* when an error status according to the list is detected.
This status is output on the *STATE* output.

If present, the first device is specified in which this state is detected.
Exiting Operational mode of a node can only be detected if Node-Guarding is activated for the device.

18.3.2 CIA405_GET_LOCAL_NODE_ID

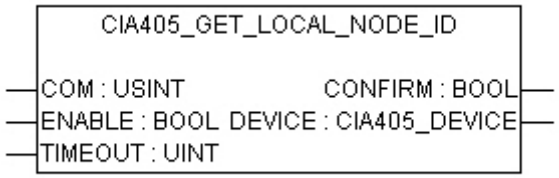
The CIA405_GET_LOCAL_NODE_ID function block outputs its own module ID.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

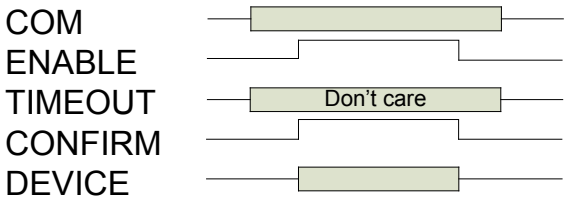
Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
ENABLE	BOOL	TRUE, FALSE	Activation of the function
TIMEOUT	UINT	0..65535d	Implemented for reason of compatibility. Input is not captured.

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	Execution confirmation
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE

Graphical illustration



Time referenced behavior



Description
After setting the *ENABLE* input, *CONFIRM* switches to *TRUE* and the *DEVICE* output makes the module ID available. No blocking access is carried out internally. Use of the Timeout input is not required. Like the *COM* input, this input is implemented for reasons of compatibility. When the *ENABLE* input is reset, the module is returned to the initial state.

18.3.3 CIA405_RECV_EMCY

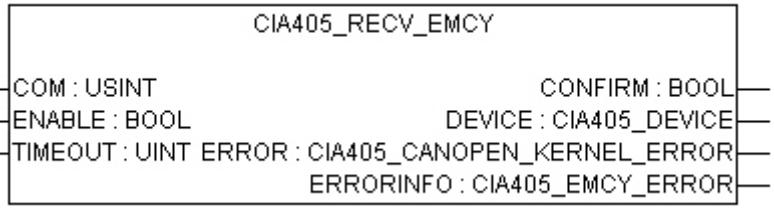
The CIA405_RECV_EMY function block checks if an emergency message has been received from any configured device.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
ENABLE	BOOL	TRUE, FALSE	Activation of the function
TIMEOUT	UINT	0..65535d	Implemented for reason of compatibility. Input is not captured.

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	TRUE: Emergency received FALSE: No emergency received
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
ERROR	CIA405_ CANOPEN_ KERNEL_ ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ ERROR
ERRORINFO	CIA405_ EMCY_ ERROR	STRUCT	See section Data Type: CIA405_EMCY_ERROR

Graphical illustration



Time referenced behavior

Description
If the *ENABLE* input is set to *TRUE*, the monitoring function is activated. Upon each PLC cycle, receipt of an emergency message and occurrence of a CANOPEN_KERNEL_ERROR is checked. Before the module is first called up with *ENABLE = TRUE*, emergencies are cleared from the diagnostic memory and not displayed. As the program continues, emergencies are also saved if the module is not activated. After activation of the module with *ENABLE = TRUE*, the last received emergency is output. When an emergency is received, *COMFIRM* is set to *TRUE*. *ERRORINFO* contains the emergency information specified according to DS301 and the *DEVICE* output contains the module ID of the emergency sender. If a CANOPEN_KERNEL_ERROR is detected ... 0010h CAN controller in the "Bus off" state 0011h CAN controller has exceeded the "Error warning limit" 0021h No response received from the node 0026h Internal error 0027h Node-Guarding error 0028h Device not in Operational mode ... *CONFIRM* stays *FALSE* and the error is output on the ERROR output. If present, the node ID is output on the *DEVICE* output. No blocking access is carried out internally. Use of the Timeout input is not required. Like the *COM* input, this input is implemented for reasons of compatibility. When the *ENABLE* input is reset, the module is returned to the initial state.

18.3.4 CIA405_RECV_EMCY_DEV

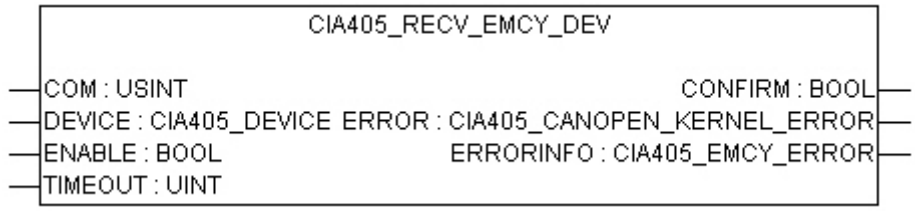
The CIA405_RECV_EMY_DEV function block checks if an emergency message has been received from a dedicated and configured device.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

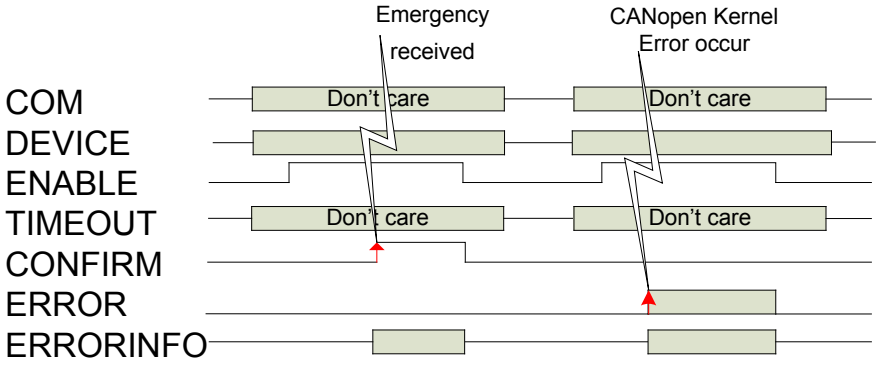
Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
ENABLE	BOOL	TRUE, FALSE	Activation of the function
TIMEOUT	UINT	0..65535d	Maximum execution time in ms. Disabled in none or input outside the valid range [10..900].

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	TRUE: Emergency received from the specified module FALSE: No emergency received from the specified module
ERROR	CIA405_ CANOPEN_ KERNEL_ ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ ERROR
ERRORINFO	CIA405_ EMCY_ ERROR	STRUCT	See section Data Type: CIA405_EMCY_ERROR

Graphical illustration



Time referenced behavior



Description
If the *ENABLE* input is set to *TRUE*, the monitoring function is activated. Upon each PLC cycle, receipt of an emergency message and occurrence of a CANOPEN_KERNEL_ERROR is checked. Before the module is first called up with *ENABLE = TRUE*, emergencies are cleared from the diagnostic memory and not displayed. As the program continues, emergencies are also saved if the module is not activated. After activation of the module with *ENABLE = TRUE*, the last received emergency is output. The emergency is displayed when the *DEVICE* input matches the node ID of the produced emergency. This error is only set if produced by the device with the node ID of the *DEVICE* input. When an emergency is received, *COMFIRM* is set to *TRUE*. *ERRORINFO* contains the emergency information specified according to DS301. If a CANOPEN_KERNEL_ERROR of the dedicated device is detected ... 0010h CAN controller in the "Bus off" state 0011h CAN controller has exceeded the "Error warning limit" 0021h No response received from the node 0026h Internal error. 0027h Node-Guarding error 0028h Device not in Operational mode ... *CONFIRM* stays *FALSE* and the error is output on the ERROR output. No blocking access is carried out internally. Use of the Timeout input is not required. Like the COM input, this input is implemented for reasons of compatibility. When the ENABLE input is reset, the module is returned to the initial state.

18.3.5 CIA405_GET_STATE

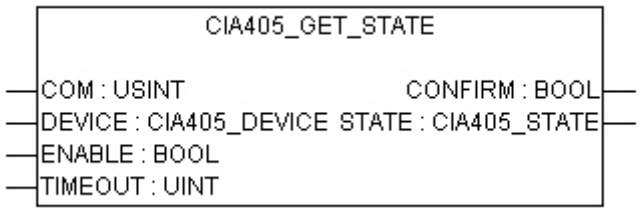
The CIA405_GET_STATE function block returns the CANopen network status of the selected CANopen node.

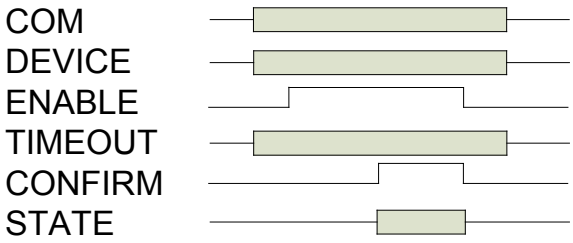
Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
ENABLE	BOOL	TRUE, FALSE	Activation of the function
TIMEOUT	UINT	0..65535d	Maximum execution time in ms. Disabled in none or input outside the valid range [10..900].

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	Execution confirmation
STATE	CIA405_ STATE	0..7	See section Data Type "CIA405_STATE".

Graphical illustration



Time referenced behavior


Description
The *UNKNOWN* status is returned if the selected device is not monitored using Node Guarding. If the device is not available, the *NOT_AVAIL* status is returned. The network status of the master is queried by entering its own device ID. The master can output the *STOPPED*, *OPERATIONAL* and *UNKNOWN* statuses. To prevent the PLC from being blocked, this module can be written with a Timeout time. When slaves are not available, this function is exited after the time set. No time is specified initially and is therefore not active.

18.3.6 CIA405_NMT

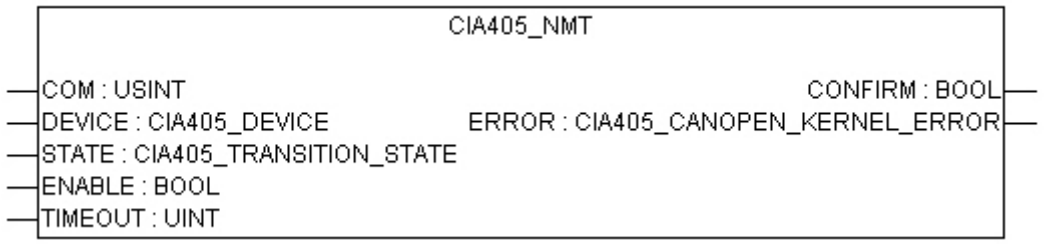
The CIA405_NMT function block can be used to trigger a network status change of a previously configured module.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
STATE	CIA405_ TRANSISITIO N_ STATE	0..4	See section Data Type: CIA405_TRANSISITION_ STATE
ENABLE	BOOL	TRUE, FALSE	Activation of the function
TIMEOUT	UINT	0..65535d	Implemented for reason of compatibility. Input is not captured.

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	Execution confirmation
ERROR	CIA405_ CANOPEN_ KERNEL_ ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ ERROR

Graphical illustration



Time referenced behavior

Description

If a '0' is passed to the *DEVICE* input, all CANopen modules communicating on the bus are transferred to the new state.

The status is changed be setting *ENABLE* to *TRUE*. After execution, *CONFIRM* is set to *TRUE*. The status of the status change is displayed in the *ERROR* output parameter. By setting the *ENABLE* parameter to *FALSE* (by the caller), *CONFIRM* is automatically set to *FALSE* and the function block is back in the initial state.

18.3.7 CIA405_SDO_READ4

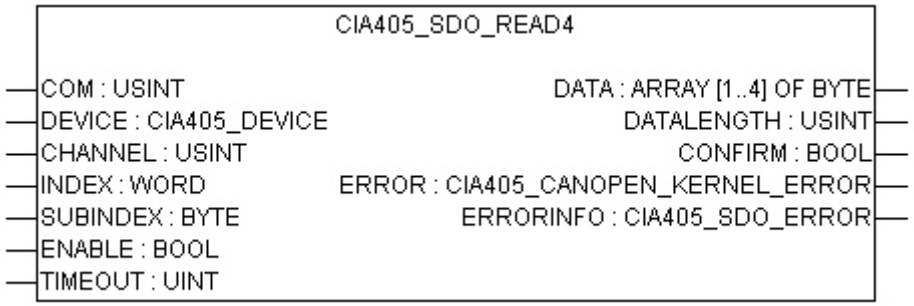
The CIA405_SDO_READ4 function block returns the value of an object directory entry with a maximum data length of four bytes.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

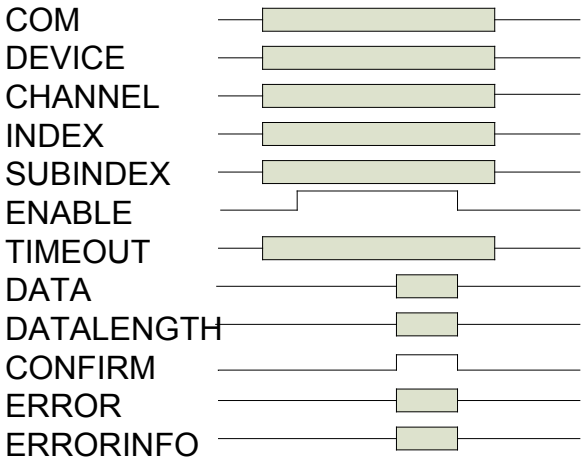
Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
CHANNEL	USINT	1..128d	SDO channel number. This input is available for reasons of compatibility and is ignored by the IPC.
INDEX	WORD	0..65535d	Index of the object directory
SUBINDEX	BYTE	0..255d	Sub index of the object directory
ENABLE	BOOL	TRUE, FALSE	Activation of the function
TIMEOUT	UINT	0..65535d	Maximum execution time in ms. Disabled in none or input outside the valid range [10..900].

Output parameters			
Name	**Data Type**	**Value range**	**Description**
DATA	ARRAY [1..4] of BYTE	for each: 0..255d	Received data
DATALENGTH	USINT	1..4	Qty. of valid data.
CONFIRM	BOOL	TRUE, FALSE	Execution confirmation
ERROR	CIA405_CANOPEN_KERNEL_ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ERROR
ERRORINFO	CIA405_SDO_ERROR	0..FFFFFFFFh	See section Data Type: CIA405_SDO_ERROR

Graphical illustration



Time referenced behavior



Description
The CANopen node is selected by the "DEVICE" parameter. Index and subindex describe the object directory entry to be read. The result of the function block call can only be available a few cycles later. To prevent the PLC from being blocked, this function can be written with a Timeout time. When slaves are not available, this function can be exited before the SDO abort time. No time is specified initially and is therefore not active. After all input parameters have valid values, *ENABLE* is set to *TRUE* by the caller. The SDO frame is sent. If the SDO response telegram is received without error, it is displayed by setting *CONFIRM* to *TRUE*. The *DATA* and *DATALENGTH* output parameters now have valid values and *ERROR* is 0. By setting the *ENABLE* parameter to *FALSE* (by the caller), *CONFIRM* is automatically set to *FALSE* and the function block is back in the initial state. If an error occurs during SDO transmission (e.g. Timeout, index not available, etc.), *CONFIRM* is also set to *TRUE*, but *ERROR* has a value not equal to 0. The *ERRORINFO* parameter contains additional information if any. This abort information is specified according to DS301. *ENABLE* must be set to *FALSE* after the error is evaluated to set the function block back to the initial state.

18.3.8 CIA405_SDO_READxx

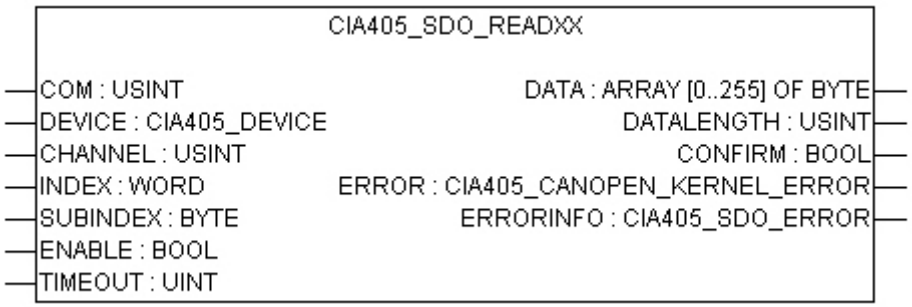
The CIA405_SDO_READxx function block returns the value of an object directory entry with a maximum data length of 256 bytes.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

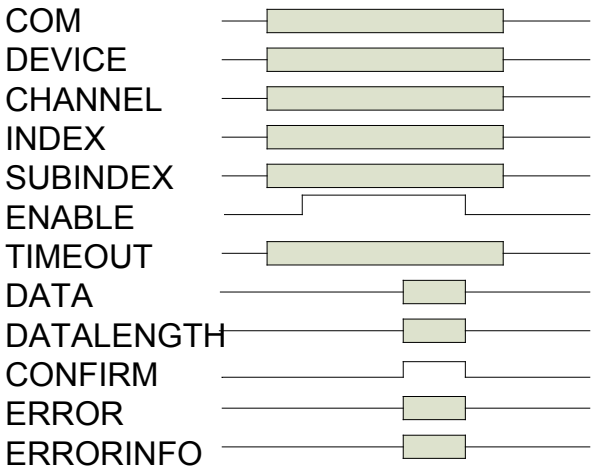
Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
CHANNEL	USINT	1..128d	SDO channel number. This input is available for reasons of compatibility and is ignored by the IPC.
INDEX	WORD	0..65535d	Index of the object directory
SUBINDEX	BYTE	0..255d	Sub index of the object directory
ENABLE	BOOL	TRUE, FALSE	Activation of the function
TIMEOUT	UINT	0..65535d	Maximum execution time in ms. Disabled in none or input outside the valid range [10..900].

Output parameters			
Name	**Data Type**	**Value range**	**Description**
DATA	ARRAY [1..248] of BYTE	for each: 0..255d	Received data
DATALENGTH	USINT	1..248	Qty. of valid data.
CONFIRM	BOOL	TRUE, FALSE	Execution confirmation
ERROR	CIA405_CANOPEN_KERNEL_ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ERROR
ERRORINFO	CIA405_SDO_ERROR	0..FFFFFFFFh	See section Data Type: CIA405_SDO_ERROR

Graphical illustration



Time referenced behavior



Description
The CANopen node is selected by the "DEVICE" parameter. Index and subindex describe the object directory entry to be read. The result of the function block call can only be available a few cycles later. To prevent the PLC from being blocked, this function can be written with a Timeout time. When slaves are not available, this function can be exited before the SDO abort time. No time is specified initially and is therefore not active. After all input parameters have valid values, *ENABLE* is set to *TRUE* by the caller. The SDO frame is sent. If the SDO response telegram is received without error, it is displayed by setting *CONFIRM* to *TRUE*. The *DATA* and *DATALENGTH* output parameters now have valid values and *ERROR* is 0. By setting the *ENABLE* parameter to FALSE (by the caller), *CONFIRM* is automatically set to *FALSE* and the function block is back in the initial state. If an error occurs during SDO transmission (e.g. Timeout, index not available, etc.), *CONFIRM* is also set to *TRUE*, but *ERROR* has a value not equal to 0. The *ERRORINFO* parameter contains additional information if any. This abort information is specified according to DS301. *ENABLE* must be set to *FALSE* after the error is evaluated to set the function block back to the initial state.

18.3.9 CIA405_SDO_WRITE4

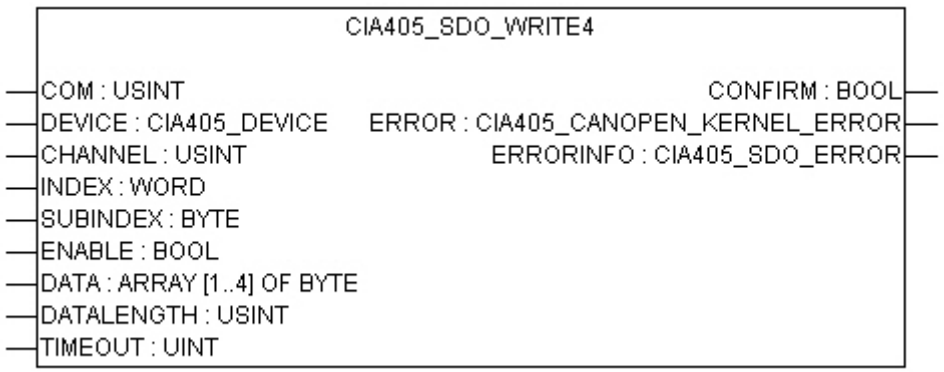
The CIA405_SDO_WRITE4 function block writes the specified data with the maximum data length of 4 bytes to the object directory of a module.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

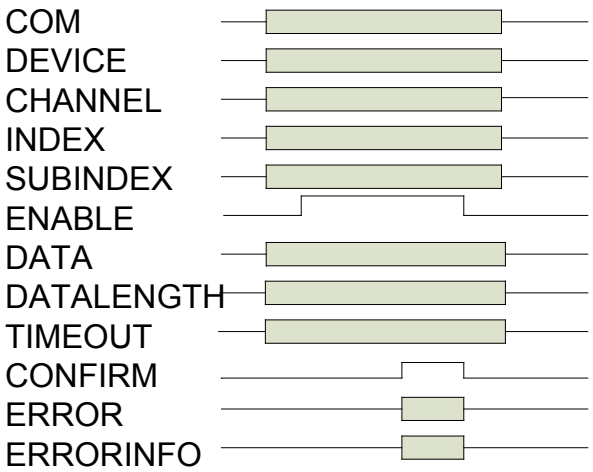
Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
CHANNEL	USINT	1..128d	SDO channel number. This input is available for reasons of compatibility and is ignored by the IPC.
INDEX	WORD	0..65535d	Index of the object directory
SUBINDEX	BYTE	0..255d	Sub index of the object directory
ENABLE	BOOL	TRUE, FALSE	Activation of the function
DATA	ARRAY [1..4] of BYTE	for each: 0..255d	Data to be written
DATALENGTH	USINT	1..4	Qty. of valid data
TIMEOUT	UINT	0..65535d	Maximum execution time in ms. Disabled in none or input outside the valid range [10..900].

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	Execution confirmation
ERROR	CIA405_ CANOPEN_ KERNEL_ ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ ERROR
ERRORINFO	CIA405_ SDO_ ERROR	0..FFFFFFFFh	See section Data Type: CIA405_SDO_ERROR

Graphical illustration



Time referenced behavior



Description
The module is selected by the "DEVICE" parameter. Index and subindex specifies the object directory to be written. Index and subindex describe the object directory entry to be read. To prevent the PLC from being blocked, this function can be written with a Timeout time. When slaves are not available, this function can be exited before the SDO abort time. No time is specified initially and is therefore not active. After all input parameters have valid values, *ENABLE* is set to *TRUE* by the caller. The SDO frame is sent. If the CAN software has transmitted the data to be written without error, it reports it by setting *CONFIRM* to *TRUE* and *ERROR = 0*. By setting the *ENABLE* parameter to *FALSE* (by the caller), *CONFIRM* is automatically set to *FALSE* and the function block is back in the initial state. If an error occurs during SDO transmission (e.g. Timeout, index not available, etc.), *CONFIRM* is also set to *TRUE*, but *ERROR* has a value not equal to 0. The *ERRORINFO* parameter contains additional information if any. This abort information is specified according to DS301. *ENABLE* must be set to *FALSE* after the error is evaluated.

18.3.10 CIA405_SDO_WRITExx

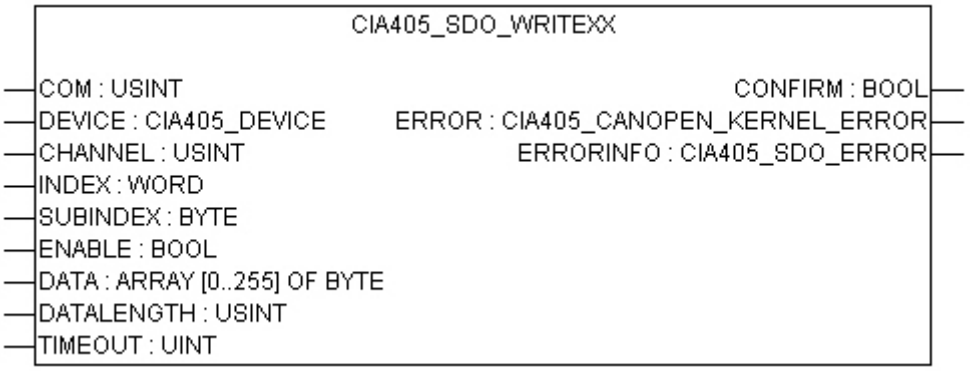
The CIA405_SDO_WRITExx function block writes the specified data with the maximum data length of 256 bytes to the object directory of a module.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

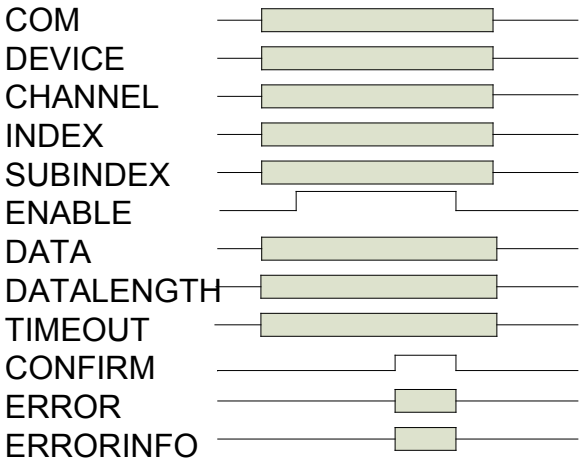
Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
CHANNEL	USINT	1..128d	SDO channel number. This input is available for reasons of compatibility and is ignored by the IPC.
INDEX	WORD	0..65535d	Index of the object directory
SUBINDEX	BYTE	0..255d	Sub index of the object directory
ENABLE	BOOL	TRUE, FALSE	Activation of the function
DATA	ARRAY [0..248] of BYTE	for each: 0..255d	Data to be written
DATALENGTH	USINT	0..248	Qty. of valid data
TIMEOUT	UINT	0..65535d	Maximum execution time in ms. Disabled in none or input outside the valid range [10..900].

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	Execution confirmation
ERROR	CIA405_ CANOPEN_ KERNEL_ ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ ERROR
ERRORINFO	CIA405_ SDO_ ERROR	0..FFFFFFFFh	See section Data Type: CIA405_SDO_ERROR

Graphical illustration



Time referenced behavior



Description
The module is selected by the "DEVICE" parameter. Index and subindex specifies the object directory to be written. Index and subindex describe the object directory entry to be read. To prevent the PLC from being blocked, this function can be written with a Timeout time. When slaves are not available, this function can be exited before the SDO abort time. No time is specified initially and is therefore not active. After all input parameters have valid values, *ENABLE* is set to *TRUE* by the caller. The SDO frame is sent. If the CAN software has transmitted the data to be written without error, it reports it by setting *CONFIRM* to *TRUE* and *ERROR = 0*. By setting the *ENABLE* parameter to *FALSE* (by the caller), *CONFIRM* is automatically set to *FALSE* and the function block is back in the initial state. If an error occurs during SDO transmission (e.g. Timeout, index not available, etc.), *CONFIRM* is also set to *TRUE*, but *ERROR* has a value not equal to 0. The *ERRORINFO* parameter contains additional information if any. This abort information is specified according to DS301. *ENABLE* must be set to *FALSE* after the error is evaluated.

18.3.11 NMT_GUARD_ERROR

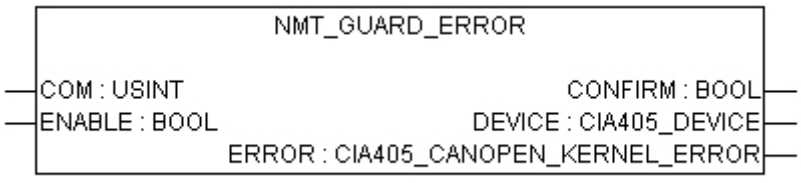
The NMT_GUARD_ERROR function block checks if a Node-Guarding protocol violation has been detected by a configured node.

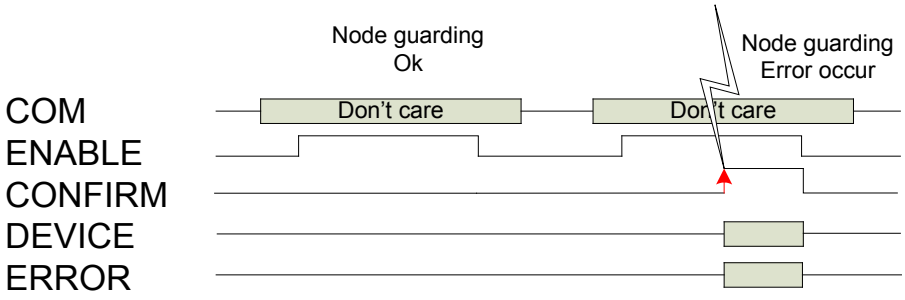
Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
ENABLE	BOOL	TRUE, FALSE	Activation of the function

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	TRUE: Guarding error has occurred FALSE: Node Guarding OK
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
ERROR	CIA405_ CANOPEN_ KERNEL_ ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ ERROR

Graphical illustration



Time referenced behavior


Description
If the *ENABLE* input is set to *TRUE*, the monitoring function is activated. Upon each PLC cycle, this verification is performed. *CONFIRM* is set to *TRUE* in the event of a protocol violation. The *ERROR* output displays the value 0x0027h (Node-Guarding error). If several nodes are affected, each node number is output sequentially each time the module is called starting with the smallest. When the ENABLE input is reset, the module is returned to the initial state.

18.3.12 NMT_GUARD_ERROR_DEV

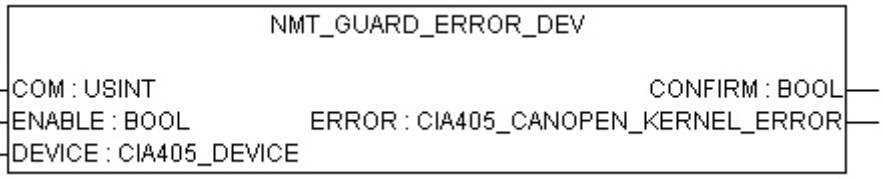
The NMT_GUARD_ERROR_DEV function block checks if a Node-Guarding protocol violation has been detected by a dedicated and configured node.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

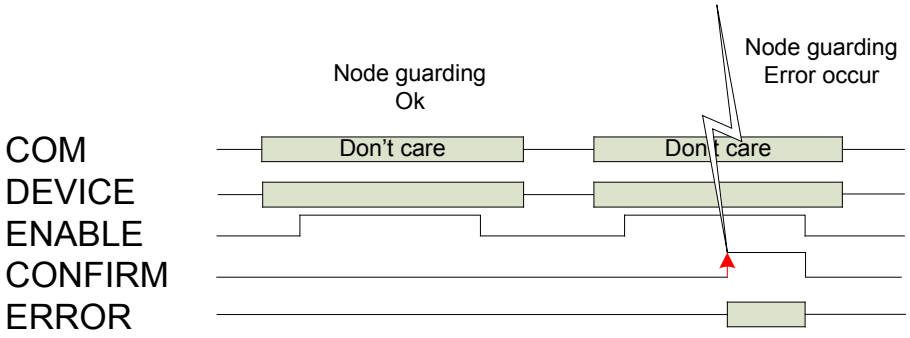
Input parameter			
Name	**Data Type**	**Value range**	**Description**
COM	USINT	0..255d	Selection of the CAN interface. This input is ignored by devices with an interface.
DEVICE	CIA405_ DEVICE	0..127d	See section Data Type: CIA405_DEVICE
ENABLE	BOOL	TRUE, FALSE	Activation of the function

Output parameters			
Name	**Data Type**	**Value range**	**Description**
CONFIRM	BOOL	TRUE, FALSE	TRUE: Guarding error has occurred FALSE: Node Guarding OK
ERROR	CIA405_ CANOPEN_ KERNEL_ ERROR	0..65535d	See section Data Type: CIA405_CANOPEN_KERNEL_ ERROR

Graphical illustration



Time referenced behavior



Description

If the *ENABLE* input is set to *TRUE*, the monitoring function is activated. Upon each PLC cycle, this verification is performed.

CONFIRM is set to *TRUE* in the event of a protocol violation. The ERROR output displays the value 0x0027h (Node-Guarding error).
Should a node be monitored that is not configured, *CONFIRM* is set to *TRUE* and *ERROR* outputs the value 0024h (node is not configured).
If a node ID is invalid, the error 0023h (No valid node address) is output.

When the ENABLE input is reset, the module is returned to the initial state.

18.3.13 CANOPEN_VERSION

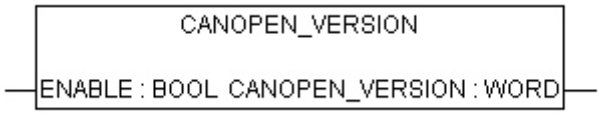
Outputs the version of the library.

Category	Function blocks for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Function block
Name of library	WAGO_CANopen_02.lib
Required libraries	
Applicable to	758-87x-112

Input parameter		
Name	**Data Type**	**Description**
ENABLE	BOOL	Activation of the function

Output parameters		
Name	Data Type	Description
CIA405_VERSION	WORD	Version of library

Graphical illustration



Time referenced behavior

Description

This function can be used during program development for information. In addition, version conflicts can be avoided at runtime.

Version:	**Description:**
1	IPC version created

18.3.14 CIA405_DEVICE

This could be an introductory text to the file type ...

Category	Data type for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Data Type
Name of library	WAGO_CANopen_02.lib
Applicable to	758-87x-112
Structure	TYPE CIA405_DEVICE : BYTE; END_TYPE

Element	Value	Description
CIA405_DEVICE	0..127	CANopen module address

Description
This type corresponds to the CANopen module address.

18.3.15 CIA405_SDO_ERROR

Category	Data type for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Data Type
Name of library	WAGO_CANopen_02.lib
Applicable to	758-87x-112
Structure	TYPE CIA405_SDO_ERROR : UDINT; END_TYPE

Element	Value	Description
CIA405_SDO_ERROR	0..FFFFFFFFh	Error information acc. DS301

Description
This structure variable contains the error information as specified in the DS301 standard in the chapter "Protocol SDO abort transfer".

18.3.16 CIA405_EMCY_ERROR

Category	Data type for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Data Type
Name of library	WAGO_CANopen_02.lib
Applicable to	758-87x-112
Structure	TYPE CIA405_EMCY_ERROR : STRUCT EMCY_ERROR_CODE : WORD; ERROR_REGISTER : BYTE; ERROR_FIELD : ARRAY [1..5] OF BYTE; END_STRUCT END_TYPE

Element	Value	Description
EMCY_ERROR_CODE	WORD	Error information acc. DS301
ERROR_REGISTER	BYTE	Error information acc. DS301
ERROR_FIELD	ARRAY [1..5]	Manufacturer specific array

Description
This structure contains the emergency error information as specified in the DS301 standard. The manufacturer-specific error codes must be taken from the device manufacturer documentation.

18.3.17 CIA405_STATE

Category	Data type for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Data Type
Name of library	WAGO_CANopen_02.lib
Applicable to	758-87x-112
Structure	TYPE CIA405_STATE : (INIT, RESET_COMM, RESET_APP, PRE_OPERATIONAL, STOPPED, OPERATIONAL, UNKNOWN, NOT_AVAIL); END_TYPE

Element	Value	Description
INIT	0	Status after power ON or hardware reset
RESET_COMM	1	Status in which the parameters of the communication profile are initialized. This status is not displayed by the IPC.
RESET_APP	2	Status in which the manufacturer-specific area is initialized. This status is not displayed by the IPC.
PRE_OPERATIONAL	3	SDO communication status
STOPPED	4	Communication stop; Node-Guarding and Heartbeat is still active.
OPERATIONAL	5	All communication objects are active.
UNKNOWN	6	Status cannot be determined because Node-Guarding or Heartbeat is not active.
NOT_AVAIL	7	Device is not available (timeout).

Description
These ENUM elements reflect the current CANopen network status as specified in Draft Standard 301.

18.3.18 CIA405_TRANSITION_STATE

Category	Data type for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Data Type
Name of library	WAGO_CANopen_02.lib
Applicable to	758-87x-112
Structure	TYPE CIA405_TRANSITION_STATE : (START_REMOTE_NODE, STOP_REMOTE_NODE, ENTER_PRE_OPERATIONAL, RESET_NODE, RESET_COMMUNICATION); END_TYPE

Element	Value	Description
START_REMOTE_NODE	0	Start module
STOP_REMOTE_NODE	1	Transfer module to Stopped
ENTER_PRE_OPERATIONAL	2	Transfer module to pre-operational
RESET_NODE	3	Reset the module
RESET_COMMUNICATION	4	Reset communication

Description
This enumeration provides an overview of the transition states to which a module can be transferred to as specified in the Draft Standard 301.

18.3.19 CANOPEN_KERNEL_ERROR

Category	Data type for CANopen acc. CiA 405
Name	CIA405_GET_LOCAL_NODE_ID
Type	Data Type
Name of library	WAGO_CANopen_02.lib
Applicable to	758-87x-112
Structure	TYPE CIA405_CANOPEN_KERNEL_ERROR : WORD; END_TYPE

Element	Value	Description
CIA405_CANOPEN_KERNEL_ERROR	0000h	No error detected by the CANopen kernel.
	0001h	Other error. Additional error information if any can be taken from the CIA405_SDO_ERROR output according to the standard DS301.
	0002h	Invalid data length.
	0003h	Timeout during module execution.
	0010h	CAN controller in "Bus off" state.
	0011h	CAN controller has exceeded the "Error warning limit".
	0021h	No response received from the node.
	0022h	The SDO channel is currently used by the IPC for the boot phase of the node and is locked for this function.
	0023h	No valid node address.
	0024h	Node is not configured.
	0025h	NMT command invalid or could not be transmitted.
	0026h	Internal error.
	0027h	Node-Guarding error.
	0028h	Device has exited Operational mode. Output only possible when Node-Guarding is activated.

Description
This table provides an overview of the values that the *CANOPEN_KERNEL_ERROR* can accept as specified in the Draft Standard 405.

18.4 WAGO_CANLayer2_01.lib

This library makes function block available that allow communication with CAN layer 2 via 11-bit and 29-bit identifiers.

18.4.1 CAN_LAYER2_VERSION

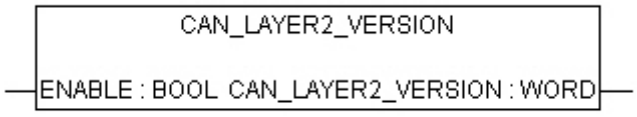
The CAN_LAYER2_VERSION function returns the current version number of the library.

Category	CAN Layer 2 communication
Name	CAN_LAYER2_VERSION
Type	Function
Name of library	WAGO_CANLayer2_01.lib
Required libraries	
Applicable to	758-87x-112, 762-3xx/000-001 or 762-3xx/000-003

Input parameter		
Name	**Data Type**	**Description**
ENABLE	BOOL	Activation of the function

Output parameters		
Name	**Data Type**	**Description**
CAN_LAYER2_VERSION	WORD	Version of library

Graphical illustration



Time referenced behavior

Description

This function can be used during program development for information. In addition, version conflicts can be avoided at runtime.

Version:	**History:**
1	I/O-IPC version created
2	Library revised

18.4.2 CAN_ERROR_INFO

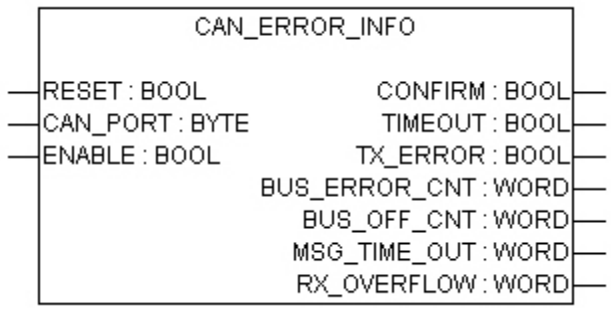
The CAN_ERROR_INFO function block outputs the physical state of the CAN bus.

Category:	CAN Layer 2 function blocks
Name:	CAN_ERROR_INFO
Type:	Function
Name of library:	WAGO_CANLayer2_01.lib
Required libraries	
Applicable to:	758-87x-112, 762-3xxx/000-001 or 762-3150/000-003

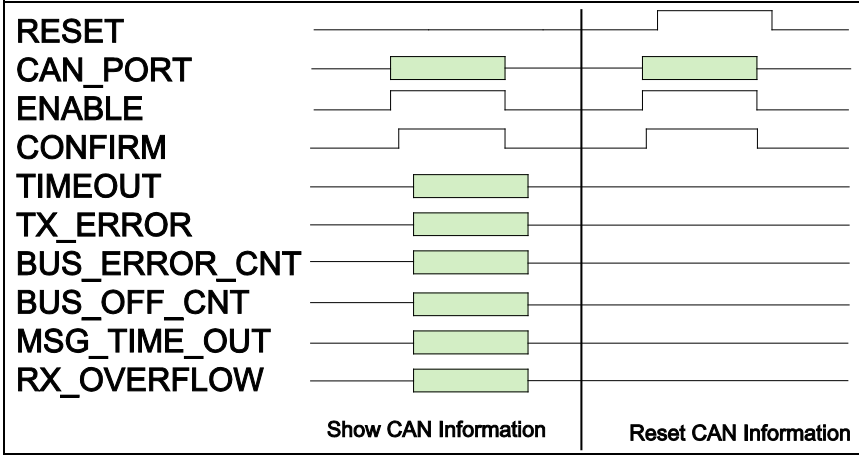
Input parameter:		
Name	**Data Type**	**Description**
RESET	BOOL	Error counter reset
CAN_PORT	BYTE	Selection of the CAN interface **0:** CAN 0 interface activated **1:** CAN 1 interface activated For devices with an interface without function.
ENABLE	BOOL	Activation of the function

Output parameters:		
Name	**Data Type**	**Description**
CONFIRM	BOOL	Execution confirmation
TIMEOUT	BOOL	A CANopen message has not been acknowledged by any CAN node. Indication that no other node on the bus is communicated.
TX_ERROR	BOOL	The I/O-IPC has detected at least one send error.
BUS_ERROR_CNT	WORD	Number of Warning Level overruns.
BUS_OFF_CNT	WORD	Number of automatically initiated CAN controller re-initializations.
MSG_TIME_OUT	WORD	Number of cancelled CAN messages due to no acknowledgement.
RX_OVERFLOW	WORD	Number of CAN messages that have not be processed because of an overflow of the reception buffer in the CAN controller.

Graphical illustration:



Time referenced behavior:



Function description:

After setting the *ENABLE* input, monitoring is active. This is displayed by the *CONFIRM* output.
The *TIMEOUT* output is set when no acknowledge is received by another bus node within 20 ms after transmission of a CAN message. The *MSG_TIME_OUT* variable specifies the number of missing acknowledges. This monitoring is only activated for CANopen messages.

The *TX_ERROR* output is set when a transmission error is detected. The number of events that lead to setting the variable can be read from the *BUS_ERROR_CNT* and *BUS_OFF_CNT* outputs.
After evaluating one or all error variables, they can be reset via the *RESET* input. Only the error variables are reset, i.e. the CAN controller is not reset. If the CAN bus is still affected, the error messages are displayed and incremented again.
The module is reset to the initial state by an edge change *ENABLE = FALSE*.

18.4.3 CAN_RX_11BIT_FRAME

With the CAN_RX_11BIT_FRAME function block, CAN messages are received in 11-bit identifier format.

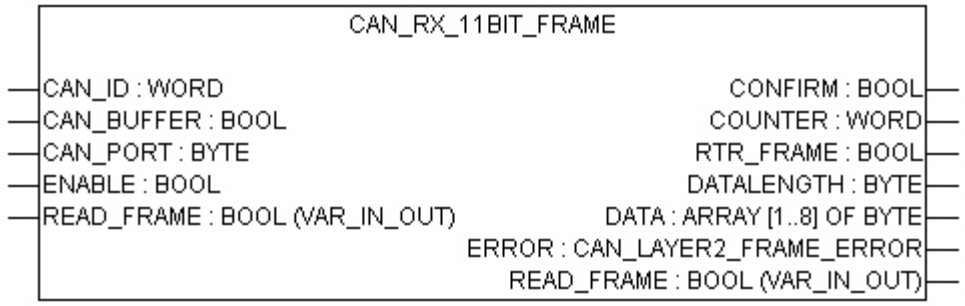
Category	CAN Layer 2 communication
Name	CAN_RX_11BIT_FRAME
Type	Function
Name of library	WAGO_CANLayer2_01.lib
Required libraries	
Applicable to	758-87x-112, 762-3xxx/000-001 or 762-3150/000-003

Input parameter		
Name	**Data Type**	**Description**
CAN_ID	WORD	Registration of the 11-bit identifier whose message should be received.
CAN_BUFFER	BOOL	Switches the internal buffer ON or OFF. **FALSE:** The last received CAN message is always output. **TRUE:** 8 messages can be cached in an internal buffer and read later sequentially (FIFO).
CAN_PORT	BYTE	Selection of the CAN interface **0:** CAN 0 interface activated **1:** CAN 1 interface activated For devices with an interface without function.
ENABLE	BOOL	Activation of the function

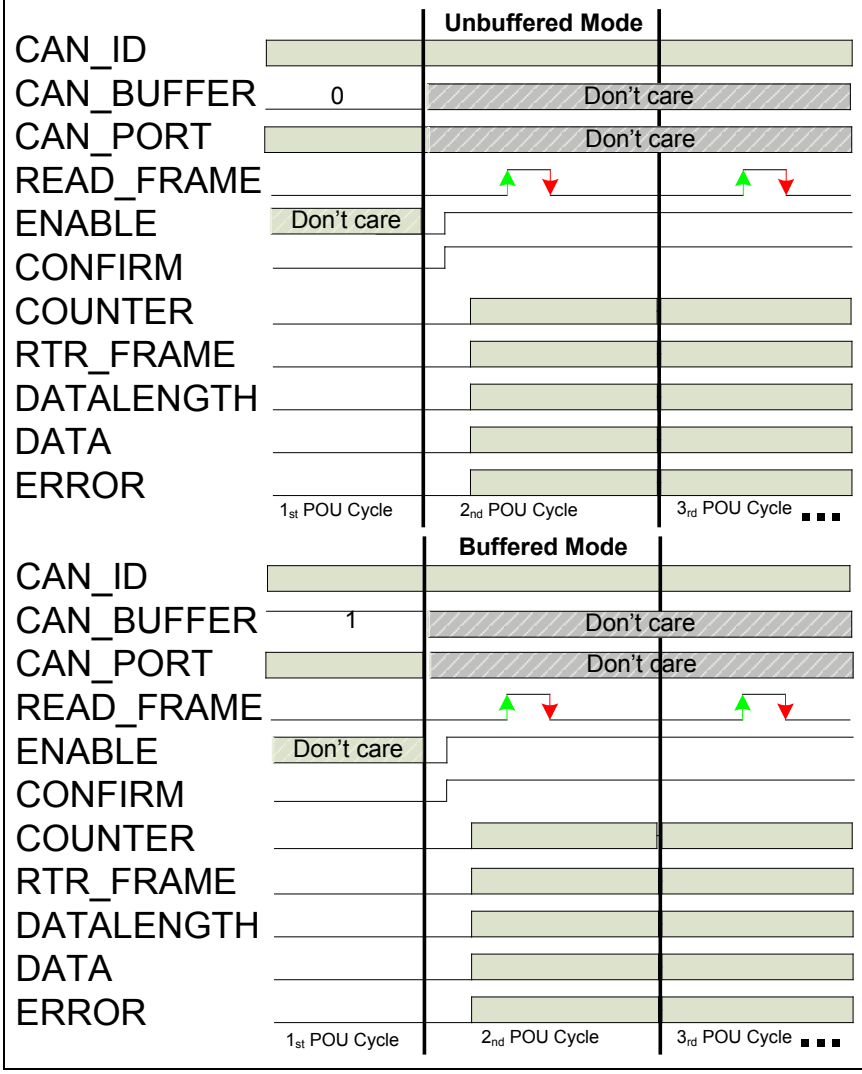
Output parameters		
Name	**Data Type**	**Description**
CONFIRM	BOOL	Execution confirmation
COUNTER	WORD	This variable is incremented each time a CAN message with registered CAN identifier is received.
RTR_FRAME	BOOL	Receipt of a remote frame.
DATALENGTH	BYTE	Data length of the message received.
DATA	ARRAY[1..8] OF BYTE	Contains the transmitted data of the CAN message. The number of bytes corresponds to the *DATALENGTH* output.
ERROR	CAN_LAYER2 _FRAME_ ERROR	See section Data Type: CAN_LAYER2_FRAME_ERROR

Input/output parameter		
Name	**Data Type**	**Description**
READ_FRAME	BOOL	Transfer of the CAN data with rising edge from the function block in the application. (***ENABLE*** = **TRUE**) & (***CAN_BUFFER*** = **FALSE**)**:** Output of the message last received. Automatic reset after copy process by the module (***ENABLE*** = **TRUE**) & (***CAN_BUFFER*** = **TRUE**)**:** Output of the oldest message in the FIFO buffer. Automatic reset after copy process by the module.

Graphical illustration



Time referenced behavior



Description
When *ENABLE = TRUE*, the CAN identifier is registered and the buffer mode set. *ENABLE = FALSE* can also be used to reset the settings and then to overwrite (*ENABLE = TRUE*) with new settings. The CAN identifier may not be changed for the timer interval *ENABLE = TRUE*. Changing CAN_PORT or CAN_BUFFER has no effect on the program sequence during this time interval. Max. 127 standard identifiers and 127 extended identifiers can be assigned in the program. **Control configuration**: In the control configuration, the CANopen master must be configured when using a purely CAN Layer 2 system. The baud rate is set here. Other slaves must not be configured in this operating mode. **Buffered mode (***CAN_BUFFER* = 1**):** In this mode, up to 8 CAN messages can be logically processed. If the module is activated via *ENABLE*, receipt of a new message is indicated by a incremented counter value. When the edge changes *READ_FRAME = TRUE*, the module is notified that the application is ready to process a message. When *READ_FRAME = FALSE*, a new message is saved by the module in the application memory. This change in status is initiated by the module. If the buffer overruns (*COUNTER* > 8), evaluating the *COUNTER* variable can determine how many messages could not longer be saved. This is an indicator of the CAN process load for the PLC cycle time. The saved telegrams in the buffer are invalid can cannot be read. Other telegrams from the CAN bus are no longer saved in the buffer. The module is reset to the initial state by an edge change *ENABLE = FALSE*. **Unbuffered mode (***CAN_BUFFER* = 0**):** In this mode, only the CAN message last received in the fixed time is output. If the module is activated via *ENABLE*, receipt of a new message is indicated by a incremented counter value. The counter value overflows at 65535. When the edge changes *READ_FRAME = TRUE*, the module is notified that the application is ready to process a message. When *READ_FRAME = FALSE*, a new message is saved by the module in the application memory. This change in status is initiated by the module. The module is reset to the initial state by an edge change *ENABLE = FALSE*.

18.4.4 CAN_RX_29BIT_FRAME

With the CAN_RX_29BIT_FRAME, CAN messages are received in 29-bit identifier format.

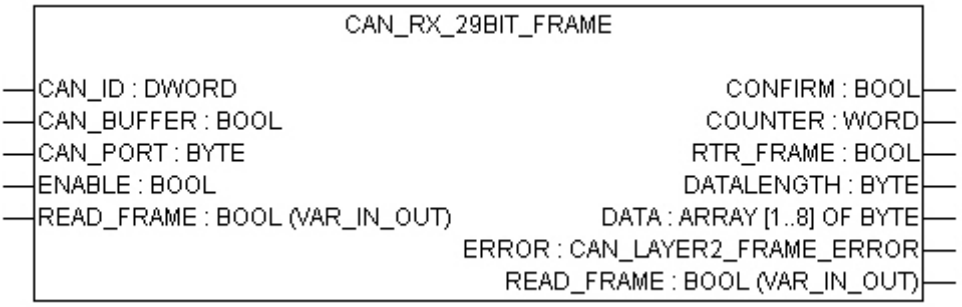
Category	CAN Layer 2 communication
Name	CAN_RX_29BIT_FRAME
Type	Function
Name of library	WAGO_CANLayer2_01.lib
Required libraries	
Applicable to	758-87x-112, 762-3xxx/000-001 or 762-3150/000-003

Input parameter		
Name	**Data Type**	**Description**
CAN_ID	DWORD	Registration of the 29-bit identifier whose message should be received.
CAN_BUFFER	BOOL	Switches the internal buffer ON or OFF. **FALSE:** The last received CAN message is always output. **TRUE:** 8 messages can be cached in an internal buffer and read later sequentially (FIFO).
CAN_PORT	BYTE	Selection of the CAN interface **0:** CAN 0 interface activated **1:** CAN 1 interface activated For devices with an interface without function.
ENABLE	BOOL	Activation of the function

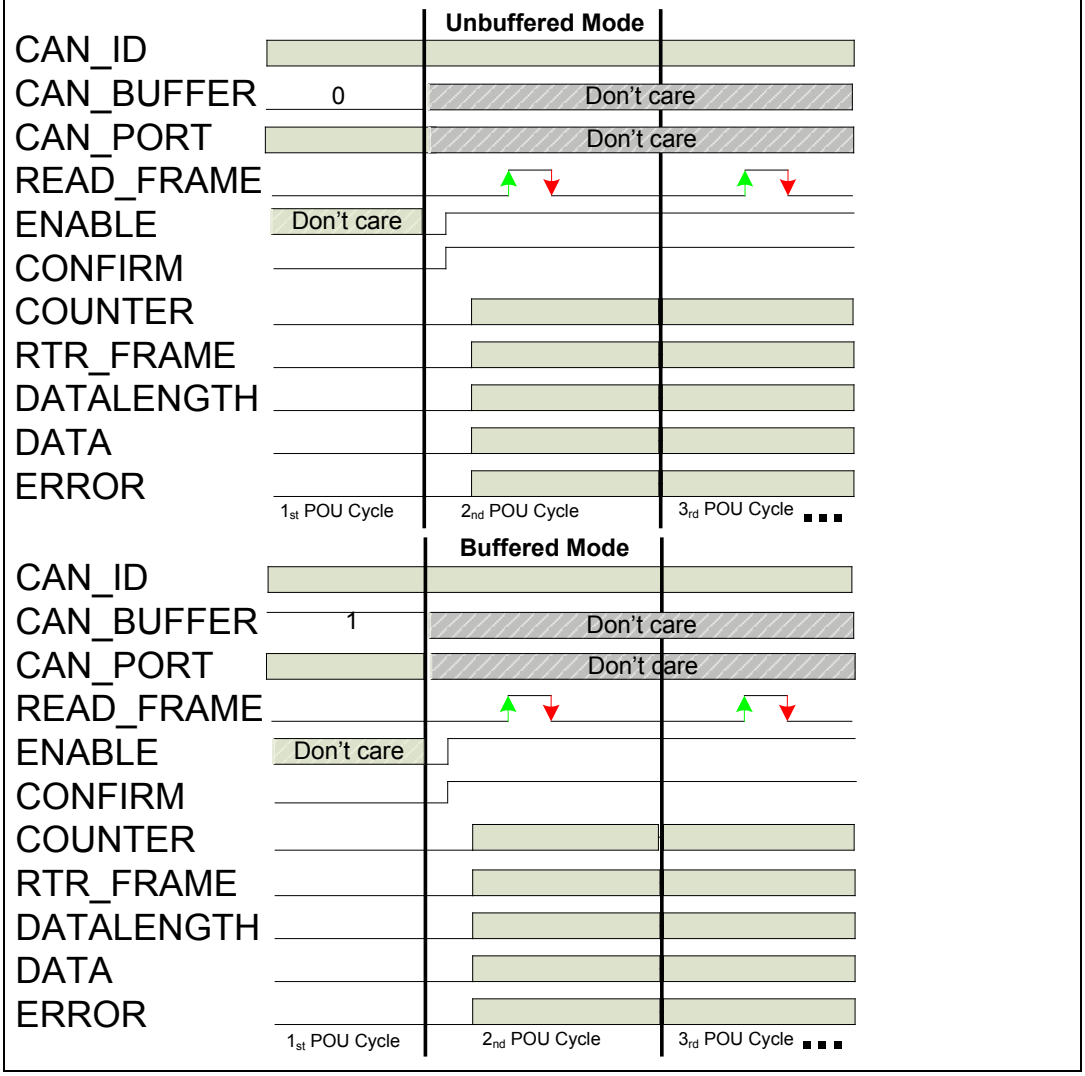
Output parameters		
Name	**Data Type**	**Description**
CONFIRM	BOOL	Execution confirmation
COUNTER	WORD	This variable is incremented each time a CAN message with registered CAN identifier is received.
RTR_ FRAME	BOOL	This variable is incremented each time a CAN message with registered CAN identifier is received.
DATALENGTH	BYTE	Receipt of a remote frame.
DATA	ARRAY [1..8] OF BYTE	Data length of the message received.
ERROR	CAN_ LAYER2_ FRAME_ ERROR	Contains the transmitted data of the CAN message. The number of bytes corresponds to the *DATALENGTH* output.

Input/output parameter		
Name	**Data Type**	**Description**
READ_FRAME	BOOL	Transfer of the CAN data with rising edge from the function block in the application. (*ENABLE* = TRUE) & (*CAN_BUFFER* = FALSE): Output of the message last received. Automatic reset after copy process by the module (*ENABLE* = TRUE) & (*CAN_BUFFER* = TRUE): Output of the oldest message in the FIFO buffer. Automatic reset after copy process by the module.

Graphical illustration



Time referenced behavior



Description
The CAN identifier is registered and buffer mode set in the first PLC cycle after the module has been activated with *ENABLE* = *TRUE*. The CAN identifier may not be changed as the program continues. Changing *CAN_PORT* or **CAN_BUFFER** has no effect on the program sequence. Max. 127 standard identifiers and 127 extended identifiers can be assigned in the program. **Control configuration**: In the control configuration, the CANopen master must be configured when using a purely CAN Layer 2 system. The baud rate is set here. Other slaves must not be configured in this operating mode. Buffered mode (*CAN_BUFFER* = 1): In this mode, up to 8 CAN messages can be logically processed. If the module is activated via *ENABLE*, receipt of a new message is indicated by a incremented counter value. When the edge changes *READ_FRAME* = *TRUE*, the module is notified that the application is ready to process a message. When *READ_FRAME* = *FALSE*, a new message is saved by the module in the application memory. This change in status is initiated by the module. If the buffer overruns (*COUNTER* > 8), evaluating the *COUNTER* variable can determine how many messages could not longer be saved. This is an indicator of the CAN process load for the PLC cycle time. The saved telegrams in the buffer are invalid can cannot be read. Other telegrams from the CAN bus are no longer saved in the buffer. The module is reset to the initial state by an edge change *ENABLE* = *FALSE*. **Unbuffered mode** (*CAN_BUFFER* = 0): In this mode, only the CAN message last received is output. If the module is activated via *ENABLE*, receipt of a new message is indicated by a incremented counter value. The counter value overflows at 65535. When the edge changes *READ_FRAME* = *TRUE*, the module is notified that the application is ready to process a message. When *READ_FRAME* = *FALSE*, a new message is saved by the module in the application memory. This change in status is initiated by the module. The module is reset to the initial state by an edge change *ENABLE* = *FALSE*.

18.4.5 CAN_TX_11BIT_FRAME

With the CAN_TX_11BIT_FRAME function block, CAN messages are transmitted in 11-bit identifier format.

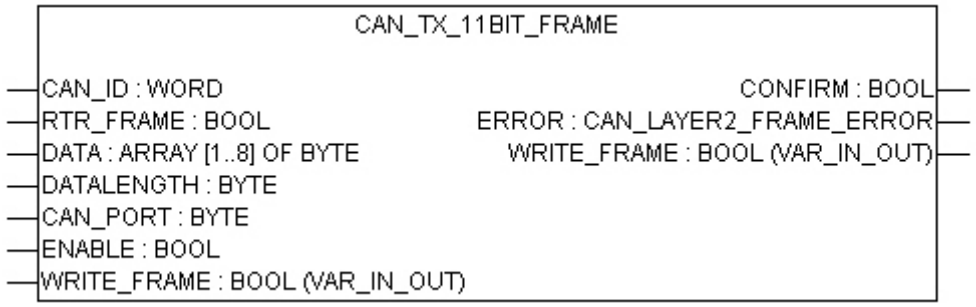
Category	CAN Layer 2 communication
Name	CAN_TX_11BIT_FRAME
Type	Function block
Name of library	WAGO_CANLayer2_01.lib
Required libraries	
Applicable to	758-87x-112, 762-3xxx/000-001 or 762-3150/000-003

Input parameter		
Name	**Data Type**	**Description**
CAN_ID	WORD	11-bit identifier of the CAN message to transmit.
RTR_FRAME	BOOL	The CAN message to transmit is a remote frame.
DATA	ARRAY[1..8] OF BYTE	Contains the data of the CAN message. The number of bytes corresponds to the *DATALENGTH* input.
DATALENGTH	BYTE	Data length of the CAN message to transmit.
CAN_PORT	BYTE	Selection of the CAN interface **0:** CAN 0 interface activated **1:** CAN 1 interface activated For devices with an interface without function.
ENABLE	BOOL	Activation of the function

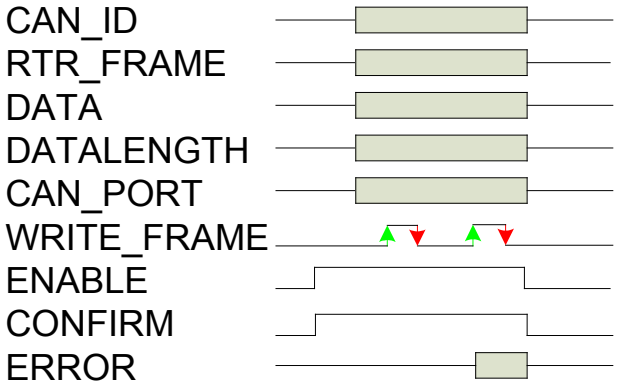
Output parameters		
Name	**Data Type**	**Description**
CONFIRM	BOOL	A CAN message has been transmitted.
ERROR	CAN_LAYER2_FRAME_ERROR	See section Data Type: CAN_LAYER2_FRAME_ERROR

Input/output parameter		
Name	**Data Type**	**Description**
WRITE_FRAME	BOOL	Transmission of the CAN message is initiated.

Graphical illustration



Time referenced behavior



Description

The function block is activated after setting the *ENABLE* input.
A CAN message is then transmitted with an edge change FALSE to TRUE of the *WRITE_FRAME* input. This is formed from the values at the inputs: *CAN_ID*, *RTR_FRAME*, *DATA* and *DATALENGTH*.
After the message is transmitted, the IN/OUT variable of the *WRITE_FRAME* is reset to FALSE.

The status of the CAN transmission is displayed in the *ERROR* output parameter. The I/O-IPC does not set the CAN_SEND_ERROR output. Error-free bus communication is identified by evaluated the CAN_ERROR_INFO module.
By an edge change TRUE to FALSE of the *ENABLE* input, the function blog is reset to the initial state.

18.4.6 CAN_TX_29BIT_FRAME

With the CAN_TX_29BIT_FRAME function block, CAN messages are transmitted in 29-bit identifier format.

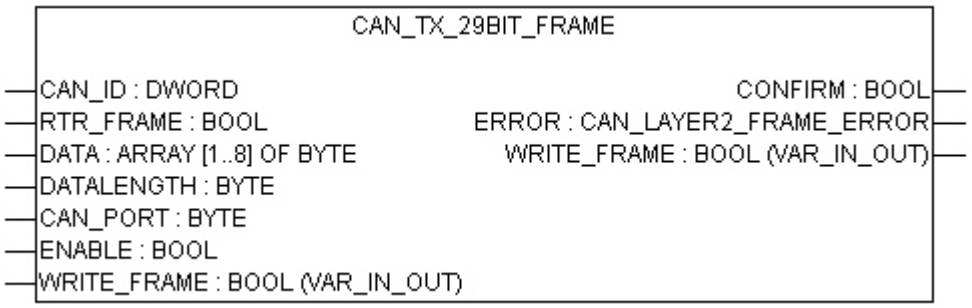
Category	CAN Layer 2 communication
Name	CAN_TX_29BIT_FRAME
Type	Function
Name of library	WAGO_CANLayer2_01.lib
Required libraries	
Applicable to	758-87x-112, 762-3xxx/000-001 or 762-3150/000-003

Input parameter		
Name	**Data Type**	**Description**
CAN_ID	WORD	29-bit identifier of the CAN message to transmit.
RTR_FRAME	BOOL	The CAN message to transmit is a remote frame.
DATA	ARRAY[1..8] OF BYTE	Contains the data of the CAN message. The number of bytes corresponds to the *DATALENGTH* input.
DATALENGTH	BYTE	Data length of the CAN message to transmit.
CAN_PORT	BYTE	Selection of the CAN interface **0:** CAN 0 interface activated **1:** CAN 1 interface activated For devices with an interface without function.
ENABLE	BOOL	Activation of the function

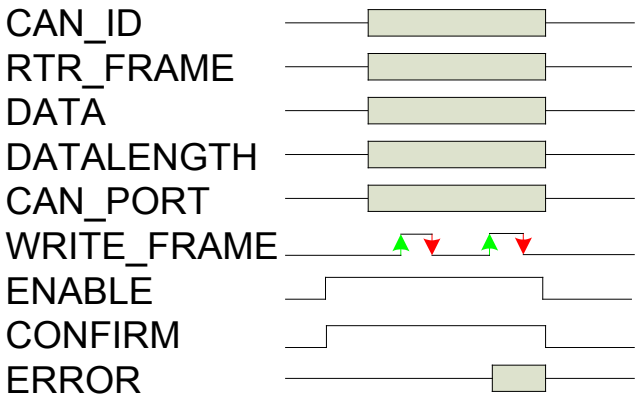
Output parameters		
Name	**Data Type**	**Description**
CONFIRM	BOOL	A CAN message has been transmitted.
ERROR	CAN_LAYER2_ FRAME_ ERROR	See section Data Type: CAN_LAYER2_FRAME_ERROR

Input/output parameter		
Name	**Data Type**	**Description**
WRITE_FRAME	BOOL	Transmission of the CAN message is initiated.

Graphical illustration



Time referenced behavior



Description

The function block is activated after setting the *ENABLE* input.
A CAN message is then transmitted with an edge change FALSE to TRUE of the *WRITE_FRAME* input. This is formed from the values at the inputs: *CAN_ID*, *RTR_FRAME*, *DATA* and *DATALENGTH*.
After the message is transmitted, the IN/OUT variable of the *WRITE_FRAME* is reset to FALSE.
The status of the CAN transmission is displayed in the *ERROR* output parameter.
The IPC does not set the CAN_SEND_ERROR output. Error-free bus communication is identified by evaluated the CAN_ERROR_INFO module.
By an edge change TRUE to FALSE of the *ENABLE* input, the function blog is reset to the initial state.

18.4.7 CAN_LAYER2_FRAME_ERROR

Category	CAN Layer 2 communication
Name	CAN_TX_29BIT_FRAME
Type	Data Type
Name of library	WAGO_CANLayer2_01.lib
Applicable to	758-87x-112, 762-3xxx/000-001 or 762-3150/000-003
Structure	TYPE CAN_LAYER2_FRAME_ERROR : (POU_NOT_ENABLED, CAN_BUS_IDLE, CAN_PORT_WRONG, CAN_PORT_BUSY, CAN_SEND_OK, CAN_RECEIVE_OK, CAN_SEND_ERROR, CAN_ID_ERROR, CAN_DATALENGTH_ERROR, CAN_RECEIVE_BUFFER_ERROR, CAN_REGISTER_ERROR); END_TYPE

Element	Value	Description
POU_NOT_ENABLED	*0*	The module is not enabled.
CAN_BUS_IDLE	*1*	No CAN frame is currently being processed.
CAN_PORT_WRONG	*2*	A CAN port has been selected that is not available to this device. For devices with a CAN port, the *CAN_PORT* input is "don't care". The error is not generated.
CAN_PORT_BUSY	*3*	The selected CAN interface is not available for CAN Layer 2 communication by CANopen slave configuration. Only applies to PERSPECTO devices.
CAN_SEND_OK	*4*	A CAN frame was been sent successfully.
CAN_RECEIVE_OK	*5*	A CAN frame has been received successfully.
CAN_SEND_ERROR	*6*	An error has occurred when sending a CAN frame.
CAN_ID_ERROR	*7*	The identifier exceeds the 11 or 29-bit range or more than 127 identifiers have been registered.
CAN_DATALENGTH_ERROR	*8*	When sending CAN frames (11 and 29-bit), the permitted range of 8 bites has been exceeded.
CAN_RECEIVE_BUFFER_ERROR	*9*	There has been an overflow of the internal CAN buffer. This suggests that the task cycle time of the PLC program for processing CAN frames is too big or the CAN bus load is too high.

CAN_REGISTER_ERROR	*10*	The registration range of 127 CAN frames has been exceeded.

Description
This enumeration specifies all fault conditions that can occur when receiving or sending CAN messages.

18.5 mod_com.lib

You can find information about the library under www.wago.com in area “Downloads”.

Note

The following functions are not implemented:
- SET_DIGITAL_INPUT_OFFSET
- SET_DIGITAL_OUTPUT_OFFSET
- WRITE_OUTPUT_BIT
- READ_OUTPUT_BIT
- READ_INPUT_BIT
- WRITE_OUTPUT_WORD
- READ_OUTPUT_WORD
- READ_INPUT_WORD

18.6 SerComm.lib

You can find information about the library under www.wago.com in area “Downloads”.

18.7 WagoLibTerminalDiag.lib

You can find information about the library under www.wago.com in area “Downloads”.

18.8 WagoLibKBUS.lib

By using the WagoLibKBUS library, you can achieve the task-synchronous, consistent access to process data via function blocks.

You can find information about the library under www.wago.com in area “Downloads”.

18.9 SysLibCom.lib

By using the SysLibCom library, you can connect an additional RS-232 interface at the I/O-IPC using the WAGO USB-to-Serial Adapter (761-9005). The adapter connected via the USB interface is reached via COM3(=3).

You can obtain additional information about the SysLibCom.lib library in the CODESYS online help.

18.10 SysLibFile, SysLibDir, SysLibFileAsync

The file system can be accessed in CODESYS using the following CODESYS libraries: SysLibFile, SysLibDir, and SysLibFile_Async

The following directories can be used for this:

1. Memory on the boot medium: home/CODESYS/
2. Memory on a CF card/USB flash drive formatted using FAT: media/<partitionname>/
3. Volatile memory (RAM disk): tmp/

Example:

h_file:=SysFileOpen(media/USBNAME1/data.log', 'a');

The partition name of CF cards/USB flash drives can be assigned using the WBM/Configtool during formatting. (see chapter “Configuration via Web-Based Management (WBM)”).
In addition, the CF card/USB flash drive can also be formatted with a different operating system. The partition name can also be assigned then.

Example using WinXP:

Figure 132: Formatting partname

Note

Additional information
Additional information about SysLibFile.lib, SysLibDir and SysLibFileAsync can be found in the CODESYS online help.

List of Figures

List of Tables

WAGO Kontakttechnik GmbH & Co. KG
Postfach 2880 • D-32385 Minden
Hansastraße 27 • D-32423 Minden
Phone: +49/5 71/8 87 – 0
Fax: +49/5 71/8 87 – 1 69
E-Mail: info@wago.com

Internet: http://www.wago.com