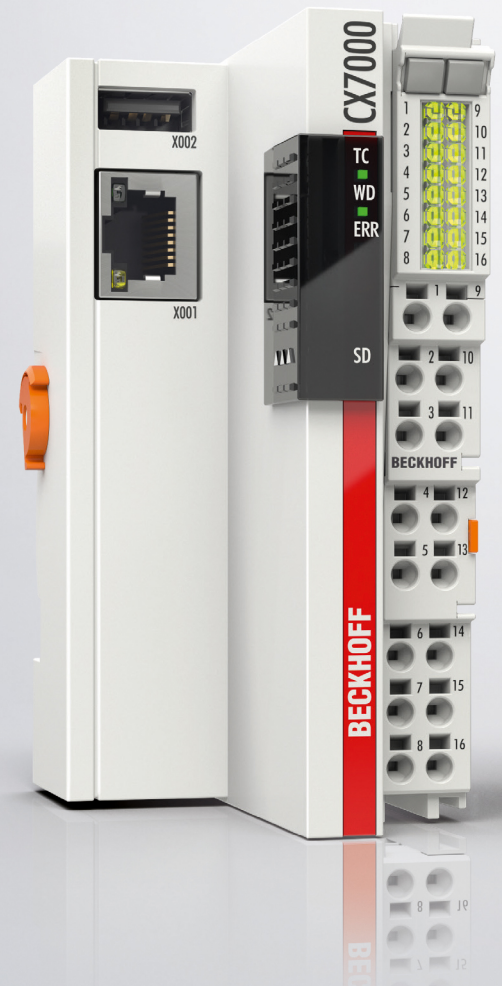
BECKHOFF New Automation Technology

Manual | EN

CX7000

Embedded PC



2025-03-11 | Version: 1.5

Table of contents

1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS®, and XPlanar® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.1 Representation and structure of warnings

The following warnings are used in the documentation. Read and follow the warnings.

Warnings relating to personal injury:

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that can result in minor injury.

Warnings relating to damage to property or the environment:

NOTICE
There is a potential hazard to the environment and equipment.

Notes showing further information or tips:

This notice provides important information that will be of assistance in dealing with the product or software. There is no immediate danger to product, people or environment.

1.2 Documentation issue status

Version	Comment
1.0	First version.
1.1	UL notes added. Chapter "Function blocks" for LEDs added. Composition of the hostname revised. Chapter "TwinCAT" revised
1.2	Technical data adapted for counter and encoder mode.
1.3	Connection cross-section adapted for multi-function I/Os. Chapters "CoE access to multi-function I/Os" and "Diagnosis of multi-function I/Os" revised.
1.4	New firmware with TF6701 TwinCAT 3 IoT Communication (MQTT), and TF6730 TwinCAT 3 IoT Communicator.
1.5	Description of VLAN, NTP provider, and I/O Idle Task added.

2 For your safety

Read the chapter on safety and follow the instructions in order to protect from personal injury and damage to equipment.

Limitation of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Unauthorized modifications and changes to the hardware or software configuration, which go beyond the documented options, are prohibited and nullify the liability of Beckhoff Automation GmbH & Co. KG.
In addition, the following actions are excluded from the liability of Beckhoff Automation GmbH & Co. KG:

- Failure to comply with this documentation.
- Improper use.
- Use of untrained personnel.
- Use of unauthorized replacement parts.

2.1 Intended use

The embedded PC is a control system for use in machine and system engineering for automation, visualization and communication. The embedded PC is designed for installation in a control cabinet or terminal box and is used together with Bus or EtherCAT Terminals to receive digital and analog signals from sensors and output them to actuators or forward them to higher-level controllers.

The Embedded PC is designed for a working environment that meets the requirements of protection class IP20. This involves finger protection and protection against solid foreign objects up to 12.5 mm, but not protection against water. Operation of the devices in wet and dusty environments is not permitted, unless specified otherwise. The specified limits for electrical and technical data must be adhered to.

Improper use

The Embedded PC is not suitable for operation in the following areas:

- Potentially explosive atmospheres.
- Areas with an aggressive environment, e.g. aggressive gases or chemicals.
- Living areas. If the devices are to be used in living areas, the relevant standards and guidelines for interference emissions must be adhered to, and the devices must be installed in housings or control boxes with suitable shielding.

2.2 Staff qualification

All operations involving Beckhoff software and hardware may only be carried out by qualified personnel with knowledge of control and automation engineering. The qualified personnel must have knowledge of the administration of the Industrial PC and the associated network.

All interventions must be carried out with knowledge of control programming, and the qualified personnel must be familiar with the current standards and guidelines for the automation environment.

2.3 Safety instructions

The following safety instructions must be followed during installation and working with networks and the software.

Mounting

- Never work on live equipment. Always switch off the power supply for the device before installation, troubleshooting or maintenance. Protect the device against unintentional switching on.

- Observe the relevant accident prevention regulations for your machine (e.g. the BGV A 3, electrical systems and equipment).
- Ensure standard-compliant connection and avoid risks to personnel. Ensure that data and supply cables are laid in a standard-compliant manner and ensure correct connection.
- Observe the relevant EMC guidelines for your application.
- Avoid polarity reversal of the data and supply cables, as this may cause damage to the equipment.
- The devices contain electronic components, which may be destroyed by electrostatic discharge when touched. Observe the safety precautions against electrostatic discharge according to DIN EN 61340-5-1/-3.

Working with networks

- Restrict access to all devices to an authorized circle of persons.
- Change the default passwords to reduce the risk of unauthorized access.
- Protect the devices with a firewall.
- Apply the IT security precautions according to IEC 62443, in order to limit access to and control of devices and networks.

2.4 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

3 Transport and storage

Transport

NOTICE

Short circuit due to moisture

Moisture can form during transport in cold weather or in the event of large temperature fluctuations.

Avoid moisture formation (condensation) in the embedded PC, and leave it to adjust to room temperature slowly. If condensation has occurred, wait at least 12 hours before switching on the embedded PC.

Despite the robust design of the unit, the components are sensitive to strong vibrations and impacts. Transporting a control cabinet with a built-in embedded PC can result in excessive impact on the embedded PC.

- During transport, the device must therefore be protected from excessive mechanical stress.
- Appropriate packaging of the industrial PC, in particular the original packaging, can improve vibration resistance during transport.
- Send the embedded PC in the original packaging and additional outer packaging.

Table 1: Dimensions and weight.

	CX7000
Dimensions (W x H x D)	49 mm x 100 mm x 73 mm
Weight	approx. 142 g

Storage

- Store the Embedded PC in the original packaging.

4 Product overview

The CX7000 embedded PC has an Arm® Cortex®-M7 single-core processor running at 480 MHz and the following basic configuration:

- a microSD card slot with integrated 512 MB microSD card,
- an Ethernet interface (10/100 Mbit/s, RJ45),
- a USB interface (max. 12 Mbit/s, max. 100 mA),
- integrated multi-function I/Os.

The CX7000 is programmed with TwinCAT 3 via the Ethernet interface. In addition, the Beckhoff Device Manger is available as a web interface for configuring the CX7000.

Multi-function I/Os

Special features of the CX7000 series are the eight integrated multifunction inputs and four integrated multifunction outputs.

- 8 digital inputs, 24 V DC, filter 3 ms, type 3, 1-wire technique
- 4 digital outputs, 24 V DC, 0.5 A, 1-wire technique

The integrated multifunction I/Os of the CX7000 can be configured via TwinCAT 3 for other operation modes in order to enable fast counting or the processing of analog values:

- Counter mode: 1 x digital counter input 100 kHz, 1 x digital input for up/down counter 20 kHz, 2 x digital counter outputs
- Incremental encoder mode: 2 x digital inputs for 250 kHz encoder signal (A/B input), 2 x digital encoder output
- Analog signal mode: 2 x digital inputs configured as analog inputs 0 to 10 V, 12-bit resolution with 16-bit representation
- PWM signal mode: 2 x digital outputs configured for PWM signal, 15 Hz…100 kHz

Power supply terminal

EtherCAT Terminals (E-bus) or bus terminals (K-bus) can optionally be connected directly on the right-hand side; the CX7000 automatically recognizes which system is connected during the start-up phase. If further electrical signals are to be processed, the CX7000 can be extended as required and extremely flexibly by EtherCAT Terminals or bus terminals in addition to the integrated I/Os.

Firmware

The real-time operating system TwinCAT/RTOS, which is based on FreeRTOS, is used as the operating system or firmware. Note that TwinCAT/RTOS is a closed system and you cannot install your own software. This provides a certain level of security, as third-party software such as viruses or similar cannot be installed and the CX7000 can be connected to a network. The CX7000 can be used from TwinCAT 3.1 Build 4024.12. The following TwinCAT functions are included and licensed:

- TC1000 TC3 ADS
- TC1100 TC3 IO
- TC1200 TC3 PLC
- TF4100 TC3 Controller Toolbox
- TF4110 TC3 Temperature Controller
- TF6255 TC3 Modbus-RTU
- TF6340 TC3 Serial Communication
- TF6701 | TwinCAT 3 IoT Communication (MQTT)*)
- TF6730 | TwinCAT 3 IoT Communicator*)

*) Image version 114606 and TwinCAT 3 XAE 4024.47 or higher required.

The open source licenses are available as a ZIP file on the microSD card.

4.1 Structure

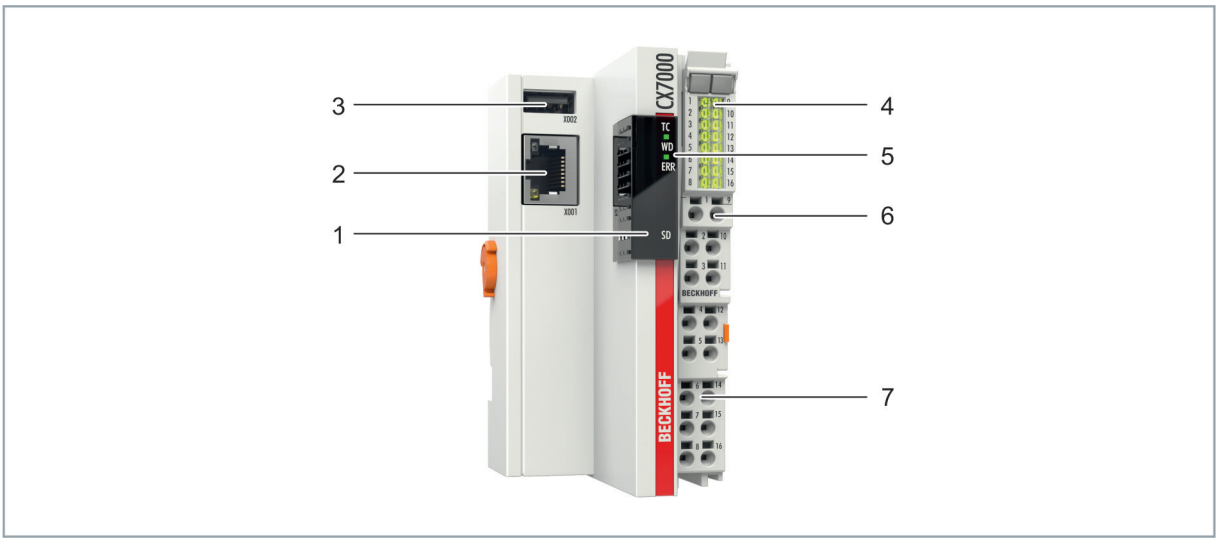


Fig. 1: Sample configuration of a CX7000 Embedded PC.

Table 2: Legend for the configuration of the basic CPU module

No.	Component	Description
1	MicroSD card slot (under the cover).	Slot for industrial MicroSD cards. Memory space for firmware and TwinCAT 3 projects.
2	Ethernet interface (X001)	For the connection to local networks. Serves as a programming interface.
3	USB interface (X002)	Interface for additional USB data storage device.
4	I/O Status LEDs	Diagnosis of the power supply for the Embedded PC and the terminal bus. Status of the E-bus or K-bus communication and multifunction I/Os.
5	Diagnostic LEDs	1 x TwinCAT Status, 1 x Flash access, 1 x Error LED.
6	Spring-loaded terminals, +24 V and 0 V	Power supply (Us) for Embedded PC.
7	Spring-loaded terminals, +24 V and 0 V	Power supply (Up) for integrated multifunction I/Os and Bus Terminals via the power contacts.

4.2 Name plate

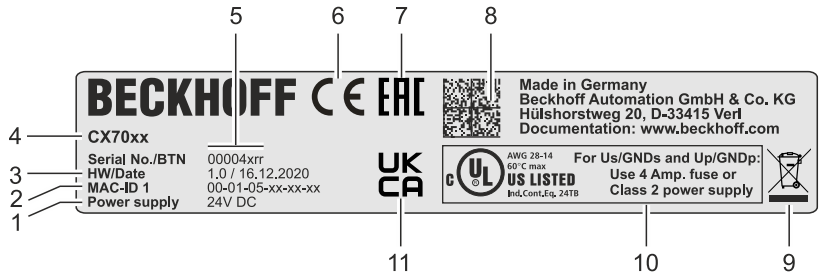


Fig. 2: Name plate example.

Table 3: Information on the name plate.

No.	Description
1	Power supply 24 V DC.
2	MAC addresses of the built-in Ethernet interface.
3	Hardware version and date of manufacture.
4	Product designation for identification of the embedded PC.
5	Serial number/Beckhoff Traceability Number (BTN) for clear identification of the product. The host name is formed from BTN and the serial number/Beckhoff Traceability Number (BTN). Example: the BTN 00004xrr results in the host name **BTN-00004xrr**.
6	CE marking
7	EAC marking
8	Machine-readable information in the form of a Data Matrix Code (DMC, code scheme ECC200) that can be used for better identification and management.
9	Marking for garbage disposal. Do not dispose of this product with household waste.
10	UL marking with prescribed information on power supply, fuse, temperature, and cable cross-sections.
11	UKCA marking

4.3 Ethernet interface (X001)

You can program and commission the CX7000 Embedded PC via the X001 Ethernet interface. The Ethernet interface achieves speeds of 10 / 100 Mbit/s.

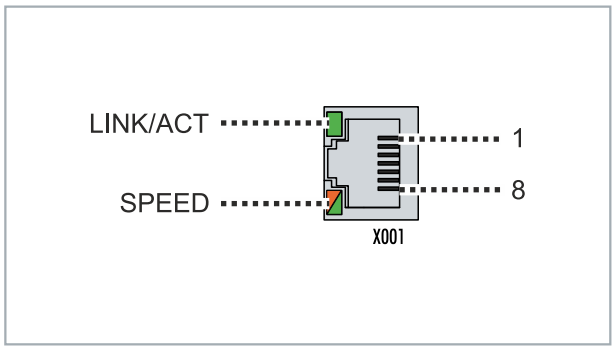


Fig. 3: Ethernet interface X001.

The LEDs on the left of the interface indicate the connection status. The upper LED (LINK/ACT) indicates whether the interface is connected to a network. If this is the case, the LED lights up green. The LED flashes when data transfer on the interface is in progress.

The lower LED (SPEED) indicates the connection speed. The LED is not lit if the speed is 10 Mbit/s. At 100 Mbit/s the LED lights up orange.

Table 4: Ethernet interface X001, pin assignment.

PIN	Signal	Description
1	TD +	Transmit +
2	TD -	Transmit -
3	RD +	Receive +
4	connected	reserved
5		
6	RD -	Receive -
7	connected	reserved
8		

Transmission standards

10Base5

The transmission medium for 10Base5 consists of a thick coaxial cable ("yellow cable") with a max. data transfer rate of 10 Mbaud arranged in a line topology with branches (drops) each of which is connected to one network device. Because all the devices are in this case connected to a common transmission medium, it is inevitable that collisions occur often in 10Base5.

10Base2

10Base2 (Cheaper net) is a further development of 10Base5, and has the advantage that the coaxial cable is cheaper and, being more flexible, is easier to lay. It is possible for several devices to be connected to one 10Base2 cable. It is frequent for branches from a 10Base5 backbone to be implemented in 10Base2.

10BaseT

Describes a twisted pair cable for 10 Mbaud. The network here is constructed as a star. It is no longer the case that every device is attached to the same medium. This means that a broken cable no longer results in failure of the entire network. The use of switches as star couplers enables collisions to be reduced. Using full-duplex connections they can even be entirely avoided.

100BaseT

Twisted pair cable for 100 Mbaud. It is necessary to use a higher cable quality and to employ appropriate hubs or switches in order to achieve the higher data rate.

10BaseF

The 10BaseF standard describes several optical fiber versions.

Short description of the 10BaseT and 100BaseT cable types

Twisted-pair copper cable for star topologies, where the distance between two devices may not exceed 100 meters.

UTP

Unshielded twisted-pair
This type of cable belongs to category 3, and is not recommended for use in an industrial environment.

S/UTP

Screened/unshielded twisted-pair (shielded with copper braid)
Has an overall shield of copper braid to reduce influence of external interference. This cable is recommended for use with Bus Couplers.

FTP

Foiled shielded twisted-pair (shielded with aluminum foil)
This cable has an outer shield of laminated aluminum and plastic foil.

S/FTP

Screened/foiled shielded twisted-pair (shielded with copper braid and aluminum foil)
Has a laminated aluminum shield with a copper braid on top. Such cables can provide up to 70 dB reduction in interference power.

STP

Shielded twisted-pair
Describes a cable with overall shielding without further specification of the type of shielding.

S/STP

Screened/shielded twisted-pair (wires are individually shielded)
This identification refers to a cable with a shield for each of the two wires as well as an outer shield.

ITP

Industrial Twisted-Pair
The structure is similar to that of S/STP, but, in contrast to S/STP, it has only two pairs of conductors.

4.4 USB interface (X002)

A USB flash drive can be connected to the USB interface and used as an additional memory. The USB interface supports transfer speeds of up to 12 Mbit/s and no more than 100 mA. The file is accessed from TwinCAT or the PLC program with the help of the associated function blocks. No other devices can be connected to the USB interface and used.

The same functional mode can be used for accessing files on the MicroSD card. Use *C:*\ as the drive letter for accessing the MicroSD card and *D:*\ for accessing the USB flash drive.

Function blocks for data access

The function blocks can be used to process files from the PLC locally on the PC. The TwinCAT target system is identified by the AMS network address. This mechanism makes it possible, amongst other things, to store or to edit files on other TwinCAT systems in the network. Access to files consists of three sequential phases:

1. Opening the file.
2. Read or write access to the opened file.
3. Closing the file.

Opening the file has the purpose of establishing a temporary connection between the external file, whose name is all that initially is known, and the running program. Closing the file has the purpose of indicating the end of the processing and placing it in a defined output state for processing by other programs.

Name	Description
FB_EOF	Check the end of file
FB_FileOpen	Open a file
FB_FileClose	Close a file
FB_FileGets	Get string from a file
FB_FilePuts	Put string to a file
FB_FileRead	Read from a file
FB_FileWrite	Write to a file
FB_FileSeek	Move the file pointer
FB_FileTell	Get the file pointer position
FB_FileDelete	Delete a file
FB_FileRename	Rename a file
FB_CreateDir	Create new directory
FB_RemoveDir	Remove directory

Requirements

Development environment	Target system type	PLC libraries to include (Category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_System (System)

4.5 MicroSD card

The basic equipment of the CX7000 includes a 512 MB microSD card. You can optionally order the embedded PC with a larger 16 GB microSD card.

The cards employed are SLC memory with extended temperature range for industrial applications. Use exclusively microSD cards approved by Beckhoff.

Order identifier	Capacity	Description
CX1900-0131	16 GB	microSD card (SLC memory) with extended temperature range for industrial applications instead of the 512 MB card (ordering option)

Order identifier	Capacity	Description
CX1900-0122	512 MB	microSD card (SLC memory) with extended temperature range for industrial applications as spare part.
CX1900-0132	16 GB	

5 Commissioning

5.1 Mounting

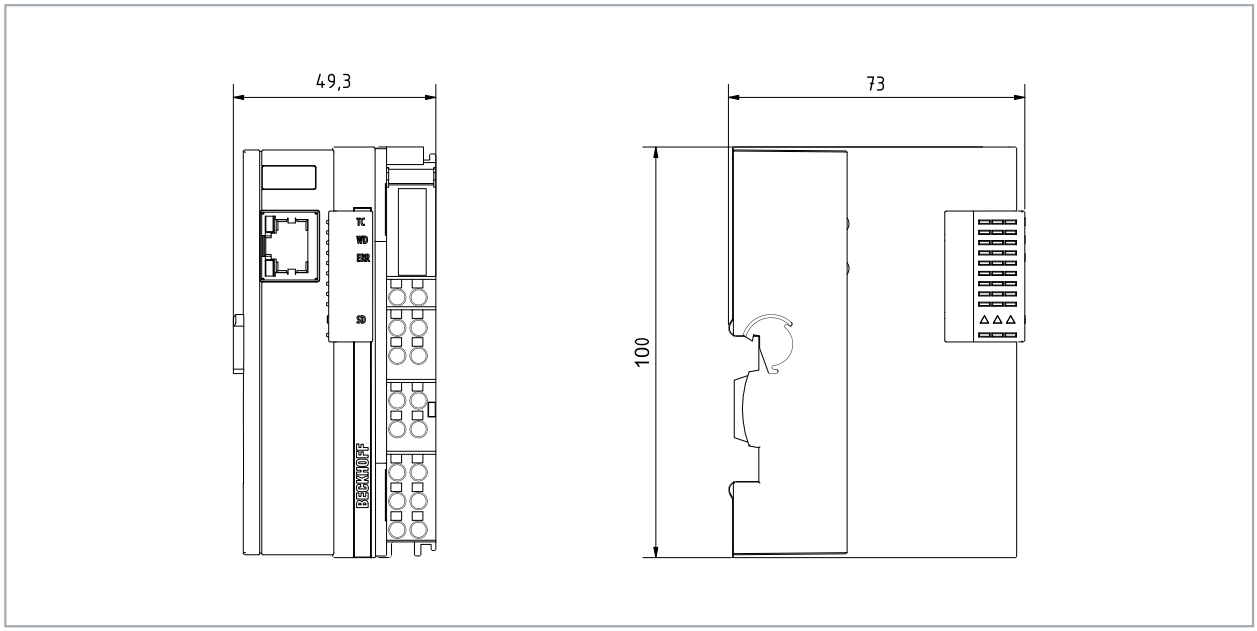


Fig. 4: CX70xx Embedded PC, dimensions.

5.1.1 Note the permissible installation positions

NOTICE

Overheating

The Embedded PC may overheat if the installation position is incorrect or the minimum distances are not adhered to. Adhere to the maximum ambient temperature of 60°C and the mounting instructions.

Install the Embedded PC horizontally in the control cabinet on a DIN rail, in order to ensure optimum heat dissipation.

Note the following specifications for the control cabinet:

- The Embedded PC should only be operated at ambient temperatures between -25 °C and 60 °C. Measure the temperature below the Embedded PC at a distance of 30 mm to the cooling fins, in order to determine the ambient temperature correctly.
- Adhere to the minimum distances of 30 mm above and below the Embedded PC.
- Additional electrical equipment affects the heat generation in the control cabinet. Select a suitable control cabinet enclosure depending on the application, or ensure that excess heat is dissipated from the control cabinet.

The Embedded PC must be mounted horizontally on the DIN rail. Ventilation openings are located at the top and bottom of the housing. This ensures an optimum airflow through the Embedded PC in vertical direction. In addition, a minimum clearance of 30 mm above and below the Embedded PC is required, in order to ensure adequate ventilation.

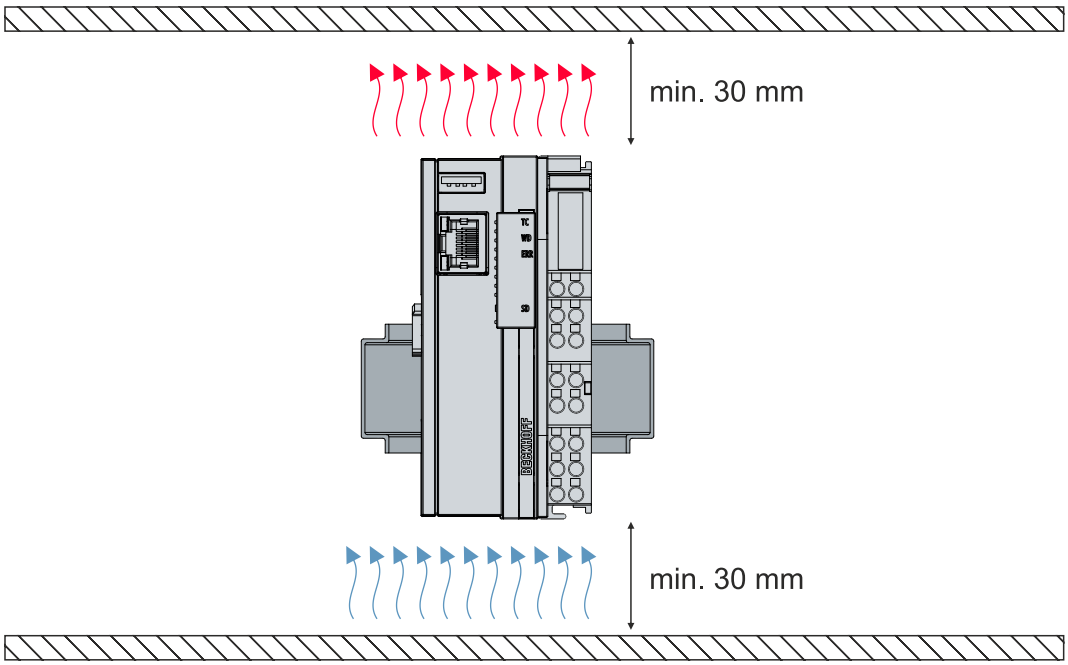


Fig. 5: CX70xx Embedded PC, permissible installation position.

If vibrations and impact occur in the same direction as the DIN rail, the Embedded PC must be secured with an additional bracket, in order to prevent it slipping.

Installation positions with reduced temperature range up to 45 °C

You can also mount the Embedded PC vertically or horizontally on the mounting rail. Note that you can then only operate the Embedded PC up to an ambient temperature of 45 °C.

Ensure that Bus Terminals that are connected to the Embedded PC are designed for operation in vertical or horizontal position.

Restrictions for E-bus/K-bus current

The maximum E-bus/K-bus current varies depending on the selected installation position and the ambient temperature.

Table 5: Maximum E-bus/K-bus current depending on the selected installation position and the ambient temperature.

E-bus/K-bus current	Installation position	Ambient temperature
max. 1.5 A	variable	-25…45 °C
max. 1.3 A	horizontal	-25…55 °C
max. 1 A	variable	-25…55 °C
max. 1 A	horizontal	-25…60 °C

5.1.2 Fastening to the DIN rail

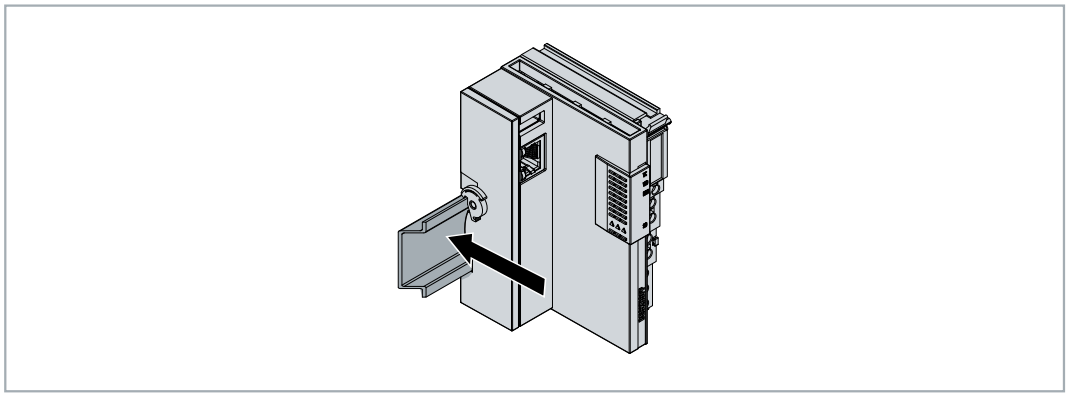
The housing is designed such that the Embedded PC can be pushed against the DIN rail and latched onto it.

Requirements:

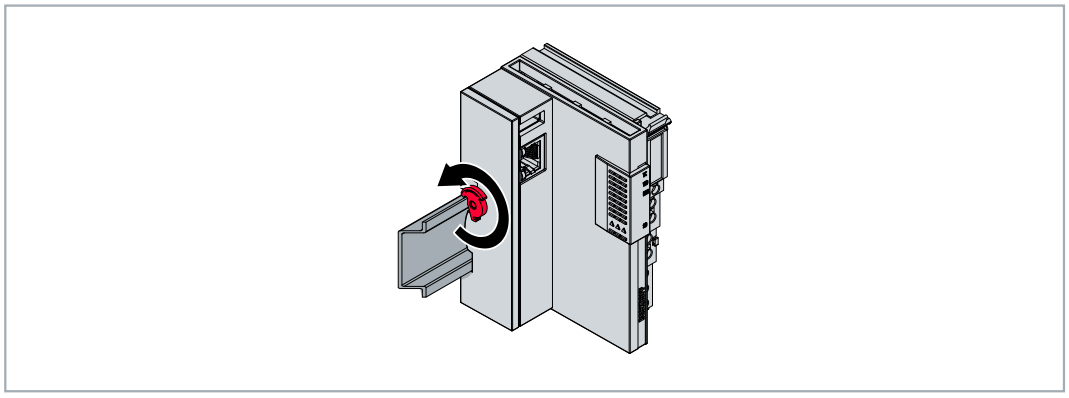
- DIN rail of the type TS35/7.5 or TS35/15 according to EN 60715.

Fasten the Embedded PC to the DIN rail as follows:

1. Place the Embedded PC on the DIN rail. Slightly press the Embedded PC onto the DIN rail until a soft click can be heard and the Embedded PC has latched.



2. Subsequently, lock the catch on the left side of the Embedded PC.
3. Turn the latch counter clockwise until the latch quietly clicks and engages.



⇨ You have installed the Embedded PC successfully. Check again that the mounting is correct and that the Embedded PC is engaged on the DIN rail.

5.1.3 Changing the MicroSD card

Loss of data

MicroSD cards are subjected to heavy load during operation and have to withstand many write cycles and extreme ambient conditions. MicroSD cards from other manufacturer may fail, resulting in data loss.

Only use industrial MicroSD cards provided by Beckhoff.

The MicroSD card slot is intended for an industrially compatible MicroSD card. The firmware of the Embedded PC is stored on the MicroSD card. If necessary, the MicroSD card can be written to from TwinCAT 3, allowing user-defined data to be stored.

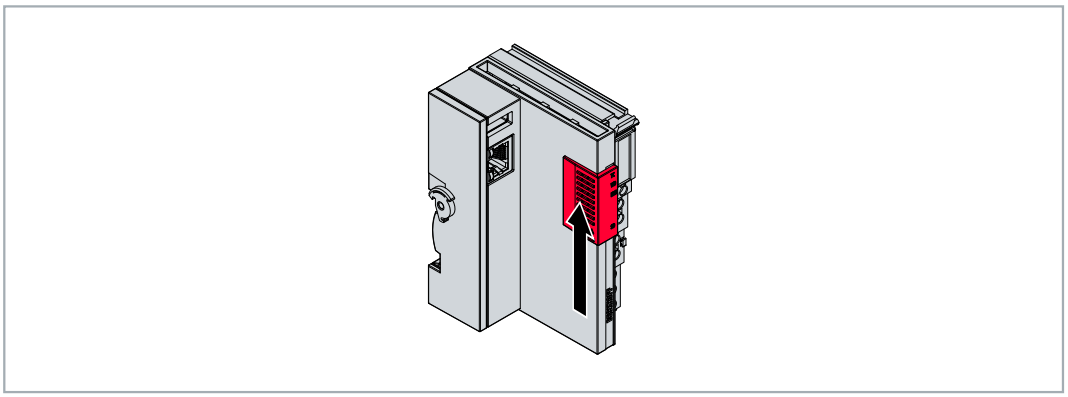
The eject mechanism is based on the push/push principle. Below, we show you how to change the MicroSD card.

Requirements:

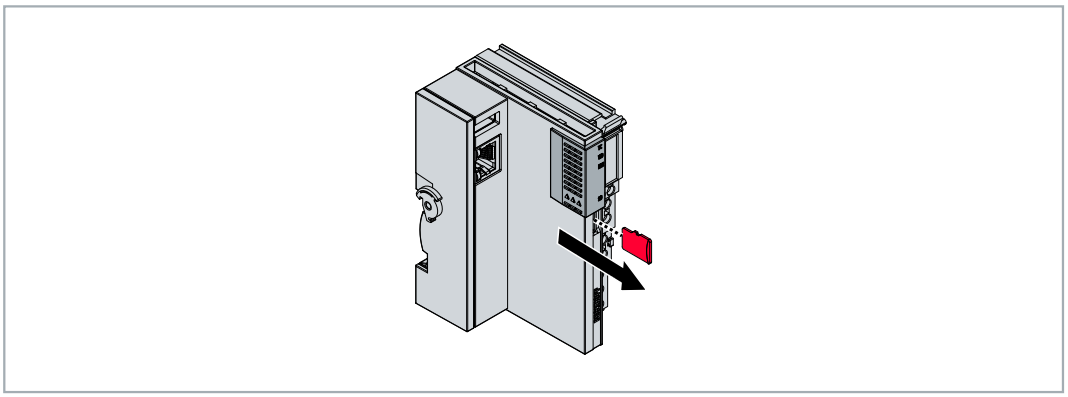
- The Embedded PC must be switched off. The MicroSD card may only be installed or removed in switched-off state.

Changing the MicroSD card

1. Push the black cover upwards.



2. Gently push the MicroSD card.
3. The card is unlatched with a quiet click and raised about 2 – 3 mm out of the housing.



4. Push the new MicroSD card into the card slot with the contacts at the front. The contacts face to the right.
5. A soft click can be heard when the MicroSD card engages.

⇨ The card is seated correctly when it is about 1 mm deeper than the front side of the housing.

5.1.4 Installing passive EtherCAT Terminals

Incorrectly installed passive EtherCAT Terminals

The E-bus signal between an Embedded PC and the EtherCAT Terminals can be impaired due to incorrectly installed passive EtherCAT Terminals.

Passive EtherCAT Terminals should not be installed directly on the power supply unit.

EtherCAT Terminals that do not take part in active data exchange are referred to as passive terminals. Passive EtherCAT Terminals have no process image and do not require current from the terminal bus (E-bus).

Passive EtherCAT Terminals (e.g. EL9195) can be detected in TwinCAT. In the tree structure the EtherCAT Terminal is displayed without process image, and the value in column “E-bus (mA)” does not change, compared to the preceding EtherCAT Terminal.

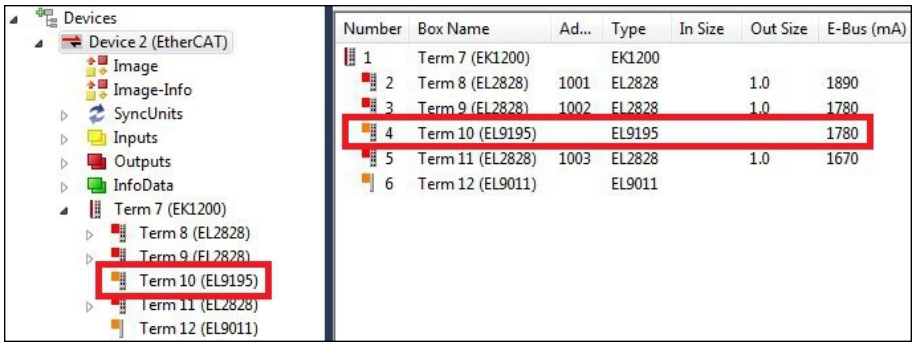


Fig. 6: Identifying a passive EtherCAT Terminal in TwinCAT.

The entry "Current consumption via E-Bus" in the technical data of an EtherCAT Terminal indicates whether a particular EtherCAT Terminal requires power from the terminal bus (E-bus).

The following diagram shows the permissible installation of a passive EtherCAT Terminal. The passive EtherCAT Terminal was not directly attached to the power supply unit.

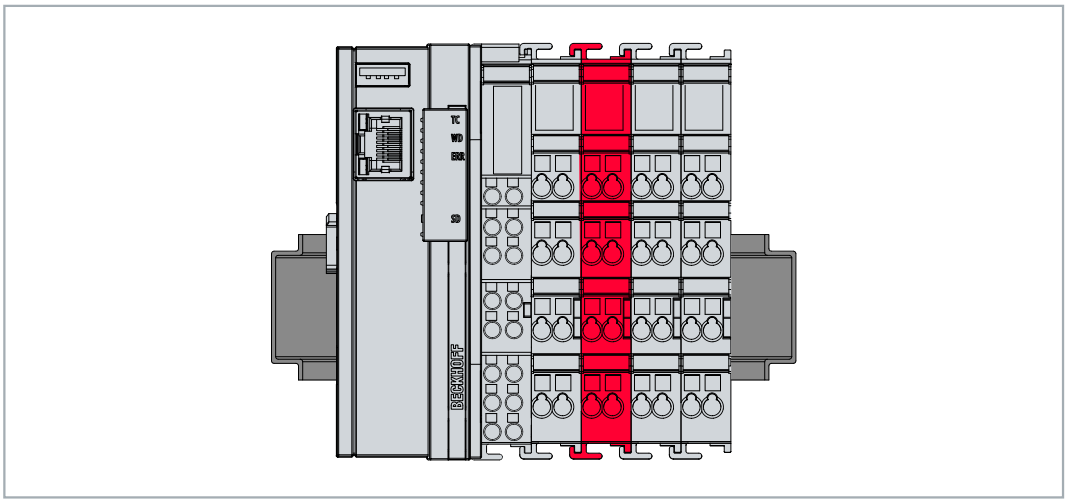


Fig. 7: Passive EtherCAT Terminals, permissible installation.

5.2 Power supply

NOTICE

Damage to the Embedded PCs

The Embedded PCs may be damaged during wiring. The cables for the power supply should only be connected in de-energized state.

The power supply terminal requires an external voltage source which provides 24 V DC (-15 % / +20 %).

The cabling of the Embedded PC in the control cabinet must be done in accordance with the standard EN 60204-1:2006 (PELV = Protective Extra Low Voltage):

- The "PE" and "0 V" conductors of the voltage source for a basic CPU module must be on the same potential (connected in the control cabinet).
- Standard EN 60204-1:2006, section 6.4.1:b stipulates that one side of the circuit, or one point of the energy source for this circuit must be connected to the protective earth conductor system.

Connections

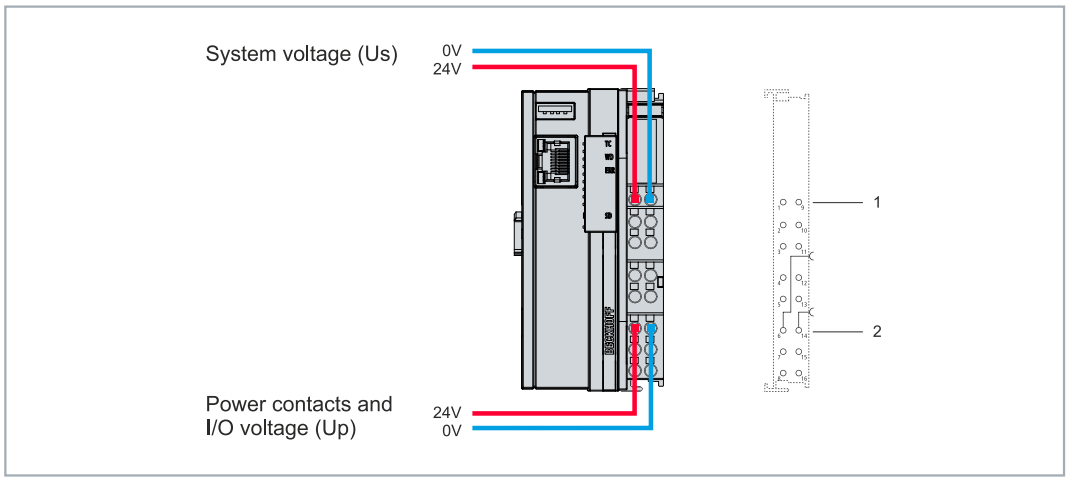


Fig. 8: Connections for system voltage (Us) and power contacts (Up).

Table 6: Key for the connection example.

No.	Description
1	The upper spring-loaded terminals labeled "24 V Us" and "0 V Us" supply the basic CPU module and the terminal bus (data transfer via K- or E-bus) with voltage.
2	The spring-loaded terminals labeled "+24 V Up" and "0 V Up" supply the multi-functional I/Os, the bus terminals, and EtherCAT Terminals with voltage via the power contacts.

Fuse

- When dimensioning the fuse for the system voltage (Us), take the maximum power consumption of the embedded PC into account (see: Technical data [▶ 109])
- Protect the power contacts (Up) with a fuse with a max. rating of 10 A (slow-blow).

Interrupting/switching off the power supply

To switch off the embedded PC, do not disconnect the ground (0 V), because otherwise current may continue to flow via the shielding, depending on the device, and damage the embedded PC or peripheral devices.

Always disconnect the 24 V line. Devices connected to the embedded PC which have their own power supply (e.g. a panel) must have the same potential for "PE" and "0 V" as the embedded PC has (no potential difference).

5.2.1 Connect Embedded PC

The cables of an external voltage source are connected to spring-loaded terminals on the power supply terminal. Observe the required conductor cross-sections and strip lengths.

Table 7: Required wire cross-sections and strip lengths.

Conductor cross-section	e*: 0.08 ... 1.5 mm²	e*: AWG 28 ... 16
	f*: 0.25 ... 1.5 mm²	f*: AWG 22 ... 16
	a*: 0.14 ... 0.75 mm²	a*: AWG 26 ... 19
Strip length	8 ... 9 mm	0.33 inch

*e: single-wire, solid wire; f: stranded wire; a: with wire end sleeve

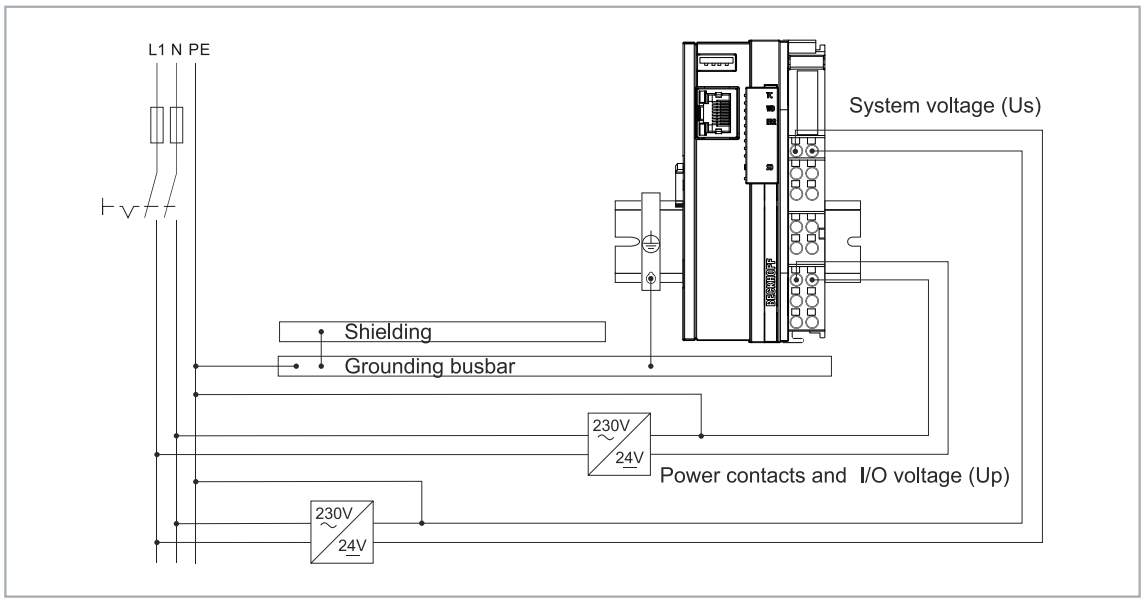
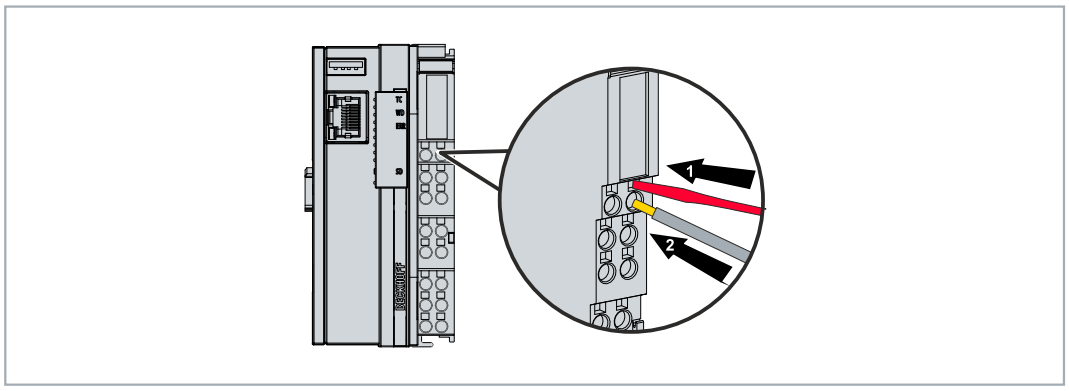


Fig. 9: Connection example with a CX7000.

Connect the Embedded PC as follows:

1. Open a spring-loaded terminal by slightly pushing with a screwdriver or a rod into the square opening above the terminal.



2. The wire can now be inserted into the round terminal opening without any force.
3. The terminal closes automatically when the pressure is released, holding the wire safely and permanently.

⇨ You have successfully connected the voltage source to the power supply terminal when the two upper LEDs of the power supply terminal light up green.

The left LED (Us 24V) indicates the supply of the basic CPU module and terminal bus. The red LED (Up 24V) indicates the Bus Terminal supply via the power contacts.

5.2.2 UL requirements

The CX7000 Embedded PCs are UL-certified. The corresponding UL label can be found on the name plate.

Fig. 10: UL label on CX7000.

The CX7000 Embedded PCs can thus be used in areas where special UL requirements have to be met. These requirements apply to the system voltage (Us) and the power contacts (Up). Applications without special UL requirements are not affected by UL regulations.

UL requirements:

- The Embedded PCs must not be connected to unlimited voltage sources.
- Embedded PCs may only be supplied from a 24 V DC voltage source. The voltage source must be insulated and protected with a fuse of maximum 4 A (corresponding to UL248).
- Or the power supply must originate from a voltage source that corresponds to NEC class 2. An NEC class 2 voltage source must not be connected in series or parallel with another NEC class 2 voltage source.

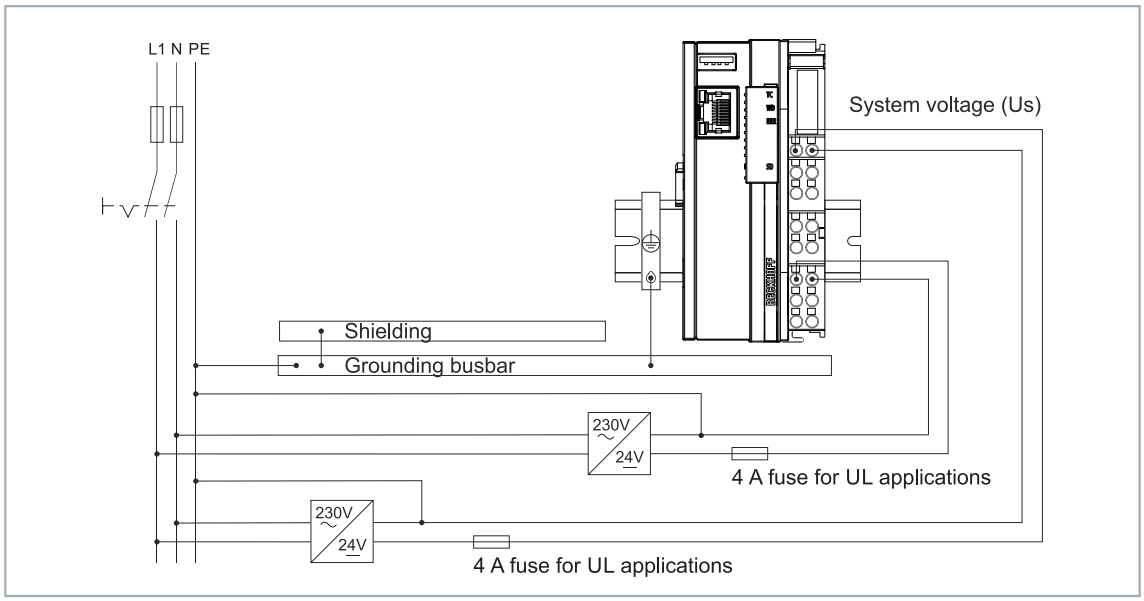


Fig. 11: Connection example for areas with special UL requirements.

6 Multifunction I/Os

A total of four adjustable slots are available for configuring the operation modes. A slot is a certain number of inputs and outputs. For each slot a maximum of one module (DI, DIO, ENC, CNT or PWM) can be assigned, which in turn determines the operation mode for the respective slot. A module is therefore a function that these inputs and outputs can assume. The current module configuration is listed in TwinCAT under the CX7028 interface. Note that the CX7028 interface for controlling the multifunction I/Os has its own CPU and the CX7028 interface is not displayed or does not work under TwinCAT if the power supply (Up) is not connected.

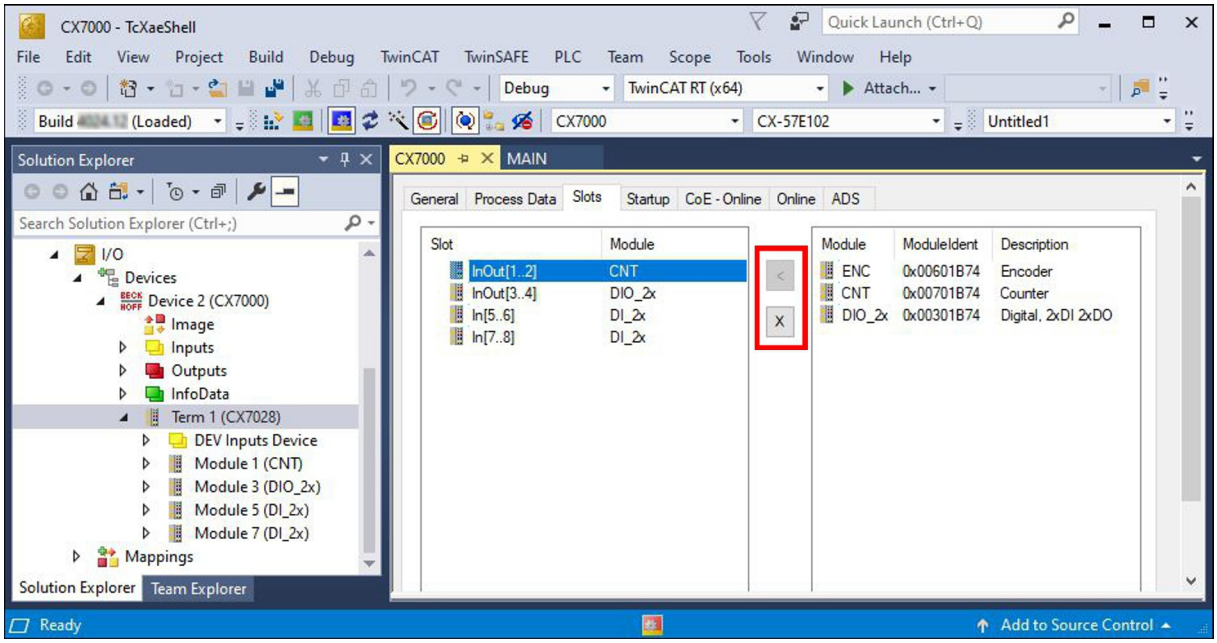


Fig. 12: CX7028 interface, slot and module configuration under TwinCAT.

Modules can be assigned to a specific slot with the button **<** or removed again with **x**. There is a choice of different modules depending on the slot used. The module used by each slot is listed in the following.

Cycle time for multifunction I/Os

Communication to the multifunction I/Os is monitored with a fixed watchdog of 100 ms. This means that the cycle time for the multifunction I/O must be faster than 100 ms.

Slot 1:

When using slot 1, inputs 1, 2 and (*3) as well as outputs 1 and 2 are configured.

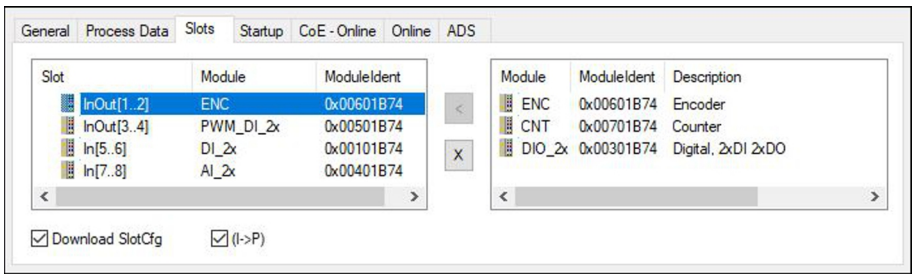


Fig. 13: Supported modules when using slot 1.

- ENC (incremental encoder mode). 2 x digital input for 250 kHz encoder signal, 2 x encoder digital output.
- CNT (counter mode). 1 x counter digital input 100 kHz, 1 x digital input as up/down counter 20 kHz, 2 x counter digital output.
- DIO_2x (digital inputs and outputs). 2 x digital input, 24 V DC, filter 3 ms, type 3, 2 x digital output, 24 V DC, 0.5 A, 1-wire technique.

*) Input 3 is only available in incremental encoder mode. If the level is high, the value of the incremental encoder can be latched or the counter reset.

Slot 2:

When using slot 2, inputs 3 and 4 as well as outputs 3 and 4 are configured.

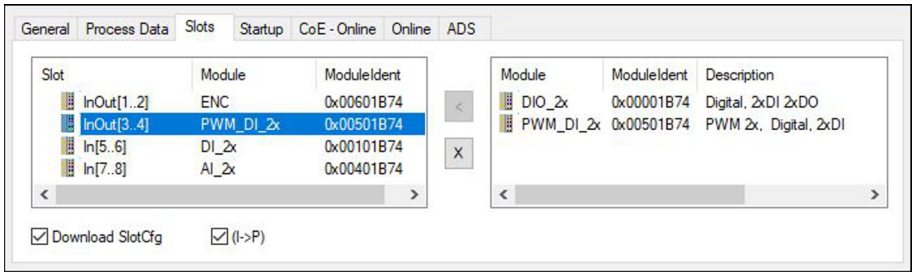


Fig. 14: Supported modules when using slot 2.

- DIO_2x (digital inputs and outputs). 2 x digital input, 24 V DC, filter 3 ms, type 3, 2 x digital output, 24 V DC, 0.5 A, 1-wire technique.
- PWM_DI_2x (PWM signal mode). 2 x digital input, 24 V DC, filter 3 ms, 2 x digital output configured for PWM signal.

Slot 3:

When using slot 3, inputs 5 and 6 are configured.

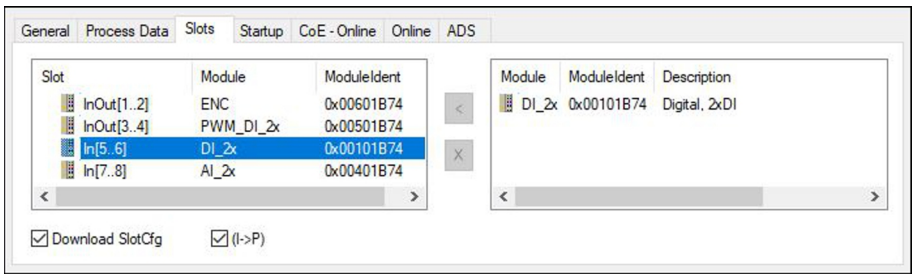


Fig. 15: Supported modules when using slot 3.

Slot 3 contains only one module and therefore cannot be configured differently. The module supports 2 x digital input, 24 V DC, filter 3 ms, type 3.

Slot 4:

When using slot 4, inputs 7 and 8 are configured.

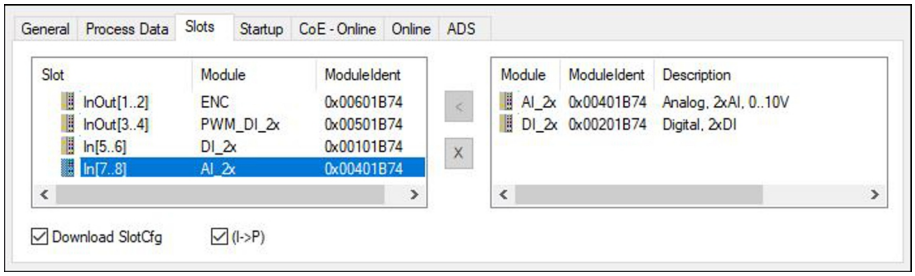


Fig. 16: Supported modules when using slot 4.

- AI_2x (analog signal mode). 2 x digital input configured as analog input 0 to 10 V, 12 bits
- DI_2x (digital input). 2 x digital input, 24 V DC, filter 3 ms, type 3

6.1 Digital inputs

The digital inputs acquire binary control signals from the process level. Typically, these are mechanical contacts such as normally closed contacts or normally open contacts, electronic sensors such as inductive proximity switches, optical sensors or other methods in order to generate a low/high signal in the sense of control technology. Thanks to integrated multi-function I/Os, the CX70xx has a total of 8 digital inputs, 24 V DC, filter 3 ms, type 3.

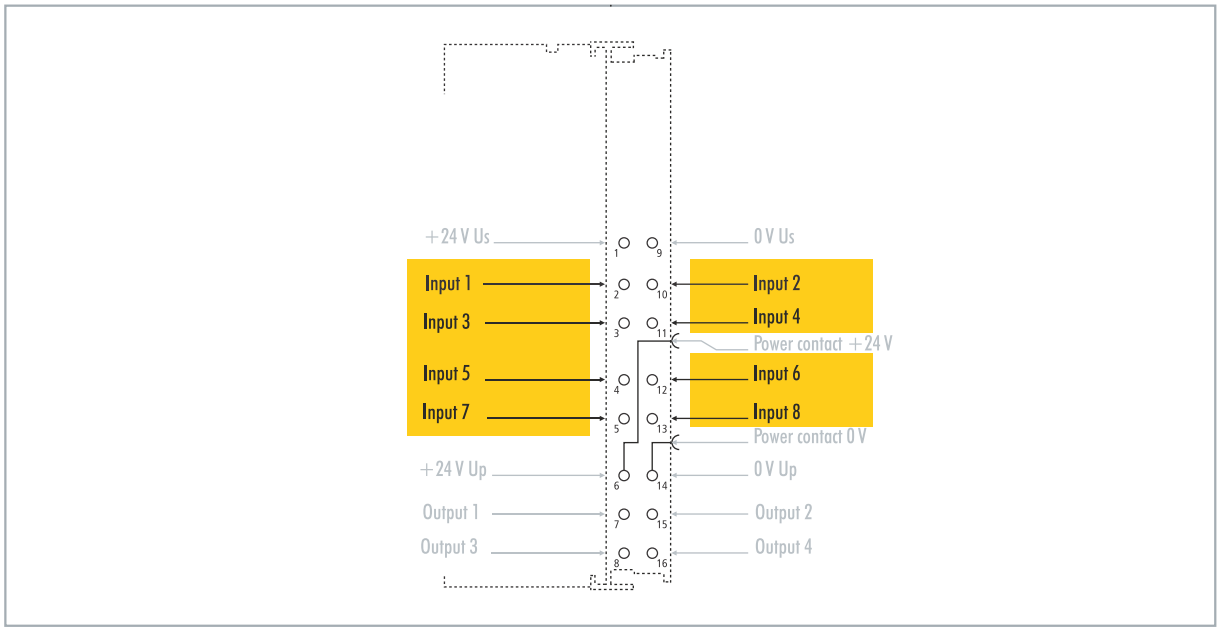


Fig. 17: Configurable digital inputs.

The digital inputs have a 3 ms input filter. The signal status of each individual input is displayed by an LED. For digital inputs 3, 4, 5 and 6, additional filter settings can be made in the appropriate CoE objects and, for example, the resolution and filter time can be set.

Table 8: Technical data, multi-function I/Os as digital inputs.

Technical data	CX7000
Connection technology	1-wire
Number of inputs	8
Nominal voltage	24 V DC (-15 %/+20 %)
Specification	EN 61131-2, type 3
Signal voltage "0"	-3…+5 V
Signal voltage "1"	11…30 V
Input filter	Configurable, default: 3 ms, min.: 10 µs
Connection cross-section	e*: 0.08…1.5 mm², f*: 0.25…1.5 mm², a*: 0.14…0.75 mm²
Connection cross section AWG	e*: AWG 28…16, f*: AWG 22…16, a*: AWG 26…19
Strip length	8 … 9 mm

*e: single-wire, solid wire; f: stranded wire; a: with ferrule

6.2 Digital outputs

NOTICE

Feedback at the 24 V outputs

A voltage of 24 V at the outputs can destroy the device if the power supply (Up) is not connected (feedback). Connect the power supply (Up) so that 24 V can be applied to the outputs.

The digital outputs forward binary 24 V DC control signals, electrically isolated, to actuators at the process level. The high level of the positive switching logic corresponds to the supply voltage.

Outputs 3 and 4 have a PWM output stage. If the two digital outputs are used as normal digital outputs, the internal wiring will cause a leakage current of less than 100 µA, which will cause a voltage of about 5 V. If you want to reach nearly 0 V at the low level of the output, you have to connect a 47 kΩ resistance to ground.

Another possibility is to operate the two outputs in PWM mode and to write the variable `PWM output` of the PWM signal for FALSE with 0x0000 and for TRUE with 0xFFFF. This activates the PWM output stage, which does not generate any leakage current.

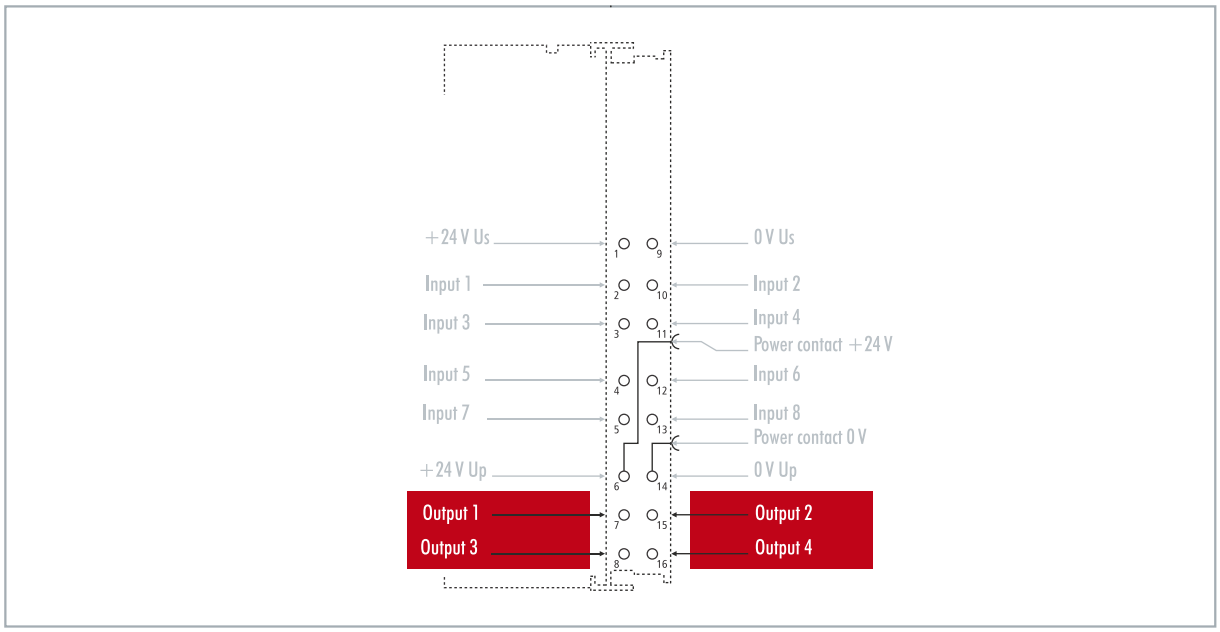


Fig. 18: Configurable digital outputs.

The CX7000 contains a total of four outputs, which indicate their signal state by means of light emitting diodes. The outputs can be used to switch standard actuators such as contactors and valves.

Table 9: Technical data, multi-function I/Os as digital outputs.

Technical data	CX7000
Connection technology	1-wire
Number of outputs	4
Nominal voltage	24 V DC (-15 %/+20 %)
Load type	ohmic, inductive, lamp load
Max. output current	24 V/0.5 A (short-circuit proof)
Changeover times	T_{ON}: 20 µs typ., T_{OFF}: 10 µs typ.
Short circuit current	< 2 A typ.
Max. breaking energy (ind.)	< 150 mJ/channel
Connection cross-section	e*: 0.08…1.5 mm², f*: 0.25…1.5 mm², a*: 0.14…0.75 mm²

Technical data	CX7000
Connection cross section AWG	e*: AWG 28…16, f*: AWG 22…16, a*: AWG 26…19
Strip length	8 … 9 mm

*e: single-wire, solid wire; f: stranded wire; a: with ferrule

6.3 Counter mode

The CX7000 Embedded PC can be configured as an up/down counter that enables the counting of a pulse. The embedded PC is suitable for fast counting tasks with a cut-off frequency of up to 100 kHz, whereby the CX7000 can be operated in 1-counter mode.

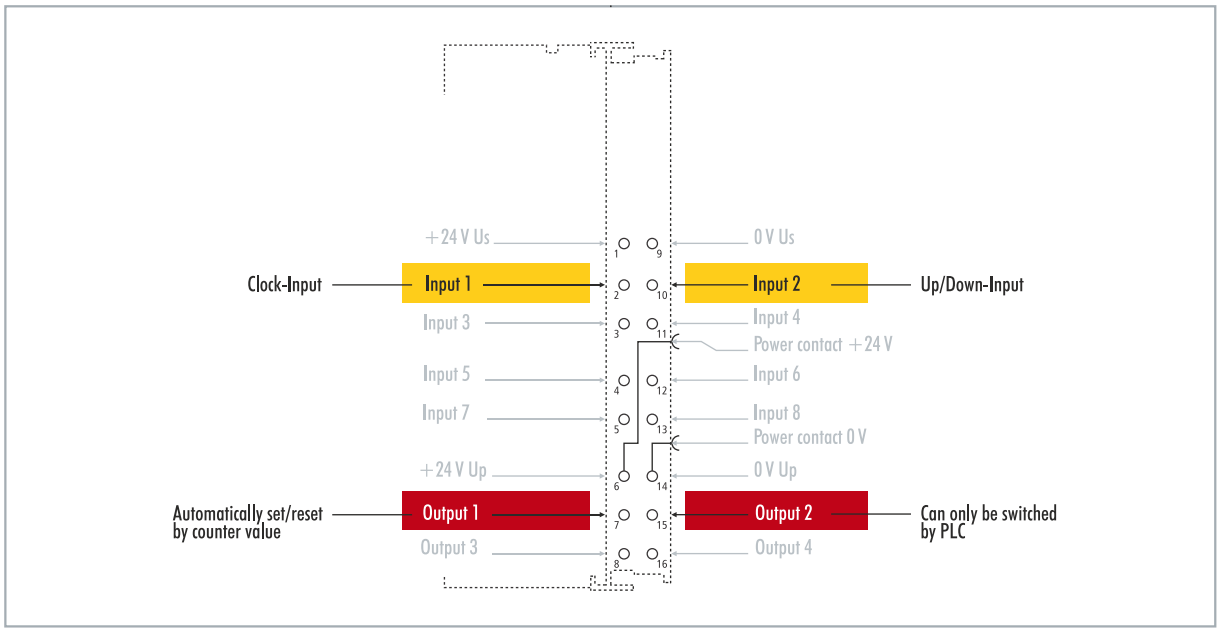


Fig. 19: Configurable inputs and outputs in counter mode.

The CX7000 supports three operation modes in counter mode:

- Up/down counter
- Up counter
- Down counter

In addition, output 1 can be switched depending on the counter value. Output 2 can be switched from the PLC. This allows fast control signals for field devices to be used and switched.

The operation modes are set in TwinCAT via CoE objects.

Up/down counter

In the up/down counter operation mode, the pulse to be counted is detected by digital input 1. The counting direction is specified by digital input 2.

If there is a high level at input 1 and at the same time at input 2, the counter counts upwards. If there is a high level at input 1 and a low level at input 2, the counter counts downwards.

Up counter

In this operation mode, the signal is detected at digital input 1.

Down counter

In this operation mode, the signal is detected at digital input 1.

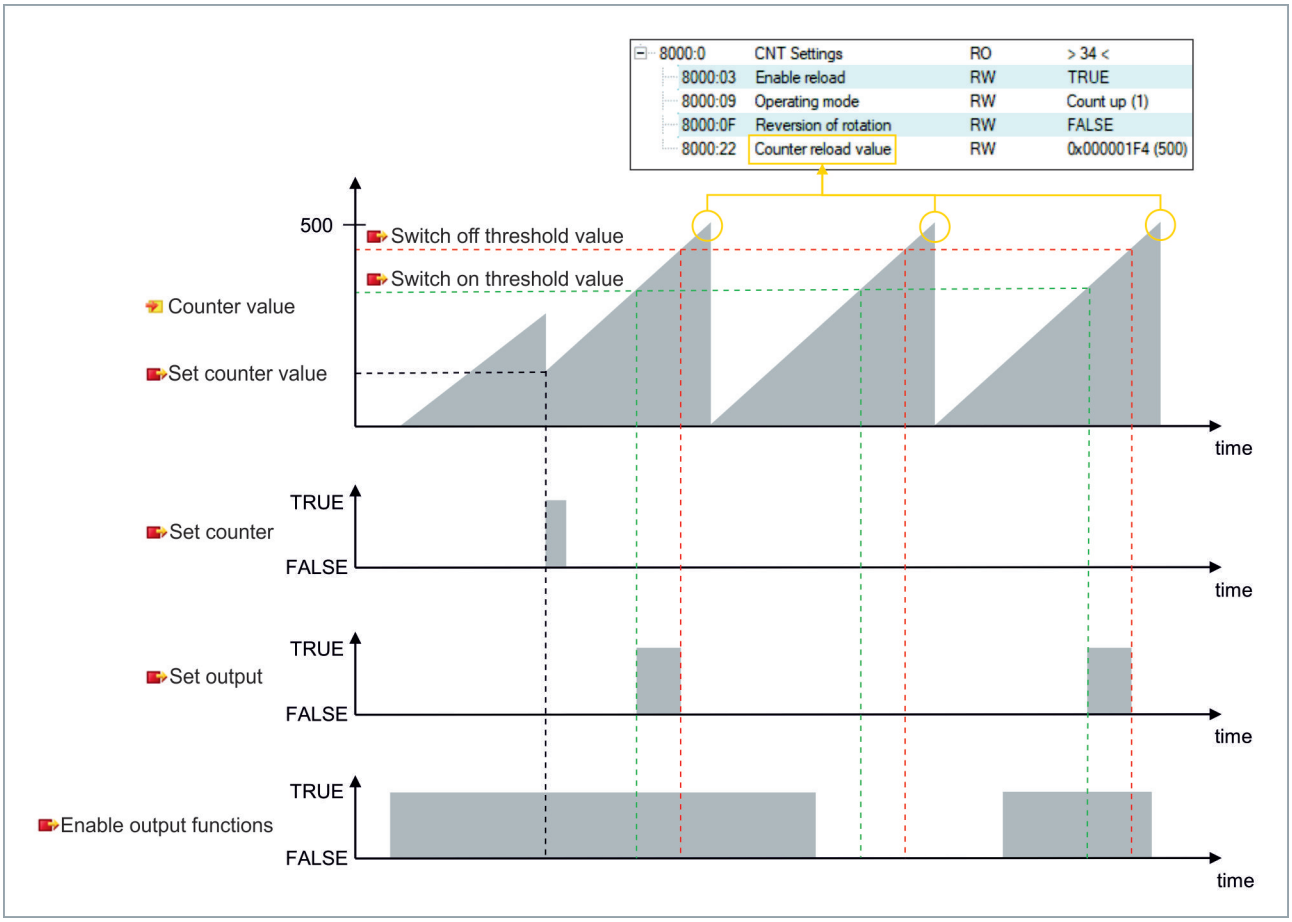


Table 10: Technical data, multi-function I/Os in counter mode.

Technical data	CX7000
Number of counters	1 x up/down counter, 1 x up or down counter
Nominal voltage	24 V DC (-15 %/+20 %)
Specification	EN 61131-2, type 3
Signal voltage "0"	-3…+5 V
Signal voltage "1"	11…30 V
Cut-off frequency	Up/down counter: 20 kHz[1)], counting in one direction only: 100 kHz
Counter depth	32-bit
Max. output current	24 V/0.5 A (short-circuit proof)
Special features	Set counter, switch outputs, reset counter
Connection cross-section	e*: 0.08…1.5 mm², f*: 0.25…1.5 mm², a*: 0.14…0.75 mm²
Connection cross section AWG	e*: AWG 28…16, f*: AWG 22…16, a*: AWG 26…19
Strip length	8 … 9 mm

[1)] The up/down counter can also count up to 100 kHz, only with a direction reversal the counting frequency must be <= 20 kHz, otherwise pulses will be lost.

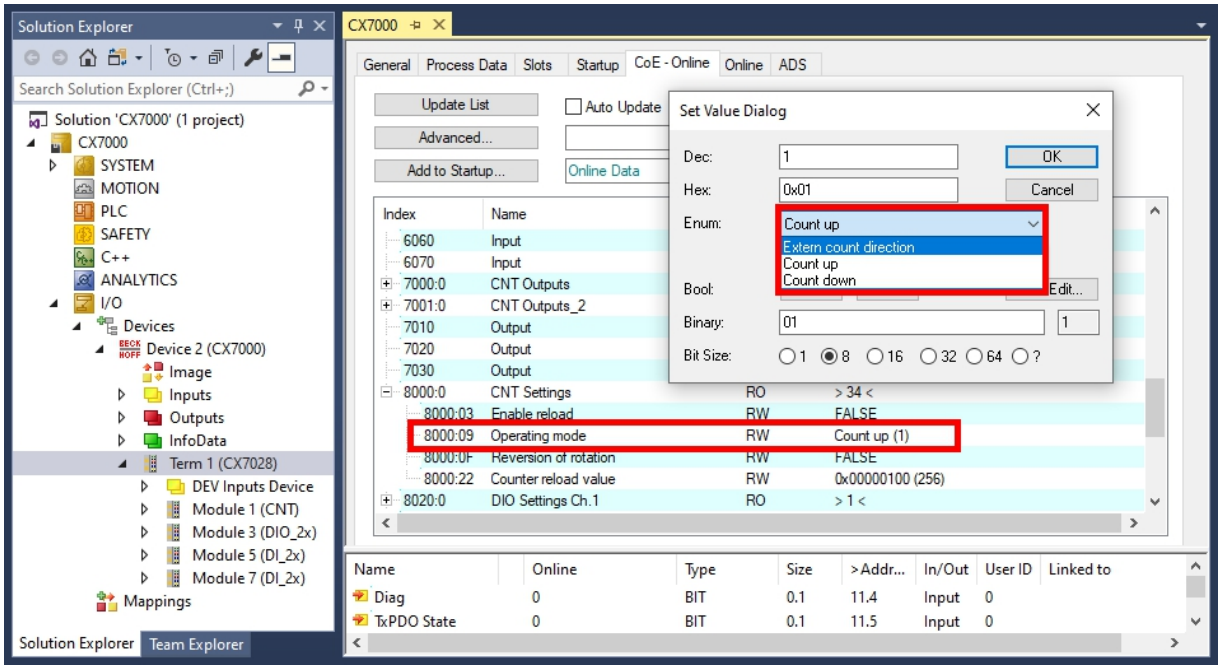
*e: single-wire, solid wire; f: stranded wire; a: with ferrule

6.3.1 Select operation mode

The CX7000 supports three operation modes in counter mode: The operation mode is set in TwinCAT via CoE objects. You can choose between the three operating modes up/down counter, up counter and down counter.

Proceed as follows:

1. Click the **CX7028 device** on the left in the structure tree.
2. Click the **CoE-Online** tab.



3. Double-click the CoE object **8000:09 Operating mode**.
4. Under the **Enum** option, select the required operation mode.

⇨ The operation mode is applied. Note that you can only use one operation mode at a time with the CX7000 and mode mixing is not possible.

6.3.2 Switching outputs

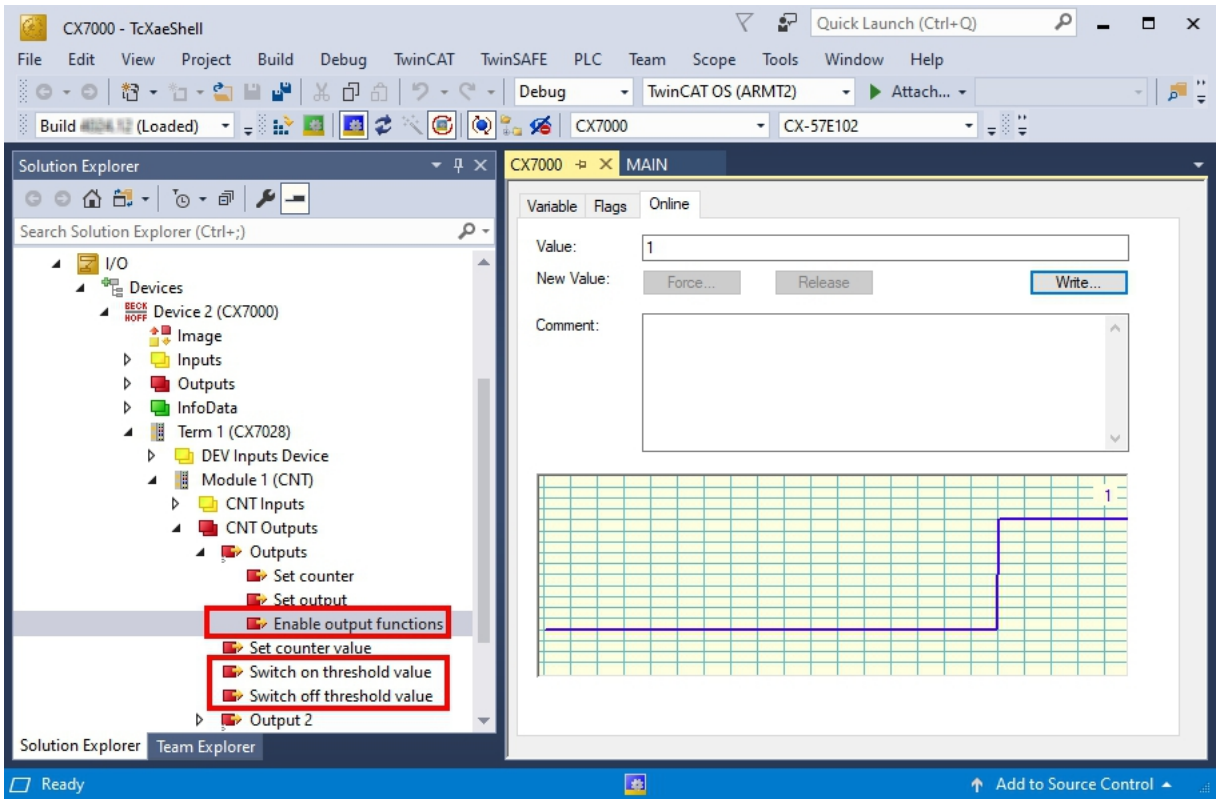
With the CX7000, it is possible to switch output 1 automatically as soon as a certain counter value is reached. This enables fast processing without the PLC. A second output, output 2, can be switched via the PLC irrespective of the counter value.

Output 1 is switched or switched off respectively by the variables **Switch on threshold value** and **Switch off threshold value**:

- If the value set under **Switch on threshold value** is reached, the output is switched.
- If the value set under **Switch off threshold value** is reached, the output is switched off.

When counting downwards, the corresponding switching instruction is executed in reverse. If the value falls below the value set in **Switch on threshold value,** output 1 is switched off.

Proceed as follows:



1. Use the variable **Switch on threshold value** to specify a counter value at which the output should be switched.
2. Use the variable **Switch off threshold value** to specify a counter value at which the output should be switched off.
3. Then set the variable **Enable output functions** to **True** so that the settings are applied.

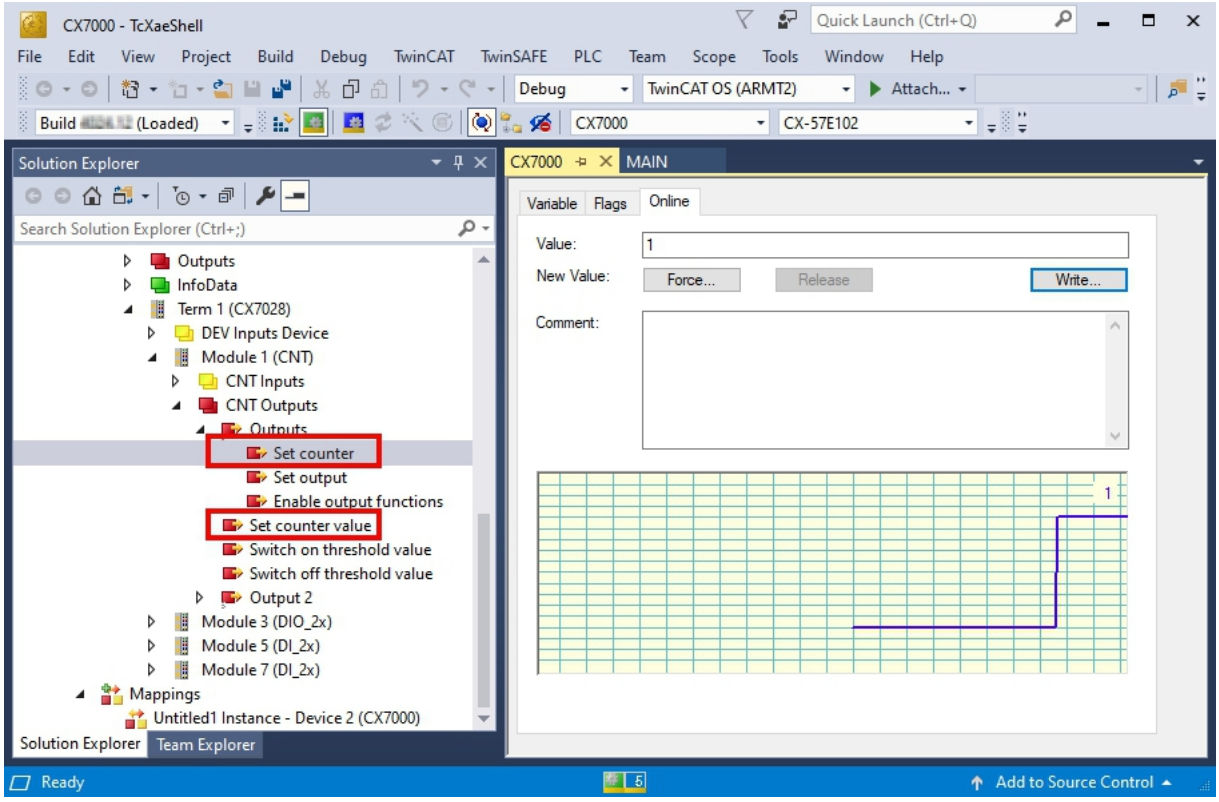
⇨ Only when the variable **Enable output functions** is set to **True** the function is enabled and the output is switched.

If the parameterized counter value from **Switch on/off threshold** is reached or exceeded, but the variable **Enable output functions** is not set, the switching order is not executed. The output is switched as soon as **Enable output functions** is set. Likewise, a subsequently activated counter value **Switch on/off threshold** affects the output immediately when the switching condition is fulfilled.

6.3.3 Set counter value

This step shows you how to set the counter value to a specific value. The variable **Set counter value is** used to specify a value and the variable **Set counter** is used to set the counter value. Both variables can be controlled from the PLC.

Proceed as follows:



1. Use the variable **Set counter value** to specify a value to set as a counter value.
2. Then set the variable **Set counter** to **True** to apply the settings.

⇨ Only when the variable **Set counter** is set to **True**, the value set under **Set counter value** is applied for the counter value.

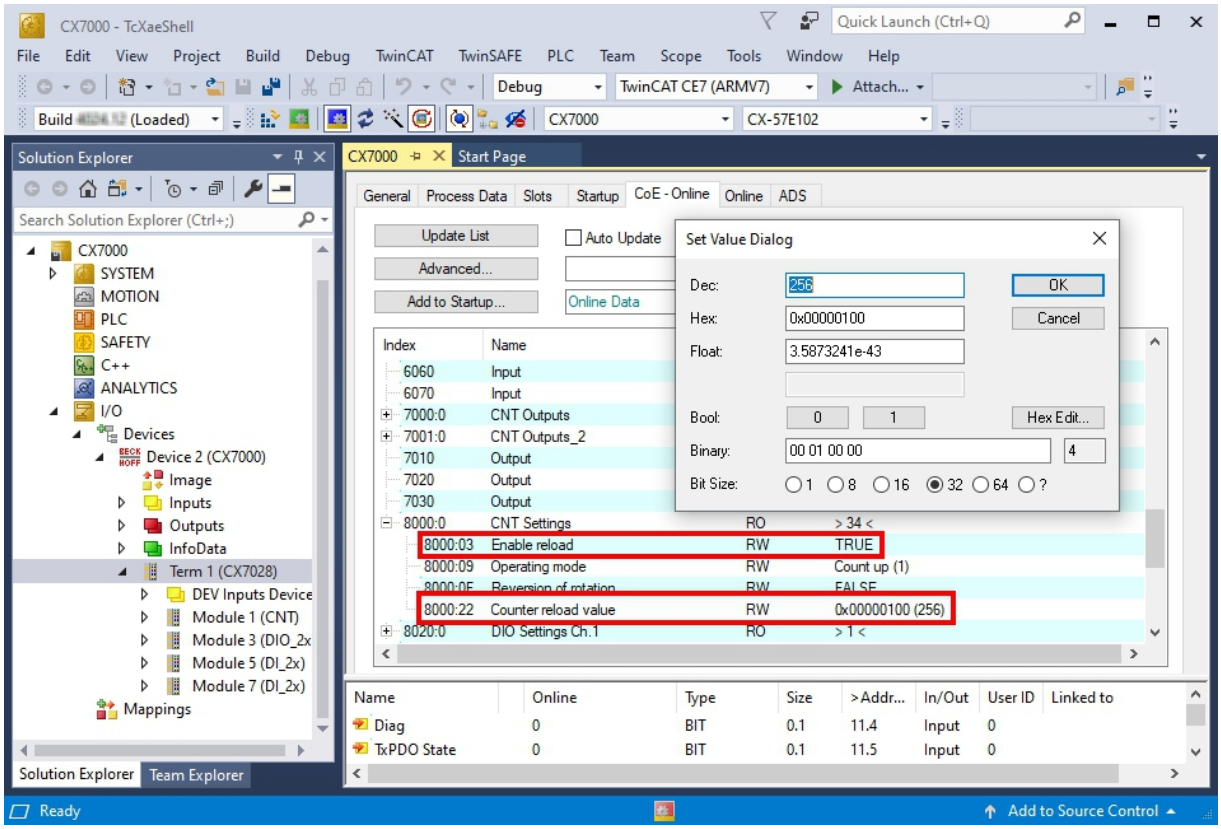
6.3.4 Setting the limit value for counters

This step shows you how to set a limit value in TwinCAT from which the counter value is automatically reset to zero. When counting upwards, the counter value is reset to zero when the limit value is reached. When counting downwards, the counter value is reset to the set limit value when zero is reached.

The counter value is a UDINT variable. The counter counts only in the positive range from 0 to 0xFFFF_FFFF (4294967295). If the value falls below zero, the counter is set to the maximum positive value. If it exceeds 4294967295, the counter is set to zero. The two variables **Counter underflow** or **Counter overflow** respectively indicate the overflow and are reset either on reaching 0x4000 in the positive direction or on reaching 0xFFFFC000 in a negative direction or if the corresponding other overflow has been reached.

Proceed as follows:

1. Click the **CX7028 device** on the left in the structure tree.
2. Click the **CoE-Online** tab.



3. Double-click the CoE object **8000:22 Counter reload value** and set the limit value.
4. Then double-click the CoE object **8000:03 Enable reload** and set the value to **True**.

⇨ Only when the CoE object **8000:03 Enable reload** is set to **True** are the function and the defined limit value active.

6.4 Incremental encoder mode

In incremental encoder mode, the CX7000 can be configured as an interface for direct connection of 24 V incremental encoders. A quadruple evaluation is used and both high level and low level are detected at input 1 and input 2.

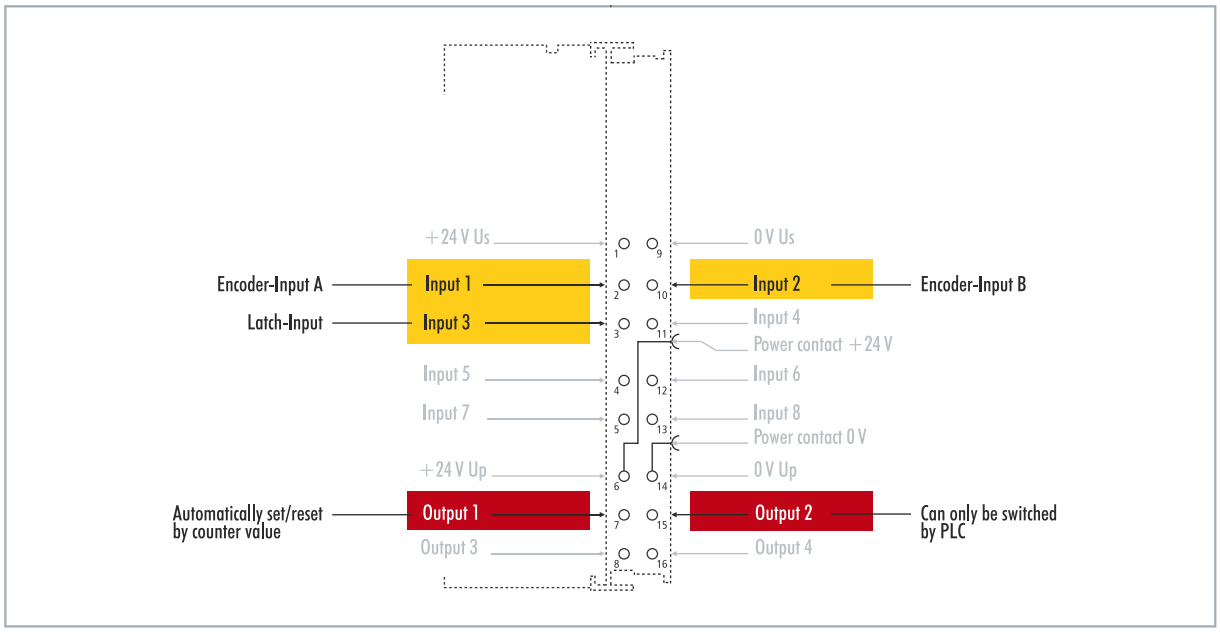


Fig. 20: Configurable inputs and outputs in incremental encoder mode.

The range of functions in encoder mode corresponds to the range of functions in counter mode. In addition, the counter value at input 3 can be latched, i.e. the value is entered in the process data on a high level at input 3. Alternatively, the counter can be reset on a high level at input 3.

In addition, output 1 can be switched depending on the counter value. Output 2 can be switched from the PLC. This allows fast control signals for field devices to be used and switched.

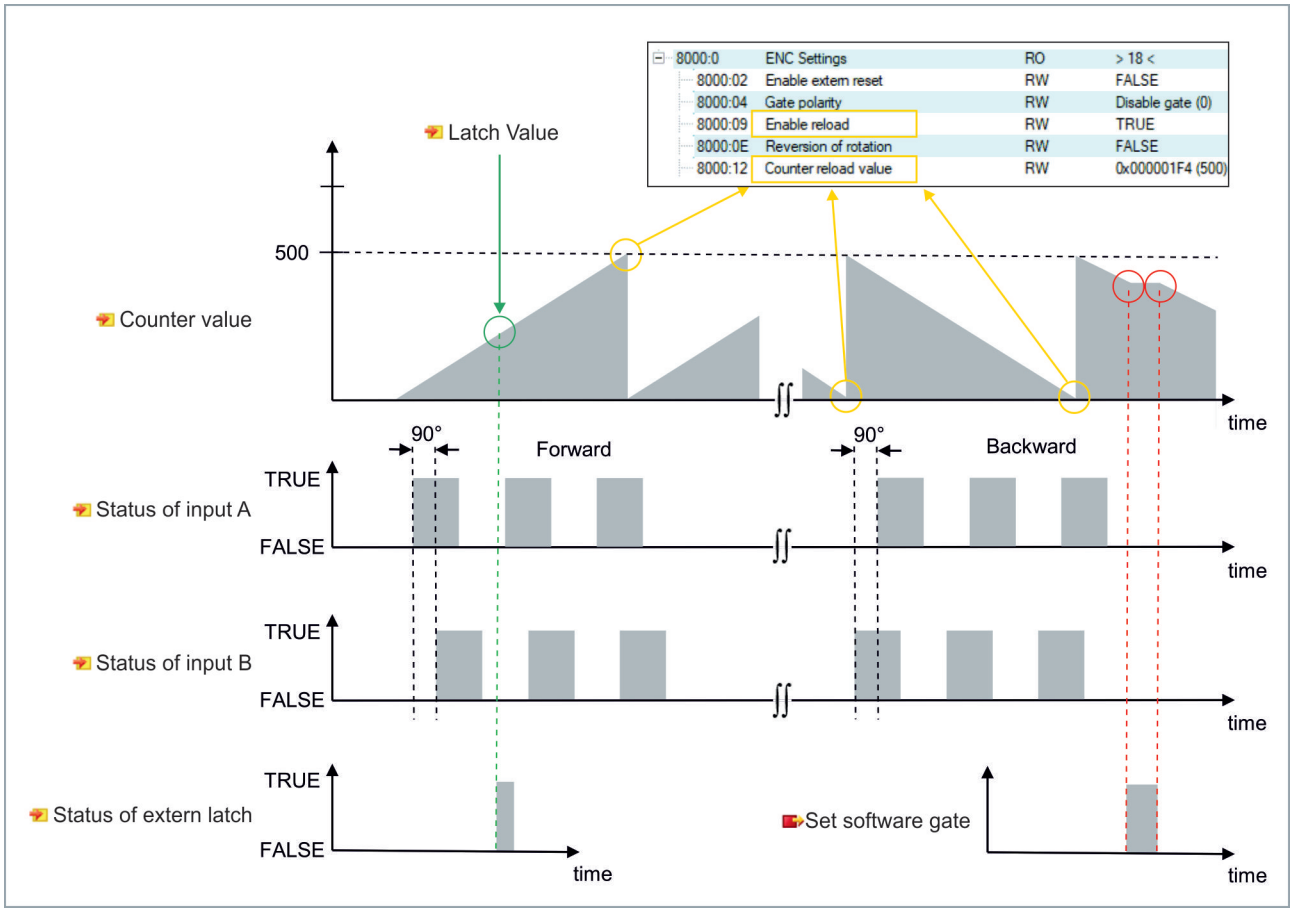


Table 11: Technical data, multi-function I/Os in encoder mode.

Technical data	CX7000
Technology	Incremental encoder interface
Nominal voltage	24 V DC (-15 %/+20 %)
Specification	EN 61131-2, type 3
Encoder connection	1 x A, B: 24 V, single-ended
Additional inputs	Latch input, 24 V DC
Cut-off frequency	250,000 increments/s (with 4-fold evaluation), corresponds to 62.5 kHz
Counter depth	32-bit
Quadrature decoder	4-fold evaluation
Max. output current	24 V/0.5 A (short-circuit proof)
Special features	Latch function, software gate, set counter, switch outputs, reset counters
Connection cross-section	e*: 0.08…1.5 mm², f*: 0.25…1.5 mm², a*: 0.14…0.75 mm²
Connection cross section AWG	e*: AWG 28…16, f*: AWG 22…16, a*: AWG 26…19
Strip length	8 … 9 mm

*e: single-wire, solid wire; f: stranded wire; a: with ferrule

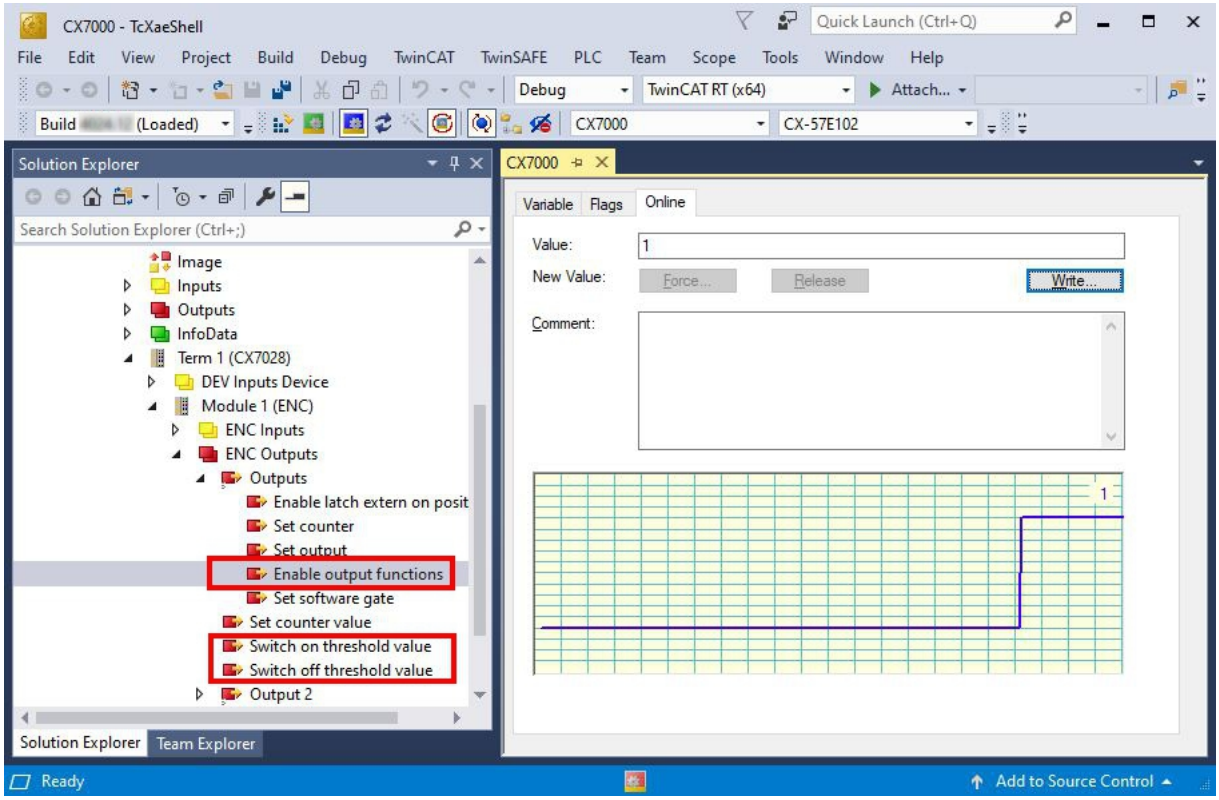
6.4.1 Switching outputs

With the CX7000, it is possible to switch output 1 automatically as soon as a certain counter value is reached. This enables fast processing without the PLC. A second output, output 2, can be switched via the PLC irrespective of the counter value.

Output 1 is switched or switched off respectively by the variables **Switch on threshold value** and **Switch off threshold value**:

- If the value set under **Switch on threshold value** is reached, the output is switched.
- If the value set under **Switch off threshold value** is reached, the output is switched off.

Proceed as follows:



1. Use the variable **Switch on threshold value** to specify a counter value at which the output should be switched.
2. Use the variable **Switch off threshold value** to specify a counter value at which the output should be switched off.
3. Then set the variable **Enable output functions** so that the settings are applied.

⇨ Only when the variable **Enable output functions** is set to **True** is the function enabled and the settings applied.

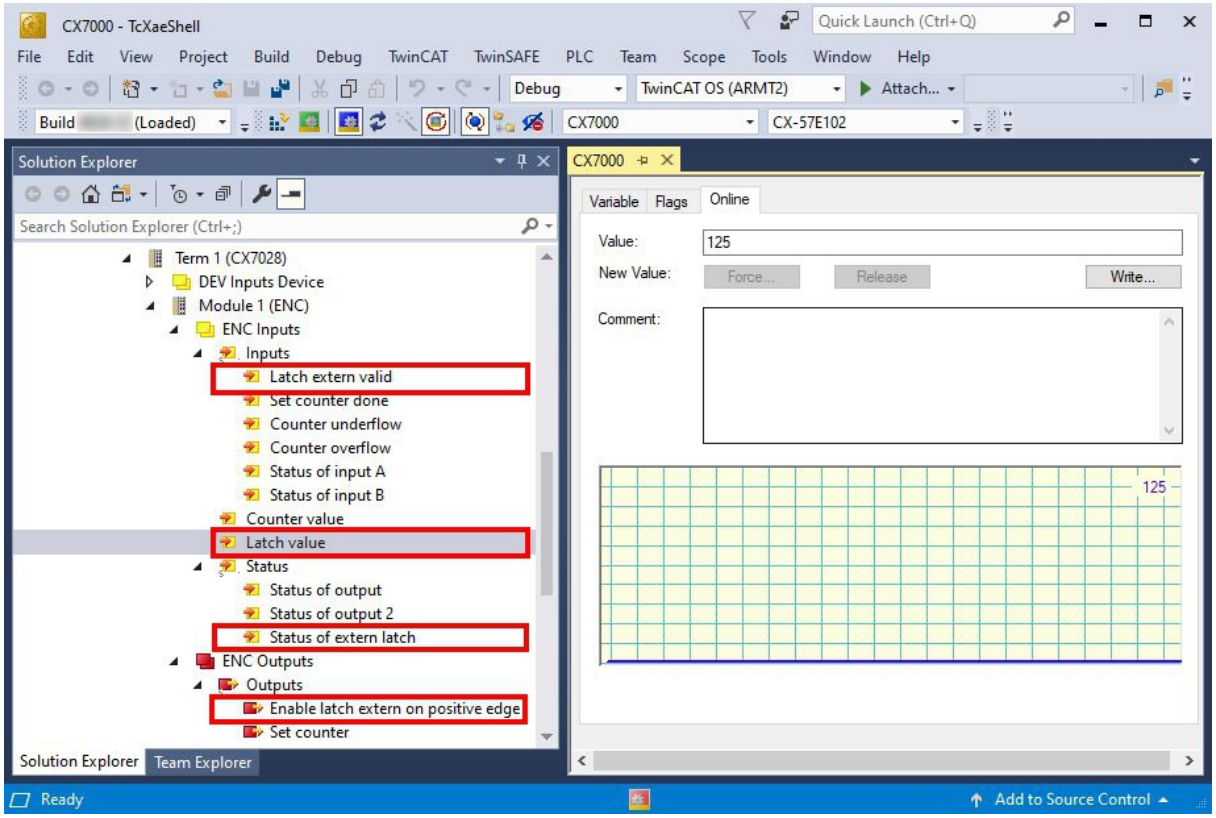
If the parameterized counter value from **Switch on/off threshold** is reached or exceeded, but the variable **Enable output functions** is not set, the switching order is not executed. The output is switched as soon as **Enable output functions** is set. Likewise, a subsequently activated counter value **Switch on/off threshold** affects the output immediately when the switching condition is fulfilled.

6.4.2 Latching the counter value

In incremental encoder mode, the counter value can be latched and thus the current value can be entered in the process data. Input 3 is used as a latch input.

To enable the function, the variable **Enable latch extern on positive edge** must be set to **True**. On a high level at input 3, the current counter value is entered into the variable **Latch Value**. You can monitor the validity of the variable. As soon as the latch value is entered, the variable **Latch extern valid** is also set to **True**.

Proceed as follows:



1. Set the variable **Enable latch extern on positive edge** to **True** to enable the latch function.
2. Monitor the status of the latch input with the variable **Status of extern latch**.
3. On a high level at input 3, the current counter value is entered into the variable **Latch Value**.
4. Monitor the validity of the latch value via the variable **Latch extern valid**. Once the latch value has been written, the variable is also set to **True**.

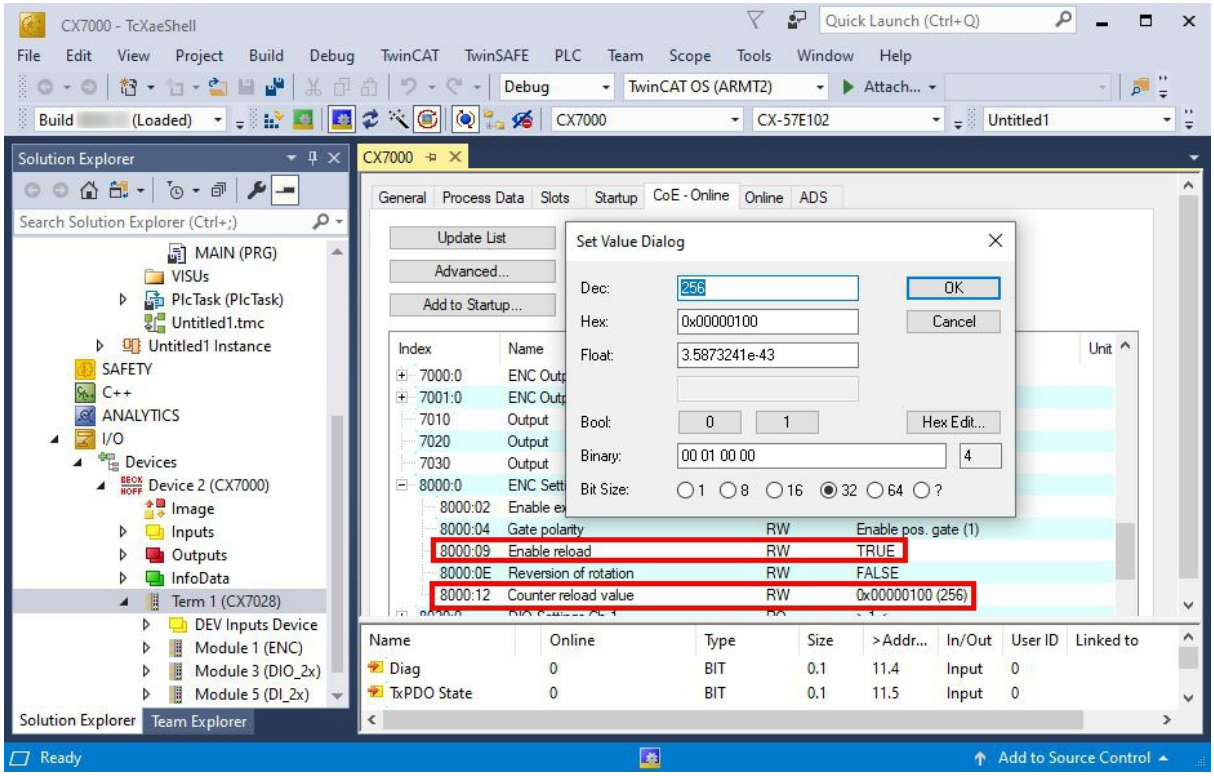
⇨ To execute a latch again, the variable **Enable latch extern on positive edge** must receive a high level again.

6.4.3 Setting the limit value for counters

This step shows how you can set a limit value in TwinCAT from which the counter value is automatically reset to zero. When counting upwards, the counter value is reset to zero when the limit value is reached. When counting downwards, the counter value is reset to the set limit value when zero is reached.

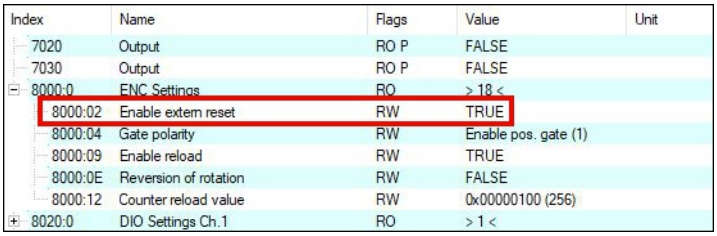
Proceed as follows:

1. Click the **CX7028 device** on the left in the structure tree.
2. Click the **CoE-Online** tab.



3. Double-click the CoE object **8000:12 Counter reload value** and set the limit value.
4. Then double-click the CoE object **8000:09 Enable reload** and set the value to **True**.

⇨ The function is only active when **Enable reload** is set. Alternatively, the latch input can be used and the counter value can thus be reset externally. To do this, the latch function must be disabled and the CoE object **Enable extern reset** set to **True**. With this setting, the current counter value is set to zero on a high level at input 3.



6.5 Analog signal mode

The single-ended inputs 7 and 8 acquire signals in the range of 0 to 10 V.

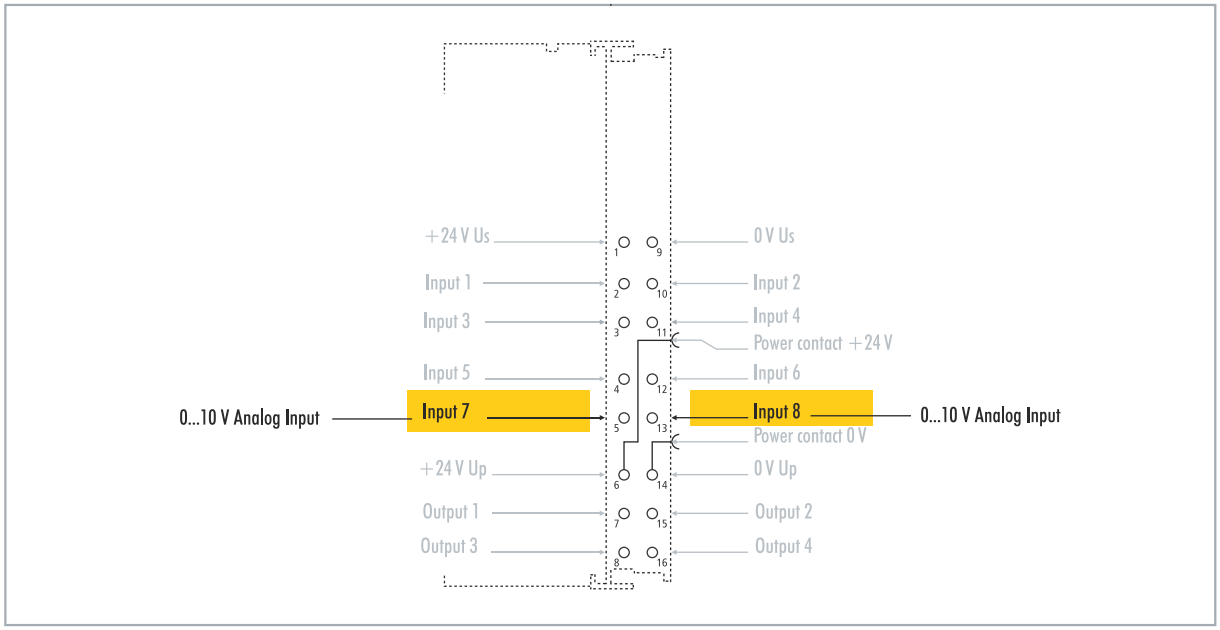


Fig. 21: Configurable analog inputs.

The voltage is digitized with a resolution of 12 bits. LEDs are used to indicate the signal state.

Table 12: Technical data, multi-function I/Os in analog mode.

Technical data	CX7000
Technology	single ended
Number of inputs	2
Signal voltage	0…10 V
Internal resistance	500 kΩ
Input filter cut-off frequency	2 kHz
Resolution	12-bit (16-bit representation)
Measuring error	< ±0.3 % (relative to full scale value)
Connection cross-section	e*: 0.08…1.5 mm², f*: 0.25…1.5 mm², a*: 0.14…0.75 mm²
Connection cross section AWG	e*: AWG 28…16, f*: AWG 22…16, a*: AWG 26…19
Strip length	8 … 9 mm

*e: single-wire, solid wire; f: stranded wire; a: with ferrule

6.6 PWM signal mode

NOTICE

Feedback at the 24 V outputs

A voltage of 24 V at outputs 3 and 4 can destroy the device (feedback). No voltage may be applied to the outputs in PWM mode.

The PWM signal mode enables a pulse width modulated binary signal to be output at outputs 3 and 4.

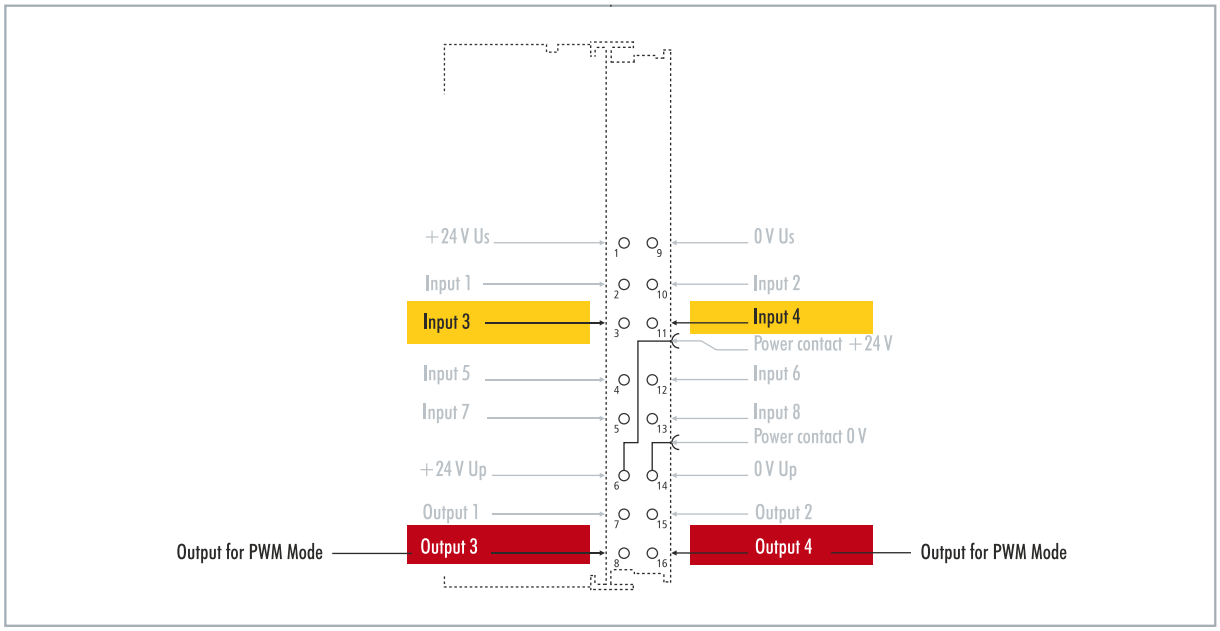


Fig. 22: Configurable inputs and outputs in PWM signal mode

This signal is separated into duty cycle (0... 100 %) and PWM clock frequency (15 Hz... 100 kHz). The LEDs are clocked with the outputs, and show the duty cycle by their brightness. The signal values are transferred in 16-bit values.

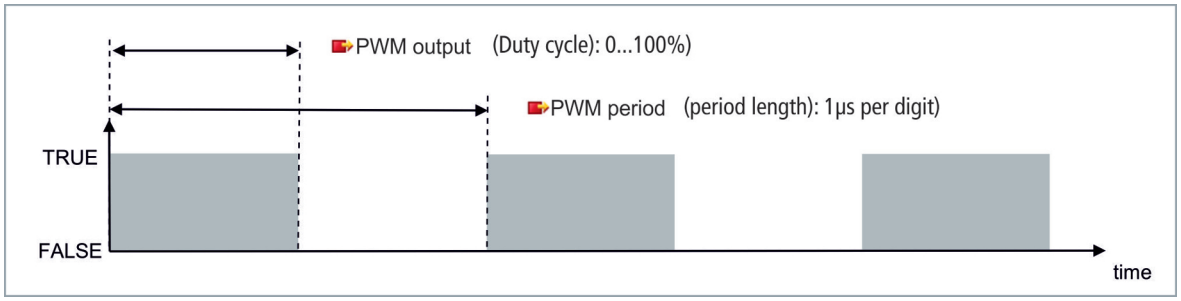


Table 13: Technical data, multi-function I/Os in PWM mode.

Technical data	Digital inputs
Connection technology	PWM output
Number of outputs	2
Nominal voltage	24 V DC (-15 %/+20 %)
Load type	ohmic, inductive, lamp load
Max. output current	24 V/0.5 A (short-circuit proof)
PWM clock frequency	15 Hz…100 kHz
Duty cycle	0…100 % (T_{ON} > 20 ns, T_{OFF} > 200 ns)
Short circuit current	< 2 A typ.
Special features	separate frequency can be set for each channel

Technical data	Digital inputs
Connection cross-section	e*: 0.08…1.5 mm², f*: 0.25…1.5 mm², a*: 0.14…0.75 mm²
Connection cross section AWG	e*: AWG 28…16, f*: AWG 22…16, a*: AWG 26…19
Strip length	8 … 9 mm

*e: single-wire, solid wire; f: stranded wire; a: with ferrule

6.6.1 Setting the PWM clock frequency and duty cycle

The signals at outputs 3 and 4 are output with pulse width modulation, the signals being separated into duty cycle and PWM clock frequency. Separate values for duty cycle and PWM clock frequency can be defined for both outputs.

Table 14: PWM output (duty cycle), representation of the PWM signal in the delivery state.

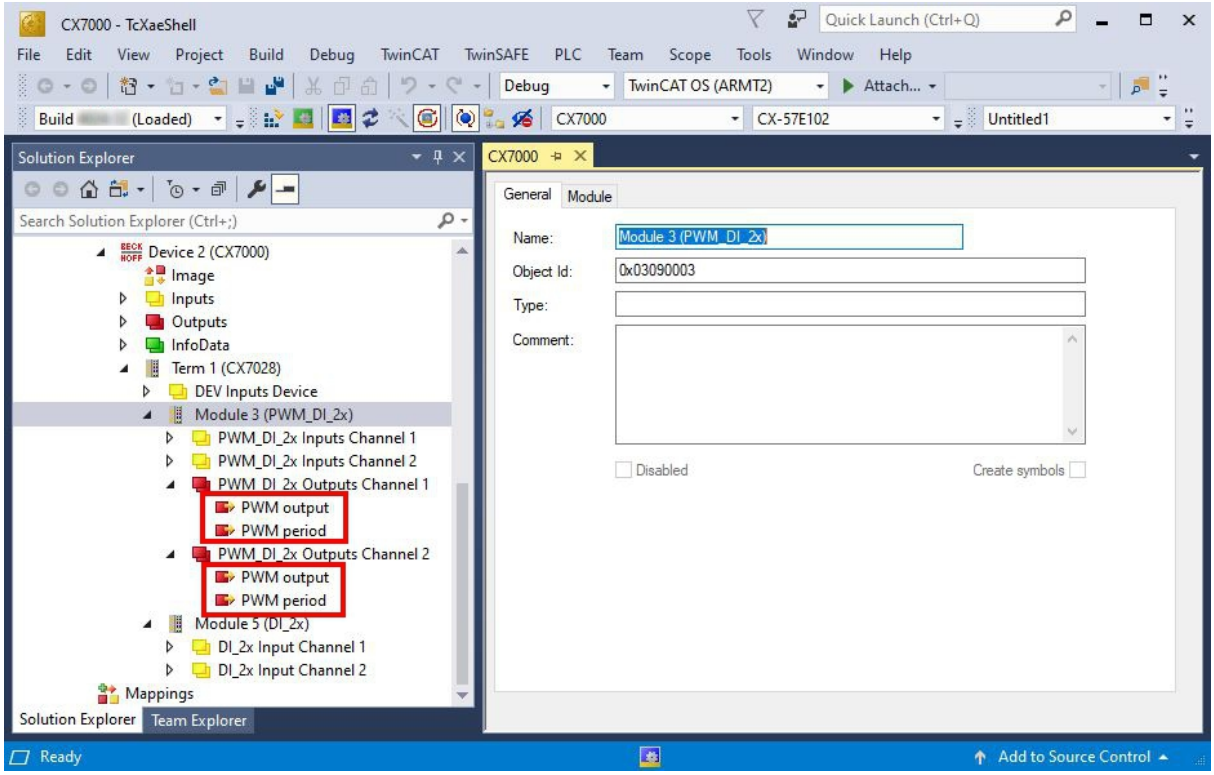
Value	Decimal	Hexadecimal
0 %	0	0x0000
25 %	16383	0x3FFF
50 %	32767	0x7FFF
100 %	65.535	0xFFFF

Table 15: PWM period (PWM clock frequency), representation of the PWM signal in the delivery state.

Value	Decimal	Hexadecimal	Frequency
0.010 ms	0..10	0x0000-0x000A	100 kHz
0.011 ms	11	0x000B	90.909 kHz
0.100 ms	100	0x0064	10 kHz
1.000 ms	1000	0x03E8	1 kHz
16.38 ms	16383	0x3FFF	61.04 Hz
65.53 ms	65535	0xFFFF	15.26 Hz

The variable **PWM output** correspond to the duty cycle and **PWM period** to the PWM clock frequency at which the signal is output.

Proceed as follows:



1. On the left in the structure tree, select an output for which you wish to set the duty cycle and PWM clock frequency.
2. Link the variables **PWM output** and **PWM period** with the appropriate variables from your PLC project.
3. In the variables, set the values for duty cycle and PWM clock frequency according to the above tables.

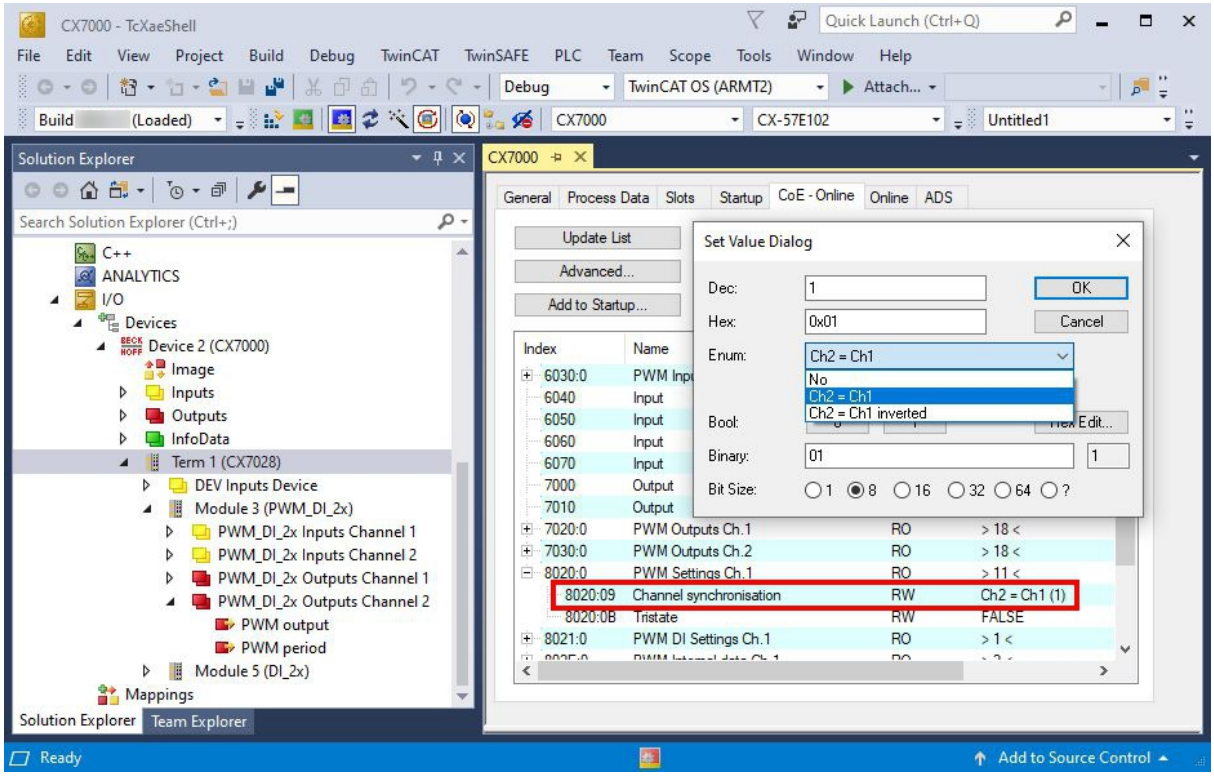
6.6.2 Setting the channel synchronization

The channel synchronization option makes the output of output 2 dependent on output 1. The following values are available in the CoE objects:

- No: no dependency
- Ch2 = Ch1: Duty cycle and PWM clock frequency of output 1 are also applied to output 2. The phase position is 0, i.e. the rising and falling edges of output 1 and output 2 are synchronized.
- Ch2 = Ch1 inverted: Duty cycle and PWM clock frequency of output 1 are also applied to output 2. However, the PWM clock frequency is inverted. The phase position is 0, i.e. a rising edge at output 1 triggers a falling edge at output 2 at the same time.

Proceed as follows:

1. Click the **CX7028 device** on the left in the structure tree.
2. Click the **CoE-Online** tab.



3. Double-click the CoEobject **8020:09 Channel synchronization**.
4. Under the option **Enum**, select the type of synchronization required.

7 Configuration

7.1 Starting the Beckhoff Device Manager

Using the Beckhoff Device Manager, an Industrial PC can be configured by remote access with the aid of a web browser. The access takes place via the HTTP protocol and Port 80 (TCP).

Requirements:

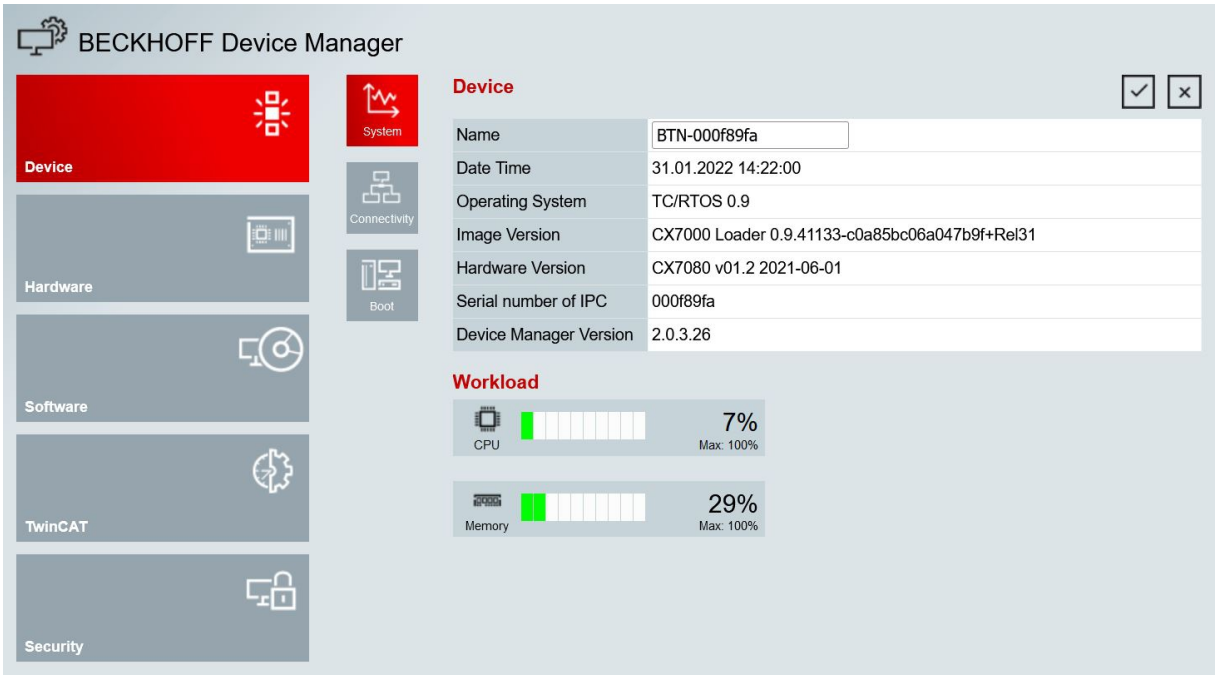
- Host PC and Embedded PC must be located in the same network. The network firewall must allow access via port 80 (HTTP).
- IP address or host name of the Embedded PC.

Table 16: Access data for the Beckhoff Device Manager on delivery.

User name	Password
Administrator	1

Start the Beckhoff Device Manager as follows:

1. Open a web browser on the host PC.
2. Enter the IP address or the host name of the Industrial PC in the web browser to start the Beckhoff Device Manager.
   - Example with IP address: http://169.254.136.237/config
   - Example with host name: http://BTN-000f89fa/config
3. Enter the user name and password. The start page appears:



⇨ Navigate forward in the menu and configure the Industrial PC. Note that modifications only become active once they have been confirmed. It may be necessary to restart the Industrial PC.

7.2 Persistent data

NOTICE

Application example

In the following example, changes to the loads, the power supply or even just aging components can lead to the application no longer fulfilling the desired function. Beckhoff takes no responsibility for the implementation of the example in an application.

Normally, persistent data are only stored during the TwinCAT stop or by a function block. This chapter shows you how to store persistent data on a CX7000 without a UPS.

In the case of an Embedded PC with UPS, the function block is usually linked to the UPS. The function block becomes active as soon as a power failure is detected, writes the persistent data and then shuts down the Embedded PC. With a 1-second UPS, the Embedded PC is not shut down because there is too little time left for this.

In the case of a small controller such as the CX7000 which is delivered without a 1-second UPS, you can still use this function. All that is needed is to use a power supply unit that has enough residual energy to supply power to the CX7000 with this residual energy for a certain period of time. A small test can show you if this is possible with your power supply unit:

Testing a power supply unit

When the CX7000 is running, turn off the AC voltage of your power supply unit and measure how long the CX7000 continues to run. If it is more than three seconds, you may be able to use the power supply unit as a replacement for a 1-second UPS. Note that power supply units also age and lose capacity. You should therefore include a safety factor, such as a factor of three, so that you have enough reserve to be able to operate the power supply unit for a longer period of time as a replacement for a 1-second UPS.

Now determine how long the power supply unit maintains the supply of power. You need an EL1722 for this, which you connect to the AC side of the power supply unit. Then write a small program:

```
VAR
    bPower230V AT %I* : BOOL; (*link to the EL1722*)
END_VAR

VAR RETAIN
    Counter : INT;
END_VAR

Program:
IF NOT bPower230V THEN (*bPower230V is linked to the EL1722*)
    Counter:=counter+1; (*the counter is a retain value*)
END_IF
```

Create a boot project and turn off the AC voltage of the power supply unit. As soon as the EL1722 no longer displays a value, the counter is incremented and the data are copied to the internal NOVRAM. Turn the AC voltage back on and log in. You must now multiply the counter value by the task time. Repeat this a few times to be sure that the power supply unit always behaves in the same way. Next, you have to insert the function block `FB_WritePersistentData`. This is contained in the Tc2_Utilities library (in the "TwinCAT PLC" folder).

Then determine how long it takes to store the persistent data. Repeat this process a few times too, so that you obtain a constant value and can determine a maximum value in case of fluctuations. You can determine the time required via the Busy flag. The function block is being processed as long as the Busy flag is set. Multiply the value determined by two to incorporate a further safety factor.

Example:

Your measurement shows that the power supply unit maintains the supply of power for three seconds and that the persistent data is written in about 400 ms. With the recommended safety factors, the power supply is maintained for one second and the persistent data is written in about 800 ms.

The power supply is therefore maintained for a longer period of time than is needed to store the persistent data. Therefore you can use the example power supply unit as a replacement for the 1-second UPS.

7.3 NOVRAM

The NOVRAM can be used to reliably save important variable values, such as production data or counter values, in the event of a power failure. The memory size of the NOVRAM is limited and only suitable for smaller data quantities up to 4 kB.

In this chapter, we show you how the NOVRAM is used in TwinCAT 3.

Operating principle

The NOVRAM (Non-Volatile Random Access Memory) is a special memory component that is used to reliably save important data. The NOVRAM consists of two sections, a volatile memory and a non-volatile memory.

TwinCAT only writes to the volatile section of the NOVRAM. In the event of a power failure, the data are automatically copied from the volatile memory into the non-volatile memory. The energy required for this process is supplied by a capacitor. As soon as the power supply is restored, the data are automatically copied back into the volatile memory so that TwinCAT can continue to use them.

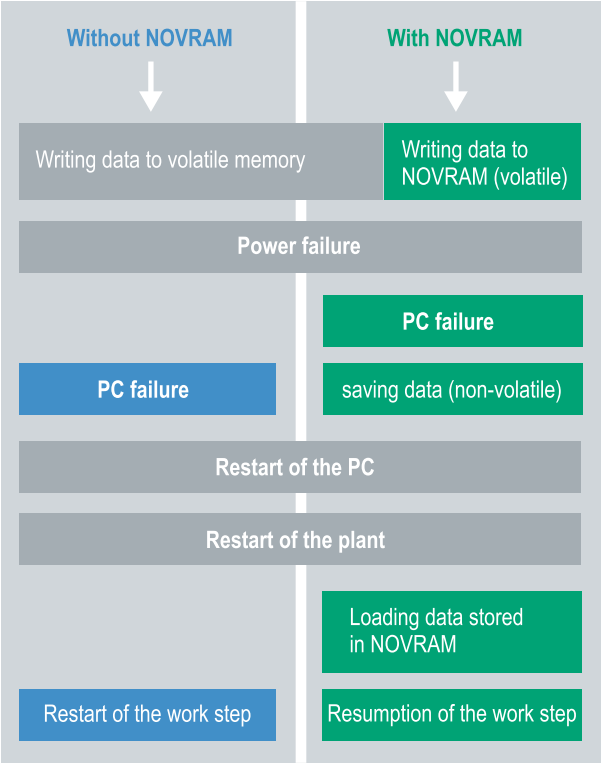


Fig. 23: Controller behavior with and without NOVRAM.

Memory size

The NOVRAM has a capacity of 4 kB. The data are saved cyclically and alternately based on the dual buffer principle, in order to avoid the risk of data inconsistency.

Requirements

Development Environment	Target platforms	Hardware	PLC libraries to include
TwinCAT 3.1 Build: 4020	PC or CX (x86, x64, Arm®)	CX70xx, CX9020, CX20x0, CX20x2, CX20x3	Tc2_IoFunctions

7.3.1 Creating a Retain Handler

Under TwinCAT 3 (from Build 4020) a delta algorithm is used to save data in the NOVRAM. The algorithm does not save all the variables in the NOVRAM. Instead, it searches for changes (delta function) compared to the previous cycle and only saves variables that have changed.

To use the delta algorithm, a Retain Handler must be created in TwinCAT 3, and the relevant variables must be declared in the PLC with the keyword VAR_RETAIN.

A new feature of this method is that no function blocks have to be used. The Retain Handler saves data in the NOVRAM in the event of a power failure and makes them available again once the power has been restored.

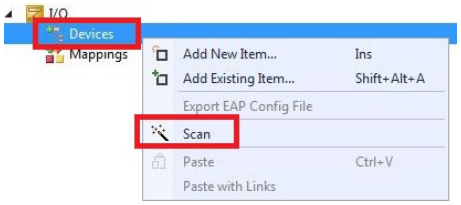
This chapter describes how to create a Retain Handler in TwinCAT 3. The Retain Handler saves data in the NOVRAM and makes them available again. In other words, important variable values such as production data or counter values are retained during a restart or power failure.

Requirements for this step:

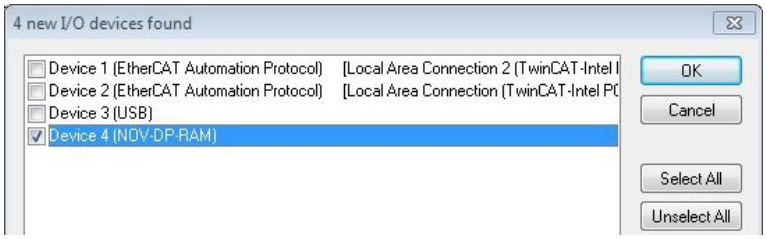
- TwinCAT 3.1 Build: 4020.
- A target device selected in TwinCAT.

Create the Retain Handler as follows:

1. Right-click on **Devices** in the tree view on the left-hand side.
2. In the context menu click on **Scan**.

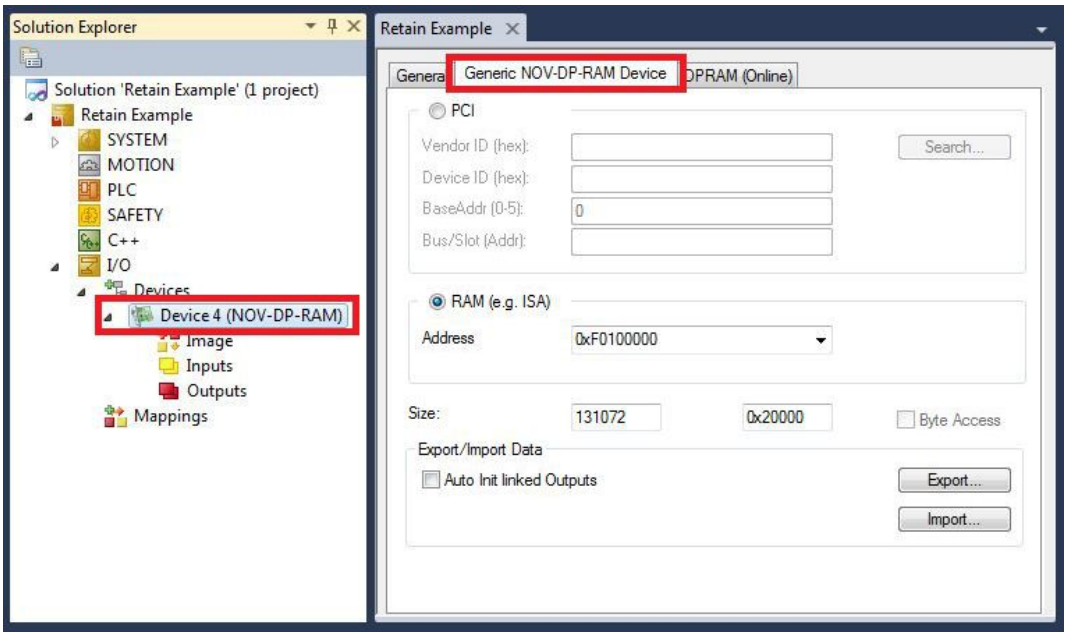


3. Select **Device (NOV-DP-RAM)** and confirm with **OK**.

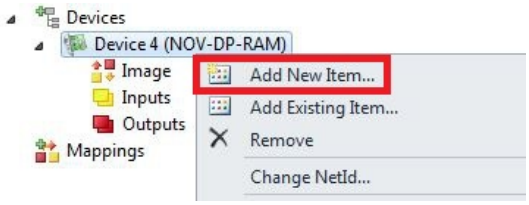


4. Click on **Yes** to search for boxes.

5. Click on **Device (NOV-DP-RAM)** in the tree view on the left-hand side and then on the tab **Generic NOV-DP-RAM Device**.

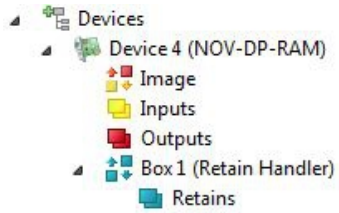


6. Click the option **RAM**.
7. Right-click on **Device (NOV-DP-RAM)** in the tree view and then on **Add New Item**.



8. Select the **Retain Handler** and click on **OK**.

⇨ You have successfully created a Retain Handler in TwinCAT.



In the next step you can create retain variables in the PLC and link them with the Retain Handler.

7.3.2 Creating and linking variables

Once you have created a Retain Handler in TwinCAT, you can declare variables in the PLC and link them to the Retain Handler. The variables have to be identified in the PLC with the keyword `VAR_RETAIN`.

If self-defined data types (DUTs) are used as retain data, the data types must be available in the TwinCAT type system. To do this, the structures must be created directly as `STRUCT RETAIN`, with all occurrences of the structure then handled by the Retain Handler.

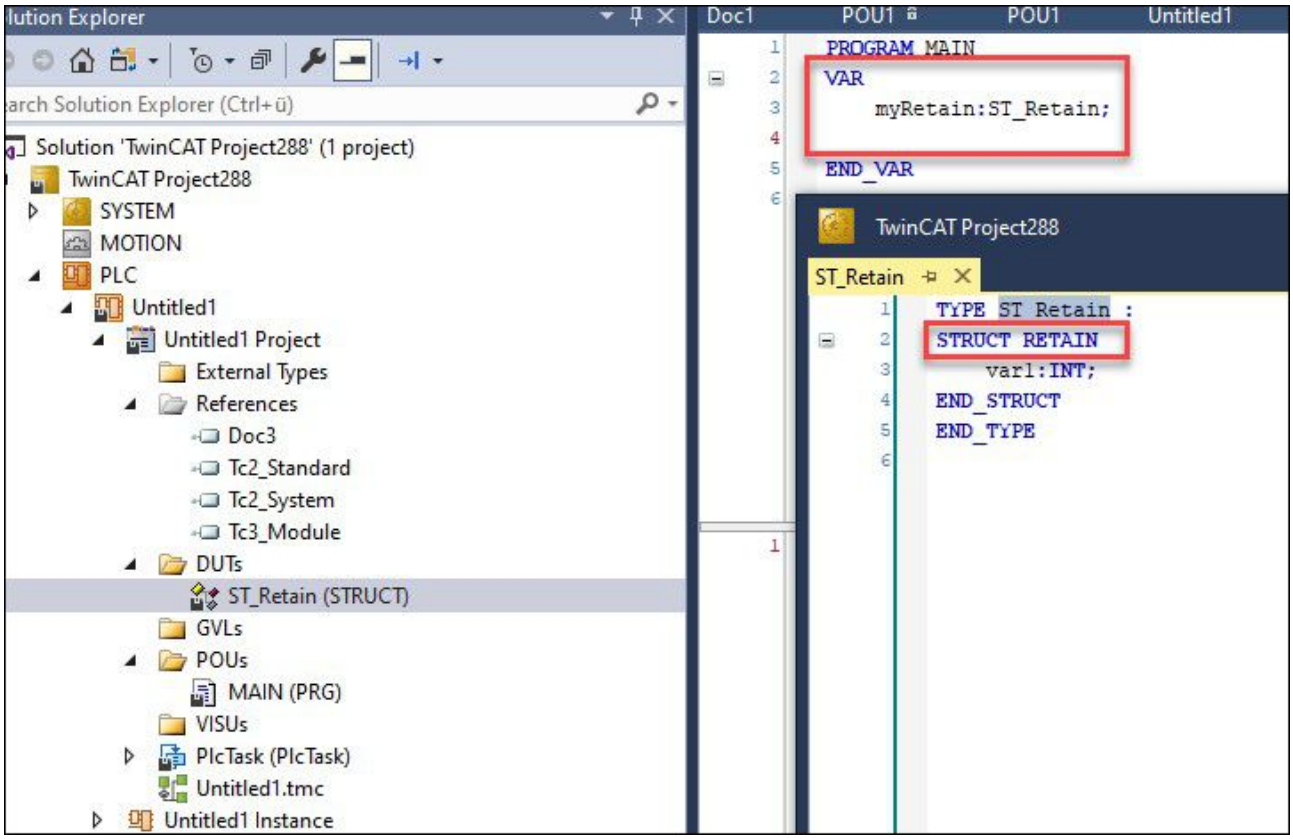


Fig. 24: Labeling with keyword `STRUCT RETAIN` for self-defined Data Unit Types (DUTs)

Retain data cannot be used for POUs (function blocks) as a whole. However, individual elements of a POU can be used.

Prerequisite for this step:

- A PLC project created in TwinCAT.

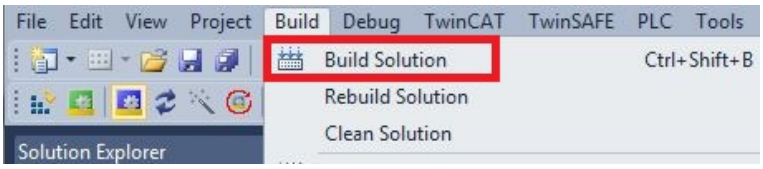
Create variables as follows:

1. Create the variables in your PLC project in a VAR RETAIN area.

```
PROGRAM MAIN

VAR RETAIN
    x       :UINT;
    y       :UINT;
    z       :UINT;
END_VAR

VAR

    datain  AT%I*: REAL;
    dataout AT%Q*: BYTE;

END_VAR
```

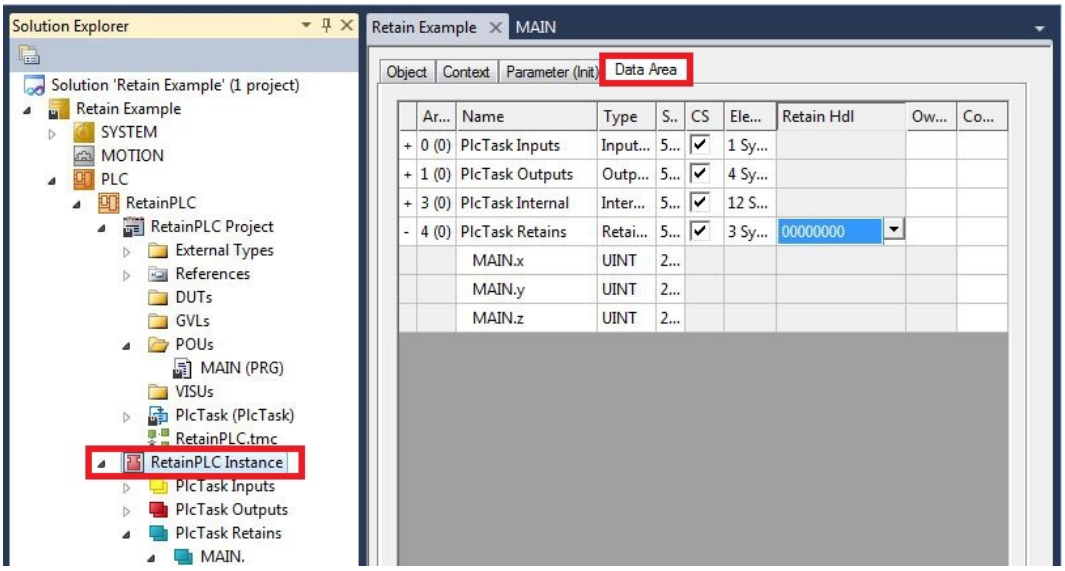
2. For self-defined Data Unit Types (DUTs), use the keyword `STRUCT RETAIN`.

3. Click on **Build** in the toolbar at the top, then on **Build Solution**

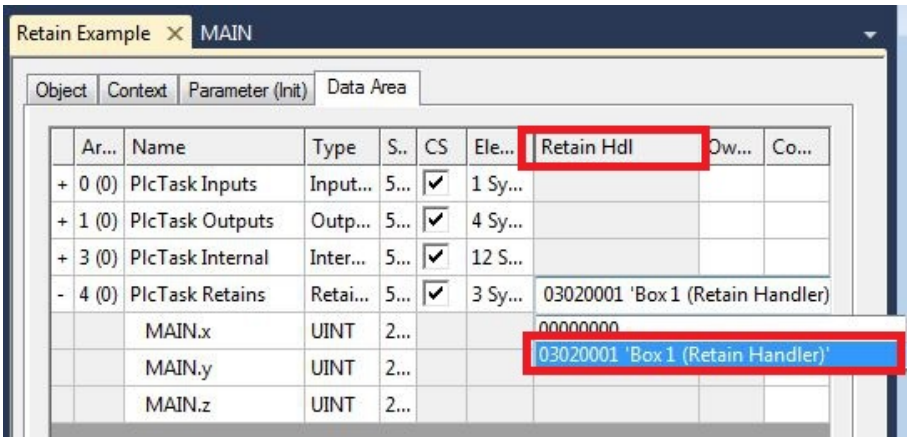


.

4. Click on your **PLC Instance** in the tree view on the left and then on the tab **Data Area**.

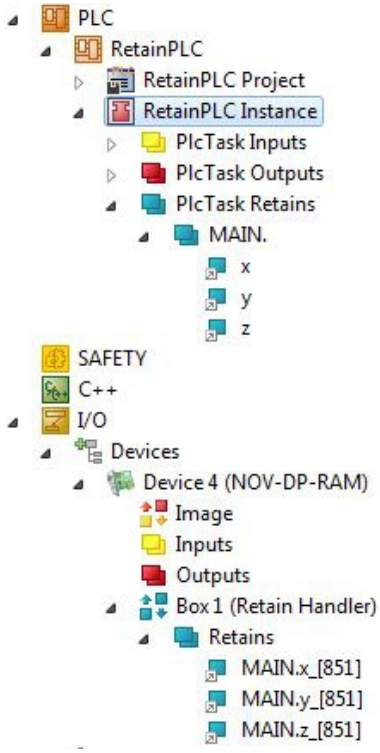


5. Under **Retain Hdl**, select the Retain Handler that you have created.



⇨ After selecting a Retain Handler as the target, the symbols are linked in the tree view and a mapping is created.
In the tree view, the PLC variables are created under the Retain Handler and linked to the PLC instance

variables.



An existing link is indicated by an arrow icon.

7.3.3 Deleting variables under the Retain Handler

If variables are deleted from the PLC, the link with the Retain Handler is cancelled. However, the variables continue to be shown under the Retain Handler and are not deleted automatically.

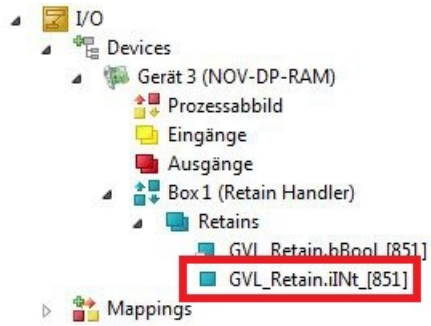
Under TwinCAT 3 the variables have to be deleted manually.

Prerequisites for this step:

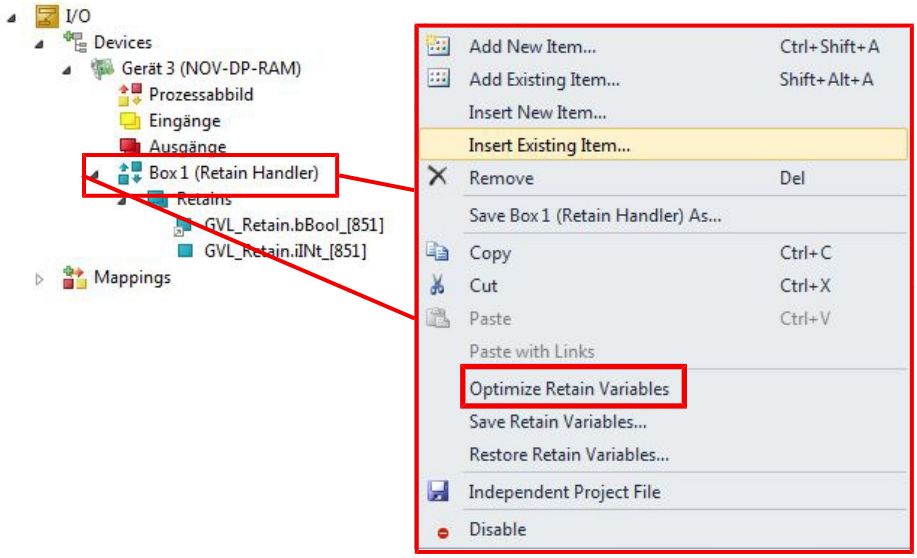
- Variables declared with VAR_RATAIN were deleted from the PLC.

Delete the variables under the Retain Handler as follows:

1. The variable GVL_Retain.iNt under the Retain Handler is to be deleted.



2. Right-click on the **Retain Handler** in the tree view on the left.
3. In the context menu click on **Optimize Retain Variables**.



⇨ The variable under the Retain Handler is deleted.

7.4 Software configuration

7.4.1 User name and password

In the delivery state, the CX7000 has a preset user name with password, which is necessary for logging in to TwinCAT or the Beckhoff Device Manager.

- User name: `Administrator`
- Password: `1`

The user name is fixed and cannot be changed. It is also not possible to add another user name. The preset password can be changed via the Beckhoff Device Manager (see: Starting the Beckhoff Device Manager). The password can contain a maximum of 32 characters. Numbers, letters and special characters are allowed and a distinction is also made between upper and lower case letters.

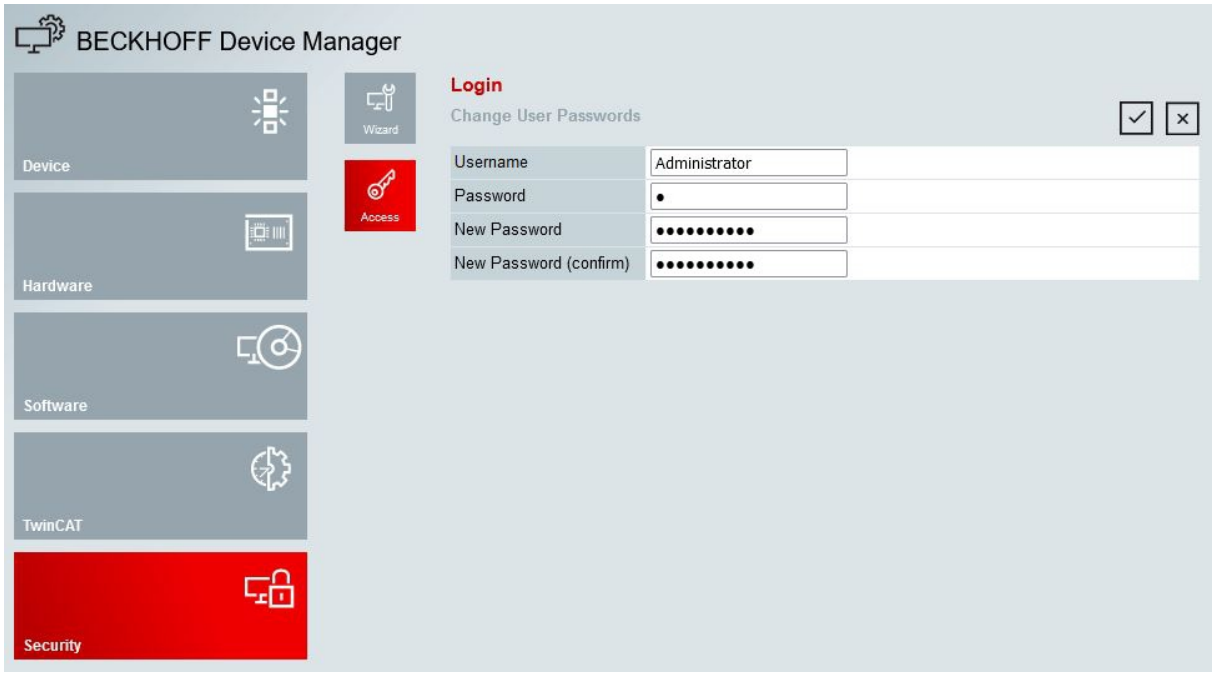


Fig. 25: Changing the password in the Beckhoff Device Manager.

You can restore the delivery state and preset password by removing the MicroSD card, accessing the MicroSD card with a card reader and deleting the `device.conf` file in the `/etc` folder. The password cannot be reset without physical access to the CX7000 and thus to the MicroSD card.

7.4.2 Setting the IP address

DHCP is enabled by default for the CX7000. Without a DHCP server, the CX7000 uses a local IP address in the address range 169.254.x.x

In the case of the CX7000 Embedded PC, there are several ways to set the IP address. One way is to call the Beckhoff Device Manager and set the IP address for the CX7000 in the browser (see: Starting the Beckhoff Device Manager).

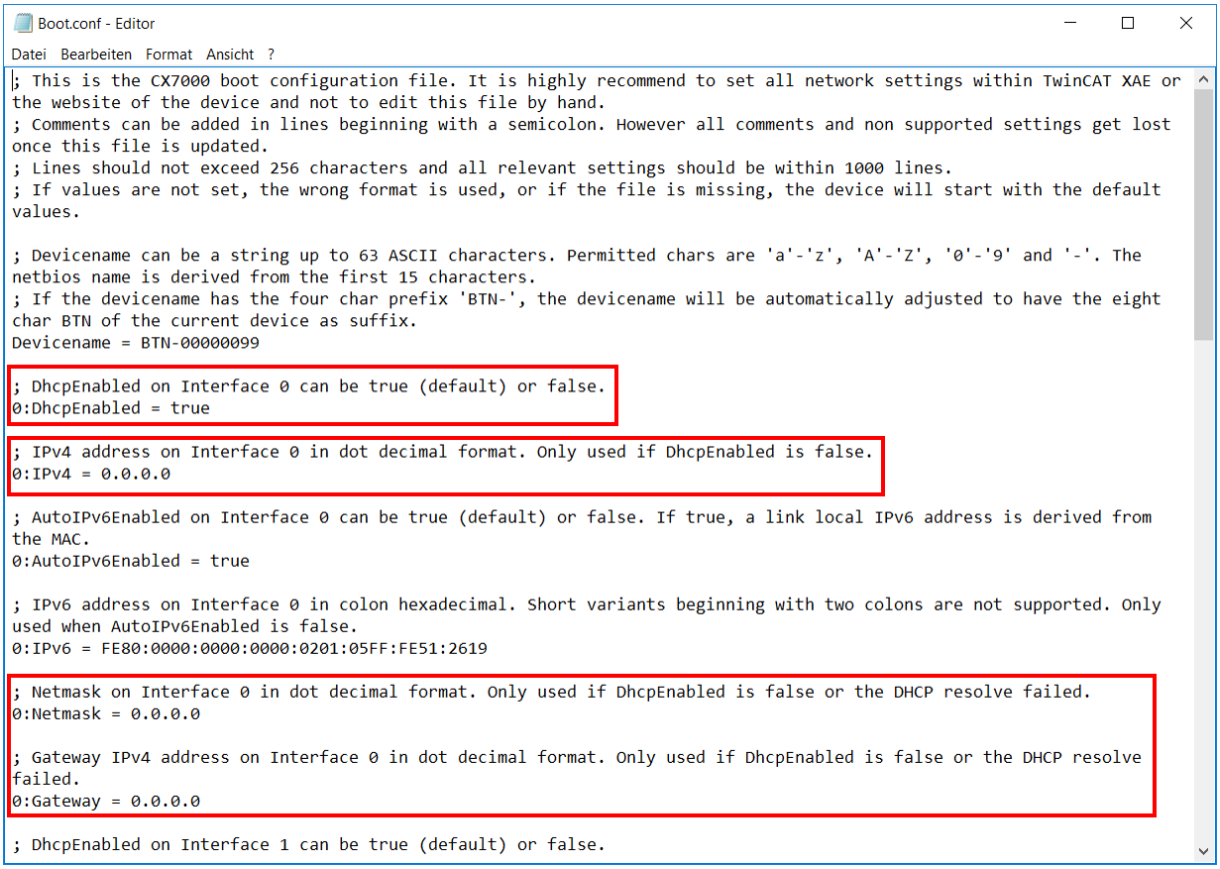
Another way to set the IP address is offered by the boot.conf file, which is created on the MicroSD card after the first start. This step shows you how to set the IP address in the boot.conf file.

Requirements:

- MicroSD card reader

Proceed as follows:

1. Switch the Embedded PC off and remove the MicroSD card from the Embedded PC.
2. Open the Boot.conf file under \etc



3. Set the **DhcpEnabled** entry to **false**.
4. Assign an IP address under **IPv4**.
5. Make the settings for subnet mask, gateway and DNS server.

⇨ Save the changes and install the MicroSD card in the Embedded PC again. The settings are effective after startup.

7.4.3 Configure virtual local area network (VLAN)

From image version 140501, a virtual local area network (VLAN) can be configured on the Ethernet interface (X001). The VLAN forms a logical network segment and its own broadcast domain. This means that the network is subdivided regardless of the physical network topology on-site. The data traffic of a VLAN's associated devices is isolated from other VLANs.

The configuration takes place in the file `/etc/boot.conf` on the microSD card. Update the image first if you are using an older image version on the embedded PC (see: Update image [▶ 60]).

Requirements:

- card reader for microSD cards

Proceed as follows:

1. Switch the embedded PC off and take the microSD card out of the embedded PC.
2. Insert the microSD card into an external card reader and open the microSD card's folder tree.
3. Open the `boot.conf` file under `\etc` and navigate to the entries for the VLAN.

```
; VLanEnabled on Interface 0 can be true or false (default). If true, VLAN Tagging is supported.
0:VLanEnabled = false
; VLanID on Interface 0 can be set from 0 to 4094. When VLAN is enabled, it defines the
operating VLAN.
0:VLanID = 1
; VLanPriority on Interface 0 can be set from 0 to 7. When VLAN is enabled, it defines the
priority of the packets, sent with a VLAN Tag. 0 has a higher priority than 7.
0:VLanPriority = 0
```

4. Edit the parameters `VLanEnabled`, `VLanID` und `VLanPriority` according to your requirements.

⇨ Save the changes and reinstall the microSD card in the embedded PC. The settings will take effect after the device is started.

7.4.4 Update image

NOTICE

Failure of the power supply

The bootloader may be corrupted if the update is interrupted. The CX70x0 thus becomes unusable and must be sent in for repair. Ensure a stable power supply during initial start-up and do not interrupt the update.

The new image will be copied directly to the MicroSD card in order to update the image of the Embedded PC. The new image is made available by Beckhoff Service. Perform the update only after consulting with Beckhoff Service.

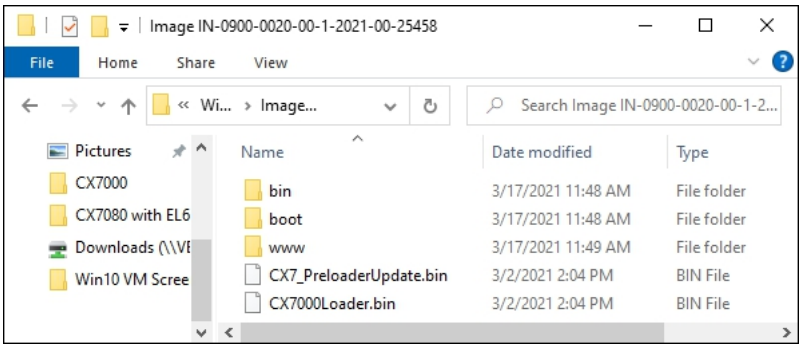
Requirements:

- Card reader for MicroSD cards.

Update the image as follows:

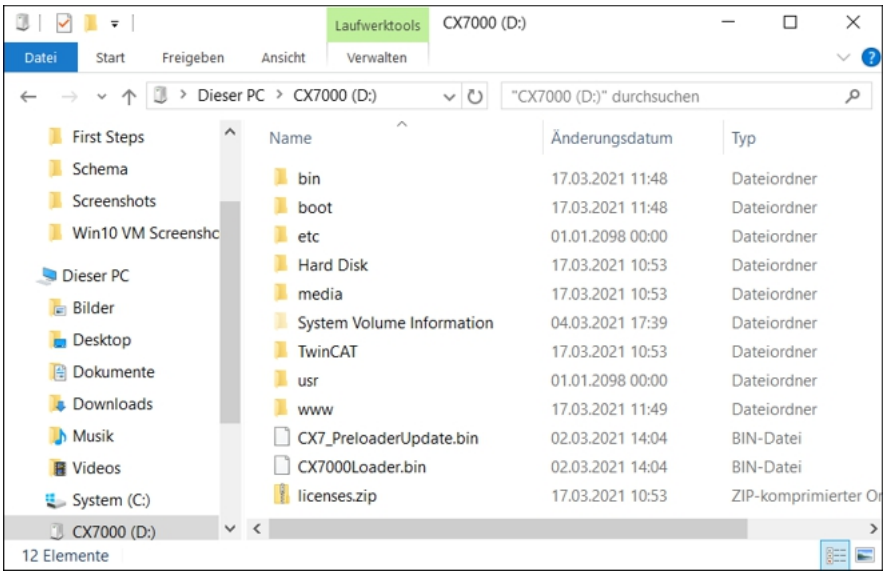
1. Switch the Embedded PC off and remove the MicroSD card from the Embedded PC.
2. Insert the MicroSD card into an external card reader and open the MicroSD card's folder tree.
3. Delete all files and folders on the MicroSD card.

4. Copy all files and folders of the new image to the empty MicroSD card.



5. Re-install the MicroSD card in the Embedded PC and start the Embedded PC.

⇨ The Embedded PC is started and saves the current hardware configuration. New folders are created, such as Hard Disk or TwinCAT. The image has now been successfully updated.



7.4.5 Updating the firmware for multifunction I/Os

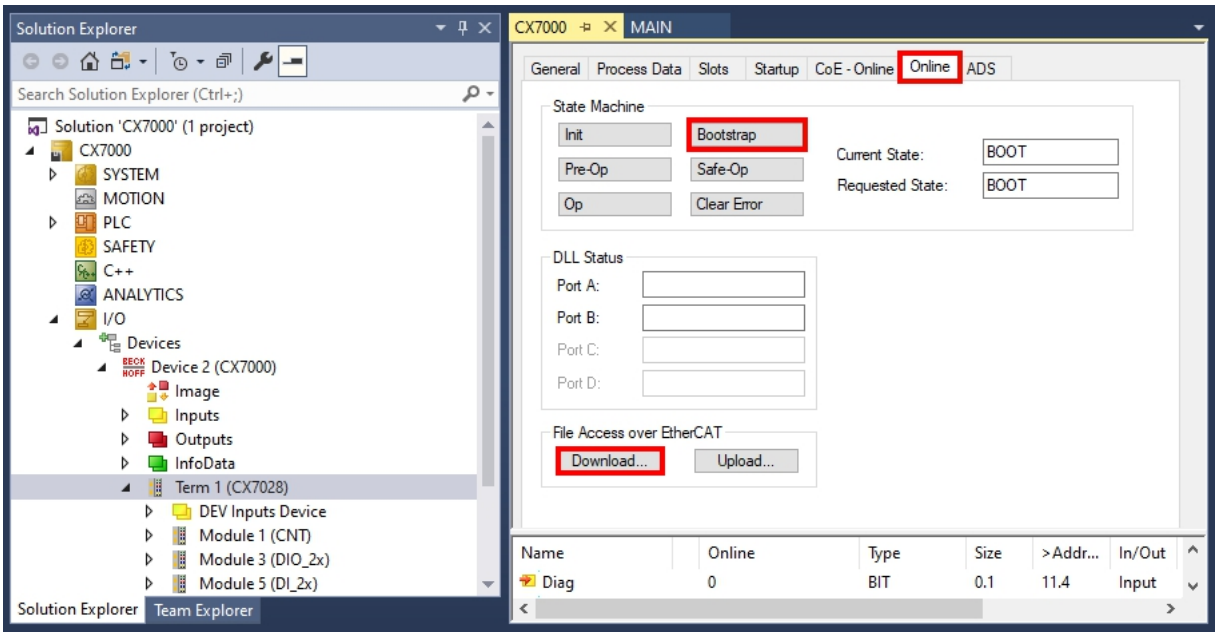
This step shows you how to update the firmware of the multifunction I/Os. The firmware is provided by Beckhoff Service and the update is carried out in TwinCAT.

Requirements:

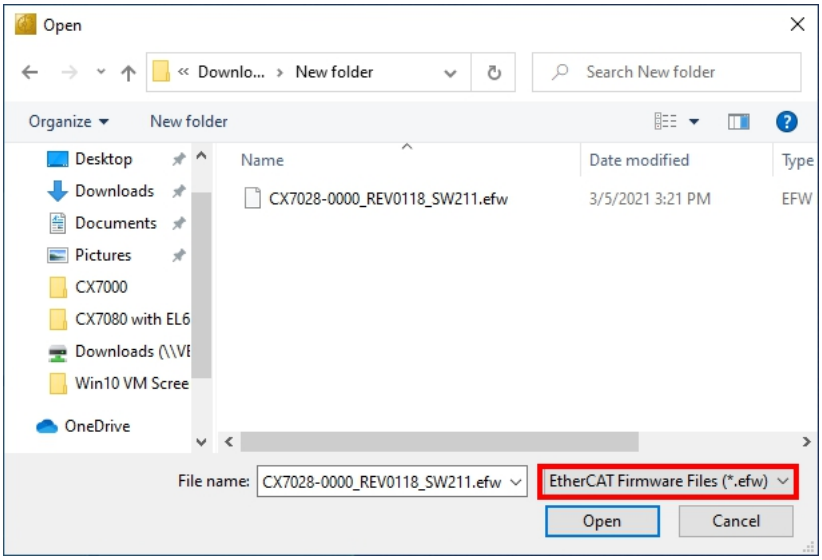
- EtherCAT firmware file (*.efw)

Proceed as follows:

1. Start TwinCAT in configuration mode (config mode).
2. On the left in the structure tree, click the CX7028 device and then click the **Online** tab.



3. Click the **Bootstrap** button to switch the multifunction I/Os to the bootstrap state.
4. Click the **Download** button and select a current efw file.



⇨ The update takes about 3 to 4 minutes. A progress bar indicates the progress of the update. Do not switch the CX7000 off during this time.

When the update is complete, return to the Operational (Op) state by clicking the **Op** button.

7.4.6 Updating the ESI device description

The TwinCAT System Manager and the TwinCAT EtherCAT Master require the device description files of all EtherCAT devices for configuration in online and offline mode. These device descriptions are the so-called ESI files (EtherCAT Slave Information) in XML format. These files can be requested from the respective vendor and are made available for download. An *.xml file may contain several device descriptions.

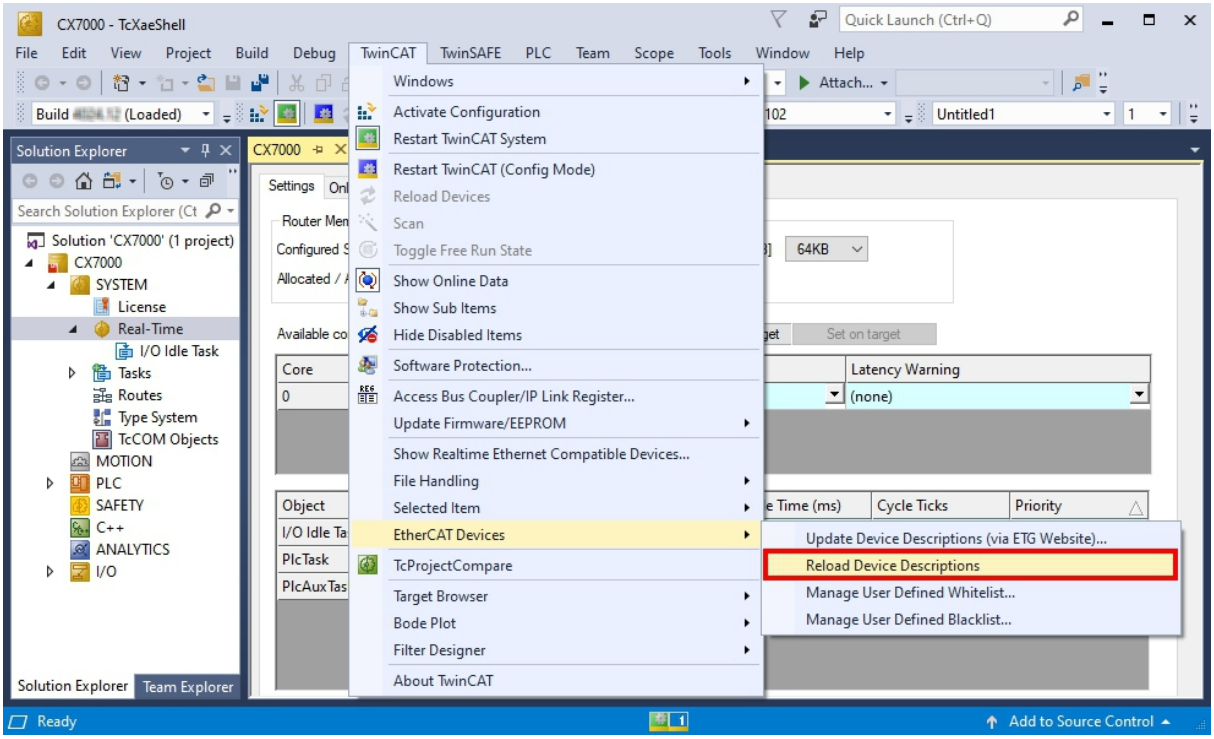
ESI files for Beckhoff EtherCAT devices are provided at https://www.beckhoff.com.

Requirements:

- ESI file for the CX7000 in XML format.
- If necessary, the associated *.xsd file, which describes the structure of the XML file.

Proceed as follows:

1. Copy the ESI file into the TwinCAT installation directory: *\TwinCAT\3.1\Config\Io\OnboardIo*.
2. Create the folder manually if it doesn't exist.
3. Open TwinCAT and click in the menu under **TwinCAT > EtherCAT Devices** on **Reload Device Description**.



⇨ The ESI file is re-read into TwinCAT. An error is returned if there is a faulty ESI file. Check whether the structure of the *.xml corresponds to the associated *.xsd file or whether the files match the CX7000.

8 TwinCAT

8.1 First Steps

8.1.1 Connect to the CX70x0

Before you can configure the CX7000 in TwinCAT, you must establish a connection between your engineering computer and the CX7000 (target system). The engineering computer and the Embedded PC must be in the same network and subnet or alternatively connected directly via an Ethernet cable (peer-to-peer).

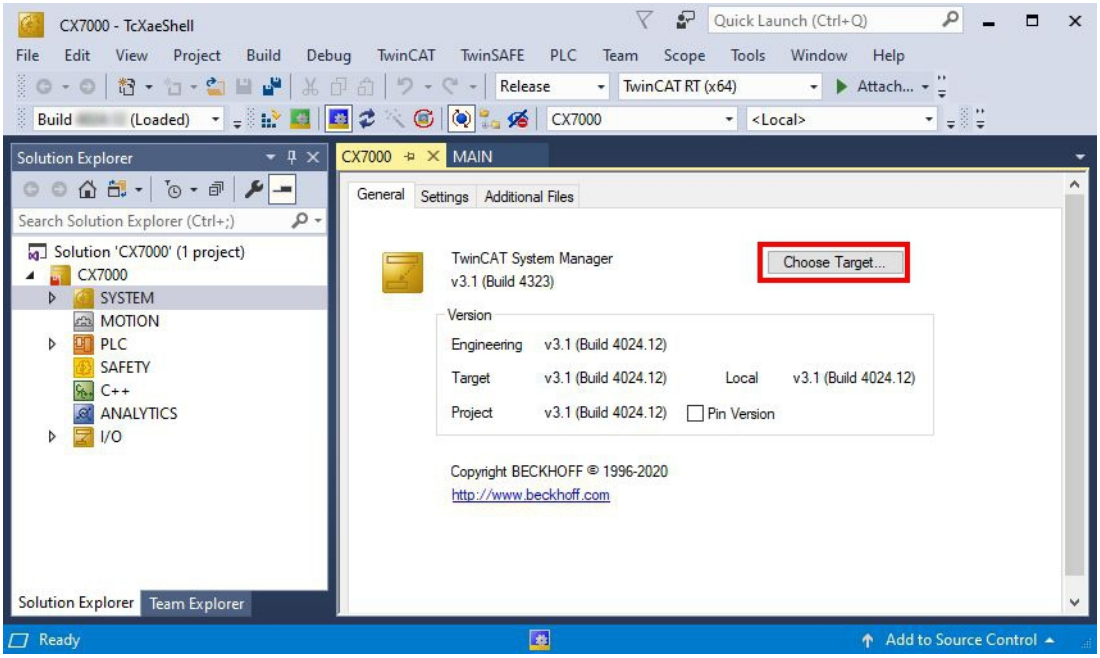
The IP address or host name of the CX7000 is required for the connection.

Requirements:

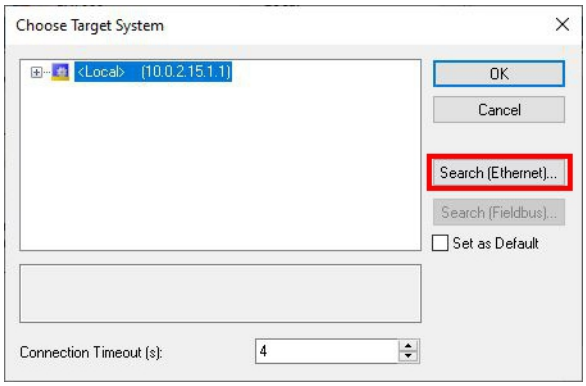
- TwinCAT must be in Config mode.
- IP address or host name of the Embedded PC.

Establish a connection as follows:

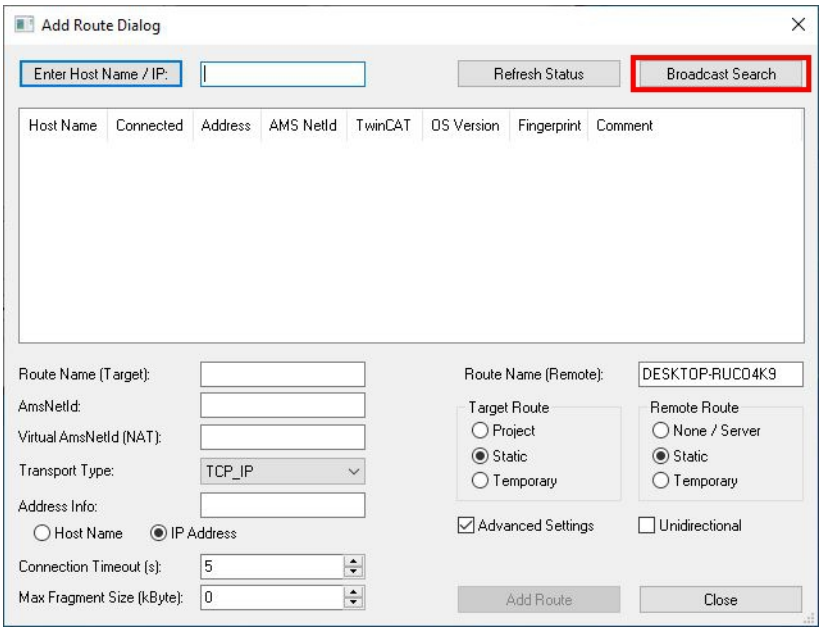
1. In the menu at the top click on **File > New > Project** and create a new TwinCAT XAE project.
2. In the tree view on the left click on **SYSTEM**, and then **Choose Target**.



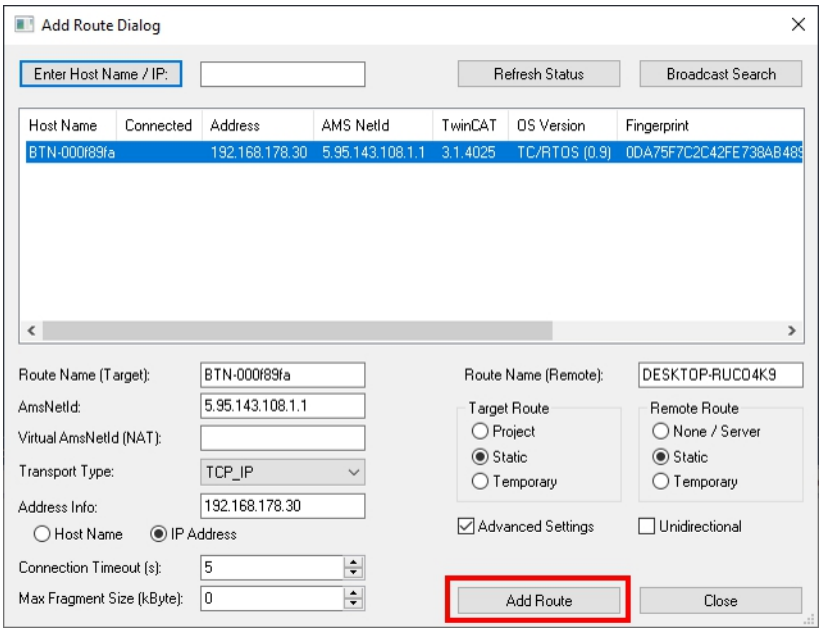
3. Click on **Search (Ethernet)**.



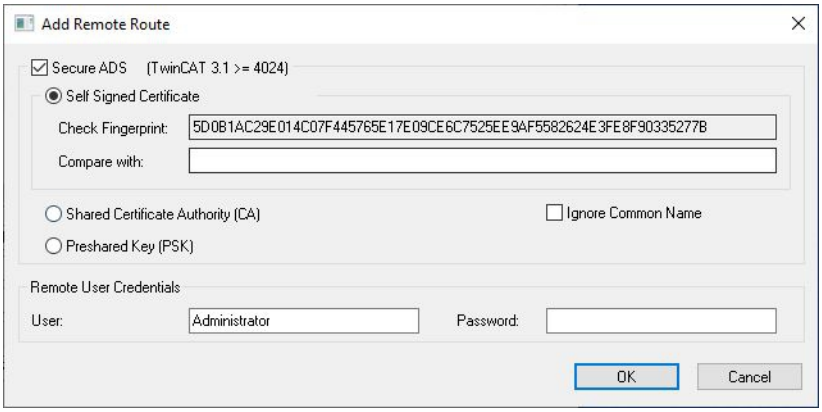
4. Click **Broadcast Search** and search for available devices on the network.



5. Mark the appropriate CX7000 and click **Add Route**. The host name and IP address facilitate identification.

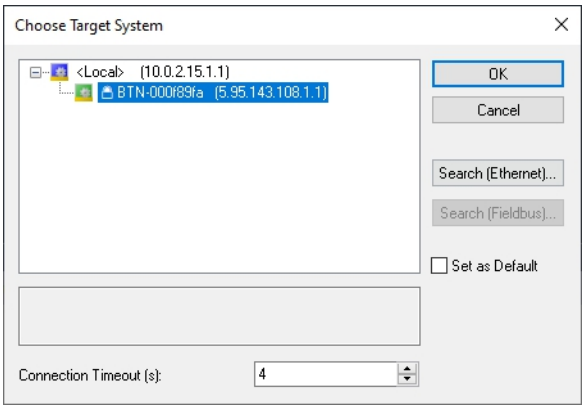


6. Enter the user name and password in the **User** and **Password** fields respectively and click **OK**. User name: `Administrator` Password: `1`

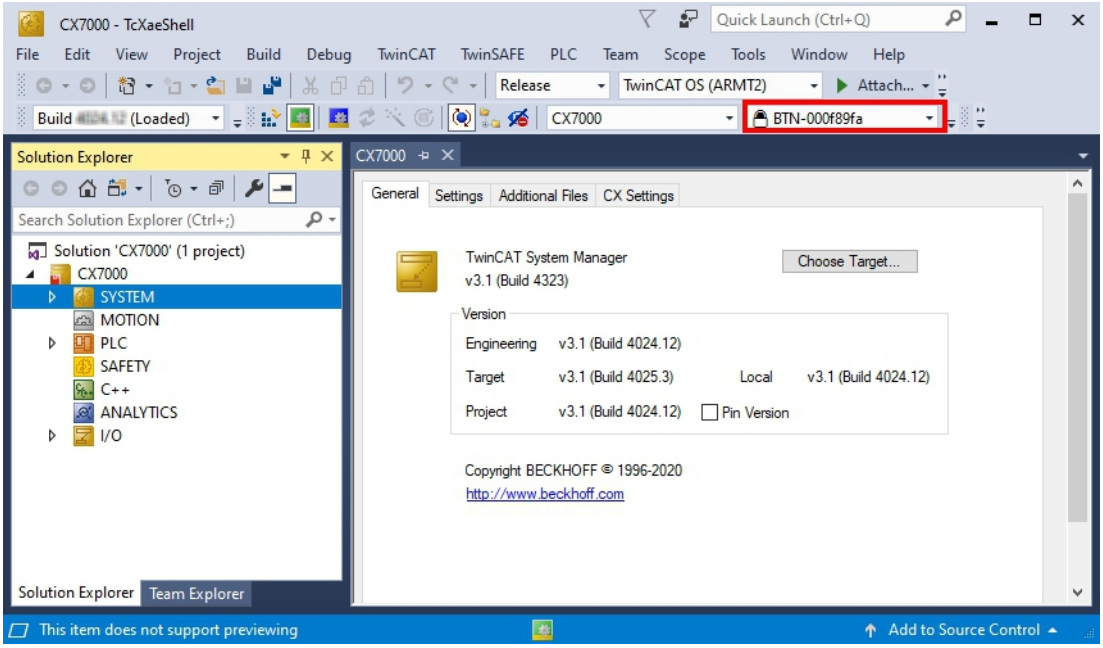


7. The new device is displayed in the **Choose Target System** window.

8. Select the device you want to specify as target system and click **OK**.



⇨ You have successfully established a connection between your engineering computer and the CX7000 (target system) in TwinCAT. The new target system and the host name are displayed in the menu bar.



Using this procedure you can search for all available devices and also switch between the target systems at any time.

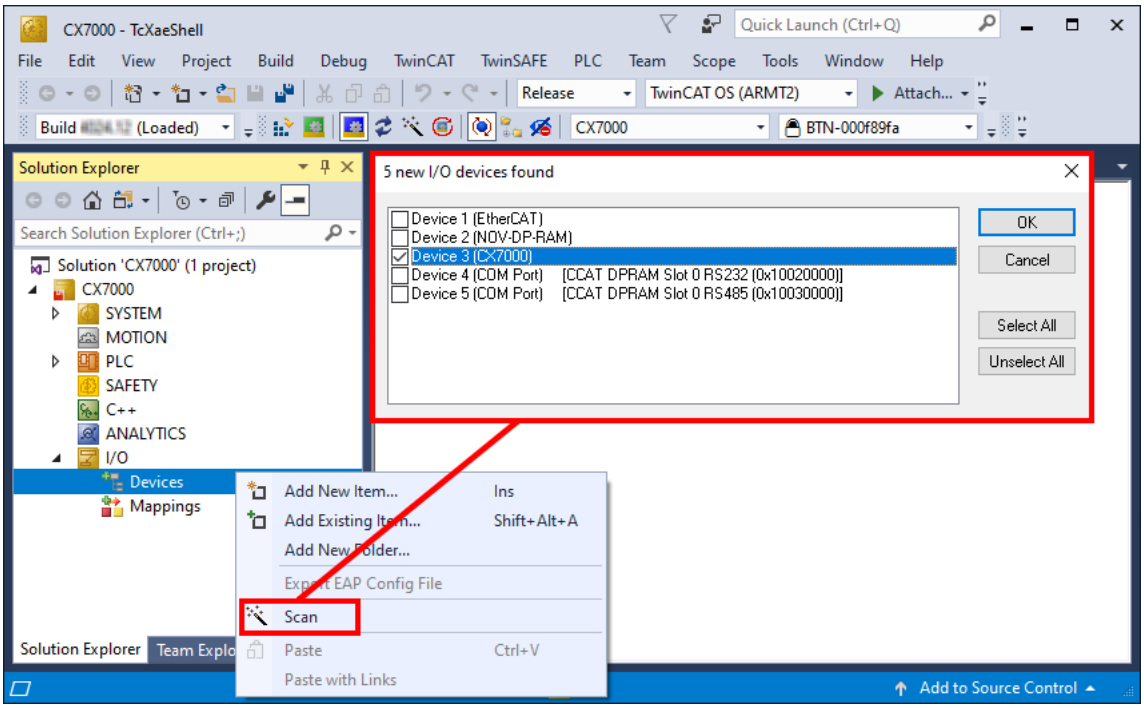
8.1.2 Scan multifunction I/Os

Special features of the CX7000 series are the eight integrated multifunction inputs and four integrated multifunction outputs. This chapter shows how to scan and create the multifunction I/Os in TwinCAT.

Note that the CX7028 interface for controlling the multifunction I/Os has its own CPU and the CX7028 interface is not displayed or does not work under TwinCAT if the power supply(Up) is not connected.

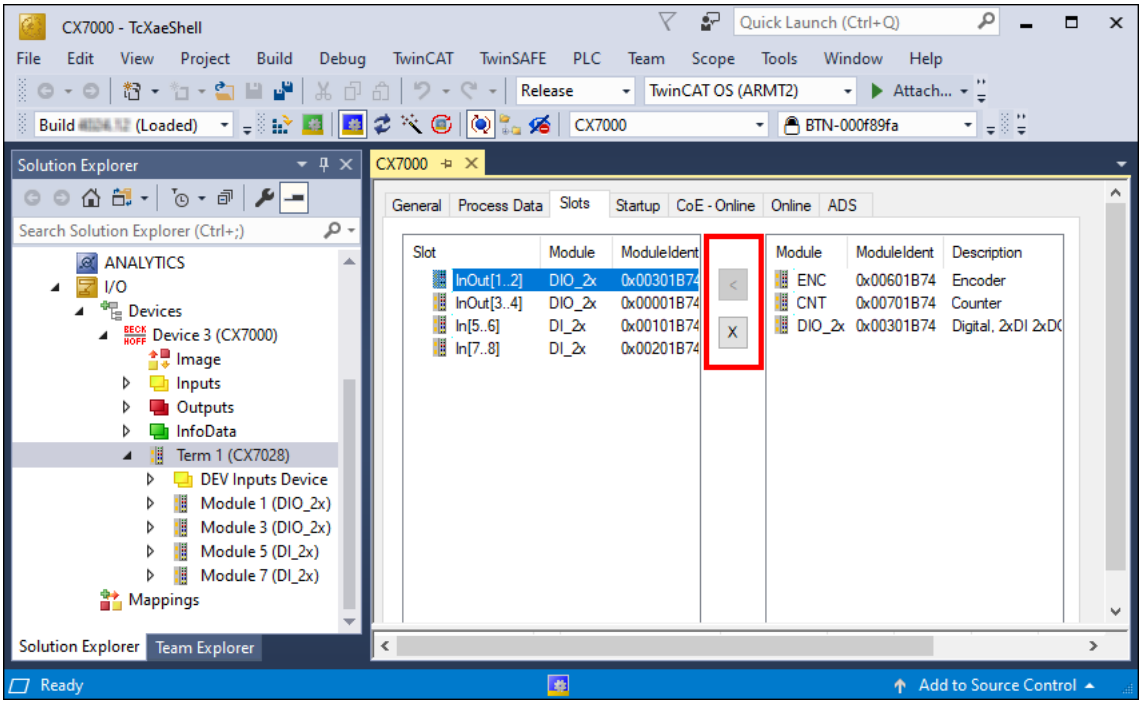
Proceed as follows:

1. On the left side of the tree view, right-click **Devices** and then click **Scan**.



All available I/O devices are displayed.

2. Select the appropriate I/O devices. For this example, at least the CX7028 interface, i.e. the CX7000 device, must be selected. If you still want to operate Bus or EtherCAT Terminals on the CX7000, then you must also select EtherCAT as a device.
3. A total of four slots are created. For each slot a maximum of one module (DI, DIO, ENC, CNT or PWM) can be assigned, which in turn determines the operation mode for the respective slot.



4. Modules can be assigned to a specific slot with the button **<** or removed again with **x**.

⇨ Define the required modules according to their requirements. There is a choice of different modules depending on the slot used. Which modules are supported by which slot is listed in the chapter Multifunction I/Os [▶ 28].

8.1.3 Establishing ADS communication

This chapter shows you how to connect a CX7000 to another CX70x0 or any TwinCAT controller. The ADS protocol provides the simplest way to connect two TwinCAT systems to each other. With the ADS protocol, data can be both read and written. ADS function blocks are normally used for communication; these are included in the Tc2_System library. In the following example, data are to be written to and read from a memory area.

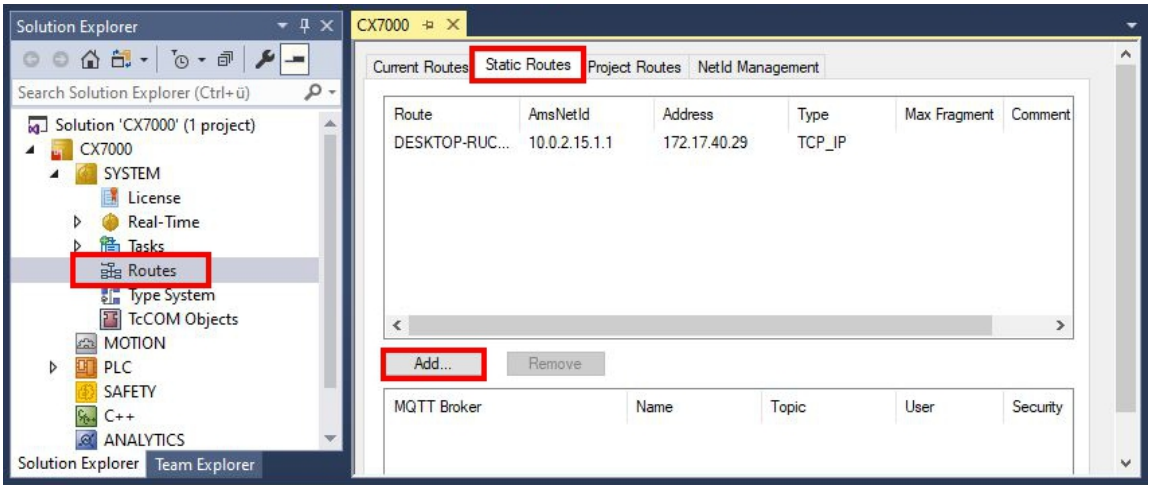
In order to set up an ADS connection, an ADS route is created first. Communication then takes place via Ethernet and data exchange via the TCP/IP protocol. The ADS route is then the interface between the ADS and TCP/IP connection. The ADS route indicates which AmsNetId is assigned to which TCP/IP address. As a result, the ADS function blocks no longer use the TCP/IP address, but the AmsNetId.

Requirements:

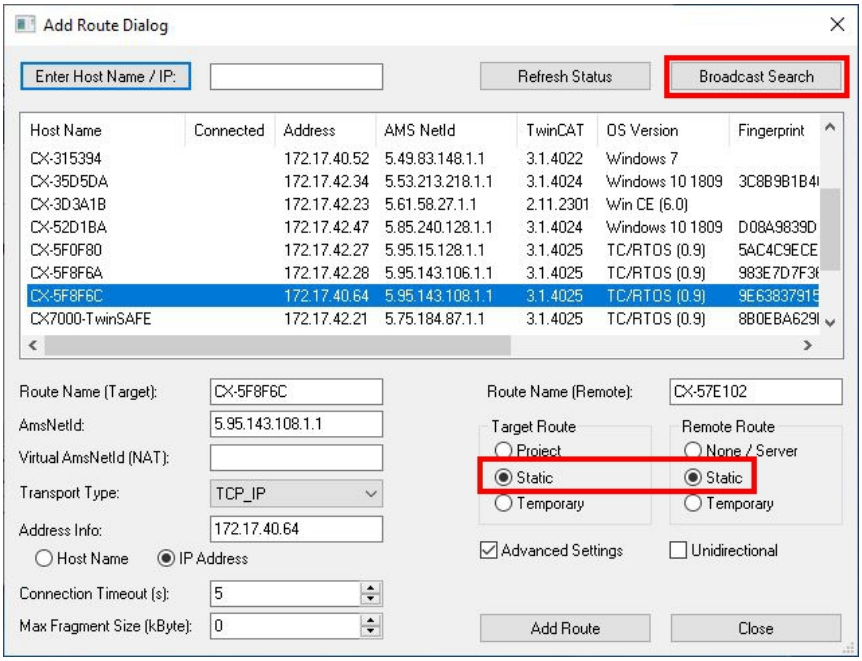
- Two CX70x0 Embedded PCs.
- Both CX70x0s are in the same network and accessible via ADS.

Proceed as follows:

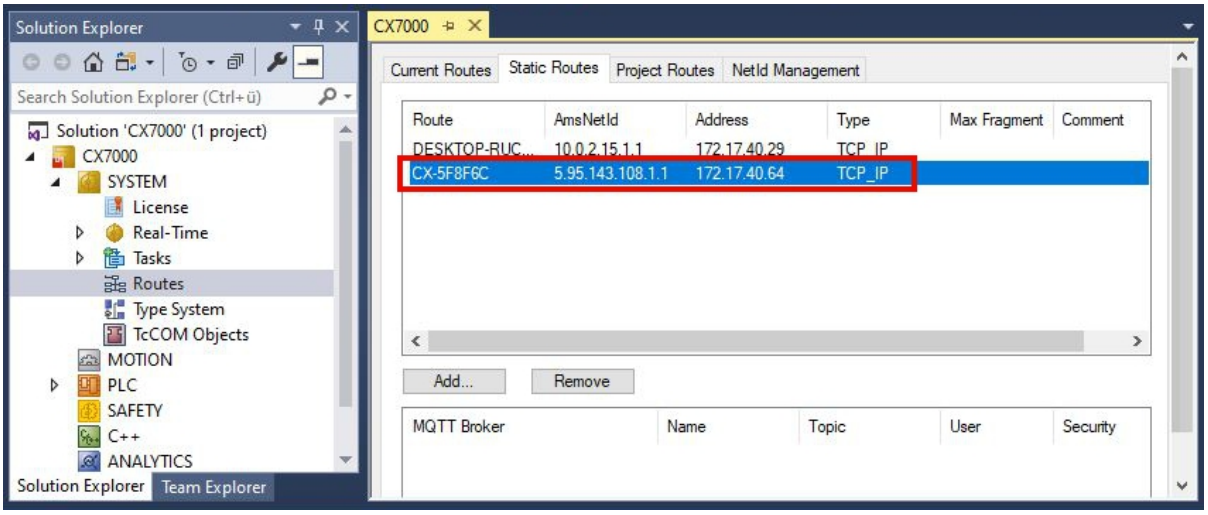
1. Start TwinCAT and connect to the first CX70x0 (see: Connect to the CX70x0 [▶ 64]).
2. On the left in the tree view, click **Routes**, select the **Static Routes** tab, and click the **Add** button.



3. Under **Remote Route,** select the **Static** option so that the ADS route remains in the project, and then click the **Broadcast Search** button.



4. Select the second CX70x0 as the destination of the ADS route. The ADS route is entered for both Embedded PCs. The AmsNetId of the second CX70x0 is displayed and can be used in the program for ADS function blocks.



5. Now connect to the second CX70x0, which has been set as the destination of the ADS route, and write a small program. Define an array and increment a value of the array.

```
VAR
    MarksTest AT %MB0     : ARRAY[0..9] of INT;
END_VAR

Program:
MarksTest[0]:=MarksTest[0]+1;
```

6. Activate the configuration and switch the CX70x0 to Run mode.
7. For the first CX70x0, write a program that reads the incremented value of the array.

```
VAR
    ADSREAD : ADSREAD;
    NetID : STRING:='5.81.38.23.1.1'; (* AMSNetId of the target*)
    Value : INT; (* value of target MarksTest[0]*)
    Error : INT;
    NoError : INT;
END_VAR

Program:
    ADSREAD(
        NETID:=NetID ,
        PORT:=851 , (* plc port of the target*)
        IDXGRP:=16#4020 , (* Marks %MB*)
        IDXOFFS:=0 , (* Marks offset in byte*)
        LEN:=2 , (* length of data in byte*)
        DESTADDR:=ADR(Value) , (* pointer to the data in which the value is to be stored *)
        READ:=TRUE ,
        TMOUT:= ,
        BUSY=> ,
        ERR=> ,
        ERRID=> );
    IF NOT ADSREAD.BUSY THEN
        IF NOT ADSREAD.ERR THEN
            NoError:=NoError+1;
        ELSE
            Error:=Error+1;
        END_IF
    ADSREAD(Read:=FALSE);
    END_IF
```

8. The incremented value is read out and transmitted to the first CX70x0.

⇨ You should see on the first CX70x0 how the value of the `Value` variable is incremented. The writing of the data works in the same way. Data can be written with the `ADSWRITE` function block. Make sure that you set the offset (`IDXOFFSET`) to 10 in this sample setup so that the array [4... 9] is written. Limit the length to 10 bytes, as an array of 0... 9 of type INT was created and the memory thus uses %MB0... MB19 (10 * 2 bytes) (The elements 0...4 for reading the array and the elements 5...9 for writing it).

Use one ADS command at a time. Wait until the ADS service is finished, i.e. the `BUSY` output of the

function block is switched to `FALSE`, and only then use the next ADS function block. To optimize the access timing, you can also use an `ADSREADWRITE` function block that reads and writes the data at the same time.

8.1.4 Creating a PLC project

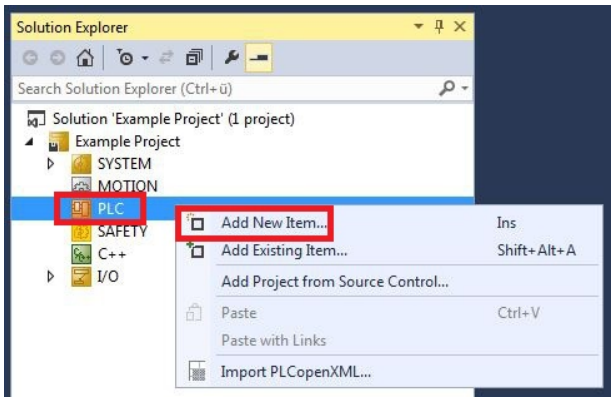
The next steps describe how to create a PLC project in TwinCAT and add it in the tree view.

Prerequisites for this step:

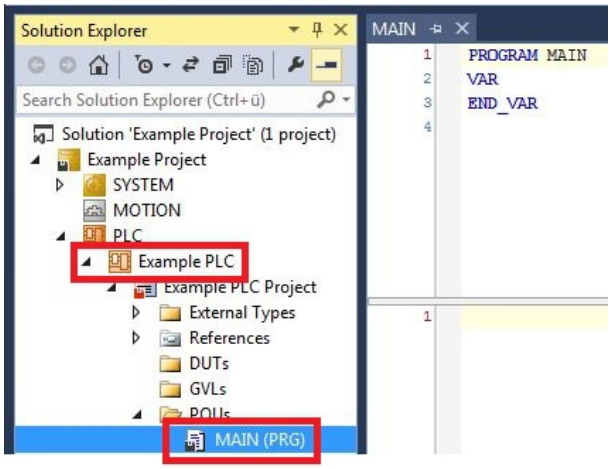
- A newly created TwinCAT XAE project.

Create a PLC project as follows:

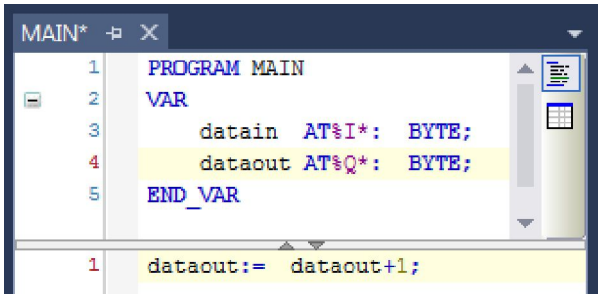
1. Right-click on **PLC** in the tree view.
2. In the context menu click on **Add New Item** and select the **Standard PLC Project**.



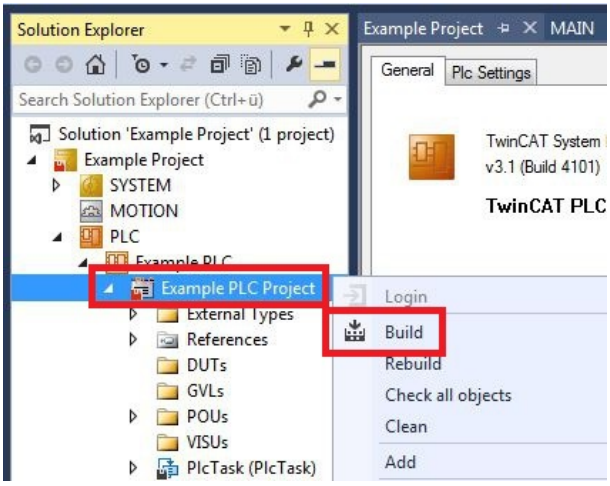
3. In the tree view click on the newly created PLC project, then double-click on **MAIN (PRG)** under **POUs**.



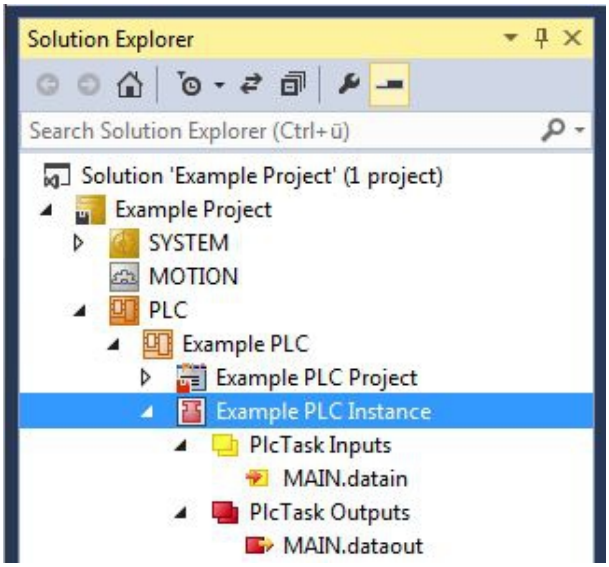
4. Write a small program, as shown in the diagram below.



5. In the tree view right-click on the PLC project, then click on **Build** in the context menu.



⇨ You have successfully created a PLC project and added the project in TwinCAT. A PLC instance with the variables for the inputs and outputs is created from the PLC project.



In the next step you can link the variables with the hardware.

8.1.5 Linking variables

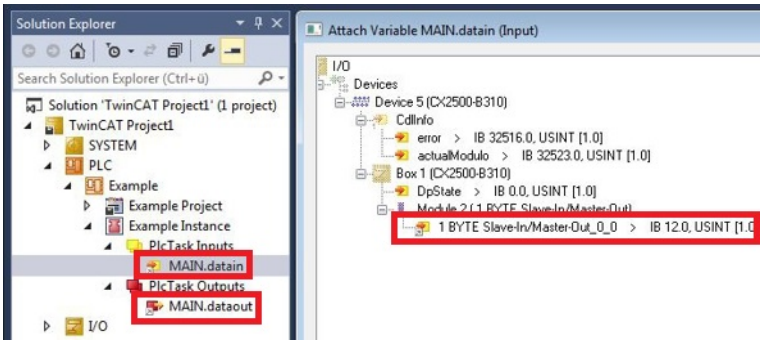
Once the PLC project was successfully added in the System Manager, you can link the newly created input and output variables from the PLC project with the inputs and outputs of your hardware.

Prerequisites for this step:

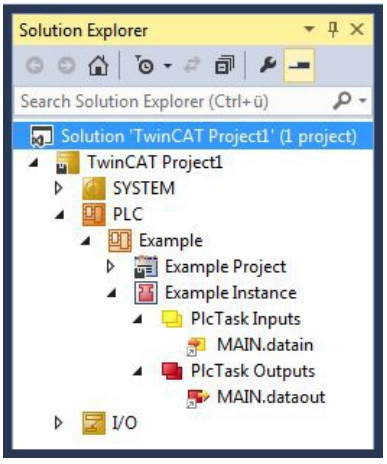
- A PLC program attached in TwinCAT.

Link the variables as follows:

1. Double-click on the input or output variables in the tree view under **PLC**.
   The **Attach Variable** window appears and shows which inputs or outputs can be linked with the variables from the PLC project.

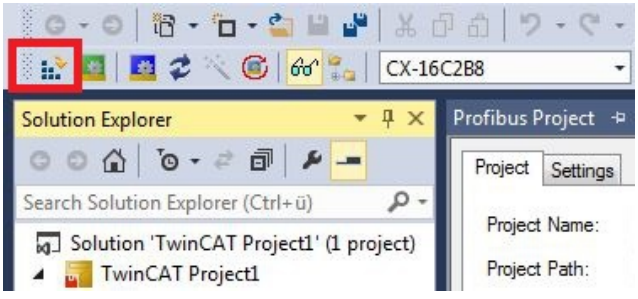


2. Double-click on the inputs or outputs of the hardware in the **Attach Variable** window.
   Link the input variables with the inputs and the output variables with the outputs of the hardware.



   Variables that are already linked are indicated with a small arrow icon in TwinCAT.

3. In the toolbar click on **Activate Configuration**.



4. Confirm the request whether TwinCAT is to start in Free Run mode with **Yes**.

⇨ You have successfully linked variables with the hardware. Use Activate Configuration to save and activate the current configuration.

The configuration can now be loaded on the CX, in order to automatically start TwinCAT in Run mode, followed by the PLC project.

8.1.6 Load configuration to CX

Once variables are linked, the configuration can be saved and loaded on the CX. This has the advantage that the PLC project is loaded and started automatically when the CX is switched on. The start of the previously created PLC project can thus be automated.

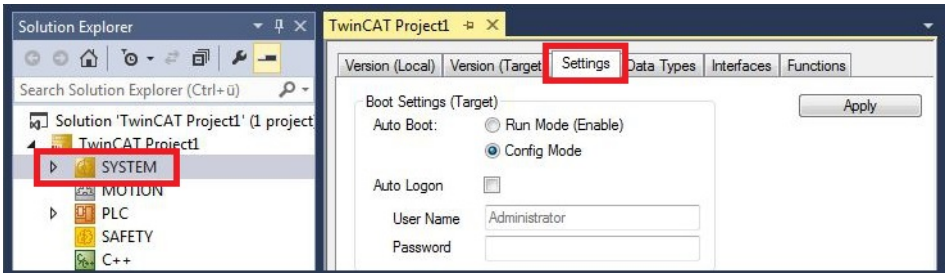
Prerequisites for this step:

- A completed PLC project, added in the System Manager.
- Variables from the PLC project, linked with the hardware in the System Manager.
- A CX selected as target system.

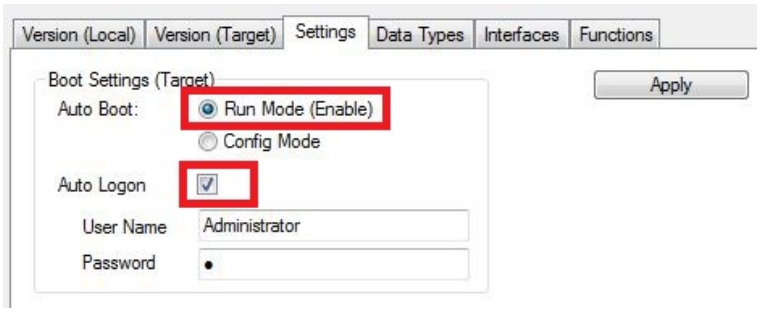
Load the configuration from the System Manager to the CX as follows:

1. In the tree view on the left click on **SYSTEM**.

2. Click on the **Settings** tab.



3. Under Boot Settings select the option **Run Mode (Enable)** and tick the **Auto Logon** checkbox.

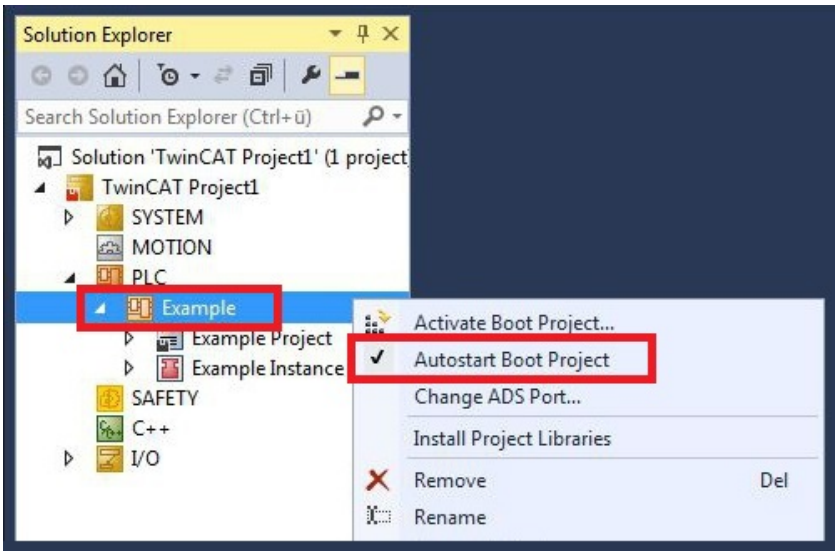


4. Enter the user name and password for the CX in the **User Name** and **Password** fields.

5. Click on **Apply**.

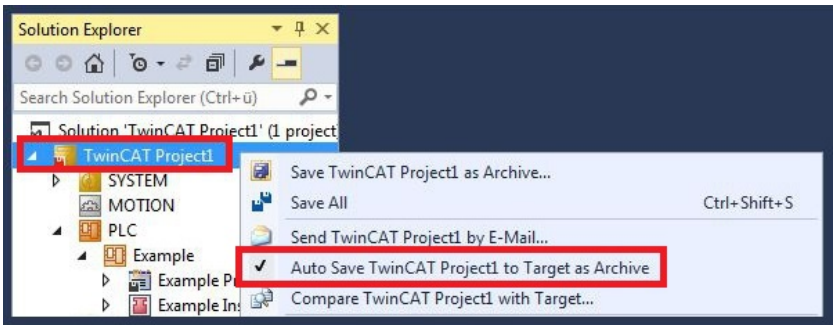
6. In the tree view on the left right-click on the PLC project under **PLC**.

7. In the context menu click on **Autostart Boot Project**.
   The setting is selected



8. Right-click on the project folder in the tree view.

9. In the context menu click on **Auto Save to Target as Archive**.
   The setting is selected.



⇨ You have successfully loaded the CX configuration. From now on, TwinCAT will start in Run mode and the PLC project will start automatically.

Next, the master can be added in a new project in the System Manager and can then be used to find slaves that have already been set up.

8.2 Reading the IP and MAC addresses

This sample shows you how to read the IP and MAC addresses. The function block FB_MDP_NIC_Read can be used to retrieve information from the network adapter.

Sample

```
Var
    FB_MDP_NIC_Read    : FB_MDP_NIC_Read;
END_VAR

PROGRAM:
FB_MDP_NIC_Read(
    bExecute:=TRUE ,
    tTimeout:= ,
    iModIdx:= ,
    sAmsNetId:= ,
    bBusy=> ,
    bError=> ,
    nErrID=> ,
    iErrPos=> ,
    stMDP_ModuleHeader=> ,
    stMDP_ModuleContent=> );
```

The output stMDP_ModuleHeader displays the header information. The output stMDP_ModuleContent displays, among other things, the information about the IP and MAC addresses.

stMDP_ModuleHeader	ST_MDP_ModuleHea...	
iLen	UINT	4
nAddr	DWORD	131072
sType	T_MaxString	'Nic'
sName	T_MaxString	'st'
nDevType	DWORD	141072
stMDP_ModuleContent	ST_MDP_NIC_Prope...	
iLen	UINT	8
sMACAddress	T_MaxString	'00:01:05:5f:0f:7a'
sIPAddress	T_MaxString	'169.254.123.15'
sSubnetMask	T_MaxString	'255.255.0.0'
bDHCP	BOOL	TRUE
iReserved	BYTE	0

Fig. 26: Content of the MDP module with IP and MAC address.

8.3 Virtual Ethernet interface

The virtual Ethernet interface integrates network adapters into the TwinCAT system. This makes it possible to establish a virtual Ethernet communication via ADS, TCP or UDP to a BK9xx0. Do not use more than two BK9xx0 and a cycle time > 50 ms.

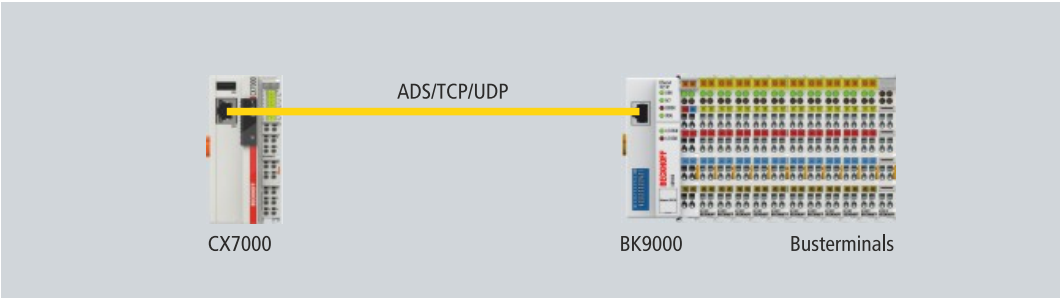
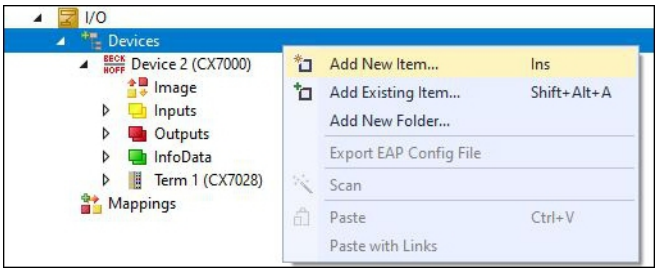


Fig. 27: Virtual Ethernet communication via ADS, TCP or UDP.

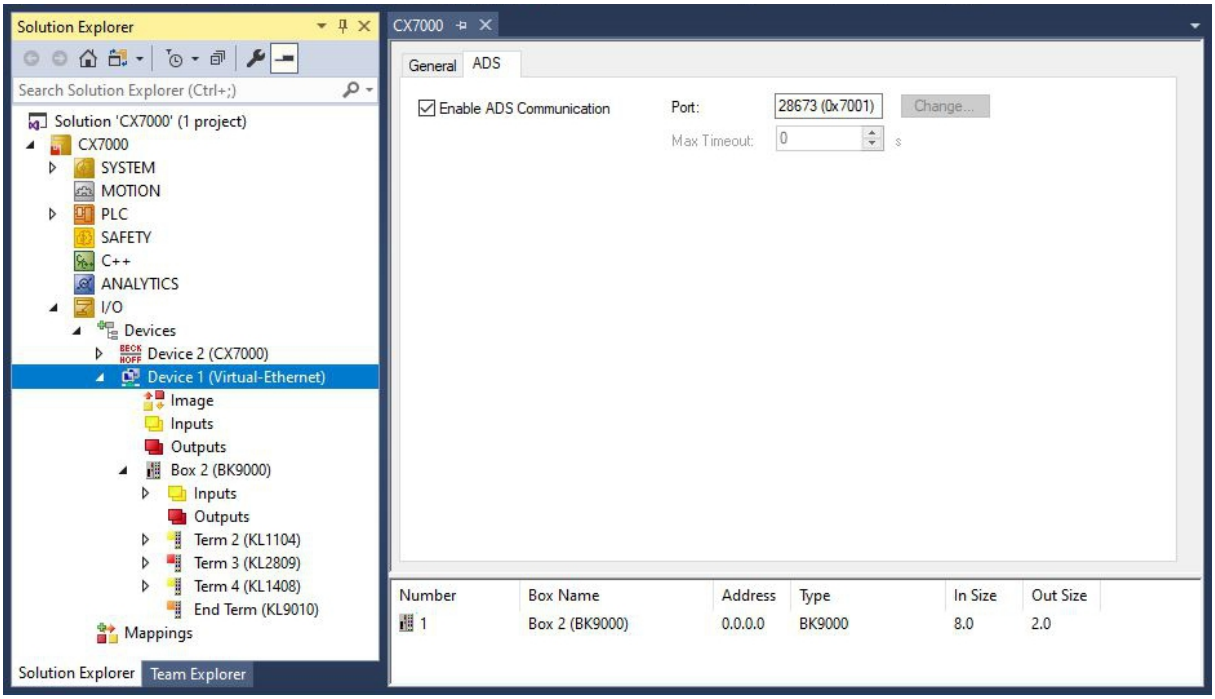
Proceed as follows:

1. In the tree view on the left, right-click on **Devices**.



2. Click on **Add New Item** and select the **Virtual Ethernet Interface**.

⇨ The Virtual Ethernet Interface is created in the tree view on the left. The ADS port number can be read out under the **ADS** tab. The **Enable ADS Communication** option must be active so that ADS communication to the BK9xx0 is possible.



8.4 CoE access to multi-function I/Os

The `FB_EcCoeSdoReadEx` function block allows data to be read from an object directory of an EtherCAT slave via SDO data (Service Data Object). The nSubIndex and nIndex parameters allow the object that is to be read to be selected. Via `bCompleteAccess := TRUE` the parameter can be read with subelements.

Sample: Read the firmware version of the multi-function I/Os.

```
VAR
AMSNetID AT %I*:T_AmsNetIdArr;
Port AT %I*:T_AmsPort;
FB_EcCoESdoReadEx: FB_EcCoESdoReadEx;
FirmwareVersion: STRING;
END_VAR
```

The AmsNetId and port number are required for communication with the CX7028 interface. The inputs of the function block `FB_EcCoeSdoReadEx` can be linked with the input variables `netId` and `port` under TwinCAT, so that the function block is permanently connected to the CX7028 interface.

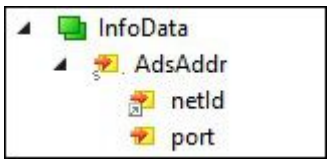


Fig. 28: CoE access to multi-function I/Os, input variables "netId" and "port" under TwinCAT.

The input `sNetId` of the function block corresponds to the input `netId` under TwinCAT. The function block requests a string and the link returns a byte array. You can convert the byte array to a string using the `F_CreateAmsNetId` function. The input `nSlaveAddr` corresponds to the input `port` under TwinCAT.

```
FB_EcCoESdoReadEx(
sNetId:=F_CreateAmsNetId(nIds:=AMSNetID ) , (* AmsNetId of the CX7028 Interface *)
nSlaveAddr:=Port , (* Port Number(nSlaveAddr): 0x1000 *)
nSubIndex:= ,
nIndex:=16#100A , (* Index Number *)
pDstBuf:=ADR(FirmwareVersion) ,
cbBufLen:=SIZEOF(FirmwareVersion) ,
bExecute:=TRUE ,
tTimeout:= ,
bCompleteAccess:= ,
bBusy=> ,
bError=> ,
nErrId=> );
```

The index number for the CoE object **Software version** is located under the CoE Online tab.

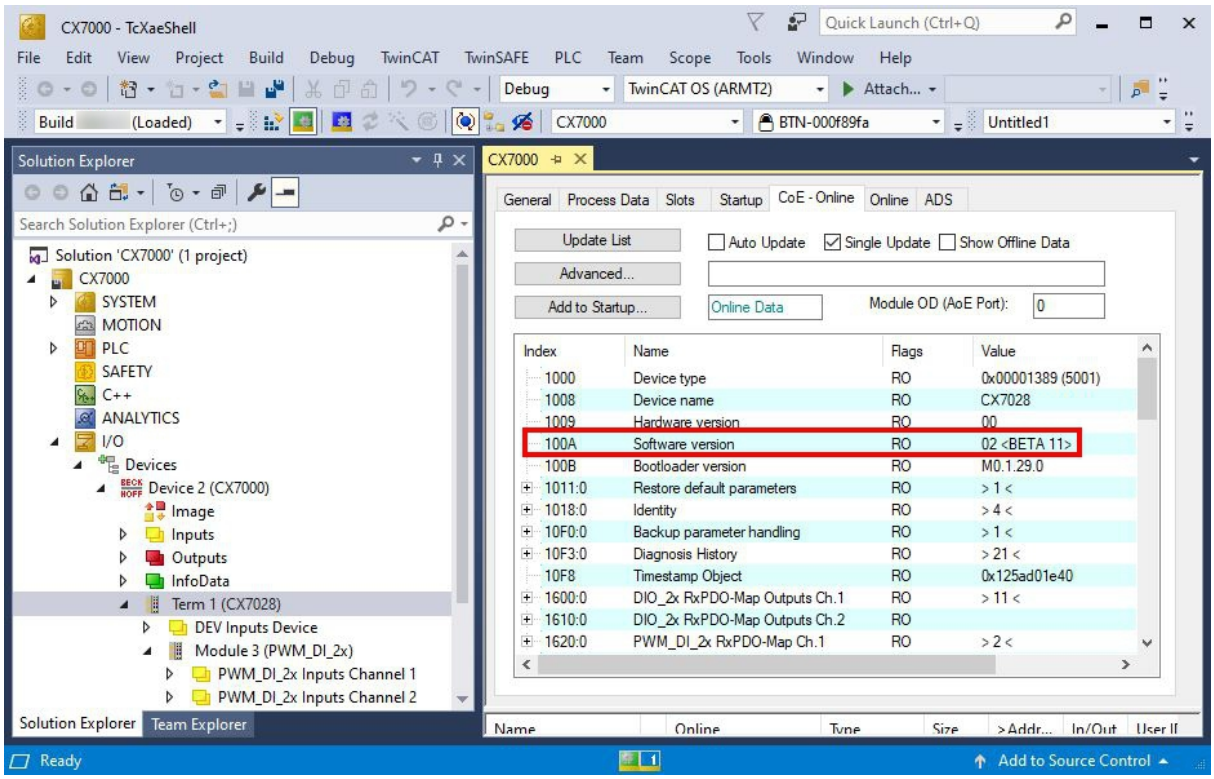


Fig. 29: CoE communication, listing of CoE objects with matching index number.

With the `FB_EcCoeSdoWriteEx` function block an object from the object directory of an EtherCAT slave can be written by SDO-Download. Pay attention to whether the object can be accessed for reading; this is displayed in the Flags column. The nSubIndex and nIndex parameters allow the object that is to be written to be selected. Via `bCompleteAccess := TRUE` the parameter can be written with subelements.

8.5 Power supply terminal

EtherCAT Terminals (E-bus) or Bus Terminals (K-bus) can optionally be connected directly on the right-hand side; the CX7000 automatically recognizes which system is connected during the start-up phase.

K-bus interface

The CX7000 reads out the terminal types during scanning and creates them in the System Manager under a Bus Coupler.

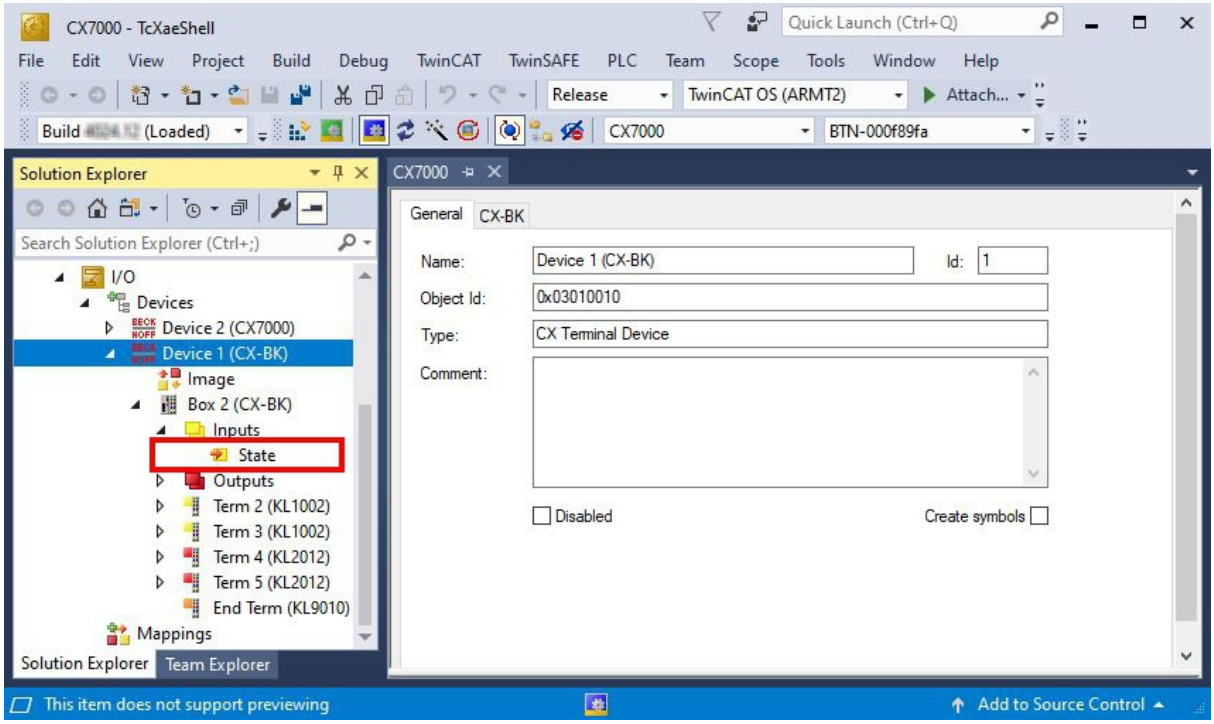


Fig. 30: K-bus interface of a CX7000 in the TwinCAT System Manager.

For K-bus diagnostics there is a status variable in TwinCAT under the Bus Coupler, which can be used for diagnostic purposes and indicates the status of the K-bus communication. For more information, refer to the chapter "Error handling and diagnostics" at K-bus [▶ 100].

E-bus interface

Distributed clocks

The Embedded PCs of the CX7000 series are not suitable for the use of EtherCAT slaves that use distributed clocks or require them.

The operation of EtherCAT Terminals and EtherCAT devices is also possible at CX7000. The CX7000 also recognizes these terminals automatically during scanning, reads out the terminal types and creates them in the System Manager under an EtherCAT Coupler.

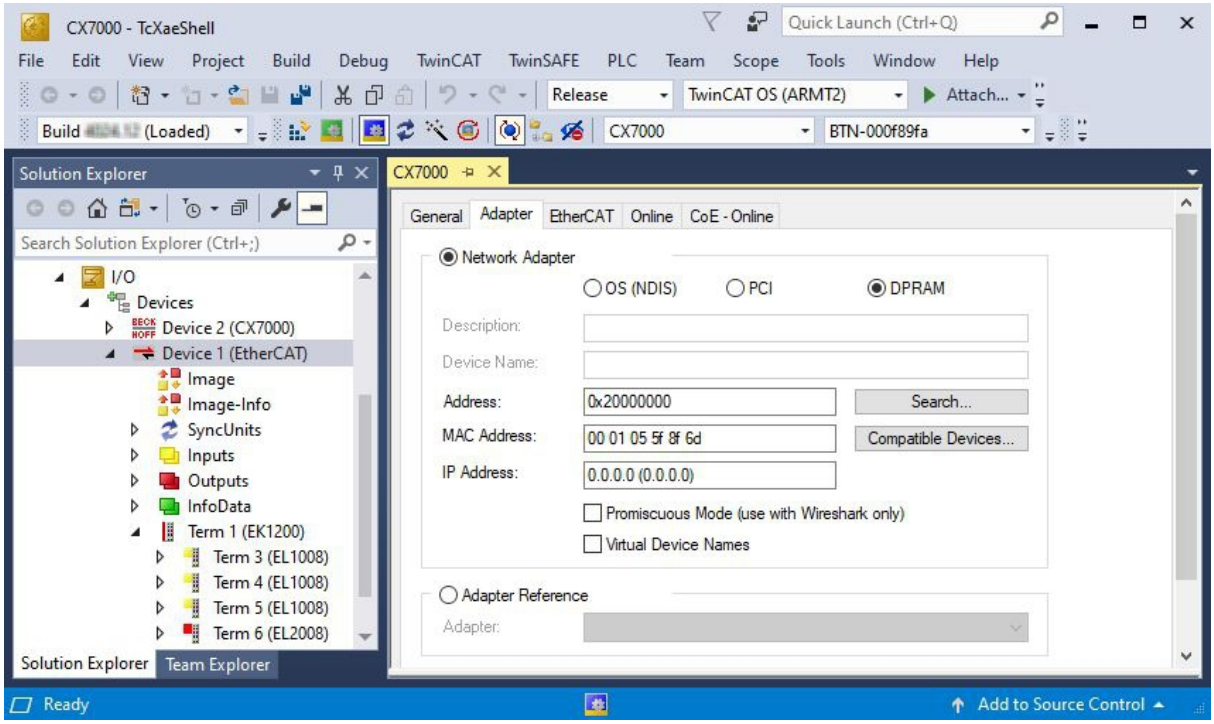


Fig. 31: E-bus interface of a CX7000 in the TwinCAT System Manager.

For more information on diagnostics, refer to the chapter "Error handling and diagnostics" at E-bus [▶ 103].

8.6 Cycle and processing times

8.6.1 Measuring processing time in the PLC program

This sample shows you how to determine the processing time of a program code with the help of a small PLC program. This allows you to measure, for example, how long the PLC needs for a mathematical function, a loop or a specific program part. The resolution is 1 ns per digit.

Sample

```
VAR
    MeasureStart    : T_DCTIME64;
    MeasureResult   : T_DCTIME64;
END_VAR

PROGRAM:
MeasureStart:=F_GetActualDcTime64(); (*Insert your program code to measure the processing time*)
MeasureResult:=F_GetActualDcTime64()-MeasureStart;
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_EtherCAT

8.6.2 Real-Time Clock (RTC)

The CX7000 has an internal, capacitor-buffered real-time clock (RTC) for time and date, which continues to run in the switched-off state. The capacitance of the capacitor is sufficient for at least 30 days and, unlike a battery-backed solution, is maintenance-free. The time is lost and must be reset if the CX7000 is turned off for more than 30 days

The following settings are possible in the boot.conf file:

- SNTP Server
- Update time (default = 1 hour)
- Change UTC Offset
- DHCP server

Sample

The sample below shows you how to read the time. In the sample, the time is output as UTC time and one hour is added to get the CET time.

```
VAR
    FB_LocalSystemTime    : FB_LocalSystemTime;
    DATEANDTIME           : DATE_AND_TIME;
    DATEANDTIME_Add1h     : DATE_AND_TIME;
END_VAR

PROGRAM:
FB_LocalSystemTime(
        sNetID:= ,
        bEnable:=TRUE ,
        dwCycle:= ,
        dwOpt:= ,
        tTimeout:= ,
        bValid=> ,
        systemTime=> ,
        tzID=> );

DATEANDTIME:=SYSTEMTIME_TO_DT(TIMESTR:=FB_LocalSystemTime.systemTime );    (*UTC Time*)
DATEANDTIME_Add1h:=DATEANDTIME+T#1H;    (*UTC Time + 1h*)
```

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (WES7/Win7/Win10: TC RT x86/x64, WEC6/7: TC RT x86, WEC7: TC CE7 ARMV7, TC/BSD: TC RT x64, TC OS ARMT2)	Tc2_Utilities (System)

8.6.3 I/O Idle Task:

The I/O Idle Task executes the asynchronous mappings and is also responsible for acyclic communication with the EtherCAT devices (e.g. writing parameters).

Recommended setting for the I/O Idle Task

By default, the I/O Idle Task is set to 1 ms for TwinCAT controllers. This standard setting has no effect on high-performance control systems and can be ignored. In a small controller such as the CX70xx, however, the default setting of 1 ms leads to a higher basic load. Therefore, proceed as follows:

- first set the I/O Idle Task to 10 ms
- the I/O Idle Task is always placed at the end of the priority list.

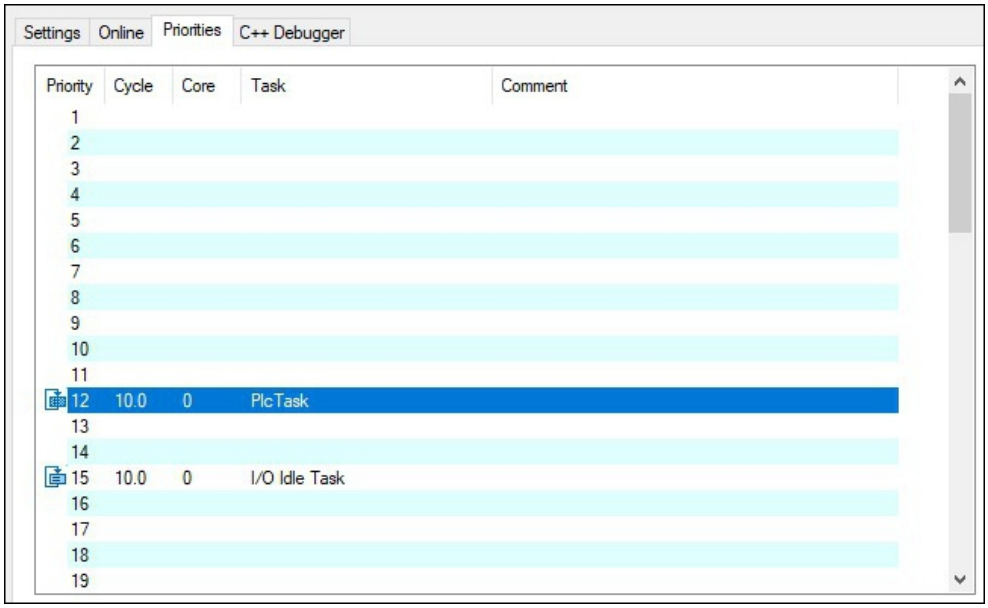


Fig. 32: Setting for the I/O Idle Task in TwinCAT 3 Engineering.

Cycle timeout of the I/O Idle Task

A cycle timeout of the I/O Idle Task can also be neglected unless the I/O Idle Task continuously counts up. In this case, attempts should be made to slow down acyclic communication, query less data, and reduce the number of acyclic communications.

8.6.4 Use NTP provider

An NTP provider is available on the CX70xx from image version 140501 or higher.

The NTP provider can be used on different CX70xx Embedded PCs to provide approximately the same time. There is typically a deviation of ±5 ms between two CX70xxs.

Further information on how to insert the NTP provider into TwinCAT as a TcCOM module and put it into operation can be found here: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_grundlagen/6326712203.html&id=

Example program for using the NTP provider

https://infosys.beckhoff.com/content/1033/CX7000/Resources/16683737611.zip

Variable declaration in the ST editor

```
PROGRAM TimeSync
VAR
(*
Time format:
FileTime64        8 Byte len        begin 1.1.1601    Bit/100ns
DCTime64          8 Byte len        begin 1.1.2000    Bit/100ns
*)
    Test_FileTime64:T_FILETIME64;    (* not used only to see where
                                        this time begin to see the different *)
    Test_DCTime64:T_DCTIME64;        (* not used only to see where
                                        this time begin to see the different *)


    nLocalTime            : T_FILETIME64;       // Filetime 64 - Local RTOS Time
    nPLC_DC_TASKTime      : T_FILETIME64;       (* Filetime 64 - Local DC Time | Local DC Time =
                                                   System Time see DiffSysTimeToPLCDCTime *)
    nSysTime AT %I*        : T_FILETIME64;      // Filetime 64 - Sytem Time
    nExtTime AT %I*        : T_FILETIME64;      // Filetime 64 - TCNet Ext NTP Time
    nSysToExtTimeOffset     AT %I*    :LINT;    // Offset this is the offset from ExtTime to SysTime
    nSysToExtTimeDeviation  AT %I*    :LINT;    (* Only for Info: Diff from NTP to the NPT Filter
                                                   time is it < 1ms then is bIsSynchronized = TRUE *)

    bIsConnected AT %I*:BOOL;            // Connection to the NTP Server
    bIsSynchronized AT %I*:BOOL;         // see nSysToExtTimeDeviation

    FB_LocalSystemTime: FB_LocalSystemTime;        // Functionblock for read the local RTOS Time

    dt_LocalTime: DT;            // Time & data - local RTOS Time
    dt_PLC_DC_TASKTime: DT;      (* Time & data - This is the time from the TaskInfo Array, this
                                    is the same time as the Sytem Time from the NTP Object *)
    dt_nSysTime: DT;             // Time & data - Sytem Time from the NTP Object
    dt_nExtTime: DT;             // Time & data - NTP Object Ext NTP Time

    PlcTaskSystemInfo                : PlcTaskSystemInfo;    // Task Info


    DiffSysTimeToPLCDCTime: LINT;            // Local DC Time = System Time
    DiffLocalTimeToExtTime: LINT;            (* local Time - ExtTime = same time like
                                                the nSysToExtTimeOffset Time *)
    FB_EcDcTimeCtrl64_1:FB_EcDcTimeCtrl64;   // FB to get Sec/min/Hour... from a DCTime64
    nExtTime64: ULINT;                       // Format DCTime64 from nExtTime
    sec_1: WORD;                             // nExtTime Sec
    Min_1: WORD;                             // nExtTime Min
    msec_1: WORD;                            // nExtTime mSec

    nLocalTimeDC64: ULINT;                   // DCtime64 - from local filetime64
    MaxPLCTime: UDINT;


    lrTimeoffset: LREAL;              // Offest Time in LREAL [ms]
    lrTimeDeviation: LREAL;           (* Deviation Time [ms] this time show you the diff to the
                                         NTP filter time, if its <1ms the is bIsSynchronized = TRUE *)

    FB_CX7000_LED_WD: FB_CX7000_LED_WD;    // NPT Status via WD LED from CX7000


END_VAR
```

Program in the ST editor

```
PlcTaskSystemInfo:=_TaskInfo[1];      (* Get Task Info Data we need it to read the
                                         PlcTaskSystemInfo.DcTaskTime *)

dt_PLC_DC_TASKTime:=Systemtime_TO_DT(DCTime64_TO_SYSTEMTIME(LINT_TO_ULINT(PlcTaskSystemInfo.DcTaskTi
me)));          // Get Local DC Time to DT
nPLC_DC_TASKTime:=DCTIME64_TO_FILETIME64(LINT_TO_ULINT(PlcTaskSystemInfo.DcTaskTime));
               // Get Local DC Time to FileTime64

MaxPLCTime:=MAX(PlcTaskSystemInfo.LastExecTime,MaxPLCTime);


lrTimeoffset:=LINT_TO_LREAL(nSysToExtTimeOffset)/1000/10; // Read NTP Offset Time and convert it to
Real in [ms]
```

```
lrTimeDeviation:=LINT_TO_LREAL(nSysToExtTimeDeviation)/1000/10; // Read NTP Offset Time and convert
it to Real in [ms]


// Read the local time from RTOS
FB_LocalSystemTime(
    sNetID:= ,
    bEnable:=TRUE ,
    dwCycle:= ,
    dwOpt:= ,
    tTimeout:= ,
    bValid=> ,
    systemTime=> ,
    tzID=> );

dt_LocalTime:=SystemTime_TO_DT(FB_LocalSystemTime.systemTime);
nLocalTime:=SystemTime_TO_Filetime64(FB_LocalSystemTime.systemTime);

dt_nSysTime:=Filetime64_TO_DT(nSysTime);    // Convert nSysTime to date & time
dt_nExtTime:=Filetime64_TO_DT(nExtTime);    // Convert nExtTime to date & time

DiffSysTimeToPLCDCTime:=ULINT_TO_LINT(nPLC_DC_TASKTime-nSysTime);        // Must be 0 because
SysTime = Local DC Time
DiffLocalTimeToExtTime:=ULINT_TO_LINT(nExtTime-nPLC_DC_TASKTime);        // Same diff like
nSysToExtTimeOffset

// Ext NTP object as trigger for Outputs
nExtTime64:=FILETIME64_TO_DCTIME64(nExtTime);
FB_EcDcTimeCtrl64_1.A_GetMilli (in:=nExtTime64,get=>msec_1);
FB_EcDcTimeCtrl64_1.A_GetSecond (in:=nExtTime64,get=>sec_1);
FB_EcDcTimeCtrl64_1.A_GetMinute (in:=nExtTime64,get=>Min_1);

// Toggle Outpt every Sec
IF (sec_1 MOD 2) =0 THEN
    gvl.bCX7028_Out_4:=TRUE;
ELSE
    gvl.bCX7028_Out_4:=FALSE;
END_IF


// Set DO NTP Status
gvl.bCX7028_Out_1:=bIsConnected;
gvl.bCX7028_Out_2:=bIsSynchronized;

// WD LED from CX7000
// Red - no connection
// Green flashing - it is connected to NTP Server but it is not sync (that means that the filter is
active and the synchonsation to the NTP Server is running - not with TC!)
// Green on - connection and synchonsation is running
FB_CX7000_LED_WD(
    bEnable:=TRUE ,
    eLED:= ,
    tFlashingTimeP1:= ,
    tFlashingTimeP2:= ,
    bError=> ,
    nErrorID=> );
IF bIsConnected AND NOT bIsSynchronized THEN
    FB_CX7000_LED_WD.eLED :=Tc2_SystemCX.E_CX7000_LED.LED_flashing_GREEN_OFF;
ELSIF bIsConnected AND bIsSynchronized THEN
    FB_CX7000_LED_WD.eLED :=Tc2_SystemCX.E_CX7000_LED.LED_GREEN ;
ELSE
    FB_CX7000_LED_WD.eLED :=Tc2_SystemCX.E_CX7000_LED.LED_RED ;
END_IF
```

8.6.5 Cycle time of 250 µs

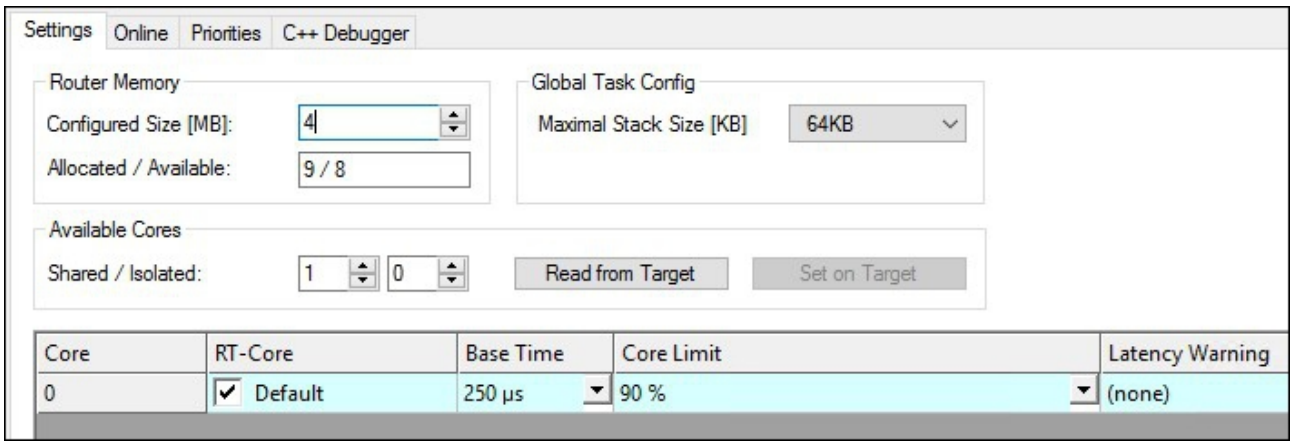
Note that a cycle time of 250 µs on a CX7000 represents an extreme optimum and all boundary conditions must be right. Furthermore, a cycle time of 250 µs only makes sense if the inputs and outputs are correspondingly fast.

The CX7000 has different interfaces, including, for example, the K-bus. The K-bus can achieve perhaps 1 ms under optimal conditions and is therefore unsuitable for cycle times of 250 µs. The E-bus (EtherCAT) is much faster, but the structure of an EtherCAT frame and the merging of the data into an EtherCAT frame is much more complex, so that only 1 ms is possible here as well.

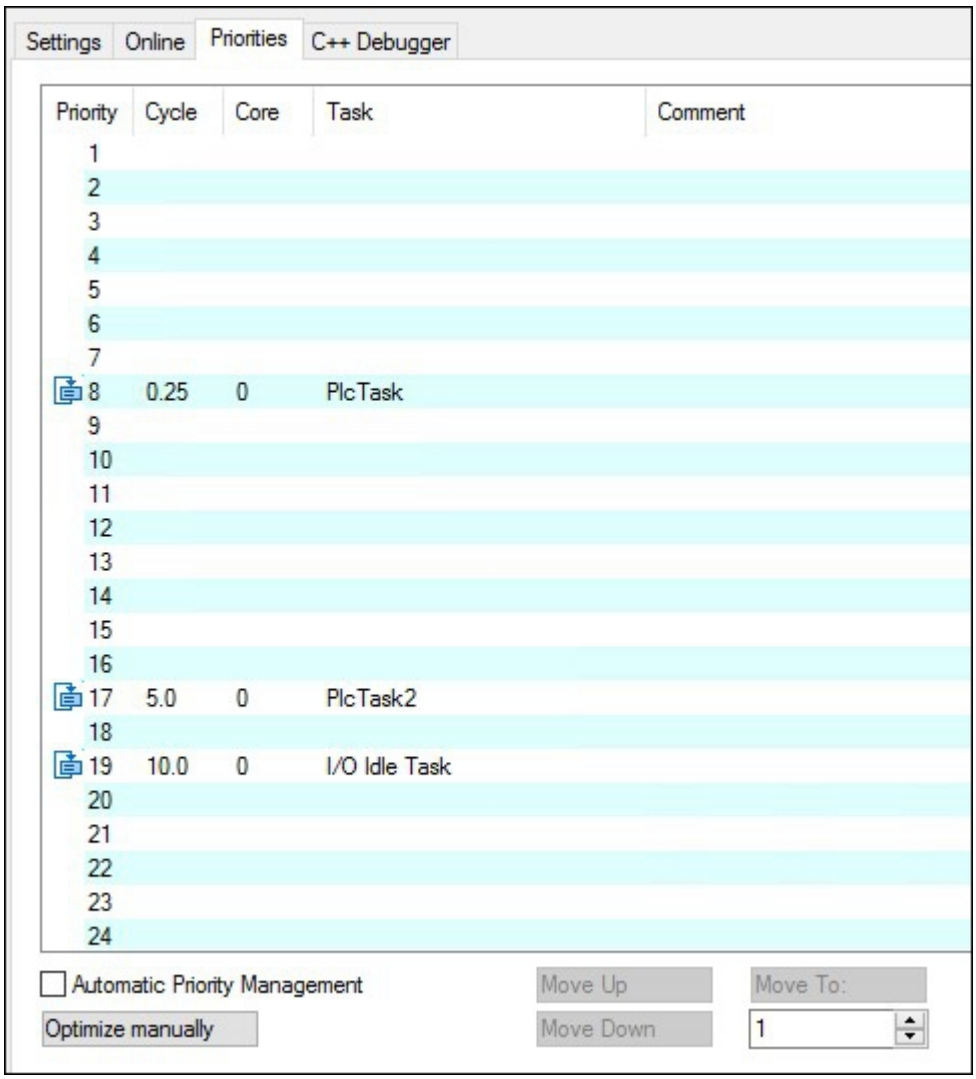
Of course, EtherCAT can be operated with other Industrial PCs under 100 µs. However, these are usually equipped with more powerful CPUs and may use a DMA controller for EtherCAT processing. That is not the case with the CX7000, however, so the CPU power and the interfaces to EtherCAT are the limiting factors. Of course, the CX7000 as a small controller was not developed for high-speed applications and, due to its cost-efficiency, should not be compared to more powerful Industrial PCs.

Setting a cycle time of 250 µs

A cycle time of 250 µs is possible on a CX7000 if the boundary conditions are right. The CX7000 is helped by the multifunction I/Os, which are connected to the CPU via a fast IO connection. The connection is kept very lean and has a correspondingly good data throughput. It is possible to reach the 250 µs with the help of the multifunction I/Os. Of course, the PLC program may contain only very little code and the core limit must be set to 90%, which in turn results in the described disadvantages (see: Real-time and CPU load [▶ 107]).



In addition, you should set the priority of the task so that the 250 µs task has the highest priority in the system.



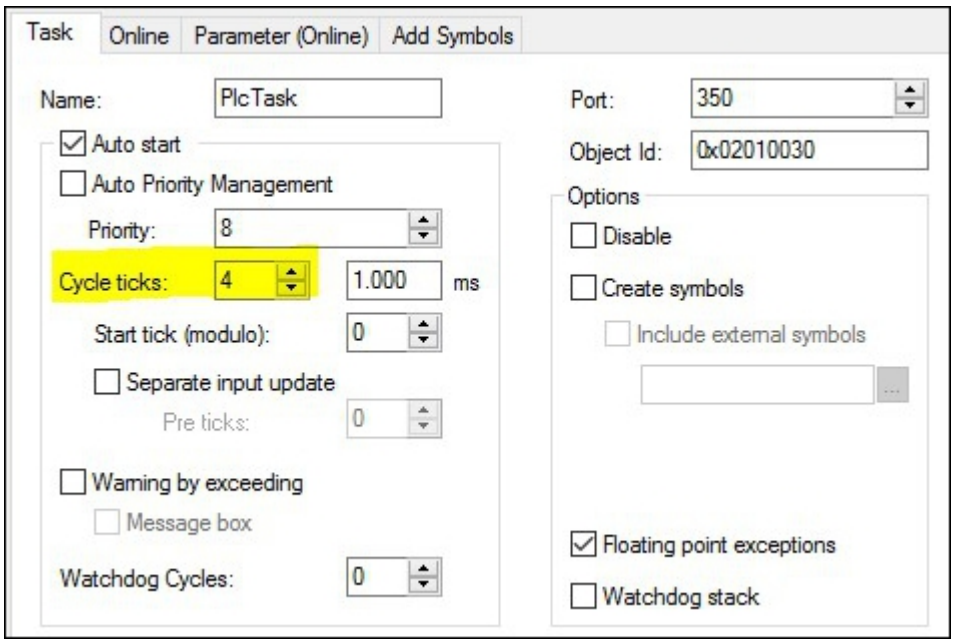
If you now allow a digital output of the CX7028 interface to toggle, for example with `Out_01:=not Out_01` in the 250 µs task, the task is output at a frequency of 2 kHz. In order for the output to be optimally fast, this output should have a load. Only wire the output with a digital input; as a result, the load is very small and the switch-off behavior of the driver is relatively slow. Slow here means in relation to the 250 µs task time. It makes a difference whether the output requires 50 µs or 100 µs to switch off. If you now wish to measure the response time, i.e. the time it takes for the CX7000 to react to an input, the following background is important:

From a cycle time of 1 ms or greater, an optimal cycle is operated, i.e. the inputs of the CX7028 interface are read by the processor of the CX7028 interface about 20% before the new task cycle. If the task time is faster than 1 ms, the time is not sufficient for the optimized response time. In this case the inputs are read with the task cycle. As a result, a task time of 500 µs achieves the same response time as a task time of 1 ms. With a task time of less than 1 ms, the update needs four task cycles for a cycle. With 1 ms or slower it needs two task cycles. This should make you aware that it is not always the shortening of the cycle time that shortens the reaction time, but also the internal process, which plays a decisive role in the reading of the data.

Here is a sample, so that you can reproduce this behavior yourself and see and measure the differences:

1. Connect the +24 V Up and 0 V Up power supply to power the multifunction I/Os.
2. Connect output 1 to input 1 to toggle the output as described.
3. Connect output 2 to input 2.
4. Set the core limit to 90%, the base time to 250 µs, the priority of the fast task to the highest priority, and the idle task to 10 ms.

The inputs have only a minimal filter time and are therefore well suited for the measurement. A load on the output is not necessary in this case. For the following samples, we always leave the base time at 250 µs and only increase the number of cycle ticks in order to set the corresponding task time.



Sample program

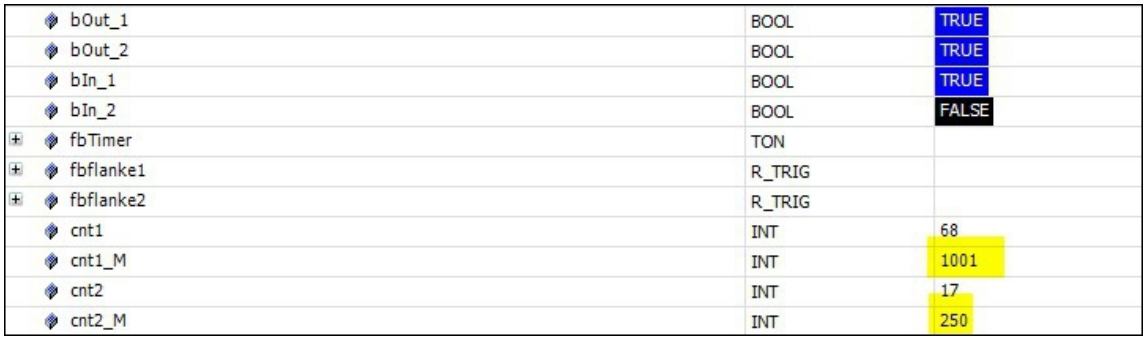
```
PROGRAM MAIN
VAR
    bOut_1 AT %Q*:BOOL; (*toggle Output link to digital Output pin 7*)
    bOut_2 AT %Q*:BOOL; (*reaction time link to digital Output pin 14*)

    bIn_1 AT %I*: BOOL; (*toggle Output link to digital Input pin 2*)
    bIn_2 AT %I*: BOOL; (*reaction time link to digital Input pin 10*)

    fbTimer : TON;
    fbflanke1 : R_TRIG;
    fbflanke2 : R_TRIG;

    cnt1: INT; (*toggle Output*)
    cnt1_M: INT; (*toggle Output*)

    cnt2: INT; (*reaction time*)
    cnt2_M: INT; (*reaction time*)
END_VAR


PROGRAM MAIN
bOut_1:= NOT bOut_1; (*toggle Output*)
bOut_2:= NOT bIn_2; (*reaction time*)

fbflanke1(CLK:=bIn_1);
IF fbflanke1.Q THEN
    cnt1:=cnt1+1; (*toggle Output*)
END_IF

fbflanke2(CLK:=bIn_2);
IF fbflanke2.Q THEN
    cnt2:=cnt2+1; (*reaction time*)
END_IF

fbTimer(PT:=T#1S,in:=NOT fbTimer.Q);

IF fbTimer.Q THEN
    cnt2_M:=cnt2; (*reaction time*)
    cnt1_M:=cnt1; (*toggle Output*)
    cnt1:=0;
    cnt2:=0;
END_IF
```

The toggling of the output results in a frequency of 2 kHz – 250 µs On, 250 µs Off – i.e. a period duration of 500 µs. When measuring the positive edge, this is 2000 edge changes in one second.

bOut_1	BOOL	TRUE
bOut_2	BOOL	TRUE
bIn_1	BOOL	FALSE
bIn_2	BOOL	FALSE
fbTimer	TON	
fbflanke1	R_TRIG	
fbflanke2	R_TRIG	
cnt1	INT	1014
cnt1_M	INT	2000
cnt2	INT	253
cnt2_M	INT	500

Fig. 33: Measurement at a task time of 250 µs.

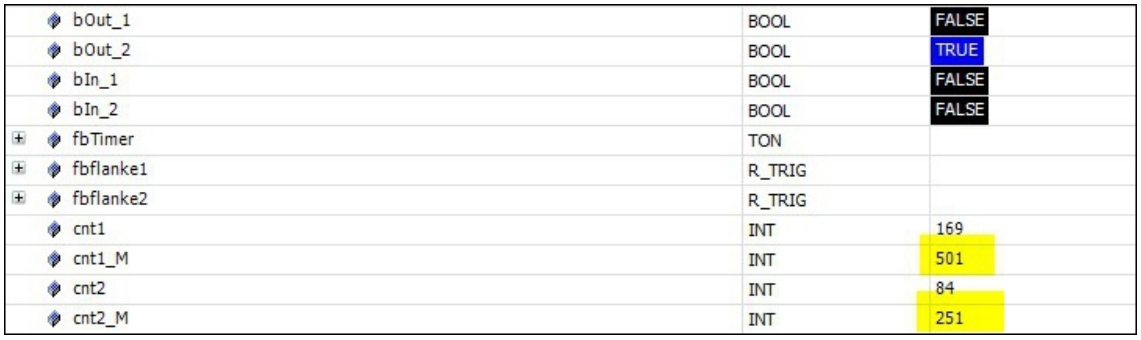
In the case of the response time, it is 500 changes in one second, as the optimized access to the inputs does not apply here.



bOut_1	BOOL	TRUE
bOut_2	BOOL	TRUE
bIn_1	BOOL	TRUE
bIn_2	BOOL	FALSE
fbTimer	TON	
fbflanke1	R_TRIG	
fbflanke2	R_TRIG	
cnt1	INT	68
cnt1_M	INT	1001
cnt2	INT	17
cnt2_M	INT	250

Fig. 34: Measurement at a task time of 500 µs.

As expected, the values are only half as large with a task time that is twice as long.



bOut_1	BOOL	FALSE
bOut_2	BOOL	TRUE
bIn_1	BOOL	FALSE
bIn_2	BOOL	FALSE
fbTimer	TON	
fbflanke1	R_TRIG	
fbflanke2	R_TRIG	
cnt1	INT	169
cnt1_M	INT	501
cnt2	INT	84
cnt2_M	INT	251

Fig. 35: Measurement at a task time of 1 ms.

With a task time of 1 ms, you can clearly see that the optimized mode actually helps to reduce the response time. While the toggle change has halved again, i.e. it is now still 500 Hz with a task time of 1 ms, the value for the response time has remained the same.

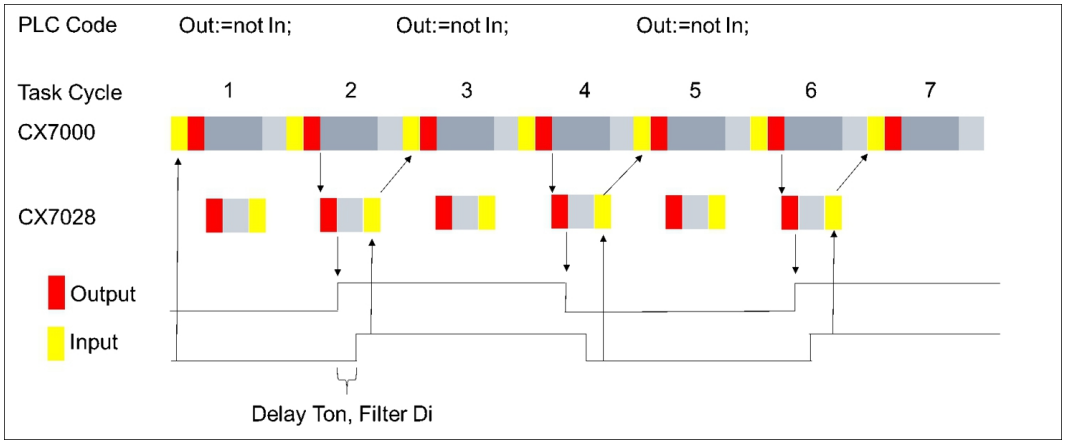
8.6.5.1 Cycle time ≥1 ms

Fig. 36: CX7000 CPU and PLC.

Yellow and red: Mapping and update of the IOs.
Light grey: Time remaining until the task begins again (OS).
Dark grey: PLC cycle.

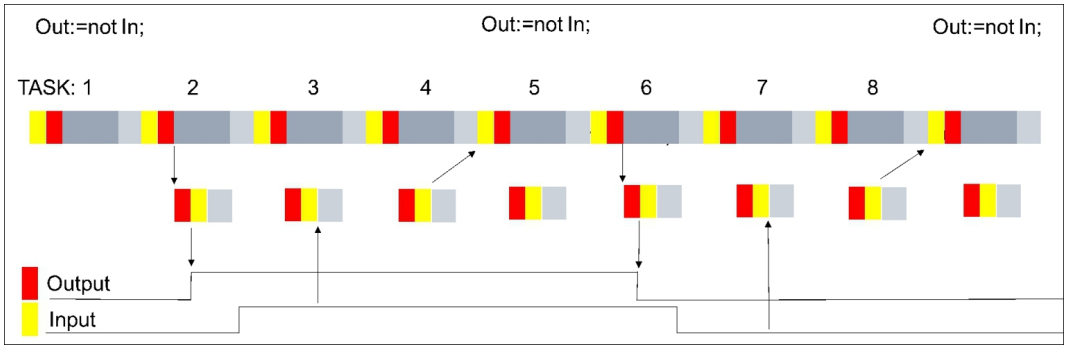
Fig. 37: CPU of the CX7028 interface.

Red: Output update.
Grey: CPU processing of the multifunction IOs.
Yellow: Input update (from a cycle time of 1 ms there is a waiting period of up to approx. 80% of the cycle time before the update of the input signals so that the inputs are read as late as possible, i.e. before the next cycle).



8.6.5.2 Cycle time < 1 ms

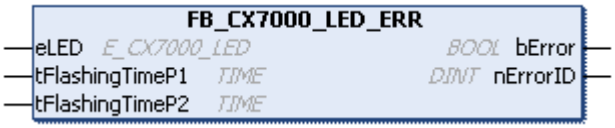
From a cycle time of < 1 ms, the update of the input signals is carried out immediately and is therefore only available with the next cycle. The input signals are therefore always one cycle old.



With this background knowledge, you should be able to make the right settings on the CX7000 for your application.

8.7 Function Blocks

8.7.1 FB_CX7000_LED_ERR



The function block allows the use of the ERR LED on the CX7000. The function block is immediately active when it is called and controls the ERR LED via the mode.

The ERR LED of the CX7000 can be used to make the states of the PLC program, communication or other indications externally visible.

The ERR LED has two colors, red and green. If both colors are switched on, the LED lights up yellow. You can either turn the LED on or make it flash.

User-specific function of the LEDs

Due to the user-specific usability of the LEDs, Beckhoff Support cannot know the meaning of a flashing code and cannot support the customer.

- Document the function of the LEDs for your customers.

Inputs

```
VAR_INPUT
    bEnable             : BOOL;             // set TRUE to enable LED handling; Reset in order to re
set error
    eLED                : E_CX7000_LED;     // LED flashing mode
    tFlashingTimeP1     : TIME:=T#250MS;    // Flashing Time >=200ms first pulse
    tFlashingTimeP2     : TIME:=T#250MS;    // Flashing Time >=200ms second pulse
END_VAR
```

Name	TYPE	Description
bEnable	BOOL	The function block controls the LED as soon as and as long as the input is TRUE.
eLED	E_CX7000_LED	LED mode
tFlashingTimeP1	TIME	Time for the first pulse (>= 200 ms)
tFlashingTimeP2	TIME	Time for the second pulse (>= 200 ms)

Outputs

```
VAR_OUTPUT
    bError          : BOOL;         // error flag
    nErrorID        : UDINT;        (* ADS Error ID. If nErrorID=DEVICE_SRVNOTSUPP probably the image
 version need to be updated to support this feature. *)
END_VAR
```

Name	Type	Description
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS Error Code Example: DEVICE_SRVNOTSUPP: the image version of the CX7000 does not support this feature. An update (>=35695) is necessary.

Sample:

```
VAR
    BK9000_BoxState AT %I* : WORD;
    fbErrorLED : FB_CX7000_LED_ERR;
END_VAR
```

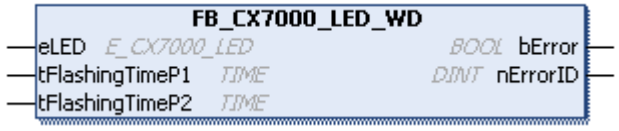
```
IF BK9000_BoxState=0 THEN
fbErrorLED.eLED :=E_CX7000_LED.LED_flashing_GREEN_OFF;
ELSE
fbErrorLED.eLED :=E_CX7000_LED.LED_flashing_RED_OFF;
END_IF
fbErrorLED(
    bEnable := TRUE,
    tFlashingTimeP1 := ,
    tFlashingTimeP2 := ,
    bError => ,
    nErrorID => );
```

NOTICE

Function block can only be used for CX7000

The function block can and must only be used for the CX7000.

8.7.2 FB_CX7000_LED_WD



The function block allows the use of the WD LED on the CX7000. The function block is immediately active with the call and controls the WD LED via the mode.

You can use the WD LED of the CX7000 to make the states of the PLC program, communication or other indications externally visible. The WD LED has two colors, red and green. If both colors are switched on, the LED lights up yellow. You can turn on the LED and/or make it flash.

User-specific function of the LEDs

Due to the user-specific usability of the LEDs, Beckhoff Support cannot know the meaning of a flashing code and cannot support the customer.

- Document the function of the LEDs for your customers.

Inputs

```
VAR_INPUT
    bEnable             : BOOL;             // set TRUE to enable LED handling; Reset in order to re
set error
    eLED                : E_CX7000_LED;     // LED flashing mode
    tFlashingTimeP1     : TIME:=T#250MS;    // Flashing Time >=200ms first pulse
    tFlashingTimeP2     : TIME:=T#250MS;    // Flashing Time >=200ms second pulse
END_VAR
```

Name	Type	Description
bEnable	BOOL	The function block controls the LED as soon as and as long as the input is TRUE.
eLED	E_CX7000_LED	LED mode
tFlashingTimeP1	TIME	Time for the first pulse (>= 200 ms)
tFlashingTimeP2	TIME	Time for the second pulse (>= 200 ms)

Outputs

```
VAR_OUTPUT
    bError          : BOOL;       // error flag
    nErrorID        : UDINT;      (* ADS Error ID. If nErrorID=DEVICE_SRVNOTSUPP probably the image v
ersion need to be updated to support this feature. *)
END_VAR
```

Name	Type	Description
bError	BOOL	The function block has an error.

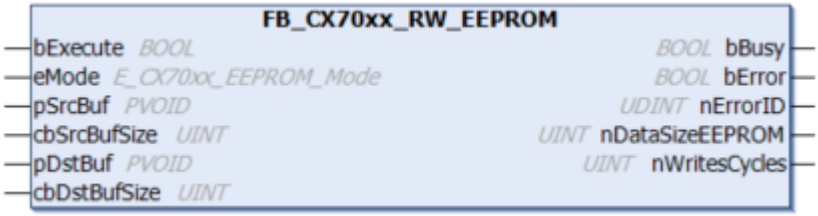
Name	Type	Description
nErrorID	UDINT	ADS Error Code Example: DEVICE_SRVNOTSUPP: the image version of the CX7000 does not support this feature. An update (>=35695) is necessary.

NOTICE

Function block can only be used for CX7000

The function block can and must only be used for the CX7000.

8.7.3 FB_CX70xx_RW_EEPROM



The function block allows a maximum of 120 bytes to be written to the EEPROM (hardware) of the CX70xx. The EEPROM may be written to a maximum of 200 times. The memory is intended for one-time writing.

This function block can be used to personalize the CX70xx. That means, in the simplest case you write your company ID into the EEPROM. When starting the CX70xx program, read the contents of the memory. For example, if it is empty, you cannot continue to run the program because it is no longer your original CX70xx that you programmed.

If you want to exchange a CX70xx for a new device, the EEPROM must be written again by you.

Inputs

```
VAR_INPUT
    bExecute       : BOOL;         // rising edge triggers process with selected mode
    eMode          : E_CX70xx_EEPROM_Mode;    // select RW mode
    pSrcBuf        : PVOID;        // pointer to WRITE EEPROM data buffer
    cbSrcBufSize   : UINT;         // size of WRITE EEPROM data buffer (max.120 Bytes)
    pDstBuf        : PVOID;        // pointer to READ EEPROM data buffer
    cbDstBufSize   : UINT;         // max.size of READ EEPROM data buffer (max.120 Bytes)
END_VAR
```

Name	Type	Description
bExecute	BOOL	A positive edge starts the function block.
eMode	E_CX70xx_EEPROM_Mode	ReadOnly: EEPROM read WriteOnly: EEPROM write WriteAndRead: EEPROM write and read
pSrcBuf	PVOID	Pointer to the data buffer to be written.
cbSrcBufLen	UINT	Length of data to be written (max. 120 bytes)
pDstBuf	PVOID	Pointer to the data buffer into which the contents of the EEPROM are to be copied.
cbDstBufLen	UINT	Length of data to be read. (maximum 120 bytes) When reading, the length information must be greater than or equal to the data contained in the EEPROM.

Outputs

```
VAR_OUTPUT
    bBusy          : BOOL;         // FB is working
    bError         : BOOL;         // FB has an Error
    nErrorID       : UDINT;        (* Error Code
```

```
    If nErrorID=DEVICE_INVALIDACCESS the EEPROM write cycles reached max. value.
    If nErrorID=DEVICE_INVALIDPARM the given pointer parameter is invalid/null.
    If nErrorID=DEVICE_INVALIDSIZE the given buffer size is too small or too big.
    If nErrorID=DEVICE_SRVNOTSUPP probably the image version need to be updated to support this feat
ure. *)
    nDataSizeEEPROM  : UINT;         // current size of (read) EEPROM data in bytes (max.120 Bytes)
    nWritesCycles    : UINT;         // already performed EEPROM write cycles (maximum possible = 20
0)
END_VAR
```

Name	Type	Description
bBusy	BOOL	The function block is active and working.
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS Error Code Examples: DEVICE_INVALIDACCESS: the EEPROM write cycles have reached the maximum value. The EEPROM cannot be rewritten. DEVICE_INVALIDPARM: the allocated pointers are invalid/NULL. DEVICE_INVALIDSIZE: the allocated buffer size is too small or too large. DEVICE_SRVNOTSUPP: the image version of the CX70xx does not support this feature. An update (>=35695) is necessary.
nDataSizeEEPROM	UINT	Current size in bytes of the read EEPROM data
nWritesCycles	UINT	Number of write operations still available

8.7.4 FB_CX70xx_ResetOnBoardIO

The function block allows to execute a reset from the OnBoard I/O of the CX70xx Embedded PC.

Typical use case is after an error in the communication to the OnBoard I/Os (CX7028). Such an error occurs when the power supply (Up) of the OnBoard I/Os is interrupted.

NOTICE

State of the I/Os

Outputs that are still set in the process image are switched on again immediately after a reset.

Further details on the OnBoard I/O can be found in the documentation of the CX70xx Embedded PC.

Inputs

```
VAR_INPUT
    bExecute      : BOOL;         // rising edge triggers process
    sNetId        : T_AmsNetID;   // AMS Net ID of the OnBoard IOs
    tTimeout      : TIME := DEFAULT_ADS_TIMEOUT; // maximum time allowed for execution of this ADS c
ommand
END_VAR
```

Name	Type	Description
bExecute	BOOL	A positive edge starts the function block.
sNetId	T_AmsNetID	AMS Net ID of the OnBoard I/Os
tTimeout	TIME	States the length of the timeout that may not be exceeded by execution of the ADS command.

Outputs

```
VAR_OUTPUT
    bBusy       : BOOL;       // FB is working
    bError      : BOOL;       // FB has an Error
    nErrorID    : UDINT;      (* Error Code. If nErrorID=DEVICE_SRVNOTSUPP probably the image versio
n need to be updated to support this feature. *)
END_VAR
```

Name	Type	Description
bBusy	BOOL	The function block is active and working.
bError	BOOL	The function block has an error.
nErrorID	UDINT	ADS Error Code Examples: DEVICE_SRVNOTSUPP: the image version of the CX70xx does not support this feature. An update (>=47912) is necessary.

Sample:

```
FUNCTION_BLOCK FB_Test_ResetOnboardIO
VAR
    AMSNetID    : T_AmsNetIdArr;    // link to the AMS Net ID of the OnBoard IOs
    State       : WORD;             // link to the State of the OnBoard IOs
    bReset      : BOOL;             // if Ready to Reset you can reset the OnBoard IOs
    fbReset     : FB_CX70xx_ResetOnBoardIO;
END_VAR
```

```
IF State<>8 AND NOT State.8 AND State.4 THEN // if OnBoard IO device signals an error and is not OP
but present
    bReset := TRUE;
ELSE
    bReset := FALSE;
END_IF

IF NOT fbReset.bBusy AND bReset THEN
    fbReset(bExecute:=TRUE, sNetId:=F_CreateAmsNetId(AMSNetID));
ELSE
    fbReset(bExecute:=FALSE);
END_IF
```

8.8 Important attribute pragmas

Attribute pragmas are used to influence compilation and pre-compilation. TwinCAT supports a number of predefined attribute pragmas. Attributes are defined in the declaration part.

8.8.1 Attribute 'Tc2GvlVarNames'

The pragma has the effect that symbols, which are declared in a GVL, are addressed via ADS just like in TwinCAT 2 (without the use of the GVL name as namespace).

Syntax: {attribute ‘Tc2GvlVarNames‘}

Sample:

```
{attribute 'Tc2GvlVarNames'}
VAR_GLOBAL
    Test : INT;
END_VAR

GVL.Test:=GVL.Test+1;     (*without attribute*)
Test:=Test+1;             (*with attribute*)
```

8.8.2 Attribute 'pack_mode'

This attribute pragma specifies how a data structure is packaged during allocation. The attribute must be inserted above the data structure and affects the packing of the whole structure.

Syntax: {attribute 'pack_mode' := '<Value>'}

Sample

```
{attribute 'pack_mode' := '0'}
TYPE str_Test :
STRUCT
      byTest1    : BYTE;
      iTest      : DINT;
      byTest2    : BYTE;
      nValue     : INT;
END_STRUCT
END_TYPE
```

In this sample, the pack mode has been set to 0. If you determine the size of the structure in the sample with SIZEOF, you get the value 8.

1 byte + 4 bytes (DINT) + 1 byte + 2 bytes (INT) = 8 bytes

If you set the pack mode to 2 (WordAlignment), you get the value 10 because a padding byte is inserted after each byte. If you set the pack mode to 4 (DWordAlignment), then you get the value 12, because this time three padding bytes are inserted after each byte. A pack mode of 8 (LWordAlignment) does not change anything, because the sample does not use variables that require 8 bytes.

The CX7000 works with the DWordAlignment (pack mode 4) if you do not use the attribute.

For more information about the pack_mode attribute, see: Attribute 'pack_mode'

8.8.3 Attribute 'TcCallAfterOutputUpdate'

The attribute pragma `TcCallAfterOutputUpdate` causes the IO update to take place before the PLC cycle and not after the PLC program as is set by default.

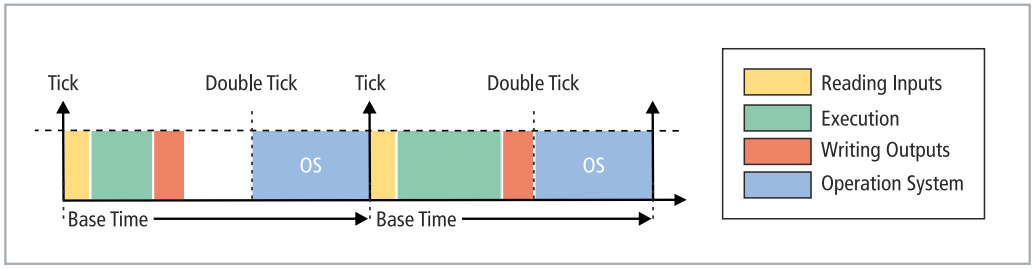


Fig. 38: Default calling of a PLC task.

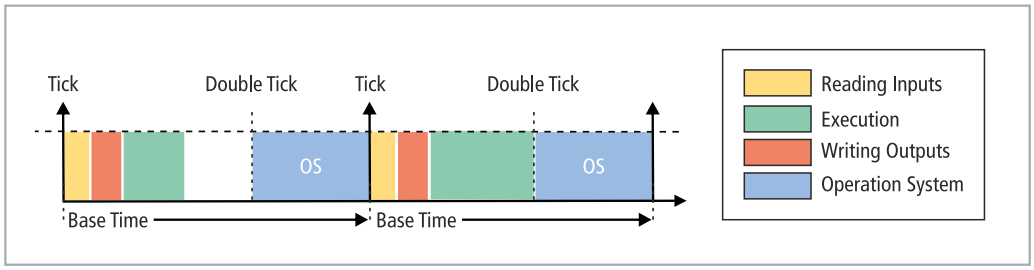


Fig. 39: Calling a PLC task with the attribute tcCallAfterOutputUpdate.

This function can be used for projects with strongly fluctuating cycle times. In projects with strongly fluctuating cycle times, the outputs, since they are written after the PLC cycle, are sometimes written earlier (short PLC cycle time) and sometimes later (long PLC cycle time). These fluctuations cause jitter in the outputs. The disadvantage is that the attribute cannot react quite as quickly and a cycle is always lost. You have to decide whether you want to react quickly to an input (default setting) or whether you prefer to have a deterministic behavior of the outputs (setting of the attribute).

Syntax: {attribute 'TcCallAfterOutputUpdate'}

Insertion location: This attribute must be added to all program POUs, which are to be called after the output update.

Sample:

To illustrate the behavior, you need a digital output terminal such as an EL2008 and an oscilloscope.

Write a small PLC program and link the variable `bOut` with a digital output:

```
bOut:=not bOut;
```

The PLC program is very simple and does not cause any fluctuations. The pulse is displayed on the oscilloscope as follows:

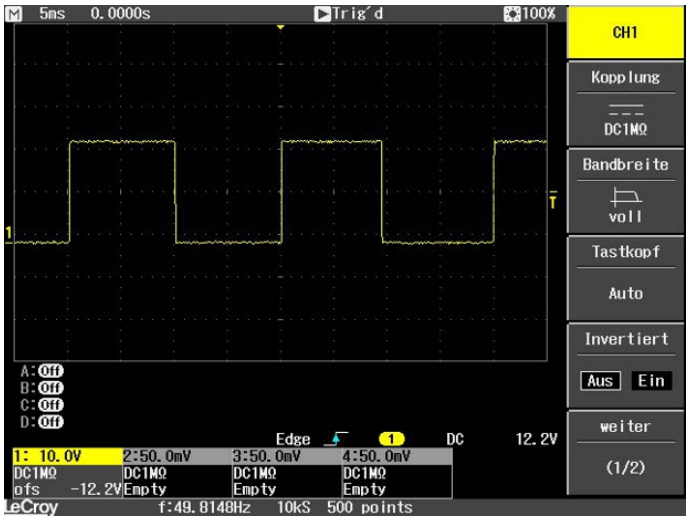


Fig. 40: Pulse of a digital output without load.

Now extend the PLC program with a For loop to create a program load. The mathematical function used does not matter and is intended only to generate a load:

```
bOut:=not bOut;

IF bOut THEN
    For loop:=1 to 2000 do
        lrTest:=SIN(INT_TO_LREAL(loop)*3.14);
    END_FOR
END_IF
```

Whenever the output is set to TRUE, the loop is run through and a load is generated. As a result, more time is needed to run the PLC and the output is written later than usual. During the next cycle, the output is set back to FALSE, the loop is not run through and the output is set to FALSE faster, because the PLC program is finished faster without a For loop. The result is that the pulse is very much shorter.

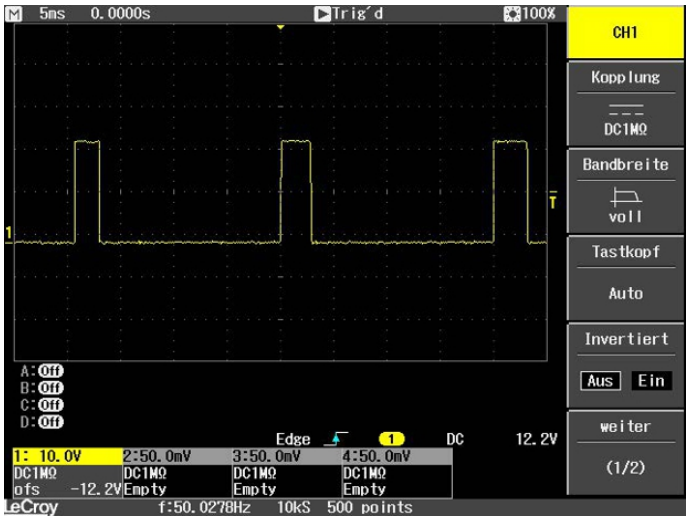


Fig. 41: Shortened pulse of a digital output with load.

If the For loop is called upon FALSE instead of TRUE, the result is inverted.

```
bOut:=not bOut;

IF not bOut THEN
    For loop:=1 to 2000 do
        lrTest:=SIN(INT_TO_LREAL(loop)*3.14);
    END_FOR
END_IF
```

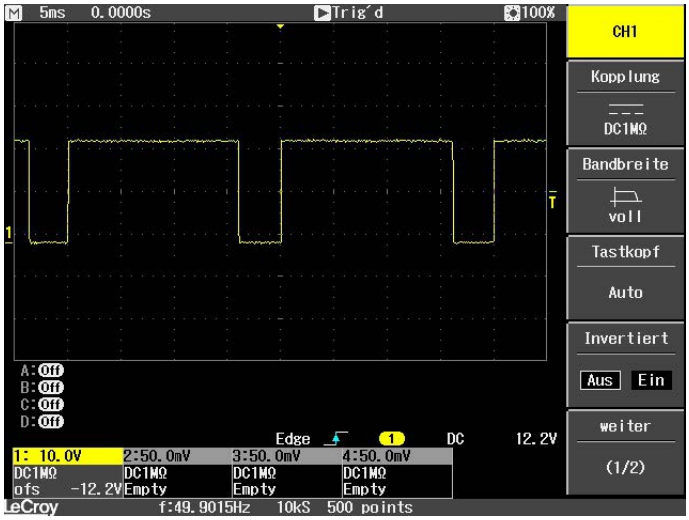


Fig. 42: Inverted representation of a digital output.

With the attribute pragma `TcCallAfterOutputUpdate`, the pulse is constant and is independent of how long the For loop takes or whether it is called. The whole thing only works if the PLC task is not exceeded. Therefore, when reproducing the sample, pay attention to the exceed counters of the task.

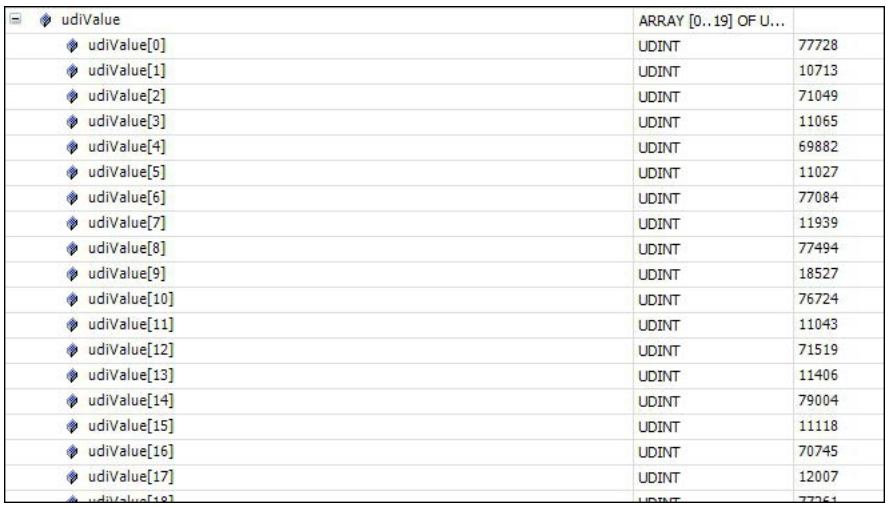
Detecting a PLC program with different runtimes

The PLC program must be supplemented in order to detect PLC programs with different runtimes. Different runtimes are not recognizable in the online view, since an average value is always formed over several cycles. Therefore, outliers can only be detected if they lie above the task time. If the outliers are still within the task time, they are not easily visible.

For this we then use the system variable: PlcTaskSystemInfo

```
VAR
    bOut : BOOL;
    PlcTaskSystemInfo : PlcTaskSystemInfo;
    udiValue : ARRAY[0..19] of UDINT;
    Cnt : INT;
END_VAR

Program:
bOut:=not bOut;

IF bOut THEN
    For loop:=1 to 2000 do
        lrTest:=SIN(INT_TO_LREAL(loop)*3.14);
    END_FOR
END_IF

PlcTaskSystemInfo:=_TaskInfo[1];

udiValue[Cnt]:= PlcTaskSystemInfo.LastExecTime;
cnt:=cnt+1;
IF Cnt >19 THEN
        Cnt:=0;
END_IF
```

With this program extension you can see that the PLC program with a For loop requires 7.7 ms and without a For loop 1.1 ms. The specification is 100 ns per digit.



udiValue	ARRAY [0..19] OF U...	
udiValue[0]	UDINT	77728
udiValue[1]	UDINT	10713
udiValue[2]	UDINT	71049
udiValue[3]	UDINT	11065
udiValue[4]	UDINT	69882
udiValue[5]	UDINT	11027
udiValue[6]	UDINT	77084
udiValue[7]	UDINT	11939
udiValue[8]	UDINT	77494
udiValue[9]	UDINT	18527
udiValue[10]	UDINT	76724
udiValue[11]	UDINT	11043
udiValue[12]	UDINT	71519
udiValue[13]	UDINT	11406
udiValue[14]	UDINT	79004
udiValue[15]	UDINT	11118
udiValue[16]	UDINT	70745
udiValue[17]	UDINT	12007

Fig. 43: Determination of different running times in the PLC program.

The measurement coincides with the displays on the oscilloscope, on which it can be seen that a pulse is sometimes 6.5 ms longer or 6.5 ms shorter. You can measure the processing time of the For loop (Measuring processing time in the PLC program [▶ 81]). The result of this measurement will coincide with the observed values through the program extension, with a certain inaccuracy and jitter.

9 Error handling and diagnostics

9.1 Diagnostic LEDs

Display	LED	Meaning
	TC	TwinCAT status LED: TwinCAT is in Run mode (green). TwinCAT is in Stop mode (red). TwinCAT is in Config mode (blue). Error or crash of the PLC (yellow). Displays errors at system startup by error code and error argument. The red LED flashes with two different frequencies.
	WD	No function. Reserved
	ERR	If only the ERR LED (red) lights up when starting the CX70xx, then the bootloader is damaged and the device must be sent in for repair.

The TC-LED flashes at a specified frequency and in a specified order, thus indicating the error code and argument.

Table 17: TC LED, order and meaning.

Sequence	Meaning
Fast flashing	Starting the sequence
First slow sequence	**Error code**
No display	Pause, the LED is off
Second slow sequence	**Error argument**

Count how many times the red TC LED flashes in order to determine the error code and argument.

Table 18: TC LED, error description and remedy.

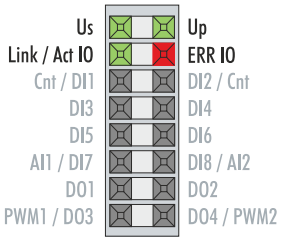
Error code	Error argument	Description	Remedy
1	1	microSD card not recognized	Check the microSD card.
	2	Card init failed - preloader	Image is defective. Install a new image on the microSD card.
	3	No partition found - preloader	
	4	Filesystem mount failed - preloader	
	5	Card init failed - loader	
	6	No partition found - loader	
	7	Filesystem mount failed - loader	
2	1	Loader not found	
	2	Loader file invalid (checksum, size, read error)	
	3	TC dll not found	
	4	TC dll checksum error	
	5	EEPROM file missing or invalid	
	6	TcOsSys.dll version not compatible with loader	
3	1	Rbf not found	
	2	CCAT 1 init failed	
	3	CCAT 2 init failed	
	4	CCAT EEPROM writing failed	

Error code	Error argument	Description	Remedy
	5	CCAT 1 EEPROM reloaded failed	
	6	CCAT 2 EEPROM reloaded failed	
4	1	Peripheral not working	Hardware defective, replace the CX
	2	Voltage Vo not reached	
	3	Low speed external oscillator not running	
	4	High speed external oscillator not running	
	5	Flash failed	
	6	Device overclocked (old Hardware)	
5	5	RAM error detected	

9.1.1 K-bus

The power supply unit checks the connected Bus Terminals for errors. The red LED "K-bus ERR" is off if no error is present. The red LED "K-bus ERR" flashes if Bus Terminal errors are present.

Table 19: Diagnostic LEDs in K-Bus mode.

Display	LED	Meaning
	Us 24 V	Power supply for basic CPU module. The LED lights green if the power supply is correct.
	Up 24V	Power supply for terminal bus. The LED lights green if the power supply is correct.
	K-BUS RUN	Diagnostic K-bus. The green LED lights up in order to indicate error-free operation. "Error-free" means that the communication with the fieldbus system is also running.
	K-BUS ERR	Diagnostic K-bus. The red LED flashes to indicate an error. The red LED flashes with two different frequencies.

The frequency and number of the flashes can be used to determine the error code and the error argument. An error is indicated by the "K-bus ERR" LED in a particular order.

Table 20: K-bus ERR LED, fault indication sequence through the LED.

Order	Meaning
Fast flashing	Starting the sequence
First slow sequence	**Error code**
No display	Pause, the LED is off
Second slow sequence	**Error code argument**

Count how often the red LED K-bus ERR flashes, in order to determine the error code and the error argument. In the error argument the number of pulses shows the position of the last Bus Terminal before the error. Passive Bus Terminals, such as a power feed terminal, are not included in the count.

Table 21: K-BUS ERR LED, fault description and troubleshooting.

Error code	Error code argument	Description	Remedy
Persistent, continuous flashing		EMC problems.	• Check power supply for undervoltage or overvoltage peaks. • Implement EMC measures.

Error code	Error code argument	Description	Remedy
			• If a K-bus error is present, it can be localized by a restart of the power supply unit (by switching it off and then on again).
3 pulses	0	K-bus command error.	• No Bus Terminal inserted. • One of the Bus Terminals is defective; halve the number of Bus Terminals attached and check whether the error is still present with the remaining Bus Terminals. Repeat this procedure until the faulty Bus Terminal has been found.
4 pulses	0	K-bus data error, break behind the power supply unit.	Check whether the Bus End Terminal 9010 is connected.
	n	Break behind Bus Terminal n.	Check whether Bus Terminal n+1 after the power supply unit is connected correctly; replace if necessary.
5 pulses	n	K-bus error in register communication with Bus Terminal n.	Replace Bus Terminal at location n.
6 pulses	0	Error at initialization.	Replace Embedded PC.
	1	Internal data error.	Hardware reset of the Embedded PC (switch off and back on again).
	8	Internal data error.	Hardware reset of the Embedded PC (switch off and back on again).
7 pulses	0	Process data lengths of the set and actual configurations do not correspond.	Check the configuration and the Bus Terminals for consistency.

For some error the LED "K-BUS ERR" does not go out, even if the error was rectified. Switch the power supply for the power supply unit off and back on again to switch off the LED after the error has been rectified.

State variable

In TwinCAT there is a State variable under the Bus Coupler for K-bus diagnostics.

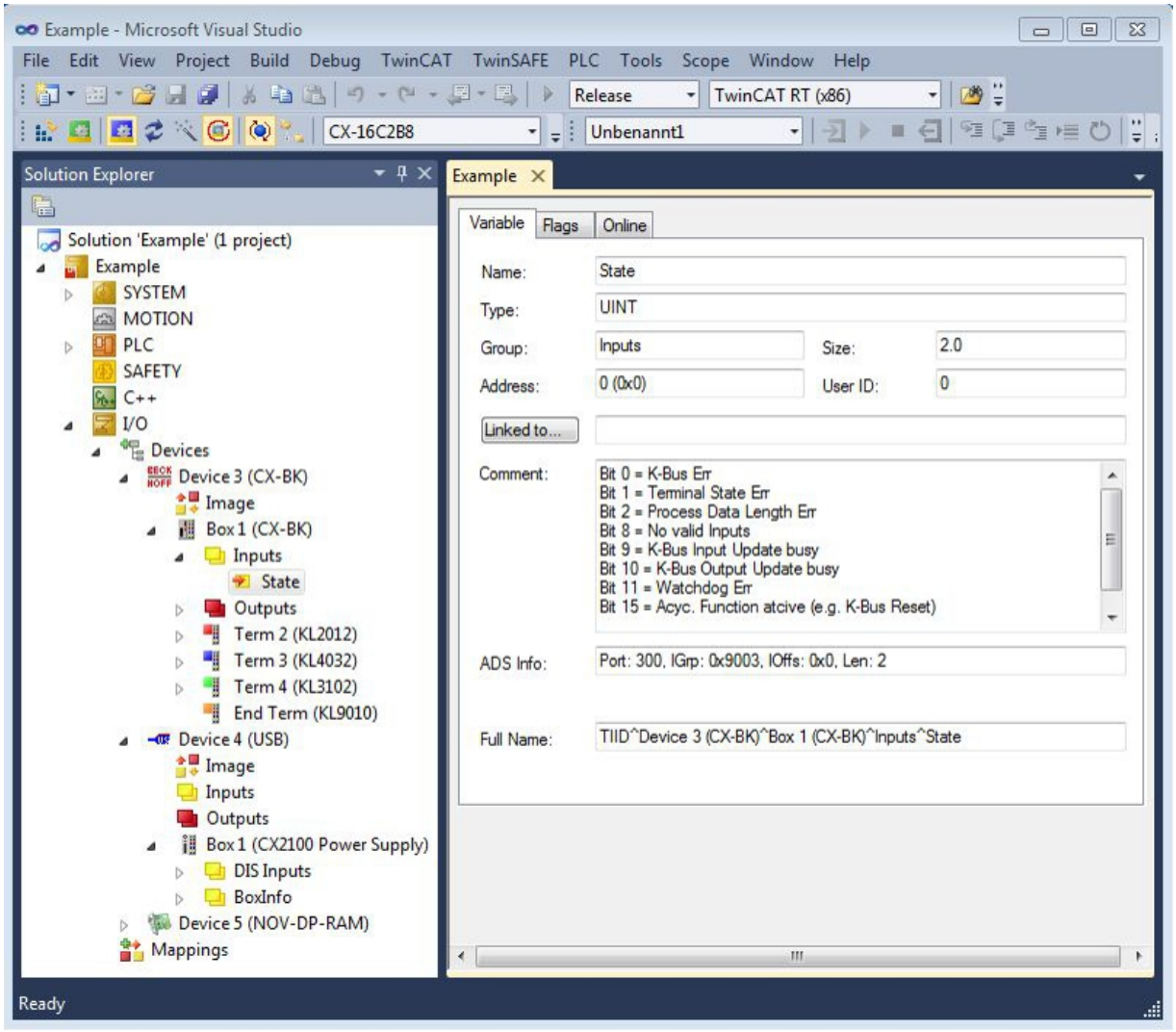


Fig. 44: Status variable for error handling and diagnostics under TwinCAT.

If the value is "0", the K-bus operates synchronous and without error. If the value is <> "0" there may be a fault, or it may only be an indication that the K-bus cycle is longer than the task. In which case it would no longer be synchronous with the task. The task time should be faster than 100 ms. We recommend a task time of less than 50 ms. The K-bus update time typically lies between one and five ms.

Table 22: Description of the State variable values.

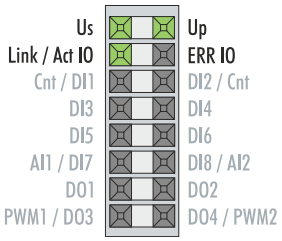
Bit	Description
Bit 0	K-bus error.
Bit 1	Terminal configuration has changed since the start.
Bit 2	Process image lengths do not match.
Bit 8	(still) no valid inputs.
Bit 9	K-bus input update not yet complete.
Bit 10	K-bus output update not yet complete.
Bit 11	Watchdog.
Bit 15	Acyclic K-bus function active (e.g. K-bus reset).

If there is a K-bus error, this can be reset via the IOF_DeviceReset function block (in the TcIoFunctions.lib).

9.1.2 E-bus

The power supply unit checks the connected EtherCAT Terminals. In E-bus mode the "Link/Act IO" LED is lit. When data are transferred, the "Link/Act IO" LED flashes.

Table 23: Diagnostic LEDs in K-Bus mode.

Display	LED		Meaning
	Us		Power supply for basic CPU module. The LED lights green if the power supply is correct.
	Up		Power supply for terminal bus. The LED lights green if the power supply is correct.
	Link/Act IO	off	E-bus not connected.
		on	E-bus connected / no data traffic.
		flashes	E-bus connected / data traffic on the E-bus.

9.2 Diagnosis of the multi-function I/Os

This chapter describes the diagnostic options for multi-function I/O communication. This is important, for example, if the 24 V power supply for the multi-function I/Os fails or the circuit breaker has triggered.

Status variable

The status variable `state` can be used for diagnostic purposes. In the normal state, the status variable takes the value `0x___8` (OP, Operational) and thus indicates that everything is error-free.

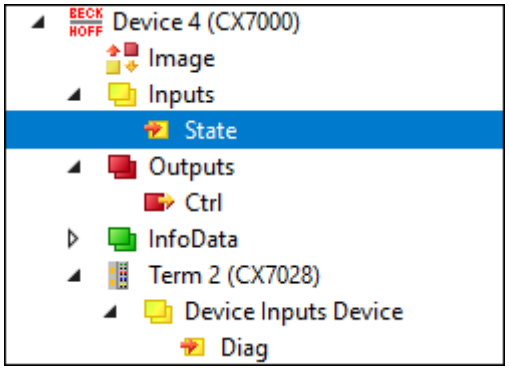


Fig. 45: Multi-function I/O status variable.

The following table shows which values the variables can assume:

Value	Meaning
0x___1	Slave in 'INIT' state
0x___2	Slave in 'PREOP' state
0x___3	Slave in 'BOOT' state
0x___4	Slave in 'SAFEOP' state
0x___8	Slave in 'OP' state
0x001_	Slave signals error
0x002_	Invalid vendorId, productCode... read
0x004_	Initialization error occurred
0x010_	Slave not present

If there is a power supply failure, the multi-function I/Os do not automatically go back into data exchange. To do this, the multi-function I/Os must be reset. A function block that can be used to reset the multi-function I/Os is the FB_CX70xx_ResetOnBoardIO [▶ 93] function block.

Notice : If outputs are still set in the PLC, the outputs of the multi-function I/Os are immediately reactivated as soon as the multi-function I/Os are reset with the function block.

Other diagnostic variables

The diagnostic variables `Diag` and `TxPDO State` are currently not in use and are reserved for future use. The variable `Input cycle counter`, on the other hand, increments with each cycle and indicates the number of I/O cycles exchanged with the multi-function I/Os. As soon as the variable is no longer incremented, no more I/O cycles are exchanged with the multi-function I/Os.

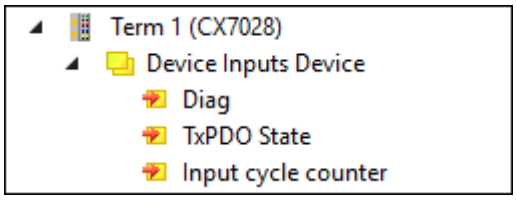


Fig. 46: Further diagnostic variables for multi-function I/Os

Variable	Meaning
Diag	Reserved, currently not used.
TxPDO State	Reserved, currently not used.
Input cycle counter	Incremented by 1 with each cycle. If this counter stops, then no more I/O cycles are exchanged with the multi-function I/Os.

9.3 Memory usage

The CX7000 has 32 MB of main memory that is used by the firmware (TwinCAT/RTOS) and TwinCAT (TwinCAT memory). The TwinCAT memory is further divided into the router memory and the PLC memory. The router memory is used for ADS communication and the PLC memory for the actual PLC program including TcConfiguration, mapping and data.

19.1 MB of TwinCAT memory are available to the CX7000. Because the size of the memory is limited, it is important to check the memory usage and to adapt your PLC project if it is exceeded.

Router memory

On the one hand, you can adjust the size of the router memory in TwinCAT and set a smaller router memory depending on the ADS communication actually used.

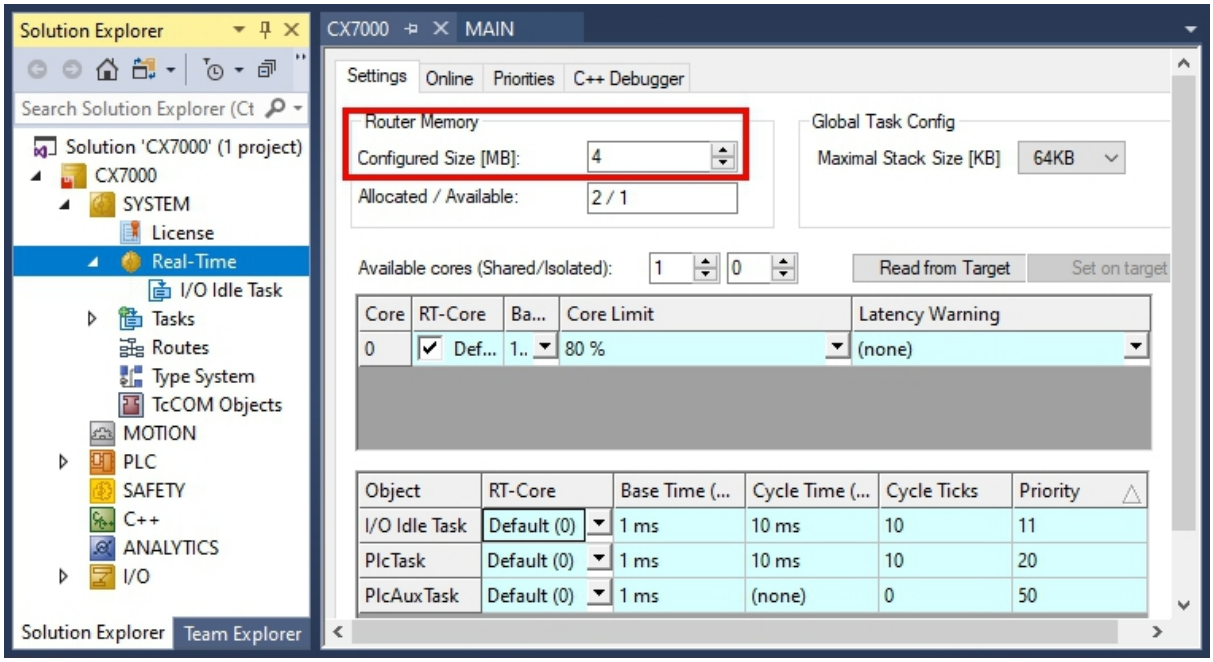


Fig. 47: Settings for router memory in the TwinCAT System Manager.

By default, a value of 32 MB is entered in TwinCAT, which in turn is limited to 9 MB for the CX7000 because of the small main memory in the CX7000. A router memory of 9 MB is usually far too large for a small controller. A router memory of 4 MB is recommended for the CX7000 and can be even smaller if little to no ADS communication is used. However, a router memory of at least 1 MB should be adhered to and should not be less. How much router memory is used can be determined with the function block FB_GetRouterStatusInfo or alternatively with the Beckhoff Device Manager.

Note that the router memory is only re-created with a power off/on of the CX7000. A TwinCAT restart is not sufficient. As a rule of thumb, the smaller the router memory for ADS communication is set, the larger the application can be, i.e. the PLC program, TcConfiguration, mapping and data.

Determining the memory usage

With the function block FB_GetRouterStatusInfo, or alternatively with the Beckhoff Device Manager, it is possible to determine how large the memory requirement of the router memory is.

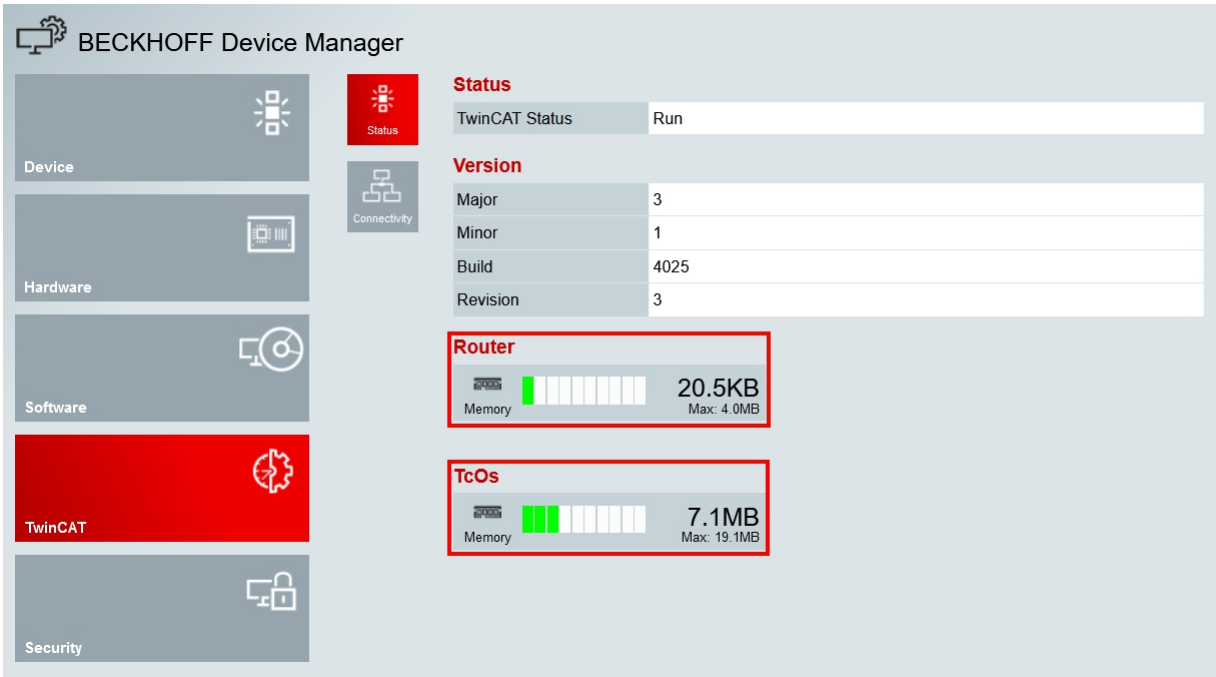


Fig. 48: Usage of the router and TwinCAT memory.

The **Router** display can be used to determine the memory requirements of the router memory. In this example, 20.5 kB of a maximum of 4 MB are used. The **TcOs** display shows the total memory usage of the TwinCAT memory including router memory and PLC program. In this example, a total of 7.1 MB is used.

With the help of this display, the size of the PLC program can also be calculated, as the router memory is fixed at 4 MB and is part of the TwinCAT memory. If you subtract the 4 MB from 7.1 MB, the PLC program occupies 3.1 MB.

Memory reserve

Since 7.1 MB of the 19.1 MB TwinCAT memory are used in this example, a reserve of 12 MB remains for the PLC program. Note that more memory is needed for a short time for an Online Change in TwinCAT. If you want to use the Online Change function, it is advisable to always have a certain reserve. In the most extreme case, twice the currently consumed PLC program may be required to execute an Online Change. An error message is displayed in TwinCAT if there is not enough memory available for the Online Change.

9.4 Real-time and CPU load

For the proper functioning of the CX7000, it is important to keep an eye on CPU load and real-time compliance. Otherwise, the CX7000 will no longer work reliably in the event of an overload. Note that in the event of an overload, the load indicator is also affected and no longer provides current values. For example, a load of 40% can be incorrectly displayed, but the PLCs are no longer working in real time and the system is overloaded. You should therefore gradually approach the load limit with a small controller.

What is meant by real-time in this context? By default, the PLC works in synchronization with the cycle, which means that a task time is always defined and called at a fixed time. The PLC works in synchronization with the cycle if the task time is not exceeded. For example, if you define a task time of 10 ms and the PLC only needs 2 ms for processing, the selected task time is fine and the PLCs work in synchronization with the cycle.

Even if you do not need the real-time, it is recommended to adhere to the real-time, because otherwise negative effects can occur. These could be connection problems or problems with subsystems such as K-bus or EtherCAT. You can perform the following steps to check whether the CX7000 is optimally set or rather overloaded:

- Observe the exceed counter.
- Check the CPU load.

Observe the exceed counter

The exceed counter is incremented as soon as the PLC no longer works in synchronization with the cycle and the defined task time is exceeded. Ideally, the counter value should be zero.

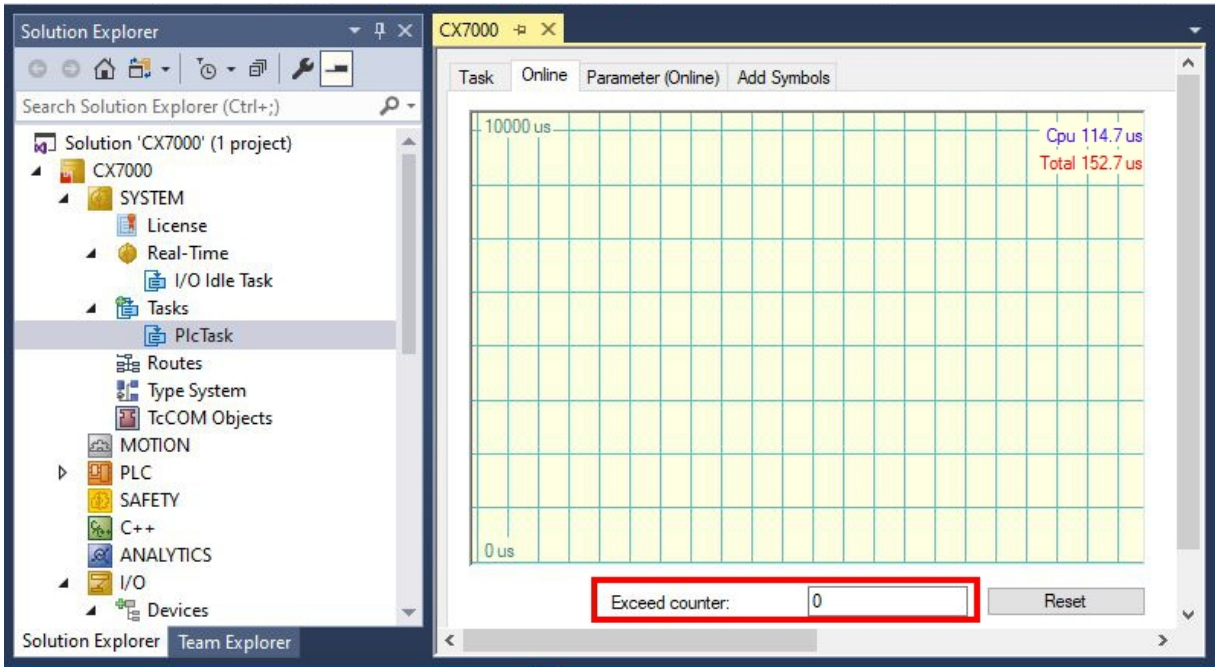


Fig. 49: Display of the exceed counter in TwinCAT.

It is possible for the exceed counter to be incremented at the start of the PLC, for example, because the PLC is called for the first time or certain components are initialized. Observe the exceed counter over a period of several hours. One can only speak of a stable state when the exceed counter is no longer incremented over a longer period of time.

Check the CPU load

In TwinCAT, the CPU load is displayed under Realtime and on the Online tab. Check the value to determine whether you can run additional program code or reduce the task time.

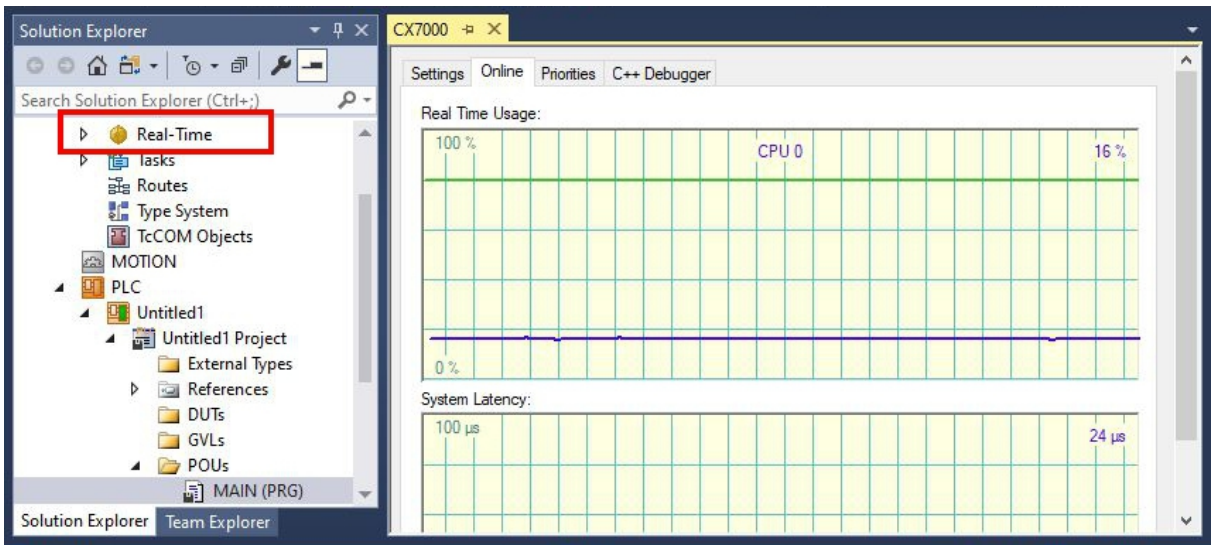


Fig. 50: Display of the CPU load in TwinCAT.

The light green line indicates the preset CPU limit. If the load is ≥ 65%, the CX7000 is already very busy and no more code should be executed or the task time shortened. You should not go to the limit and use the CX7000 to full capacity.

Measures in the event of overload

If an overload is detected with the help of the steps shown, the load can be reduced by improving the programming or increasing the task time. To find places in the program code with long processing times, the sample in Measuring processing time in the PLC program [▶ 81] can be used.

The selected terminal system also has an influence on the real-time. Depending on the number of terminals, the K-bus, for example, can also take several milliseconds and must be taken into account when choosing the task time. It may well be that, with a set task time of 10 ms, the PLC program only needs 5 ms, but the exceed counter still increments. This is due to the fact that the K-bus requires more than 5 ms for processing and the task time of 10 ms including PLC program and K-bus is exceeded. This problem can be solved by reducing the number of terminals or increasing the task time.

By default, the real-time is set to 80%. This is already the maximum value and an increase to 90% is equivalent to an increase to 100%.

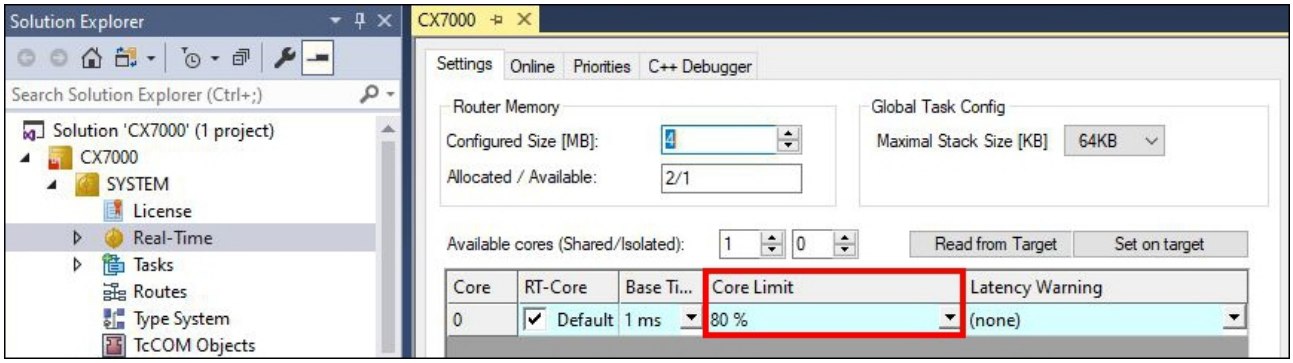


Fig. 51: Setting the real-time load in TwinCAT.

TwinCAT would then consume all the CPU power, and services that the operating system serves would no longer work or would not work adequately. If you increase the real-time load to 90%, you should be aware of the potential consequences for the operating system.

10 Technical data

Table 24: Technical data, dimensions and weights.

	CX7000
Dimensions (W x H x D)	49 mm x 100 mm x 73 mm
Weight	142 g

Table 25: Technical data, general data.

Technical data	CX7000
Processor	Arm® Cortex®-M7, 480 MHz
Number of cores	1
Flash memory	512 MB MicroSD (optional 16 GB)
Main memory	32 MB SDR (internal, not expandable)
Number of inputs	8 multifunction inputs (24 V DC)
Number of outputs	4 multifunction outputs (24 V DC, 0.5 A, 1-wire technique)
NOVRAM	4 kB
Interfaces	1 x RJ45 10/100 Mbit/s, 1x USB (max 12 Mbit/s, max 100 mA)
Cooling	passive
Bus interface	–
Data transfer rate	–
Diagnostic LED	1 x TC Status, 1 x WD LED, 1 x ERR LED
Clock	internal, capacitor-buffered real-time clock for time and date (memory > 21 days)
Operating system	TwinCAT/RTOS
Control software	TwinCAT 3 Runtime (XAR)
Power supply	24 V_{DC} (-15 %/+20 %)
Max. power consumption	< 2 W (max. 12 W with E-bus/K-bus)
Max. power consumption E-bus/K-bus	7.5 W
TwinCAT 3 functions included	TC1000 TwinCAT 3 ADS, TC1100 TwinCAT 3 I/O, TC1200 TwinCAT 3 PLC, TF4100 TwinCAT 3 Controller Toolbox, TF4110 TwinCAT 3 Temperature Controller, TF6255 TwinCAT 3 Modbus RTU, TF6340 TwinCAT 3 Serial Communication, TF6701 TwinCAT 3 IoT Communication (MQTT), TF6730 TwinCAT 3 IoT Communicator
Approvals	CE, UL

*) Image version 114606 and TwinCAT 3 XAE 4024.47 or higher required.

Table 26: Technical data, I/O terminals.

Technical data	CX7000
I/O connection	via power supply terminal (E-bus or K-bus, automatic recognition)
Power supply for I/O terminals	max. 1.5 A (installation position any, temp. -25...45 °C) max. 1.3 A (installation position horizontal, temp. -25...55 °C) max. 1 A (installation position any, temp. -25...55 °C) max. 1 A (installation position horizontal, temp. -25...60 °C)
Power contacts current load	max. 10 A
Process data on the K-bus	max. 512 bytes In and 512 bytes Out
max. number of terminals (K-bus)	64 (255 with K-bus extension)
E-bus process data	max. 4 kB In and 4 kB Out
max. number of terminals (E-bus)	up to 65534 terminals.

Table 27: Technical data, environmental conditions.

Technical data	CX7000
Ambient temperature during operation	-25° C ... +60 °C
Ambient temperature during storage	-40° C ... +85° C see notes under: Transport and storage [▶ 12]
Relative humidity	95 % no condensation
Vibration resistance	conforms to EN 60068-2-6
Shock resistance	conforms to EN 60068-2-27
EMC immunity	conforms to EN 61000-6-2
EMC emission	conforms to EN 61000-6-4
Protection rating	IP20

Table 28: Technical data, Ethernet interface X001.

Technical data	CX7000
Data transfer medium	4 x 2 twisted pair copper cables category 5 (100 Mbit/s)
Cable length	100 m from switch to CX7000
Data transfer rate	10/100 Mbit/s
Topology	star wiring
Protocols	all non-real-time-capable protocols that are based on TCP or UDP and do not require a real-time extension

11 Appendix

11.1 Third-Party components

This device contains Beckhoff software and third-party software.
Please refer to the license file on the storage medium.

11.2 Accessories

Table 29: microSD cards.

Order number	Description
CX1900-0122	512 MB microSD card
CX1900-0132	16 GB microSD card

Table 30: Further spare parts.

Order number	Description
ZB8701	Slotted screwdriver 2.0 x 40 mm, HD terminals

11.3 Certifications

FCC Approvals for the United States of America

FCC: Federal Communications Commission Radio Frequency Interference Statement

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

FCC Approval for Canada

FCC: Canadian Notice

This equipment does not exceed the Class A limits for radiated emissions as described in the Radio Interference Regulations of the Canadian Department of Communications.

List of tables

List of figures

Trademark statements

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Third-party trademark statements

Arm, Arm9 and Cortex are trademarks or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

More Information:
www.beckhoff.com/CX7000

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com