



TIA-PRO1

SITRAIN

Training for Industry

SIMATIC TIA Portal Programming 1

siemens.com/sitrain



SITRAIN
Training for Industry

SIMATIC S7

**TIA Portal
Programming 1**

Course TIA-PRO1

Name: _____

Course from:_____ to:_____

Instructor: _____

Location: _____

This document was produced for training purposes.

SIEMENS assumes no responsibility for its contents.

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable to damages.

Copyright © Siemens AG 2018. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

SITRAIN course offer on the Internet: www.siemens.de/sitrain

Training Document Version: V15.00.00
(for STEP7 Version 15)

- 1 Training Device and Addressing
- 2 System Overview
- 3 Engineering Software TIA Portal
- 4 Devices and Networks: Online Functions and Hardware Configuration
- 5 PLC Tags
- 6 Binary Operations 1
- 7 Program Blocks
- 8 Binary Operations 2
- 9 Functions and Function Blocks
- 10 Digital Operations
- 11 Data Blocks
- 12 Connecting an HMI Device
- 13 Organization Blocks
- 14 Distributed I/O
PROFINET and PROFIBUS
- 15 Troubleshooting
- 16 Integrating and Commissioning a Drive with Startdrive
- 17 Training and Support
- 18
- 19
- 20

Contents

1.	Training Devices and Addressing.....	1-2
1.1.	Training Area Setup	1-3
1.2.	Training Case.....	1-4
1.2.1.	Configuration of the Controller (S7-1500).....	1-5
1.2.2.	Configuration of the I/O-Device (ET 200SP).....	1-6
1.2.3.	Operating and Display Elements of the Training Device	1-7
1.3.	Setup and Connection of the Conveyor Model.....	1-8
1.3.1.	Connection to Central I/O of the S7-1500.....	1-8
1.3.2.	Connection to Distributed I/O of the ET 200SP	1-9
1.4.	Networking and IP Addresses of the Modules.....	1-10
1.5.	Training Area as Plant with Distribution Conveyor and Touchpanel	1-11

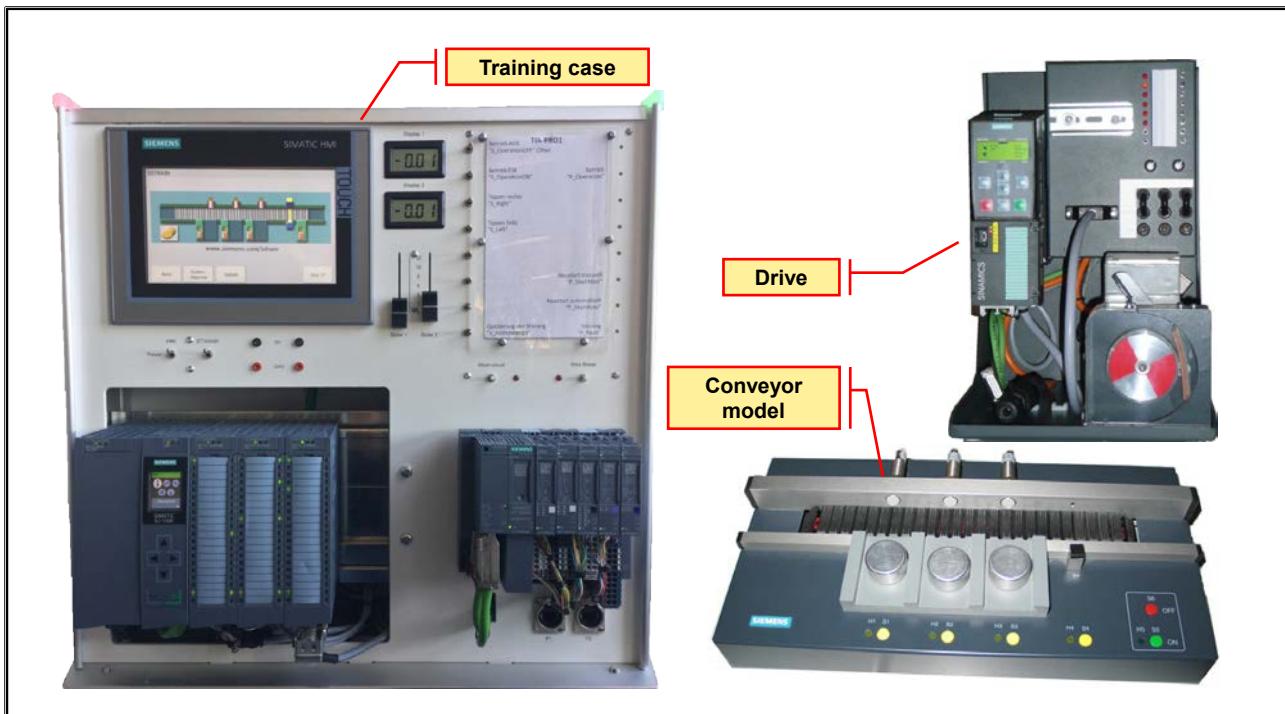
1. Training Devices and Addressing

At the end of the chapter the participant will ...



- ... be familiar with the configuration of the training area
- ... be familiar with the wiring of the training area components

1.1. Training Area Setup



Components of the Training Area

The training area for this course contains the following components:

- Training case
- Sinamics G120 drive
- Conveyor model
- And additionally, a SIMATIC Field PG or PC for configuring and programming

1.2. Training Case

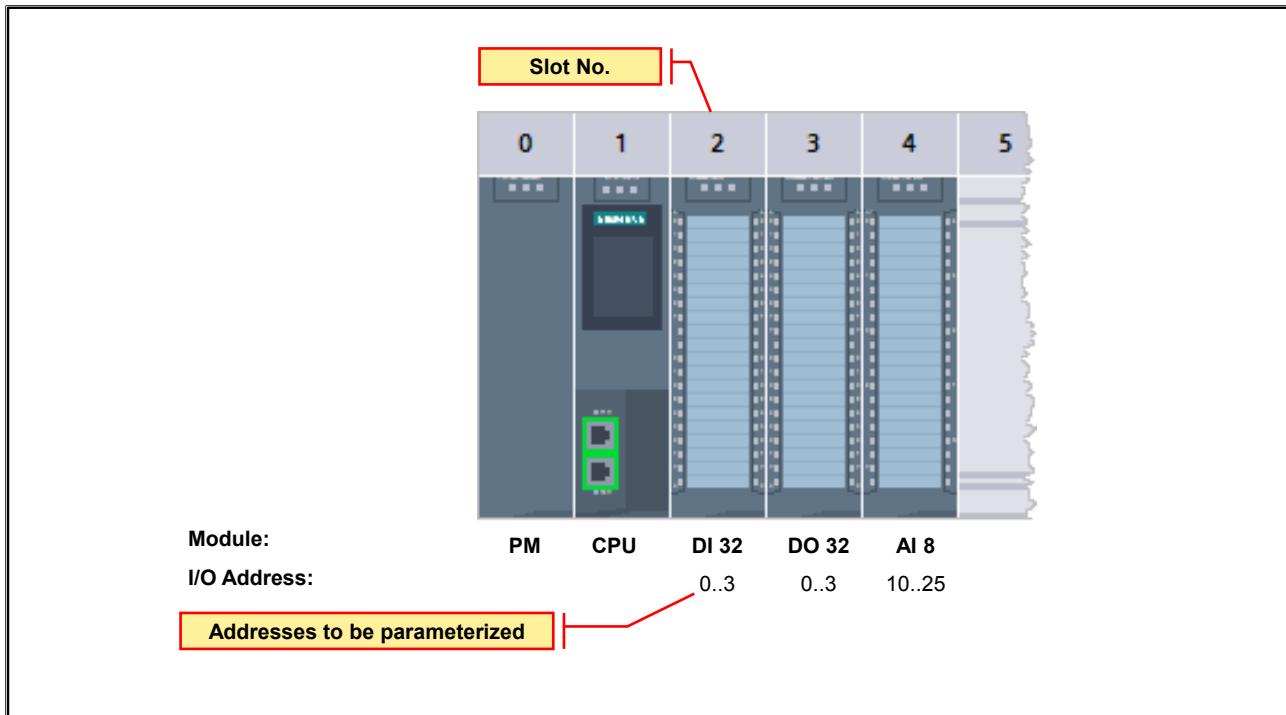


Components of the Training Case

The training case consists of the following components:

- S7-1500 controller with central I/O,
- I/O-Device (ET 200SP) with distributed I/O,
- Touchpanel as operator control and monitoring device (HMI),
- An operating tableau for digital signals with input elements (switches) and display elements (lights [LEDs])
- An analog display and operating area with slide controls and voltage displays

1.2.1. Configuration of the Controller (S7-1500)

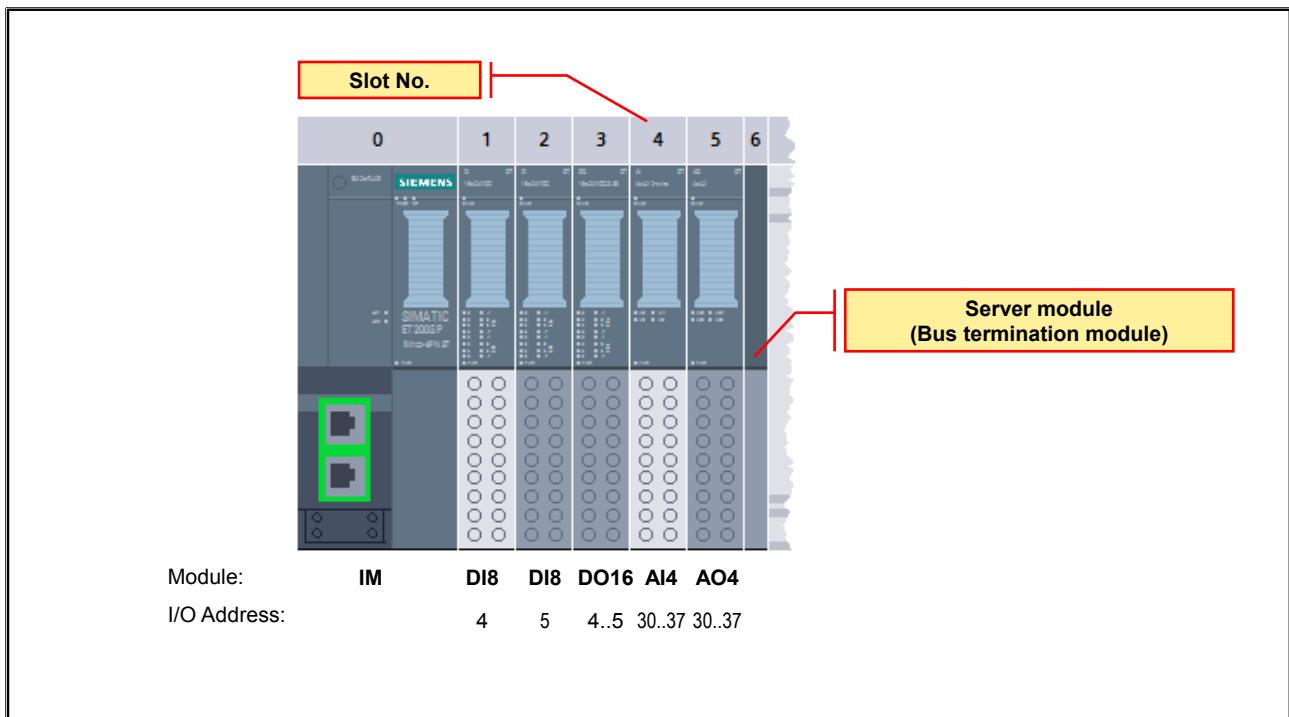


Addresses of the Central S7-1500 I/O Modules

Two digital 32-channel modules are available as central I/O. These are to begin as of Address =0.

Since digital channels are also available on the distributed I/O, the analog module of the central I/O is to begin as of Address =10.

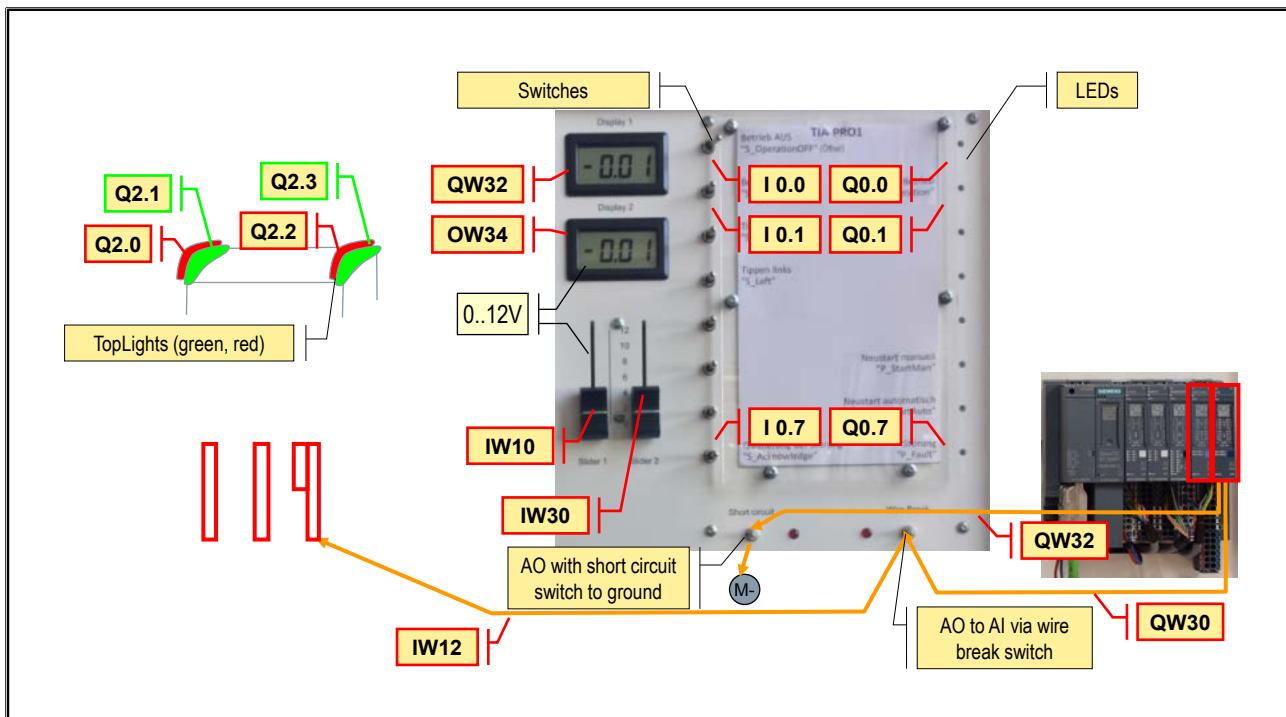
1.2.2. Configuration of the I/O-Device (ET 200SP)



Three digital modules are available as distributed I/O. These are to follow the central digital I/O in the address space as of Address =4.

The analog distributed I/O is to begin as of Address =30.

1.2.3. Operating and Display Elements of the Training Device



Operating Elements

In addition to the touchpanel, separate operating elements are also available for operating the system:

- 8 switches
- 2 potentiometers for setting or simulating analog input signals
- Wire break switch that interrupts the connection AO1 distributed I/O to AI2 central I/O
- Short circuit switch that short-circuits the AO2 of the distributed I/O to ground

Display Elements

In addition to the touchpanel, separate display elements are also available for the visualization of process information:

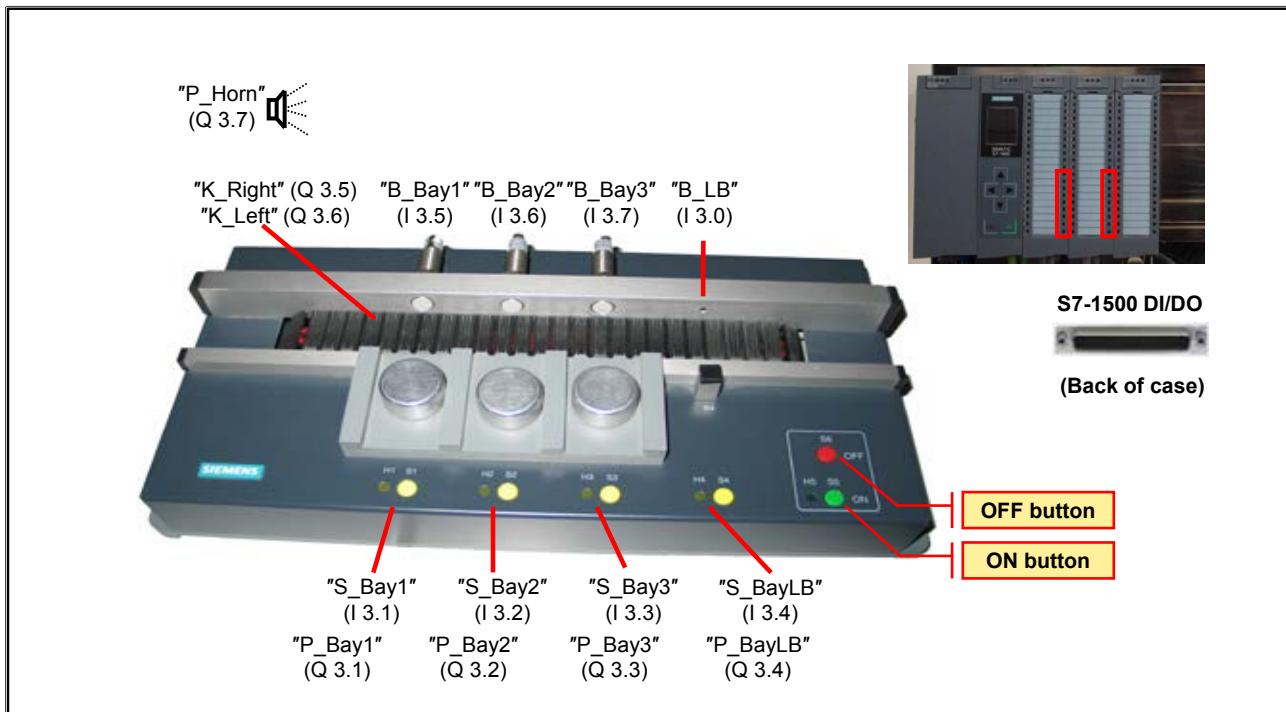
- 8 LEDs
- 2 digital voltage displays for displaying analog output signals
- On top of the training device there are, to the right and left, 2 LED bars "TopLights" (2x green, 2x red). These can be controlled by means of 4 DOs.

Addressing

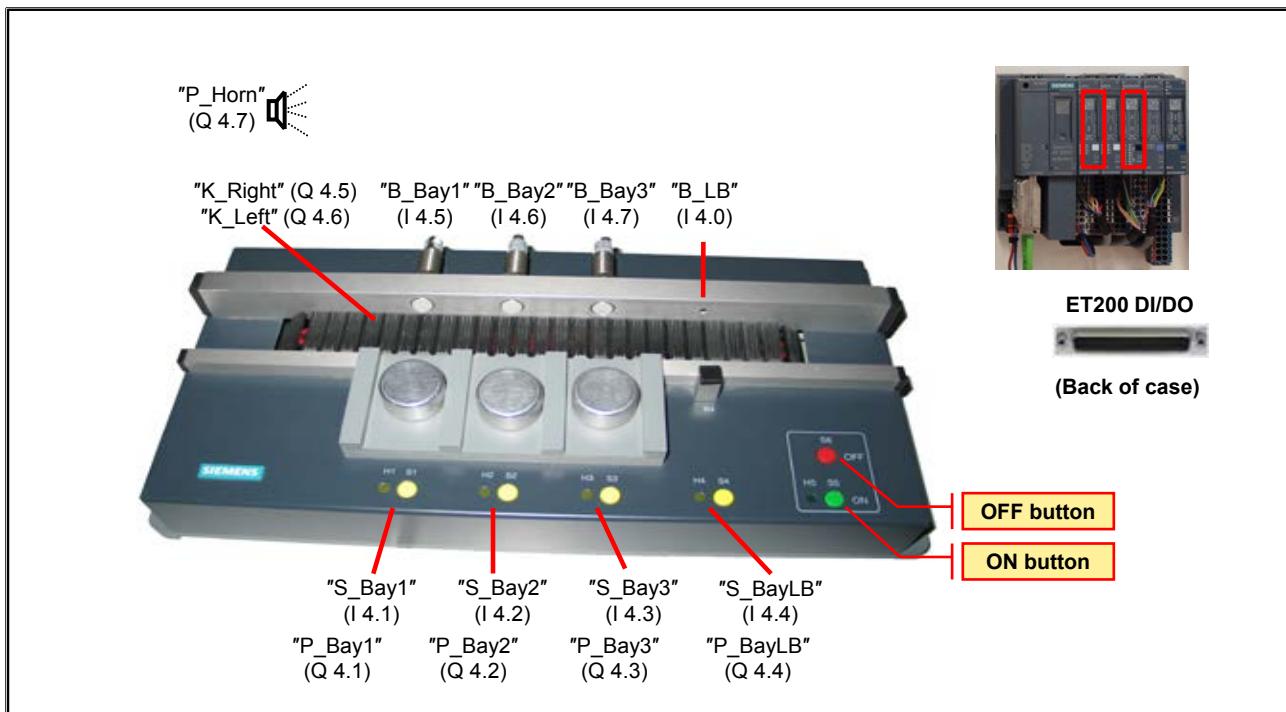
For the addressing shown in the picture, the relevant module address settings must be made in the device configuration.

1.3. Setup and Connection of the Conveyor Model

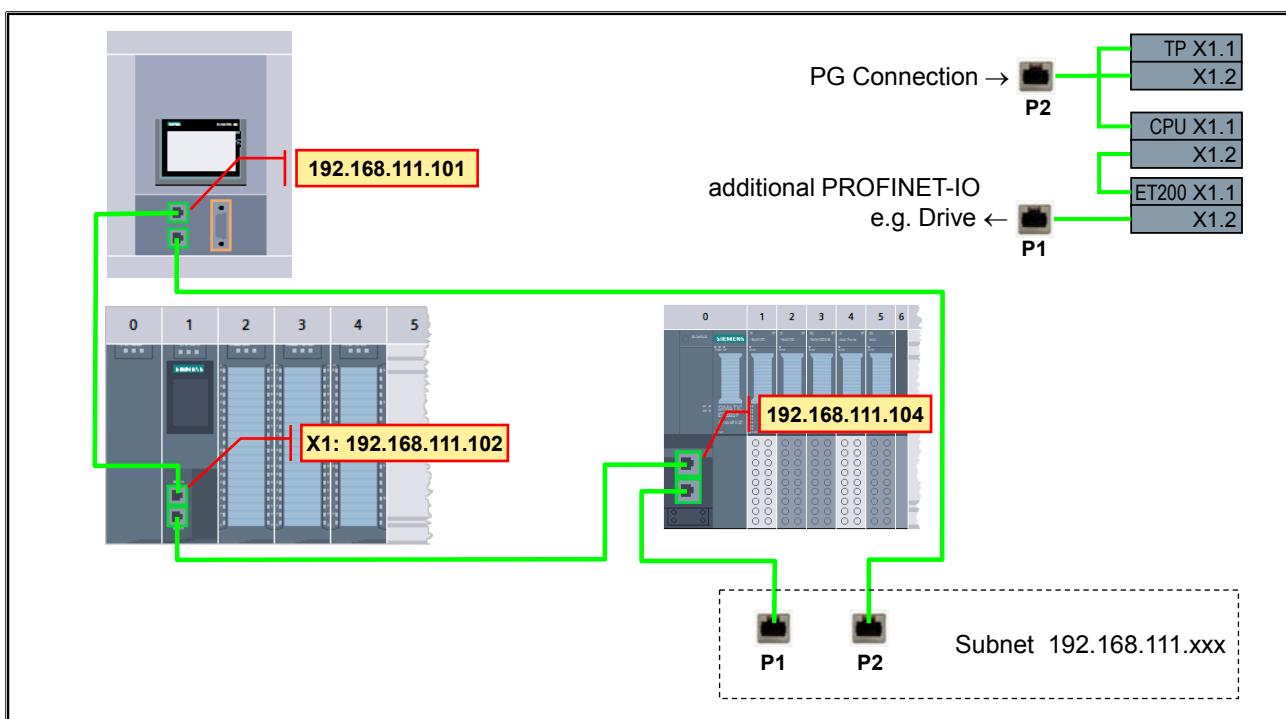
1.3.1. Connection to Central I/O of the S7-1500



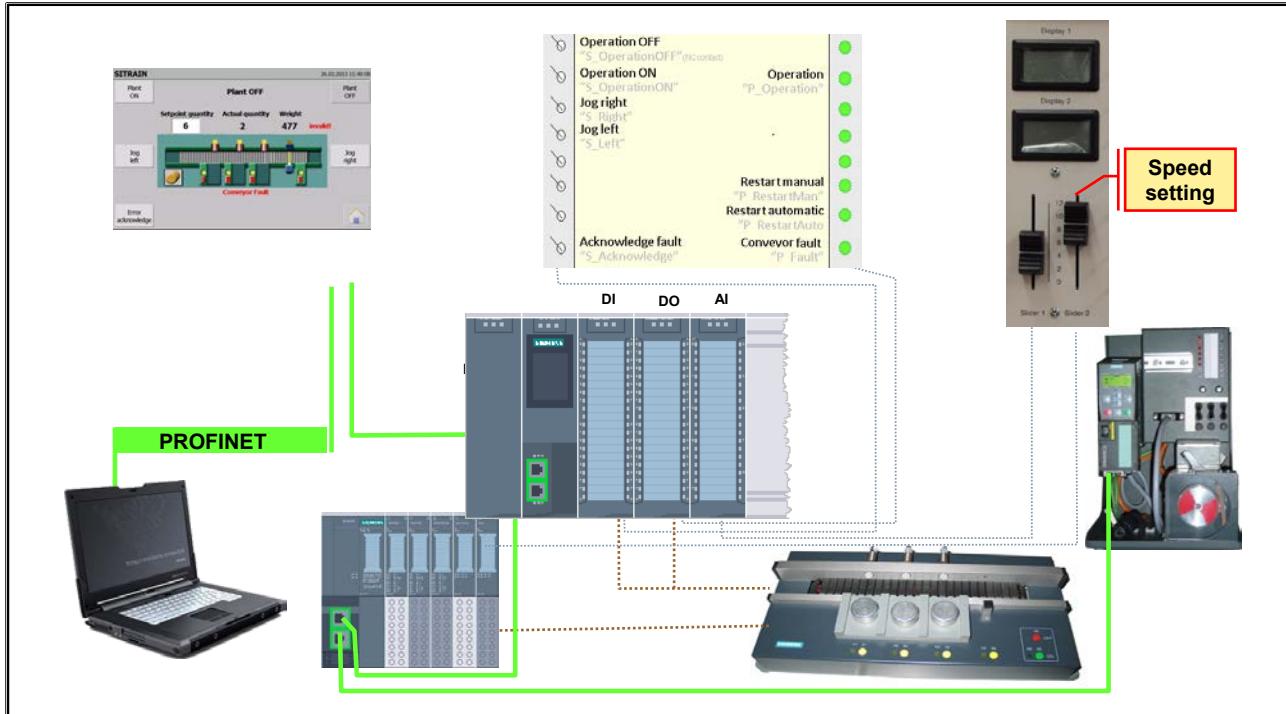
1.3.2. Connection to Distributed I/O of the ET 200SP



1.4. Networking and IP Addresses of the Modules



1.5. Training Area as Plant with Distribution Conveyor and Touchpanel



Function Description:

The distribution conveyor is used to transport parts and can be operated in two different operating modes.

For now, the simulator switches are used to select the operating mode and later, the associated buttons on the touchpanel.

Operation Switched Off "P_Operation" (Q0.1) = Off

In Manual mode ("P_Operation" = FALSE), the conveyor motor can be jogged to the right and left. For now, the simulator switches are to be used for this, later, the associated buttons on the touchpanel.

Later, the G120 drive will also be controlled as if it would drive the conveyor. The speed of the motor is preset through a parameter in the converter.

Continued on the next page

Operation Switched On "P_Operation" (Q0.1) = On

When Automatic mode ("P_Operation" = TRUE) is switched on, parts are transported on the conveyor model from Bay 1 or 2 to the right until they are through the light barrier.

If a transport sequence takes longer than 6 seconds, the conveyor motor is automatically switched off and the fault is displayed on the simulator as well as on the touchpanel. Only after the fault is acknowledged with the simulator switch or on the touchpanel, can a new transport sequence be started.

All parts that pass the light barrier when in Automatic mode ("P_Operation" = On) are counted.

If the setpoint quantity (can be preset on the touchpanel) of parts is reached, it is indicated on the conveyor model LED ("P_BayLB") of the light barrier bay with a 1Hz flashing light. Only after the message is acknowledged with the pushbutton ("S_BayLB") of the light barrier bay or renewed Automatic mode ON-OFF, can a new transport sequence be started.

The indicator lights at the Bays 1 and 2 show...

- Continuous light when a new part can be placed on the conveyor.
- 1Hz flashing light at the Bay at which a part is detected by the associated proximity switch, however, only as long as the conveyor has not yet been started.
- 2Hz flashing light as long as the conveyor motor is running.

The indicator light at the Light Barrier Bay shows...

- 2Hz flashing light as long as the conveyor motor is running.
- Continuous light when the SETPOINT quantity has been reached.

Later, the drive will also be controlled as if it would drive the conveyor. The speed of the motor can be set through the right slider "S_Slider2".

Contents

2

2.	System Overview.....	2-2
2.1.	SIMATIC S7 Overview	2-3
2.2.	TIA Portal Information Center	2-4
2.3.	Overview Controller	2-5
2.3.1.	Positioning the Modular S7 Controllers	2-6
2.4.	Overview: Available Modules.....	2-7
2.4.1.	Central Modules.....	2-7
2.4.2.	Signal Modules (Central)	2-8
2.5.	SIMATIC S7-1200: The Modular Mini-PLC.....	2-9
2.5.1.	SIMATIC S7-1200: Modules	2-10
2.5.2.	SIMATIC S7-1200: Installation and Mounting Positions.....	2-11
2.5.3.	SIMATIC S7-1200: Signal, Communication or Battery Board	2-12
2.6.	SIMATIC S7-1500: Modular Controller for the Mid to Upper Performance Range.....	2-13
2.6.1.	SIMATIC S7-1500: Modules	2-14
2.6.2.	I/O Addressing of the S7-1500.....	2-16
2.6.2.1.	Channel Addressing of Digital S7-1500 Modules	2-17
2.6.3.	SIMATIC S7-1500: Installation and Mounting Positions	2-18
2.6.4.	SIMATIC S7-1500: Connection Technology / Front Connector.....	2-19
2.6.5.	SIMATIC S7-1500: CPU-Display → Overview	2-20
2.6.6.	SIMATIC S7-1500: CPU-Display → Menu and Colors	2-21
2.7.	SIMATIC S7-1200/1500: Memory Card(s).....	2-22
2.8.	Exercise 1: Display	2-23
2.9.	Exercise 2: Loading the Program onto the SMC	2-24
2.9.1.	Exercise 3: Diagnostics and Program Test.....	2-25
2.10.	Additional Information	2-26
2.10.1.	ET 200SP and ET 200pro Controller	2-27
2.10.2.	Software Controller	2-28
2.10.3.	ET 200SP Open Controller "All in one"	2-29
2.10.4.	SIMATIC S7-300: Modular Automation System	2-30
2.10.4.1.	SIMATIC S7-300: Modules	2-31
2.10.5.	SIMATIC S7-400: Modular Automation System	2-32
2.10.5.1.	SIMATIC S7-400: Modules	2-33

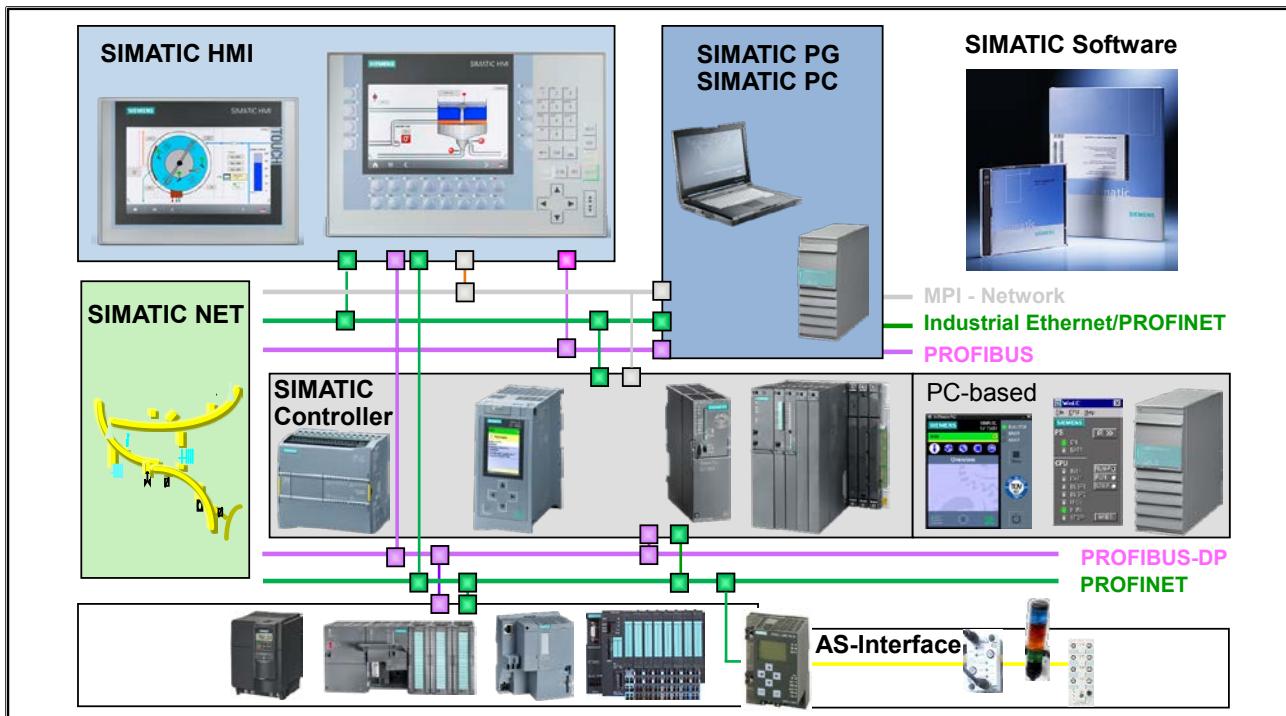
2. System Overview

At the end of the chapter the participant will ...



- ... be familiar with the concept of "Totally Integrated Automation" (T.I.A.)
- ... be familiar with the TIA Portal Information Center
- ... have an overview of the available modules
- ... have an overview of the new SIMATIC S7-1200/1500 system family
- ... be familiar with the SIMATIC Memory Card as well as the S7-1500 Display
- ... know the S7-300 and S7-400 automation systems

2.1. SIMATIC S7 Overview



Introduction

For the operation of machines, equipment and processes in almost all areas of manufacturing you require control elements in addition to energy supply. It must be possible to initiate, control, monitor and end the operation of any given machine or process.

Hard-wired Programmed Controller → PLC

In the hard-wired controllers of the past, the program logic was governed by the task-specific wiring of contactors and relays.

Today, programmable logic controllers are used to solve automation tasks. The logic stored in the program memory of an automation system does not depend on equipment design and wiring and can be modified at any time with the help of a programming device.

Totally Integrated Automation

Production processes are no longer seen as individual partial processes, but rather as integral components of an entire production process. The total integration of the entire automation environment is today achieved with the help of:

- one common software environment that integrates all components and tasks into one uniform easy to use system
- a common data management (central database)
- a common communication between all participating automation components.

2.2. TIA Portal Information Center

<https://support.industry.siemens.com>
Entry ID 65601780

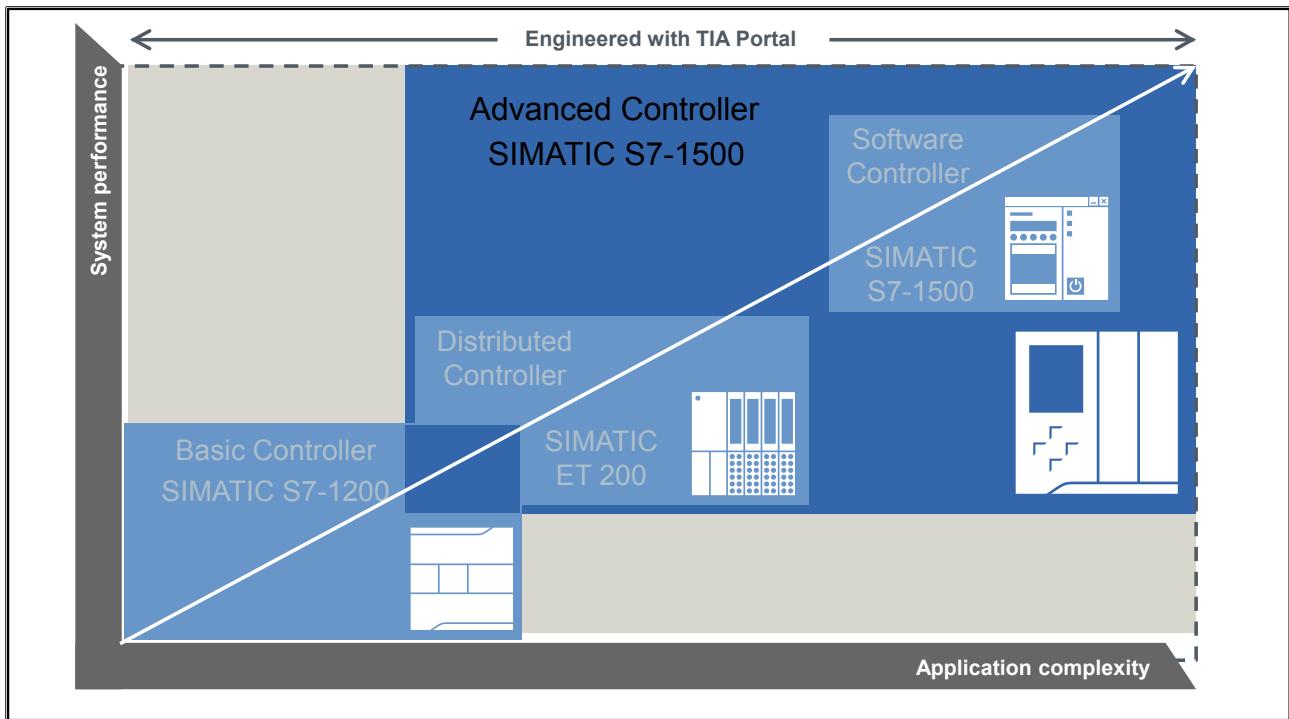
The screenshot shows the SIMATIC TIA Portal Information Center website. At the top left is the Siemens logo with the tagline "Ingenuity for life". At the top right is a language selection button and the text "TIA Portal Information Center". Below the header is a grid of twelve cards with various links: "First steps", "Application examples", "Product information", "Tools & Apps", "Training / Kurse", "Migration", "Dokumentation", "Downloads", "News", "References", "Ordering & Licenses", and "Social Media". To the right of the grid is a photograph of an open laptop displaying the TIA Portal software interface. The bottom of the page has a footer with copyright information: "© Siemens AG, 1996 – 2018 | Impressum | Datenschutz | Nutzungsbedingungen | Digitales Zertifikat".

By entering the Product/Article No. (Entry ID) 65601780, you arrive at the start page "TIA Portal - An Overview of the Most Important Documents and Links".

Here you will find all important documents and links about the TIA Portal as well as the controllers S7-1200 and S7-1500.

In addition, you can get to the "TIA Portal Information Center". Through it you can also get to all important links and information.

2.3. Overview Controller



Depending on the complexity, different controllers from S7-1200 to S7-1500 can be used.

2.3.1. Positioning the Modular S7 Controllers



SIMATIC S7

The programmable logic controllers can be divided into the performance ranges Basic (S7-1200) and Advanced (S7-1500).

The product range of the S7-1200 and S7-1500 will be expanded in the next few years such that the S7-200, S7-300 and even the S7-400 can be completely replaced.

2.4. Overview: Available Modules

2.4.1. Central Modules

	S7-1200	S7-1500	S7-300	S7-400
Standard	✗	✓	✓	✓
Fail-safe	✓	✓	✓	✓
Compact	✓	✓	✓	✗
High availability	✗		✗	✓
Technology	✓ Different functions	Different functions T-CPU ✓	✓ T-CPU	✗

More Information under the Link:

TIA Portal Information Center > Product information > Controllers

2.4.2. Signal Modules (Central)

	S7-1200	S7-1500	S7-300	S7-400
DI/DQ	✓	✓	✓	✓
AI/AQ	✓	✓	✓	✓
F-DI/F-DQ	✓	✓	✓	✗
F-AI			✓	✗

More Information under the Link:

TIA Portal Information Center > Product information > Controllers

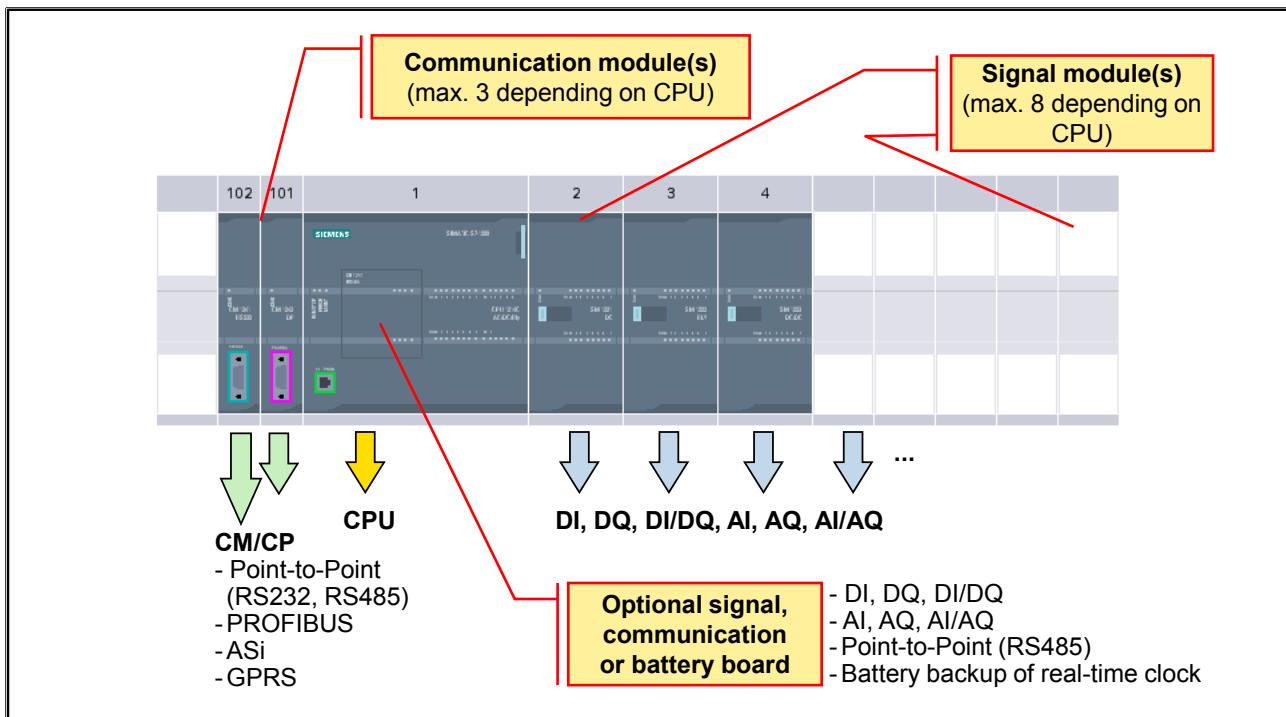
2.5. SIMATIC S7-1200: The Modular Mini-PLC



Features

- Modular compact control system for the low-end performance range
- Scaled CPU range
- Extensive range of modules
- Can be expanded to up to 11 modules (depends on the CPU)
- Can be networked with PROFIBUS or PROFINET
- Slot rules
- CM left of the CPU (number depends on the CPU)
- SM right of the CPU (number depends on the CPU)
 - "Total package" with CPU and I/O in one device
- integrated digital and analog I/O
- an expansion with signal board
 - "Micro PLC" with integrated functions

2.5.1. SIMATIC S7-1200: Modules



Slot Rules

- CM left of the CPU (number depends on the CPU)
- Signal modules (digital, analog) right of the CPU (number depends on the CPU)

Signal Modules

- Digital input, output or mixed modules (24VDC, relay)
- Analog input, output or mixed modules (voltage, current, resistance, thermocouple)

Communication Modules (CM - Communication Module, CP - Communication Processor)

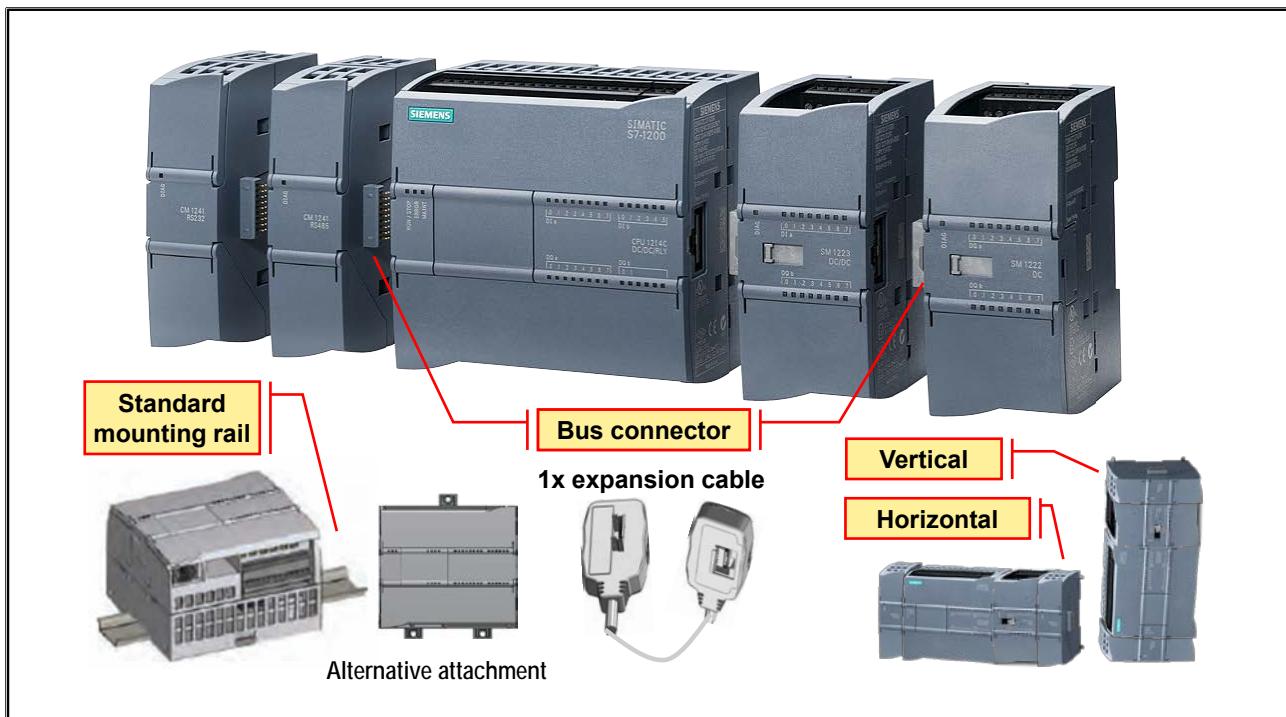
- Point-to-point connection (RS232, RS485)
- PROFIBUS
- ASI-Master
- Telecontrol (GPRS functionality)

Expansion Board

With this, the CPU can be expanded by onboard I/O or an interface.

A battery board ensures the long-term battery backup (buffering) of the real-time clock.

2.5.2. SIMATIC S7-1200: Installation and Mounting Positions



Installation

The modules are mounted on a standard mounting rail or alternatively screwed into the control cabinet.

S7-1200 Expansion Cable

It offers additional flexibility in configuring the S7-1200 system. One expansion cable can be used for each CPU system.

- Either between the CPU and the first SM or between two SMs

Bus Connector

It is located as a mechanical slide on the left side of the SM modules.

It is mechanically attached on the right side of the CMs/CPs.

Mounting Positions

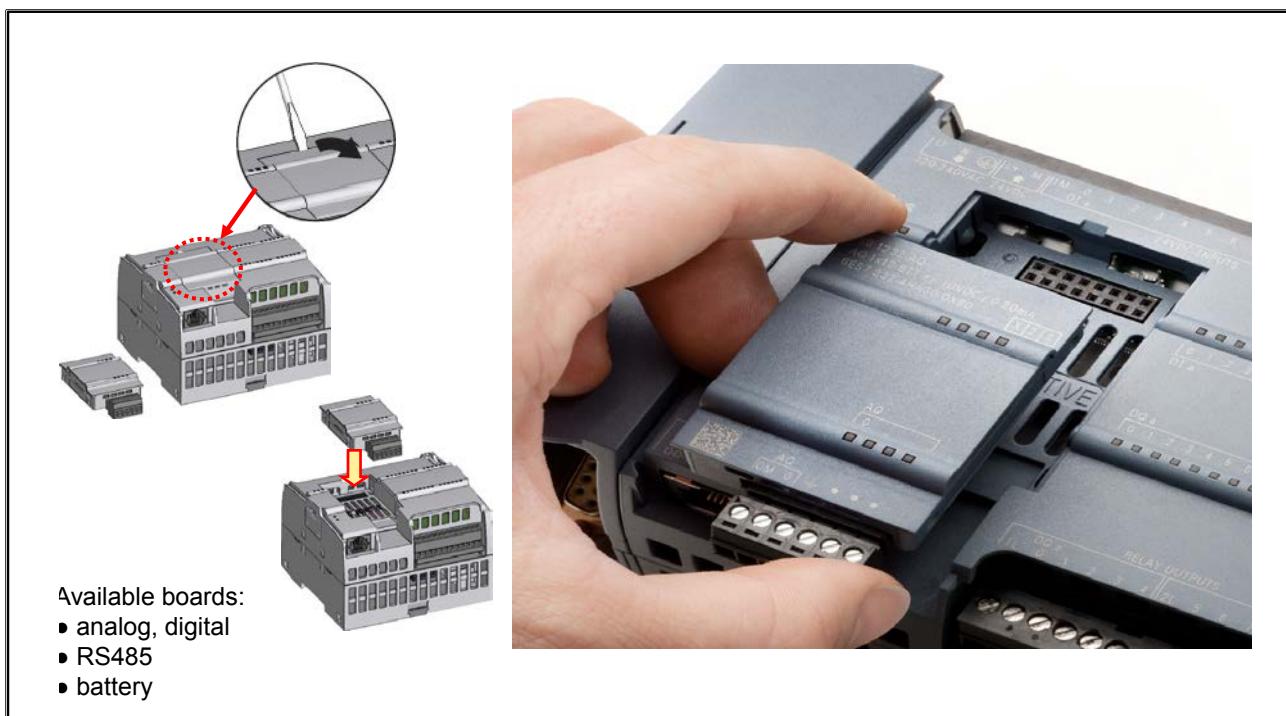
A horizontal or vertical mounting is possible.



Caution!

With a vertical mounting, the maximum allowed ambient temperature is 10 °C lower.

2.5.3. SIMATIC S7-1200: Signal, Communication or Battery Board



Application

These boards are used for application-specific adaptation of the CPU to the requirements of the plant. The physical size of the CPU remains unchanged.

Signal Board (SB)

- Digital signal board
 - only inputs
 - only outputs
 - inputs and outputs
- Analog signal board
 - only inputs
 - only outputs

Communication Board (CB)

- RS485 interface

Battery Board (BB)

A battery board (housing for CR1025 battery) ensures the long-term battery backup (buffering) of the real-time clock.

- Buffering time without battery board typically 20 days / minimum 12 days at 40°C
- Buffering time with battery board approximately 1 year

2.6. SIMATIC S7-1500: Modular Controller for the Mid to Upper Performance Range



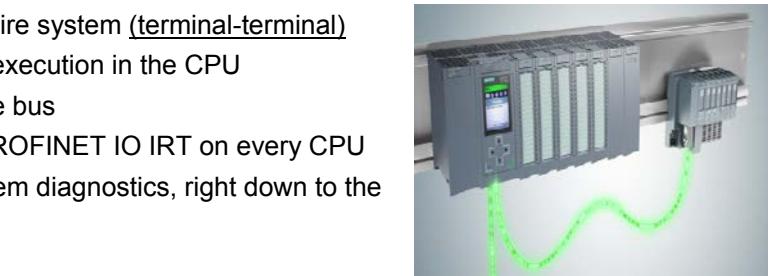
Highlights of the SIMATIC S7-1500 System

- Highest performance of the entire system (terminal-terminal)
 - High performance program execution in the CPU
 - High performance backplane bus
 - PROFINET interface with PROFINET IO IRT on every CPU
 - Automatically activated system diagnostics, right down to the IO channel

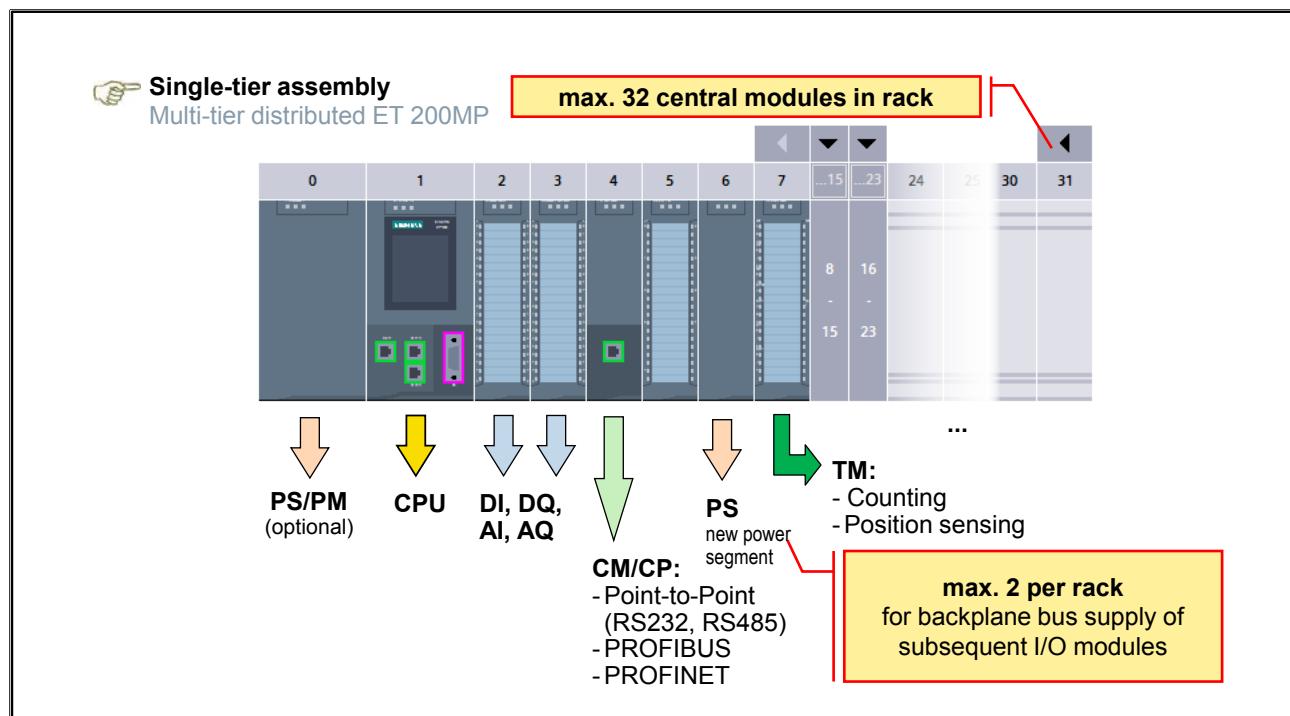
- Trace for all CPU tags

- CPU - Display for:
 - Access to MLFB, FW version and serial number
 - Commissioning (e.g. Setting the IP address, station name)
 - Backup/Restore
 - Diagnostics

- Simplified programming through user-friendly new instructions in LAD/FBD/STL



2.6.1. SIMATIC S7-1500: Modules



Slot Rules

- 1x PS/PM Slot 0
- 1x CPU in Slot 1
- As of Slot 2 any

Signal Modules

- Digital input modules: 24VDC, 230VAC
- Digital output modules: 24VDC, 230VAC
- Analog input modules: voltage, current, resistance, thermocouple
- Analog output modules: voltage, current

Communication Modules (CP - Communication Processor, CM - Communication Module)

- Point-to-Point connection
- PROFIBUS
- PROFINET



CPs and CMs are both communication modules.

CPs have, as a rule, somewhat more functionality than CMs (e.g. own web server, firewall, or the like).

Technology Modules (TM - Technology Module)

- Counting
- Position sensing

Power Supply

I/O modules in the central rack of the S7-1500 require a system power supply via the backplane bus (communication connection to the CPU) and a load power supply (input or output circuits for sensors/encoders and actuators).

- **PM - Power Module → Load Power Supply**
supplies modules with 24VDC for input and output circuits as well as sensors/encoders and actuators
 - ☞ If the CPU is supplied 24V via a load power supply (PM), it supplies the system power supply of 12W for the first inserted I/O modules.
- **PS - Power System → System Power Supply**
supplies S7-1500 modules in the central rack via the backplane bus
 - ☞ Each CPU offers a system power supply of 12W for the first inserted I/O modules. Depending on the I/O modules used, further power segments have to be set up, as required.
 - ☞ A system power supply (PS) can also supply the load circuit for 24VDC modules in addition to the CPU.

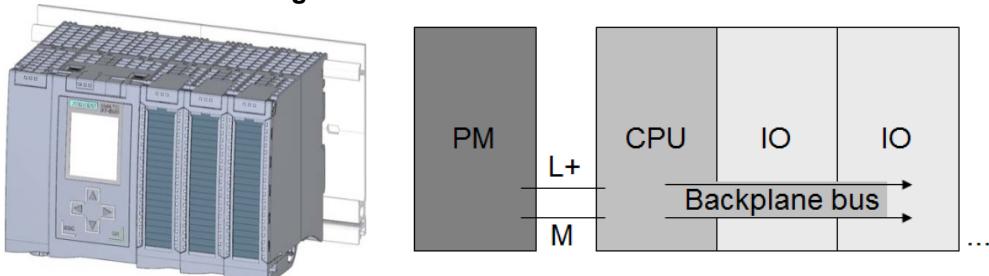
Power Supply and Power Segments of the I/O Modules

It is necessary to set up power segments in the central rack for larger configurations or configurations with greater I/O module power requirements (as a rule, when using CP, CM, TM).

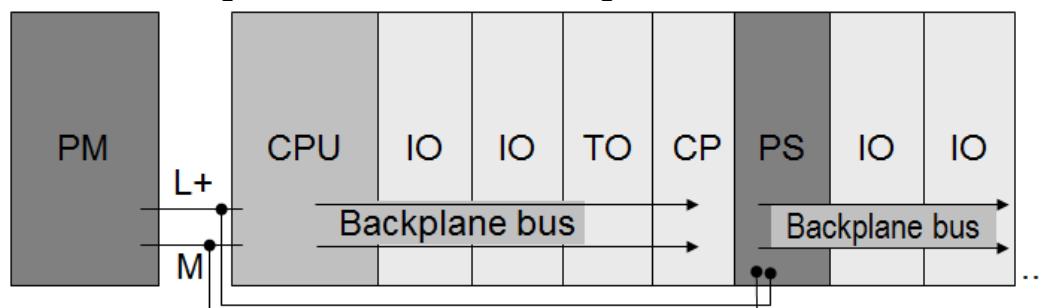
A maximum of 3 power segments can be set up per rack (1xCPU segment plus 2 more).

If the configuration includes additional power segments, additional system power supply modules (PS) are inserted to the right next to the CPU. The CPU continues to control all modules of the rack. Only the system power supply of the I/O modules is subdivided here.

Example of a Small S7-1500 Configuration



Example of an S7-1500 Configuration with a 2nd. Power Segment



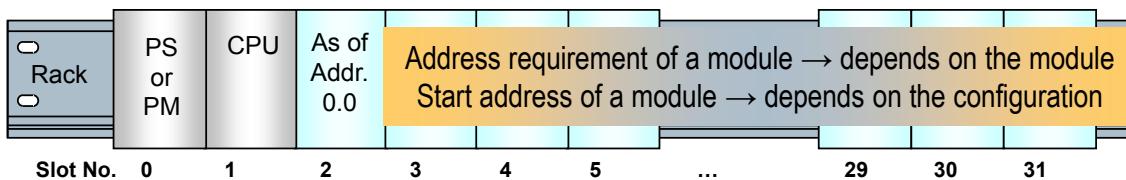
Interface Modules for Expansion Rack

There are no plans for a central multi-tier assembly. An expansion can be realized using the distributed ET 200MP I/O system.

2.6.2. I/O Addressing of the S7-1500

Default addresses of the I/O modules:

- Address assignment does not depend on the slot
- Begin as of I/O address = 0
- Addresses are assigned consecutively in the order in which the manual configuration of the I/O modules occurs

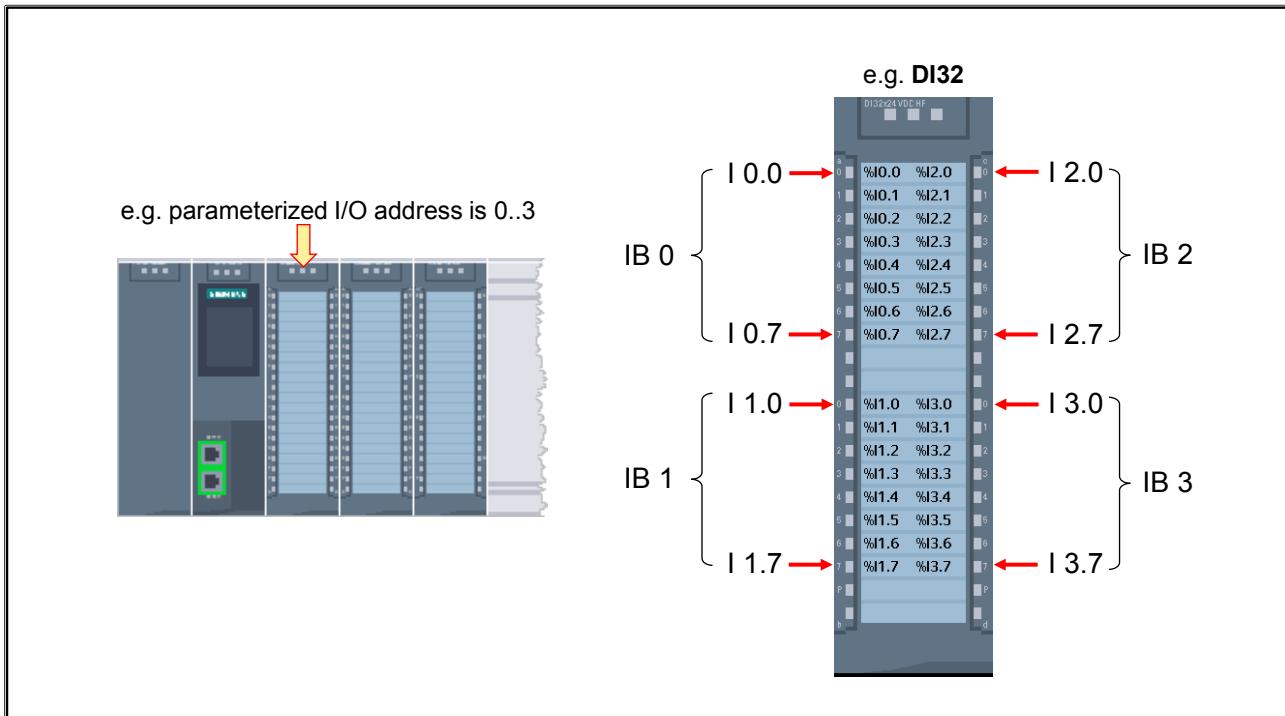


In order to be used by the user program, each I/O module is assigned "Module Address(es)" for the status/modify data area of its channels. By reading and writing the address areas, the user program can work with the process peripherals which are connected to the modules.

Default Addressing

During the device configuration, the address assignment for an added module is suggested as a **consecutive allocation of the I/O address area** as of byte-address 0. This address can be changed during the configuration of the hardware.

2.6.2.1. Channel Addressing of Digital S7-1500 Modules



Each digital channel has an assigned Status LED next to the terminal connection.

The address of the binary process signal results from...

- the terminal used
- the I/O address of the module

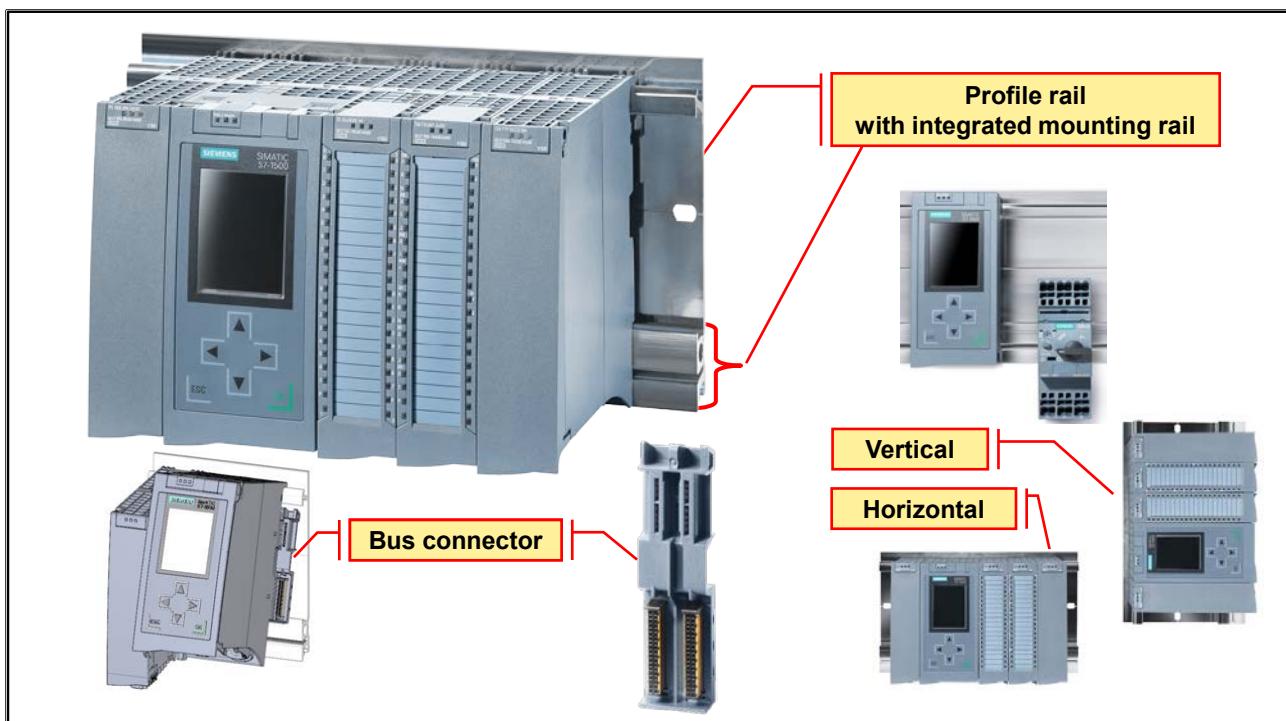
Address Assignment of the Byte Addresses

From top to bottom and (for > 16 channels) from left to right

Address Assignment of the Bit Addresses

From top to bottom

2.6.3. SIMATIC S7-1500: Installation and Mounting Positions



Installation

The modules are mounted on an S7-1500 profile rail.

Bus Connector

If the installation is made on the profile rail, the modules are then connected with the U-connector.

The U-connector establishes the mechanical and electrical connection between the modules and is included with every I/O module.

Mounting Positions

A horizontal or vertical mounting is possible.



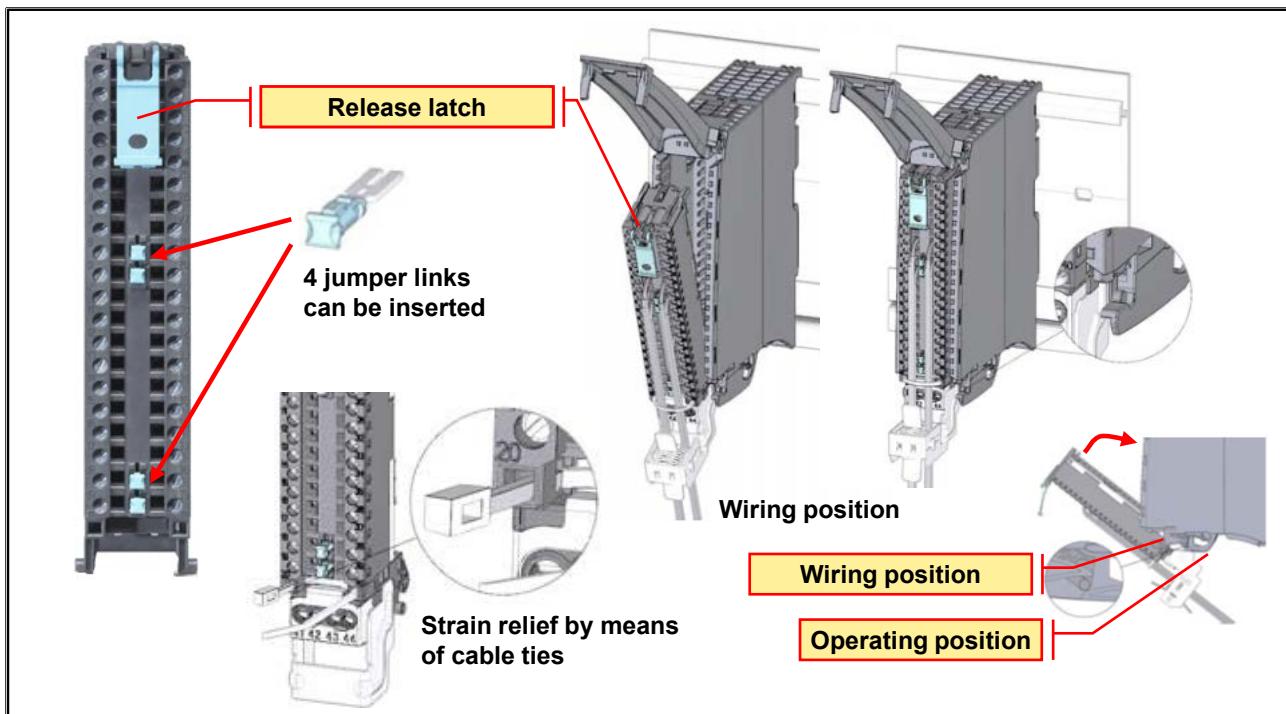
Caution!

With a vertical mounting, the maximum allowed ambient temperature is 20 °C lower (ambient temperature 0 to 40 °C).

Integrated Mounting Rail

For the S7-1500 CPU, there is a profile rail on which components can also be mounted according to EN 60715. With that, even terminals, miniature circuit breakers, small contactors or similar components can be mounted in addition to the S7-1500.

2.6.4. SIMATIC S7-1500: Connection Technology / Front Connector



Properties of the Front Connector

- In each case 40 terminals
 - Clamping techniques:
 - Screw-type terminal
 - SIMATIC TOP connect
- System wiring for the connection of sensors/encoders and actuators
→ S7-1500 front connector wired with 20 or 40 single conductors (prefabricated)



Prewiring Position

The front connector latches up in the front cover. In this position, the front connector still juts out of the module, but the front connector and the module are not yet electrically connected.

Jumper Links

The jumper links can be inserted in the front connector in four places for easier set up of load groups. That is, looping the supply voltage to several potential groups.

Only one connection from left to right exists!

Automatic Coding of the I/O Modules

This enables a faster and safer exchange of the front connector.

Two Front Door Latching Positions

At the bottom of the front cover there are 2 different latching positions for different space requirements of the conductor bundle.

- Cable storage space that grows with the need (AWG cabling)
American Wire Gauge (AWG) is an American standard measure for copper wires which defines the wire strength and the allowed damping, whereby a lower AWG value represents a thicker wire. By swinging up the release latch, you can pull the front connector and remove it from the module.

2.6.5. SIMATIC S7-1500: CPU-Display → Overview

- Every S7-1500 CPU is delivered with a Display
- Two sizes depending on the CPU type
 - 1.36" up to CPU1513
 - 2.4" as of CPU1516
- Has its own MLFB
→ can be ordered as replacement part
- CPU can be operated without the Display
(different front cover)
- Removal and insertion possible during running operation
→ CPU stays in RUN
- Multi-lingual display (menu)
- Message/Alarm texts and comments can be loaded in 3 languages
- Language can be switched during running operation
- Alarm acknowledgement
- Backup/Restore
- Format SMC



The available Display languages are the available user-interface languages of STEP 7.

Operation



Selecting the main menu item



Shifting the selection of the submenu item



Choosing the selected submenu item



Go back one menu item



After choosing the menu item
Edit setting



Accept change



Discard change

2.6.6. SIMATIC S7-1500: CPU-Display → Menu and Colors

Main menu items and their meaning:



Overview (Info on: CPU, program protection, memory card, fail-safe (operation) ...)



Diagnostics (alarms, diagnostic buffer, message display, watch tables, cycle time, current memory)



Settings (addresses, date & time, operating mode, CPU Reset, unlock Display, Backup, FW update ...)



Modules (status, MLFBs, version, information,... of individual modules)



Display (setting: brightness, language Display/diagnostic message, standby,..., Display infos: MLFB, version, ...)

Colors of status information and their meaning:

green RUN of CPU,

yellow STOP or CPU HOLD

red Error

white Connection setup or connection to CPU lost



Additional Symbols in the Status Information



Password is configured but not entered



Password is configured and entered



An Alarm exists



A Force job is active on the CPU



F-ability activated. Safety operation active (for fail-safe CPUs)
With deactivated safety operation, the icon is greyed out.

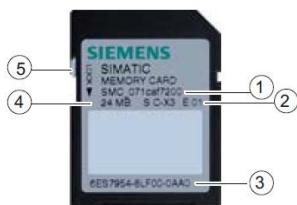


Fail-safe CPU (for fail-safe CPU)

S7-1500 Display Simulator

Under the link "TIA Portal Information Center" > First steps > Getting Started > SIMATIC S7-1500 and STEP 7 - Getting Started > Displays of the SIMATIC S7-1500 controller family" you can test the S7-1500 display simulator as an online version or you can download it and test it offline.

2.7. SIMATIC S7-1200/1500: Memory Card(s)



- ① Serial number of the SMC card
- ② Product version
- ③ Order number
- ④ Card size
- ⑤ Slide switch for write-protect (must not be write-protected)

Written with:

- Commercially available SD card reader
- Field PG

SIMATIC Memory Card in the S7-1200:

- External load memory
- Distribution of programs
- Firmware update
- Documentation
- Memory Card Binding
- Unlinked DBs
- Module exchange without PG

SIMATIC Memory Card in the S7-1500:

- Load memory
- Firmware update
- Documentation
- Memory Card Binding
- Unlinked DBs
- Archiving of data
- Module exchange without PG

Memory Card Binding – Copy Protection

The executability of the program can be bound to the serial number of the card.

Load Memory

- S7-1500
This has no integrated load memory and therefore it is imperative that a card is inserted.
- S7-1200
This has an integrated load memory. Here, an inserted memory card can replace (expand) the integrated load memory or the card can be used for program updates (distribution of programs).

Distribution of Programs ← only S7-1200

The use as Transfer card (card mode = "Transfer") is only supported by the S7-1200. Here, a program can be downloaded into the CPU without a PG if a card is inserted.

Archiving of Data ← only S7-1500

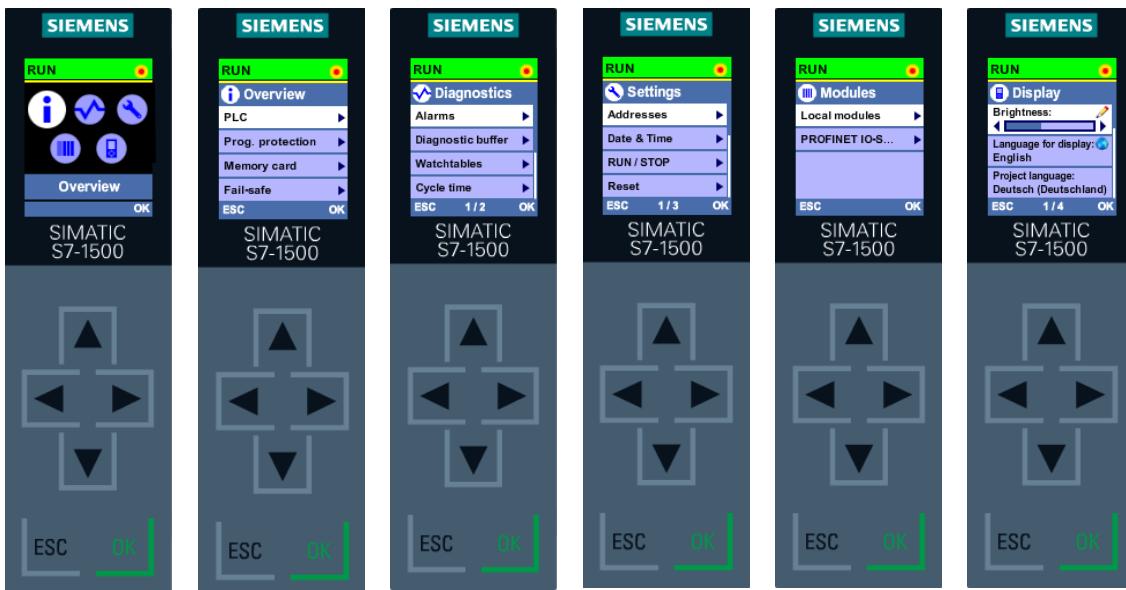
It is possible to archive process values on the card.



The use of this functionality influences the operating life of the Memory Card

2.8. Exercise 1: Display

Familiarize yourself with the Display!



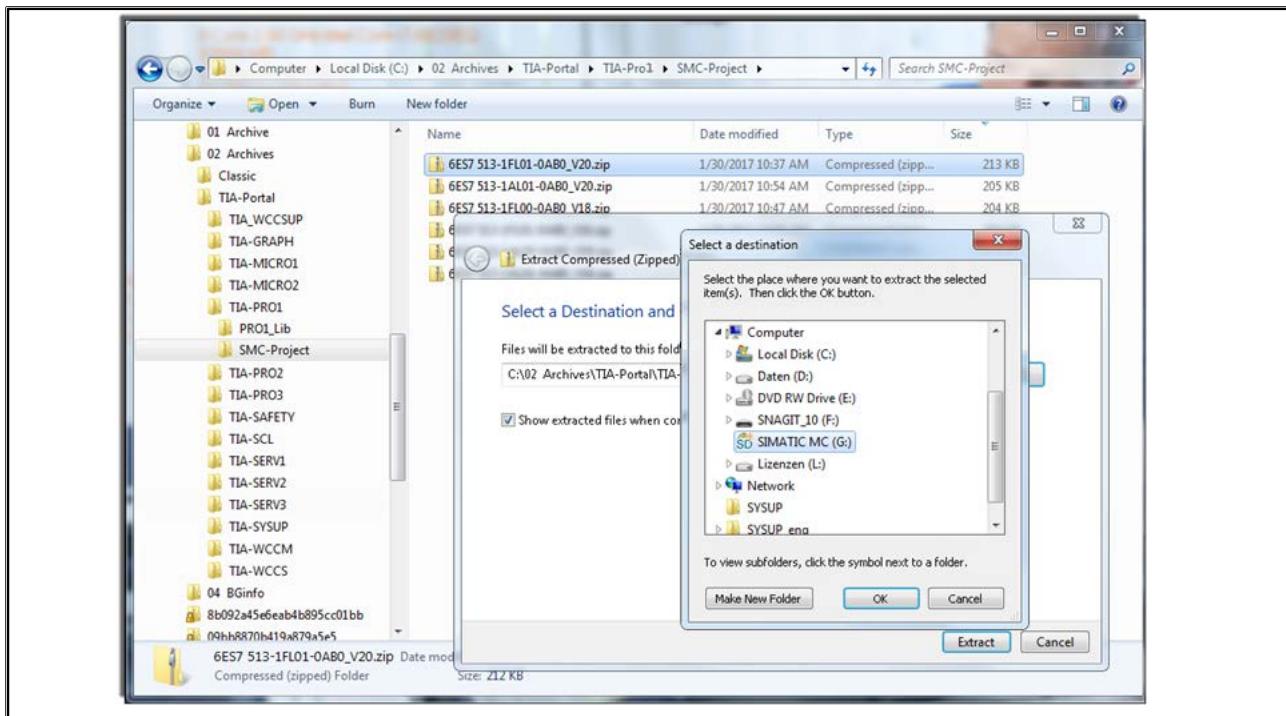
What to Do:

1. Change the language and the brightness on the Display.
2. Read-out the memory card type and how much memory still exists.
3. Take a look at the diagnostic buffer.
4. Make a note of the MLFB (order number) and the Firmware of the CPU and all central modules.

Module	MLFB (order number)	Firmware
CPU		
DI		
DO		
AI		

5. Change the time (of day) of the CPU, the IP address of the Interface X1, and format the memory card.

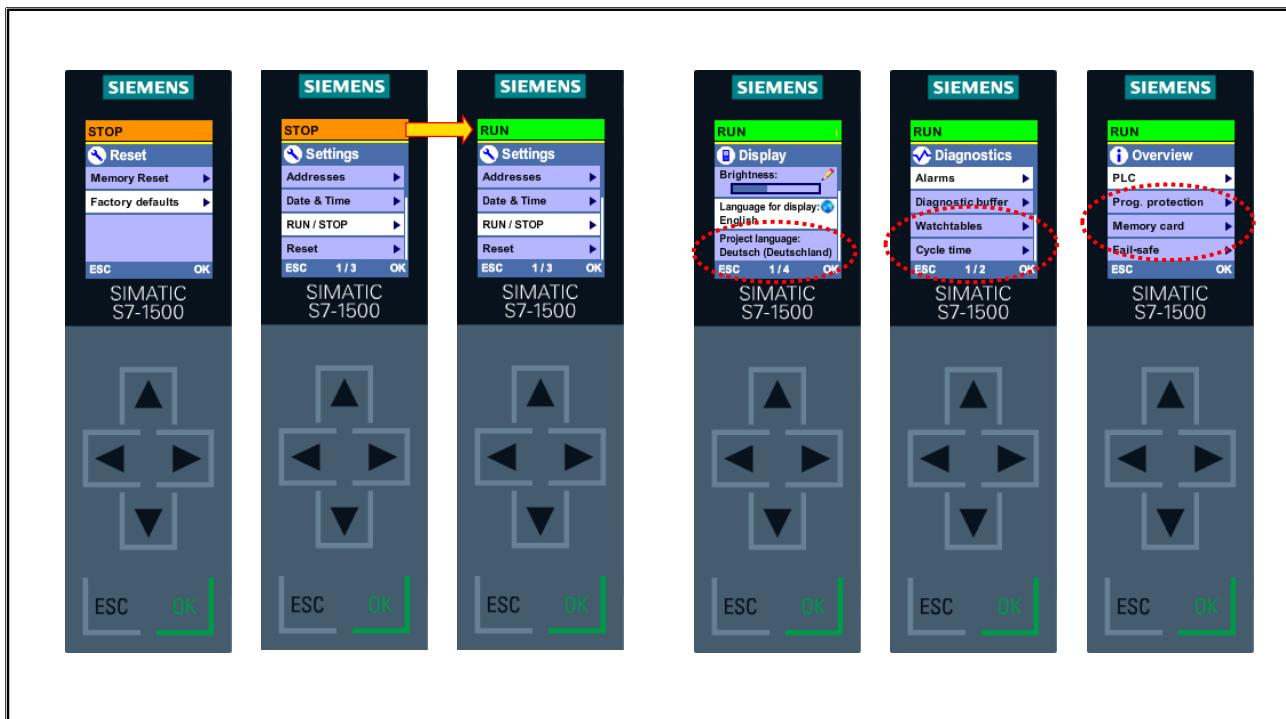
2.9. Exercise 2: Loading the Program onto the SMC



What to Do:

1. Set the CPU to STOP using the Display.
2. Remove the SIMATIC Memory Card.
3. Insert the SMC in the SD card reader of the programming device (PC).
4. Erase the card using Windows Explorer.
5. Open the folder "C:\02_Archives\TIA_Portal\TIA-PRO1\SMC-Project".
6. Extract (unzip) the ZIP file (Name according to the MLFB and the firmware of the CPU) onto the SIMATIC Memory Card.

2.9.1. Exercise 3: Diagnostics and Program Test



What to Do:

1. Insert the SIMATIC Memory Card in the CPU and wait until the MAIN-LED no longer flashes.
2. Reset the CPU to factory defaults using the Display,
(delete all data since there could still be an old program on the CPU)
3. Switch the CPU to Run.
4. Set the language.
5. Diagnostics using the Display: Check the Watchtables, Cycle time and Memory.
6. Check whether the broadband cable of your conveyor model is connected to the "S7-1500 DI/DO" socket on the back of your training case.
7. Switch on your conveyor model (green button).
8. Carry out a program test.

Function Description

The distribution conveyor is used to transport parts and can be operated in two different operating modes. (Manual and Automatic)

These can be set using the control panel buttons "Operation ON" and "Operation OFF".

Manual Mode "Operation" = off

Now, the conveyor motor can be jogged to the right "Jog right" and to the left "Jog left".

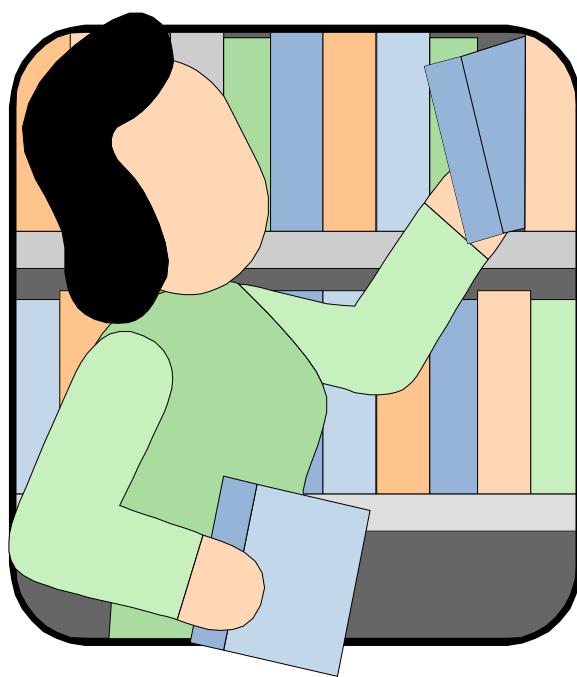
Automatic Mode "Operation" = on

On the conveyor model, parts can be transported from Bay 1, 2 or 3 to the right, right up to the light barrier, using the appropriate button at the respective Bay. Only one Bay may be occupied (at a time).

The indicator lights at Bays 1, 2 and 3 show...

- A continuous light at the bay at which the associated sensor detects a part, however only as long as the conveyor has not yet been started.
- A 2Hz flashing light as long as the conveyor motor is running.

2.10. Additional Information



2.10.1. ET 200SP and ET 200pro Controller



Further Information under the Link:

TIA Portal Information Center > Product information > Controllers > SIMATIC controllers in general > Distributed Controllers

2.10.2. Software Controller

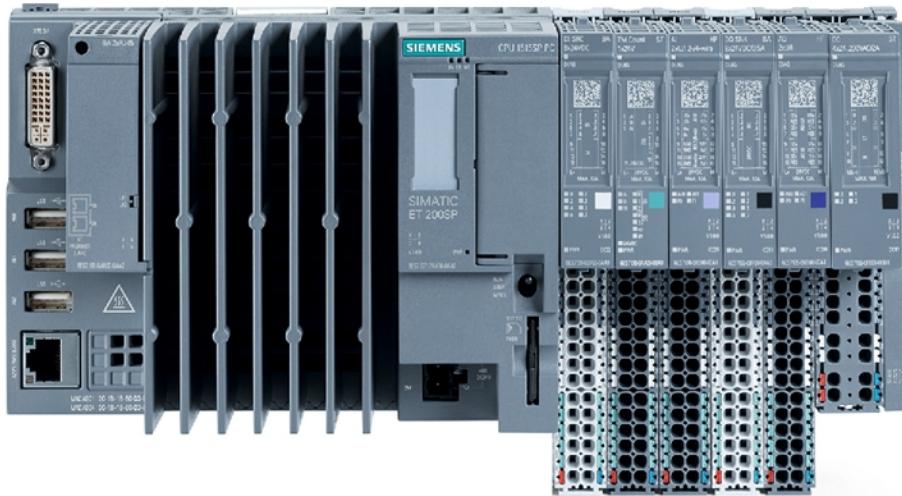
- Use with industry-suitable SIMATIC IPCs
- Runs completely independently of the Windows system (even with restart or failure of Windows)
- Flexible controller for special-purpose machines with high performance and functional requirements
- Integration of user-specific functions via open interfaces (for example C++ / Matlab)



Further Information under the Link:

TIA Portal Information Center > Product information > PC-Based Automation > SIMATIC Software Controller

2.10.3. ET 200SP Open Controller "All in one"



- - Controller with central, modular I/Os
- - Visualization and Windows applications
- - PC interfaces for monitor, mouse and keyboard
- - Gigabit Ethernet

Further Information under the Link:

TIA Portal Information Center > First steps > Getting Started > SIMATIC Open Controller - Getting Started

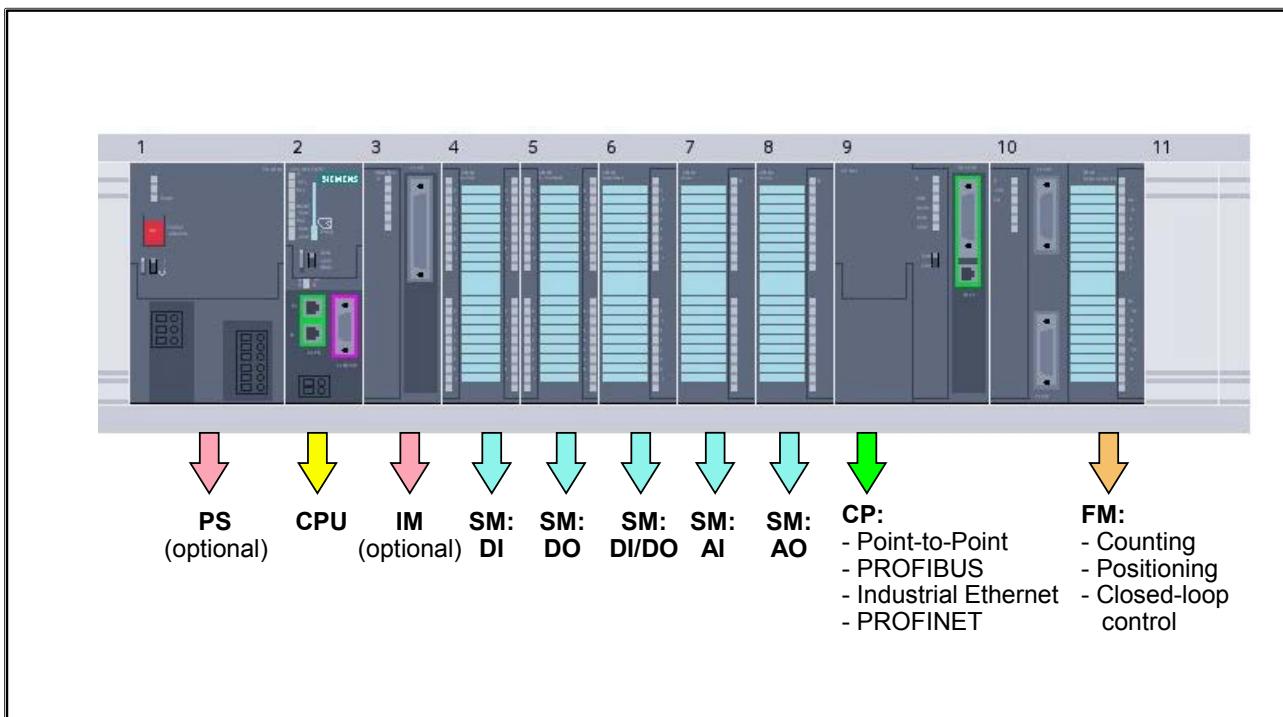
2.10.4. SIMATIC S7-300: Modular Automation System



Features

- Modular compact control system for the lower and middle performance range
- Scaled CPU range
- Extensive range of modules
- Can be expanded to up to 32 modules
- Backplane bus integrated in the modules
- Can be networked with
 - Multipoint Interface (MPI),
 - PROFIBUS or
 - Industrial Ethernet or
 - PROFINET
- Central PG/PC connection with access to all modules
- No slot rules for I/O modules.

2.10.4.1. SIMATIC S7-300: Modules



Signal Modules (SM)

- Digital input modules: 24VDC, 120/230V AC
- Digital output modules: 24VDC, Relay
- Analog input modules: Voltage, Current, Resistance, Thermocouple
- Analog output modules: Voltage, Current

Interface Modules (IM)

The IM360/IM361 and IM365 make multi-tier configurations possible.
The interface modules loop the bus from one tier to the next.

Dummy Modules (DM)

The DM 370 dummy module reserves a slot for a signal module whose parameters have not yet been assigned. A dummy module can also be used to reserve a slot for installation of an interface module at a later date.

Function Modules (FM)

- Counting
- Positioning
- Closed-loop control.

Communication Processors (CP)

- Point-to-Point connections
- PROFIBUS
- Industrial Ethernet
- PROFINET.

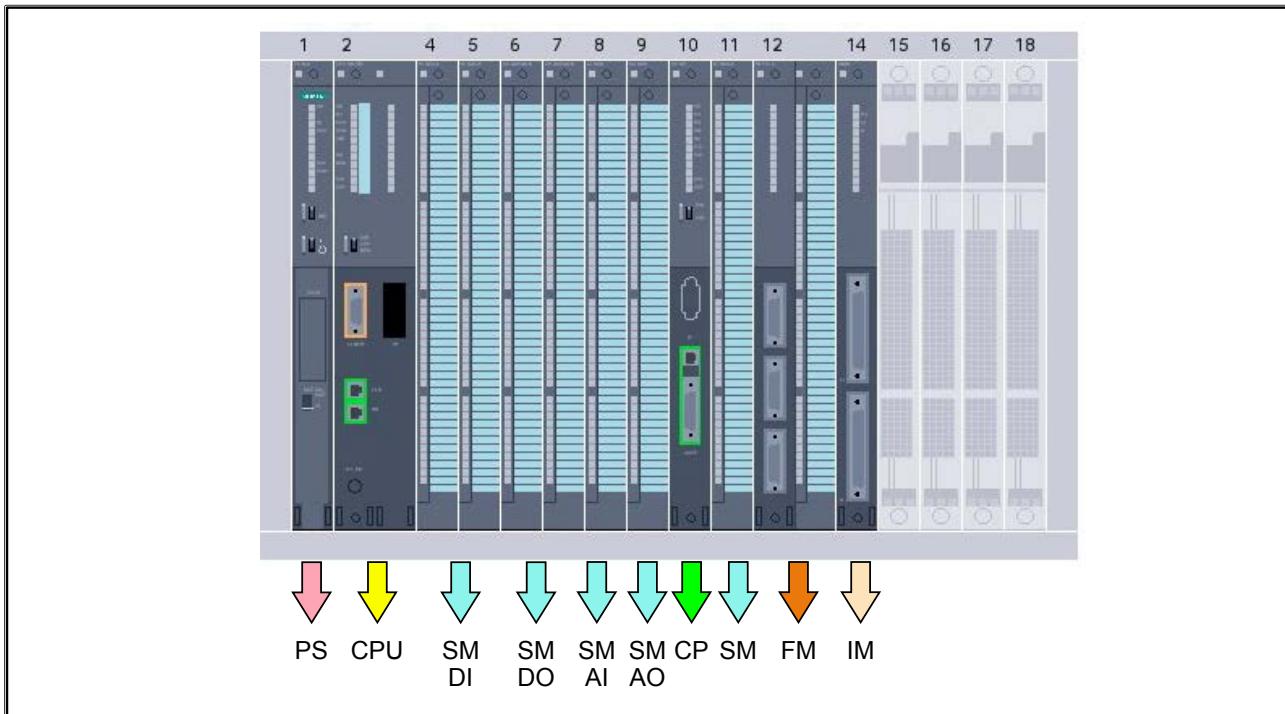
2.10.5. SIMATIC S7-400: Modular Automation System



Features

- The power PLC for the mid to upper performance range,
- Scaled CPU range
- Extensive range of modules
- Can be expanded to over 300 modules,
- Backplane bus integrated in the mounting rack
- can be networked with
Multipoint Interface (MPI),
PROFIBUS or
Industrial Ethernet or
PROFINET
- Central PG/PC connection with access to all modules,
- Only a few slot rules,
- Multicomputing (up to four CPUs can be used in the central rack).

2.10.5.1. SIMATIC S7-400: Modules



Signal Modules (SM)

- Digital input modules: 24VDC, 120/230VAC
- Digital output modules: 24VDC, Relay
- Analog input modules: Voltage, Current, Resistance, Thermocouple
- Analog output modules: Voltage, Current.

Interface Modules (IM)

The IM460, IM461, IM463, IM467 interface modules provide the connection between various racks:

- UR1 (Universal Rack) with up to 18 modules
- UR2 (Universal Rack) with up to 9 modules
- ER1 (Extension Rack) with up to 18 modules
- ER2 (Extension Rack) with up to 9 modules.

Function Modules (FM)

- Counting
- Positioning
- Closed-loop control.

Communication Processors (CP)

- Point-to-Point connections
- PROFIBUS
- Industrial Ethernet
- PROFINET.

Contents

3

3.	Engineering Software TIA Portal.....	3-2
3.1.	Product Lifecycle Management.....	3-3
3.2.	Digital Enterprise Suite -> Answer to Industrie 4.0	3-4
3.3.	TIA Portal - Central Engineering Framework.....	3-5
3.4.	Scope of the Products.....	3-6
3.4.1.	STEP 7 Range of Products.....	3-7
3.4.1.1.	STEP 7 Licensing	3-45
3.4.2.	WinCC Range of Products.....	3-8
3.4.2.1.	WinCC Licensing	3-46
3.4.3.	Startdrive Range of Products and Licensing	3-9
3.5.	Automation License Manager	3-10
3.5.1.	Operating Systems for PC/PGs	3-11
3.5.2.	Virtualization (Released Software).....	3-12
3.5.3.	Side-by-Side Installation	3-13
3.6.	TIA Portal: Portal View and Project View.....	3-14
3.6.1.	Portal View	3-15
3.6.2.	Project View	3-16
3.6.3.	Menu Bar and Toolbar	3-17
3.6.4.	Project Tree (First Level)	3-18
3.6.4.1.	Project Tree (Second Level)	3-19
3.6.5.	Task Cards.....	3-20
3.6.6.	Inspector Window	3-21
3.7.	Uploading a Device as a New Station into the Project (1).....	3-22
3.7.1.	Uploading a Device as a New Station into the Project (2).....	3-23
3.8.	Exercise 1: Deleting Old Projects	3-24
3.8.1.	Exercise 2: Creating a Project	3-25
3.8.2.	Exercise 3: Uploading Devices as a New Station.....	3-26
3.9.	Window Arrangement	3-27
3.9.1.	Splitting and Arrangement of the Working Area	3-28
3.9.2.	Keeping the Editor Window in the Foreground (when Editor Space is Split)	3-29
3.9.3.	Save / Manage / Use Window Layouts.....	3-30
3.10.	Undo and Redo	3-31
3.11.	Saving a Project.....	3-32
3.11.1.	Archiving / Retrieving a Project.....	3-33
3.12.	TIA Portal - Settings: User Interface Language	3-34
3.12.1.	TIA Portal - Settings: Language, Storage Location, Layout	3-35
3.13.	Libraries	3-36
3.14.	Help	3-37
3.14.1.	Help (Information System).....	3-38
3.15.	Additional Information	3-39
3.15.1.	Keyboard Shortcuts of the TIA Portal	3-40
3.15.2.	Project Migration	3-41
3.15.2.1.	Migration of STEP 7 V5.x – Projects: Supported Hardware	3-42
3.15.3.	Installation with Record Function in the Setup.....	3-43
3.15.4.	Update Tool.....	3-44

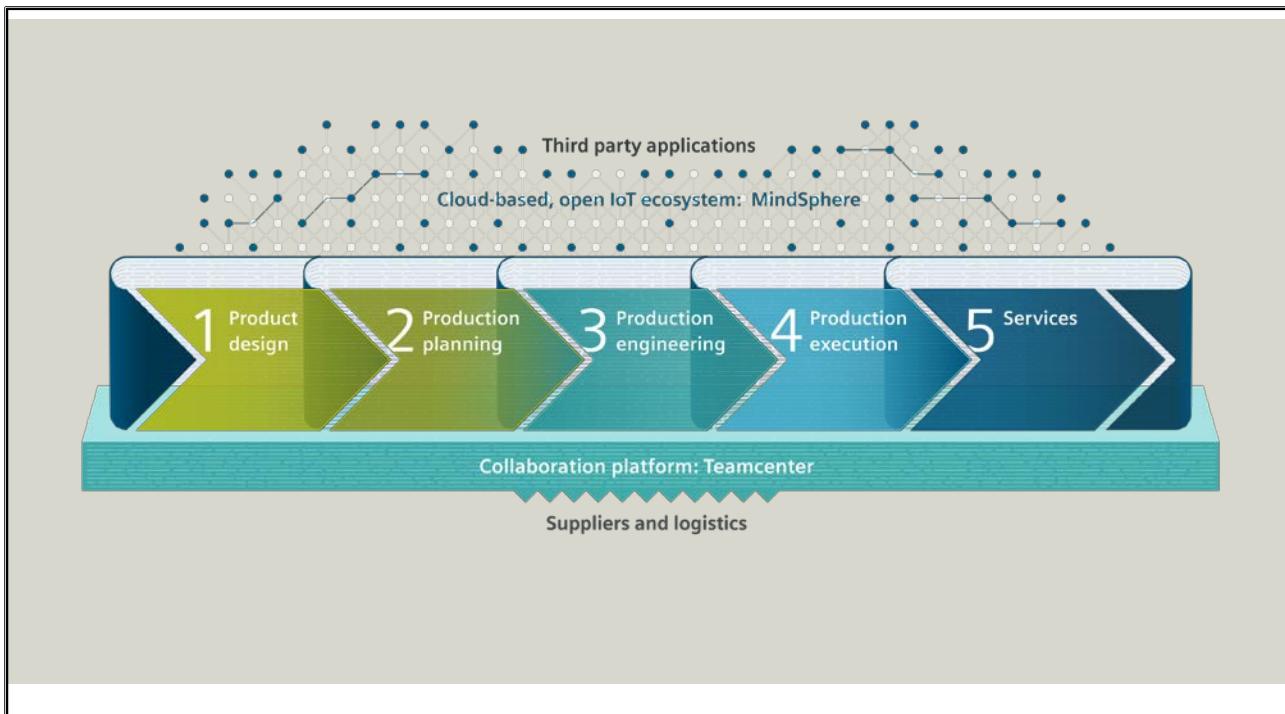
3. Engineering Software TIA Portal

At the end of the chapter the participant will ...



- ... have an overview of the scope of the engineering framework
- ... be familiar with the engineering products and their range of products
- ... be familiar with the operator interface of the framework
- ... be able to upload a program existing online

3.1. Product Lifecycle Management



Product- Lifecycle

Products and processes are becoming more and more complex. Siemens offers solutions for the Product Lifecycle Management (PLM) for all areas from concept development right up to the end of product life and which make it possible to successfully develop products, produce them and bring them to market.

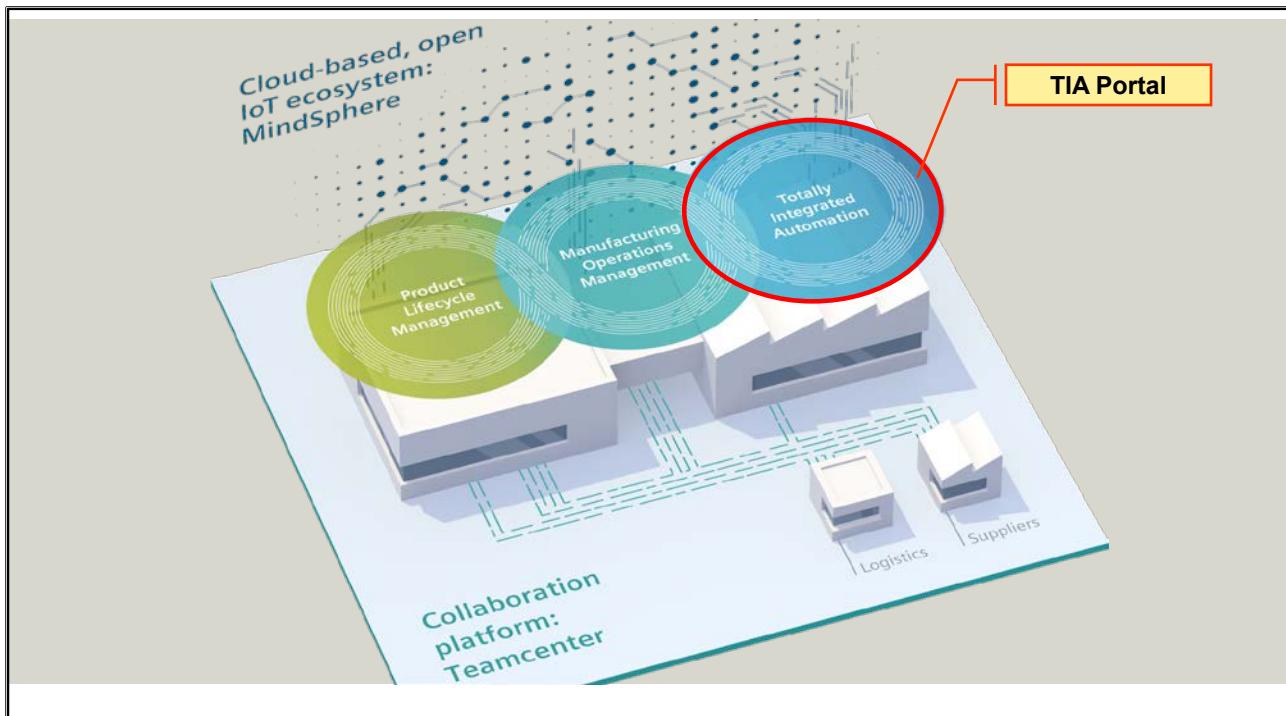
Siemens' holistic approach is to transform a traditional value-added chain into an integrated product and production lifecycle - starting with the product design, through the production planning, the manufacturing technology right up to production and services.

Only a fully digitalized business model can offer the strength and flexibility to accelerate processes and to optimize production processes.

This also includes a common data storage and data management system. With "Teamcenter", Siemens offers the industry-leading platform for interplay in all steps of the value-added chain – the "data backbone".

In the holistic value-added chain, the cloud-based open IoT-ecosystem "MindSphere" can be found.

3.2. Digital Enterprise Suite -> Answer to Industry 4.0



TIA Portal – your gateway to automation in the Digital Enterprise

The Totally Integrated Automation Portal (TIA Portal) provides you with unrestricted access to our complete range of digitalized automation services, from digital planning and integrated engineering to transparent operation.

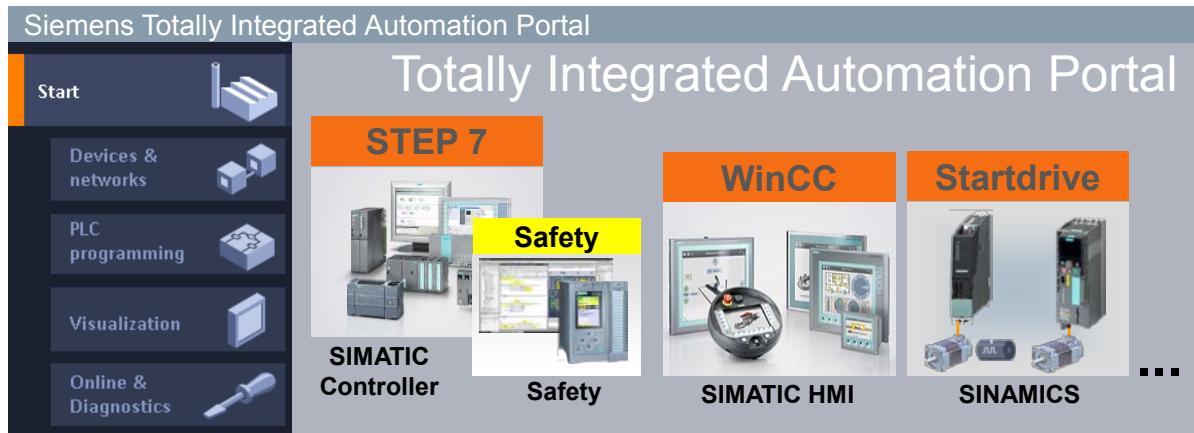
The new version shortens your time to market, for example by means of simulation tools, increases the productivity of your plant through additional diagnostics and energy management functions, and offers you broader flexibility by connecting to the management level. The new options benefit system integrators and machine builders as well as plant operators.

The TIA Portal is thus your perfect gateway to automation in the Digital Enterprise. As part of the Digital Enterprise Suite along with PLM and MES, it complements the comprehensive range of offerings from Siemens for companies on their path to Industry 4.0.

3.3. TIA Portal - Central Engineering Framework

The **Totally Integrated Automation Portal** is the **Engineering Framework** that is, it forms the framework for a consistent engineering for...

- ...programming automation systems
- ...visualizing processes



Stand-alone software packages are limited because they lack consistency and integration.

It takes a common working environment - that is, an **engineering framework** - to achieve full integration and consistency of individual products.

Advantages of a Central Engineering Framework

- Uniform operator control concept for all automation tasks with common services (for example configuration, communication, diagnostics)
- Automatic data and project consistency
- Powerful libraries covering all automation objects

The Most Important Engineering Products are:

- SIMATIC STEP 7
for PLC programming
- SIMATIC Safety
for programming fail-safe PLCs
- SIMATIC HMI
for configuring process visualization
- SINAMICS Startdrive
for parameterizing drives

3.4. Scope of the Products

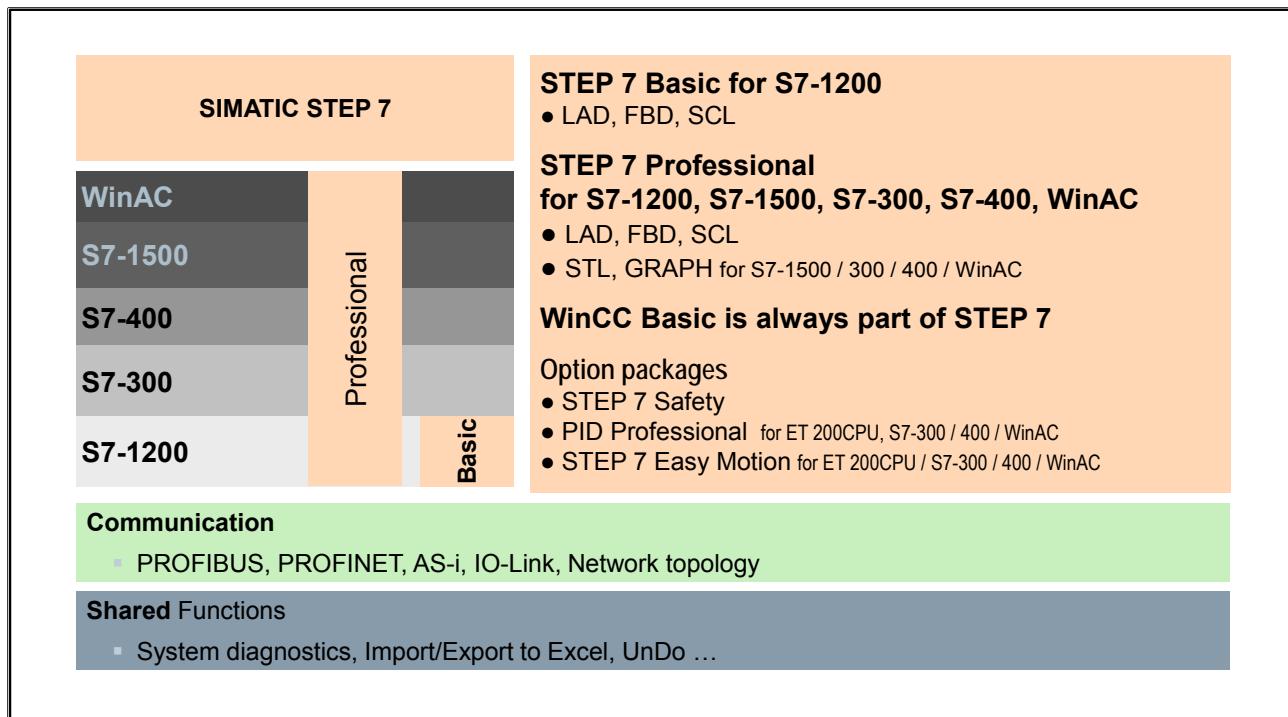
SIMATIC STEP 7		SIMATIC WinCC			SINAMICS Startdrive	
Configuration/Programming of controllers		Machine-level operator control/monitor. As well as process visualization			Integration of drives	
S7-1500	Professional				SCADA	S120 CU320 ≥V4.8
WinAC					PC Single station	G120 Uxxx-2 ≥V4.4
S7-400					Comfort Panels and x77 and Mobiles (without Micro Panels)	
S7-300			Comfort	Advanced		
S7-1200	Basic	Basic		Professional	Basic Panels	G110M CU240M ≥V4.6
Communication <ul style="list-style-type: none"> ▪ PROFIBUS, PROFINET, AS-i, IO-Link, Network topology 						
Shared Functions <ul style="list-style-type: none"> ▪ System diagnostics, Import/Export to Excel, UnDo ... 						

TIA Portal

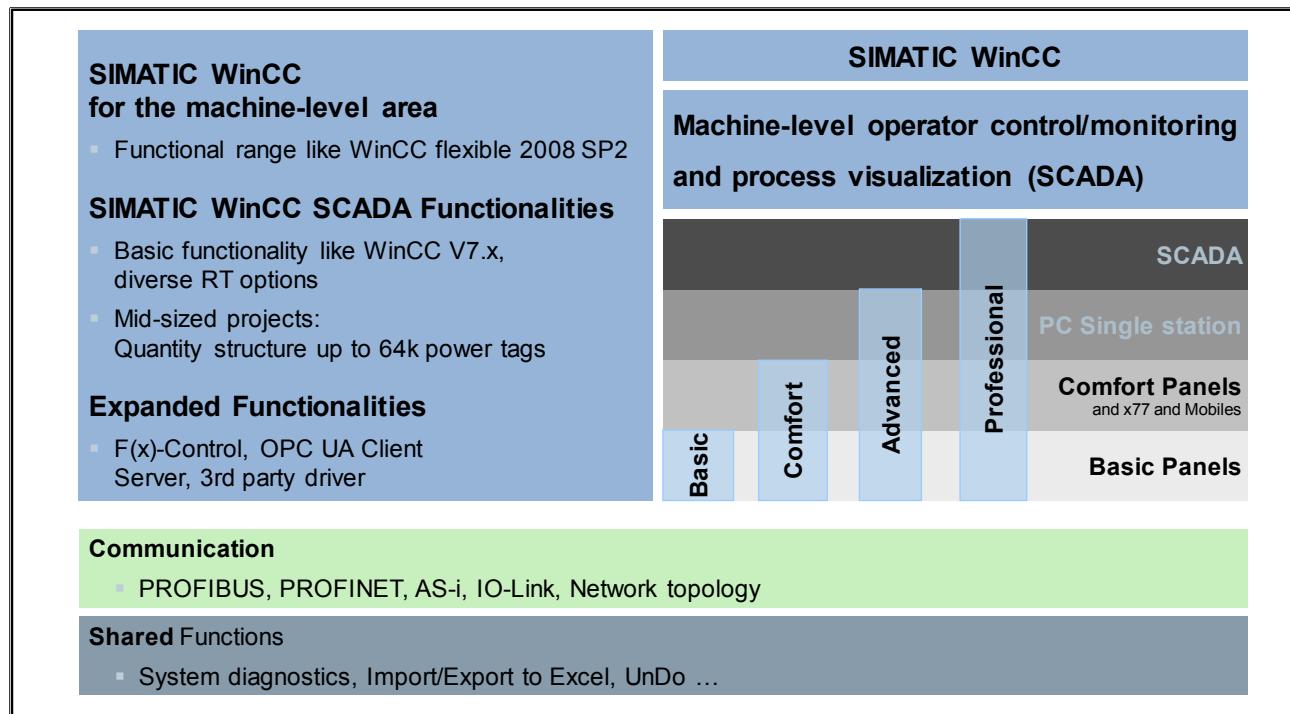
The Totally Integrated Automation Portal constitutes the working environment for an integrated engineering with SIMATIC STEP 7 and SIMATIC WinCC.

- Central engineering framework
- Automatic data and project consistency
- Uniform operator control concept for all automation tasks
- Powerful libraries covering all automation objects

3.4.1. STEP 7 Range of Products



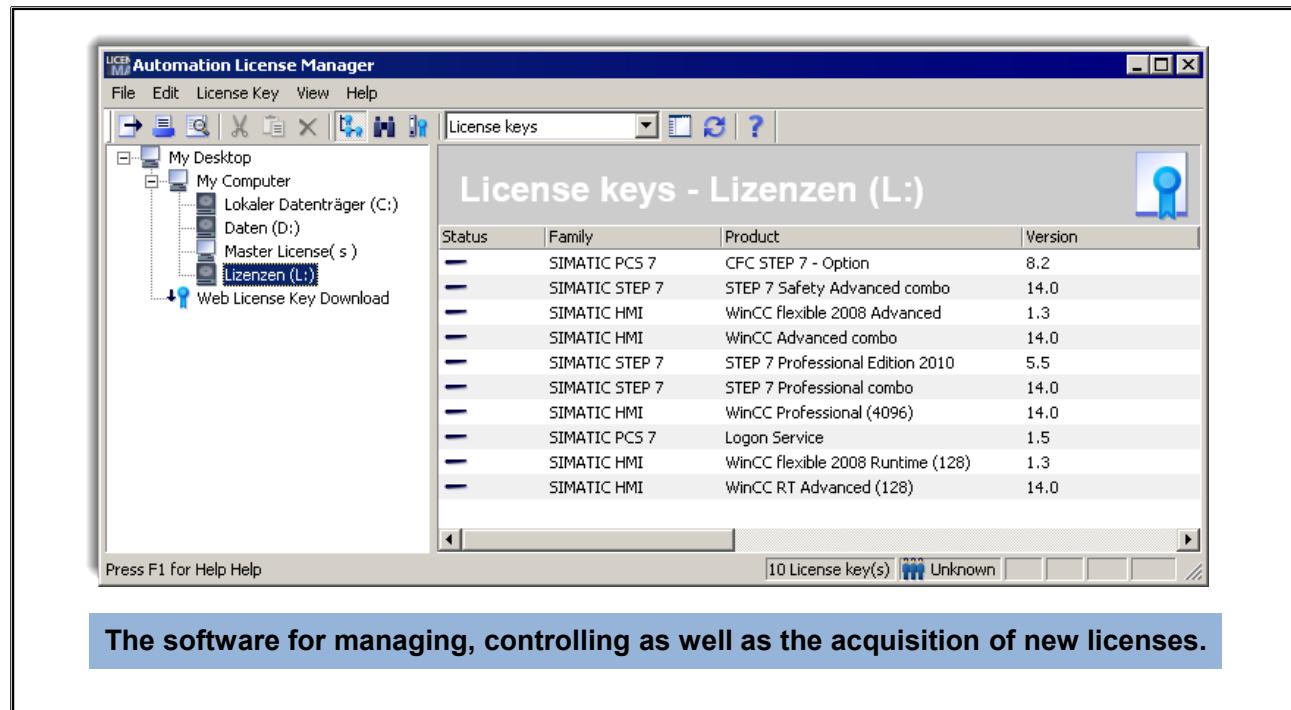
3.4.2. WinCC Range of Products



3.4.3. Startdrive Range of Products and Licensing

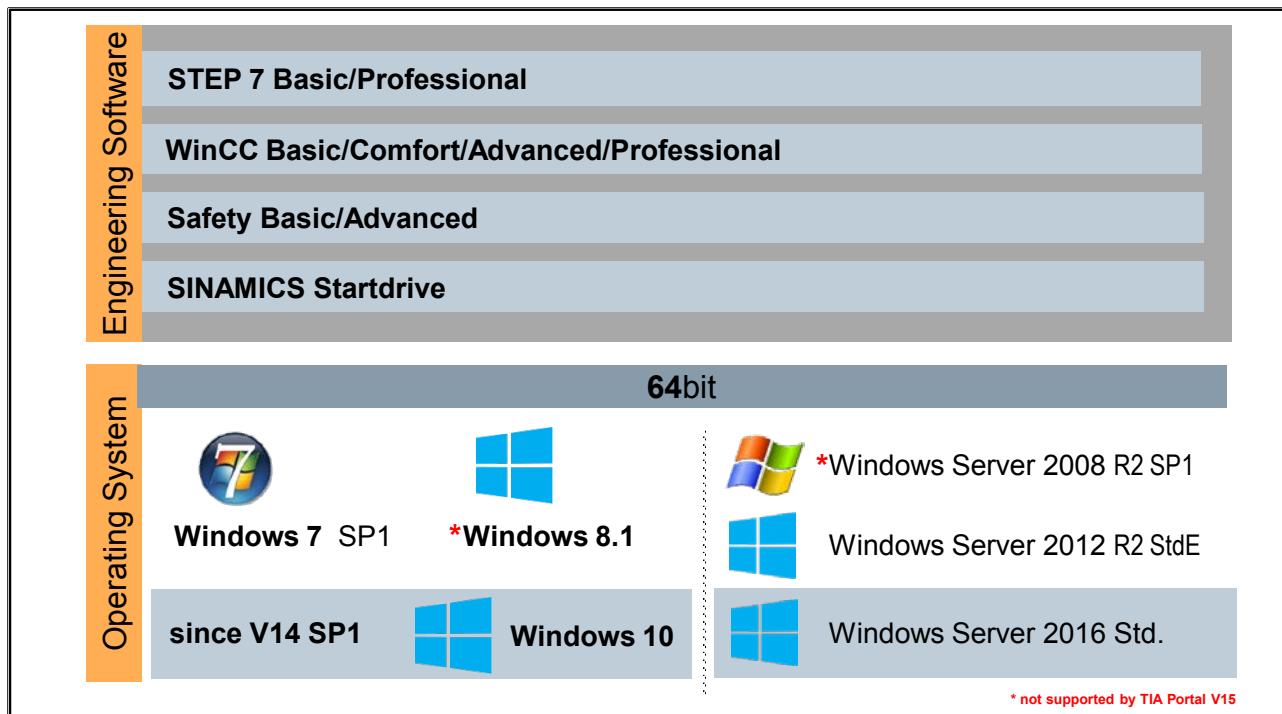
SINAMICS Startdrive		Functions:
Integration of drives		<ul style="list-style-type: none">• Parameterize drive• Commission drive• Test drive
Startdrive	S120 CU320 ≥V4.8	Supported product lines :
	G120 CUxxx and CUxxxX-2 ≥V4.4	<ul style="list-style-type: none">• SINAMICS S120• SINAMICS G120-series• SINAMICS G110M
	G110M CU240M ≥V4.6	
Communication		<ul style="list-style-type: none">▪ PROFIBUS, PROFINET, AS-i, IO-Link, Network topology
Shared Functions		<ul style="list-style-type: none">▪ System diagnostics, Import/Export to Excel, UnDo ...
	No additional license required	  STEP 7 Professional 

3.5. Automation License Manager



The Automation License Manager is part of the SIMATIC Software Installation and organizes the licensing of the SIMATIC software.

3.5.1. Operating Systems for PC/PGs



3.5.2. Virtualization (Released Software)

Released Virtualization Software	Released TIA Portal V14 Versions
VMware Player 12.5.5	✓ STEP 7 Basic/Professional V15
VMware Workstation 12.5.5	✓ WinCC Basic/Comfort/Advanced V15 WinCC Professional V15
VMware vSphere Hypervisor (ESXi) 6.5	✓ SINAMICS Startdrive V15 STEP 7 Safety Basic/Advanced V15
Microsoft Hyper-V Server 2016	✓ WinCC Runtime Advanced V15
Guest Operating Systems (virtual machine)	
<ul style="list-style-type: none">- Windows 7 SP1 64-Bit (Prof. / Ultimate / Enterprise)- Windows 8.1 64-Bit (Professional / Enterprise)- Windows 10 64-Bit (Professional / Enterprise)- Windows Server 2008 RS2 SP1- Windows Server 2012 RS and Windows Server 2016	

3.5.3. Side-by-Side Installation



TIA Portal V15, V14 and V13 can be installed on one and the same PG/PC

V14 and V15 requires 64bit system

The **Compatibility Tool** is available as an aid for checking parallel installations, see:

TIA Portal Information Center >
Tools & Apps > Configurators

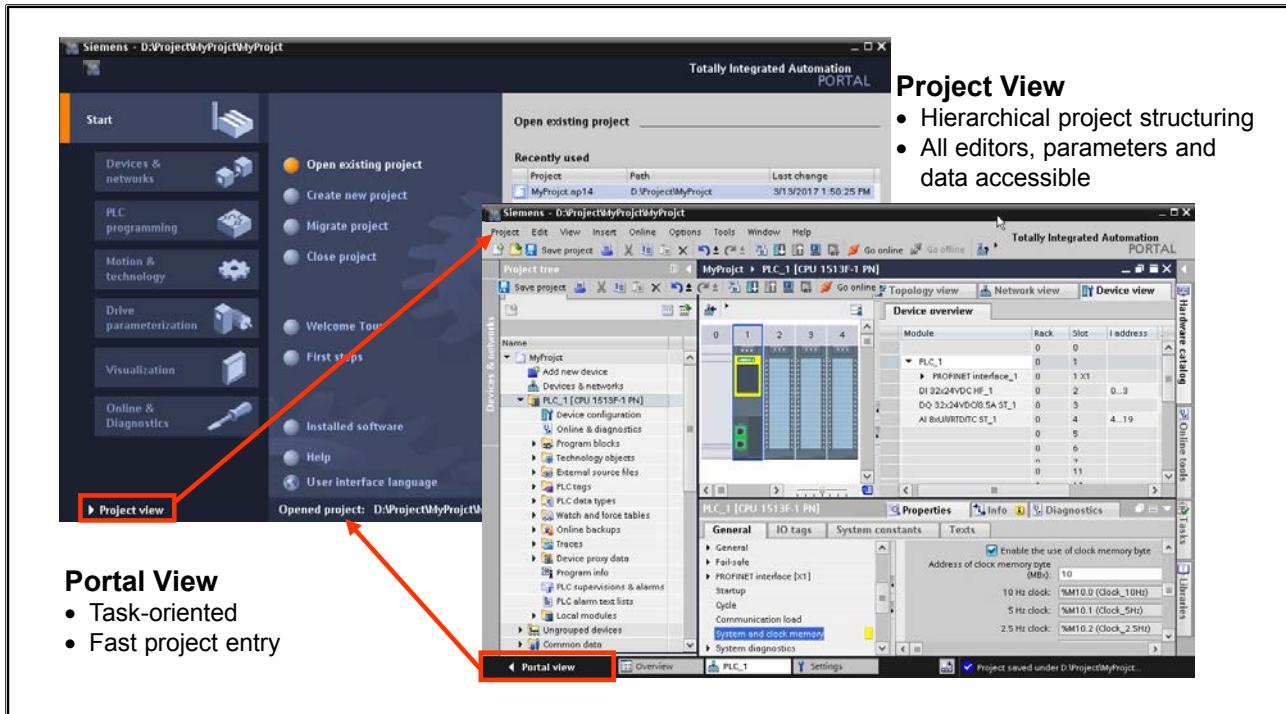
or

under the Entry ID: 64847781
in the Online Support

Compatibility Tool for Automation and Drives

With the help of the Compatibility Tool, you can check the compatibility of the various SIMATIC software versions, either through the TIA Portal Information Center or on the Support pages (<https://support.industry.siemens.com>) under the Entry ID: 64847781.

3.6. TIA Portal: Portal View and Project View



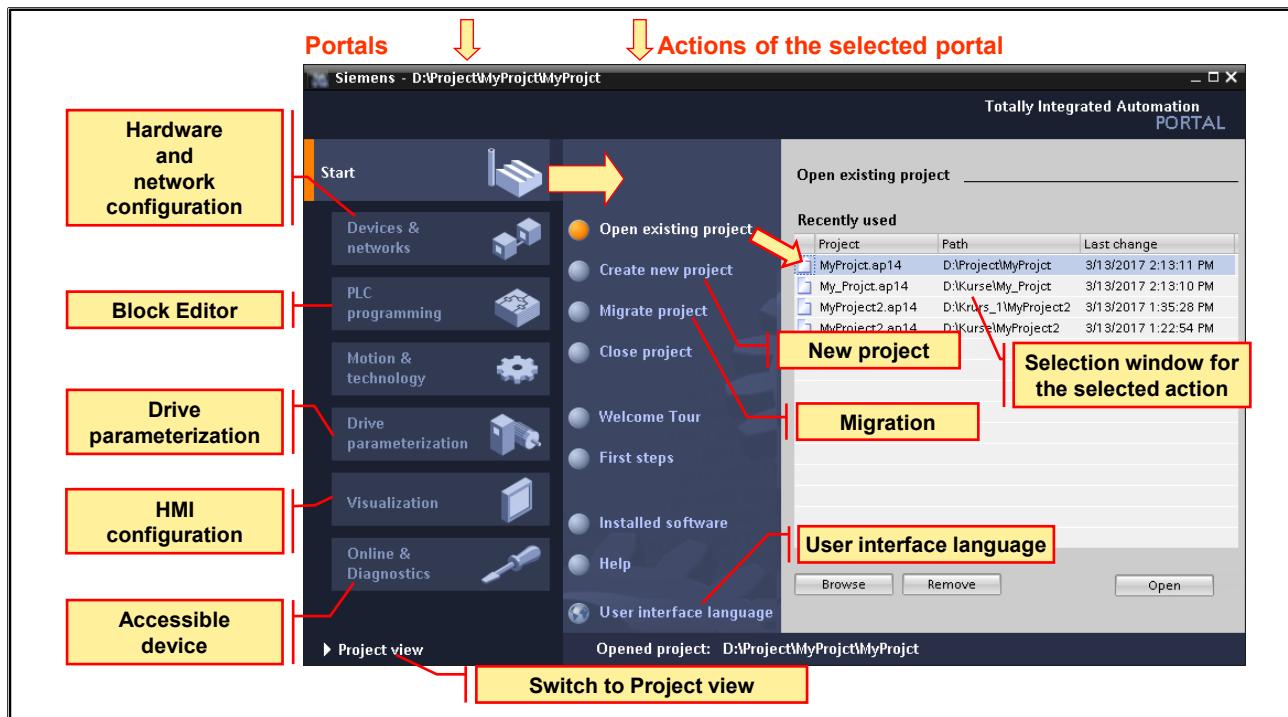
Portal View

- Task-oriented mode of working
- Fast project entry with user guidance

Project View

- Hierarchical structuring of the project
- The necessary editors open according to the task in hand
- All editors, parameters and data are found in one view

3.6.1. Portal View



Layout of the Portal View:

- Portals for the different tasks
- Actions for the selected portal
- Selection window for the selected action

Portals

Access to devices, components and their connections

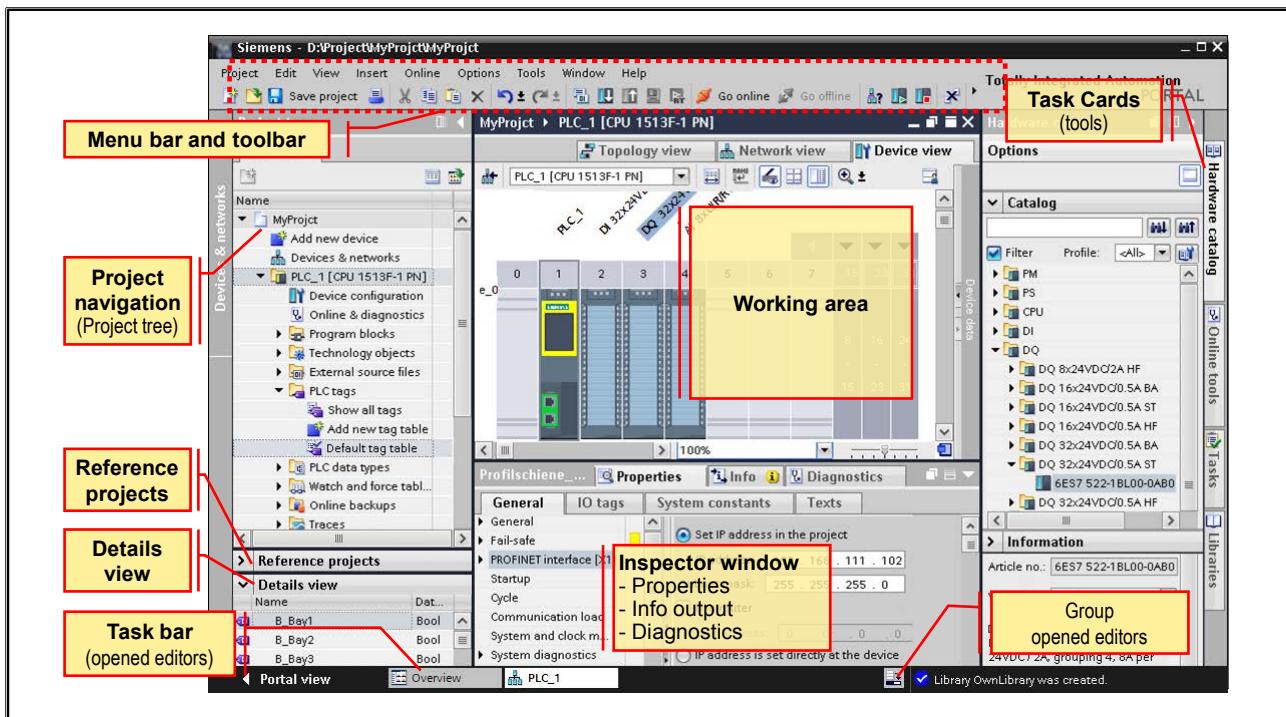
Actions

Depending on the selected portal, actions are available here that can be executed in the selected portal. Context-sensitive help is available in every portal.

Selection Window

The selection window is available in all portals. The content of the window adapts to your current selection.

3.6.2. Project View



Project Navigation (Tree)

The Project tree contains all components and project data of an automation solution. All components can be opened from there.

Working Area

The objects opened for editing are displayed in the working area. These objects include, for example hardware components, blocks, PLC tag tables, screens of HMI devices etc. If several objects are open at the same time, they are displayed in the task bar as tabs (individually or grouped according to editors).

Task Cards

These provide tools for configuring/programming. The content of the Task Cards depends on the object displayed in the working area.

Inspector Window

Additional information on a selected object or on executed actions is displayed in the Inspector window. The available properties of the selected objects can also be edited here (for example, properties of screens, screen objects, tags).

The Inspector window displays all system messages from the engineering, for example, those resulting from generating a project. This window should always be checked for any errors and warnings after a generation is completed.

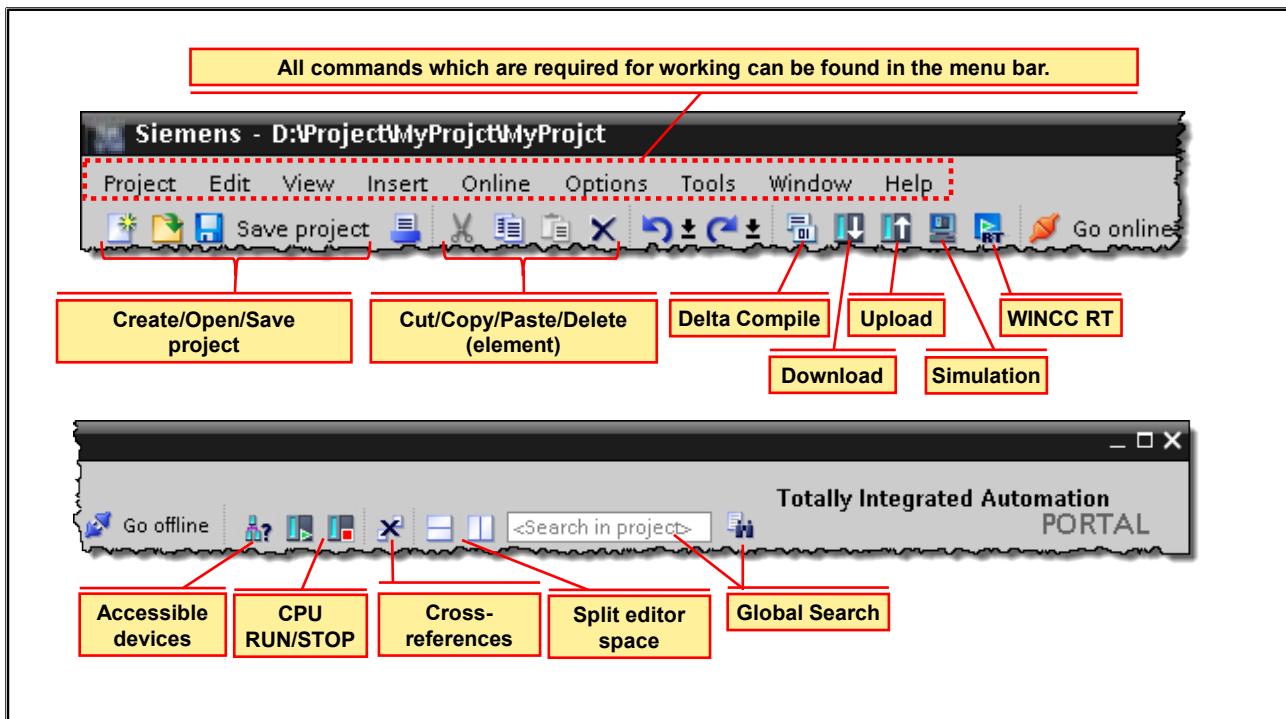
Details View

The Details view contains a help window. In it, the elements of the configuration object selected in the Project tree are displayed. These can be used in the active working area (by dragging them to the working area using drag & drop).

Reference Projects

Reference projects are write-protected projects which can be used for comparison or as a template.

3.6.3. Menu Bar and Toolbar



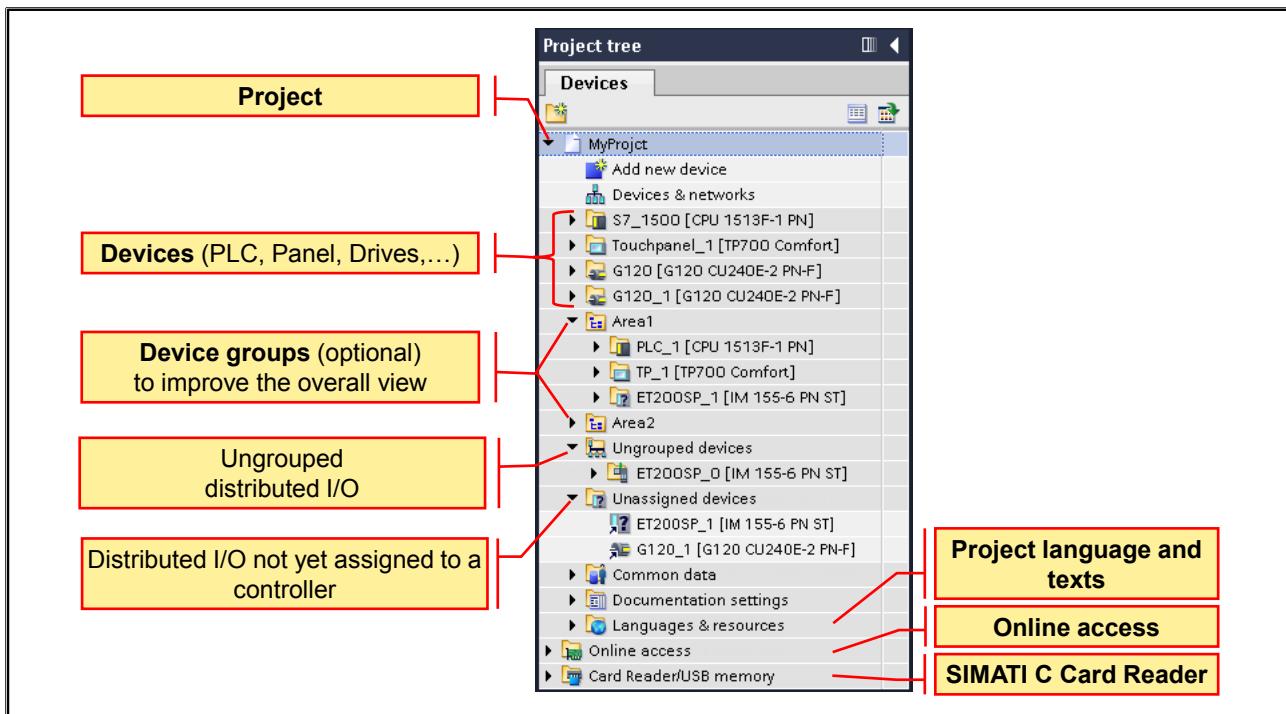
Menu Bar

The menu bar contains all commands required for your work.

Toolbar

The toolbar provides buttons for frequently required commands. That way, you can access these commands faster.

3.6.4. Project Tree (First Level)



The "Project tree" window provides access to all components and project data. All components and all available objects of a project appear in the Project tree in a tree structure and can be opened from there by double-clicking on them.

The following actions can be carried out:

- adding new components (controllers, HMI devices etc.)
- editing existing components
- querying and modifying properties of existing components
- diagnosing accessible components

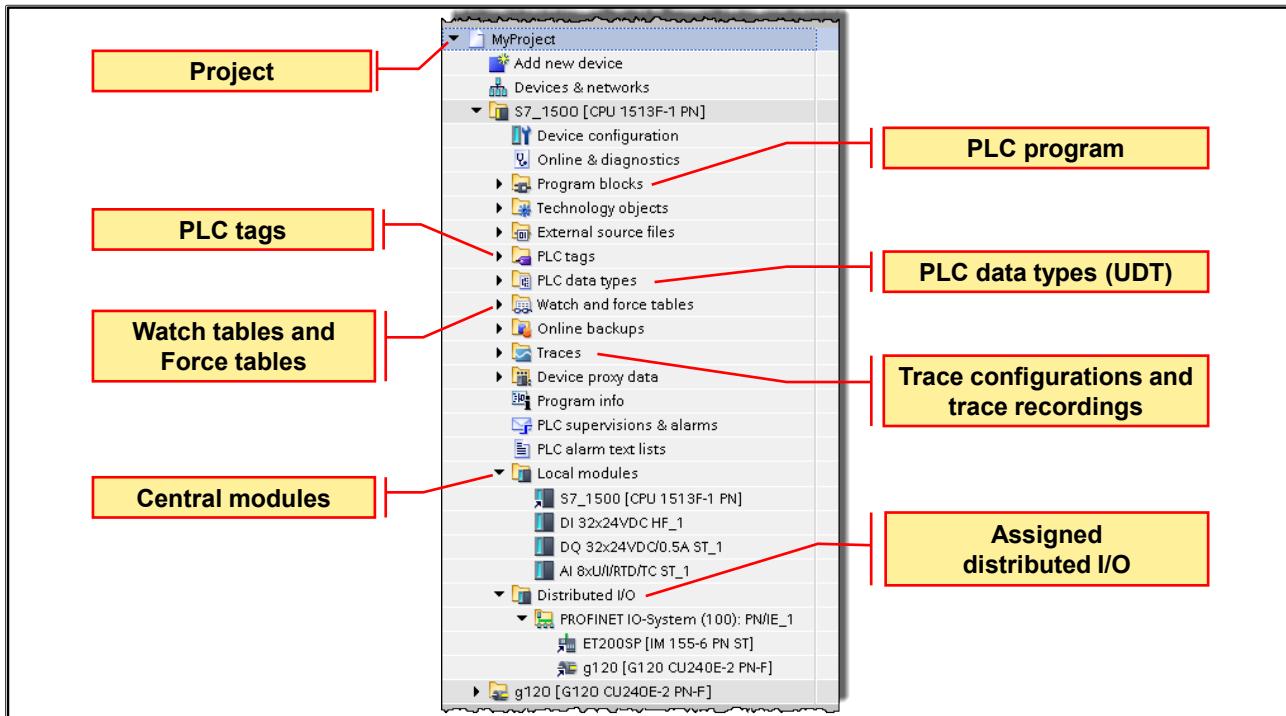
To improve clarity, objects (entire stations) can be grouped together.

Newly inserted distributed I/Os are stored in the folder "Ungrouped devices" and can be moved to the groups which you have created yourself.

A link to a distributed I/O is found in the folder "Unassigned devices" until it is assigned to a controller or master.

The folders "Common data", "Documentation settings" and "Languages & resources" refer to the project; the folders "Online access" and "Card Reader/USB memory" are project independent.

3.6.4.1. Project Tree (Second Level)



For a better overall view, blocks can be arranged in block groups which you create yourself. This grouping merely serves the overview of the program and has no impact on the execution of the program. This information is not loaded into the CPU.

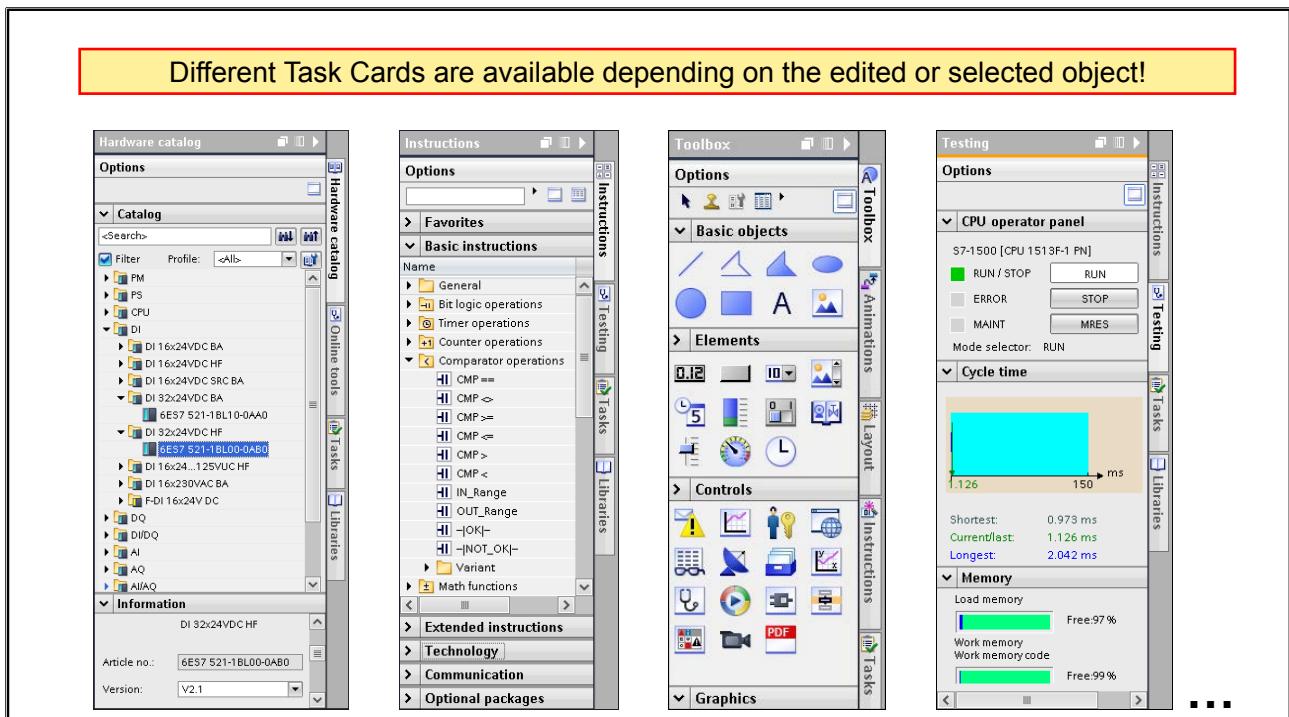
All central modules are stored in the folder "Local modules".

If a device or slave was assigned to a controller or master, the device can be found in the folder "Distributed I/O" of the relevant controller/master.

Hiding/Showing a Structure Section

An underlying structure is indicated by the black triangle ▶. By clicking on the triangle, the underlying structure level can be shown ▶→▼ or hidden again ▼→▶.

3.6.5. Task Cards



Which task cards are available depends on the products that have been installed and on the object currently being edited or open in the working area. If not all Task Cards are visible, the Task Card-bar can be shifted using the cursor buttons at the bottom right.

- **Hardware Catalog**

Here, all the available hardware components such as CPUs, modules etc. can be selected.

- **Instructions**

Instructions for programming blocks;

Code templates and function list wizard for script programming (VBS as well as C scripts with WinCC Professional)

- **Toolbox**

Configurable screen objects (graphics, display and operator control objects) in different panes (basic objects, elements, controls - optional customized controls, graphics)

- **Online Tools**

If there is an online connection established, diagnostics and online information can be called, such as, the current cycle time of the CPU and the configuration of the load and work memory of the CPU. As well, the CPU can be switched to the STOP and RUN mode.

- **Animations**

Templates for making screen objects dynamic in different panes (movements, display, tag link for making dynamic)

- **Layout**

Tools for adapting the presentation when designing screens during configuration of HMI devices (zoom, level assignment, grid alignment, objects outside the area)

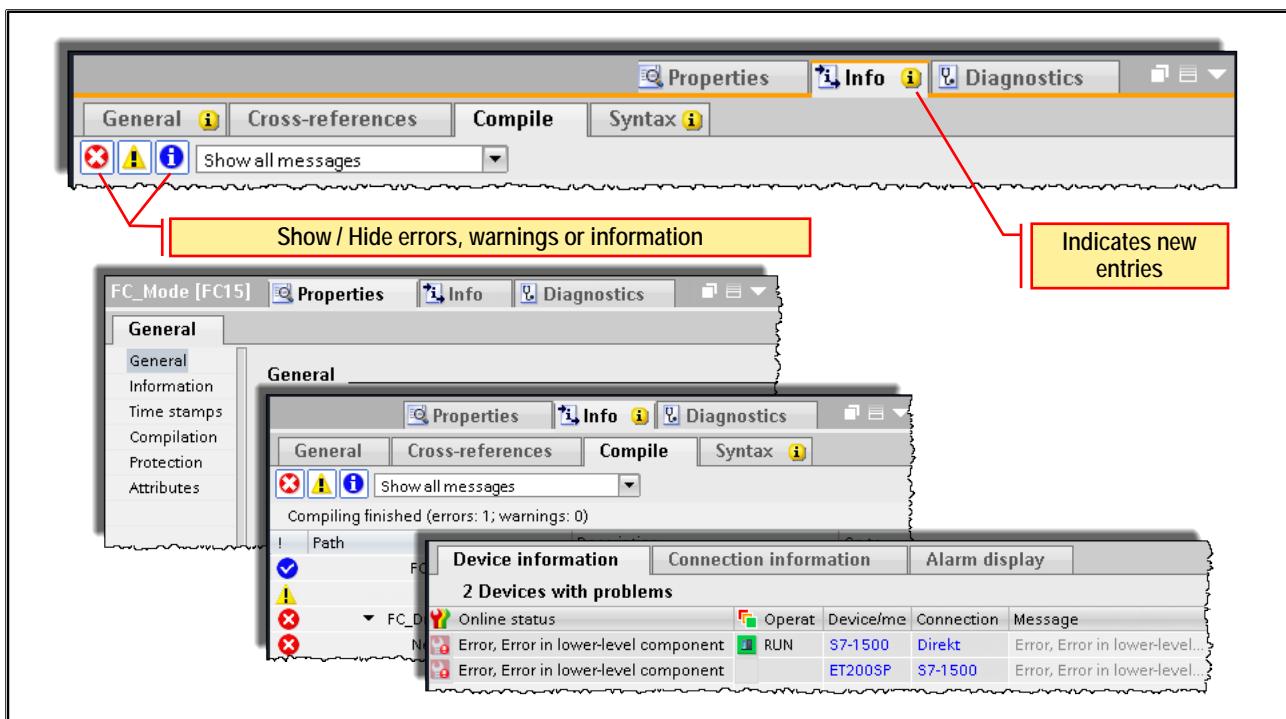
- **Tasks**

Here, classic editor functions are available such as finding and replacing tags, instructions etc.

- **Libraries**

Management of the local project library and global libraries

3.6.6. Inspector Window



Additional information on a selected object or on actions to be executed is displayed in the Inspector window. The Inspector window consists of the following tabs:

→ can be selected by clicking the tabs

This symbol in the tab indicates new entries.

If errors are displayed, you can jump to the error location or into the associated editor by double-clicking on the error information.

"Properties" Area

This tab displays the properties of the object selected in the working area and editable properties can be changed.

"Info" Area

This is the output area of the engineering. This tab displays further information for the object selected. In addition to this, messages relating to executed actions, for example, compilation and download of blocks to the CPU are output.

"General" tab → general status output

"Cross-references" tab → display of the current locations at which the selected object is used

"Compile" tab → status display of compilation progress

"Syntax" tab → status display for invalid programming commands

"Diagnostics" Area

This tab displays information on system diagnostics and configured alarm events

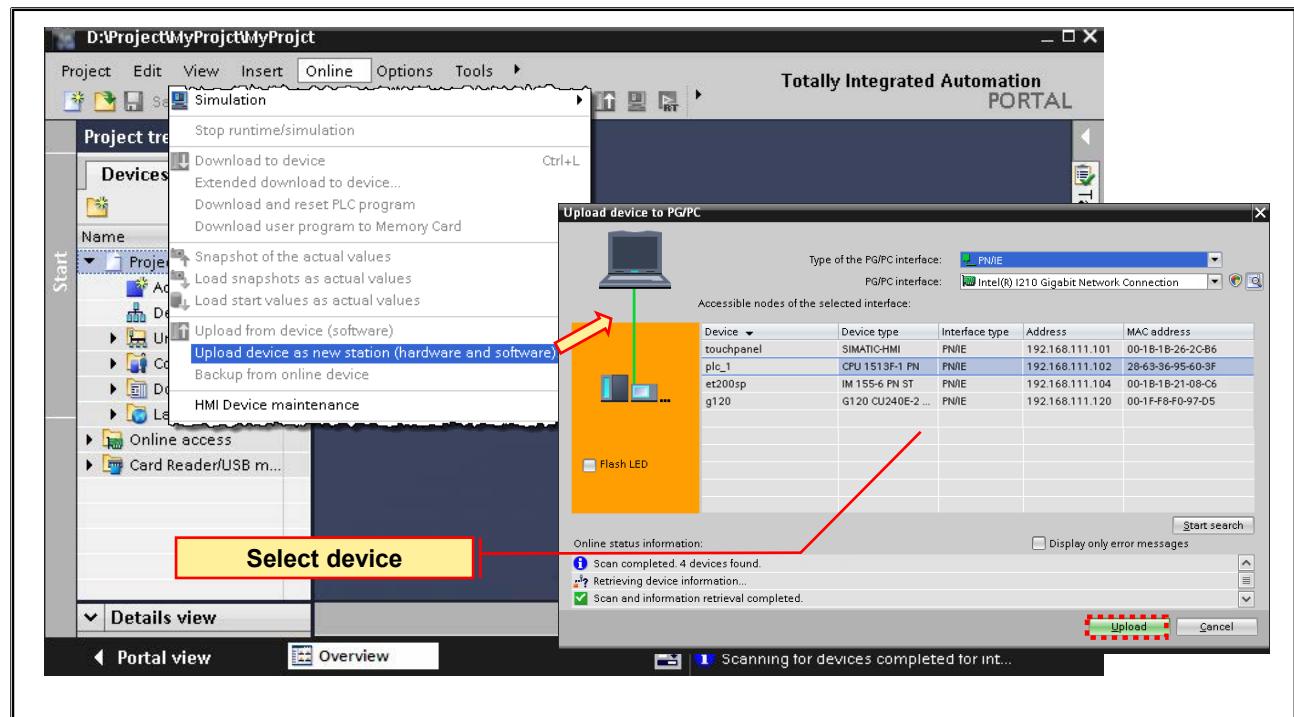
"Device information" tab → Information about the state of the devices

"Connection information" tab → detailed diagnostics of connections

"Alarm display" tab → Display of currently pending CPU alarms

"Monitor value" tab → Monitoring of structures in a block

3.7. Uploading a Device as a New Station into the Project (1) (Uploading the Entire Online Project in the Offline Project)



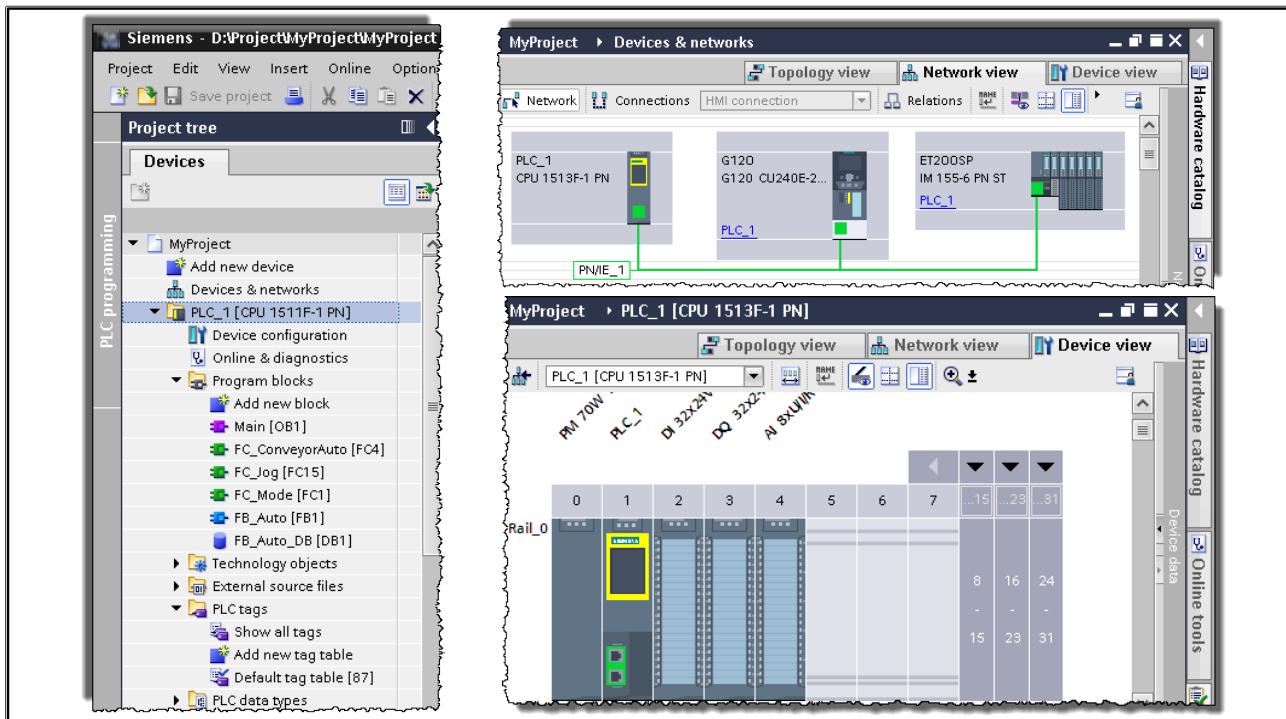
An already existing configuration of an S7 station can be read-out by means of the function "Upload device as new station".

This is then necessary when the appropriate offline station **doesn't** exist on the PG. After reading out the S7 station, the hardware as well as the program can be adjusted or modified, saved and downloaded into the CPU.

Requirement:

An IP address has been assigned to an interface of the station or the station has already been configured.

3.7.1. Uploading a Device as a New Station into the Project (2) (Uploading the Entire Online Project in the Offline Project)



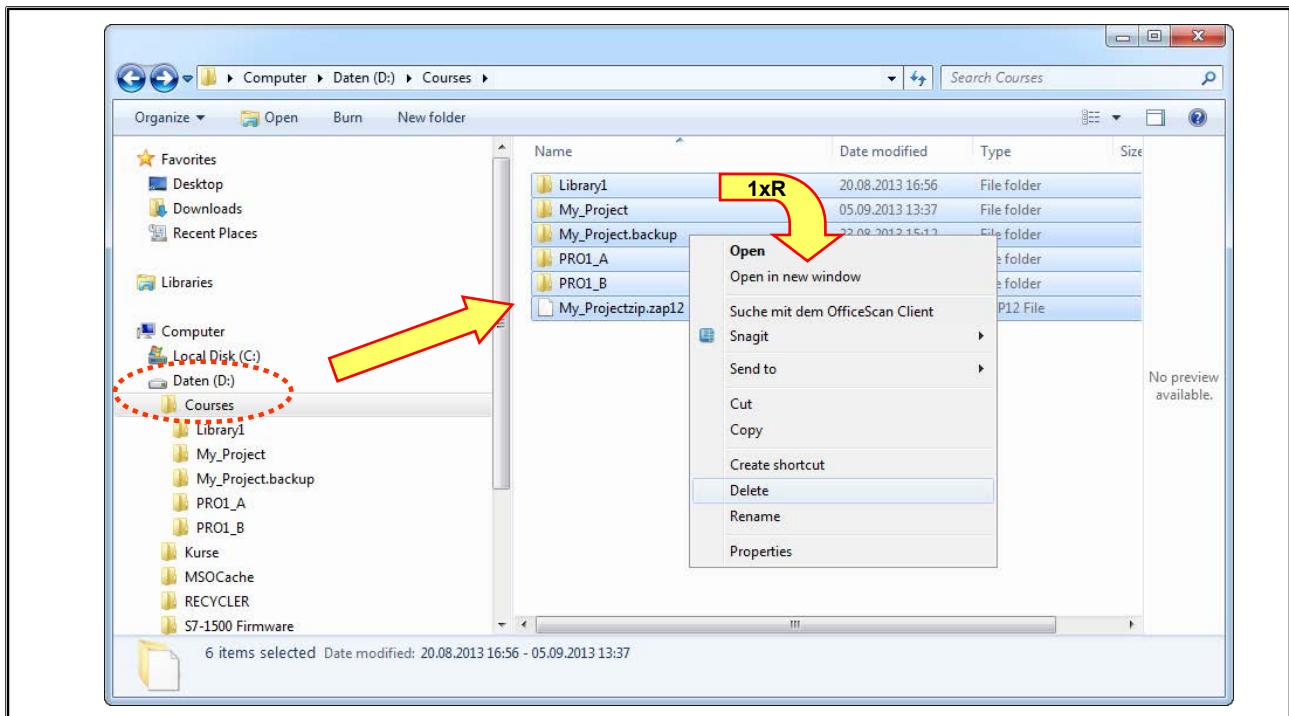
Station with Configuration:

If the station is already configured, then the entire station (central and distributed hardware with parameter assignments, the entire program with comments and symbols) is available to the user offline for further processing after the device is uploaded.

Station without Configuration:

If the CPU was only assigned an IP address without further configuration, only the actual configuration of the central hardware is available to the user after the upload.

3.8. Exercise 1: Deleting Old Projects



Task

You are to delete the TIA Portal projects on the PG.

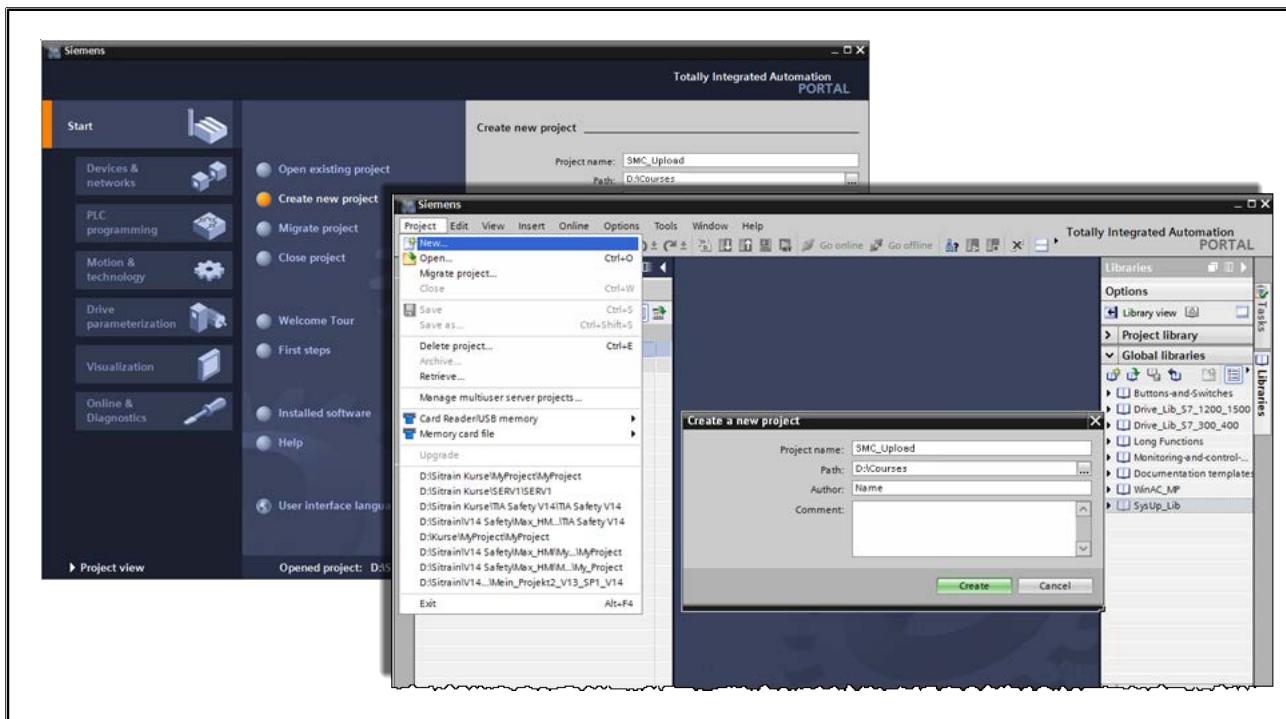
What to Do

1. Start the Windows Explorer.
2. In the directory D:\Courses, delete all projects.

Note

Projects that are to be deleted must be closed!

3.8.1. Exercise 2: Creating a Project



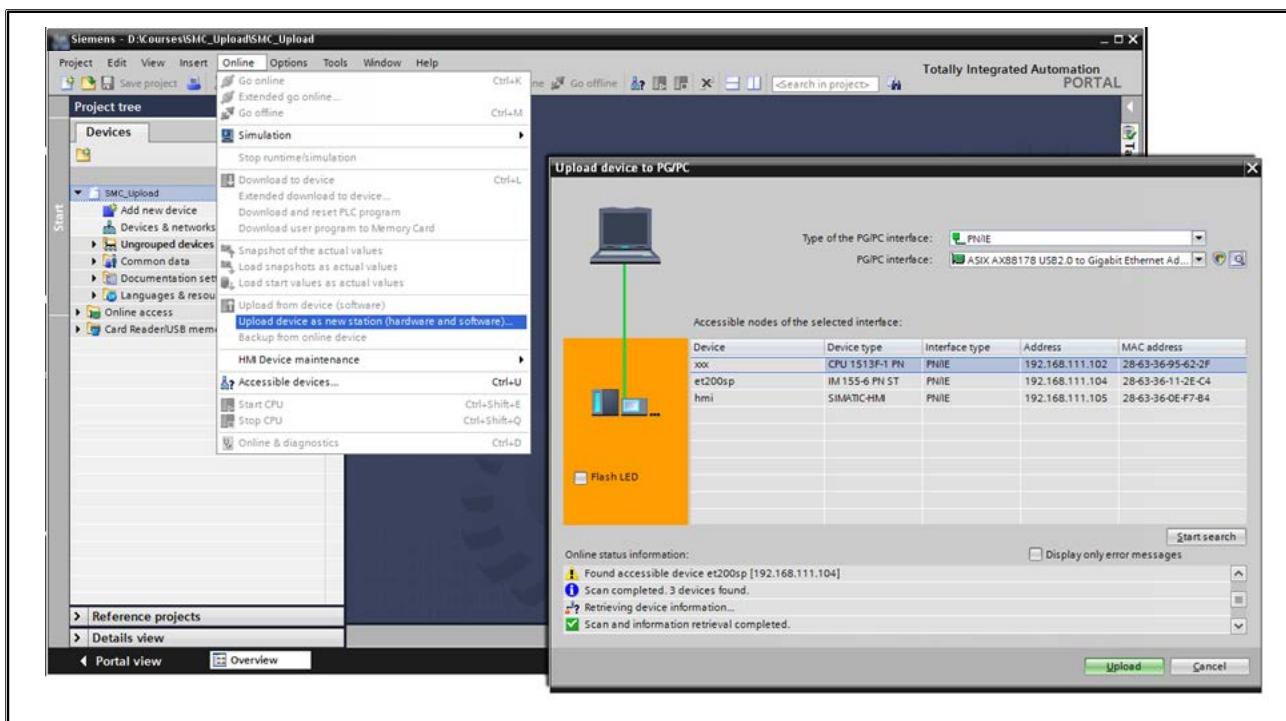
Task:

You are to create a new TIA Portal project.

What to Do:

1. Open the TIA Portal.
2. Create a new project with the name "SMC_Uplod" in the folder D:\Courses.
Portal view > Start > Create new project
or
Project view > Project > New
or
via the "New" button in the toolbar of the Project view.

3.8.2. Exercise 3: Uploading Devices as a New Station



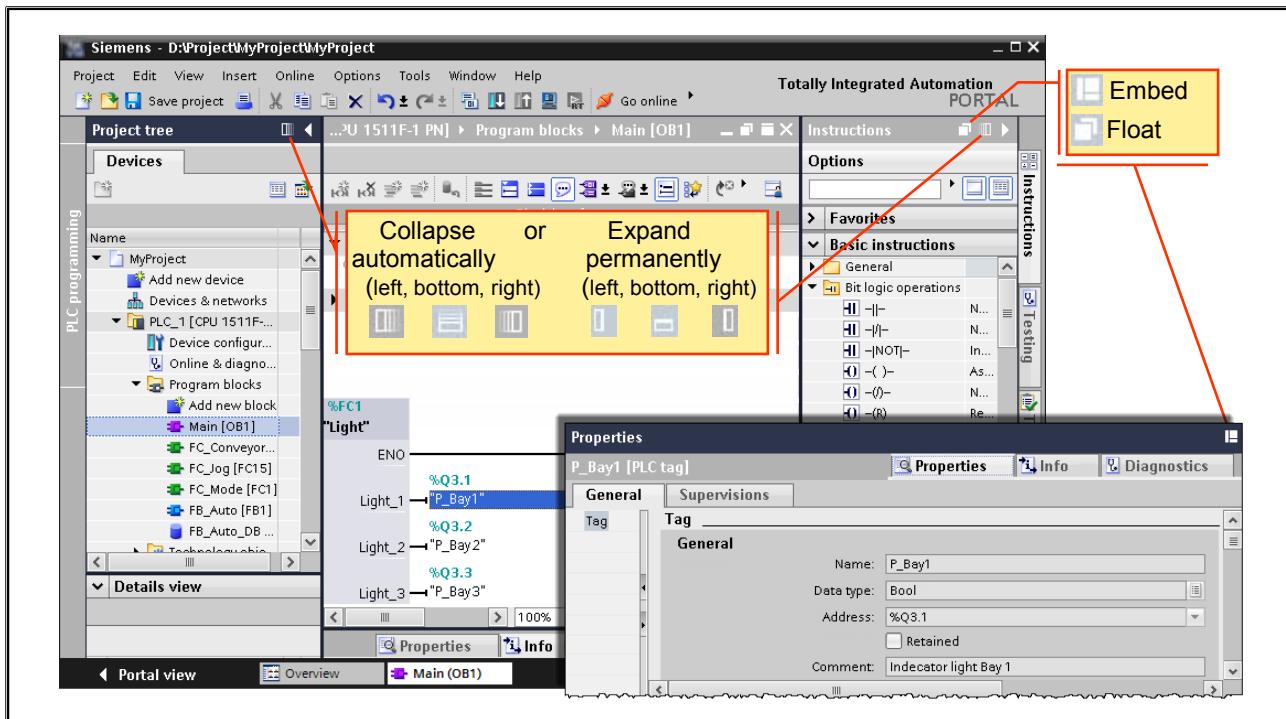
Task:

You are to upload the current program which is located on the CPU as a new station in the project which you created.

What to Do:

1. Select the project name in the Project tree,
2. Open the menu Online > "Upload device as new station (hardware and software)..."
3. Select the type of interface with which your PG is connected to the CPU.
(PN/IE for PROFINET)
4. Select the PG/PC interface with which your PG is connected to the CPU.
5. Start the search via the button "Start search"
6. Choose the S7-1500 device and start the device upload.
7. Save the project and become familiar with the program.

3.9. Window Arrangement



The current configuration of the engineering user interface is saved in the user profile of Windows. On saving the project, the positions and characteristics of windows are automatically saved with it.

Window Arrangement Options

- When the window is 'Expand permanently'
 - fixed location and fixed size on user interface
 - position at left, bottom or right outside of the working area is possible
 - always open, reduces the working area
- When the window is 'Collapse automatically'
 - hidden at edge of user interface
 - position at left, bottom or right is possible, superimposed on working area when open
 - default status = window closed, and tab displayed at edge of the user interface
 - mouse click on the tab opens the window
 - closed automatically the next time there is a click outside the window area
- 'Float' window!! **Makes sense if a 2nd monitor is used!!**
 - can be positioned anywhere on the user interface
 - permanently covers the user interface area underneath it

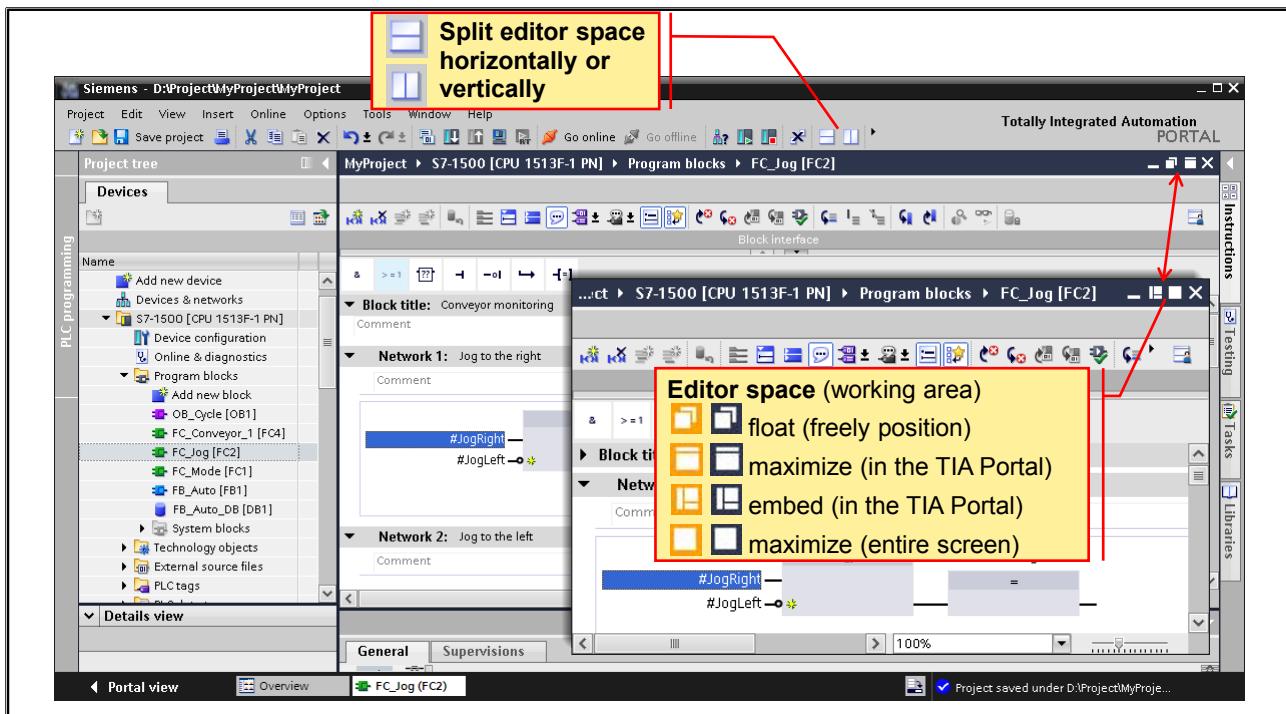
By clicking the functions in the window title bar you can switch between the modes "float" and "embed" or "collapse automatically" and "expand permanently".



In addition, the windows can be expanded and collapsed via the buttons

Hidden windows are opened by clicking on the tab and closed again by clicking outside the window area.

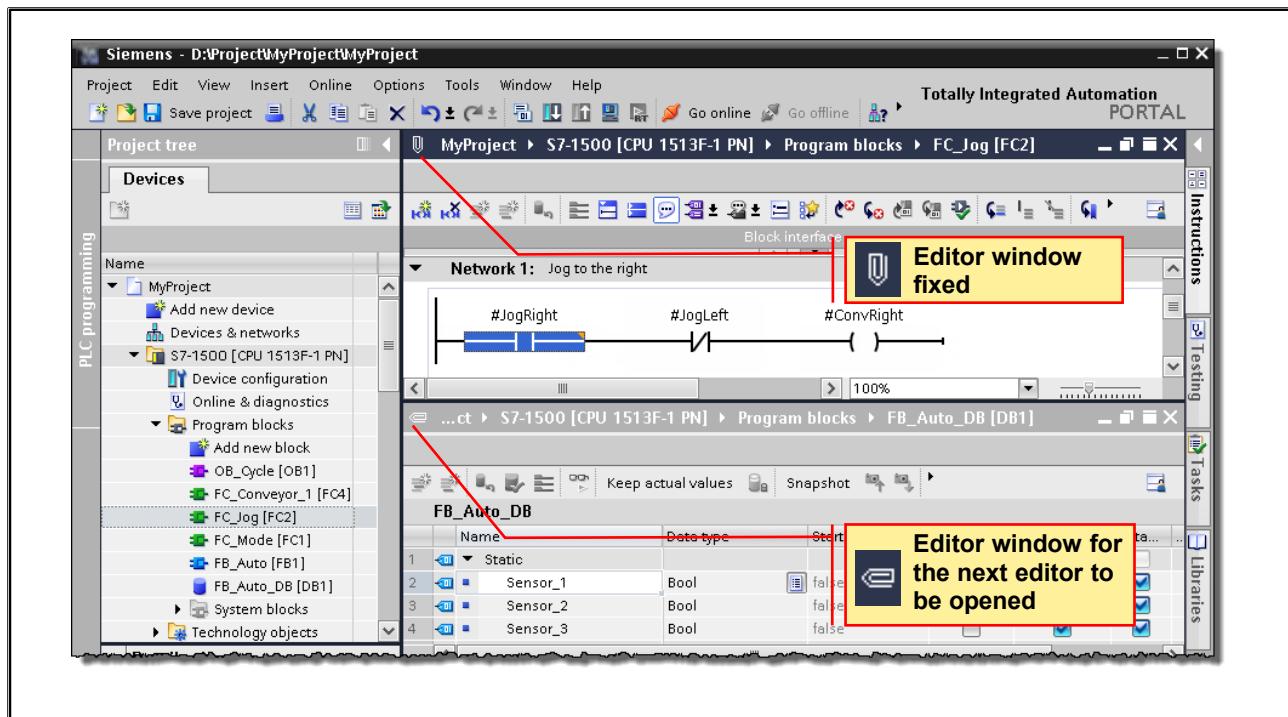
3.9.1. Splitting and Arrangement of the Working Area



The windows of the editor space (working area) can be arranged as follows:

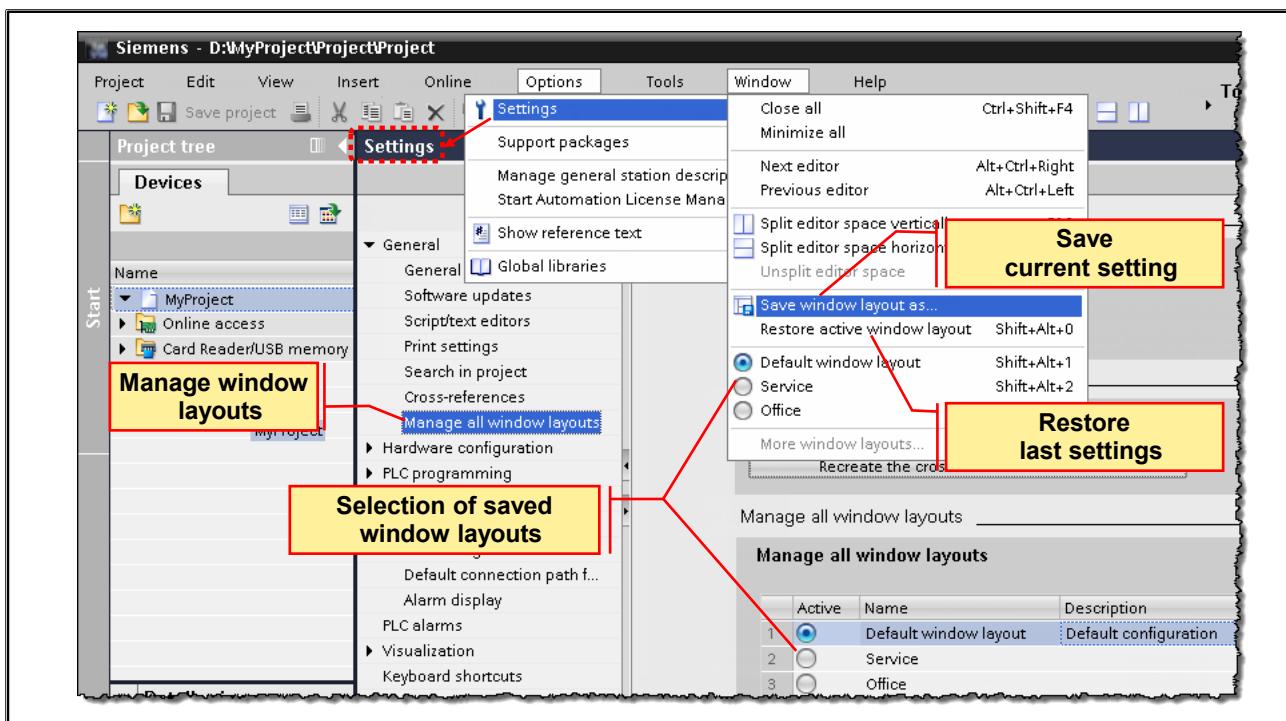
- Maximize (full size) a working area to cover the entire screen
(color depending on View online/offline)
- Maximize a working area in the TIA Portal (Project tree, Task Card and Inspector window are minimized)
(color depending on View online/offline)
- Float or release a window from the working area
(color depending on View online/offline)
- Embed a window in the working area once again
(color depending on View online/offline)
- Split the editor space (working area) horizontally into two windows
- Split the editor space (working area) vertically into two windows

3.9.2. Keeping the Editor Window in the Foreground (when Editor Space is Split)



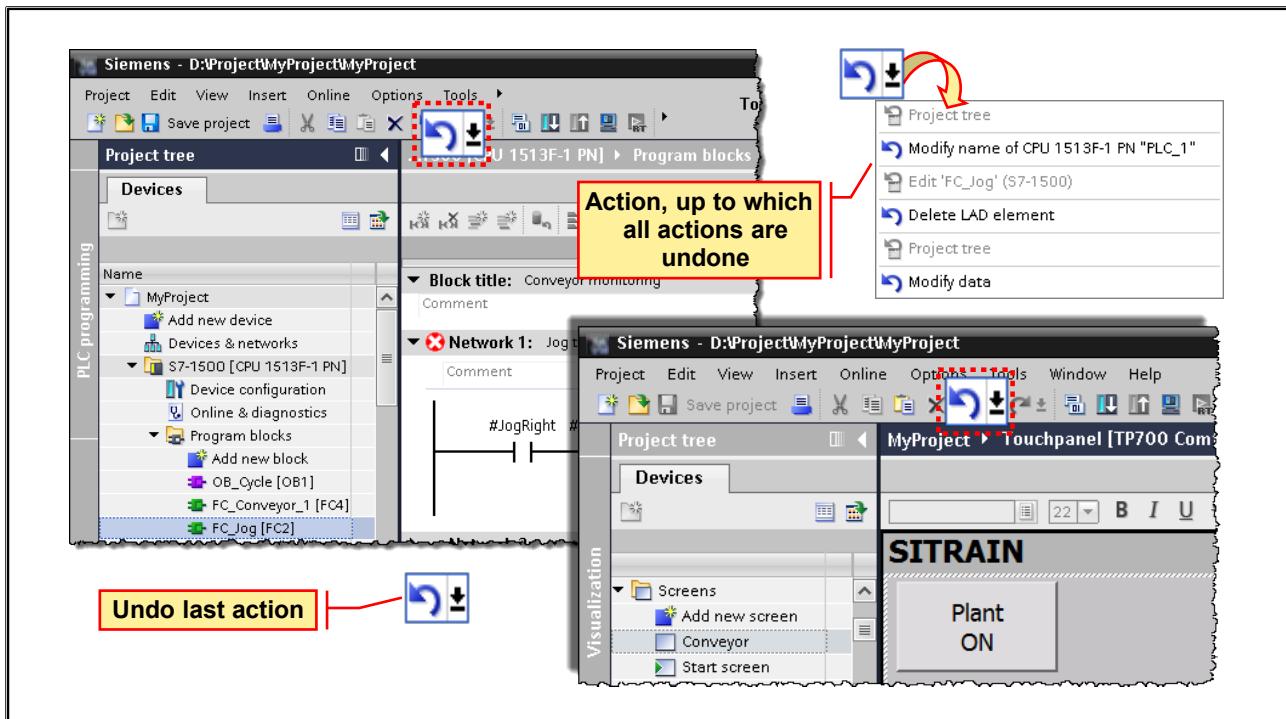
If you work with a split editor space (working area), one of the two working areas can be fixed (attached) by clicking on the "paper-clip" (paper-clip is vertical) so that when you open a further editor, this first one always remains fixed in the foreground and the newly opened one always becomes the second visible editor.

3.9.3. Save / Manage / Use Window Layouts



The different window arrangements of the user interface can be saved and then restored.

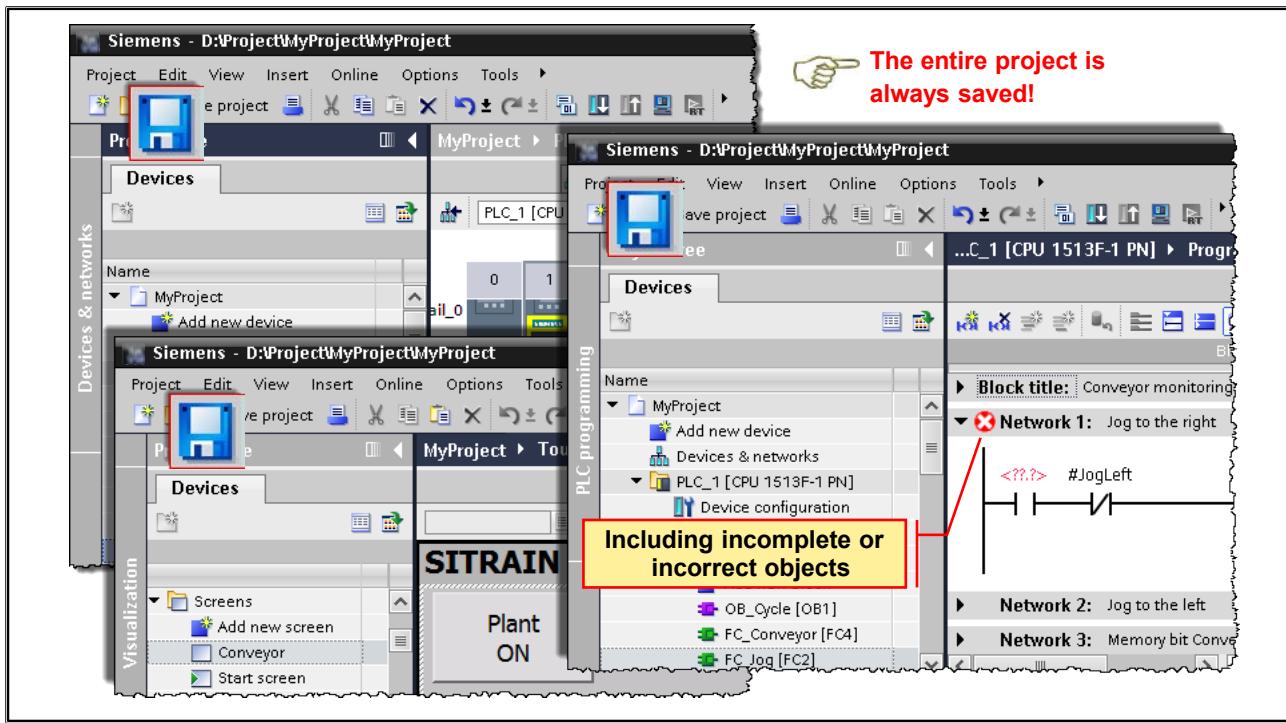
3.10. Undo and Redo



Undo Concept of the TIA Portal

The drop-down menu shows the user in which editor the "Undo" function is executed. Closed editors are then automatically opened. Since all actions are only undone up to the selected action, the consistency of the project is ensured.

3.11. Saving a Project

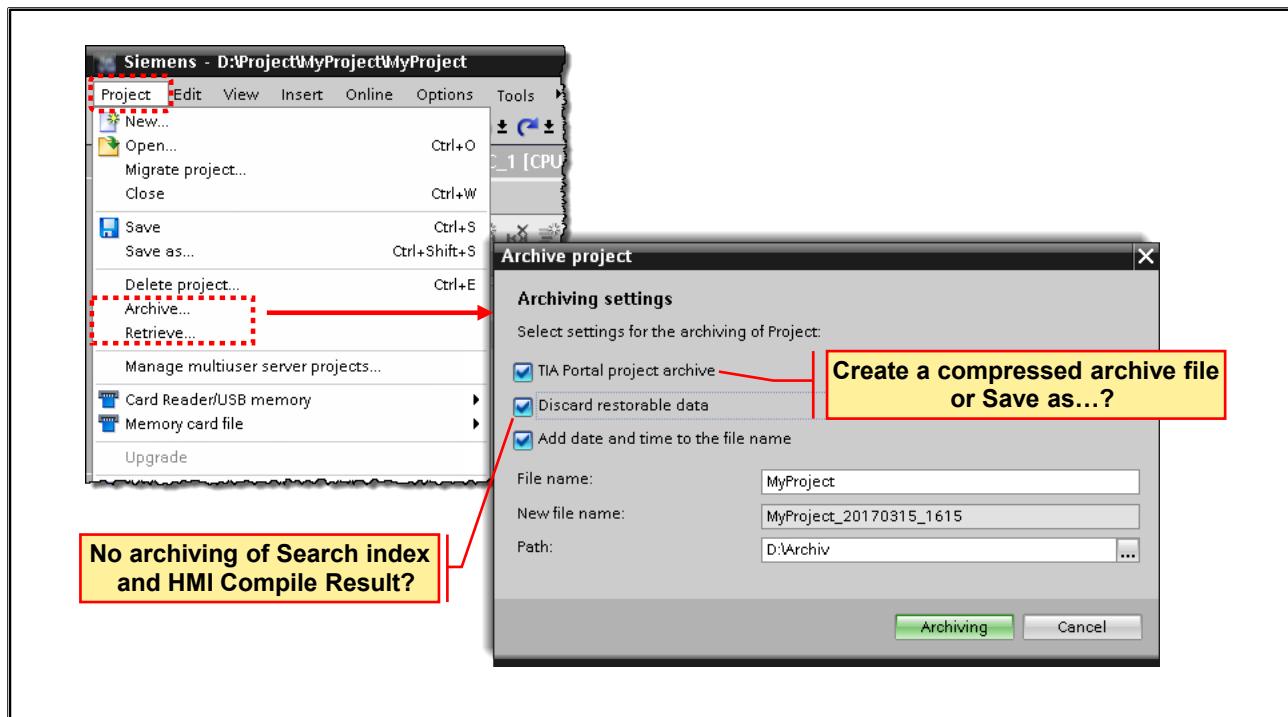


Save

Regardless of the object that is open in the working area, it is **always the entire project** in the current state that is saved when the Save icon is pressed, even if some objects of the project are still incomplete or faulty [incorrect] (for example, syntax faulty blocks or symbols which have not yet been assigned an absolute operand in the global symbol list).

If the project is closed without saving, all changes made or objects created during the session are discarded.

3.11.1. Archiving / Retrieving a Project



Archiving

The current state of the project can be archived at any time.

When you archive projects, you can choose the following:

- **TIA Portal project archive**
The project is minimized (all files are reduced to their essential components) and then stored compressed in a project archive (file with the ending zapXX).
If this item is not selected, the project is saved under the given name and path as an independent project. (Save as... without closing current project)
- **Discard restorable data**
Search index and the HMI Compile Result are not archived.
- **Add date and time to the file name**
The current date and time is added to the selected archive name.

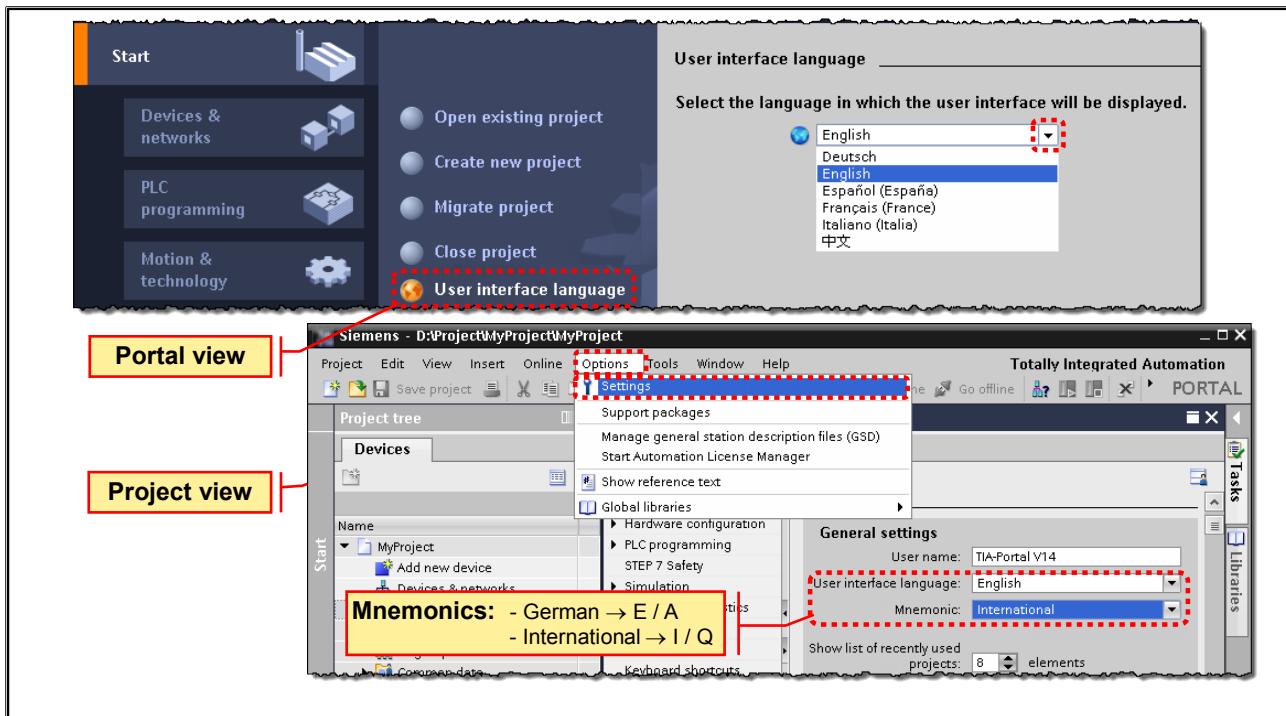
Note

The most recently saved version of the project is archived. If the last changes to the project are also to be included in the archive, the project must be saved before archiving.

Retrieving

Only project archives (file with the ending zapXX) can be retrieved, that is, unzipped.

3.12. TIA Portal - Settings: User Interface Language

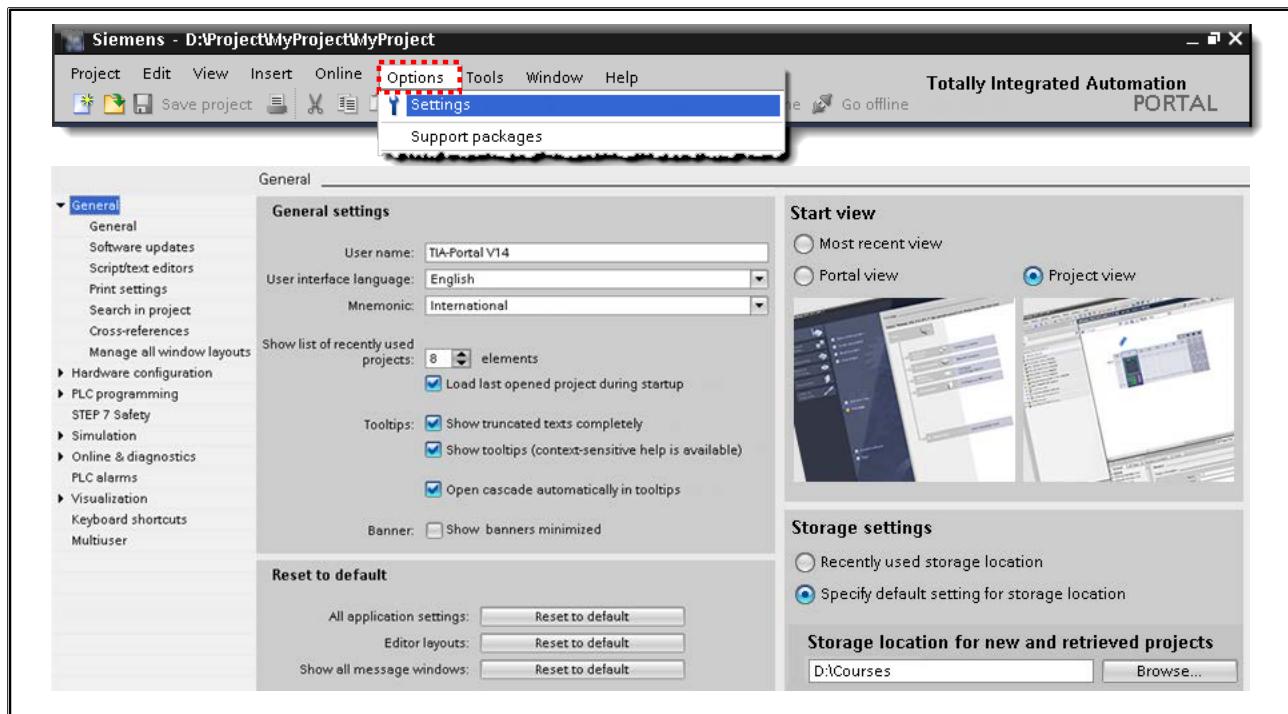


Available User Interface Languages

The user interface language of the TIA Portal can be changed during running operation. The following languages are available:

- German
- English
- French
- Spanish
- Italian
- Russian
- Korean
- Japanese
- Chinese (simplified)

3.12.1. TIA Portal - Settings: Language, Storage Location, Layout



Language

The user interface language of the TIA Portal can be changed at any time without needing to restart. The TIA Portal always starts in the language in which it was last used.

Storage Settings

Storage location for projects:

Storage location of newly created projects and their project libraries

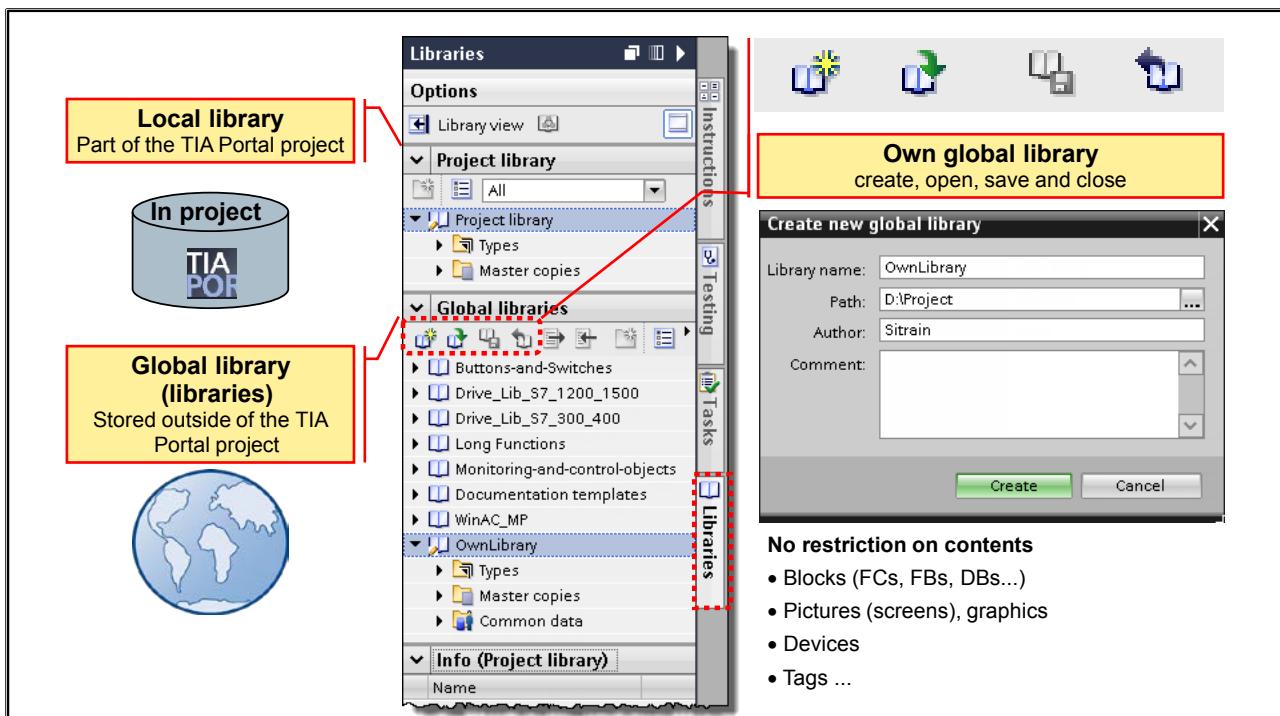
Storage location for libraries:

Storage location for global libraries

Layout

If the layout is reset, the original window layout arrangement of the TIA Portal is restored.

3.13. Libraries



Project Library

Each project has its own library. Here, objects can be stored that are to be reused within the project. This project library is always opened, saved, and closed together with the current project.

Global Libraries

Global libraries are stored independently of the projects and are used to store objects that are to be used and reused in different projects.

The area of the global libraries also contains libraries supplied with the TIA Portal that, for example, contain ready-made functions and function blocks.

Library Objects

A library is a collection of any project objects, such as, blocks, devices, PLC data types, watch tables, screens, graphics, faceplates....

Uses of Global Libraries

Library objects can either be used as a master copy or as an instance.

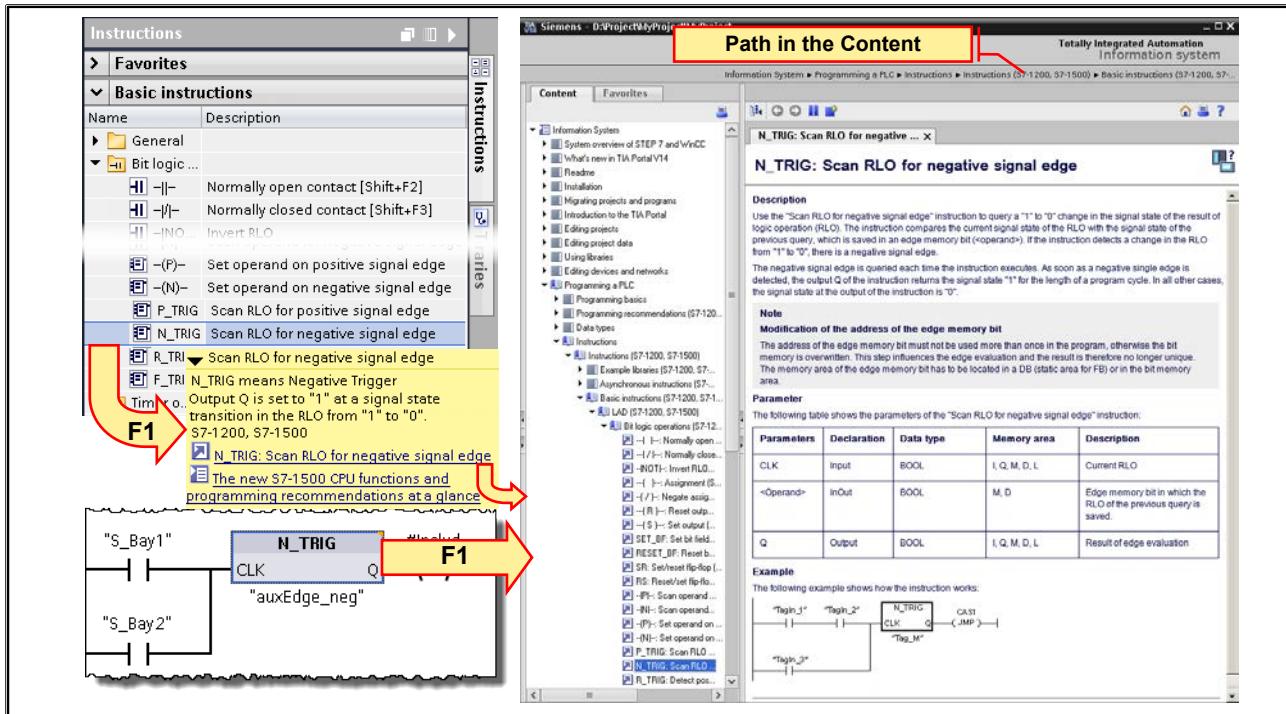
- Objects from the **Master copies** folder are copied to the project when used. If subsequent changes are made to this master copy, these changes are not made to the copies in the project.
- Objects from the **Types** folder are copied to the Types folder of the project library when they are used and an instance (location of use) is created in the project. These objects are then stored in the local project library. The object itself is not used in the project, rather only a reference to it.

You will find more information about libraries in the section **Programming Guideline Libraries**:

"TIA Portal Information Center" > Documentation > Manuals > Control Technology

or in the Online Support under the Entry ID: 90885040

3.14. Help



Wide-ranging help functions are available for solving your tasks; these describe basic concepts, actions and functions.

- **Tooltip** for information on the user interface elements, for example, instructions, input boxes, buttons and symbols
Some tooltips provide cascades with more precise information.



Operating instructions:

Step-by-step approach for implementing a task



Example:

Practice-oriented application example with solution of an automation task



Factual information:

Background information about the functions of the TIA Portal



Reference:

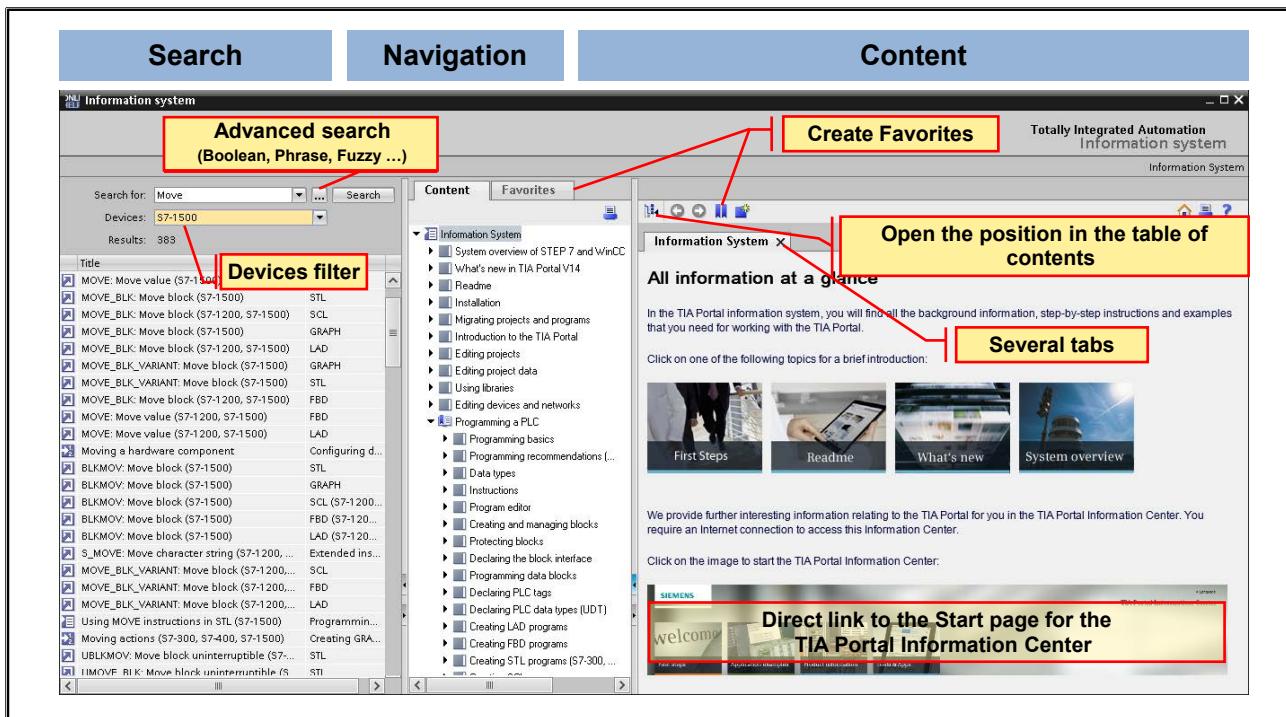
Detailed reference information about instructions and objects

These are activated by clicking (Information system is opened)

- Help on the current context
For example, on menu commands by pressing the <F1> key.
- In the input boxes (for example, in the Properties in the Inspector window), the roll-out provides information about the permitted value ranges and data types for the input.



3.14.1. Help (Information System)

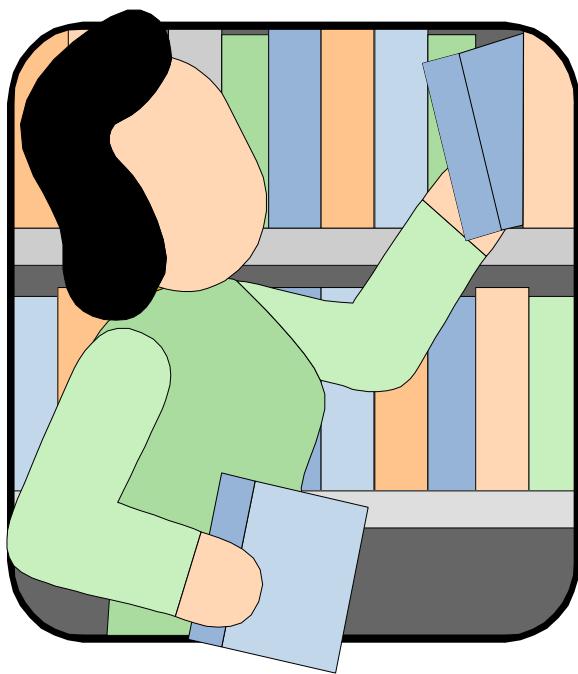


Contents of the Help Functions

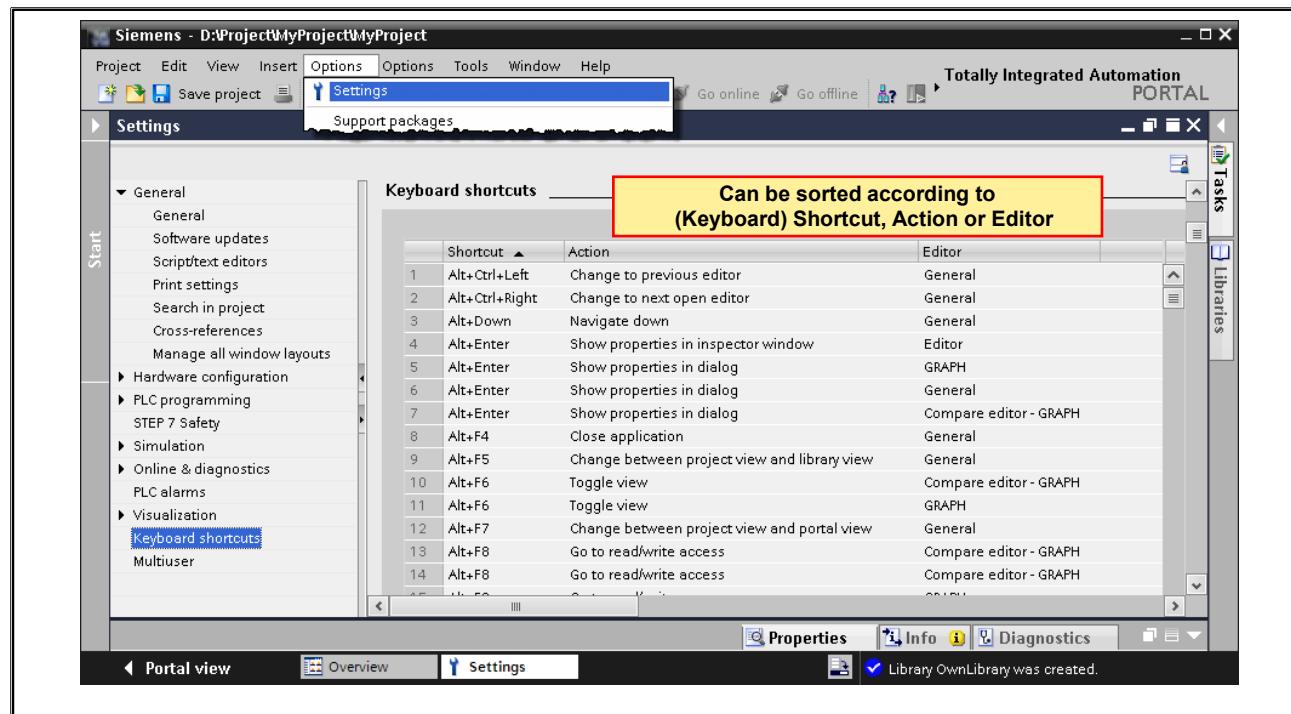
- **Search area**
In the Search area, you can perform a full-text search across all help topics.
- **Navigation area**
In the Navigation area, you find the (table of) Content and the Favorites.
- **Contents area**
The help pages are displayed in the Content area. You can open several tabs in order to simultaneously display different help pages.

You can show and hide the individual areas using the arrows on the window splitters. In that way, you can close the Search area as well as the Navigation area in order to enlarge the Contents area if required.

3.15. Additional Information

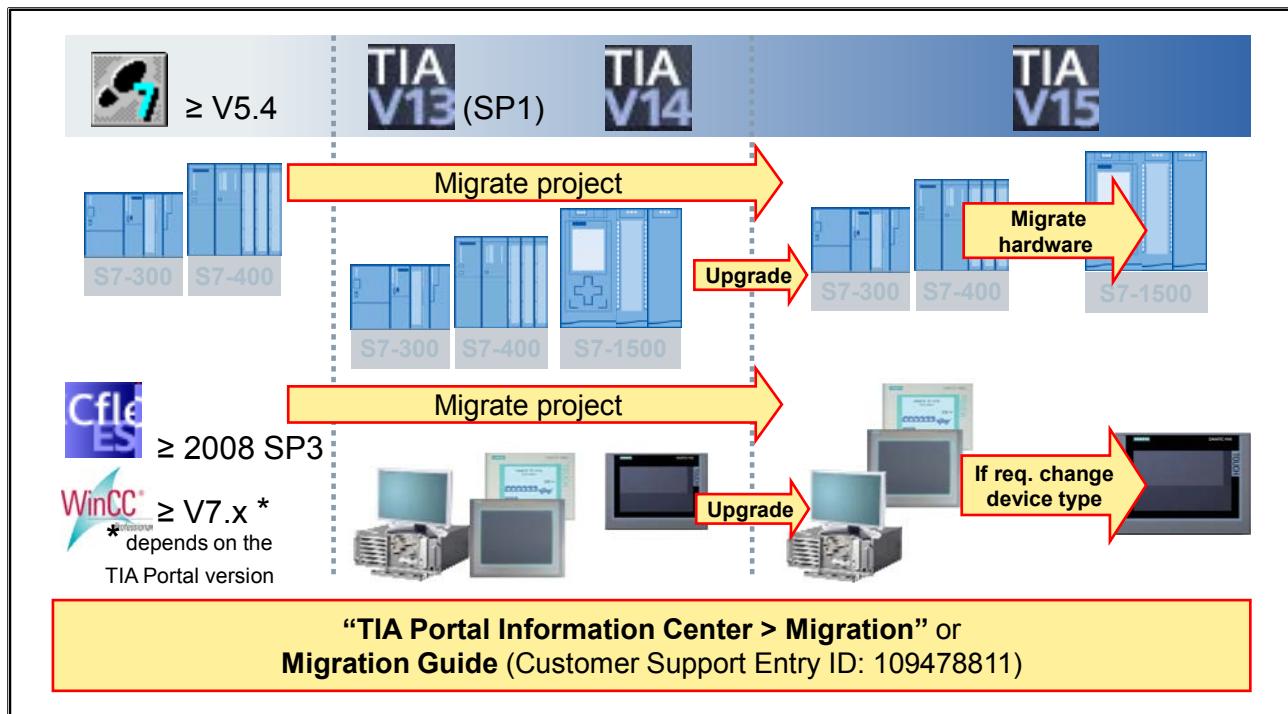


3.15.1. Keyboard Shortcuts of the TIA Portal



The keyboard shortcuts can be displayed through the menu item Options > Settings. The view can be sorted according to (Keyboard) Shortcut, Action and Editor.

3.15.2. Project Migration



What does Migration Actually Mean?

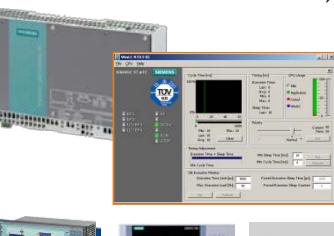
- Switch or change of a system / a technology
- "Step-by-step" modernization of the installed basis and adaptation to modern technology...

Why Migration? – Motivation for Migration

- Investment protection
- Switch to the latest engineering
- Basis for future retrofits/renovations
- Products with higher performance
- Innovative products
- Long-term availability of products
- Reduced product time-to-market
- Lower operating costs

3.15.2.1. Migration of STEP 7 V5.x – Projects: Supported Hardware

Effective date 1.10.2007
→ Availability of Hardware

S7-300/ET 200CPU	S7-400	WinAC	
			
Basic Panels			
			
Mobile Panels	x77er Panels	Panel PC, Standard PC	
Single Station ... based on WinCC flexible RT ... based on WinCC V7 Runtime			
Possible check before migration with the help of the tool: " READINESS CHECK " (TIA Portal Information Center > Tools & Apps > Planning and Configuration or Entry ID: 60162195 in the Online Support)			

With this hardware, the hardware configuration can also be migrated. Otherwise, only the software can be migrated and must be adjusted to the supported hardware.

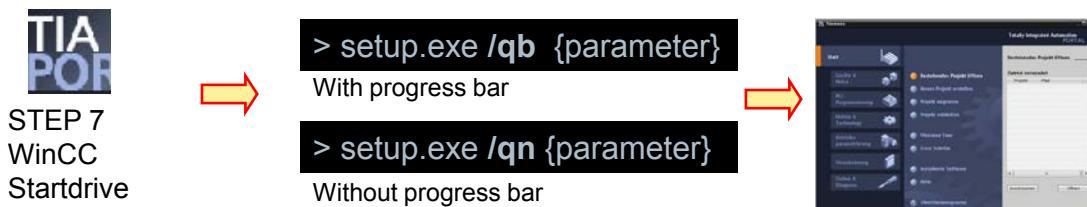
To check whether the hardware is supported by TIA Portal, it is possible to check this in advance. For this, you can use the "READINESS CHECK" tool which is available for download on the Support Pages (<https://support.industry.siemens.com>) under the Entry ID: **60162195** or via the "TIA Portal Information Center".

3.15.3. Installation with Record Function in the Setup

Initial installation with Record function (config file)

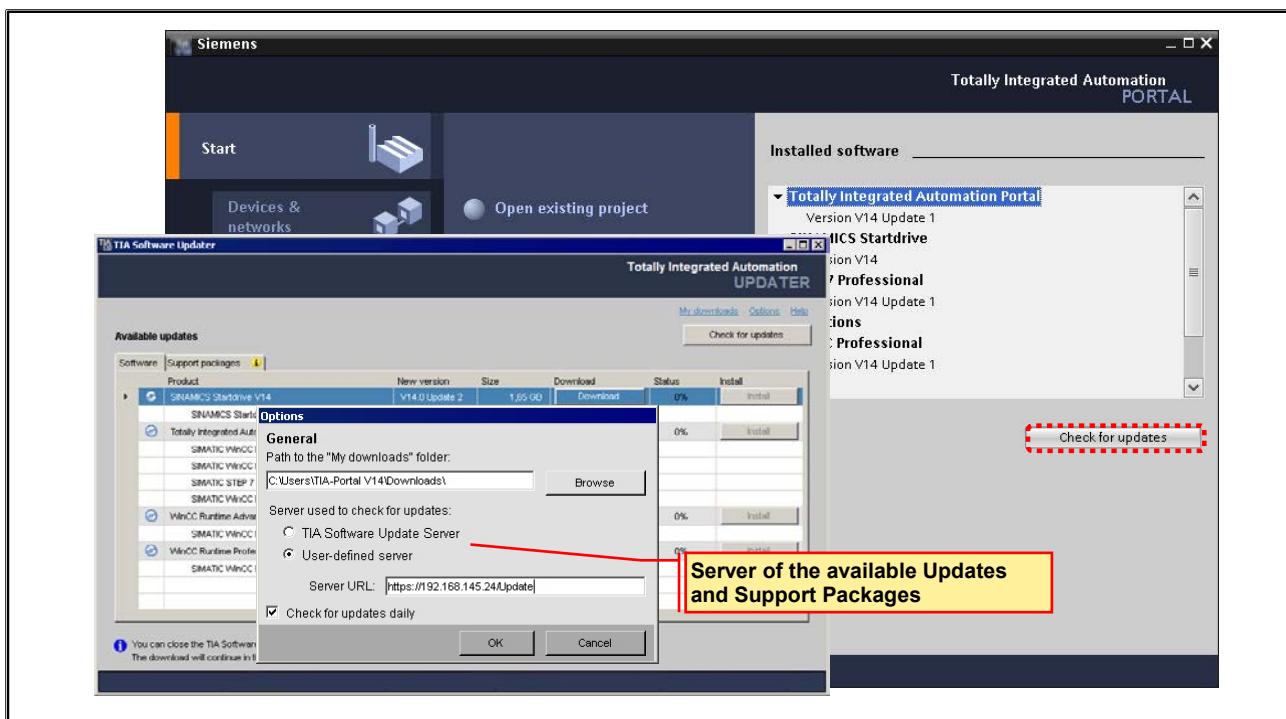


Further installation → Command lines installation (Silent installation)



With the command "setup.exe /record", an installation file is created during installation. With the help of this file, exactly the same installation can be carried out on other computers. For this, the installation file must be saved on the computer on which the installation is to be carried out and the installation must be started with the command setup.exe /qb {parameter} or setup.exe /qn {parameter}. {parameter} corresponds exactly to the path in which the installation file was saved. The installation is executed without further settings having to be made.

3.15.4. Update Tool



Update Tool

You can search for updates in the dialog "Installed software".

- Start via: Portal view: Start > Installed software > Check for updates
Project view: Help > Installed software > Check for updates
- Checks and informs about possible updates of installed software
- Download of updates
- Pause / Continue downloads
- Installation of Updates

Options:

- **Location for Download Files**

Server used to check for updates.

You can choose either the TIA Automation Software Update Server or a user-defined server on which the updates are provided by an Administrator.

- **Check for updates daily**

3.15.5. STEP 7 Licensing

	S7-1200 (LAD, FBD, SCL)	Engineering System (ES) S7-300 / S7-400 / S7-1500 / WinAC (LAD, FBD, STL, SCL, S7-GRAF, PLCSIM)
STEP 7 Basic	 Single license 	---
STEP 7 Professional	 Floating license   Upgrade license  +  generates Combo license → STEP 7 V5.4 (and higher) and → STEP 7 V1x Professional  parallel installation possible	
SIMATIC Startdrive	No license required	

To program the controllers, only one license (engineering license) is required in the engineering.
The program does not require any further licenses on the CPU (runtime license).

3.15.6. WinCC Licensing

	Engineering System (ES)	Runtime (RT)
WinCC Basic	 --- (Component of STEP 7) Floating license 	--- (Component of the Panels)
WinCC Comfort	 Floating license   Upgrade license  +  generates Combo license → WinCC flexible 2008 Standard and → WinCC Comfort  parallel installation possible	--- (Component of the Panels)
WinCC Advanced	 Floating license   Upgrade license  +  generates Combo license → WinCC flexible 2008 Advanced and → WinCC Advanced  parallel installation possible	 Single license   Upgrade license  → 
WinCC Professional	 Floating license   Upgrade license not available, since another license model exists for WinCC V7 (and higher)  Parallel installation WinCC V7 with WinCC Professional currently not yet possible	 Single license   Upgrade license not available, since another license model exists for WinCC V7 (and higher)

Contents

4

4.	Devices & Networks: Online Functions and Hardware Configuration.....	4-2
4.1.	Setpoint and Actual Configuration	4-3
4.2.	Online Tools, Configuring and Parameterizing the Hardware	4-4
4.3.	Online Connection via Industrial Ethernet: IP Address and Subnet Mask	4-5
4.4.	Default for Online Access and Visible Interfaces.....	4-6
4.4.1.	Online Access: Accessible Devices	4-7
4.4.1.1.	Accessible Devices: Online & Diagnostics, Task Card: Online Tools	4-8
4.4.1.2.	Accessible Devices: Online & Diagnostics: Diagnostics Buffer	4-9
4.4.1.3.	Accessible Devices: Online & Diagnostics: IP Address, Name, Time, FW Update, Memory Card	4-10
4.5.	Resetting the CPU using the Mode Selector Switch	4-11
4.5.1.	SIMATIC S7-1200/1500: Memory Concept for CPU Memory Reset.....	4-12
4.5.2.	SIMATIC S7-1200/1500: Memory Concept for CPU Reset to Factory Settings.....	4-13
4.6.	Card Reader / USB Memory Device	4-14
4.7.	Working Areas of the Hardware and Network Editor.....	4-15
4.7.1.	Hardware and Network Editor: Device View.....	4-16
4.7.2.	Hardware Catalog	4-17
4.8.	Adding a New Device (Controller).....	4-18
4.8.1.	Selecting the Controller and the Modules.....	4-19
4.8.2.	CPU Properties: Ethernet Address	4-20
4.8.2.1.	CPU Properties: Maximum Cycle Time	4-21
4.8.2.2.	CPU Properties: System and Clock Memory	4-22
4.8.2.3.	CPU Properties: Password Protection.....	4-23
4.8.3.	Inserting / Deleting a Module	4-25
4.8.4.	Changing a Device / Module.....	4-26
4.8.5.	Compiling the Hardware / Software and Downloading it into the CPU.....	4-27
4.9.	Task Description: Creating a Project with an S7-1500 Station.....	4-28
4.9.1.	Exercise 1: Setting the IP Address of the PG	4-29
4.9.2.	Exercise 2: Erasing the SIMATIC Memory Card of the CPU	4-30
4.9.3.	Exercise 3: Resetting the CPU to Factory Settings using the Mode Selector Switch	4-31
4.9.4.	Exercise 4: Creating a New Project and Adding a New Device (Controller)	4-32
4.9.5.	Exercise 5: Changing the Device Name and Disabling the F-activation	4-33
4.9.6.	Exercise 6: Configuring the S7-1500 Station.....	4-34
4.9.7.	Exercise 7: CPU Properties: IP Address	4-35
4.9.8.	Exercise 8: CPU Properties: Parameterizing the Clock Memory Byte	4-36
4.9.9.	Exercise 9: CPU Properties: Display Language and Display Protection	4-37
4.9.10.	Exercise 10: Addresses of the DI Module.....	4-38
4.9.11.	Exercise 11: Addresses of the DO Module	4-39
4.9.12.	Exercise 12: Addresses of the AI Module	4-40
4.9.13.	Exercise 13: Setting the Channel Parameters of the Analog Input Module	4-41
4.9.14.	Exercise 14: Compiling the Device Configuration and Downloading it into the CPU	4-42
4.9.15.	Exercise 15: Setting the Time and Trying to Switch the Controller to RUN Mode	4-44
4.10.	Additional Information	4-45
4.10.1.	Swapping a Slot / Inserting a Module between Two Modules	4-46
4.10.2.	Copying Modules from a Reference Project.....	4-47
4.10.3.	Unspecified CPU.....	4-48
4.10.4.	'View' Settings of the Task Cards	4-49

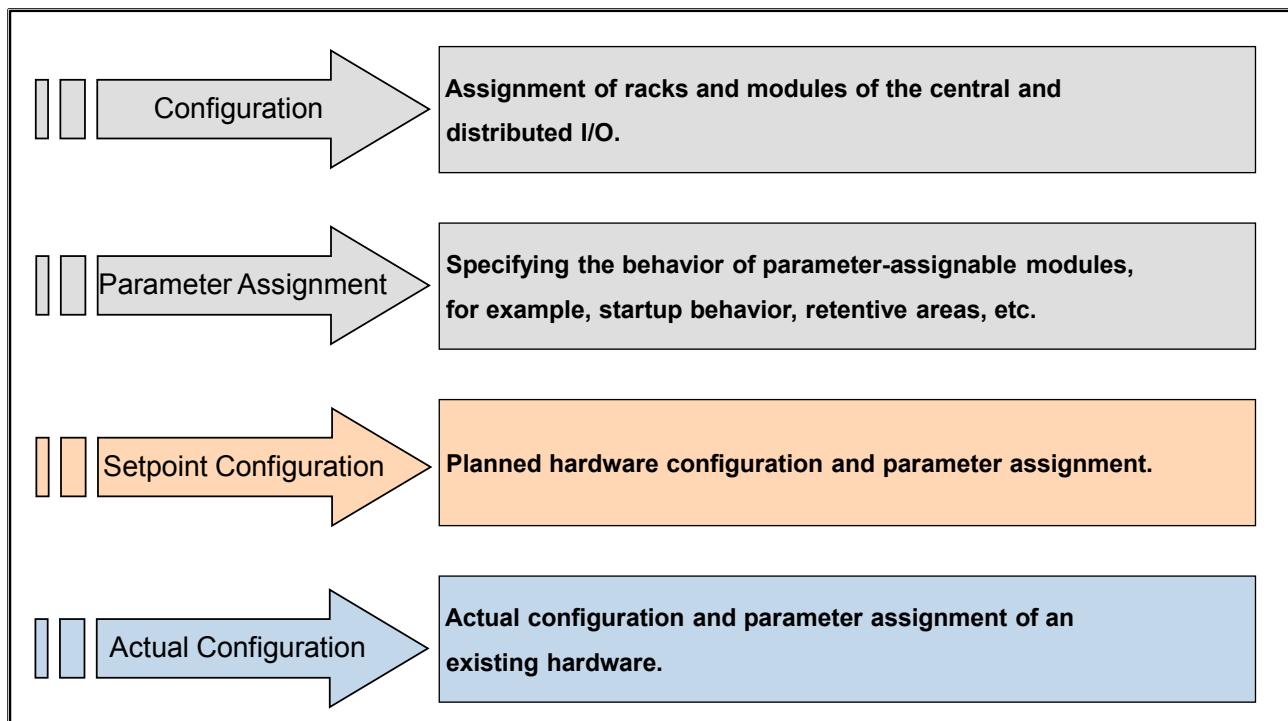
4. Devices & Networks: Online Functions and Hardware Configuration

At the end of the chapter the participant will ...



- ... be able to establish an online connection between PG and CPU via Industrial Ethernet
- ... be able to use online functions to start and stop the CPU and to reset it to factory settings
- ... be able to create and parameterize a new station
- ... be able to create and parameterize a setpoint (offline) configuration
- ... be familiar with addressing the input and output modules of an S7-1500 and be able to do it

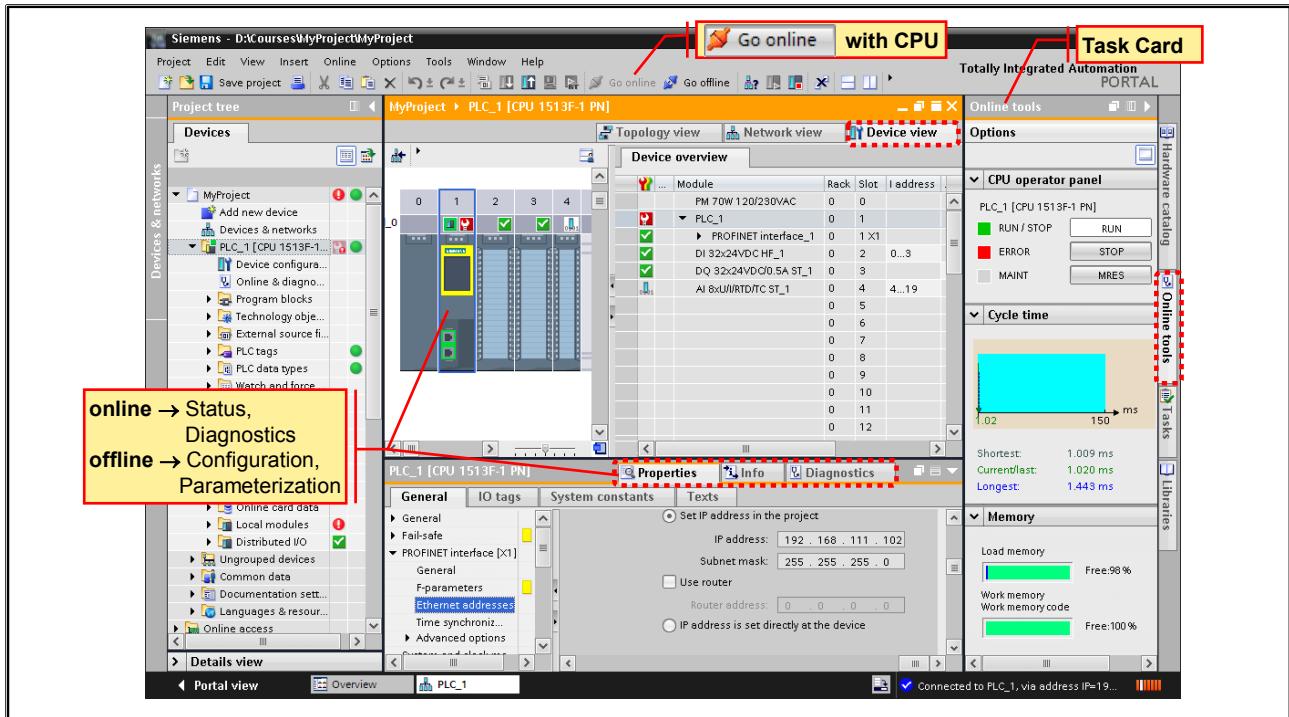
4.1. Setpoint and Actual Configuration



Setpoint (Offline) and Actual Configuration

When you configure a system, a setpoint (offline) configuration is created. It contains a hardware station with the planned modules and the associated parameters. The PLC system is assembled according to the setpoint (offline) configuration. During commissioning, the setpoint (offline) configuration is downloaded to the CPU.

4.2. Online Tools, Configuring and Parameterizing the Hardware



Online Tools

If it is possible to establish an online connection to the CPU, diagnostics and status information of all modules can be called.

With CPUs that can be accessed online, the mode can also be controlled using the "Online tools" task card and further status information (cycle time statistics and memory load) can be called.

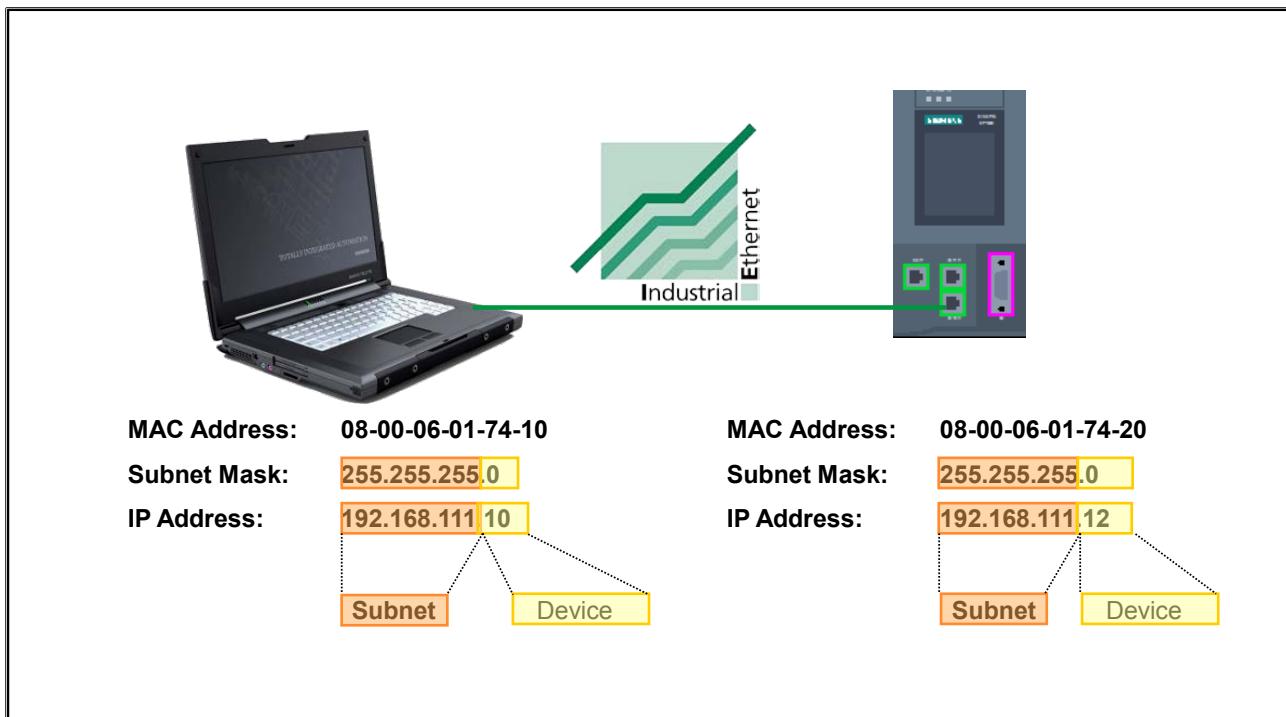
Configuring and Parameterizing the Hardware

Almost all devices or components of an automation solution such as PLCs or touch panels can be assigned parameters. The parameter assignment of the devices and network settings required for commissioning is handled using the Hardware and Network Editor.

With this, for example, all components of an Ethernet network are assigned IP addresses via which they communicate during later operation.

But even inside the automation device, address areas of the I/O modules must be specified and the cycle monitoring time of the CPU must be set, for example.

4.3. Online Connection via Industrial Ethernet: IP Address and Subnet Mask



Internet Protocol

The Internet Protocol (IP) is the basis for all TCP/IP networks. It creates the so-called datagrams (data packets specially tailored to the Internet protocol) and handles their transport within the local subnet or their "routing" (forwarding) to other subnets.

IP Addresses

IP addresses are not assigned to a specific computer, but rather to the network interfaces of the computer. A computer with several network connections (for example routers) must therefore be assigned an IP address for each connection.

IP addresses consist of 4 bytes. With the dot notation, each byte of the IP address is expressed by a decimal number between 0 and 255. The four decimal numbers are separated by dots (see picture).

MAC Address

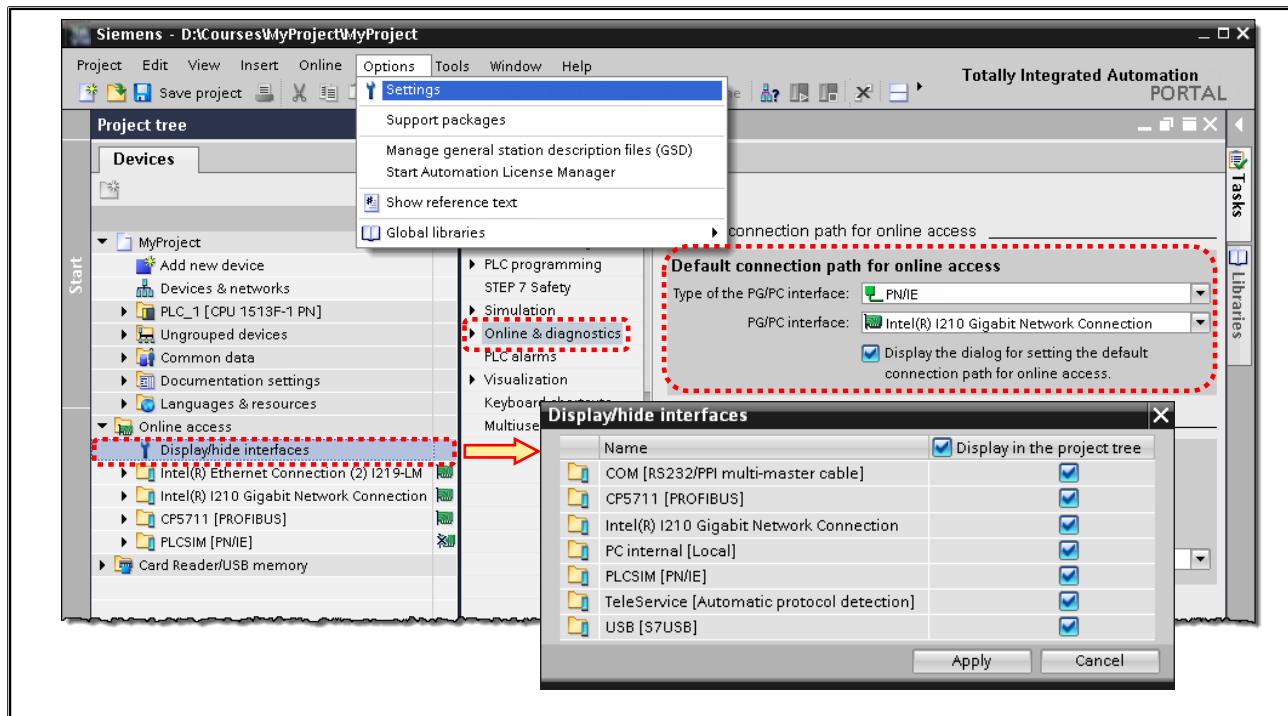
Every Ethernet interface is assigned a fixed address by the manufacturer that is unique worldwide. This address is referred to as the hardware or MAC address (Media Access Control). It is stored on the network card and uniquely identifies the Ethernet interface in a local network. Cooperation among the manufacturers ensures that the address is unique worldwide.

Subnet Mask

The subnet mask specifies which IP addresses in the local network can be accessed. It separates the IP address into the network and device part.

Only IP addresses whose network part is the same can be accessed.
e.g.: Subnet mask = 255.255.255.0 and IP address = 192.168.111.10
accessible IP addresses: 192.168.111.1 to 192.168.111.254

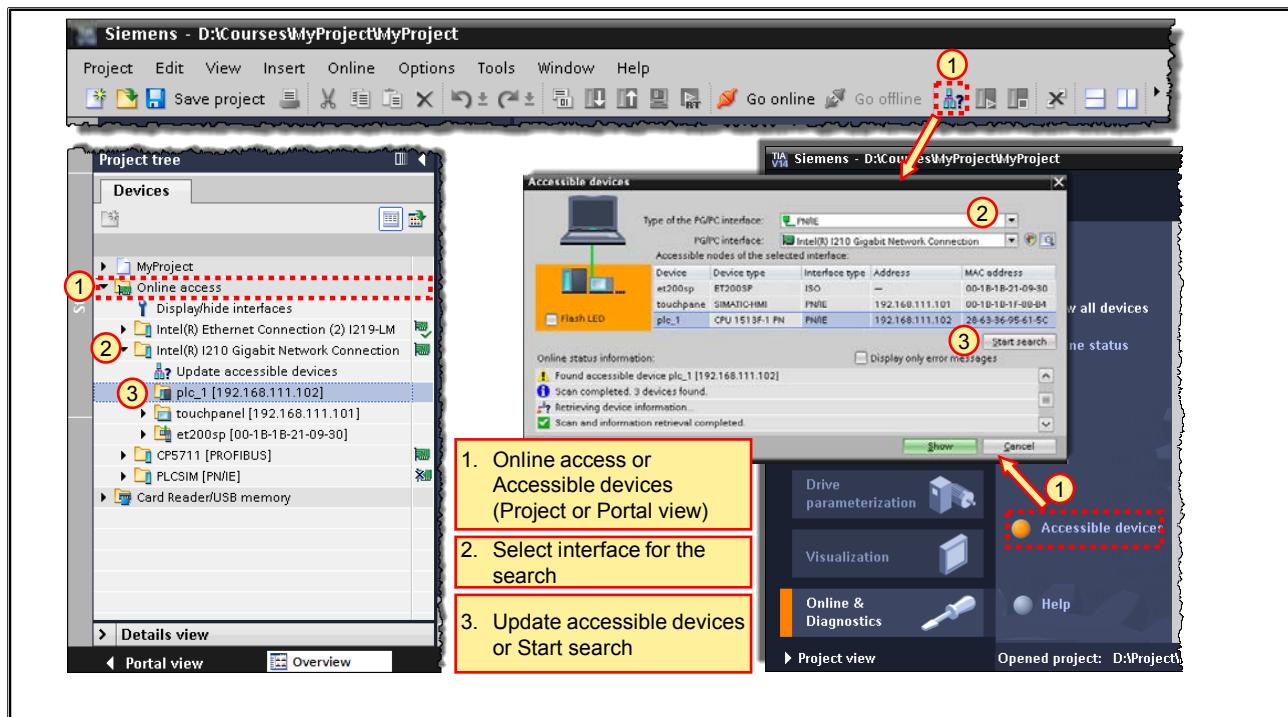
4.4. Default for Online Access and Visible Interfaces



In the Settings, you can have a default setting for the connection path for online access.

In the Online access folder, all possible interfaces of the PG/PC are displayed. Since not all of these are required or can be used, interfaces can be hidden for better clarity.

4.4.1. Online Access: Accessible Devices



Accessible Devices in the Portal View

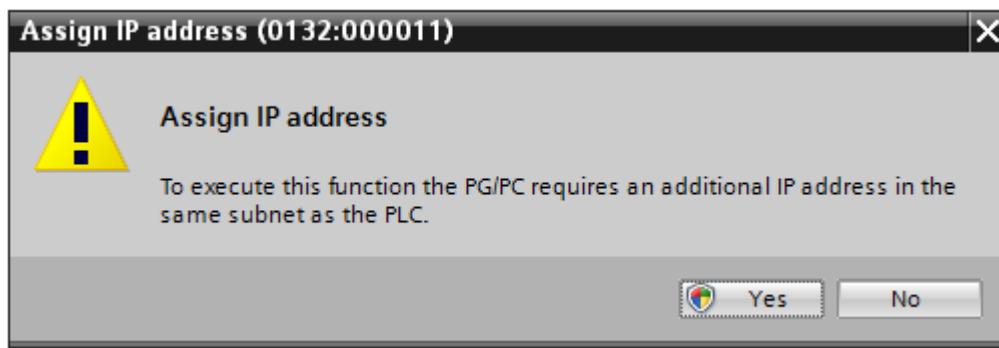
This function provides the option of fast access (for example for service purposes) even when **there is no offline project data for the target systems on the PG**.

All accessible, programmable modules (CPUs, FMs CPs, HMI devices) are listed in the Portal view, even if they are located in other subnets.

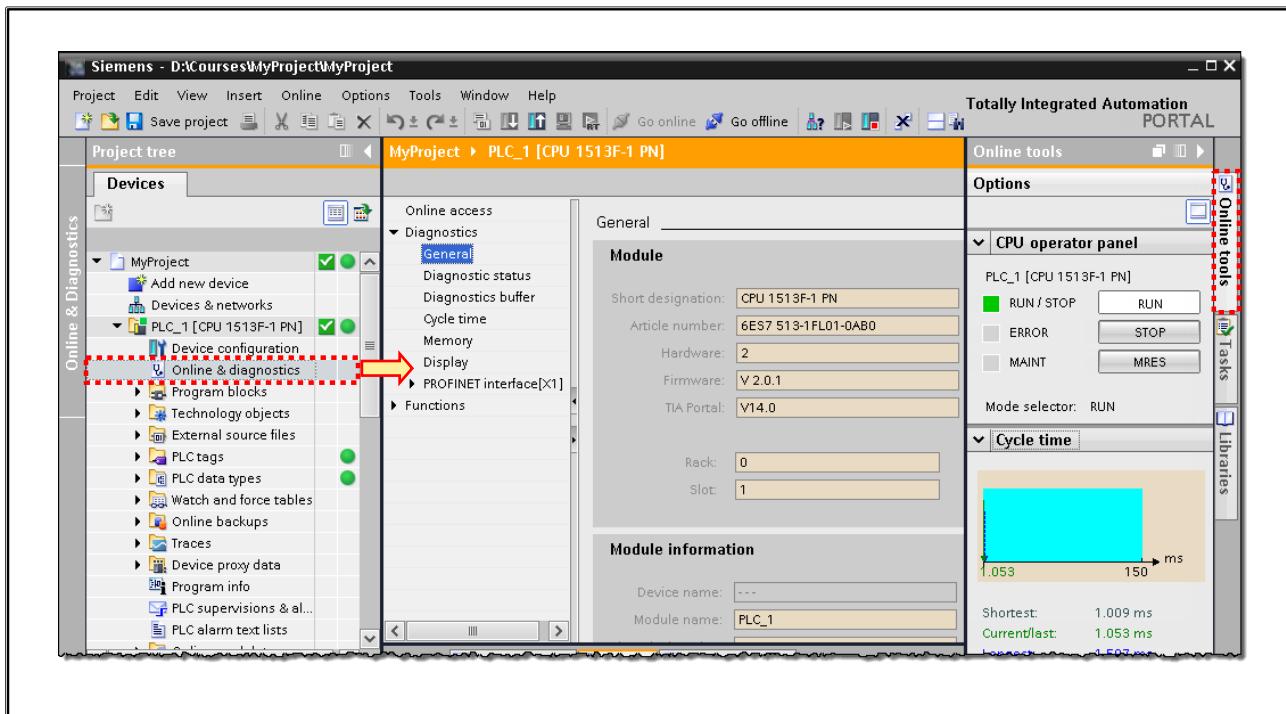
Access Online Functions → **Show** Button

Whenever there is an attempt to access a module online with the "Show" button and this is located in a different subnet from the PG, a dialog opens asking whether an additional IP address should be assigned to the PG.

Following confirmation, an additional IP address is assigned to the PG that is located in the same subnet as the address of the CPU. After that, all online functions can be used.



4.4.1.1. Accessible Devices: Online & Diagnostics, Task Card: Online Tools



CPU Operator Panel: Mode Selector Switch

The operating mode of the CPU can be changed.

- **RUN → STOP:**
If there is a change from RUN to STOP, the CPU terminates the running user program.
- **STOP → RUN:**
If there is a change from STOP to RUN, the CPU performs a restart.

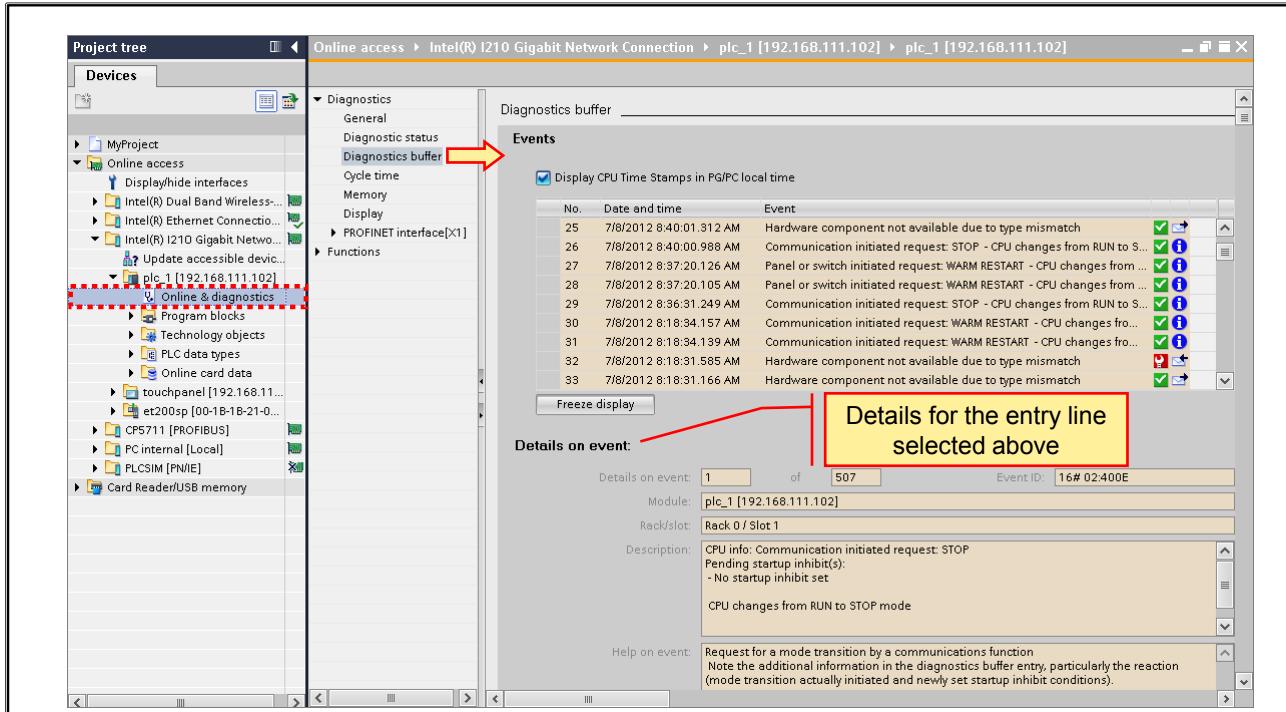
Cycle Time:

"Shortest", "Current" and "Longest" are the cycle times since the last CPU restart

With a Memory Reset (MRES), a CPU reset is carried out:

- All user data (even the retentive) is deleted (delete work memory)
(process images, memory bits, timers, counters, all program/data blocks)
- Retained are: IP addresses, the retentive part of the diagnostics buffer, operating hours counter, time-of-day.

4.4.1.2. Accessible Devices: Online & Diagnostics: Diagnostics Buffer



Online Access to the CPU

If the PG and the target system (for example CPU) are located in the same subnet, various Online & diagnostics functions are available in the "Accessible devices" function.

- in the working area of the TIA Portal
- in the "Online tools" task card (see next page)

Diagnostics Buffer

The diagnostics buffer is a buffered memory area on the CPU organized as a circular buffer. It contains all diagnostics events (error alarms, diagnostics interrupts, start-up information etc.) of the CPU in the order in which they occurred. The highest entry is the last event to occur.

All events can be displayed on the programming device in plain language and in the order in which they occurred.

All events can be displayed on the programming device in plain language and in the order in which they occurred. In addition, not the entire diagnostics buffer is buffered with Power OFF (only a part is retentive).

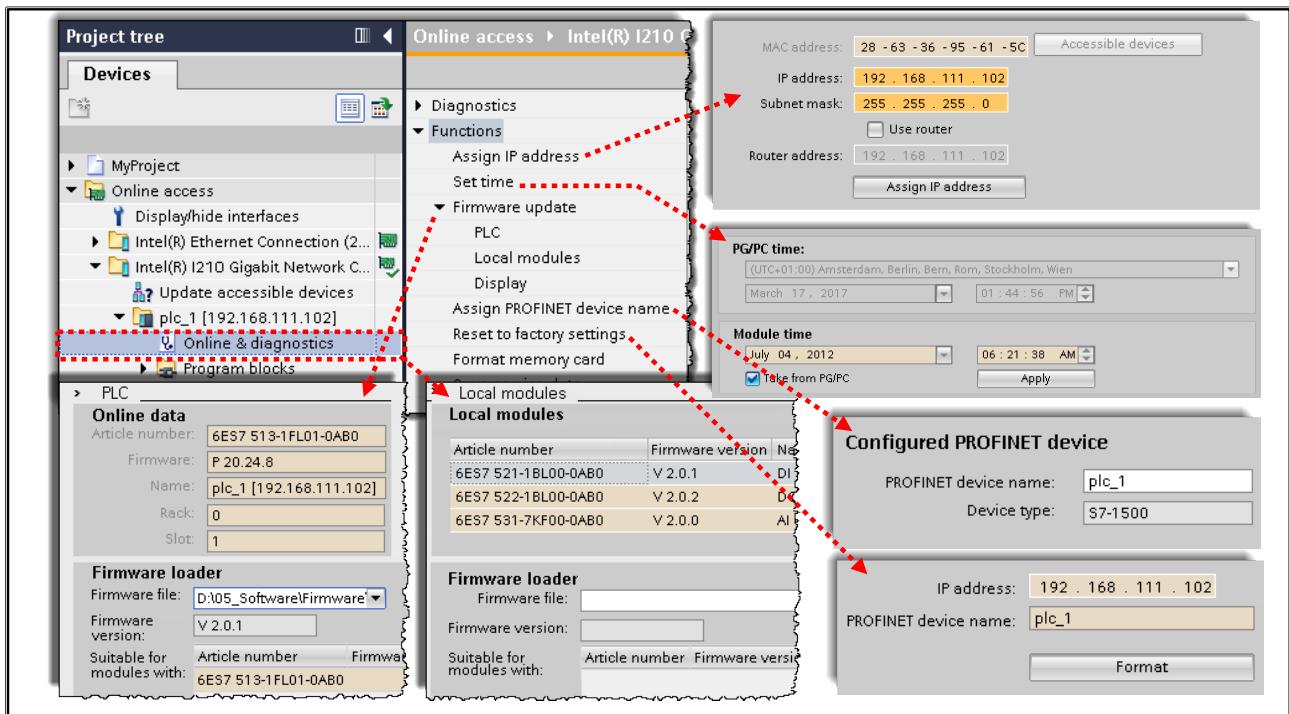
- Number of entries, 1000 to 3200
- Of that, retentive 250 to 500

Details on Event

Some additional information is also provided for the selected event in the "Details on event" box:

- Event name and number
- Additional information depending on the event, such as, the address of the instruction that caused the event etc.

4.4.1.3. Accessible Devices: Online & Diagnostics: IP Address, Name, Time, FW Update, Memory Card



- **Set Time (of Day)**

Each S7 CPU has a real-time clock that can be set here.

- **Assign IP Address**

As long as no IP address has been specified already by a hardware configuration that was downloaded earlier, this can be assigned or modified here (this function is also available when the PG/PC and the CPU are not assigned to the same subnet).

- **Reset to Factory Settings**

Unlike the "memory reset", all the memory areas of the CPU (work, load and retentive memory, diagnostics buffer and time) are deleted. Optionally (see dialog in the picture), the IP address can also be deleted so that the CPU then only has a MAC address (Media Access Control).

- **Format Memory Card**

The CPU memory card can also be deleted in the CPU via this online function. After that, the CPU only has its IP address. All other data (including the device configuration) is deleted.

The card cannot be deleted in the card reader via the Project tree. Device configuration and blocks have a gray background, that is, are write-protected (only status information or open with a double-click).

- **Assign Name**

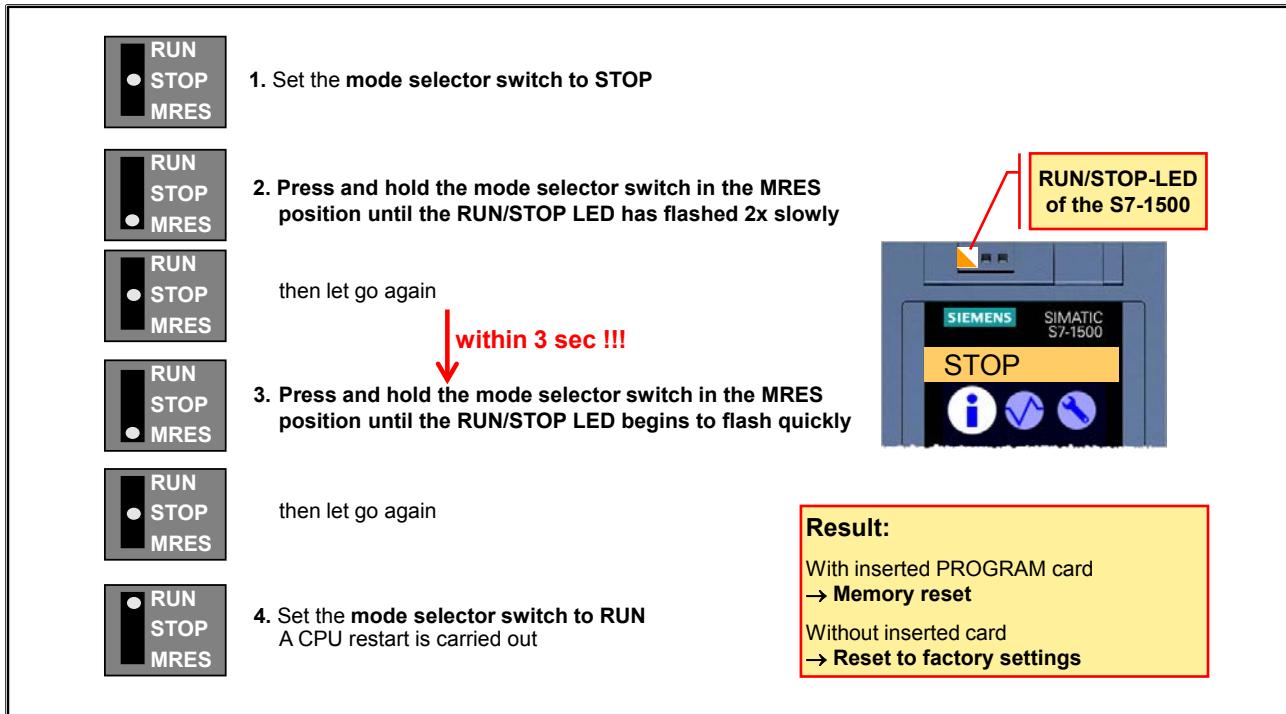
In PROFINET, each device must be assigned a unique device name that is stored retentively on the device. The device name identifies a distributed I/O module (PROFINET IO) and allows module replacement without a PG/PC.

- **Firmware Update**

The firmware version of the device and the modules can be updated. Under "Diagnostics -> General", the current firmware version is displayed.

Caution: If the CPU and Display have to be updated, first the Display and then the CPU.

4.5. Resetting the CPU using the Mode Selector Switch



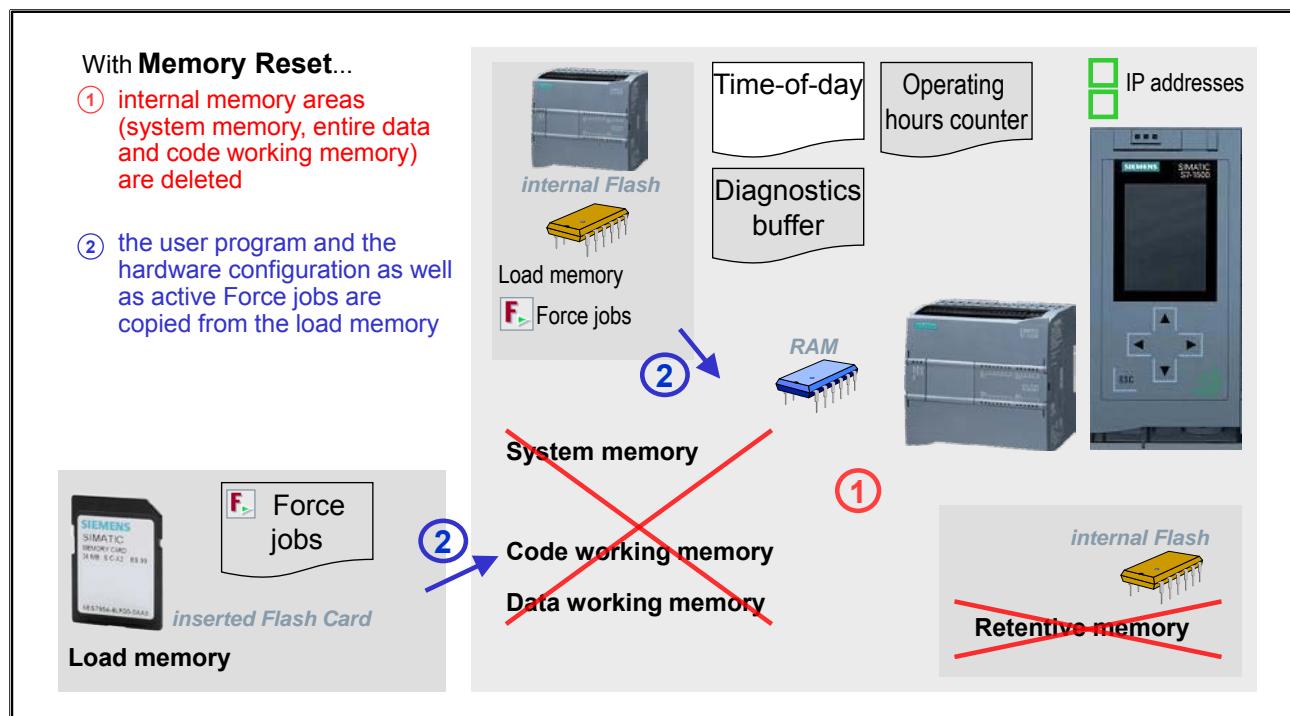
Particular Feature for CPU Memory Reset (MRES) using the Mode Selector Switch:

- when SIMATIC Memory Card (SMC) is inserted => Memory Reset
 - All user data is deleted (work memory, retentive memory) (process images, memory bits, counters, timers, all program/data blocks)
 - Retained are: parameter assignment of the X1 (Ethernet) interface, the retentive part of the diagnostics buffer, operating hours counter, CPU time-of-day
 - The CPU copies all load memory data relevant for execution (memory card) into the internal RAM work memory. (Data relevant for execution: device configuration, program blocks, data blocks).
- when no SIMATIC Memory Card (SMC) is inserted => Reset to factory settings
 - All memory areas of the CPU (work memory, retentive memory, diagnostics buffer, time-of-day) and the IP address are deleted.

After the SMC is inserted, the load memory data relevant for execution is reloaded into the internal RAM work memory from the memory card:

Device configuration (with IP address), program blocks, data blocks

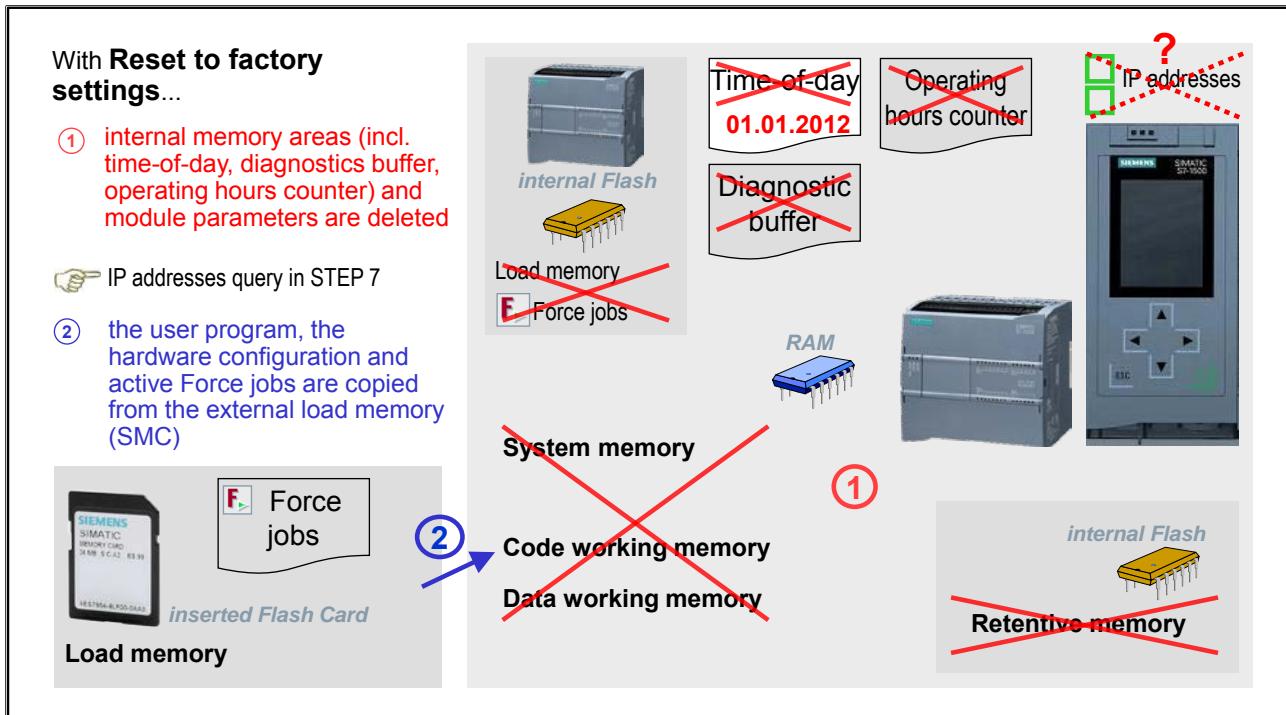
4.5.1. SIMATIC S7-1200/1500: Memory Concept for CPU Memory Reset



CPU Memory Reset

- **What to do:**
 - STEP 7 online function → MRES in "CPU operator panel" of "Test" and "Online tools" Task Cards
 - Display (only S7-1500) → Main menu "Settings", submenu "Memory reset"
 - CPU mode selector switch (with inserted memory card)
- **Impact**
 - An existing online connection between PG/PC and the CPU is disconnected.
 - The entire RAM work memory is deleted, that is, all user data (process images, bit memories, counters, timers, all program/data blocks, even the retentive ones)
 - Retained are: IP addresses, diagnostic buffer, operating hours counter, CPU time-of-day.
 - After that, the CPU copies all data relevant for execution into the RAM work memory from the memory card. (Data relevant for execution: device configuration, program blocks, data blocks, current Force jobs).

4.5.2. SIMATIC S7-1200/1500: Memory Concept for CPU Reset to Factory Settings

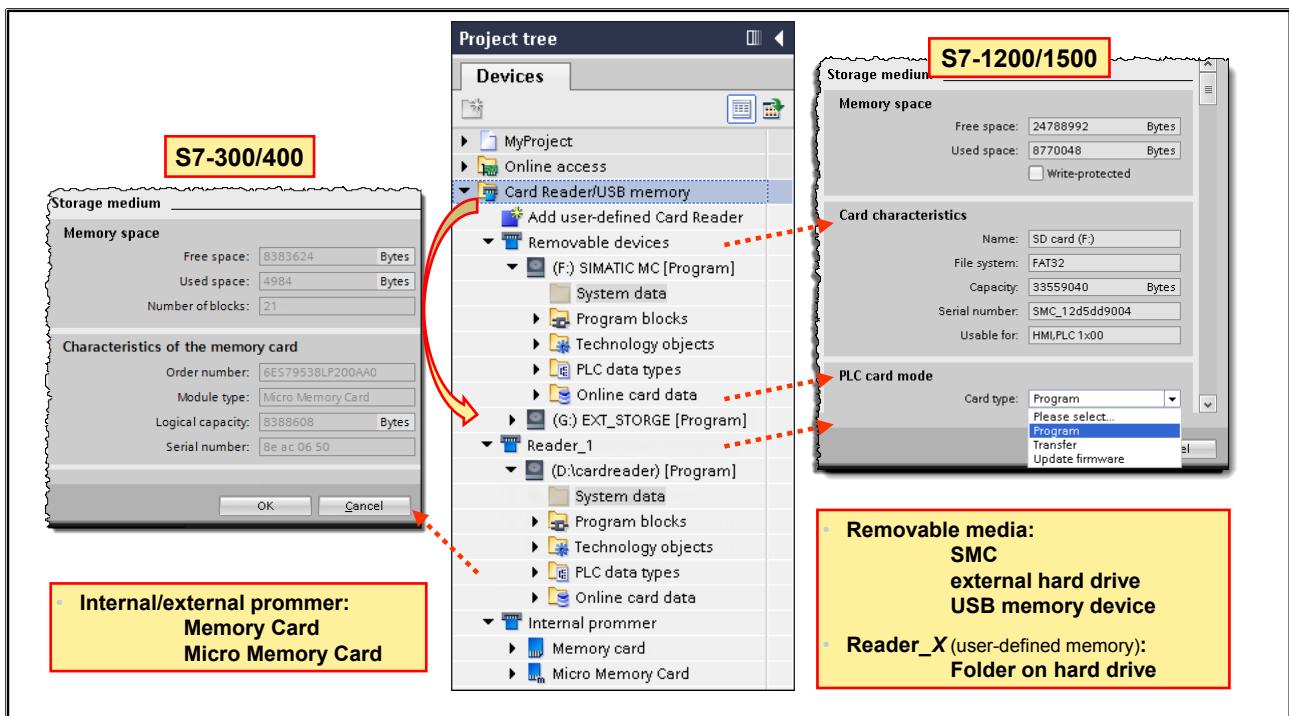


CPU Reset to Factory Settings

- **What to do:**
 - STEP 7 online function → MRES in "CPU operator panel" of "Test" and "Online tools" Task Cards
 - Display (only S7-1500) → Main menu "Settings", submenu "Memory reset" → Factory Defaults
 - Mode selector switch (only without memory card)
- **Impact**
 - An existing online connection between PG/PC and the CPU is disconnected.
 - The entire RAM work memory is deleted, that is, all user data (process images, bit memories, counters, timers, all program/data blocks, even the retentive ones, diagnostic buffer), IP addresses are deleted if this is selected in STEP 7.
 - All IP addresses are retained if this was specified in STEP 7.

If a memory card is inserted (or is already inserted), the CPU copies all data relevant for execution into the internal RAM work memory from the memory card. (Data relevant for execution: device configuration incl. IP address, program blocks, data blocks, current Force jobs).

4.6. Card Reader / USB Memory Device



Card Reader / USB Memory

In the Card Reader/USB memory folder, you can access an SMC inserted in the SD Reader, the internal/external prommer, removable (media) devices or user-defined folders.

Card Type of the SIMATIC Card for S7-1200/1500:

The SIMATIC Memory Card is used as a Program card or a Transfer card or for Firmware Updates. Before the relevant data is stored on the SMC, the card type must be selected as shown in the picture.

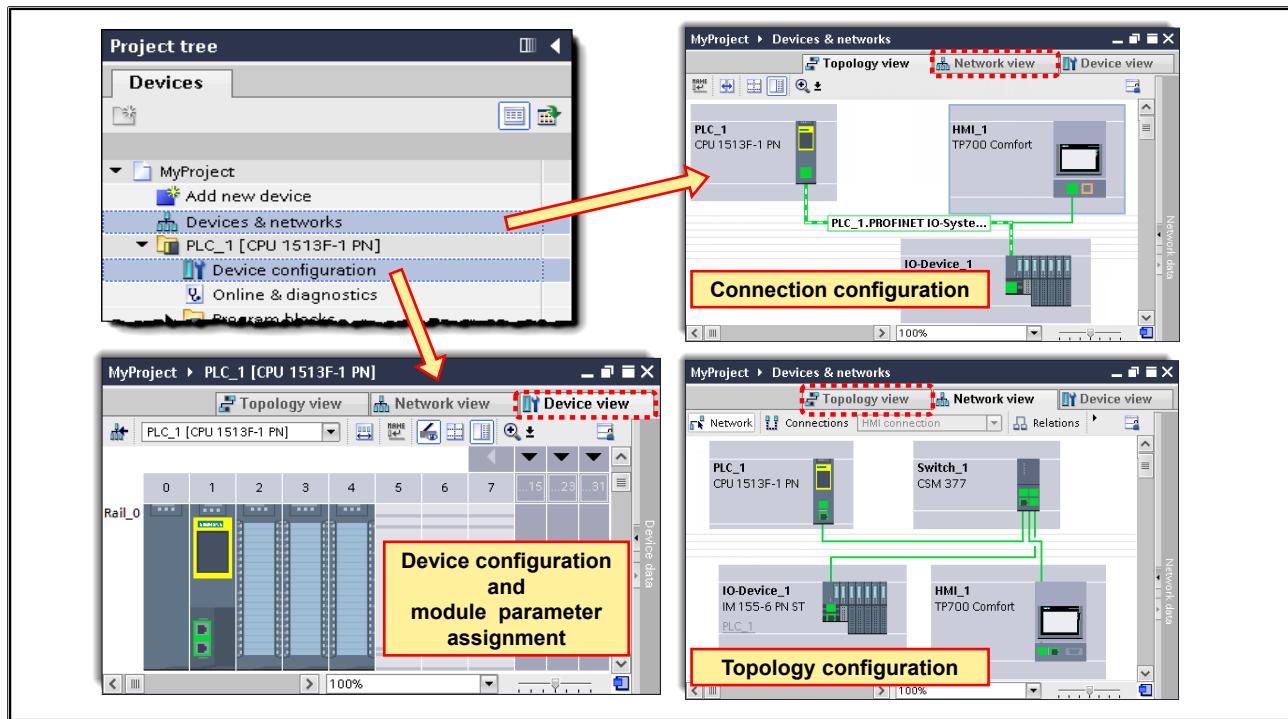
- **SIMATIC Memory Card as Program Card:**
The card contains all configuration and parameterization data for the station as well as the entire user program with documentation. During operation, the card must remain inserted in the CPU because it is used as a replacement for the internal CPU load memory which remains unused.
- **SIMATIC Memory Card as Transfer Card (only for S7-1200):**
The card contains the same data as a Program card but it doesn't have to remain inserted during operation. After inserting the card and subsequent Power ON, all data is copied into the internal load memory of the CPU. Then the card has to be removed and a restart has to take place.
- **SIMATIC Memory Card to Update Firmware:**
The SIMATIC Memory Card contains the files required for a firmware update. After execution, the SIMATIC Memory Card must be removed.

S7-300/400:

An S7-300 or S7-400 CPU does not have an SMC as load memory but a Memory Card or Micro Memory Card. You can only access these cards with the help of an internal or external prommer.

Note: A SIMATIC Field PG has an internal prommer.

4.7. Working Areas of the Hardware and Network Editor



Components of the Hardware and Network Editor

The Hardware and Network editor consists of a Device, Network and Topology view.

Device View

The Device view is used for configuring and parameterizing devices and modules.

- Hardware configuration
- Device and module parameter assignment

Network View

The Network view is used for configuring, parameterizing and networking devices.

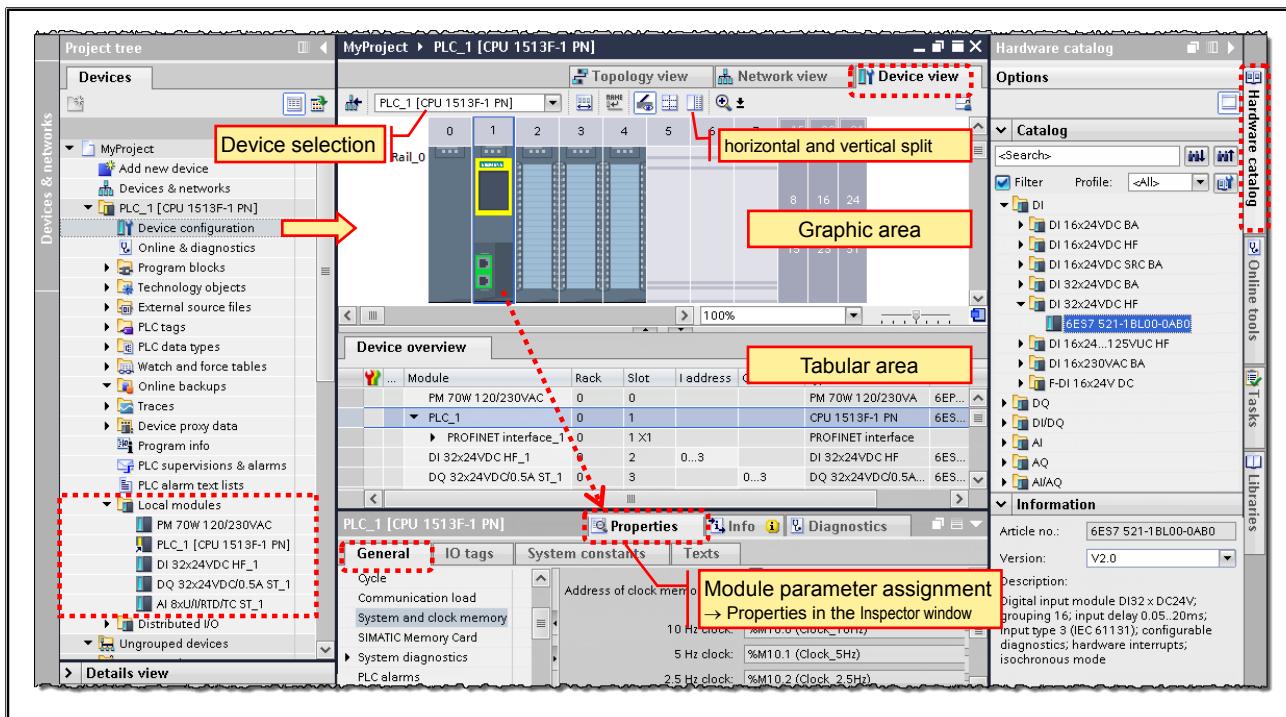
- Configure and parameterize devices
- Connection configuration

Topology View

The Topology view is used for displaying, configuring and determining the physical structure of networks.

- Configure the port assignment and the relationship between devices
- Online-Offline comparison as well as synchronization of the port assignment and relationships
- Topology makes it possible to exchange devices without a node initialization

4.7.1. Hardware and Network Editor: Device View



Components of the "Hardware and Network Editor"

"Device view" section in the working area

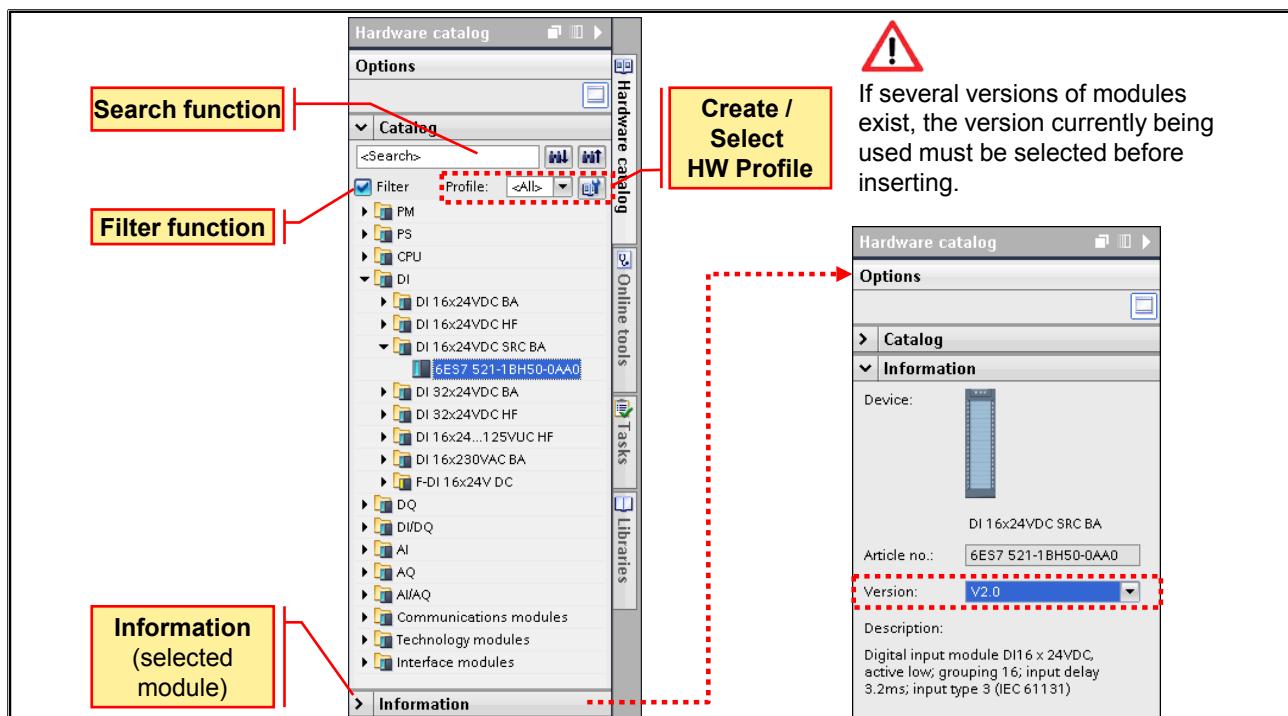
This editor consists of 2 areas, a tabular (left/top) and a graphic (right/bottom). The splitting left-right or top-bottom can be changed as required.

- Graphic area = module configuration
- Tabular area = Address parameterization of configured modules
- **"Properties" tab in the Inspector window**
This tab is used to assign parameters to the module selected in the working area. Here, all the properties or parameters of the selected module are displayed and can also be modified. In the left-hand part of the Properties tab there is a navigation section in which the parameters are arranged in groups.
- **"Hardware Catalog" Task Card**
Module catalog for the configuration (module grouping) in the working area

Project Tree → "Local Modules"

In the Project tree, the modules along with their parameter assignments (for example, addresses) are stored under the relevant device in the "Local modules" folder.

4.7.2. Hardware Catalog



The Hardware catalog contains all devices and hardware components in a tree structure. From the catalog, selected devices or modules can be dragged to the graphic working area of the "Hardware and Network" editor.

Search Function

This allows a convenient search for specific hardware components. The search also includes the module description texts.

Filter Function

Filter enabled: Only modules that match the current context are displayed.

Filter disabled: All existing objects of the catalog are displayed

Contents of the Hardware Catalog for Enabled Filter

- Network view → only objects that can be networked
- Device view → all modules or, for enabled filter, only the modules that belong to the current device in the working area

Profile

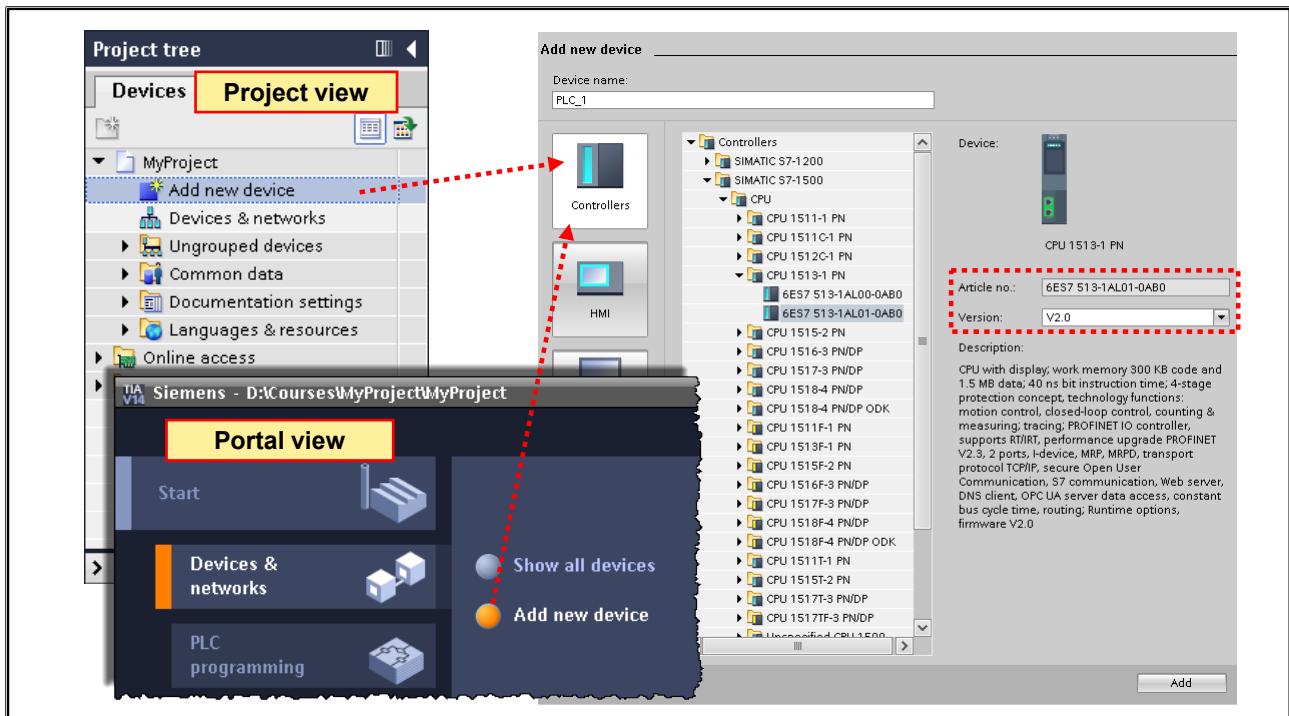
It is possible to create and to use your own profiles. This expands the filter possibilities.

Information

The "Information" pane shows detailed information about the object selected in the catalog.

- Name
- Order number (Article no.)
- Version number

4.8. Adding a New Device (Controller)



Add New Device

It is possible to create a new device in the project using the Hardware and Network editor with the help of the "Hardware catalog" task card or through the Project tree "Add new device".

When a new device is created, a suitable rack is also created automatically. The selected device is inserted into the first permitted slot in the rack.

Regardless of the method selected, the added device is visible in the Device view and in the Network view of the Hardware and Network editor.

4.8.1. Selecting the Controller and the Modules

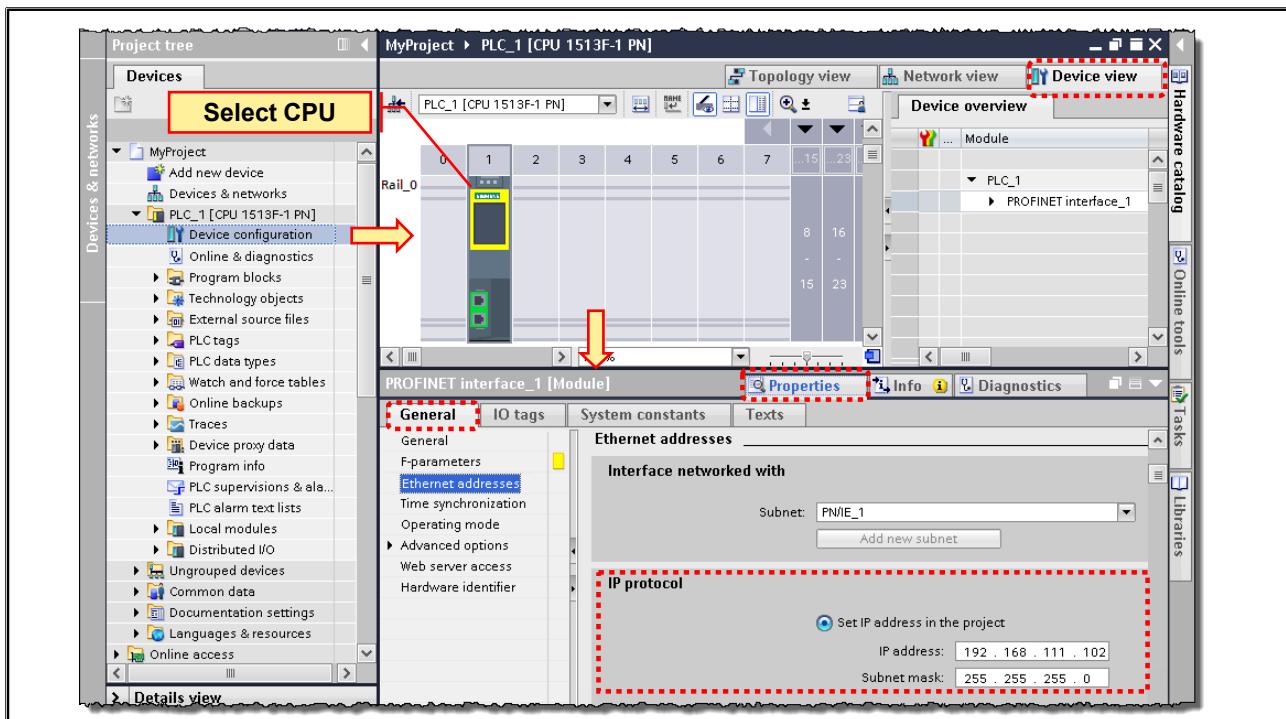
<p>Manual: "SIMATIC S7-1500 / ET 200MP Automation system In a nutshell"</p> <pre> graph TD A[Do you need a CPU for safety applications?] -- No --> B[Do you need Motion Control functions such as synchronous operation and ramps beyond the standard functionality?] B -- No --> C[Do you want to program the CPU in C/C++?] C -- Yes --> D[Do you need 1, 2 or 3 PROFINET interfaces?] D -- No --> E[CPU 1511C-1 PN 1 MB] D -- Yes --> F[At the CPUs, select the variant technology (T)] F --> G[CPU 1512C-1 PN 1 MB] G --> H[CPU 1511(F)-1 PN 1 MB] H --> I[CPU 1513(F)-1 PN 1 MB] I --> J[CPU 1511T-1 PN 1 MB] J --> K[Do you need a PROFIBUS DP interface? At the CPUs, select the variant PN/DP] K --> L[CPU 1515(F)-2 PN 3 MB] L --> M[2 interfaces 3 ports] M --> N[CPU 1518(F)-4 PN/DP-ODK 20 MB] N --> O[3 interfaces 4 ports] O --> P[CPU 1518(F)-4 PN/DP-ODK 20 MB] P --> Q[At the CPUs, select the variant technology (T)] Q --> R[CPU 1518(F)-4 PN/DP-ODK 20 MB] R --> S[1 MB] S --> T[1.5 MB] T --> U[1.5 MB] U --> V[1 MB] V --> W[1 MB] W --> X[Select the compact version (C) among the CPUs] X --> Y[Select the fail-safe version (F) among the CPUs] </pre>	<p>Software: "TIA Selection Tool" (online or offline)</p>
Entry ID: 109481357	TIA Portal Information Center > Tools & Apps > Configurators

SIMATICS7-1500 provides you with a wide range of CPUs that can be integrated. You can expand each CPU with I/O modules, communication modules and technology modules. If, for example, the memory and performance of a CPU 1511-1 PN are sufficient for you, then you expand it with communication modules for PROFINET and PROFIBUS. For technology functions, technology CPUs and technology modules are available in addition to the Compact CPUs.

To select the correct controller there is the manual **"SIMATIC S7-1500 / ET 200MP Automation system In a nutshell"** which contains further useful guidelines. It can be found under the Entry ID: 109481357.

There is also the software **TIA Selection Tool** which provides an opportunity for selecting, configuring and ordering the devices for Totally Integrated Automation. After configuring the hardware in the TIA Selection Tool, you are given a list with all hardware components which are required (modules, plugs, cables, profile rails etc.).
In addition, the order via the Industry Mall can be started directly from the TIA Selection Tool.

4.8.2. CPU Properties: Ethernet Address

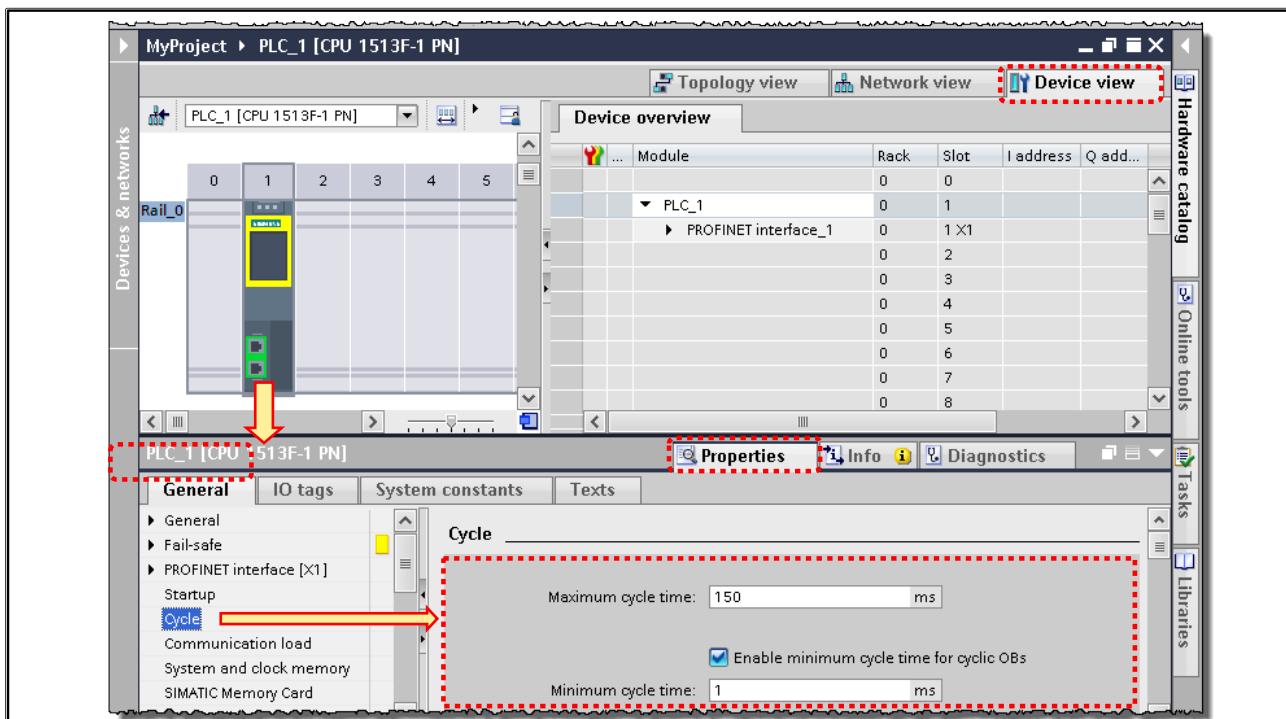


PROFINET Interface

Regardless of whether the editor is in the Device view or Network view, if the CPU is selected, the settings of the CPU PROFINET interface can be made in the Inspector window in the "Properties" tab.

 If an online connection needs to be established between the programming device and CPU, both devices must be assigned the same subnet mask and the IP addresses must be located in the same subnet.

4.8.2.1. CPU Properties: Maximum Cycle Time



Cycle Time

This is the time that the CPU requires for one complete program execution, that is, one cycle. Since parts of the user program can also be processed conditionally and the program execution can also be interrupted (for example, by diagnostics interrupts, time interrupts, hardware interrupts etc.), the length of the cycle time is not the same in every cycle.

Maximum Cycle Time

The operating system monitors the runtime of the program for the configured upper limit. If the runtime of the program is longer than the "Maximum cycle time" set here

- ... the operating system calls the associated time error interrupt OB.
- ... the operating system enters the event in the diagnostics buffer.
- ... the operating system indicates the error on the error LED of the CPU.

Behavior when the maximum cycle time is exceeded:

- S7-1200:**
The CPU remains in RUN mode even if no time error interrupt OB is programmed.
- S7-1500:**
If no time error interrupt OB is programmed, the CPU changes to STOP mode.

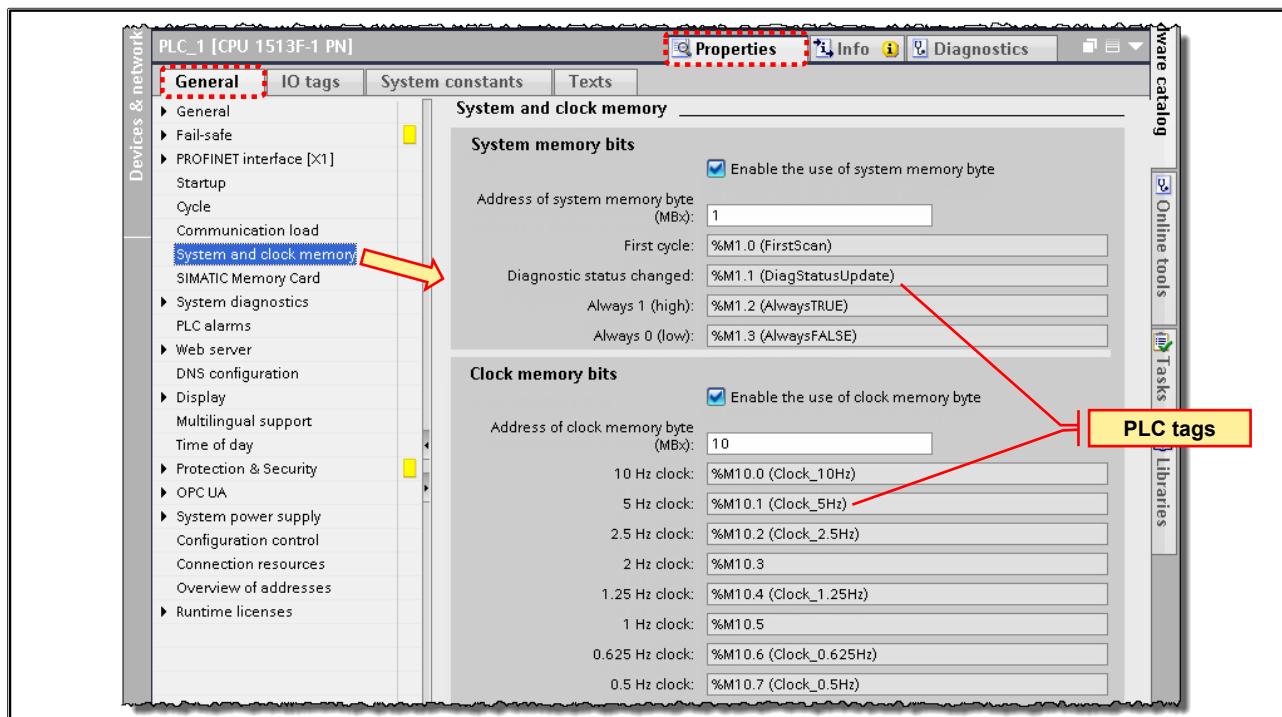
If the runtime of the S7-1200/1500 program is more than twice as long as the set maximum cycle time (2xMaxCycleTime error), the CPU changes to STOP mode without attempting to call the time error interrupt OB.

With the RE_TRIGR instruction, the monitoring of the cycle time can be retriggered or reset to 0.

Minimum Cycle Time

The minimum cycle time is the minimum time that should pass for the one-time execution of the cyclic user program and the updating of the associated I/O. The start of the next CPU cycle is delayed if this time has not yet expired.

4.8.2.2. CPU Properties: System and Clock Memory



A PLC tag is automatically created for each available system or clock memory bit.

System Memory (4 bits)

These are memory bits that provide system status information.

- "FirstScan" =1 in the first CPU cycle; otherwise =0,
- "DiagStatusUpdate" =1, if the diagnostic status has changed,
- One static 1-memory bit and 0-memory bit each ("AlwaysTRUE", "AlwaysFALSE"),

Clock Memory (8 bits)

These are memory bits whose binary state is changed periodically by the operating system of the CPU with a pulse-pause ratio of 1:1. The various frequencies are shown in the picture. Clock memory (bits) is used to trigger actions periodically.

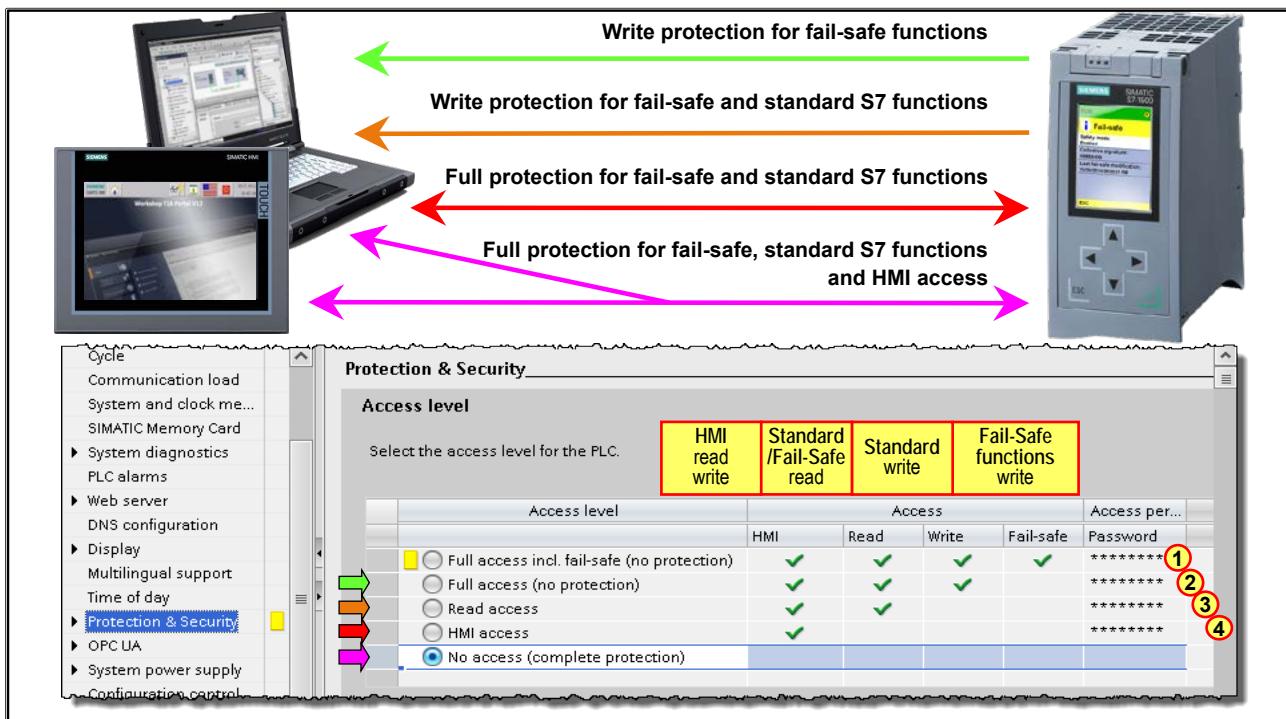
For example, to make an indicator light flash



Attention!

Clock memory (bits) are not synchronized with the CPU cycle; in other words, with long cycle times, the state of the clock memory (bits) can change more than once within one cycle.

4.8.2.3. CPU Properties: Password Protection



Protection Levels

With the following protection levels, the access rights (read / write) of the programming device to the CPU are specified:

- Full access incl. fail-safe (no protection): → Default setting for F-CPU
Read and write access is always permitted.
- Full access (no protection): → Default setting for non-F-CPU
Read access is always permitted, write access only to standard program.
- Read access: → Write protection
Read-only access possible. No data can be changed in the CPU and no blocks or modified hardware configuration or parameter assignment can be downloaded to the CPU without specifying a password.
- HMI access: → Write and read protection for STEP 7
No write or read access is possible from the engineering. Only the CPU type and identification data can be displayed in the Project tree under "Accessible devices". It is not possible to display online information or blocks under "Accessible devices" without entering a password.
- No access (complete protection): → General write and read protection for STEP 7 and HMI.
Now, access for HMI devices without a configured password in the connection is also not possible.

Access Permitted through Passwords

In the example shown, "No access (complete protection)" is selected. This means that without a password, STEP 7 and HMI devices can neither read-access nor write-access the CPU.

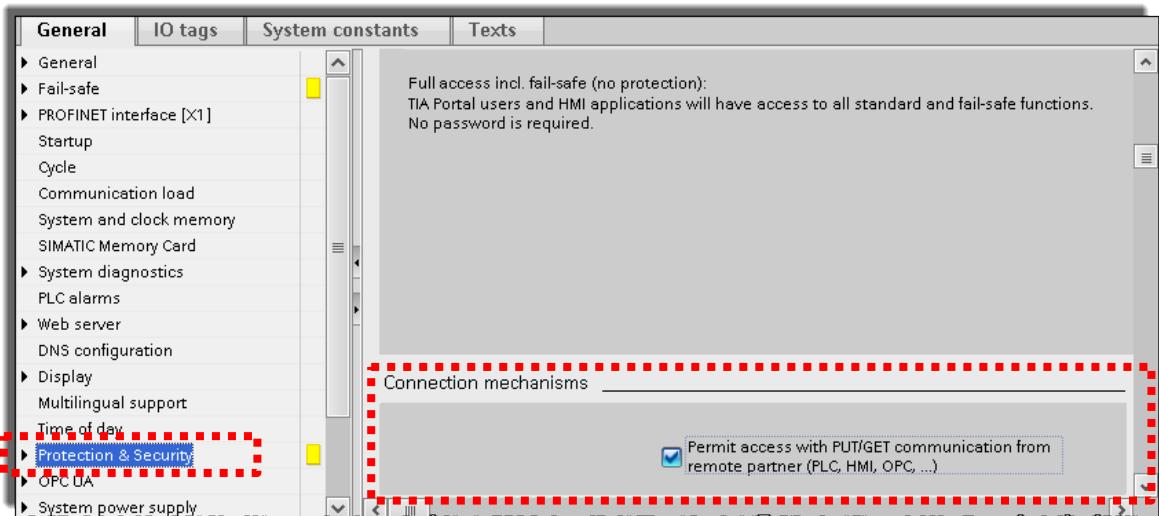
The above explained protection levels can, however, be lifted again with passwords:

- By specifying a password **4** an HMI device can once again read-access and write-access the CPU. For STEP 7, however, neither read-accesses nor write-accesses are possible.
- By specifying a password **3** an HMI device can once again read-access and write-access the CPU and for STEP 7, only read-accesses are permitted, not write-accesses.
- By specifying a password **2** read-accesses and write-accesses of the standard program of the CPU are possible for both an HMI device as well as for STEP 7.

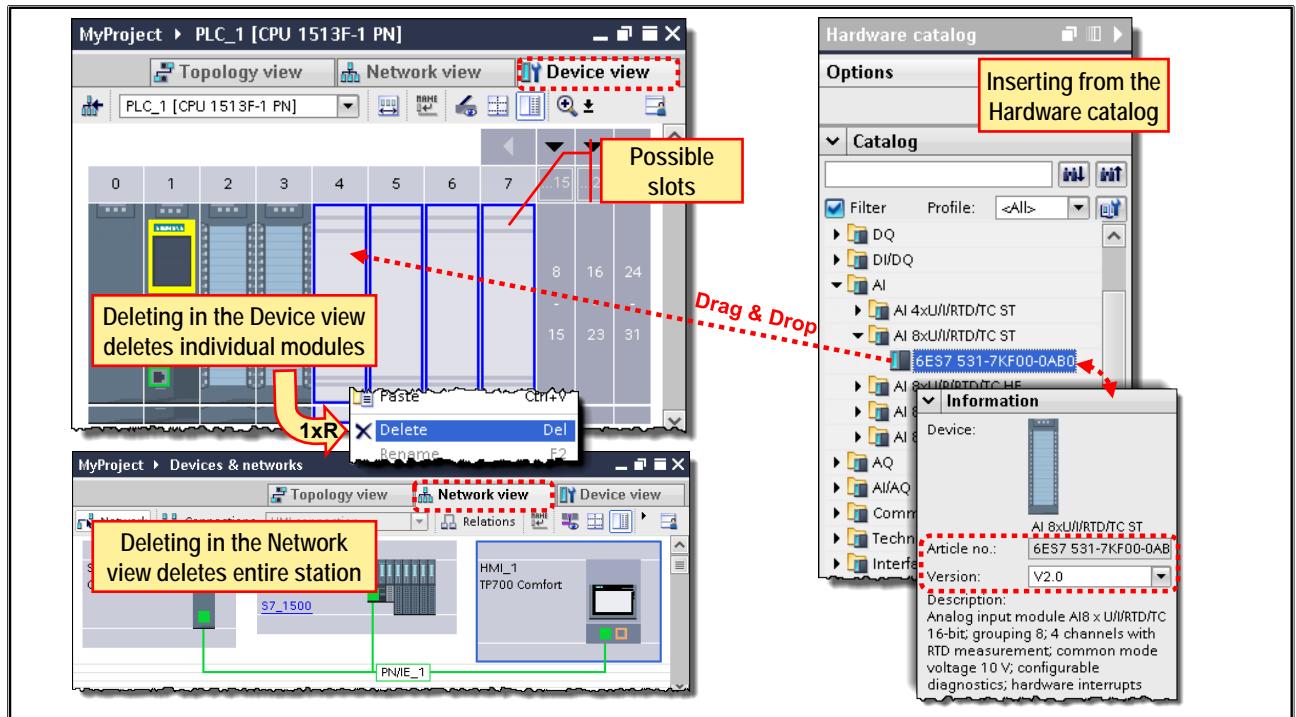
- By specifying a password **①** read-accesses and write-accesses of the CPU are possible for both an HMI device as well as for STEP 7.

Permitting Access by Means of PUT/GET Communication:

- So that other controllers can access the CPU by means of PUT and GET functions, this must be permitted in the Settings of the CPU under Protection & Security > Connection mechanisms.



4.8.3. Inserting / Deleting a Module



Inserting a Module

Modules can be inserted using drag & drop or by means of a double-click.

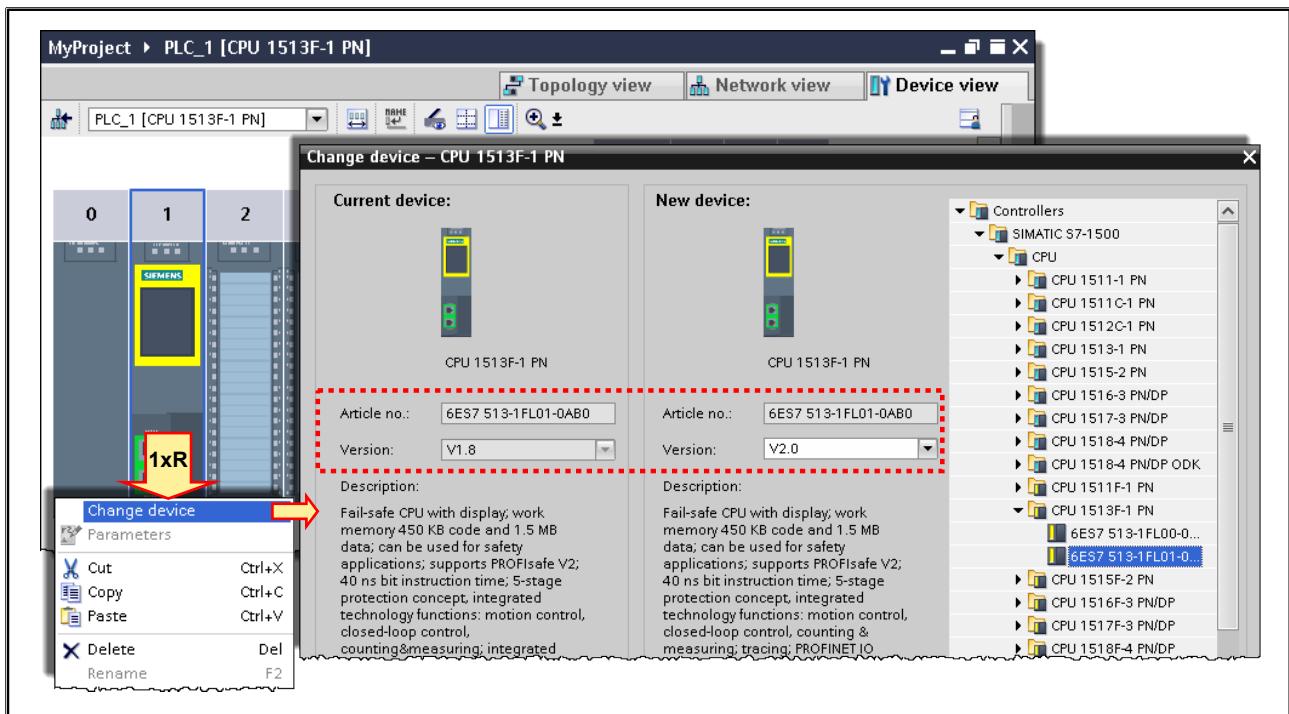
Selecting a Version

When selecting a module, you must pay attention to the correct version. If the module is selected (highlighted) in the task card "Hardware catalog > Catalog", the version can be selected in the task card "Hardware catalog > Information".

Deleting a Module

Deleted hardware components are removed from the system and assigned addresses are made available again.

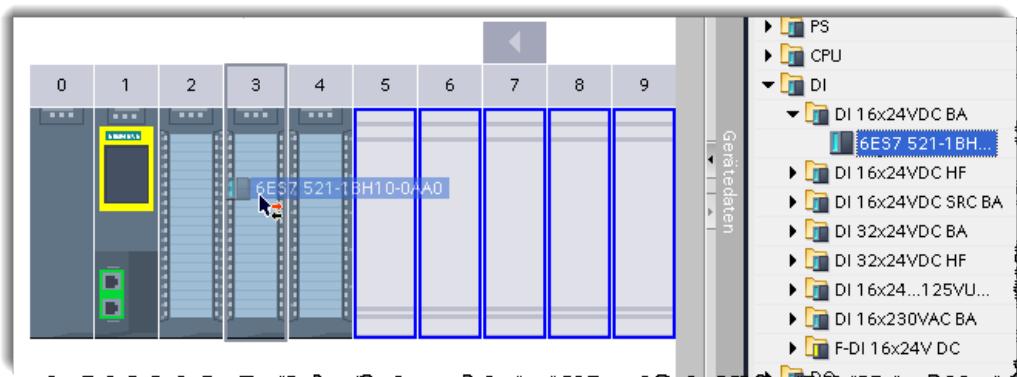
4.8.4. Changing a Device / Module



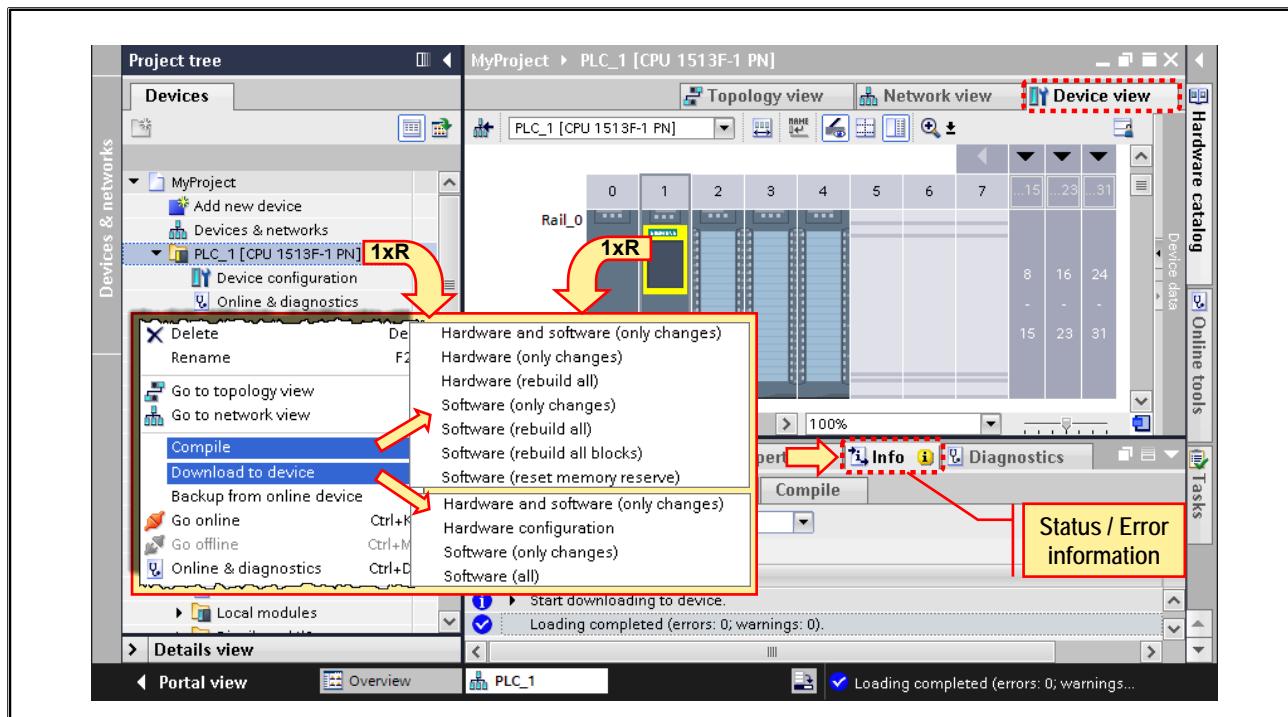
Changing a Module

Compared to deleting and then inserting a new module, the advantage of changing is that when a module is changed (replaced), all the parameters of the old module are adopted on the new module. A module exchange can, for example, then be necessary when the CPU version in the offline project is to be adapted to the CPU version (online) following a firmware update. Hardware components can only be exchanged if the components are compatible.

It is also possible to change a device by dragging the new module from the Hardware catalog onto the old module using drag & drop.



4.8.5. Compiling the Hardware / Software and Downloading it into the CPU

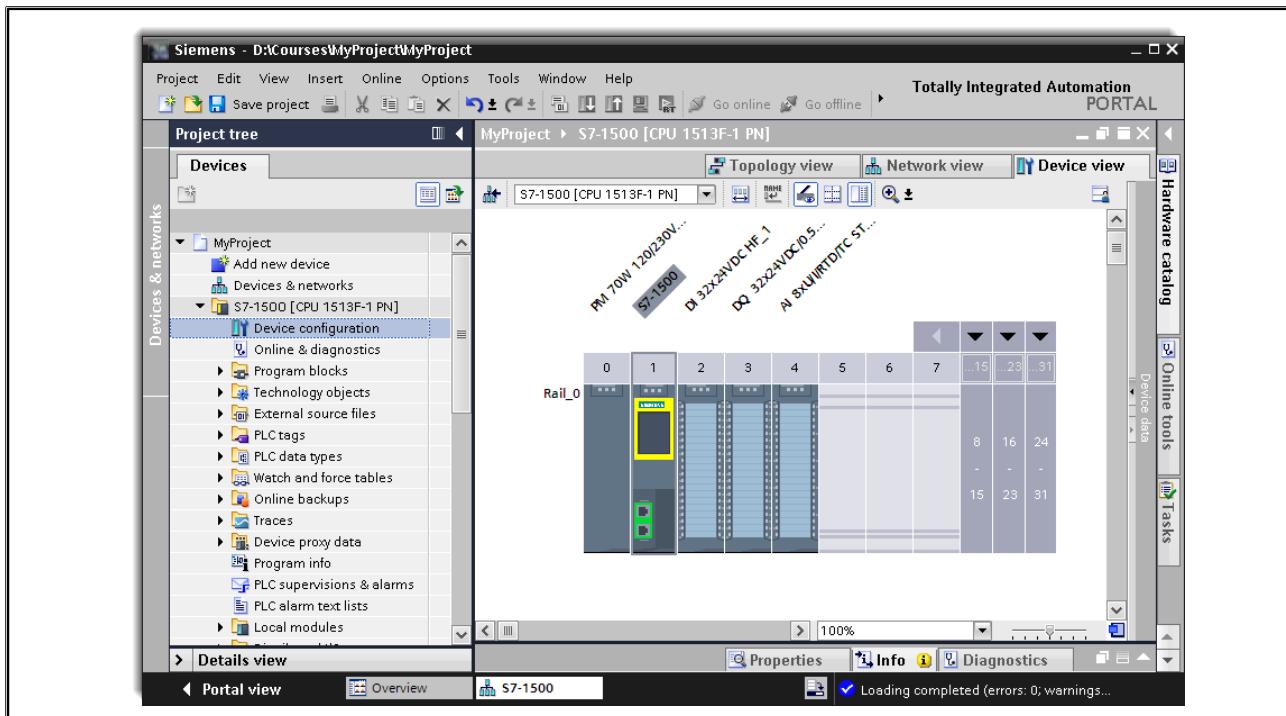


Compiling / Downloading the Hardware Configuration

The following components of a hardware station can be compiled and downloaded:

- **Hardware and software (only changes)**
All changes to the hardware configuration and hardware parameter assignment as well as all changes to the user program are compiled/downloaded.
- **Hardware (only changes) / Hardware configuration**
Only the changes are compiled/downloaded,
- **Hardware (rebuild all)**
The entire hardware configuration and hardware parameter assignment is compiled/downloaded.

4.9. Task Description: Creating a Project with an S7-1500 Station



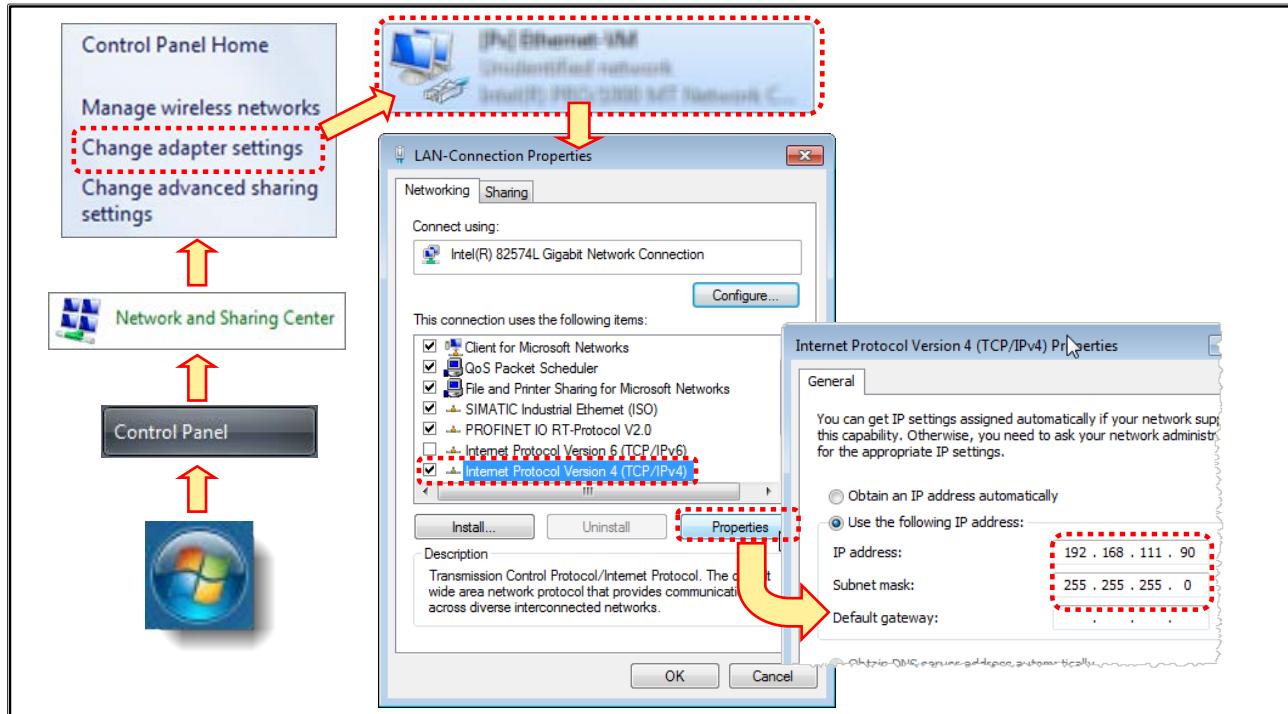
Task Description:

A new project with the name "MyProject2" is to be created. It is to contain an S7-1500 station whose configuration is to correspond exactly to that of your training device.

Furthermore, the modules are to be assigned parameters and the input and output addresses are to be set so that they match those specified in the chapter "Training Devices".

4.9.1. Exercise 1: Setting the IP Address of the PG

PG with Windows 7



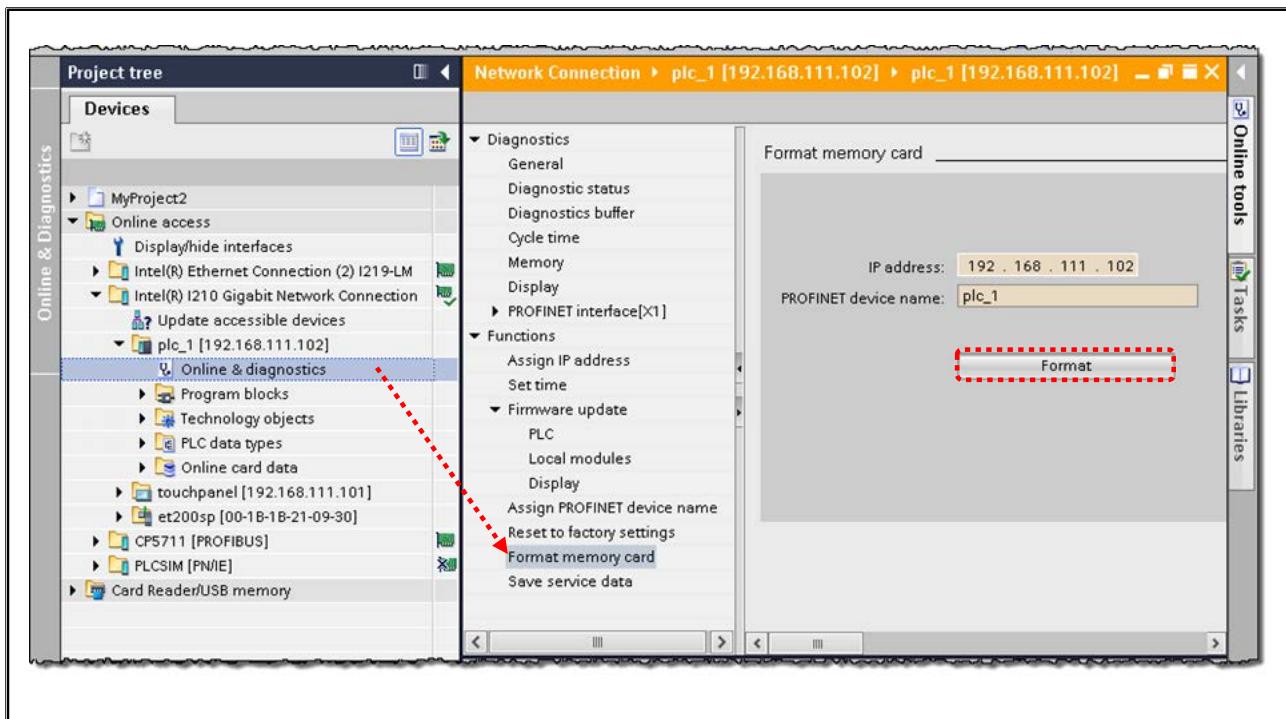
Task

You are to set the IP address of the Ethernet interface of the PG.

What to Do:

1. Connect the Ethernet interface of the PG to the "P2" connection on the training device using an Ethernet cable.
2. Assign the IP address 192.168.111.90 and the subnet mask 255.255.255.0 to this PG interface. Proceed as shown in the picture.

4.9.2. Exercise 2: Erasing the SIMATIC Memory Card of the CPU



Task

In order to completely erase the CPU, the SIMATIC Memory Card of the CPU must also be erased. This can be carried out as follows:

- with the Windows Explorer (SMC is inserted in the PG's Card Reader)
- with the TIA Portal (SMC is inserted in the PG's Card Reader)
- with the TIA Portal (SMC is inserted in the CPU)

What to Do:

1. Check whether the SMC is inserted in the CPU.
2. In the Project view, under the interface through which there is a connection to the controller, display all "Accessible devices"
3. Under the S7-1500 station, activate "Online & diagnostics" (see picture)
4. There under "Functions", activate "Format memory card" (see picture)

Note

If a password is stored on the CPU that is unknown to you, it is only possible to erase the SMC if it is inserted in the PG's Card Reader.

4.9.3. Exercise 3: Resetting the CPU to Factory Settings using the Mode Selector Switch

1. Set the mode selector switch to STOP and remove the SMC

2. Press and hold the mode selector switch in the MRES position until the RUN/STOP LED has flashed 2x slow
then let go again
within 3 sec !!!

3. Press and hold the mode selector switch in the MRES position until the RUN/STOP LED begins to flash quickly
then let go again

4. Insert the SMC and set the mode selector switch to RUN
A CPU restart is carried out

RUN/STOP LED of the S7-1500

Task

In the last exercise you erased the SMC of the CPU. Now, you are to reset the CPU to its factory settings using the mode selector switch.

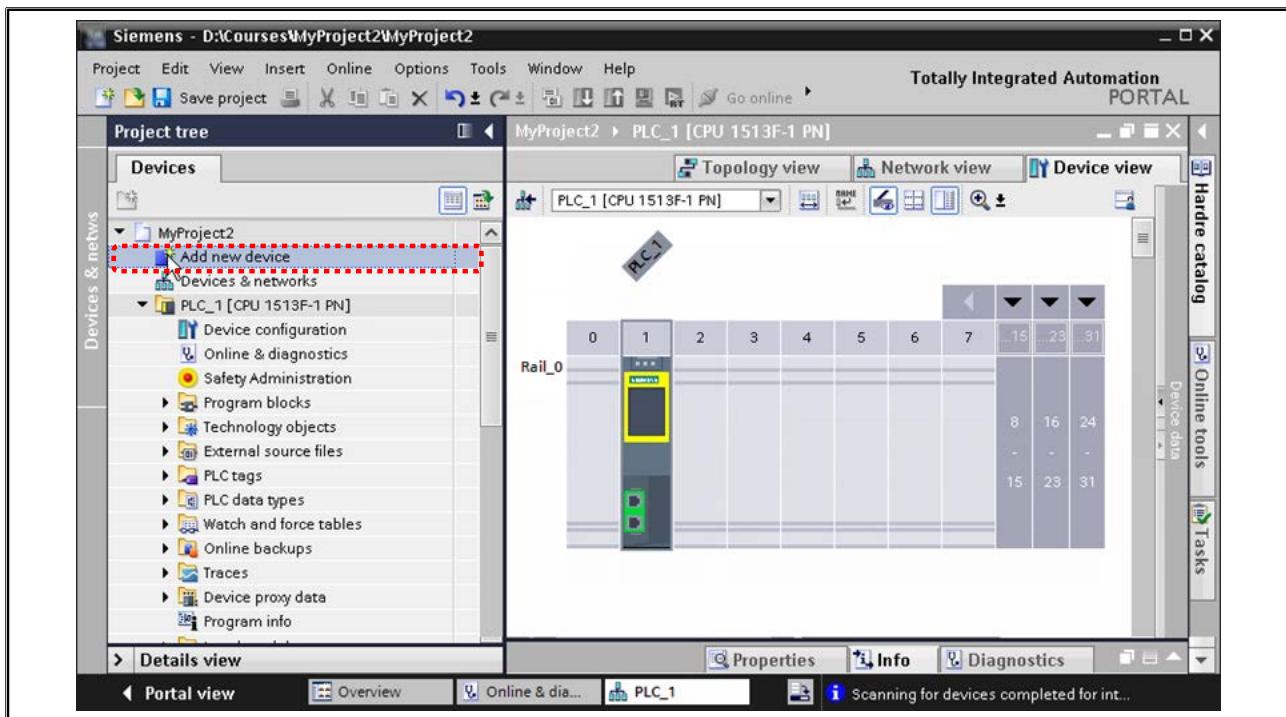
What to Do

1. Switch the CPU to STOP and remove the SMC.
2. Carry out the reset to factory settings directly on the CPU following the steps shown in the picture.

Note:

A CPU restart is not yet possible since no program (Organization Block) has been loaded.

4.9.4. Exercise 4: Creating a New Project and Adding a New Device (Controller)



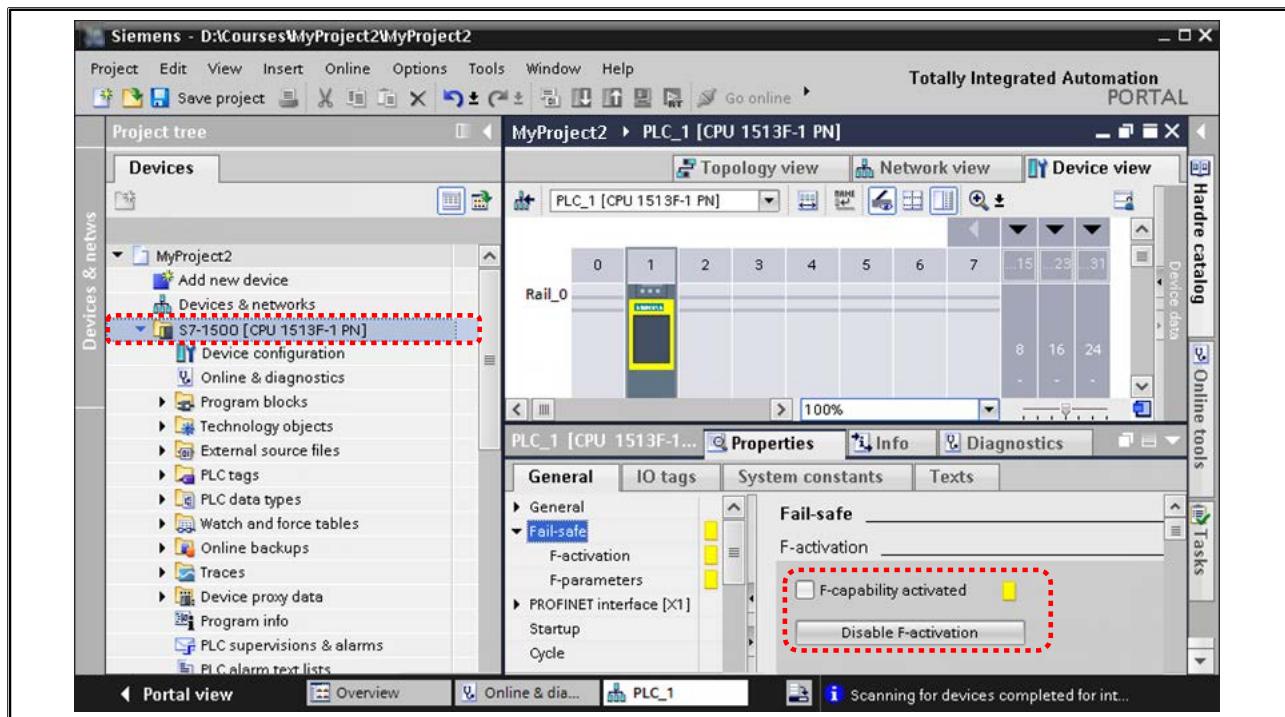
Task

You are to create a new project with the name "MyProject2" and you are to add a new device.

What to Do

1. Save your current project.
2. Create a new one and give it the name "MyProject2".
3. Add the controller off-line which corresponds to your training device as a new device.
(In Exercise 1 of Chapter 2 "System Overview", you already read out the required information via the Display and made a note of it.)

4.9.5. Exercise 5: Changing the Device Name and Disabling the F-activation



Task

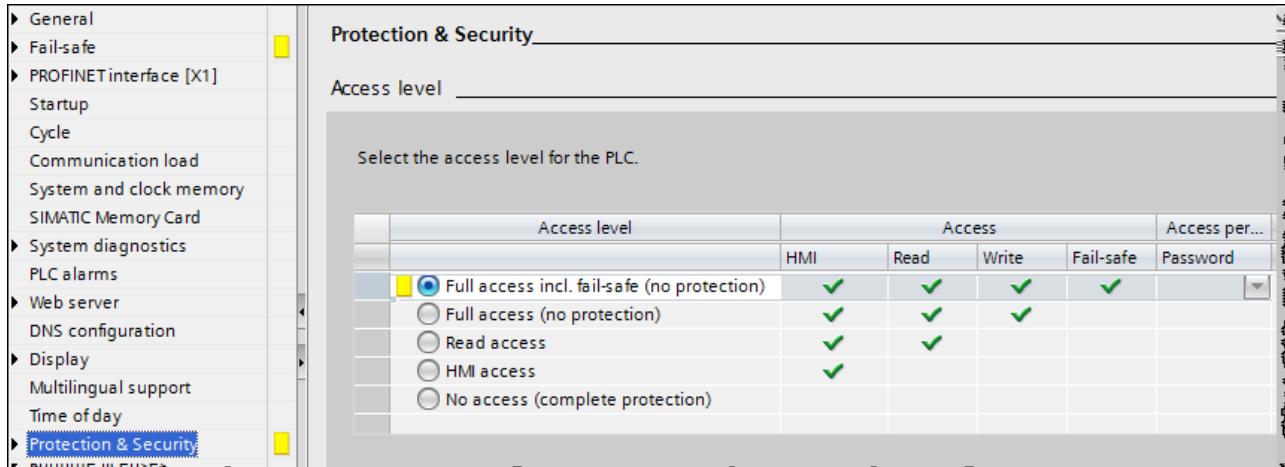
You are to change the device name of the CPU and disable the F-activation.

What to Do

1. Select (highlight) the controller in the Project tree and rename it S7-1500.
2. Open the Device view and the Properties of the CPU in the Inspector window.
3. In the "Fail-safe" menu, deactivate the F-capability.

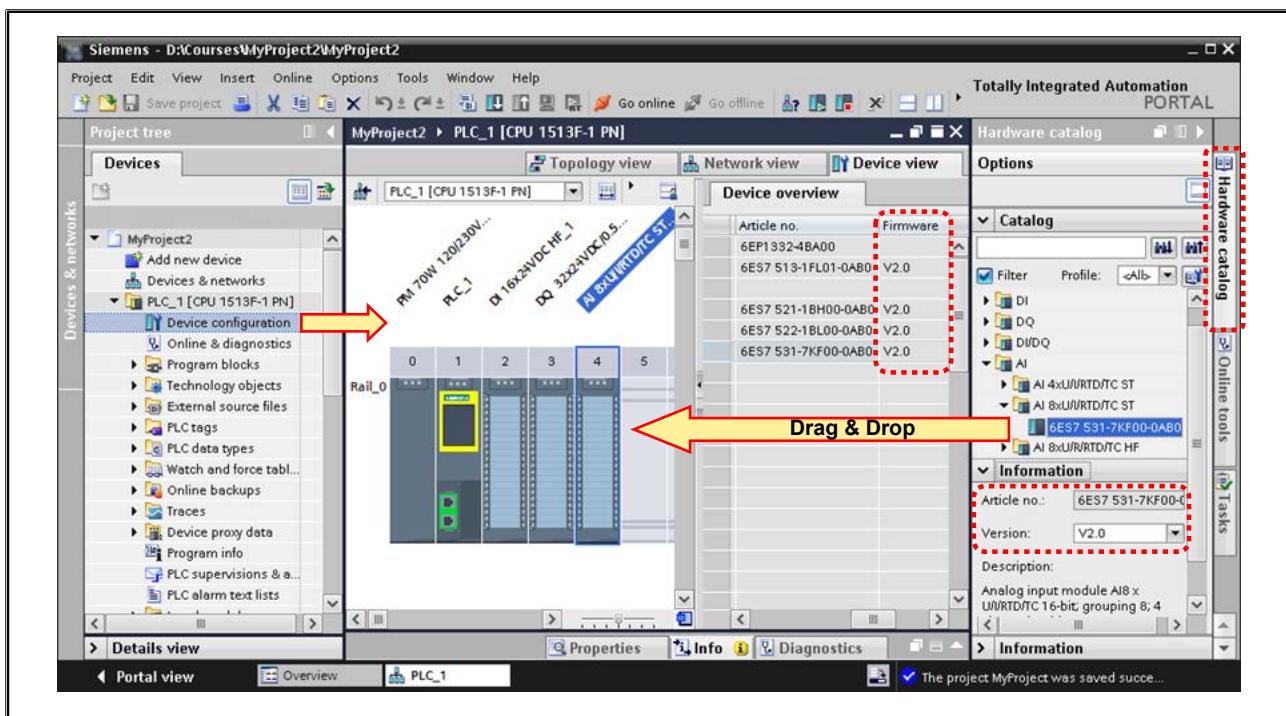
Result: The folder "Safety Administration" is no longer visible in the Project tree.

4. Switch to the "**Protection & Security**" folder and activate the protection level "**Full access incl. fail-safe (no protection)**" (see picture below).



5. Save your project.

4.9.6. Exercise 6: Configuring the S7-1500 Station



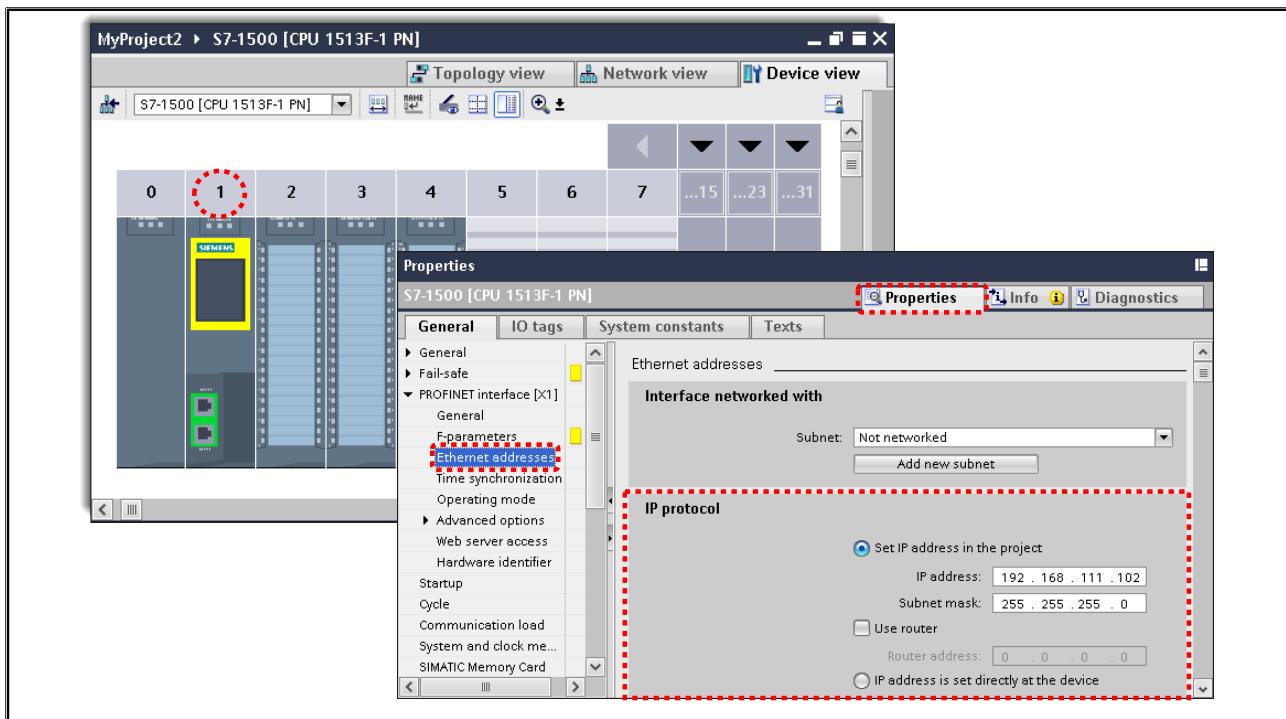
Task

In your offline project, you are to configure the S7-1500 station in such a way that the module arrangement matches that of your actual training device.

What to Do

1. In slots 2 to 4, add the I/O modules from the "Hardware catalog" task card using drag & drop. (In Exercise 1 of Chapter 2 "System Overview", you already read out the required information via the Display and made a note of it.)
2. Insert the appropriate power module from the Catalog into Slot 1.
3. Save your project.

4.9.7. Exercise 7: CPU Properties: IP Address



Task

The S7-1500 CPU is to be assigned an IP address offline in the project.

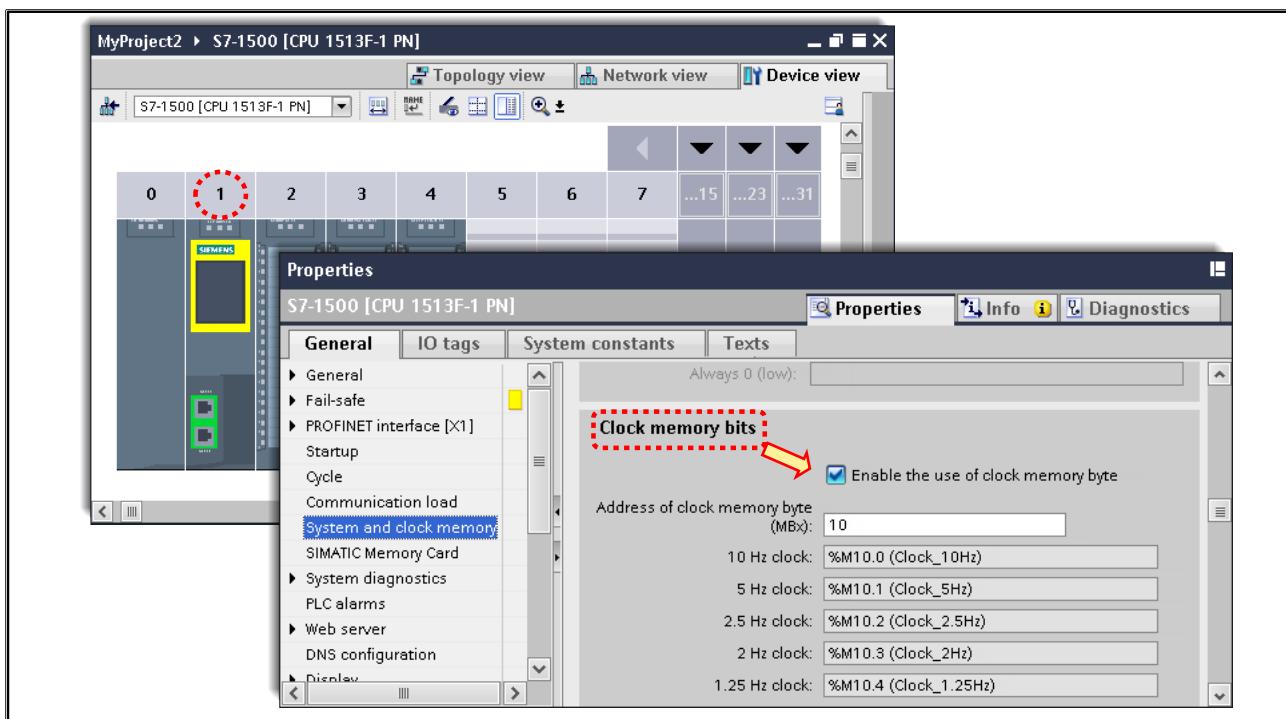


For PROFINET-IO Controllers, a device name is not absolutely necessary. For IO-Devices, it is required for the device identification by the IO-Controller.

What to Do

1. In the "Device view", select the CPU.
2. In the Inspector window under "Properties", select the folder "PROFINET interface [X1] > Ethernet addresses". Enter the IP address and the subnet mask shown in the picture.
3. Save your project.

4.9.8. Exercise 8: CPU Properties: Parameterizing the Clock Memory Byte



Task

In the CPU Properties, you are to parameterize memory byte 10 as a clock memory byte.

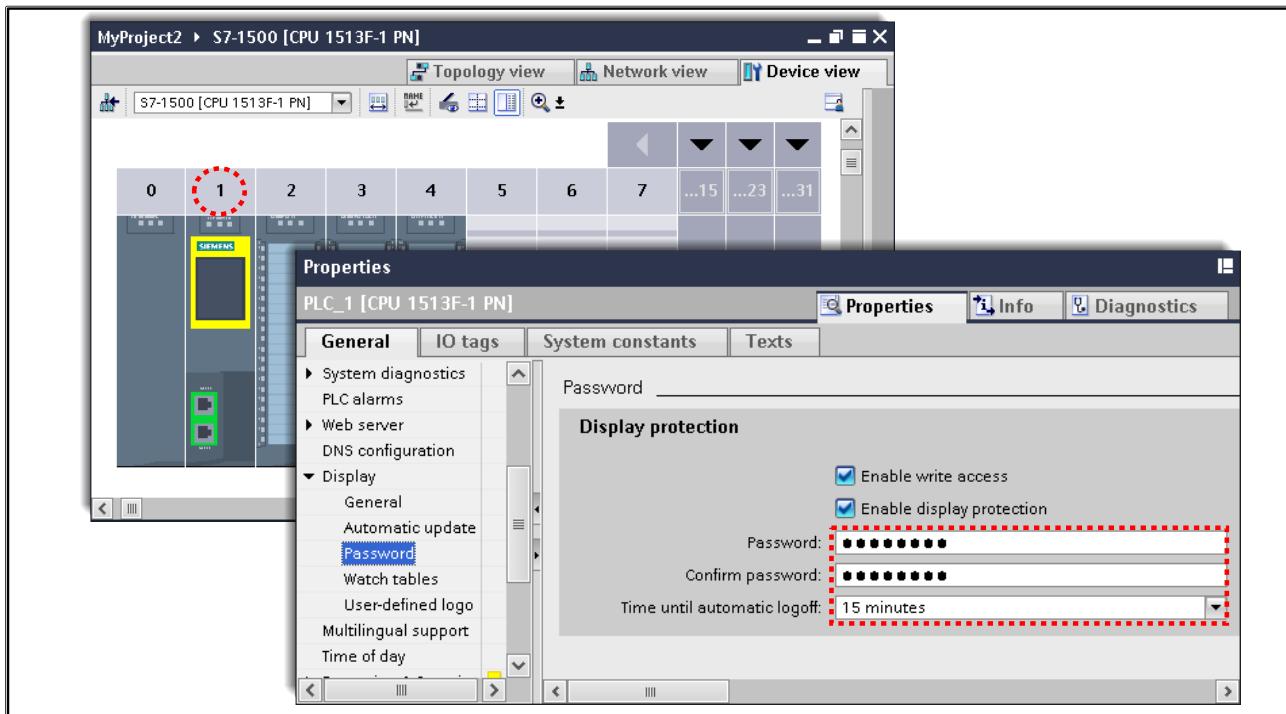
What to Do

1. In the "Properties" tab, select the folder "System and clock memory".
2. Enable the clock memory (byte) and specify address 10 for the byte address.
3. Save your project.

Note:

Only the Clock memory and not the System memory is required. For that reason, deactivate the System memory byte.

4.9.9. Exercise 9: CPU Properties: Display Language and Display Protection



Task

In the CPU Properties, you are to parameterize the display language of the CPU-Display and the display protection.

What to Do

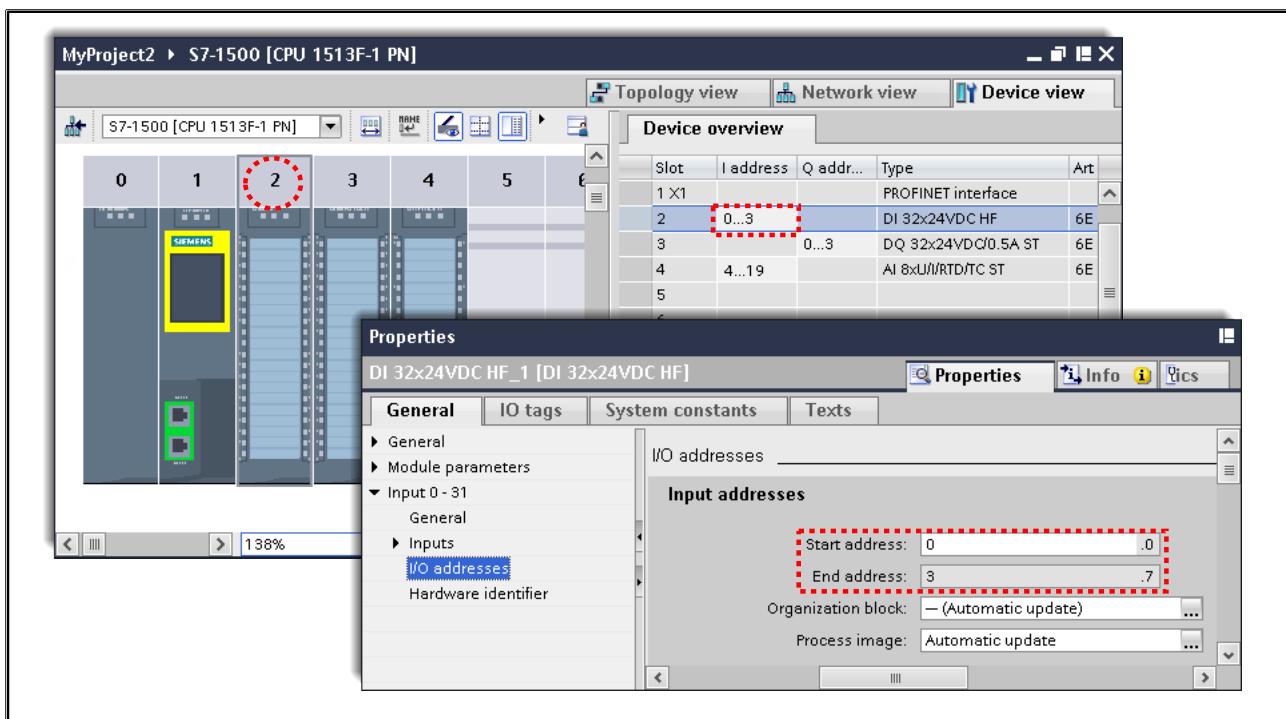
1. In the "Properties" tab under "Display > General", select the item Display language and set the display language to "English".
2. In the Password menu, enable the display protection.
3. Enter a password.
4. Save your project.

Note on Password Assignment:



Upper and lower case is not relevant, since only the letters A to Z and digits 0 to 9 can be selected when making entries on the Display.
Since there is no Display keypad, it is recommended for this exercise that you select a simple (possibly only numerical) password.

4.9.10. Exercise 10: Addresses of the DI Module



Task

You are to parameterize the I/O addresses of the DI module as shown in the picture.

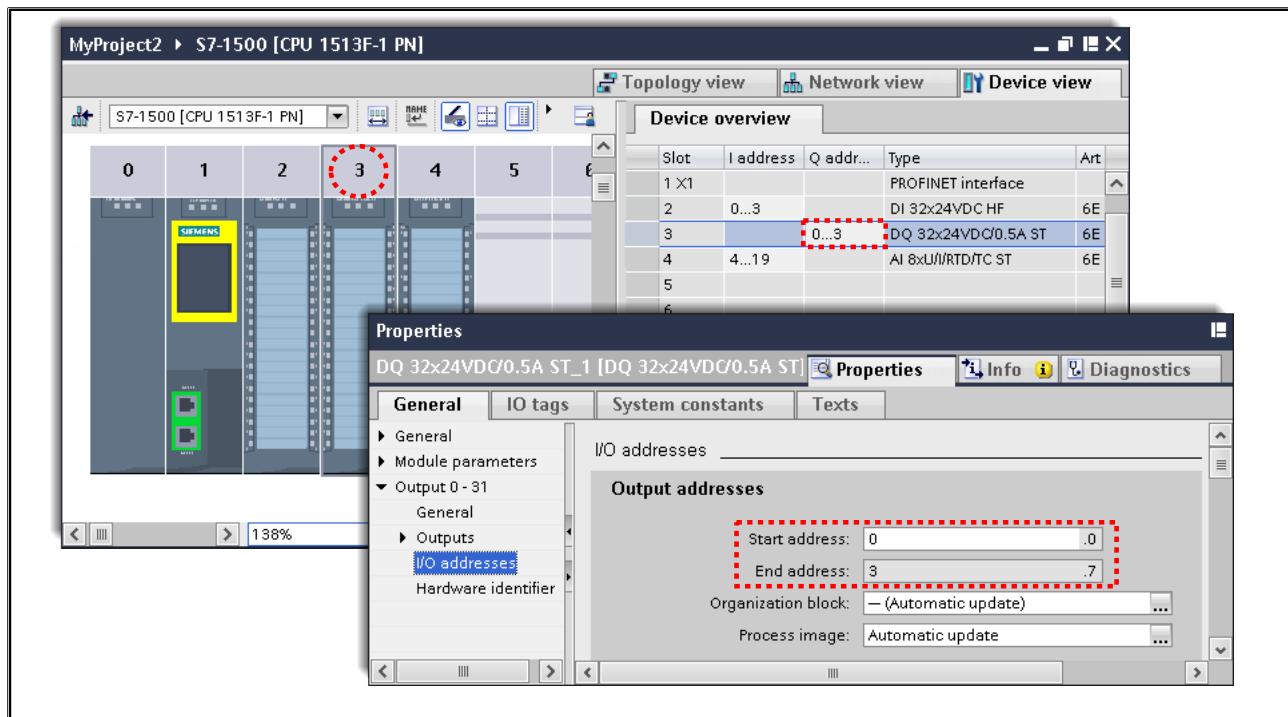
What to Do

1. In the Device view, select the DI module (see picture).
2. In the Inspector window, in the "Properties > General" tab, go to the point "Input 0 - 31 > I/O addresses".
3. Enter the I/O address 0 shown in the picture, (this can also be done in the tabular area of the Device view, see picture)
4. Set the update of the Process image to Automatic so that the address is automatically updated by the system in every program cycle.
5. Save your project.

Notes:

1. The 1500 CPU offers the possibility of using up to 31 process image partitions. "PIP 1" to "PIP 31" process image partitions can be assigned to certain Organization Blocks. After the OB is started, the assigned process image partition for the inputs is updated by the system. At the end of the OB, the outputs of the assigned process image partition are written to the I/O outputs by the system. The process image partitions are excluded from the automatic update.
2. A process image partition can be updated in the user program with special instructions. For this, there are the functions "UPDAT_PI" for the process image partition of inputs and "UPDAT_PO" for the process image partition of outputs.

4.9.11. Exercise 11: Addresses of the DO Module



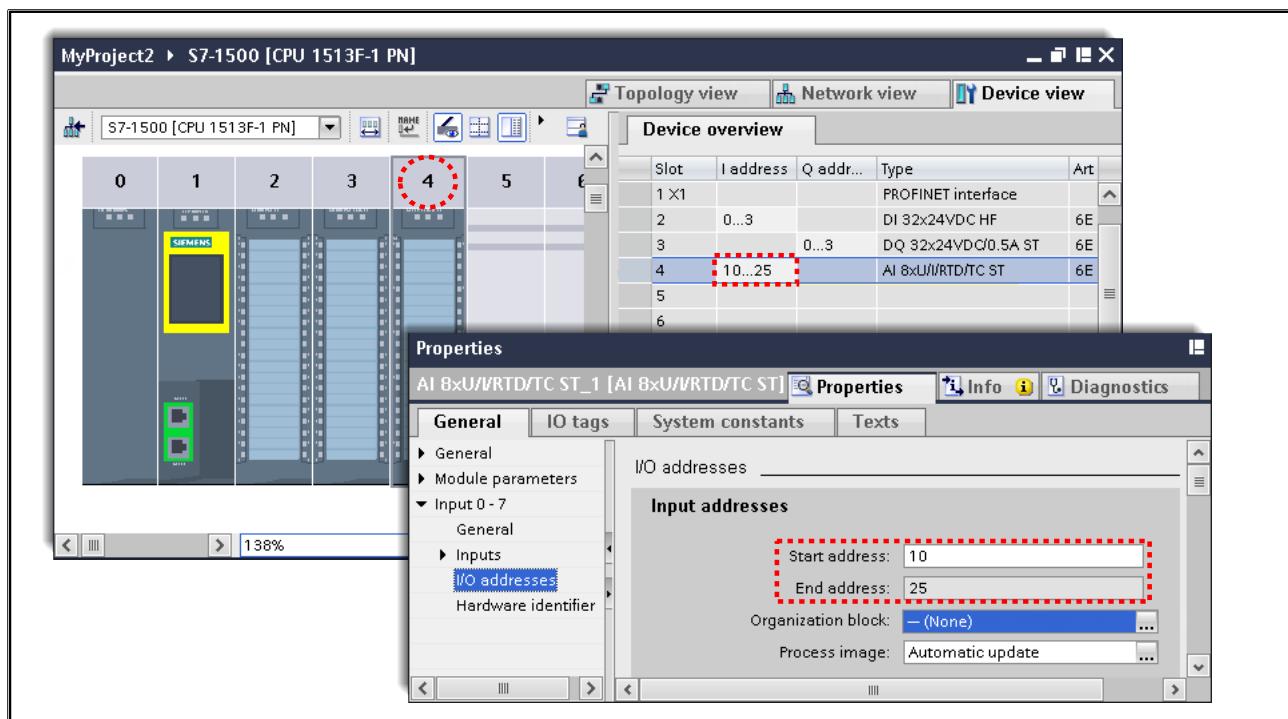
Task

You are to parameterize the I/O addresses of the DO module as shown in the picture.

What to Do

1. In the Device view, select the DO module (see picture).
2. In the Inspector window, in the "Properties > General" tab, go to the point "Output 0 - 31 > I/O addresses".
3. Enter the I/O address 0 shown in the picture and set the update of the Process image to Automatic.
4. Save your project.

4.9.12. Exercise 12: Addresses of the AI Module



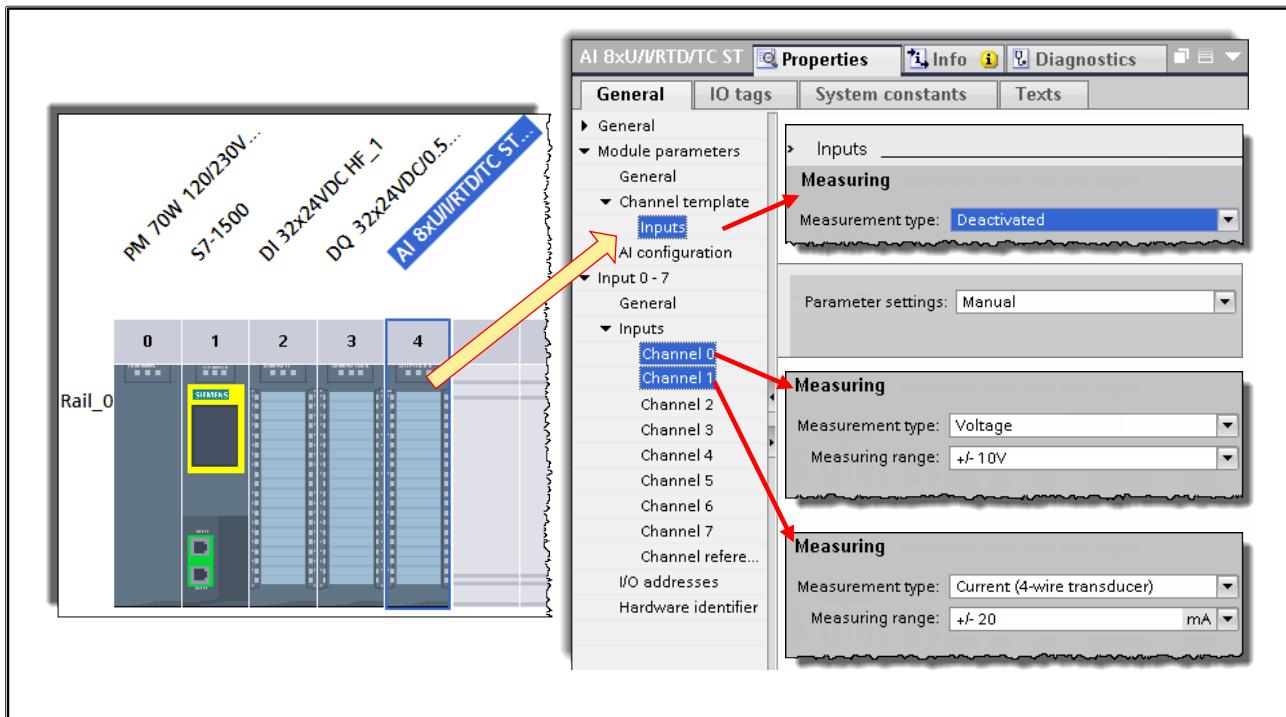
Task

You are to parameterize the I/O addresses of the AI module as shown in the picture.

What to Do

1. In the Device view, select the AI module (see picture).
2. In the Inspector window, in the "Properties > General" tab, go to the point "Input 0 - 7 > I/O addresses".
3. Enter the I/O address 10 shown in the picture.
4. In the assignment of the process image, set "None" since the analog value of the module is to be read directly by the I/O later in the program and thus does not have to be updated with any process image.
5. Save your project.

4.9.13. Exercise 13: Setting the Channel Parameters of the Analog Input Module



Task:

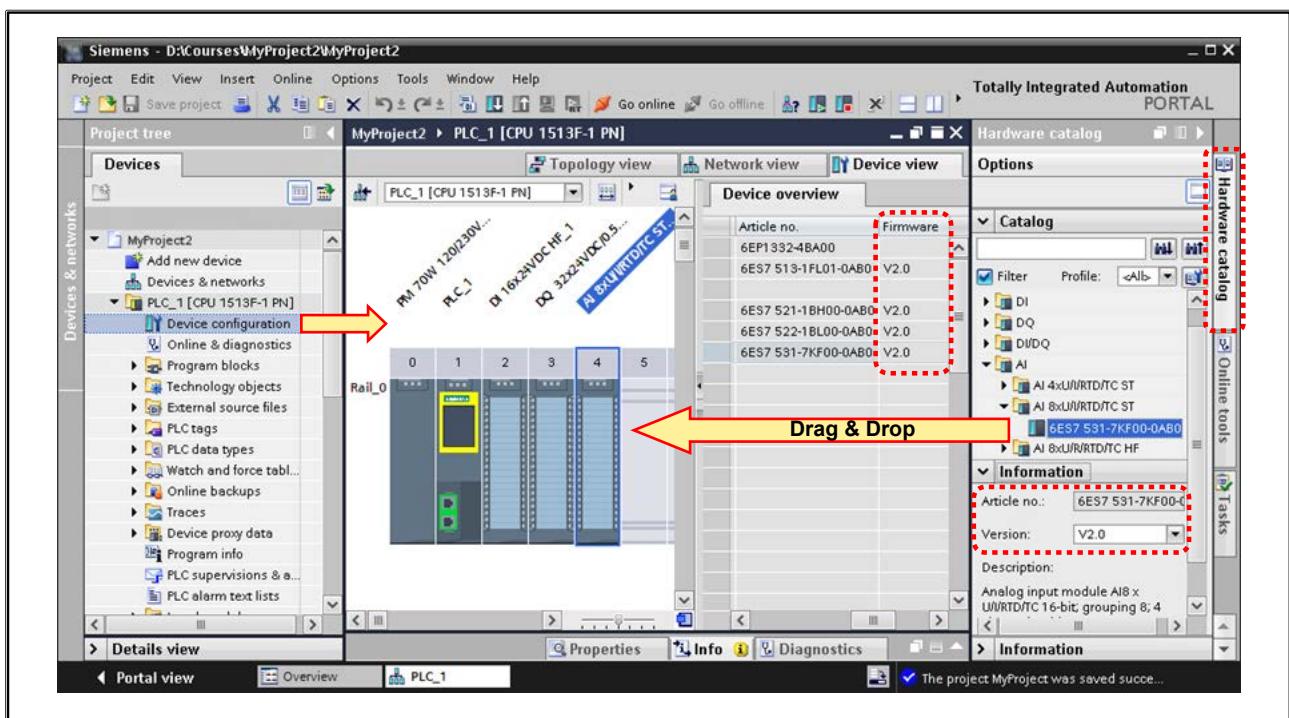
You are to set the analog channels to the appropriate parameters:

- Set the Channel template (Measurement type) to 'Deactivated'
- Channel 0 → Voltage +/-10V
- Channel 1 → Current (4-wire transducer) +/-20 mA

What to Do

1. In the Inspector window, in the "Properties > Module parameters" tab, go to the point "Channel template > Inputs" and set Measurement type 'Deactivated' as the template.
2. Under "Properties > Input 0 - 7 > Inputs" set the Measurement type "Voltage" +/-10V for Channel 0 and for Channel 1 set the Measurement type "Current (4-wire transducer)" +/-20 mA.
3. Save your project.

4.9.14. Exercise 14: Compiling the Device Configuration and Downloading it into the CPU



Task

You are to compile the configuration and parameterization of the S7-1500 hardware station and then download it into the CPU.

Note:

As long as the CPU doesn't have a program, the CPU does not go into RUN mode when there is a restart! That is, if, as shown in the picture, you only download the hardware configuration into the CPU in this exercise, the CPU will not switch into the RUN mode with a subsequent restart!

What to Do

1. In the Project view, select your S7-1500 station.
2. Compile the HW-Station (right-click on the station, see picture)
3. After an error-free compilation, download the hardware configuration into the CPU (right-click on the station, see picture)



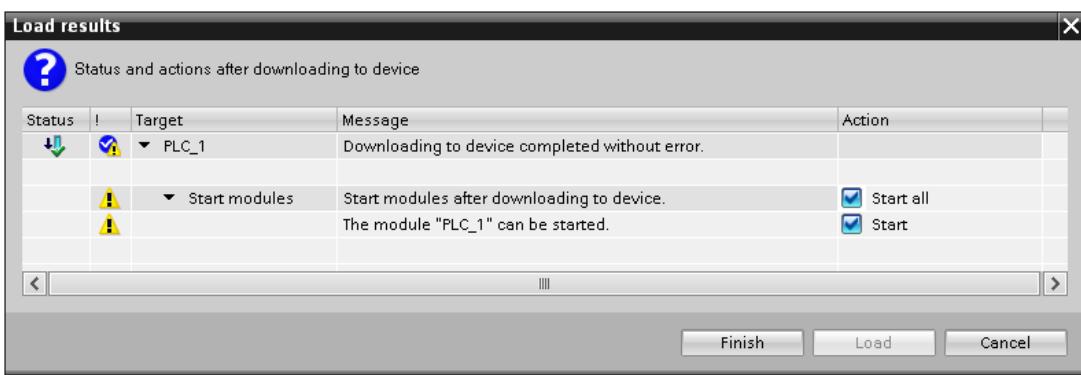
Note: When using the buttons shown here to the left, a Delta compilation or Delta download of the folder(s) and subfolder(s) selected in the Project tree is always carried out. That is, if a station (CPU) is selected, a Delta compilation or Delta download of the entire station, (hardware and software) is carried out.

Continued on the next page

4. Confirm the information in the dialog by pressing the button "Load":

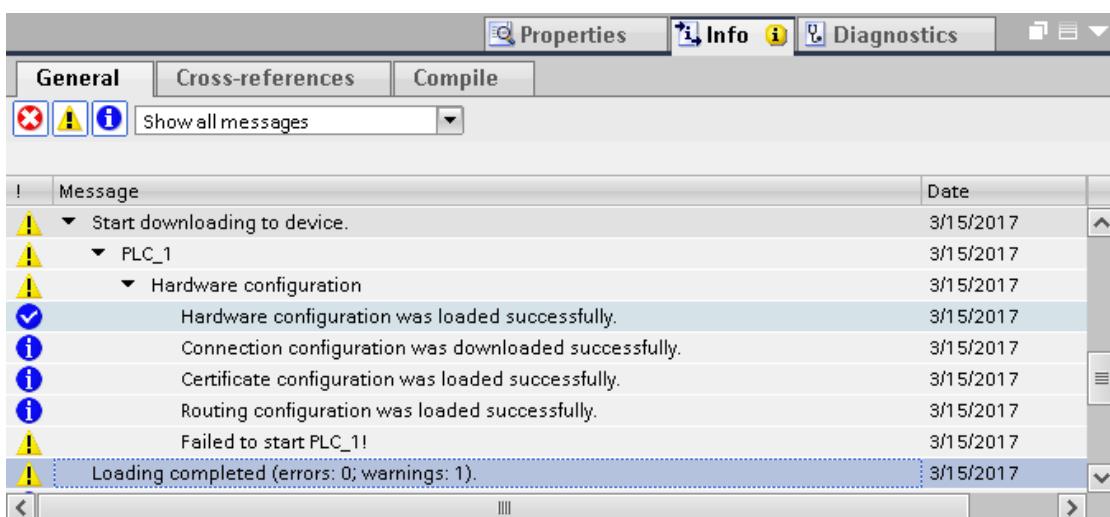


5. After you have activated "Load", another dialog appears which you fill-in as follows and then conclude with "Finish":



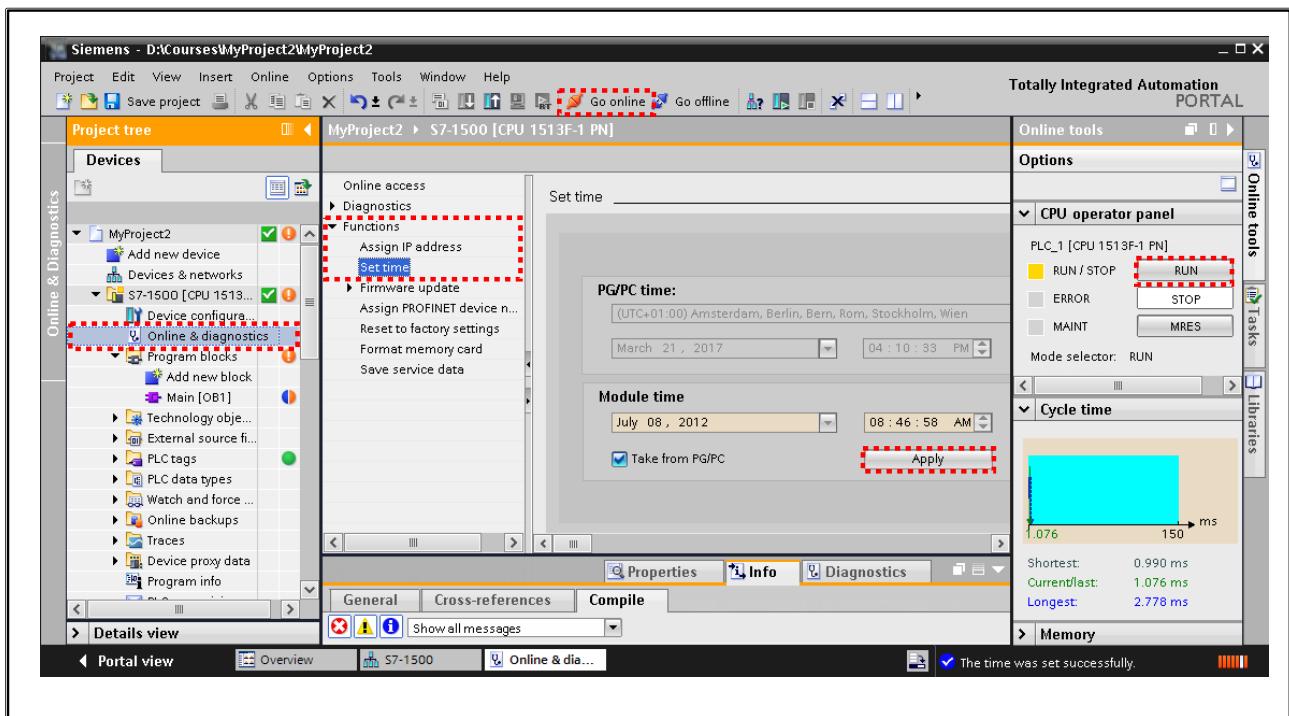
If you have only downloaded the hardware configuration into the CPU, the CPU should now remain in the STOP mode.

6. In the "Inspector window" under "Info -> General" check the result of the hardware configuration download:



7. Save your project.

4.9.15. Exercise 15: Setting the Time and Trying to Switch the Controller to RUN Mode



Task

By means of the TIA Portal, you are to set the time on the controller and try to start the controller.

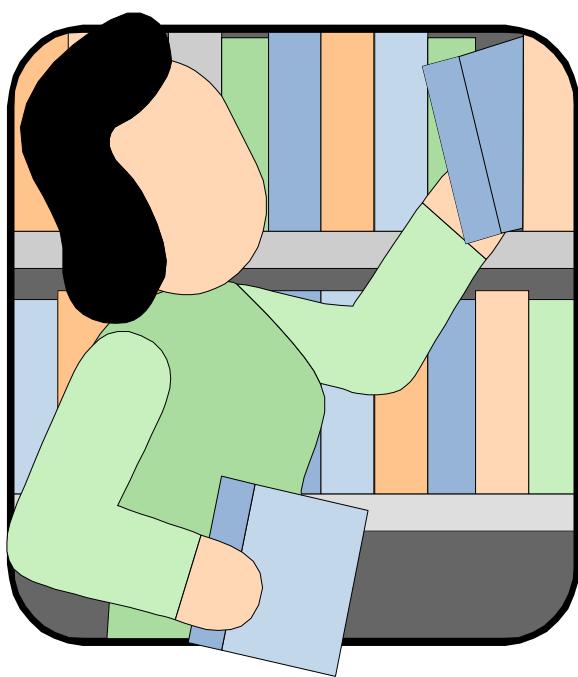
What to Do

1. Open the Online access of the S7-1500 CPU via the object "Online & diagnostics" in the Project tree.
2. Establish an online connection to the controller via the button "Go online" (The "Go online" button is located in the toolbar and in the 'Online access' window opened in the working area).
3. In the 'Online access' window, switch to the menu "Functions > Set time". (The PG/PC time and the Module time can be seen.)
4. Adopt the PG/PC's time by activating the item "Take from PG/PC" and confirm with the "Apply" button.

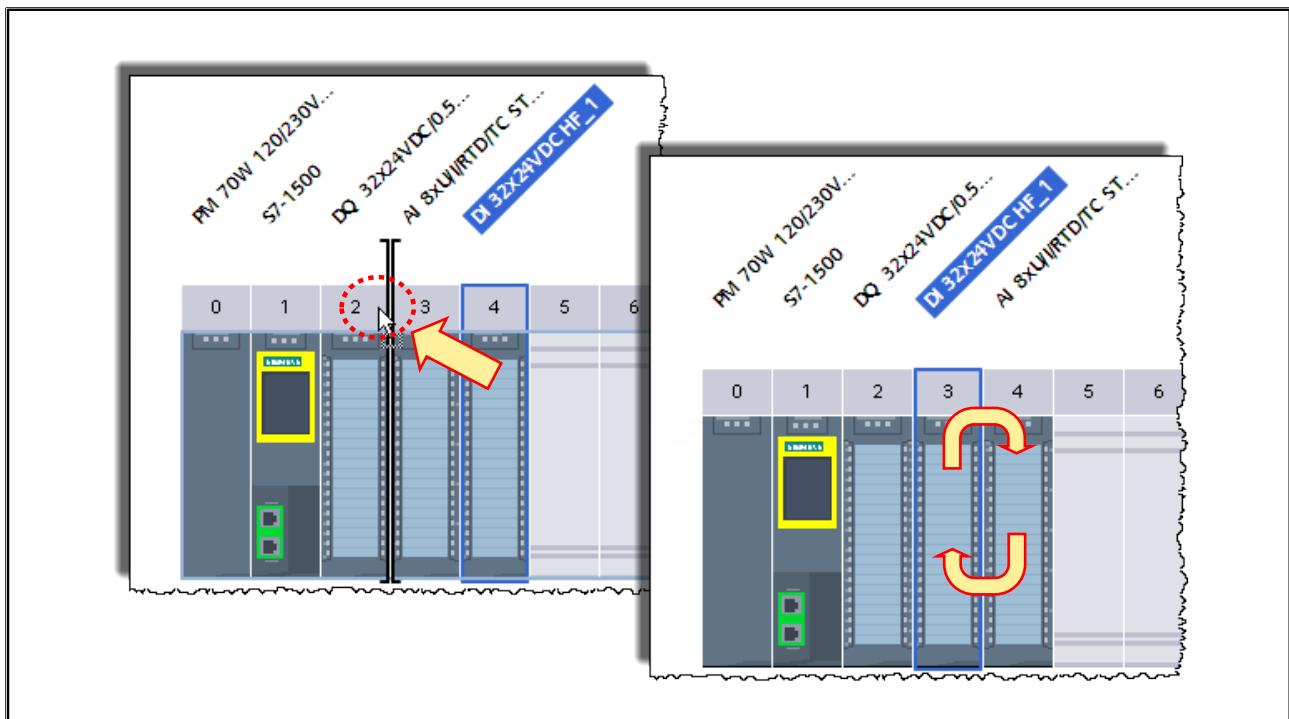
Note:

If the item "Take from PG/PC" is deactivated, the module time can be manually changed.

4.10. Additional Information



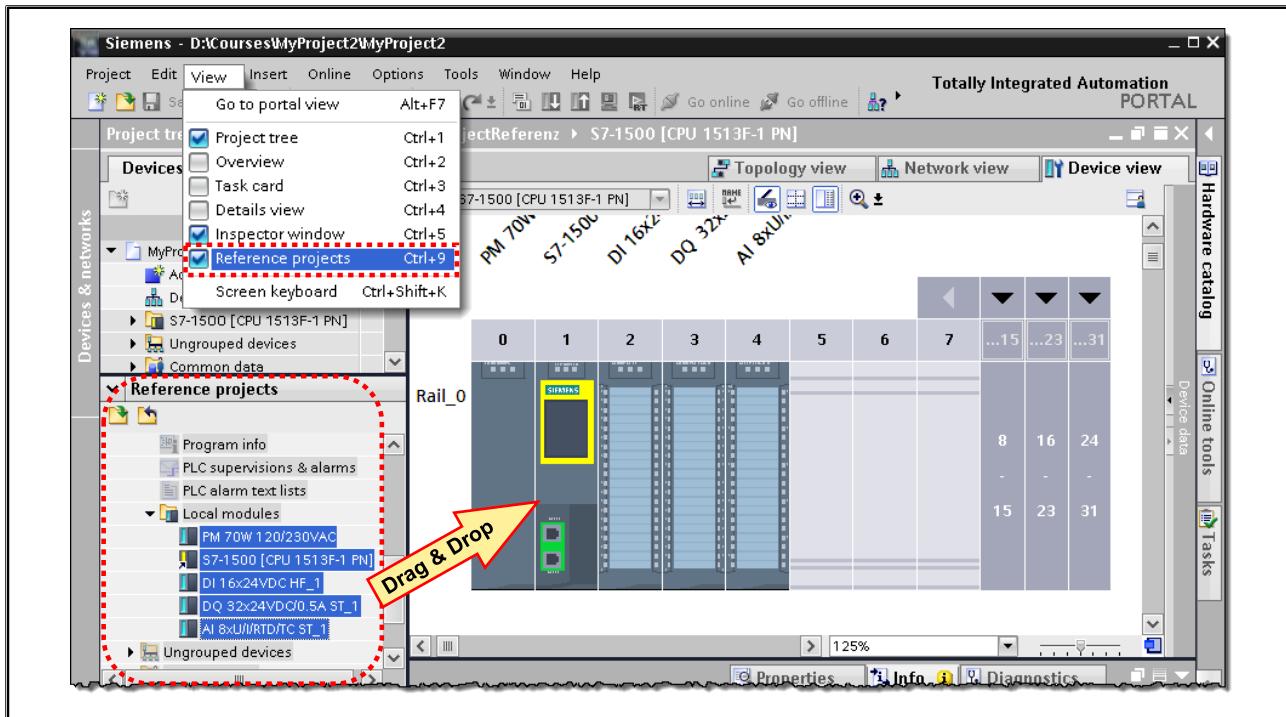
4.10.1. Swapping a Slot / Inserting a Module between Two Modules



Swapping a Slot or Inserting a Module between Two Modules.

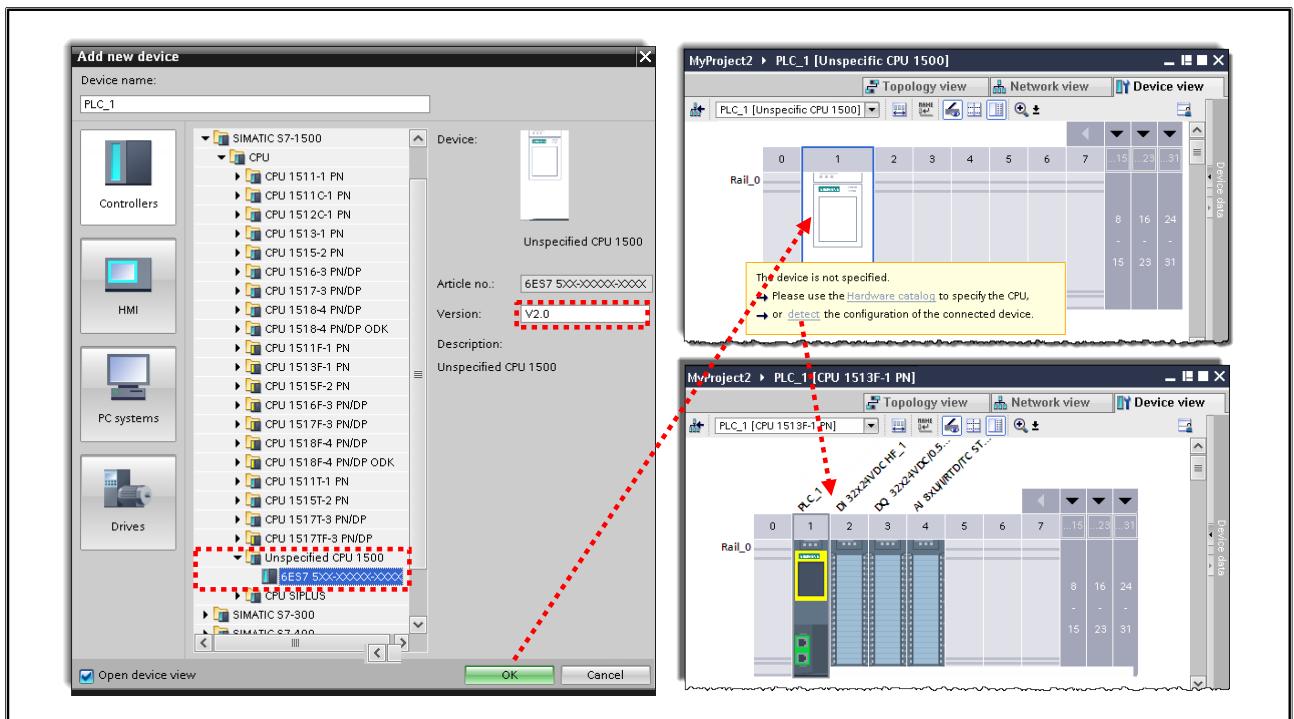
Using drag & drop, drag the modules in front of or behind the slot number until the marking appears at the desired location, as shown in the picture. The module is placed where the marking is and the modules behind it are moved one slot to the right.

4.10.2. Copying Modules from a Reference Project



Via the menu "View", the "Reference projects" view can be shown, in which projects can be opened as write-protected. Modules can be copied into the Device view from a Reference Project. In doing so, all parameter assignments are adopted.

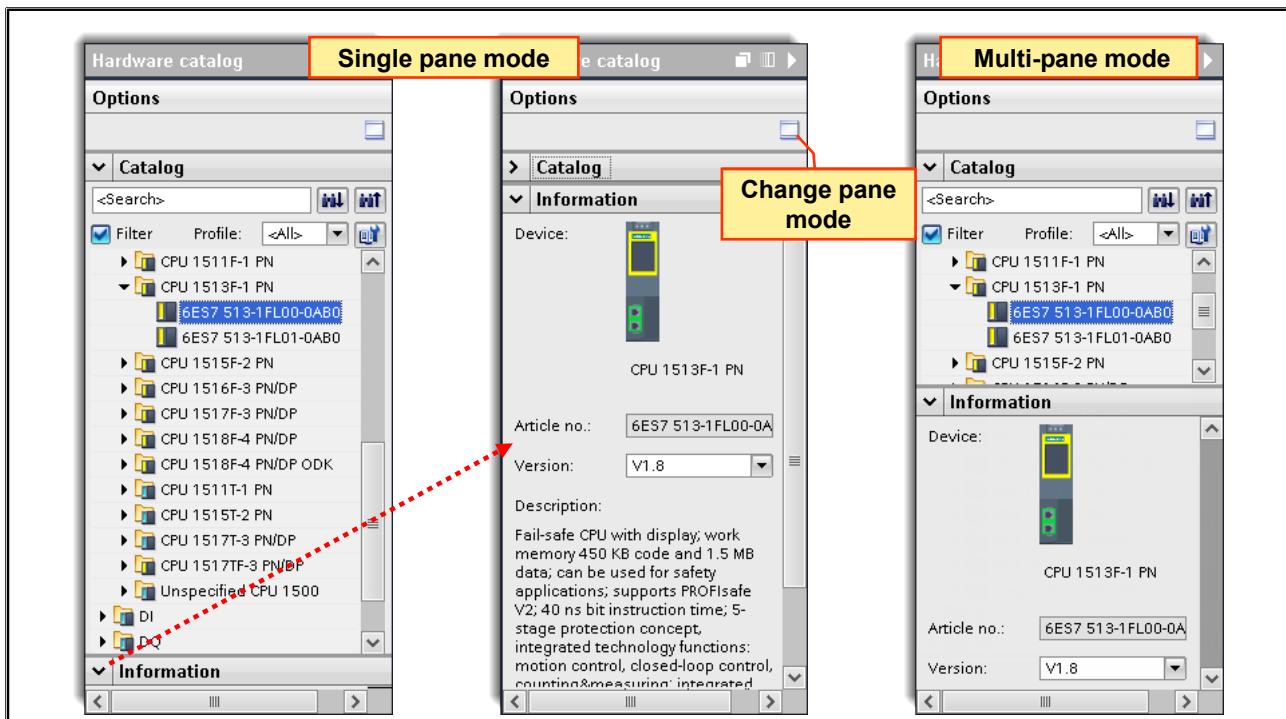
4.10.3. Unspecified CPU



For the selection of the controller, you can also choose an "Unspecified CPU". This is necessary when the real CPU is not yet known but you would already like to start programming. Here, you only need to specify the firmware of the CPU which is to be used later on. The firmware version must be specified since some functions and instructions which are used for programming depend on the firmware version.

Then, you can write the program without having configured the actual hardware. The hardware can later be configured from the Hardware catalog or determined online.

4.10.4. ‘View’ Settings of the Task Cards



You can choose between two pane modes:

- **Single pane mode:**
There is only one pane open at a time. If a different pane is selected, the previously open pane is closed automatically.
- **Multi-pane mode:**
Several panes can be open at the same time.

Setting for the Device Configuration

Since there is generally more than one version of a module when configuring the devices (CPUs, I/O modules), the required version must be selected.

Since this additional information on the modules selected in the catalog is shown in the "Information" pane, it is recommended that the **multi-pane mode** is set here.

Contents

5

5.	PLC Tags.....	5-2
5.1.	What are Tags and Why do You Need them?.....	5-3
5.1.1.	Addresses of PLC Tags	5-4
5.2.	Elementary Data Types Bit and Numeric.....	5-5
5.2.1.	Elementary Data Types Date, Time and Character.....	5-6
5.3.	Declaration and Definition of PLC Tags.....	5-8
5.3.1.	PLC Tags and PLC Constants.....	5-9
5.3.2.	PLC Tags in the Device View	5-11
5.3.3.	Finding / Replacing / Sorting PLC Tags.....	5-12
5.3.4.	Error Indication in the PLC Tag Table	5-13
5.3.5.	Copy & Paste PLC Tags to Excel	5-14
5.3.6.	Retentiveness of PLC Tags	5-15
5.4.	Details View of PLC Tags	5-16
5.5.	Monitoring PLC Tags	5-17
5.5.1.	Modifying PLC Tags by means of the Watch Table.....	5-18
5.6.	Exercise 1: Copying the PLC Tag Table from the Library	5-19
5.6.1.	Exercise 2: Creating the "Conveyor" Tag Table	5-20
5.6.2.	Exercise 3: Monitoring the "Conveyor" PLC Tag Table	5-21
5.6.3.	Exercise 4: Modifying using the Watch Table.....	5-22

5. PLC Tags

At the end of the chapter the participant will ...

- ... be familiar with PLC tags and their memory areas
- ... be able to create PLC tags and address them
- ... know elementary data types
- ... be familiar with global constants and system constants
- ... be able to apply the details view
- ... learn how to monitor and modify PLC tags



5.1. What are Tags and Why do You Need them?

A tag...

- is a memory space on the CPU
- saves values/data
- is described by its name, memory area, address and data type (size, possible value range, use, allowed instructions)

Memory areas of PLC tags:

- Process (image) tags for **I**nputs or **Q** Outputs
 - Auxiliary tags for saving values in the **M**emory byte area
 - CPU-internal **C**ount area and Time area (**T**imer)
→ not available for S7-1200
-

The Importance of PLC Tags

Next to commands, tags (variables) are the most important elements of a programming system. Their task is to save values in a program so that they can be further processed at a later time.

Data Types

The data type determines which values are accepted by data and which instructions can be carried out with these values.

Memory Areas of Inputs and Outputs

Within a controller, the data is stored in different memory areas. The input signals of input modules are stored in the process image for inputs where they can be consistently read out throughout a cycle. The control of the process happens via the process image for outputs which is then written to the output modules.

Memory Byte Area

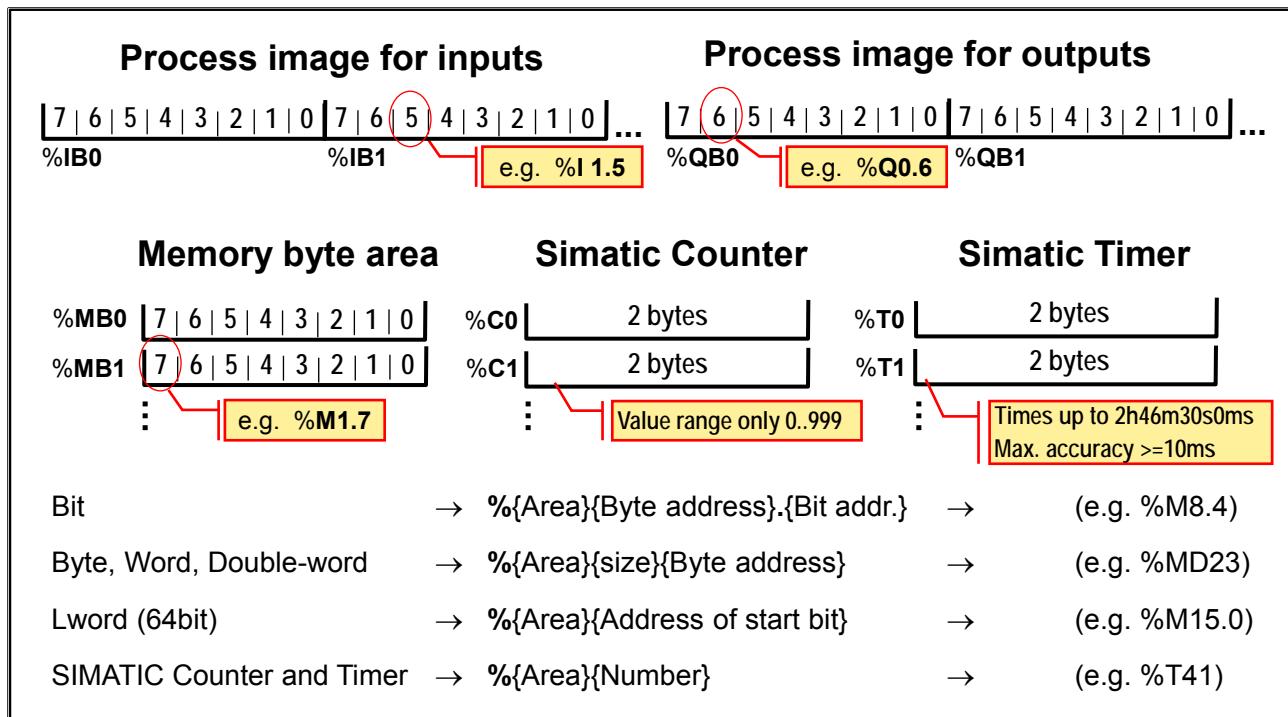
Internally, there is an additional storage area in which values can be saved. Since the clarity and the re-usability of individual tags (variables) of the memory byte area are not fulfilled, there are still other areas which will be discussed in later chapters.

CPU-internal Count and Time Areas (SIMATIC Counter and Timer)

These are a simple way to use count and time functions. SIMATIC counters and timers are, however, very limited. The counters only have a value range of 0 to 999 and the timers can only record times up to 2 hours 46 minutes and 30 seconds with an accuracy of 10 milliseconds and are not available on 1200-series CPUs.

For greater demands on timers and counters there are functions which also will be dealt with in later chapters.

5.1.1. Addresses of PLC Tags



Addressing

An address exactly defines where values are written or read.

They begin at byte address, that is, at the number "0".

The addresses are consecutively numbered, and, within a byte, the bit address 0..7 is numbered from right to left.

Address Style With/Without %

The %-character identifies the presentation of an address. It can be left out during address entry since it is automatically added by the engineering framework.

PLC Tags > 32 Bits:

For tags that are greater than 32 bits (e.g. LWORD), only the start bit is specified during addressing since this can only be accessed symbolically. The required number of bytes that must be reserved for these tags results from the data type of the PLC tag.

5.2. Elementary Data Types

Bit and Numeric

	Description	Size (Bit)	S7-1200	S7-1500	Example
Bit Data Types	BOOL	1			TRUE
	BYTE	8	✓	✓	B#16#F5
Numeric Data Types	WORD	16			W#16#F0F0
	DWORD	32			DW#16#F0F0FF0F
	LWORD	64	✗	✓	LW#16#5F52DE8B
	SINT USINT	8			50 20
	INT UINT	16			-23 64530
	DINT UDINT REAL	32	✓	✓	DINT# -2133548520 UDINT#435676 1.0
	LREAL	64			LREAL#-1.0e-5
	LINT ULINT	64	✗	✓	LINT#1543258759 ULINT#154316159

BOOL, BYTE, WORD

Variables of the data type BOOL consist of one bit. Variables of the data types BYTE and WORD are bit sequences of 8 or 16 bits. The individual bits are not evaluated in these data types. Special forms of these data types are the BCD numbers and the count value as it is used in conjunction with the count function.

Note:

In the illustration above, #16# denotes that the value is represented in Hexadecimal format.

INT, REAL

Variables of these data types represent numbers with which relevant arithmetical calculation operations can be carried out. (INT → integer, REAL → floating point number)

Extensions of INT, REAL and WORD

U - Unsigned

Variables with the extension "U" represent a variable without sign of the relevant data type. **Data types:** USINT, UINT, ULINT, UDINT

S - Short

Variables with the extension "S" represent a variable with a length of 8 bits of the relevant data type. **Data types:** SINT, USINT

D - Double

Variables with the extension "D" represent a variable with a length of 32 bits of the relevant data type. **Data types:** DWORD, DINT, UDINT

L - Long

Variables with the extension "L" represent a variable with a length of 64 bits of the relevant data type. **Data types:** LWORD, LINT, ULINT, LREAL

5.2.1. Elementary Data Types

Date, Time and Character

	Description	Size (Bit)	S7-1200	S7-1500	Example
Time Types	TIME	32			T#2h46m30s630ms
	DATE	16	✓	✓	D#1994-01-21
	TIME_OF_DAY	32			TOD#18:15:18:999
	S5TIME	16			S5T#1h24m10s
	LTime		✗	✓	LT#11350d20h25m14s830ms652µs315ns
	LTIME_OF_DAY	64			LTOD#10:20:30.400_365_215
	LDT (DATE_AND_LTIME)				LDT#2008-10-25-08:12:34.567
Character Type	CHAR	8	✓	✓	'R'
	WCHAR	16			WCHAR#'w'

TIME, LTIME

A variable of the data type TIME (duration in [ms]) occupies a double-word. This variable is used, for example, for specifying time values in IEC timer functions. The contents of the variable are interpreted as a DINT number in milliseconds and can be either positive or negative (for example: T#1s=L#1000, T#24d20h31m23s647ms = L#2147486470).

Just like TIME, LTIME represents a duration whereby the duration is saved with nanosecond resolution in 64 bits. That means, compared with the data type TIME, longer durations with greater resolution can be saved in variables of the data type LTIME.

DATE

A variable of the data type DATE is stored in a word in the form of an unsigned integer. The contents of the variable represent the number of days since 01.01.1990.

TIME_OF_DAY, LTIME_OF_DAY

The data type TOD (TIME_OF_DAY) occupies a double-word and stores the number of milliseconds since the beginning of the day (0:00 o'clock) as an unsigned integer. Variables of the data type LTOD occupy two double-words and state the number of nanoseconds since the beginning of the day.

LDT (Date_AND_LTIME)

The data type LDT (DATE_AND_LTIME) occupies 8 bytes and stores information on date and time in nanoseconds since 01.01.1970 0:00.

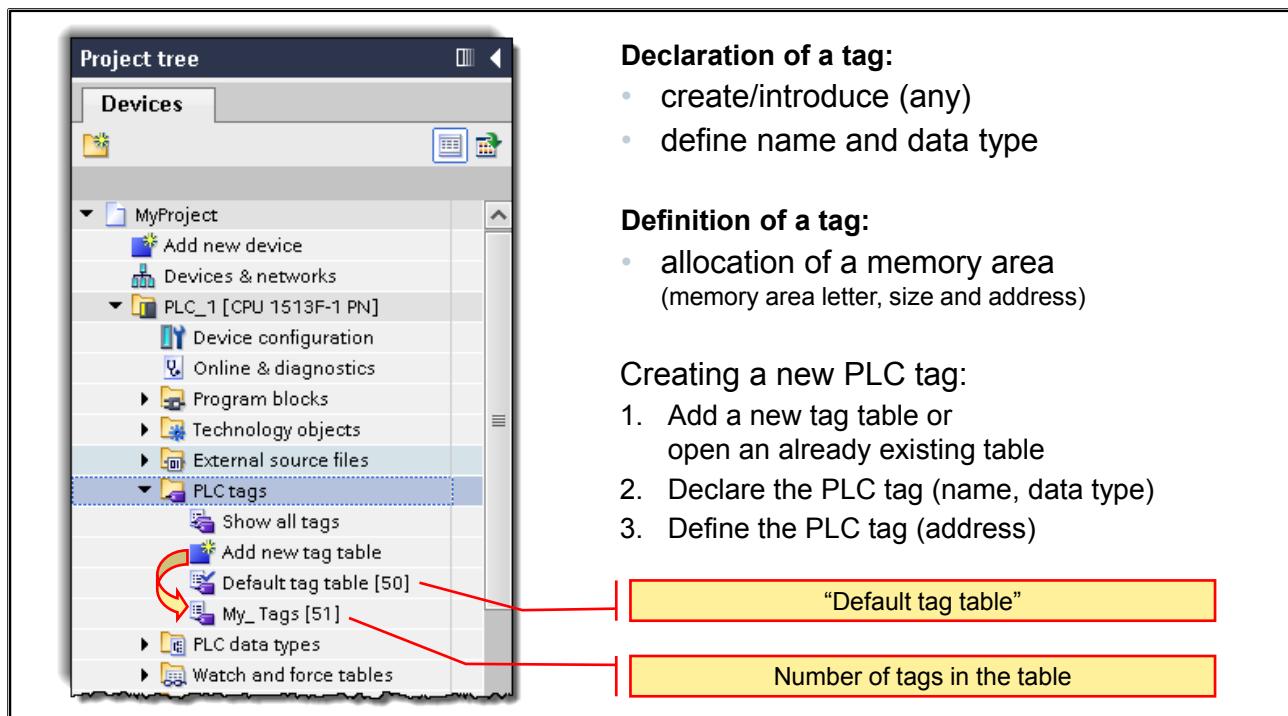
S5TIME

Variables of the data type S5TIME are required for specifying time values in timer functions (S5-timing elements). You specify the time in hours, minutes, seconds or milliseconds. You can enter the timer values with an underline (1h_4m) or without an underline (1h4m).

CHAR, WCHAR

The data type CHAR (8 bits) represents a character in ASCII representation and WCHAR (16 bits) a character in Unicode format.

5.3. Declaration and Definition of PLC Tags



PLC Tag Tables

In order to achieve a good readability of the CPU program, it makes sense to structure the data storage. For this, there are PLC tag tables for the PLC tags.

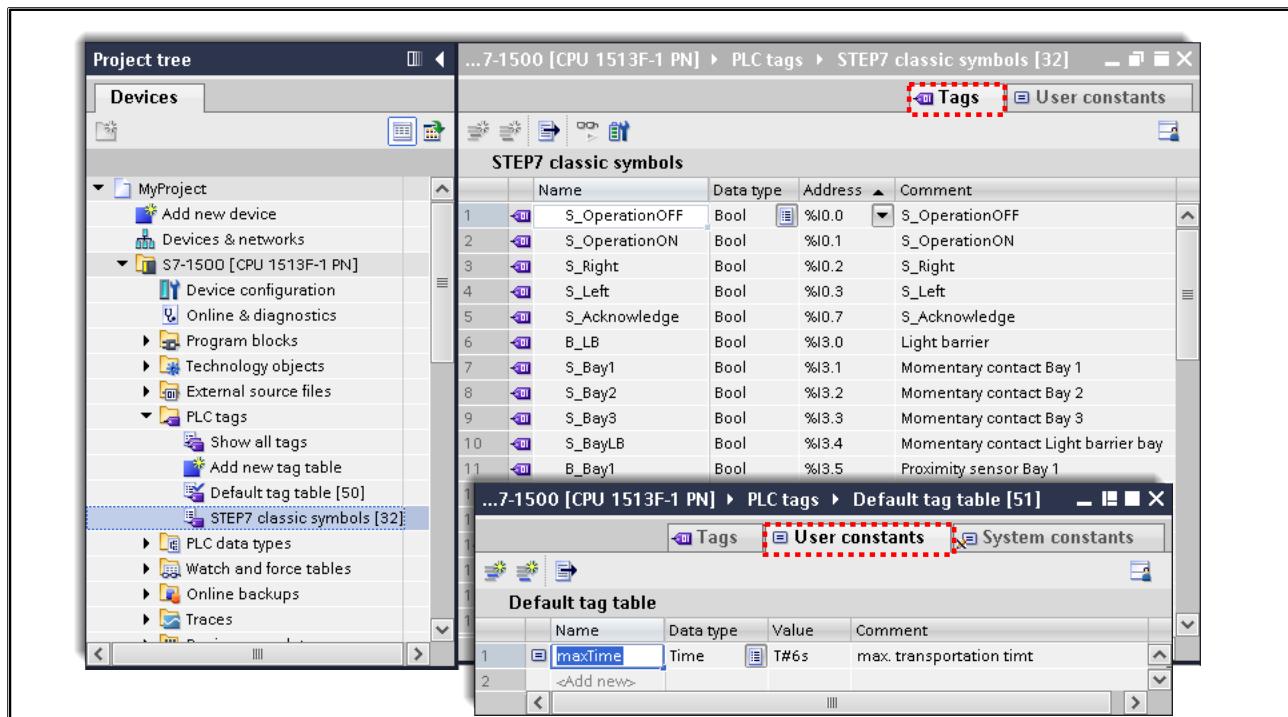
Tag Tables

The default tag table (name can be changed) contains additional CPU information and thus cannot be deleted. It can be used for any PLC tags, but for better clarity, it is recommended that several tag tables be created.

Declaration and Definition of PLC Tags

In the declaration of a tag in the PLC tag table, the symbolic name (for example, "M_Jog_Right"), the data type (for example, Bool) and the absolute address (for example, M16.2) are defined.

5.3.1. PLC Tags and PLC Constants



PLC Tags

The PLC tag table contains the declaration (definition) of CPU-wide valid and thus global tags and constants. For each CPU added in the project, a PLC tag table is automatically created. A PLC tag table contains one tab each for Tags and User constants; the Default tag table also contains a tab for System constants. Tags are operands with changeable content used in the user program.

User Constants

A constant defines an unchangeable data value. During program execution, constants can be read by various program elements, but they cannot be overwritten. Changing the constant value while the program is running is not possible.

In TIA Portal, it is possible to declare symbolic names for constants so as to make static values available in the program under one name. These symbolic constants are valid throughout the CPU. The declaration of the constants is made in the "User constants" tab of the PLC tag table. Constants are operands with unchangeable content, and in addition, constants do not require an absolute address.

Creating Tags and Constants with Group Function

By clicking on the "Fill" symbol in the lower right corner of the cell and then dragging it down, tags and constants are automatically created (comparable to Excel).

It is possible to automatically create tags and constants through the "Name" and "Address" (only for tags) columns. The new tags and constants are created with the name of the current tag/constant appended by a consecutive number. From a tag or constant with the name "T_Station", new tags/constants are then created with the names "T_Station_1", "T_Station_2" etc.

System Constants

System constants are CPU-wide unique, global constants which are required by the system and are automatically created. System constants can, for example, serve the addressing and identification of hardware objects.

Rules

System constants are automatically assigned in the Device view or Network view when components are inserted and are entered in the Default tag table ("System constants" tab). A system constant is created for each module but also for each submodule. In that way, for example, an integrated counter is also given a system constant. System constants consist of a symbolic name as well as a numeric HW-ID and cannot be changed.

System Constant Names

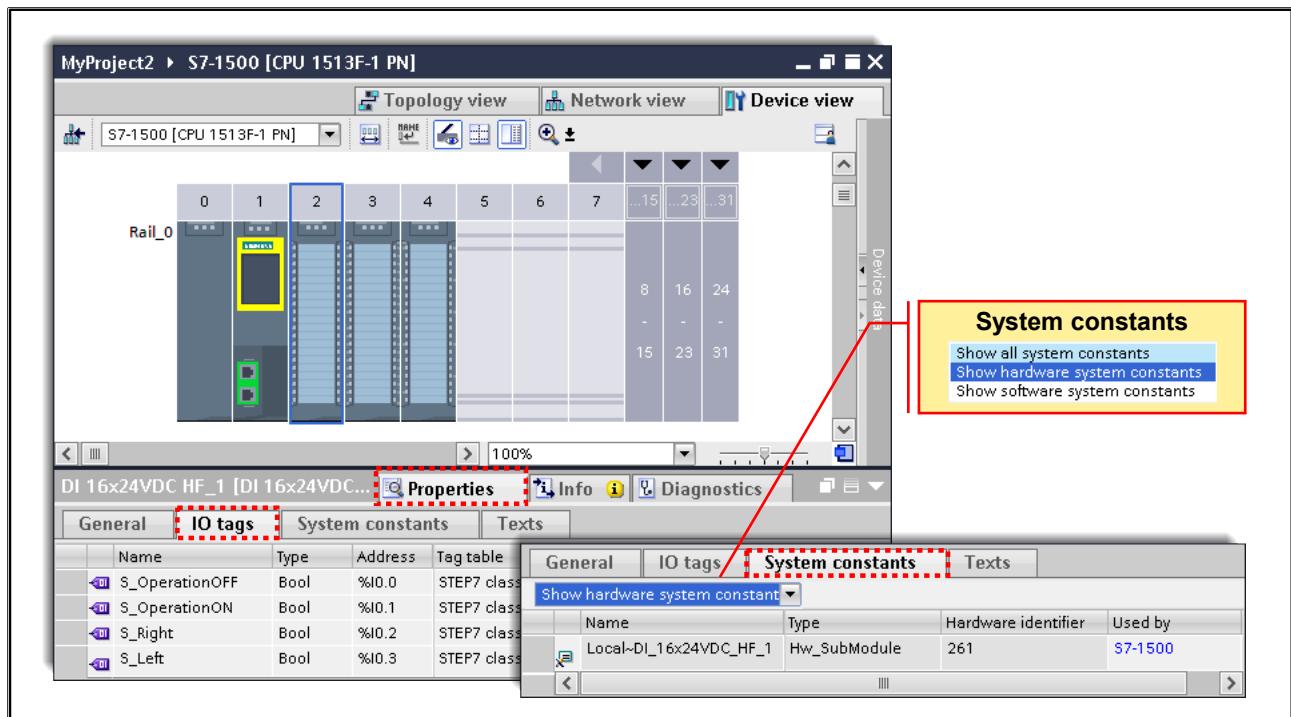
The names of system constants are hierarchically structured. They consist of a maximum of four hierarchy levels which in each case is separated by a tilde "~". In this way, you can recognize the "path" to the relevant hardware module based on the name.

	Name	Data type	Value	Comment
39	Local~Display	Hw_SubModule	54	
40	Local~Exec	Hw_SubModule	52	
41	Local	Hw_SubModule	49	
42	Local~FExec	Hw_SubModule	55	
43	Local~PROFINET_interface_1	Hw_Interface	64	
44	Local~PROFINET_interface_1~Port_1	Hw_Interface	65	
45	Local~PROFINET_interface_1~Port_2	Hw_Interface	66	

Example

A system constant with the name "Local~PROFINET_interface_1~Port_1" denotes Port 1 of the PROFINET interface 1 of the local CPU.

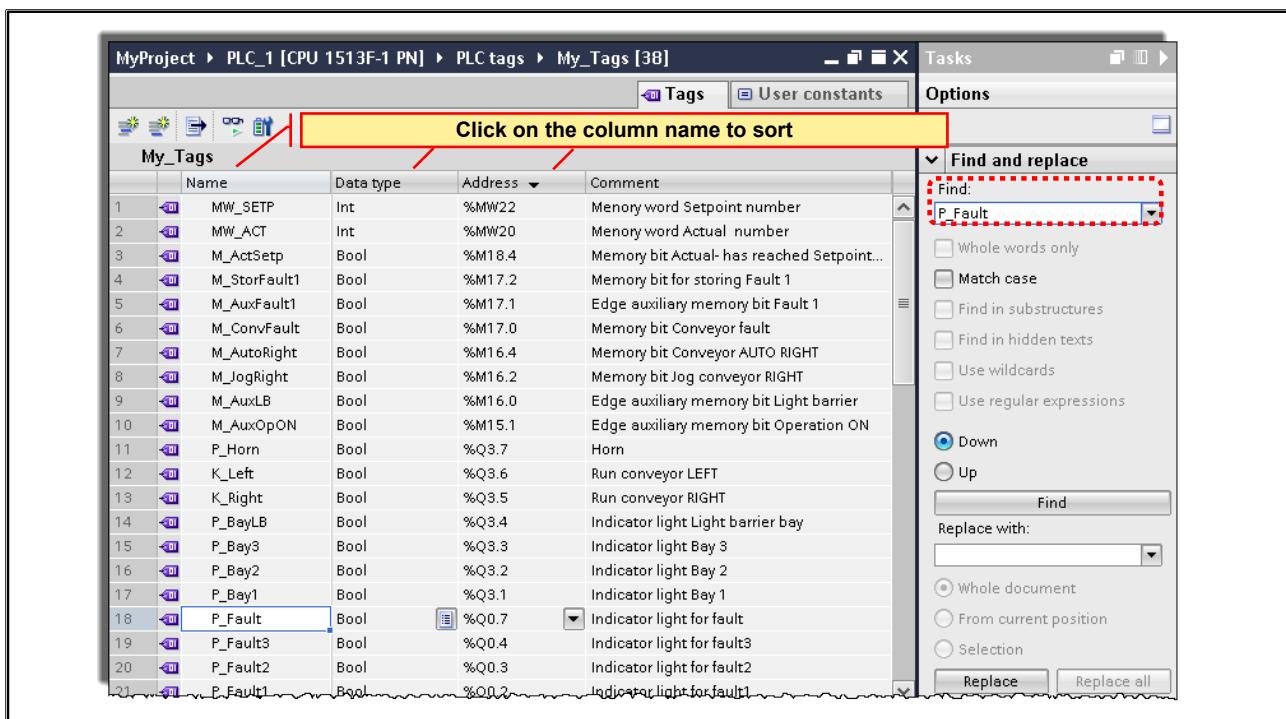
5.3.2. PLC Tags in the Device View



The PLC tags of inputs and outputs can also be declared and changed in the Device view. In the Properties, in the Inspector window you open the IO tags tab for this.

In addition, the system constants of the selected hardware are displayed in the "System constants" tab.

5.3.3. Finding / Replacing / Sorting PLC Tags



Sorting

By clicking on one of the column names "Name", "Data type" or "Address", the tags are sorted alphabetically or according to address (ascending or descending) depending on the column.

Finding / Replacing

In the PLC tag table, tags can be found and replaced via the "Tasks" Task Card. Dummies can also be used (? for one character, * for several characters).

Example of "Find and replace":

Assign byte address 4. to all outputs with byte address 8.:

Find: Q 8. and Replace with: Q 4.

5.3.4. Error Indication in the PLC Tag Table

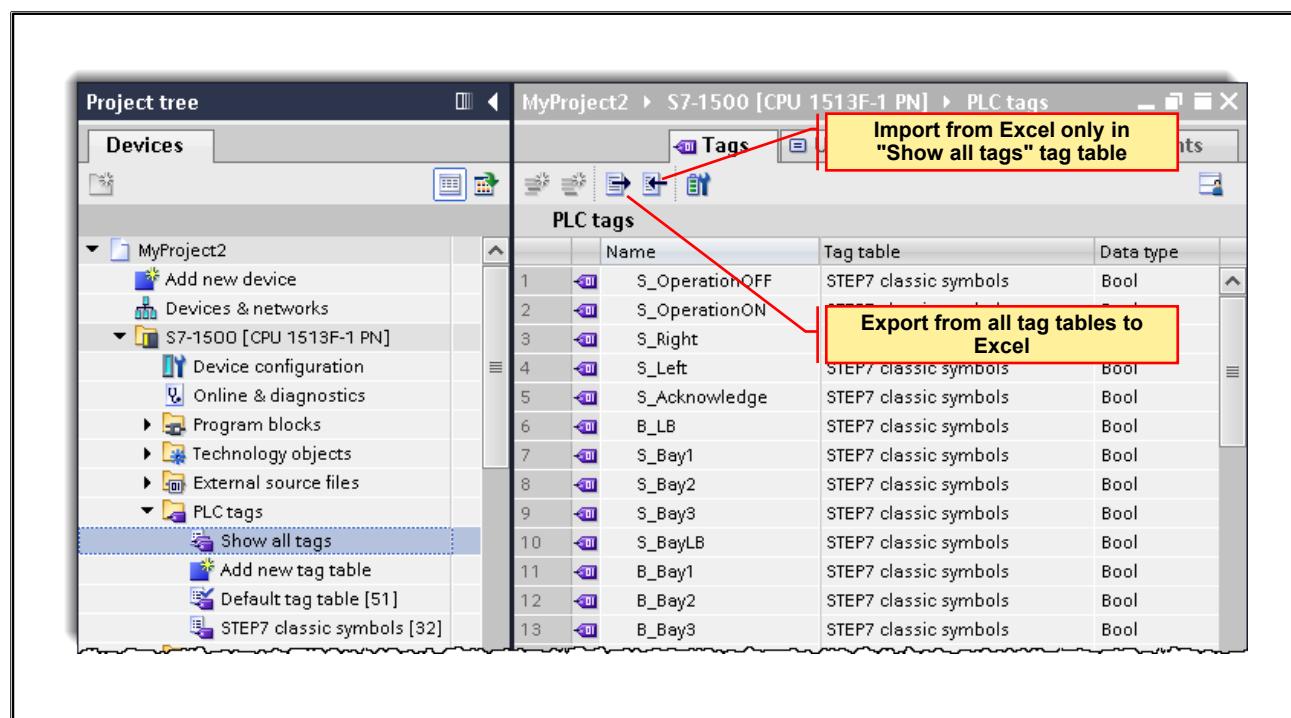
The screenshot shows a 'Tags' table in the SIMATIC TIA Portal. The table has columns for Name, Data type, Address, Retain, Accessible from HMI/OPC UA, Writable fr..., and Visible in HMI engineering. Several rows contain errors:

- Row 3: 'B_LB' is highlighted with a red dashed box. A yellow box with a red arrow points to it, stating: "If the name (symbol) already exists in the table, a '(1)' is automatically added".
- Row 5: 'K_Left' has a red exclamation mark (!!) next to its address '%Q3.6'. A yellow box with a red arrow points to it, stating: "The absolute address is not compatible with the data type of the tag!".
- Row 17: 'P_Bay3' has a red exclamation mark (!!) next to its address '%I3.3'. A yellow box with a red arrow points to it, stating: "Absolute address is used twice!".
- Row 18: 'P_BayLB' also has a red exclamation mark (!!) next to its address '%I3.3'.

Syntax Check

With every entry, there is a syntax check in which existing errors are displayed in RED, or for warnings in YELLOW. A still faulty PLC tag table can be saved but as long as it is still faulty, the program cannot be compiled and downloaded into the CPU.

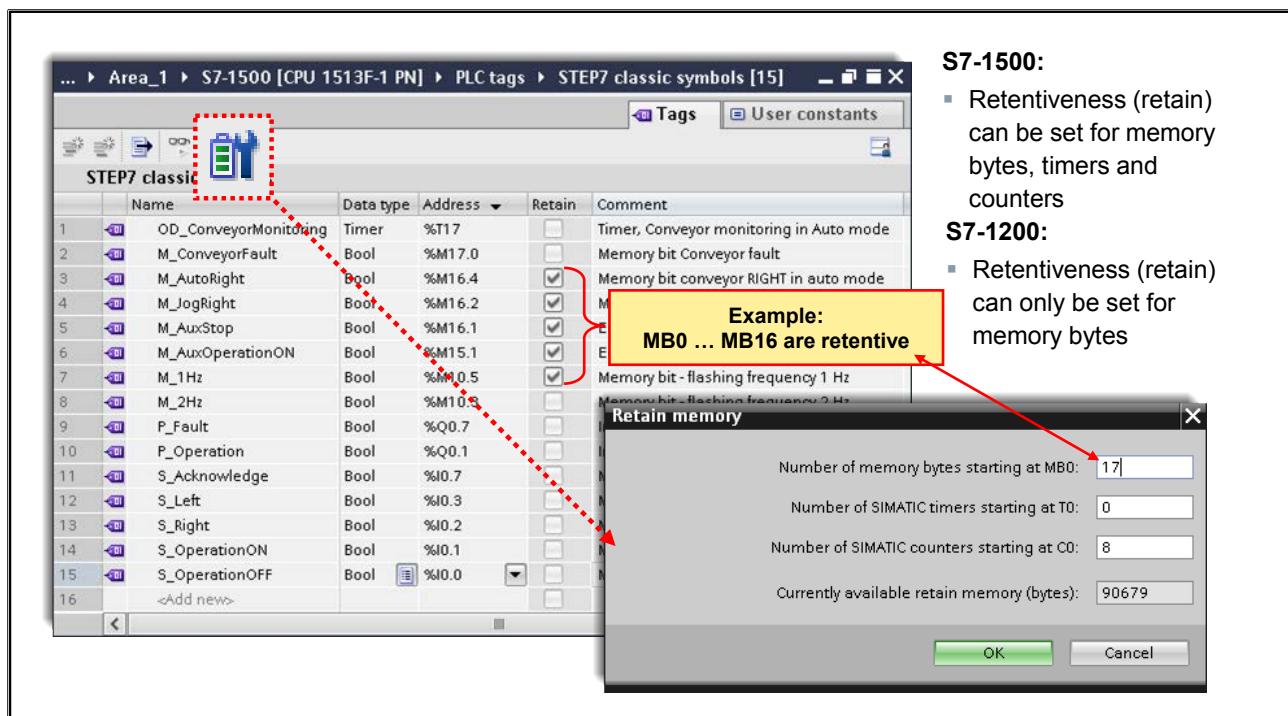
5.3.5. Copy & Paste PLC Tags to Excel



Copy & Paste from and to Excel

The Windows Copy & Paste function as well as the Import / Export function can be used to easily copy individual or several tags from a PLC tag table or a data block to Excel to further process it/them there and then to copy it/them back from Excel to the PLC tag table or the data block.

5.3.6. Retentiveness of PLC Tags



Retentive (Retain) Memory

The S7-1500 CPUs have a retentive memory for storing retentive data when the power is switched OFF. The size of the retentive memory is documented in the technical data of the CPU.

The utilization of the retentive memory of the configured CPU is shown offline in the Project tree under "Program info > Resources of..." or online in the Project tree under "Online & diagnostics > Diagnostics > Memory".

When you define data as retentive, their contents are retained after a power failure, during CPU start-up and during loading of a modified program.

You can define the following data or objects as retentive:

- Memory bytes, timers, counters
- Tags of global data blocks
- Tags of instance data blocks of a function block

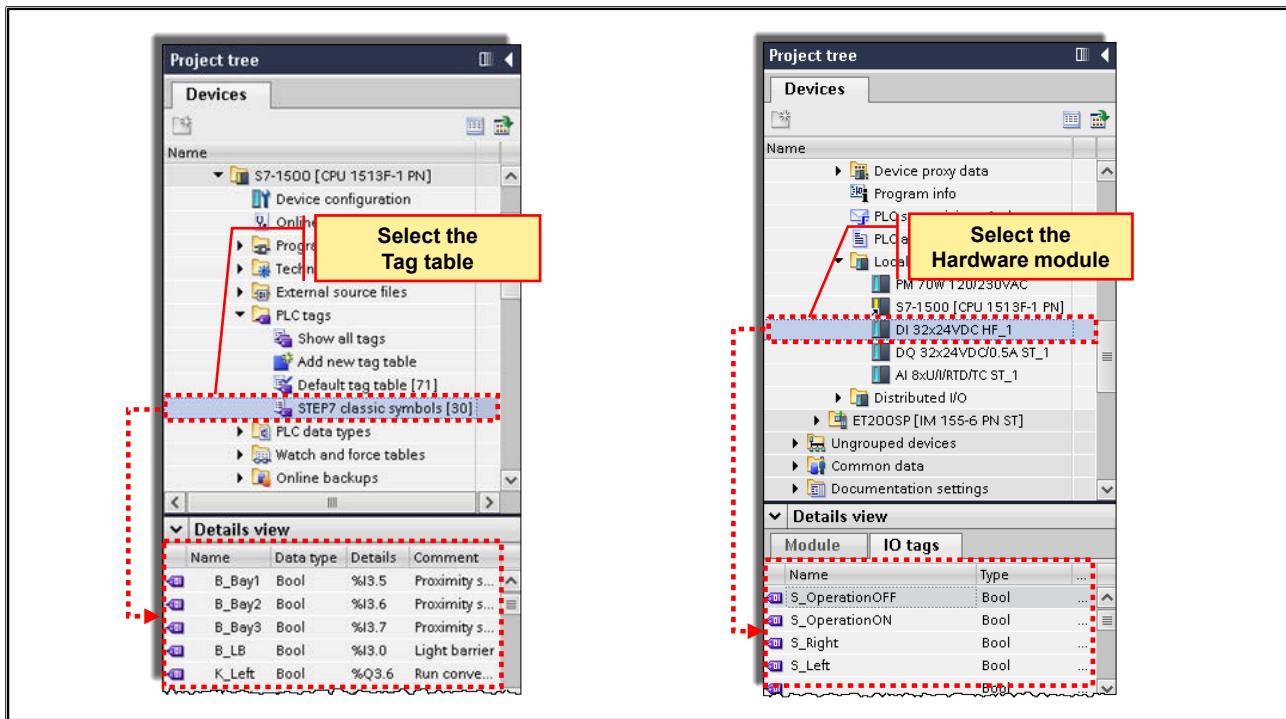
Certain tags of technology objects are always retentive, for example, adjustment values of absolute value encoders.

Memory Bytes, Timers, Counters

For the S7-1500, the number of retentive memory bytes, timers and counters can be defined in the PLC tag table via the "Retain" button.

For the S7-1200, only the number of retentive memory bytes can be defined in the PLC tag table via the "Retain" button.

5.4. Details View of PLC Tags



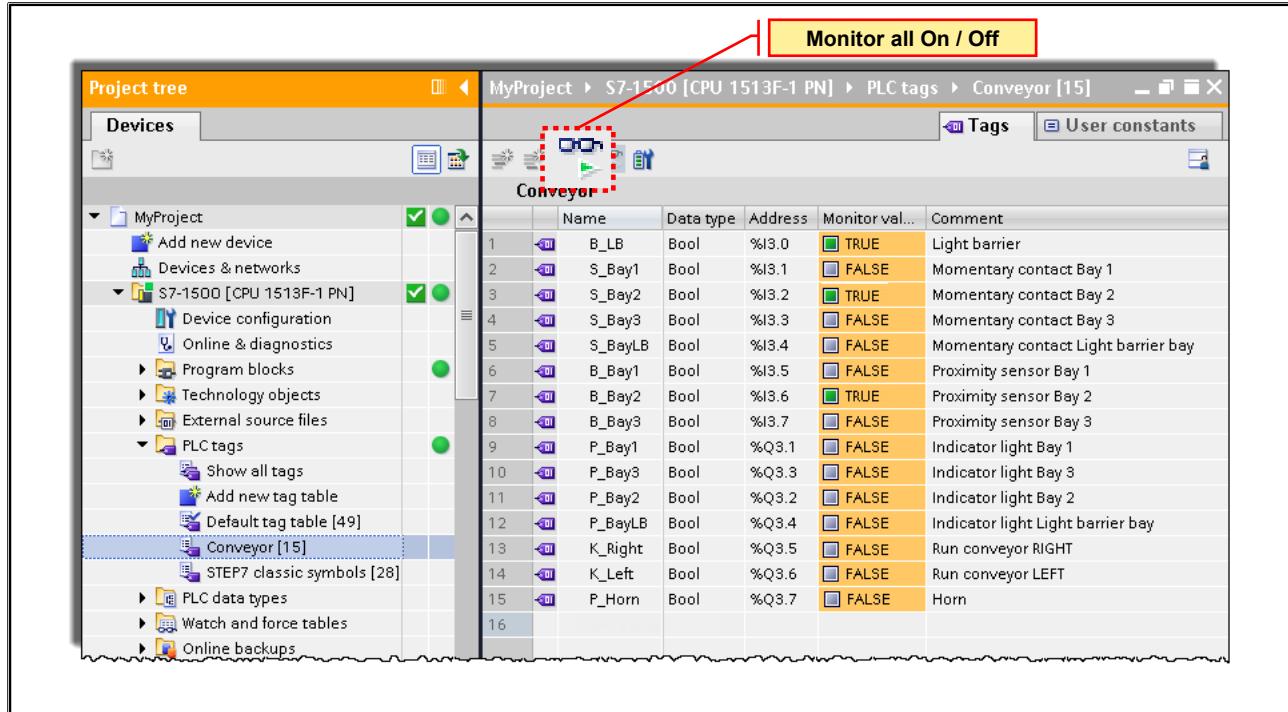
The Details view shows the tags of the object selected (highlighted) in the Project tree:

For example:

- ...tags of the selected tag table
- ...channels of the selected local modules and their tags

That way it is easy for the user, using drag & drop, to integrate tags in the user program, for example.

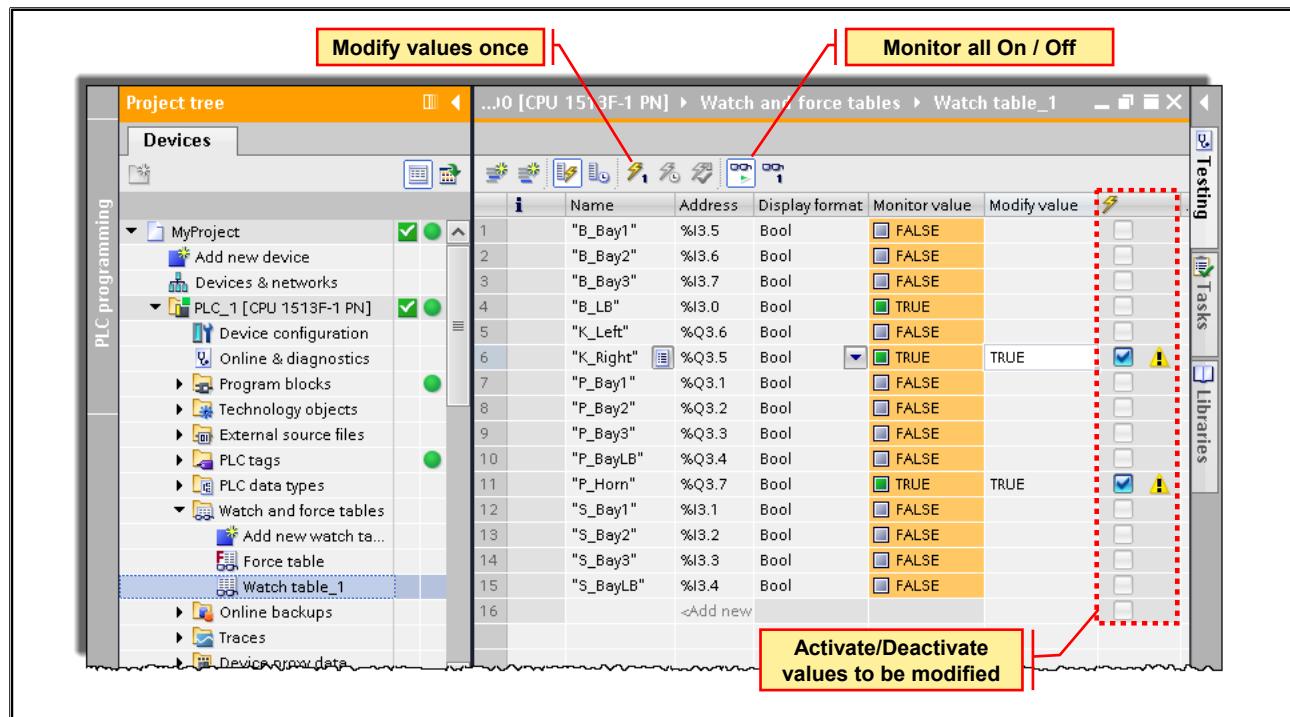
5.5. Monitoring PLC Tags



Monitoring PLC Tags

PLC tags can be monitored directly through the PLC tag table. In so doing, the "Monitor value" shows the current value of the tags in the CPU.

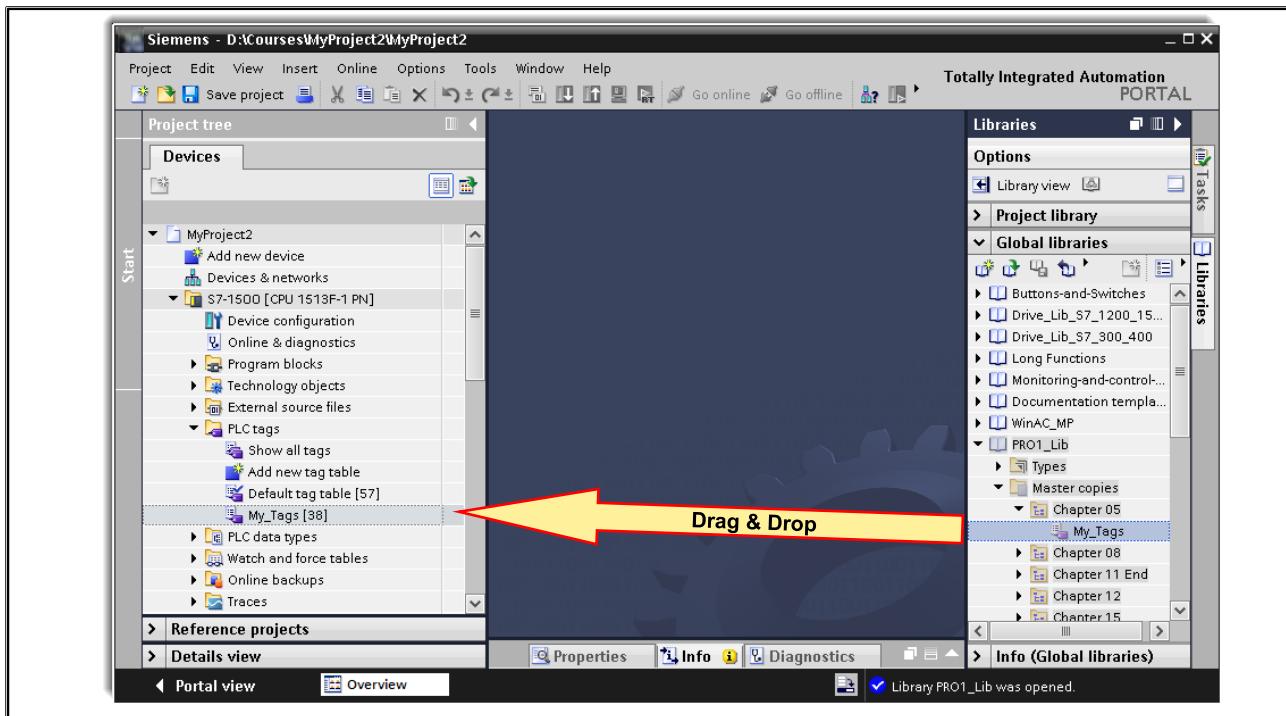
5.5.1. Modifying PLC Tags by means of the Watch Table



In order to be able to modify tags (variables) you require a watch table which you can create in the 'Watch and force tables' folder.

Any tags (values) can be monitored and modified in a Watch table. To modify a tag, a modify value is specified, the tag to be modified is activated (is automatically activated during creation) and by means of the button "Modify all selected values once and now" the values of the activated tag are loaded into the controller.

5.6. Exercise 1: Copying the PLC Tag Table from the Library



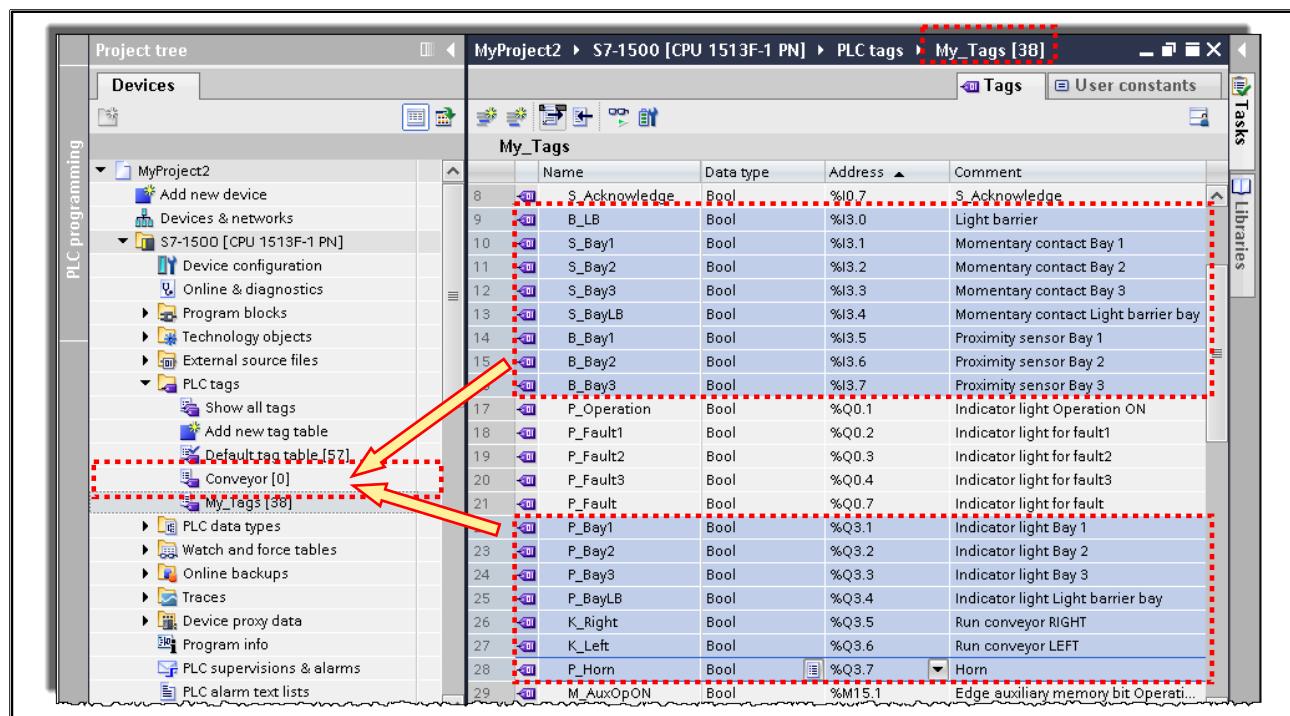
Task

You are to copy the prepared PLC tag table "My_Tags" from the "PRO1_Lib" library into your own project.

What to Do

1. In the Task Card "Libraries > Global libraries", open the library "PRO1_Lib" which is located in the folder C:\02_Archives\TIA_Portal\TIA-PRO1 of your programming device.
2. Using drag & drop, copy the PLC tag table "My_Tags" from the Libraries' folder Master copies\Chapter 05 into your project's folder PLC tags.
3. Save your project.

5.6.1. Exercise 2: Creating the "Conveyor" Tag Table



Task

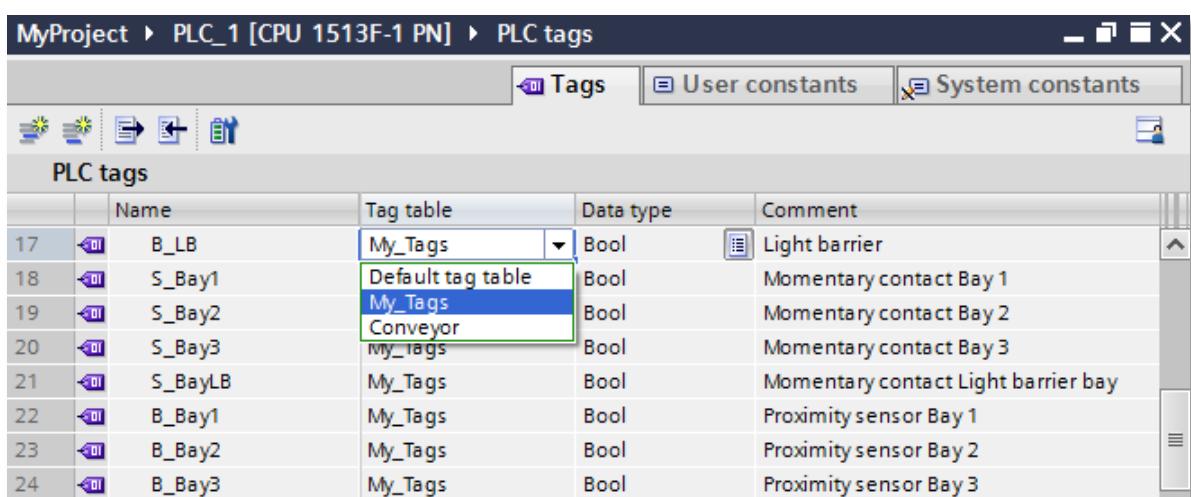
You are to create a separate PLC tag table "Conveyor" for the inputs and outputs to which the sensors and actuators of the conveyor model are connected.

What to Do

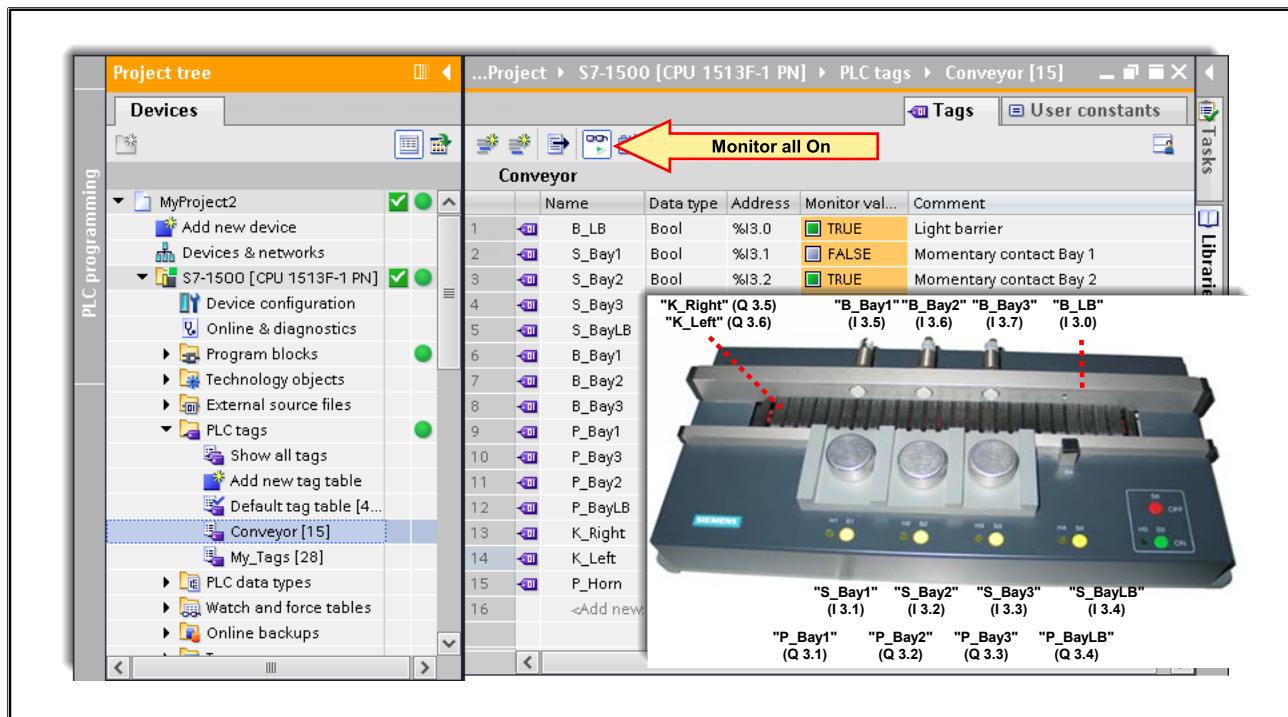
1. Using "Add new tag table", create the new PLC tag table "Conveyor".
2. Open the PLC tag table "My_Tags" and move the inputs and outputs into the newly created PLC tag table "Conveyor" using drag & drop (mouse pointer on tag icons, see picture).
3. Save your project.

Note:

As an alternative, for every tag you can define in which tag table it is to be saved. You do this in the tag table "Show all tags" via the column "Tag table".



5.6.2. Exercise 3: Monitoring the "Conveyor" PLC Tag Table



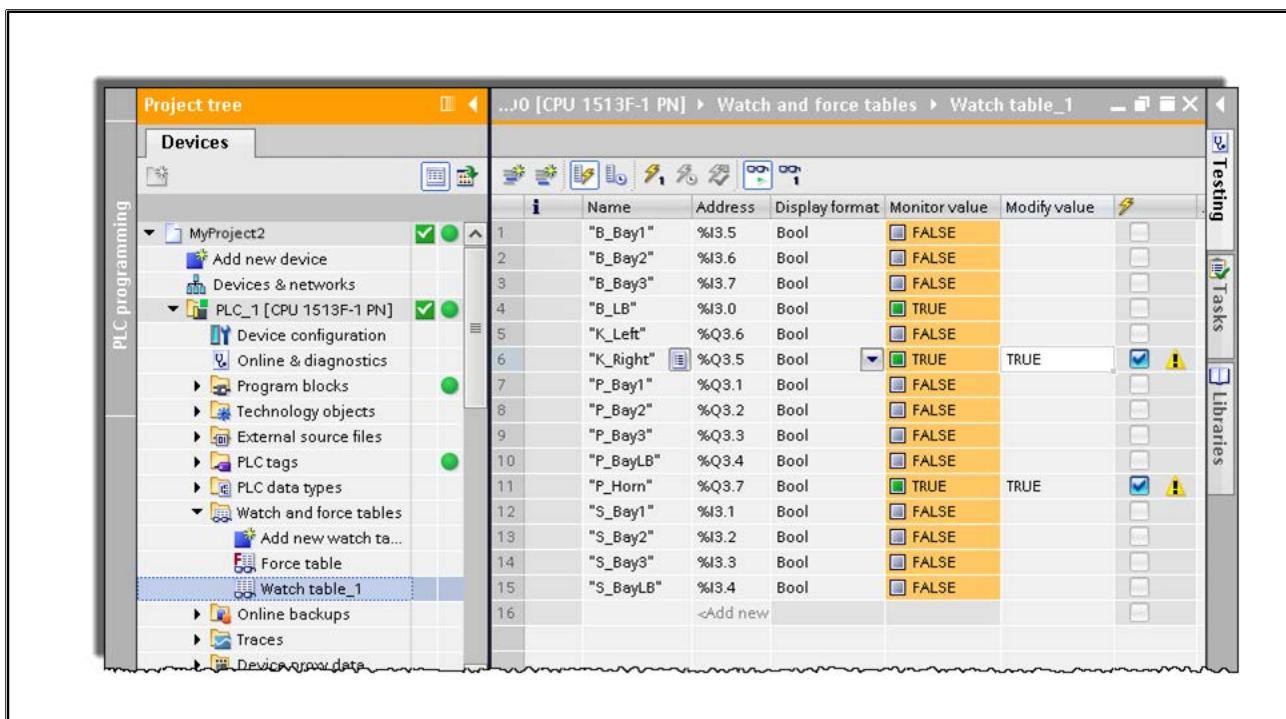
Task

You are to monitor the inputs in the tag table.

What to Do

1. In order to adopt the changes and so that the CPU can switch to RUN, compile and download the software.
2. Check whether the broadband cable of your conveyor model is connected to the "S7-1500 DI/DO" socket on the back of your training case.
3. Switch your conveyor model on.
4. Open the newly created PLC tag table "Conveyor" and activate the function "Monitoring".
5. On the conveyor model, press the Bay pushbuttons and check whether Status '1' or "TRUE" is displayed at the corresponding inputs.
6. Save your project.

5.6.3. Exercise 4: Modifying using the Watch Table



Task

You are to modify the outputs with the help of a watch table.

What to Do

1. In the Watch and force tables folder, add a new watch table.
2. Open the newly created watch table.
3. In the Project tree, select the PLC tag table "Conveyor".
4. From the Detail view, copy the PLC tags of the "Conveyor" PLC tag table into your watch table using drag & drop.
5. For some of the outputs, specify the Modify value TRUE, that is, 1.
6. Check whether the relevant tags are activated for modifying.
7. Activate the function "Monitor all" and modify the outputs.
8. Check whether the relevant outputs are activated.
9. For the outputs, specify the Modify value FALSE, that is, 0 in order to switch them off again.
10. Modify the outputs once again.
11. Save your project.

Contents

6

6.	Binary Operations 1	6-2
6.1.	Plant Description: The Conveyor Model as a Distribution Conveyor.....	6-3
6.2.	Assignment	6-4
6.2.1.	Sensors and Check Symbols.....	6-5
6.2.2.	Binary Logic Operations: AND, OR and Negation.....	6-6
6.2.3.	Theory Exercise 1: Sensor and Check Symbols	6-7
6.2.4.	Binary Logic Operations: Exclusive OR (XOR).....	6-8
6.3.	Process Images	6-9
6.4.	Cyclic Program Execution.....	6-10
6.5.	Linear Program	6-11
6.6.	Programming.....	6-12
6.6.1.	Networks	6-13
6.6.2.	Absolute and Symbolic Addressing	6-14
6.6.3.	Using a PLC Tag as an Operand.....	6-15
6.6.4.	Renaming / Rewiring PLC Tags.....	6-16
6.6.5.	Defining (Declaring) Tags while Programming	6-17
6.6.6.	Closing / Saving / Rejecting a Block	6-18
6.6.7.	Compiling a Program	6-19
6.6.8.	Downloading a Program into the CPU.....	6-20
6.6.9.	Monitoring a Block	6-21
6.7.	Exercise 1: Renaming Main OB.....	6-22
6.7.1.	Exercise 2: Programming the Jog Mode.....	6-23
6.7.2.	Exercise 3: Compiling the Program, Downloading it into the CPU	6-24
6.7.3.	Exercise 4: Monitoring the Program	6-25

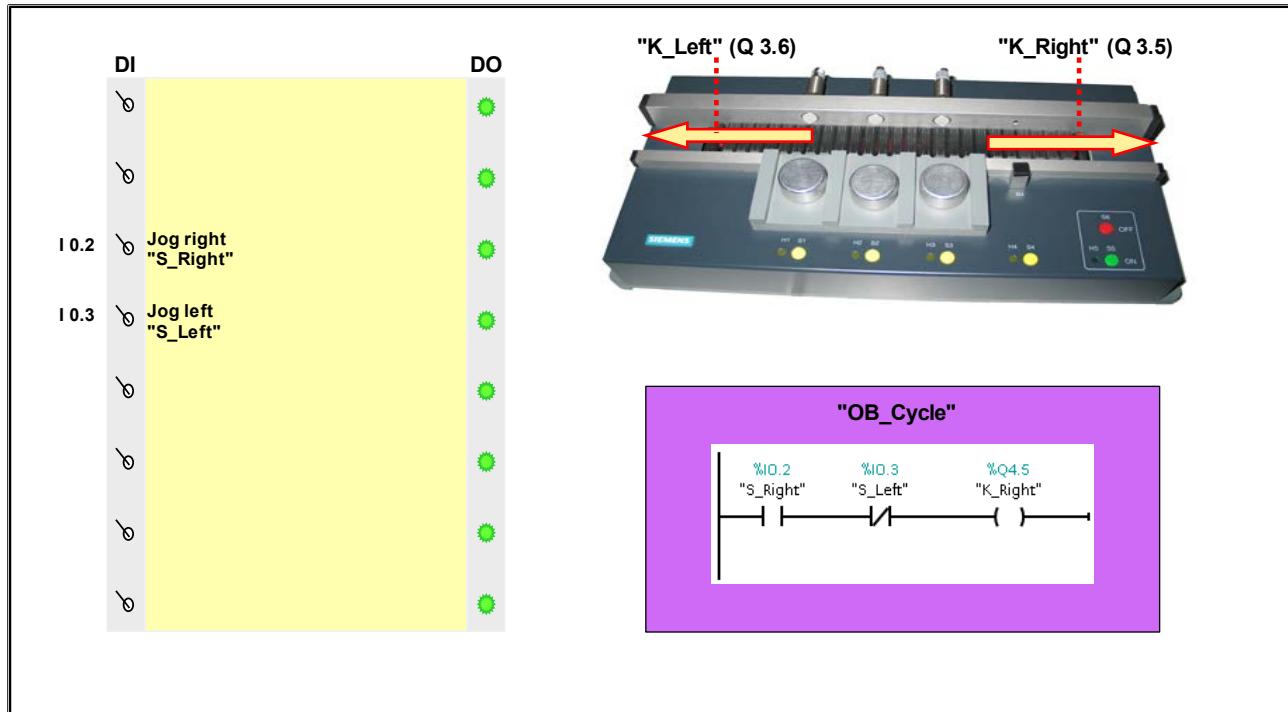
6. Binary Operations 1

At the end of the chapter the participant will ...

- ... know what an assignment is
- ... understand the difference between 'real' NC contacts and NO contacts connected in the hardware, and programmed check symbols
- ... be familiar with the logic operations AND, OR and Exclusive-OR and be able to apply them
- ... be familiar with the process image for inputs and outputs
- ... understand cyclic program execution
- ... know what a linear program is
- ... be familiar with the program editor
- ... be able to rename and rewire PLC tags
- ... be able to carry out a program test with the "Monitor (block)" function



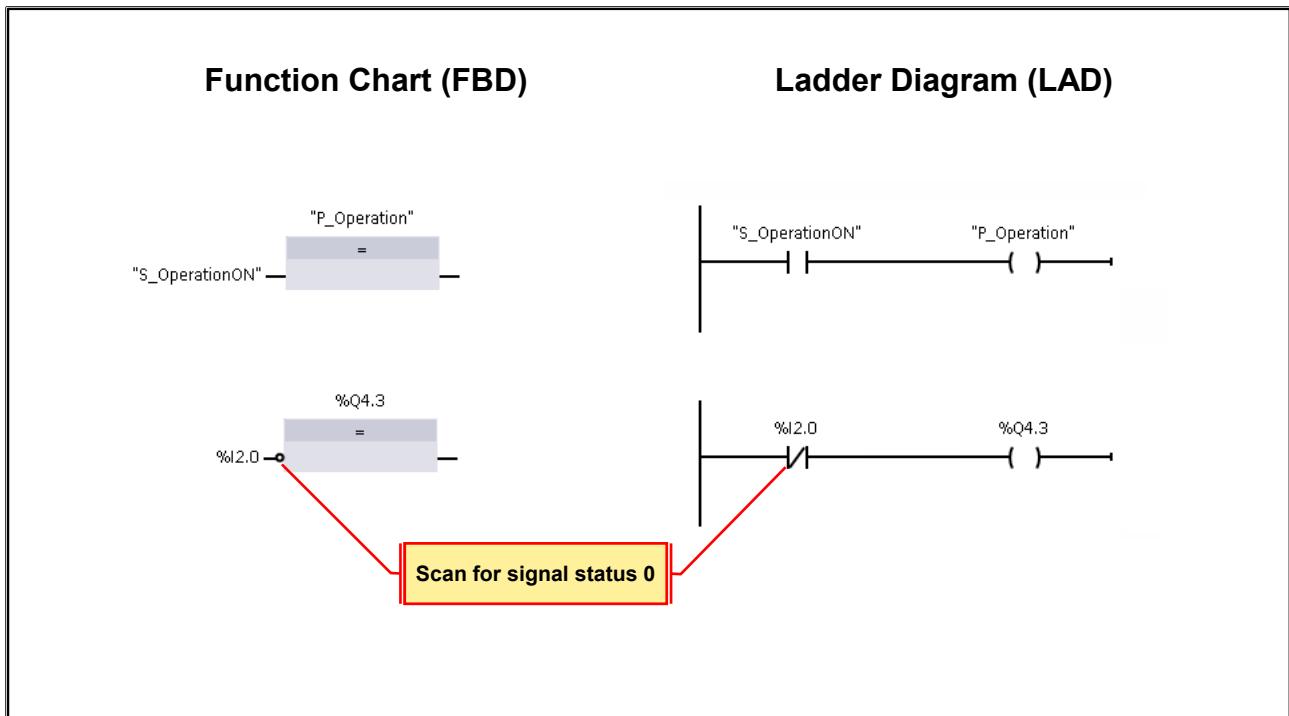
6.1. Plant Description: The Conveyor Model as a Distribution Conveyor



Conveyor Model as a Distribution Conveyor

The distribution conveyor can be jogged to the left and to the right using the switches "S_Right" (I 0.2) and "S_Left" (I 0.3). If both switches are activated simultaneously, then the conveyor must not move (Lock-out!).

6.2. Assignment



Assignment

With an assignment, the specified operand is always assigned the current RLO as status. The assigned RLO remains available after the assignment and can be assigned to a further operand or it can be further logically linked.

Scan for 0 (False)

The scan for 0, that is, the signal status FALSE is fulfilled when the operand has the signal status '0'.

6.2.1. Sensors and Check Symbols

Process			Interpretation in the PLC Program					
The sensor is a ...	The sensor is ...	Voltage present at input?	Signal status at input	Scan for signal status "1"		Scan for signal status "0"		
				Symbol / Instruction	Result of check	Symbol / Instruction	Result of check	
NO contact	activated		yes	1	LAD: - - "NO contact"	"Yes" 1	LAD: - - "NC contact"	"No" 0
	not activated		no	0		"No" 0		"Yes" 1
NC contact	activated		no	0	FBD: 	"No" 0	FBD: 	"Yes" 1
	not activated		yes	1		"Yes" 1		"No" 0

Sensors of the Process

The use of normally open or normally closed contacts for the sensors in a controlled process depends on the safety regulations for that process. Normally closed contacts are always used for limit switches and safety switches, so that dangerous conditions do not arise if a wire break occurs in the sensor circuit. Normally closed contacts are also used for switching off machinery for the same reason.

All scans can be programmed for signal status, that is, Status '0' and '1' regardless of whether a hardware NO contact or hardware NC contact is connected in the process.

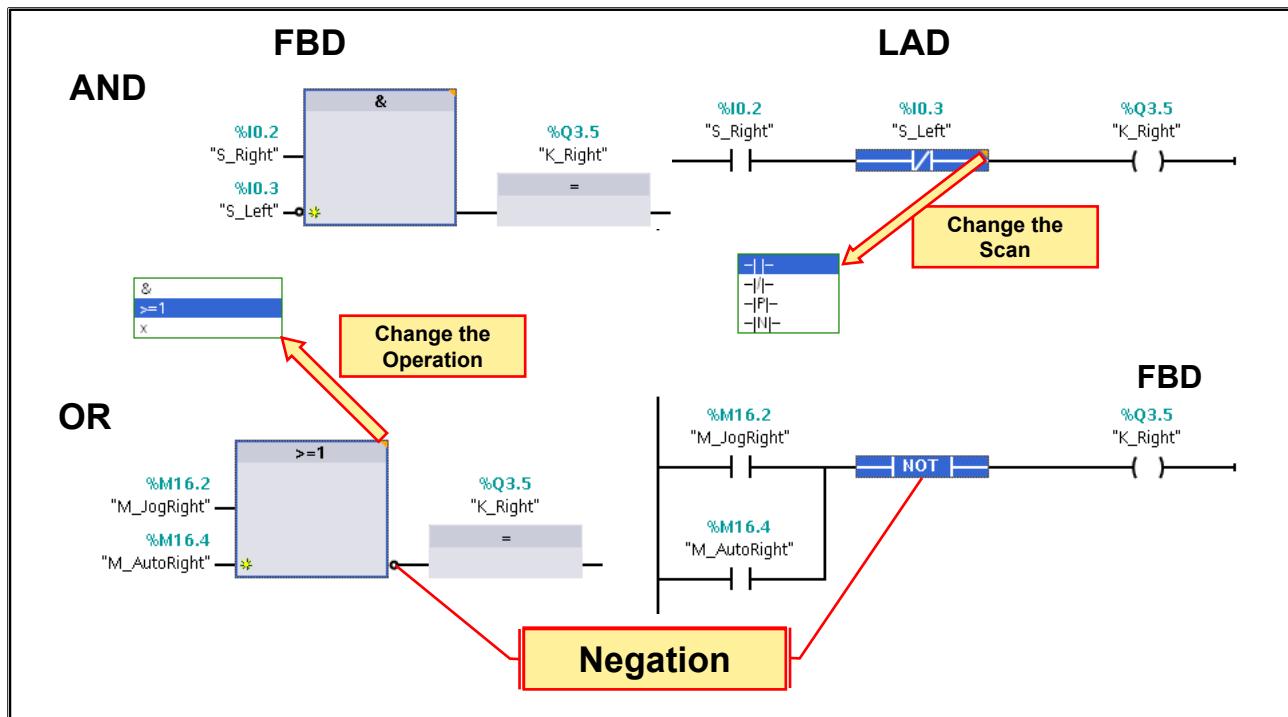
Check Symbols of the Program

In LAD, a symbol with the name "NO contact" is used for checking for signal status "1" and a symbol with the name "NC contact" to check for signal status "0". It makes no difference whether the process signal "1" is supplied by an activated NO contact or a non-activated NC contact.

The "NO contact" symbol delivers the result of check "1" when the scanned operand has Status "1".

The "NC contact" symbol delivers the result of check "1" when the scanned operand has Status "0".

6.2.2. Binary Logic Operations: AND, OR and Negation



AND and OR Logic Operations

With the AND and OR logic operations, basically all binary operands can be scanned, even outputs. Instead of individual operands, the results of other logic operations can also be further logically linked. Also, the logic operations can also be combined.

AND Logic Operation

For an AND logic operation, the result of logic operation (RLO) = '1', when all input signals have Status '1'.

OR Logic Operation

For an OR logic operation, the result of logic operation (RLO) = '1', when at least one input signal has Status '1'.

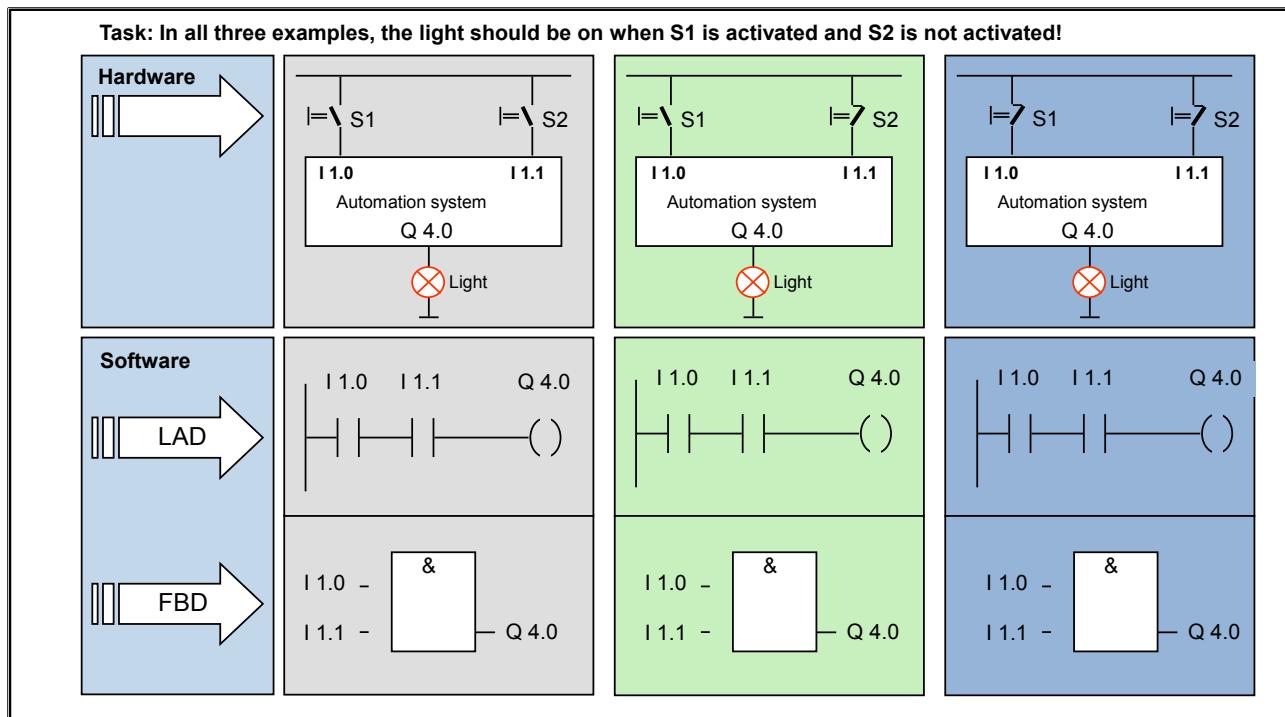
Negation (NOT)

The NOT instruction inverts the result of logic operation (RLO).

If, in the example shown, the RLO of the OR logic operation = '1', the NOT instruction inverts it to RLO '0' and the output is assigned the Status "0".

If the RLO of the OR logic operation = '0', the NOT instruction inverts it to RLO '1' and the output is assigned the Status '1'.

6.2.3. Theory Exercise 1: Sensor and Check Symbols



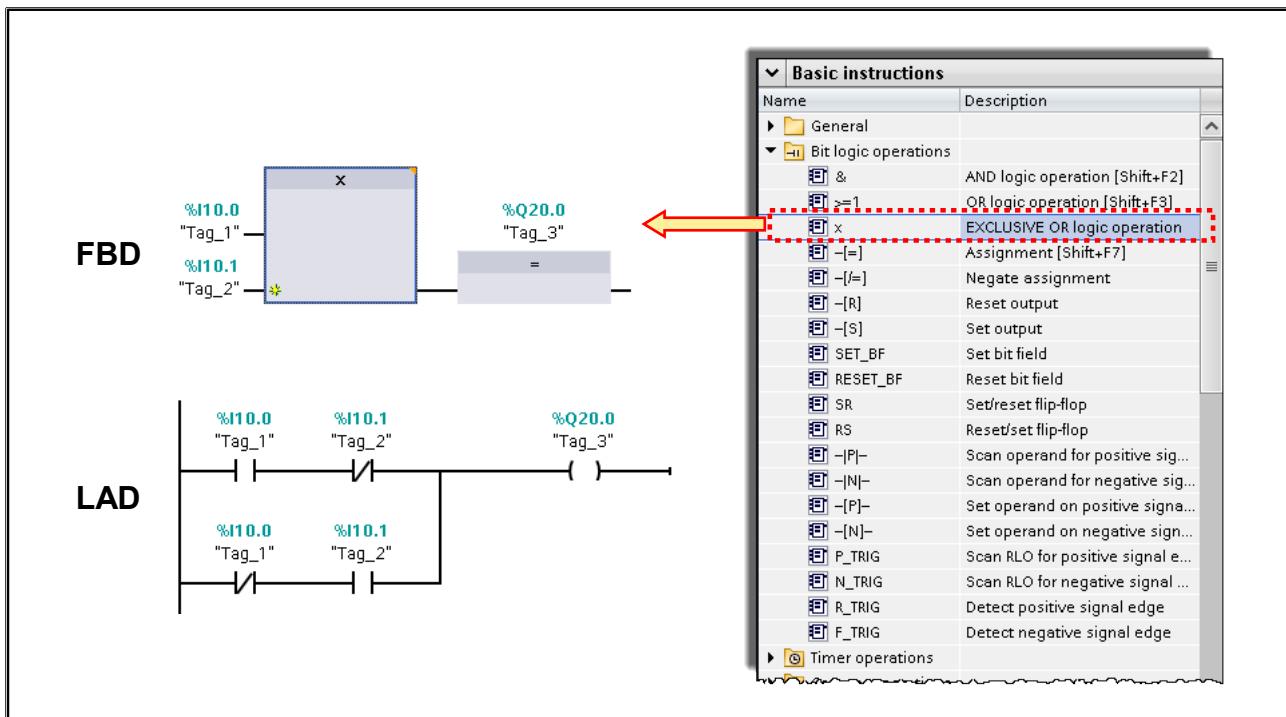
Task

In FBD, complete the logic symbols, and, in LAD correct the check symbols so that the required function is fulfilled.

Note

The terms - "NO contact" and "NC contact" - have different meanings depending on whether they are used in the process-hardware-context or as check symbols in the software.

6.2.4. Binary Logic Operations: Exclusive OR (XOR)



XOR Logic Operation

With the XOR logic operation, basically all binary operands can be scanned, even outputs. Instead of individual operands, the results of other logic operations can also be further logically linked. Also, the logic operations can also be combined.

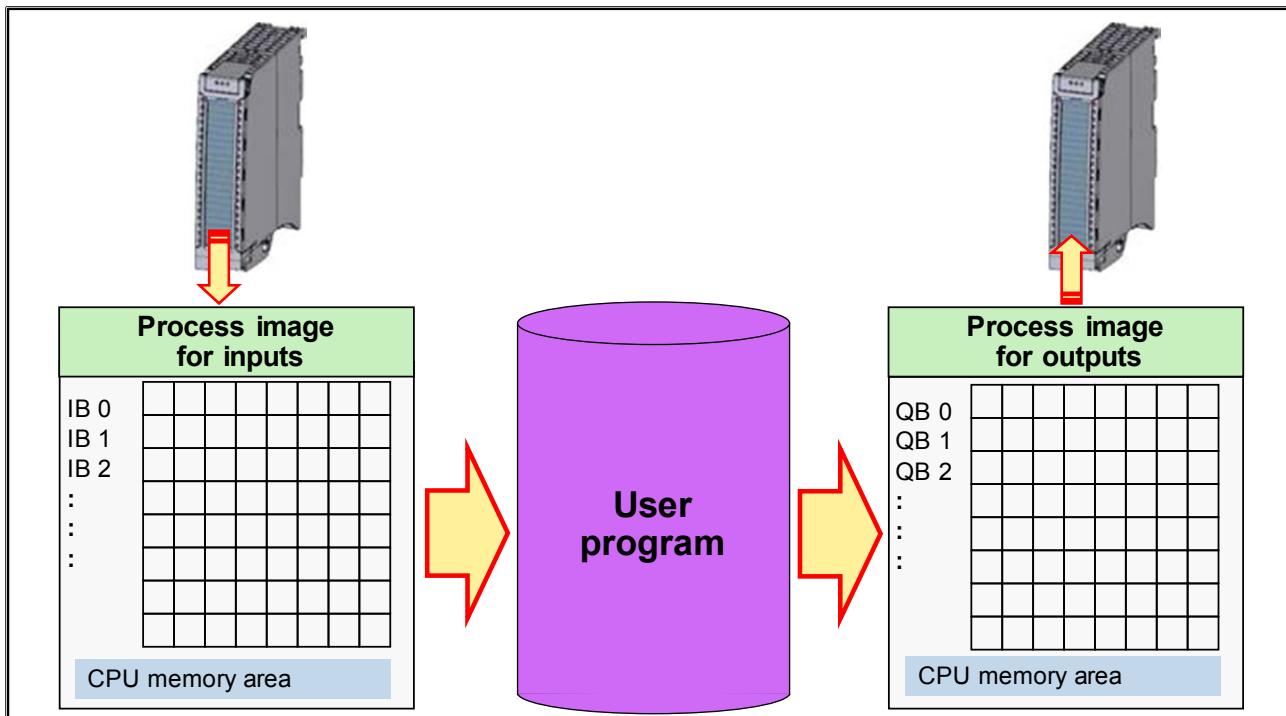
All inputs of the logic operations can be programmed as scan for signal status or Status '0' and '1', regardless of whether a hardware NO contact or NC contact is connected in the process.

- For an XOR logic operation with 2 inputs, the result of logic operation (RLO) = '1', when one and only one of the two input signals has Status '1'.
- For an XOR logic operation with more than 2 operands, the RLO ...
= '1', when an uneven number of checked operands has Status '1'
= '0', when an even number of checked operands has Status '1'.

XOR in the LAD Programming Language

In the LAD programming language, there is no explicit XOR logic operation. It must be generated by programming the discrete instructions shown in the picture above.

6.3. Process Images



Process Images

For the storage of **all** digital input and output states, the CPU has reserved memory areas: the process image for inputs (PII) and the process image for outputs (PIQ). During program execution, the CPU accesses these memory areas exclusively. It does not access the digital input and output modules directly.

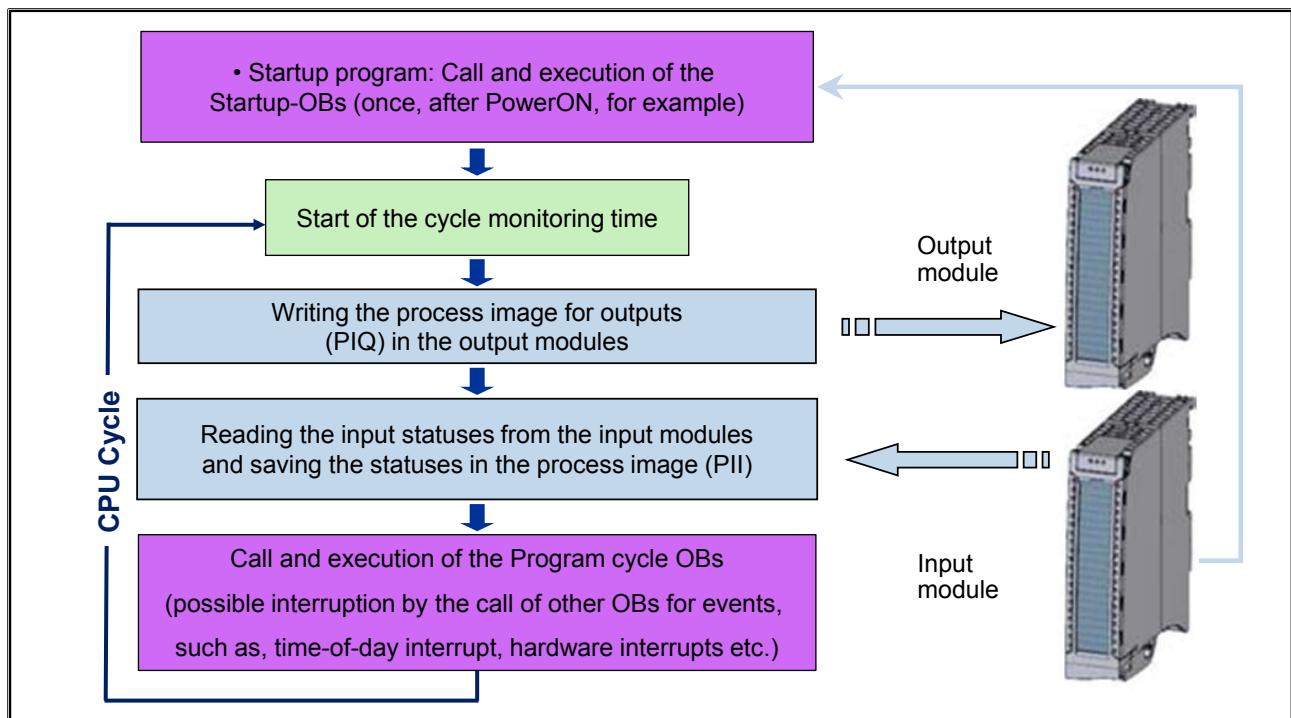
PII

The Process Image for Inputs (PII) is the memory area in which the statuses of all digital inputs are stored. At the beginning of the cycle, the digital input modules read-in to the PII. When an input is linked, the status of this input stored in the PII is linked. This status cannot change within a cycle since the PII is only updated or read-in at the beginning of a cycle. This guarantees that when there are multiple scans of the input in one cycle, the same result is always supplied.

PIQ

The Process Image for Outputs (PIQ) is the memory area in which the statuses of all digital outputs are stored. The PIQ is output to the digital output modules at the beginning of the cycle before the process image for inputs is updated. Outputs can be assigned as well as scanned in the program.

6.4. Cyclic Program Execution



Restart

When you switch on or switch from STOP --> RUN, the CPU carries out a complete restart whereby the programmed Startup-OBs are executed. During restart, the operating system deletes all non-retentive memory bits and, after the Startup-OBs have been executed, it starts the cycle monitoring time. (**The Startup-OBs are dealt with in the chapter "Organization Blocks"**)

Cyclic Program Execution

Cyclic program execution occurs in an endless loop. After the execution of a program cycle is completed, the execution of the next cycle occurs automatically. In every program cycle, the CPU carries out the following steps.

- Transfer the output statuses from the process image for outputs to the output modules.
- Scan the statuses of the input signals and update the process image for inputs.
- Execute the instructions of the user program. This happens mainly with the process images, not with the input and output modules directly.

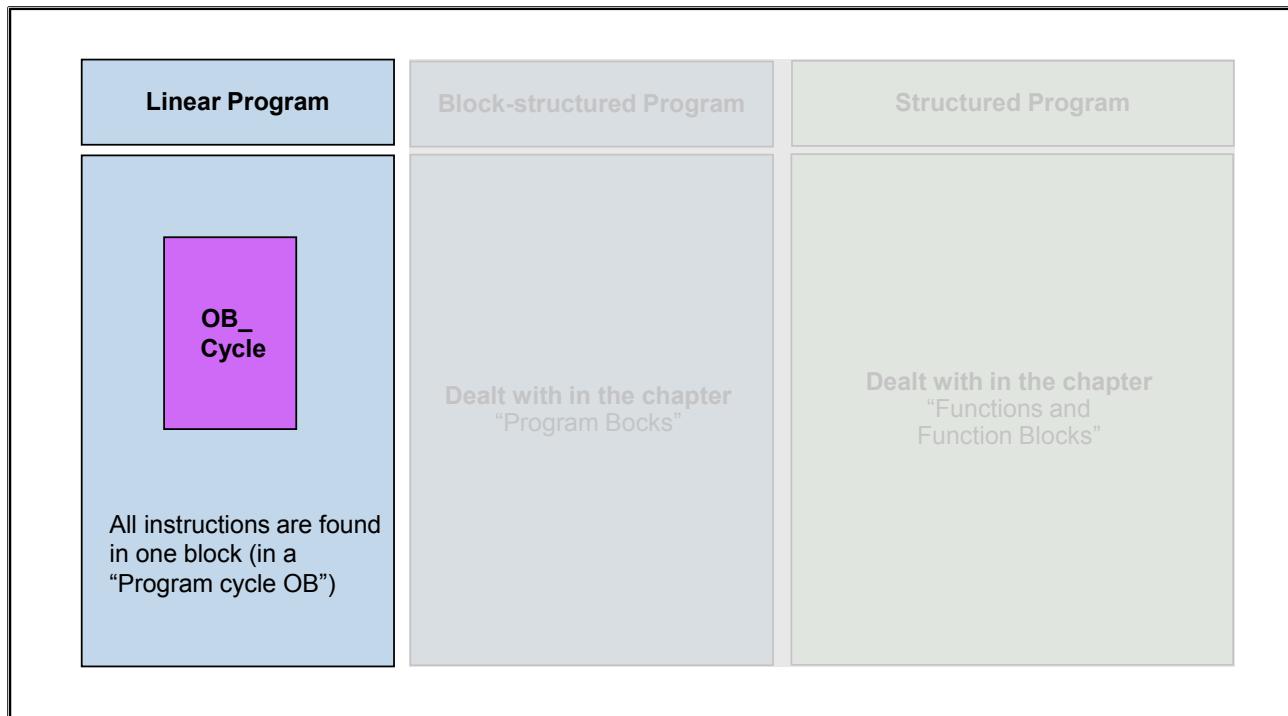
Cycle Time and Cycle Monitoring Time

The time that the CPU requires for the execution of the complete program cycle, is the cycle time which is monitored for time by the CPU operating system. If the cycle time exceeds the cycle monitoring time defined in the CPU properties (**Chapter Devices and Networks**), the "Time-Error-Interrupt-OB" is called. If this is not configured or if the cycle monitoring time is double, the CPU goes into the STOP state.

Using the Process Image or deselecting it for Individual Modules

In the properties of the modules you can select whether the input values are to be automatically adopted in the PII or whether the output values are to be automatically written with values from the PIQ (at the beginning of the cycle), only if a certain OB is executed or never.

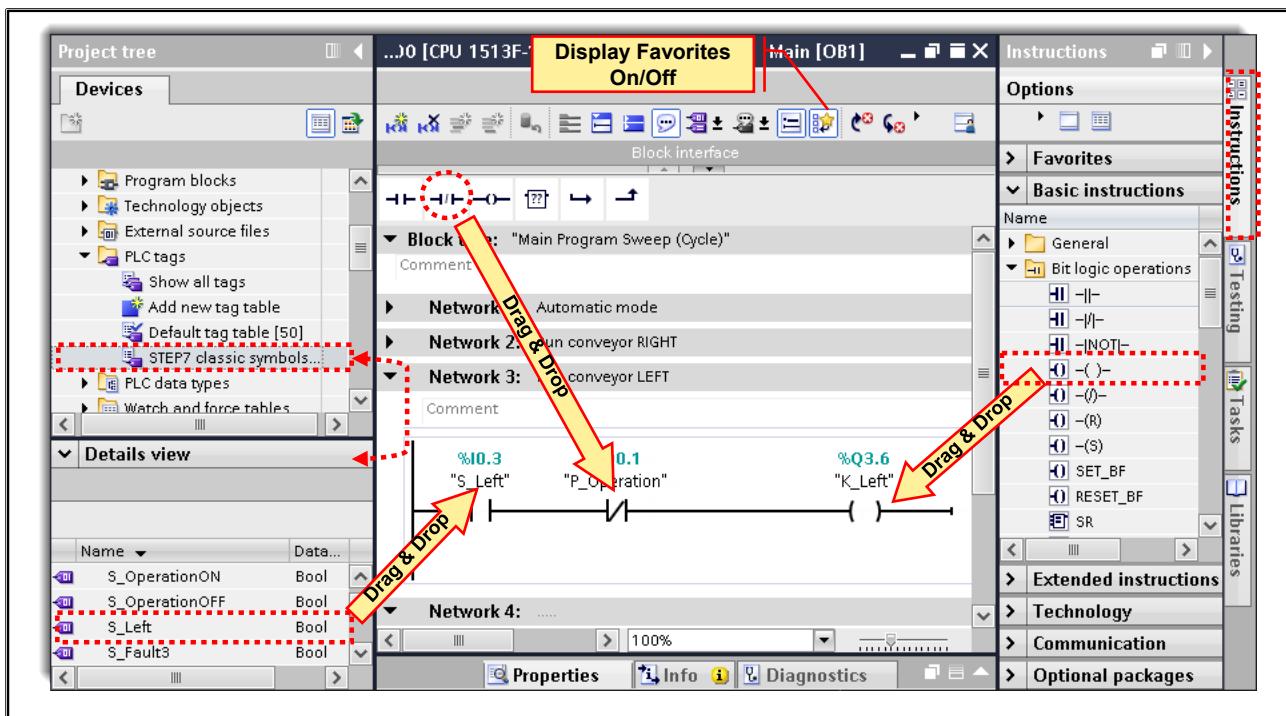
6.5. Linear Program



Linear Program

The entire program is found in one continuous program block (Program cycle OB) which is automatically called by the system. This model resembles a hard-wired relay control that was replaced by an automation system (programmable logic controller). The CPU processes the individual instructions one after the other.

6.6. Programming



Open Block

In the Program blocks folder there is a Program-Cycle-OB which can be opened by double-clicking in the instruction section.

Programming

The instructions within a block can be programmed as follows:

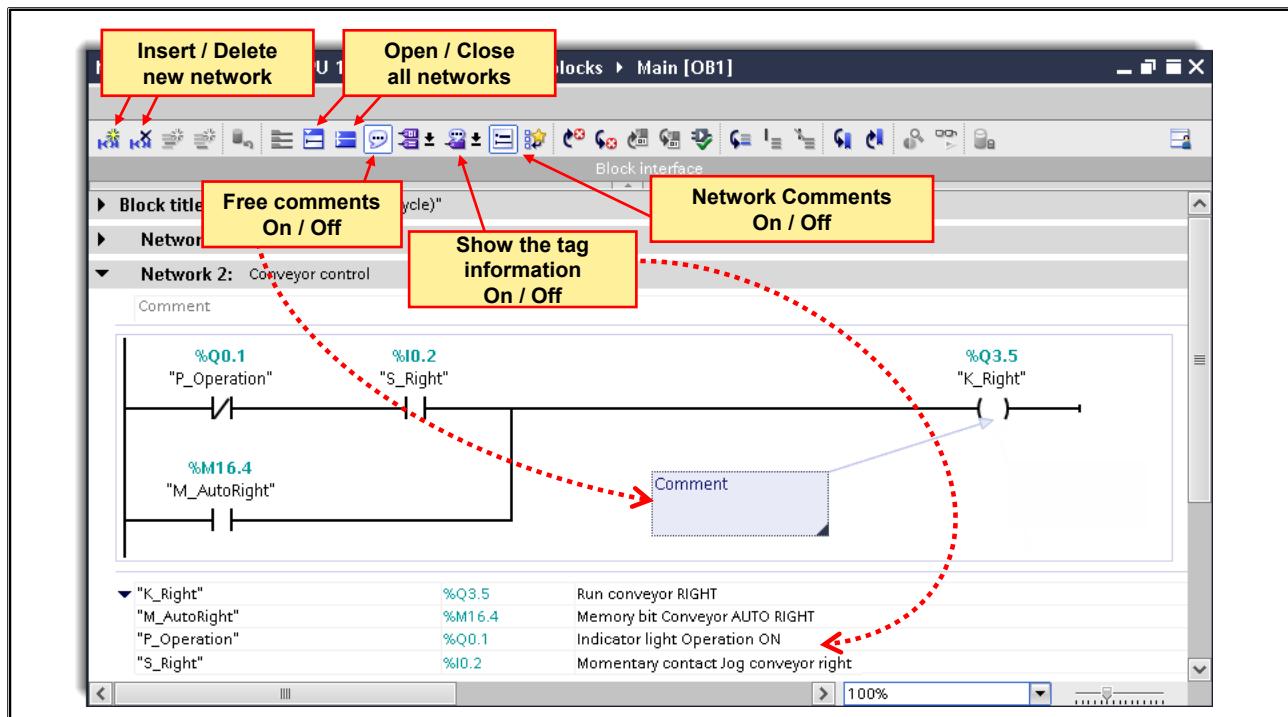
- using drag & drop from the Favorites or the Instructions catalog to anywhere in the program
- by first selecting the location in the program and then double-clicking on the desired instruction in the Favorites or the Instructions catalog
- by selecting (highlighting) the location in the program and the relevant shortcut (for information on the possible shortcuts see Options > Settings > Keyboard shortcuts)

Operands can be entered with an absolute or a symbolic address. If the tag table is highlighted (not opened!) in the Project tree, tags (variables) can also be pulled from the Details view to the appropriate location in the program using drag & drop.

Favorites

Frequently used elements, functions and instructions are available in the Favorites which can be expanded individually from the Instructions catalog using drag & drop.

6.6.1. Networks

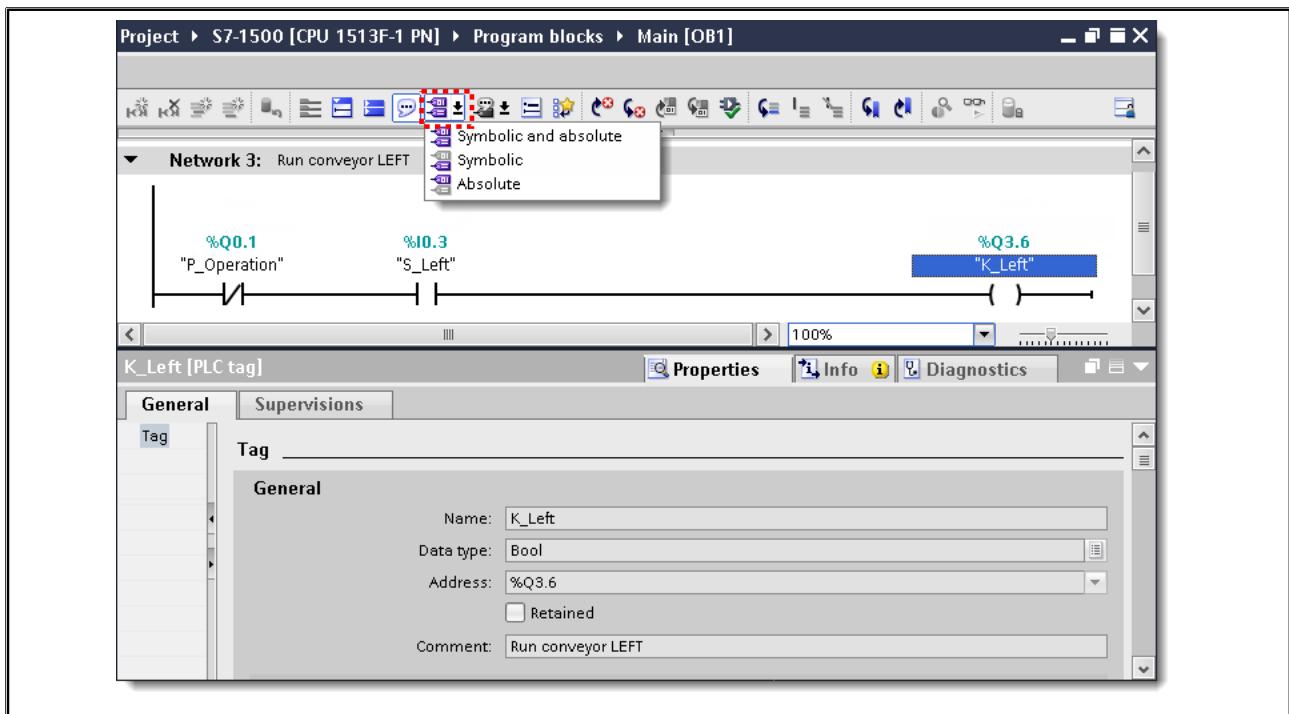


Networks

A block can be divided into individual networks which makes it easier to follow and gives you a better understanding of the program.

Each network can be given a network label and a comment. Within the networks, the individual instructions can be clarified with instruction comments and the comments of the tags/variables used can be displayed via the "Tag information" button.

6.6.2. Absolute and Symbolic Addressing



Absolute and Symbolic Addressing

All global tags/variables (such as, inputs, outputs, memory bits) have both an absolute and a symbolic address. You can define which is to be displayed or with which is to be programmed (see picture).

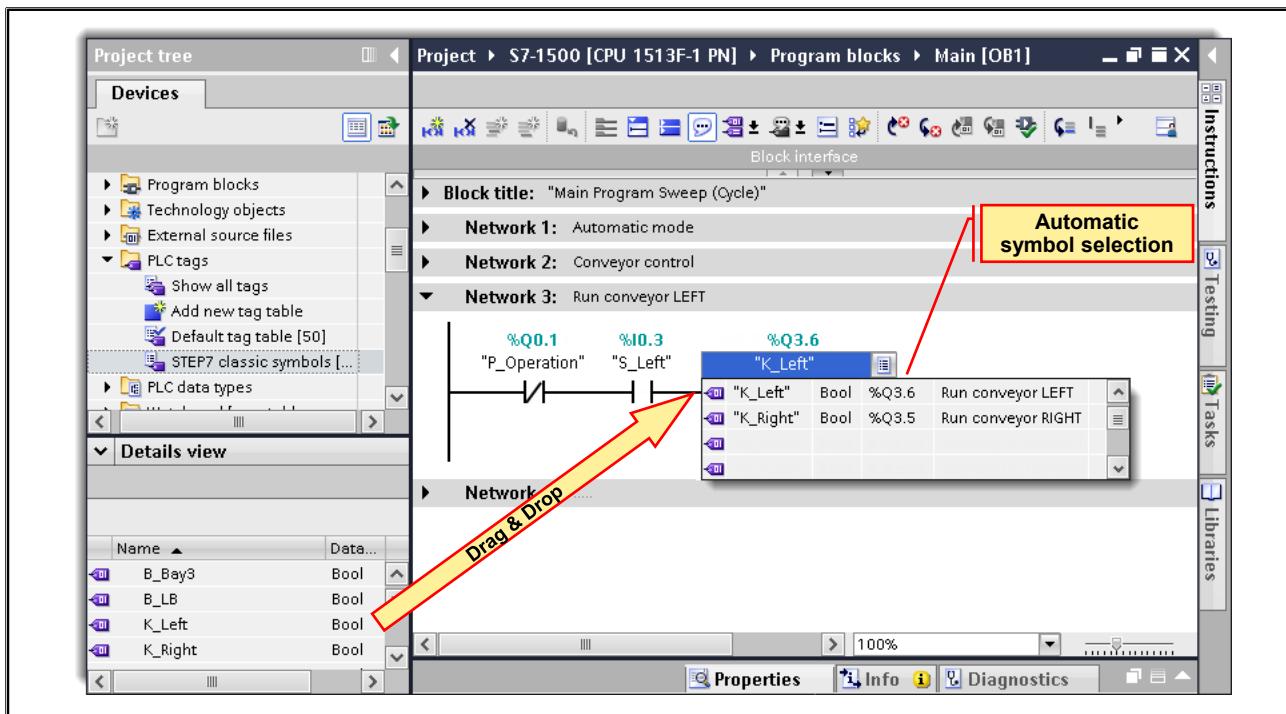
When you use a symbolic address (for example, "K_Left") which has not yet been assigned an absolute address, you can save the block but you cannot compile it and download it into the controller.

When you use an absolute address (for example, M16.2), it is automatically assigned a symbolic default address (for example, "Tag_1") which you can change (later on).

Properties

If a block or the PLC tag table is open in the working area and a tag is selected (highlighted) there, then all details are displayed in the "Properties" tab in the Inspector window.

6.6.3. Using a PLC Tag as an Operand



Using a Tag as Operand

During programming, the name of the tag or the address can be entered. When the symbolic name or the address is input, the Autocompletion automatically appears from which you can select the tag. Furthermore, you can use the Details view to adopt the tag using drag & drop.

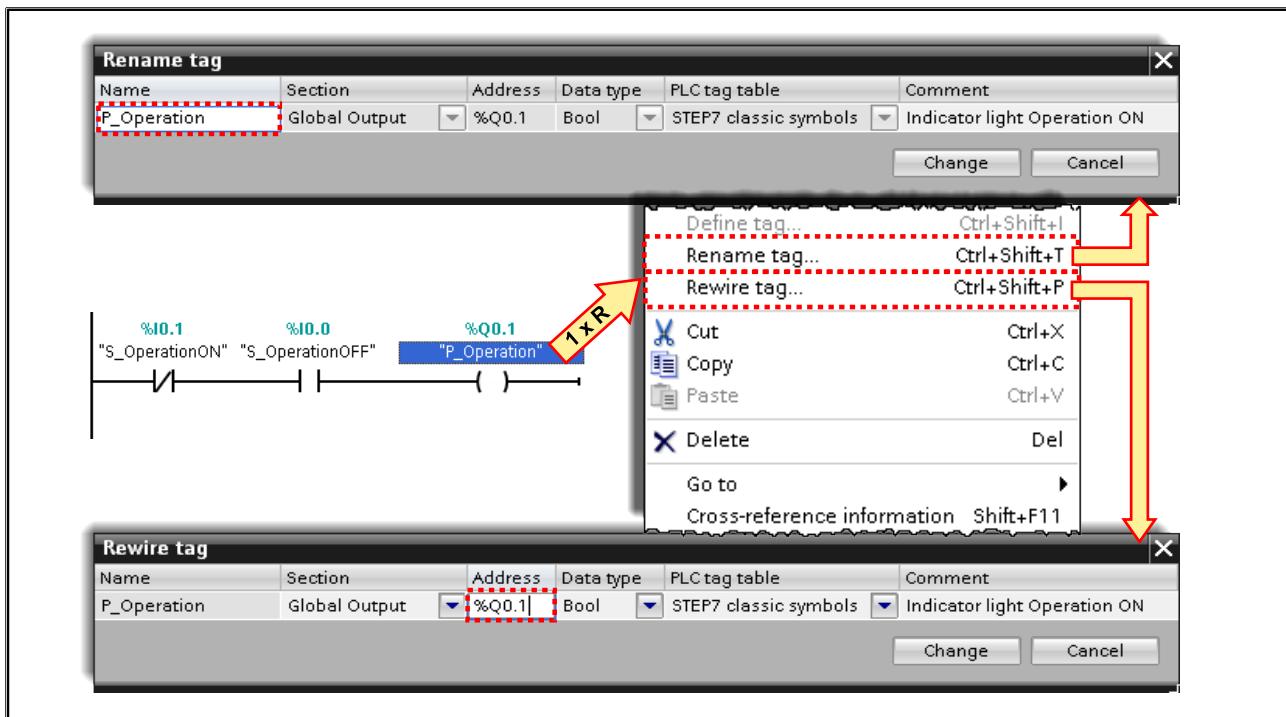
Autocompletion

When operands are selected, after the first letter of the symbolic operand name has been entered, a selection of all the operands whose name start with the entered letter and are of the corresponding data type is displayed. All the operands that are valid for this block are displayed. These are all global tags (also those that are declared in data blocks), local variables (temporary and static) as well as the parameters of the block.

You can also filter directly according to the type of tag, as required:

- Begin with # to only select from local tags of the current block interface,
- Begin with " to only have global tags displayed,
- Begin with % to have all tags filtered according to absolute addresses displayed,

6.6.4. Renaming / Rewiring PLC Tags



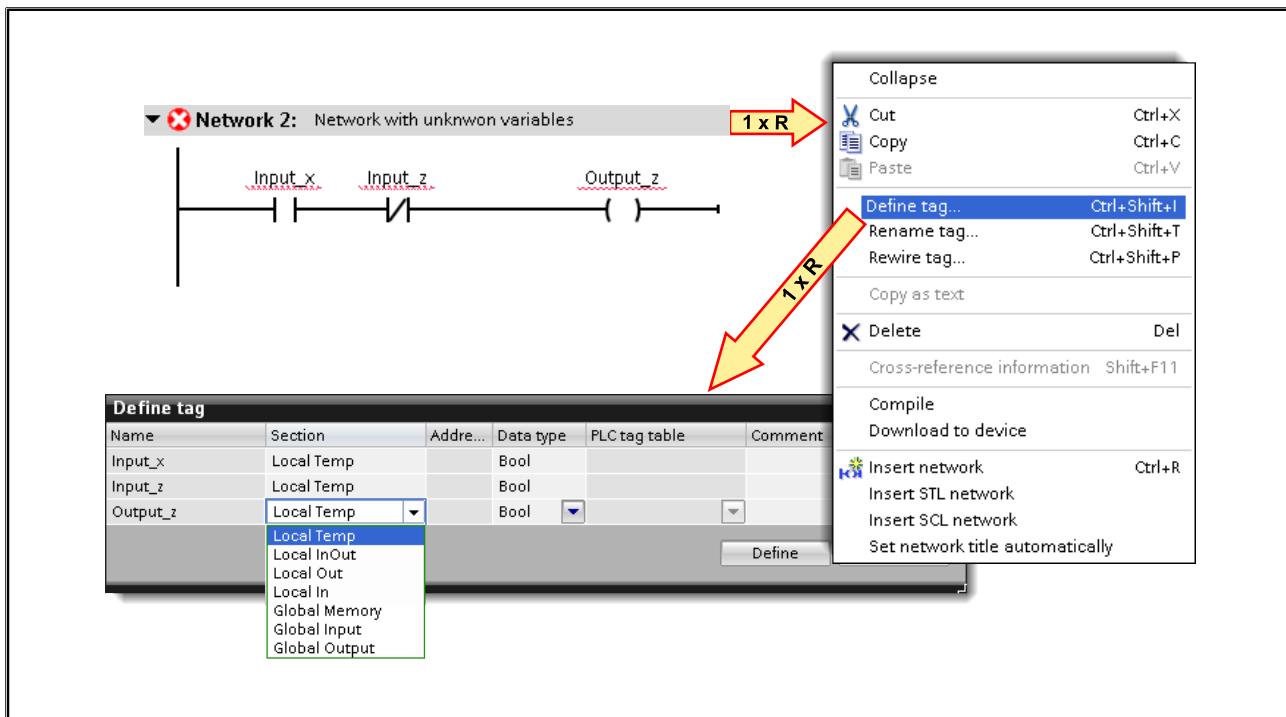
Renaming and Rewiring Tags

Tags can be renamed or rewired as shown in the picture using the Blocks Editor. The changes are immediately adopted in the PLC tag table and affect the entire program.

Tags can also be renamed and rewired directly in the PLC tag table.

- **Rename:**
Change the tag name, while the absolute address remains unchanged.
- **Rewire:**
Change the associated absolute address, while the name remains unchanged.

6.6.5. Defining (Declaring) Tags while Programming



Defining (Declaring) Tags while Programming

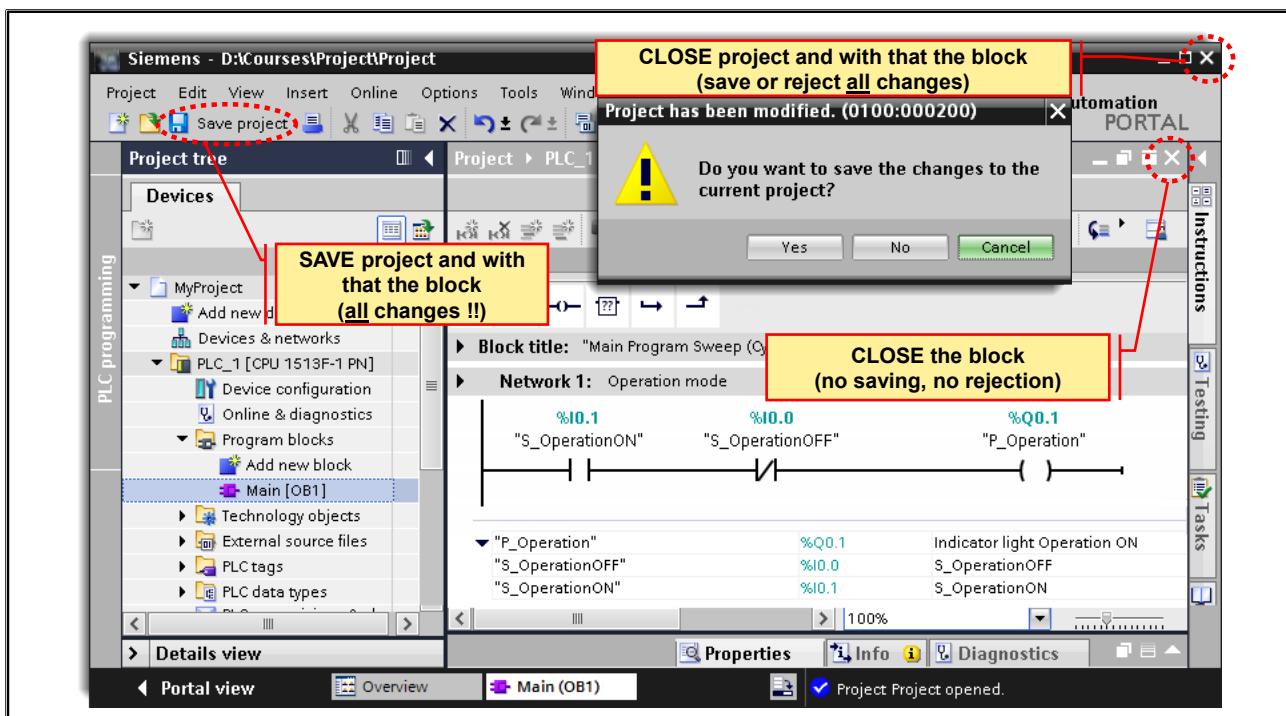
If unknown tags are used during programming, they can be defined later-on via the context menu of the network.

In the "Define tag" dialog, the [Local Temp] temporary memory area (Section) is always suggested. When you change the area, the next free address is suggested.

Advantage:

In the dialog that appears, only addresses which have not been used so far are suggested. In this way errors are avoided, for example, such as the use of bits which belong to an already used word (overlapping accesses).

6.6.6. Closing / Saving / Rejecting a Block



Closing a Block

By clicking on the symbol in the title bar, the block is merely closed. Changes are neither rejected nor are they saved on the hard drive!

Saving a Block

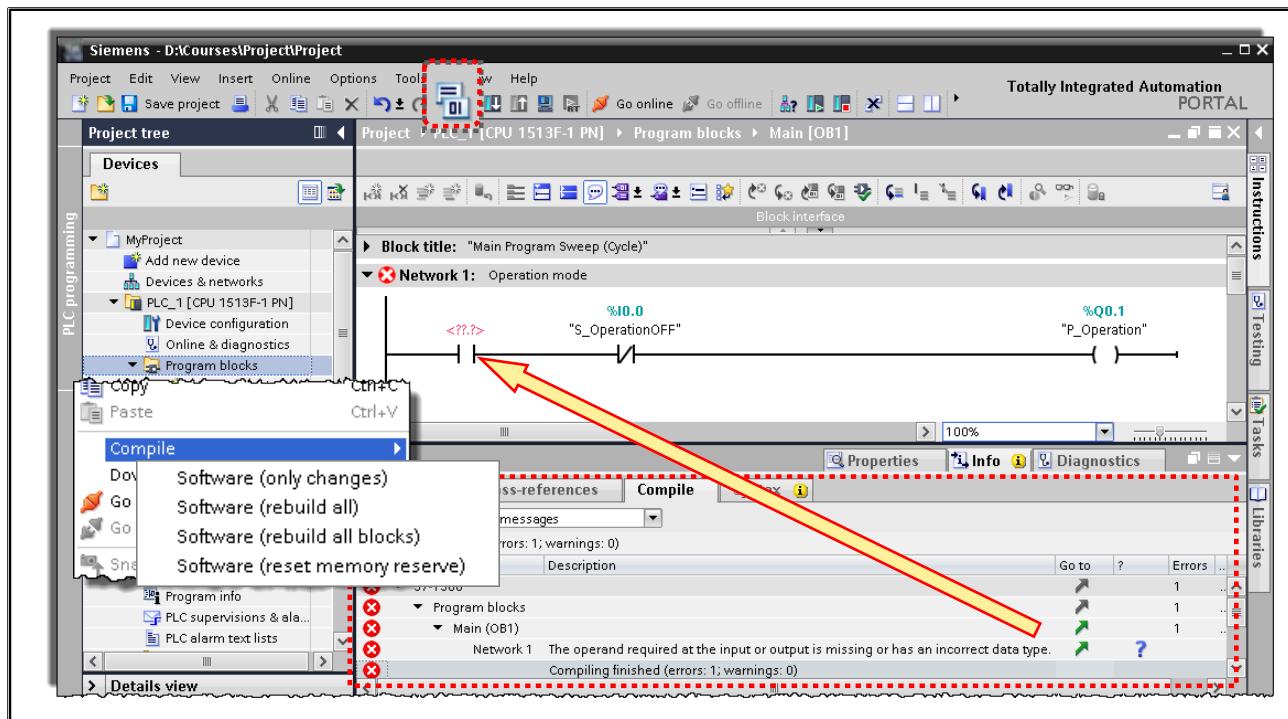


By using "Save project" the entire project, and with that also the block, is saved on the hard drive. All changes made to the project are saved.

Rejecting a Block

It is possible to reject block changes using the function Undo or by closing the entire project without saving. All changes made in the project are rejected.

6.6.7. Compiling a Program



Program Compilation

A delta compilation is always carried out via the "Compile" button, that is, only the changes to the object selected (highlighted) in the Project tree are compiled. By selecting the Program blocks folder, all modified objects (blocks) within the folder are compiled.

Via the context menu in the Project tree, you can select whether only changes or the entire software is to be compiled.

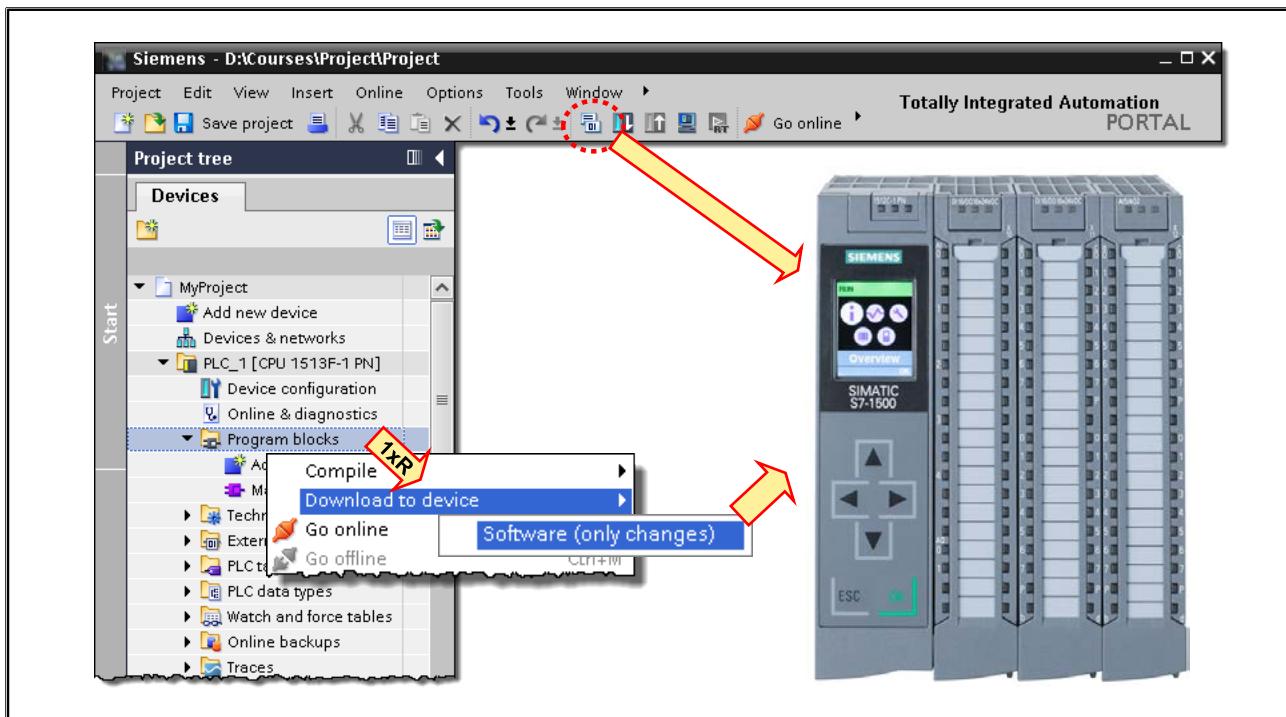
The status of the compilation is displayed in the Inspector window "Info -> Compile". If errors occurred during compilation, you can jump directly from the error entry to the error location by double-clicking on it.

Note:

The menu items Software (rebuild all) and Software (rebuild all blocks) currently have the same meaning.

Software (reset memory reserve) is not yet relevant in this chapter.

6.6.8. Downloading a Program into the CPU



Downloading a Program:

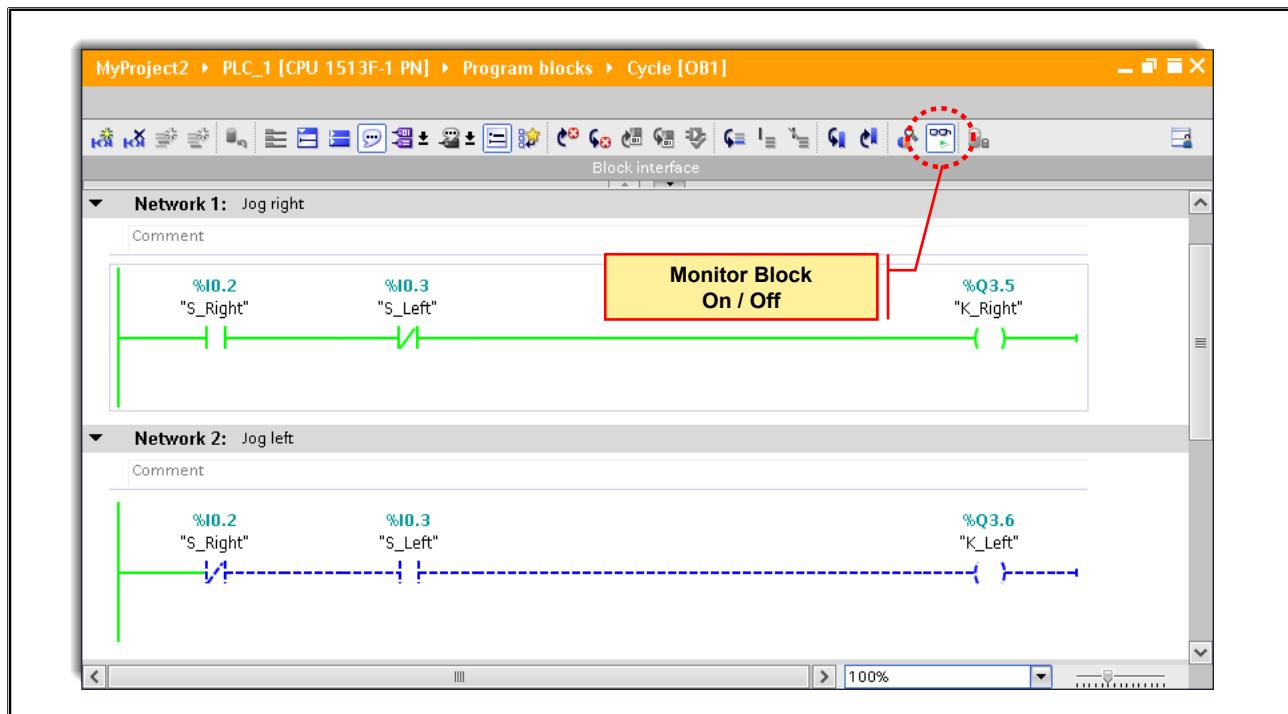
Program (Software) includes the blocks of the user program. The first time you download, they are completely loaded. In subsequent downloads, only the software changes are loaded.

However, all changes are always downloaded, that is, the offline program and the online program are always the same after the download.

Note:

In the context menu of the device (CPU) there is also the menu item "Software (all)" in which all blocks are loaded even if these have already been downloaded to the CPU. You have to make sure, however, that the CPU is stopped in this case.

6.6.9. Monitoring a Block



Monitor

The test function *Monitor* is used to track the program execution within a block. The statuses or contents of the operands used in the block at the time of program execution are displayed on the monitor.

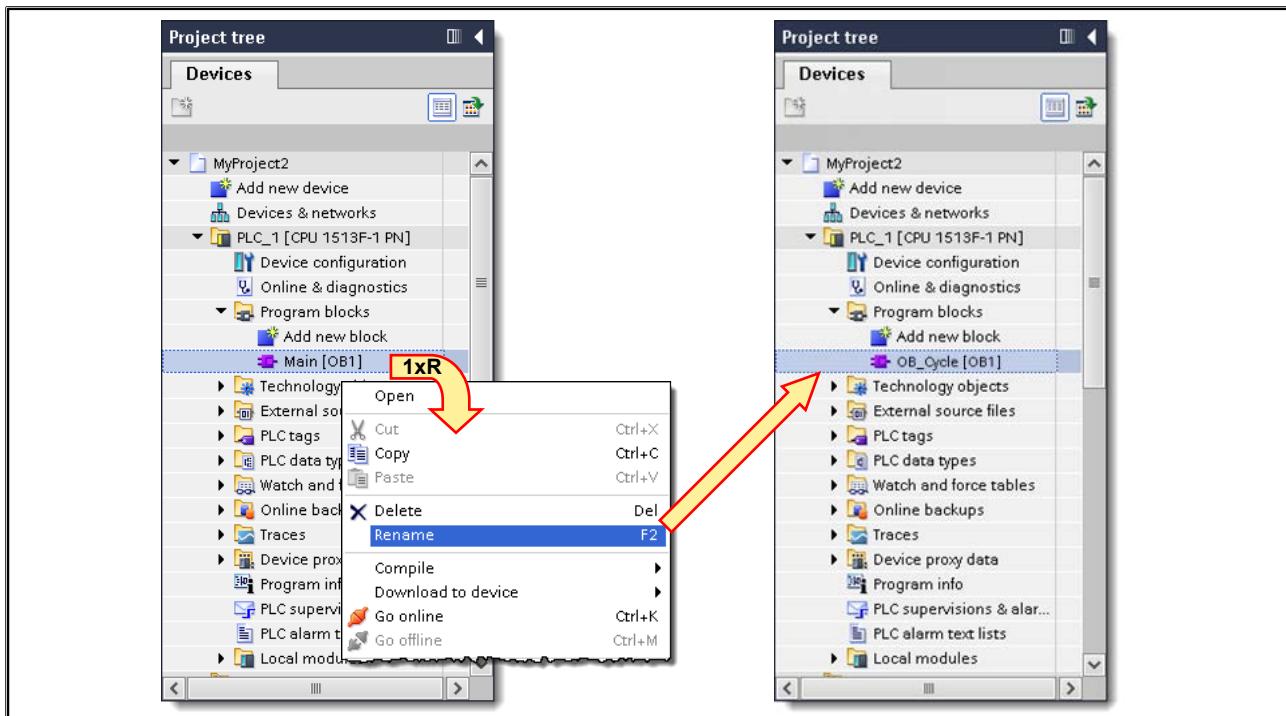
You can only monitor if there is an online connection to the CPU and the offline program is identical to the online program.

In test mode, the statuses of the operands and LAD / FBD elements are represented by different colors.

Examples:

- Status fulfilled → "Element is represented with a green color"
- Status not fulfilled → "Element is represented with a blue color"

6.7. Exercise 1: Renaming Main OB



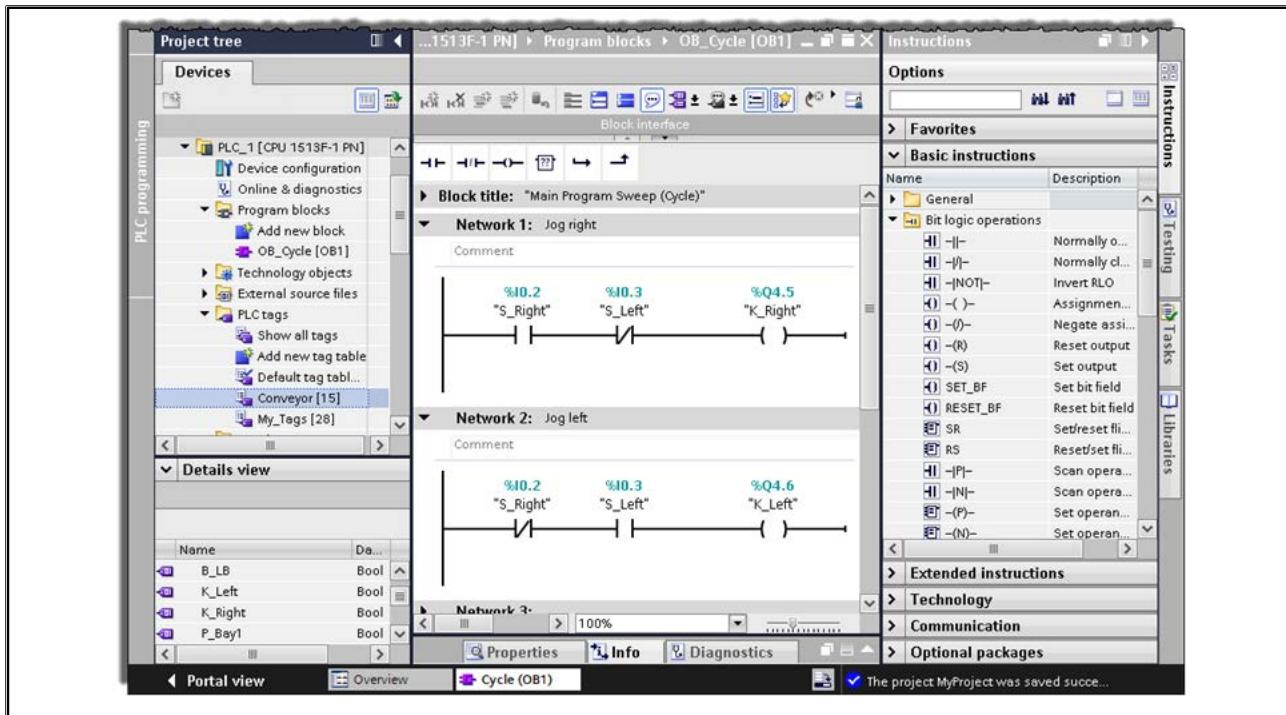
Task

You are to rename the Organization block.

What to Do

1. Open the "Program blocks" folder.
2. Open the context menu of the block "Main" by right-clicking on it.
3. Select the menu item "Rename".
4. Give the block the symbolic name "OB_Cycle".
5. Save your project.

6.7.1. Exercise 2: Programming the Jog Mode



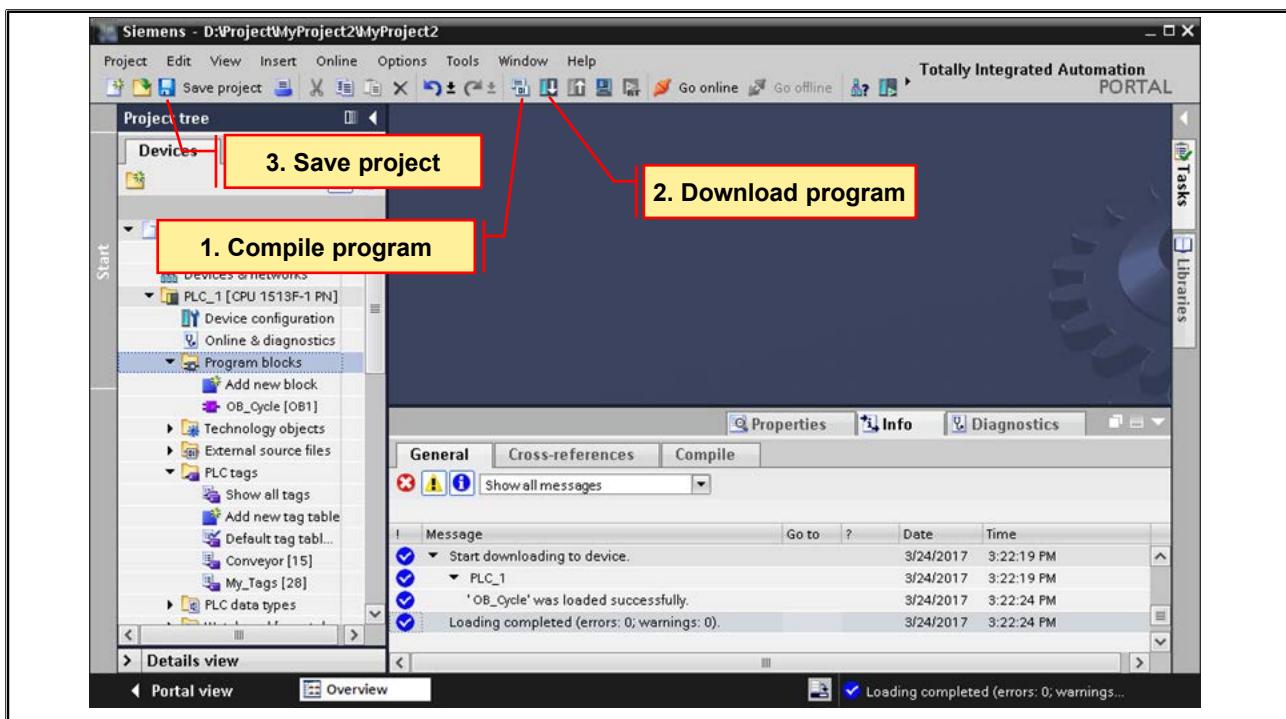
Task

You are to program the Jog mode of the conveyor as shown in the picture.

What to Do

1. Program an AND logic operation by dragging a "Normally open contact" and a "Normally closed contact" from the Favorites into the network using drag & drop.
2. Program an "Assignment" by dragging it from the "Instructions" Task Card to the AND logic operation using drag & drop as shown above.
3. At the first input of the AND logic operation enter the input "S_Right" (I 0.2) as operand (you can enter the symbol as well as the absolute address). Do the same for "S_Left" (I 0.3).
4. In the Project tree, select (do not open!) the tag table "Conveyor" and drag the tag "K_Right" (Q3.5) from the "Details view" as operand above the assignment.
5. Give the network a title.
6. Add a new network and there program an appropriate logic operation for jogging the conveyor to the left.
7. Close the Organization block.
8. Save your project.

6.7.2. Exercise 3: Compiling the Program and Downloading it into the CPU



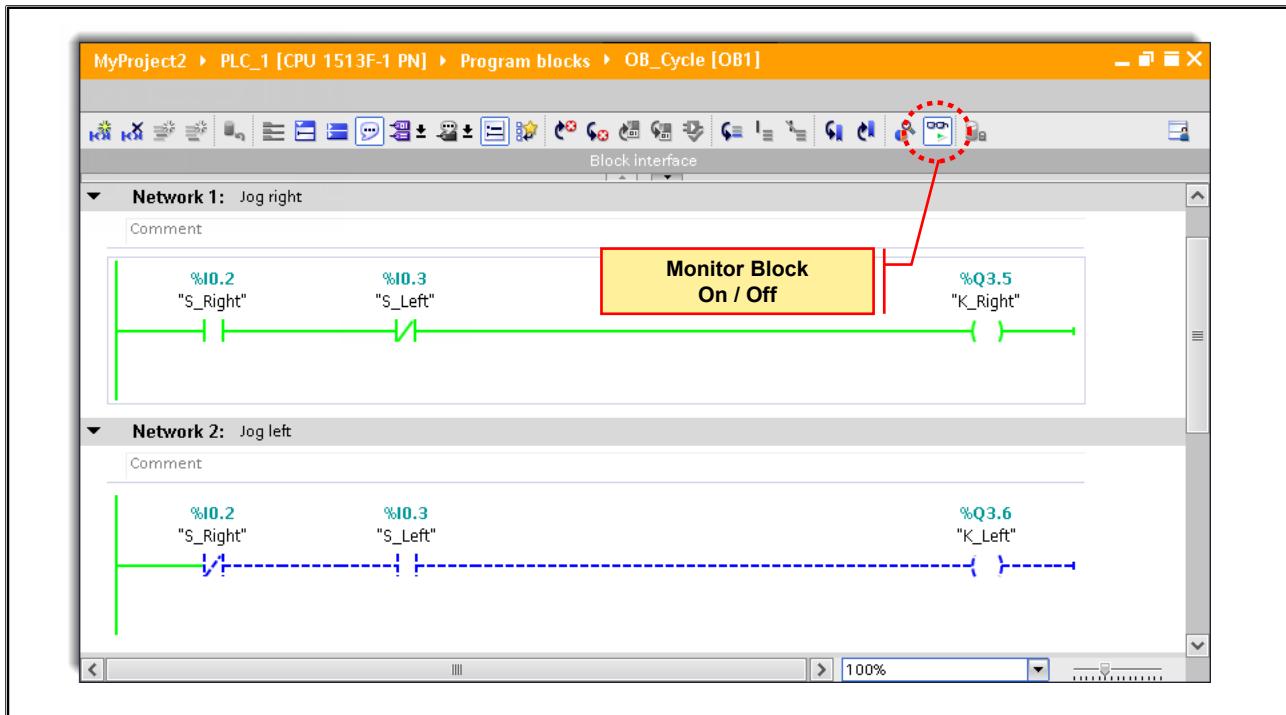
Task

You are to compile and download the program and save the project.

What to Do

1. Select the "Program blocks" folder.
2. Compile your program by means of the "Compile" button in the toolbar.
3. Download your program by means of the "Download" button in the toolbar.
4. Save your project.

6.7.3. Exercise 4: Monitoring the Program



Task

You are to monitor the program online.

What to Do

1. Open "OB_Cycle".
2. Monitor the block (the program) by means of the "Monitor On/Off" button.
3. Activate the switches "S_Left" and "S_Right" on your training case.

Contents

7

7.	Program Blocks and Program Editor	7-2
7.1.	Plant Description: The Conveyor Model as Distribution Conveyor with Mode Selection	7-3
7.2.	Overview of the Blocks in STEP 7	7-4
7.2.1.	Block-structured Programming	7-5
7.2.2.	Program Sequence	7-6
7.3.	Adding a New Block.....	7-7
7.4.	Block Properties: General, Time Stamps.....	7-8
7.4.1.	Block Properties: Know-how Protection and Copy Protection.....	7-9
7.5.	Block Editor Settings.....	7-10
7.6.	Programming Block Calls.....	7-11
7.7.	Deleting Blocks	7-12
7.8.	"Upload" Blocks "from Device" (Upload into Project).....	7-13
7.9.	Task Description: Programming the Mode Selection and Manual Mode	7-14
7.9.1.	Exercise 1: Adding the "FC_Mode" Block.....	7-15
7.9.2.	Exercise 2: Programming the Mode Selection in "FC_Mode"	7-16
7.9.3.	Exercise 3: Adding the "FC_Conveyor" Block	7-17
7.9.4.	Exercise 4: Shifting the Networks from "OB_Cycle" to "FC_Conveyor" and Expanding it.	7-18
7.9.5.	Exercise 5: Checking the "OB_Cycle" Properties.....	7-19
7.9.6.	Exercise 6: Calling "FC_Mode" and "FC_Conveyor" in "OB_Cycle"	7-20
7.9.7.	Exercise 7: Compiling, Downloading and Saving the Program	7-21
7.10.	Additional Information	7-22
7.10.1.	Block Groups.....	7-23
7.10.2.	S7-1500 - Memory Concept.....	7-24

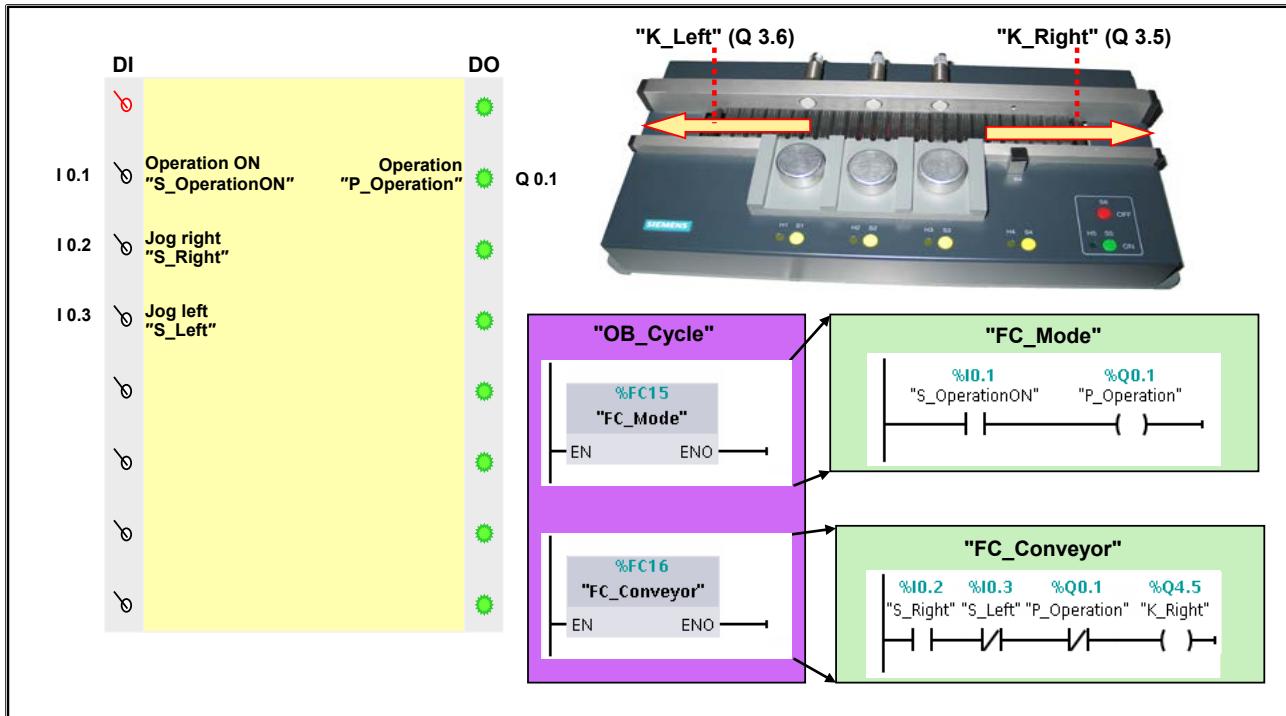
7. Program Blocks and Program Editor

At the end of the chapter the participant will ...

- ... be familiar with the different S7 block types
- ... be familiar with the principle of "structured programming"
- ... be able to add/create new blocks (functions)
- ... be able to set block properties
- ... be able to program a block call
- ... know what happens when a block is uploaded
- ... be familiar with the memory concept of the S7-1500



7.1. Plant Description: The Conveyor Model as Distribution Conveyor with Mode Selection



Conveyor Model as a Distribution Conveyor

The Plant is switched ON/OFF and in Automatic or Manual mode using the switch "S_OperationON".

In Manual mode ("P_Operation" = FALSE), the distribution conveyor can be moved to the right and left using the switches "S_Right" and "S_Left". If both switches are activated simultaneously, then the conveyor must not move (Lock-out!).

7.2. Overview of the Blocks in STEP 7

Blocks/Instructions	Properties
Organization block (OB)	<ul style="list-style-type: none"> - User interface - Graduated priorities (0 to 27) - Specific start information in temporary variables
Function (FC)	<ul style="list-style-type: none"> - Parameter-assignable (parameters must be assigned with the call) - Without (dedicated) memory (only temporary variables)
Function block (FB)	<ul style="list-style-type: none"> - Parameter-assignable (parameters can be assigned with the call) - With (dedicated) memory (static variables)
Data block (DB)	<ul style="list-style-type: none"> - Structured local data storage (Instance DB) (memory of an FB) - Structured global data storage (Global DB) (valid in the entire program)
System instruction without instance	<ul style="list-style-type: none"> - Instruction without memory stored in the CPU's operating system and callable by the user
System instruction with instance	<ul style="list-style-type: none"> - Instruction without memory stored in the CPU's operating system and callable by the user (therefore requires an instance)

Code Blocks

The automation system provides various types of blocks in which the user program and the related data can be stored. Depending on the requirements of the process, the program can be structured in different blocks. You can use the entire operation set in all blocks (FB, FC and OB).

Organization Blocks (OBs)

Organization blocks (OBs) form the interface between the operating system and the user program.

Functions (FCs)

A function (FC) contains a partial functionality of the program. It is possible to program functions as parameter-assignable so that when the function is called it can be assigned parameters. As a result, functions are also suited for programming frequently recurring, complex partial functionalities such as calculations.

Function Blocks (FBs)

Basically, function blocks offer the same possibilities as functions. In addition, function blocks have their own memory area in the form of instance data blocks. As a result, function blocks are suited for programming frequently recurring, complex functionalities in which data has to be stored over several cycles (such as closed-loop control tasks).

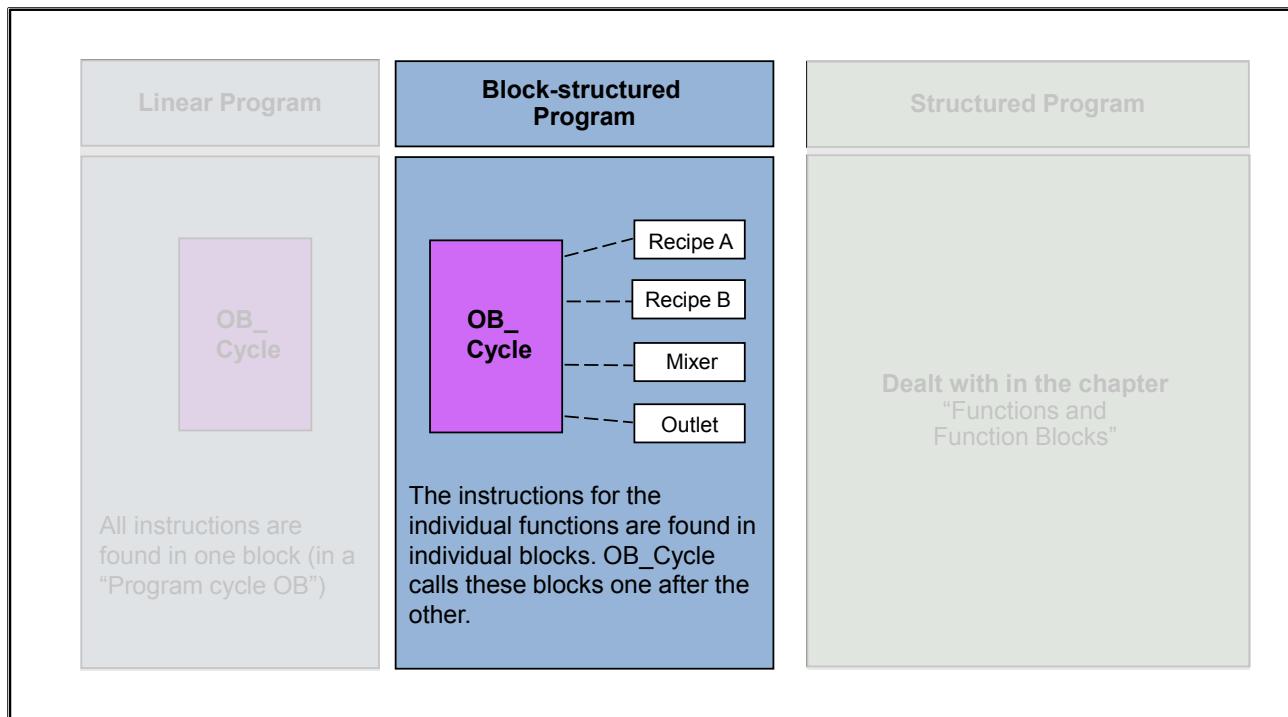
Data Blocks:

Data blocks are used to store data. There are global data blocks and instance data blocks which are used for the data backup of an FB call.

System Instructions:

System instructions are used to solve frequently required standard tasks. They are integrated in the CPU's operating system and therefore do not require any additional memory.

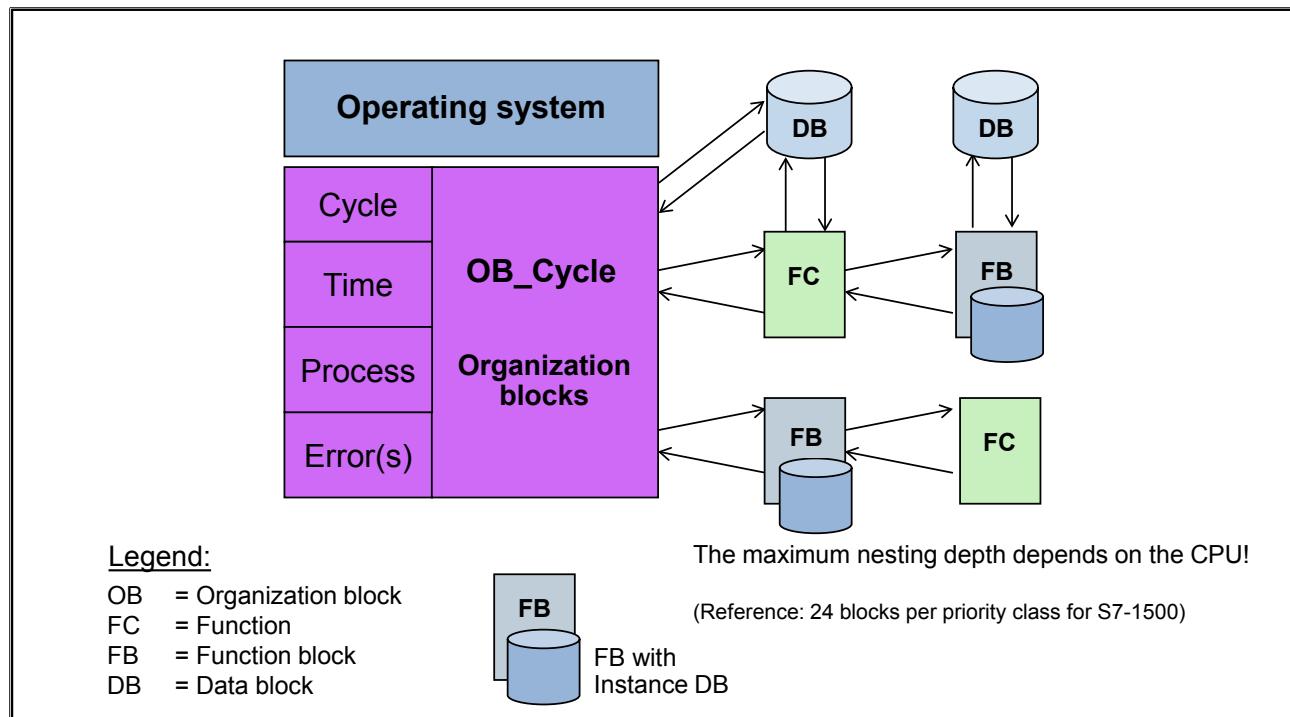
7.2.1. Block-structured Programming



Block-structured Program

The program is divided into blocks, whereby every block only contains the program for solving a partial task. Further structuring through networks is possible within a block. You can generate network templates for networks of the same type. Normally, a cyclically called Organization block contains instructions which call the other blocks in a defined sequence.

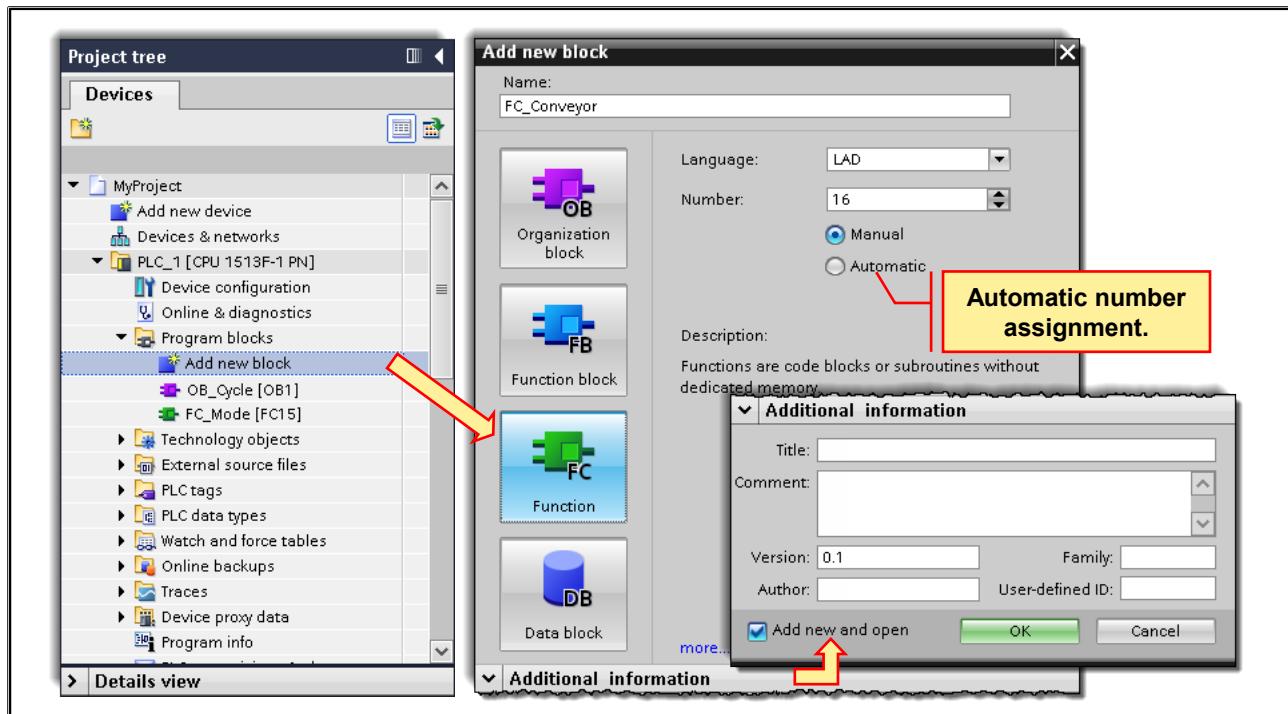
7.2.2. Program Sequence



Organization blocks are automatically called by the CPU's operating system according to certain call events. The Program cycle OBs are responsible for the cyclic program execution. So that blocks, such as FBs and FCs, can be processed, they have to be called by an OB. These blocks, in turn, can call further FBs or FCs.

Data blocks are used for data backup; for this there are two types. Firstly, global data blocks whose structure can be freely defined by the user. And second, the data blocks which are used to store the data of individual function blocks (instance data blocks). The structure of these data blocks depends on the interface of the respective function block. (see chapter "Functions and Function Blocks")

7.3. Adding a New Block

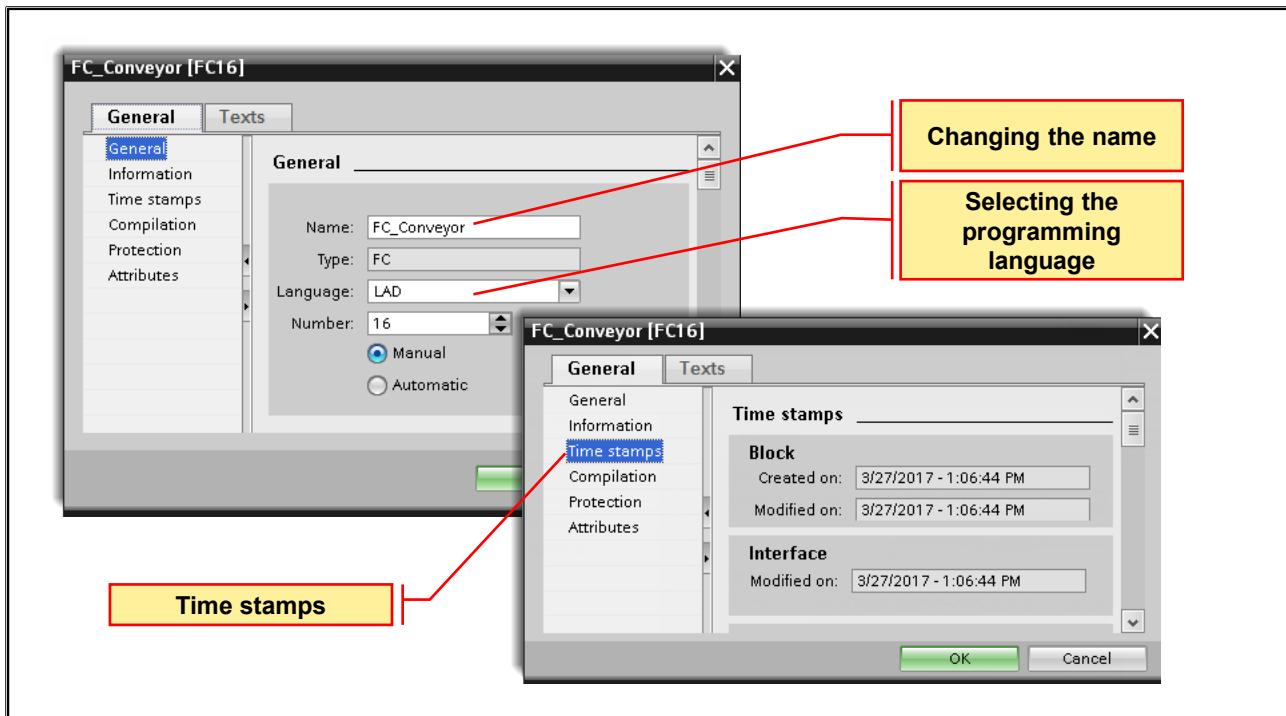


Inserting a Block

A new block is created as shown in the picture. When you create a block, the type of block (OB, FB, FC or DB), the programming language, the symbolic name and number, among other things, must be defined. The block numbers can also be assigned automatically or manually.

Under "Additional information", the block can be documented in more detail, among other things, with a Version number and an Author.

7.4. Block Properties: General, Time Stamps

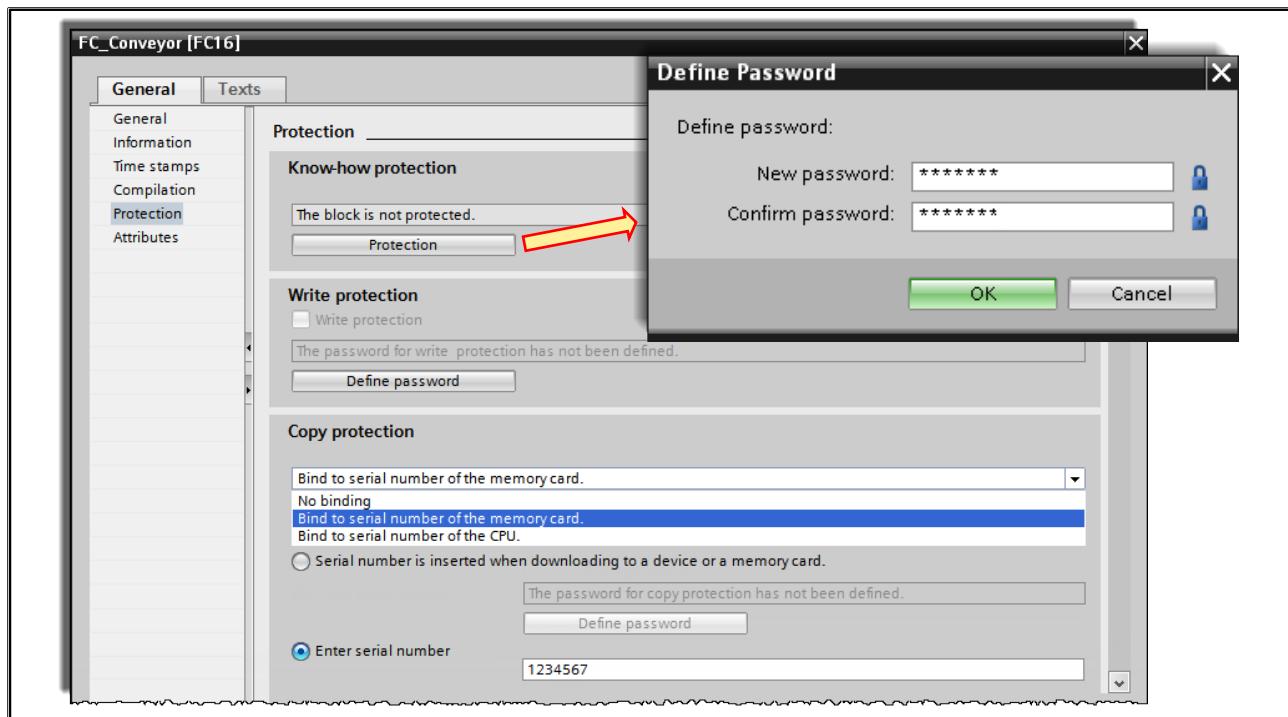


Properties General

Each block has certain properties that you can display and edit. These properties are used to:

- Identify the block
- Display the memory requirements and the compilation status of the block
- Display the time stamp
- Display the reference information
- Specify the access protection
- Display and change the programming language

7.4.1. Block Properties: Know-how Protection and Copy Protection



Know-how Protection

Blocks can be protected from unauthorized access with a password. With a know-how protected block, only the following data can be read:

- Parameters (Input, Output, InOut, Return)
- Block title
- Block comment
- Block properties
- Cross references with the information about which global tags are used

Just like unprotected blocks, know-how protected blocks can also be copied, called, downloaded into the CPU and deleted. The code of the block, however, is protected from unauthorized reading and changing.

Write protection

You can assign write protection for code blocks, which prevent unintentional changes to the block.

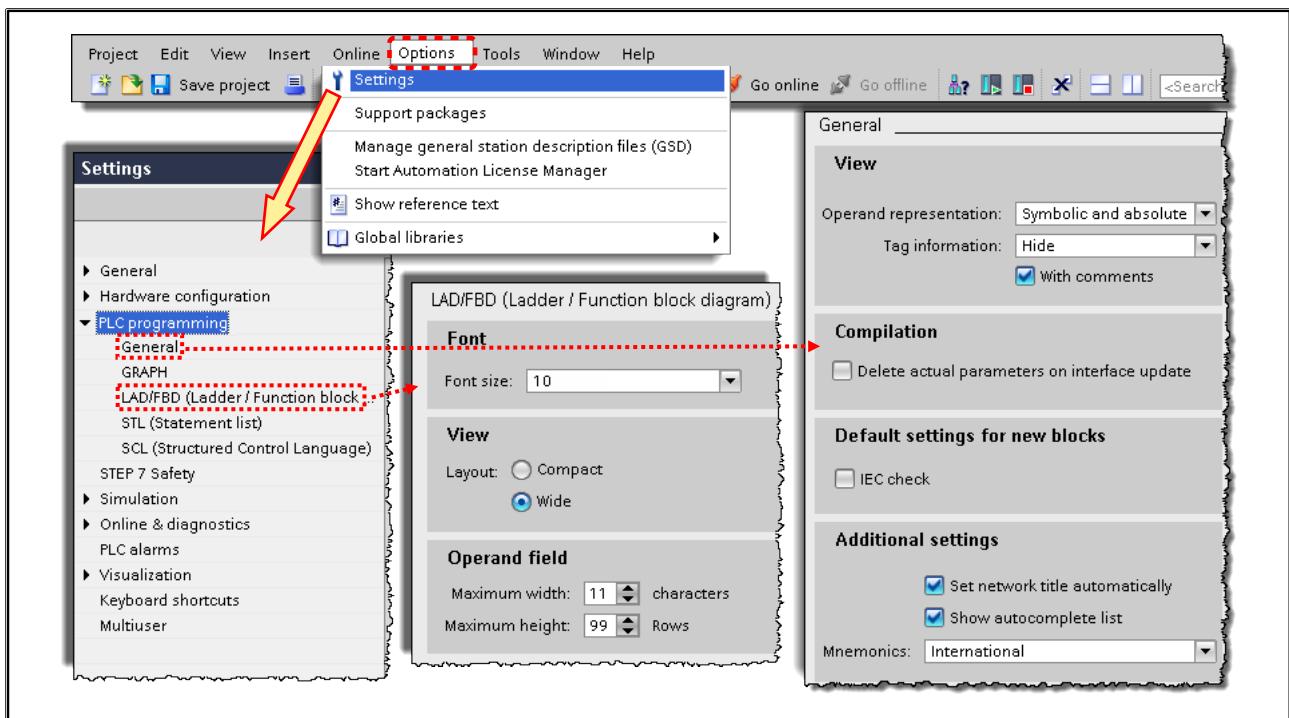
Copy Protection

In addition to know-how protection and write protection, you can also generate a copy protection in which you define with which memory card or on which CPU (each identified through a serial number) the block can be executed.

Caution!

If you forget the password, it is no longer possible to access the block.

7.5. Block Editor Settings

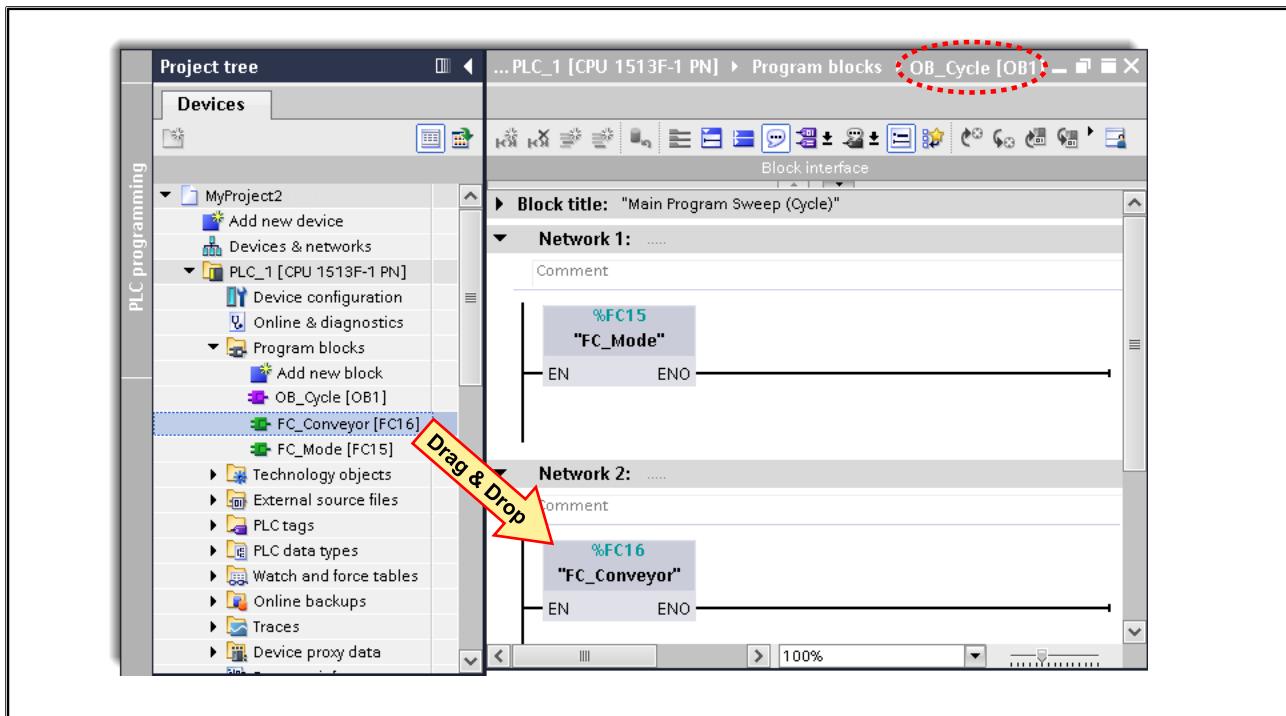


Block Editor Settings

With the settings, you merely define how a block is to be represented when it is opened. In the editor, you can make changes to the representation (such as showing and hiding comments) at any time.

- **View**
Setting the Operand representation and Tag information with and without comments when the block is opened
- **Compilation**
When "Delete actual parameters on interface update" is activated, the calls of parameterized blocks are also then automatically adjusted if, within the block, parameters which are already supplied with an actual parameter during the call, are deleted after the fact.
- **IEC Check**
When IEC check is activated, the compatibility of operands is checked in accordance with IEC 61131.
- **Mnemonics**
Setting the syntax for the programming language: German (e.g. E for Eingang (Input)) or International (e.g. I for Input)
- **View - Layout**
This changes the vertical spacing between operands and other objects (for example, operands and contact). The change only becomes visible the next time a block is opened.
- **Operand Field**
Setting the maximum width and height of function block diagram and ladder diagram symbols

7.6. Programming Block Calls

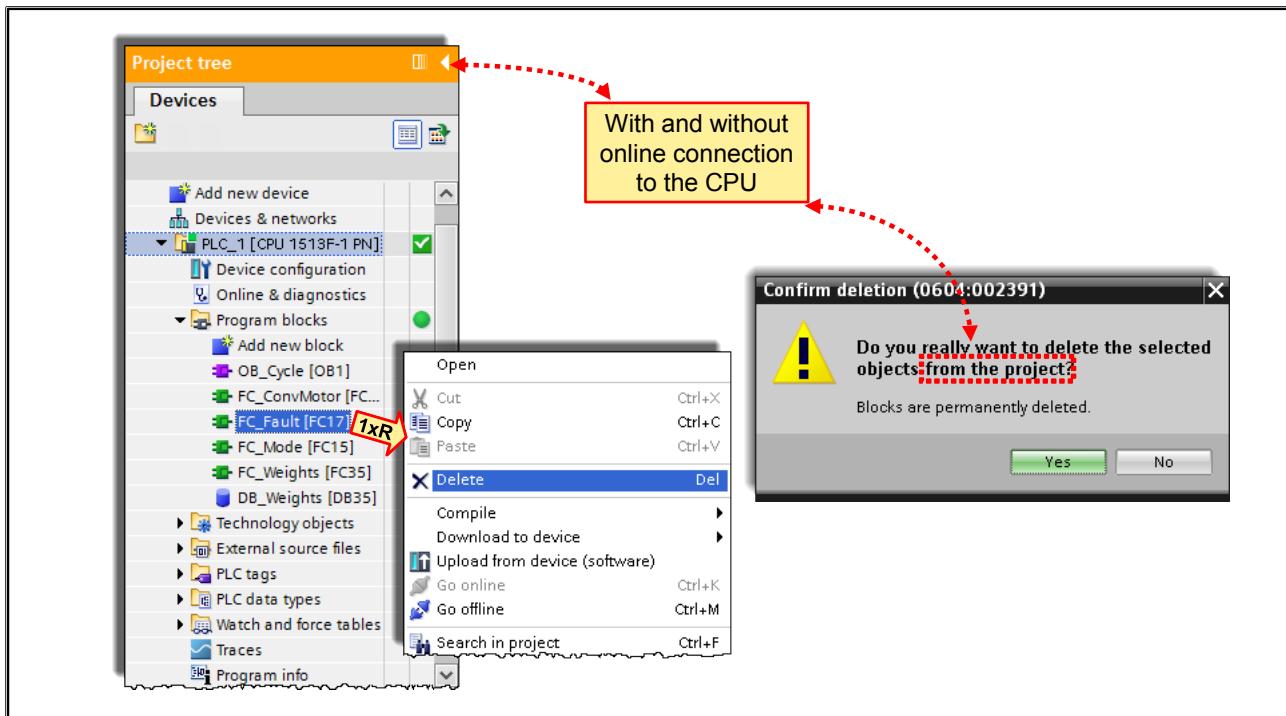


In order to program the call of a block, you simply have to drag the block into the program code of the calling block using drag & drop or copy it using copy & paste.

Block Calls

If a block calls another block, the instructions of the called block are executed. Only when the execution of the called block is completed, is the execution of the calling block taken up again and execution continues with the instruction that follows the block call.

7.7. Deleting Blocks

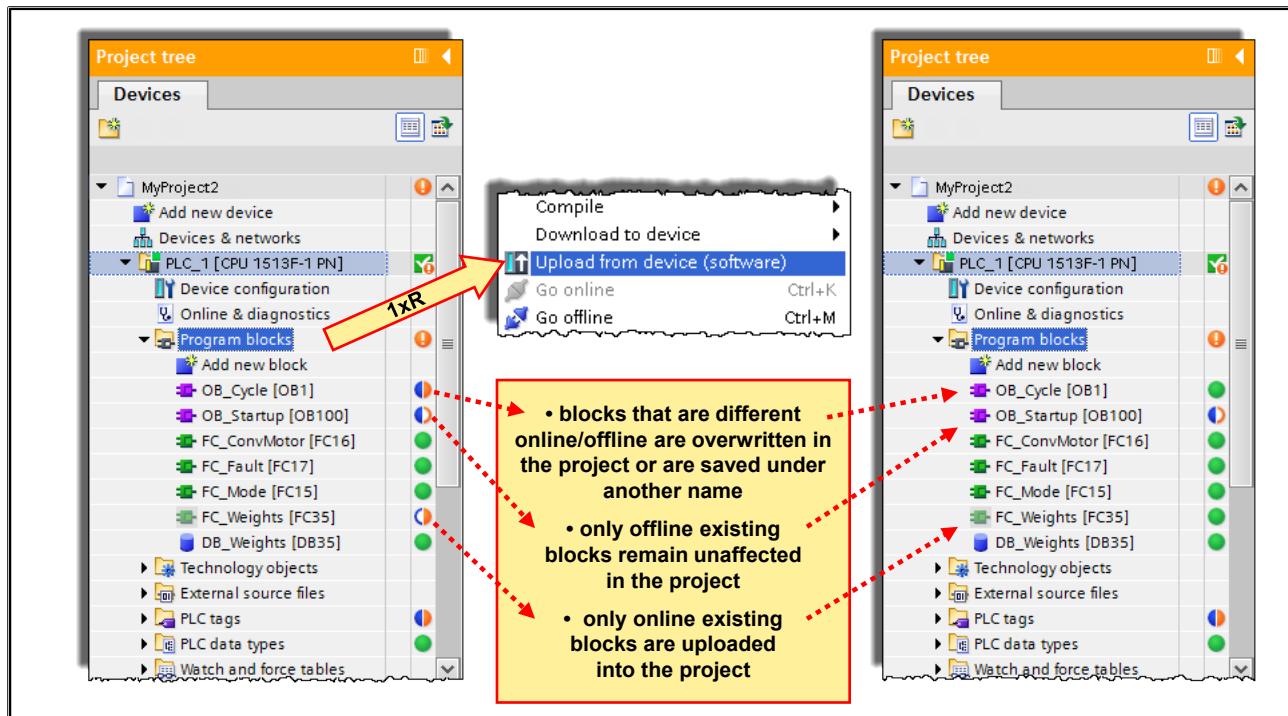


Deleting Blocks

Online, blocks cannot be deleted directly in the CPU. If a block (even with an existing online connection) is selected and "Delete" is activated, the dialog shown in the picture appears with the question of whether the block is to be deleted *offline*.

With subsequent, consistent loading of the entire program, the blocks which only exist in the CPU are deleted there online.

7.8. "Upload" Blocks "from Device" (Upload into Project)



Uploading Blocks into the Project:

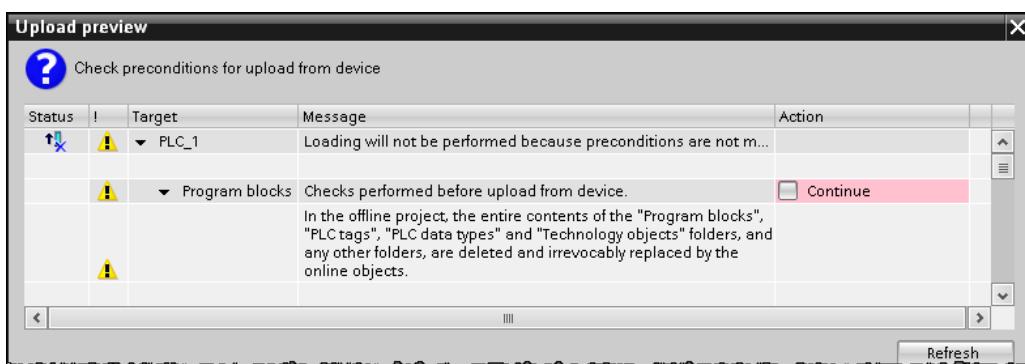
With "Upload from device (software)", individual blocks or the entire CPU program including technology objects, PLC tag tables and PLC data types can be uploaded into the project from the CPU.

If the Blocks folder or individual blocks is/are selected, then...

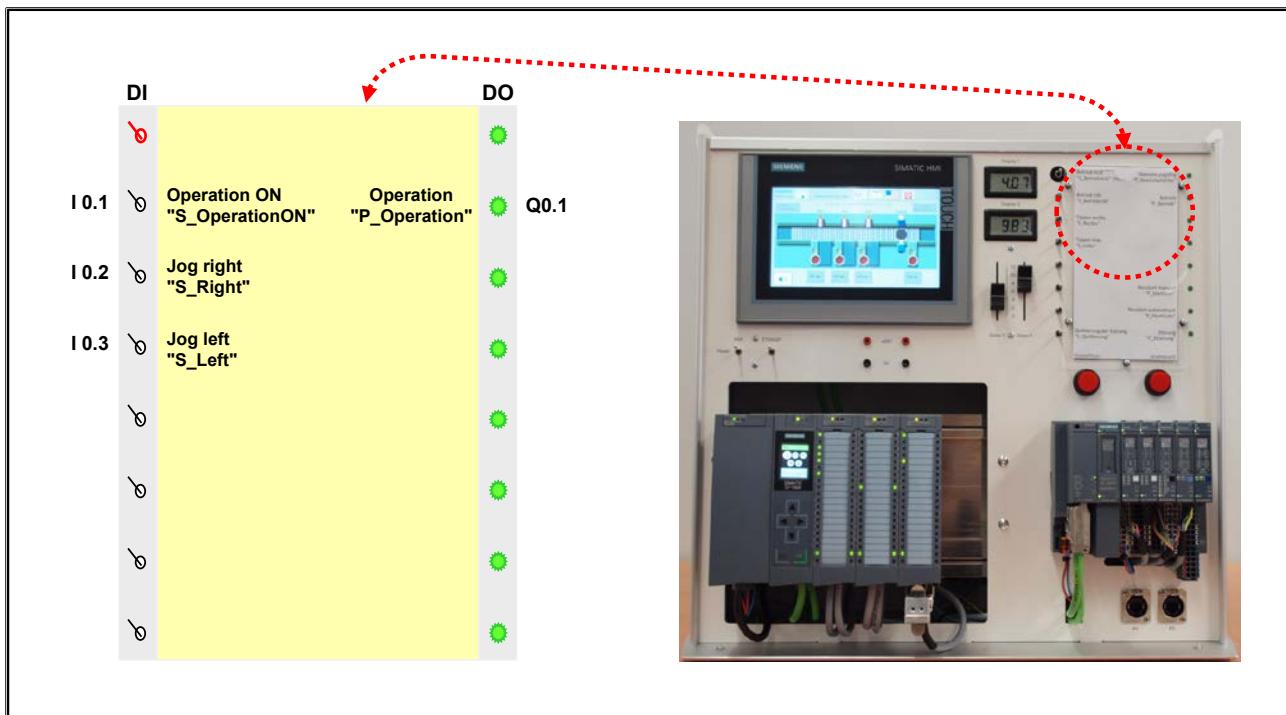
- ... blocks that only exist online in the CPU are uploaded into the offline project.
- ... blocks that are different online / offline are either overwritten in the offline project with the uploaded blocks, or the uploaded blocks are additionally saved under a different name (but with the same number!) in the project. (selectable in the Upload dialog)
- ... blocks that only exists offline are not affected.

If the Station is selected, then...

- ... offline all blocks, PLC data types, technology objects and symbols are **deleted (!)** and the online blocks, PLC data types, technology objects and symbols are uploaded into the project.



7.9. Task Description: Programming the Mode Selection and Manual Mode

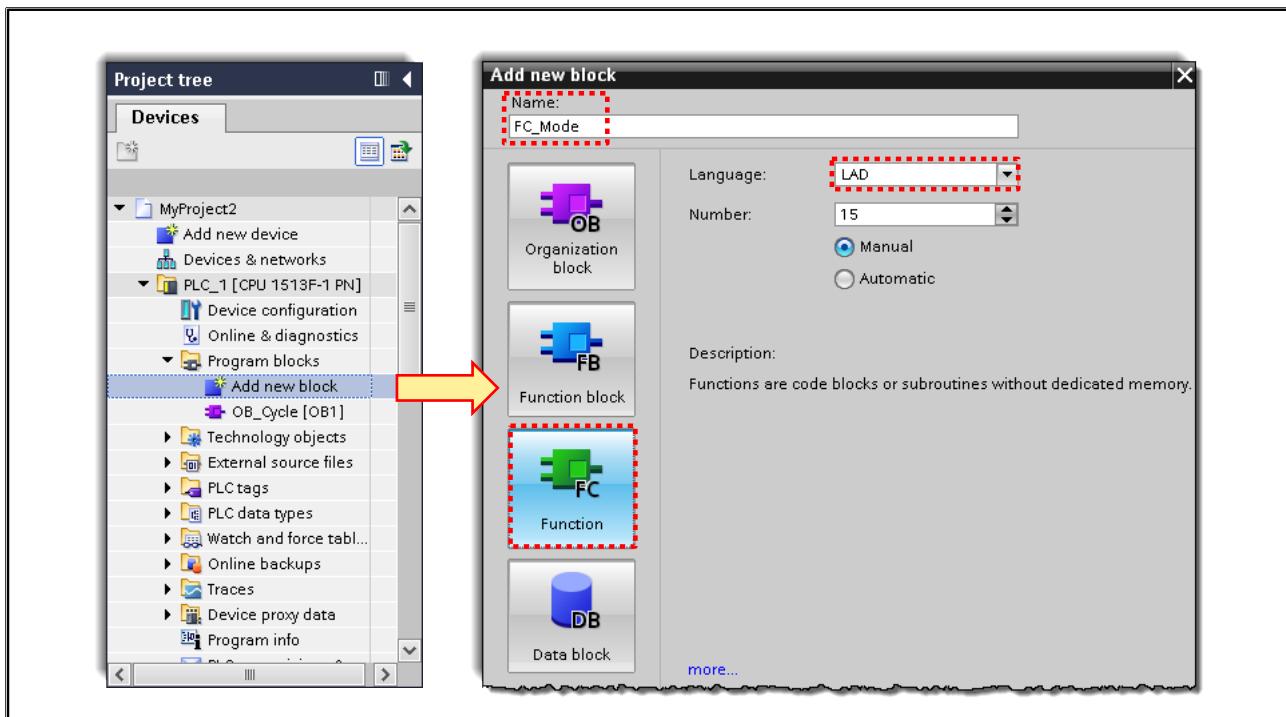


Task Description

The Plant is switched ON/OFF and in Automatic or Manual mode using the switch "S_OperationON".

In Manual mode ("P_Operation" = FALSE), the distribution conveyor can be moved to the right and left using the switches "S_Right" and "S_Left". If both switches are activated simultaneously, then the conveyor must not move (Lock-out!).

7.9.1. Exercise 1: Adding the "FC_Mode" Block



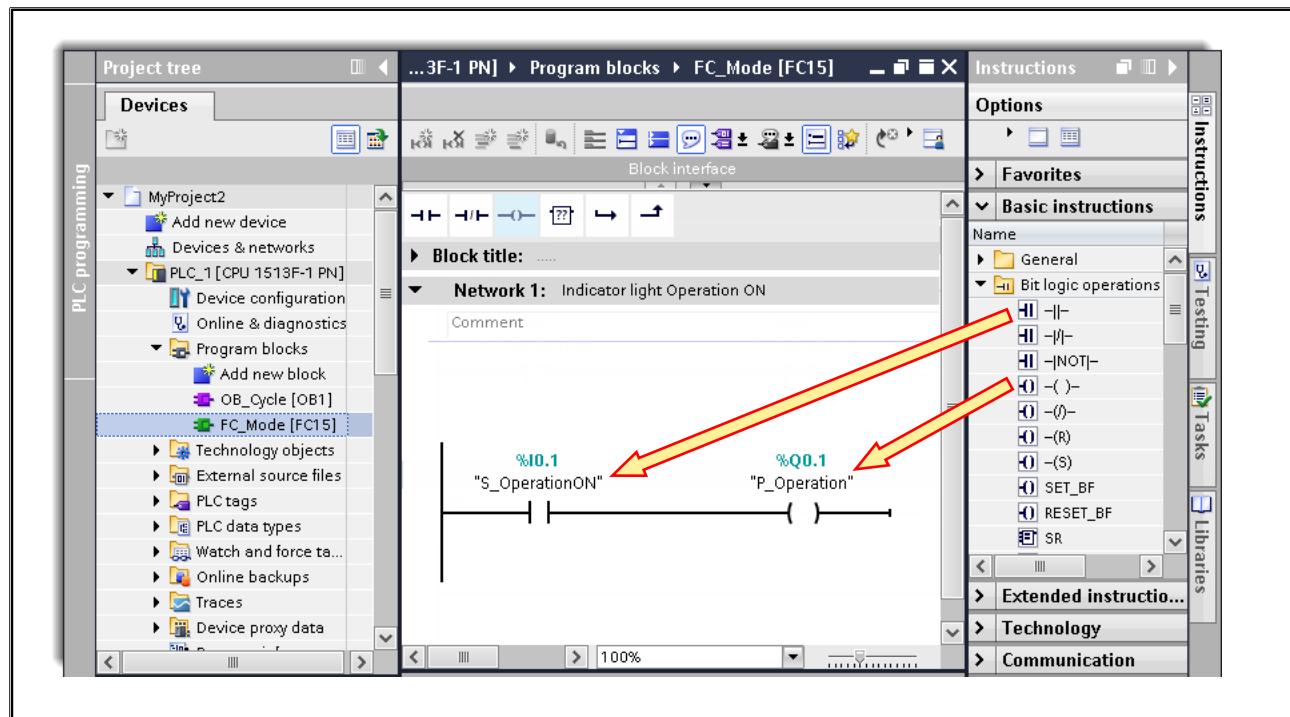
Task

You are to create "FC_Mode" as a new block in which you will program the required mode selection in the next exercise.

What to Do

1. In the "Program blocks" container, double-click on "Add new block".
2. In the dialog that appears, make the entries as shown in the picture above.

7.9.2. Exercise 2: Programming the Mode Selection in "FC_Mode"



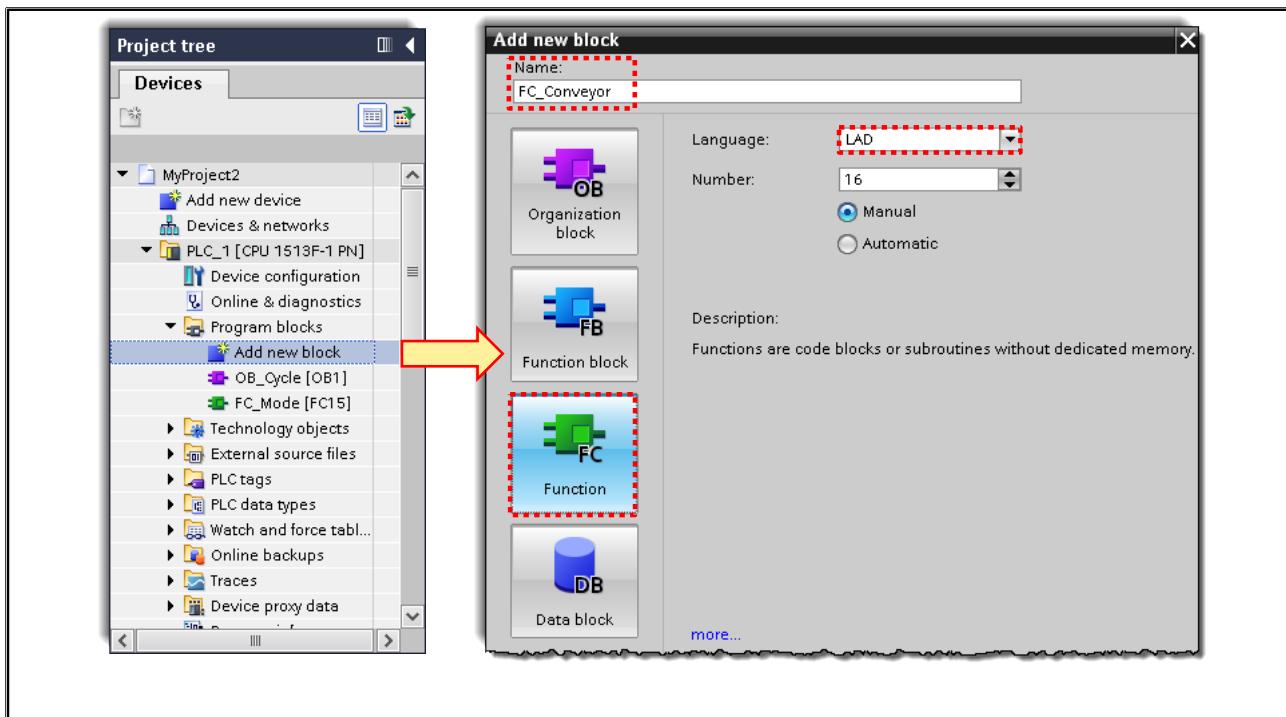
Task

You are to program the mode selection as shown in the picture.

What to Do

1. Program an Instruction by pulling it from the "Instructions" task card using drag & drop.
2. Above the instruction, enter the output "P_Operation" (Q0.1) as the operand (you can enter the symbol as well as the absolute address)
3. Assign the tag "S_OperationON" (I 0.1) as the operand to the input of the Instruction.
4. Label both the block and the network.
5. Close the block.
6. Save your project.

7.9.3. Exercise 3: Adding the "FC_Conveyor" Block



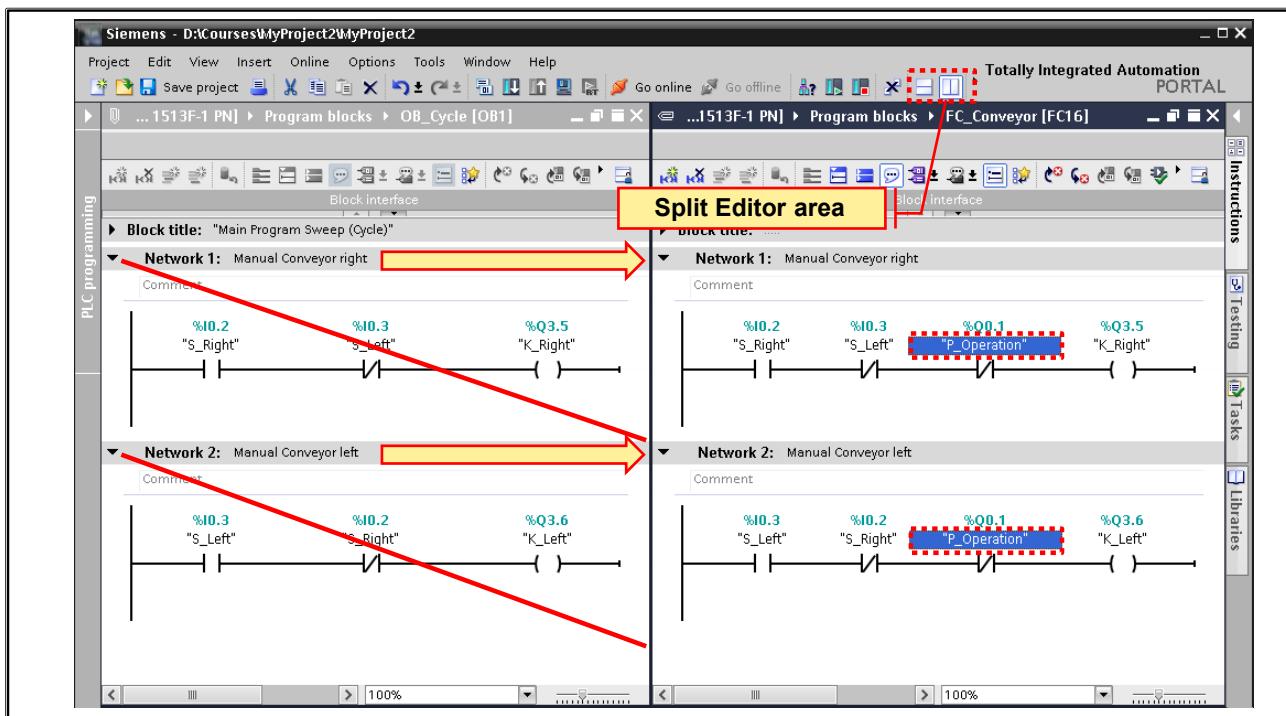
Task

You are to create the function "FC_Conveyor" as a new block in which you will then program the Jog mode that is possible in Manual mode in the next exercise.

What to Do

1. In the "Program blocks" container, double-click on "Add new block".
2. In the dialog that appears, make the entries as shown in the picture above.

7.9.4. Exercise 4: Shifting the Networks from "OB_Cycle" to "FC_Conveyor" and Expanding it



Task

You are to program the Jog mode of the conveyor as shown in the picture.

What to Do

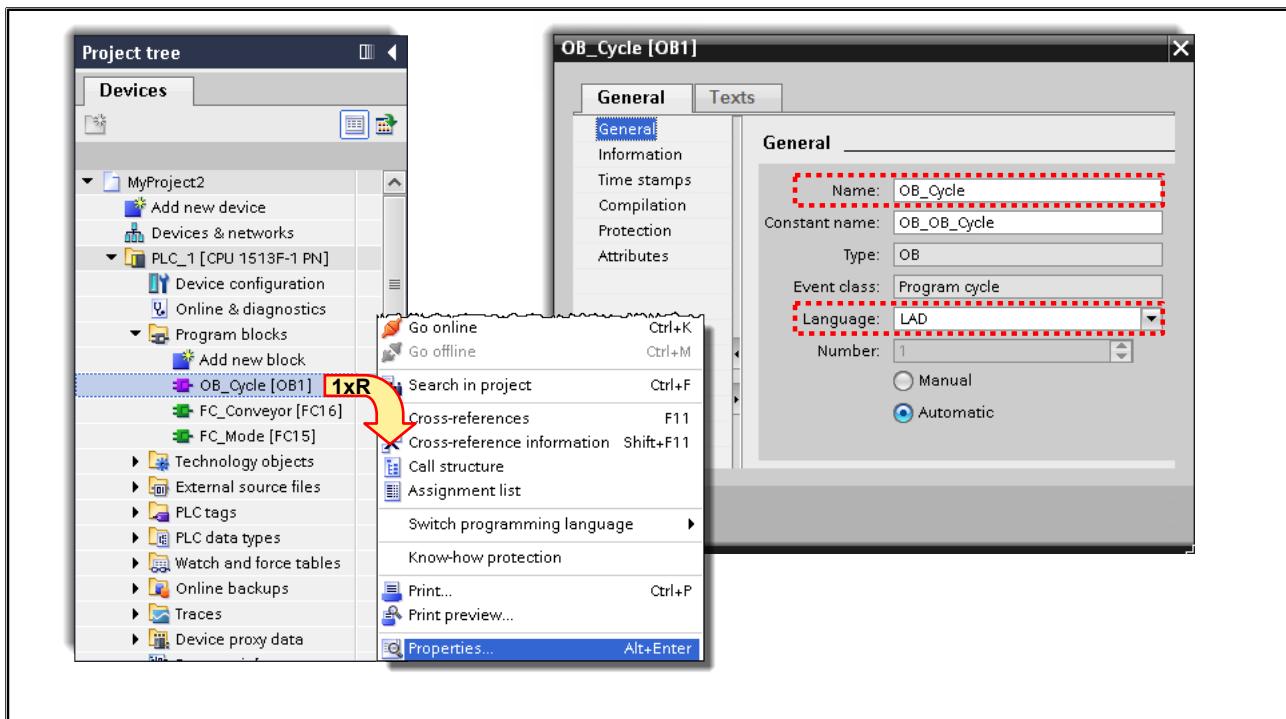
1. Split the Editor (area) horizontally or vertically via one of the appropriate buttons in the toolbar.
2. In each of the areas, open the blocks "OB_Cycle" and "FC_Conveyor".
3. Copy the networks from the "OB_Cycle" block into the "FC_Conveyor" blocks using drag & drop.
4. Since Jogging the conveyor belt is only to be possible when "P_Operation" is FALSE, expand the AND logic in the first network by adding a "Normally closed contact" You do this by dragging it from the Task Card "Instructions" or from the "Favorites".

Note:

If an incorrect type of contact was inserted, click on the contact and use the small red "arrow" in the right corner of the contact to access an expansion menu with the available changes.

5. Assign the "P_Operation" (Q0.1) tag to the new "Normally closed contact" ..
6. Change the logic in Network 2 for Jogging the conveyor to the left in the same way.
7. Delete the networks in "OB_Cycle".
8. Close the blocks.
9. Save your project.

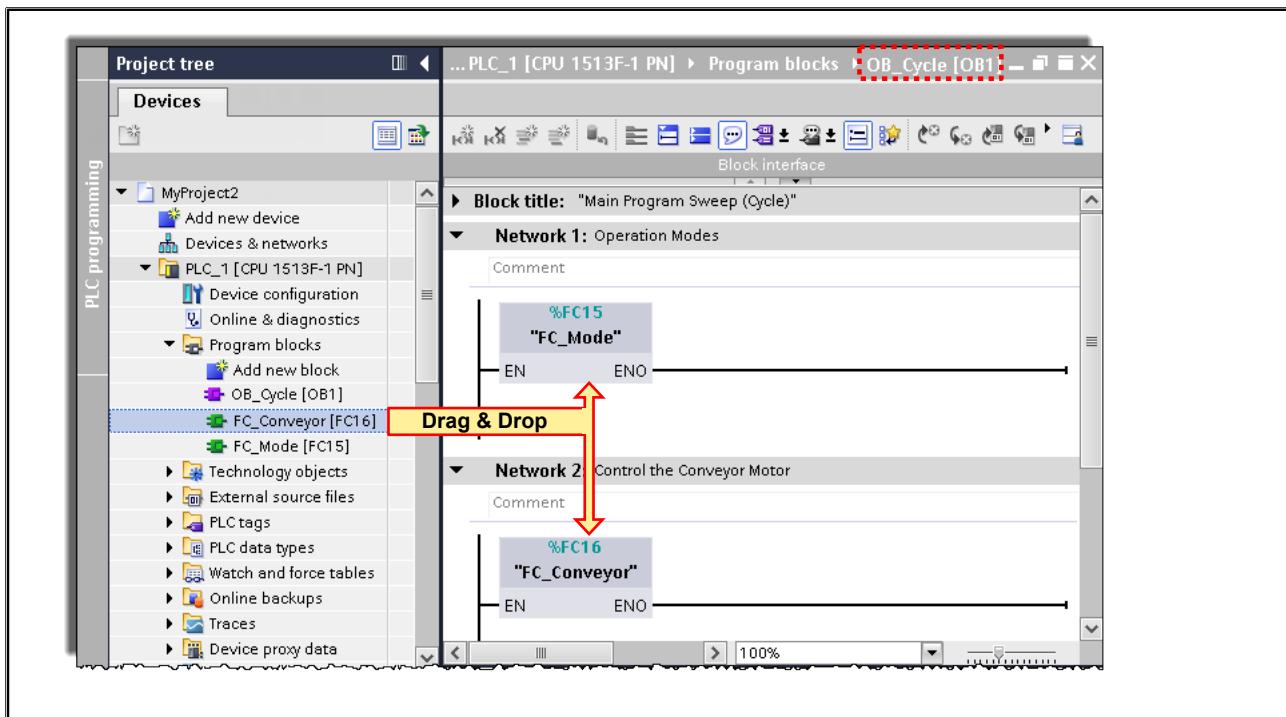
7.9.5. Exercise 5: Checking the "OB_Cycle" Properties



Task

You are to check the Properties of the OB1 block as shown in the picture, and, if necessary, correct them.

7.9.6. Exercise 6: Calling "FC_Mode" and "FC_Conveyor" in "OB_Cycle"



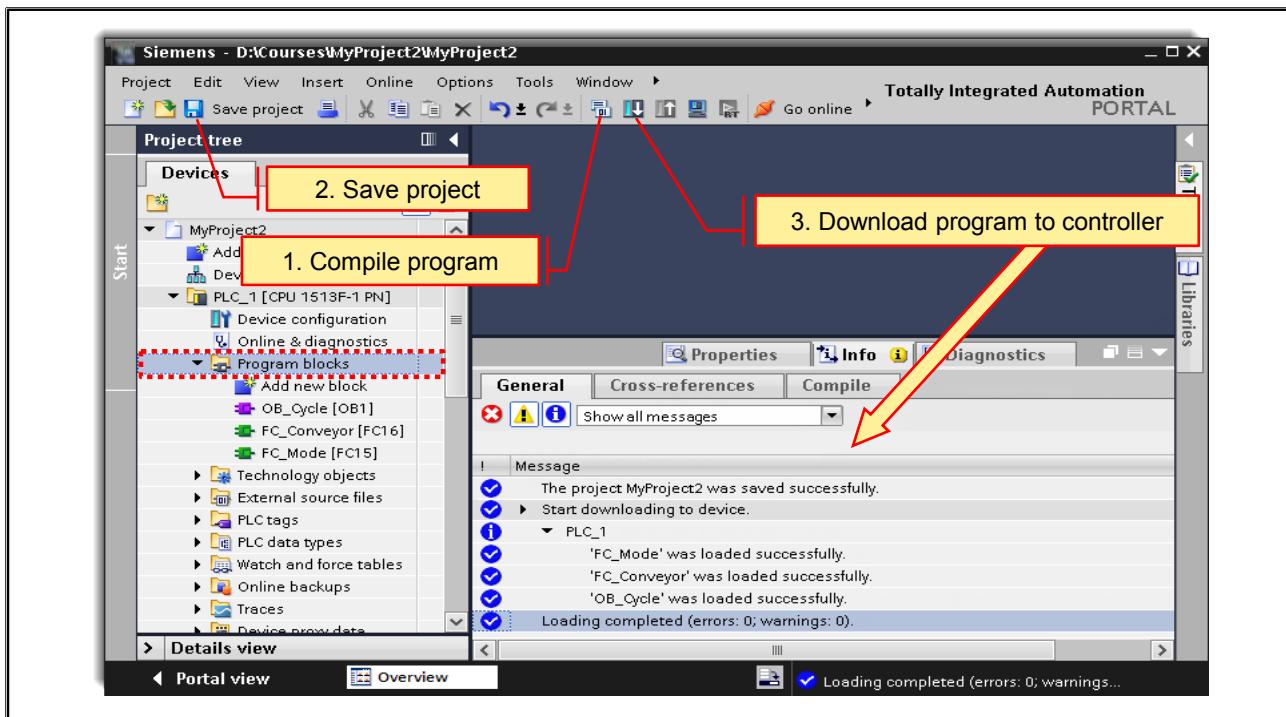
Task

So that the newly created blocks "FC_Mode" and "FC_Conveyor" can be executed cyclically, they must be called in "OB_Cycle".

What to Do

1. Open the "OB_Cycle" block by double-clicking on it.
2. Program the calls of the "FC_Mode" and "FC_Conveyor" blocks as shown in the picture using drag & drop.
3. Edit the labels for the Networks (as in the picture).
4. Close the block.
5. Save your project.

7.9.7. Exercise 7: Compiling, Downloading and Saving the Program



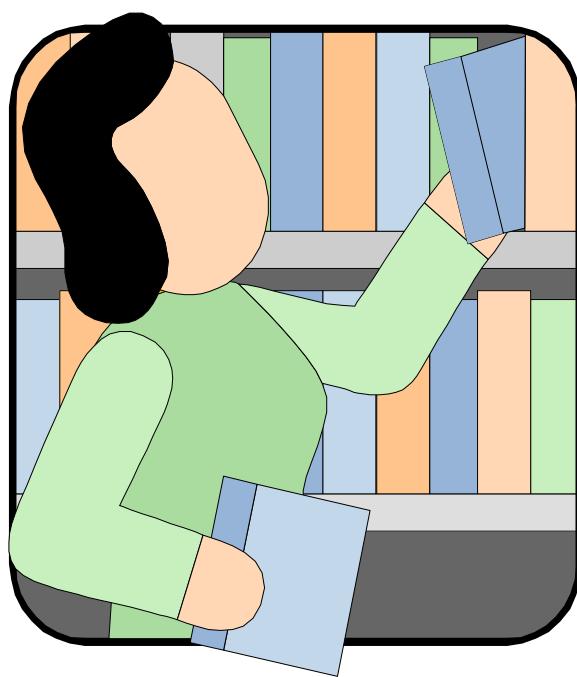
Task

The newly programmed blocks are to be compiled, downloaded into the CPU and saved offline in the project data storage.

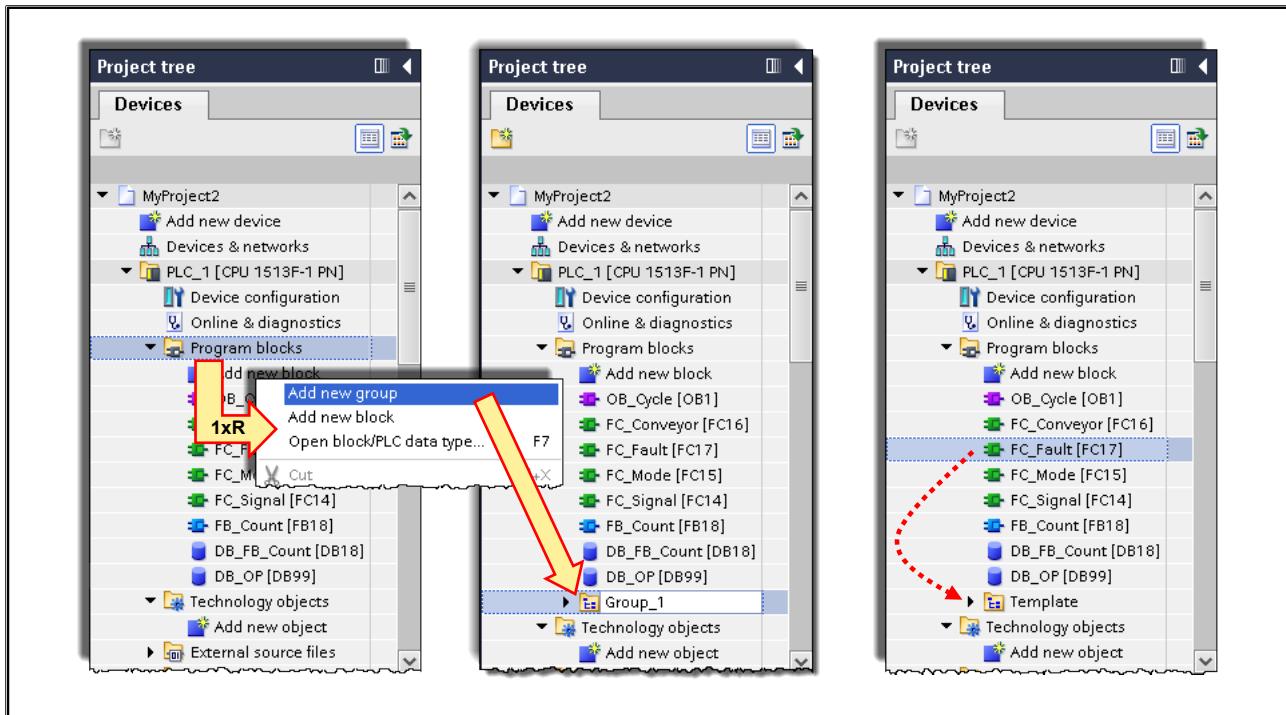
What to Do

1. To compile the entire program (all blocks) and to download it into the CPU, select the "Program blocks" folder in the Project tree.
2. Carry out the steps shown in the picture and check the program functioning
 - a. Switching between Manual and Automatic mode by activating the simulator switch "S_OperationON" accordingly as well as checking the LED "P_Operation".
 - b. Try jogging with Manual or Automatic mode by activating the simulator switches "S_Right" (I 0.2) and "S_Left" (I 0.3).
3. If necessary, also use the Test function "Monitor in block" or use a watch table.

7.10. Additional Information



7.10.1. Block Groups

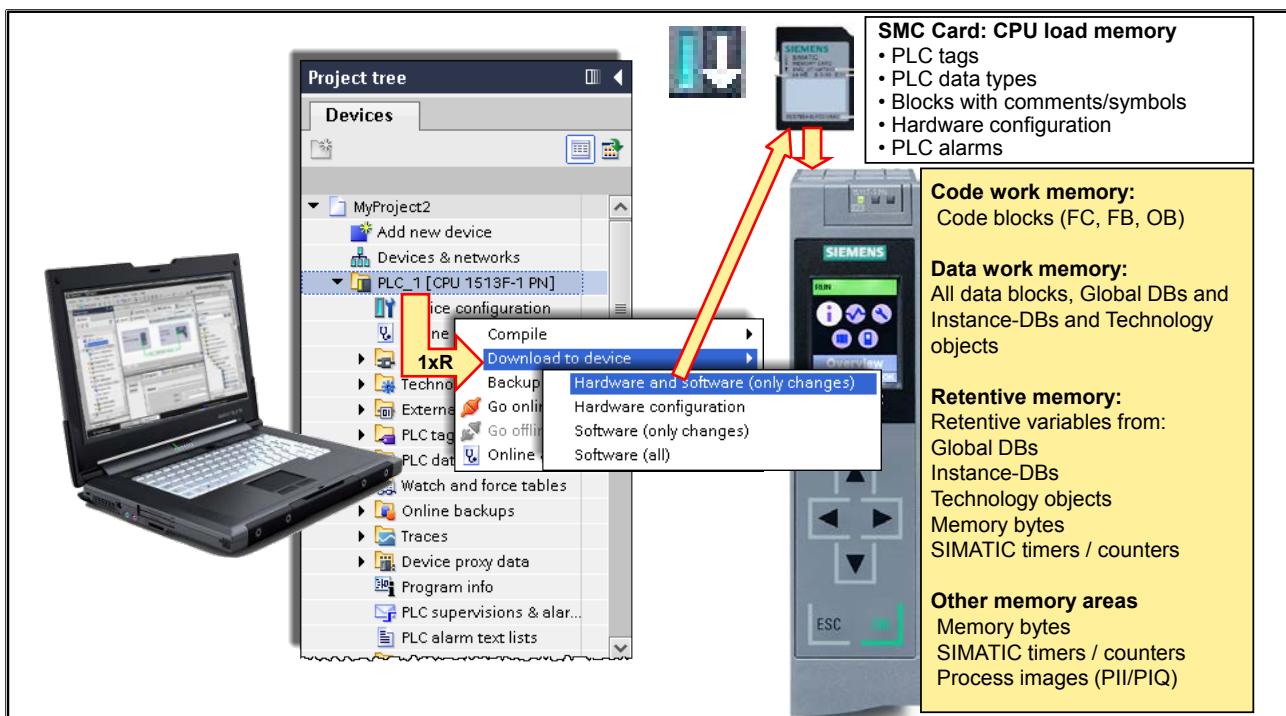


Block Groups

To achieve more clarity, large programs with many blocks can be divided into several block groups. The groupings can, for example, be related to the structure of the system to be controlled. Even if the blocks are managed in different groups, each block must have a unique symbolic name. Regardless of the groupings, the sum of all blocks represents the user program of the controller.

The blocks can easily be shifted between the block groups using drag & drop.

7.10.2. S7-1500 - Memory Concept



Memory Areas of the CPU

The SMC (Simatic Memory Card) is the load memory of the CPU. Accordingly, an inserted SMC is absolutely necessary for operating the CPU.

The data of the entire station is stored on the SMC, that is, the complete S7 program including documentation, PLC tags and data types as well as the complete hardware configuration including distributed I/O and parameter assignments. In addition, other files can also be found on the SMC, such as, recipes, HMI backups.....

When downloading the S7 program into the CPU, all blocks are always first loaded into the load memory from where the CPU automatically copies the parts of the blocks that are relevant for execution into the work memory.

Work Memory

The work memory is a volatile RAM memory which cannot be expanded. It is divided into two areas:

- Code work memory which contains the parts of the code (logic) blocks relevant for execution.
- Data work memory which contains the data of the data blocks and technology objects relevant for execution.

Retentive Memory Depends on the CPU (88-700kByte)

The retentive memory is a non-volatile memory for saving the variables declared as retentive whose value is retained during a power failure.

The contents of the retentive memory is only erased by a

- memory reset
- resetting of the CPU to factory settings

8

Contents

8.	Binary Operations	8-2
8.1.	Task Description: The Conveyor Model as Distribution Conveyor with Manual and Automatic Mode	8-3
8.2.	Set, Reset	8-4
8.2.1.	Flip Flops.....	8-5
8.3.	Task Description: Expanding the "FC_Mode" Function.....	8-6
8.3.1.	Exercise 1: Programming the "FC_Mode" Block Expansion	8-7
8.4.	Task Description: Parts Transportation in Automatic Mode	8-8
8.4.1.	Multiple Assignment.....	8-9
8.4.2.	Exercise 2: Expanding "FC_Conveyor" (Automatic Mode).....	8-10
8.5.	Task Description: Parts Transportation THROUGH the Light Barrier	8-11
8.5.1.	Signal Edge Evaluation.....	8-12
8.5.2.	Exercise 3: Integrating an Edge Evaluation in "FC_Conveyor"	8-14
8.6.	Task Description: Controlling the Indicator Lights, Commissioning "FC_Signal"	8-15
8.6.1.	Exercise 4: Writing/Correcting "FC_Signal" and Commissioning It	8-16
8.7.	Additional Exercise: Optimizing "FC_Mode".....	8-17
8.8.	Additional Information	8-18
8.8.1.	Jump Instructions JMP, JMPN, RET.....	8-19

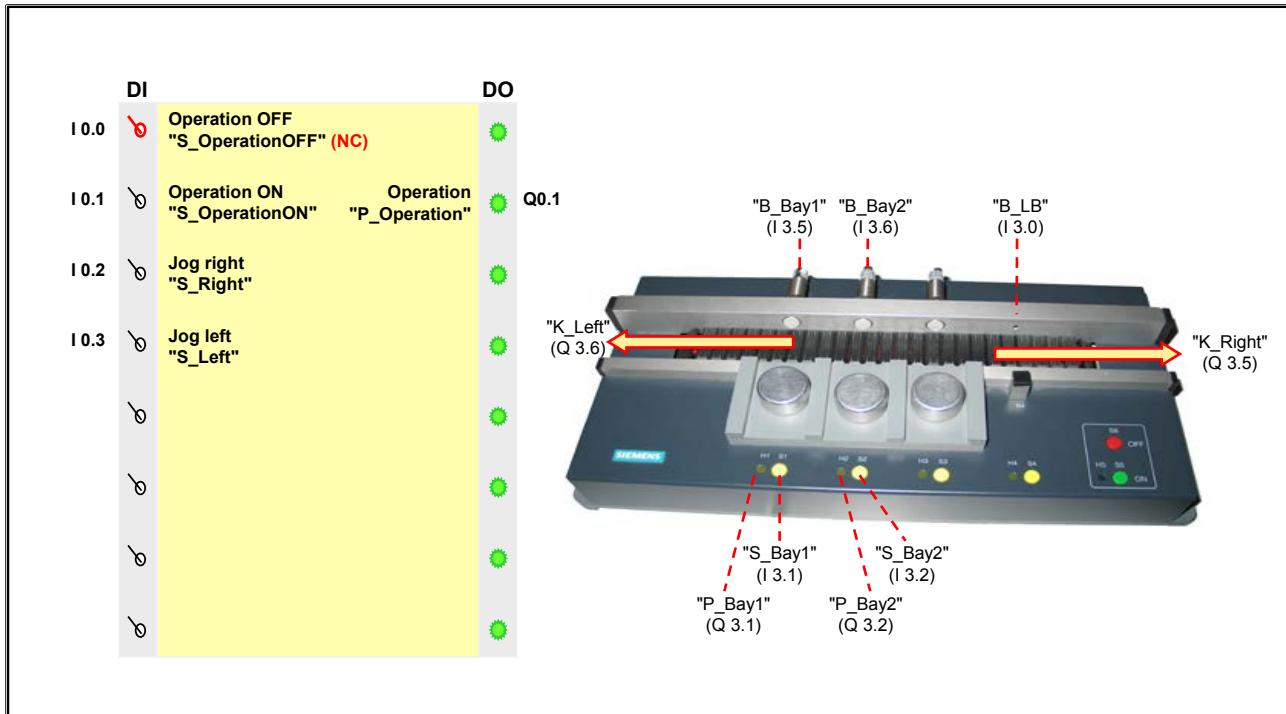
8. Binary Operations

At the end of the chapter the participant will ...



- ... be familiar with the instructions Set, Reset and will be able to apply them
- ... know what needs to be considered for a multiple assignment
- ... know how edges are evaluated
- ... be familiar with jump instructions as well as absolute and conditional block aborts

8.1. Task Description: The Conveyor Model as Distribution Conveyor with Manual and Automatic Mode



Function Description

The distribution conveyor is used to transport parts from Bay 1 or 2 to the light barrier bay. The Operation (Simulator LED "P_Operation", Q0.1) can be switched on and switched off (Manual / Automatic mode) via the simulator switch "S_OperationON" (I0.1) and "S_OperationOFF" (I0.0, NC).

- In Manual mode, switched off "P_Operation" (Q0.1)...
 - ...the distribution conveyor can be jogged to the right via the simulator switch "S_Right" (I 0.2) and jogged to the left via the simulator switch "S_Left" (I 0.3).
- In Automatic mode, switched on "P_Operation" (Q0.1)...
 - ... parts are transported from Bay 1 or 2 up to, or, through the light barrier. For this, the part must be placed on the conveyor at exactly one of the two bays and the associated bay pushbutton pressed.

The indicator lights at Bays 1 and 2 show...

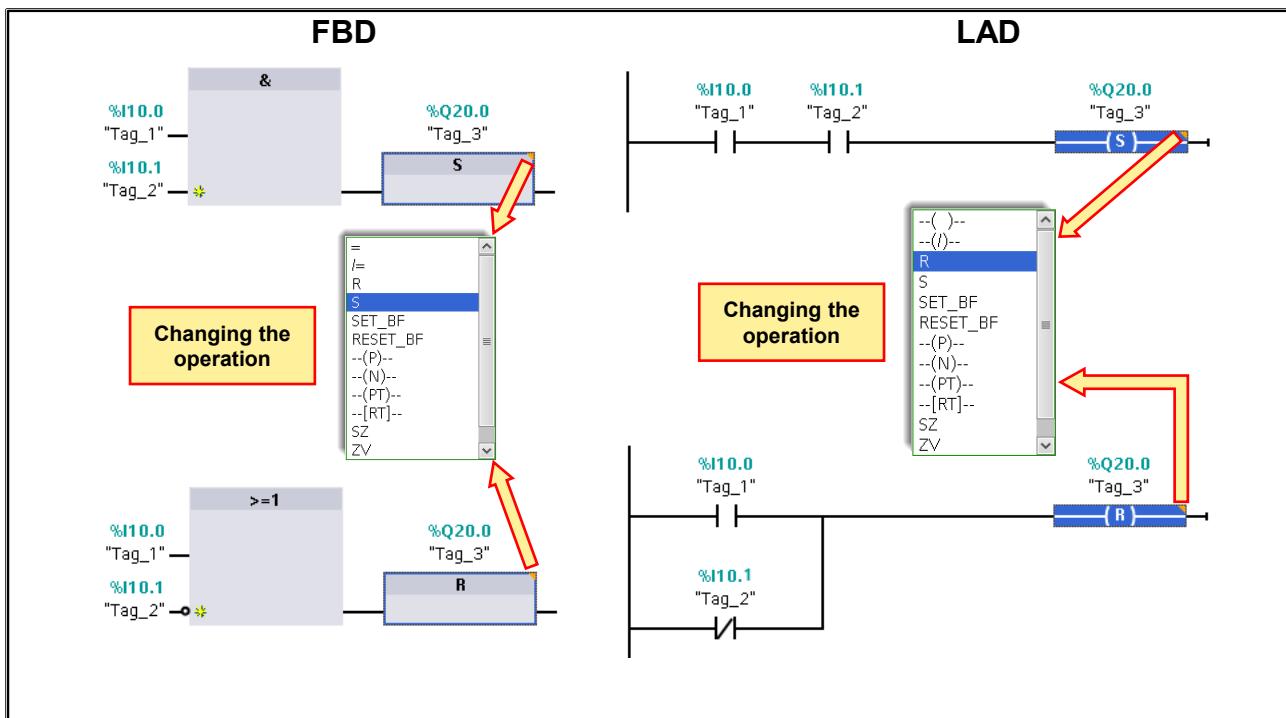
... a constant light when a new part may be placed on the conveyor (distribution conveyor is stopped and both proximity sensors are free)

... a 1Hz flashing light at the bay where a part is detected by the associated proximity sensor, however, only as long as the conveyor has not yet been started (if parts are placed on the conveyor at both proximity sensors, neither indicator light must light up)

... a 2Hz flashing light as long as the distribution conveyor is running.

The indicator light at the light barrier bay shows a 2Hz flashing light as long as the distribution conveyor is running.

8.2. Set, Reset



Set

If the result of a logic operation (RLO), that is, result of scan = "1", the specified operand is assigned Status '1'; for RLO / result of scan = "0", the status of the operand remains unchanged.

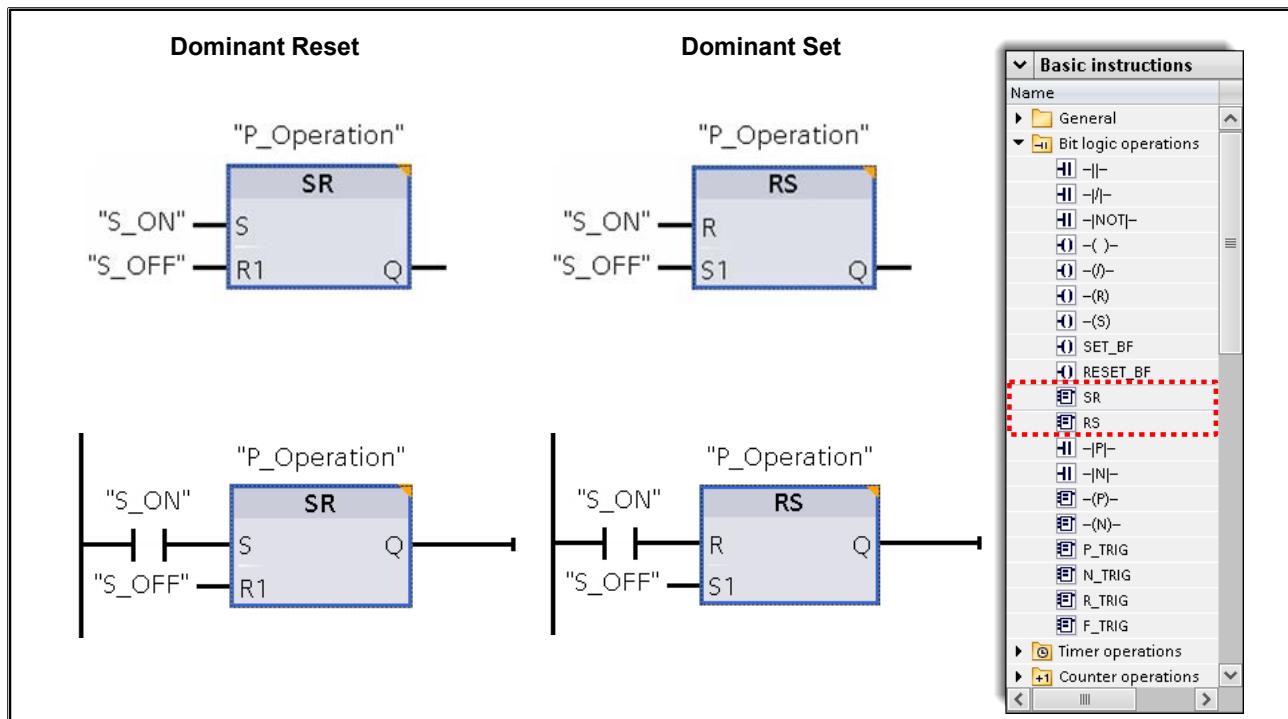
Reset

If the RLO, that is, result of scan = "1" the specified operand is assigned Status ' 0'; otherwise the status of the operand remains unchanged.

Note

In addition to the SET and RESET functions, the graphic above also shows the methodology to change an elements attributes without deleting and inserting a new element type. Simply highlight the element and click the small red "array" in the right corner of the element. Available changes are shown in a drop down menu.

8.2.1. Flip Flops



Memory Function "Flip Flop"

A flip flop has a Set input and a Reset input. The operand is set (TRUE) or reset (FALSE), depending at which input the scan condition is fulfilled.

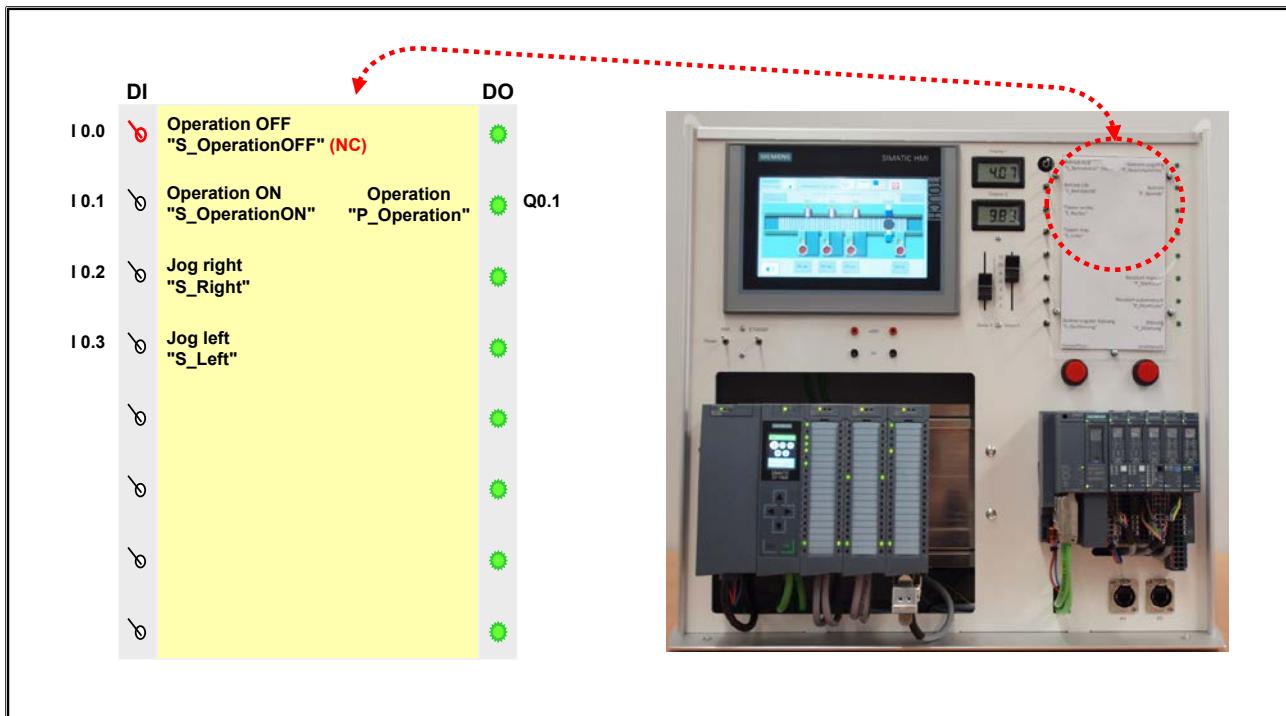
Priority

If the scan condition at both inputs is fulfilled simultaneously, then the priority of the operation decides whether the operand is set or reset. For that reason, there are different symbols for the Dominant Set and Dominant Reset memory functions in LAD and FBD. (The last scan always has priority, Set for "RS" and Reset for "SR").

Note

With a CPU restart, all outputs are reset. That is, they are overwritten with Status '0'.

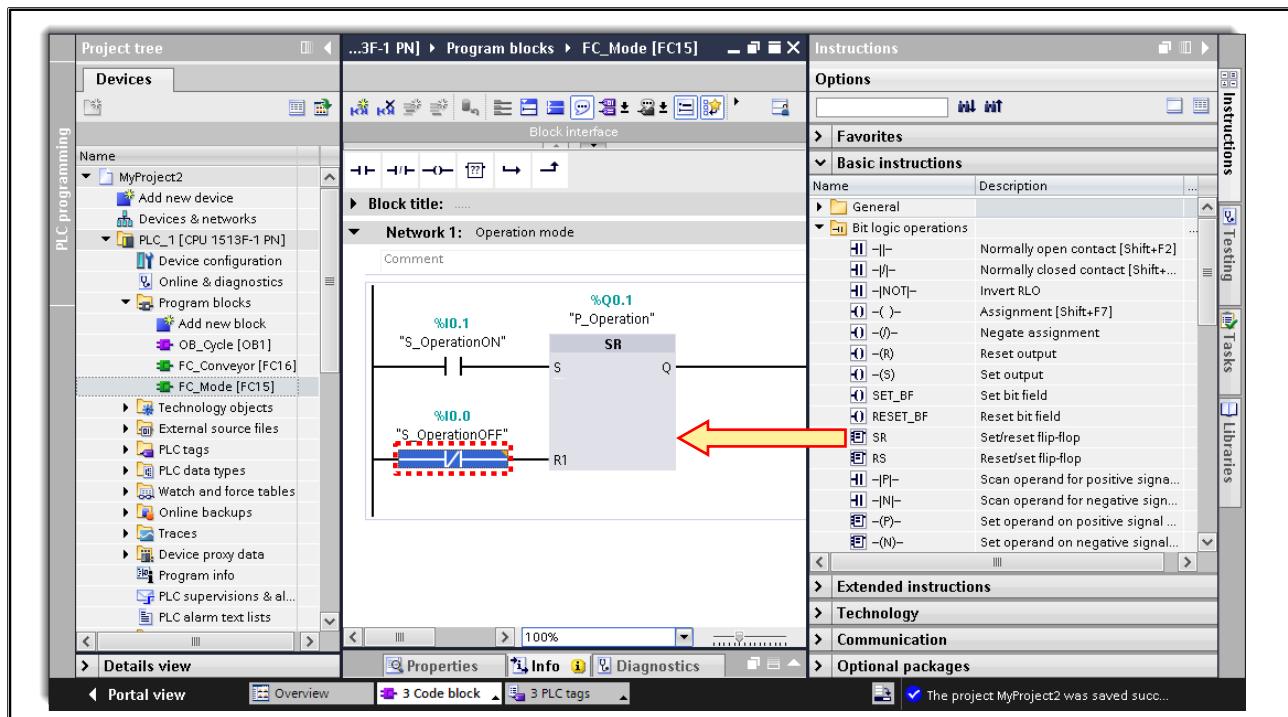
8.3. Task Description: Expanding the "FC_Mode" Function



Task Description

The "FC_Mode" function is to be changed in such a way that the output "P_Operation" is not assigned via the input "S_OperationON" but set by the input (NO) "S_OperationON" and reset via the input (NC) "S_OperationOFF".

8.3.1. Exercise 1: Programming the "FC_Mode" Block Expansion



Task

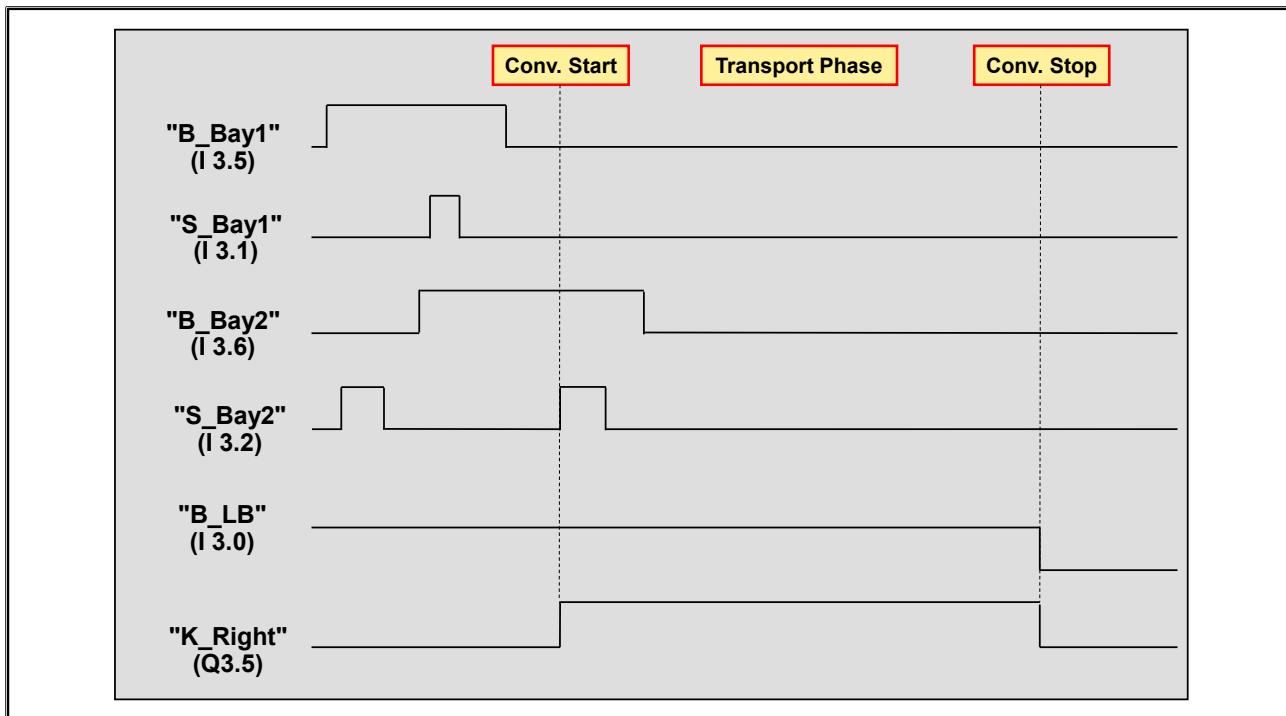
In "FC_Mode" you are to change the code in the operating mode section for the distribution conveyor:

The operation, that is, the indicator light "P_Operation" (simulator LED Q0.1) is switched on via the simulator switch "S_OperationON" (I 0.1) and is switched off via the simulator switch "S_OperationOFF" (I 0.0, **NC**).

What to Do

1. Open the "FC_Mode" block.
2. Delete the assignment programmed in the last chapter.
3. Open the "Instructions" Task Card and drag the instructions shown in the picture into the block using drag & drop.
4. Link the input for 'Set' with the variable "S_OperationON" and the 'Reset' input with the variable "S_OperationOFF".
5. Since NC contacts are used to switch off systems for safety reasons (interruption-safe), the input "S_OperationOFF" must be negated.
6. Above the instruction, specify "P_Operation" as the operand.
7. Save, compile, download and test your newly programmed function.

8.4. Task Description: Parts Transportation in Automatic Mode

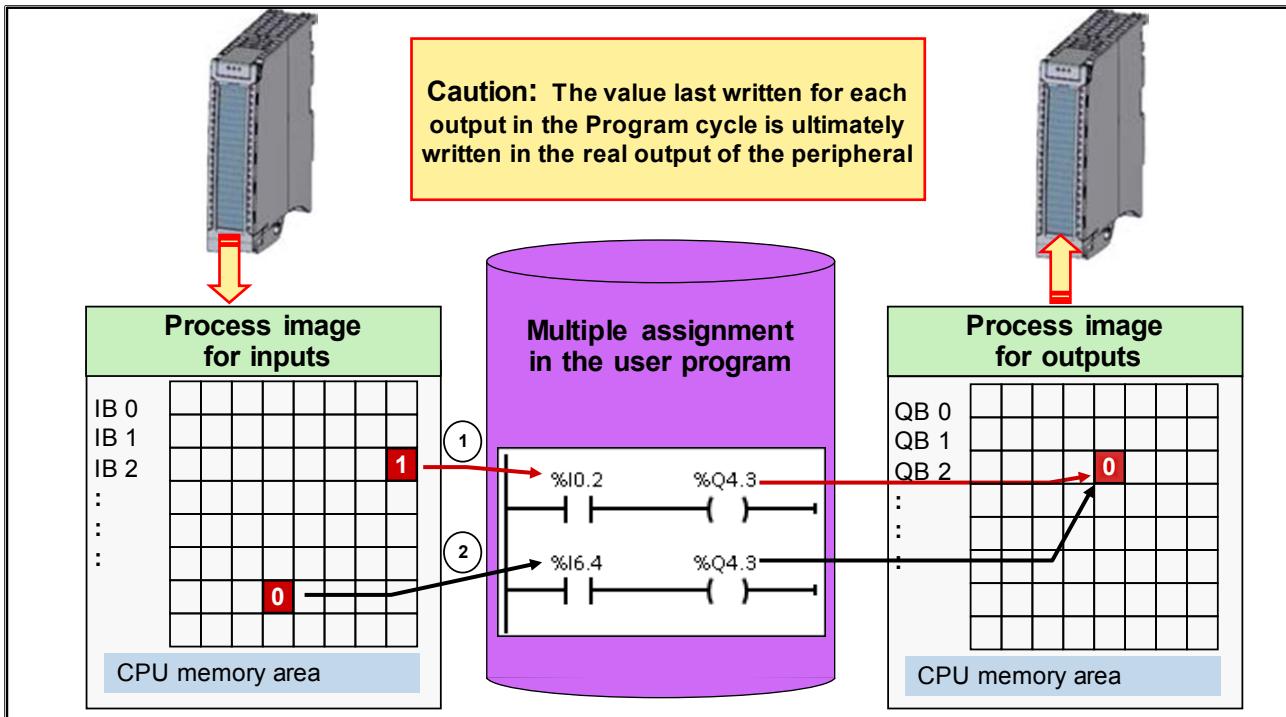


Task Description:

When "P_Operation" (Q0.1) is switched on, parts are to be transported from Bay 1 or Bay 2 into the light barrier "B_LB" (I 3.0). The precondition is that a part is only placed on the conveyor at one of the two bays. If parts are placed on the conveyor at both bays, no transport sequence can be started.

- The conveyor motor "K_Right" (Q3.5) is started when
 - at Bay 1 the proximity sensor "B_Bay1" (I 3.5) is occupied AND at Bay 2 the proximity sensor "B_Bay2" (I 3.6) is NOT occupied AND the pushbutton at Bay 1 "S_Bay1" (I 3.1) is pressed
 OR
 - at Bay 2 the proximity sensor "B_Bay2" (I 3.6) is occupied AND at Bay 1 the proximity sensor "B_Bay1" (I 3.5) is NOT occupied AND the pushbutton at Bay 2 "S_Bay2" (I 3.2) is pressed.
- The conveyor motor "K_Right" (Q3.5) is stopped when
 - the part has reached the light barrier "B_LB" (I 3.0)
 OR
 - "P_Operation" (Q0.1) is switched off.

8.4.1. Multiple Assignment



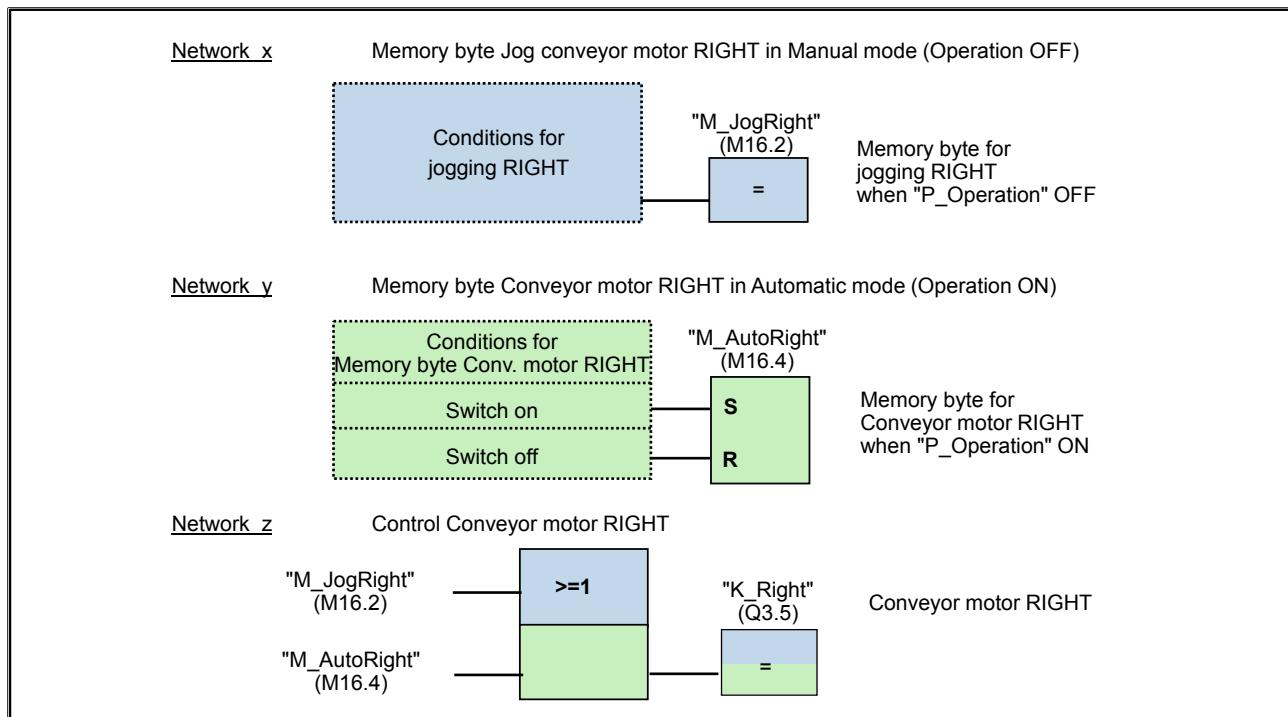
Process Images

For the storage of **all** digital input and output statuses, the CPU has reserved memory areas: the process image for inputs (PII) and the process image for outputs (PIQ). During program execution, the CPU accesses these memory areas exclusively. It does not access the digital input and output modules directly.

"Classic Error": Double Assignment

If an output is assigned a status in several locations in the program, then only the status that was assigned last is transferred to the particular output module. As a rule, these types of double assignments are programming errors.

8.4.2. Exercise 2: Expanding "FC_Conveyor" (Automatic Mode)



Task

You are to expand the "FC_Conveyor" block with the previously described functions.

Solution Notes

The output to move the distribution conveyor to the right "K_Right" (Q3.5) must now be controlled given the following two conditions:

Either when "P_Operation" (Q0.1) is switched off for Jogging RIGHT (in the picture Network x)
OR when "P_Operation" (Q0.1) is switched on under the conditions described in the Task Description (in the picture Network y).

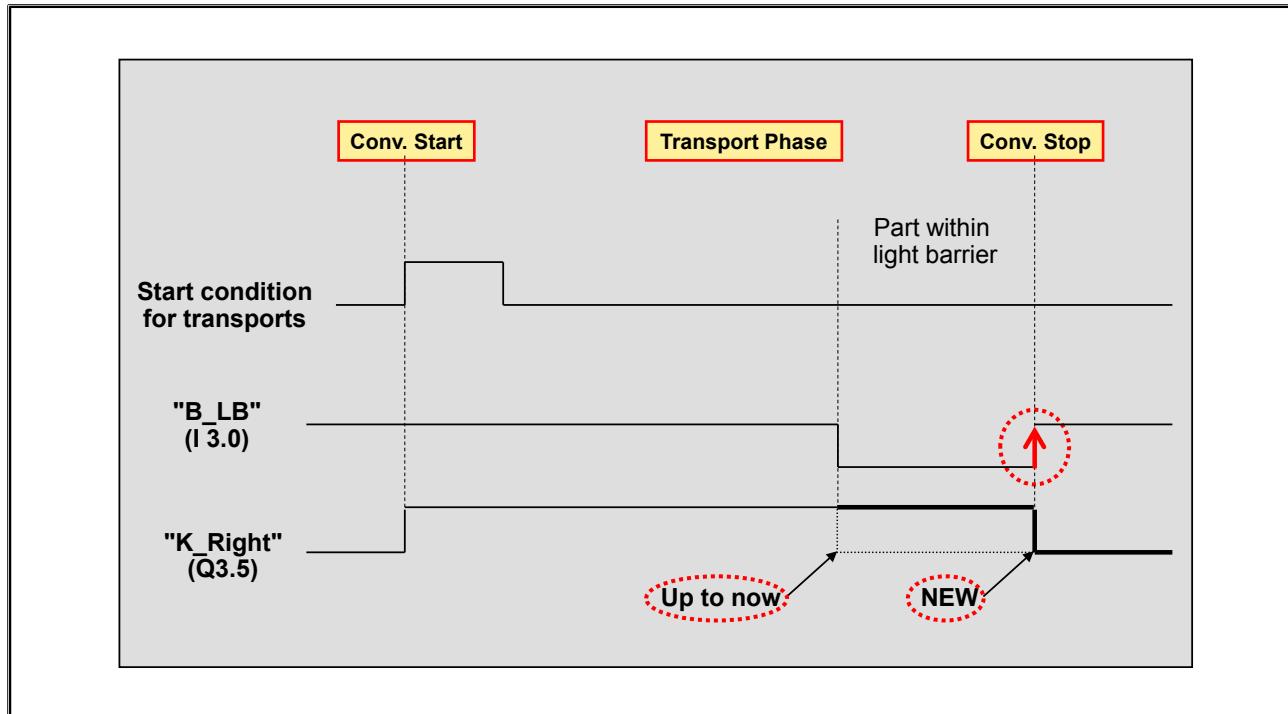
If, in both Network x and y, the result of logic operation of the conditions were each directly assigned to the output "K_Right" (Q 3.5), an error in the form of a double assignment would occur. Jogging the motor RIGHT in manual mode (Network x) would no longer function, since the status assigned to the output here would then be overwritten in Network y.

The problem can be solved by programming a memory bit for each condition or by first assigning the results of the logic operations to a memory bit each in both Network x and y. These are then used in the Network to control the conveyor motor.

What to Do

1. Open the "FC_Conveyor" block.
2. Program the required functions in new networks (see graphic above and the graphic at the previous page). Use the memory bits shown in the picture and provide them with the symbols shown.
3. Modify the already existing network for jogging to the right as described in the solution notes.
4. Download the expanded block into the CPU and check the program function.
5. Save your project.

8.5. Task Description: Parts Transportation THROUGH the Light Barrier



Function Up to Now in FC_Conveyor

When "P_Operation" (Q0.1 = '0') is switched off, you can jog the conveyor motor "K_Right" (Q3.5) using the simulator switches "S_Left" (I 0.3) and "S_Right" (I 0.2).

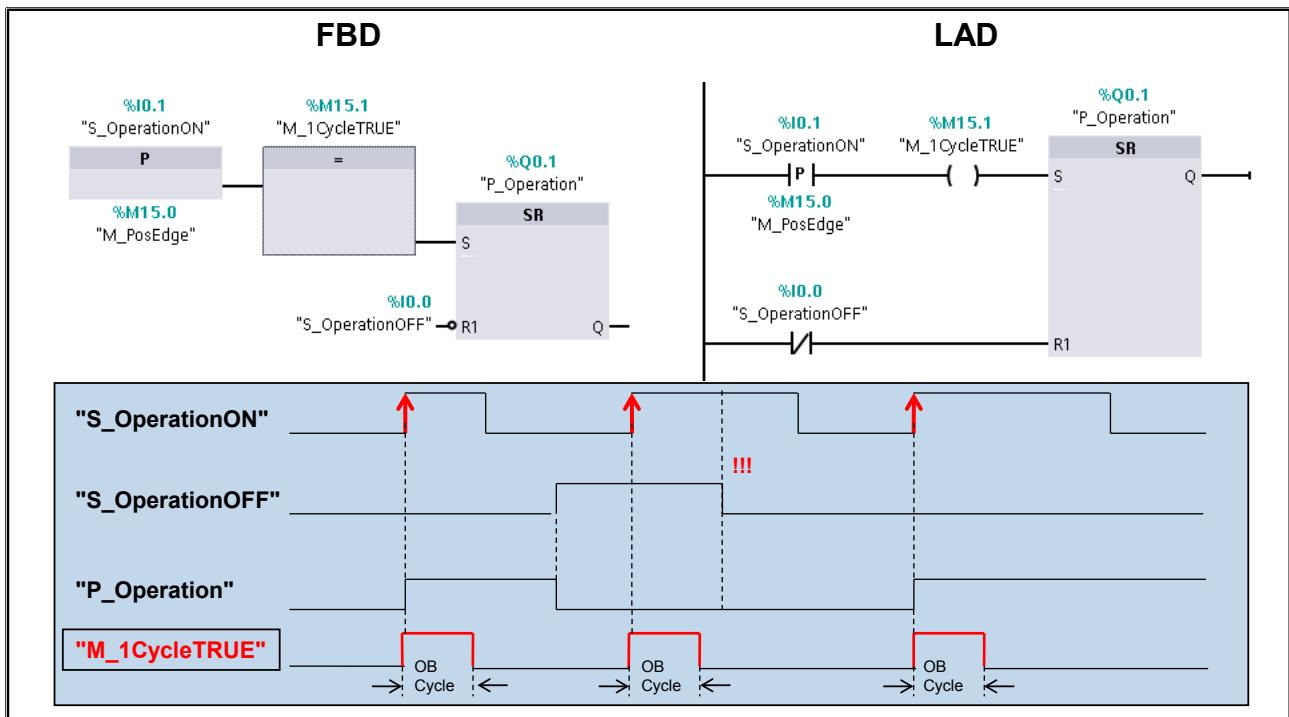
When "P_Operation" (Q0.1) is switched on, the conveyor motor "K_Right" (Q3.5) is switched on when a part is placed on the conveyor exactly in front of one of the proximity sensors of Bay 1 or 2, and the occupied bay's pushbutton is pressed.

The conveyor motor "K_Right" (Q3.5) is stopped when the part has reached the light barrier or "P_Operation" (Q0.1) is switched off.

Task Description:

The function of "FC_Conveyor" to control the conveyor motor when "P_Operation" (Q0.1) is switched on is to remain fundamentally unchanged. However, the conveyor motor is only to stop when the part has passed through the light barrier.

8.5.1. Signal Edge Evaluation



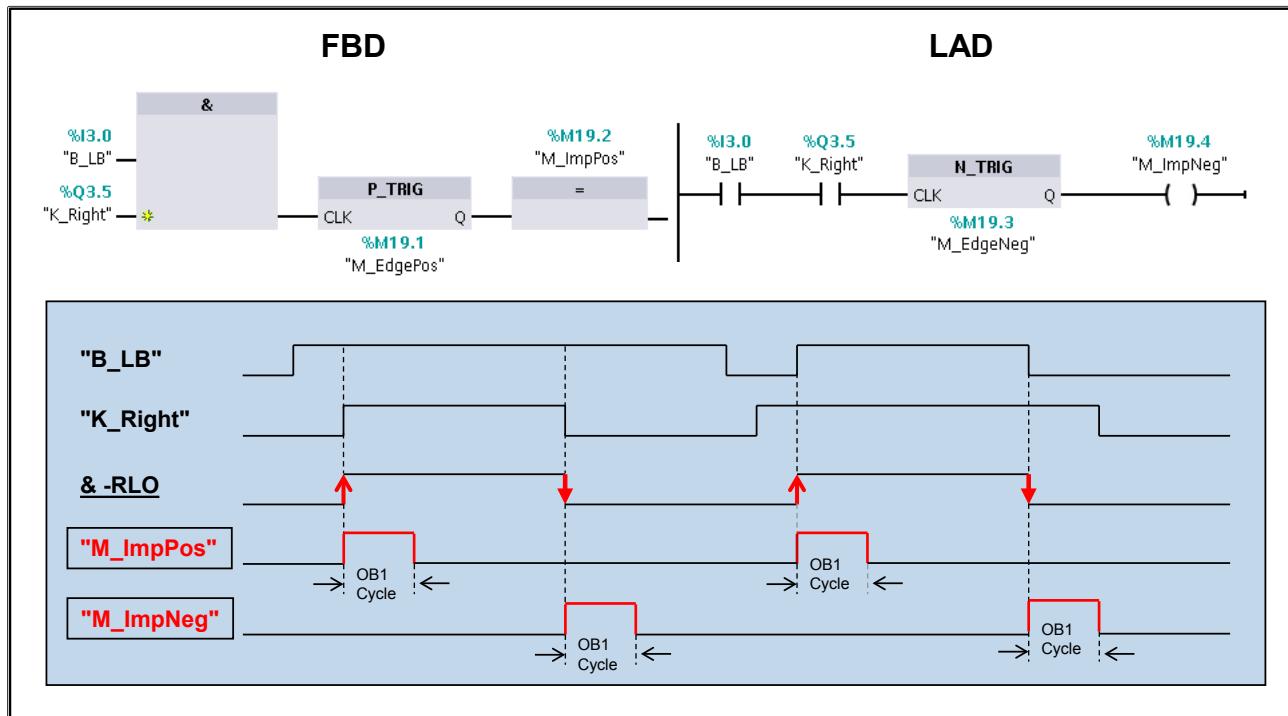
Scanning the Operand for Positive ($-|P|-$) or Negative ($-|N|-$) Signal Edge

With a signal edge evaluation, it is possible to detect whether the status of an individual operand (in the example "S_OperationON") has changed from '0' to '1' (rising or positive edge) or from '1' to '0' (falling or negative edge). If this is the case the instruction supplies RLO '1' as the result, which can be further logically linked (in the example as Set condition) or can be assigned to another operand (for example, a memory bit) as status. In the following cycle, the instruction then once again supplies '0' as the result even if "S_OperationON" still is status '1'.

The instruction compares the current status of the operand "S_OperationON" with its status in the previous program cycle. This status is stored in a so-called edge memory bit (in the example "M_PosEdge"). It must be ensured that the status of this edge memory bit is not overwritten elsewhere in the program. For each edge evaluation, a separate edge memory bit must be used accordingly, even then when the same operand (in the example "S_OperationON") is evaluated again, for example, in another block!

As well, the edge is assigned to a pulse memory bit (M_1CycleTRUE) in the example. This has the advantage that the edge of the operand is only scanned once in the program. If, in the further course of the program, the edge of the operand must once again be evaluated, no further edge memory bit is required but rather the status of the pulse memory bit can be scanned.

8.5.1.1. RLO Edge Evaluation



RLO Edge Evaluation (P_TRIG, N_TRIG)

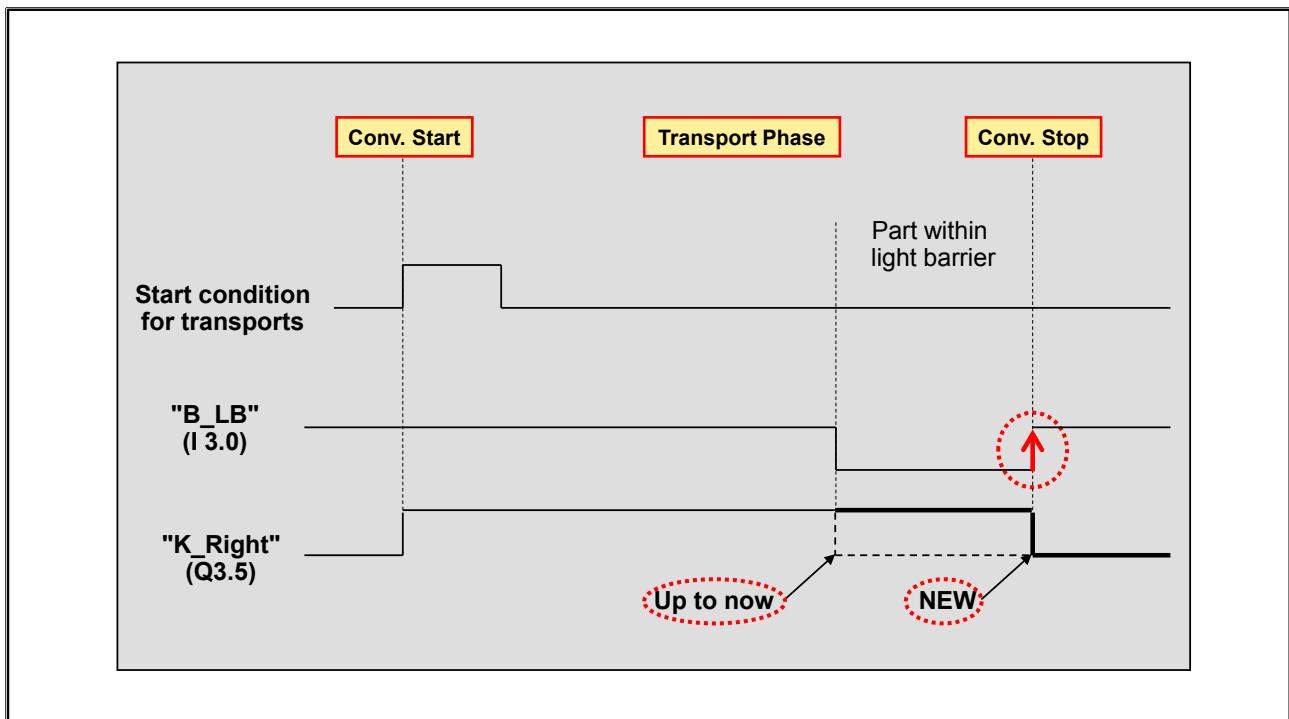
With an RLO edge evaluation, it is possible to detect whether the status of an individual operand or the result of a logic operation has changed from '0' to '1' (rising or positive edge) or from '1' to '0' (falling or negative edge). If this is the case, both edge evaluations supply the result RLO '1' to their output for the duration of one cycle. In the following cycle, the instructions then once again supply RLO '0' as the result even if the status or the RLO of the operand or the logic operation has not changed.

The instructions compare the current status of the operand or the result of a logic operation with its status in the previous program cycle which is stored in a so-called edge memory bit for this (in the example, "M_EdgePos" or "M_EdgeNeg"). It must be ensured that the status of this edge memory bit is not overwritten elsewhere in the program. For each edge evaluation, a separate edge memory bit must be used accordingly, even then when the same operand is evaluated again, for example, in another block!

About the examples in the picture:

- For the instructions P_TRIG and N_TRIG, the result, that is, the RLO of the edge evaluation is available on the right at the output. It can be further logically linked or assigned to an operand (in the example, it is assigned to the operand "M_ImpPos" or "M_ImpNeg").

8.5.2. Exercise 3: Integrating an Edge Evaluation in "FC_Conveyor"



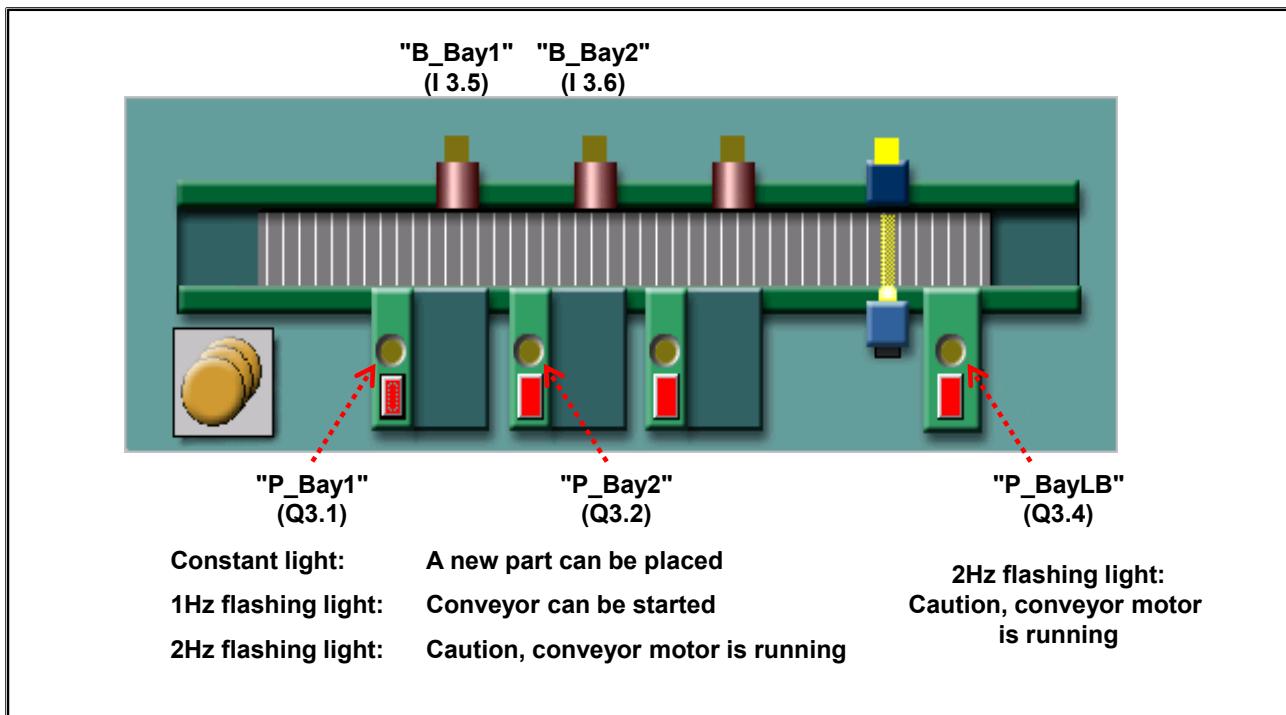
Task:

When "P_Operation" (Q0.1 = '1') is switched on, the parts are to be transported from Bay 1 or 2 THROUGH the light barrier.

What to Do:

1. Program the necessary changes in "FC_Conveyor", by now linking the result of the edge evaluation as the reset condition of the memory byte "M_AutoRight" (memory byte for conveyor motor RIGHT) instead of the light barrier signal "B_LB" (I 3.0) itself. For the necessary edge evaluation of the light barrier signal use the memory byte "M_AuxLB" (M16.0) as an edge memory bit.
2. Download the modified "FC_Conveyor" block into the CPU and check the program function.

8.6. Task Description: Controlling the Indicator Lights, Commissioning "FC_Signal"



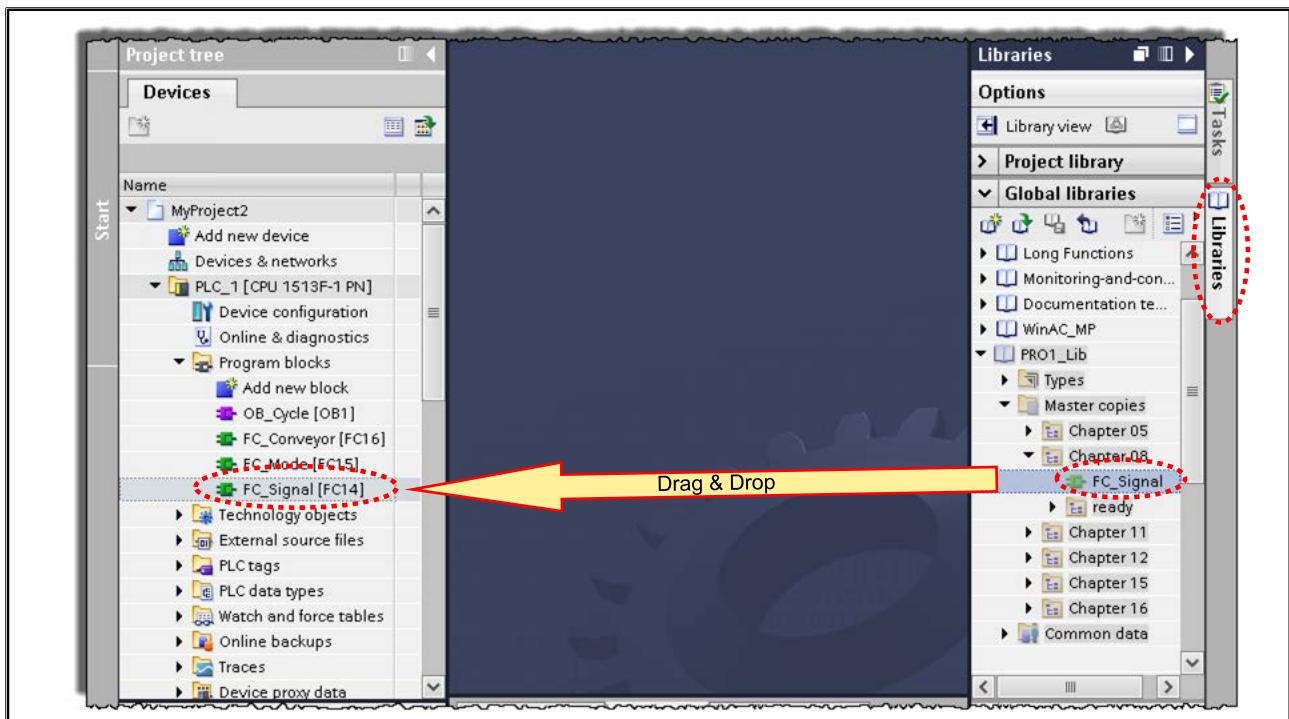
Task Description

When "P_Operation" (Q0.1) is switched on, the indicator lights "P_Bay1" (Q3.1) and "P_Bay2" (Q3.2) are to be controlled as follows:

- They show
 - a constant light when a new part may be placed on the conveyor (distribution conveyor is stopped and both proximity sensors are free)
 - a 1Hz flashing light at the bay where a part is detected by the associated proximity sensor, however, only as long as the conveyor has not yet been started (if parts are placed on the conveyor at both proximity sensors, neither indicator light must light up)
 - a 2Hz flashing light as long as the distribution conveyor is running
- The indicator light at the light barrier bay "P_BayLB" (Q3.4) shows a 2Hz flashing light as long as the conveyor motor is running.

The described functions are already partially programmed in the "FC_Signal" block which is stored in the "PRO1_Lib" global library. The block is to be commissioned and completed by you in the next exercise.

8.6.1. Exercise 4: Writing/Correcting "FC_Signal" and Commissioning It



Task

You are to either program the "FC_Signal" block yourself or you are to copy it from the "PRO1_Lib" global library into the project, and then commission and complete it.

What to Do

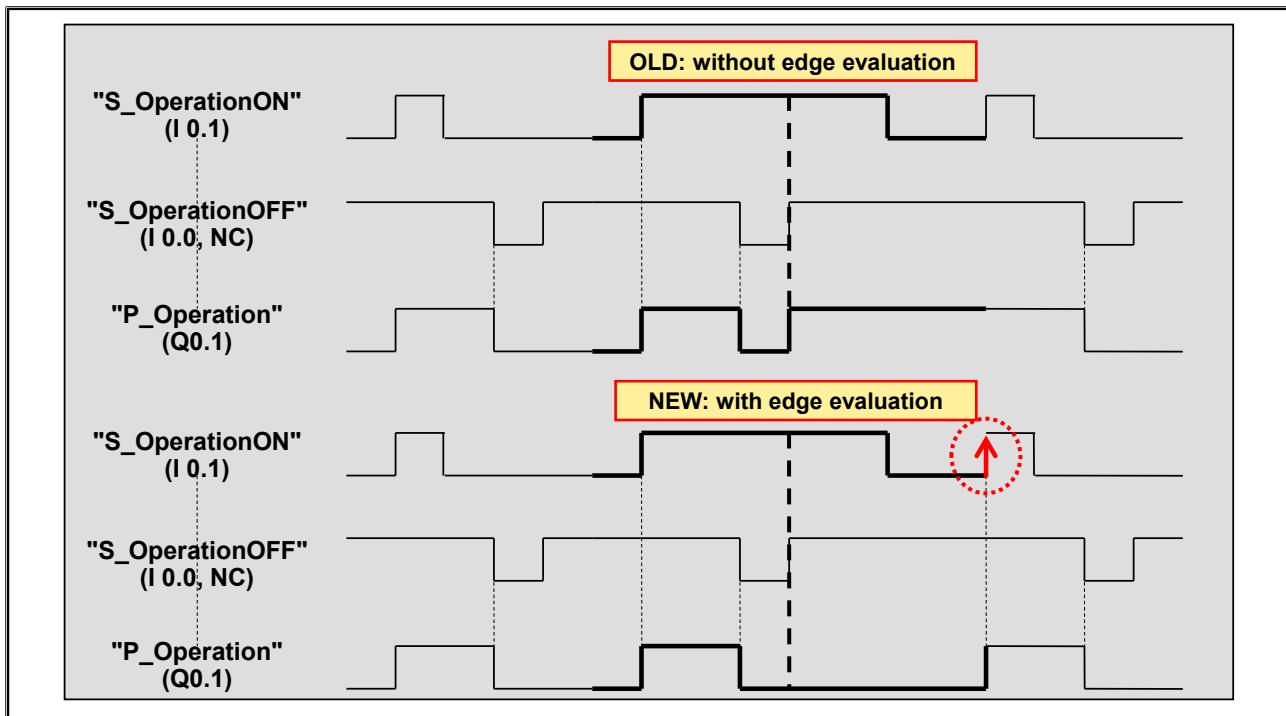
Create the new "FC_Signal" block and program the required functions or:

1. Using drag & drop, copy the "FC_Signal" block into the "Program blocks" folder from the "PRO1_Lib" global library (as shown in the picture).
2. Complete the block so that the indicator lights "P_Bay1" (Q3.1) and "P_Bay2" (Q3.2) show the required signals. (The indicator light "P_BayLB" is already programmed.)
3. Program the call of "FC_Signal" in "OB_Cycle".
4. Download all modified blocks into the CPU and test the program function.
5. Save your project.

Note:

The completely programmed blocks can be found in the library in the folder "Chapter 08 > ready".

8.7. Additional Exercise: Optimizing "FC_Mode"



Function Up to Now of "FC_Mode"

"P_Operation" (Q0.1) is switched on with the simulator switch "S_OperationON" (I 0.1) and switched off with the simulator switch "S_OperationOFF" (I 0.0, NC). If you activate both switches simultaneously, the operation remains switched off or is switched off if currently on. If, however, both switches are activated and the OFF switch is let go while the ON switch is activated, the operation switches back on without the ON switch having to be activated again (see picture, upper function diagram "OLD: without edge evaluation").

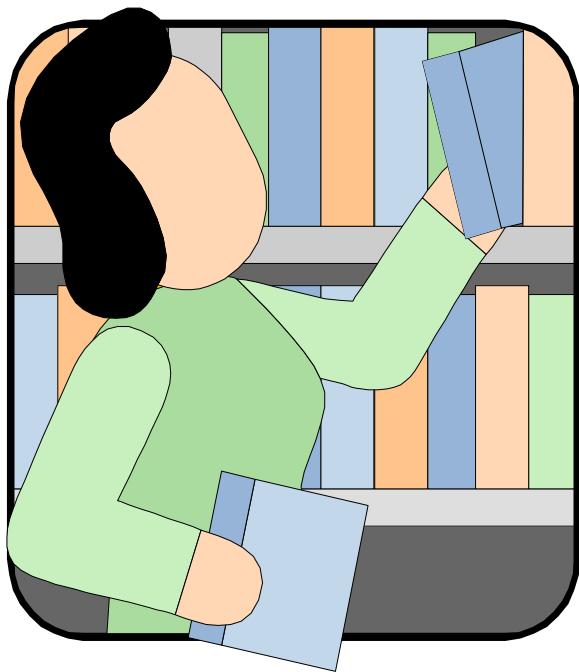
Task:

You are to expand the functionality of "FC_Mode" using edge evaluation so that the ON switch must be activated every time the operation is switched on (see picture, lower function diagram "NEW: with edge evaluation"). The criteria for switching on the system is no longer to be the activated ON switch or its "1" signal, but the function of activating or the "positive edge" of the ON switch signal.

What to Do:

1. In the set condition for "P_Operation" (Q0.1), insert an edge evaluation of the switch "S_OperationON" (I 0.1). For the edge evaluation, use the memory byte "M_AuxOpON" (M15.1) as edge memory bit.
2. Download the modified "FC_Mode" into the CPU and check whether it fulfills the desired functions!

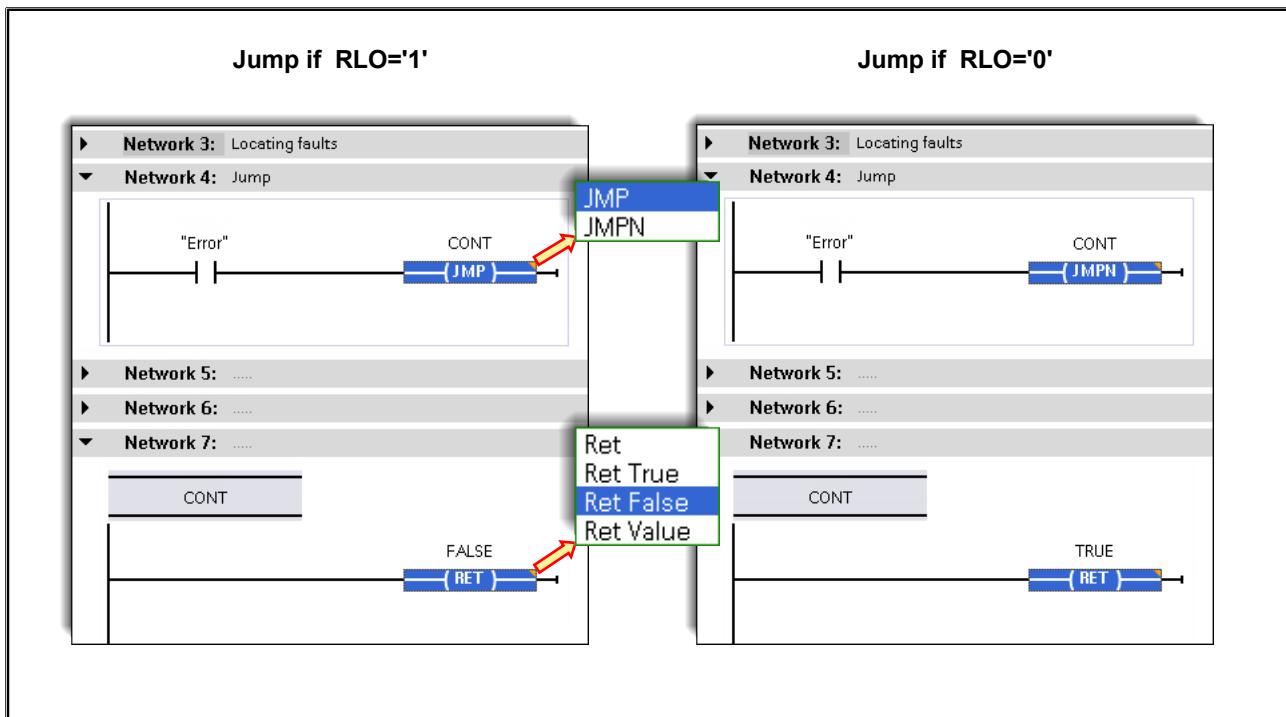
8.8. Additional Information



Note

The following pages contain either further information or are for reference to complete a topic.
For more in-depth study we offer advanced courses and self-learning mediums.

8.8.1. Jump Instructions JMP, JMPN, RET



Jump Instructions JMP and JMPN

With the jump instructions JMP and JMPN, the linear execution of the program can be interrupted within a block and continued in another network. With the jump instruction, a Label is specified which also identifies the target network. The specified label must be located in the same block and be unique. Each label can be jumped to from several locations. The jump can take place in networks with higher (forwards) or lower numbers (backwards).

- **JMP:**
If RLO = '1', the jump into the target network is executed; if RLO = '0', the jump is not executed and the linear program execution continues.
- **JMPN:**
If RLO = '0', the jump into the target network is executed; if RLO = '1', the jump is not executed and the linear program execution continues.

End Block Execution RET

With the instruction RET the program execution of the entire block is ended. If the input of the instruction is assigned then this is a "Conditional block end" and only leads to a block abort when the scan is fulfilled.

Each block has an "ENO" enable output through which the information as to whether the block was executed without error can be communicated to the calling block. With the help of the RET function, the "ENO" enable output can be written.

- **Ret** (RLO = the result of logic operation. The calling block is supplied the signal status "1" since the RET instruction, as conditional instruction, is only executed when the condition is TRUE.)
- **Ret True or Ret False** (The respective value of the constants, TRUE or FALSE, is supplied to the calling program block.)
- **Ret Value** (The value of the Boolean variable <Operand>, which is specified above the instruction, is supplied to the calling program block.)

9

Contents

9.	Functions and Function Blocks	9-2
9.1.	Task Description: Fault Evaluation with Parameter-assignable Blocks.....	9-3
9.2.	Structured Programming.....	9-4
9.2.1.	Modular and Re-usable Blocks	9-5
9.2.2.	Local and Global Operands	9-6
9.3.	Solution with Parameter-assignable Block	9-7
9.3.1.	Declaration of Formal Parameters	9-8
9.3.2.	Editing a Parameter-assignable Block.....	9-9
9.4.	Local, Temporary Variables	9-10
9.4.1.	Local Data Stack.....	9-11
9.5.	Calling a Parameter-assignable Block.....	9-12
9.6.	Task Description: Fault Evaluation by means of a Function (FC)	9-13
9.6.1.	Fault Evaluation	9-14
9.6.2.	Exercise 1: Creating the "FC_FaultEvaluation" Function	9-15
9.6.3.	Exercise 2: Calling and Parameterizing "FC_FaultEvaluation"	9-16
9.7.	Task Description: Fault Evaluation by means of a Function Block (FB).....	9-17
9.7.1.	Instantiating Function Blocks	9-18
9.7.2.	FB - Declaration Section	9-19
9.7.3.	Generating Instance Data Blocks	9-20
9.8.	Changing the Block Call.....	9-21
9.9.	Exercise 3: Creating the Function Block "FB_FaultEvaluation"	9-22
9.9.1.	Exercise 4: Calling and Parameterizing "FB_FaultEvaluation"	9-23
9.10.	Adding Block Parameters Later On	9-24
9.10.1.	Removing Block Parameters Later On	9-25
9.10.2.	Manually Updating a Block Call	9-26
9.11.	Additional Information	9-27
9.11.1.	Compiling Individual / All Changed Blocks	9-28
9.11.2.	Global and Local Tags	9-29

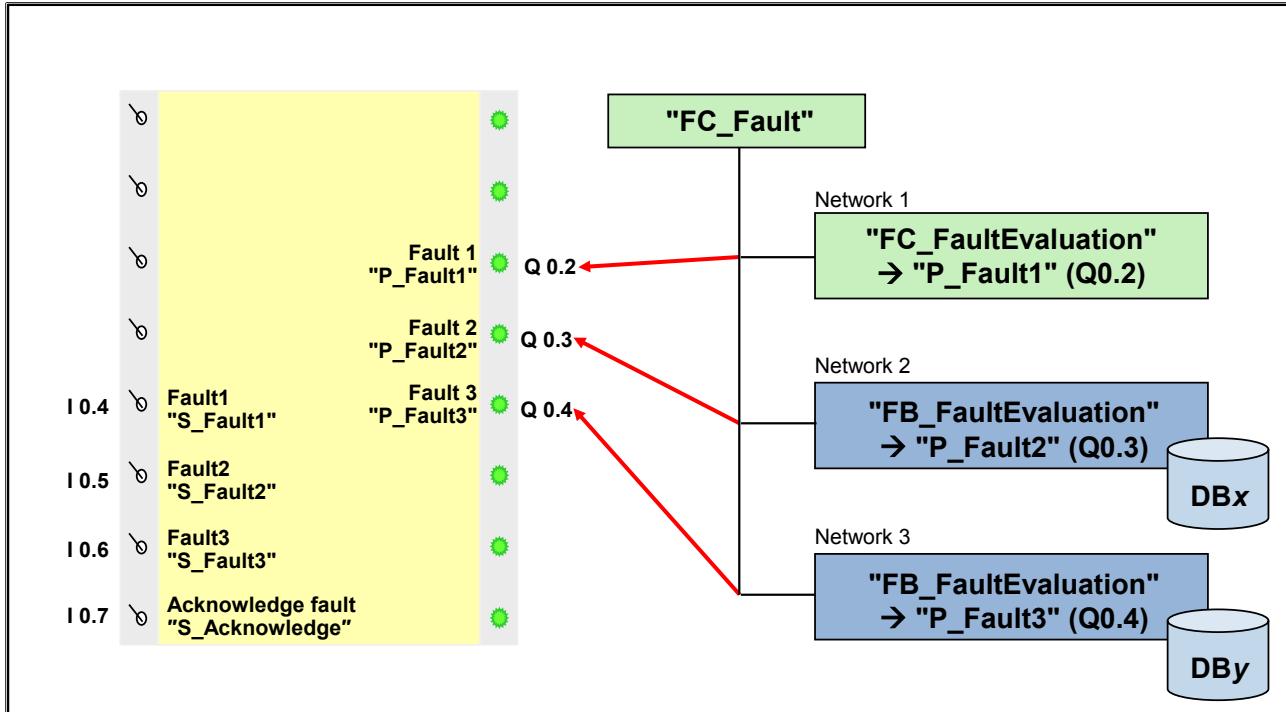
9. Functions and Function Blocks

At the end of the chapter the participant will ...

- ... **be familiar with the purpose of parameter-assignable blocks**
- ... **be familiar with the declaration section of a block**
- ... **be familiar with the purpose of temporary variables**
- ... **be familiar with the purpose of static variables**
- ... **know what a structured programming is**
- ... **be able to program parameter-assignable functions and function blocks and their calls**



9.1. Task Description: Fault Evaluation with Parameter-assignable Blocks



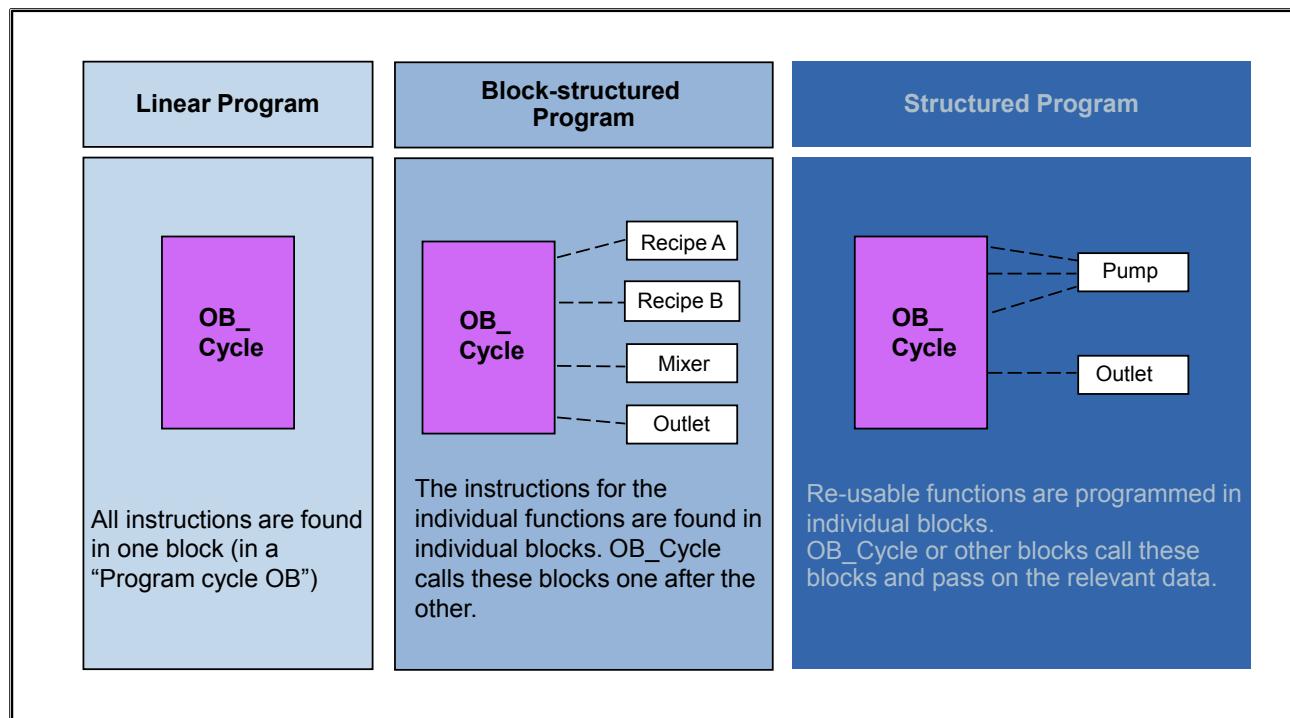
Task Description

Independent of the conveyor model functions so far, 3 different faults are to be evaluated as follows.

If a fault is triggered at the simulator inputs "S_Fault1" (I 0.4) to "S_Fault3" (I 0.6), the associated simulator LEDs "P_Fault1" (Q0.2) to "P_Fault3" (Q0.4) begin to flash.

A group acknowledgement for all faults takes place using the simulator input "S_Acknowledge" (I 0.7). If the fault still exists after acknowledgement, the LED changes to constant light; if the fault no longer exists, the LED goes dark.

9.2. Structured Programming



Linear Program

The entire program is found in one continuous program block (Program cycle OB) which is automatically called by the system. This model resembles a hard-wired relay control that was replaced by an automation system (programmable logic controller). The CPU processes the individual instructions one after the other.

Block-structured Program

The program is divided into blocks, whereby every block only contains the program for solving a partial task. Further structuring through networks is possible within a block. You can generate network templates for networks of the same type. Normally, a cyclically called Organization block contains instructions which call the other blocks in a defined sequence.

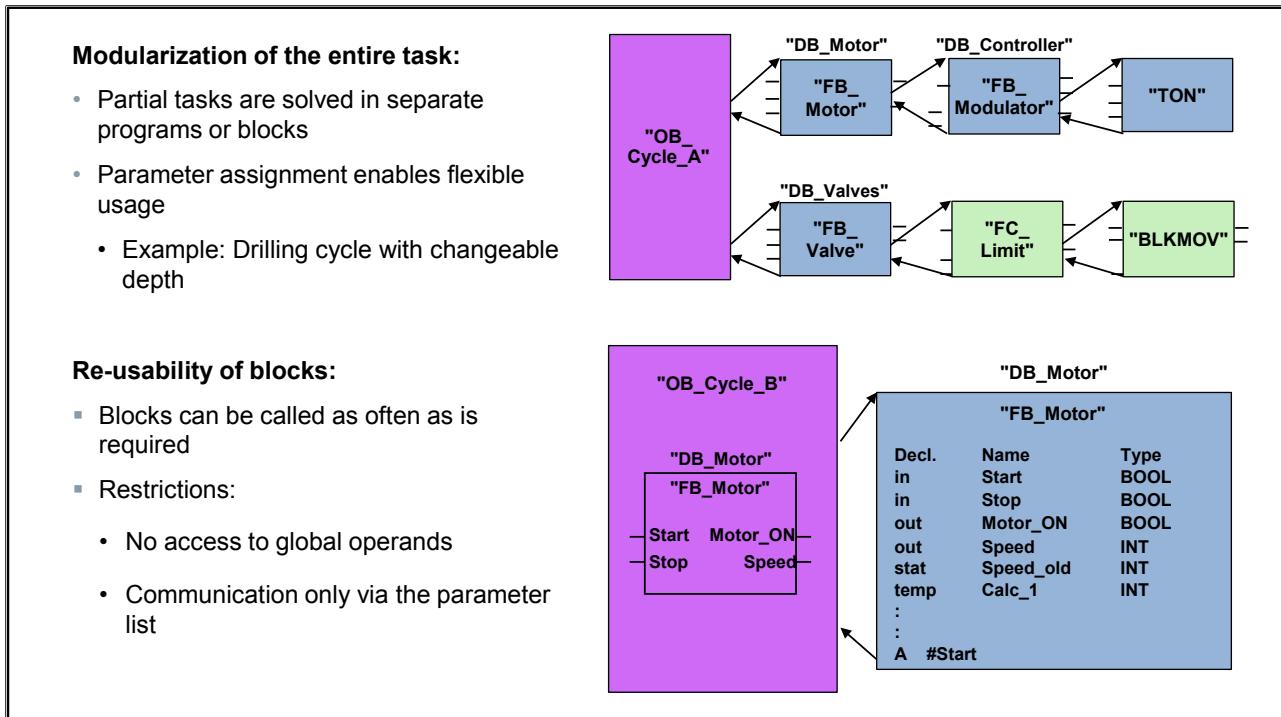
Structured Program

A structured program contains parameter-assignable blocks that are set up in such a way that they can be used universally. When a parameter-assignable block is called, it is passed current parameters (for example, the specific addresses of inputs and outputs as well as parameter values).

Example:

- A "pump block" contains instructions for the control of a pump.
- The program blocks, which are responsible for the control of special pumps, call the "pump block" and provide it with information about which pump is to be controlled with which parameters.

9.2.1. Modular and Re-usable Blocks



Modularization of the Entire Task

Abstraction is the basis for solving complex problems, in which we concentrate on the fundamental aspects of a program in every abstraction level and ignore all the details that are not essential. Abstraction helps us to divide complex tasks into partial tasks which can then be solved on their own.

Parameter-assignable (Re-usable) Blocks

STEP 7 supports this concept of modularization with its block model. The partial tasks that result from the division of the entire task are assigned to blocks in which the necessary algorithms and data for solving the partial problems are stored. STEP 7 blocks such as functions (FC) and function blocks (FB) can be assigned parameters so that the concepts of structured programming can be implemented with them. This means:

- Blocks for solving partial tasks implement their own data management with the help of local variables.
- Blocks communicate with the "outside world", that is, with the sensors and actuators of the process control or with other blocks of the user program, exclusively through their block parameters. No access to global operands such as inputs, outputs, memory bits or variables in DBs can be made from within the statement section of blocks.

Advantages

- The blocks for the partial tasks can be created and tested independent of one another.
- Blocks can be called as often as is required in different locations with different parameter sets, that is, they can be reused.
- "Re-usable" blocks for special tasks can be delivered in pre-designed libraries.

9.2.2. Local and Global Operands

Global Operands (valid in the entire program)	Local Operands (only valid in one block)
<ul style="list-style-type: none"> PII / PIQ I / O peripherals Memory bits Variables in DBs (Chapter Data Blocks) S5-Timers and Counters (not for S7-1200) Constants 	<p>Formal Parameters (Input, Output, InOut)</p> <ul style="list-style-type: none"> interface for data exchange between calling and called block temporary storage in the L-stack for FCs or storage in the IDB for FBs can be used in FCs / FBs <p>Temporary Variables (Temp)</p> <ul style="list-style-type: none"> are overwritten after the block is executed temporary storage in the local data stack (L-stack) can be used in OBs / FCs / FBs <p>Static Variables (Static)</p> <ul style="list-style-type: none"> retain their value after the block is executed permanent storage in instance data block (IDB) can <u>only</u> be declared in FBs <p>Constants (Constant)</p> <ul style="list-style-type: none"> read-only as well as only symbolic access no memory usage can be used in OBs / FCs / FBs

Global Operands

Global operands are valid throughout the entire S7 program. Accordingly, every code (logic) block (OB, FC, FB) can access these operands.

Global operands include inputs, outputs, memory bits, SIMATIC timers, SIMATIC counters, constants and variables which are declared in global data blocks (Chapter: Data Blocks).

Local Operands

Local operands are only valid in the block in which they were declared in the declaration part. Accordingly, only this block can access them.

- Formal Parameters

Formal parameters form the interface between the calling and the called block (FC, FB). They are used to realize a data exchange between the calling and the called block.

- Temporary Variables

Temporary variables can be declared in every code (logic) block (OB, FC, FB) and are managed in the local data stack of the CPU. Accordingly, they only retain their values while the block is being executed. For that reason, it is important that in the current cycle, a write access must have taken place on the temporary variable in the block before a read access can take place. They are, for example, unsuitable as auxiliary variables for edge evaluations or to store quantities. They are, in fact, used to store intermediate results, such as, for complex calculations or format conversions.

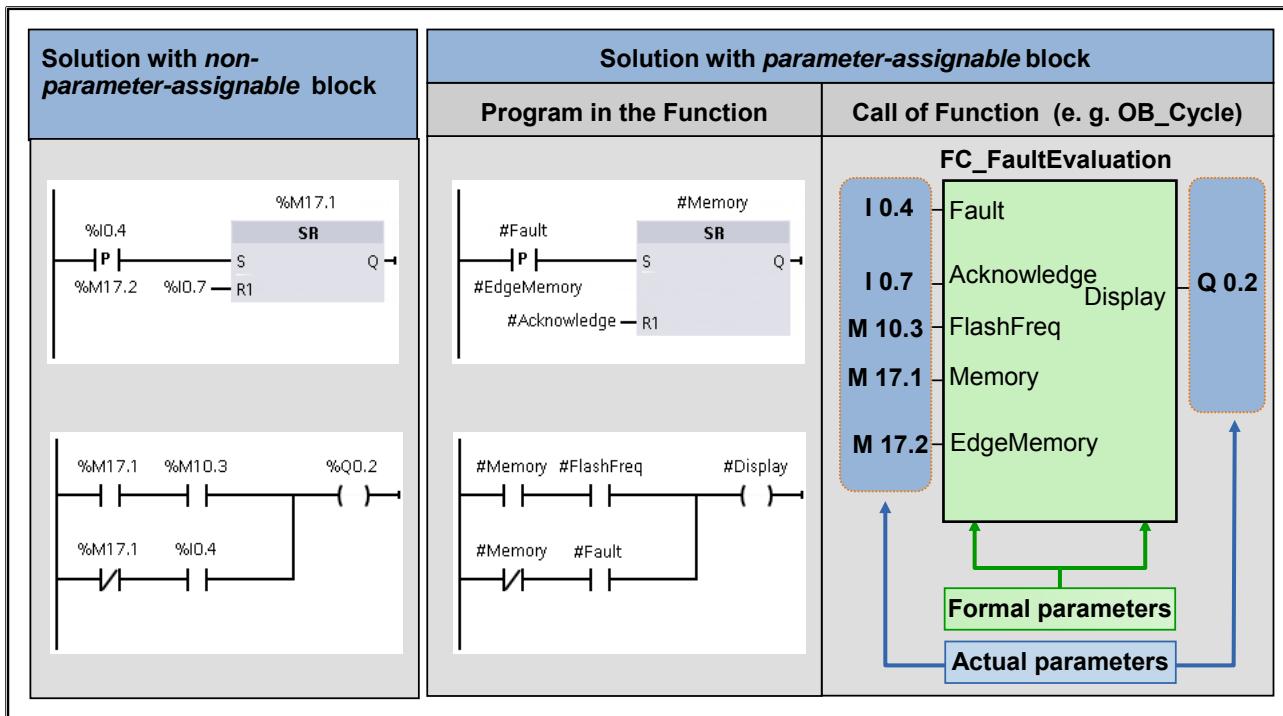
- Static Variables

Static variables can only be declared in FBs and are stored in the associated instance data block. Accordingly, these variables retain their value even after the FB is executed.

- Constants

Constants are fixed values which have a read-only access and which do not take up any memory space.

9.3. Solution with Parameter-assignable Block



Application

You can program parameter-assignable blocks for frequently recurring program functions. This has the following advantages:

- The program only has to be created once, which significantly reduces programming effort.
- The block is only stored in the user memory once, which significantly reduces the amount of memory used.
- The block or the functionality implemented with the block can be called as often as you like, each time with different operands. For this, the formal parameters (input, output, or in/out parameters) are supplied with different actual parameters every time they are called.

Program Execution

When the block is executed, the formal parameters are replaced with the actual parameters passed during the call.

If, as in the example, during the call of the block, the memory byte M17.1 is passed as the actual parameter for the formal parameter `#Memory`, then, at runtime, the memory byte M17.1 is set or reset and its signal status is scanned etc.

Parameter-assignability

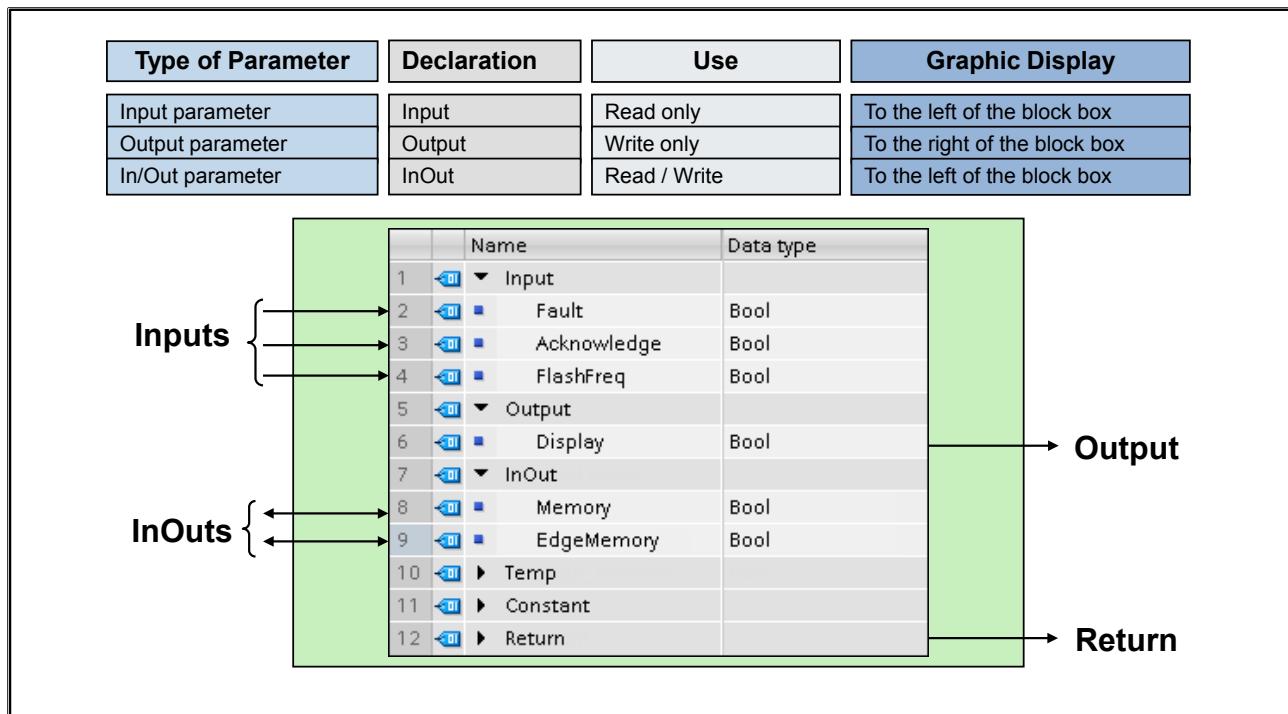
You can program FC or FB blocks as parameter-assignable. You cannot program organization blocks as parameter-assignable since they are called by the operating system and so the call cannot be programmed and also no actual parameters can be specified.

Our Example

Even if the function is required repeatedly in the system, you only have to program "FC_FaultEvaluation" once as parameter-assignable.

"FC_FaultEvaluation" is then called several times for the different fault evaluations and is assigned a different actual operand each time.

9.3.1. Declaration of Formal Parameters



Formal Operands

Before you can create the program in the parameter-assignable block, you have to define the formal parameters in the declaration part.

Type of Parameter

In the table in the picture, you can see the three possible types of parameters and their use. Please note that formal operands that have a reading and a writing access have to be declared as 'In/Out' parameters.

Interface

The **Input**, **Output** and **InOut** parameters as well as the **Return** parameter form the interface of a block. The **Return** parameter is an additional Output parameter and, defined according to IEC 61131-3, the **Return value** of the function. The Return parameter only exists for FCs. If it has the data type VOID, it is not used and also does not appear as a parameter when the function is called.

The variables **Temp** and **Constant** are – even though they are listed in the declaration section of the interface – not components of the block interface, since they do not become visible when the block is called.

Example:

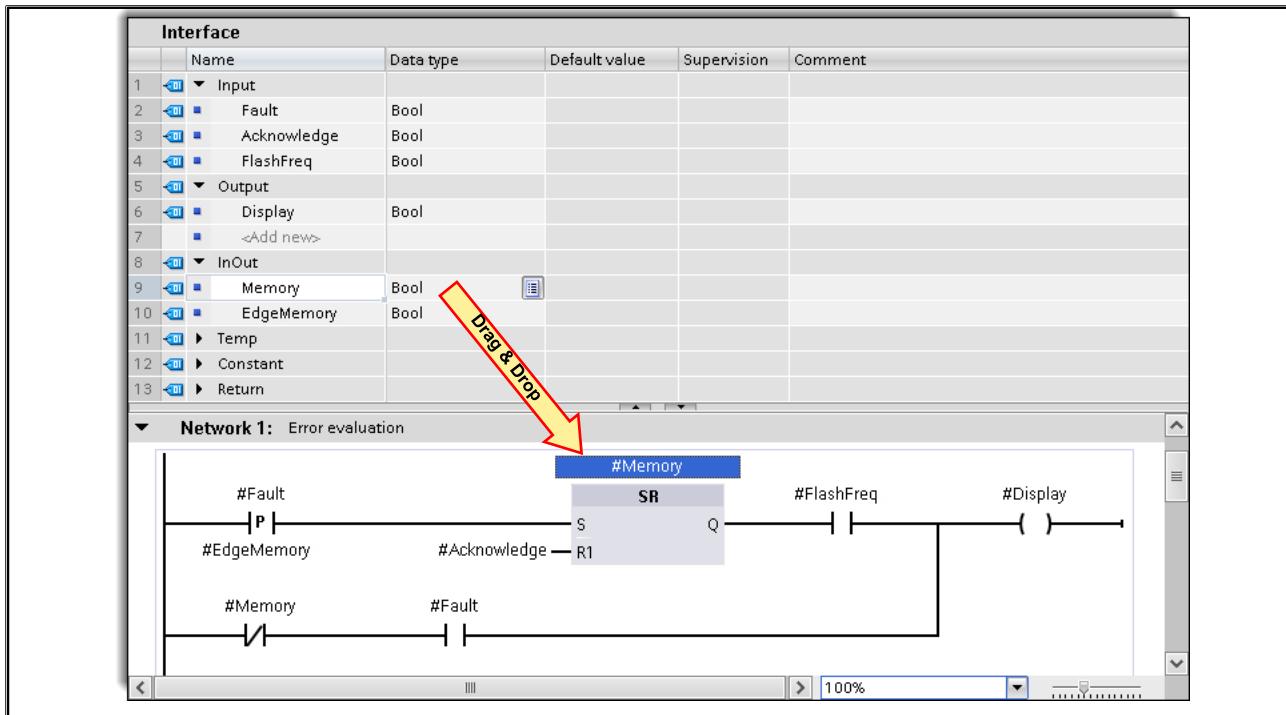
The picture shows the declaration section, that is, the interface of a block. Since the formal parameters #Memory and #EdgeMemory are to be accessed both reading and writing (see next page), they are declared as InOut parameters.



Caution!

The declared formal parameters (Input, Output, InOut and Return) of a block are its interface to the "outside". That is, they are "visible" or relevant to other blocks that call this block. If the interface of a block is changed by deleting or adding formal parameters later on, then the calls of the modified block have to be updated or corrected in all calling blocks.

9.3.2. Editing a Parameter-assignable Block



Notes

It doesn't matter whether the names of the formal parameters are written with capital or small letters. The "#" character in front of the name is automatically inserted by the PG. The character is used to indicate that the parameter is a local operand that was defined in the variable (tag) declaration table of this block.

It is possible, that when you write the program in KOP or FUP, that the name is not completely displayed in one line. This depends on how you have customized the settings in the Program Editor:

Options → Settings → PLC programming → LAD/FBD → Operand field → Maximum width

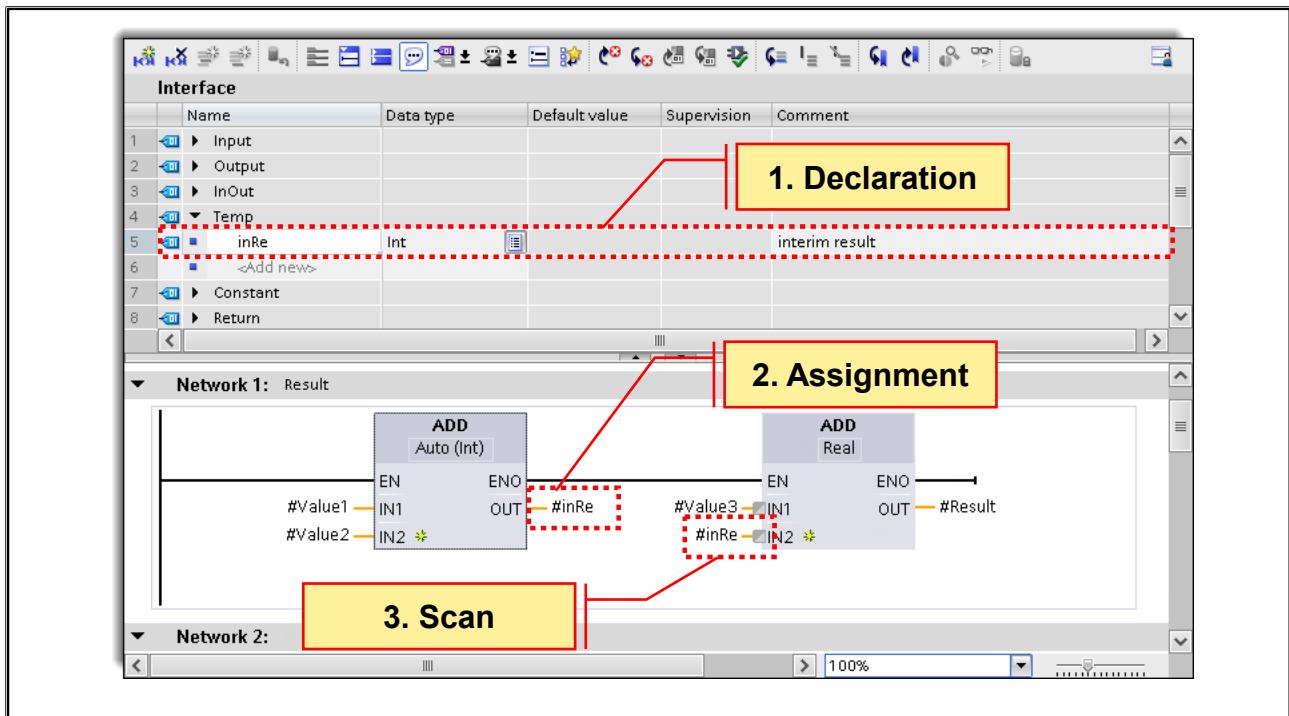
Symbols

- If you use a symbolic name when you edit a block, the Editor first of all searches through the interface of the block. If the symbolic name is there, the symbol with # in front of it is accepted in the program as a local operand.
- If a symbol cannot be found as a local operand, the Editor searches through the PLC tags for the global symbol. If the symbol is found there, the symbol is placed in quotation marks and is accepted in the program as a global operand.
- If you specified the same symbolic name globally (in the PLC tags) as well as locally (in the variable (tag) declaration table) the Editor will always insert the local operand. If, however, you want to work with the global symbol, you must select the relevant operand when you make the entry, place the symbol name in quotation marks or change it later on.

Drag & Drop

Just as with global operands (for example, from the PLC tags) local operands can be dragged into the program part of the Editor from the block interface using drag & drop and placed in the desired position there.

9.4. Local, Temporary Variables



Declaration

The variables are also defined in the declaration part of the block. The name of the variable and the data type must be specified.

Access

With optimized blocks, all temporary variables are initialized with 0 at the beginning of block execution.

With not-optimized blocks, all temporary variables have an undefined value at the beginning of block execution. When working with temporary variables, you must therefore make sure that the variable is first of all assigned a defined value before it is scanned.

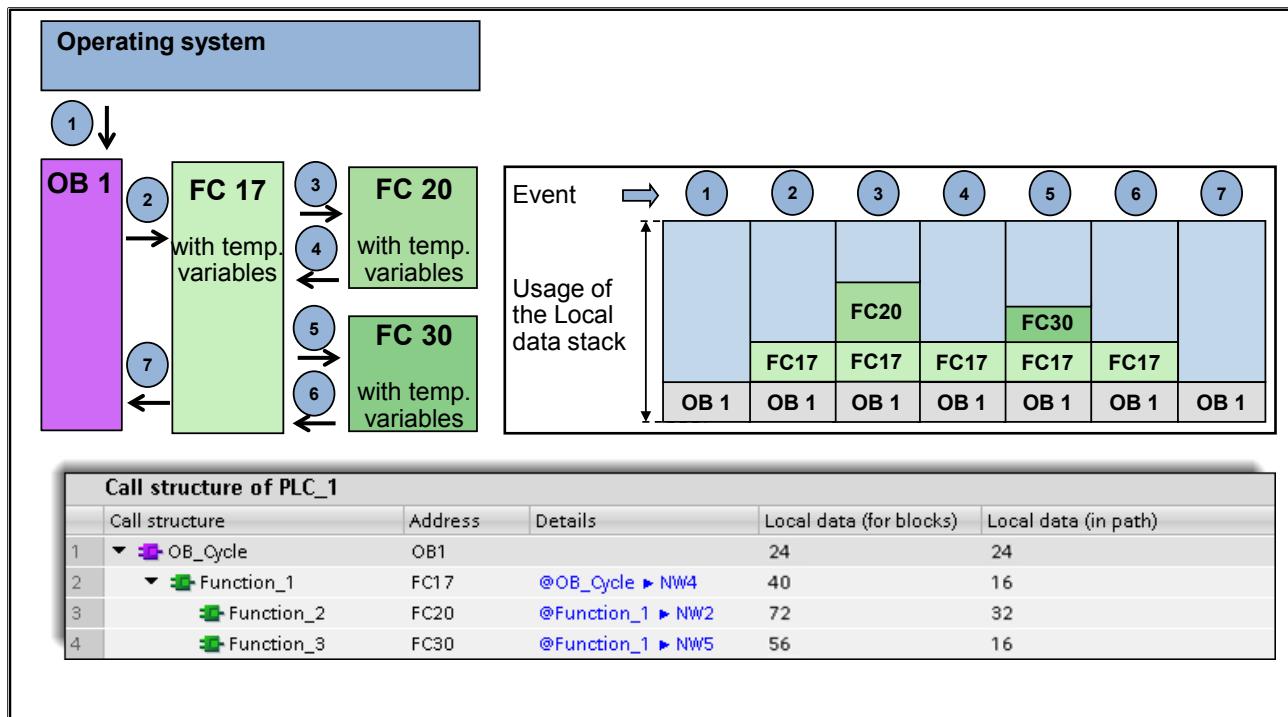
In the example, the result of the Addition is assigned to the temporary variable `#inRe` before it is then scanned during the Multiplication. (The arithmetic operations are dealt with in the chapter "Digital Operations".)

Note

Operands that begin with the # special character are local operands (parameters or local variables) that must be declared in the declaration part of the block. Local operands are only valid and usable in the block in which they were declared.

The Program Editor automatically inserts the # character.

9.4.1. Local Data Stack



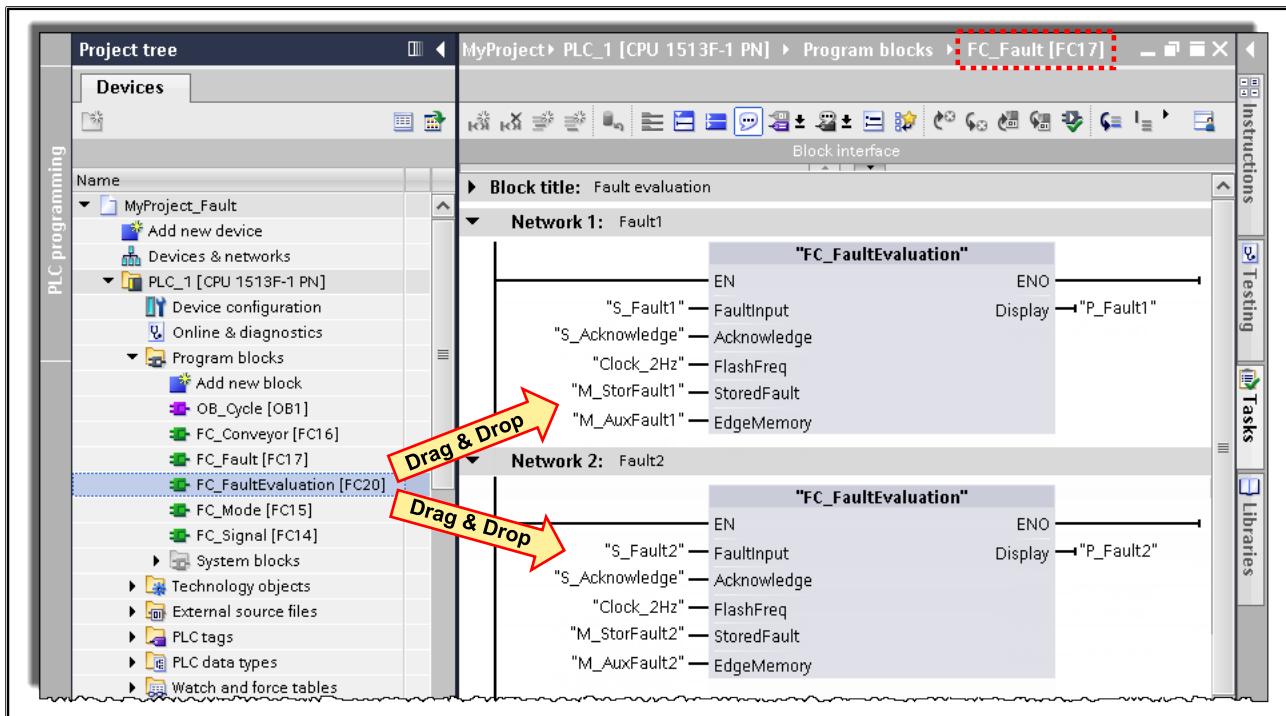
Total Usage of the Local Data Stack (L-Stack)

For every program execution level or priority class (such as, OB 1 with all blocks that are called in it), a separate local data stack is reserved. That is, a segment of defined size is reserved on the L stack of the CPU (allocation or reservation of memory space).

The local variables/operands of OB 1 as well as the local, temporary variables of the blocks (FCs and FBs) called in or by OB 1 are stored in this local data stack.

You can use the "Reference Data" tool to display the "Program Structure" to see to what extent an S7 program puts a burden on the local data stack. (The reference data and where they are displayed is dealt with in the chapter "Troubleshooting".)

9.5. Calling a Parameter-assignable Block



Block Call

A block can be called by dragging it from the "Program blocks" folder & dropping (inserting) it in the statement (code) part of the calling block.

Note

When a parameter-assignable function (FC) is called, an actual parameter must be passed for every formal parameter.

Exception:

In the graphic programming languages LAD and FBD, the assignment of the EN and ENO parameters, which are automatically added by the Editor, is optional.

Parameter Assignment

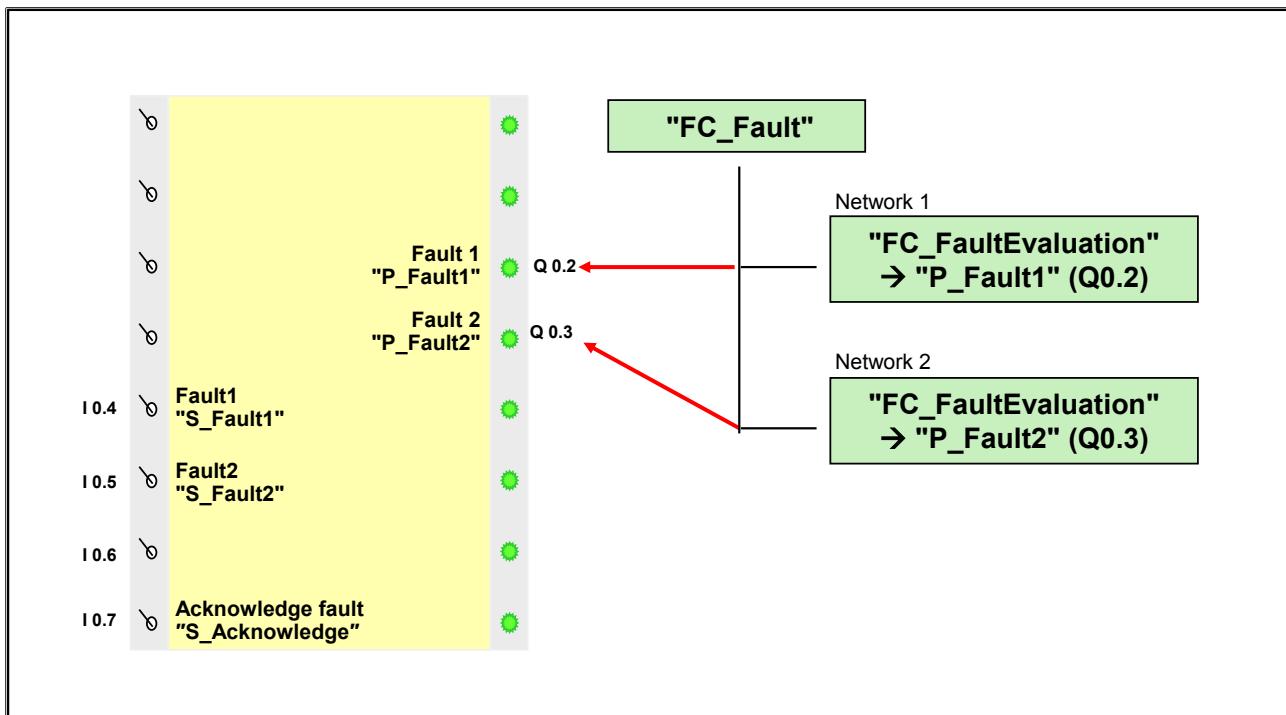
All global and local operands whose data type corresponds to the formal parameters of the called block can be passed as actual parameters.

The actual parameters can be passed with an absolute address or with a symbolic name - as declared in the PLC tags or in the declaration part of the calling block.

Passing On of Parameters

Basically, a "passing on of parameters" is also possible. That is, formal parameters of the calling block are passed on as actual parameters to the called block. For parameters of complex data types (see chapter "Data Blocks") this is however only possible with limitations.

9.6. Task Description: Fault Evaluation by means of a Function (FC)



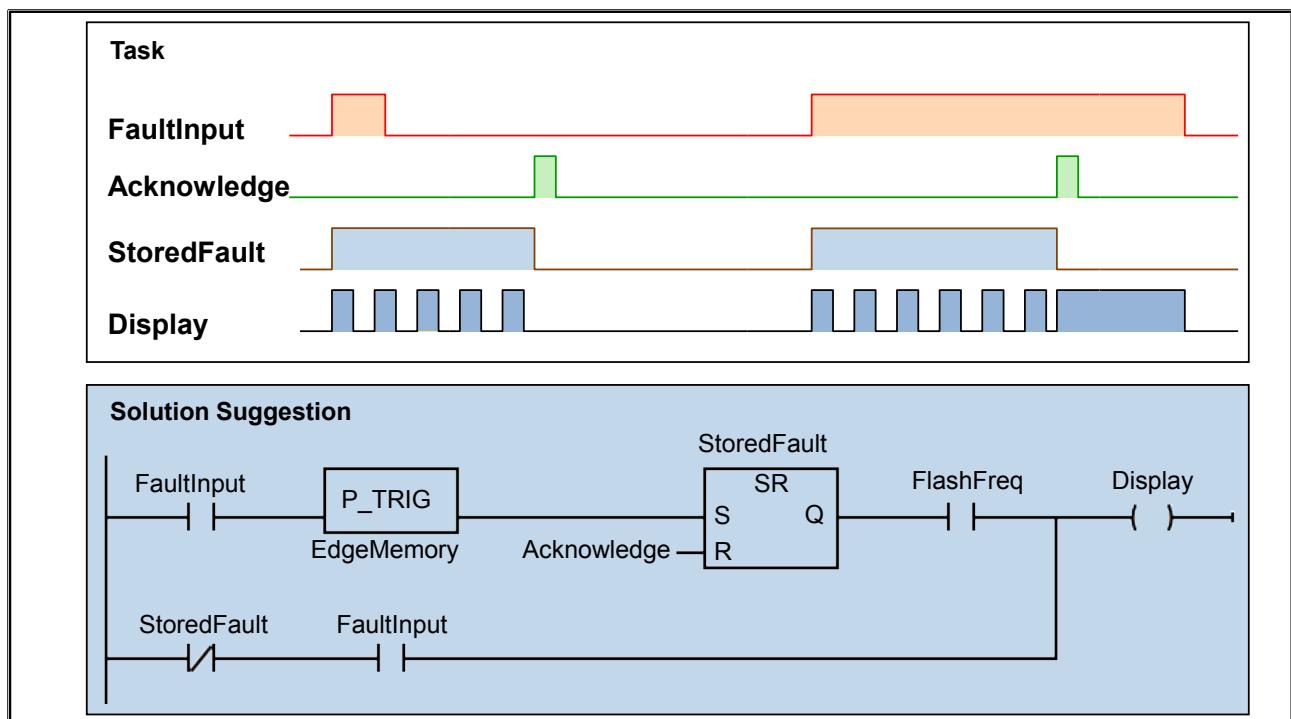
Task Description

If a fault is triggered at the inputs I 0.4 "S_Fault1" or I 0.5 "S_Fault2", the associated LED Q0.2 "P_Fault1" or Q0.3 "P_Fault2" begins to flash.

The input I 0.7 "S_Acknowledge" is a group acknowledgement for all faults. If the fault still exists after acknowledgement, the LED changes to constant light; if the fault no longer exists, the LED goes dark.

First, an "FC_Fault" is to be created. Then, the required function is to be programmed in the parameter-assignable "FC_FaultEvaluation" which is to be called twice in "FC_Fault" for the evaluation of the two faults.

9.6.1. Fault Evaluation



Task

Faults that occur are to be displayed by an indicator light on the operator console. When there is a signal change from 0 → 1 at the input, the output shows a 2Hz flashing light.

After the fault is acknowledged but still exists, the output (light) switches to a constant light. When the fault no longer exists, the light at the output goes dark.

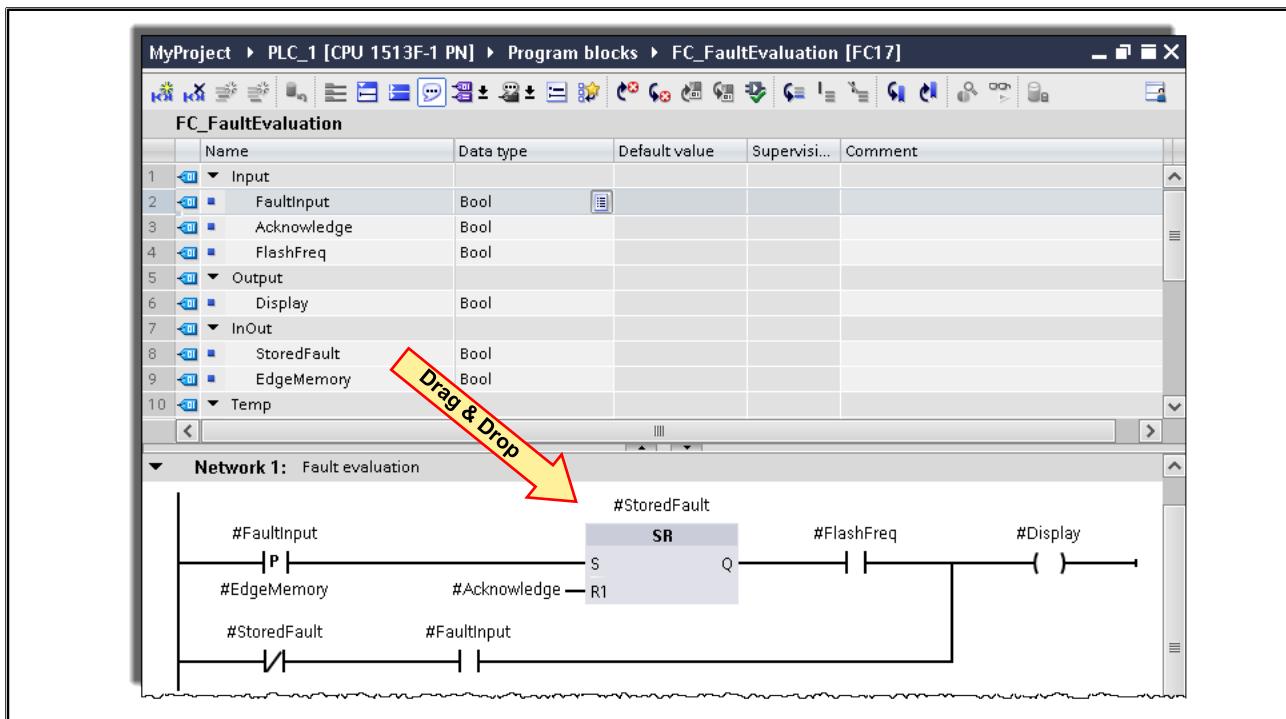
Solution Suggestion

An edge evaluation of the fault (**FaultInput**) is required since the message buffer (**StoredFault**) would otherwise immediately be set again after an acknowledgement (**Acknowledge**) and a still existing fault, thus making the display (**Display**) flash once more.

When an acknowledgement (**Acknowledge**) has not yet occurred, that is, the message buffer (**StoredFault**) still exists, the upper AND logic operation with the linked flash frequency (**FlashFreq**) causes the display (**Display**) to flash.

When acknowledgement has already occurred (**Acknowledge**) and therefore the message buffer (**StoredFault**) no longer exists, but the fault input (**FaultInput**) still exists, the lower AND logic operation causes a constant light at the display (**Display**).

9.6.2. Exercise 1: Creating the "FC_FaultEvaluation" Function



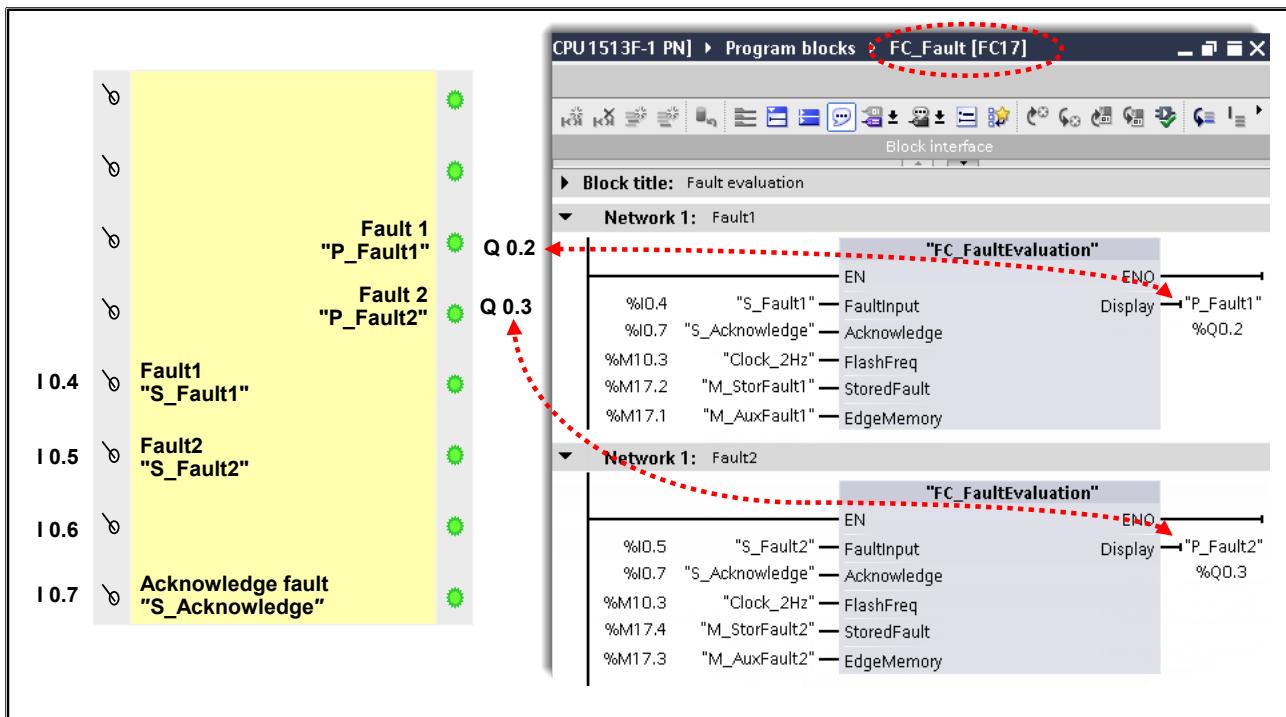
Task

You are to create the program for the fault evaluation in the parameter-assignable "FC_FaultEvaluation".

What to Do

1. Insert the "FC_FaultEvaluation" block in the "Program blocks" folder.
2. Declare the formal parameters as shown in the picture.
3. Create the program as shown in the picture.
4. Save the block.

9.6.3. Exercise 2: Calling and Parameterizing "FC_FaultEvaluation"



Task

You are to create the new block "FC_Fault" which will process the fault handling and fault evaluation in later exercises.

In the new blocks, 2 faults from the process (signals of the two simulator switches) are to be evaluated. For this, the previously programmed "FC_FaultEvaluation" must be called twice.

What to Do

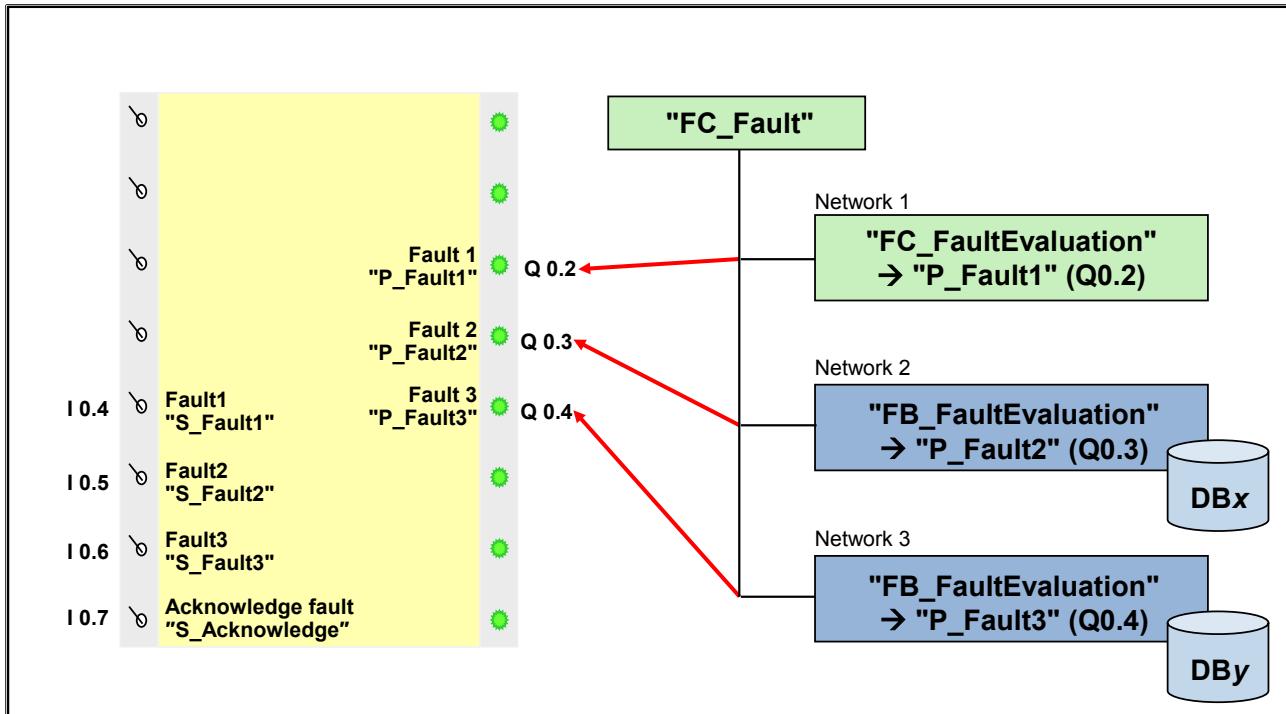
1. Create the new "FC_Fault" block.
2. In "FC_Fault", program the two calls of the previously created "FC_FaultEvaluation" block as shown in the picture.
3. Call "FC_Fault" in "OB_Cycle".
4. Save the change and transfer the program into the CPU.
5. Check your program to see whether it fulfills the described functions for fault evaluation.

Note

The MB 10 memory byte was already parameterized as a clock memory byte in the device configuration.

The "Clock_2Hz" (M10.3) memory bit has a flashing frequency of 2Hz and was already created as a PLC tag.

9.7. Task Description: Fault Evaluation by means of a Function Block (FB)

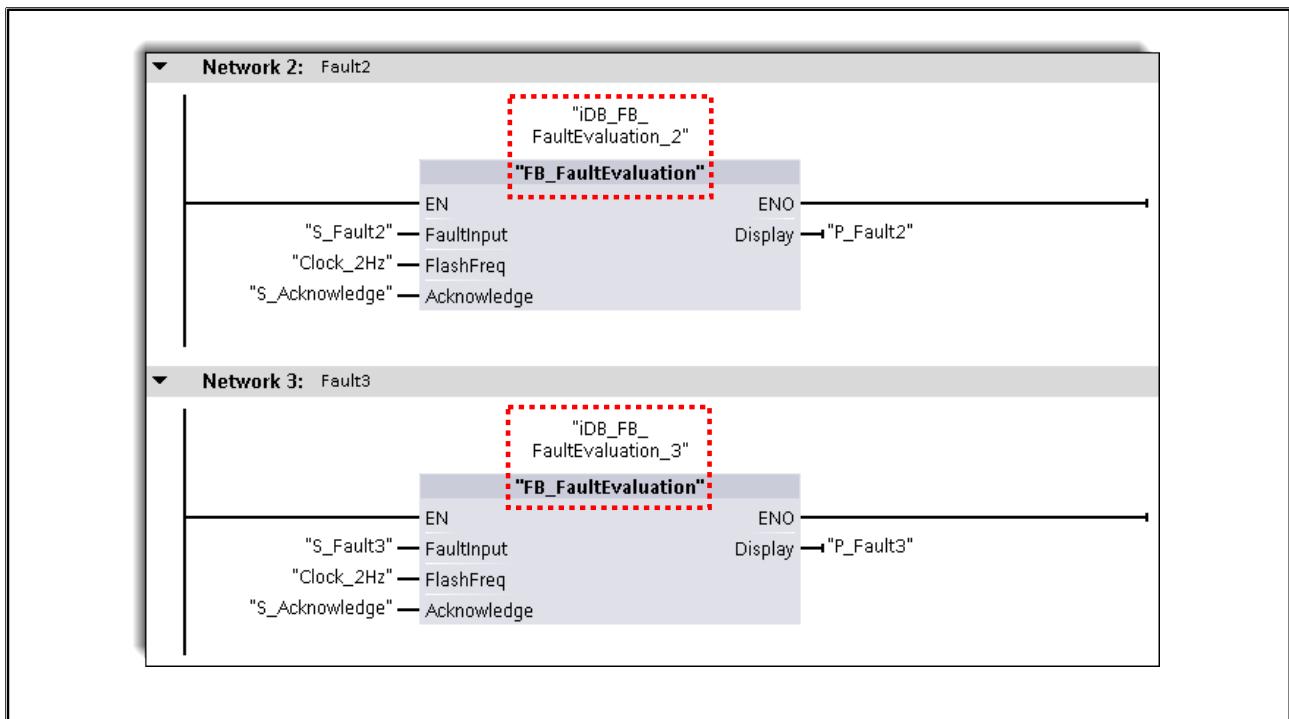


Task Description

The previously described fault evaluation is now to be implemented with an FB instead of an FC. This offers the advantage that for the internally required edge evaluation of the fault and as stored fault, the FB doesn't have to be passed any global operands from outside since local, static variables can be used.

Fault 1 is to continue to be evaluated by the already existing "FC_FaultEvaluation". The evaluation of Fault 2 and 3 is to be carried out by the "FB_FaultEvaluation" which is now to be created.

9.7.1. Instantiating Function Blocks



Special Features

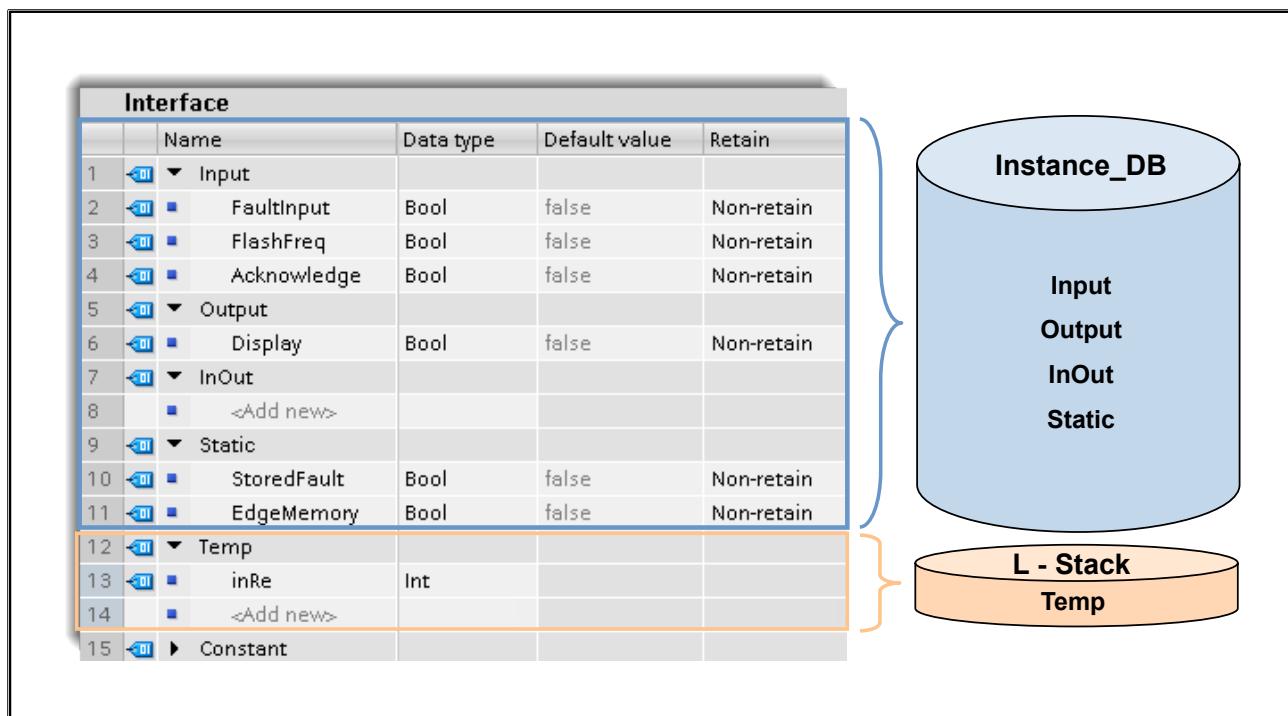
Unlike functions (FCs), function blocks (FBs) have a (recall) memory. That means that a local data block is assigned to the function block. This data block is known as an instance data block. When you call an FB, you must also specify the Instance-DB which is then automatically used as an instance for this FB call.

An instance DB is used to save static variables, among other things. These local variables can only be used in the FB, in whose declaration table they were declared. When the block is exited, they are retained.

FB Advantages

- For the FC programming, the user must search for free memory areas and maintain them himself. The static variables of an FB, on the other hand, are maintained by the STEP 7 software.
- The known danger of memory bit double assignments in FC programming is avoided with the use of static variables.
- Instead of the InOut formal parameters "StoredFault" and "EdgeMemory" of the "FC_FaultEvaluation", static variables are used in the "FB_FaultEvaluation". This makes the block call simpler since the two formal parameters are dropped.

9.7.2. FB - Declaration Section



Parameters

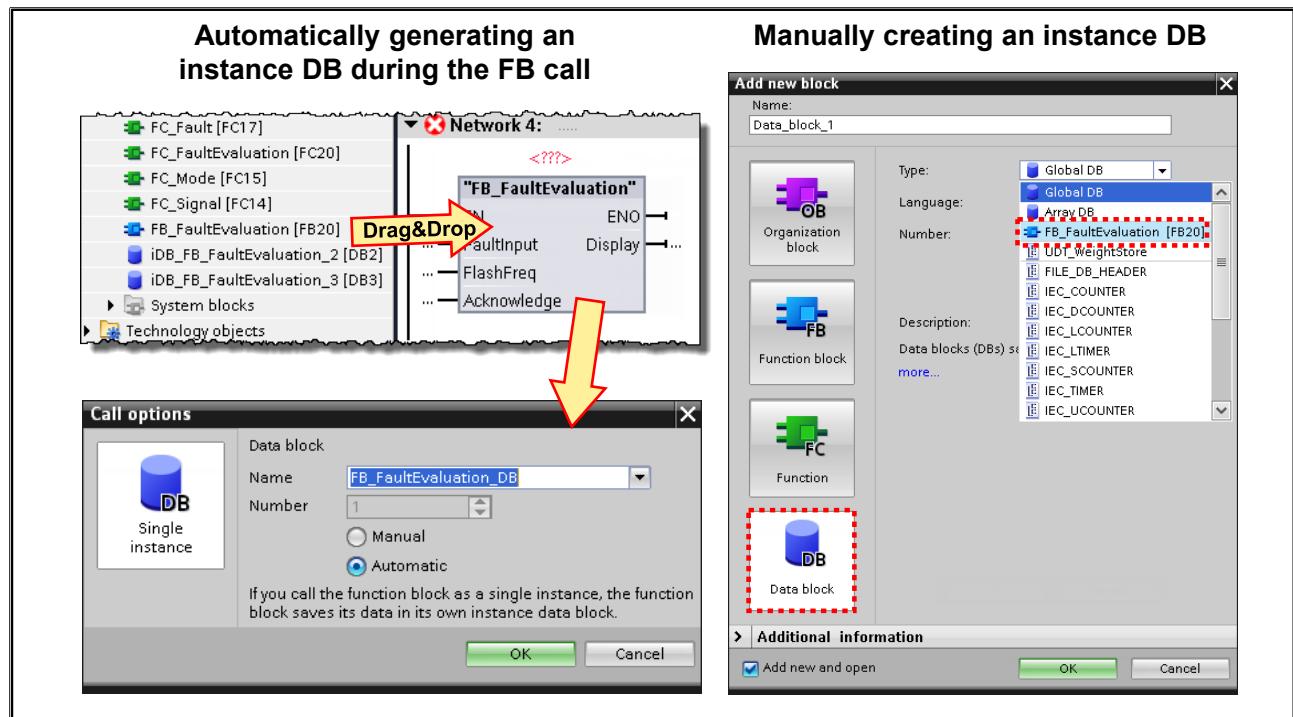
When the function block is called, the values of the actual parameters are stored in the instance data block. If no actual parameter was assigned to one of these formal parameters in a block call, then the last value stored in the instance DB for this parameter is used in the program execution. One exception is InOut parameters whose data types are not elementary. These must be assigned since the values of the actual parameters are not stored in the instance DB but rather the information about the storage location of the actual parameter.

Just as for a function, different actual parameters can be passed for each FB call. When the function block is exited, the data is retained in the instance data block.

Static Variables

Unlike functions, function blocks additionally have "static variables" (Static). These variables form the memory of the FB. They are not stored in the L-Stack but also in their own instance data block.

9.7.3. Generating Instance Data Blocks



Generating

There are two ways of generating an instance data block:

- Create a new block (data block) and select "Function block XY" as Type.
- For an FB call, the user specifies with which instance DB the FB is to work. A dialog automatically opens in which the symbolic name and, if desired, a manual number of the instance DB can be preset.

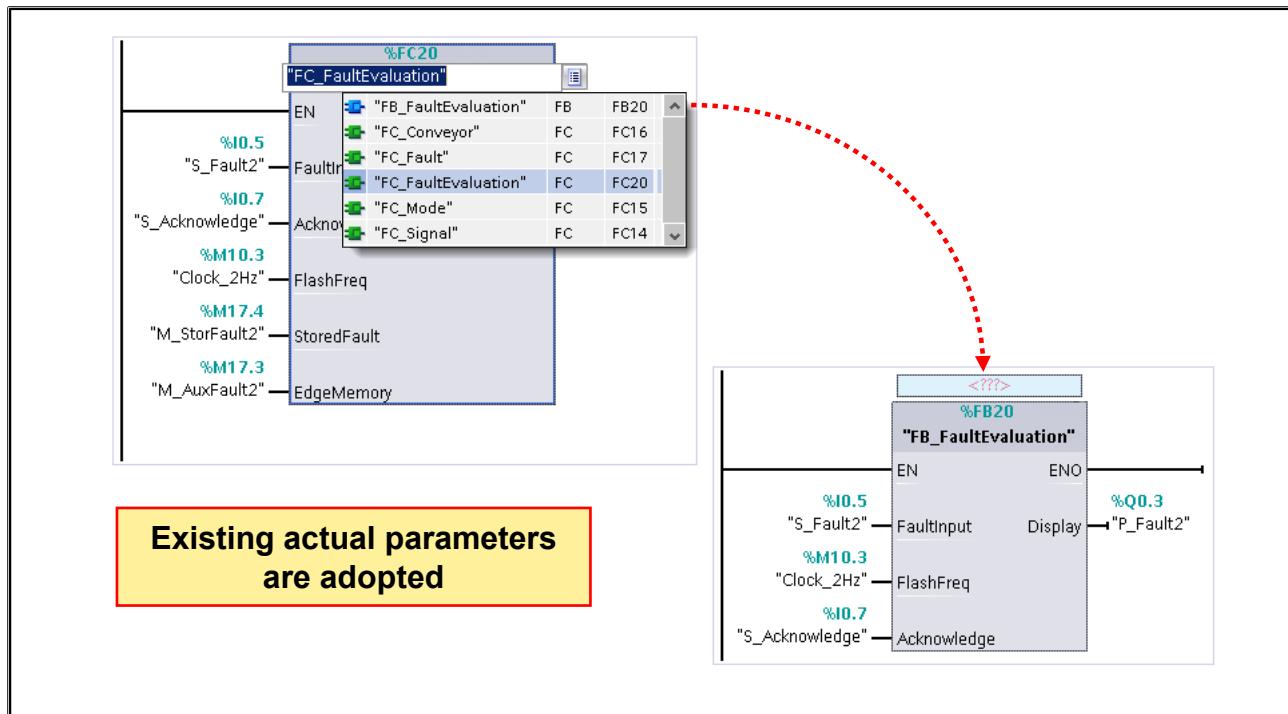
Already existing instance data blocks can also be selected here.



Caution!

If you modify the FB (by adding additional parameters or static variables), you must then also generate the instance DB again.

9.8. Changing the Block Call

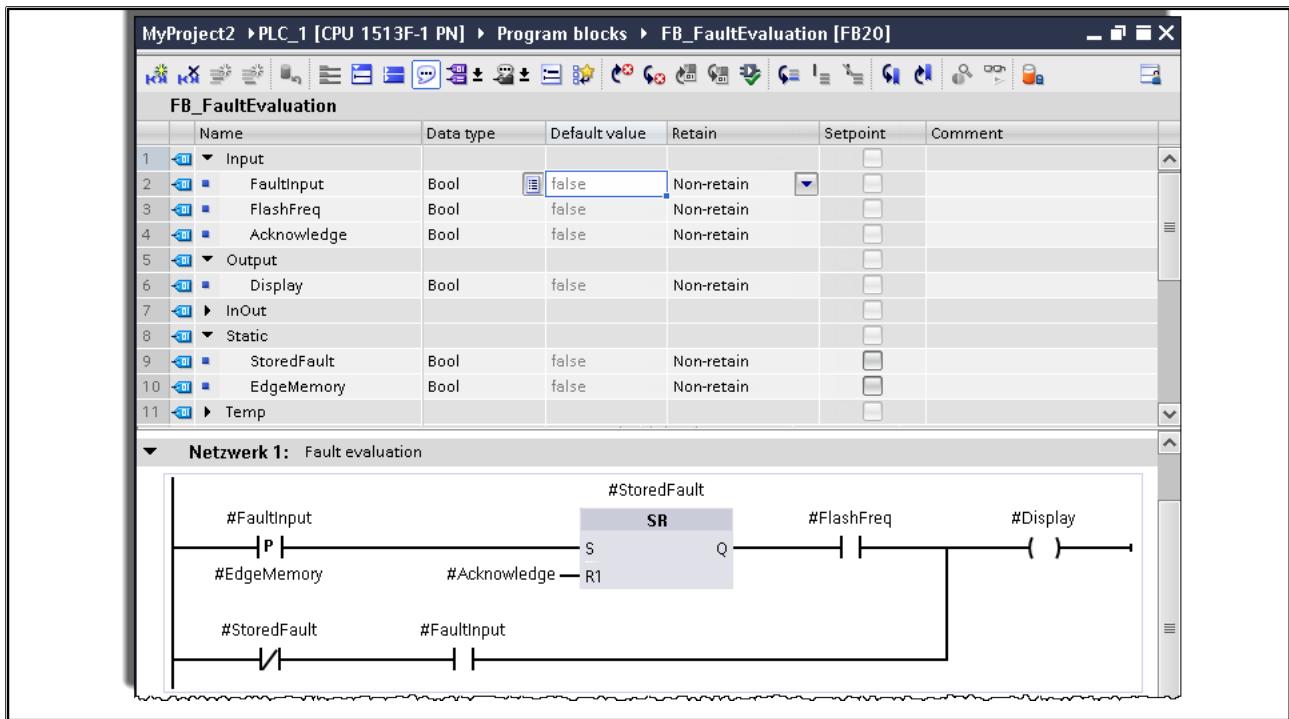


In order to replace the call of a block with another block call, a selection list of all FCs and FBs can be opened at the calling point by double-clicking on the name of the already called block.

Advantage:

If both blocks have the same formal parameters, then they retain their actual parameters and all formal parameters do not have to be supplied with new actual parameters.

9.9. Exercise 3: Creating the Function Block "FB_FaultEvaluation"



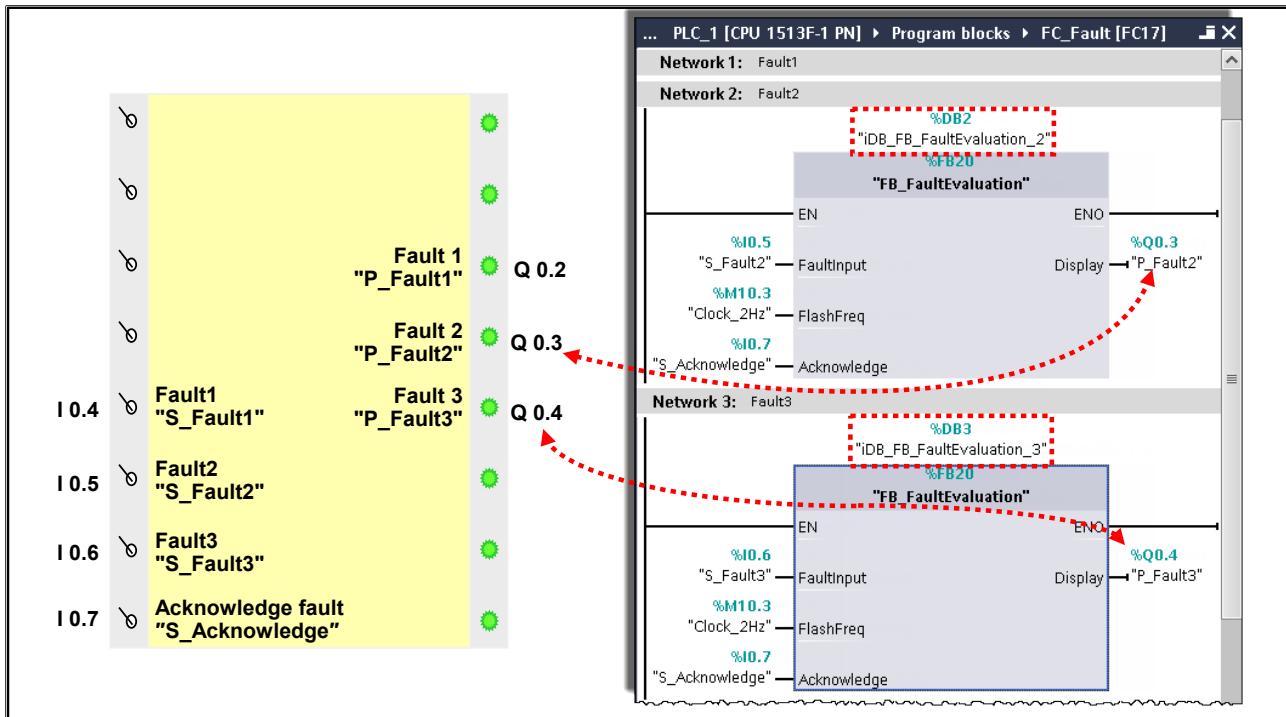
Task

You are to create the new "FB_FaultEvaluation" block for the subsequent evaluation of Fault 2 and 3.

What to Do

1. Insert the new "FB_FaultEvaluation" block.
2. Declare the formal parameters and the static variables of the block as shown in the picture. For this, you can copy the required variables from the already programmed "FC_FaultEvaluation".
3. Program "FB_FaultEvaluation". For this, you can copy the required program parts from the already programmed "FC_FaultEvaluation".
4. Save the block and download it into the CPU.

9.9.1. Exercise 4: Calling and Parameterizing "FB_FaultEvaluation"



Task

The evaluation of the old Fault 2 (programmed up until now through the call of "FC_FaultEvaluation") and the evaluation of the new Fault 3 is to be implemented with the newly created "FB_FaultEvaluation".

For this, the parameter-assignable block "FB_FaultEvaluation" must be called twice in "FC_Fault", each time with a different instance data block.

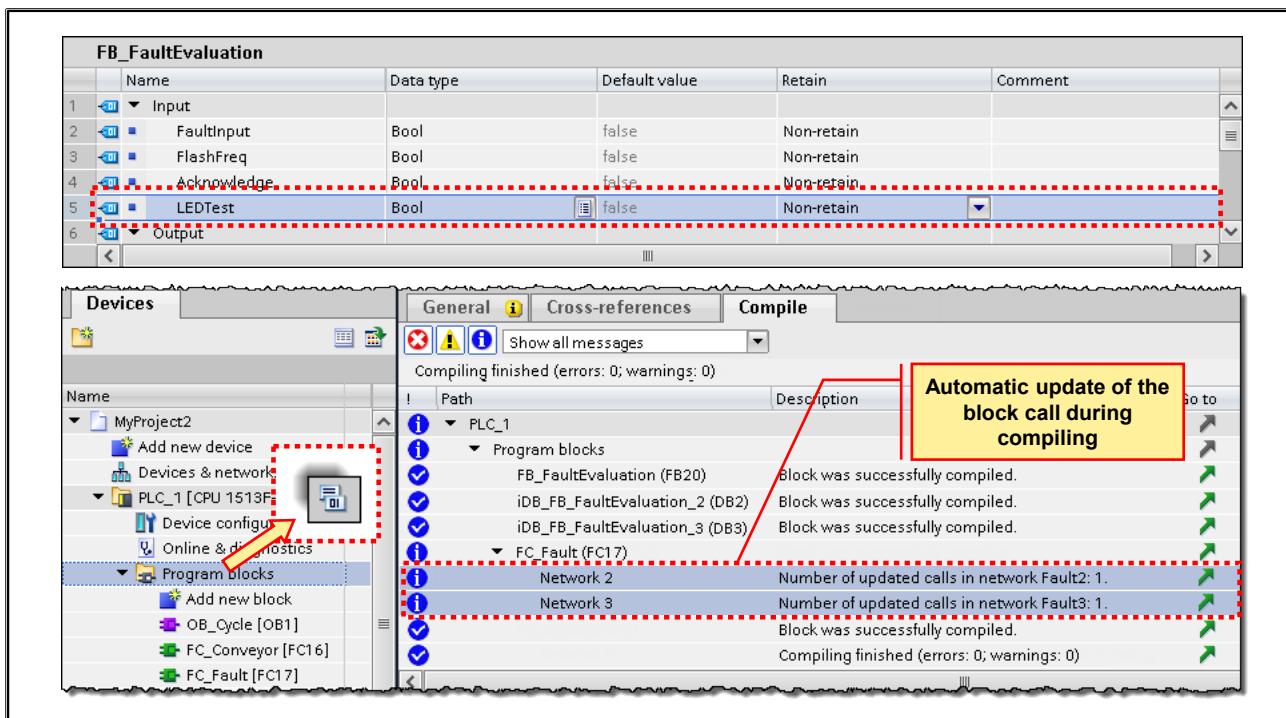
What to Do

1. In "FC_Fault", replace the second call of "FC_FaultEvaluation" with the call of "FB_FaultEvaluation".
2. Generate the instance data block "iDB_FB_FaultEvaluation_2" and specify it as the instance for the programmed call of "FB_FaultEvaluation".

You can generate a new instance via the context menu item of the call (right-click on the Block call > "Create instance") or you create a new block (data block of the type "FB_FaultEvaluation") and insert it using drag & drop.

3. Program the second call of "FB_FaultEvaluation" - as shown in the picture - in a new network and let the Editor generate the instance "iDB_FB_FaultEvaluation_3".
4. Save the modified "FC_Fault".
5. Download the entire program into the CPU and check the program function.

9.10. Adding Block Parameters Later On



Problem

If you have to adjust or supplement the interfaces or the code of individual blocks during or after program creation, it can lead to time stamp conflicts. Time stamp conflicts can, in turn, lead to block inconsistencies between calling and called blocks or reference blocks and thus to a high degree of correction effort.

If block parameters are **added** later on to a block already called in the program, you also have to update the calls of the block in other blocks.

- Automatic Update

Time stamp conflicts are also detected when the entire user program is compiled and in case of added parameters, affected block calls are automatically updated.

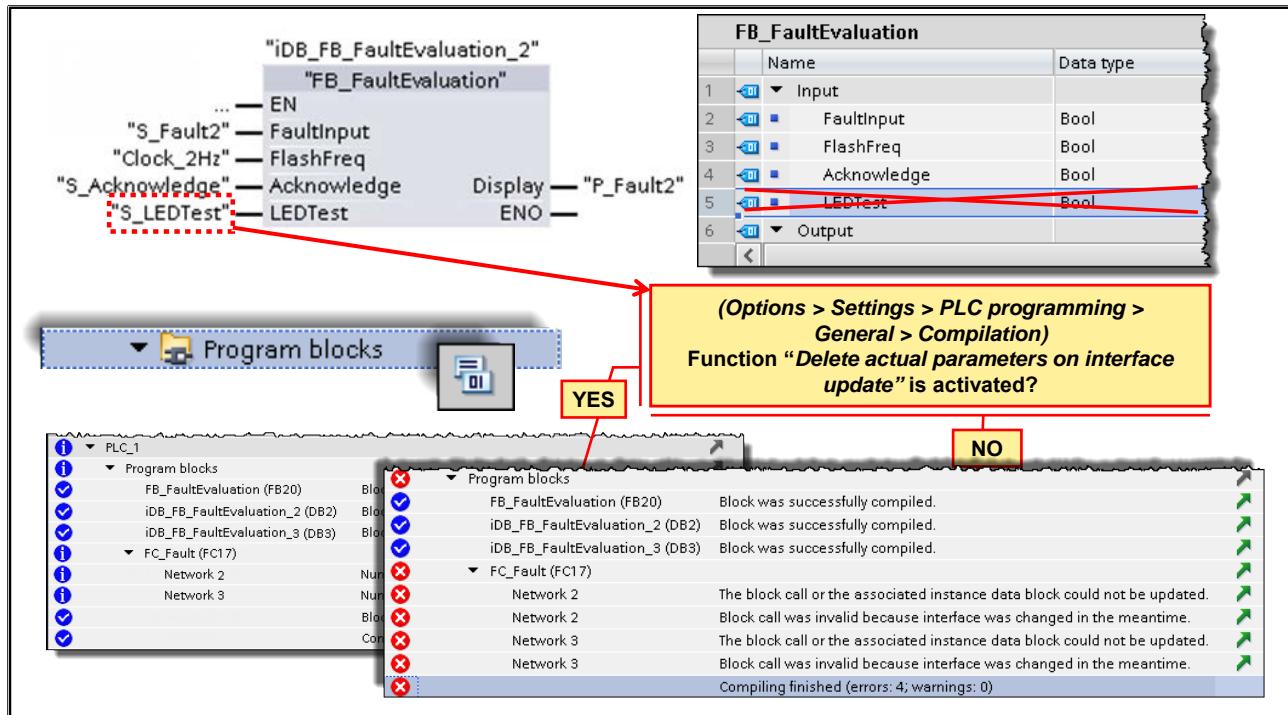
For functions, the added formal parameter must still be assigned before downloading into the CPU, since this is a "Must Assign".

For function blocks, the default value from the associated instance DB is used when the formal parameter is not assigned ("Can Assign").

- Manual Update

See 9.10.2 Manually Updating a Block Call

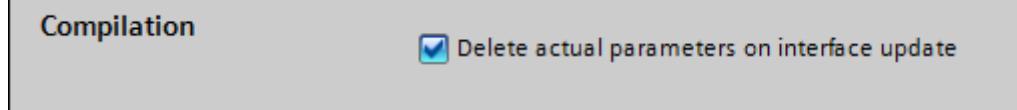
9.10.1. Removing Block Parameters Later On



If block parameters are **deleted (removed)** later on from a block already called in the program, you also have to update the calls of the block in the calling blocks.

- Automatic Update

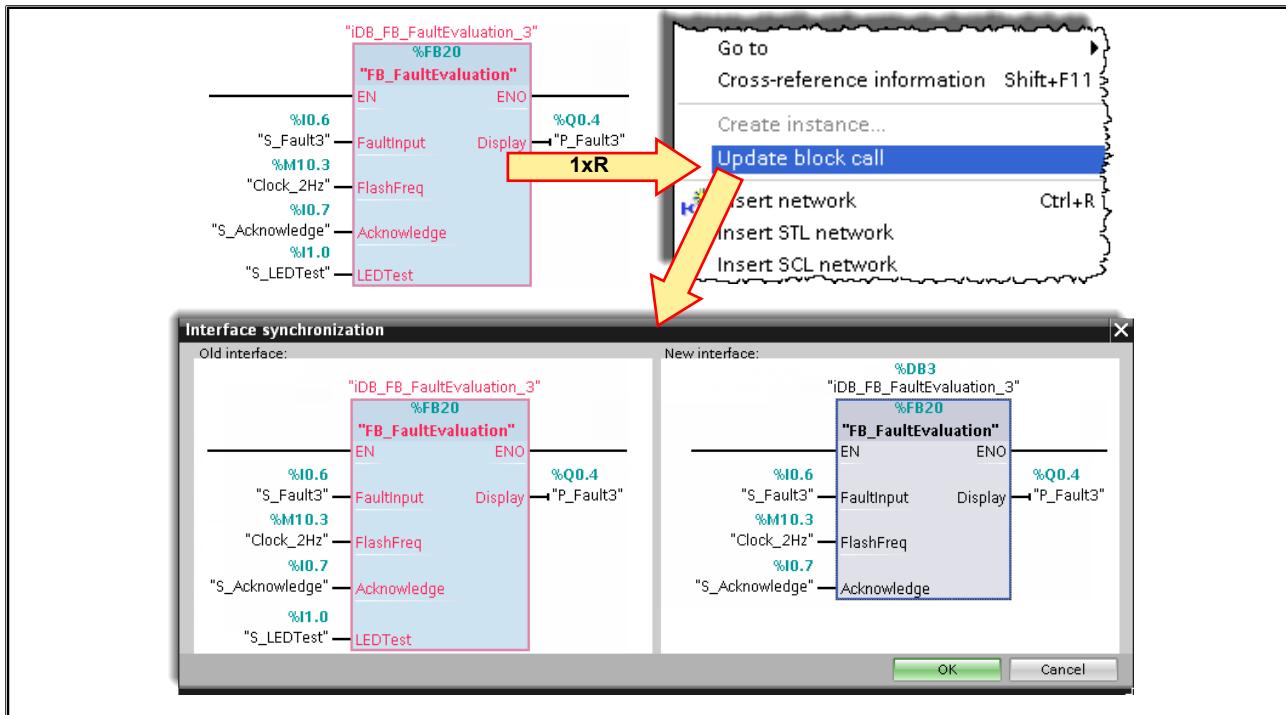
Attention/Caution: If the deleted formal parameters have already been assigned with actual parameters, then this automatic update only occurs if the function "Delete actual parameters on interface update" is activated under Options > Settings > PLC-programming > General > Compilation.



- Manual Update

See 9.10.2 Manually Updating a Block Call

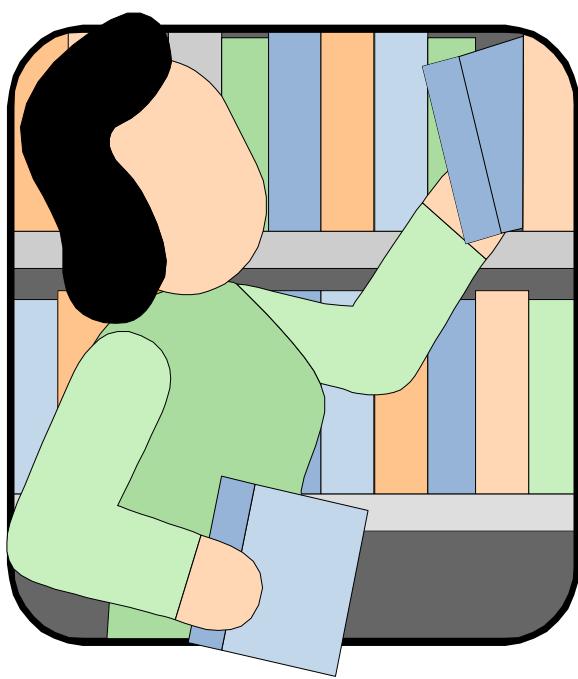
9.10.2. Manually Updating a Block Call



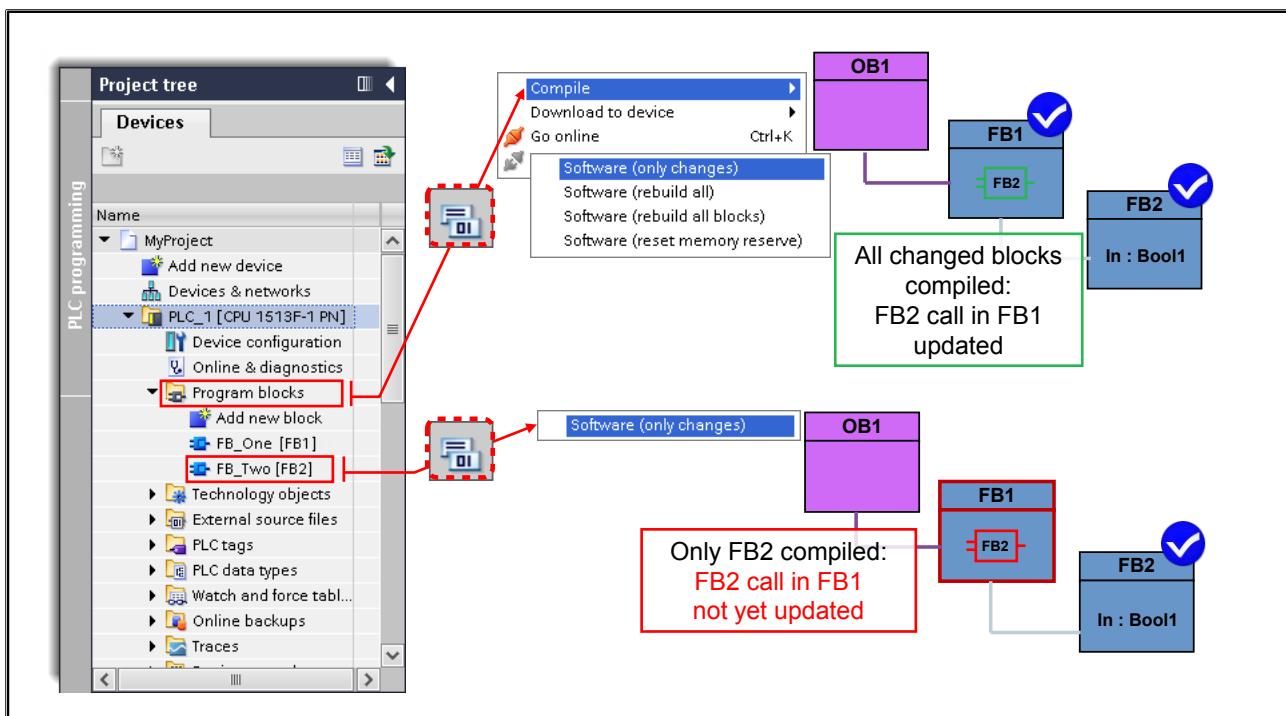
Manual Update

In the open, calling block, the inconsistent calls of a block are highlighted in red. By right-clicking the inconsistent call, the function "Update" can be selected in the context menu. A window then appears in which the old (faulty) and the new block call (in the picture without the parameter "LEDTest") are displayed. For function blocks, the associated instance DB is subsequently regenerated.

9.11. Additional Information



9.11.1. Compiling Individual / All Changed Blocks



Compile → Software (Only Changes):



If one individual block is selected in the Project tree or if a compilation is triggered through the button shown to the left when a block is open, only this single block is compiled (if it was changed). The disadvantage of compiling one individual block is that interface conflicts in the calling blocks caused by interface changes are not corrected.

If several blocks are selected or a block group, only those blocks modified since the last compilation are compiled (delta compilation).

If the "Program blocks" folder is selected, the delta compilation for the entire program is carried out.

Compile → Software (Rebuild All Blocks)/ or Rebuild All):

All, even those blocks not modified since the last compilation, are compiled.

Compile → Reset Memory Reserve:

Note: What a memory reserve is, is dealt with in Programming 2.

With the compilation process, the memory reserve of data blocks is also reset that is, that variables that were created later on in the course of data block expansions are removed from the memory reserve and integrated in the regular part of the data block.

Compilation Results

The status of the compilation is hierarchically displayed in the Inspector window "Info -> Compile". If errors occurred during compilation, you can jump directly to the error location by double-clicking on the error entry.

9.11.2. Global and Local Tags

	Global Tags	Local Tags
Validity range	<ul style="list-style-type: none"> Valid throughout the entire CPU, i.e. all blocks have access The name of the tag must be unique within the entire CPU 	<ul style="list-style-type: none"> Are only valid in the block in which they have been declared (defined) The name of the tag must be unique within the block
Operands	<ul style="list-style-type: none"> Inputs Outputs Memory bits Tags in data blocks SIMATIC Timers / Counters 	<ul style="list-style-type: none"> Temporary variables (in all code (logic) blocks) Static variables (only in function blocks)
Location of declaration	<ul style="list-style-type: none"> PLC tag table Global data blocks (Chap. 11) 	<ul style="list-style-type: none"> Declaration part of the block
Presentation	<ul style="list-style-type: none"> Presented in quotation marks Example: "Max" 	<ul style="list-style-type: none"> Presented preceded by # Example: #Max

Validity Range of Tags

Tags that are declared in the PLC tag table or in a global data block can be addressed by all CPU program blocks. For that reason, these tags are called global tags.

Tags and parameters that are declared in the declaration part of a code (logic) block are local operands; they can only be used in the statement part of the same block.

10

Contents

10. Digital Operations	10-2
10.1. Task Description: Counting the Transported Parts and Timed Monitoring of the Transportation	10-3
10.1.1. Acquiring, Processing and Outputting Data.....	10-4
10.1.2. Integer (INT, 16-Bit Integer) Data Type	10-5
10.1.2.1. Double Integer (DINT, 32-Bit Integer) Data Type	10-6
10.1.3. REAL and LREAL (Floating-point Number) Data Type	10-7
10.1.4. Data Types and Display Formats.....	10-8
10.2. Task Description: Counting the Transported Parts using Addition in "FB_CountADD".....	10-9
10.2.1. Basic Mathematical Functions: Addition	10-10
10.2.1.1. CALCULATE Box.....	10-11
10.2.2. Comparison Operations	10-12
10.2.3. Value Assignment of a Variable.....	10-13
10.2.4. Programming All Instructions using Empty Box.....	10-14
10.2.5. Exercise 1: Counting the Transported Parts using Addition in "FB_CountADD".....	10-15
10.2.6. Exercise 2: Calling "FB_CountADD".....	10-16
10.3. Task Description: Timed Monitoring of the Transport Sequences and Counting Parts using IEC Functions	10-17
10.3.1. IEC Timer	10-18
10.3.2. IEC Timer TON (ON Delay) Pulse Diagram	10-19
10.3.3. Exercise 3: Programming the Timed Monitoring of the Transports in "FC_Fault"	10-20
10.4. IEC Counters: CTU, CTD, CTUD.....	10-21
10.4.1. IEC Counters UP/DOWN: Inputs	10-22
10.4.2. IEC Counters UP/DOWN: Outputs	10-23
10.4.3. Exercise 4: Counting the Transported Parts using an IEC Counter	10-24
10.4.4. Exercise 5: Replacing the "FB_CountADD" Call with the "FB_Count" Call.....	10-25
10.5. Additional Exercise 6: Counting the Conveyor Faults by Expanding "FC_Fault"	10-26
10.5.1. Additional Exercise 7: Timely Lock-out of the Conveyor Motor Jogging	10-27
10.6. Additional Information	10-28
10.6.1. Comparator Operations: IN_RANGE, OUT_RANGE	10-29
10.6.2. Digital Logic Operations	10-30
10.6.2.1. Application Example: Digital Edge Evaluation	10-31
10.6.3. SIMATIC-Timer	10-32
10.6.3.1. SIMATIC-Counter	10-33
10.6.4. Value Ranges of Various Number Formats	10-34

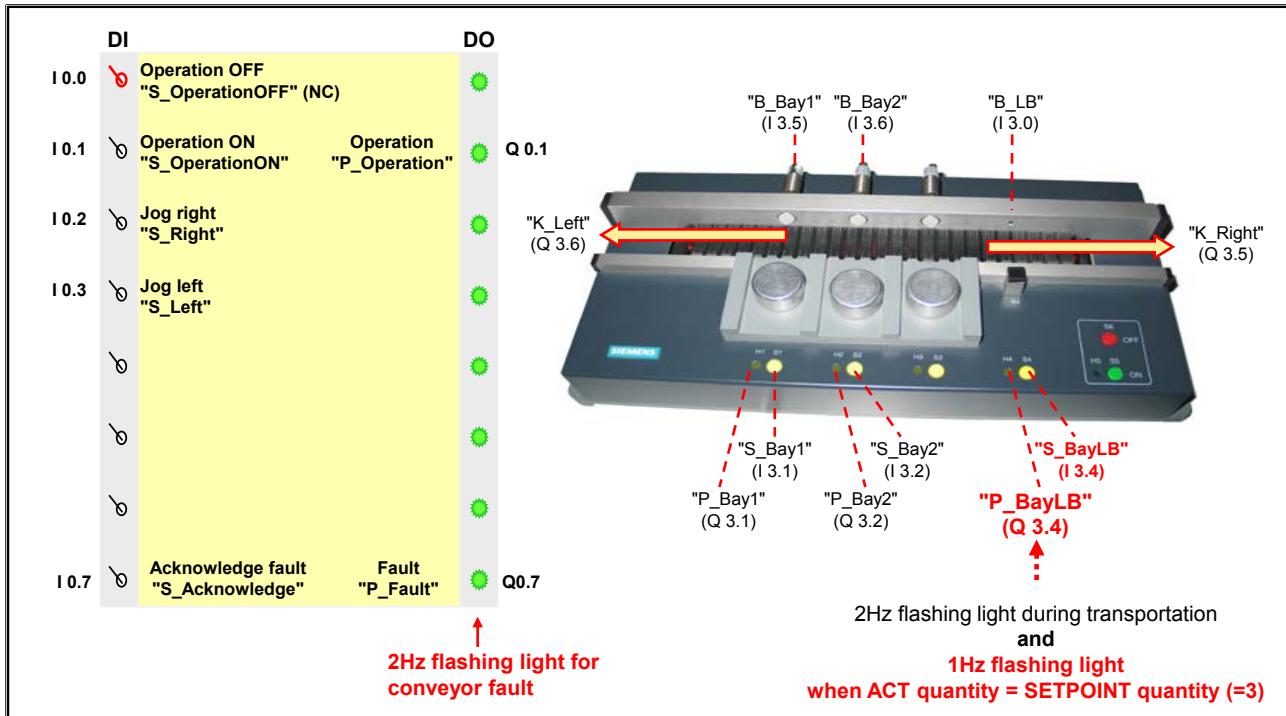
10. Digital Operations

At the end of the chapter the participant will ...

- ... be familiar with the structure of the data types INT, DINT, REAL and LREAL
- ... be familiar with various presentation types
- ... be able to use basic mathematical operations
- ... be familiar with comparison functions
- ... be familiar with the MOVE function for value assignment
- ... be familiar with the different counter and timer functions
- ... be able to use and program counter and timer functions



10.1. Task Description: Counting the Transported Parts and Time Monitoring of the Transportation



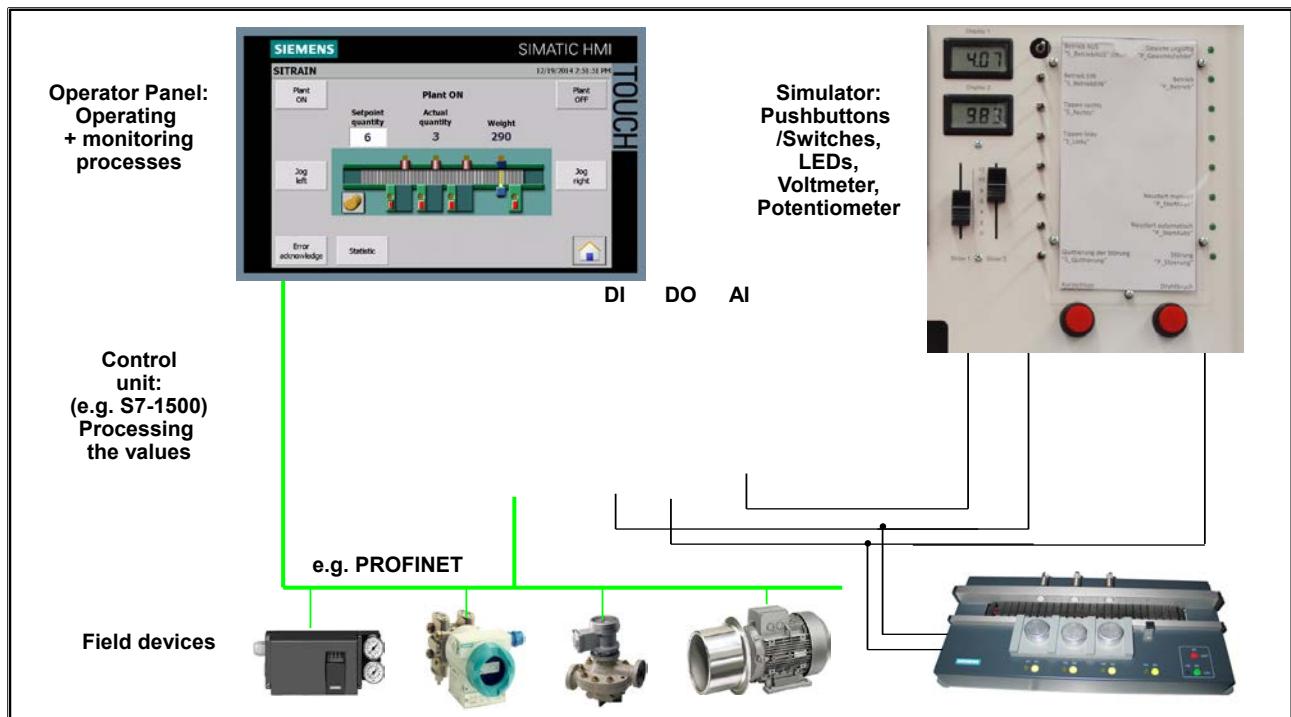
Function Up Till Now

When "P_Operation" (Q0.1) is switched on, parts are transported from Bay 1 or Bay 2 through the light barrier. The transport sequence starts as soon as a part is placed on Bay 1 or Bay 2 and the associated bay's pushbutton is pressed and it ends as soon as the part has passed the light barrier.

Task Description

- When "P_Operation" (Q0.1) is switched on, the transported parts are to be counted as soon as they have passed through the "B_LB" (I 3.0) light barrier ("B_LB" 0 → 1). The number of transported parts (ACTUAL quantity) is to be counted.
- The counter can be acknowledged (reset to 0) via the pushbutton "S_BayLB" (I 3.4). When "P_Operation" (Q0.1) is switched on, the ACTUAL quantity is also reset to 0.
- The indicator light "P_BayLB" (Q3.4) shows a 1Hz flashing light when the actual quantity has reached the setpoint quantity of 3 (= a new part must not be placed on the conveyor -> lock-out in "FC_Signal") and no further transport sequence can be started (-> lock-out in "FC_Conveyor"), and a 2Hz flashing light during parts transportation.
- Furthermore the time, that is required for the transport of a workpiece from Bay 1 and Bay 2 until it is through the light barrier, is measured.
- If the time it takes to transport exceeds 6 seconds, the conveyor is stopped and this is indicated at the output "P_Fault" (Q0.7) with a 1Hz flashing light.
- Only after this error is acknowledged with "S_Acknowledge" (I 0.7) can a new part be transported.

10.1.1. Acquiring, Processing and Outputting Data



Binary/Digital Processing

True logic control systems are recognizable in the fact that they exclusively process binary data. The performance of today's control computer, as well as tasks in the areas of data processing, quality control, among others, has increased the importance of digital data processing using PLCs. Digital process variables can be found in all areas of open-loop control - such as in connected devices for process operating and monitoring or in the control of field devices.

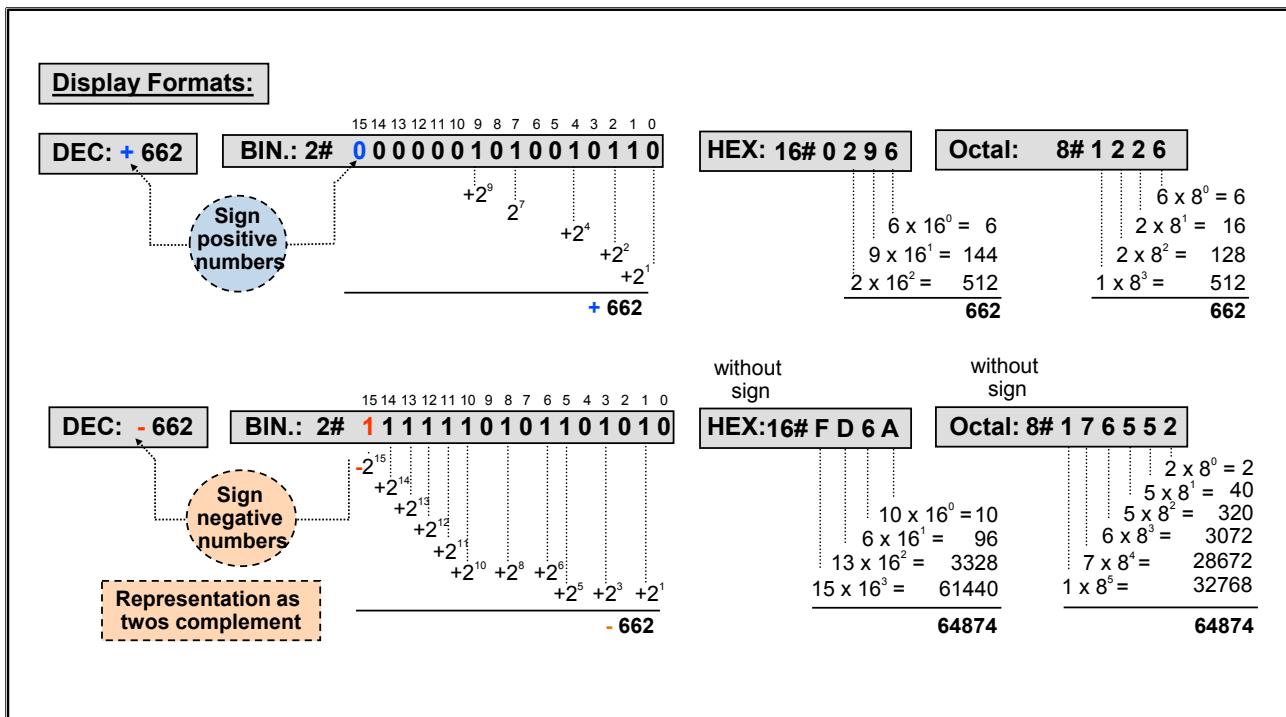
Operating and Monitoring

The goal of process monitoring is to provide the operator with up-to-the-minute information about the working machine or system quickly, concisely and clearly as well as the opportunity to intervene, control and influence the process. Depending on the type of device connected, different number formats for the coding of data are used to transmit data between devices and PLC, as well as for storing and processing data in the PLC.

Field Devices

Today as well, field devices that acquire process data or that control the process are supplied directly with digital variables through field bus systems.

10.1.2. Integer (INT, 16-Bit Integer) Data Type



Integer (16-Bit) Data Type

An *Integer* data type value is a whole number value, that is, a value without a decimal point. SIMATIC S7 stores *Integer* data type values with sign in 16 bit code. This results in a value range from -32768 to +32767. As well, SIMATIC S7 provides arithmetic operations for processing Integer values.

Decimal

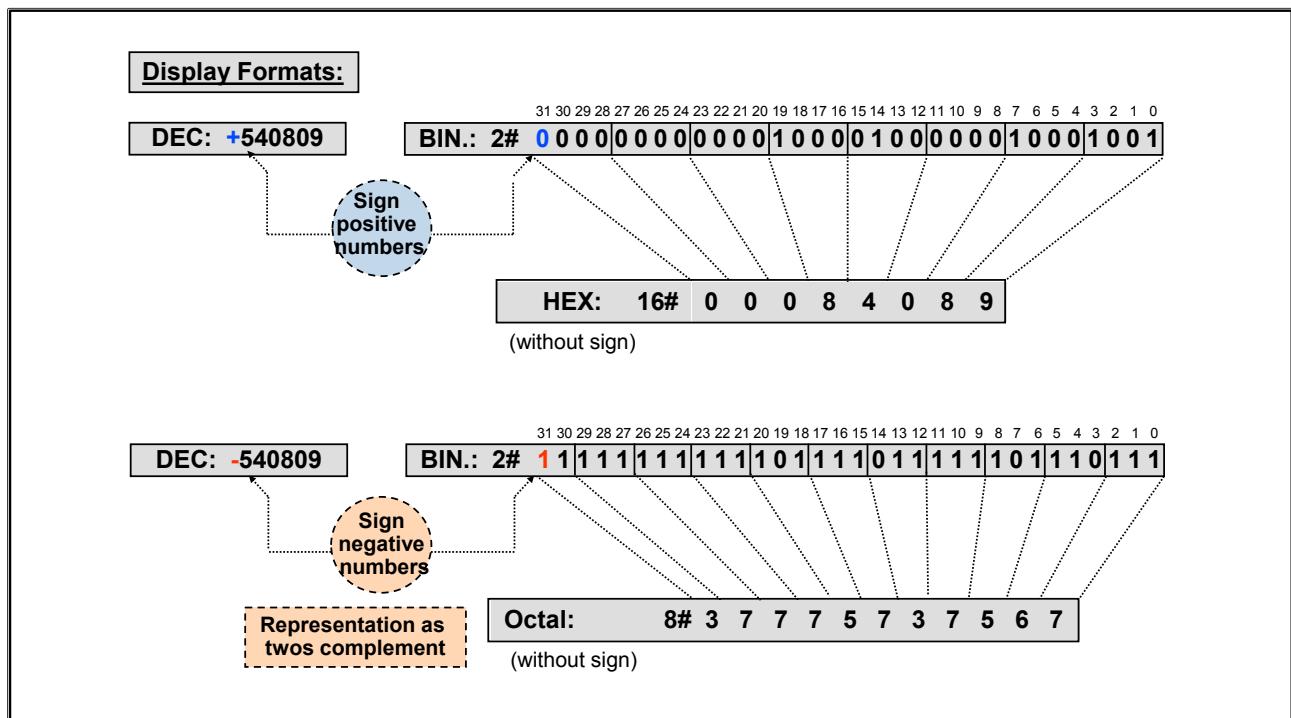
STEP 7 uses the *Decimal* display format, that is, with sign and without explicit format description, to specify the constants of the *Integer* data type. The use of constant Integer values in the *Binary*, *Hexadecimal*, *Octal* display formats is possible in principle, but because of the poor legibility, they are more or less not suitable.

Binary

In a digital computer system, all values are stored in a binary-coded form. Only the digits 0 and 1 are available in the binary number system. Base 2 of this number system results from the number of available digits. Accordingly, the value of every bit of a binary number results from a power of Base 2. This is also expressed in the format specification **2#....**.

Negative values are represented as binary numbers in two's complement. In this representation, the most significant bit (bit no. 15 for the Integer data type) has the value -2^{15} . Since this value is greater than the sum of all residual values, this bit also has the sign information. That is, if this bit = 0, then the value is positive; if the bit is = 1, then the value is negative. The conversion of a binary number into a decimal number is made by adding the values of the bits that have a 1 (see picture).

10.1.2.1. Double Integer (DINT, 32-Bit Integer) Data Type



Double Integer (32-Bit Integer)

SIMATIC S7 stores *Double Integer* data type values with sign as 32 bit code. This results in the value range from -2147483648 to +2147483648.

Hexadecimal

The hexadecimal number system provides 16 different digits (0 to 9 and A to F). This results in Base 16 of this numbers system. Accordingly, the value of every bit of a hexadecimal number results from a power of Base 16. Hexadecimal numbers are specified with **16#** for identifying the basic numbering system. The number of specifiable bits is variable from 1 to 16. The digits A to F correspond to the decimal values 10 to 15. The value 15 is the last value that can be binary-coded - without sign - with 4 bits. Out of this correlation, the simple conversion of a binary number into a hexadecimal number and vice versa can be obtained. In this way, four binary bits each can easily make up one digit of a hexadecimal number.

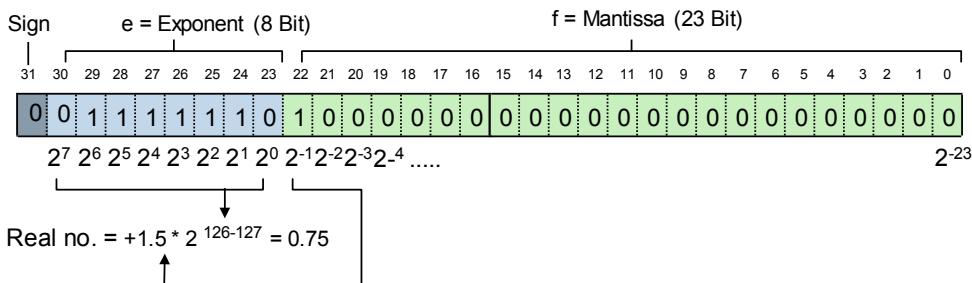
Octal Number

The octal number system provides 8 different digits (0 to 7). This results in Base 8 of this numbers system. Accordingly, the value of every bit of an octal number results from a power of Base 8. Octal numbers are specified with **8#** for identifying the basic numbering system. The value 7 is the value that can be binary-coded - without sign - with 3 bits. In this way, three binary bits each can make up one digit of an octal number.

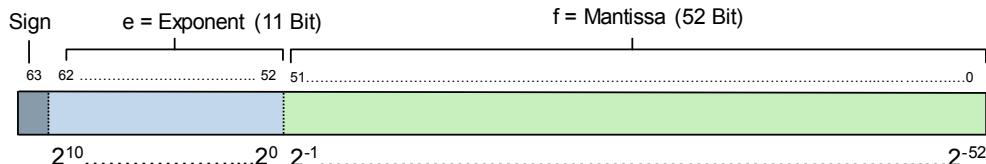
10.1.3. REAL and LREAL (Floating-point Number) Data Type

General format of a Real number (32 Bit) = (Sign) • (x.f) • (2^{e-127})

Example: 0.75



General format of an LReal number (64 Bit) = (Sign) • (x.f) • (2^{e-1023})



Real/LReal

The previously described INT and DINT data types are used to store whole number values with sign. Accordingly, only operations that supply a whole number value as the result can be performed with these values. In cases where analog process variables such as voltage, current, and temperature etc., have to be processed, it becomes necessary to use *Real* values (real numbers, "decimal numbers"). In order to be able to represent such values, binary digits have to be defined whose value is less than 1 (power of base 2 with negative exponent).

Format

In order to be able to form the greatest possible value range within a defined memory capacity, you must be able to select the decimal point position as required. Early on, IEEE defined a format for floating-point numbers. This format was laid down in IEC 61131 and was included in STEP 7. This format makes it easy to process a variable decimal point position. In the binary code of a 32 Bit floating-point number, a portion of the binary digits contain the mantissa (23 Bit) and the rest contain the exponent (8 Bit) and the sign bit of the floating-point number. A 64 Bit floating-point number also has the sign bit; however, the exponent is 11 Bit and the mantissa 52 Bit.

After you enter a constant real value (for example: 0.75), the Editor automatically makes a conversion to scientific notation (for example: 7.5000e-001).

Application

Floating-point numbers are used for "analog value processing", among other things. A great advantage of floating-point numbers is in the number of operations possible with such numbers. These include, in addition to the standard operations such as: +, -, *, / also instructions such as sin, cos, exp, ln, etc, that are used mainly in closed-loop control algorithms.

10.1.4. Data Types and Display Formats

i	Name	Address	Display format	Monitor value	Modify value	Comment
1		%IW2	DEC	1		
2		%IW2	Bin	2#0000_0000_0000_0001		
3	// BOOL					
4		%I2.7	Bool	FALSE		
5		%I2.0	Bool	FALSE		
6	"B_LB"	%I3.0	Bool	TRUE		
7	// Monitoring IW4 Hexadecimal, Decimal and Binary data format					
8		%IW4	Hex	16#1234		
9		%IW4	Bin	2#0001_0010_0011_0100		
10		%IW4	DEC	4660		
11						
12	"MW_ACT"	%MW20	DEC+/-	1234	1234	
13	"MW_ACT"	%MW20	Hex	16#04D2		
14	"MW_ACT"	%MW20	Bin	2#0000_0100_1101_0010		
15	// MW20 high and low-Byte					
16		%MB20	Hex	16#04		
17		%MB21	Hex	16#D2		
18	// "Test_2" Floating-Point Number (REAL) and Hexadecimal					
19	"Test"	%MD80	DEC	1234	1234	
20	"Test"	%MD80	Hex	16#0000_04D2		
21						
22		%MW83	Hex	16#D244	Invalid value !!! "accessed in between"	
23						
24	"Test_2"	%MD84	Floating-point nu...	1234.0	1234.0	
25	"Test_2"	%MD84	Hex	16#449A_4000		
26						

Display Formats

Different display formats can be selected in both the "Monitor / Modify Variables" and the "Monitor (Block)" test functions to display variables or register contents. Basically, every variable can be monitored with several display format options. Depending on the variable's data type, it becomes apparent that monitoring with the appropriate display format makes more sense.

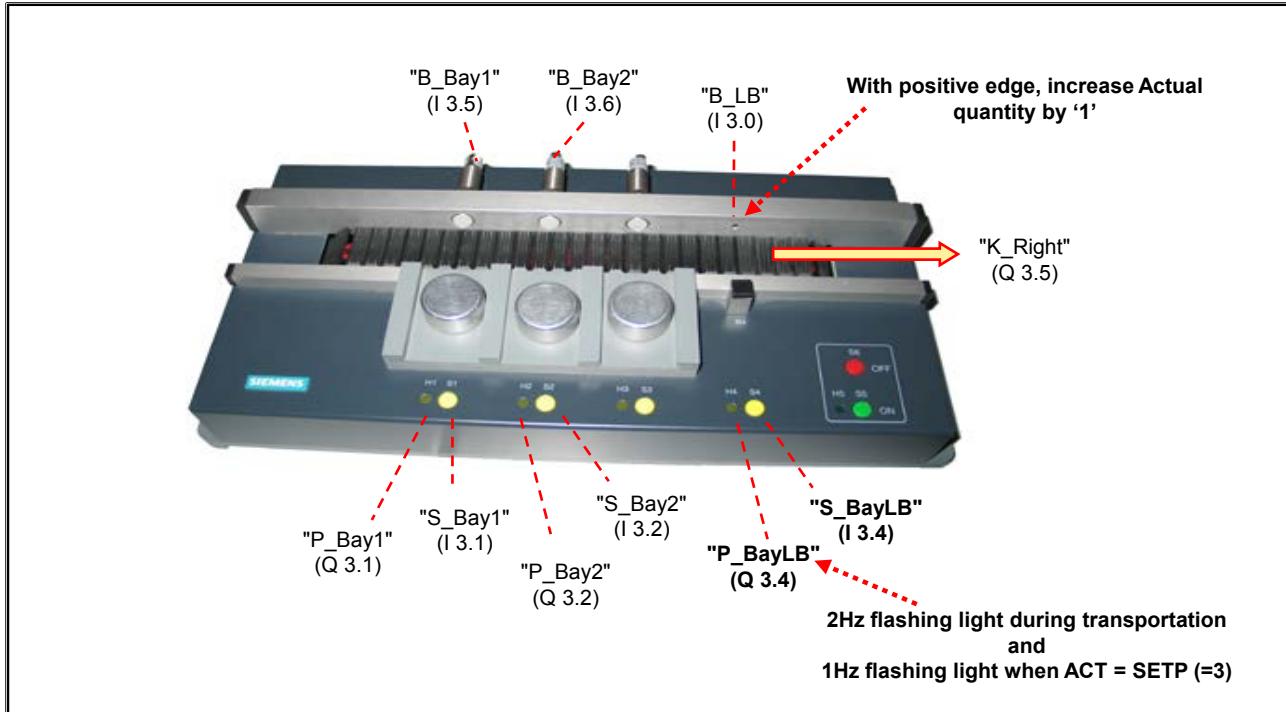
- **BOOL:** Display a single bit (only possible for a variable of the BOOL data type)
- **BIN:** Display the individual bits of a variable (makes sense for variables of the BYTE, WORD, DWORD, LWORD data types)
- **HEX, BCD:** Display the contents of a variable as a hexadecimal number, or, a BCD number (makes sense for variables of the BYTE, WORD, DWORD, LWORD data types)
- **DEC:** Display the contents of a variable as decimal number (not BCD!) **without** sign (makes sense for variables of the USINT, UINT, UDINT, ULINT data types)
- **DEC+/-:** Display the contents of a variable as decimal number (not BCD!) **with** sign (makes sense for variables of the SINT, INT, DINT, LINT data types)
- **FLOATING Point:** Display the contents of a variable as floating-point number (makes sense for variables of the REAL, LREAL data types)
- **and others...**

Addressing

The SIMATIC S7 memory is universally byte-oriented. Accordingly, memory word MW 20, for example, contains the memory bytes MB 20 (high byte) and MB 21 (low byte, see picture), the memory double-word MD 80, the memory bytes MB 80, 81, 82 and 83.

For absolute accesses to variables (such as, with *MD 82*), you must make sure that the dimension of the access (here *MD...*) as well as the address (always equal to the address of the high byte, here 82) is correct. Through an inadvertent "accessing in between", an invalid value would be loaded (such as, with *MW 83*, see picture). Such errors can be avoided with the symbolic addressing of variables.

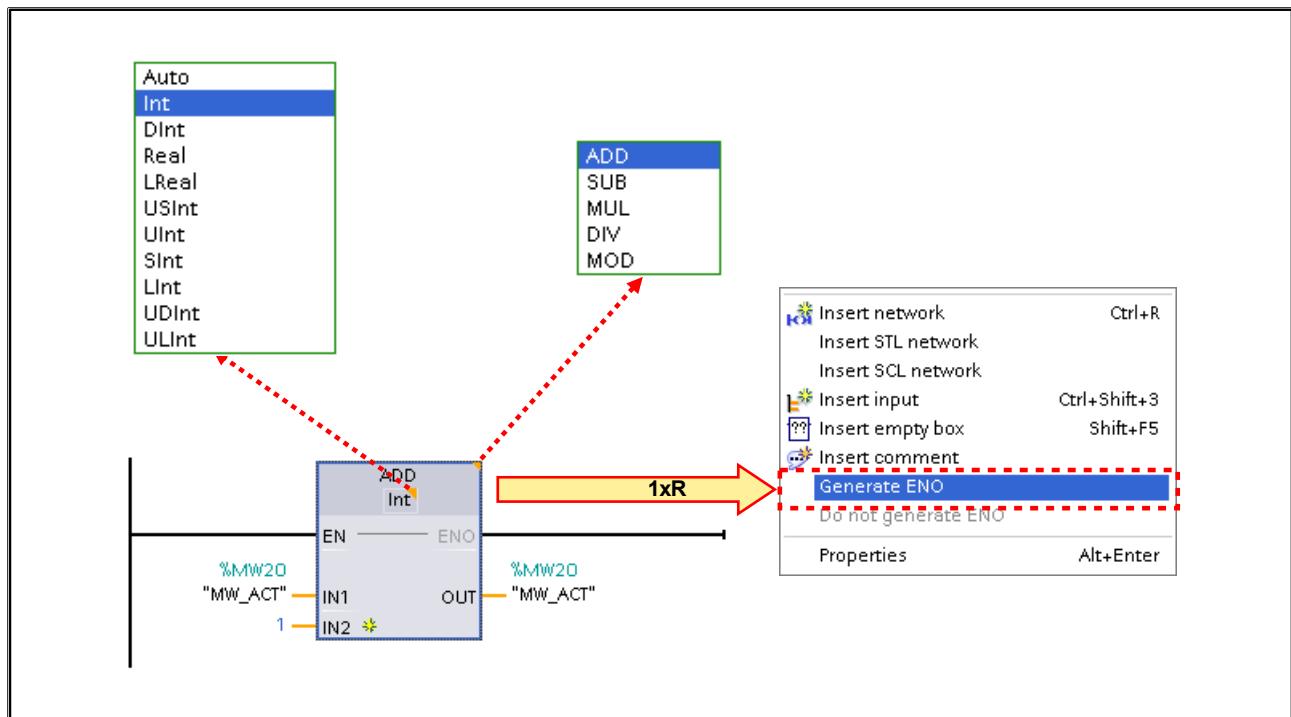
10.2. Task Description: Counting the Transported Parts using Addition in "FB_CountADD"



Task Description

- When "P_Operation" (Q0.1) is switched on, the transported parts are to be counted as soon as they have passed the light barrier "B_LB" (I 3.0) ("B_LB" 0->1).
- When the maximum quantity is reached, it is displayed via the LED "P_Bay_LB" (Q3.4) and further transport is no longer possible.
- As long as no transport is possible, the LEDs at the Bays 1 and 2 are dark and thus also signal that no transport is possible.
- The counter can be acknowledged (reset to 0) at any time via the pushbutton "S_BayLB" (I 3.4). When "P_Operation" (Q0.1) is switched on, the ACTUAL quantity is also reset to 0.

10.2.1. Basic Mathematical Functions: Addition



Arithmetic Operations

There is a series of arithmetic operations available for the processing of variables of the arithmetic data types, such as, integer (INT), double integer (DINT) and real (REAL).

Inputs and Outputs of the LAD/FBD Elements:

- **EN (enable)**

The execution of the operation can be determined as follows via the EN input:

- EN is not connected: The operation is always (regardless of the RLO) executed
- Logic operation at EN is fulfilled (RLO = 1): The operation is executed
- Logic operation at EN is not fulfilled (RLO = 0): the operation is not executed

- **IN1 / IN2**

The arithmetic calculation is applied to the values delivered to IN1 and IN2 and the result is output to OUT.

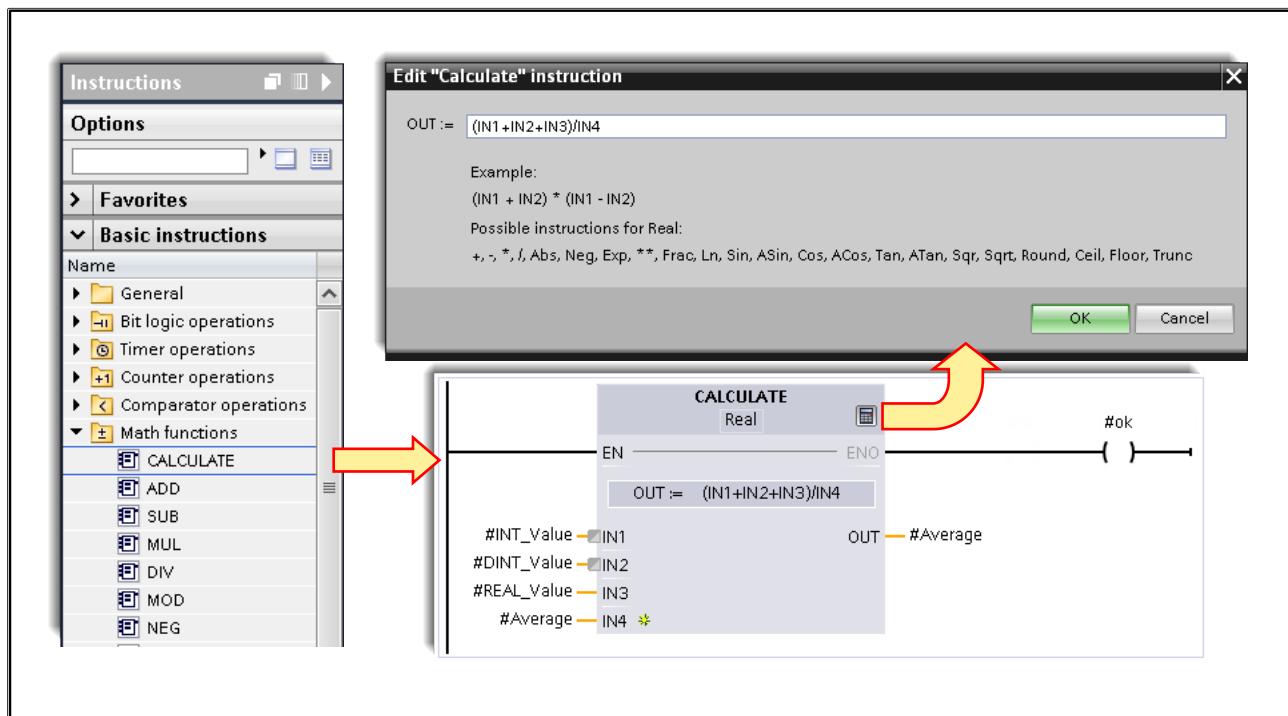
- **ENO (enable output)**

(You can activate and deactivate the generation of the ENO output via the context menu.
Caution: The setting is valid for all further functions that are inserted into the project.)

If ENO is activated, it can accept the following values:

- ENO = 0 since the actual parameter at the EN input has the value FALSE
 - OUT is not written (the variable delivered to OUT is not written, that is, it keeps its original value)
- ENO = 0 because an error has occurred
 - OUT contains an invalid value (the variable delivered to OUT is overwritten with an invalid value)
- ENO = 1 (instruction was executed without error):
 - OUT contains result (the variable delivered to OUT is overwritten with the result)

10.2.1.1. CALCULATE Box

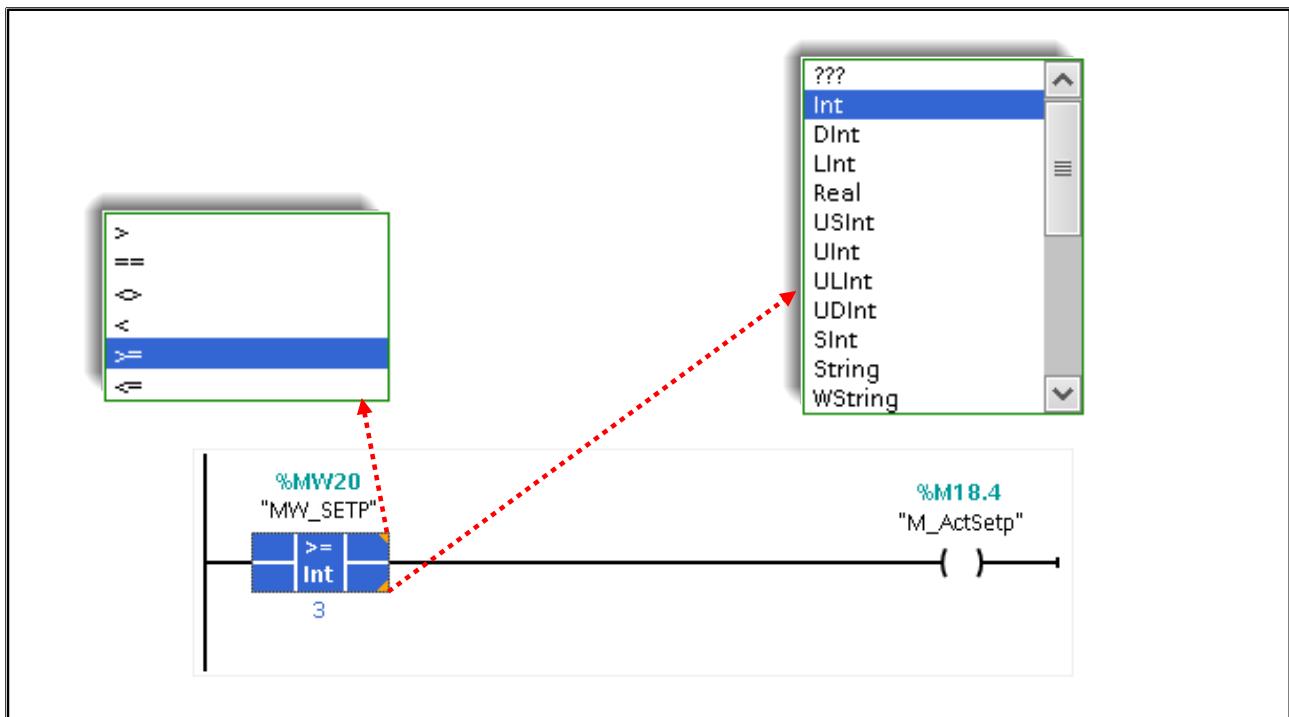


Calculate Box

With the Calculate Box, calculations can be combined which contain several different math operations.

The implicit data type conversion is available at the inputs and outputs of the box and a check can be made through the ENO output as to whether errors occurred in the calculation, such as, overflows in the type conversions or the math operations, or, that the result of the calculate box is error-free.

10.2.2. Comparison Operations



Comparison Function

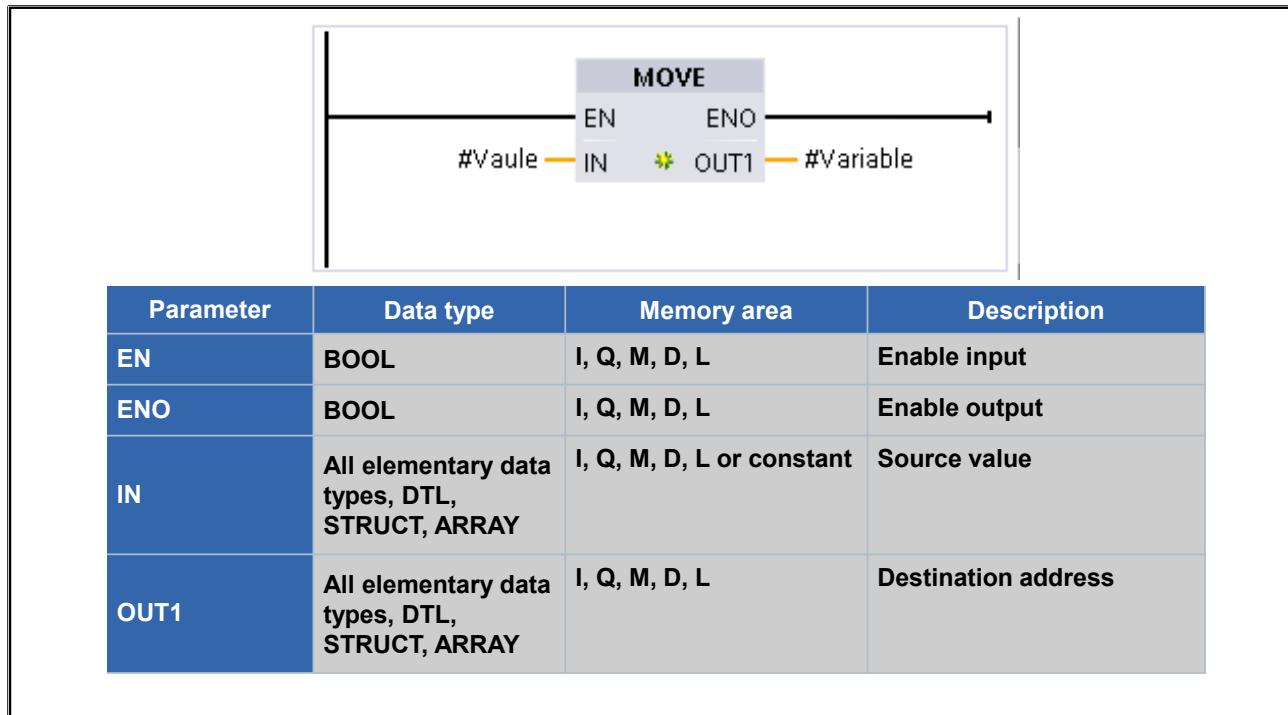
With the comparison instructions, the following pairs of numerical values can be compared:

- all variations of integers
- all variations of floating-point numbers (Real = IEEE floating-point numbers)
- all variations of TIME data types

If the result of the comparison is "true", then the output of the operation is "1", otherwise it is "0". The input IN1 is compared with IN2 according to the selected type of comparison:

- | | | | |
|------|-----|-----------------------------|------|
| • == | IN1 | is equal to | IN2 |
| • <> | IN1 | is not equal to | IN2 |
| • > | IN1 | is greater than | IN2 |
| • < | IN1 | is less than | IN2 |
| • >= | IN1 | is greater than or equal to | IN2 |
| • <= | IN1 | is less than or equal to | IN2. |

10.2.3. Value Assignment of a Variable

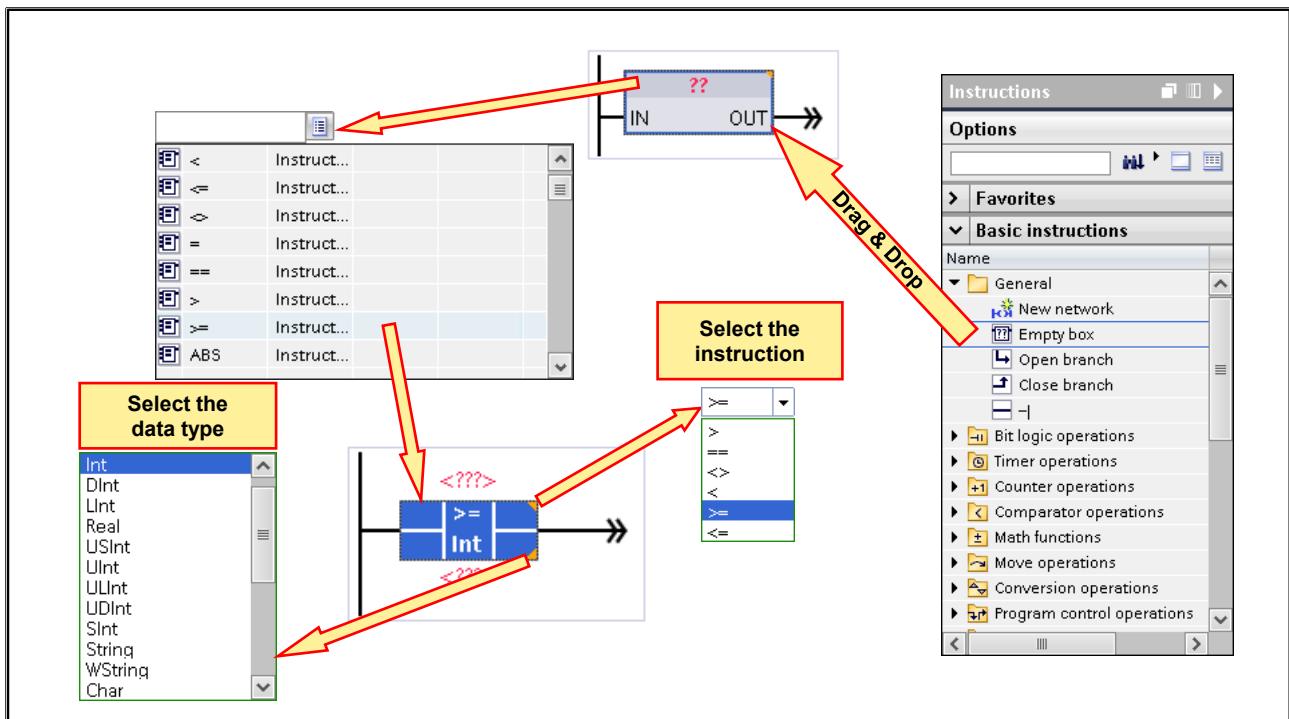


MOVE

You use the "MOVE" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output.

The operation is only executed if the signal status at the enable input EN is "1" or is not assigned. In this case and with error-free transfer, the ENO output also has signal status "1".

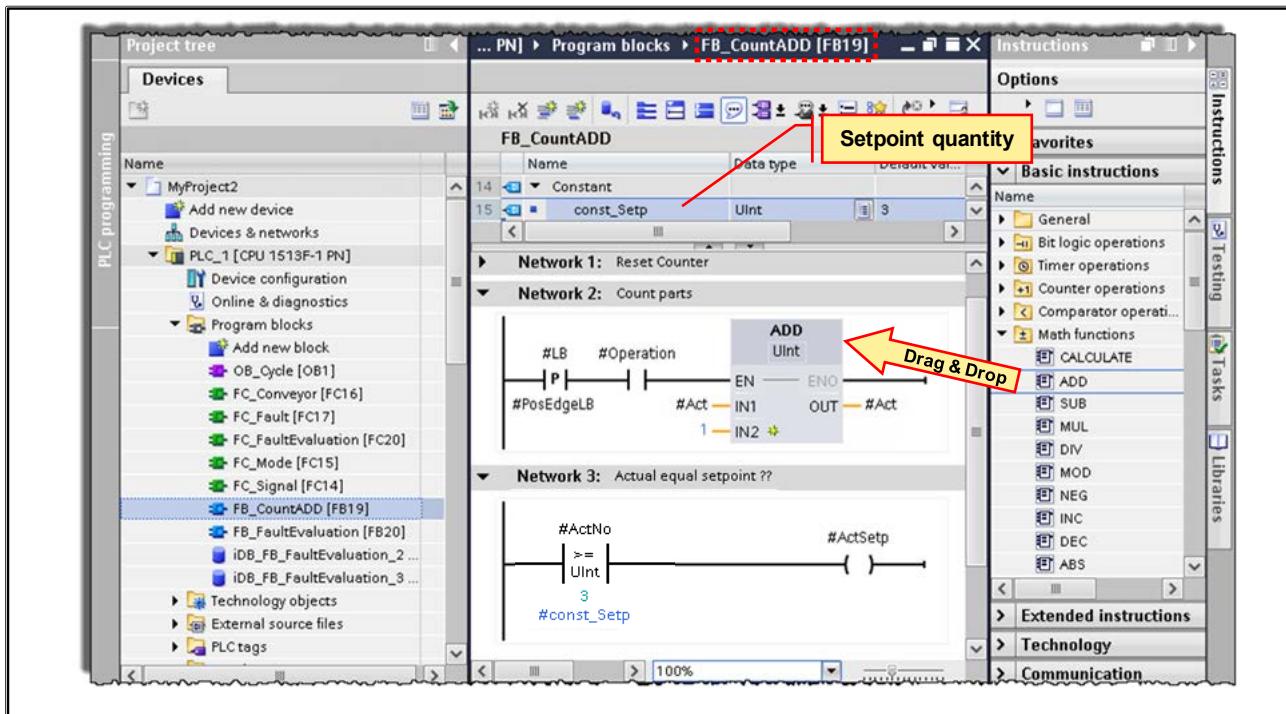
10.2.4. Programming Instructions using Empty Box



Programming an Instruction using "Empty Box"

An instruction can also be programmed in the so-called "Empty box". The empty box is first pulled from the "Instructions" task card using drag & drop, or it is pulled from the "Favorites" onto the network. Then at the empty box, you select which instruction is to be used with which data type.

10.2.5. Exercise 1: Counting the Transported Parts using Addition in "FB_CountADD"



Task

- When "P_Operation" (Q0.1) is switched on, the transported parts are to be counted as soon as they have passed the light barrier "B_LB" (I 3.0) ("B_LB" 0->1). The number of transported parts (ACTUAL quantity) is to be recorded with a counter and stored in the static variable #ACT.
- When the setpoint quantity (constant value 3) is reached, then no new transport sequence can take place which is also visible on the LEDs "P_Bay1" and "P_Bay2".
- The counter can be acknowledged (reset to 0) at any time via the pushbutton "S_BayLB" (I 3.4). When the operation is switched on, that is, with a positive edge at "P_Operation" (Q0.1), the ACTUAL quantity is also reset to 0.

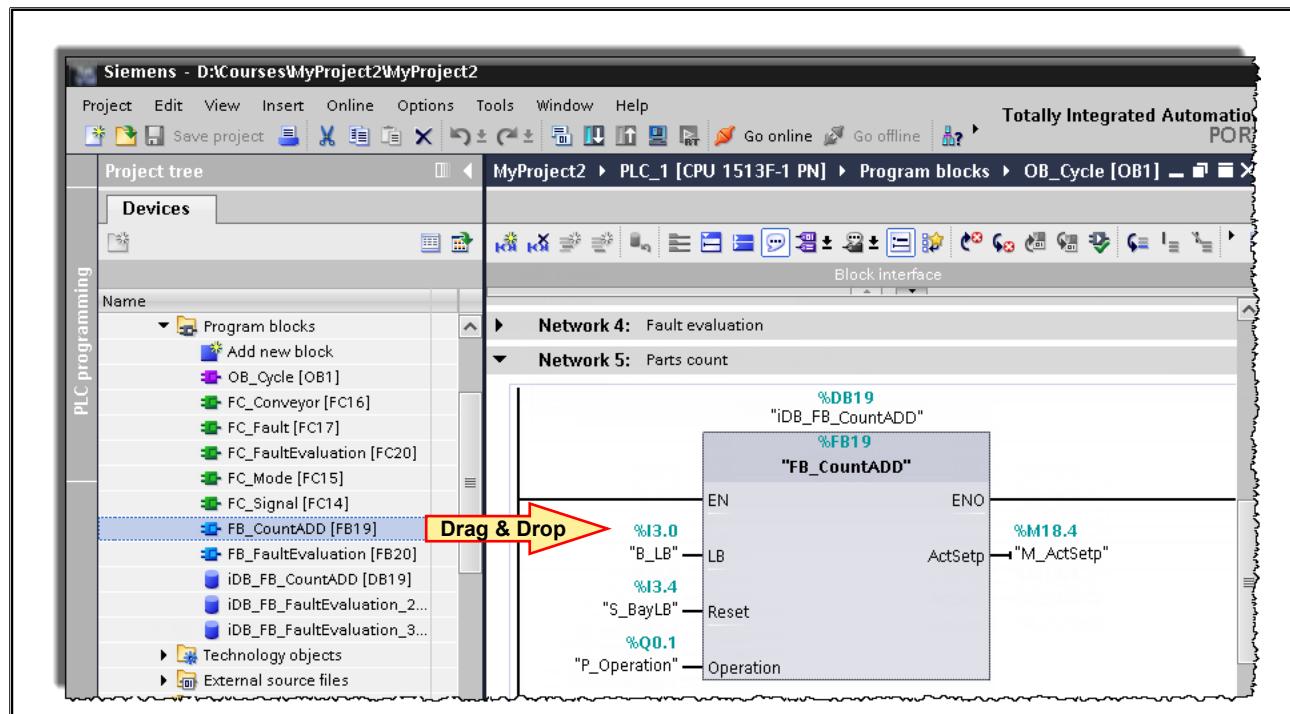
What to Do:

- Create the new function block "FB_CountADD".
- Declare the necessary parameters and variables:

Input:	- LB - Reset - Operation	BOOL, BOOL, BOOL,	Light barrier signal Reset the actual no. of parts to 0 Operation mode
Output:	- ActSetup	BOOL,	the actual no. of parts has reached the setpoint value
Static:	- PosEdgeLB - PosEdgeOp	BOOL, BOOL,	auxiliary bit for the edge evaluation of the Light barrier signal auxiliary bit for the edge evaluation of the Operation mode signal
	- Act	UINT,	actual no. of transported parts
Constant:	- const_Setp	UINT,	setpoint no. of parts to be transported

- Program the new "FB_CountADD" block with the appropriate arithmetic operations and don't use any global variables (tags) to do so. For the edge evaluation of the light barrier, use the static variable #PosEdgeLB and save the current quantity of the transported parts in the static variable #ACT.

10.2.6. Exercise 2: Calling "FB_CountADD"



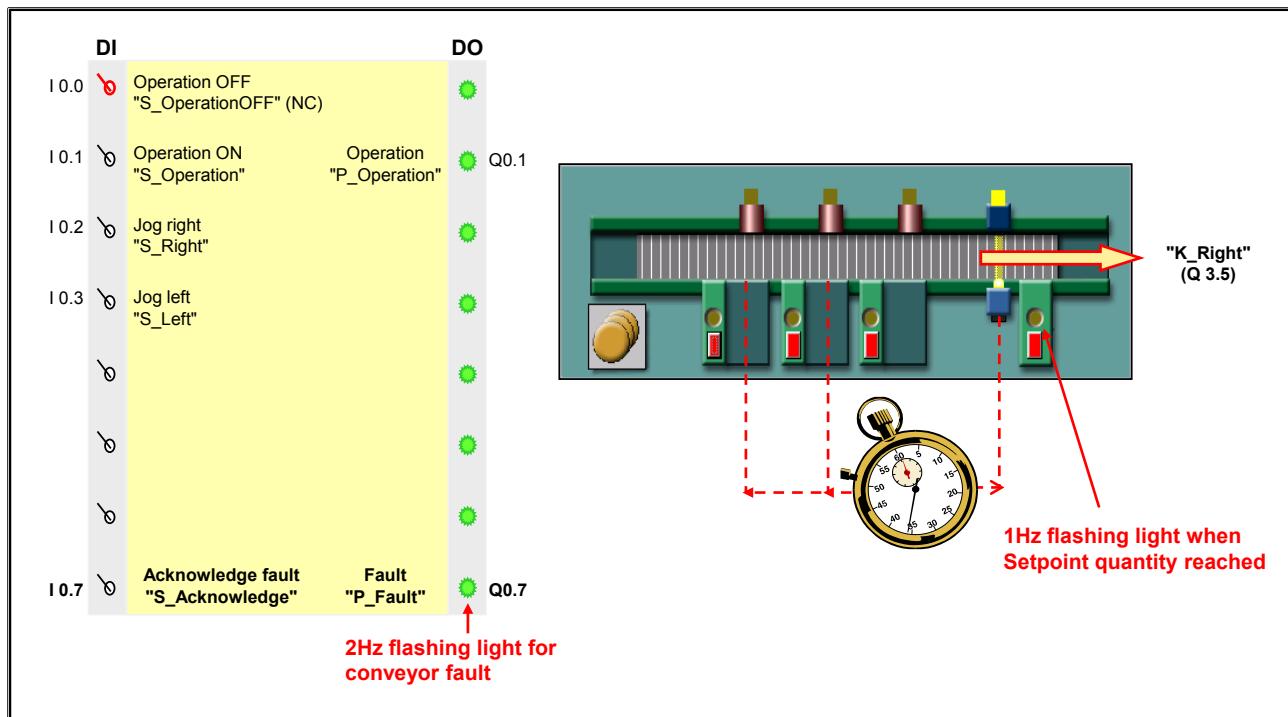
Task

You are to call the "FB_CountADD" block and assign it the relevant parameters.

What to Do:

1. In OB_Cycle, call the new "FB_CountADD" block.
2. Supply the formal parameters with the relevant actual parameters. (see picture)
3. Modify the blocks "FC_Conveyor" and "FC_Signal" in such a way that when the quantity of 3 is reached, it is no longer possible to transport a part, the LEDs of Bays 1 and 2 are switched off and the LED at the light barrier bay flashes with a 1 Hz frequency. For this, use the global variable "M_ActSetup"
4. Compile and save your project and check that it functions correctly.

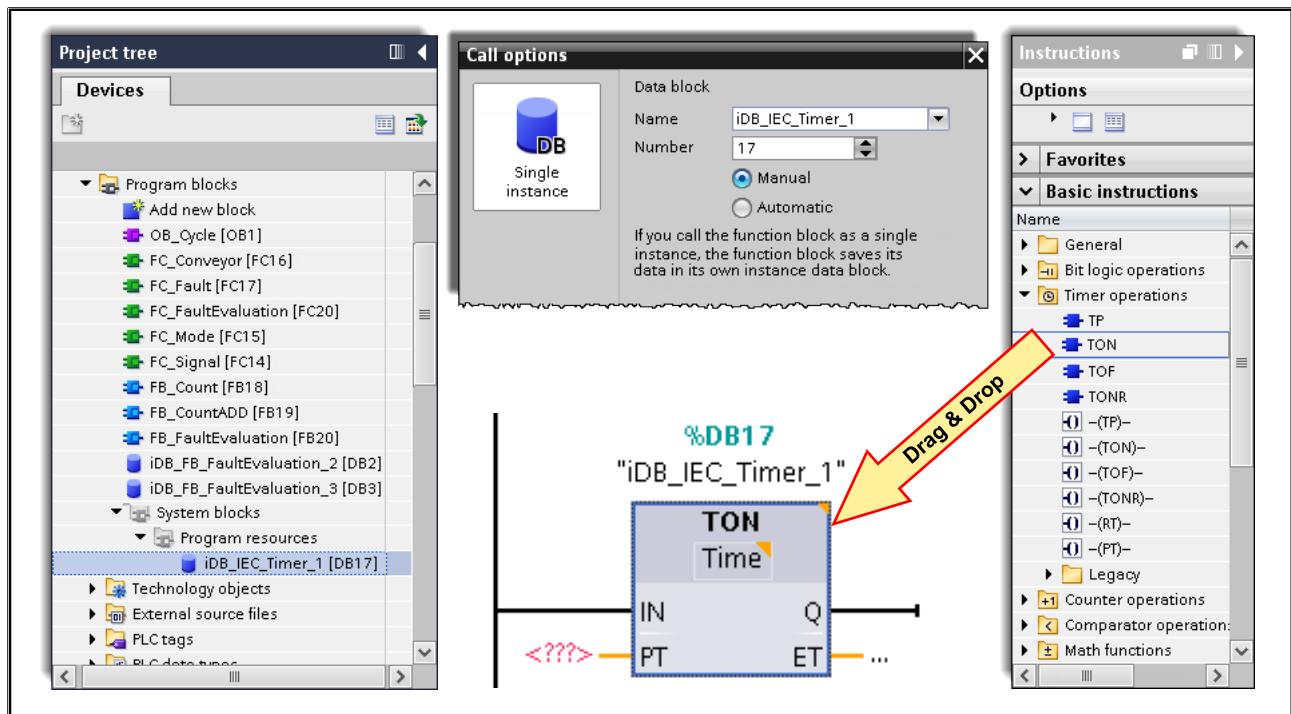
10.3. Task Description: Timed Monitoring of the Transport Sequences and Counting Parts using IEC Functions



Task Description

1. The automatic transport sequences are to be monitored for time with the help of an IEC function. The monitoring is to function as follows:
 - If a transport sequence takes longer than the 6 second monitoring time, there is a fault and the conveyor motor is automatically switched off.
 - A fault is displayed with a 2Hz flashing light on the simulator LED "P_Fault" (Q0.7).
 - A fault can be acknowledged via the simulator switch "S_Acknowledge" (I 0.7).
 - As long as there is an unacknowledged fault, the indicator lights "P_Bay1" (Q3.1) and "P_Bay2" (Q3.2) are dark and no new transport sequence can be started.
2. The counting of the transported parts is to be implemented with an IEC function.

10.3.1. IEC Timer



Data Block

In addition to internally required variables, the timer function also stores the current already expired time in a data block which must be specified when programming the timer function. The specified data block is automatically generated by the Editor with exactly the internal structure that the timer function requires. The user has no further programming effort with this data block other than having to download it into the CPU.

Input Variable "PT"

The variable PT of the time function can be of the type Time or LTime.

Data Type TIME

The contents of a variable or constant of the data type TIME is interpreted as an integer number in milliseconds and stored in the memory as a 32-bit integer with sign. The representation contains information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms).

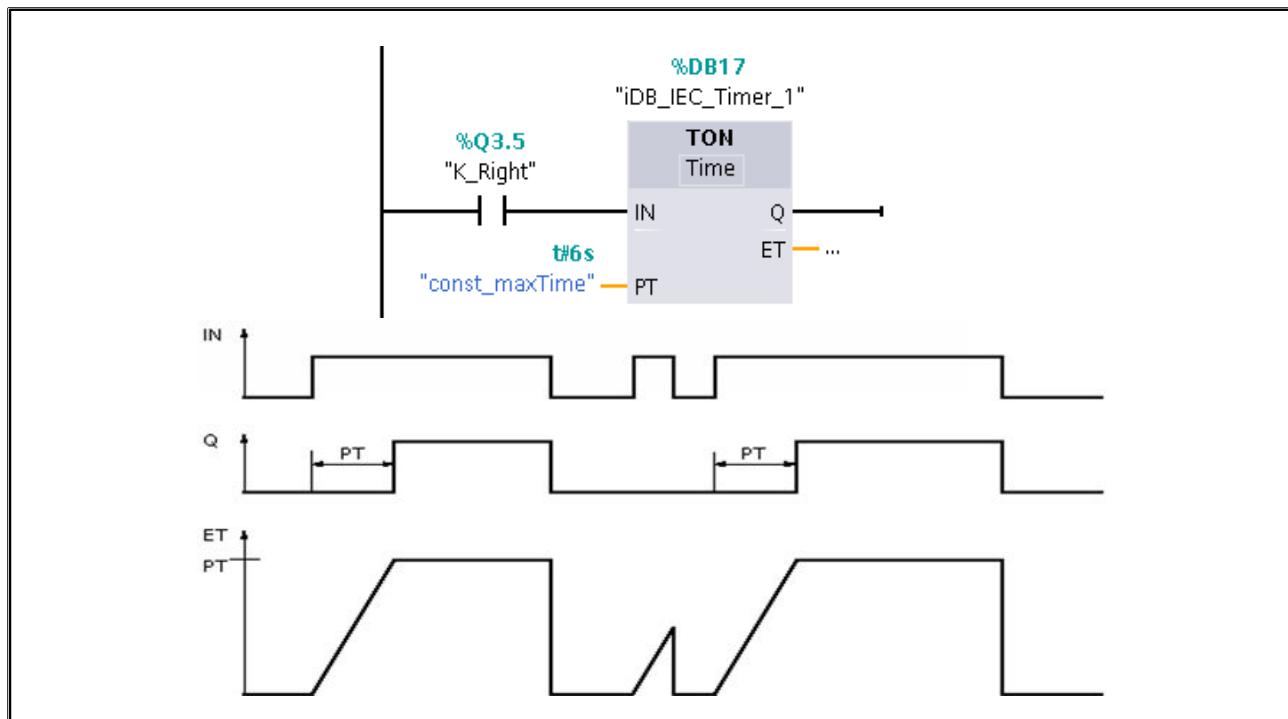
Value range: T#-24d20h31m23s648ms to T#+24d20h31m23s647ms

Data Type LTIME

The contents of an operand of the data type LTIME is interpreted as nanoseconds. The representation contains information for days (d), hours (h), minutes (m), seconds (s), milliseconds (ms), microseconds (us) and nanoseconds (ns).

Value range: -106751d 23h 47m 16s 854ms 775us 808ns
to
+106751d 23h 47m 16s 854ms 775us 807ns

10.3.2. IEC Timer TON (ON Delay) Pulse Diagram

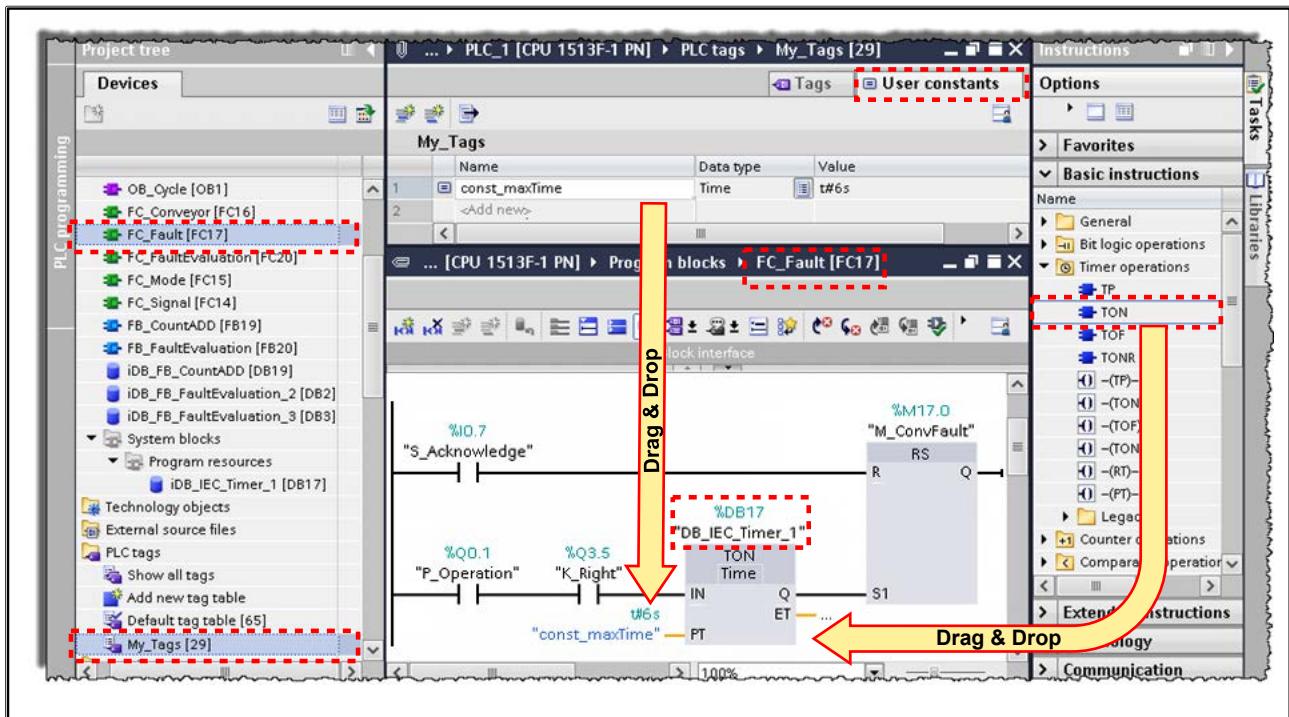


TON

The timer function TON (Timer on delay, "ON Delay") is started with a rising edge at input IN. So that the time expires, RLO must continue to be '1'. The timer function supplies a '1' signal at output Q, as soon as the specified time (variable or constant of data type TIME or LTIME) at input PT has expired and as long as the start signal at input IN still exists. The already expired time can be queried at output ET by passing a variable of data type TIME or LTIME.

Parameter	Data type	Memory area	Description
IN	BOOL	I, Q, M, D, L	Start input
PT	TIME	I, Q, M, D, L or constant	Duration by which the rising edge at input IN is delayed
Q	BOOL	I, Q, M, D, L	Output that is delayed by the time PT
ET	TIME	I, Q, M, D, L	Expired time

10.3.3. Exercise 3: Programming the Time Monitoring of the Transports in "FC_Fault"



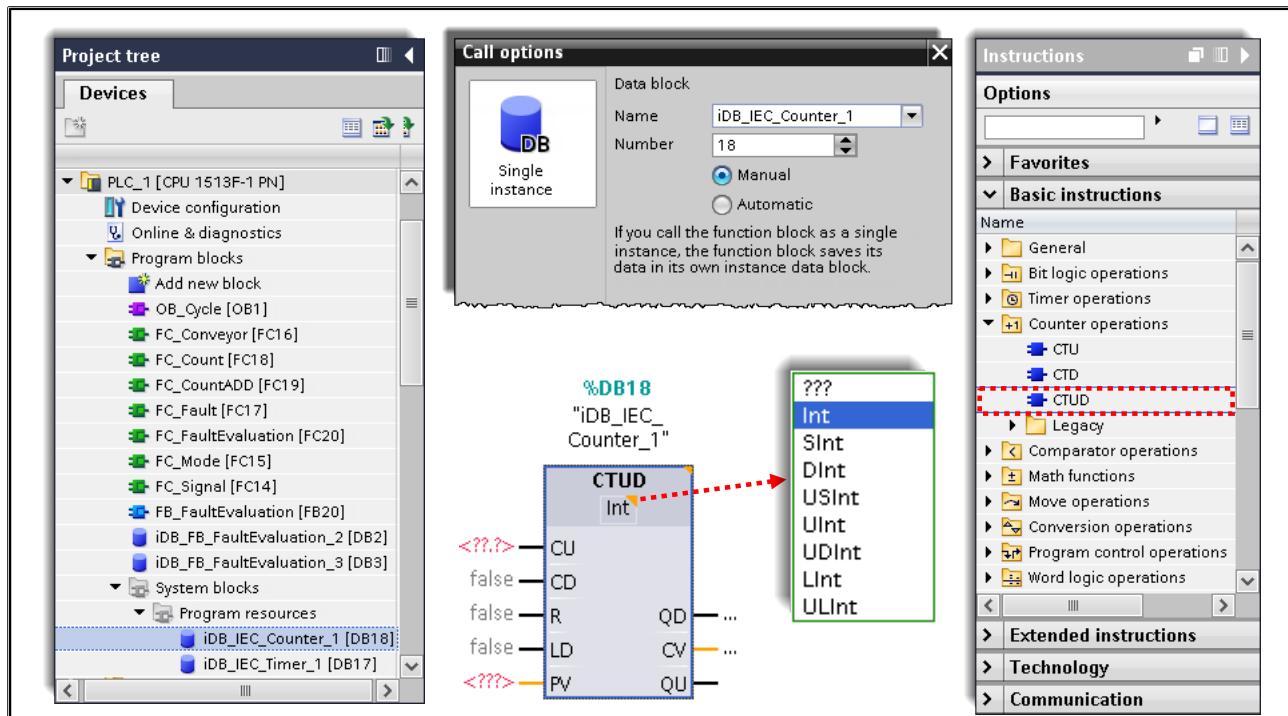
Task

The automatic transport sequences are to be monitored for time as previously described. If a transport sequence takes longer than 6 seconds, the conveyor motor is automatically switched off and the fault is displayed with a 2Hz flashing light on the simulator LED "P_Fault" (Q0.7). As long as a fault is not acknowledged "S_Acknowledge" (I 0.7) no new transport sequence can be started.

What to Do

- In the PLC tag table "My_Tags", declare the user constant "const_maxTime" of the data type Time with the value T#6s as shown in the picture.
- Expand the "FC_Fault" block with the necessary functions:
 - At input IN, program the relevant start conditions
(the timer must be started when an automatic transport sequence starts)
 - Pass the data block "iDB_IEC_Timer_1" as the instance-DB to the IEC timer function TON and as a time duration, the user constant "const_maxTime" (see picture).
 - In case the maximum transportation time is exceeded, set the memory bit "M_ConvFault" (M17.0), in order to be able to further logically link it in other blocks later on.
- In the "FC_Conveyor" block, program the required switching off the conveyor motor when there is a conveyor fault.
- In "FC_Signal", program the 2Hz flashing of the simulator LED "P_Fault" (Q0.7) and the lock-outs of the indicator lights "P_Bay1" (Q3.1) and "P_Bay2" (Q3.2) when a fault exists.
- Download all modified blocks into the CPU, check the program function.
- Save your project.

10.4. IEC Counters: CTU, CTD, CTUD



Counters

Counters are used to count events, record quantities, etc. There are up counters and down counters as well as counters that can count in both directions.

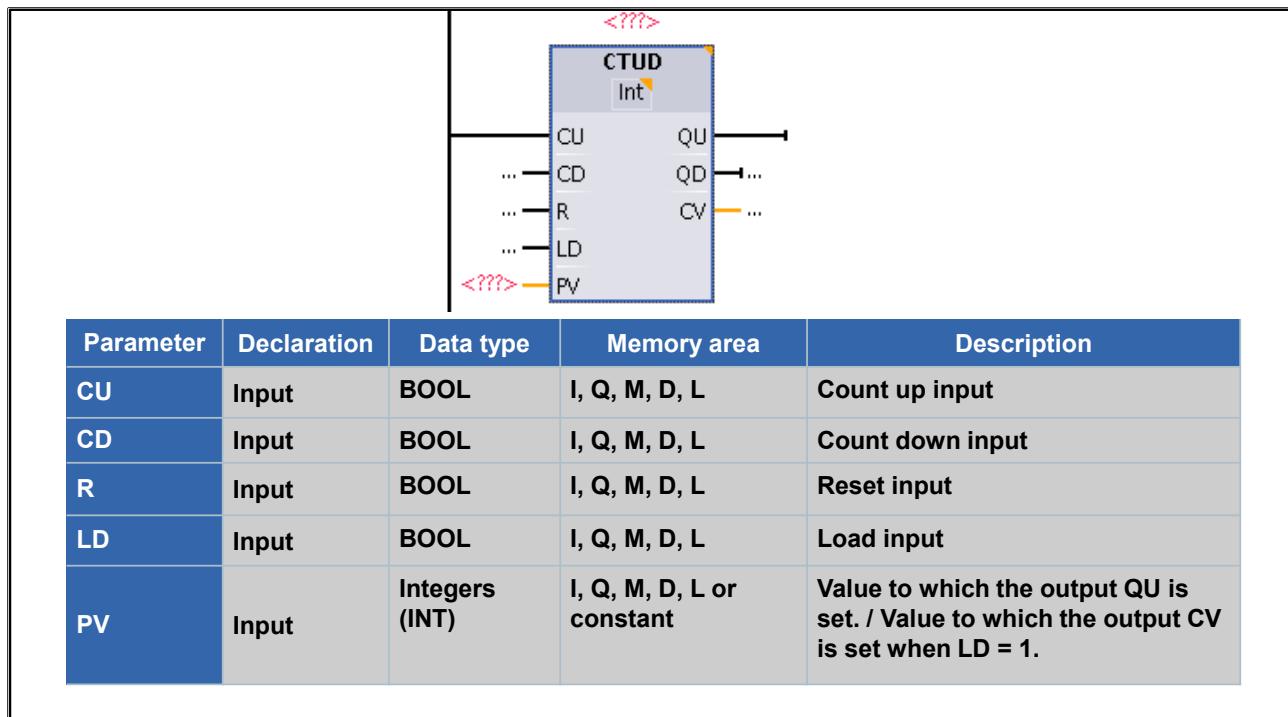
Value Range of a Counter

The count or value range of a counter depends on its data type (see picture) which is always an integer. The various selectable Integer data types merely differentiate themselves in their value range and thus determine the count range of the counter.

Instance Data Block

In addition to internally required variables, the counter also stores its current counter value in a so-called instance data block which must be specified when programming a counter. The specified instance data block is automatically generated by the Editor with exactly the internal structure that the counter requires. The user has no further programming effort with this data block other than having to download it into the CPU.

10.4.1. IEC Counters UP/DOWN: Inputs



Input CU and CD

With a positive edge at input CU, the current count is increased by one; with a positive edge at input CD, the current count is decreased by 1. This means that the user **doesn't** have to program an edge evaluation.

If a positive edge is detected at both inputs simultaneously or in the same cycle, the current count remains unchanged. If the upper or lower limit of the specified data type is reached, the count is no longer increased or decreased for a positive edge at CU or CD.

Input R

The input R acts statically, that is, as long as RLO '1' is at input R, the count is set to 0 and rising edges or RLO '1' at the inputs CU, CD and LOAD have no effect on the current count.

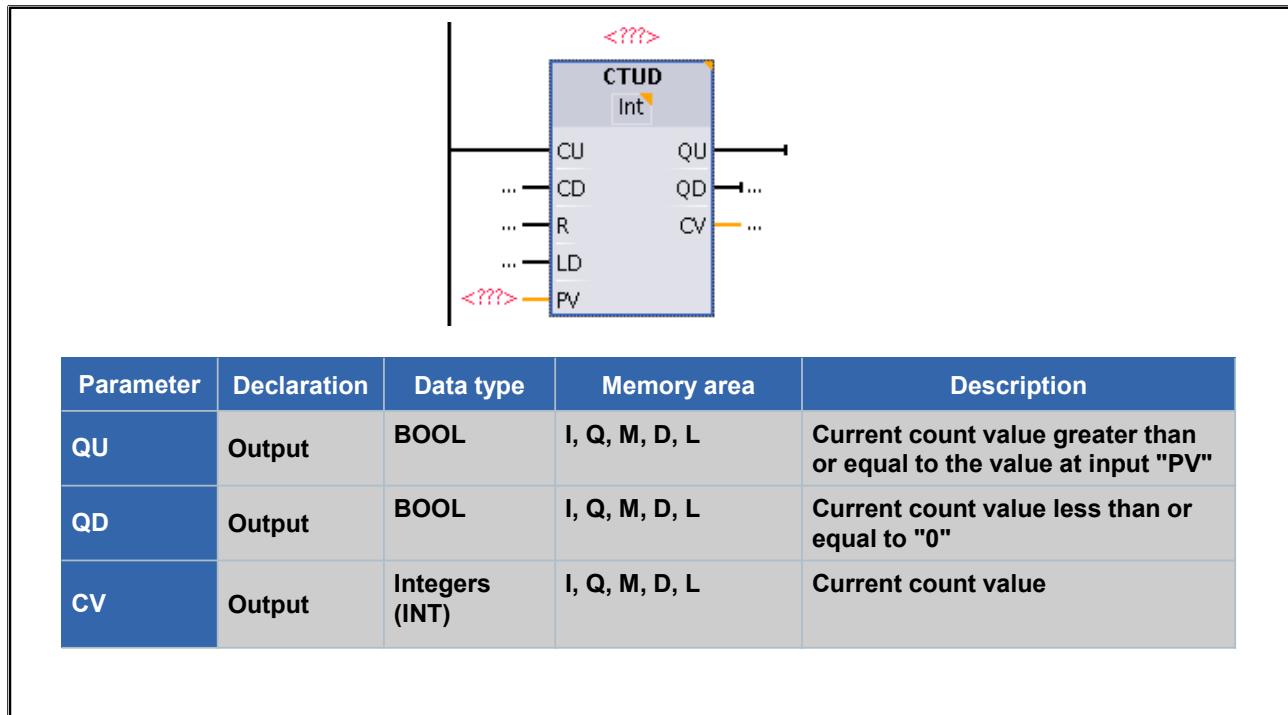
Input LD (Load, only for down counters)

The input LD acts statically, that is, as long as RLO '1' is at input LOAD, the current count is set to the value that is passed to the input PV, and positive edges at the inputs CU and CD have no effect on the count.

Input PV

The value to which the count is to be set must be passed to the input PV as long as RLO '1' is at input LD. The variable or constant passed to the input must be compatible with the data type of the counter.

10.4.2. IEC Counters UP/DOWN: Outputs



Output QU

The current status of the up counter can be checked at the output QU. As long as the current count is greater than or equal to the value of the parameter PV, the output QU has Status '1', otherwise, Status '0'.

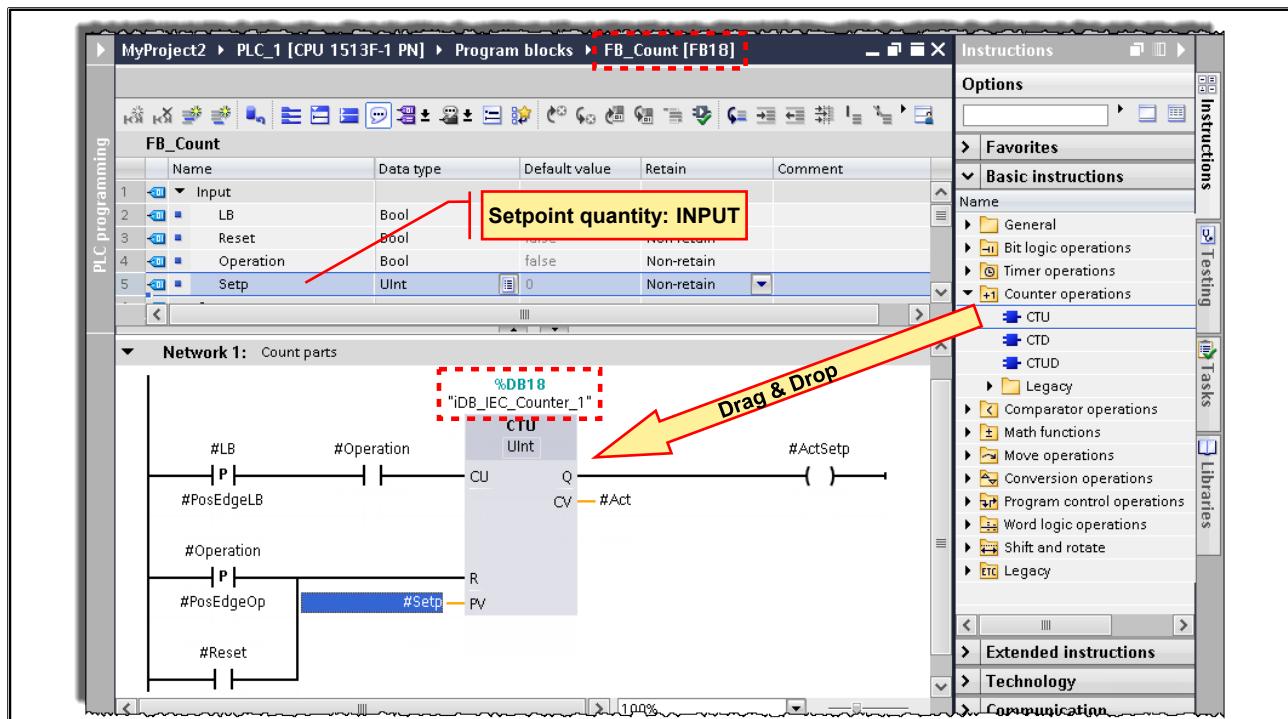
Output QD (Only for down counters)

The current status of the down counter can be checked at the output QD. As long as the current count is less than or equal to zero, the output QD has Status "1", otherwise, Status '0'.

Output CV

The current count is output at output CV. The variable passed to the output must be compatible with the data type of the counter.

10.4.3. Exercise 4: Counting the Transported Parts using an IEC Counter



Task

- When "P_Operation" (Q0.1) is switched on, the transported parts are to be counted as soon as they have passed the light barrier "B_LB" (I 3.0) ("B_LB" 0->1). The number of transported parts (ACTUAL quantity) is to be recorded with a counter and stored in the static variable #ACT.
- When the setpoint quantity (constant value 3) is reached, then no new transport sequence can take place which is also visible on the LEDs "P_Bay1" and "P_Bay2".
- The counter can be acknowledged (reset to 0) at any time via the pushbutton "S_BayLB" (I 3.4). When the operation is switched on, that is, with a positive edge at "P_Operation" (Q0.1), the ACTUAL quantity is also reset to 0.

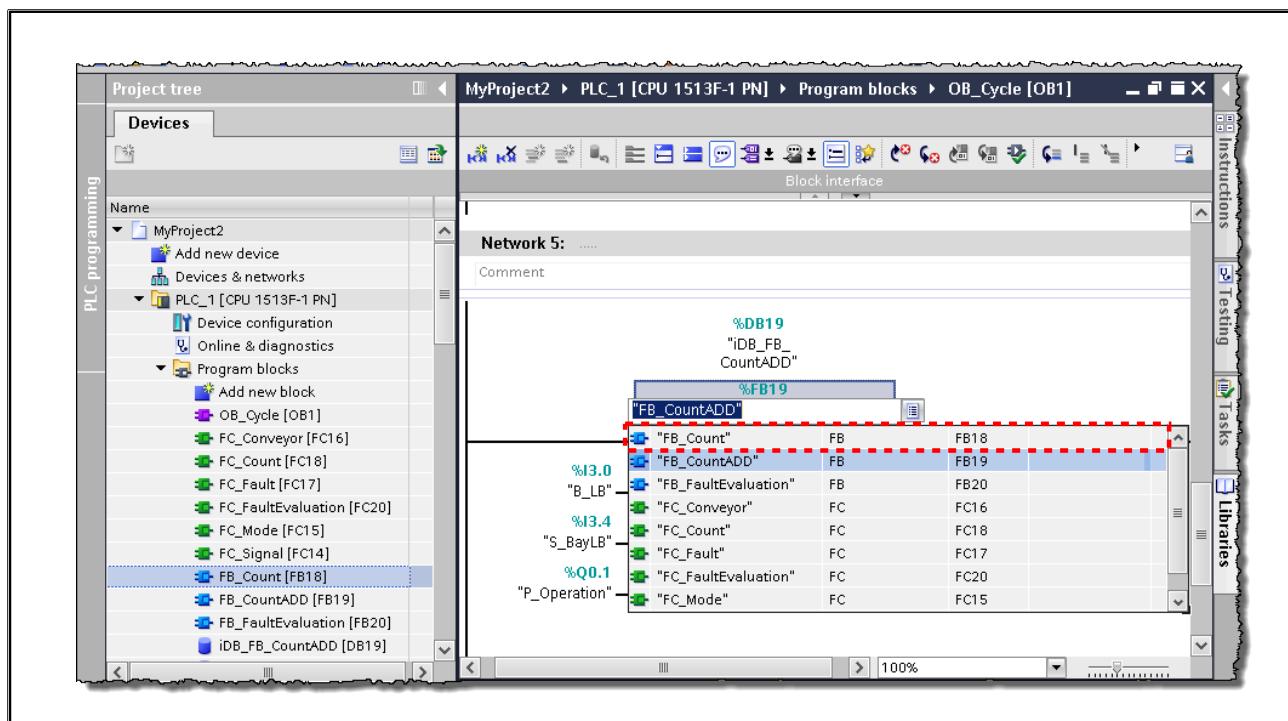
What to Do:

- Create the new function block "FB_Count".
- Declare the necessary parameters and variables:

Input:	- LB - Reset - Operation	BOOL, BOOL, BOOL,	Light barrier signal Reset the actual no. of parts to 0 Operation mode
Output:	- ActSetp	BOOL,	the actual no. of parts has reached the setpoint value
Static:	- PosEdgeLB - PosEdgeOp	BOOL, BOOL,	auxiliary bit for the edge evaluation of the Light barrier signal auxiliary bit for the edge evaluation of the Operation mode signal
Constant:	- Act - const_Setp	UINT, UINT,	actual no. of transported parts setpoint no. of parts to be transported

- Program the new "FB_CountADD" block with the appropriate arithmetic operations and don't use any global variables (tags) to do so. For the edge evaluation of the light barrier, use the static variable #PosEdgeLB and save the current quantity of the transported parts in the static variable #ACT.

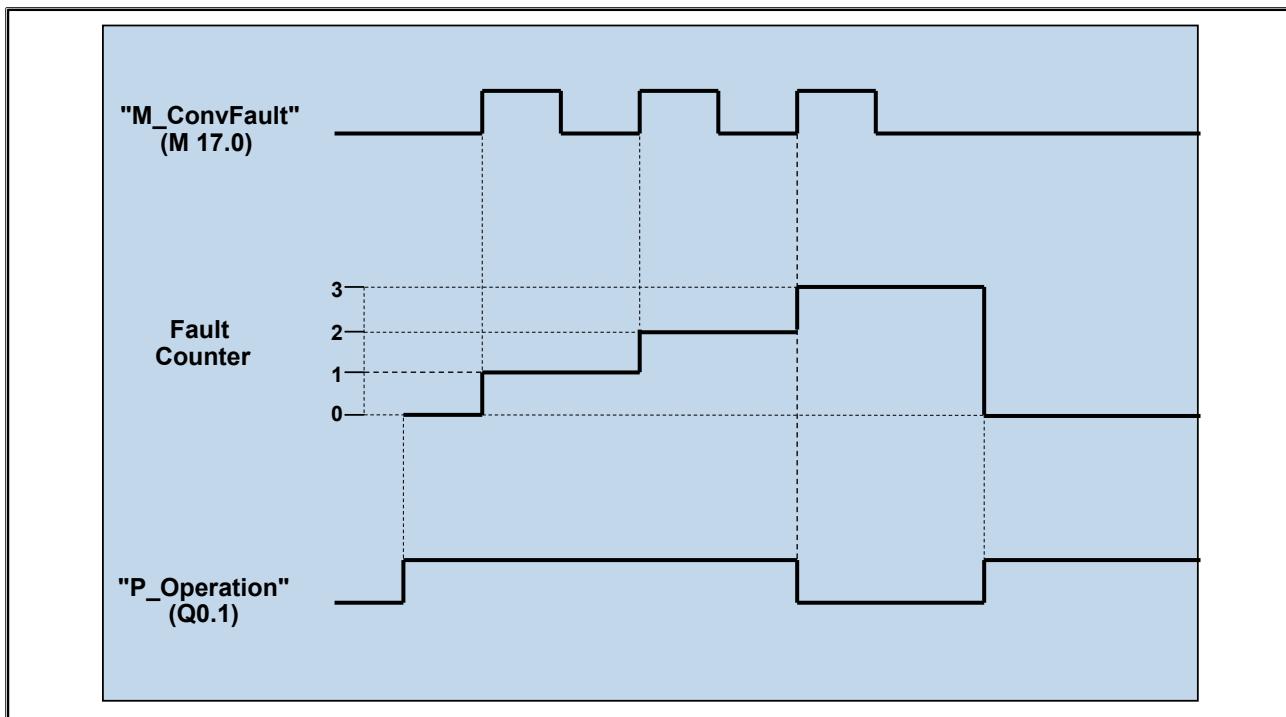
10.4.4. Exercise 5: Replacing the "FB_CountADD" Call with the "FB_Count" Call



What to Do

1. In the tag table "My_Tags", create a new tag "const_Setp" of the type UINT and give it the value '3'.
2. Replace the call of "FB_CountADD" with the call of "FB_Count".
For this use the function "Changing the block call" in which the actual parameters are adopted.
3. Create a new instance (iDB_FB_Count) for the block calls by right-clicking on the call of the FB and then selecting the context menu item "Change instance...".
4. Interconnect the new input formal parameter "Setp" with the new global constant "const_Setp"
5. Download all modified blocks into the CPU and check the program function.
6. Save your project.

10.5. Additional Exercise 6: Counting the Conveyor Faults by Expanding "FC_Fault"



Function Up Until Now

When "P_Operation" (Q0.1) is switched on, the transport sequences are monitored for time. If a transport sequence takes longer than the monitoring time of 6 seconds, there is a conveyor fault and the conveyor motor is automatically switched off (logically linked in "FC_Conveyor").

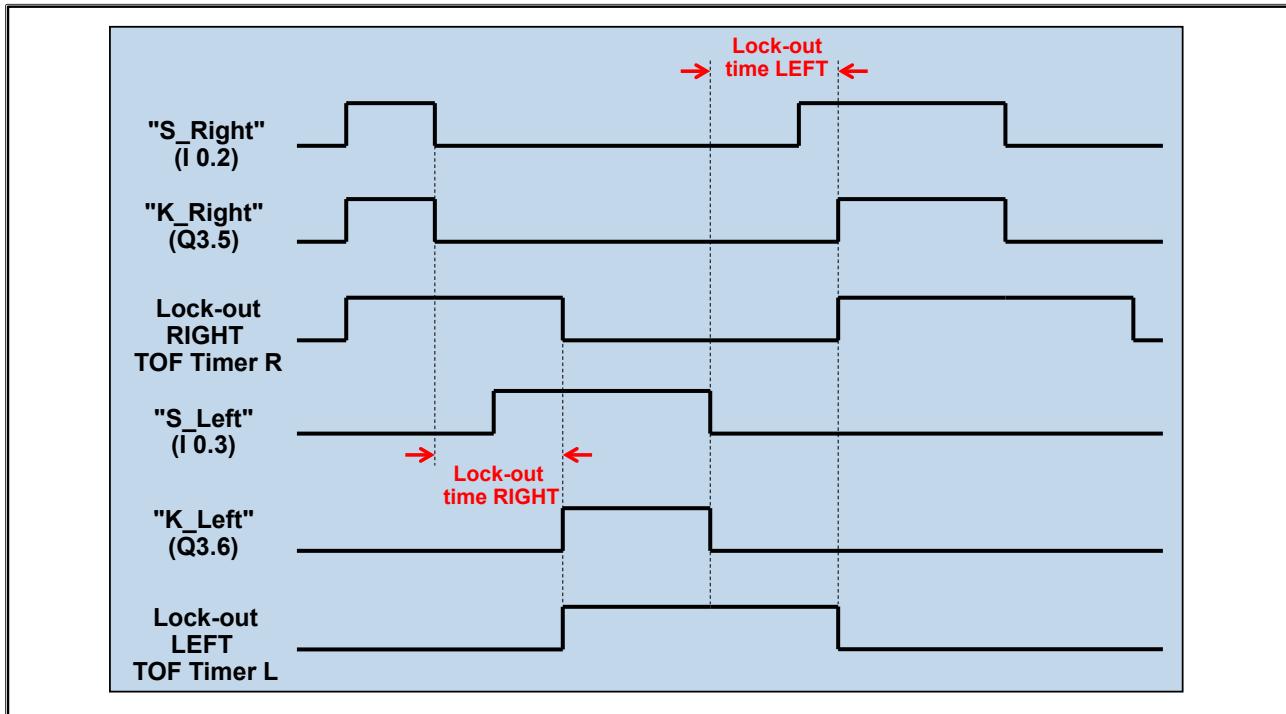
Task:

When "P_Operation" (Q0.1) is switched on, the conveyor faults are to be counted. After 3 conveyor faults have occurred, "P_Operation" (Q0.1) is to be switched off for safety reasons. To start a new transport sequence, the fault (as already programmed) must be acknowledged and "P_Operation" (Q0.1) must be switched on once again.

What to Do:

1. In "FC_Fault" in a new network, program the counting of the conveyor faults. The counter counts up '1' every time a conveyor fault occurs ("M_ConvFault" (M17.0) = "1").
2. In "FC_Fault", pass the value at counter output Q to any memory bit you like.
3. In "FC_Mode", program the switching off (reset) of "P_Operation" (Q0.1) after three conveyor faults. For this, use the memory bit to which you passed the counter output Q in "FC_Fault".
4. Download all modified blocks into the CPU and check the program function.
5. Save your project.

10.5.1.1. Additional Exercise 7: Timely Lock-out of the Conveyor Motor Jogging



Function Up Until Now

When the indicator light of "P_Operation" (Q0.1) is switched off, the conveyor motor can be jogged to the RIGHT and LEFT using the simulator switches "S_Right" (I 0.2) and "S_Left" (I 0.3).

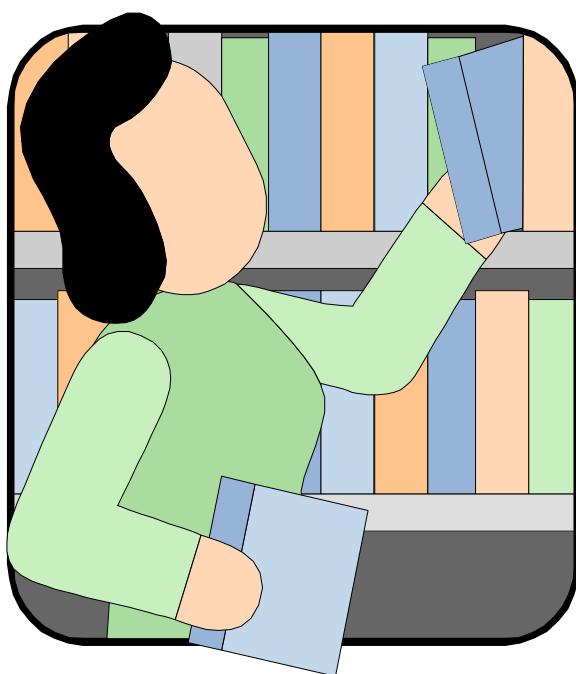
Task:

In order to avoid too great a load change, it should only be possible to jog the conveyor motor in the opposite direction after it has been jogged to the RIGHT or to the LEFT after a lock-out time of 2 seconds (see picture). If, for example, the motor has been jogged to the RIGHT, then it can only be jogged back to the LEFT after the lock-out time of 2 seconds has expired.

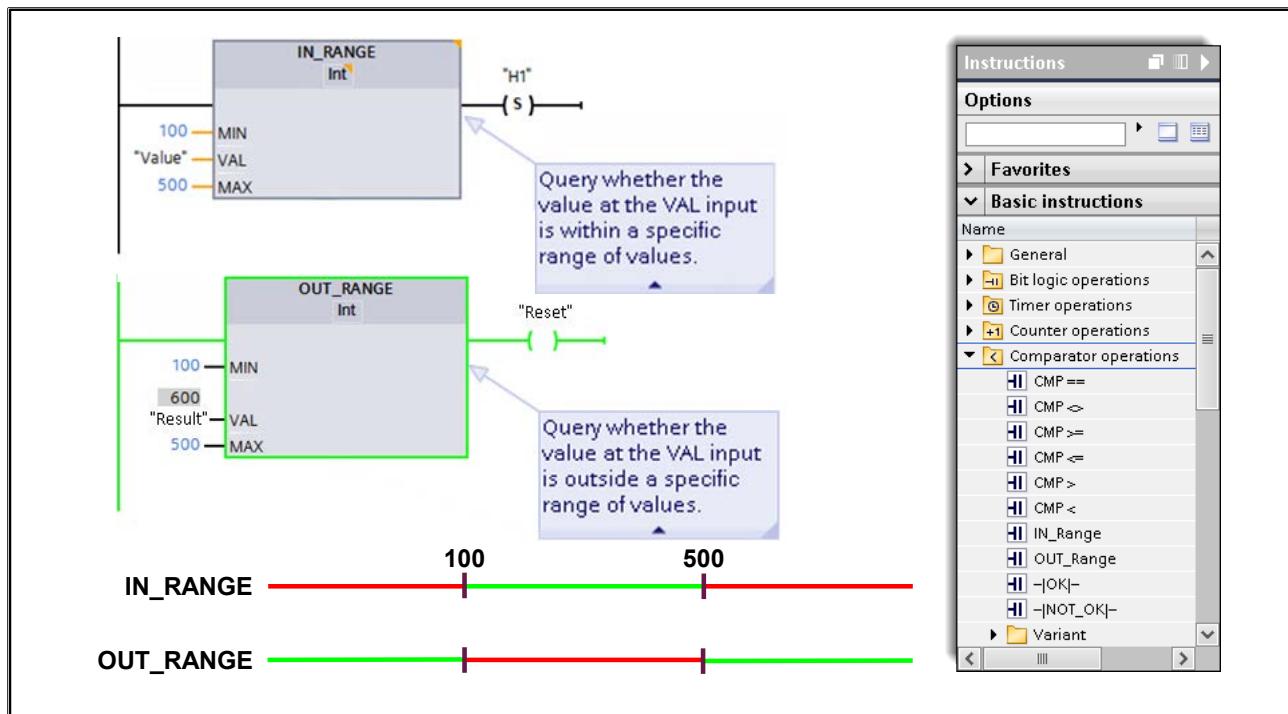
What to Do:

1. In "FC_Conveyor", program two TOF timers (Off Delay) as the lock-out timers RIGHT and LEFT and assign one memory bit each to the Timer result Q.
2. Gate these memory bits to the jog conditions.
3. Download the modified "FC_Conveyor" into the CPU and check the program function.

10.6. Additional Information



10.6.1. Comparator Operations: IN_RANGE, OUT_RANGE



IN_RANGE

You can use the "Value within range" instruction to determine if the value at the VAL input is within a specific value range. You specify the limits of the value range with the MIN and MAX inputs (parameters). If the value at the VAL input fulfills the comparison $\text{MIN} \leq \text{VAL}$ or $\text{VAL} \leq \text{MAX}$, the box output has the signal status "1". If the comparison is not fulfilled, the box output has the signal status "0".

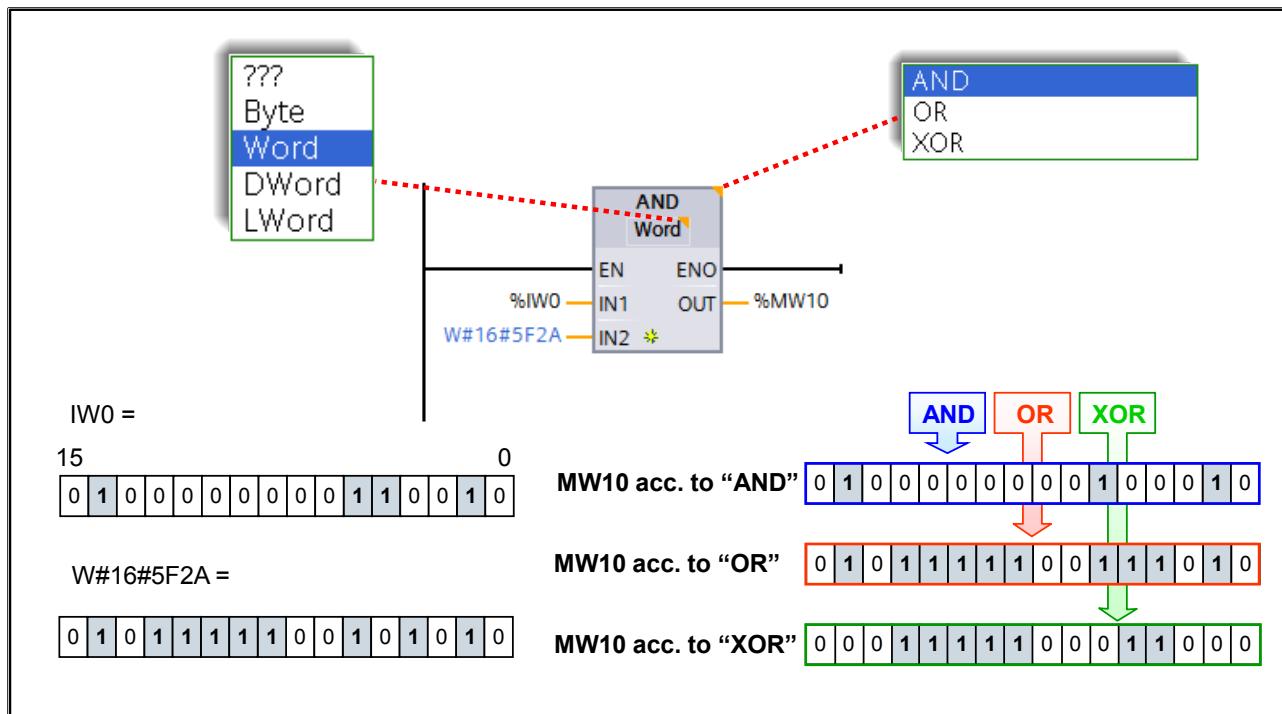
The comparison function can only be executed if the values to be compared are of the same data type.

OUT_RANGE

You can use the "Value outside range" instruction to determine if the value at the VAL input is outside of a specific value range. You specify the limits of the value range with the MIN and MAX inputs (parameters). If the value at the VAL input fulfills the comparison $\text{MIN} > \text{VAL}$ or $\text{VAL} > \text{MAX}$, the box output has signal status "1". If the comparison is not fulfilled, the box output has the signal status "0".

The comparison function can only be executed if the values to be compared are of the same data type.

10.6.2. Digital Logic Operations



AND

The "AND" instruction gates (combines) the two digital values at inputs IN1 and IN2 bit-by-bit in accordance with the AND truth table. The result of the AND operation is stored at output OUT. The instruction is executed when EN = 1.

Example: Masking out the 4th decade of the thumbwheel buttons:

IW2=	=	0100	0100	1100	0100
W#16#0FFF =		0000	1111	1111	1111
MW30 =		0000	0100	1100	0100

OR

The "Or" instruction gates (combines) the two digital values at inputs IN1 and IN2 bit-by-bit in accordance with the OR truth table. The result of the OR operation is stored at output OUT. The instruction is executed when EN = 1.

Example: Setting bit 0 in MW32:

MW32 =	0100	0010	0110	1010
W#16#0001 =	0000	0000	0000	0001
MW32 =	0100	0010	0110	1011

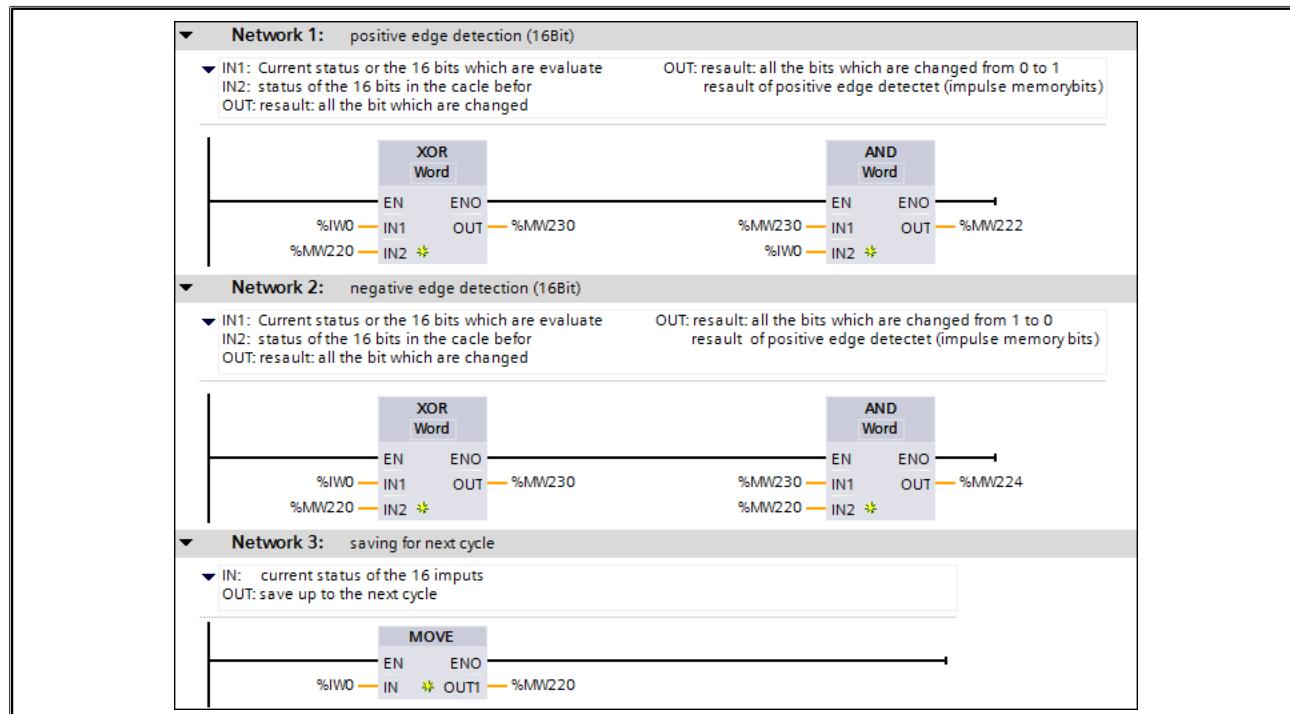
XOR

The "Exclusive OR" instruction gates (combines) the two digital values at inputs IN1 and IN2 bit-by-bit in accordance with the XOR truth table. The result of the XOR operation is stored at output OUT. The instruction is executed when EN=1.

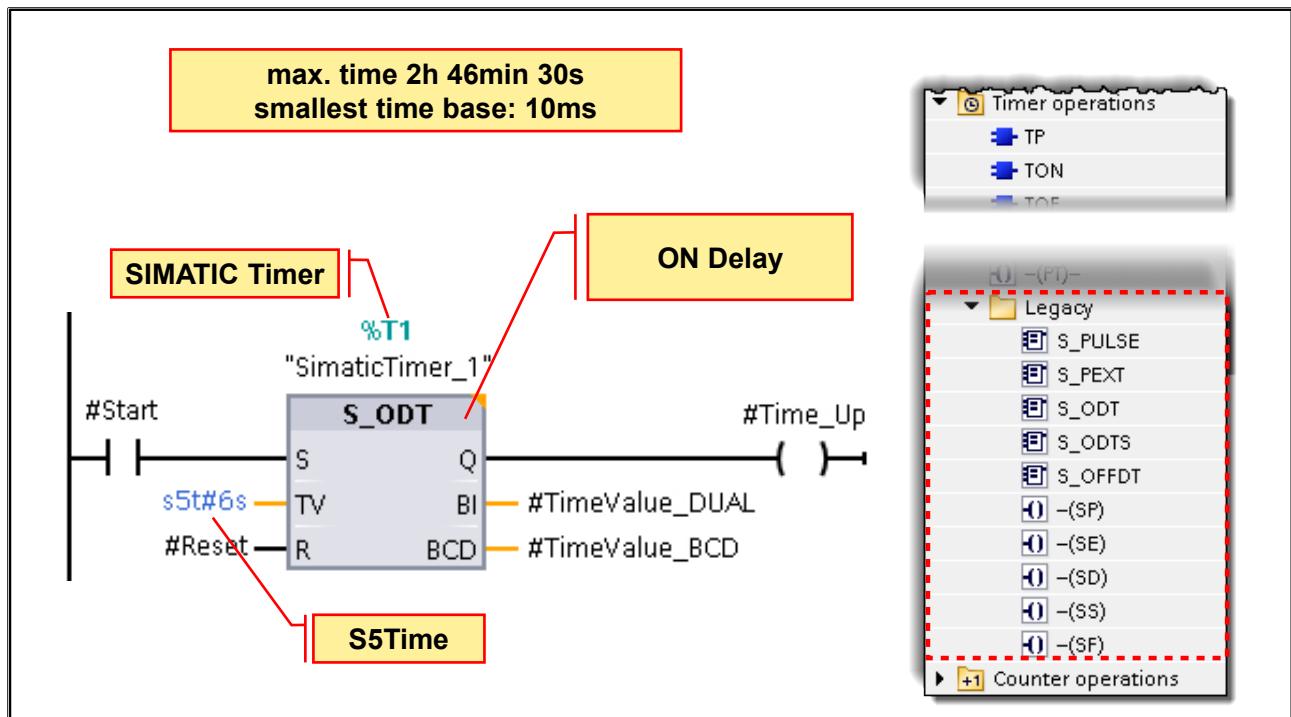
Example: Detecting signal changes in IW0:

IW0 =	0100	0100	1100	1010
MW28 =	0110	0010	1011	1001
MW24 =	0010	0110	0111	0011

10.6.2.1. Application Example: Digital Edge Evaluation



10.6.3. SIMATIC-Timer



For reasons of compatibility to STEP5, you have the possibility of using SIMATIC-Timers in an S7-1500. These timers are global. During the Start, the accuracy of the value range and what kind of timer it is (ON Delay, OFF Delay etc.) is defined.

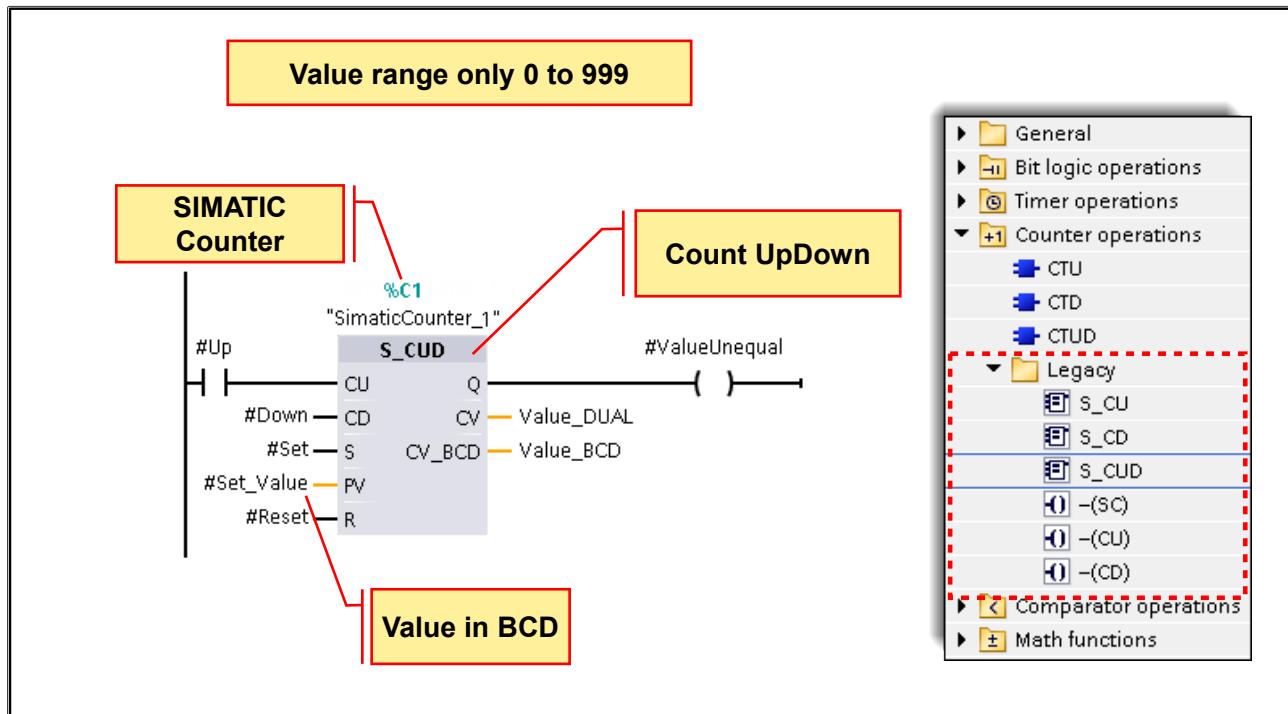
SIMATIC-Timers have the following system properties:

- They are assigned to a fixed number,
- They cannot be used repeatedly,
- The time frame lies between 0.01 and 9.99, 0.1 and 99.9, 1 and 999 or 10 and 9990s
- The accuracy depends on the selected time frame 10ms, 100ms, 1s or 10s

Note

In the time-cell (in the picture T1 of the memory area Timer), the operating system reduces the time value in an interval (time frame) defined by the time base by one unit respectively until the time value equals "0". The reduction occurs asynchronous to the user program. Consequently, the resulting time is maximally up to one time interval of the time base shorter than the desired time value.

10.6.3.1. SIMATIC-Counter



Just as with SIMATIC-Timers, SIMATIC-Counters can also be used for reasons of compatibility to STEP5. The counters are also global.

SIMATIC-Counters have the following system properties:

- They are assigned to a fixed number,
- They cannot be used repeatedly,
- The value range lies between 0 and max. 999

Note

If you use a SIMATIC-Counter, then only use it in one location in the program, in order to avoid count errors.

10.6.4. Value Ranges of Various Number Formats

Format	1 Bit	1 Byte	1 WORD	1 DWORD	1 LWORD
Binary number	2#0 or 2#1	2#0 to 2#1111_111	2#0 to 2#1111_1111_1111_1111	2#0 to 2#1111_1111_1111_1111_1111_1111_1111	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111
Octal number	8#0 or 8#1	8#0 to 8#377	8#0 to 8#177_777	8#0 to 8#37_777_777_777	8#0 to 8#1_777_777_777_777_777_777_777
Hexadecimal number (BCD each digit only 0-9)	16#0 or 16#1	16#0 to 16#FF BCD	16#0 to 16#FFFF	16#0 to 16#FFFF_FFFF	16#0 to 16#FFFF_FFFF_FFFF_FFFF
Decimal number	0 or 1	-128 to +127	-32 768 to +32 767	-2 147 483 648 to +2 147 483 648	-9 223 372 036 854 775 808 to +9 223 372 036 854 775 807
Decimal number without sign	0 or 1	0 to 255	0 to 65 535	0 to 4 294 967 295	0 to 18 446 744 073 709 551 615
Real number (Floating-point number)	/	/	/	-3.402823e+38 to -1.175495e-38 ±0,0 +1.175495e-38 to +3.402823e+38	-1.7976931348623158e+308 to -2.2250738585072014e-308 ±0,0 +2.2250738585072014e-308 to +1.7976931348623158e+308

11

Contents

11. Data Blocks.....	11-2
11.1. Data Blocks and their Usage	11-3
11.2. Meaning of Variables and Data Types.....	11-4
11.2.1. Overview: Data Types in STEP 7	11-5
11.2.1.1. Complex Data Types 1	11-6
11.2.1.2. Complex Data Types 2	11-8
11.3. Creating a Global Data Block.....	11-9
11.3.1. Editing a Data Block.....	11-10
11.3.2. Default, Start and Monitor Values	11-11
11.3.3. Retentiveness, Download DB into the CPU / Upload from the CPU	11-12
11.3.3.1. Downloading Changed Data Blocks into the CPU.....	11-13
11.3.4. Snapshot, Setpoint, Start Value.....	11-14
11.3.4.1. Initializing Setpoints Online.....	11-15
11.3.4.2. Changing the Snapshot / Start Value of Several / All Data Blocks.....	11-16
11.3.5. Copy & Paste from / to Microsoft Excel	11-17
11.4. Exercise 1: Creating Data Block "DB_OP"	11-18
11.4.1. Exercise 2: Adjusting "FB_Count" and Updating the Call.....	11-19
11.4.2. Exercise 3: Using DB Variables as Actual Parameters	11-20
11.5. Additional Information	11-21
11.5.1. Example of a Variable of the Data Type ARRAY.....	11-22
11.5.2. Example of a Variable of the Data Type STRUCTURE.....	11-23
11.5.3. PLC Data Type.....	11-24
11.5.4. Functions RD_SYS_T and RD_LOC_T	11-25

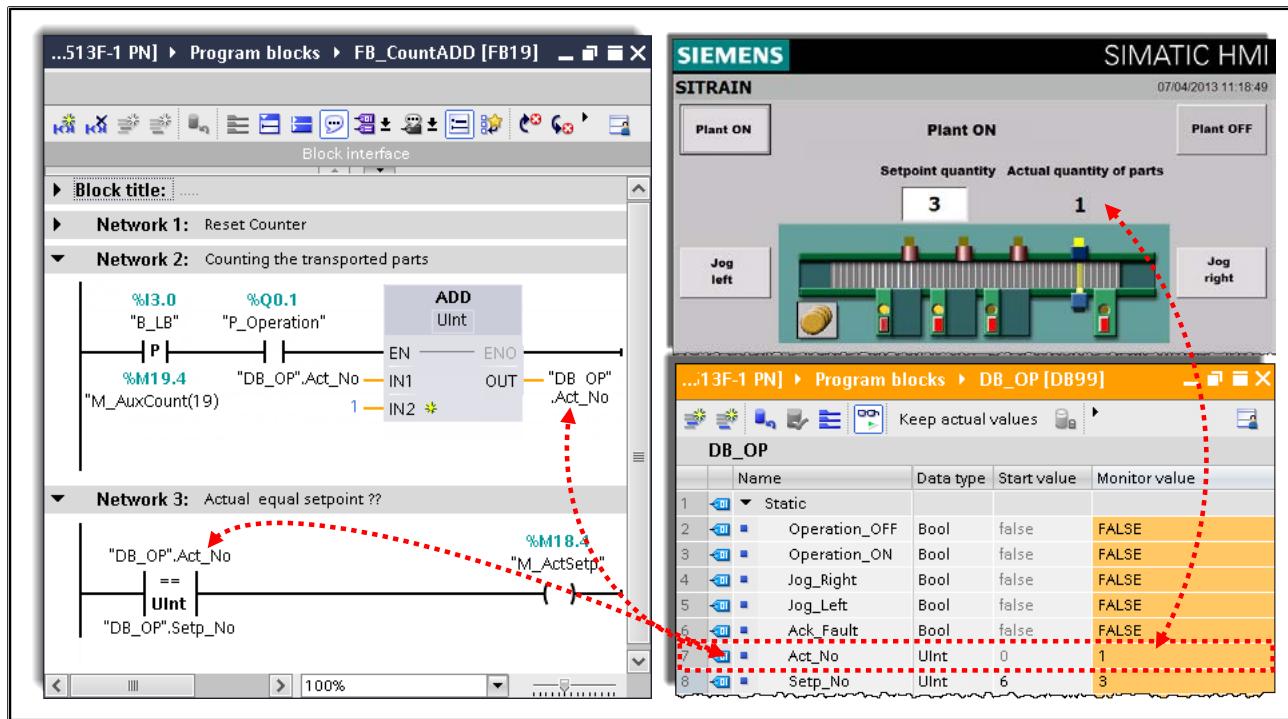
11. Data Blocks

At the end of the chapter the participant will ...

- ... understand the purpose of global data blocks
- ... be familiar with complex data types
- ... be able to create and edit global data blocks and use them in the program
- ... be familiar with the possibilities for addressing data block variables
- ... be able to monitor and initialize a data block
- ... be familiar with the difference between default values, start values, monitoring values, setpoints and snapshots



11.1. Data Blocks and their Usage



Data blocks contain variables for storing user data and accordingly occupy memory space in the work memory of the CPU.

Area of Application

You can use data blocks in different ways, depending on their contents. You differentiate between:

- Global data blocks: These contain information that all the code (logic) blocks in the user program can access. Often, global data blocks are also used as an interface to HMI devices (operating and monitoring devices).
- Instance data blocks: These are always assigned to a particular FB. The data of these instance DBs should only be processed by the associated FB.

Creating DBs

Global DBs are created either with the Program Editor or according to a previously created PLC data type. Instance data blocks are generated by the Editor according to a function block.

11.2. Meaning of Variables and Data Types

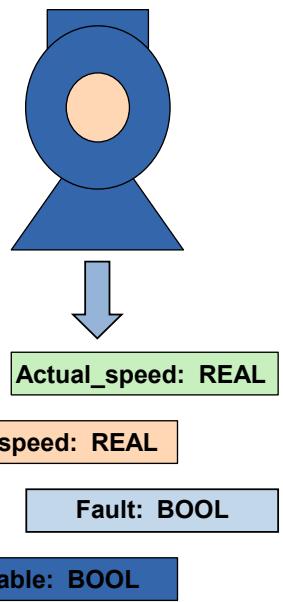
Variables represent the abstraction of reality and permit you to save and later continue to process values.

By declaring a variable, the following properties are defined:

- Symbolic name
- Memory area
- Validity range
- Data type

The data type establishes:

- The possible value range
(e.g. INT: -32 768 to +32 767)
- The permitted instructions
(e.g. math instructions: +I, -I)
- How the bits in the memory are to be interpreted
(integer; hexadecimal number; floating-point number; etc.)



The Meaning of Variables

Next to commands, variables are the most important elements of a programming system. Their task is to save values in a program so that they can be further processed at a later time. The value of a variable can be saved "anywhere" in the PLC memory.

The data represents an abstraction of reality in which irrelevant properties of objects are ignored.

Data Types

It is often quite difficult to decide how data is to be represented and the available possibilities quite often restrict the choice. On the one hand, the object properties the data describe must be correctly reflected. On the other hand, it must also be possible to carry out the instructions necessary for process with the data.

The data type determines which values are accepted by data and which instructions can be carried out with these values.

The data type uniquely defines

- the possible value range
- the permitted instructions
- how the bit pattern is to be interpreted

11.2.1. Overview: Data Types in STEP 7

	Type	Data types
Elementary Data types	Binary number	BOOL
	Bit sequences	BYTE; WORD; DWORD; LWORD
	Integers	SINT; USINT; INT; UNIT; DINT; UDINT; LINT ULINT
	Floating-point numbers	REAL; LREAL
	Timers	S5TIME; TIME; LTIME
	Date, Time-of-day	DATE; TIME_OF_DAY; LTIME_OF_DAY; LDT(DATE_AND_LTIME);
	Characters	CHAR; WCHAR
Complex Data types	Date, Time-of-day	DT(DATE_AND_TIME); DTL;
	Character string	STRING; WSTRING
	Array	ARRAY [...] of <Datatype>
	Anonymous Structure	STRUCT
	User-defined	PLC-data type (User Defined Data Type)

Elementary Data Types

Elementary data types are predefined in accordance with IEC 61131-3. They always have a length less than or equal to 64 bits.

Complex Data Types

Complex data types contain data structures that can be made up of elementary and/or complex data types. Complex data types can be used for the declaration of variables only in global data blocks and within blocks for the declaration of local variables (TEMP, STAT) as well as parameters (IN, OUT and INOUT).

11.2.1.1. Complex Data Types 1

Data type	Length (Bits)	S7-1200	S7-1500	Example
DT (DATE_AND_TIME)	64	✗	✓	DT#2008-10-25-08:12:34.567
DTL	96	✓	✓	DTL#1976-12-16-20:30:20.250
STRING	8 * (Number of characters+2)	✓	✓	'This is a String' max. 254 characters in ASCII format
WSTRING (Wide Character String)	16 * (Number of characters+2)	✓	✓	WSTRING#'STRING in UNICODE format' Up to 16382 characters in Unicode format

DT

The data type DATE_AND_TIME represents a point in time consisting of the date and the time-of-day. Instead of DATE_AND_TIME, the abbreviation DT can also be used.

Byte	Contents	Range of values
0	Year	0 to 99 (Years 1990 to 2089) BCD#90 = 1990 ... BCD#0 = 2000 ... BCD#89 = 2089
1	Month	BCD#0 to BCD#12
2	Day	BCD#0 to BCD#31
3	Hour	BCD#0 to BCD#23
4	Minute	BCD#0 to BCD#59
5	Second	BCD#0 to BCD#59
6	The two most significant digits of MSEC	BCD#0 to BCD#999
7 (4MSB) 1)	The least significant digit of MSEC	BCD#0 to BCD#9
7 (4MSB) 2)	Weekday	BCD#1 to BCD#7 BCD#1 = Sunday ... BCD#7 = Saturday

DTL

The data type DTL has a length of 12 bytes and, like LDT, stores information on date and time-of-day precise to the nanosecond since 1.1.1970, only in a pre-defined structure.

Advantage: the individual values (day, hour, etc.) are easier to read out.

Byte	Component	Data type	Range of values
0 - 1	Year	UINT	1970 to 2554
2	Month	USINT	1 to 12
3	Day	USINT	1 to 31
4	Weekday	USINT	1 (Sunday) to 7 (Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8 -11	Nanosecond	UDINT	0 to 999 999 999

STRING

The data type STRING stores several ASCII characters of the data type CHAR in a character string with a maximum of 254 characters.

WSTRING

The data type WSTRING (Wide Character String) stores several Unicode characters of the data type WCHAR in a character string with a maximum of 16382 characters.

11.2.1.2. Complex Data Types 2

Data type	Length (Bits)	S7- 1200	S7- 1500	Example				
ARRAY	User-defined	✓	✓	Measured values: ARRAY[1..20] of INT;				
STRUCT	User-defined	✓	✓	<p>Motor: STRUCT</p> <pre>ACT-Speed : REAL; Set-Speed : UINT; Fault : BOOL; Enable : BOOL; END_STRUCT</pre>				
PLC-Data type (UDT)	User-defined	✓	✓	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">Define (in the PLC-data types folder)</td> <td style="padding: 5px;">Use (in data blocks and interfaces)</td> </tr> <tr> <td style="padding: 5px;"> <pre>PLC-DT1 : STRUCT Speed : REAL; Enable : BOOL; Fault : BOOL; END_STRUCT</pre> </td> <td style="padding: 5px;"> <pre>Motor1 : PLC-DT1; Motor2 : PLC-DT1; Motor3 : PLC-DT1; :</pre> </td> </tr> </table>	Define (in the PLC-data types folder)	Use (in data blocks and interfaces)	<pre>PLC-DT1 : STRUCT Speed : REAL; Enable : BOOL; Fault : BOOL; END_STRUCT</pre>	<pre>Motor1 : PLC-DT1; Motor2 : PLC-DT1; Motor3 : PLC-DT1; :</pre>
Define (in the PLC-data types folder)	Use (in data blocks and interfaces)							
<pre>PLC-DT1 : STRUCT Speed : REAL; Enable : BOOL; Fault : BOOL; END_STRUCT</pre>	<pre>Motor1 : PLC-DT1; Motor2 : PLC-DT1; Motor3 : PLC-DT1; :</pre>							

Arrays and Structures consist of groups of elementary or complex data types.

They enable you to create data types suitable for your task with which you can structure large quantities of data and process it symbolically.

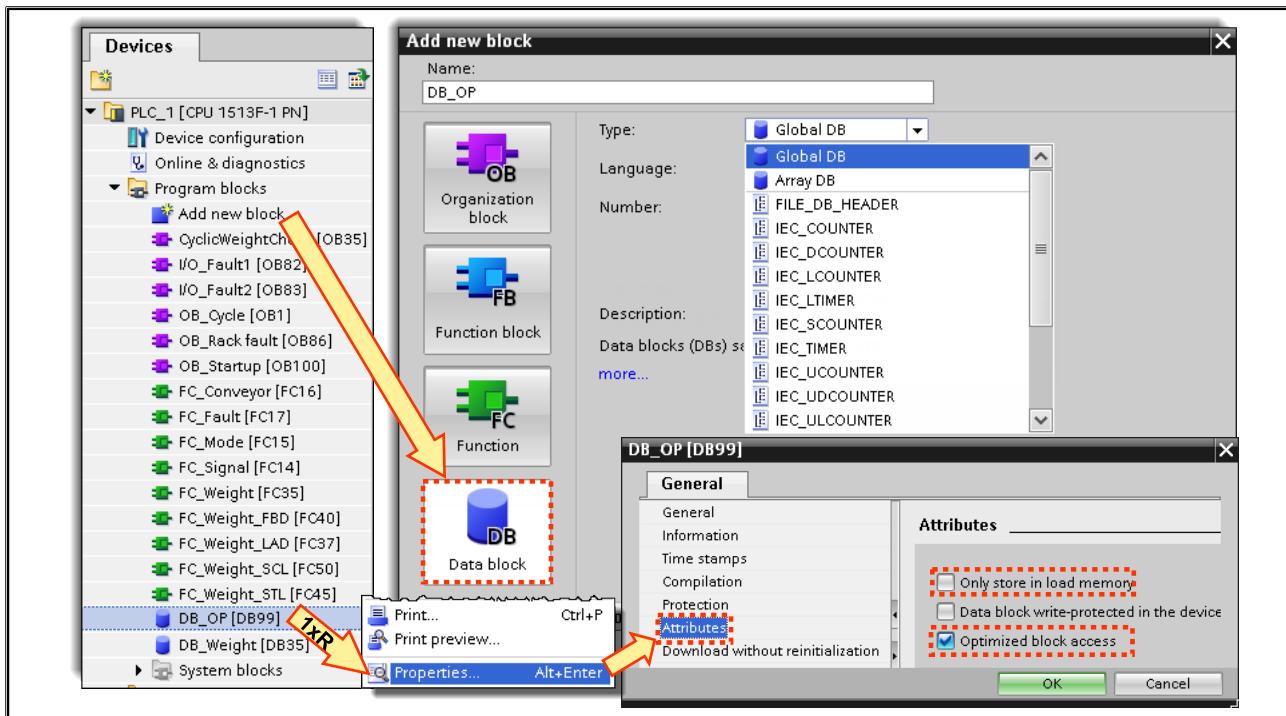
ARRAY and STRUCT

ARRAYs and Structures can only be declared within global data blocks and as parameters or local variables of code blocks.

PLC-Data Type (User-defined Data Type)

User-defined data types represent self-defined structures. This structure is stored in the PLC data types and can be used as a "template" in another variable's data type.

11.3. Creating a Global Data Block



Creating a Global Data Block

(Global) data blocks are created in exactly the same way as code (logic) blocks.

In creating a data block you can select of which type the data block is to be or for what purpose it is to be used:

- Global-DB for saving global data or for creating global variables
- Instance-DB or "private memory" for a user function block or a particular "instruction", behind which a function block (FB) is also ultimately hidden

"Optimized Block Access" Attribute

Data blocks can be created with the attribute "Optimized block access":

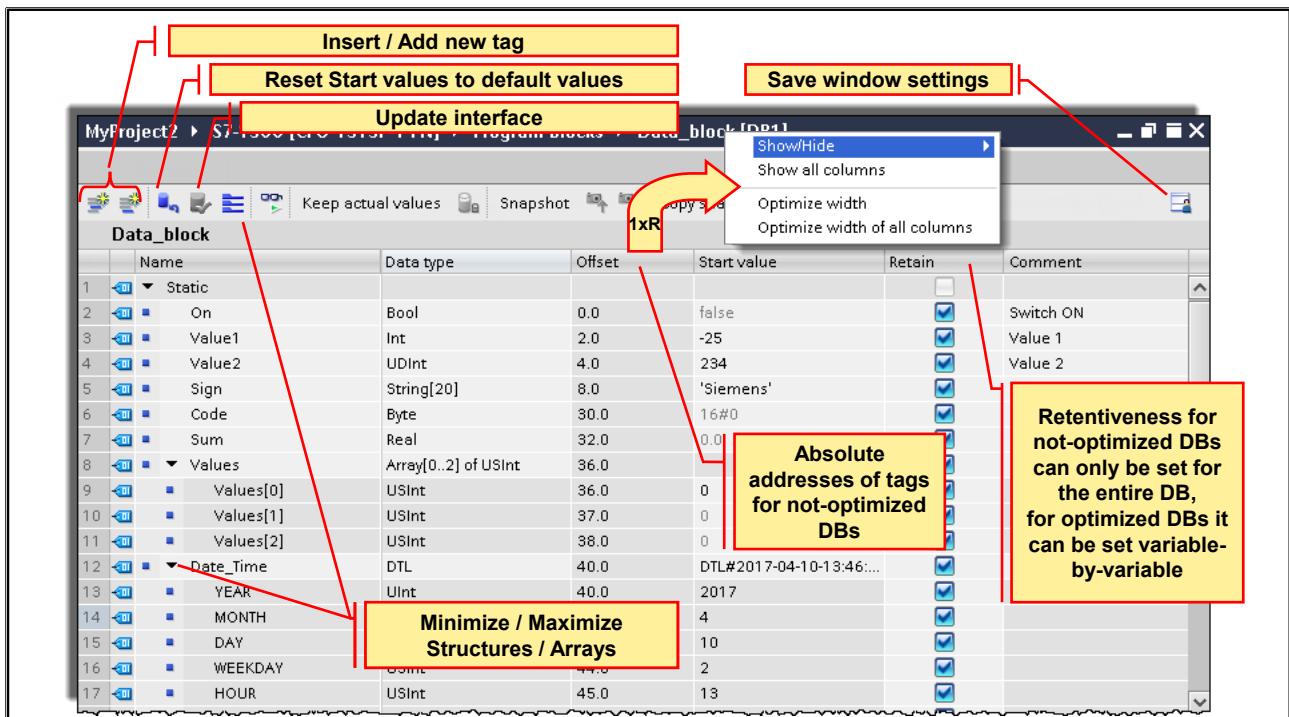
- In S7-1200, these data blocks are created memory-optimized, that is, the variables are stored in sequence so that fill bytes, through the changing, successive creating of variables of the dimensions bit, byte word and double-word, are no longer required.
- In S7-1500, these data blocks are created access-optimized taking the S7 machine code into consideration, so that – regardless of the memory requirements of the variables – the access times to these are minimized.

Optimized Block Access see: TIA Portal Information Center > Documentation > Manuals > Control Technology > Programming Guideline for an optimal programming of SIMATIC S7-1200/1500 controllers

"Only Store in Load Memory" Attribute

This attribute means that in downloading into the CPU, the data block is only loaded into the load memory of the CPU and from there is not automatically adopted in the work memory.

11.3.1. Editing a Data Block



Offset

The offset denotes the absolute address of a variable within a DB. In the STEP 7 program, the use of the symbolic address or name is preferable since it is easier to read and less prone to errors than the absolute addressing.

The absolute addresses of variables within the data block are not displayed for optimized blocks and can also not be read out.

Retentiveness

For not-optimized blocks, the retentive behavior cannot be defined for the individual variables, but always only for all variables or the entire data block.

When "Retain" is activated, the monitoring values are retained in the CPU until the data block is initialized.

When "Retain" is not activated, the monitoring values in the work memory are overwritten with the start values from the load memory after every STOP - RUN - transition (by PG, mode selector switch or Power OFF->ON) of the CPU.

11.3.2. Default, Start and Monitor Values

The screenshot shows a SIMATIC TIA Portal interface for a Data_block [DB1]. The table has columns: Name, Data type, Offset, Default value, Start value, Snapshot, Monitor value, and Setpoint. Red boxes highlight specific rows:

- Current value of variable in the CPU (online)**: Points to the first row.
- Snapshot of current value (offline)**: Points to the second row.
- Start value of variable**: Points to the third row.
- Default start value**: Points to the fourth row.

Data_block (snapshot created: 4/21/2017 2:06:20 PM)

	Name	Data type	Offset	Default value	Start value	Snapshot	Monitor value	Setpoint
1	Static							
2	On	Bool	0.0	false	false	FALSE	TRUE	
3	Value1	Int	2.0	0	-25	456	543	
4	Value2	UDInt	4.0	0	234	0	0	
5	Sign	String[20]	8.0	"	'Siemens'	'Siemens'	'Siemens'	
6	Code	Byte	30.0	16#0	16#0	16#3F	16#23	<input checked="" type="checkbox"/>
7	Sum	Real	32.0	0.0	0.0	0.0	345.23	<input checked="" type="checkbox"/>
8	Values	Array[0..2] of USInt	36.0					<input checked="" type="checkbox"/>
9	Values[0]	USInt	36.0	0	0	234	234	<input checked="" type="checkbox"/>
10	Values[1]	USInt	37.0	0	0	0	23	<input checked="" type="checkbox"/>
11	Values[2]	USInt	38.0	0	0	0	45	<input checked="" type="checkbox"/>
12	Date_Time	DTL	40.0	DTL#1970-01-01 00:00:00	DTL#2017-04-10-10:00:00	DTL#2017-04-10-15:00:00	DTL#2017-04-12-13:00:00	
13	YEAR	UInt	40.0	1970	2017	2017	2017	
14	MONTH	UInt	42.0	1	4	4	4	
15	DAY	UInt	43.0	1	10	10	10	

Default Value

Default values cannot be edited within global data blocks, only within PLC data types (the default values of structure elements) and within FBs (the default values of parameters and static variables).

If, within a global DB, a variable of the data type Structure is declared according to the PLC data type x, the default values edited in the PLC data type are then displayed (as not editable) in the data block and adopted as (editable) start values.

Within an FB, their default values also have to be specified in the declaration of parameters and static variables. In the instance DBs of the FB, these default values are then displayed (as not editable) and adopted as (editable) start values.

Start Value

In declaring a variable, the default value of a variable is automatically adopted as a start value; however, this value can be overwritten at any time and downloaded into the controller. After a data block was loaded into the CPU the first time, the CPU starts the program execution with exactly this value. For all non-retentive variables, this value is written in the Monitor value with every STOP-RUN-transition or for with an initialization; for retentive variables, only for a re-initialization.

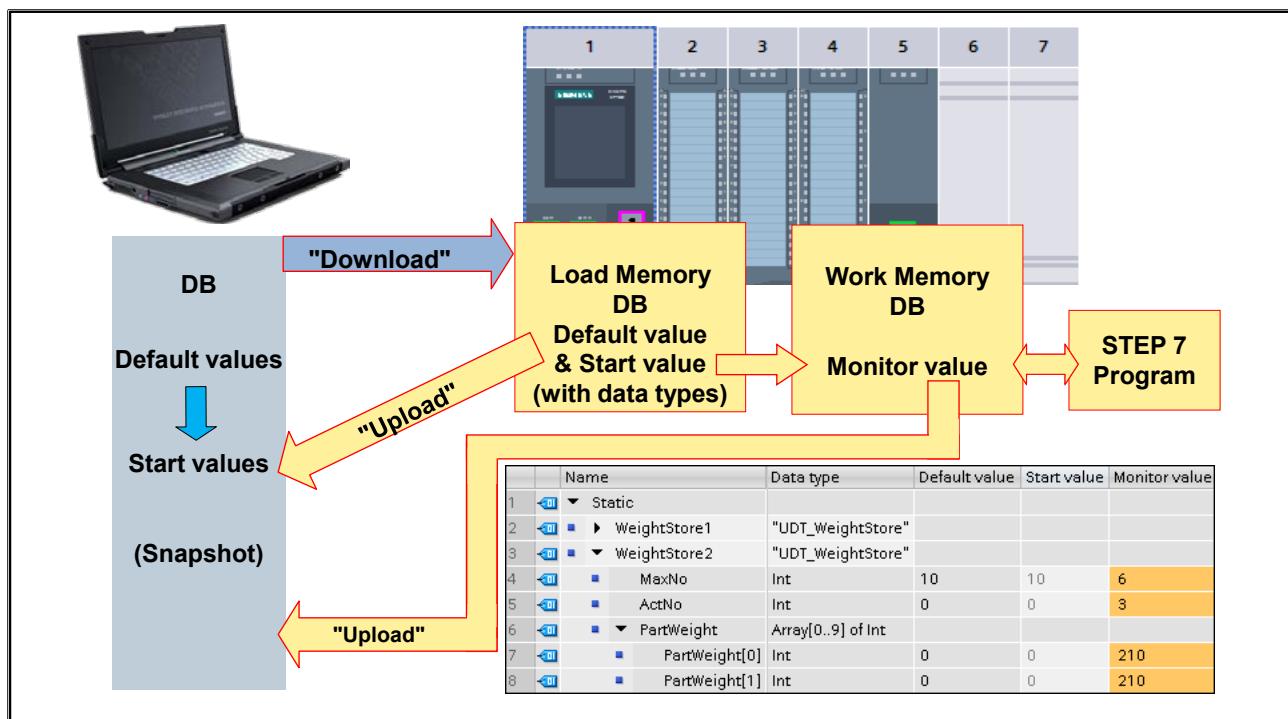
Monitoring Value

The monitoring value of a variable is the current value that the variable presently has in the work memory of the CPU.

Snapshot

The value "Snapshot" is a monitoring value for an already passed point in time x, at which the current monitoring values are read out of the CPU and stored offline as the values of the "Snapshot".

11.3.3. Retentiveness, Download DB into the CPU / Upload from the CPU



"Download" (to Device)

In downloading a data block from the PG into the CPU, the default values and start values as well as the data types of the variables are transferred into the load memory of the CPU. From the load memory, the start values are then automatically applied as monitoring values in the work memory of the CPU. The STEP 7 program works with the monitoring values of the work memory.

"Upload" (from Device)

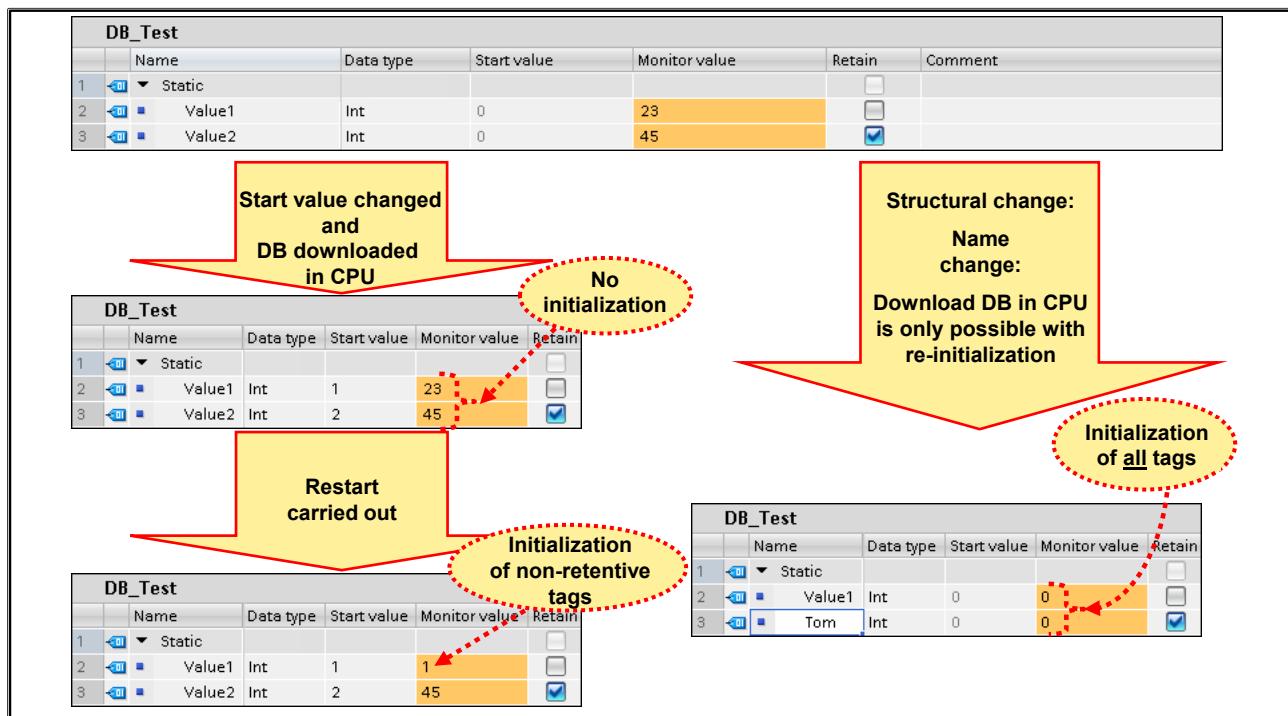
In uploading a data block from the CPU into the PG, the start values from the load memory and the current monitoring values from the work memory are transferred into the PG. In doing so, the "snapshot" values stored offline are overwritten with the monitoring values loaded from the CPU. The values from this "snapshot" can then be applied offline in the project as new "start values".

Retentiveness (Retain)

With retentive variables, the "monitor values" (actual values) are retained after a CPU restart.

Non-retentive variables occupy work memory but no retentive memory and are thus reset to the start values from the load memory with every CPU restart (Power→Off→On or after every STOP - RUN).

11.3.3.1. Downloading Changed Data Blocks into the CPU



Downloading Data Blocks with Re-initialization

No matter whether a DB was created with standard or optimized block access, a change of start values is not a structural change. Therefore, it can be done and downloaded into the CPU without a re-initialization of the data block variables (tags) being necessary.

On the other hand, after structural changes, it is only possible to download the data block by re-initializing all tags.

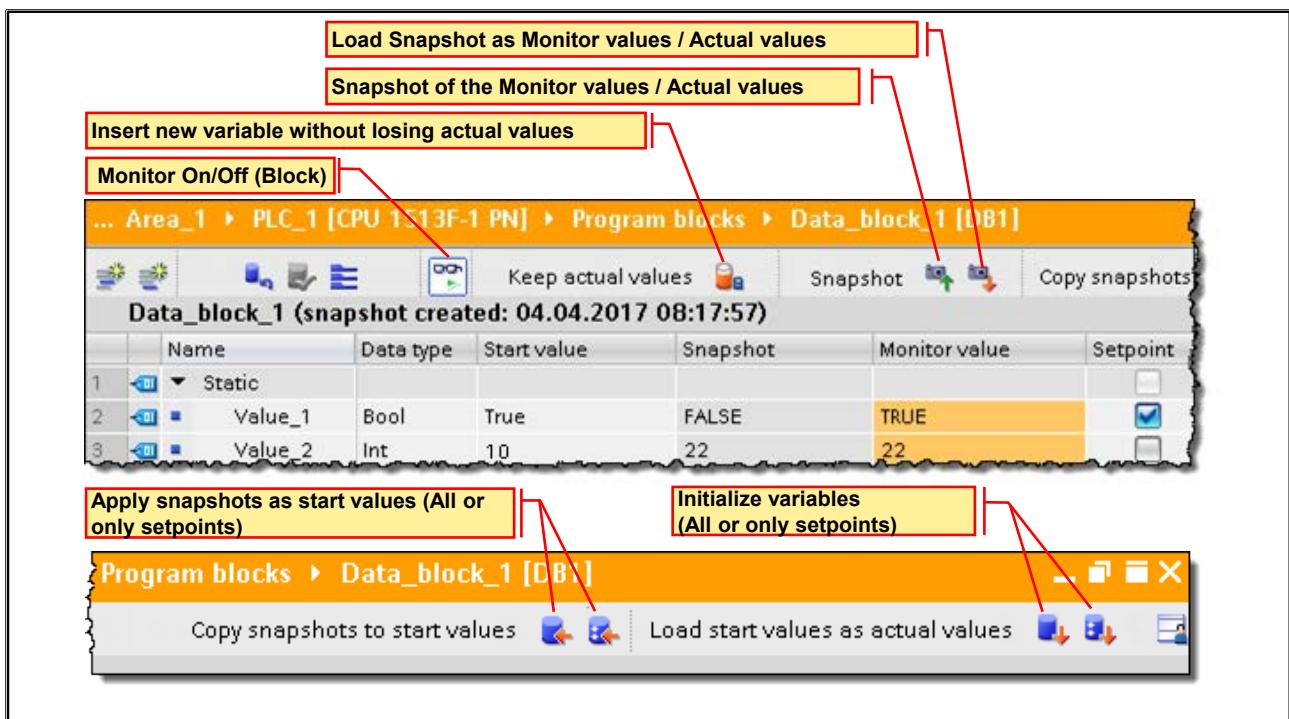
Structural changes are:

- Name changes (since synonymous with deleting a variable (tag) and creating a new one)
- Changes to the retentive behavior
- Adding / removing tags

DB Re-initialization during Restart

No matter whether a DB is created with standard or optimized block access, all monitor values of non-retentive tags are overwritten with their start values when the CPU is restarted.

11.3.4. Snapshot, Setpoint, Start Value



Snapshot

With the "Snapshot of monitor value" function, the actual values of the online DB are stored in the offline DB. With the "Load snapshot as monitor value / actual value" function, the values are once more loaded into the online DB.

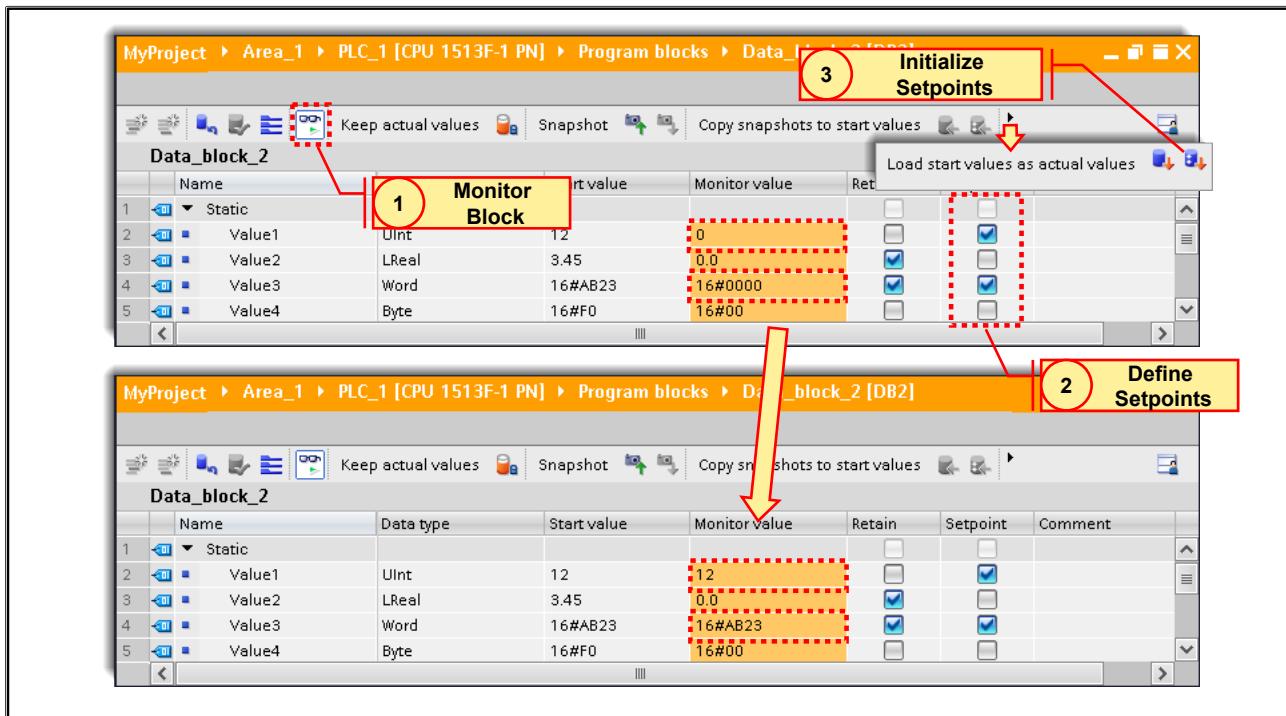
Copy Snapshot to Start Values

Saved values can be adopted in the Start value column by means of the button (for all snapshots selected in the Setpoint column) or the button (for all snapshots). The next time this DB is transferred, these values are applied as start value (no structural change of the DB -> a download without re-initialization is possible).

Adjusting Actual Values

Furthermore, the entire DB can be initialized with the button and all values selected in the Setpoint column can be initialized with the button.

11.3.4.1. Initializing Setpoints Online



Initializing Setpoints in the Online Program

All variables that are checkmarked as "Setpoint" can be initialized online in the CPU. Online, the monitor values are overwritten with the start values. The CPU remains in "RUN". The changed monitor values are applied once at the next cycle control point. This applies for retentive as well as for non-retentive variables. The program execution then continues with the new variable values. Prerequisite is an online connection to the CPU, the structure of the data block is identical online and offline and one or more variables are checkmarked as "Setpoint".

What to Do:

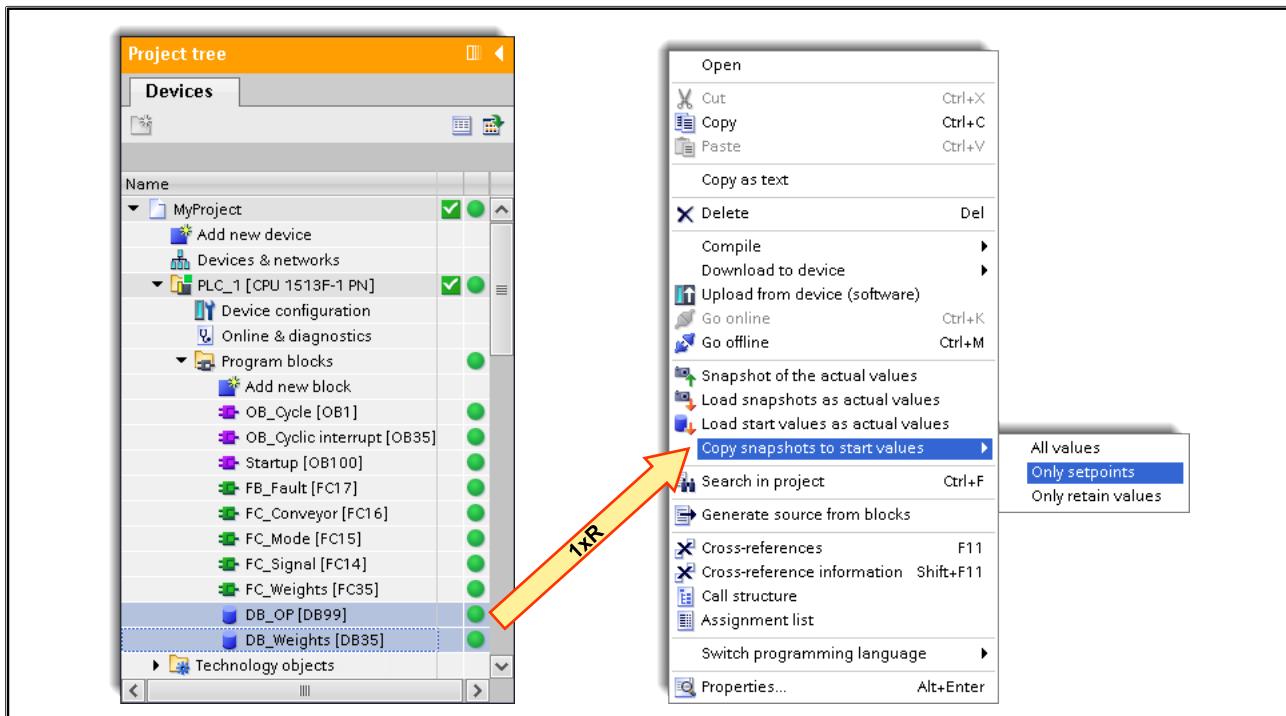
1. Open the global data block or instance data block and establish an online connection by monitoring it.
2. In the "Setpoint" column, checkmark the variables whose monitor values are to be overwritten with the start values in the CPU.
3. Click on the "Initialize Setpoints" button in order to initialize the variables checked as "Setpoints".

Note:

For global data blocks, the "Setpoints" checkmark can only be set and reset for those variables which have not been declared according to PLC-data type, without having to reload the block. For variables that are declared according to PLC-data type, the "Setpoint" checkmark must be made in the PLC-data type and then the DB has to be reloaded.

For instance data blocks, only the static variables can be initialized with "Initialize Setpoints".

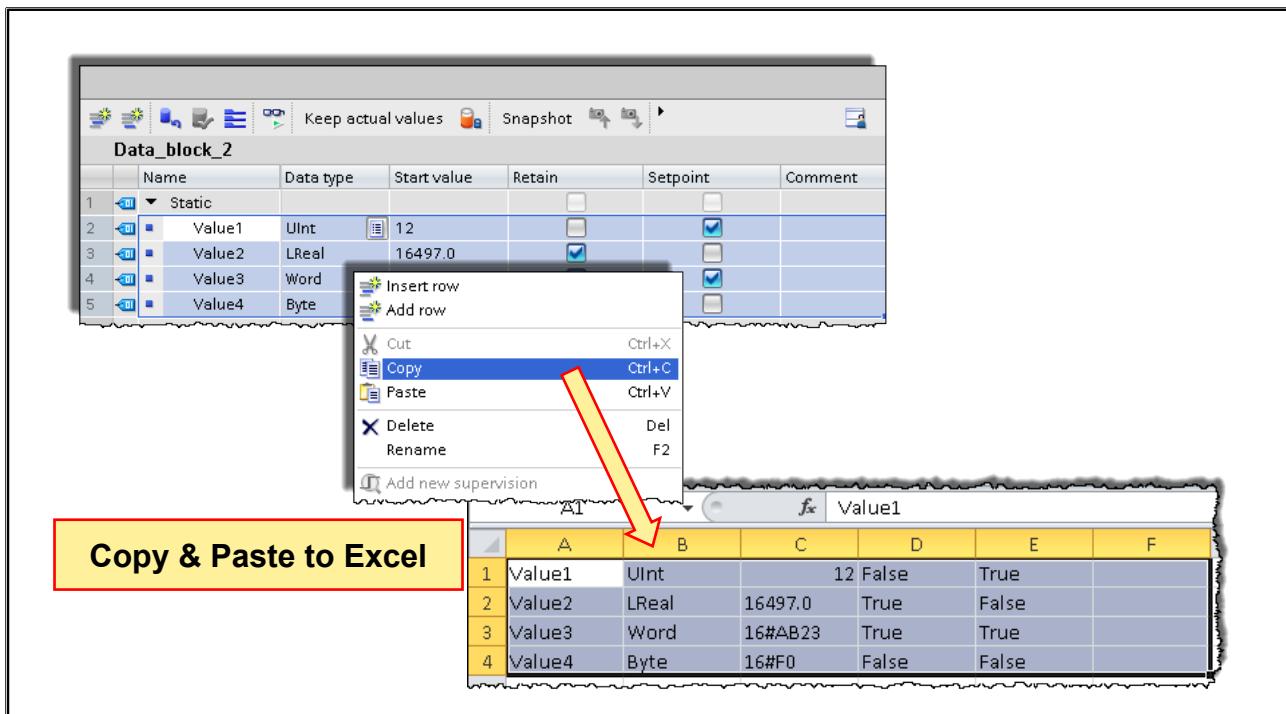
11.3.4.2. Changing the Snapshot / Start Value of Several / All Data Blocks



Snapshot of Several / All Data Blocks

Just as a snapshot can be made, written back, the DB initialized and snapshots copied into the start values for an individual data block, this can also be done for several or all blocks.

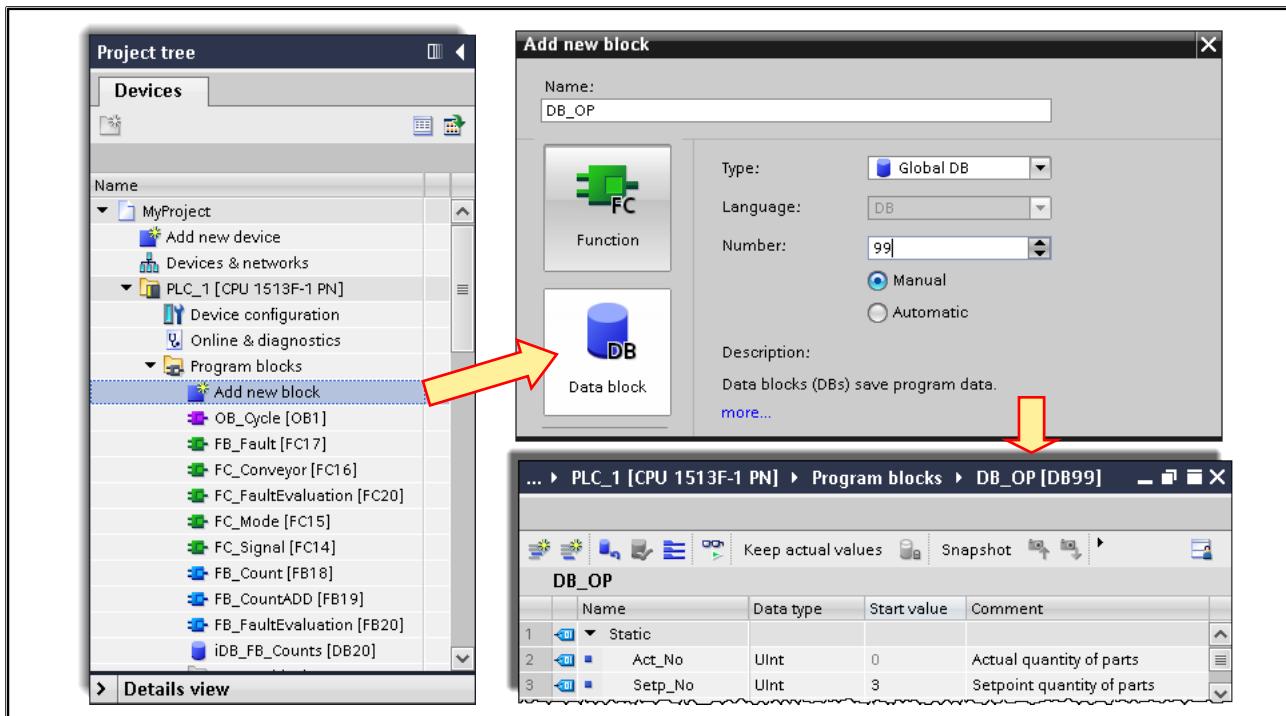
11.3.5. Copy & Paste from / to Microsoft Excel



Copy & Paste from and to Excel

The Windows Copy & Paste function can be used to easily copy individual or several variables from a data block to Excel to further process it/them there and then to copy it/them back from Excel to the data block.

11.4. Exercise 1: Creating Data Block "DB_OP"



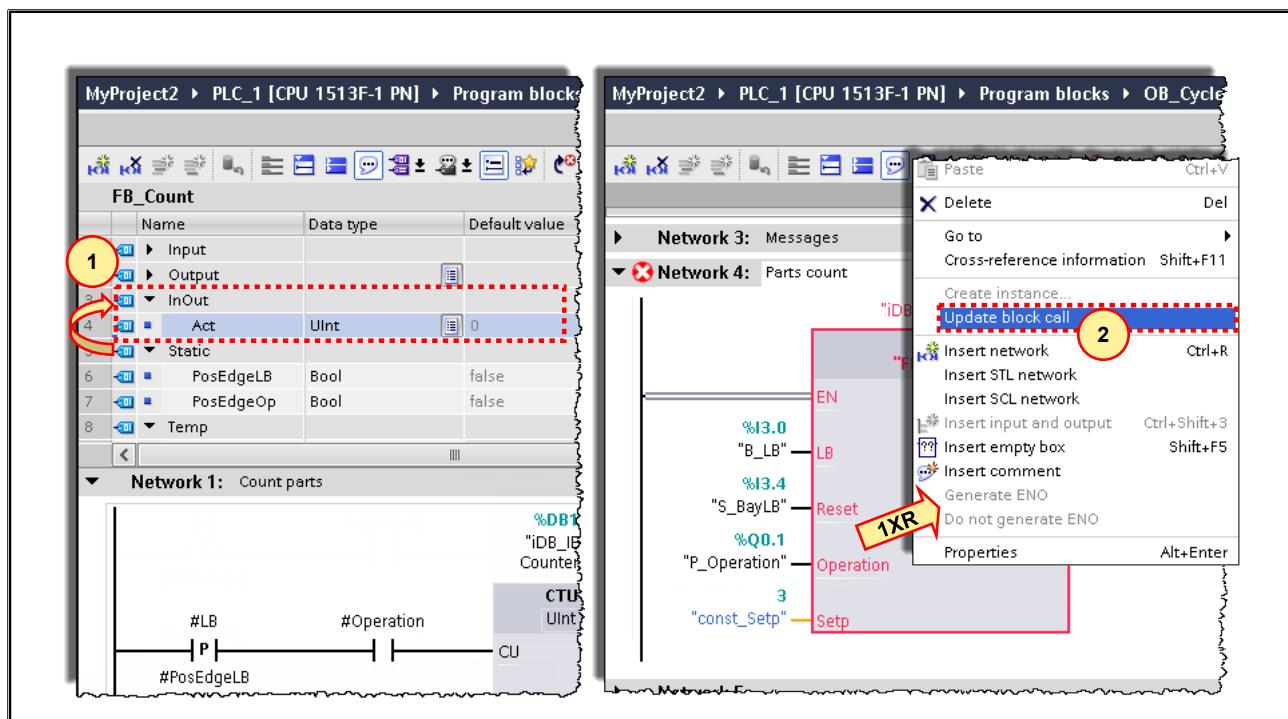
Task:

You are to create the data block "DB_OP" with the variables (tags) shown in the picture. The variables are to be used in the STEP 7 program and are also to serve as an interface to the touchpanel.

What to Do:

1. Create the new "DB_OP" as a Global DB.
2. Declare the variables as shown in the picture above. Give the variable "SetpNo" the Start value 3.
3. Save your project.

11.4.1. Exercise 2: Adjusting "FB_Count" and Updating the Call



Function Up Until Now:

The transported parts are counted in "FB_Count" as soon as they have passed the light barrier.

The current count, that is, the actual quantity is stored in the static variable #Act and the setpoint quantity is preset with constant 3 in the program.

If the actual quantity has reached the setpoint quantity, it is indicated on the conveyor model indicator light "P_BayLB" (Q3.4) with a 1Hz flashing light and no new transport sequence can be started. By acknowledging this signal with the conveyor model pushbutton "S_BayLB" (I 3.4), the static variable #Act is overwritten with 0 and further transport sequences can be started.

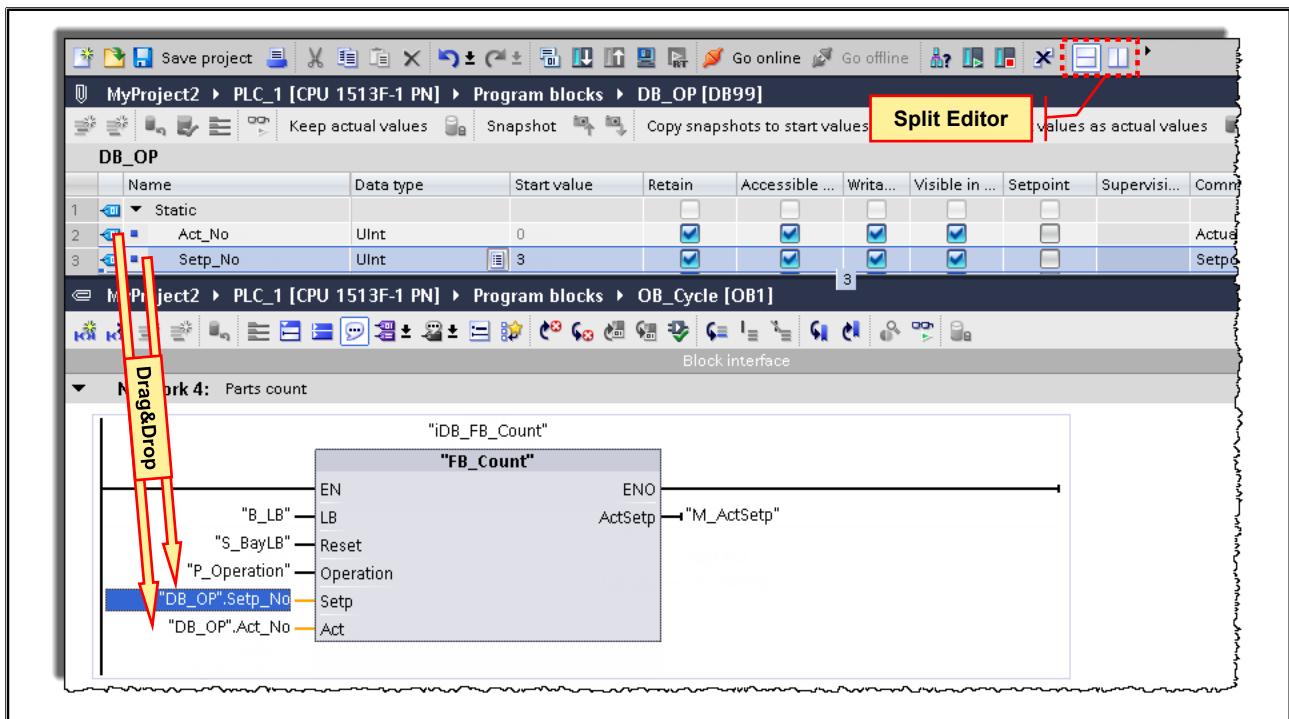
Task

The function of "FB_Count" remains unchanged, however, the actual quantity is no longer to be stored in the static variable #Act and the setpoint quantity is no longer to be preset with constant 3. Instead, the data block variables (tags) "DB_OP".ActNo and "DB_OP".SetpNo are to be used.

What to Do:

1. Change the static variable #Act into an InOut parameter of the same data type (UINT) and save the function block.
2. Open the Organization block "OB_Cycle".
3. Update the call of "FB_Count". For this, open the context menu by right-clicking on the call of "FB_Count" and activating the function "Update block call".
4. Confirm the dialog "Interface synchronization" with "OK".

11.4.2. Exercise 3: Using DB Variables as Actual Parameters



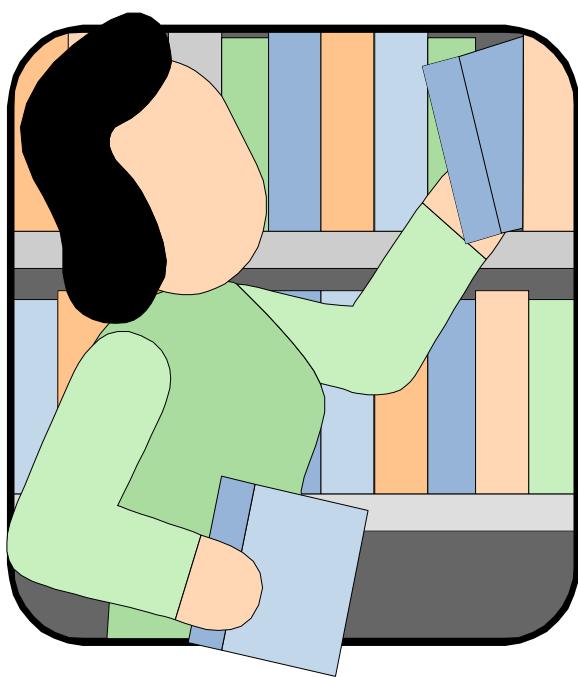
Task

You are to supply the formal parameters #Setp and #Act with the data block variables "ActNo" and "SetpNo" of the data block "DB_OP".

What to Do:

1. Split the working area using the "Split Editor" button.
2. In one area open the data block "DB_OP" and in the other the organization block "OB_Cycle".
3. Supply the formal parameters #Setp and #Act with the data block variables "ActNo" and "SetpNo" of the data block "DB_OP" using drag & drop.
4. Compile and save your project.
5. Monitor the "DB_OP" data block while you transport parts in Automatic mode and increase the ACT quantity.

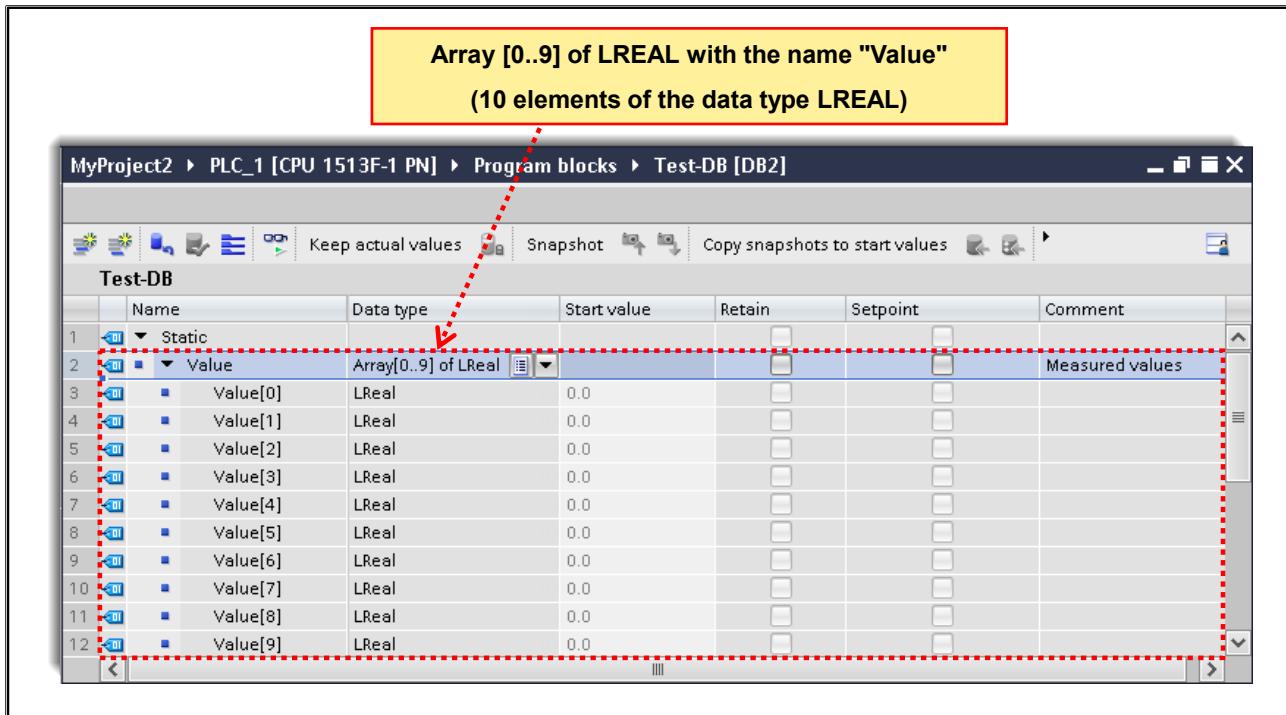
11.5. Additional Information



Note

The following pages contain either further information or are for reference to complete a topic.
For more in-depth study we offer advanced courses and self-learning mediums.

11.5.1. Example of a Variable of the Data Type ARRAY



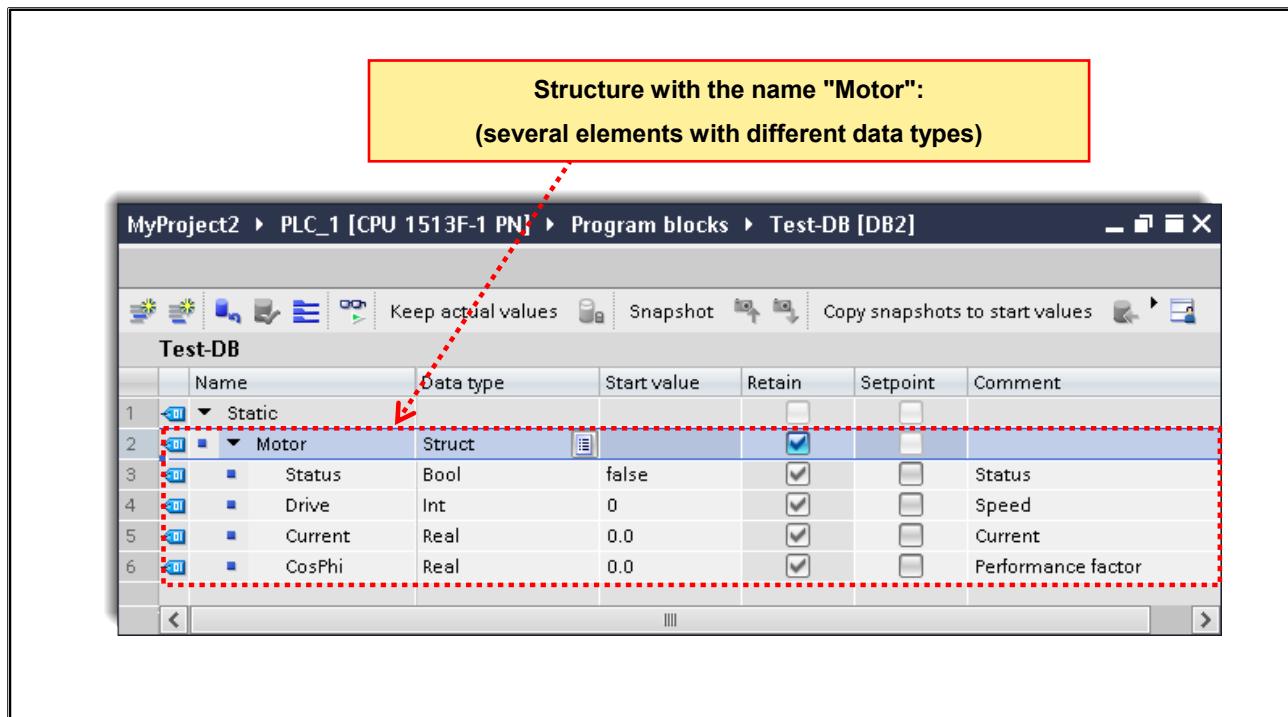
Array

An array consists of several elements of the same data type. In the picture above, you can see the array "Value" with 10 elements of the data type LREAL. Later, various measured values are to be stored in this array.

Declaring an Array in the DB

The data type of an array is called "ARRAY[n..m]". The first element (n) and the last element (m) are specified in the square brackets. In the example, [0..9] means 10 elements, whereby the first element is addressed with the index [0] and the last with the index [9]. Instead of [0..9] you could, for example, define [20..29]. This only influences the access to the elements.

11.5.2. Example of a Variable of the Data Type STRUCTURE



Structure

The picture shows an example of a structure named "Motor". The structure consists of several elements of different data types. The individual elements of a structure can be elementary or complex data types.

The access to the individual elements of a structure contains the structure name and the name of the element. This makes the program easier to read.

Example: accessing individual elements of a structure:

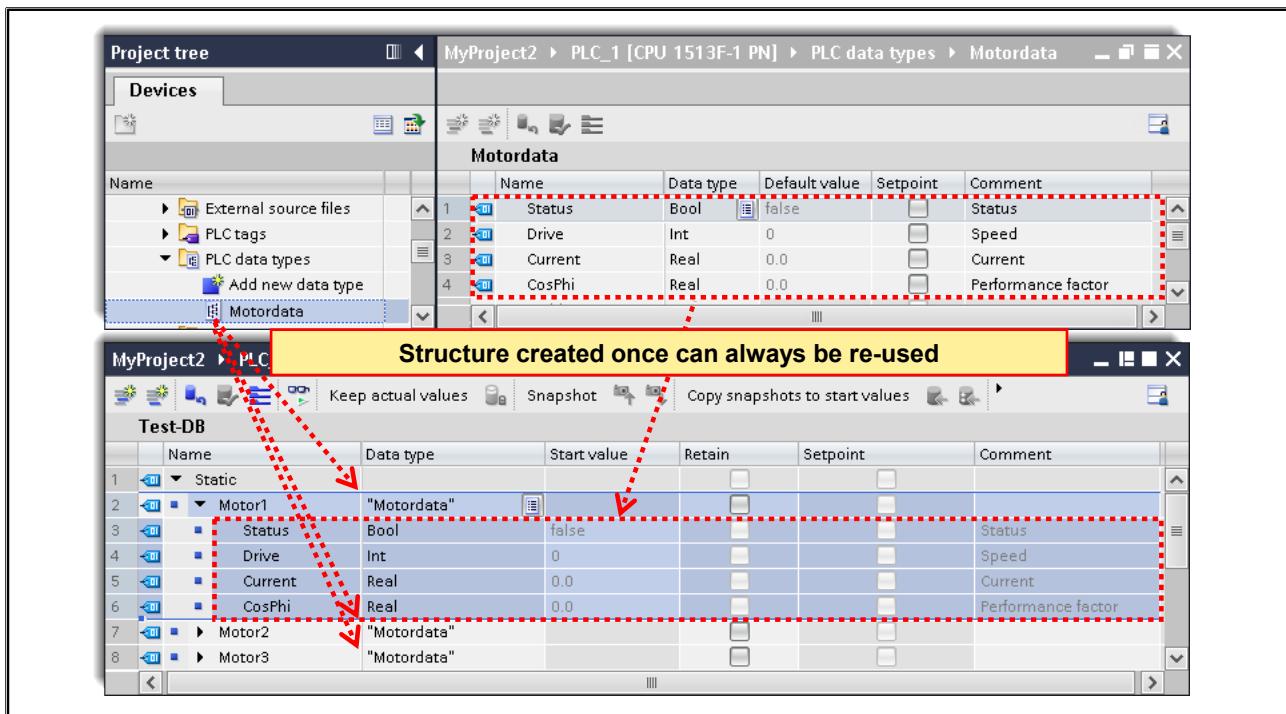
- "Test-DB".Motor.Status
- "Test-DB".Motor.Drive

"Test-DB" is the symbol name of the data block which contains the structure. After the symbol name, (separated by a dot) the structure name is specified. After the structure name (separated by a dot) an element name of the structure follows.

Declaring a Structure in the DB

As a keyword for a structure, "STRUCT" is used. The end of the structure is automatically identified with "END_STRUCT". A name is defined for the structure (in the example: "Motor").

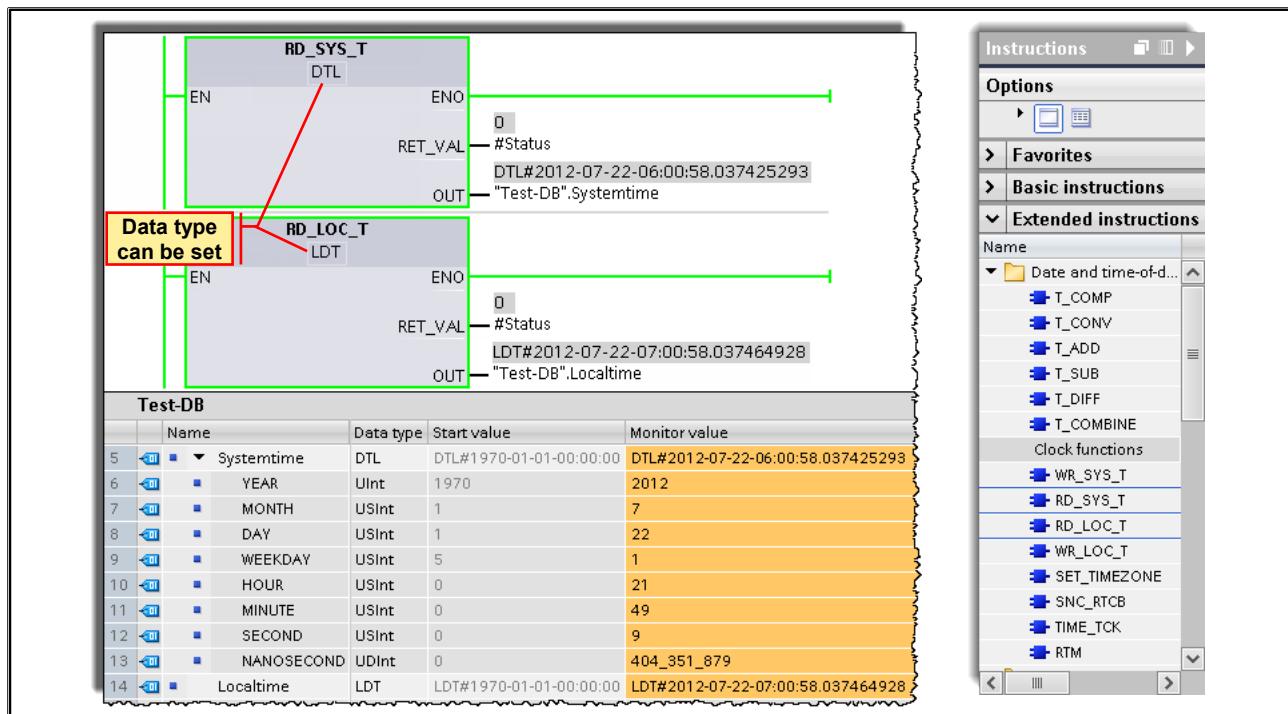
11.5.3. PLC Data Type



PLC Data Types

PLC data types are data types defined by you that are used as templates for declaring parameters and variables of complex data types (e.g. structure variables). PLC data types are created and stored in the PLC data types folder, and contain a data structure that is made up of elementary and/or complex data types. In the declaration of a variable according to PLC data type, a structure variable is created whose inner data structure is defined by the PLC data type. PLC data types can be used for the declaration of variables in global data blocks and within blocks for the declaration of local variables (TEMP, STAT) as well as parameters (IN, OUT and INOUT).

11.5.4. Functions RD_SYS_T and RD_LOC_T



In the Task Card Instructions > Extended instructions > Date and time-of-day you will find functions and instructions that are intended specifically for the handling of Date / time data types.

Here you will find functions with which, for example, times can be linked, compared, written, read out etc., with one another.

RD_SYS_T

You use this instruction to read the current date and the current time-of-day (module time) of the CPU clock.

RD_LOC_T

You use this instruction to read the current local time from the CPU clock and output this at the output OUT. To display the local time, the information about time zone as well as the start of daylight saving time and standard time which you set during the configuration of the CPU clock is used.

Contents

12

12. Connecting an HMI Device	12-2
12.1. Task Description: Operating the 'Plant' via the Touchpanel.....	12-3
12.2. Data Exchange between HMI Devices and CPU.....	12-4
12.3. WinCC Configuration Interface	12-5
12.4. Adding an HMI Device	12-6
12.4.1. Configuring the IP Address of an HMI Device	12-7
12.4.2. Networking an HMI Device	12-8
12.4.3. Configuring an HMI Connection.....	12-9
12.4.3.1. Checking the Connection and Entering the Password for CPU Accesses	12-10
12.4.4. Creating HMI Tags and Connecting them with PLC Tags.....	12-11
12.5. I/O Field for Inputting and Outputting Values.....	12-12
12.5.1. Buttons for Executing Functions	12-13
12.6. Compiling the Configuration.....	12-14
12.7. Downloading the Project into the HMI Device	12-15
12.8. Exercise 1: Copying the Touchpanel Project and the Interface DB into the Project	12-16
12.8.1. Exercise 2: Networking the Touchpanel	12-17
12.8.2. Exercise 3: Configuring the HMI Connection.....	12-18
12.8.3. Exercise 4: Updating and Completing the HMI Tag Table	12-19
12.8.4. Exercise 5: Configuring the SETPOINT Quantity Display	12-20
12.8.5. Exercise 6: Compiling and Saving the Touchpanel	12-21
12.8.6. Exercise 7: Adjusting the STEP 7 Program	12-22
12.8.7. Exercise 8: Downloading the HMI and CPU Project.....	12-23
12.9. Additional Information	12-24
12.9.1. HMI/OPC UA Access to PLC Tags and DB Variables.....	12-25
12.9.2. Manually Setting the IP Address on the Panel	12-26

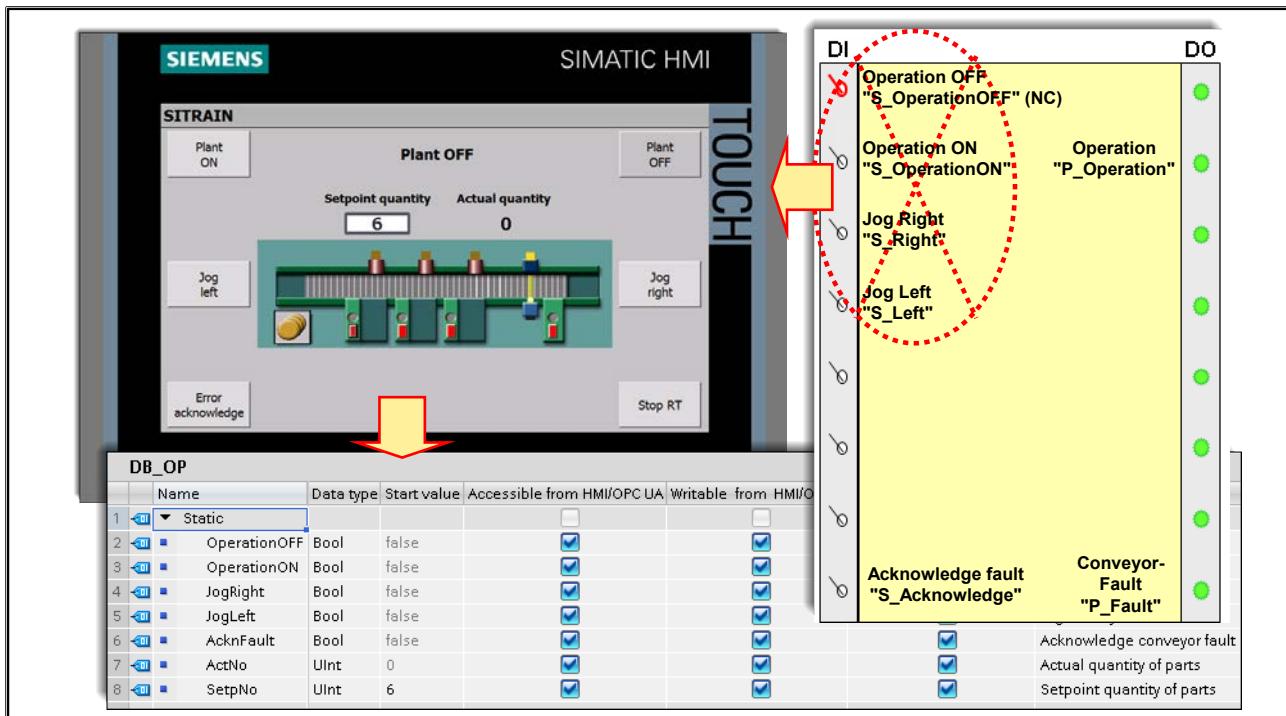
12. Connecting an HMI Device

At the end of the chapter the participant will ...

- ... be able to explain the principle of data exchange between HMI device and CPU using tags
- ... be able to set the interface of the touchpanel
- ... be able to commission a touchpanel project
- ... be able to adjust a STEP 7 program



12.1. Task Description: Operating the ‘Plant’ via the Touchpanel

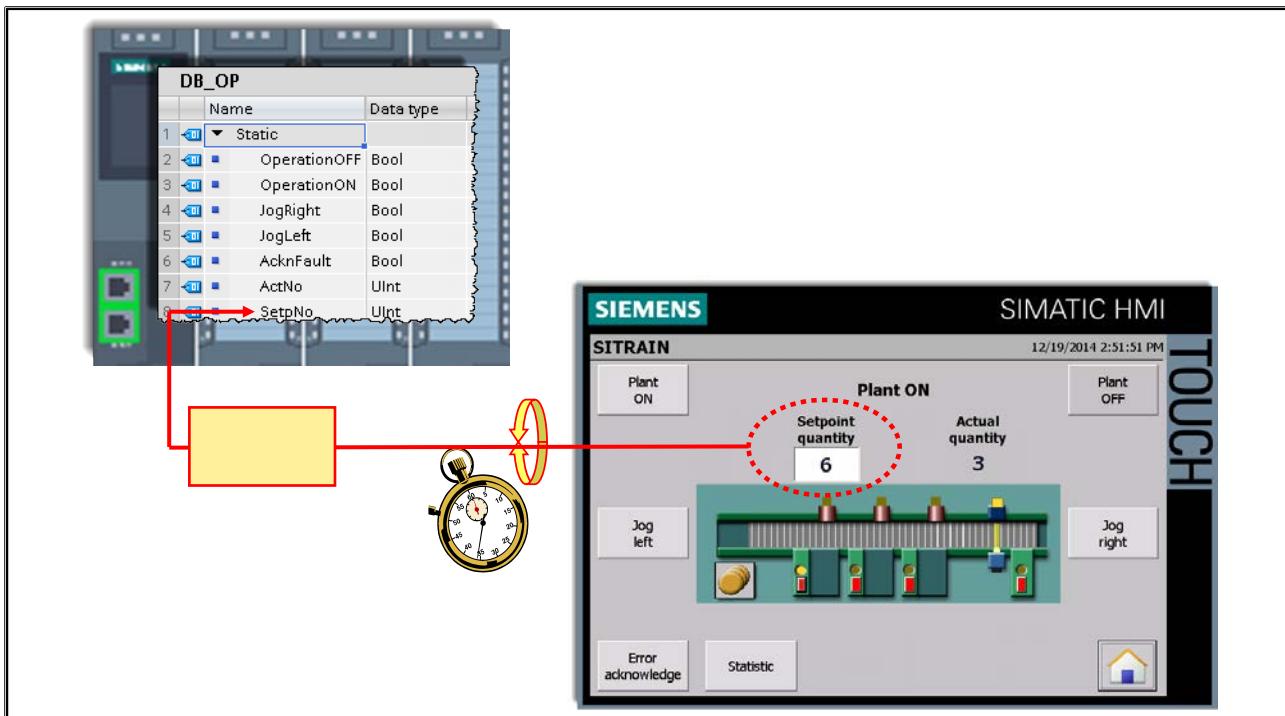


Task Description

The touchpanel project is to be commissioned and the S7 program of the controller is to be adjusted in such a way that...

- the functions "Operation ON/OFF" and "Jog Right/Left" are no longer realized via the simulator switches but via the corresponding buttons on the touchpanel.
- the acknowledgement of a conveyor fault should still be possible via the simulator switch "S_Acknowledge", and, in addition, also via the corresponding button on the touchpanel.
- The SETPOINT quantity is no longer a constant 3, but can be preset via an input field on the touchpanel.

12.2. Data Exchange between HMI Devices and CPU



Data Exchange between HMI Devices and CPU

Data is exchanged between SIMATIC S7 and the HMI system via tags. In the configuration of the HMI device, the screen objects, such as, buttons and input/output fields are linked to HMI tags which in turn are connected to PLC tags of the CPU. The HMI system cyclically exchanges the values between these tags. Data is transferred cyclically between SIMATIC S7 and the HMI system, that is, process variables are cyclically read and written by the HMI device depending on the configured update time.

HMI Tags

HMI tags can be connected to the global PLC tags or to the following global data areas of the CPU:

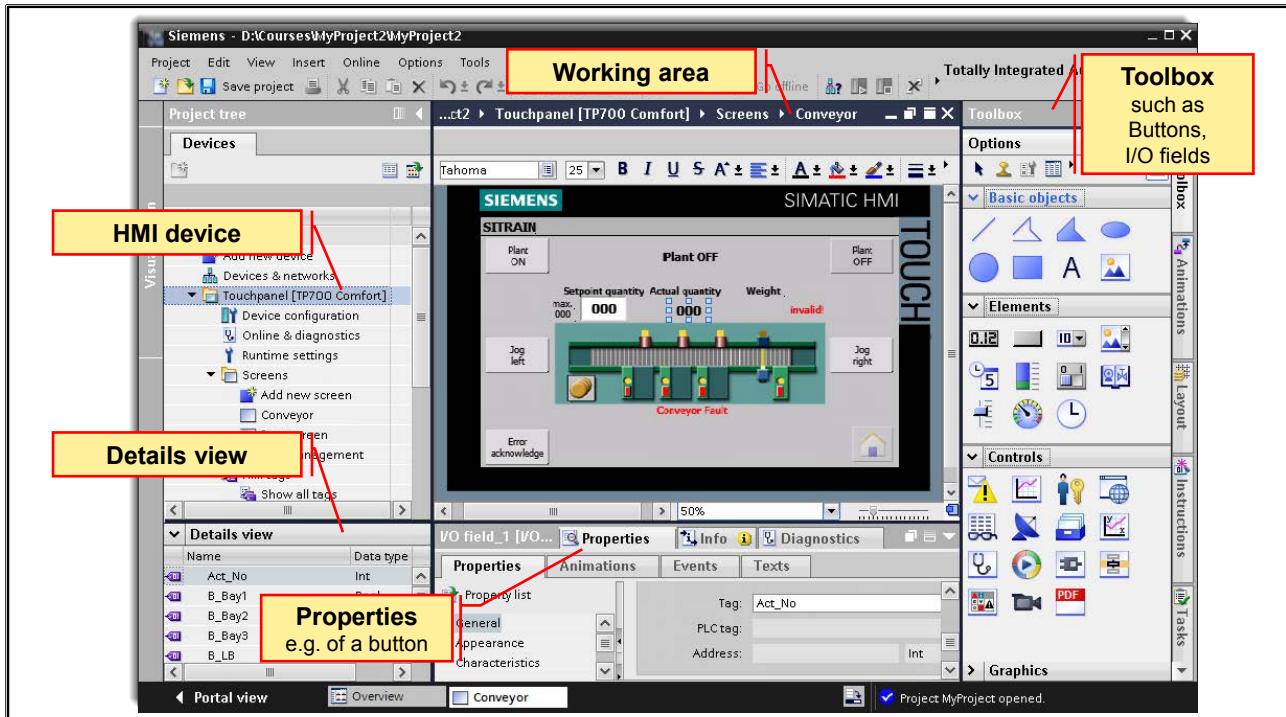
- Data blocks (DB)
- Memory bits (M)
- Inputs (I) and outputs (Q)
- I/O inputs (PI) and I/O outputs (PQ)

HMI systems also recognize local tags without a link to the PLC, i.e. these tags are exclusively processed internally and also do not reserve any communication resources whatsoever.

Communication

The operator panels can communicate with the (PLC) controller via the PROFIBUS or Industrial Ethernet bus systems. The S7 protocol is used for this purpose. Communication is organized by the operating systems of the S7 CPU and the HMI system. There is no user programming effort required. An operator panel can also exchange data with several (PLC) controllers.

12.3. WinCC Configuration Interface



Project Window

In the Project tree, all devices and their configuration and parameter assignments are displayed in a tree structure. From there, the relevant editors can be opened. Furthermore, the "language support" and the "version management" can be found here.

Working Area

This is the central configuration area in which objects of the operator panel are edited with the started editor. Several editors can be open at the same time.

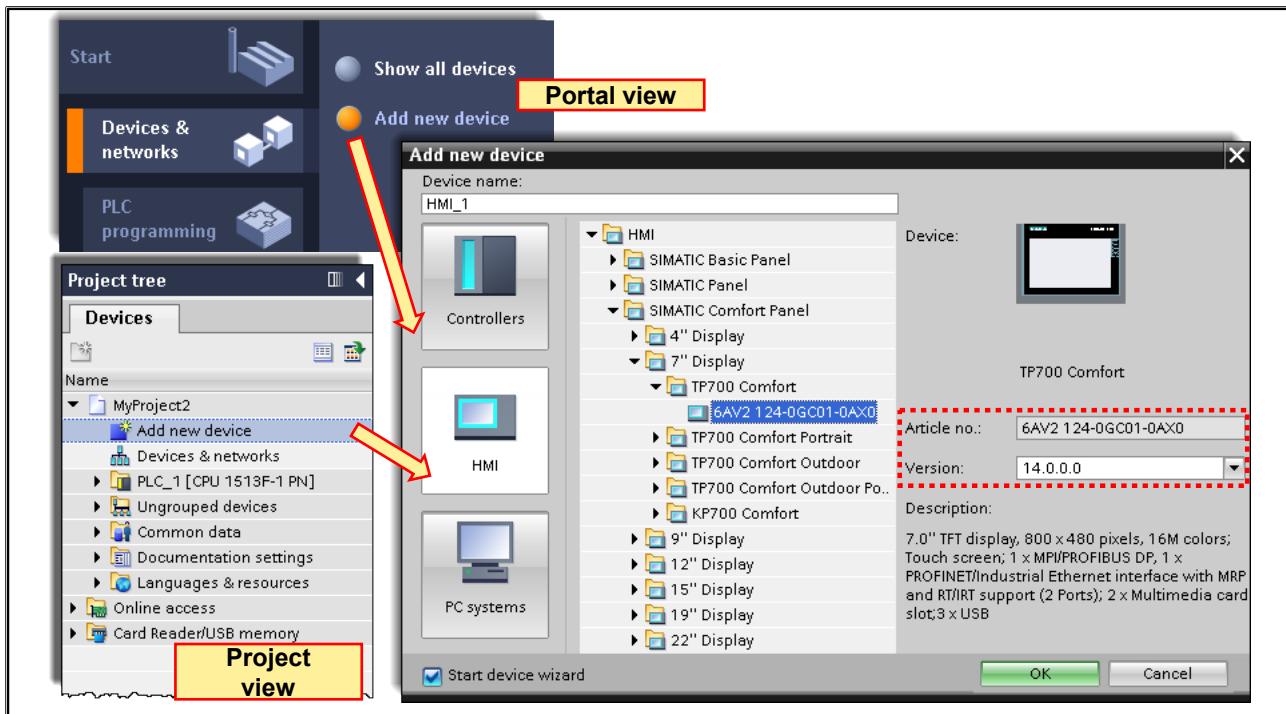
Properties Window

The properties of selected objects (for example, of screens, screen objects, tags) can be edited in the Properties window. This window is only available in those editors where object properties have to be set.

Toolbox Window

The toolbox window contains all configurable objects which can be configured in screens, and permits access to libraries.

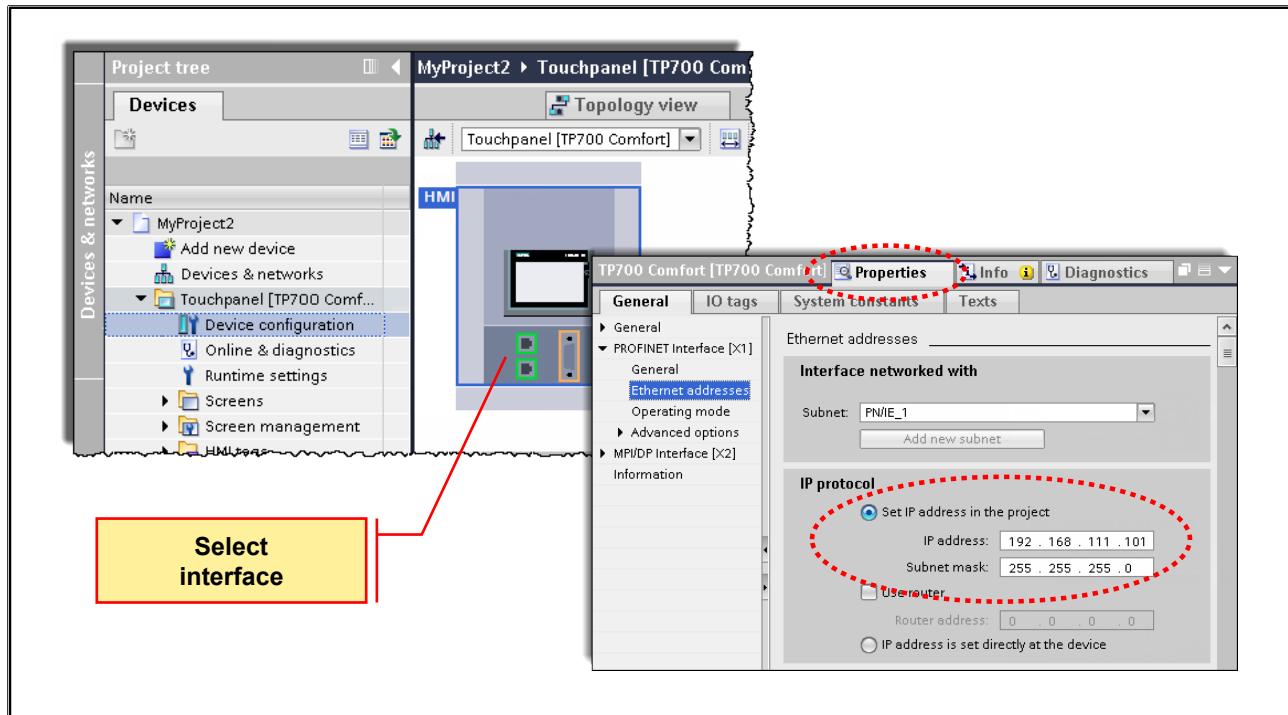
12.4. Adding an HMI Device



Adding an HMI Device

New HMI devices can be added from both the Portal view and the Project view. More than anything else, attention has to be paid to the device data such as Article number (order number) and version number.

12.4.1. Configuring the IP Address of an HMI Device

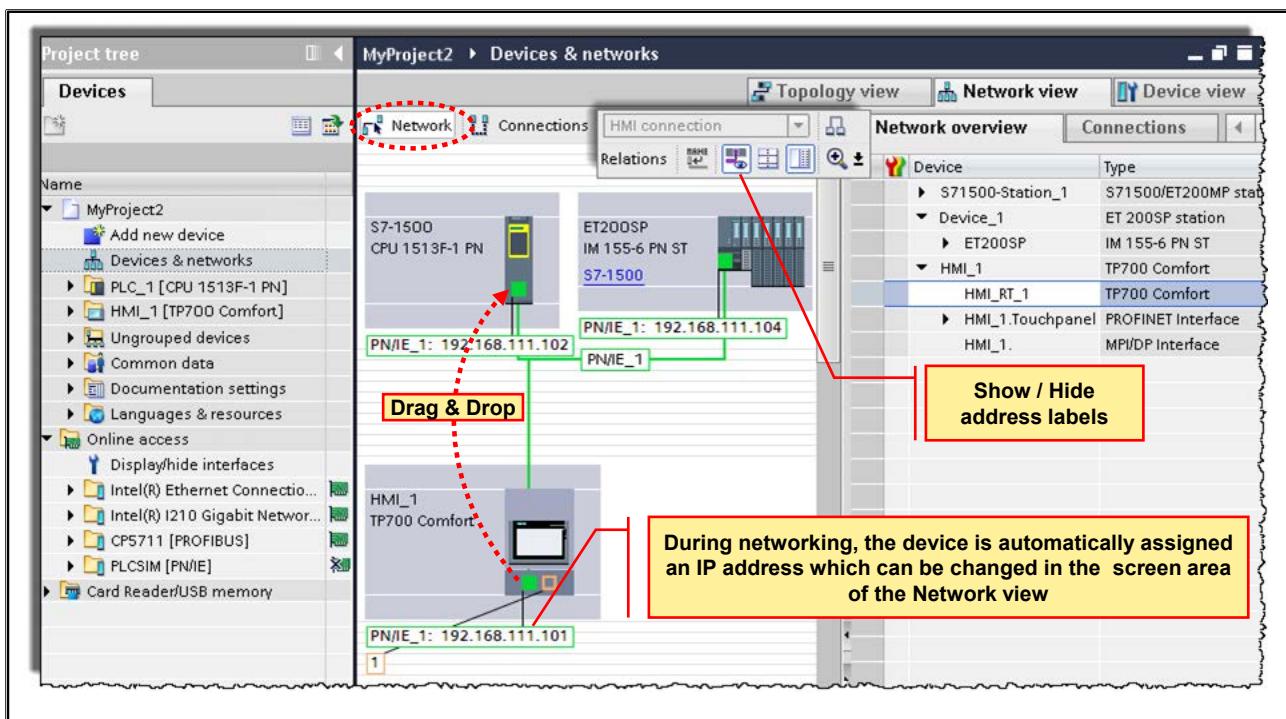


PROFINET Interface of the HMI Device

Regardless of whether the Hardware and Network editor is in the Devices view or the Network view, the settings of the PROFINET interface (IP address and subnet mask) can be made in the "Properties" tab in the Inspector window for a selected HMI device interface.

If an online connection between the HMI device and the CPU is to be established, both devices must be assigned the same subnet mask and IP addresses that are in the same subnet.

12.4.2. Networking an HMI Device

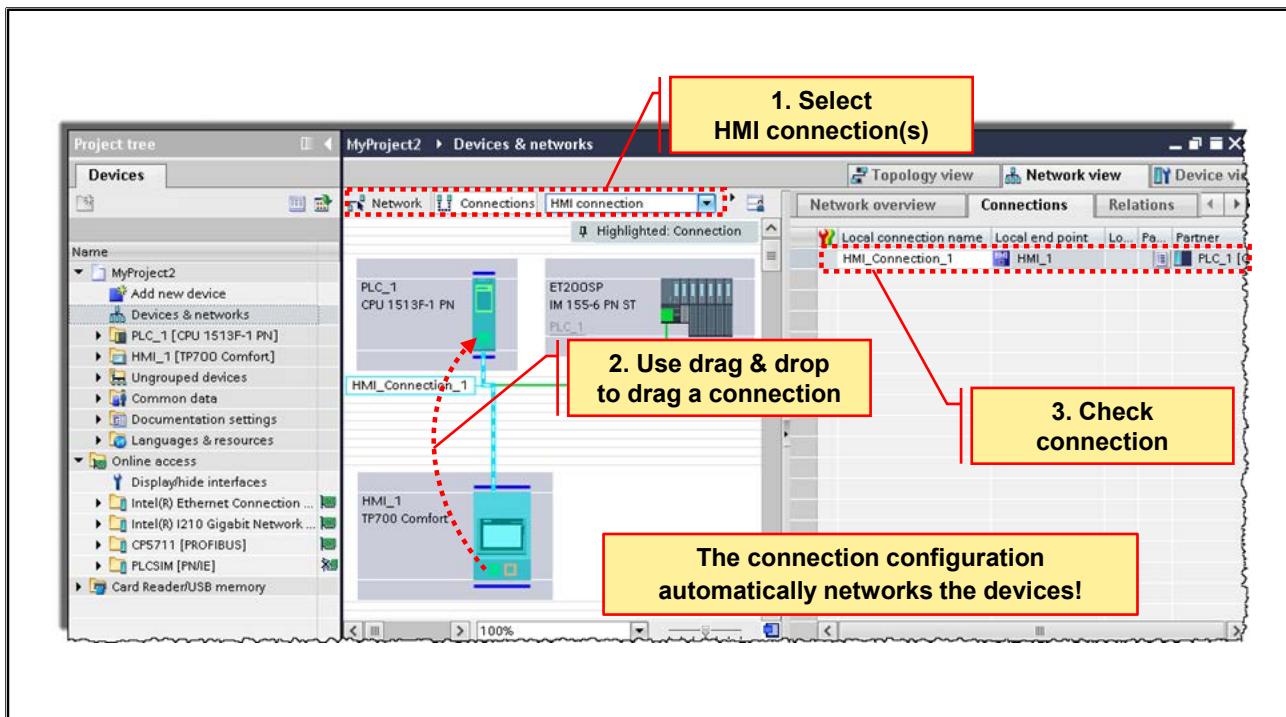


Networking an HMI Device

During networking, devices are connected to a subnet. The device interface must be compatible with the type of network.

The devices are networked in the Network view of the Hardware and Network editor under "Network" by connecting the device interfaces using drag & drop.

12.4.3. Configuring an HMI Connection

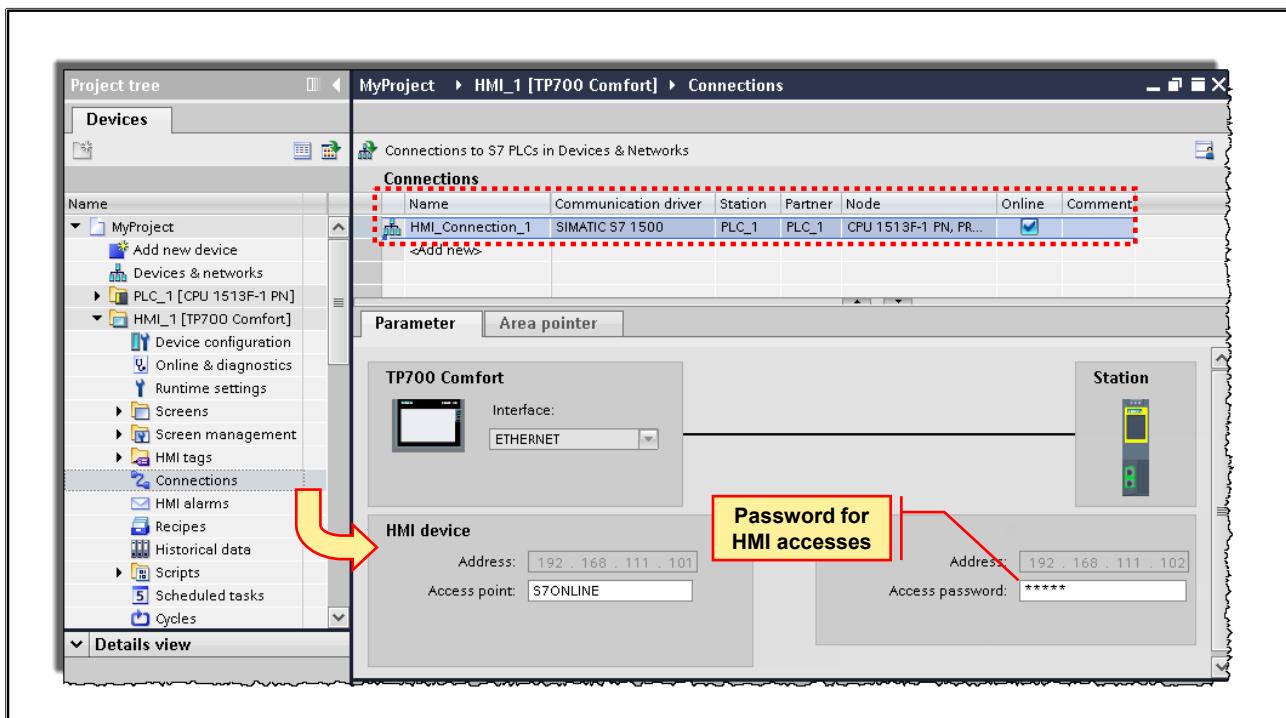


Configuring HMI Connections

In configuring HMI connection(s), the communications partners are defined with which the HMI device will later exchange data in the process control phase. The HMI device can also be connected to or exchange data with several controllers.

There can also be controllers in the same network with which the HMI device does not exchange data. Then, the HMI device is "networked" with these controllers but it is not "connected".

12.4.3.1. Checking the Connection and Entering the Password for CPU Accesses



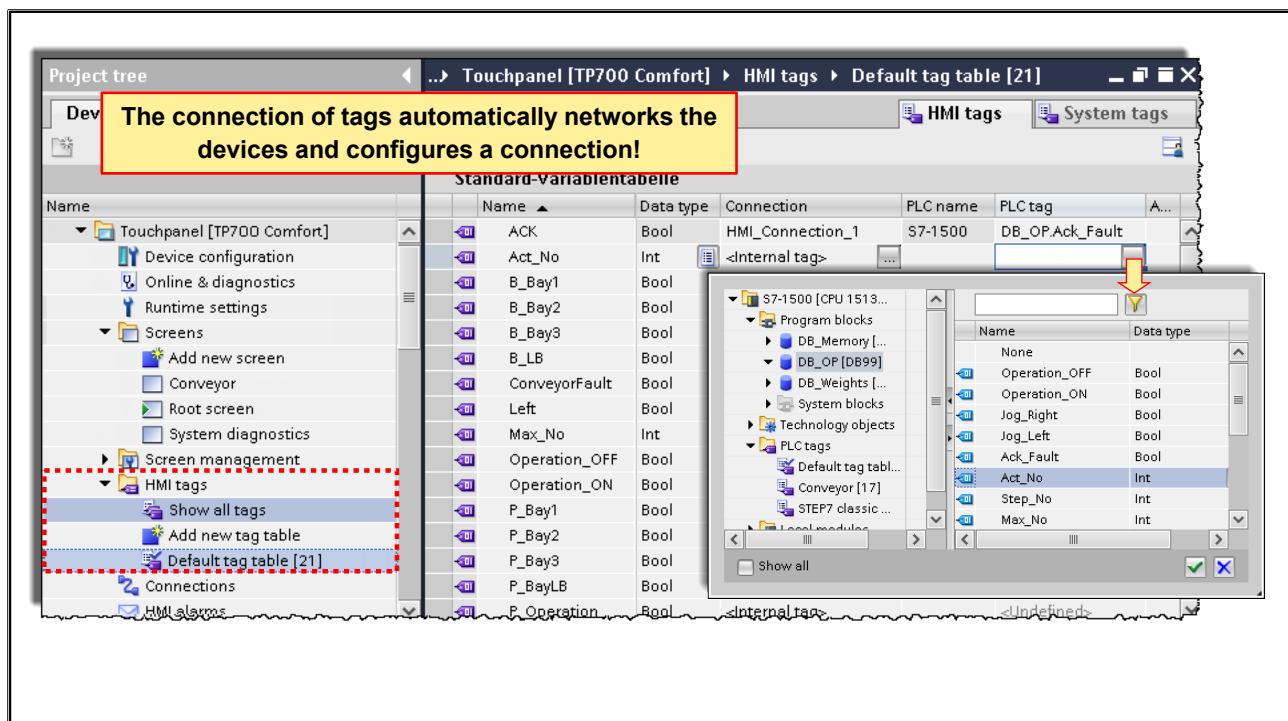
Connection:

The configured connections are visible in the Connections of the HMI device.

Password Query for HMI Accesses

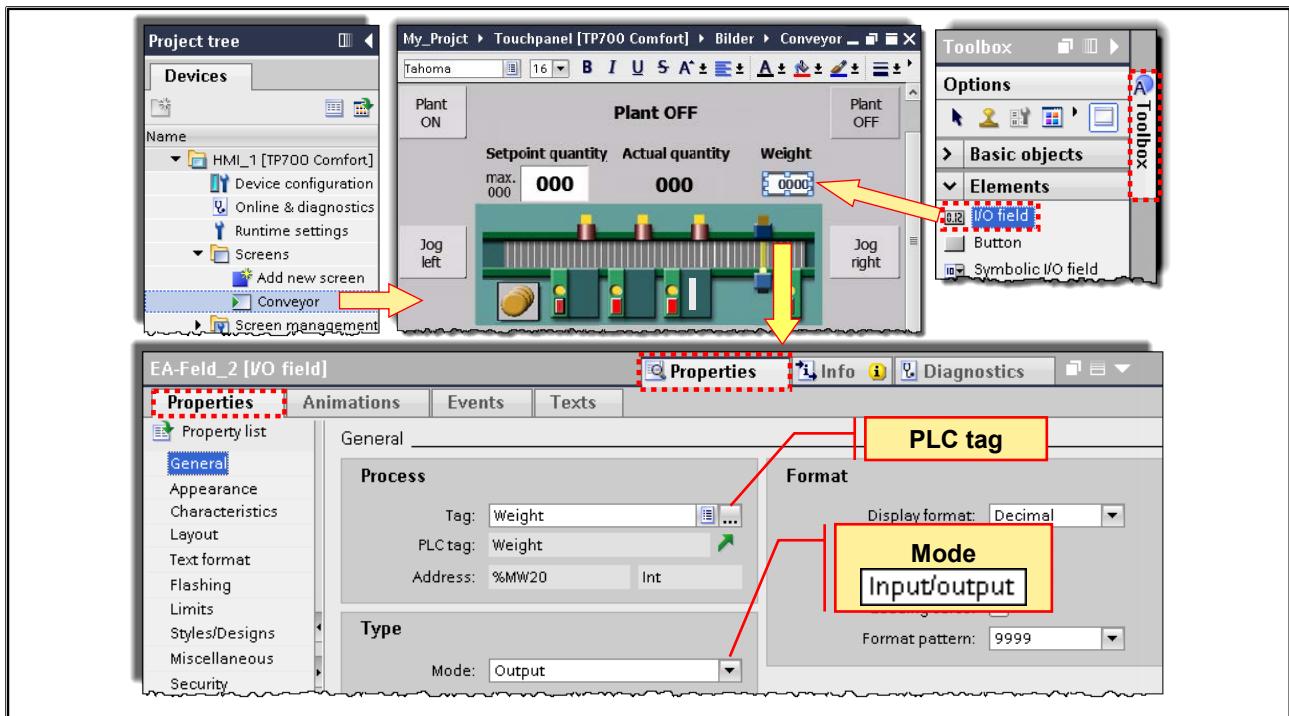
In order to get access rights to a CPU that is password-protected, the HMI device must log on to the CPU with a password when Runtime is started. This password must be specified in the Connections configuration of the HMI device (see picture bottom right).

12.4.4. Creating HMI Tags and Connecting them with PLC Tags



In the HMI tags folder, the HMI tags can be created and they can be connected with the associated PLC tags via an existing connection. The properties, such as, Access mode, Acquisition cycle etc., can be defined.

12.5. I/O Field for Inputting and Outputting Values



I/O Fields

The values of tags are displayed via output fields; the values of tags can also be preset via input fields. The mode can be set in the Properties window.

Mode

Output

The value of the tags is only displayed. The tag is read and updated in the interval of the configured acquisition cycle.

A value change (input) on the operator panel is not possible.

Input/Output

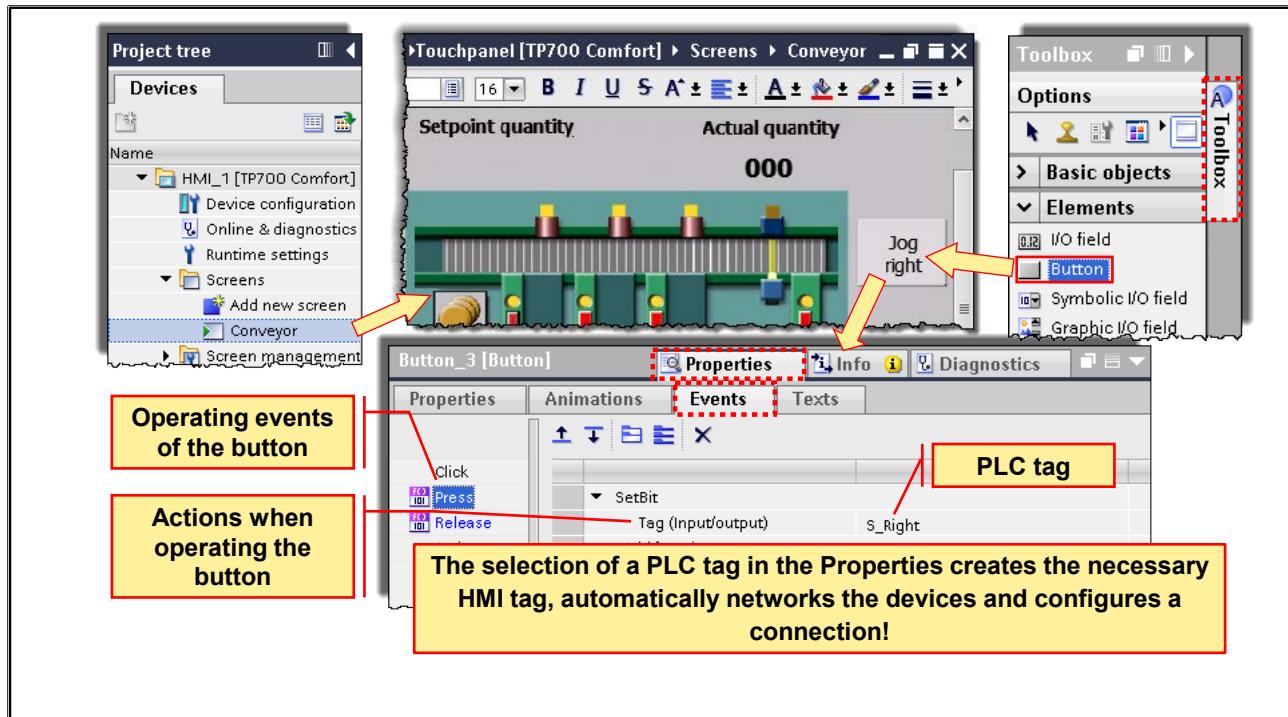
The value of the tags is displayed. The tag is read and updated in the interval of the configured acquisition cycle.

A value change (input) on the operator panel is possible – on touch displays, using the automatically displayed Display (screen) keyboard, and, with key devices, using the device keyboard.

Note:

I/O fields can also be generated by dragging & dropping a tag directly from a tag table or a data block (or with the help of the Details view). In so doing, an HMI tag is automatically created which is connected via an existing connection to the PLC tag.

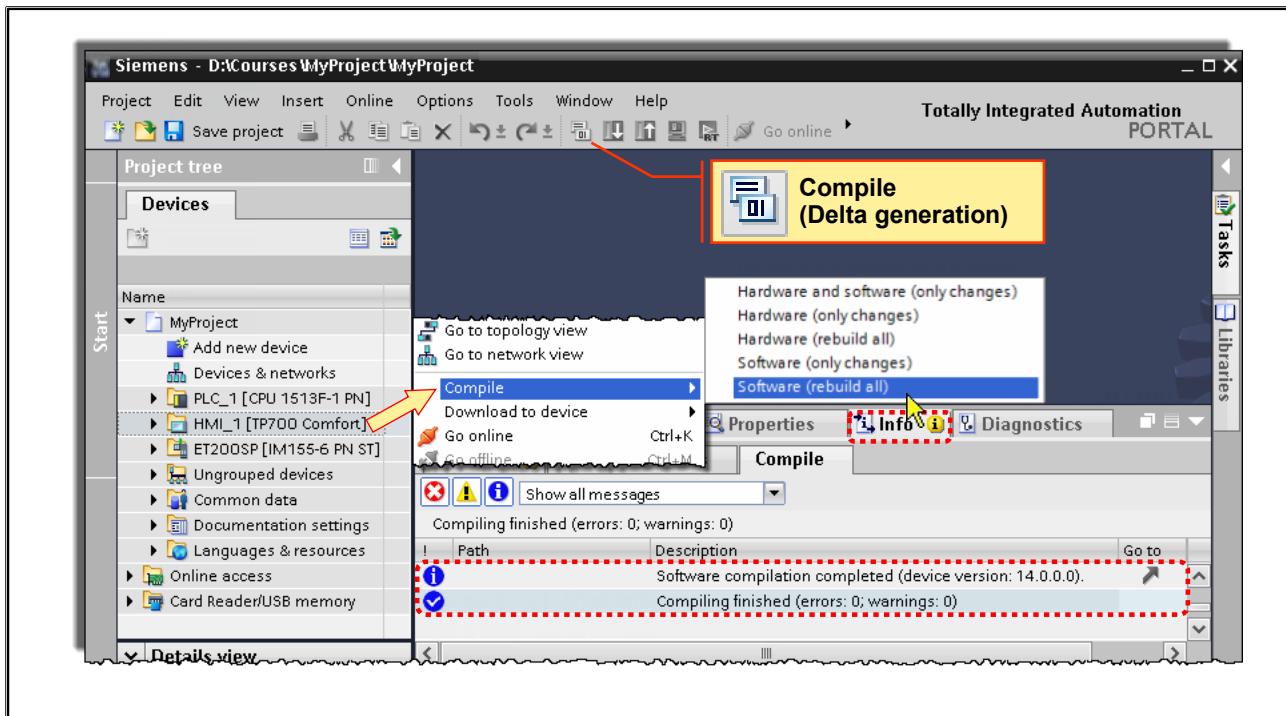
12.5.1. Buttons for Executing Functions



Buttons

System functions can be initiated by the operator via buttons, such as, the selection of a screen or the setting and resetting of a tag shown in the screen. The "Events" of a button are used to specify for which event which system function is to be executed.

12.6. Compiling the Configuration



Compile (only changes)

- Here, all changes since the last generation are regenerated.
→ Delta generation

Software/Hardware (rebuild all blocks)

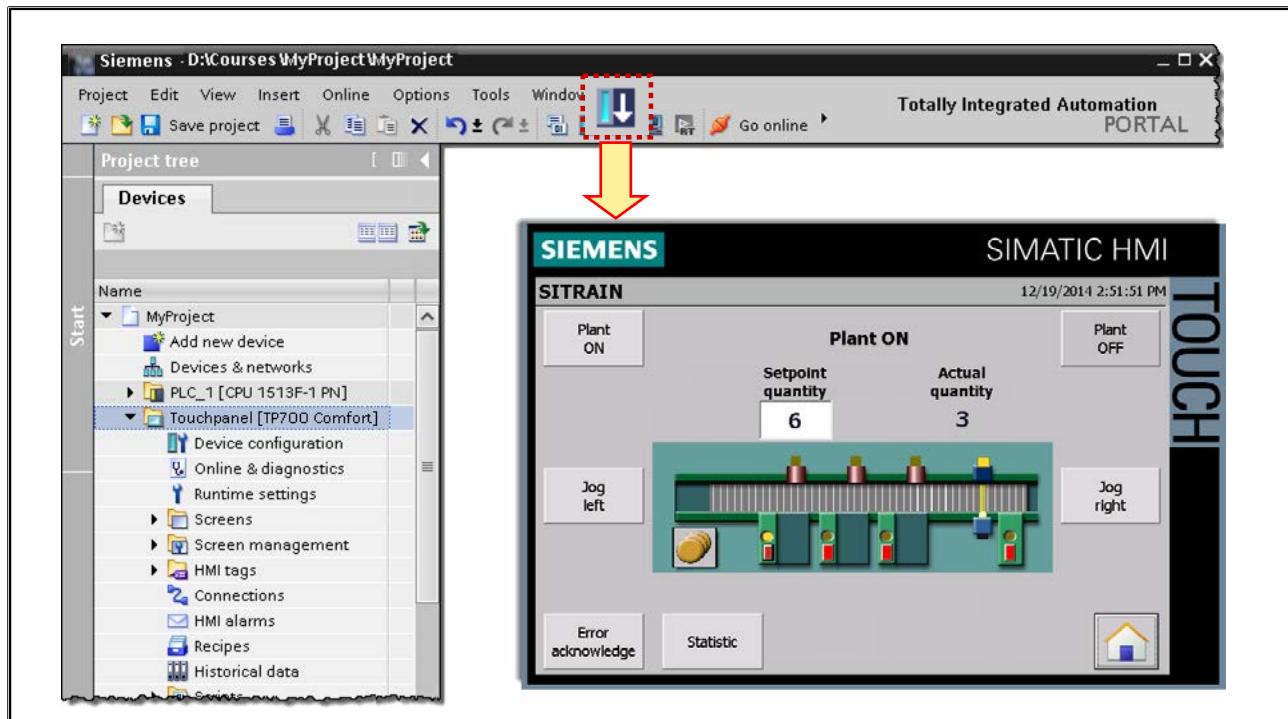
- Regenerates the entire operator panel.

When is a Rebuild all blocks necessary?

- When correctly configured functions are not correctly executed or are not executed at all.

When error messages occur during compiling that are not "correct".

12.7. Downloading the Project into the HMI Device



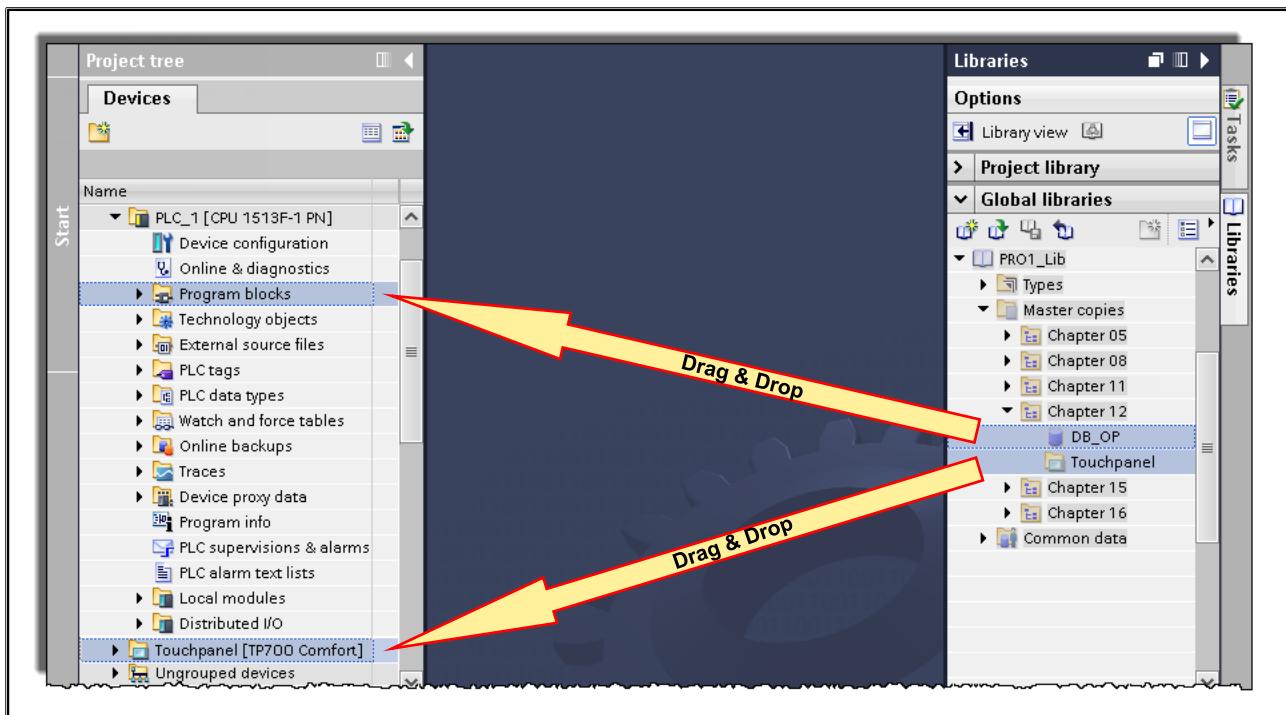
Downloading the Project into the HMI Device

When you transfer an HMI project to one or more operator panels, the part of the project that has been changed since the last transfer is automatically compiled before downloading. This ensures that the current project status is always transferred. Beyond that, it is also possible to activate the option "Overwrite all" before loading starts.

For commissioning, the project should be completely compiled using the command "Compile > Software (rebuild all blocks)" in the context menu of the operator panel. If HMI tags that are connected to PLC tags are also used in the project, all modified STEP 7 blocks should also be compiled using the command "Compile > Software" in the context menu and then be downloaded into the CPU.

In order to reduce the time required for compiling delta data in current engineering sessions, it is also recommended that you occasionally use the "Compile > Software (rebuild all blocks)" command.

12.8. Exercise 1: Copying the Touchpanel Project and the Interface DB into the Project

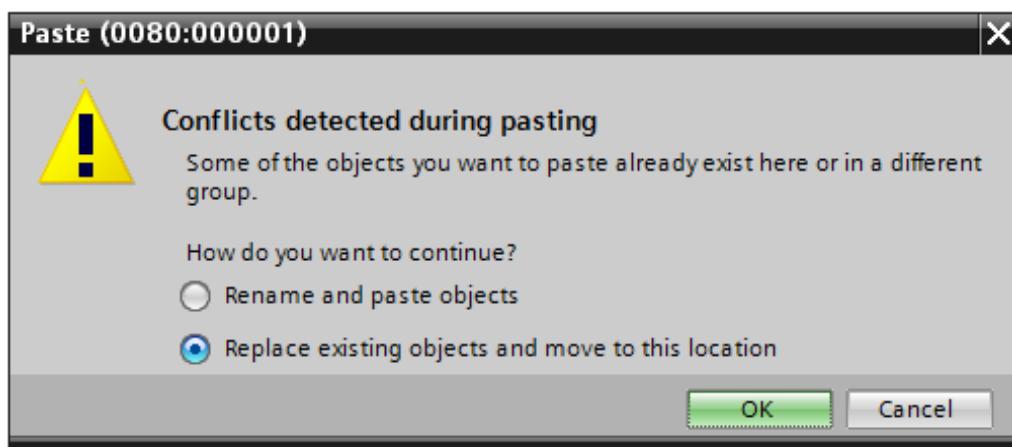


Task

Up until now, your project doesn't contain an HMI device. Instead of a completely new configuration, you are to copy an already configured Panel and the data block "DB_OP" that is to serve as the interface between the controller and the touchpanel, from the global library "PRO1_Lib" into your project.

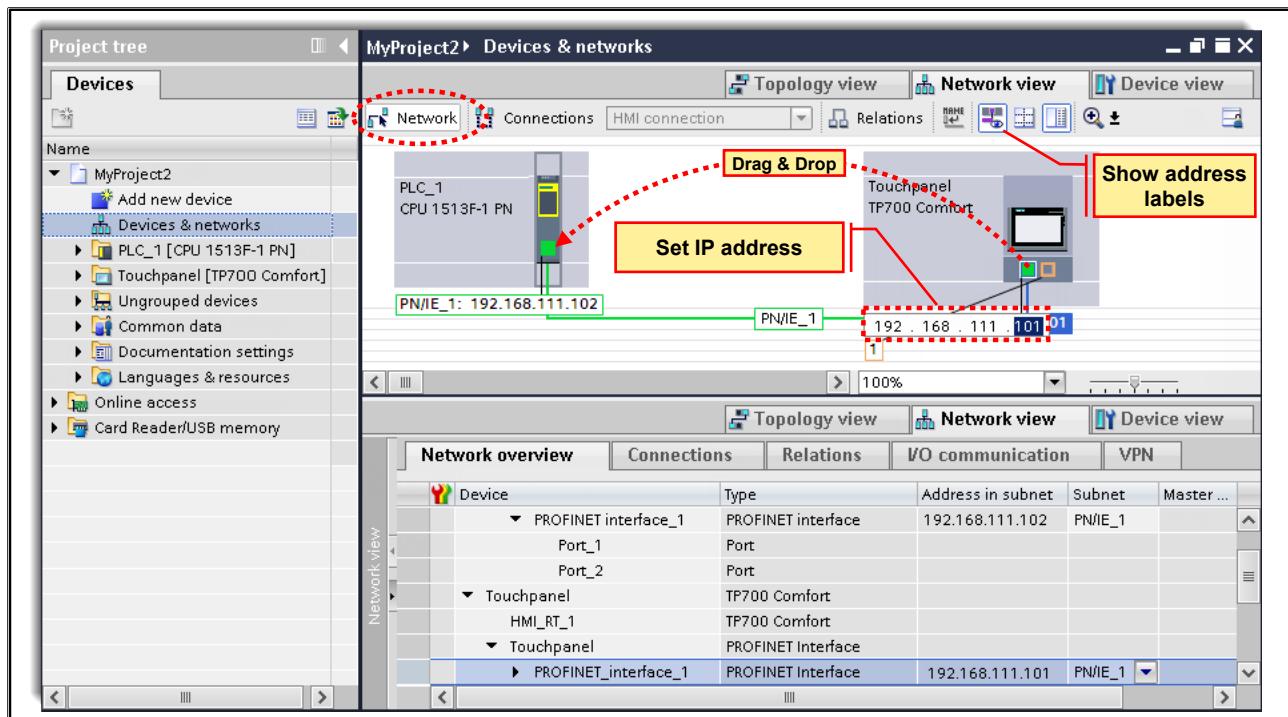
What to Do

1. Open the global library <Drive>:\02 Archives\TIA_Portal\TIA-PRO1\PRO1.Lib.
2. Using drag & drop, copy the library element "DB_OP" into your project (see picture) and, in the dialog that appears, confirm that objects that already exist in the project are to be overwritten.



3. Using drag & drop, copy the library element "Touchpanel" into your project (see picture).
4. Save your project.

12.8.1. Exercise 2: Networking the Touchpanel

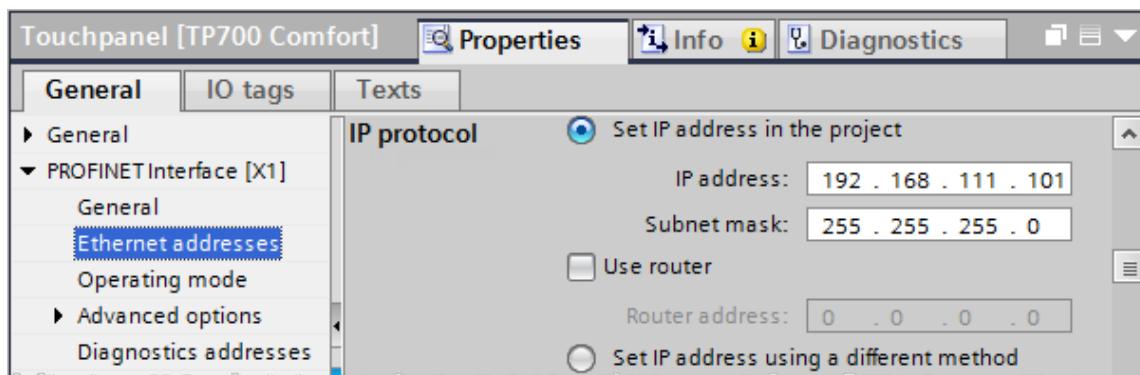


Task

The added touchpanel is to be networked offline with the Ethernet network.

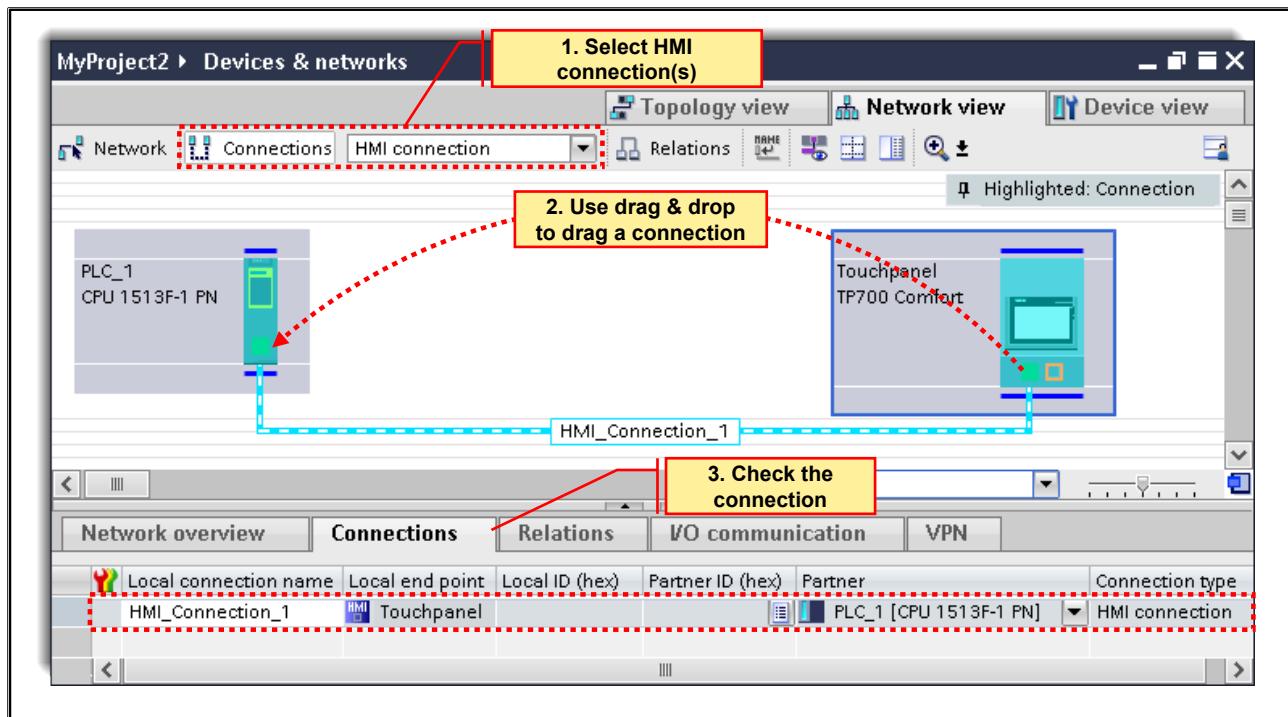
What to Do

1. In the Project tree, start the Hardware and Network editor, switch to the "Network view" and there select "Network".
2. Position the mouse pointer on the small green square of the HMI device and, while keeping the left mouse button pressed down, drag a connection to the CPU. The network is created; the associated subnet and the parameters appropriate for the network (IP address and subnet mask) are automatically created.
3. With the help of the "Show address labels" button, show the IP addresses, check the IP addresses of the CPU (192.168.111.102) and the touchpanel (192.168.111.101) and, if necessary, correct these.
This can be set directly at the shown address or also via the Properties of the devices in the Inspector window.



4. In the Network view, the interfaces of the CPU and the touchpanel should be connected with the same subnet.

12.8.2. Exercise 3: Configuring the HMI Connection



Task

After the TP has been networked with the Ethernet network, an HMI connection between the TP and the CPU must be created. You can use the automatically assigned name of the connection, "HMI_Connection_1", without changing it.

What to Do

1. In the Network view, switch from "Network" to "Connections" and there select "HMI connection" (see picture).
2. Position the mouse pointer on the small green square of the HMI device and, while keeping the left mouse button pressed down, drag a connection to the CPU. The connection is created and given the default name "HMI_Connection_1".

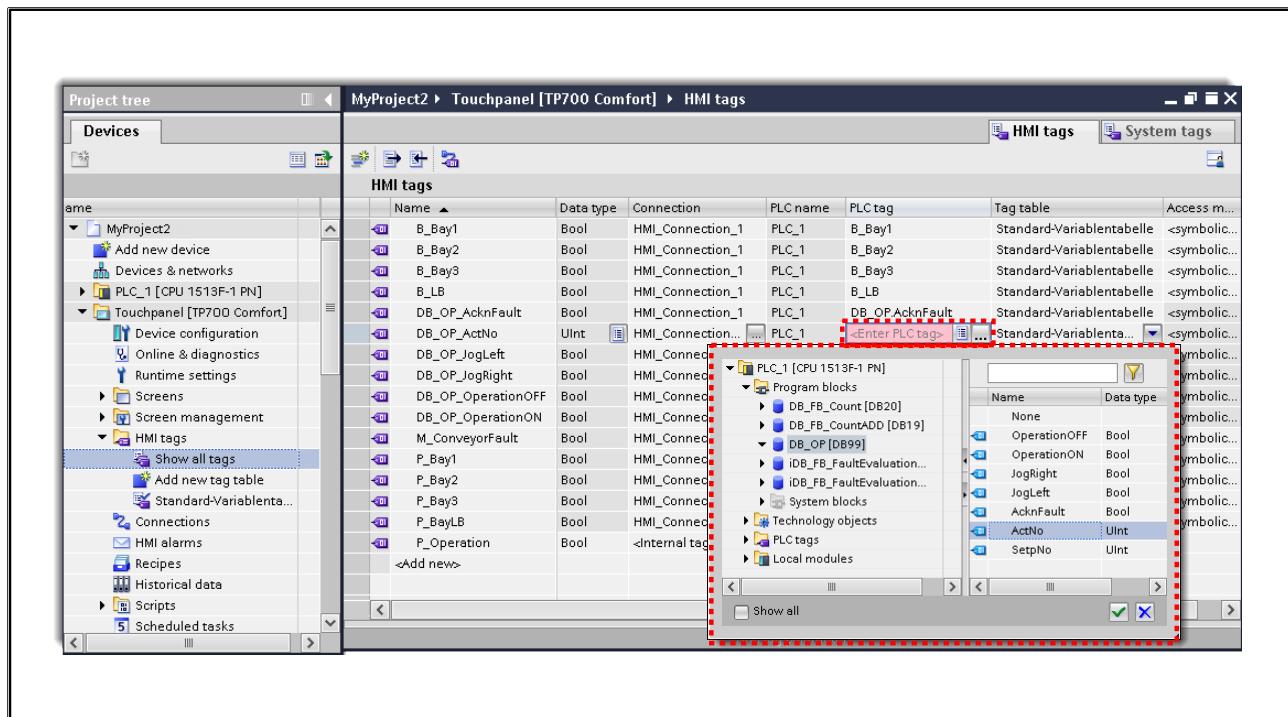
In the Details window in the "Connections" tab, check whether the HMI connection was correctly created (see picture).

You should also be able to see the connection in the Connections folder of the touchpanel.

- If, in the graphic area, the type of connection between the S7-CPU and the HMI device is not displayed but only the network(ing) is displayed, then position the mouse pointer on the network and, in the dialog window that appears, select the connection or double-click on the connection in the tabular area.

3. Save your project.

12.8.3. Exercise 4: Updating and Completing the HMI Tag Table



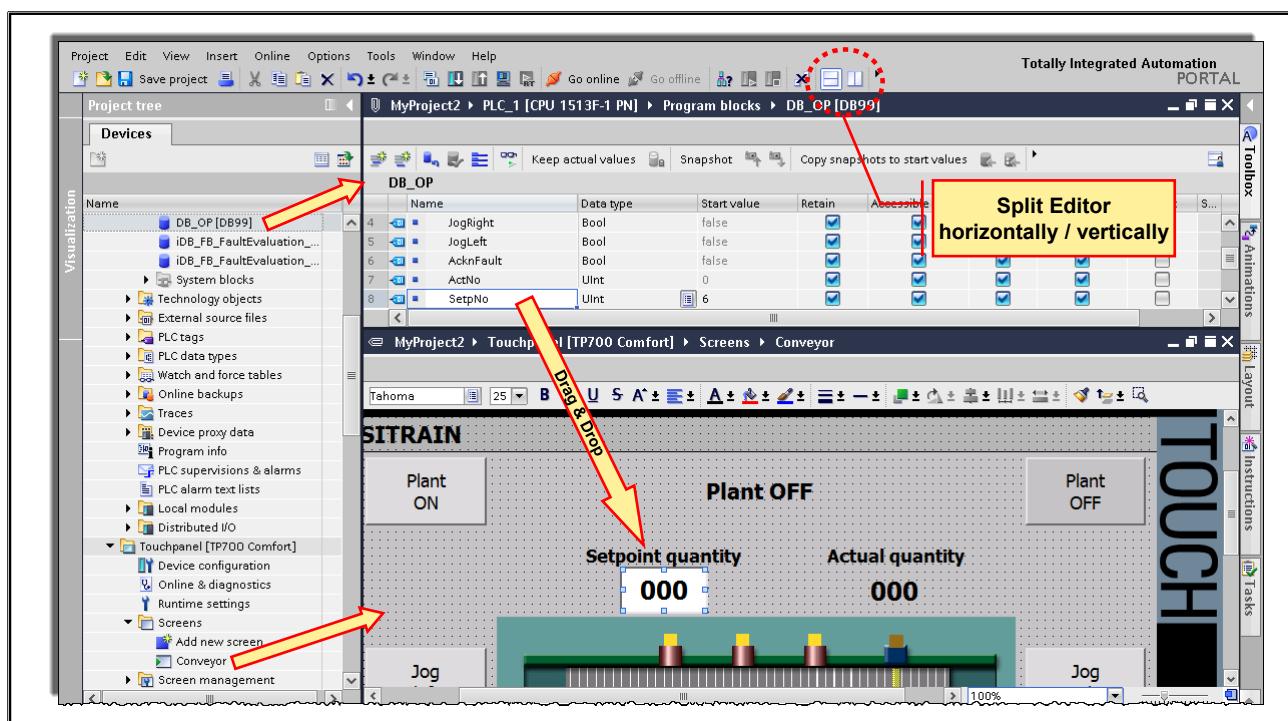
Task

The HMI tags "P_Operation", "M_ConvFault" and "DB_OP_ActNo" are not yet connected to PLC tags; for other tags, the "connection" is possibly highlighted in red. Your task is now to supplement missing tag connections and to correct faulty connections.

What to Do

1. Open "Show all tags" in the HMI tags folder of the Touchpanel.
2. Correct the first faulty (highlighted in red) connection by selecting the existing connection "HMI_Connection_1".
3. Adopt the connection of the corrected tag for all other tags. The fastest way to do this is to select the tag connection, click on the lower right corner of the cell and, while keeping the left mouse button pressed down, drag it over all other cells (familiar Excel function). Confirm the "Autocompletion" dialog that follows with the selection "Overwrite Tag attributes".
4. Connect the HMI tags that are not yet connected to the corresponding PLC tags. Proceed as shown in the picture.
5. Save your project.

12.8.4. Exercise 5: Configuring the SETPOINT Quantity Display

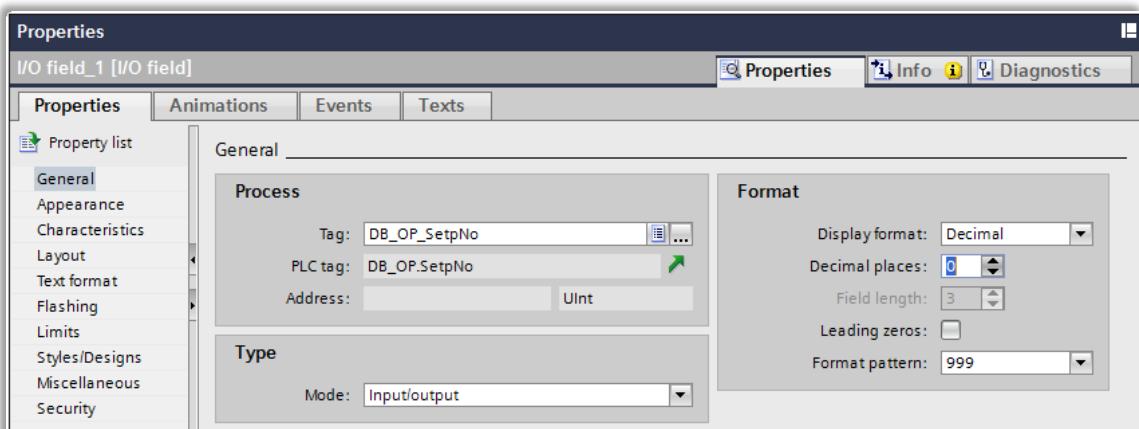


Task

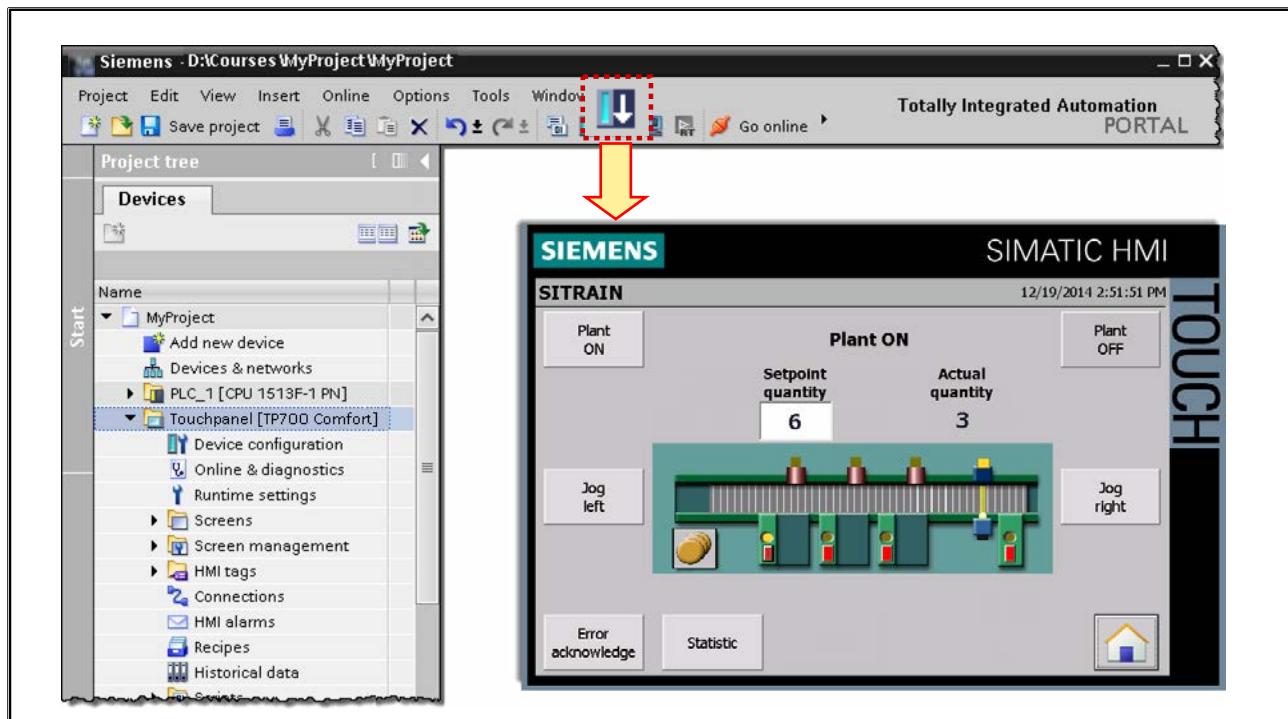
Up until now, the SETPOINT quantity of the parts to be transported was preset with the value 3 via the tag "DB_OP.SetpNo". Now, you should be able to preset the SETPOINT quantity via an input field on the touchpanel instead. For that, an additional input/output field must be configured in the touchpanel screen (see picture).

What to Do

1. Close all objects that are open in the Editor.
2. Open the "DP_OP" data block in which the SETPOINT quantity is stored.
3. As well, open the "Conveyor" screen in the "Screens" folder of the Touchpanel.
4. Split the Editor area in two using the button shown in the picture.
5. Configure the input field for specifying the setpoint quantity by dragging the "SetpNo" tag from the data block into the screen using drag & drop.
6. Select the newly created input/output field and in the Inspector window in "Properties -> General -> Format" set the "Format pattern" (three digits). In addition, you can adjust how everything looks in the menus "Appearance" and "Text format".



12.8.5. Exercise 6: Compiling and Saving the Touchpanel



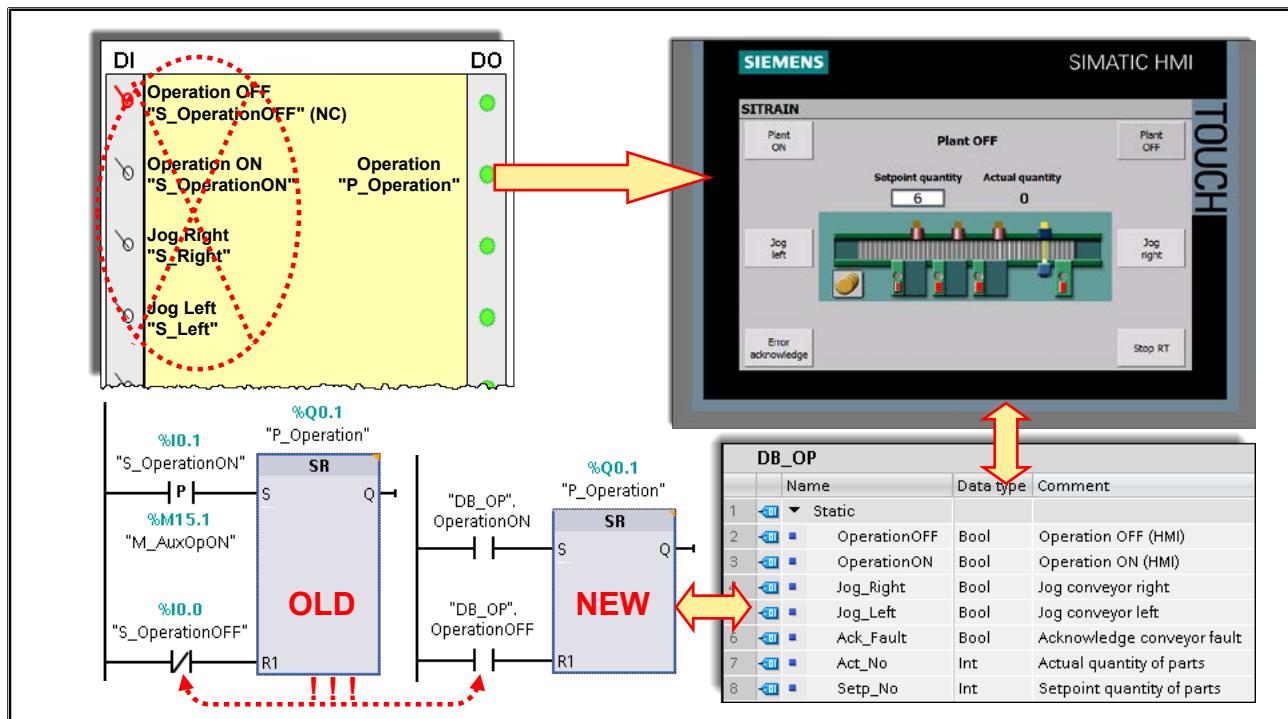
Task

You are to compile and save the now complete HMI project.

What to Do

1. Compile the HMI project by selecting the touchpanel in the Project tree and then clicking on the "Compile" button (see picture).
2. In the Inspector window under "Info", read the results of the compilation and eliminate any errors which may have occurred.
3. Save your project.

12.8.6. Exercise 7: Adjusting the STEP 7 Program



Task

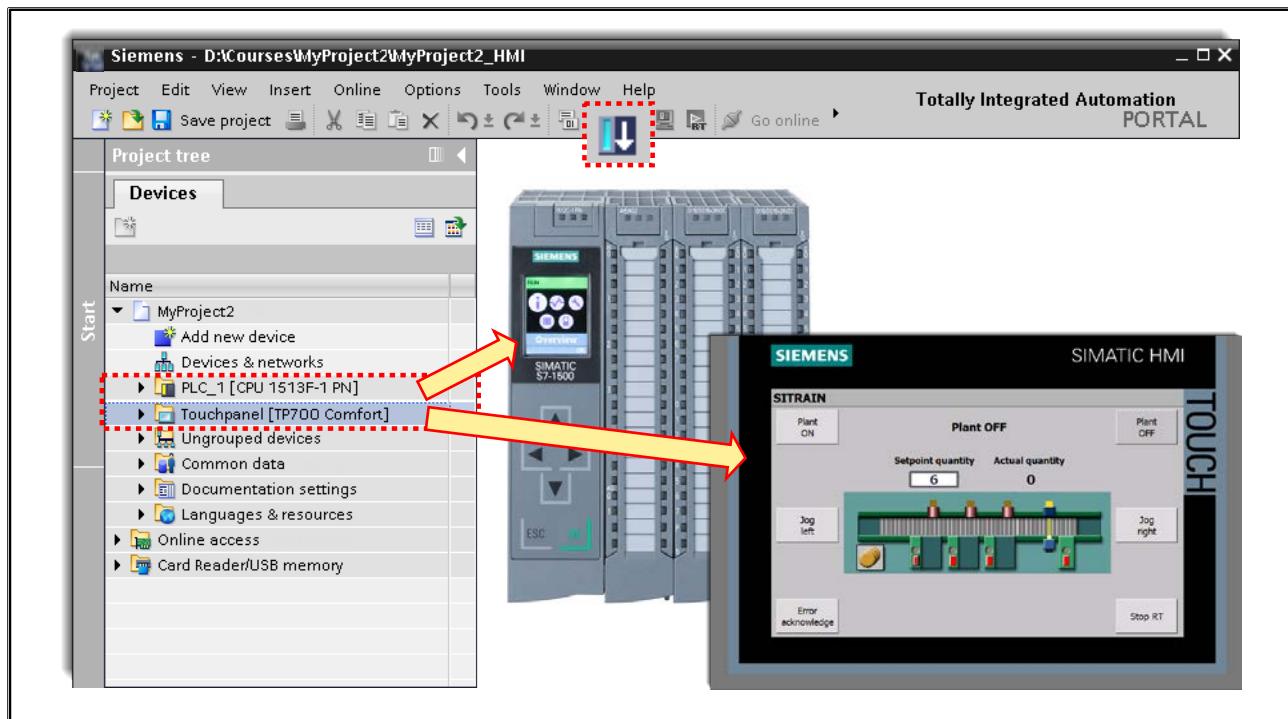
The S7 program of the controller is to be adjusted in such a way that ...

- the functions "Operation ON/OFF" and "Jog Right/Left" are no longer realized via the simulator switches but via the corresponding buttons on the touchpanel.
- the acknowledgement of a conveyor fault should still be possible via the simulator switch "S_Acknowledge" (I 0.7), and, in addition, also via the touchpanel, that is, the DB tag "DB_OP".AcknFault .
- The SETPOINT quantity is no longer a constant 3, but can be preset via an input/output field on the touchpanel.

What to Do

- In "FC_Mode", replace the variables "S_OperationON" (I 0.1) and "S_OperationOFF" (I 0.0) with the corresponding DB tags (see picture). Remember to also take into account that a hardware-based NC contact is connected to "S_OperationOFF" (I 0.0).
- Expand "FC_Fault" in such a way that a fault acknowledgement is still possible via "S_Acknowledge" (I 0.7) and, in addition, also via the touchpanel, that is, the DB tag "DB_OP".AcknFault .
- In "FC_Conveyor", replace the variables "S_Right" (I 0.2) and "S_Left" (I 0.3) with the corresponding DB tags.
- Compile and save your program.

12.8.7. Exercise 8: Downloading the HMI and CPU Project



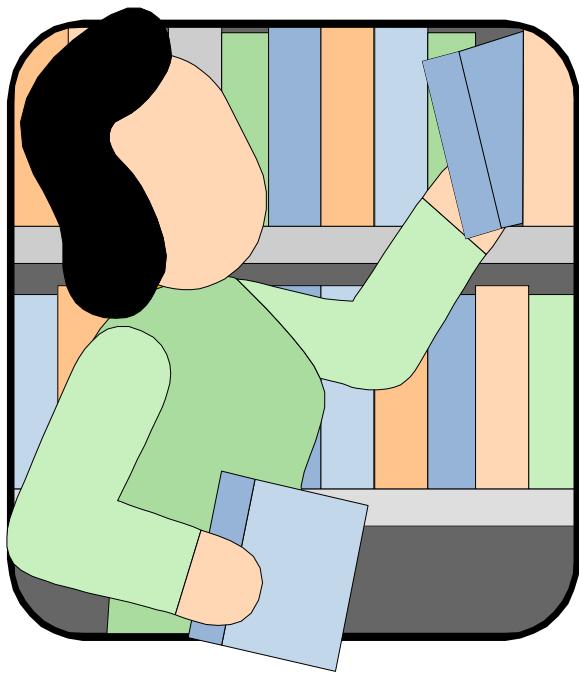
Task

From the now complete project, all S7 blocks are now to be transferred once more into the CPU and the entire Panel project is to be transferred into the touchpanel.

What to Do

1. Download all S7 blocks into the CPU.
2. Download the Panel project into the touchpanel.
3. Carry out a function test by
 - Switching the Operation ON and OFF via the touchpanel.
 - When Operation is OFF, jog the conveyor model to the right and left using the touchpanel.
 - When Operation is ON, preset a SETPOINT value and start transporting parts,
 - Monitor how the ACTUAL number is updated.
 - Cause a conveyor fault, monitor its display on the touchpanel and acknowledge it on the touchpanel.
4. Correct – if necessary – your program and check the functions once more.
5. Save your project.

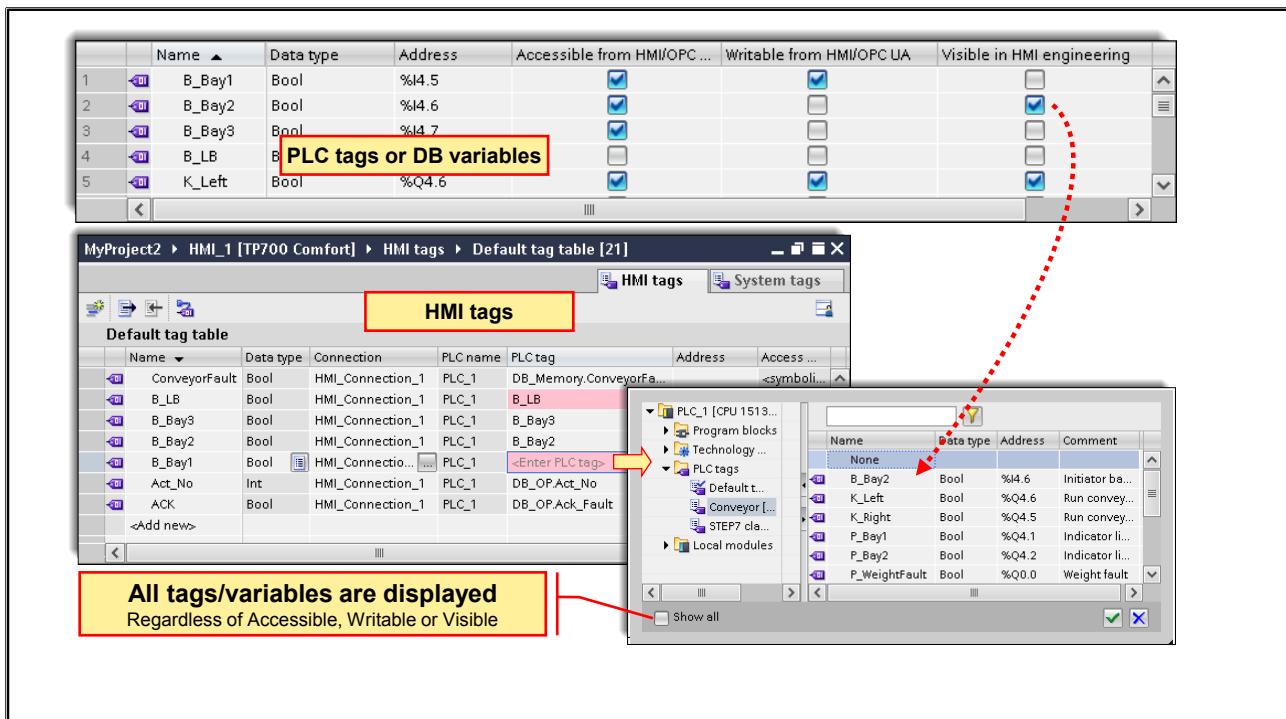
12.9. Additional Information



Note

The following pages contain either further information or are for reference to complete a topic.
For more in-depth study we offer advanced courses and self-learning mediums.

12.9.1. HMI/OPC UA Access to PLC Tags and DB Variables



HMI Tag Access

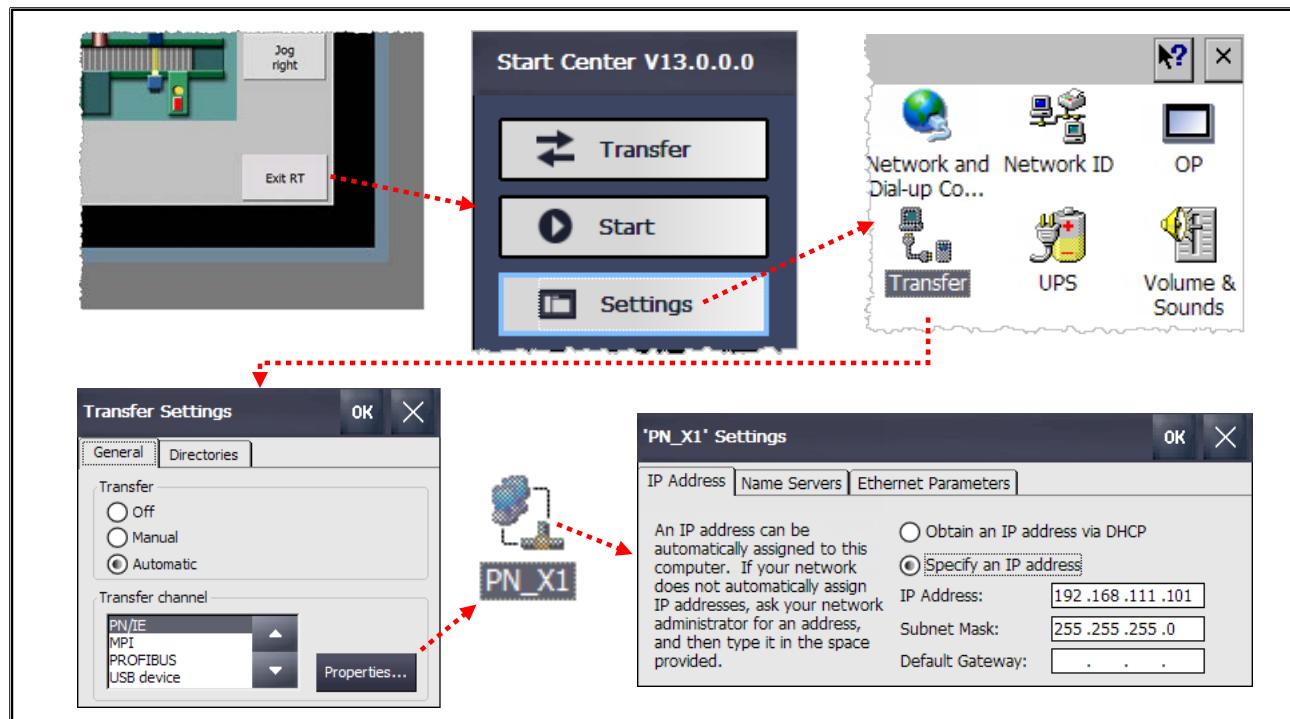
In the PLC tag tables and data blocks, protective mechanisms can be declared with whose help unwanted accesses to PLC tags and DB variables from HMI devices or OPC UA-Clients can be prevented:

- "Visible in HMI Engineering":
During HMI configuration, only tags/variables with the attribute "Visible in HMI Engineering" can be selected. This filter function can, however, be disabled in the selection dialog shown by activating "Show all".
- "Writable from HMI/OPC UA" (only S7-1200 and S7-1500):
This indicates whether the tag/variable can be written from HMI / OPC UA at runtime. This protective function integrated in the S7-1200/1500 operating system ensures that the HMI device or OPC UA-Client does not overwrite certain tags/variables.
- "Accessible from HMI/OPC UA" (only S7-1200 and S7-1500):
The HMI device can only access online the tags/variables which have the attribute "Accessible from HMI/OPC UA". This protective function integrated in the S7-1200/1500 operating system ensures that the HMI device or OPC UA-Client can neither read-access nor write-access certain tags/variables. Tags/variables which are not "Accessible from HMI/OPC UA", accordingly are also not "Visible in HMI Engineering".

Note:

OPC UA is a standard which enables operating system platform-independent data exchange.

12.9.2. Manually Setting the IP Address on the Panel



Manually Setting the IP Address of a Panel

You can manually set the IP address of the Panel's interface via the Start Center > Settings.

13

Contents

13. Organization Blocks.....	13-2
13.1. Types of Program Blocks.....	13-3
13.1.1. Organization Blocks of the S7-1500	13-4
13.2. S7-1500 Start and Cyclic Sequence.....	13-5
13.2.1. Interrupting the Cyclic Program	13-6
13.2.2. Process Image Partitions	13-7
13.3. Creating a New OB	13-8
13.3.1. OB Start Information using "OB_Startup" as an Example	13-9
13.4. Time-of-Day Interrupt OB.....	13-10
13.4.1. Starting the Time-delay Interrupt OB	13-11
13.4.2. Executing Cyclic Interrupt OBs	13-12
13.4.2.1. Phase Offset for "Cyclic interrupt" OBs	13-13
13.4.3. Hardware Interrupt	13-14
13.5. Task Description	13-15
13.5.1. Exercise 1: Preparing the Startup Initialization	13-16
13.5.2. Exercise 2: Initializing Transport using Startup and Programming the Time-delay Interrupt OB.....	13-17
13.6. Additional Task Description	13-18
13.6.1. Additional Exercise 3: Preparing for the Initialization Expansion.....	13-19
13.6.2. Additional Exercise 4: Initialization to the Left/Right.....	13-20
13.6.3. Additional Exercise 5: Displaying the Initialization.....	13-21
13.7. Additional Information	13-22
13.7.1. S7-1200/1500: Global Error Handling with Asynchronous Error OBs	13-23
13.7.2. S7-1200/1500: Global Error Handling with Synchronous Error OBs	13-24
13.7.3. OB Priorities and System Reaction	13-25

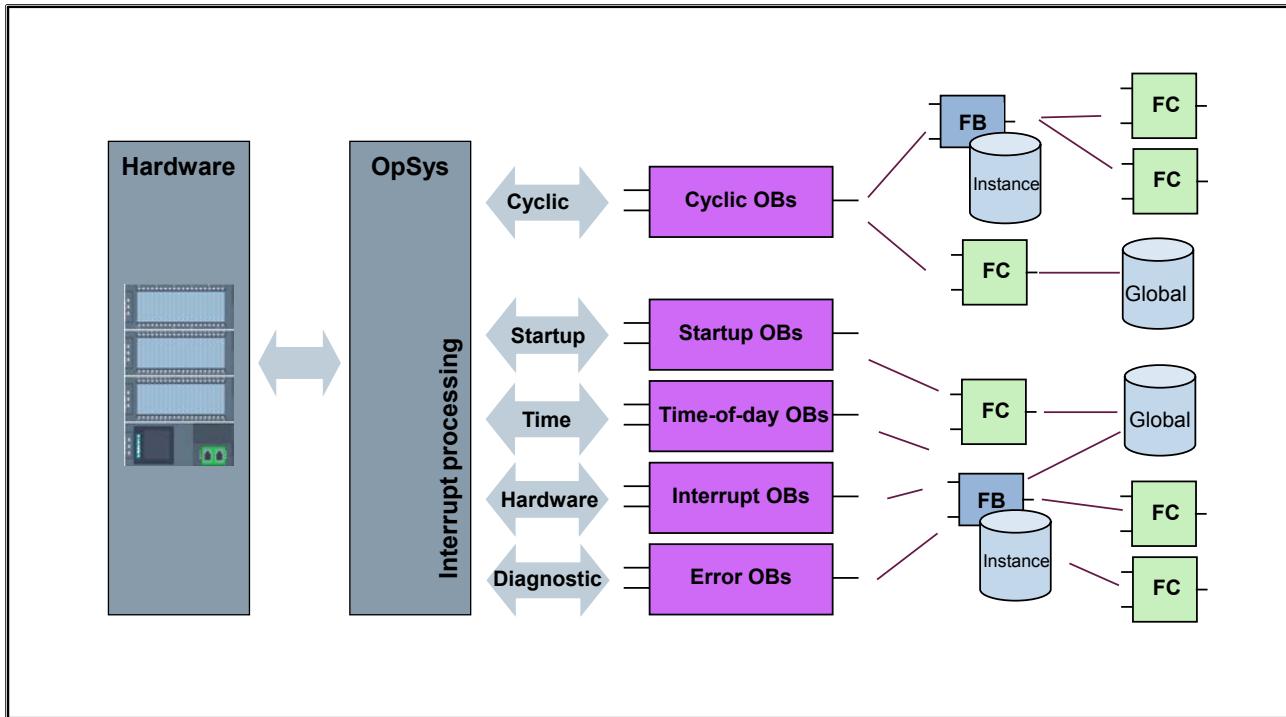
13. Organization Blocks

At the end of the chapter the participant will ...

- ... be familiar with the different types of organization blocks
- ... understand the principle of interrupt processing
- ... be familiar with the meaning of process image partitions
- ... be able to interpret the start information of OBs
- ... be able to use startup and time-delay interrupt OBs



13.1. Types of Organization Blocks



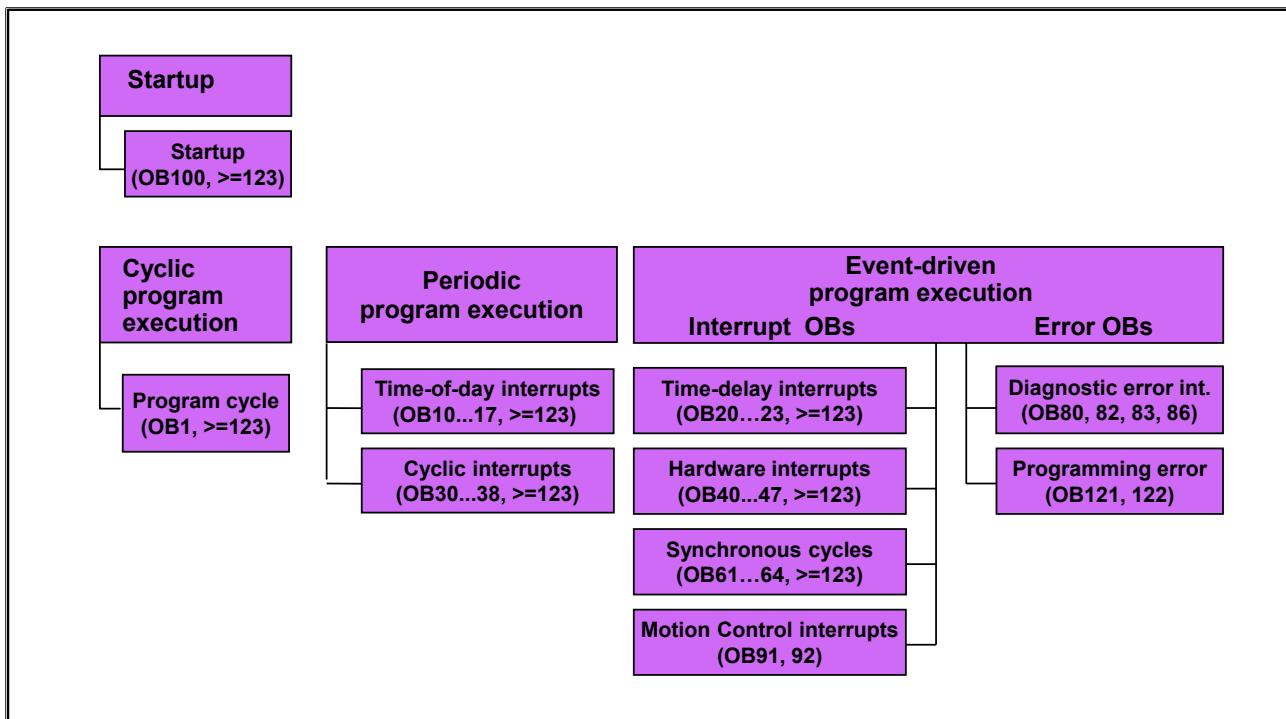
Organization Blocks (OBs)

Organization blocks (OBs) form the interface between the operating system and the user program. The organization blocks are called event-driven by the operating system.

The events can be cyclic, the STOP-RUN transition, time-dependent, hardware-dependent or an error.

If one of the events occurs, then the relevant OB is called if it is loaded and if its priority is greater than that of the OB currently being processed.

13.1.1. Organization Blocks of the S7-1500



Startup Program

After voltage recovery, or a change of operating mode (through the CPU's mode selector or through PG operation), a startup program is carried out in the Startup OBs before the cyclic program execution. In these Startup OBs you can, for example, carry out a pre-assignment of communication connections or initializations.

Cyclic Program Execution

The program stored in the Program Cycle OBs is executed in a continuous loop. With this cyclic program execution, the reaction time results from the execution time for the CPU's operating system and the sum of the command runtimes of all executed instructions. The reaction time, that is, how fast an output can be switched in relation to an input signal, amounts to a minimum of one time and a maximum of two times the cycle time.

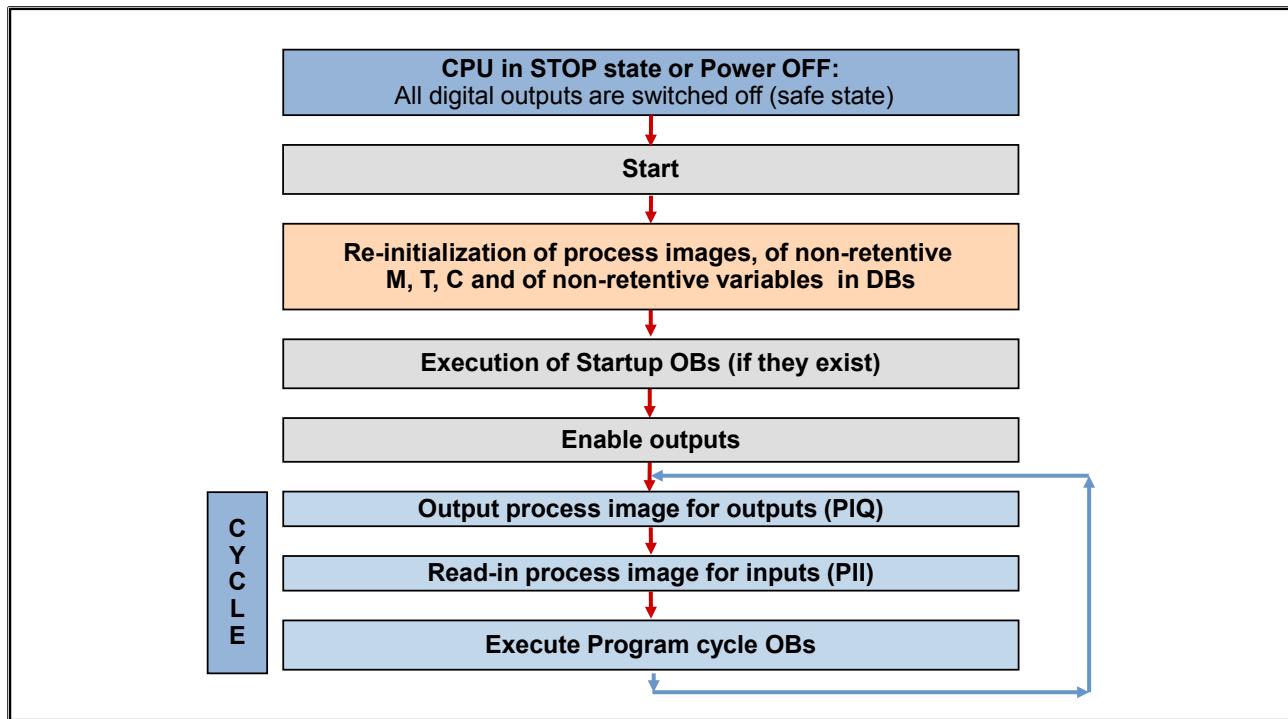
Periodic Program Execution

This makes it possible to interrupt the cyclic program execution at fixed intervals. With the Cyclic Interrupt OBs, an organization block (for example OB35) is executed after an adjustable time base (for example, every 100ms) has expired. In these blocks, for example, closed-loop control blocks with their sampling time are called. With the Time-of-day Interrupt OBs, an OB which could carry out a data backup, for example, is executed at a specific time, for example, every day at 17:00 hours (5 p.m.).

Event-driven Program Execution

Hardware interrupts are used to quickly react to process events. After an event occurs, the cycle is immediately interrupted and an interrupt program is executed. With Time-delay Interrupt OBs, a freely definable event can be reacted to with a time-delay; with the Error OBs, the user can influence the behavior of the controller in case there is an error.

13.2. S7-1500 Start and Cyclic Sequence



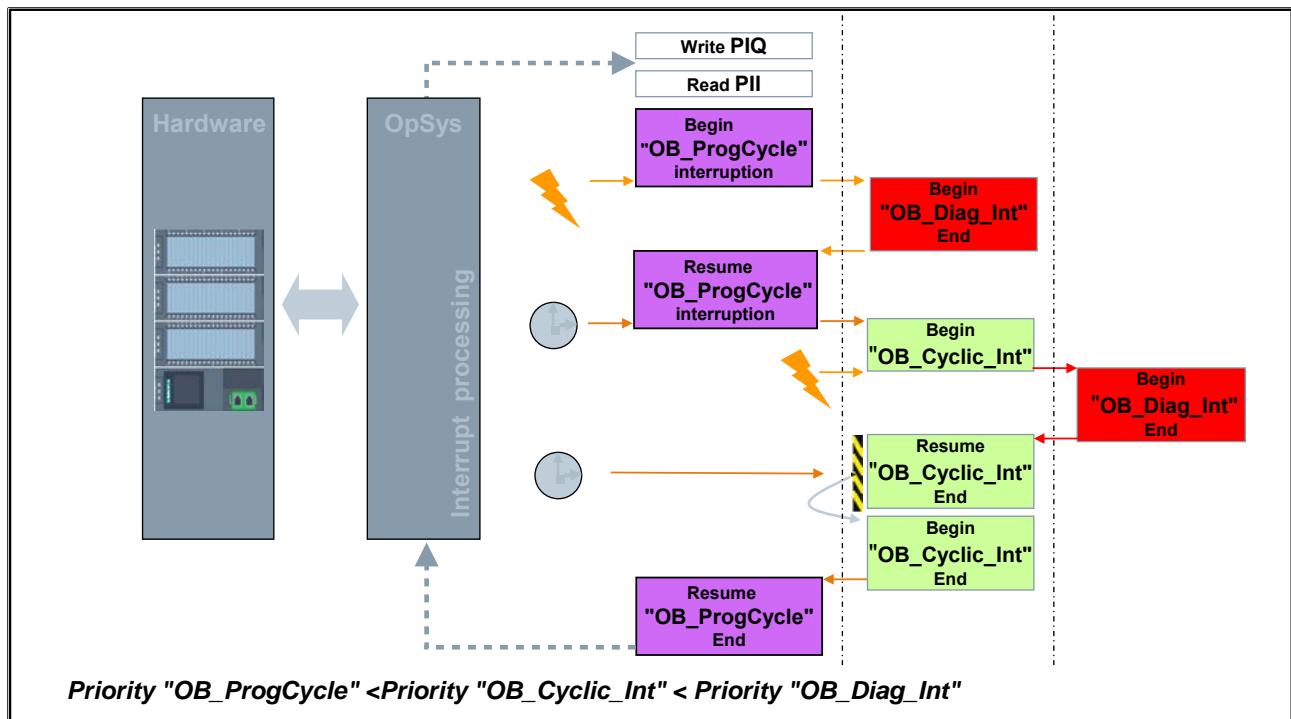
General

Before the CPU begins with the execution of the cyclic user program, a startup program is executed. In the startup program, initialization variables can be set by programming Startup OBs accordingly.

Warm Restart

The S7-1500 carries out a so-called warm restart in which the process images (PII, PIQ) and all non-retentive memory bits, timers and counters are deleted. Non-retentive DBs are reset to the start values of the load memory and retentive memory bits, timers and counters as well as retentive DB-contents are retained.

13.2.1. Interrupting the Cyclic Program



OB Calls

Organization blocks (OBs) form the interface between the CPU's operating system and the user program.

Organization blocks are called exclusively by the operating system. There are various start events (time-of-day interrupts, hardware interrupts - see picture) that each lead to the start of their associated organization block.

Interrupting the Cyclic Program

When the operating system calls another OB, it interrupts the cyclic program execution because a "Program Cycle"-OB has the lowest priority. Any other OB can therefore interrupt the main program and execute its own program. Afterwards, the "Program Cycle"-OB resumes execution at the point of interruption.

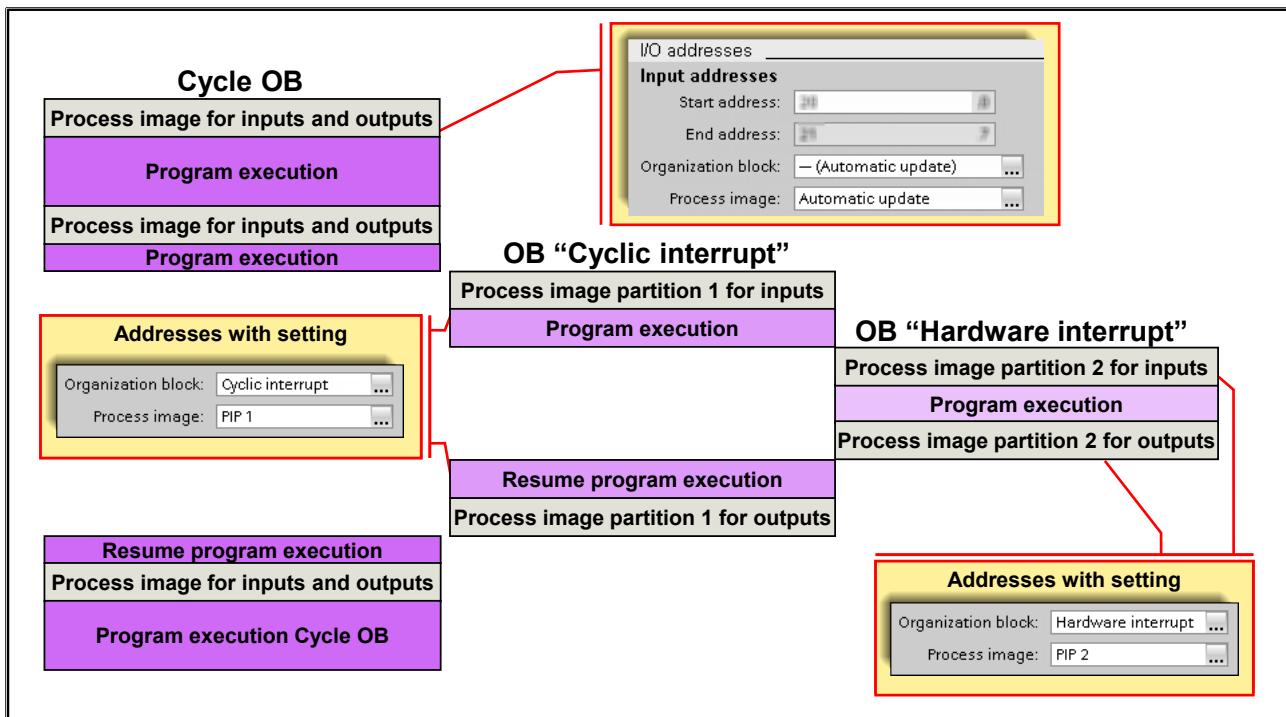
Priorities

The S7-1500 CPUs support the priorities 1 (lowest priority) to 26 (highest priority).

The OBs are executed on a purely priority-driven basis. This means that when several OB requests exist at the same time, the OB with the highest priority is executed first. When an event occurs that has a higher priority than the currently active OB, this OB is interrupted. Events of the same priority are executed in the order that they occur.

If this is also the same, for example for Startup OBs, then the OBs are executed according to their number.

13.2.2. Process Image Partitions



Process Image for Inputs and Outputs

For reasons of access speed and the consistency of the status of individual inputs throughout the entire cycle, a copy is stored in the process image for inputs (PII) which is accessed during program execution. The writing of outputs occurs in the process image for outputs (PIQ). The statuses of outputs stored here are written into the actual outputs at the beginning of every cycle.

The inputs and outputs of an S7-1500 are automatically made available in the process image for inputs and outputs as long as this is not changed in the settings of the individual blocks.

Under the Properties of the module > General > Input.../Output... > I/O addresses, "Automatic update" ("Automatische Aktualisierung") is set in *Organization block* and *Process image*.

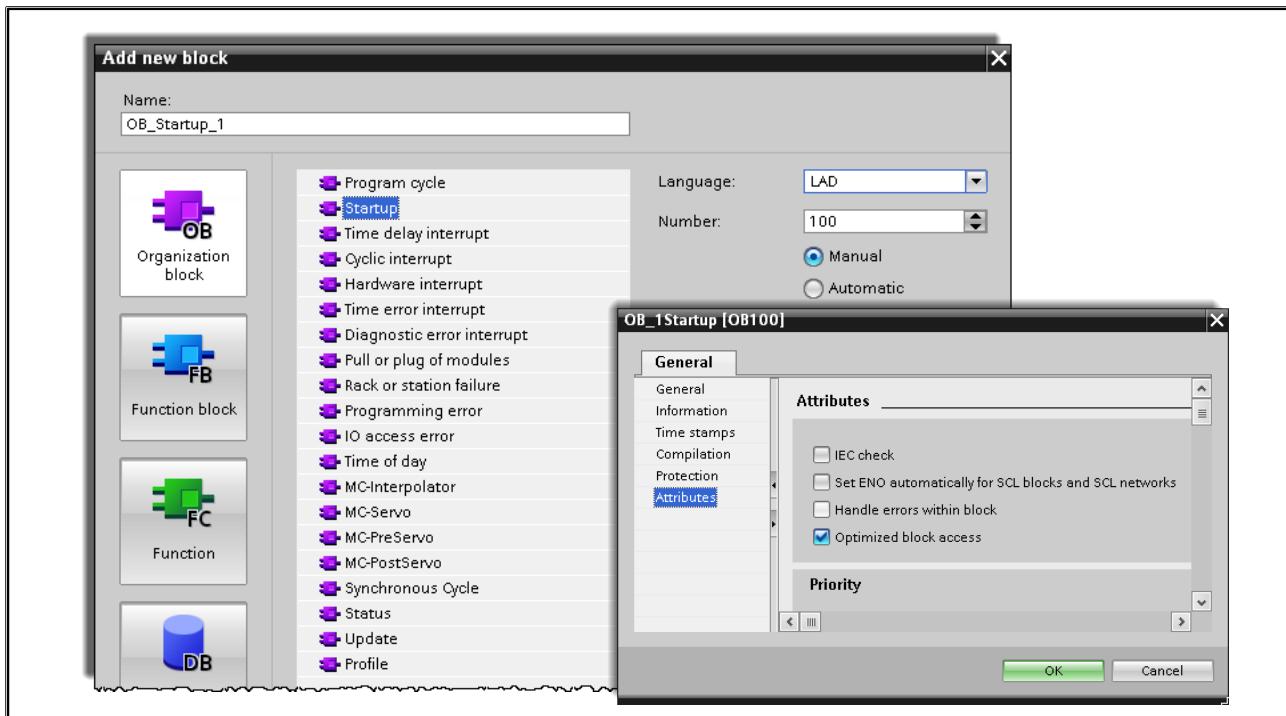
Process Image Partition (PIP)

Inputs and outputs which are not used in the cyclic program, that is, not in every cycle, do not have to be updated with the process image at the beginning of every cycle. For that reason, there are the process image partitions which in turn can be assigned to individual organization blocks. If an OB is executed which has a PIP assigned to it, then the inputs of the associated PIP are read-in at the beginning and at the end, the statuses of the outputs of the PIP are written in the relevant peripherals. If a process image partition is not assigned to any organization block, then the update of this PIP must be done with the help of instructions.

Updating a Process Image Partition in the User Program

Each PIP can be updated in the user program with special instructions. For this, the instructions "UPDAT_PI" for updating inputs and "UPDAT_PO" for updating outputs is used.

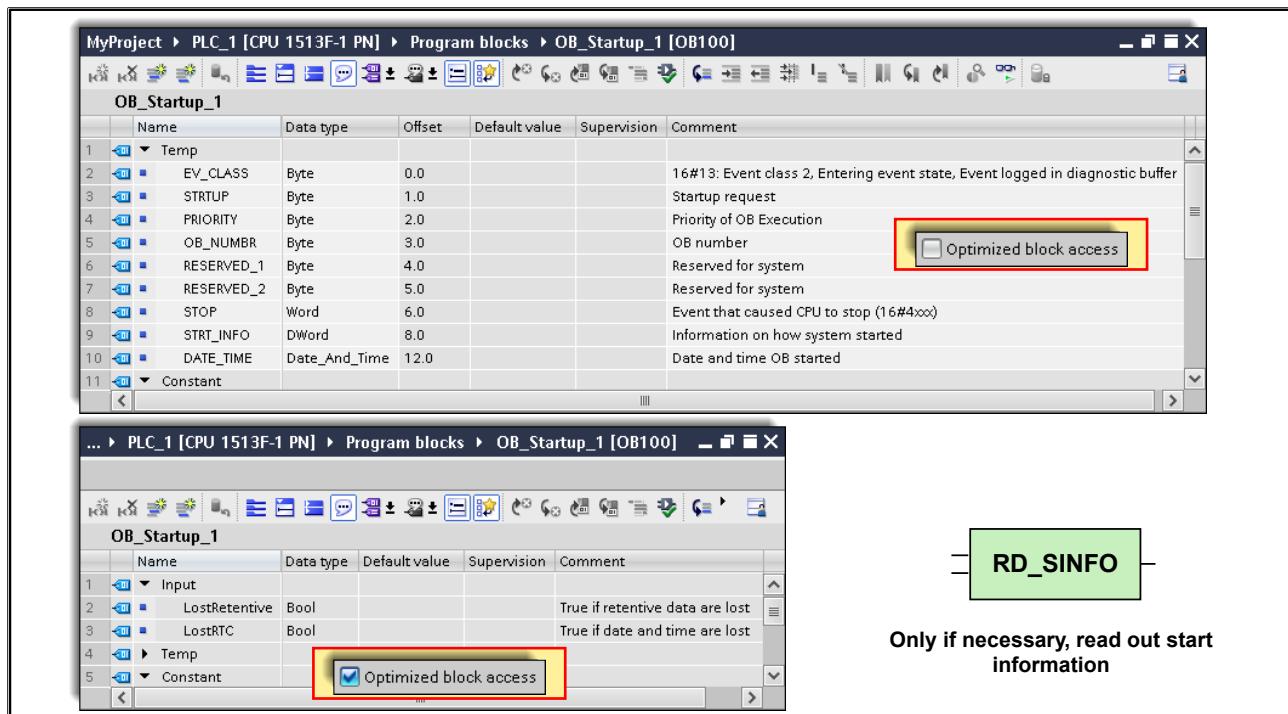
13.3. Creating a new OB



Create new OB

When creating an organization block, the type of event is first of all selected. In addition, the number and the name can be changed. By default, new OBs are created with the attribute "Optimized block access" with the result that only reduced start information is available in the OBs (see next page). In the Properties of the OB, the behavior can be individually set depending on the type.

13.3.1. OB Start Information using "OB_Startup" as an Example



Start Information for Not Optimized Block Access

When the operating system calls organization blocks, the user is provided with OB-specific start information on the local data stack.

This start information has a length of 20 bytes and is available after the OB starts execution.

The start information as well as their absolute L-stack addresses is only completely available for those OBs where the block attribute "Optimized block access" is not activated (as shown in the picture).

In order to avoid errors, the structure of the standard declaration section should not be changed by the user. Following the standard declaration section, the user can declare further, additional temporary variables.

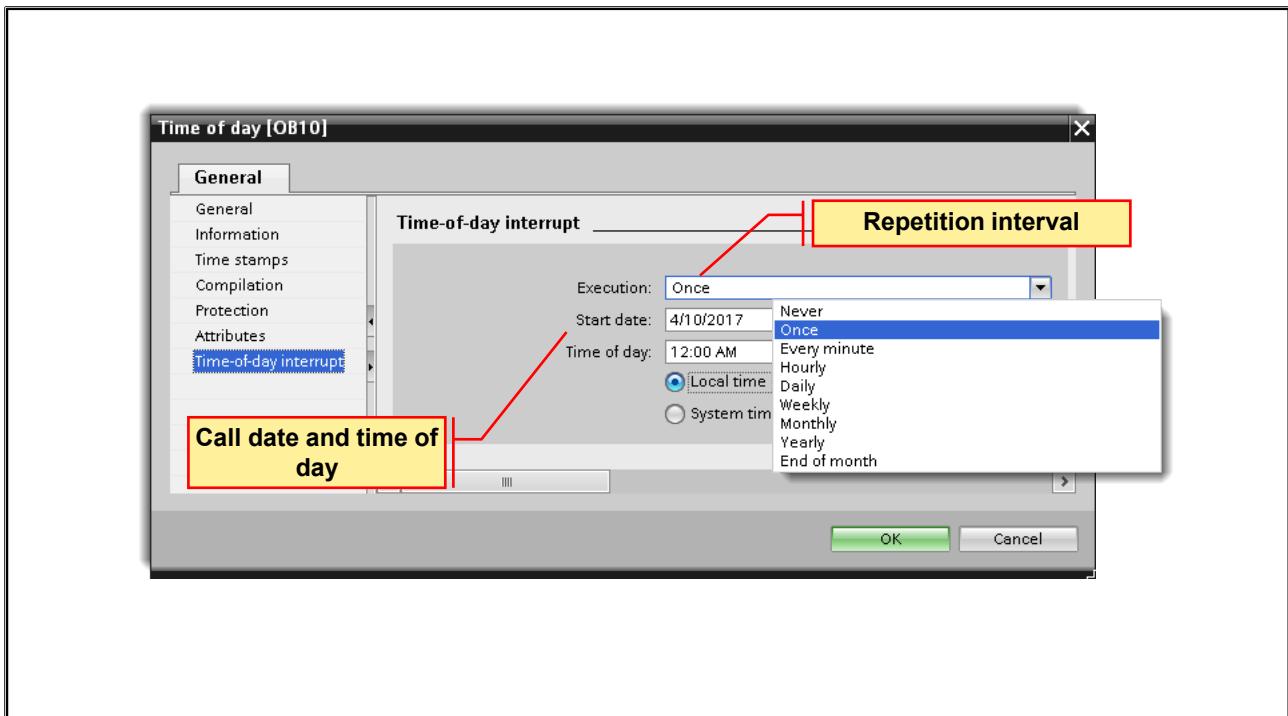
Start Information for Optimized Block Access

The start information of an organization block with optimized block access is limited to the essentials and is passed by means of input parameters. If necessary, the start information, with the exception of the date and time, can be read out by means of the function "RD-SINFO".

Variables

An explanation of the meaning of the variables can be found in the online help.

13.4. Time-of-Day Interrupt OB



Time-of-Day Interrupts

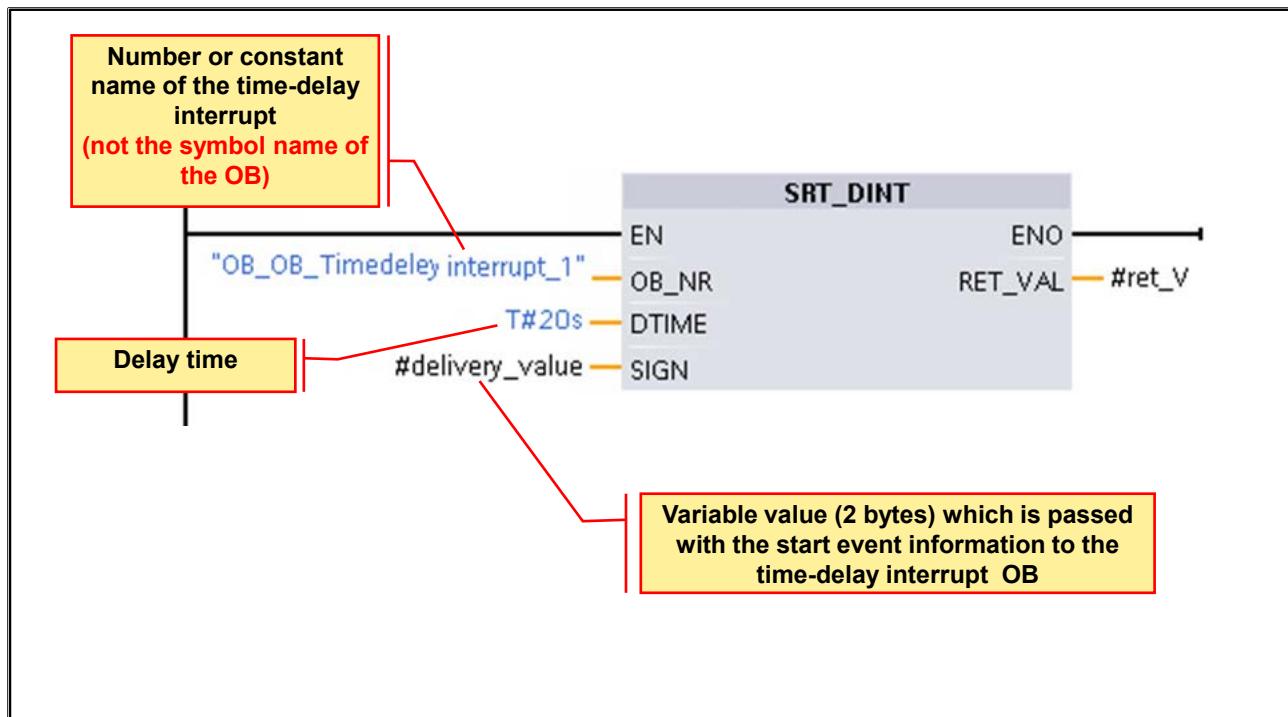
Time-of-day interrupts are used for executing a certain program called in OB 10 (as an example) either once only at a certain time or periodically (every minute, hourly, daily, weekly, monthly, yearly) starting at that time.

Note

In addition, the time-of-day interrupts can be controlled at runtime with the following instructions (Task Card Instructions "Extended instructions -> Interrupts > Time-of-day interrupt"):

- "SET_TINT" Set start date, time and period
- "SET_TINTL" Set start date, time and period
- "CAN_TINT" Cancel time-of-day interrupt
- "ACT_TINT" Activate time-of-day interrupt
- "QRY_TINT" Query time-of-day interrupt

13.4.1. Starting the Time-delay Interrupt OB



Time-Delay Interrupts

Time-delay interrupt OBs are used in order to be able to react to freely definable events after a time delay. With the function "SRT_DINT", you define which time-delay interrupt after expiration of which time is to be called by the operating system. With the help of the input parameter "SIGN", a value in the size of a word can be passed to the time-delay interrupt OB.

Caution:

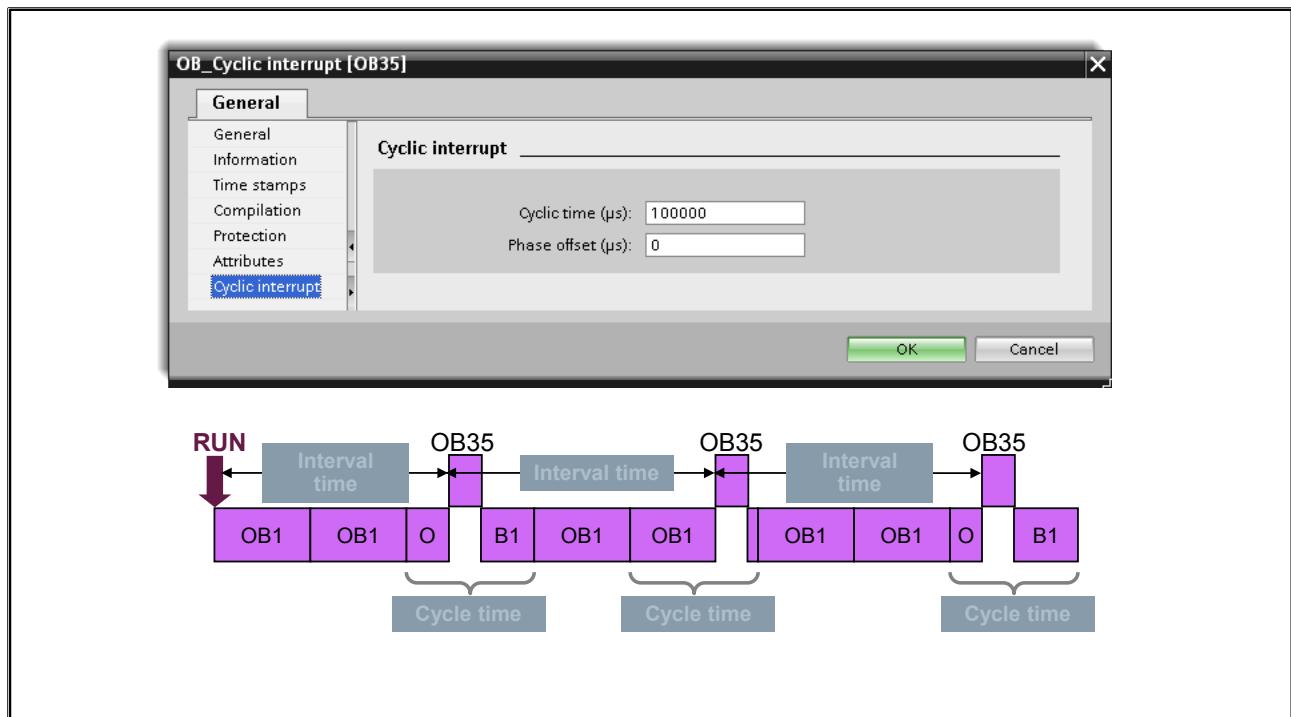
As the actual parameter of the formal parameter "OB_NR", the OB number or the constant name of the OB must be specified. The symbol name of the OB to be called is not acceptable. You will wind the constant name in the Properties of the OB under "General > Constant name".

Note

In addition to the instruction "RSD_DINT", there are also other instructions in the Task Card Instructions under "Extended instructions -> Interrupts > Time delay instruction":

- "CAN_DINT" Cancel time delay interrupt
- "QRY_DINT" Query status of time delay interrupt

13.4.2. Executing Cyclic Interrupt OBs



Cyclic Interrupt

With a Cyclic interrupt OB, a block can be executed at fixed time intervals. The S7-1500 offers the OB 35, for example, as a Cyclic interrupt OB. The default setting for its call interval is 100000μs; the selectable range is from 500μs to 60000000μs (60sec).

Interval Time

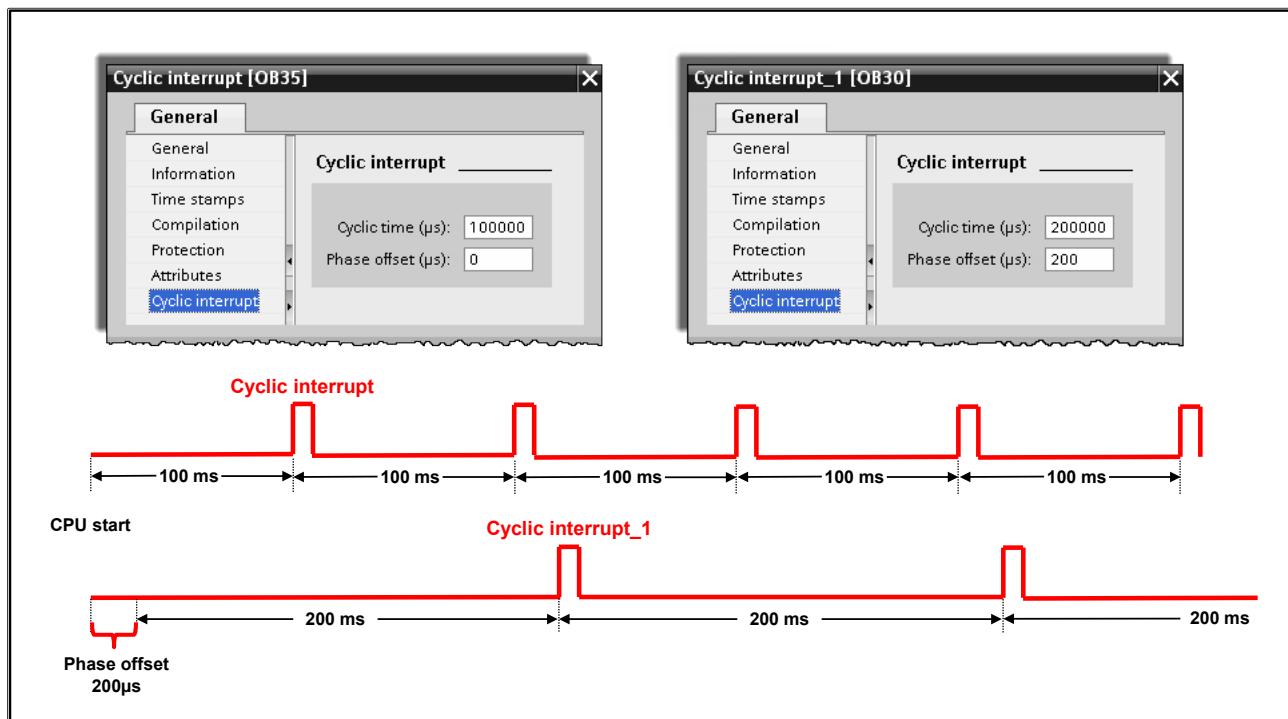
You must make sure that the interval you specify is longer than the time required for execution. The operating system calls the "Cyclic interrupt" OB at the specified time. If the "Cyclic interrupt" OB is still active at this time, the operating system calls the "Time error interrupt" (OB 80).

Note

Cyclic interrupts can also be controlled and queried at runtime with "Extended instructions" (Task Card Instructions "Extended instructions -> Interrupts > Cyclic interrupt"):

- "SET_CINT" Set cyclic interrupt parameters
- "QRY_CINT" Query cyclic interrupt parameters

13.4.2.1. Phase Offset for "Cyclic interrupt" OBs



Example for the Use of a Phase Offset

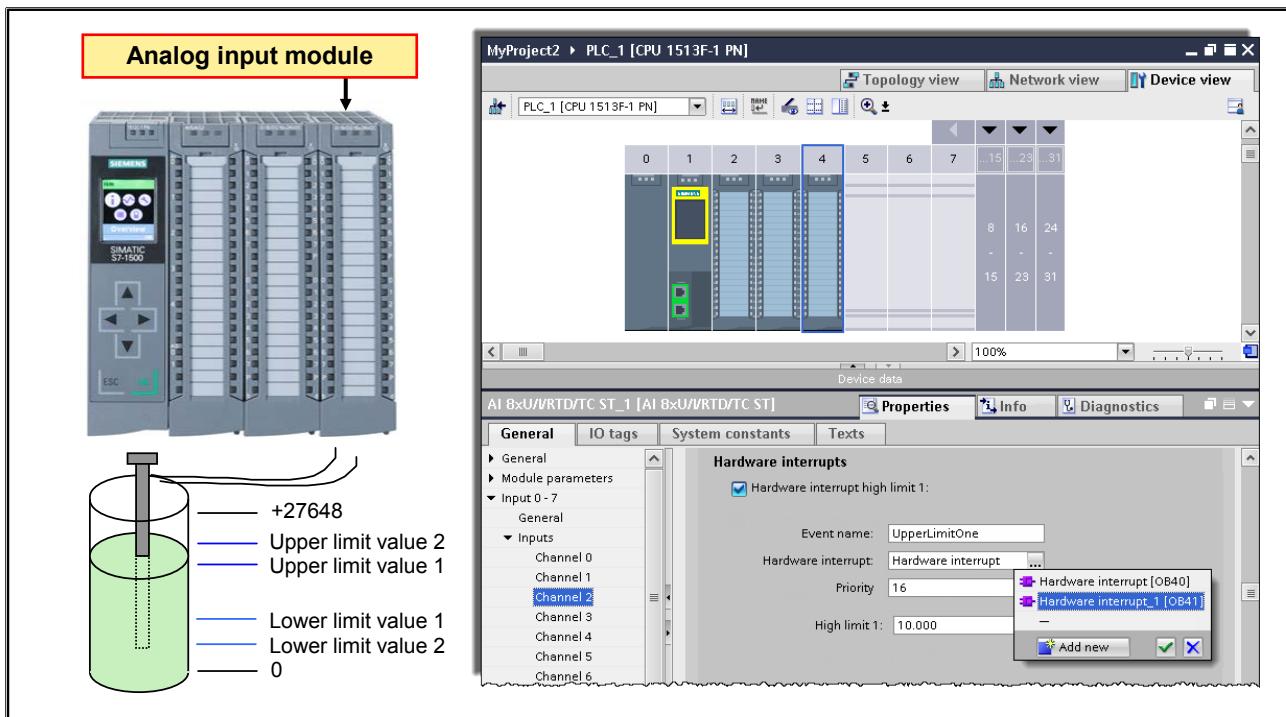
Two "Cyclic interrupt" OBs are required in the program:

"Cyclic interrupt"

"Cyclic interrupt_1"

For the OB "Cyclic interrupt" and for the OB "Cyclic interrupt_1", a time period of 100 ms was set. After the time period of 100 ms has expired, both "Cyclic interrupt" OBs are given their starting time. However, in order to execute the OBs with a time lag, a phase offset is configured for one of the two OBs.

13.4.3. Hardware Interrupt



Hardware Interrupt

The program execution of a "Hardware interrupt" OB is started as soon as a certain event occurs.

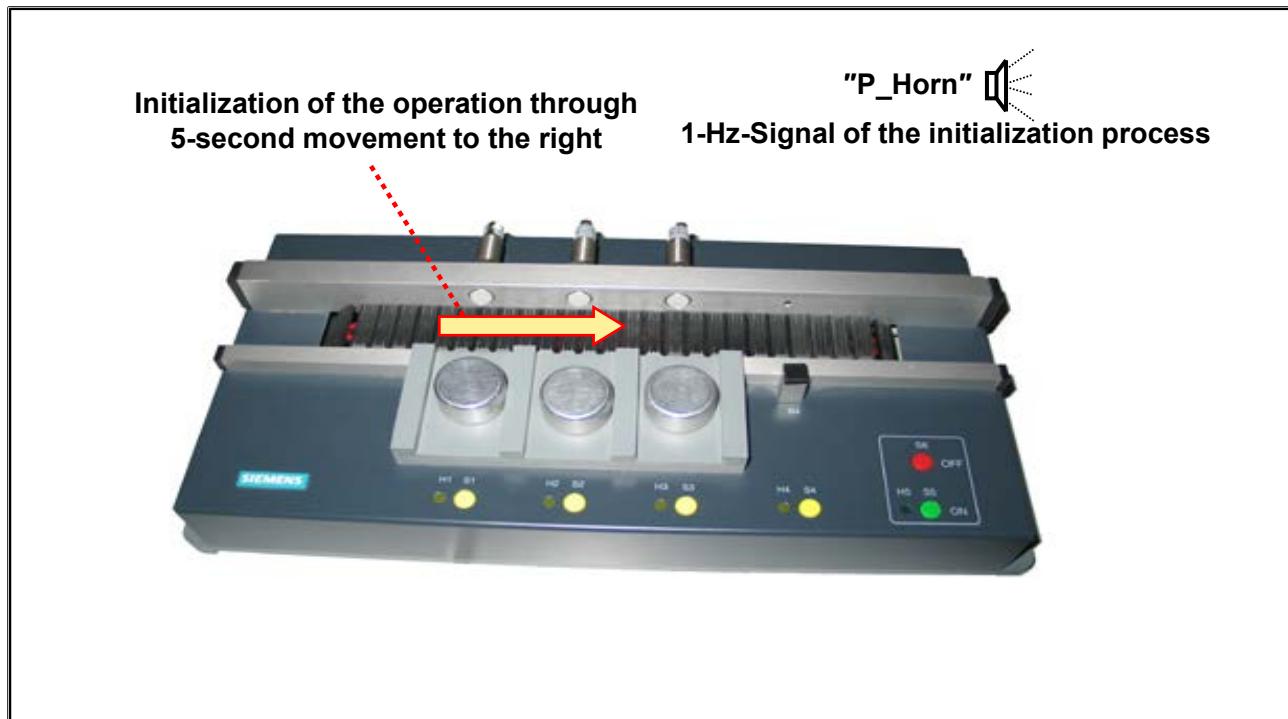
Hardware interrupts can be triggered by various module-specific signals:

For parameter-assignable signal modules (DI, DO, AI, AO) you specify which signal is to trigger the hardware interrupt in the Properties of the modules.

Example

In configuring an analog input module, suitable limit values were specified in the above example. If the measured value then exceeds this limit, an interrupt is triggered on the CPU which causes the program to be interrupted and the OB "Hardware interrupt" to be called for execution.

13.5. Task Description



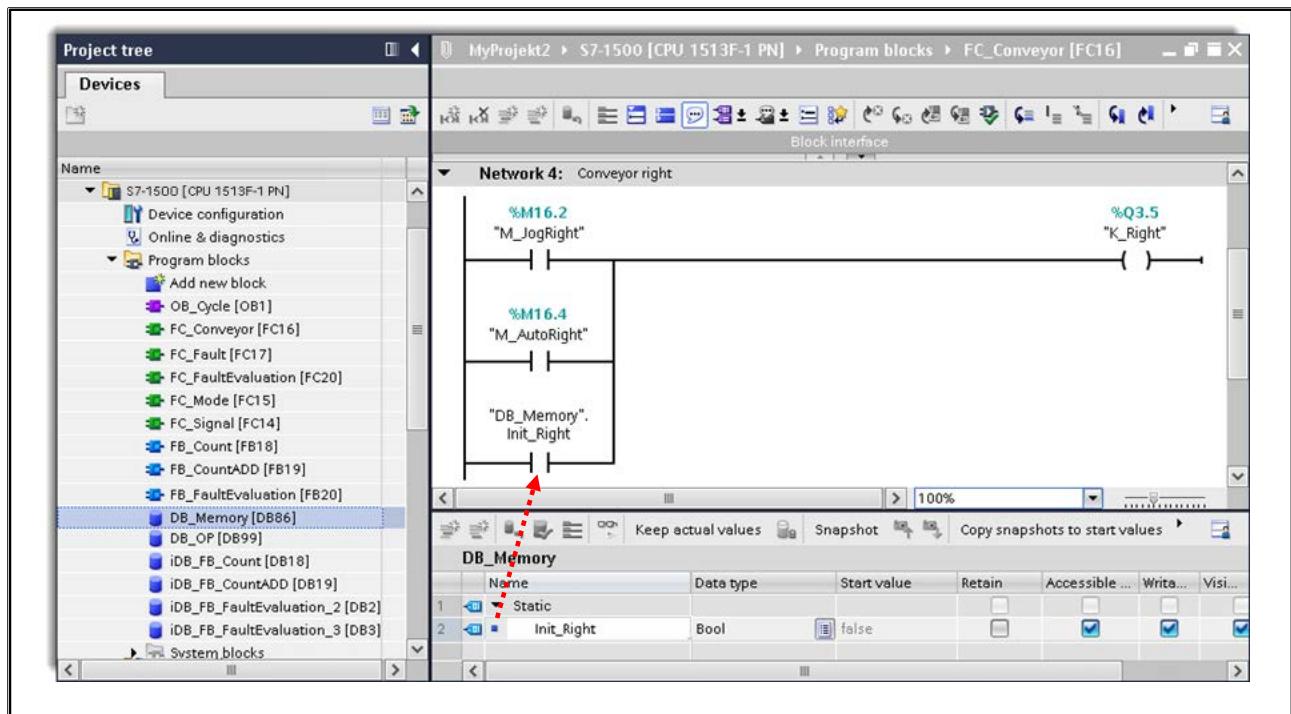
Task Description

In order to prevent parts being on the transport conveyor after switch on or warm restart, the conveyor is to move to the right for 5 seconds when there is a STOP-RUN transition.

For this, the conveyor is started with the help of a Startup OB during startup and is stopped again with the help of a Time-delay interrupt OB which is called after 5 seconds.

The initialization run is signaled with a 1 Hz signal at the horn.

13.5.1. Exercise 1: Preparing the Startup Initialization

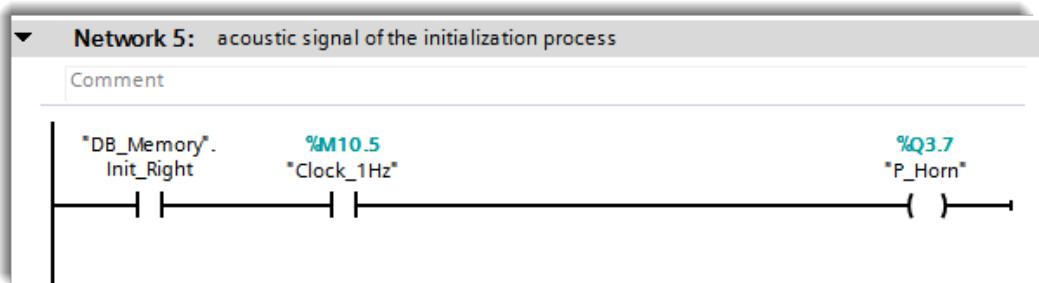


Task

So that the conveyor moves to the right after starting, a variable is required which has the Status True during the initialization time.

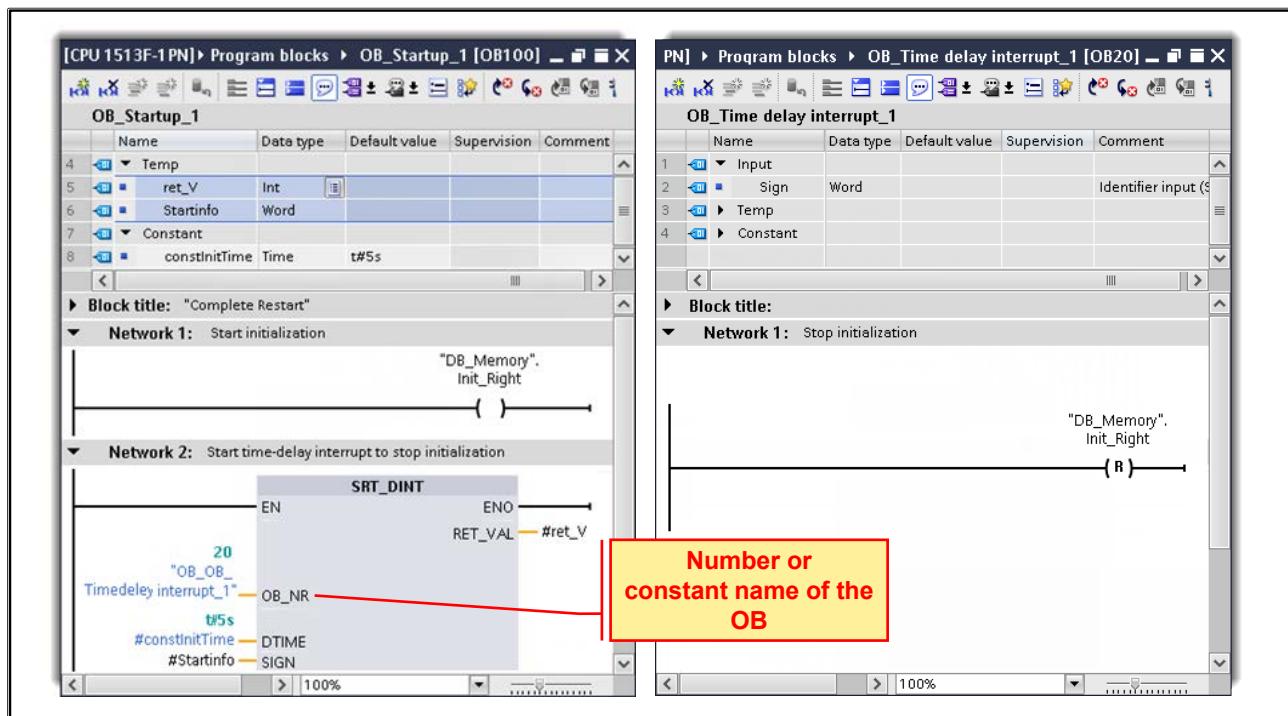
What to Do

1. Insert the new data block "DB_Memory" and declare the variable "Init_Right".
2. Link the variable "DB_Memory".Init_Right in "FC_Conveyor" as an additional OR-condition for transport conveyor movement to the right.
3. Program the acoustic signal of the initialization process in "FC_Signal". (see picture).



4. Save your project.

13.5.2. Exercise 2: Initializing Transport using Startup and Programming the Time-delay Interrupt OB



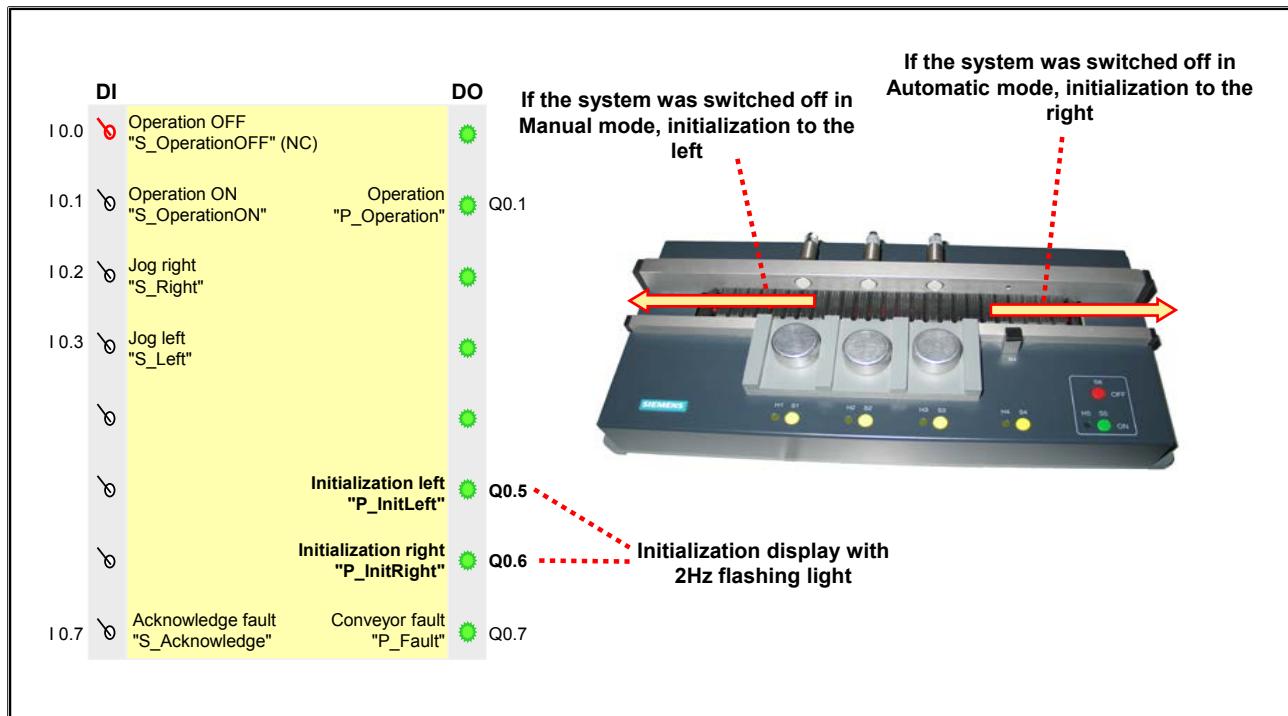
Task

The new variable "DB_Memory".Init_Right must be set in a Startup OB and reset in a Time-delay interrupt OB after 5 seconds.

What to Do

1. Insert the new organization blocks "OB_Startup_1" of the type Startup and "OB_Time delay interrupt_1" of the type Time-delay interrupt.
2. In the OB "OB_Startup_1", make an assignment to the variable "DB_Memory".Init_Right so that it is assigned the value TRUE.
3. In a further network, call the function "SRT_DINT" and declare it so that the OB "Time delay interrupt_1" is started after 5 seconds.
Caution: The parameter OB_NR only accepts the OB number or the constant name of the OB not the symbol name.
4. In order to be able to supply the parameters SIGN and RET_VAL with actual parameters, declare the relevant temporary variables "#Startinfo" (WORD) and "#ret_V" (INT).
5. In "OB_Time delay interrupt_1", reset the variable "DB_Memory".Init_Right.
6. Save, compile and download your project.
7. Test the new function.

13.6. Additional Task Description

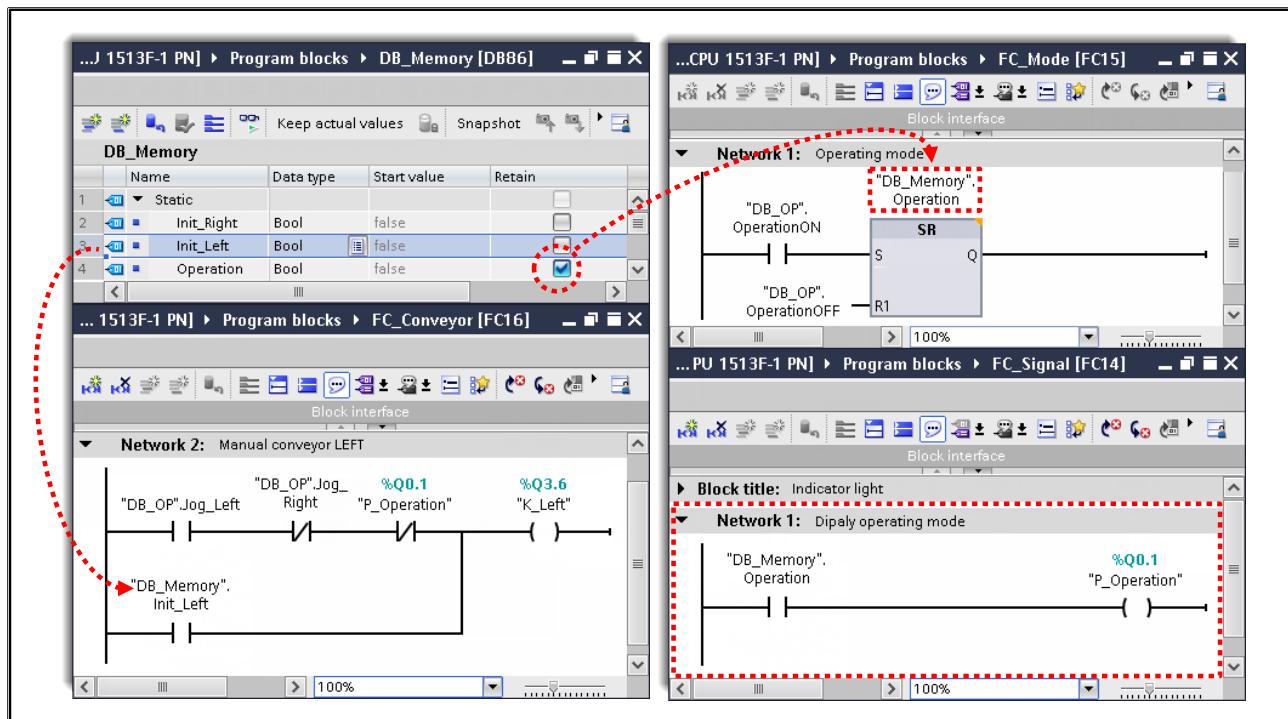


Task Description

When switching on the system, the initialization is to depend on the operating status at the time when the system was switched off. If the system had the operating status Operation ON (Automatic) when it was switched off, then the transport conveyor is to move to the right. If the system had the operating status Operation OFF (Manual), when it was switched off, then the initialization movement is to the left. In addition, the relevant initialization movement is signaled with a 2Hz flashing light at the LEDs Q0.5 "P_InitLeft" and Q0.6 "P_InitRight".

In order to achieve this, the operating mode must be stored as retentive.

13.6.1. Additional Exercise 3: Preparing for the Initialization Expansion



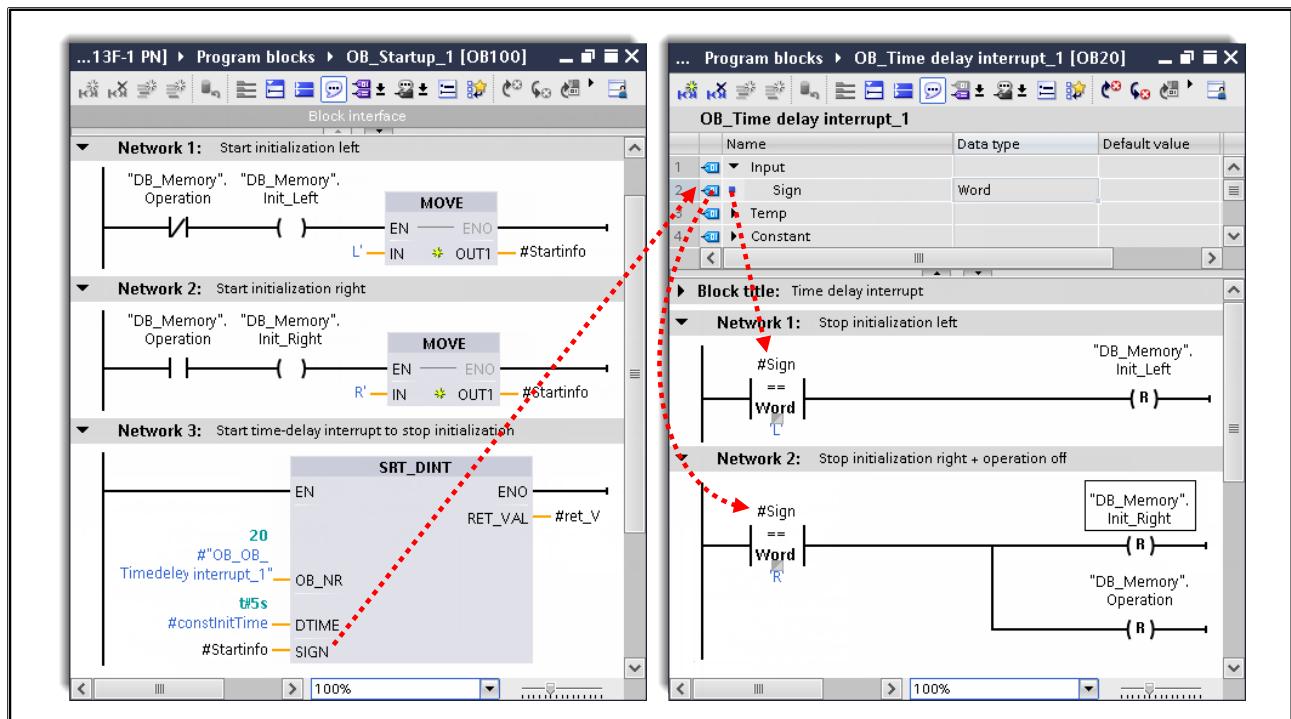
Task

You are to declare a variable for the movement to the left and assign it as a condition for a movement to the left. Furthermore, a retentive variable is required for the operating mode.

What to Do

1. In "DB_Memory", declare two new variables "Init_Left" and "Operation".
2. Give the variable "Operation" the property "Retentive".
3. In "FC_Conveyor", insert a new OR logic operation for the control of the output "K_Left" and as a further condition give the variable "DB_Memory".Init_Left a TRUE signal.
4. For the operating mode in "FC_Mode", set the variable "DB_Memory".Operation instead of the output "P_Operation".
5. To signalize the operating mode in "FC_Signal", assign the status of the variable "DB_Memory".Operation to the output "P_Operation".
6. Save the program modifications.

13.6.2. Additional Exercise 4: Initialization to the Left/Right



What to Do

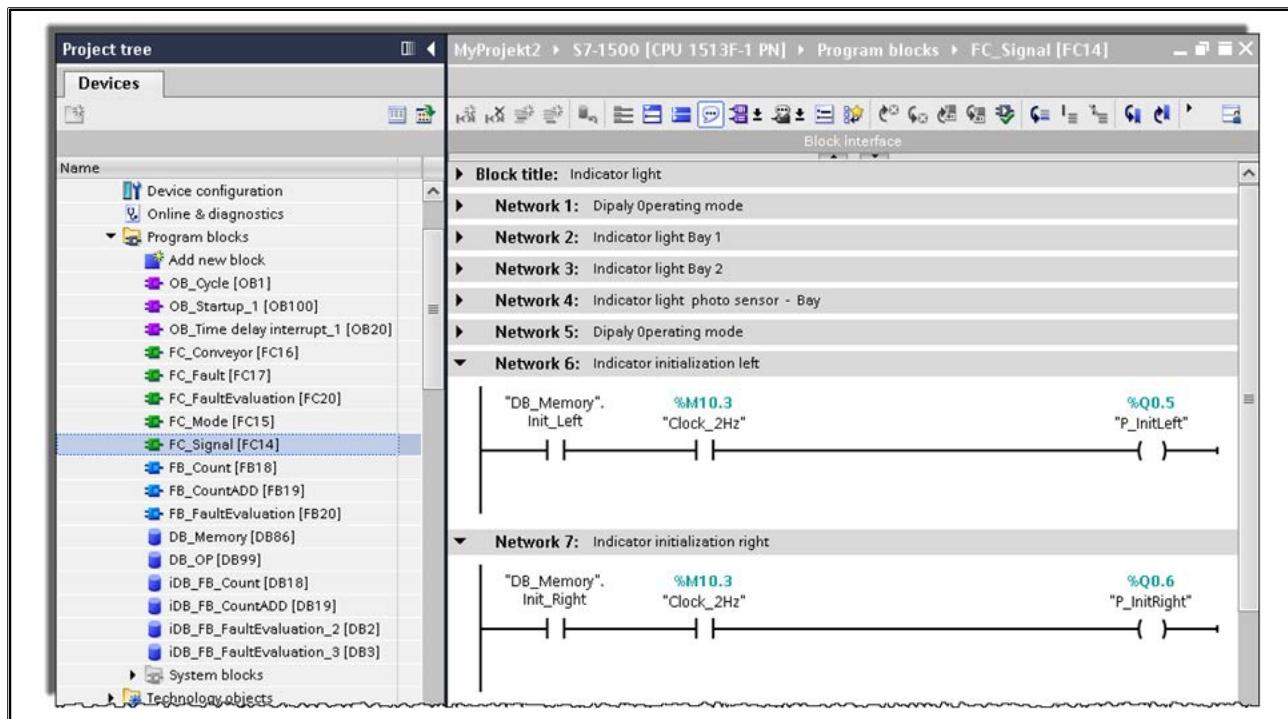
In the Startup OB, after evaluating the variable "DB_Memory".Operation, the variable "DB_Memory".Init_Right or "DB_Memory".Init_Left is assigned the value TRUE.

Furthermore, the character 'L' for left or 'R' for right is written in the temporary variable #Startinfo. The call of "OB_Time delay interrupt_1" remains unchanged. However, after evaluating the input parameter #Sign, the variables "DB_Memory".Operation and "DB_Memory".Init_Right or "DB_Memory".Init_Left are reset in this.

Solution Hints

1. In "OB_Startup_1", assign the variable "DB_Memory".Init_Left the value TRUE and write the CHAR value 'L' in the variable #Startinfo when the variable "DB_Memory".Operation has the status FALSE.
2. Change the assignment of the variable "DB_Memory".Init_Right so that it only gets the value TRUE when the variable "DB_Memory".Operation has the status TRUE. Additionally in this case, the CHAR value 'R' is written in the variable #Startinfo.
3. The call of the function "SRT_DINT" remains unchanged.
4. Reset the variable "DB_Memory".Init_Left in "OB_Time delay interrupt_1" when the CHAR value 'L' is passed with the input parameter #Sign.
5. If the CHAR value 'R' is passed with the parameter #Sign, then the variables "DB_Memory".Operation and "DB_Memory".Init_Right are reset.
6. Save the program modifications.

13.6.3. Additional Exercise 5: Displaying the Initialization



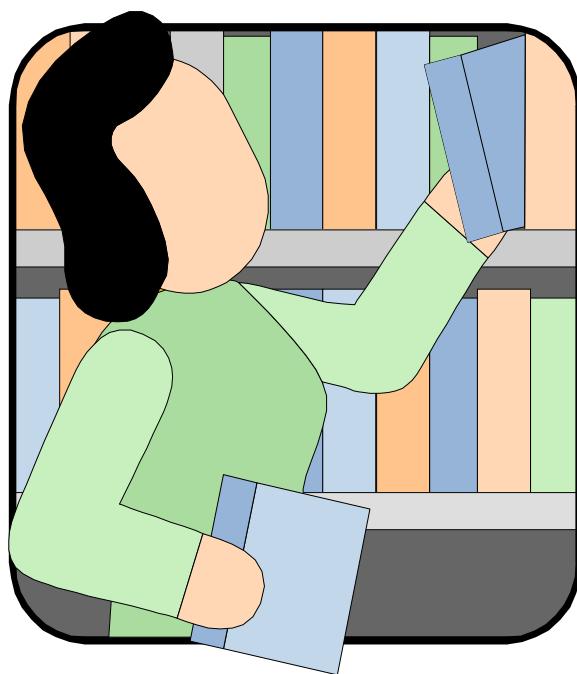
Task

The current initialization is to be displayed on the relevant LED "P_InitLeft" or "P_InitRight" with a 2Hz flashing light.

What to Do:

1. Open the function "FC_Signal" and insert two new networks.
2. With an AND instruction, interconnect the variable "DB_Memory".Init_Left and the clock memory "Clock_2Hz" and assign the result to the output Q0.5.
3. Give the output Q0.5 the symbol name "P_InitLeft".
4. Repeat the logic operation with the variable "DB_Memory".Init_Right and "Clock_2Hz" for the output Q0.6 and give it the symbol name "P_InitRight"
5. Save, compile and download the program modifications.
6. Test the new function.

13.7. Additional Information



13.7.1. S7-1200/1500: Global Error Handling with Asynchronous Error OBs

Type of Error	Example	Error OB	Priority
Time Error: Max. allowed cycle time exceeded once System reaction with OB: RUN without OB: STOP Max. allowed cycle time exceeded by more than double System reaction with OB: STOP	Exceeding the max. allowed cycle time, delayed call of a time OB	OB 80	22
Diagnostic Interrupt System reaction w/o OB: RUN	Wire break at diagnostics-capable module, power supply error	OB 82	Can be set: 2..26
Remove / Insert Interrupt System reaction w/o OB: RUN	Remove / Insert a module	OB 83	Can be set: 2..26
Rack Failure System reaction w/o OB: RUN	Failure of a DP-Slave or an IO-Device	OB 86	Can be set: 2..26
All events lead to an entry in the diagnostics buffer			

Asynchronous Errors

Asynchronous errors occur asynchronous (independent) to the program execution and accordingly cannot be assigned to a defined program location.

Time Errors

They occur when the current cycle time exceeds the cycle monitoring time set in the Properties of the CPU.

Diagnostic Interrupts

They are triggered by diagnostics-capable modules, such as, analog modules in case of a fault (for example, wire break).

Remove/Insert Interrupts

These interrupts are triggered when modules are removed or inserted. When a module is inserted, the operating system checks whether the correct module type was used. With this function, it is possible to remove/insert modules while the system is running.

Rack Failure

A rack failure is detected with the failure of a rack, a subnet or a station of distributed I/O.

13.7.2. S7-1200/1500: Global Error Handling with Synchronous Error OBs

Type of Error	Example	OB	Priority
S7-1200 Programming error Access error System reaction w/o OB: RUN	Access to non-existing DB Direct access to non-existing or defective I/O module	no OB exists	✗
Programming error System reaction w/o OB: STOP	Access to non-existing DB	OB 121 (only S7-1500)	
S7-1500 Access error System reaction w/o OB: RUN	Direct access to non-existing or defective I/O module	OB 122 (only S7-1500)	Can be set: 2..26

All errors lead to an entry in the diagnostic buffer

Synchronous Errors

Synchronous errors occur synchronously (dependent) to the program execution and accordingly can be assigned to a defined program location.

With a programming error, OB121 is called; with an access error, OB122. If, in case of an error, the appropriate synchronous error OB does not exist in the CPU, the CPU switches to the STOP state.

S7-1500:

You can set the priority of the synchronous error OBs from 2 to 26. The register contents that the interrupted block has used are not available in the error OB and cannot be manipulated by means of system functions.

13.7.3. OB Priorities and System Reaction

Types of event sources	Possible priorities (default priority)	Possible OB numbers	Default system reaction	Number of OBs
Startup*	1	100, ≥ 123	Ignore	0 to 100
Cyclic program*	1	1, ≥ 123	Ignore	0 to 100
Time-of-day interrupt*	2 to 24 (2)	10 to 17, ≥ 123	not applicable	0 to 20
time-delay interrupt*	2 to 24 (3)	20 to 23, ≥ 123	not applicable	0 to 20
Cyclic interrupt*	2 to 24 (8 to 17, frequency dependent)	30 to 38, ≥ 123	not applicable	0 to 20
Hardware interrupt*	2 to 26 (18)	40 to 47, ≥ 123	Ignore	0 to 50
Status interrupt	2 to 24 (4)	55	Ignore	0 or 1
Update interrupt	2 to 24 (4)	56	Ignore	0 or 1
Manufacturer-specific or profile-specific interrupt	2 to 24 (4)	57	Ignore	0 or 1
Isochronous mode interrupt	16 to 26 (21)	61 to 64, ≥ 123	Ignore	0 to 2
Time error	22	80	Ignore	0 or 1
Maximum cycle time exceeded once			STOP	
Diagnostic error interrupt	2 to 26 (5)	82	Ignore	0 or 1
Removal/insertion of modules	2 to 26 (6)	83	Ignore	0 or 1
Rack error	2 to 26 (6)	86	Ignore	0 or 1
MC servo interrupt	17 to 26 (25)	91	not applicable	0 or 1
MC interpolator interrupt	16 to 26 (24)	92	not applicable	0 or 1
Programming error (only for global error handling)	2 to 26 (7)	121	STOP	0 or 1
I/O access error (only for global error handling)	2 to 26 (7)	122	Ignore	0 or 1

Varying amounts of OBs can be created for every OB type. Numbers that are smaller than 123 are permanently assigned to certain OBs and numbers that are larger / equal to 123 are freely selectable.

Contents

14

14. Distributed I/O.....	14-2
14.1. Task Description: Operating the Conveyor Model via the ET 200SP Distributed I/O.....	14-3
14.2. Distributed I/O Systems	14-4
14.2.1. ET 200SP	14-5
14.2.1.1. ET 200SP: Configuration and Maximum Number of Modules	14-6
14.2.2. ET 200 MP	14-7
14.2.2.1. ET 200MP: Configuration and Maximum Number of Modules	14-8
14.2.3. Overview: Distributed Signal Modules	14-9
14.3. Fieldbus Systems for SIMATIC S7	14-10
14.3.1. Identification of Distributed I/O Devices	14-11
14.3.2. Components of the PROFINET Standard.....	14-12
14.3.2.1. PROFINET IO Device Types	14-13
14.3.2.2. PROFINET Addresses	14-14
14.3.2.3. PROFINET Communication Model.....	14-15
14.4. Inserting and Networking Distributed I/O	14-16
14.4.1. PROFINET IO Device ET 200SP: Assigning the IP Address and Device Name OFFLINE	14-17
14.4.2. PROFINET IO Device ET 200SP: Assigning the Device Name ONLINE	14-18
14.5. Grouping Devices	14-19
14.6. Task Description: Commissioning the ET 200SP	14-20
14.6.1. Exercise 1: ET 200SP: Reset to Factory Settings	14-21
14.6.2. Exercise 2: Reading-out the Firmware Version of the ET 200SP.....	14-22
14.6.3. Exercise 3: Offline Project: Adding the ET 200SP	14-23
14.6.4. Exercise 4: Networking the ET 200SP	14-24
14.6.5. Exercise 5: Configuring and Parameterizing the ET 200SP	14-25
14.6.6. Exercise 6: Setting the Channel Parameters of the Analog Modules.....	14-26
14.6.7. Exercise 7: ET 200SP: Assigning the IP Address / PROFINET Name <u>OFFLINE</u>	14-27
14.6.8. Exercise 8: ET 200SP: Assigning the PROFINET Name <u>ONLINE</u>	14-28
14.6.9. Exercise 9: Creating a New Device Group and Grouping Devices	14-29
14.6.10. Exercise 10: Compiling the Changes and Downloading them into the Device.....	14-30
14.6.11. Exercise 11: Adjusting the S7 Program via "Rewiring"	14-31
14.6.12. Exercise 12: Function Test with Conveyor Model via Distributed I/O.....	14-32
14.7. Additional Information	14-33
14.7.1. Installing Distributed Peripheral Components Later On via GSD	14-34

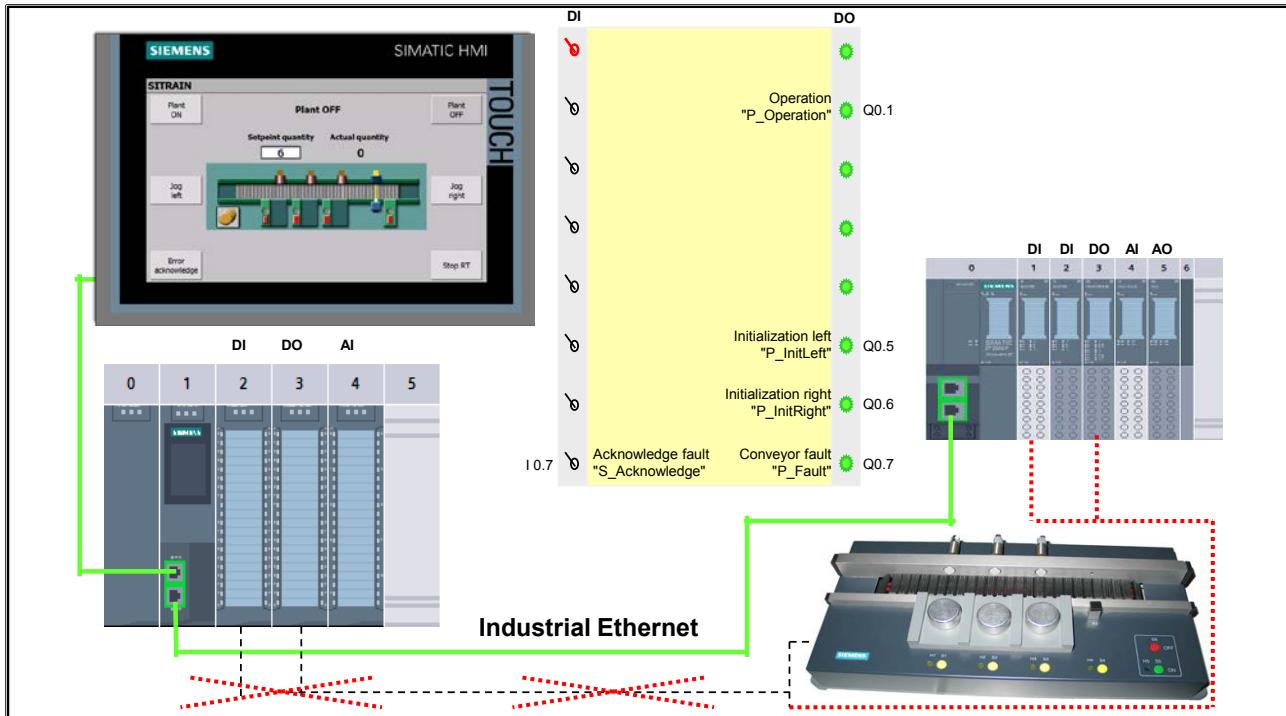
14. Distributed I/O

At the end of the chapter the participant will ...



- ... be familiar with the various distributed I/O systems
- ... be familiar with the PROFINET and PROFIBUS bus systems
- ... be able to explain the functional principle of PROFINET
- ... be able to configure, network and commission a distributed PROFINET I/O station

14.1. Task Description: Operating the Conveyor Model via the ET 200SP Distributed I/O



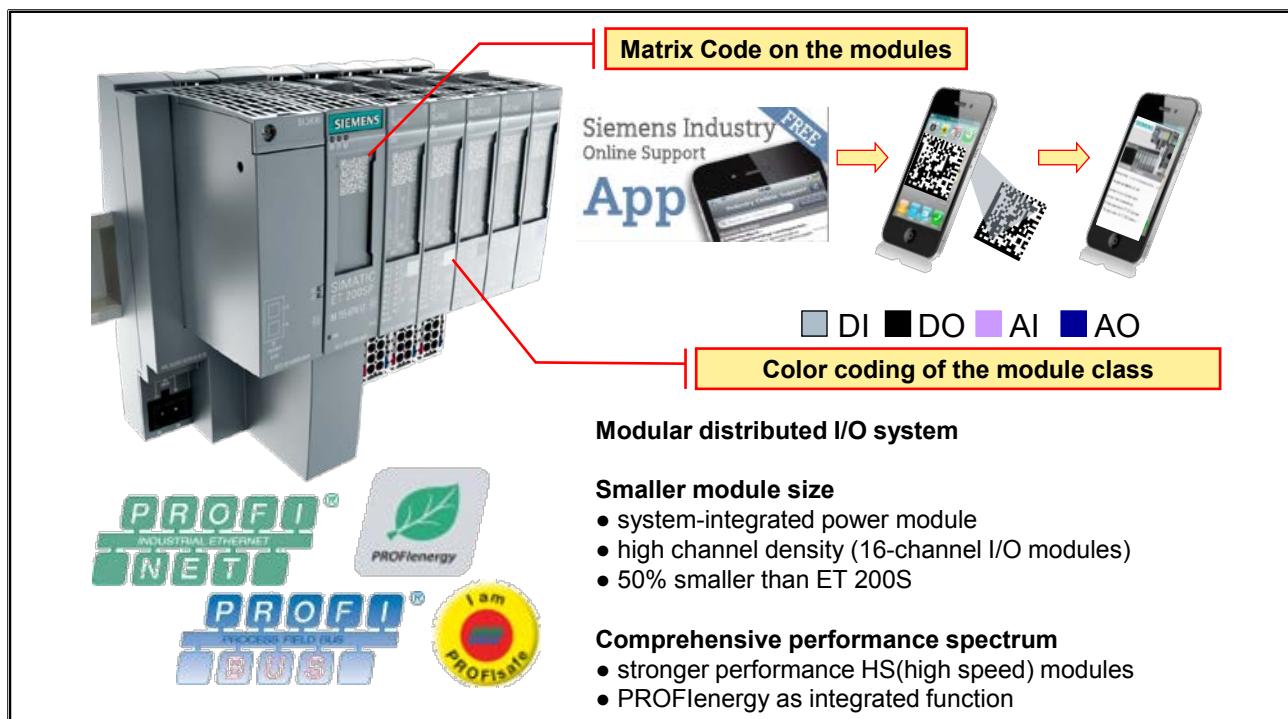
Task Description

- The conveyor model is no longer to be controlled via the DI/DO modules of the S7-1500 central device, but is to be controlled via the DI and DO modules of the distributed ET 200SP station.
- For this, the ET 200SP must be networked with the central S7-1500 station via PROFINET and the S7 program has to be adjusted.

14.2. Distributed I/O Systems

	SIMATIC ET 200M High functionality, S7-300 modules ...PROFIBUS or PROFINET (up to 12 modules per station)		SIMATIC ET 200MP I/O modules in the S7-1500 design ...the multi-channel and multi-functional I/O of the S7-1500
	SIMATIC ET 200S Discretely modular and multi-functional Hot Swapping and fixed wiring Installation up into Ex-Zone 2		SIMATIC ET 200SP Compact I/O modules for variable design ...the scalable I/O with largest portfolio and smallest space requirement
	SIMATIC ET 200pro Modular and multi-functional: Power modules, Motor starters, RFID modules, Fail-safe modules, CPU, Frequency inverters		SIMATIC ET 200iSP - Intrinsically safe (Ex i) and modular - Fail-safe modules (SIL3/Cat.4/PLe) - PROFIBUS
	SIMATIC ET 200pro CPU ... the power of the S7-1500 in the compact design of the ET 200pro		SIMATIC ET 200SP CPU ...the power of the S7-1500 in the compact design of the ET 200SP
	SIMATIC ET 200AL I/O modules for IP65/67 ...the robust I/O for the easiest assembly anywhere		SIMATIC ET 200eco and ET 200eco PN Compact block I/O for IP65/66/67 ... for demanding applications

14.2.1. ET 200SP

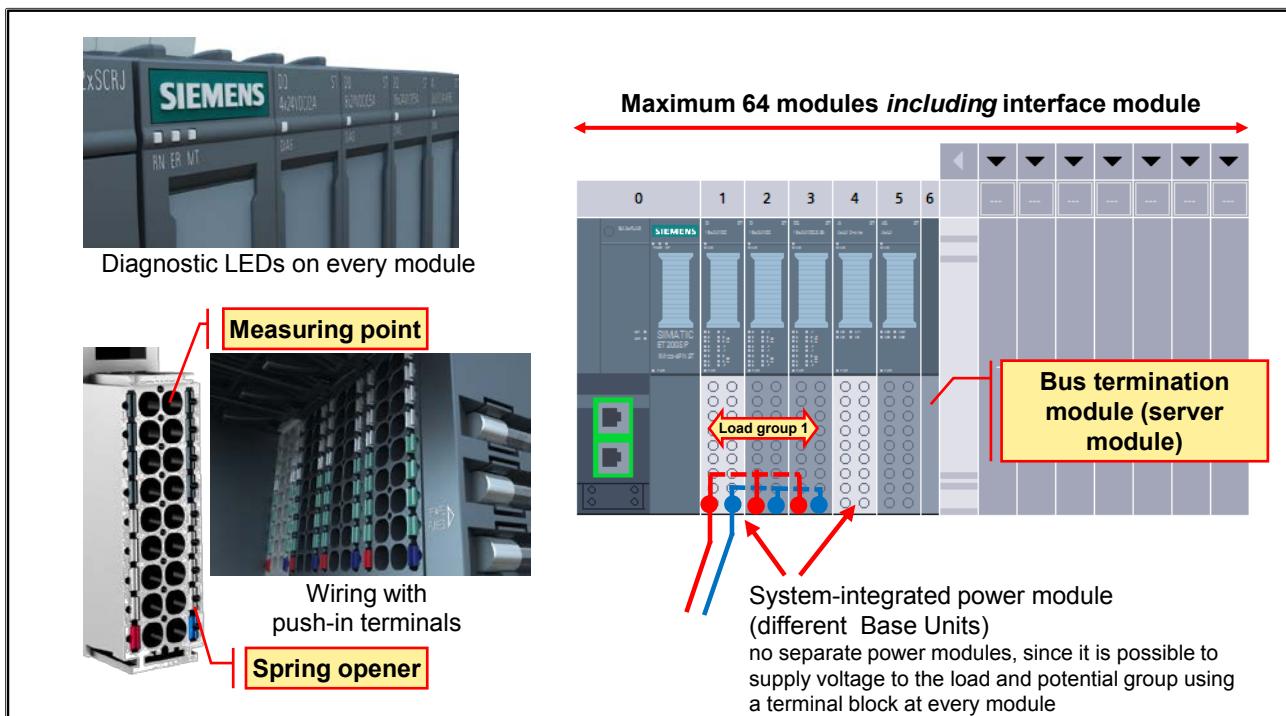


2d Matrix Code (Data Matrix)

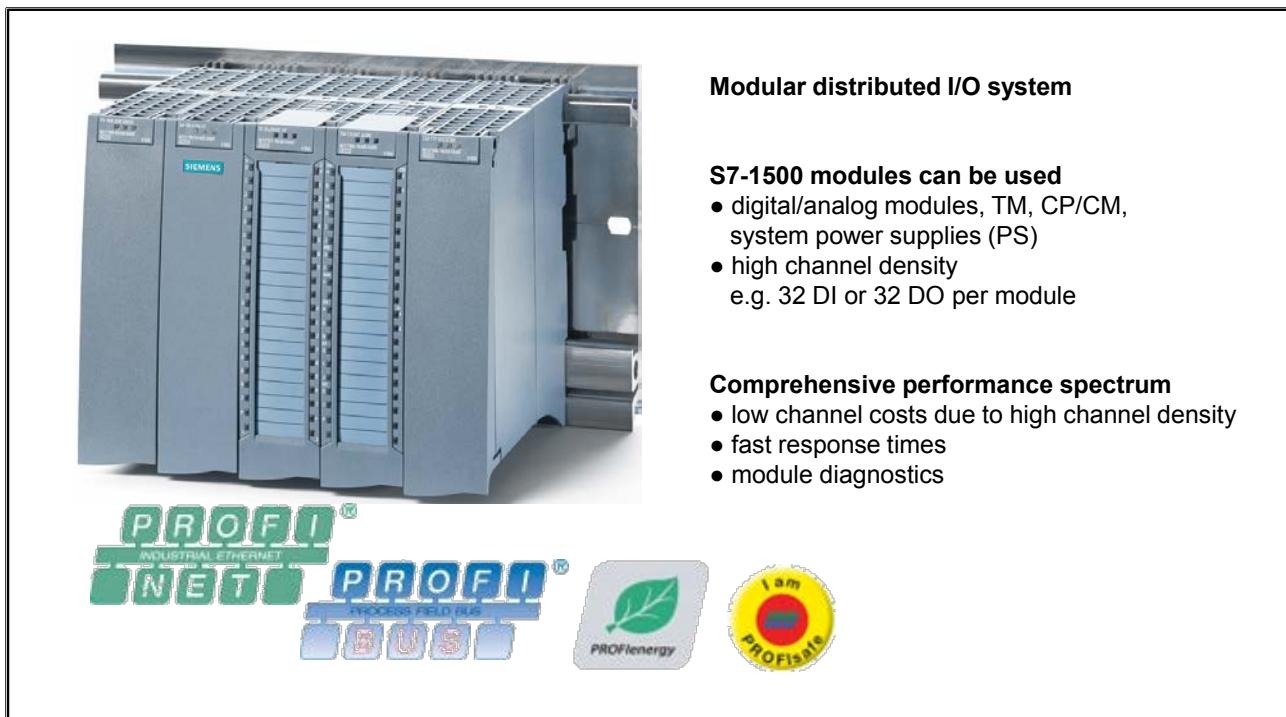
This code is used for identifying modules and can be photographed or decoded by Smart Phones, PDAs, and the like.

With the ET 200SP modules, the code contains an Internet link to the product page of the associated module.

14.2.1.1. ET 200SP: Configuration and Maximum Number of Modules



14.2.2. ET 200 MP

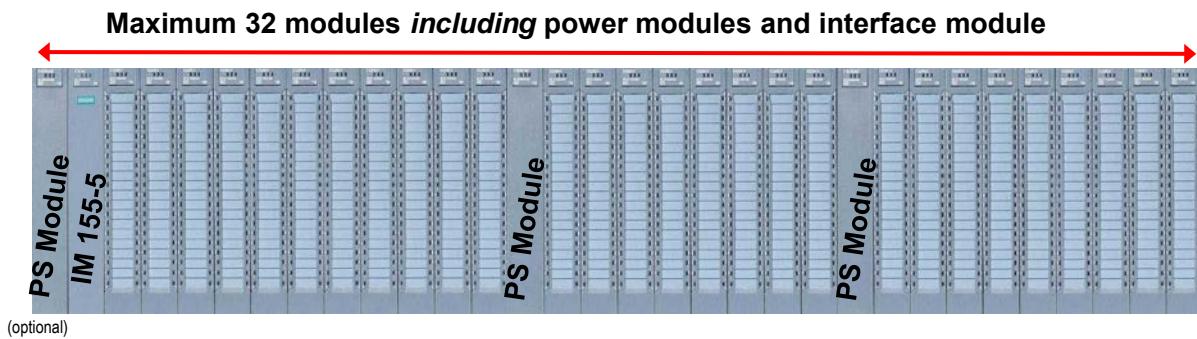


The ET 200MP enables the distributed connection of S7-1500 series central I/O modules.

Connection is made using an interface module.

14.2.2.1. ET 200MP: Configuration and Maximum Number of Modules

- Maximum of 32 modules
 - 1st. module = system power supply (PS)
 - 2nd. module = interface modules
 - 3rd.-32nd. module = max. of any 30 I/O modules of the S7-1500
- Formation of load groups through additional voltage supplies
(similar to centralized configuration of S7-1500)



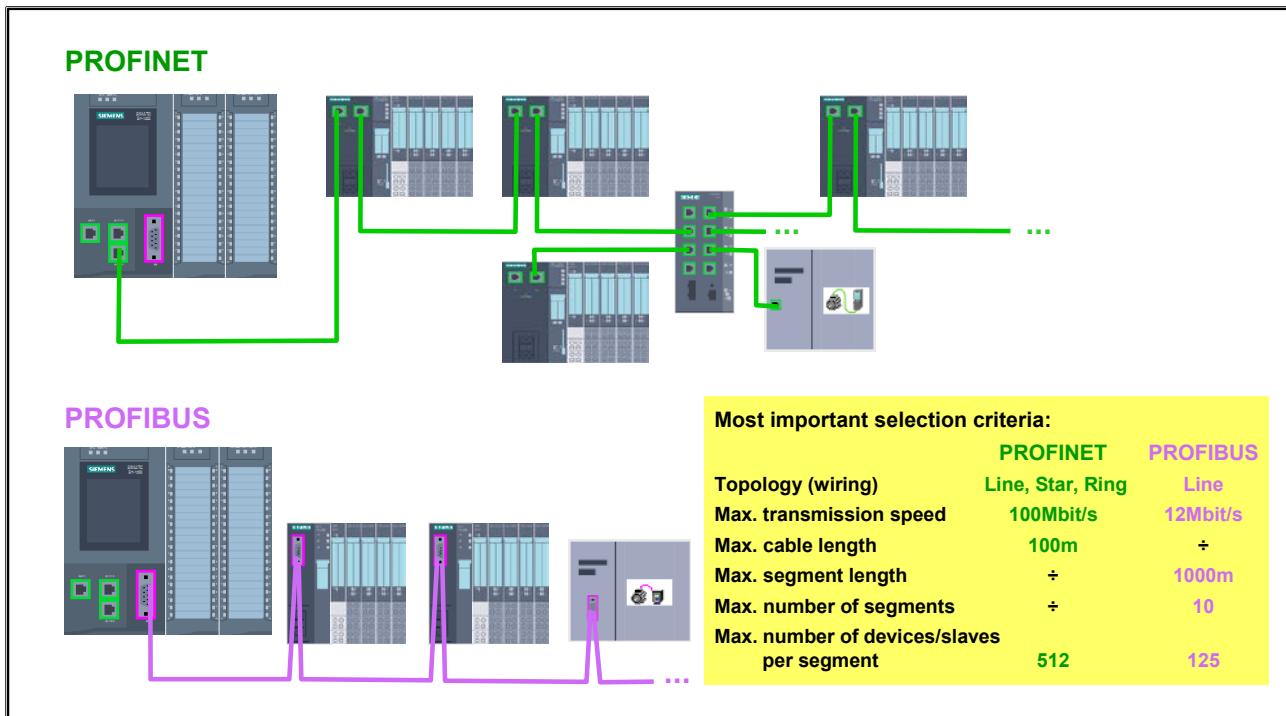
14.2.3. Overview: Distributed Signal Modules

	ET 200SP	ET 200MP	ET 200S	ET 200M	ET 200pro
DI/DQ	✓	✓	✓	✓	✓
AI/AQ	✓	✓	✓	✓	✓
F-DI/F-DQ	✓	✓	✓	✓	✓
F-AI	✗		✗	✓	✗

The ET 200SP and ET 200MP product range will be expanded in the next years such that these two product lines completely cover the applications of the ET 200S and ET 200M.

The ET 200pro (interface modules for connection to PROFINET or PROFIBUS) in the degree of protection IP65/67 for use directly at the machine will continue to be offered.

14.3. Fieldbus Systems for SIMATIC S7



Fieldbus Systems for SIMATIC S7

To connect distributed I/Os, there are different bus systems.
The most important for SIMATIC S7 are:

- **PROFINET**
As the standard for communication applications at the field level it enables the connection of distributed field devices via Industrial Ethernet.
The Industrial Ethernet network is a local area network (LAN) according to the international Standard IEEE 802.3 (Ethernet) and is designed for the industrial sector. It enables open and comprehensive network solutions with a high transmission performance.
- **PROFIBUS**
It is the bus system for local area networks (LANs) with only a few participants. Through its fulfillment of requirements according to EN 50170, PROFIBUS ensures openness for the connection of standard-conforming components of all manufacturers.

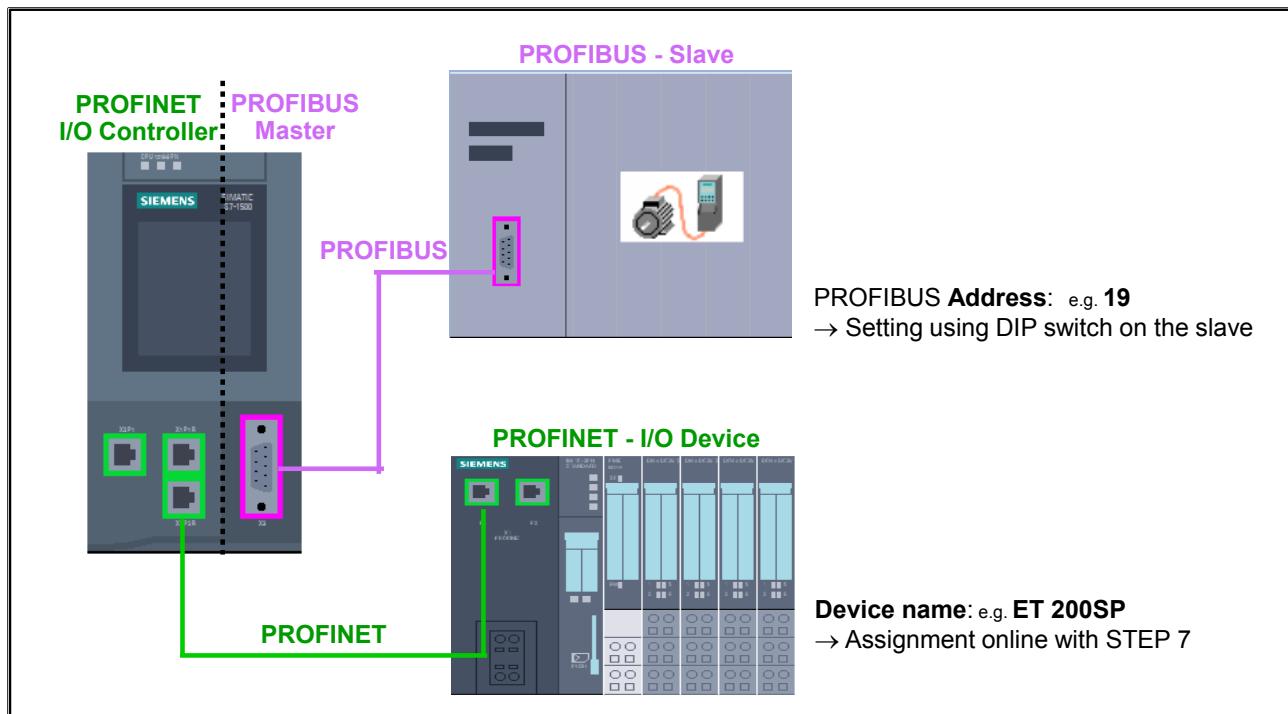
Due to the physical and communication-related differences of the two bus systems, there are various criteria which are used for the selection of the most suitable bus system.

Cable Length, Segment Length

For PROFIBUS, a module line has to be reinforced after 100-1000m (depending on the transmission speed used); otherwise, the maximum bus length is reached.

For PROFINET, every connected component takes over this function. For that reason, only the cable length between two modules is relevant here.

14.3.1. Identification of Distributed I/O Devices



Distributed I/O Devices

During start-up, the CPU searches the configured PNIO devices or DP slaves and parameterizes these according to the loaded device configuration.

Both fieldbus systems use different methods for identifying I/O modules:

- **PROFIBUS**
The set PROFIBUS address is used to search for the configured DP slave.
The setting is typically made through the DIP switch on the slave.
- **PROFINET**
The assigned device name is used to search for the configured PNIO device.
The assignment of the device name (device initialization) is done from the STEP 7 engineering through an online function.
The parameterized IP address is then assigned to the PNIO device by the PNIO controller (CPU).

14.3.2. Components of the PROFINET Standard

PROFINET IO	Integration of distributed field devices via Industrial Ethernet
PROFIdrive	Applications profile for drives connected to PROFIBUS and PROFINET
PROFIsafe	Integration of fail-safe technology (fail-safe controllers / communication) in the PROFINET standard
PROFIenergy	Coordinated and centrally controlled switch-off of power consuming devices during break times

PROFINET

It completely covers the requirements of automation. PROFINET brings together the expertise of PROFIBUS and Industrial Ethernet. The utilization of the open standard, the easy handling and the integration of existing parts of a system (e.g. a plant) determined the definition of PROFINET right from the beginning.

PROFINET IO

With PROFINET IO, the integration of distributed field devices takes place directly on the Ethernet. For that, the Master-Slave procedure from PROFIBUS DP is carried over into a Provider-Consumer model. From the communication point of view, all devices on the Ethernet have equal rights. Through the configuration, however, the field devices are assigned to a central controller. The distributed I/O device reads-in the I/O signals and transfers them to the controller. The controller processes them and transfers the outputs back to the distributed I/O device.

PROFIdrive

With PROFIdrive, very fast, clock-synchronous drive controls for high performance Motion Control applications are implemented.

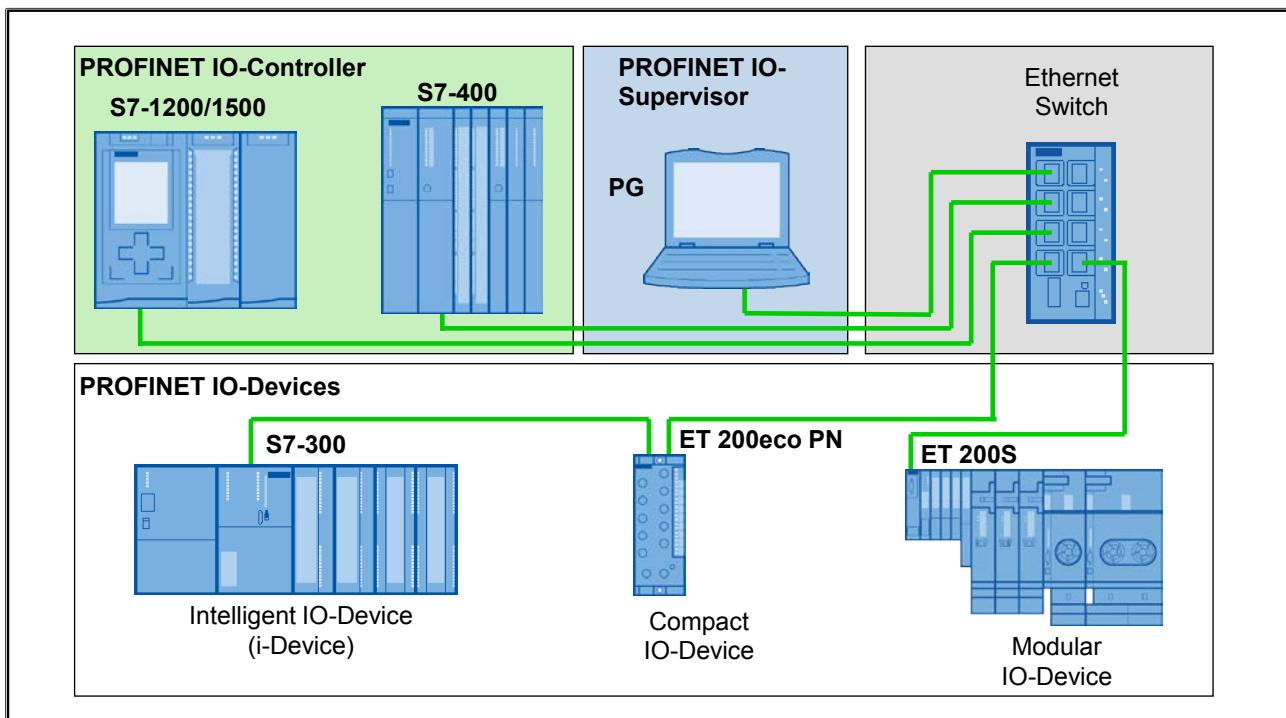
PROFIsafe

With PROFIsafe, the network infrastructure already existing for standard communication can also be used at the same time for fail-safe communication. The existing bus protocols, such as, PROFIBUS and PROFINET (so-called "black channel") are used to transport fail-safe data as additional data (so-called PROFIsafe layer).

PROFIenergy

PROFIenergy permits a coordinated and centrally controlled switch off of power consuming devices during break times. In this way, the process uses only the absolute necessary energy. The process itself saves the majority of the energy, the PROFINET device itself only has a savings potential of several watts. For PROFIenergy, this operating state is called "Pause".

14.3.2.1. PROFINET IO Device Types



PROFINET IO-Controller

The IO-Controller (typically the PLC) establishes a logical connection to the connected IO-Devices after Power-On and subsequently parameterizes these (module parameters, address, etc.). (This corresponds to the function of a Class 1 Master in PROFIBUS).

PROFINET IO-Device

An IO-Device is a distributed IO device that is connected via PROFINET IO (this corresponds to the function of a slave in PROFIBUS).

Differentiation is made for the following IO-Device types:

- Compact IO-Device: Fixed degree of expansion.
- Modular IO-Device: Variable degree of expansion; can be expanded or reduced as required.
- Intelligent IO-Device: A PLC is configured not as an IO-Controller but as an IO-Device and provides a higher-level controller with I/O data.

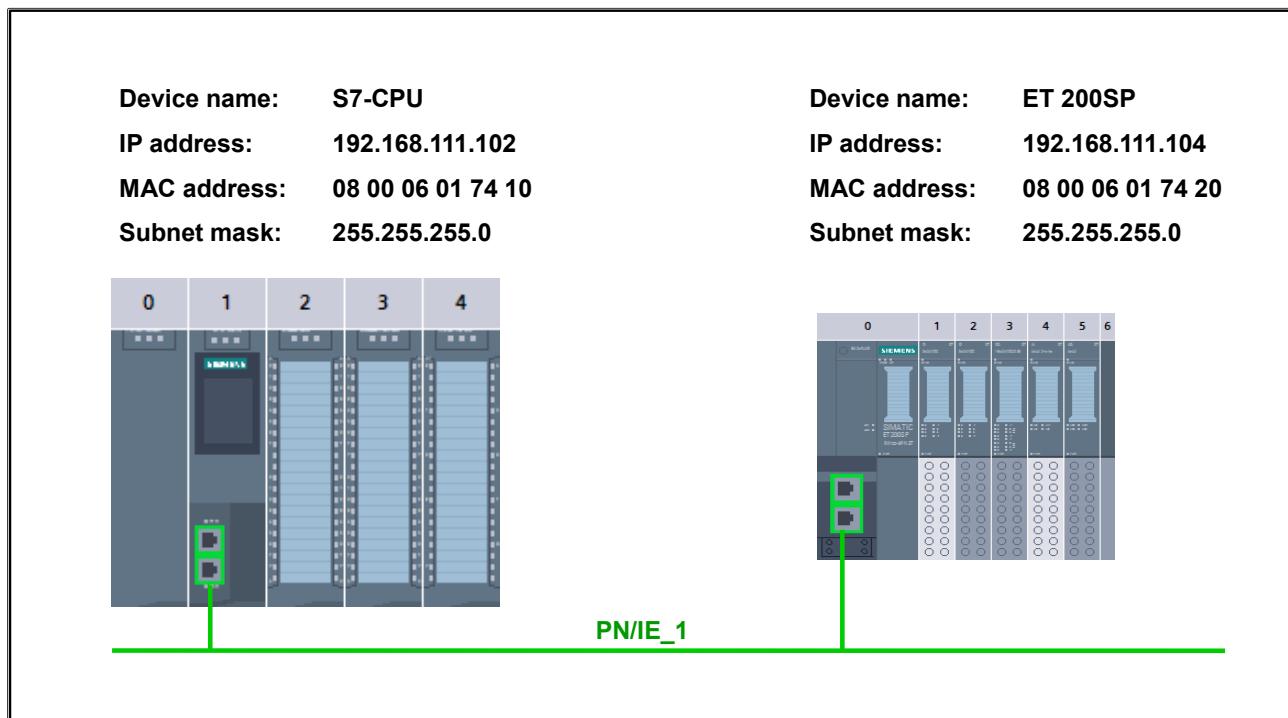
IO-Supervisor

This can be a programming device (PG), personal computer (PC) or Human Machine Interface (HMI) for commissioning or diagnostic purposes. (This corresponds to a Class 2 Master in PROFIBUS).

Ethernet Switch

PROFINET is based on Ethernet. For that reason, switches are always used as "network distributors". Every node is connected to a switch via a so-called "point-to-point" connection. This is also referred to as a **"Switched Ethernet"**. In most PROFINET devices, a 2 or multi-port switch is already integrated so that it is very easy to establish a line structure (comparable to PROFIBUS).

14.3.2.2. PROFINET Addresses



Internet Protocol

The **Internet Protocol (IP)** is the basis for all TCP/IP networks. It creates the so-called datagrams (data packets specially tailored to the Internet protocol) and handles their transport within the local subnet or their "routing" (forwarding) to other subnets.

IP Addresses

IP addresses consist of 4 bytes. With the dot notation, each byte of the IP address is expressed by a decimal number between 0 and 255. The four decimal numbers are separated from one another through dots (see picture).

PROFINET Device Name

In PROFINET, each RT / IRT device must be assigned a unique device name that is retentively stored in the device. A module exchange without PG/PC is made possible through the device names.

MAC Address

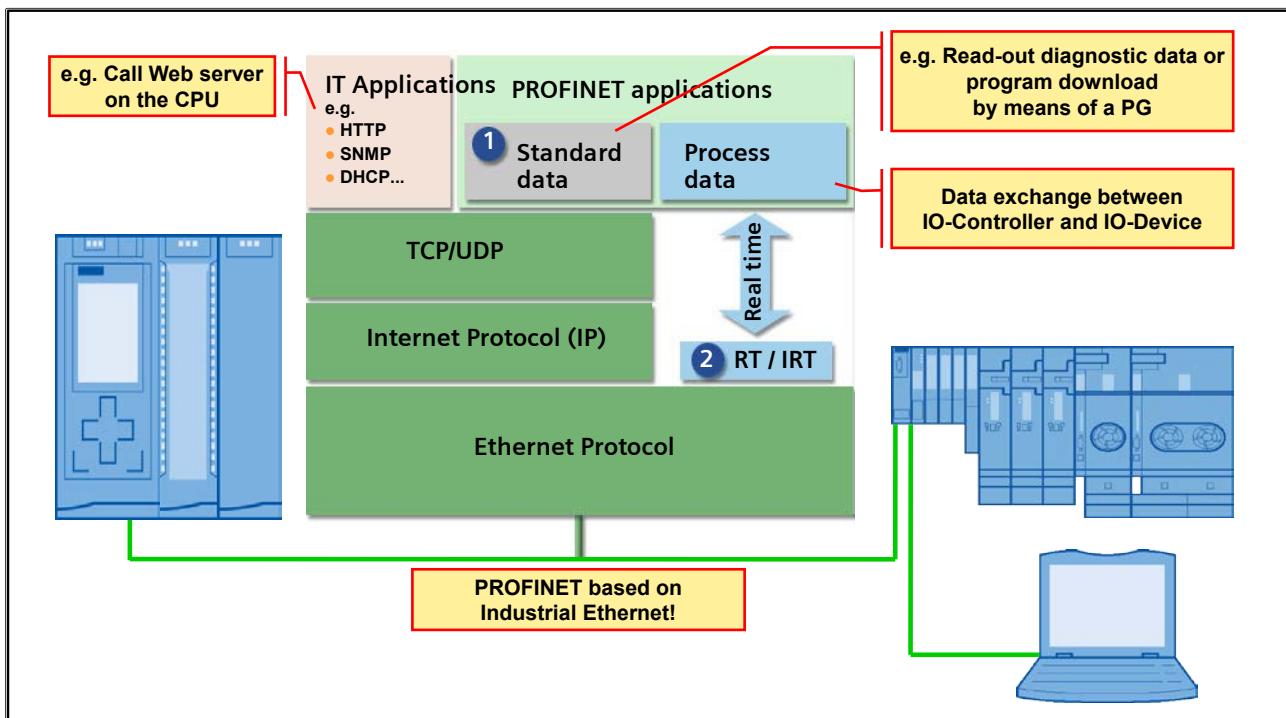
Every Ethernet interface is assigned a fixed address by the manufacturer that is unique worldwide. This address is referred to as the hardware or MAC address (**Media Access Control**). It is stored on the network card and uniquely identifies the Ethernet interface in a local network. Cooperation among the manufacturers ensures that the address is unique worldwide.

Subnet Mask

The subnet mask specifies which IP addresses in the local network can be accessed. It separates the IP address into the network and device part.

Only IP addresses whose network part is the same can be accessed.
e.g.: Subnet mask = **255.255.255.0** and IP address = **192.168.111.10**
accessible IP addresses: **192.168.111.1** to **192.168.111.254**

14.3.2.3. PROFINET Communication Model



Real-time Channel

To be able to fulfill real-time requirements in automation, an optimized real-time communication channel, the Realtime Channel (RT Channel), was specified in PROFINET. It uses Ethernet (Layer 2) as a base.

The addressing of the data packets does not take place in this case via an IP address, rather by means of the MAC addresses of the participating devices. Such a solution minimizes the throughput times in the communications stack considerably and leads to an increase in performance with regards to the updating rate of automation data.

IRT Channel

Isochronous Real-time (IRT) as a further development with the following features:

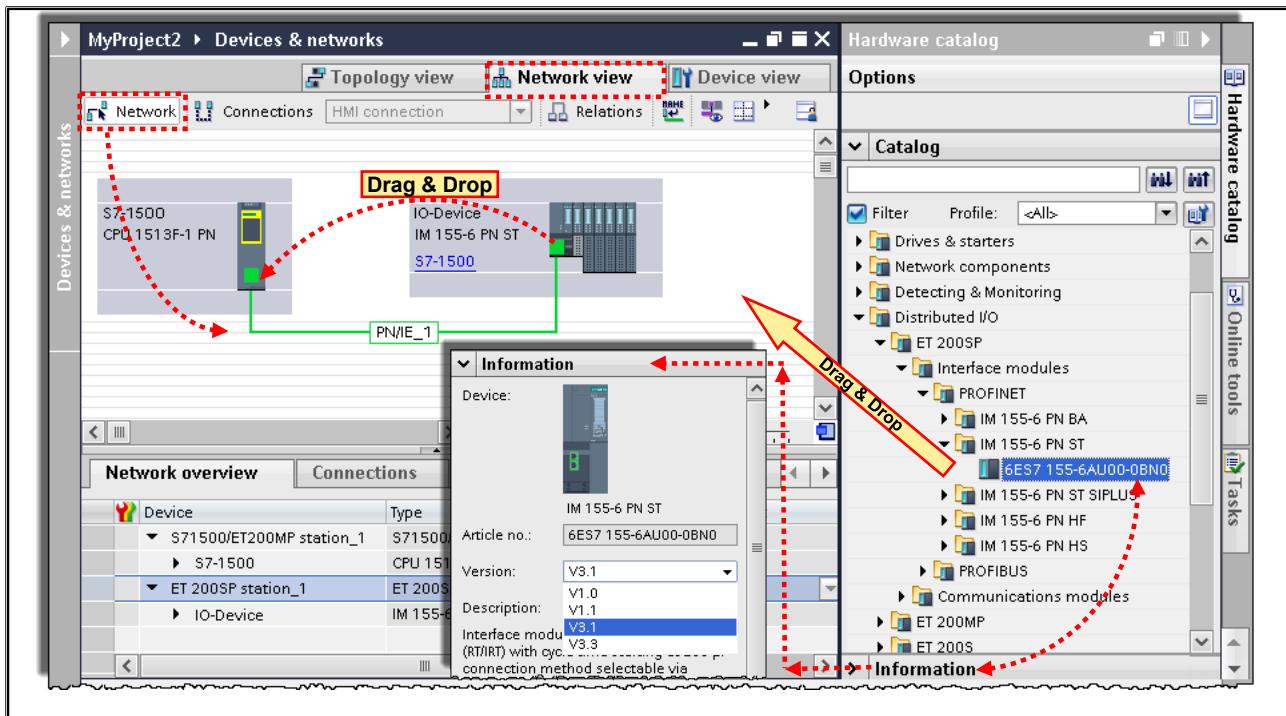
- Clock-synchronous data transmission
- Cycle times <1ms with jitter accuracy <1µs
- Typical field of application is Motion Control

IT Standards

The design of PROFINET WEB Integration focuses on commissioning and diagnostics. Access to a PROFINET device from the Internet or Intranet is done with standard protocols (for example, http). The data is transmitted in standard formats such as HTML or XML and can be presented with standard browsers such as Opera or Internet Explorer.

This worldwide accessibility makes it easy for the application manufacturer to support the user with commissioning, device diagnostics etc. Access to the data is done via Web servers which are integrated in the modules.

14.4. Inserting and Networking Distributed I/O



Inserting Distributed I/O

PROFINET IO-Devices are added in the Network view. Here, you can insert the relevant devices into the project by dragging & dropping them from the Hardware catalog. The correct Firmware must also be selected here before the device is inserted.

The new device (ET 200SP) is stored in the "Ungrouped devices" folder.

Furthermore, there is a link to the ET 200SP in the "Unassigned devices" folder since it is not yet assigned to a controller.

Networking Distributed I/O

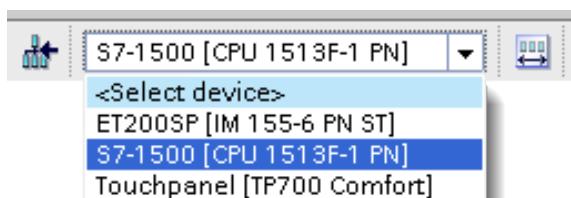
After the ET 200SP IO-Device is added, it must be assigned to an IO-Controller or networked with a CPU. In case there are several CPUs in the network, a co-ordination or monitoring of the I/O addresses by the IO-Controller and IO-Device can only be done through this unique assignment.

If the ET 200SP is assigned to a Controller, the link is stored in the "Distributed I/O" subfolder of the Controller.

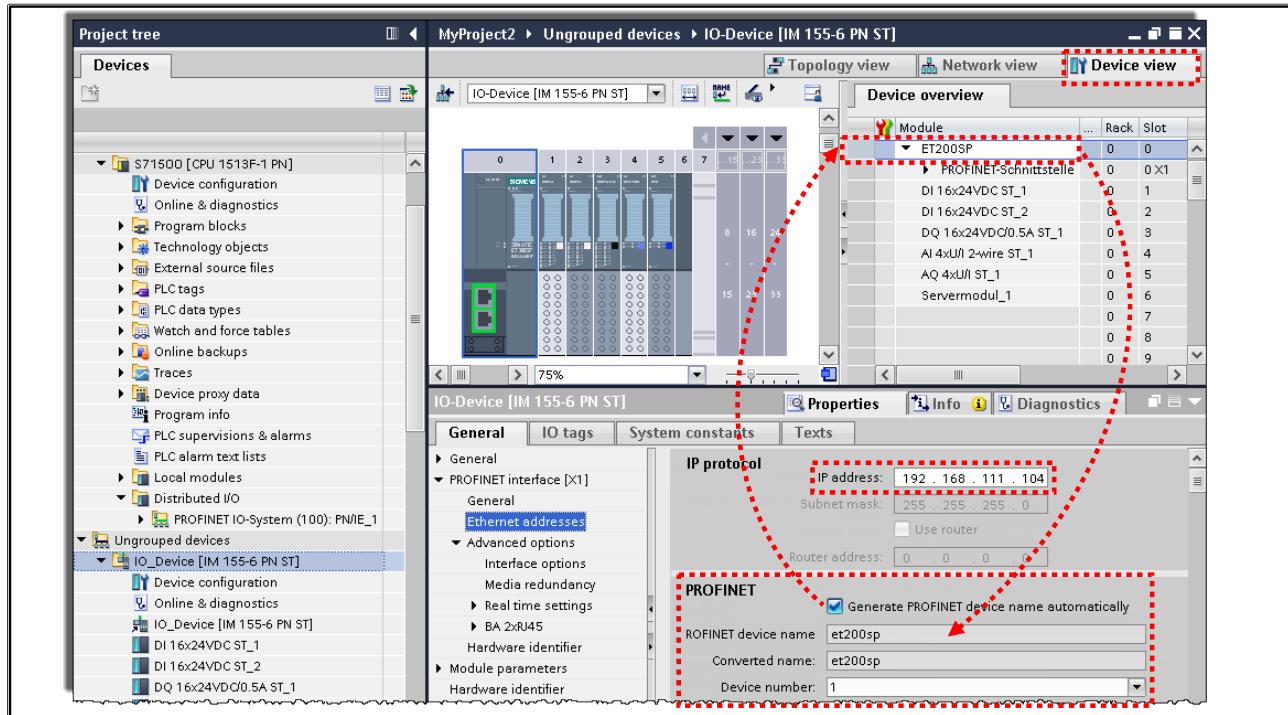
The actual device is still stored in the "Unassigned devices" folder.

I/O Modules

Just as for a CPU, the individual input and output modules can be configured and assigned parameters in the Device view. For this, the device is selected and the Device view opened or in the Device view, the relevant device is selected via the selection menu.



14.4.1. PROFINET IO Device ET 200SP: Assigning the IP Address and Device Name OFFLINE



IP Address, Subnet Mask and PROFINET Device Name

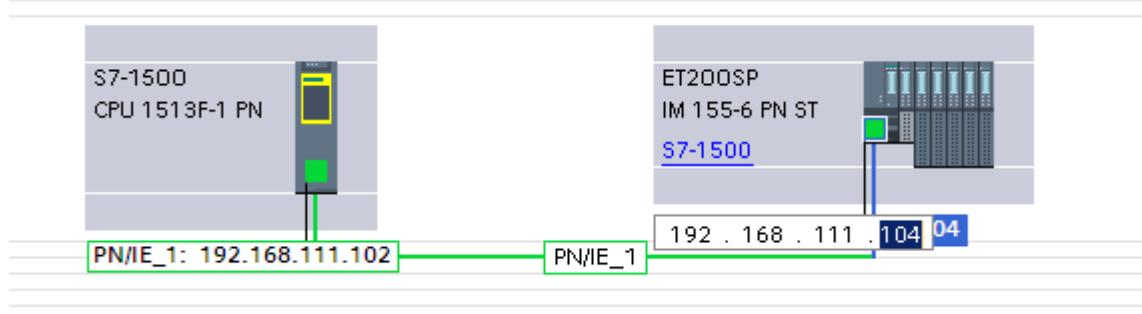
For communication with the IO-Controller, a PROFINET device name must be assigned to the IO-Device (ET 200SP) OFFLINE. The IO-Controller then assigns the IO-Device a valid IP address. If the IO-Device is assigned an IP address OFFLINE, this IP address is adopted. These parameters are downloaded to the IO-Controller (CPU) with the programming device. The IO-Controller (CPU) then transfers these and other parameters (such as, the I/O addresses) to the IO-Device (ET 200SP).

Attention:

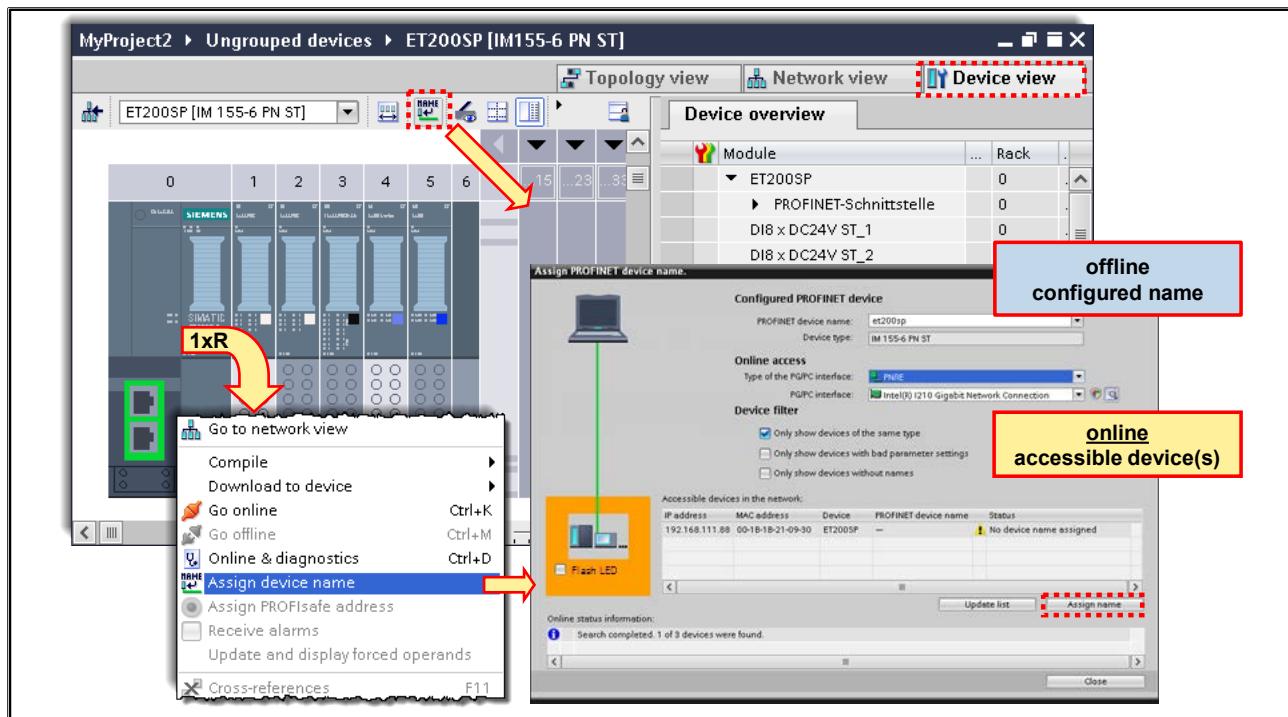
Only the PROFINET device name is relevant for the transmission of the offline configuration into the online device (Controller), not the IP address. The offline configured PROFINET device name and the online existing PROFINET device name must match. If the IO-Device has a different PROFINET device name or doesn't have a name at all, the IO-Controller cannot transfer the hardware configuration or the hardware parameter assignments to the IO-Device thus preventing a PROFINET system startup.

Note:

The IP addresses can also be entered directly in the Network view in the graphic area. For this, the IP addresses must be displayed with the button.



14.4.2. PROFINET IO Device ET 200SP: Assigning the Device Name ONLINE



IP Address and PROFINET Device Name

The PROFINET device name of the IO-Device configured offline and the device name existing online must match since the IO-Controller first checks the device names of the connected IO-Devices and then assigns the configured IP addresses during system startup. If an IO-Device is not accessible under its configured device name, the IO-Controller cannot establish a connection to the IO-Device.

The IP address of the IO-Device configured offline and the address existing online do not have to match. The PROFINET name is relevant for the downloading of the hardware configuration. If it exists, the online existing IP address is overwritten with the offline configured IP address.

Ways of Assigning a Name Online

In principle, there are two ways of assigning a PROFINET device name to an IO-Device online:

Version 1 (safe, since there is no chance of typing errors)

The assignment of the device name is triggered from the device configuration of the IO-Device.

Device configuration of IO-Device → Right-click on the Interface module (Slot 0) → Assign device name (see picture).

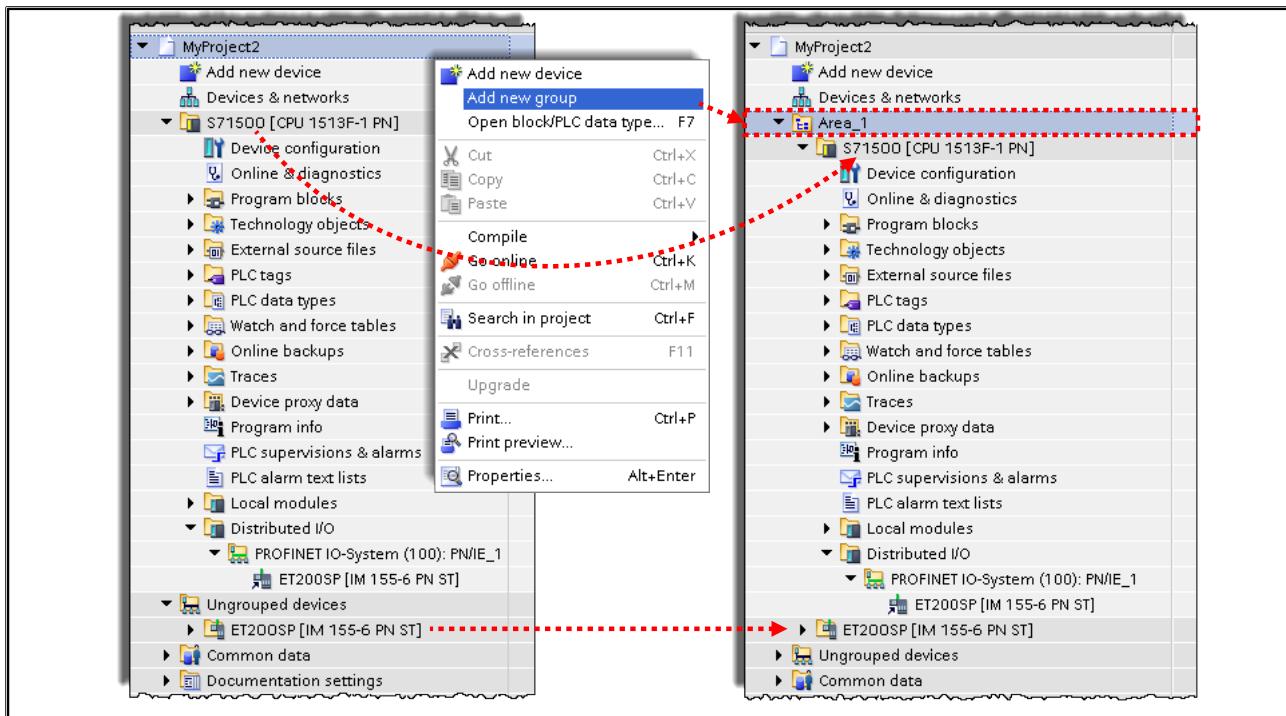
The advantage of this version is that the offline configured device name is adopted 1:1 and so no typing errors can be made.

Version 2 (typing errors possible)

The assignment of the device name is triggered via "Online access":

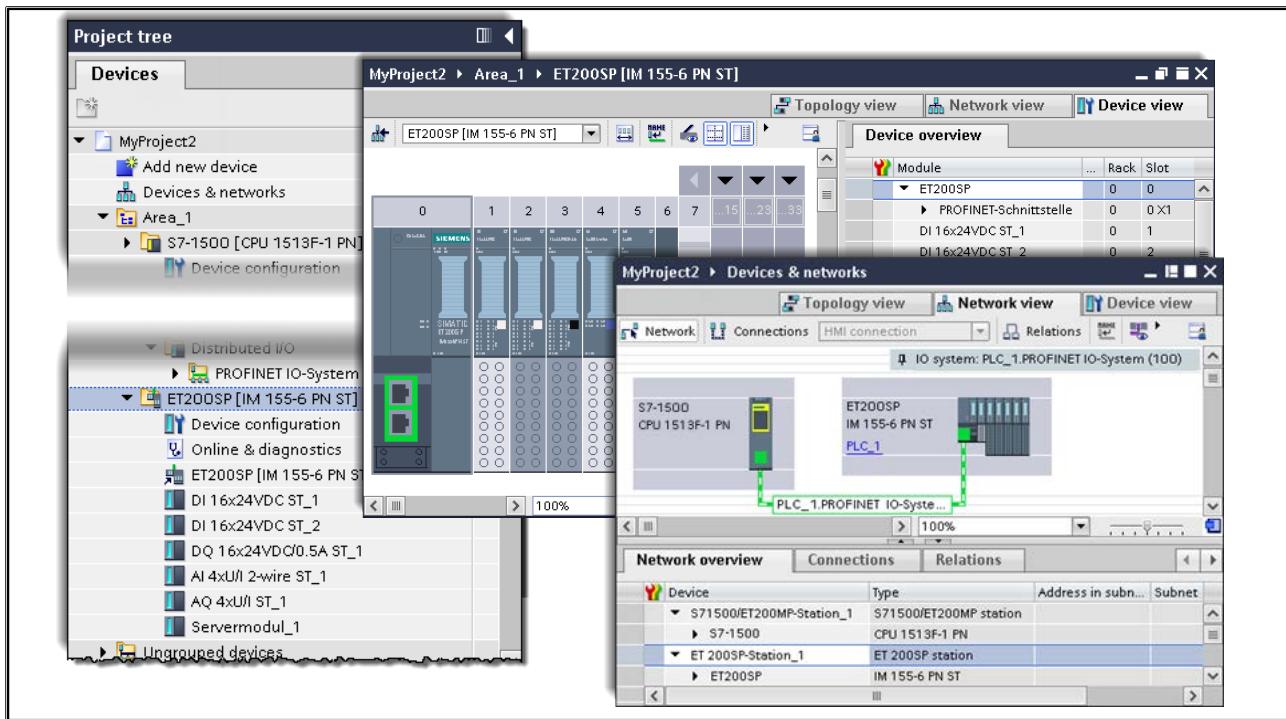
Project tree → Online access → Ethernet interface → IO-Device → Online & diagnostics → Functions → Assign name

14.5. Grouping Devices



The individual devices (distributed I/O as well) can all be stored directly in the project. For better readability of the project, it is recommended to group the individual devices. For this, device groups can be created in which the individual devices can be stored using drag & drop.

14.6. Task Description: Commissioning the ET 200SP



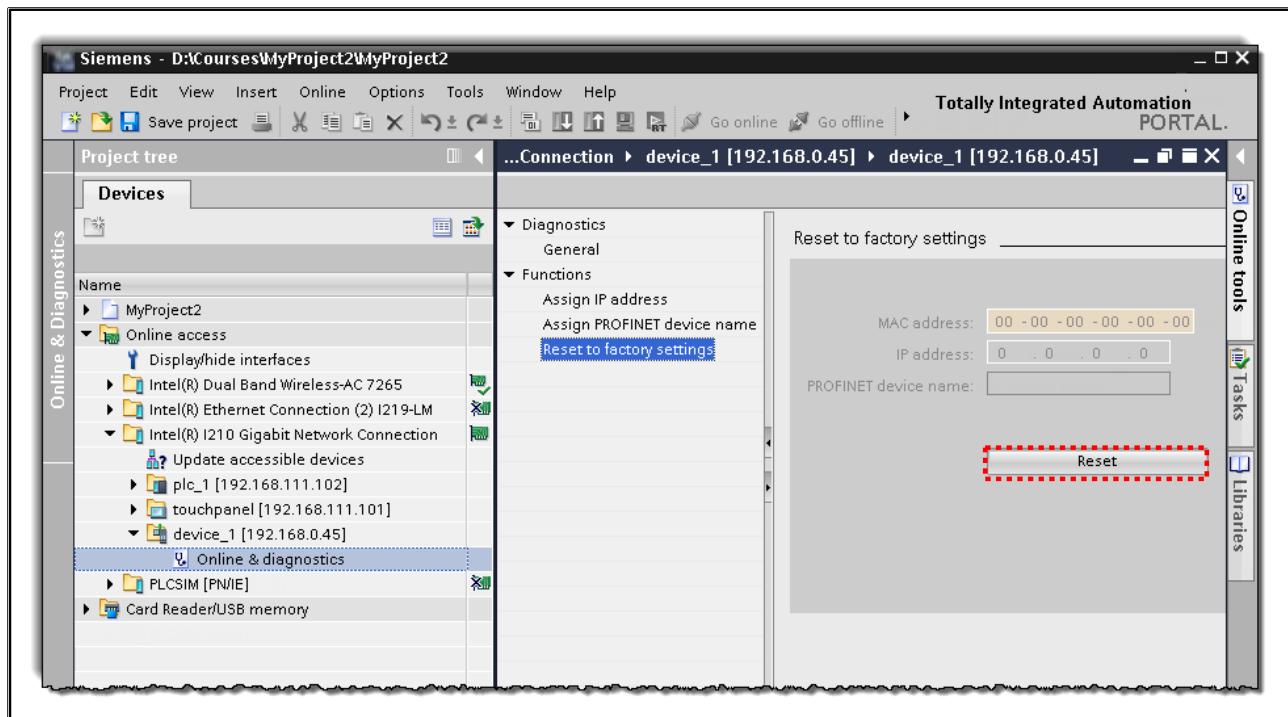
Task Description

The ET 200SP distributed I/O station is to be commissioned since the conveyor model is later to be controlled via its input and output modules.

For this, the ET 200SP station must be configured, assigned parameters and networked with the S7-1500 station in the offline project.

After compiling the new hardware configuration, it must be downloaded into the CPU. In the function of an I/O Controller, the CPU then automatically undertakes the parameterization of the ET 200SP I/O-device.

14.6.1. Exercise 1: ET 200SP: Reset to Factory Settings



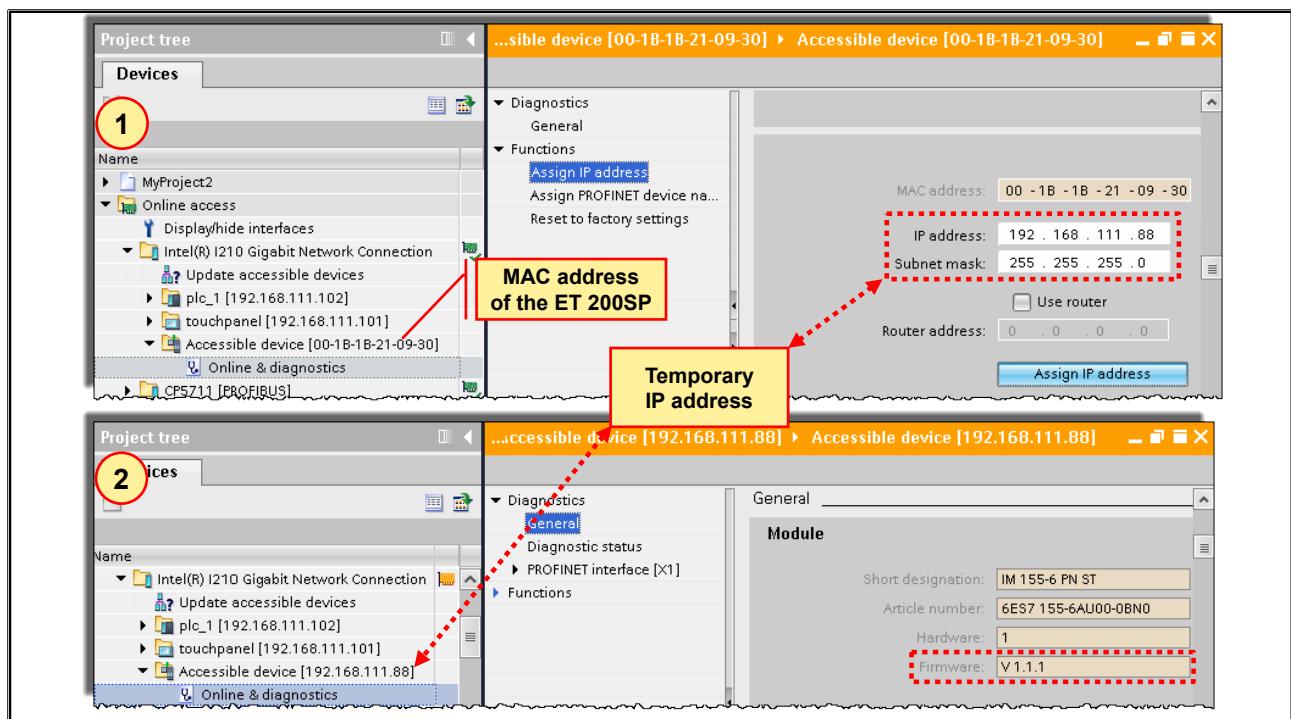
Task

All settings so far (IP address, subnet mask and PROFINET name) of the Interface module and the memory card of the ET 200SP station are to be deleted through a "Reset to factory settings". In the following exercises, you will then transfer your own settings onto the ET 200SP station.

What to Do:

1. Open the Online access and there select the interface that is connected to your training case.
2. Activate it by double-clicking on "Update accessible devices" and wait until the list is completed. This is indicated by a green checkmark by the interface.
3. Open the ET 200SP and there activate the function "Online & diagnostics" by double-clicking on it.
4. In the "Online & diagnostics" window, open the menu "Functions > Reset to factory settings".
5. Start the "Reset" function and confirm the follow-up dialog with "Yes".
6. Close the "Online & diagnostics" window.
7. Check the success of the reset in the Inspector window under "INFO > General". In addition, after updating the accessible devices, you will find the ET 200SP in the list without an IP address and without a device name.
8. Leave all windows open for the next exercise.

14.6.2. Exercise 2: Reading-out the Firmware Version of the ET 200SP



Task

In the following exercises, in order to be able to configure an ET 200SP in the offline project which corresponds exactly to that of the training device, you now have to read out the Firmware version of the ET 200SP online.

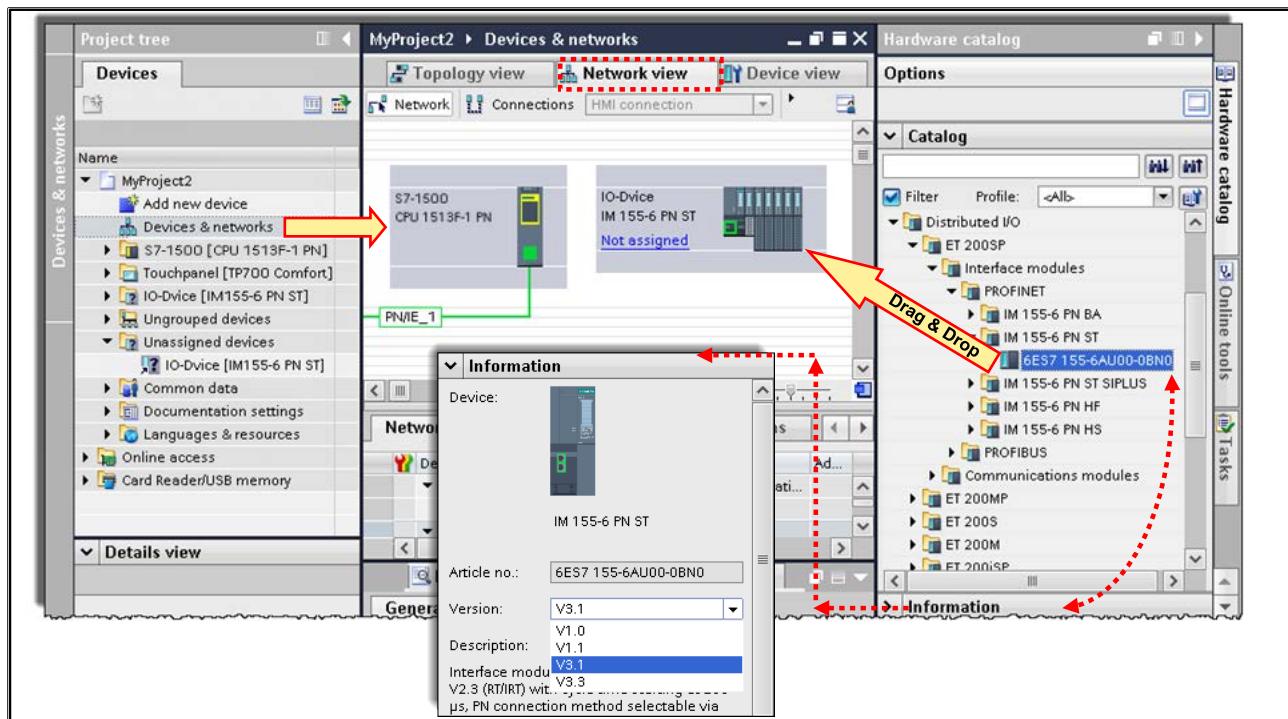
Problem

Due to the previous "Reset to factory settings", the ET 200SP now no longer has a PROFINET device name nor an IP address, only a MAC address (see top picture). The Firmware version, however, cannot be read out via the MAC address, since an IP address is required for this diagnostic service.

What to Do

- 1**
 - 1. Open the ET 200SP and, with a double-click, activate the "Online & diagnostics" function and there check whether the ET 200SP Firmware version is displayed in the menu "Diagnostics -> General".
 - 2. No Firmware version is displayed since the ET 200SP doesn't have an IP address. To assign a temporary IP address, switch to the "Functions -> Assign IP address" menu. There enter the temporary IP address as well as the subnet mask shown in the picture and adopt the entry by clicking "Assign IP address" (see top picture)
 - 3. In the Project tree, once again "Update" the list of "accessible devices".
 - 4. In the device list, the ET 200SP is now displayed as a device with Order number and IP address. Once again activate "Online & diagnostics" (see bottom picture).
 - 5. Make note of the Firmware version shown in the "Diagnostics -> General" tab.
 - 6. Close the window and then, in the Project tree, the "Online access".
- 2**

14.6.3. Exercise 3: Offline Project: Adding the ET 200SP



Task

An ET 200SP is to be inserted into the project as a distributed I/O station.

PROFINET IO-Devices are added in the Network view. Here, you can insert the relevant devices into the project by dragging & dropping them from the Hardware catalog.

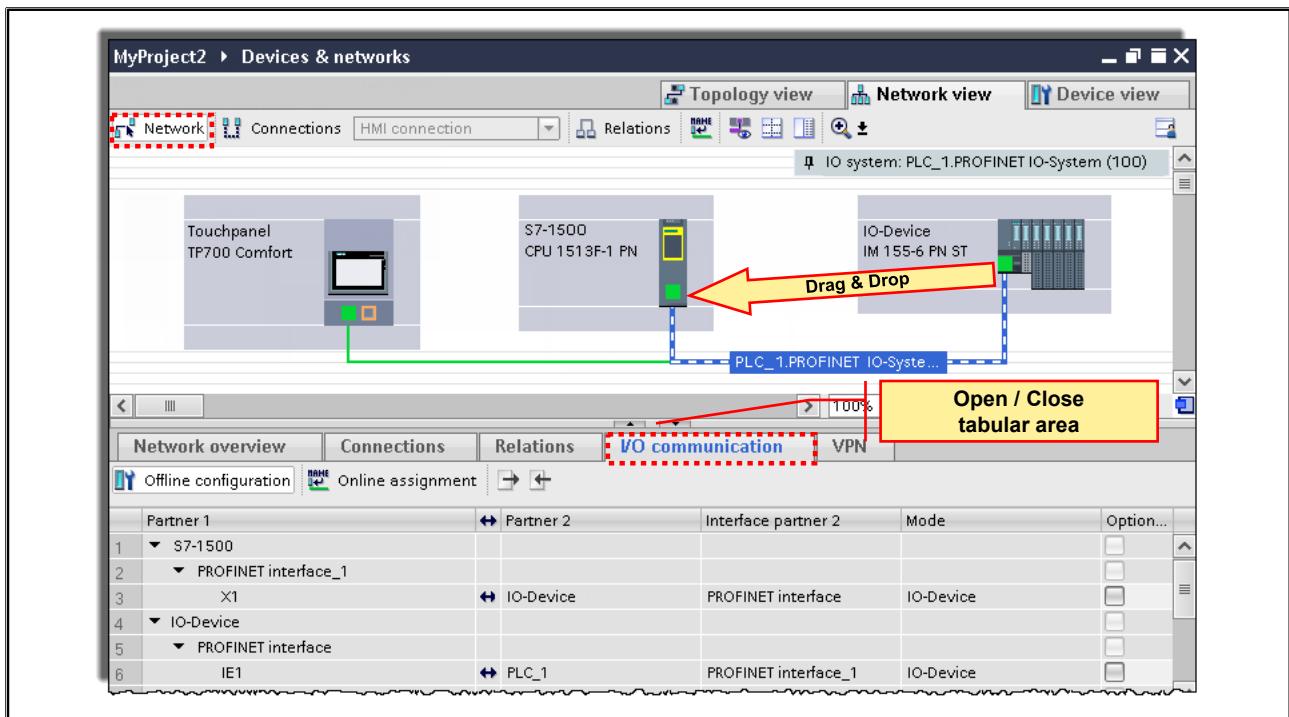
The newly added ET 200SP is stored in the Project tree in the "Ungrouped devices" folder and since it initially is not assigned to any controller, a reference is entered in the "Unassigned devices" folder.

What to Do

1. In the Project tree, open the "Hardware and Network editor" by double-clicking on it.
2. Open the Hardware catalog Task Card and there
Distributed I/O -> ET 200SP -> Interface modules -> PROFINET -> IM155-6PN ST
3. Select the IM module used in your training device, open the Information window and there select the previously read out Firmware version of your IM module.
4. Using drag & drop, drag the IM module into the "Hardware and Network editor" (see picture).

Leave all windows open because they are still needed for the next exercises!

14.6.4. Exercise 4: Networking the ET 200SP



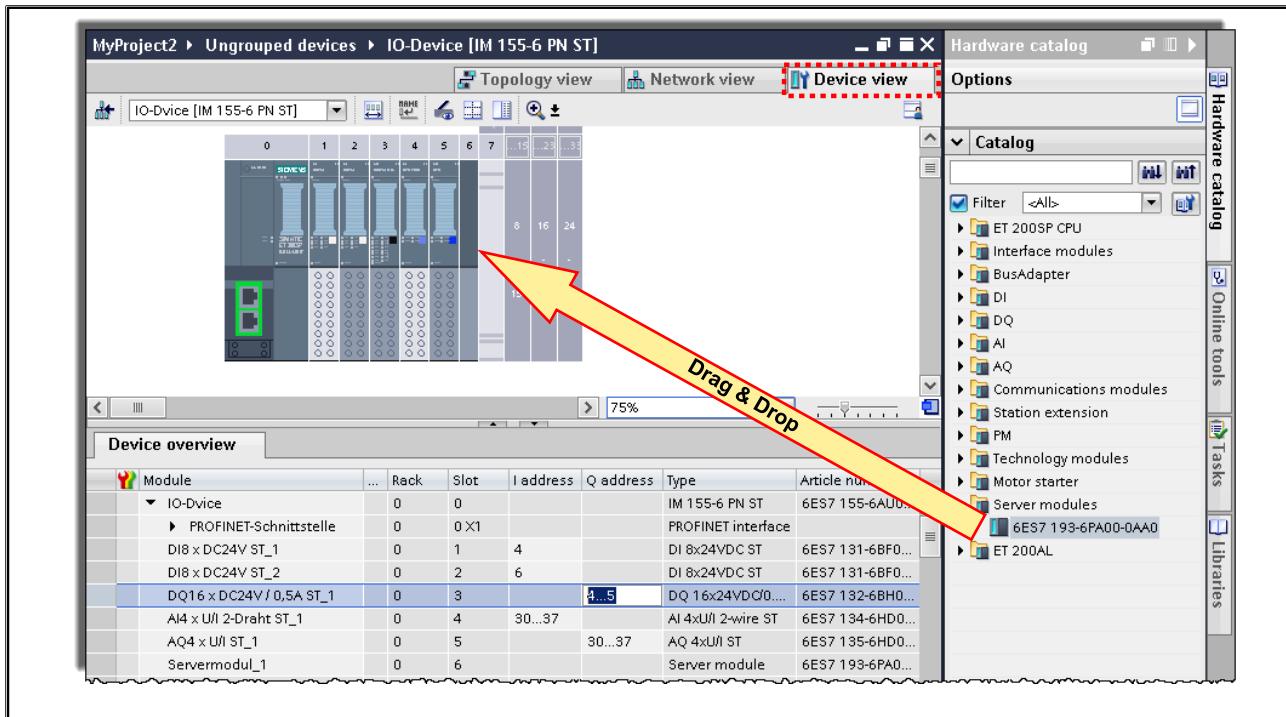
Task

After the ET 200SP IO-Device is added, it must now be assigned to an IO-Controller or networked with a CPU. In case there are several CPUs in the network, a co-ordination or monitoring of the I/O addresses by the IO-Controller and IO-Device can only be done through this unique assignment.

What to Do

1. In the "Hardware and Network editor", select the Network view and there click "Network" in the menu bar.
2. Network the ET 200SP with the CPU by connecting the Ethernet interface of the ET 200SP with the Ethernet interface of the CPU using drag & drop.
3. Select the newly created PROFINET IO system and, in the Inspector window under "I/O communication", check the generated communication partners.

14.6.5. Exercise 5: Configuring and Parameterizing the ET 200SP



Task

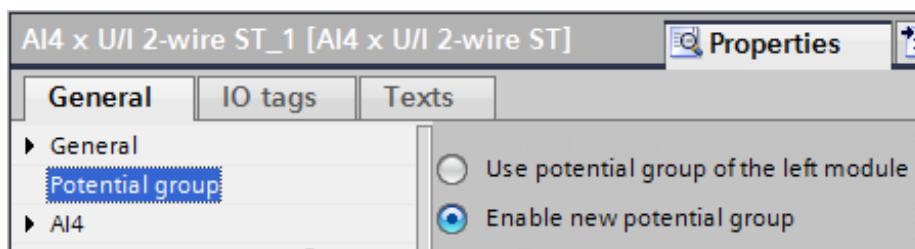
The configuration of the ET 200SP in the offline project must match exactly with the configuration of your training device. Attention should be given in particular to the order numbers and versions of the modules.

The ET 200SP has digital and analog input and output modules to which the conveyor model is to be connected in the following. The address assignment can be made in the Properties of the individual module, or, as can be seen in the picture, in the "tabular area" of the "Device view".

What to Do

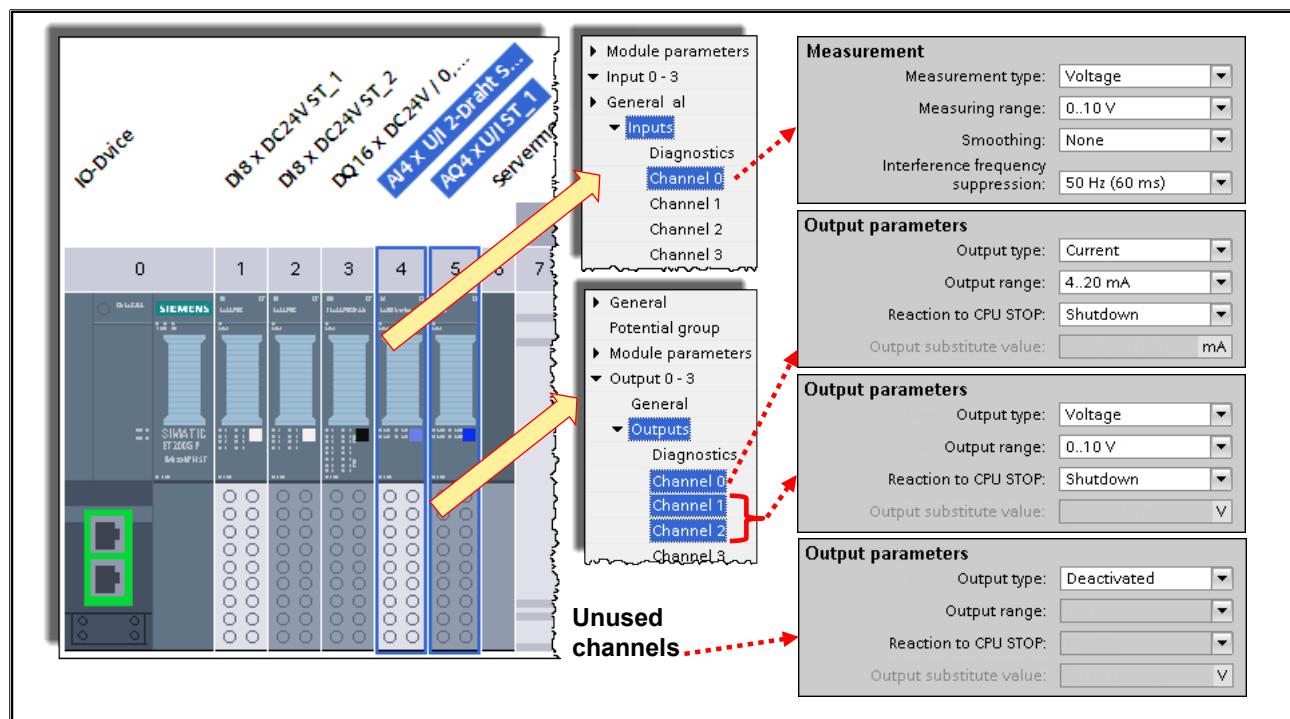
1. In the "Hardware and Network editor", select the "Device view" of the ET 200SP.
2. In the Task Cards, open the "Hardware catalog".
3. Configure the ET 200SP station according to your training device.

Make sure that a new potential group is opened with the AI module on Slot 4 and set this in the Properties:



4. Open the tabular area of the "Device overview" (see picture) and, in the table, enter the I/O addresses shown in the picture.
5. Save your project.

14.6.6. Exercise 6: Setting the Channel Parameters of the Analog Modules

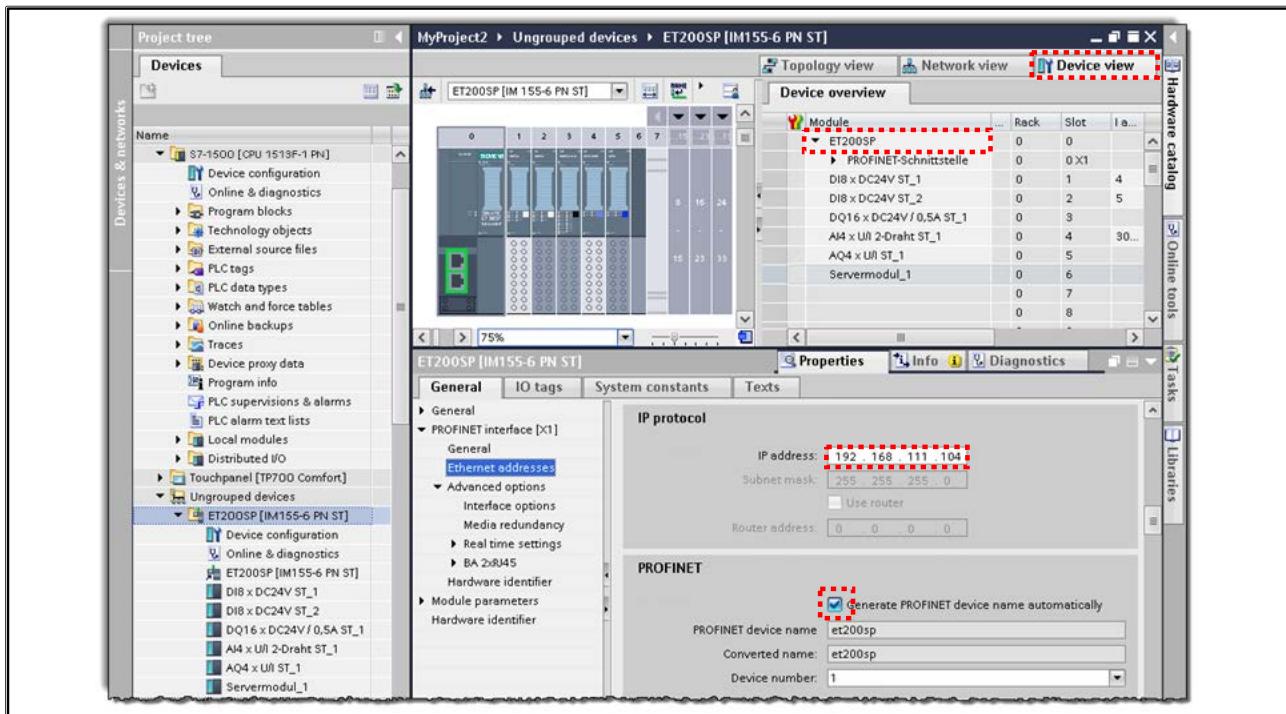


Task and What to Do:

Set the analog channels to the relevant parameters:

- Inputs
Channel 0 → Voltage 0..10V
- Outputs
Channel 0 → Current 4..20mA
Channel 1 → Voltage 0..10V
Channel 2 → Voltage 0..10V
- Deactivate all unused analog inputs and outputs.

14.6.7. Exercise 7: ET 200SP: Assigning the IP Address / PROFINET Name OFFLINE



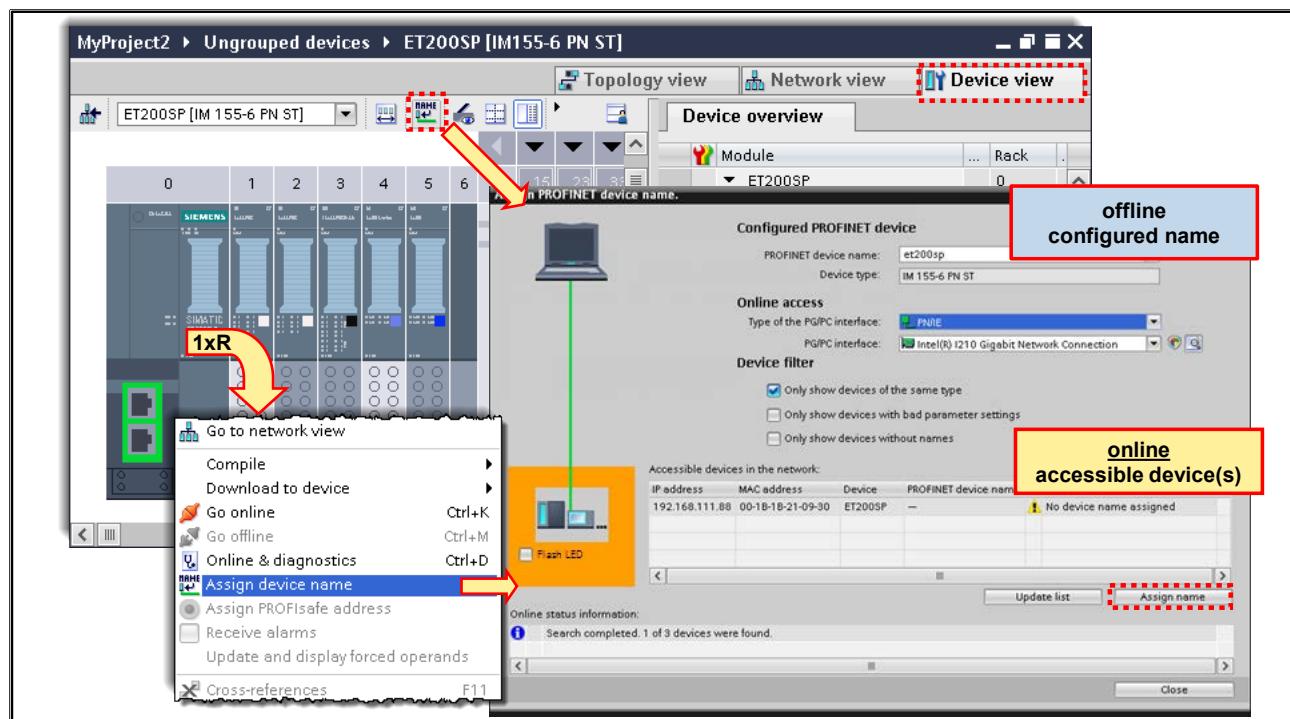
Task

The ET 200SP station is later to work with the IP address, subnet mask and PROFINET device name shown in the picture above.

What to Do

1. In the "Hardware and Network" editor, select the "Device view" of the ET 200SP.
2. Select the IM module on Slot 0 and open the "Properties" tab in the Inspector window.
3. There, in the "General" tab, select the "General" menu and under "Name" enter the PROFINET device name.
4. Then select the "Ethernet addresses" menu and under "IP protocol" enter the IP address and subnet mask shown (see picture). In the same menu you will also find the PROFINET device name that you previously edited in the "General" menu. If the property "Generate PROFINET device name automatically" is activated, it cannot be changed here.
5. Save your project.

14.6.8. Exercise 8: ET 200SP: Assigning the PROFINET Name ONLINE



Task

The PROFINET device name previously assigned offline must now be assigned to the ET 200SP online, so that the IO-Controller or the CPU can assign the offline-configured IP address during system startup of the ET 200SP.

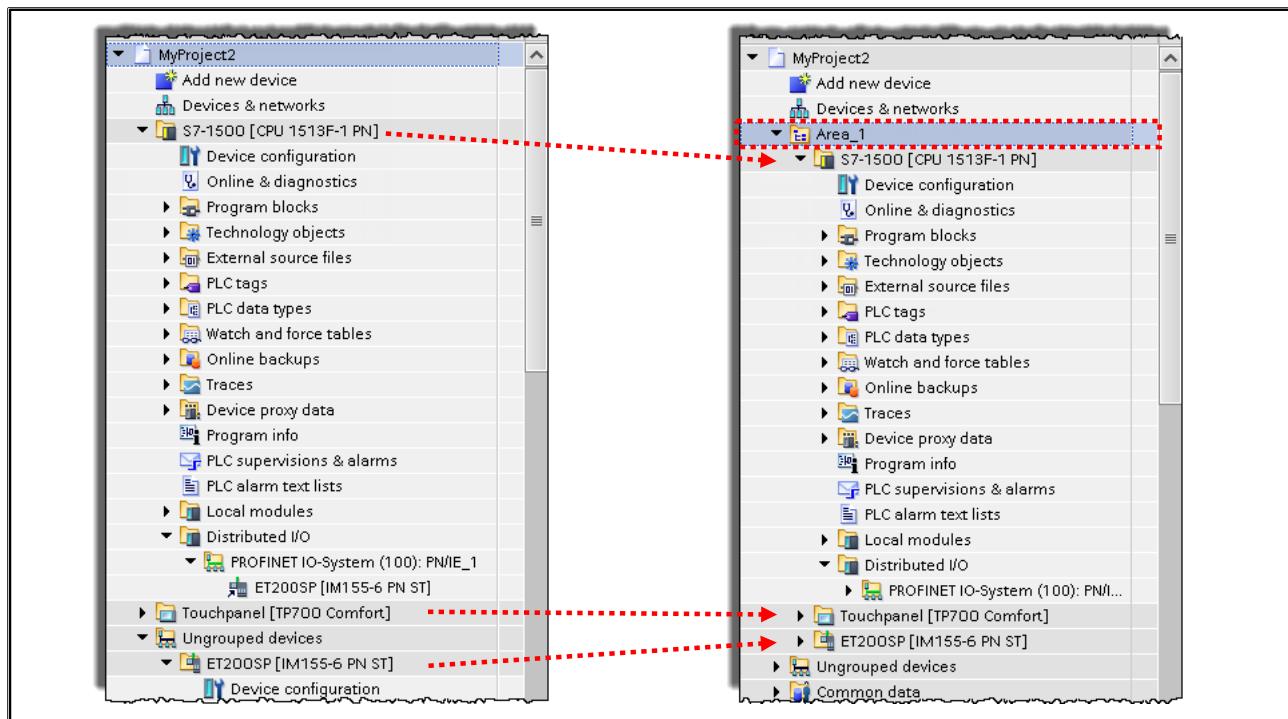
What to Do

1. In the "Hardware and Network editor", select the "Device view" of the ET 200SP.
2. Right-click on the Interface module or the module on Slot 0 and in the menu that appears, activate the item "Assign device name".
3. In the dialog that appears, check the (offline) PROFINET device name.
4. Under "Type of the PG/PC interface", select the interface through which you are connected to the PROFINET.
5. In the dialog table, select (highlight) the (online) "Accessible devices in the network" with the IP address which you assigned in Exercise 2 or (if known) with the relevant MAC address and assign it the OFFLINE name via the button "Assign name".

Note:

If the function "Flash LED" is selected in the dialog, then all LEDs on the device selected in the table flash. For a Panel, the screen would flash.

14.6.9. Exercise 9: Creating a New Device Group and Grouping Devices



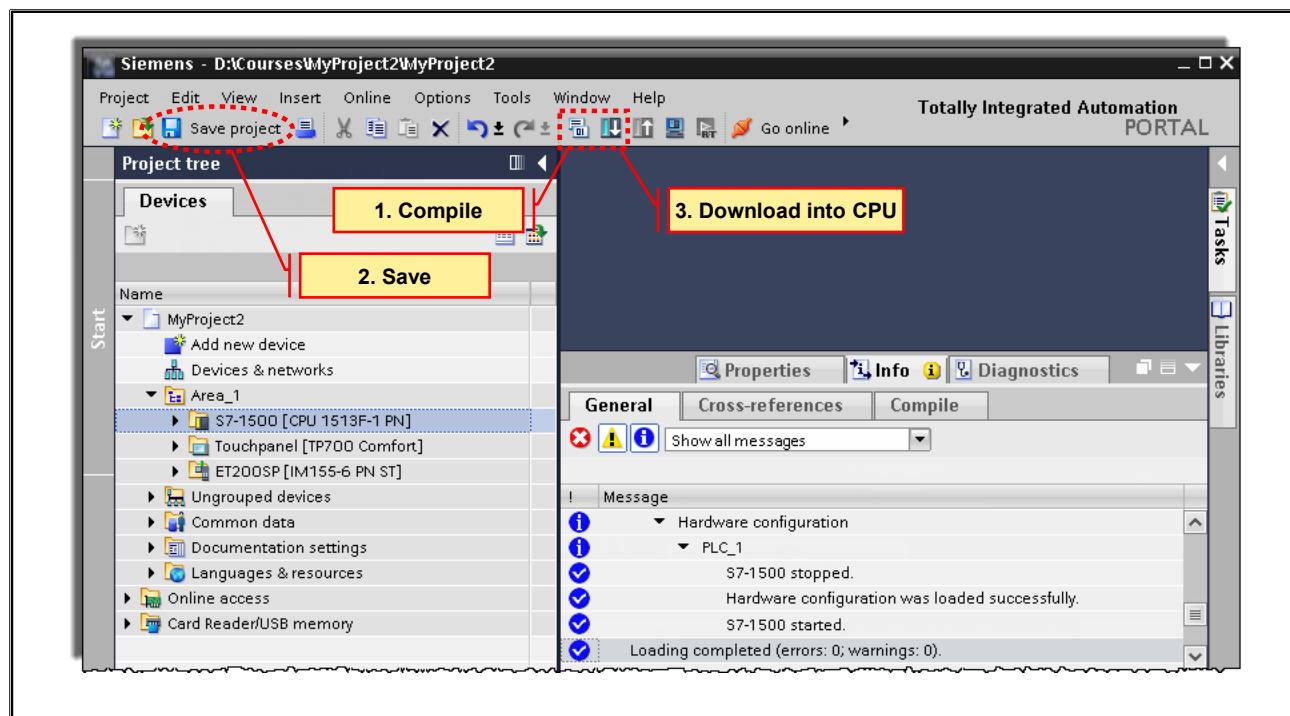
Task:

The devices (Controller, HMI and IO-Device) are to be stored in a common group.

What to Do:

1. Select the Project name and, through the context menu, insert a new device group.
2. Rename it "Area_1".
3. Move the Controller "S7-1500", the HMI "Touchpanel" and the IO-Device "ET 200SP" into the folder "Area_1".
4. Save your project.

14.6.10. Exercise 10: Compiling the Changes and Downloading them into the Device



Task

Now that the PROFINET I/O system is completely configured and parameterized, the project is to be compiled, saved and downloaded into the CPU.

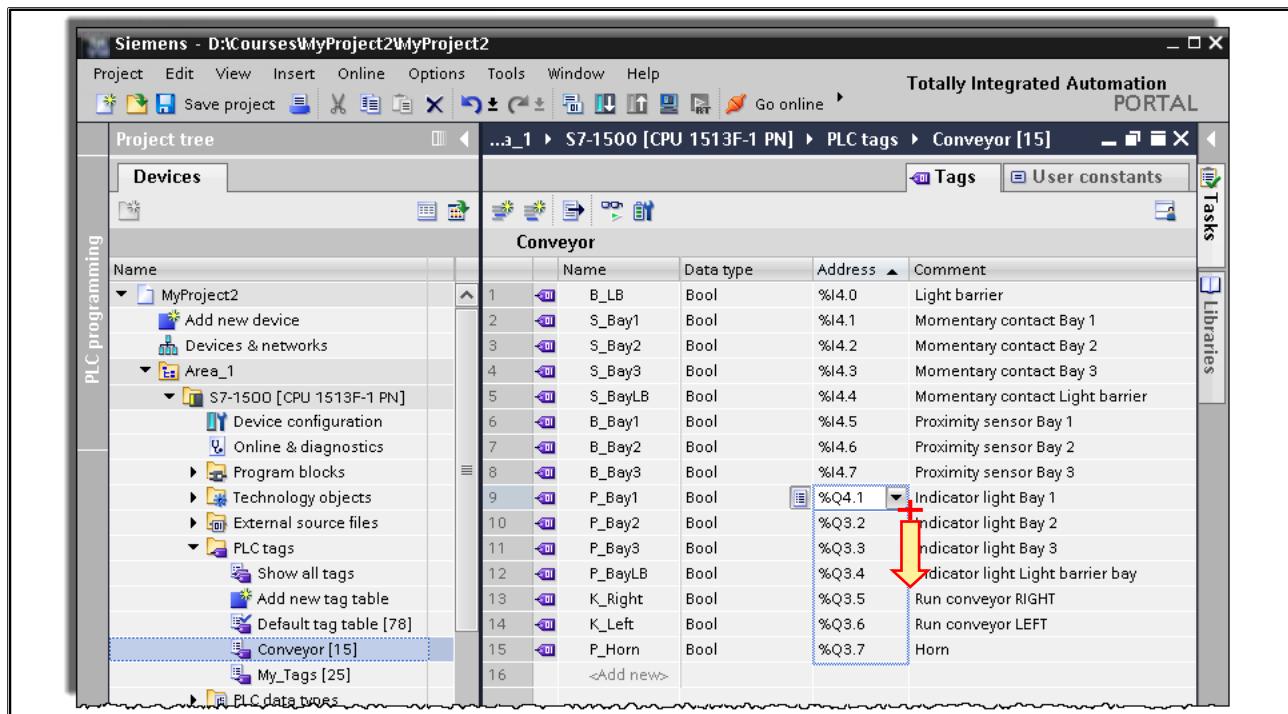
What to Do

1. Compile the changes by selecting the S7-1500 station in the Project tree and then clicking on the Compile button (see picture). In the Inspector window under "Info", check whether the compilation was successful. Should errors have occurred, correct them.
2. Save your project.
3. Download the entire station into the CPU by clicking on the Download button (see picture). In the Inspector window under "Info", check whether the loading was successful.
4. Check the module LEDs of your training device: Only green LEDs should be lit and not flashing!!!
5. Save your project.

Result:

Only green LEDs should be lit on the CPU as well as on all modules of the ET 200SP!

14.6.11. Exercise 11: Adjusting the S7 Program via "Rewiring"



Adjusting the S7 Program via "Rewiring"

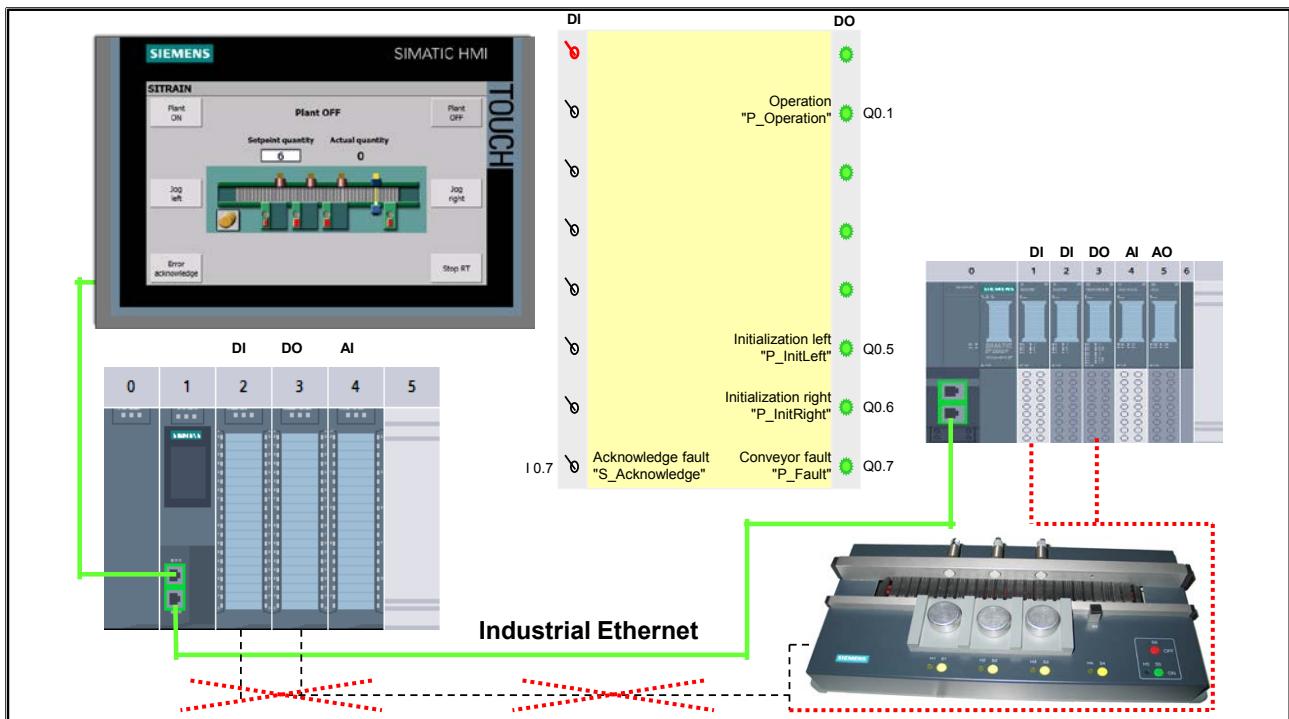
So that the conveyor model can be operated via the ET 200SP, the PLC tags (input and output addresses 3) used in the S7 program must be "rewired" to the I/O addresses of the ET 200SP modules (input and output addresses 4).

The "rewiring" can be carried out directly on the tag in the Blocks editor or, as shown in the picture, via the PLC tag table. All "rewiring" done here has immediate effect in the entire program.

What to Do:

1. Open the PLC tag table "Conveyor".
2. Sort the view according to addresses.
3. Change the address of the PLC tag "B_LB" from I 3.0 to I 4.0 and complete the entry with Return.
4. Select the address field, whereby a small square appears in the lower right corner.
5. Position the mouse pointer on this square so that the square changes to a cross (shown in the picture in red).
6. With the mouse pointer pressed down, drag the small square onto the address fields below it so that the change of the byte address is also adopted there.
7. Do the same for rewiring the outputs.
8. Save your project.
9. Compile, save and download the adjusted S7 program.

14.6.12. Exercise 12: Function Test with Conveyor Model via Distributed I/O



Task:

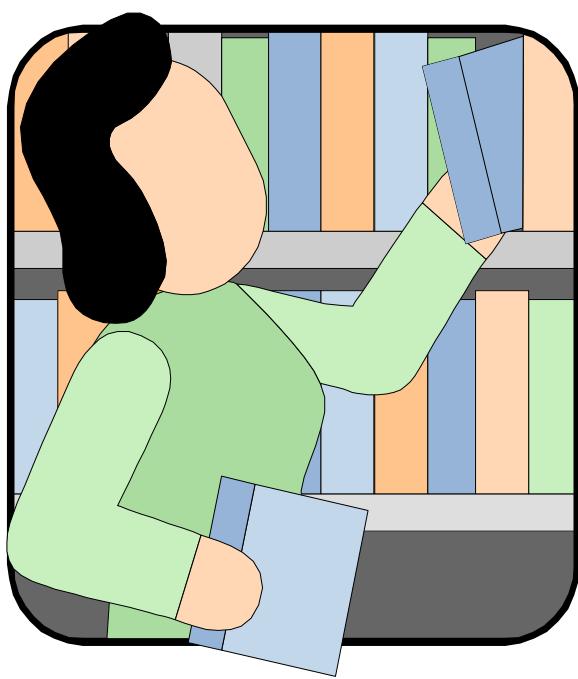
The conveyor model is now to be operated via the ET 200SP station. For this, the conveyor model connector cable must be connected to the SUB-D connector of the ET 200SP station on the backside of the training case.

With successful commissioning of the ET 200SP IO-Device and rewiring of the S7 program, the control of the conveyor model should function unchanged.

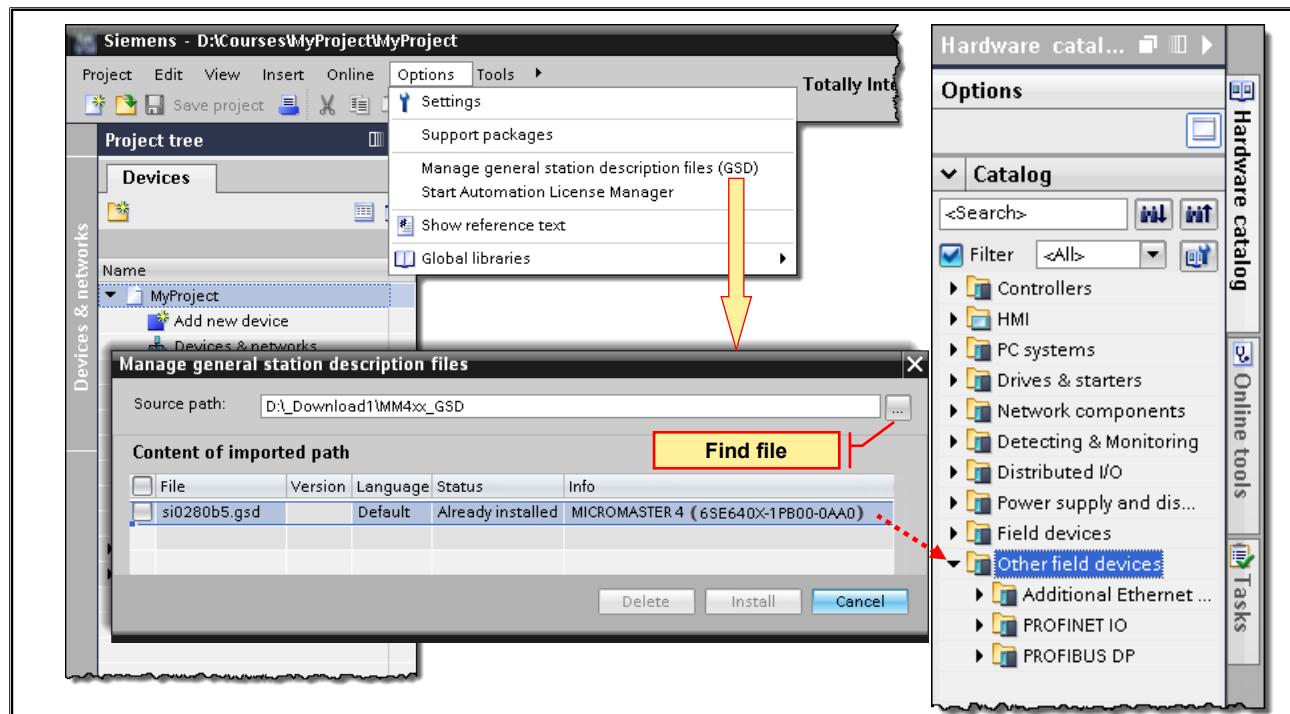
What to Do

1. Remove the conveyor model connector cable from the connector "S7-1500 DI/DO" on the backside of the training case and insert it in the connector "ET 200 DI/DO".
2. Check whether your system functions unchanged.

14.7. Additional Information



14.7.1. Installing Distributed Peripheral Components Later On via GSD



GSD (General Station Description) File

STEP 7 requires a GSD or a type file for every distributed peripheral slave so that the module can be configured from the Hardware catalog.

GSD files contain all necessary communication properties of a distributed peripheral module according to the PROFIBUS/PROFINET standard.

A GSD file is normally supplied with distributed peripheral modules or is available as a download from the Internet.

Many SIEMENS modules are already generally a part of the Hardware catalog.

Modules Added to the Hardware Catalog

Following installation, these are entered in the "Other field devices" section and are available there for the configuration.

15

Contents

15. Troubleshooting	15-3
15.1. Categories of Errors	15-4
15.2. STEP 7 - Test Functions, Overview	15-5
15.3. Overview Window	15-6
15.3.1. Global Search / Find and Replace in the Editor	15-7
15.3.4. Detailed Information in the Overview of Addresses	15-8
15.5. System Diagnostics - Overview	15-9
15.6. Status LEDs	15-10
15.6.1. Status LEDs of the S7-1500 CPU	15-10
15.6.2. Status LEDs of the Central DI/DO Modules of the S7-1500 CPU	15-11
15.7. Hardware Diagnostics	15-12
15.8. Online & Diagnostics: General	15-13
15.8.1. Online & Diagnostics: CPU Diagnostics Buffer	15-14
15.8.2. CPU Diagnostics Buffer: Interpreting Error Messages	15-15
15.8.3. CPU Diagnostics Buffer: Opening a Faulty Block	15-16
15.9. Call Hierarchy (Block Stack)	15-17
15.9.1. Exercise 1: Creating a Program Backup Copy in the Project Library	15-18
15.9.2. Exercise 2: Copying the Faulty Program	15-19
15.9.3. Exercise 3: STOP Troubleshooting Worksheet	15-20
15.10. Monitor Block (Block Status)	15-21
15.10.1. Monitor Block: Modify Tags	15-22
15.10.2. Monitoring Structures	15-23
15.10.3. Monitor Block: Trigger Conditions / Call Environment	15-24
15.11. Monitor / Modify Variables (Tags): Watch Tables	15-25
15.11.1. Monitor / Modify Variables (Tags): Trigger Points	15-26
15.11.2. Enable Peripheral Outputs (in planning for S7-1500)	15-27
15.11.3. Force Variables (Tags)	15-28
15.12. Reference Data: Cross-references of Tags	15-29
15.12.1. Reference Data: Cross-references / Show Overlapping Accesses	15-30
15.12.2. Reference Data: Cross-references of a Variable (Tag) in the Block Editor	15-31
15.12.3. Go To	15-32
15.12.4. Reference data: Assignment I/Q/M/T/C	15-33
15.12.5. Reference Data: Call Structure	15-34
15.12.6. Reference Data: Dependency Structure	15-35
15.13. Resources	15-36
15.14. Reference Projects	15-37
15.15. Compare (1) - Offline / Online	15-38
15.15.1. Compare (2) – Block Detailed Comparison	15-39
15.15.2. Compare (3) - Software Offline / Offline	15-40

15.15.3. Compare (4) - Hardware Offline / Offline	15-41
15.16. Exercise 4: Testing the Motor Jog	15-42
15.17. Exercise 5: Entering the Setpoint Quantity	15-43
15.18. Exercise 6: Testing the Evaluation of Fault 3	15-44
15.19. TRACE Analyzer Function	15-45
15.19.1. Configuring a TRACE - Signals and Sampling	15-46
15.19.2. Configuring a TRACE – Trigger and Saving Measurement on Device.....	15-47
15.19.3. Downloading a TRACE into the CPU and Activating It.....	15-48
15.19.4. Evaluating, Saving, Exporting a TRACE in STEP 7	15-49
15.19.5. Trace Task Card	15-50
15.20. Task Description: Creating, Looking at and Saving a TRACE	15-51
15.20.1. Exercise 7: Creating a Cyclic Interrupt OB for the Recording Level.....	15-52
15.20.2. Exercise 8: Creating, Looking at and Saving a TRACE	15-53
15.21. Additional Information	15-55
15.21.1. Diagnostic Information about the SIMATIC Memory Card	15-56
15.21.2. Reading-out the Checksum	15-57

15. Troubleshooting

At the end of the chapter the participant will...

- ... be familiar with the search functions
- ... be familiar with the possibilities of troubleshooting
- ... be familiar with the possibilities of project analysis
- ... be able to apply troubleshooting functions for STOP troubleshooting and to be able to use them for error correction
- ... be able to apply troubleshooting functions for searching for logical errors and to be able to use them for error correction
- ... be able to use the Trace analysis function

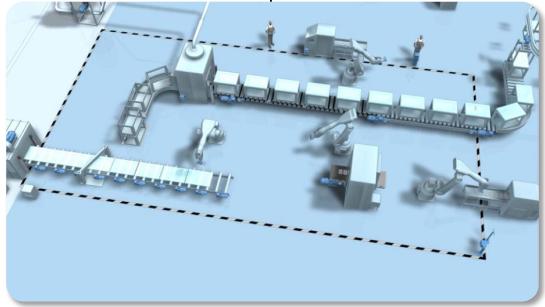


15.1. Categories of Errors

Errors Detected by the System

- Acquiring, evaluating and indicating errors within a PLC
 - Module failure
 - Short-circuit in signal cables
 - Scan time overrun
 - Programming error (accessing a non-existent block)





Functional Errors

- Desired function is either not executed at all or is not correctly executed
 - Process fault (sensor/actuator, cable defective)
 - Logical programming error (not detected during creation and commissioning)

Monitoring Functions

Diagnosis is important in the operating phase of a system or machine. Diagnosis usually occurs when a problem (disturbance) leads to standstill or to the incorrect functioning of the system or machine. Due to the costs associated with downtimes or faulty functions, the associated cause of the disturbance has to be found quickly and then eliminated.

Categories of Errors

Errors that occur can be divided into two categories, depending on whether or not they are detected by the PLC:

- Errors that are detected by the PLC's operating system and that normally lead to the Stop state of the CPU.
- Functional errors, that is, the CPU executes the program as usual, but the desired function is either not executed at all or it is executed incorrectly. The search for these types of errors is much more difficult, as a rule, since the cause of the error is initially hard to determine.

Possible causes could be:

- A logical programming error (software error) that was not detected during creation and commissioning of the user program and probably occurs only on extremely rare occasions.
- A process fault that was triggered by the faulty functioning of components directly associated with the process control, such as cables to sensors/actuators or by a defect in the sensor/actuator itself.

15.2. STEP 7 - Test Functions, Overview

Errors Detected by the System General rule: CPU in Run	Functional Errors General rule: CPU in RUN
<ul style="list-style-type: none"> • Programming error (w/o OB121): STOP • Access error (w/o OB122): RUN • Asynchronous error (w/o OB82, 83, 86): RUN <p>• Online & Diagnostics - Diagnostics buffer,</p> <p>• Task Card "Testing" - Call hierarchy/Block stack - Local data stack (in planning stage)</p> <p>• Diagnose Modules - Diagnostic status (all modules)</p>	<ul style="list-style-type: none"> • Process fault (e.g. wire break) • Logical programming error (e.g. double assignment) <p>• Monitor and Modify Variables → Watch and Force tables</p> <p>• Monitor Blocks (Block Status) → Monitor in the Blocks editor - with call environment</p> <p>• Tools - Cross references - Assignment list (I/Q/M/T/C)</p> <p>• "Trace" analyzer function • Program/Block comparison • Set breakpoints</p>

Test Functions

There are various STEP 7 test functions for troubleshooting, depending on the type of error caused:

- when CPU in STOP

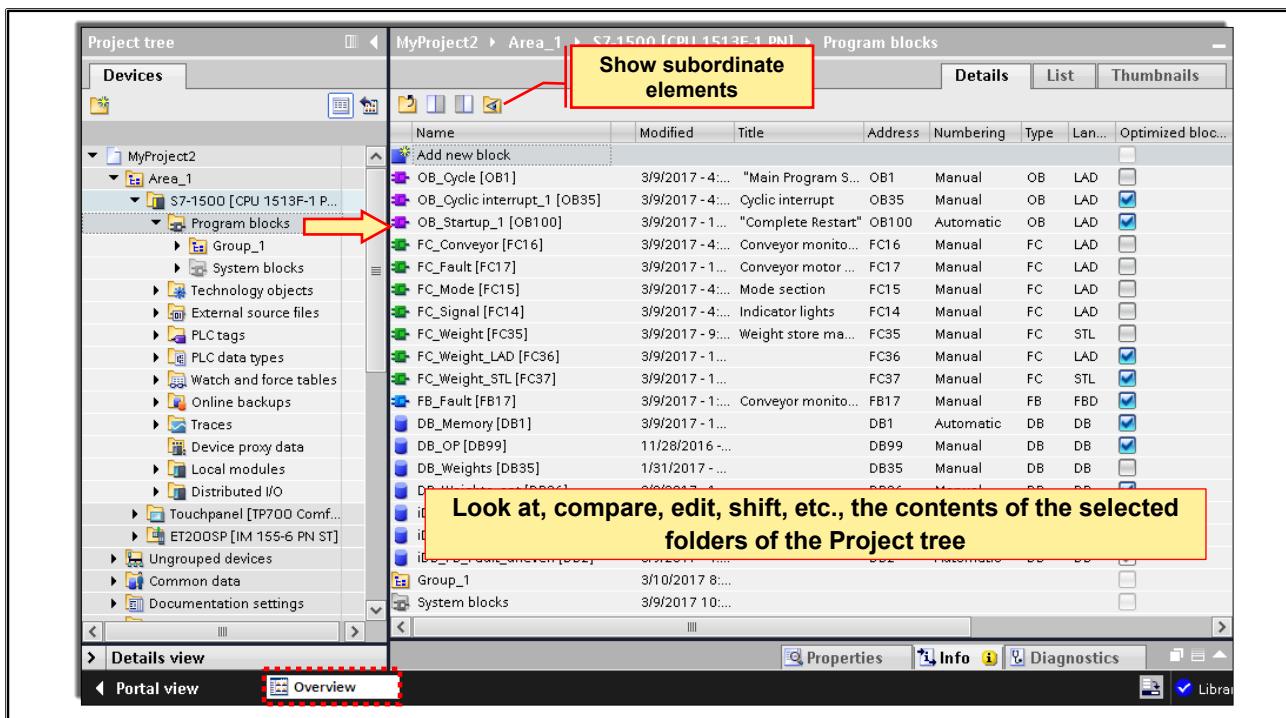
For errors that are detected by the system, the test functions Diagnostics buffer, Call hierarchy/Block stack, Local data stack and Hardware diagnostics give detailed information on the cause of the error and the location of the interruption. By programming Error OBs, information on the error that occurred can be evaluated by program and the transition of the CPU into the STOP state can be prevented. If the CPU has stopped, the use of the test functions Monitor / Modify Variable and Monitor Blocks makes little sense since the CPU neither reads nor outputs process images while in the STOP state, and also no longer executes the program.

- when CPU in RUN

Vice versa, it makes little sense, as a rule, to use test functions such as Local data stack for troubleshooting when the CPU is in RUN, since program execution has not been interrupted and the system does not provide any information on the error that occurred. The Module Information test function merely provides general information on the CPU's operating status or on errors that occurred in the past. Functional errors can be diagnosed as follows:

- Process Fault (such as a wiring error)
Wiring test of the inputs: Monitor Variable
Wiring test of the outputs: Enable Peripheral Outputs (only for CPU-STOP)
- Logical Programming Errors (such as a double assignment)
All test functions listed, with the exception of Enable Peripheral Outputs, can be used for searching for logical program errors.
- Force: Forced control of operands regardless of the program logic
- Breakpoints: Program execution in single steps

15.3. Overview Window



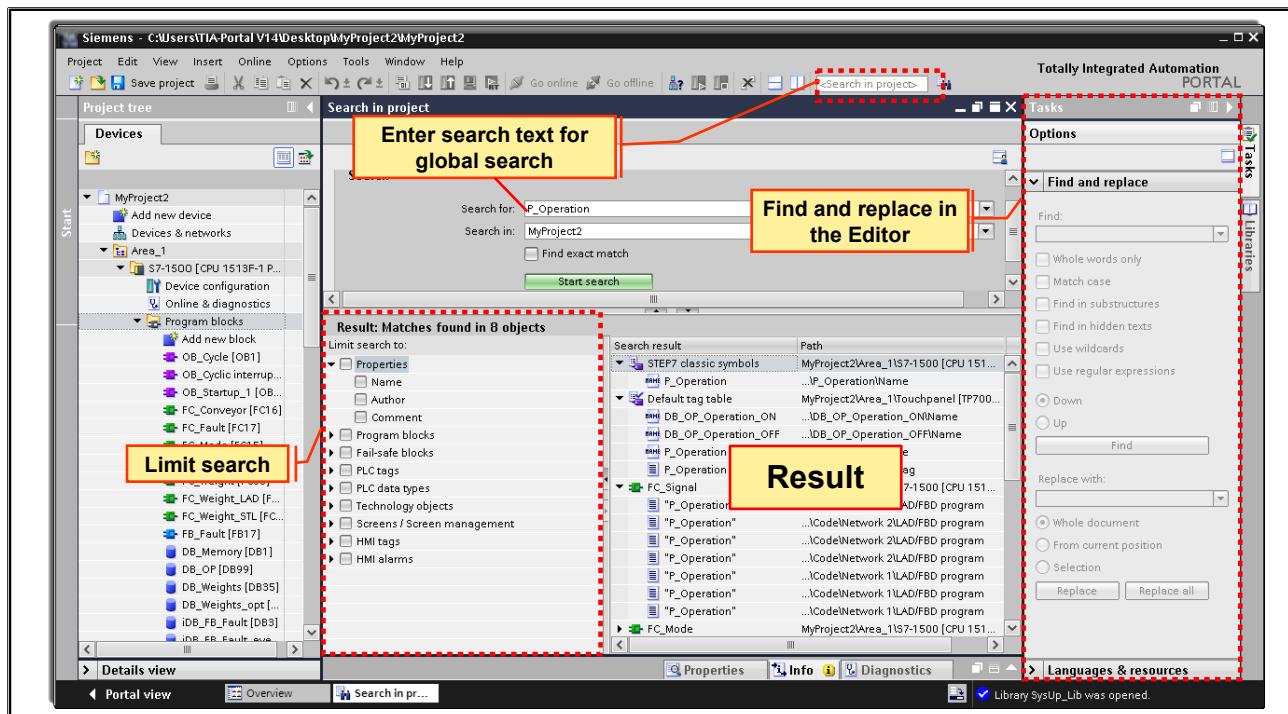
The Overview window is used to supplement the Project tree. The Overview window displays the contents of the folder currently selected in the Project tree.

As well, you can carry out the following actions in the Overview window:

- Open objects
- Display and edit Properties of objects in the Inspector window
- Rename objects
- Call object-specific actions via the Context menu
- Compare objects
- Carry out different object operations, such as, insert objects from the library using drag & drop, shifting, copying, inserting and deleting objects

It is possible to compare the contents of two folders or objects. To do so, the View can be split. Beyond that, it is possible to shift objects between the two split windows using drag & drop.

15.3.1. Global Search / Find and Replace in the Editor



Within the TIA Portal, you can use the following search possibilities:

- Search entire project (toolbar or CTRL+F when the focus is not in the editor (working) area)
- Find and replace within an editor ('Tasks' task card > Find and replace)
- Search the Hardware catalog

Search Entire Project

You can search the entire project for a specific text. For this, there is a Search editor available in which you can, for example, narrow down the search. The objects which contain the searched-for text are presented clearly in a table. You can open each object from the Search editor in order to look at the relevant location.

The Global Search (Search in project) can be started by means of CTRL+F (when the focus/selection is not in the editor (working) area), or, via the Context menu of the project name, or, via the menu bar, or, via the menu Edit > Search in project.

If the focus is in an editor, then the function "Find and replace" is started with the key combination CTRL+F.

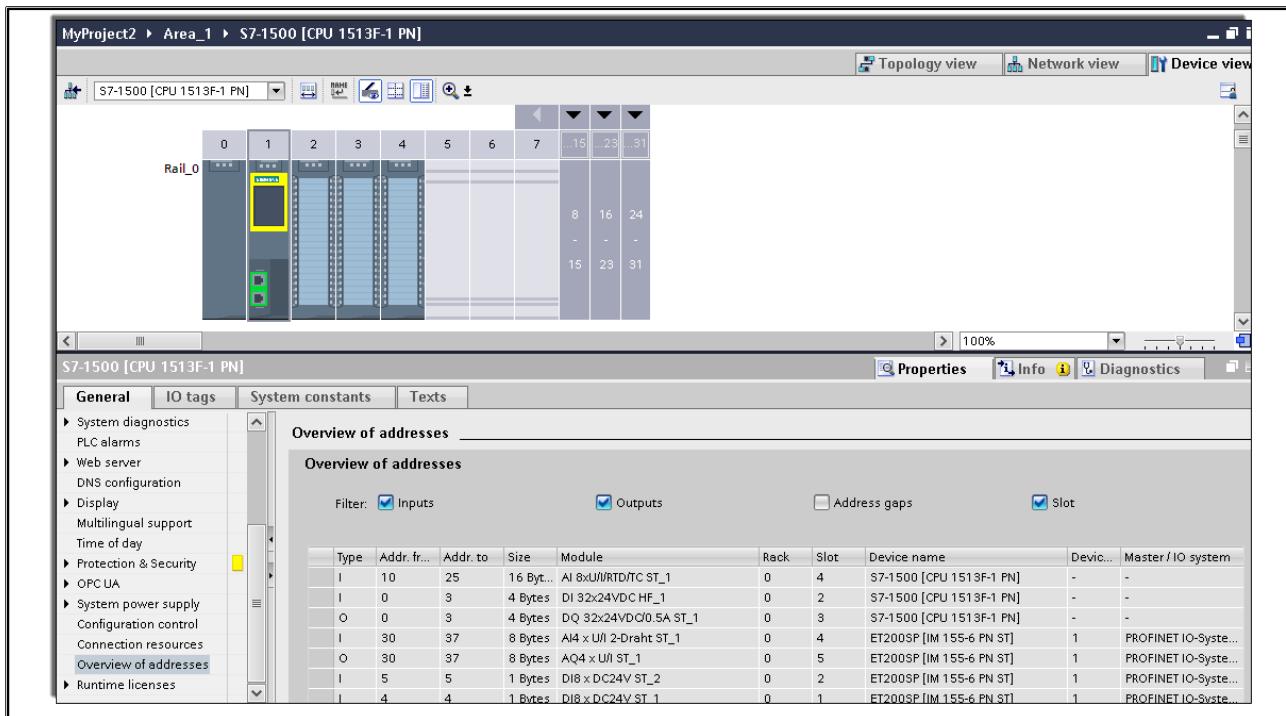
Search and Replace within an Editor (CTRL+F when the focus is in the editor (working) area)

It is possible to search for texts within an editor. The search function finds all texts within the currently opened editor which contain the search term. The results are selected one after the other in the opened editor.

Furthermore, you have the following possibilities:

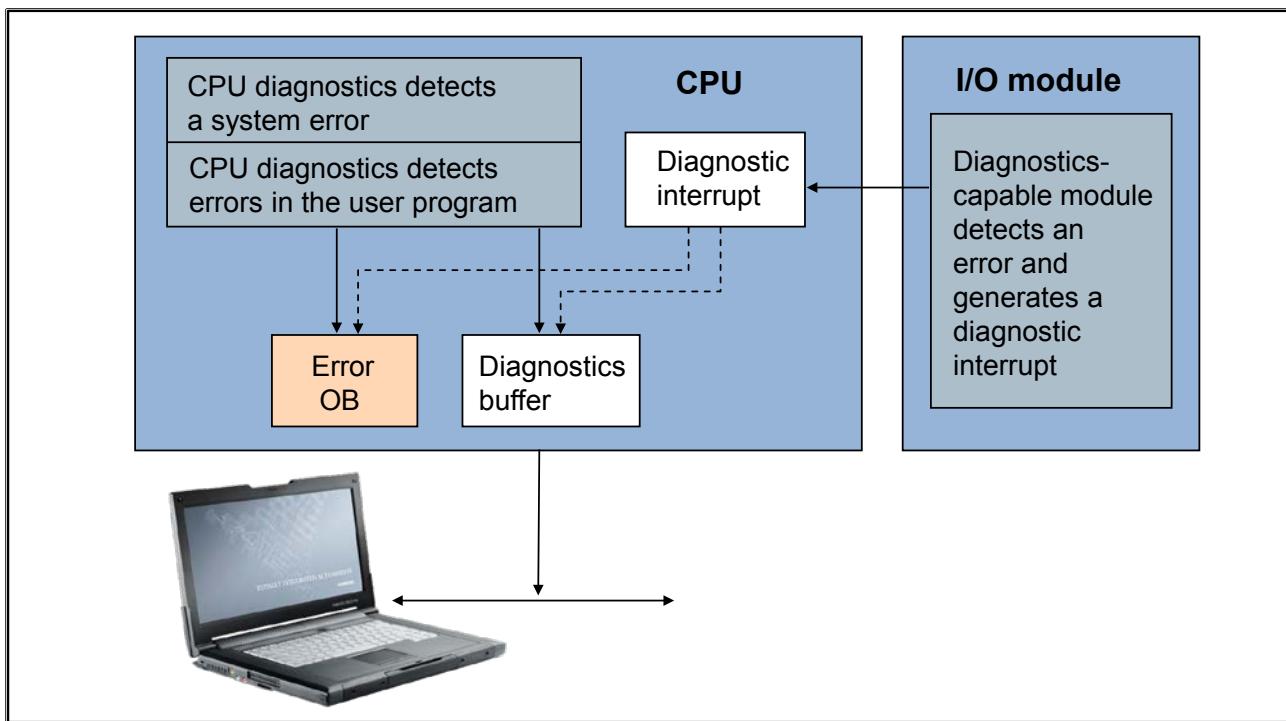
- Refine the search through additional options
- Replace found text

15.4. Detailed Information in the Overview of Addresses



In menu 'Overview of addresses' in the Properties of the CPU, an entire overview of addresses with central and distributed I/O can be displayed.

15.5. System Diagnostics - Overview



System Diagnostics

All those monitoring functions that deal with the correct functioning of the components of an automation system are grouped together under System Diagnostics. All S7-CPUs have an intelligent diagnostics system. The acquisition of diagnostic data by the system diagnostics does not have to be programmed. It is integrated in the operating system of the CPU and in other diagnostics-capable modules and runs automatically. The CPU (temporarily) stores errors that occur in the diagnostics buffer and thus enables a fast and targeted error diagnosis by service personnel, even for sporadically occurring errors.

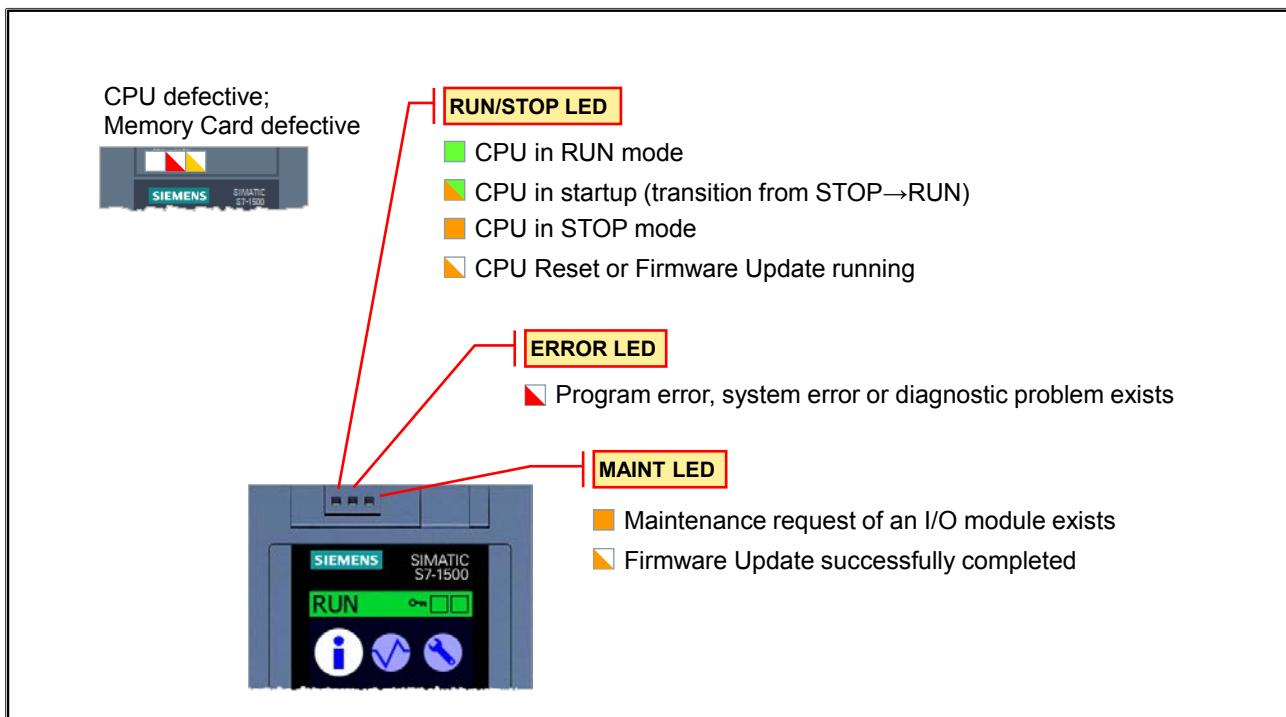
System Reaction

The operating system takes the following actions when it detects an error or a STOP event, such as an operating mode change (RUN → STOP):

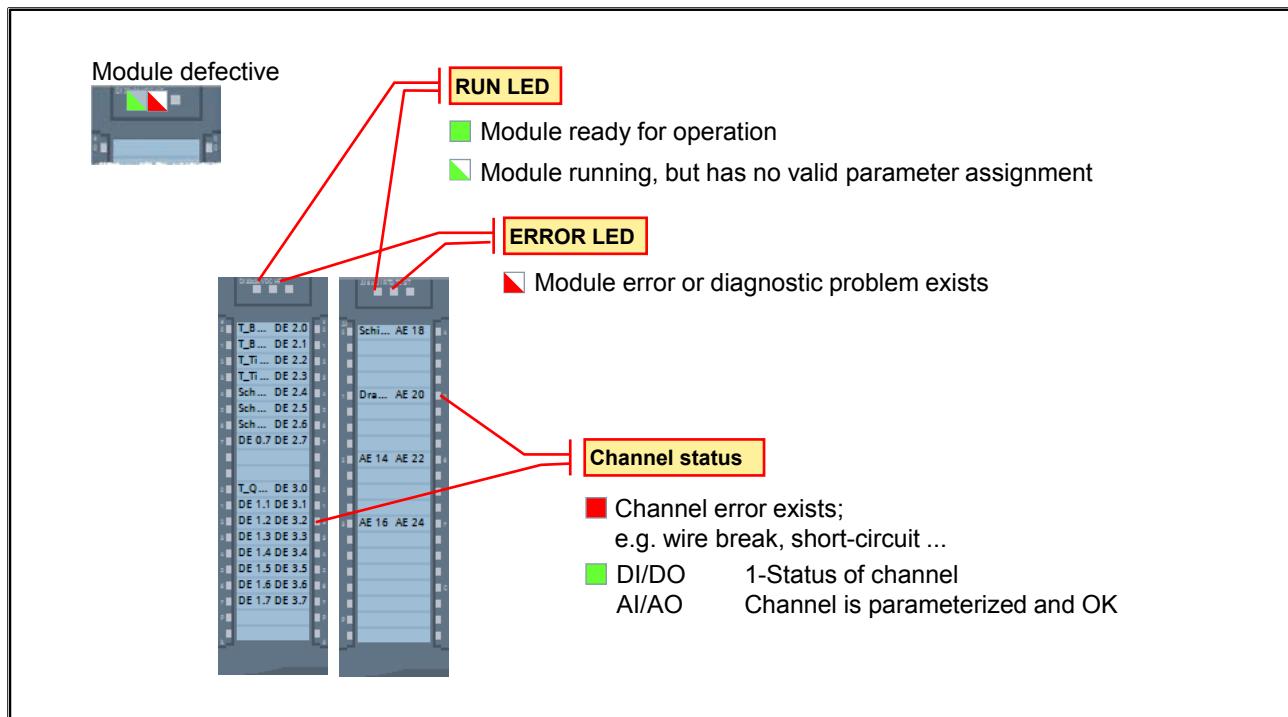
- A message on the cause and the effect of the occurring error is entered in the diagnostics buffer, complete with the date and time. The diagnostics buffer is a FIFO (circular) buffer on the CPU module for storing error events. The size of the diagnostics buffer depends on the CPU. In the FIFO buffer structure, the most recently entered message overwrites the oldest diagnostics buffer entry. A CPU Memory Reset cannot delete the diagnostics buffer.
- The Error OB associated with this error is called. This gives the user the opportunity of carrying out an own error handling.

15.6. Status LEDs

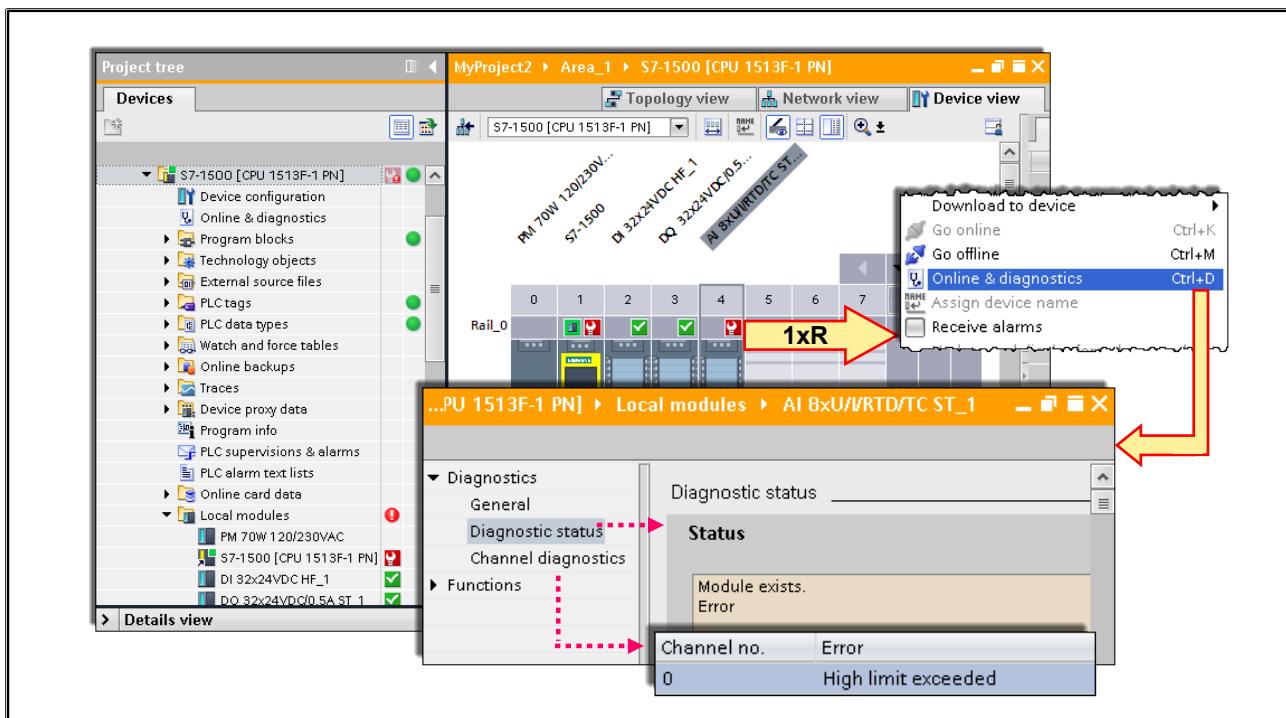
15.6.1. Status LEDs of the S7-1500 CPU



15.6.2. Status LEDs of the Central DI/DO Modules of the S7-1500 CPU



15.7. Hardware Diagnostics



Diagnosing Hardware

To use this function you must open the "Device configuration" and establish an online connection. The online view of the hardware gives information about the status or operating status of the modules. You can see that there is diagnostic information for a module when you see the diagnostic symbols that indicate the status of the associated module or the operating status of the CPU.

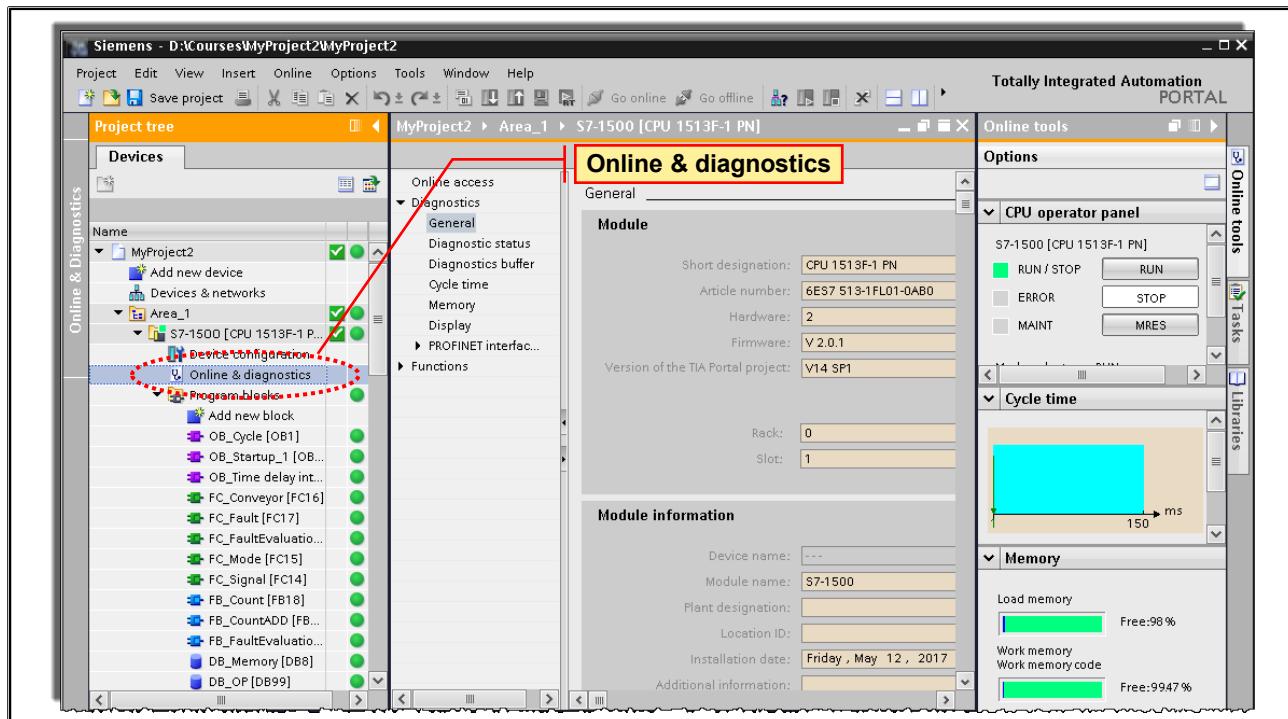
In the example shown, the analog input module (slot 4) has triggered a diagnostic interrupt. As a result, the CPU has gone into the STOP mode.

Both modules have been given symbols accordingly. Via the Context menu of the CPU and subsequent starting of "Online & diagnostics", the CPU's diagnostic buffer is output; for the analog module, the associated diagnostic data (see picture).

Symbol Meanings

		Icon	Meaning
Icon	Meaning		
!	Hardware error in lower-level component: The online and offline versions differ (only in the project tree) in at least one lower-level hardware component.		
✓	No fault		
!	Maintenance required		
!	Maintenance demanded		
!	Error		
			Online and offline versions of the object are different
			Object only exists online
			Object only exists offline
			Online and offline versions of the object are the same

15.8. Online & Diagnostics: General

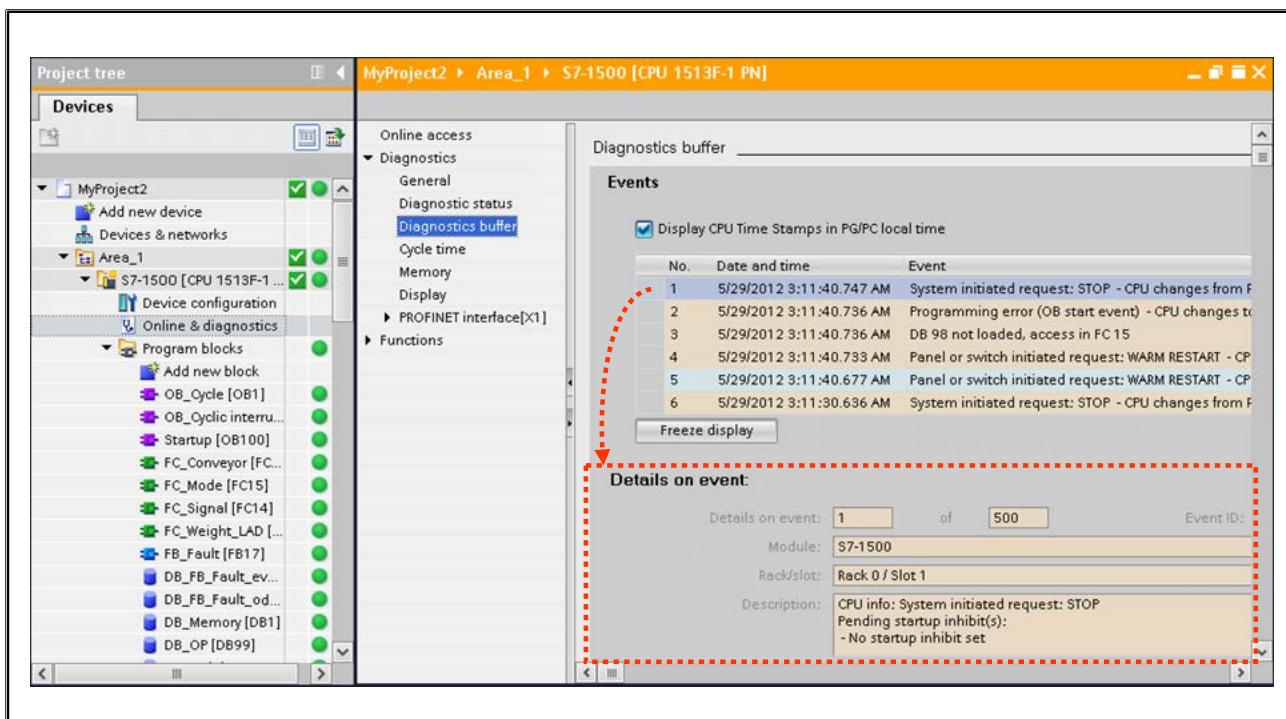


Online and Diagnostics

The Module Information function reads the most important data from the directly connected module. You will find additional information in the individual tabs:

- General: Among other things, the module designation, hardware and firmware versions
- Diagnostic status: Current status of the module
- Diagnostics buffer: It contains all diagnostic events in the order they occurred. In the display, all events are listed in plain language and in the order they occurred.
- Cycle time: Displays the selected minimum time and monitoring time as well as displaying the shortest, the longest and the current cycle time.
- Memory: Information about the entire size, how many bytes are used (occupied) and how many are free in the Load memory, Code work-memory, Data work-memory, and Retain memory.
- Display: General information about the Display used (Article number, Firmware etc.)

15.8.1. Online & Diagnostics: CPU Diagnostics Buffer



Diagnostics Buffer

The diagnostics buffer is a buffered memory area on the CPU organized as a circular buffer. It contains all diagnostics events (error messages, diagnostic interrupts, startup information etc.) of the CPU in the order in which they occurred. The highest entry is the last event to occur.

All events can be displayed on the programming device in plain language and in the order in which they occurred.



The size of the diagnostics buffer depends on the CPU. As well, not all of the diagnostics buffer is buffered with PowerOFF (only a part is retentive).

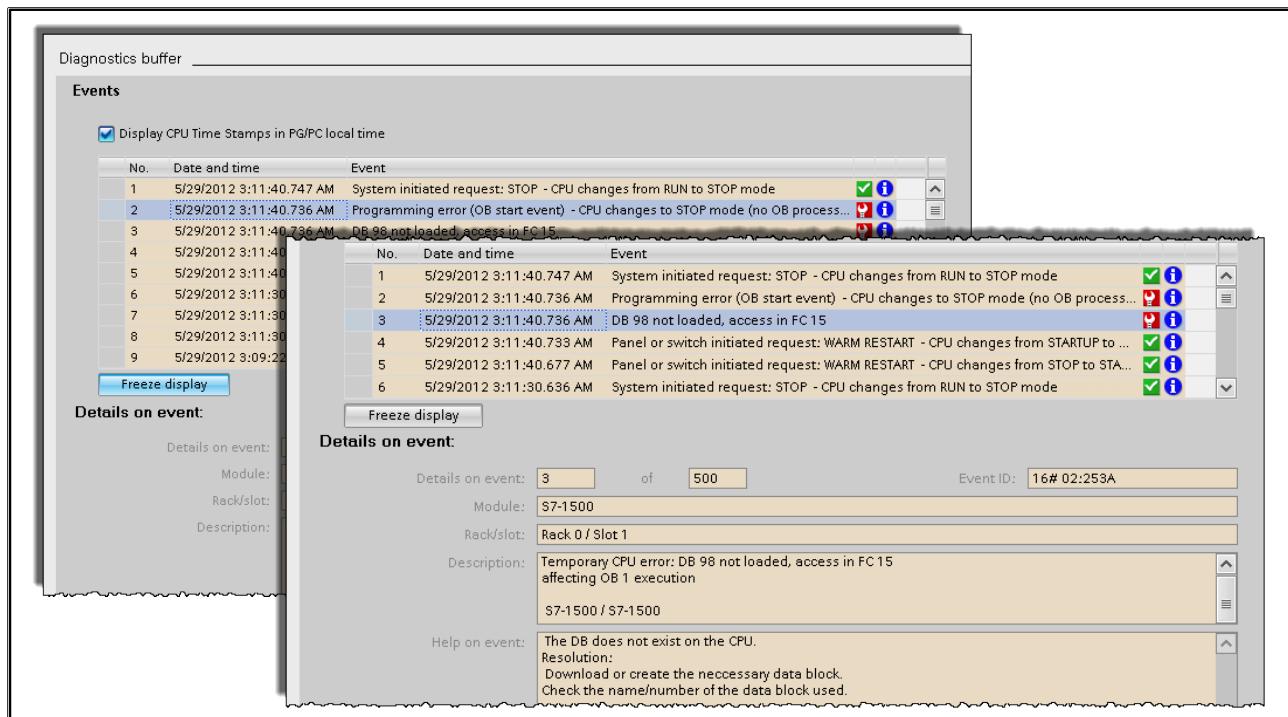
- Number of entries, 1000 to 3200
- Of that, 500 to 1000 retentive

Details on Event

Some additional information is also provided for the selected event in the "Details on event" box:

- Event name and number,
- Additional information depending on the event, such as, the address of the instruction that caused the event etc.

15.8.2. CPU Diagnostics Buffer: Interpreting Error Messages



Interpreting the Diagnostics Buffer

To interpret the diagnostics buffer, you have to look at the events that belong together in the sequence in which they occurred, in other words, from bottom to top:

For orientation:

- In our example, a WARM RESTART was performed before the most recent error occurred (events no.4 and 5).

Entries in the Diagnostics Buffer

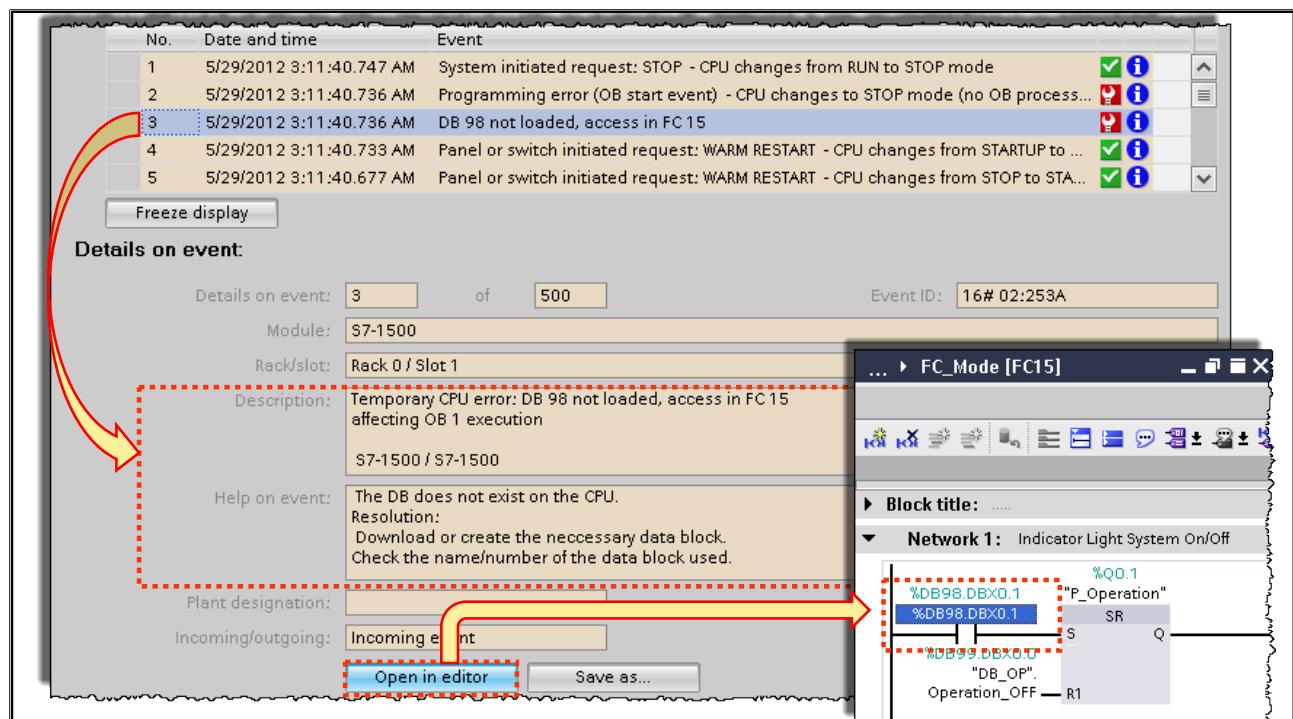
The last error that occurred after this warm restart leads to the following entries in the diagnostics buffer:

- Event No. 3:
DB 98 not loaded, access in FC15
- Details on event:**
- DB 98 not loaded, access in FC15
(DB does not exist on the CPU)
 - affects OB1 execution
(FC15 is called in the cyclic program)

- Event No. 2:
Programming error (OB start event)
(if it exists, the operating system calls an OB for a programming error)

- Details on event**
- CPU changes to STOP mode
 - no OB processing
(because the programming error OB121 was not programmed)
- Event No. 1:
CPU changes to STOP mode

15.8.3. CPU Diagnostics Buffer: Opening a Faulty Block

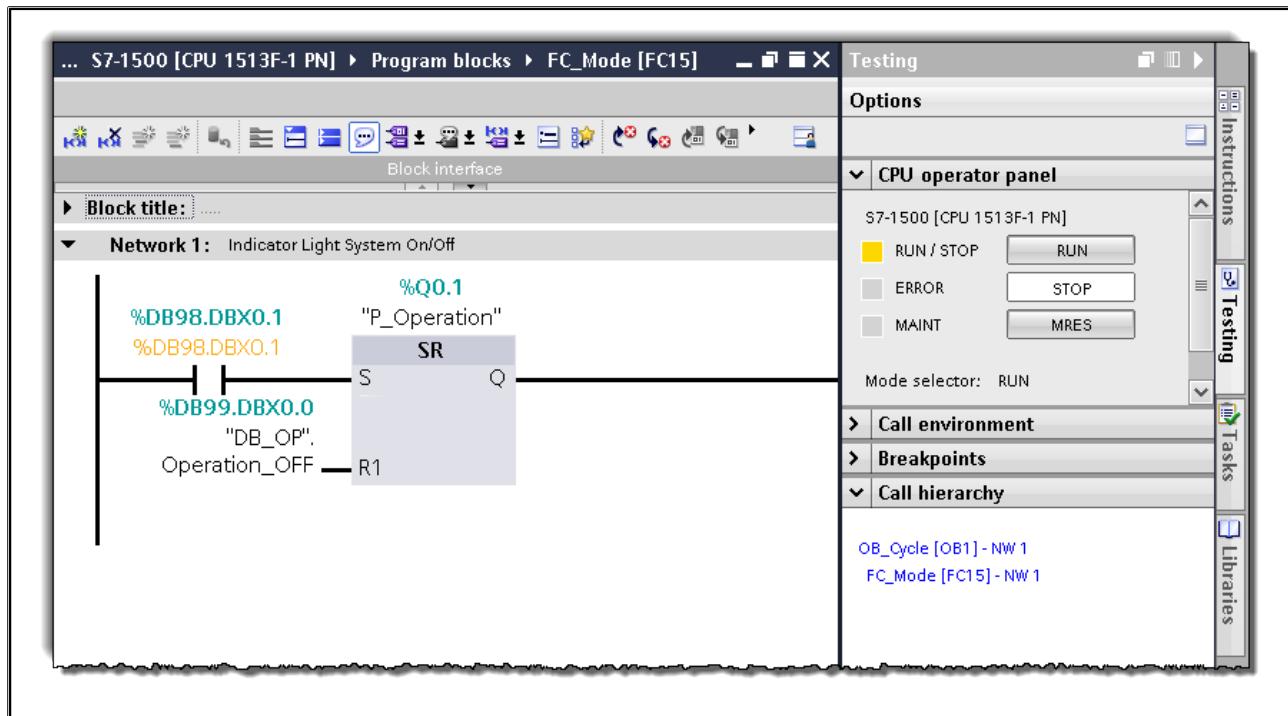


Opening a Block

For synchronous errors, that is, for errors that were triggered by a faulty instruction in the user program, you can open the block in which the interruption occurred by clicking on the "Open in editor" button.

If the SCL or STL language is selected, the cursor is positioned directly in front of the instruction that caused the interruption. In LAD/FBD, the network causing the interruption is highlighted. In the example shown, an attempt is made to access a data block which does not exist.

15.9. Call Hierarchy (Block Stack)



Call Hierarchy

The "Call hierarchy" gives you the information in which call path the block is opened.

If the block was opened from the diagnostic buffer via the button "Open in editor", then by looking at the entry in the Call hierarchy, you can see in which path the error occurred.

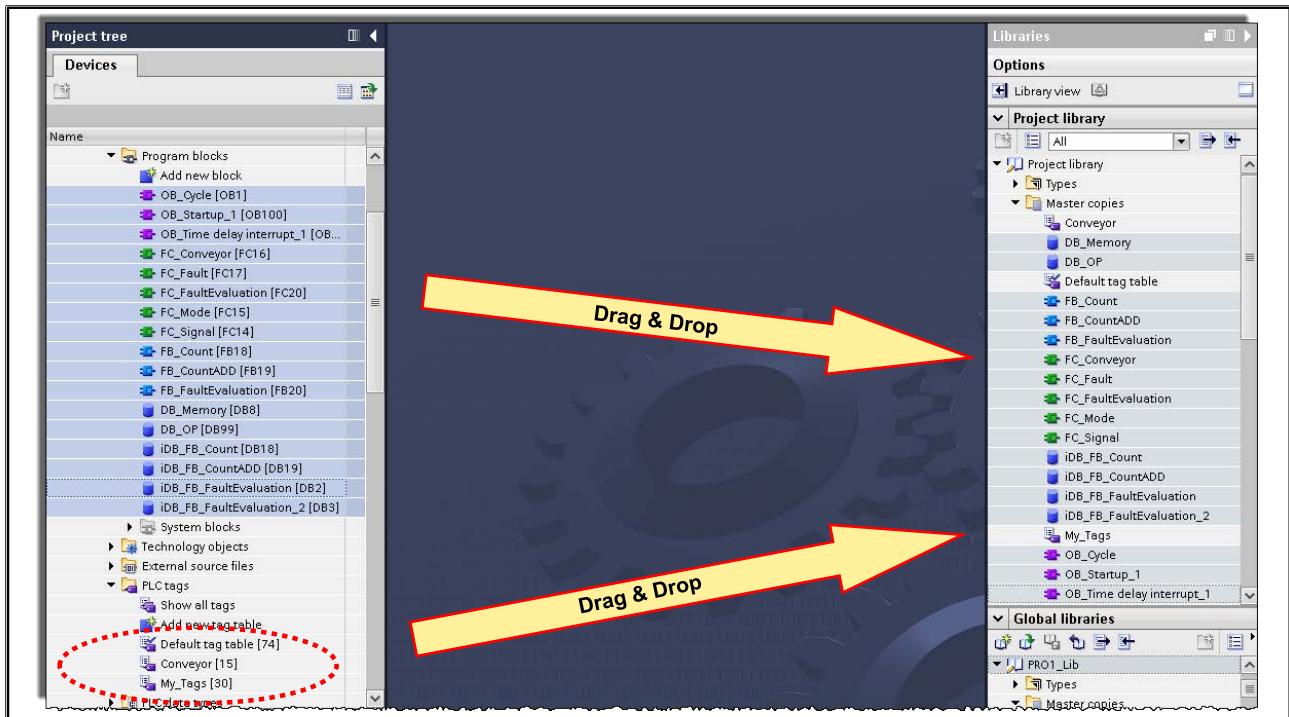
You can open the calling block by clicking on the relevant (appropriate) Link.

Note:

If the CPU is in "STOP" mode, the current block stack at the time of the STOP transition can be read out via the call hierarchy. The block stack lists all blocks whose execution was not completed at the moment when the CPU went into "STOP". The blocks are listed in the order in which the execution was started.

With an existing online connection, open any block for this and switch into the Task Card "Test" > Call hierarchy.

15.9.1. Exercise 1: Creating a Program Backup Copy in the Project Library



Task

You are to make a backup copy of your own program and the associated tags in the Project library of your project "MyProject2" since you will be working with a prepared faulty program afterwards.

What to Do

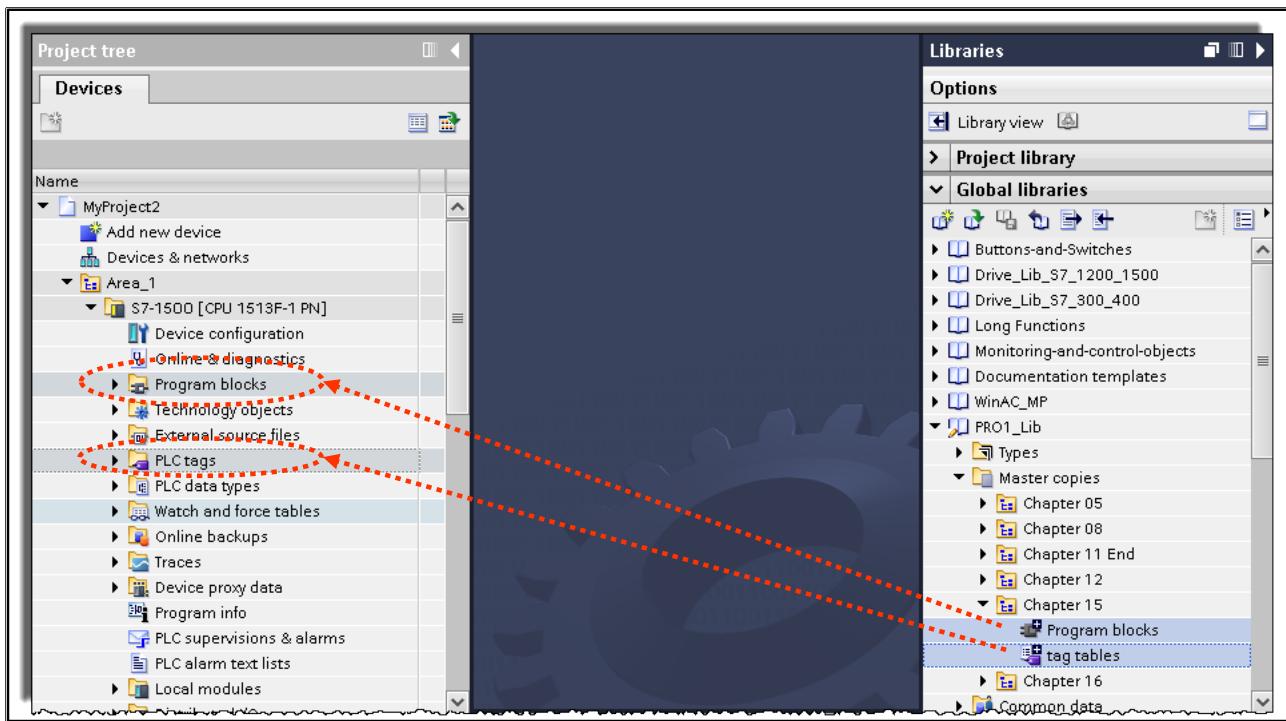
1. In the "Libraries", open the Project library.
2. Select all S7 blocks of the CPU program and copy them into your Project library using drag & drop.
3. Select all tag tables and copy them into your Project library using drag & drop.
4. Save your project and with that also the Project library.

Note:

If you would like to copy several objects combined as one object into the library, then highlight the objects, select "Copy" in the Context menu or press CTRL+C. Then, in the library, in the Context menu of the Master copies folder, you will find the command "Paste as a single master copy".

Combined objects can also only be copied together into a project/device.

15.9.2. Exercise 2: Copying the Faulty Program



Task

You are to copy a prepared program from the "PRO1_Lib" Global library into your own project.

What to Do

1. In your project, delete all S7 blocks and all PLC tag tables.
2. In the task card Libraries, open the "Global library" PRO1_Lib
<Drive>:\02_Archives\TIA_Portal\TIA-PRO1\PRO1_Lib
3. Using drag & drop, copy the blocks "Program blocks" and the PLC tag tables "tag tables" into your project (see picture).
4. Download all blocks of the faulty program into the CPU by selecting the CPU in the Project tree and then clicking on the Download button.
5. Carry out a CPU restart.
6. Save your project.

15.9.3. Exercise 3: STOP Troubleshooting Worksheet

Task

The faulty program contains one STOP error which is now to be corrected by you. If the CPU remains in RUN after error correction and subsequent restart, the exercise has been successfully completed.

In addition to the error (STOP error) detected by the system, the program also contains logical errors (RUN errors) so that the correct functioning of the program is still not established even after the STOP error is eliminated. The logical errors will be eliminated in the next exercises.

What to Do

Please note that after every STOP error correction, a CPU restart must be carried out. If the CPU once again goes into the STOP mode after the restart, a further STOP error exists or/that is, the error is not eliminated.

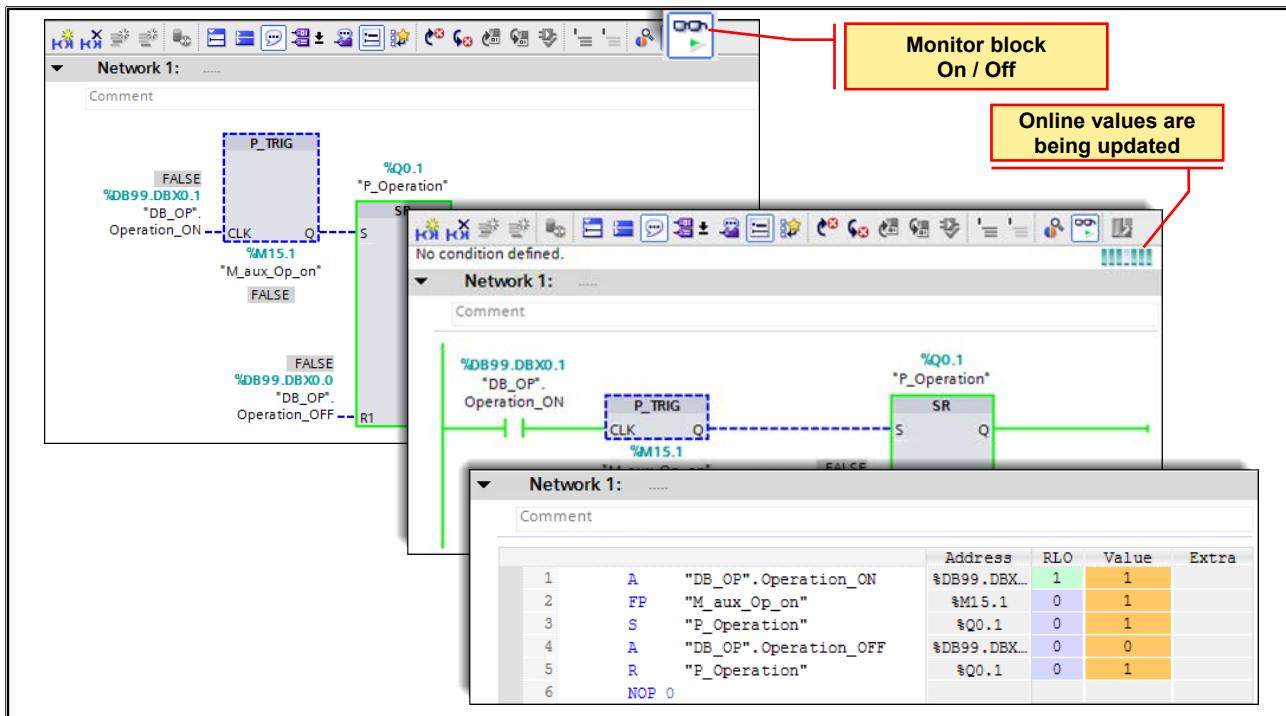
During error correction, answer the following questions on the error that occurs:

- STOP error determined:



- Interrupted block:
- Error
– Correction: (old instruction -> new instruction)
–

15.10. Monitor Block (Block Status)



Area of Use

The Monitor Block test function is used to be able to follow the program execution within a block. For this, the states or contents of the operands used in the block at the time of program execution are displayed on the screen. You can activate (switch on) the "Monitor" ("Block Status") test mode for the block which is currently open in the LAD/STL/FBD Editor by clicking the Glasses icon.

At the beginning of the test function, it is insignificant whether the block to be monitored is opened online or offline in the Editor. Should, however, the block opened offline not match the block saved online in the CPU, you first either have to open the block saved online or load the block opened offline into the CPU and then monitor it.

In the test mode, the states of the operands and LAD / FBD elements are displayed in different colors. You define these by selecting the menu option Options → Settings:

Examples:

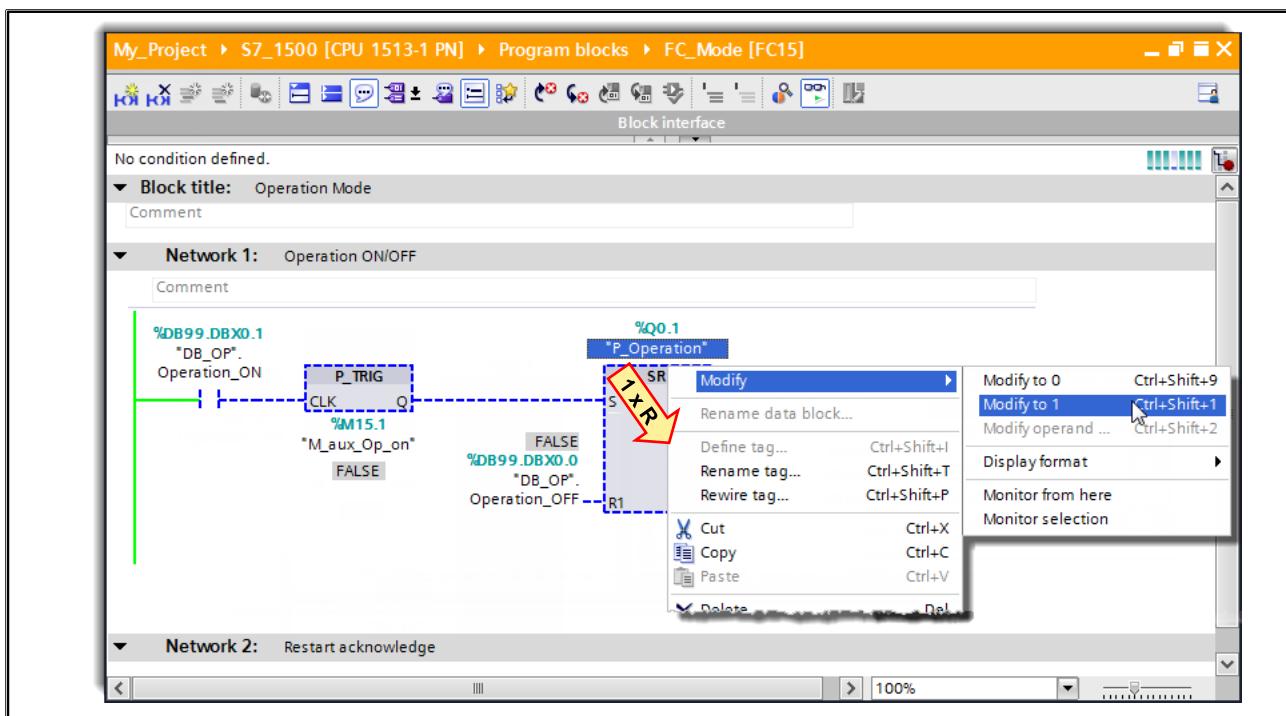
- Status fulfilled → "Element is displayed in green"
- Status not fulfilled → "Element is displayed in blue"

Notes

The monitoring values are only active when the CPU is in RUN mode and the instructions to be monitored are being processed!

This is indicated by the progress bar "Online values are being updated" in the upper right corner of the block.

15.10.1. Monitor Block: Modify Tags

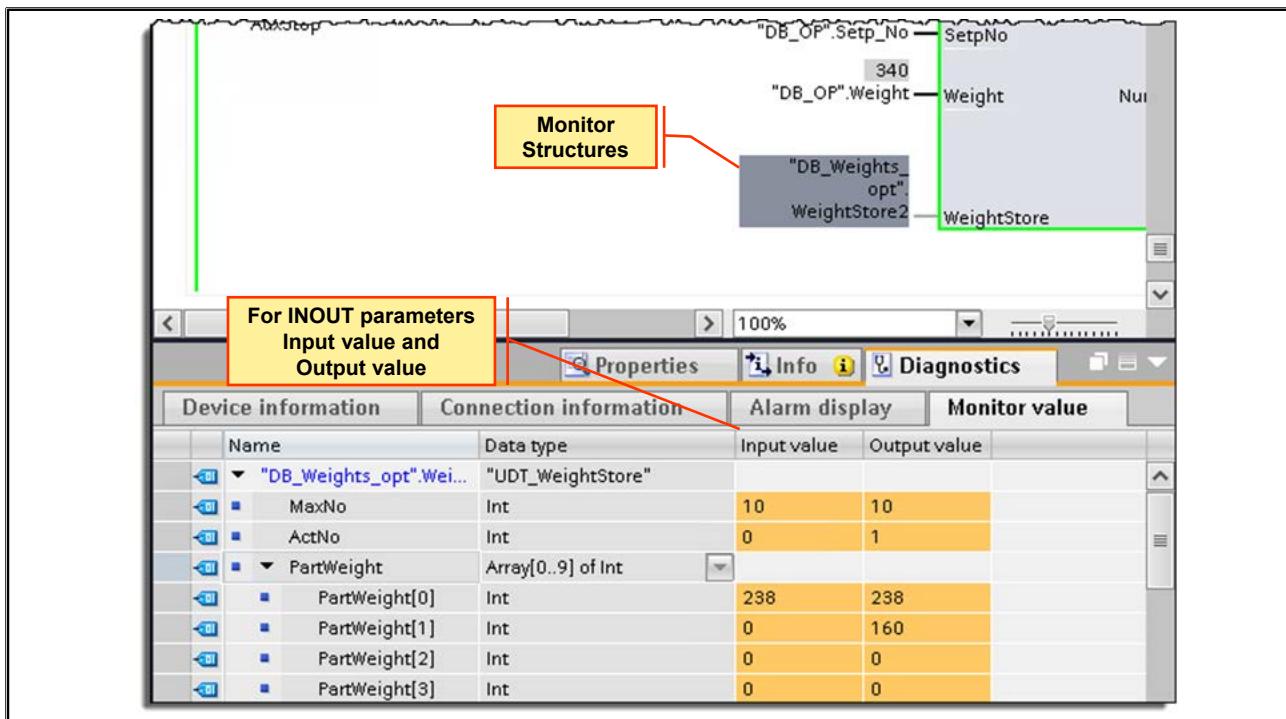


Modify Tags

When the "Monitor block" test function is activated, it is possible to modify tags to status '0' or '1'. The assignment of the status occurs once. When you use tags that are not Boolean, you can modify via the menu item "Modify operand...".

If the tag, whose status was changed, is not overwritten by the program, the tag remains at the assigned status. If, for example, an output is modified to status '1' and this tag is not overwritten by the program, the output remains switched on or to status '1'.

15.10.2. Monitoring Structures



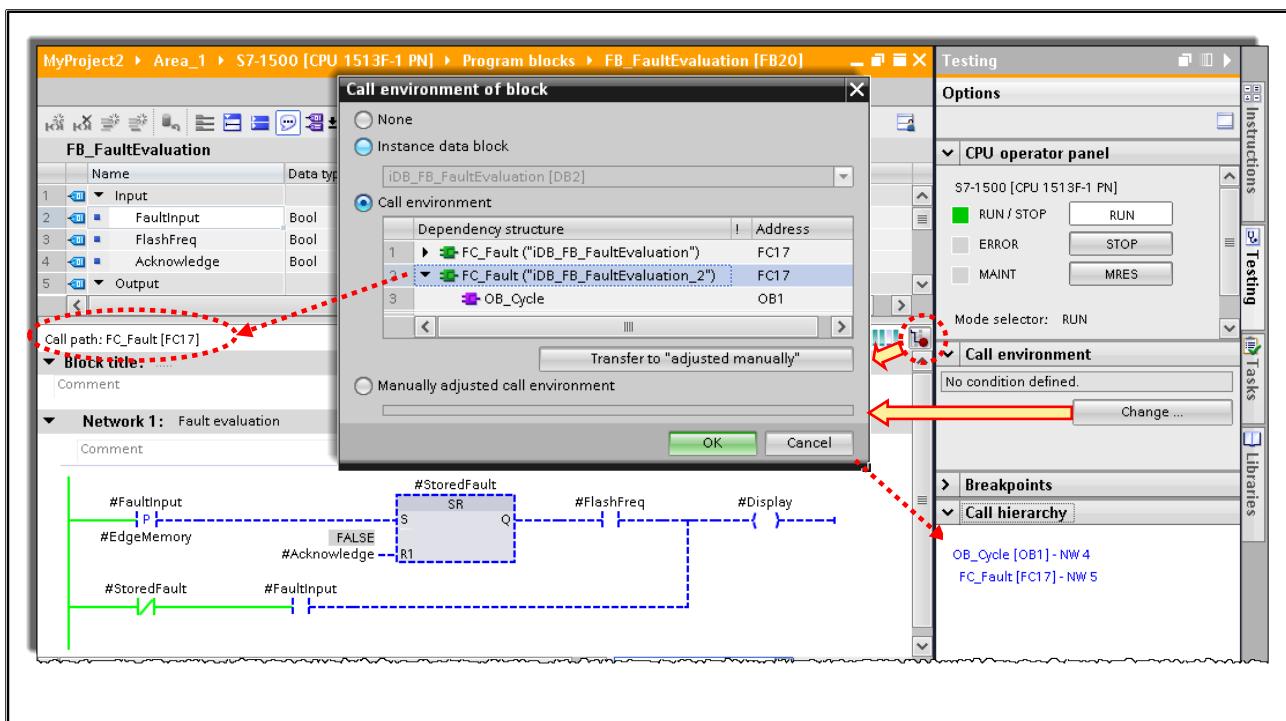
Rules for Monitoring Structures (S7-1200/1500)

When monitoring structures, the values of a structured PLC tag are displayed in the Inspector window → Diagnostics → Monitor value, with the following exception:

- Structures, whose elements have adjustable retentive properties, cannot be monitored.

In order to display the Monitor values for a structure, you must first of all activate the monitoring via the Context menu.

15.10.3. Monitor Block: Trigger Conditions / Call Environment

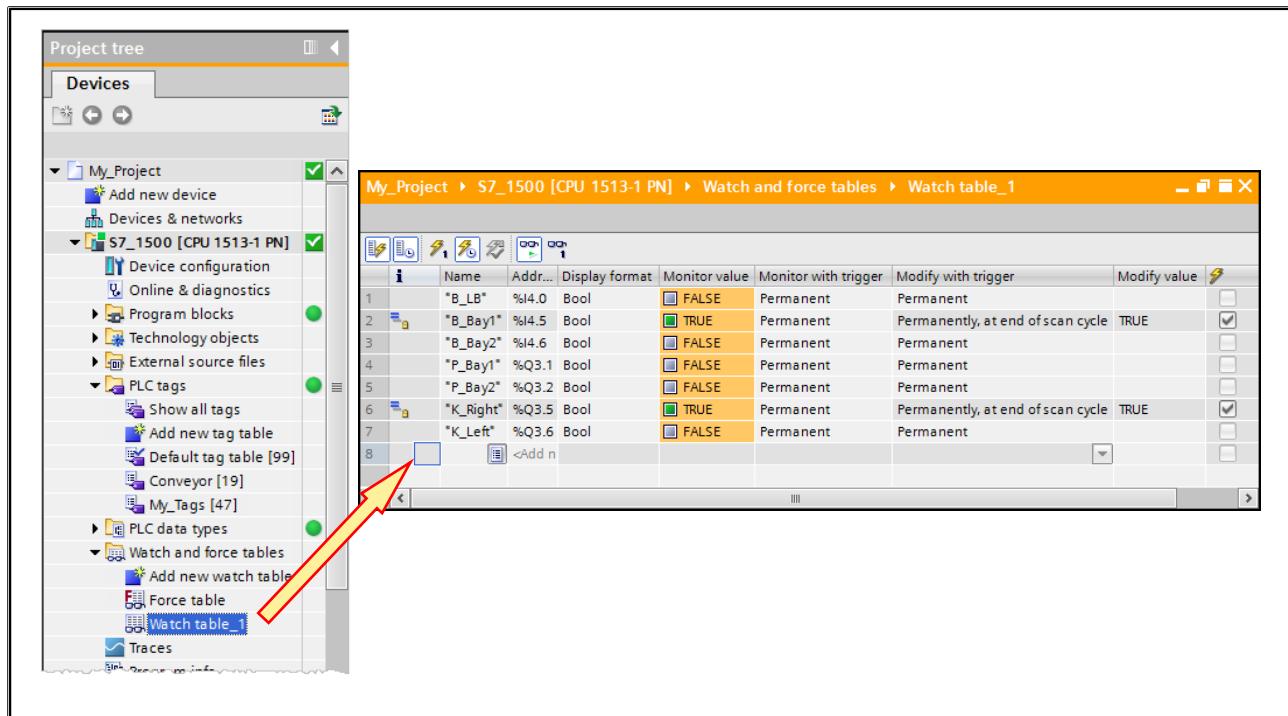


Function

The call conditions for blocks and for breakpoints can be defined. Thereby it is determined under which conditions the program status of a block is displayed or the program execution is interrupted at a breakpoint. The following conditions can be selected:

- Instance data block
The program status of a function block is only displayed when the function block is called with the selected instance data block.
- Call path
The program status of a block is only displayed when the block is called by a specific block or from a specific path.

15.11. Monitor / Modify Variables (Tags): Watch Tables



Area of Use

The "Monitor/Modify Variables (Tags)" test function is used to monitor and / or modify variables (tags) in any format you choose. For this, the desired variables are entered in a watch table. With the exception of block-local, temporary variables, you can monitor and/or modify all variables (tags) or operands.

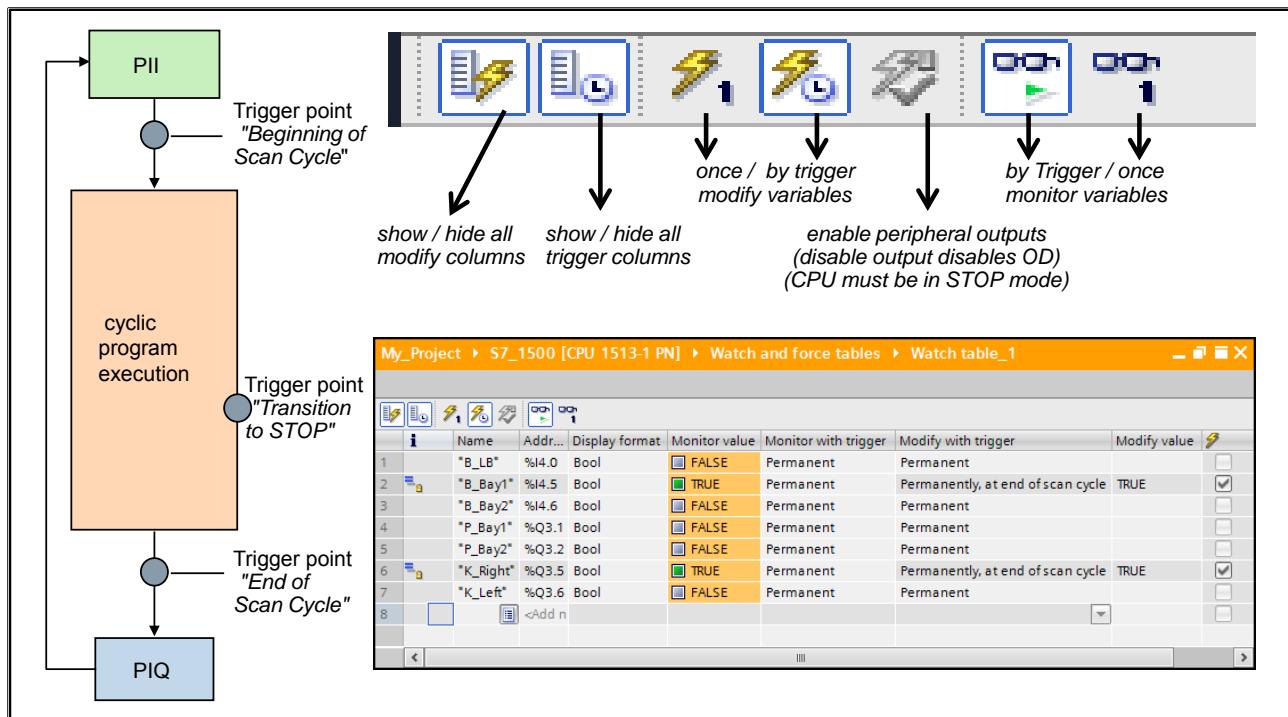
You can choose the columns displayed in the Watch table via the menu 'View'. The columns have the following meanings:

- Name: symbolic name of the variable (tag)
- Address: absolute address of the variable (tag).
- Symbol comment: comment on the variable (tag) displayed
- Display format: a data format you can choose per mouse click (such as binary or decimal), in which the contents of the variable (tag) is displayed
- Monitor value: variable (tag) value in the selected status format
- Modify value: value to be assigned to the variable (tag)

Watch Table

You can choose any name for the Watch table. Saved Watch tables can be reused to monitor and modify so that a renewed input of the variables to be monitored is no longer necessary.

15.11.1. Monitor / Modify Variables (Tags): Trigger Points



Trigger Points

Through the "Monitor with trigger or Modify with trigger" columns, you can define the trigger points for monitoring and modifying. The "Trigger Point for Monitoring" specifies when the values of the variables being monitored are to be updated on the screen. The "Trigger Point for Modifying" specifies when the given modify values are to be assigned to the variables being modified.

Trigger Condition

In the "Monitor with trigger" column you can specify whether the values are to be updated on the screen once only when the trigger point is reached or permanently (when the trigger point is reached).

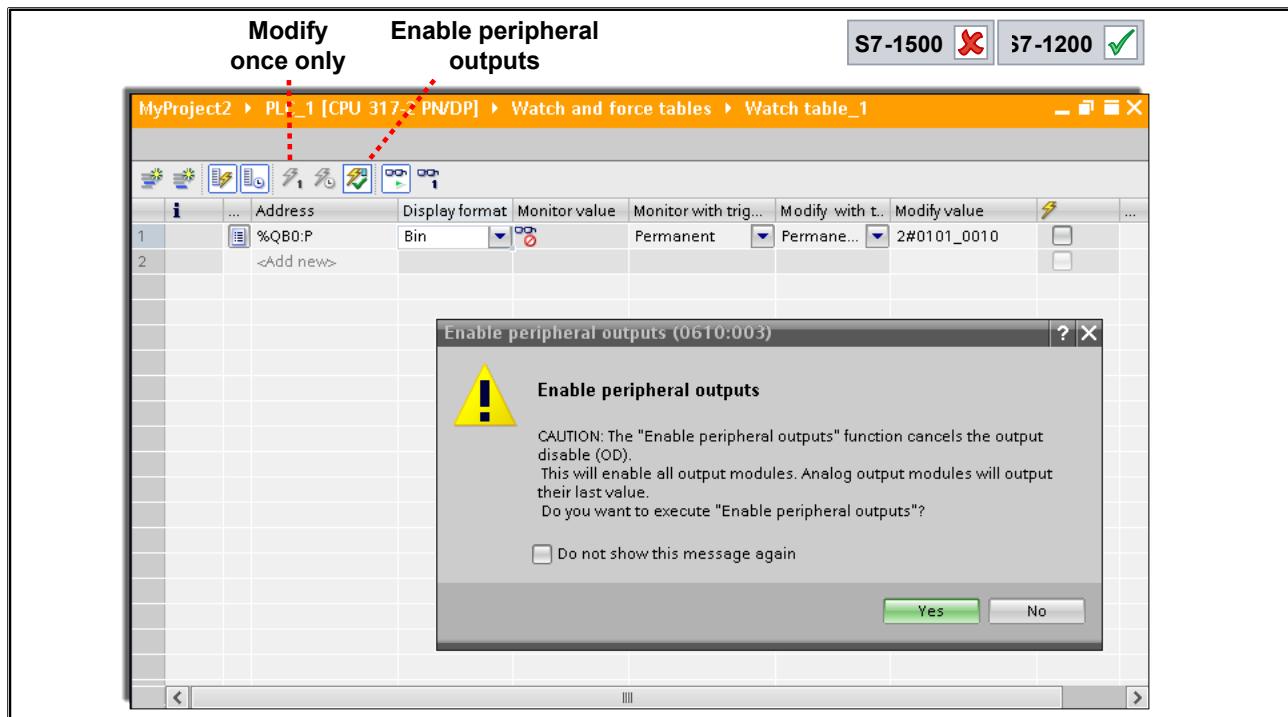
In the "Modify with trigger" column you can specify whether the given modify values are to be assigned to the variables being modified once only or permanently (every time the trigger point is reached).

Area of Use

The following tests, among others, can be implemented with the appropriate selection of trigger points and conditions:

- Wiring test of the inputs:
Monitor variables, Trigger point: Start of scan cycle, Trigger condition: Permanent
- Simulate input states (user specified, independent of process):
Modify variables, Trigger point: Start of scan cycle, Trigger condition: Permanent
- Differentiation between hardware / software errors (an actuator that should be activated in the process is not controlled)
Monitor variables, in order to monitor the relevant output, Trigger point: End of scan cycle, Trigger condition: Permanent
(output state = '1' > program logic OK > process error (hardware)
(output state = '0' > program logic error (such as double assignment))
- Control Outputs (independent of the program logic)
Modify variables, Trigger point: End of scan cycle, Trigger condition: Permanent

15.11.2. Enable Peripheral Outputs (in planning for S7-1500)



The Function "Enable Peripheral Outputs" (S71500 in Planning)

The "Enable Peripheral Outputs" function is used to check the functioning of the output modules, the wiring of the digital output modules or it can be used to continue to control actuators in the process even though the CPU finds itself in the STOP state because of an error that has occurred.

The "Enable Peripheral Outputs" function cancels the output disable of the peripheral outputs (PQ), which enables you to control the outputs in spite of the CPU's STOP state.

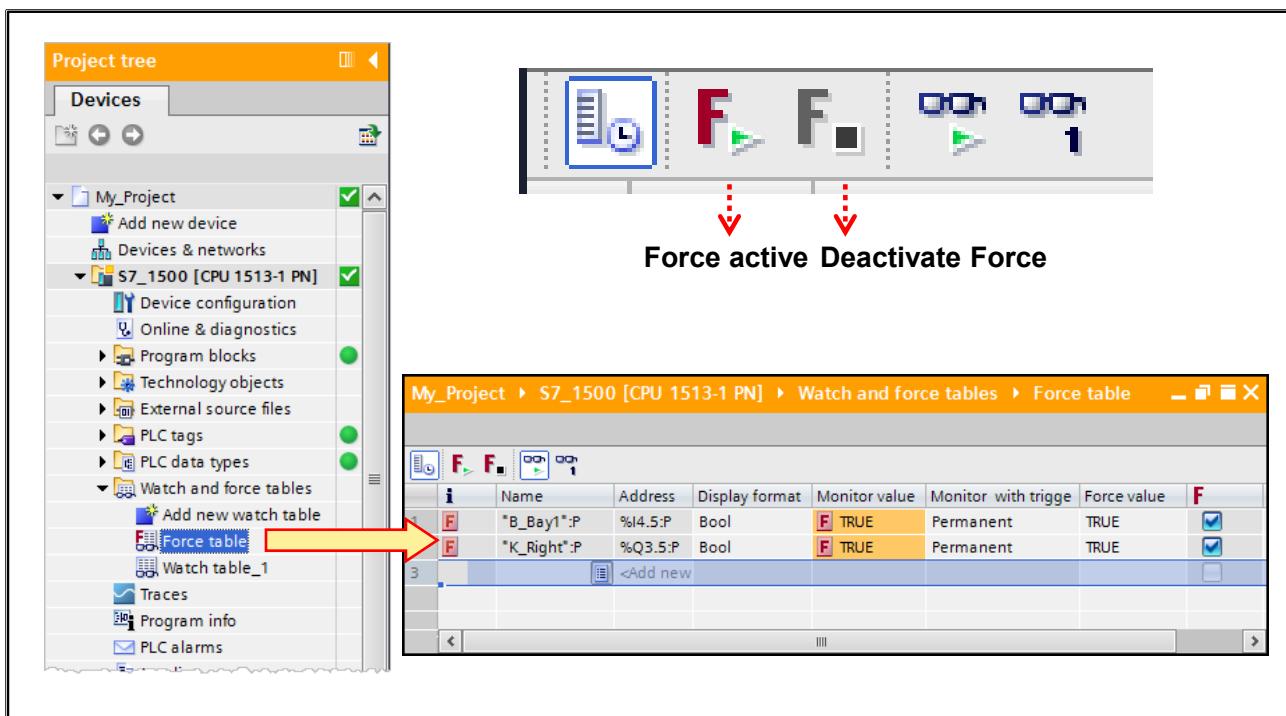
Conditions

- The CPU must be in STOP mode
- A Force task must not be active in the CPU
- The Watch table must be displayed in "extended mode", in other words, displayed with trigger columns
- The peripheral outputs to be enabled are to be specified byte by byte, word by word or double-word by double-word with the suffix :P (for peripheral)
- After the peripheral outputs have been enabled, the modify values can be activated via the "Modify once only" button (not via "Modify with trigger").

Note

When changing the CPU's operating status from STOP to RUN or STARTUP, Enable Peripheral Outputs is deactivated and a message pops up.

15.11.3. Force Variables (Tags)



Function and Area of Use

With Force, you can overwrite variables (tags) with any values you like, independent of the user program. Only one "Force Values" window can be open at a time for a CPU.

With the S7-300, you can only force the inputs and outputs in the process image; with the S7-400 you can also force memory bits and peripherals.

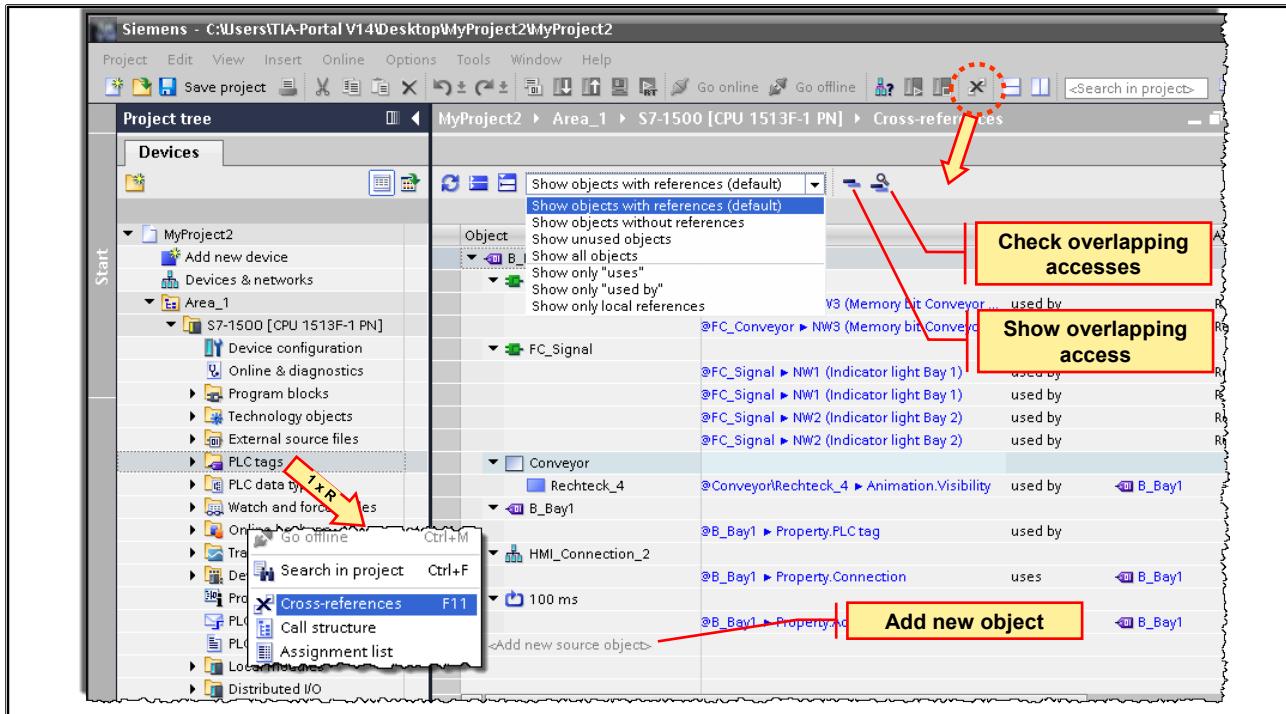
With the S7-1500/1200, the peripherals of the inputs and outputs can be forced.

Notes on Forcing

Before you start the "Force" function, you should make sure that no one else is carrying out this function at the same time on the same CPU.

A force task can only be canceled through an explicit "stop forcing" [deactivating the icon] (not via **Edit → Undo!**). Closing the Force table or exiting the application does not cancel the force task on the CPU.

15.12. Reference Data: Cross-references of Tags



Introduction

The cross-references list offers an overview of the use of operands and variables (tags) within the user program. From the cross-references list, you can jump directly to the point of use.

The cross-references list contains the following information:

- Which operand is used in which block with which instruction,
- Which tag is used in which HMI screen,
- Which block is called by which other block

As part of the project documentation, the cross-references supply a comprehensive overview of all operands, memory areas, blocks, variables (tags) and screens used.

Views

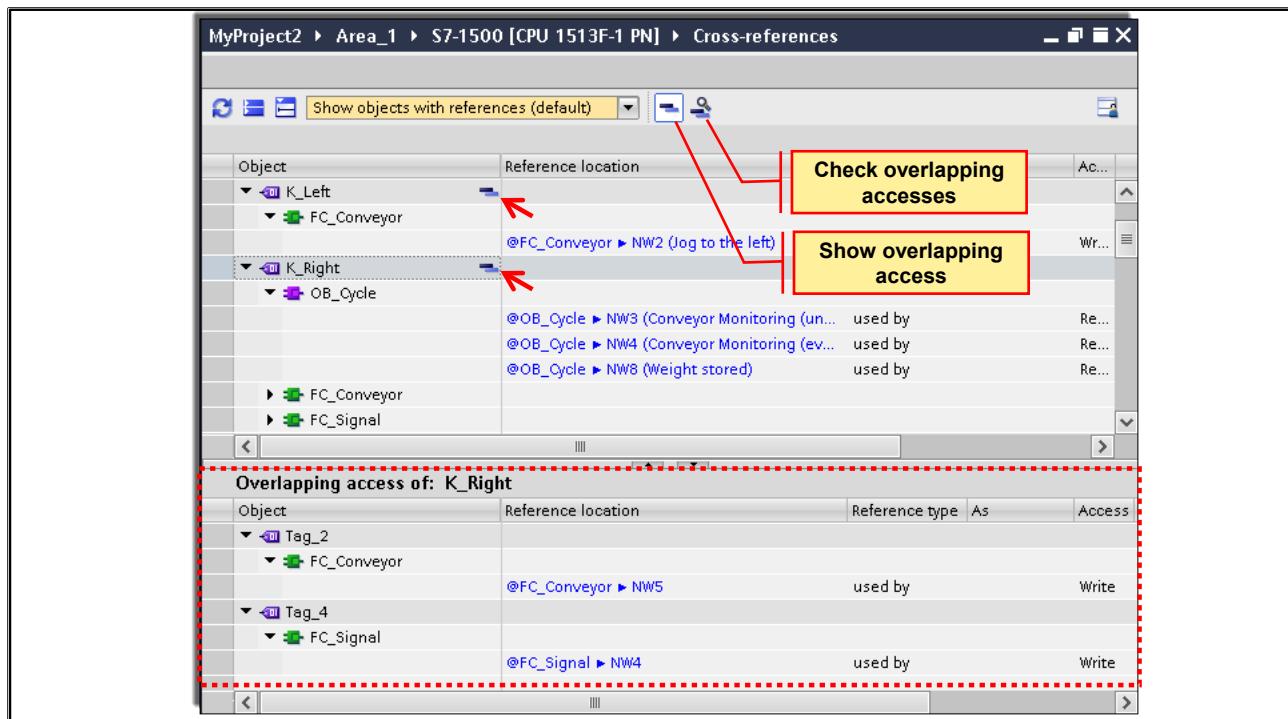
There are two views of the cross-references list which differentiate themselves by which objects are displayed in the first column:

- Used by:
Displays the referenced objects
Here, the Reference location where the object is used are displayed.
- Used:
Displays the referencing objects.
Here, the users of the object are displayed.
The associated tooltips give further information on the respective objects.

Show Unused

This is a list of tags which are declared in the PLC tag table but are not used in the S7 user program.

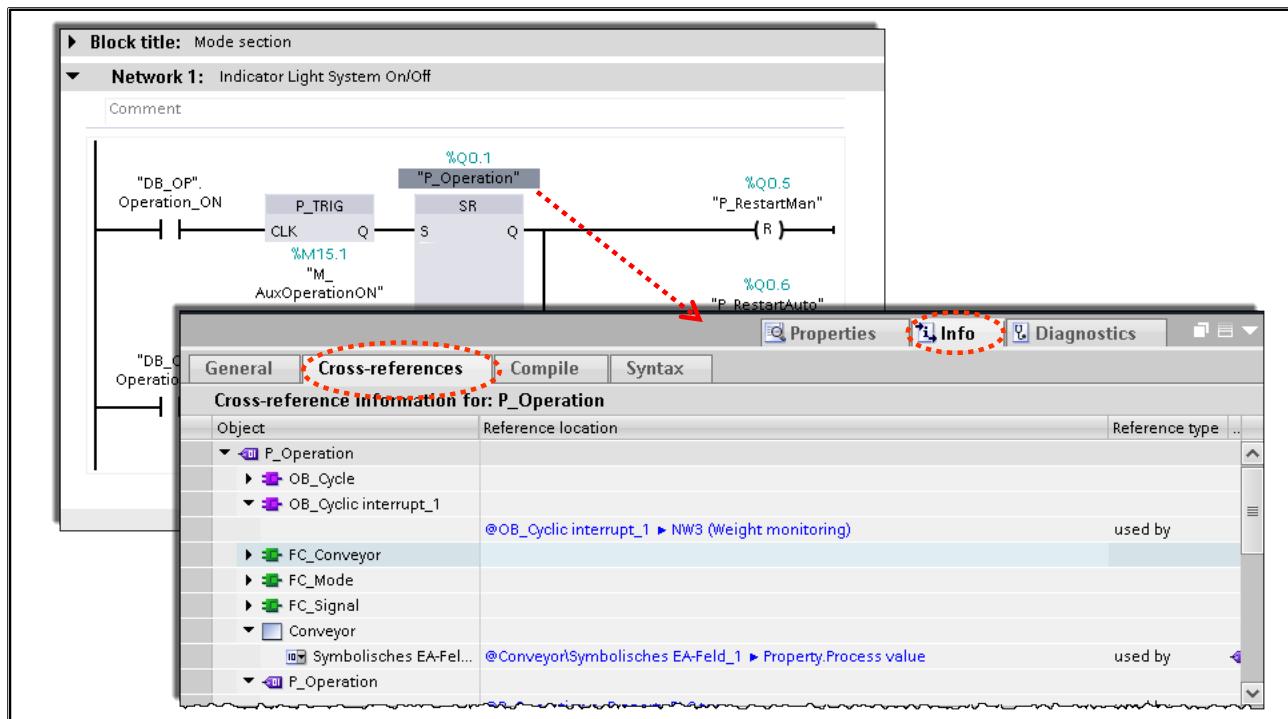
15.12.1. Reference Data: Cross-references / Show Overlapping Accesses



With the help of the "Check overlapping accesses" button, you can check whether overlapping accesses exist for one of the variables (tags).

If this is the case, they can be displayed in a separate table with the help of the "Show overlapping accesses" button.

15.12.2. Reference Data: Cross-references of a Variable (Tag) in the Block Editor



Introduction

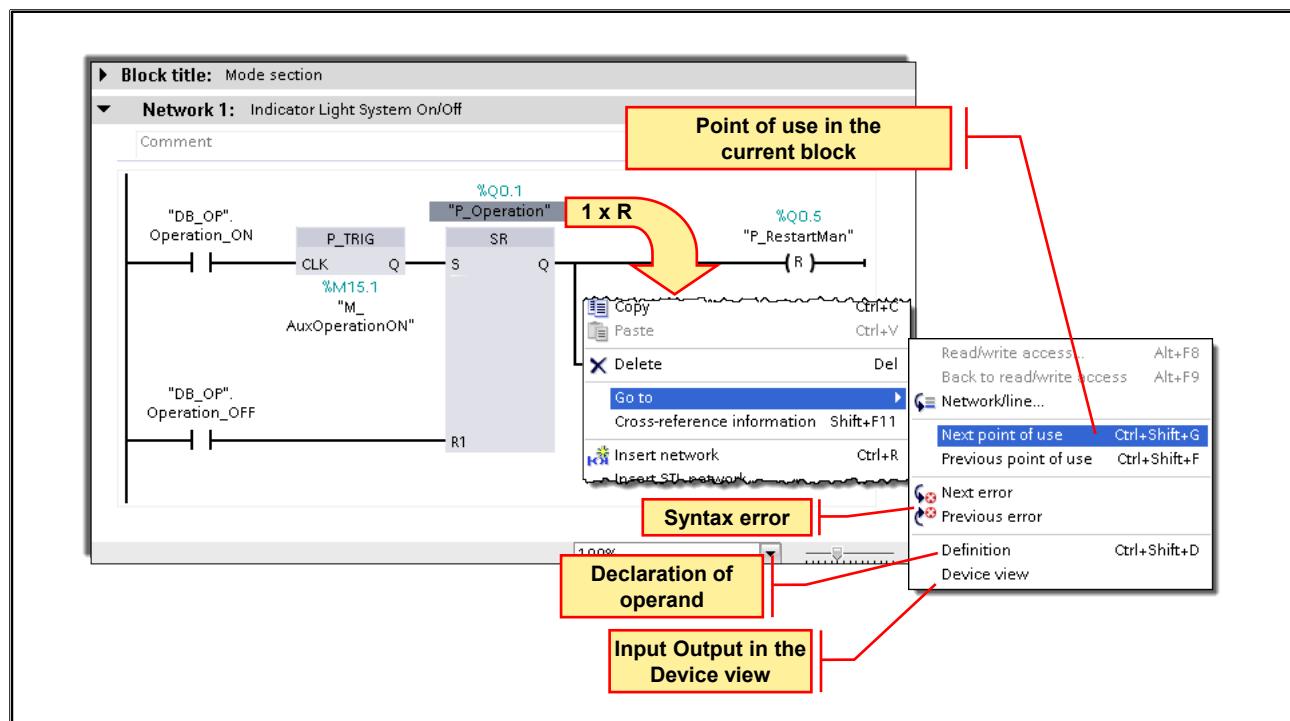
In the Inspector window, the cross-reference information for a selected object is displayed in the tabs "Info > Cross-references". In this tab, you will see at which Reference locations and from which other objects every selected object is used.

In the Inspector window, even those blocks that only exist online are displayed in the Cross-references.

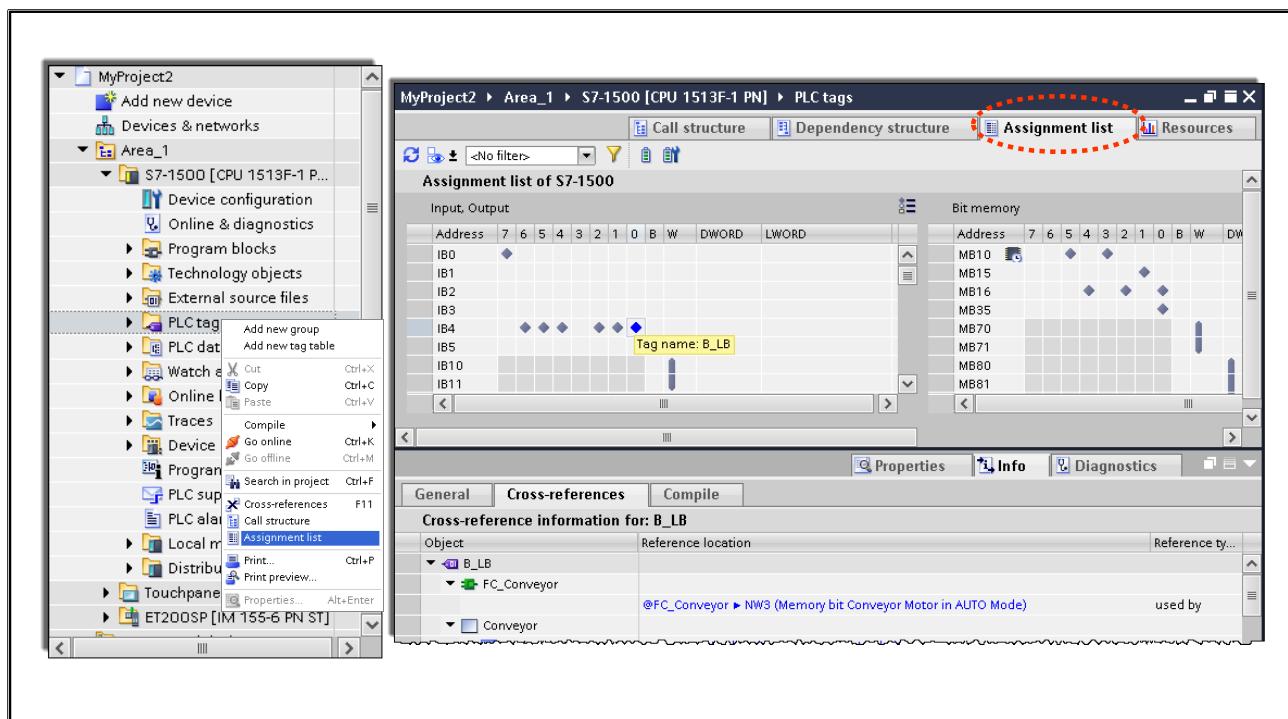
Structure

The cross-reference information is displayed in tabular form in the Inspector window. Each column contains specific detailed information on the selected object and its use.

15.12.3. Go To...



15.12.4. Reference data: Assignment I/Q/M/T/C



Assignment I/Q/M/T/C

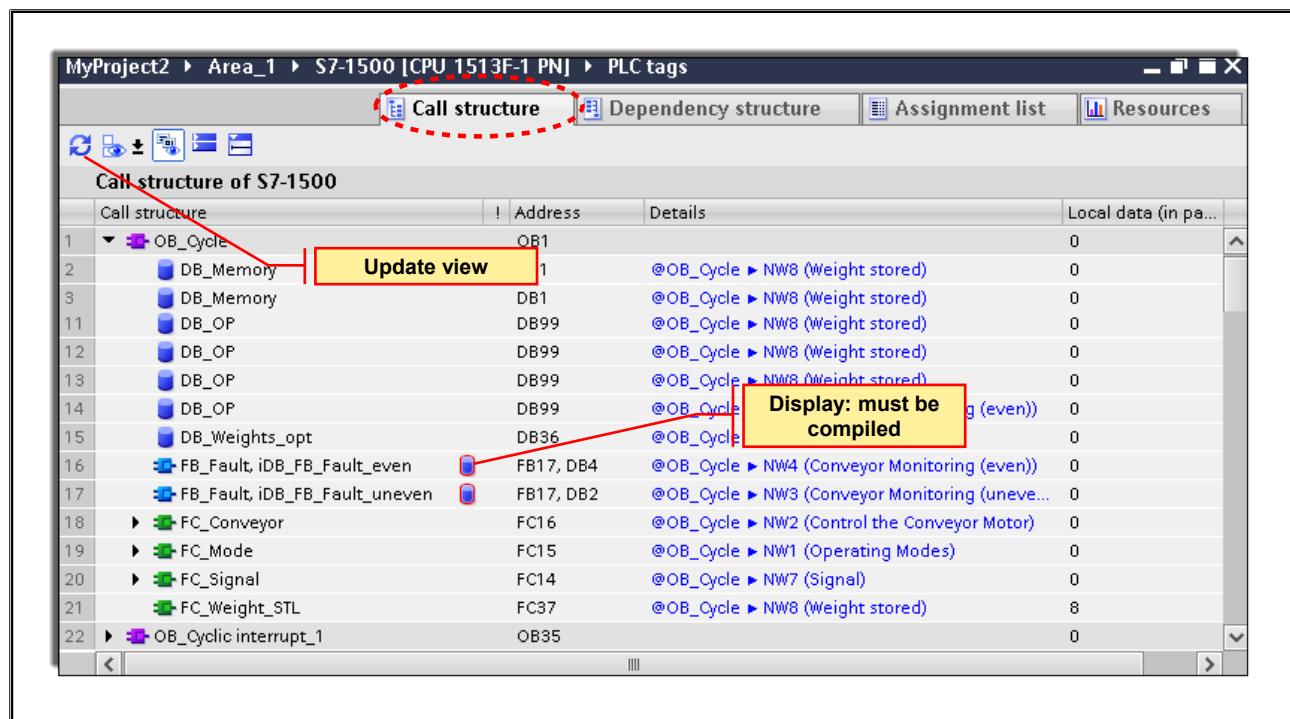
The assignment list for I/Q/M/T/C is opened via "Right-click on the relevant device -> Assignment list" or via the menu "Tools > Assignment list".

This assignment list gives you an overview of which bit is used from which byte of the memory areas input (I), output (Q) and memory bit (M) and which SIMATIC timers and counters are used. The type of use (reading or writing) is not displayed.

The memory areas inputs (I), outputs (Q) and memory bits (M) are displayed byte-by-byte in lines.

- The bits identified with a small diamond, that is, binary operands (in the picture, for example, I 4.0 or M 16.4) are used explicitly in the program.
- The fields of the individual bits which have a gray background identify byte, word, double-word or long word operands that are used in the user program. The operand dimension (byte, word, double-word or long word) comes from the vertical line in one of the columns "B" (Byte), "W" (Word), "DWORD" (Double word) and "LWORD" (Long word).
- Bits that are marked with both a diamond and a gray background field are used explicitly as a binary operand in the user program and are used via a byte, word, double-word or long word operand.

15.12.5. Reference Data: Call Structure



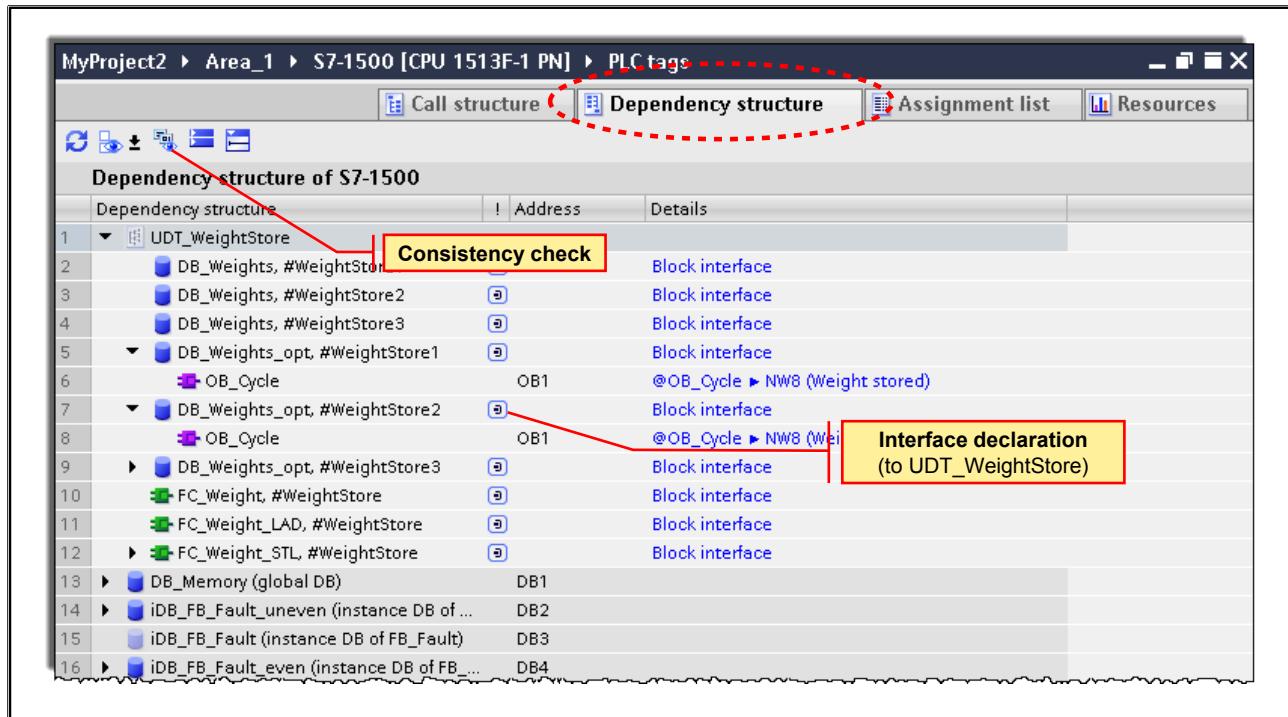
Call Structure

The call structure is opened via "Right-click on the relevant device -> Call structure" or via the menu "Tools > Call structure" and describes the call hierarchy of the blocks within an S7 program.

It gives an overview of:

- The blocks used
- Jumps to the points of use of the blocks
- Dependencies between the blocks
- Local data requirements of the blocks
- Status of the blocks

15.12.6. Reference Data: Dependency Structure



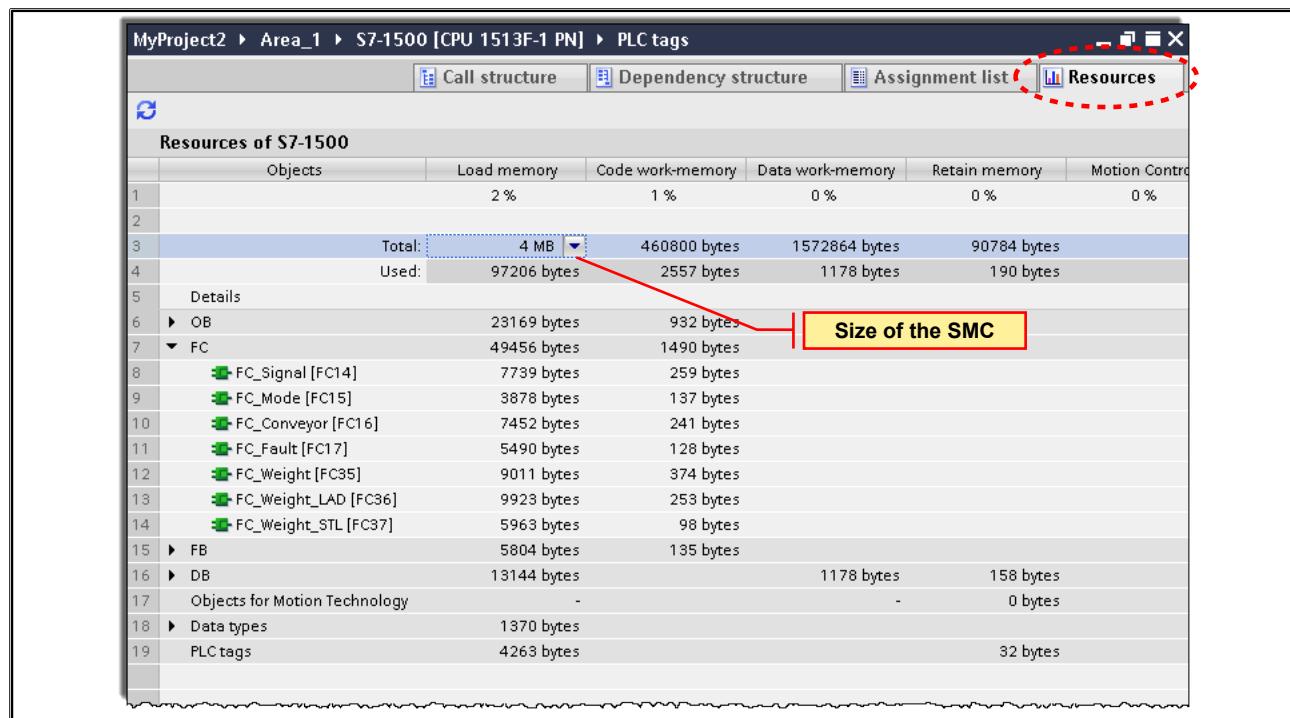
Dependency Structure

The display of the dependency structure is opened via the menu "Tools > Dependency structure" and with it you get a list of blocks used in the user program. In the first level (to the very left) is the respective block and indented underneath it are the blocks which call this block or use it.

The dependency structure also shows the status of the individual blocks through the use of symbols. Objects which cause a time stamp conflict and which can lead to an inconsistency in the program are identified with different symbols.

The dependency structure represents an extension of the cross-references list for objects.

15.13. Resources



The screenshot shows the 'Resources of S7-1500' window in the SIMATIC TIA Portal. The window has tabs at the top: Call structure, Dependency structure, Assignment list, and Resources (which is highlighted with a red dashed circle). Below the tabs is a table with columns: Objects, Load memory, Code work-memory, Data work-memory, Retain memory, and Motion Control. The table lists various objects and their memory usage. A red arrow points from the text 'Size of the SMC' to the 'Used' column for the OB row, which shows 97206 bytes.

Objects	Load memory	Code work-memory	Data work-memory	Retain memory	Motion Control
1	2 %	1 %	0 %	0 %	0 %
2					
3 Total:	4 MB	460800 bytes	1572864 bytes	90784 bytes	
4 Used:	97206 bytes	2557 bytes	1178 bytes	190 bytes	
5 Details					
6 ▶ OB	23169 bytes	932 bytes			
7 ▼ FC	49456 bytes	1490 bytes			
8 ▶ FC_Signal [FC14]	7739 bytes	259 bytes			
9 ▶ FC_Mode [FC15]	3878 bytes	137 bytes			
10 ▶ FC_Conveyor [FC16]	7452 bytes	241 bytes			
11 ▶ FC_Fault [FC17]	5490 bytes	128 bytes			
12 ▶ FC_Weight [FC35]	9011 bytes	374 bytes			
13 ▶ FC_Weight_LAD [FC36]	9923 bytes	253 bytes			
14 ▶ FC_Weight_STL [FC37]	5963 bytes	98 bytes			
15 ▶ FB	5804 bytes	135 bytes			
16 ▶ DB	13144 bytes		1178 bytes	158 bytes	
17 Objects for Motion Technology	-		-	0 bytes	
18 ▶ Data types	1370 bytes				
19 PLC tags	4263 bytes			32 bytes	

The Resources is opened via the menu "Tools > Resources" and shows you which (how much) memory area is used by which objects in the CPU.

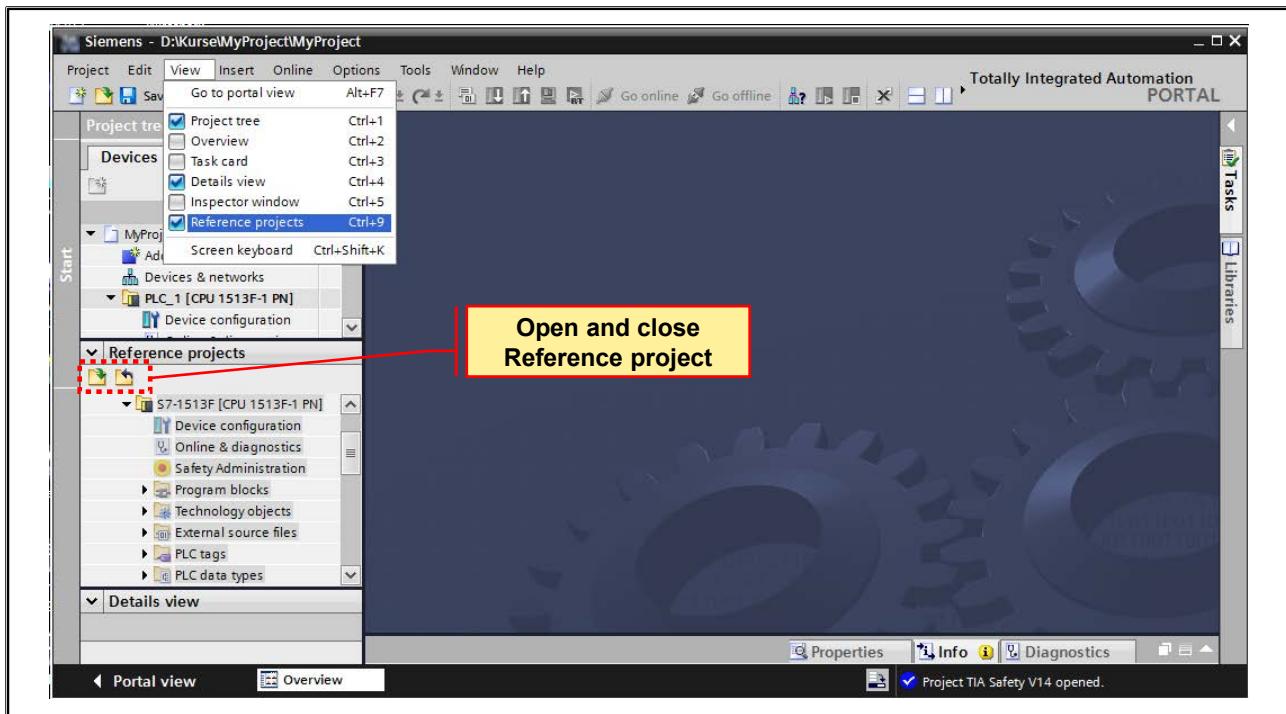
Note

Display of the 'Used' Load memory in the CPU

Please note that the sum of the used load memory cannot be exactly determined if not all blocks have been compiled.

In this case, a ">" placed in front of the sum indicates that the value for the used memory area could be larger than displayed since blocks that are not compiled are not taken into account for the total formation.

15.14. Reference Projects

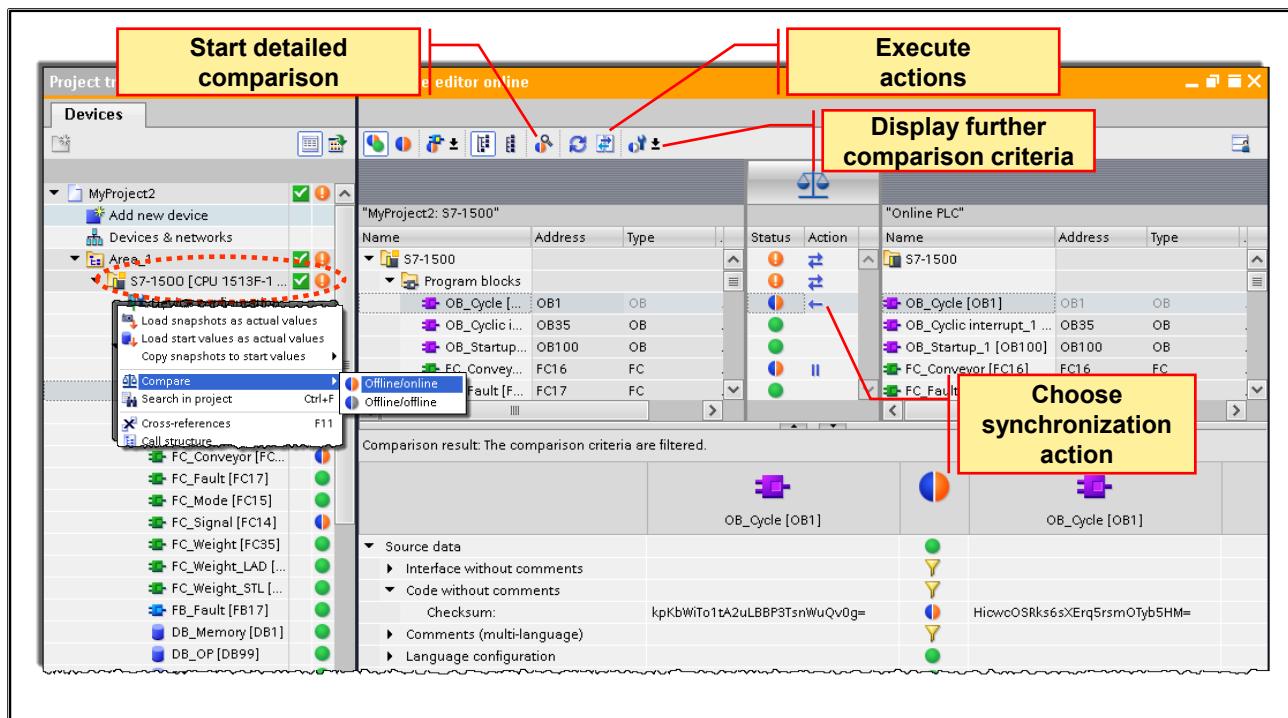


Reference Projects

In addition to the user project, a write-protected Reference Project (see picture, all project elements have a grey background) can be opened via the menu "View". Several elements (for example, Program blocks) can be opened in the editor but they cannot be changed.

Similar to the Libraries, all project elements can be copied from the Reference Project into the user project.

15.15. Compare (1) - Offline / Online



Types of Comparison

In principle, there are two different types of comparison:

Online/Offline comparison:

- The objects in the project are compared with the objects of the relevant device. For this, an online connection to the device is necessary.

Offline/Offline comparison:

- Either the objects of two devices within a project or from different projects are compared.

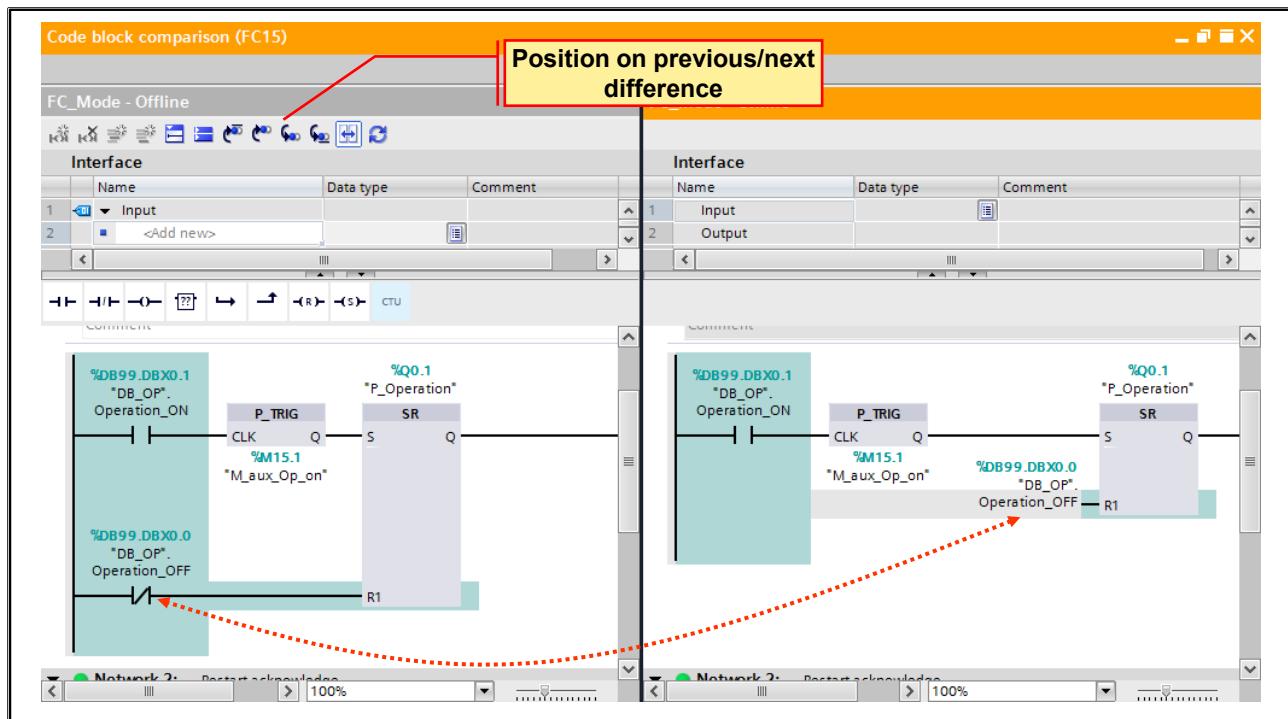
Symbols of the Result Display

The result of the comparison is presented by means of symbols.

The following table shows the symbols for the comparison results of an Online/Offline comparison:

Symbol	Meaning
!	Folder contains objects whose online and offline versions are different
?	Comparison result is unknown
green circle	Online and offline versions of the object are identical
blue/red circle	Online and offline versions of the object are different
blue circle	Object only exists offline
red circle	Object only exists online

15.15.1. Compare (2) – Block Detailed Comparison

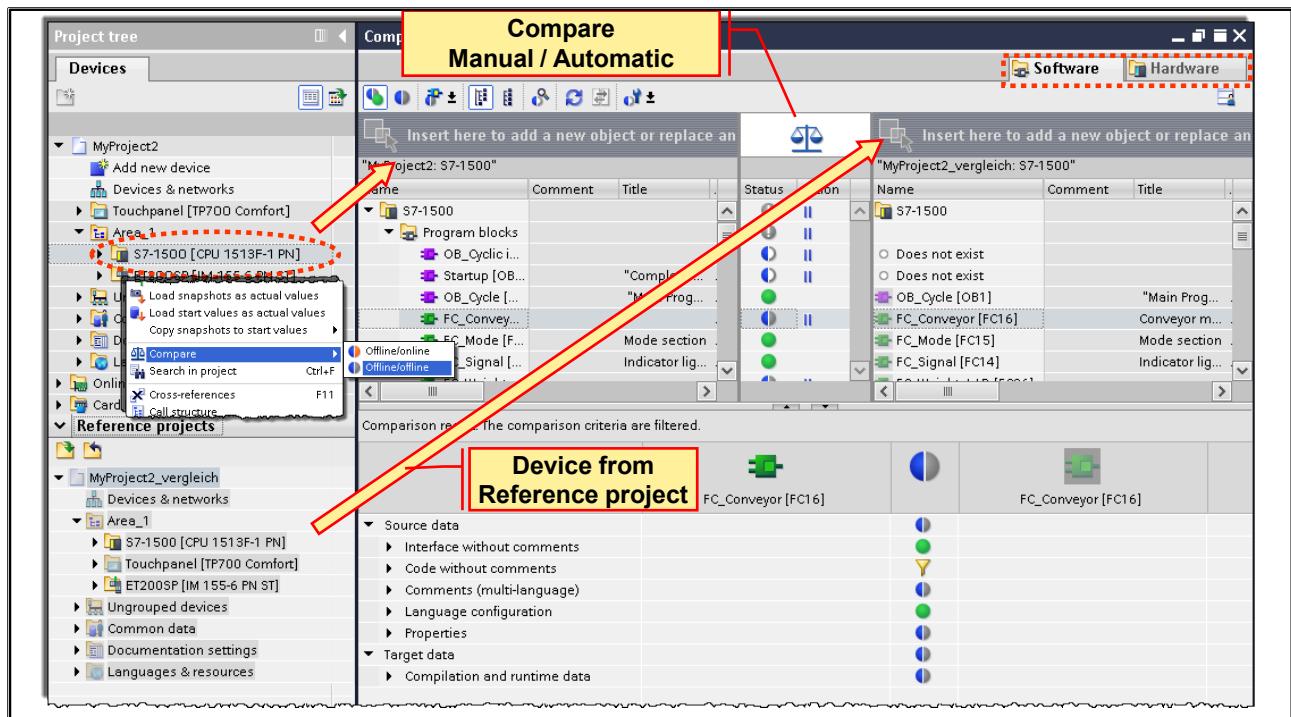


Detailed Comparison

Through the detailed comparison you can identify exactly those locations that are different in the online and offline version of a block. The following identifiers are used:

- Lines in which there are differences are highlighted in grey.
- Different operands and operations are highlighted in green.
- When the number of networks is different, pseudo networks are inserted so that a synchronized representation of identical networks is possible. These pseudo networks are highlighted in grey and contain the text "No corresponding network was found" in the title-bar of the network. Pseudo networks cannot be processed.
- If the sequence of the networks is mixed up, pseudo networks are inserted at the appropriate locations. These pseudo networks are highlighted in grey and contain the text "The networks are not synchronized" in the title-bar of the network. The pseudo network also contains a link "Go to network <No>", through which you can navigate to the associated network.

15.15.2. Compare (3) - Software Offline / Offline



Offline / Offline Software Comparison:

- Objects of two devices within a project,
- Blocks of two devices within a project,
- Blocks in one device,
- Objects from different projects,
- Blocks from different projects,

For this, you can switch between

automatic and manual comparison using a mouse click.

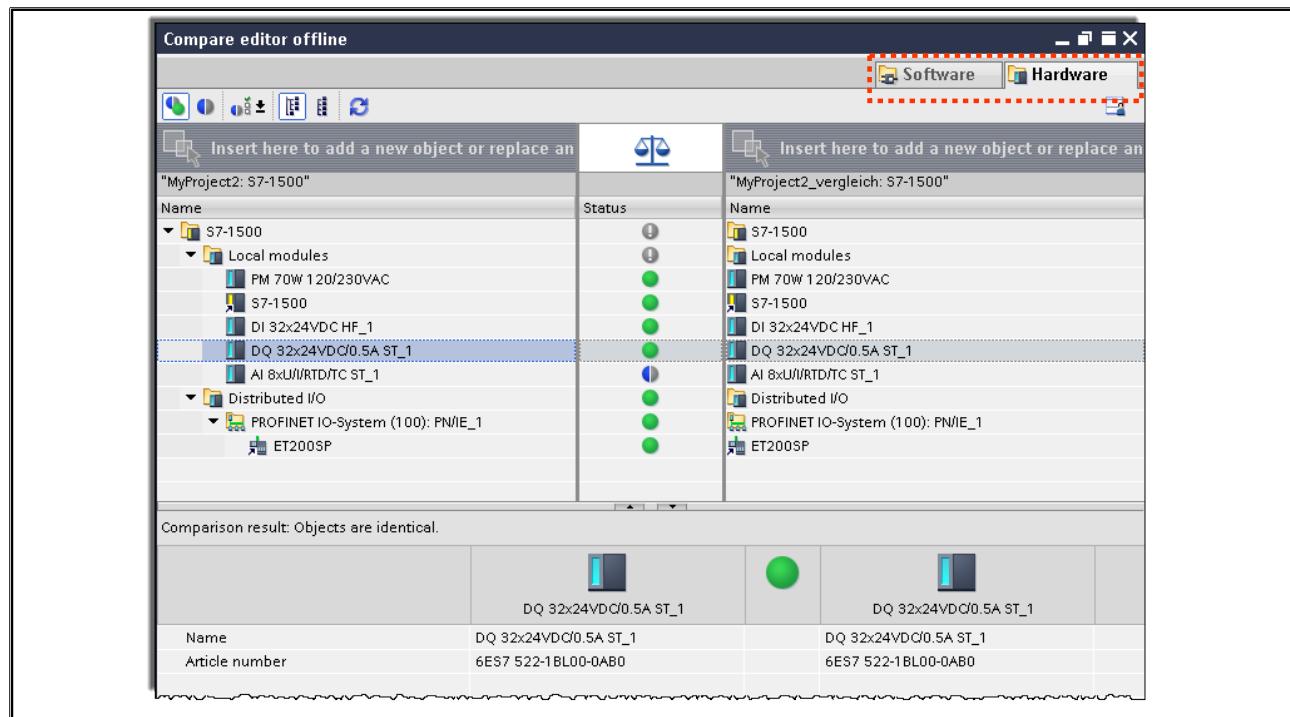
Automatic Comparison:

Blocks and objects of the same type and the same name are compared

Manual Comparison:

You can select which blocks are to be compared. That way, it is possible to compare all blocks.

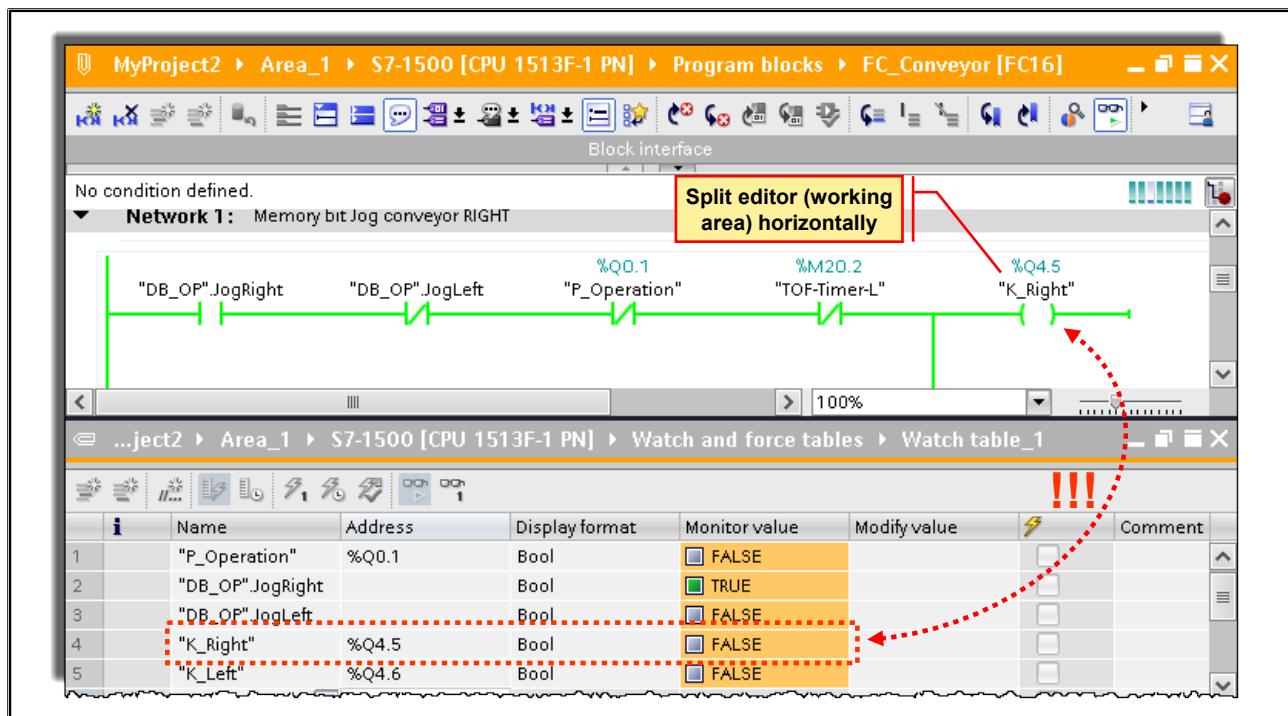
15.15.3. Compare (4) - Hardware Offline / Offline



Offline / Offline Hardware Comparison:

In addition, it is possible to compare the hardware between two devices or modules in one device.

15.16. Exercise 4: Testing the Motor Jog



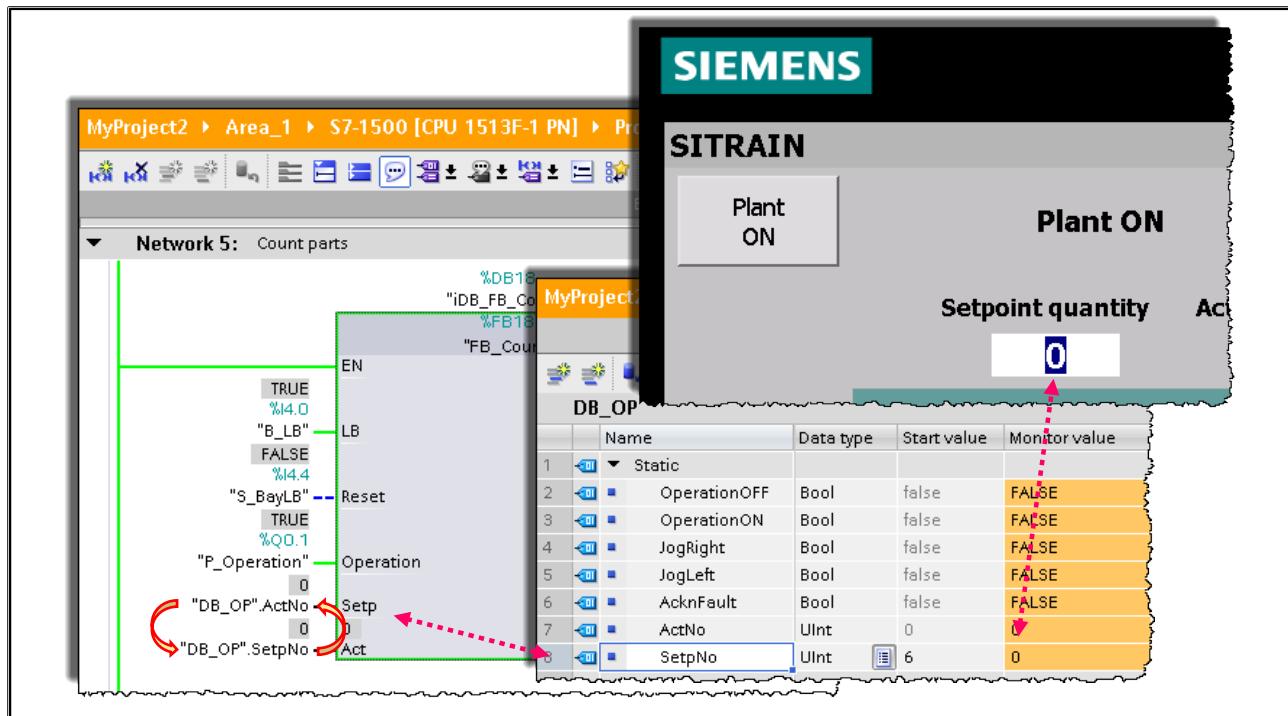
Task

The function "Jog conveyor motor" does not work. The combined use of the PG functions "Monitor block" and "Watch table" (monitor tags [variables]) indicates that there must be a double assignment at output "K_Right" (Q4.5). The task now is to find all instructions in the entire user program that write-access this output.

What to Do:

1. Carry out a CPU restart.
2. On the touchpanel, switch off the "Plant".
3. Open the "FC_Conveyor" block and activate the "Monitor" test function.
4. In the Project tree, under "Watch and force tables" create a new Watch table and in it monitor the output "K_Right" (Q4.5).
5. Display the Blocks Editor with the opened "FC_Conveyor" and the Watch table one below the other by splitting the working area (see picture).
6. Interpret the different status displays of the two test functions.
7. Localize the double assignment at output "K_Right" (Q4.5) with the help of the reference data, correct the error and save the change.
8. Download all modified blocks into the CPU and check how the program functions.

15.17. Exercise 5: Entering the Setpoint Quantity



Task

In the search for this logical error, you are to use the test function Cross-references.

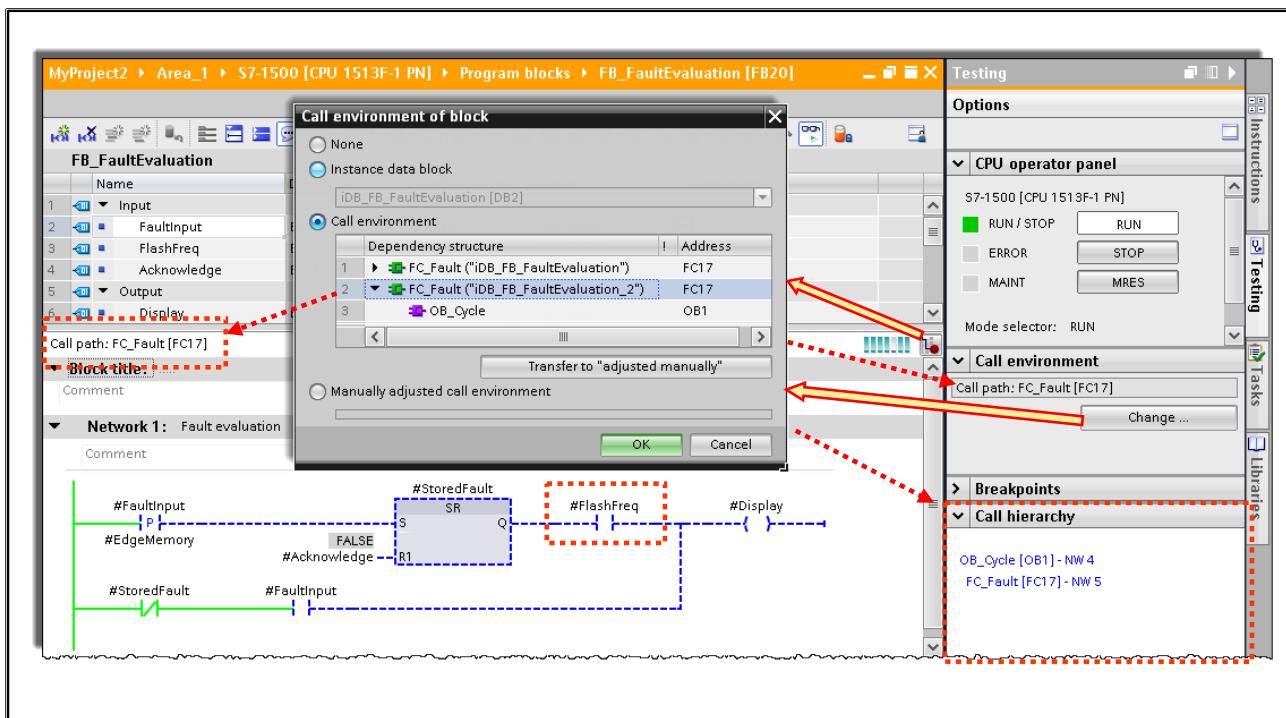
Function Test

Check whether you can enter/change the setpoint quantity of the parts to be transported. Should this not work, correct the error.

What to Do:

1. Try to change the setpoint quantity on the touchpanel.
2. In the project, in the Touchpanel device, open the "Conveyor" screen, highlight the input field "Setpoint quantity" and display the Cross-references in the Inspector window.
Note: via the menu item "Cross-reference information" in the context menu of the input field, you can open the Cross-references in the Inspector window.
3. Switch to the connected PLC tags by clicking the link in the "Address" column.
4. Find a "write" or "read + write" access in the "Access" column.
5. Now you jump to where the tag is used in the PLC by clicking the relevant link in the "Reference location" column.
6. You will see that the tag "DB_OP".SetpNo was assigned as actual parameter to the formal parameter "Act" for the call of "FB_Count" and the tag "DB_OP".ActNo to the formal parameter "Setp".
7. Correct the parameterization of "FB_Count".
8. Save your project, download the modified program and once again test the entry of the setpoint quantity.

15.18. Exercise 6: Testing the Evaluation of Fault 3



Task

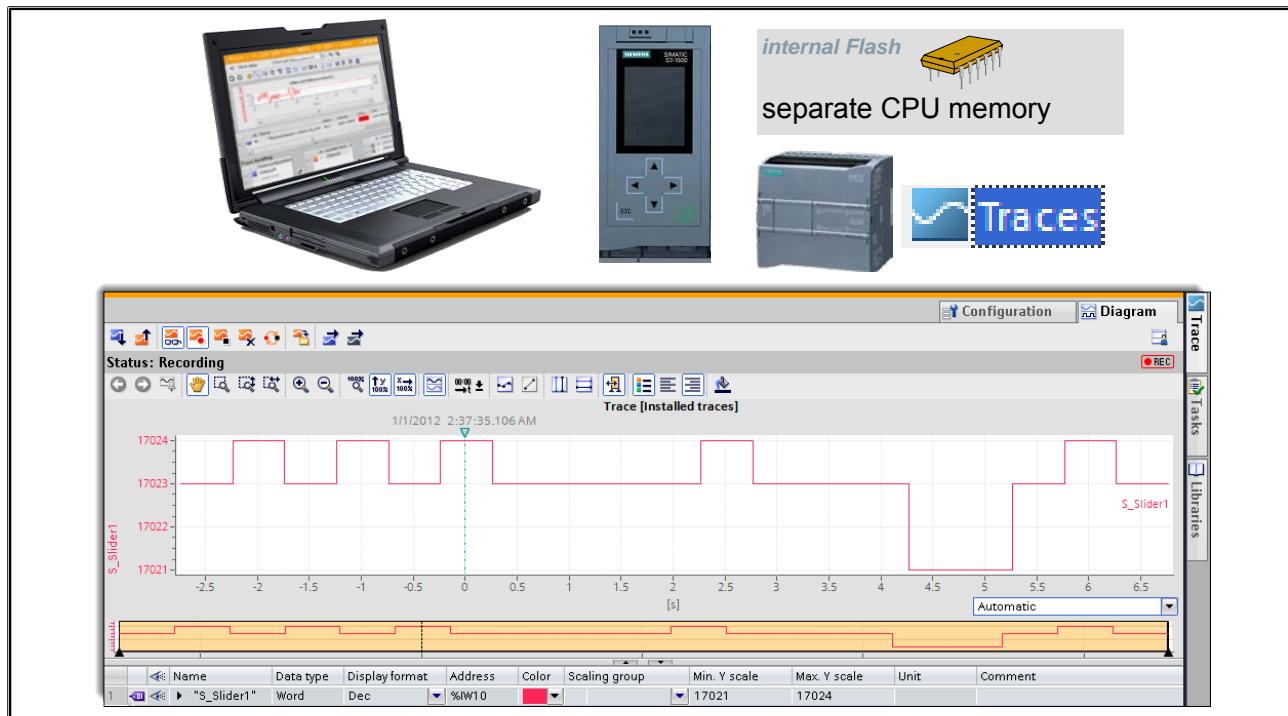
In the search for the last logical error, you are to use the test function 'Monitor block with trigger' on a certain block call.

Check whether the occurrence of *Faults 2 and 3* (switches I 0.5 and I 0.6 on the simulator) is displayed on the simulator LEDs "P_Fault2"(Q0.3) and "P_Fault3" (Q0.4) with a flashing light, and whether, after acknowledging with the simulator pushbutton "S_Acknowledge" (I 0.7), the flashing light of the individual faults switches to a constant light.

What to Do

1. To troubleshoot why no flashing light is displayed after *Fault 3* occurs, first of all monitor "FB_FaultEvaluation" with the test function "Monitor block". You will see that the static variable #StoredFault is controlled when Fault 2 exists, however, not for Fault 3. The cause of this lies in the fact that you are monitoring the execution of "FB_FaultEvaluation" for the evaluation of Fault 2 or the first FB call in "FC_Fault".
2. Change the call environment as shown in the picture in such a way that you specifically monitor the execution of "FB_FaultEvaluation" for the evaluation of Fault 3.
3. In monitoring the second block call you will see that the status of the input parameter #FlashFreq does not change to 2Hz flashing frequency as expected.
4. Since the parameter in the block is not overwritten, the input parameter must be assigned incorrectly or not assigned at all for the call of the function block.
5. Correct the error and once again monitor the evaluation of Fault 3.
6. Download all modified blocks into the CPU and check the function.
7. Save your project.

15.19. TRACE Analyzer Function



"Trace" Analyzer Function

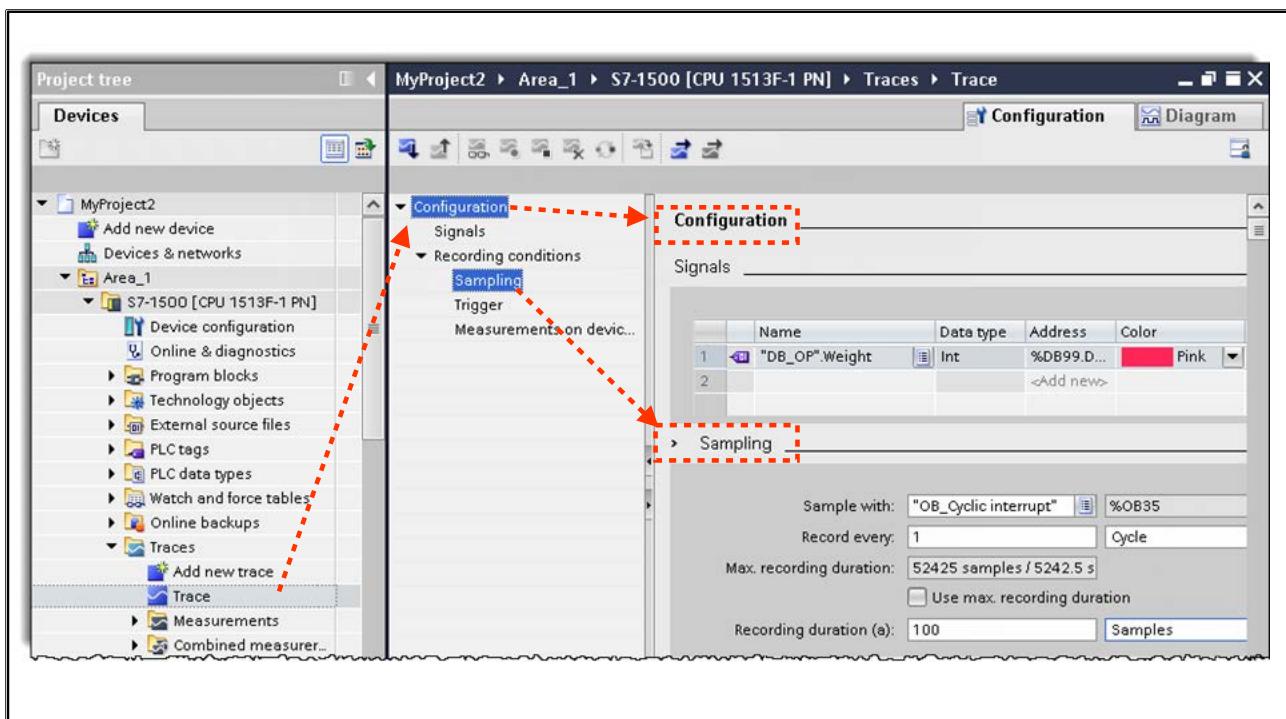
The value-over-time of one or several CPU tags (max. 16) can be stored in a TRACE. In STEP 7, a TRACE recording can be presented graphically.

The number of Traces depends on the CPU.

Depending on the CPU, internal TRACE memories with 512 Kbyte each are available.

- S7-1200 2x TRACE (FW ≥ V4.0)
- Up to S7-1517 4x TRACE, S7-1518 8x TRACE
- A maximum of 16 Trace signals or CPU tags (variables) can be recorded per Trace.

15.19.1. Configuring a TRACE - Signals and Sampling



Trace – Signals

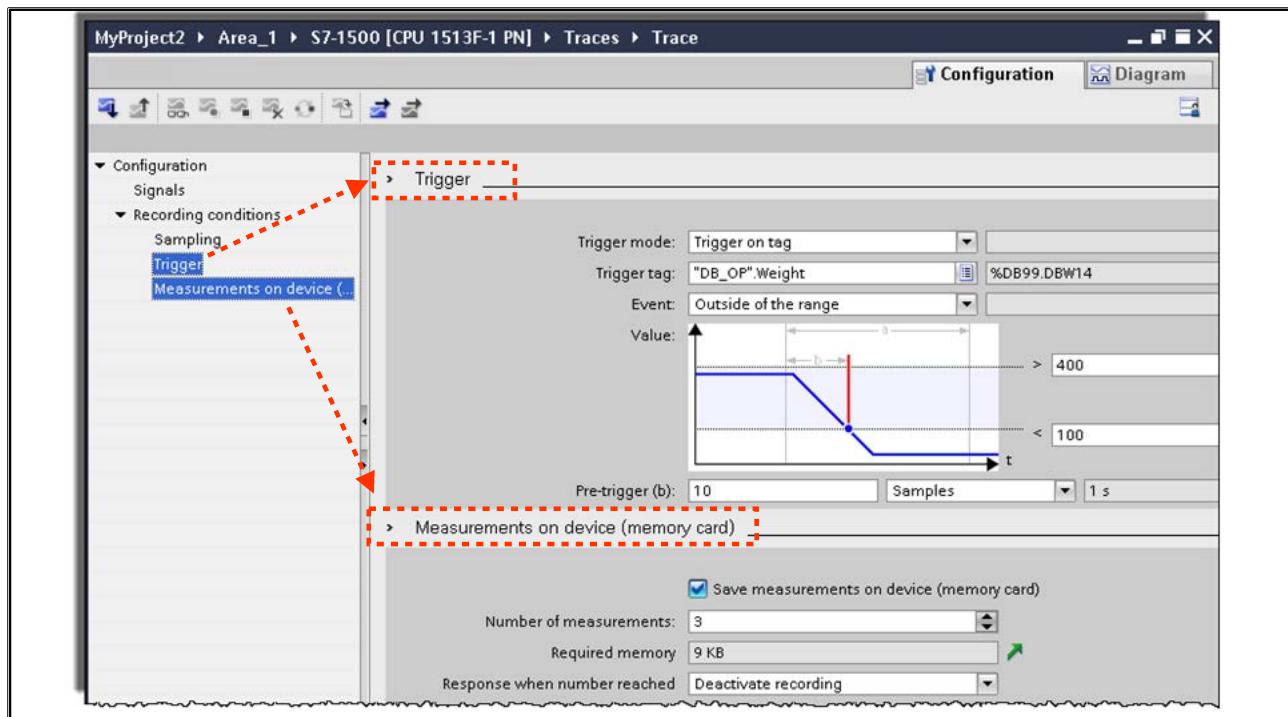
All global PLC tags (variables) of an elementary data type can be recorded.

Trace – Sampling and – Recording Duration

Here, you define how often or in which intervals the Trace signals are to be recorded. From these sampling intervals and the data type or the dimension of the Trace signals you get the maximum duration of a Trace recording since the memory space available for the recording is limited.

- Maximum memory space per Trace: 512 Kbytes – 30 bytes (for internal management) = 524,258 bytes
Each sample is saved with a time stamp (8 bytes). This results in a...
- number of bytes per sample = 8 bytes + number of bytes of a sample
(for Boolean Trace signals, the number of bytes of a measured value is 1 byte)
- Example: Trace with 1x INT variable and a sampling interval of 100ms
 - Trace signal of the data type INT -> 8+2 bytes/sample -> 52,425 possible samples
 - Sampling interval = 100ms -> 10 samples per second
 - -> maximum recording duration: 52425 samples / 10 samples per second = 5242 seconds

15.19.2. Configuring a TRACE – Trigger and Saving Measurement on Device



Triggering the Trace Recording

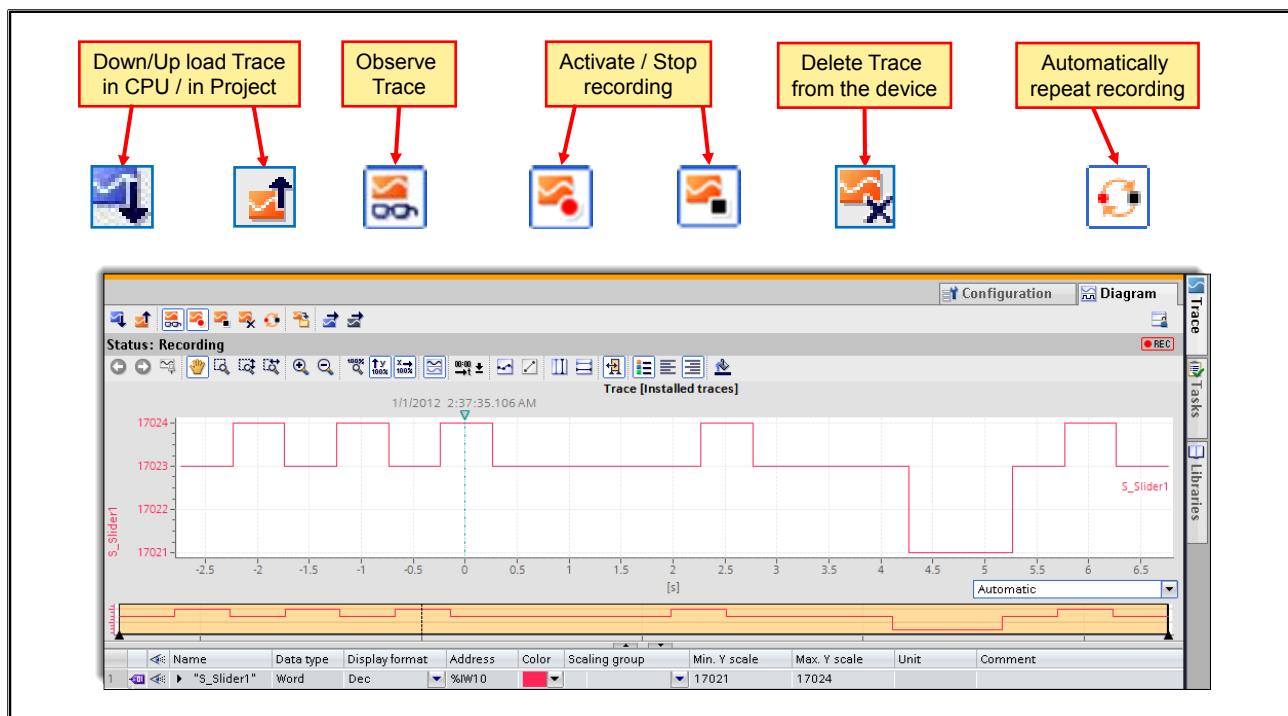
Activating the Trace recording starts the measurement and recording of the Trace signals, however, not the permanent saving of values since these are merely only temporarily saved in a ring buffer which is continuously overwritten with new values. Only when the configured trigger event is fulfilled, are the temporarily saved values permanently saved and no longer overwritten with new values, whereby the trigger event is dependent on the data type of the trigger variable. The Trace recording ends as soon as the maximum recording duration configured in Trace Sampling is reached.

By defining a pre-trigger, you determine how many of the samples recorded before the trigger event occurs are to remain stored.

Measurement on Device (Memory Card)

Completed measurements can be stored on the memory card in order to start a new measurement. In the item "Measurements on device", you can define if several and, if yes, how many measurements are to be made. In addition, you define whether the oldest measurement is to be deleted when the set number of measurements is reached or whether no more measurements are to be made.

15.19.3. Downloading a TRACE into the CPU and Activating It



Transfer Trace Configuration to Device / **Add Trace Configuration from the Device to Trace Configurations**

After the Trace has been configured offline, that is, in the project, the configuration must be downloaded into the CPU, since it is not the engineering tool that executed the Trace but the CPU.

Only Trace configurations that exist online in the CPU can be uploaded from the CPU into the project.

Observe Trace

Observe Trace displays the status of a Trace on the CPU:

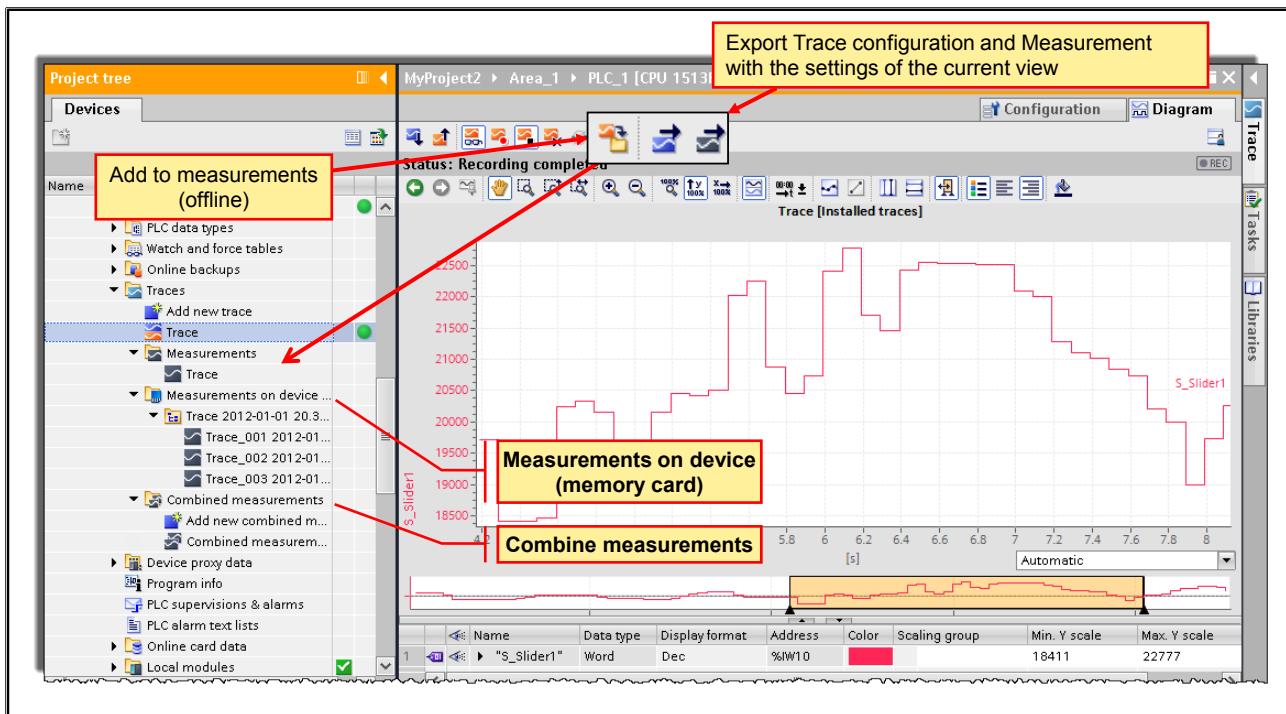
- inactive (Trace already loaded in the CPU, but not yet activated)
- wait for trigger (Trace activated in the CPU, but trigger event not yet fulfilled)
- recording running (Trace activated in the CPU and recording running)
- recording completed (Trace activated in the CPU and recording already completed)

Activate Recording / **Deactivate Recording**

The Trace is activated with "Activate recording", that means that the measuring and recording of the Trace signals is started immediately, even if the possibly configured trigger event is not yet fulfilled. The recorded values are continuously displayed and stored in a ring buffer which is continuously overwritten with new values. Only when the trigger event is fulfilled, are the recorded values no longer overwritten and remain saved. As of this time, the recording is still continued until the maximum recording duration is reached.

Through "Deactivate recording", a Trace with the status "Wait for trigger" is deactivated (stopped) or an already running recording is aborted.

15.19.4. Evaluating, Saving, Exporting a TRACE in STEP 7



View and Evaluate Trace **Diagram**

When an online connection exists, the Trace recording currently saved in the CPU is displayed in the Diagram view of the Trace editor.

You will find the measurements stored on the memory card in the "Measurements on device" folder.

Trace recordings saved offline in the project can be looked at by double-clicking on the Trace recordings saved in the Project tree in the "Measurements" folder.

Furthermore, in the "Combined measurements" folder, measurements can be simultaneously evaluated and compared.

Save Trace in Project

With this function, Traces saved online in the CPU can be uploaded into the offline project (Add to measurements). A Trace can only be saved in the project when it is full or the recording has been stopped.

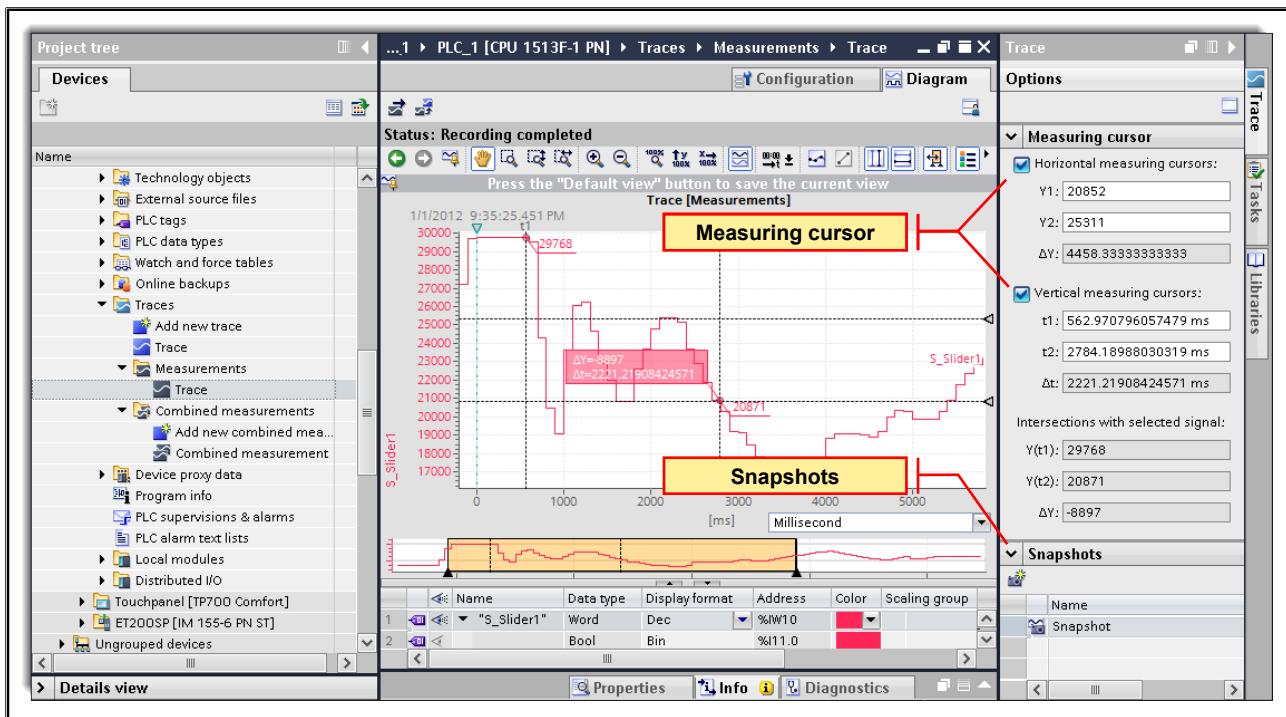
Trace Configuration

This function exports the configuration of a Trace which can then be imported into other projects. (TTCRX-file)

Trace Measurement:

This function exports a Trace recording in CSV-format which can then be further processed with MS Excel, for example, or, in TTRCRX-format which can be imported into a TIA project.

15.19.5. Trace Task Card



"Measuring Cursor" Pane

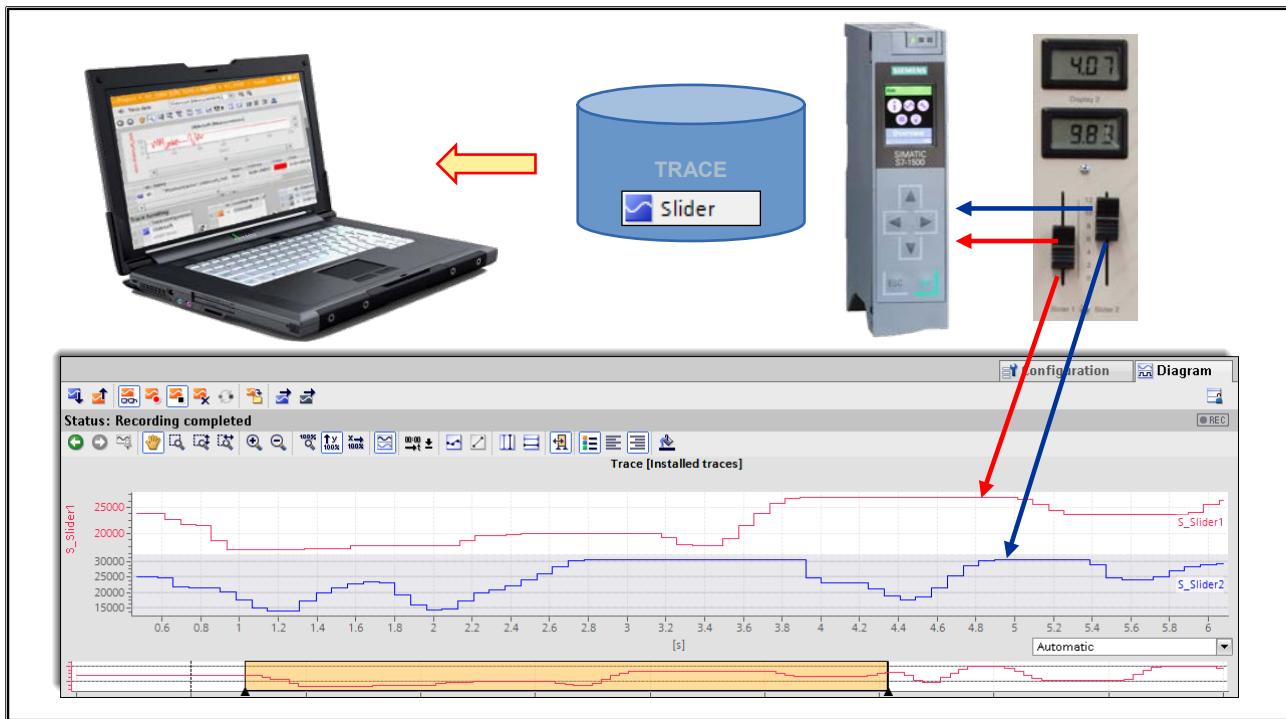
The "Measuring cursor" pane shows the position of the measuring cursor in the graph and the values at the intersections.

"S_snapshots" Pane

The "S_snapshots" pane enables the saving and restoring of different views of a measurement.

A snapshot is created from the current view in the "Diagram" tab. The snapshots are saved in the measurement with the project.

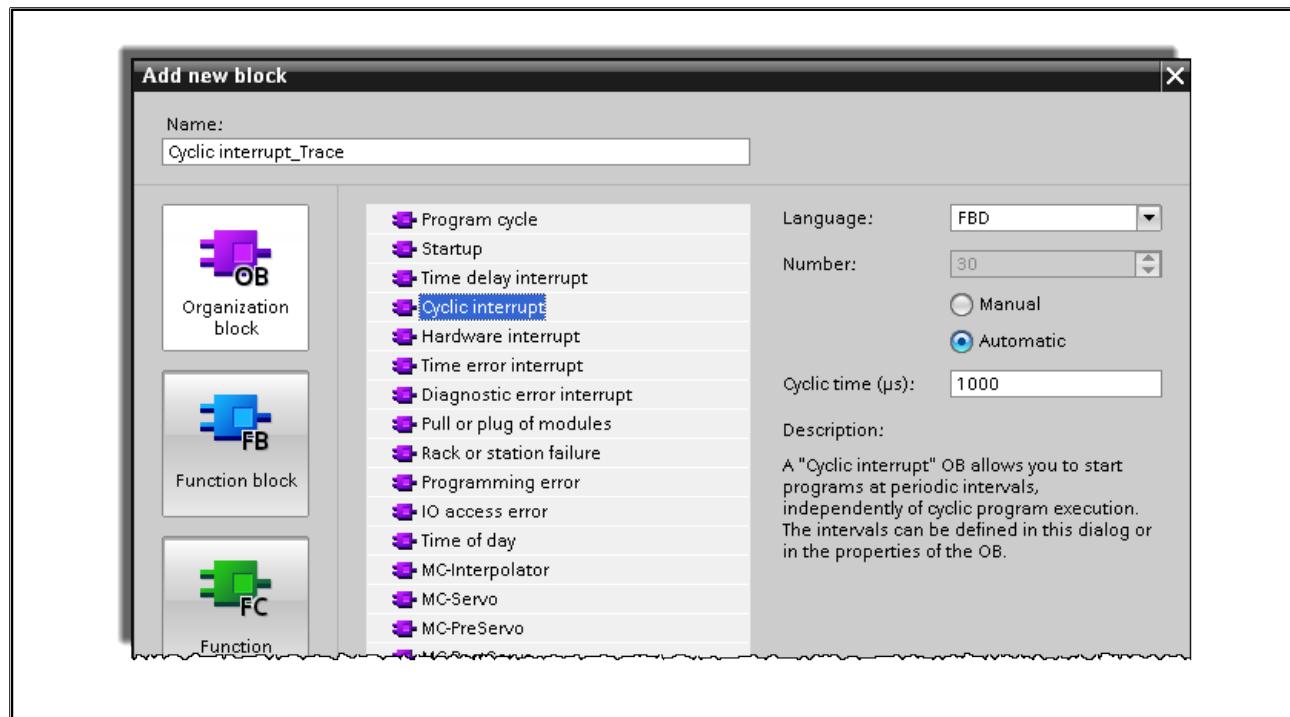
15.20. Task Description: Creating, Looking at and Saving a TRACE



Task Description

The values of Slider 1 and 2 are to be recorded for 10 seconds. The recordings each begin when "P_Operation" (Q0.1) is switched on and are automatically repeated three times. Then, the measurements are to be saved in the project.

15.20.1. Exercise 7: Creating a Cyclic Interrupt OB for the Recording Level



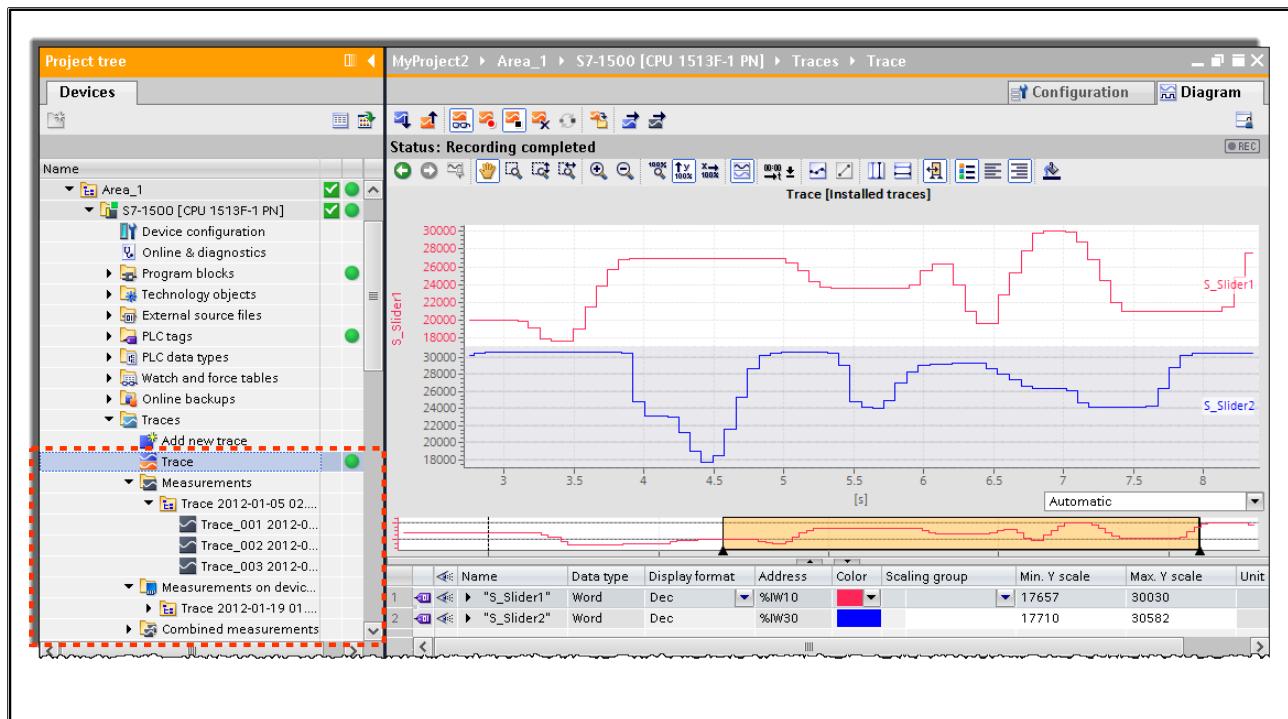
Task

You are to add a cyclic interrupt OB to the program for a regular sampling.

What to Do

1. Start the function "Add new block" in the Program blocks folder.
2. Select the block type "OB" and the start event "Cycle interrupt".
3. Change the Cycle time to 1000μs.
4. Confirm the selection and save your project.

15.20.2. Exercise 8: Creating, Looking at and Saving a TRACE



Task

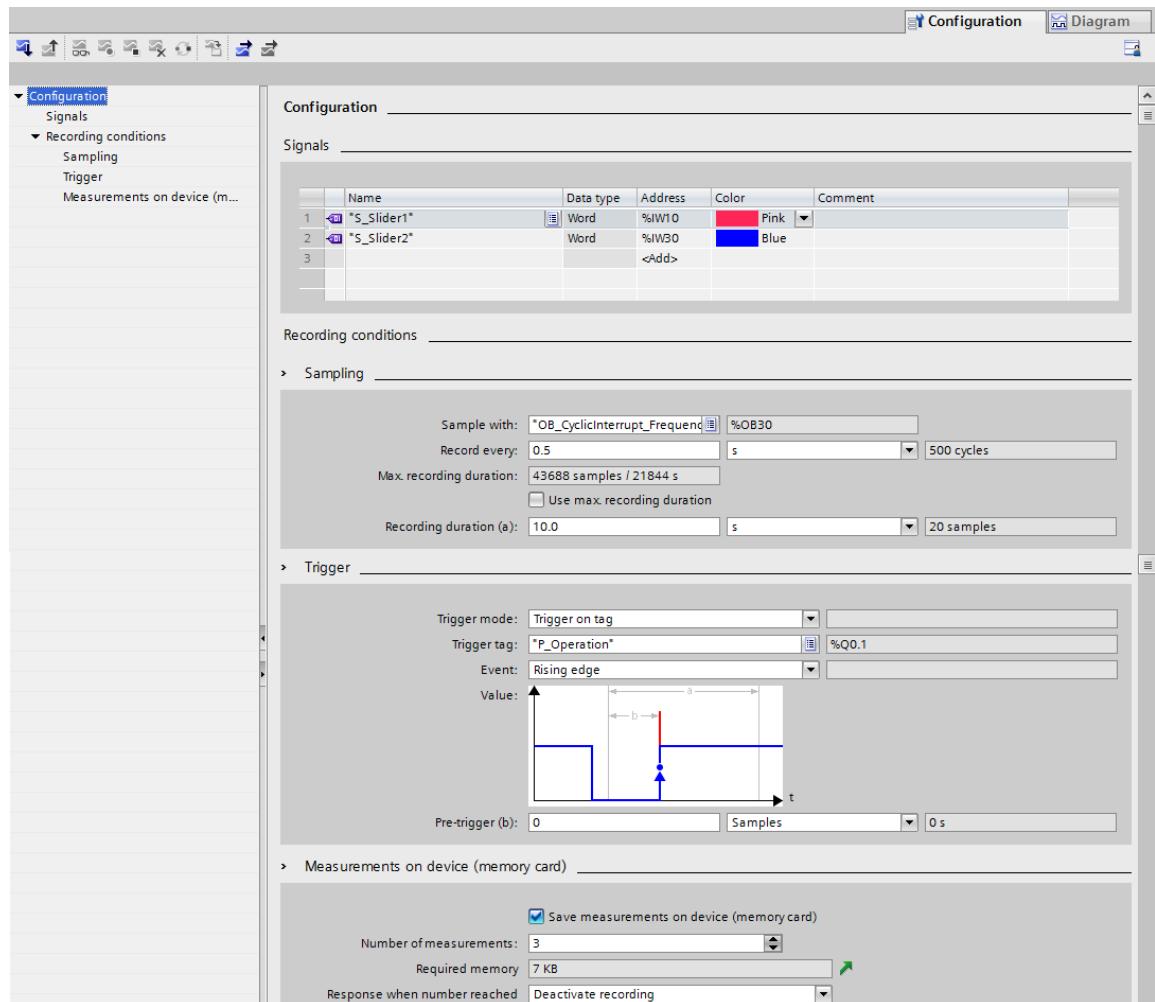
You are to create a Trace configuration with the following properties:

- The Trace signals are to be the variables "S_Slider1" (IW10) and "S_Slider2" (IW30).
- The variables are to be sampled every 0.5 seconds.
- The Trace is to have a recording duration of 10 seconds.
- The trigger condition is a rising edge at the operand "P_Operation".
- The recordings are to begin 1 second before the trigger event.
- After the recording is activated, three measurements are to be recorded and then the recordings are to be stopped.

Continued on the next page

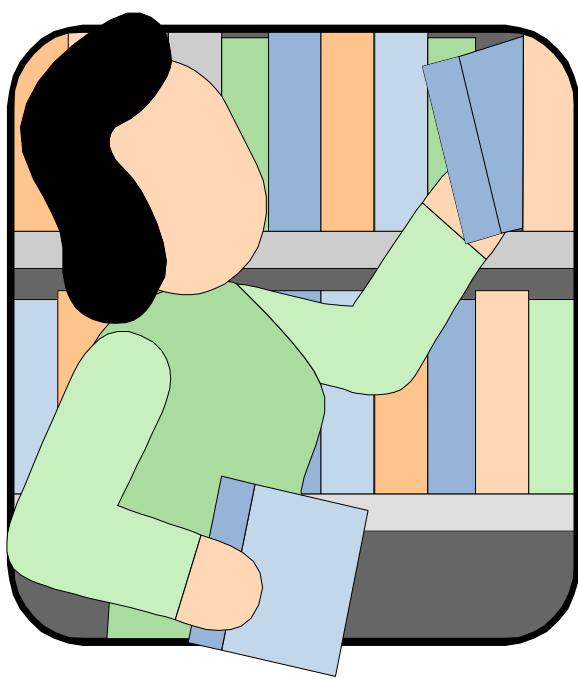
What to Do

1. Create the new trace "Slider" and configure it as shown below.



2. Download the trace into the CPU.
3. Activate the recording of the trace and monitor it.
4. Save the measurements in the "Measurements" folder.
5. Save your project.

15.21. Additional Information



15.21.1. Diagnostic Information about the SIMATIC Memory Card

Checking the “Aging” (service life) of the SIMATIC Memory Card

The screenshot shows the 'Properties' dialog for a 'S7-1500 [CPU 1513F-1 PN]' device. The 'Diagnostics' tab is active. In the 'Diagnostics' section, there is a checked checkbox labeled 'Aging of the SIMATIC memory card' and a text input field labeled 'Threshold value:' containing the value '80 %'. A red arrow points from the text 'CPU Properties' to the 'Properties' button in the dialog title bar.

max. delete and write cycles, see Entry ID: 109482591

Diagnostics Setting CPU

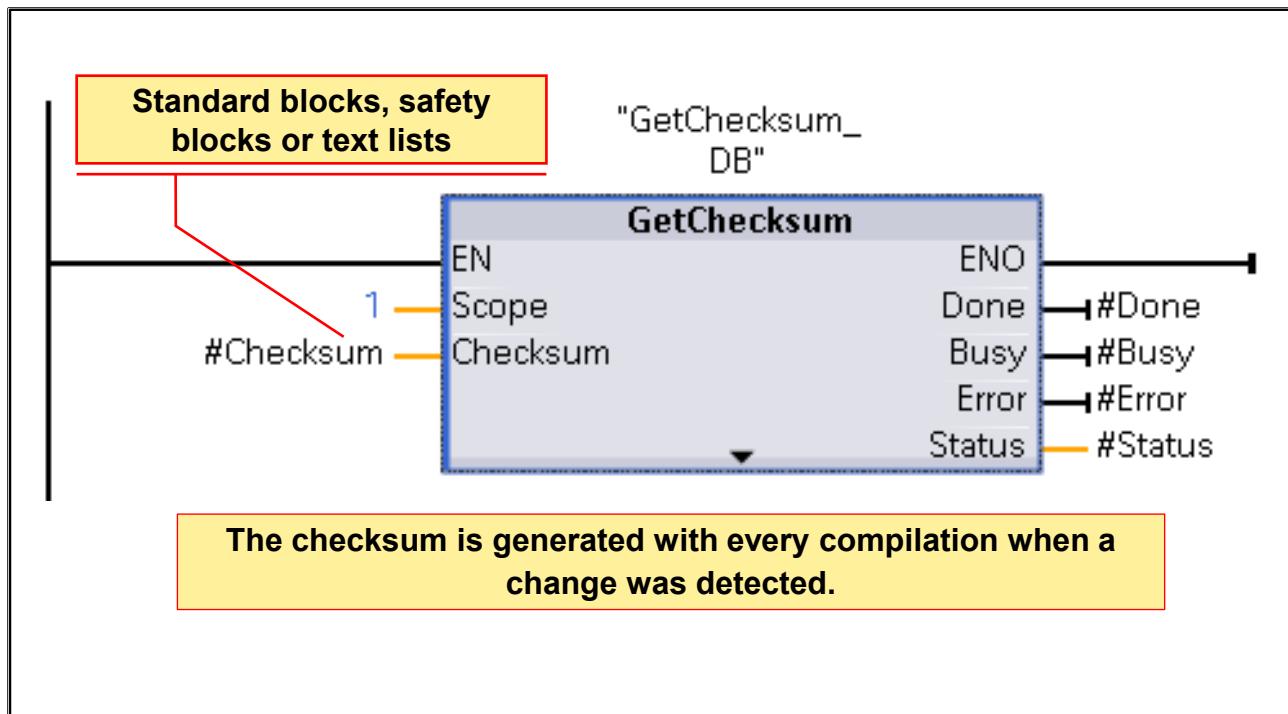
Generation of a diagnostic entry when a service life ‘aging’ defined by you is exceeded (in percent of the guaranteed write/read cycles)

Optic display on the CPU via the Maintenance-LED

Note:

See **FAQ: 109482591** – "How long is the service life of a 32 GB memory card with the S7-1500 when the minimum number of delete/write procedures is 50,000?"

15.21.2. Reading-out the Checksum



Generating the Checksum

During compilation, PLC programs automatically receive a unique checksum. If, during the next compilation, it is determined that the PLC program was changed, the program receives a new checksum. If the PLC program has not changed and is nevertheless recompiled, the checksum remains the same.

Even when changes are carried out and then are undone, the checksum remains unchanged.

Downloading the Checksum

The checksum is downloaded into the CPU along with the PLC program and is available in the online program. Blocks which are generated or modified during runtime (for example, "WRIT_DB", "CREAT_DB" and "DELETE_DB") do not change the checksum. When it is uploaded from the CPU, the checksum is not adopted in the offline project since it is automatically regenerated with the next compilation.

Evaluating the Checksum

The checksum is displayed in the CPU Properties (Properties > General > Checksums). From there, you can manually adopt them in your documents. In order to read out the checksum in the program at runtime, the extended instruction "GetChecksum" is available.

16

Contents

16. Integrating and Commissioning a Drive with Startdrive	16-2
16.1. Task Description: G120 as an Additional Conveyor Drive.....	16-3
16.2. Communication Standard PROFIdrive	16-4
16.2.1. CPU - Drive Communication: CPU - G120	16-5
16.2.2. Standard Telegrams	16-6
16.2.3. Structure of the Control Word	16-7
16.2.4. Structure of the Status Word.....	16-8
16.2.5. Setpoint / Actual Value → Speed Values	16-9
16.3. Inserting a Drive into the Project.....	16-10
16.3.1. Networking a Drive with the CPU.....	16-11
16.3.2. Parameterizing the Module Address and Module Name	16-12
16.3.3. Configuring a Power Unit	16-13
16.3.4. Parameterizing the Process Data Area (PZD).....	16-14
16.3.5. Assigning a Device Name ONLINE (Module Initialization)	16-15
16.4. Parameterizing the Drive with the "Commissioning Wizard"	16-16
16.5. Online Commissioning: Activating / Deactivating the Control Panel	16-17
16.5.1. Operating the Control Panel for Commissioning	16-18
16.5.2. Monitoring Control and Status Word(s) ONLINE	16-19
16.6. Exercise 1: Resetting to Factory Settings.....	16-20
16.6.1. Exercise 2: Reading-out the Firmware Version of the Drive.....	16-21
16.6.2. Exercise 3: Inserting and Networking the Drive in the Offline Project	16-22
16.6.3. Exercise 4: Parameterizing the Drive Communication	16-23
16.6.4. Exercise 5: Parameterizing the Drive OFFLINE with the Commissioning Wizard.....	16-24
16.6.5. Exercise 6: Downloading Drive Parameterization to the Drive and Hardware Configuration to the CPU	16-30
16.6.6. Exercise 7: Assigning the PROFINET Device Name ONLINE	16-31
16.6.7. Exercise 8: Operating the Drive via the Control Panel	16-32
16.6.8. Exercise 9: Commissioning "FC_Drive".....	16-33
16.7. Additional Information	16-34
16.7.1. Monitoring Active Alarms ONLINE.....	16-35
16.7.2. Changing Parameters in the Inverter	16-36
16.7.3. G120 Reset to Factory Settings via BOP-2	16-37

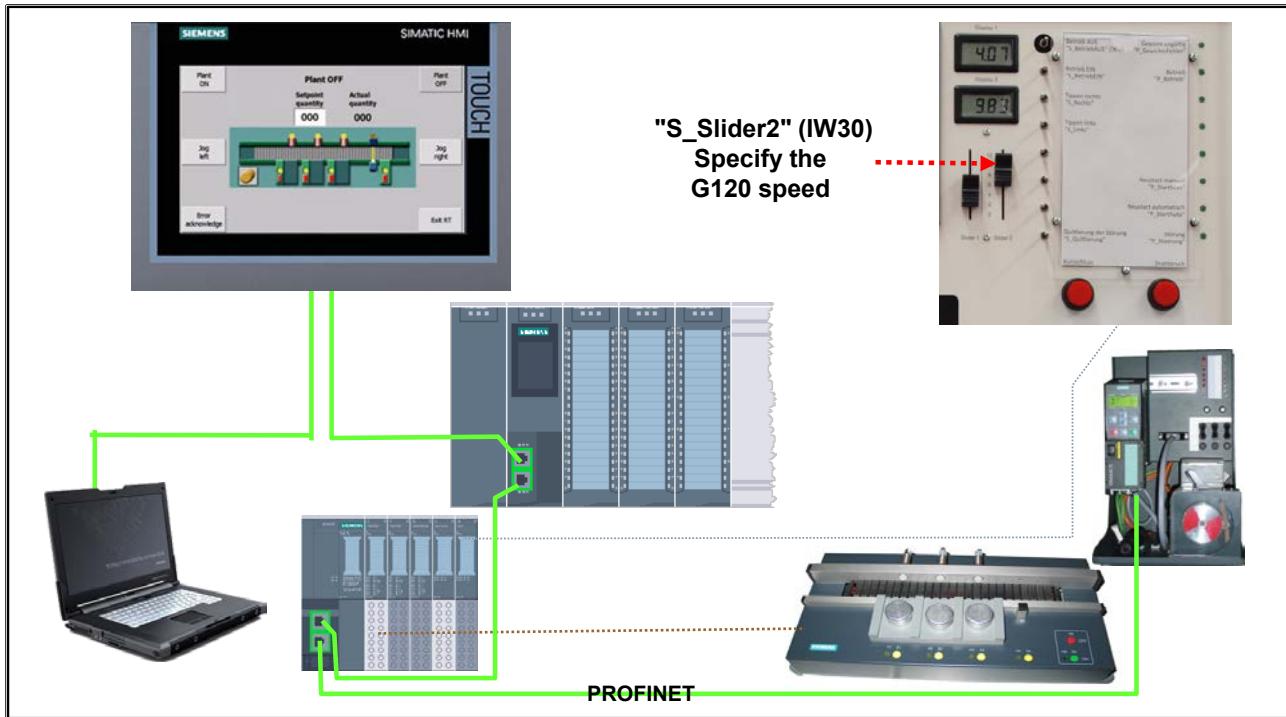
16. Integrating and Commissioning a Drive with Startdrive

At the end of the chapter the participant will ...



- ... be able to parameterize and test a drive with Startdrive
- ... be able to integrate a drive in the device configuration
- ... be able to reset (restore) the inverter to factory settings
- ... be able to set basic parameters via Startdrive
- ... be able to control the drive via a PLC

16.1. Task Description: G120 as an Additional Conveyor Drive



Task Description

Imagine that the drive is being used as an additional drive for the conveyor model.

The drive is to be controlled as follows:

- When "P_Operation" (Q0.1) is switched on (activated operation), the G120 drive is automatically also switched on and off parallel to the conveyor motor.
- The speed of the G120 can be set via the right slide control (potentiometer) on the simulator.

16.2. Communication Standard PROFIdrive

PROFIdrive

- Consistent industrial communication
- Comprehensive range of applications
- PROFIdrive is the standard profile for drives technology in conjunction with the PROFIBUS and PROFINET communication systems
- Proven method for the easy and integrated connection of drives and controllers of different manufacturers

Communication via PROFIBUS and PROFINET

PROFIBUS and PROFINET are the solutions which provide absolute consistency and are highly application-oriented. The main reason that PROFIBUS and PROFINET stand out from other industrial communication systems is because they span such an extraordinary breadth of applications.

One of the most important applications in industrial automation is drives technology.

Open technologies need, for their maintenance, ongoing development and market penetration, a company-independent institution that can serve as a working platform. For this purpose, the PROFIBUS User Organization (PNO) was founded in 1989 as a non-profit organization representing the interests of manufacturers, users and institutes (world's largest interest association in the field of industrial communication).

PNO is a member of the international umbrella organization PI (PROFIBUS & PROFINET International) founded in 1995.

PROFIdrive

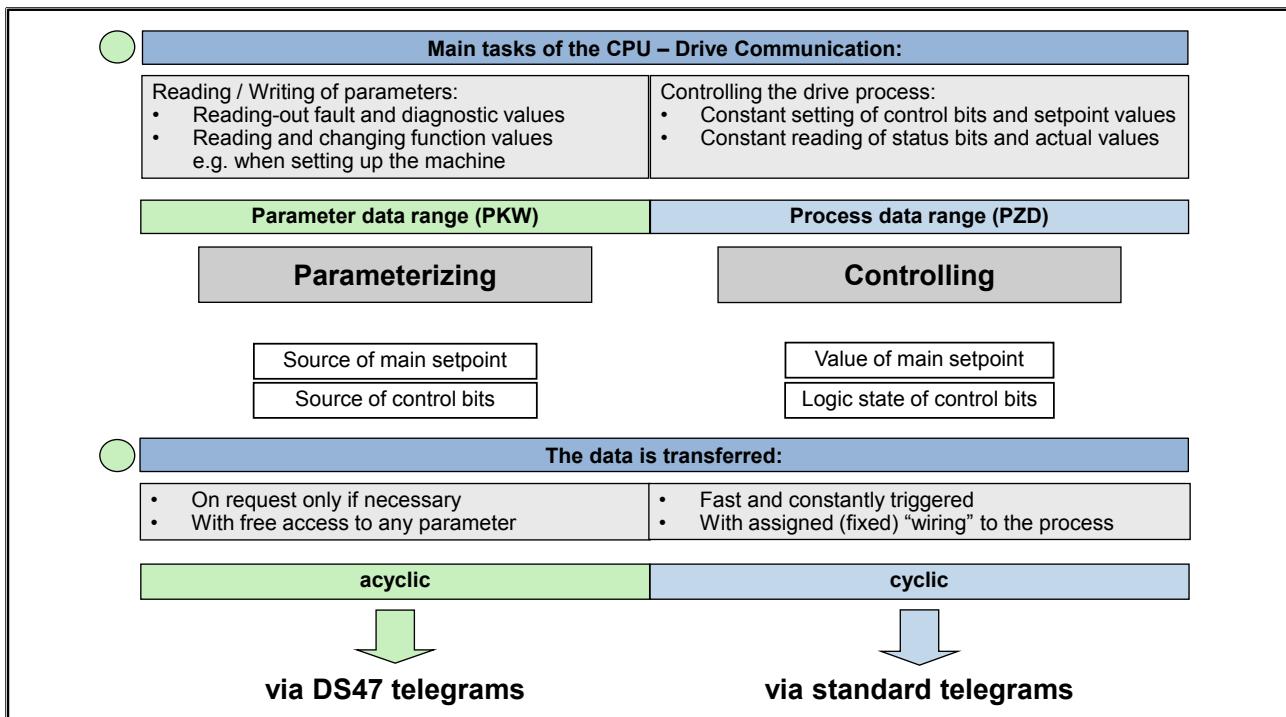
PROFIdrive is the standard profile for drives technology in conjunction with the communication systems PROFIBUS and PROFINET.

→ open "application profile" for connecting drives and controllers of different manufacturers via communication systems



The control word and status word of the G120 comply with the specifications of PROFIdrive profile Version 2.0 or Version 3.0 for the "speed control" operating mode.

16.2.1. CPU - Drive Communication: CPU - G120



Process Control

Fast data transmission of short data telegrams to all participating stations with the maximum speed available on the bus whereby all participants can receive different data. In drive systems these are typically setpoints, control commands, status replies and actual values (measured values).

Operating Control

In addition to this permanently available data, there is data that is only needed in particular cases. It would therefore be senseless to permanently put load on the bus if this data is only needed once per hour or per day, for example, when starting up the machine (Class 1 Master). Another reason for an expanded communication need could be the commissioning, optimization or diagnosis of machine components from a central location. In this case, (such as a fault) a detailed access of an Engineering Tool to the system components or the drive is enabled. Since, as a rule, only one affected device/component is directly addressed, this window is only made available once per bus cycle for one bus station and not simultaneously for every station.

Cyclic Data Exchange:

This is used to fulfill the demand for fast, permanent data exchange.

Acyclic Data Exchange:

The acyclic exchange of entire data sets (for example, DS47, DS100) between Master/Controller and Slave/Device occurs according to standardized procedures which are implemented with FBs from the libraries (for example, Drive ES SIMATIC).

The initiation of a data transmission for the acyclic communication of a Class 1 Master/Controller with a Slave/Device is always triggered by the user program. That is, the user program decides whether a data exchange with a Slave/Device is necessary or not based on further conditions. Only when there is a need, does a data transmission request get signaled to the DP Master/Controller which then executes it.

16.2.2. Standard Telegrams

Telegram	1	20		352		354		999		
PZD1	STW1	ZSW1	STW1	ZSW1	STW1	ZSW1	STW1	ZSW1	STW1 <4>	ZSW1 <4>
PZD2	NSOLL_A	NIST_A	NSOLL_A	NIST_A_GL	NSOLL_A	NIST_A_GL	NSOLL_A	NIST_A_GL		
PZD3				IAIST_GL	<3>	IAIST_GL	<3>	IAIST_GL		
PZD4				MIST_GL	<3>	MIST_GL	<3>	MIST_GL		
PZD5				PIST_GL	<3>	WARN_CODE	<3>	WARN_CODE		
PZD6				<2>	<3>	FAULT_CODE	<3>	FAULT_CODE		
PZD7										
PZD8										
PZD9										
PZD10										
PZD11										
PZD12										

Telegrams 1, 20
Telegrams 352 to 391
Telegram 999

Manufacturer-independent Standard Tel.
Siemens-specific Standard Telegrams
Free telegram

automatic configuration in drive
automatic configuration in drive
manual configuration required

Receive telegram length freely
selectable via central PROFinet
configuration in the master

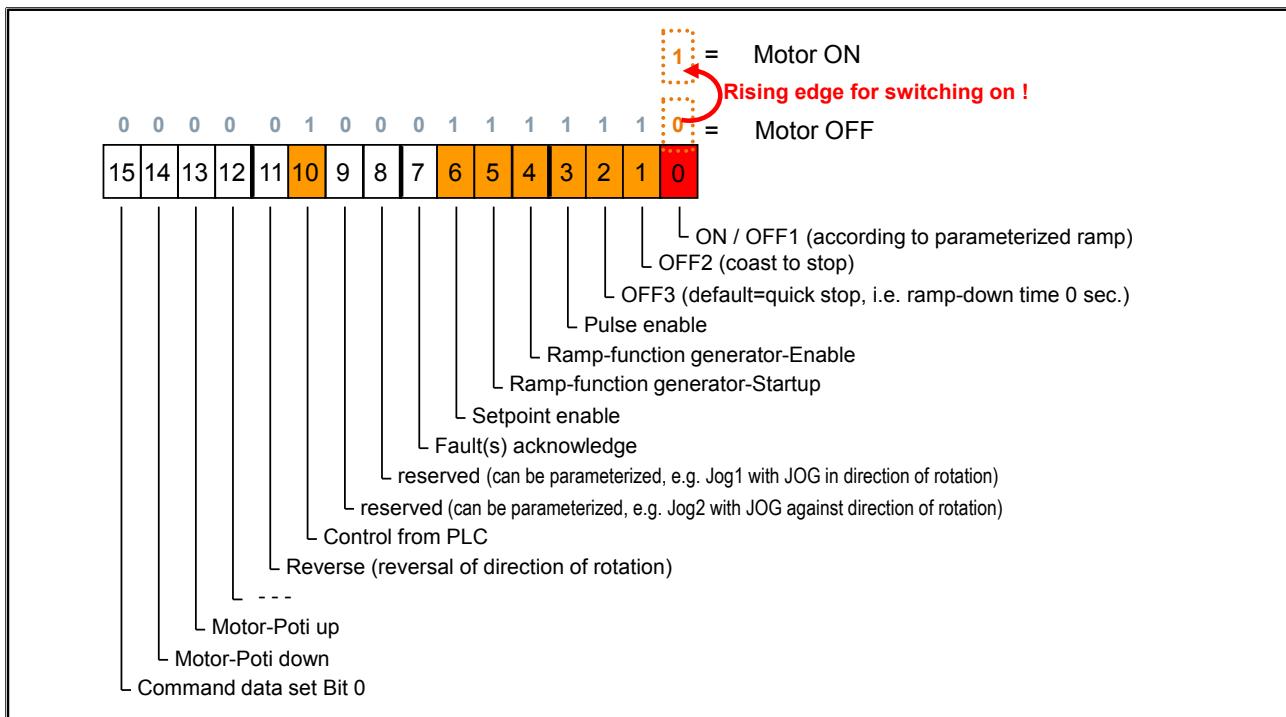
Transmit telegram length freely
selectable via central PROFinet
configuration in the master

Master-Slave / Controller-Device Communication via Standard Telegram

The best way to implement a cyclic data exchange between Master/Controller (CPU) and Slave/Device (G120) is using a standard telegram, which typically is also used for simple speed controls. Between the Master/Controller and Slave/Device, two input words and two output words with the following contents are exchanged:

- Output word 1: Control word
- Output word 2: Main setpoint value
- Input word 1: Status word
- Input word 2: Main actual value

16.2.3. Structure of the Control Word



The inverter can be controlled in various ways. The G120 can either be operated via the field bus or via the terminal strip.

This is controlled through a relevant Control Data Set (CDS)

default for G120 = two CDS:

- Control data set CDS0: Control of the inverter via field bus
- Control data set CDS1: Control of the inverter via terminal strip

Control Word

The control word (Bit 0 to 10) complies with the PROFIdrive profile standard. Bits 11 to 15 are inverter-specific.

For reasons of safety in the case of a wire break, the inverter is always switched off when the relevant OFF bit has signal 0.

ON/OFF1

A rising edge of this bit is generally required for switching on the inverter, and bits OFF2 and OFF3 must have status 1.

When it is switched off using this bit, the motor is braked by the ramp down of the ramp-function generator (Parameter 1121) and the inverter then switches itself off.

OFF2

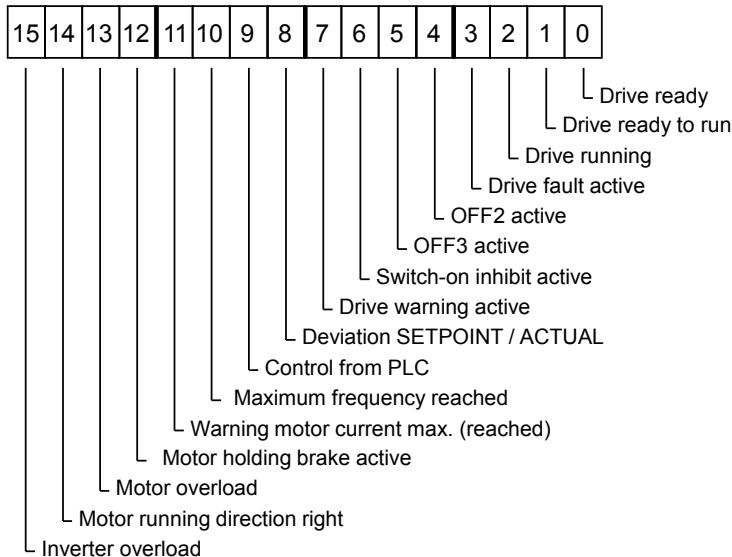
The inverter switches itself off immediately and as a result, the motor coasts down without braking (coasts to a standstill). To switch the inverter on again, this bit must be set to 1 again and a rising edge is necessary at ON/OFF1.

OFF3

The motor is braked by the OFF3 ramp down and the inverter remains on. This function is often used as an EMERGENCY STOP

→ default for OFF3 = 0 seconds (in case of larger motors, this cannot always be achieved).

16.2.4. Structure of the Status Word



Status Word

The status word (Bit 0 to 10) complies with the PROFIdrive profile standard.

Bits 11 to 15 are inverter-specific.

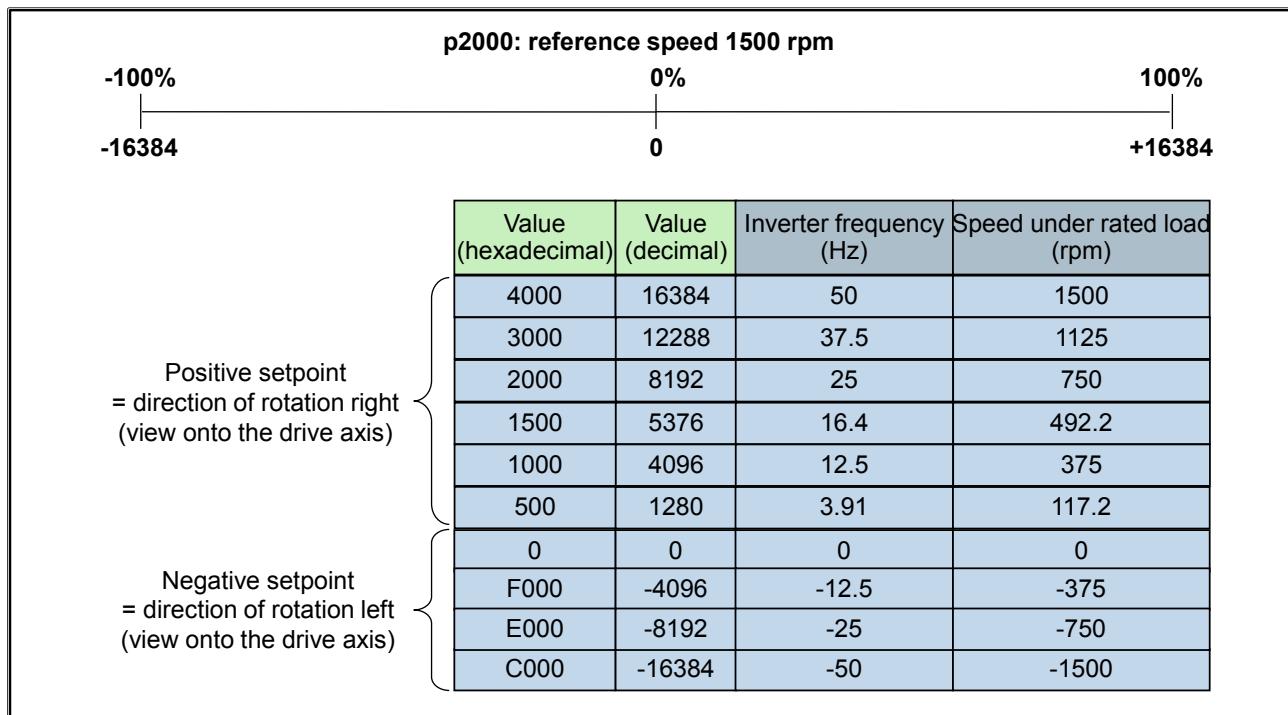
Switch-on Inhibit

This status is reached following the error elimination and acknowledgement of a drive fault. A subsequent switch on is only possible with a "0"→"1"-edge of the bit ON/OFF1.

Warning, Message

Messages (bit 13, bit 15) and Warnings (bit 11)

16.2.5. Setpoint / Actual Value → Speed Values

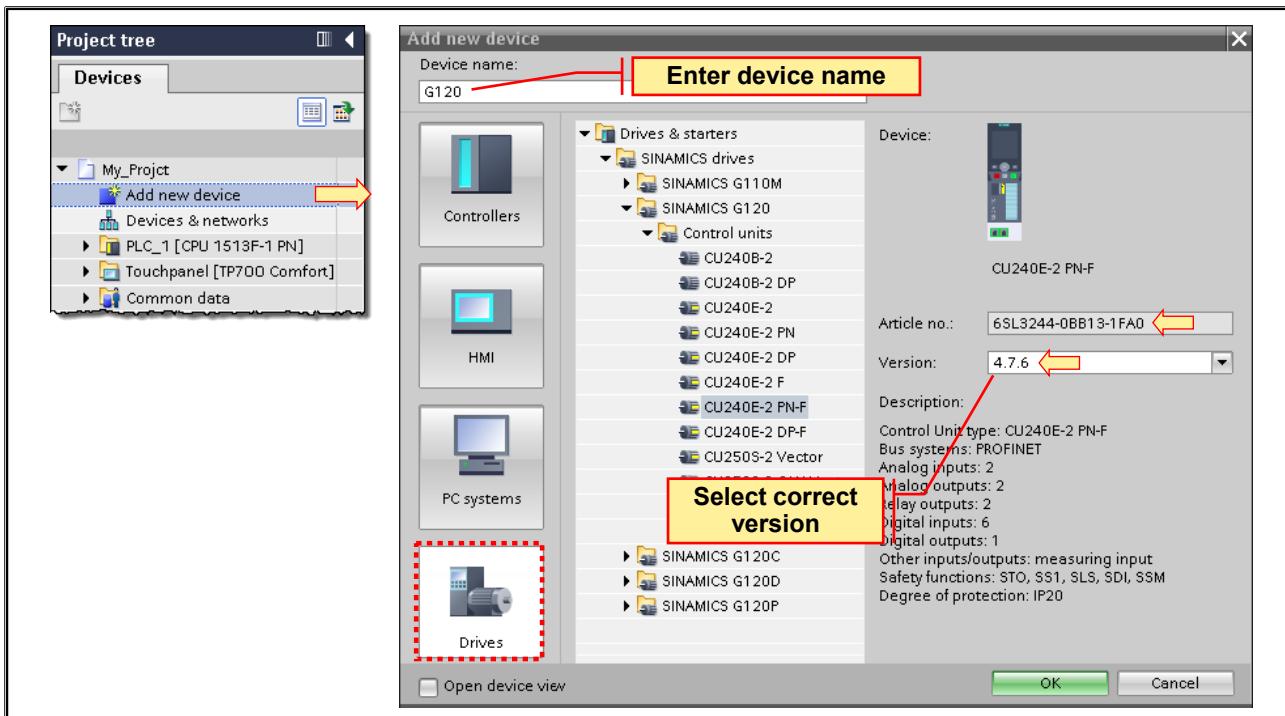


Scaling the Speed Values

Via the parameter p2000, a reference speed is defined which all setpoint and actual values use as a reference.

The range of -100% to 100% of the setpoint or actual value, in turn, is scaled to the range -16384 ($C000_{HEX}$) to +16384 (4000_{HEX}).

16.3. Inserting a Drive into the Project

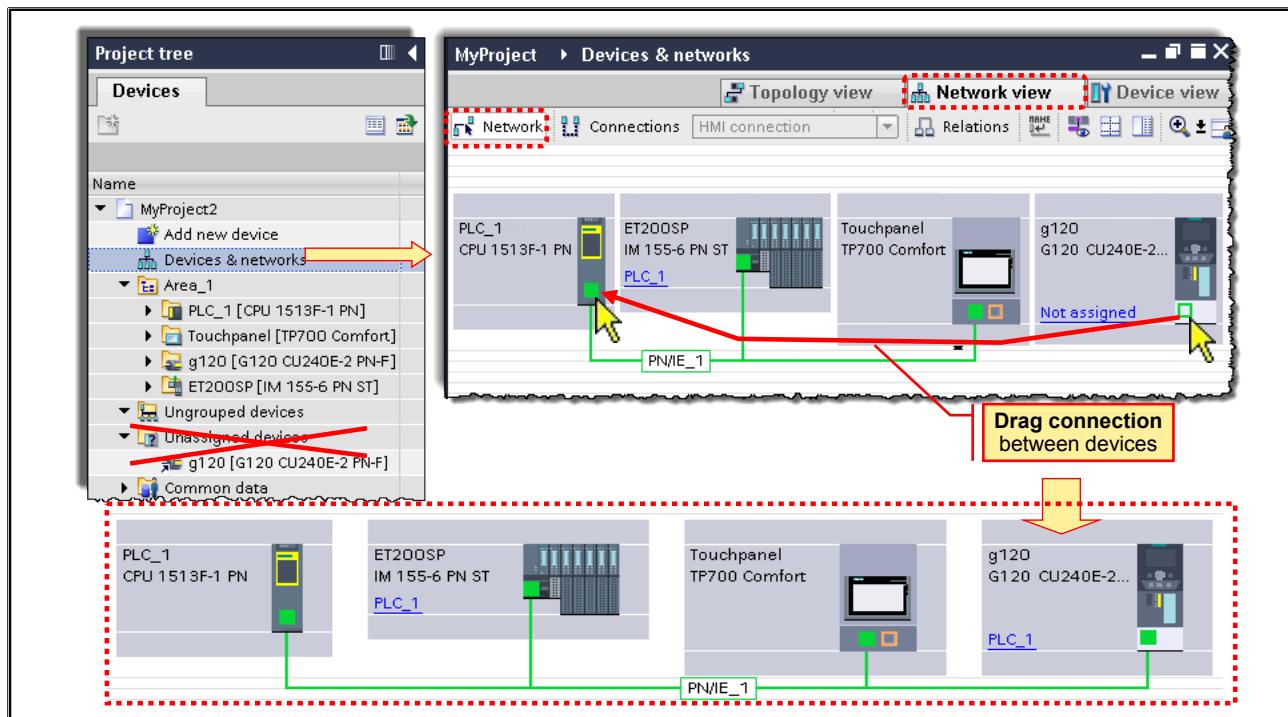


"Startdrive" Option Package

If the "Startdrive" option package is installed in TIA Portal, drives can be added, configured and assigned parameters as a new device.

Otherwise, a drive can be inserted in the project via GSD file as a distributed I/O station using the Hardware catalog. However, then only the communication with the CPU can be configured in the TIA Portal, no hardware configuration, no parameter assignment.

16.3.1. Networking a Drive with the CPU



Editor

Hardware & Networks → "Network view"

Networking

Click on the interface of the one module, hold the left mouse button and drag onto the interface of the other module.

→ here in the picture: PN interface of the G120 dragged onto PN interface of the CPU

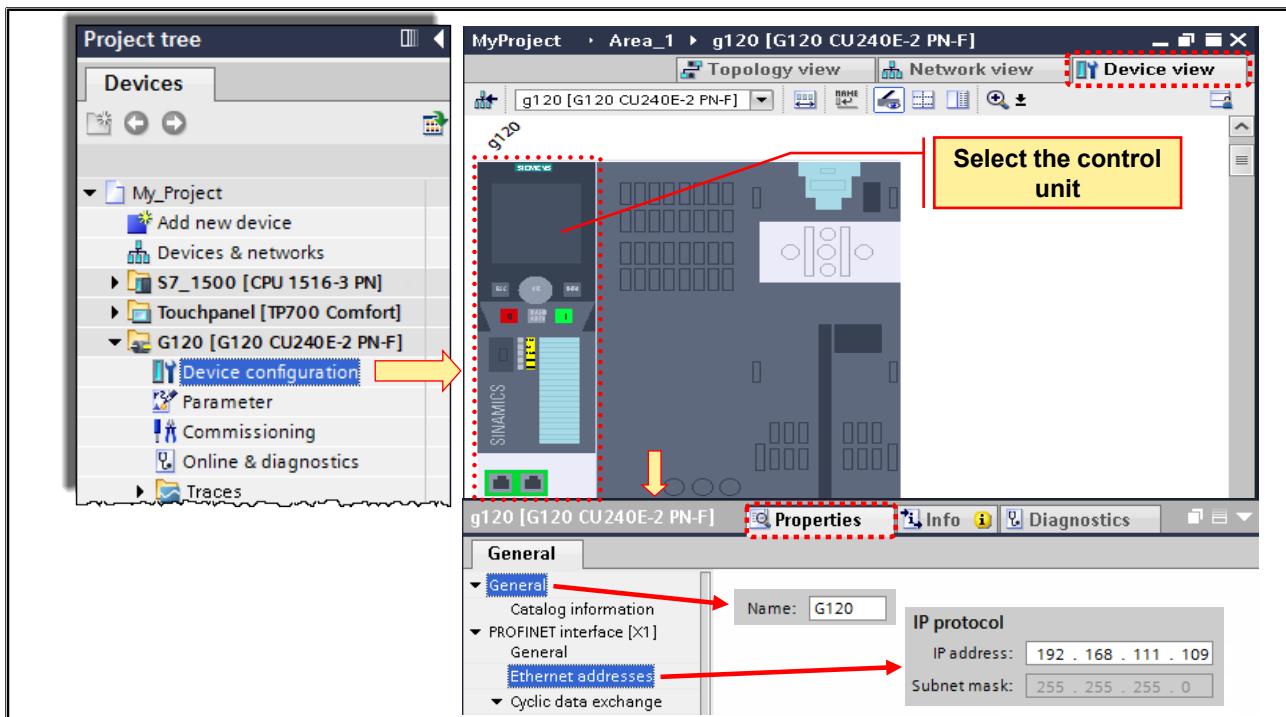
Unassigned Devices

This section then disappears since the device is now assigned to the CPU.

And the link can be found in the Distributed I/O folder of the controller.



16.3.2. Parameterizing the Module Address and Module Name



Name of the Module

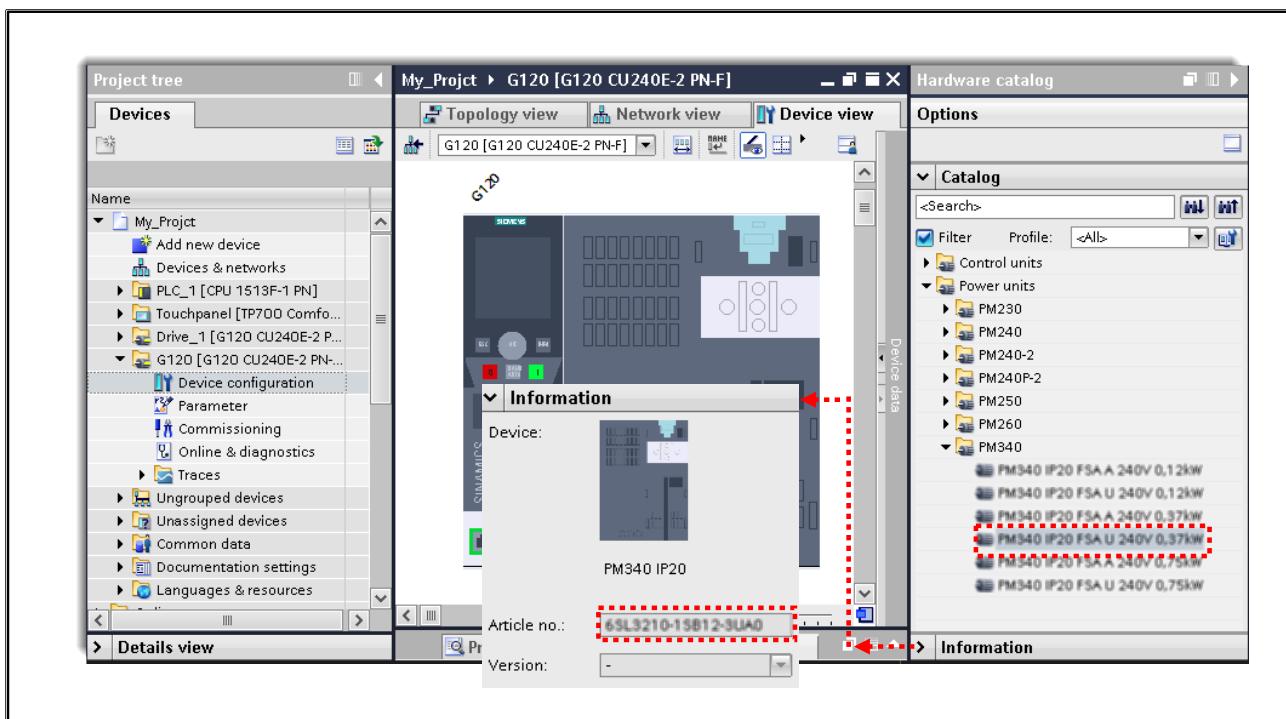
If the property "Generate PROFINET device name automatically" is activated in the Properties of the drive under *PROFINET interface > Ethernet addresses > PROFINET* then the module name set under "Properties > General" is also the PROFINET device name. With the help of this name, it is possible for the CPU to assign the module, that is, the PROFINET-IO Device is identified by the PROFINET-IO Controller through this name.

 This name must be assigned to the PROFINET-IO module online; otherwise, the CPU (PROFINET-I/O Controller) cannot identify the module.

Address Assignment of the Distributed IO-Module

For PROFINET-IO, the parameterized IP address of the module is assigned by the CPU through the configured device name.

16.3.3. Configuring a Power Unit

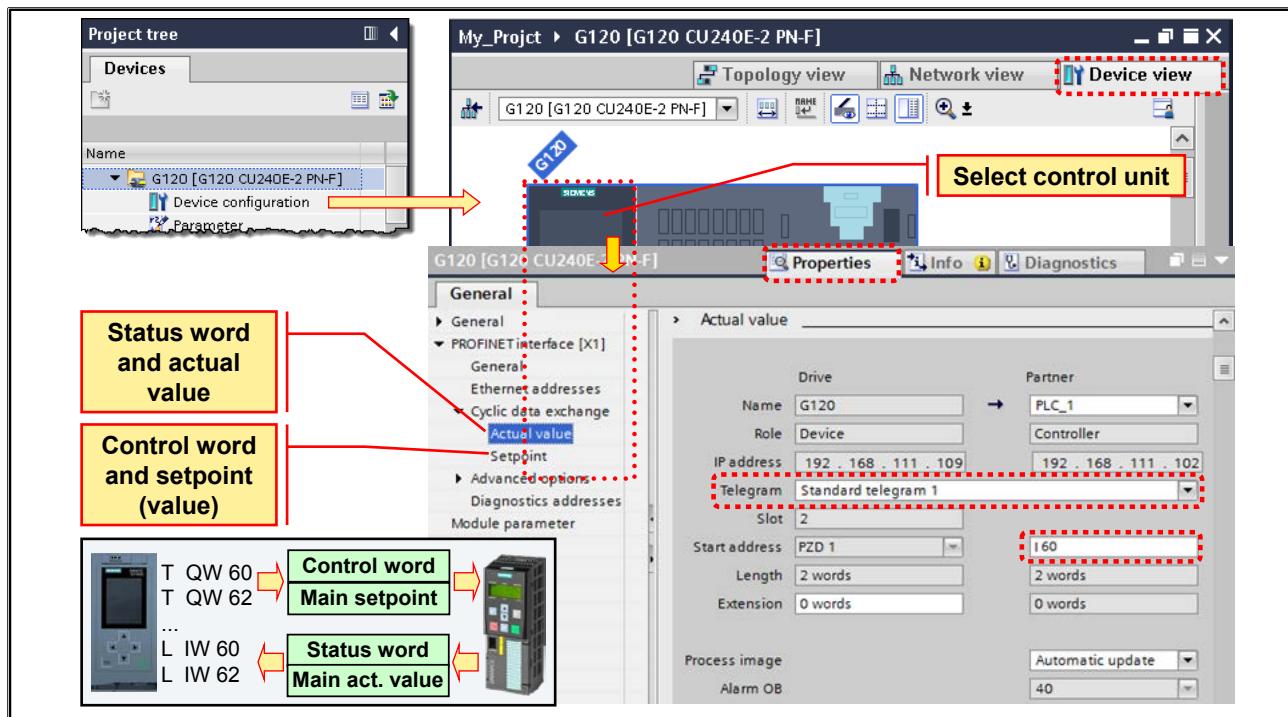


Configuring a Power Unit:

A drive consists of two components

- Control unit → already configured with the device that is added
- Power unit → must be configured

16.3.4. Parameterizing the Process Data Area (PZD)



Process Data Area

Communication using Standard telegram 1 is based on a cyclic process data exchange that is very easy to program.

 The control word and status word comply with the specifications of PROFldrive profile Version 2.0 or Version 3.0 for the "speed control" operating mode.

The control word and, if necessary, the main setpoint are sent from the CPU to the drive. In the response telegram, the drive returns the status word and the main actual value.

Since only two words are exchanged with the drive, simple load and transfer operations are all that are required in the program. When a double word is transferred, data consistency is also assured.

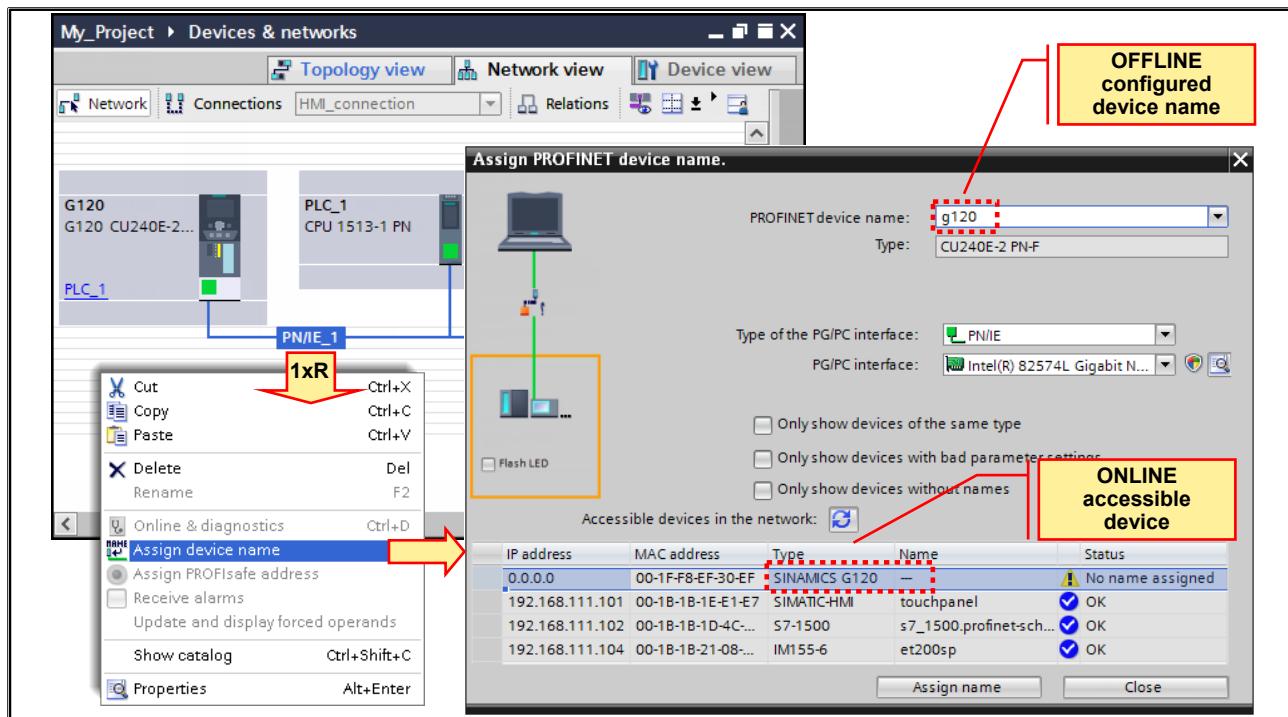
Control Word

The control word contains 16 binary signals for controlling (On/Off, direction of rotation) the drive.

Main Setpoint

According to standard parameterization of the drive, the main setpoint specifies the setpoint speed.

16.3.5. Assigning a Device Name ONLINE (Module Initialization)



Device Name

The PROFINET device name that was configured OFFLINE must be assigned to the drive that actually exists ONLINE.

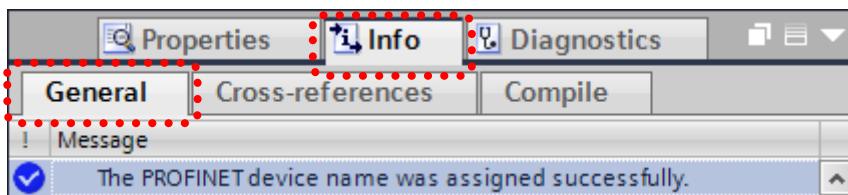
Name of the Module

The name is the assignment CPU ↔ module.

Through this, the PROFINET-IO is identified by the PROFINET-Controller.

This name must be assigned to the PROFINET-IO module online; otherwise, the CPU (PROFINET-Controller) cannot identify the module.

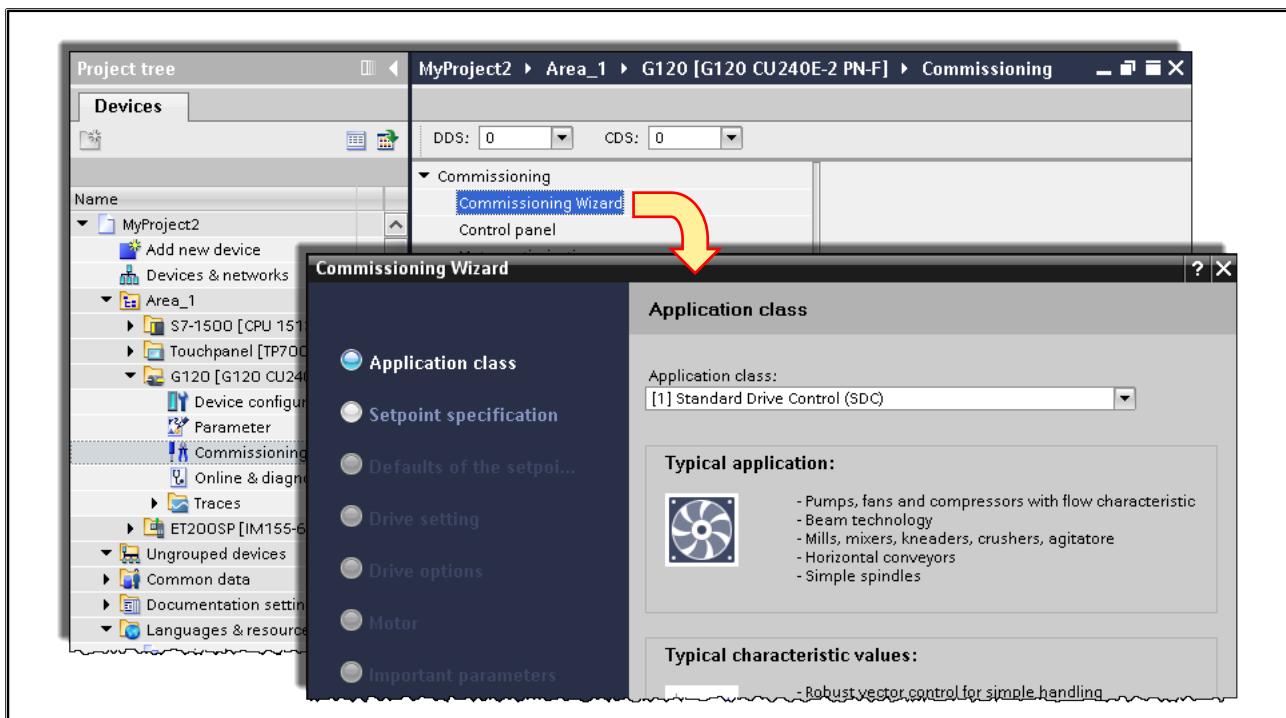
Checking the Result in the Inspector Window



Note:

If you do not use the "Assign device name" dialog but instead the Online access (Online & diagnostics/Functions/Assign name) to set the device name on a device which already has a PROFINET device name, then you will have to restart the device.

16.4. Parameterizing the Drive with the "Commissioning Wizard"

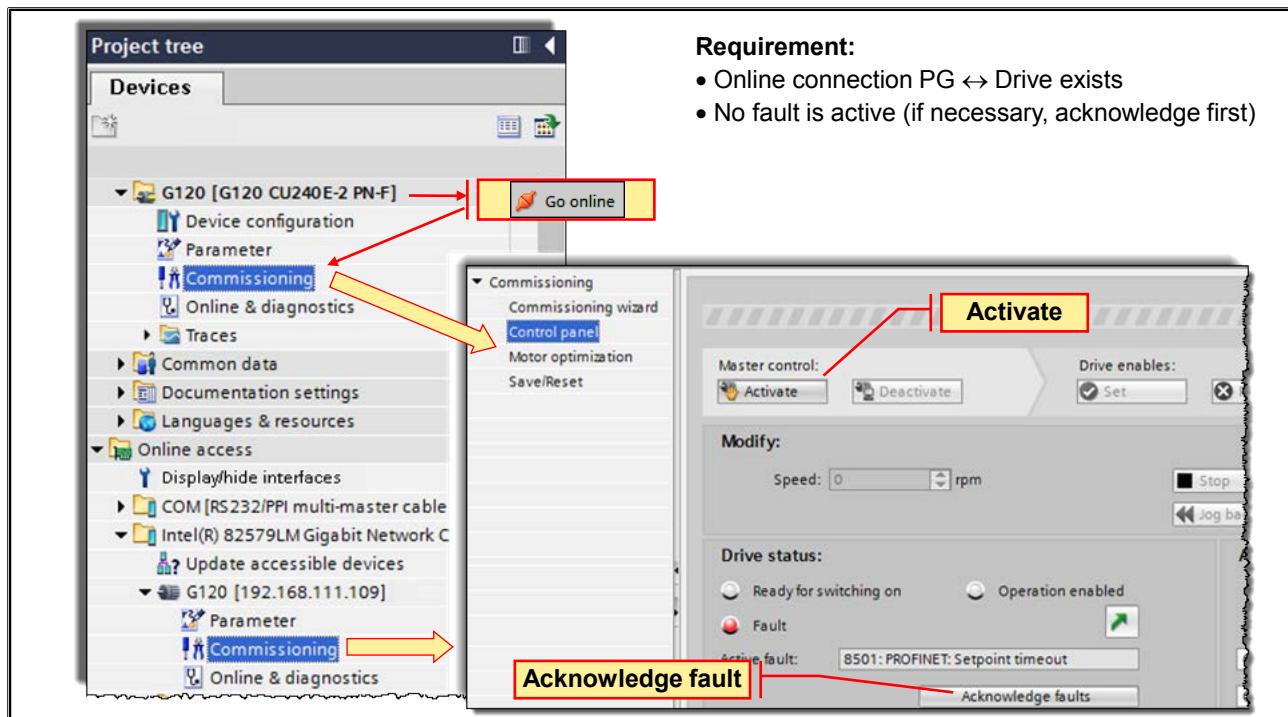


Commissioning Wizard

The most important settings for the drive can be made with the menu-driven Commissioning Wizard. These are, among others:

- Details of the setpoints / command sources
- Drive setting(s)
- Motor (data)
- Important parameters (dynamic data)
- Drive functions

16.5. Online Commissioning: Activating / Deactivating the Control Panel



As soon as the drive has a valid IP address and thus is accessible online, the parameterization and functions of commissioning are available.

Online Accesses

As soon as the drive has been assigned a valid IP address in the subnet used, the functions of commissioning are available here.

Device in the Project Tree

If the drive was inserted as a new device, it is assigned the default IP address 192.168.0.1.

is only possible when the IP address of the device configuration corresponds to the subnet used and is also assigned to the drive.

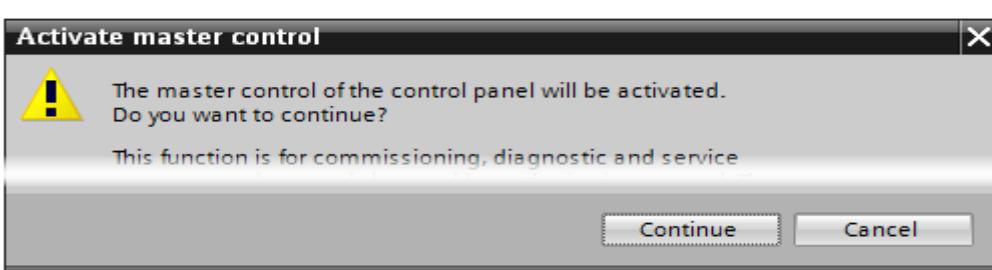
Control Panel

This can only be activated when an online connection exists and no fault exists.

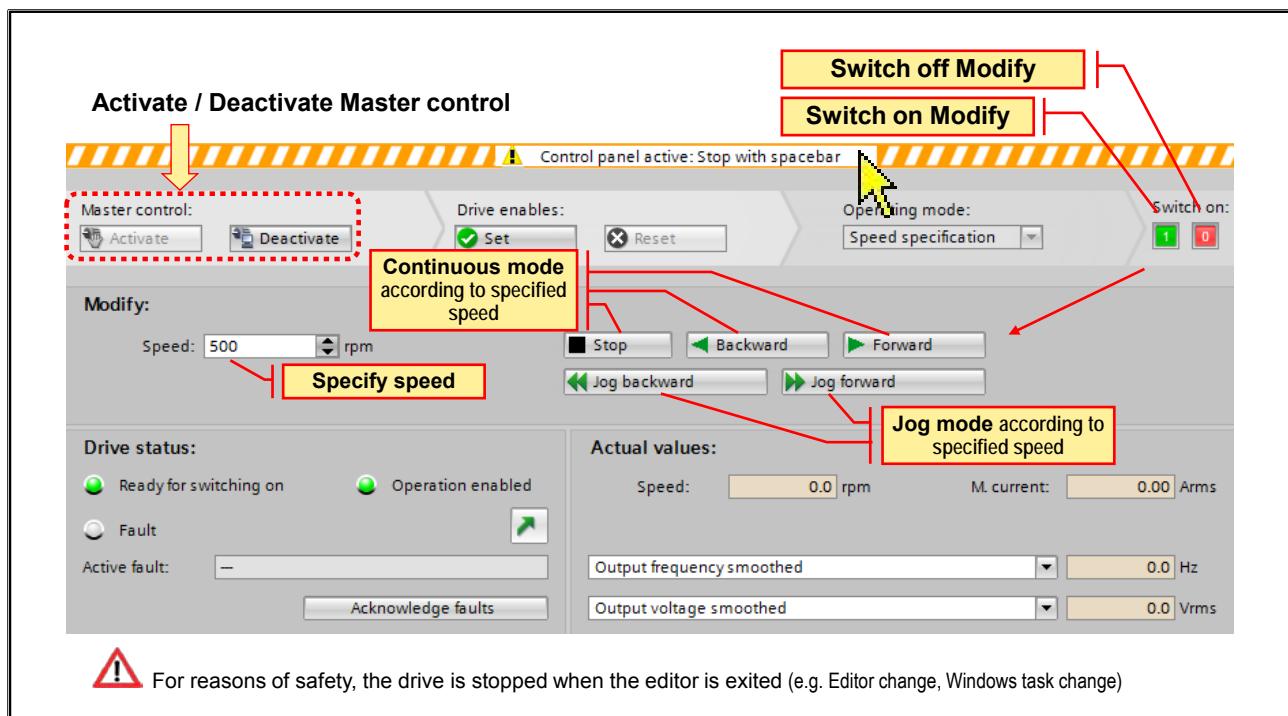


Caution!

Activation terminates a possibly existing connection to the CPU and gives the operator full control of the drive. Therefore it is important to have at your disposal a hardware solution for an Emergency Stop.



16.5.1. Operating the Control Panel for Commissioning



If the Master control is "switched to the PG" you can switch on "Modify".

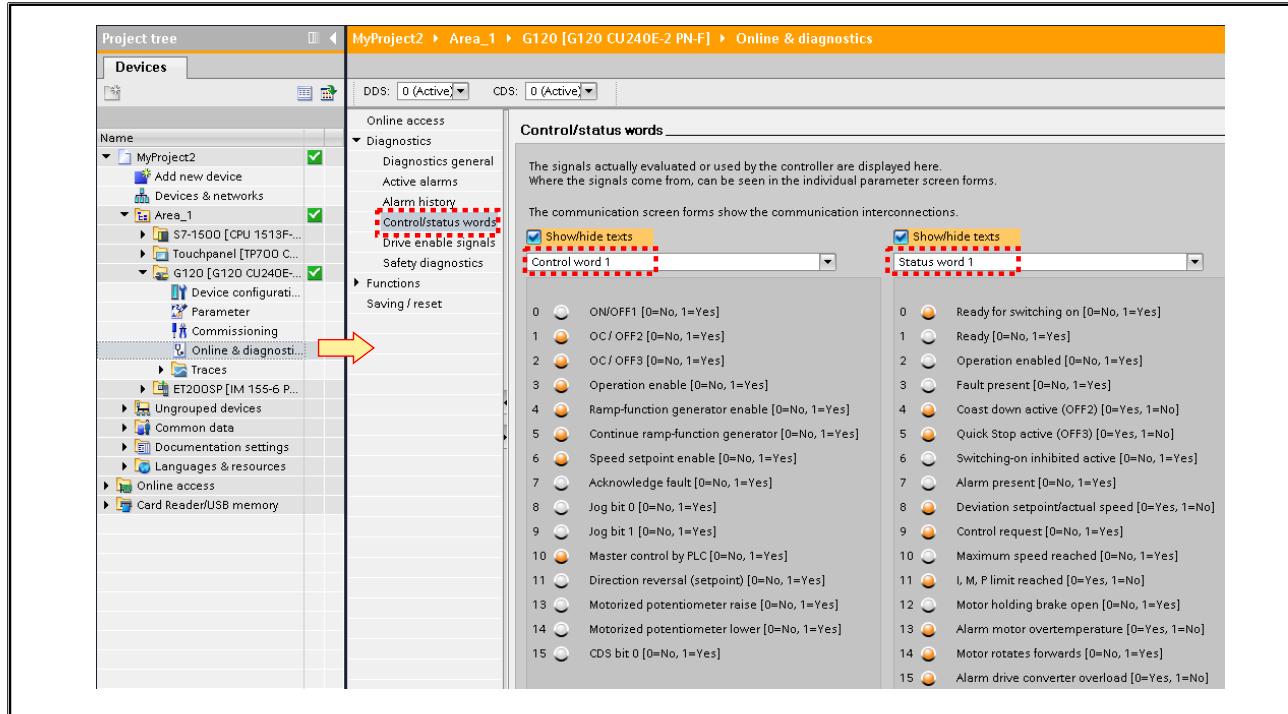
Operation is enabled and the buttons for modifying are active.

Safety Shutdown

The drive is stopped and...

- operating enables are cancelled,
when the "Commissioning" editor is exited through a Windows task change.
⇒ **Set**
- the Master control is deactivated with a fault message,
when a switch is made to the editor of another device.
⇒ **Acknowledge faults** and **Activate**

16.5.2. Monitoring Control and Status Word(s) ONLINE

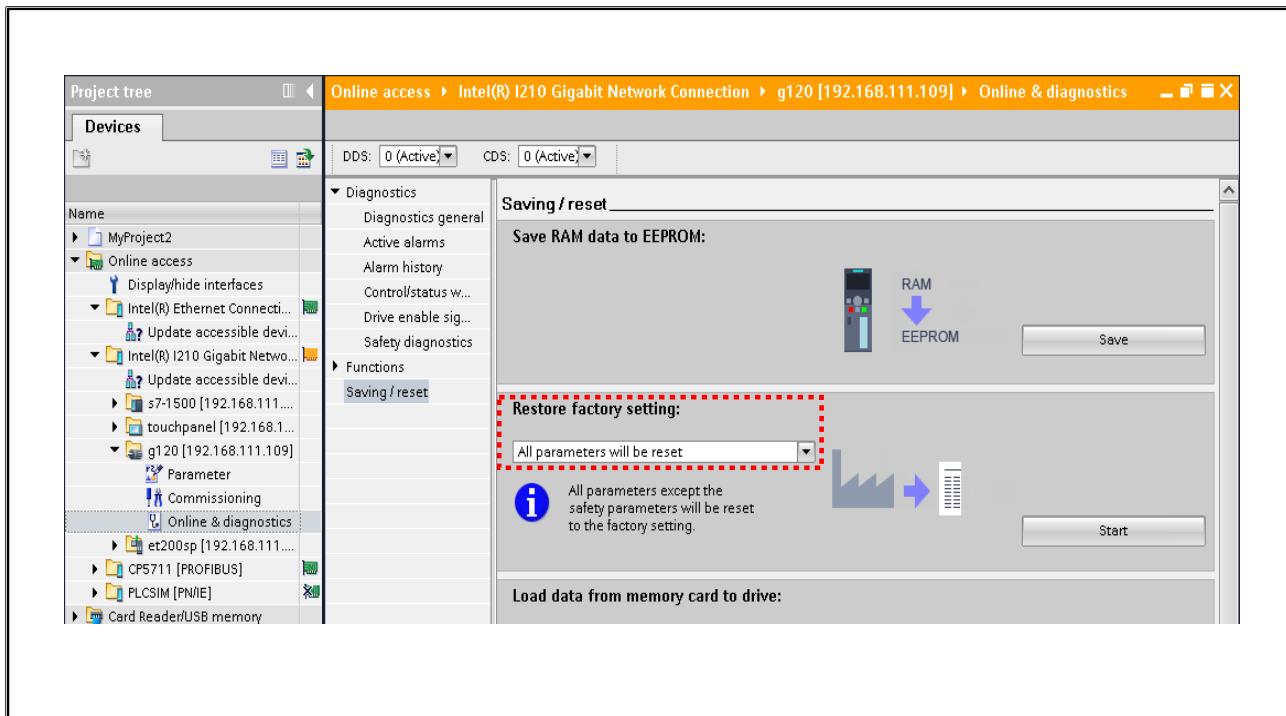


Monitoring Control and Status Word(s)

The Control and Status words dialog displays the different control and status words for diagnostics.

The signals actually used by the inverter are indicated by LED lights. The display is only relevant if the drive is controlled via the fieldbus.

16.6. Exercise 1: Resetting to Factory Settings



Task

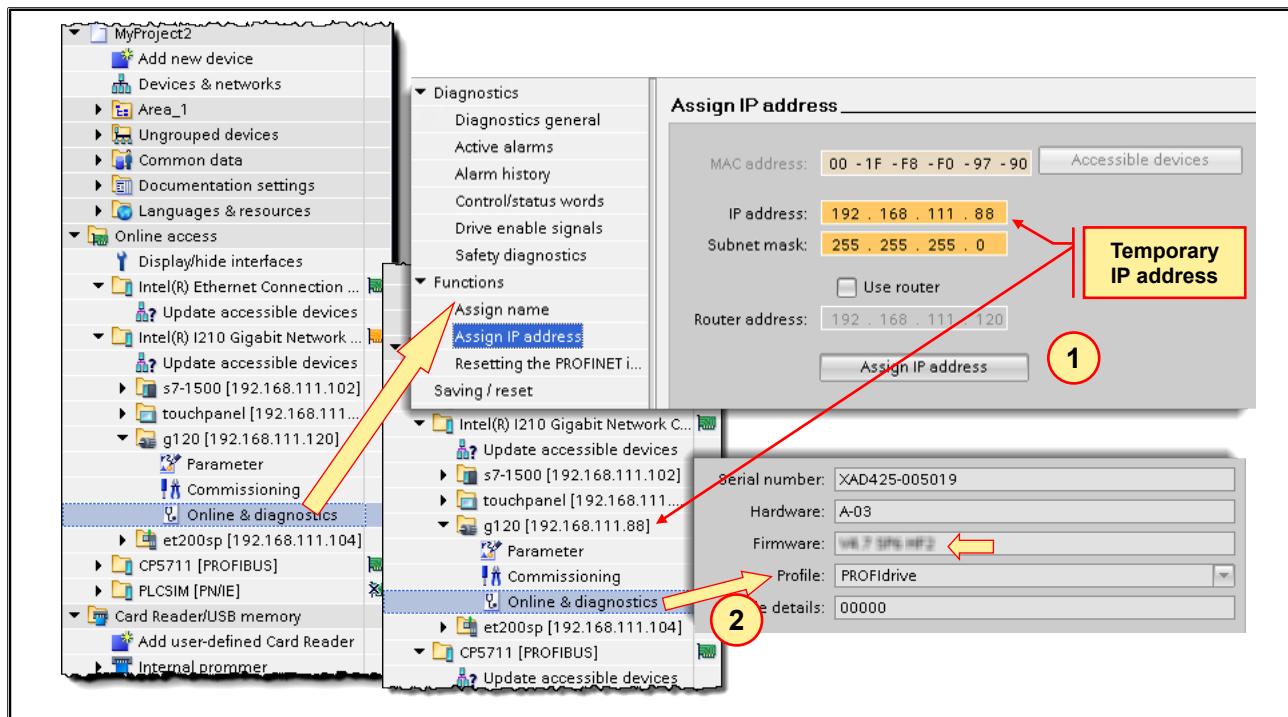
The drive is to be reset to its factory settings.

With the "Restore factory settings" function, the PROFINET device name and the IP address are also deleted.

What to Do

1. Connect a LAN cable between the drive and the PROFINET network of the training unit.
2. Under the Online access, select the relevant interface and update the list "Update accessible devices".
3. Open the drive and there activate the function "Online & diagnostics".
4. Give the interface of the drive an IP address, if one hasn't been assigned.
5. In the window that appears on the right, select "All parameters will be reset" for the function "Restore factory setting:" and Start the function.
6. Close the window.

16.6.1. Exercise 2: Reading-out the Firmware Version of the Drive



Task

In order to be able to configure the drive in the project in the next exercises, you must know the firmware version of the training unit. Since the firmware of the drive can be upgraded, the version of the firmware specified on the outside does not necessarily reflect reality. For that reason, the current version of the firmware is now to be read out online from the drive.

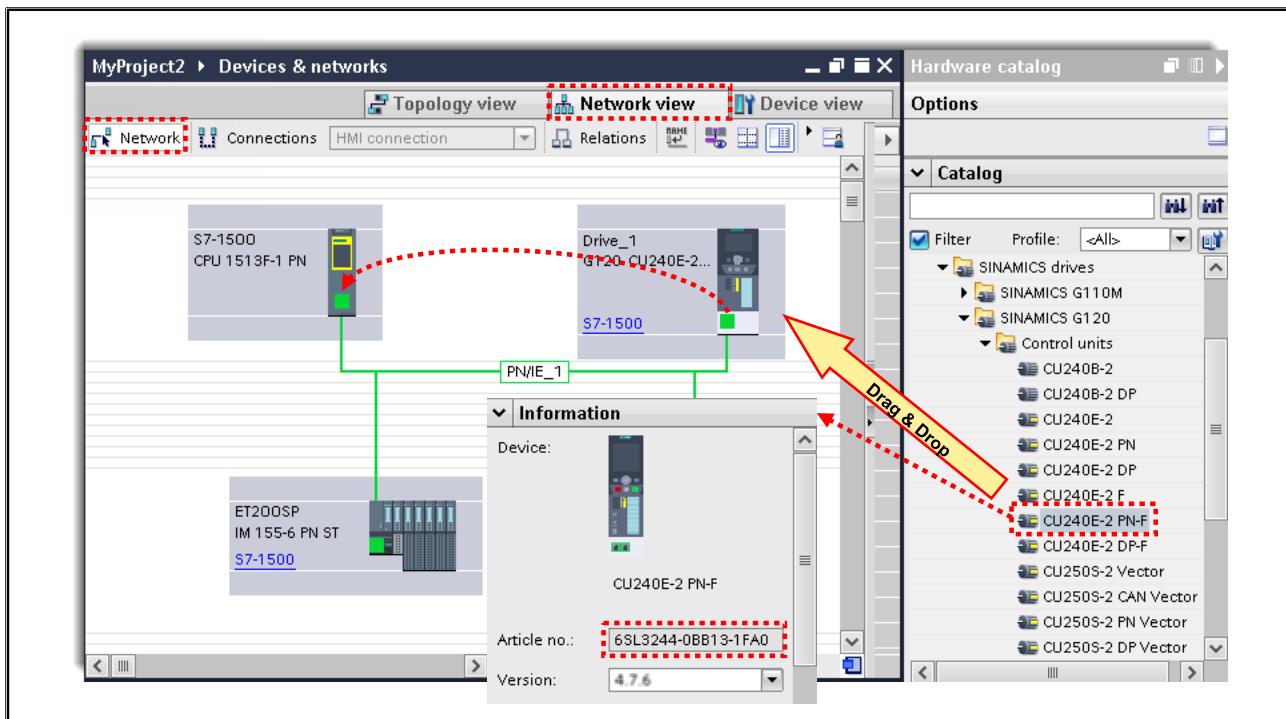
Problem

Should the drive only have a MAC address (see Project tree, to the left in the picture) and neither a PROFINET device name nor an IP address, the firmware version cannot be read out, since an IP address is required for this diagnostic service.

What to Do

1. In case the drive does not yet have an IP address:
Assign a temporary IP address as shown in the picture and update the list "Update accessible devices".
2. In the list of "accessible devices", open the drive and activate "Online & diagnostics".
3. In the work area, activate "Diagnostics -> General diagnostics" and make note of the firmware version of the inverter.
4. Close the window.

16.6.2. Exercise 3: Inserting and Networking the Drive in the Offline Project



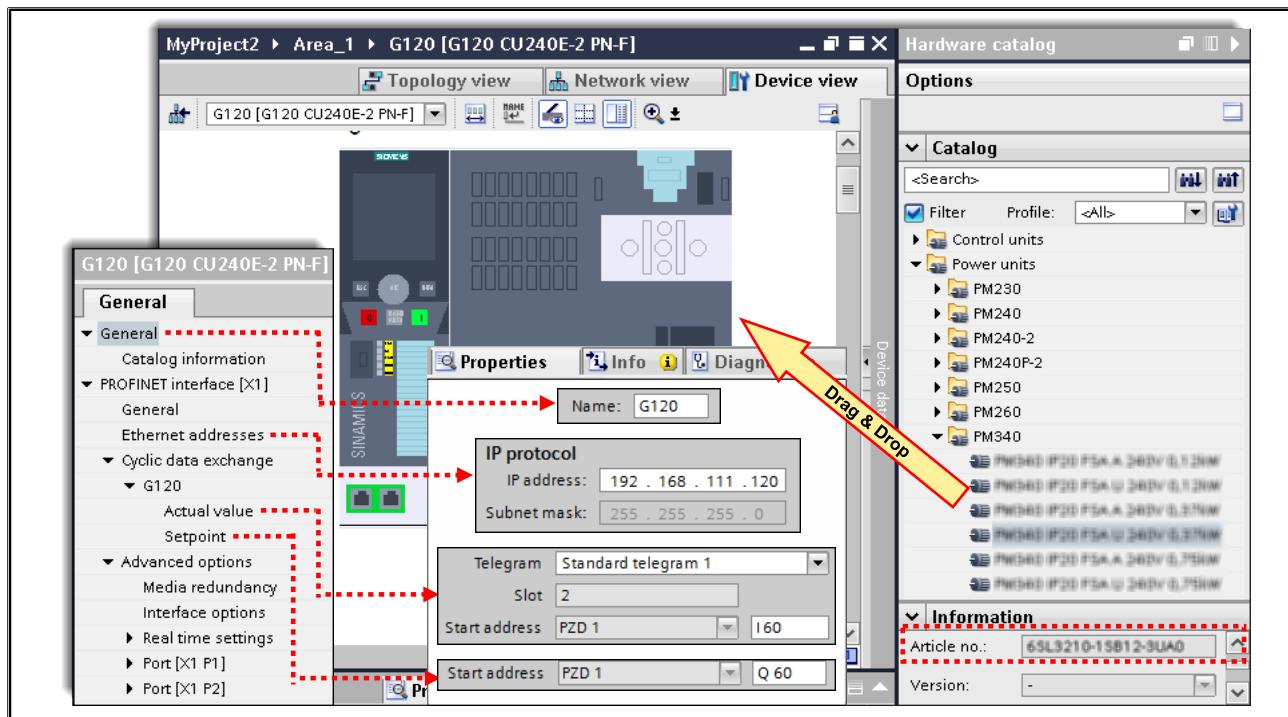
Task

The drive is to be inserted in the project and networked with the CPU via PROFINET.

What to Do

1. Open the "Hardware & Networks" editor and there activate the Network view.
2. In the Task Cards, open the Hardware catalog.
3. Insert the G120 drive by dragging the CU240E-2PN-F control unit into the Network view using drag & drop.
Pay attention to the Article no (order number) and the Version in the "Information".
4. Network the drive with the CPU by dragging the PROFINET interface of the drive to the PROFINET interface of the CPU using drag & drop.
5. Save your project.

16.6.3. Exercise 4: Parameterizing the Drive Communication



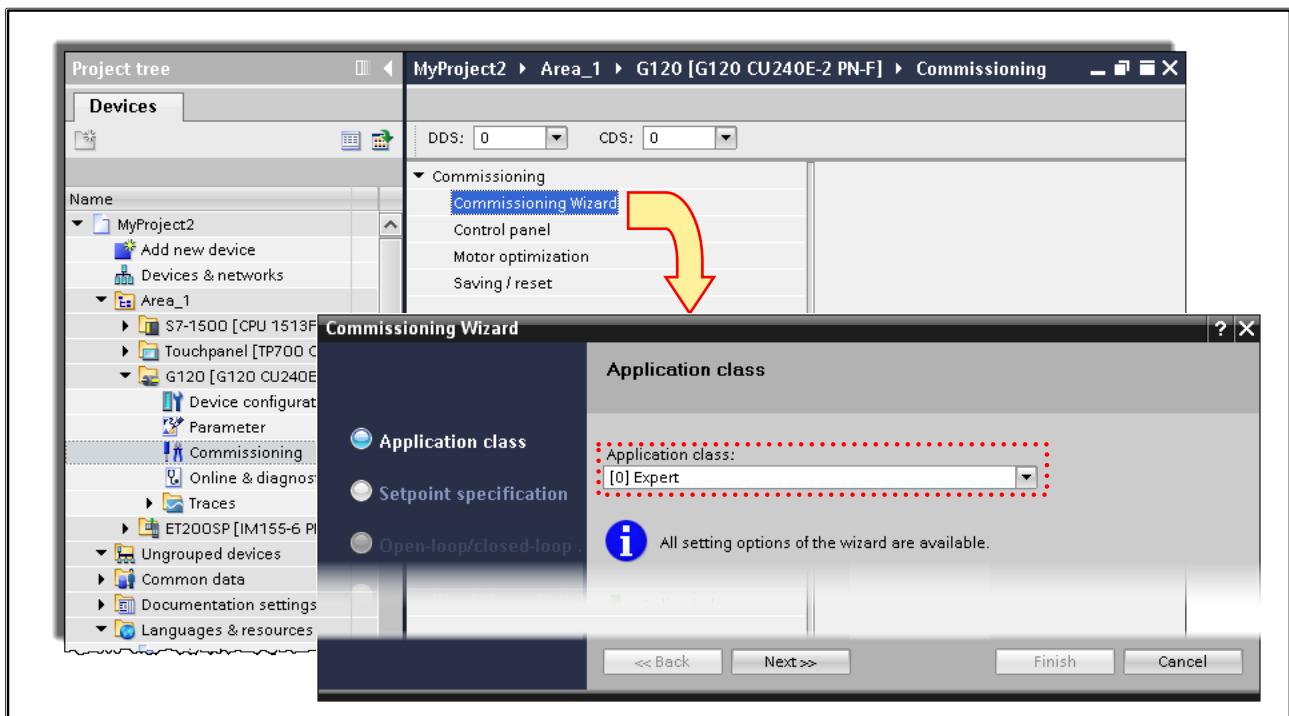
Task

So that the configuration of the drive is complete, a power unit still has to be added. Furthermore, the control unit still has to be parameterized.

What to Do:

1. Open the Device configuration of the drive.
2. Open the Hardware catalog and, using drag & drop, add the power unit. Pay attention to its Article no (order number) and Version in the "Information".
3. Select the control unit and, in the Inspector window under "Properties", parameterize the module with the parameters shown in the picture.
4. Save your project.

16.6.4. Exercise 5: Parameterizing the Drive OFFLINE with the Commissioning Wizard



Task

OFFLINE, you are to set the most important drive parameters and use the "Commissioning Wizard" to do this.

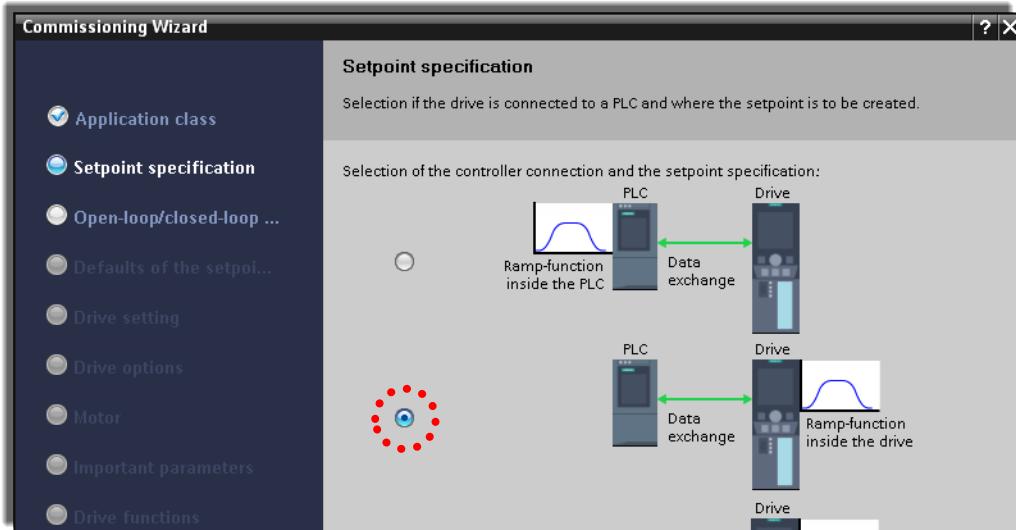
What to Do

1. Start the Commissioning Wizard as shown in the picture. Implement the parameterization as shown in the following.

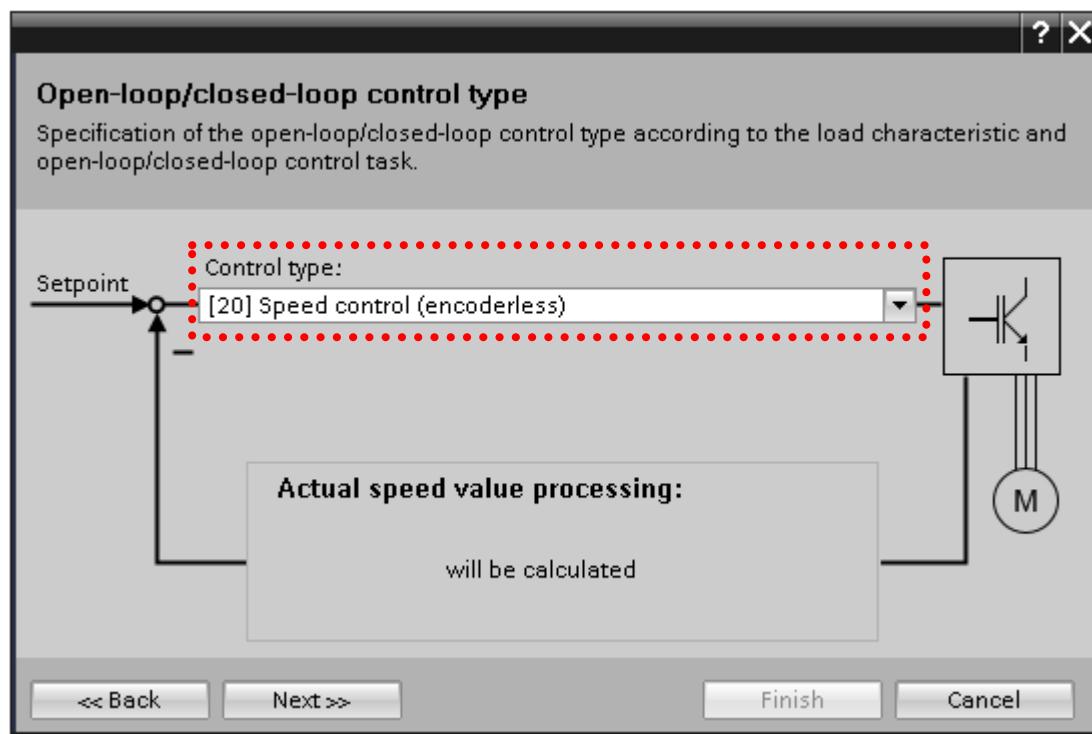
Note if Firmware of the drive is lower than 4.7.3

For a Firmware of the drive that is lower than 4.7.3., the Commissioning Wizard starts with the setting "Setpoint specification".

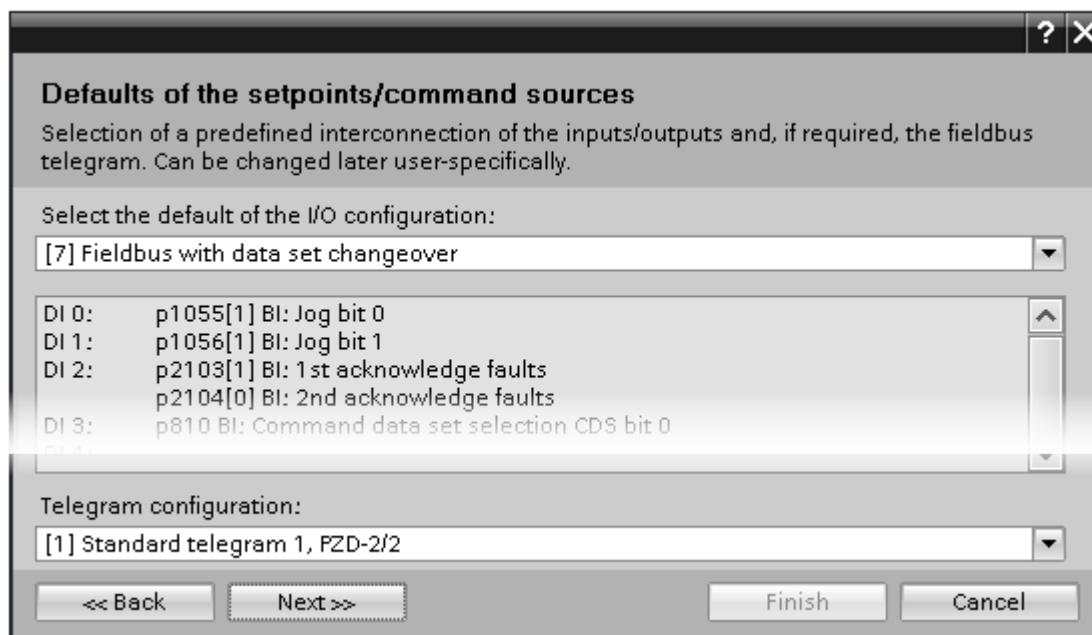
- Setpoint specification



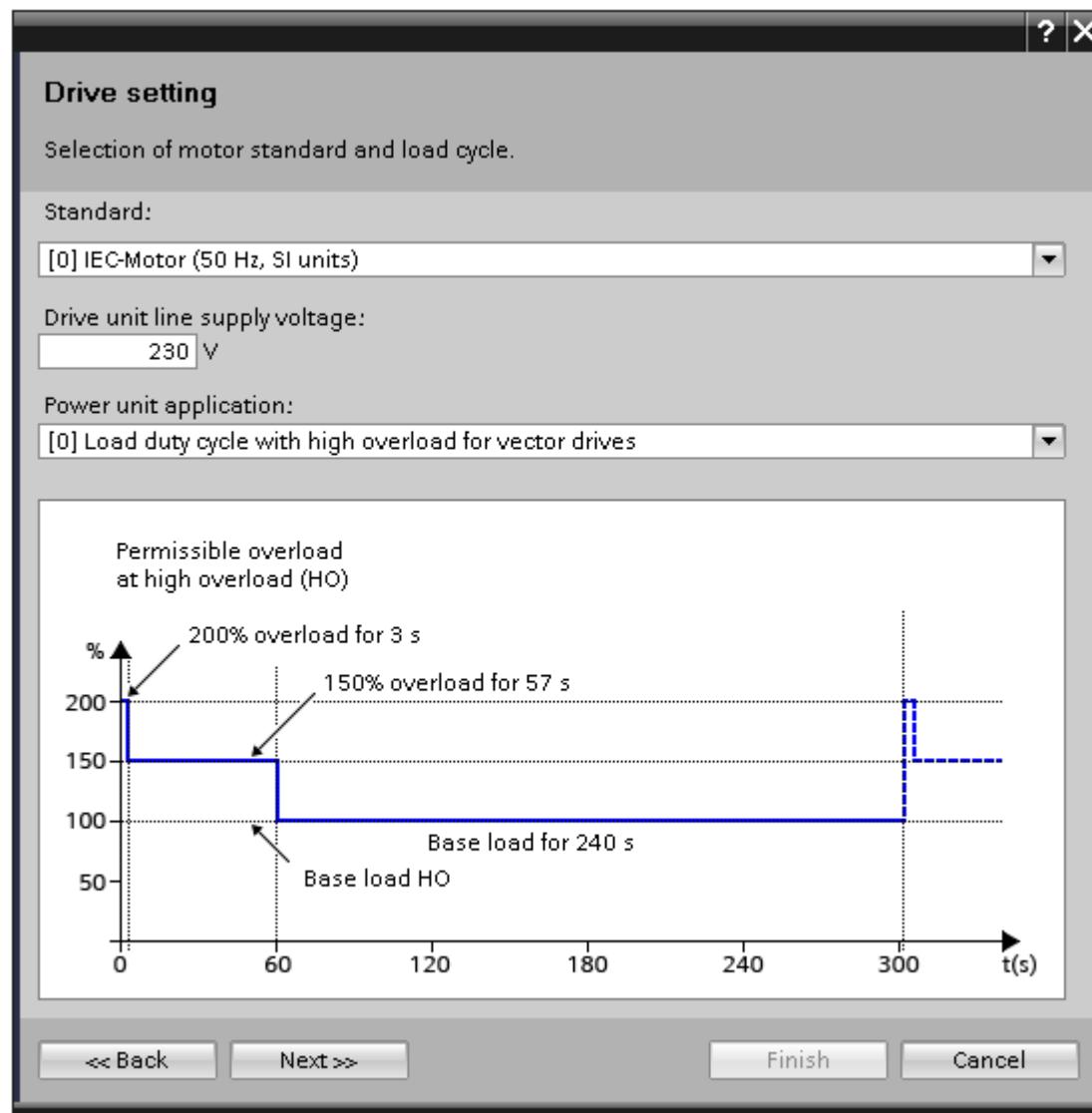
- Open-loop/closed-loop control type



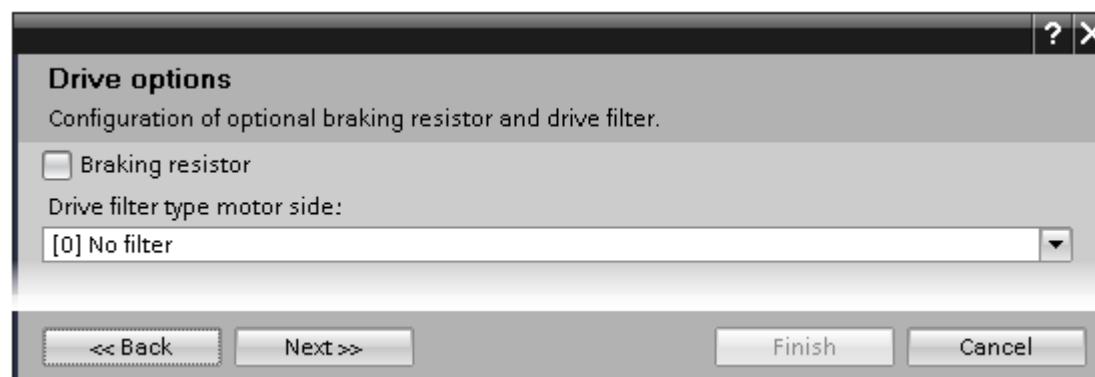
- Defaults of the setpoints / command sources



- **Drive setting**

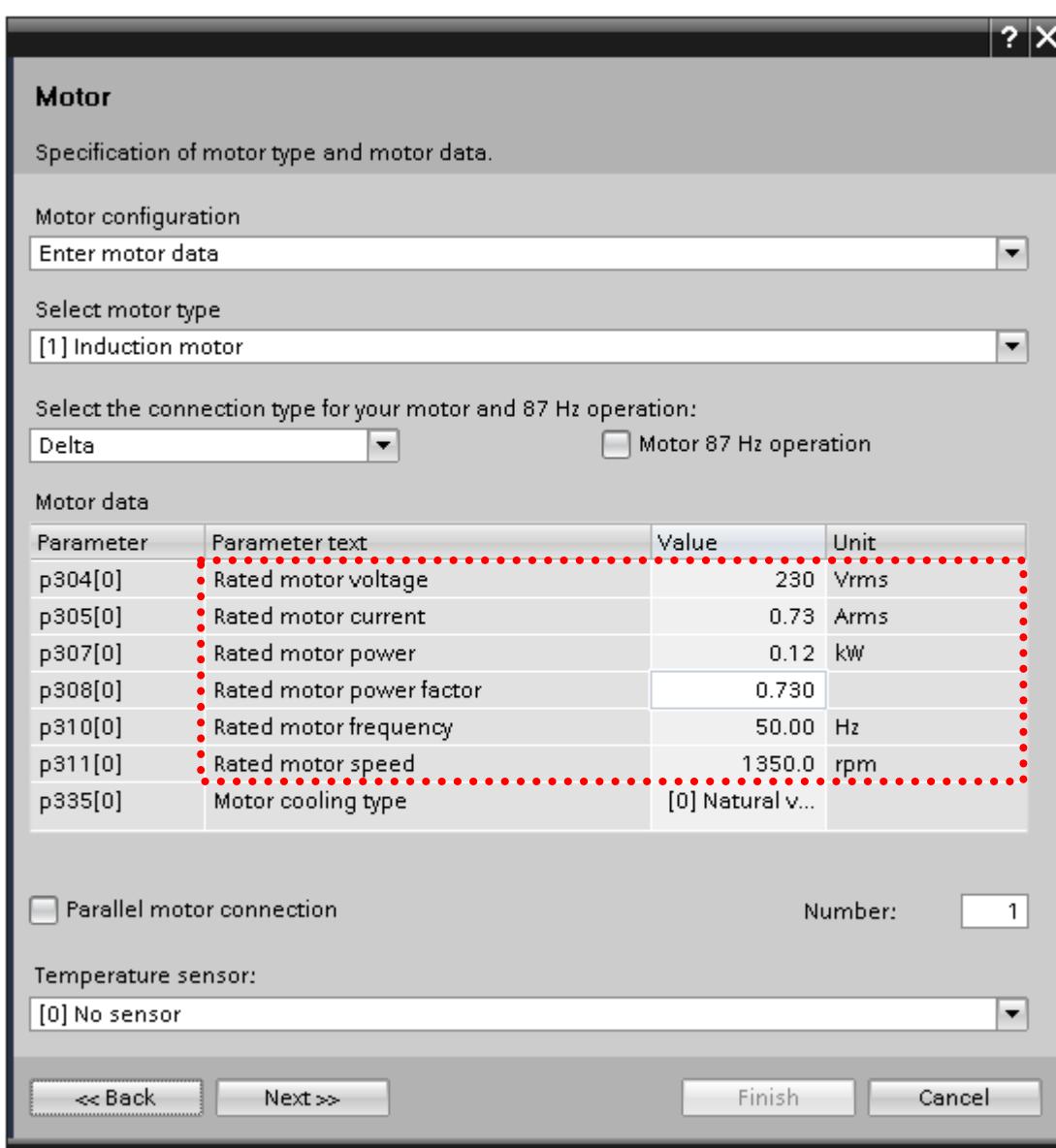


- **Drive options**

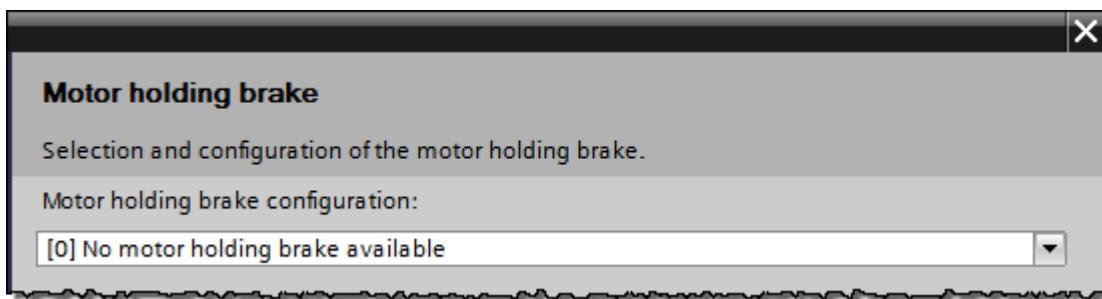


- Motor

Take the motor data from the motor's nameplate and enter it in the Motor dialog:

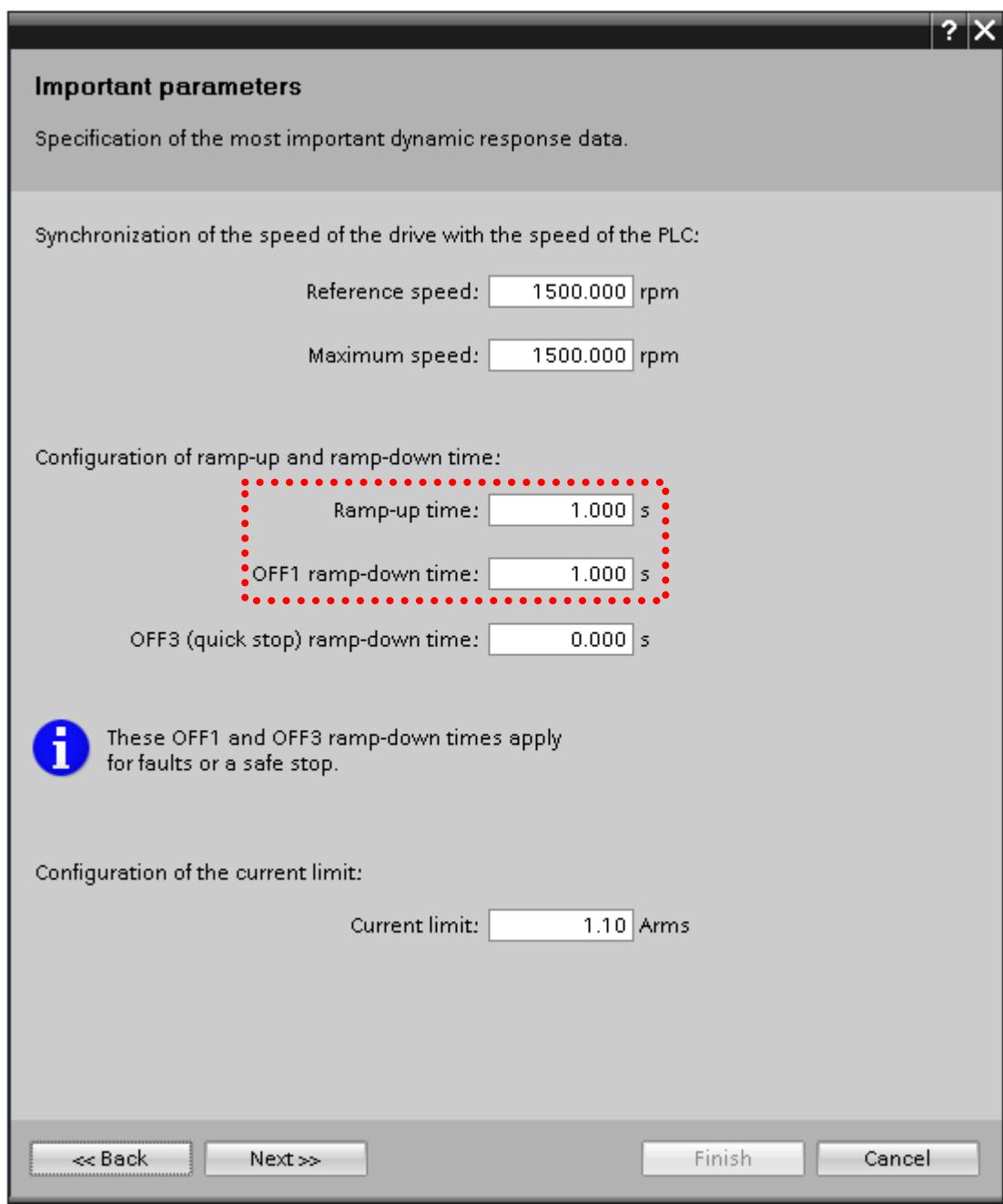


- Motor holding brake

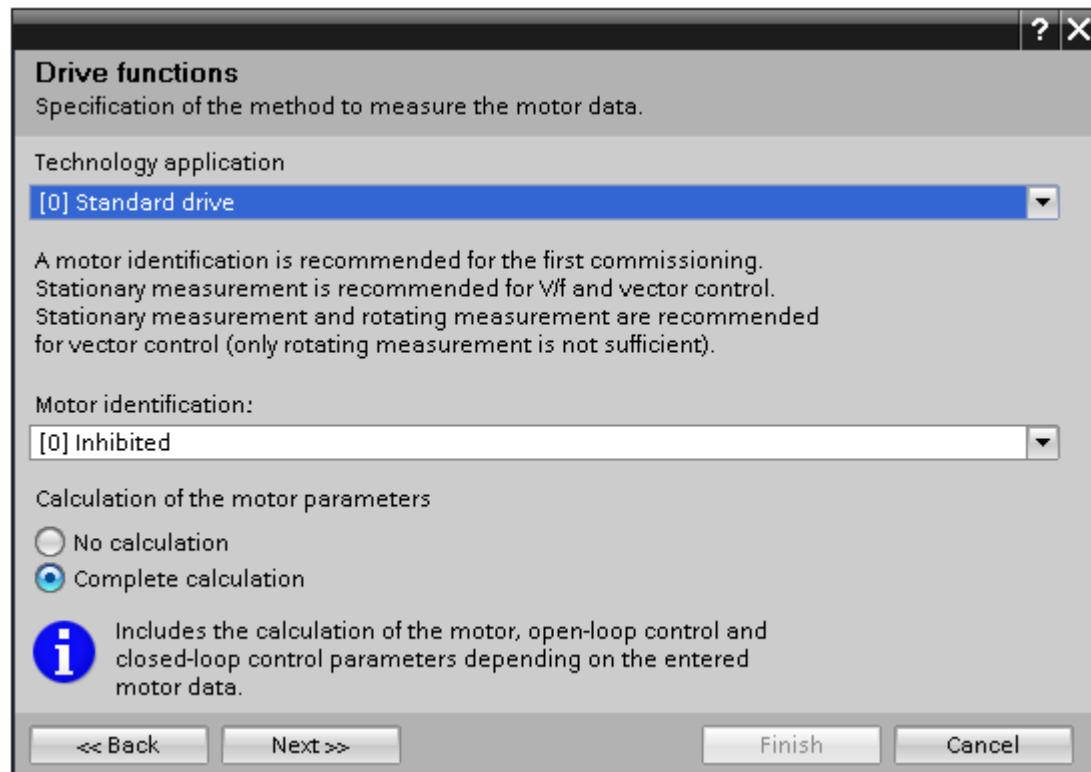


- Important parameters

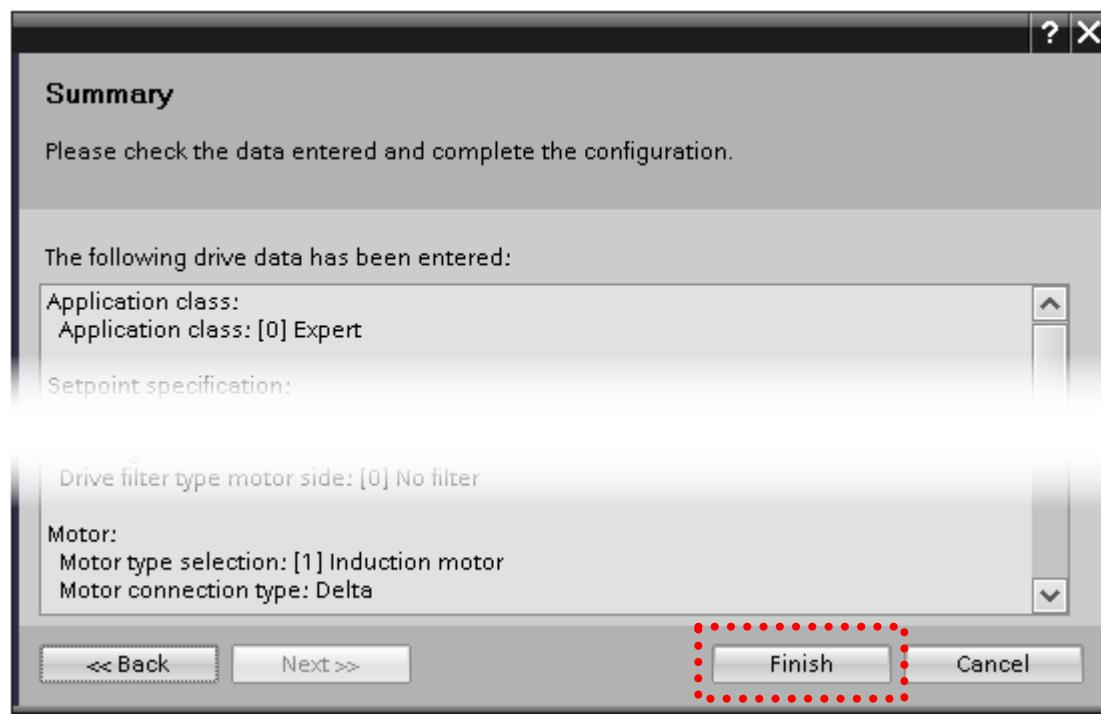
Change the ramp-up and ramp-down times:



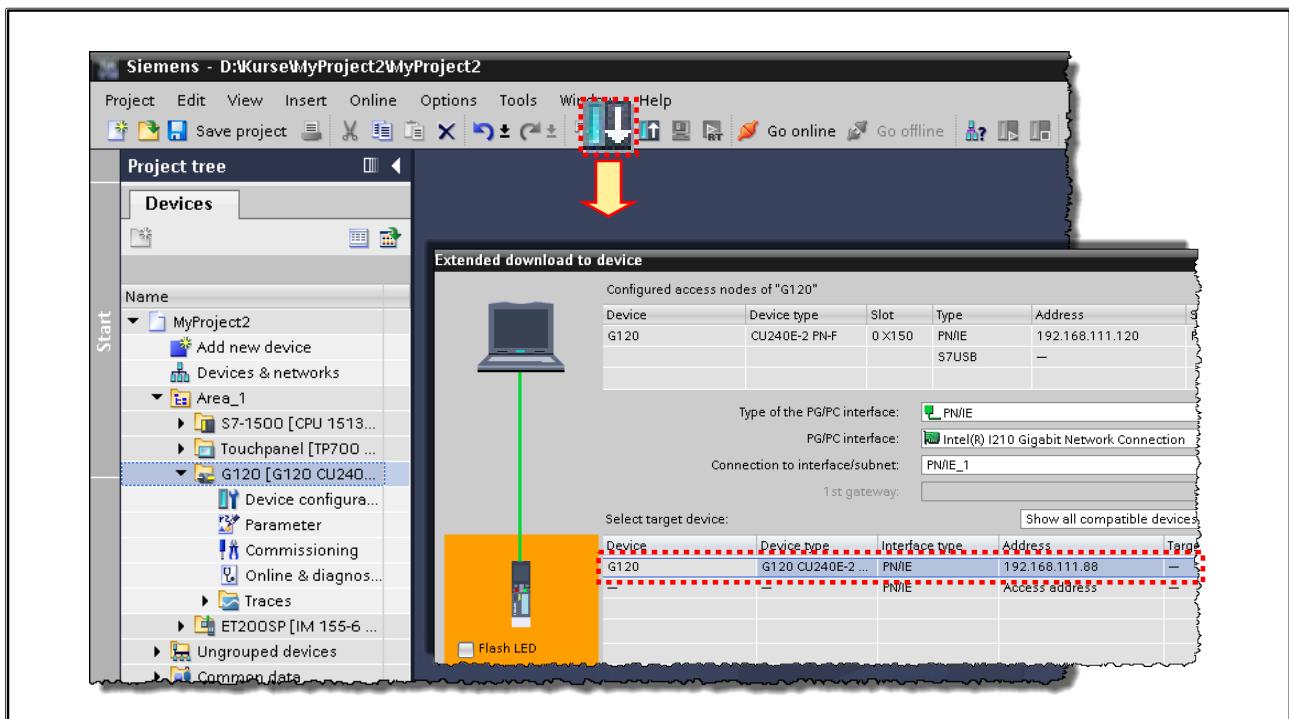
- **Drive functions**



- **Summary**



16.6.5. Exercise 6: Downloading Drive Parameterization to the Drive and Hardware Configuration to the CPU



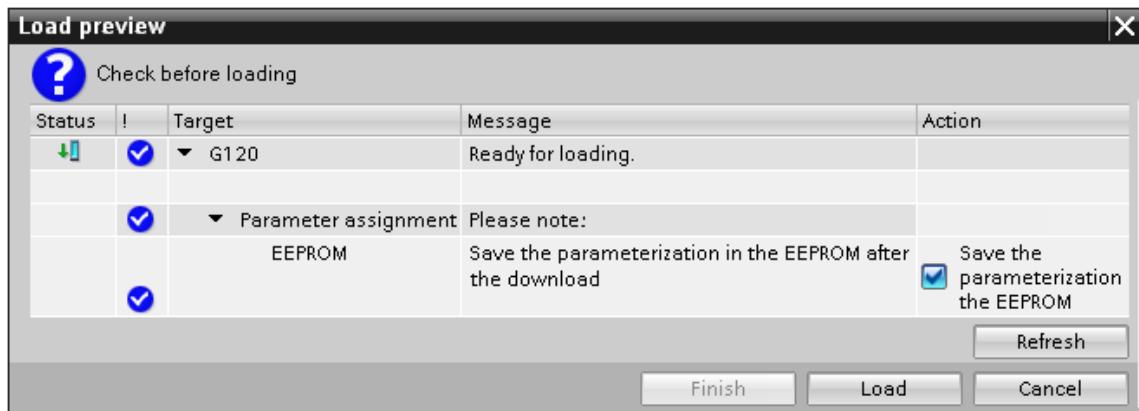
Task

Now that you have implemented the basic parameterization offline, it is now to be downloaded into the drive online. In addition, the hardware change of the CPU is also to be downloaded.

What to Do

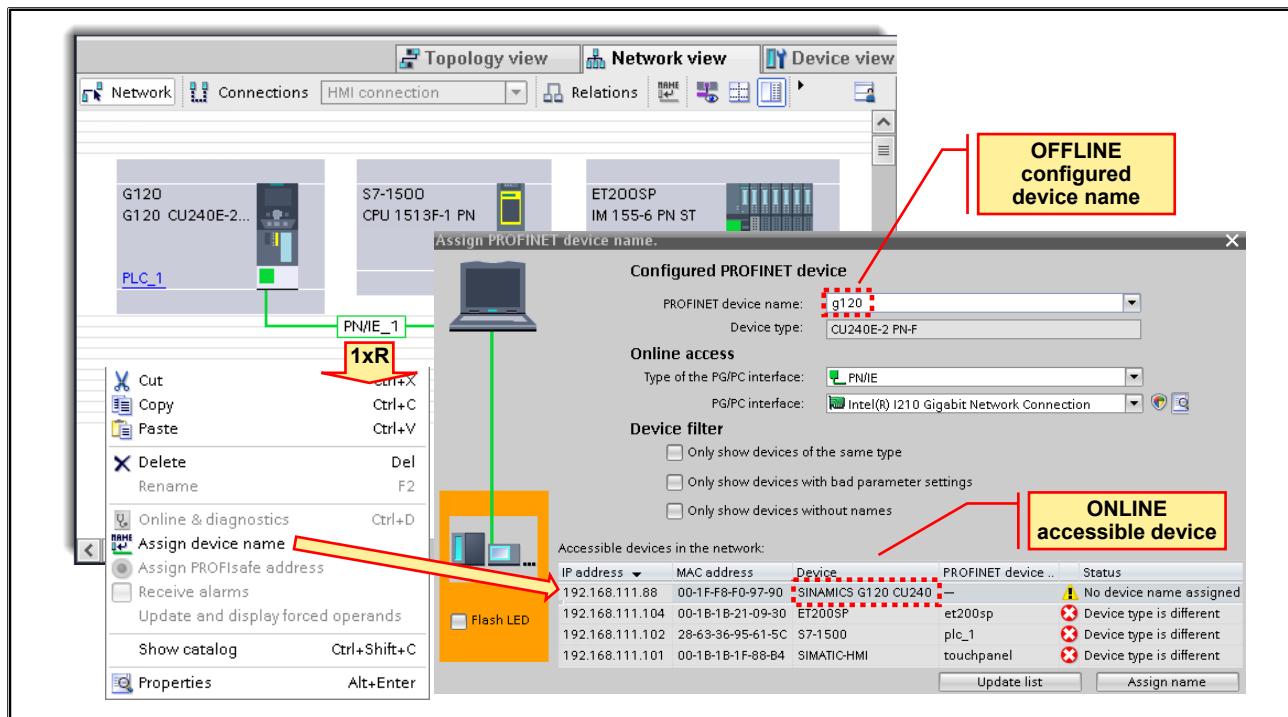
1. In the Project tree, select the drive and start the download of the hardware configuration and parameter assignment using the button "Download to device" (see picture).
2. The dialog "Extended download to device" appears which contains the list of compatible devices in the target subnet in the lower part. Should the drive of your training area not be listed there, activate the option "Show all compatible devices" (see picture).
3. In the list "Select target device", select the drive of your training area and start the "Download".

So that the drive doesn't lose the parameterization even after a power failure, answer the following dialog as follows:



4. Also download the hardware configuration of the SIMATIC station once more into the S7-1500 so that the G120 is entered (listed) as a new IO-Device.

16.6.6. Exercise 7: Assigning the PROFINET Device Name ONLINE



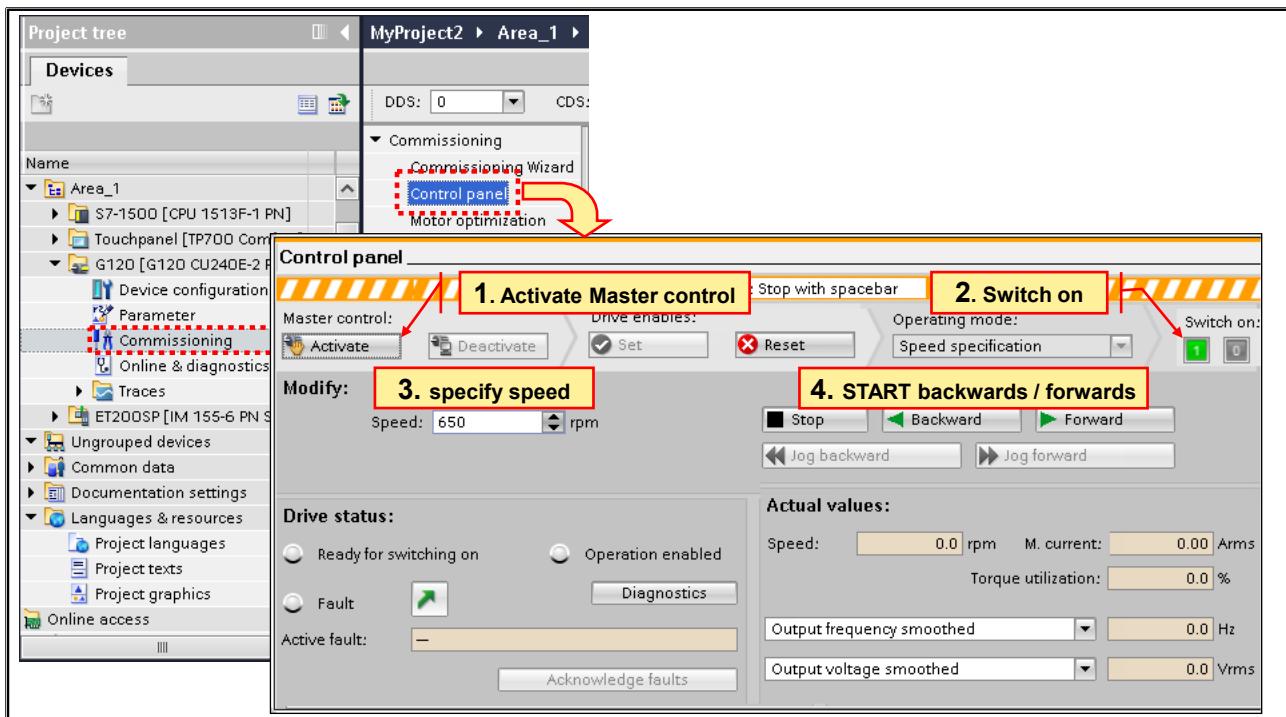
Task

ONLINE, assign the OFFLINE configured PROFINET device name and the IP address to the drive.

What to Do

1. Open the "Hardware & Networks" editor and there activate the Network view.
2. Select the subnet (see picture) and via the right mouse button, activate the function "Assign device name"
3. In the dialog that follows, complete it as shown in the picture and assign the name.
4. Check whether the name assignment was successful by "Updating the accessible devices" under the Online access.
5. Save your project.

16.6.7. Exercise 8: Operating the Drive via the Control Panel



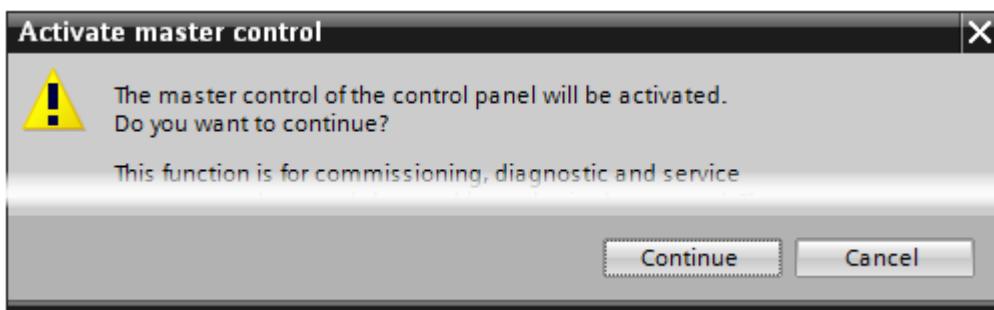
Task

Since the drive is already configured and has been assigned an IP address by the Controller, it can be operated via the Control panel.

What to Do

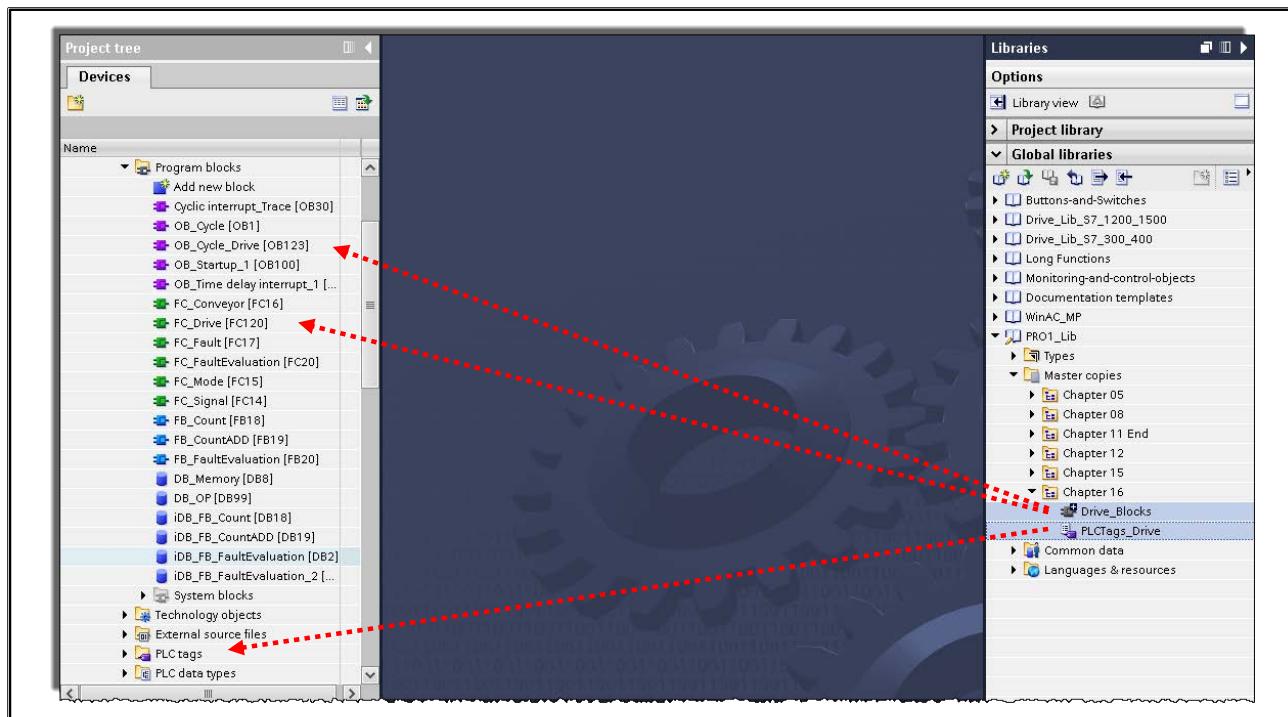
For controlling the drive open the "Control panel" and follow the numbers shown in the picture:

1. Activate the "Master control" (see picture) and acknowledge the dialog that appears with "Continue".



2. Switch on the drive.
3. Specify a speed.
4. To start the motor use one of the two buttons "Forward" or "Backward" and let the motor move with different speeds.
5. Switch the motor off and deactivate the master control.
6. Close the "Commissioning" editor and disconnect the online connection to the

16.6.8. Exercise 9: Commissioning "FC_Drive"



Task

Imagine that the G120 serves as the drive for the conveyor model when parts are transported from Bay1 or 2 automatically through the light barrier.

Accordingly, the G120 must be switched on and off parallel to the conveyor model motor when "P_Operation" (Q0.1) is switched on. The speed can be set using the right slide control potentiometer.

The described function is already implemented in the "FC_Drive" block and is now to be copied from a library and commissioned.

What to Do

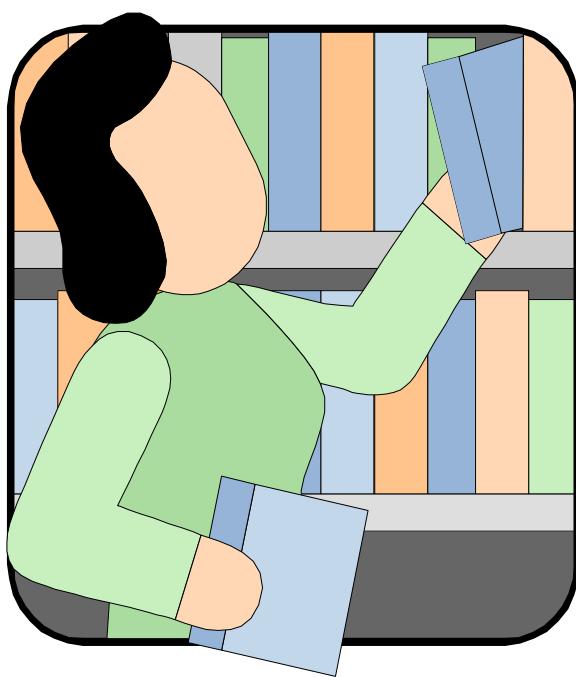
1. Open the library C:\02_Archives\TIA_Portal\TIA-PRO1\PRO1_Lib.
2. Using drag & drop, copy the library elements "Drive_Blocks" and "PLCTags_Drive" (tag table) into your project.

Note:

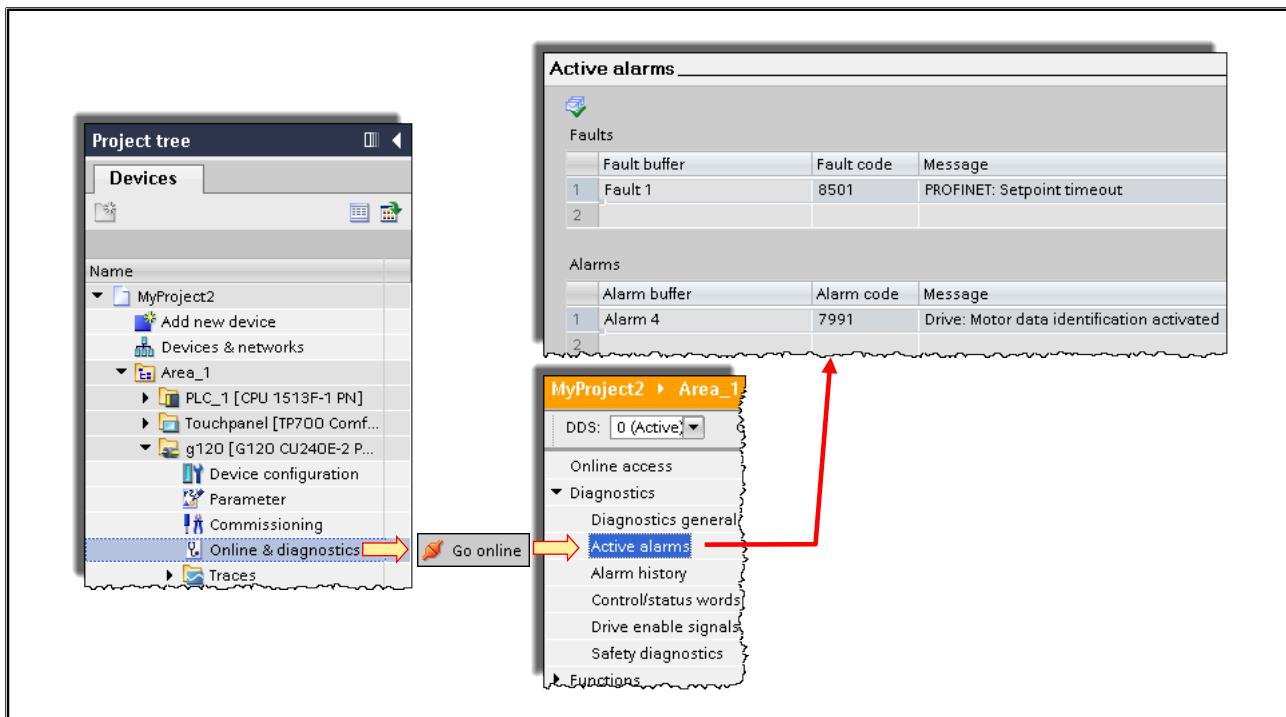
Since the object "Drive_Blocks" contains the FC_Drive and a "Program cycle OB" which calls the FC, the new program can be loaded directly.

3. Download all blocks into the CPU and check whether the G120 is switched on and switched off parallel to the conveyor model motor when "P_Operation" (Q0.1) is switched on.
4. Check whether the speed of the G120 can be changed using the right slide control potentiometer on the simulator.
5. Save your project.

16.7. Additional Information



16.7.1. Monitoring Active Alarms ONLINE



Description

"Active alarms" displays the currently pending alarms and faults.

- For a fault, the parameterized fault reaction is initiated and the status signal word 1.3 is sent. In addition, the fault is entered in the Fault buffer. Faults must be acknowledged after the cause is eliminated.
- For an alarm, the status signal word 1.7 is set. In addition, the alarm is entered in the Alarm buffer.

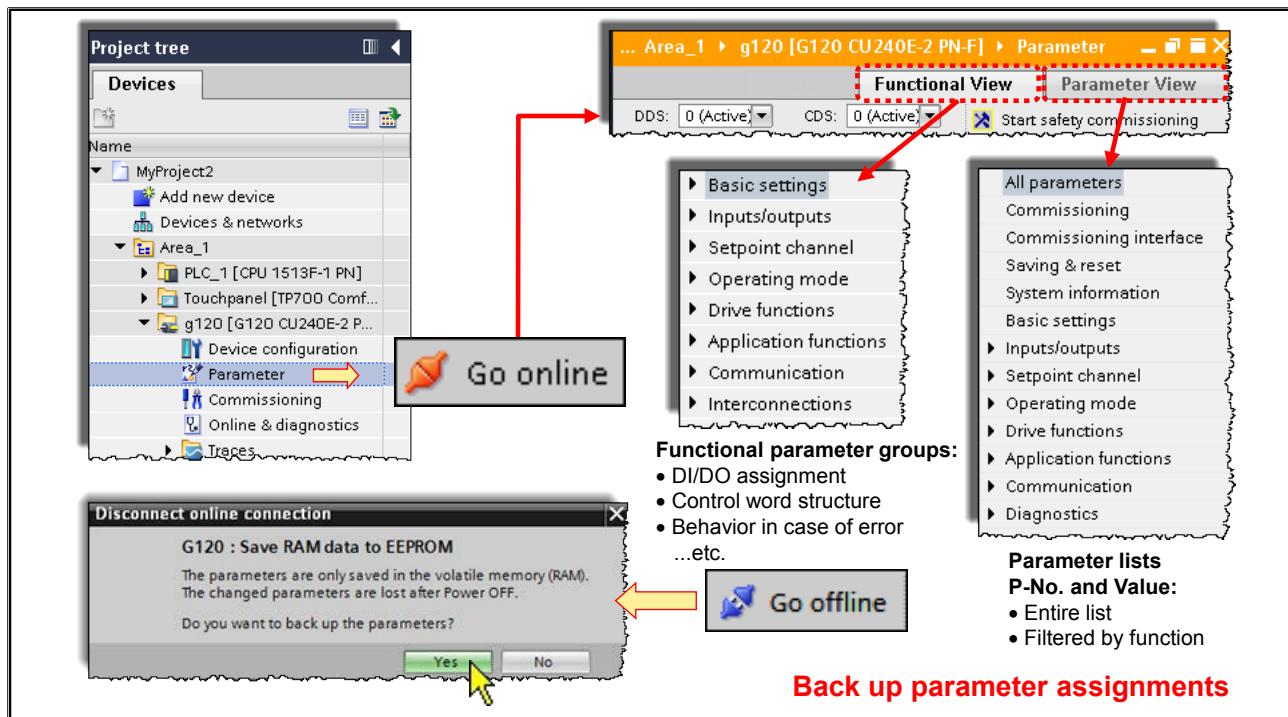
Faults and Alarms

Fault code / Alarm code	Number of the fault / alarm
Message	Description of the fault / alarm
Fault time / Alarm time	Time stamp that shows when the fault / alarm occurred.

Acknowledging Faults

Click on the  button in order to acknowledge all active faults. Faults and alarms are moved to the Alarm history after they have been acknowledged.

16.7.2. Changing Parameters in the Inverter



Parameters can be changed through the "Parameter" editor.

Calling the "Parameter" Editor from the Device in the Project Tree

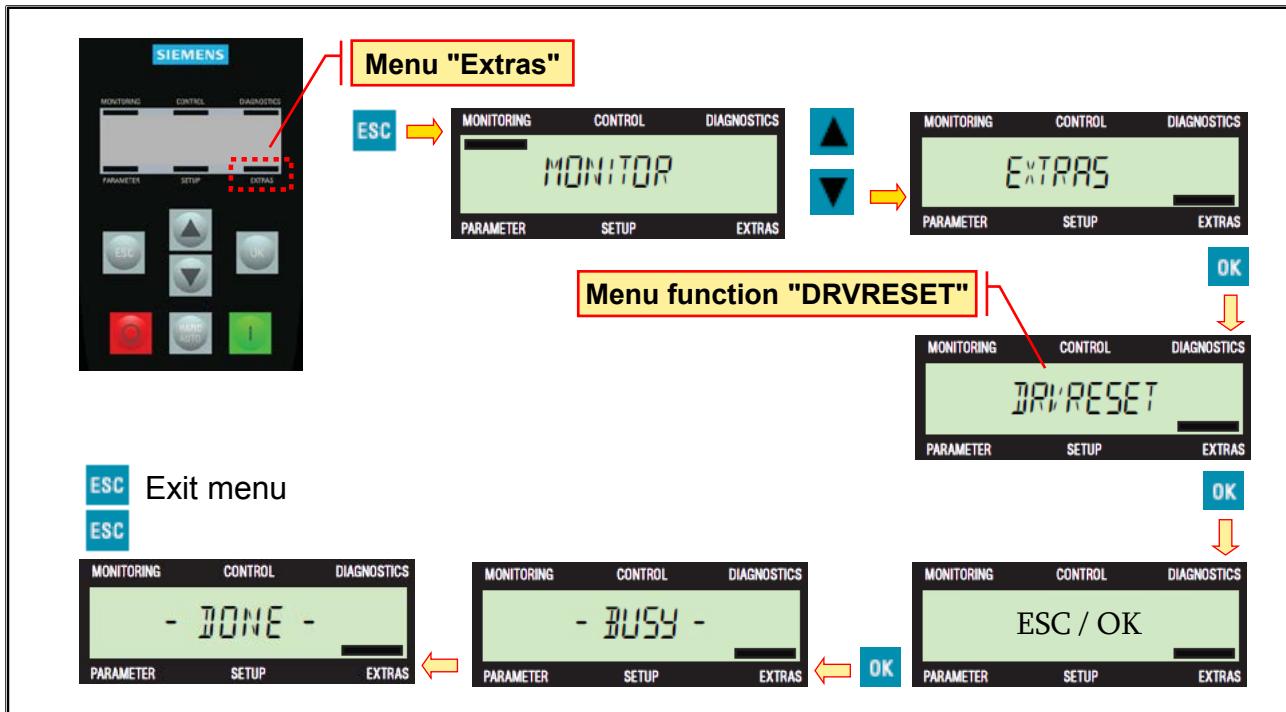
When the online connection is to be terminated, a prompt comes up questioning whether the data is to be copied to EEPROM (nonvolatile backup of the parameter changes).

If the change is only to be tried for test purposes, you can answer the query for copying with NO and after a PowerOFF/ON, the inverter then works with its previous values.

Calling the "Parameter" Editor from the Online Access

The change of these parameters is immediately written to the EEPROM.

16.7.3. G120 Reset to Factory Settings via BOP-2



17

Contents

17.	Training and Support	17-2
17.1.	Any Questions on our Training Courses Offered??.....	17-3
17.2.	www.siemens.com/sitrain	17-4
17.3.	Learning path: SIMATIC S7 Programming in the TIA Portal	17-6
17.4.	Download the training documents	17-7
17.5.	The Industry Online Support – the most important innovations.....	17-8
17.6.	The Principle of Navigation	17-9
17.7.	Complete product information.....	17-10
17.8.	mySupport – Overview.....	17-11
17.9.	Support Request	17-12
17.10.	Support Request	17-13
17.11.	Industry Online Support – wherever you go	17-14
17.11.1.	Scanning product/EAN code.....	17-15
17.11.2.	Scan functionality	17-16
17.12.	Forum - the communication platform for Siemens Industry products.....	17-17
17.12.1.	Conferences and Forum management	17-17
17.12.2.	Interactions in the Forum	17-19
17.13.	Task and Checkpoint	17-21

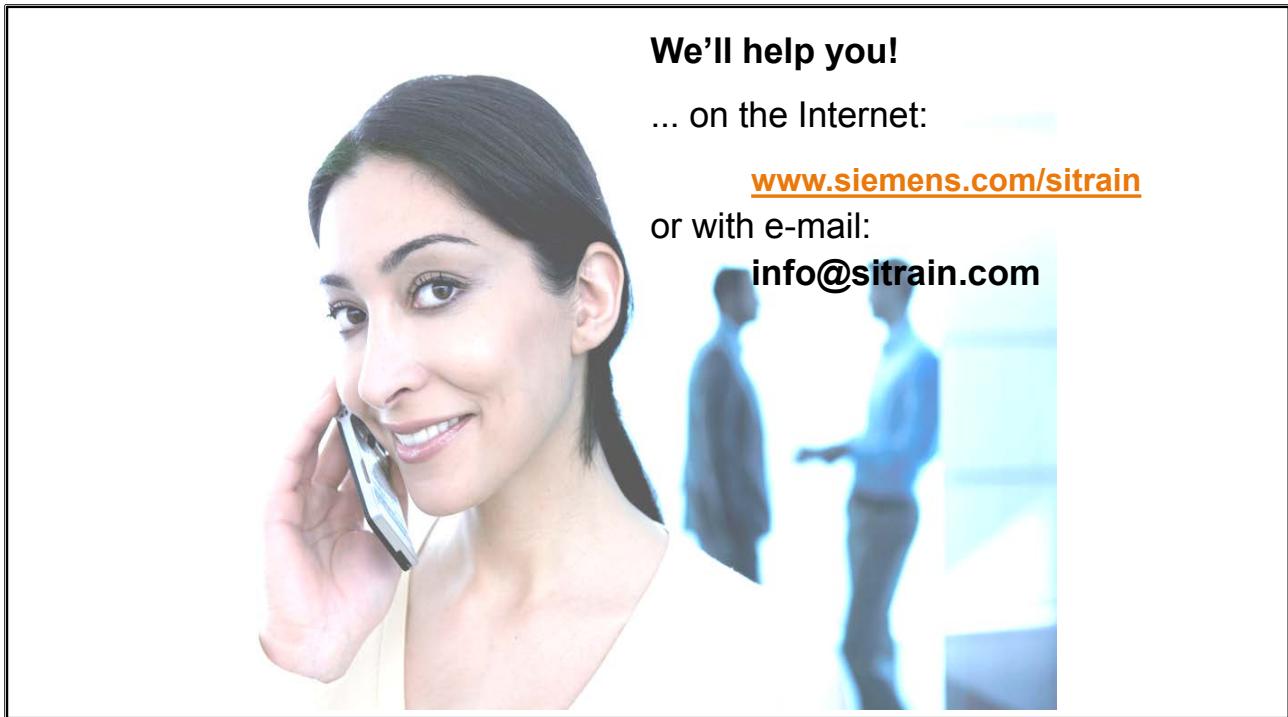
17. Training and Support

The image shows two screenshots of Siemens' training and support websites.

Left Screenshot: A screenshot of the SIMATIC TIA Portal Programming 1 course page. It features a banner with three people working on a computer. Below the banner, there's a navigation bar with links like "Home", "SITRAIN - Training for Industry", "My Training", and "Logout". The main content area is titled "SITRAIN - Training for Industry" and discusses training for industry. It includes sections for "Drive Technology", "Industrial Automation", "Energy", and "Building Technologies". Each section has a thumbnail image and a brief description.

Right Screenshot: A screenshot of the "Industry Online Support" portal. The top navigation bar includes "Logout", "Language", "Site Explorer", "Search in Online Support", and "mySupport". The main content area is titled "TIA Portal - Topic Page" and shows a grid of support articles. To the right, there are sections for "mySupport Cockpit" (with links to "Favorites", "Personal messages", "My requests", "My documents", "My Products", and "User online"), "Useful functions in the Online Support (videos)" (with a video thumbnail), and "Social Media" (with links to "Follow us on Twitter" and "Subscribe to Online Support").

17.1. Any Questions on our Training Courses Offered??



We'll help you!

... on the Internet:

www.siemens.com/sitrain

or with e-mail:

info@sitrain.com

General Information

We'll be glad to help you regarding any questions on our training courses offered.

17.2. www.siemens.com/sitrain

The complete range of courses offered can be accessed via the following links:

www.siemens.de/sitrain or

www.siemens.com/sitrain

Course Search

1

The course search permits the user to find the required courses by applying different search filters such as keyword, target group, etc. The filters can also be combined.

Course Catalog

The course catalog permits you to find the required course via learning paths or via the Siemens Mall structure.

Top Links

Various courses, e.g. SIMATIC S7-1500 solution line, etc., can be reached directly via the top links.

2

› Home > Industrial Automation > Automation Systems > SIMATIC Industrial Automation Systems

SIMATIC Industrial Automation Systems

Consistent and efficient



A centerpiece of our comprehensive range of products and services for industrial automation is SIMATIC, a unique, consistent system of first-class products for every field of application, in all industries. Regardless of whether it's manufacturing and process automation or solutions for infrastructure tasks: with SIMATIC we make an important contribution toward improving your productivity.

SITRAIN has a portfolio of training courses that are perfectly matched to your requirements and your plant's lifecycle.

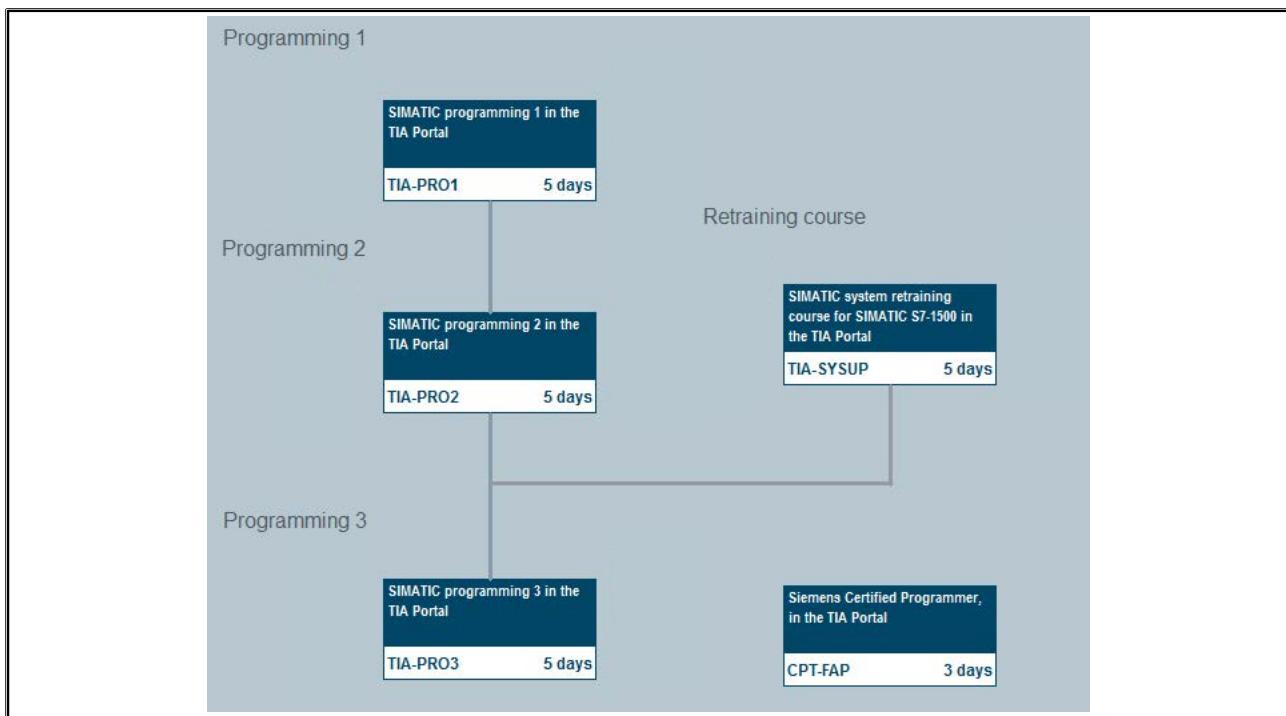
SIMATIC S7 TIA Portal

- › On the path to the digital enterprise - discover your potential with training courses for SIMATIC S7-1500 training in the TIA Portal
- › SIMATIC TIA Übersicht
- › SIMATIC S7 Programming in the TIA Portal
- › SIMATIC S7 Service Training in the TIA Portal
- › SIMATIC Safety Integrated in the TIA Portal
- › SIMATIC S7 Engineering Tools in the TIA Portal
- › SIMATIC Technology im TIA Portal
- › SIMATIC S7-1200

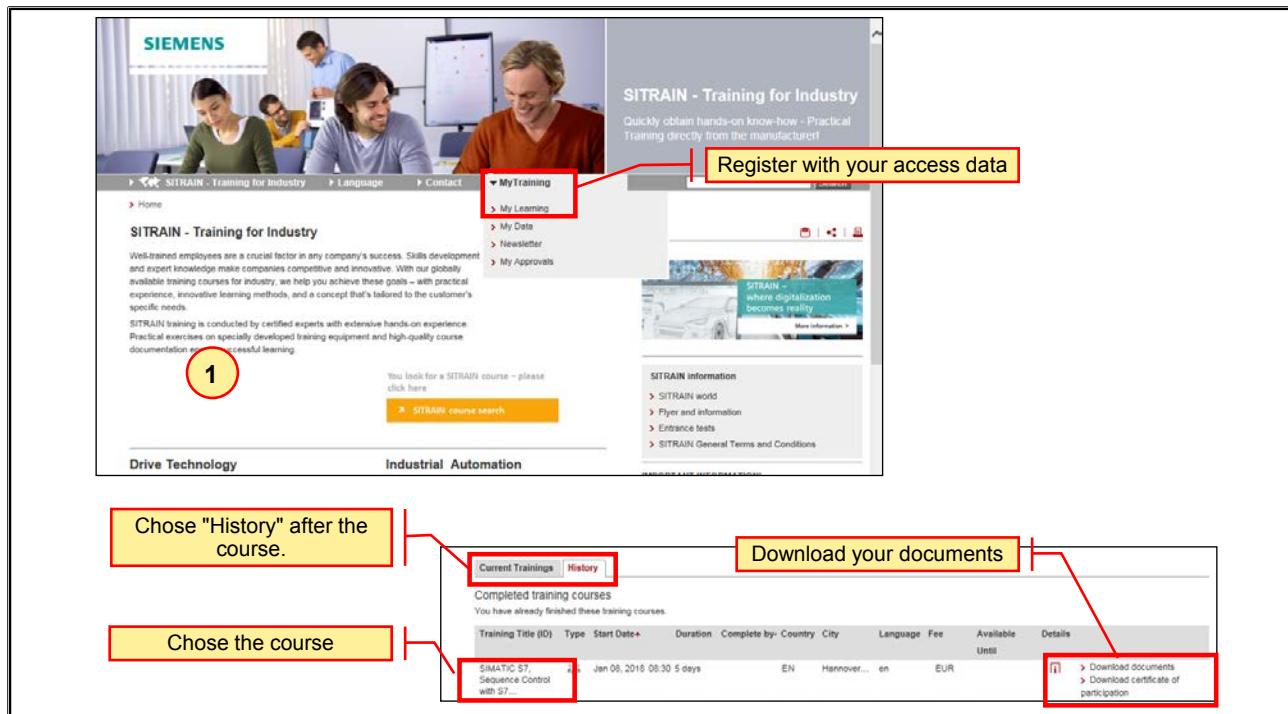
SIMATIC S7-300/-400 with STEP 7 V5.x

- › SIMATIC S7 Trainings based on SIMATIC S7-300/-400 with STEP 7 V5.x
- › SIMATIC S7 Programming based on STEP 7 V5.x
- › SIMATIC S7 Service Training based on STEP 7 V5.x
- › SIMATIC Safety Integrated based on STEP 7 V5.x
- › SIMATIC S7 Engineering Tools based on STEP 7 V5.x
- › SIMATIC Technology based on STEP 7 V5.x

17.3. Learning path: SIMATIC S7 Programming in the TIA Portal



17.4. Download the training documents



If you want to download the training documents, proceed as follows:

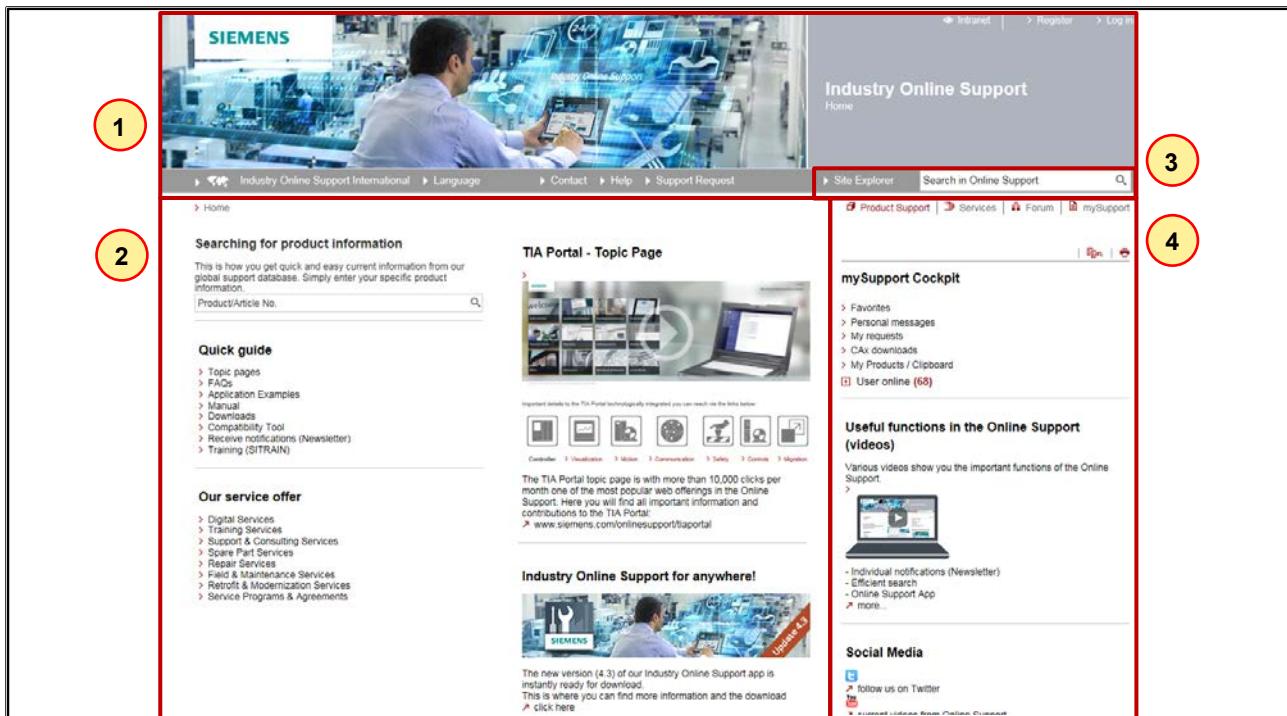
- Visit our new SITRAIN homepage at <http://www.siemens.de/sitrain>
- Register with your access data under the menu option **MyTraining**.
- Select **MyLearning** on the right-hand side of the submenu.
- Select your course and download your documents with a click on "Download documents".

Documents		X
Name	Size	
> SIMATIC S7 Sequence Control with ...	18,47 MB	

Hint:

Please note that the training documents may be used for personal purposes exclusively. You agree that you will not copy the training documents or make them accessible to third parties and that you will be liable for any damage resulting thereof.

17.5. The Industry Online Support – the most important innovations



The most important functions are always in the same place on all the pages:

- 1** The menu bar links to the main areas of the site. You can subscribe and register at any time to benefit from the features the personalized mySupport option offers.
- 2** Links to our service offerings are in the center. On the start page, you will find up-to-date information and links, which quickly brings you to your destination in other areas of Online Support.
- 3** Links from the menu bar are repeated at the top of the page: Product Support, Services, Forum and mySupport.
- 4** On every page, you will find your personal mySupport cockpit. There, for example, you can see when the status of your support inquiry changes.

17.6. The Principle of Navigation

The screenshot shows the SIMATIC Industry Online Support Product Support interface. At the top, there's a banner with a man working on a control panel. The main navigation bar includes links for Home, Product Support, Contact, Help, Support Request, Site Explorer, and a search bar. A filter bar on the left allows filtering by product tree or products, with 'Standard-CPUs (493)' selected. The search bar contains 'Standard-CPUs'. Below the search bar are filters for Product (All), Entry type (All), Date (From, To), and sorting options (Date descending). The results section shows 493 entries for 'Standard-CPUs' with various application examples and certificates listed. On the right, there's a sidebar for 'mySupport Cockpit' showing user information and links for Favorites, Personal messages, My requests, CX downloads, My Products / Clipboard, Entries last viewed, My tags, and User online. Another sidebar on the right lists 'All information on Standard-CPUs' with links for Presses info, Catalog and ordering system online, Technical info, Support, and Service offer.

Here, you will find information about all the current and discontinued products, such as:

- Frequently Asked Questions (FAQ)
- Manuals and Operating Instructions
- Downloads
- Product Notes (product announcements, discontinuation, etc.)
- Certificates
- Characteristics
- Application Examples

You will not only be able to access these articles through the product tree, but also through a central filter bar. The integration of various search filters will give you access to relevant information after only a few clicks. The product tree has been moved to an equivalent filter. This has the effect that several filter steps can be combined clearly and comprehensibly.

Based on the preview numbers you can see the expected set of results before using a filter. This makes finding relevant information considerably easier and more efficient.

For example, you can customize your search by combining the product tree, a search keyword and a document type in your search.

There will be no hidden search parameters; all the settings and results will be clearly displayed.

17.7. Complete product information

The screenshot shows the Siemens Industry Online Support Product Support interface. At the top, there's a navigation bar with links for Intranet, user profile, and Log out. Below that is a search bar and a 'Topic Page' button. The main area displays a search result for the product 6ES7513-1AL01-0AB0. The result card includes a thumbnail image of the product, its name, a brief description, and a link to 'Product details > Technical data > CAx data > Add to mySupport products'. Below this, a list of 276 entries is shown, filtered by the same product code. The list includes various links such as 'Application example', 'Digitalization with TIA Portal: Virtual Commissioning with SIMATIC and Simulink', and 'Certificate Declaration of Conformity, RCM, ACMA'. Each entry has a small thumbnail, a title, a date, an ID, and a rating. On the right side, there's a 'mySupport Cockpit' sidebar with links to Favorites, Personal messages, My requests, CAx downloads, My Products / Clipboard, Entries last viewed, My tags, and User online.

A powerful function of the Industry Online Support is the direct access to complete product information. You can use it if you are looking for a quick and easy access to all the technical information about a Siemens Industry product. For example, for comparing products, if you are expanding your system or replacing individual components, this is how to do it:

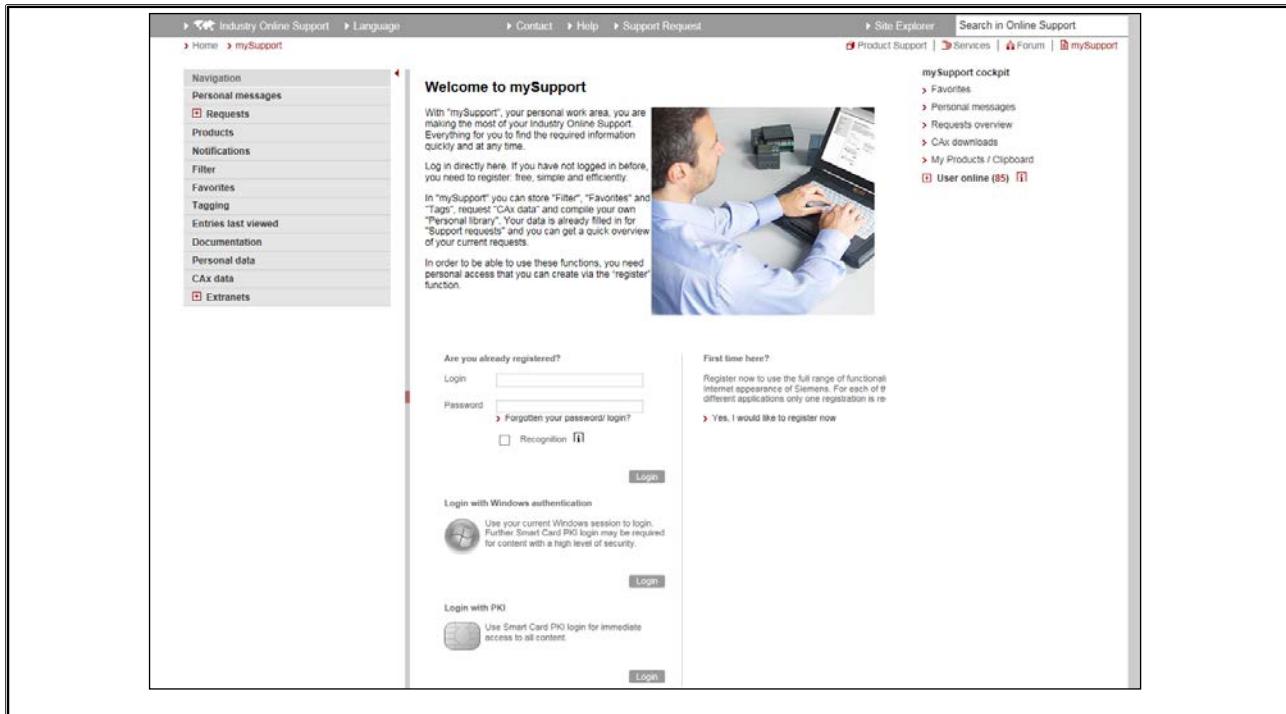
In the Product Support area, there is the central navigation bar.

To select a product, simply select the filter “Product.” Enter an order number or a product name here. You will be supported by a dynamic display of suitable products (list of suggestions).

One more click and the details of the selected product will be displayed – always up to date:

- Product life cycle, consisting of milestones with dates (e.g. delivery release, discontinuation of the product, ...). You will find out whether the selected product is a current product or whether the product is already in the discontinuation phase.
- Successor products for discontinued products and new developments will be suggested. If there is a successor product, you will get a direct link to the product information of this product.
- Technical data – clear, compact and complete. You get all the available technical data concerning the selected product here – dimensions, operating voltage or the number of inputs/outputs, etc.

17.8. mySupport – Overview



mySupport

The mySupport area will always remain your personal workplace; with this feature you can make the best of your Industry Online Support experience.

The most important thing, if you're already working with mySupport, you can take all your previous personal data and information you've filed away with you to the Industry Online Support.

In this area, you can compile the information that is important for your daily work – we provide you with the suitable tools. Create your own folder structures and file information such as bookmarks. There are numerous options, whether you want to file items by project or by products.

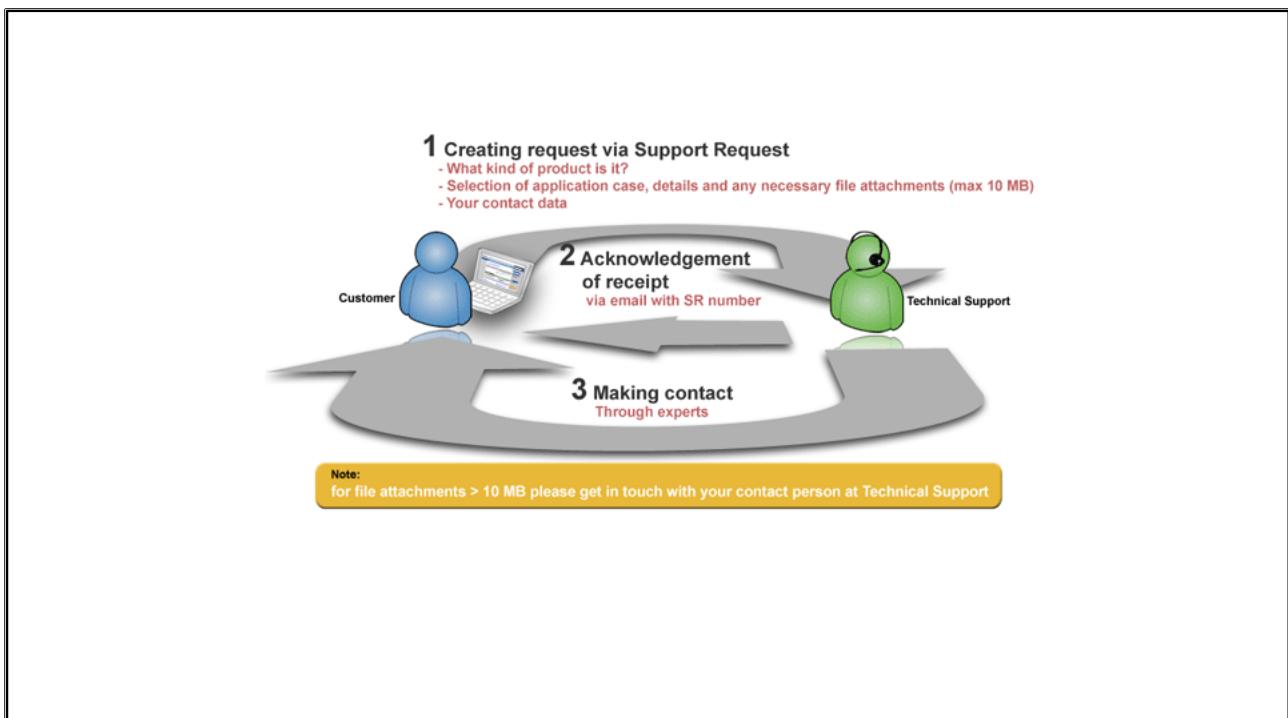
Moreover, you can now add notes, comments and tags (keywords). The system automatically creates a “Tag Cloud” based on your entries so you can access information quickly and easily by means of your own terms. The operation is consistent throughout mySupport so that you will easily find your way around. “Drag & drop” is also possible.

As soon as you are logged on, the mySupport cockpit is always at your side. It will immediately show you when the status of a support request changes, or when you receive new personal messages. You also have direct access to your personal keywords in the tag cloud, to the entries last visited, and you can see which user is online.

Here, just a few highlights:

- The previous MyDocumentationManager is now completely integrated into mySupport under the name of “mySupport-Documentation.” The function category “Documentation” contains all the functions of the MyDocumentationManager and provides a few innovations, too.
- The Service & Support Newsletter has been completely revamped. An individual messaging system will more than replace it.

17.9. Support Request



Support Request

To create a Support Request, different options are available to you in Online Support:

- You will find the "Support Request" option in the menu on all Online Support pages.
- Alternatively, you can create a new request in mySupport in the "Requests" category.
- Or directly click on the following link:

<http://www.siemens.com/automation/support-request>

Tips for creating a request:

- Select your product and use case as accurately as possible; try to avoid selecting "Other". By doing so, you ensure optimum support by our experts and appropriate suggested solutions.
- Did other users have a similar problem? This step already offers frequent problems and solutions. Take a look – it will be worth your while!
- Describe your problem with as much detail as possible. Pictures or explanatory attachments allow our experts to consider your problem from all sides and develop solutions. You can upload multiple attachments up to 10 MB per file.
- Before each sending, verify your personal contact information and the data you have entered. The final step additionally offers the option to print the summary.

As a logged in user, you can track the status of your requests online. To do so, navigate to "My requests" in the "Requests" category in mySupport.

17.10. Support Request

The screenshot shows the 'My requests' section of the SIMATIC TIA Portal Online Support interface. On the left, a navigation sidebar lists various categories like Personal messages, Requests, Products, Notifications, etc. The 'Requests' category is expanded, showing 'Overview' and 'Create new request'. The main area is titled 'My requests' and contains a search bar and a status dropdown set to 'Everything'. Below this, there's a list of requests with columns for SR number, Product and subject, Status, and Created on. One request is listed: '1-3871916175' for 'STEP7 Professional V13', which is 'Closed' and was created on '2/12/2015 8:49 AM'. There are also links to 'Show details...' and 'Add note'.

SR number	Product and subject	Status	Created on
1-3871916175	STEP7 Professional V13	Closed	2/12/2015 8:49 AM

17.11. Industry Online Support – wherever you go



- Mobile access to more than 300,000 entries on all Siemens Industry products
- Reduced to the essential functions
- Application case: initial diagnosis of problem or in case of failures directly at the system or machine



*Quick and easy access to technical information, anytime.
Scanning function, search and Support Request – everything at your fingertips at any time.*

The app supports you, for example, in the following fields:

- Problem solving during the implementation of a project
- Troubleshooting of failures
- Expanding or restructuring your system

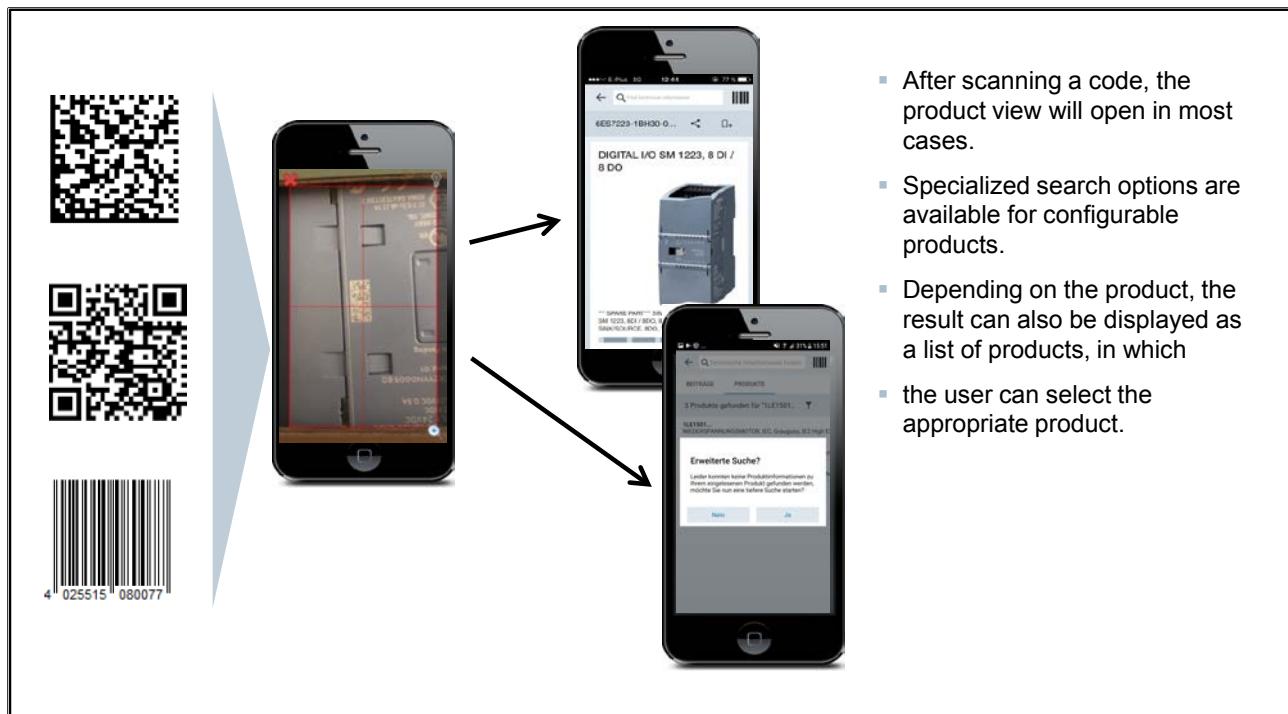
It also provides you with access to the Technical Forum and to further entries created for you by our experts:

- FAQs
- Application examples
- Manuals
- Certificates
- Product notes and many others

The main functions at a glance:

- Scan your product codes / EAN codes for a direct display of all technical and graphic data (e.g. CAx data) about your Siemens Industry product.
- Send your product information or entries per e-mail in order to process the information directly at the workstation.
- Send your requests to Technical Support at your convenience. Detail information can easily be added using the scan or photo function.
- Use the offline cache function to save your favorites to your device. In this way you can call these entries, products and conferences even without network coverage.
- Transfer PDF documents to an external library.
- The contents and surfaces are available in six languages (German, English, French, Italian, Spanish and Chinese) - including a temporary switching to English.

17.11.1. Scanning product/EAN code



17.11.2. Scan functionality

Data matrix codes  on Siemens products as per standard SN60450	QR code  e.g.: in advertisements relating to Siemens content	The scan functionality in the Online Support app supports the following types of code: Data matrix code QR code EAN13 bar code Code39 bar code When one of these codes is recognized, the respective product view is called up in the app. Exception: The QR codes contain URLs – these are directly called up and displayed in the app by the integrated browser (but only, if "siemens" is contained in the URL).
EAN13 bar code  on Siemens products	Code39 bar code  (very hard to recognize / scan) on Siemens products as per standard SN60450	

17.12. Forum - the communication platform for Siemens Industry products

17.12.1. Conferences and Forum management

The screenshot shows the SIMATIC TIA Portal Forum interface. At the top, there's a navigation bar with links like 'Industry Online Support', 'Deutsch', 'Contact', 'Help', 'Support Request', 'Site Explorer', 'Product Support', 'Services', 'Forum' (which is highlighted with a red box), and 'mySupport'. Below the navigation is a 'mySupport Cockpit' sidebar with various links. The main content area is divided into two main sections: 'Conference overview' and 'Forum management'.

Conference overview: This section displays a list of conferences. A search bar at the top says 'Search in "Forum"' with the input 'Drive ES'. Below it, there are filters for 'Actions', 'Title', 'Last post', and 'Topics'. The list includes topics like 'LOGO!', 'SIMATIC TDC, FM458, T400', and 'Can't import Profinet DP data from CP50MO'. Each topic has a brief description, the number of experts, and the last post date.

Forum management: This section is located on the left side of the main content area. It contains a 'Navigation' sidebar with categories like 'Product Conferences', 'Solutions, Applications and Initiatives', 'General Conferences', 'Forum management' (which is circled with a yellow circle labeled '1'), 'Quicklinks', and 'Forum Terms'. Under 'Forum management', there are links for 'Manage profile', 'User filter', 'Conference filter', 'My topics', 'My suggestions', 'Received Feedback', 'Quicklinks', and 'Forum Terms'.

- 1** On the left side, you will find the so-called conference tree. It allows you to navigate through the individual discussion areas.
- 2** The conference overview is the central discussion area of the Technical Forum. This is where the community meets to discuss technical questions about Siemens Industry products.
- 3** In forum management, you will find your personal control center for the Technical Forum. It allows you to manage your specific profile data and filters.

The screenshot shows the 'Manage conference filter' page. On the left is a navigation sidebar with links like 'Product Conferences', 'Solutions, Applications and Initiatives', 'General Conferences', 'Forum management' (selected), 'Quicklinks', and 'Forum Terms'. The main area has a title 'Manage conference filter' and a sub-instruction 'Add or remove conferences to the filter'. It includes a link 'Show topics in "My conferences"' and two lists: 'All conferences' (a long list of various SIMATIC-related topics) and 'My conferences' (a shorter list showing 'Industrie-PC SIMATIC PC (de)' and 'Prozesseleitsystem SIMATIC PCS 7 (de)'). Below these lists are '">>>' and '<<' buttons. At the bottom, there's a section for notifications with the text 'Enable notifications if there are new topics.' and a checkbox 'Set notifications'.

Conference filter

Add conferences to your personal filter of preferred conferences.

This allows you to enable a notification that informs you when new topics are started in these conferences.

In Quicklinks, the Technical Forum additionally offers an overview page that contains all topics of your preferred conferences.

Managing profile

Profile management provides interesting information and functions:

- You get an overview of your activities in the Technical Forum.
- You can view your rank, any special permissions and your ranking progress.
- You can store a signature and a personal description for your profile in the forum.
- You have direct access to the quick links to get an overview of all topics you have contributed to.

User filter

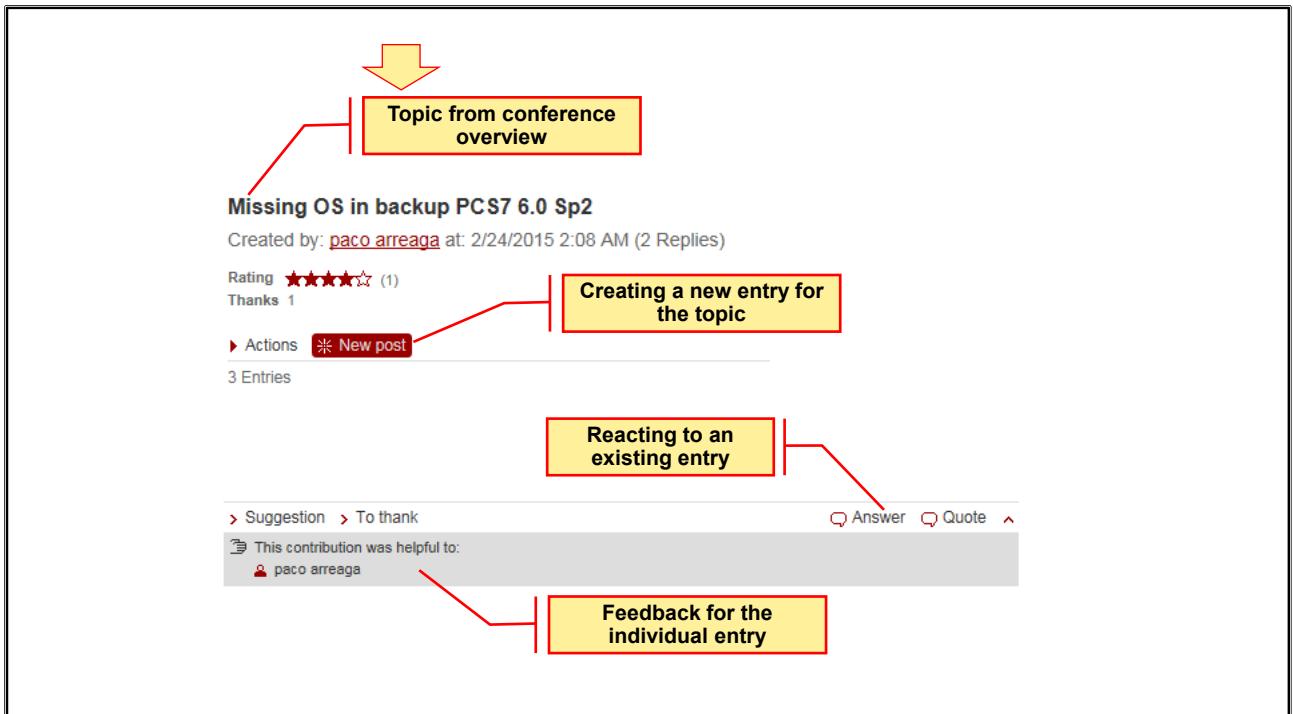
Have you found a user in the Technical Forum who posts entries that are particularly interesting? Then add this user to your list of "preferred users".

This allows you to enable a notification that informs you when the user has posted a new entry.

In Quicklinks, the Technical Forum additionally offers an overview page that contains all topics of your preferred users.

17.12.2. Interactions in the Forum

The screenshot shows the SIMATIC TIA Portal forum interface for "Process Control Systems SIMATIC PCS 7". The top navigation bar includes "Actions", "Legend", "Experts", "New topic" (highlighted with a red box), "Edit filter", and "Load filter". Below the navigation is a search bar with "Search in 'Process Control Systems SIMATIC PCS 7'" and a "for" dropdown. The main area displays 4748 entries per page, with 10 selected. A yellow box highlights the "Creating a new topic in a conference" link. Below this, a topic from the conference overview is shown with a status of "solved" (highlighted with a red box). The topic title is "Missing OS in backup PCS7 6.0 Sp2" and it was posted by "paco arreaga" on 2/26/2015 at 3:41 AM. It has 2 replies and 81 views. A yellow arrow points down to the topic title. To the right, a yellow box highlights the "Rating of the topic" section, which shows a 5-star rating with "(1)" reviews.



Creating a new entry

Do you want to create or format a new entry? The entry editor provides all the necessary functions.

- You can upload and publish in the forum a file with "Add attachment".
- You would like to check before the publication how your entry will actually look? A preview is available for this purpose.
- You would like to look at the topic again to which you create an entry? Please, you used the link over the input area (right mouse button > open in a new tab or window)

Posting / replying to an entry

Do you want to participate in an existing discussion with your own entry? Click on "Reply" and post your personal entry to support other users in answering the question.

- Use the "Reply" link to go to the entry editor and create a reply without quoting the entry.
- If you want to quote the entry, possibly only excerpts of it, use the "Quote" link. The content of the quoted entry is then displayed accordingly in the entry editor.

Rating an entry / saying thank you

Do you find an entry particularly interesting? Use the available functions and rate the entry or say thank you to provide personal feedback. Ratings and thank yours are the rewards our community members get for the support they provide. When you rate an author or entry, this will be added to the already existing ratings. The average value of all ratings is displayed.

Aside from feedback to the author of the entry, you also draw other readers' attention to particularly valuable entries and helpful authors.

17.13. Task and Checkpoint

Task: Software compatibility

Goal

Find out which current version of virus scanners is compatible with your engineering software.

Use all information sources available:

- Readme files in the installation folder
- The compatibility tool of the Industry Online Support
- Entries in the Product support
- Entries in the Forum
- Create a Support Request.

Checkpoint



Let's think about this:

- Name some reasons for registration in MySupport.
- What do you think is the best way to have always the latest version of the required manuals for your job with you?