SIEMENS
Ingenuity for life

TIA-MICRO1

SITRAIN
Digital Industry Academy

SIMATIC SIMATIC S7- S7-1200 basic course

siemens.com/sitrain

SIEMENS

SITRAIN
Training for Industry

SIMATIC S7-
S7-1200 basic course

Course TIA-MICRO1

Name: _____

Course from: _____ to: _____

Trainer: _____

Training site: _____

SITRAIN courses on the internet: www.siemens.com/sitrain

Course folder Version: V16.00.00 (for STEP 7 V16)

Contents

1

1. System Overview

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

... be familiar with the concept of “Totally Integrated Automation” (T.I.A.)

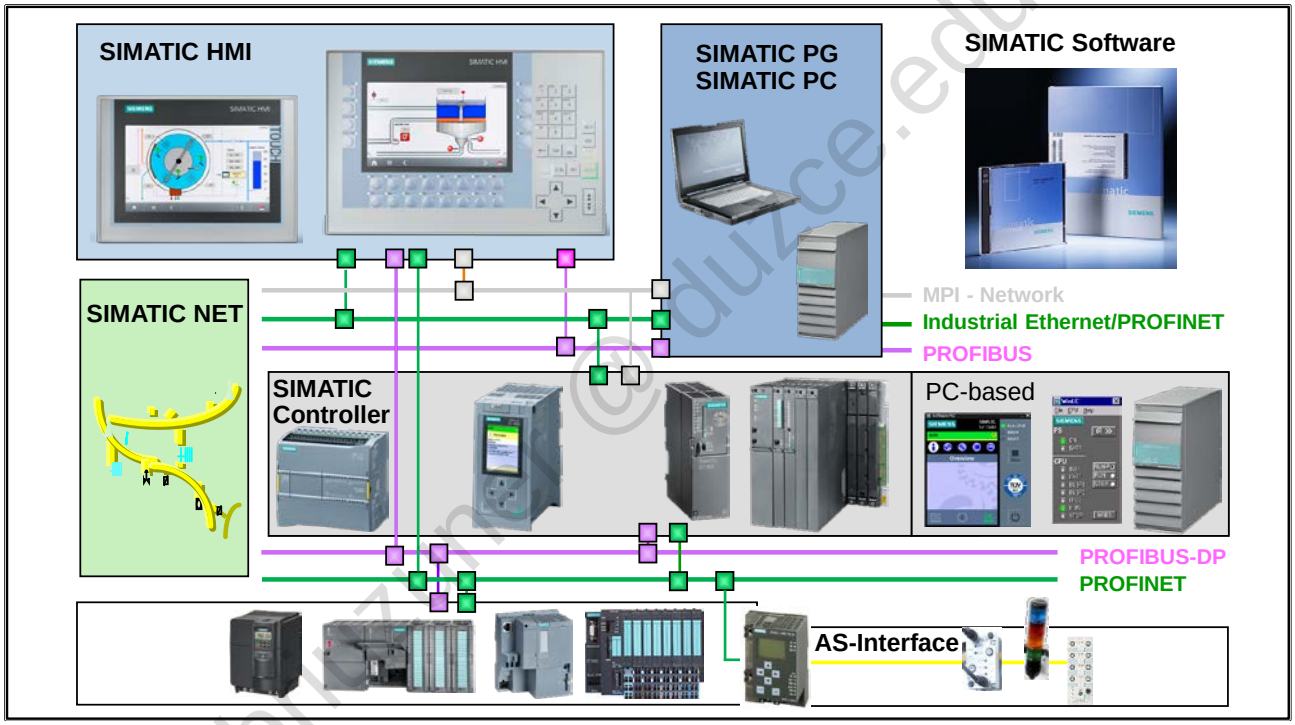
... be familiar with the TIA Portal Information Center

... have an overview of the available modules

... have an overview of the new SIMATIC S7-1200/1500 system family

... know the S7-300 and S7-400 automation systems

1.1. 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

1.2. TIA Portal Information Center

https://support.industry.siemens.com/cs/ww/en/view/65601780

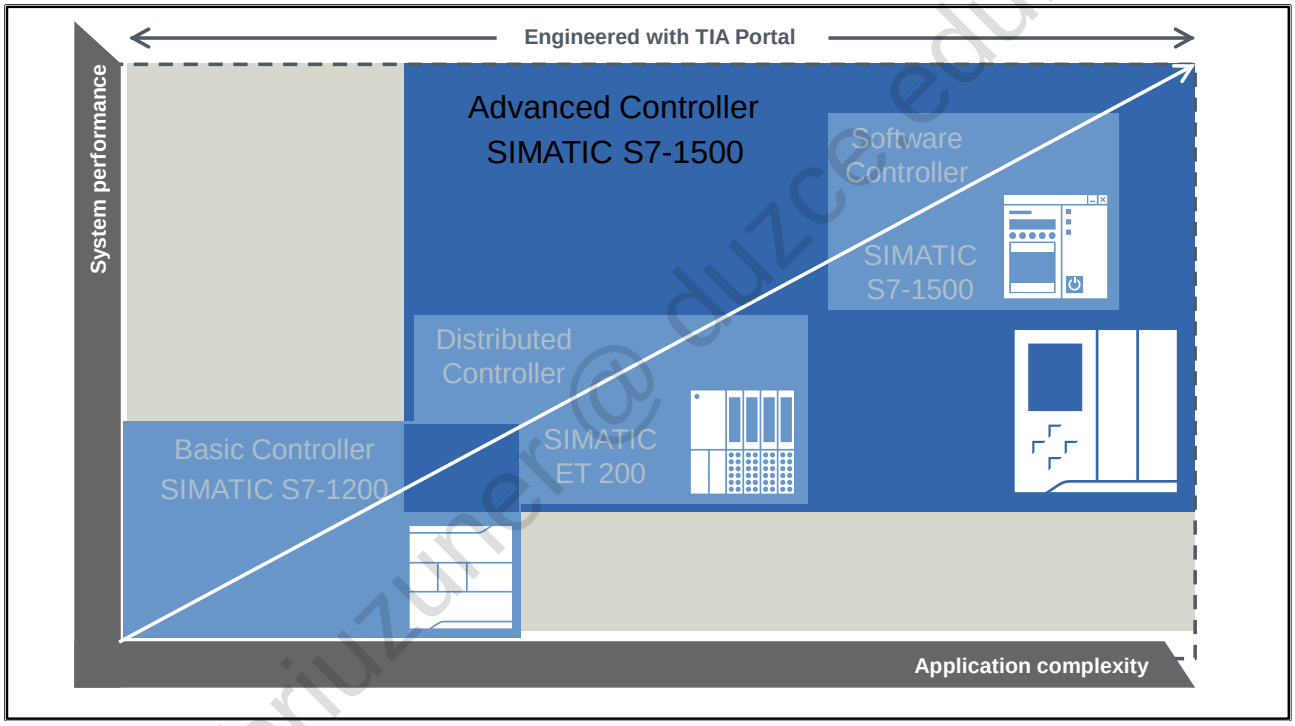
Entry ID 65601780

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.

1.3. Overview Controller



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

1.3.1. 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.

1.4. Overview: Available Modules

1.4.1. 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

1.4.2. 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:

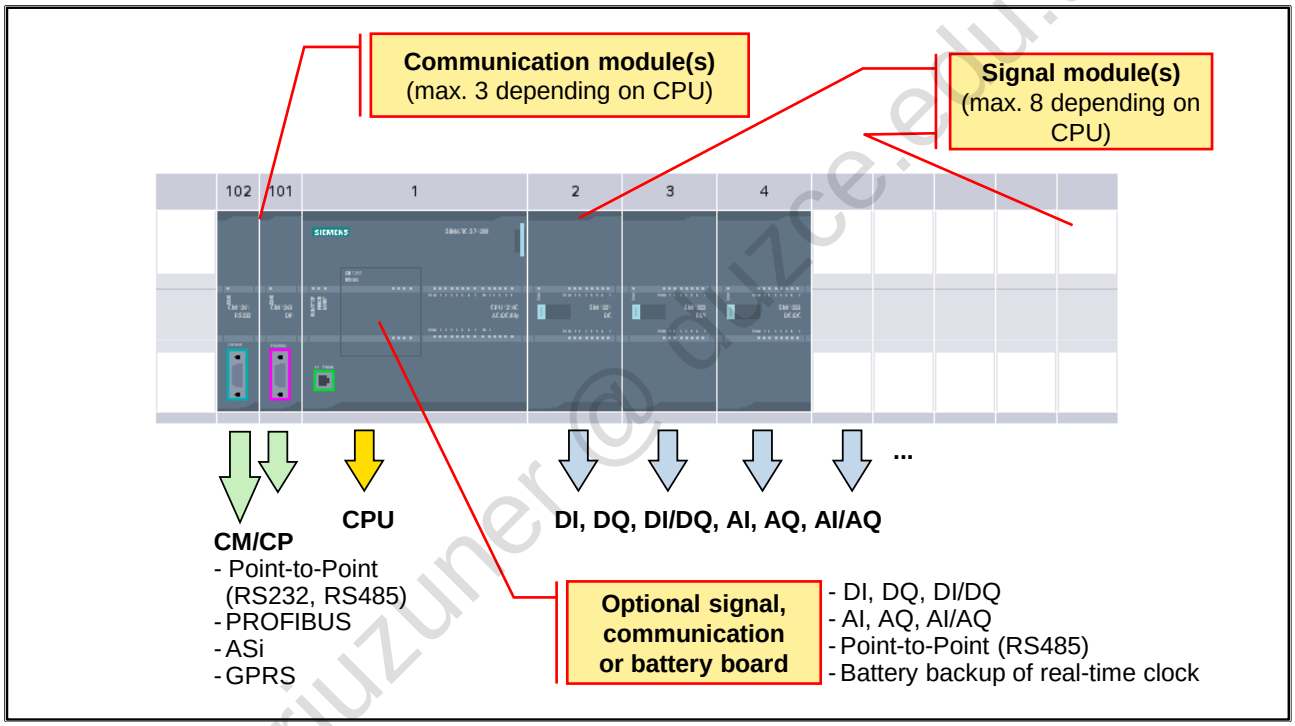
TIA Portal Information Center **>** Product information **>** Controllers

1.5. 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 (depending 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

1.5.1. 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)

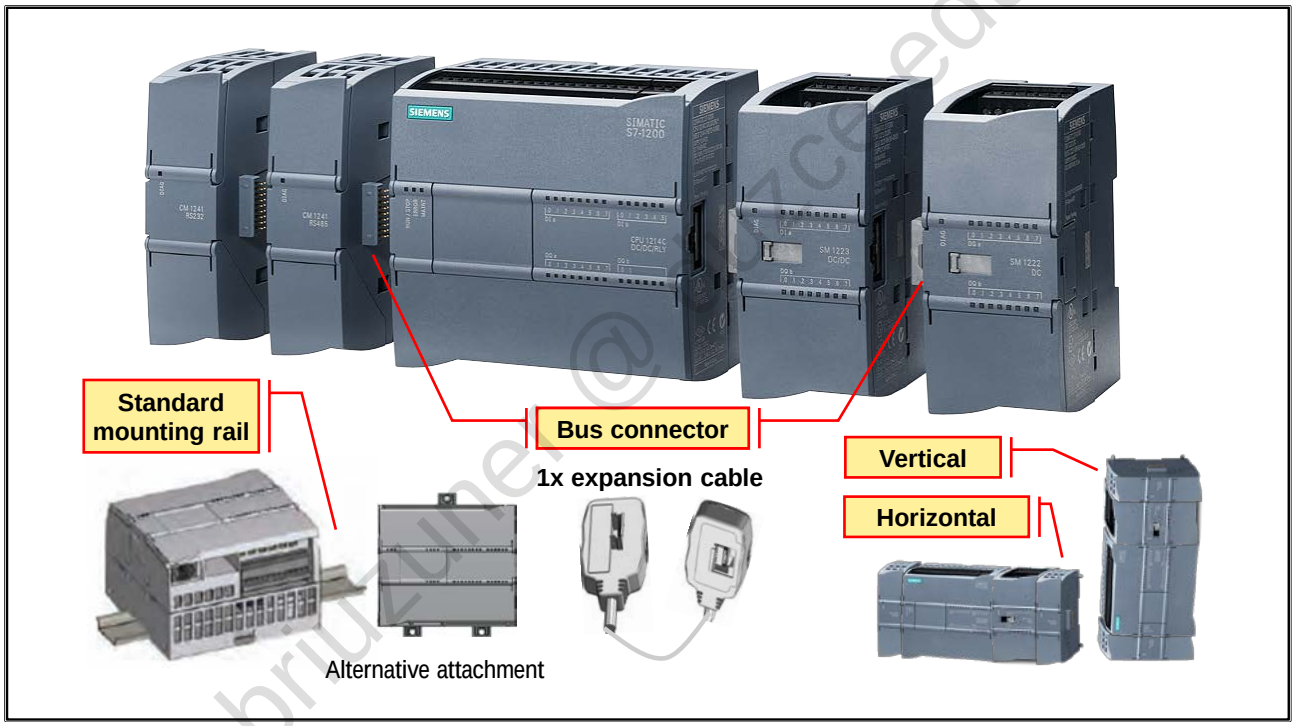
- Point-to-point connection (RS232, RS485)
- PROFIBUS
- ASi-Master
- Telecontrol (GPRS functionality)

Expansion Board

The CPU can be expanded with a signal board for additional onboard I/O or with a communication board.

Alternatively, a battery board can be added to provide a long-term battery backup of the CPU real-time clock.

1.5.2. 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 each SM module.

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

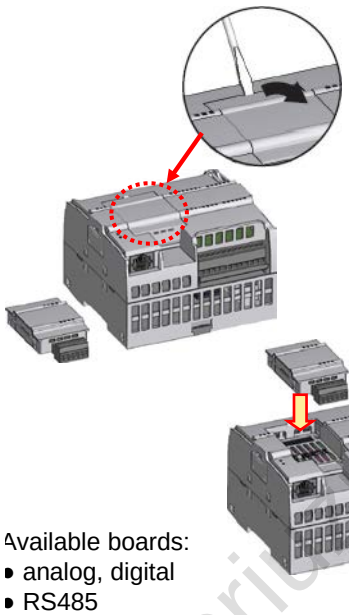
Mounting Positions

A horizontal or vertical mounting is possible.

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

Further information regarding technical data can be found in S7-1200 system manual (Online support entry ID: 109741593) or in the information center “Documentation > hardware manuals > SIMATIC controllers > SIMATIC S7-1200”.

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



Available boards:
- analog, digital
- RS485
- battery

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) provides long-term battery backup for the CPU's real-time clock.

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

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

Highlights of the SIMATIC S7-1500 System

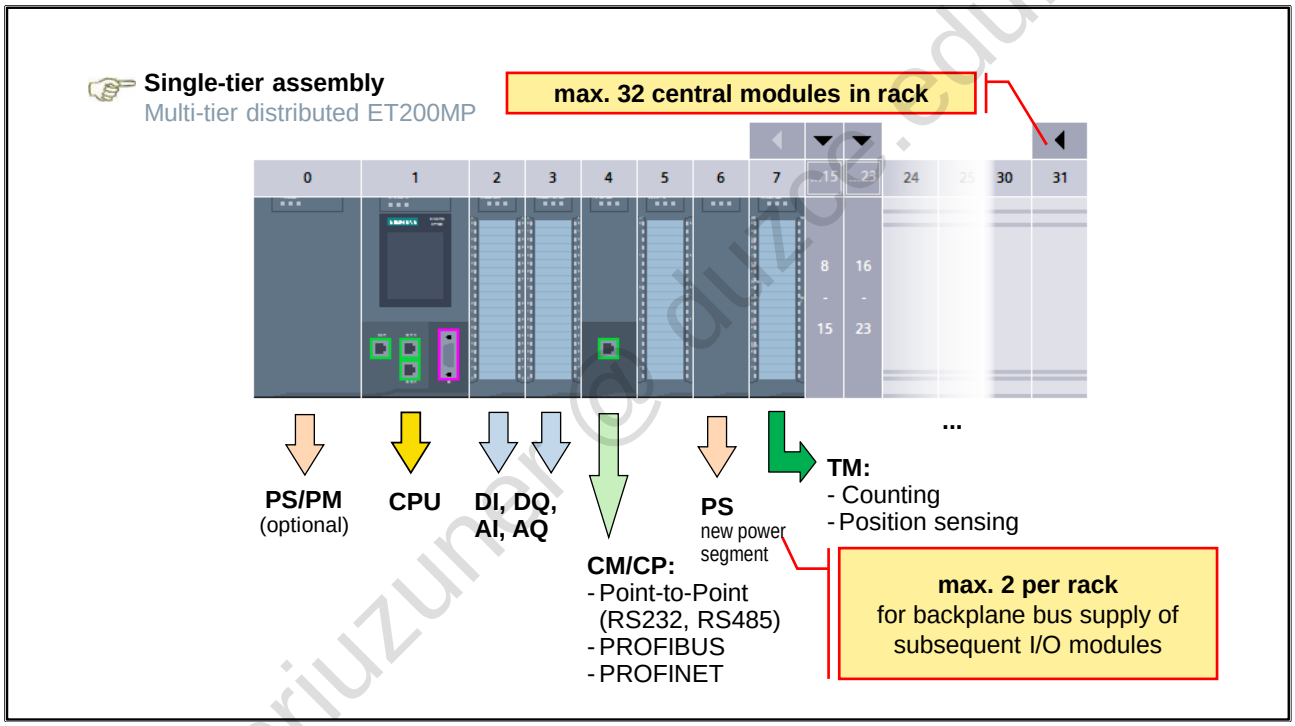
- Highest performance of the entire system (terminal-terminal)
  - High performance program execution in the CPU
  - High performance backplane bus
  - PROFINET interface with PROFINET IO IRT on every CPU
  - Automatically activated system diagnostics, right down to the IO channel

- 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

1.6.1. 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 must be set up, as required.

  A system power supply (PS) can also supply the load circuit for 24VDC modules in addition to the CPU.

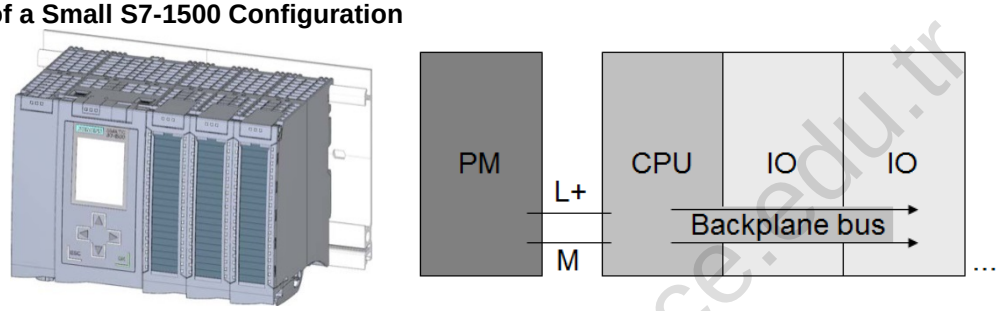
Power Supply and Power Segments of the I/O Modules

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).

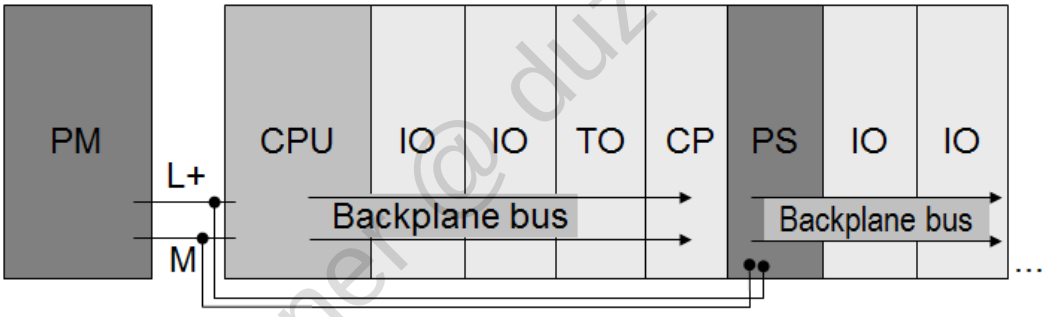
A maximum of 3 power segments can be set up per rack (1xCPU segment plus 2 more).

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.

1.7. 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
  has no integrated load memory and therefore it is imperative that a card is inserted.
- S7-1200
  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

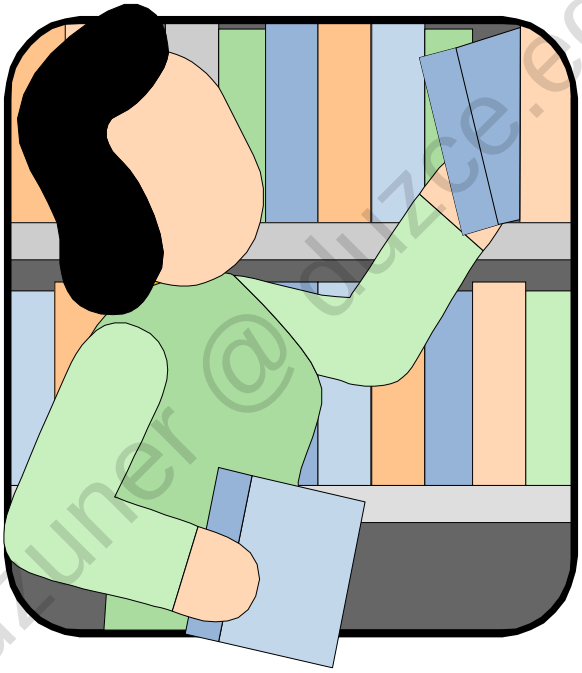
The use as Transfer card (card mode = "Transfer") is only supported by the S7-1200. Here, a program can be downloaded into the CPU without a PG if a card is inserted.

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

1.8. Additional Information



1.8.1. ET 200SP and ET 200pro Controller

- **SIMATIC S7-1500 with the design of a SIMATIC ET 200SP or ET 200pro**
- **for machines with distributed architecture and serial machines with limited space**
- **Transfer of the intelligence from the central control cabinet to Distributed Controller**
- **Available in standard and fail-safe version**

Further Information under the Link:

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

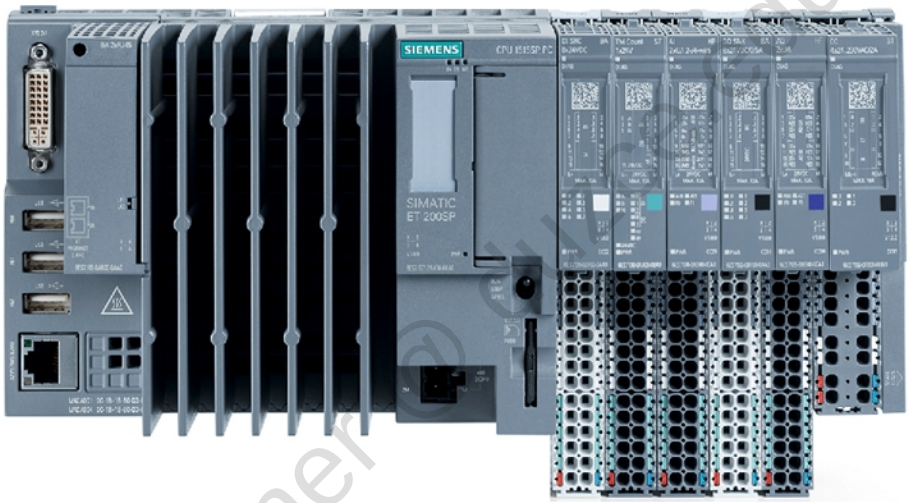
1.8.2. Software Controller

- **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

1.8.3. 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:

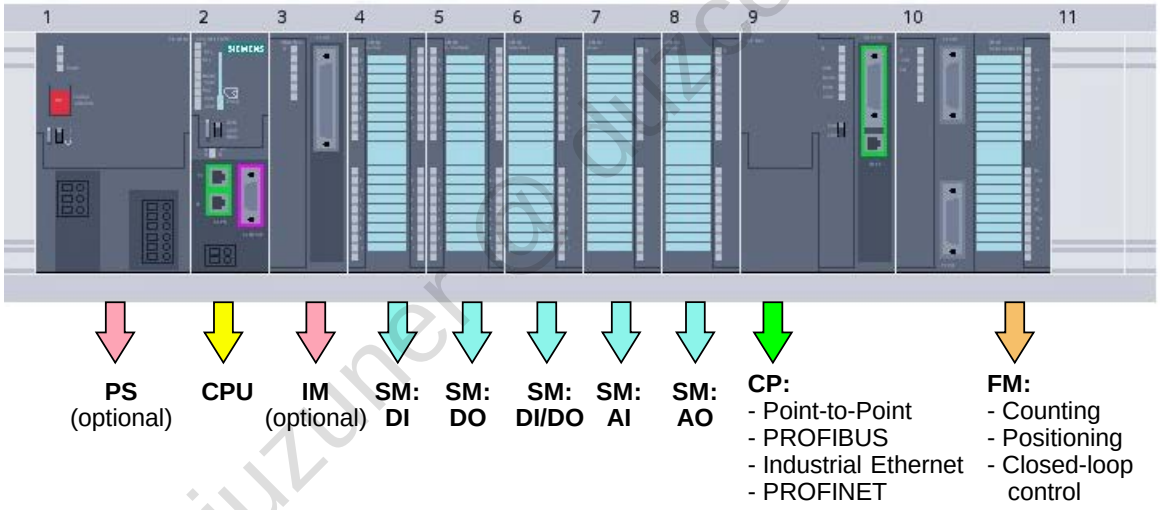
TIA Portal Information Center **>** First steps **>** Getting Started **>** SIMATIC Open Controller - Getting Started

1.8.4. 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 the 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

1.8.4.1. 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 later

Function Modules (FM)

- Counting
- Positioning
- Closed-loop control

Communication Processors (CP)

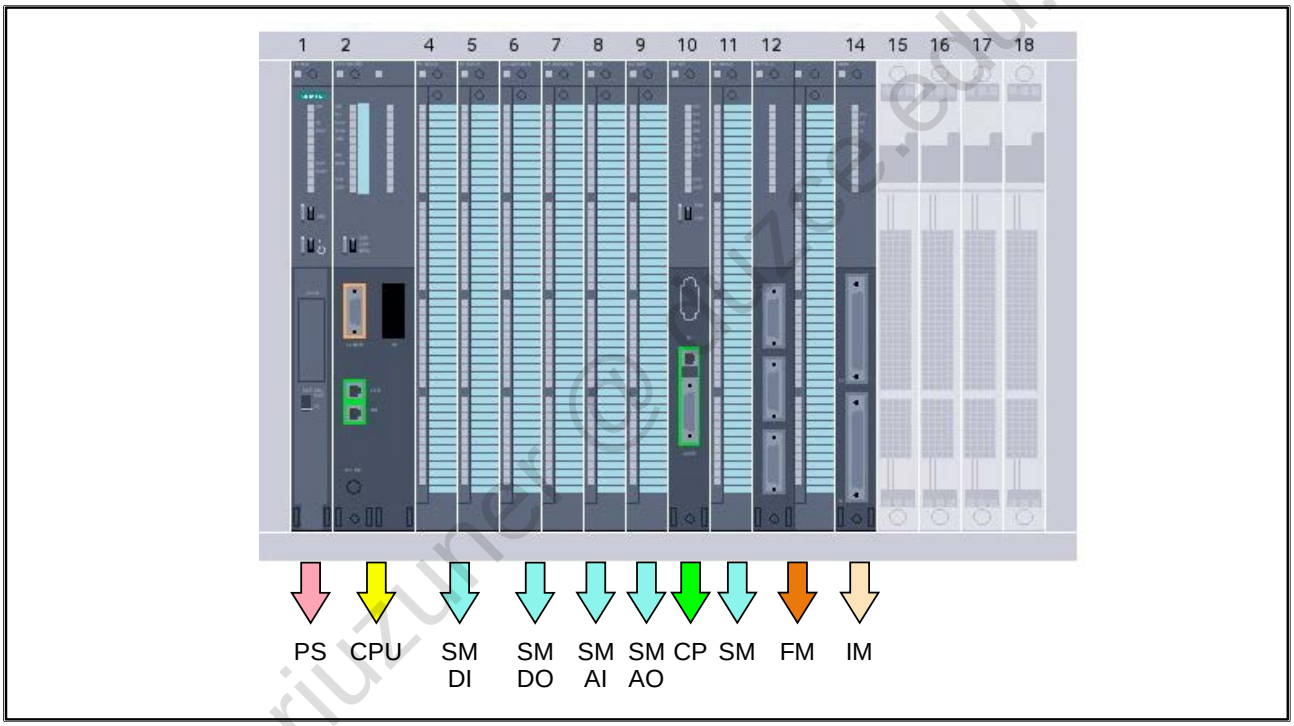
- Point-to-Point connections
- PROFIBUS
- Industrial Ethernet
- PROFINET

1.8.5. 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 the 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)

1.8.5.1. 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

2

2. Training Area Setup with S7-1200

Components of the Training Area with S7-1200

The training area for this course contains the following components:

- SIMATIC Field PG
- Training case with S7-1200, simulator and touchpanel
- Conveyor model

2.1. Configuration of the S7-1200 Training Device

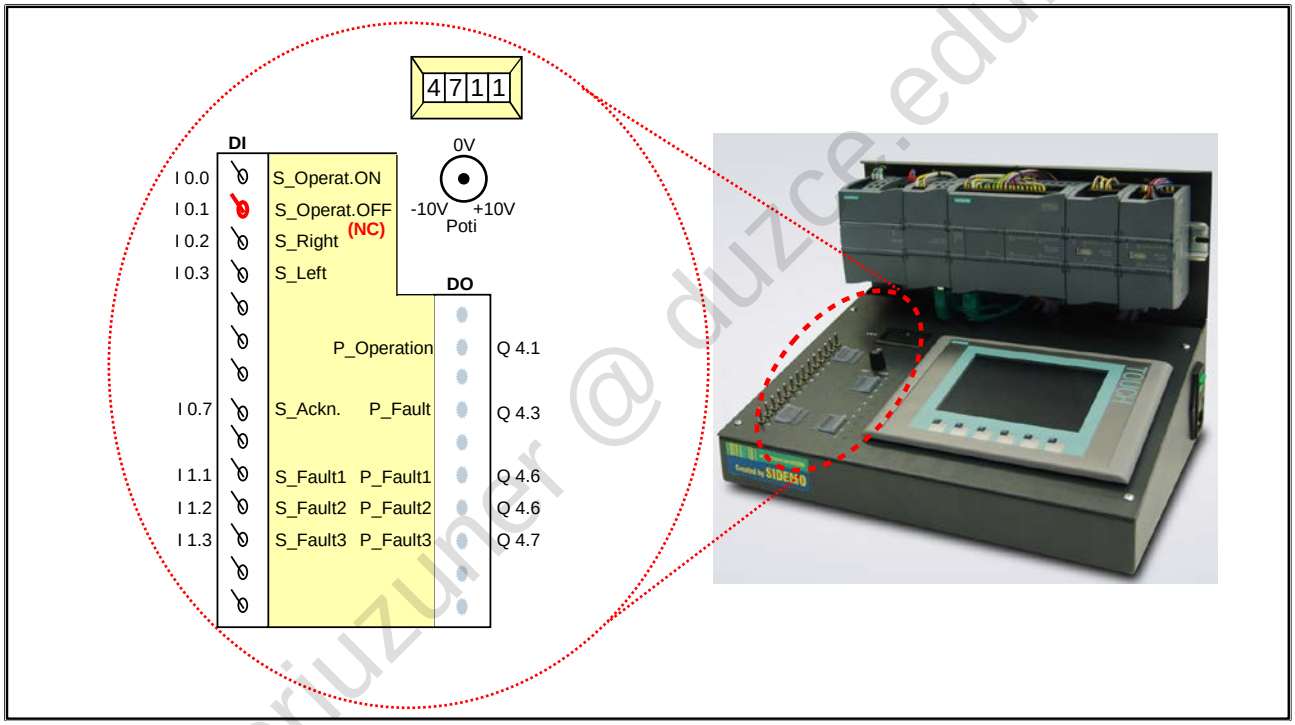
Device overview

	Module	Slot	I address	Q address	Type	Article no.	Firmware	Comment
		103						
		102						
		101						
	PLC_1	1			CPU 1214C DC/DC/DC	6ES7 214-1AG40-0XB0	V4.0	
	AI 4x13BIT/AQ 2x14BIT_1	2	96...103	96...99	SM 1234 AI4/AQ2	6ES7 234-4HE30-0XB0	V1.0	
	DI 8/DQ 8x24VDC_1	3	8	8	SM 1223 DI8/DQ8 x 24VDC	6ES7 223-1BH30-0XB0	V1.0	
		4						

Modules

The module addresses shown in the picture will be assigned in the chapter "Devices and networks".

2.2. Simulator Setup



Simulator Setup:

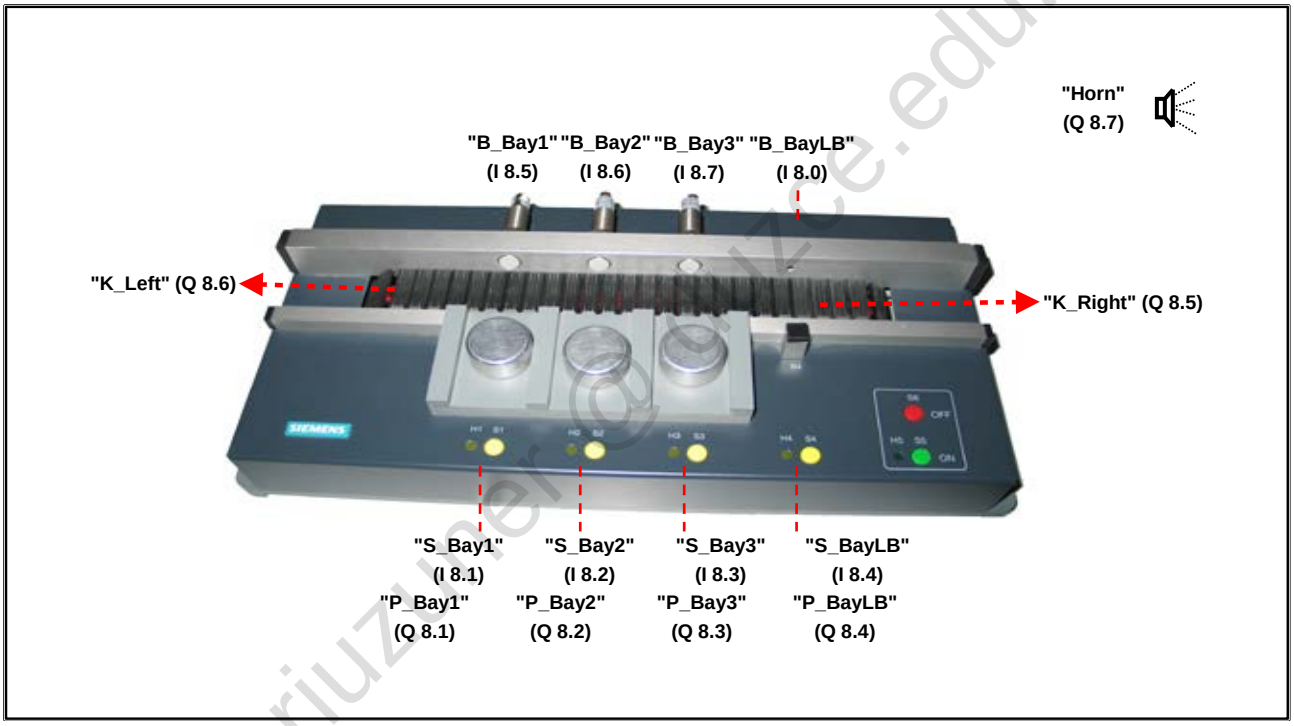
Together with the touchpanel, the simulator is used to operate the plant. It contains the following components:

- 14 switches, whereby the switch "S_OperationOFF" (I 0.1) is an NC contact
- 10 LEDs
- Potentiometers for specifying or simulating analog input signals
- Voltmeter for displaying the voltage set on the potentiometer

Addressing

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

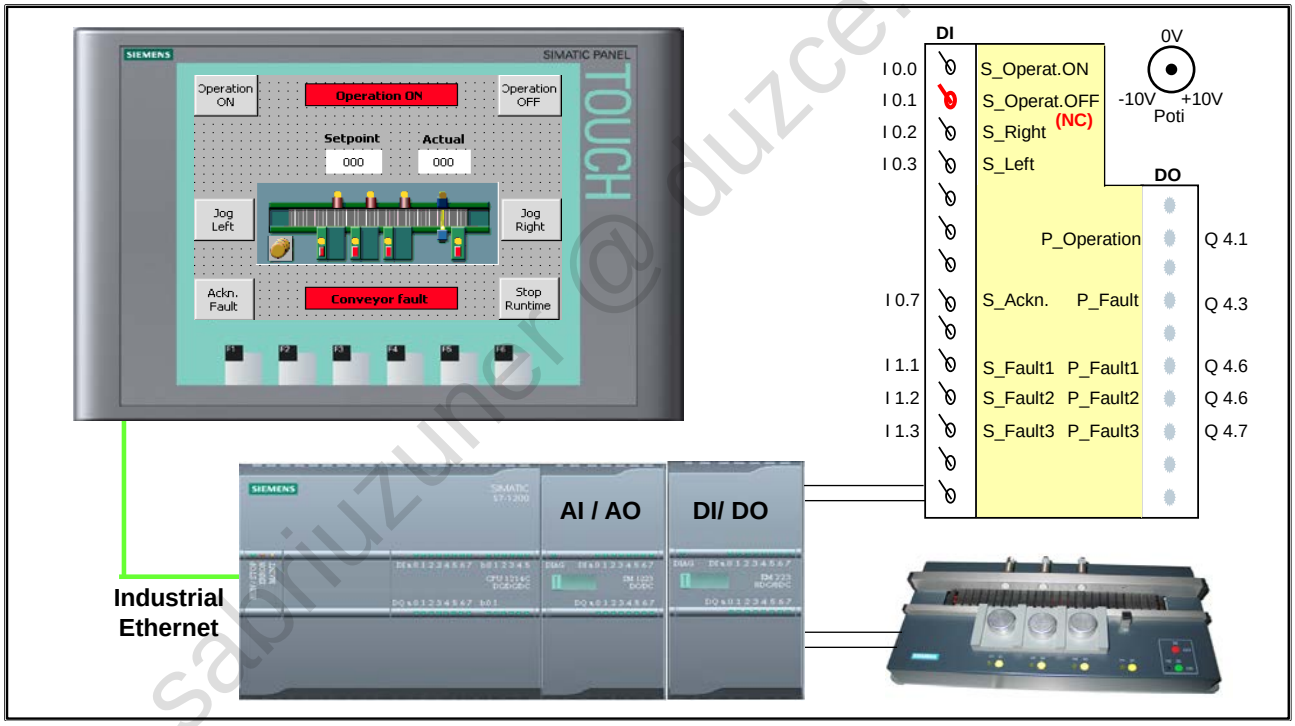
2.3. Conveyor Model Setup



Setup

The picture shows the sensors and actuators of the conveyor model as well as the I/O addresses to which they are connected.

2.4. Training Area as Plant with Distribution Conveyor and Touchpanel



The Training Area as Plant with Distribution Conveyor and Touchpanel

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

- When "P_Operation" (Q4.0) is switched off...

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.

- When "P_Operation" (Q4.0) is switched on...

parts are transported on the conveyor model from Bay 1 or 2 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 "P_Operation" is switched on are counted. When the ACTUAL number of transported parts reaches the SETPOINT quantity (which can be preset on the touchpanel) it is indicated on the conveyor model LED of the light barrier bay. Only after the message is acknowledged with the associated bay pushbutton, can a new transport sequence be started.

The indicator lights at Bays 1 and 2 show continuous light when a new part can be placed on the conveyor (conveyor motor is stopped and both proximity sensors are free); 1Hz flashing light at the Bay at which 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, no indicator lights are to light up); 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 and continuous light when the SETPOINT quantity has been reached.

Contents

3

3. Engineering Software TIA Portal

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

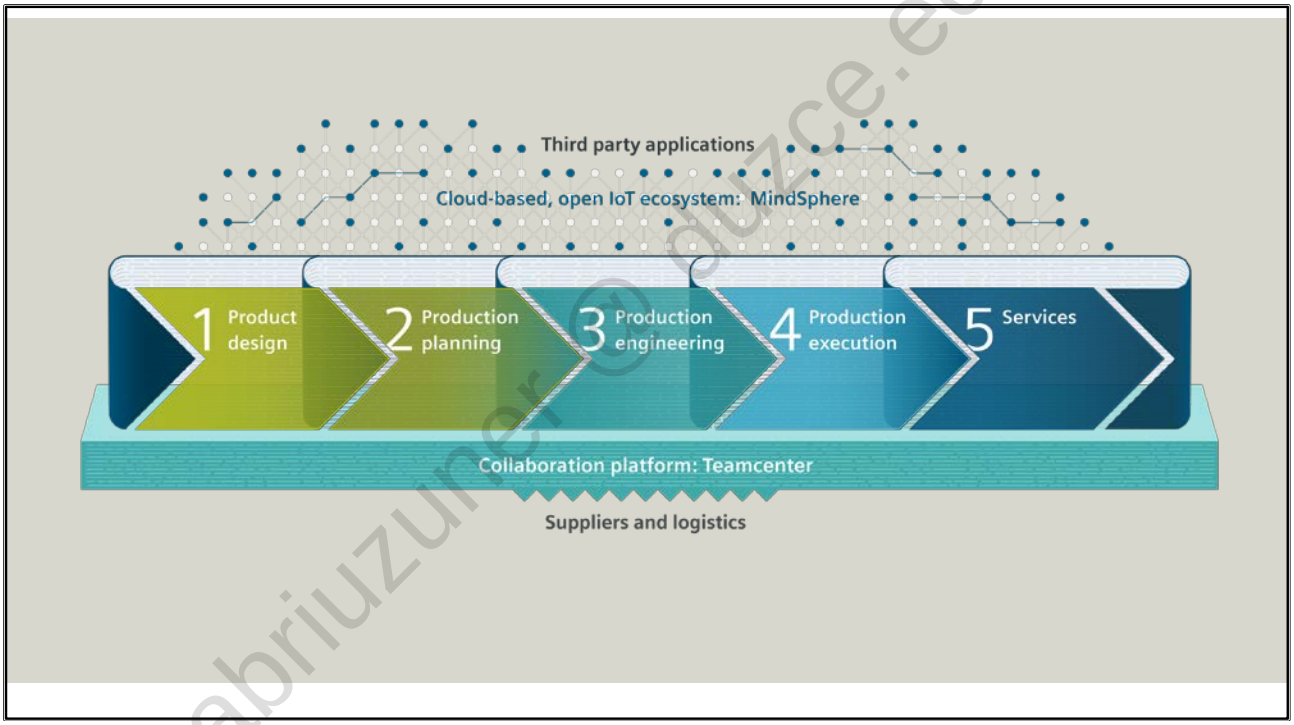
... have an overview of the scope of the engineering framework

... 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.

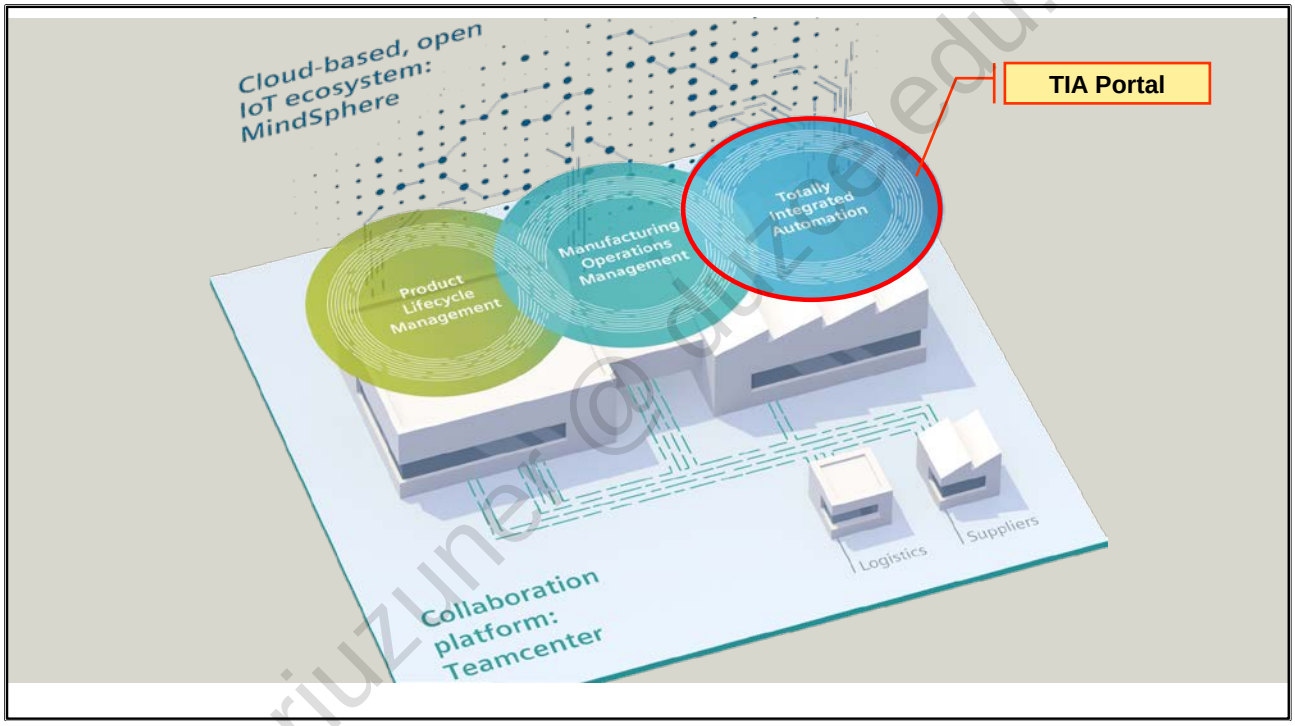
Siemens' holistic approach is to transform a traditional value-added chain into an integrated product and production lifecycle - starting with the product design, through the production planning, the manufacturing technology right up to production and services.

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.

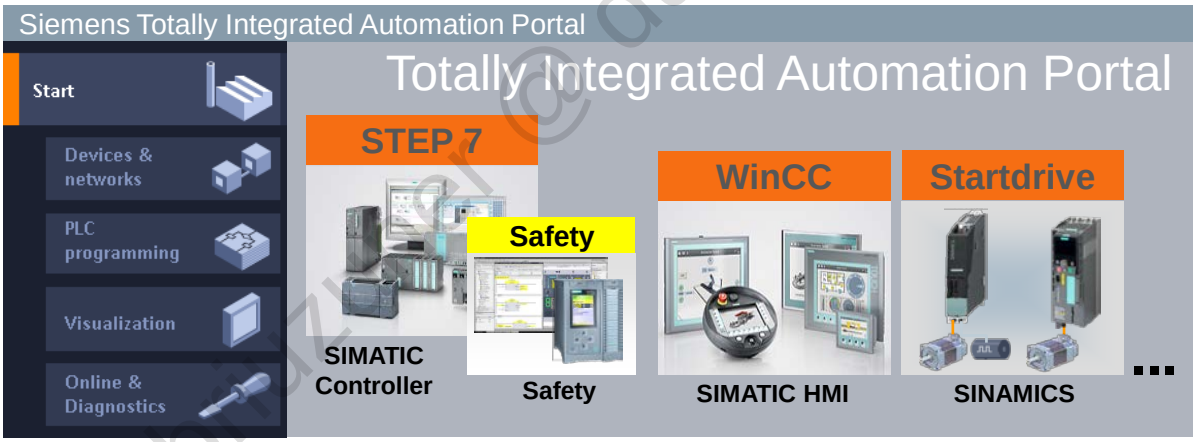
The new version shortens your time to market, for example by means of simulation tools, increases the productivity of your plant through additional diagnostics and energy management functions, and offers you broader flexibility by connecting to the management level. The new options benefit system integrators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

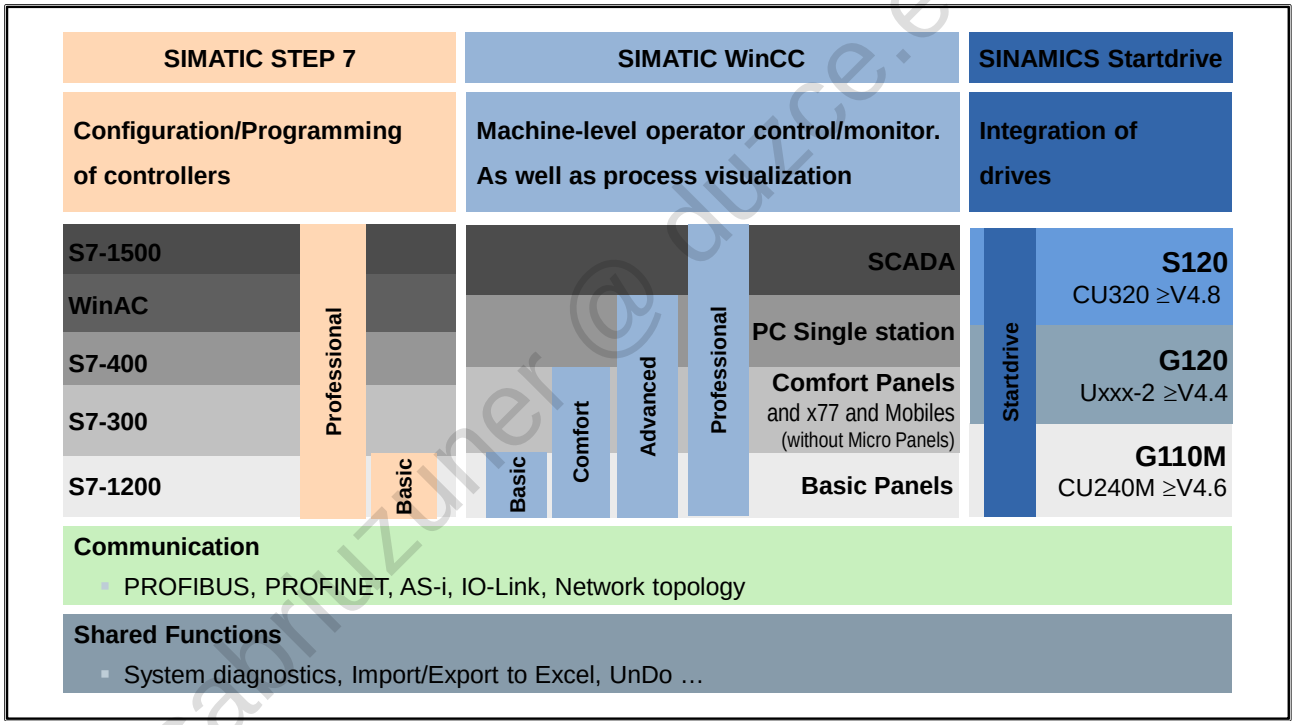
Advantages of a Central Engineering Framework

- 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
- Startdrive
  for parameterizing drives

3.4. Scope of the Products

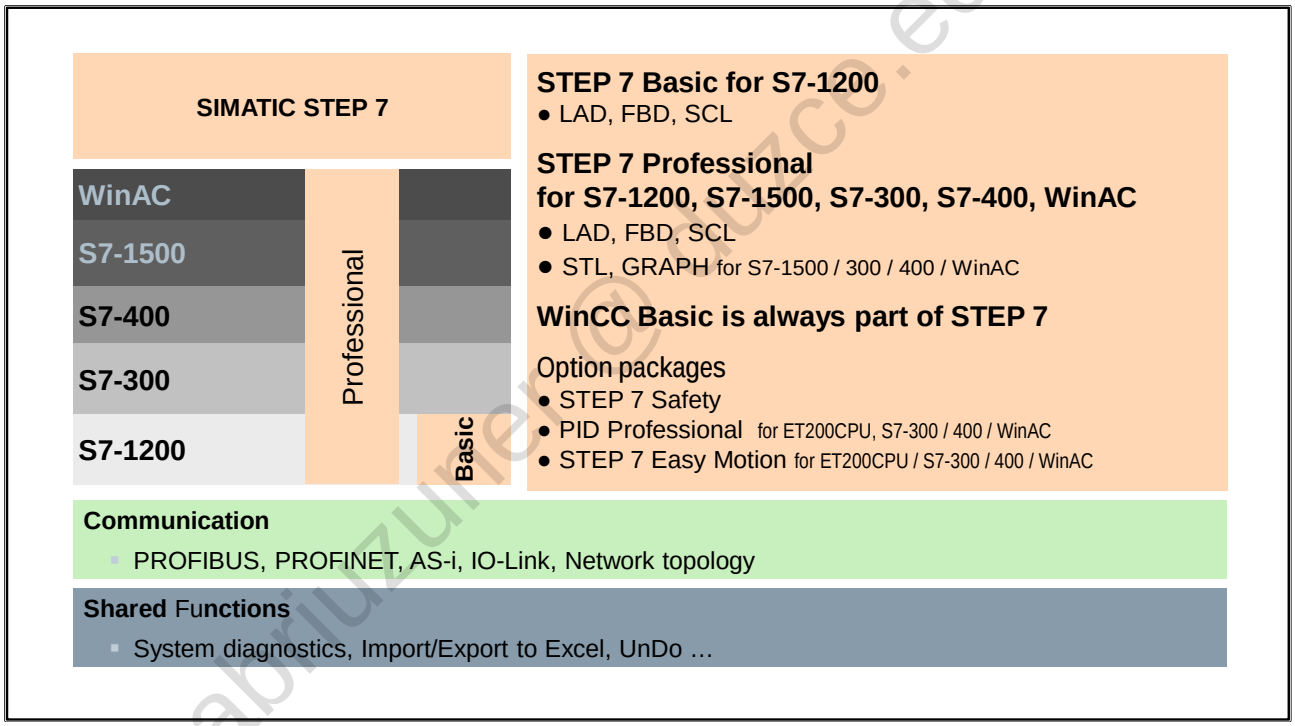


TIA Portal

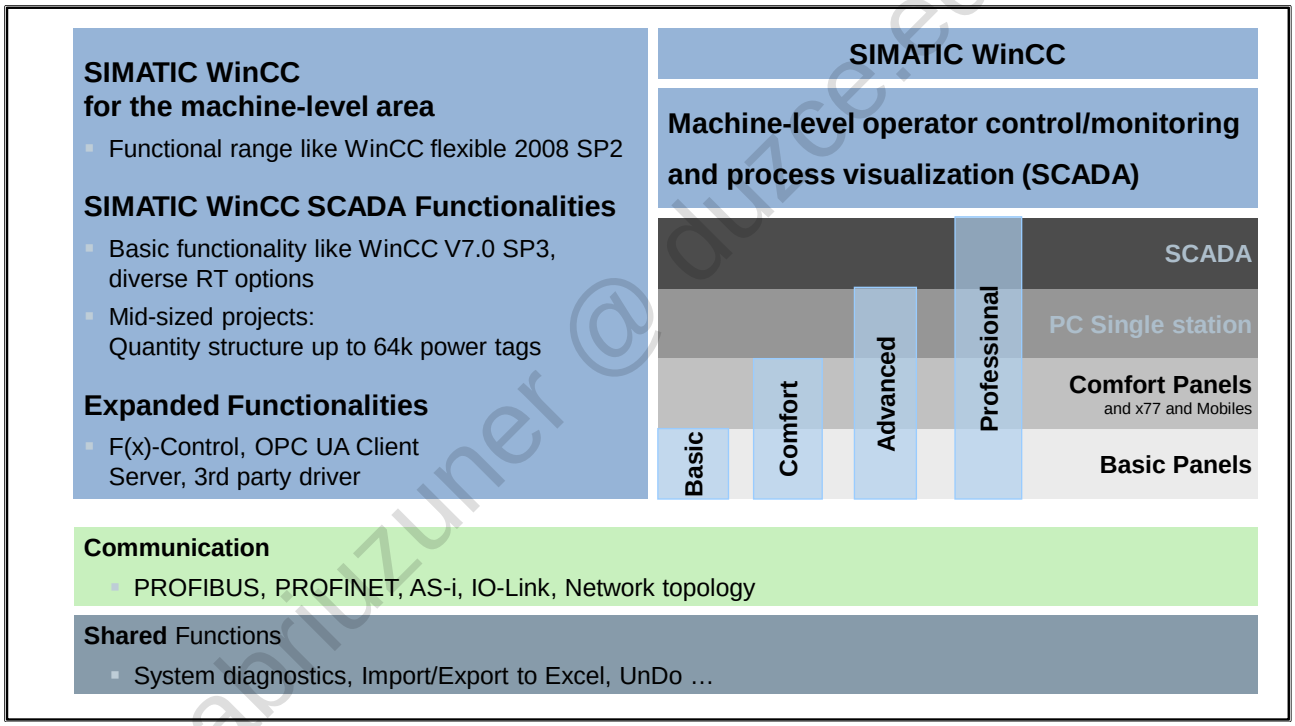
The Totally Integrated Automation Portal constitutes the working environment for an integrated engineering with SIMATIC STEP 7 V16 and SIMATIC WinCC V16.

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

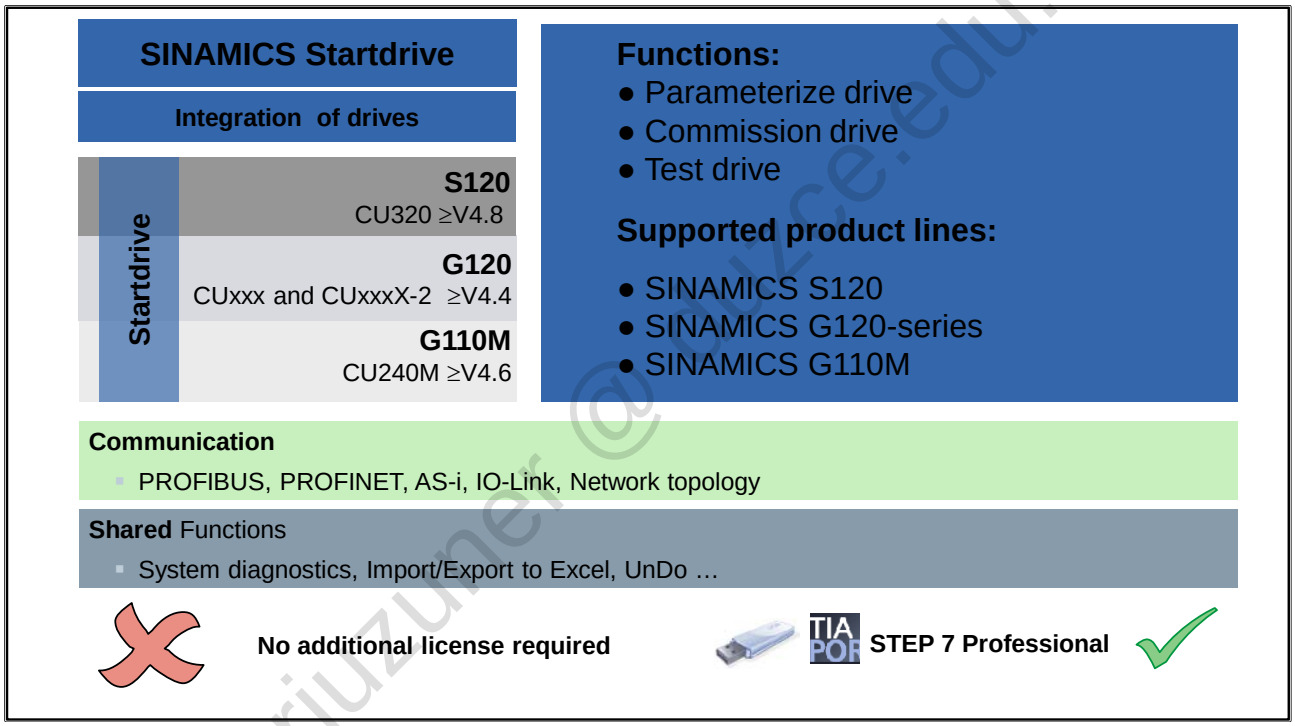
3.4.1. STEP 7 Range of Products



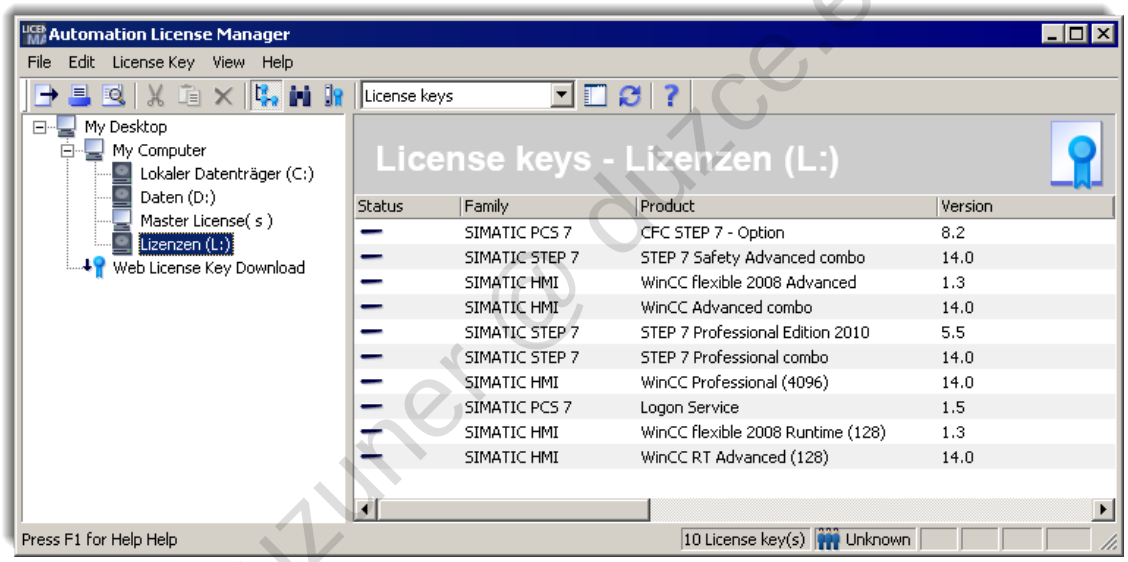
3.4.2. WinCC Range of Products



3.4.3. Startdrive Range of Products and Licensing



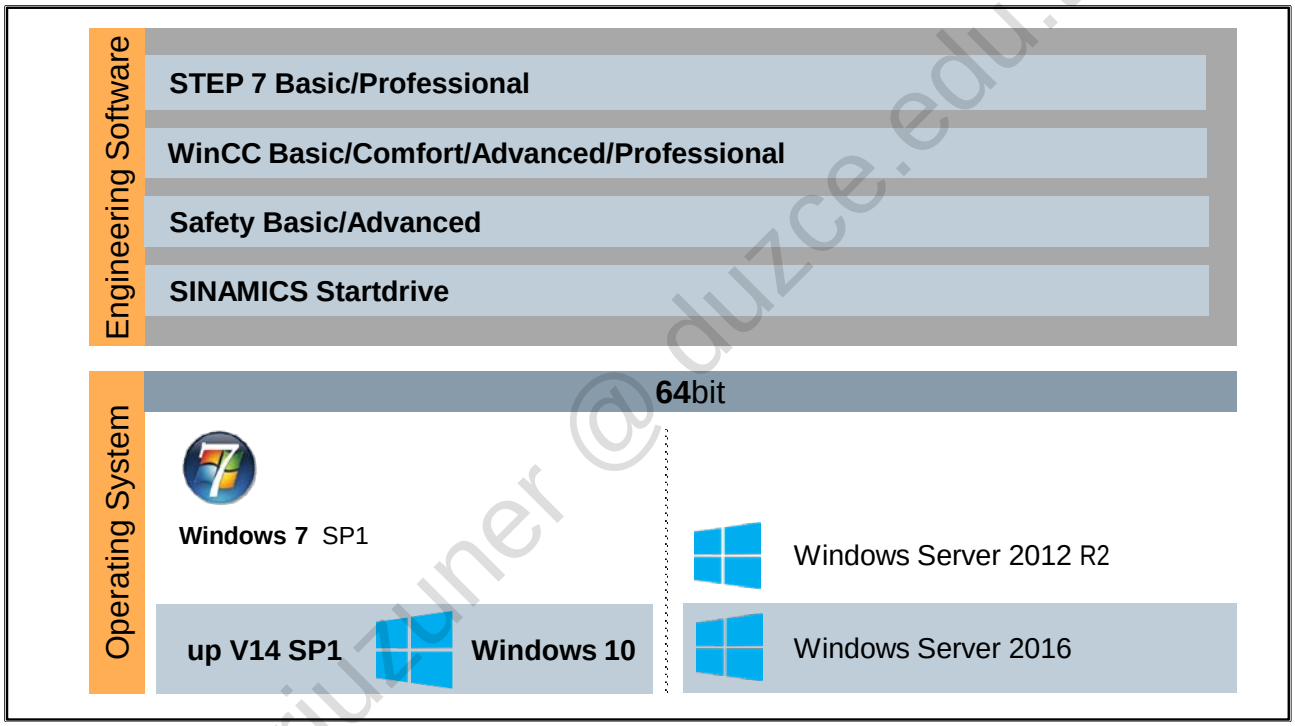
3.5. Automation License Manager



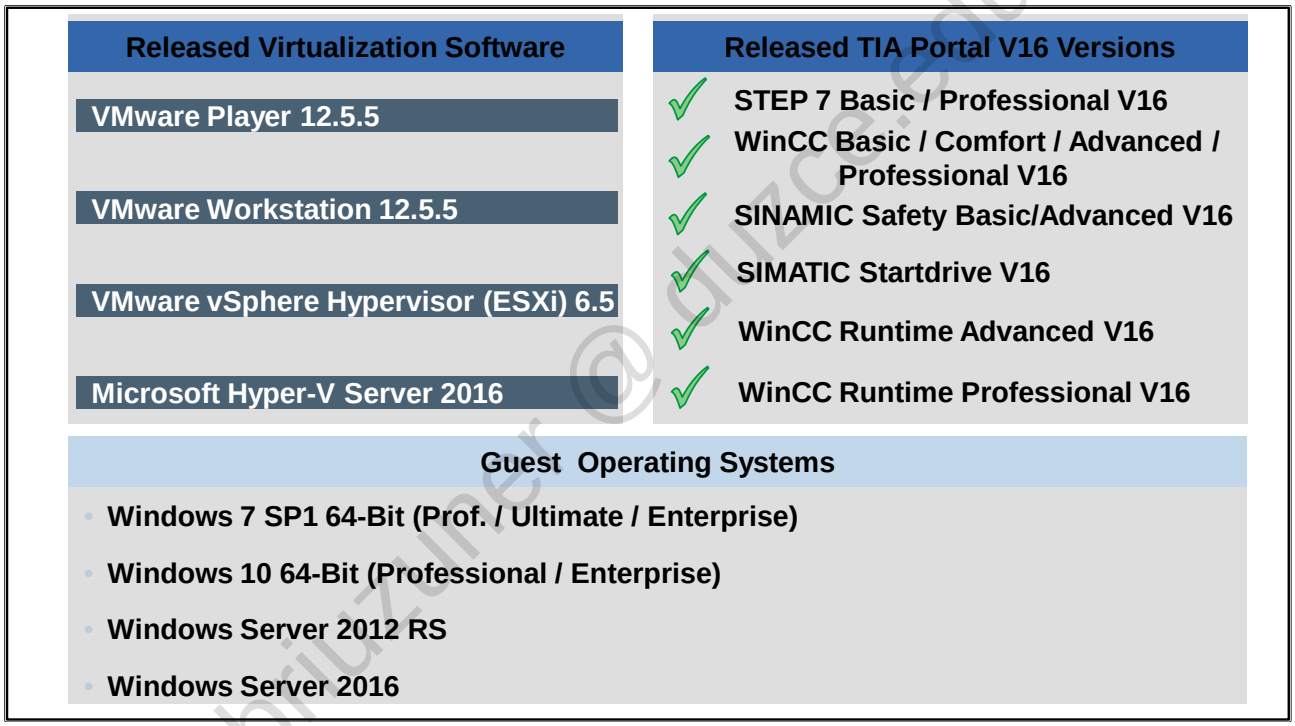
The software for managing, controlling as well as the acquisition of new licenses.

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)



Virtualization

TIA is released for usage in virtual machines. It gives you the possibility to have different versions installed separately on one physical machine. This is necessary with WinCC Professional or with older versions of TIA Portal which do not run on the actual operating system.

3.5.3. Side-by-Side Installation

TIA Portal V16, V15, V14 and V13 can be installed on one and the same PG/PC

V14, V15 and V16 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

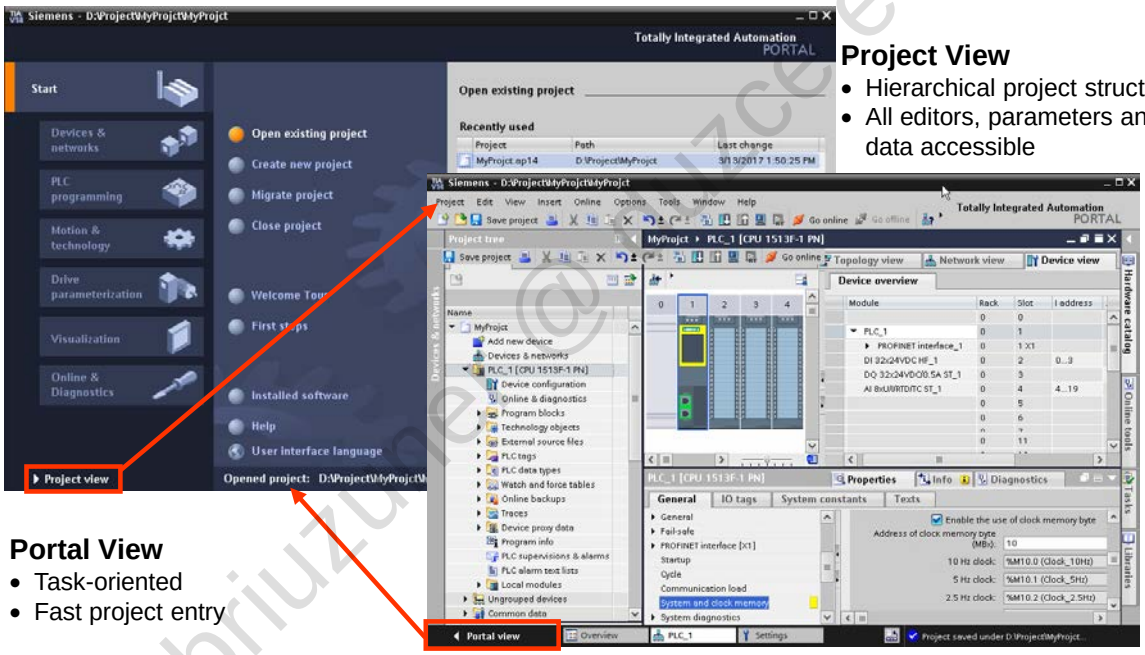
Compatibility Tool for Automation and Drives

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.

Note

To install STEP 7 V13, V14, V15 and V16 in WINDOWS 10, STEP 7 V13 SP2 is released for WINDOWS 10.

3.6. TIA Portal: Portal View and Project View



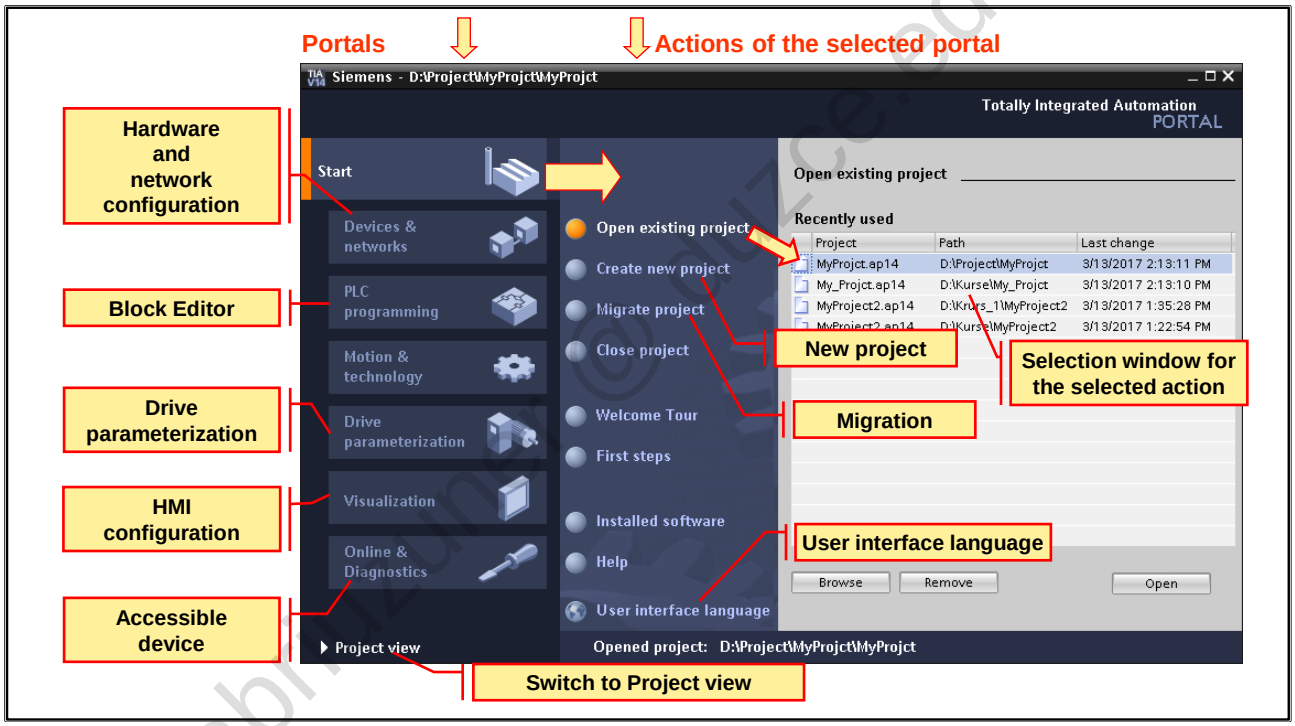
Portal View

- 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

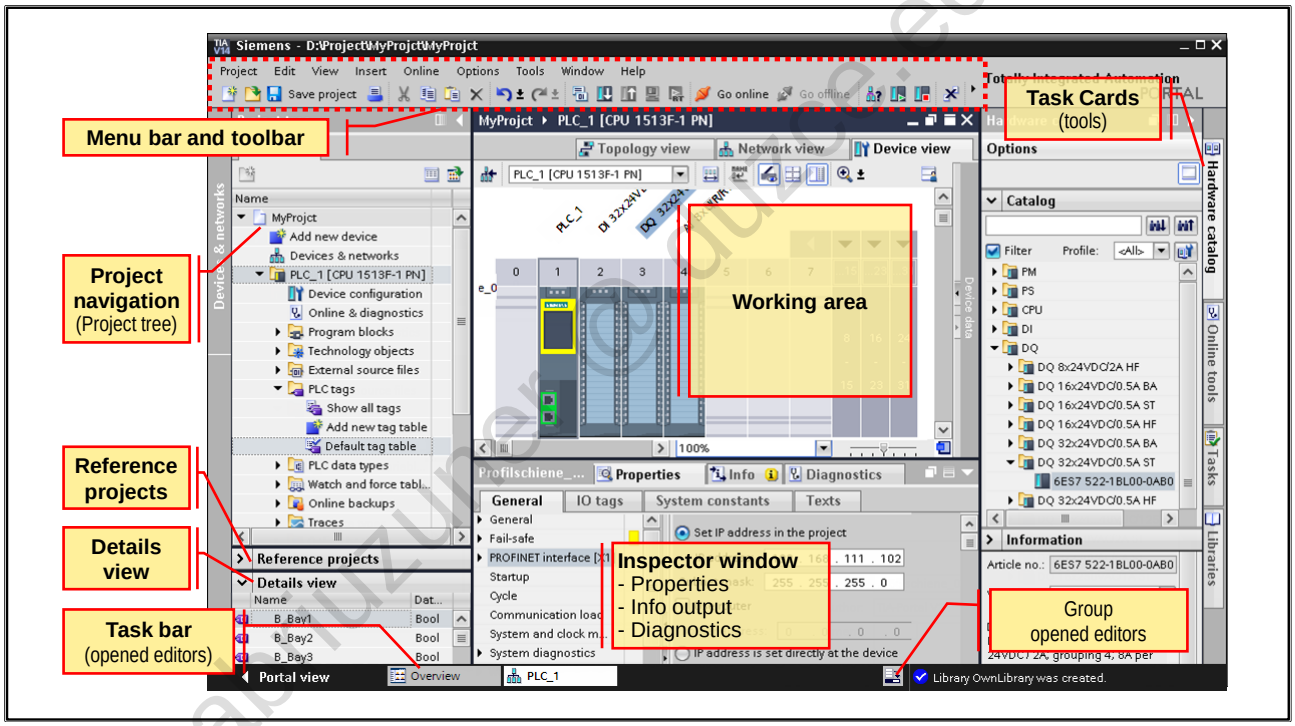
Actions

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

Task Cards 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.

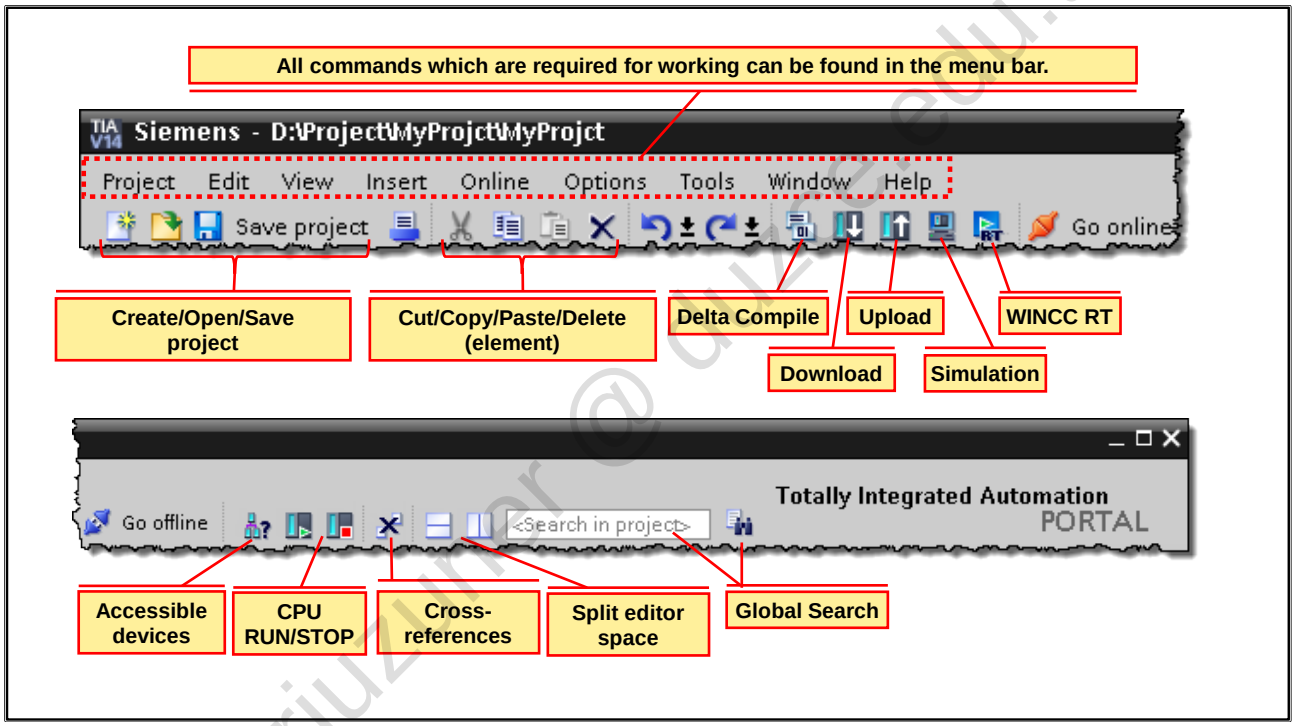
Details View

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



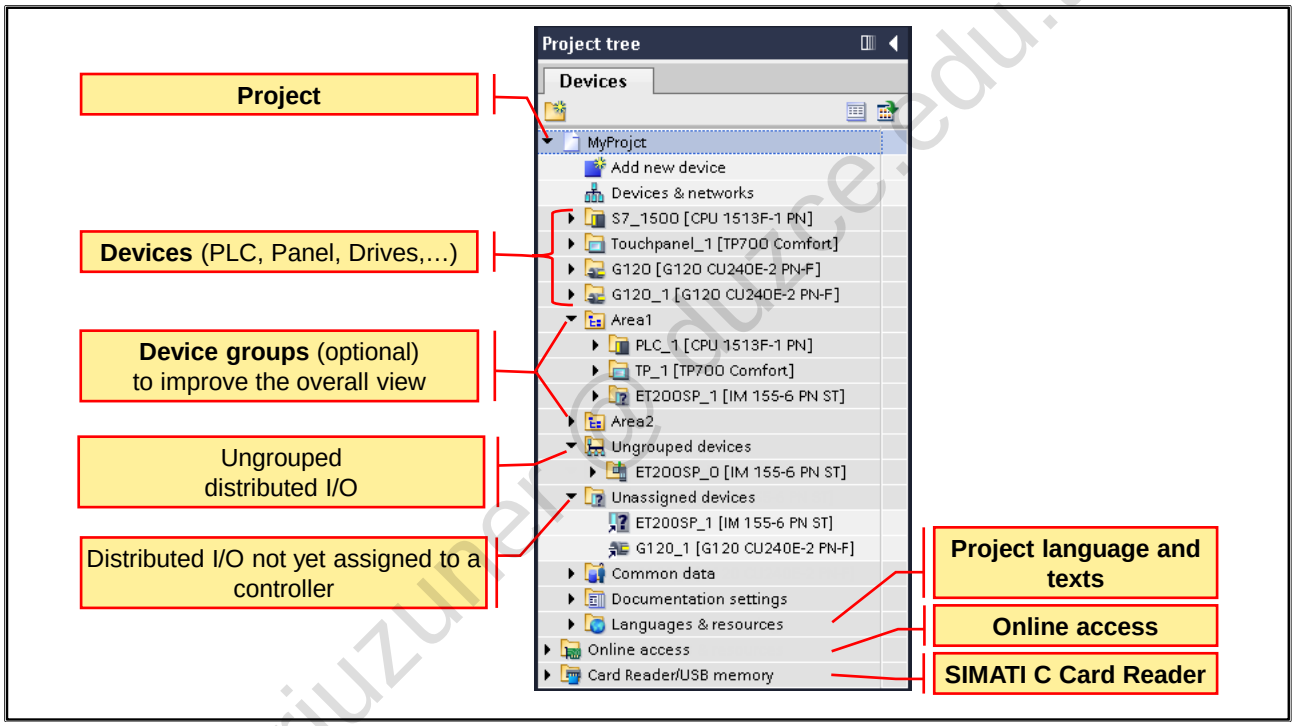
Menu Bar

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

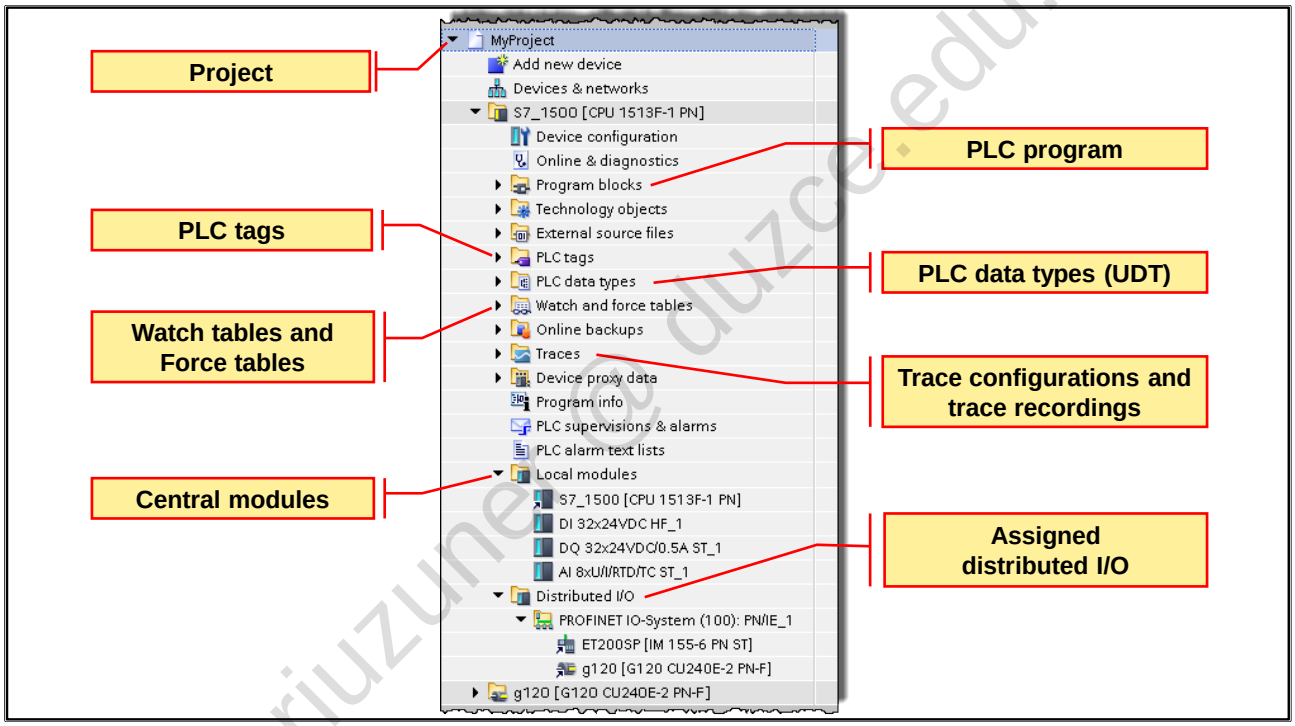
To improve clarity, objects (entire stations) can be grouped together.

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”.

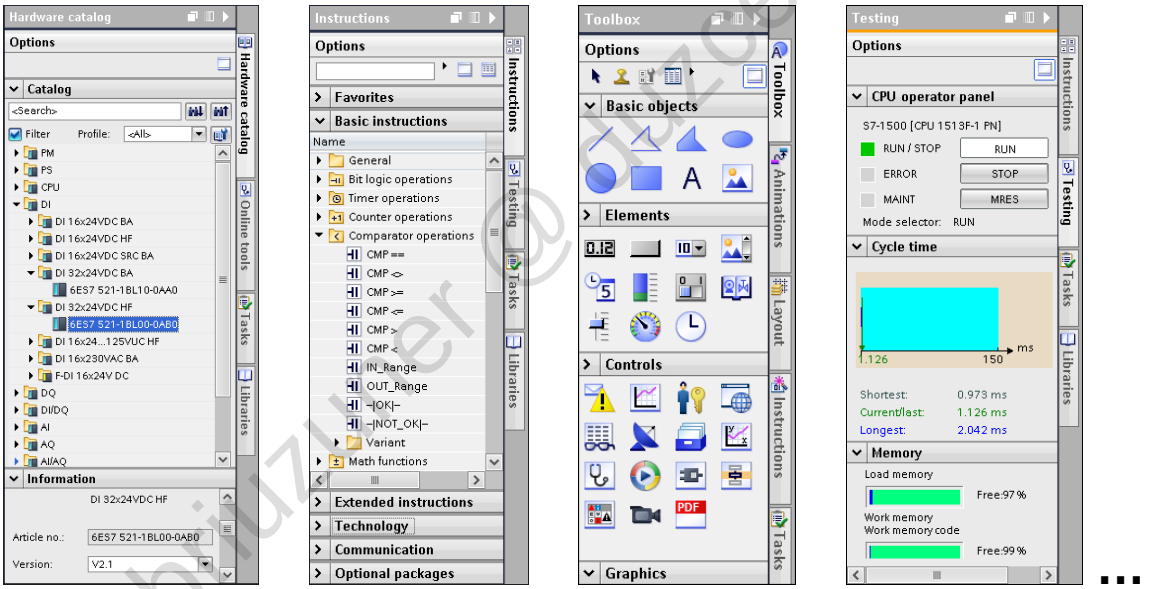
If a device or slave was assigned to a controller or master, the device can be found in the folder “Distributed I/O” of the relevant controller/master.

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

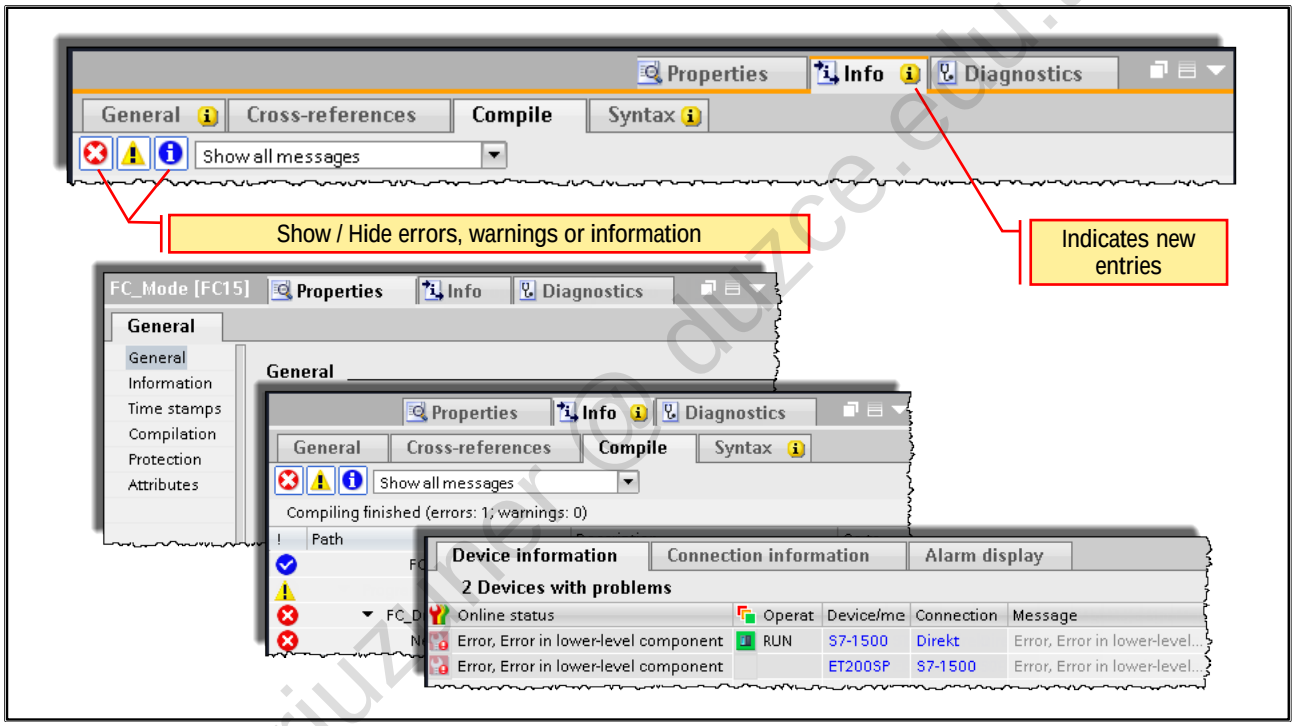
Different Task Cards are available depending on the edited or selected object!



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 all Task Cards are not visible, the Task Card-bar can be shifted using the cursor buttons at the bottom right.

- **Hardware Catalog**
  All the available hardware components (such as CPUs, modules etc.) can be selected in the Hardware Catalog
- **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. Also, 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**
  Classic editor functions (such as find and replacing tags, instructions etc.) are available such here
- **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.

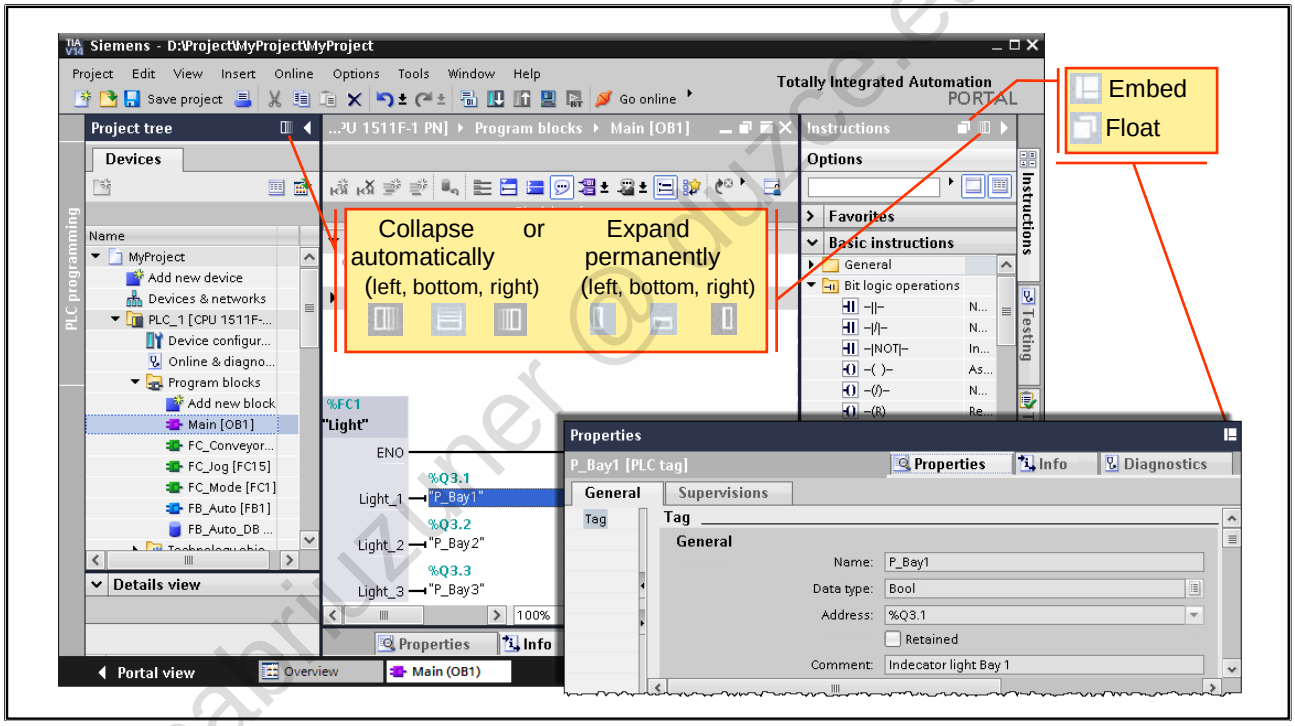
"General" tab → general status output
"Cross-references" tab → display of the current locations at which the selected object is used
"Compile" tab → status display of compilation progress
"Syntax" tab → status display for invalid programming commands

"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. Window Arrangement



The positions and characteristics of windows can be configured individually. You can hide windows that are seldom required to enlarge the working area.
The current configuration of the engineering user interface is saved in the user profile of Windows. When the project is saved, the positions and characteristics of windows are automatically saved with it.

Window Arrangement Options

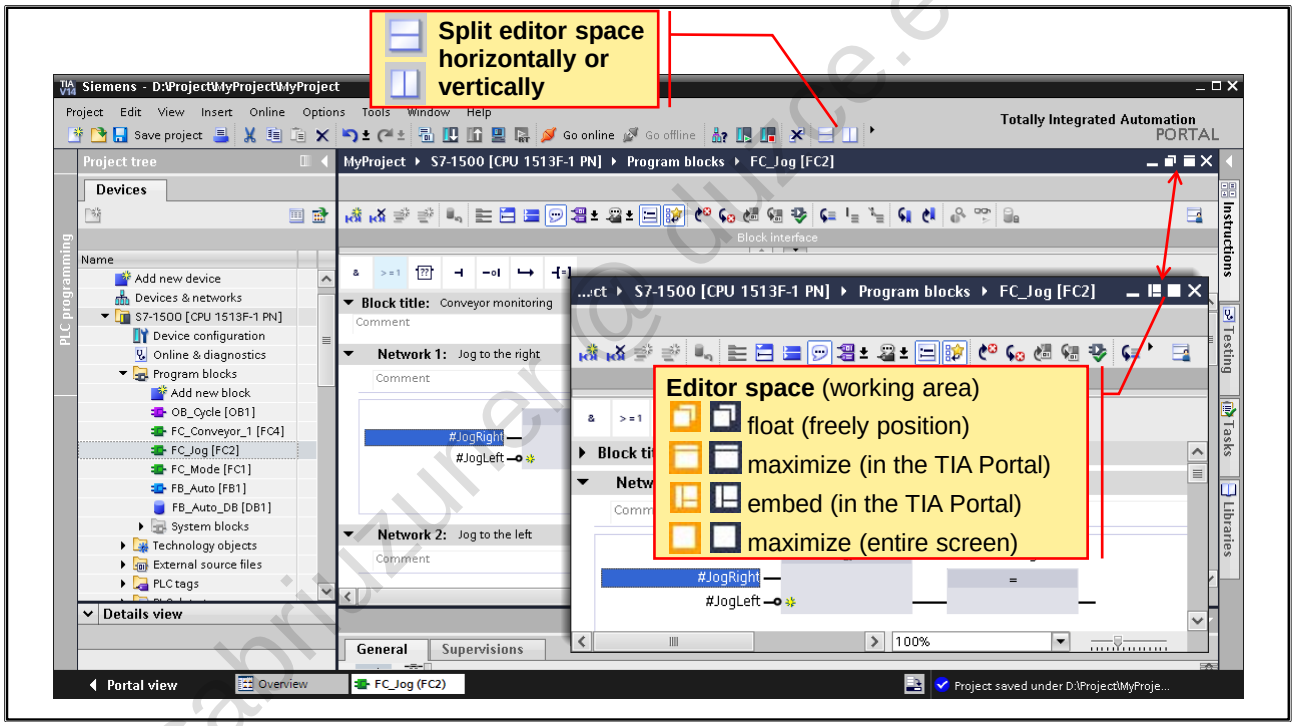
- When the window is ‘Expand permanently
  - fixed location and fixed size on user interface
  - position at left, bottom or right outside of the working area is possible
  - always open, reduces the working area
- When the window is ‘Collapse automatically
  - hidden at edge of user interface
  - position 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.

3.7.1. 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)

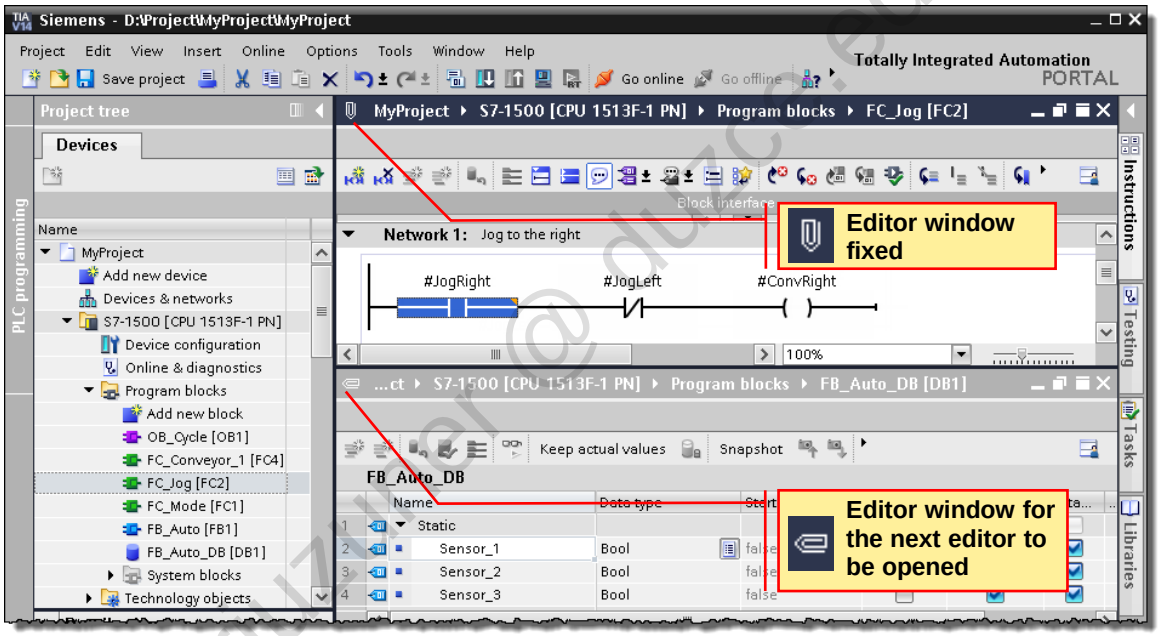
Float or release a window from the working area (color depending on View online/offline)

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

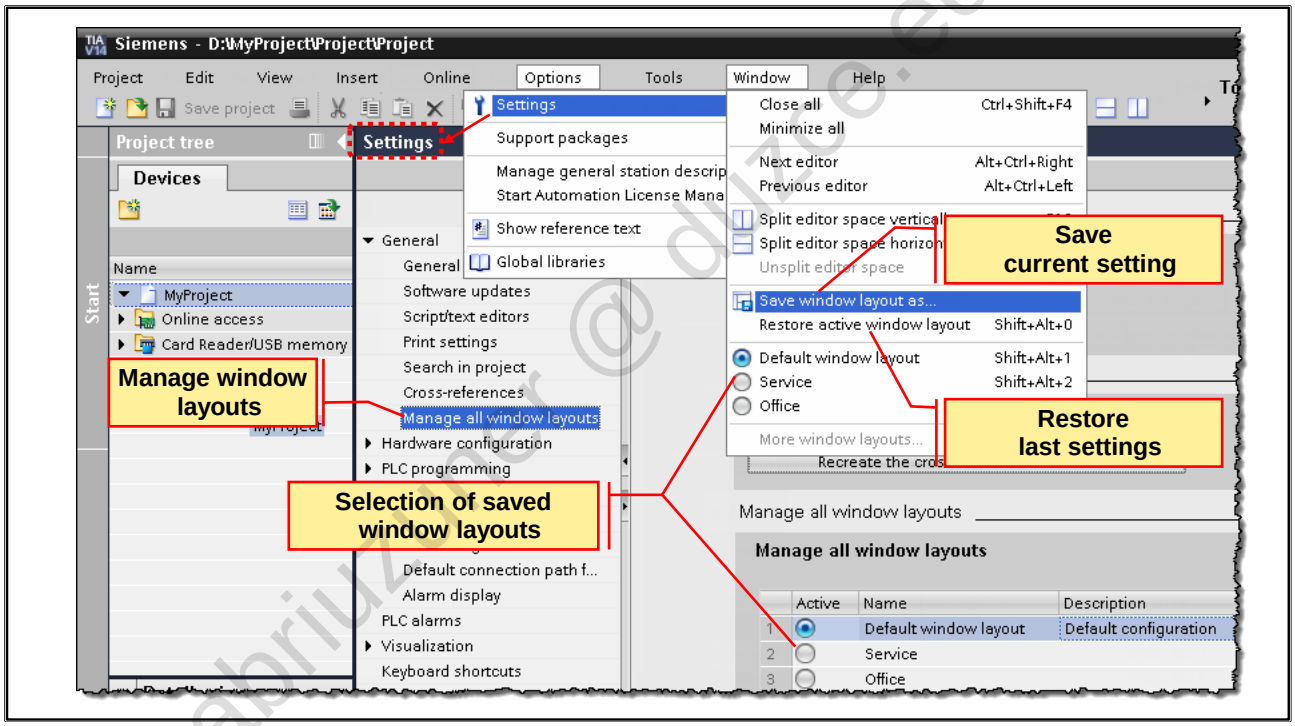
Split the editor space (working area) vertically into two windows

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



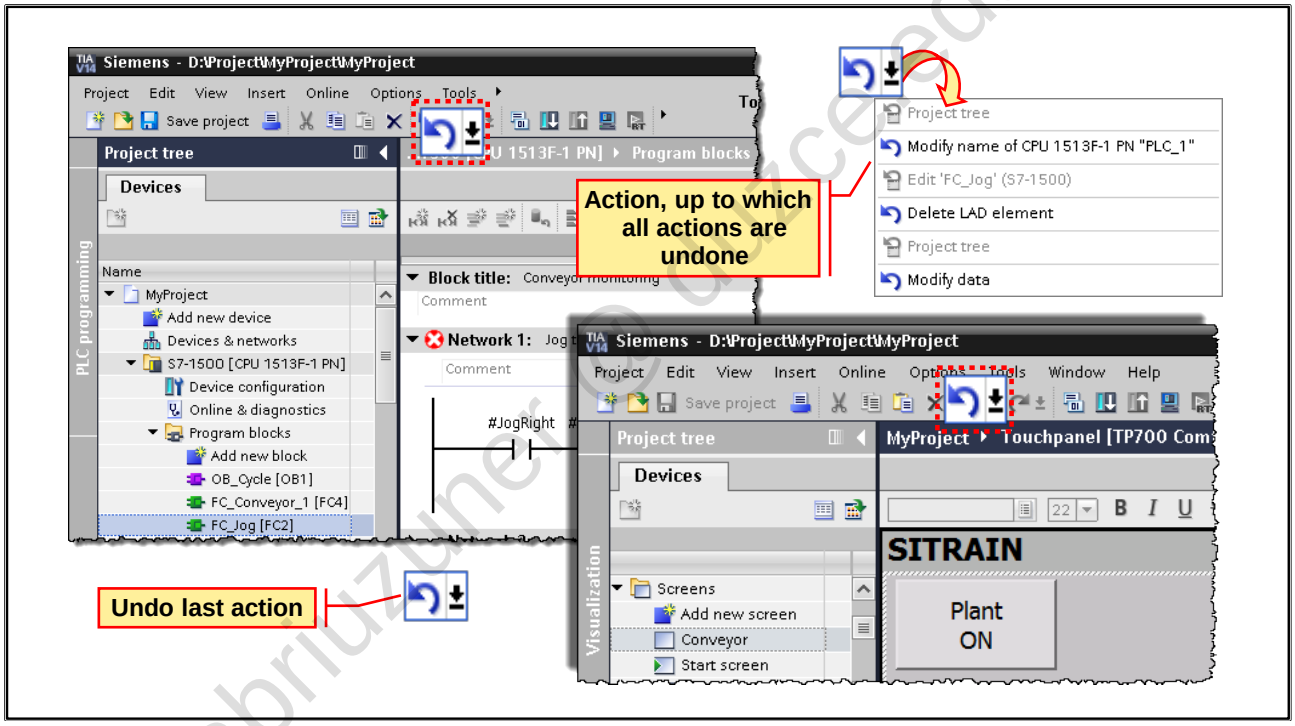
If you work with a split editor space (working area), one of the two working areas can be fixed (attached) by clicking on the “paper-clip” (paper-clip is vertical) so that when you open a further editor, this first one always remains fixed in the foreground and the newly opened one always becomes the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

3.7.3. Save / Manage / Use Window Layouts



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

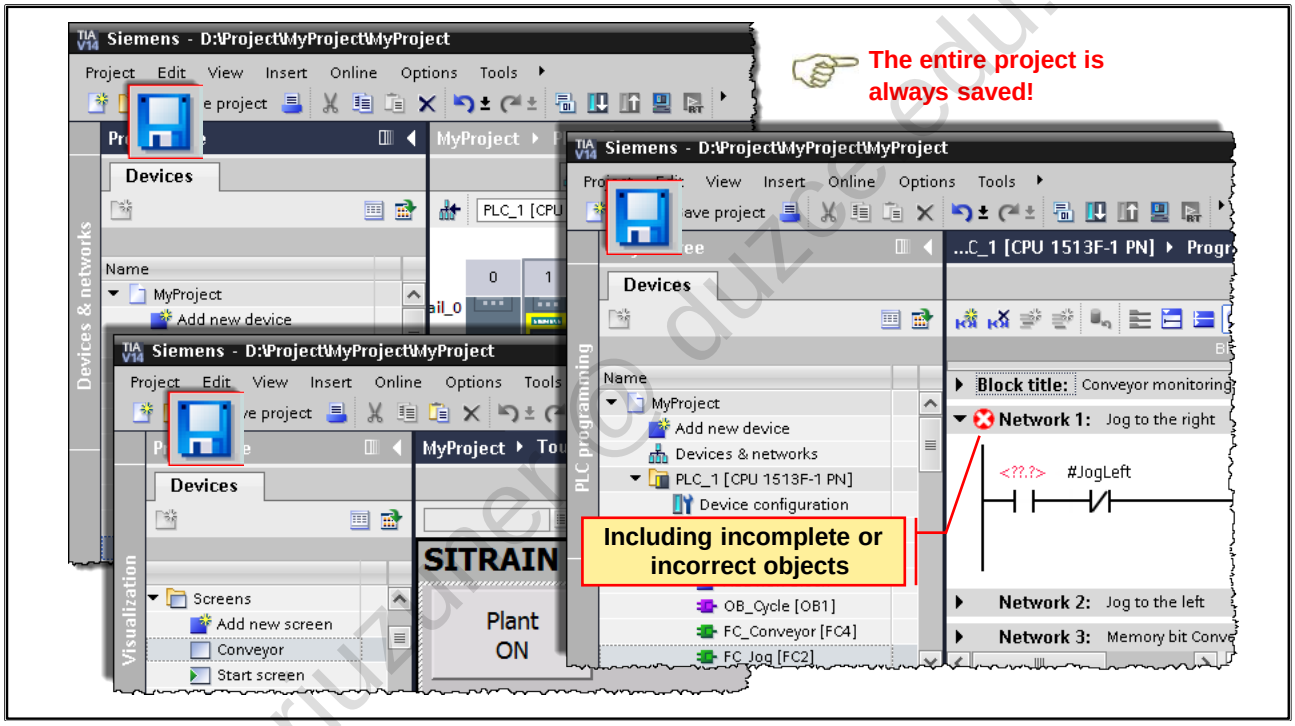
3.8. Undo and Redo



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. Because all actions are only undone up to the selected action, the consistency of the project is ensured.

3.9. Saving a Project

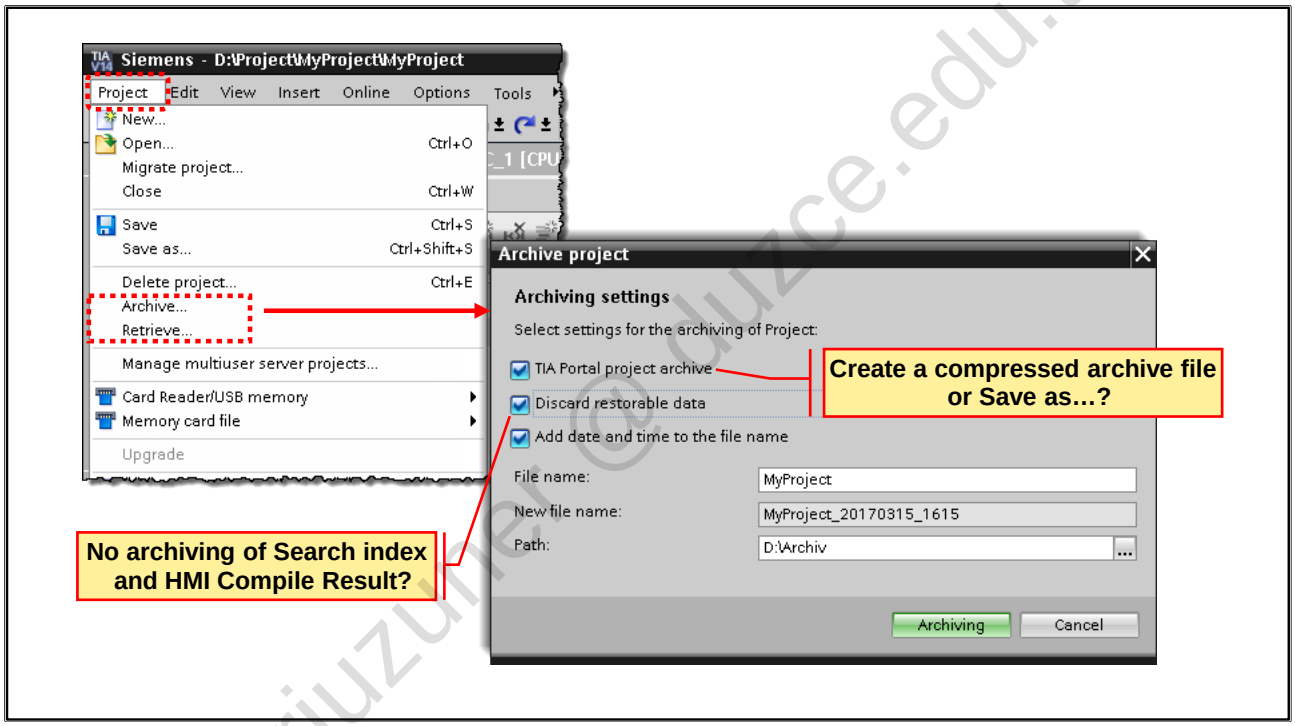


Save

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.

3.9.1. 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 zap16).
  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.

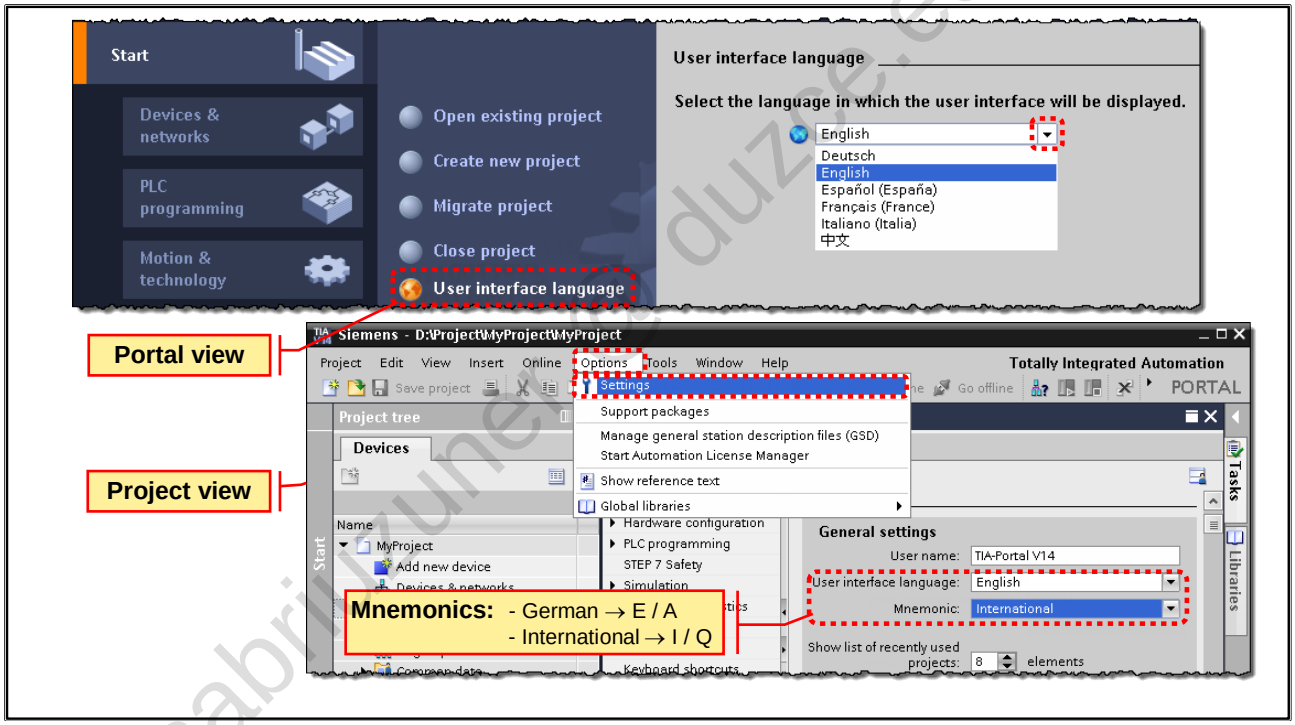
Note

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 *zap16*) can be retrieved, that is, unzipped.

3.10. TIA Portal - Settings: User Interface Language

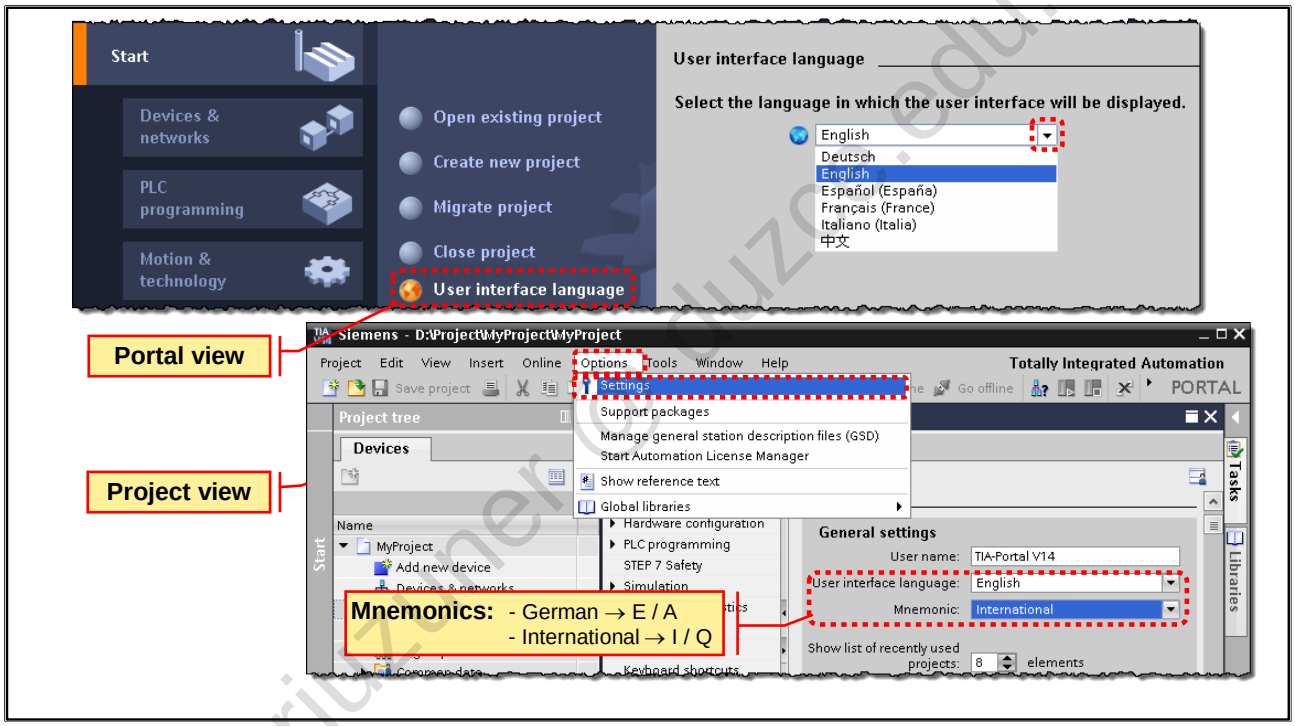


Available User Interface Languages

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)

3.10.1. 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 that was last selected.

Storage Settings

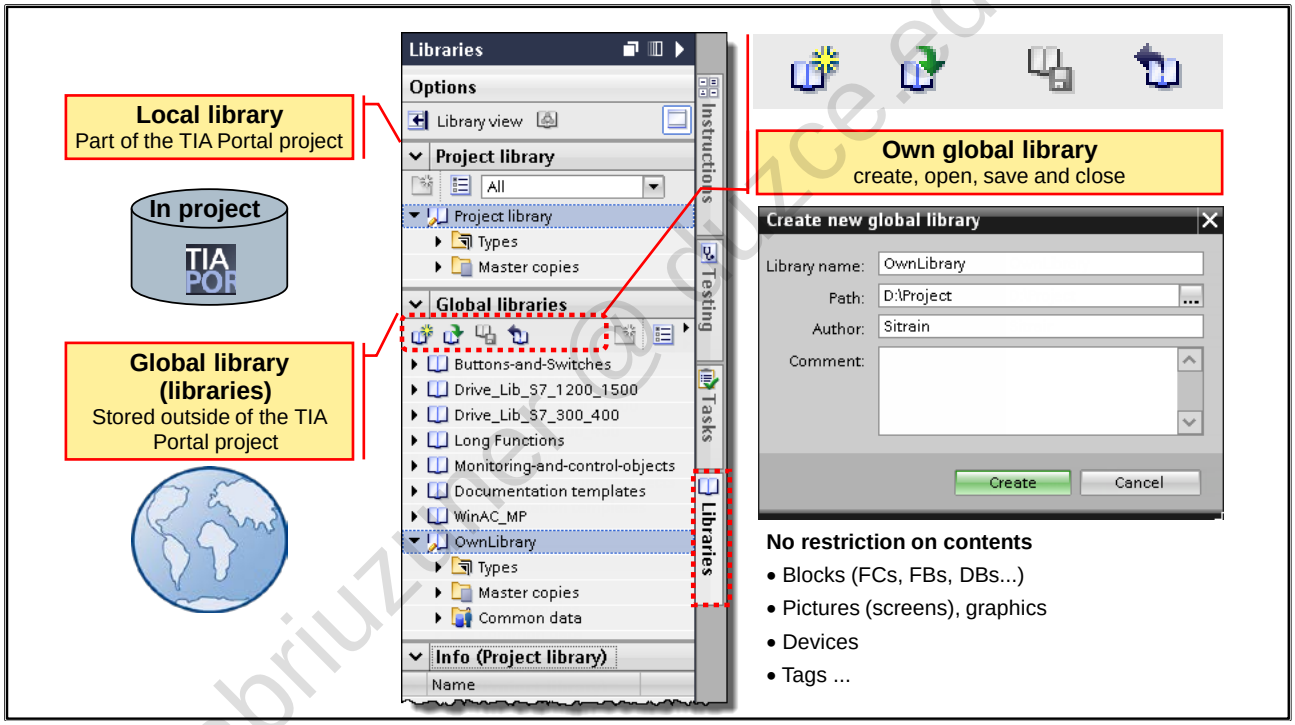
Storage location for projects:
Storage location of newly created projects and their project libraries

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.

3.11. 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

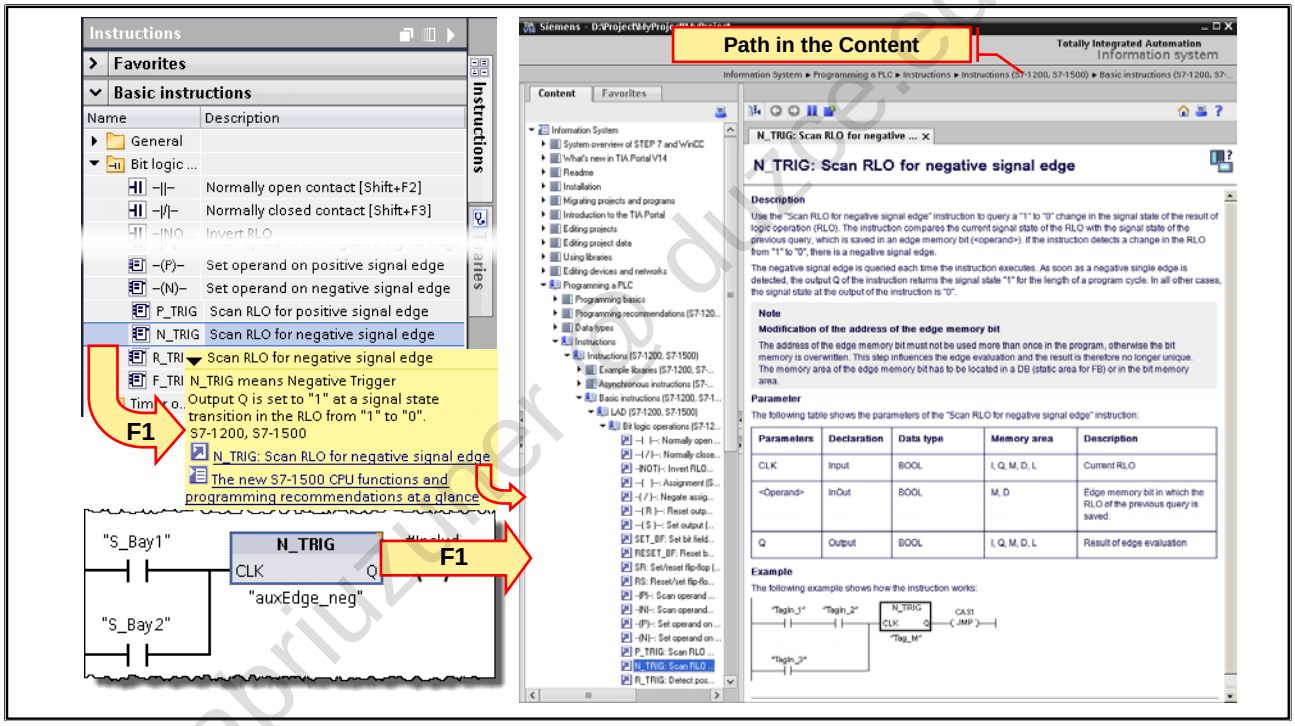
Library objects can either be used as a master copy or as an instance.

- 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

3.12. 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

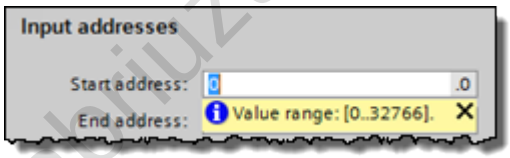
  Example: Practice-oriented application example with solution of an automation task

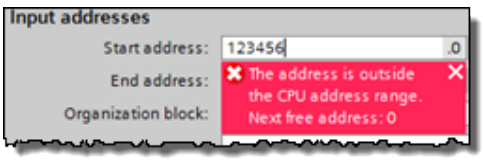
  Factual information: Background information about the functions of the TIA Portal

  Reference: Detailed reference information about instructions and objects

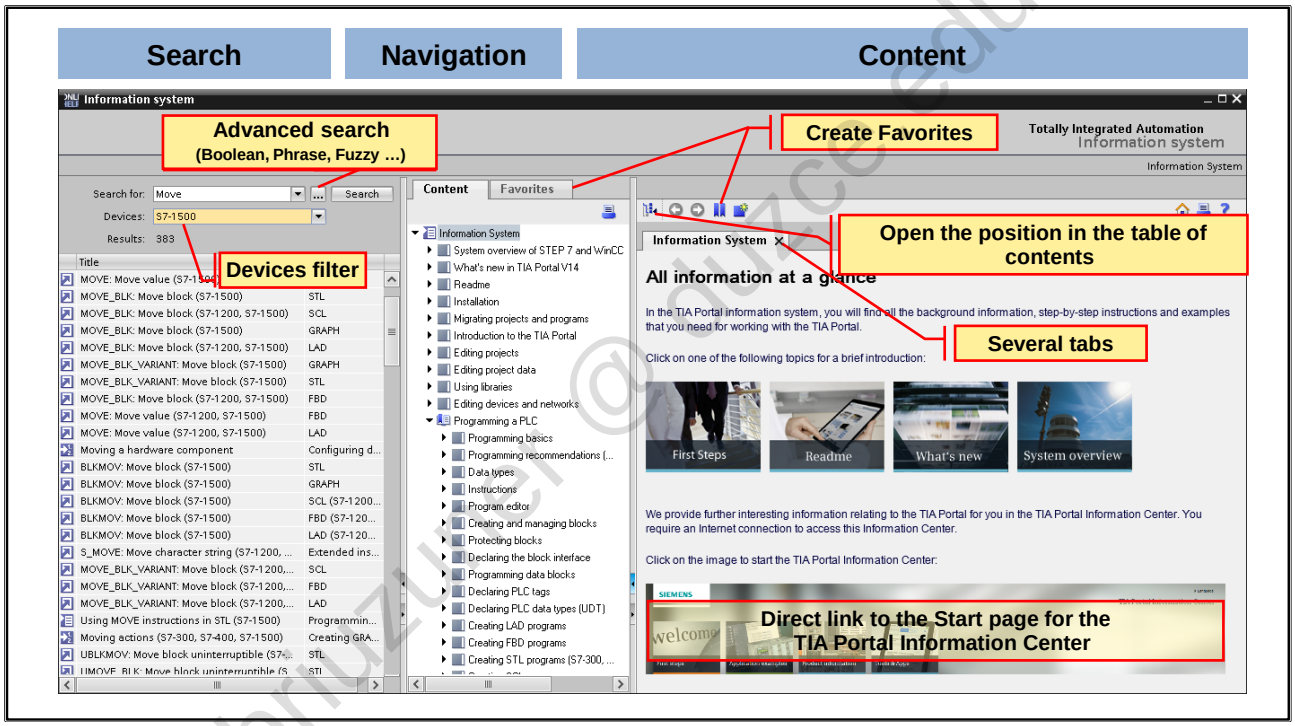
  These are activated by clicking (Information system is opened)

- 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.





3.12.1. Help (Information System)

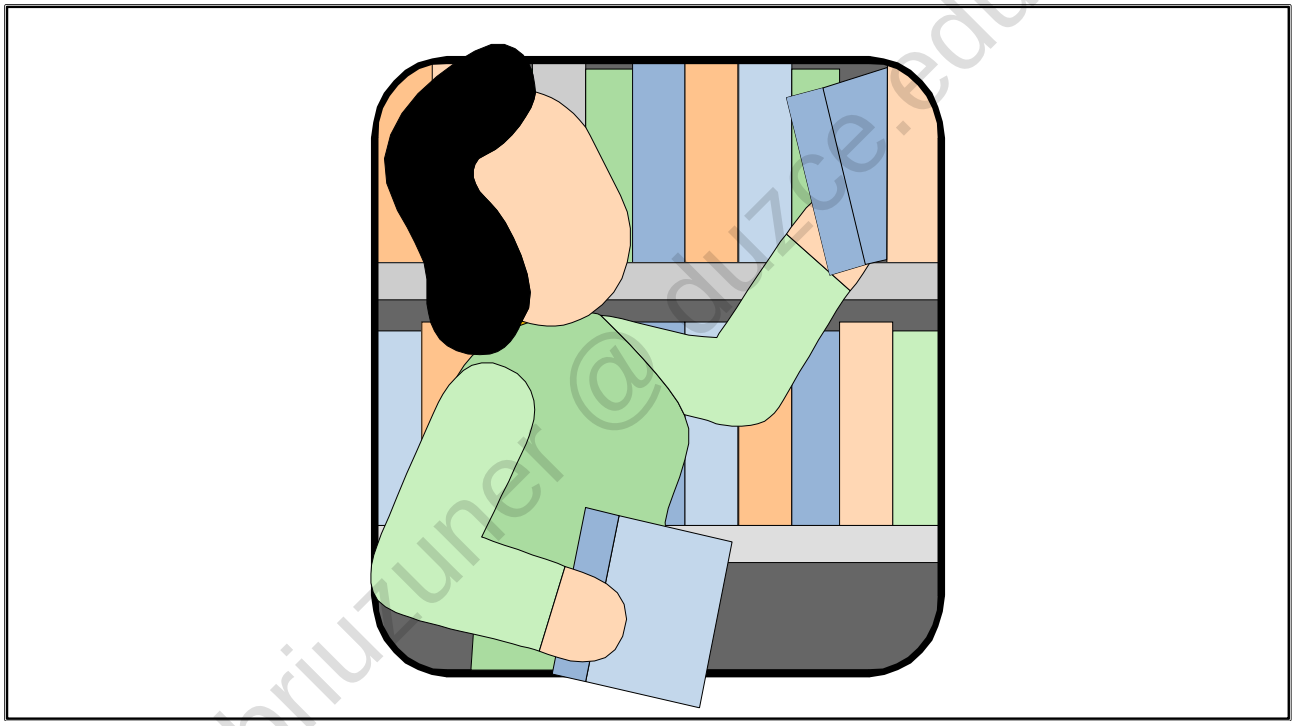


Contents of the Help Functions

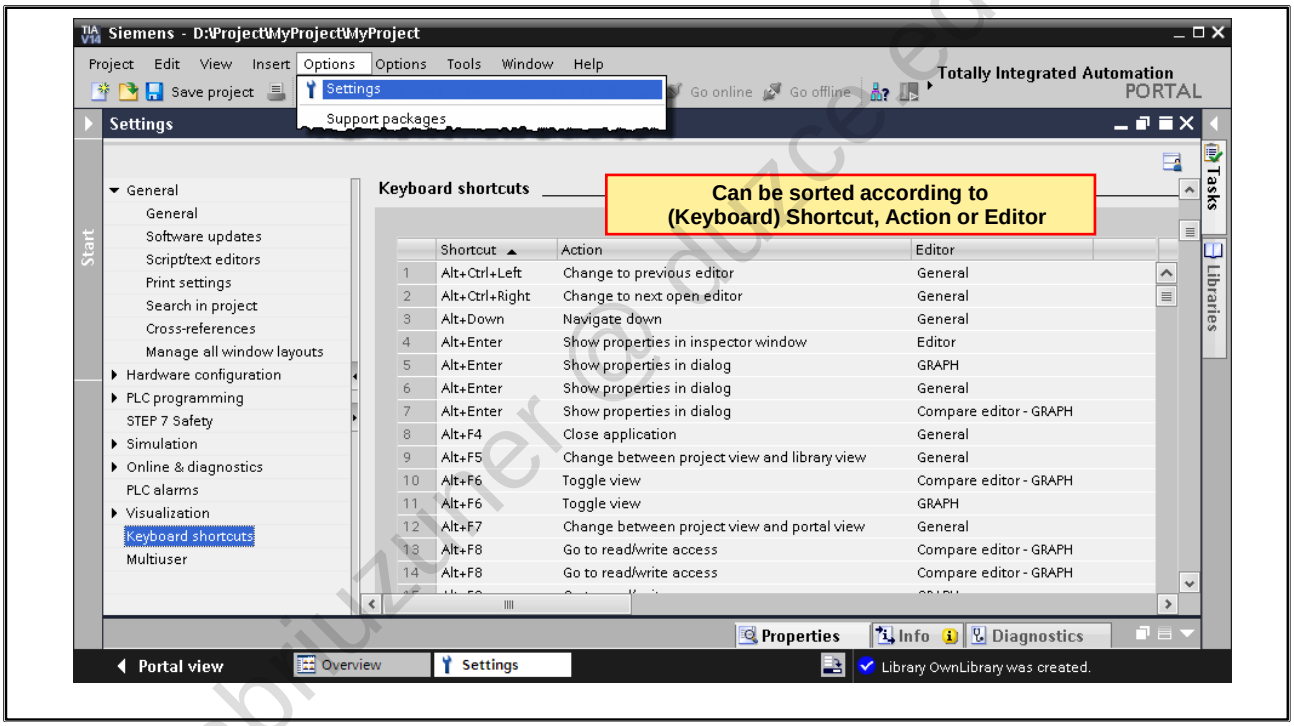
- Search area
  In the Search area, you can perform a full-text search across all help topics.
- Navigation area
  In the Navigation area, you find the (table of) Content and the Favorites.
- Contents area
  The help pages are displayed in the Content area. You can open several tabs to simultaneously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

3.13. Additional Information

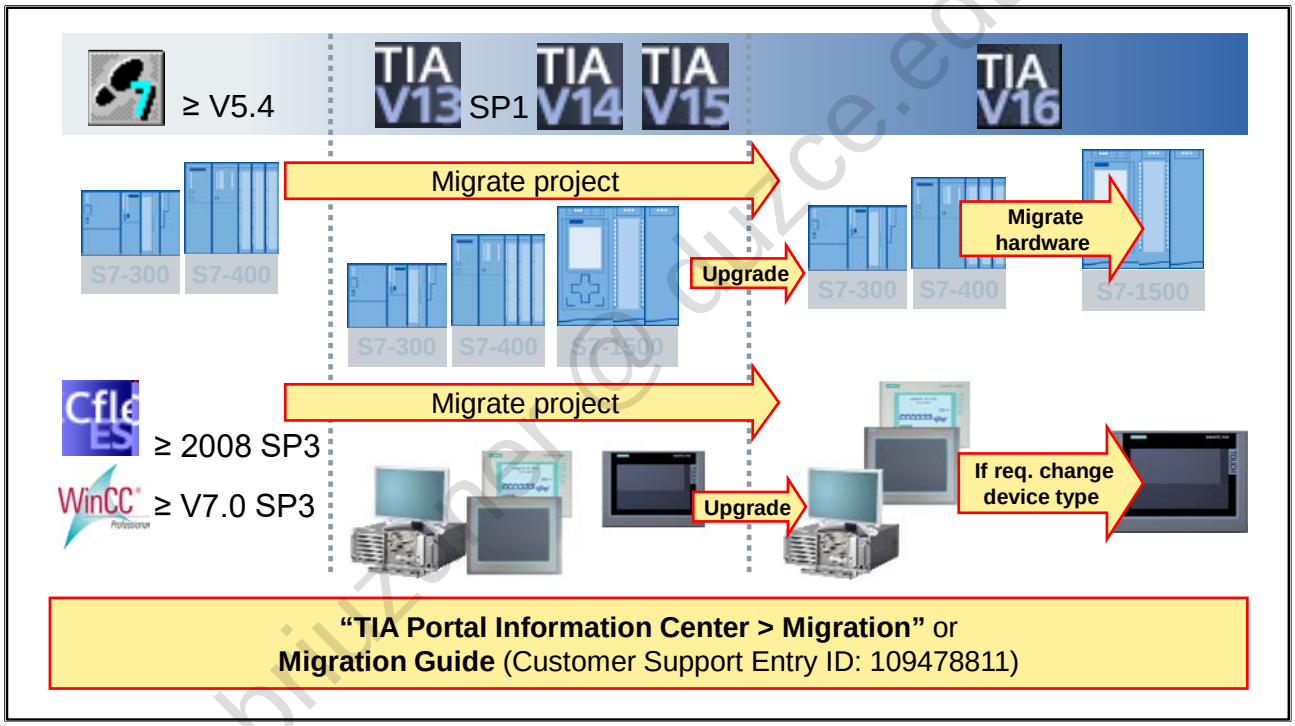


3.13.1. Keyboard Shortcuts of the TIA Portal



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

3.13.2. Project Migration



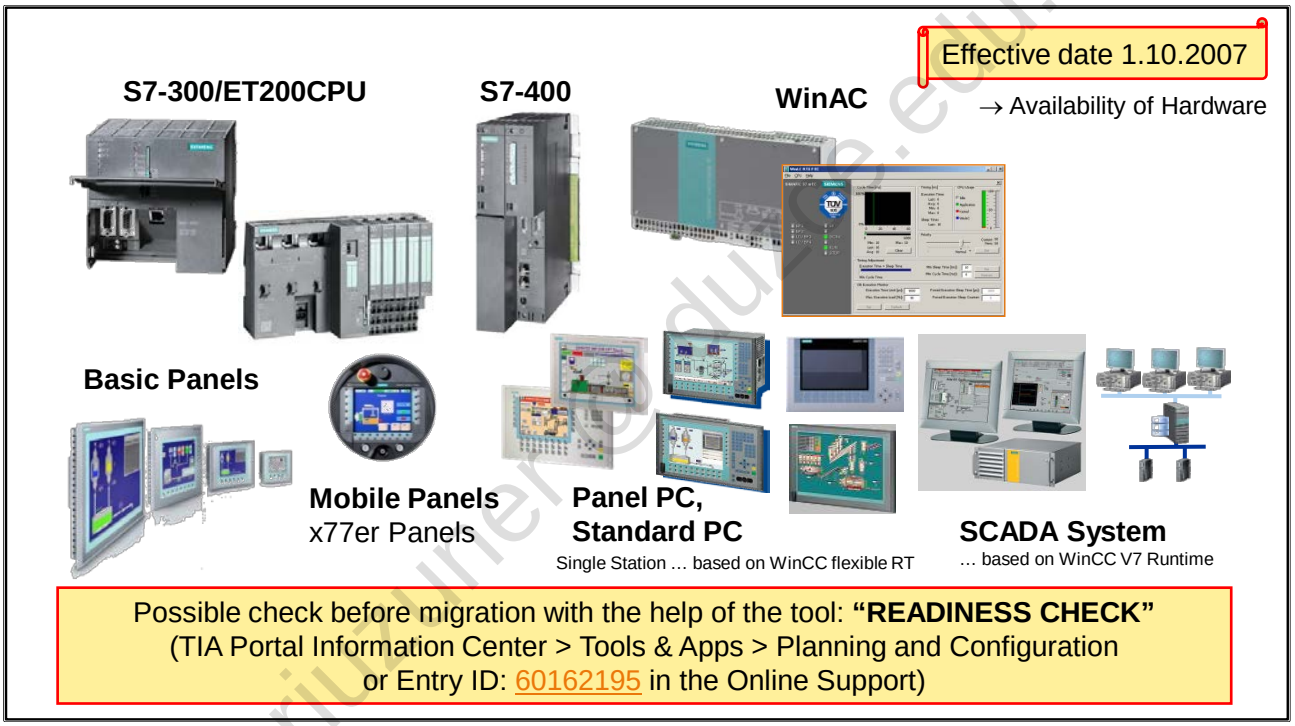
What does Migration Actually Mean?

- 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

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



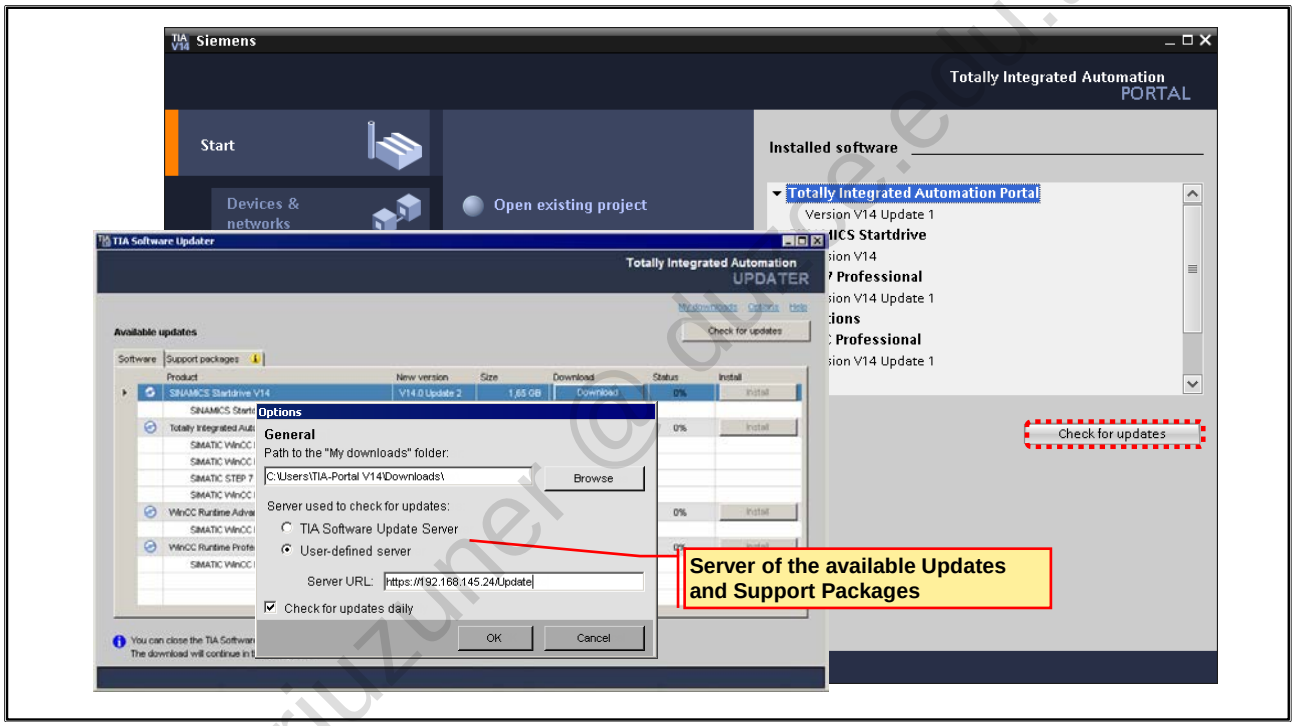
The hardware configuration for supported hardware can also be migrated. Otherwise, only the software can be migrated and must be adjusted to the supported hardware.

To check if a specific hardware item is supported by TIA Portal, 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".

3.13.3. Installation with Record Function in the Setup

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.

3.13.4. 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**

Contents

4

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

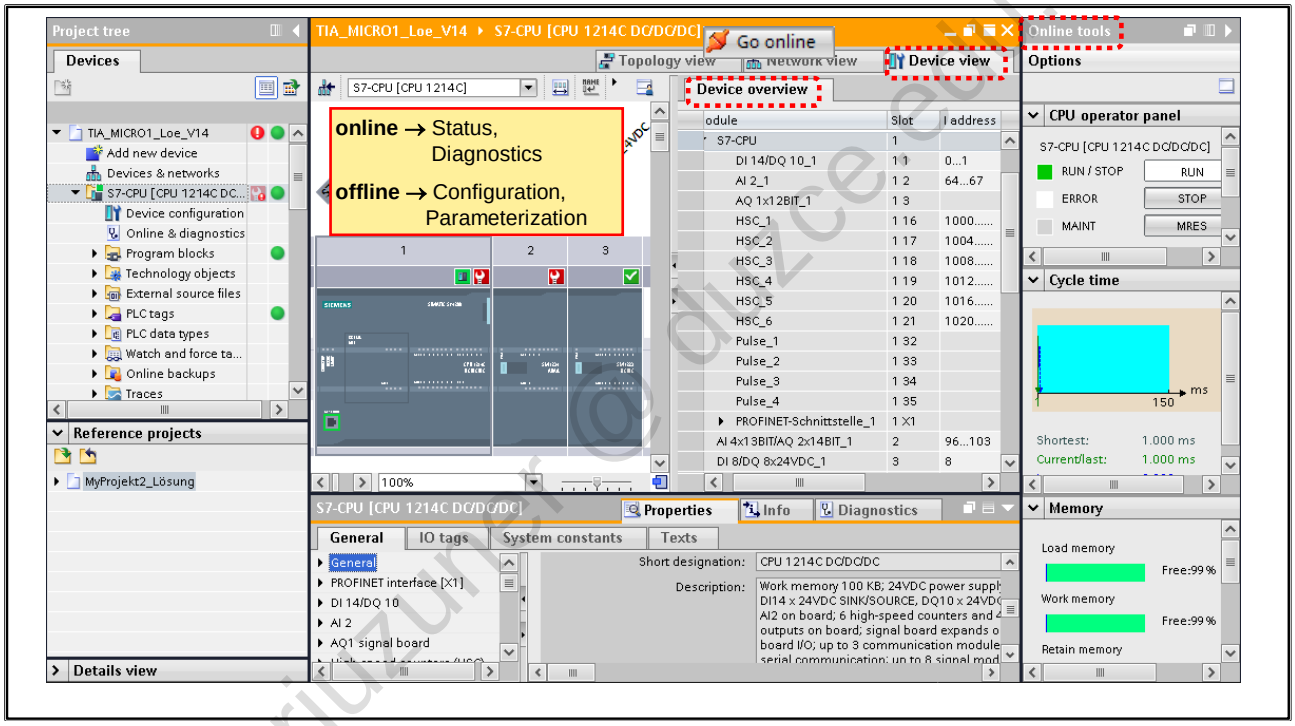
… be able to use online functions to start and stop the CPU and to reset it to factory settings

... 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 S7-1200 and be able to do it

4.1. Online Tools, Configuring and Parameterizing the Hardware



Online Tools

An online connection to the CPU permits diagnostics and status information for all modules to be accessed.

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

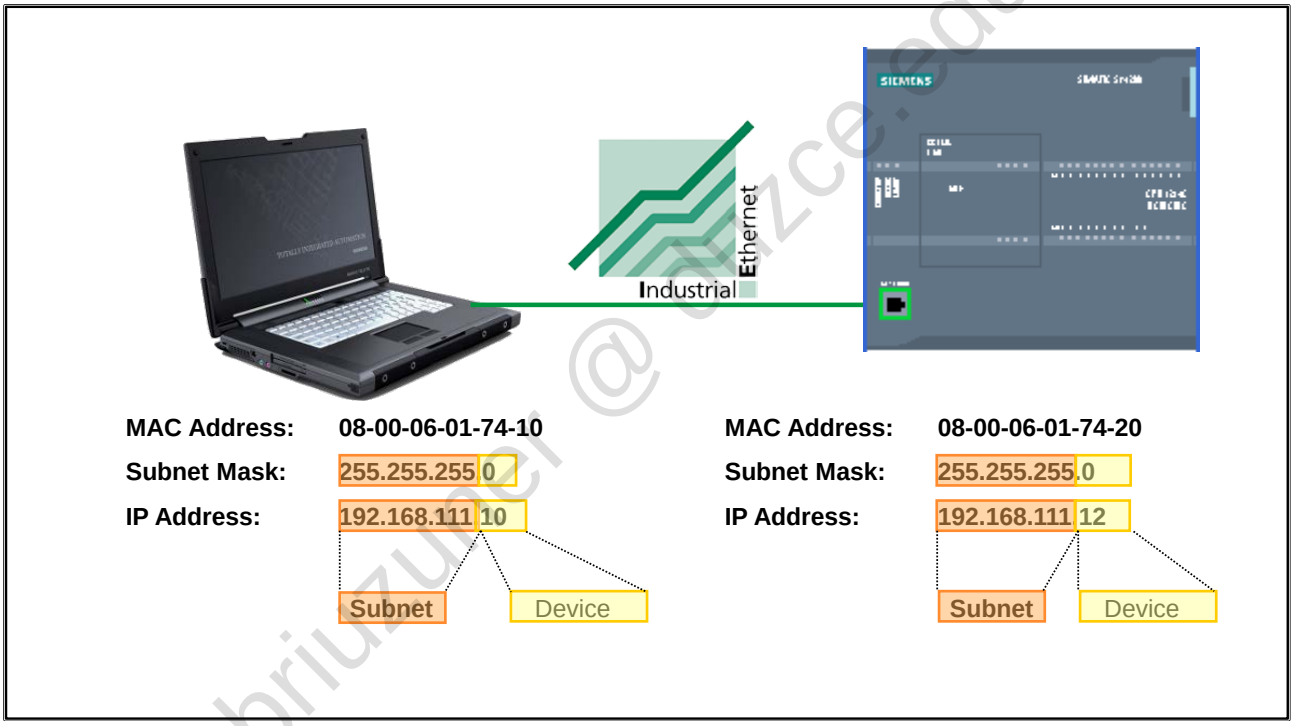
Configuring and Parameterizing the Hardware

Almost all devices or components of an automation solution such as PLCs or touchpanels 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.

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



Internet Protocol

Internet **P**rotocol (**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

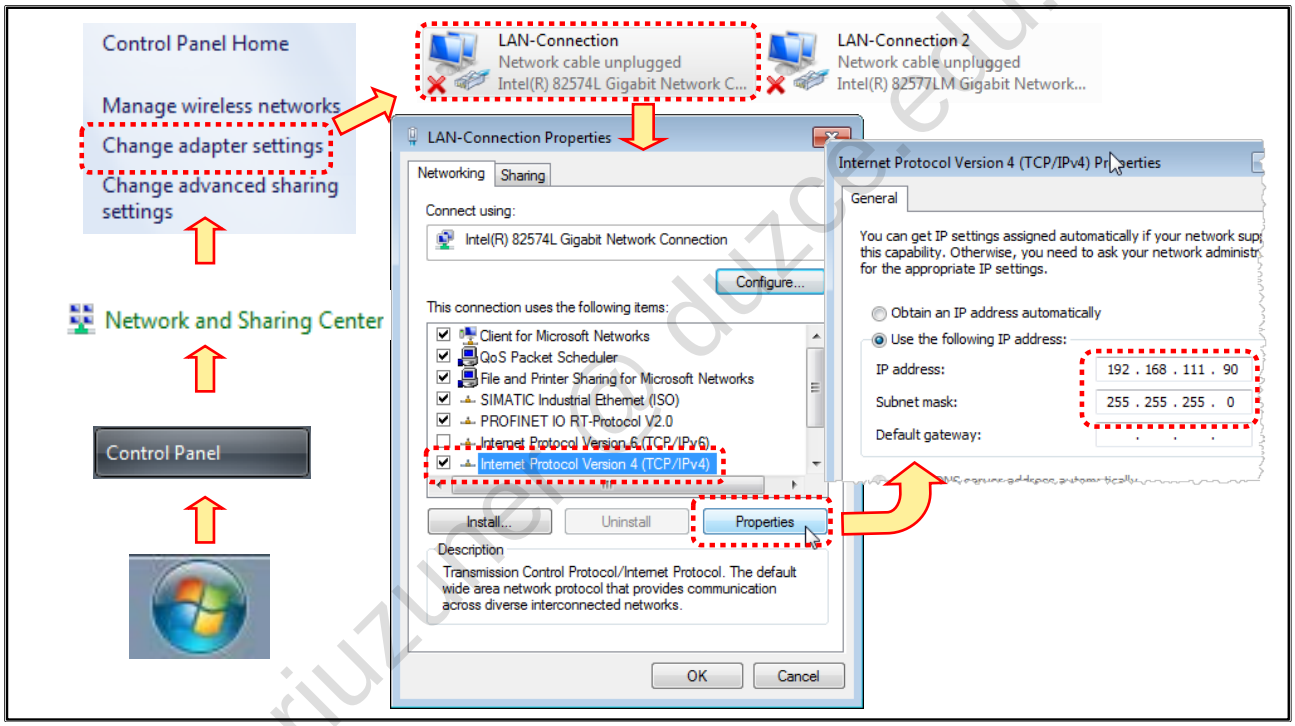
Every Ethernet interface is assigned a fixed address by the manufacturer that is unique worldwide. This address is referred to as the hardware or MAC address (Media Access Control). It is stored on the network card and uniquely identifies the Ethernet interface in a local network. Cooperation among the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

4.2.1. Online Connection: Assigning an IP Address for the PG

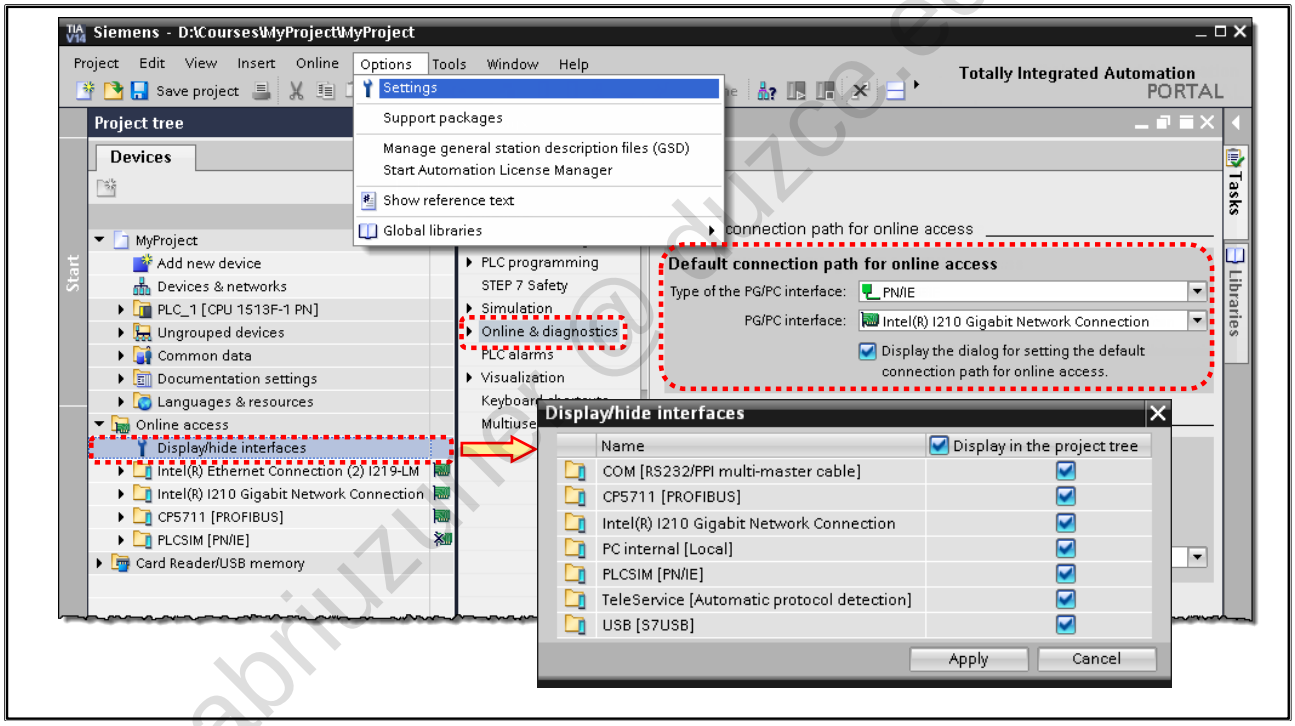


IP Address of the Programming Device

The setting of the PG's IP address can be made as shown in the picture.

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

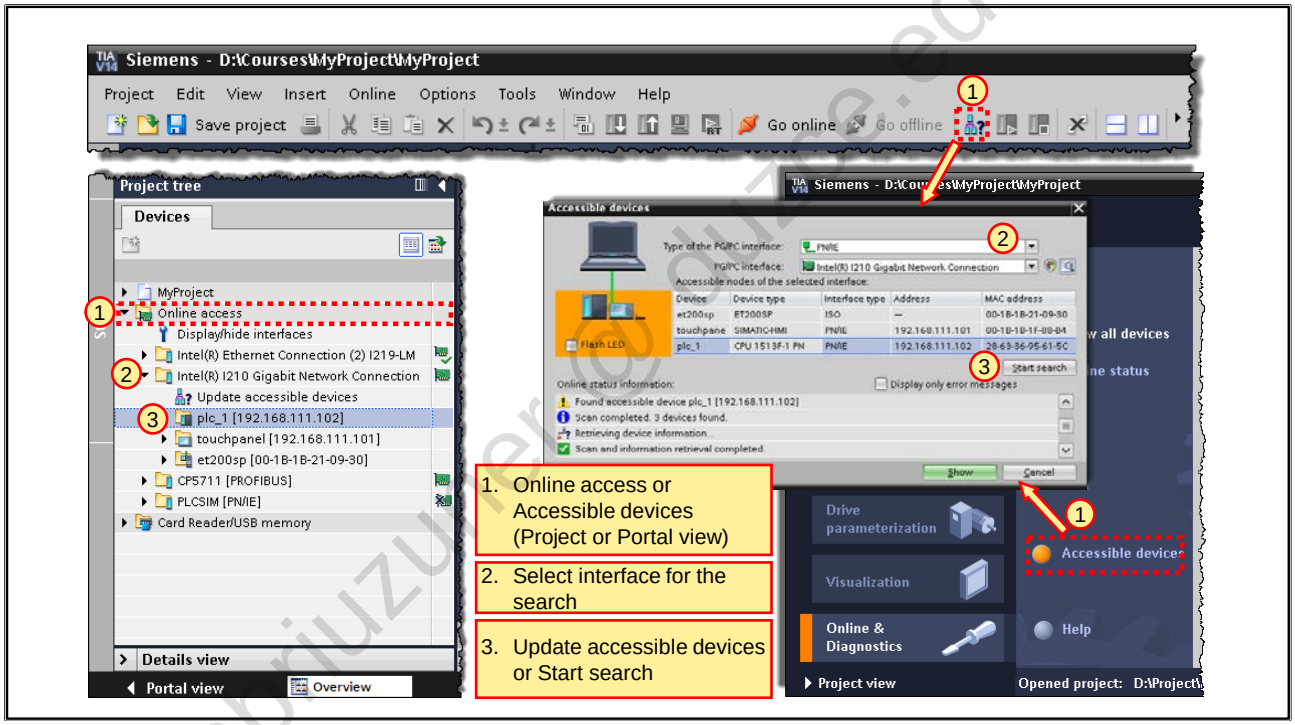
4.3. Default for Online Access and Visible Interfaces



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

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

4.4. Online Access: Accessible Devices



Accessible Devices in the Portal View

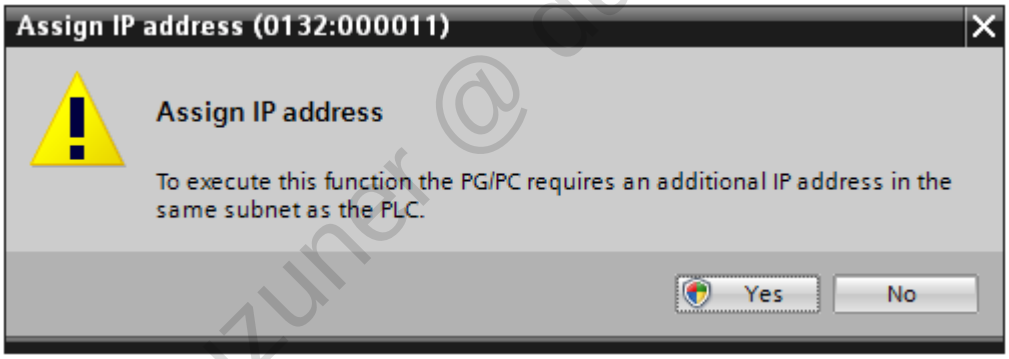
This function provides the option of fast access (for example for service purposes) even when **there is no offline project data for the target systems on the PG**.

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

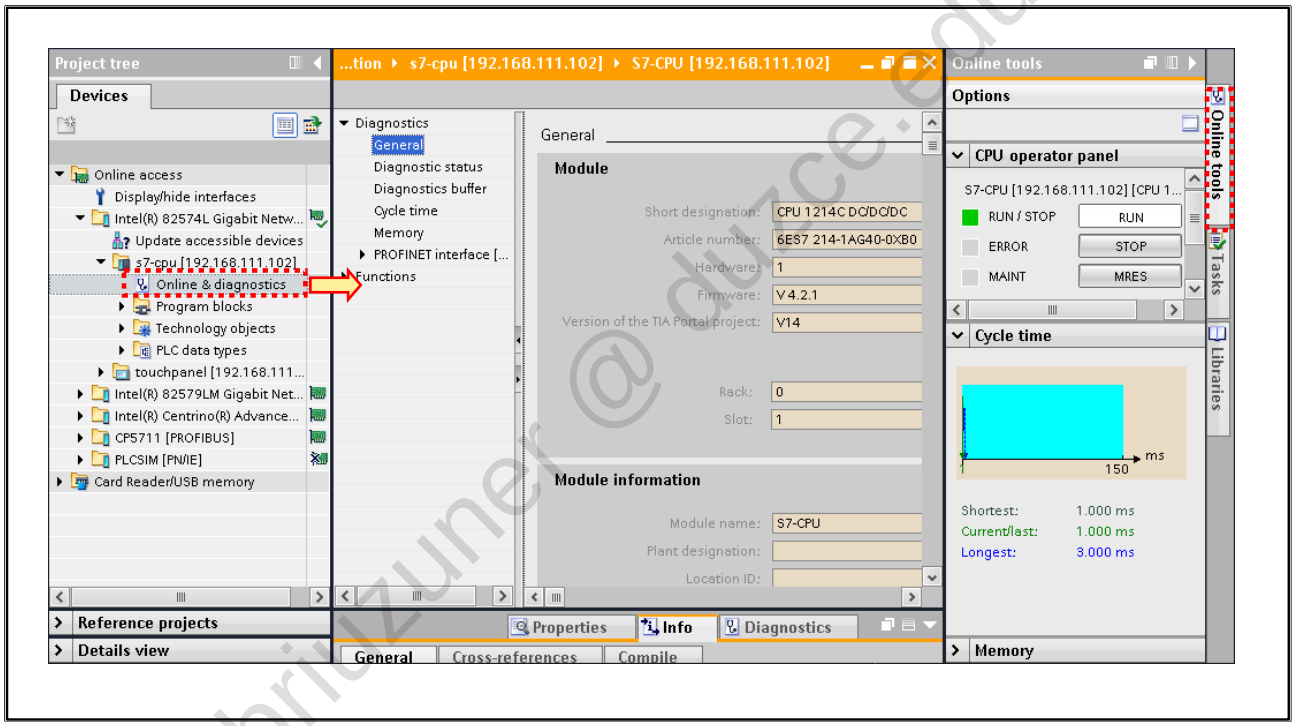
Access Online Functions → Show Button

Whenever there is an attempt to access a module online with the "Show" button which is 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 in the same subnet as the address of the CPU. After that, all online functions can be used.



4.4.1. 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.

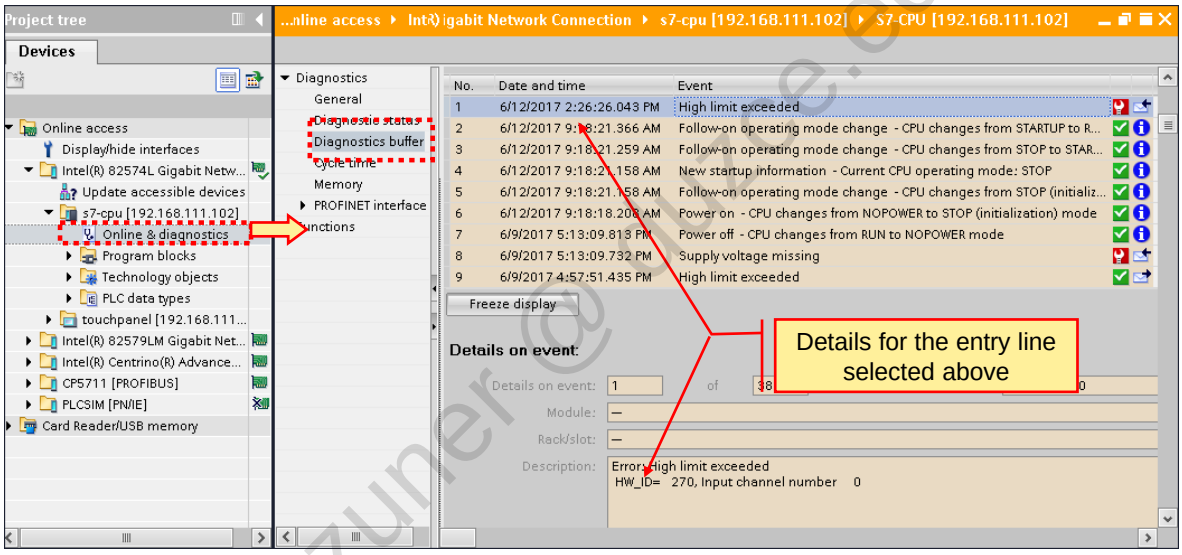
Cycle Time:

"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

4.4.2. Accessible Devices: Online & Diagnostics: Diagnostics Buffer



Online Access to the CPU

If the PG and the target system (for example CPU) are 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, some of the diagnostics buffer is not buffered with Power OFF (only a part is retentive).

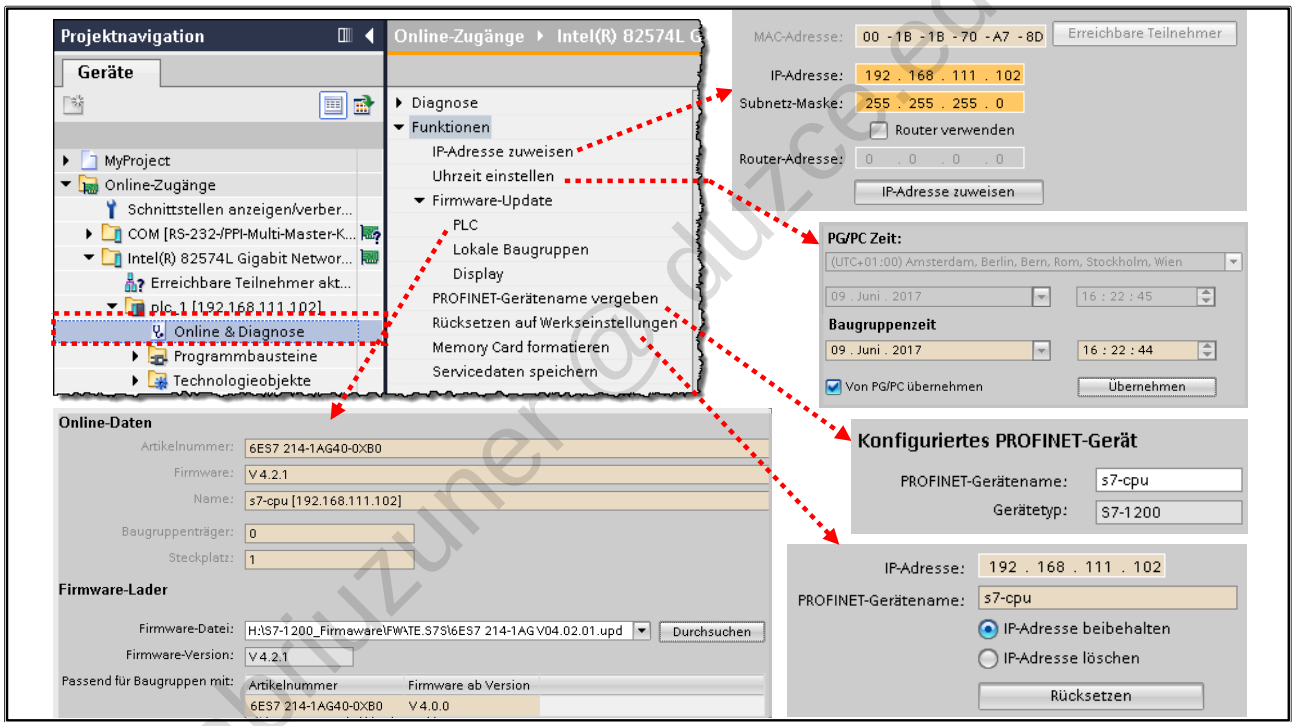
- Number of entries, 1000 to 3200
- Of that, retentive 250 to 500

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.

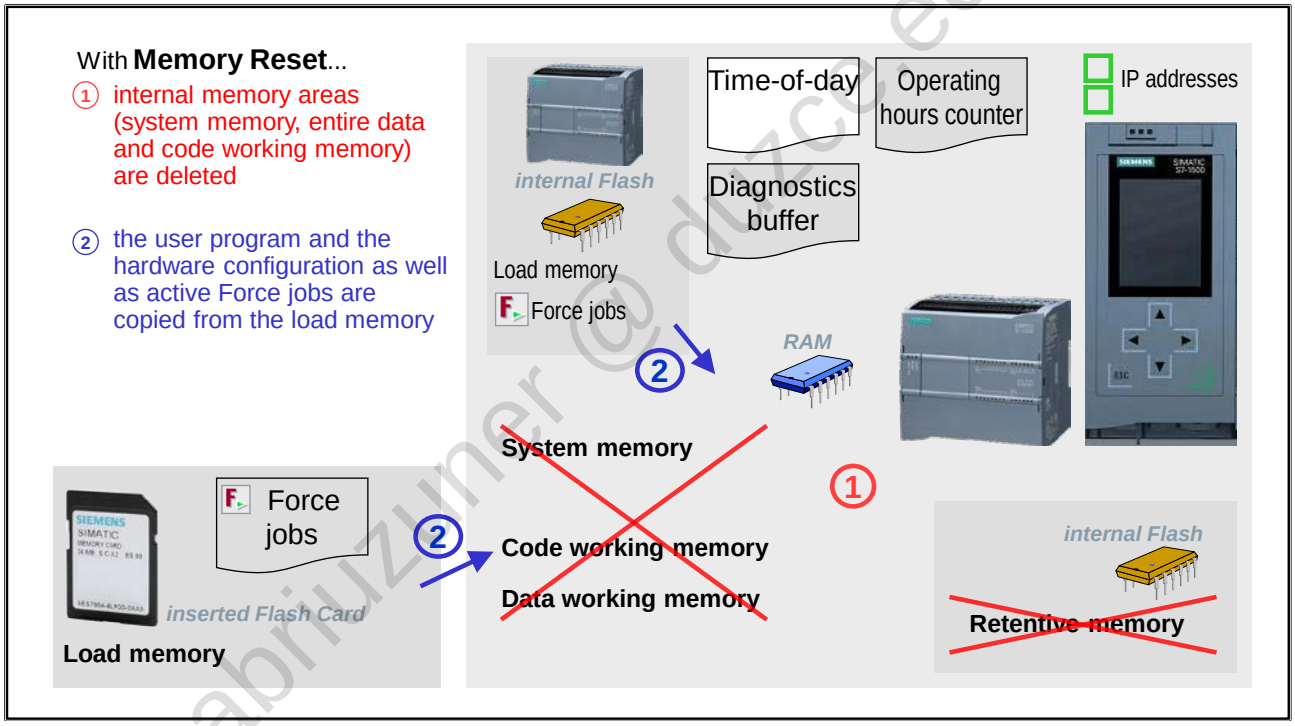
4.4.3. Accessible Devices: Online & Diagnostics: IP Address, Name, Time, FW Update, Memory Card



- **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**
  Here, 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 must be updated, first the Display and then the CPU.

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



CPU Memory Reset

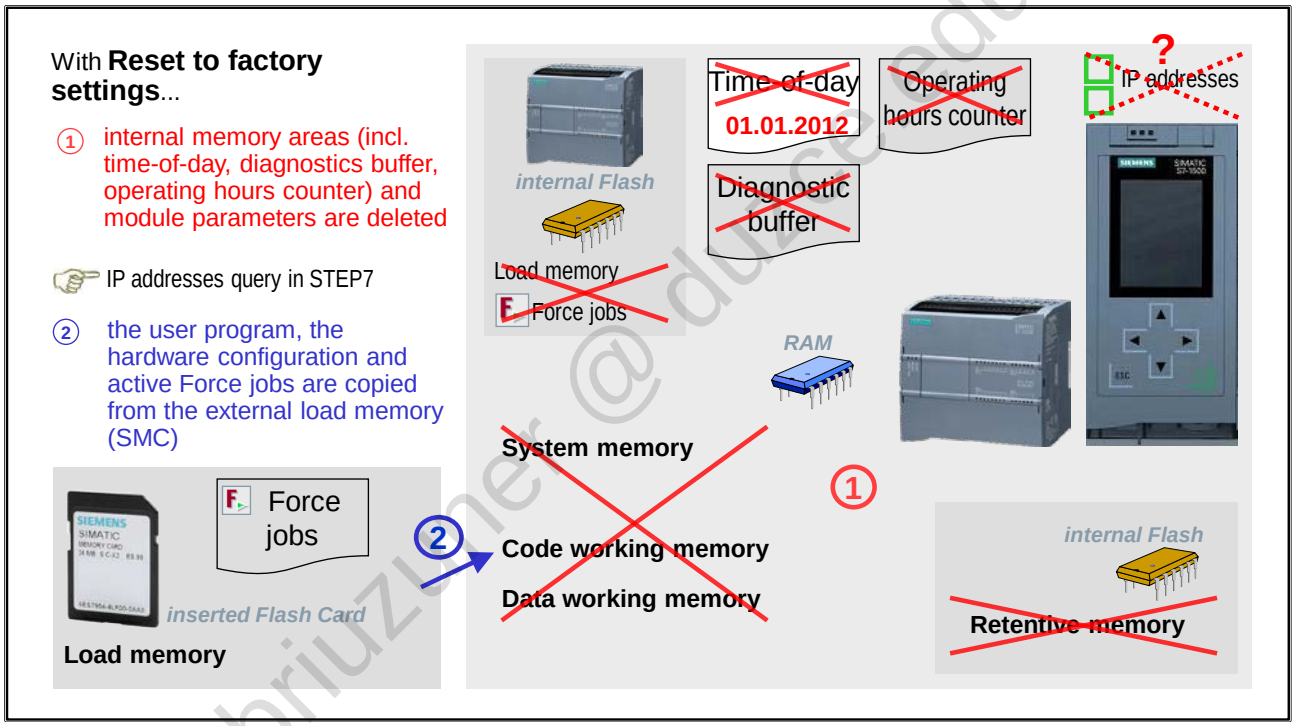
What to do:

- 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, including 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-da.
- 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)

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



CPU Reset to Factory Settings

What to do

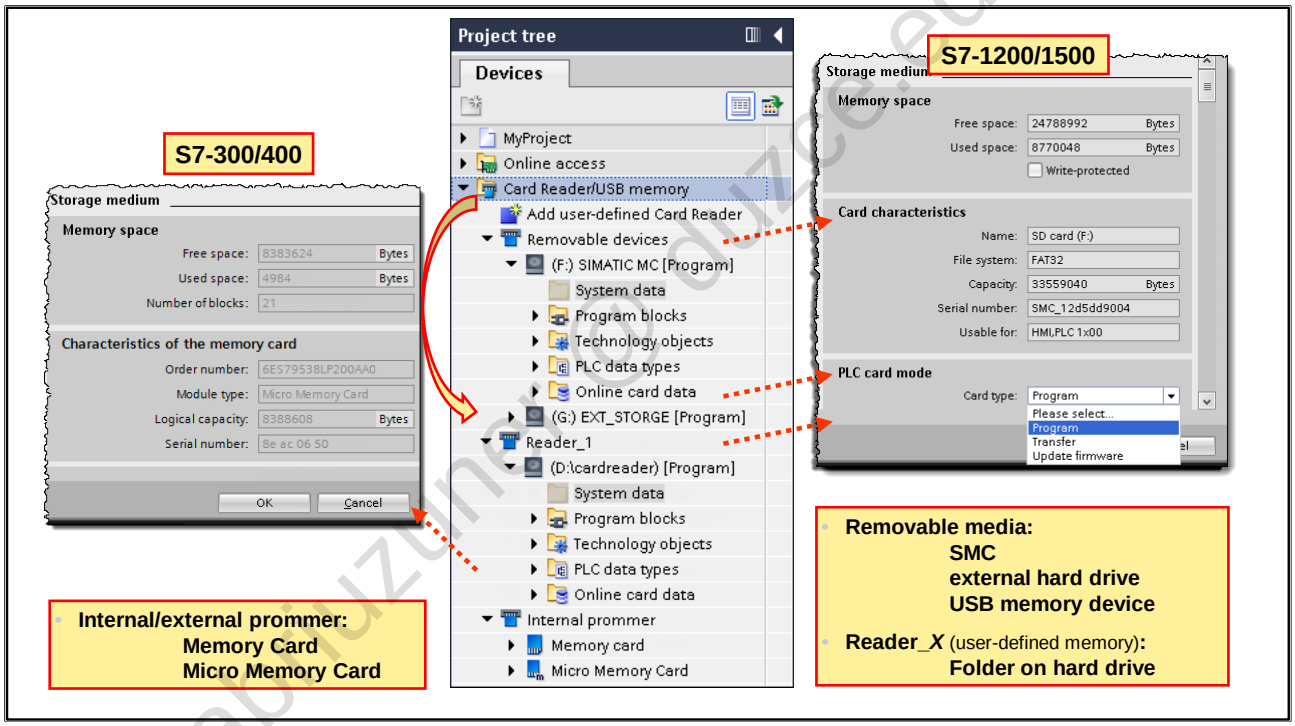
- STEP 7 online function → MRES in "CPU operator panel" of "Test" and "Online tools" Task Cards
- Display (only S7-1500) → Main menu "Settings", submenu "Memory reset"→ Factory Defaults
- Mode selector switch (only without memory card)

Impact

- An existing online connection between PG/PC and the CPU is disconnected
- The entire RAM work memory is deleted, including 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).

4.5.2. 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:

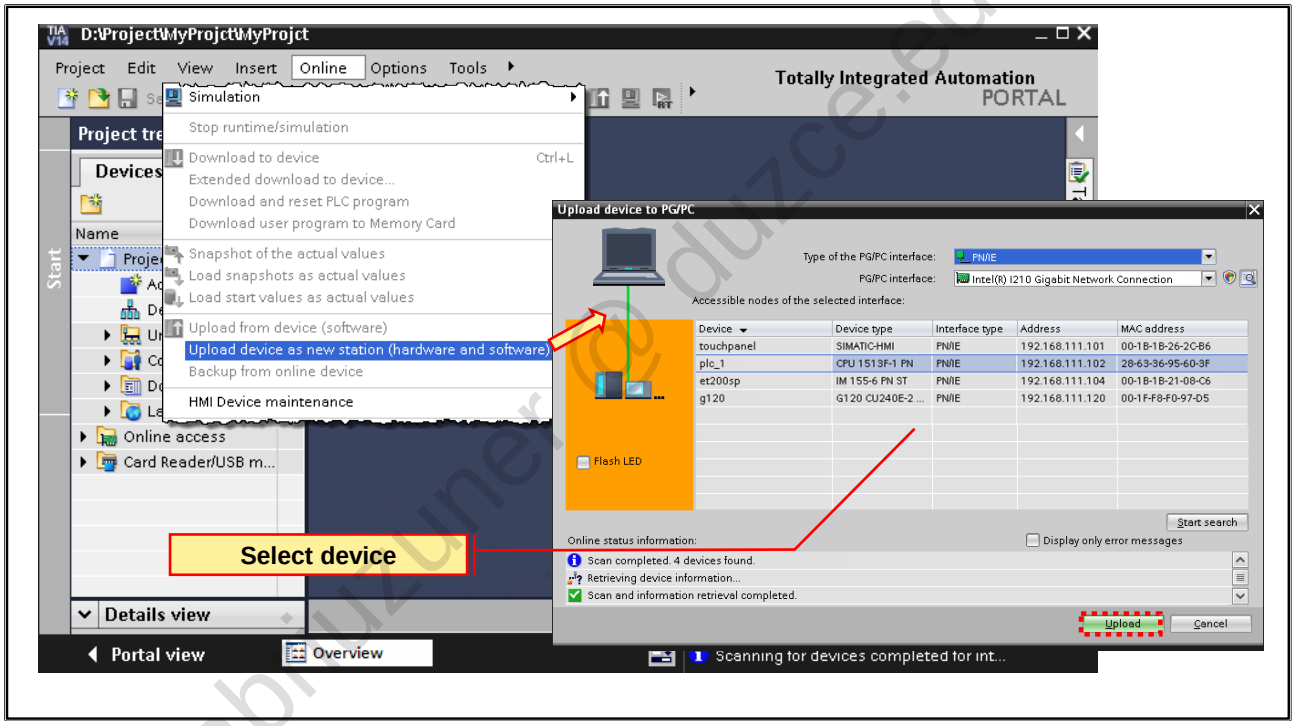
The SIMATIC Memory Card is used as a Program card or a Transfer card or for Firmware Updates. Before the relevant data is stored on the SMC, the card type must be selected as shown in the picture.

- **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 must be removed and a restart must 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.

4.6. Uploading a Hardware Station into the Project (1) (Hardware with Parameterization and Software)



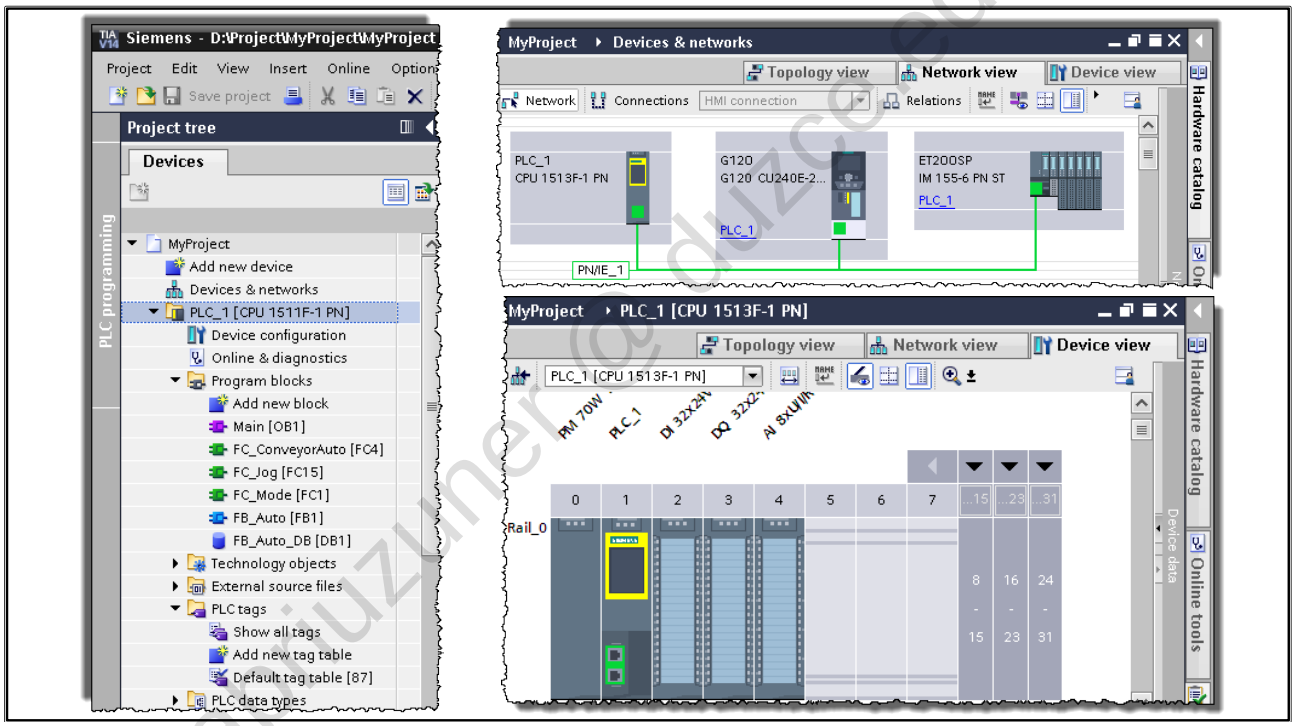
Unlike the option “Upload actual configuration without parameterization”, an already existing configuration of an S7 station can be read out using the function “Upload device as new station”.

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:

The station already has a configuration.

4.6.1. Uploading a Hardware Station into the Project (2) (Hardware with Parameterization and Software)



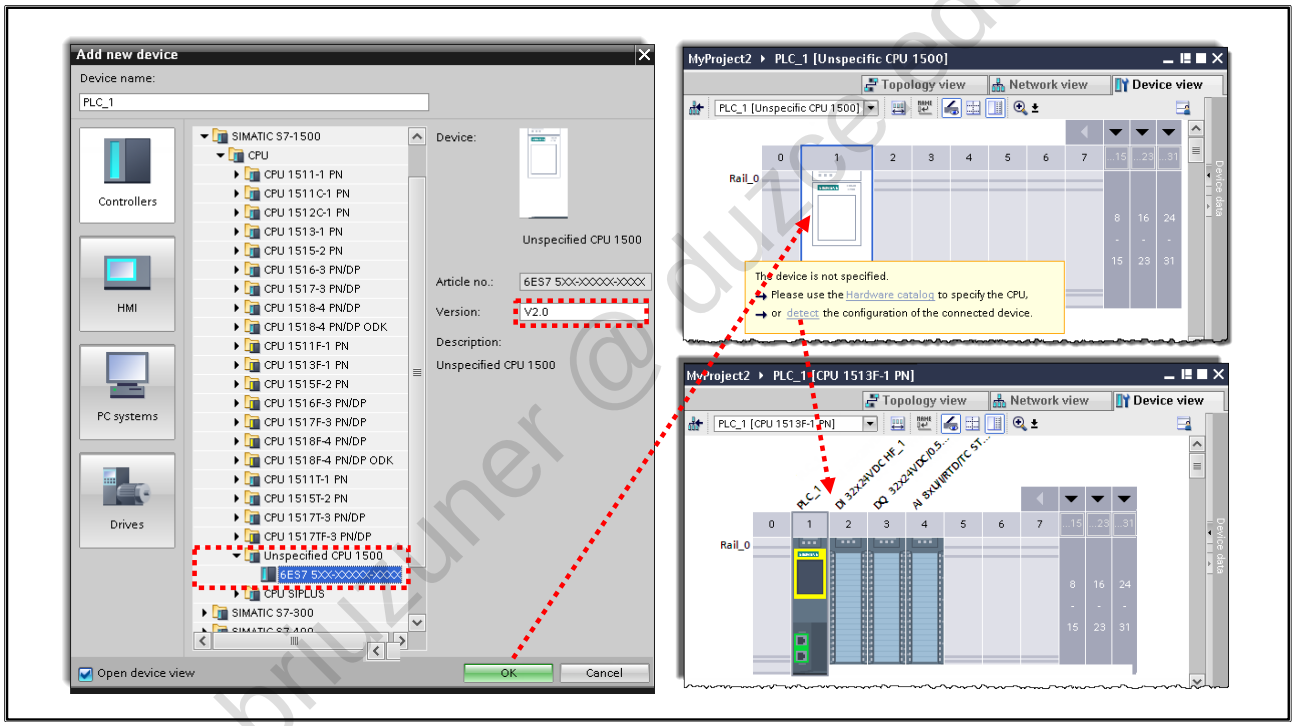
Station with project

After the device is uploaded, the entire station (central and distributed hardware with parameterization, the entire program with comments and symbols) is available to the user offline for further processing.

Station without project

If just an IP address has been assigned without transferring a project, only the central hardware without configuration is available.

4.6.2. Adding an unspecific CPU

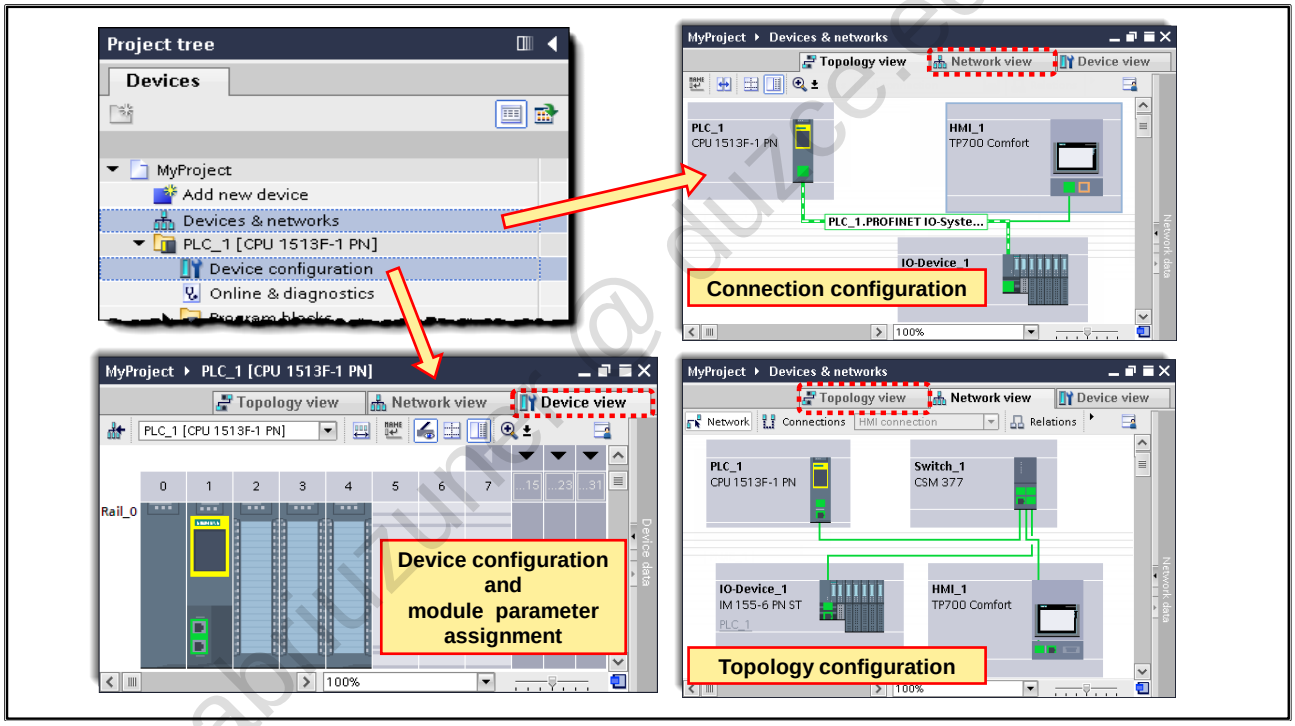


Unspecified CPU

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. 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.

4.7. 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.

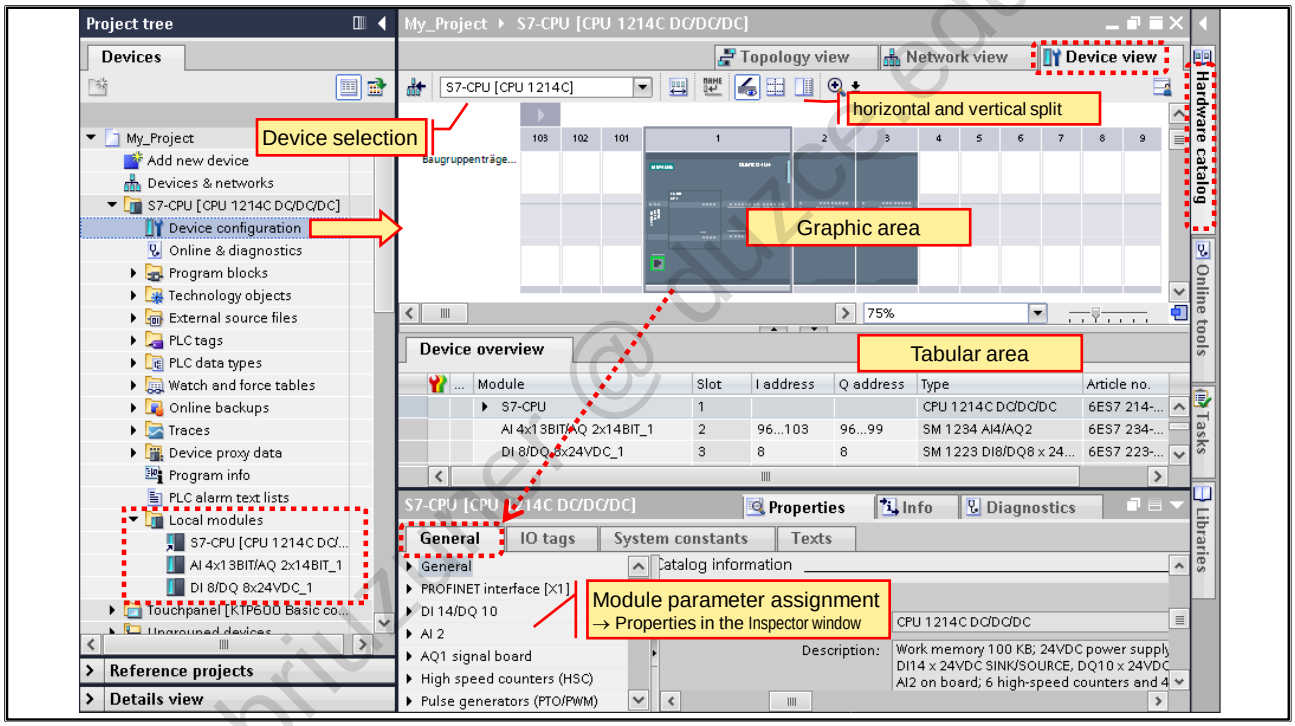
- Configure and parameterize devices
- Connection configuration

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



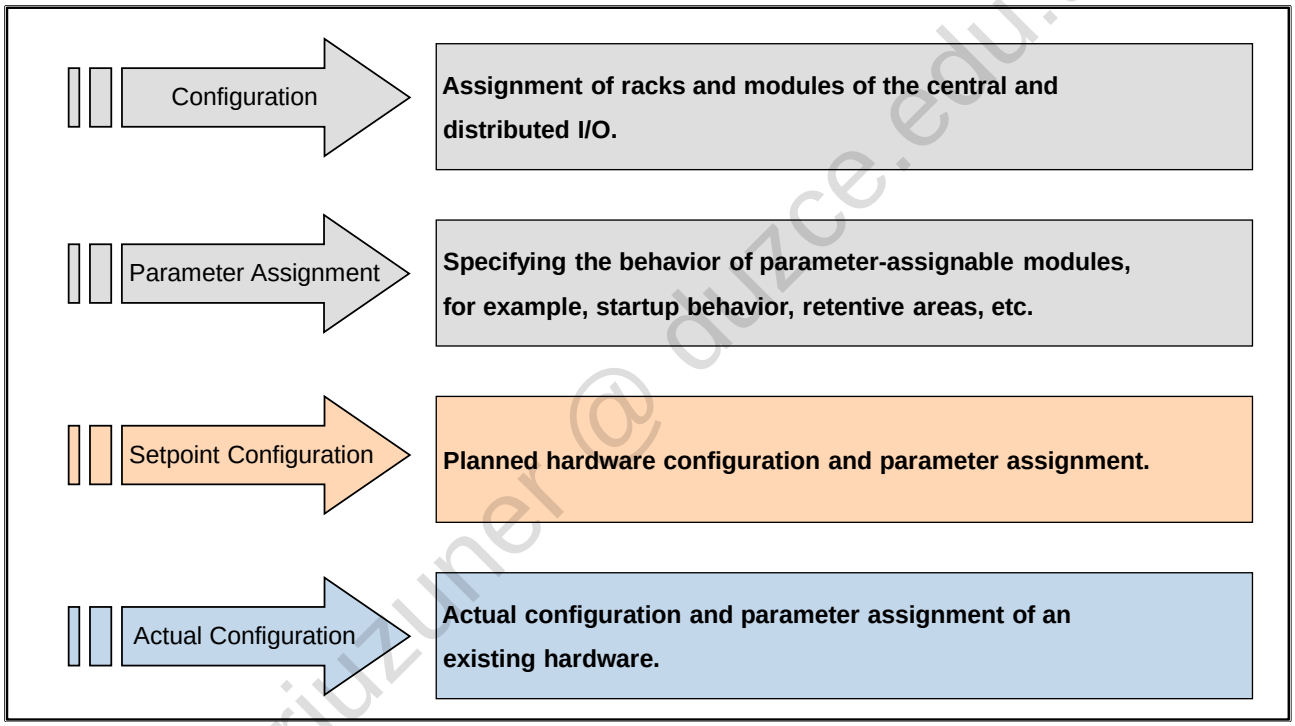
Components of the "Hardware and Network Editor"

- "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 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.

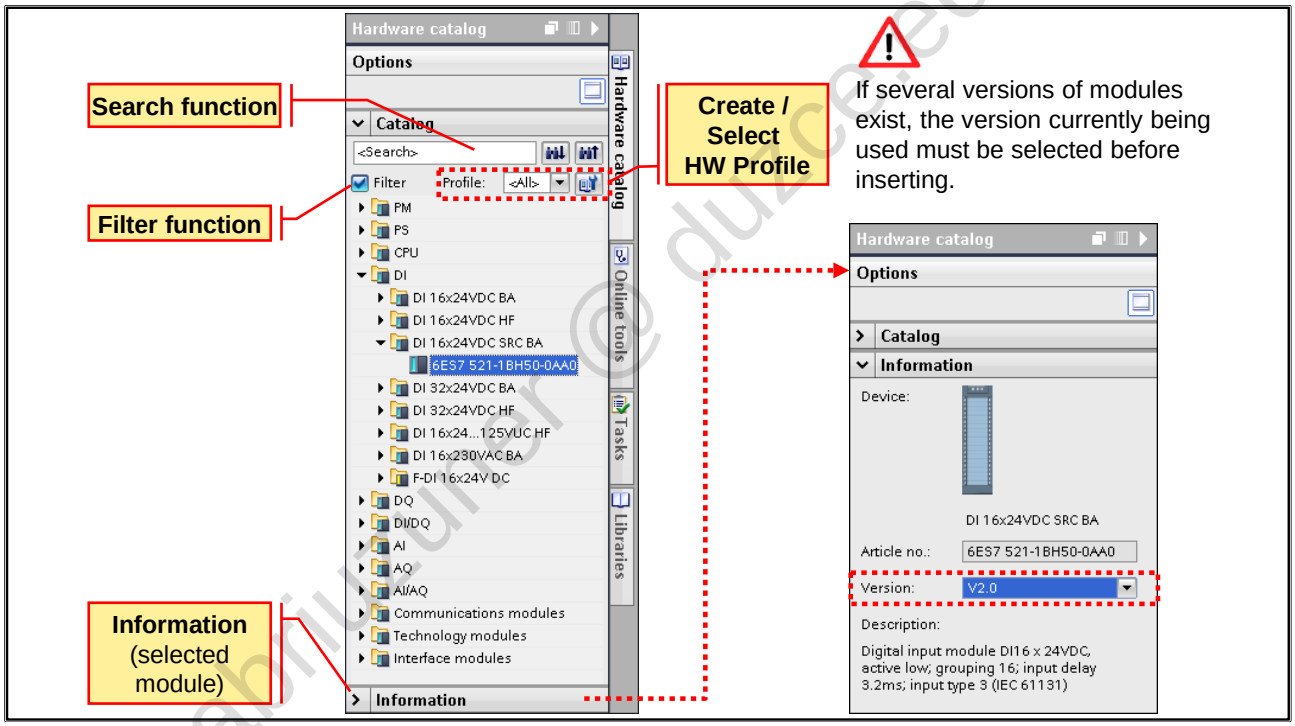
4.7.2. Setpoint and Actual Configuration



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.

4.7.3. 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

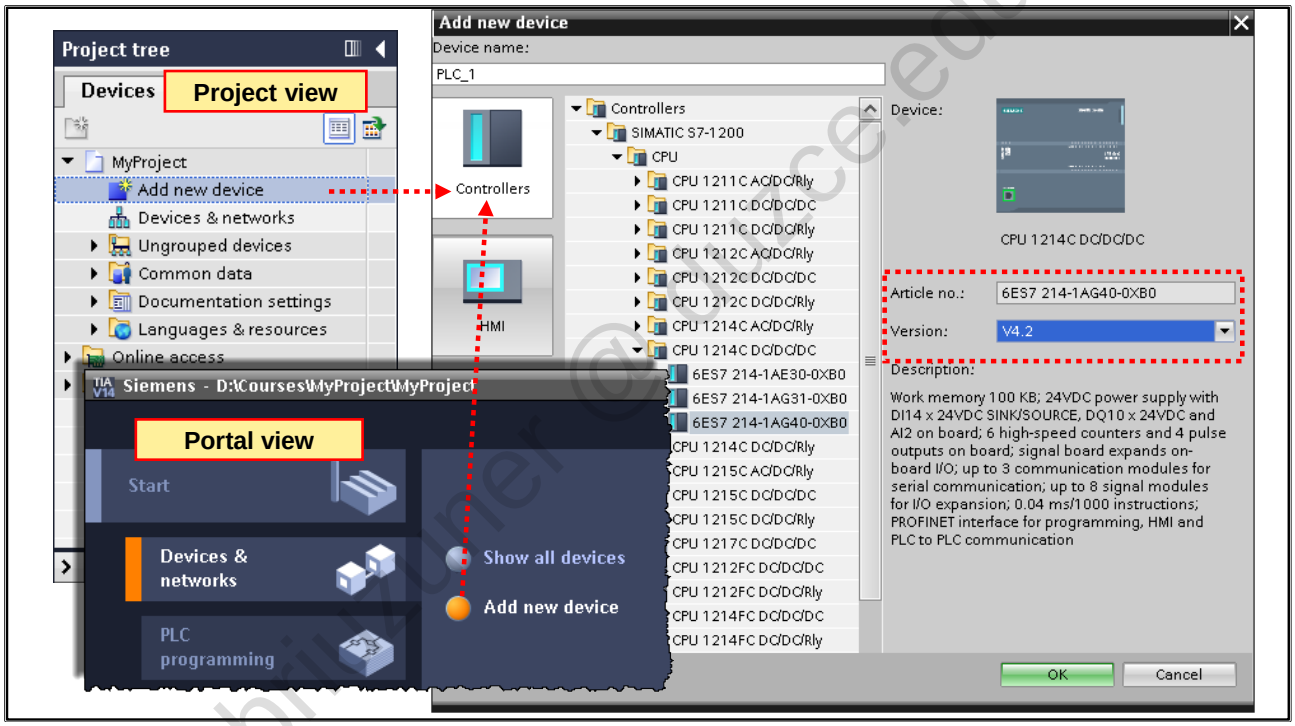
It is possible to create and to use your own profiles. This expands the filter possibilities.

Information

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

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

4.7.4. Setpoint Configuration: Creating a Hardware Station (Controller)



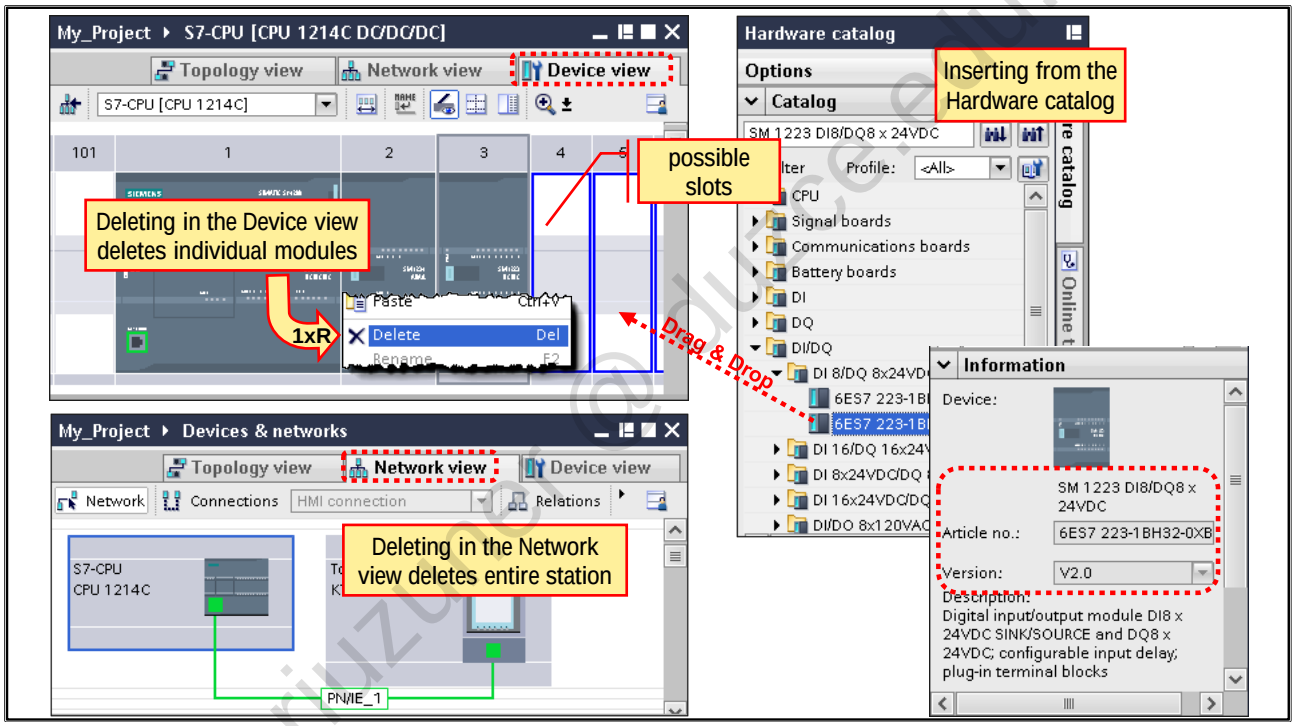
Add New Device

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.

4.7.5. Inserting / Deleting a Module



Inserting a Module

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

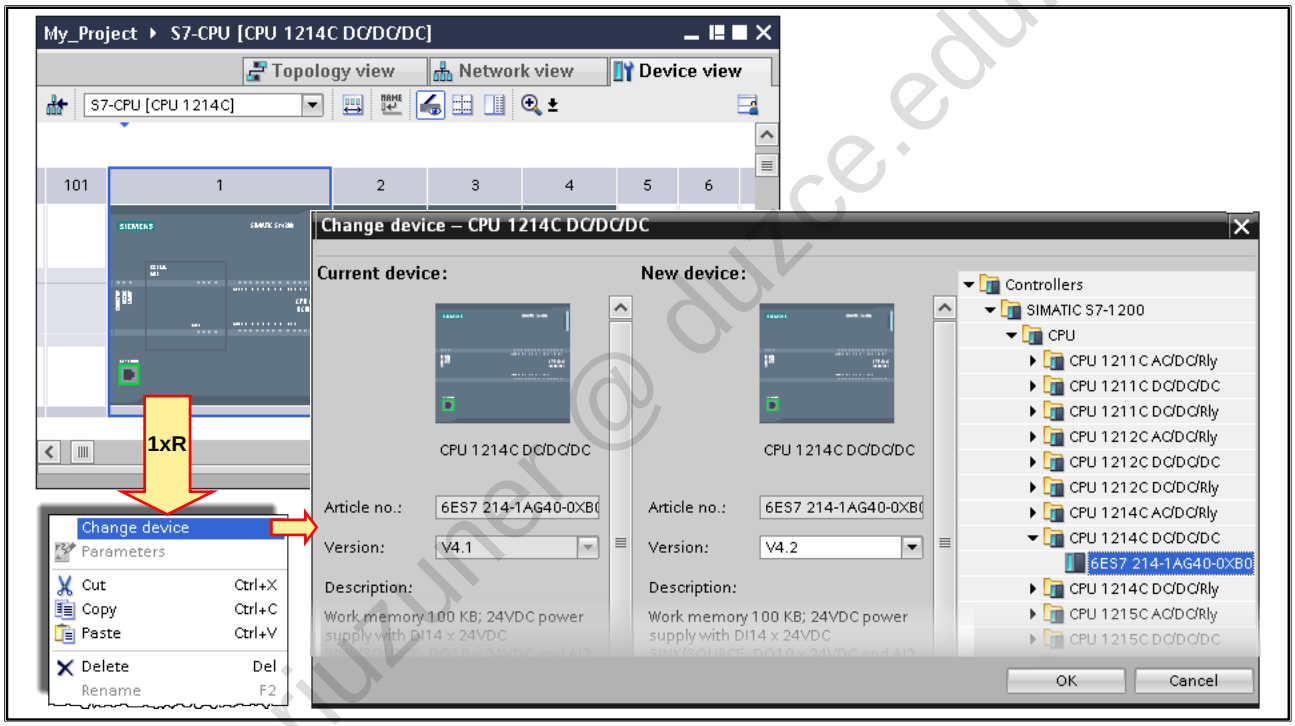
Selecting a Version

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.

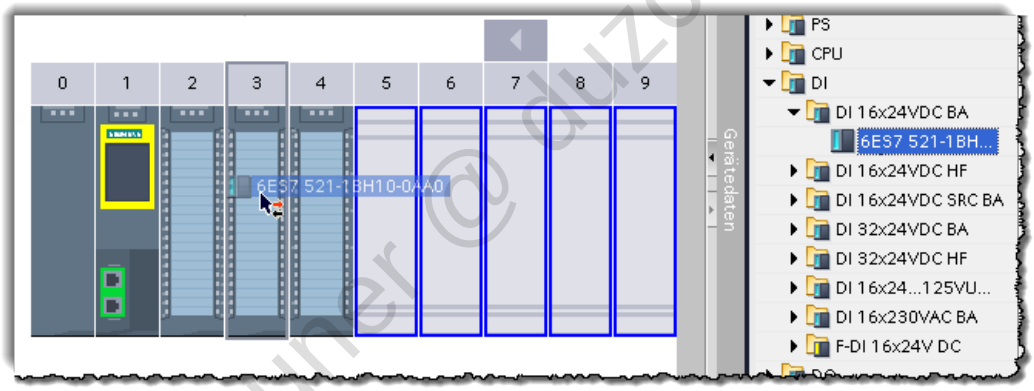
4.7.6. Changing a Device / Module



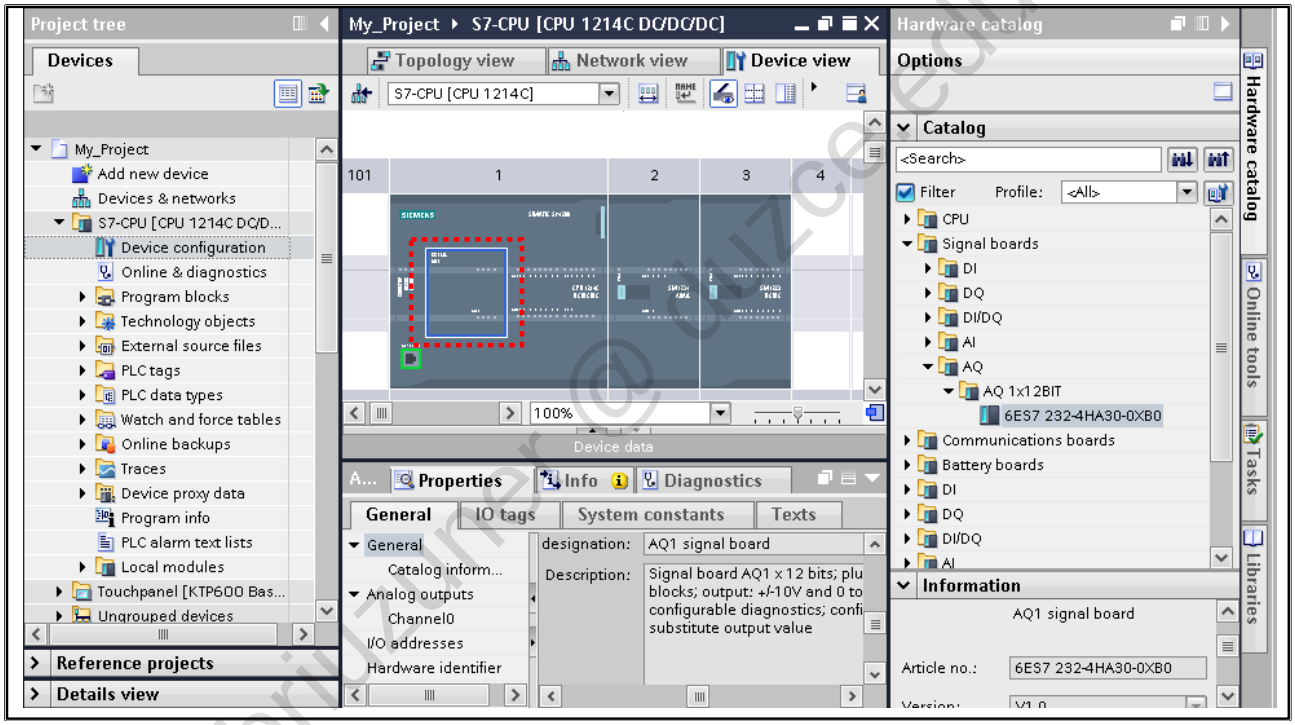
Changing a Module

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.



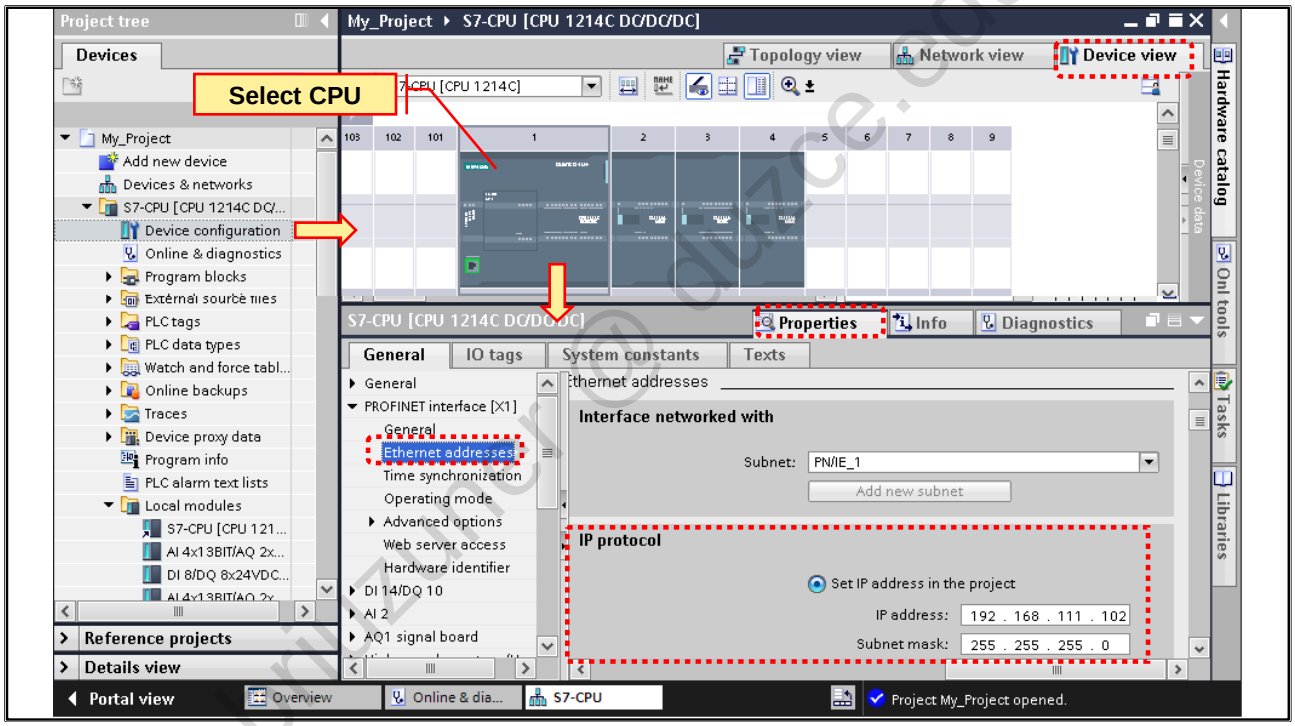
4.8. CPU Signal Board



CPU Signal Boards

Signal boards allow you to increase the number of CPU-specific inputs and outputs. You will find signal boards in the Hardware catalog along with all the other hardware components. Signal boards are not inserted into the rack like other modules. Instead, they are inserted directly into a CPU-specific slot. Each CPU can only accept one signal board.

4.8.1. CPU Properties: Ethernet Address

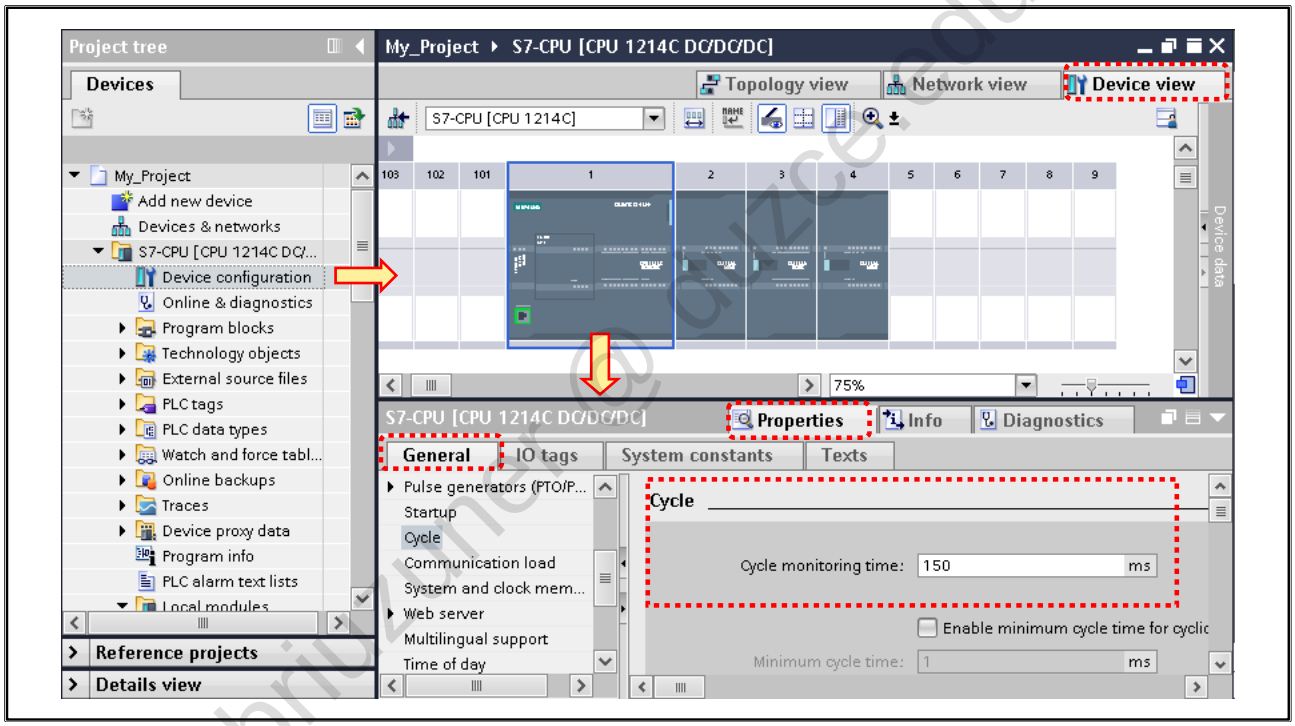


PROFINET Interface

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 in the same subnet.

4.8.2. CPU Properties: Maximum Cycle Time



Cycle Time

Cycle time is the time it takes for the CPU to complete one program cycle. Because 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:

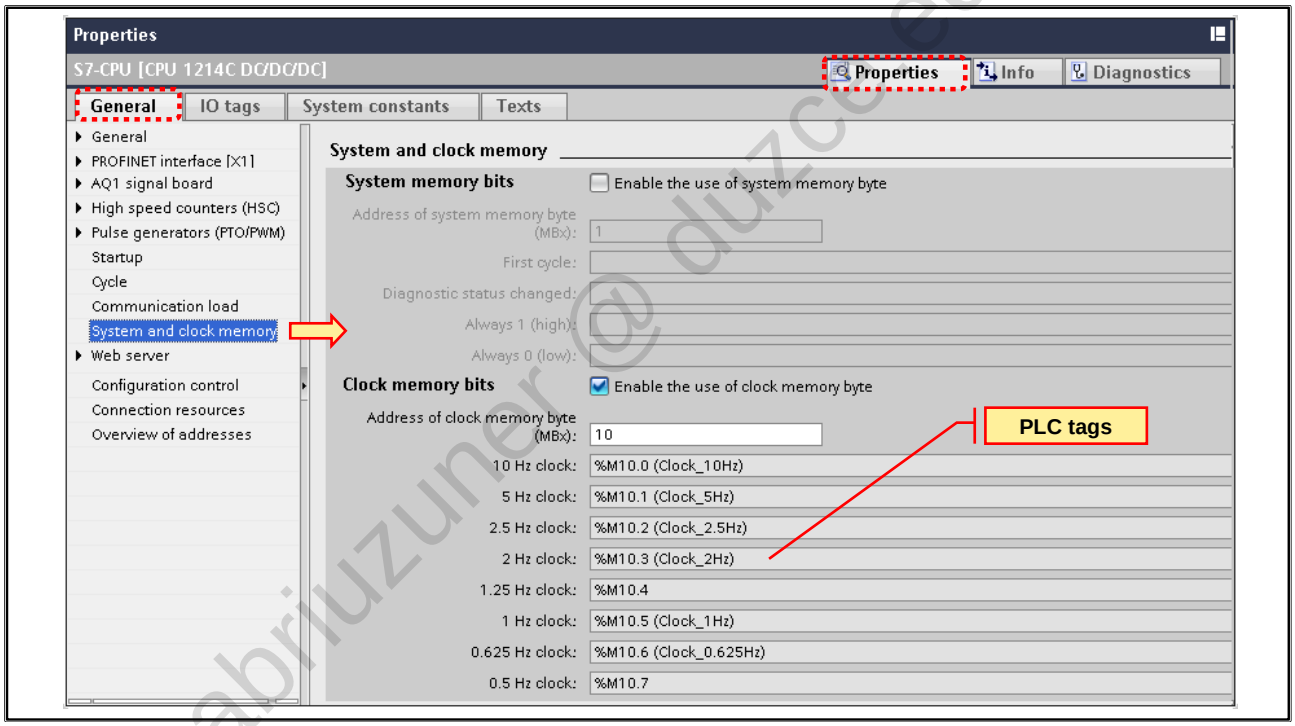
- **S7-1200 with firmware V1.x to V3.x:**
  The CPU remains in RUN mode even if no time error interrupt OB is programmed.
- **S7-1500 and S7-1200 with firmware from V4.x:**
  If no time error interrupt OB is programmed, the CPU changes to STOP mode.

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.

4.8.3. 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.

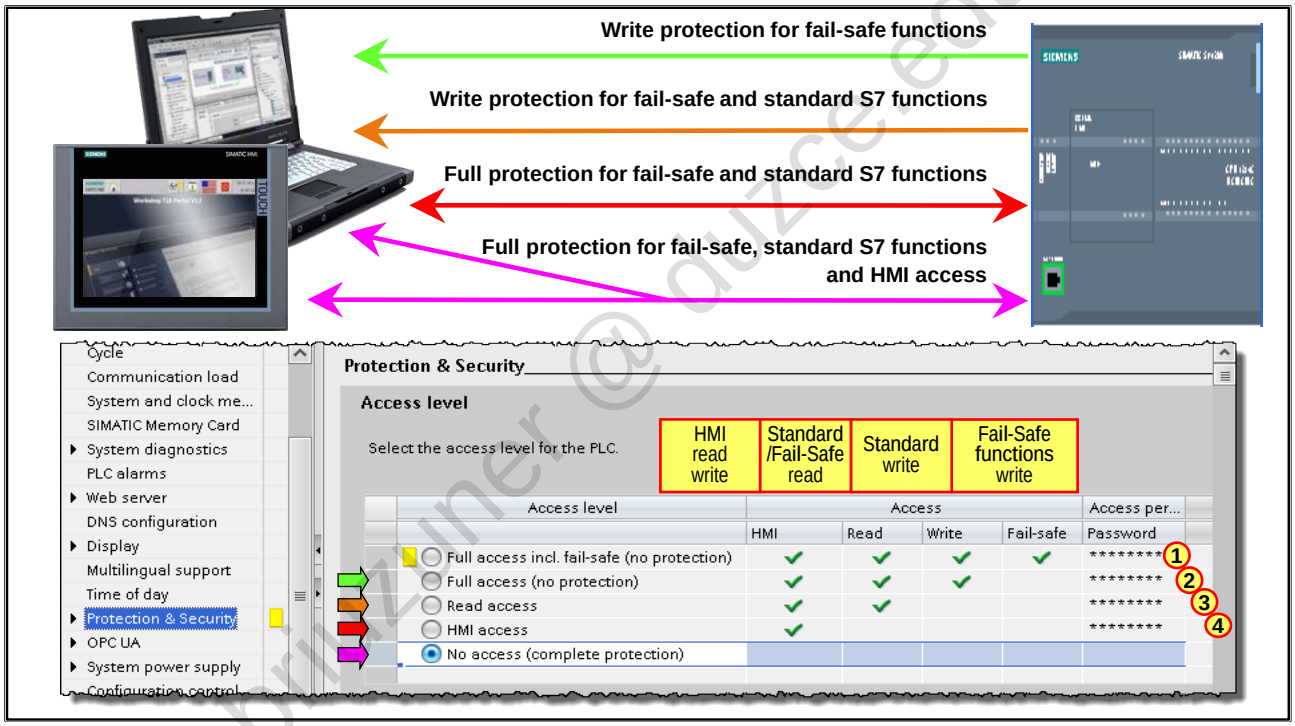
- "FirstScan" =1 in the first CPU cycle; otherwise =0
- "DiagStatusUpdate" =1, if the diagnostic status has changed
- One static 1-memory bit and 0-memory bit each (“AlwaysTRUE”, “AlwaysFALSE”)

Clock Memory (8 bits)

These are memory bits for which the binary value 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.

4.8.4. 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.

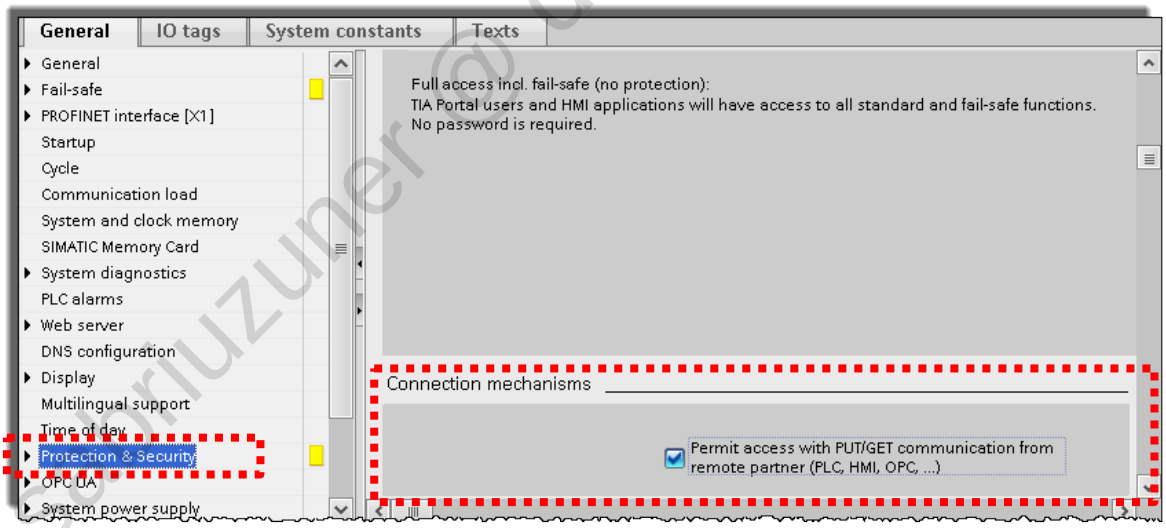
The above explained protection levels can, however, be lifted again with passwords:

- 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.

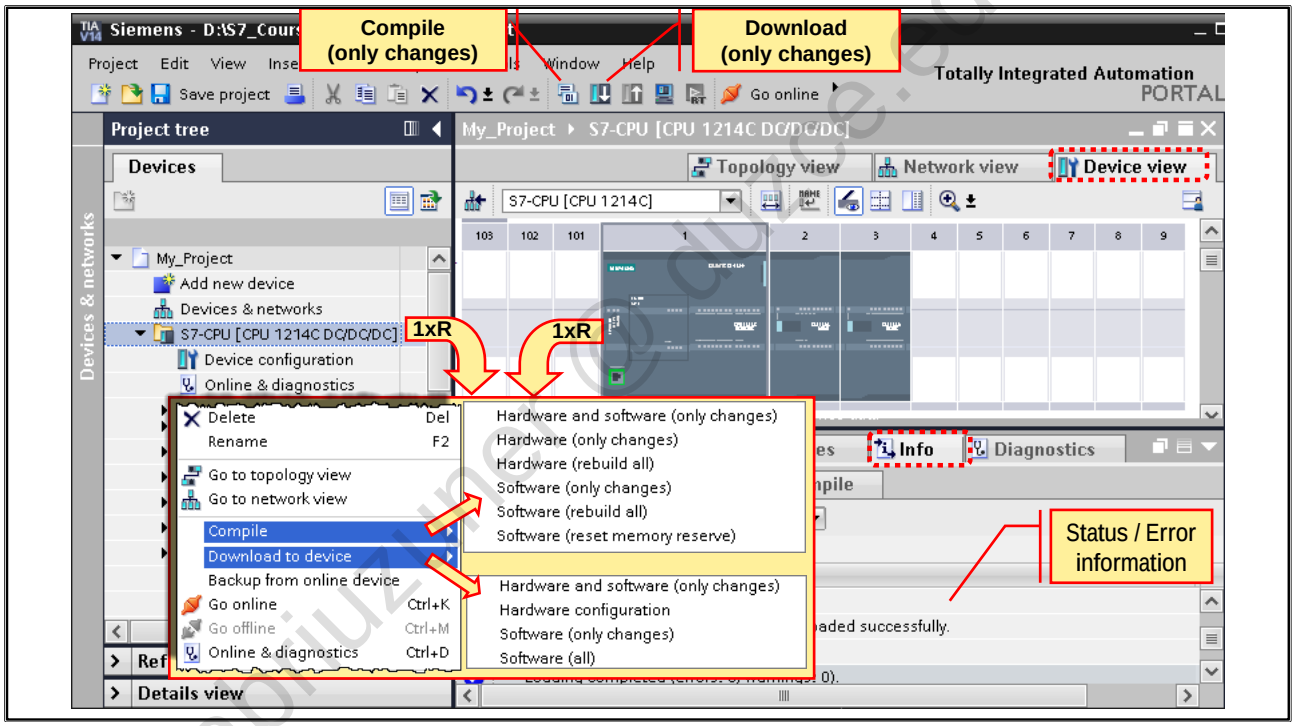
By specifying a password (1) read-accesses and write-accesses of the CPU are possible for both an HMI device as well as for STEP 7.

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.



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

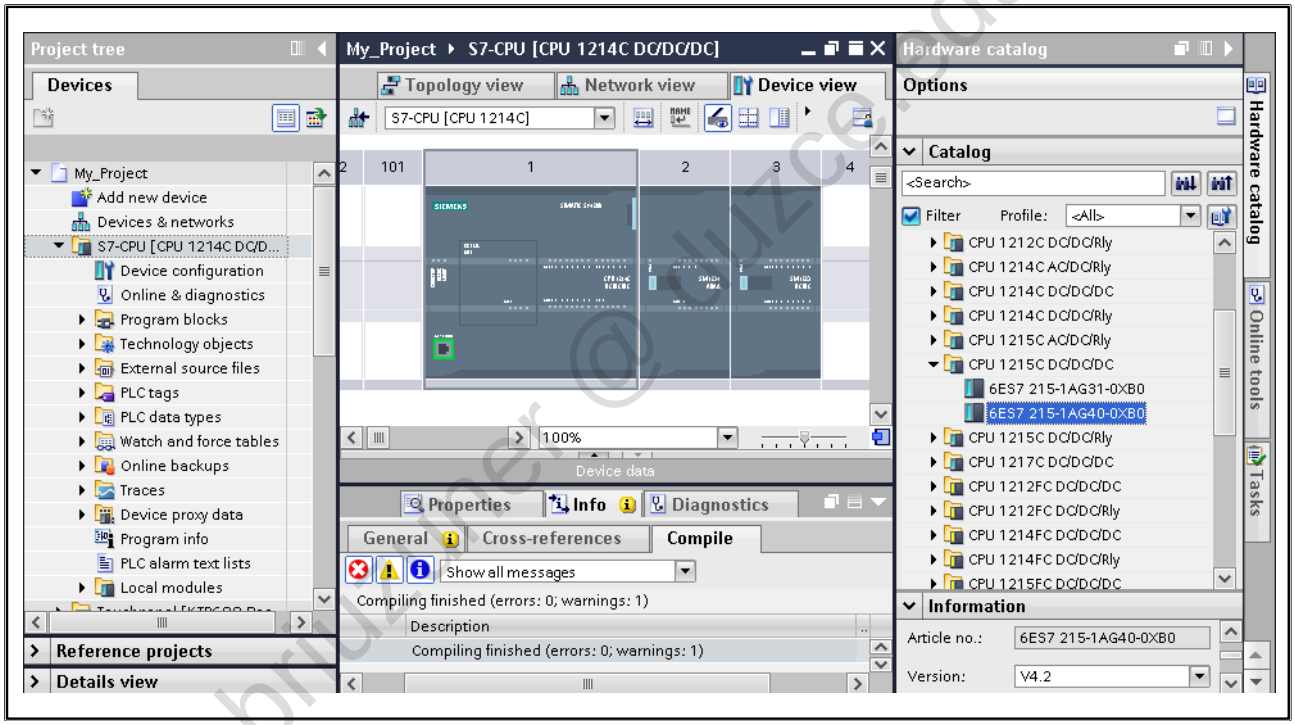


Compiling / Downloading the Hardware Configuration

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 S7-1200 Station

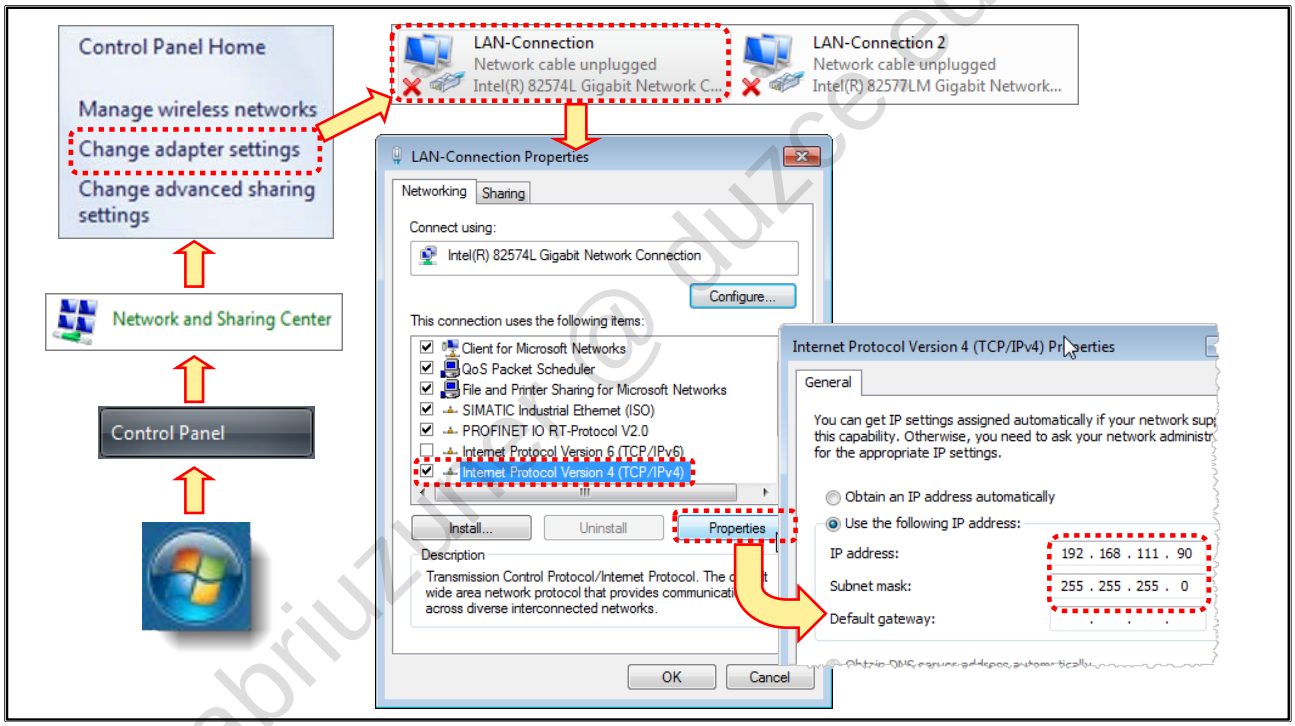


Task Description:

A new project with the name "My_Project" is to be created. It is to contain an S7-1200 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

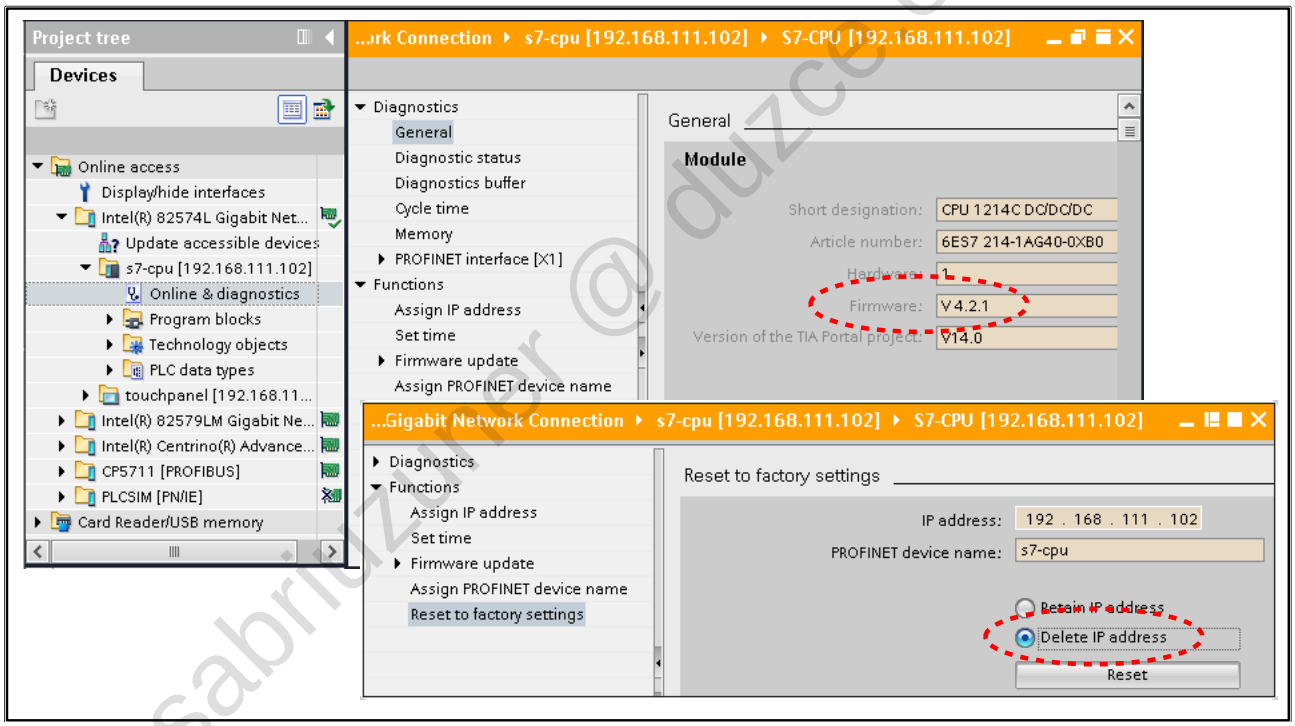
Task

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: Read out the Firmware version and reset the CPU to Factory Settings



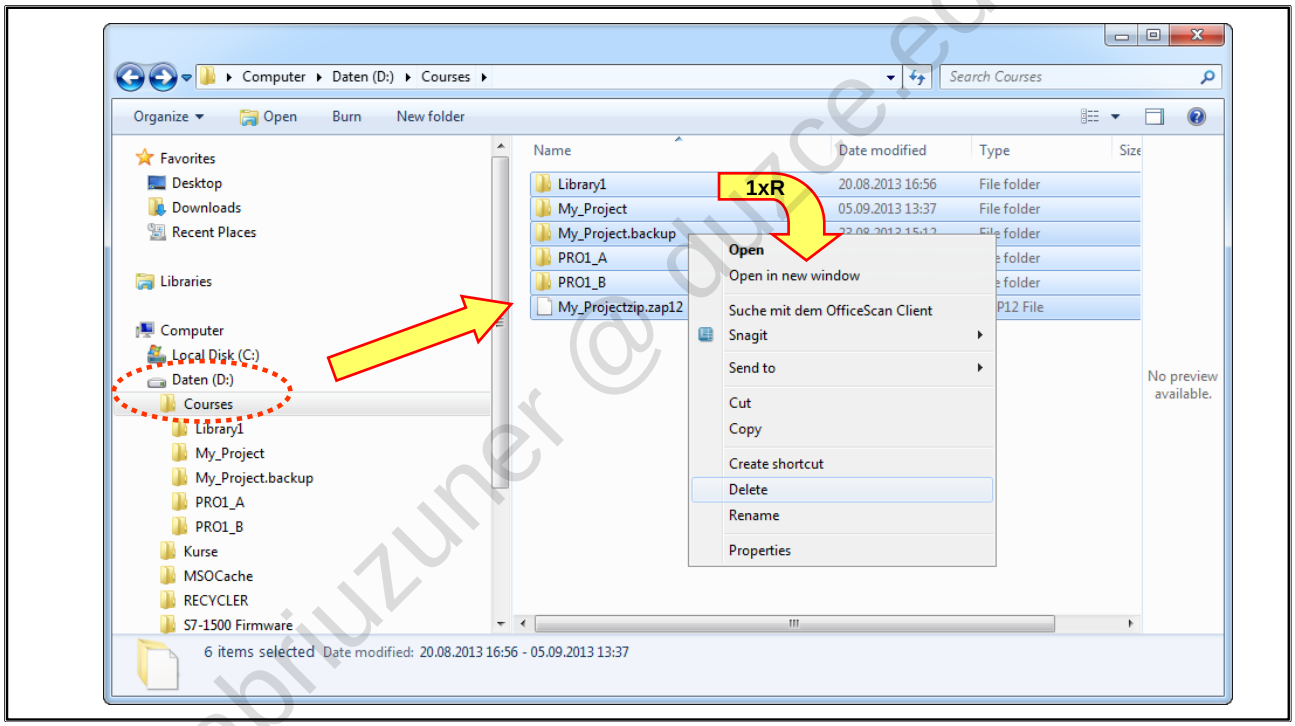
Task

Read out the Firmware version of the CPU and reset the CPU to factory settings and also reset the IP address of the CPU.

What to Do:

1. Open the Online access
2. Open the Interface which you are using (Intel(R) 82574L Giga Network Connection)
3. Activate "Update accessible devices"
4. In the appearing list open the CPU and activate "Online & diagnostics"
5. In "General" read out the Firmware version
6. in "Functions" perform a "Reset to factory settings" with the option "Delete IO address"

4.9.3. Exercise 3: Deleting Old Projects



Task

Delete the TIA Portal projects on the PG.

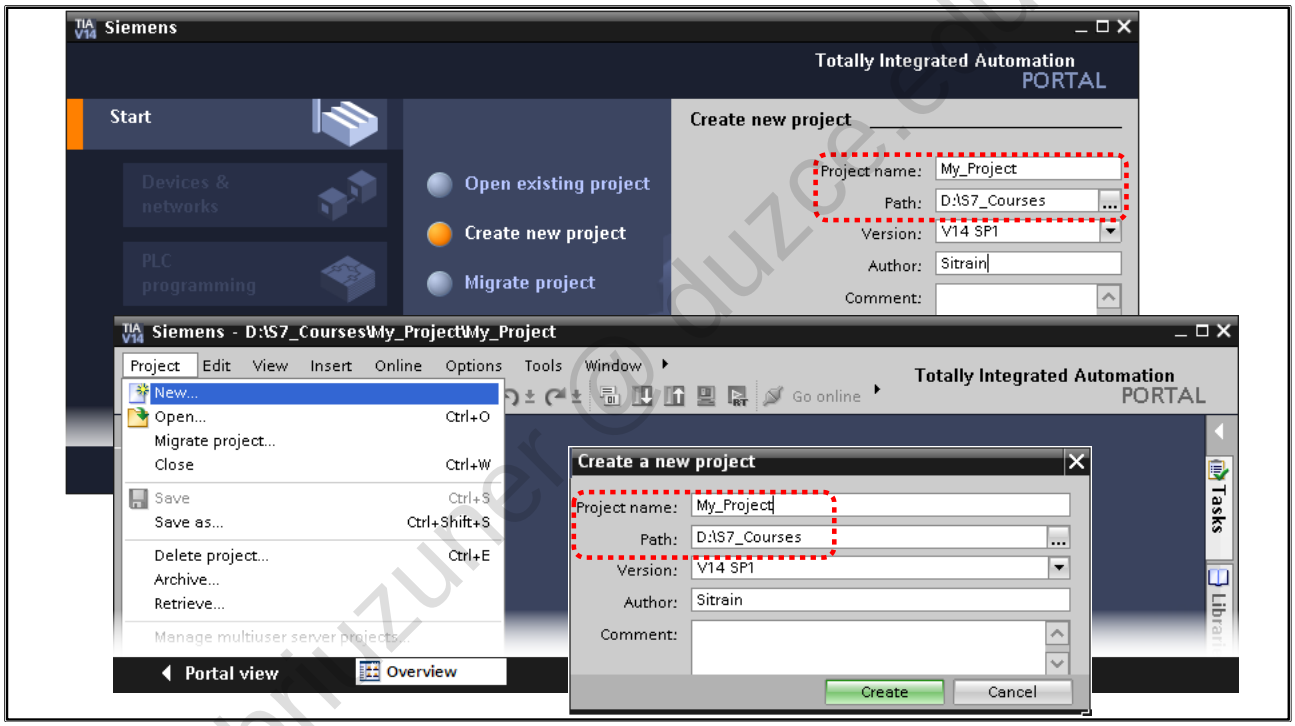
What to Do

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

Note

Projects that are to be deleted must be closed!

4.9.4. Exercise 4: Creating a New Project



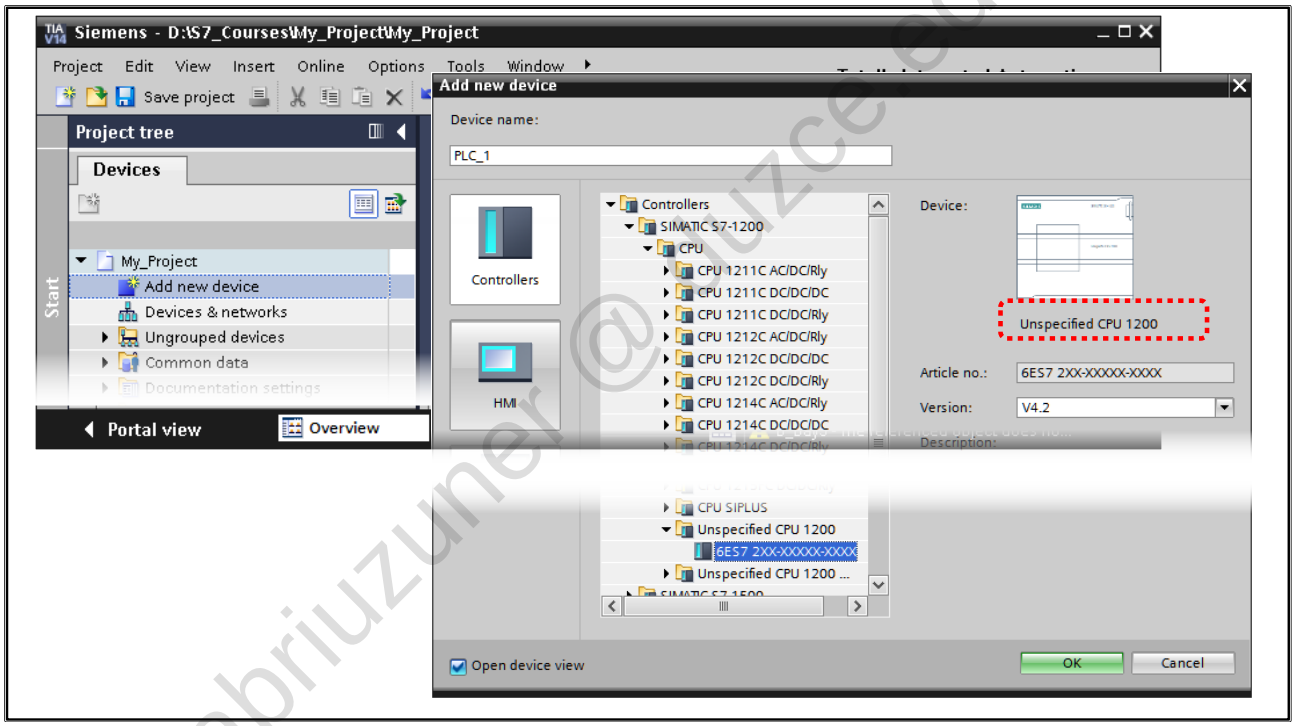
Task

A new project “My_Project” should be created.

What to Do

Create a new project in the shown path and give it the name “My_Project”.

4.9.5. Exercise 5: Creating a S7-1200 – Station (unspecific)



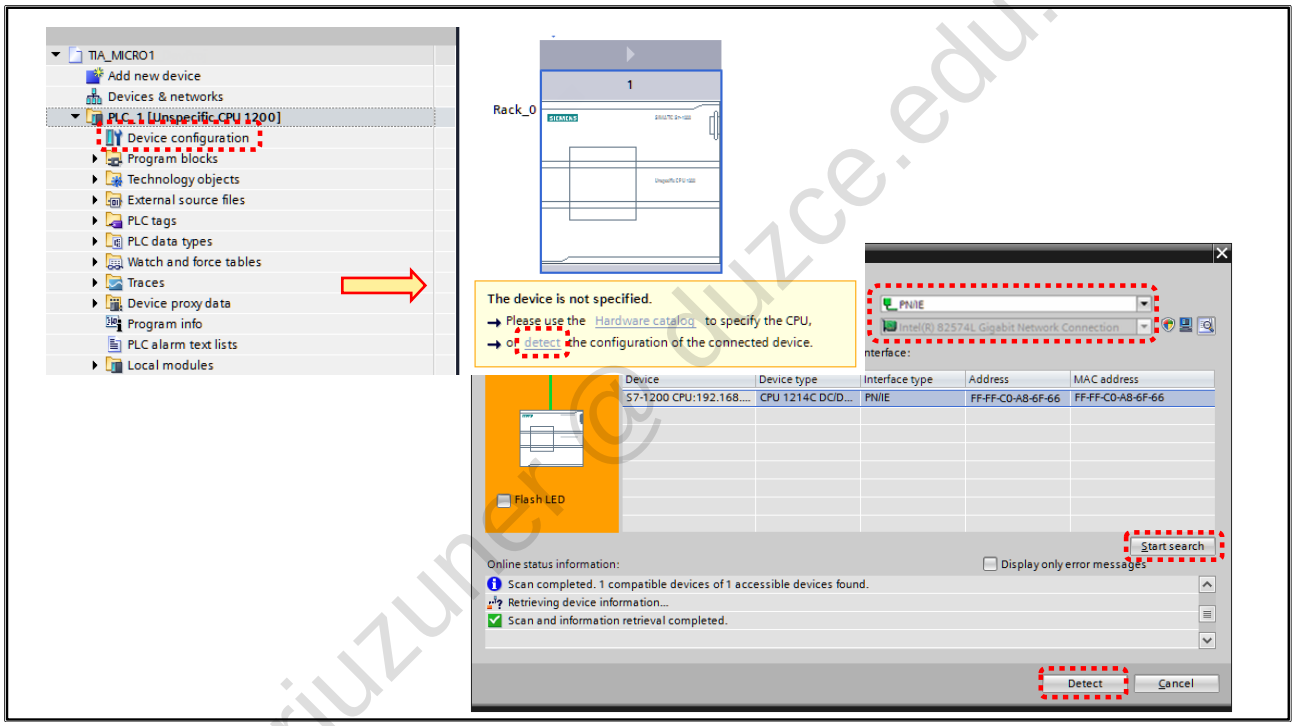
Task

Create a new S7-1200 in the project. The device should be detected automatically.

What to Do

Activate “add new device” in the project tree and select an unspecific CPU 1200

4.9.6. Exercise 6: S7-1200 – Station detection



Task

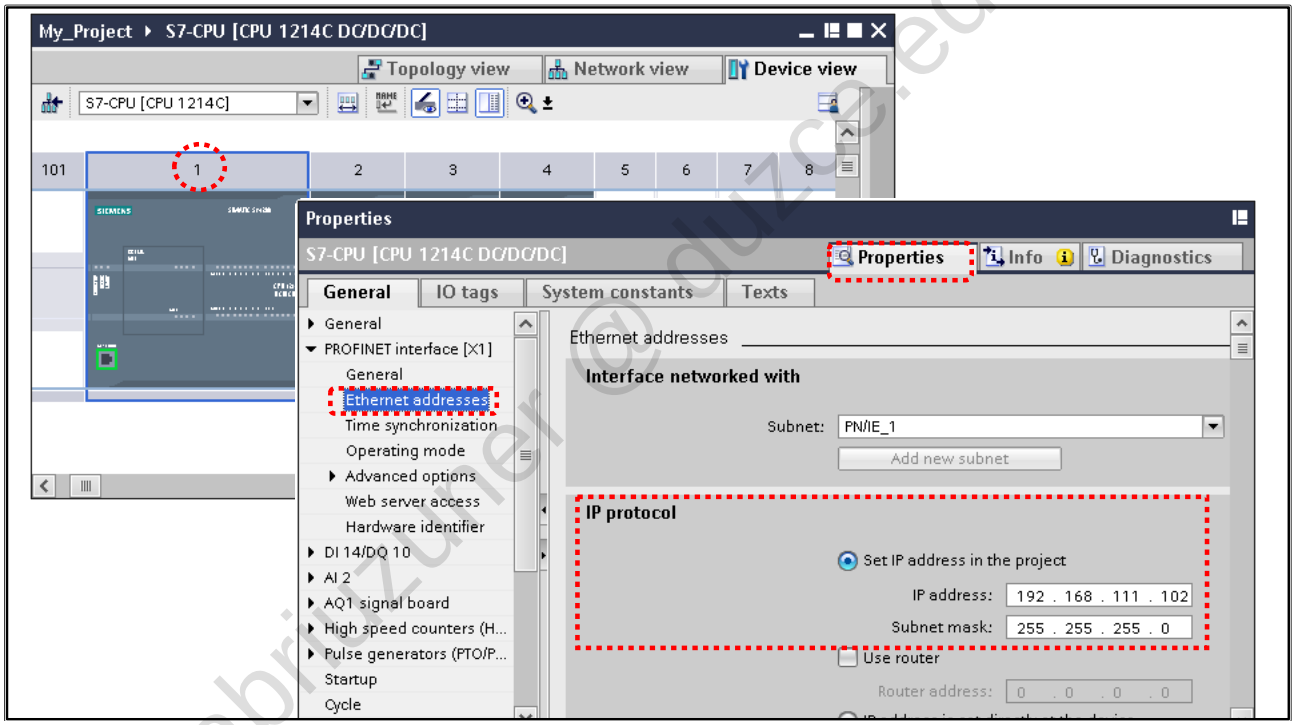
The real hardware should be recognized by the unspecific CPU and be inserted in the project.

What to Do

Open the device configuration of the S7-1200 and select “detect”. Select the interface in the opened window and start the search. When the PLC is found, detect the hardware.

The controller will be inserted in your project.

4.9.7. Exercise 7: CPU Properties: Assigning the IP Address



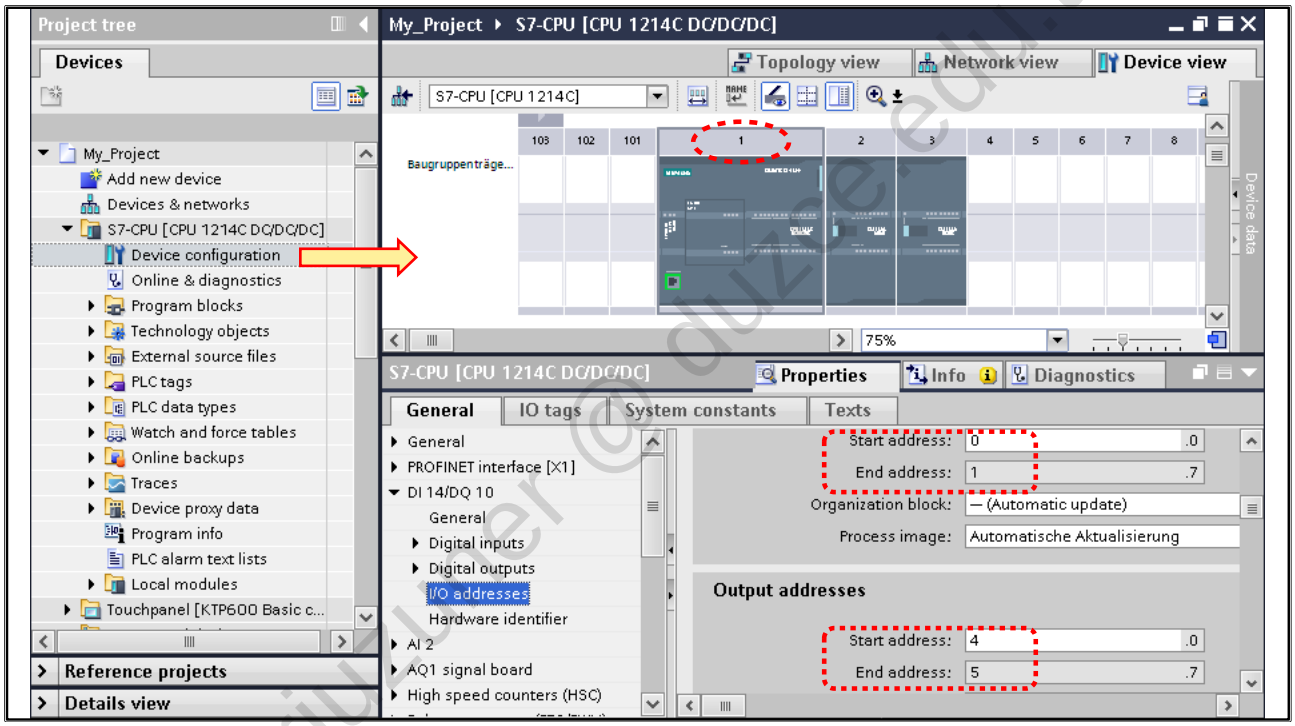
Task

Configure the S7-1200 CPU's IP address and subnet mask as shown in the pictures above.

What to Do

1. In the "Devices & networks" editor, select the "Device view" of the S7-1200 CPU
2. Select the CPU and open the "Properties" tab in the Inspector window
3. There, select the "PROFINET interface [x1]" tab and then under "IP protocol" enter the IP address and subnet mask shown (see lower picture)
4. Save your project

4.9.8. Exercise 8: CPU Properties: Addresses of the integrated Inputs / Outputs



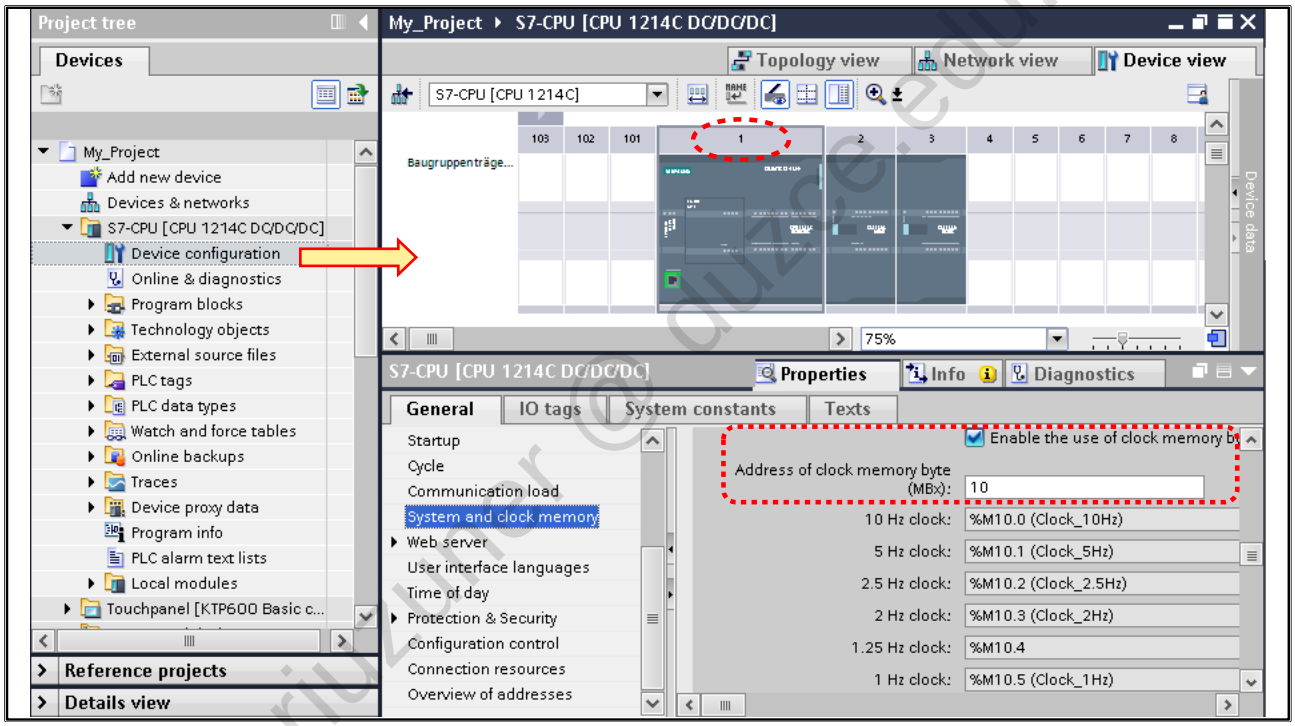
Task

You are to parameterize the I/O addresses of the integrated Inputs and Outputs of the CPU as shown in the picture.

What to Do

1. In the Device view, select CPU (see picture)
2. In the Inspector window, in the “DI14 / DQ10” tab, you can enter the I/O address as shown in the picture
3. Save your project

4.9.9. Exercise 9: CPU Properties: Parameterizing the Clock Memory Byte



Task

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

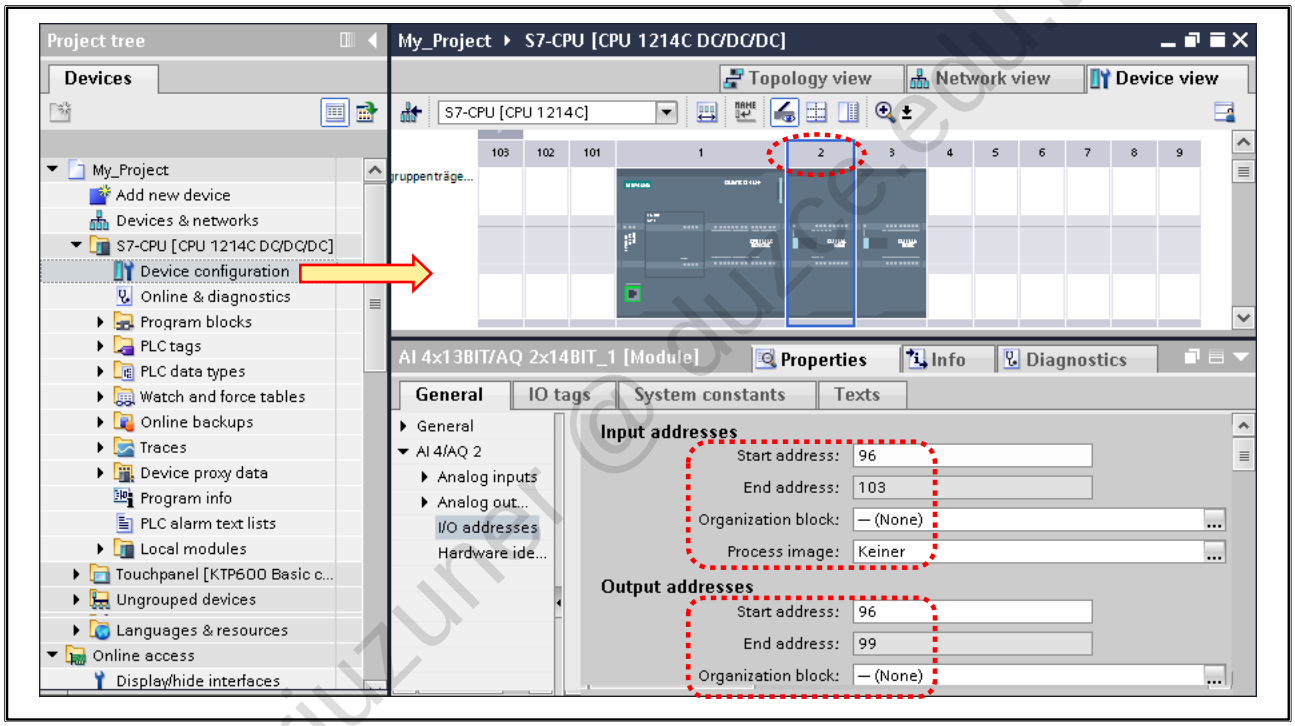
What to Do

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.

4.9.10. Exercise 10: Analog I/O module: I/O addresses



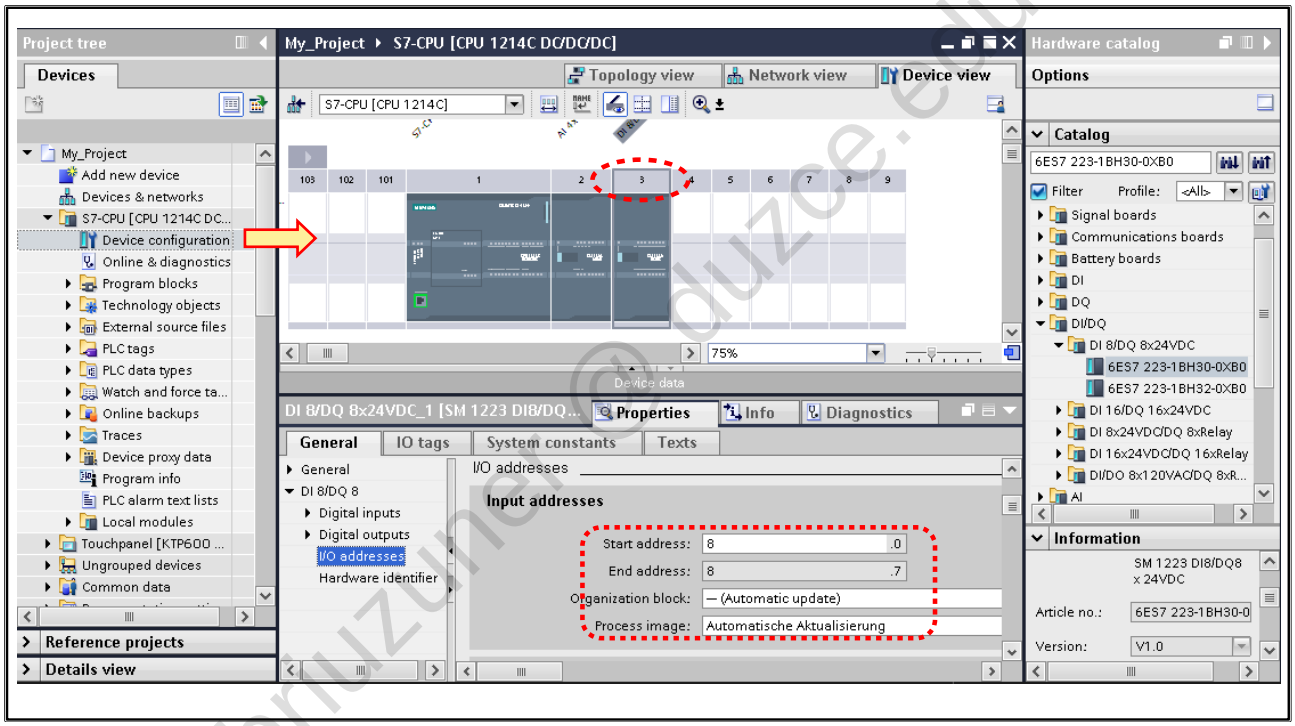
Task

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

What to Do

1. In the Device view, select the analog I/O module (see picture)
2. In the Inspector window, in the “Properties > AI4/AQ2” tab, you can enter the I/O address as shown in the picture
3. In the assignment of the process image, set “None” because the analog value of the module will not be used in the program
4. Save your project

4.9.11. Exercise 11: DI/DQ module: I/O addresses



Task

You are to parameterize the I/O addresses of the DI/DQ module to which the conveyor model is connected. The I/O addresses used in the STEP 7 program must match the addresses of the DI/DQ module parameterized here.

What to Do

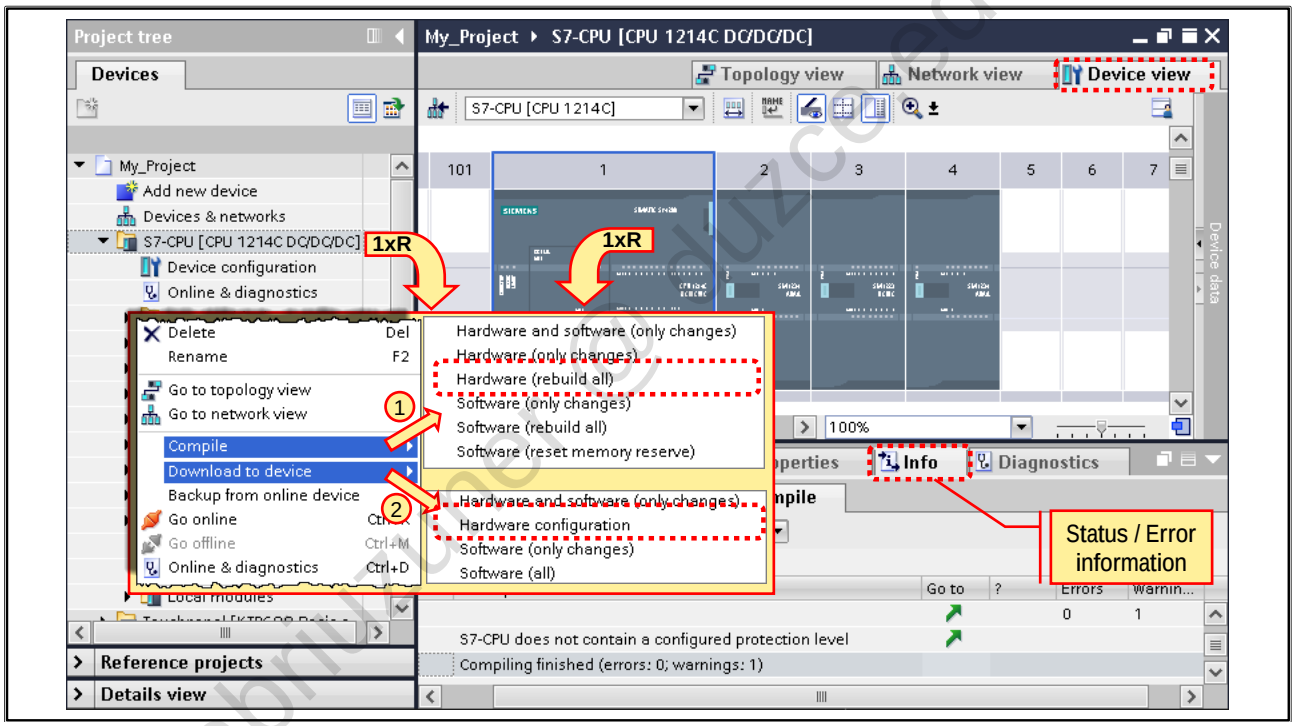
1. In the Device view, select the DI/DQ module (see picture)
2. In the Inspector window, in the “Properties > DI8/DQ8” tab you can enter the I/O address as shown in the picture
3. Set the update of the Process image to "Automatic update" so that the address is automatically updated by the system in every program cycle
4. Save your project

Notes:

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.

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.

4.9.12. Exercise 12: 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:

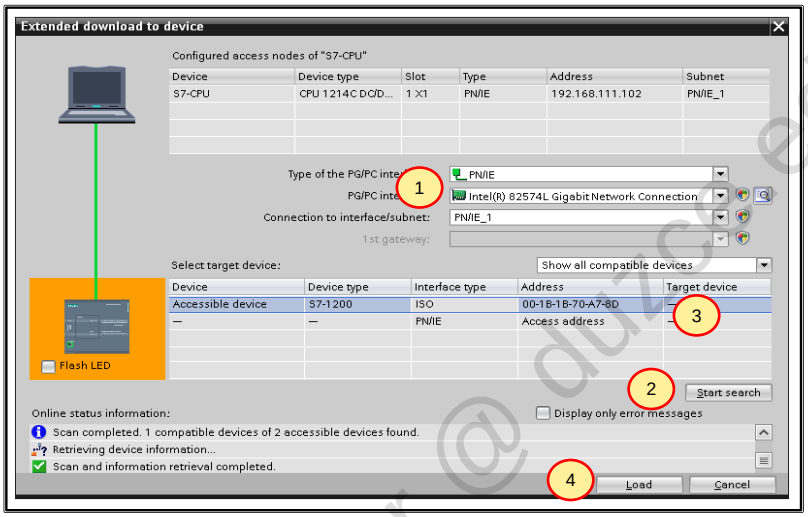
If the CPU doesn't have a program, the CPU does not go into RUN mode when there is a restart! That is, if, as shown in the picture, you only download the hardware configuration into the CPU in this exercise, and the CPU will not switch into the RUN mode with a subsequent restart!

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)

Fill in the dialog "Extended download to device" on the next page

Fill in the dialog "Extended download to device"



Fill in the dialog in the order which is shown in the picture above.

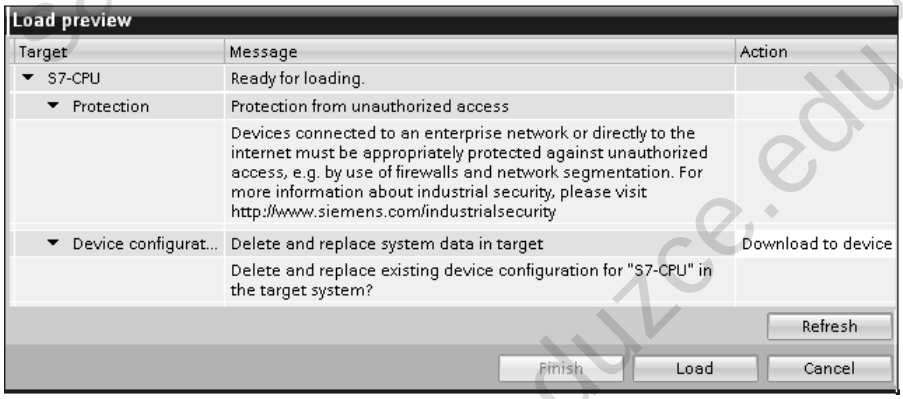
to 1 In "Connection to interface/subnet" select PN/IE

to 2: For refreshing the list "Show all compatible devices" activate "Start search"

to 3: In refreshed list "Show all compatible devices" select the S7-1200 CPU

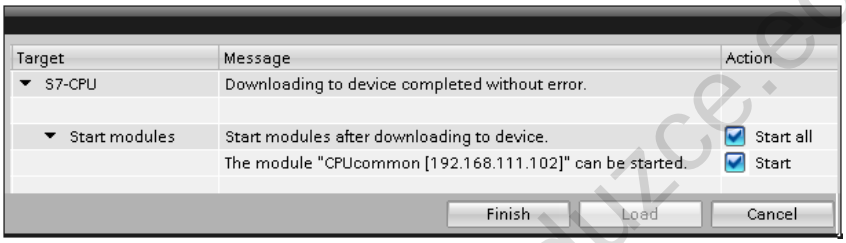
to 4: Start the download using the "Load" button

Accept the follow dialog and activate "Load"

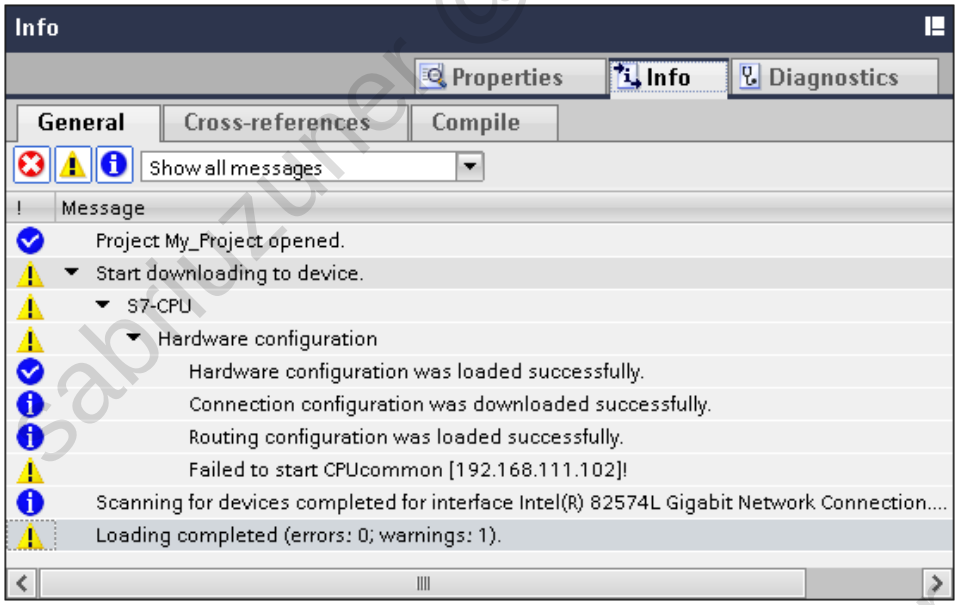


continue the next page

In the following dialog activate "Start" the CPU after the download

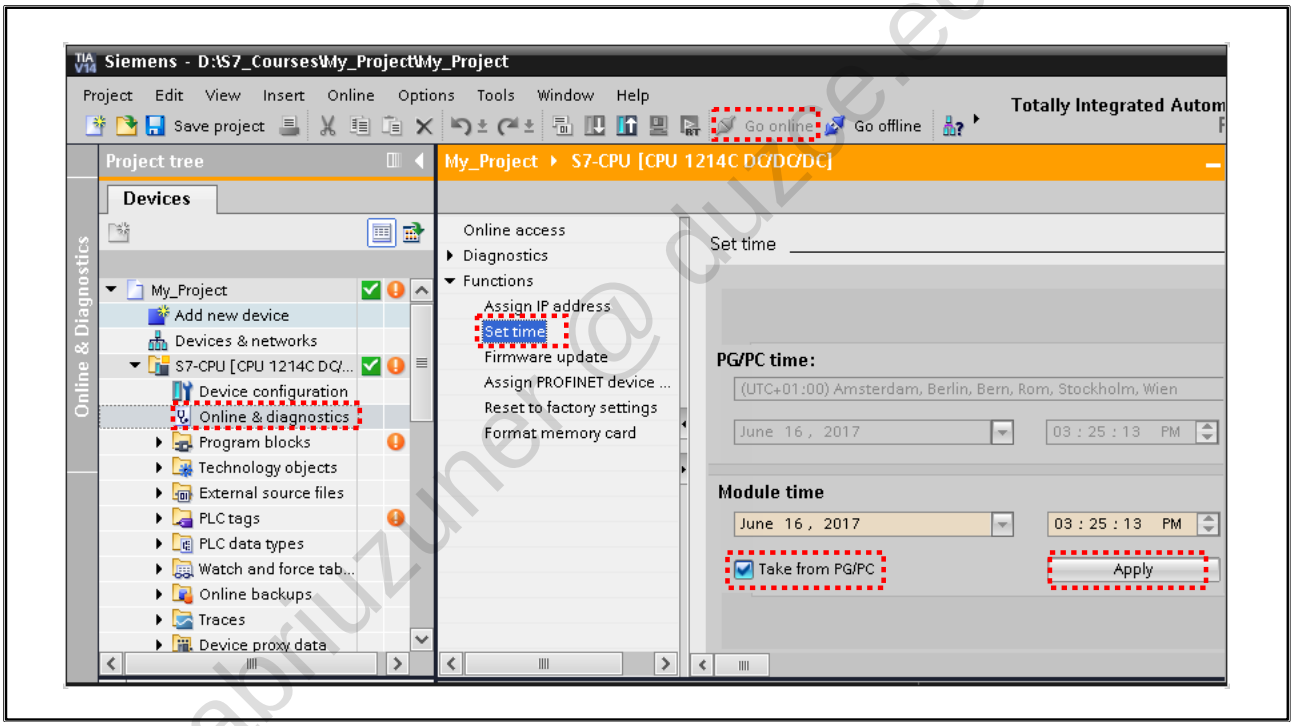


In the inspector window check if the download was successful



Note: The CPU couldn't be started because no user program has been downloaded.

4.9.13. Exercise 13: Setting the Time



Task

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

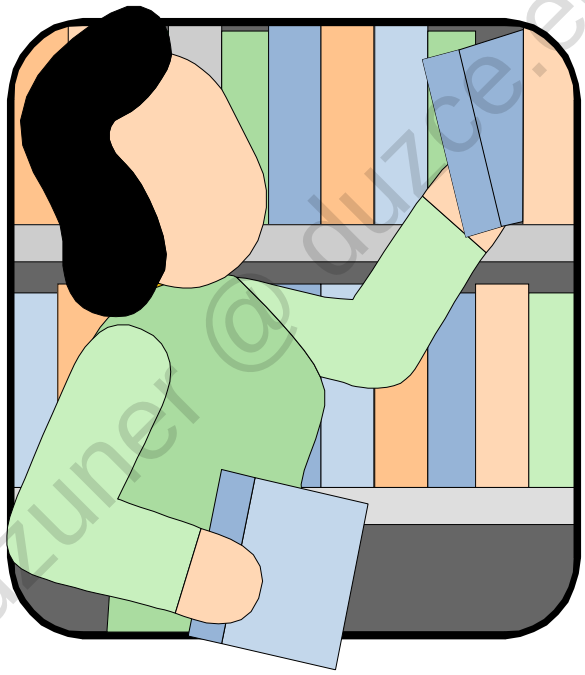
What to Do

1. Open the Online access of the S7-1200 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 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

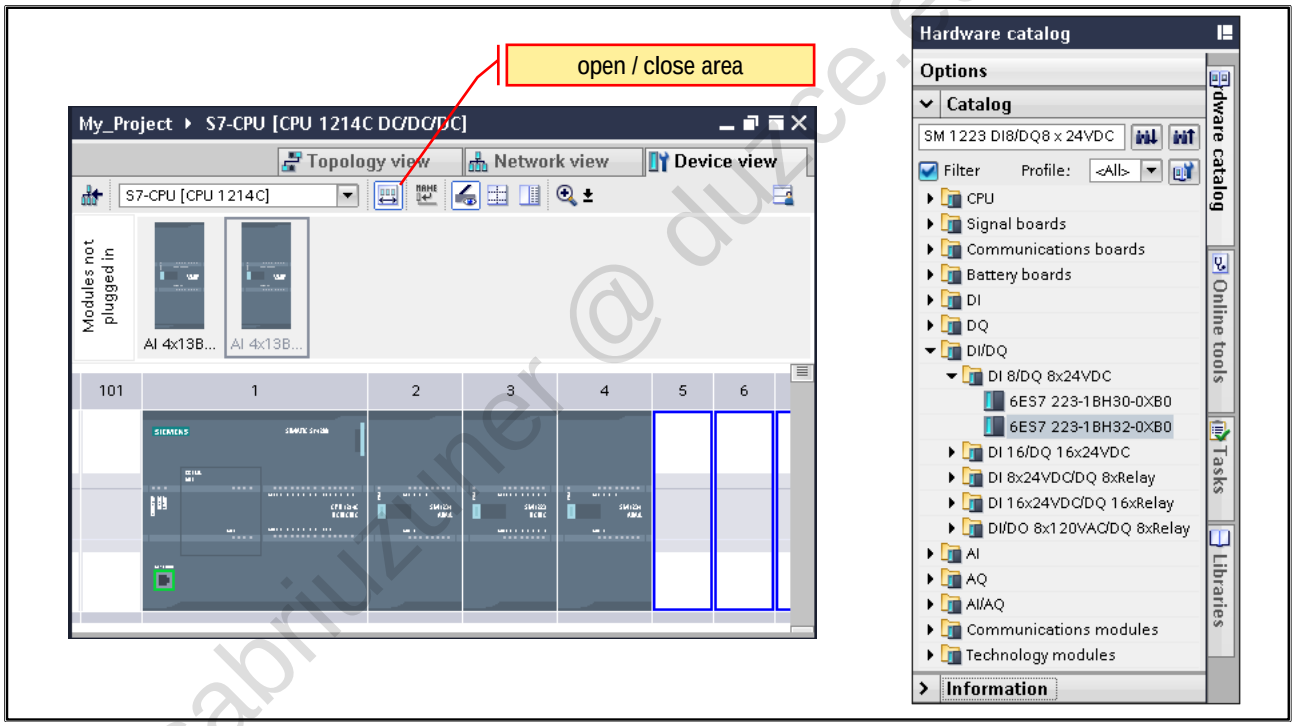
Note:

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

4.10. Additional Information



4.10.1. Area for Modules Not Plugged In



Area for Modules Not Plugged In

Modules that are intended to be assigned to a device, for example as the result of a copy action, but for which the corresponding rack does not have a free compatible slot are moved automatically to the "area for modules not plugged in". Modules are added to the area for modules not plugged in under the following conditions:

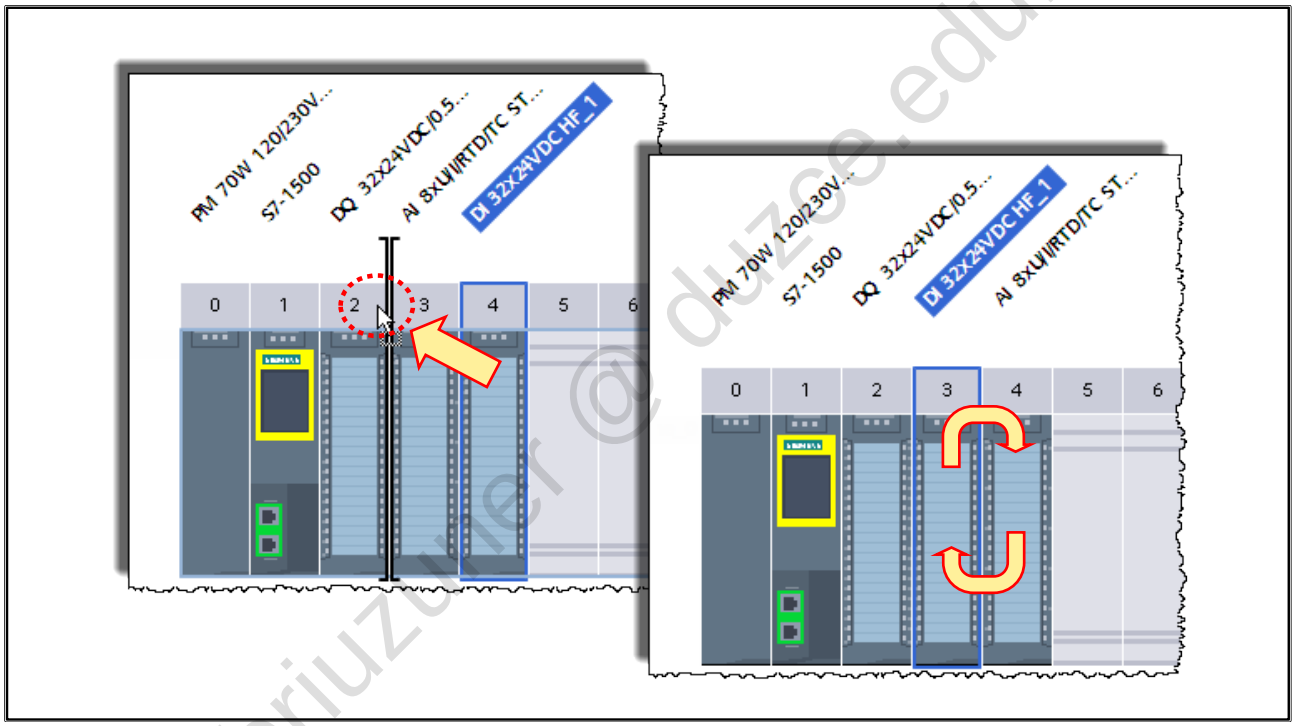
- When a module is copied or moved within the Project tree
- When a module is dragged to a device in the Network view, even though there is no compatible slot free on its rack
- In the Device view, when a module is directly moved into the area from the rack or Hardware catalog

The modules are ignored when downloading to the target system. There is therefore no consistency check for modules in the "area for modules not plugged in".

Modules in that area have these rules

- In the Project tree, the modules with parameterization (e.g. addresses) are stored under the relevant device in the folder "Local modules"
- The modules keep all settings and parameters
- The modules are ignored when downloading to the target system. There is therefore no consistency check for modules in the "area for modules not plugged in"
- Modules can be copied, cut or deleted by use of the context menu

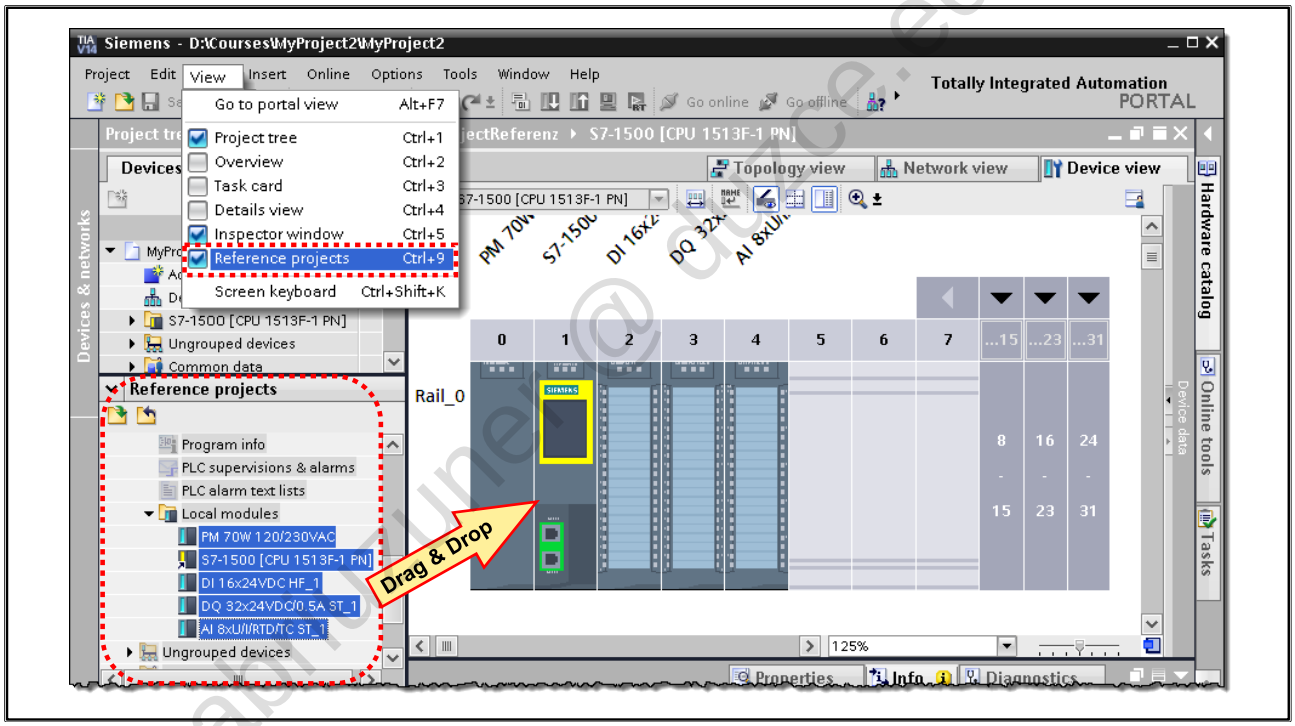
4.10.2. Swapping a Slot / Inserting a Module between Two Modules



Swapping a Slot or Inserting a Module between Two Modules

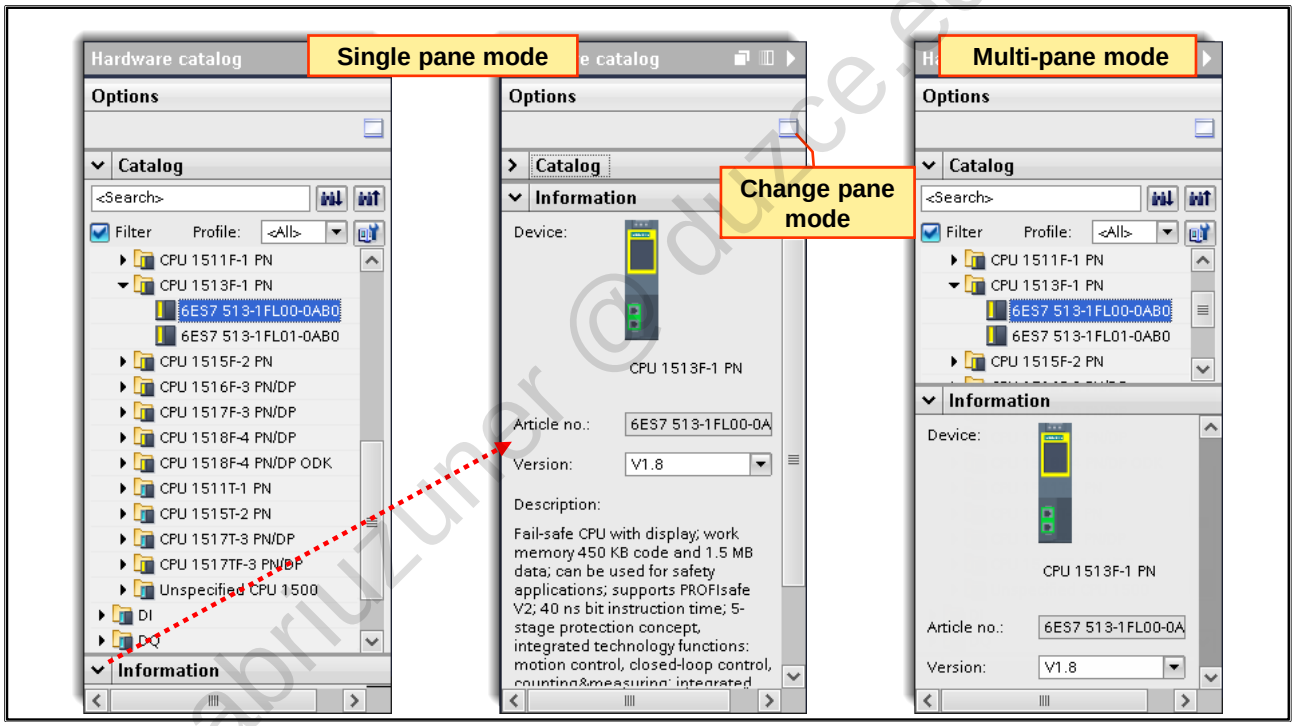
Using drag & drop, drag the modules in front of or behind the slot number until the marking appears at the desired location, as shown in the picture. The module is placed where the marking is and the modules behind it are moved one slot to the right.

4.10.3. 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.

4.10.5. ‘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

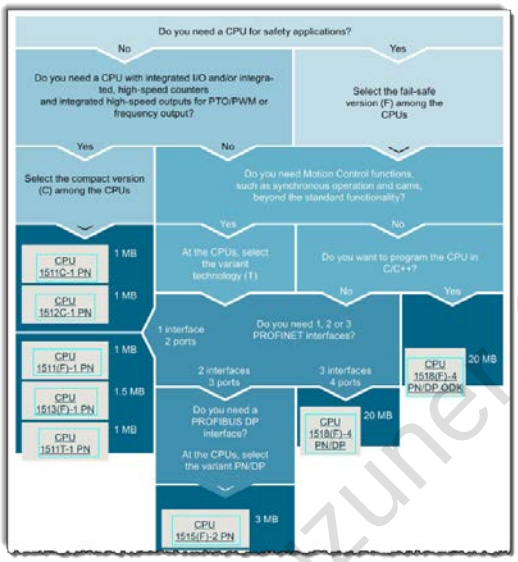
Setting for the Device Configuration

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.

Because 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.

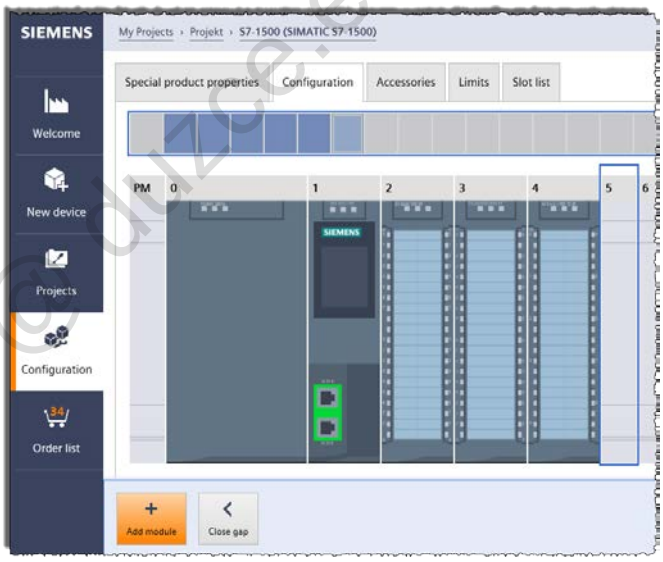
4.10.6. Selecting the Controller and the Modules

Manual: "SIMATIC S7-1500 / ET 200MP Automation system In a nutshell"



Entry ID: 109481357

Software: "TIA Selection Tool" (online or offline)



TIA Portal Information Center > Tools & Apps > Configurators

SIMATIC S7-1500 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 enough 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.

Contents

5

5. PLC Tags

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

... be familiar with PLC tags and their memory areas

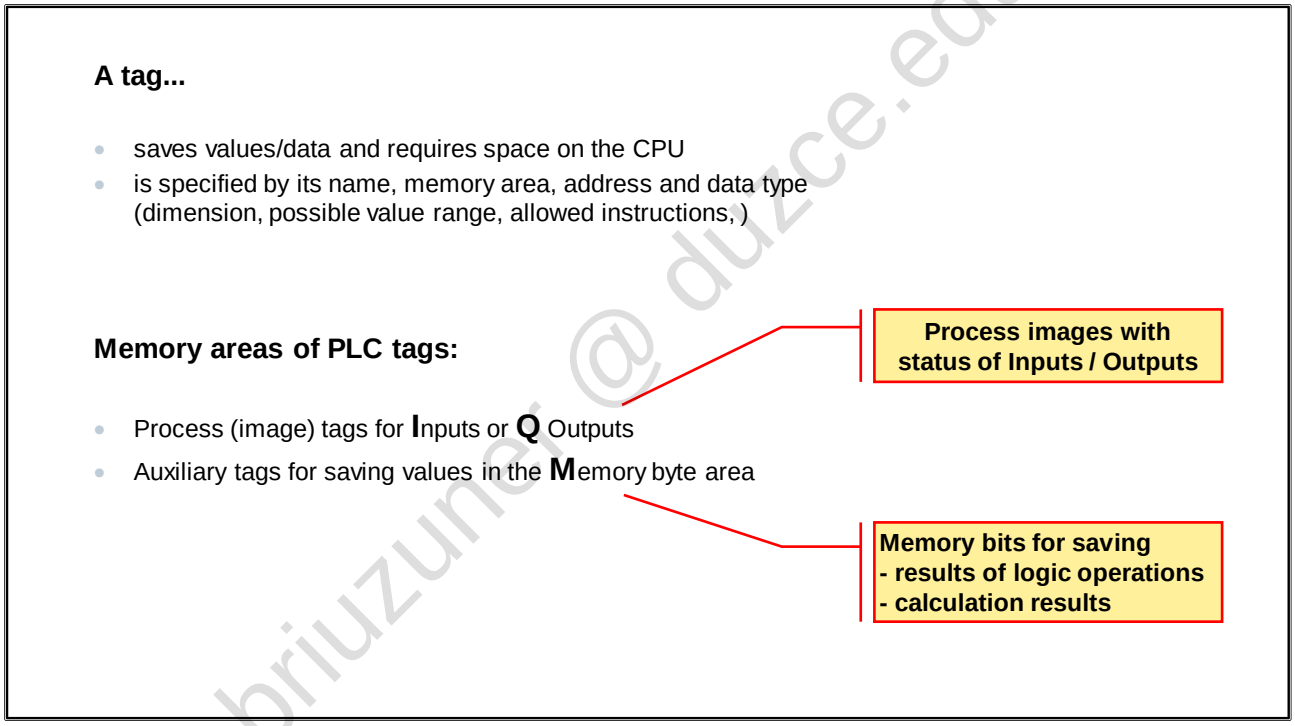
… be able to create PLC tags and address them

… know elementary data types

… be familiar with global constants and system constants

... learn how to monitor and modify PLC tags

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



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.

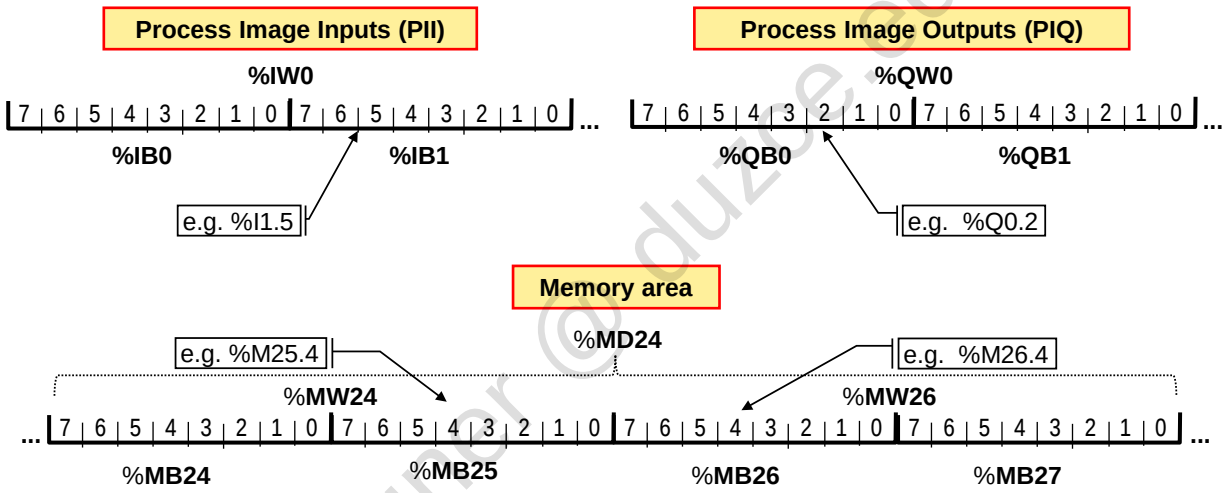
Memory Areas of Inputs and Outputs

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.

5.1.1. Addresses of PLC Tags



Bit	→	**%**{Area}{Byte-address}**.**{Bit-address}	→ (e.g. %M8.4)
Byte, Word, DWORD	→	**%**{Area}{Size}{Byte-address.}	→ (e.g. %MD24)
LREAL (64bit)	→	**%**{Area}{Address of the start bit}	(e.g. M15.0)

Addressing

An address exactly defines where values are written or from where they are 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.

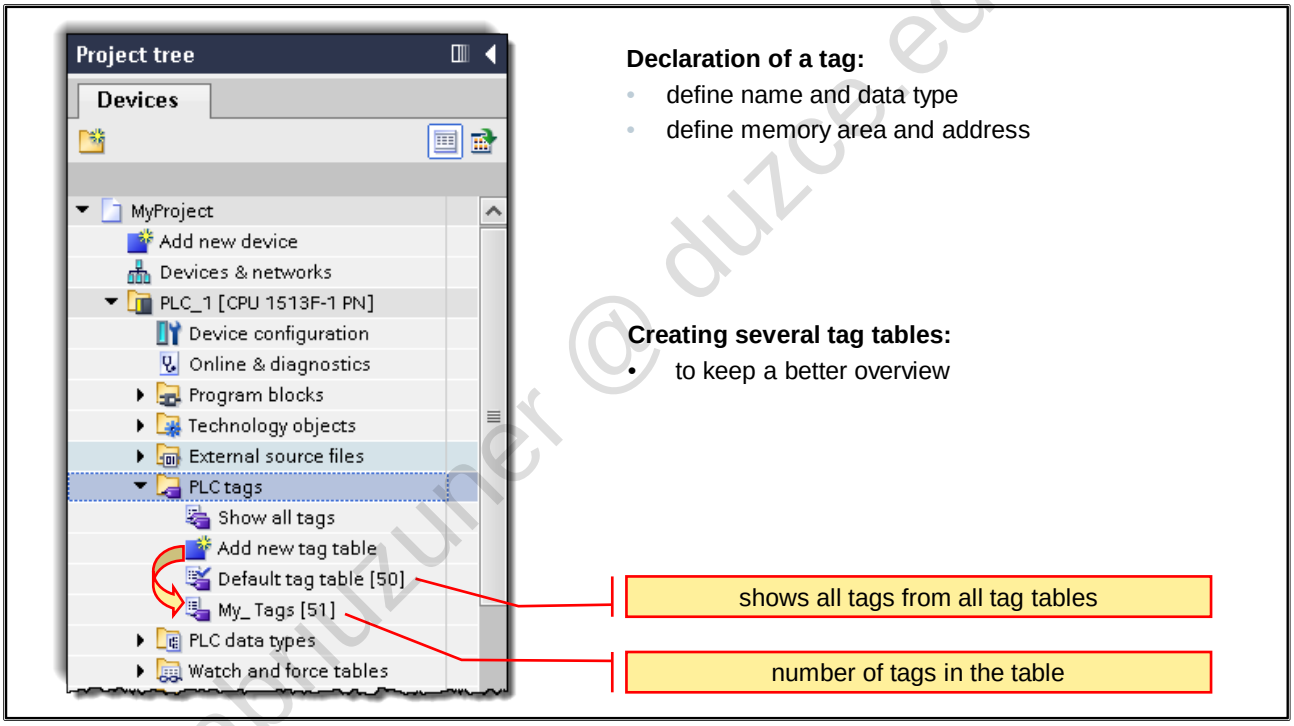
Address Style With/Without %

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.

5.3. PLC Tag tables



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.

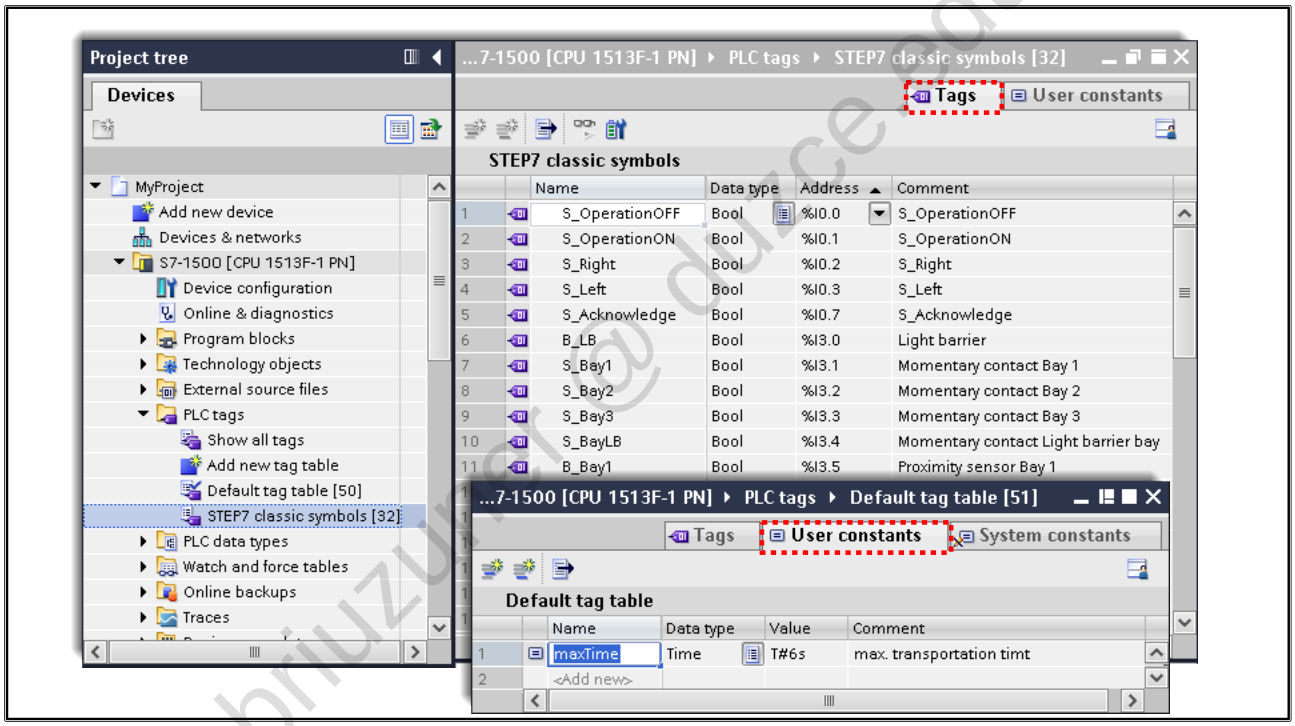
Tag Tables

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

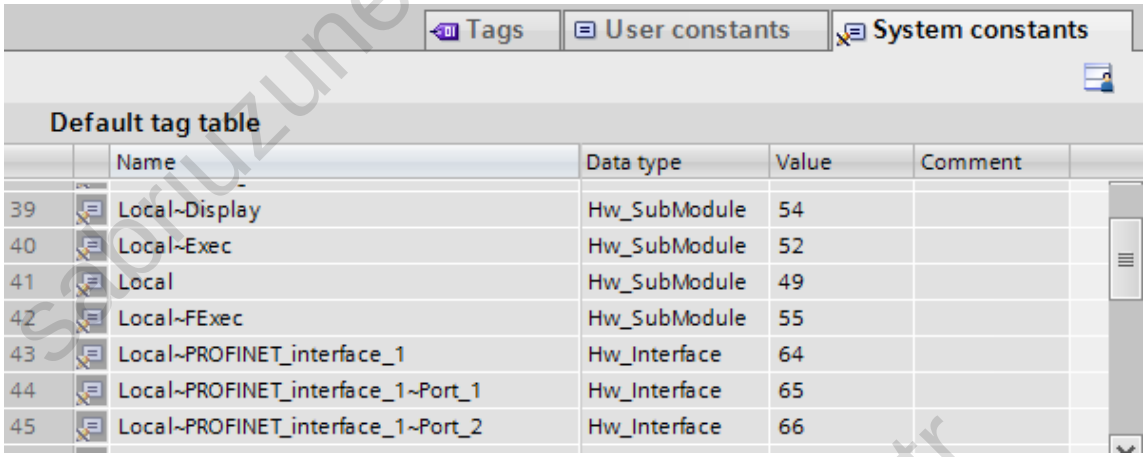
System constants are unique throughout the CPU. Global constants which are required by the system and are automatically created. System constants can, for example, serve the addressing and identification of hardware objects.

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.

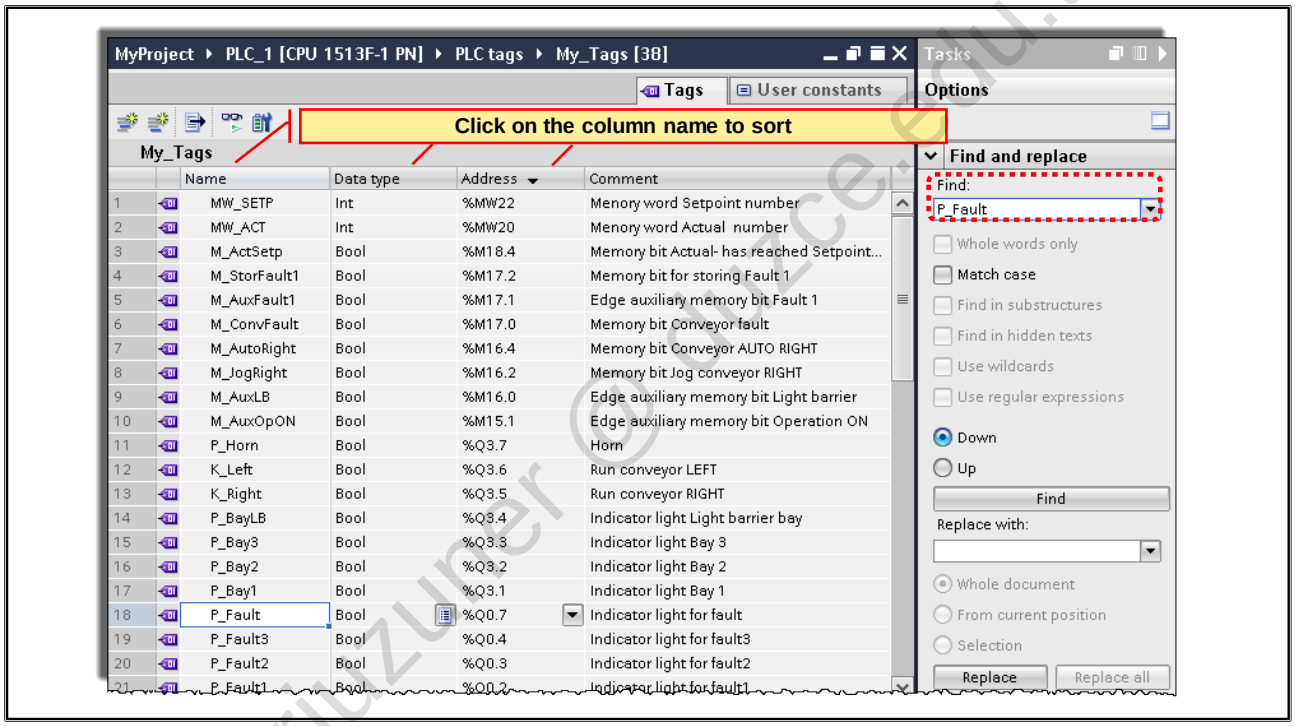


	Name	Data type	Value	Comment
39	Local~Display	Hw_SubModule	54	
40	Local~Exec	Hw_SubModule	52	
41	Local	Hw_SubModule	49	
42	Local~FExec	Hw_SubModule	55	
43	Local~PROFINET_interface_1	Hw_Interface	64	
44	Local~PROFINET_interface_1~Port_1	Hw_Interface	65	
45	Local~PROFINET_interface_1~Port_2	Hw_Interface	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. Finding / Replacing / Sorting PLC Tags



Sorting

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

Finding / Replacing

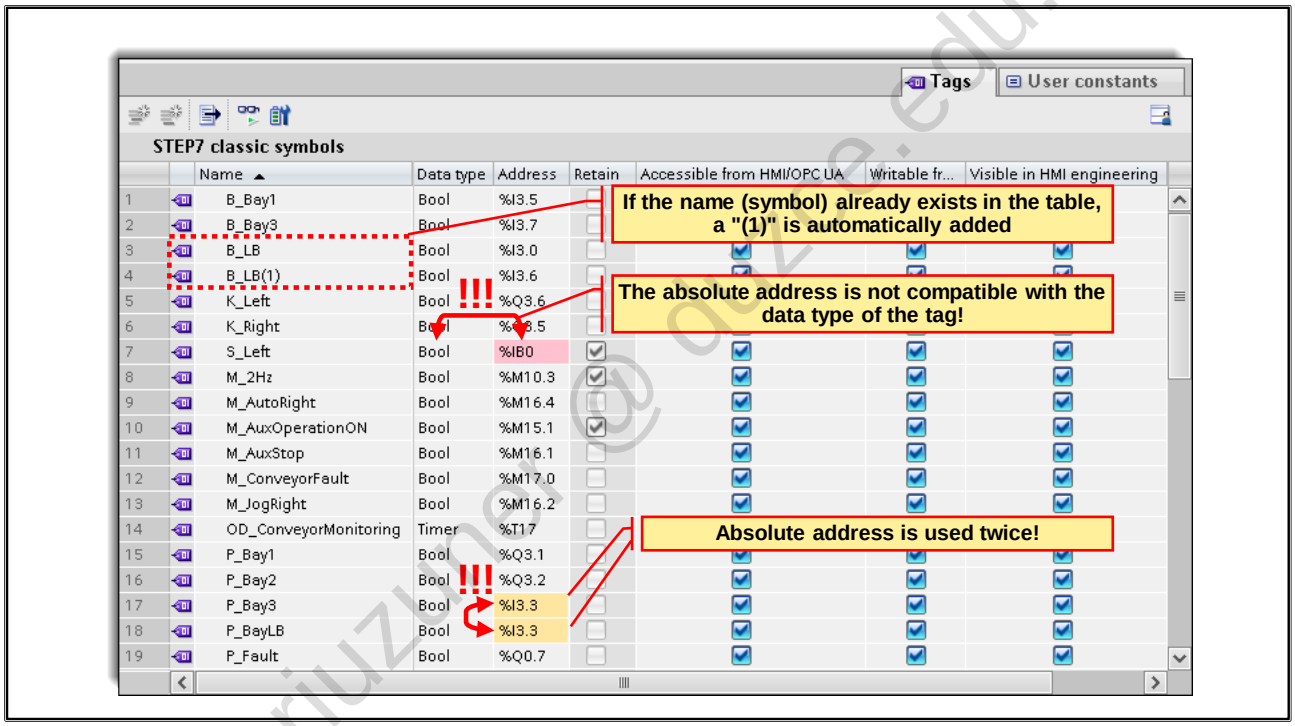
In the PLC tag table, tags can be found and replaced via the "Tasks" Task Card. Dummies can also be used (? for one character, * for several characters).

Example of "Find and replace":

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

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

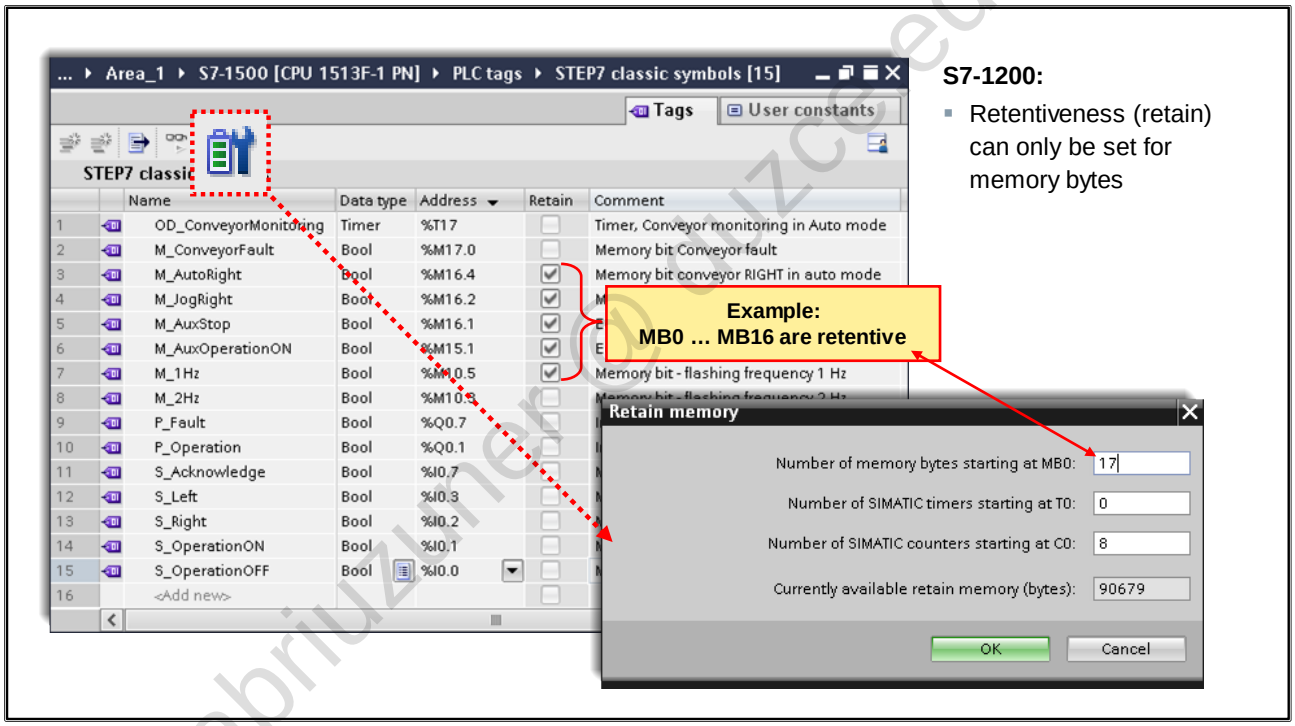
5.3.3. Error Indication in the PLC Tag Table



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.

5.3.4. Retentiveness of PLC Tags



S7-1200:

- Retentiveness (retain) can only be set for memory bytes

Retentive (Retain) Memory

S7-1500 CPUs have a retentive memory for storing retentive data even if power is lost or turned 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

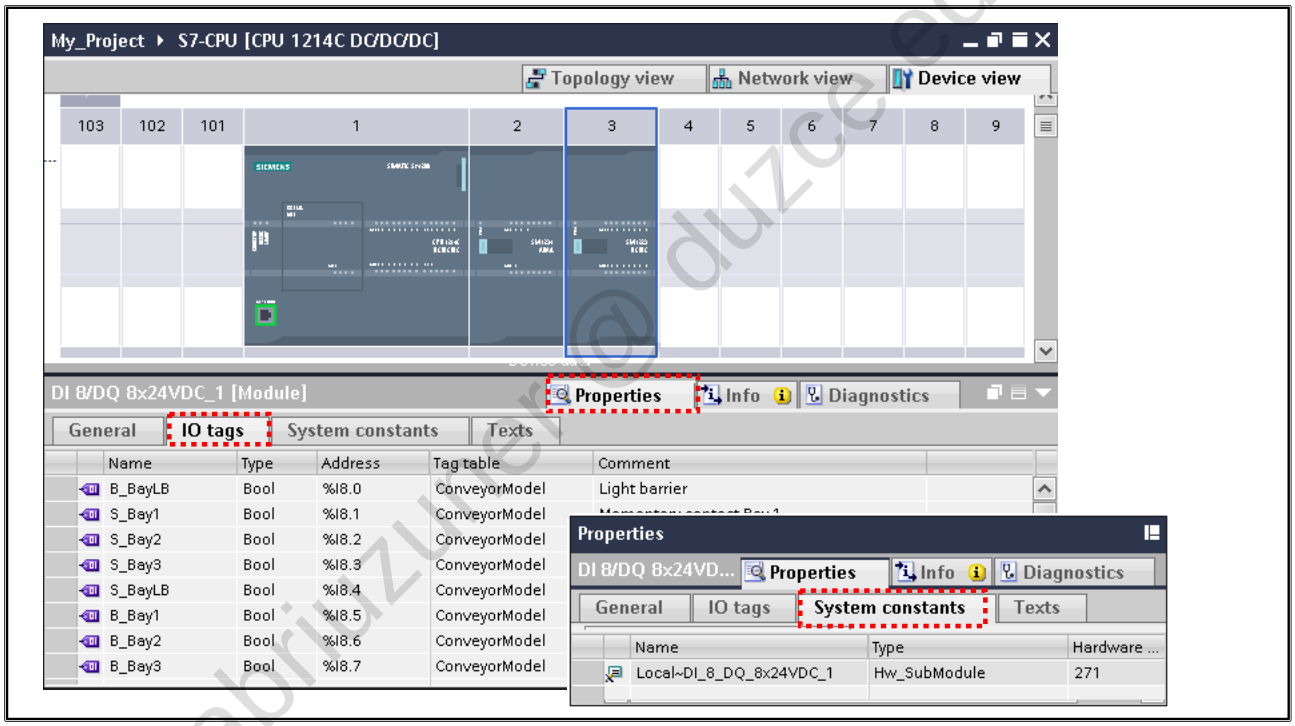
Certain tags of technology objects are always retentive, for example, adjustment values of absolute value encoders.

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.

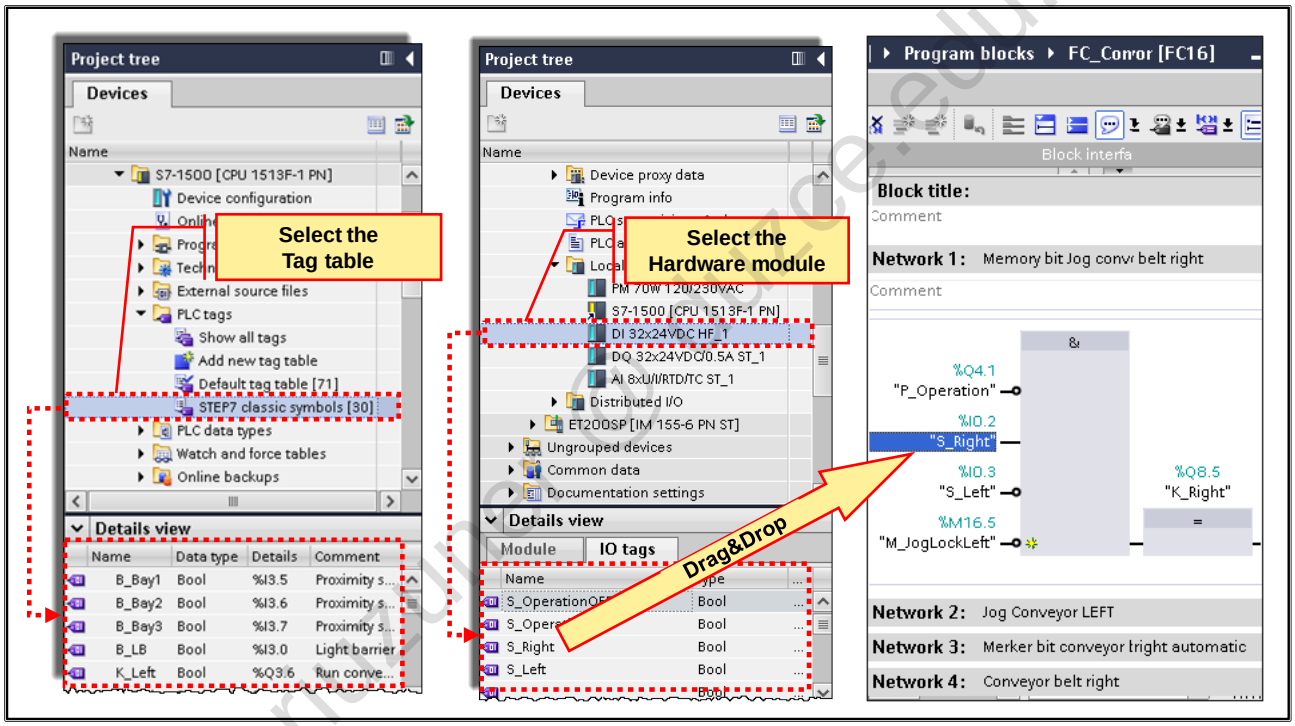
5.3.5. PLC Tags in the Device View



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.

5.3.7. PLC Tags: Detail view



Detail view

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

For example

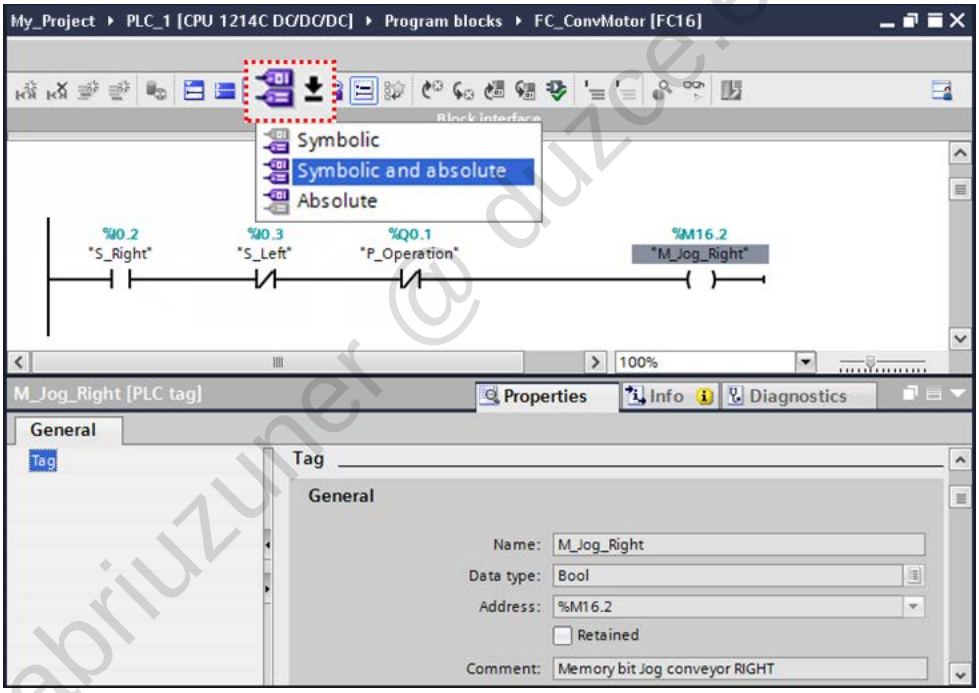
- ...tags of the selected tag table
- ...channels of the selected local modules and their tags

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

Using a Tag as Operand

During programming, the name of the tag can be entered, or, it can be selected from the automatic symbol selection as well as the Details view.

5.4. Absolute and Symbolic Addressing



Absolute and Symbolic Addressing

All global tags (such as, inputs, outputs, bit memories) 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).

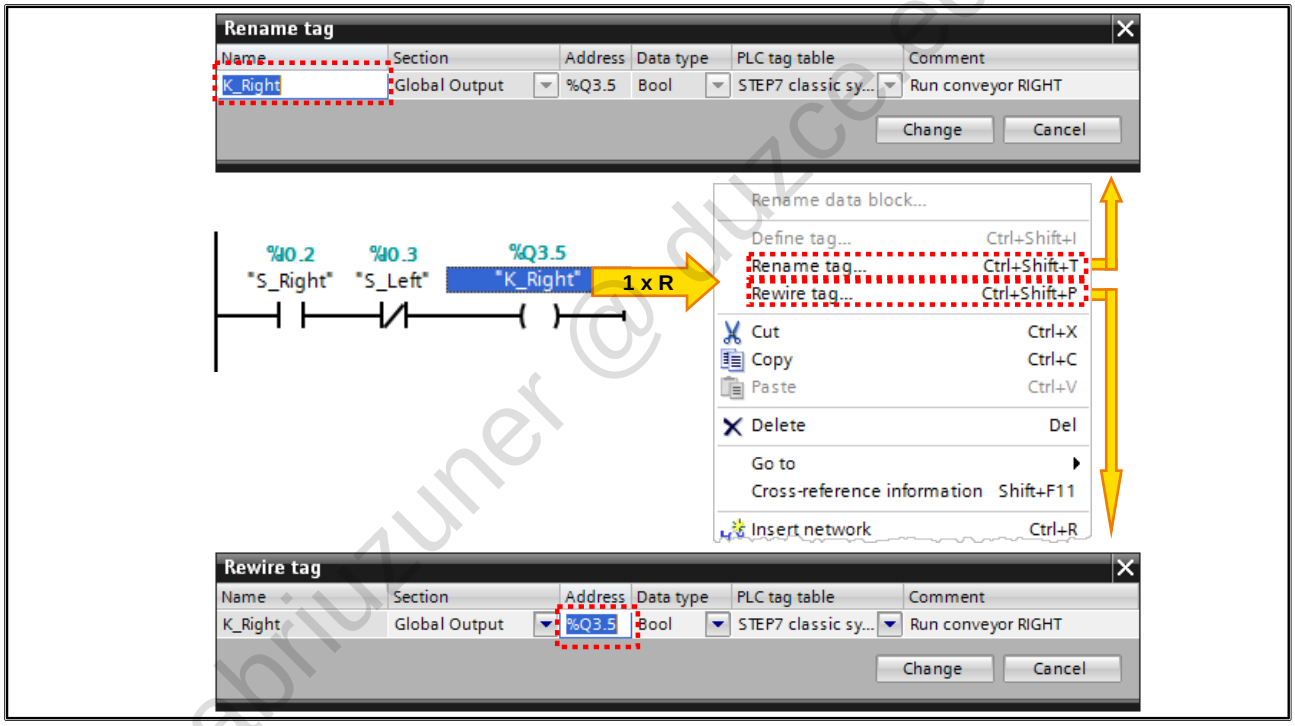
When you use a symbolic address (for example, "M_Jog_RIGHT") to which an absolute address has not yet been assigned, you can save the block but you cannot compile and download it into the controller.

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.

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 where they can also be edited.

5.5. Renaming / Rewiring Tags

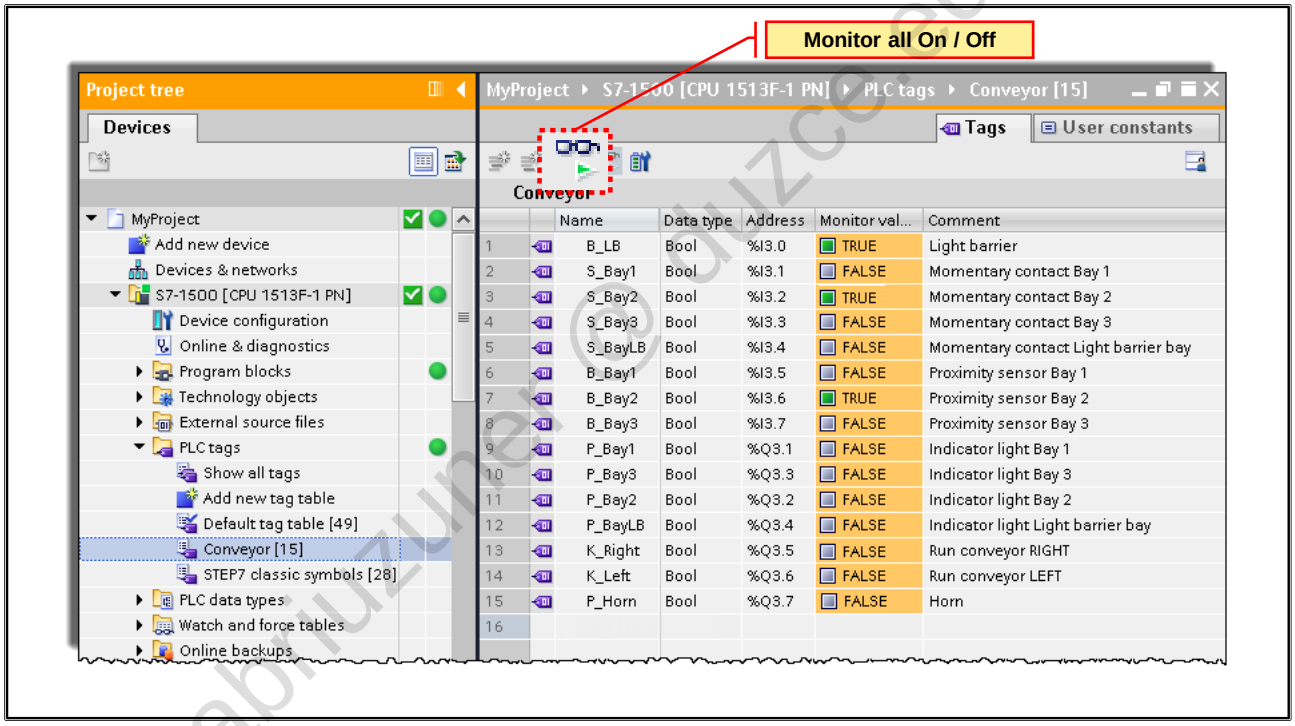


Renaming and Rewiring Tags

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

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

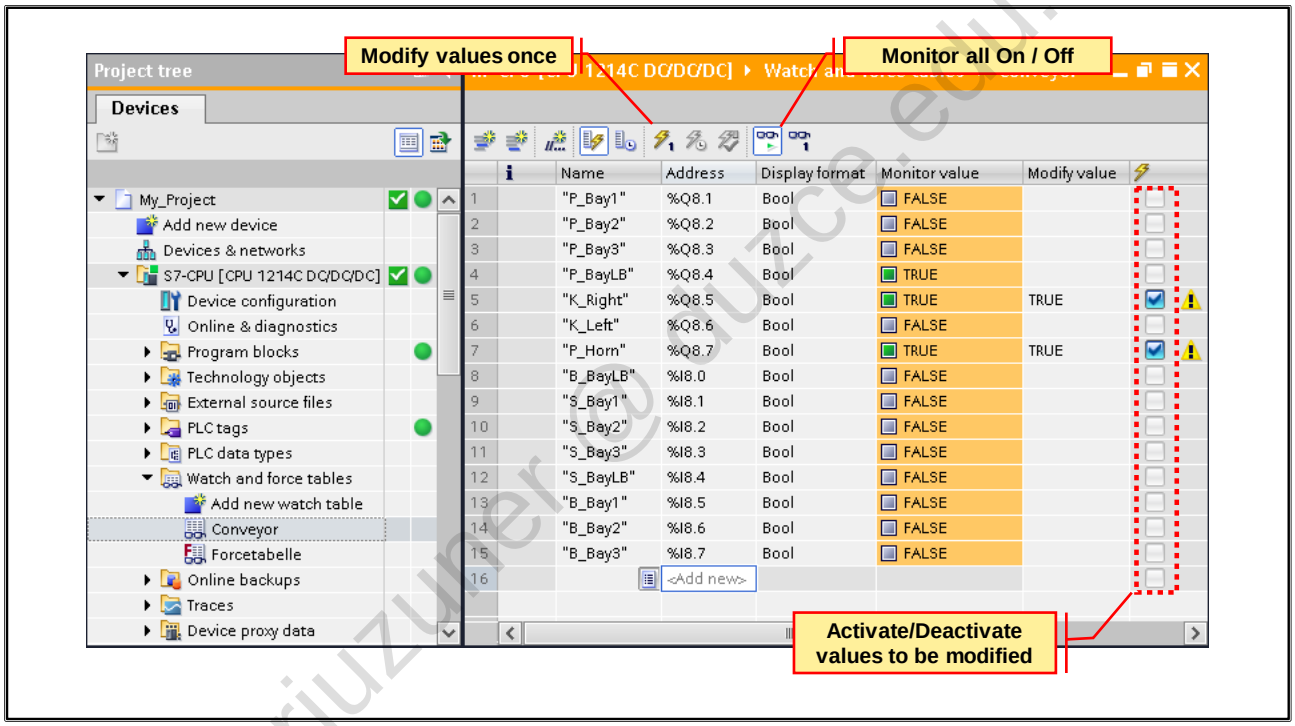
5.6. Monitoring PLC Tags



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.

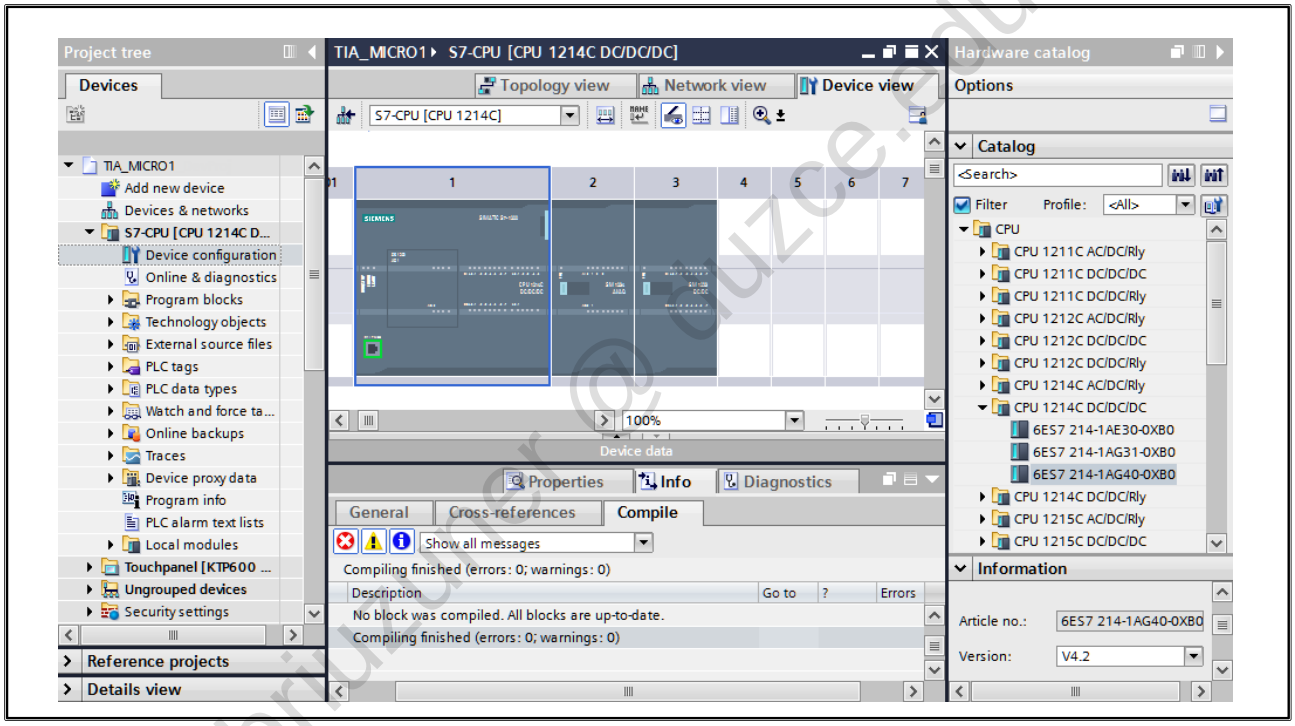
5.6.1. Modifying PLC Tags by means of the Watch Table



Before you can modify tags (variables) you must create a watch table in the ‘Watch and force tables’ folder.

Any tags (except block internal temporary tags) 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.

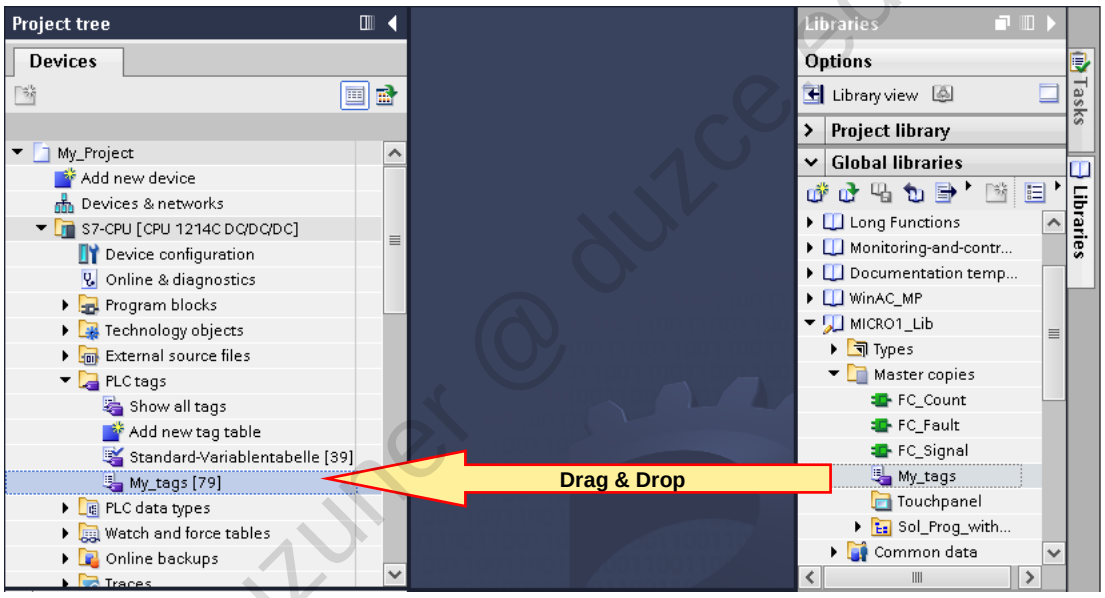
5.7. Task description: Insert and test the PLC tags



Task depiction

PLC tag tables have to be inserted in the project and the tags have to be tested. The PLC tag tables can be copied form the global library into the project. PLC tags can be monitored in the tag table or in a watch table.

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



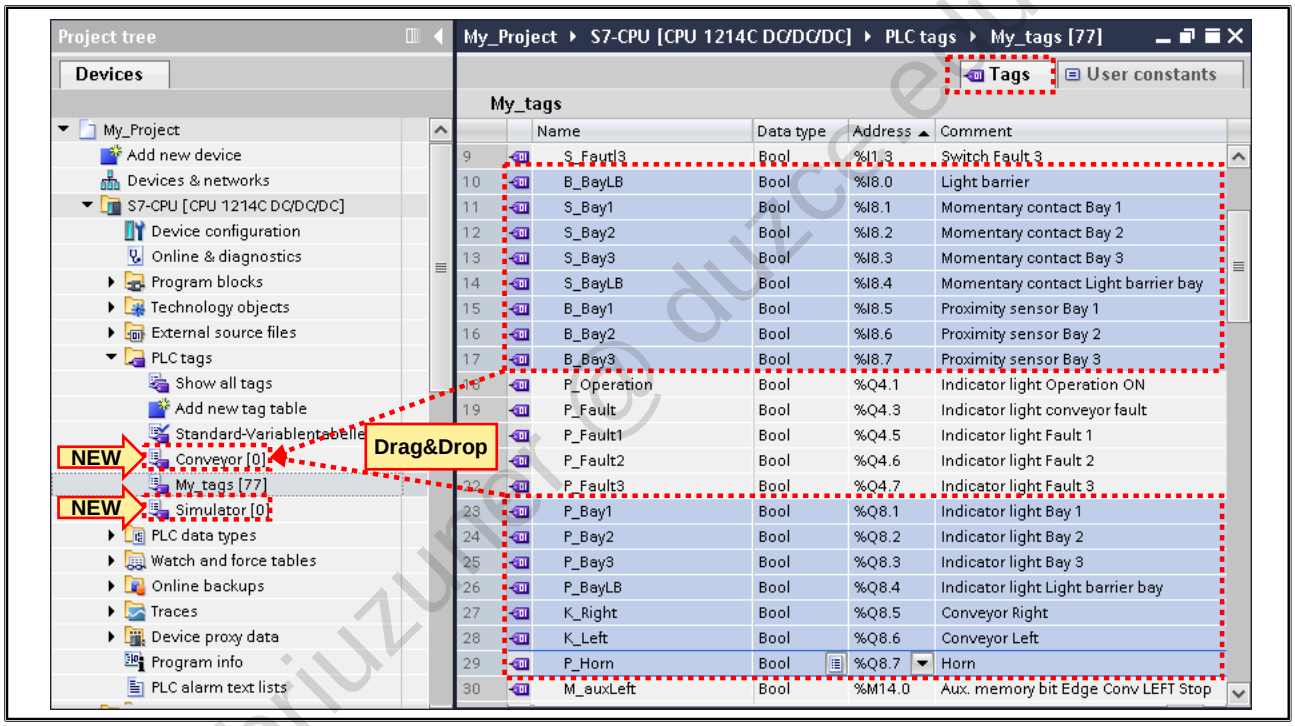
Task

You are to copy the prepared PLC tag table "My_Tags" from the "MICRO1_Lib" library into your own project.

What to Do

1. In the Task Card “Libraries > Global libraries”, open the library "MICRO1_Lib" which is located in the folder C:_Archives\TIA-MICRO1*[Version]* 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

5.7.2. Exercise 2: Creating the PLC tag table "Conveyor" and "Simulator"



Task

You are to create the new PLC tag tables "Conveyor" for the inputs and outputs of the conveyor model and "Simulator" for the Inputs and outputs of the simulator.

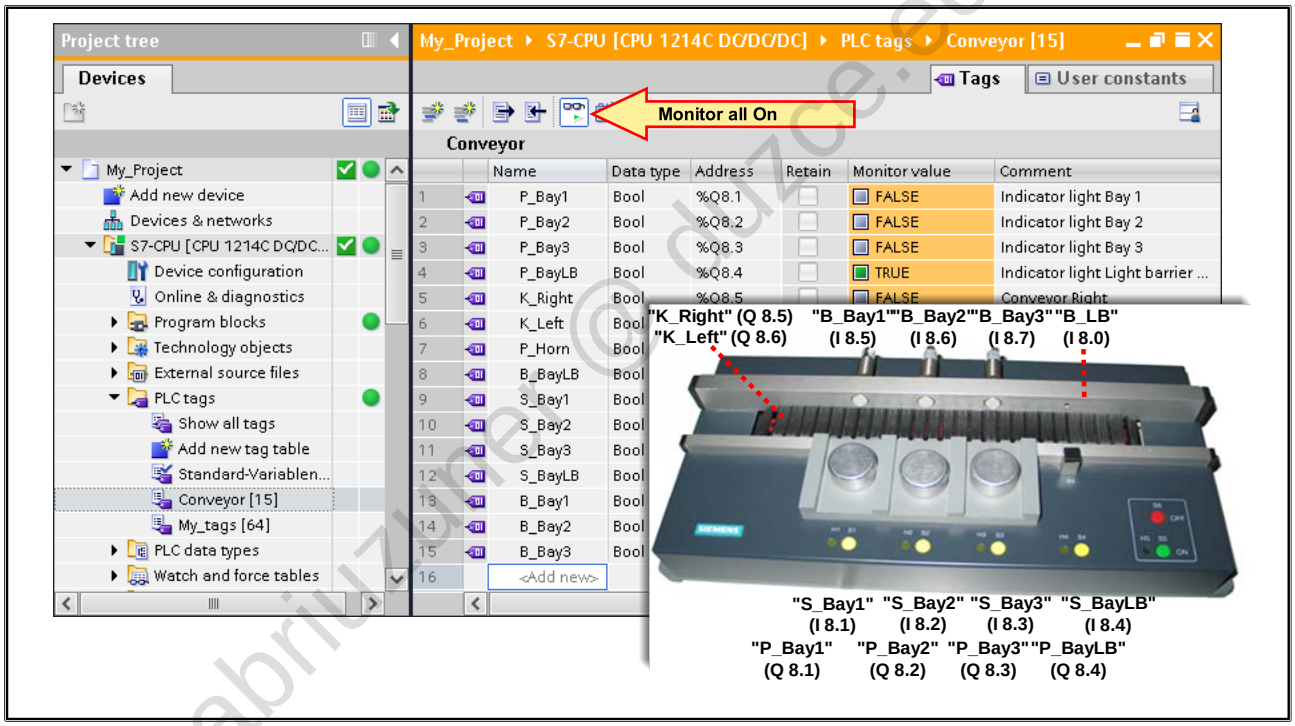
What to Do

1. Using "Add new tag table", create the new PLC tag table "Conveyor"
2. Open the PLC tag table "My_Tags", select the inputs and outputs of the conveyor and move them inputs and outputs into the newly created PLC tag table "Conveyor" using drag & drop (mouse pointer on tag icons, see picture)
3. With the same work flow move the inputs and outputs of the simulator into the newly created PLC tag table "Simulator"
4. 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".

5.7.3. Exercise 3: Monitoring the “Conveyor” PLC Tag Table



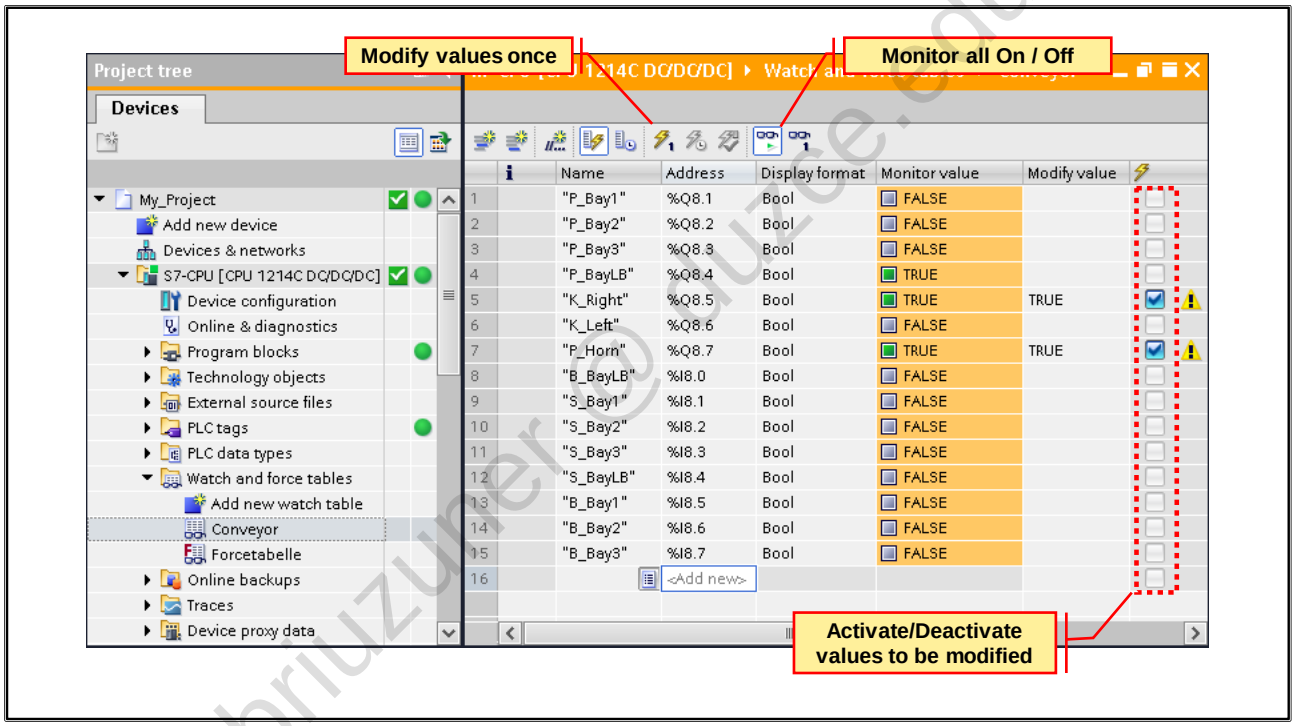
Task

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. Switch your conveyor model on
3. Open the newly created PLC tag table "Conveyor" and activate the function "Monitoring"
4. On the conveyor model, press the Bay pushbuttons and check whether Status '1' or "TRUE" is displayed at the corresponding inputs

5.7.4. Exercise 4: Modifying using the Watch Table



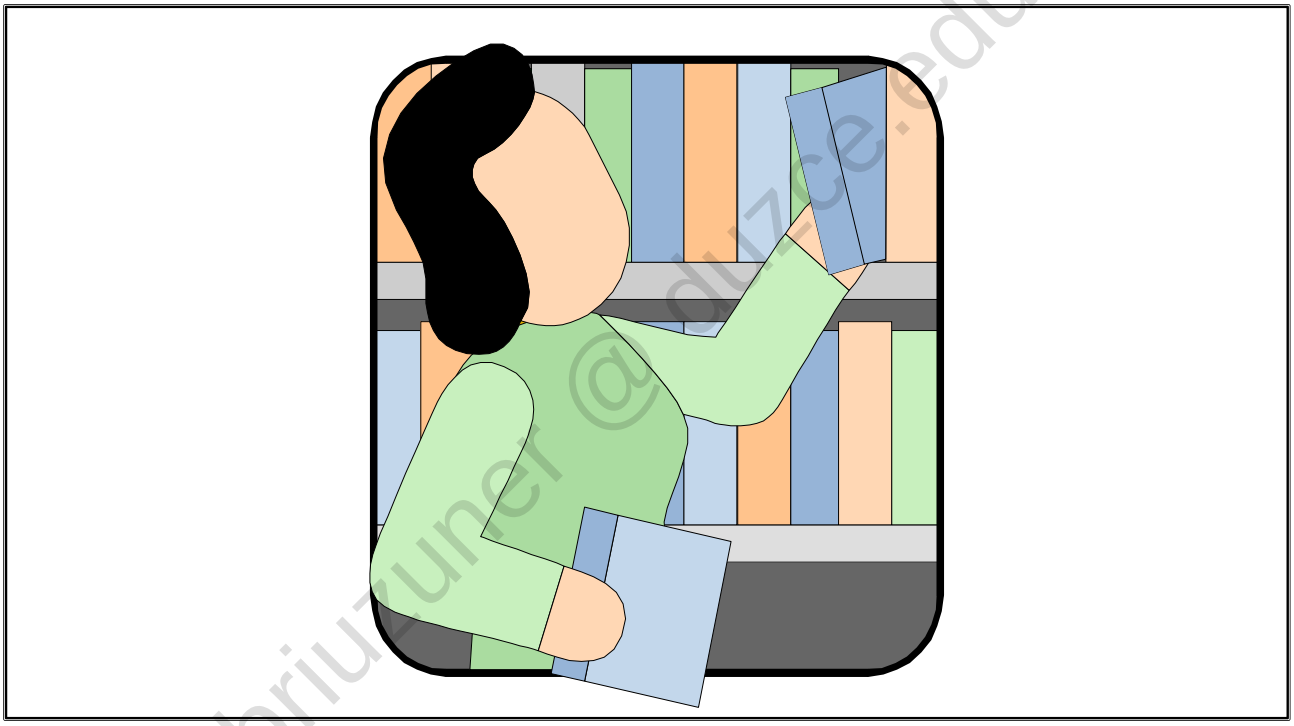
Task

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

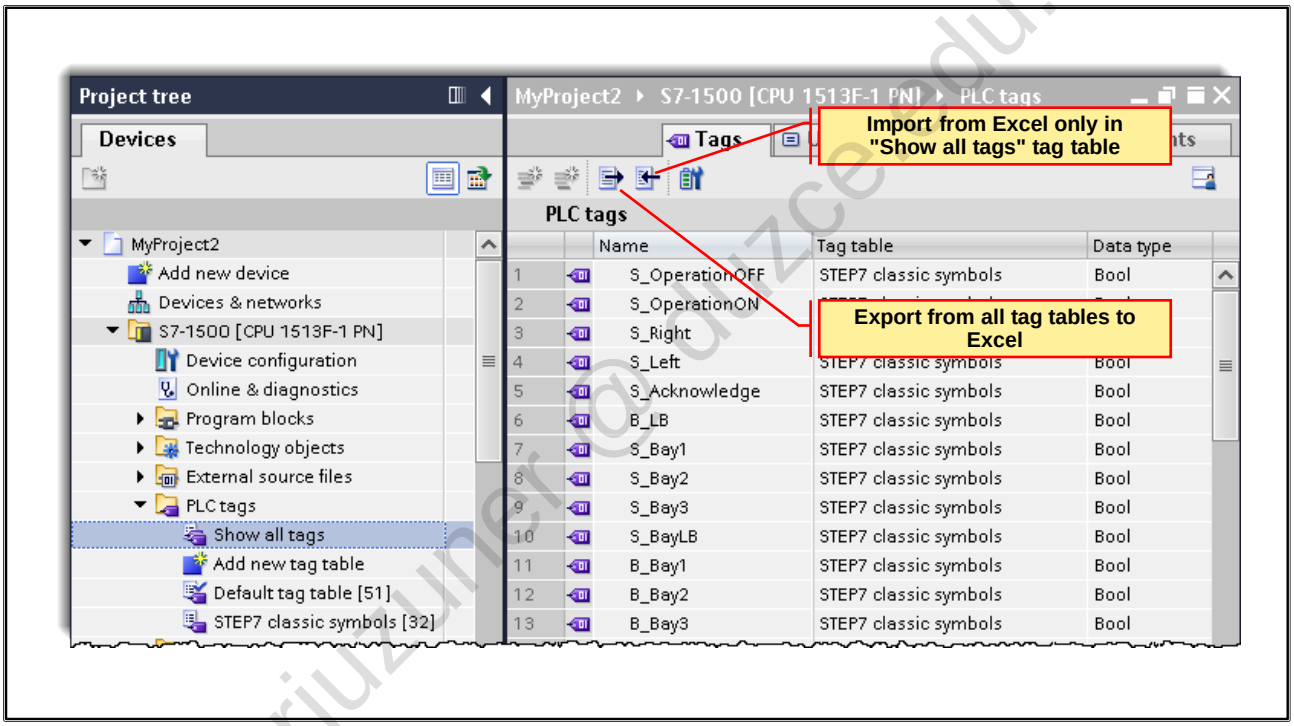
What to Do

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

5.8. Additional Information



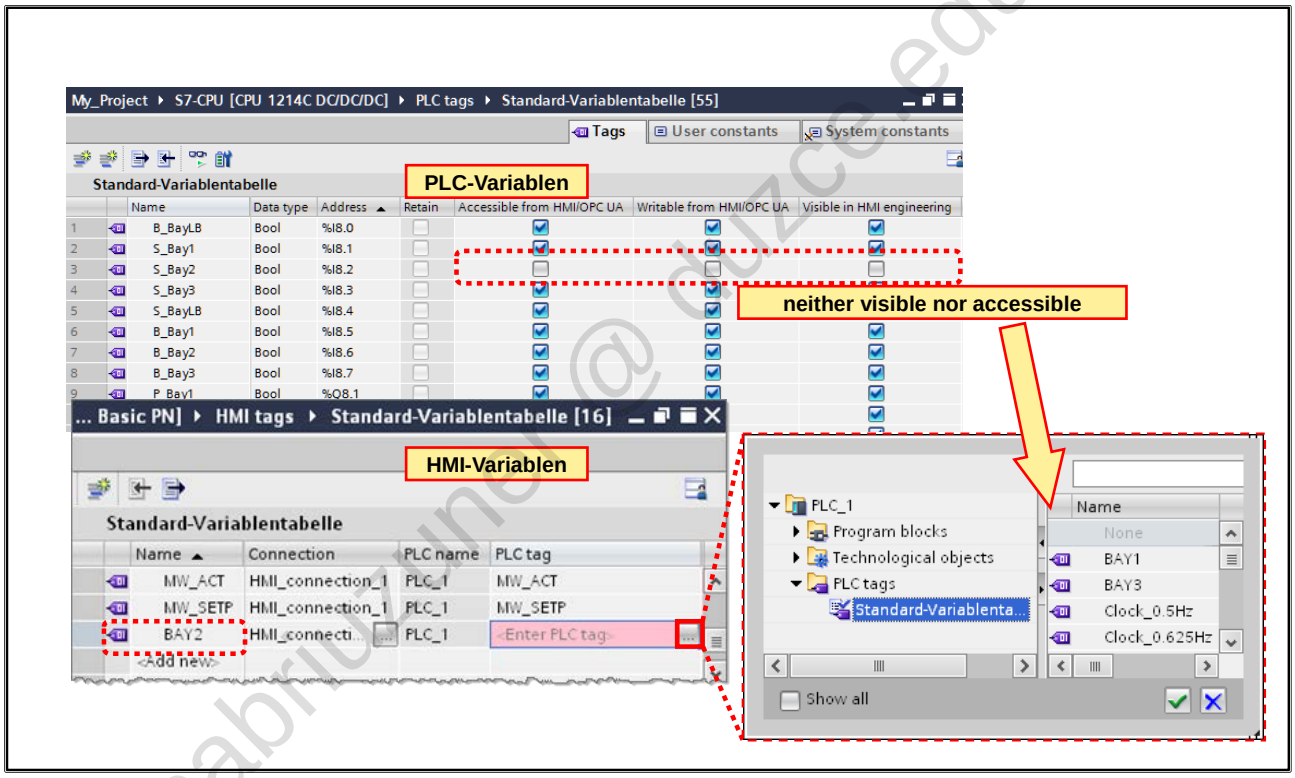
5.8.1. Copy & Paste PLC Tags to Excel



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.

5.8.2. HMI Access to PLC Tags



HMI Tag Access

Here, protective mechanisms can be declared with whose help unwanted accesses to PLC tags from HMI devices can be prevented:

- "Accessible from HMI/UPC UA":
  Shows whether the HMI / OPC UA can access the tag during runtime
- Writable from HMI/OPC UA:
  Shows whether HMI / OPC UA can write the tag during runtime
- Visible in HMI engineering:
  Shows whether the tag is shown in tag selection in HMI

Contents

6

6. 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 familiar with the meaning of process images (PII, PIQ)

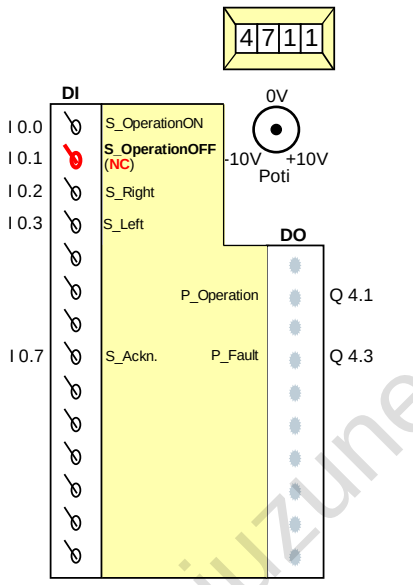
... be able to explain the principle of cyclic program execution

... be familiar with the LAD and FBD presentation formats and be able to select them

... be able to edit, save and load a block using the editor

... be able to carry out a simple program test with "Monitor block"

6.1. Plant Description: The Conveyor Model as Distribution Conveyor



"FC_Mode":	**Operation ON/OFF**
"FC_ConvMotor":	**Controlling the conveyor motor**
"FC_Fault":	**Monitoring the transport sequences for time**
"FC_Count":	**Counting the transported parts**
"FC_Indicate":	**Controlling the indicator lights**

Conveyor Model as Distribution Conveyor

On the conveyor, parts are transported from Bay 1 or 2 through to the light barrier, whereby the transport sequences are monitored for time and the transported parts are counted.

Operating Modes

The operation can be switched on via the switch "S_OperationON" (I 0.0) and switched off via the switch "S_OperationOFF" (I 0.1, NC).

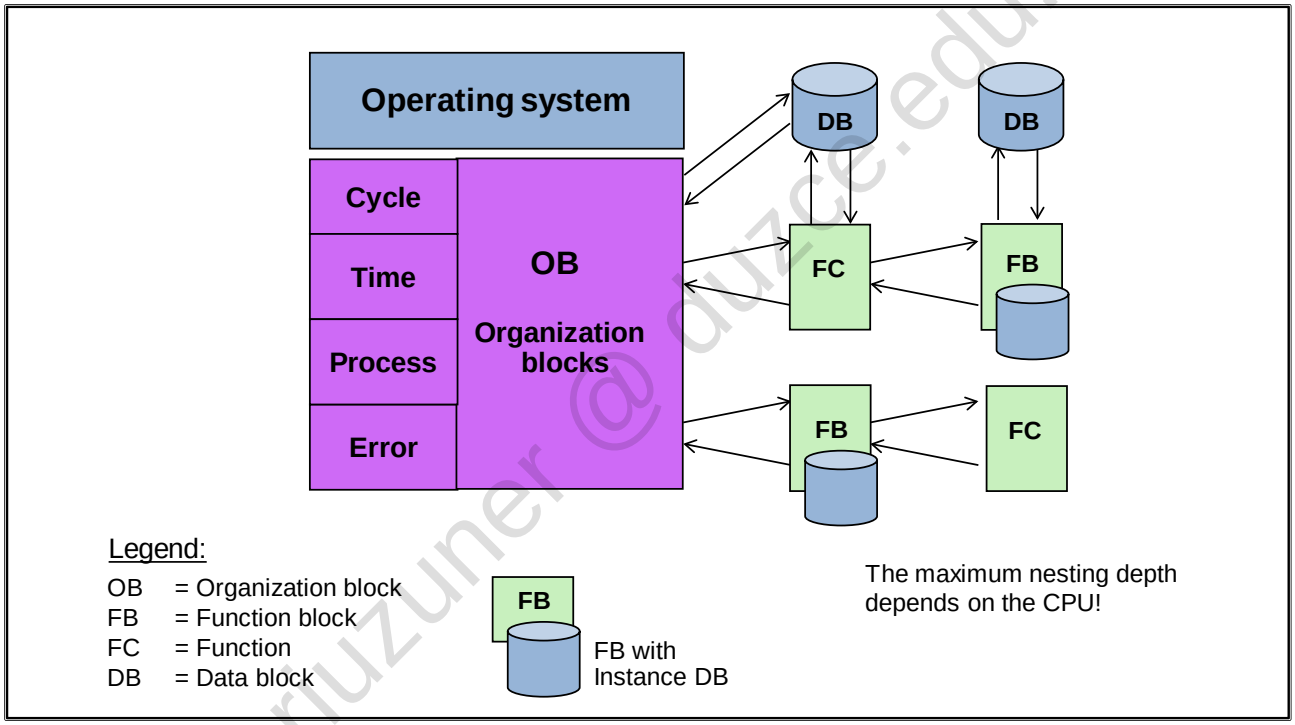
Operation OFF ("P_Operation", Q4.1 = 0)

When the operation is switched off, the conveyor can be jogged to the right via the switch "S_Right" (I 0.2) and to the left via the switch "S_Left" (I 0.3).

Operation ON ("P_Operation", Q4.1 = 1)

When the operation is switched on, parts are transported from Bay 1 or 2 through the light barrier. For this, the part must be placed on the conveyor at Bay 1 or 2 and the relevant bay pushbutton must be pressed. The condition is that exactly one of the two bays is occupied. The transport sequences are monitored for time. If a transport sequence takes longer than 6 seconds, "P_Operation" (Q4.1) and with that the conveyor motor is automatically switched off. The fault is indicated with a 2Hz flashing light on the Simulator-LED "P_Fault" (Q4.3). "P_Operation" (Q4.1) can only be switched back on and a new transport sequence can only then be restarted when the fault has been acknowledged with the switch "S_Acknowledge" (I 0.7).

6.2. Types of Program Blocks



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. The entire program can be stored in OB1 that is cyclically called by the operating system (linear program) or the program can be divided and stored in several blocks (structured program).

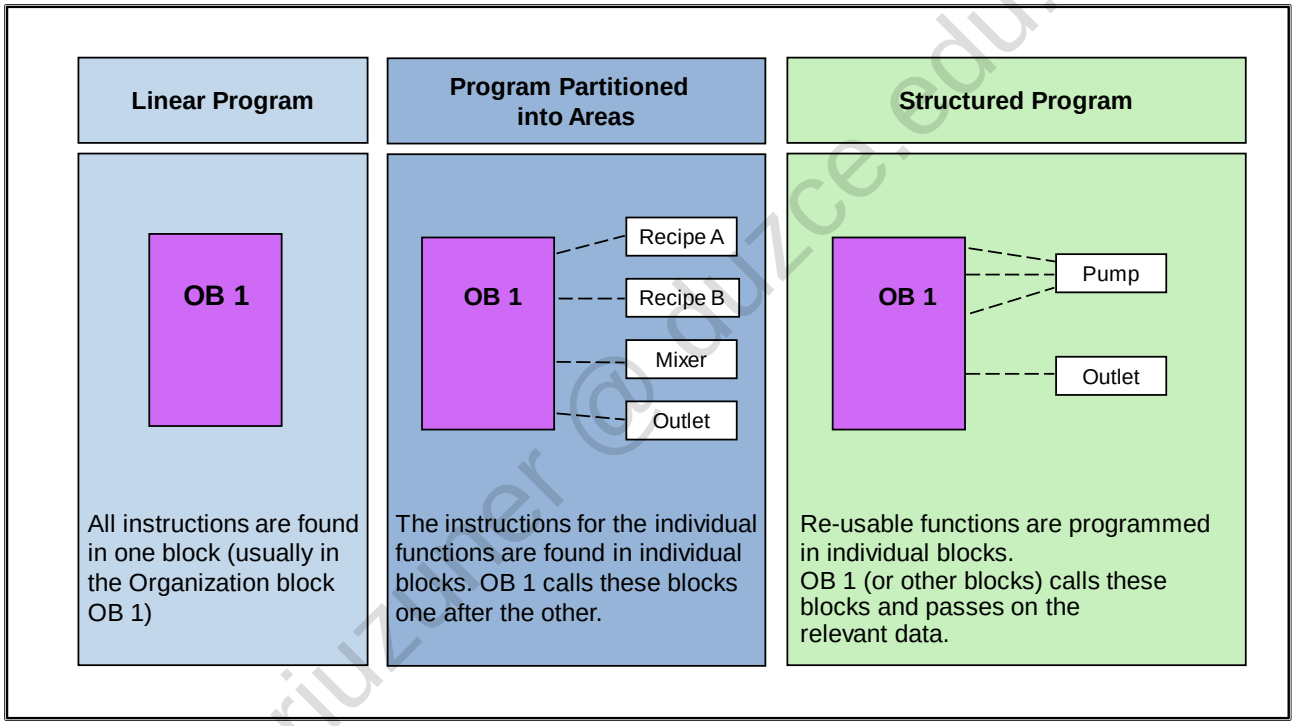
Functions (FCs)

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 such as closed-loop control tasks.

6.3. Structured Programming



Linear Program

The entire program is found in one continuous program block. 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.

Partitioned Program

The program is divided into blocks, whereby every block only contains the program for solving a partial task. Further partitioning through networks is possible within a block. You can generate network templates for networks of the same type. Normally, the OB 1 organization block contains instructions that call the other blocks in a defined sequence.

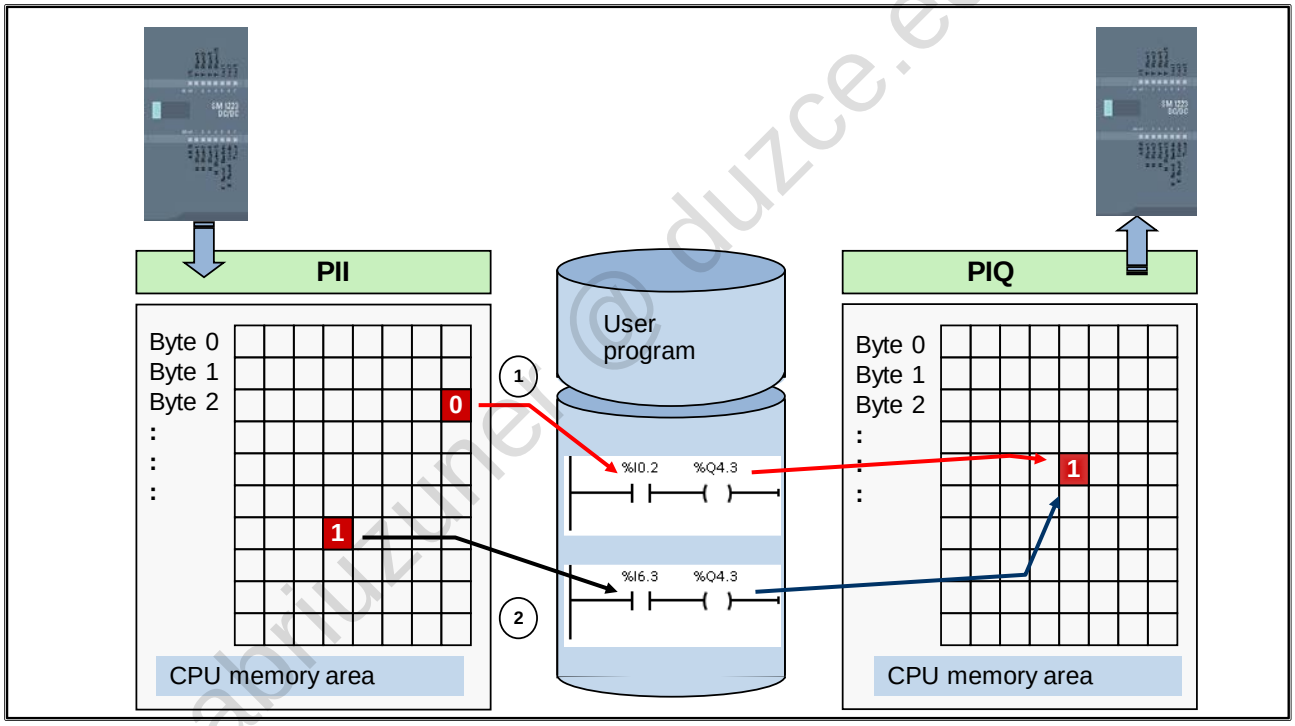
Structured Program

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 actual 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.

6.4. Process Images



Process Images

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

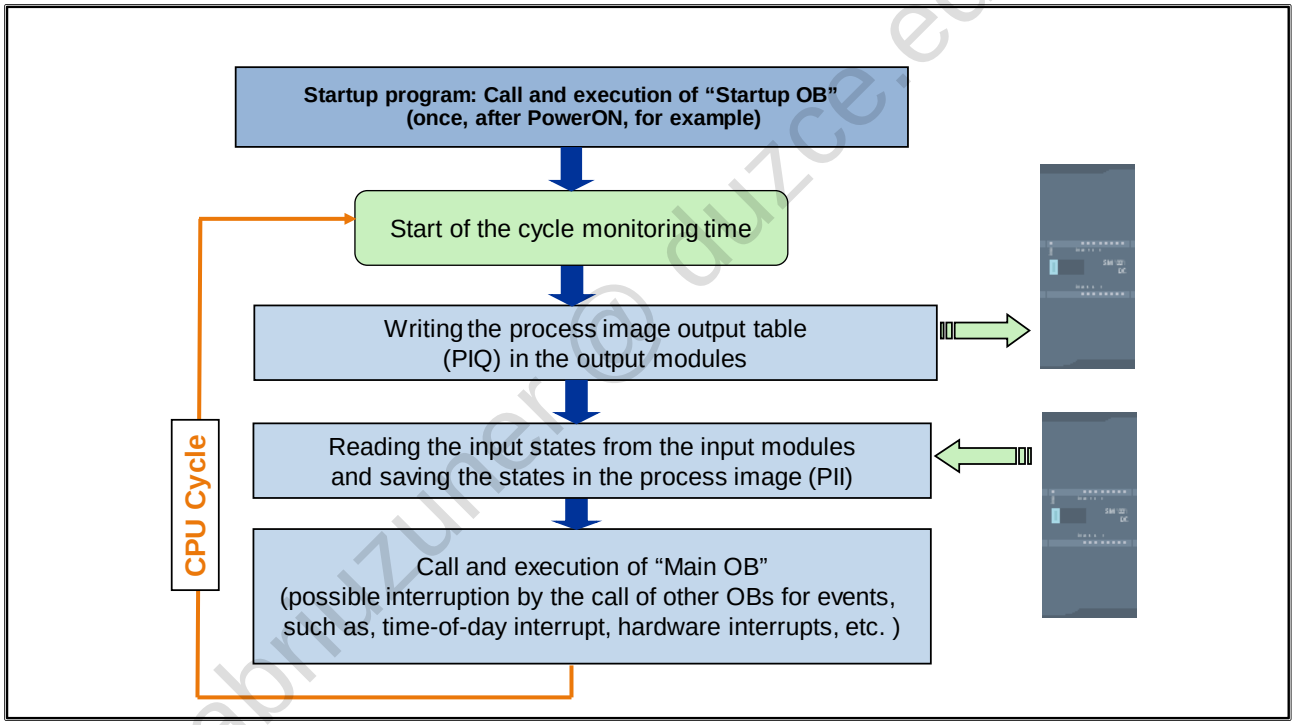
Process Image of Outputs (PIQ)

The Process-Image Output table (PIQ) is the memory area in which the states of all digital outputs are stored. The PIQ is output to the digital output modules at the beginning of the cycle. Outputs can be assigned as well as queried in the program. If an output is assigned a state in several locations in the program, then only the state that was assigned last is transferred to the output module. As a rule, these types of double assignments are programming errors.

Process Image of Inputs (PII)

The Process-Image Input table (PII) is the memory area in which the states of all digital inputs are stored. After the PIQ is output, the PII is read by the digital input modules. When an input is linked, the state of this input stored in the PII is linked. This state 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 queries of the input in one cycle, the same result is always supplied.

6.5. Cyclic Program Execution



Restart

When you switch on or switch from STOP --> RUN, the CPU carries out a complete restart (execution of all Startup-OBs). During restart, the operating system deletes the non-retentive bit memories and resets all stored hardware and diagnostic interrupts.

Cyclic Program Execution

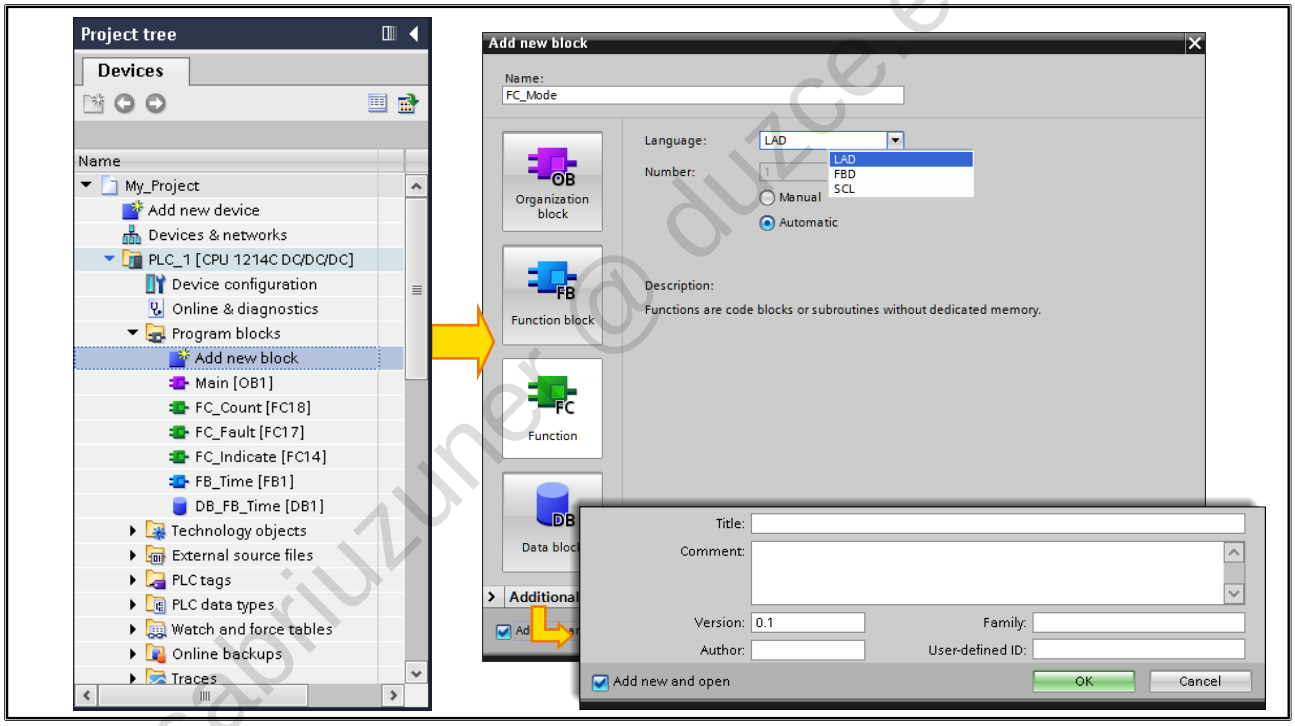
Cyclic program execution occurs in an endless loop. After the execution of a program cycle is completed, the execution of the next cycle occurs automatically. In every program cycle, the CPU carries out the following steps:

- The CPU starts the cycle monitoring time
- The CPU transfers the output states from the process image output table to the digital output modules
- The CPU scans the states of the input signals and updates the process image input table
- The CPU sequentially processes the instructions of the user program using the process images, not the inputs and outputs of the digital input / output modules

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, the system requests the call of the "Time error OB". If it is loaded, it is executed. If the CPU exceeds twice the cycle monitoring time, the CPU goes into the STOP state.

6.6. Adding a New Block

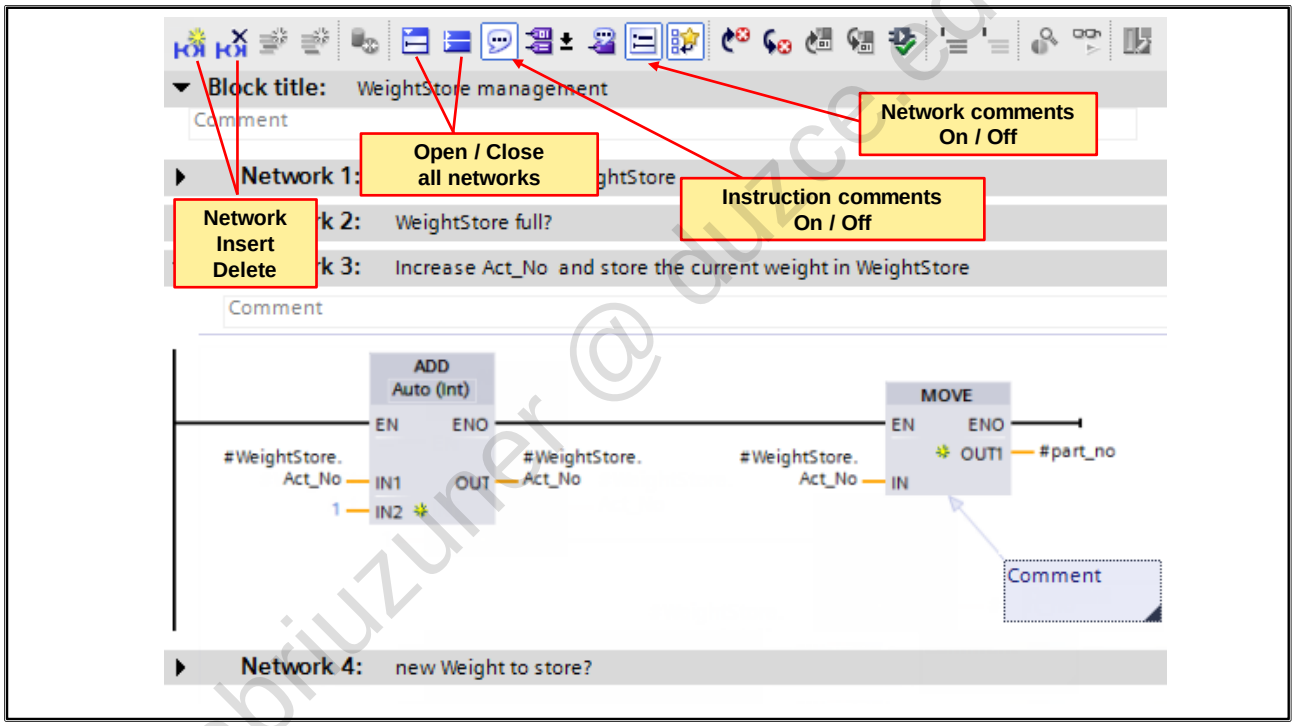


Inserting a Block

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.

6.6.1. Block Networks

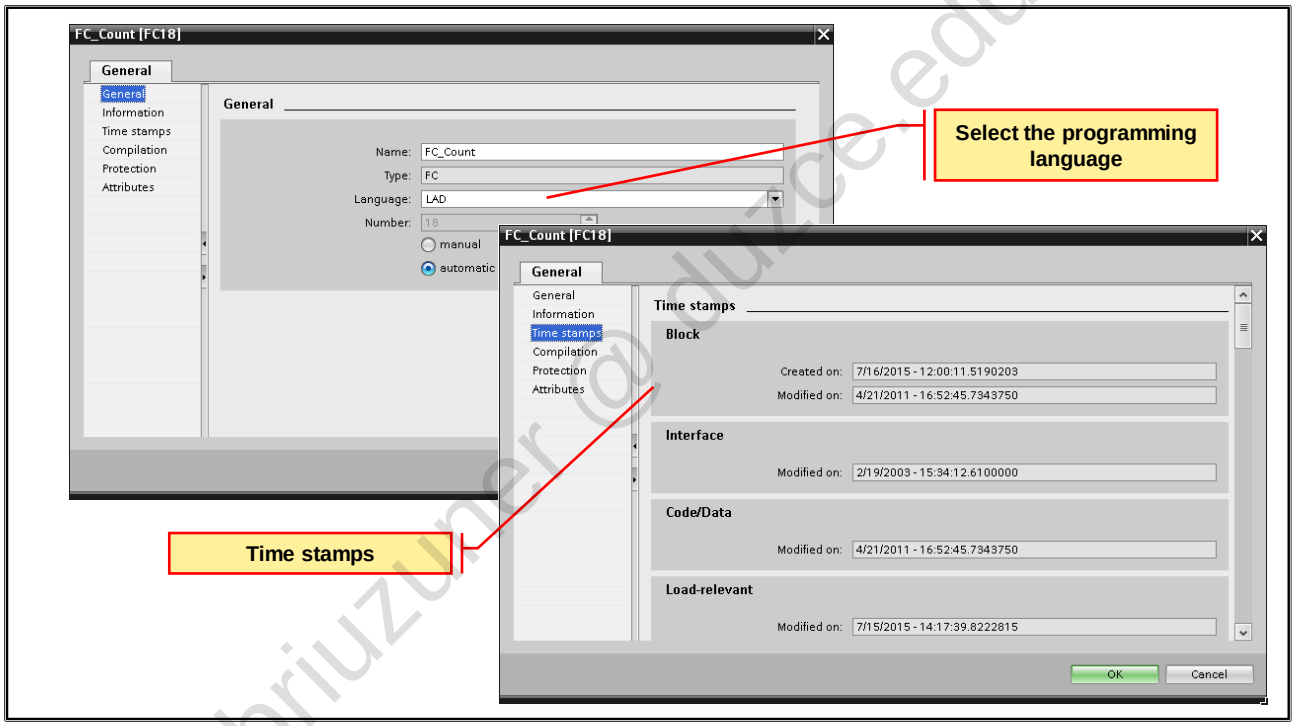


Networks

Just as the entire user program is subdivided into individual blocks, individual blocks are made up of networks designed by the user.

Each network can be given a network label and a comment. Within the networks, the individual instructions can be given instruction comments.

6.6.2. Block Properties: Programming Language, Time Stamps

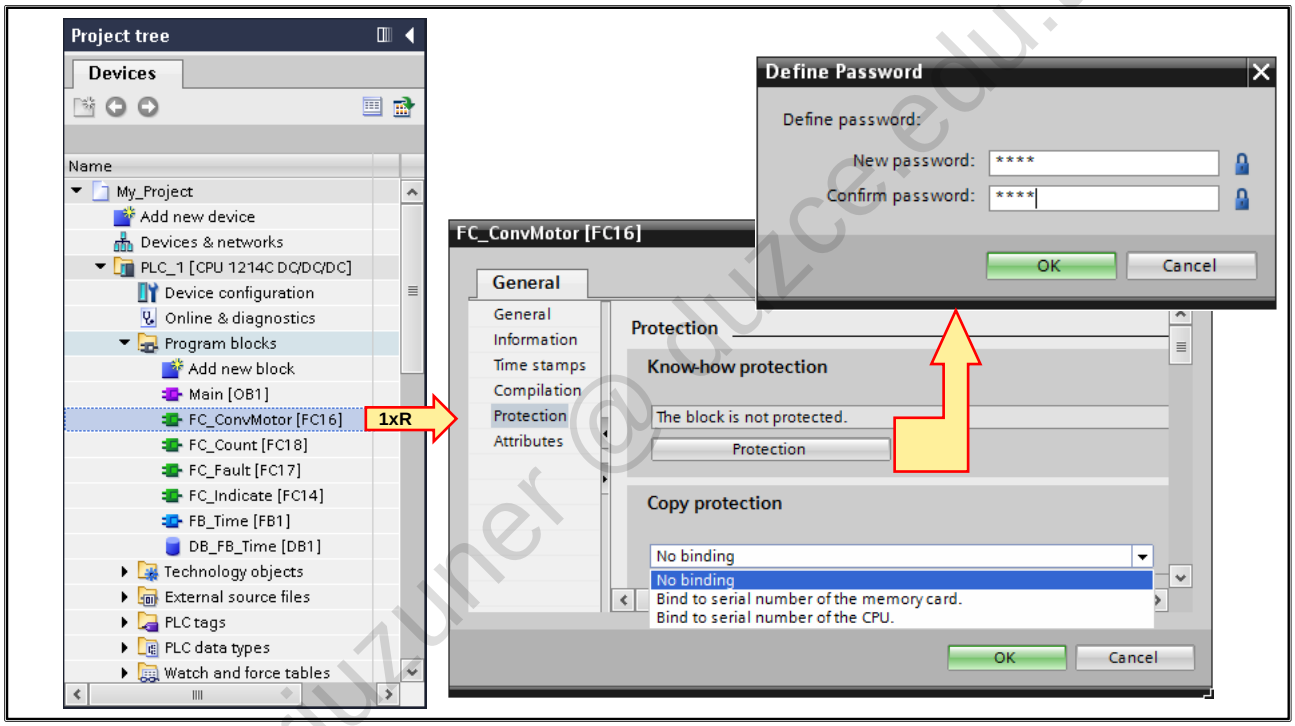


Properties

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

6.6.3. Block Properties: Know-how 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:

- Transfer parameters Input, Output, InOut, Return
- Block title
- Block comment
- Block properties
- Cross references with the information 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 against unauthorized reading and changing.

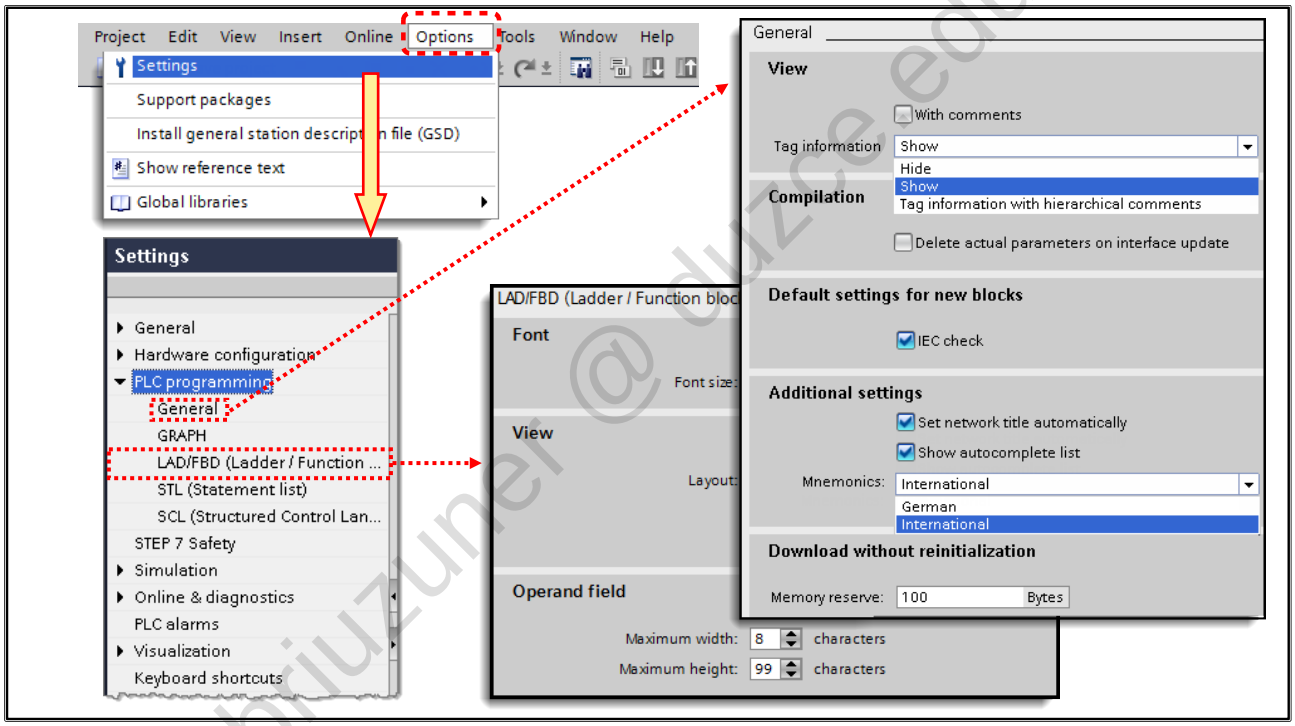
Note

When downloading a know-how protected block into the CPU, the know-how protection is also loaded in the CPU, but not the retrieval information. This results in the following consequence: if a know-how protected block is uploaded from the CPU into an offline project, this block can no longer be opened offline even when the correct password is given. A block comparison between the offline and the online version of the block is, however, still possible with the correct password.

Caution!

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

6.6.4. Block Editor Settings

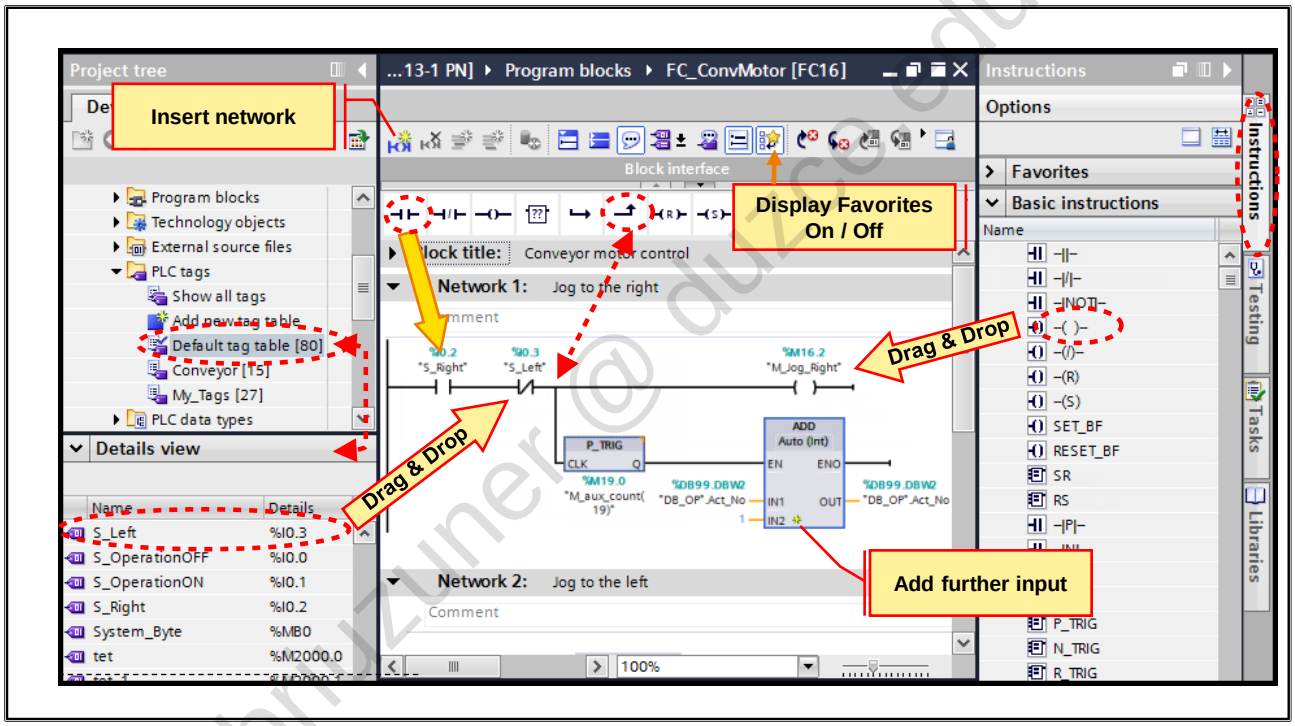


Block Editor Settings

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

- Compilation
  When "Delete actual parameters on interface update" is activated, the calls of parameterized blocks are automatically adjusted if, within the block, parameters are deleted after the fact.
- IEC check
  Only variables of a correct data type can be used. If an operation requires a variable of the data type INT, no variable of the data type WORD can be used even if the dimension (16 bits) is the same.
- Optimized block access
  Data block variables and local variables within blocks can only be addressed symbolically and not absolutely. Benefit: optimum memory allocation and shorter access times
- Mnemonics
  Setting the syntax for the programming language: German (e.g. E for Eingang (Input)) or International (e.g. I for Input)
- Layout
  When "With absolute information" is activated the absolute addresses of global operands are also displayed.
- Operand field
  Setting the maximum width and height of function block diagram and ladder diagram symbols.

6.6.5. Block Programming



Block Programming

The instructions within a block can be programmed as follows:

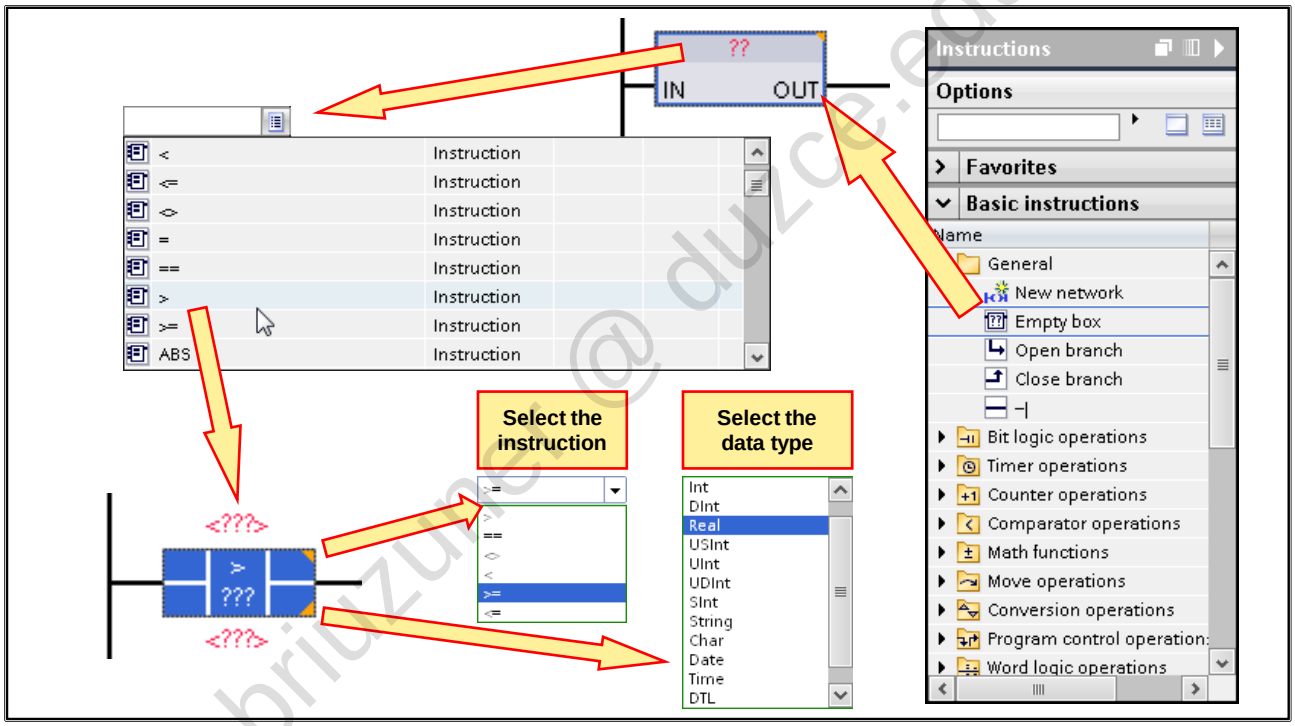
- using drag & drop from the Favorites or the Instructions catalog to anywhere in the program
- by first selecting the location in the program and then double-clicking on the desired instruction in the Favorites or the Instructions catalog

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 using drag & drop to the appropriate location in the program.

Favorites

Frequently used LAD (FBD) elements are available in the symbol bar which can be expanded individually using drag & drop from the Instructions catalog.

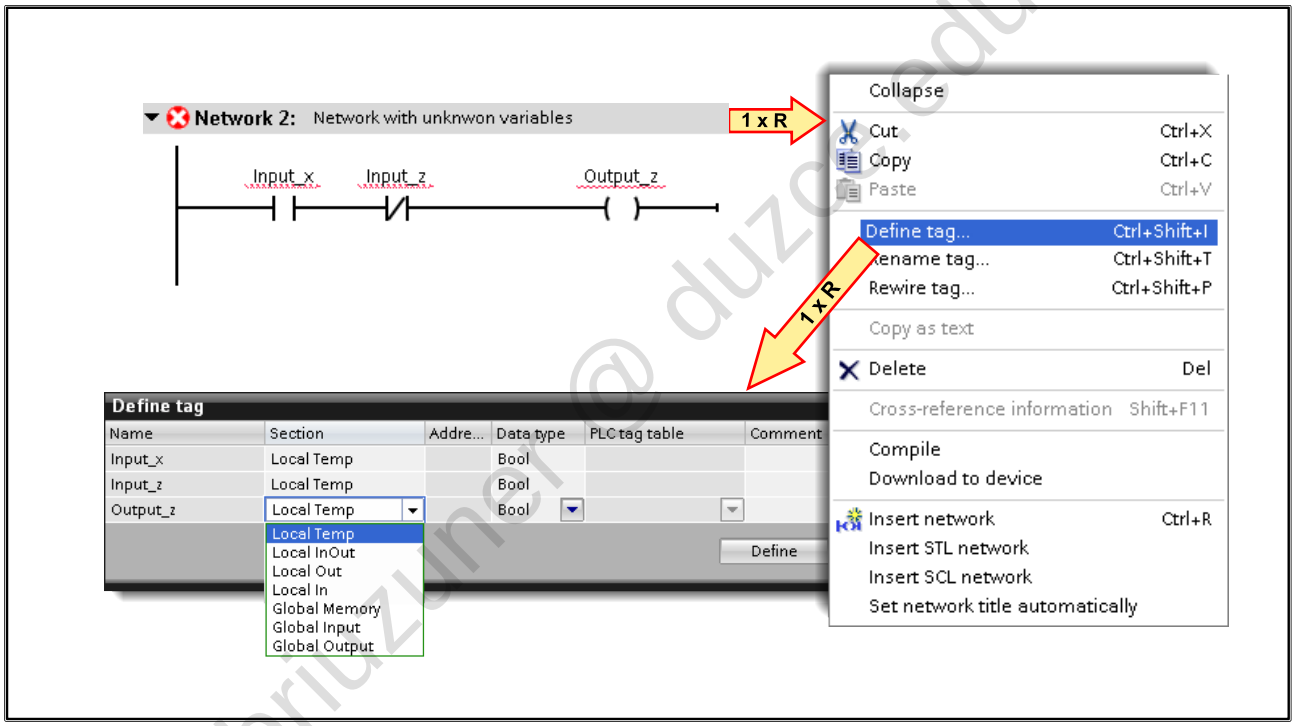
6.6.6. Programming an Instruction using "Empty Box"



Programming an Instruction using Empty Box

An instruction can also be programmed in the so-called "Empty box". First, the empty box is pulled into the network from the "Instructions" task card using drag & drop and then a selection is made on the empty box as to which instruction is to be used with which data type.

6.6.7. Defining (Declaring) Tags while Programming

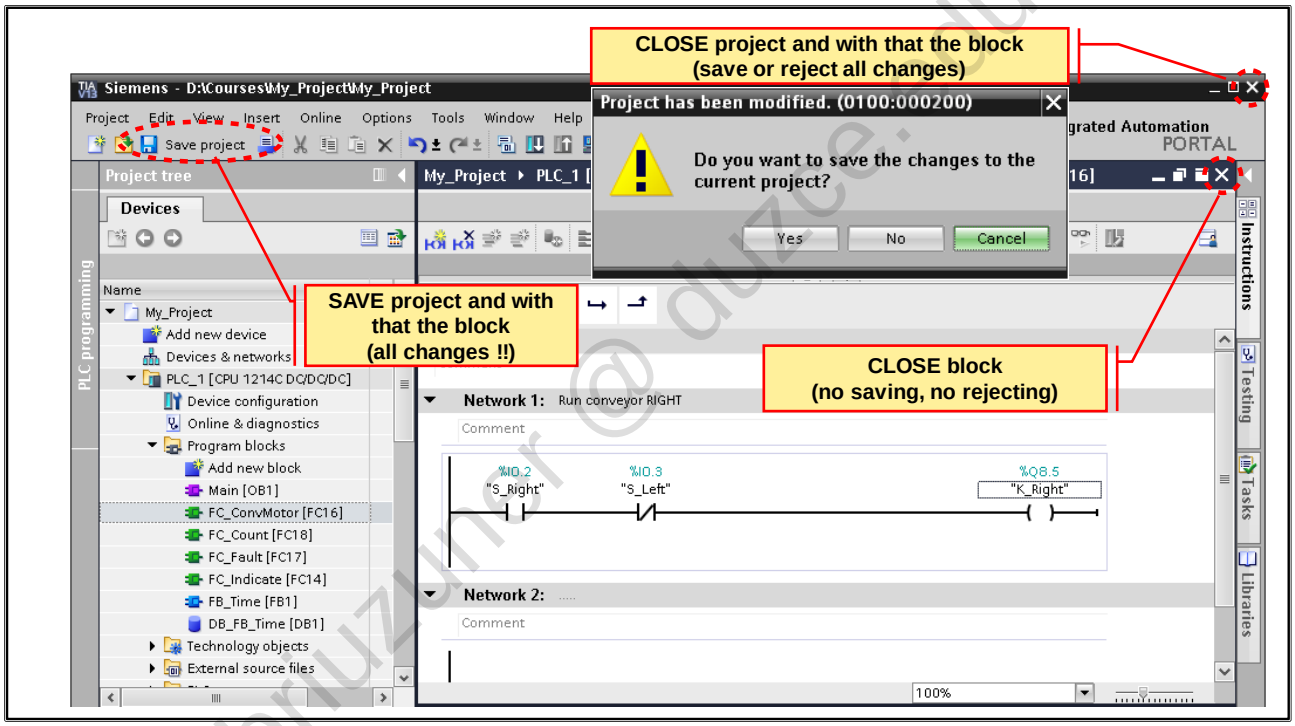


Defining (Declaring) Tags while Programming

If, during programming, unknown tags are used, they can be defined later-on network-by-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.

6.6.8. Closing / Saving / Rejecting a Block



Closing a Block

Clicking on the  symbol in the title bar, closes a block but changes are not rejected or saved on the hard drive!

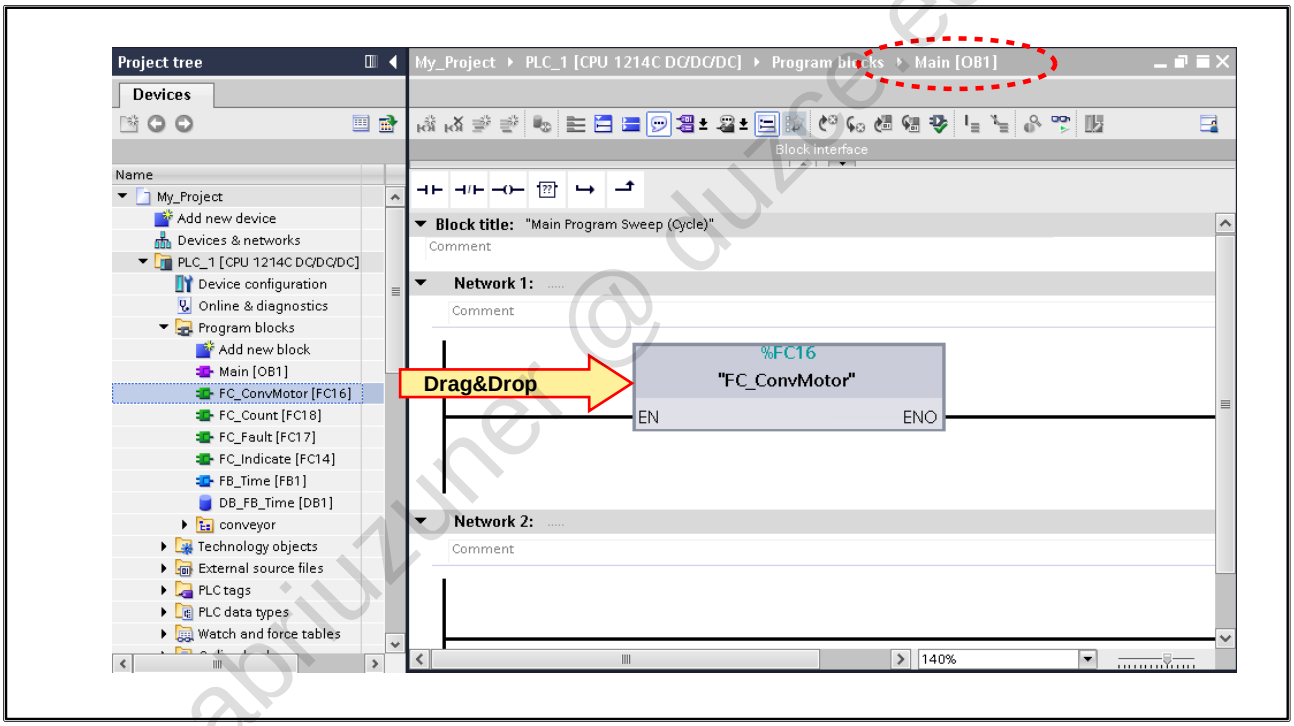
Saving a Block

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 only possible to reject block changes by closing the entire project without saving. All changes made in the project are rejected.

6.6.9. Block Calls

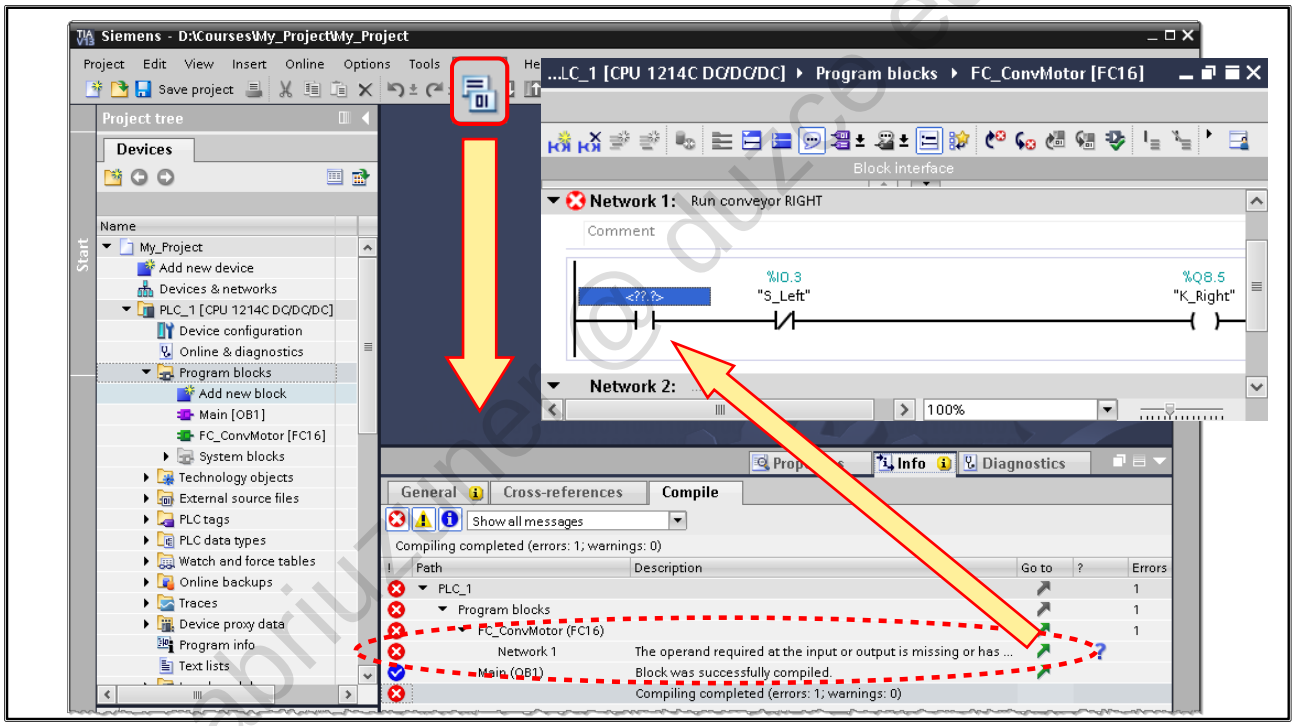


Block Calls

If one 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.

The block call can be programmed using drag & drop or copy & paste.

6.6.10. Compiling a Block



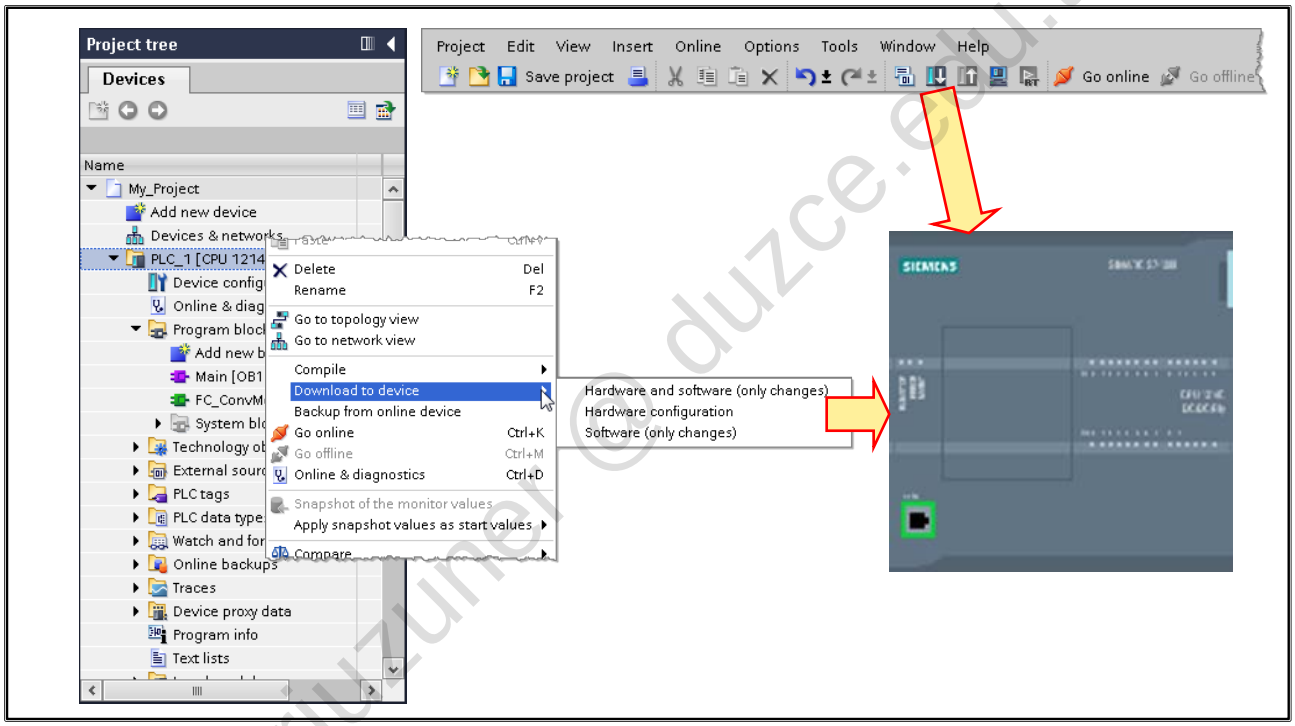
Compiling a Block

With the Compile icon, all changes of whatever is selected (highlighted) in the Project tree are compiled (in the example shown, all changes of the entire program are compiled). The changes of individual blocks (select relevant block), the changes of the entire program or the changes of the entire station with software and hardware ("Station" is selected) can be compiled.

To completely compile the blocks or the station, the context menu (right click) of the folder "Program blocks" or the station is opened and the function "Software (rebuild all blocks)" or the function "Hardware (rebuild all)" is selected in the menu "Compile".

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

6.6.11. Downloading Blocks into the CPU



Downloading into the CPU

The project data which is downloaded into the devices is divided into hardware and software project data.

Hardware project data results from configuring the hardware, networks, and connections. The first time you download the data using the icon "Download to device" the hardware project data is completely loaded. In subsequent downloads, only configuration changes are downloaded.

To once more download the entire configuration, you open the context menu of the station and select the function "Download to device > Hardware configuration".

Software project data involves the blocks of the user program. The first time you download, the software project data is completely loaded. In subsequent downloads, either by means of the icon "Download to device" or via the context menu, only changes are downloaded.

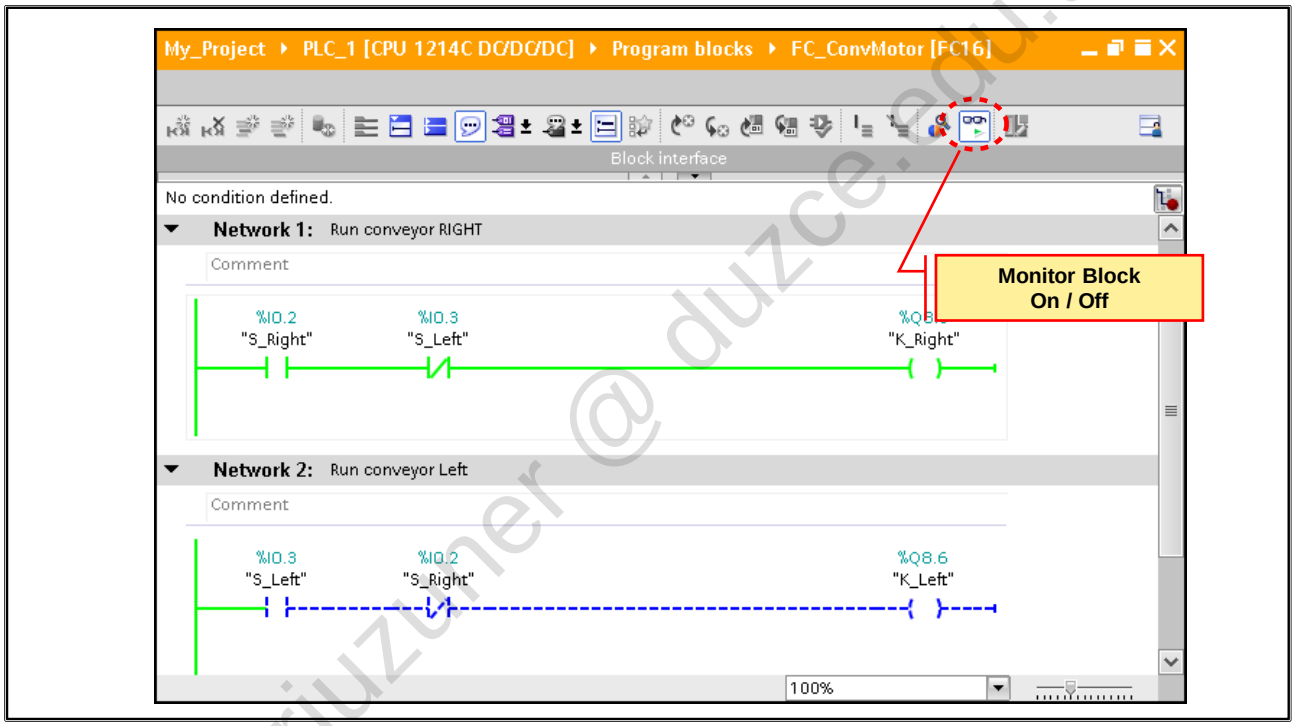
What is to be downloaded?
Selection via: Context menu of device > Download to device

- Hardware and Software (only changes): Download all new and modified software project data as well as the new and modified hardware configuration
- Hardware configuration: Download the entire hardware configuration
- Software (only changes): Download all new and modified software project data

☞ If the changes to the objects to be downloaded were not compiled before the loading, then the compilation is automatically carried out before the download.

☞ The download is only carried out if the compilation is error-free.

6.6.12. Monitoring a Block



Monitoring Blocks

The test function Monitor block is used to follow 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.

Monitoring

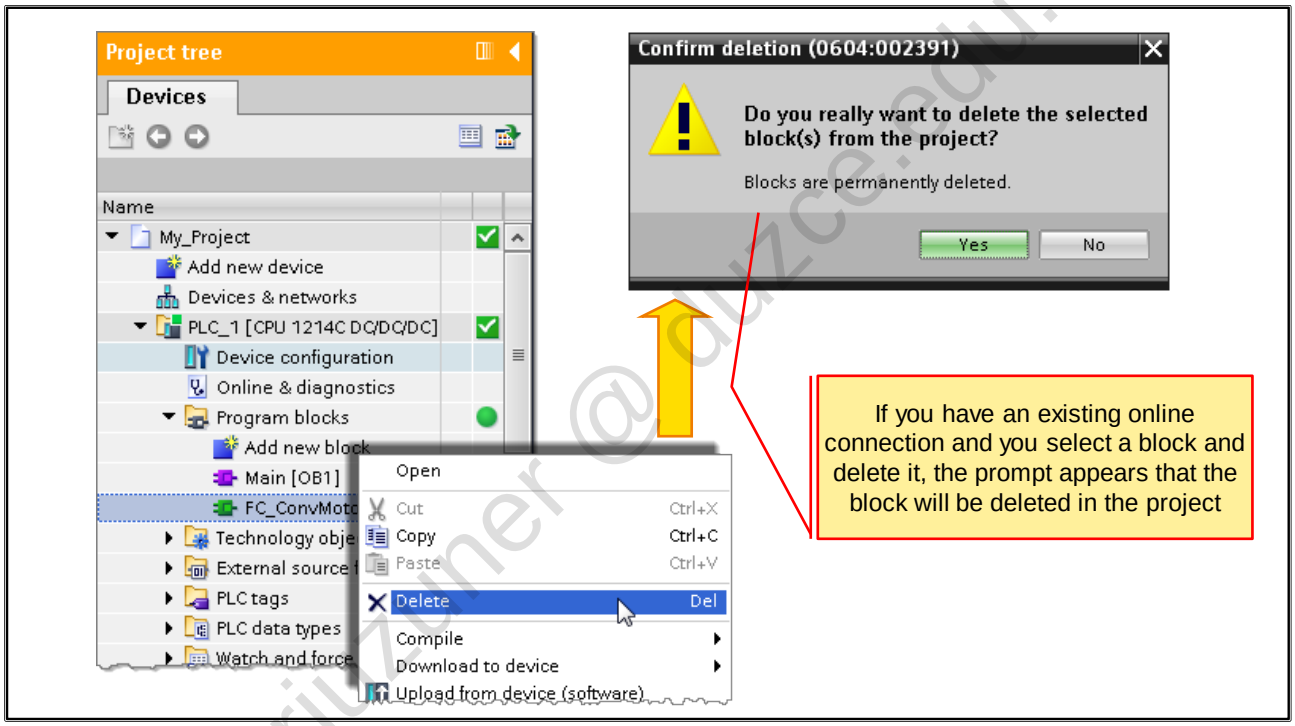
Blocks can only be monitored if an online connection to the CPU exists. Furthermore, the offline block must be identical to the online block. If the offline opened block does not match the block stored online in the CPU, either the online stored block must be opened, or the offline opened block must be downloaded into the CPU before you can monitor and then you can monitor the block.

In test mode, the statuses of operands and LAD / FBD elements are represented by different colors. You can make the settings for this via Options → Settings:

Examples:

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

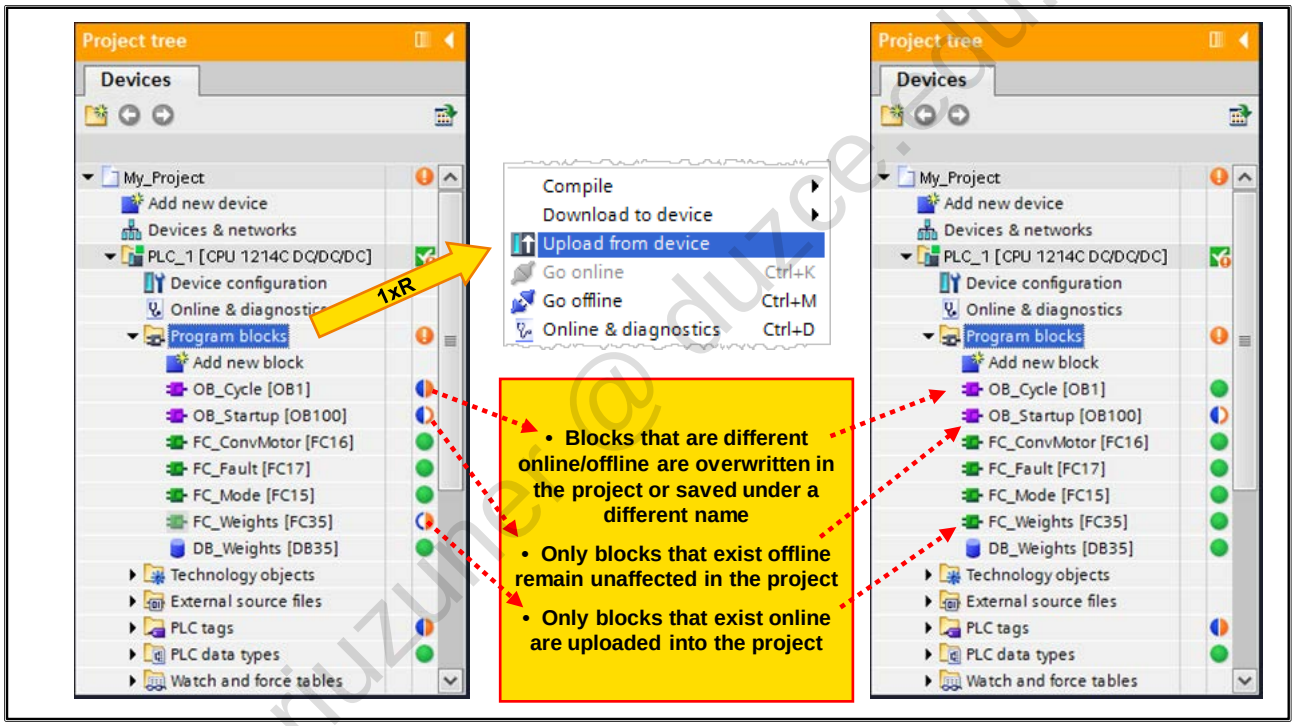
6.6.13. Deleting Blocks



Deleting Blocks

Blocks can only be deleted in the project. If you have an existing online connection and you select a block and delete it, the dialog shown in the picture appears.

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



Uploading Blocks into the Project:

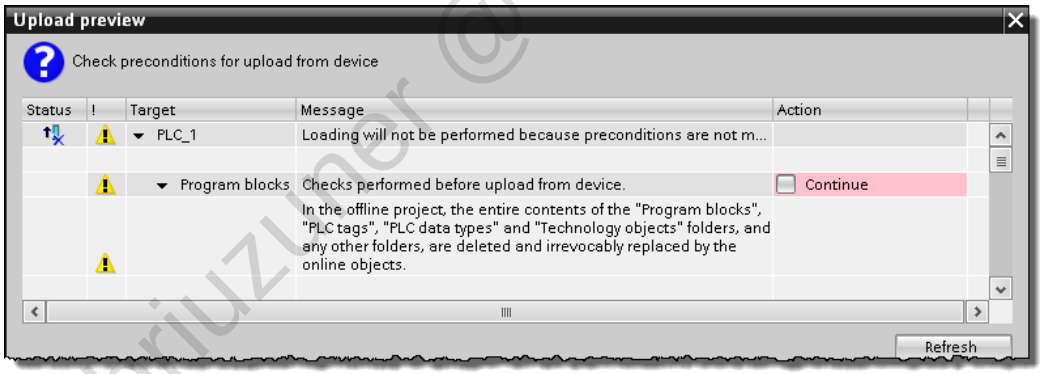
With "Upload from device", individual blocks or the entire CPU program including technology objects, PLC tag tables and PLC data types can be uploaded into the project from the CPU.

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

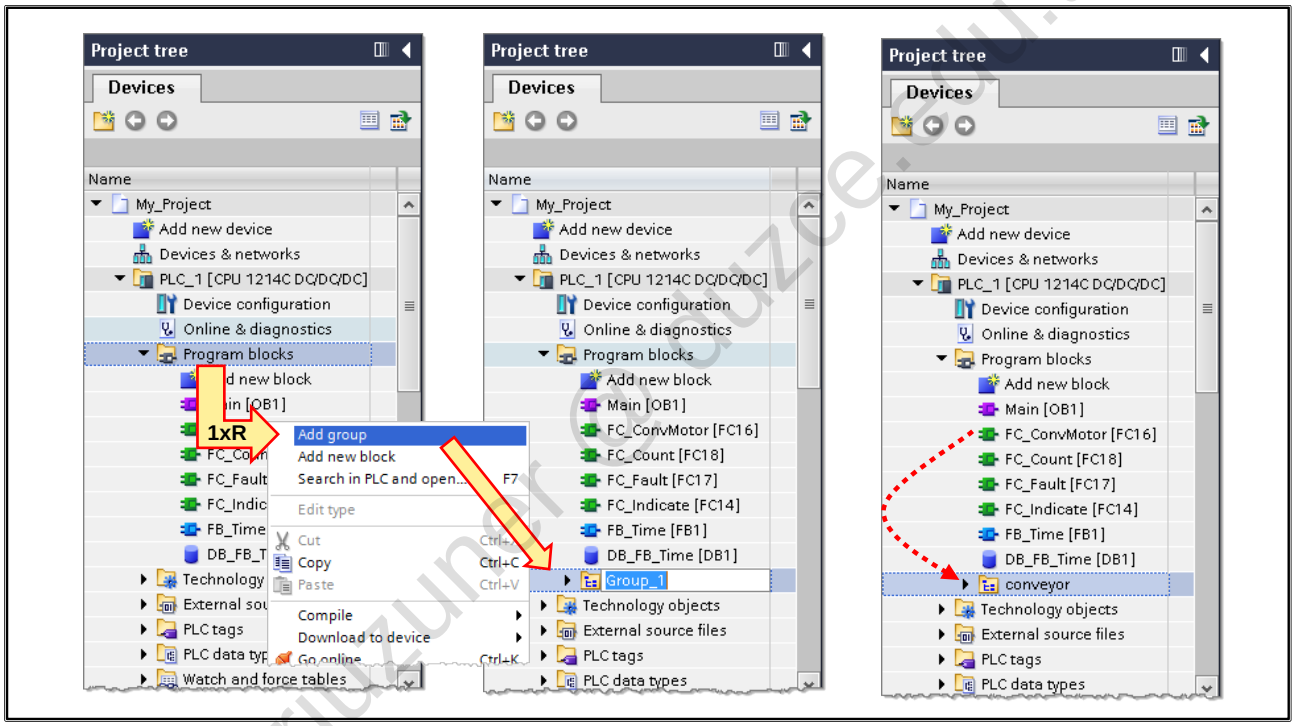
- ... blocks that only exist online in the CPU are uploaded into the offline project
- ... blocks that are different online / offline are either overwritten in the offline project with the uploaded blocks, or the uploaded blocks are additionally saved under a different name (but with the same number!) in the project. (selectable in the Upload dialog)
- ... blocks that only exist 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.



6.6.15. Block Groups

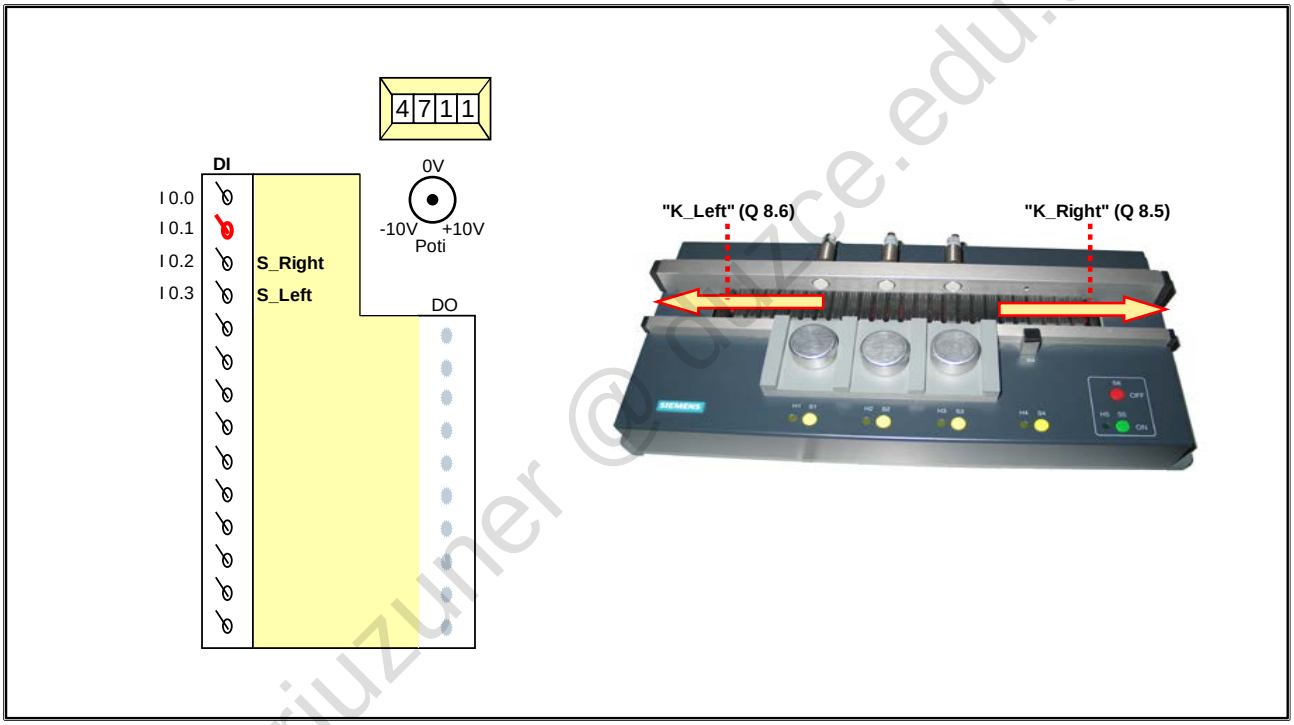


Block Groups

To achieve more clarity, large programs with many blocks can be divided into different 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.

Blocks can simply be shifted between the block groups using drag & drop.

6.7. Task Description: Jogging the Conveyor Motor

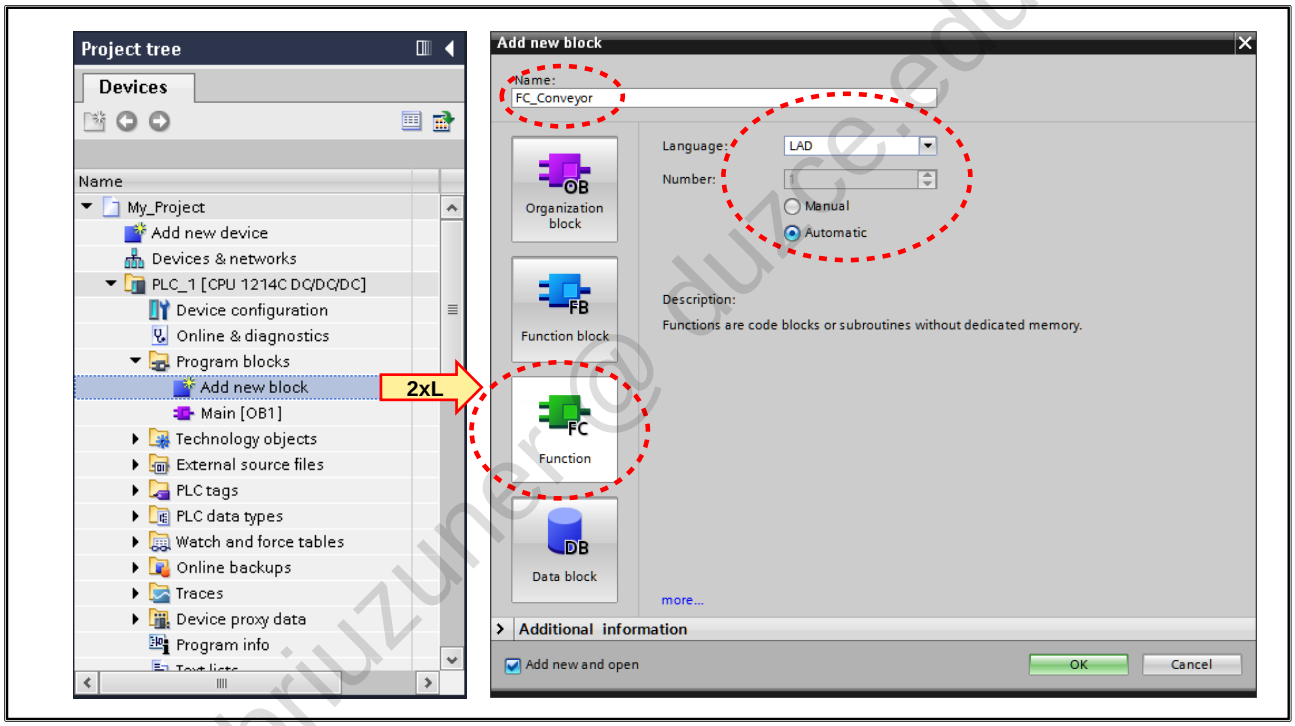


Task Description

Using the simulator switch "S_Right" (I 0.2), you should be able to jog the conveyor motor to the RIGHT "K_Right" (Q 8.5). Using the simulator switch "S_Left" (I 0.3), you should be able to jog the conveyor motor to the LEFT "K_Left" (Q8.6).

If both switches are pressed simultaneously, the conveyor motor must not start in either direction (Lock-out!).

6.7.1. Exercise 1: Adding the "FC_Conveyor" Block



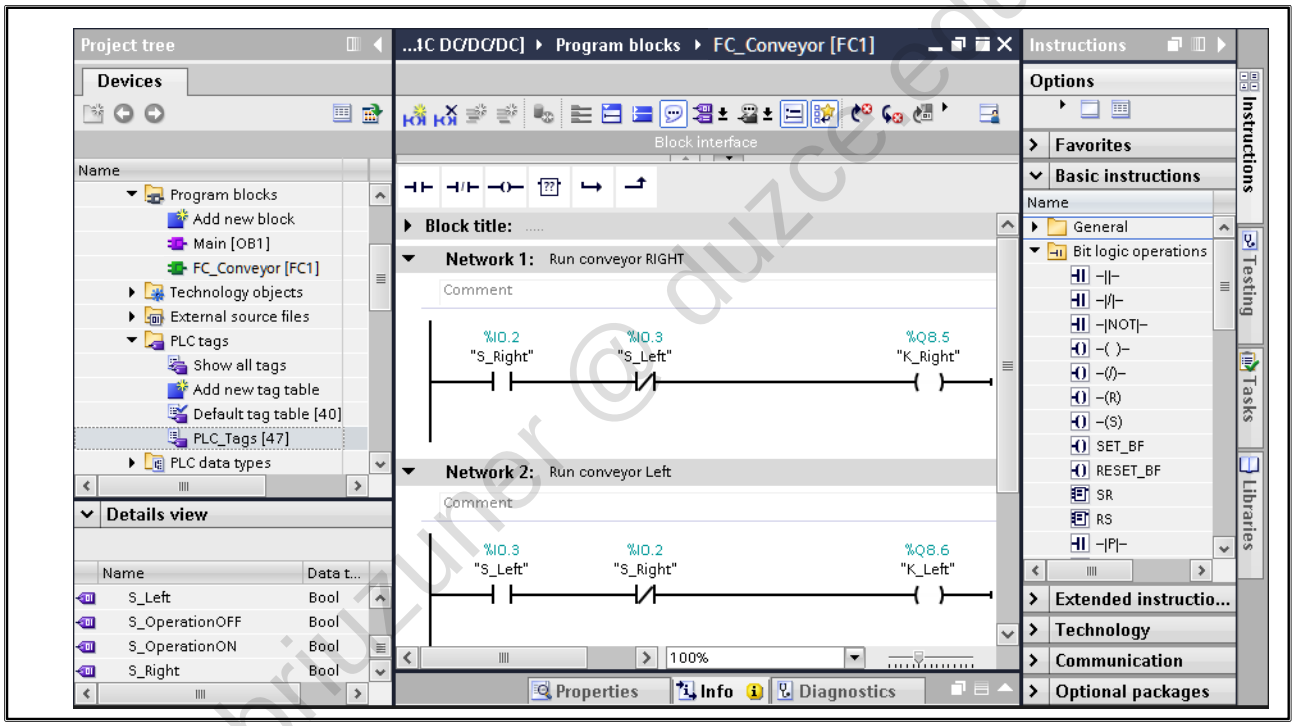
Task

Create "FC_Conveyor" as a new block in which you will then program the required jog operation of the conveyor motor 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

6.7.2. Exercise 2: Programming the "FC_Conveyor" Block



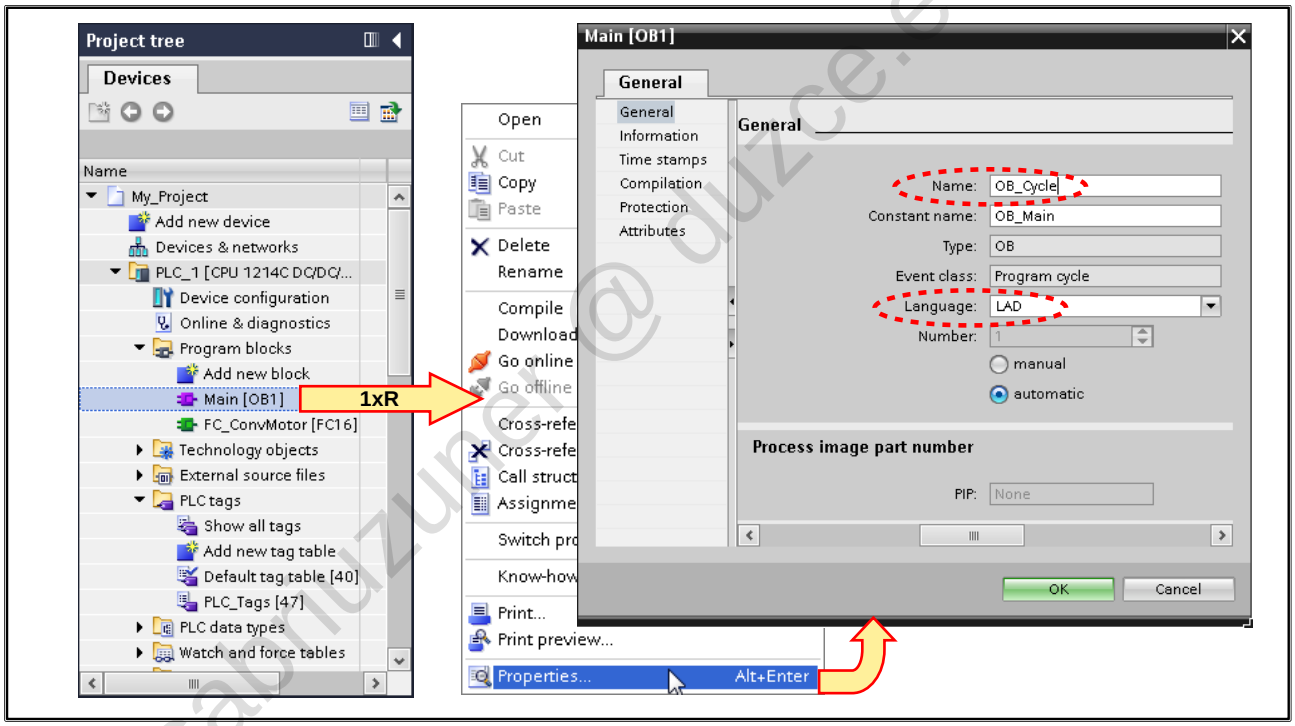
Task

Program the jog operation of the conveyor motor as shown in the picture.

What to Do

1. Program an AND logic operation by pulling it from the Favorites into the network using drag & drop
2. Program an instruction by pulling it from the "Instructions" task card to the AND logic operation using drag & drop
3. Above the instruction, enter the output "K_Right" (Q8.5) as the operand (you can enter the symbol as well as the absolute address)
4. In the Project tree, select (don't open) the "PLC_Tags" tag table and from the "Details view" drag the tag "S_Left" (I 0.3) as the operand to the AND logic operation. Do the same for "S_Right" (I 0.2)
5. Label both the block and the network
6. Add a new network and program a corresponding logic operation for jogging the conveyor motor to the left
7. Close the block
8. Save your project

6.7.3. Exercise 3: Adjusting the OB1 Properties

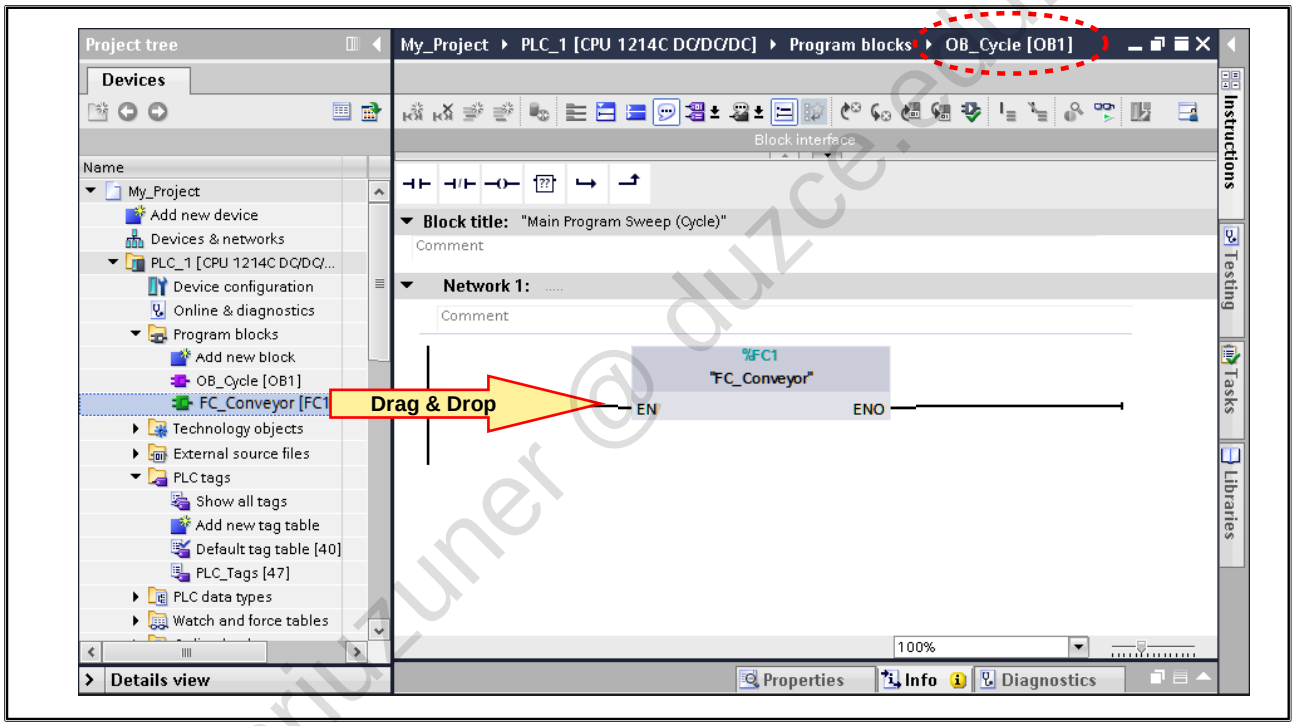


Task Description

The OB1 properties are to be adjusted as shown in the picture.

The symbolic name is to be changed and the programming language LAD is to be set. Proceed as shown in the picture.

6.7.4. Exercise 4: Calling "FC_Conveyor" in OB1



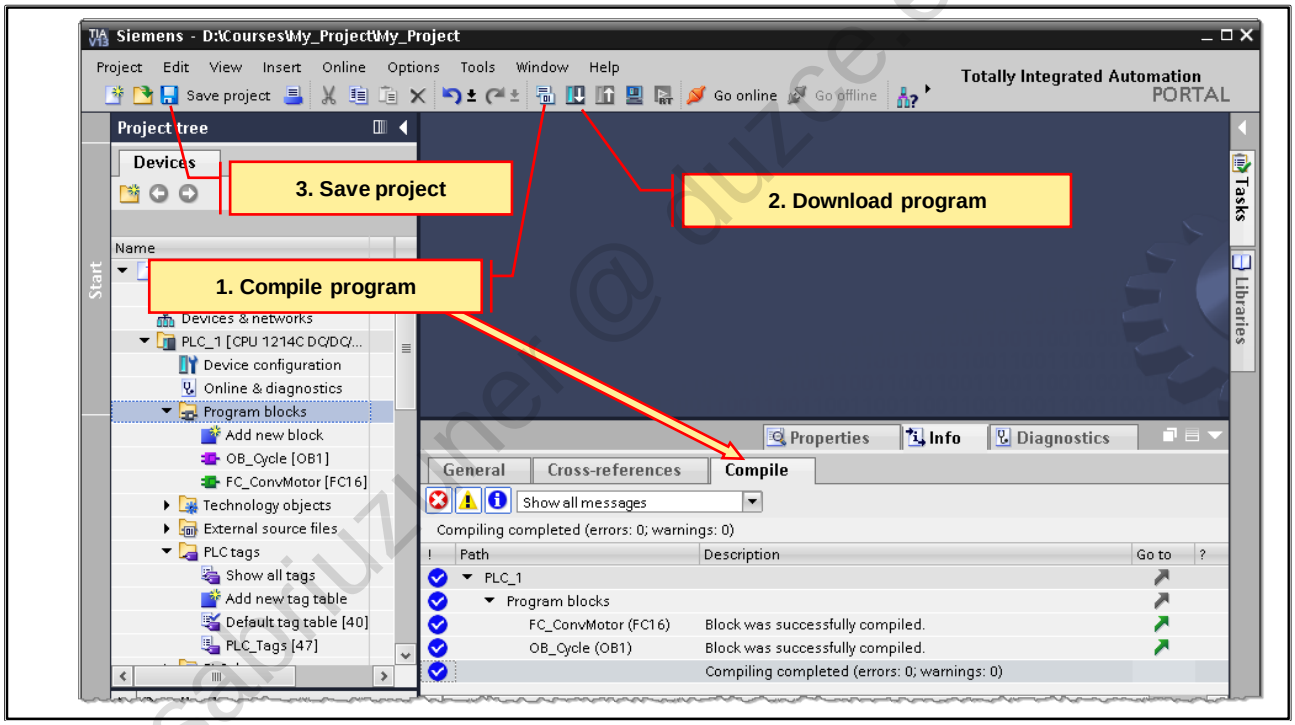
Task

So that the newly created "FC_Conveyor" block can be executed cyclically, its call must be programmed in OB1.

What to Do

1. Open the "OB_Cycle" (OB1) block by double-clicking on it
2. Program the call of "FC_Conveyor" as shown in the picture using drag & drop
3. Edit a block title and a label for Network 1
4. Close the block

6.7.5. Exercise 5: Compiling the Program, downloading it into the CPU and Saving it



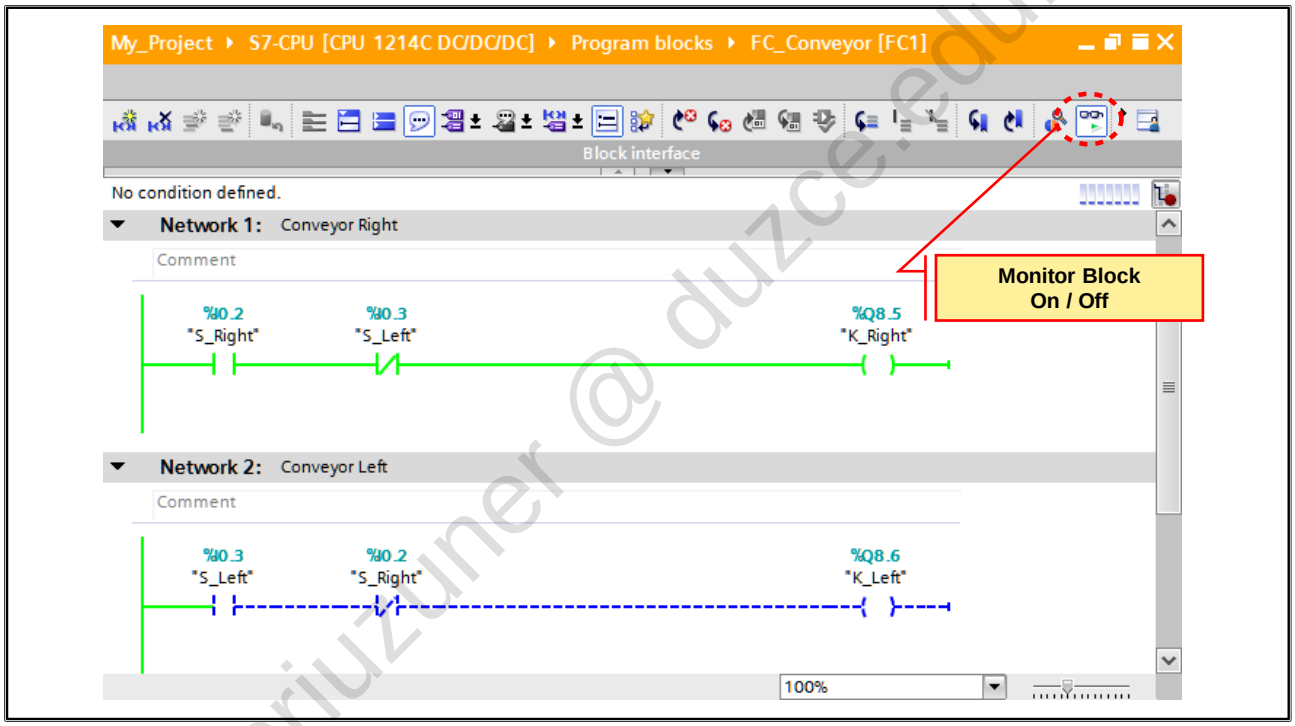
Task

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 (OB and FC) and to download it into the CPU, select the "Program blocks" container in the Project tree
2. Carry out the steps shown in the picture and check the program functioning by pressing the simulator switches "S_Right" (I 0.2) and "S_Left" (I 0.3)
3. Save your project

6.7.6. Exercise 6: Monitoring "FC_Conveyor"



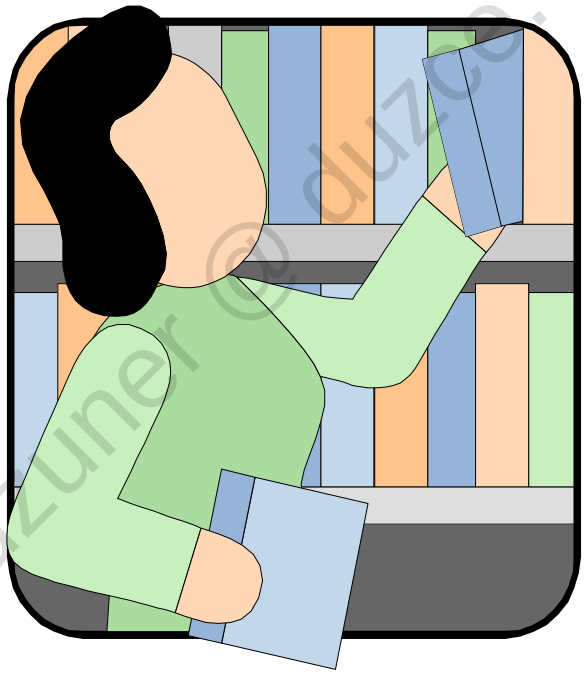
Task

You are to monitor the program execution of the newly programmed "FC_Conveyor" block. To do so, press the "S_Right" (I 0.2) and "S_Left" (I 0.3) simulator switches and interpret the statuses shown on the PG screen.

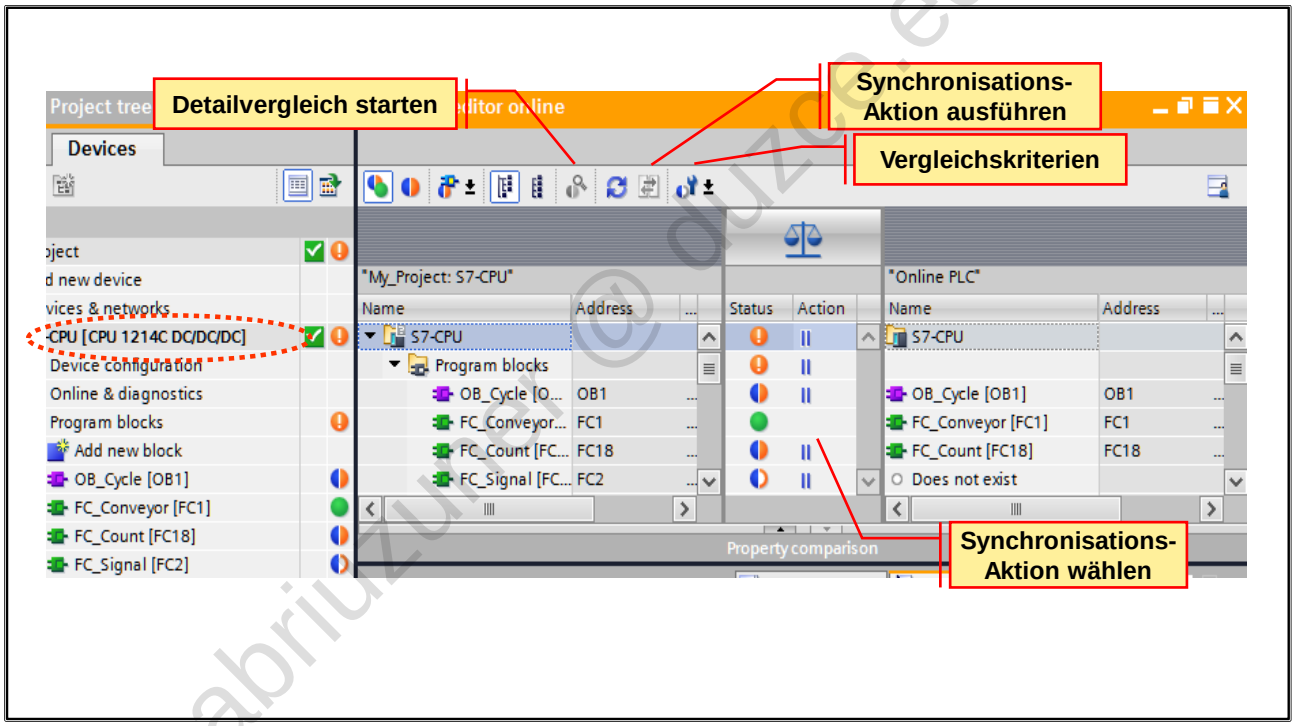
What to Do

1. In the Editor, open the "FC_Conveyor" block
2. Activate the test function "Monitor block" (see picture)
3. Alternately and also simultaneously press the "S_Right" (I 0.2) and "S_Left" (I 0.3) simulator switches and interpret the signal statuses of the operands shown

6.8. Additional information



6.8.1. Compare (1) - Blocks (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:

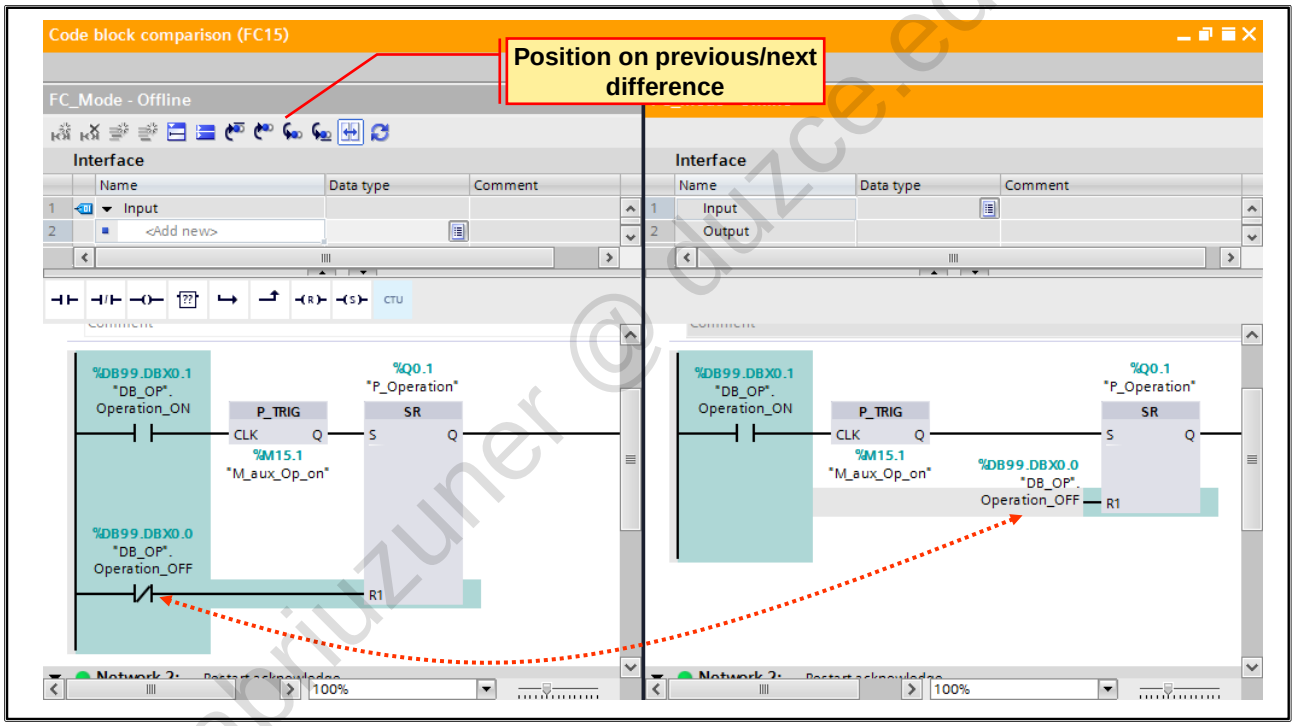
- Either the objects of two devices within a project or from different projects are compared.

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
	Online and offline versions of the object are identical
	Online and offline versions of the object are different
	Object only exists offline
	Object only exists online

6.8.2. Compare (2) – Block Detailed comparison

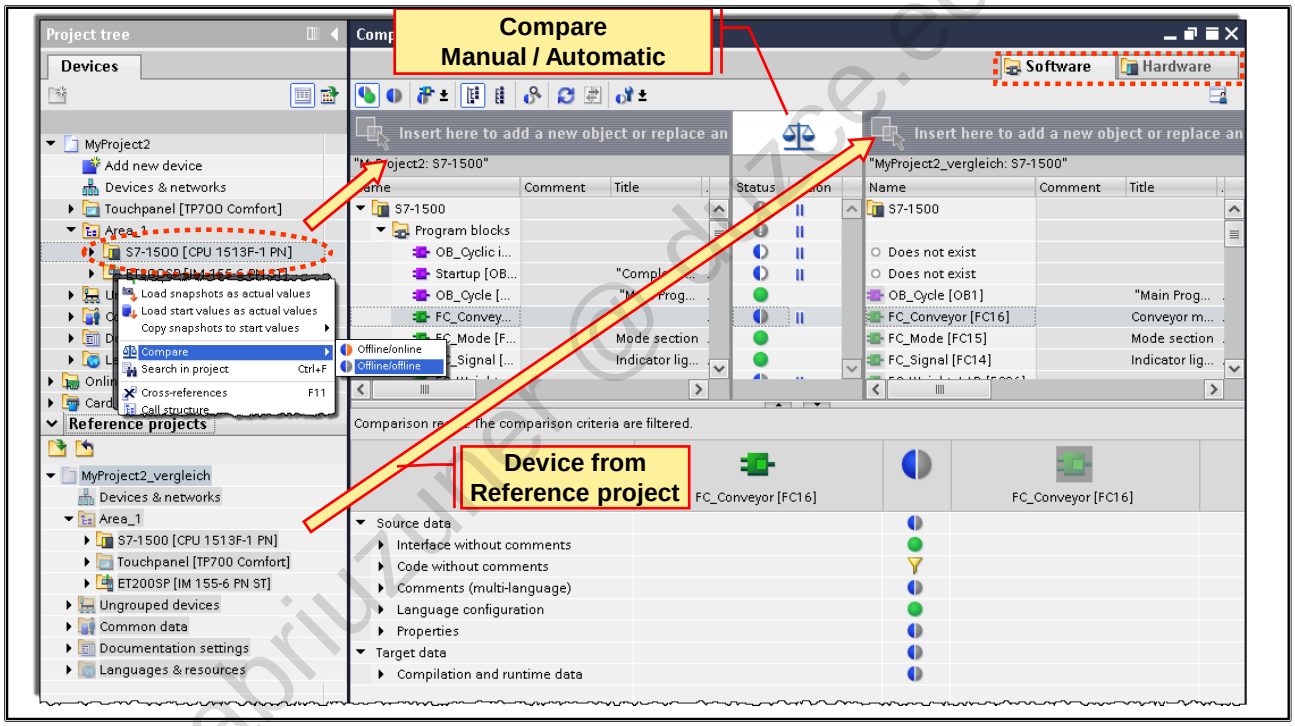


Detailed Comparison

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

6.8.3. 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

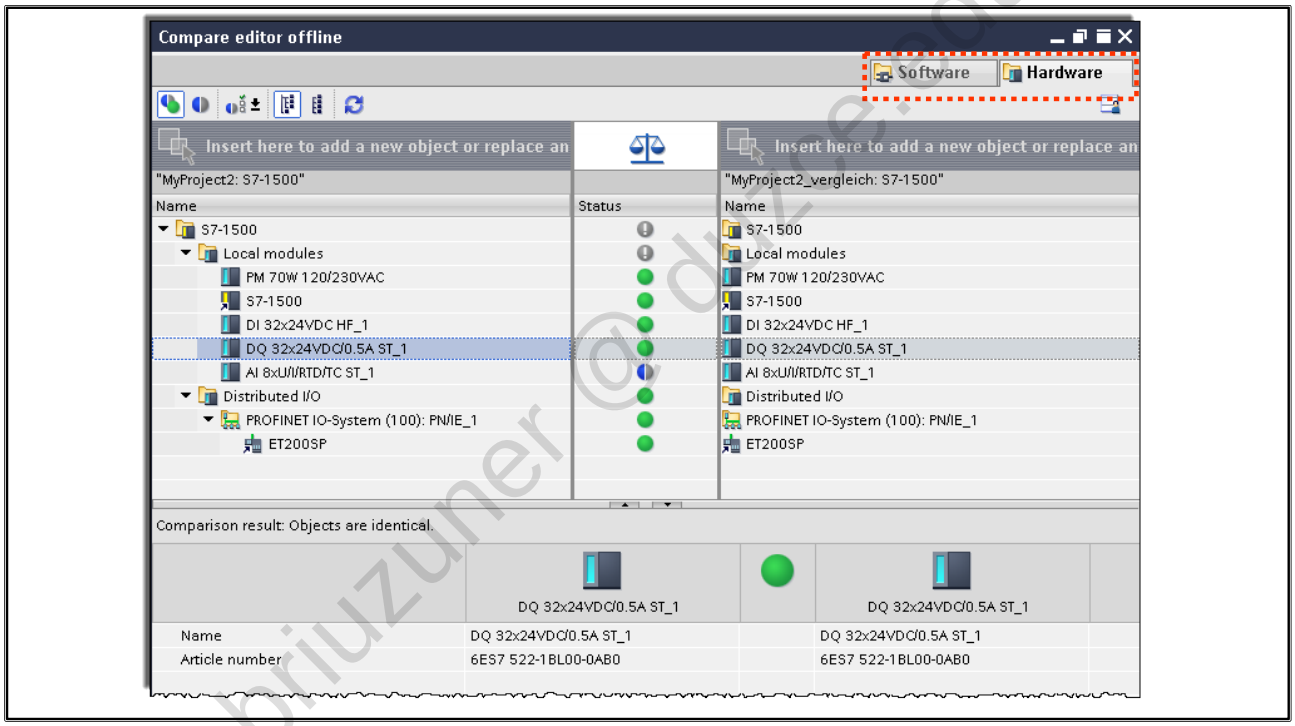
Automatic Comparison:

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.

6.8.4. 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.

Contents

7

7. Binary Operations

At the end of the chapter the participant will …

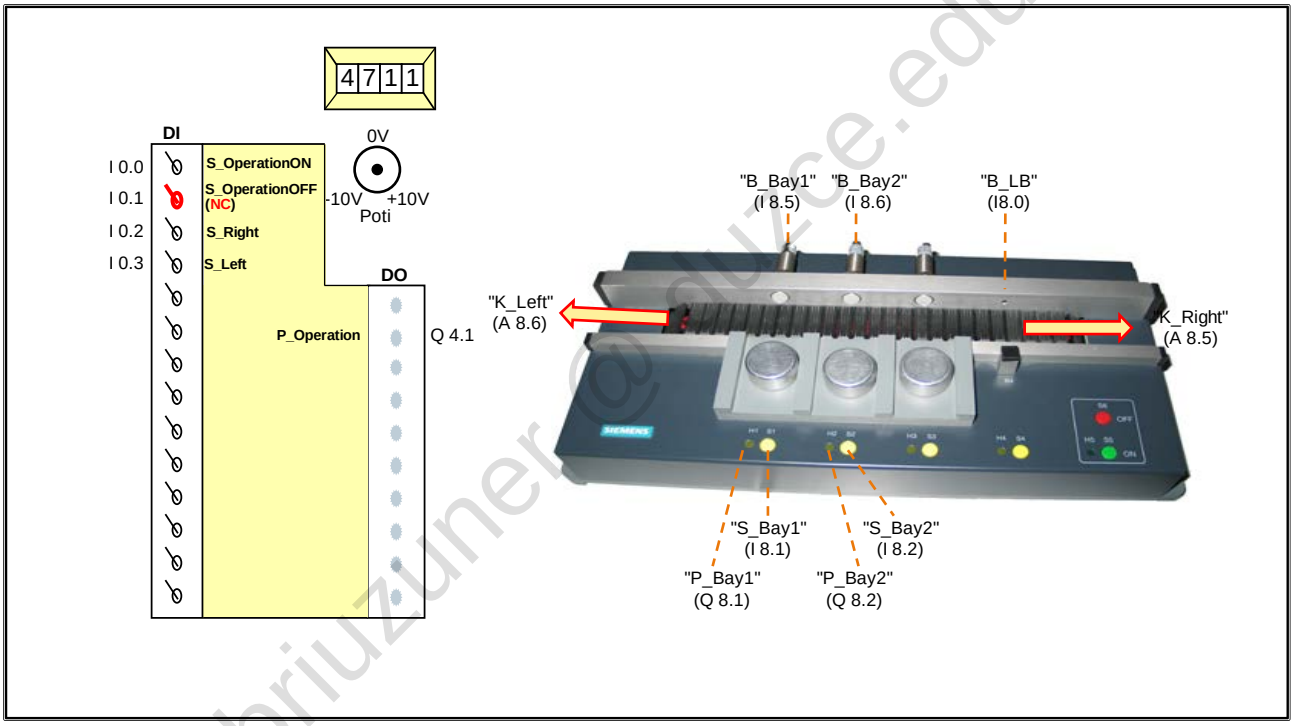
... be able to explain the difference between 'real' NC contacts and NO contacts connected in the hardware, and programmed check symbols

... be able to explain the terms Result of Logic Operation (RLO) and Status (STAT)

... be able to program basic binary logic operations

… be able to carry out simple troubleshooting

7.1. Task Description: The Conveyor Model as Distribution Conveyor



The Conveyor Model as Distribution Conveyor

The distribution conveyor is used to transport parts from Bay 1 or 2 to the light barrier bay. Operation (Simulator LED "P_Operation", Q4.1) can be switched on via the simulator switch "S_OperationON" (I 0.0) and switched off via the simulator switch "S_OperationOFF" (I 0.1, NC).

- When "P_Operation" (Q4.1) is switched off...

…the conveyor motor 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).

- When "P_Operation" (Q4.1.) is switched on...

...parts are transported from Bay 1 or 2 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...

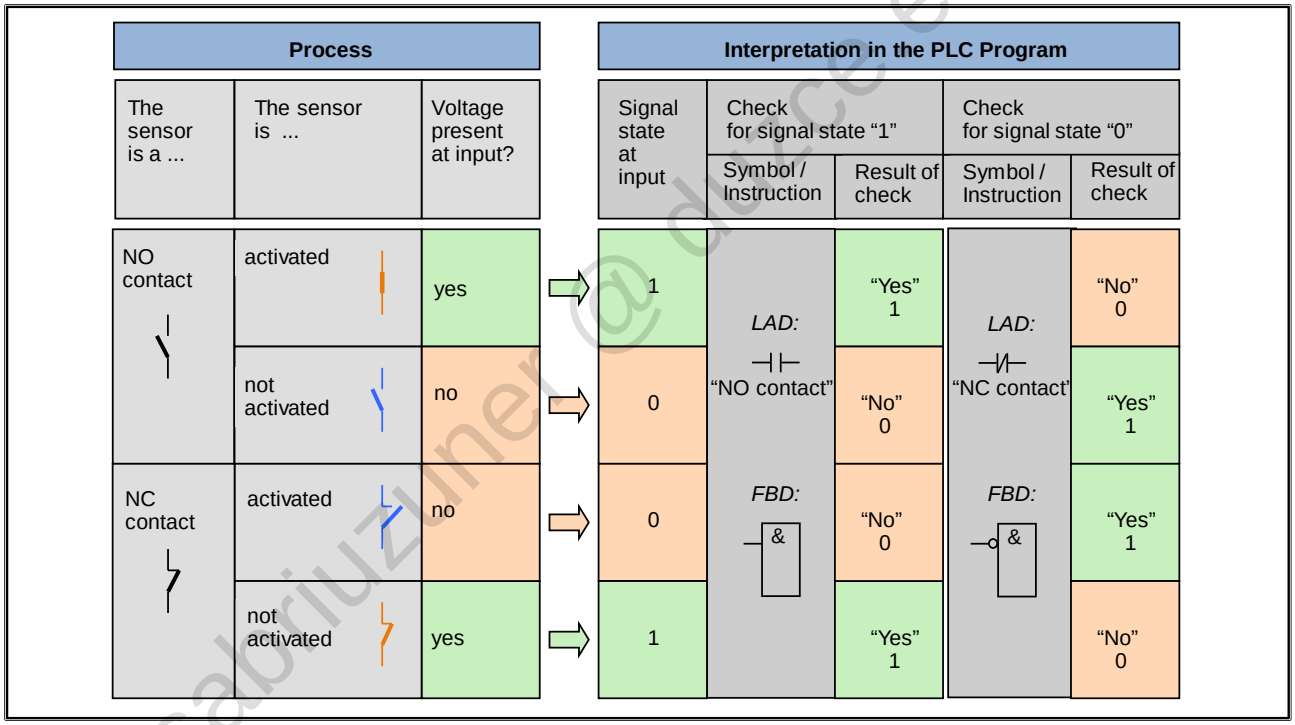
...a continuous light when a new part can be placed on the conveyor (conveyor motor is stopped and both proximity sensors are free)

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

...a 2Hz flashing light if the conveyor motor is running.

The indicator light at the light barrier bay shows a 2Hz flashing light if the conveyor motor is running and continuous light when the setpoint quantity has been reached.

7.2. Sensors and Check Symbols



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.

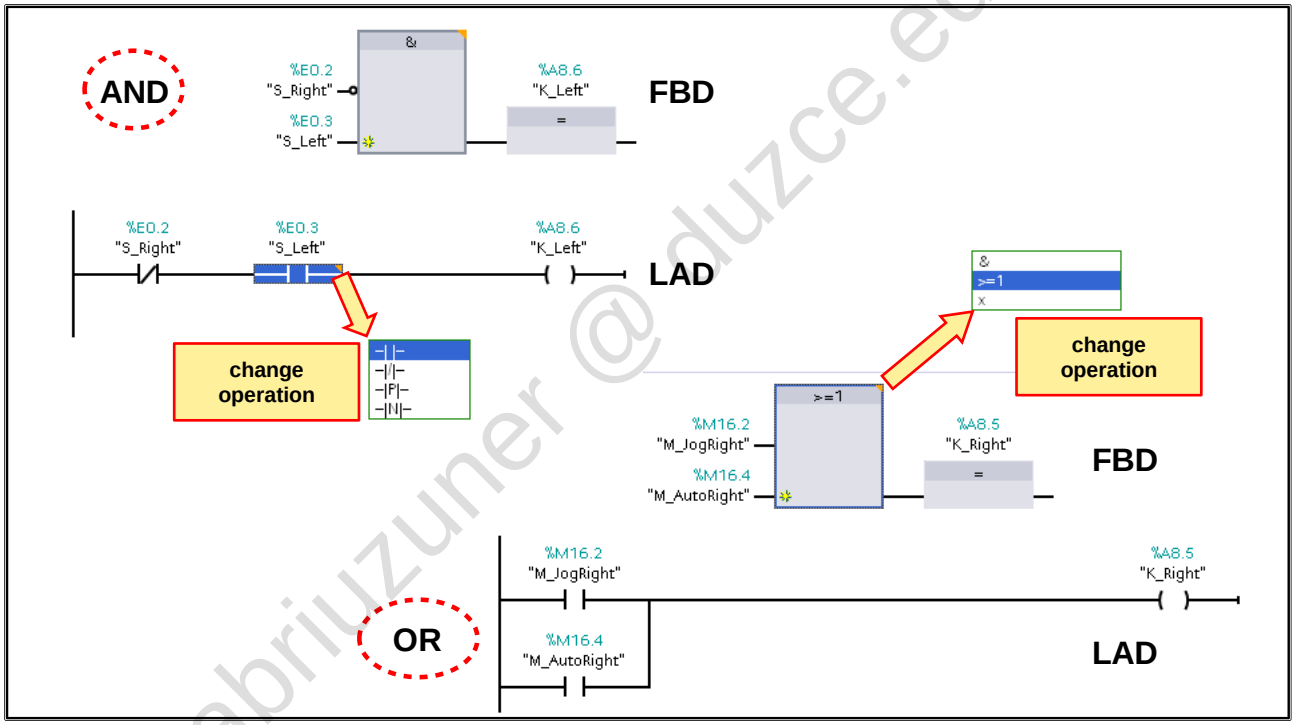
Check Symbols of the Program

In LAD, a symbol with the name "NO contact" is used for checking for signal state "1" and a symbol with the name "NC contact" to check for signal state "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 or result of logic operation (RLO) "1" when the checked operand has state or status "1".

The "NC contact" symbol delivers the result of check or result of logic operation (RLO) "1" when the checked operand has state or status "0".

7.3. Binary Logic Operations: AND, OR



AND and OR Logic Operations

With AND and OR logic operations, all binary operands can be queried, 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 check for signal state or Status "0" and "1", regardless of whether a hardware NO contact or NC contact is connected in the process.

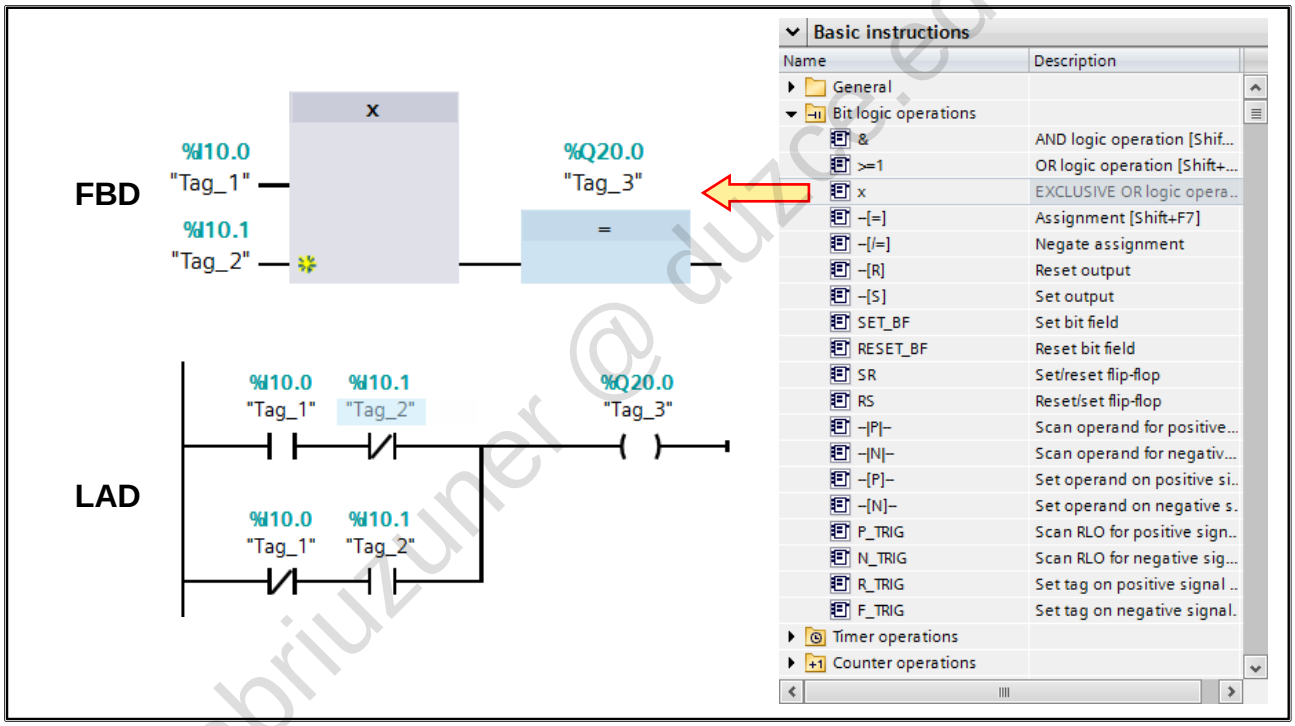
AND Logic Operation

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".

7.3.1. Binary Logic Operations: Exclusive OR (XOR)



XOR Logic Operation

With the XOR logic operation, all binary operands can be queried, 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.

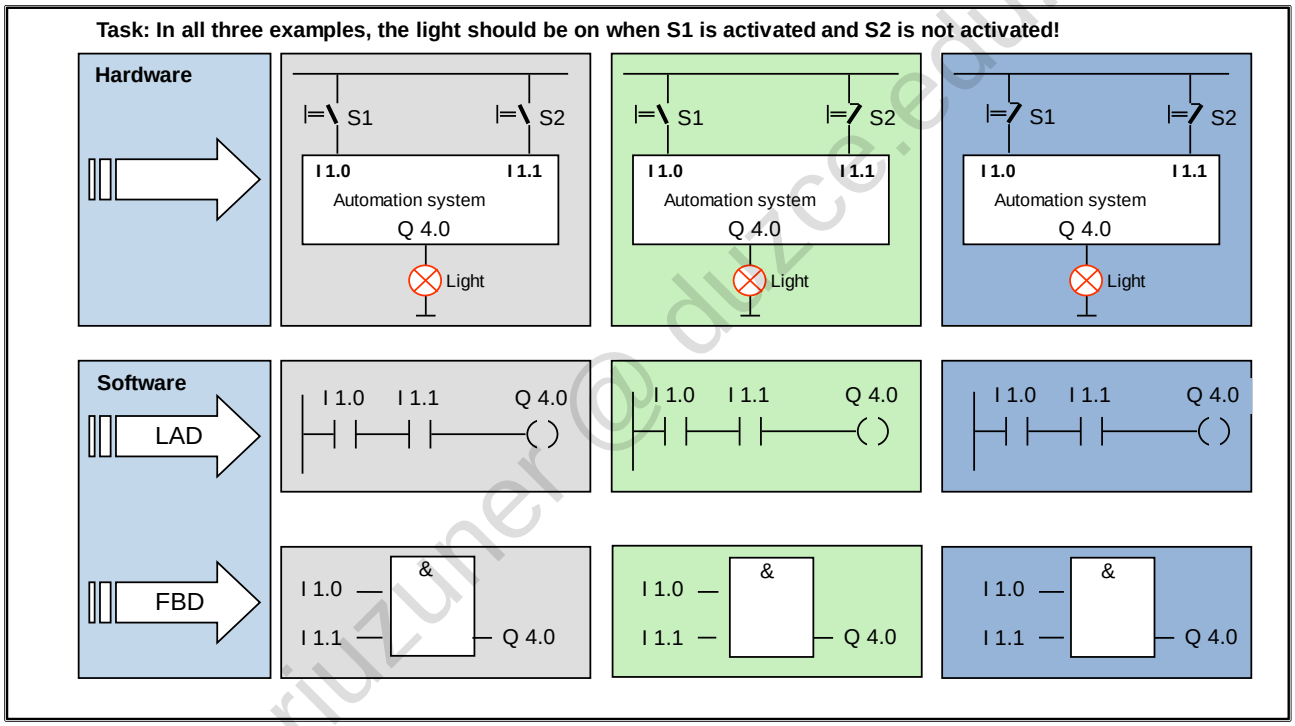
All inputs of the logic operations can be programmed as check for signal state or Status "0" and "1", regardless of whether a hardware NO contact or NC contact is connected in the process.

- 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 Programming Languages FBD and LAD

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.

7.3.2. Theory Exercise 1: Sensor and Check Symbols



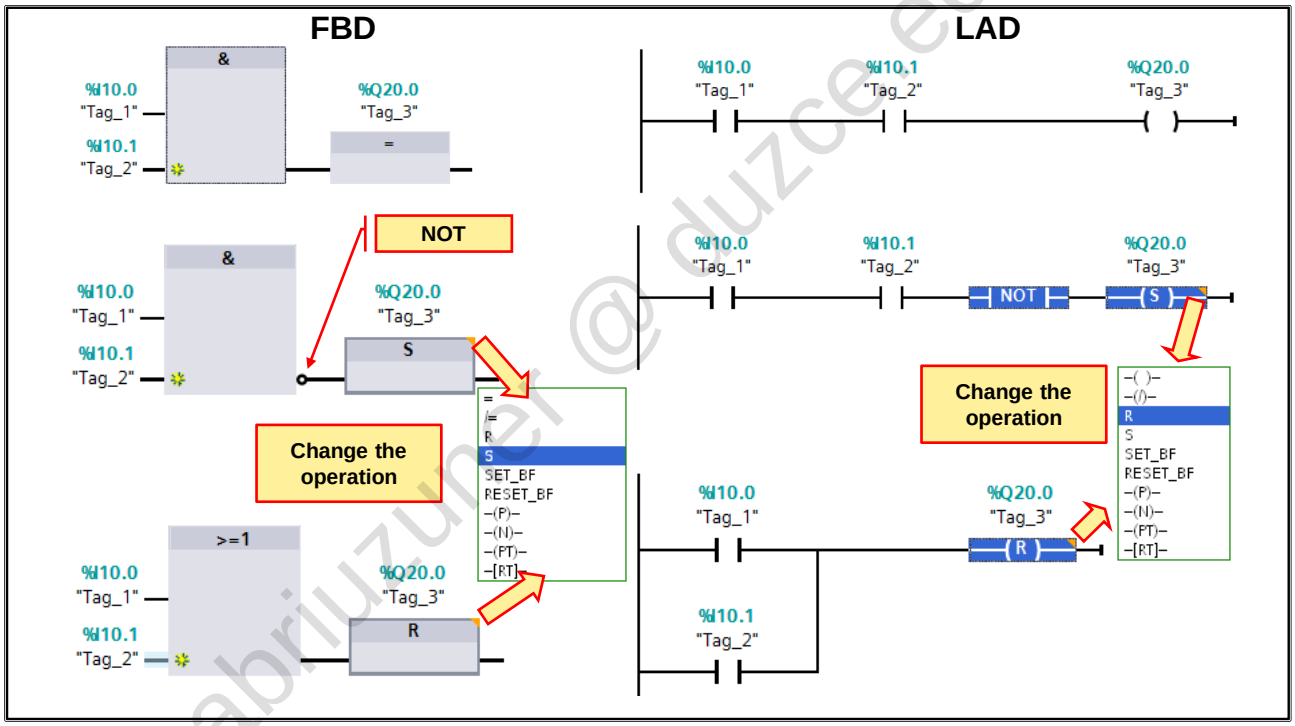
Task

Complete the programs in the picture above so that the following functionality is fulfilled: When the switch S1 is activated and the switch S2 is not activated, the light should be on in all three examples.

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.

7.3.3. Assignment, Set, Reset, NOT



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.

Set

If RLO = "1", the specified operand is assigned Status "1"

If RLO = "0", the status of the operand remains unchanged

Reset

If RLO = "1", the specified operand is assigned Status " 0"

If RLO = "0", the status of the operand remains unchanged

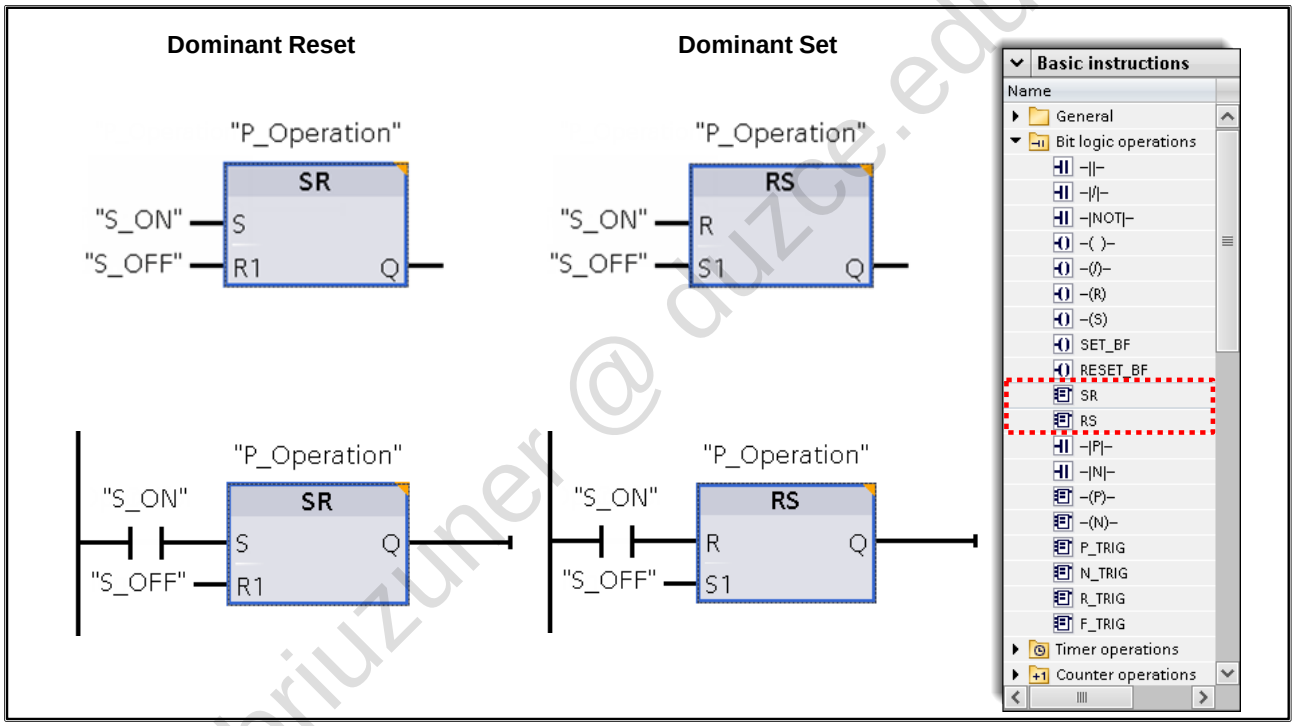
NOT

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

If, in the example shown, the RLO of the AND logic operation = "1", the NOT instruction inverts it to RLO "0" and the Set instruction is not executed (the status of "Tag_3" (Q20.0) then remains unchanged).

If the RLO of the AND logic operation = "0", the NOT instruction inverts it to RLO "1" and the Set instruction is executed ("Tag_3" (Q20.0) is assigned Status "1").

7.3.4. Flip Flops



Memory Function "Flip Flop"

A flip flop has a Set input and a Reset input. The operand is set or reset, depending on which input has an RLO = "1".

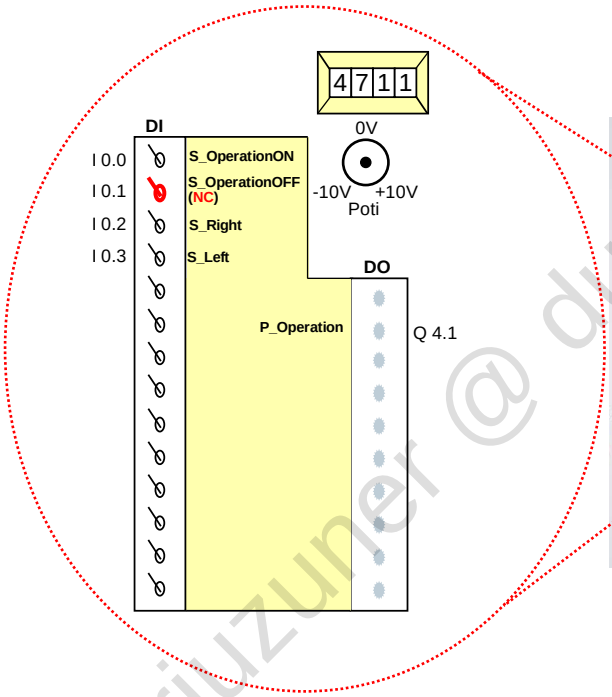
Priority (Dominancy)

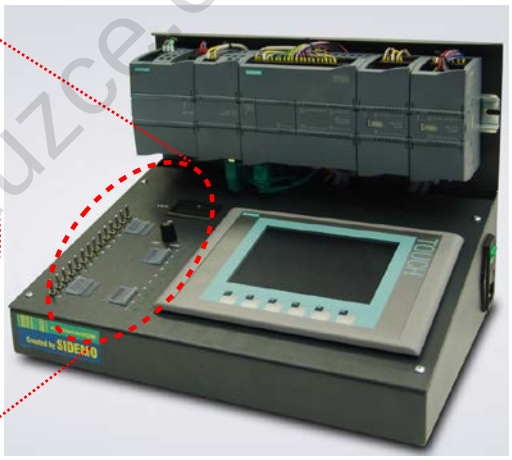
If there is a signal state or RLO = "1" at both inputs at the same time, the priority of the operation decides whether the operand is set or reset. For that reason, there are different symbols for the Dominant Set and Dominant Reset memory functions in LAD and FBD.

Note

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

7.4. Task Description: "FC_Mode"



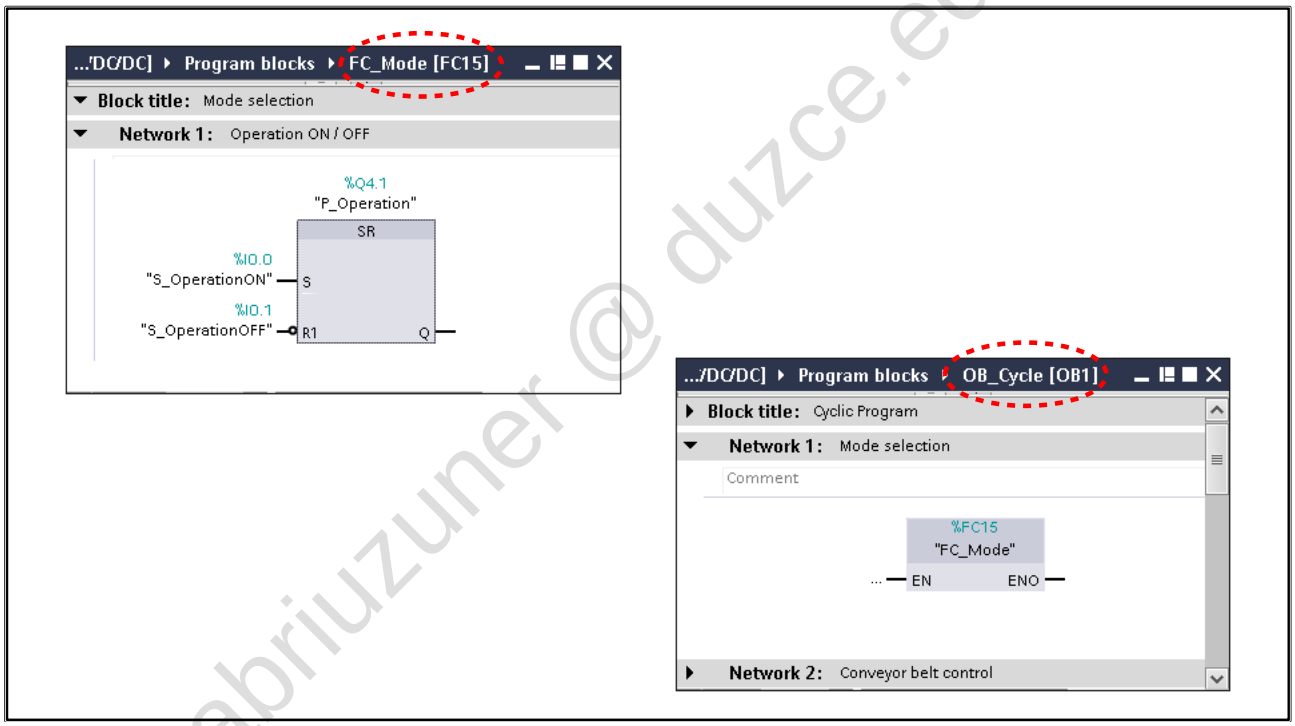


Task Description

A mode section for the system is to be programmed in "FC_Mode", which controls the indicator light "P_Operation" (Q4.1). This is then to be linked in "FC_Conveyor" as a further condition for jogging the conveyor motor.

- Function of the mode section in "FC_Mode":
  The system or "P_Operation" (Q4.1) is switched on via the simulator switch "S_OperationON" (I 0.0) and switched off via the simulator switch "S_OperationOFF" (I 0.1, NC).
- Integrate "P_Operation" (Q4.1) in "FC_Conveyor":
  The "Jogging the conveyor motor" programmed in "FC_Conveyor" is now only to be possible when "P_Operation" (Q4.1) is switched off.

7.4.1. Exercise 2: Programming the "FC_Mode" and the call of this block in the "OB_Cycle"



Task

In "FC_Mode" program a mode section for the distribution conveyor:

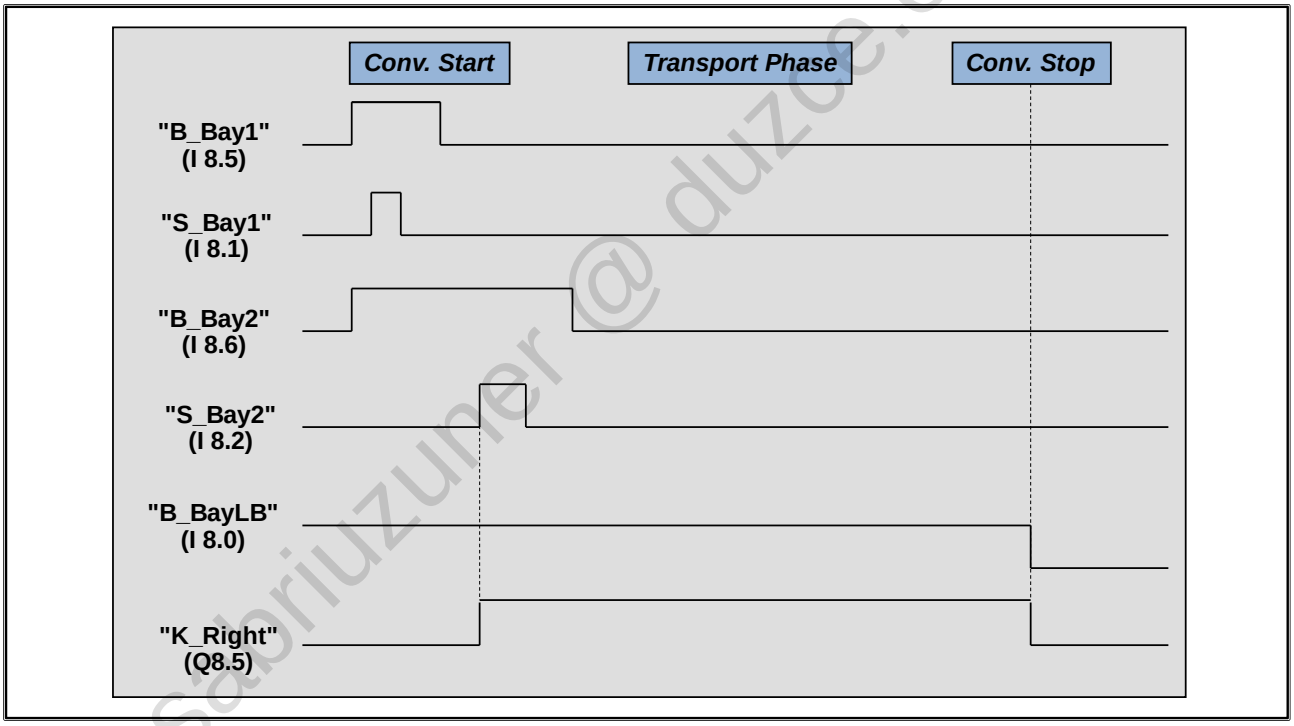
The operation or the indicator light "P_Operation" (Simulator LED Q4.1) is switched on via the simulator switch "S_OperationON" (I 0.0) and is switched off via the simulator switch "S_OperationOFF" (I 0.1, NC).

Then program the CALL of the "FC_Mode" in the "OB_Cycle".

What to Do

1. Create the new block "FC_Mode"
2. Open the "Instructions" Task Card, program the instructions shown in the picture using drag & drop and use the absolute addresses shown
3. Program the CALL of the "FC_Mode" in the "OB_Cycle"
4. In "FC_Conveyor", link the simulator LED "P_Operation" (Q4.1) logically in such a way that the conveyor motor can only be jogged when this is switched off
5. Download the modified blocks into the CPU and check the program function
6. Save your project

7.5. Task Description: Parts Transportation when "P_Operation" (Q4.1) is Switched On



Task Description:

When "P_Operation" (Q4.1) is switched ON, parts are to be transported from Bay 1 or Bay 2 to the light barrier "B_BayLB" (I 8.0). 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" (Q8.5) is started when

- on Bay 1 the proximity sensor "B_Bay1" (I 8.5) is occupied AND on Bay 2 the proximity sensor "B_Bay2" (I 8.6) is NOT occupied AND the pushbutton at Bay 1 "S_Bay1" (I 8.1) is pressed

OR

- on Bay 2 the proximity sensor "B_Bay2" (I 8.6) is occupied AND on Bay 1 the proximity sensor "B_Bay1" (I 8.5) is NOT occupied AND the pushbutton at Bay 2 "S_Bay2" (I 8.2) is pressed

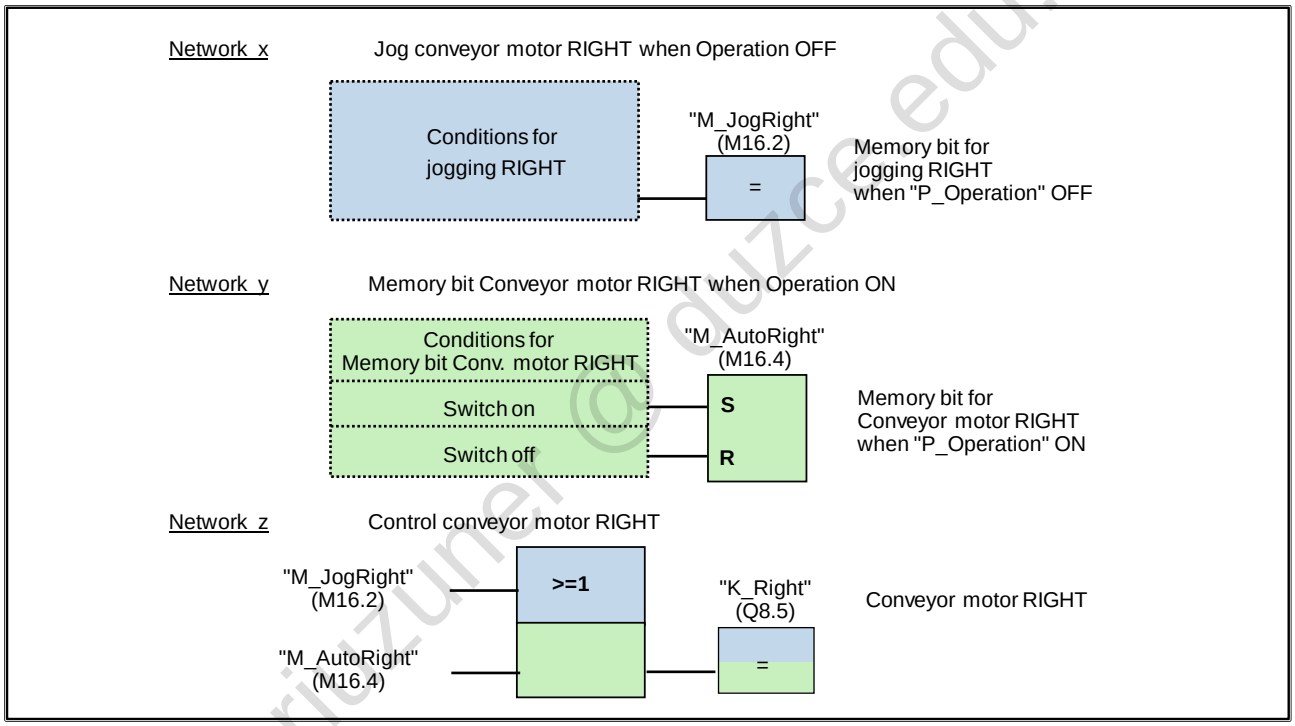
The conveyor motor "K_Right" (Q8.5) is stopped when

- the part has reached the light barrier "B_BayLB" (I 8.0)

OR

- "P_Operation" (Q4.1) is switched off

7.5.1. Exercise 3: Expanding "FC_Conveyor"



Task

Expand the "FC_Conveyor" block with the previously described functions.

Solution Notes

The conveyor motor "K_Right" (Q8.5) must now be controlled under two conditions:

- when "P_Operation" (Q4.1) is switched off while jogging RIGHT (in the picture Network x)
- OR, when "P_Operation" (Q4.1) is switched on, under the conditions described in the task (in the picture Network y)

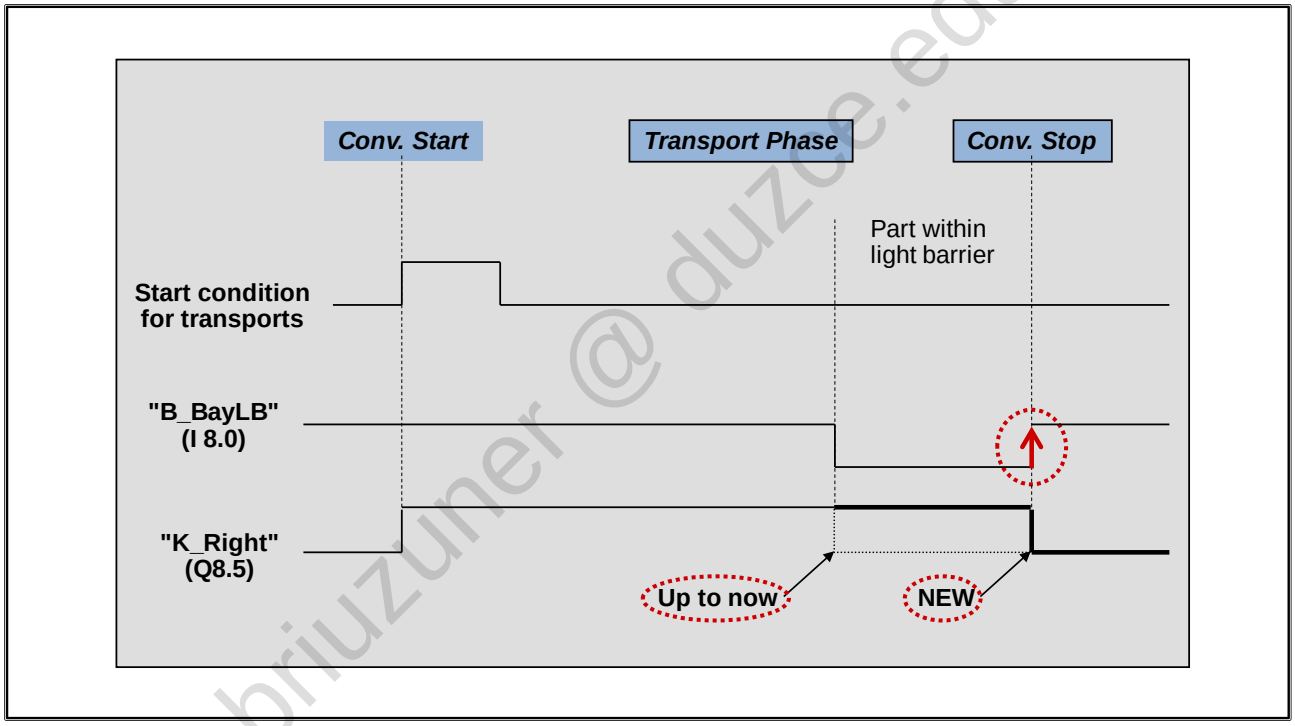
If, in both Network x and y, the results of logic operation of the conditions were each assigned to the output "K_Right" (Q 8.5), an error in the form of a double assignment would occur. Jogging the motor RIGHT in manual mode (Network x) would no longer function, since the state assigned to the output 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 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 picture). Use the memory bits shown in the picture and provide them with the symbols shown
3. Modify the already existing network "Jog Right" as described in the solution notes
4. Download the expanded block into the CPU and check the program function
5. Save your project

7.6. Task Description: Parts Transportation THROUGH the Light Barrier



Function Up to Now in “FC_Conveyor”

When "P_Operation" (Q4.1 = "0") is switched off, you can jog the conveyor motor "K_Right" (Q8.5) using the simulator switches "S_Right" (I 0.2) and "S_Left" (I 0.3).

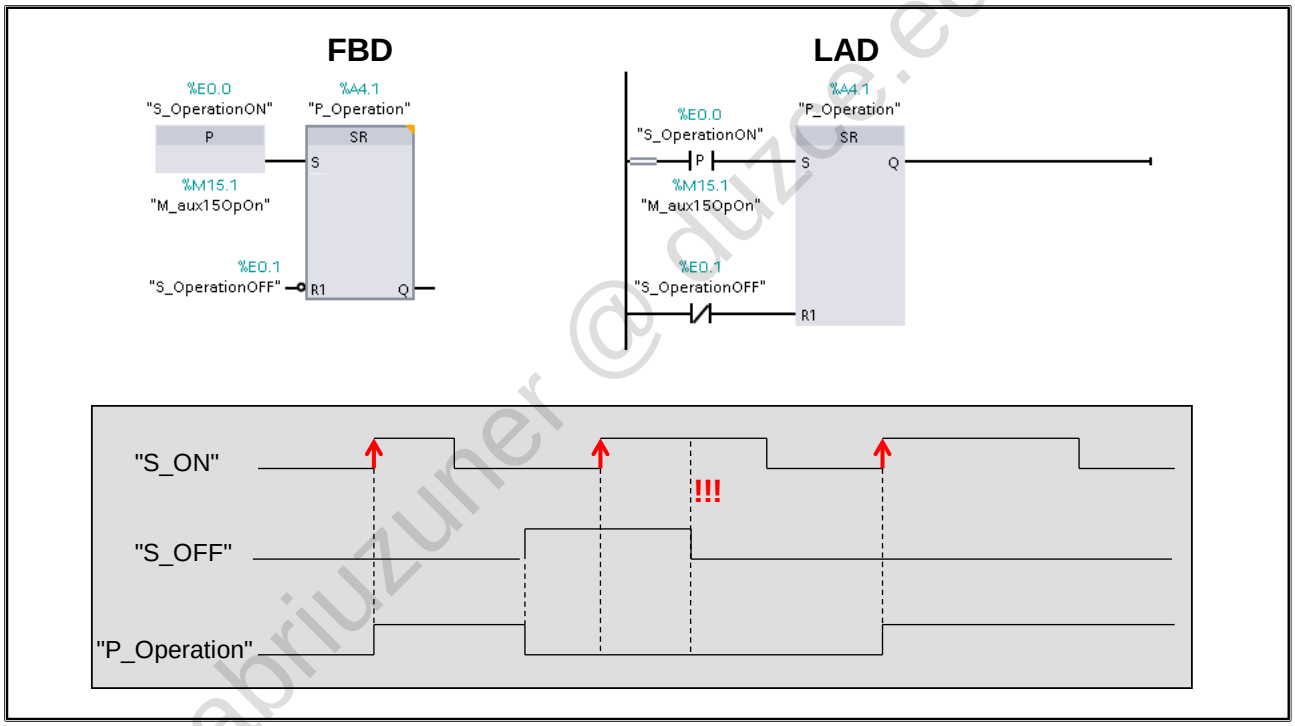
When "P_Operation" (Q4.1) is switched on, the conveyor motor "K_Right" (Q 8.5) is switched on when a part is placed on the conveyor exactly in front of one of the proximity sensors of Bay 1 or 2, and the pushbutton of the occupied bay is pressed.

The conveyor motor "K_Right" (Q8.5) is stopped when the part has reached the light barrier or "P_Operation" (Q4.1) is switched off.

Task Description:

The function of the "FC_Conveyor" to control the conveyor motor when "P_Operation" (Q4.1) 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.

7.6.1. Operand Edge Evaluations



Scan Operand for Positive Signal Edge

If the state of the operand at the input to the box changes from "0" to "1" ("rising edge"), RLO "1" is output at the output to the box for the duration of one program cycle.

Set Operand on Positive Signal Edge

This has the same behavior as with "scan operand for positive signal edge" whereby, however, the input signal (in the example "B_LB") is available at the output of the box and can be used by other logic operations.

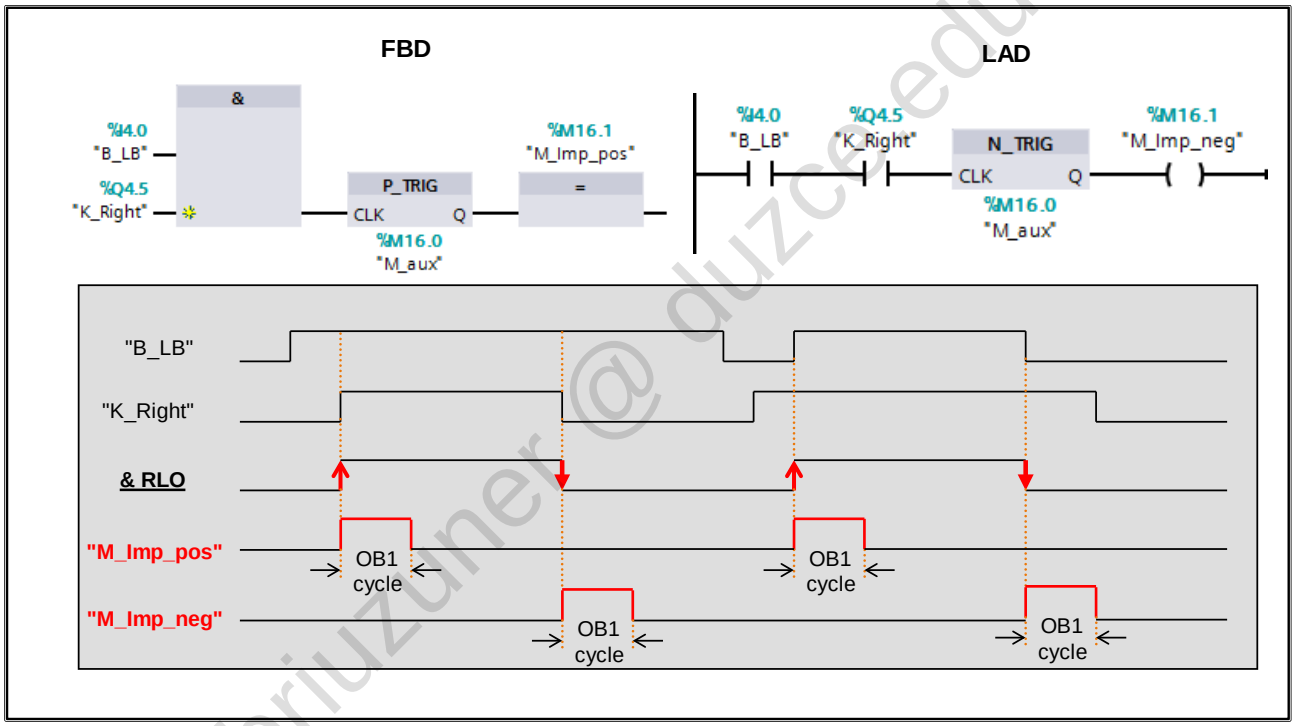
Set Tag on Positive Signal Edge

This has the same behavior as with "scan operand for positive signal edge" whereby, however, the evaluation occurs via a library function block (FB1001/1002) which is passed an instance data block, or which can also be created as a multiple instance within another FB.

Note

The instructions compare the current signal state of the input "B_LB" with the signal state in the previous cycle which is stored in an edge memory bit "M_aux_LB" or in the instance DB "DB_R_TRIG". When the instruction detects a change from "0" to "1", a positive signal edge exists. Subsequently, the current signal state is stored in the edge memory bit or in the instance DB. The status of the edge memory bit and the contents of the IDB must not be overwritten at another location in the program.

7.6.2. RLO Edge Evaluation

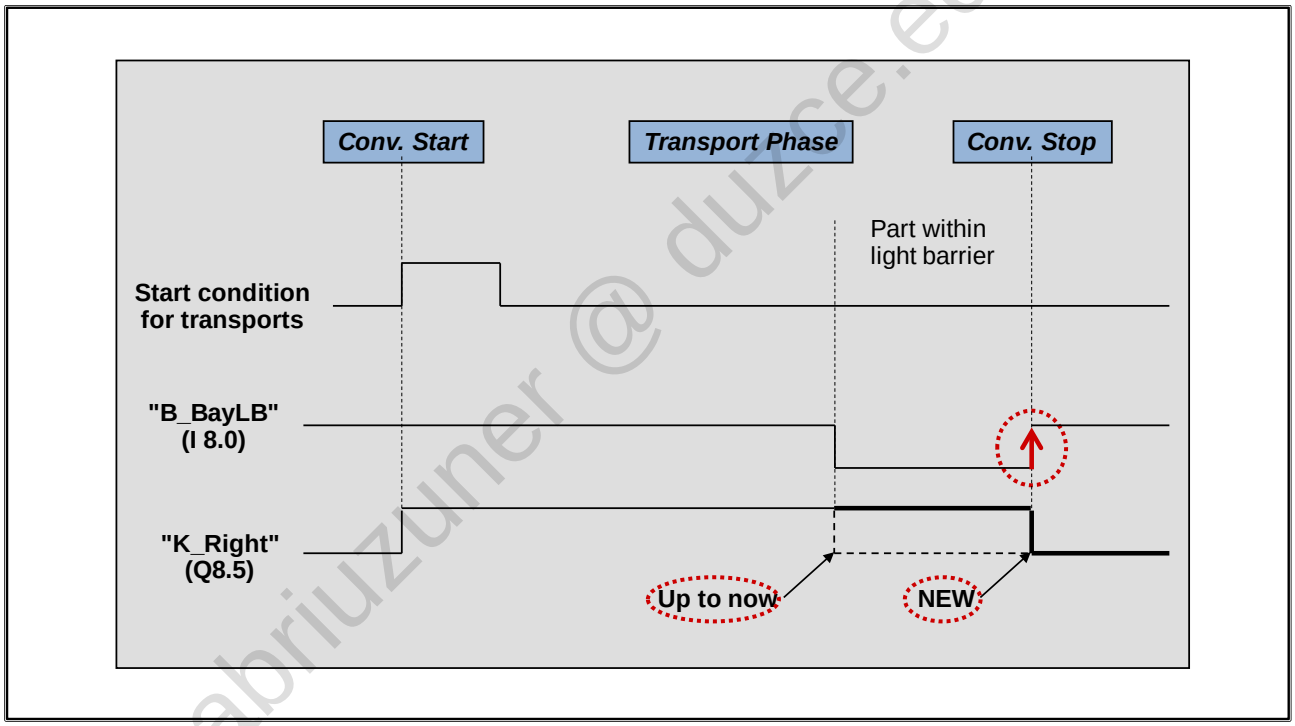


RLO Edge Evaluation (P_TRIG, N_TRIG)

With an RLO edge evaluation, it is possible to detect whether the status of a logic operation (in the example an AND RLO) 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, for the duration of one cycle, RLO "1" as the result, which can be further logically linked or can be assigned to another operand (in the example, the memory bit "M_Imp_pos" (M16.1)) as status. In the following cycle, the instruction then once again supplies "0" as the result even if the AND RLO still is status "1".

The instruction compares the current result or the RLO of the logic operation with its RLO in the previous program cycle. This is stored in a so-called edge memory bit for this (in the example "M_aux"). It must be ensured that the status of this edge memory bit is not overwritten at another location in the program. For every RLO edge evaluation, a separate edge memory bit must be used accordingly, even then when the same operand (in the example, AND RLO) is evaluated once again, for example, in another block!

7.6.3. Exercise 4: Integrating an Edge Evaluation in "FC_Conveyor"



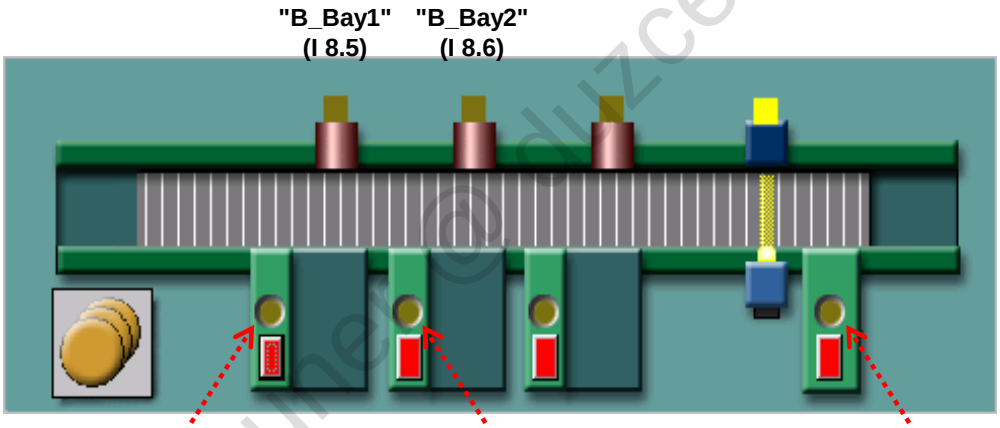
Task:

When "P_Operation" (Q4.0 = "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 for "M_AutoRight" (M16.4) instead of the light barrier signal "B_BayLB" (I 8.0) itself. For the necessary edge evaluation of the light barrier signal use the bit memory "M_aux16LB" (M16.0) as an edge memory bit
2. Download the modified "FC_Conveyor" block into the CPU and check the program function
3 Save your project

7.7. Task Description: Controlling the Indicator Lights



Task Description

When "P_Operation" (Q4.0) is switched on, the indicators lights are to be controlled as follows:

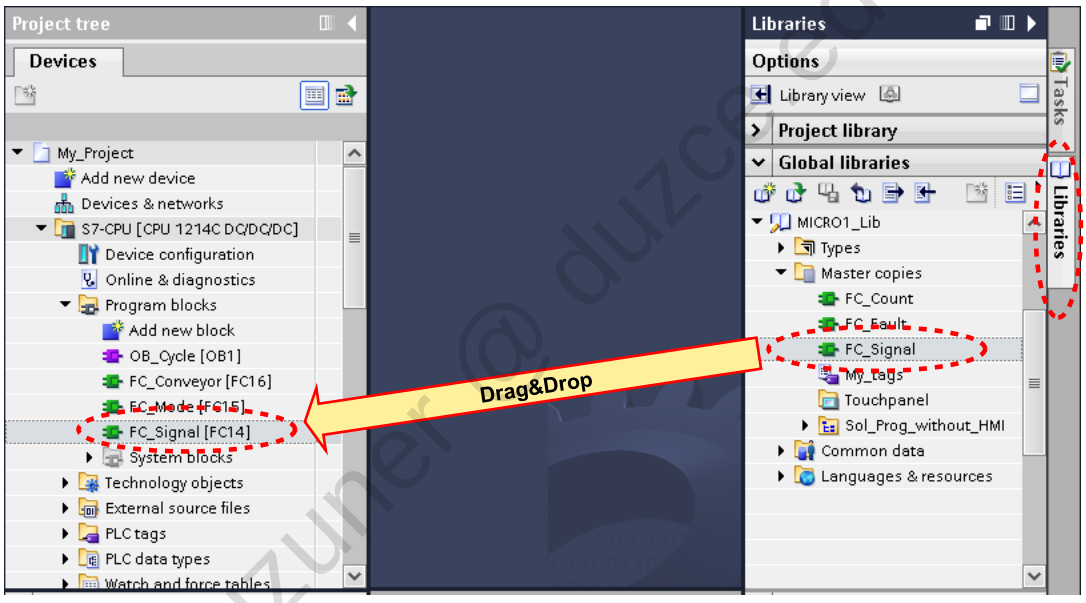
The indicator lights "P_Bay1" (Q8.1) and "P_Bay2" (Q8.2) show

- a constant light when a new part can be placed on the conveyor (conveyor motor is stopped and both proximity sensors are free)
- 1Hz flashing light at the bay where the associated proximity sensor detects a part, but only if the conveyor has not yet been started (if parts are placed on the conveyor at both proximity sensors, neither indicator light must light up)
- 2Hz flashing light as long as the conveyor motor is running

The indicator light at the light barrier bay "P_BayLB" (Q8.4) shows 2Hz flashing light if the conveyor motor is running.

The described functions are already programmed in the "FC_Signal" block, which is stored in the "Micro1_Lib" global library. The block still contains errors and is to be commissioned by you in the next exercise.

7.7.1. Exercise 5: Commissioning "FC_Signal"



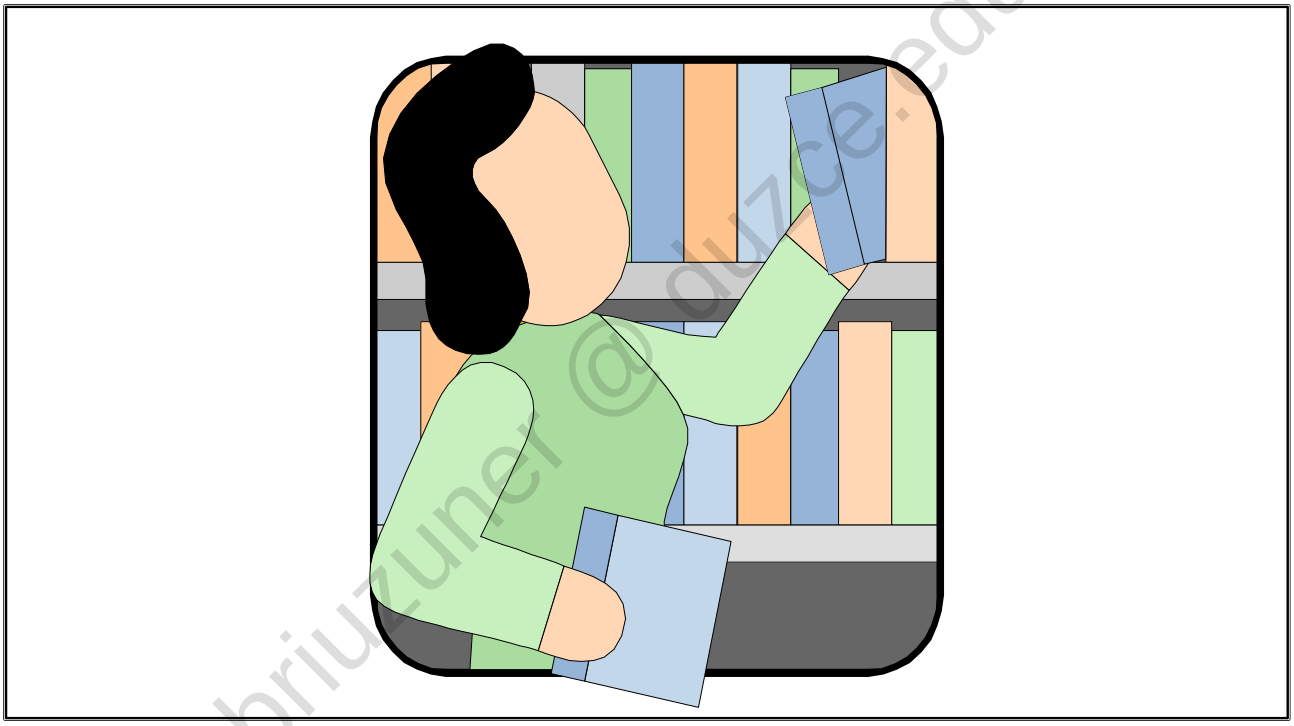
Task

The "FC_Signal" block is to be copied from the "MICRO1_Lib" global library into the project and commissioned.

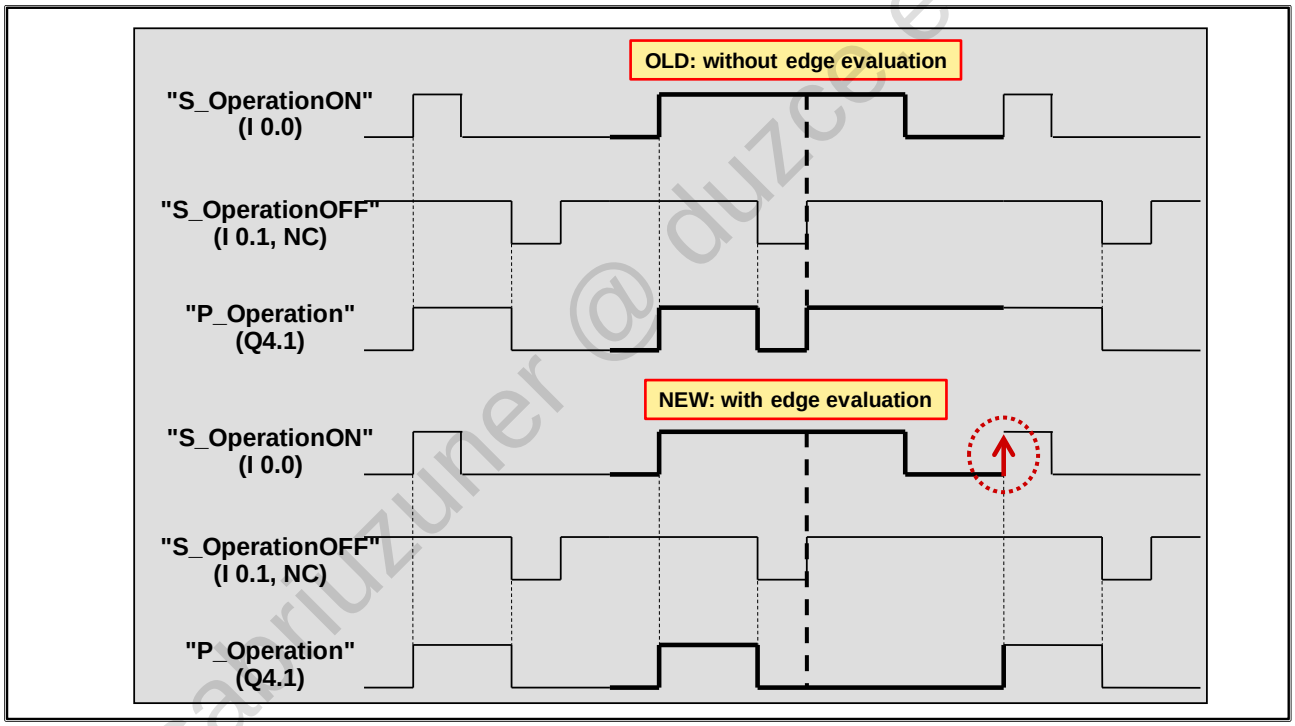
What to Do

1. Using drag & drop, copy the "FC_Signal" block into the "Program blocks" container from the "MICRO1_Lib" global library (as shown in the picture)
2. Program the call of "FC_Signal" in "OB_Cycle"
3. Download all modified blocks into the CPU
4. Test the program function and correct the block in such a way that the indicator lights are controlled as required. To do so, use the test function "Monitor block", even if you have already recognized the error!
6. Save your project

7.8. Additional Information



7.8.1. Additional Exercise 6: Optimizing "FC_Mode"



Function Up to Now of "FC_Mode"

"P_Operation" (Q4.1) is switched on with the simulator switch "S_OperationON" (I 0.0) and switched off with the simulator switch "S_OperationOFF" (I 0.1, 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").

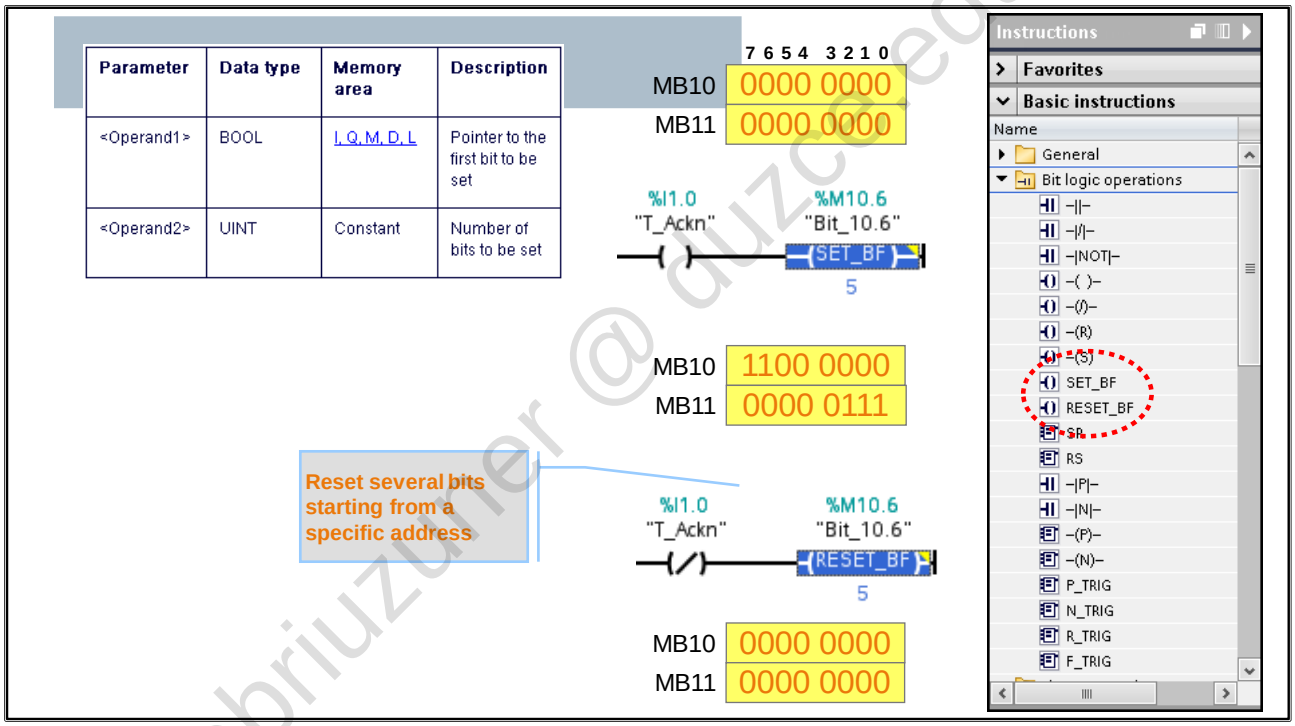
Task

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" (Q4.1), insert an edge evaluation of the switch "S_OperationON" (I 0.0). For the edge evaluation, use the bit memory "M_aux15Opon" (M15.0) as edge memory bit
2. Download the modified "FC_Mode" block into the CPU and check whether it fulfils the desired functions!
3. Save your project

7.8.2. Setting / Resetting Bit Fields

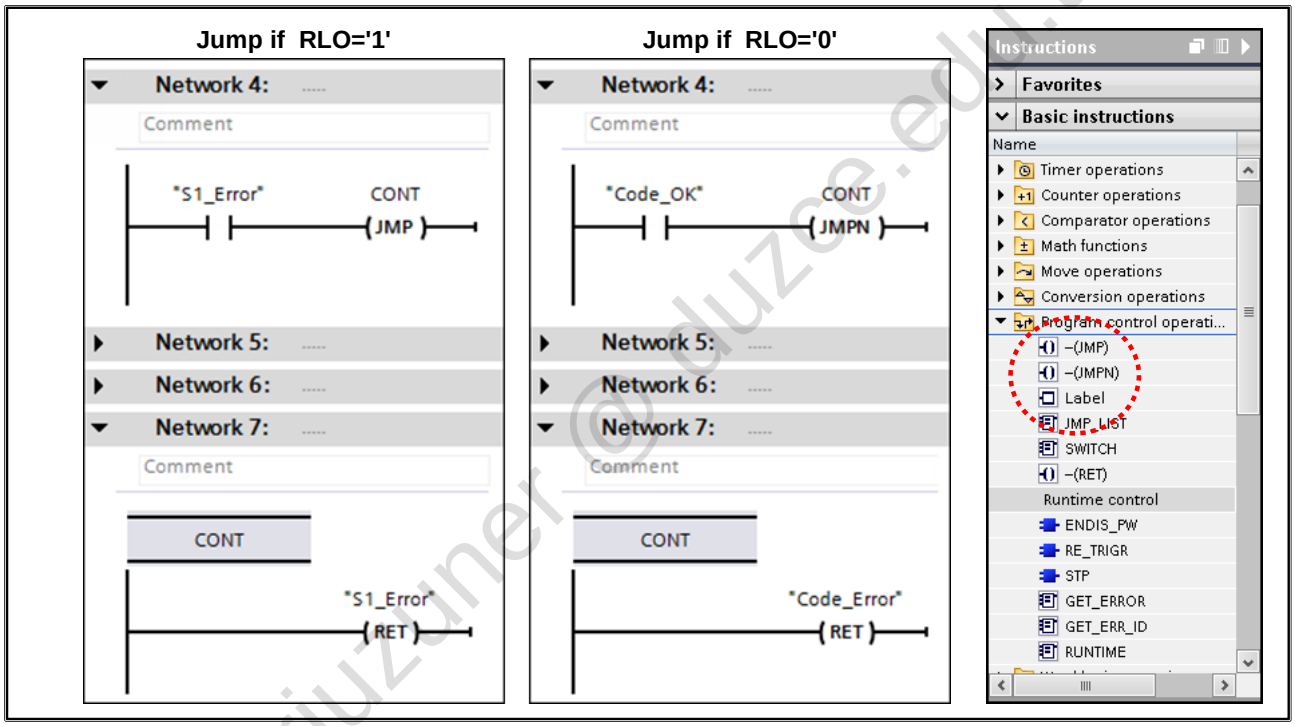


Setting / Resetting a Bit Field RESET_BF

With the operation "Reset bit field" (RESET_BF), several bits starting from a specific address can be reset. The number of bits which are to be reset can be determined via the Parameter N. Through the operation, the bit with the lowest address in the bit field must be specified. The bits are reset in ascending bit and byte address.

The operation is only executed if an RLO "1" exists at input EN. For RLO "0", the operation is not executed.

7.8.3. 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 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. The program execution is continued in the calling block with the instruction that follows the call of this block.

Contents

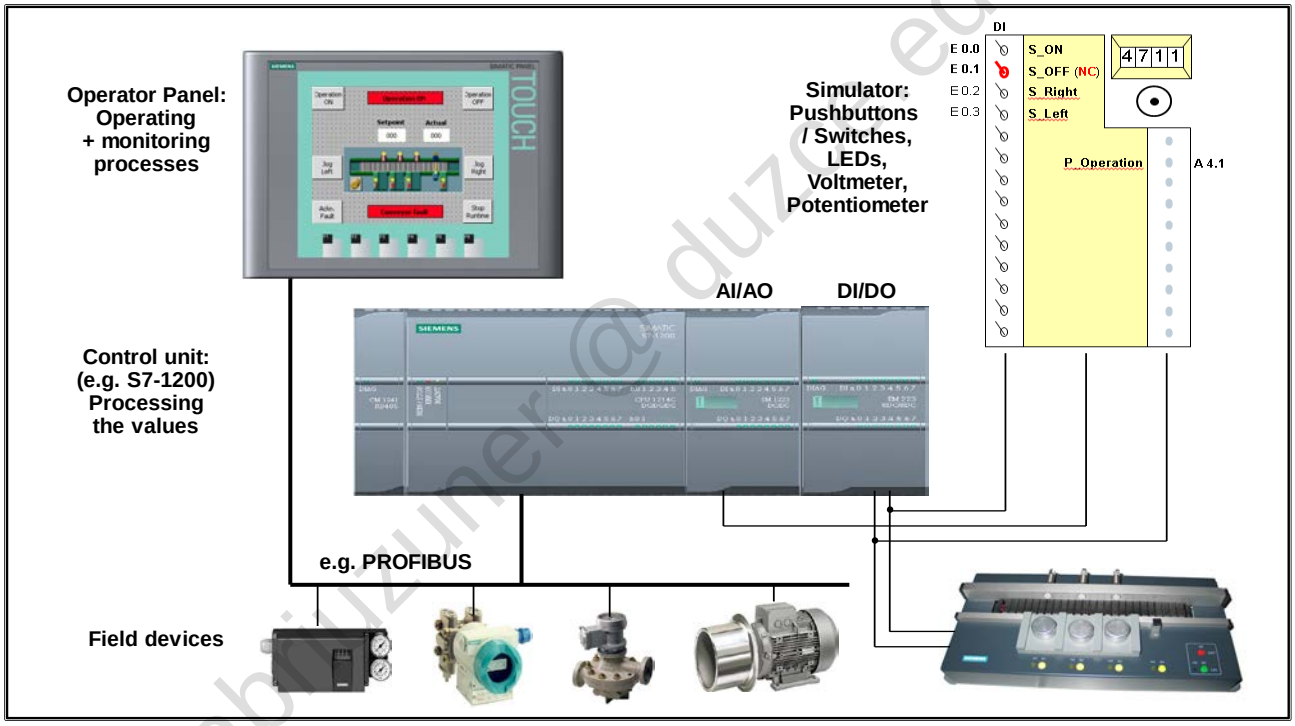
8

8. Digital Operations

At the end of the chapter the participant will…

- **... be familiar with the application purpose of data blocks**
- **… be familiar with the S7-1200 data types**
- **… be familiar with the different counter and timer functions**
- **… be able to use and program counter and timer functions**
  be able to select a suitable data type

8.1. 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.

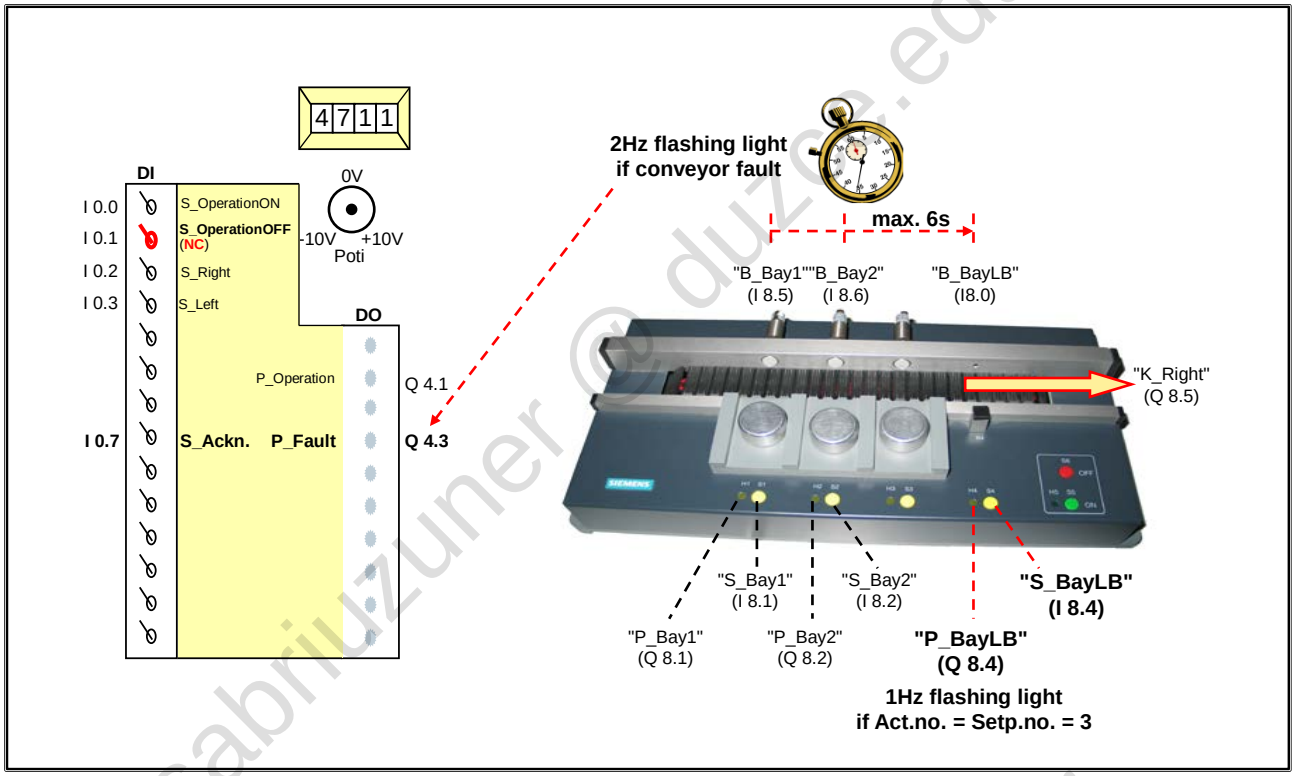
Operating and Monitoring

The goal of process monitoring is too quickly, clearly and concisely providing the operator with up-to-the-minute information about the working machine or system and to allow the operator to control the process. While in the past mostly simple, that is, "dumb" input and output devices, such as 7-segment displays and thumbwheel buttons were used to display and enter digital values, today "intelligent" operating and monitoring devices are frequently connected to a PLC. Depending on the type of device connected, different number formats for the coding of data are used to transmit data between device and PLC, as well as for storing and processing data in the PLC.

Field Devices

Today field devices that acquire process data or that control the process are supplied directly with digital variables through field bus systems. The connection of field devices, such as drives or weighing systems, using analog input and output modules is becoming more and more a thing of the past.

8.2. Task description: Counting the Transported Parts and Monitoring the Transportation time



Counting the transported part

The number of transported parts (actual quantity) is to be recorded with a counter and stored in the variable "MW_Act" (MW20) If the ACTUAL quantity has reached the SETPOINT quantity of 3, the indicator light "P_BayLB" (Q8.4) shows a continuous light and the indicator lights at Bay 1 and 2 are dark.

The counter is reset to ‘0’ when "P_Operation" (Q4.1) is switched on or when ACTUAL = SETPOINT = 3 is acknowledged via the pushbutton "S_BayLB".

Monitoring the transportation time

The automatic transport sequences are to be monitored for time.

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" (Q 4.3).and can be acknowledged via the simulator switch "S_Acknowledge" (I 0.7).

If there is an unacknowledged fault, no new transport sequence can be started.

8.3. 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 (U**ser **D**efined **D**ata **T**ype**)**

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).

8.3.1. Elementary Data Types: Bit and Numeric

	Description	Size (Bit)	S7-1200	S7-1500	Example
Bit Data Types	BOOL BYTE WORD DWORD	1 8 16 32	✓	✓	TRUE B#16#F5 W#16#F0F0 DW#16#F0F0FF0F
	LWORD	64	✗	✓	LW#16#5F52DE8B
Numeric Data Types	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.

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

8.3.2. 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	64	✗	✓	LT#11350d20h25m14s830ms652µs315ns
	LTIME_OF_DAY				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.

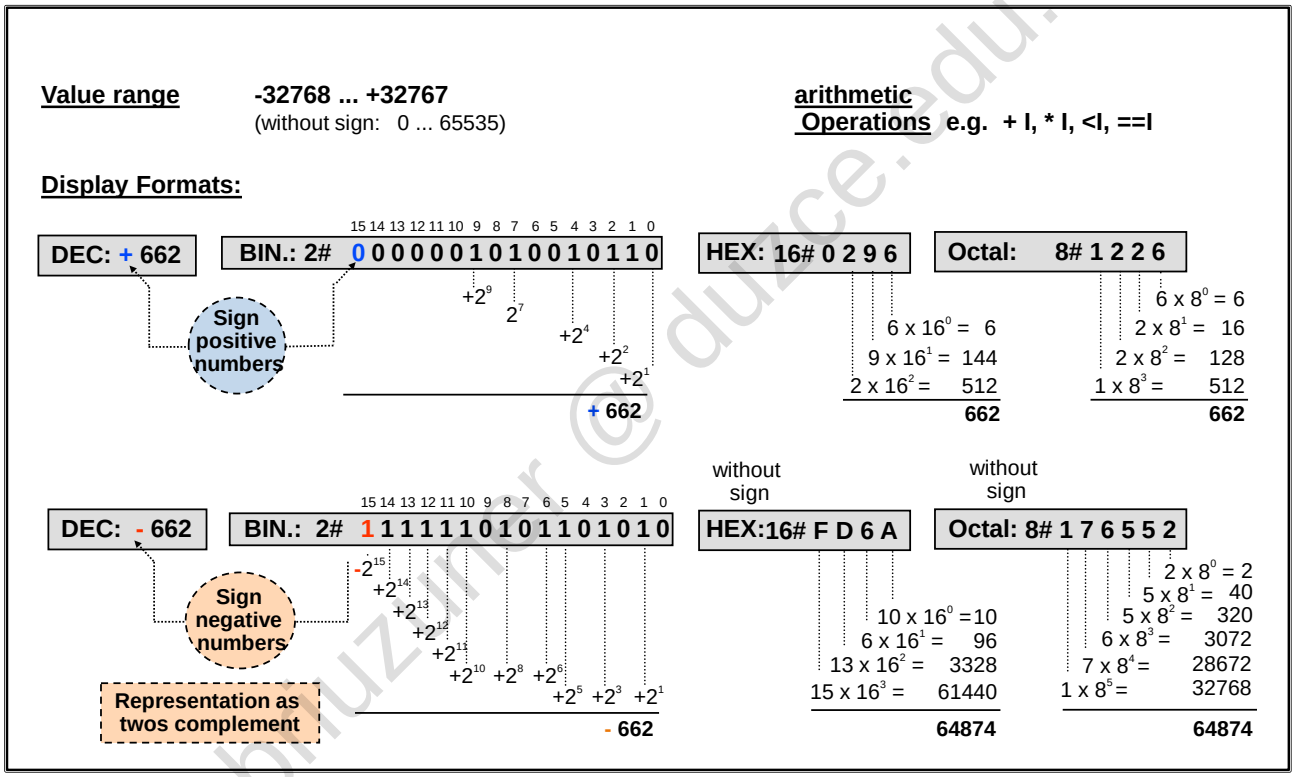
S5TIME

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.

8.3.2.1. 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

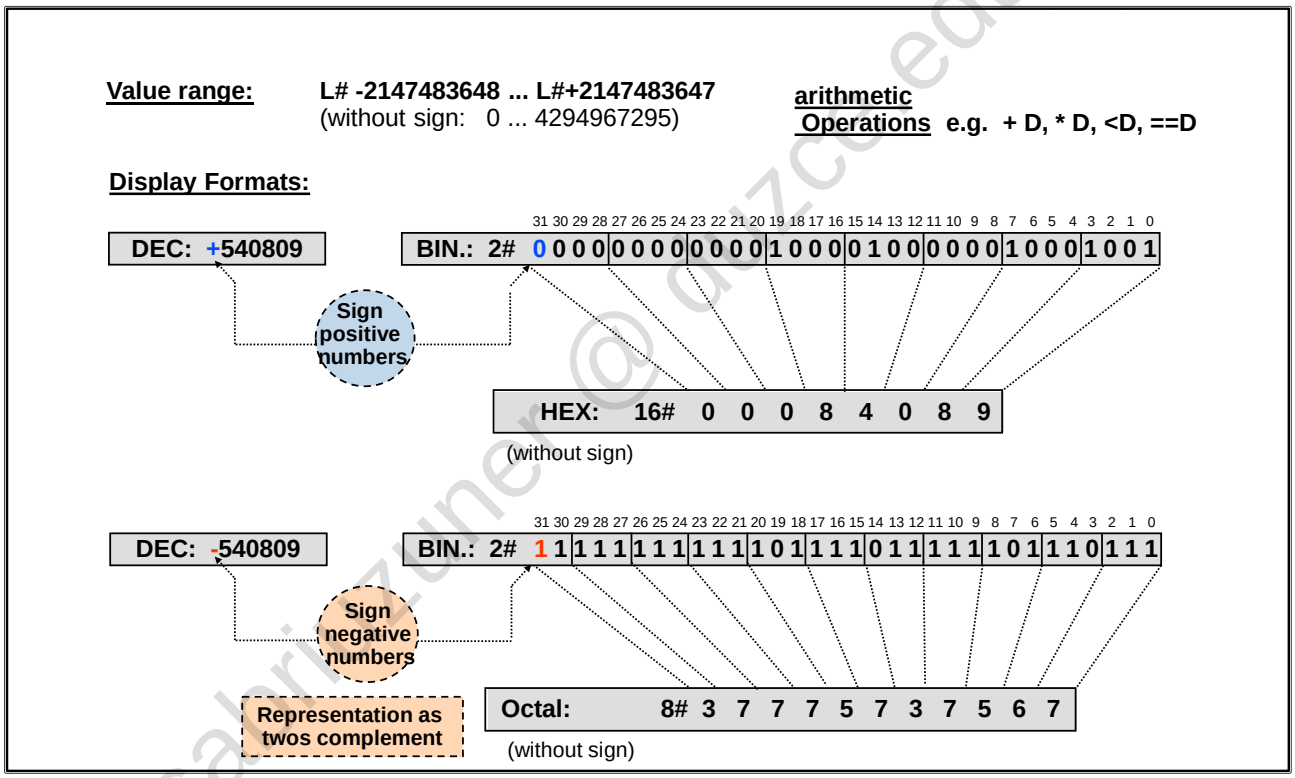
STEP7 uses the Decimal display format, that is, with sign and without explicit format description, to specify the constants of the Integer data type. The use of constant Integer values in the Binary, Hexadecimal, Octal display formats is possible in principle, but because of the poor legibility, they are more or less not suitable.

Binary

In a digital computer system, all values are stored in a binary 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 twos 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).

8.3.2.2. 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 +2147483647.

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.

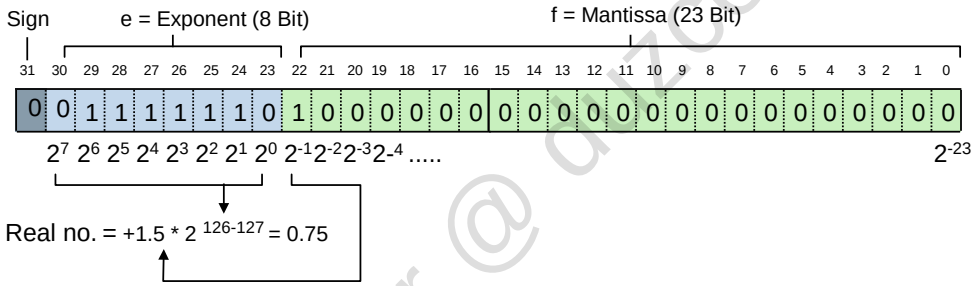
Octal Number

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.

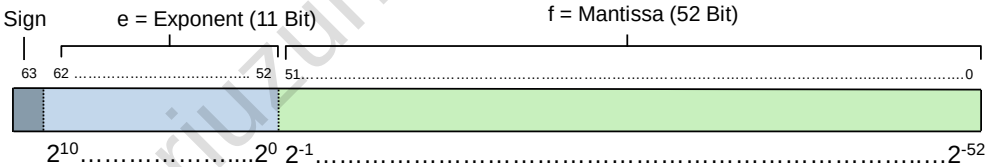
8.3.2.3. REAL and LREAL (Floating-point Number) Data Type

General format of a Real number (32 Bit) = (Sign) • (1.f) • (2^{e-127}) **arithmetic Operations e.g. + R, *R, <R, ==R sin, ln, exp, …..**

Example: 0.75



General format of an LReal number (64 Bit) = (Sign) • (1.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., must be processed, it becomes necessary to use Real values (real numbers, "decimal numbers"). To be able to represent such values, binary digits must be defined whose value is less than 1 (power of base 2 with negative exponent).

Format

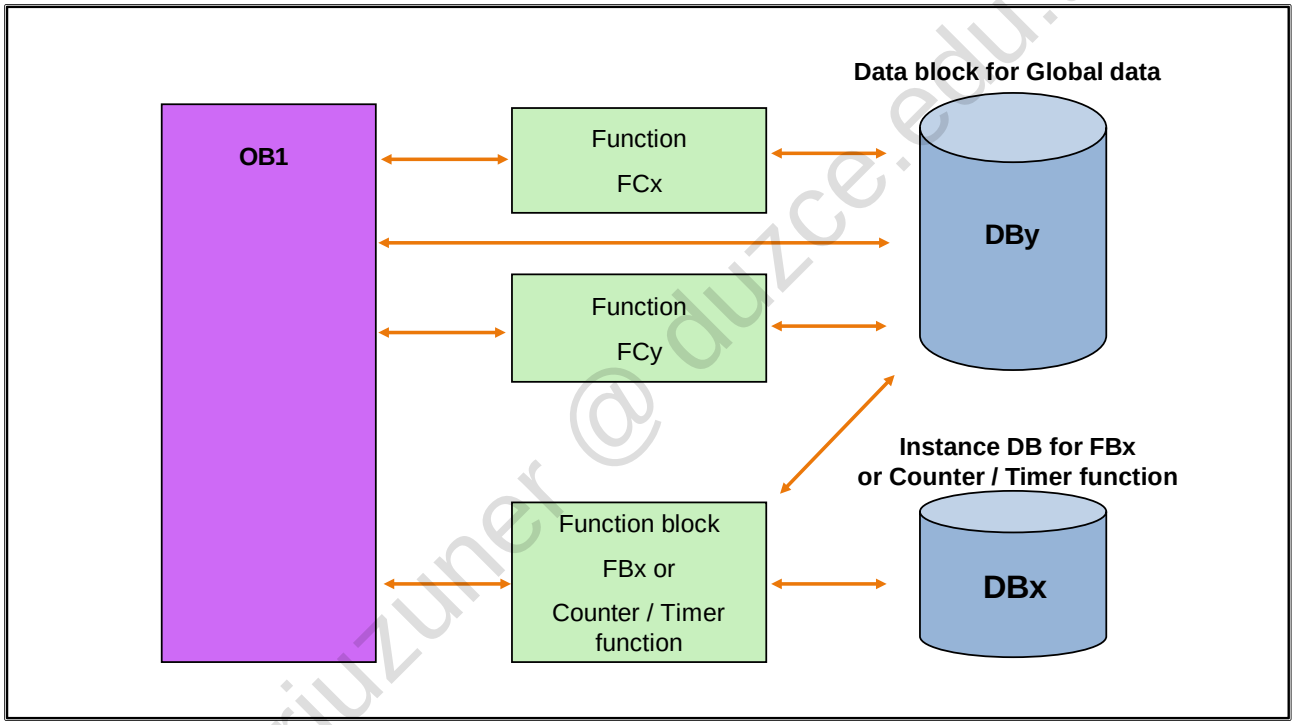
In order the greatest possible value range within a defined memory capacity, you must be able to select the decimal point position as required. Early on, IEEE defined a format for floating-point numbers. This format was laid down in IEC 61131 and was included in STEP 7. This format makes it easy to process 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.

8.4. Timer / Counter Instance Data Blocks



Data Blocks

Data blocks are used for storing user data. They occupy memory space in the user memory of the CPU. Data blocks contain variable data (such as numeric values) with which the user program works. The user program can access the data in a data block. Access can be made symbolically or absolutely.

Area of Application

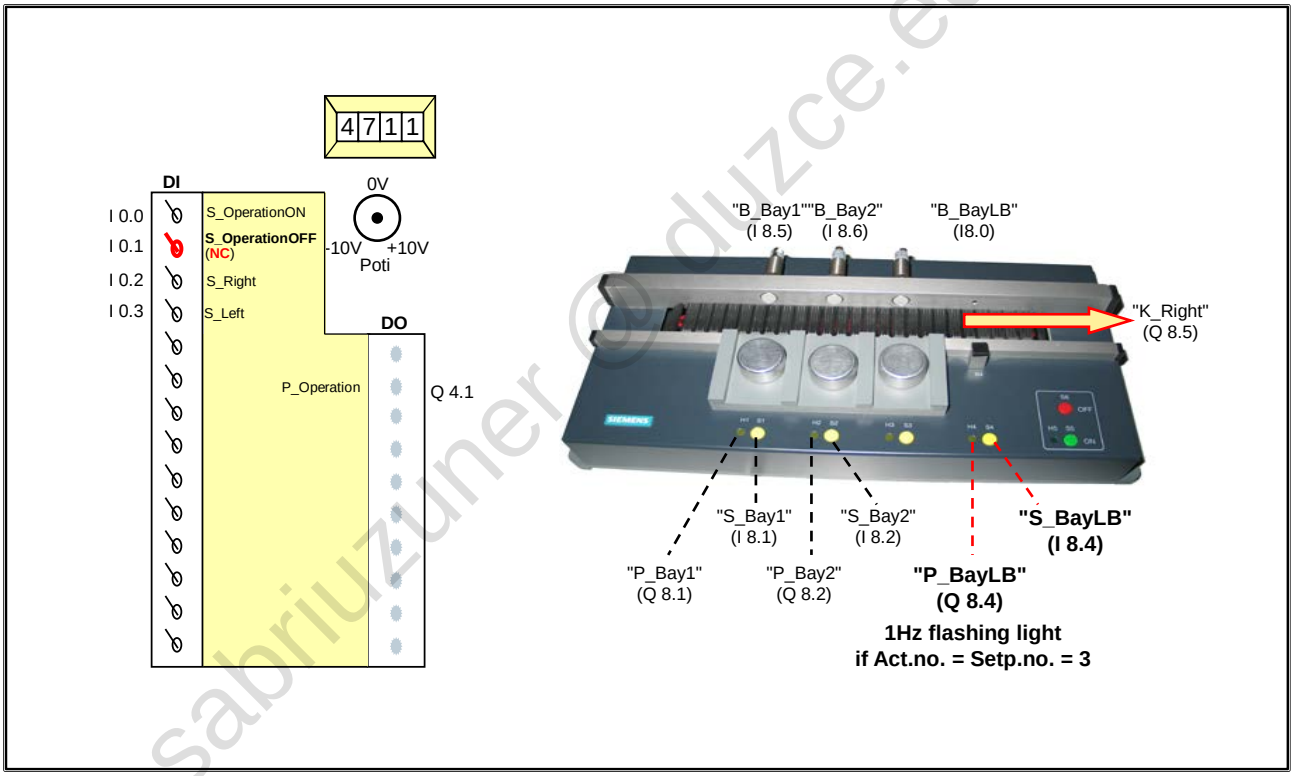
Data blocks are used with two different purposes:

- Global data blocks: These contain information that all the logic blocks in the user program can access
- Instance data blocks: These are always assigned to a particular FB or functions such as counters and timers. The data of these instance DBs should only be processed by the associated FB or counter / timer function.

Creation of DBs

Global DBs are created either with the Program Editor or according to a previously created "user-defined data type". Instance data blocks are generated when a function block or a counter / timer function is called.

8.5. Task Description: Counting the Transported Parts



Function Up till Now

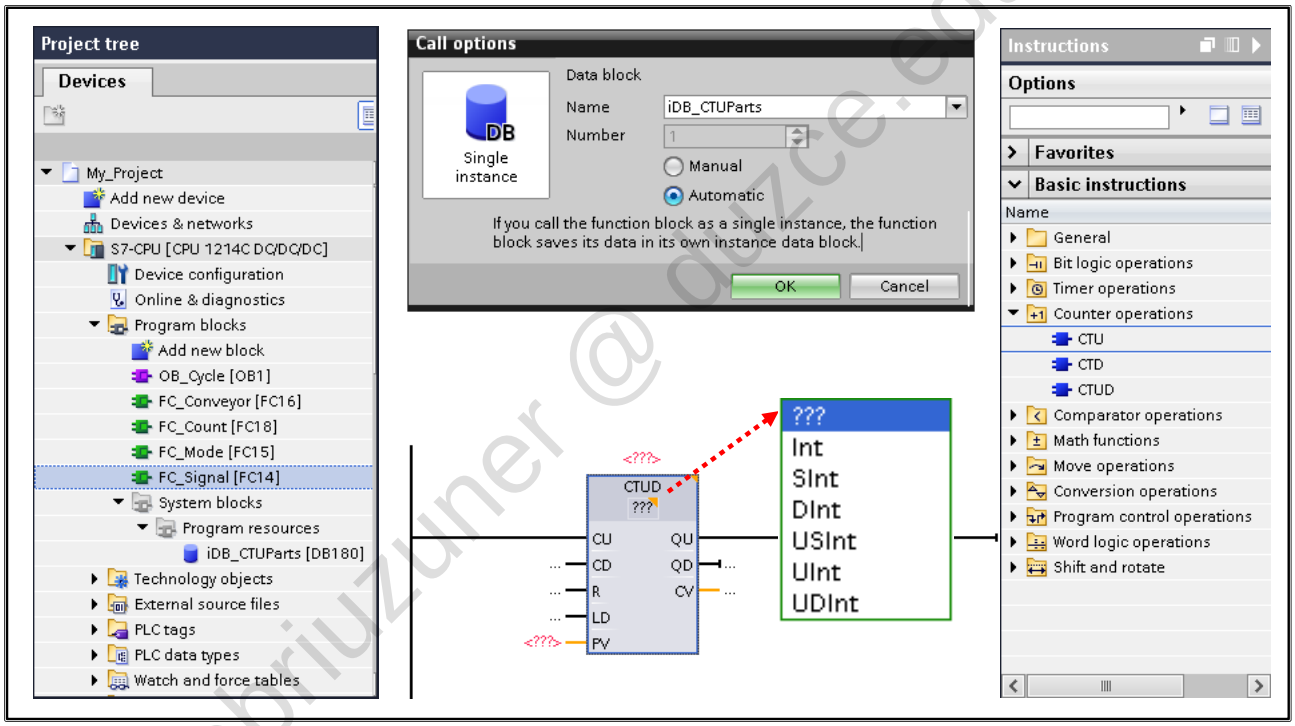
When "P_Operation" (Q4.1) is switched on, parts are transported from Bay 1 or Bay 2 through the light barrier. The transport sequence starts as soon as a part is placed on Bay 1 or Bay 2 and the associated bay's pushbutton is pressed and it ends as soon as the part has passed the light barrier.

Task

- When "P_Operation" (Q4.1) is switched on, the transported parts are to be counted as soon as they have passed through the "B_BayLB" (I 8.0) light barrier ("B_BayLB" 0 → 1)
- The number of transported parts (ACTUAL quantity) is to be recorded with a counter and stored in the variable "MW_ACT" (MW20)
- If the ACTUAL quantity has reached the SETPOINT quantity of 3, the indicator light "P_BayLB" (Q8.4) shows a continuous light and the indicator lights at Bay 1 and 2 are dark (= 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")
- The counter is reset to '0' when "P_Operation" (Q4.1) is switched on or when ACTUAL=SETPOINT is acknowledged via the pushbutton "S_BayLB"

The described functions are already programmed in the block "FC_Count" which is stored in the "MICRO1_Lib" global library. The block still contains errors and is to be commissioned by you in the next exercise.

8.5.1. 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.

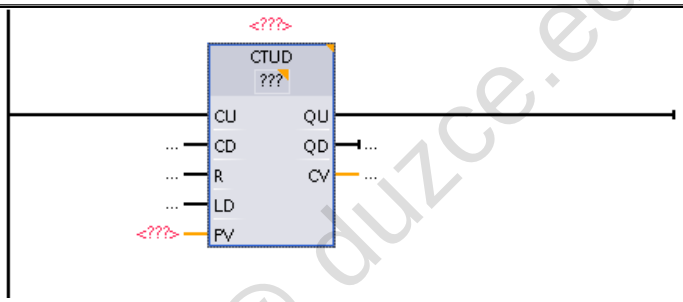
Value Range of a Counter

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.

8.5.2. Counter Function: Inputs



Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the QU output is set. / Value to which the CV output is set with LD = 1.

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. 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, if RLO '1' is at input R, the count is set to 0 and rising edges or RLO '1' at the inputs CU, CD and LD have no effect on the current count.

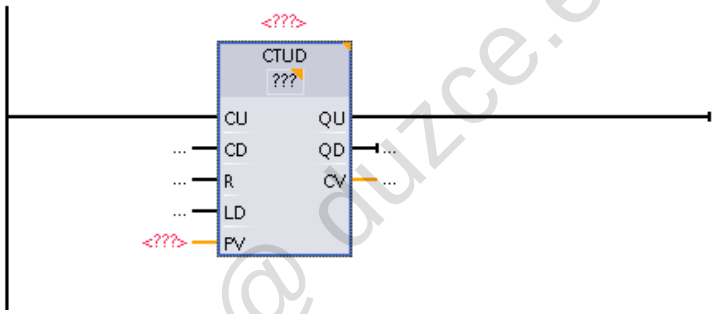
Input LD (Load, Only for Down Counters)

The input LD acts statically, that is, if RLO '1' is at input LD, 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 if RLO '1' is at input LD. The variable or constant passed to the input must be compatible with the data type of the counter.

8.5.3. Counter Function: Outputs



Parameter	Declaration	Data type	Memory area	Description
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, WCHAR, DATE (INT)	I, Q, M, D, L, P	Current counter value

Output QD (Only for Down Counters)

The status of the down counter can be checked at the output QD. If the current count is less than or equal to zero, the output QD has Status "1", otherwise, Status '0'.

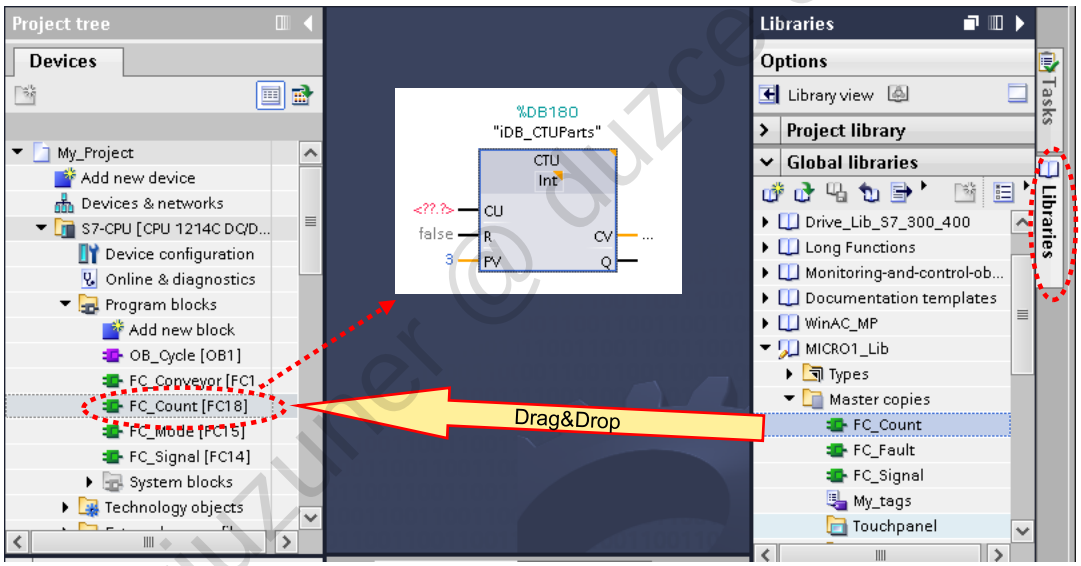
Output CV

The current count is output at output CV. The variable passed to the output must be compatible with the data type of the counter.

Output QU

The status of the up counter can be checked at the output QU. If the current count is greater than or equal to the value of the parameter PV, the output QU has Status '1', otherwise, Status '0'.

8.5.4. Exercise 1: Counting the Transported Parts – Commissioning "FC_Count"



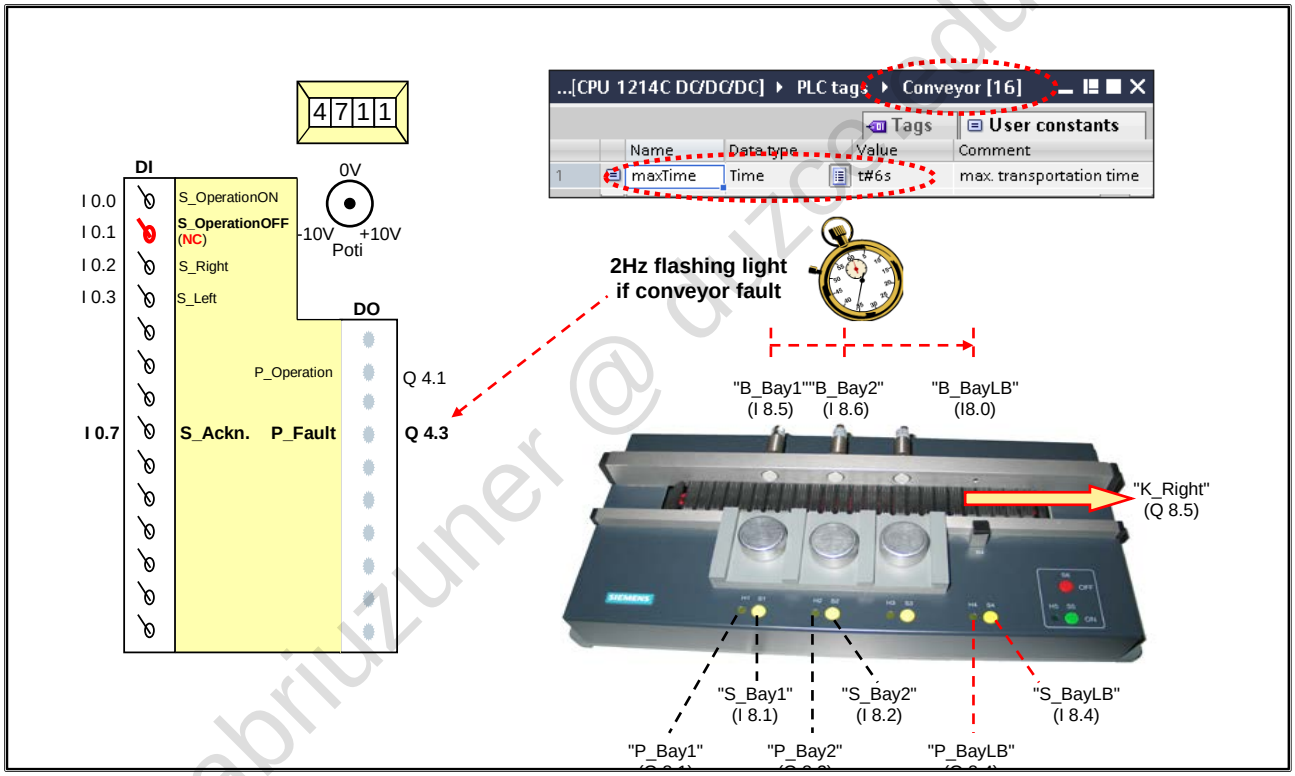
Task

When "P_Operation" (Q4.1) is switched on, the transported parts are to be counted and the ACTUAL quantity is to be stored in the variable "MW_ACT" (MW20). If the ACTUAL quantity has reached the SETPOINT quantity of 3, the indicator light "P_BayLB" (Q8.4) shows a continuous light. If this is not acknowledged via "S_BayLB" (I 8.4), the indicator lights "P_Bay1" (Q8.1) and "P_Bay2" (Q8.2) are dark and no new part transportation can be started.

What to Do

1. From the "MICRO1_Lib" global library, copy the "FC_Count" function into the "Program blocks" container using drag & drop as shown in the picture
2. In "FC_Count", pass the "iDB_CTUParts" data block as a single instance DB to the count function (is automatically created in the System blocks -> Program resources container, see picture)
3. Program the call of "FC_Count" in "OB_Cycle" (OB1). Download all blocks into the CPU and check the program function
4. Correct the "FC_Count" function in such a way that the parts are counted correctly (the "iDB_CTUParts" data block is error-free!)
5. In the blocks "FC_Signal" and "FC_Conveyor" program the required lock-outs
6. Download all modified blocks into the CPU and check the program function
7. Save your project

8.6. Task Description: Monitoring the Transportation time



Function Up Until Now

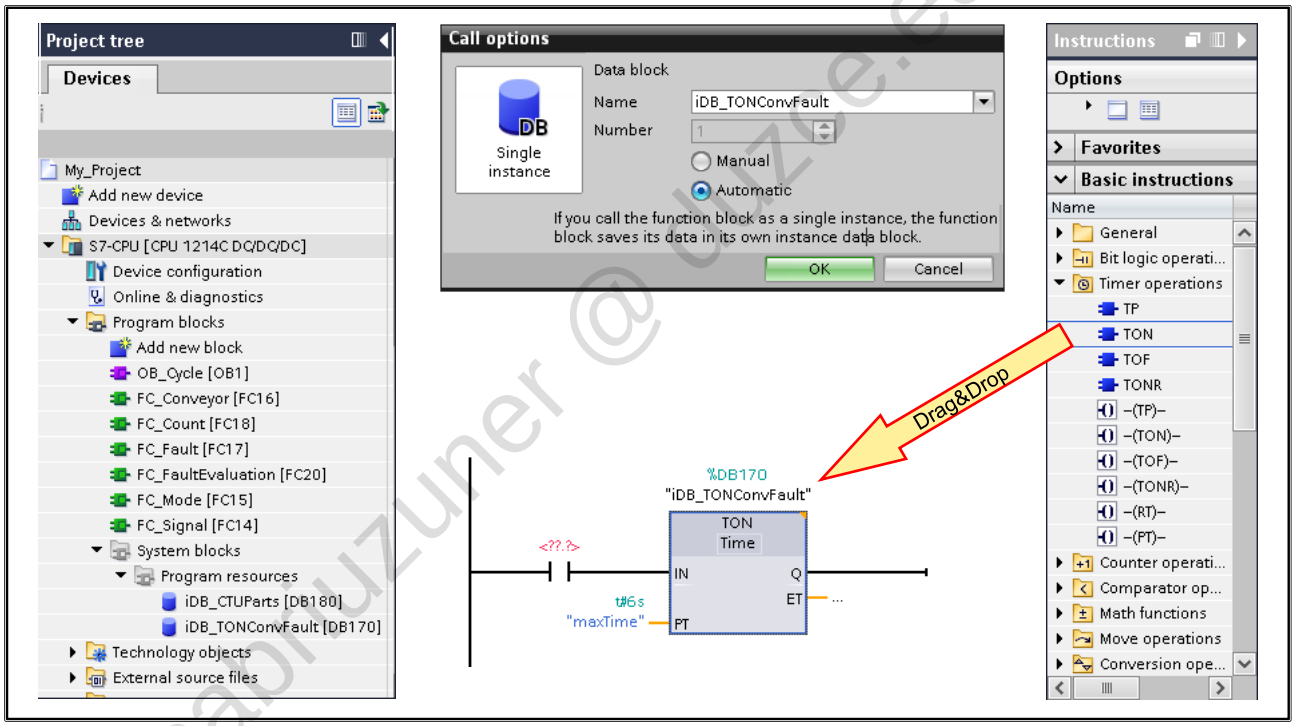
When "P_Operation" (Q4.1) is switched on, parts are transported from Bay 1 or 2 until they are through the light barrier. A transport sequence starts as soon as a part is placed on the conveyor at Bay 1 or 2 and the associated bay pushbutton is pressed. The transport sequence ends as soon as the part has passed the light barrier.

Task Description

The automatic transport sequences are to be monitored for time. 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" (Q 4.3)
- A fault can be acknowledged via the simulator switch "S_Acknowledge" (I 0.7)
- If there is an unacknowledged fault, no new transport sequence can be started

8.6.1. Timer Function TON



Data Type TIME

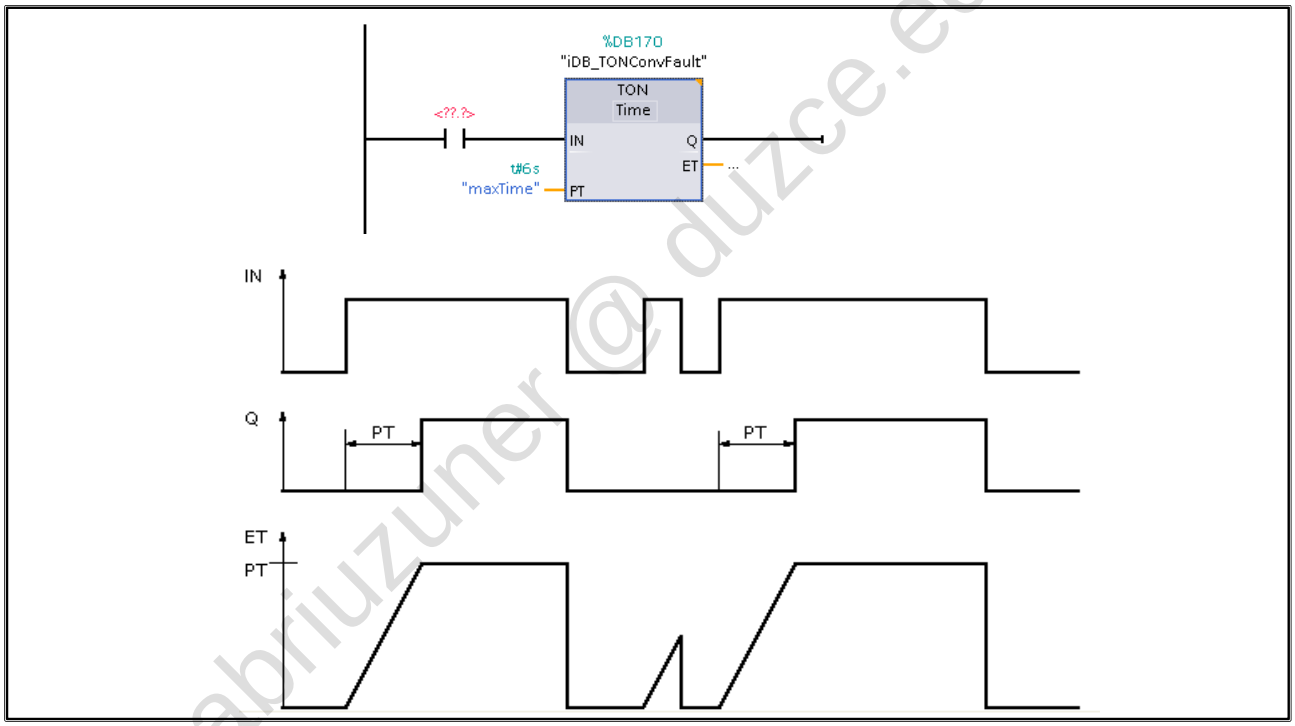
The contents of a variable or constant of the data type TIME is interpreted as an integer number in milliseconds and stored in the memory as a 32-bit integer with sign. The representation contains information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms). Examples:

Length (bits)	Format	Value range	Examples of value input
32	Signed duration	T#-24d20h31m23s648ms to T#+24d20h31m23s647ms	T#10d20h30m20s630ms, TIME#10d20h30m20s630ms

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.

8.6.2. Timer Function TON (ON Delay) Pulse Diagram

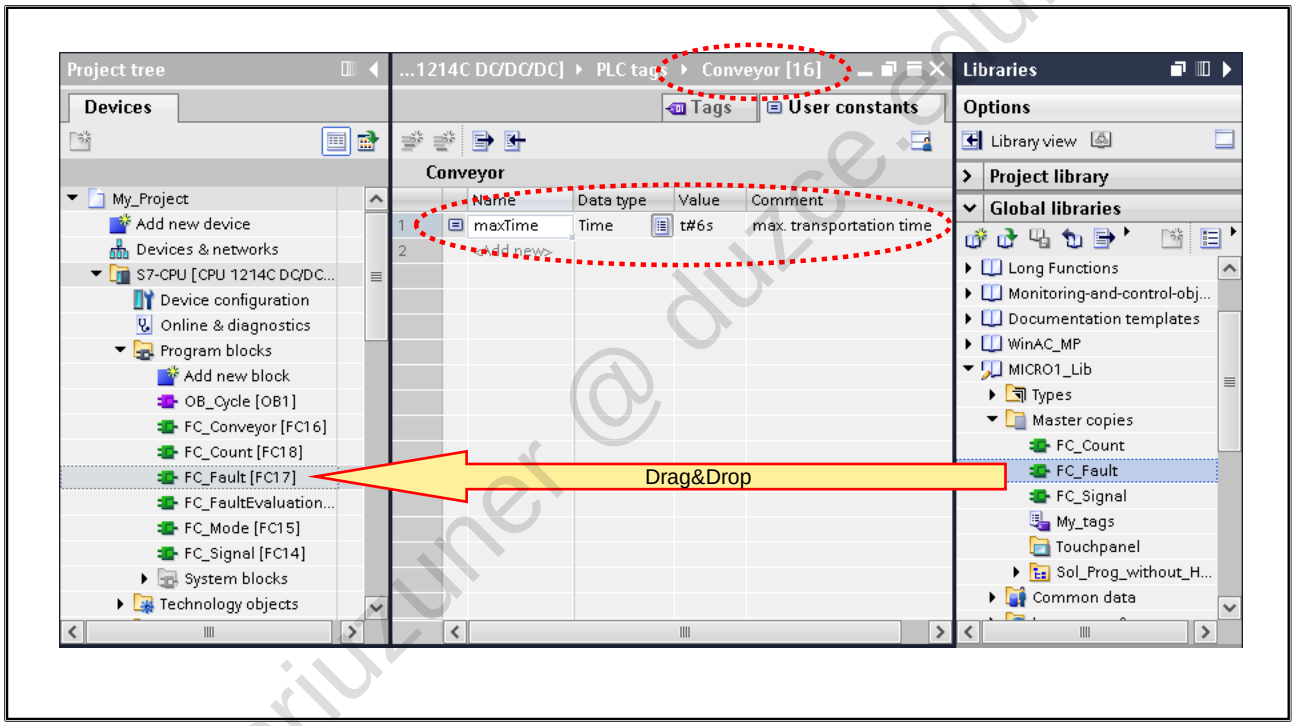


TON

The timer function "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) at input PT has expired and if 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.

Parameter	Data type	Memory area	Description
IN	BOOL	I, Q, M, D, L, P	Start input
PT	TIME	I, Q, M, D, L, P	Duration of the on delay. The value of the PT parameter must be positive.
Q	BOOL	I, Q, M, D, L, P	Operand that is set when the timer PT expires
ET	TIME	I, Q, M, D, L, P	Current time value

8.6.3. Exercise 2: Monitoring the Transportation Time Commissioning "FC_Fault"



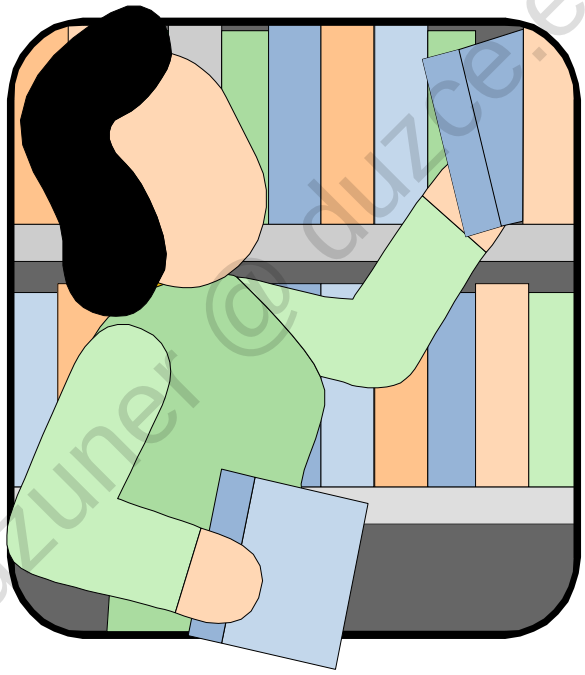
Task

The 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. If a fault is not acknowledged, no new transport sequence can be started.

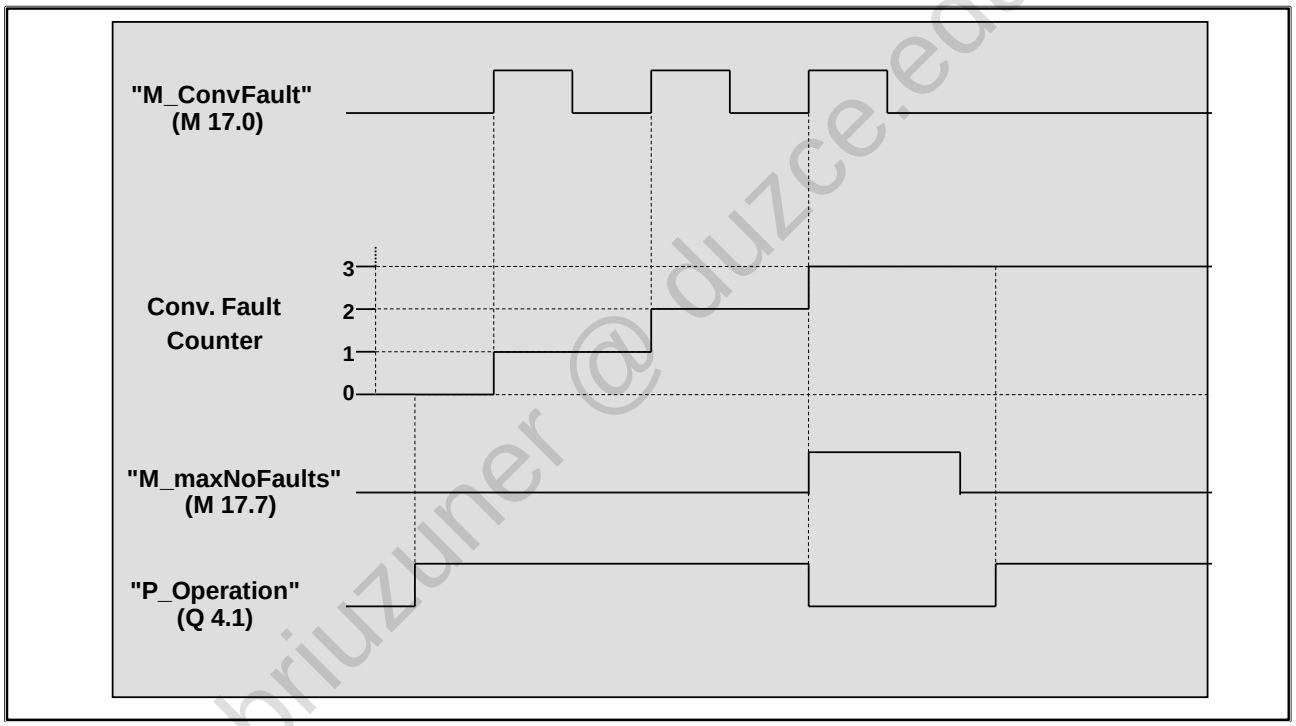
What to Do

1. From the "MICRO1_Lib" global library, copy the "FC_Fault" function into the "Program blocks" container using drag & drop as shown in the picture
2. In the PLC tag table "Conveyor" create the user constant "maxTime" of the data type Time with the value T#6s as shown in the picture
3. In "FC_Fault" pass the "iDB_TONConFault" data block as a single instance DB to the timer function and as the time period the user constant "maxTime"
4. Program the call of "FC_Fault" in "OB_Cycle" (OB1)
5. In the "FC_Conveyor" block program the required switching off of the conveyor motor when there is a conveyor fault
6. In "FC_Signal" program the relevant lock-outs so that the indicator lights "P_Bay1" (Q8.1) and "P_Bay2" (Q8.2) show the required behavior
7. Download all modified blocks into the CPU and check the program function
8. Correct "FC_Fault" in such a way that the monitoring function is fulfilled as required
9. Save your project

8.7. Additional Information



8.7.1. Additional Exercise 3: Counting the Conveyor Faults - Expanding "FC_Fault"



Function Up Until Now

When "P_Operation" (Q4.1) 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".

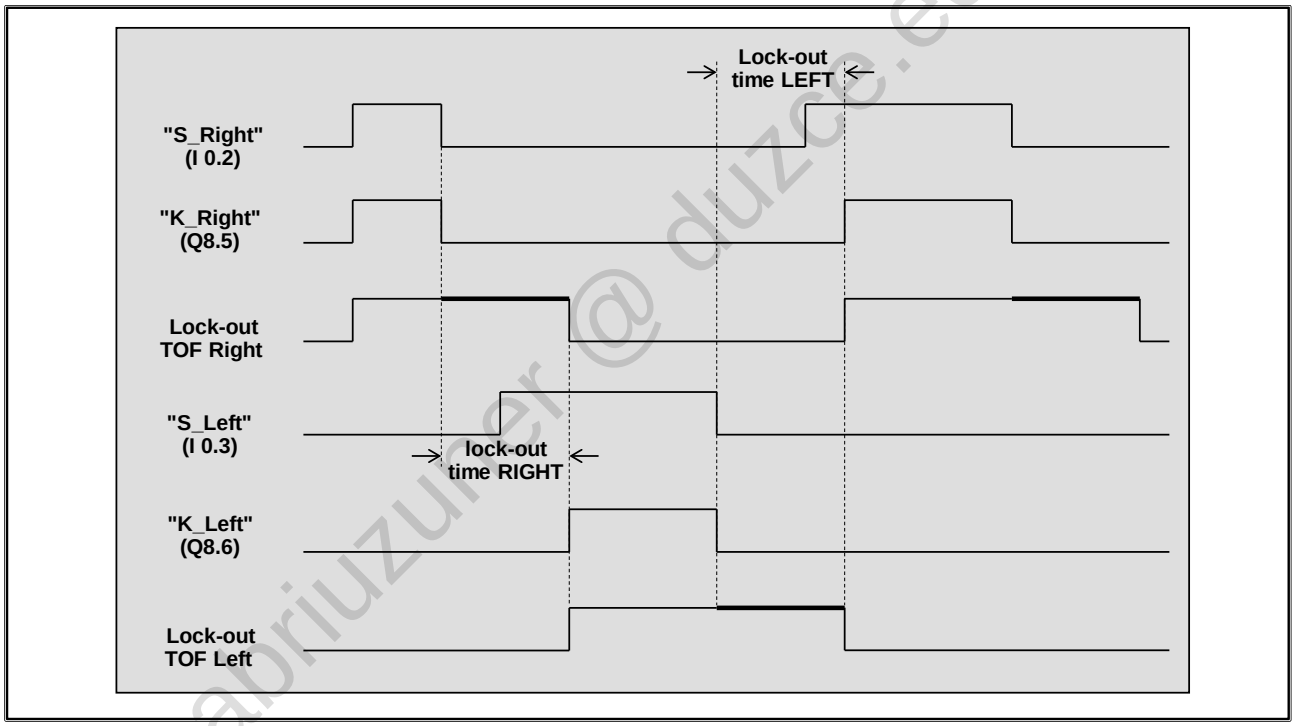
Task

When "P_Operation" (Q4.1) is switched on, the conveyor faults are to be counted. After 3 conveyor faults, "P_Operation" (Q4.1) 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" (Q4.1) 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_ConFault" (M17.0) = "1")
2. In "FC_Fault", pass any bit memory you like to the counter output Q
3. In "FC_Mode", program the switching off (reset) of "P_Operation" (Q4.1) after three conveyor faults. For this, use the bit memory which you passed to the counter output Q in "FC_Fault"
4. Save your project

8.7.2. Additional Exercise 4: Timely Lock-out of the Conveyor Motor Jogging



Function Up Until Now

When "P_Operation" (Q4.1) 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

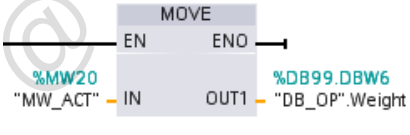
To avoid too great a load change, it should only be possible to jog the conveyor motor in the opposite direction after it has been jogged to the RIGHT or to the LEFT after a lock-out time of 2 seconds (see picture). If, for example, the motor has been jogged to the RIGHT, then it can only be jogged back to the LEFT after the lock-out time of 2 seconds has expired.

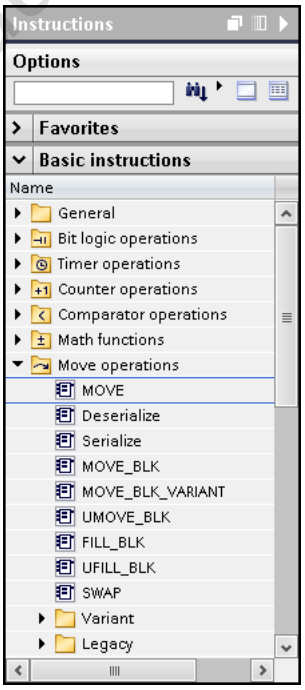
What to Do

1. In "FC_Conveyor", program two TOF timers (Off Delay) as the lock-out timers RIGHT and LEFT and assign one bit memory each to the Timer result Q
2. Gate these bit memories to the jog conditions
3. Download the modified "FC_Conveyor" into the CPU and check the program function
4. Save your project

8.7.3. Move Operations: MOVE

Parameter	Data type	Memory area	Description
EN	BOOL	I, Q, M, D, L	Enable input
ENO	BOOL	I, Q, M, D, L	Enable Output
IN	All elementary data types, DTL, STRUCT, ARRAY	I, Q, M, D, L or constant	Source value
OUT1	All elementary data types, DTL, STRUCT, ARRAY	I, Q, M, D, L	Destination address



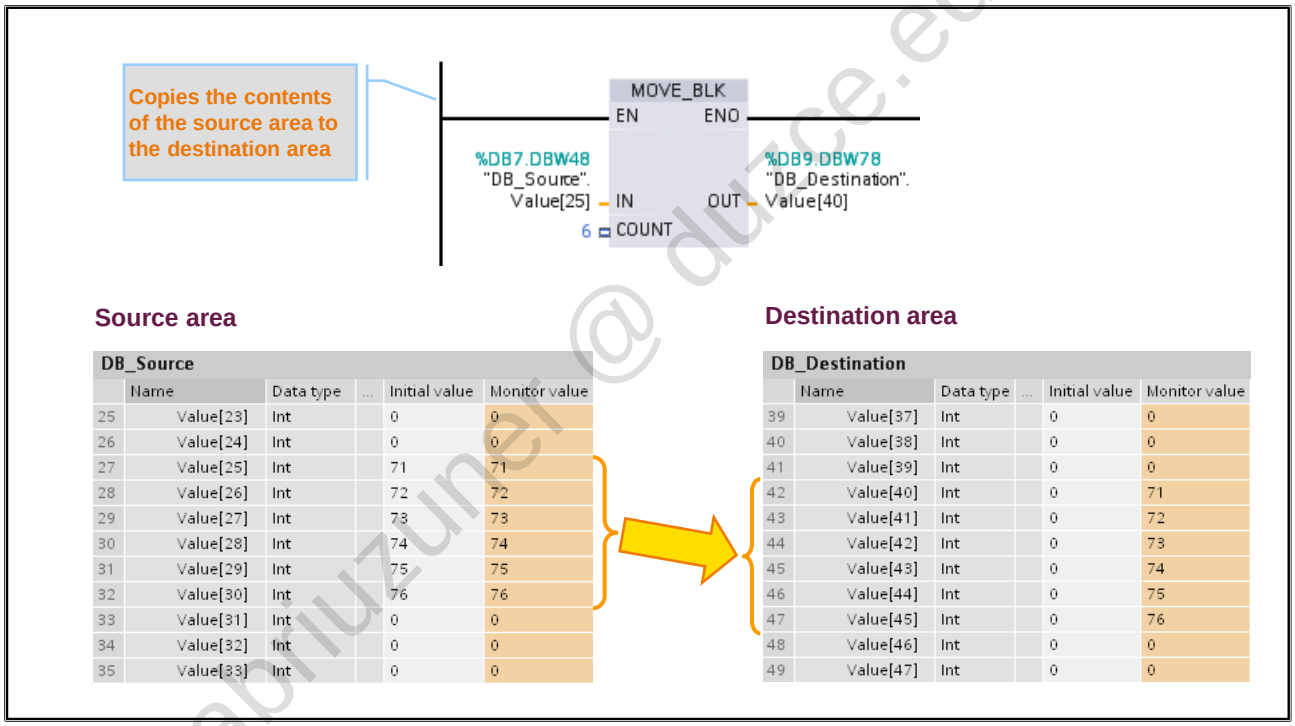


MOVE

You use the "MOVE - Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of ascending address.

The operation is only executed if the signal state is "1" at the enable input EN. In this case, the ENO output also has signal state "1".

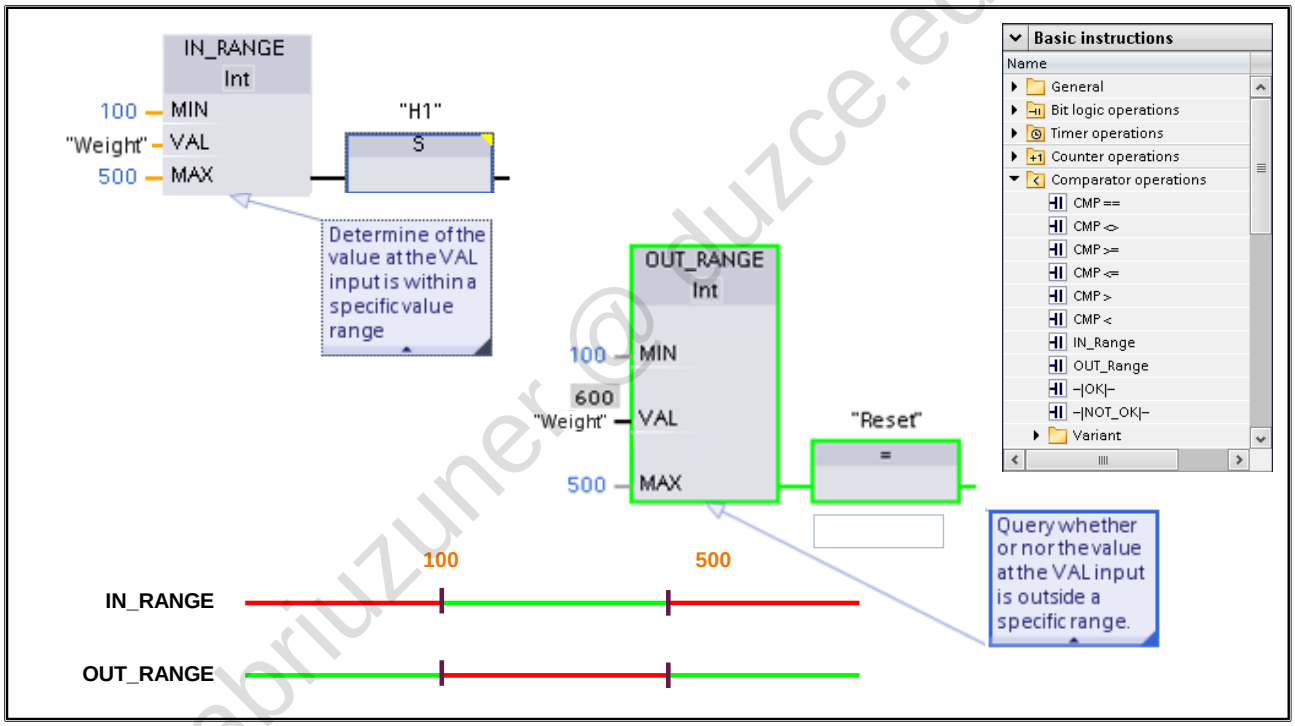
8.7.4. Move Operations: MOVE_BLK



MOVE_BLK

You can use the "Move block" instruction to move (copy) the content of a memory area (source area) to another memory area (destination area). The number of elements to be moved (copied) to the destination area is specified at input COUNT. The width of the elements to be moved (copied) is defined by the width of the element at input. The copy operation is always made in the direction of ascending address.

8.7.5. 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. When the query is processed, the "Value within range" instruction compares the value at the VAL input with the values of the MIN and MAX inputs and sends the result to the box output. If the value at the VAL input fulfills the comparison MIN <= VAL or VAL <= MAX, the box output has the signal state "1". If the comparison is not fulfilled, the box output has the signal state "0".

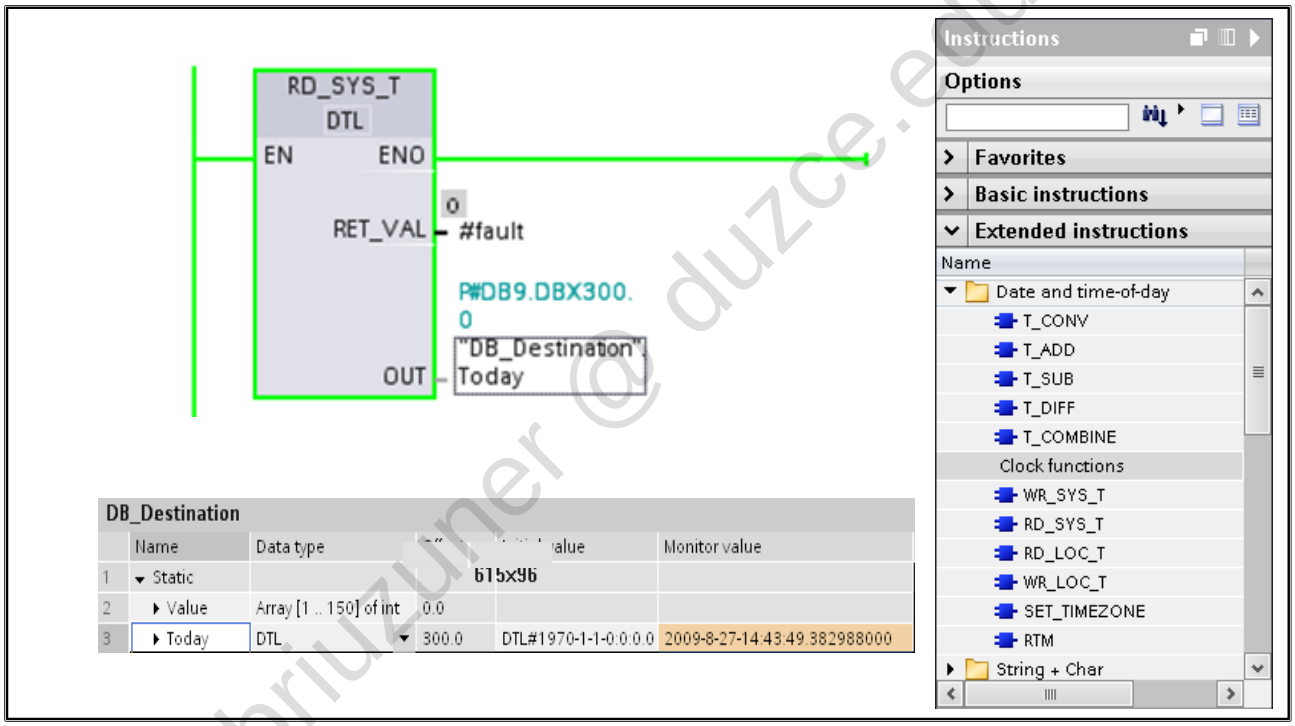
The comparison function can only be executed if the values to be compared are of the same data type and the box output is interconnected.

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. When the query is processed, the "Value outside range" instruction compares the value at the VAL input to the values of the MIN and MAX inputs and sends the result to the box output. If the value at the VAL input fulfills the comparison MIN > VAL or VAL > MAX, the box output has signal state "1". The box output returns the signal state "0", if the value at input VAL does not satisfy the MIN > VAL or VAL > MAX condition.

The comparison function can only be executed if the values to be compared are of the same data type and the box output is interconnected.

8.7.6. Date and Time-of-day: RD_SYS_T



RD_SYS_T

You use the RD_SYS_T instruction to read the current date and current time-of-day of the CPU clock. The read dates are output in DTL format at the OUT output parameter of the instruction. The provided value does not include information about the local time zone or daylight saving time. You can query whether errors have occurred during execution of the instruction in the RET_VAL output.

Byte	Component	Data type	Value range
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

Contents

9

9. Functions and Function Blocks

9.1. Objectives

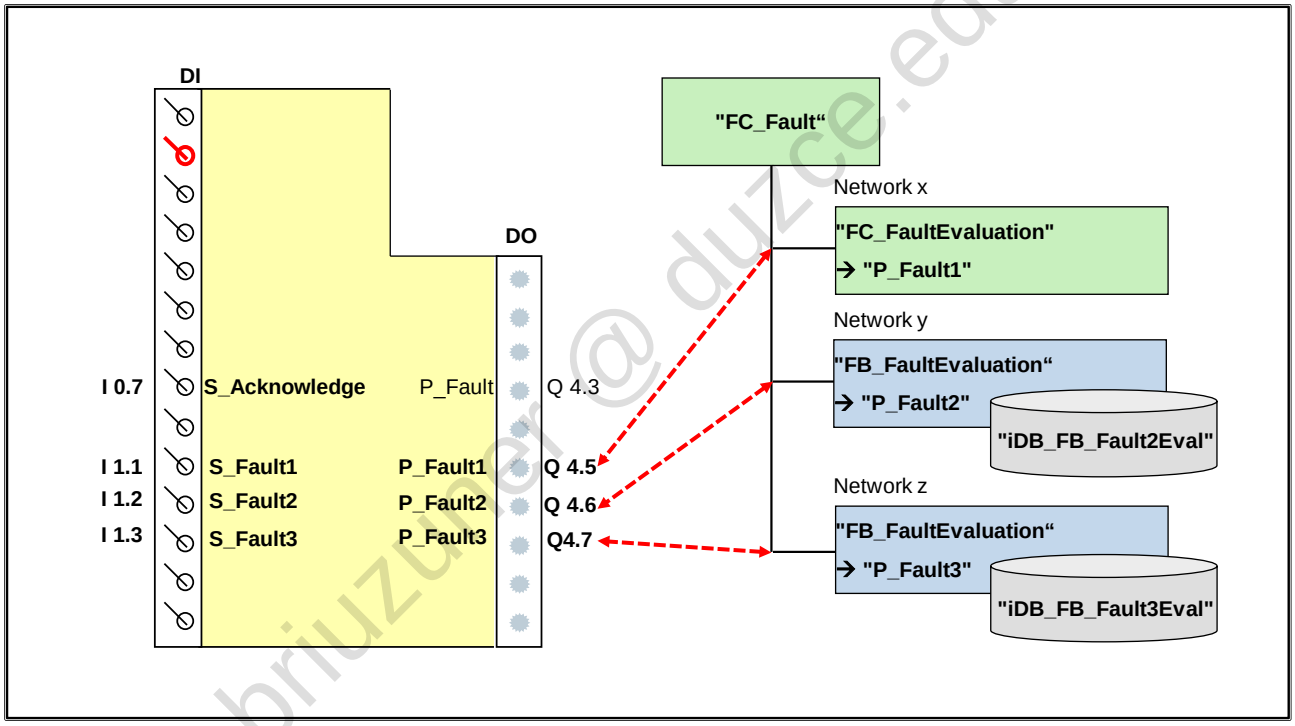
At the end of the chapter the participant will...

- … be familiar with the purpose of temporary variables
- ... be familiar with the purpose of parameter-assignable blocks
- ... be able to program parameter-assignable functions and their calls
- ... know the difference between Functions (FCs) and Function blocks (FBs)
- ... be familiar with the purpose of static variables
- ... be able to declare static variables and apply them in the program
- be able to program parameter-assignable blocks and call them

Objectives

In this chapter, the logic (code) blocks "Function" and "Function Block", as well as their features are presented.

9.2. Task Description: Evaluating Fault Messages with Re-usable Logic (Code) Blocks

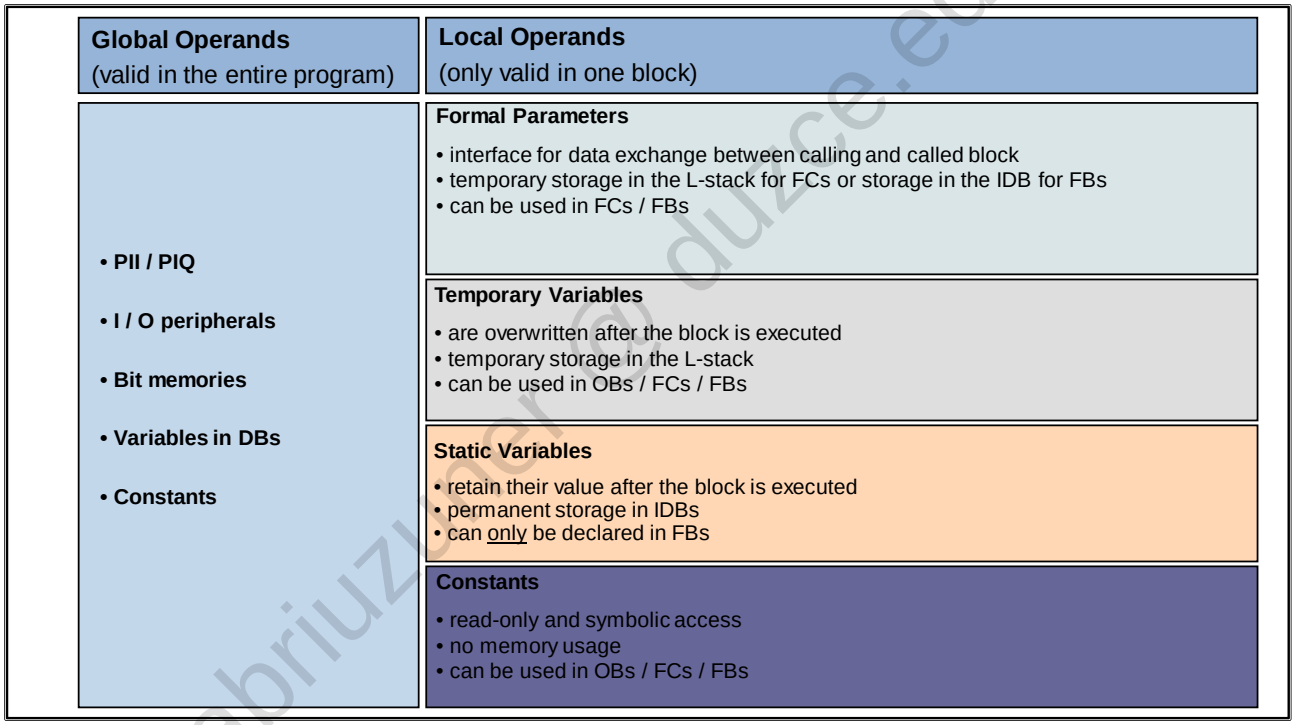


Task Description

Program a reusable function or function block for fault evaluation:

If a fault is triggered at one of the inputs I 1.1 to I 1.3, the associated LED Q4.5 to Q4.7 begins to flash. A group acknowledgement of all fault messages can be triggered via the input I 1.0 ("S_Acknowledge"). If, in the meantime, one of the messages is still pending, the LED switches to continuous light, otherwise it goes out.

9.3. Operand Overview for S7-1200



General

In the program up until now, the inputs and outputs were addressed with their actual operands. The blocks were not parameter-assignable.

This approach would be used, for example, for the creation of a program that would be used once for a special machine.

For frequently re-occurring functions in larger systems, universally usable, parameter-assignable blocks (FC, FB) are created. These have formal input and output parameters (formal parameters) which are assigned actual parameters when the block is called.

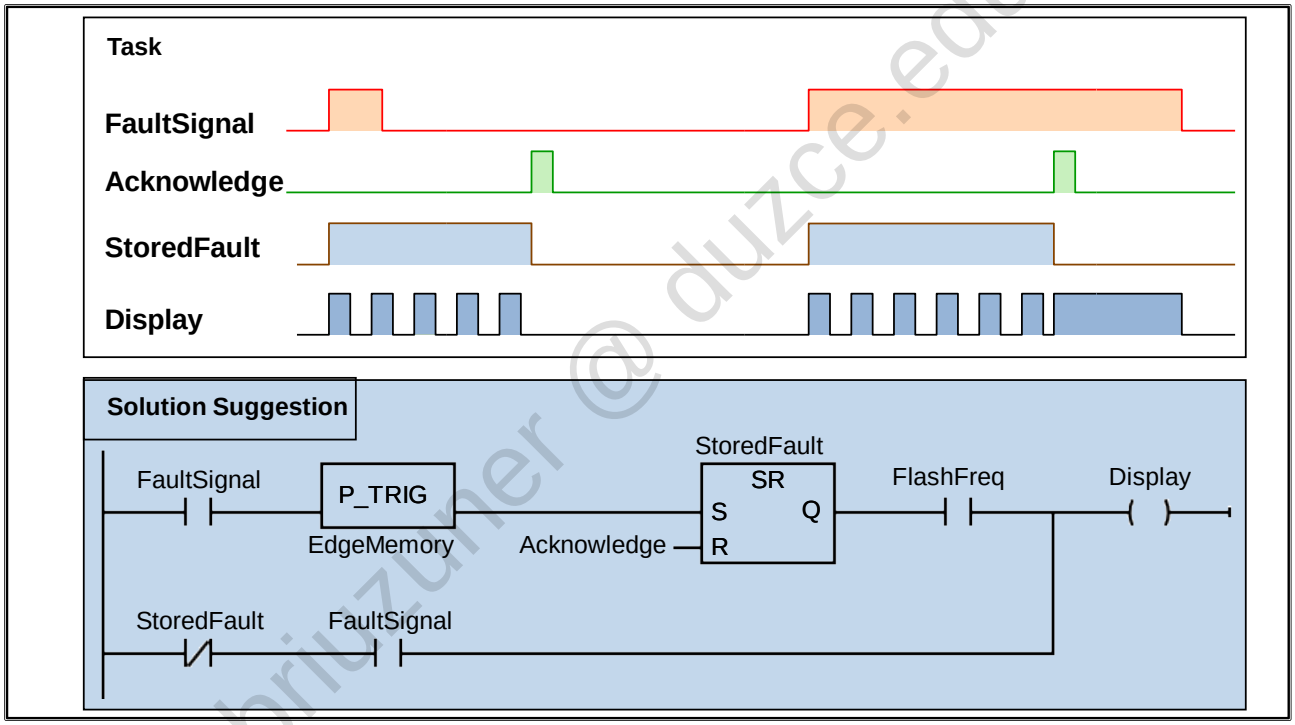
The adaptation of the block functionality to the hardware occurs through the assignment (parameterization) during the block call; the "inner life" of the block remains unchanged.

Local Operands

Up until now you used global operands (bit memories and data blocks) to save production data, for example. Instead of global operands, data can be stored in local operands which can be divided into four different categories:

- Formal parameters: Formal parameters form the interface between calling and called block.
- Temporary Variables: Temporary variables are variables that are stored only while the block is being executed. They can be declared in all blocks (OB, FC, FB).
- Static Variables: If the data is to be retained even after the block is executed, the data must be stored in static variables. Static variables can only be declared in function blocks.
- Constants: Constants are fixed values which have a read-only access and which do not take up any memory space.

9.4. Example of a Fault Display



Description

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 "Fault_Input" the "Display" shows a 2Hz flashing light.

When the fault is confirmed via "Acknowledge" and "Fault_Input" still exists, the "Display" switches to a continuous light. When "Fault_Input" no longer exists, the display (light) does dark.

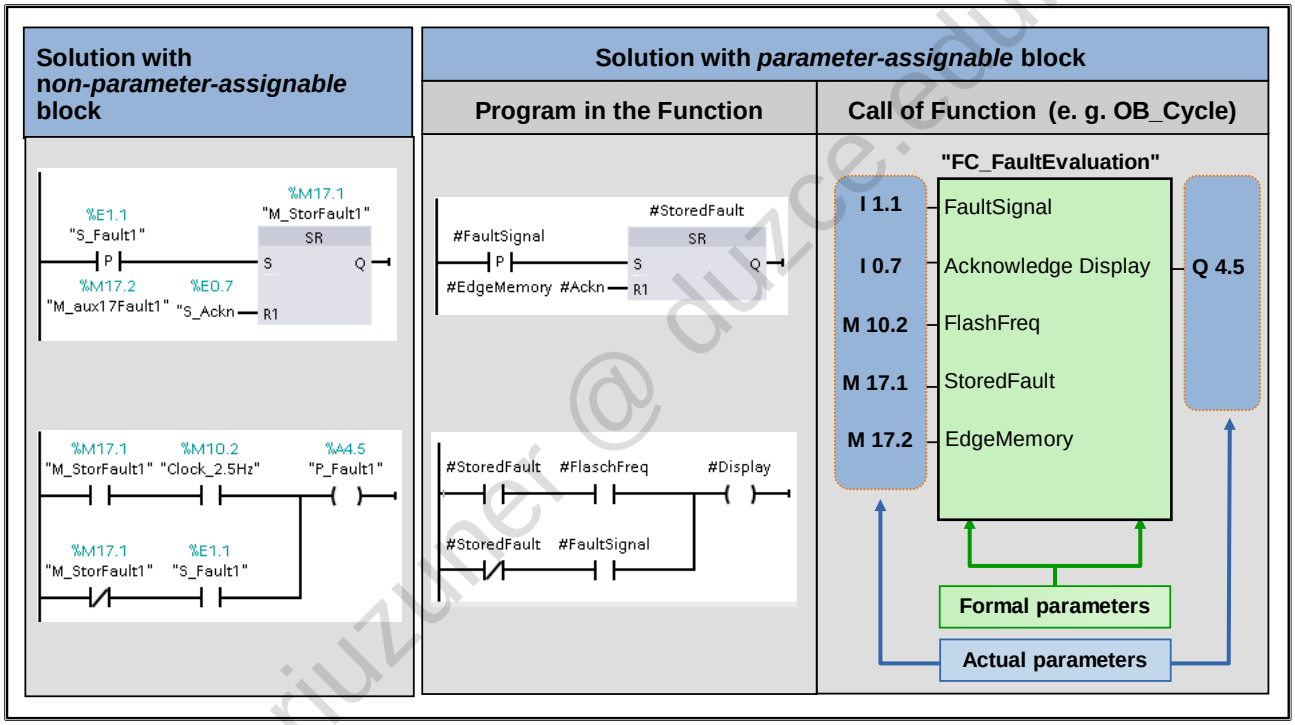
Program

An edge evaluation of "Fault_Input" is carried out because the "Stored_Fault" would otherwise immediately be set again after an acknowledgement and a still existing "Fault_Input" thus making the "Display" flash once again.

When "Acknowledge" has not yet occurred ("Stored_Fault" is still set), the upper path (AND logic operation) with the "Flash_Freq" causes the "Display" to flash.

When "Acknowledge" has already occurred ("Stored_Fault" is no longer set), but "Fault_Input" still exists, the lower path (AND logic operation) causes the "Display" to have a continuous light.

9.5. 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 must 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 parameter #FaultSignal is replaced with the actual parameter passed during the call.

If, during the call of the block, the input "S_Fault1“ was passed as the actual parameter for the parameter #FaultSignal, the statement "A #FaultSignal" becomes the statement "A "S_Fault1"" during runtime.

Parameter-assignability

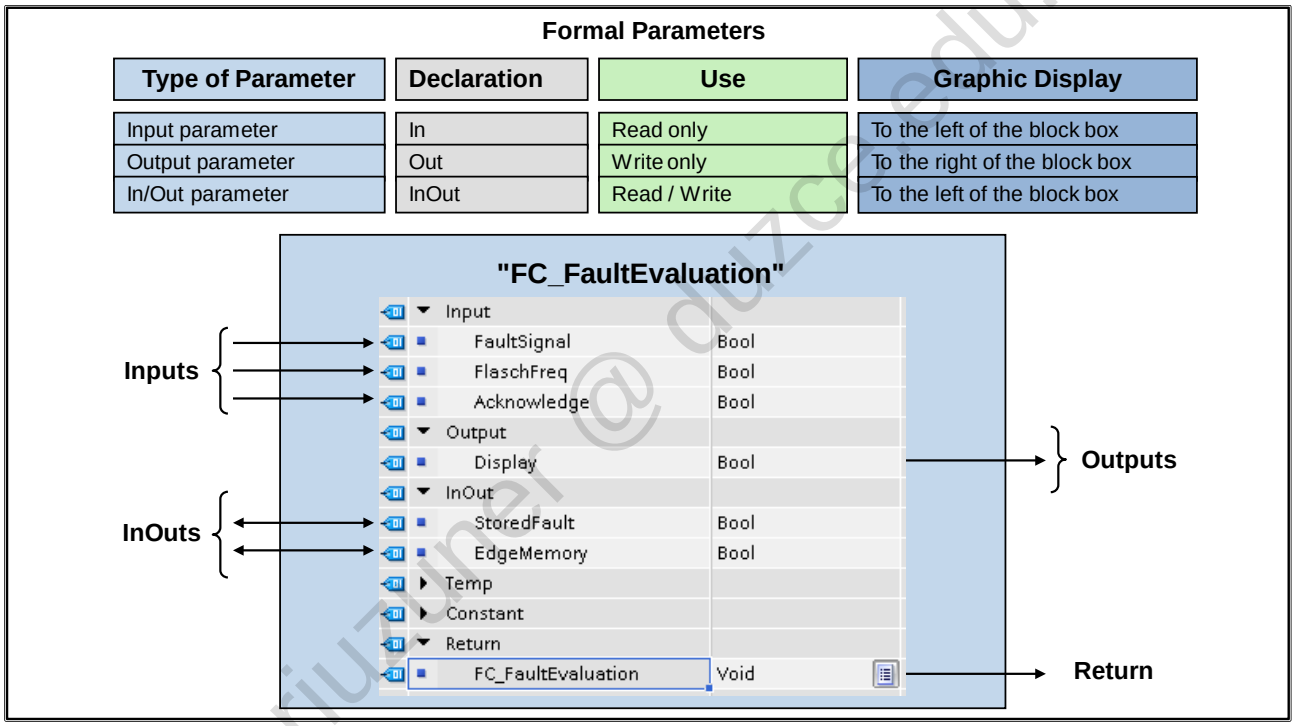
You can program FC or FB blocks as parameter-assignable. You cannot program organization blocks as parameter-assignable because they are called directly by the operating system. As no block call takes place in the user program, it is not possible to pass actual operands to an OB.

Our Example

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

The "FC_FaultEvaluation" is then called twice for the two different fault evaluations and is assigned a different actual operand each time.

9.5.1. Declaration of Formal Parameters



Formal Operands

Before you can create the program in the parameter-assignable block, you must 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 must 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.

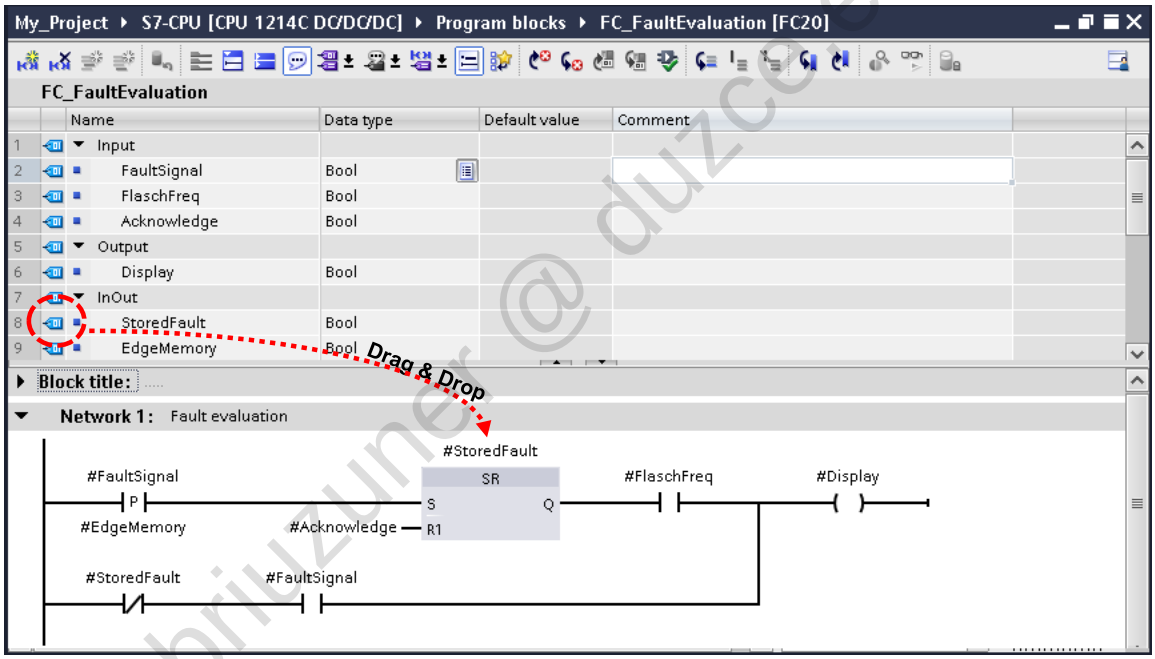
The variables Temp and Constant are – even though they are listed in the declaration section of the interface – not components of the block interface, because they do not become visible when the block is called.

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 must be updated or corrected in all calling blocks.

9.5.2. 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.

When you write a program in LAD or FBD 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

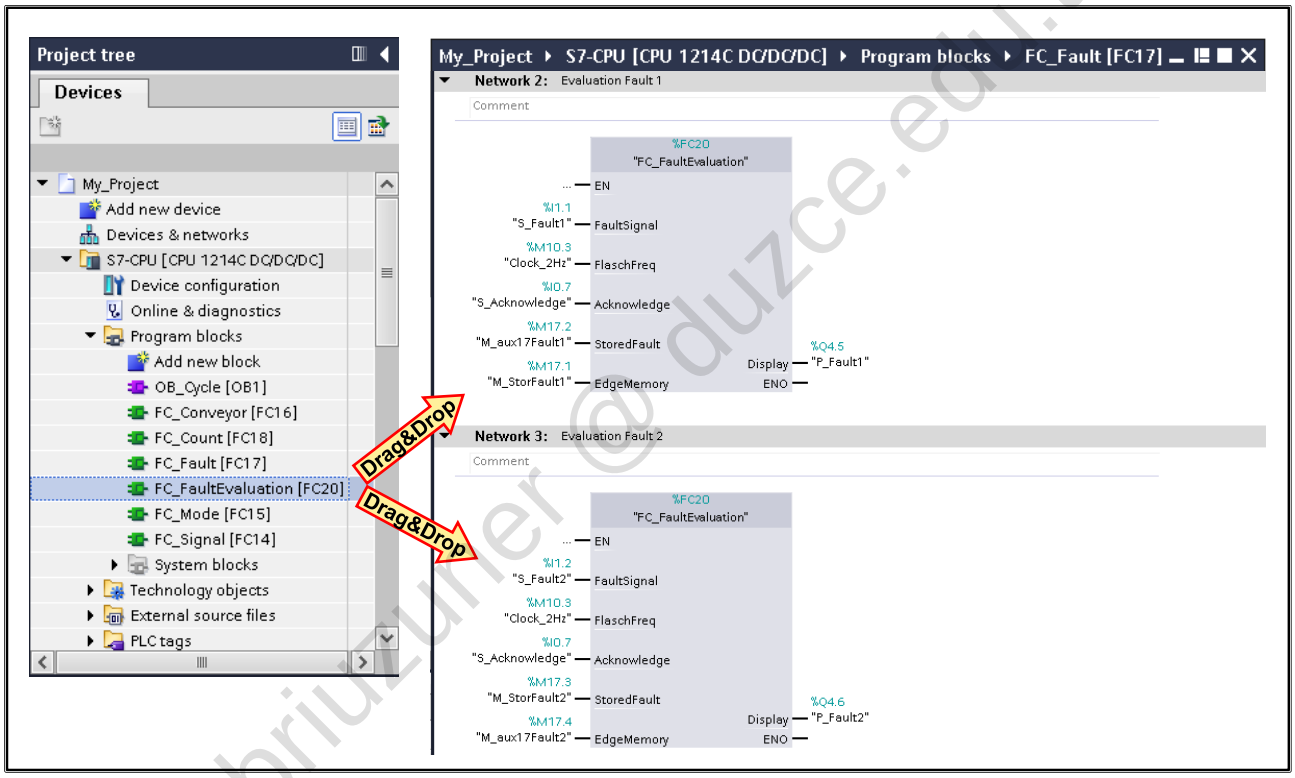
Symbols

1. If you use a symbolic name when you edit a block, the Editor first 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
2. 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
3. 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

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

9.5.3. Calling a Parameter-assignable Block



Block Call

The block can be called by dragging it from the "Program blocks" container & 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

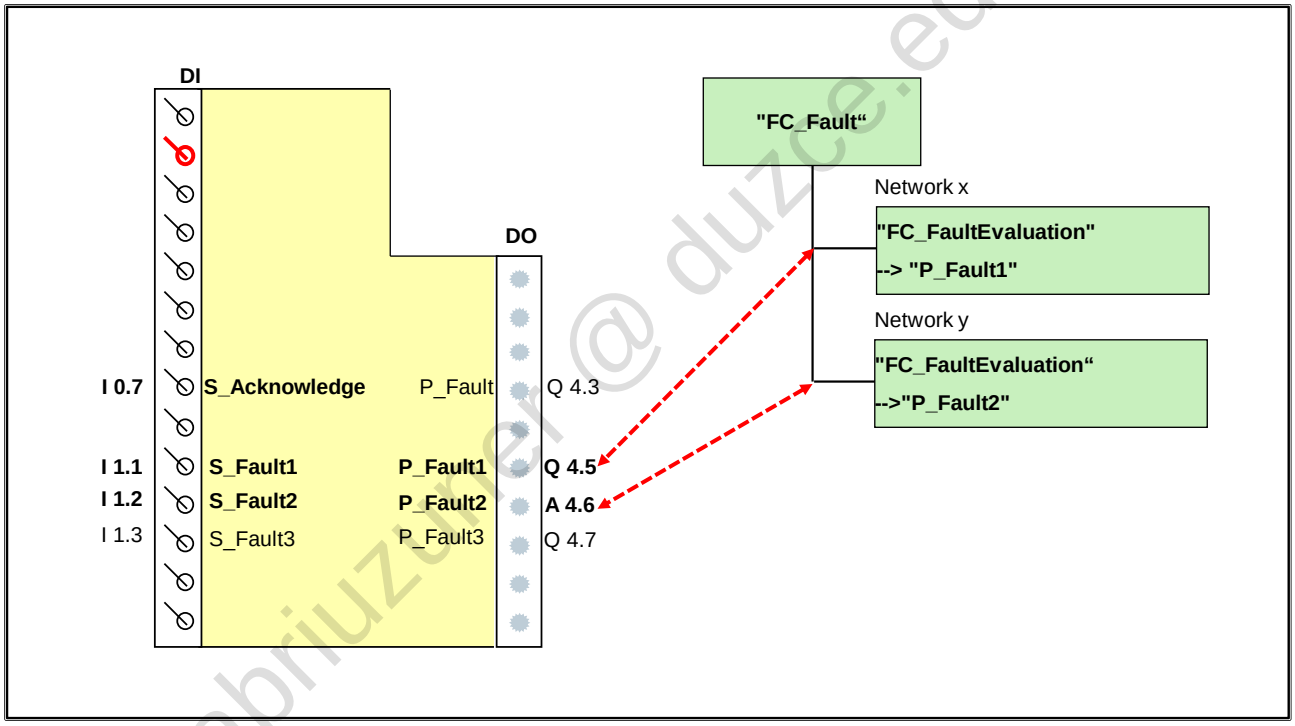
All global and local operands whose data type corresponds to the formal parameters of the called block can be passed as actual parameters.

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 this is however only possible with limitations. It is dealt with in greater detail in another more advanced course.

9.6. Task Description: Fault Evaluation by means of a Function (FC)



Task Description

You are to program the reusable function "FC_FaultEvaluation" to evaluate fault signals.

Fault signals:

- "S_Fault1" (I 1.1)
- "S_Fault2" (I 1.2).

Call the function twice in "FC_Fault" to evaluate two faults.

The faults are displayed on the LEDs

- "P_Fault1" (Q4.5)
- "P_Fault2" (Q4.6)

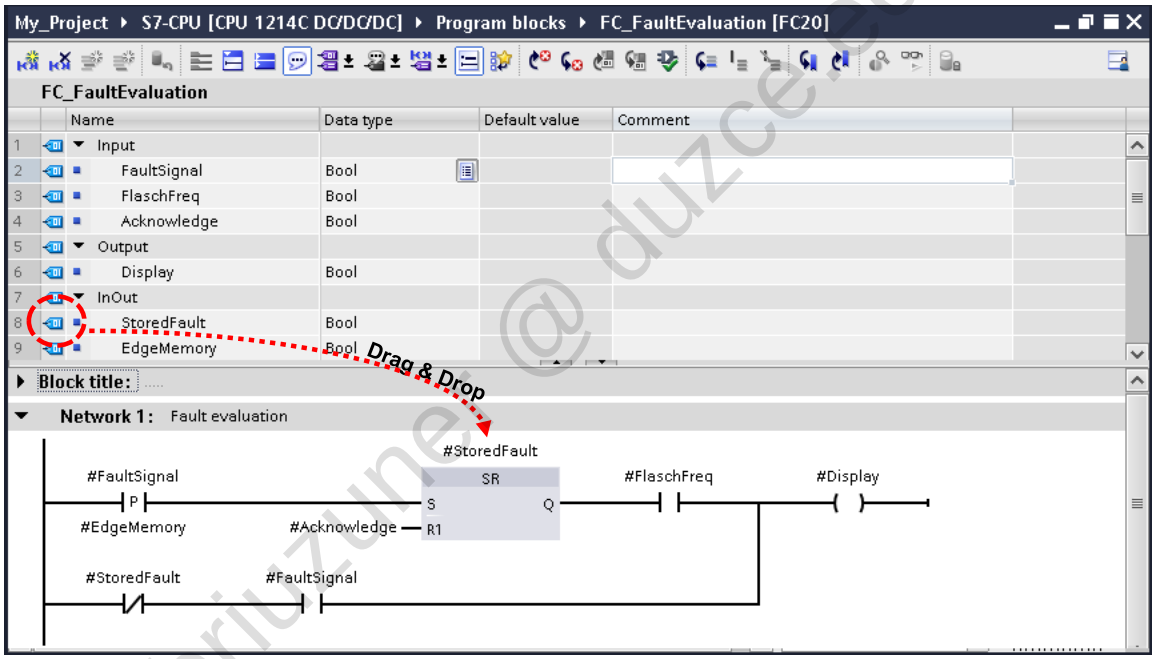
Program

An edge evaluation of #FaultSignal is carried out since the #StoredFault would otherwise immediately be set again after an acknowledgement and a still existing #FaultSignal thus making the #Display flash once again.

When #Acknowledge has not yet occurred (#StoredFault is still set), the upper path (AND logic operation) with the #FlashFreq causes the #Display to flash.

When #Acknowledge has already occurred (#StoredFault is no longer set), but #FaultSignal still exists, the lower path (AND logic operation) causes the #Display to have a continuous light.

9.6.1. Exercise 1: Programming the "FC_FaultEvaluation"



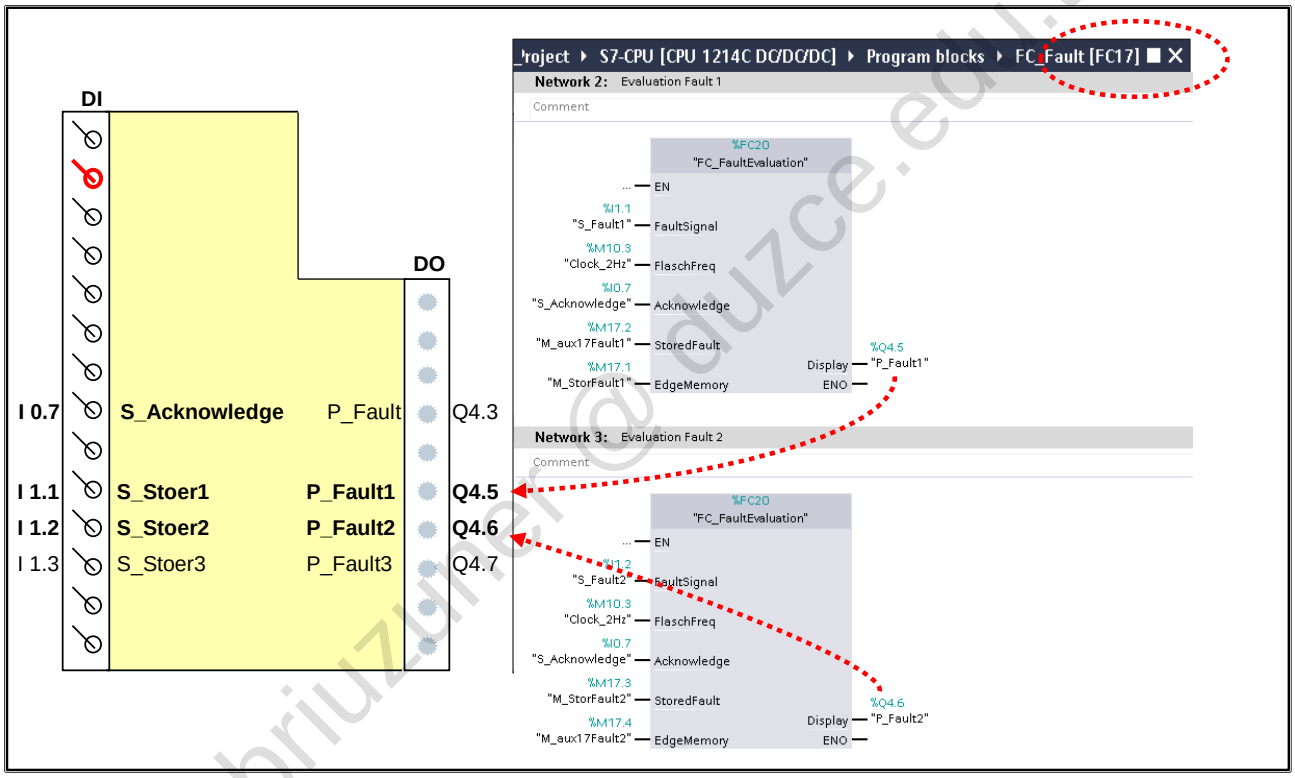
Task

Create the program for the fault evaluation in the parameter-assignable block "FaultEvaluation".

What to Do

1. Insert the "FC_FaultEvaluation" block into 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 your project

9.6.2. Exercise 2: Calling and Assigning the "FC_FaultEvaluation"



Task

In addition to the conveyor model indicator lights, two process faults (two switches on the simulator) are also to be evaluated or displayed by means of LED on the simulator. For this, program two calls of the new "FC_FaultEvaluation" block and parameterize it with the actual parameters shown in the picture.

What to Do

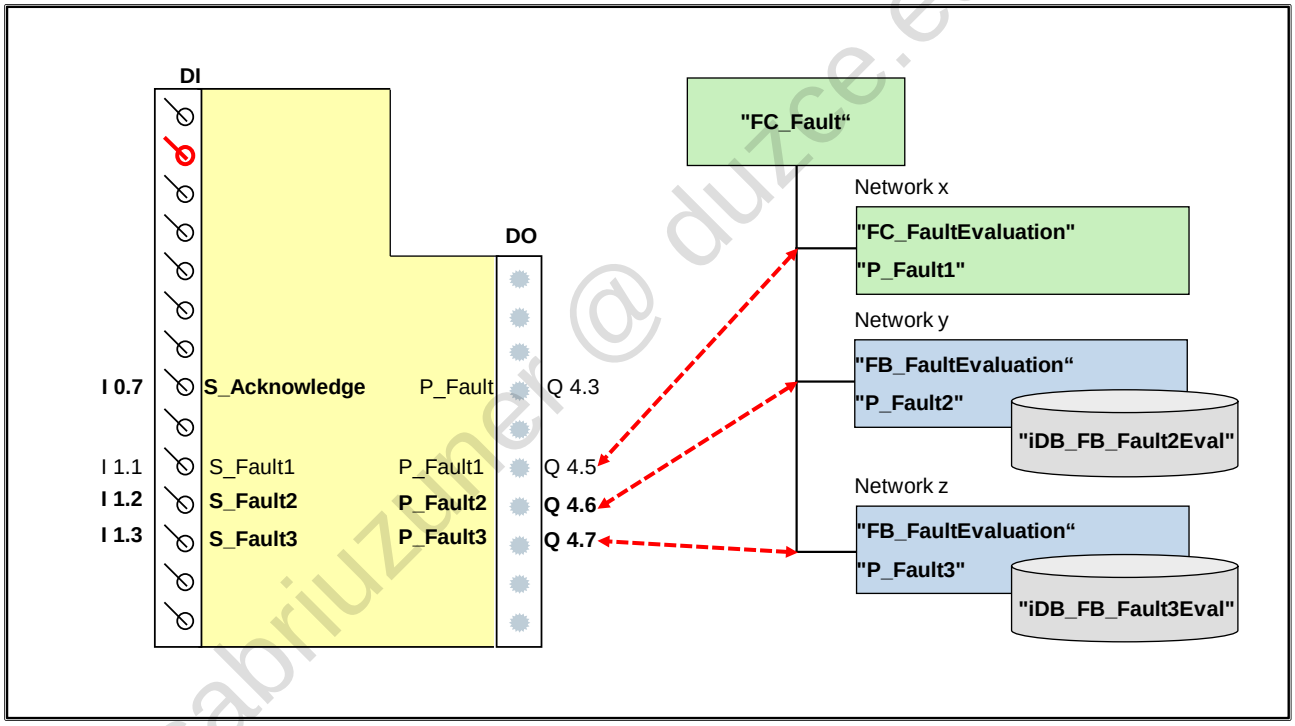
1. In the "FC_Fault" block, program the two calls of the new "FC_FaultEvaluation block as shown in the picture
2. Download the modified "FC_Fault" into the CPU and test your program to see if it carries out the described functions
3. Save your project

Note

The MB10 memory byte was already parameterized as the clock memory byte in the device configuration.

The "Clock_2Hz" (M10.3) bit memory 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

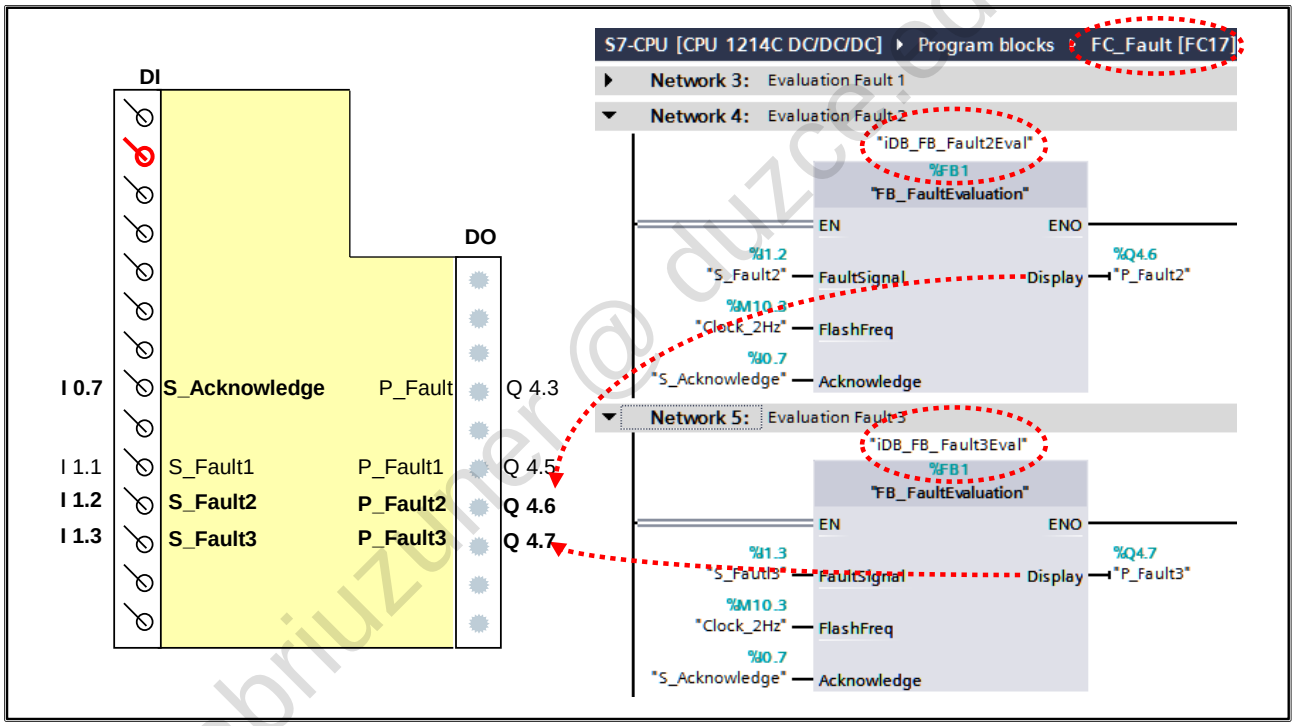
An additional process fault (switch on the simulator) is to be evaluated. The easiest way to do this would be to program another "FC_FaultEvaluation" call.

However, to make use of the given advantages of an FB solution, you are to program a parameter-assignable "FB_FaultEvaluation" for the evaluation of this process fault:

Static variables are to be used here to store the #EdgeMemory and the #StoredFault instead of in/out parameters.

Compared to the "FC_FaultEvaluation" function, the actual user program remains unchanged.

9.7.1. Instantiating Function Blocks



Special Features

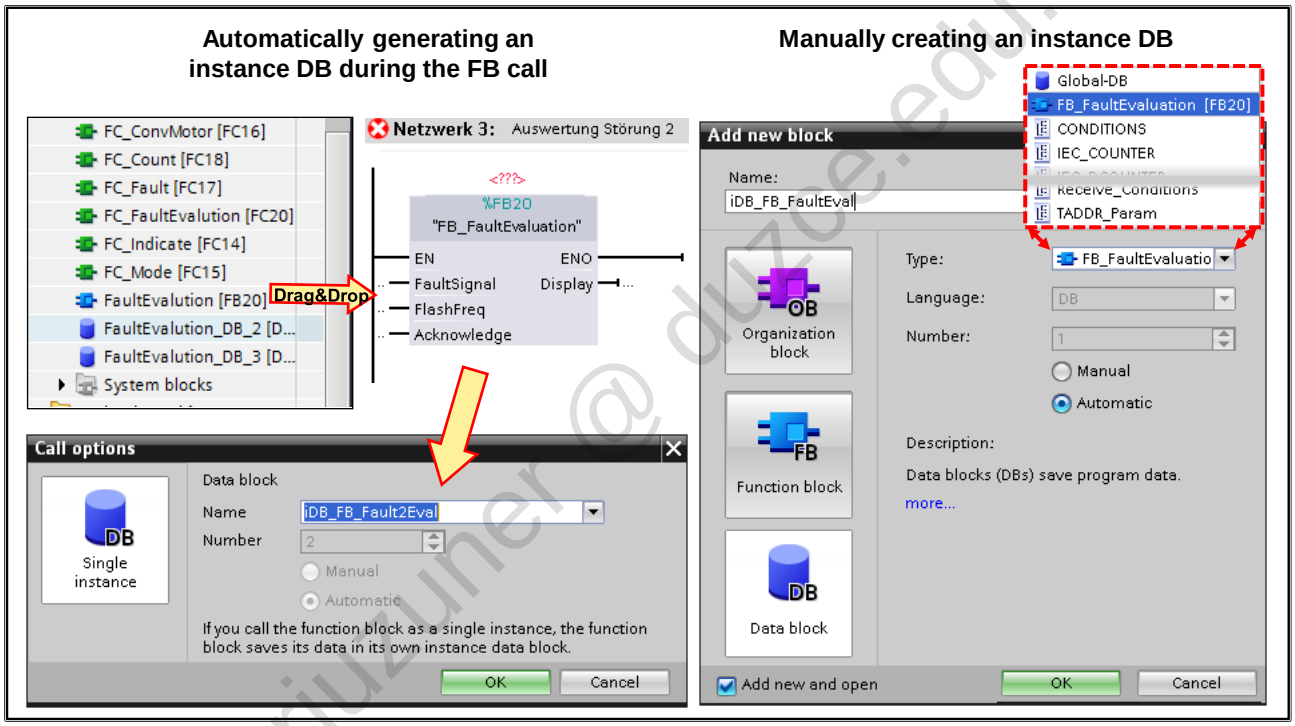
Unlike functions (FCs), function blocks (FBs) have a (recall) memory. That means that a local data block is assigned to the function block. This data block is known as an instance data block. When you call an FB, you must also specify the Instance-DB which is then automatically used as an instance for 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. 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. Generating Instance Data Blocks



Generating

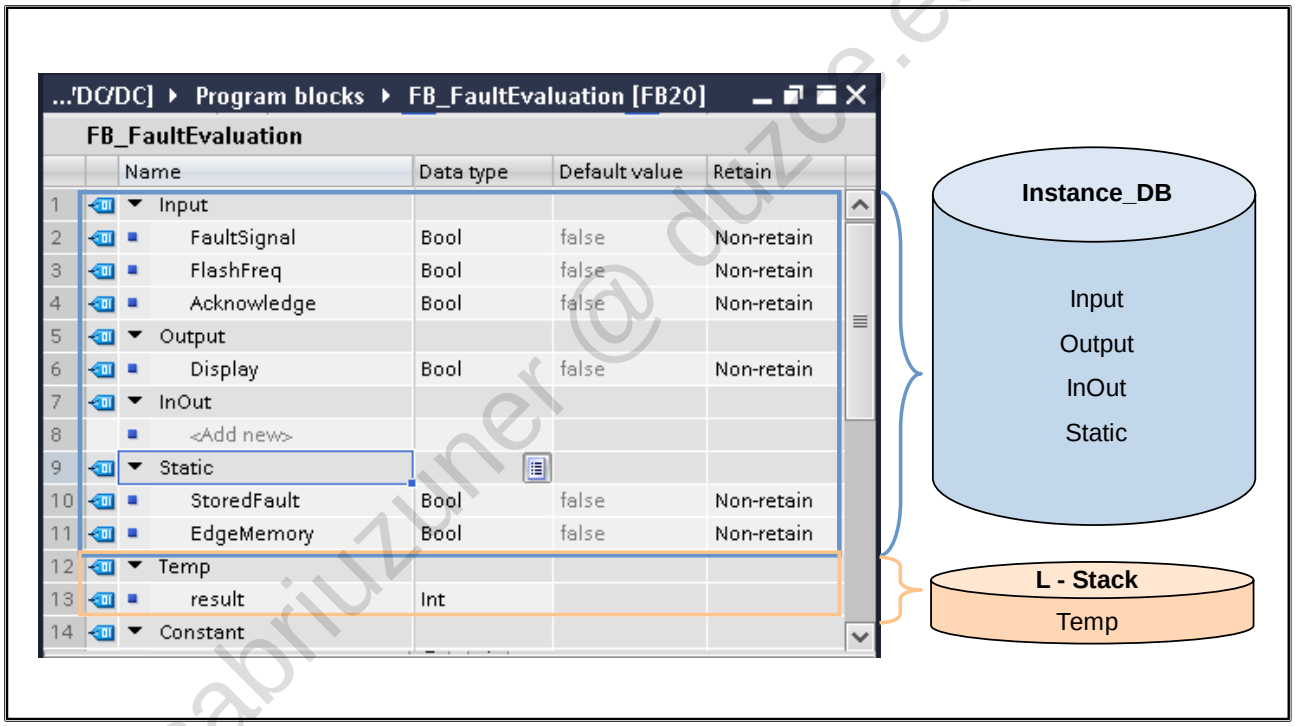
There are two ways of generating an instance data block:

- When you call an FB, you specify with which instance DB the FB is to work.
  A dialog opens in which the symbolic name and, if desired, a manual number of the instance DB can be preset
- When you create a new DB, you select the Type "Function block XY"

An instance DB can only be assigned to one FB. For each call, however, an FB can be assigned a different instance DB.

Caution!
If you modify the FB (by adding additional parameters or static variables), you must then also generate the instance DB again.

9.7.3. Block Interface for the FB



Parameters

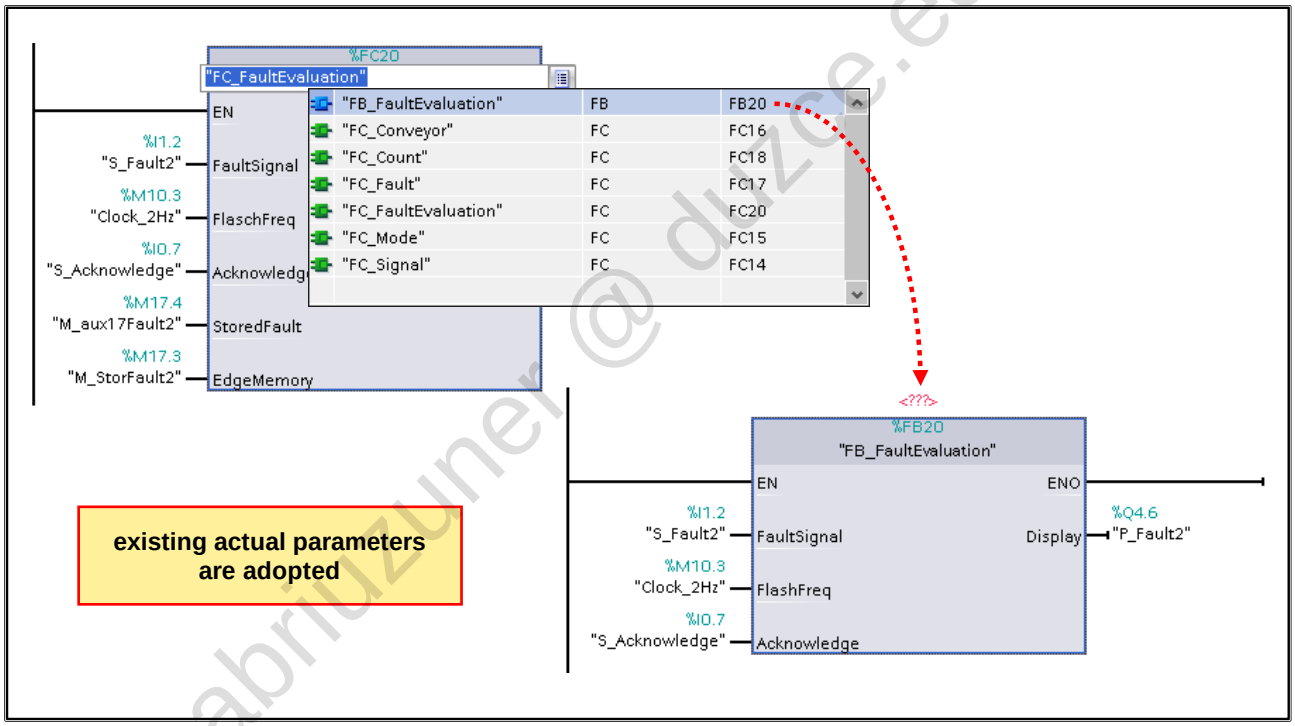
When the function block is called, the values of the actual parameters are stored in the instance data block. If no actual parameters were assigned to a formal parameter in a block call, then the last value stored in the instance DB for this parameter is used in the program execution.

You can specify different actual parameters with every FB call. When the function block is exited, the data in the data block is retained.

Static Variables

Unlike functions, function blocks have "static variables" (Static). These variables form the memory of the FB since they are not stored in the L-Stack but in their own memory area, the instance DB.

9.7.4. Changing the Block Call

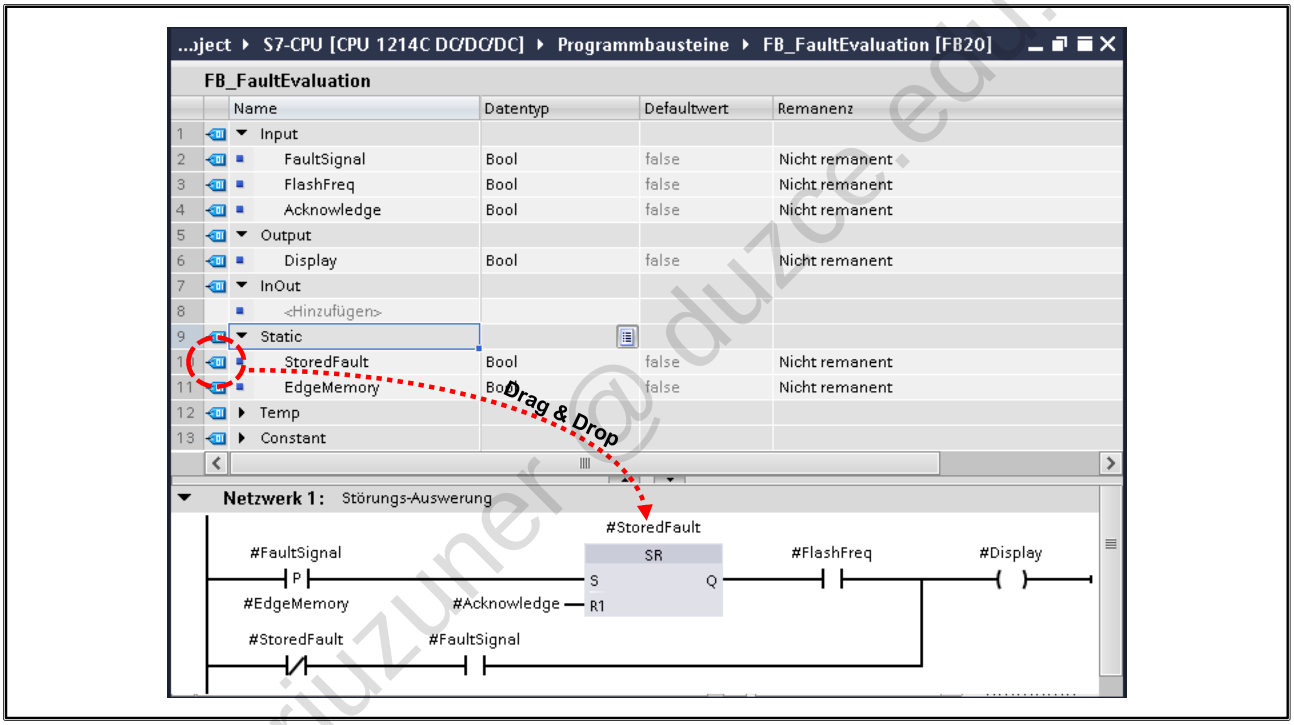


To replace the call of a block with another block call, a selection list of all FCs and FBs can be opened at the calling point by double-clicking on the name of the already called block.

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.

9.7.5. Exercise 3: Programming "FB_FaultEvaluation"



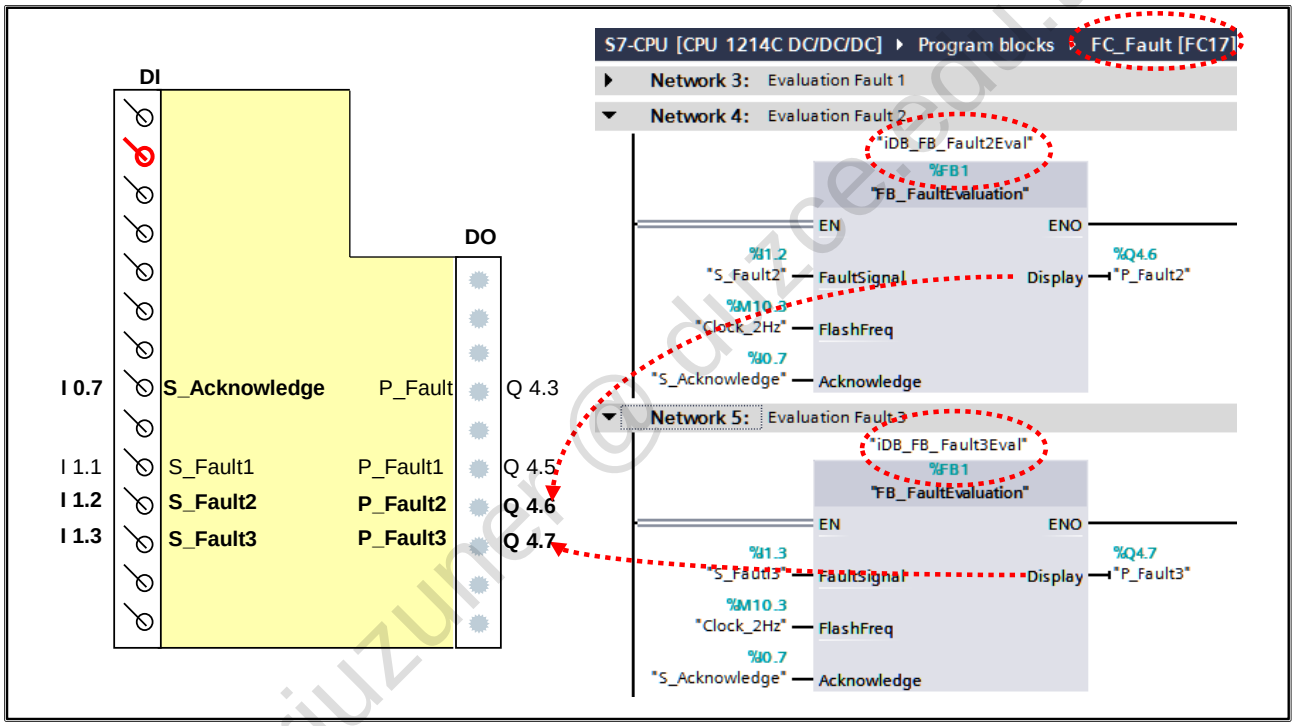
Task

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
3. Program "FB_FaultEvaluation" by copying the necessary program parts from the already programmed "FC_FaultEvaluation"
4. Save your project

9.7.6. Exercise 4: Calling the Function Block "FB_FaultEvaluation" (FB20)



Task

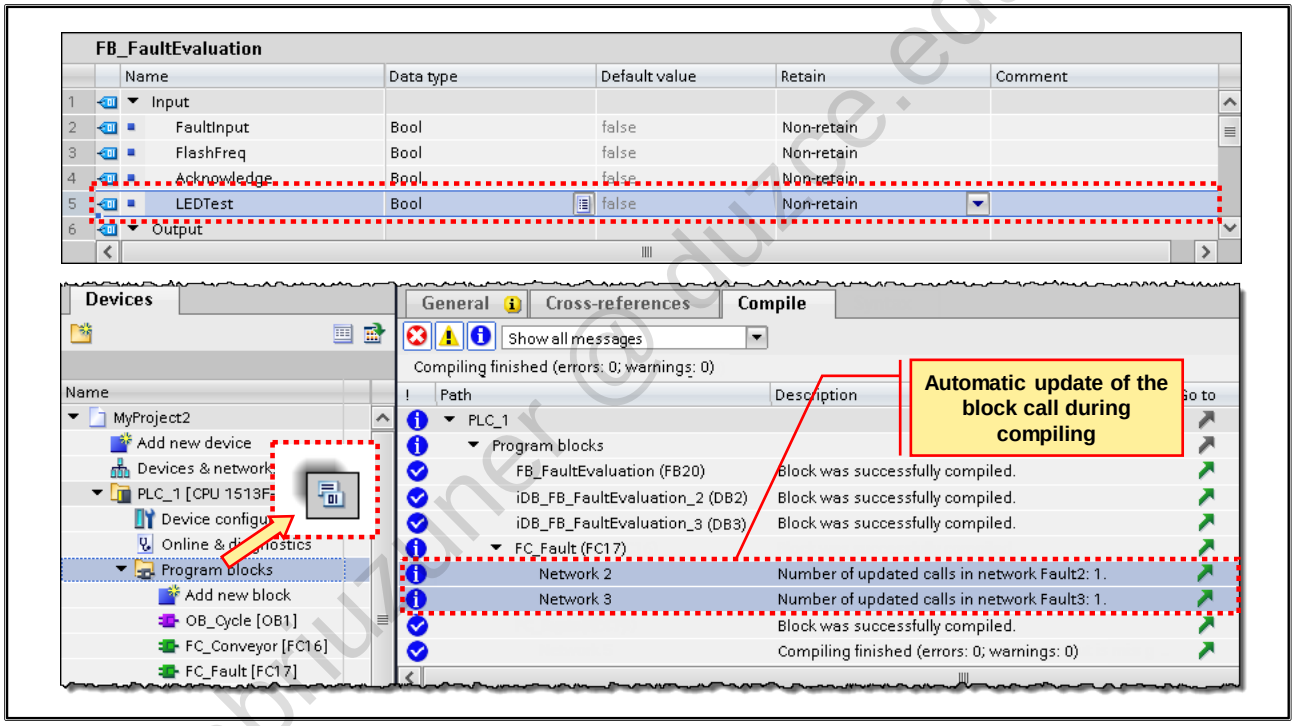
The evaluation of the old Fault 2 (programmed up until now through the call of "FC_FaultEvaluation") and the evaluation of the new Fault 3 is to be implemented with the newly created "FB_FaultEvaluation".

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 17, delete the second call of FC 20 since the evaluation of Fault 2 is to be subsequently implemented with FB 20
2. Program both calls of FB 20 - as shown in the picture - in two new networks in FC 17. Let the Editor generate the instance DBs 2 and 3
3. Save the modified "FC_Fault"
4. First, download the two newly generated instance data blocks DB 2 and DB 3 into the CPU and then download the modified "FC_FaultEvaluation"
5. Test the function of your program

9.8. Inserting Block Parameters Later On



Problem

If you must 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 great degree of correction.

If block parameters are added later to a block already called in the program, you also must update the calls of the block in other blocks.

- 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 follow-up dialog box. A window then appears in which the old (faulty) and the new block call (in the picture with the additional parameter "LEDTest") are displayed. With function blocks, the instance DB is subsequently regenerated.

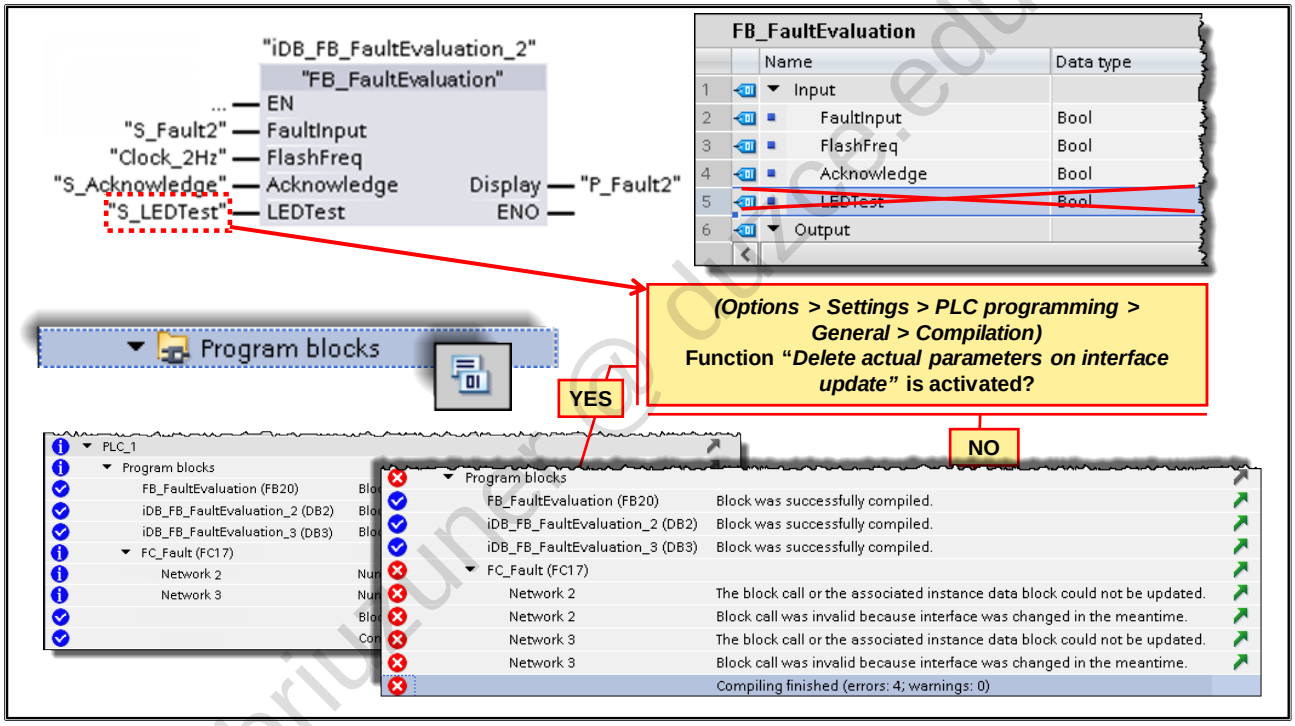
- Automatic Update

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.

With functions, the added formal parameter still must be assigned before downloading into the CPU, since this is a "Must Assign".

With function blocks, the default value from the associated instance DB is used when the formal parameter is not assigned ("Can Assign").

9.9. Removing Block Parameters Later On

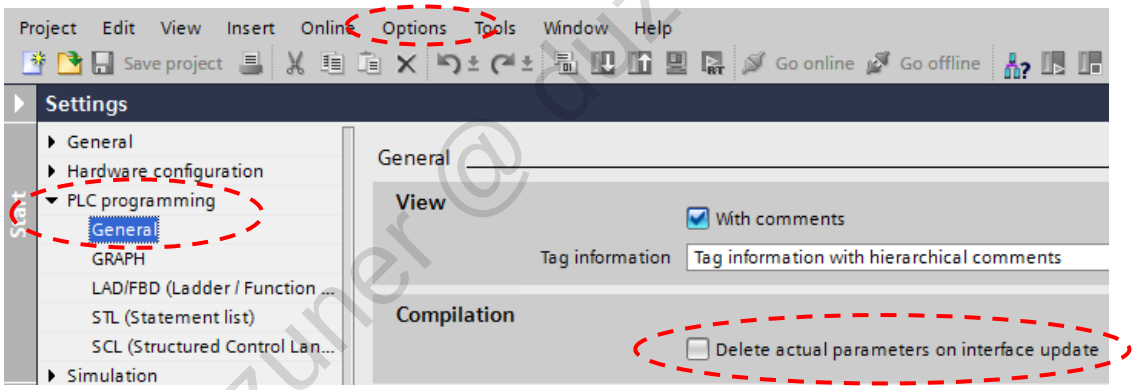


Problem

If you must 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 great degree of correction.

- Automatic Update

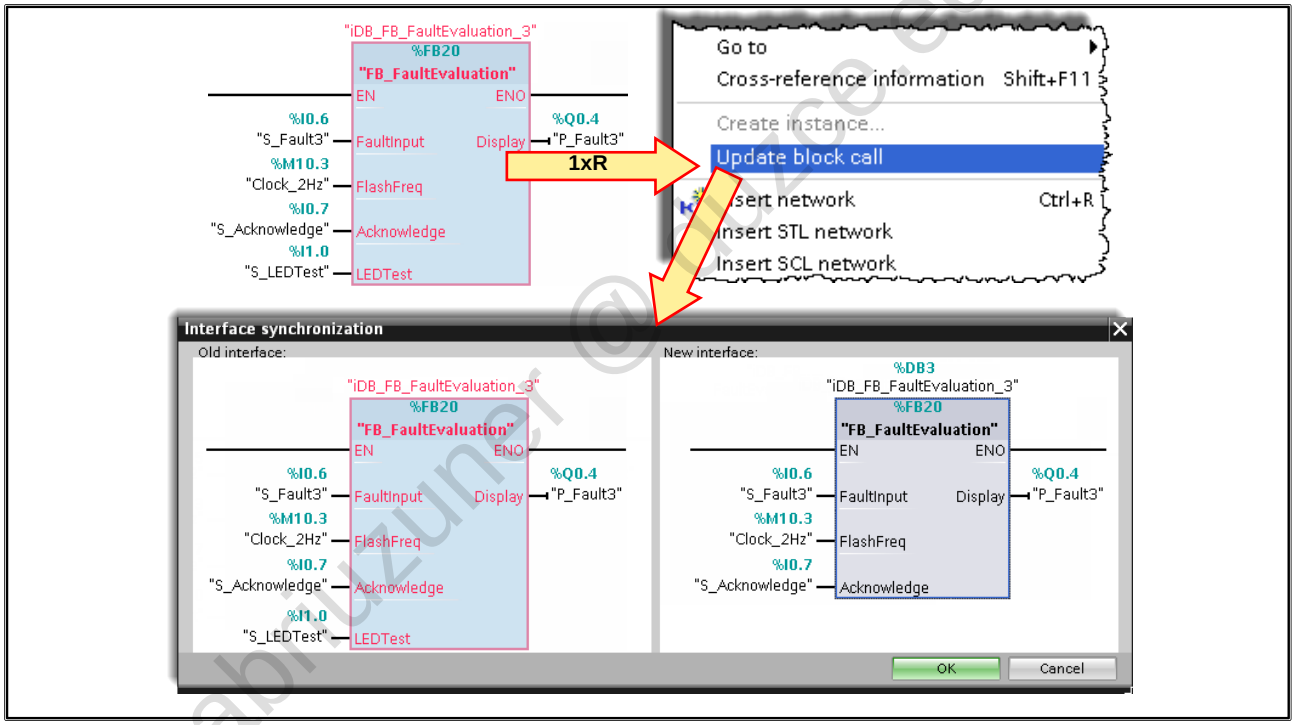
If block parameters are removed later from a block already called in the program and the function “Delete actual parameters on interface update” is not activated in the menu item Options > Settings > PLC-programming > General > Compilation, the actual parameters of the deleted formal parameters are not automatically deleted during compiling. That is, these block calls in other blocks must be updated manually.



- 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 follow-up dialog box. A window then appears in which the old (faulty) and the new block call (in the picture with the additional parameter "LEDTest") are displayed. With function blocks, the instance DB is subsequently regenerated.

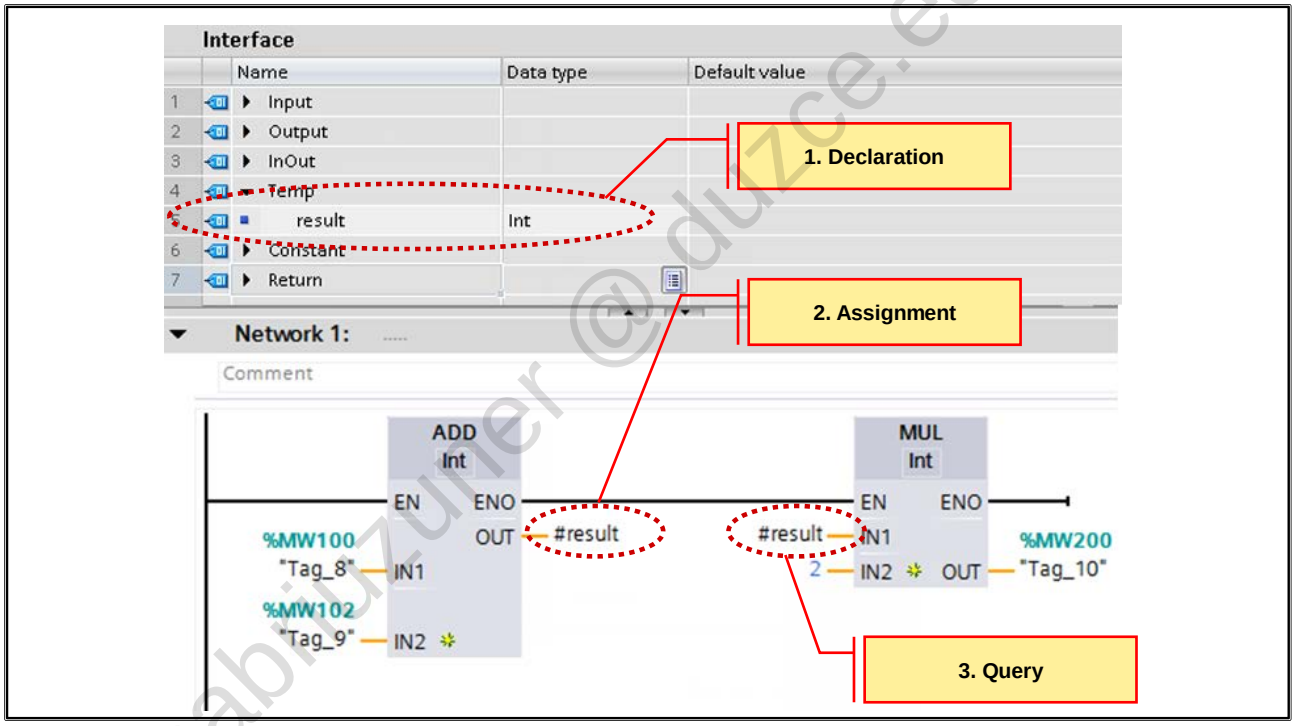
9.10. Manually Updating a Block Call



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 follow-up dialog box. A window then appears in which the old (faulty) and the new block call (in the picture with the additional parameter "LEDTest") are displayed. With function blocks, the instance DB is subsequently regenerated.

9.11. Temporary Variables



General

Temporary variables can be used in all blocks (OB, FC, FB). They are used to temporarily store information while the block is being executed. The data is lost when the block is exited.

The data is stored in the L stack (local data stack). It is a separate memory area in the CPU.

Declaration

Before they can be used, variables must be declared in the declaration part of the block. The name of the variable and its data type must be specified. You cannot predefine the variable with an initial value.

After you have saved the block, the relative address of the variable in the L-stack is displayed in the "Offset" column (only for non-optimized block access).

Access

At the beginning of a block execution, all temporary variables have an indefinite value. When working with temporary variables, you must therefore make sure that the variable is first assigned a defined value before it is queried.

In the example, the result of Add(ition) is first assigned to the temporary variable #result before it is then queried at the following Mul(tiplication).

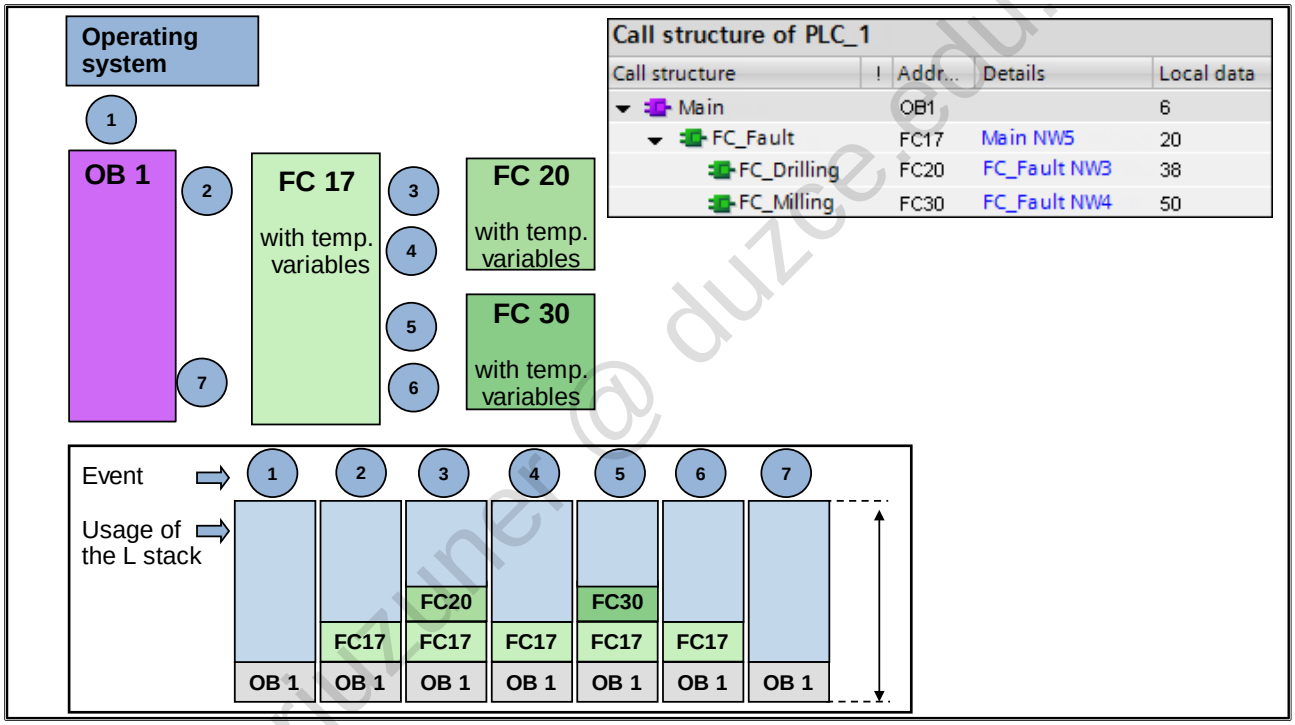
For "Non-optimized block access", you can also access temporary variables absolutely (e.g. L0.0 → Local data bit 0.0). You should, however, try to avoid this since the program is more difficult to read.

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.

9.12. Total Usage of the Local Data Stack



Total Usage

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 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.

Note

If the (CPU-specific) maximum number of local data is exceeded (overflow of the L stack) during program execution in the CPU, the CPU goes into the STOP state. "STOP caused by error when allocating local data" is entered as the cause of error in the diagnostics buffer.

Contents

10

10. Connecting an HMI device

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

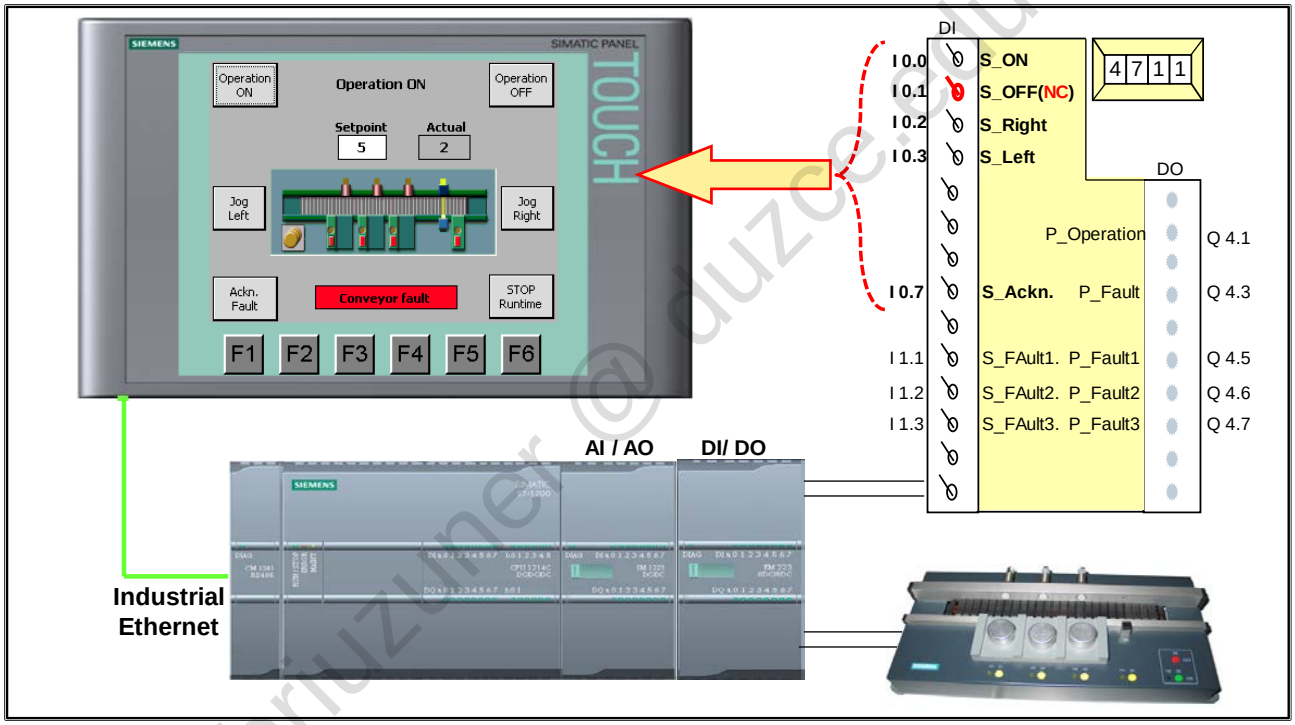
... be able to set the interface for the touchpanel

... be familiar with the principle of data exchange between touchpanel and CPU using tags

... be able to commission a touchpanel project

be able to adjust a STEP 7 program

10.1. Task Description: Operating the 'Plant' via the Touchpanel

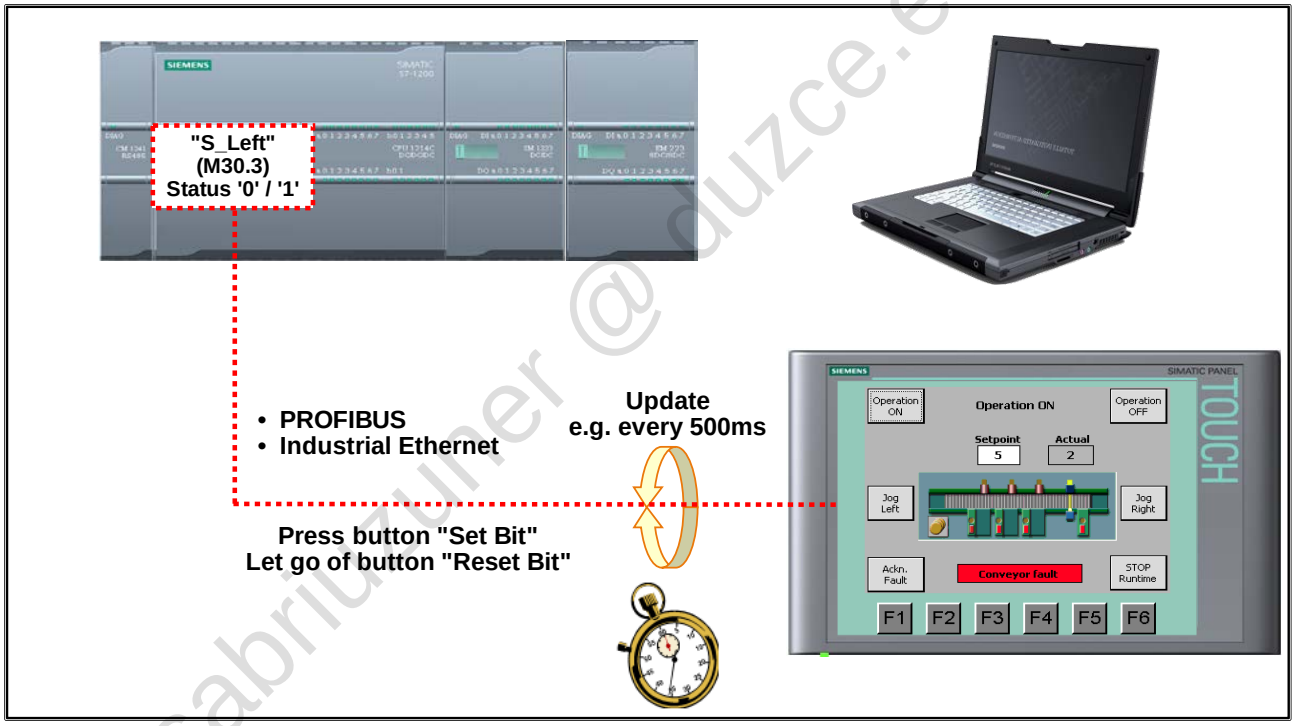


Task Description

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 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 corresponding button on the Touchpanel
- the SETPOINT quantity is no longer constant 3, but can be specified via an input/output field on the Touchpanel

10.2. Introduction to HMI: Data Exchange between Touchpanel and CPU



Tags

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 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:

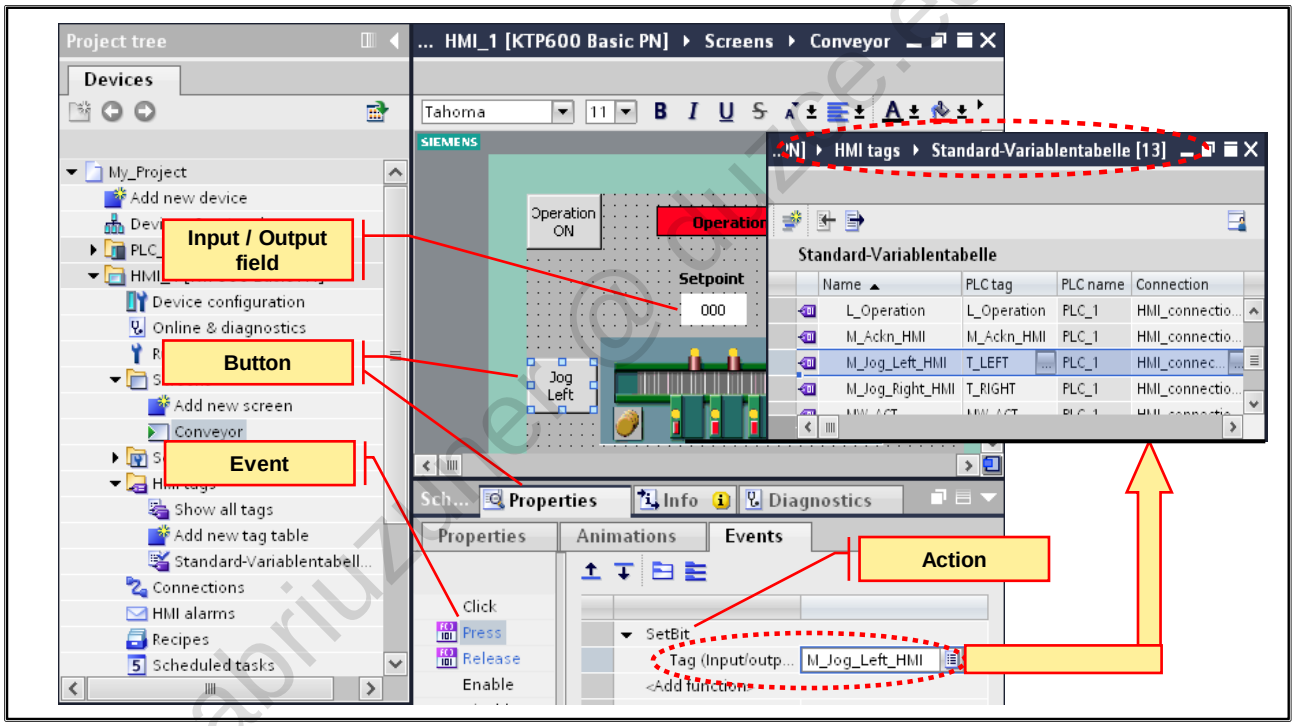
- Data blocks (DB)
- Bit memories (M)
- Inputs (I) and outputs (Q)
- Peripheral inputs and outputs

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 handled 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.

10.2.1. Buttons and Input / Output Fields



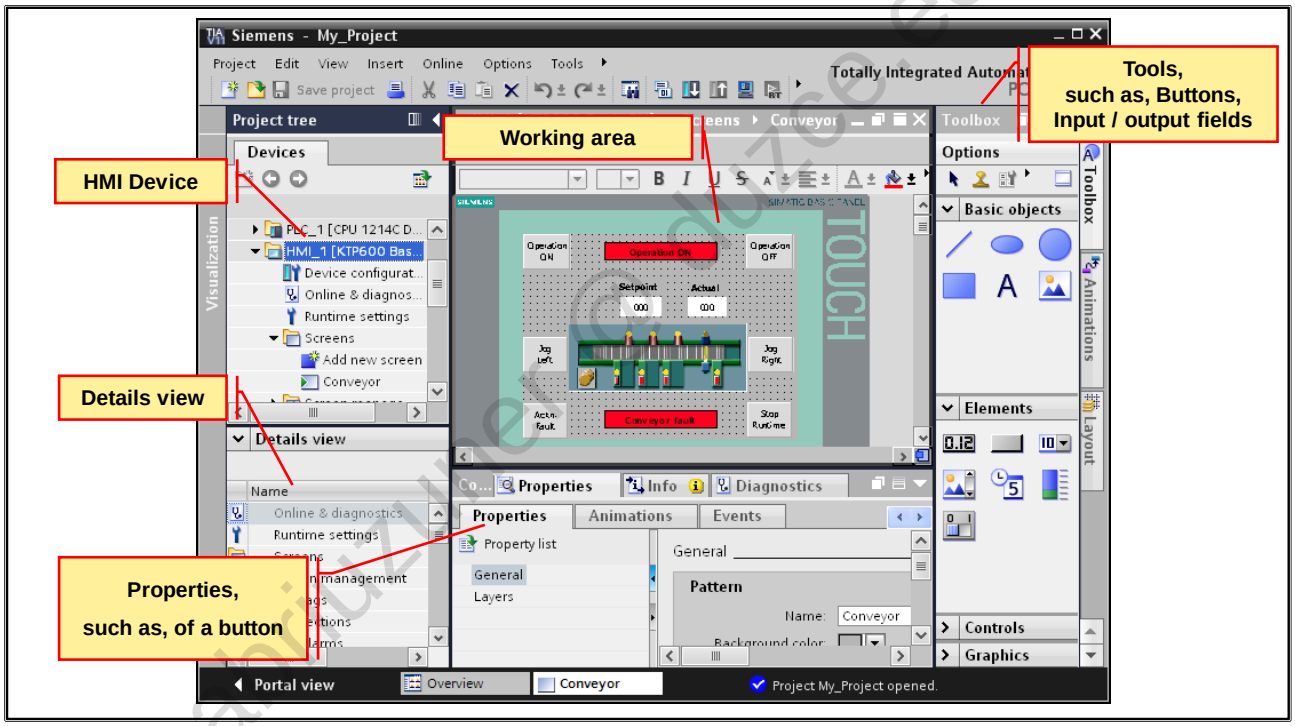
Buttons

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 picture above). The "Events" of a button are used to specify for which event which system function is to be executed.

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.

10.2.2. WinCC - Basic 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 also 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.

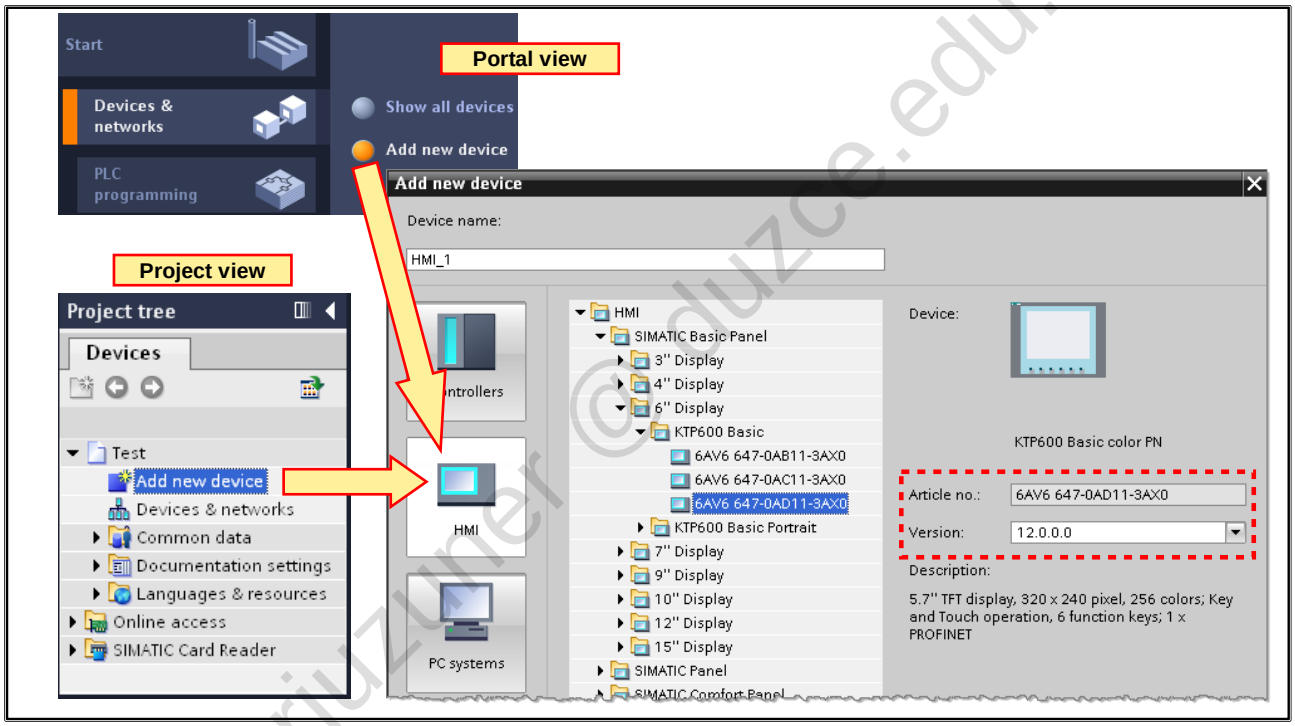
Properties Window

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 must be set.

Toolbox Window

The toolbox window contains all configurable objects which can be configured in screens and permits access to libraries.

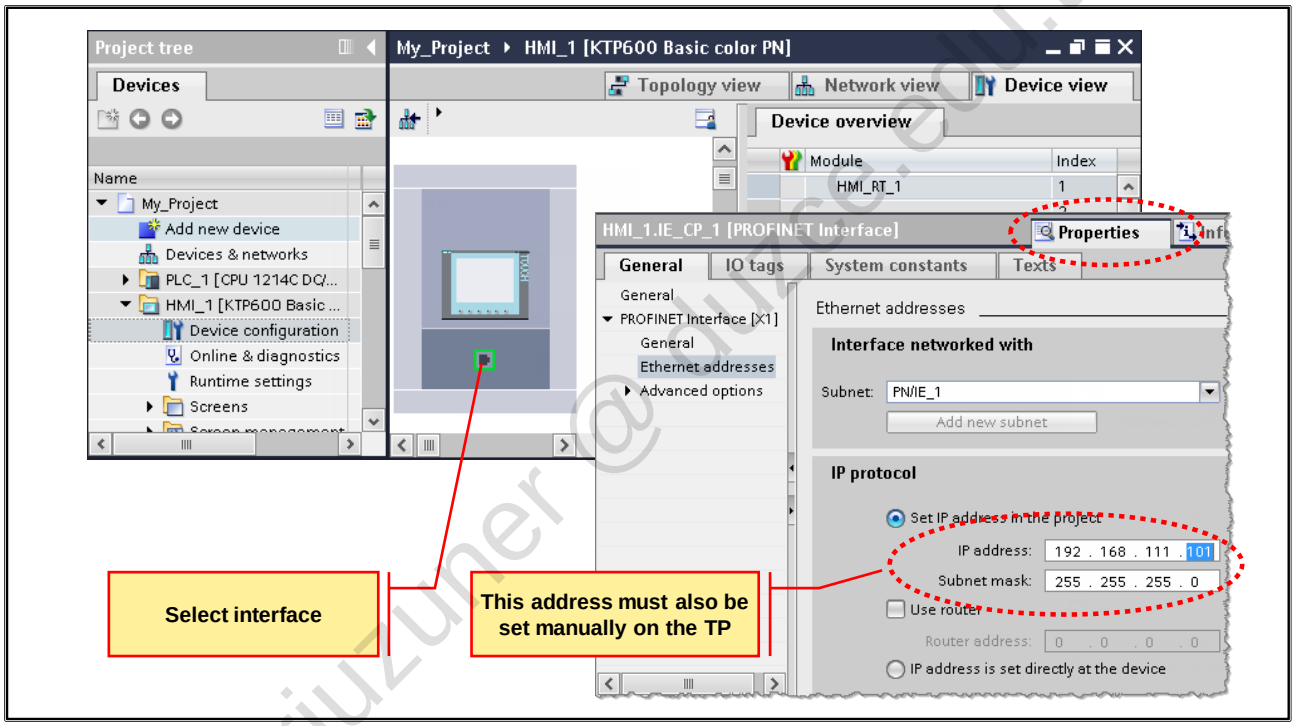
10.2.3. Adding an HMI Device



Adding an HMI Device

New HMI devices can be added in both the Portal view and the Project view. More than anything else, attention must be paid to the device data such as article (order) number and version number.

10.2.4. Configuring the IP Address of a Touchpanel

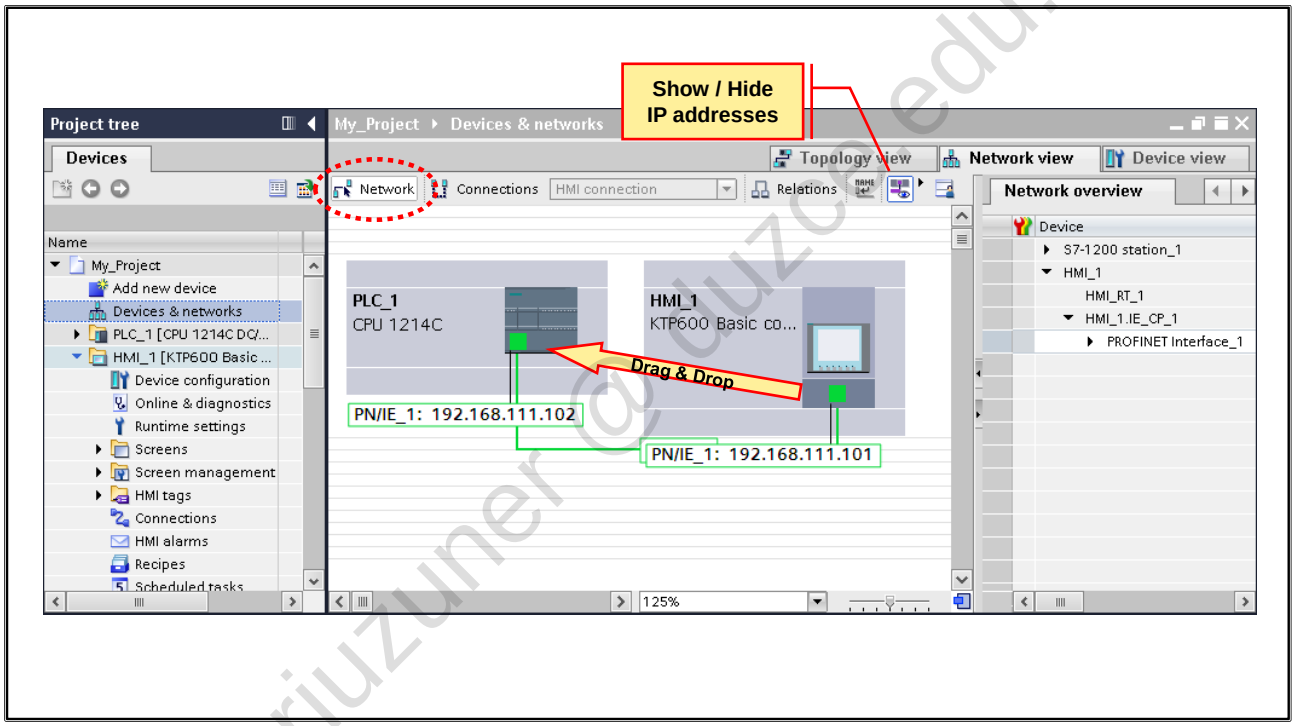


PROFINET Interface of the Touchpanel

Regardless of whether the Devices & networks 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.

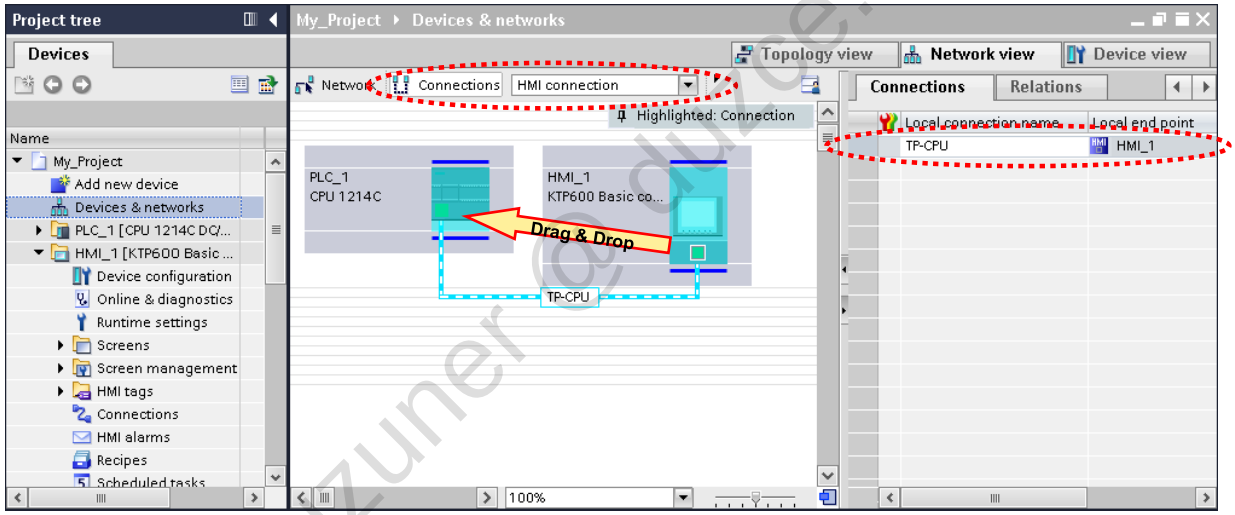
10.2.5. Networking a Touchpanel



Networking

During networking, devices are connected to a subnet. The device interface must be compatible with the type of network. The devices are networked with the "Devices & networks" editor in the "Network" view by connecting the device interfaces using drag & drop.

10.2.6. Configuring an HMI Connection

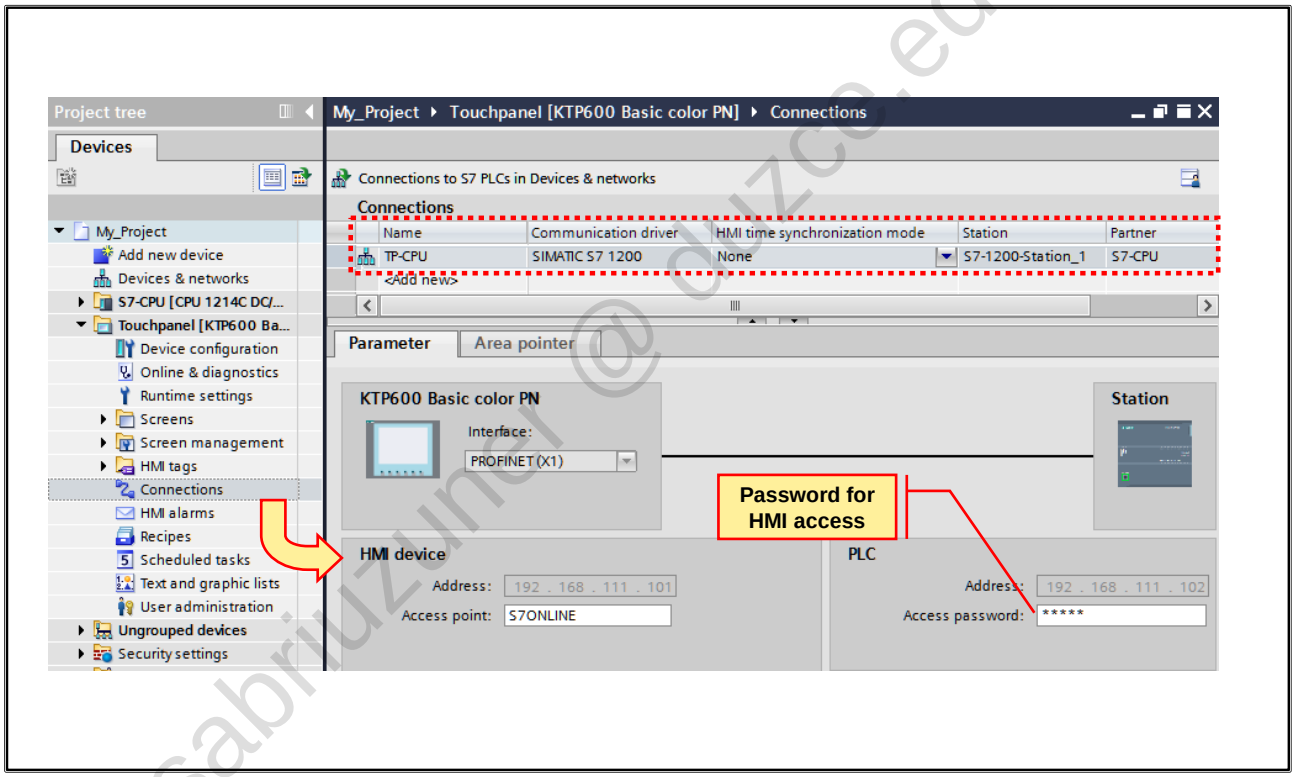


Configuring HMI Connections

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".

10.2.6.1. HMI Connection: Entering the CPU Password



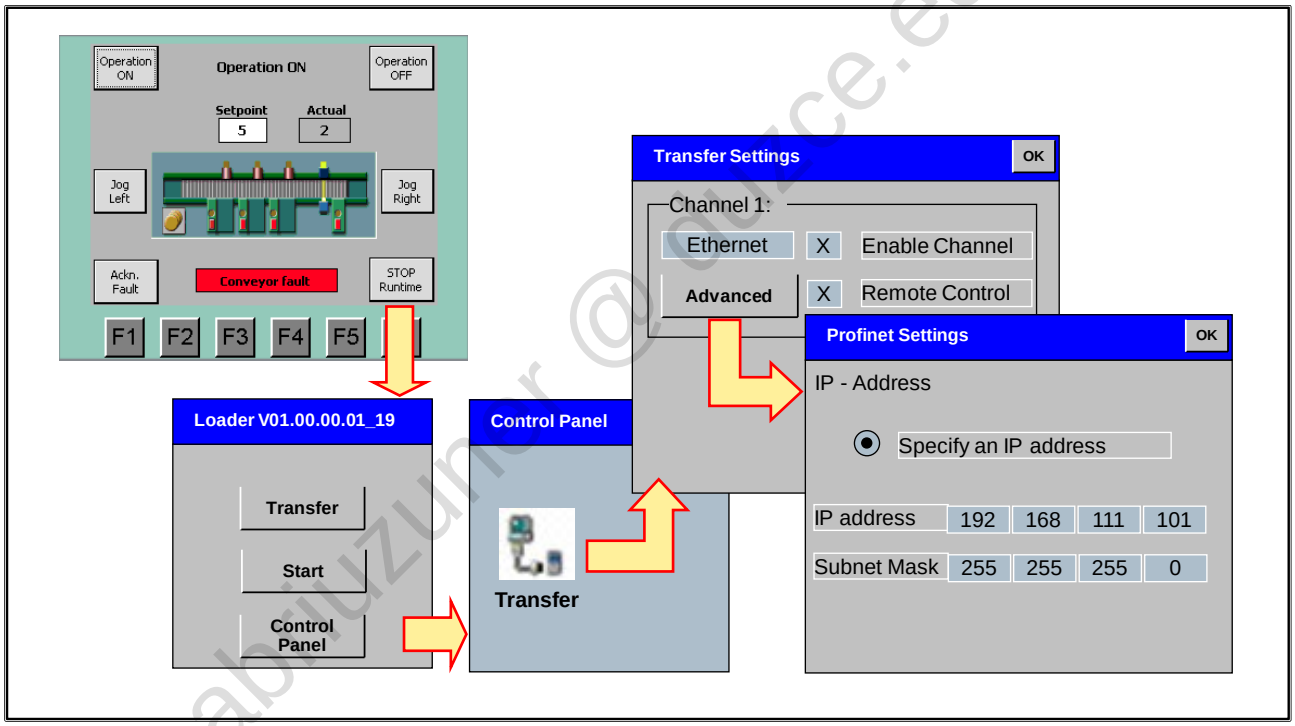
Connection

The configured connections are visible in the Connections of the HMI device.

Password Query for HMI Accesses

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).

10.2.7. Setting the IP Address on the Touchpanel



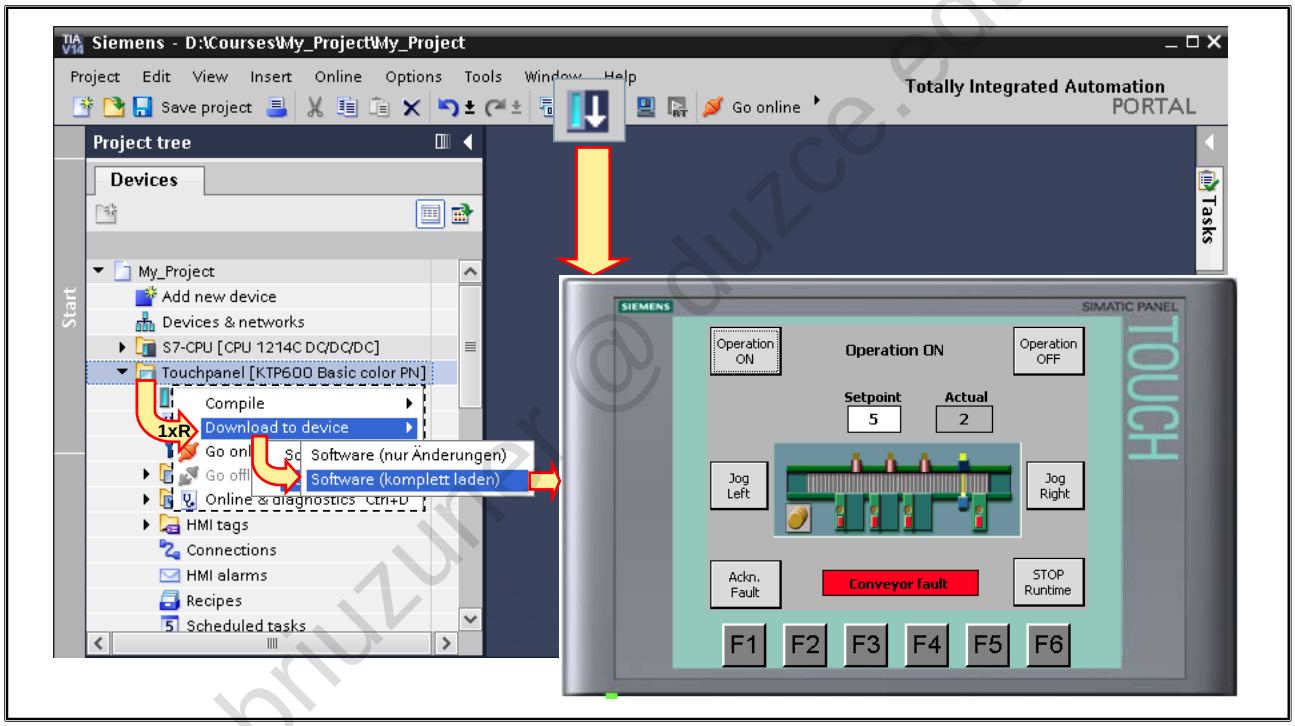
IP Address of the Touchpanel

The interface of the touchpanel must be set to the same IP address and subnet mask as it is also configured in the offline project.

Remote Control

You can initiate the loading of the WinCC flexible project without having to first manually end the Runtime of the HMI device. The Panel automatically ends the Runtime and switches to the Transfer Mode.

10.2.8. Downloading an HMI Project into the Touchpanel



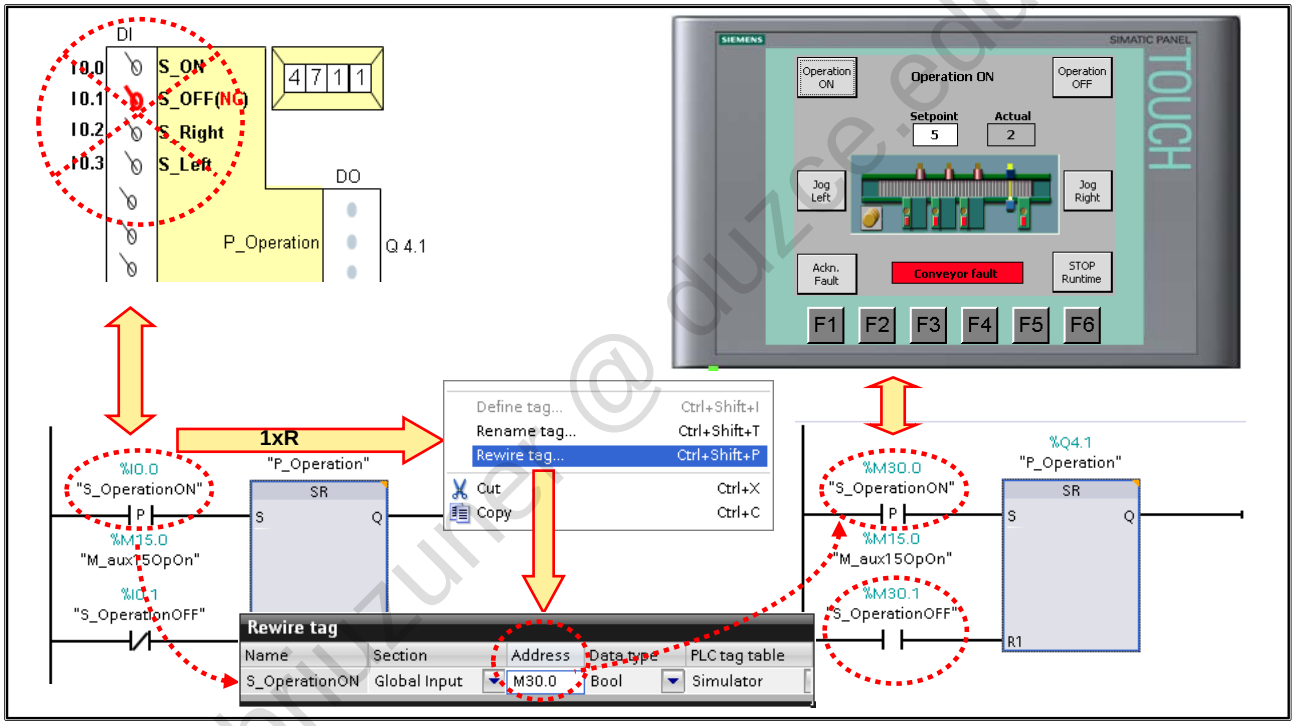
Downloading an HMI Project into the Touchpanel

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. 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 linked 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.

It is also advisable to execute the "Compile > Software (rebuild all blocks)" command occasionally to reduce the time required for compiling delta data in current engineering sessions.

10.3. Adjusting the S7 Program with "Rewire"



Adjusting the S7 Program using "Rewire"

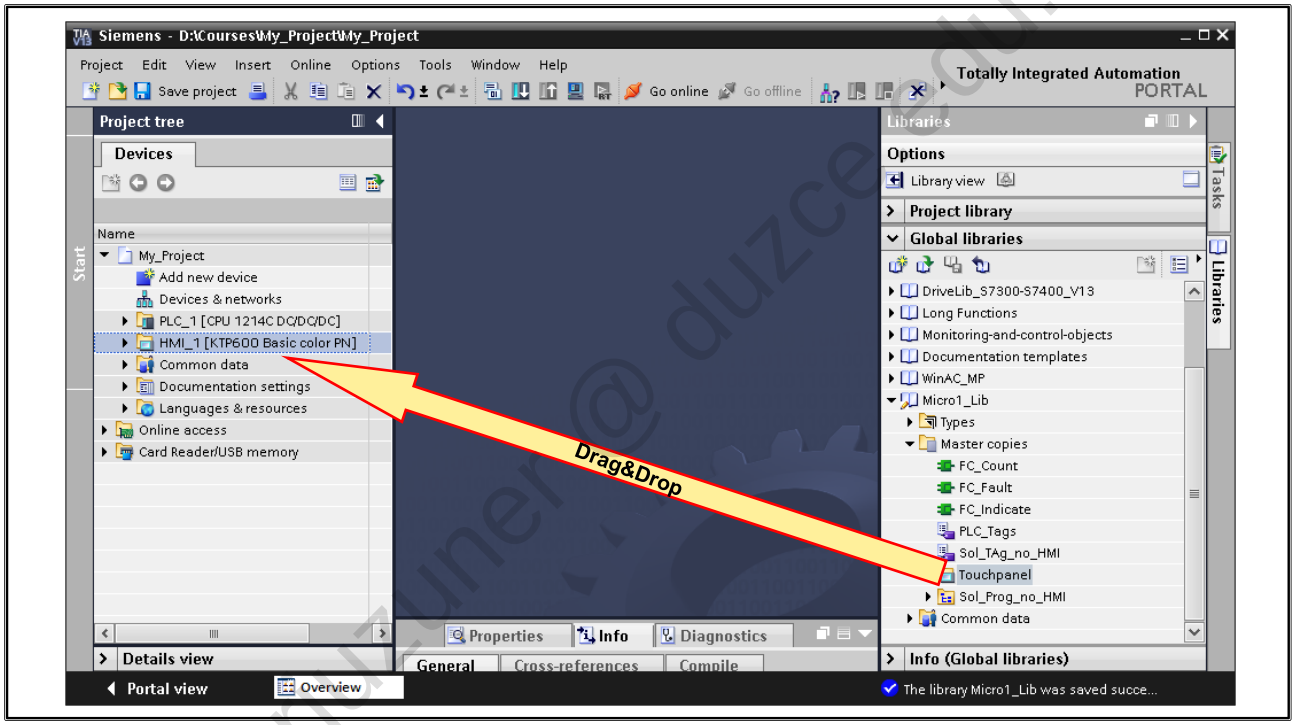
So that the plant can be operated via the touchpanel, PLC tags must be used in the S7 program which can be read or overwritten by the touchpanel. That is, that the variables used up until now in the S7 program must be replaced accordingly with tags which can be read or overwritten by the touchpanel.

Another possibility is to "rewire" the variables used up until now in the S7 program as shown in the picture. With this so-called "rewiring" of a variable, the symbolic name of the variable is kept, but it is assigned a different absolute address which is read or overwritten by the touchpanel.

"Rewiring" can be carried out directly on the tag in the Block Editor (as shown in the picture). The changes are adopted immediately in the PLC tag table and thus in all program blocks.

Rewiring can also be done directly in the PLC tag table. The changes immediately affect all program blocks.

10.4. Exercise 1: Copying the Touchpanel Project



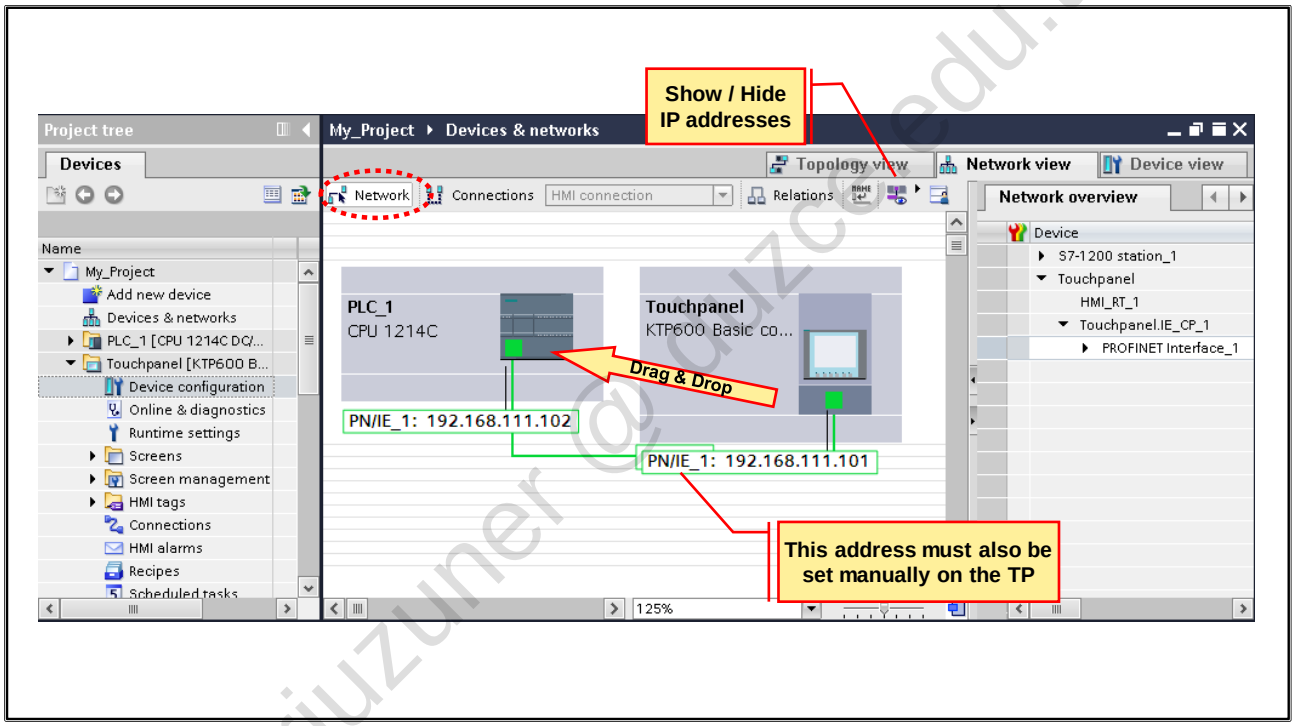
Task

Your project currently doesn't contain an HMI device. Instead of a completely new configuration, you are to copy a prepared panel project from the global library "Micro1_Lib" into your project.

What to Do

1. Using drag & drop, copy the library element "Touchpanel" from the global library "Micro1_Lib" into your project (see picture)
2. Save your project

10.4.1. Exercise 2: Networking the Touchpanel



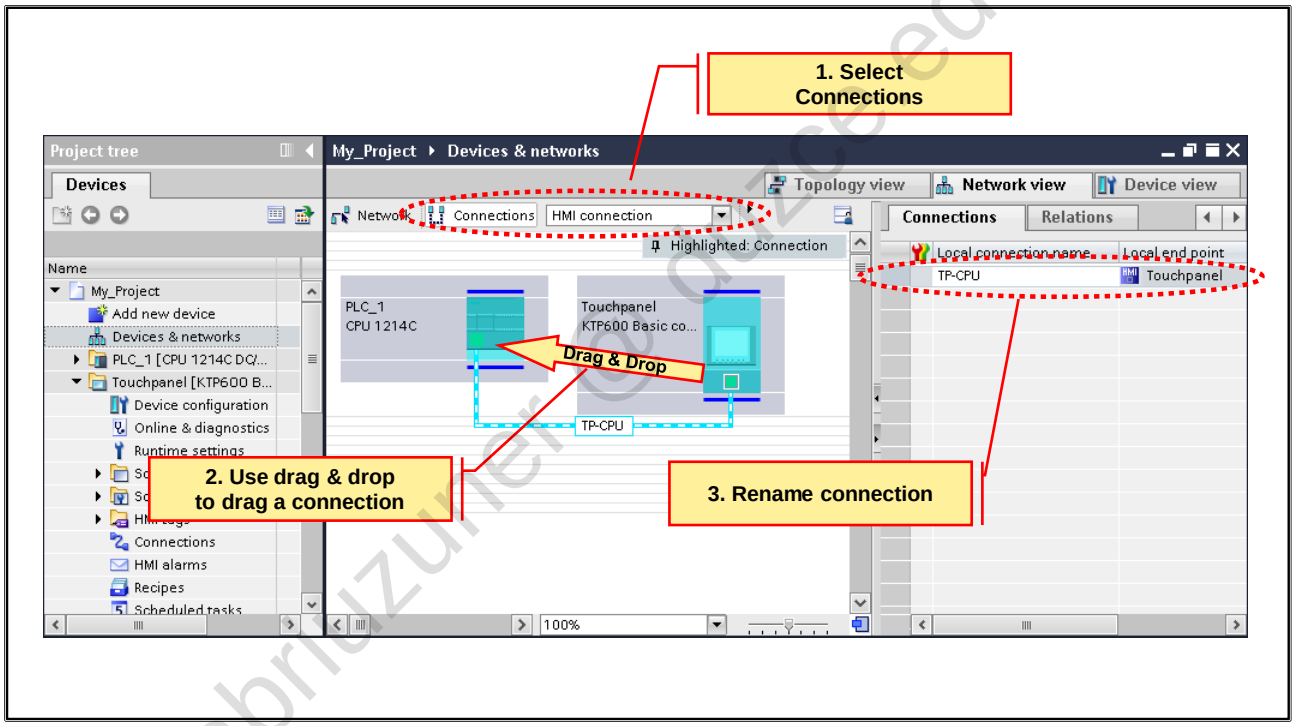
Task

The added touchpanel is to be networked with the Ethernet network.

What to Do

1. In the Project tree, start the "Devices & networks" editor and switch to the "Network" view
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. 'Show' the IP addresses of the CPU and the touchpanel using the button highlighted in the picture

10.4.2. Exercise 3: Configuring the HMI Connection



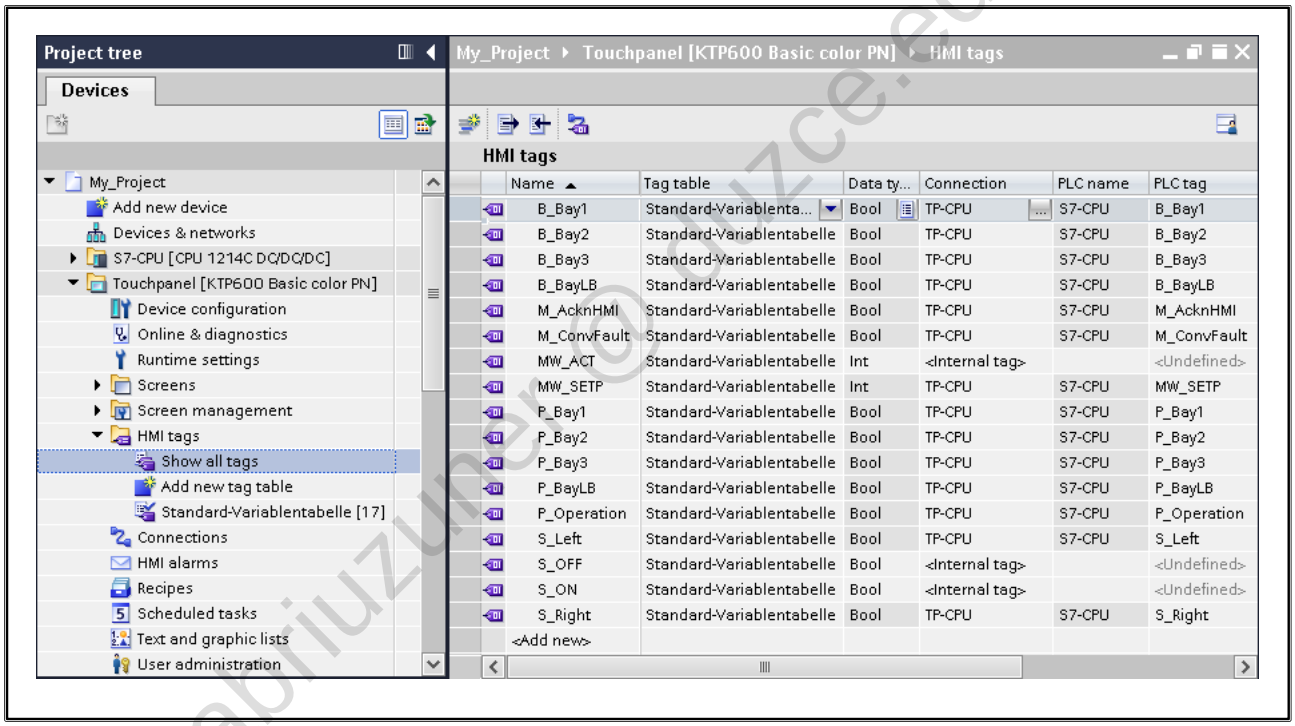
Task

Now that the TP is networked with the Ethernet network, an HMI connection between the TP and the CPU must be created. This connection is to give the name "TP-CPU" because this name is already used in panel project from the library.

What to Do

1. In the Project tree, start the "Devices & networks" editor and switch to the "Connections" view and there select "HMI connection"
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 and given a default name
3. Drag open the window containing the connection table (see picture) and rename the connection "TP-CPU"

10.4.3. Exercise 4: Checking the HMI Tag Connections



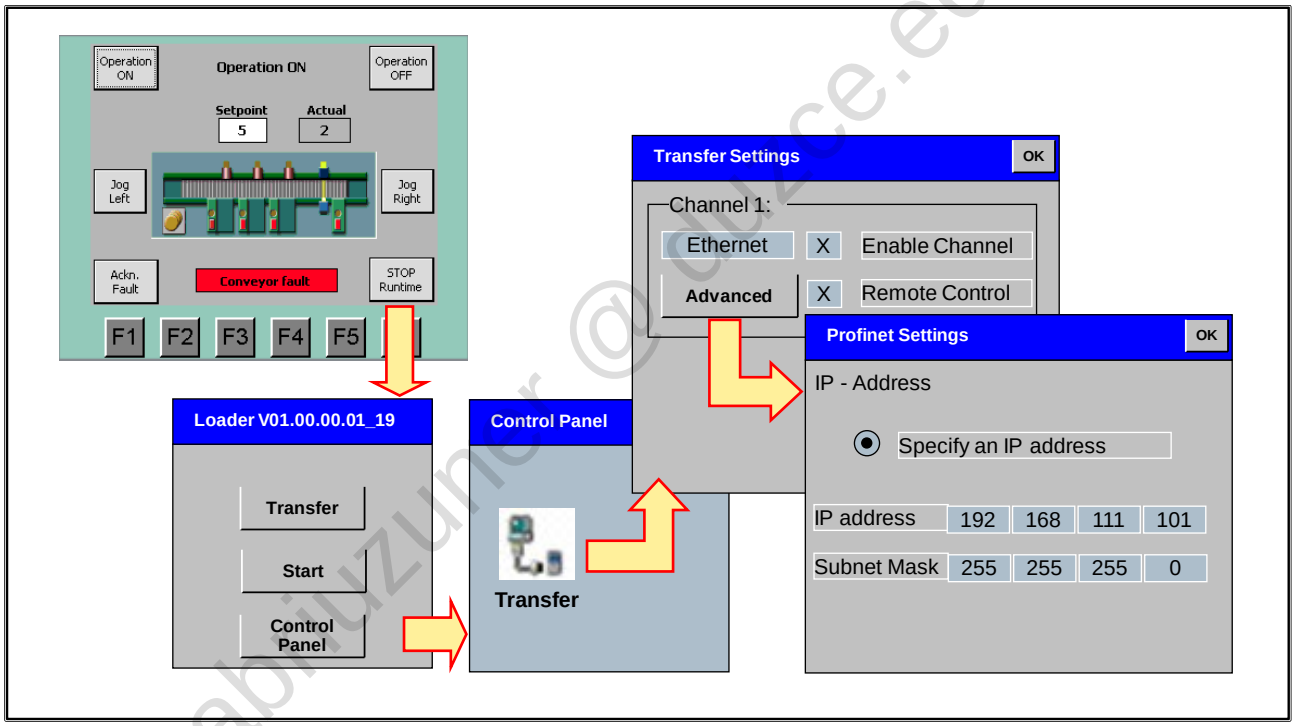
Task

Several HMI tags are already connected to PLC tags (inputs and outputs whose addresses are defined by the conveyor model ‘wiring’). Others still must be connected by you.

What to Do

1. Open the "HMI tags" of the HMI device
2. Connect the HMI tags that are not yet connected to the corresponding PLC tags
3. Save your project

10.4.4. Exercise 5: Setting the IP Address on the TP



Task

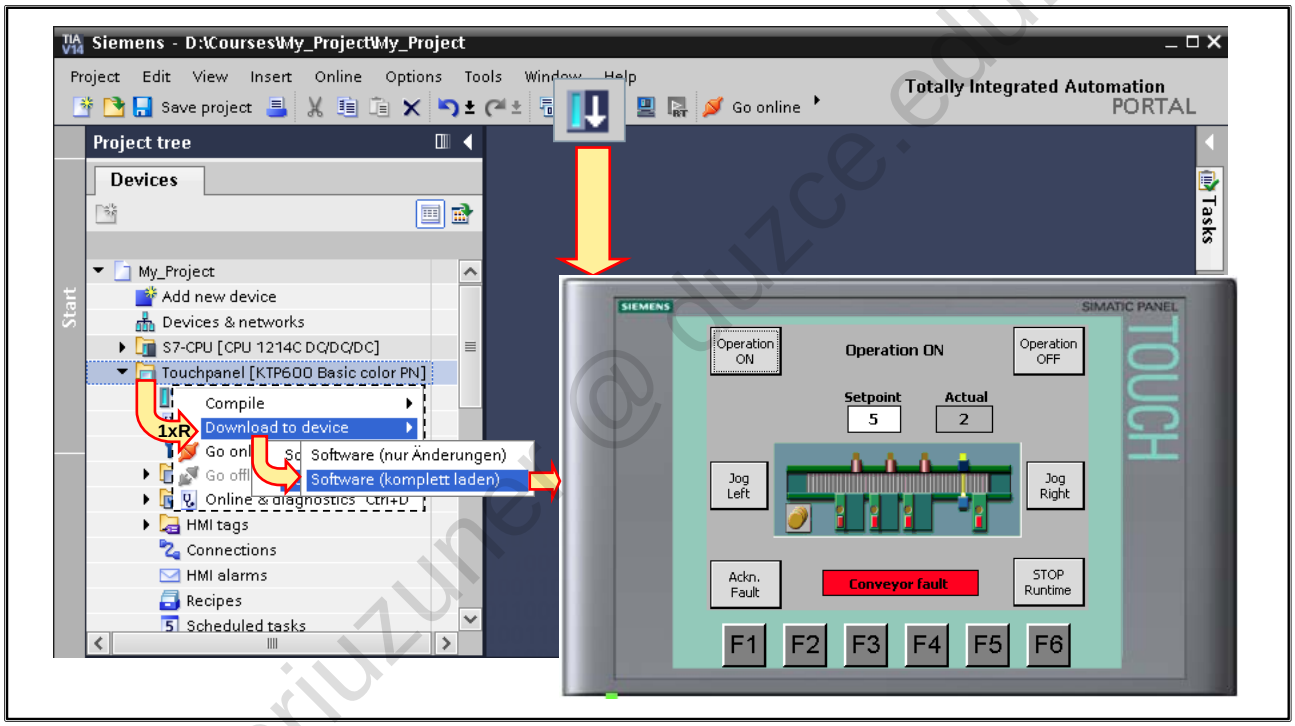
The interface of the touchpanel is to be set as shown in the picture so that the configuration can then be loaded into the panel.

Runtime must be exited before the interface can be assigned parameters. For this, a button for exiting Runtime is generally provided in the Start screen. When Runtime has been exited, the "Loader" appears through which the " Control Panel" can be activated. The Loader also appears every time power is restored.

What to Do

1. Exit the Runtime of the Touchpanel
2. Activate the Control Panel
3. Select "Transfer" by double-clicking on it
4. Implement the settings shown in the picture
5. Go back to the Loader by closing the windows with "OK"
6. Activate the "Transfer" button so that the touchpanel waits for a connection to be established by the PG

10.4.5. Exercise 6: Downloading the HMI Project into the Touchpanel



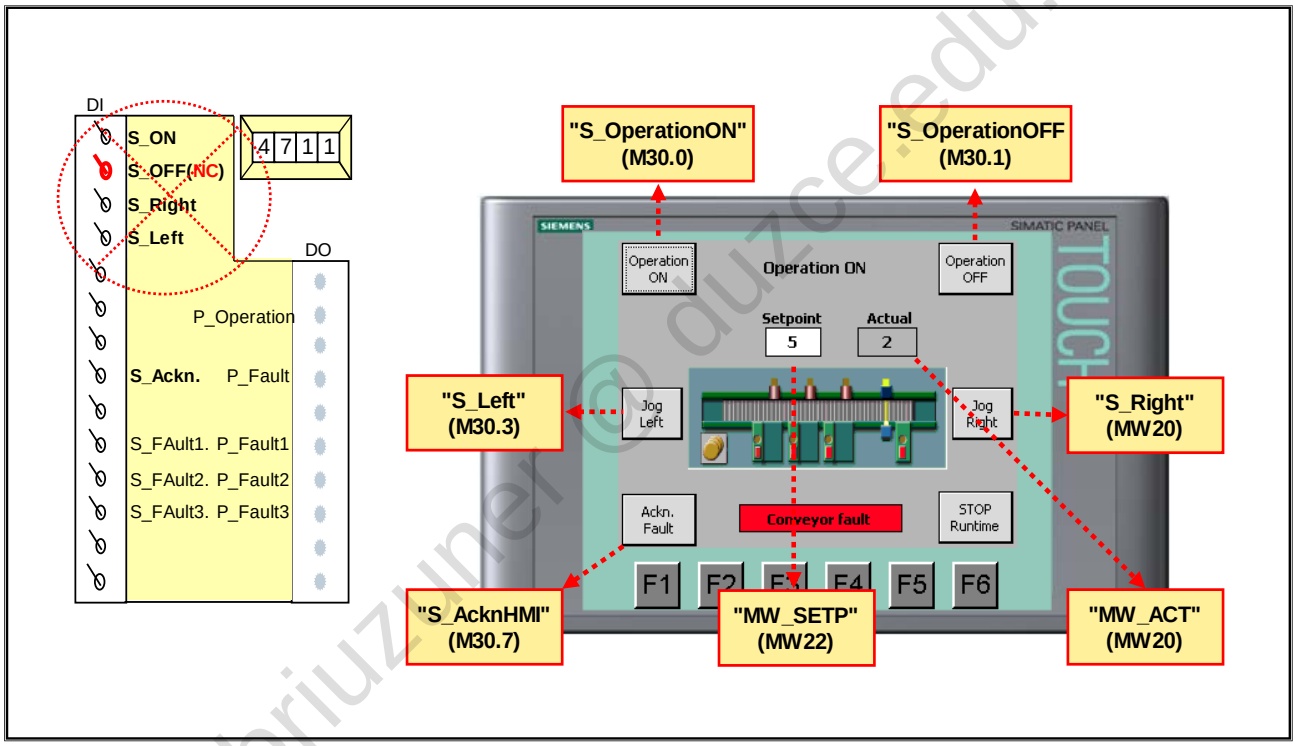
Task

From the now complete project, all S7 blocks will be downloaded once more from the programming device into the CPU and the entire panel project will be transferred from the programming device into the touchpanel.

What to Do

1. Completely compile the entire CPU program and the entire panel project
2. In the Inspector window, in the "Compile" tab, check whether the compilation was successful. If need be, make the necessary corrections
3. Download all S7 blocks into the CPU
4. Download the panel project into the Touchpanel
5. Save your project

10.4.6. Exercise 7: Operating the 'Plant' via the Touchpanel



Task

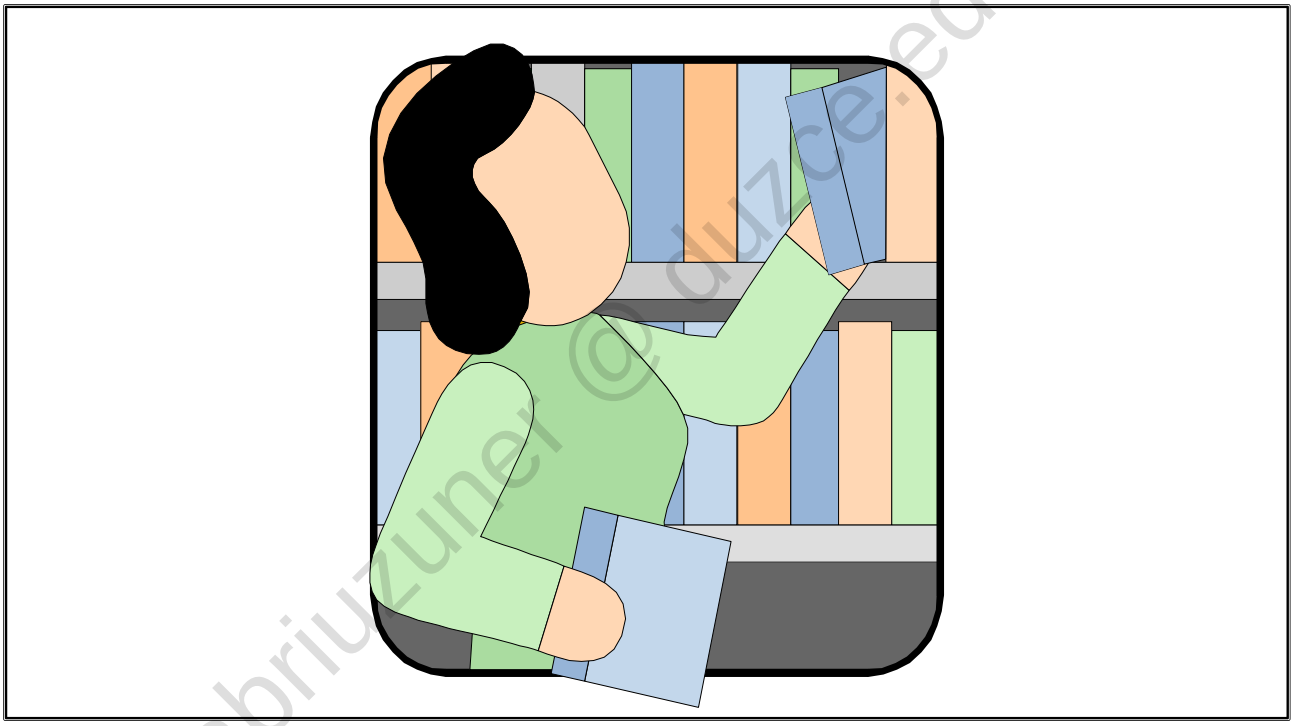
The touchpanel project is to be commissioned and the S7 program of the controller is to be adjusted in such a way that …

- the functions "Operation ON/OFF" and "Jog Right/Left" are no longer done using the simulator switches but instead are done using the touchpanel buttons
- acknowledgement of conveyor faults should be possible, as before, via the simulator switch "S_Acknowledge" (I 0.7), and in addition via the corresponding button on the Touchpanel
- the SETPOINT quantity is no longer to be constant 3, but can be specified via an input/output field on the touchpanel (the display of the ACTUAL quantity is already configured)

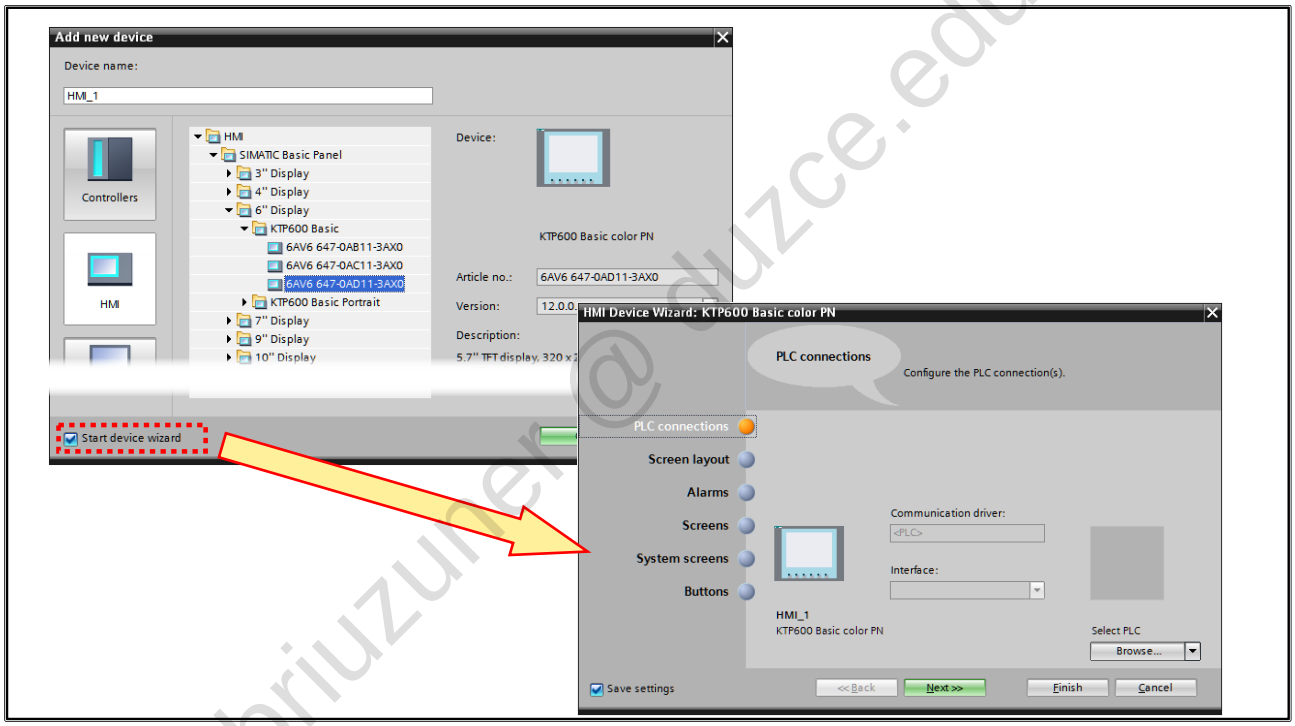
What to Do

1. Establish the functions "Operation ON / OFF" and "Jog Left / Right" by rewiring the variables "S_ON", "S_OFF (NC)", "S_Right" and "S_Left" used up until now to the absolute addresses M30.0, M30.1, M30.2 and M30.3 shown in the picture
2. Establish the function "Acknowledge fault" by also connecting the memory bit "M_AcknHMI" (M30.7) shown in the picture in addition to the already connected simulator switch "S_Acknowledge" (I 0.7)
3. Establish the function "Setpoint" by exchanging the constant (=3) used in the program with the memory word "MW_SETP" (MW22)
4. Download all modified blocks into the CPU
5. Check all required functions
6. Save your project

10.5. Additional Information



10.5.1. HMI device wizard



Device Wizard

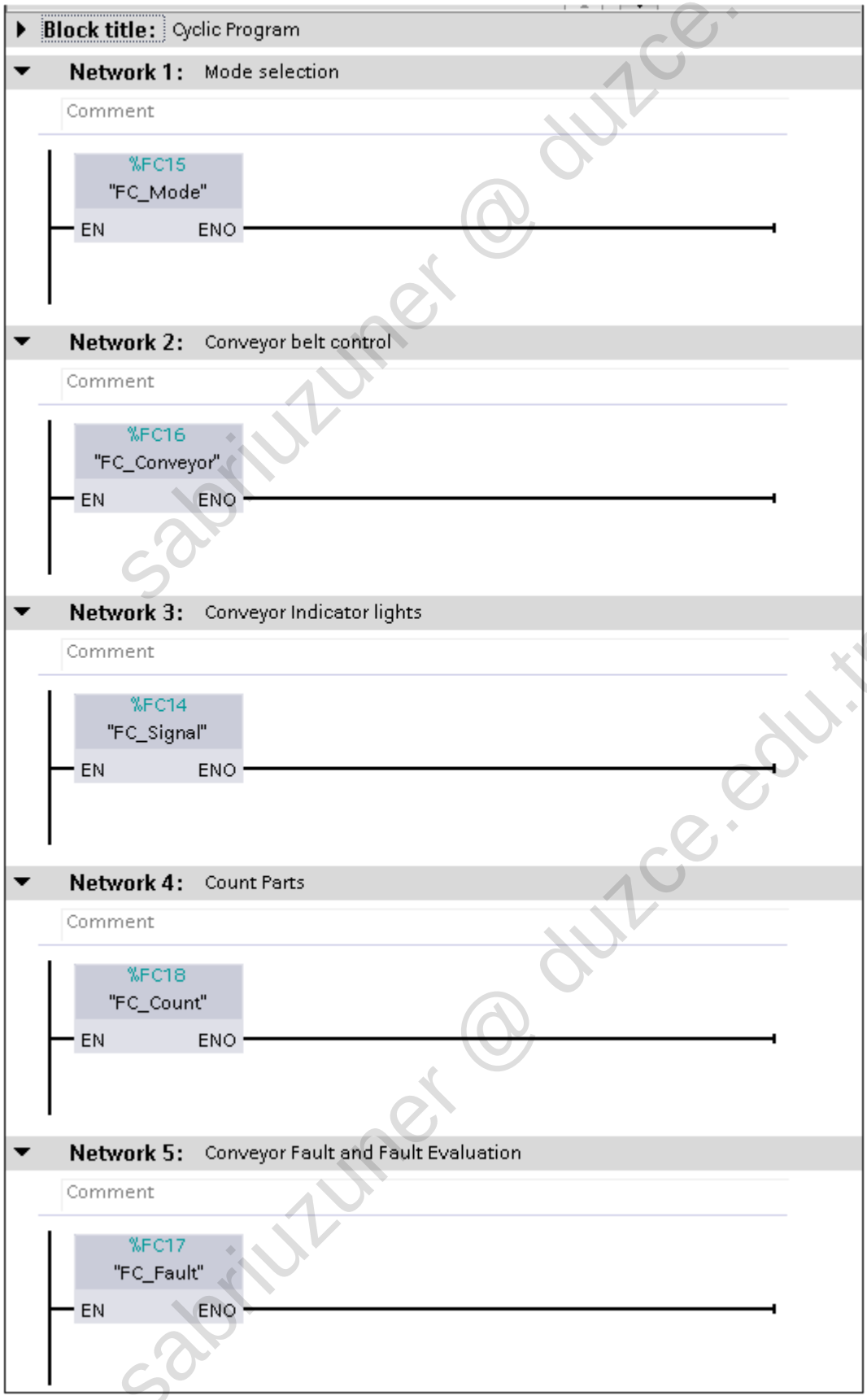
The HMI device wizard can be used to preset some settings for the device to be inserted. It's a step-by-step dialog-based wizard.

Contents

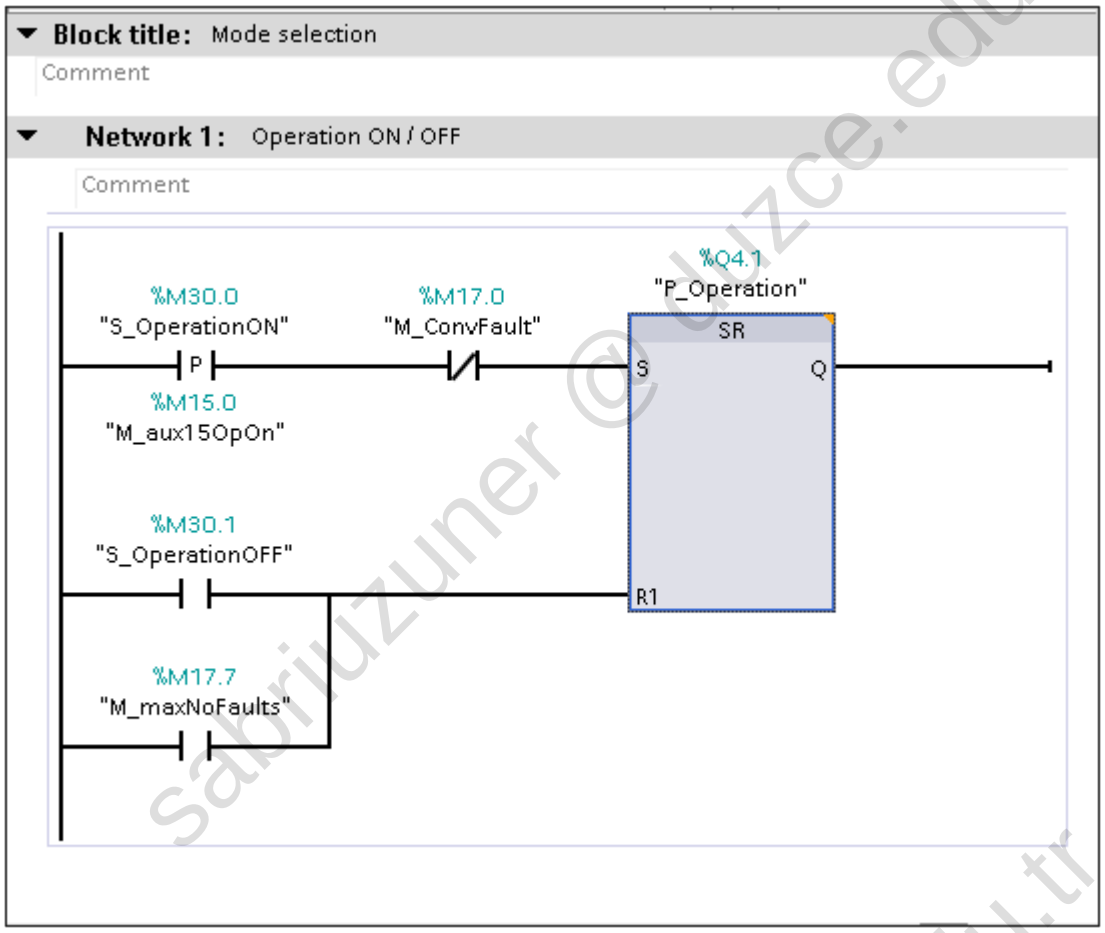
11

11. Solutions

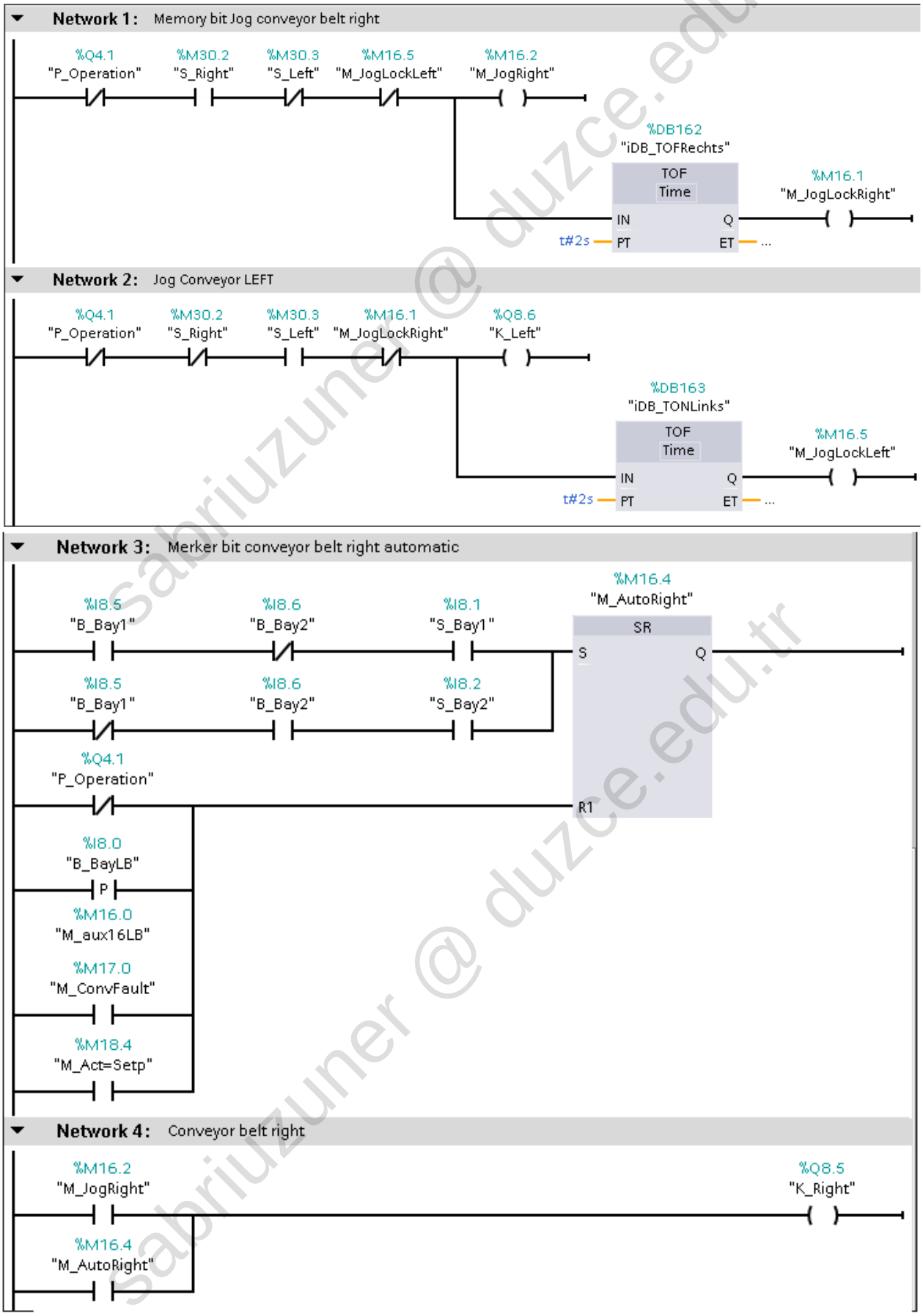
11.1. "OB_Cycle" (OB1)



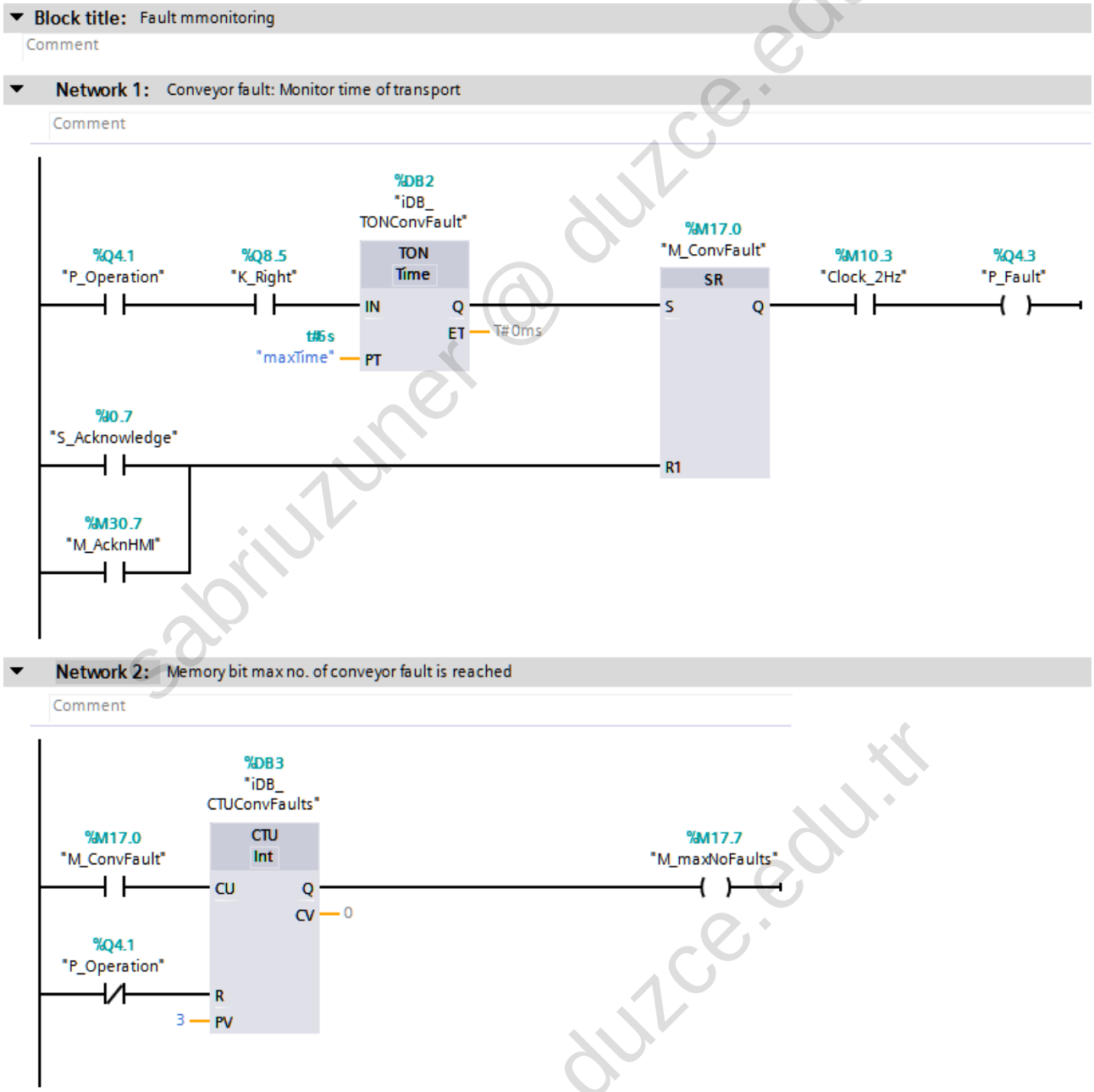
11.2. "FC_Mode"



11.3. "FC_Conveyor"

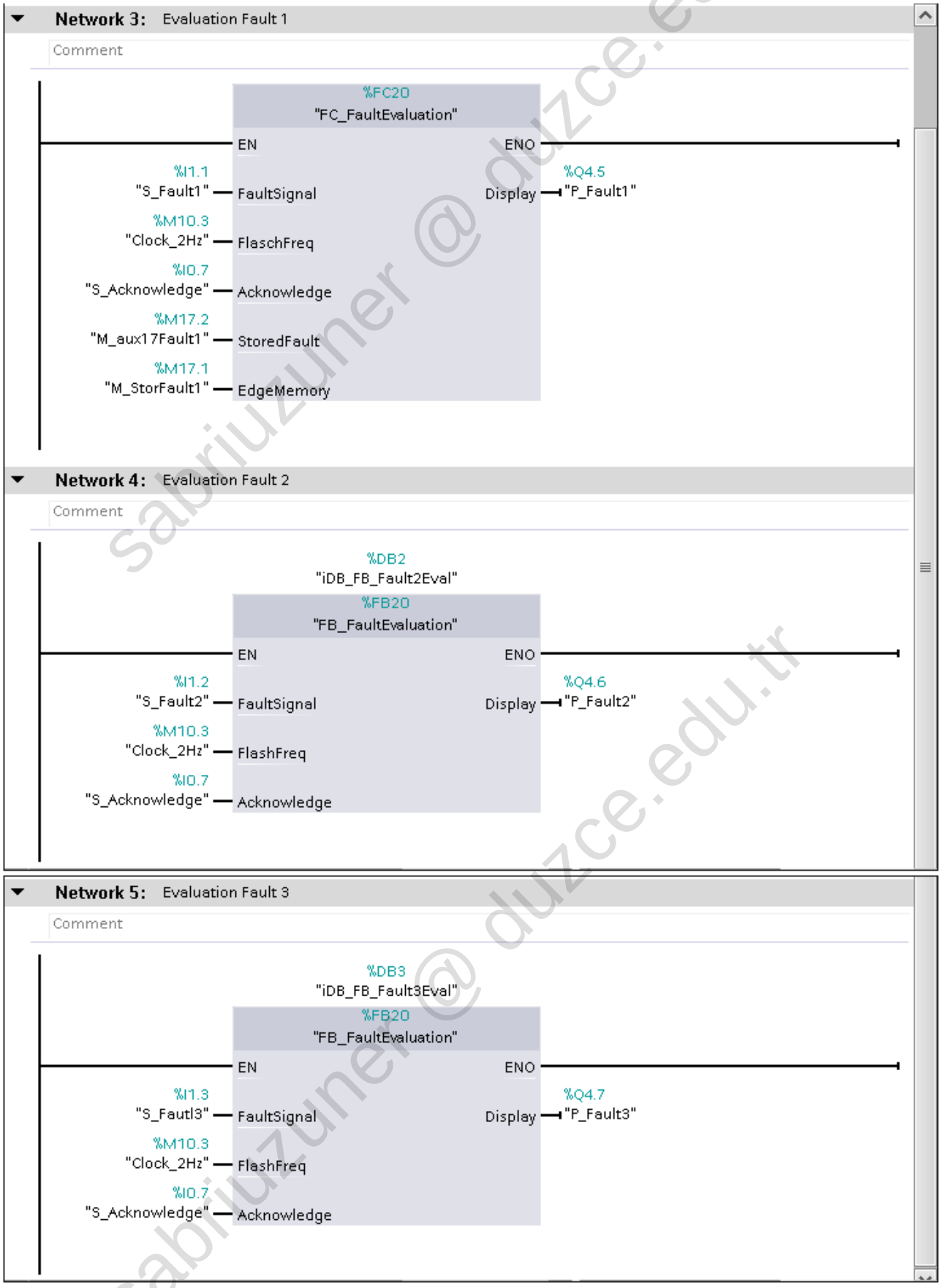


11.4. "FC_Fault"



Continuation on the next page

Continuation "FC_Fault"



11.5. "FC_FaultEvaluation"

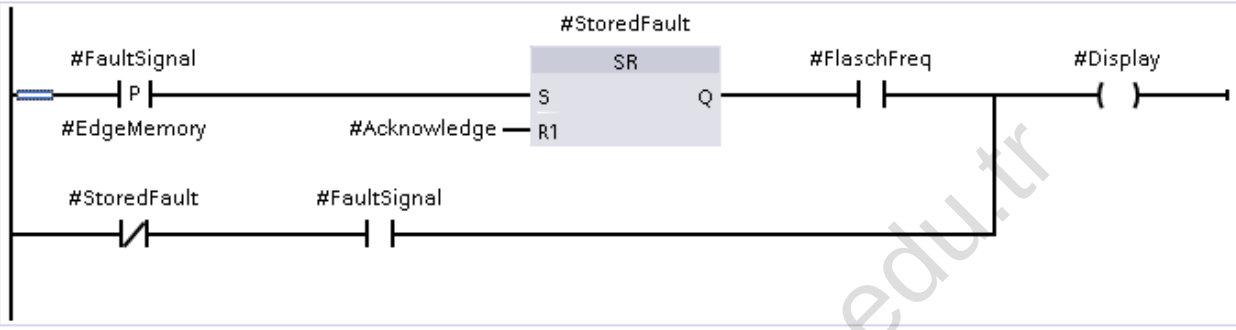
		Name	Data type	Default value	Comment
1		Input			
2		FaultSignal	Bool		
3		FlaschFreq	Bool		
4		Acknowledge	Bool		
5		Output			
6		Display	Bool		
7		InOut			
8		StoredFault	Bool		
9		EdgeMemory	Bool		
10		Temp			
11		<Add new>			
12		Constant			
13		<Add new>			
14		Return			
15		FC_FaultEvaluation	Void		

Block title: .....

Comment

Network 1: Fault evaluation

Comment



11.6. "FB_FaultEvaluation"

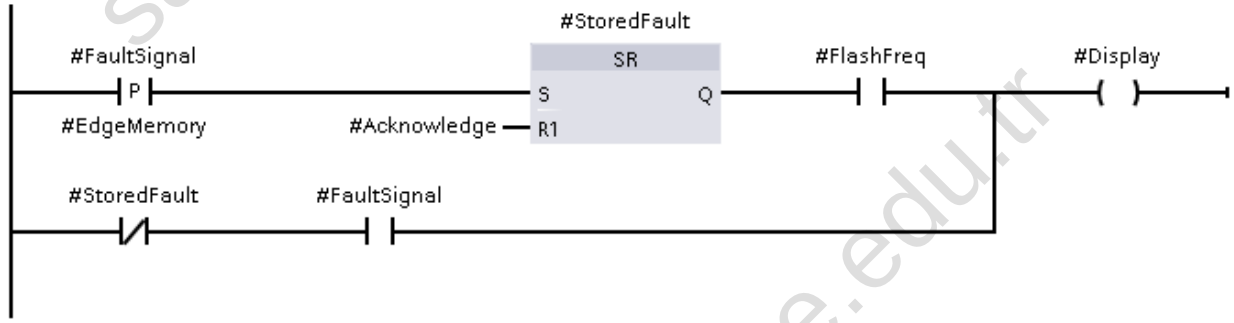
		Name	Data type	Default value	Retain	Accessible ...	Writa...	Visible in
1		Input						
2		FaultSignal	Bool	false	Non-ret...			
3		FlashFreq	Bool	false	Non-retain			
4		Acknowledge	Bool	false	Non-retain			
5		Output						
6		Display	Bool	false	Non-retain			
7		InOut						
8		<Add new>						
9		Static						
10		StoredFault	Bool	false	Non-retain			
11		EdgeMemory	Bool	false	Non-retain			
12		Temp						
13		<Add new>						
14		Constant						
15		<Add new>						

Block title: .....

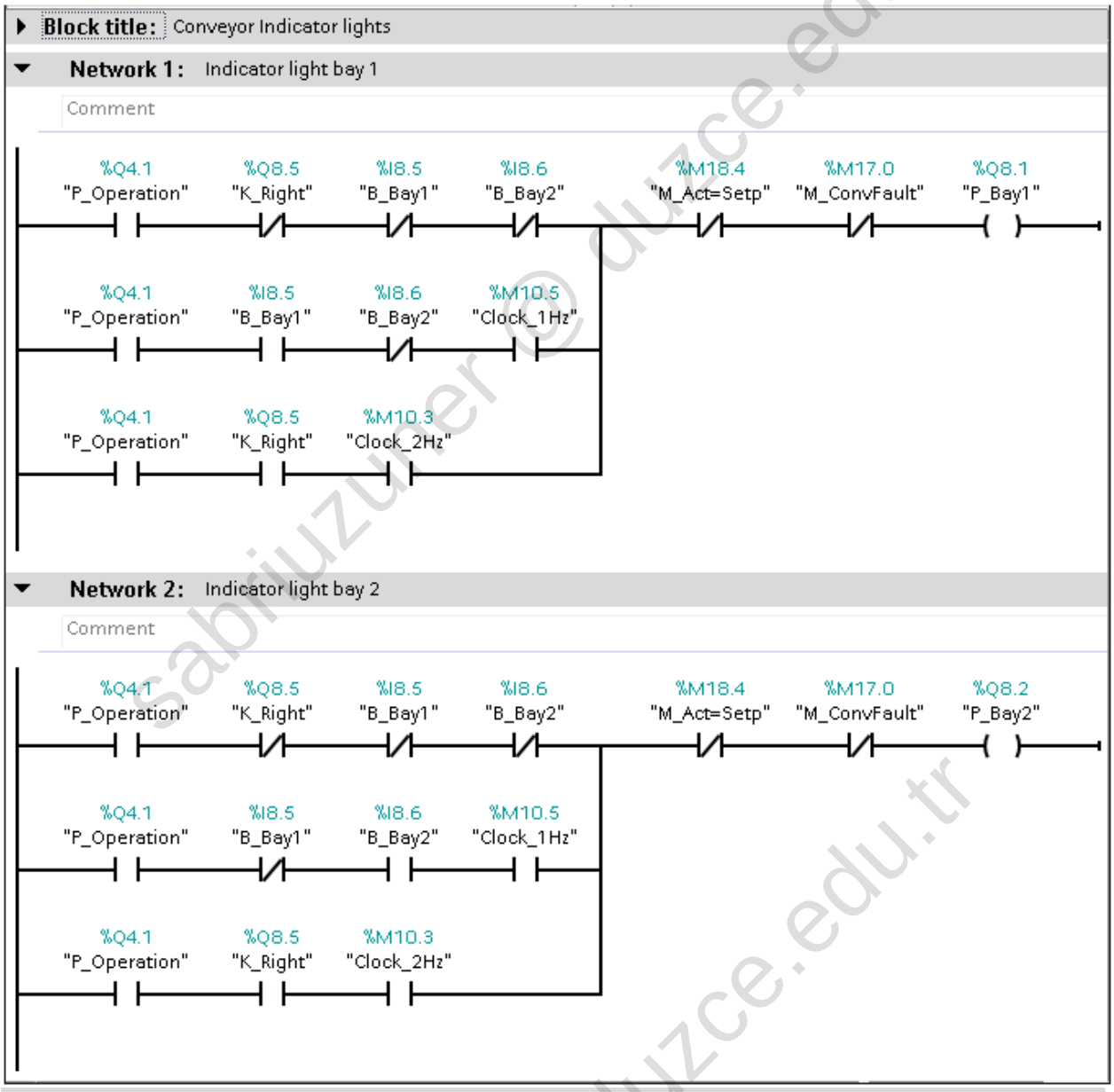
Comment

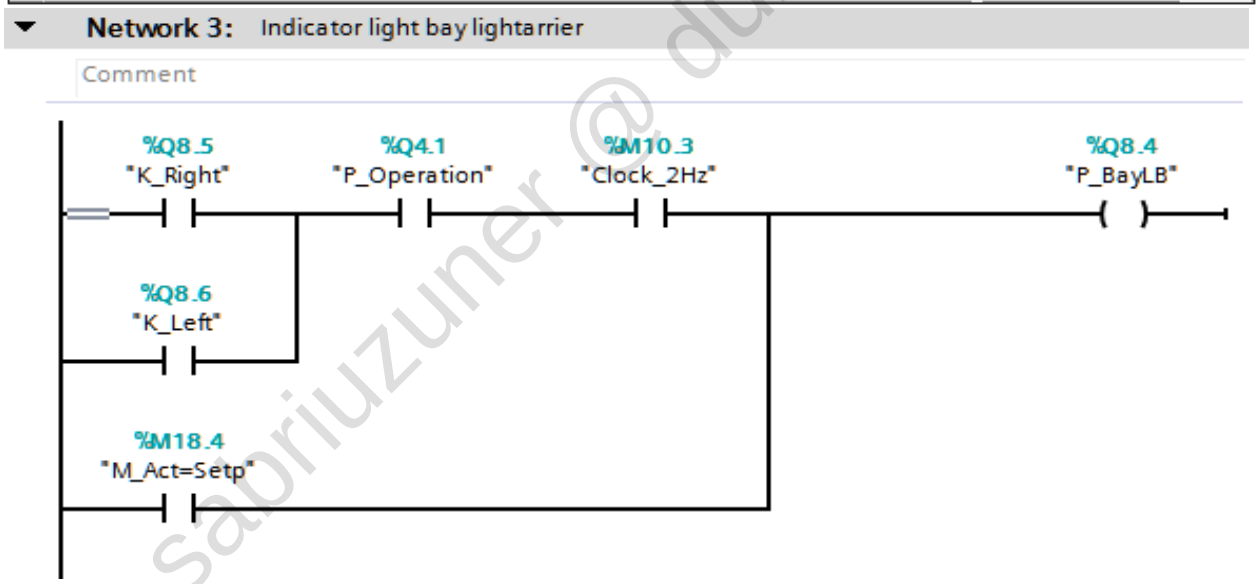
Network 1: Fault evaluation

Comment

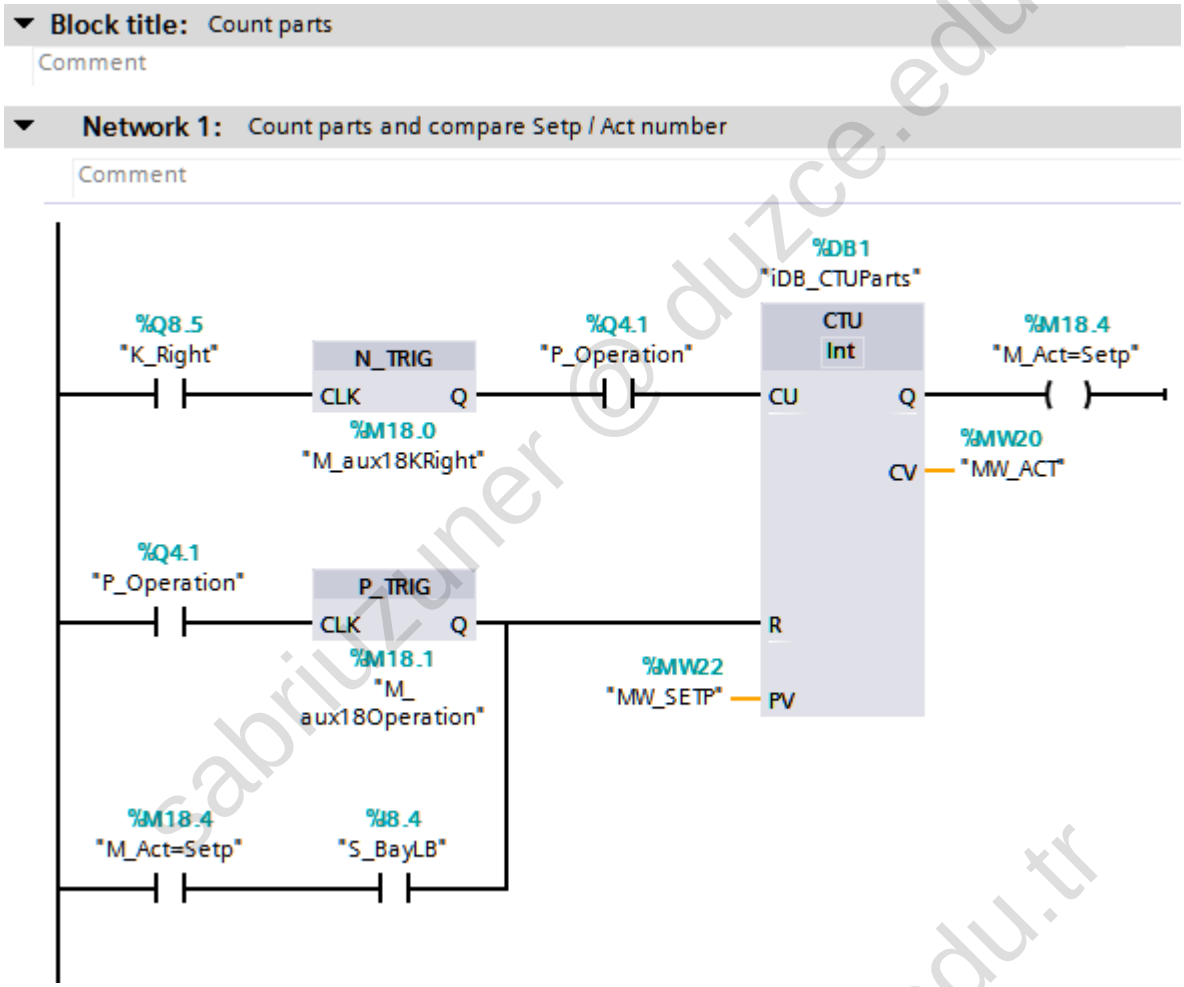


11.7. "FC_Signal"





11.8. "FC_Count"



Contents

12

12. Training and Support

12.1. 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.

12.2. 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.

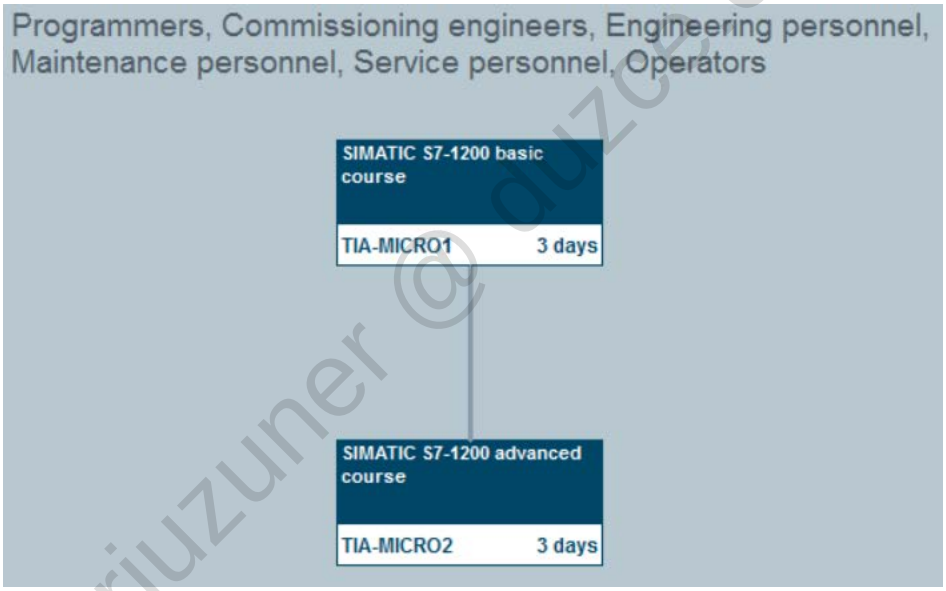
SIMATIC S7 TIA Portal

- 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

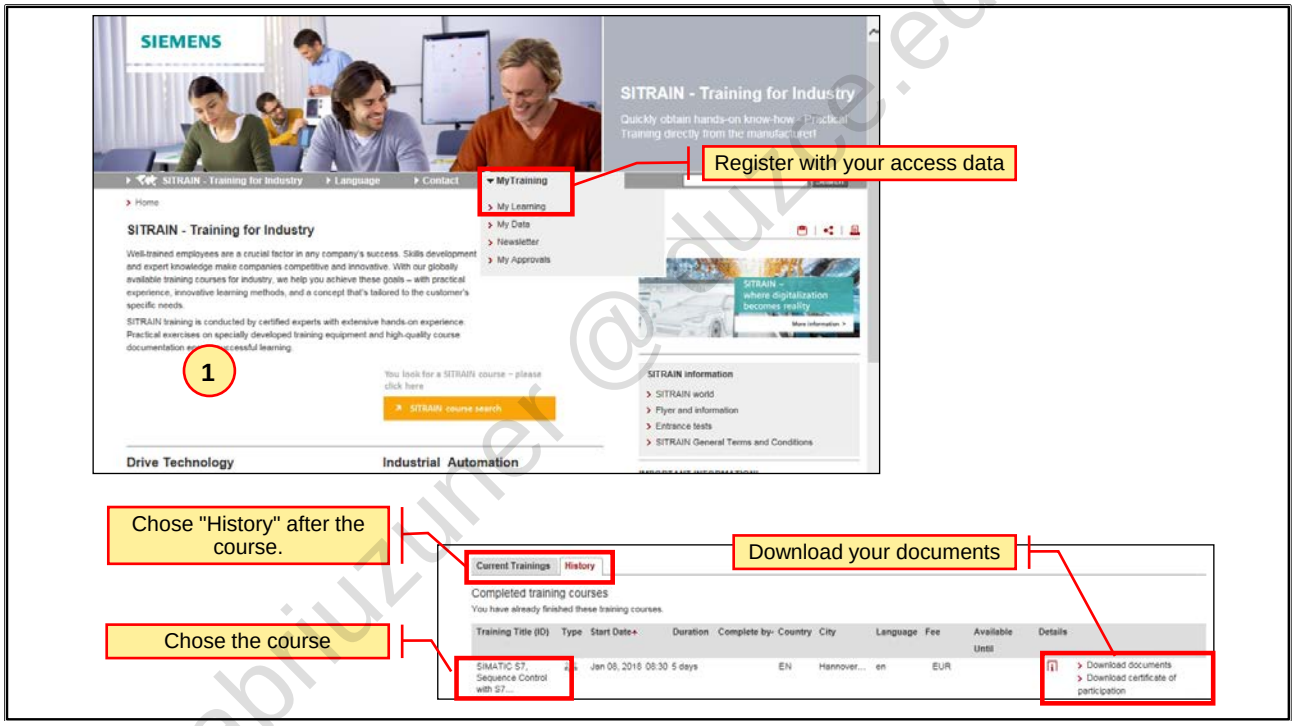
SIMATIC S7-300/-400 with STEP 7 V5.x

- 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

12.3. Learning path: SIMATIC S7 Prgramming in the TIA Portal

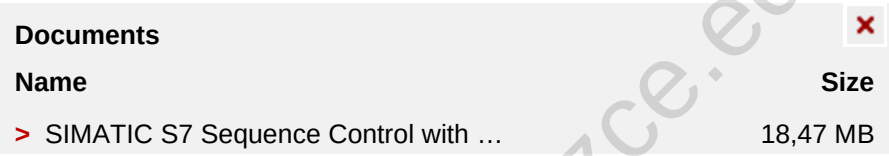


12.4. Download the training documents



If you want to download the training documents, proceed as follows:

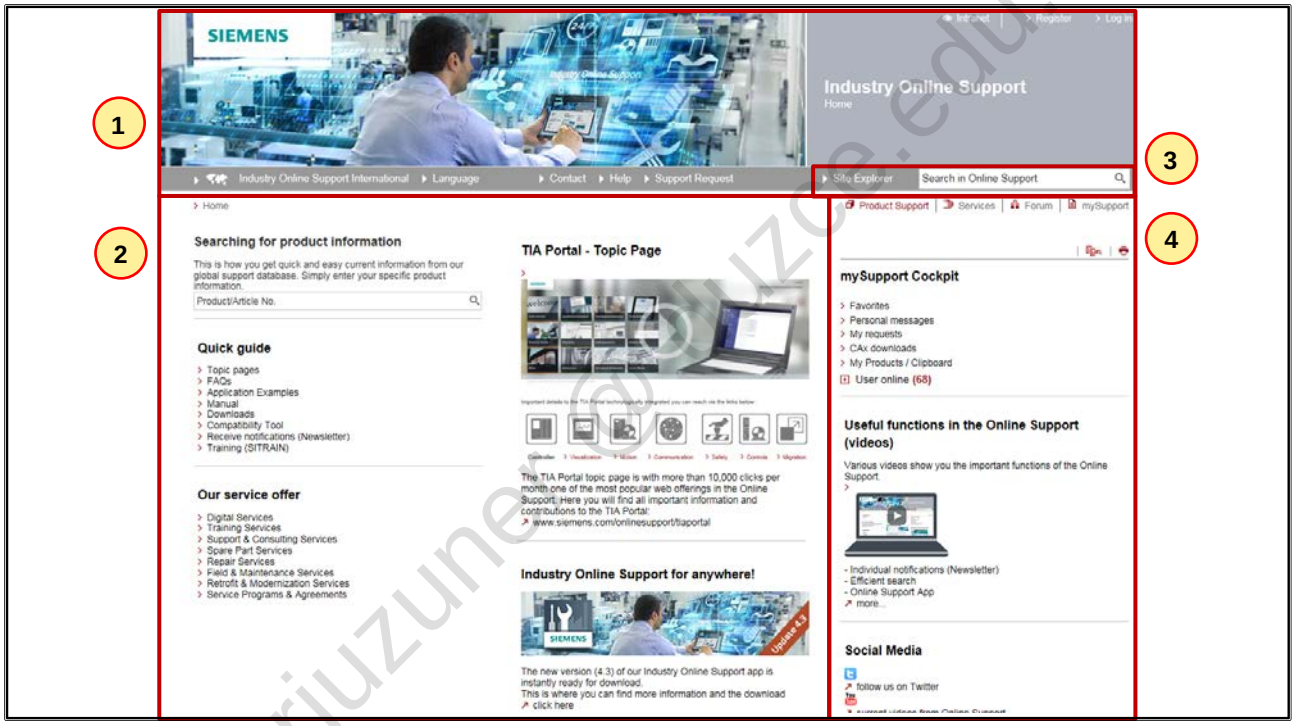
- Visit our new SITRAIN homepage at http://www.siemens.de/sitrain
- Register with your access data under the menu option **MyTraining**.
- Select **MyLearning** on the right-hand side of the submenu.
- Select your course and download your documents with a click on ”Download documents”.



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.

12.5. The Industry Online Support – the most important innovations



The most important functions are always in the same place on all the pages:

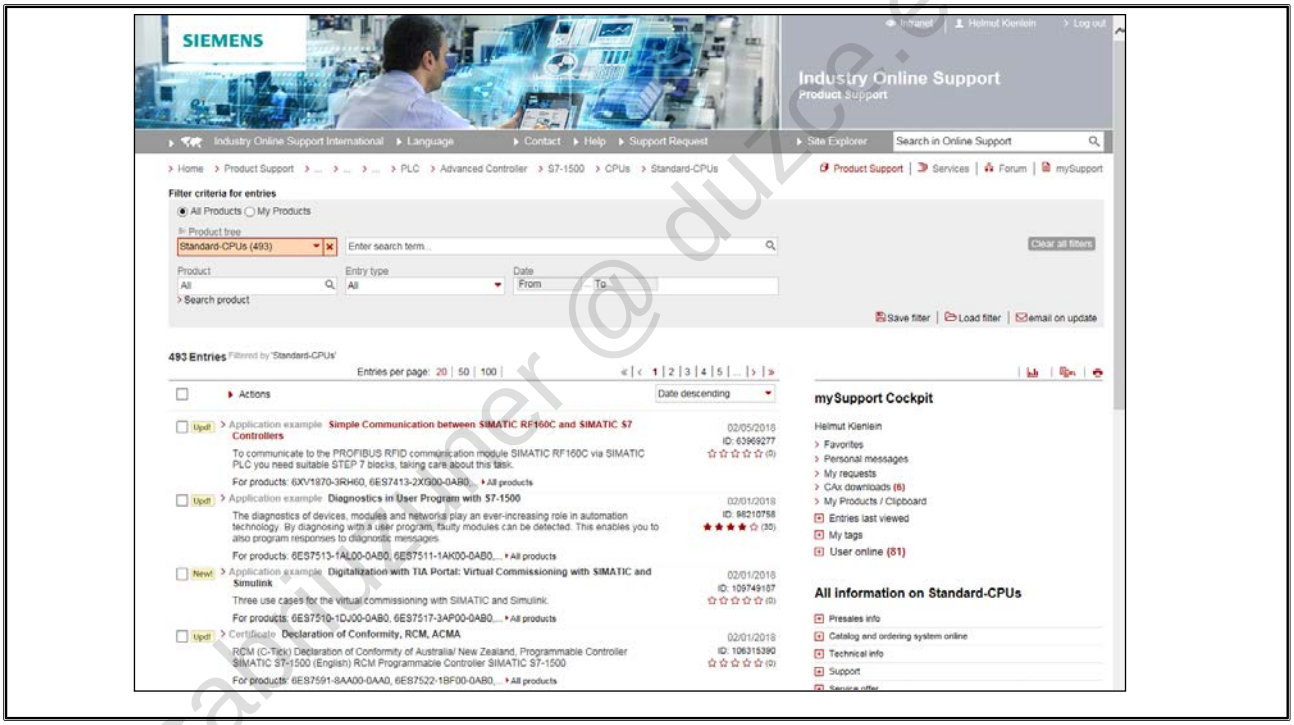
1 The menu bar links to the main areas of the site. You can subscribe and register at any time to benefit from the features the personalized mySupport option offers.

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.

12.6. The Principle of Navigation



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 though 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.

12.7. Complete product information

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:

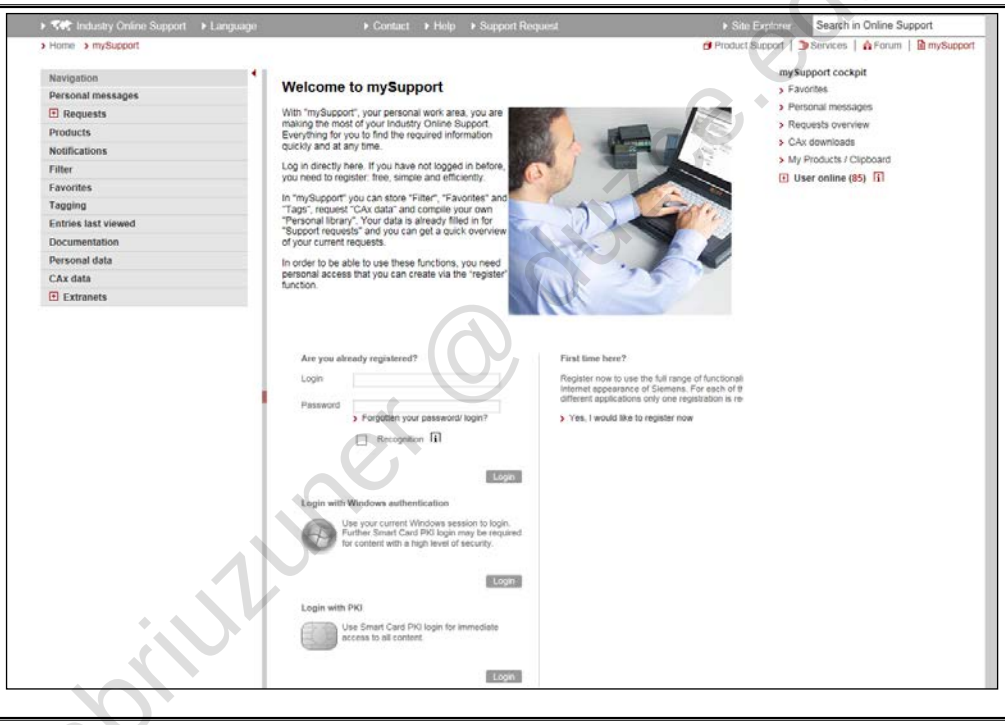
In the Product Support area, there is the central navigation bar.

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.

12.8. 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.

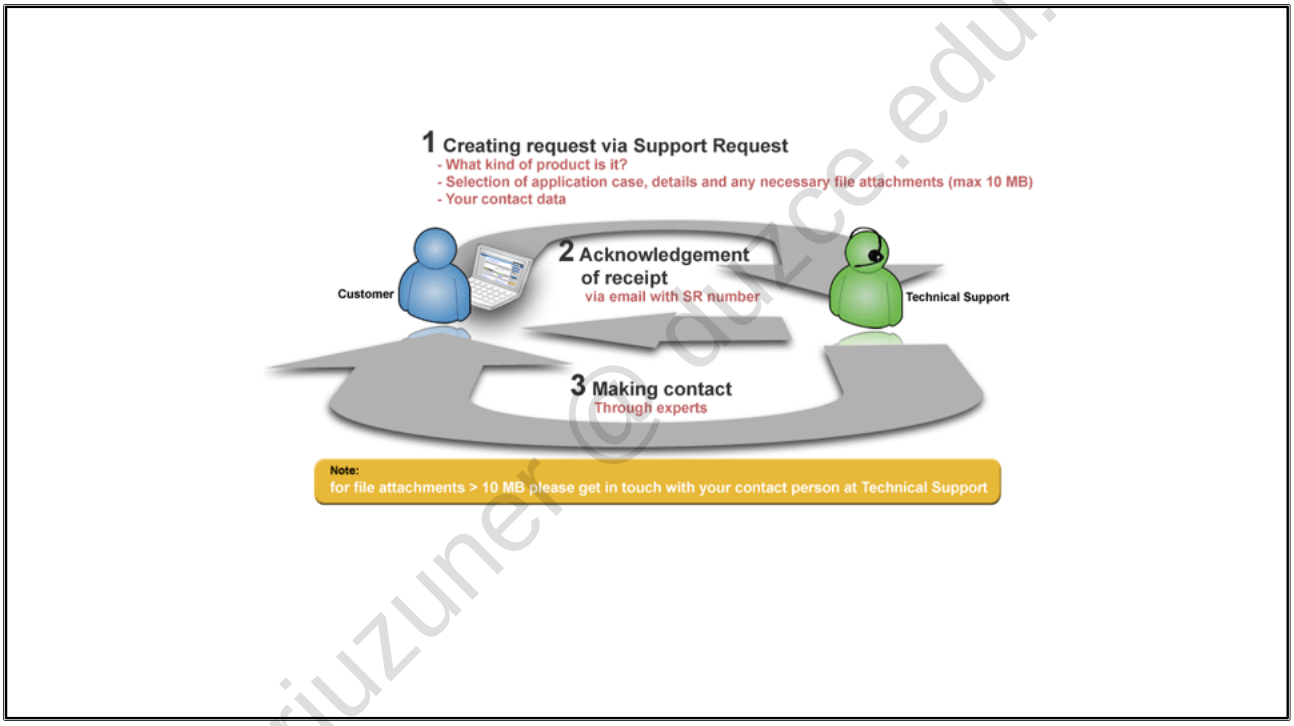
Moreover, you can now add notes, comments and tags (keywords). The system automatically creates a "Tag Cloud" based on your entries so you can access information quickly and easily by means of your own terms. The operation is consistent throughout mySupport so that you will easily find your way around. "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.

12.9. 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:

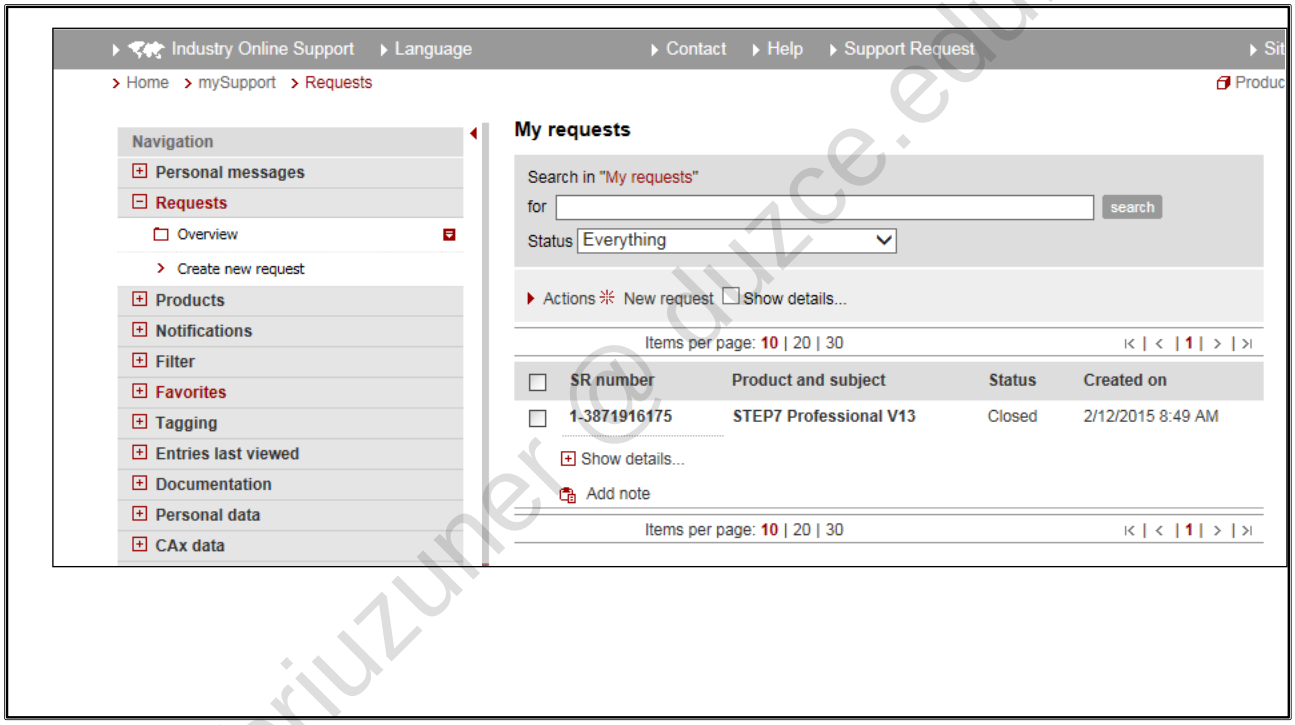
http://www.siemens.com/automation/support-request

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.

12.10. Support Request



12.11. 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.

12.11.1. Scanning product/EAN code

- After scanning a code, the product view will open in most cases.
- Specialized search options are available for configurable products.
- Depending on the product, the result can also be displayed as a list of products, in which
- the user can select the appropriate product.

12.11.2. Scan functionality

Data matrix codes

on Siemens products
as per standard SN60450

QR code

e.g.: in advertisements
relating to Siemens content

EAN13 bar code

on Siemens products

Code39 bar code

(very hard to recognize /
scan) on Siemens products
as per standard SN60450

The scan functionality in the Online Support app supports the following types of code:

Data matrix code

QR code

EAN13 bar code

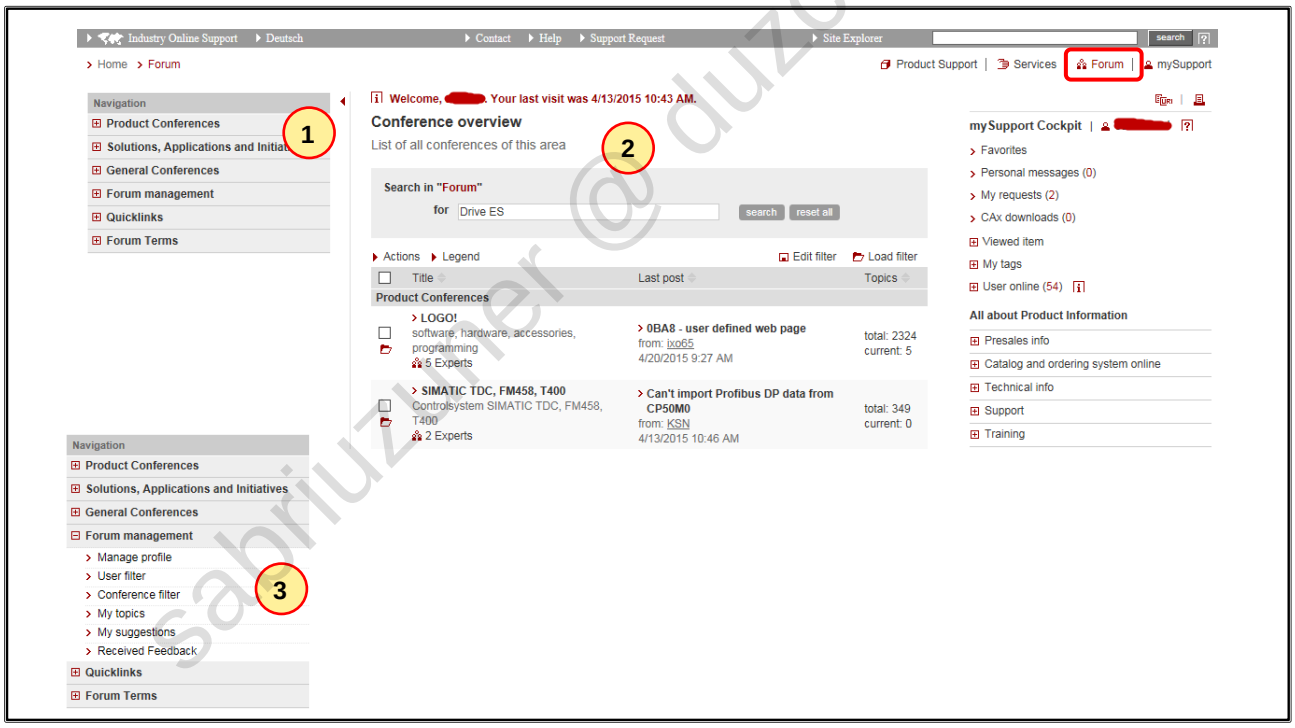
Code39 bar code

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).

12.12. Forum - the communication platform for Siemens Industry products

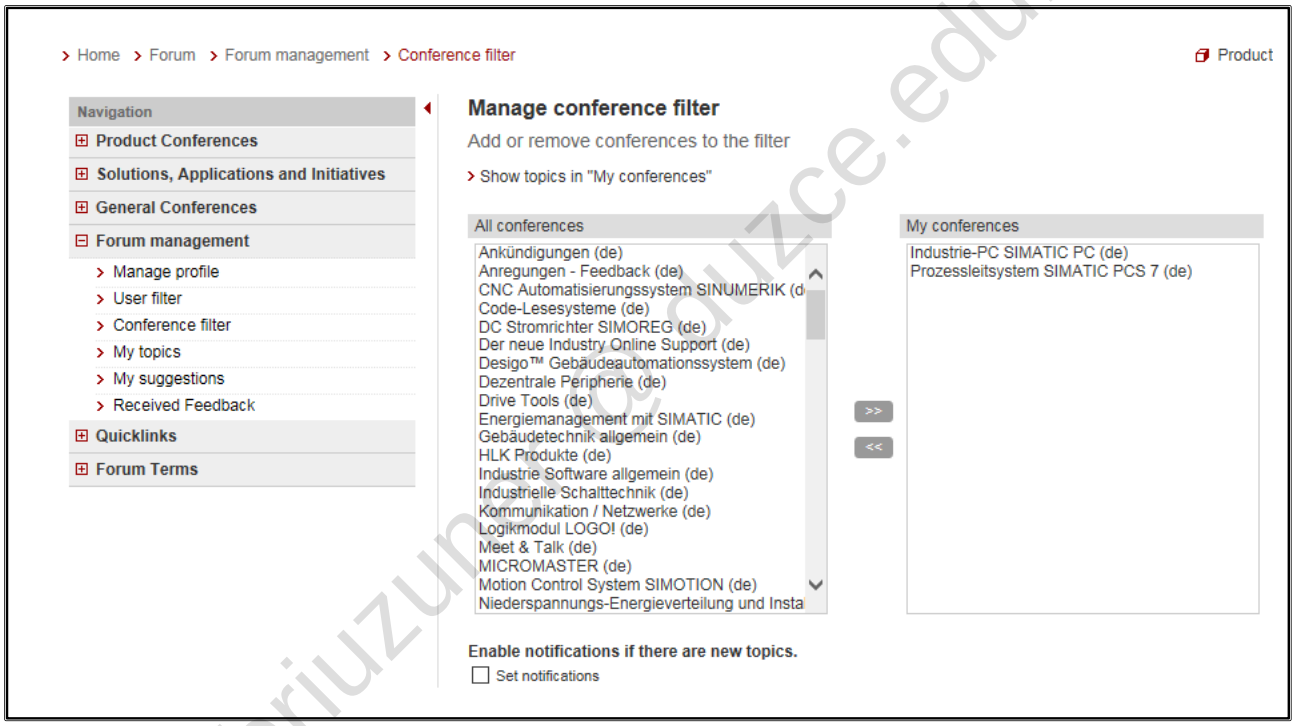
12.12.1. Conferences and Forum management



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.



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.

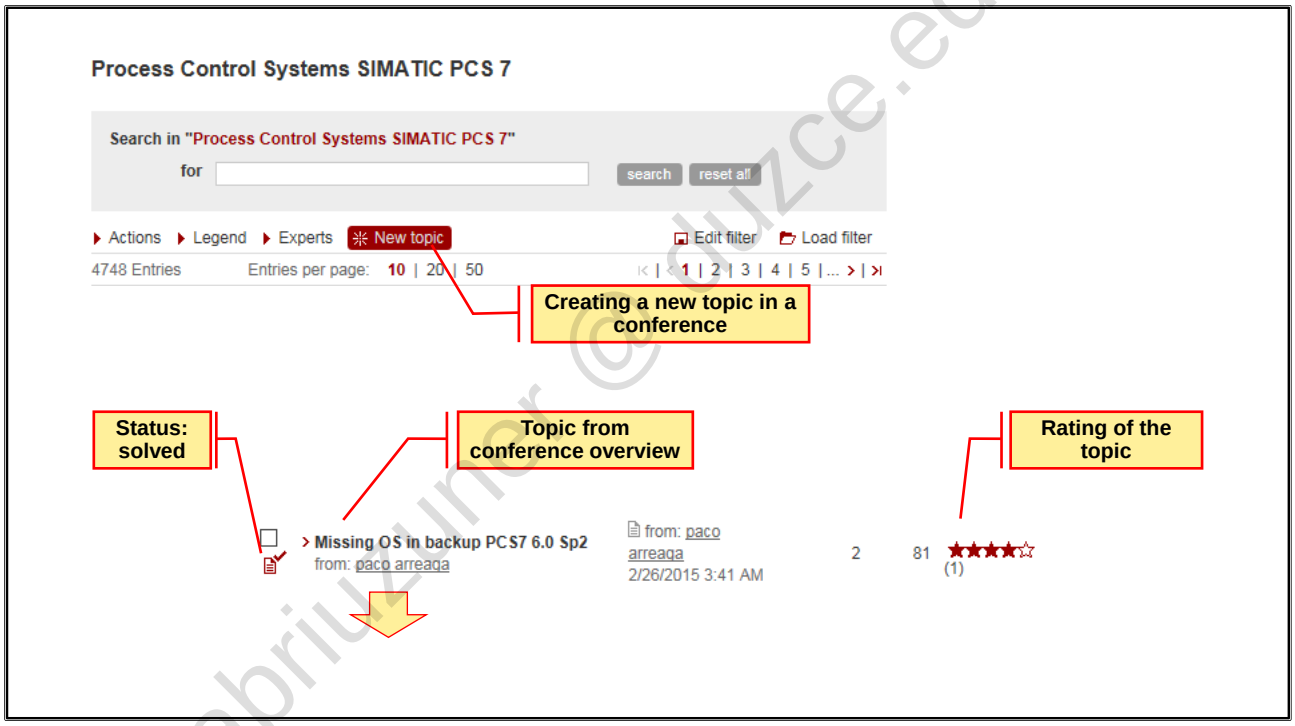
User filter

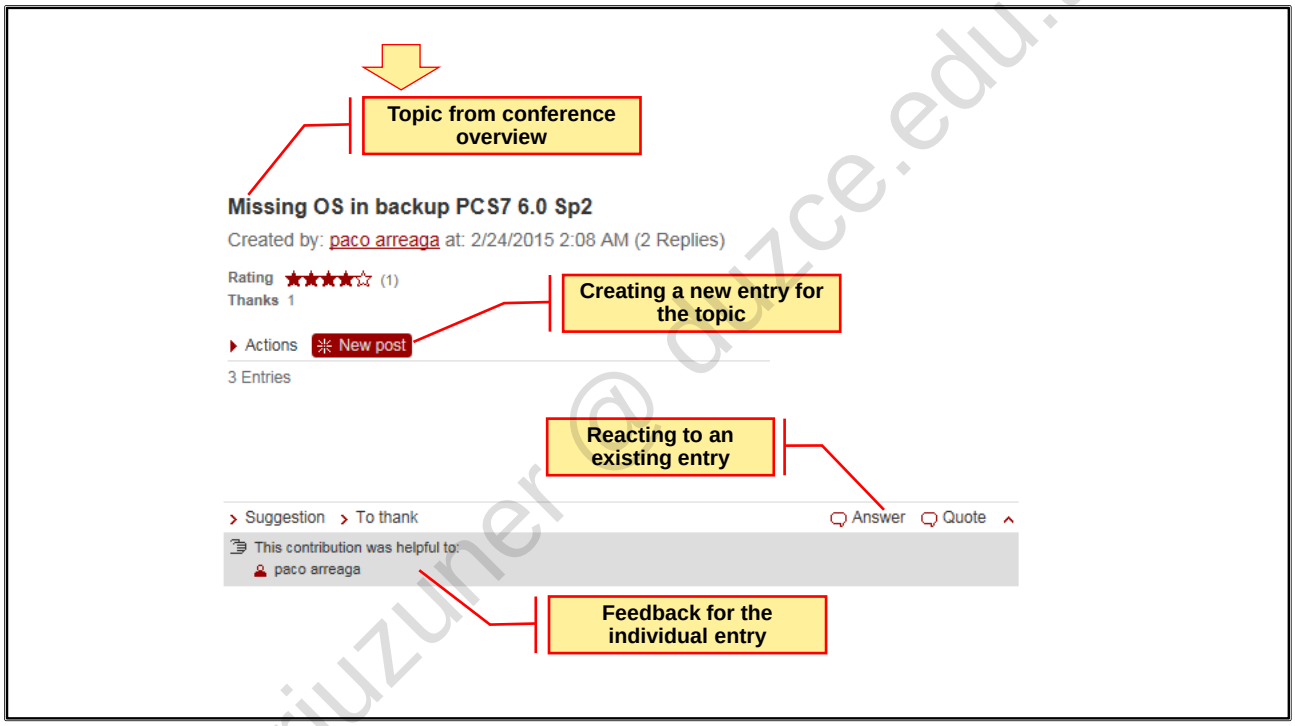
Have you found a user in the Technical Forum who posts entries that are particularly interesting? Then add this user to your list of "preferred users".

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.

12.12.2. Interactions in the Forum





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.

12.13. 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?**

Siemens AG
Digital Factory
SITRAIN – Digital Industry Academy
Postfach 48 48
90026 Nürnberg

siemens.com/sitrain