

**SIEMENS**

*Ingenuity for life*

Private copy for Sabri Uzuner, sabriuzuner@duzce.edu.tr

TIA-MICRO2

**SITRAIN**

**Digital Industry Academy**

SIMATIC S7- S7-1200 Advanced Course

[siemens.com/sitrain](https://www.siemens.com/sitrain)

# SIEMENS

## SITRAIN

Training for Industry

## SIMATIC S7 1200 Advanced Course

## Course TIA-MICRO2

Name: \_\_\_\_\_

Course from: \_\_\_\_\_ to: \_\_\_\_\_

Instructor: \_\_\_\_\_

Location: \_\_\_\_\_

This document was produced for training purposes. SIEMENS assumes no responsibility for its contents. The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable to damages.

Copyright © Siemens AG 2020. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

SITRAIN course offer on the Internet: [www.siemens.com/sitrain](http://www.siemens.com/sitrain)

Training Document Version: V16.00.00 (for STEP7 V16)

1 Training Devices

2 Commissioning Hardware and Software

3 Analog Value Processing

4 Data Blocks

5 Introduction to PROFINET

6 Introduction to Industrial Communication

7 Tags and Messages in HMI

8 Technology Objects

9 Troubleshooting

10 SCL

11 Training and Support

12

13

14

15

# Contents

# 1

<b>1.</b>	<b>Presentation of the Course Contents and Training Devices .....</b>	<b>1-2</b>
1.1.	Objectives .....	1-2
1.2.	Course Contents .....	1-3
1.3.	Training Area with S7-1200 .....	1-5
1.4.	Schematic Diagram Industrial Ethernet/ PROFINET Networking.....	1-6
1.5.	Configuration of the S7-1214 Training Device.....	1-7
1.6.	The Simulator.....	1-8
1.7.	The Conveyor Model.....	1-9
1.8.	PLC Tags .....	1-10

Private copy for Sabri Uzuner, sabriuzuner@duzce.edu.tr

sabriuzuner @ duzce.edu.tr

# 1. Presentation of the Course Contents and Training Devices

## 1.1. Objectives

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

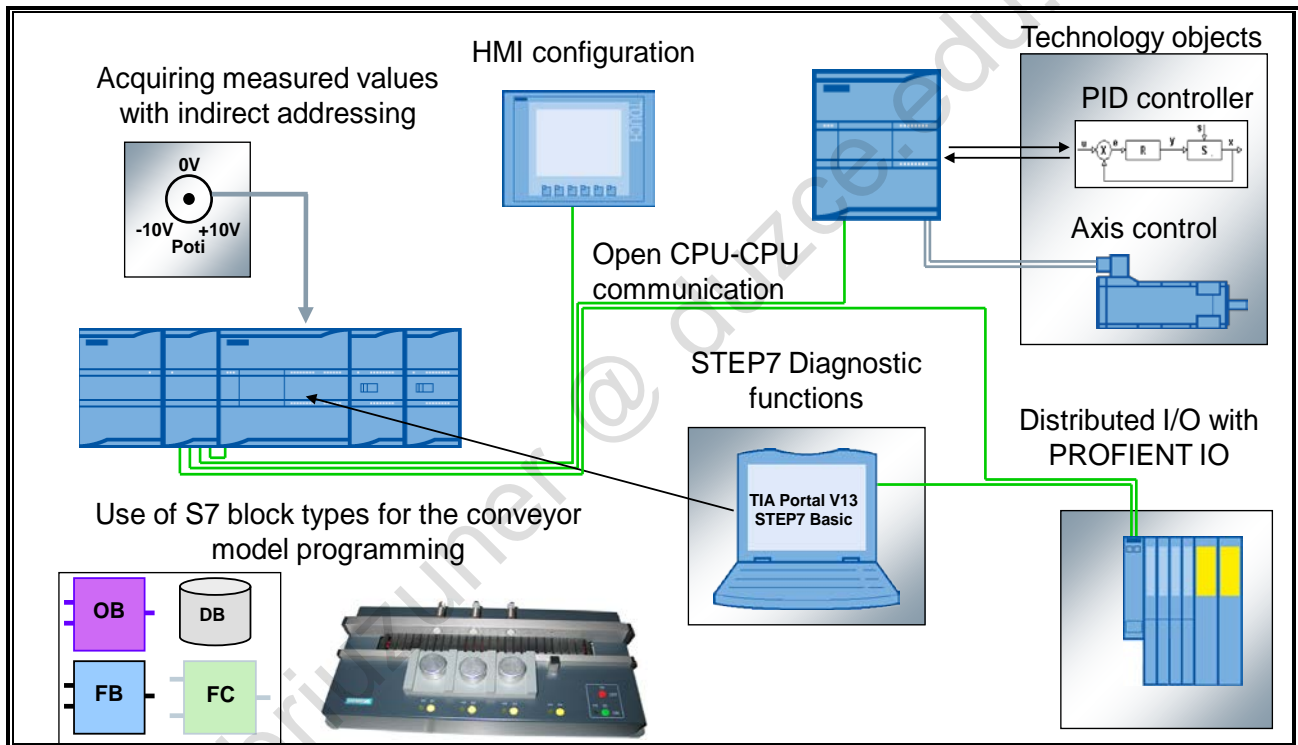
- ... be familiar with the main course contents
- ... be familiar with the training devices
- ... be familiar with the networking of the training devices

### Objectives

In this chapter, the main course contents and the training devices are presented.



## 1.2. Course Contents



### Course Contents

The following topics are dealt with in this course:

- Analog value processing**  
 Analog value processing is used to convert process values (for example, the measured values (measuring points) of a level sensor) into a "tangible" unit (e.g. m<sup>3</sup>), to then display it on an HMI, for example.
- S7 block types**  
 In this course, the S7 block types and their use are presented. Differentiation is made between logic (code) and data blocks. Data blocks are structured, user-defined data memory. They can be used for simple data storage or as interface to HMI devices. They can also be used to create complex database structures.  
 In the section Logic Blocks (FC, FB, OB), the properties and the fields of application are presented.
- Programming in SCL**  
 SCL is a further IEC programming language, in addition to LAD and FBD, for S7. Programming in SCL offers decisive advantages in the implementation of complex, mathematical calculations, as well as in the handling of large amounts of data.
- Indirect addressing**  
 Indirect addressing enables you to dynamically address memory cells within the PLC during program runtime. For example, measured value series can thus be formed by writing each new measured value into a different memory cell. Fundamental mechanisms for indirect addressing are available in LAD and FBD. The complete instruction set is available in SCL.

- **Introduction to PROFINET IO**  
PROFINET (similar to PROFIBUS) is used to connect distributed I/Os to the CPU. In this course, the fundamental addressing mechanisms and procedures for configuring distributed PROFINET field devices are presented.
- **Expanded HMI configuration**  
In addition to the basic functions of an HMI device presented in the Basic Course, the alarm message system for the display of discrete and analog alarms as well as the creation of input/output fields and the time-of-day synchronization between HMI device and PLC are dealt with in this course.
- **Integrated CPU technology objects**  
The S7-1200 offers integrated technology functions for motion control of axes and for PID control loops. The necessary steps for creating a technology object and the commissioning of a PID control loop and stepper motor are part of this course.
- **Troubleshooting with STEP7 (TIA Portal)**  
As the diagnostic functions of the TIA Portal are crucial for troubleshooting and system analysis, the available online and offline functions for quick and efficient elimination of arising faults are presented in this course.
- **Open CPU-CPU Ethernet communication**  
For data exchange between controllers, the S7-1200 is equipped with Ethernet communication concepts which are presented in more detail in this course and are also practiced.

### 1.3. Training Area with S7-1200

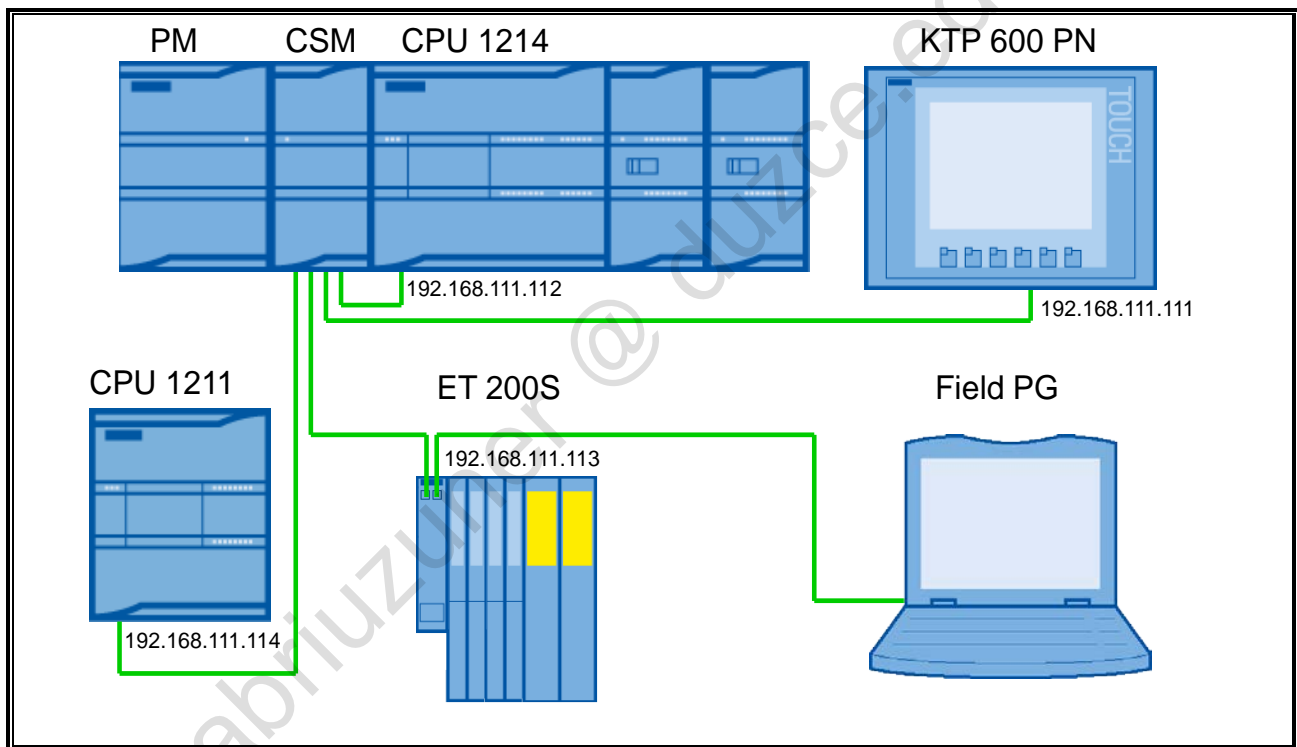


#### Training Area Setup

The training area for this course contains the following components:

- SIMATIC Field PG
- Training case with S7-1214, touchpanel and simulator
- Training case with S7-1211, stepper motor and control loop
- Training setup ET 200S as distributed I/O
- Conveyor model

## 1.4. Schematic Diagram Industrial Ethernet/ PROFINET Networking



### Networking of the Individual Components


The components of the training area are all networked to one another via (industrial) Ethernet. The RJ45 connection technology is the most widely used and can also be found in the home.

A point-to-point connection always exists between the components. This makes it necessary to use a switch (here in the form of a Compact Switch Module) which is used here as a network distributor.

The (industrial) Ethernet is the basis for the communication between the components. Depending on the type of communication, different protocols are used, for example:

- TCP communication between CPU and HMI and
- PROFINET IO communication between CPU and ET 200S

## 1.5. Configuration of the S7-1214 Training Device



The photograph shows a rack of SIMATIC S7-1214 modules. From left to right, the modules are: PM (Power Supply), CSM (Control Supply Module), CPU 1214C (Central Processing Unit), AI4/AO2 (Analog Input/Output), and DI8/DO8 (Digital Input/Output).

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		
	AI4 x 13 bits / AQ2 x 14 bits_1	2	96...103	96...99	SM 1234 AI4/AQ2	6ES7 234-4HE30-0XB0	V1.0		
	DI8/DQ8 x 24VDC_1	3	8	8	SM 1223 DI8/DQ8 x 24VDC	6ES7 223-1BH30-0XB0	V1.0		
		4							

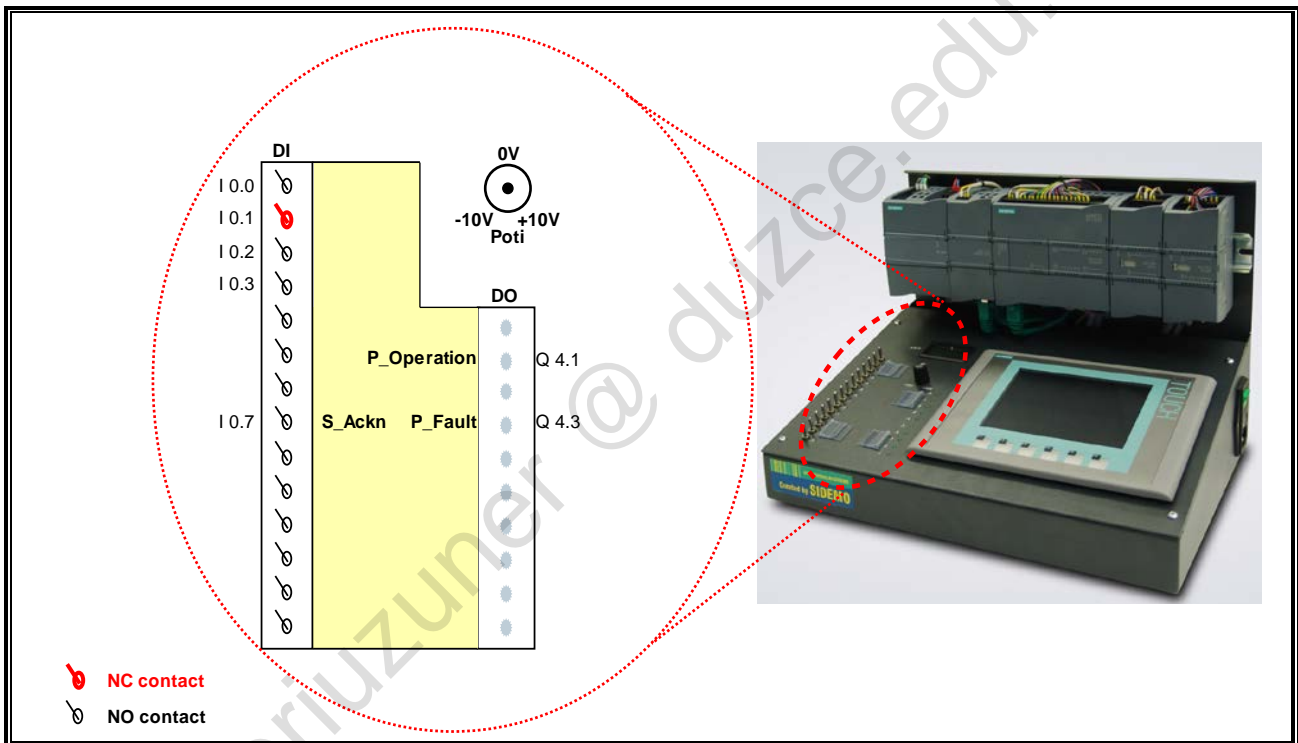
### Configuration of the S7-1214 Training Device

The picture shows the central module of the S7-1214 training case. The CPU has two signal modules (SMs) for digital and analog I/Os, as well as a signal board (SB) with an analog output as an expansion. The I/O addresses of the modules shown in the picture are already stored in the start project (→ next chapter) and do not have to be parameterized separately.

Module addresses at a glance:

- CPU 1214
  - DI14 → I 0.0 to I 1.5
  - DO10 → Q4.0 to Q5.1
  - AI2 → IW64, IW66
  - AO1 → QW80
- SM 1234
  - AI4 → IW96, IW98, IW100, IW102
  - AO2 → QW96, QW98
- SM1223
  - DI8 → I 8.0 to I 8.7
  - DO8 → Q8.0 to Q8.7

## 1.6. The Simulator



### The Simulator

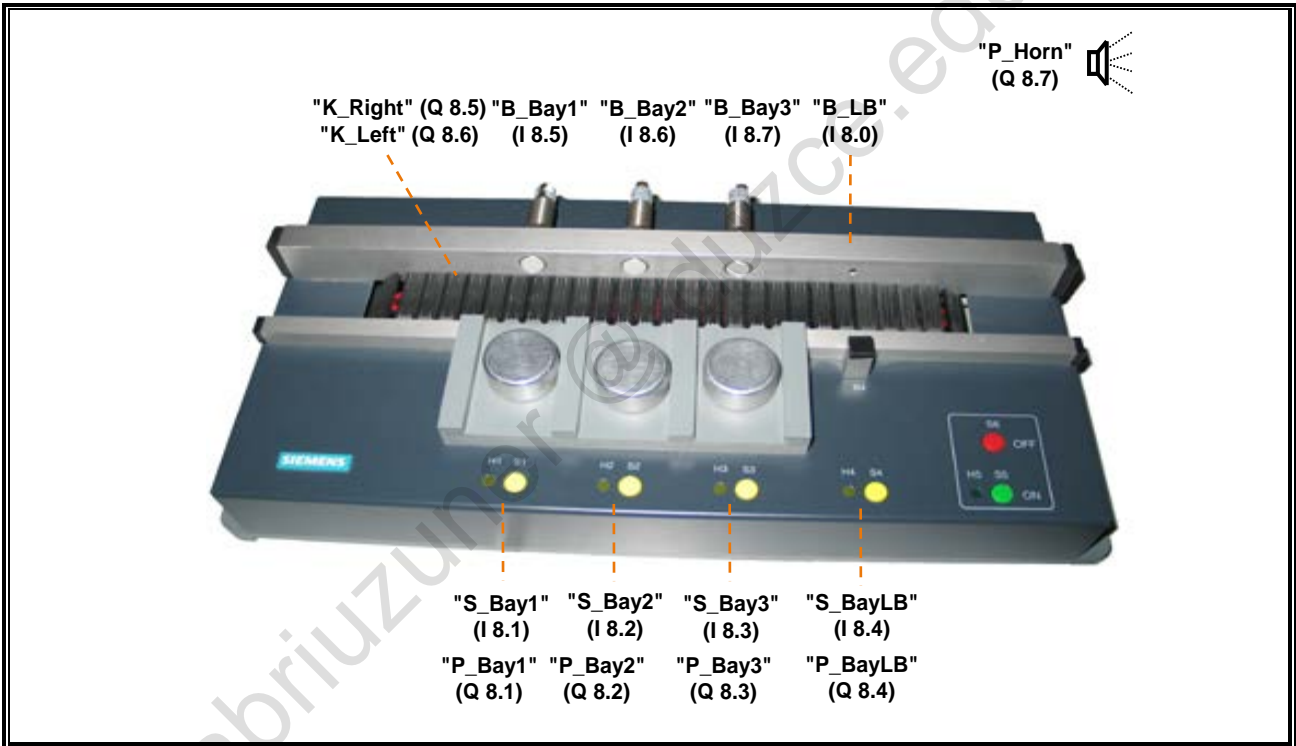
Together with the touchpanel, the simulator is used to operate the system. It consists of the following components:

- 14 switches, whereby the I 0.1 switch is an NC contact
- 10 LEDs
- Rotary potentiometer for setting or simulating analog input signals

The digital signals are connected to the I/Os of the CPU. The analog signal is processed by the SM 1234 analog module.



## 1.7. The Conveyor Model



### The Conveyor Model

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



# Contents

# 02

<b>2.</b>	<b>Commissioning the Hardware and Software .....</b>	<b>2-2</b>
2.1.	Objectives .....	2-2
2.2.	Task Description: The Conveyor Model as Distribution Conveyor .....	2-3
2.3.	Types of Program Blocks .....	2-4
2.4.	Possibilities for Program Structuring .....	2-5
2.5.	Process Images .....	2-6
2.6.	Cyclic Program Execution .....	2-7
2.7.	Data Exchange between Touchpanel and CPU .....	2-9
2.8.	Task Description: Commissioning the Training Case .....	2-10
2.8.1.	Exercise 1: Deleting Old Projects .....	2-11
2.8.2.	Exercise 2: Establishing an Online Connection to the CPU .....	2-12
2.8.3.	Exercise 3: Resetting the CPU to Factory Settings .....	2-13
2.8.4.	Exercise 4: Opening existing Project and save it with new name .....	2-14
2.8.5.	Exercise 5: Checking and, if necessary, adjusting the device configuration .....	2-15
2.8.6.	Exercise 6: Downloading the device configuration and user program into the CPU .....	2-16
2.8.7.	Exercise 7: Setting the IP address of the touchpanel .....	2-17
2.8.8.	Exercise 8: Transferring the touchpanel project .....	2-18
2.8.9.	Exercise 9: Function test touchpanel project and CPU program .....	2-19
2.8.10.	Exercise 10: Selecting the Editing Language .....	2-20
2.8.11.	Exercise 11: Runtime Settings .....	2-21
2.9.	Additional Information .....	2-22
2.9.1.	Industrial Ethernet: IP Address and Subnet Mask .....	2-23
2.9.2.	Online Access: Assigning an IP Address for the PG .....	2-24
2.9.3.	OB – Organization Blocks .....	2-25
2.9.4.	Events which Start an OB .....	2-26
2.9.5.	Events which Cannot Start an OB .....	2-27
2.9.6.	Interrupting the Cyclic Program .....	2-28
2.9.7.	DB – Data Block .....	2-29
2.9.8.	FC – Function .....	2-30
2.9.9.	FB – Function Block .....	2-31
2.9.10.	Adding a New Block .....	2-32
2.9.11.	Block programming .....	2-33
2.9.12.	Block Calls .....	2-34
2.9.13.	Block Groups .....	2-35
2.9.14.	Compiling a Block .....	2-36
2.9.15.	Downloading Blocks into the CPU .....	2-37
2.9.16.	Monitoring a Block .....	2-38
2.9.17.	Block Networks .....	2-39

## 2. Commissioning the Hardware and Software

### 2.1. Objectives

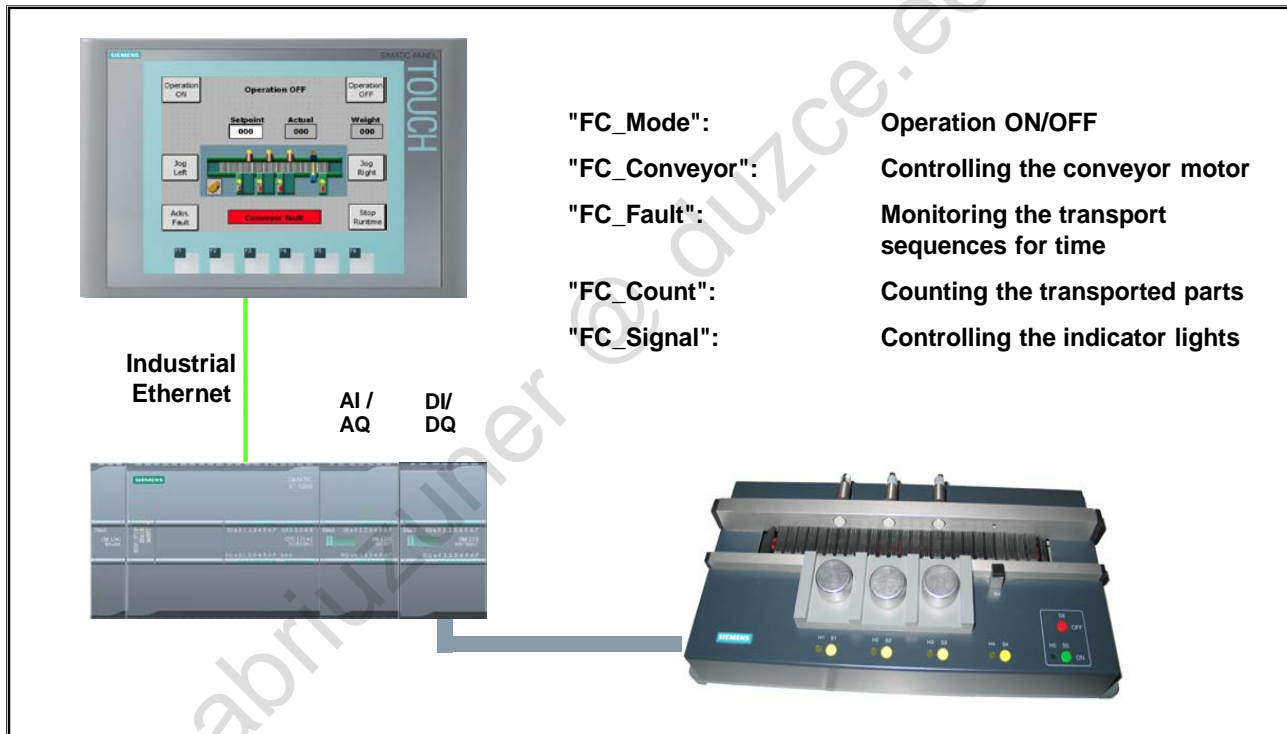
At the end of the chapter the participant will ...

- ... be familiar with the different S7 block types
- ... be familiar with the principle of "structured programming"
- ... be familiar with the meaning of the process image tables (PII, PIQ)
- ... be able to explain the principle of cyclic program execution
- ... be able to establish an online connection to the controller
- ... be able to create a hardware station and parameterize it
- ... be able to commission an existing PLC project
- ... be able to commission a touchpanel

#### Objectives

Important basics from the TIA-MICRO1 course are first of all repeated in this chapter. Then, an already configured system with described basic functions is commissioned.

## 2.2. Task Description: The Conveyor Model as Distribution Conveyor



### Task Description

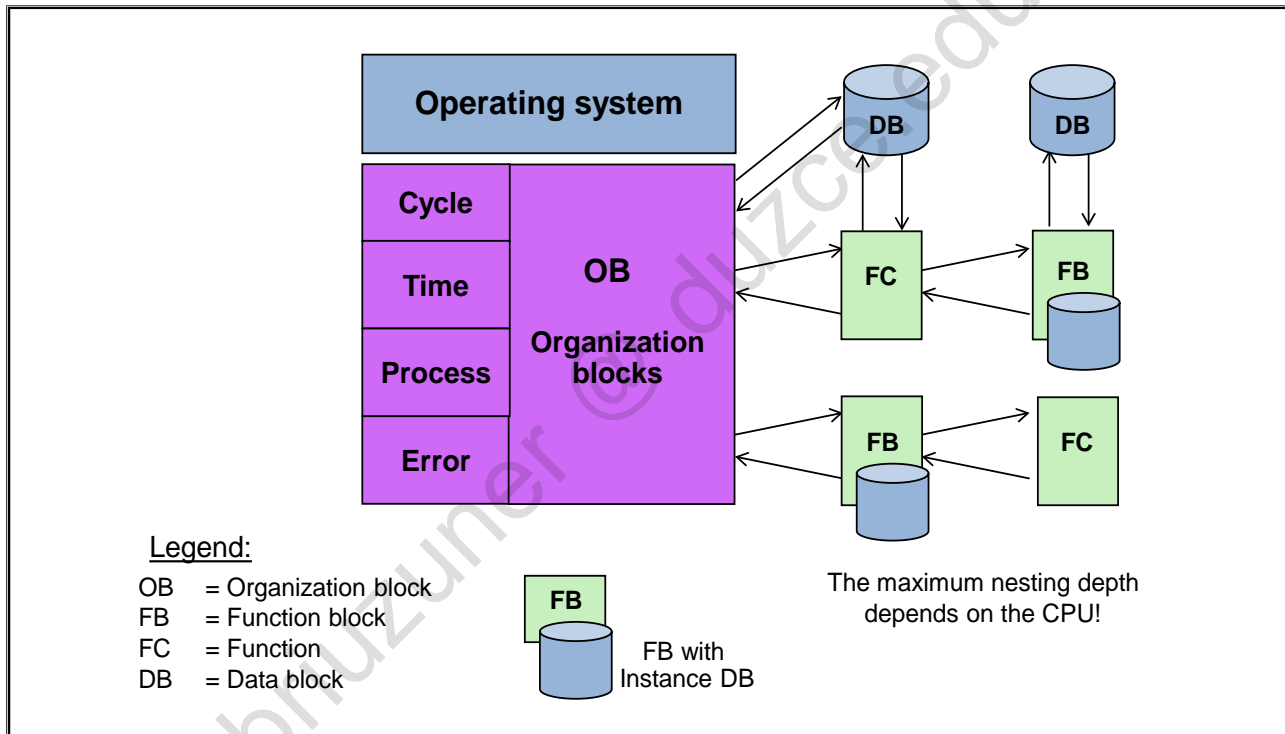
In this chapter, the conveyor model is commissioned as a distribution conveyor. For this, an appropriate exercise is carried out at the end of the chapter.

A CPU and touchpanel are already generated in the initial project. The user program of the CPU is structured as follows:

- "FC\_Mode"  
Switching on and switching off the operation is implemented in this function.
- "FC\_Conveyor"  
The control of the conveyor motor for the states Operation ON and Operation OFF is programmed in this function.
- "FC\_Fault"  
In this function, the conveyor is monitored for time. If a transport sequence takes longer than 6 seconds, the conveyor is stopped and an error message is triggered.
- "FC\_Count"  
Counting the already transported parts is programmed in this function. Setpoint and actual quantity are specified or read out via the touchpanel.
- "FC\_Signal"  
The control of the indicator lights during operation is programmed in this function.

The conveyor model can be completely controlled via the touchpanel. The functions necessary for this are already stored in the HMI project.

## 2.3. Types of Program Blocks



### Blocks

The programmable logic controller 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)

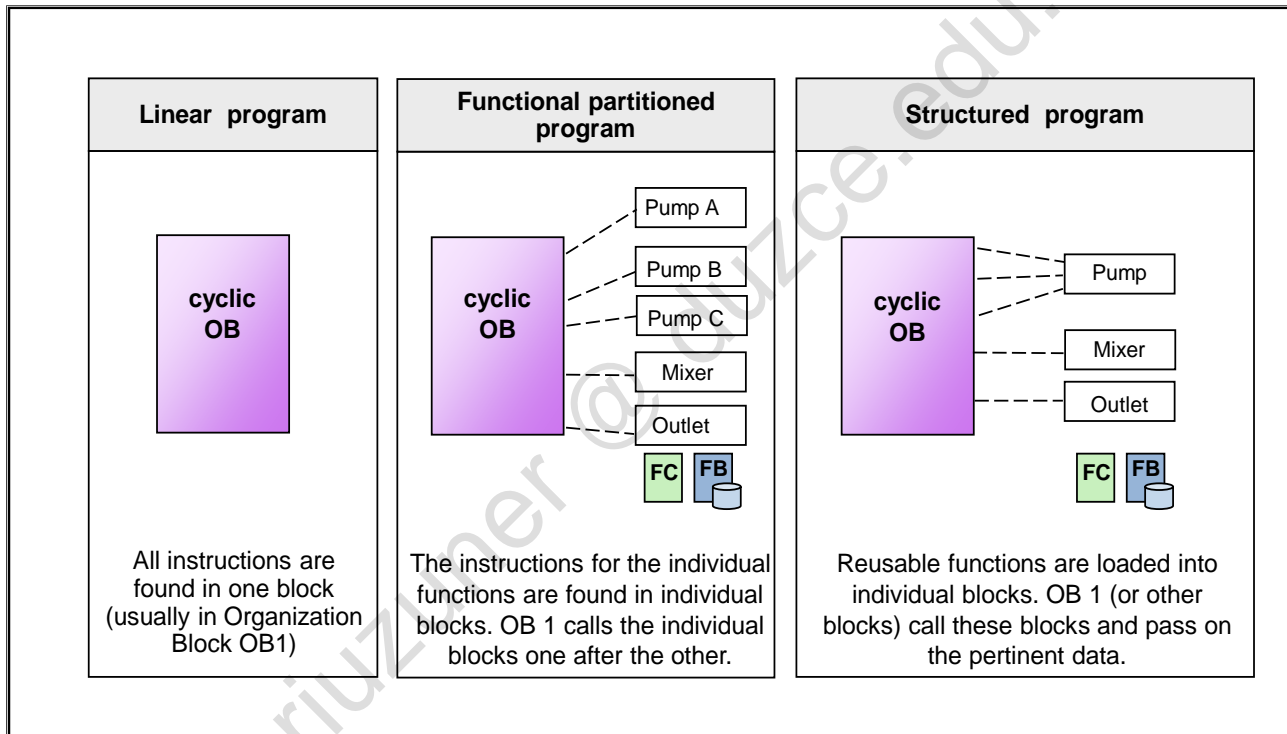
A function (FC) contains a partial functionality of the program. It is possible to program functions as parameter-assignable so that when the function is called it can be assigned parameters. As a result, functions are also suited for programming frequently recurring, complex partial functionalities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



## 2.4. Possibilities for Program Structuring



### Linear Programming

You can solve small automation tasks by writing the entire user program linearly in one cycle OB. This is only recommended for simple programs.

### Partitioned Programming

The program is partitioned into blocks and each block contains individual functions. Inside the blocks the code may be structured with networks. Usually the cyclic OB1 only calls other blocks.

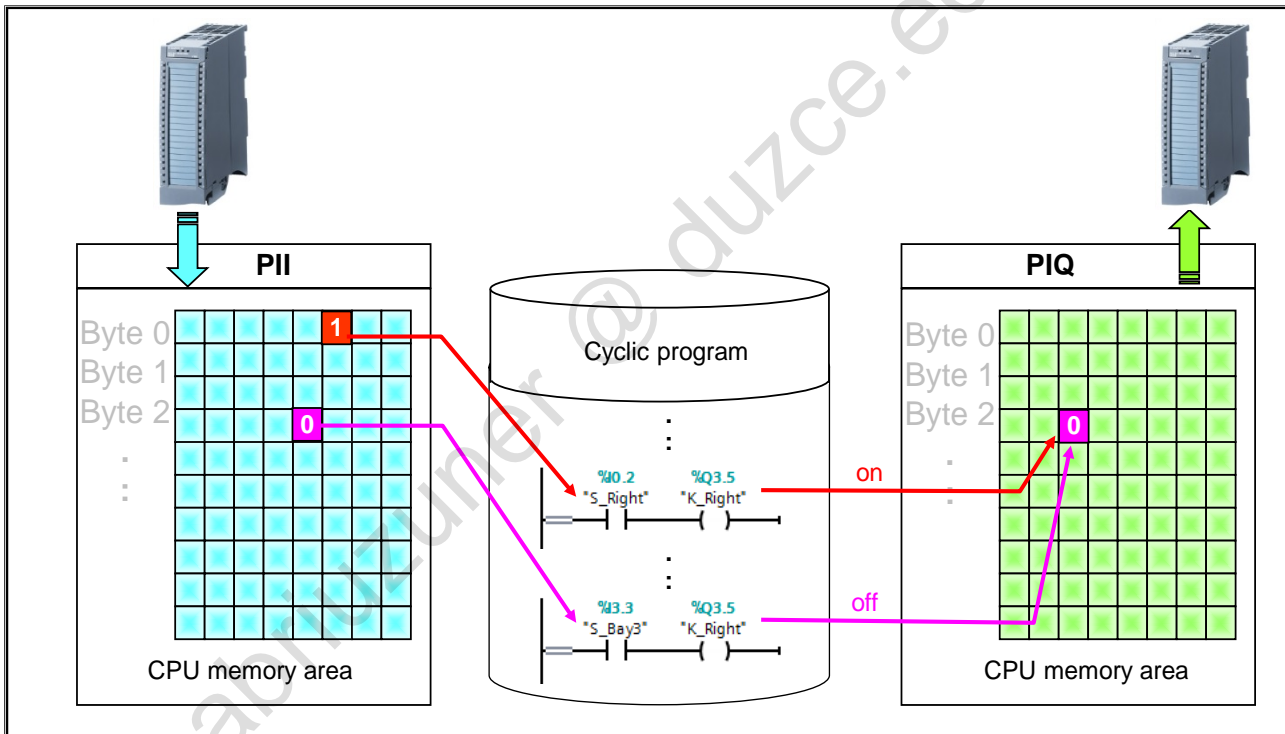
### Structured Programming

Complex automation tasks can be implemented and maintained more easily when they are divided into smaller partial tasks which reflect the technological functions of the automation process or if they are to be used repeatedly. In the user program, these partial tasks are represented by appropriate program parts, the blocks. Each block is an independent segment of the user program.

### Advantages

- Extensive programs can be clearly programmed
- Individual program parts can be standardized
- The program organization is simplified
- Changes to the program can be carried out more easily
- The program test is simplified because it can be made section by section
- Commissioning is made easier

## 2.5. Process Images



### Process Images

To store all input and output states, the CPU has reserved memory areas: the process image inputs (PII) and process image outputs (PIQ). During program execution, the user program then accesses these memory areas and not the digital inputs and output modules directly.

### Process image inputs (PII)

The process image inputs (PII) is the memory area in which the states of inputs is stored. The PII is read in from the input modules at the start of the cycle. When an input is linked in the user program, the state of this input stored in the PII is linked. This cannot change within a cycle, because the PII can only be updated or read in at the beginning of a cycle. This ensures that the same result will be obtained if an input is queried several times within a cycle.

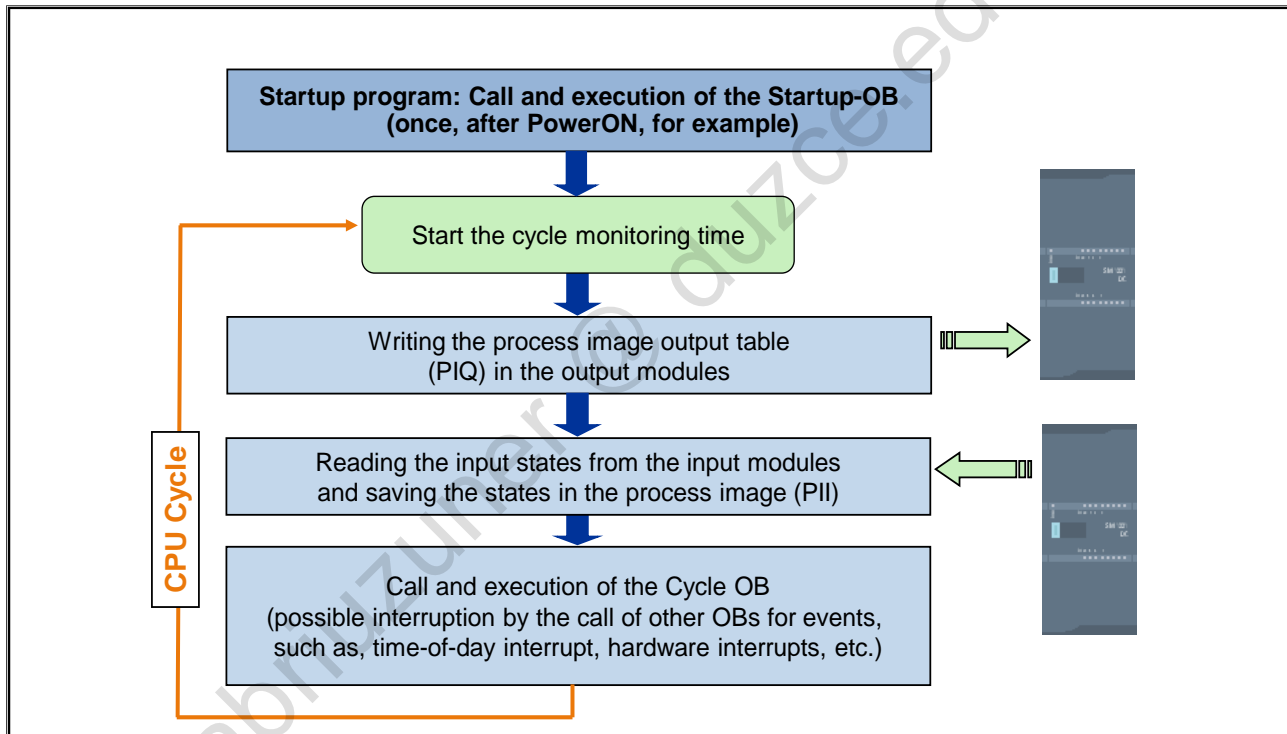
### Process image outputs (PIQ)

The process image outputs (PIQ) is the memory area in which the states of outputs are stored. The PIQ is put out to the output modules at the beginning of the cycle. Outputs can be assigned as well as queried in the program. This ensures the process control only takes effect after processing the cycling program.

### “Classic error” → double assignment of outputs

If a state is assigned to an output 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.

## 2.6. 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 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 next cycle starts 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 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 input / output modules
- Checking the elapsed cycle time

## Cycle time and cycle monitoring time

The time that the CPU requires for a complete cycle is the cycle time. It is monitored by the operating system of the CPU in every cycle.

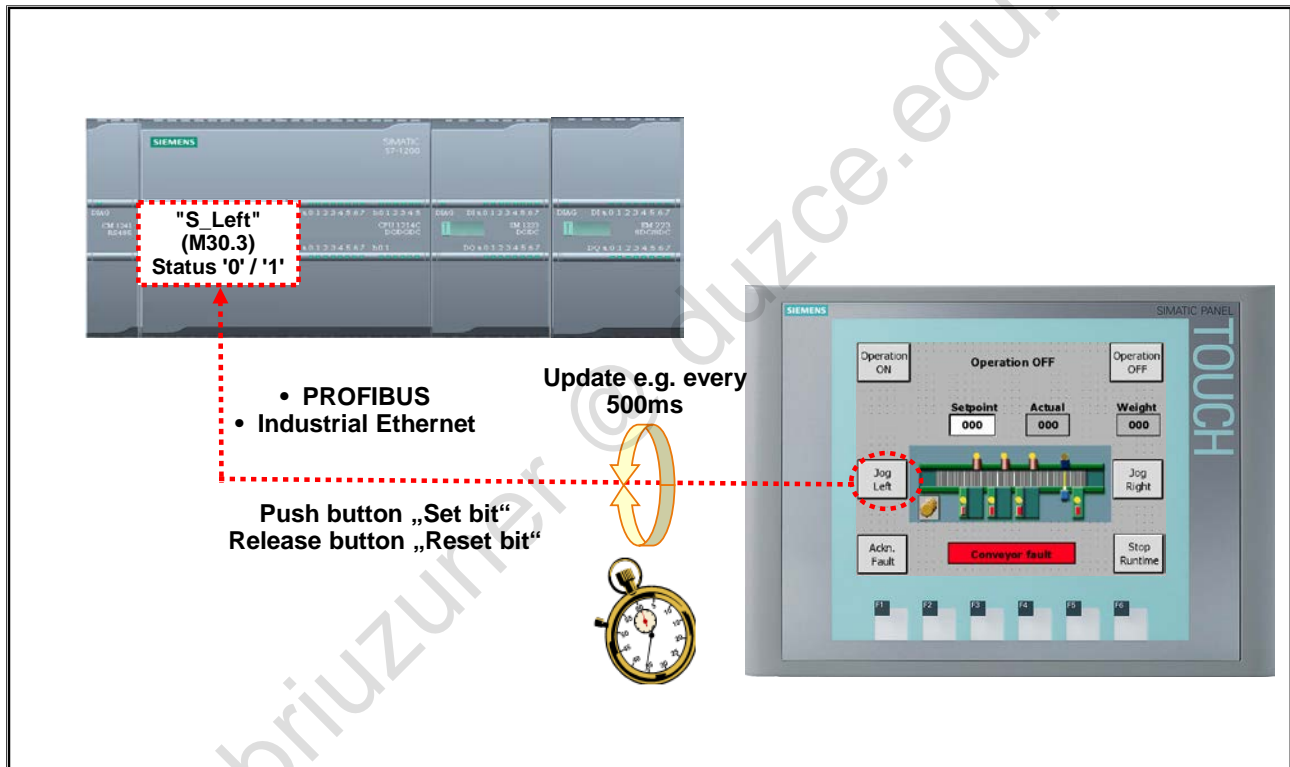
If the current cycle time exceeds the cycle monitoring time set in the CPU properties, the CPU starts error handling.

The error OB "Time error interrupt" OB80 is started in the CPU.

- S7-1500 and S7-1200 with firmware  $\geq V4.0$ :
  - If it exists, the CPU continues the cycle processing after the end of OB80
  - If it is not present, the CPU switches to STOP mode
- S7-1200 with firmware  $\leq V3.x$ :
  - If it exists, the CPU continues the cycle processing after the end of OB80
  - If it is not present, the CPU will remain in RUN mode

Violation of the maximum cycle time twice does not lead to the calling of an OB, but to the STOP of the CPU.

## 2.7. Data Exchange between Touchpanel and CPU



### Tags

Data is exchanged between SIMATIC S7 and the HMI system via process tags. For this, tags are created in the configuration of the WinCC system and are then assigned to a data area of the CPU. The HMI system reads out the value of the tags cyclically and displays it, for example, in an output field.

### Data Areas

For the configuration of the tags, the following global data areas of the CPU can be used:

- Data blocks (DB)
- Bit memories (M)
- Inputs (I) and outputs (Q)
- I/O (peripheral) inputs and I/O (peripheral) outputs

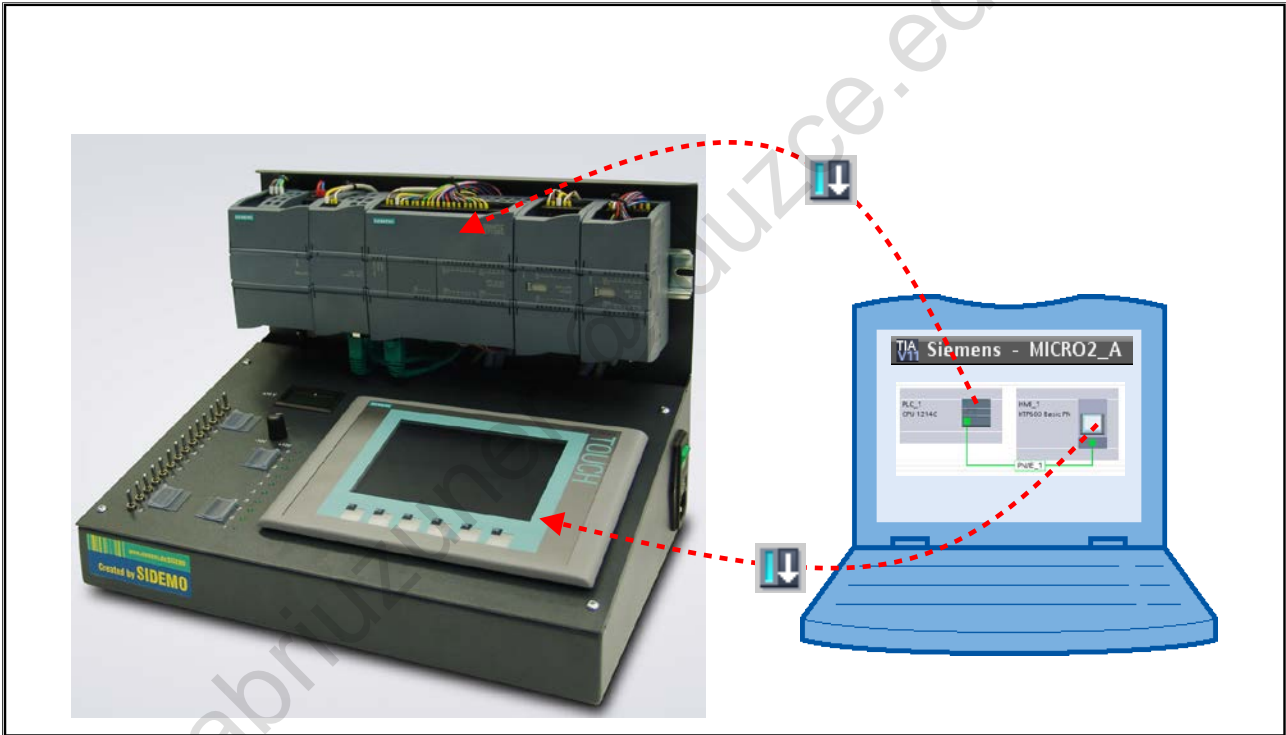
HMI systems also recognize local tags without process connection, these tags are exclusively processed internally and also do not reserve any communication resources whatsoever.

### Communication

The HMI devices can communicate with the controller via the bus systems MPI, PROFIBUS DP or Industrial Ethernet. The S7 protocol is used for this purpose. Communication is handled by the operating systems of the S7 CPU and the HMI system. No user programming on the S7 is required for this purpose.

An HMI device can exchange data with more than one controller at the same time.

## 2.8. Task Description: Commissioning the Training Case



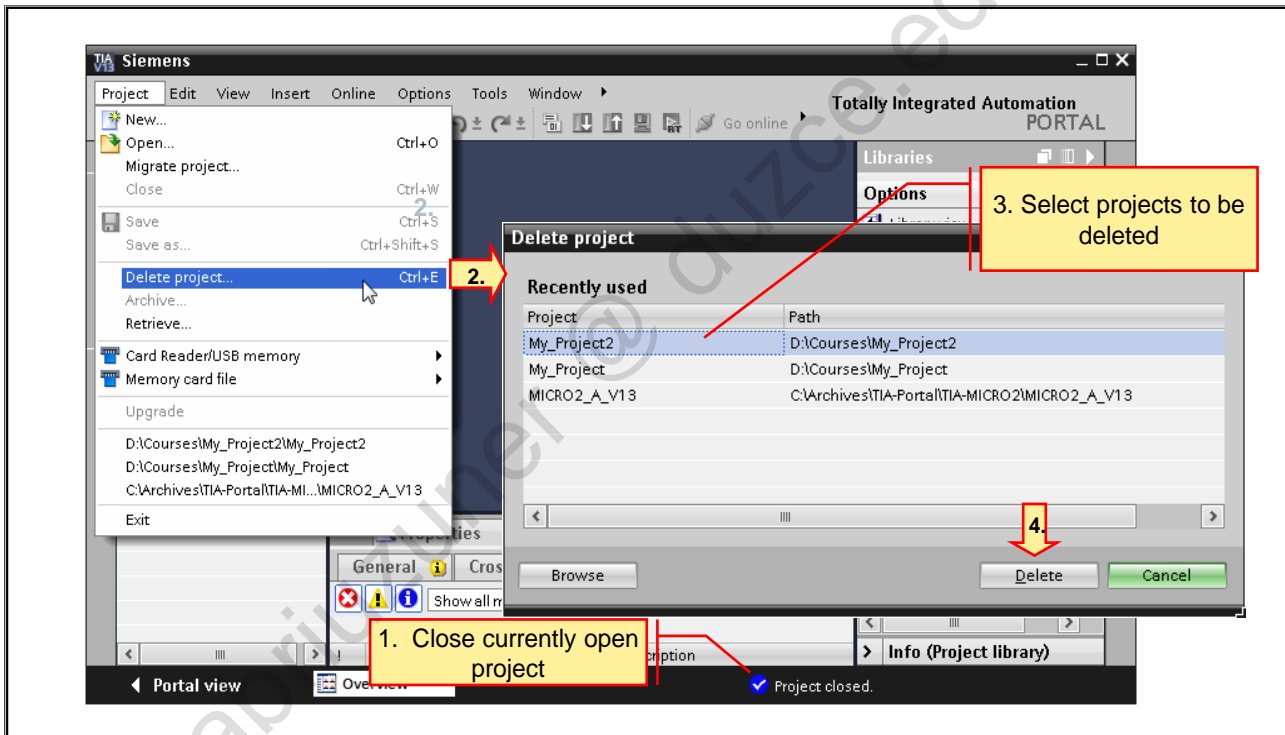
### Task Description

The S7-1214 training case with touchpanel is to be commissioned.

The basic project required for this is already located on your hard drive and must, if necessary, be adjusted to the existing hardware and loaded into the CPU or the touchpanel.



## 2.8.1. Exercise 1: Deleting Old Projects



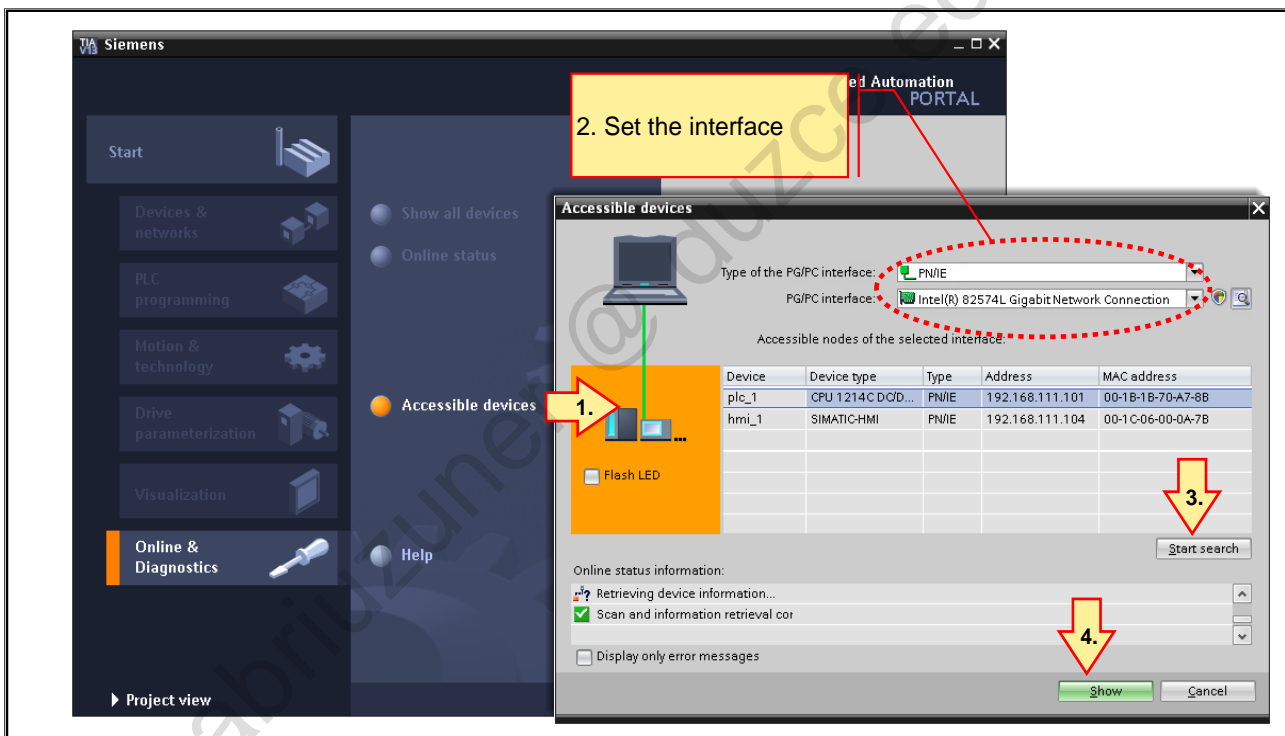
### Task

Delete all existing projects.

### What to Do

1. Start the TIA Portal
2. Switch to the Project view
3. Close any open projects
4. Let the system show you the existing projects and delete them as shown

## 2.8.2. Exercise 2: Establishing an Online Connection to the CPU




### Task


Establish a connection to the CPU 1214 and use the automatic IP address assignment of the TIA Portal for this.

### What to Do


1. Switch to the Portal view
2. Start the "Accessible devices" function "Online & Diagnostics" → "Accessible devices"
3. Select "PN/IE" for the "Type of the PG/PC interface"
4. Select the respective Ethernet interface of your field PG

 Field PG: This device has 2 Ethernet interfaces! Make sure that you use the correct interface!

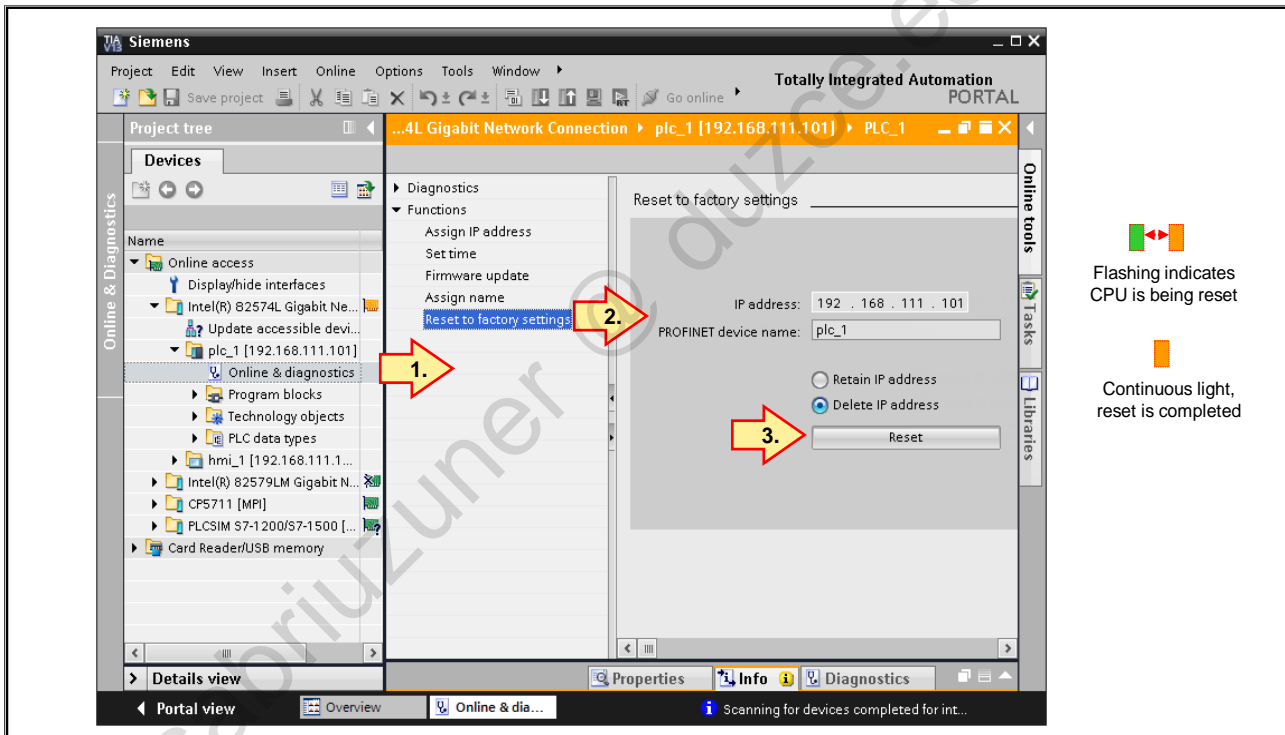
5. After searching, select the CPU 1214 → Device type "S7-1200"

 Should more than one S7-1200 be found, you can find out whether you have selected the correct CPU through the function "Flash LED". The Status LEDs of the respective CPU flash.

6. Click on "Show" and confirm the follow-up prompt with "Yes".

 The telegram service "TCP/IP" is used to execute certain functions. For this, the CPU and PG must be in the same IP subnet. The TIA Portal assigns your PG a temporary, alternative IP address which is in the same subnet as the CPU. That way you can work freely.

### 2.8.3. Exercise 3: Resetting the CPU to Factory Settings



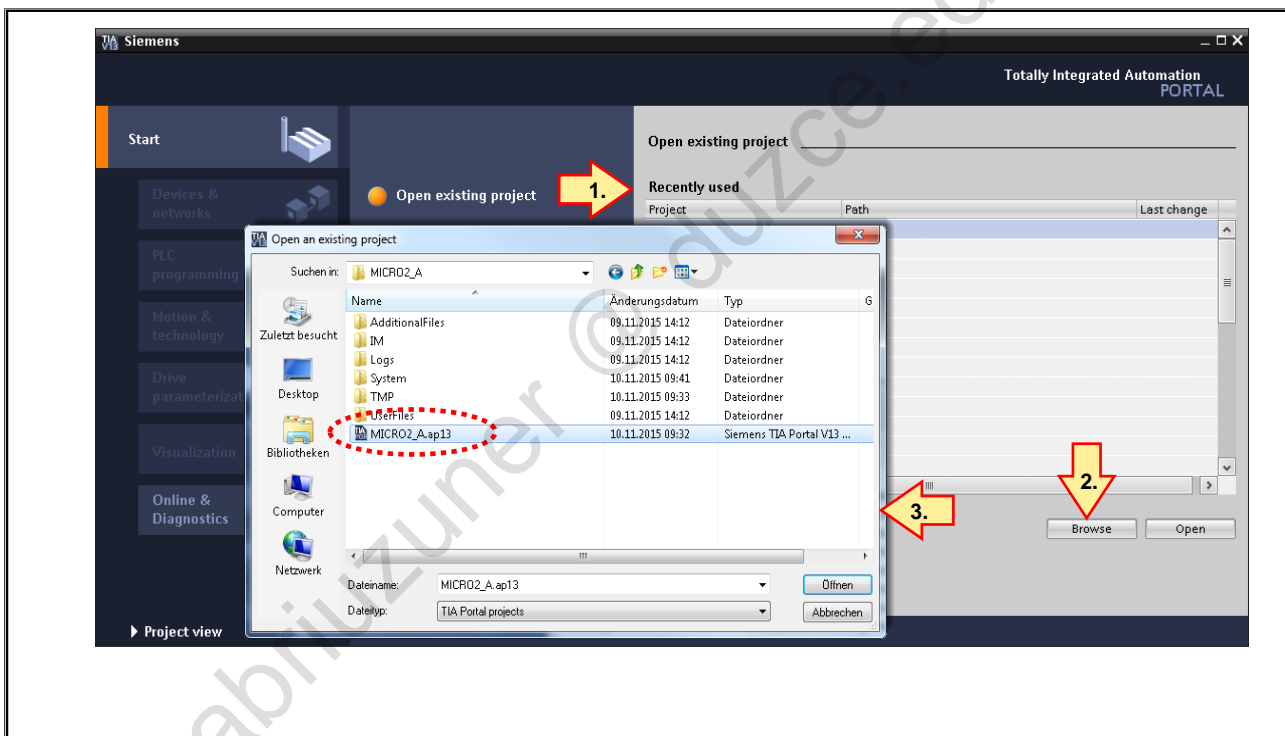
#### Task

Establish a defined initial state for the following configuration steps by resetting the CPU to factory settings.

#### What to Do

1. Navigate to your CPU and open "Online & diagnostics"
2. Open the dialog "Reset to factory settings"
3. Select the item "Delete IP address" and click on "Reset"

## 2.8.4. Exercise 4: Opening existing Project and save it with new name



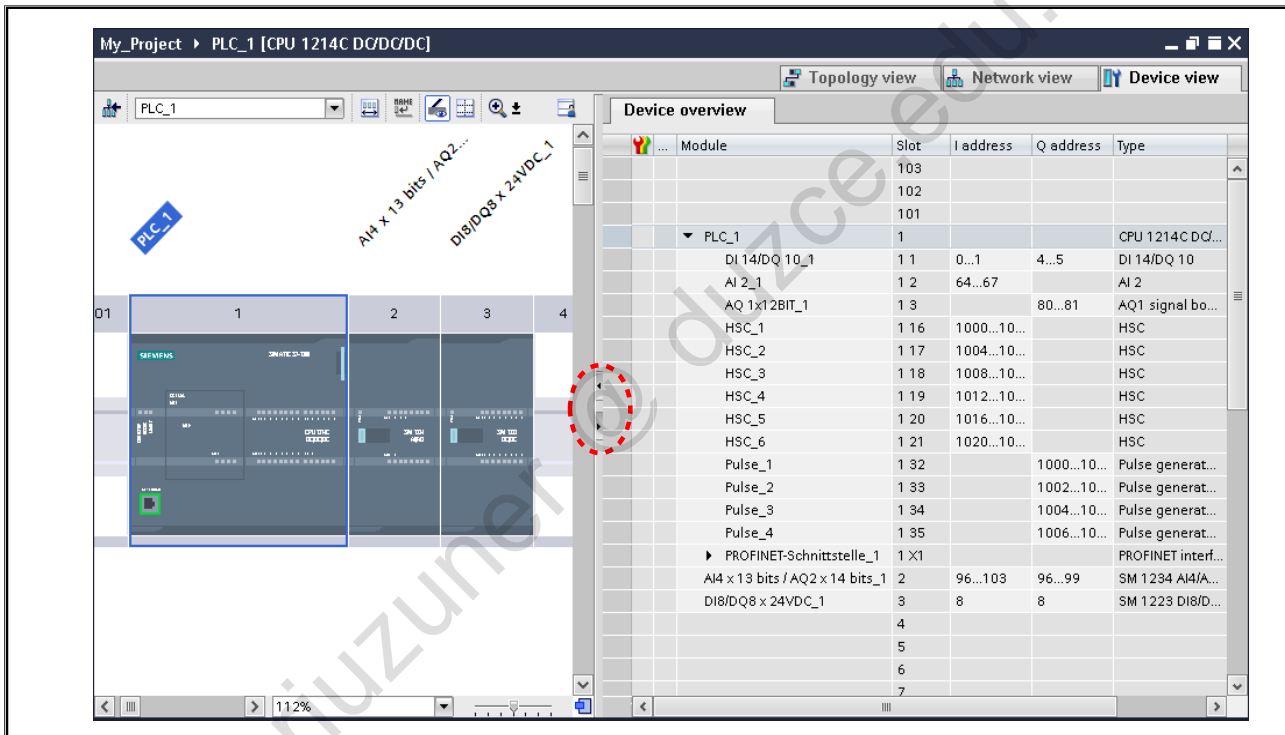
### Task

Open the basic project and save it using the name "My Project".

### What to Do

1. Switch to the Portal view and select the menu item "Open existing project"
2. Click on "Browse" and navigate to the basic project folder. Open the project.  
<Drive>:\\_Archive\TIA-MICRO2[version]\MICRO2\_A → Double-click on MICRO2\_A.ap16
3. Open the Project view
4. Save your project using the name "My\_Project"  
Project → Save as... → "My\_Project"

## 2.8.5. Exercise 5: Checking and, if necessary, adjusting the device configuration



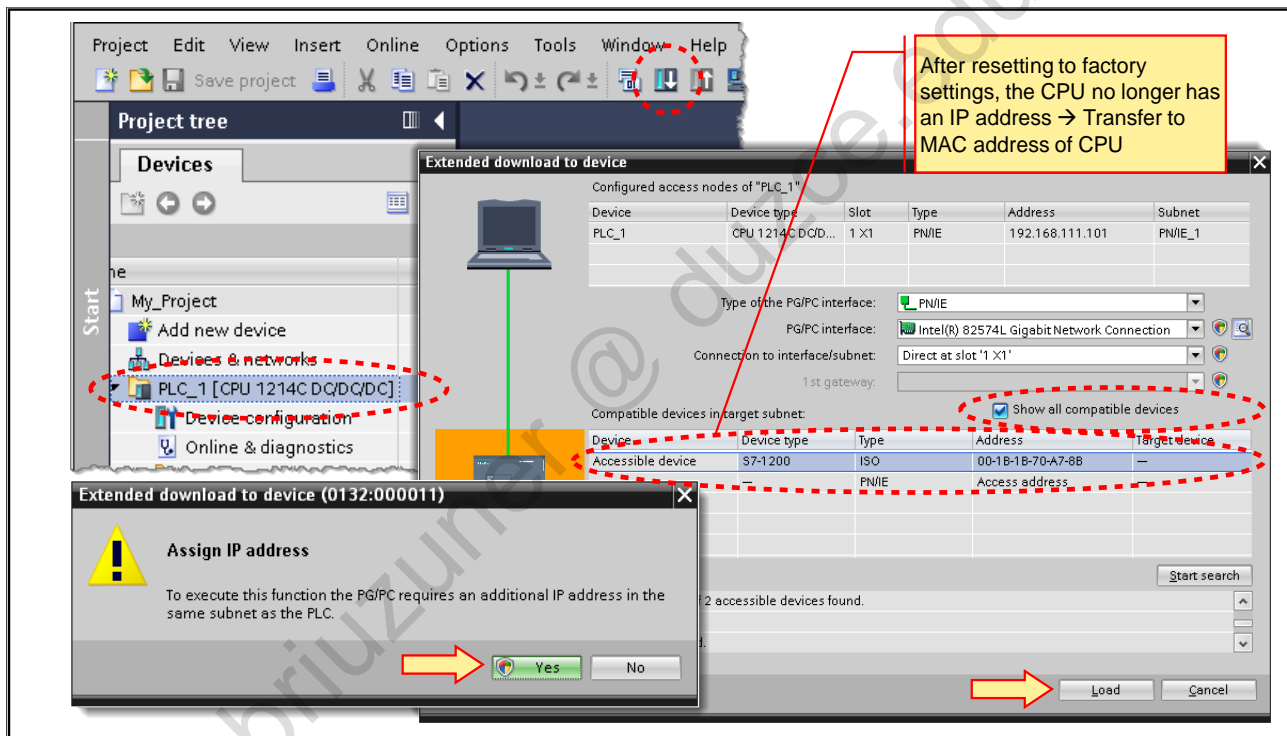
### Task

Check the already existing device configuration and the set I/O addresses.

### What to Do

If necessary, exchange deviating modules and check the I/O addresses of the CPU and SMs shown in the picture.

## 2.8.6. Exercise 6: Downloading the device configuration and user program into the CPU



### Task

Transfer the PLC\_1 (hardware and software) to the S7-1214

### What to Do

1. Open the context menu of the CPU and then the Download dialog  
Right-click on "PLC\_1" → "Download to device" → "Hardware and Software"
2. Make the same PG/PC interface settings as in Exercise 2

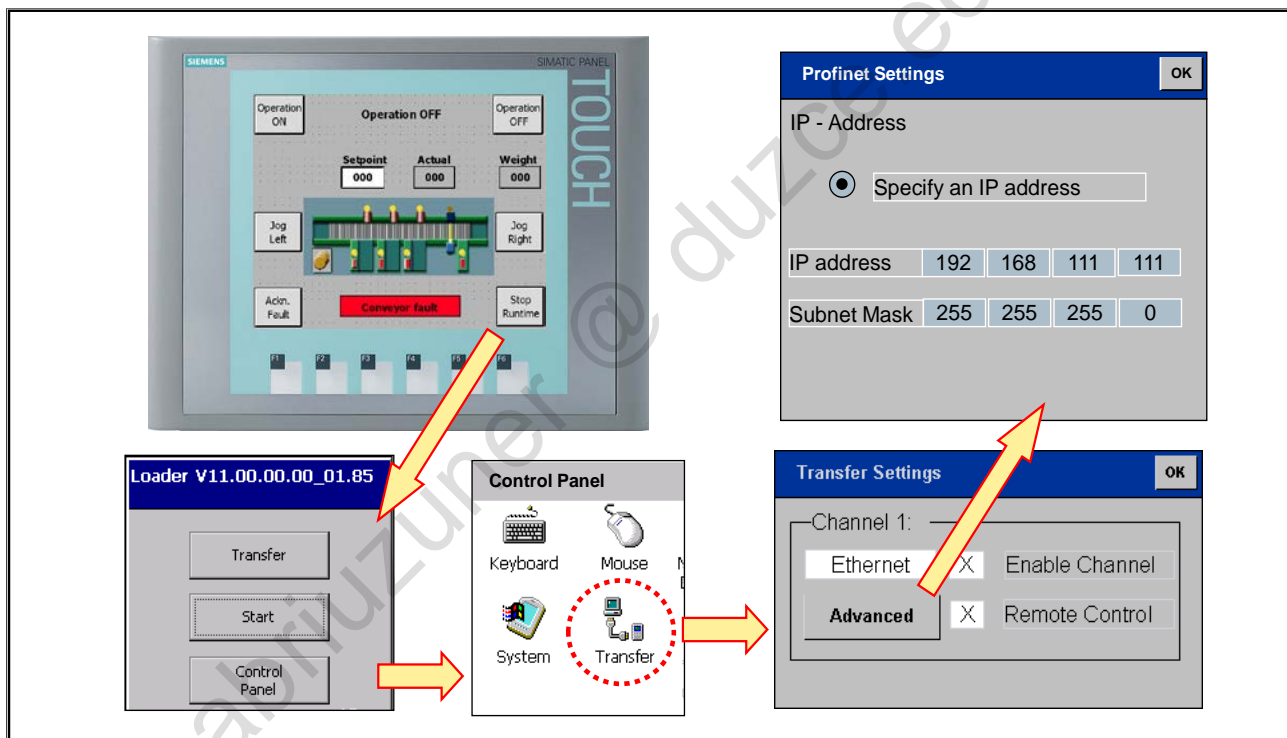


A first search starts automatically. The CPU with the configured IP address is searched for. Since there is currently no station with this address, no results are shown.

3. Check the option "Show all compatible devices"
4. Select the device "S7-1200 | ISO | MAC address". If this applies to more than one station, you can use the function "Flash LED" (see Exercise 2)
5. Confirm the follow-up prompt with "Yes"
6. In the following dialog, click on "Load"



## 2.8.7. Exercise 7: Setting the IP address of the touchpanel



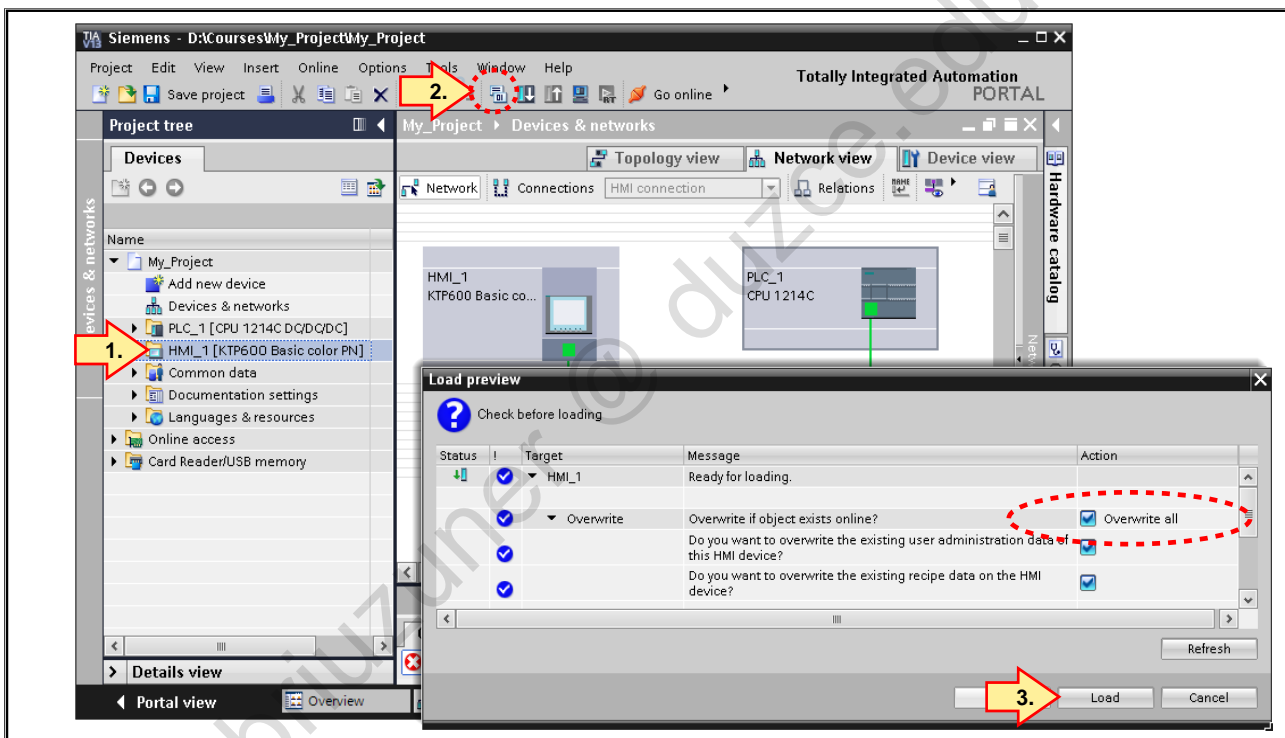
### Task

On the touchpanel, set the IP address stored in the project.

### What to Do

1. Stop the current Runtime
2. Follow the instructions shown in the picture

## 2.8.8. Exercise 8: Transferring the touchpanel project



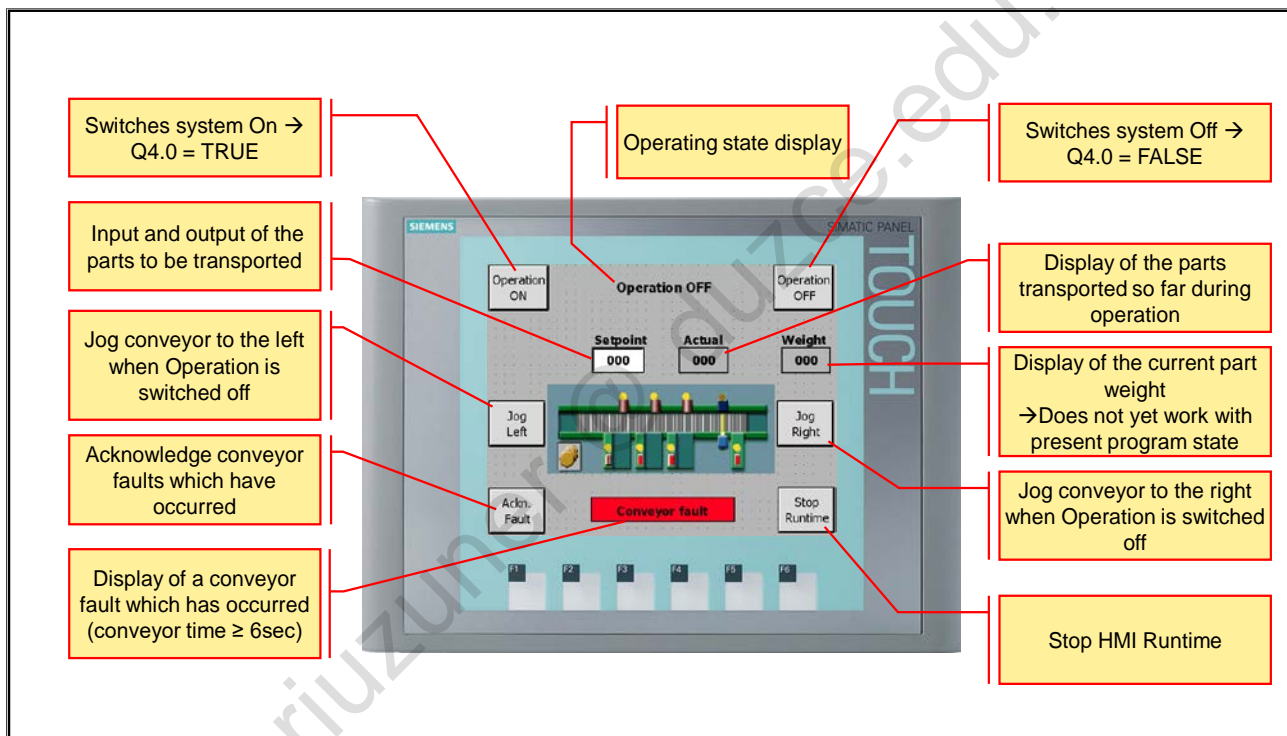
### Task

Download the existing HMI project into the touchpanel.

### What to Do

1. Select the touchpanel in the Project tree
2. Click on the "Download to device" button as shown in the picture
3. Check the option "Overwrite all" and then click on "Load"

## 2.8.9. Exercise 9: Function test touchpanel project and CPU program



### What to Do

Check the system functions described in the following.

### Operation

The system can be switched on and off via the buttons "Operation ON" and "Operation OFF".

### Operation OFF

The conveyor can be moved in the appropriate direction via the buttons "Jog Right" and "Jog Left".

### Operation ON

A value  $> 0$  must be entered via the input field "Setpoint (quantity)".

When the conveyor is standing, the indicator lights show with a continuous light at Bays 1 and 2 that a new part can be placed on the conveyor. If this has happened, the indicator light at the particular bay shows with a 1Hz flashing light that the transport sequence can be started by pressing the bay pushbutton. The part is transported until it has passed through the light barrier. During transport, the bay indicator lights show a 2Hz flashing light.

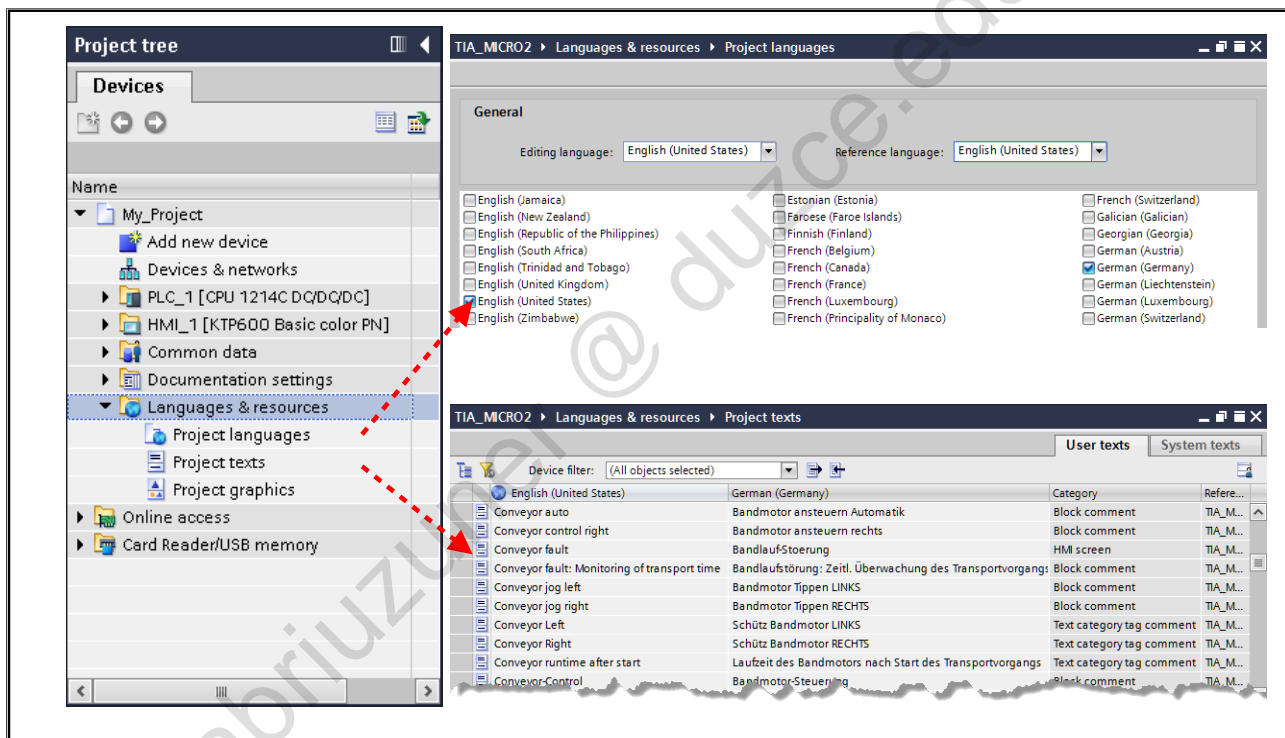
If a transport sequence takes longer than 6 seconds, the conveyor is automatically stopped and the fault is indicated with a flashing light on the touchpanel and the simulator LED "P\_Fault" (Q5.0). Only after the fault has been acknowledged via the appropriate touchpanel button or via the simulator pushbutton "S\_Acknowledge" (I 1.0), a new transport sequence can be started.

The transported parts are counted as they pass through the light barrier and the number is displayed on the output field "Actual". If the number, which has been input via the input field "Setpoint", is reached, it is indicated via the conveyor indicator light "P\_BayLB" (Q 8.4). Only after this is acknowledged via the conveyor pushbutton "S\_BayLB" (I 8.4), can a new transport sequence be started.



The output field "Weight" does not yet display any values with the present program state. The associated programming is done in the chapter "Analog Value Processing".

## 2.8.10. Exercise 10: Selecting the Editing Language



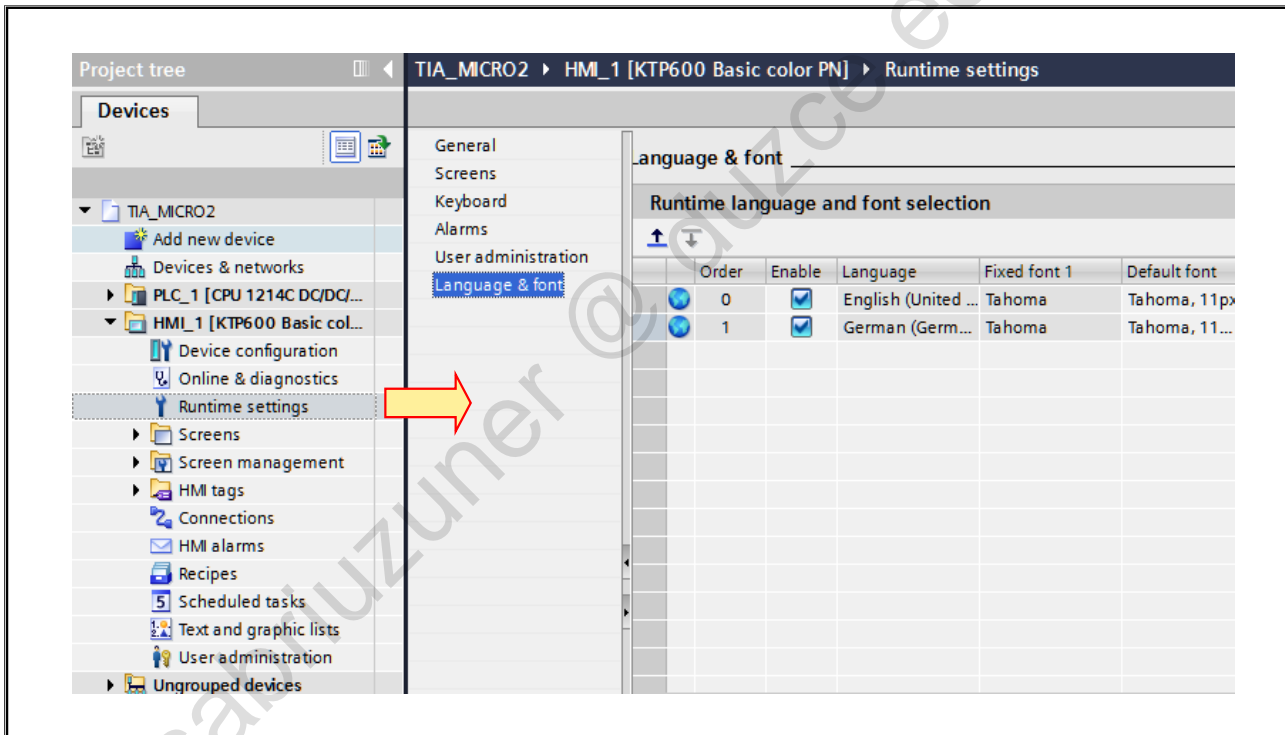
### Task

Activate the project languages as shown in the picture to enable a language switch in the panel.

### What to do

1. Open the folder „Languages & resources“ in the project tree and select the point “Project languages”
2. Activate the languages “English (United States” and “German (Germany)”

## 2.8.11. Exercise 11: Runtime Settings



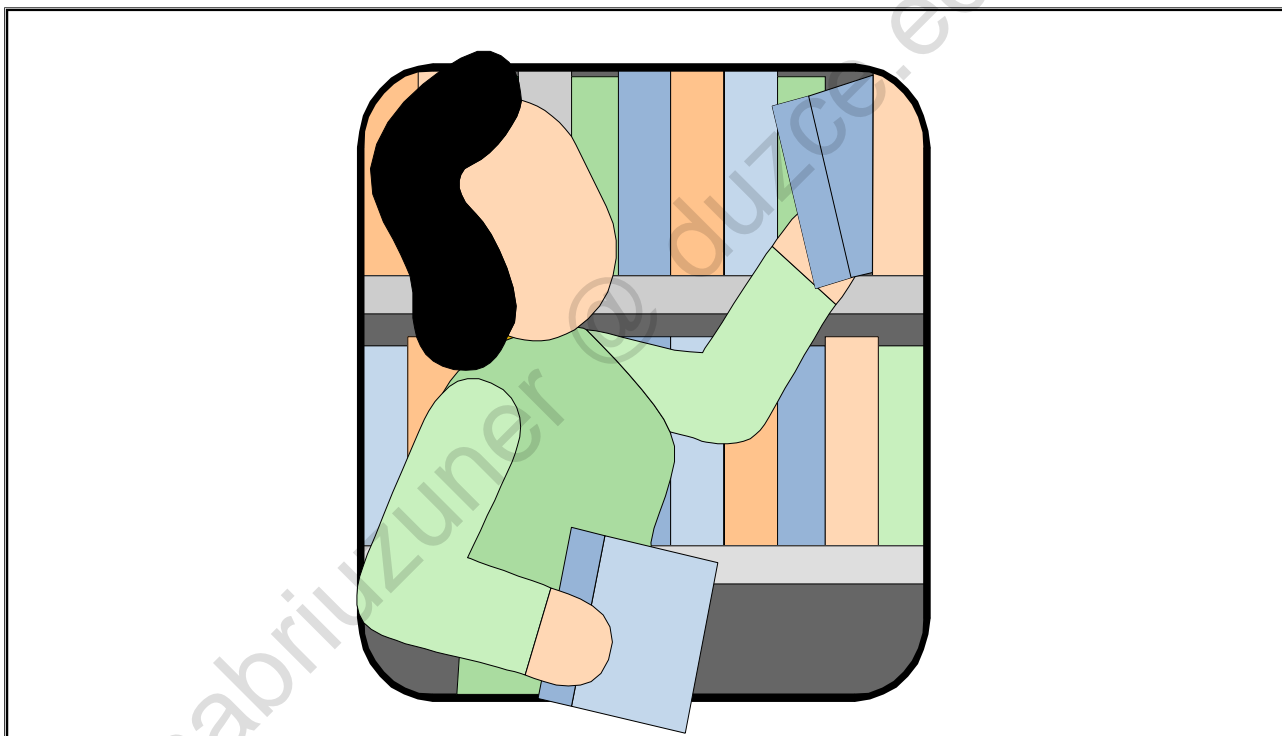
### Task

The two shown languages should be available in the touch panel

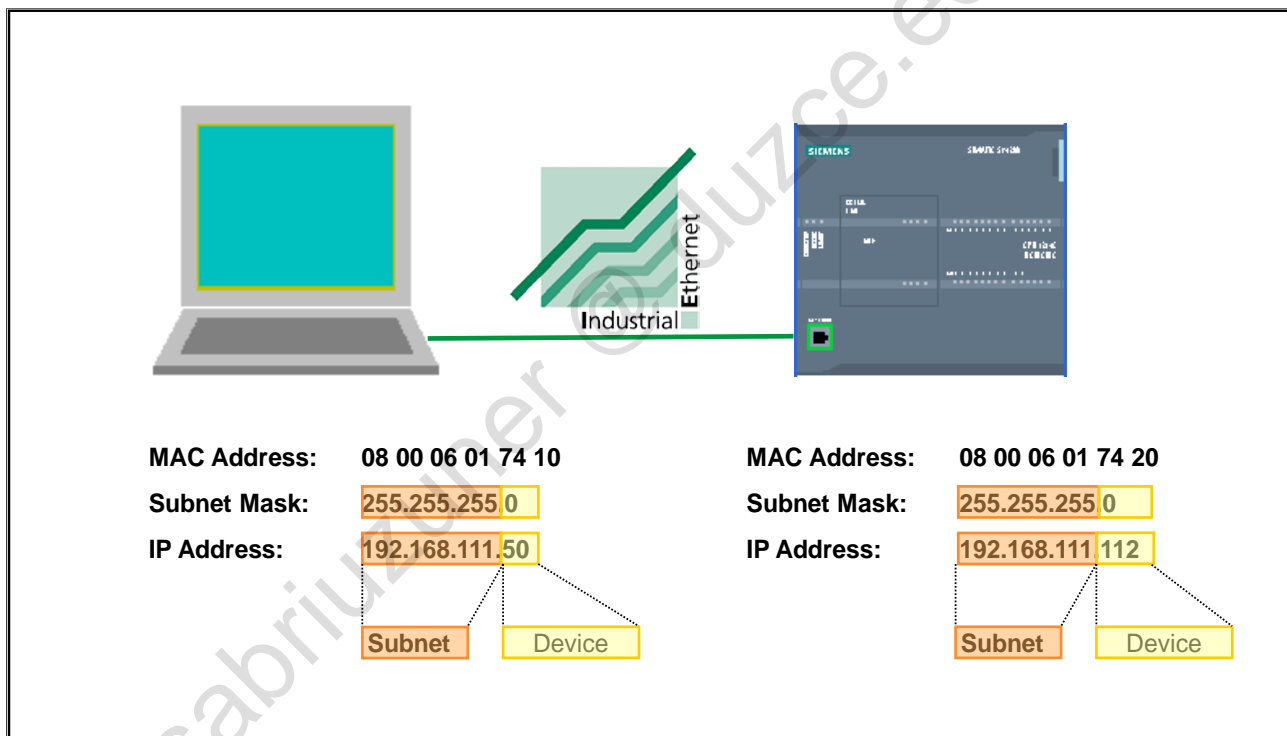
### What to do

1. Open the runtime settings and select "Languages & font"
2. Activate the both shown languages. Check the order of the languages

## 2.9. Additional Information



## 2.9.1. Industrial Ethernet: IP Address and Subnet Mask



### Internet Protocol

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

### IP Addresses

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

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

### MAC Address

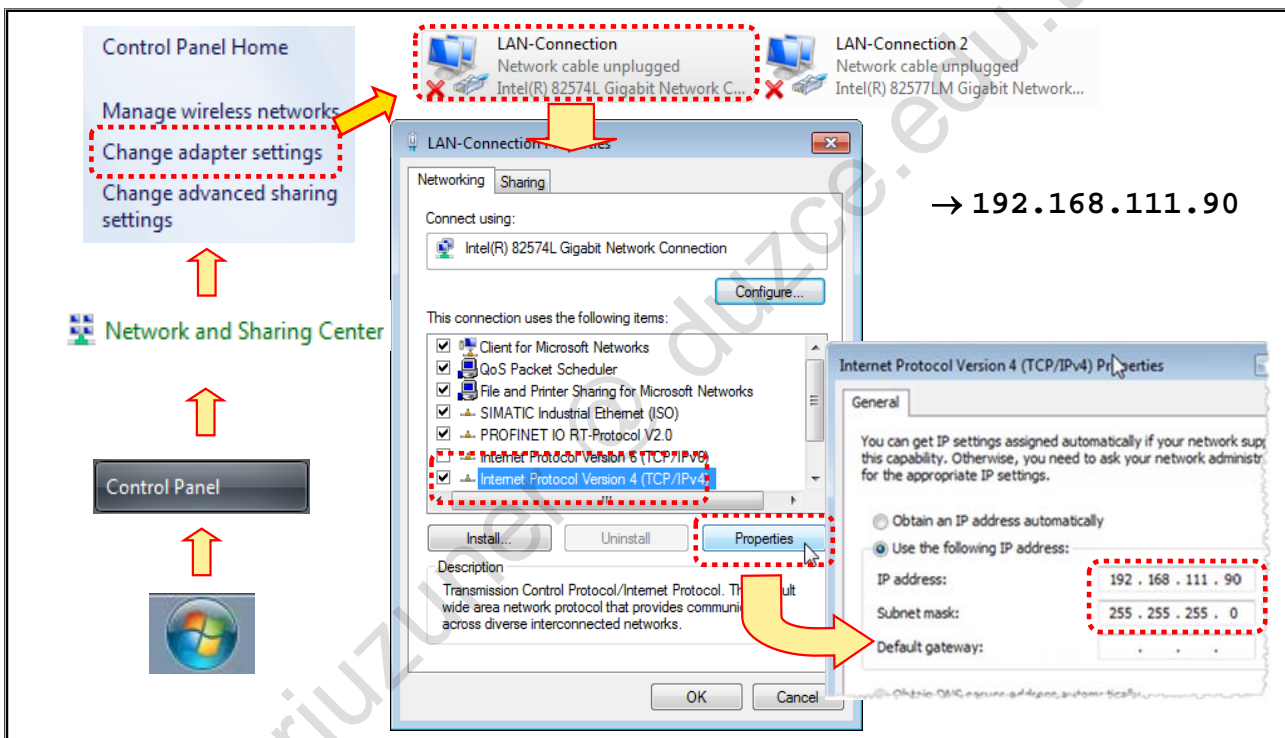
Every Ethernet interface is assigned a fixed address by the manufacturer that is unique worldwide. This address is referred to as the hardware or MAC address (Media Access Control). It is stored on the network card and uniquely identifies the Ethernet interface in a local network. Cooperation among the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

### Subnet Mask

The subnet mask separates the IP address into network and device (computer) address. Only IP addresses which network part is identically can be reached.



## 2.9.2. Online Access: Assigning an IP Address for the PG

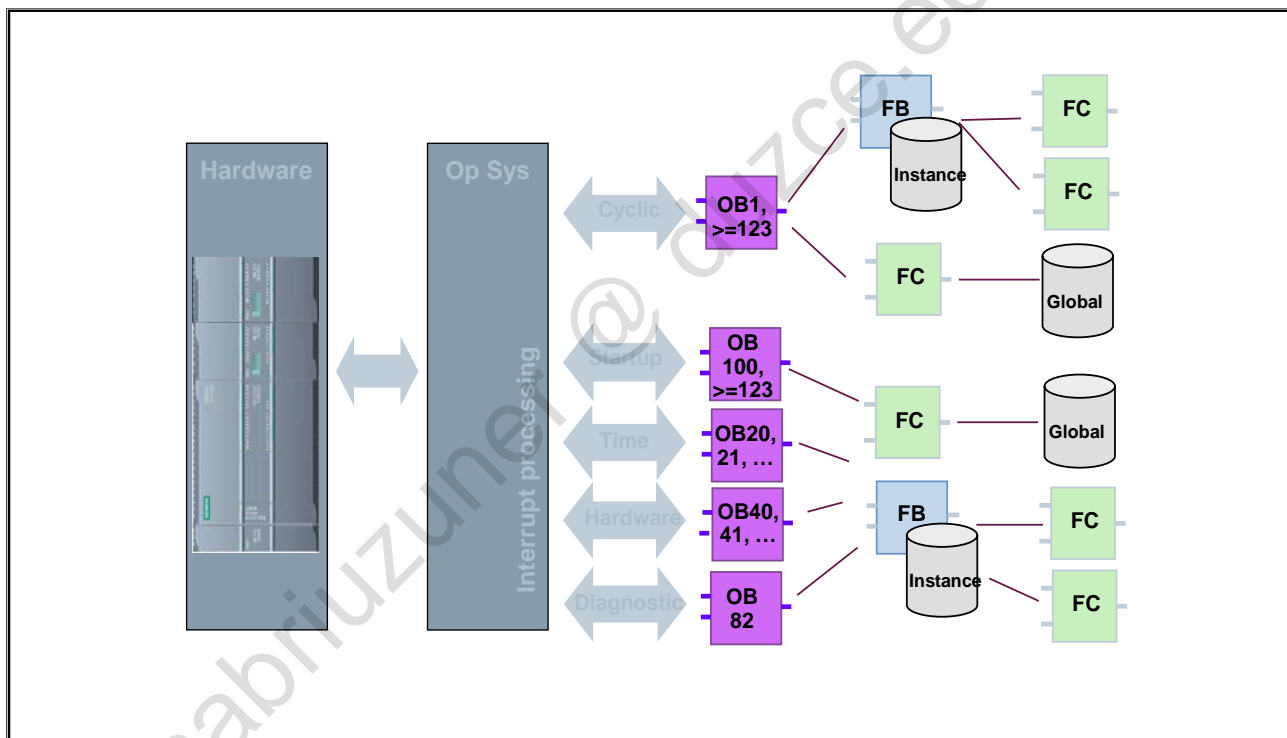


### IP Address of the Programming Device

You can set the IP address of the PG as shown in the picture.

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

### 2.9.3. OB – Organization Blocks



#### OBs

Organization blocks form the interface between the user program and the CPU's operating system.

Organization blocks are called exclusively by the operating system. There are various start events (time interrupt, hardware interrupt, ...see picture).

#### Startup Program

After a restart, a startup program is executed. In the startup OBs you can, for example, carry out a pre-assignment of communication connections.

#### Cyclic program execution

The program stored in the cyclic OB is executed cyclically, after it is executed completely it is executed again. With this cyclic program execution, the reaction time results from the execution time for the CPU's operating system and the sum of the command runtimes of all executed instructions. The reaction time, that is, how fast an output can be switched in relation to an input signal, amounts to a minimum of one time and a maximum of two times the cycle time.

#### Periodic Program Execution

This makes it possible to interrupt the cyclic program execution at fixed intervals. With the cyclic interrupts, an organization block (for example OB30) is executed after an adjustable time base (for example, every 100ms) has expired. In these blocks, closed-loop control blocks with their sampling time are called.

#### Event-driven program execution

In order to be able to react quickly to a process event, the hardware interrupt can be used. After an event occurs, the cycle is immediately interrupted and an interrupt program is executed.

With time-delay interrupts, a freely definable event can be reacted to with a time-delay; with an error OB, the user can influence the behavior of the controller in case there is an error.

## 2.9.4. Events which Start an OB

Event Class	OB No.	Number	Start Event	Priority
Cyclic Program	1, >= 123	>= 1	End of startup or End of last cycle OB	1
Startup	100, >= 123	>=0	STOP-RUN transition	1
Time-of-day interrupt	>= 10	Max. 2	Start time has been reached	2
Time-delay interrupt	>= 20	Max. 4	Time-delay expired	3
Cyclic interrupt	>= 30		Constant bus cycle time expired	8
Hardware interrupt	>= 40	Max. 50 (more can be used with DETACH and ATTACH)	•Positive (rising) edge (max. 16)	18
			•Negative (falling) edge (max. 16)	
			•HSC: Count value= Reference value (max. 6) •HSC: Count direction changed (max. 6) •HSC: External reset (max. 6)	18
Status interrupt	55	0 or 1	CPU has received status interrupt	4
Update interrupt	56	0 or 1	CPU has received update interrupt	4
Manufacturer or profile-specific interrupt	57	0 or 1	CPU has received manufacturer interrupt or profile-specific interrupt	4
Diagnostic interrupt	82	0 or 1	Module has detected an error	5
Pull/Plug interrupt	83	0 or 1	Removal / Insertion of modules of distributed I/O	6
Rack error	86	0 or 1	Error in the input/output system of the distributed I/O	6
Time error	80	0 or 1	Cycle monitoring time exceeded, Called OB is still being executed, Time-of-day interrupt missed, Time-of-day interrupt missed during STOP, Queue overflowed, Interrupt loss due to high interrupt load	22

### Events

The operating system of S7-1200-CPU is based on events. There are two types of events:

- Events which can start an OB
- Events which cannot start an OB

An event which can start an OB triggers the following reaction:

- If you have assigned an OB to the event, this OB is called  
If it is currently not possible to call this OB, the event is entered into a queue according to its priority.
- If you have not assigned an OB to the event, the predefined default system reaction is carried out.

An event which cannot start an OB triggers the predefined default system reaction for the associated event class.

The user program cycle is therefore based on events, the assignment of OBs to those events, and on the code which is either contained in the OB or called in the OB.

The table above gives an overview of the events which can start an OB. It is sorted according to OB priority. 1 is the lowest priority.

### OB Priority

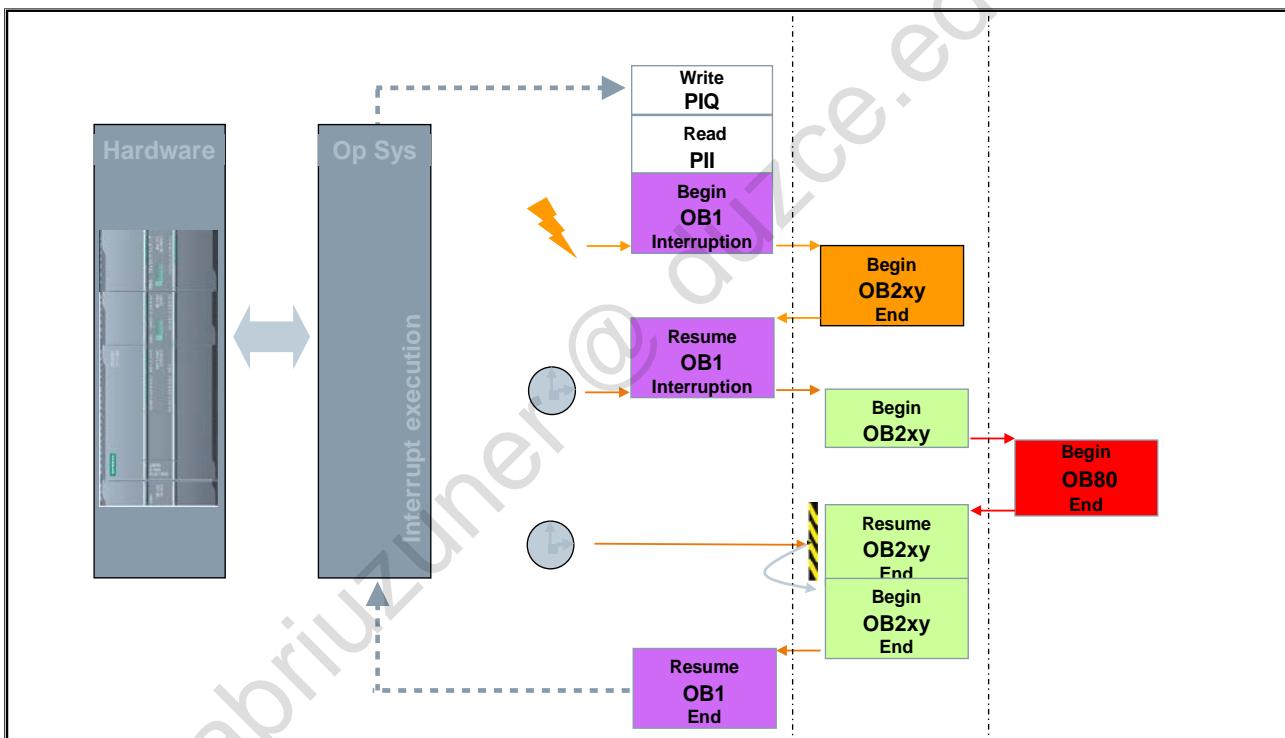
With the exception of the startup and cyclic OBs, all OBs have a priority which can be changed between 2 and 24. In all, the priorities are staggered from 1 - 27, whereby 1 is the lowest priority and 27 is the highest priority.

### 2.9.5. Events which Cannot Start an OB

Event Class	Event	Event Priority	System Reaction
Insert / Remove central modules	Insert / Remove a module	21	STOP
I/O access error during process image update	I/O access error during process image update	22	Ignore
Programming error	Programming error in a block for which you use the system reactions provided by the operating system (Note: If you have activated the local error handling, the error handling routine programmed in the block takes effect.)	23	RUN
I/O access error	I/O access error in a block for which you use the system reactions provided by the operating system (Note: If you have activated the local error handling, the error handling routine programmed in the block takes effect.)	24	RUN
Maximum cycle time exceeded twice	Cycle monitoring time was exceeded twice	27	STOP

Compare to previous chapter.

## 2.9.6. Interrupting the Cyclic Program



### Interruption of OBs

Every OB program execution can be interrupted between instructions by an event (OB) with a higher priority if this is set in the properties of the CPU. (CPU > Properties > Startup > OBs should be interruptible).

### Queue

If the OBs (with the exception of cyclic OBs) are not parameterized as interruptible or have the same or a lower priority, then this event is entered into a queue according to its priority. The start events of a queue are processed at a later point in time in the order they occurred.

### Interruption of the Cycle Program

Cyclic OBs have the lowest priority and are therefore interrupted when there are call requests from all other OBs, even if the OBs are not parameterized as interruptible in the CPU properties.

## 2.9.7. DB – Data Block

Name	Data type	Start value
▼ Static		
■ Var_1	Bool	false
■ Var_2	Int	0
■ Var_3	Bool	false
■ Var_4	Real	0.0
■ ▶ Measuring_point	array[1..5] of Real	
▼ Motor	Struct	
■ speed	Int	0
■ rated_current	Real	0.0
■ started_current	Real	0.0
■ direction	Bool	false
■ Var_5	Byte	0
■ Var_6	Bool	false
■ ▶ Timer_1	IEC_TIMER	false

### Overview

Data blocks are used to store user data. Data blocks occupy memory space in the user memory of the CPU. Variable data (e.g. numeric values) with which the user program works is in the data blocks.

The user program can access the data of a data block via bit, byte, word or double-word operations. Access can occur symbolically or absolutely.

### Applications

Data blocks can be used in different ways by the user depending on their contents. Differentiation is made between:

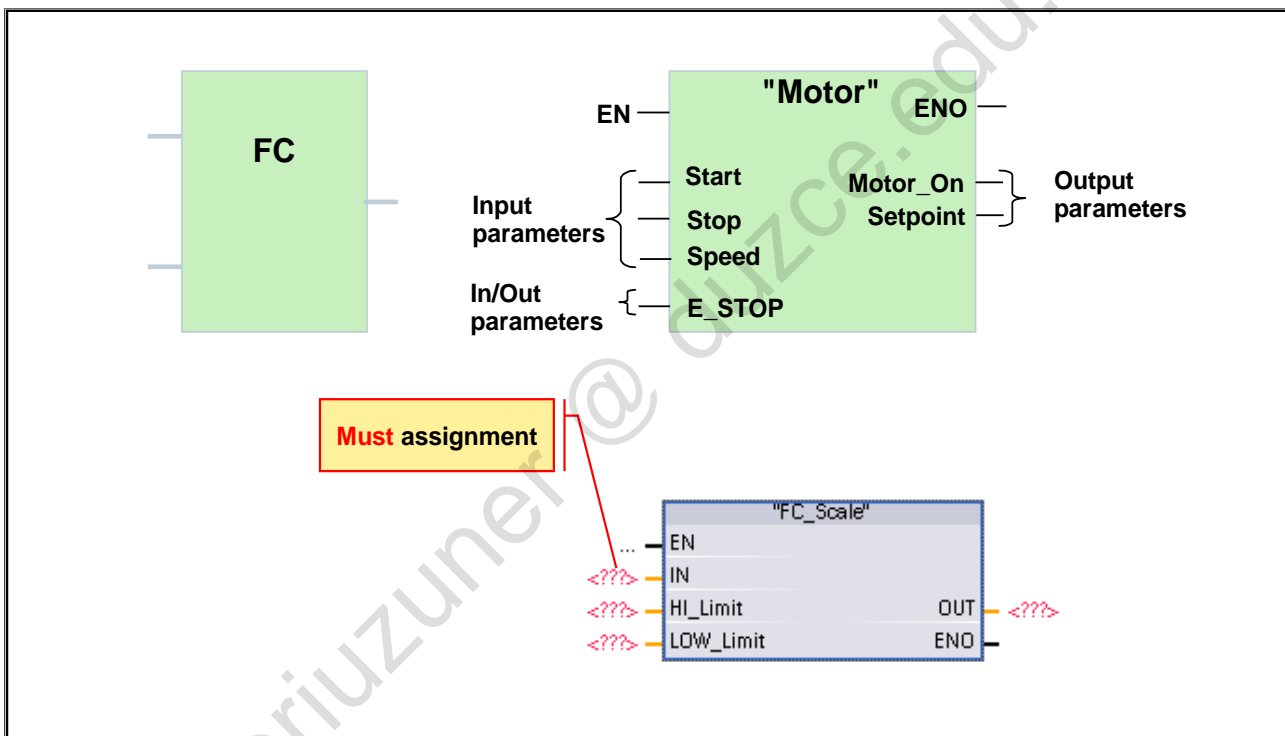
- Global data blocks: they contain information which can be accessed by all logic (code) blocks in the user program
- Instance data blocks: they are always assigned to an FB. The data of this DB should only be processed by the associated FB

### Creating DBs

Global DBs are created either via the Program editor or according to a previously created "PLC data type".

Instance data blocks are generated when a function block is called.

## 2.9.8. FC – Function



### Overview

Functions represent parameter-assignable blocks without memory. In STEP 7 they can have as many input parameters, output parameters and in/out parameters as are required.

Functions have no memory; no separate, permanent data area for storing results exists. Temporary results that occur during function execution can only be stored in the temporary variables of the respective local data stack.


### Application

Functions are primarily used when function values are to be returned to the calling blocks (for example, mathematical functions, single control with binary logic operation).

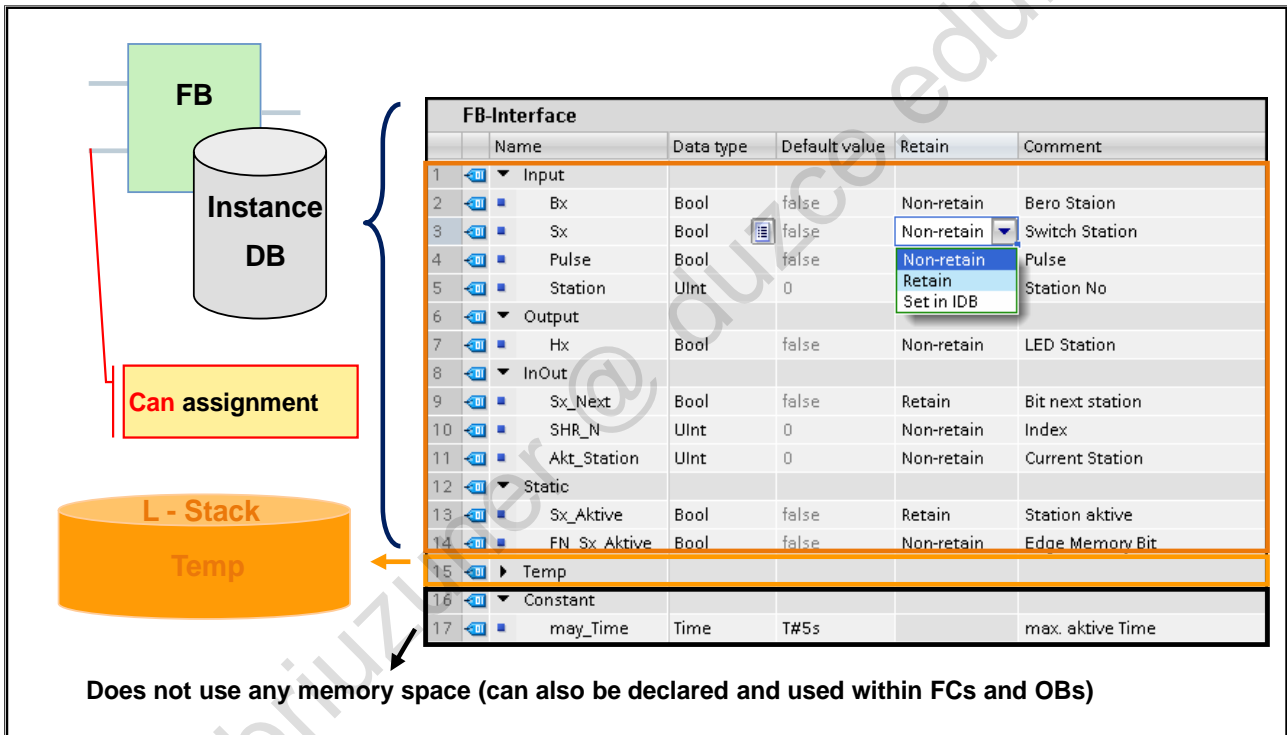
### IEC-61131-Conforming Functions

- Functions can have as many input parameters as is required. They can, however, only return one result to the output parameter RET\_VAL
- Global operands can neither be read nor written within functions
- No instances of function blocks can be called within functions
- Because of the missing "memory", the returned result of a norm-conforming function is solely dependent on the values of the input parameter. For identical values of the input parameter, a function always returns the identical result

It is therefore up to the programming person to create norm-conforming functions or to individually do the block programming and block structuring in STEP 7.

 If the last three points are fulfilled, then this is recognized in the Properties under the attribute "Block can be used as know-how protected library element" after compilation and the block can be used in every other project.

### 2.9.9. FB – Function Block



#### Overview

Function blocks (FB) are blocks of the user program and represent logic blocks with memory according to the IEC Standard 61131-3. They can be called by all blocks.

Function blocks can each have as many input, output and in/out parameters as well as static and temporary variables as are required.

Unlike FCs, FBs are instantiated, i.e. an FB is assigned its own data area in which the FB can "remember" process states from call to call, for example. In the simplest form, this private data area is its own DB (Instance DB).

#### "Memory"

You can declare static variables in the declaration section of a function block. The function block can remember information from call to call in these variables.

The ability of a function block to remember information over several calls is the essential difference to functions.

#### Application

With the help of this "memory", a function block can implement counter and timer functions or control process units, such as processing stations, drives, boilers etc., for example.

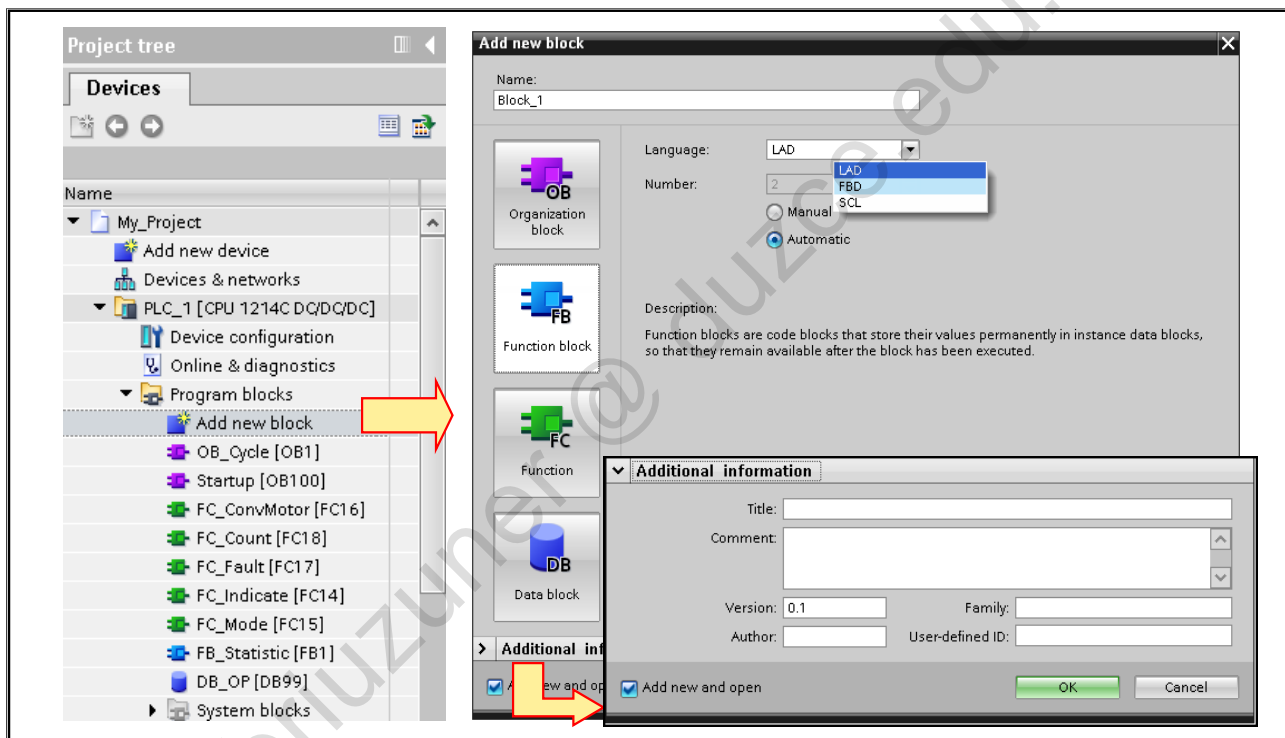
Function blocks are well suited for controlling all those process units whose performance depends not only on outside influences but also on internal states, such as processing step, speed, temperature etc.

When controlling such units, the internal status data of the process unit are then copied to the static variables of the function block.

👉 If no global variables are used and only multiple instances are used for FB calls, then this is recognized in the properties under the attribute "Block can be used as know-how protected library element" after compilation and the function block can be used in every other project.



## 2.9.10. Adding a New Block

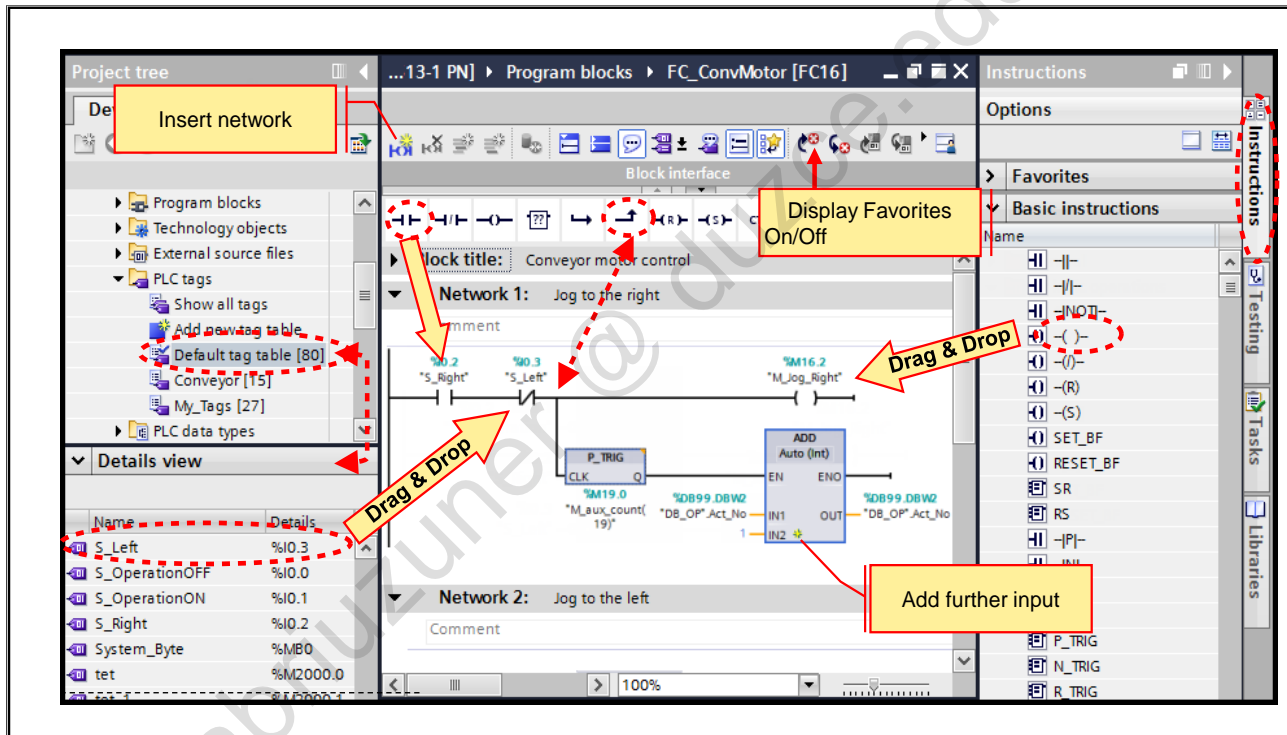


### Inserting a Block

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 be assigned automatically or manually.

In "Additional information", the block can be documented in more detail, among other things, with a version number and an author.

## 2.9.11. Block programming



### Block programming

The instructions within a block can be programmed as follows:

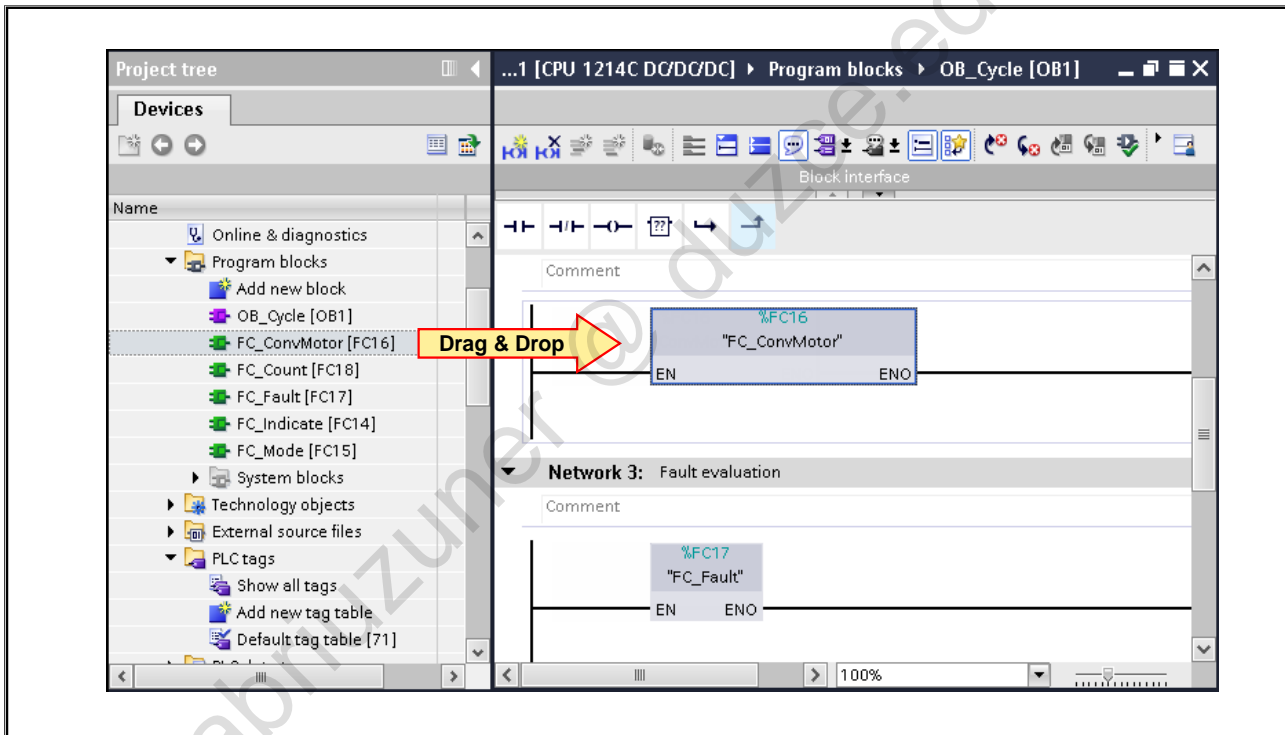
- using drag & drop from the favorites or the instructions catalog to anywhere in the program
- by first selecting the location in the program and then double-clicking on the desired instruction in the favorites or the instructions catalog

Operands can be entered with an absolute or a symbolic address. If a tag table or a data block 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.

## 2.9.12. Block Calls

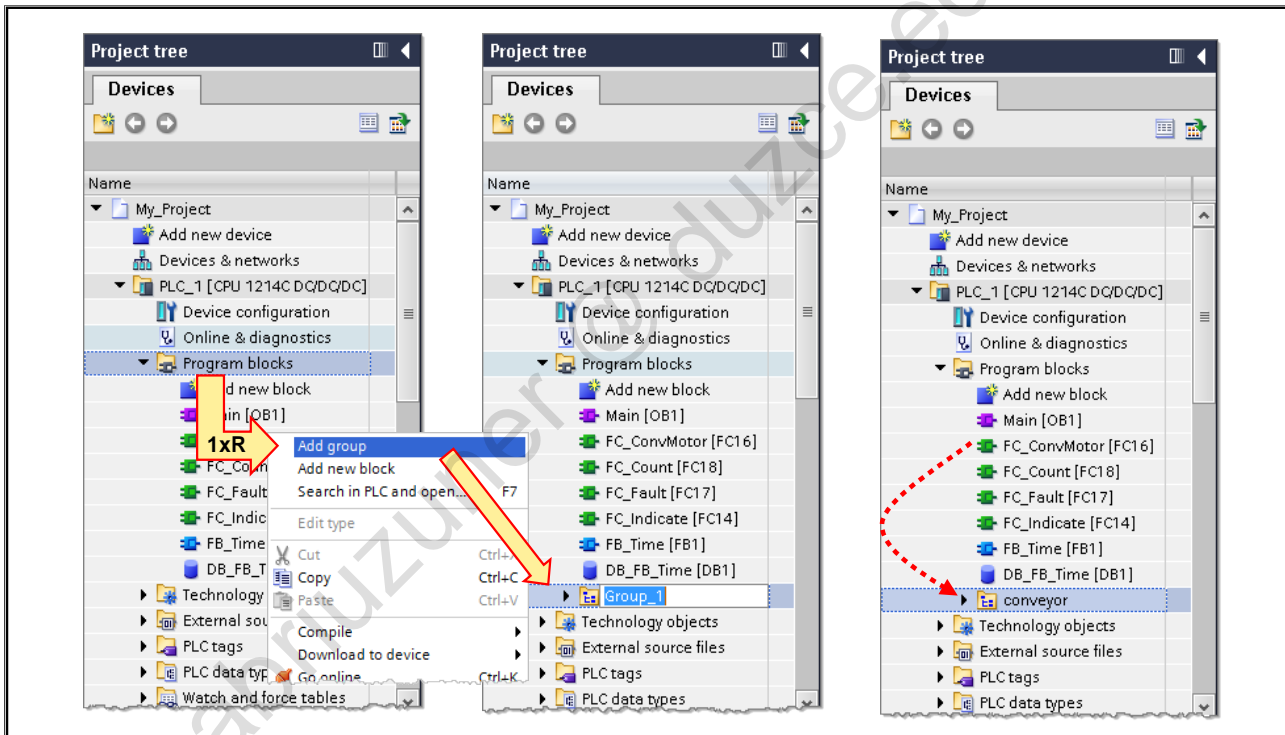


### Block Calls

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 continues with the instruction that follows the block call.

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

### 2.9.13. Block Groups

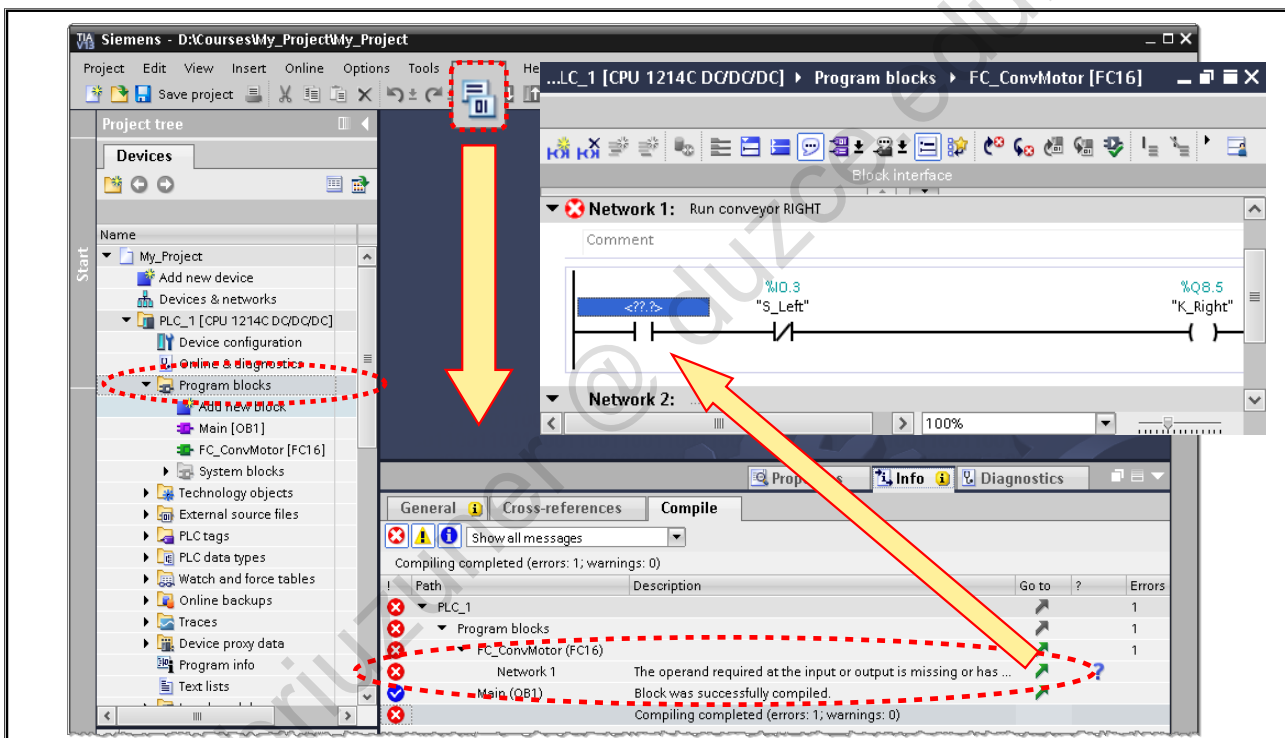


#### Block Groups

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.

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

## 2.9.14. Compiling a Block



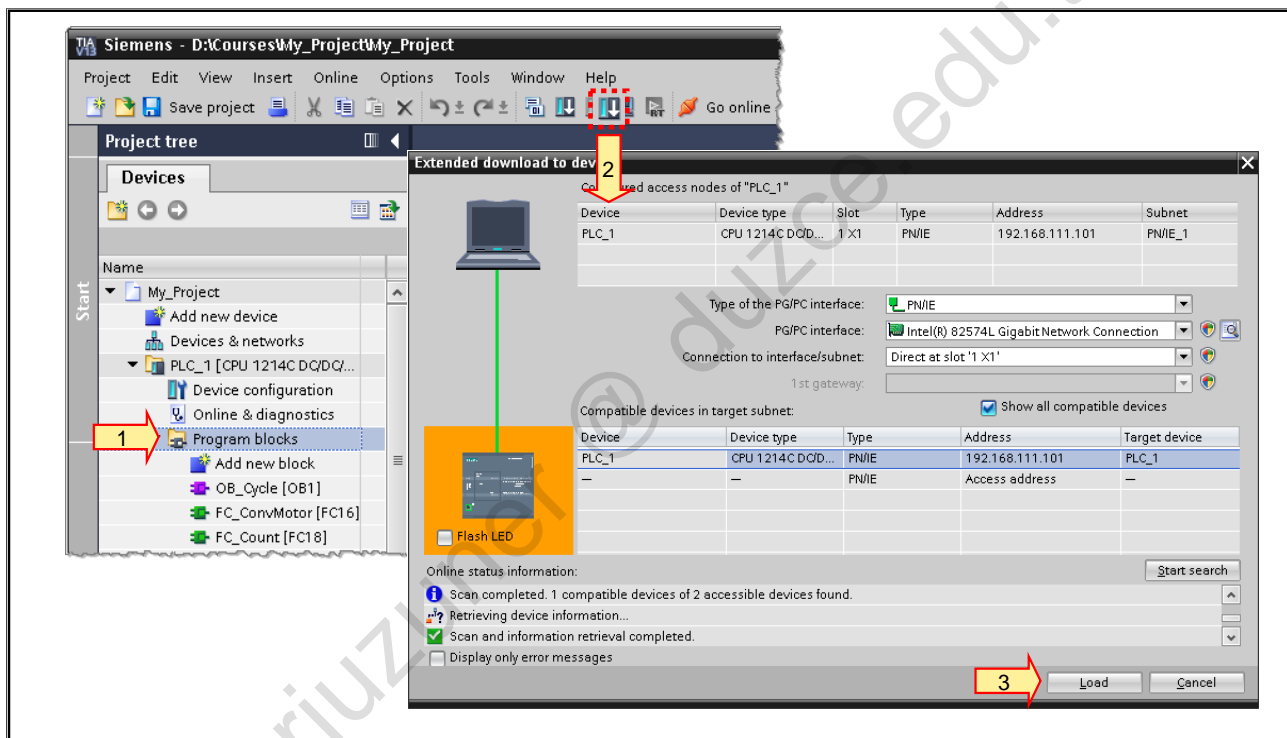
### Compiling a Block

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.

## 2.9.15. 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?

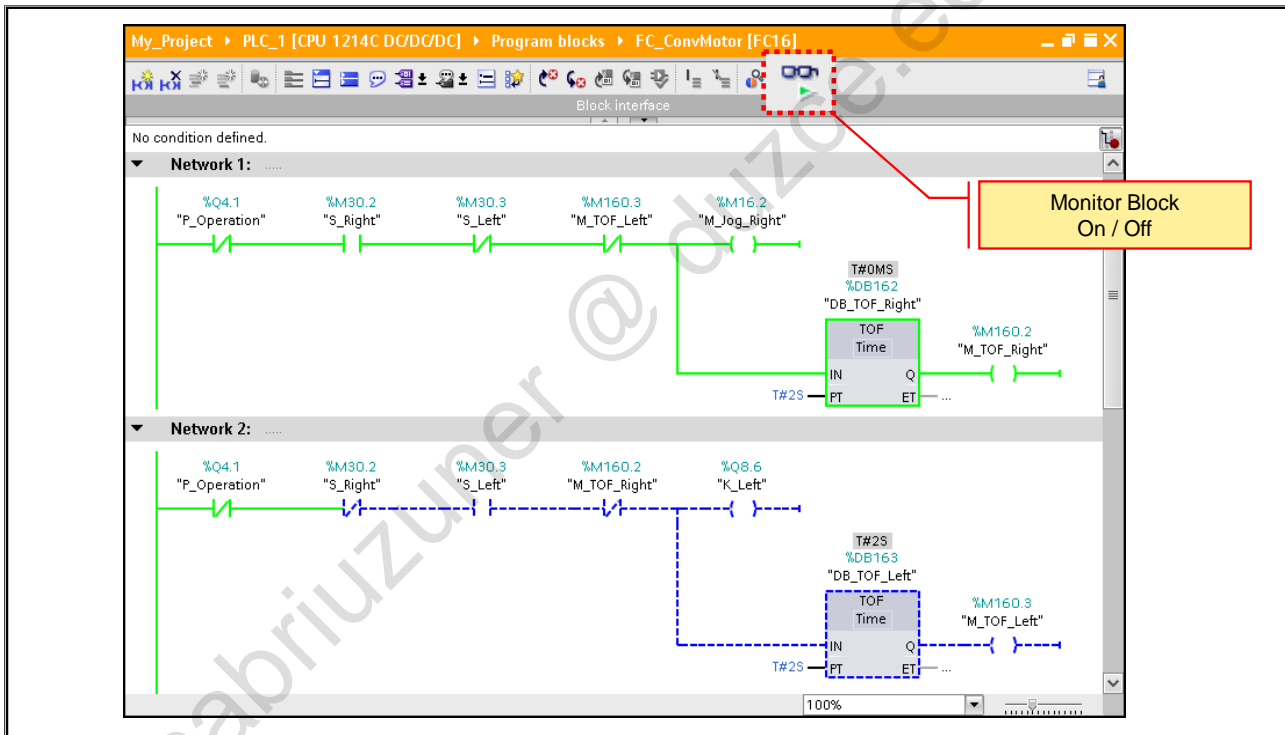
#### Selection via: Context menu of device > Download to device

- 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.

## 2.9.16. Monitoring a Block



### Monitoring blocks

The test function monitor block is used to be able to follow the program execution within a block. The status 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 the block.

Examples:

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

## 2.9.17. Block Networks

The screenshot displays the SIMATIC Manager interface for configuring a block network. The main window shows 'Network 4: Count Parts' with the title 'Detect the current number'. The network contains a block call for 'FB\_Count' with the following parameters:

- EN: P\_Operation
- CU: B\_LB
- CD: ...
- Set\_Nr: 5

Below the block call, there are two instructions:

- QU → "M\_Act\_No=0"
- QD → "M\_Act=M\_Set"

A comment 'Count the parts' is associated with the network. The interface includes various toolbars and options for network management, such as 'Open / Close all networks', 'Open / Close a network', 'Absolute / Symbolic operands', 'Update inconsistent block calls', 'Network comments On / Off', 'Free comments On / Off', 'Shows tag information On / Off', and 'Networks Insert / Delete'.

Symbolic Name	Address
"Act_No"	%MW200
"B_LB"	%I8.0
"M_Act_No=0"	%M20.0
"M_Act=M_Set"	%M20.1
"P_Operation"	%Q4.1

### Block networks

Just like the user program, a single block can be separated in different networks. Each network can have a headline and comment. Inside a network, free comments can be assigned to instructions.



# Contents

<b>3.</b>	<b>Analog value processing .....</b>	<b>3-2</b>
3.1.	Objectives .....	3-2
3.2.	Task description .....	3-3
3.3.	Principle of analog value processing .....	3-4
3.4.	Properties of analog input modules .....	3-6
3.5.	Properties of analog output modules .....	3-8
3.6.	Analog value representation and measured value resolution .....	3-10
3.7.	Analog value representation of different measuring ranges .....	3-11
3.8.	Analog value representation for the analog outputs .....	3-12
3.9.	Scaling analog inputs with NORM_X and SCALE_X (1) .....	3-13
3.9.1.	Scaling analog inputs with NORM_X and SCALE_X (2) .....	3-14
3.10.	Controlling analog outputs with NORM_X and SCALE_X .....	3-15
3.11.	Comparator operations: IN_RANGE and OUT_RANGE .....	3-16
3.12.	Cyclic interrupts .....	3-17
3.12.1.	Phase offsets for cyclic interrupts .....	3-18
3.13.	Task description: Fault evaluation on the analog channel .....	3-19
3.13.1.	Exercise 1: Parameterizing the Analog Module SM 1234 .....	3-20
3.13.2.	Exercise 2: Hardware diagnostics for diagnostic interrupt .....	3-21
3.13.3.	Exercise 3: Evaluating the diagnostics buffer of the CPU .....	3-22
3.14.	Task description: Converting the analog value and outputting it on the touchpanel .....	3-23
3.14.1.	Exercise 4: Inserting "OB_Cyclic interrupt" .....	3-24
3.14.2.	Exercise 5: Programming Analog Value Processing and Lock Outs .....	3-25
3.14.3.	Exercise 6: Downloading blocks into the CPU and testing the display on the touchpanel .....	3-26
3.15.	Additional Information .....	3-27
3.15.1.	Additional exercise: Return of reject parts .....	3-28

### 3. Analog value processing

#### 3.1. Objectives

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

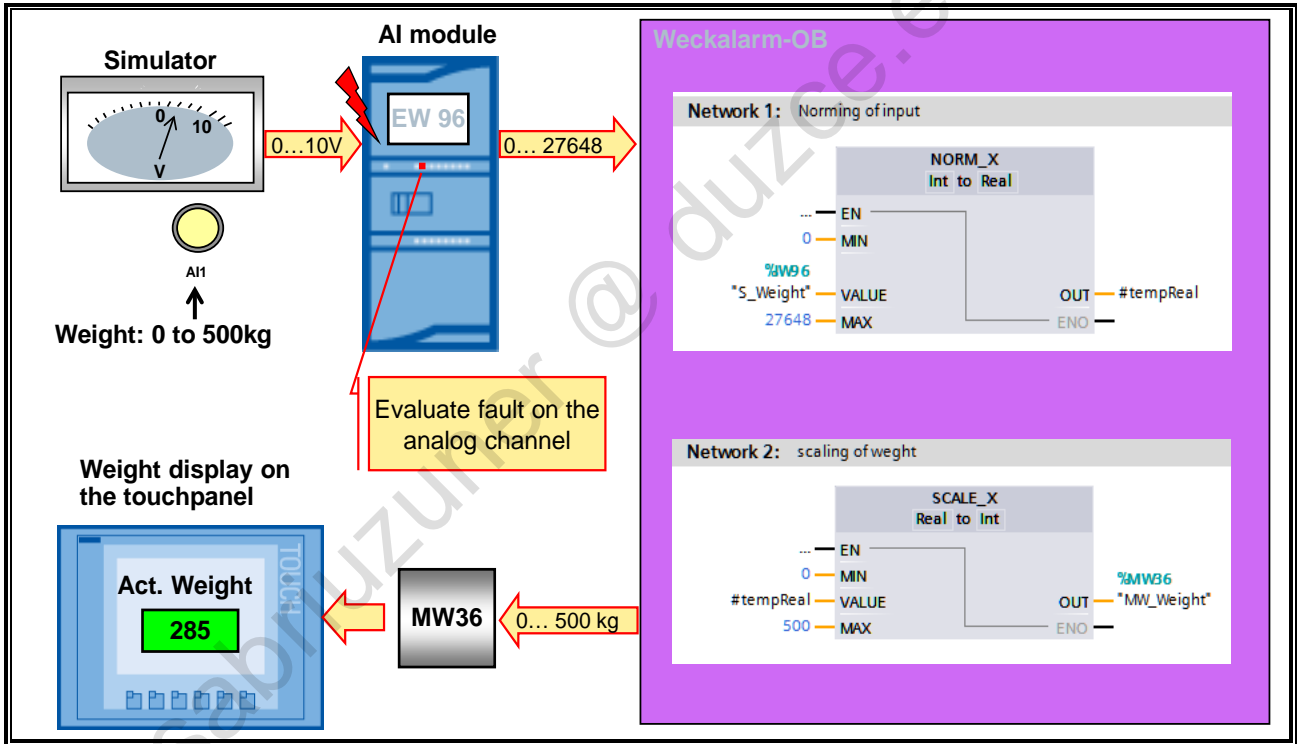
- ... be familiar with the principle of analog value processing
- ... be able to assign parameters to an analog module
- ... be able to address an analog module
- ... be able to interpret the resolution of a module
- ... be familiar with the operations for the analog value conversion
- ... be able to program a simple analog value conversion
- ... be able to evaluate the diagnostics interrupt of the analog module
- ... be familiar with the principle of interrupt processing
- ... be able to generate and program a cyclic interrupt

#### Objectives

In this chapter, the principle of analog value processing is presented. The goal is that the participant can parameterize an analog module and of interpreting the resolution.

Furthermore, the necessary conversion operations are presented to be able to process an analog value. The participant should be able to program a simple analog value conversion and be able to interpret a diagnostics interrupt of an analog module.

### 3.2. Task description



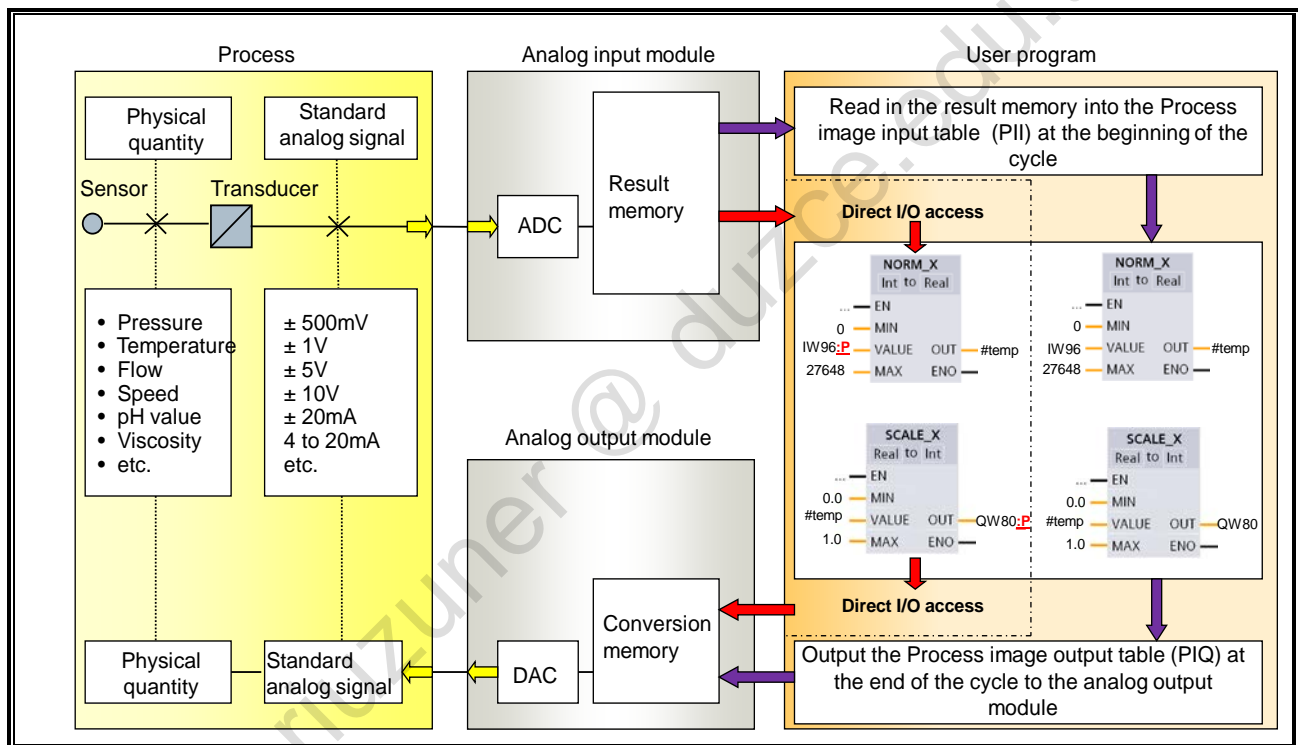
#### Task description

In this chapter, the conversion and processing of analog signals is handled.

For this, a voltage is to be set and read in on the simulator potentiometer. This voltage simulates part weight values. It will be your task to convert the read in values every 250ms in the cyclic interrupt into weight values between 0 kg and 500 kg using the operations NORM\_X and SCALE\_X. The weight is only valid in the range of 100kg to 400kg. If the weight of the part exceeds or falls below these limits, the part is considered invalid and no further transport sequence can be started (Bay LEDs remain dark and conveyor movement to the right cannot be started).

As well, you will learn how you must proceed when there is a channel fault of an analog module to get detailed information on the fault event.

### 3.3. Principle of analog value processing



#### Principle of analog value processing

In a production process, there are a variety of physical quantities (such as pressure, temperature, speed, rotational speed, pH value, and viscosity etc.) that need to be processed in the PLC for automation purposes.

#### Sensor

Measuring sensors respond to changes in the quantity to be measured by such things as linear expansion, angular ductability, and alteration of electrical conductivity.

#### Transducer

Measuring transducers convert these above-mentioned changes into standard analog signals, such as:  $\pm 500\text{mV}$ ,  $\pm 10\text{V}$ ,  $\pm 20\text{mA}$ , 4 to 20mA.

These signals are supplied to the analog input modules.

#### ADC

Before these analog values can be processed in the CPU, they must be converted to digital form. The ADC (Analog-to-Digital Converter) on the analog input module handles this conversion.

The analog-to-digital conversion is performed sequentially. This means the signals are converted for each analog input channel in turn.

#### Result memory

The result of the conversion is stored in the result memory and remains there until it is overwritten by a new value.

You can use the "IW...:P" addressing to read the converted analog value directly from the I/O.

### Analog output

The (MOVE) transfer instruction is used to write the analog values the user program calculated to an analog output module, where a DAC (Digital-to-Analog Converter) converts them to standard analog signals.

### Analog Actuators

You can connect standard actuators directly to the analog output modules.

### Analog value conversion <-> physical unit

For this purpose there are system blocks in converters in the task card "instructions": NORM\_X, SCALE\_X,

### Direct peripheral access ":P"

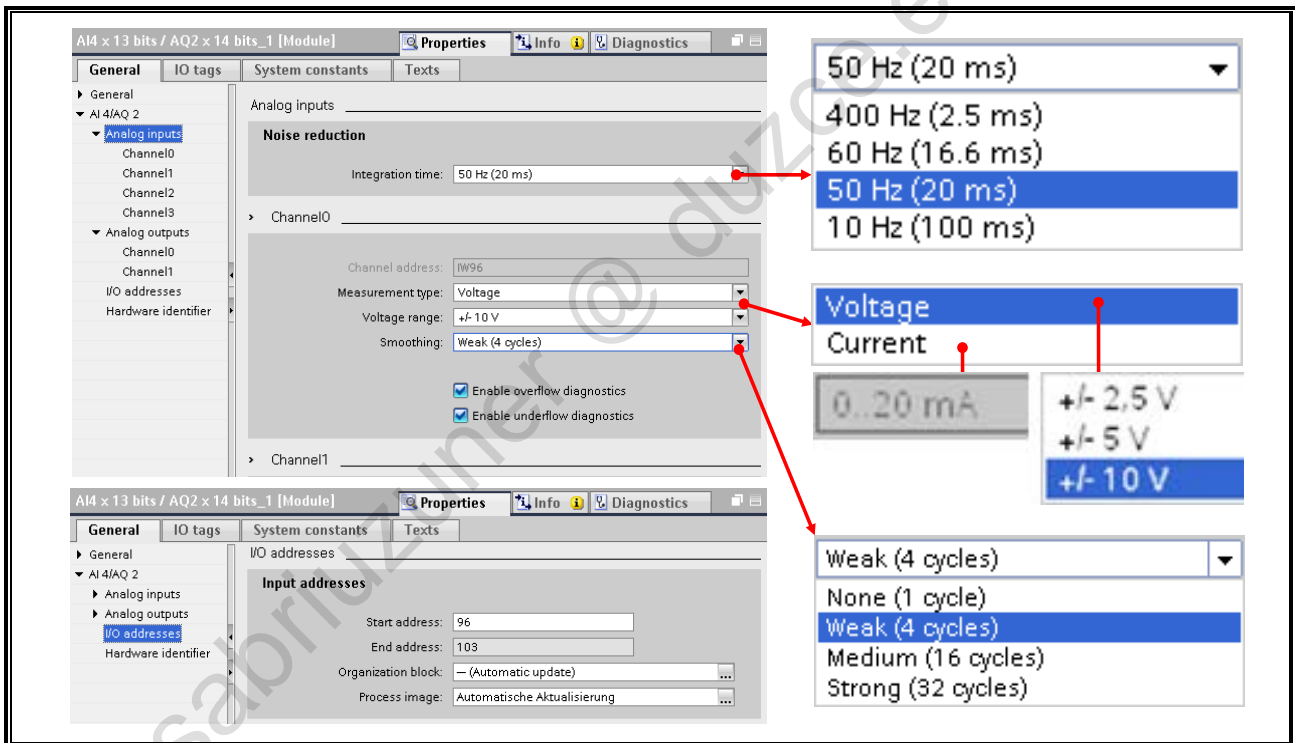
Direct peripheral access is identified by the ":P" addition, which can be programmed in conjunction with the absolute address or symbolic name of the analog channel.

- Reading: current instantaneous value of the analog channel is read
- Writing: output value to the analog channel becomes process effective immediately

### Access via process image

- Reading: The value "frozen for one cycle" from the process image of the inputs is read. This is the value on the AI module at the beginning of the CPU cycle or start of an OB.
- - Writing: The present value is written into the process image of the outputs. This value is only written to the AA module after the cycle or the parameterized OB has ended and only then is it effective in the process.

### 3.4. Properties of analog input modules



#### Analog input modules

In STEP7, analog input modules are configured and assigned parameters in the Device configuration of the respective PLC. The settings or parameters of all modules are downloaded into the CPU. The CPU must be in the STOP state to do this. In a subsequent CPU warm restart, the CPU transfers these parameters to the relevant modules.

#### Parameters

For the respective module, differentiation is made between module parameters and channel parameters.

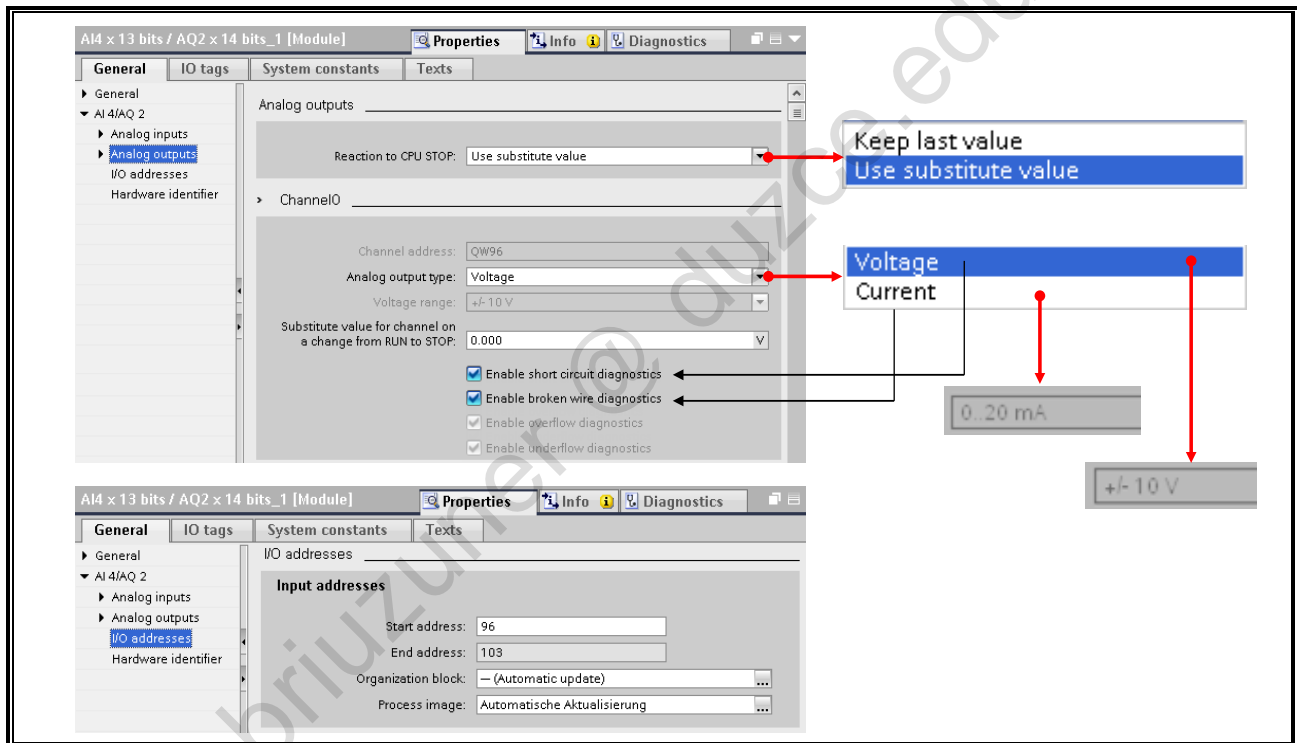
#### Module parameters

- **General**  
Name and comment for the integrated analog inputs of the CPU.
- **Noise Reduction**  
In the noise reduction, the noise frequencies of the specified frequency (in Hz) are suppressed by the integration time which is set.
- **I/O Addresses and Hardware Identifier**  
The address space of the entry addresses as well as the process image is defined. The hardware identity of the device is displayed.

### Channel parameters

- **Measurement type**  
The type of measurement, such as voltage, is set with this parameter. An unused channel must then be deactivated since it is otherwise also converted which would result in a longer total conversion time of the module.
- **Measuring Range (in the picture – Voltage range)**  
With this parameter, the measuring range of the selected type of measurement is set.
- **Smoothing**  
The smoothing of analog values generates a stable analog signal for further processing. Smoothing the analog values is recommended in case of fast signal changes (measured value changes), for example, in the level measurement of fluctuating liquids.
- **Underflow Diagnostics**  
Through this parameter, the underflow diagnostics is activated. If the measured value falls below the underflow range of the channel, a diagnostic interrupt is triggered.
- **Overflow Diagnostics**  
Through this parameter, the overflow diagnostics is activated. If the measured value exceeds the overflow range of the channel, a diagnostic interrupt is triggered.

### 3.5. Properties of analog output modules



#### Analog output modules

In STEP7, analog output modules are configured and assigned parameters in the Device configuration of the respective PLC. The settings or parameters of all modules are downloaded into the CPU. The CPU must be in the STOP state to do this. In a subsequent CPU warm restart, the CPU transfers these parameters to the relevant modules.

#### Parameters

For the respective module, differentiation is made between module parameters and channel parameters.

#### Module parameters

- General  
Name and comment for the integrated analog outputs of the CPU.
- Reaction to CPU STOP
  - Use substitute value  
The peripheral device outputs the value previously set for the channel.
  - Keep last value  
The peripheral device retains the value last put out before STOP.



#### Caution!

Make sure that the system is always in safe mode in the case of "Keep last value"!

- I/O Addresses and Hardware Identifier  
The address space of the entry addresses as well as the process image is defined. The hardware identity of the device is displayed.



### Channel Parameters

- **Output Type**  
The type of output, such as voltage, is set with this parameter. Unused outputs must be deactivated since these are otherwise also converted which would result in a longer total conversion time of the module.
- **Output Range (in the picture – Voltage range)**  
The output range of the selected type of output is set with this parameter.
- **Broken Wire Diagnostics (in Current mode)**  
With this parameter, the diagnostic Wire break is generated when there is a wire break. This diagnostic is not noticeable in the zero range.
- **Short-circuit Diagnostics (in Voltage mode)**  
With this parameter, a diagnostic is generated when there is a short-circuit of the output wire. This diagnostic is not noticeable in the zero range.
- **Overload Diagnostics**  
With this parameter, a diagnostic is generated when there is an overload.
- **Substitute value**  
With this parameter, a substitute value is specified which the module is to output when the CPU goes into STOP. The substitute value must be in the rated range, the over range or the under range.

### 3.6. Analog value representation and measured value resolution

Bit no.		min. units		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit value		Dec.	Hex.	VZ	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
Reso- lution in bits + sign (VZ)	8	128	80	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
	9	64	40	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0
	10	32	20	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0
	11	16	10	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
	12	8	8	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0
	13	4	4	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0
	14	2	2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0
	15	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

\* = 0 or 1

#### Representation

Negative analog values are represented as the two's complement.  
The value is positive if bit No. 15=0 and negative if bit No.15=1.

#### Resolution

If the resolution of an analog module is less than 16 bits, the analog value is written into the accumulator (module result memory) left-justified. The unused less significant bit positions are filled with "0"s.

#### Accuracy

Resolutions of between 8 and 16 bits are possible, depending on the type of module.

### 3.7. Analog value representation of different measuring ranges

Range	Voltage such as:		Current such as:		Resistance such as:		Temperature e.g. Pt100 (Standard)	
	Meas.range $\pm 10V$	Units	Meas.range 4 to 20mA	Units	Meas.range 0 to 300Ohm	Units	Meas.range -200 to +850°C	Units
Overflow	$\geq 11.76$	32767	$\geq 22.815$	32767	$\geq 352.778$	32767	$\geq 1000.1$	32767
Over range	11.7589 ⋮ 10.0004	32511 ⋮ 27649	22.810 ⋮ 20.0005	32511 ⋮ 27649	352.767 ⋮ 300.011	32511 ⋮ 27649	1000.0 ⋮ 850.1	10000 ⋮ 8501
Rated range	10.00 7.50 ⋮ -7.5 -10.00	27648 20736 ⋮ -20736 -27648	20.000 16.000 ⋮ 4.000	27648 20736 ⋮ 0	300.000 225.000 ⋮ 0.000	27648 20736 ⋮ 0	850.0 ⋮ -200.0	8500 ⋮ -2000
Under range	- 10.0004 ⋮ - 11.759	- 27649 ⋮ - 32512	3.9995 ⋮ 1.1852	- 1 ⋮ - 4864	negative values not possible	- 1 ⋮ - 4864	- 200.1 ⋮ - 243.0	- 2001 ⋮ - 2430
Underflow	$\leq - 11.76$	- 32768	$\leq 1.1845$	- 32768		- 32768	$\leq - 243.1$	- 32768

#### Voltage, Current (Symmetrical)

Converting the symmetrical voltage or current ranges results in a rated range of -27648 to +27648.

- $\pm 80\text{mV}$
- $\pm 250\text{ mV}$
- $\pm 500\text{ mV}$
- $\pm 1\text{ V}$
- $\pm 2,5\text{ V}$
- $\pm 5\text{V}$
- $\pm 10\text{V}$
- $\pm 3,2\text{ mA}$
- $\pm 10\text{ mA}$
- $\pm 20\text{ mA}$

#### Voltage, Current (Asymmetrical)

Converting the asymmetrical voltage or current ranges results in a rated range of 0 to +27648.

- 0 ... 2 V
- 1 ... 5 V
- 0 ... 20 mA
- 4 ... 20 mA

#### Resistance

Converting the resistance ranges results in a rated range of 0 to +27648.

- 0 to 150 Ohm
- 0 to 300 Ohm
- 0 to 600 Ohm

#### Temperature

Temperatures are measured with resistance thermometers or thermocouples. Converting results in a rated range of ten times the temperature range

### 3.8. Analog value representation for the analog outputs

Range	Units	Voltage			Current		
		Output ranges:			Output ranges:		
		0 to 10V	1 to 5V	± 10V	0 to 20mA	4 to 20mA	± 20mA
Overflow	$\geq 32767$	0	0	0	0	0	0
Over range	32511 ⋮ 27649	11.7589 ⋮ 10.0004	5.8794 ⋮ 5.0002	11.7589 ⋮ 10.0004	23.515 ⋮ 20.0007	22.81 ⋮ 20.005	23.515 ⋮ 20.0007
Rated range	27648 ⋮ 0	10.0000 ⋮ 0	5.0000 ⋮ 1.0000	10.0000 ⋮ 0	20.000 ⋮ 0	20.000 ⋮ 4.000	20.000 ⋮ 0
	-6912 ⋮ -6913 ⋮ -27648	0	0.9999 ⋮ 0	0	0	3.9995 ⋮ 0	0
Under range	-27649 ⋮ -32512		0	-10.0000 ⋮ -11.7589			-20.000 ⋮ -23.515
Underflow	$\leq -32513$			0			0

#### Voltage, Current (Symmetrical)

For symmetrical voltage or current ranges, a rated range of -27648 to +27648 is converted to:

- ± 10V
- ± 20mA

#### Voltage, Current (Asymmetrical)

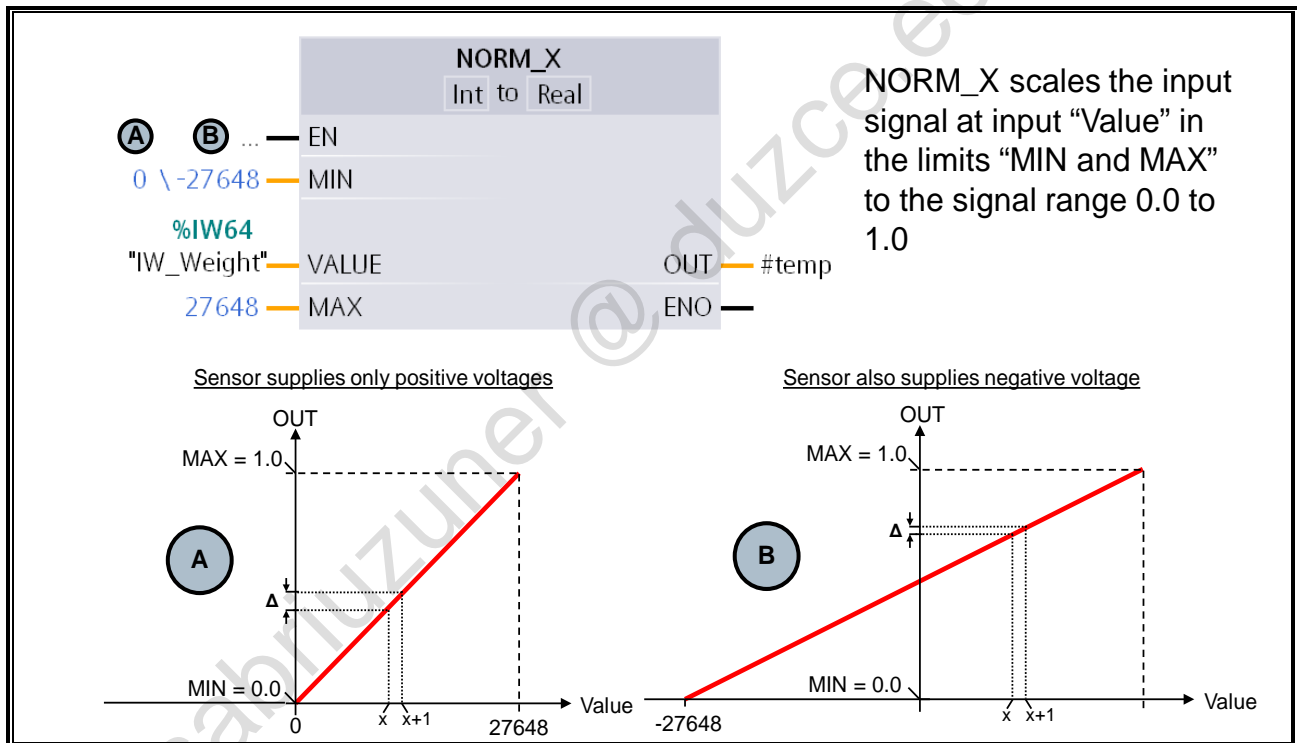
For asymmetrical voltage or current ranges, a rated range of 0 to +27648 is converted to:

- 0 to 10V
- 1 to 5V
- 0 to 20mA
- 4 to 20mA

#### Overflow

If the value to be converted reaches the overflow range, the analog output module is disabled (0V, 0mA).

### 3.9. Scaling analog inputs with NORM\_X and SCALE\_X (1)



#### Norm\_X

The analog module converts the voltage range of -10V to +10V into the value range of -27648 to +27648. The "Normalize" instruction scales a value by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. Depending on the position of this value to be scaled in the value range, the result is calculated and stored as a floating-point number. If the value to be scaled is equal to the value at the MIN input, the instruction returns the value "0.0" as the result. If the value to be scaled is equal to the value at the MAX input, the instruction supplies the result "1.0".

#### Resolution

In example B, the measurement occurs with twice the resolution or with half as much measuring tolerance  $\Delta$ , since the measured value is mapped to the greater units range of -27648 to +27648.

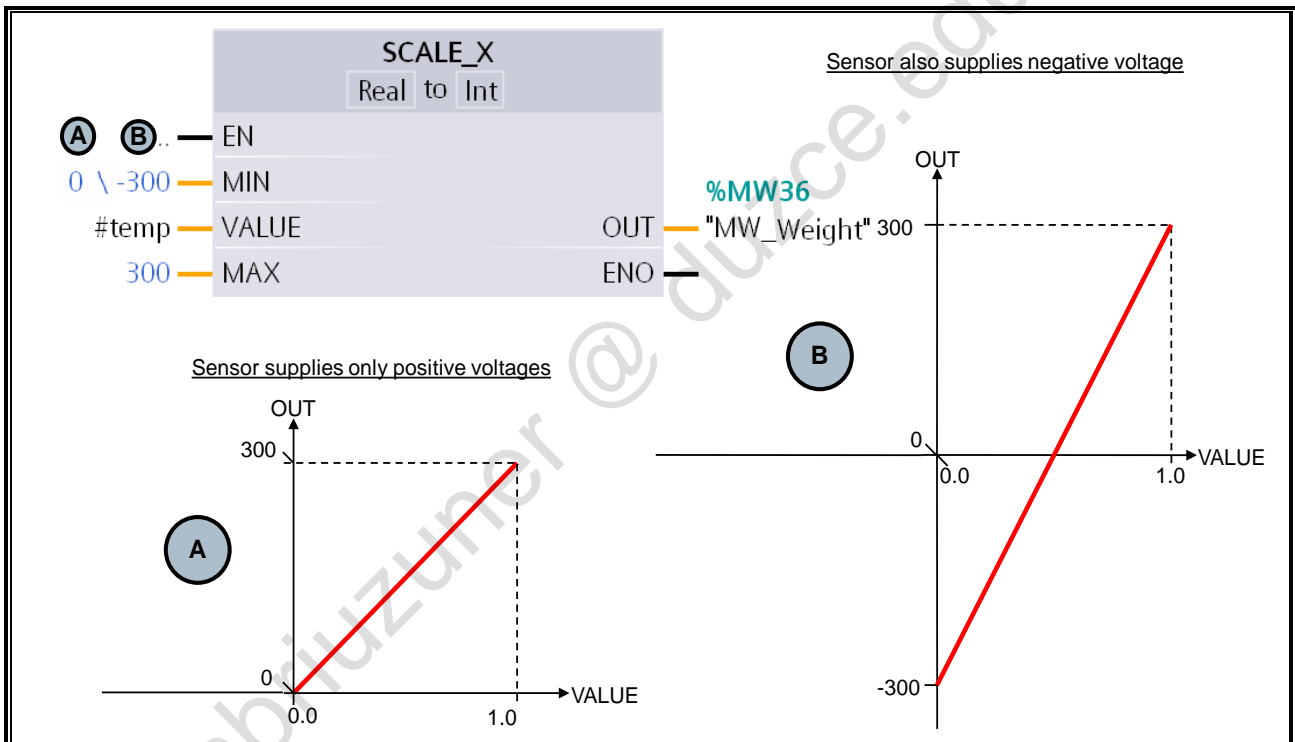
#### Data Types

- The parameters on the input-side can be one of the following data types: SINT, INT, DINT, USINT, UINT, UDINT or REAL
- The parameter OUT can be one of the following data types: REAL or LREAL

#### Parameters

- VALUE: Value which is scaled
- MIN: Lower limit of the value range
- MAX: Upper limit of the value range
- OUT: Scaled signal 0.0 to 1.0

### 3.9.1. Scaling analog inputs with NORM\_X and SCALE\_X (2)




#### SCALE\_X

The "Scale" instruction scales the value at the `VALUE` input linearly by mapping it to a specified value range. When the "Scale" instruction is executed, the floating-point value at the `VALUE` input is scaled to the value range which was defined by the `MIN` and `MAX` parameters. The result of the scaling is an integer which is stored at the `OUT` output.

#### Example

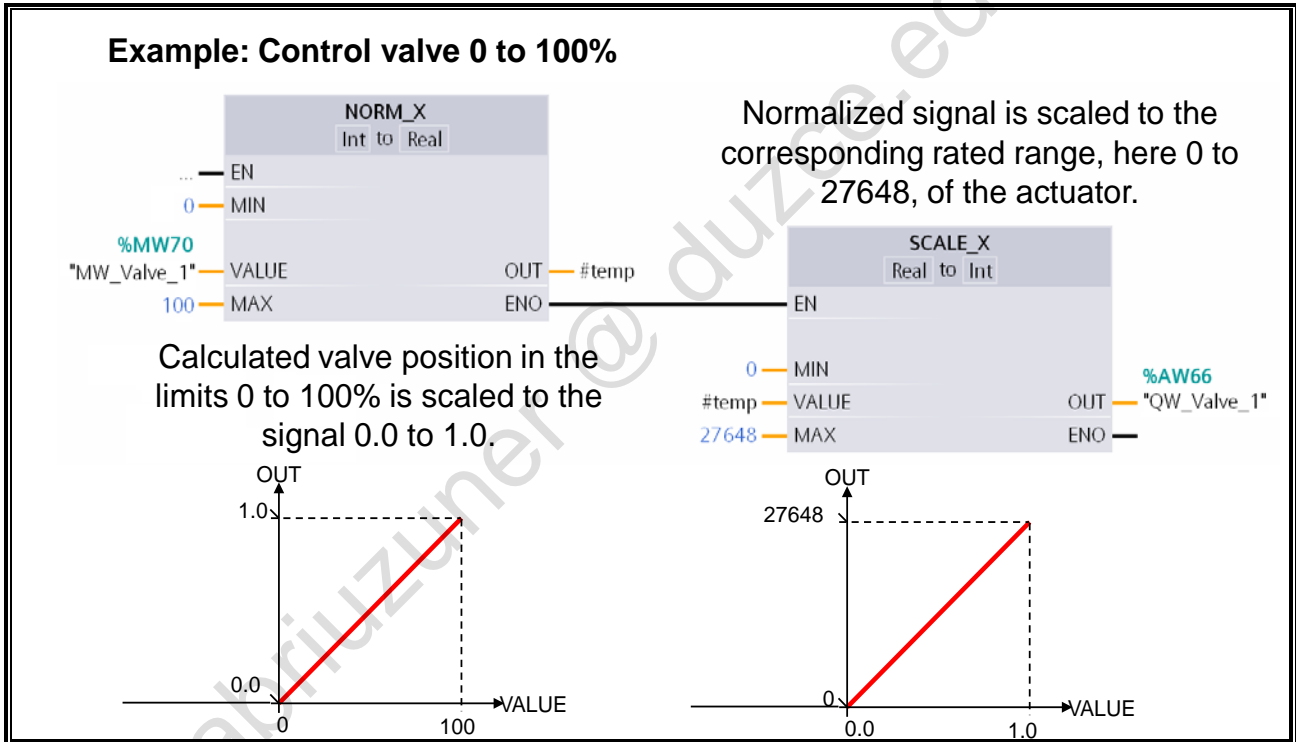
In the example shown, the value at the `VALUE` input is scaled within the limits 0 to 300 for case A. In case B, `VALUE` is scaled to the limits -300 to 300.

 The `VALUE` input may only be within the limits 0.0 to 1.0!

#### Parameters

- `VALUE`: Value which is scaled
- `MIN`: Lower limit of the value range
- `MAX`: Upper limit of the value range
- `OUT`: Result of scaling

### 3.10. Controlling analog outputs with NORM\_X and SCALE\_X



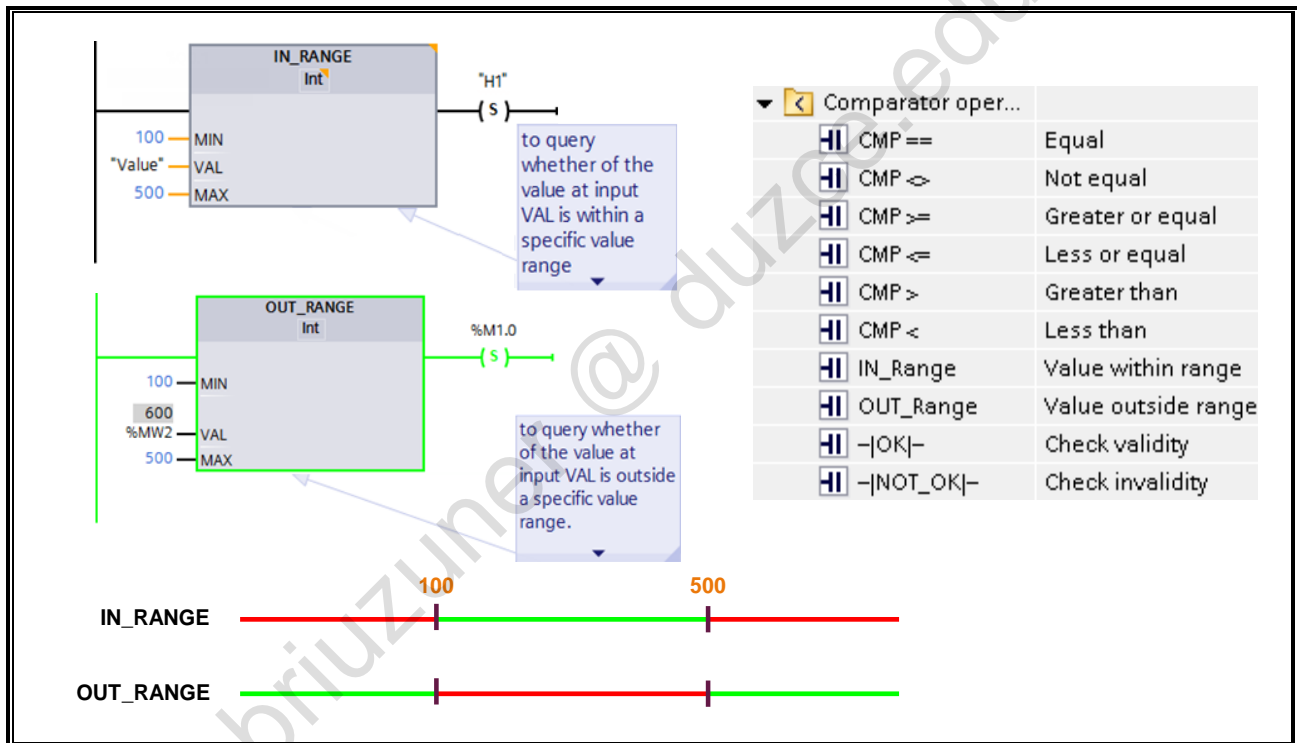
#### Controlling analog outputs (Example)

An analog value (valve position) calculated by the user program in the range 0 to 100% is converted to the range 0 to +27648 through the combination of NORM\_X and SCALE\_X. In outputting the unscaled value to an analog output module, it will control the analog actuator (for example, a servo valve) with, for example, 0V to +10V (depending on the output range set).

The example shows the scaling for an actuator that is to be controlled with the value 0 (0V or 0mA) when the program value is 0%, and with the maximum value (for example, +10V or 20mA) when it is 100%.

Private copy for Sabri Uzuner, sabriuzuner@duzce.edu.tr

### 3.11. Comparator operations: IN\_RANGE and OUT\_RANGE



#### IN\_RANGE

With the "Value within range" instruction you can query whether the value at the VAL input is within a specific value range. You define the limits of the value range with the parameters MIN and MAX. In executing the query, the "Value within range" instruction compares the value at the VAL input with the values of the parameters MIN and MAX and assigns the result to the box output. If the value at the VAL input fulfills the comparison  $\text{MIN} \leq \text{VAL} \leq \text{MAX}$ , the box output has signal state "1". When the comparison is not fulfilled, the box output has signal state "0".

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

#### OUT\_RANGE

With the "Value outside range" instruction you can query whether the value at the VAL input is outside of a specific value range. The limits of the value range are defined through the parameters MIN and MAX. In executing the query, the "Value outside range" instruction compares the value at the VAL input with the values of the parameters MIN and MAX and assigns the result to the box output. If the value at the VAL input fulfills the comparison  $\text{MIN} > \text{VAL}$  or  $\text{VAL} > \text{MAX}$ , the box output has signal state "1". When the comparison is not fulfilled, the box output has signal state "0".

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

#### OK / NOT\_OK

The OK (NOT\_OK) instruction checks whether the value of the variable specified through the box corresponds to a valid REAL or LREAL. If this is the case, the box supplies RLO '1' at its output.



### 3.12. Cyclic interrupts

**Execution of cyclic interrupts:**

The diagram shows a sequence of program execution. It starts with a 'RUN' state. The execution proceeds through several OB1 blocks, followed by an 'O' (off) state, then an OB200 block. This sequence repeats: OB1 blocks, 'O', OB200 block. The intervals between OB200 blocks are labeled 'Interval time'. The diagram also shows 'B1' (block) states for OB1 and OB200. Priority levels 'Prio 7' and 'Prio 1' are indicated on the right.

#### Description

Cyclic interrupt OBs are used to start programs in equidistant intervals regardless of the cyclic program execution.

The time interval defines the intervals in which the cyclic interrupt OB is started and is an integral multiple of the basic clock of 1ms. The phase offset is the time by which the start time is shifted vis-à-vis the basic clock. When several cyclic interrupt OBs are used, you can use this offset to prevent a simultaneous starting time should the time intervals of the cyclic interrupt OBs have a common multiple. You can specify a period between 1 ms and 60000 ms as the time interval.

#### Note

The runtime of every cyclic interrupt OB must be considerably less than its time interval. If a cyclic interrupt OB is not yet completed but is once again pending for processing because the clock has run out, the time error interrupt OB is started. After that, the error-causing cyclic interrupt is carried out or discarded.

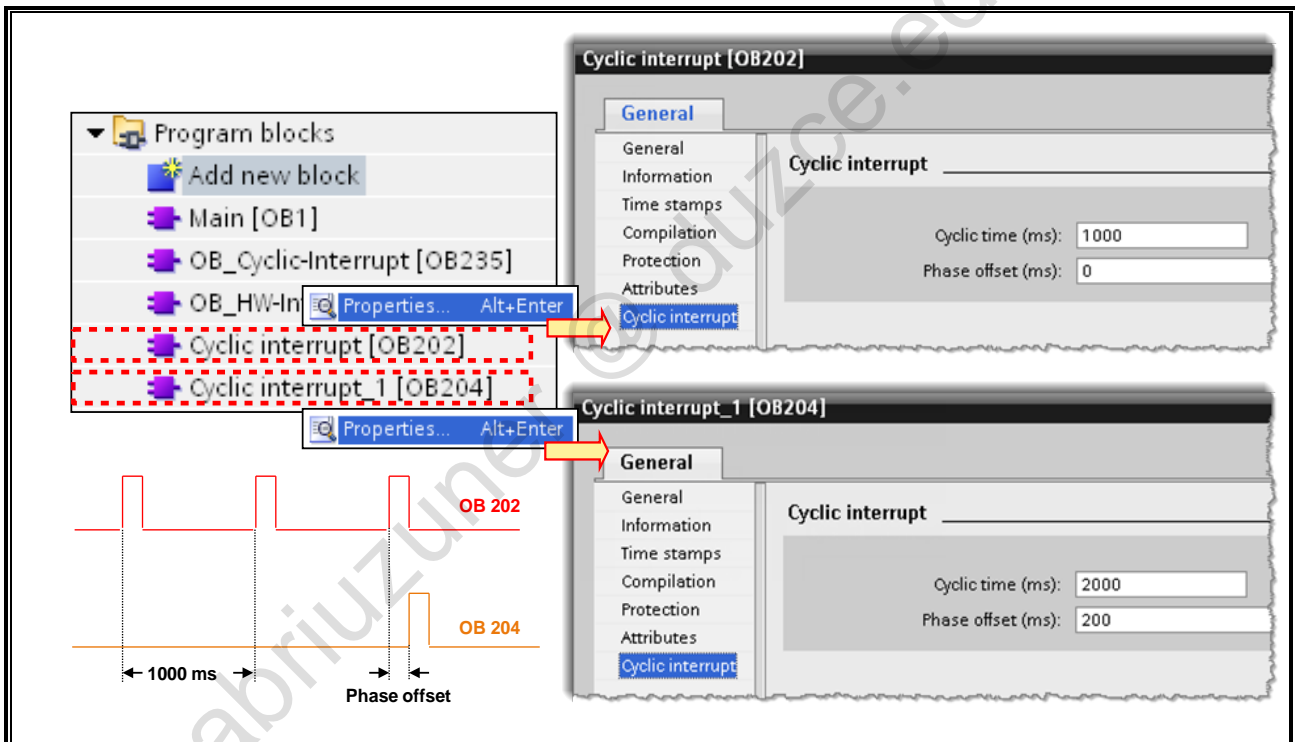
#### Example

You have inserted two cyclic interrupt OBs into your program:

- cyclic interrupt OB1
- cyclic interrupt OB2

For cyclic interrupt OB1, you have set a time interval of 20 ms and for cyclic interrupt OB2, a time interval of 100 ms. After the time interval of 100 ms has run out, cyclic interrupt OB1 reaches its starting time for the fifth time, cyclic interrupt OB2 for the first time. In order to nevertheless process the cyclic interrupt OBs with a time delay, enter a phase offset for one of the two cyclic interrupt OBs.

### 3.12.1. Phase offsets for cyclic interrupts



#### Phase offset

With cyclic interrupt OBs, you can start programs at regular (equidistant) intervals. For this, you have to enter a time interval and a phase offset for every cyclic interrupt OB used.

#### Note

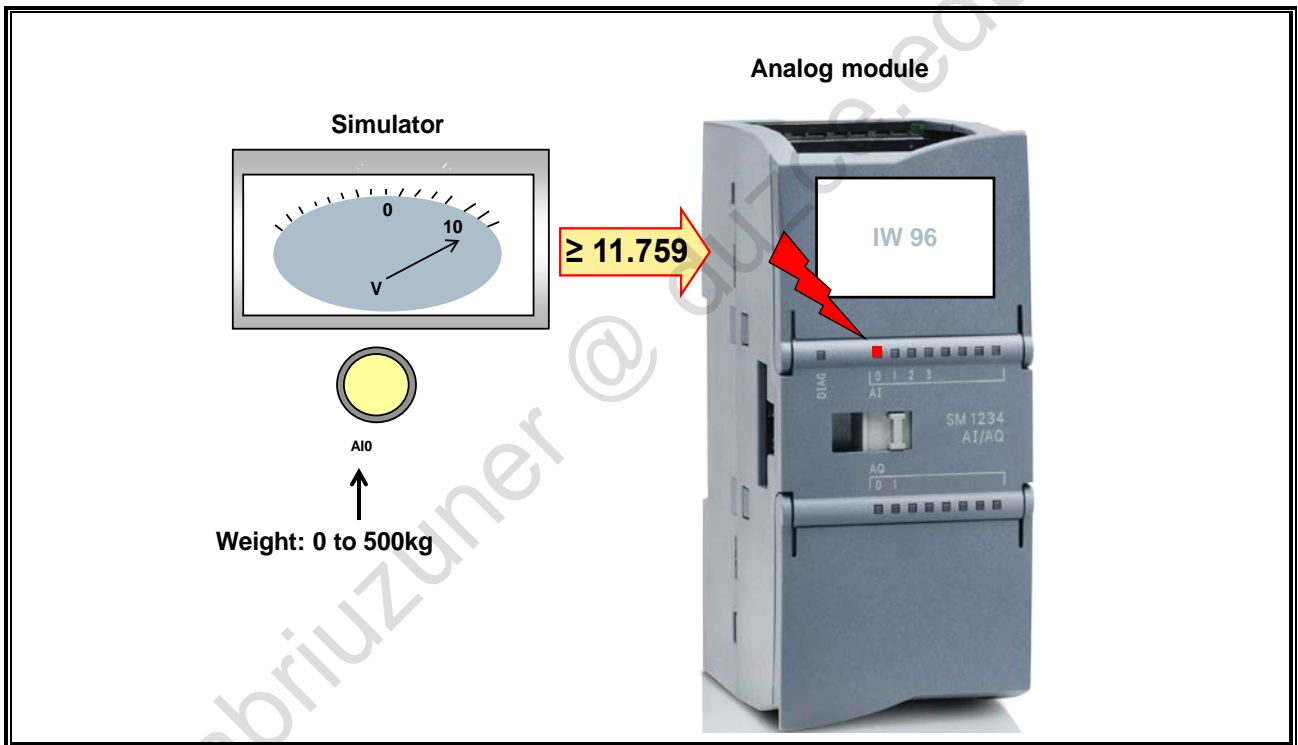
When you assign parameters to several cyclic interrupt OBs, you must give each cyclic interrupt OB a different cyclic time or phase offset in order to prevent a simultaneous execution or a queue. When a cyclic interrupt OB is created, the cyclic time of 100 and the phase offset of 0 are entered as the start value.

#### Procedure

To enter a time interval (cyclic time) and a phase offset for a cyclic interrupt OB, please proceed as follows:

- in the Project tree, open the folder "Program blocks"
- right-click on an existing cyclic interrupt OB
- in the context menu select the command "Properties"
- the dialog "<Name of the cyclic interrupt OB>" is opened
- in the area tree, click on the group "Cyclic interrupt"
- the input fields for the time interval (cyclic time) and the phase offset are displayed
- enter the time interval and the phase offset
- confirm the entries with "OK"

### 3.13. Task description: Fault evaluation on the analog channel

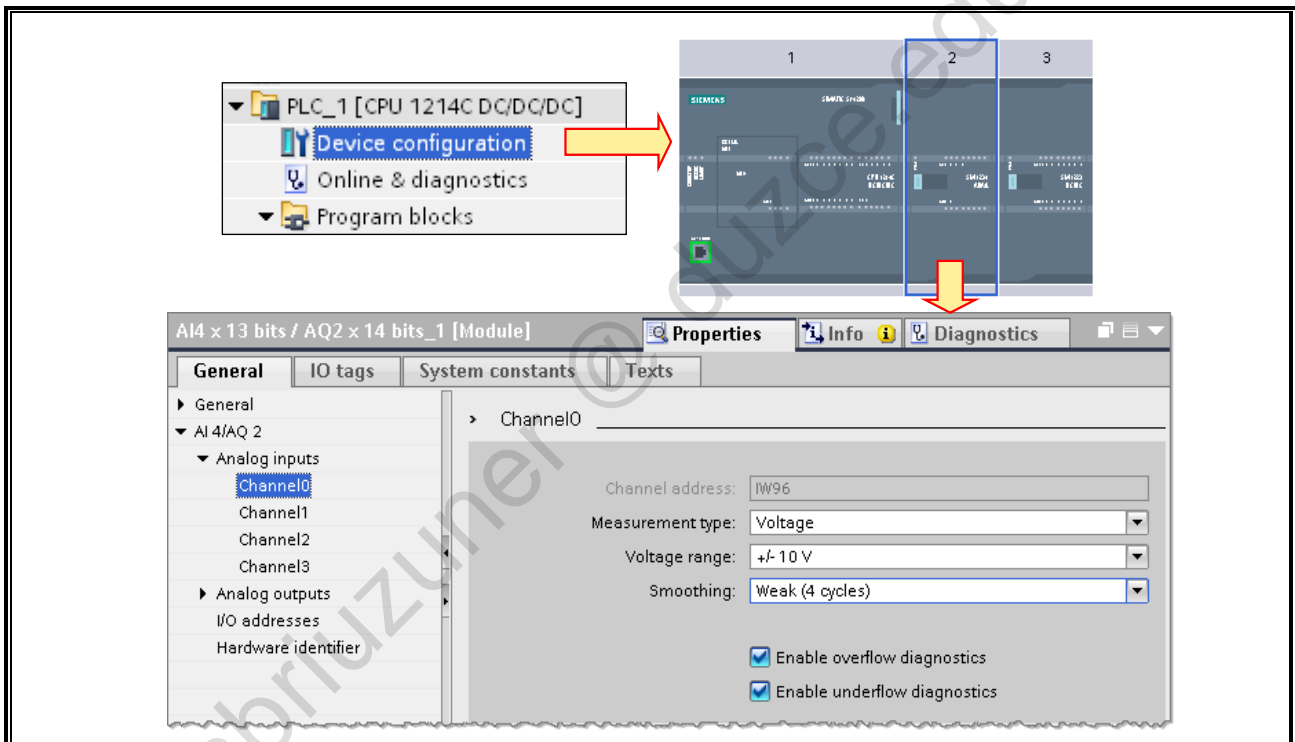


#### Task description

You are to provoke a channel fault on analog channel 0 of the AI4/AO2 module and then evaluate it. For this, the measuring range of the analog input is first to be set to  $\pm 5V$  and then you are to set an input voltage which is too high on the simulator potentiometer.

You are to investigate the fault condition which occurs using the STEP7 online functions.

### 3.13.1. Exercise 1: Parameterizing the Analog Module SM 1234



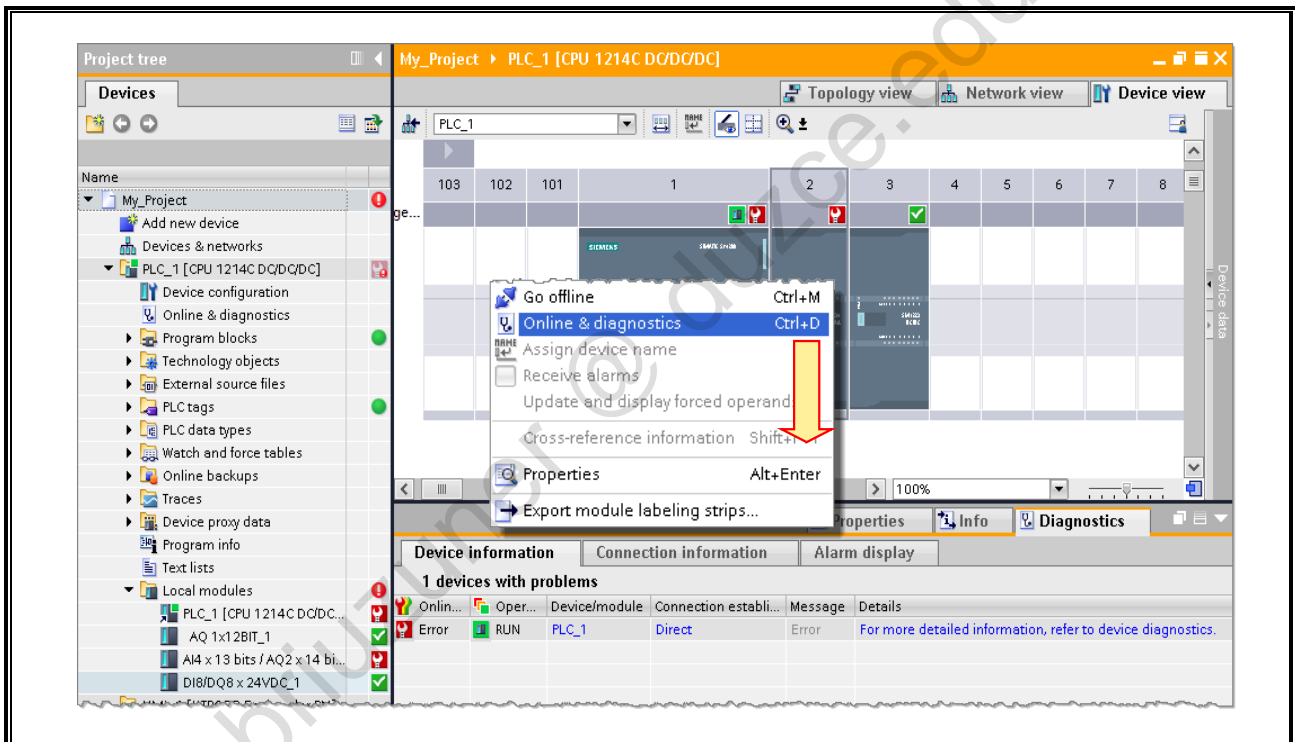
#### Task

Parameterize the analog module SM 1234.

#### What to Do

Make the settings for the analog module as shown in the picture.

### 3.13.2. Exercise 2: Hardware diagnostics for diagnostic interrupt



#### Task

After you have assigned parameters to your analog module in the previous exercise and have activated the diagnostics interrupt, you are now to initiate a diagnostic interrupt by knowingly setting the voltage too high.

After the CPU signals an ERROR because the voltage is too high at the input of the analog module, you are to localize the "error" that occurred by using a simple online connection and, in the next step, read out detailed information from the CPU (see picture).

#### What to Do

1. On the simulator, set a voltage which is either too low or too high (voltage < -11.759V or voltage > +11.759V)
2. Establish an online connection to the CPU  
My\_Project → select Station → Go online
3. Open the list of Local modules  
My\_Project → PLC\_1 → Local modules
4. Open the faulty analog module  
Double-click on the module
5. In the Inspector window, display the "Diagnostics" tab and click the link in "Details". → The diagnostics buffer of the CPU opens.



The **Diagnostic Interrupt OB 82** can be programmed to give you detailed information about the error event. (evaluation of the OB 82 start information).

### 3.13.3. Exercise 3: Evaluating the diagnostics buffer of the CPU

Diagnostics buffer

**Events**

Display CPU Time Stamps in PG/PC local time

No.	Date and time	Event	Event class
1	1/7/2012 12:23:05.641 ...	High limit exceeded	✓ i
2	1/7/2012 12:23:04.646 ...	High limit exceeded	✓ i
3	1/7/2012 12:10:09.329 ...	High limit exceeded	✗ i
4	1/7/2012 12:10:00.509 ...	Follow-on operating mode change - CPU changes from STARTUP to RUN mode	✓ i
5	1/7/2012 12:10:00.405 ...	Communication initiated request: WARM RESTART - CPU changes from STOP to...	✓ i
6	1/7/2012 12:10:00.405 ...	New startup information - Current CPU operating mode: STOP	✓ i
7	1/7/2012 12:09:58.805 ...	New startup information - Current CPU operating mode: STOP	✓ i
8	1/7/2012 12:09:58.705 ...	New startup information - Current CPU operating mode: STOP	✓ i

Freeze display

**Details on event:**

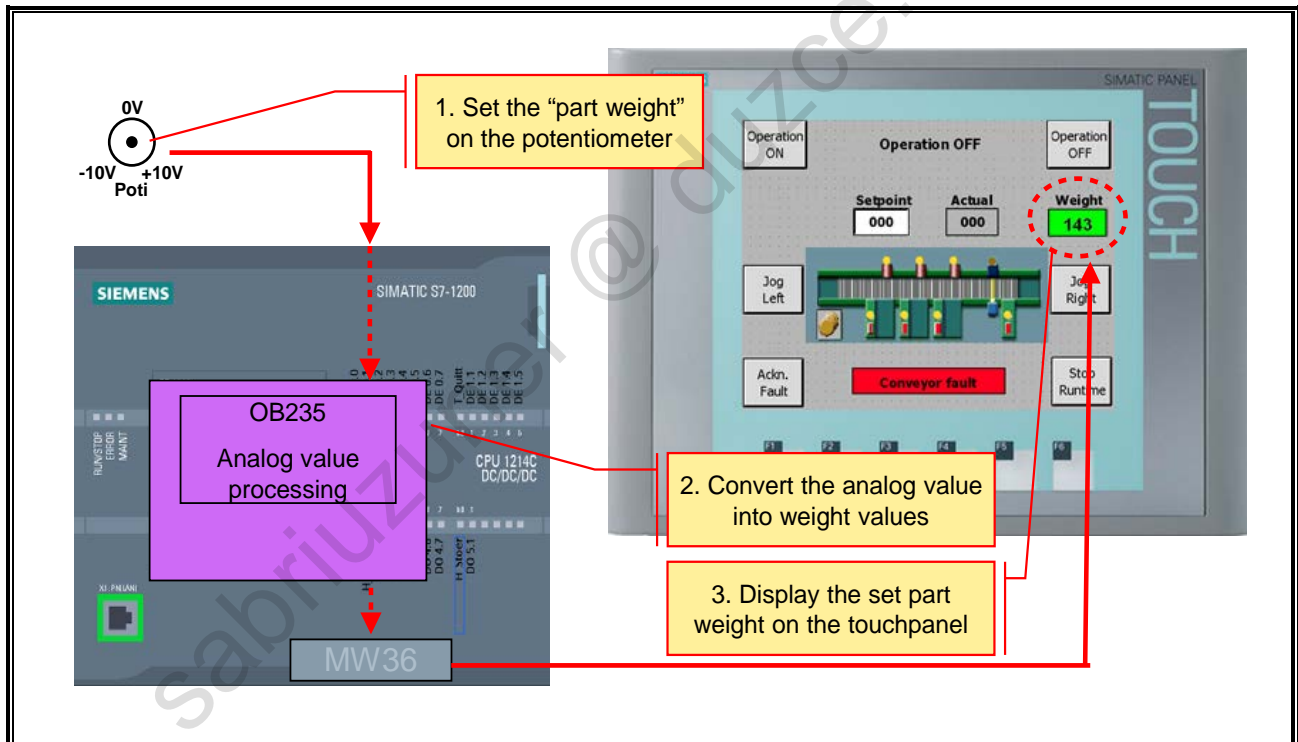
Details on event: 3 of 50      Event ID: 16# 06:01 C0

Description: Error: High limit exceeded on I0 PLC\_1 / AI4 x 13 bits / AQ2 x 14 bits\_1.

Task

Evaluate the information highlighted in the picture.

### 3.14. Task description: Converting the analog value and outputting it on the touchpanel



#### Task Description

An analog value processing is to be programmed in a cyclic interrupt. The converted analog value (part weight: 0 to 500kg) is stored in memory word "MW\_Weight" (MW36) and is then to be displayed on the touchpanel.

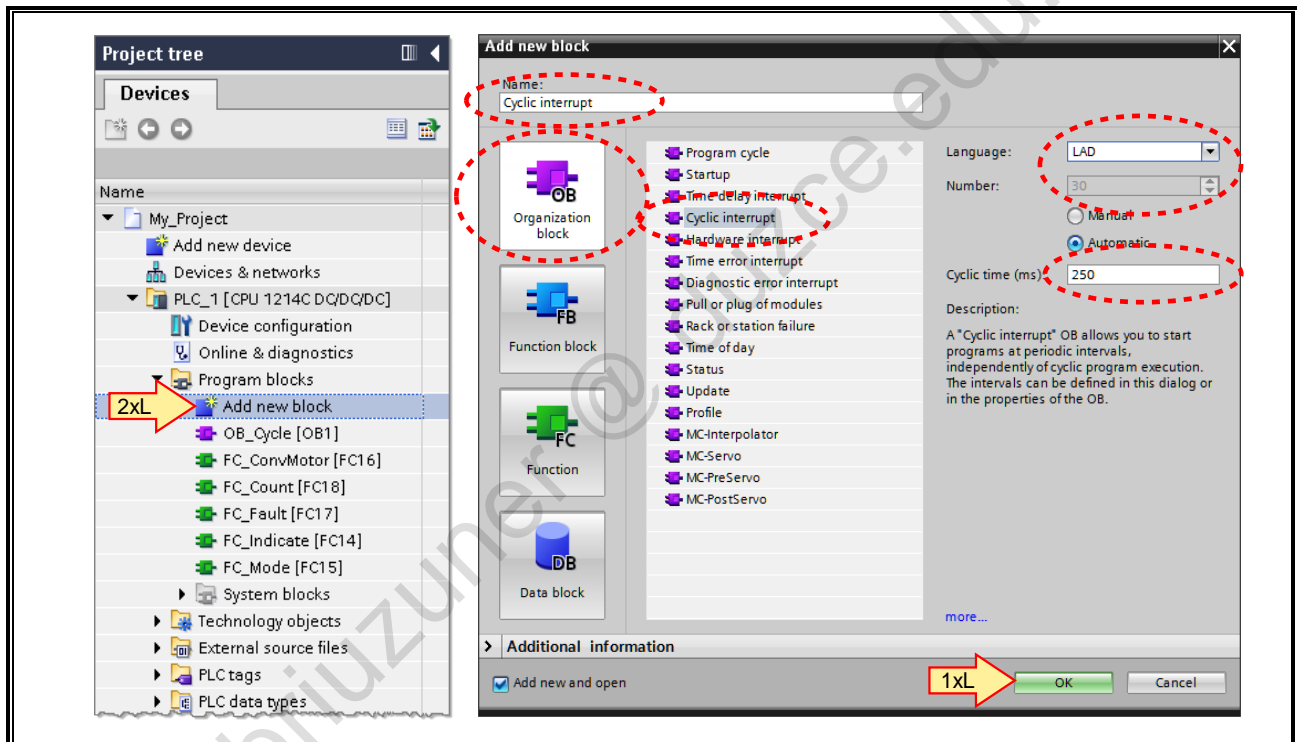
Furthermore, the part weight is to be checked for validity and the result is to be assigned to bit memory "M\_Weight\_OK" (M35.0):

- $100\text{kg} \leq \text{MW36} \leq 400\text{kg} \rightarrow \text{"M\_Weight\_OK"} = \text{TRUE}$
- otherwise  $\rightarrow \text{"M\_Weight\_OK"} = \text{FALSE}$

The bit memory is already linked to the I/O field "Act. Weight" on the touchpanel and influences its background color.

If "M\_Weight\_OK" delivers the value FALSE, the bay indicator lights on the conveyor must remain dark and no new transport sequence can be started. The lock outs "FC\_Signal" and "FC\_Conveyor" necessary for this must be programmed by you.

### 3.14.1. Exercise 4: Inserting "OB\_Cyclic interrupt"



#### Task

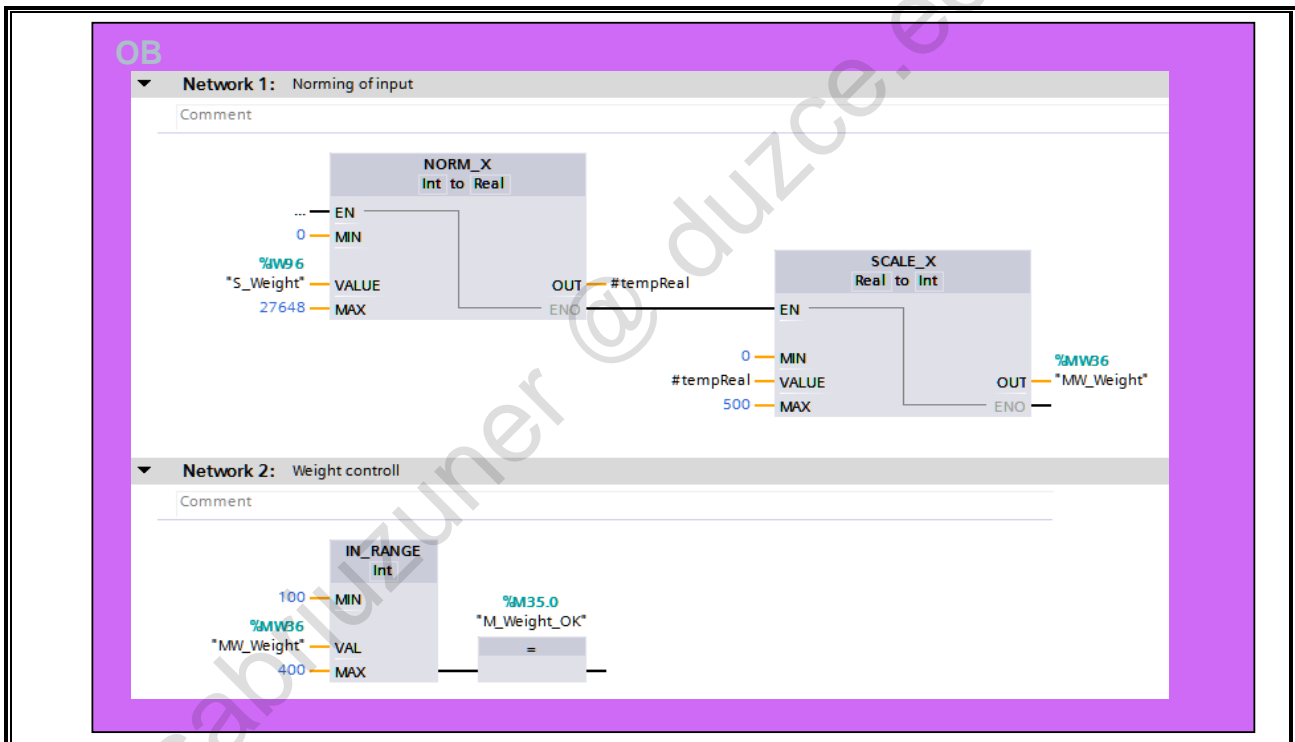
Insert the Cyclic interrupt OB into your user program.

#### What to Do

1. Open the "Add new block" dialog  
My\_Project → PLC\_1 → Program blocks → Double-click on "Add new block"
2. Select the OB type "Cyclic interrupt"
3. Assign the block name "OB\_Cyclic interrupt"
4. Set a cyclic time of 250ms
5. Click on OK



## 3.14.2. Exercise 5: Programming Analog Value Processing and Lock Outs

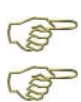


## Task

Program the analog value processing represented in the picture in "Cyclic\_interrupt" and then the lock outs dependent on the part weight in "FC\_Signal" and "FC\_Conveyor".

## What to Do

1. Activate the instructions NORM\_X and SCALE\_X and assign the parameters as shown in the picture.

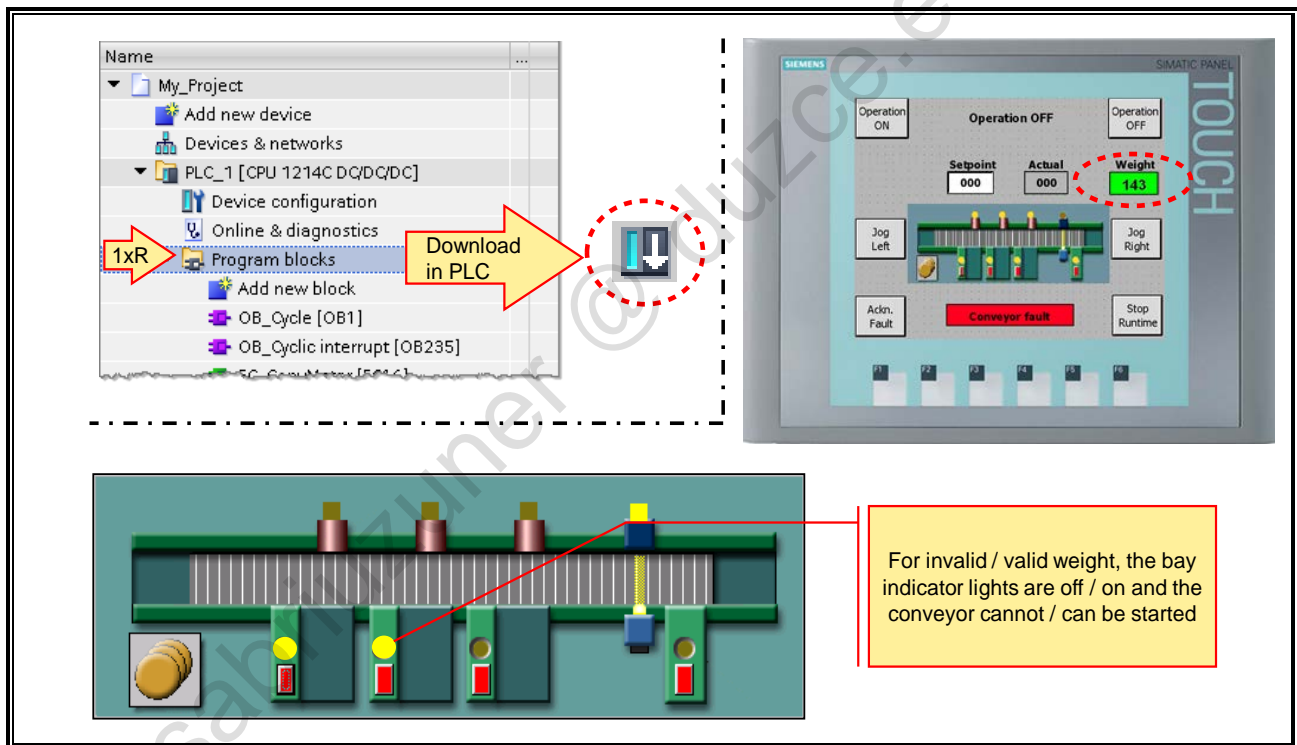


#tempReal is a local, temporary tag variable of the type REAL

PLC tags are already created for the tag variables "S\_Weight" and "MW\_Weight".

2. Program the comparison to find out whether the weight is within the limits 100 to 400 kg. For this, use the instruction "IN\_Range" and assign the RLO to "M\_Weight\_OK".
3. Insert "M\_Weight\_OK" as a lock out in the correct locations in "FC\_Signal" and "FC\_Conveyor".

### 3.14.3. Exercise 6: Downloading blocks into the CPU and testing the display on the touchpanel



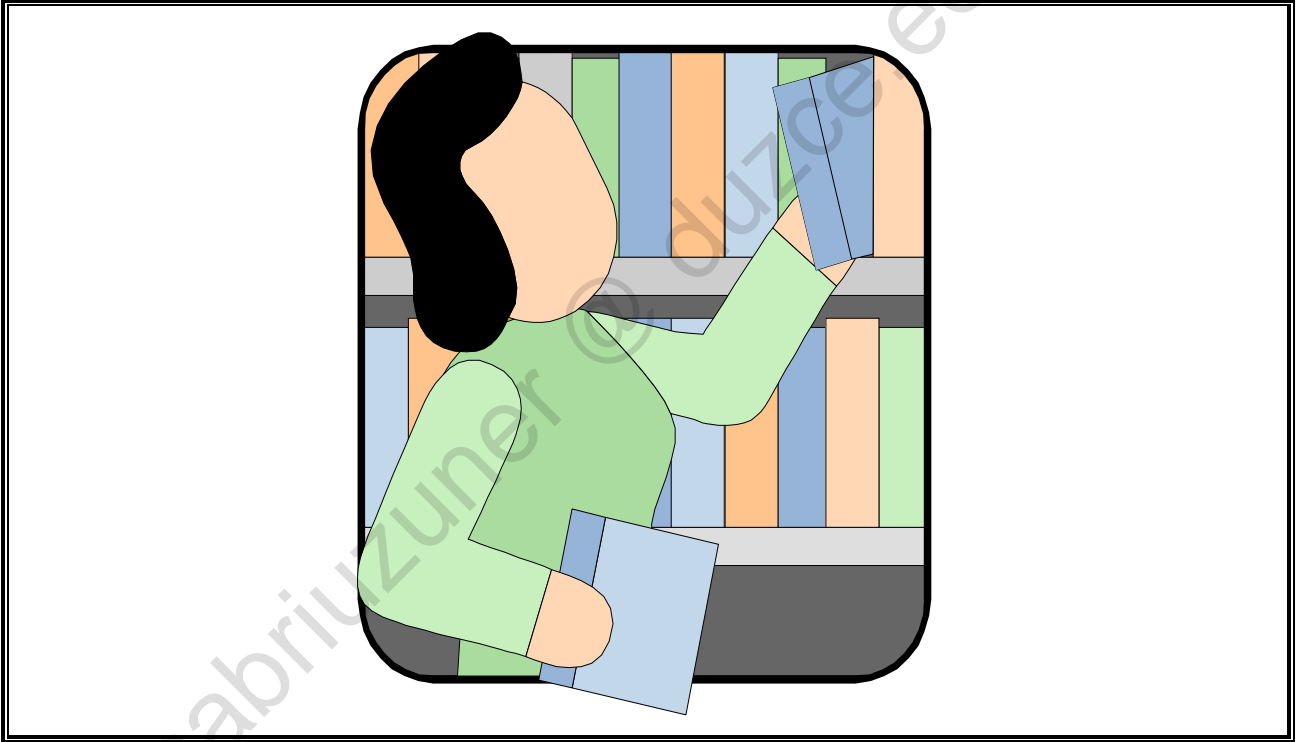
#### Task

Check the functions you previously programmed.

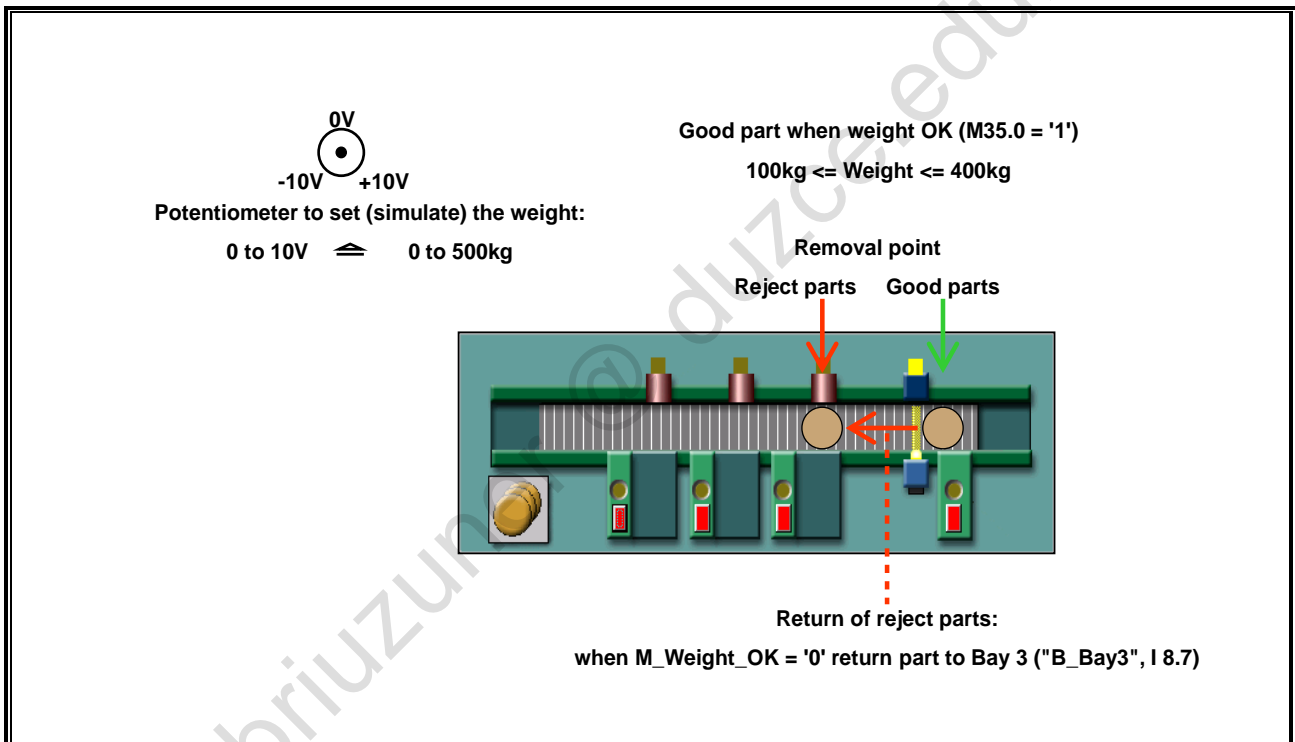
#### What to Do

1. Download all modified blocks into the CPU  
Right-click on PLC\_1 → Download to device → Software
2. On the touchpanel, check whether the weight value is displayed correctly and whether the background color changes when there is an invalid weight
3. Switch the operation on and enter a setpoint quantity
4. Set an invalid weight and check whether the bay indicator lights on the conveyor model remain dark and whether the conveyor motor can no longer be started

### 3.15. Additional Information



### 3.15.1. Additional exercise: Return of reject parts



#### Function up until now

Parts are transported from Bay 1 or 2 through the light barrier. A transport sequence is started 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 through the light barrier.

The acquisition of the weight (to be set on the potentiometer) of the transported parts is already programmed in "Cyclic\_interrupt". If the part weight is outside the allowed range of 100 to 400kg, the bit memory "M\_Weight\_OK" (M35.0) is assigned the status '0'.

#### Task

Parts whose weights lie outside of the allowable range are to be returned to Bay 3 ("B\_Bay3", I8.7). As well, these parts are not to be counted.

#### What to Do

1. Expand the block "FC\_Conveyor" to include the described return function
2. Expand the block "FC\_Count" in such a way that the reject parts are not counted
3. Save your project and download all blocks into the CPU

# Contents

<b>4.</b>	<b>Data Blocks.....</b>	<b>4-2</b>
4.1.	Objectives .....	4-2
4.2.	Data blocks (DBs) .....	4-3
4.3.	Overview of data types in S7-1200 .....	4-4
4.3.1.	Elementary data types for S7-1200 .....	4-5
4.3.2.	Data types for Timers, Date and Time-of-day.....	4-6
4.3.3.	Complex data types for S7-1200 .....	4-8
4.4.	Creating a data block .....	4-10
4.5.	Block access for DBs without the attribute "Optimized block access" .....	4-11
4.6.	Block access for DBs with the attribute "Optimized block access" .....	4-12
4.7.	Start value, monitor value, retain (retentivity) .....	4-13
4.8.	Editing and monitoring a data block.....	4-14
4.9.	Function for modifying tags in data blocks.....	4-15
4.10.	Retentivity in system FBs (1): Separate instance DBs .....	4-16
4.10.1.	Retentivity in system FBs (2): Storage in global DB .....	4-17
4.10.2.	Retentivity in system FBs (3): Multiple instance in the FB.....	4-18
4.11.	Accessing DB variables .....	4-19
4.12.	Task description: DB_Parts .....	4-20
4.12.1.	Exercise 1: Creating and declaring DB_Parts .....	4-21
4.12.2.	Exercise 2: Replacing bit memories with DB variables.....	4-22
4.12.3.	Exercise 3: Making the IEC-Counter retentive (Global DB).....	4-23
4.12.4.	Exercise 4: Transferring the modified program into the CPU and monitoring "DB_Parts" ..	4-24
4.12.5.	Exercise 5: Updating the HMI tag interfacing and transferring it to the Touchpanel .....	4-25
4.13.	Task Description: Archiving part weights in "DB_Parts" using "FieldWrite" .....	4-26
4.13.1.	Indirect addressing of Array elements with "FieldRead" and "FieldWrite" (1) .....	4-27
4.13.2.	Indirect addressing of Array elements with "FieldRead" and "FieldWrite" (2) .....	4-28
4.13.3.	Exercise 6: Creating a re-usable function "FC_Ind_Weight" and declaring the interface ..	4-29
4.13.4.	Exercise 7: Programming the DB access as re-usable using "FieldWrite".....	4-30
4.13.5.	Exercise 8: Calling the new function in "FC_Count" .....	4-31
4.13.6.	Exercise 9: Monitoring "DB_Parts" .....	4-32
4.14.	Additional Information .....	4-33
4.14.1.	Additional exercise: Reading back the setpoint (quantity) and adopting it as the start value	4-34
4.14.2.	Type conversion.....	4-35

## 4. Data Blocks

### 4.1. Objectives

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

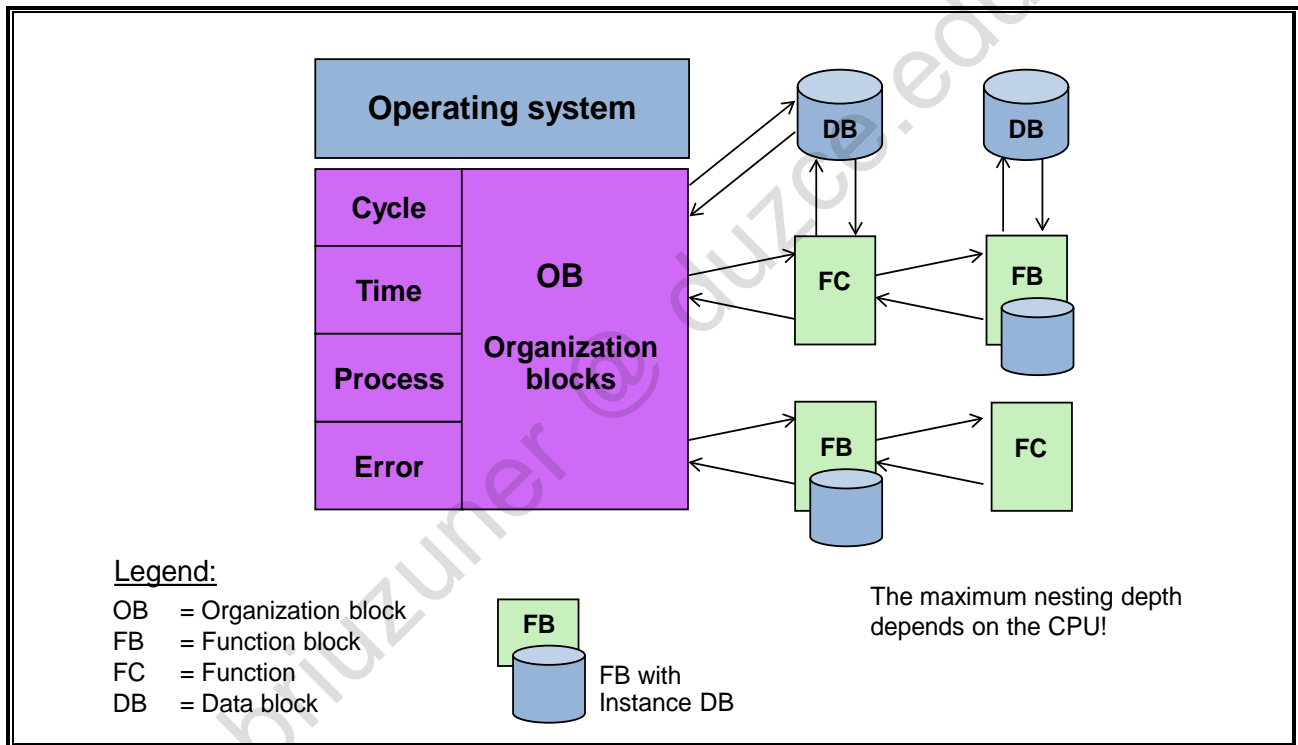
- ... understand the purpose of global data blocks
- ... be familiar with elementary and complex data types
- ... be able to monitor a data block
- ... be familiar with the possibilities for addressing data block variables
- ... understand the principle of retentiveness in data blocks
- ... be able to indirectly address Array elements in FBD/LAD

#### Objectives

In this chapter, the purpose of data blocks and the S7 data types are presented. The goal is that the participant can create a data block and declaring it.

In this context, the term retentiveness is also presented and the possibilities of using it.

## 4.2. Data blocks (DBs)



### Overview

Data blocks are used for storing user data and take up space in the user memory of the CPU. Data blocks contain variable data (for example, numeric values) with which the user program works.

The user program can access the data in a data block with bit, byte, word or double-word operations. The access can be either symbolic or absolute.

### Area of application

You can use data blocks in different ways, depending on their contents. You differentiate between:

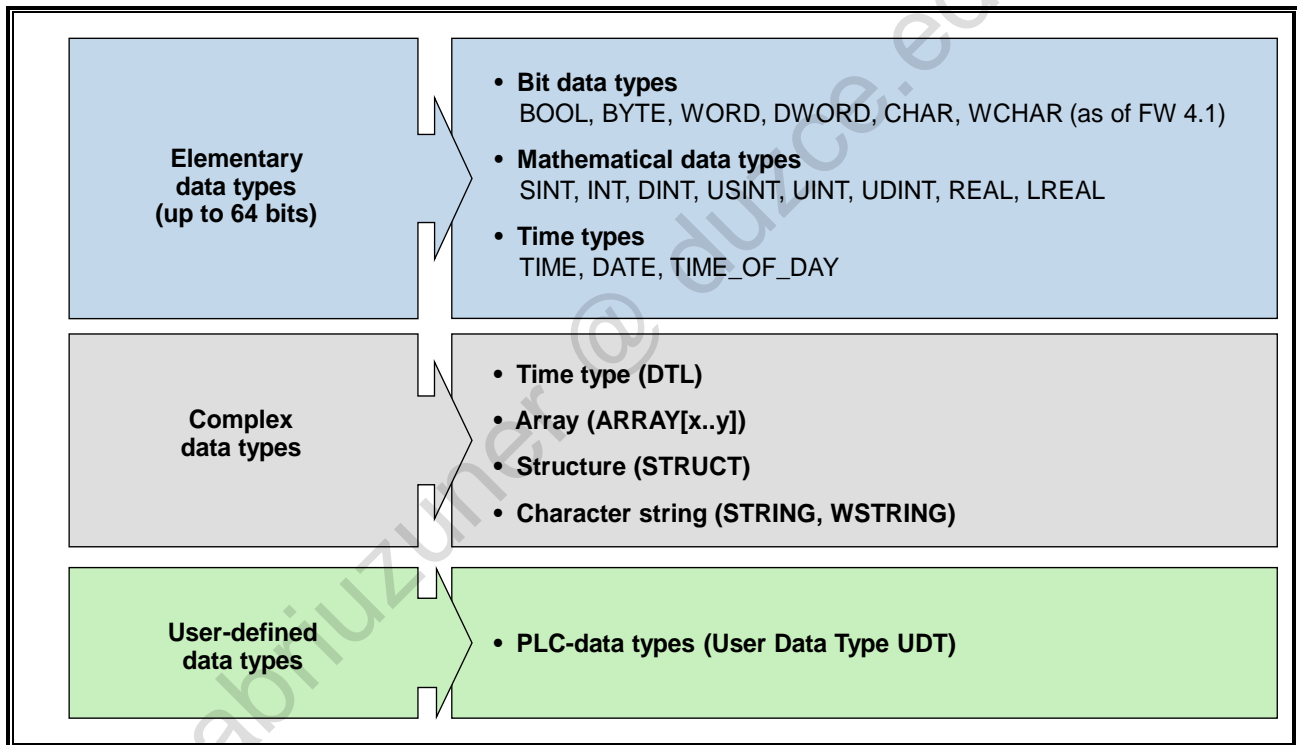
- **Global data blocks:** These contain information that all the logic (code) blocks in the user program can access.
- **Instance data blocks:** These are always assigned to a particular FB. The data of these instance DBs should only be processed by the associated FB.

### Creating DBs

Global DBs are created either with the Program Editor or according to a previously created "user-defined data type" (UDT).

Instance data blocks are generated when a function block is called.

### 4.3. Overview of data types in S7-1200



#### Overview

Variables are used to store data. The data type of a variable determines the amount of memory space it requires, its value range and the presentation of the variable value in the editor. Furthermore, the possible operations with which a variable can be processed arise from the data type.

#### Elementary data types

Elementary data types are predefined in accordance with IEC 61131-3. They always have a length less than or equal to 64 bits.

#### Complex data types

Complex data types contain data structures that can be made up of elementary and/or complex data types.

Complex data types can be used for the declaration of variables only in global data blocks and within blocks for the declaration of local variables (TEMP, STAT) as well as parameters (IN, OUT and INOUT).

Variables of complex data types cannot be processed completely with instructions (such as comparator), but only the components of elementary data types.

#### User-defined data types (UDT)

UDTs are templates for declaring variables of complex data types or structure variables. UDTs are created with the Data Block Editor and contain a data structure that is made up of elementary and/or complex data types. In the declaration of a variable according to data type UDTx, a structure variable is created whose inner data structure is defined by UDTx.

UDTs can be used for the declaration of variables in global data blocks and within blocks for the declaration of local variables (TEMP, STAT) as well as parameters (IN, OUT and INOUT).



### 4.3.1. Elementary data types for S7-1200

Data type	Length (in bits)	Constants	Variables
BOOL	1	1 or 0	I 1.0
BYTE	8	B#16#A9	MB70
WORD	16	W#16#12AF	MW72
DWORD	32	DW#16#ADAC1EF5	QD40
CHAR	8	'w'	DBB4
WCHAR <sub>(as of FW 4.1)</sub>	16	WCHAR#'a'	MW50
INT	16	123	#Value
DINT	32	L#65539	MD80
REAL	32	1.2 or 34.5E-12	DBD60
SINT	8	+/-50	MB24
USINT	8	50	MB24
UINT	16	12654	IW22
UDINT	32	4875678	DBD64
LREAL	64	LREAL#1.0e-5	

#### BOOL, BYTE, WORD, DWORD, CHAR, WCHAR

Variables of the data type BOOL consist of one bit. Variables of the data types BYTE, WORD, DWORD are bit sequences of 8, 16 or 32 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 as well as the data type CHAR which represents a character in the ASCII code and as of FW 4.1 WCHAR for representing characters in the extended character set (Format Unicode).

#### INT, DINT, REAL

Variables with these data types represent numbers with which relevant mathematical operations can be carried out.

#### Extensions of INT, DINT, REAL

- U – Unsigned  
Variables with the extension U represent an integer without sign.  
Data types: USINT, UINT, UDINT
- S – Short  
Variables with the extension S represent an integer with a length of 8 bits.  
Data types: SINT, USINT
- L – Long  
Variables with the extension L represent a number with a length of 64 bits of the data type.  
Data types: LREAL

### 4.3.2. Data types for Timers, Date and Time-of-day

Data type	Length (in bits)	Example
<b>Timers</b>		
TIME	32	T#2h46m30s630ms
<b>Date and Time-of-day</b>		
DATE	32	D#1984-01-01
TIME_OF_DAY (TOD)	32	TOD#18:15:18:999
DTL	96	DTL#1984-01-01-18:00:30:250000000 (see next picture)

#### TIME

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#1 000, T#24d20h31m23s647ms = L#2147486470).

#### 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 (for example: D#2003-10-31 = W#16#13BB).

#### Time\_Of\_Day

The data type TIME\_OF\_DAY (TOD) occupies a double word and stores the number of milliseconds since the beginning of the day (0:00 o'clock) as an unsigned integer.

## 4.3.2.1. Complex data type: DTL

Static			
▼ Date_Time	DTL	DTL#1970-01-01-00:00:00	
YEAR	UInt	1970	
MONTH	USInt	1	
DAY	USInt	1	
WEEKDAY	USInt	5	
HOUR	USInt	0	
MINUTE	USInt	0	
SECOND	USInt	0	
NANOSECOND	UDInt	0	

**Data type**

↓

**DTL**

↓

The data type DTL represents a point in time which consists of information on date and time-of-day. The individual components can be accessed directly.

↓

#Date\_Time.HOUR

↓

Hour

12

## DTL

The data type DTL has a length of 12 bytes and stores information on date and time-of-day precise to the nanosecond since 1.1.1970 in a pre-defined structure.

The structure of the data type DTL is made up of several components which, in each case, can have a different data type and value range. The data type of a specified value must be compatible with the data type of the respective component.

The following table shows the structure components of the data type DTL and their properties:

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

Advantage: the individual values (day, hour, etc.) are easier to read out.

### 4.3.3. Complex data types for S7-1200

Data type	Length (in bits)	Example
<b>STRING</b> (character string with max. 254 characters)	8 * (number of characters +2)	'This is a String' 'SIEMENS'
<b>WSTRING</b> (as of FW 4.1) (character string with max. 254 characters)	16 * (number of characters +2)	WSTRING# 'A String in the extended format UNICODE'
<b>ARRAY</b> (Group of components of the same data type)	User-defined	Measured values: ARRAY[1..20] of INT
<b>STRUCT</b> (Structure, Group of components of different data types)	User-defined	Motor: STRUCT Speed      : INT Current   : REAL END_STRUCT
<b>PLC-data type (UDT)</b> (User Defined Data Type) "Template" consisting of elementary or complex data types	User-defined	UDT as block            UDT as array element
		STRUCT Speed : INT Current: REAL END_STRUCT

#### Complex Data Types

Complex data types (arrays and structures) consist of groups of elementary or complex data types.

They enable you to create data types suitable for your problem with which you can structure large quantities of data and process it symbolically.

Complex data types cannot be processed directly with STEP 7 instructions all at once. Only one component at a time can be processed.

The lengths of complex data types are defined by the user.

Variables with complex data types can only be declared within global data blocks and as parameters or local variables of logic (code) blocks.

#### PLC-Data Type/User-defined Data Type (UDT)

User-defined data types represent self-defined structures. This structure is stored in UDT blocks and can be used as a "template" in another variable's data type. You can save typing time when you input a data block if you need the same structure several times.

#### Example

You need the same structure 10 times in a data block. First, you define the structure and save it, for example, as Userdatatype\_1.

In the DB, you then define a variable "Addresses" as an array with 10 elements of the type Userdatatype\_1:

```
Addresses: array[1..10] Userdatatype_1
```

That way, you have created 10 data ranges with the structure that is defined in Userdatatype\_1 without "typing".

## 4.3.3.1. Array, STRUCT, PLC-datatypes

DB_Parts							
	Name	Data type	Start value	Retain	Accessible from HMI/OPC UA	Writable from HMI/OPC UA	Visible in HMI engineering
1	Static			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	PartCounter	IEC_COUNTER		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	WeightStore	Struct		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	setpointNo	Int	5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	actualNo	Int	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	PartWeights	Array[1..10] of Int		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
7	PartWeights[1]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
8	PartWeights[2]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
9	PartWeights[3]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
10	PartWeights[4]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
11	PartWeights[5]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
12	PartWeights[6]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
13	PartWeights[7]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
14	PartWeights[8]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
15	PartWeights[9]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
16	PartWeights[10]	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
17	motor1	*drive*		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
18	start	Bool	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
19	stop	Bool	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
20	setSpeed	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
21	actSpeed	Int	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
22	temperature	Real	0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Array**

Array (or field) represents a data structure that consists of a fixed number of components of the same data type.

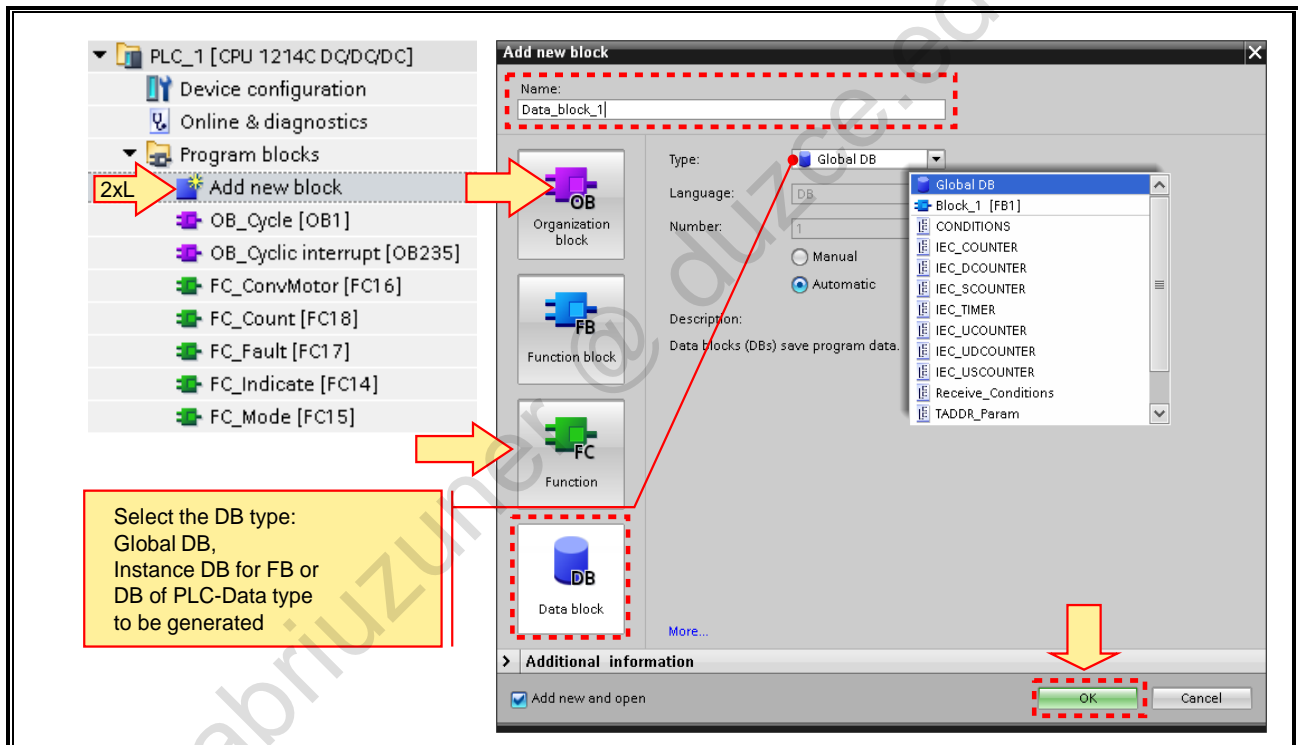
**STRUCT**

STRUCT (or structure) represents a data structure that consists of a fixed number of components of different data types. The structure must be reassembled at each point of use. If a structure is to be used more than once, it is advantageous to create a PLC data type.

**PLC-datatype**

User-defined data types that serve as templates for declaring parameters and variables of complex data types (e.g. structure variables). UDTs are created with the data block editor and contain a data structure consisting of elementary and/or complex data types. When a variable is declared according to data type UDTx, a structured variable is created whose internal data structure is defined by UDTx. UDTs can be used to declare variables in global data blocks and within blocks to declare local variables (TEMP, STAT) and parameters (IN, OUT and INOUT).

## 4.4. Creating a data block



### Creating a DB

A new data block can be inserted – as shown in the picture.

You can create a new data block in the Portal view as well as in the Project view of the respective project.

### Global DB

Global data blocks are used to store global data, that is, to store general data which can be accessed by every logic (code) block (OB, FC, FB).

You must edit global data blocks yourself by declaring the necessary variables for storing the data in the data block.

### Instance DB

Instance data blocks serve as "private memory area" or as "memory" for a function block (FB). In the instance DB of an FB, its parameters and static variables are managed.

Instance data blocks are never edited by you, rather are generated by the Editor.

### DB of Type

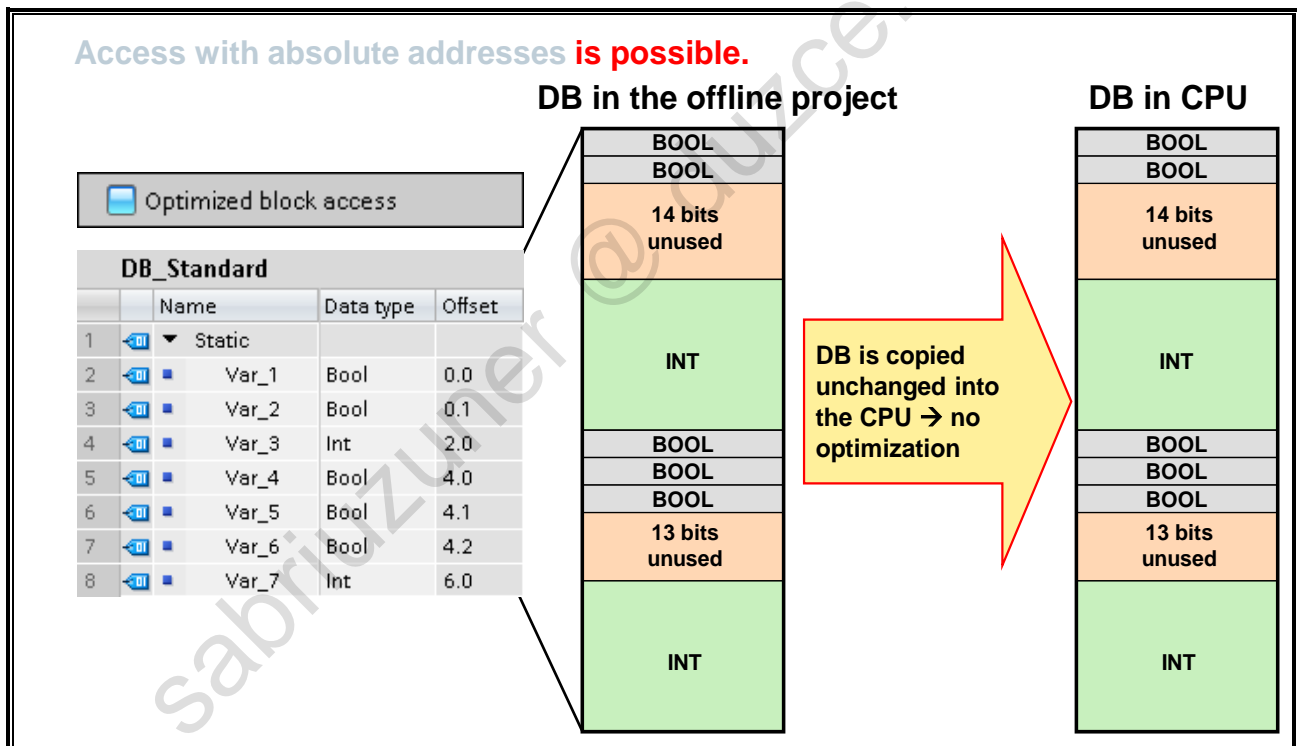
Data blocks can also be generated according to a PLC-data type by the Editor. For this the PLC-data type, which you must first edit just like a data block, is used as a template.

The PLC-data type can be used as a template to create further data blocks and / or also in general to declare variables and block parameters.

### Type of Access

The type of access is defined as "Optimized block access" when the block is created. It is possible to change this later through the Properties of the block in "Attributes".

#### 4.5. Block access for DBs without the attribute "Optimized block access"



#### Data Block without the Attribute "Optimized block access"

When a data block does not have the attribute "Optimized block access", it has the following properties:

- Variable addressing

Data blocks with standard access have a fixed structure. In the declaration, the data elements contain a symbolic name as well as a fixed address within the block. The address is displayed in the column "Offset".

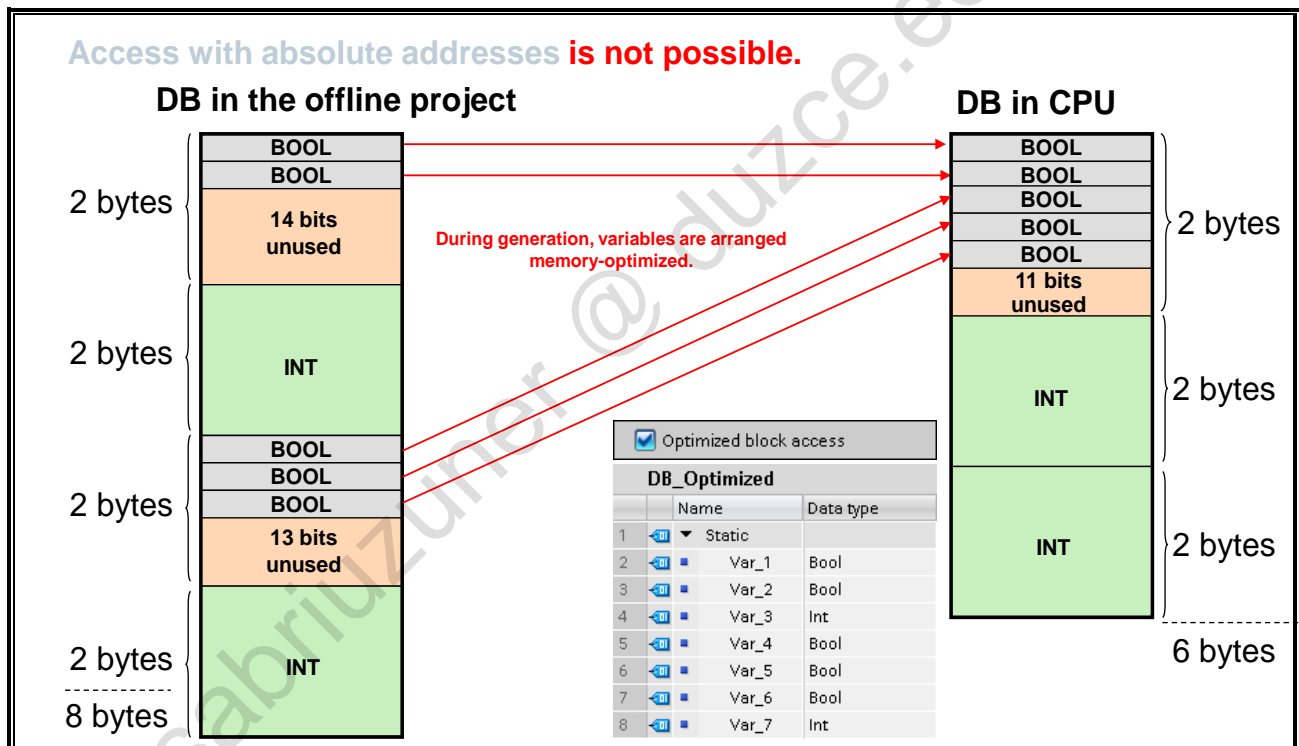
- Retentivity

Only the entire DB can be defined as retentive or non-retentive → poorer utilization of the retentive memory of the CPU

- Use (for example)

- "PUT" and "GET" calls  
So that the DBs in an S7-1200 can be addressed by a remote S7 controller (such as, S7-300) with the blocks PUT/GET, the DB variables must be addressed absolutely.
- HMI with SIMATIC WinCC flexible  
When WinCC flexible is used for the configuration of (older) HMI devices (such as, TP170B), DB accesses are always absolute.
- T\_Communication  
Even for the configured/programmed open T\_Communication (between S7-1200 and S7-300/400), DB accesses (for example, to Send/Receive buffers) must be made absolutely.

## 4.6. Block access for DBs with the attribute "Optimized block access"



### Data block with the attribute "Optimized block access"

With the attribute "Optimized block access", you can optimally store the variables in the DB.

### Variable Addressing

Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are only given a symbolic name, no fixed address within the block. The elements are automatically arranged in the available memory area of the block in such a way that its capacity is optimally exploited.

Variables in these data blocks can only be addressed symbolically.

### Retentivity

The retentivity can be selected separately for each DB variable → optimum utilization of the retentive memory of the CPU.

### Advantages

The optimized access provides the following advantages:

- The data is structured and stored in a way that is optimum for the CPU used. In that way, you increase the performance of the CPU
- Access errors, for example, from the HMI, are not possible
- You can specifically define individual variables as retentive



## 4.7. Start value, monitor value, retain (retentivity)

If the DB's attribute "Optimized block access" is activated, the retentivity for all variables can be selected separately. If the attribute is not activated, either all variables are retentive or none are.

### Retain

To prevent data loss when there is a power loss, you can identify certain data as retentive. These are stored in a retentive memory area. A retentive memory area is an area whose contents are retained after a restart (warm restart), that is, after switching off the supply voltage and after switching on during a transition from STOP to RUN.

You can define the following data as retentive:

- Bit memories:  
You can define the exact length of the retentive memory area for bit memories in the PLC tag table or in the assignment list.
- Variables of a function block (FB):  
You can define individual variables as retentive in the interface of an FB if the symbolic addressing of the variable is activated for this block. If the symbolic addressing for an FB is not activated, retentive settings can only be made in the associated instance data block.
- Variables of a global data block:  
In a global data block, you can define either individual or all variables of the block as retentive depending on the setting of the symbolic addressing.

## 4.8. Editing and monitoring a data block

The screenshot shows the 'DB\_Parts' data block configuration in SIMATIC Manager. The table below represents the data shown in the interface:

Name	Data type	Start value	Retain	Accessible from HMI	Visible in HMI
Static					
Parts_Counter	IEC_COUNTER		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CU	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CD	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
R	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
LD	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
QU	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
QD	Bool	false	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
PV	Int	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CV	Int	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Parts_Management	Struct		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Setpoint	USInt	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Actual	USInt	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Part_Weight	Array[1..100] of Int		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Part_Weight[1]			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Part_Weight[2]			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Part_Weight[3]			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Part_Weight[4]			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Annotations in the image explain the following features:

- Add new line:** A button in the top toolbar used to add new rows to the data block.
- Monitor DB:** A button in the top toolbar used to monitor the data block.
- Visible in HMI:** A column of checkboxes that, when checked, indicates the variable is displayed in the WinCC tag selection dialog.
- Retain:** A column of checkboxes that, when checked, indicates the variable is retentive. For optimized block access, it can be individually changed.
- Start value:** A column for defining the start value. The value is used after the first STOP-RUN when the variable is retentive, or with each STOP-RUN when the variable is not retentive.
- Structures and Arrays can be minimized → better overview:** A note indicating that structures and arrays can be minimized in the table for a better overview.

### Editing a data block

The settings shown in the picture can be made for the individual DB variables.

### Start value

Value which the variable is to adopt during every startup when the variable is not retentive.  
Value which the variable is to adopt only during the first startup when the variable is retentive.

Start value for FBs for static variables:

In creating a data block, the default values defined in the FB are used as start values. These adopted values can be replaced here with instance-specific start values.

The specification of a start value is optional. If no value is defined, the variable adopts the default value during startup. If a default value is also not defined, the standard value valid for the data type is used. For BOOL, for example, the standard value "FALSE" is defined.

### Monitor values

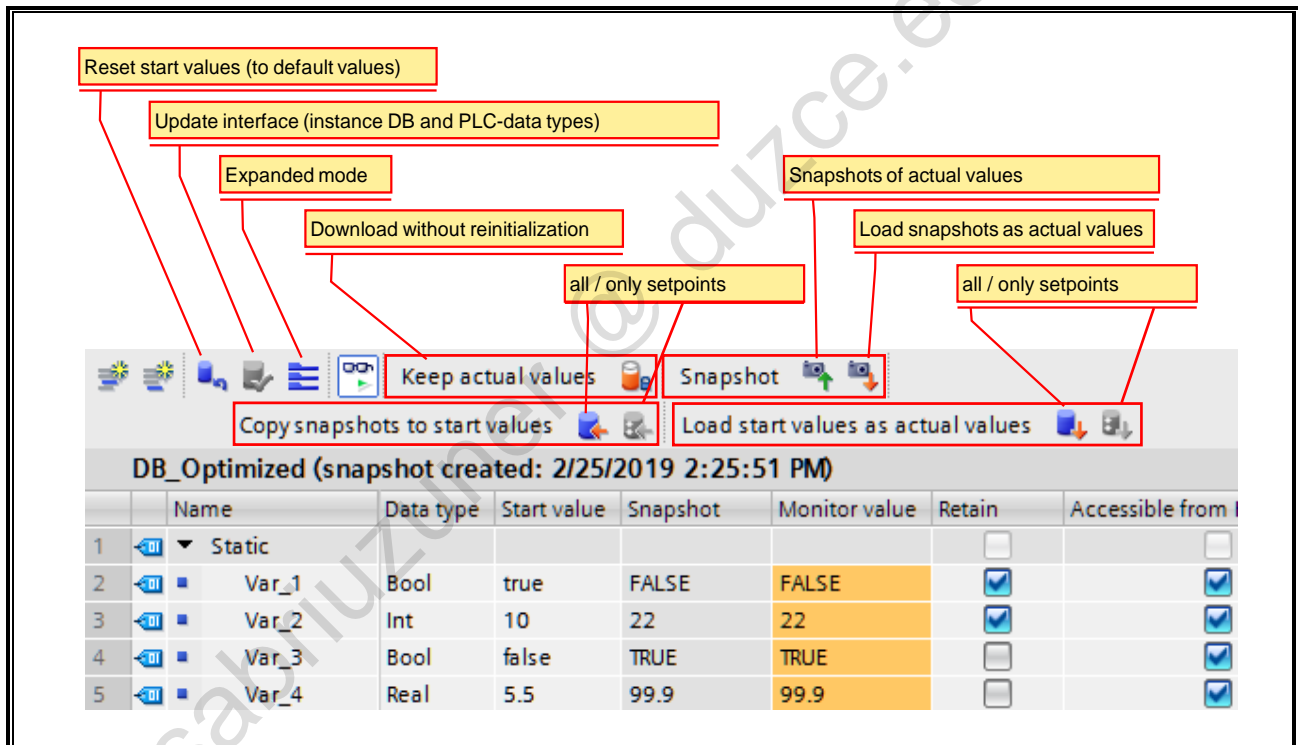
Current data value in the CPU.

The column appears when an online connection exists, and you select the button "Monitor".

### Snapshot

Displays values which were loaded from the device at a specific point in time.

## 4.9. Function for modifying tags in data blocks





### Securing a snapshot

Under certain circumstances it is necessary to read back the monitor values of a data block from the controller.

With the "Snapshot of actual values" function, the current monitor values of the online DB are stored in your offline DB.

With the "load snapshot as actual values", the saved snapshot is put back into the block (actual value).

### Copy snapshot to start values

Then, the secured values can be adopted into the start value column using the  button (for all snapshots highlighted in the Setpoint column) or the  button (for all snapshots), whereby the secured values are adopted as start values the next time the DB is transferred.

### Load start values as actual values

The monitored values (actual values) can be reset (initialized) to their start value with this function. The function is available for all variables or only for the setpoint values (marked in the setpoint column).

### Keep actual values

With the function (Activate memory reserve) blocks can be extended without existing monitor values (actual values) being lost. The extensions are inserted into a previously created memory reserve.

## 4.10. Retentivity in system FBs (1): Separate instance DBs

**Retentivity:** Either all variables are retentive or none are

**Instance DBs of system FBs are stored in the Program resources folder → User program remains uncluttered**

	Name	Data type	Start value	Retain
1	Static			<input type="checkbox"/>
2	CU	Bool	false	<input type="checkbox"/>
3	CD	Bool	false	<input type="checkbox"/>
4	R	Bool	false	<input type="checkbox"/>
5	LD	Bool	false	<input type="checkbox"/>
6	QU	Bool	false	<input type="checkbox"/>
7	QD	Bool	false	<input type="checkbox"/>
8	PV	Int	0	<input type="checkbox"/>
9	CV	Int	0	<input type="checkbox"/>

### Retentivity

In function blocks which are provided by the system, such as:

- Counters
- Timers
- communication etc.

the instance data can be declared retentive (all or none).

### Storage

Instance DBs of system FBs are stored in the program resources folder. That way, the user program remains uncluttered.

### 4.10.1. Retentivity in system FBs (2): Storage in global DB

DB_Optimized		Optimized Global DB		
	Name	Data type	Start value	Retain
1	Static			
2	VAR_1	Bool	true	<input type="checkbox"/>
3	VAR_2	Int	10	<input checked="" type="checkbox"/>
4	VAR_3	Bool	false	<input checked="" type="checkbox"/>
5	VAR_4	Real	5.5	<input type="checkbox"/>
6	Reading			<input checked="" type="checkbox"/>
7	VAR_5			<input type="checkbox"/>
8	VAR_6	USInt	0	<input type="checkbox"/>
9	Counter_1	IEC_COUNTER		<input type="checkbox"/>
10	Counter_2	IEC_COUNTER		<input checked="" type="checkbox"/>
11	CU	Bool	false	<input checked="" type="checkbox"/>
12	CD	Bool	false	<input checked="" type="checkbox"/>
13	R	Bool	false	<input checked="" type="checkbox"/>
14	LD	Bool		<input type="checkbox"/>
15	QU	Bool		<input type="checkbox"/>
16	QD	Bool		<input type="checkbox"/>
17	PV	Int	0	<input checked="" type="checkbox"/>
18	CV	Int	0	<input checked="" type="checkbox"/>

#### Storage in global DBs

The data from function blocks provided by the system can be combined in a higher-level global data module. The retentivity can be set there, as for all other variables. The variables to be defined are of type:

- IEC\_Counter
- IEC\_Timer etc.

#### 4.10.2. Retentivity in system FBs (3): Multiple instance in the FB

Instance data of the counter is created as static variable of the calling FB.

With optimized block call, retentivity can be set separately for every variable of the interface.

	Name	Data type	Default value	Retain
1	Input			
2	Output			
3	InOut			
4	Static			
5	Counter_1	IEC_COUNTER		Non-retain
6	Temp			
7	Constant			

Call options dialog: Multiple instance, Name in the interface: Counter\_1

Basic instructions: Counter operations (CTU, CTD, CTUD)

Network 1: #Counter\_1, CTD, Int, LD, Q, CV, PV

Annotations: Query, Drag & Drop

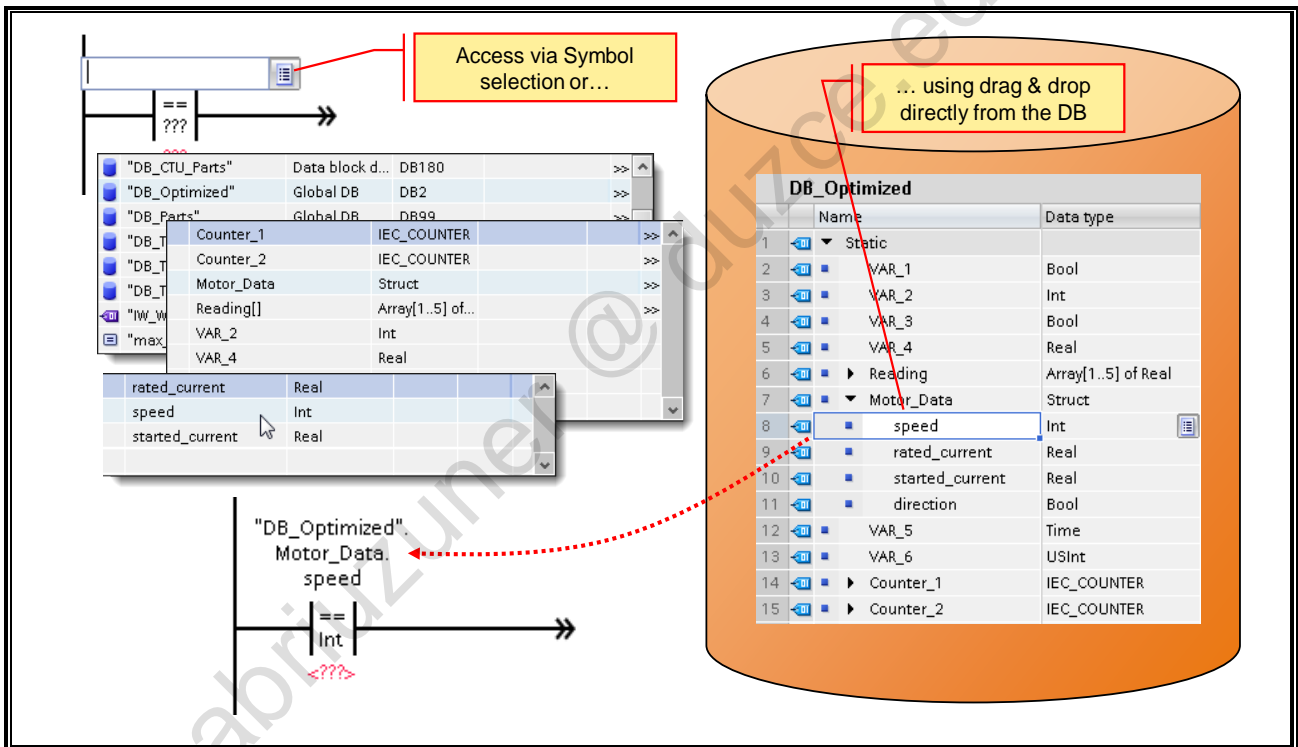
#### Multiple instance in the FB

Like the global DB, static variables, for example, of the type IEC\_COUNTER can also be created in the interface of an FB. If such a variable is assigned a corresponding block call (for example, a counter) in the instruction part, you speak of a multi instance.

The retentivity of the variables in an FB can be set or not set under certain conditions:

- In the higher-level function block, the optimized block access is activated:  
For parameters and static variables, the property "Retain" can be individually activated or deactivated.
- In the higher-level function block, the standard block access is activated:  
In the interface of the FB, the property "Retain" cannot be activated.  
In the associated instance DB, the property "Retain" can only be activated or deactivated for the entire DB.

### 4.11. Accessing DB variables



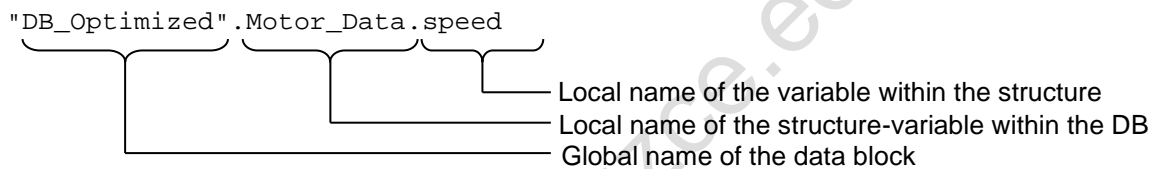
#### Accessing DB variables

Basically, there are two ways of accessing a DB variable:

- Access via Symbol Selection

With the symbolic selection of a DB variable, the individual elements of the DB are accessed via dot notation. The same holds true for variables of complex data types.

An access could look something like this:

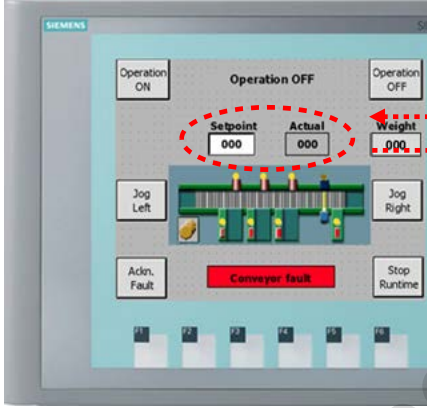


- Access via Drag & Drop

For this, a DB variable is clicked with the mouse and held and dragged to the appropriate position in the user program and there let go.

This function is particularly practical when you divide the Editor area in such a way that the DB is displayed next to the logic (code) block.

## 4.12. Task description: DB\_Parts



The screenshot shows a SIMATIC Manager interface with a conveyor control panel on the left and a data block configuration table on the right. The control panel includes buttons for 'Operation ON', 'Operation OFF', 'Jog Left', 'Jog Right', 'Ackn. Fault', and 'Stop Runtime'. It also displays 'Setpoint' and 'Actual' values (both 000) and a 'Weight' value (000). A red dashed circle highlights the 'Setpoint', 'Actual', and 'Weight' fields. Red arrows point from these fields to the corresponding entries in the 'DB\_Parts' table.

DB_Parts				
	Name	Data type	Start value	Retain
1	Static			<input type="checkbox"/>
2	Parts_Counter	IEC_COUNTER		<input checked="" type="checkbox"/>
3	Parts_Management	Struct		<input type="checkbox"/>
4	Setpoint	Int	5	<input type="checkbox"/>
5	Actual	Int	0	<input type="checkbox"/>
6	Part_Weight	Array[1..100] of Int		<input type="checkbox"/>
7	Part_Weight[1]	Int	0	<input type="checkbox"/>
8	Part_Weight[2]	Int	0	<input type="checkbox"/>
9	Part_Weight[3]	Int	0	<input type="checkbox"/>
10	Part_Weight[4]	Int	0	<input type="checkbox"/>
11	Part_Weight[5]	Int	0	<input type="checkbox"/>
12	Part_Weight[6]	Int	0	<input type="checkbox"/>
13	Part_Weight[7]	Int	0	<input type="checkbox"/>
14	Part_Weight[8]	Int	0	<input type="checkbox"/>
15	Part_Weight[9]	Int	0	<input type="checkbox"/>
16	Part_Weight[10]	Int	0	<input type="checkbox"/>
17	Part_Weight[11]	Int	0	<input type="checkbox"/>

### Situation up until now

To manage the setpoint and actual quantities, the memory variables "MW\_SET" (MW22) and "MW\_ACT" (MW20) are used. These are linked with the corresponding I/O fields on the touchpanel.

### Task description

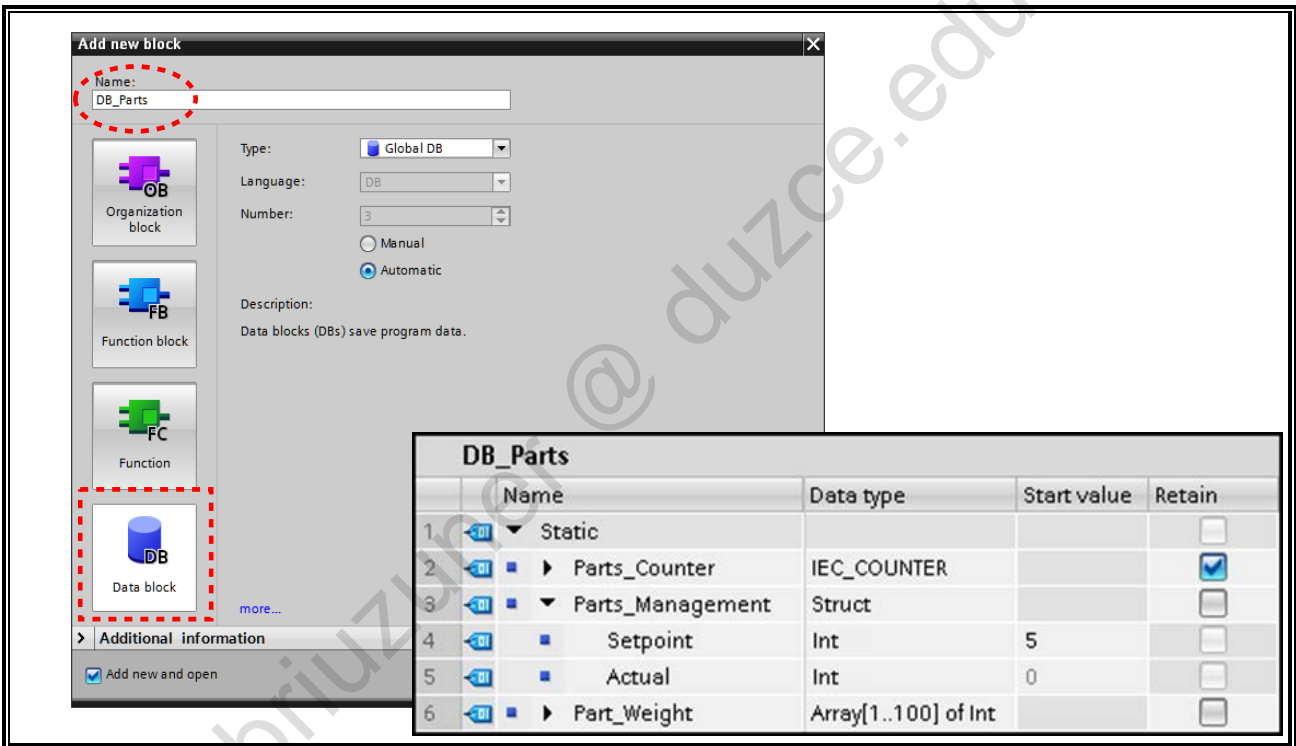
For the later management of part weights, a data block "DB\_Parts" is to be created. For this, a Structure variable "WeightStore" and an Array variable "PartWeights" are created. In "WeightStore", the elements "SetpointNo" and "ActualNo" replace the present memory variables "MW\_SET" (MW22) and "MW\_ACT" (MW20) in "FC\_Count".

The weight values of transported parts will be stored in the Array variable "PartWeights" using indirect addressing.

Furthermore, a variable of the type "IEC\_COUNTER" is to be created in this data block. The instance of the IEC\_COUNTER in "FC\_Count" should be replaced by the new variable. The variable is to be made retentive.



### 4.12.1. Exercise 1: Creating and declaring DB\_Parts



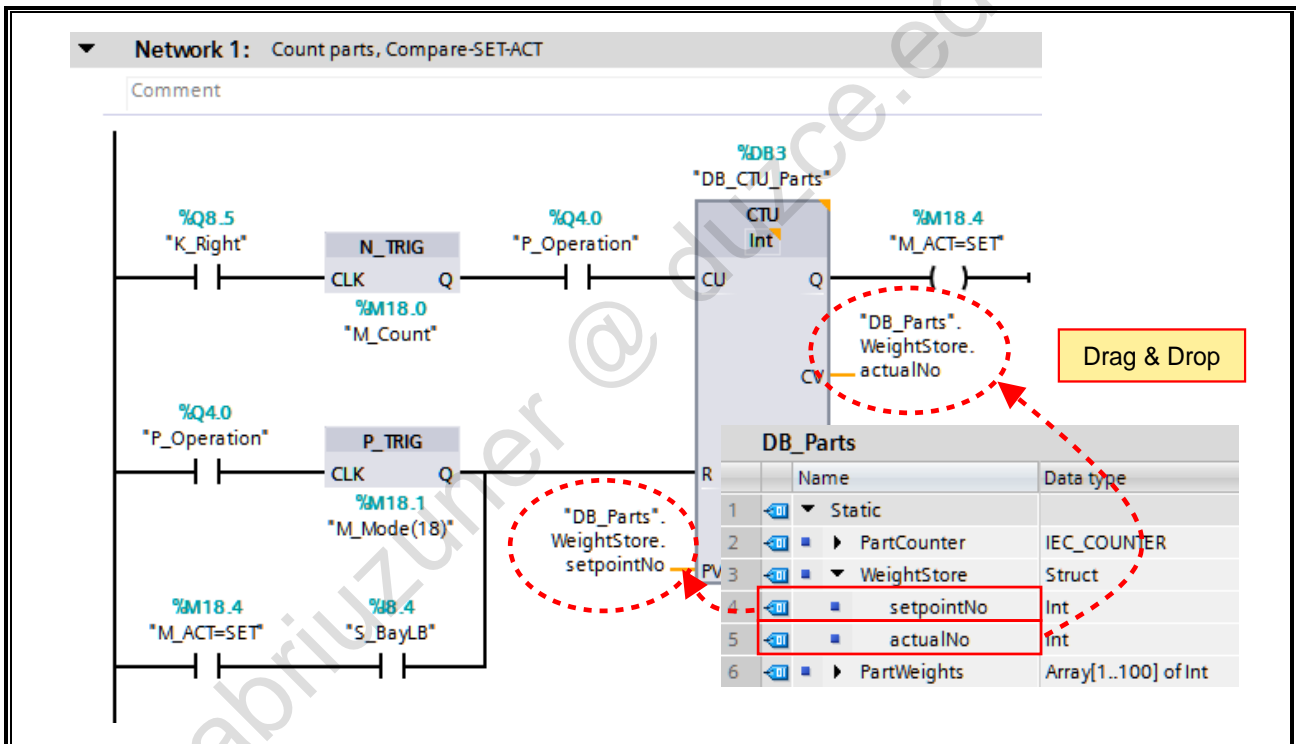
**Task**

You are to create and declare a data block "DB\_Parts".

**What to do**

1. Create a new data block with optimized access as well as the settings shown above
2. Declare the DB variables shown in the picture

#### 4.12.2. Exercise 2: Replacing bit memories with DB variables



#### Task

In "FC\_Count", replace the memory variables "MW\_SET" and "MW\_ACT" with the previously created DB variables "WeightStore.setpointNo" and "WeightStore.actualNo".

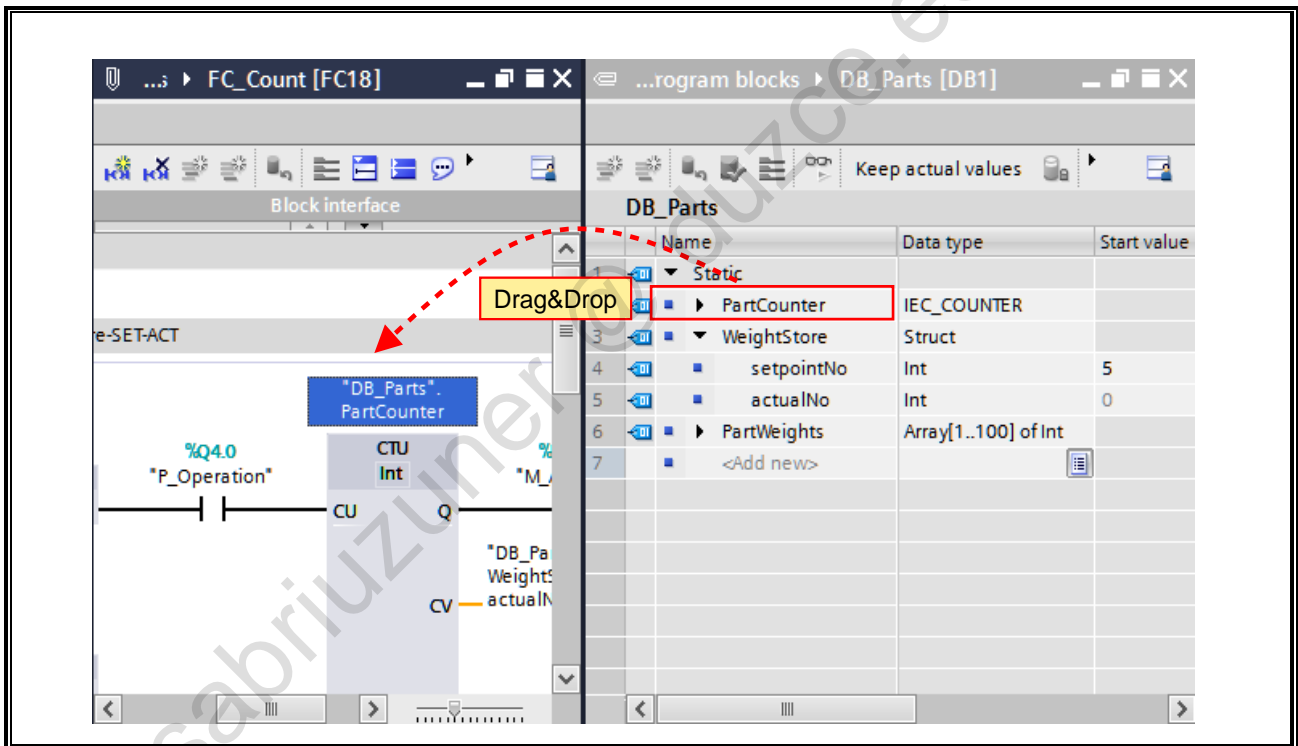
#### What to Do

1. Open data block "DB\_Parts"
2. Now open the function "FC\_Count"
3. Display the editor area split either horizontally or vertically  
click or
4. Using drag & drop, drag the DB variable "WeightStore.setpointNo" onto the parameter "PV" of the counter in "FC\_Count"
5. Using drag & drop, drag the DB variable "WeightStore.actualNo" onto the parameter "CV" of the counter in "FC\_Count"
6. Save the changes you made



#### Result

Setpoint number and Actual number are now managed in DB\_Parts. The Setpoint has the start value "5".

### 4.12.3. Exercise 3: Making the IEC-Counter retentive (Global DB)



#### What to do

1. Open data block "DB\_Parts"
2. Now open the function "FC\_Count"
3. Display the Editor area split either horizontally or vertically.  
click  or 
4. Using drag & drop, drag the DB variable "Part\_Counter" onto the IEC\_COUNTER in "FC\_Count"

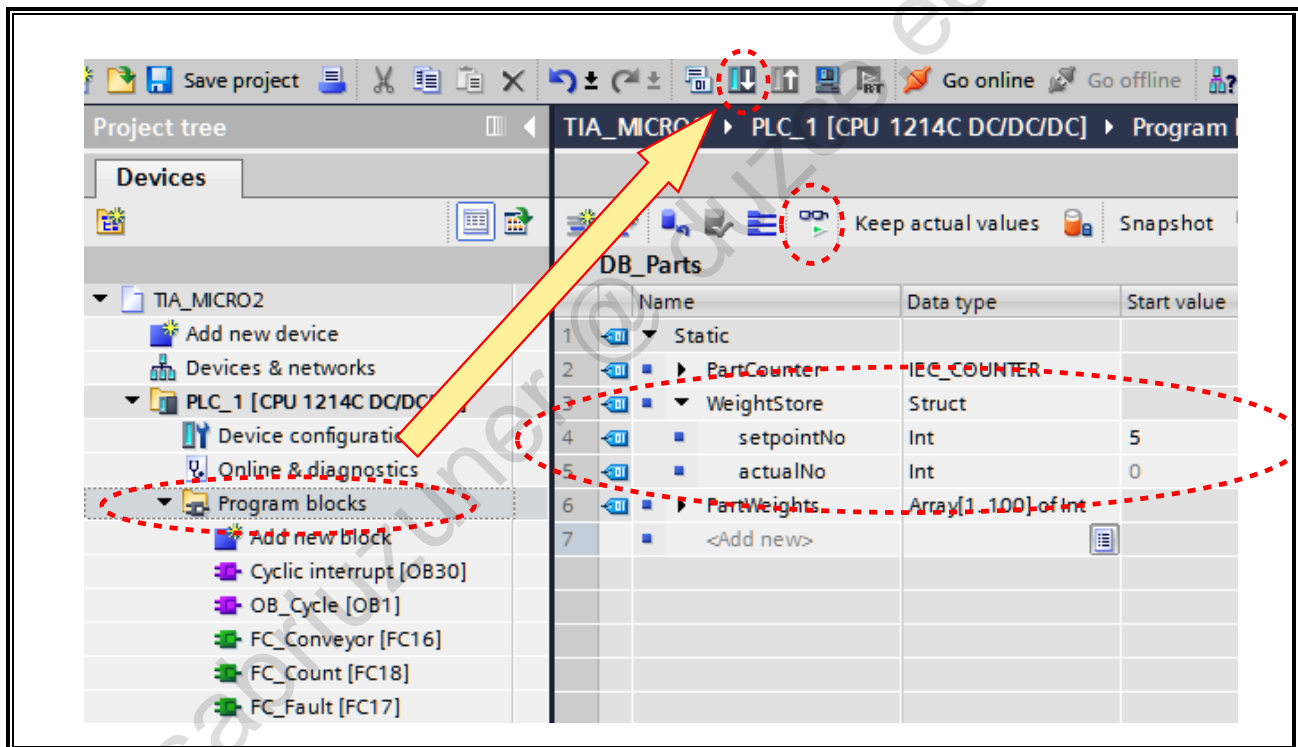
#### Result

The IEC\_COUNTER retentively manages its instance data in "DB\_Parts". That way, the current count value exists even after a CPU (warm) restart.



You can also achieve the same behavior by activating the retentivity of the previously used instance DB.

#### 4.12.4. Exercise 4: Transferring the modified program into the CPU and monitoring "DB\_Parts"



#### Task

Save the changes you made previously and then transfer the entire user program into the CPU.

#### What to do

1. Save your project

click Save project

2. Transfer the user program into the CPU

Select the Program blocks folder → click to start the transfer.

3. Monitor the Setpoint and Actual of "DB\_Parts".

4. Test the retentiveness of the counter by producing at least one part and then restarting the

CPU with, for example, the buttons and .

5. Save your project

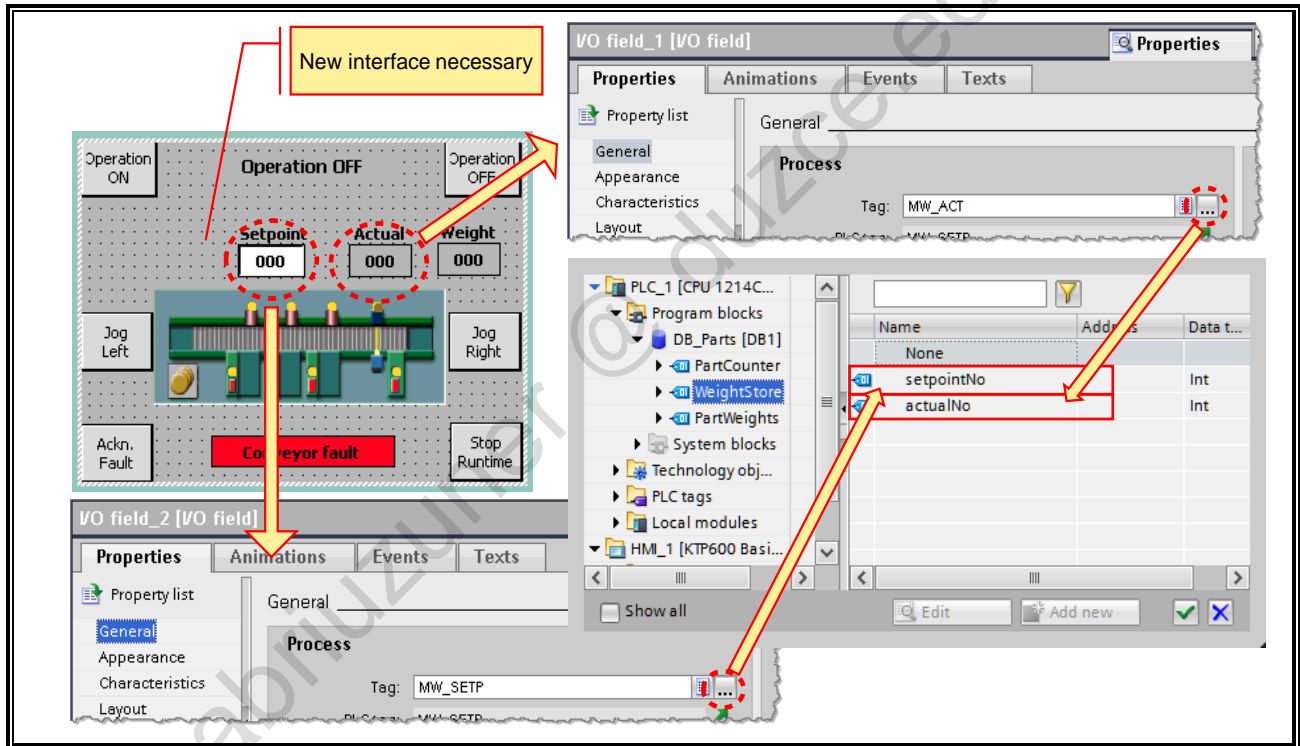
#### Result

The count (ActualNo) is retained even after a CPU (warm) restart.

#### What Doesn't Work Yet

At the moment, the I/O fields on the touchpanel are still linked with the old memory variables and therefore do not show the correct values. This behavior will be corrected in the next exercise.

#### 4.12.5. Exercise 5: Updating the HMI tag interfacing and transferring it to the Touchpanel



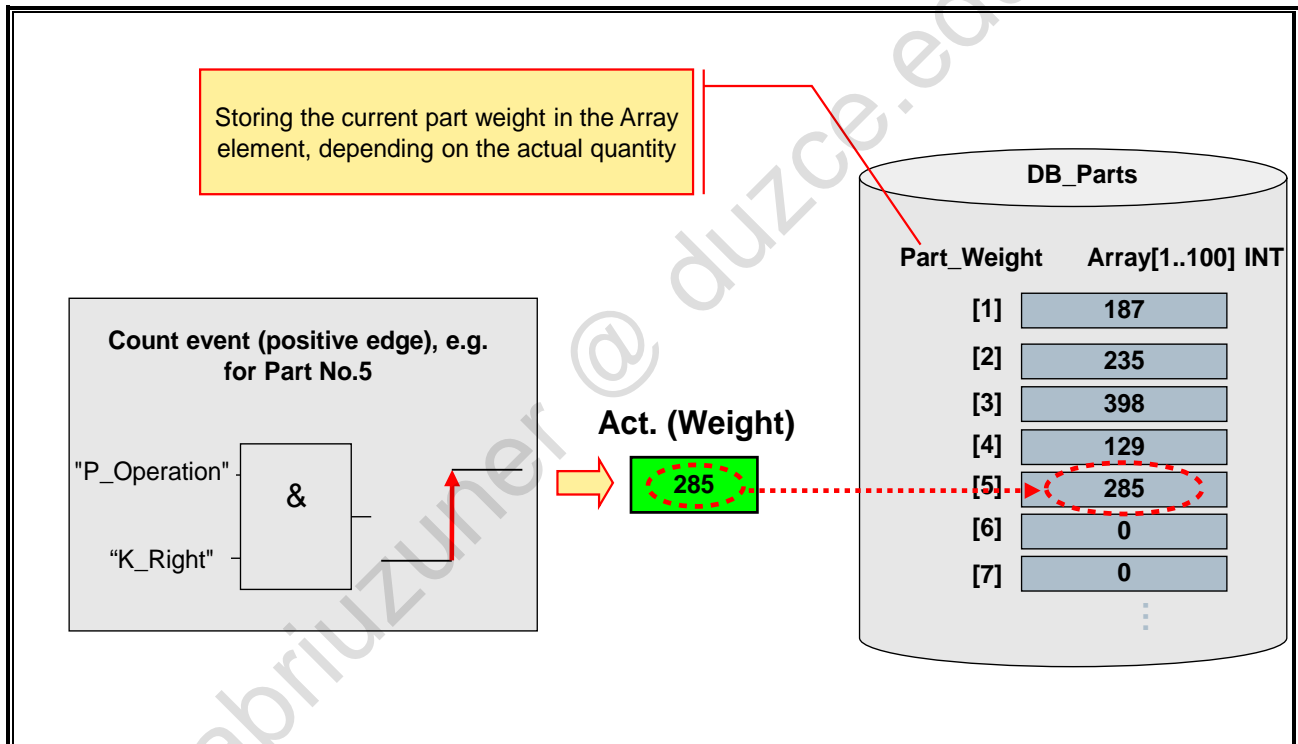
#### Task

To adjust the HMI project to the previously modified PLC program, the HMI tags used must be linked to the data block variables "DB\_Parts".WeightStore.actualNo and "DB\_Parts".WeightStore.setpointNo used in the PLC program.

#### What to do

1. Open the HMI project and there the screen "Conveyor"  
My\_Project → HMI\_1 → Screens → Conveyor
2. Open the General Properties of the I/O field "Setpoint"  
Click on I/O field → Inspector window → Properties → General
3. As shown in the picture, exchange the present process tag with the DB variable "DB\_Parts".WeightStore.setpointNo  
Process → Tag → "..." → navigate to "DB\_Parts" → "WeightStore" → select "setpointNo"
4. Repeat Step 3 for the I/O field "Actual".
5. Transfer the modified project to the touchpanel.
6. Monitor whether the Setpoint is written into the variable "DB\_Parts".WeightStore.setpointNo when you specify a new value on the panel.

#### 4.13. Task Description: Archiving part weights in "DB\_Parts" using "FieldWrite"



##### Situation up until now

In the operation, parts are placed on Bay 1 or 2 and after they have passed through the light barrier, they are counted with the IEC-counter "DB\_Parts".Part\_Counter. As well, the weight of the relative part is determined using analog value conversion and is tested for validity.

##### Task description

The individual weights of the transported parts are now to be stored in the components of the Array variable "DB\_Parts".PartWeight whenever a part has passed through the light barrier during operation. For this, use the instruction "FieldWrite".

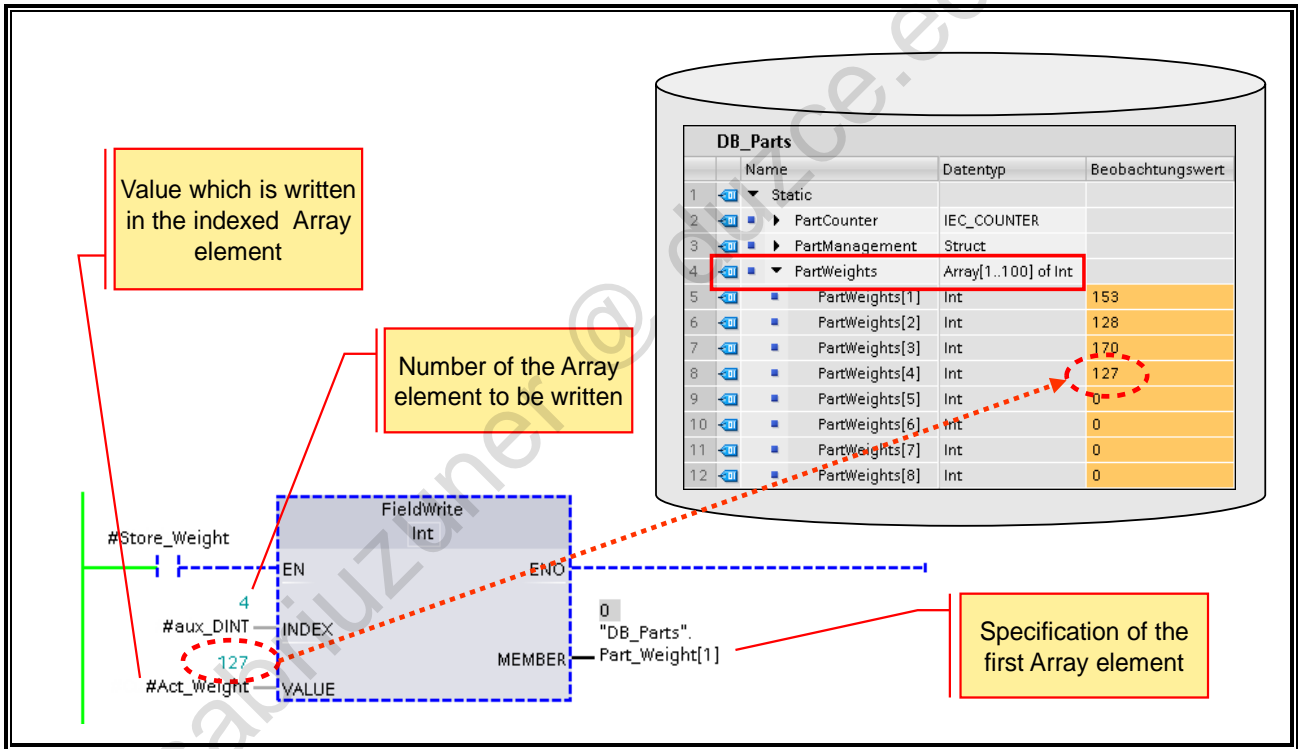
The variable "DB\_Parts".WeightStore.actualNo specifies which Array component is to be written.

For this, program all necessary instructions in the re-usable function "FC\_Ind\_Weight".

In the function, declare the following parameters

- IN
  - "NewWeight" BOOL "Trigger to save the weight"
  - "Weight" INT "Transfer parameter of weight value"
  - "ActualNo" INT "Transfer parameter of actual quantity"
- OUT
  - "WeightStore" ARRAY[1..100] of INT "Transfer parameter for DB-Array"

4.13.1. Indirect addressing of Array elements with "FieldRead" and "FieldWrite" (1)



**FieldWrite**

The instruction "FieldWrite" transfers the contents of the variable at input VALUE into a certain component of the Array variable at output MEMBER. The parameter INDEX specifies which Array component is written.

The first component of the field into which is written is specified at output MEMBER.

**Example**

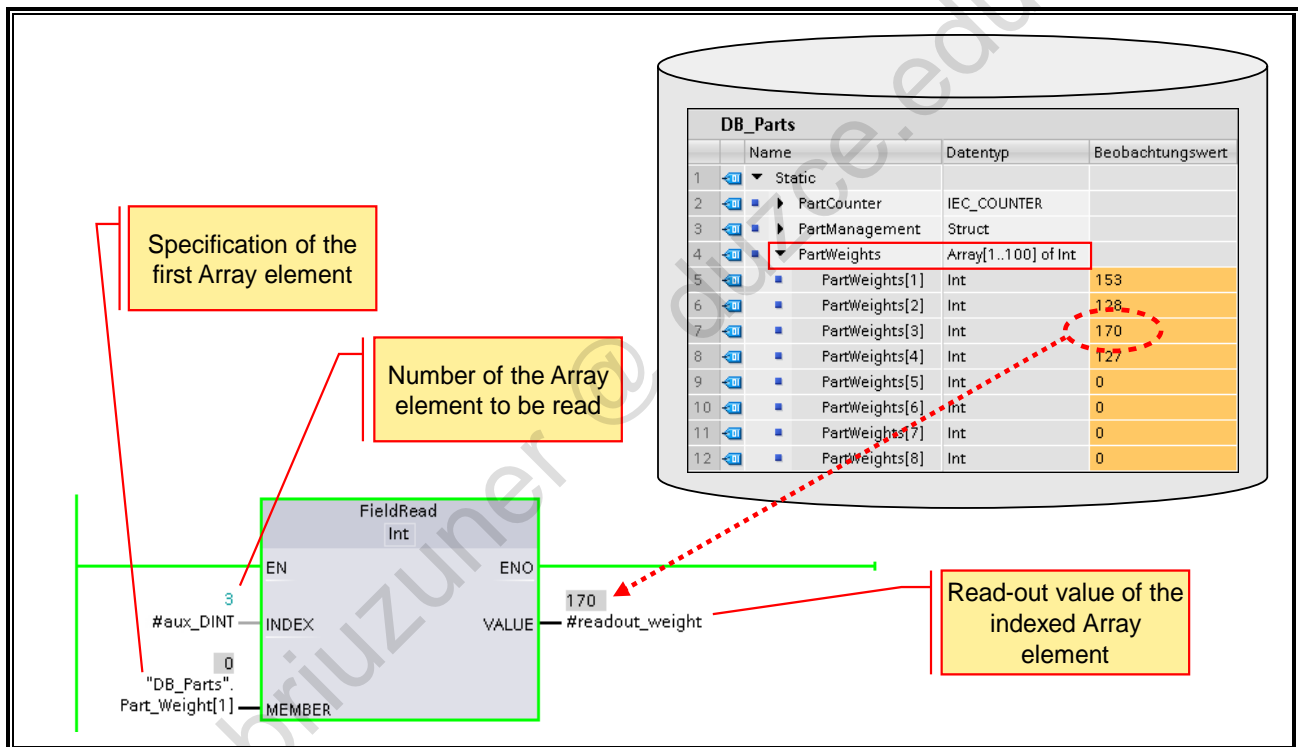
In the example shown, the weight value "127" is written in the 4th component of the Array "PartWeights". This, in turn, is in "DB\_Parts".

"127" → "DB\_Parts".PartWeights[4]



The data types of the Array component specified at output MEMBER and the variable at input VALUE must match.

#### 4.13.2. Indirect addressing of Array elements with "FieldRead" and "FieldWrite" (2)



#### FieldRead

With the instruction "FieldRead", a certain component can be read out of the Array specified at the parameter MEMBER and its contents transferred into the variable at the parameter VALUE. You define the Index of the field components to be read at the parameter INDEX.

The first component of the field which is read is specified at input MEMBER.

#### Example

In the example shown, the third element of the variable "DB\_Parts".PartWeights is read-out and assigned to the variable #Weight\_read.



The data types of the Array component specified at input MEMBER and the variable at output VALUE must match.



### 4.13.3. Exercise 6: Creating a re-usable function "FC\_Ind\_Weight" and declaring the interface

**"FC\_Ind\_Weight"**

FC_Ind_Weight				
	Name	Data type	Default value	Comment
1	Input			
2	New_Weight	Bool		The trigger to save the weight
3	Actual_Weight	Int		Actual weight
4	Part_No	Int		Actual parts quantity
5	Output			
6	Weight_Store	Array[1..100] of Int		Transfer parameter for DB-Array
7	InOut			
8	<Add new>			
9	Temp			
10	aux_DINT	DInt		auxiliary variable

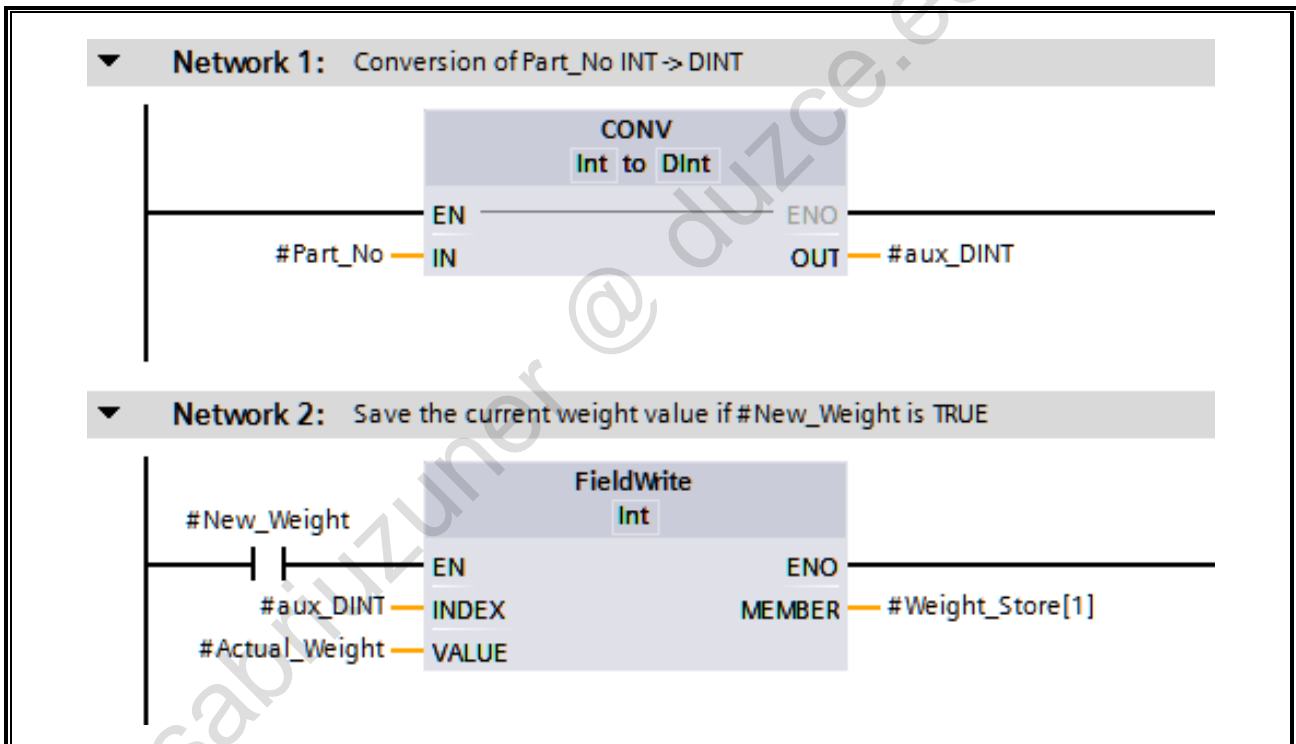
#### Task

Create the new function "FC\_Ind\_Weight" and declare the parameters and local variables shown.

#### What to do

1. Create a new function "FC\_Ind\_Weight"
2. Declare local variables as shown in the picture

## 4.13.4. Exercise 7: Programming the DB access as re-usable using "FieldWrite"

**Task**

Program the saving of the part weight values as shown in the picture.

**What to do**

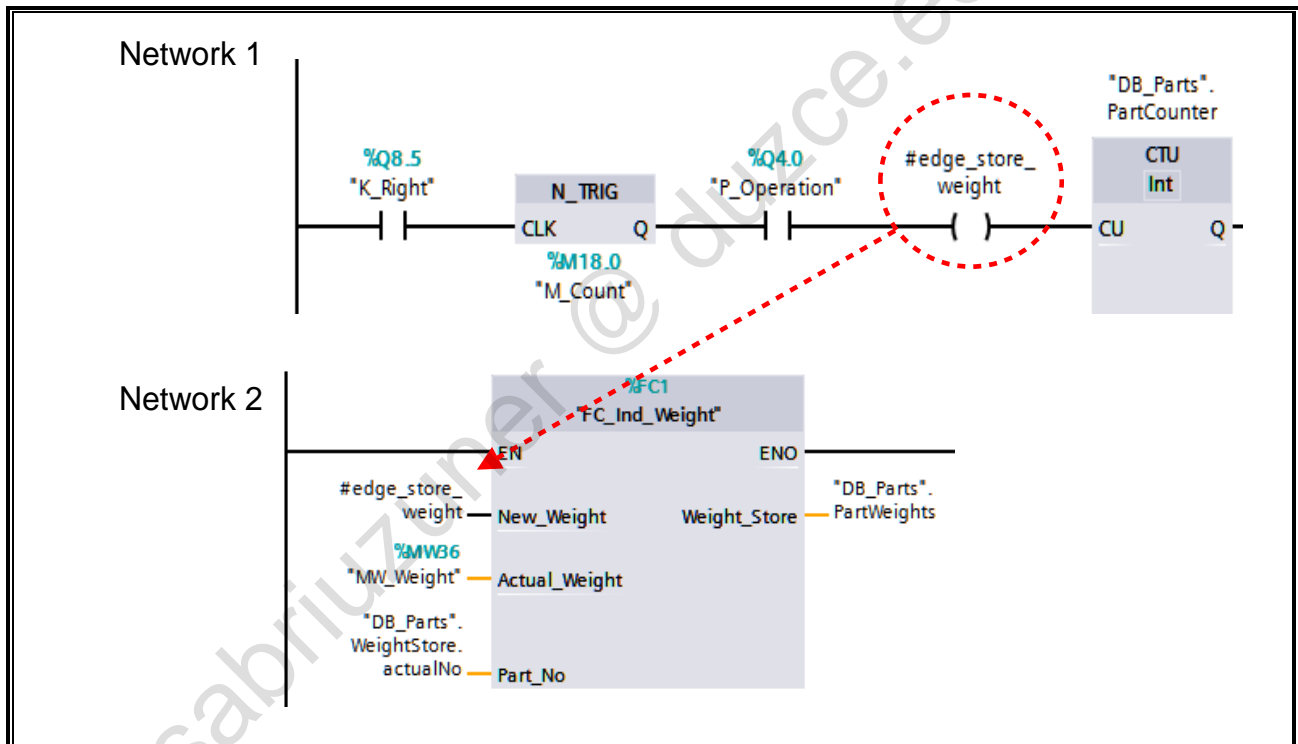
1. Program the instruction "CONV" in order to convert the local variable #ActualNo into a variable of the type DINT.  
Basic instructions → Conversion operations → CONVERT



This is necessary because the parameter INDEX of "FieldWrite" must be assigned with a variable of the type DINT.

2. Program the call of "FieldWrite" as shown in the picture.

## 4.13.5. Exercise 8: Calling the new function in "FC\_Count"



## Task

Call "FC\_Ind\_Weight" in "FC\_Count" and pass it the actual parameters shown above.

The saving of weights is triggered under the same conditions as the counting of parts. To "pick up" the signal, use an assignment and a temporary variable.

## What to do

1. Open the function "FC\_Count" and in it call the function "FC\_Ind\_Weight"
2. In Network 1, insert an assignment after the AND-query and assign the RLO to the temporary variable #temp\_store\_weight
3. Pass the current weight ("MW\_Weight"), the actual quantity ("DB\_Parts".WeightStore.actualNo), as well as the Array variable "DB\_Parts".PartWeights to the function "FC\_Ind\_Weight"
4. Save your project and download the modified program blocks into the CPU. ("FC\_Ind\_Weight" and "FC\_Count")

## 4.13.6. Exercise 9: Monitoring "DB\_Parts"

**Exercise 9: Monitoring "DB\_Parts"**

**SIEMENS**  
Ingenuity for life

DB_Parts				
	Name	Data type	Start value	Monitor value
1	Static			
2	PartCounter	IEC_COUNTER		
3	WeightStore	Struct		
4	setpointNo	Int	5	5
5	actualNo	Int	0	5
6	PartWeights	Array[1..100] of Int		
7	PartWeights[1]	Int	0	379
8	PartWeights[2]	Int	0	223
9	PartWeights[3]	Int	0	361
10	PartWeights[4]	Int	0	286
11	PartWeights[5]	Int	0	190
12	PartWeights[6]	Int	0	0
13	PartWeights[7]	Int	0	0
14	PartWeights[8]	Int	0	0
15	PartWeights[9]	Int	0	0
16	PartWeights[10]	Int	0	0

Example: Setpoint = 5

After the 5th Part Weight, Array elements remain unchanged.

SITRAIN © Siemens AG 2019  
Seite 32

TIA-MICRO2  
Data Blocks

**Task**

Test the new function by monitoring the DB "DB\_Parts".

**What to do**

1. On the touchpanel, switch on the operation
2. Enter a Setpoint on the touchpanel, such as, 3
3. Open and monitor the DB "DB\_Parts"
4. Produce parts until the Actual is equal to the Setpoint and in the process change the weight on the rotary potentiometer for every part
5. Press the pushbutton at the light barrier bay ("S\_BayLB") to reset
6. Now, once again change the weight on the rotary potentiometer
7. Produce a further part

**Result**

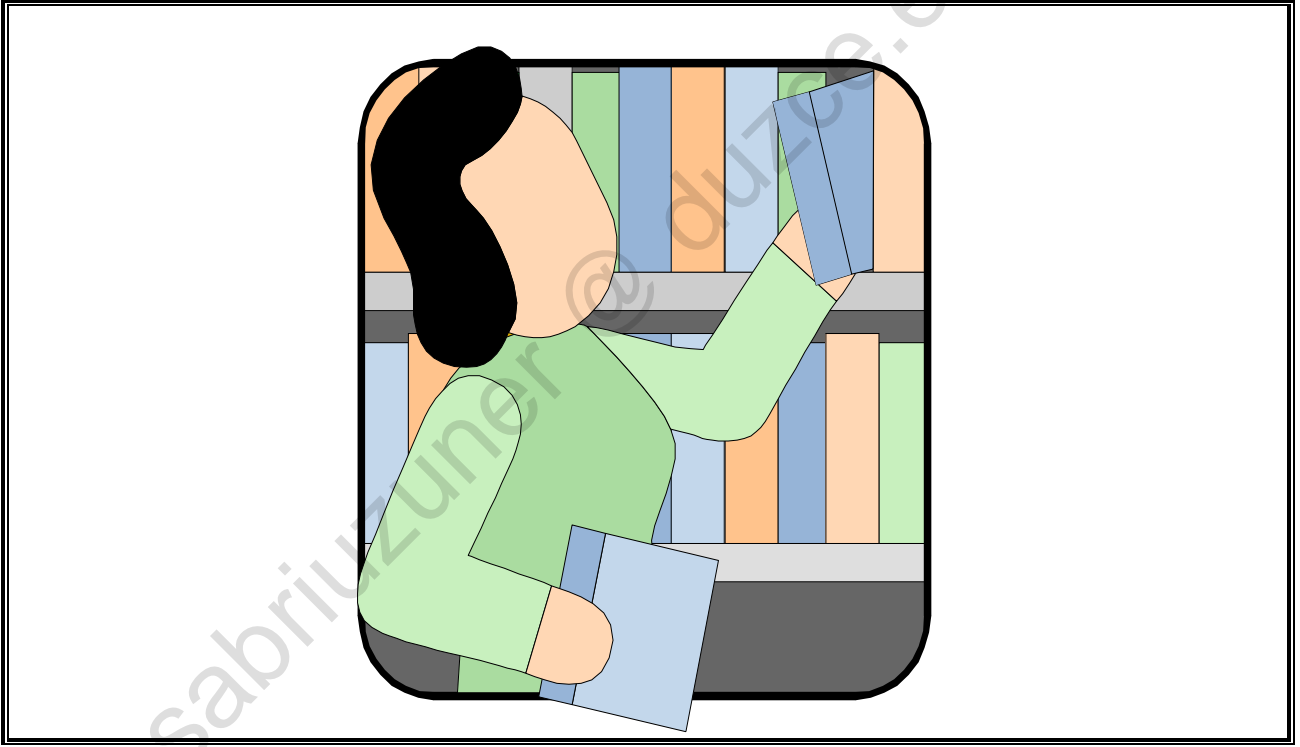
The weight values of the parts are entered one after the other in the individual Array components until the Setpoint is reached.

If a further part is produced after the Actual is reset, the old weight values are overwritten.

**What doesn't work yet**

When the Setpoint (quantity) is reached and before the new weight values can be saved, all Array components are to be initialized, i.e. written with a certain value, for example, "0". The function will be implemented in the next chapter.

#### 4.14. Additional Information



#### 4.14.1. Additional exercise: Reading back the setpoint (quantity) and adopting it as the start value

**1**

Name	Data type	Start value	Snapshot	Monitor value
Static				
PartCounter	IEC_COUNTER			
WeightStore	Struct			
SetpointNo	Int	5		10
ActualNo	Int	0		6
PartWeights	Array[1..100] of Int			

Change the Setpoint (quantity) on the touchpanel, e.g. "10"

**2**

Name	Data type	Start value	Snapshot	Monitor value
Static				
PartCounter	IEC_COUNTER			
WeightStore	Struct			
SetpointNo	Int	5	10	10
ActualNo	Int	0	6	6
PartWeights	Array[1..100] of Int			

Create Snapshot of the Monitor values

**3**

Name	Data type	Start value	Snapshot	Monitor value
Static				
PartCounter	IEC_COUNTER			
WeightStore	Struct			
SetpointNo	Int	10	10	10
ActualNo	Int	0	6	6
PartWeights	Array[1..100] of Int			

Adopt Snapshot of the Setpoint as the Start value using Copy & Paste

#### Situation up until now

After every CPU (warm) restart, the DB variable "DB\_Parts".WeightStore.setpointNo is overwritten with the Start value "5" (non-retentive). If the production processes have changed and now, for example, 10 instead of the 5 parts up until now must be transported, a new value must be specified after every CPU (warm) restart.

#### Task

Read back the changed monitor value from "Setpoint" from the online DB and adopt this value as the new start value.

#### What to do

1. Open and monitor "DB\_Parts".

My\_Project → PLC\_1 → Program blocks → Double-click "DB\_Parts" → click 

2. Create a Snapshot of the current Monitor values.

click 

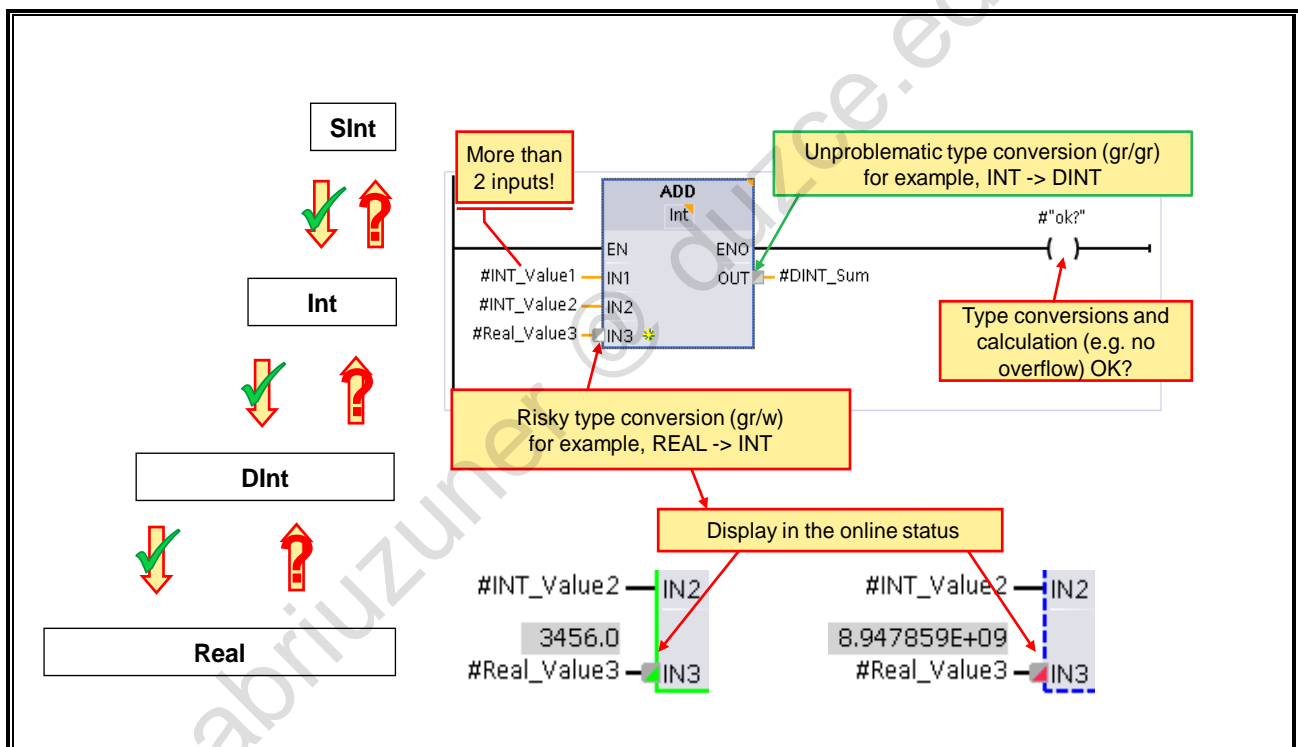


You can also display the "Snapshot" column as follows:

Right-click on "Start value", for example → show/hide column → Snapshot

3. Copy the snapshot value for "Setpoint" to the clipboard.  
Right-click → Copy
4. Insert the copied value in the Start value field of "Setpoint".  
Right-click → Paste
5. Transfer the modified block to the CPU and carry out a (warm) restart. The value that was read back should now be adopted as the Start value.

## 4.14.2. Type conversion



### Implicit Type Conversion

Basically, math operations are only possible with operands of the same data type. If, for example, two variables of different data types are to be processed mathematically (e.g. addition, multiplication...), the data types of the two variables must be adapted to one another with explicit data type conversions.

In S7-1200/1500, with the math instructions in LAD and FBD, data type conversions are implicit, that is, possibly necessary data type conversions are integrated in the math instruction.

The programming editor identifies operands that are implicitly converted with a gray rectangle. A dark gray rectangle signals that an implicit conversion without the loss of accuracy is possible, for example, when you convert the data type SINT to INT. A light gray rectangle signals that an implicit conversion is possible but during runtime errors could occur (only without IEC check). If, for example, you convert the data type DINT to INT and an overflow results, the enable output ENO is set to "0".

The example shows an integer addition of "INT\_Value" and "REAL\_Value". For this, "REAL\_Value" is implicitly converted from REAL to INT and then added to "INT\_Value". The integer result of the addition is converted in an implicit type conversion from INT to DINT and assigned to "DINT\_sum".

### Attention

For data type conversions into a data type with a smaller value range – in the example the conversion of the value of the variable "REAL\_Value" from REAL to INT – is in this respect risky, in that an overflow leads to an invalid conversion result. Accordingly, the result of the integer addition would then also be invalid. By evaluating the ENO output, errors that result in this way can be discovered.

### Note

If you monitor such a risky data type conversion online, then you can see by way of color whether the current conversion of data occurs without error or not.

Red indicates that the data type conversion is faulty and green that the conversion is without errors.

# Contents

<b>5.</b>	<b>Introduction to PROFINET .....</b>	<b>5-2</b>
5.1.	Objectives .....	5-2
5.2.	Task Description: Replacing a central I/O module with distributed I/O.....	5-3
5.3.	Industrial Ethernet: IP address and subnet mask .....	5-4
5.4.	PROFINET IO Device types .....	5-5
5.5.	Fieldbus Systems for SIMATIC S7 .....	5-6
5.5.1.	Identification of distributed I/O devices .....	5-7
5.6.	PROFINET device addressing.....	5-8
5.7.	Inserting distributed I/O into the project (Network view) .....	5-10
5.8.	Configuring a connection to the CPU and setting the address parameters .....	5-11
5.9.	Configuring distributed I/Os (Device view).....	5-12
5.10.	Writing the device name in the IO device (Device initialization) .....	5-13
5.11.	Task description: Controlling the conveyor model via the ET 200S .....	5-15
5.11.1.	Exercise 1: Inserting the ET 200S in the project and networking it .....	5-16
5.11.2.	Exercise 2: Configuring the ET 200S and setting the PROFINET address parameters ....	5-17
5.11.3.	Exercise 3: Changing the I/O addresses .....	5-18
5.11.4.	Exercise 4: Writing the device name in the IO device (Device initialization) .....	5-20
5.11.5.	Exercise 5: Compiling the modified device configuration and testing the program .....	5-21
5.12.	Additional information .....	5-22
5.12.1.	Topology editor .....	5-23
5.12.2.	Topologies.....	5-25
5.12.3.	Topology View - Topology comparison.....	5-27
5.12.4.	PROFINET proxy concept .....	5-28
5.12.5.	PROFINET Communications model .....	5-29
5.12.6.	The MAC Address.....	5-30
5.12.7.	The partitioning of the IP Address.....	5-31
5.12.8.	Detecting decentral devices automatically.....	5-32



## 5. Introduction to PROFINET

### 5.1. Objectives

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

- ... be familiar with the PROFINET IO device types
- ... understand the term "Switched Ethernet"
- ... be able to explain the principle of PROFINET IO device addressing
- ... be familiar with the procedure and necessary editors for configuring a PROFINET IO device
- ... be able to commission a distributed I/O (peripheral) with PROFINET IO

#### Objectives

In this chapter, the basics of PROFINET are handled. This includes not only the addressing procedure and the device types, but also the configuration of distributed field devices in TIA Portal.

## 5.2. Task Description: Replacing a central I/O module with distributed I/O



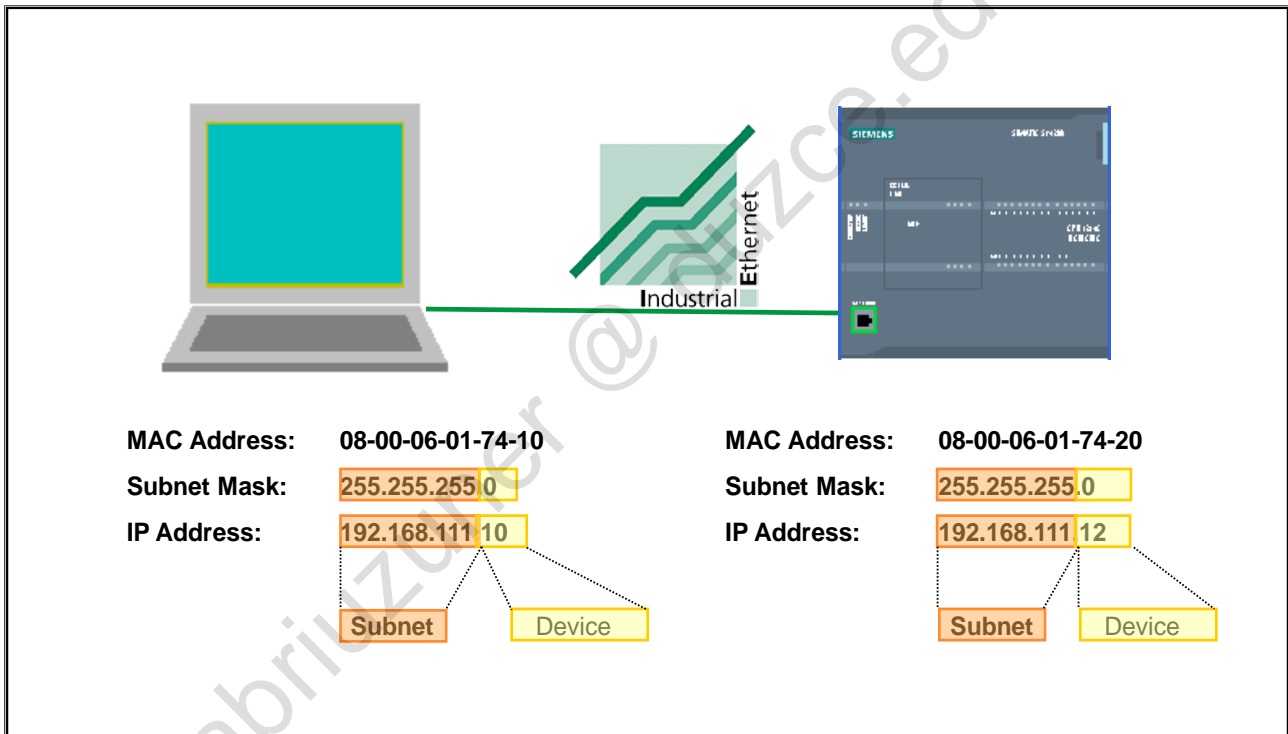
### Situation up until now

The conveyor is controlled through the 8DI/8DO module of the central rack.

### Goal

You are to commission the PROFINET system for your training device in such a way that the conveyor model can be controlled via the ET 200S with the same functionality without having to change the S7 program.

### 5.3. Industrial Ethernet: IP address and subnet mask



#### Internet Protocol

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

#### IP Addresses

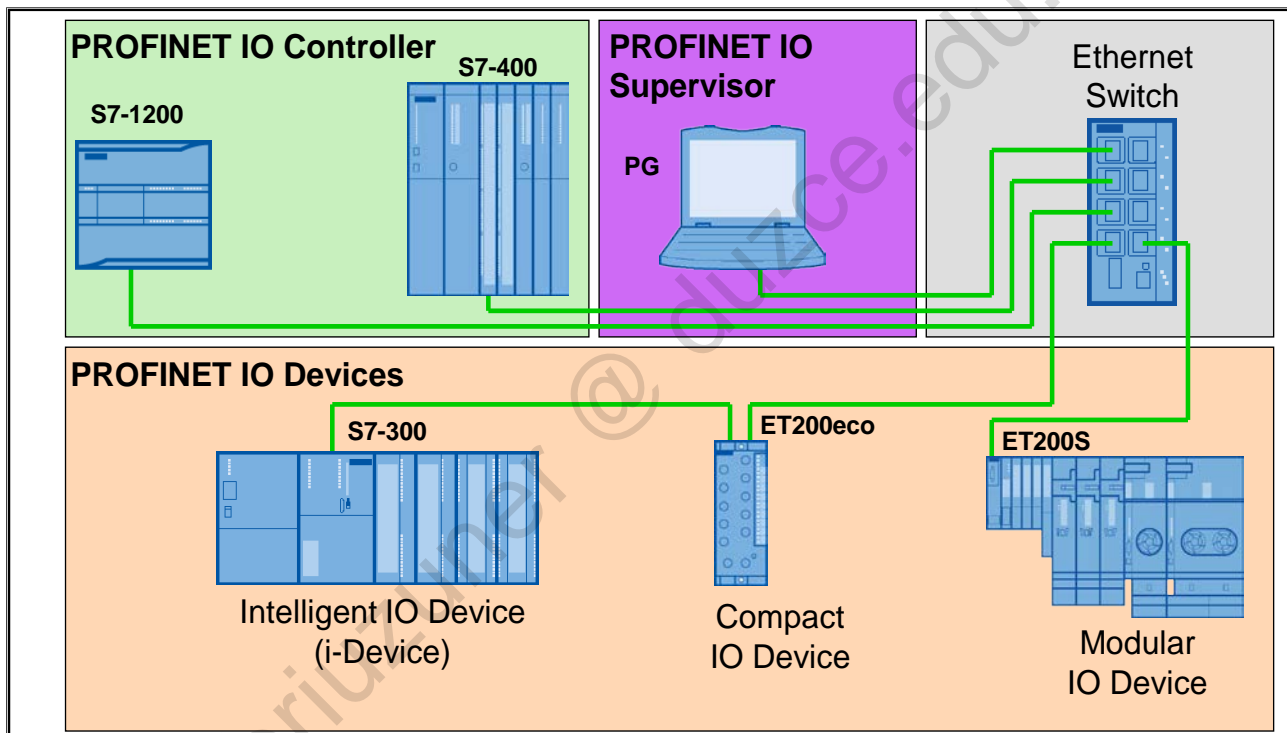
IP addresses are not assigned to a specific computer, but rather to the network interfaces of the computer. A computer with several network connections (for example routers) must therefore be assigned an IP address for each connection.

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

#### Subnet Mask

The subnet mask (also net mask or network mask) is (in binary notation) a sequence of ones followed by a sequence of zeros. The partition between ones and zeros marks the separation between the network part and the computer (host) part of the IP address.

## 5.4. PROFINET IO Device types



### PROFINET IO Controller

The IO controller (typically the PLC) establishes a logical connection to the connected IO devices after Power-On and subsequently parameterizes these (module parameters, address, etc.). (This corresponds to the function of a Class 1 Master in PROFIBUS).

### PROFINET IO Device

An IO device is a distributed IO device that is connected via PROFINET IO (this corresponds to the function of a slave in PROFIBUS).

Differentiation is made for the following IO device types:

- Compact IO device: Fixed degree of expansion
- Modular IO device: Variable degree of expansion; can be expanded or reduced as required
- Intelligent IO device: A PLC is configured not as an IO controller but as an IO device and provides a higher-level controller with I/O data

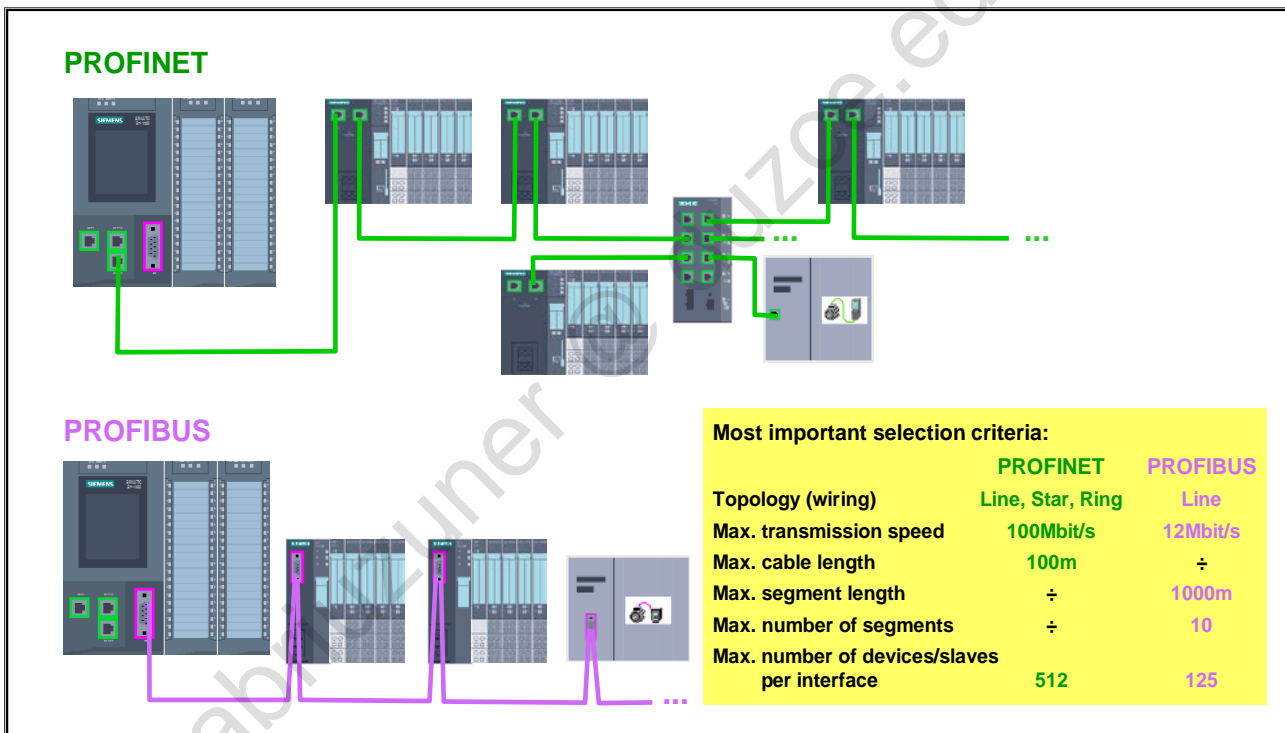
### IO Supervisor

This can be a programming device (PG), personal computer (PC) or Human Machine Interface (HMI) for commissioning or diagnostic purposes. (This corresponds to a Class 2 Master in PROFIBUS).

### Ethernet Switch

PROFINET is based on Ethernet. For that reason, switches are always used as "network distributors". Every node device is connected to a switch via a so-called "point-to-point" connection. This is also referred to as a "Switched Ethernet". In most PROFINET devices, a 2 or multi-port switch is already integrated so that it is very easy to establish a line structure (comparable to PROFIBUS).

## 5.5. Fieldbus Systems for SIMATIC S7



### Fieldbus systems for SIMATIC S7

To connect distributed I/Os, there are different bus systems.

The most important for SIMATIC S7 are:

- PROFINET**  
 As the standard for communication applications at the field level it enables the connection of distributed field devices via Industrial Ethernet.  
 The Industrial Ethernet network is a local area network (LAN) according to the international Standard IEEE 802.3 (Ethernet) and is designed for the industrial sector. It enables open and comprehensive network solutions with a high transmission performance.
- PROFIBUS**  
 It is the bus system for local area networks (LANs) with only a few participants. Through its fulfillment of requirements according to EN 50170, PROFIBUS ensures openness for the connection of standard-conforming components of all manufacturers.

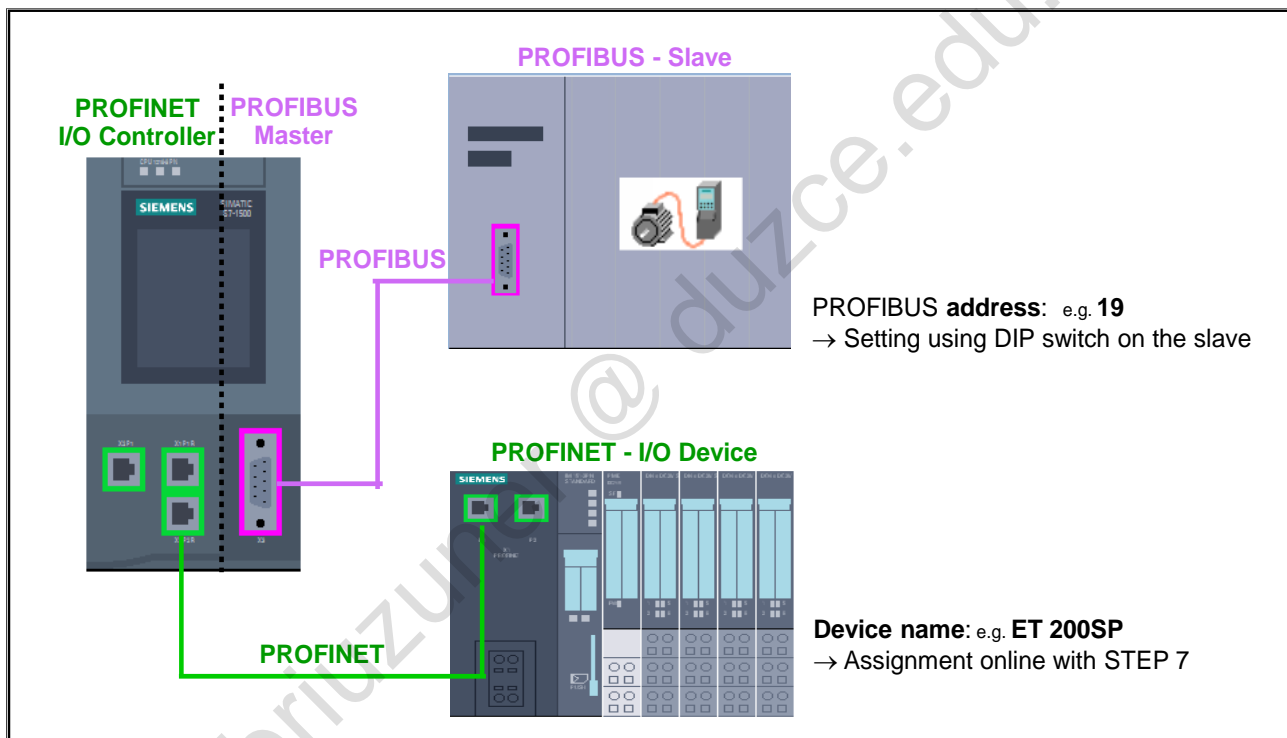
Due to the physical and communication-related differences of the two bus systems, there are various criteria which are used for the selection of the most suitable bus system.

### Cable length, segment length

For PROFIBUS, a module line must be reinforced after 100-1000m (depending on the transmission speed used); otherwise, the maximum bus length is reached.

For PROFINET, every connected component takes over this function. For that reason, only the cable length between two modules is relevant here.

### 5.5.1. Identification of distributed I/O devices



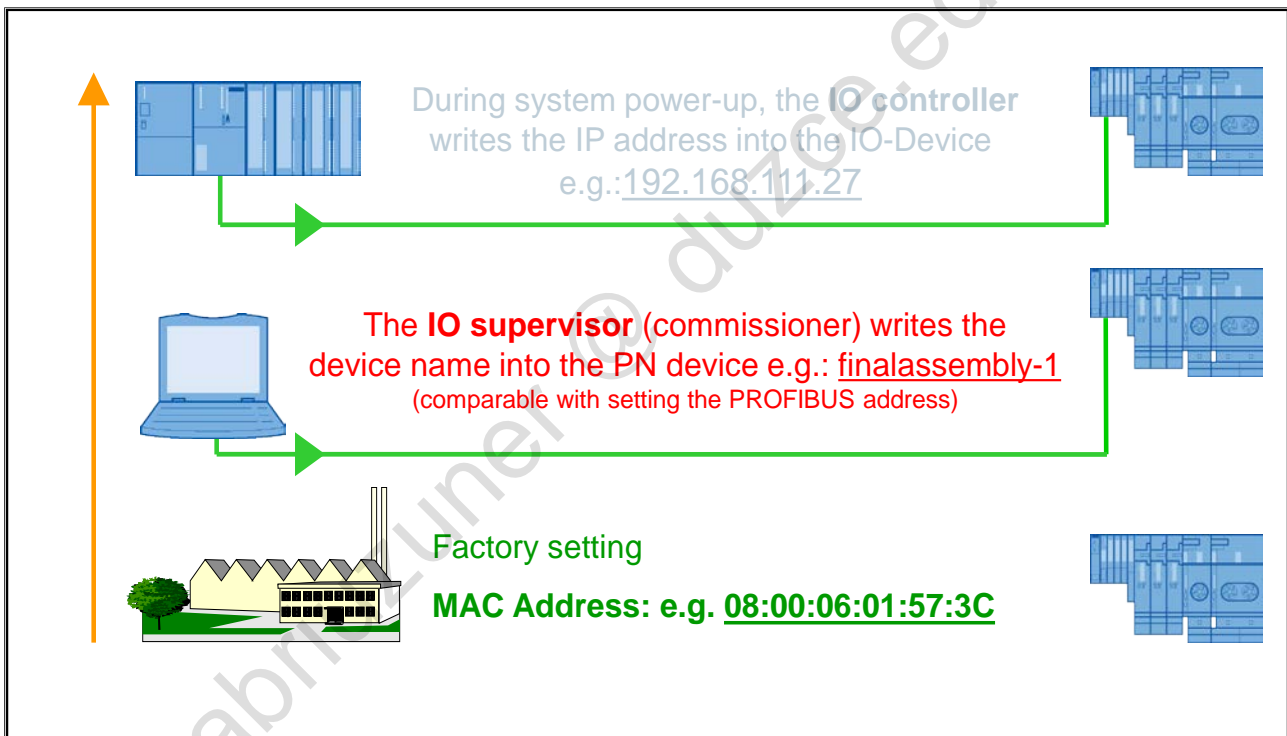
#### Distributed I/O devices

During start-up, the CPU searches the configured PNIO devices or DP slaves and parameterizes these according to the loaded device configuration.

Both fieldbus systems use different methods for identifying I/O modules:

- **PROFIBUS**  
The set PROFIBUS address is used to search for the configured DP slave.  
The setting is typically made through the DIP switch on the slave.
- **PROFINET**  
The assigned device name is used to search for the configured PNIO device.  
The assignment of the device name (device initialization) is done from SIMATIC STEP 7 engineering platform through an online function.  
The parameterized IP address is then assigned to the PNIO device by the PNIO controller (CPU).

## 5.6. PROFINET device addressing



### PROFINET device addressing

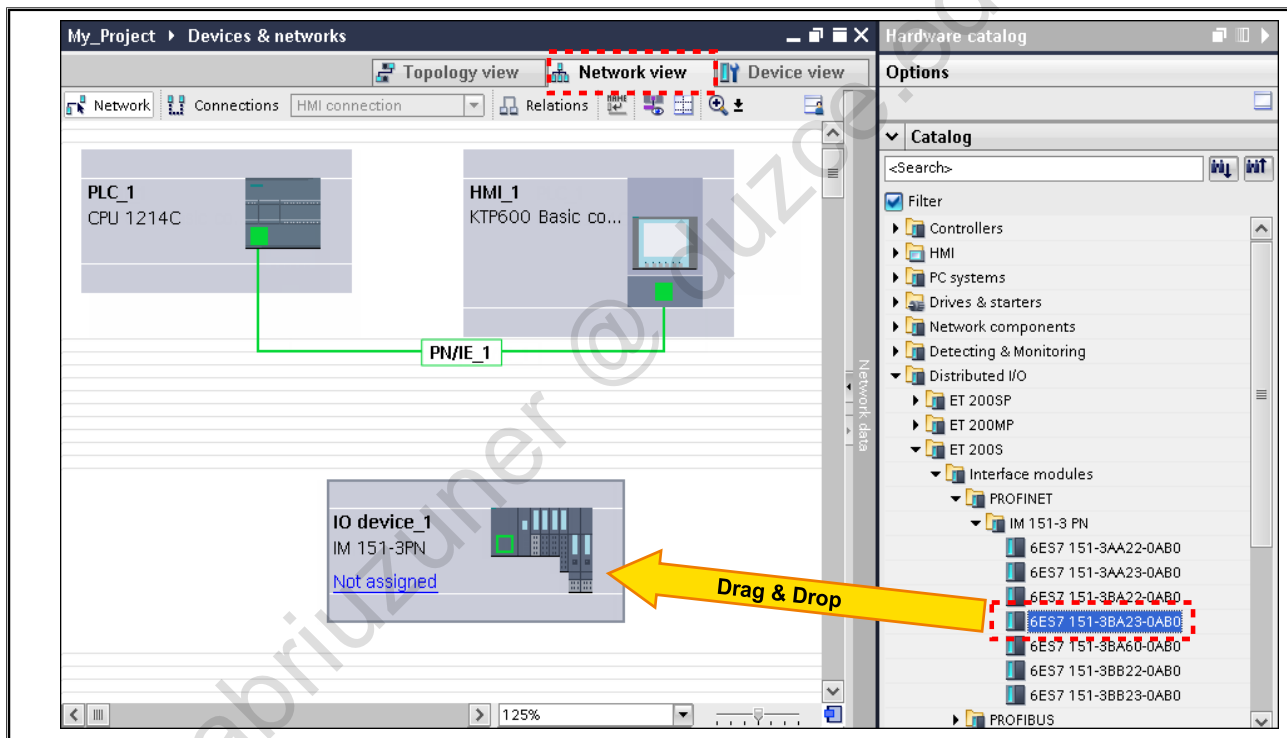
So that the IO devices are accessible for the IO controller, they must be supplied with unique address parameters. After the addressing, every IO device has three address parameters

- **MAC Address**  
In its factory settings, every PROFINET device already has a fixed, world-wide unique MAC address. As a rule, this cannot be changed. It is required for the Real-Time communication.
- **Device Name**  
Before an IO device can be addressed by an IO controller it must have a device name. This procedure was chosen for PROFINET because names are easier to handle than complex IP addresses.  
So that the individual devices are accessible during the IO controller's system power-up, they are given device names. This occurs through the Supervisor. The device names which were given for the individual IO devices offline must match the online device names. This is comparable with setting the PROFIBUS address. If errors are made here, the IO controller cannot reach the IO device.  
The rules for the converted names are listed in the following. If the converted name is not to be different from the name of the module, then the name of the module must comply with these rules.
  - The name consists of one or more labels which are separated by a dot [.]
  - Total length of the name: 1 to 240 characters
  - Length of a label: 1 to 63 characters
  - A label consists of the characters [a-z; 0-9]
  - Labels must not begin or end with the characters "-"
  - The first label must not begin with "port-xyz" or "port-xyz-abcde" (a,b,c,d, e,x,y,z=0-9)
  - The name must not have the following format: n.n.n.n (n=0-999)

- IP Address  
In addition to the device name and the MAC address, an IO device also requires an IP address so that acyclic Read/Write services can be executed, for example. During system power-up, the IO controller assigns the IP addresses stored in the device configuration to the IO devices after checking the device names.



## 5.7. Inserting distributed I/O into the project (Network view)

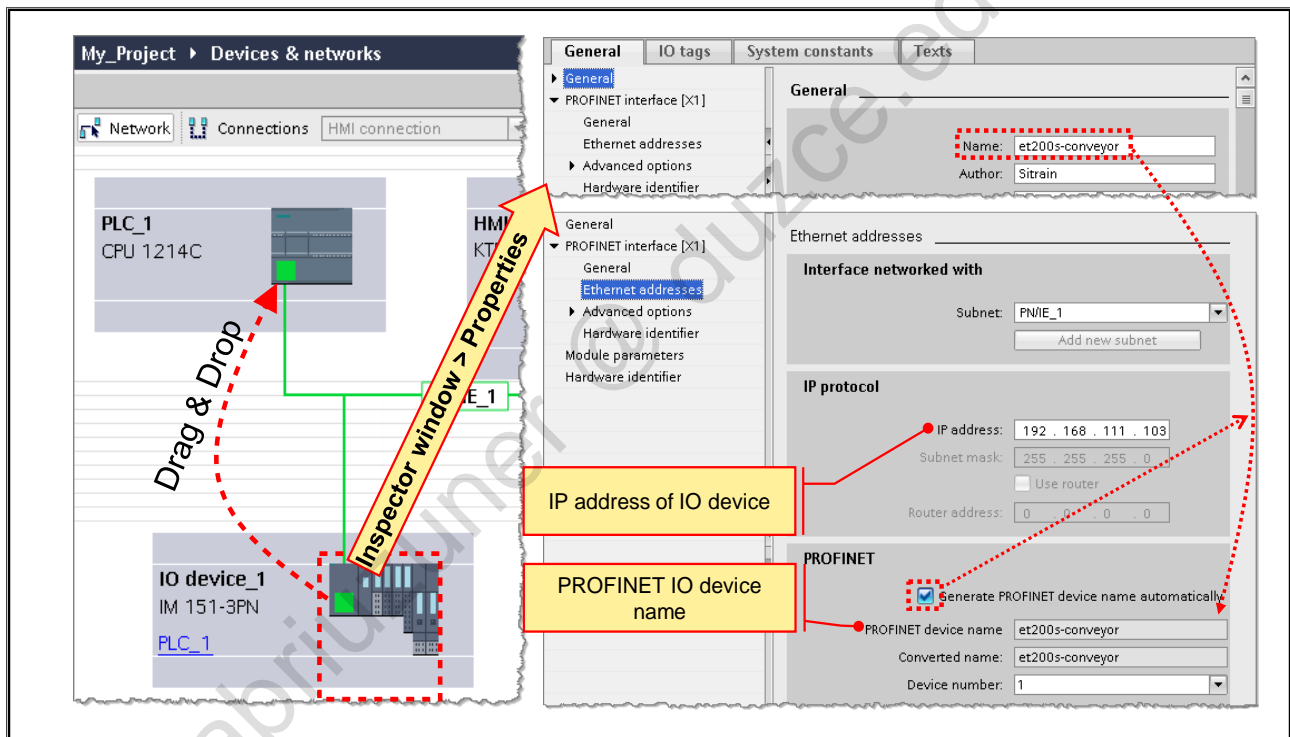


Inserting distributed I/O into the project

PROFINET IO devices are added in the Network view. Here, you can insert the relevant devices into the project by dragging & dropping them from the Hardware catalog.

In the beginning, the added device is not assigned to any controller and therefore appears in the Project tree as (an) "Unassigned devices" in the same level as the PLCs and HMIs.

## 5.8. Configuring a connection to the CPU and setting the address parameters



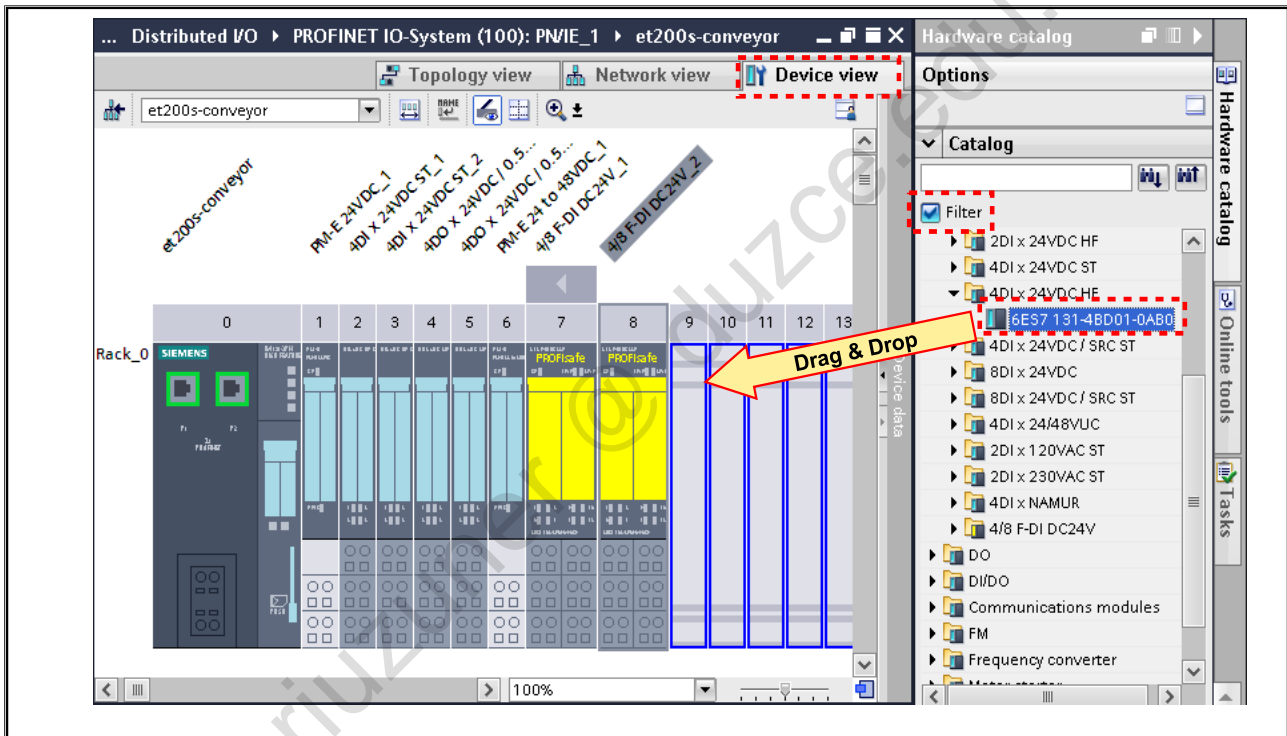
### Configuring the connection to the CPU

IO devices must be assigned to an IO controller. This is done by dragging & dropping the PROFINET interface of the IO device onto the PROFINET interface of the IO controller. The IO controller now recognizes the assignment and, during system power-up, queries the IO device with the configured device name.

### Address parameters

- **Device Name**  
The PROFINET device name is entered in the General part of the PROFINET interface of the respective IO device.
- **IP Address**  
The IP address is automatically assigned (first free subnet address) through the assignment of the IO device to the IO controller. It can be changed later in the Properties of the PROFINET interface of the IO device.

## 5.9. Configuring distributed I/Os (Device view)

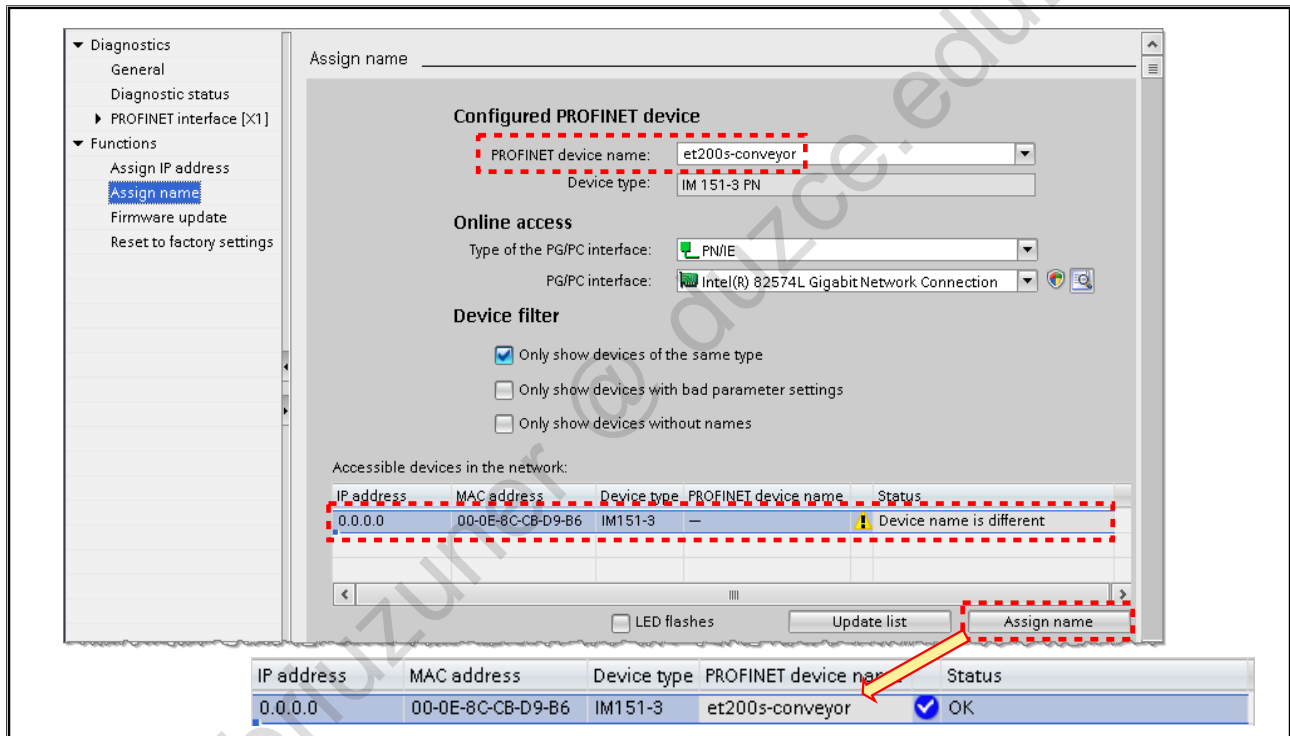


### Configuring distributed I/Os

Distributed I/Os are configured in the device view by dragging the desired components from the hardware catalog onto the individual module slots.

If the checkmark at "filter" is set, only the compatible modules are automatically displayed.

## 5.10. Writing the device name in the IO device (Device initialization)



### Device initialization

The assignment of the device name to an IO device is the most important step in PROFINET addressing. The device name configured offline and the device name that exists online must match since the IO controller first checks the device names of the connected IO devices and then assigns the configured IP addresses during system power-up. If an IO device is not accessible under its configured device name, the IO controller cannot establish a connection to the IO device.



The IP address does not have to be assigned manually. It is assigned by the IO controller during system power-up (after checking the device name). If, however, an IP address is assigned manually, it is then overwritten by the IO controller.

### Ways of assigning a name

In principle, there are three ways of writing (assigning) the device name in an IO device, whereby only two of the three ensure that the offline configured device name really gets written into the IO device without errors.

- Version 1 and 2 (sure) → if possible, use one of these procedures!

The assignment of the device name is triggered from the device configuration of the IO device.

Device configuration of IO device → Right-click on the Interface module (Slot 0) → Online & diagnostics → Functions → Assign name (see picture)

Or

Device configuration of IO device → Right-click on the Interface module (Slot0) → Assign device name

For both versions, the IO device is selected based on the MAC address and, to be sure that you have selected the right one, you can make the LEDs on the selected IO device flash by checking "LED flashes". Since the configured device name is adopted, you can't make any typing errors.

- Version 3 (possible typing errors)

The assignment of the device name is triggered via "Accessible devices" or "Online accesses".

Project view → Online accesses → Ethernet interface → IO device → Online & diagnostics → Functions → Assign name

Here, any device name you choose can be assigned. This procedure can be used if you do not have the original project. The disadvantage vis-à-vis Version 1 or 2 is that you can make typing errors, that is, the device name does not match the offline configuration. In this way, the IO device is no longer accessible to the IO controller.

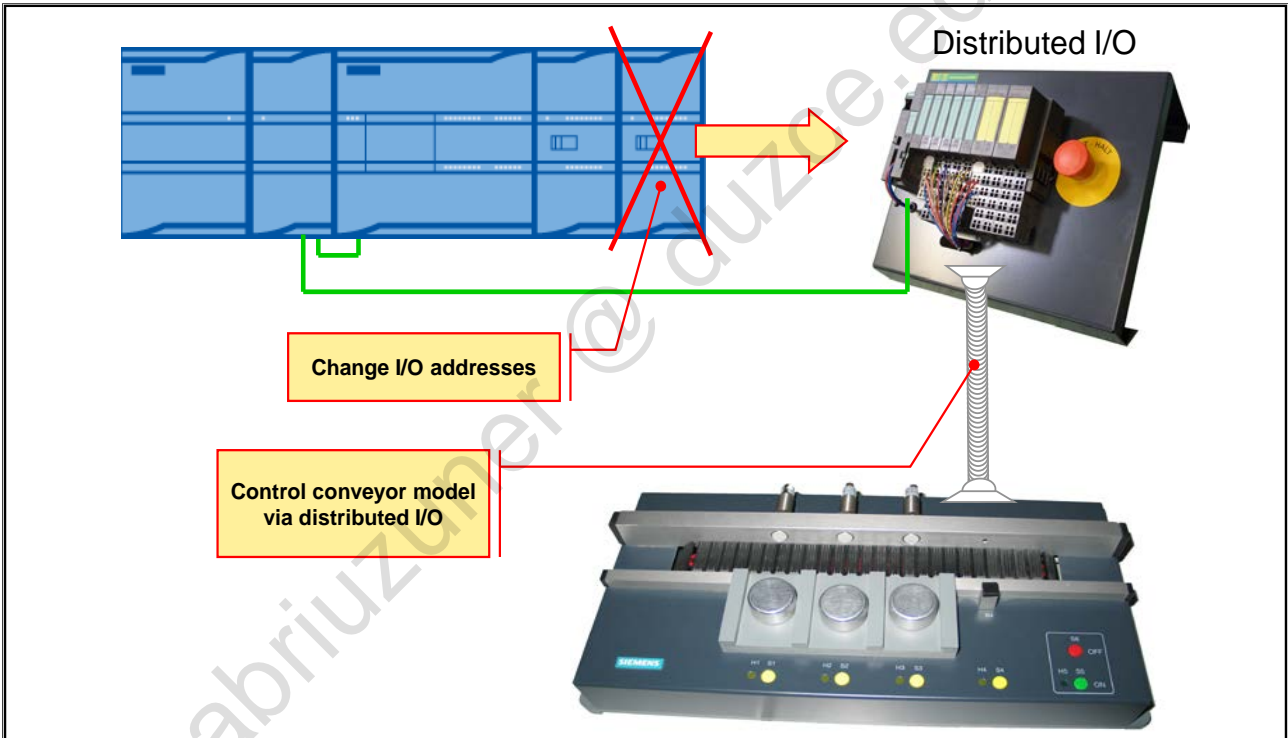
### More than one PROFINET device of the same type

Should several nameless PROFINET devices of the same type be available on the network, they can only be differentiated by their MAC address. This is printed on the Interface modules.

Alternatively, the function "LED flashes" can also be used to differentiate the PROFINET devices from one another. For this, a device is selected and the function "LED flashes" is started.

The LINK-LED(s) of the selected device always flash. This/These LED(s) normally show(s) that there is a physical connection between the device and the next switch.

### 5.11. Task description: Controlling the conveyor model via the ET 200S

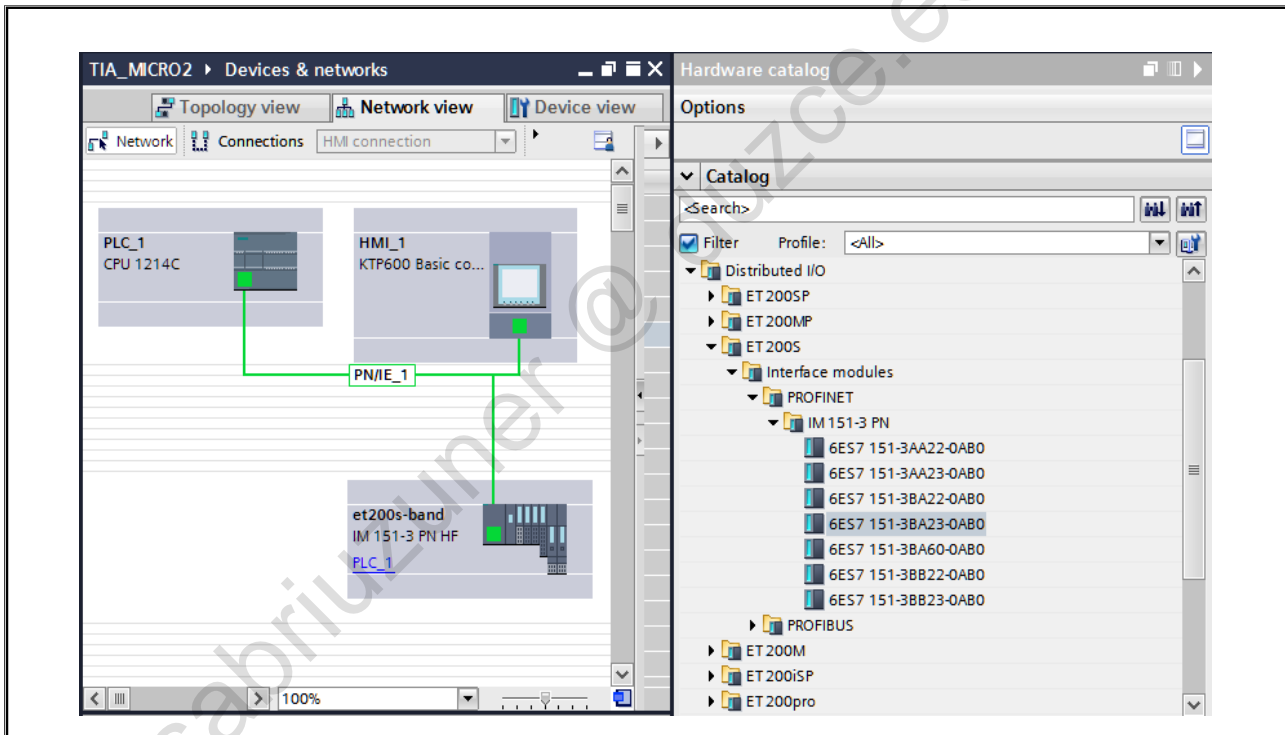


#### Task description

The control of the conveyor model through the central 8DI/8DO module is now to be replaced by the I/O modules of the ET 200S.

For this, you must insert the new IO device in your project and configure it. Furthermore, the current I/O addresses of the conveyor model are to be used (QB8 and IB8) so that the user program doesn't have to be changed.

### 5.11.1. Exercise 1: Inserting the ET 200S in the project and networking it



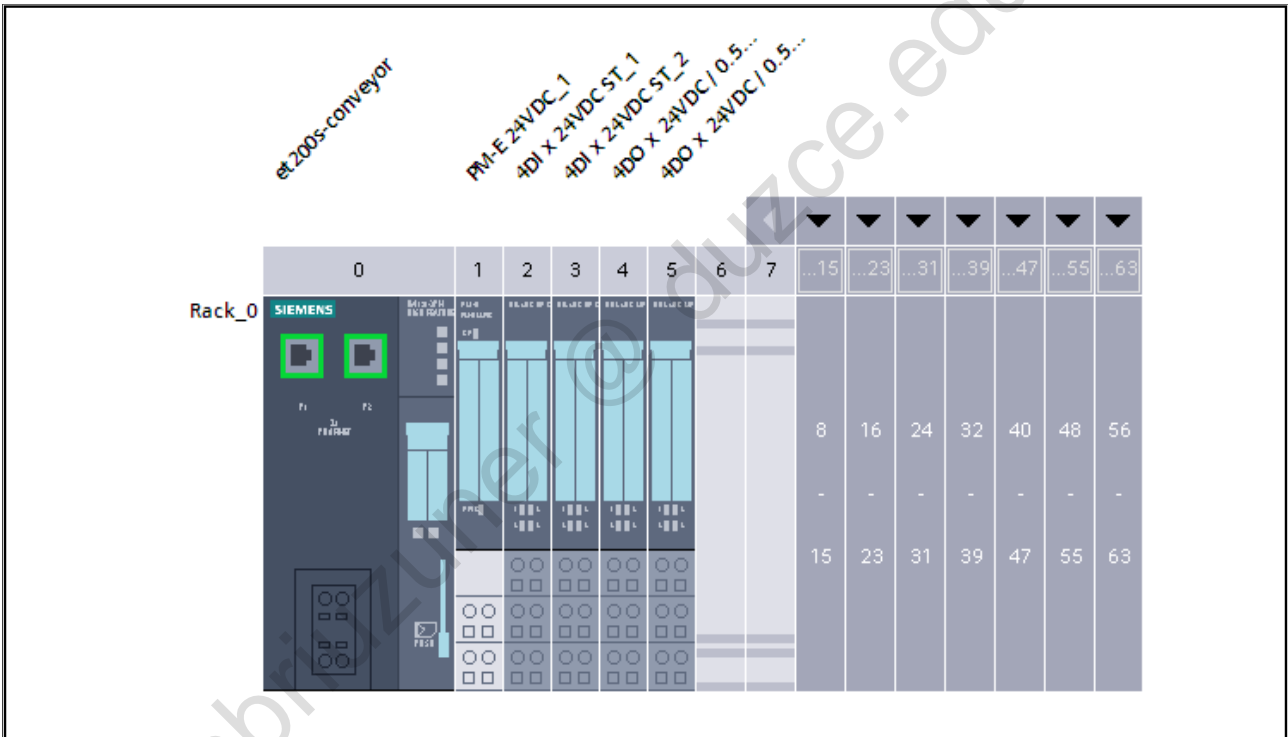
#### Task

Insert the Interface module of your ET 200S into your project and network it with the CPU Station "PLC\_1".

#### What to do

1. Open the Network view of your project.  
My\_Project → Devices & networks → Network view
2. From the Hardware catalog, select the head module of your ET 200S and drag it onto the network plan using drag & drop.
3. Network ET 200S with the CPU.  
Drag the PROFINET interface of the ET 200S to the PROFINET interface of the CPU and drop it there.

**5.11.2. Exercise 2: Configuring the ET 200S and setting the PROFINET address parameters**



**Task**

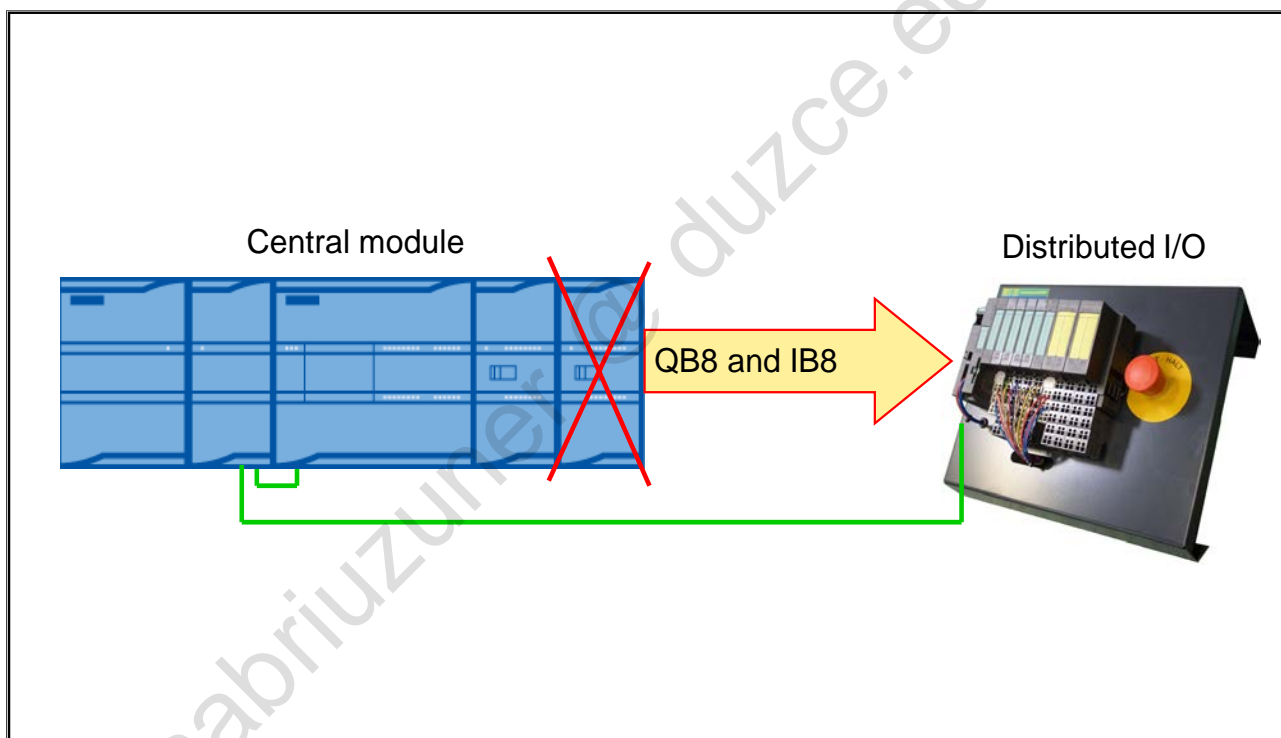
Configure the modules of the ET 200S and set the address parameters.

**What to do**

1. Open the Device view of the ET 200S.
2. Equip the ET 200S with the modules according to the order number on the individual modules.
3. Set "et200s-conveyor" as the PROFINET device name  
Select the Interface module (Slot 0) → Inspector window → Properties → General → Name
4. Set the IP address 192.168.111.103 for the IO device.  
Select the Interface module (Slot 0) → Inspector window → Properties → PROFINET interface → Ethernet addresses → IP protocol



### 5.11.3. Exercise 3: Changing the I/O addresses



#### Situation up until now

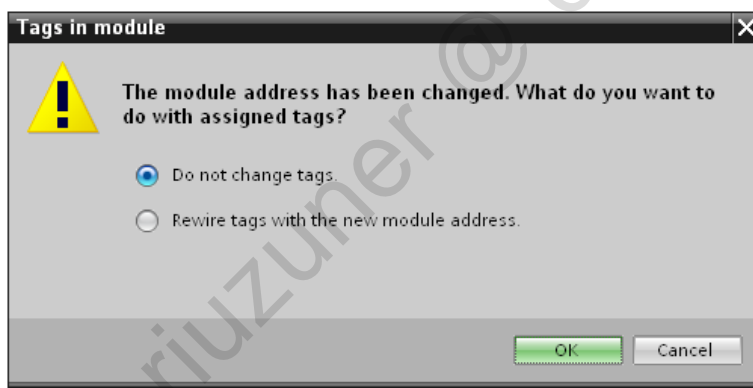
The binary input signals of the conveyor model are in IB8; the output signals in QB8. Currently, QB8 and IB8 are processed by the central 8DI/8DO module.

#### Task

QB8 and IB8 are to be assigned to the I/O modules of the ET 200S.

#### What to do

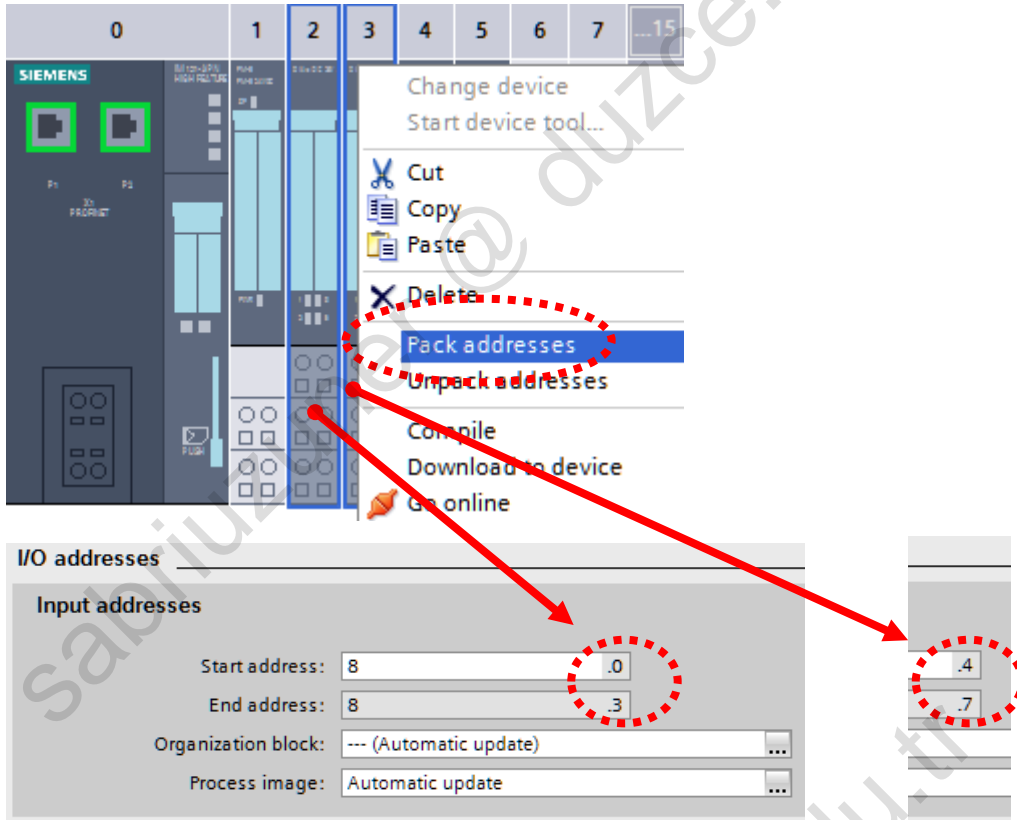
1. Open the Device view of "PLC\_1" and navigate to the address properties of the 8DI/8DO module and assign the following addresses:  
Module → 8DI/8DQ → I/O addresses  
Input addresses → Start address: 88  
Output addresses → Start address: 88
2. In the pop-up message, select "Do not change tags", since the PLC tags of IB8 or QB8 are otherwise rewired to IB88 or QB88.



3. Open the Device view of the ET 200S.
4. For the input and output addresses of the ET 200S, configure the addresses IB8 and QB8.

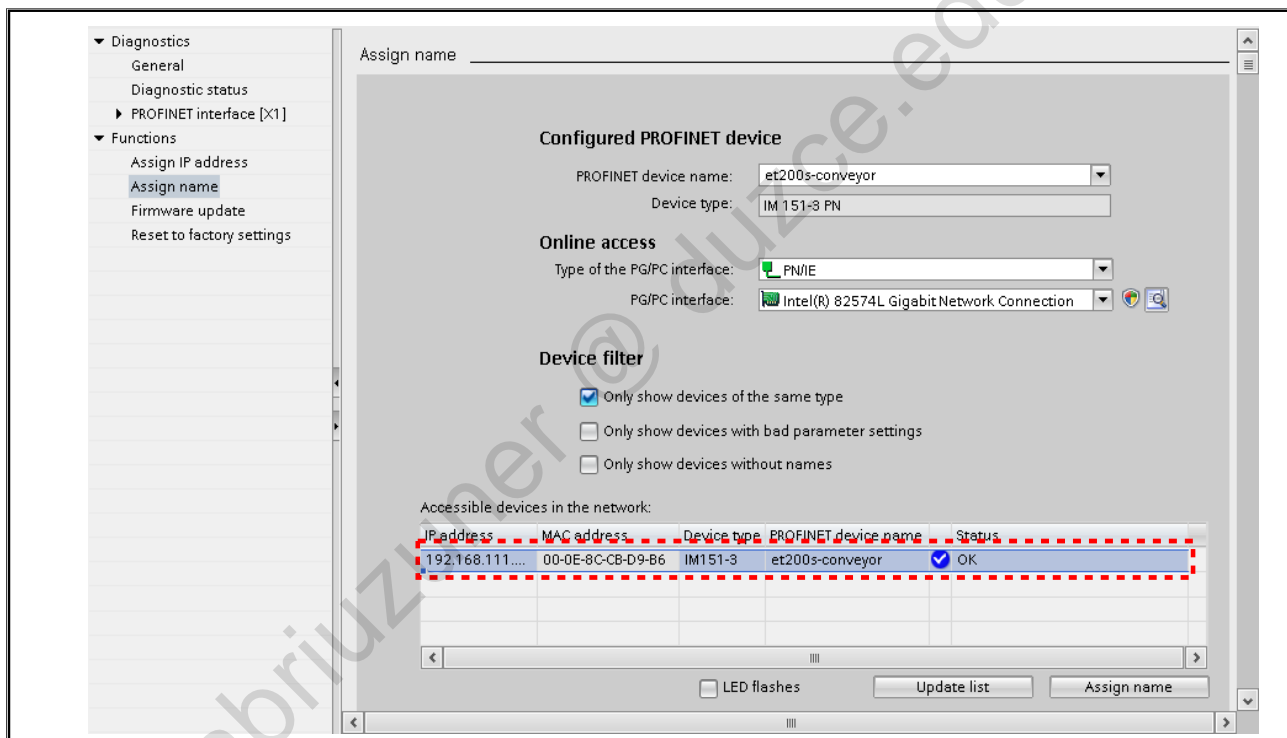


So that the 8 channels of a module are in one and the same byte, you must pack the addresses. For this, select both modules, right-click on one of the modules and then click on "Pack addresses"



sabriuzuner@duzce.edu.tr

### 5.11.4. Exercise 4: Writing the device name in the IO device (Device initialization)



#### Task

Assign the device name "et200s-conveyor" to your ET 200S.

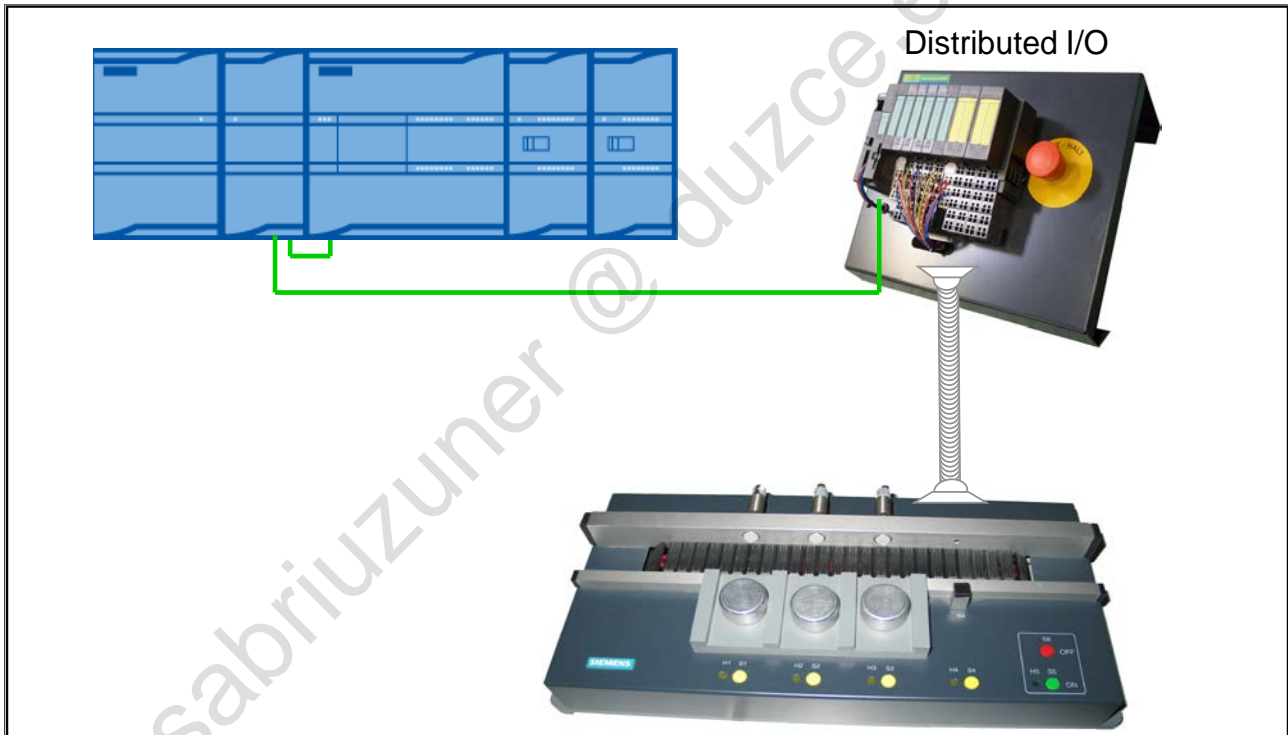
#### What to Do

1. Open the device configuration of the ET 200S
2. Open the context menu of the Interface module and select "Online & diagnostics"  
Right-click on the Interface module (Slot 0) → "Online & diagnostics"
3. Navigate to "Assign name"  
"Online & diagnostics" → Functions → Assign name
4. From the list of accessible devices select the device of the type "IM151-3" and click on "Assign name"
5. Check whether the device name was assigned correctly by updating the list of accessible devices.



The ET 200S still doesn't have an IP address. This will be assigned by the IO controller after you have transferred the modified hardware configuration of "PLC\_1" and the IO controller (CPU 1214) starts up again.

### 5.11.5. Exercise 5: Compiling the modified device configuration and testing the program



#### Preparation

Remove the conveyor cable connector from the central training device and insert it in the ET 200S' socket.

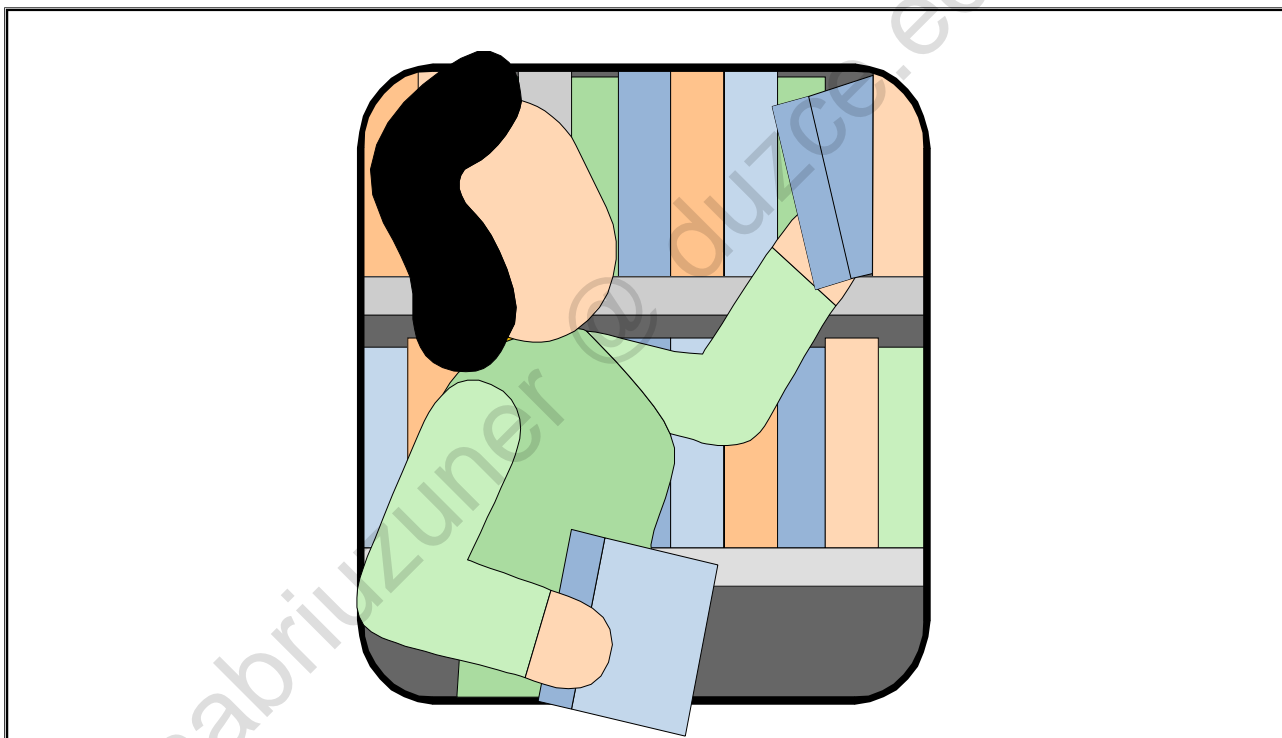
#### Task

Download the modified hardware configuration of "PLC\_1" into the controller and test whether your user program behaves as usual.

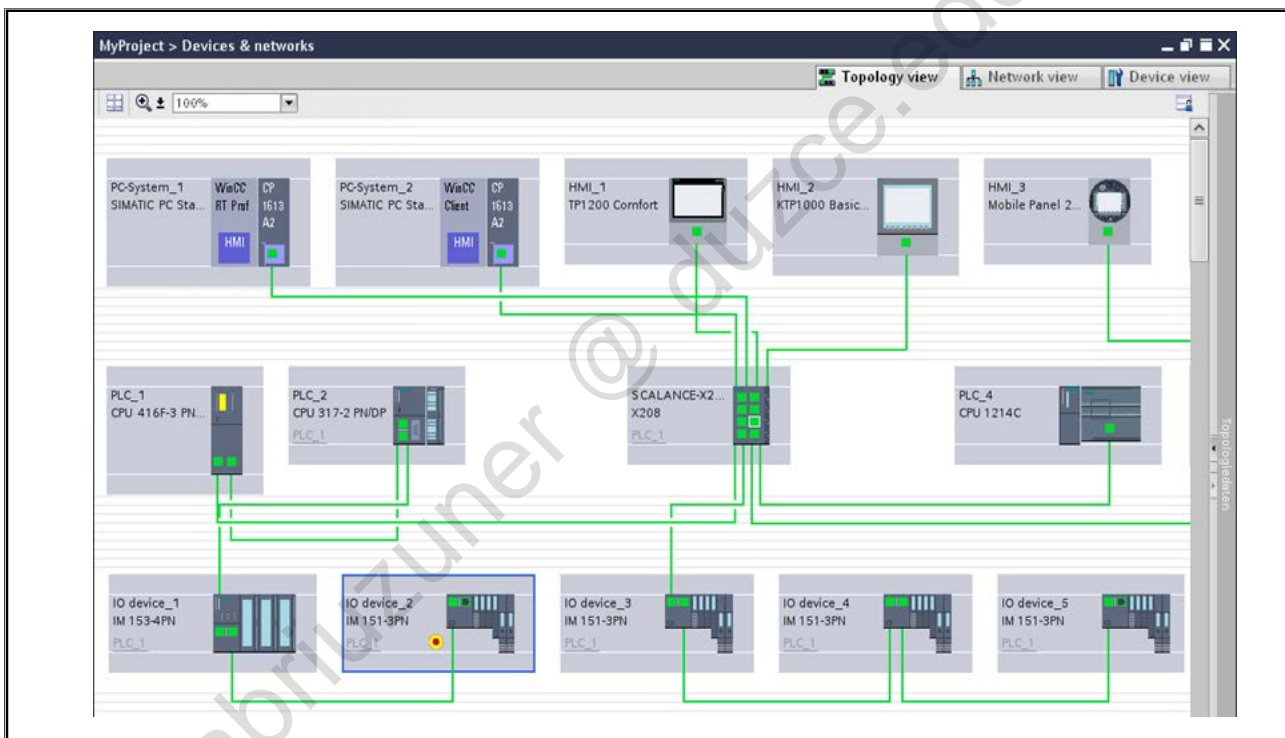
#### What to do

1. Transfer the modified hardware configuration of "PLC\_1" into the controller.
2. Test your user program by producing at least 1 part and test whether the conveyor model can be jogged to the left and the right when "P\_Operation" = FALSE.

## 5.12. Additional information



### 5.12.1. Topology editor



#### Topology editor

The configuration of the network topology is required for certain PROFINET functions. This includes, for example, the functions "Enabling device replacement without exchangeable medium" or "PROFINET IRT".



#### Caution!

The planned topology is loaded into the IO controllers involved. If the real topology is changed later, the functions that are built up on it are no longer executed correctly. The LEDs "BF" and/or "MAINT" light up.

#### Functions of the topology view

The Topology view is one of three work areas of the Devices & networks editor. The following tasks can be carried out:

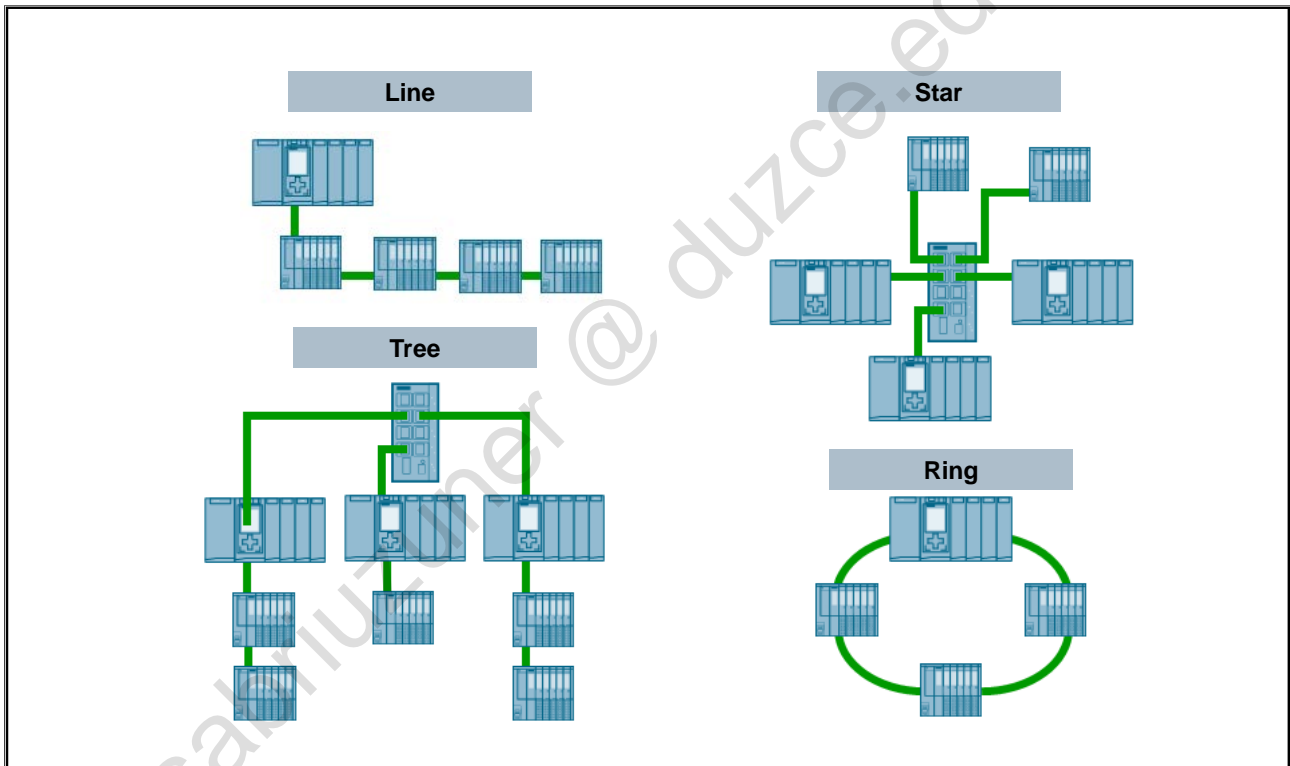
- Display Ethernet topology
  - Display all PROFINET devices and passive Ethernet components of the project including ports
  - Display interconnections between the ports
  - Display associated logical networks
  - Display diagnostic information of all ports
- Configure Ethernet topology
  - Create, change and delete port interconnections
  - Rename stations, devices, interfaces, ports
  - Add PROFINET devices and passive Ethernet components to the project from the Hardware catalog

- Determine and minimize differences between the setpoint topology and the actual topology
  - Carry out offline/online comparison of Ethernet modules, Ethernet ports and Ethernet port interconnections
  - Adopt topology information existing online into the offline project

#### **Differences between Network View and Topology View**

- The Network view displays all logical subnets of the project.
- The Topology view displays all Ethernet components of the project. This also includes passive components such as switches and media converters and cables.
- The position of a device in the Network view and its position in the Topology view are independent of one another, that is, as a rule, one and the same device is in a different position in each of the two views.

## 5.12.2. Topologies



### Star Structure

The simplest network structure is a central Switch that enables the data transmission between the connected devices.

- Advantages
  - Easy managing, monitoring and diagnosis in the network
  - Flexible adding and removing of connections
- Disadvantages:
  - Single point of failure (one single critical point in case of error)
  - Wiring costs
- Appropriate for:
  - Small production areas
  - Individual production machines
  - Host system of a larger system



## Line Structure

The devices are arranged in series.

- Advantages:
  - Cable cost savings for larger installations
  - Traditional fieldbus structure
- Disadvantages:
  - The transmission time can be influenced by the routing

## Tree Structure

In principle, the tree structure is a combination of the line and star structure.

- Advantages:
  - The system is very transparent
  - Little data traffic since local data is restricted to the source location
  - Greater safety since local data remains in the originating area
  - Is also used to divide complex systems into logical subsystems
- Disadvantages:
  - Single point of failure (one single critical point in case of error)
  - Wiring costs

## Ring Structure

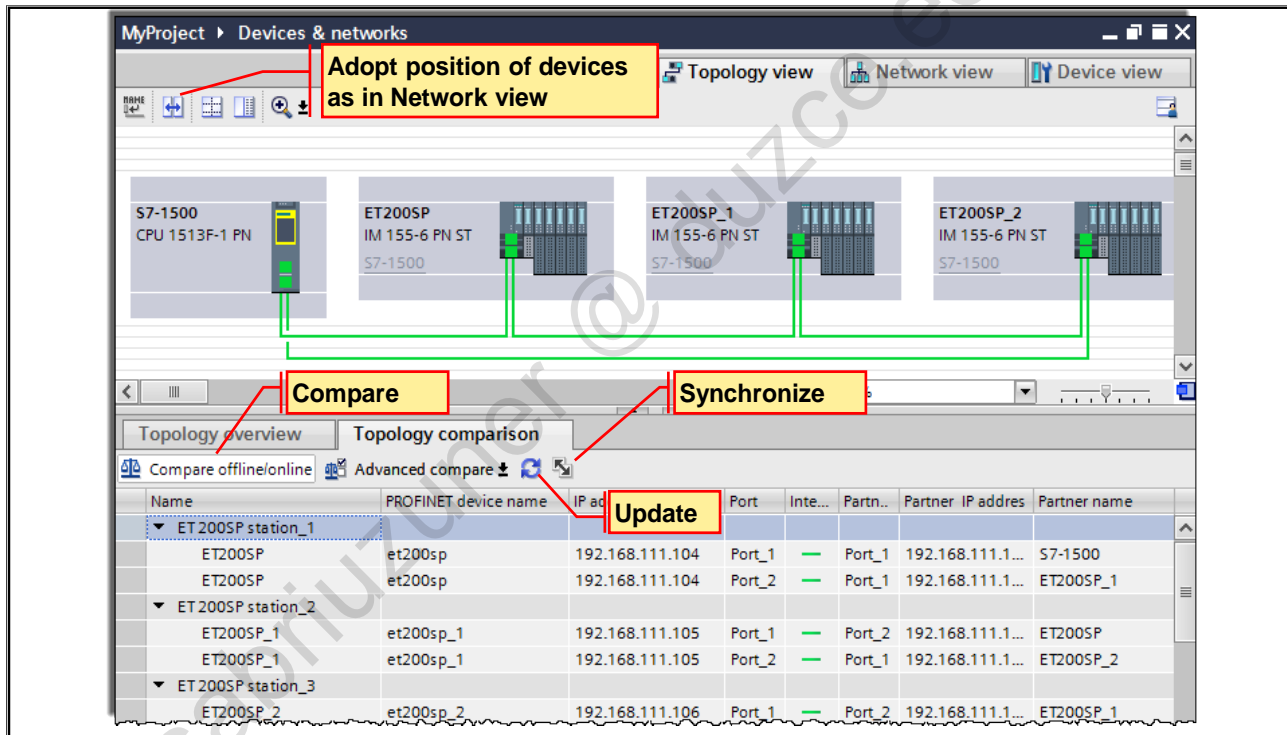
The ring structure is the result of connecting the ends of a line structure.

An internal interruption of the ring at a switch ensures that data packages do not circulate. With an interruption at another location, it is automatically closed.

The redundant path should not be combined with the outgoing path.

- Advantages:
  - Additional failure security
- Disadvantages:
  - Increased expenditures for hardware

### 5.12.3. Topology View - Topology comparison



In the topology view, you carry out the following tasks:

- Display Ethernet topology
  - Display all PROFINET devices and passive Ethernet components of the project including ports
  - Display interconnections between the ports
  - Display associated logical networks
  - Display the diagnostic information of all ports
- Configure Ethernet topology
  - Create, change and delete interconnections of ports
  - Rename stations, devices, interfaces, ports
  - Adding PROFINET devices and passive Ethernet components to the project from the Hardware catalog
- Determine differences between the offline and online topology and repair them
  - Carry out offline/online comparisons of Ethernet modules, Ethernet ports and Ethernet port interconnections
  - Adopt topology information existing online in the offline project

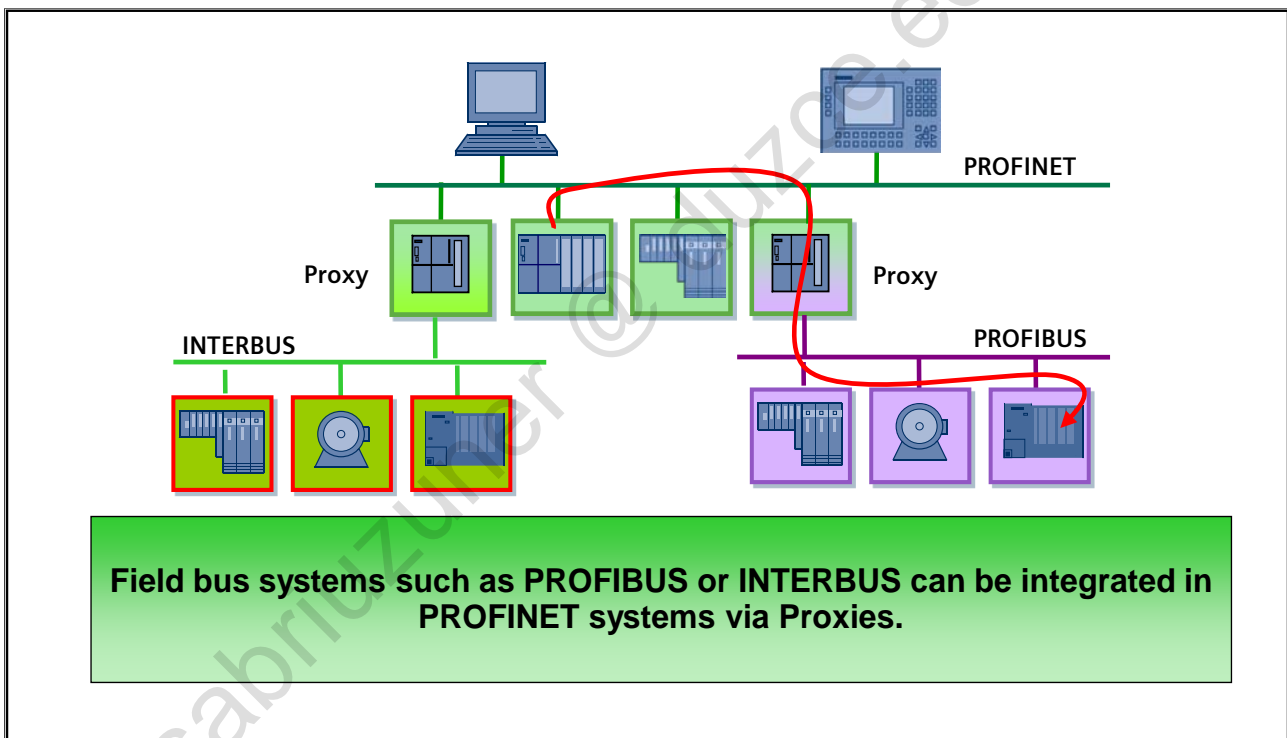
The "Topology Overview" tab offers the following functions:

- Display of configured topology
- Display of the diagnostic status of hardware components

The "Topology Comparison" tab offers the following function:

- Determine and repair the difference between the offline and online topology
  - In the "Actual" column, you specify which OFFLINE-ONLINE differences are to be adopted. All differences to be adopted are activated via the context menu (Apply > Apply all). Then the selected differences can be adopted with the "Synchronize" button.

#### 5.12.4. PROFINET proxy concept

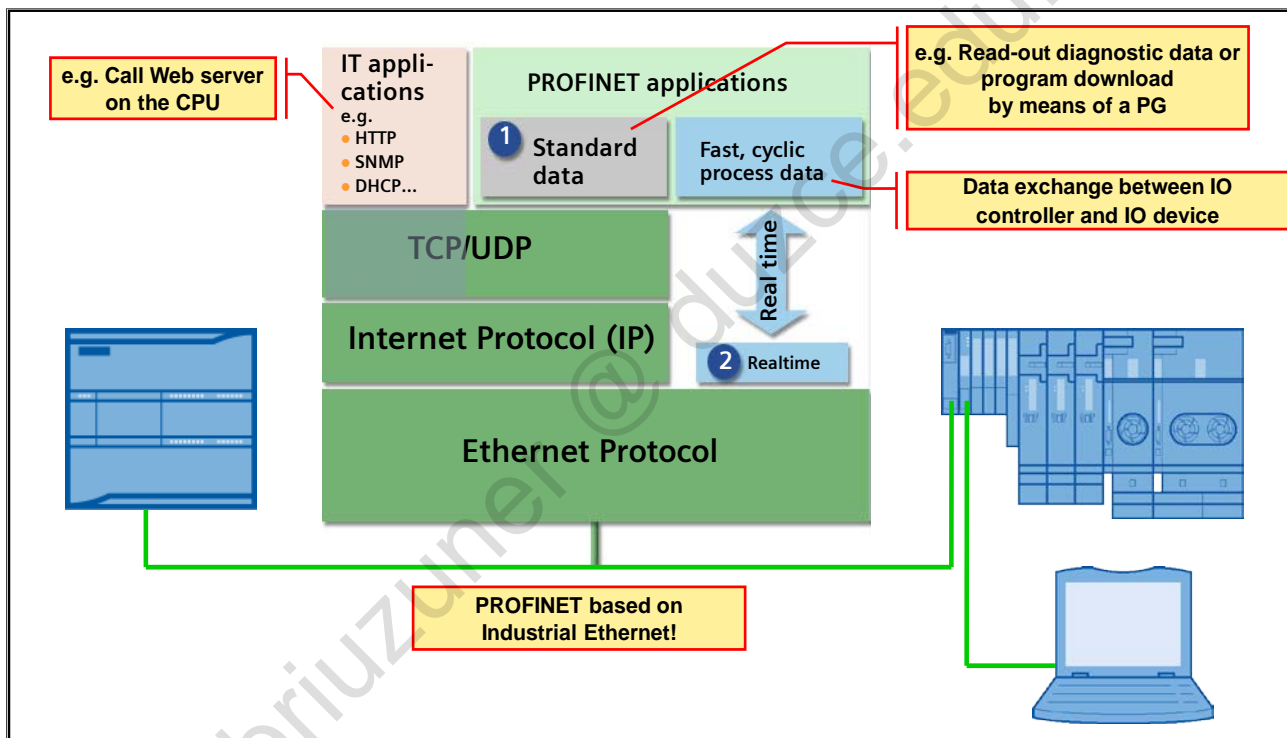


#### PN proxy concept

Using Proxies/Gateways/Links, it is possible to integrate existing bus systems in a PROFINET network. For this, the manufacturers offer different solutions, for example the "IE/PB-Link" from Siemens for the integration of PROFIBUS devices.

From the point of view of PROFINET, a proxy is an IO device. For the connected PROFIBUS slaves, the proxy represents the PROFIBUS master.

### 5.12.5. PROFINET Communications model



#### Real-time channel

To be able to fulfill real-time requirements in automation, an optimized real-time communication channel, the Realtime Channel (RT Channel), was specified in PROFINET. It uses Ethernet (Layer 2) as a base.

The addressing of the data packets does not take place in this case via an IP address, rather by means of the MAC addresses of the participating devices. Such a solution minimizes the throughput times in the communications stack considerably and leads to an increase in performance with regards to the updating rate of automation data.

#### IRT channel

Isochronous Real-time (IRT) as a further development with the following features:

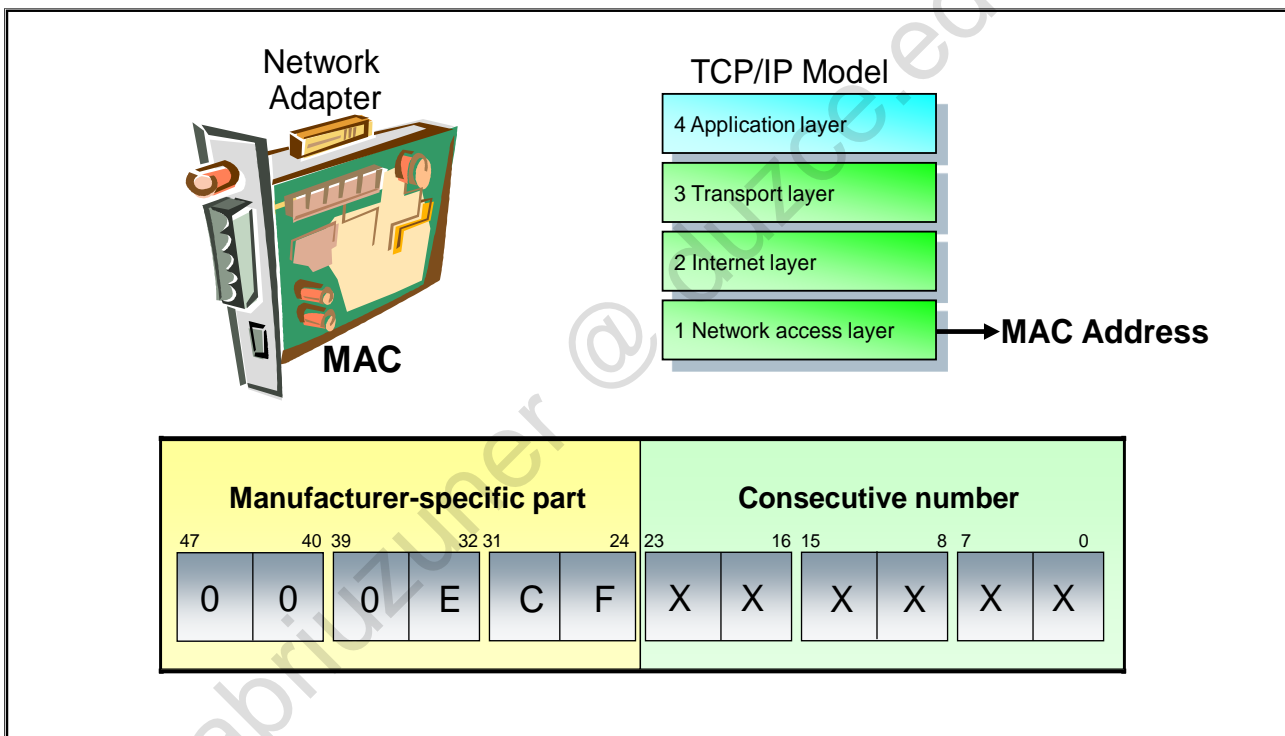
- Clock-synchronous data transmission
- Cycle times <1ms with jitter accuracy <1μs
- Typical field of application is Motion Control

#### IT standards

The design of PROFINET WEB Integration focuses on commissioning and diagnostics. Access to a PROFINET device from the Internet or Intranet is done with standard protocols (for example, http). The data is transmitted in standard formats such as HTML or XML and can be presented with standard browsers such as Opera or Internet Explorer.

This worldwide accessibility makes it easy for the application manufacturer to support the user with commissioning, device diagnostics etc. Access to the data is done via Web servers which are integrated in the modules.

### 5.12.6. The MAC Address



#### MAC address

Every Ethernet device (node) requires, for the identification in the network on Layer 2 of the ISO/OSI model, a unique address as a network access point for the layers above it. For that reason, each device has a fixed, world-wide unique address which is given by the factory. This is called the MAC (Media Access Control) address or MAC for short.

A MAC address has a length of 48 bits and is usually depicted in "canonical representation" (LSB format) (e.g. with "ipconfig /all"). For the transmission of data, it is defined that the least significant bit of an octet (LSB) is sent first.

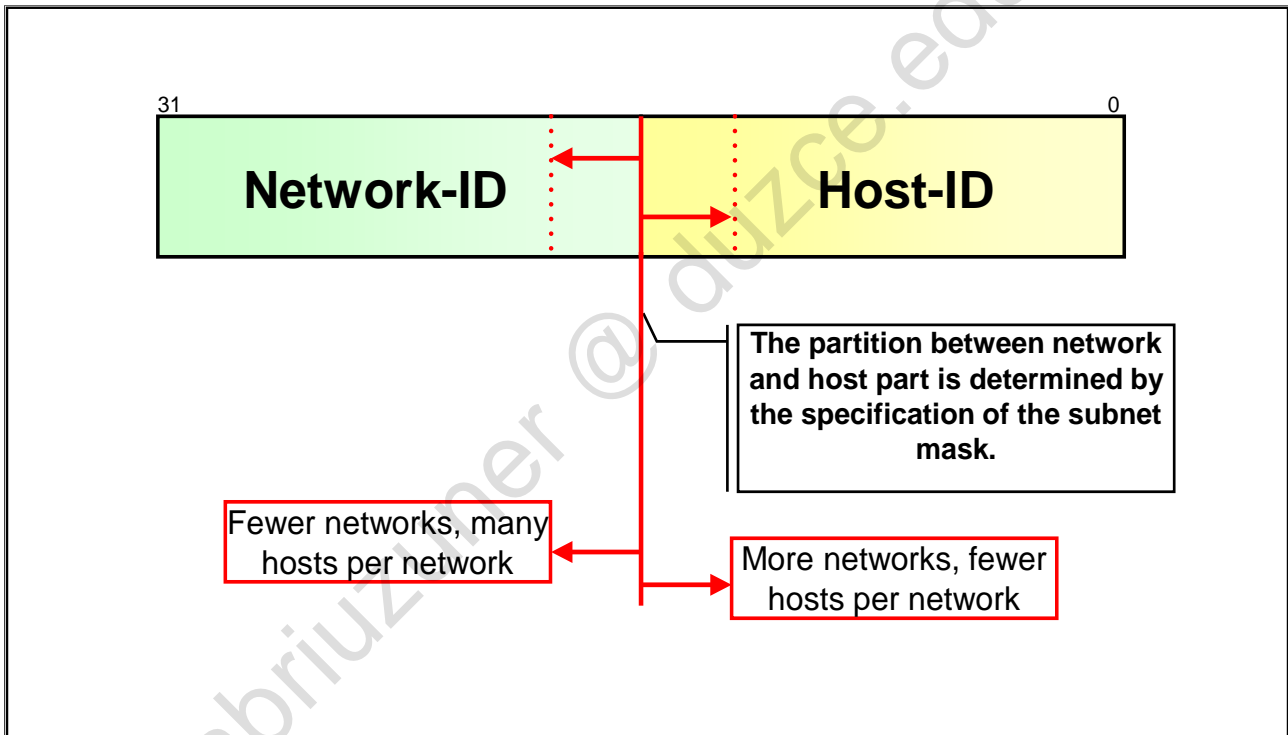
#### Ethernet address

The Ethernet address is 6 bytes long in hexadecimal notation. It is divided into a manufacturer-specific part and a consecutive number. For smaller companies it may make sense to use the PNO Ethernet address. That way, they don't have to apply for their own Ethernet address. The OUI (Organizationally Unique Identifier) of the PNO is 00-0E-CF.

#### Examples

- 00-0E-CF      PROFIBUS User Organization
- 00-0E-8C      Siemens AG A&D ET
- 08-00-06      Siemens AG IT Solutions
- 00-01-E3      Siemens AG
- 00-0E-F0      Festo AG & Co. KG

### 5.12.7. The partitioning of the IP Address



#### The partitioning of the IP address

The IP address is divided into a device part (Host-ID) and a network part (Network-ID). Originally, 5 different network classes were defined worldwide (A,B,C,D,E). For the network classes A to C, it was uniquely defined which part of the IP address represents the Network-ID and which part represents the Host-ID.



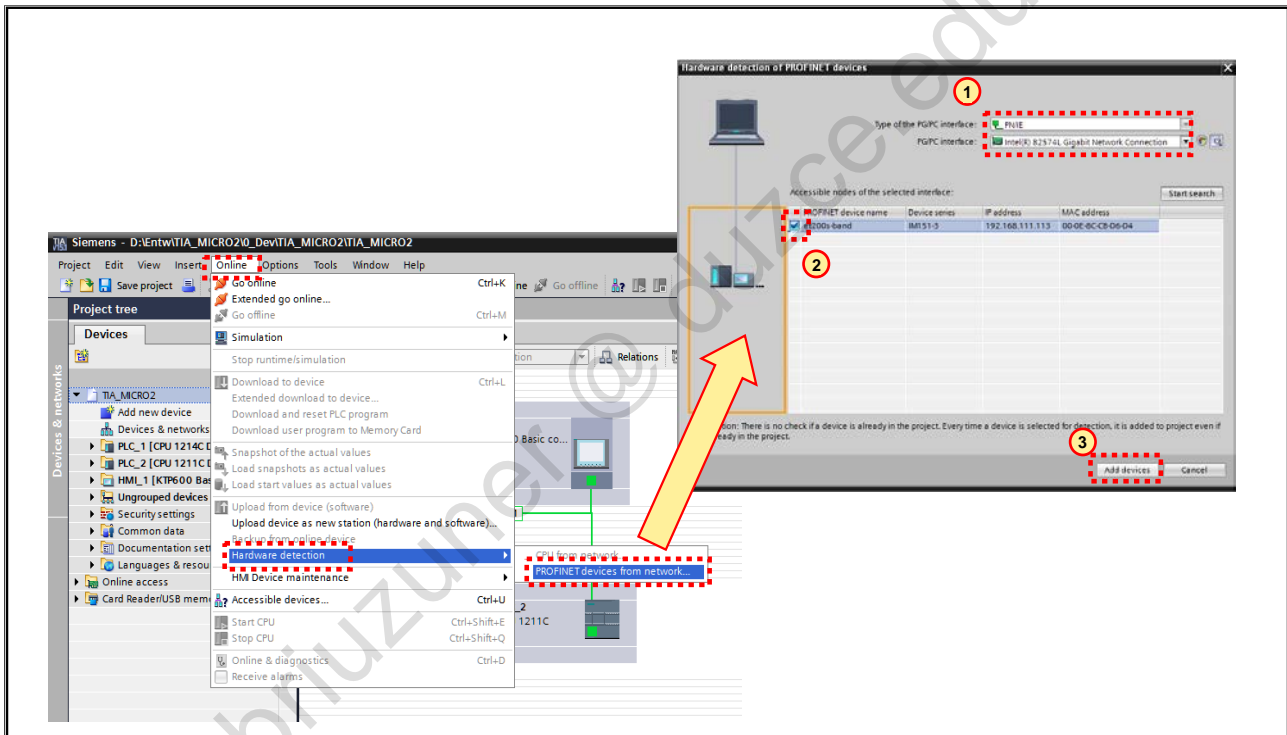
#### Attention!

Today, IP addresses are no longer divided into classes. So that the partition between Network-ID and Host-ID can still be determined, a subnet mask is also specified.

#### Network size

If you shift the partition between Network part and Host part to the left, you have fewer networks with many devices per network. If, however, you shift it to the right, you have many networks with few devices per network.

### 5.12.8. Detecting decentral devices automatically



#### Automatic detection

Via the function "Online>Hardware detection>PROFINET devices from network..." you can automatically detect IO-devices. This is especially interesting for modular devices, since the firmware of the individual modules is inserted correctly here.

# Contents

<b>6.</b>	<b>Introduction to industrial communication.....</b>	<b>6-2</b>
6.1.	Objectives .....	6-2
6.2.	Task Description: Creating an "ISO-on-TCP" connection.....	6-3
6.3.	S7-1200 ethernet communication services in the ISO/OSI communication model .....	6-4
6.3.6.	Data flow-oriented and message-oriented communication .....	6-5
6.3.7.	Combined blocks for the connection programming .....	6-6
6.3.8.	Connection parameterization via block properties (sending station with TSEND_C).....	6-8
6.3.9.	Parameterized send block TSEND_C .....	6-10
6.3.10.	Connection parameterization via block properties (receiving station with TRCV_C) .....	6-12
6.3.11.	Parameterized receive block TRCV_C .....	6-13
6.4.	Task description: Program CPU-CPU communication and send 200 Bytes of data .....	6-14
6.4.1.	Exercise 1: Preparing the CPU 1211C .....	6-15
6.4.2.	Exercise 2: Calling TSEND_C ("PLC_1": "FC_Send") .....	6-16
6.4.3.	Exercise 3: Calling "FC_Send" .....	6-17
6.4.4.	Exercise 4: Calling TRCV_C ("PLC_2": "FC_Receive") .....	6-18
6.4.5.	Exercise 5: Function test.....	6-20
6.5.	Additional Information .....	6-21
6.5.1.	UDP Communication .....	6-22
6.5.2.	TCP Communication .....	6-23
6.5.3.	ISO-on-TCP communication .....	6-24
6.5.4.	S7 Communication.....	6-25
6.5.5.	Connections .....	6-26
6.5.6.	Connection resources .....	6-27
6.5.7.	Diagnosing the Open User Communication .....	6-28



## 6. Introduction to industrial communication

### 6.1. Objectives

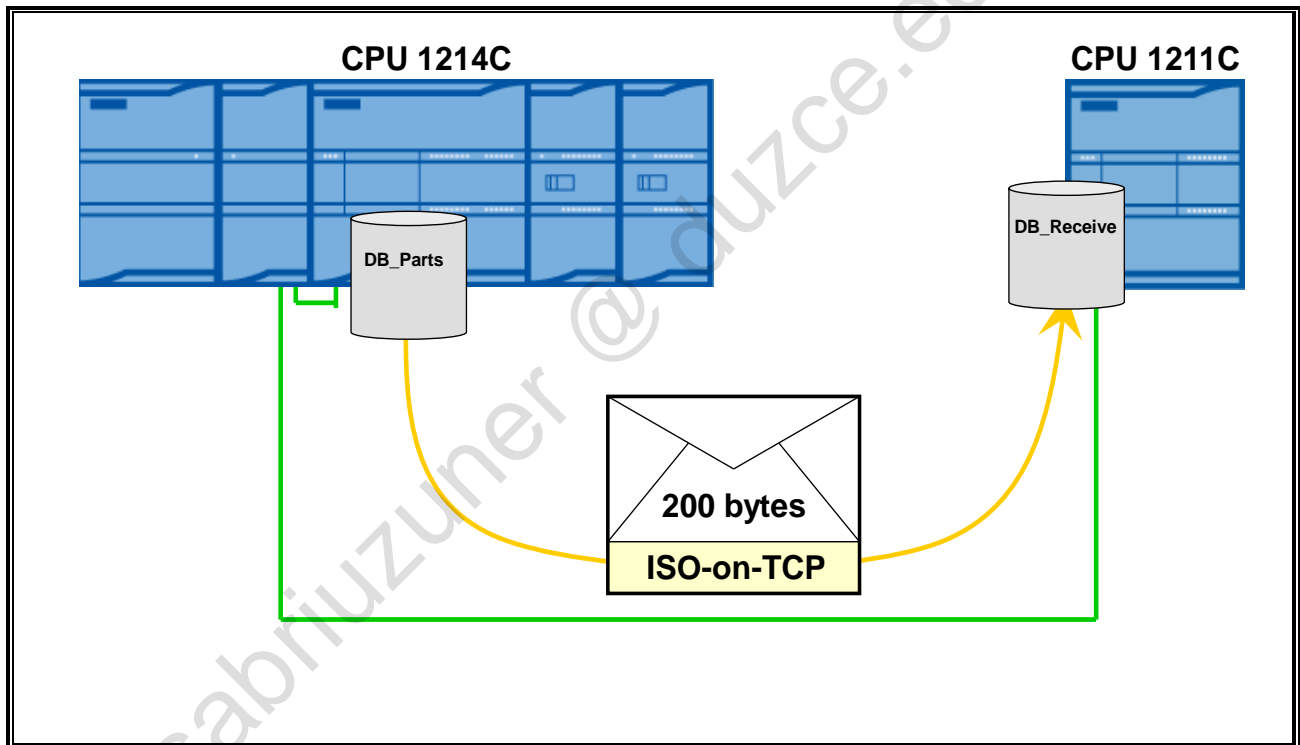
**At the end of the chapter the participant will ...**

- ... understand the principle of CPU-CPU communication.
- ... be familiar with the features of the transport protocol "ISO-on-TCP".
- ... understand the difference between packet-oriented and data flow-oriented communication.
- ... be able to create a communications connection between 2 CPUs.

#### Objectives

In this chapter, the industrial communication via Ethernet between two S7-1200 CPUs is dealt with. The available communication services are compared. In a concluding exercise, an ISO-on-TCP connection between the two CPUs of the training area is programmed.

## 6.2. Task Description: Creating an "ISO-on-TCP" connection

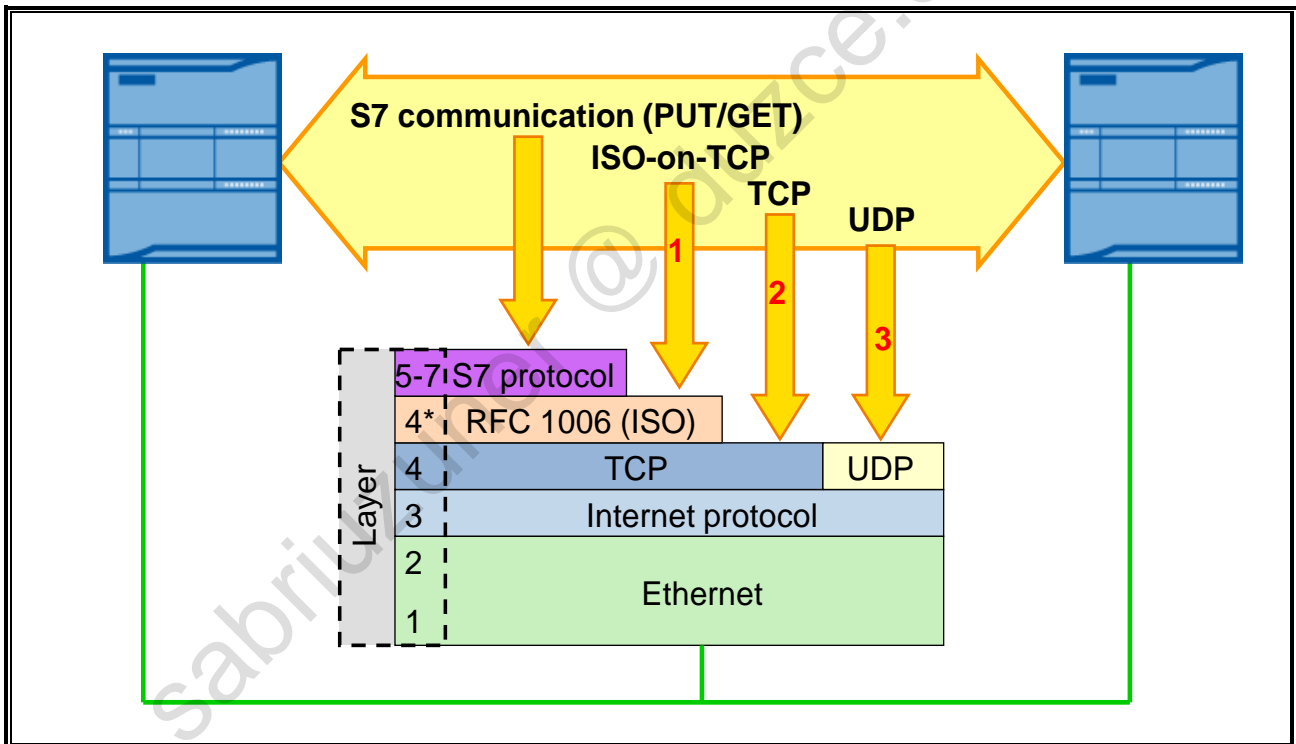


### Task description

An ISO-on-TCP connection between the CPUs of your training area is to be programmed. Via this connection, the ARRAY variable "DB\_Parts".PartWeights is to be sent from the controller "PLC\_1" to the still to be configured controller "PLC\_2" and stored there in the DB "DB\_Receive" in the ARRAY variable "Receivebuffer".

To minimize data traffic, sending is not to be continuous. Instead, sending is to occur under the same conditions as the saving of weight values in "DB\_Parts".

### 6.3. S7-1200 ethernet communication services in the ISO/OSI communication model



#### ISO/OSI model

Communication tasks are divided for international comparison in 7 layers according to the ISO/OSI model. Included are also, among other things, the types of communication shown in the picture which are supported by the S7-1200.

Every layer has an exactly defined task area and, in each case, a defined interface to the higher-level and the subordinate layer.

#### Abbreviation

ISO:

International Organization for Standardization

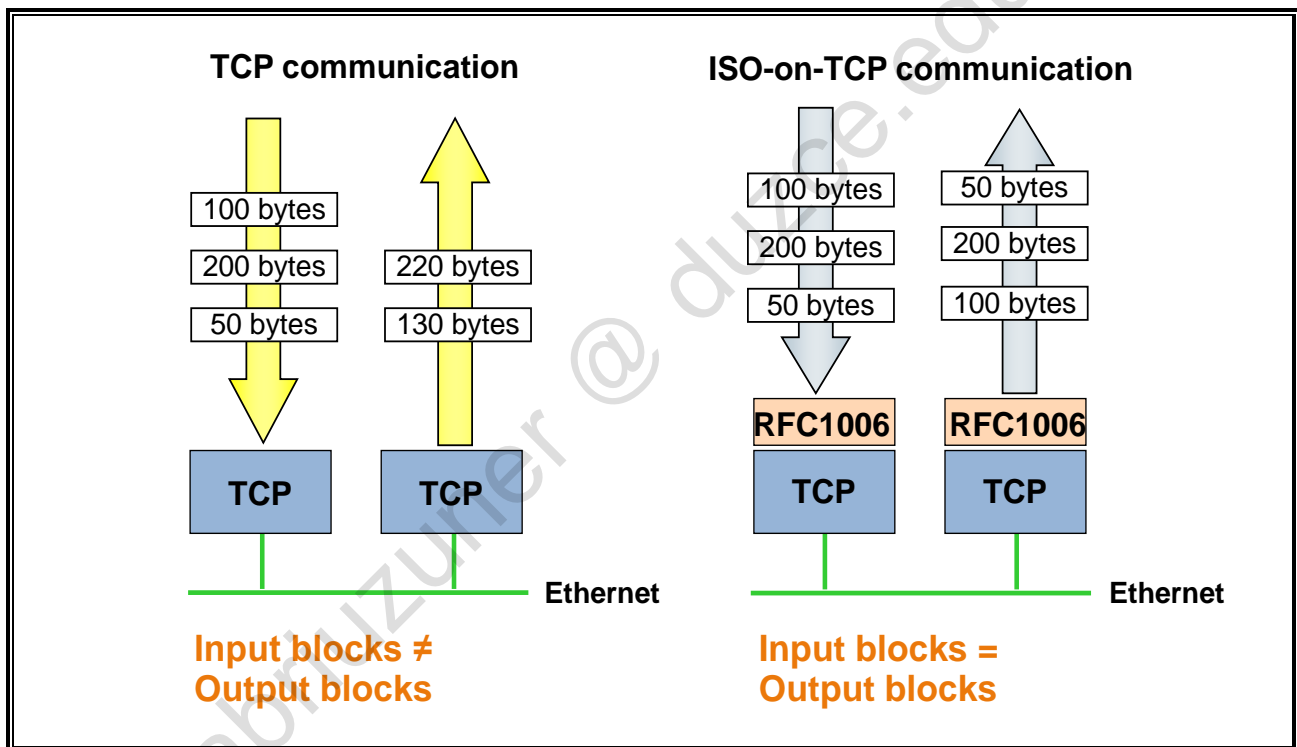
OSI: Open Systems Interconnection Reference Model

#### Ethernet communication services

For communication via (industrial) ethernet, there are various services in the SIMATIC environment. These differ about:

- Data security
- Amounts of data
- Data handling
- Routing capability and
- Engineering effort

### 6.3.6. Data flow-oriented and message-oriented communication



#### Data flow-oriented and packet-oriented communication

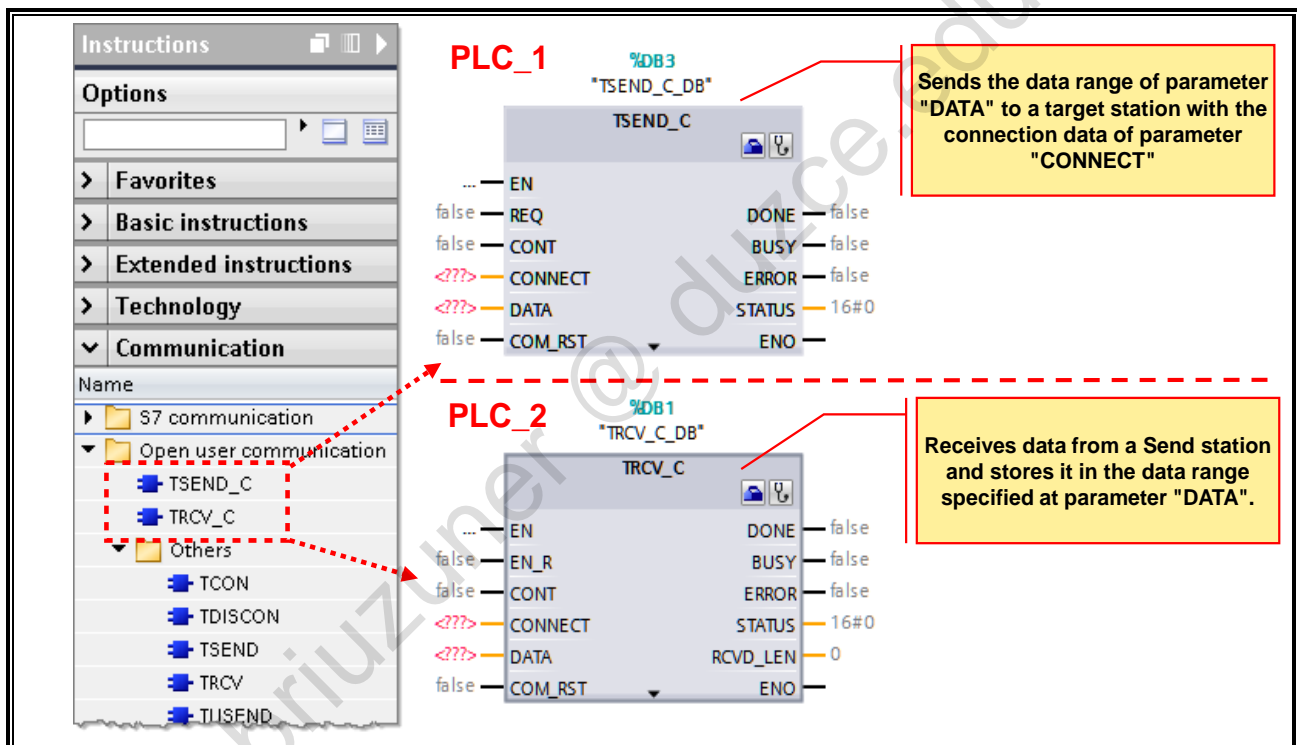
When data is transferred with the Transmission-Control-Protocol (TCP), the transmission takes place in the form of a data flow. Neither information on length nor information about the beginning and end of a message is transferred. The receiver, however, cannot recognize where a message ends in the data flow and where the next one begins in the data flow. A read task of the receiver thus only supplies as much data as is currently found in the receive buffer. This means that possibly more than one data block can be found in the receive buffer. This process is well suited for communicating with third-party systems or computer systems.

#### Behavior of the RFC 1006 protocol expansion

In most automation applications it is, however, essential to work message-oriented. Self-contained message blocks are sent via a connection which is also recognized as such by the receiver. In order to ensure this, RFC 1006 specifies which information (in the form of a header) must be added to the data to be transferred.

RFC 1006 thus provides applications which are based on the data flow-oriented TCP protocol with a message-oriented transmission.

### 6.3.7. Combined blocks for the connection programming



#### Open User Communication

If the integrated Ethernet interface of the CPU is used for the Ethernet communication, the so-called "Open User Communication" is used.

Included in the "open" communication services, that is those whose inner structure is open, are TCP, UDP and ISO-on-TCP. Connections between SIMATIC controllers which use one of these services are not configured in the Network view of STEP7 (such as connections to HMI devices) but are programmed. For this, there are various blocks available in the Instruction Catalog.

#### Connection-oriented and connectionless services

Connection-oriented services, this includes TCP and ISO-on-TCP, first establish a connection to the communication partner and then send the data (also bidirectional). If the transmission process is completed, the connection is disconnected. All data is acknowledged by the receiver. The sender resends all unacknowledged telegrams. This is comparable to telephoning with a telephone. First, a connection is established (dial number + pick up), then, information is exchanged, and, in the end, you hang up, that is, the connection is disconnected.

Connectionless services, such as UDP, send their data without first establishing a connection, like a "walkie-talkie" (blocks: TUSEND / TURCV). It can therefore not be ensured whether the data is received by the receiver. The advantage lies here in the speed, since less administrative data for flow control must be sent and interpreted.

### Combined connection blocks

Connection-oriented communication services can be used in two ways:

- Single blocks  
With the single blocks, targeted connection actions can be executed.
  - TCON / TDISCON: connect and disconnect a connection
  - TSEND / TRCV: send or receive data after establishing a connection
- Combined connection blocks
  - TSEND\_C: Connect a connection (when Active connection establishment is active) , Send data and disconnect the connection (when Active connection establishment is active)
  - TRCV\_C: Connect a connection (when Active connection establishment is active), Receive data and disconnect the connection (when Active connection establishment is active)

With combined connection blocks, a connection to the communication partner is established, data is sent/received, and the connection is disconnected with just one call.

### 6.3.8. Connection parameterization via block properties (sending station with TSEND\_C)

The screenshot shows the 'Connection parameter' dialog box for the TSEND\_C block. The 'General' section contains the following fields:

- Local End point:** PLC\_1 [CPU 1214C DC/DC/DC]
- Partner End point:** PLC\_2 [CPU 1211C DC/DC/DC]
- Interface:** PLC\_1, PROFINET-Schnittstelle[X1 : PN(LAI)] / PLC\_2, PROFINET-Schnittstelle\_1[X1 : PN(I)]
- Subnet:** PN/IE\_1 / PN/IE\_1
- Address:** 192.168.111.112 / 192.168.111.114
- Connection type:** ISO-on-TCP
- Connection ID (dec):** 1
- Connection data:** PLC\_1\_Send\_DB / PLC\_2\_Receive\_DB
- Active connection establishment:**  Active connection establishment

The 'Address details' section contains:

- Local TSAP:** TSAP (ASCII): PLC\_1, TSAP ID: 50.4C.43.5F.31
- Partner TSAP:** TSAP (ASCII): PLC\_2, TSAP ID: 50.4C.43.5F.32

Callout boxes on the right side of the dialog provide additional information:

- Select the target station:** Points to the 'Partner' end point dropdown menu.
- IP addresses of stations:** Points to the 'Address' fields.
- Select the connection type:** Points to the 'Connection type' dropdown menu.
- DB for connection data in target station (is automatically generated):** Points to the 'Partner' 'Connection data' dropdown menu.
- DB for connection data in sending station (is automatically generated):** Points to the 'Local' 'Connection data' dropdown menu.
- Establishment of TSAPs:** Points to the 'Local TSAP' and 'Partner TSAP' fields.

#### Connection parameterization via block properties

The TSEND\_C block works connection-oriented, that is, a partner CPU must first be configured. After the block is called, this can be done very easily in the Inspector window under Properties > Configuration > Connection parameter.

#### Connection parameters

- **End point**  
Here, the partner CPU is configured. It can be in the same project (here: "PLC\_2"), or a "not specified" partner is created. In both cases, the addressing of the partner occurs via its IP address.
- **Address**  
Here, the IP address of the partner is specified. If a specified partner was selected before, its IP address is automatically adopted. For not specified partners, it must be entered manually.
- **Connection type**  
Via the connection type, a communication service is selected and thus the properties of the communication.
- **Connection ID (dec)**  
Via the connection ID, the number of the connection within the CPU is specified. Depending on the CPU used, several simultaneous connections are mastered. The connection ID must be unique within the CPU. The exact number of simultaneously possible Ethernet connections can be found in the manual.
- **Connection data**  
The connection data, which the "TSEND\_C" block accesses, is stored in a DB. This DB contains the configured information on the connection type and the partner CPU (IP address etc.). The DB is automatically created when "New" is selected in the field "Connection data".

### Address details

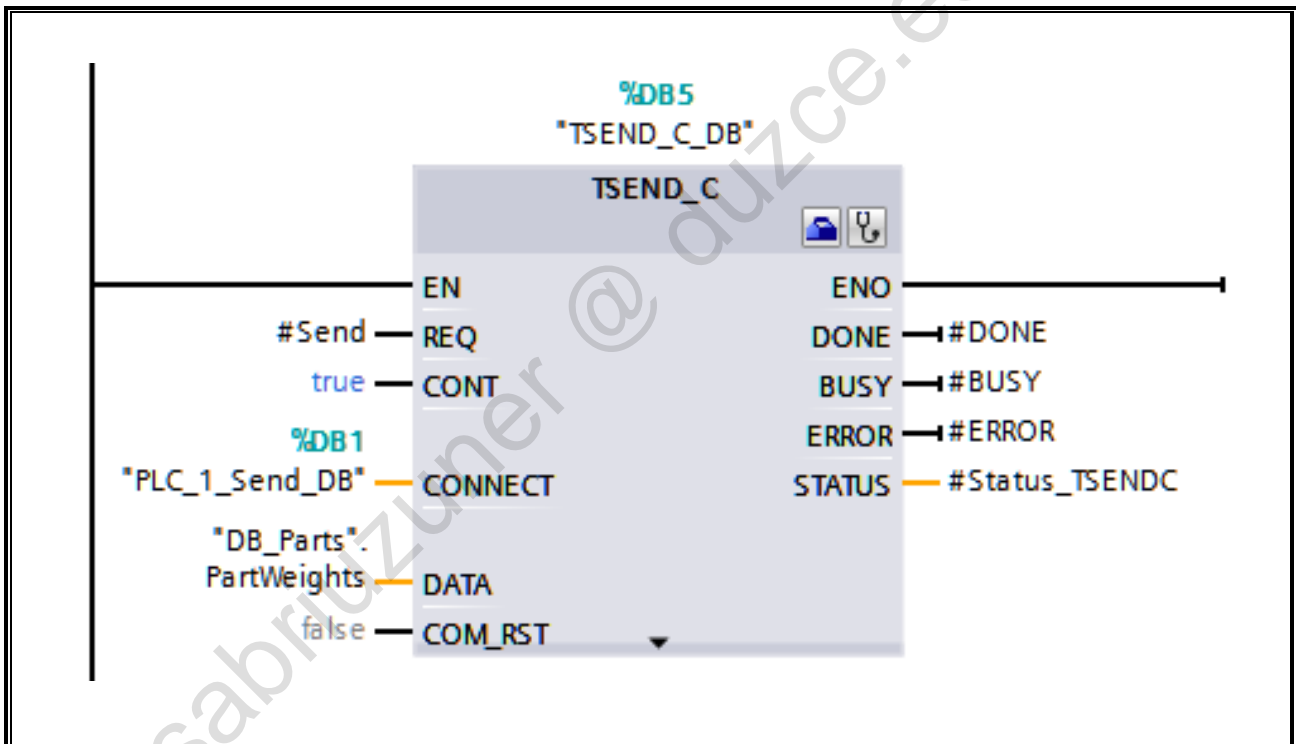
- **TSAP (ASCII)**  
The Transport Service Access Point (TSAP) is used for ISO-on-TCP communication in order to address the transmitted data at the receiver. In the network, the receiver is addressed by means of the MAC address. At the receiver, the received data is first stored in the receive buffer of the Ethernet interface and then fetched from there by the operating system of the CPU. The application within the CPU is addressed via the TASP-ID.

This type of addressing is comparable to a multiple family dwelling with a collective mailbox and a caretaker who distributes the mail. In order to now address an occupant, the house number (MAC address) must first be specified. The letter (data) first lands in the mailbox (receive buffer of the Ethernet interface). So that the caretaker can deliver the letter, the name of the occupant (TSAP-IP) must also be on the letter.

- **TSAP-ID**  
The TASP-ID is automatically generated from the entered TSAP.



### 6.3.9. Parameterized send block TSEND\_C



#### TSEND\_C

The TSEND\_C block processes send tasks as follows:

1. Establish connection to the communication partner
2. Send data at parameter DATA
3. Disconnect connection

#### Parameterized Send Block TSEND\_C

The TSEND\_C block has several parameters which are explained in the following.

- REQ  
If a rising edge is detected at input REQ, a send task is started
- CONT  
The input CONT controls the communication connection  
FALSE: The connection is disconnected  
TRUE: The connection is established
- LEN  
The parameter LEN specifies the maximum number of bytes which can be sent with the task.  
If purely symbolic values are specified at parameter DATA, the parameter LEN must have the value "0"
- CONNECT  
In order to parameterize the communication connections for TCP, UDP and ISO-on-TCP, a connection describing DB with a fixed structure is used. The data structure contains the necessary parameters which are required to establish the connection. The connection describing DB is automatically created for a new connection by the connection parameterization of the Open User Communication when using the instructions TSEND\_C, TRCV\_C or TCON

- **DATA**  
The send data range is specified via the parameter DATA. The addressing takes place via a pointer to the send range which contains the address and the length of the data to be sent (for example, "P#DB80.DBX0.0 byte 20" (pointer to a data range of 20 bytes in DB80, beginning from bit 0.0)) or symbolically, in order to address, for example, ARRAYS or structures (for example, ""DB\_Parts".WeightStore" (pointer to the structure variable "WeightStore" in "DB\_Parts"))
- **COM\_RST**  
Causes a restart of the instruction:  
FALSE: Irrelevant  
TRUE: Complete restart of the instruction whereby the existing connection is disconnected, and a new connection is established
- **DONE**  
The status parameter DONE indicates the processing status of the current send task  
FALSE: Task has not yet started or is still being processed  
TRUE: Task was executed without error
- **BUSY**  
The status parameter BUSY indicates whether the current call of the block has already been completed or is not active since "TSEND\_C" is executed asynchronously. As long as BUSY = "TRUE", no new send task is accepted
- **ERROR**  
The status parameter ERROR indicates with "TRUE" whether errors occurred during the send task. Error details can be queried at the parameter STATUS
- **STATUS**  
The parameter STATUS delivers a 2-byte HEX code which contains information on the send state or errors that have occurred

### 6.3.10. Connection parameterization via block properties (receiving station with TRCV\_C)

#### Connection parameterization via block properties

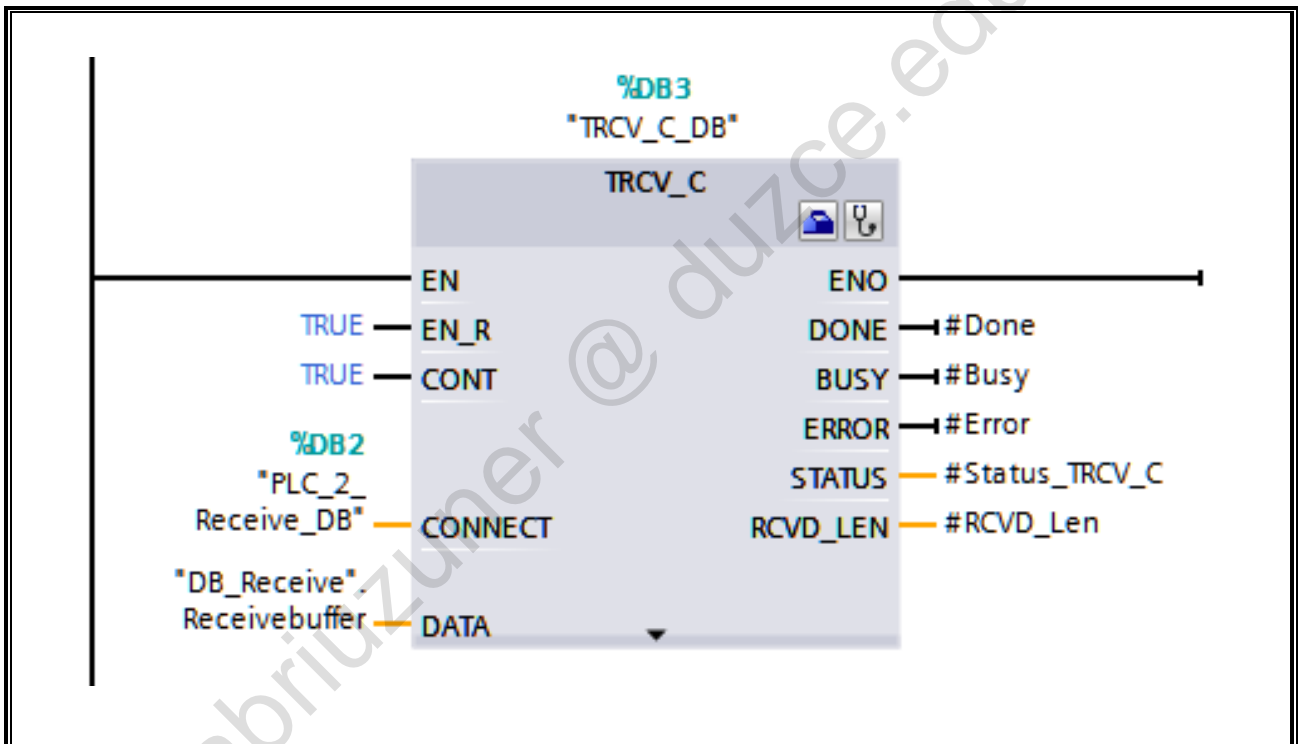
The TRCV\_C block, as well, works connection-oriented, that is, a partner CPU must first be configured. After the block is called, this can be done very easily in the Inspector window under Properties > Configuration > Connection parameter.

If the partner CPU was already programmed beforehand, and both stations are in the same project, only the "connection data" DBs still must be selected. The rest of the connection parameters are automatically entered.

#### Connection Parameters

See page "Parameterized Send Block TSEND\_C"

### 6.3.11. Parameterized receive block TRCV\_C



#### TRCV\_C

The TRCV\_C block processes send tasks as follows:

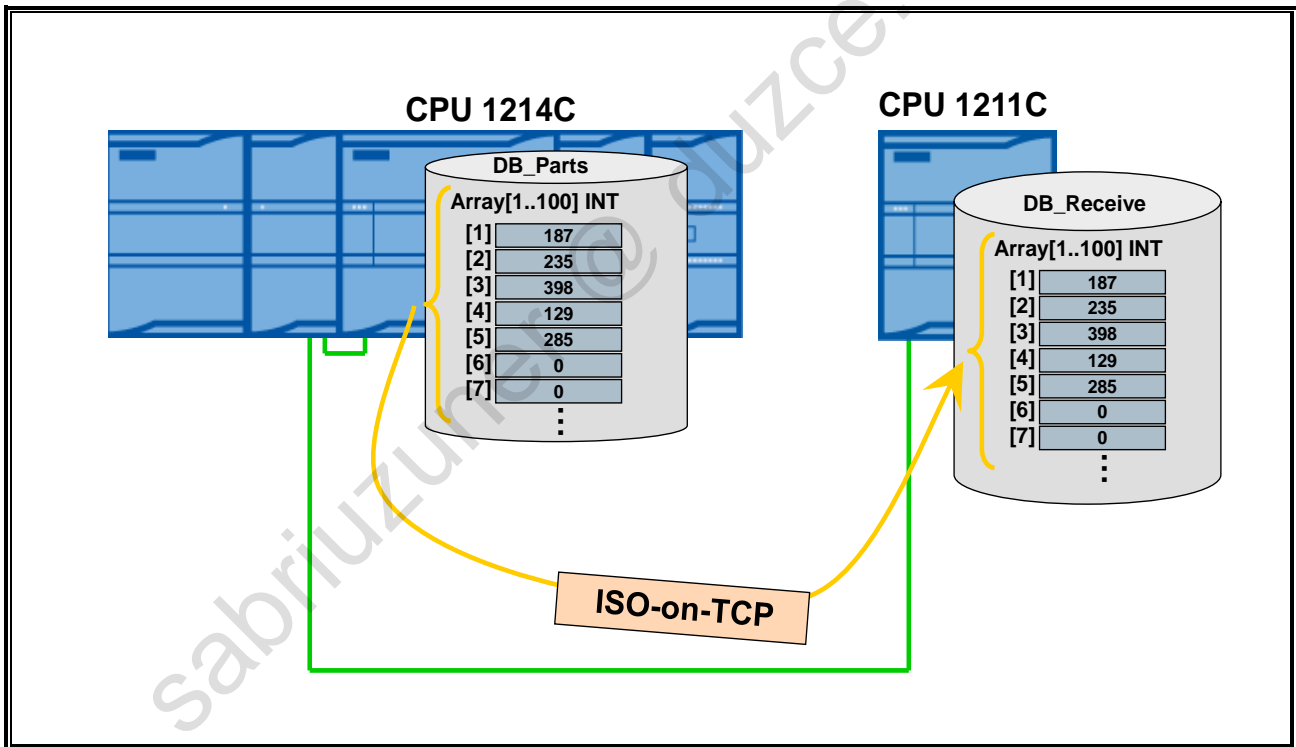
1. Establish connection to the communication partner
2. Receive data when EN\_R = TRUE. When receiving data, the parameter CONT must have the value TRUE in order to maintain the existing connection
3. Disconnect connection

#### Parameterized send block TRCV\_C

In the following, the differences to the "TSEND\_C" block are presented:

- EN\_R  
The receiving of data is enabled (EN\_R = TRUE) via the parameter EN\_R
- DATA  
At the parameter DATA, the data range is specified in which the received data is to be stored
- RCVD\_LEN  
This parameter outputs the number of received bytes after successful receipt

## 6.4. Task description: Program CPU-CPU communication and send 200 Bytes of data



### Situation up until now

The individual weight values of the produced, valid parts are stored in the ARRAY variable "DB\_Parts".PartWeights whenever the part has passed through the light barrier.

### Task description

A connection between the CPUs of your training area is to be programmed. Via this connection, the ARRAY variable "DB\_Parts".PartWeights is to be sent from the controller "PLC\_1" to the still to be configured controller "PLC\_2" and stored there in the DB "DB\_RECEIVE" in the ARRAY variable "Receivebuffer".

To minimize data traffic, sending is not to be continuous. Instead, sending is to occur under the same conditions as the saving of weight values in "DB\_Parts".

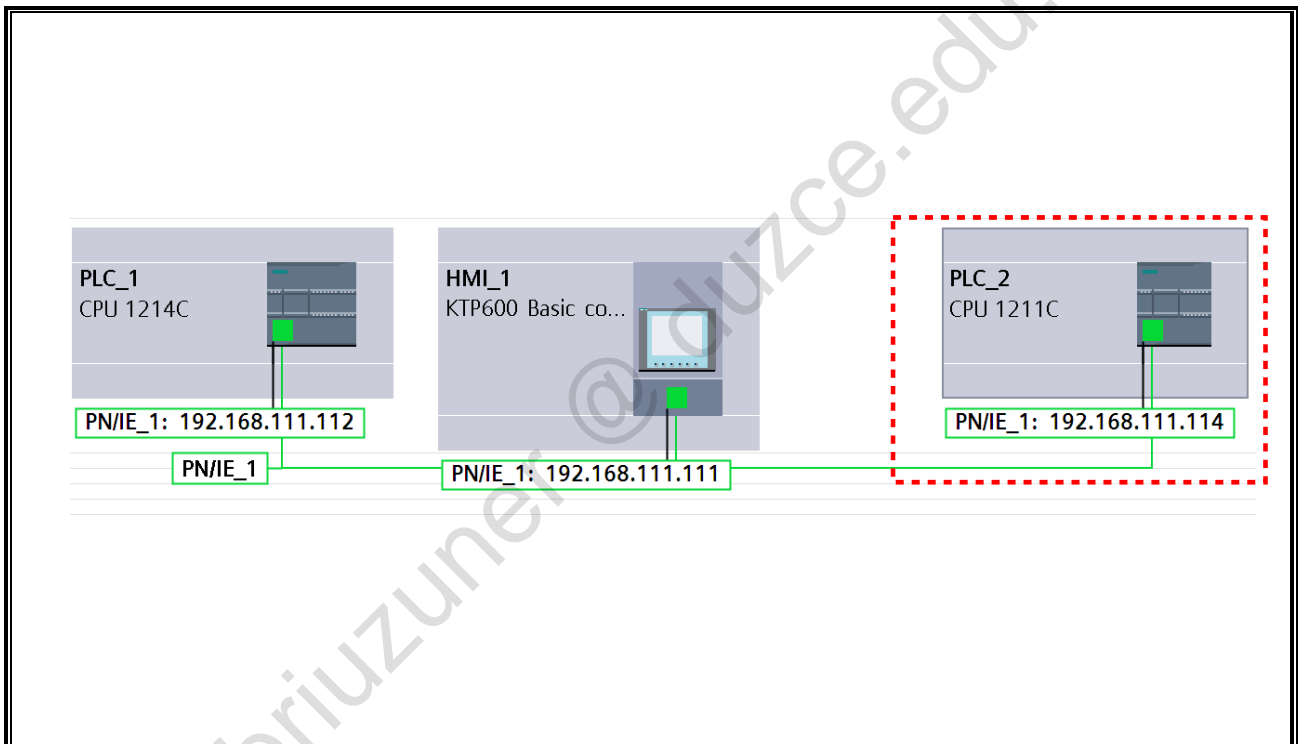
### Blocks

- PLC\_1  
FC\_Send (FC30) → Call in FC\_Count (FC18)
- PLC\_2  
FC\_Receive (FC31) → Call in MAIN (OB1)  
DB\_RECEIVE (DB31)

### Note

The "PLC\_2" controller is given the IP address 192.168.111.114

### 6.4.1. Exercise 1: Preparing the CPU 1211C



#### Task

Configure and network the so far unused controller of your training area.

#### What to do

1. Reset the controller of your training area that hasn't been used up until now to the factory settings to establish a defined initial state.
2. Add a CPU of the type S7-1211C to your project and assign the name "PLC\_2".
3. In the Network view of your project, network the new station with the rest of the components.
4. Assign "PLC\_2" the IP address 192.168.111.114.
5. Activate the clock memory (MB10) in "PLC\_2".
6. Generate a new DB with the name "DB\_Receive" and in it create the ARRAY variable "Receivebuffer" (ARRAY[1..100] of INT).

DB_Receive		Name	Data type
1	Static		
2	Receivebuffer	Array[1..100...	
3	Receivebuffer[1]	Int	
4	Receivebuffer[2]	Int	
5	Receivebuffer[3]	Int	
6	Receivebuffer[4]	Int	
7	Receivebuffer[5]	Int	
8	Receivebuffer[6]	Int	
9	Receivebuffer[7]	Int	
10	Receivebuffer[8]	Int	
11	Receivebuffer[9]	Int	

7. Download the modified hardware and software into the controller.

## 6.4.2. Exercise 2: Calling TSEND\_C ("PLC\_1": "FC\_Send")

Interface		
	Name	Data type
1	Input	
2	Send	Bool
3	Output	
4	InOut	
5	Temp	
6	Done	Bool
7	Busy	Bool
8	Error	Bool
9	Status_TSEND_C	Int
10	Constant	
11	Return	

### Task

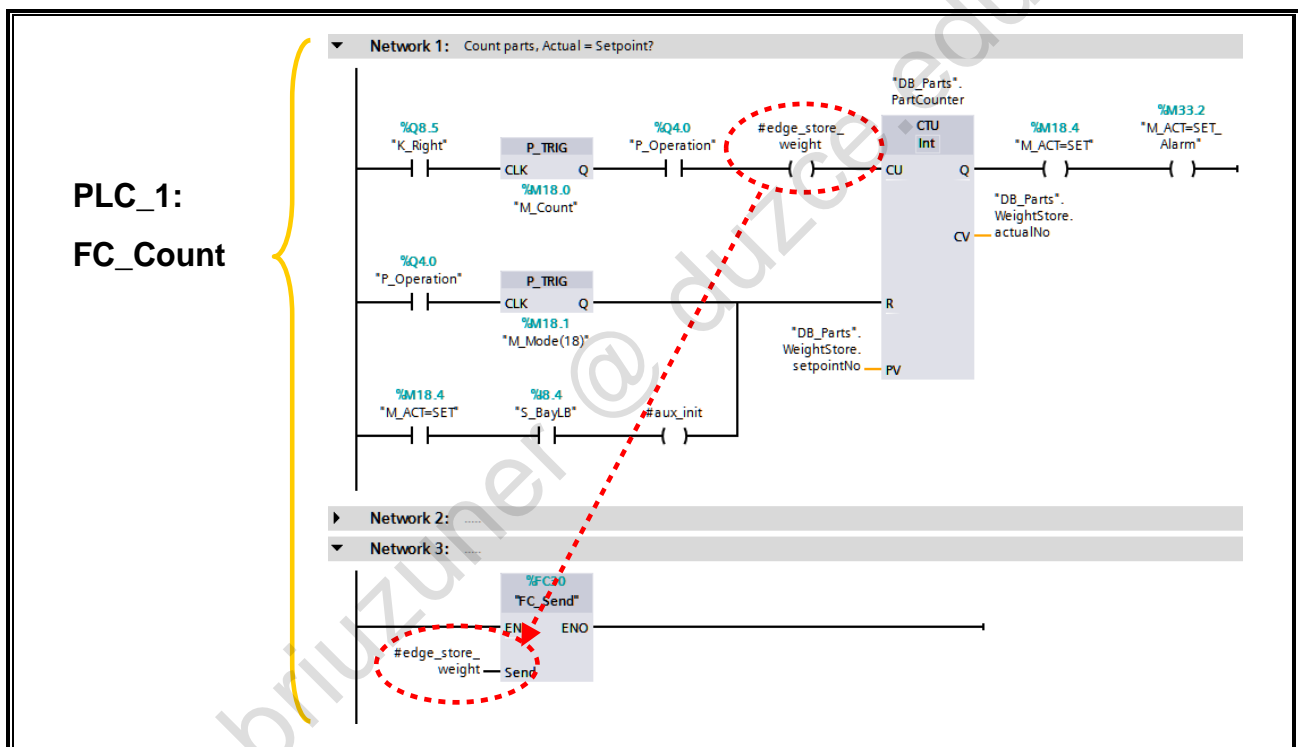
In PLC\_1, create a new function "FC\_Send" and in it program the call of "TSEND\_C".

### What to do

1. Add the new function "FC\_Send" to your user program.
2. Call the block "TSEND\_C".  
Instructions → Communication → Open User Comm. → TSEND\_C
3. Implement the connection parameter settings as shown in the following:

4. Parameterize the block call as shown in the picture above.

## 6.4.3. Exercise 3: Calling "FC\_Send"



## Task

Call the new block "FC\_Send" in "FC\_Count" and assign the parameter "Send" the same RLO as the counter input "CU".

## What to do

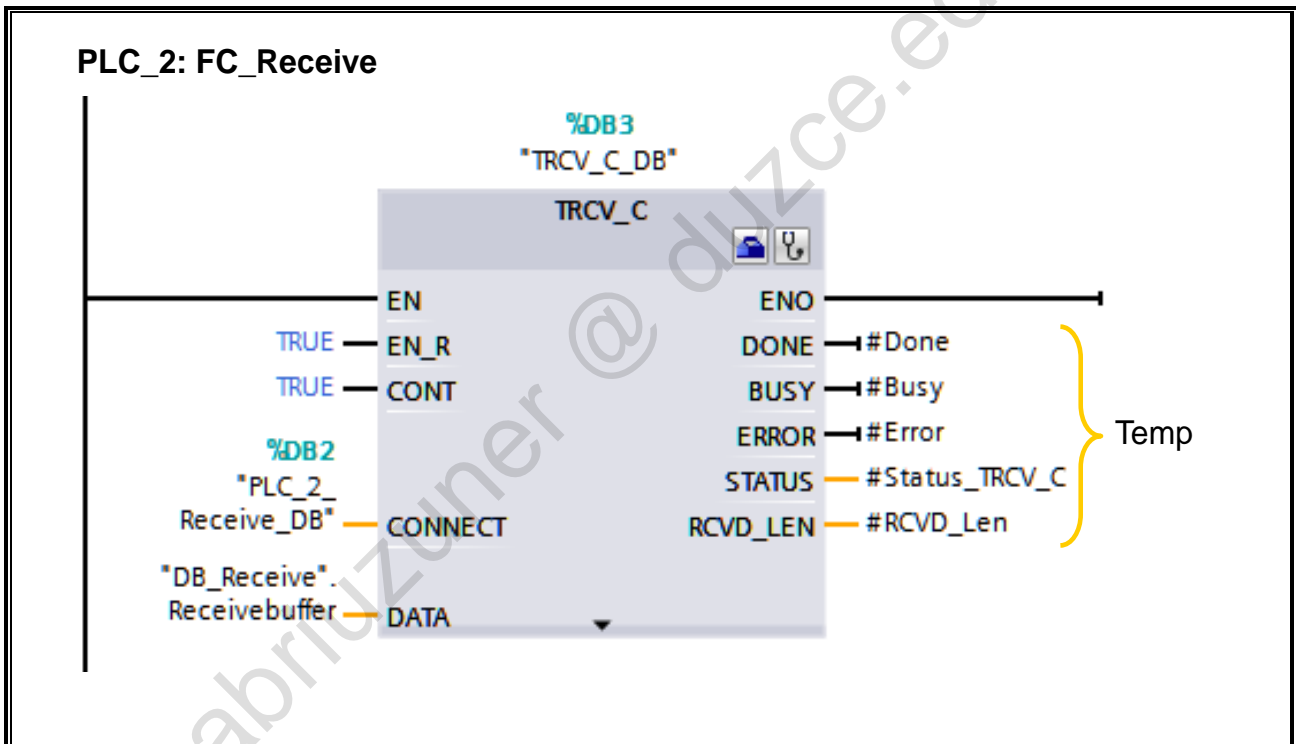
1. Open "FC\_Count" and call "FC\_Send"
2. Assign the input "Send" the RLO of the counter input "CU"
3. Download the entire user program into the controller "PLC\_1"



In the call of "FC\_Send" pay attention to the call sequence within "FC\_Count"! In NW2, "FC\_Ind\_Weight" is called. Only after the current part weight has been stored in DB\_Parts, can "PartWeights" be sent to PLC\_2.



#### 6.4.4. Exercise 4: Calling TRCV\_C ("PLC\_2": "FC\_Receive")



#### Task

In PLC\_2, create a new function "FC\_Receive" and in it program the call of "TRCV\_C". Then call "FC\_Receive" in OB1 and transfer the entire user program.

#### What to do

1. Add the new function "FC\_Receive" to your user program
2. Call the block "TRCV\_C"  
Instructions → Communication → Open User Comm. → TRCV\_C
3. Implement the connection parameter settings as shown in the following:

**Connection parameter**

General	
Local	Partner
End point: PLC_2	PLC_1
Interface: PLC_2, PROFINET inte	PLC_1, PROFINET inte
Subnet: PN/IE_1	PN/IE_1
Address: 192.168.111.114	192.168.111.112
Connection type: ISO-on-TCP	
Connection ID (dec): 1	1
Connection data: PLC_2_Receive_DB	PLC_1_Send_DB
<input type="radio"/> Active connection establishment	<input checked="" type="radio"/> Active connection establishment
Address details	
Local TSAP	Partner TSAP
TSAP (ASCII): PLC_2	PLC_1
TSAP ID: 50.4C.43.5F.32	50.4C.43.5F.31

Select  
PLC\_2\_Receive\_DB

4. Parameterize the block call as shown in the picture above
5. Call "FC\_Receive" in OB1
6. Download the entire user program into PLC\_2
7. Save your project

### Result

Due to the TRUE signal at EN\_R, receiving is continuous. The received data is stored in the DB variable "Receivebuffer".

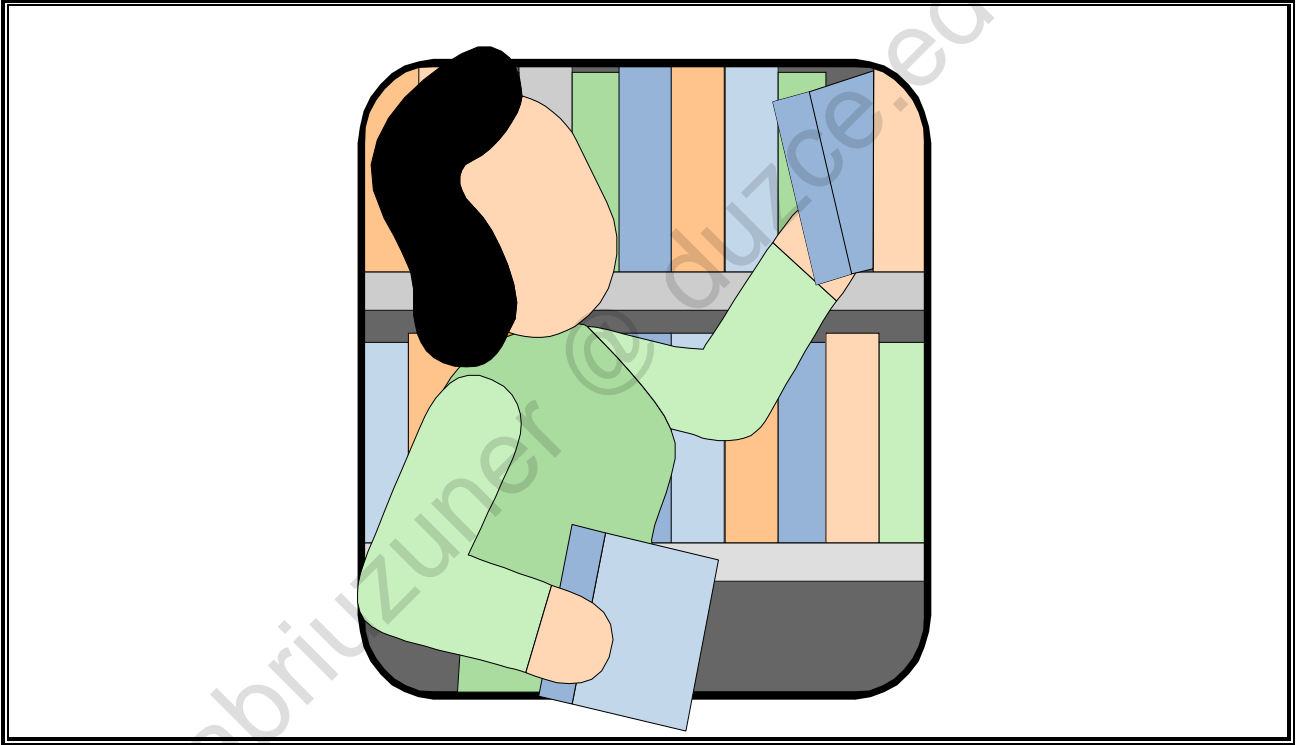
## 6.4.5. Exercise 5: Function test

DB_Receive					
	Name	Data type	Start value	Monitor value	Retain
1	Static				<input type="checkbox"/>
2	Receivebuffer	Array[1..100] of Int			<input type="checkbox"/>
3	Receivebuffer[1]	Int	0	193	<input type="checkbox"/>
4	Receivebuffer[2]	Int	0	184	<input type="checkbox"/>
5	Receivebuffer[3]	Int	0	230	<input type="checkbox"/>
6	Receivebuffer[4]	Int	0	184	<input type="checkbox"/>
7	Receivebuffer[5]	Int	0	274	<input type="checkbox"/>
8	Receivebuffer[6]	Int	0	135	<input type="checkbox"/>
9	Receivebuffer[7]	Int	0	0	<input type="checkbox"/>
10	Receivebuffer[8]	Int	0	0	<input type="checkbox"/>
11	Receivebuffer[9]	Int	0	0	<input type="checkbox"/>
12	Receivebuffer[10]	Int	0	0	<input type="checkbox"/>
13	Receivebuffer[11]	Int	0	0	<input type="checkbox"/>

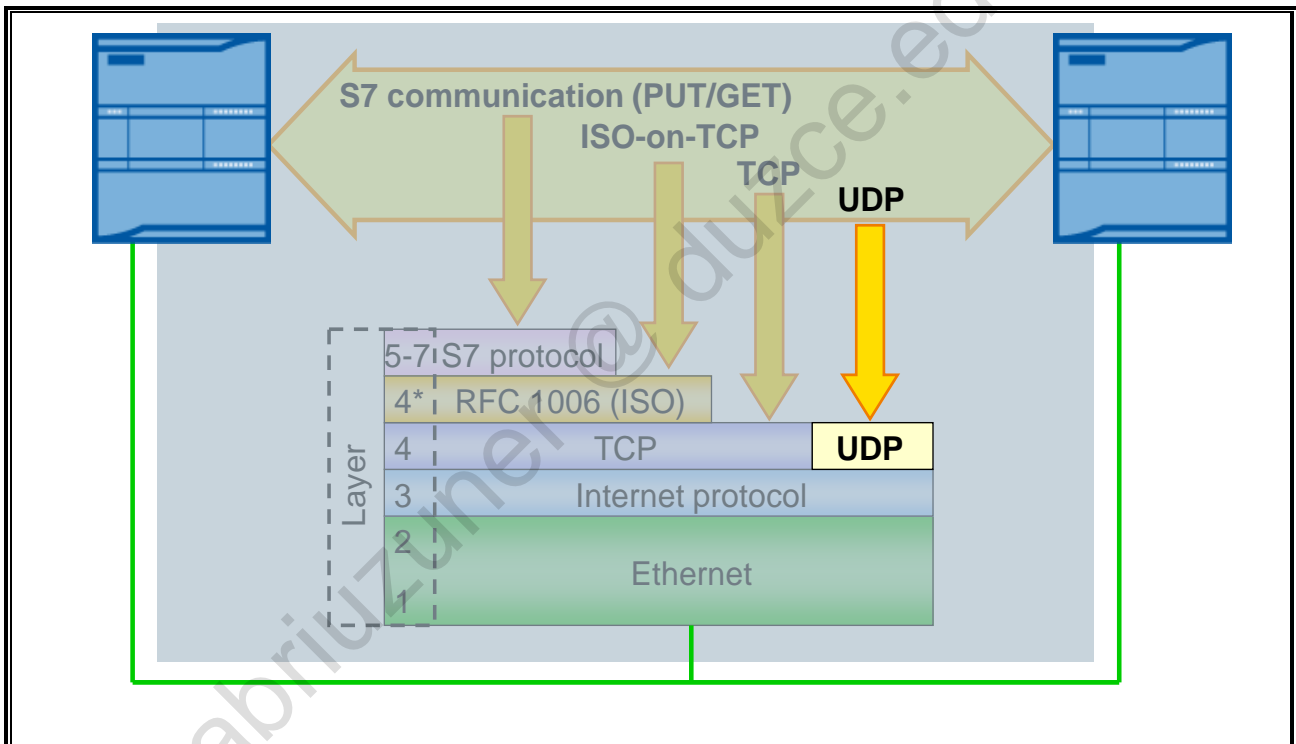
## Task

Check whether the communication between PLC\_1 and PLC\_2 works. For this, monitor "DB\_Receive" of PLC\_2. First, no data is displayed. Only after a part has passed through the light barrier is a send sequence triggered and the 200 bytes in total of data is sent to PLC\_2.

## 6.5. Additional Information



### 6.5.1. UDP Communication



#### User Datagram Protocol

The UDP protocol was introduced to transfer data quickly and straightforward. The UDP protocol is in Layer 4 (transport layer) of the ISO-OSI reference model and thus is also based on the IP layer. The receiver of data is therefore addressed with the help of IP addresses. The data packet to be sent is only increased by a minimum of administrative information so that a higher data throughput compared to TCP/IP results.

Because of the demand that data be transferred quickly, the UDP protocol merely provides basic functions. Hence, data can be exchanged between communicating partners with a minimum of effort. Security mechanisms as they exist for TCP/IP are thereby dispensed with. The UDP protocol is connectionless and packet-oriented.

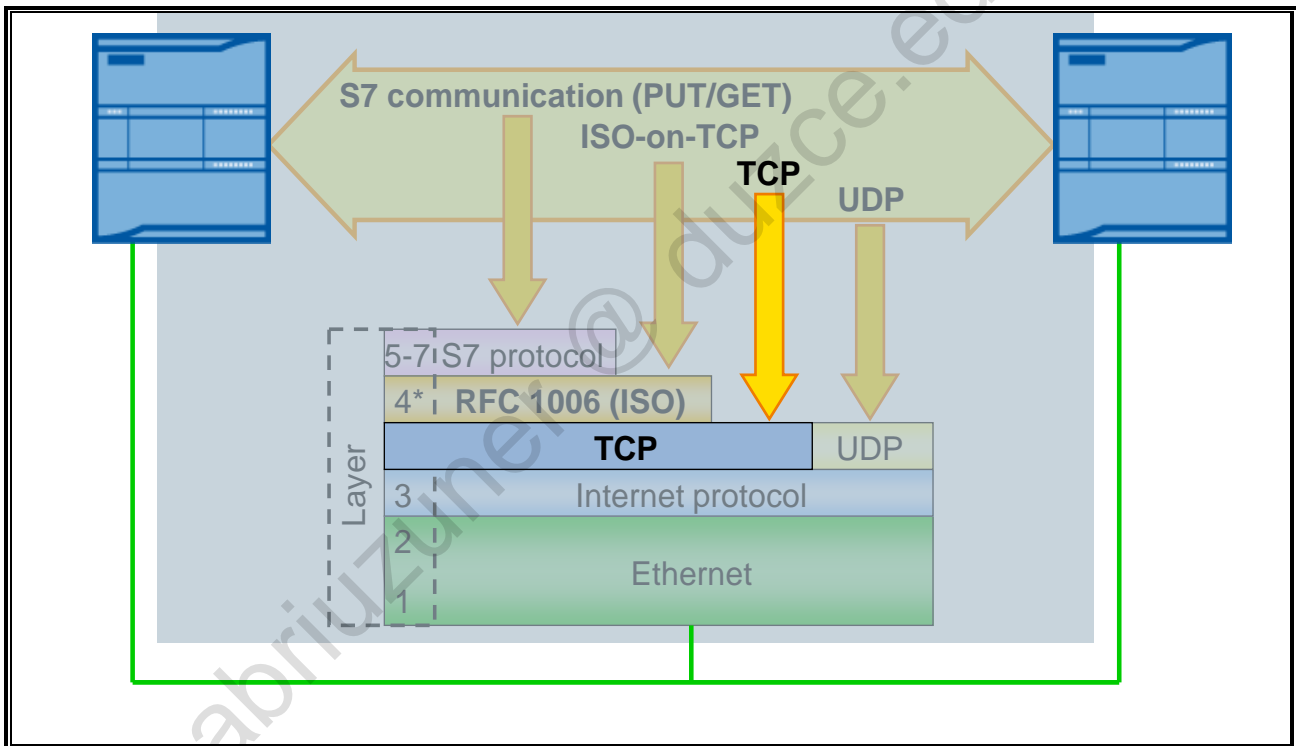
#### Advantages of UDP

- Very fast data transmission
- Very flexible, ideal for use with third-party systems
- Routing-capable
- Multicast / Broadcast capable
- Suitable for small to medium amounts of data ( $\leq 2048$  bytes)

#### Disadvantages of UDP

- Lost data packets are not resent
- Multiple deliveries of individual packets are possible
- The arrival sequence of packets at the receiver cannot be predicted

## 6.5.2. TCP Communication



### Transmission Control Protocol

When data is transferred with TCP, the transmission takes place in the form of a data flow. Neither information on length nor information about the beginning and end of a message is transferred. The receiver, however, cannot recognize where a message ends in the data flow and where the next one begins in the data flow. For that reason, the sender must define a message structure which can be interpreted by the receiver. The message structure can be composed of, for example, the data and a concluding control character such as "carriage return", which signals the end of a message.

In most cases, TCP is based on the IP (Internet protocol) and so you also talk about the "TCP/IP protocol". It is established in Layer 4 of the ISO-OSI reference model.

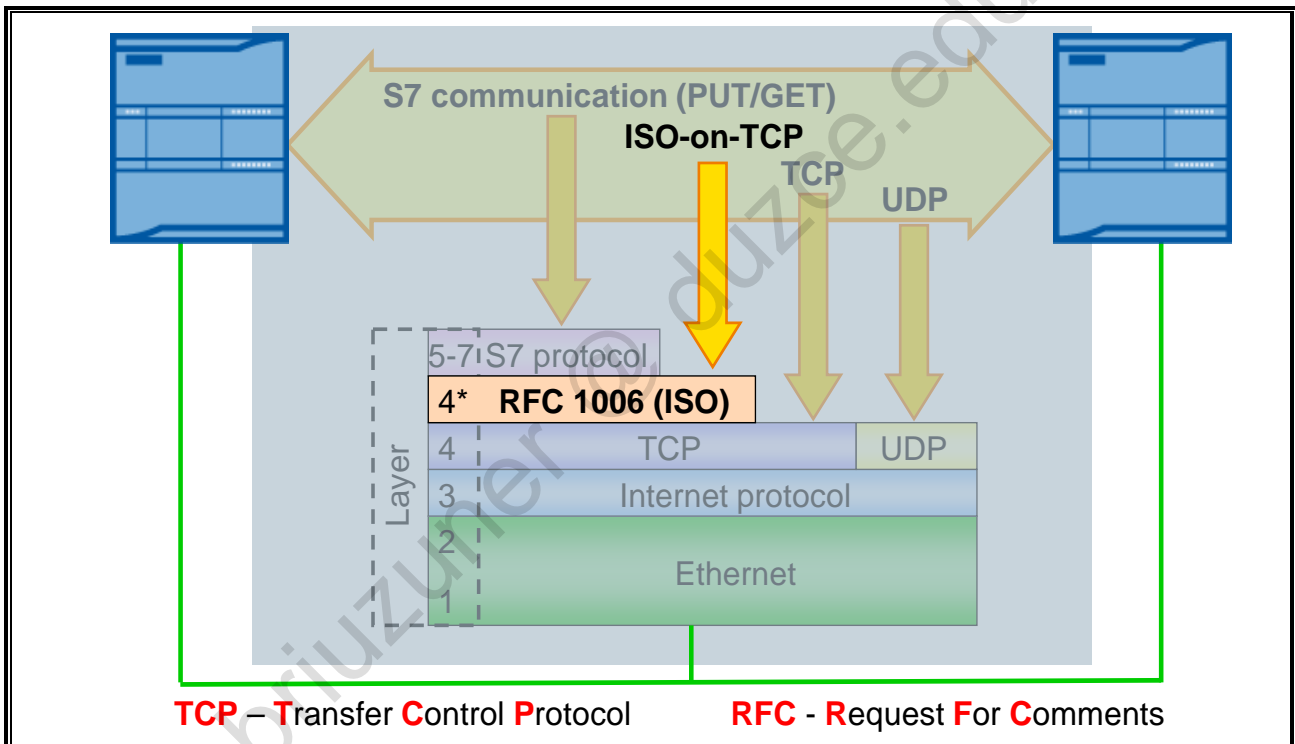
### Advantages of the TCP Protocol:

- Fast communication
- Suitable for the transmission of medium to large amounts of data (<= 8192 bytes)
- Routing-capable (i.e. can be used in WAN)
- Flexible, can be used with third-party systems
- Acknowledged

### Disadvantages of the TCP Protocol:

- Only static data lengths can be transmitted
- Greater programming effort required to manage data
- Data is transmitted as data flow

### 6.5.3. ISO-on-TCP communication



#### Background

Historically seen, the ISO Transport Protocol, as Layer 4 interface of the ISO-OSI reference model, was the first Ethernet protocol in SIMATIC. The great advantage of this protocol lies in the message-oriented transmission of data whereby the processing within the automation system becomes easier.

Since, however, the Layer 3 implementation is missing (no IP addresses) with the ISO Transport Protocol, no network addressing and thus no routing is possible.

#### ISO-on-TCP

The packet-oriented transmission of data is the great advantage of the ISO Transport Protocol. However, because of the expanding networking, the missing routing functionality developed into an increasing disadvantage.

Since the routing-capable TCP/IP protocol became increasingly popular because of the internet, an attempt was made to combine the advantages of both protocols. In the expansion RFC1006 (RFC = Request for Comments) "ISO on top of TCP", also called "ISO-on-TCP", the mapping of characteristics of the ISO transport on the TCP/IP protocol is set down. This protocol is available in all current modules of SIMATIC S7.

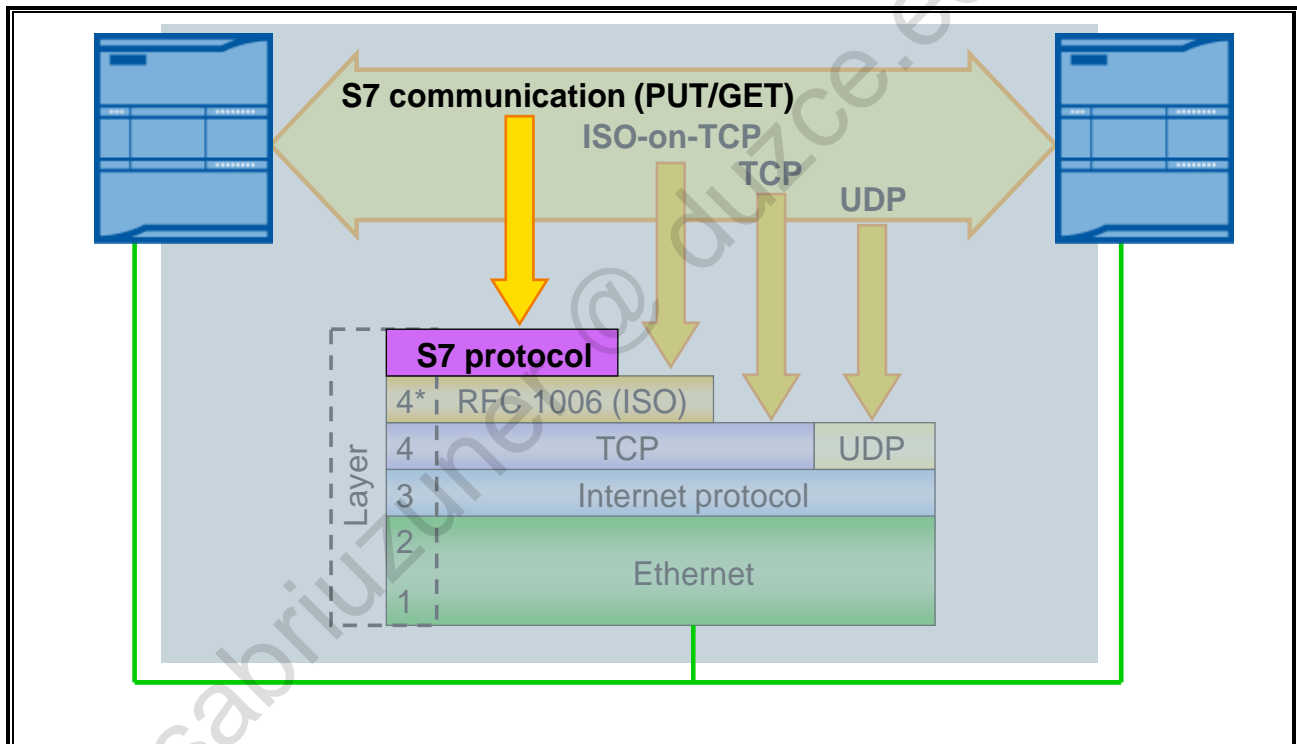
#### Advantages of the ISO-on-TCP Protocol:

- Fast communication
- Suitable for the transmission of medium to large amounts of data (<= 8192 bytes)
- Routing-capable (i.e. can be used in WAN)
- Packet-oriented data transmission
- Dynamic data lengths are possible

#### Disadvantages/Features of the ISO-on-TCP Protocol:

- Mainly applicable in SIMATIC homogenous structures
- Greater programming effort needed to manage the data

### 6.5.4. S7 Communication



#### S7 Communication

The S7 protocol is supported by all available S7 controllers and communications processors. As well, PC systems with the appropriate hardware and software equipment support communication via the S7 protocol. The S7-400 controllers use SFBs, the S7-300 and 1200 controllers use FBs. These functions are available regardless of the bus system used so that you can use the S7 communication via Industrial Ethernet, PROFIBUS or MPI.

#### Advantages of the S7 Protocol

- Regardless of the bus medium (PROFIBUS, Industrial Ethernet (ISO o. TCP), MPI)
- Applies to all S7 data areas
- Transmission of up to 64KByte in one task
- Layer 7 protocol independently ensures for acknowledgement of the data records
- Low processor and bus load in the transmission of larger amounts of data since it is optimized for SIMATIC communication

#### Disadvantages of the S7 Protocol

- Manufacturer-dependent, the S7 protocol is only implemented in the SIMATIC S7 spectrum
- Not compatible with S5 communication



## 6.5.5. Connections

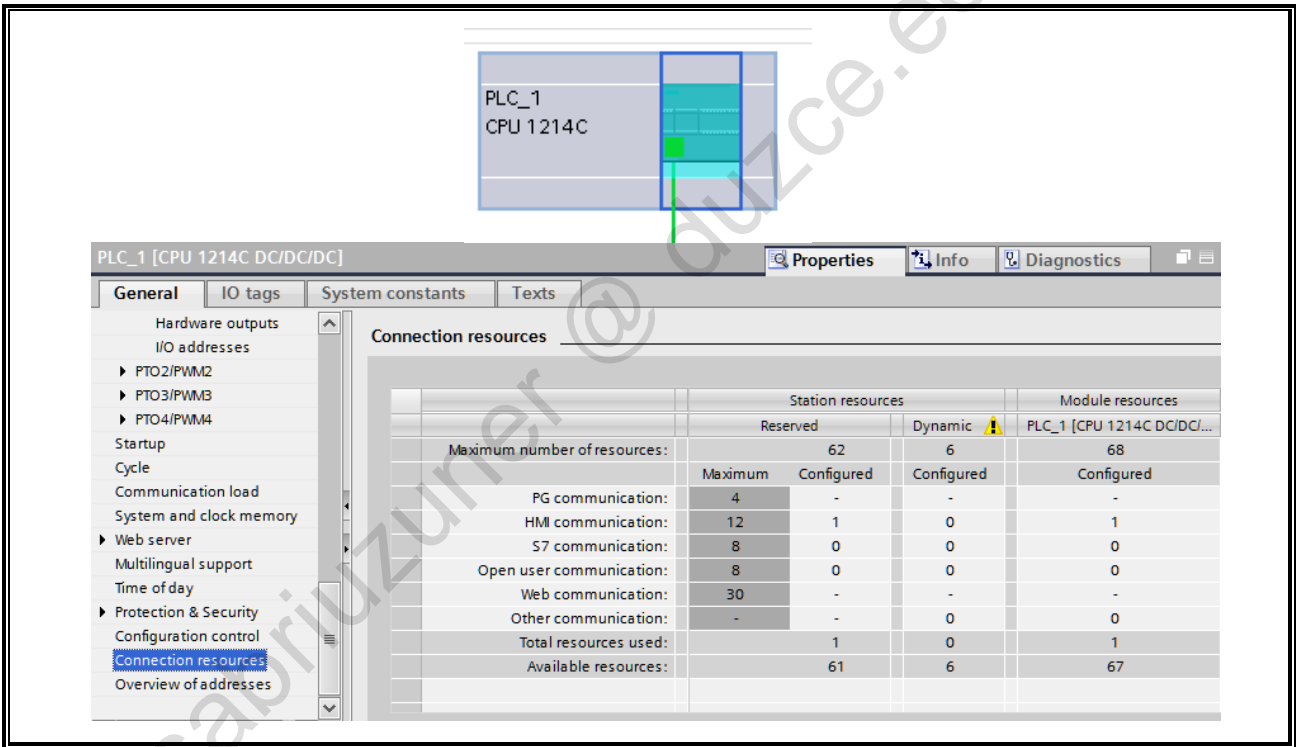
The screenshot displays two views of the SIMATIC Manager interface. The top view is a graphical network diagram showing four devices: PLC\_1 (CPU 1214C), HMI\_1 (KTP600 Basic co...), IO device\_1 (IM 151-3PN), and PLC\_2 (CPU 1211C). They are interconnected via a network labeled PNIE\_1. The bottom view is a tabular 'Connections' table under the 'Network view' tab. The table lists connections between local and partner devices.

Local connection name	Local end point	Local ID (hex)	Partner ID (hex)	Partner	Connection type
TP-CPU	HMI_1			PLC_1	HMI connection
S7_Connection_1	PLC_1	100	100	PLC_2	S7 connection
S7_Connection_1	PLC_2	100	100	PLC_1	S7 connection

### Connections

In the editor “devices and networks” (network view) projected connections can be displayed in the tabular area.

### 6.5.6. Connection resources



#### Connection resources

Connection resources are PLC dependent. For a projected PLC available connection resources are displayed in the PLC properties.

### 6.5.7. Diagnosing the Open User Communication

**Go online**  
In the connection table you can see the status of the connections  
You will find further details in the Inspector window

Local connection name	Local end point	Local ID (hex)	Partner ID (hex)	Partner	Connection type
TP-CPU	HMI_1			PLC_1 [CPU 1214C D...	HMI connection
Programmed open user...	PLC_1		192.168.111.114		Programmed open user communication
Programming device co...	PLC_1		Unknown		Programming device connection

**Connection details**

Connection name: Programmed open user communication\_192.168.111.114  
 Local ID (hex): 1  
 Connection type: Programmed OUC connection  
 Protocol: ISO-on-TCP  
 Online status: **Disconnected**  
 Details: Faulty: Connection exists only online. Connection is disconnected (either because no connection attempt was ever made or after disconnecting).

#### Connection diagnosis

Configured connections can be diagnosed in the tabular area of the "devices and networks" editor. Only an online connection must be established for this. If the connection is selected, further details are displayed in the Inspector window.

# Contents

<b>7.</b>	<b>Tags and messages in HMI.....</b>	<b>7-2</b>
7.1.	Objectives .....	7-2
7.2.	Task description: Outputting the analog value, configuring messages and time-of-day synchronization with the CPU.....	7-3
7.3.	SIMATIC WinCC .....	7-4
7.4.	HMI device maintenance: Backup/restore with ProSave.....	7-5
7.5.	HMI device maintenance: Pack & Go .....	7-6
7.6.	HMI project structure.....	7-7
7.7.	Configuring an I/O field (conventional) .....	7-8
7.8.	Configuring an I/O field (drag & drop).....	7-9
7.9.	Task description: Outputting the transport time on the Touchpanel .....	7-10
7.9.1.	Exercise 1: Creating the I/O field using drag & drop.....	7-11
7.9.2.	Exercise 2: Configuring the I/O field and creating text fields .....	7-12
7.10.	Tasks of an alarm (message) system .....	7-13
7.11.	Structure of an alarm (message) .....	7-14
7.12.	Alarm (message) procedures.....	7-15
7.13.	Trigger tags for discrete alarms .....	7-16
7.14.	Configuring discrete alarms .....	7-17
7.14.1.	Slice access (all languages) .....	7-18
7.15.	Configuring analog alarms .....	7-19
7.16.	Displaying alarms (messages).....	7-20
7.17.	Task description: Configuring a discrete alarm and analog alarms.....	7-21
7.17.1.	Exercise 3: Configuring a discrete alarm .....	7-22
7.17.2.	Exercise 4: Configuring analog alarms .....	7-23
7.18.	Possibilities for Time-of-day synchronization.....	7-24
7.19.	Cyclic Time-of-day synchronization .....	7-26
7.20.	Cyclic Time-of-day synchronization by means of area pointer .....	7-27
7.20.1.	Exercise 5: Configuring the Time-of-day synchronization CPU → TP by means of a global area pointer .....	7-28
7.21.	Additional information .....	7-29
7.21.1.	Daylight saving time / standard time change .....	7-30
7.21.2.	Additional exercise: Adopting the time from the CPU .....	7-31

## 7. Tags and messages in HMI

### 7.1. Objectives

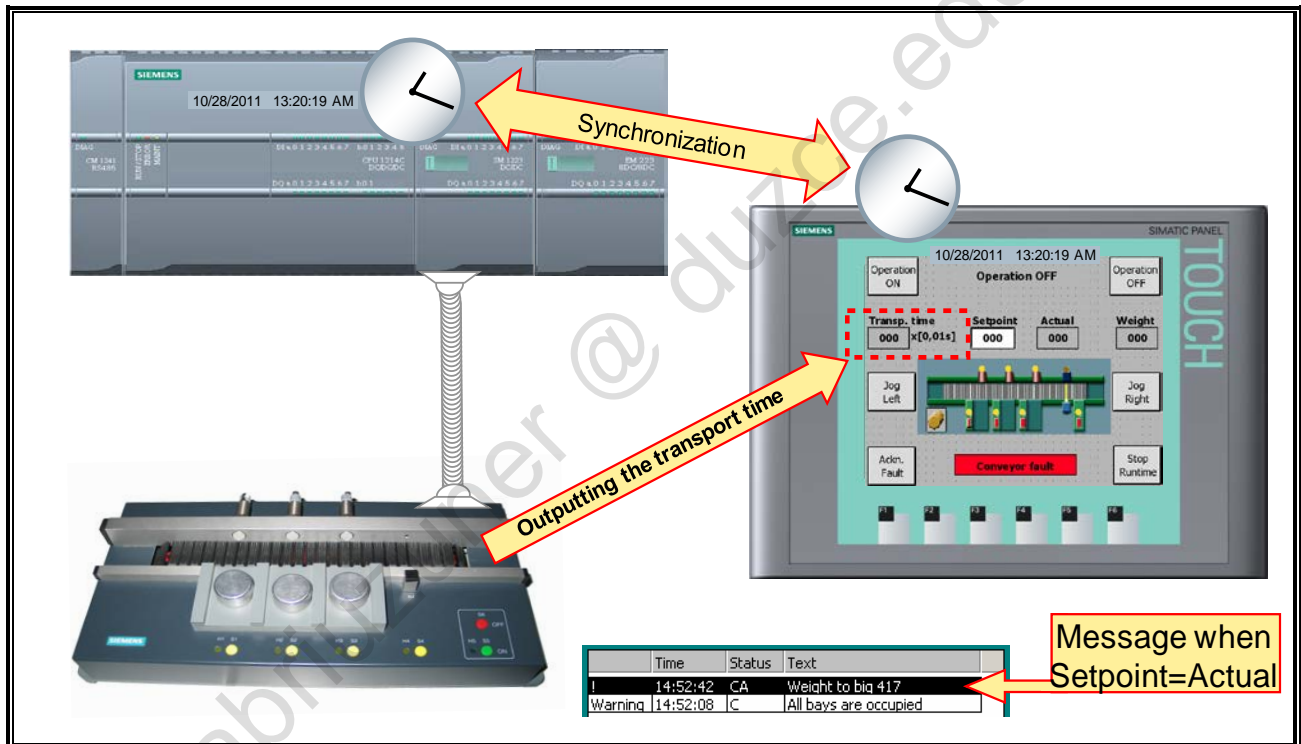
At the end of the chapter the participant will ...

- ... be familiar with the possibilities of backing up and restoring an online HMI project
- ... be able to configure an I/O field
- ... be familiar with the various alarm message procedures
- ... be able to create trigger or monitoring tags
- ... be able to configure discrete and analog alarm messages
- ... be familiar with the procedures for time-of-day synchronization between CPU and HMI

#### Objectives

In this chapter, the possibilities of backing up and restoring an HMI project are presented. As well, the procedure for creating I/O fields and alarm messages is dealt with.

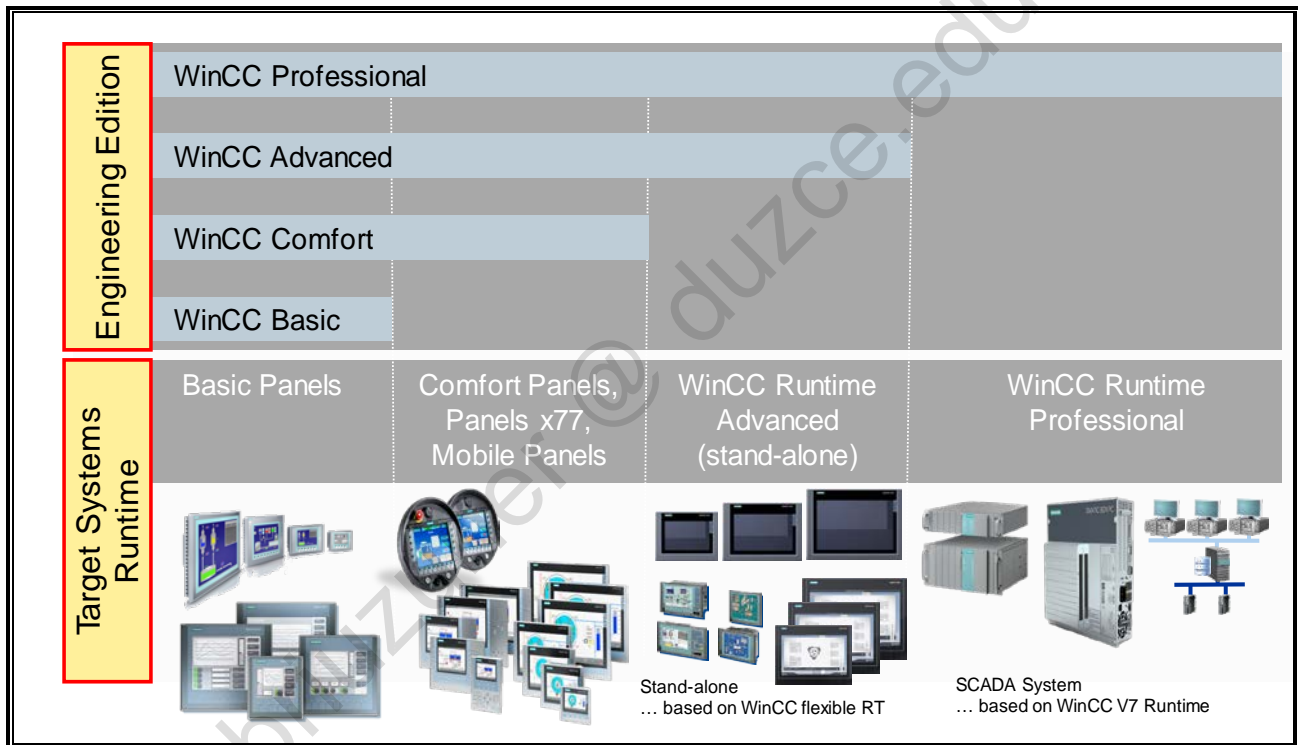
## 7.2. Task description: Outputting the analog value, configuring messages and time-of-day synchronization with the CPU



### Task description

- You are to configure an output field on the HMI in which the elapsed transport monitoring time is displayed. As process value, use the PLC tag "MD\_TranspTime".
- When the preset setpoint quantity (Actual=Setpoint) is reached, a discrete alarm is to be generated. For this, create the discrete alarm "Setpoint quant. reached!" and, as message bit, use the M33.2 bit memory in memory word "MW\_Messages" (MW32).
- When the weight exceeds or falls below the valid weight, one message each including the current weight is to be displayed. For this, configure one analog alarm each with the appropriate alarm text and also output the weight value ("MW\_Weight") with this.
- Configure the cyclic adoption of the CPU time. For this, read out the local CPU time with the function "RD\_LOC\_T" and configure the area pointer "Date/time PLC" in the HMI.

## 7.3. SIMATIC WinCC



### SIMATIC WinCC Engineering System

The engineering system is the software with which you carry out all the necessary configuring tasks in order to create an interface for controlling and monitoring machines and systems.

#### Engineering editions

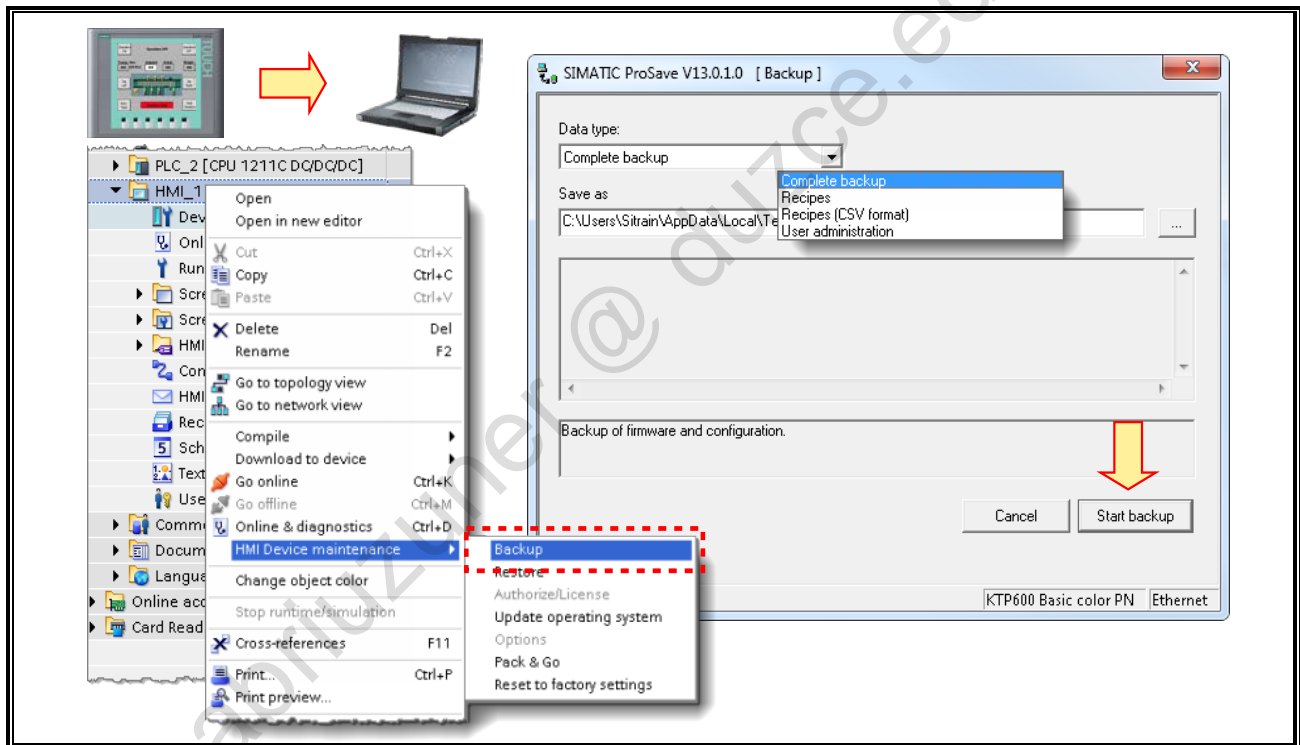
The engineering system of WinCC is modularly graded into different editions. The edition determines which systems of the SIMATIC WinCC spectrum can be configured.

- Basic: Configuration of basic panels
- Comfort: Configuration of basic panels, comfort panels, mobile panels, multi panels and operating devices of the type x77
- Advanced: Configuration of PC-based stand-alone systems
- Professional: Configuration of PC-based multi-user systems (SCADA system)

#### Runtime editions

A license appropriate for the engineering system is required on the respective target system for operation.

## 7.4. HMI device maintenance: Backup/restore with ProSave



### ProSave

With ProSave it is possible to easily make a central data archiving of WinCC. This provides you with the possibility of fast re-commissioning after a system failure or a device exchange. The backed-up data is simply transferred onto the new target device thus re-establishing the original state.

A backup of a device can be made on any storage medium and can be restored, for example, on a data server.

### Functions

ProSave provides all functions which are necessary for the transfer of data between configuring PC and operating device:

- Installation and de-installation of drivers and options
- Information about installed and installable options on an operating device
- Saving data via Backup/Restore
- Windows-CE-based devices
- Transfer of authorizations

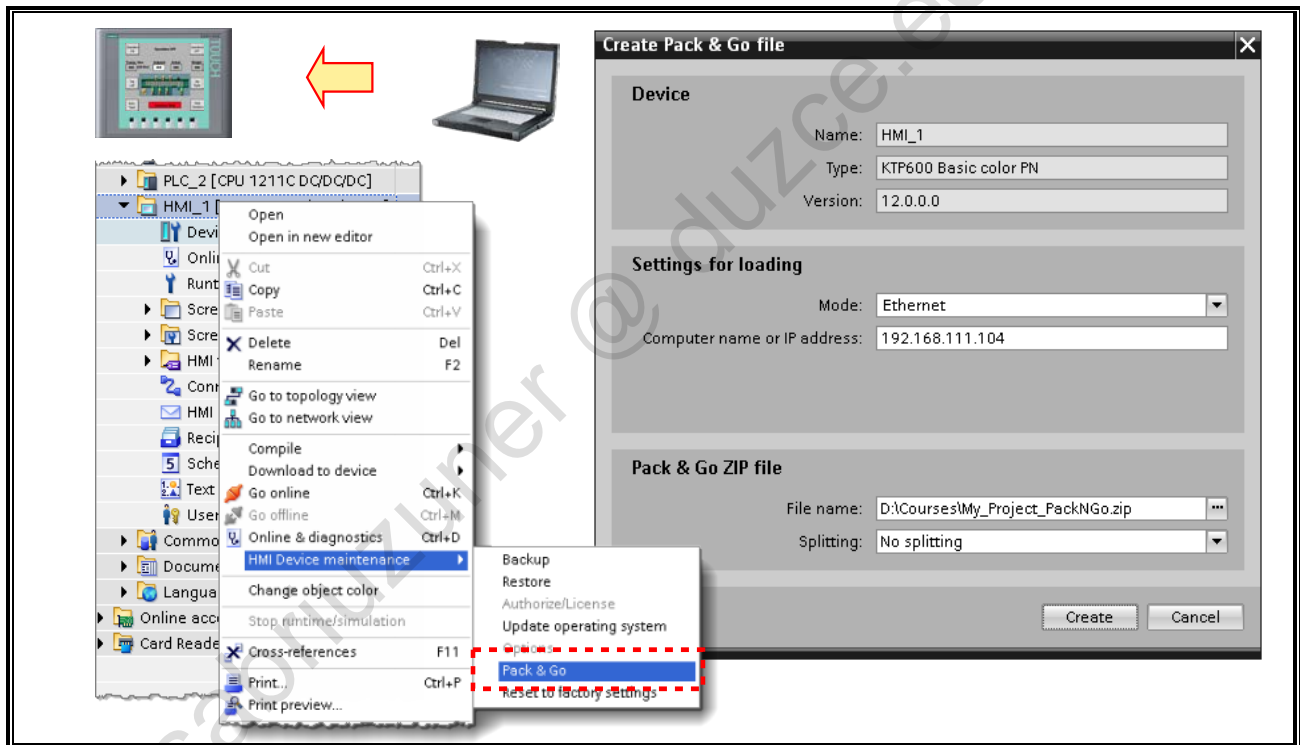
ProSave is integrated in WinCC and can, optionally, also be installed as stand-alone. Then, when servicing, you can use the stand-alone installation to restore the original data, which was previously saved with ProSave, to a target device after exchanging a device, for example.



For Windows-CE devices, a Backup/Restore can be done independent of ProSave directly from the device onto an external storage medium. External storage media are, for example, memory cards or USB mass storage devices. You will find more information in the appropriate device manuals.



## 7.5. HMI device maintenance: Pack & Go



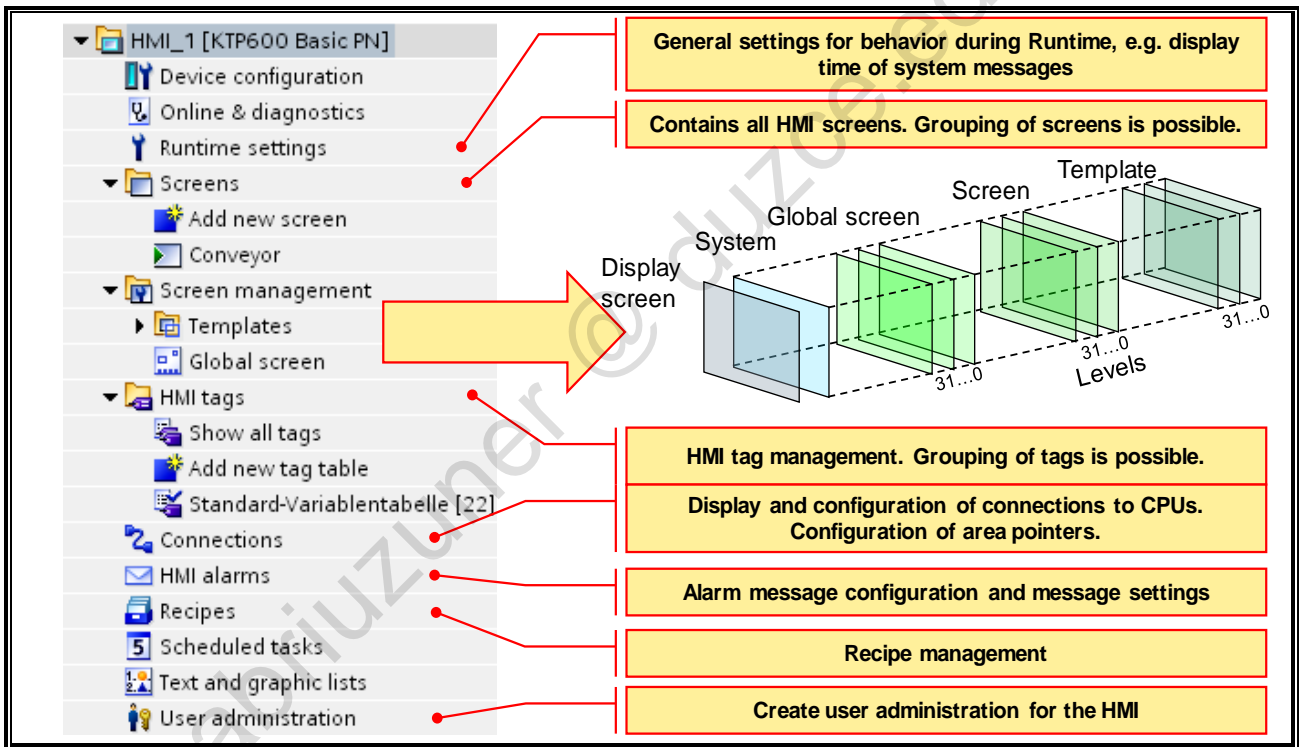
### Pack & Go

With the function "Pack & Go", it is possible to transfer an HMI project onto an operating device, even without WinCC being installed on the transfer computer.

### Operating principle

Pack & Go creates a ZIP-archive which contains your generated HMI project and a program for the project transfer. This ZIP-archive can then, for example, be sent by e-mail. The ZIP-archive must simply be unzipped on the transfer computer so that the transfer program can be started.

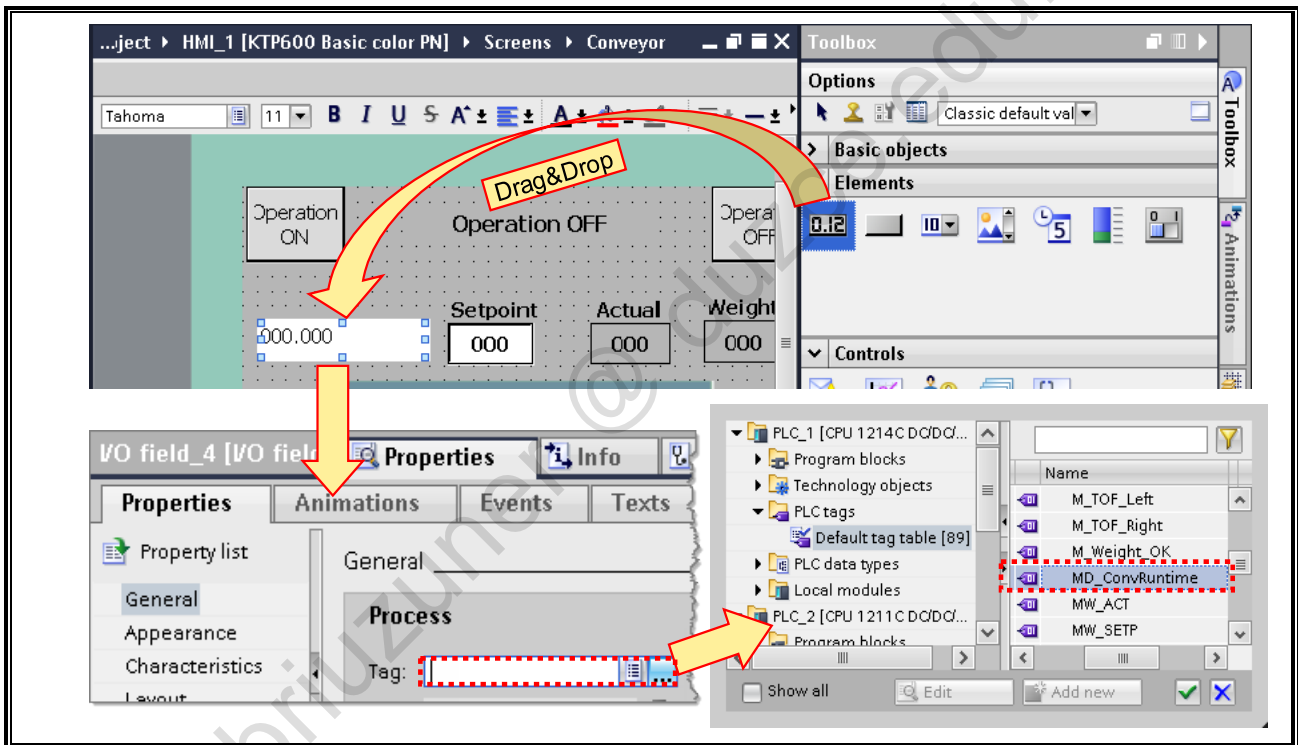
## 7.6. HMI project structure



### HMI project structure

In the Project window, only elements which are supported by the selected operating device are displayed. In the Project window, you have access to the device settings of the operating device, the language support and the version management.

## 7.7. Configuring an I/O field (conventional)

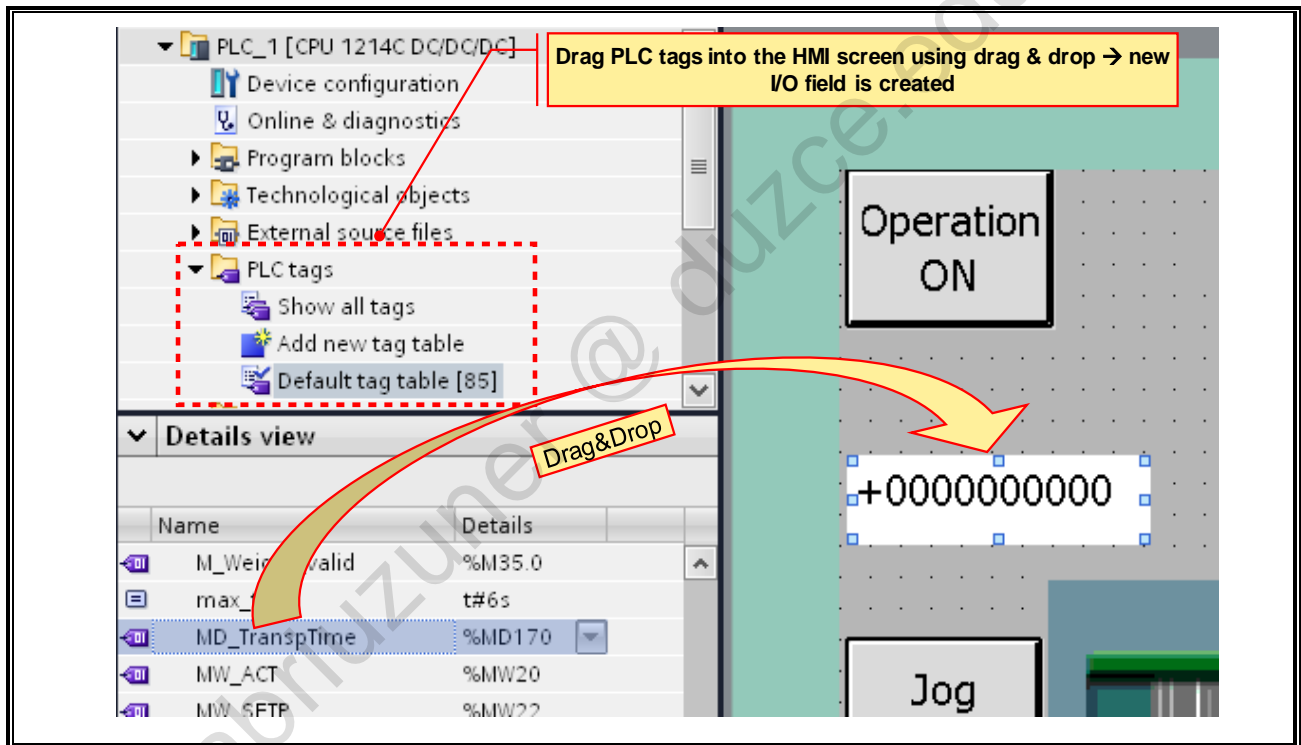


### Configuring an I/O field (conventional)

In order to create an I/O field, it is dragged from the "Toolbox" task card to the desired location in the configured screen using drag & drop.

Then, a process tag has to be assigned to the I/O field via the Properties (see picture).

## 7.8. Configuring an I/O field (drag & drop)

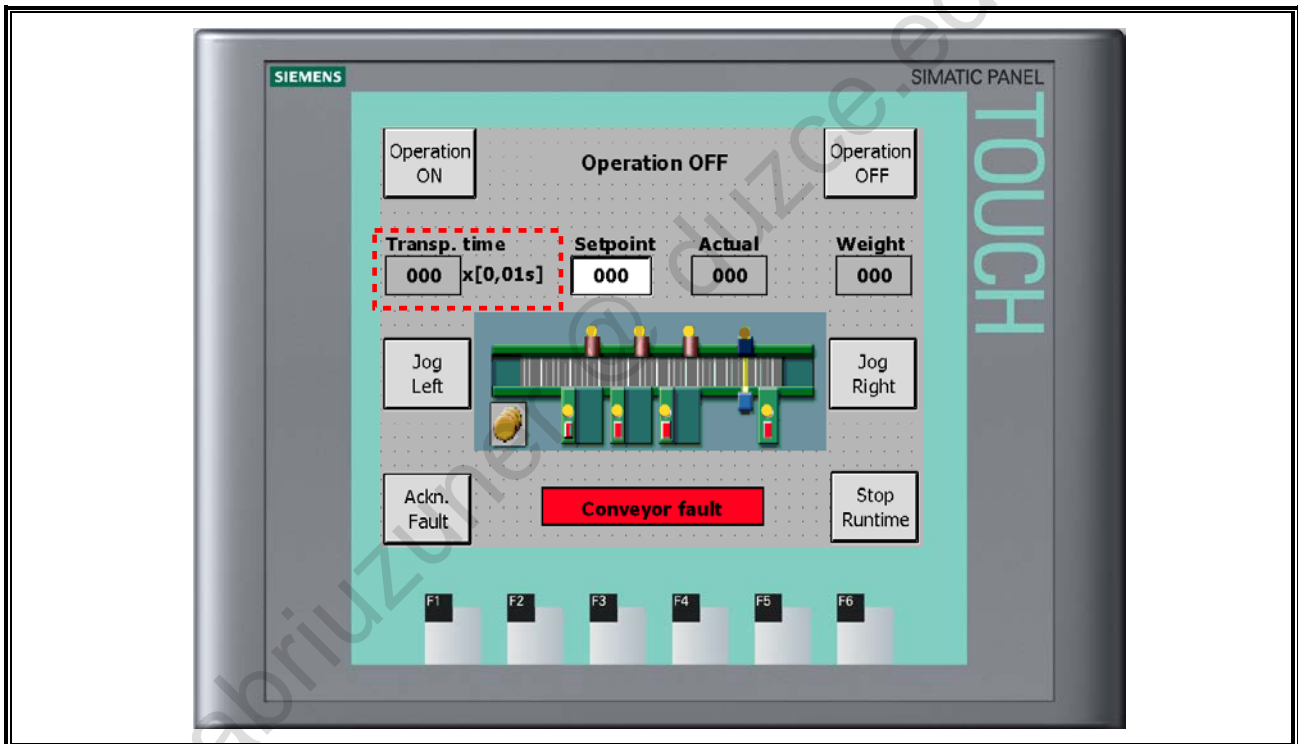


### Configuring an I/O field (drag & drop)

In order to visualize process values on the HMI, I/O fields are used. A very easy way of creating such an I/O field is to drag the desired PLC tag onto the HMI screen using drag & drop. The engineering system then automatically creates an I/O field.

The output format and the appearance of the I/O field can then be adjusted via its properties.

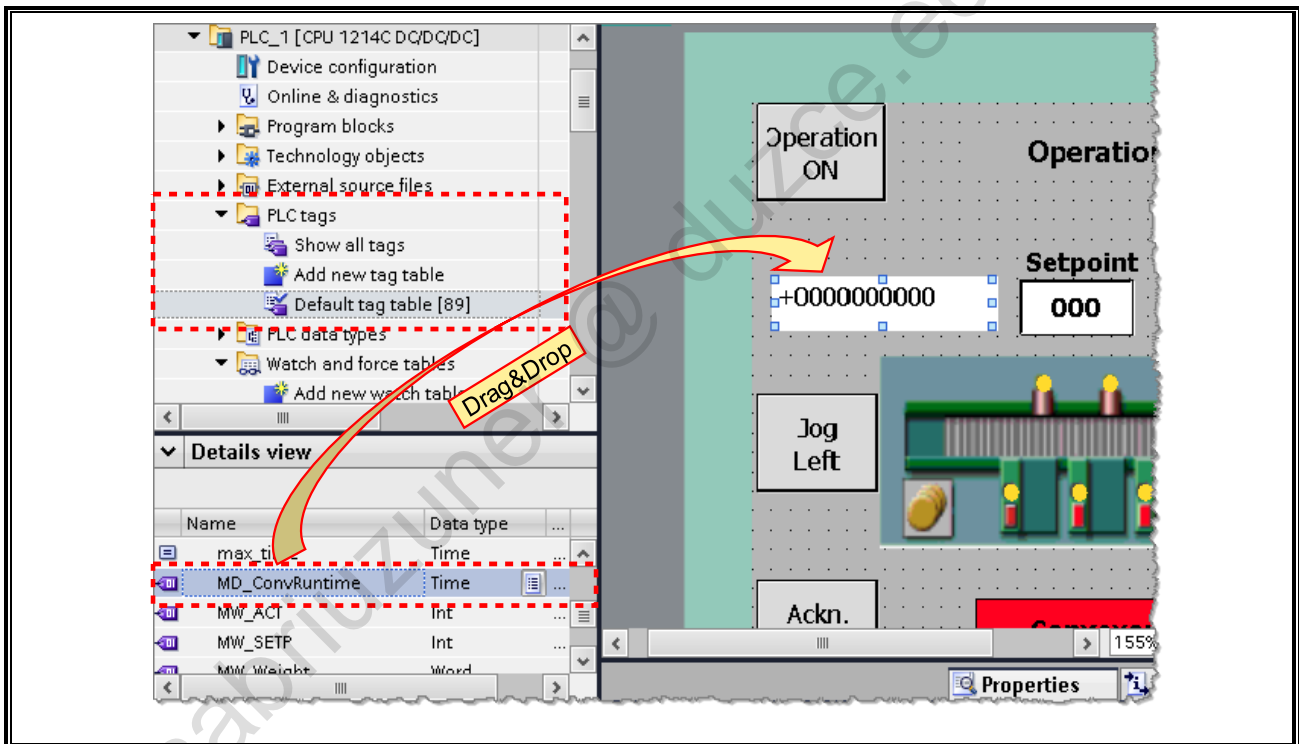
## 7.9. Task description: Outputting the transport time on the Touchpanel



### Task description

You are to configure an output field on the HMI in which the elapsed transport monitoring time is displayed. As process value, use the PLC tag "MD\_Runtime".

### 7.9.1. Exercise 1: Creating the I/O field using drag & drop



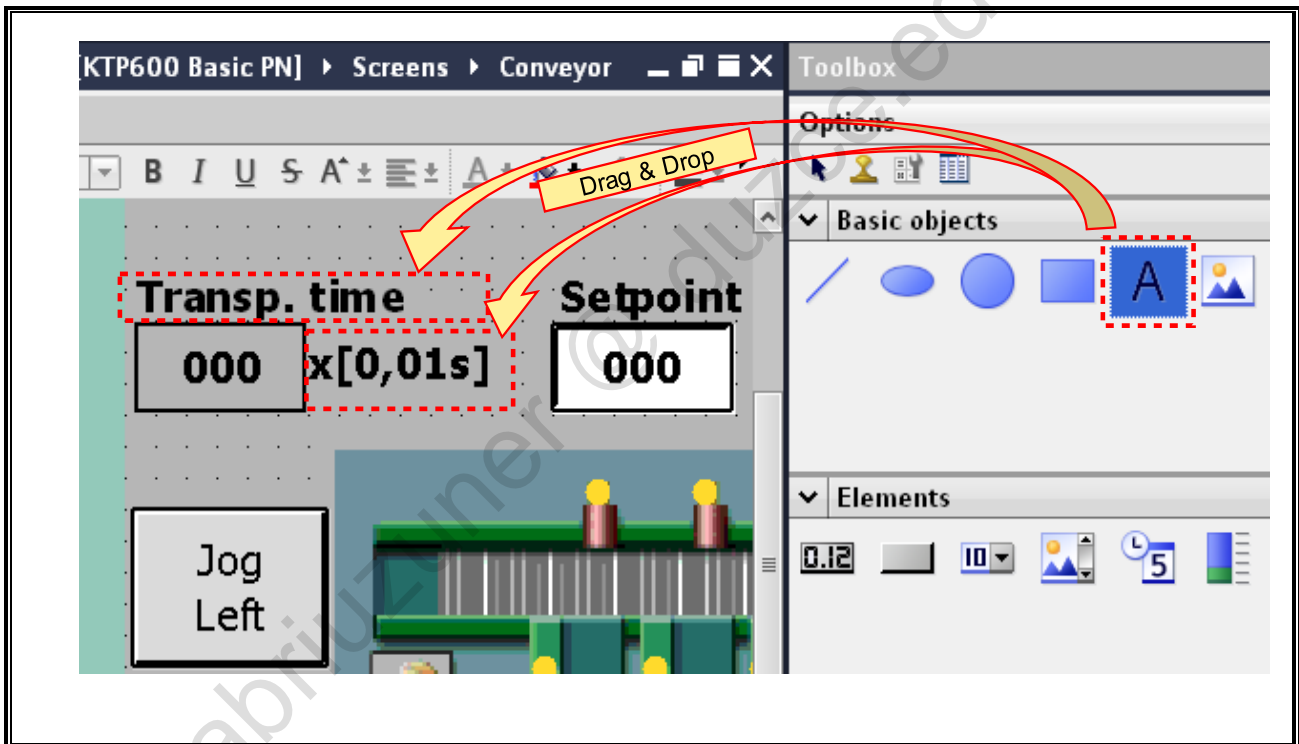
#### Task

In the HMI screen "Conveyor", create a new I/O field in the position shown on the screen.

#### What to Do

1. Open the HMI screen "Conveyor".
2. In the Project window, select the "Default tag table" in the folder "PLC tags".
3. Drag the PLC tag " MD\_Runtime" from the Details view onto the HMI screen using drag & drop.

## 7.9.2. Exercise 2: Configuring the I/O field and creating text fields



### Task

Adjust the properties of the previously created I/O field and add two text fields as labeling.

### What to do

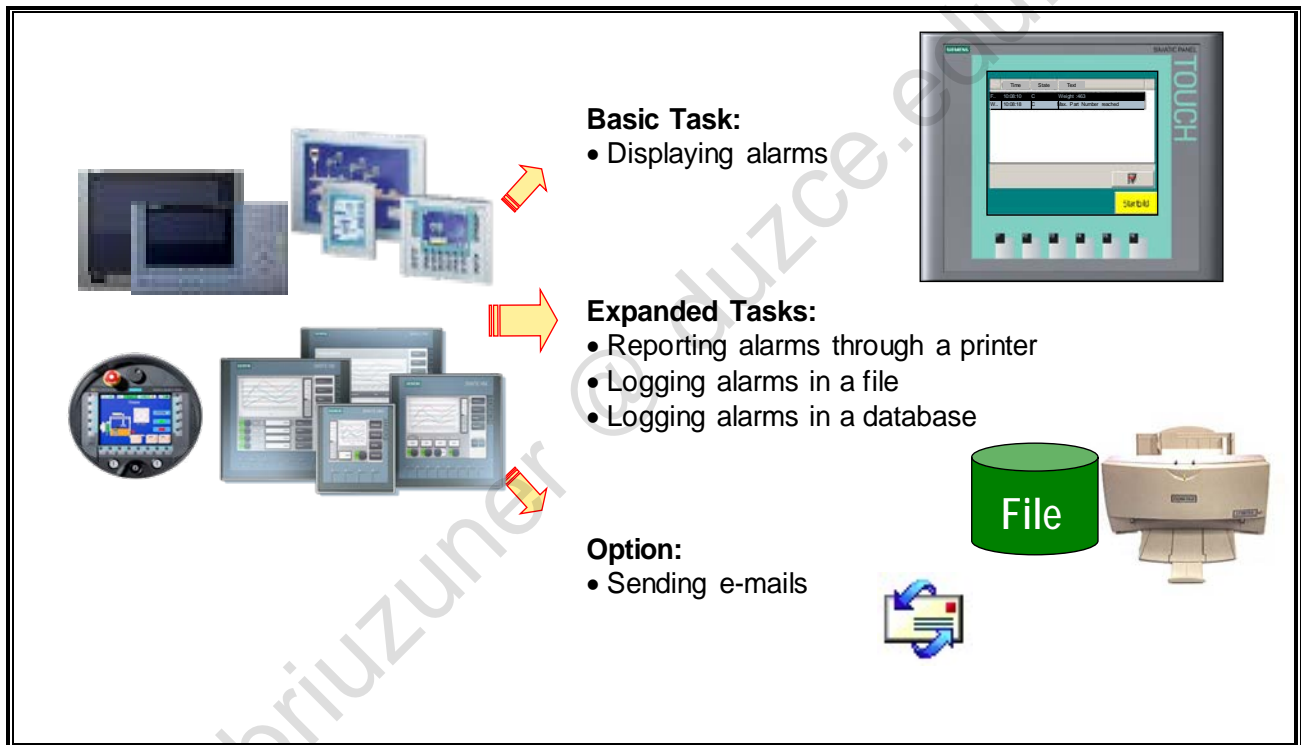
1. Adopt the settings for the I/O field shown in the following.

2. Adjust the I/O field optically to the one that already exist.
3. Change the Acquisition cycle of the HMI tag "MD\_TranspTime" in the HMI tag table.
4. Generate two text fields for the labeling of the I/O field.
5. Save your project and transfer your HMI project onto the touchpanel.



The PLC tag "MD\_TranspTime" has the data type TIME. This is defined in milliseconds. On the touchpanel, the only meaningful choice for the display format is Decimal to display this data type. A scaling was already configured for the tag "MD\_TranspTime" so that hundredth of a second instead of milliseconds is output in the just configured I/O field.

## 7.10. Tasks of an alarm (message) system



### Tasks of an alarm (message) system

Alarm messages are output depending on events or states that occur in the plant or in the process.

- **Displays:** The alarms are displayed on the operating device. This task is supported by every operating device.
- **Reporting:** The alarms are output to a printer.
- **Logging (archiving):** The alarms are stored for further processing and analysis in a file (flash or hard drive).
- **Sending e-mails:** Through this task, the "SendEmail" system function can also be configured to a message.

 Whether the tasks named are supported depends on the respective operating device type. Information on the functional scope can be found in the manual.



## 7.11. Structure of an alarm (message)

No.	Time	Date	Status	Text	GR
\$ 70022	3:48:53 PM	8/24/2009	C	Password list import started.	0
!!! 1	3:44:37 PM	8/24/2009	CDA	Speed limit activated	0
!!! 3	3:44:14 PM	8/24/2009	CD	Maximum speed reached: 1149	0
1	3:44:14 PM	8/24/2009	CD	Actual speed high	0
!!! 1	3:44:14 PM	8/24/2009	CD	Speed limit activated	0
2	3:44:14 PM	8/24/2009	CD	Actual speed too high	0
!!! 3	3:41:52 PM	8/24/2009	C	Maximum speed reached: 1350	0
!!! 1	3:41:52 PM	8/24/2009	C	Speed limit activated	0

Callouts from the diagram:

- Time stamp**: Date and time-of-day (points to Date and Time columns)
- Alarm text**: with a max. 8 insertable process values (points to Text column)
- Alarm number within an alarm class** (points to No. column)
- Alarm class**: Name or identifier can be configured (points to No. column)
- Alarm status** (points to Status column)
- Alarm group for group acknowledgement (optional)** (points to GR column)

### Alarm classes

Each alarm is assigned to an alarm class that has specific properties. In addition to the following standard alarm classes, you can also declare your own alarm classes.

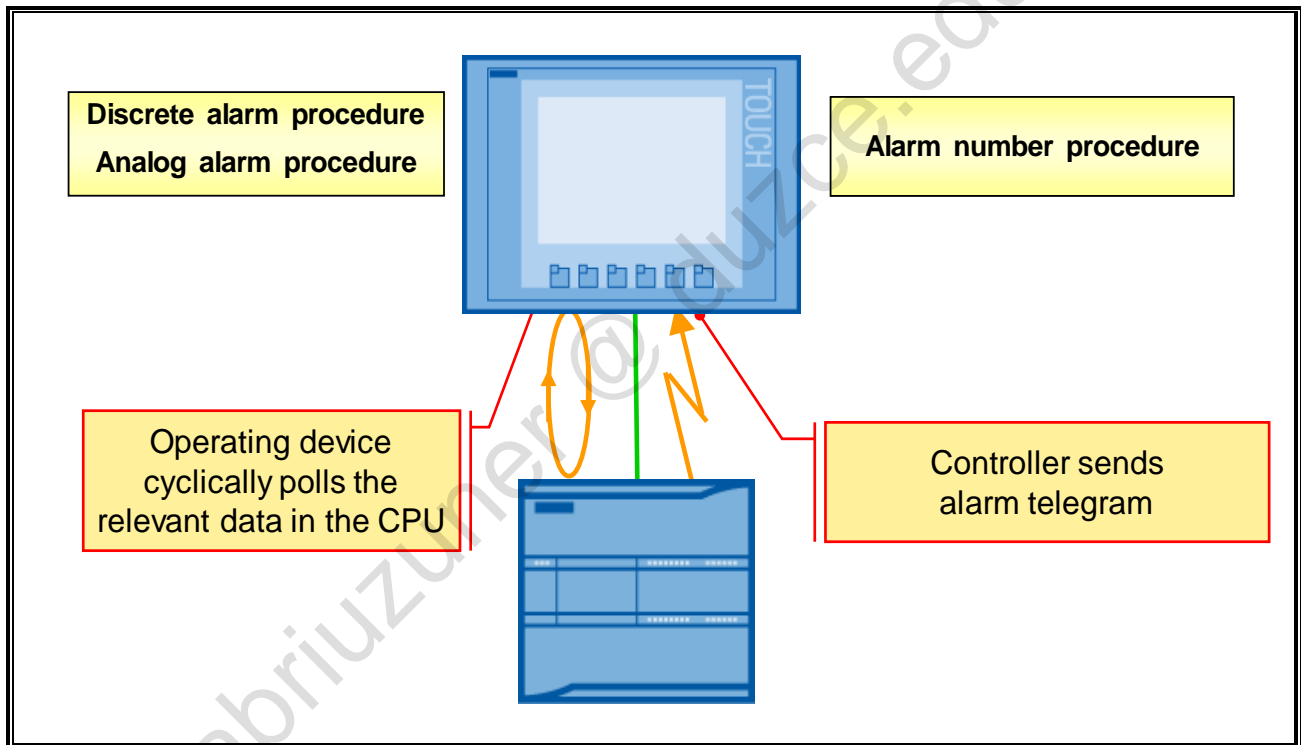
- Diagnosis Events are generated by the controller and report diagnostic events with standard texts (such as, the exceeding of the measuring range of an analog input module)
- System Alarms are alarms with standard texts that are output by the operating panel (such as, connection setup to the controller ...)
- Error Alarms must always be acknowledged and can be freely configured by the user. That is, you define the alarm event (→ discrete or analog alarms) as well as the alarm text.
- Warnings do not have to be acknowledged and like error alarms can be freely configured by you.

### Alarm statuses

The following alarm statuses exist:

- Alarm statuses that do not need to be acknowledged:
  - C: came
  - CD: came and done (gone)
- Alarm statuses that must be acknowledged:
  - C: came
  - CA: came, still existing (pending) and already acknowledged
  - CD: came, done (gone), but not yet acknowledged
  - CDA: came, done (gone) and later acknowledged
  - CAD: came, acknowledged and done (gone)

## 7.12. Alarm (message) procedures



### Alarm procedures

WinCC supports the following alarm procedures:

- **Discrete alarm procedure:**  
The operator panel displays an alarm if a certain bit of a so-called trigger tag is set in the controller. For this, the operator panel must read out the trigger tag cyclically from the S7 controller.
- **Analog Alarm Procedure:**  
The operator panel displays an alarm if a certain tag exceeds specified upper or lower limit values. For this, the operator panel must read out the tag cyclically from the S7 controller.
- **Alarm Number Procedure:**  
The operator panel displays an alarm if the controller sends an alarm telegram of the associated alarm number to the operator panel.  
This procedure has the following advantages:
  - the alarms are displayed in exactly the same sequence as they actually occur in the process
  - the alarms have a CPU time stamp and not an operator panel time stamp
  - reduced bus load because communication only takes place in case of an alarm
  - no alarms are lost

## 7.13. Trigger tags for discrete alarms

The screenshot displays the SIMATIC Manager interface. On the left, the project tree shows 'HMI tags' expanded. The main window shows the 'Default tag table' with the following data:

Name	Data type	T...	...
MD_TranspTime	Time	...	T...
MW_ACT	Int	...	T...
MW_Messages	Word	...	T...
MW_SETP	Int	...	T...
MW_Weight	Int	...	T...
T_Left	Bool	...	T...
T_Off	Bool	...	T...
T_On	Bool	...	T...

Below the table, the 'Properties' window for 'MW\_Mes...' is open, showing the 'Settings' tab. The 'Acquisition mode' is set to 'Cyclic continuous' and the 'Acquisition cycle' is set to '100 ms'.

### Trigger tags

To generate a discrete alarm, a bit edge change is necessary. These bits are managed in trigger tags. These trigger tags are edited in the HMI tags or can be created directly when configuring a discrete alarm.

### Data type

The trigger tag must be of the data type Word.

### Array elements

If more than 16 bits are needed, the range can be increased by the number of ARRAY elements. Several, different trigger tags can also be created.

### Acquisition mode "cyclic continuous"

Trigger tags must be constantly monitored in the background by the operating device since alarm events are possibly not registered otherwise.



For performance reasons, it is wise to declare the trigger tags in a continuous data area, rather than to scatter them throughout. That way, all trigger tags can be read out of the controller by the operating device with fewer reading accesses.

### 7.14. Configuring discrete alarms

The screenshot displays the SIMATIC Manager interface for configuring discrete alarms. The 'Discrete alarms' tab is selected, showing a table with the following data:

ID	Alarm text	Alarm class	Trigger tag	Trigger bit	Trigger address
1	Setpoint quant. reached: <Tag: 3, MW_SETP>	Errors	MW_Messages	6	MW_Messages.x6

Below the table, the 'Process' dialog is open, showing 'Tag: MW\_SETP' and 'PLC tag: MW\_SETP'. To the right, a diagram of the 'MW\_Messages' (MW32) word register is shown, divided into two 16-bit halves (MB32 and MB33). Bit 6 of MB33 is highlighted and labeled 'M33.6'. Red boxes and arrows highlight the 'Trigger tag' and 'Trigger bit' fields in the table, and the bit position in the diagram.

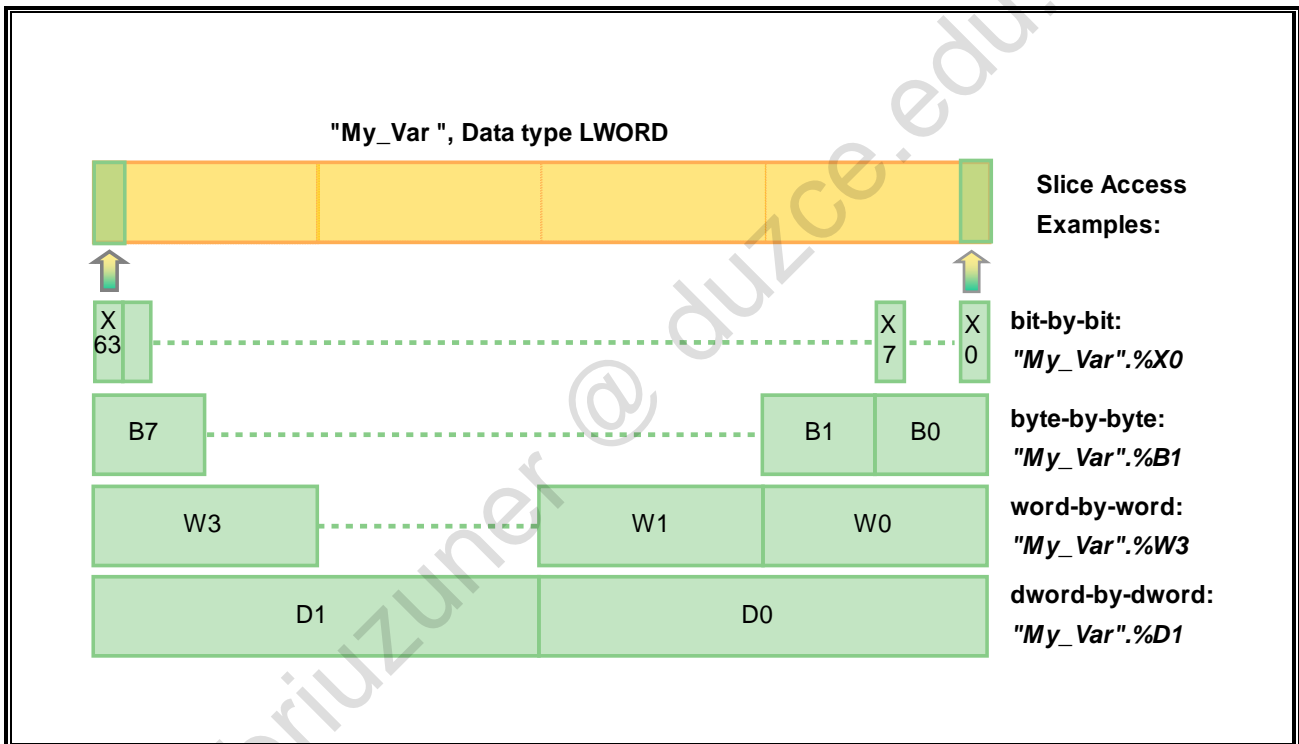
#### New alarm message

By clicking in the next empty line in the "Alarm text" column, a new alarm is created. The new alarm must be assigned to an alarm class and a trigger tag (trigger bit of a Word tag) must be defined.

#### Alarm text

The alarm text can be a maximum of 254 characters long and can be formatted character by character. Tag values and/or texts from text lists can be inserted in the alarm text. The current value of the tag used is updated or once again read in when an alarm occurs.

## 7.14.1. Slice access (all languages)

**Slice access**

In all programming languages, the slice access enables a bit, byte, word and double-word access to variables of greater dimensions respectively.

With a bit access, for example, the individual bits of an integer variable can be scanned or low and high byte can be loaded even though no symbolic names exist for these variable elements.

## 7.15. Configuring analog alarms

ID	Alarm text	Alarm class	Trigger tag	Limit	Limit mode
1	Weight is too high: <Tag : 3, MW_Weight>	Warnings	MW_Weight	400	Higher
2	Weight is too low: <Tag : 3, MW_Weight>	Warnings	MW_Weight	100	Lower

Trigger condition

### Analog alarms

An analog alarm is triggered when the value of a monitored (trigger) tag exceeds or falls below a limit value. The limit value can be specified by means of a constant or via a tag.

### Properties

- **Exceeding**  
The alarm comes if the specified limit value is exceeded, and it goes when the limit value once again falls below
- **Falling below**  
The alarm comes if the specified limit value falls below and it goes when the limit value once again exceeds (the limit)
- **Delay**  
To "debounce" the monitored (trigger) tag, a delay time can be specified. With it, you can define how long (time) the limit value must be exceeded or have fallen below before an alarm is output
- **Dead zone**  
To prevent an "alarm flood", a dead zone can be specified. With it, you can define by what amount (absolute or percental) the value of the monitored (trigger) tag must once again fall below (exceed) the limit value after the limit value has been exceeded (fallen below) so that a new exceeding of the limit value (falling below) can once again be recognized as such

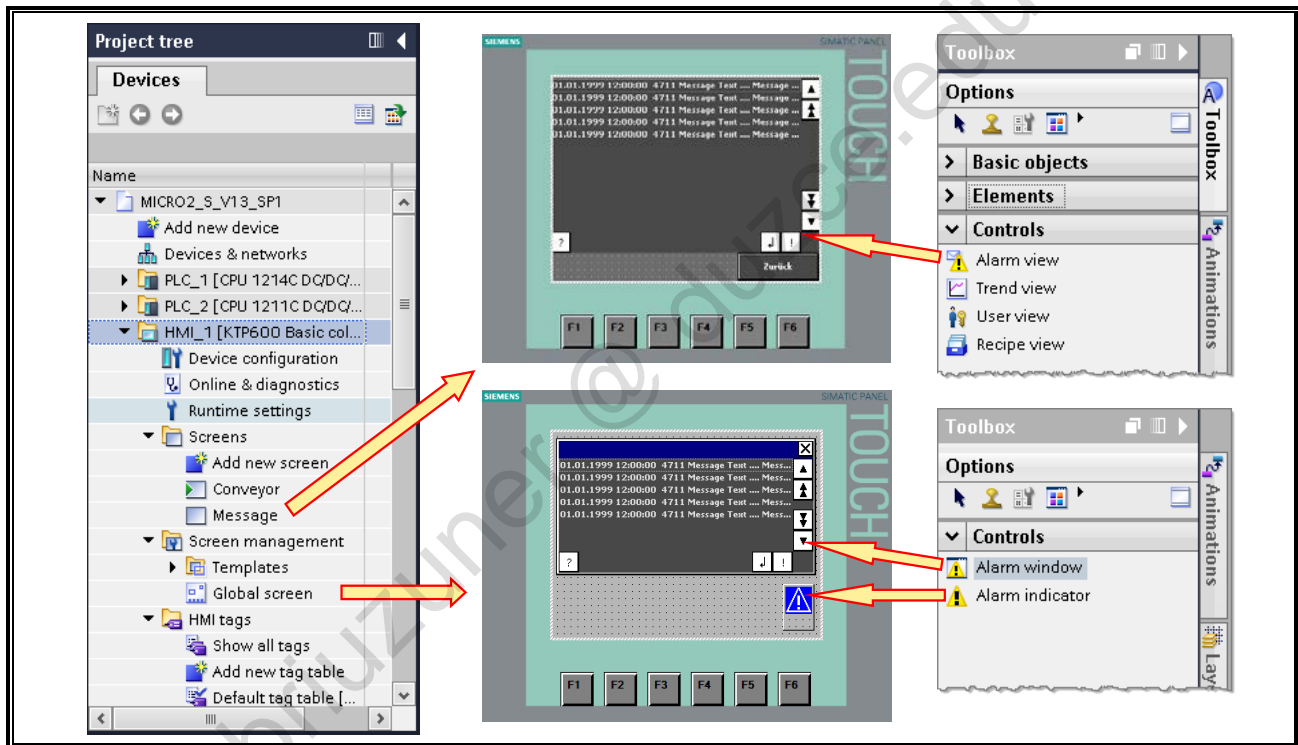
### Acquisition mode "cyclic continuous"

Monitored (trigger) tags must be constantly monitored in the background by the operating device since alarm events are possibly not registered otherwise.



When you create a tag, the default setting "Cyclic on Use" is set as the Acquisition mode. This must be explicitly switched to "Cyclic continuous" for tags that are used as "monitored (trigger) tags" for analog alarms.

## 7.16. Displaying alarms (messages)



### Alarm view

Several alarm views can be configured for different alarm classes and in various screens or various windows. The alarm view is only visible when the relevant screen is called.

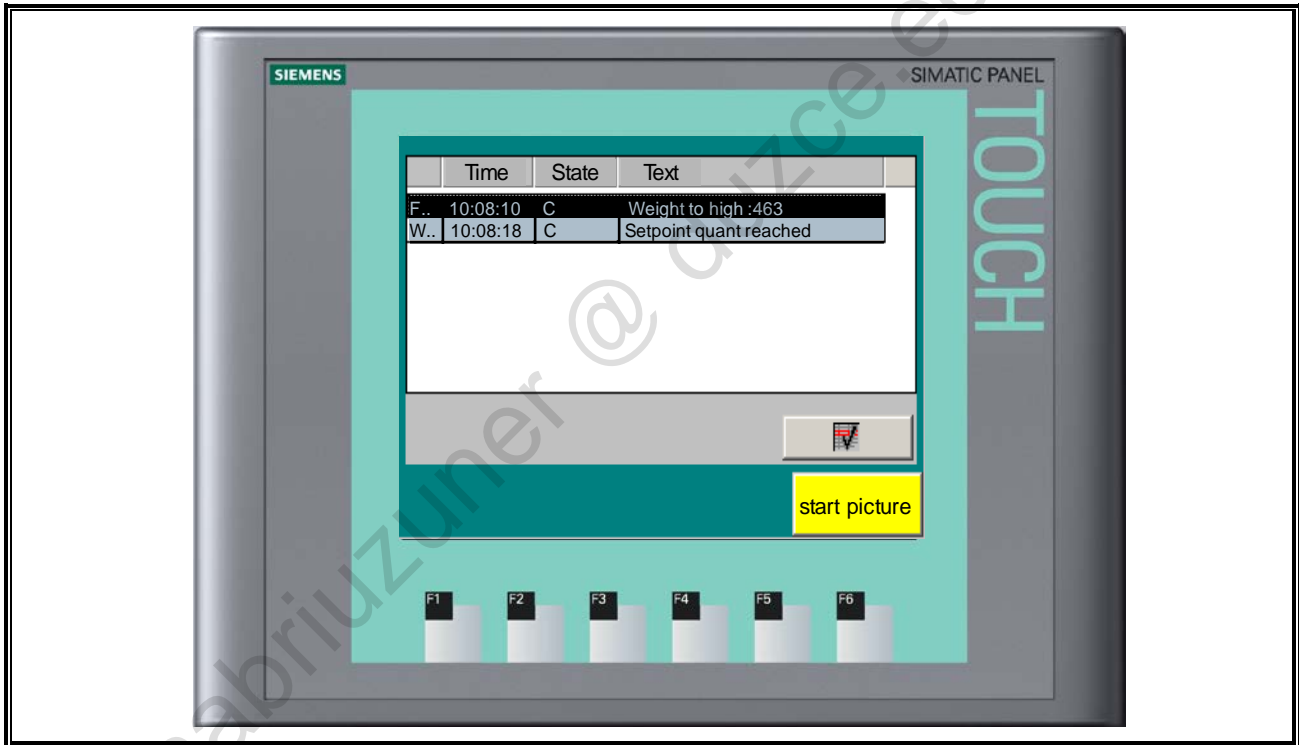
### Alarm window

The alarm window can only be configured in the "Global screen". Just as with the alarm view you can select the alarm classes to be displayed. You can define whether or not the alarm window is to appear in all screens when an alarm "comes".

### Alarm indicator

Just as the alarm window, the alarm indicator can only be configured in the "Global screen". It is a graphic symbol that points to pending alarms or alarms to be acknowledged in all screens, depending on the configuration. With the help of the alarm indicator, a defined alarm window can be shown.


## 7.17. Task description: Configuring a discrete alarm and analog alarms



### Task description

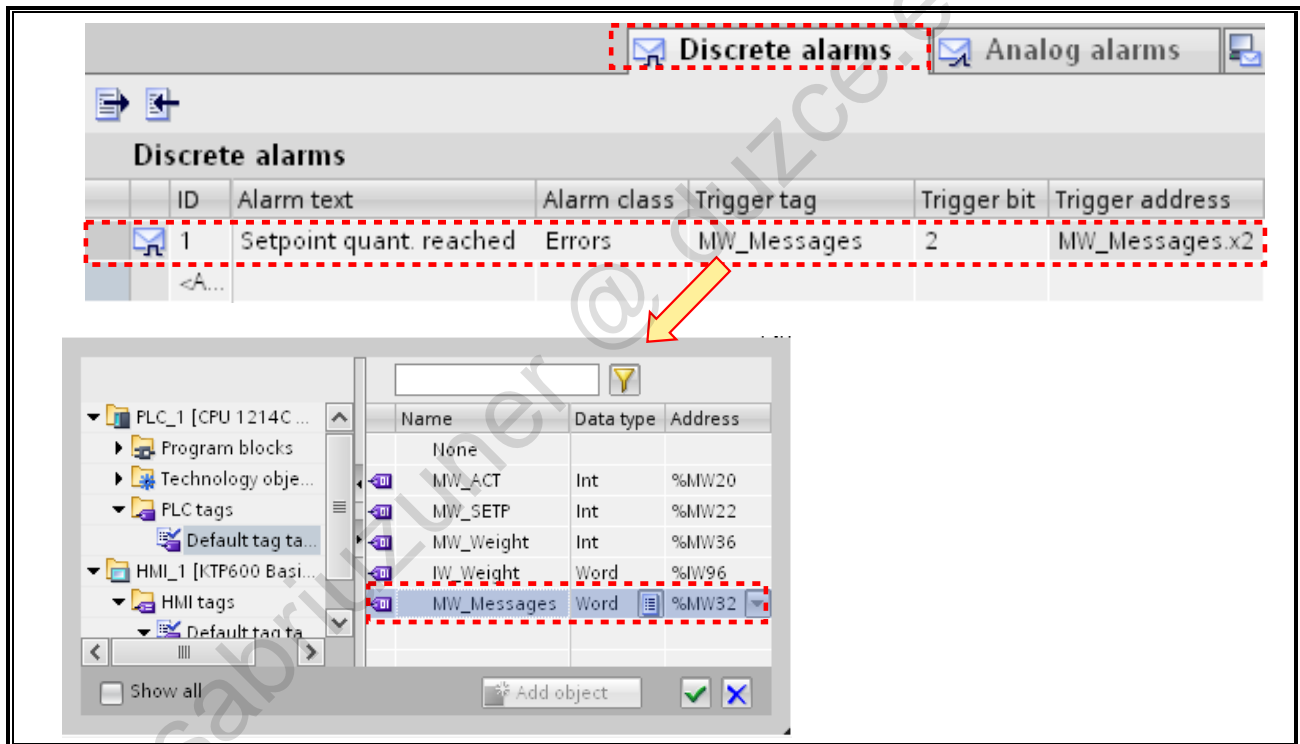
When the preset setpoint quantity (Actual=Setpoint) is reached, a discrete alarm is to be generated. For this, create the discrete alarm "Setpoint quant. reached!" and, as message bit, use the M33.2 bit memory in memory word "MW\_Alarm" (MW32).

When the weight exceeds or falls below the valid weight, one message each including the current weight is to be displayed. For this, configure one analog alarm each with the appropriate alarm text and also output the weight value ("MW\_Weight") with this.

 An appropriate alarm window for the output of discrete and analog alarms is already configured in the global screen. Alarm messages of the alarm classes "Errors", "Warnings" and "System" are displayed.



### 7.17.1. Exercise 3: Configuring a discrete alarm



#### Task

When the preset setpoint quantity (Actual=Setpoint) is reached, a discrete alarm is to be generated. For this, create the discrete alarm "Setpoint quant. reached!" and, as message bit, use the M33.2 bit memory in memory word "MW\_Alarm" (MW32).

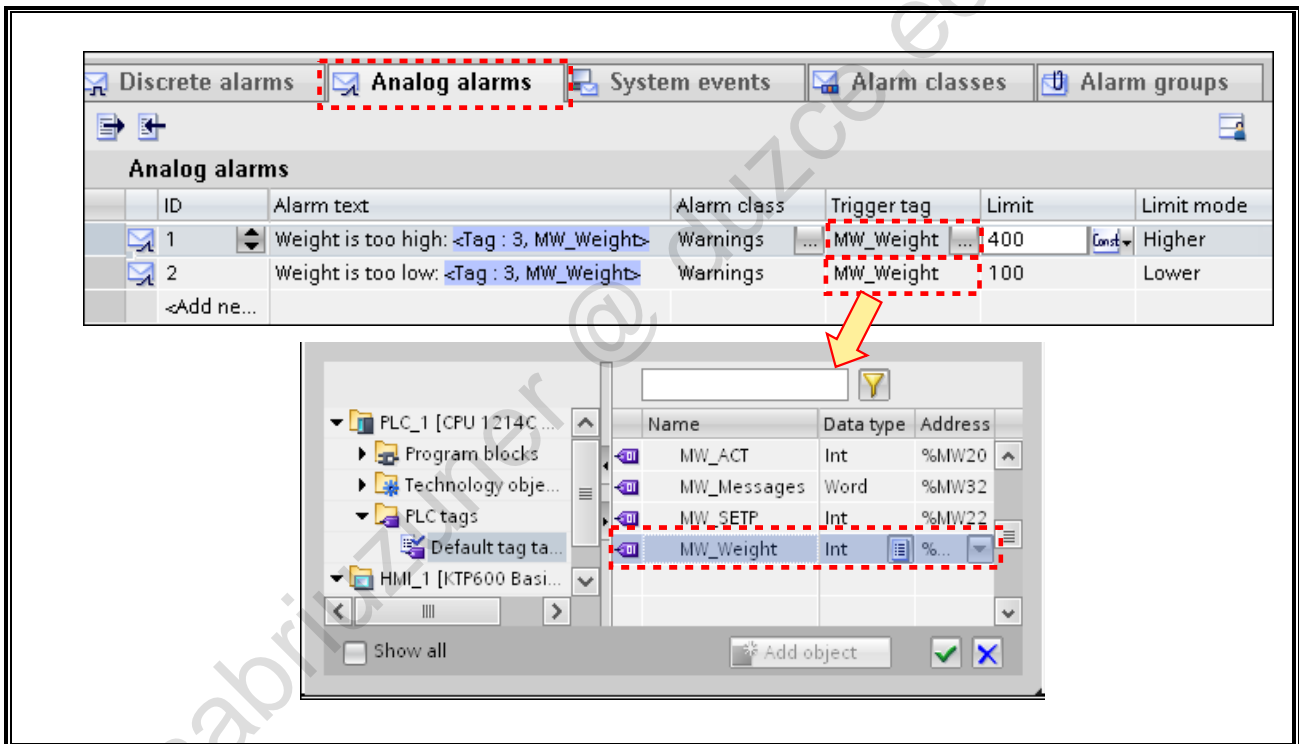
#### What to do

1. In the HMI project, open "HMI alarms"
2. Select the tab "Discrete alarms"
3. Enter the alarm text, alarm class and the trigger tag/bit
4. Load the HMI project into the touchpanel
5. Open the function "FC\_Count"
6. At output "Q" of the IEC counter, insert a further assignment to the bit memory "M\_ACT=SET\_Alarm" (M33.2)
7. Transfer the modified function to the CPU
8. Produce as many parts as you need so that the preset setpoint quantity is reached

#### Result

The message "Setpoint quant. reached!" appears on the HMI.

### 7.17.2. Exercise 4: Configuring analog alarms



#### Task

When the weight exceeds or falls below the valid weight, one message each including the current weight is to be displayed. For this, configure one analog alarm each with the appropriate alarm text and also output the weight value ("MW\_Weight") with this.

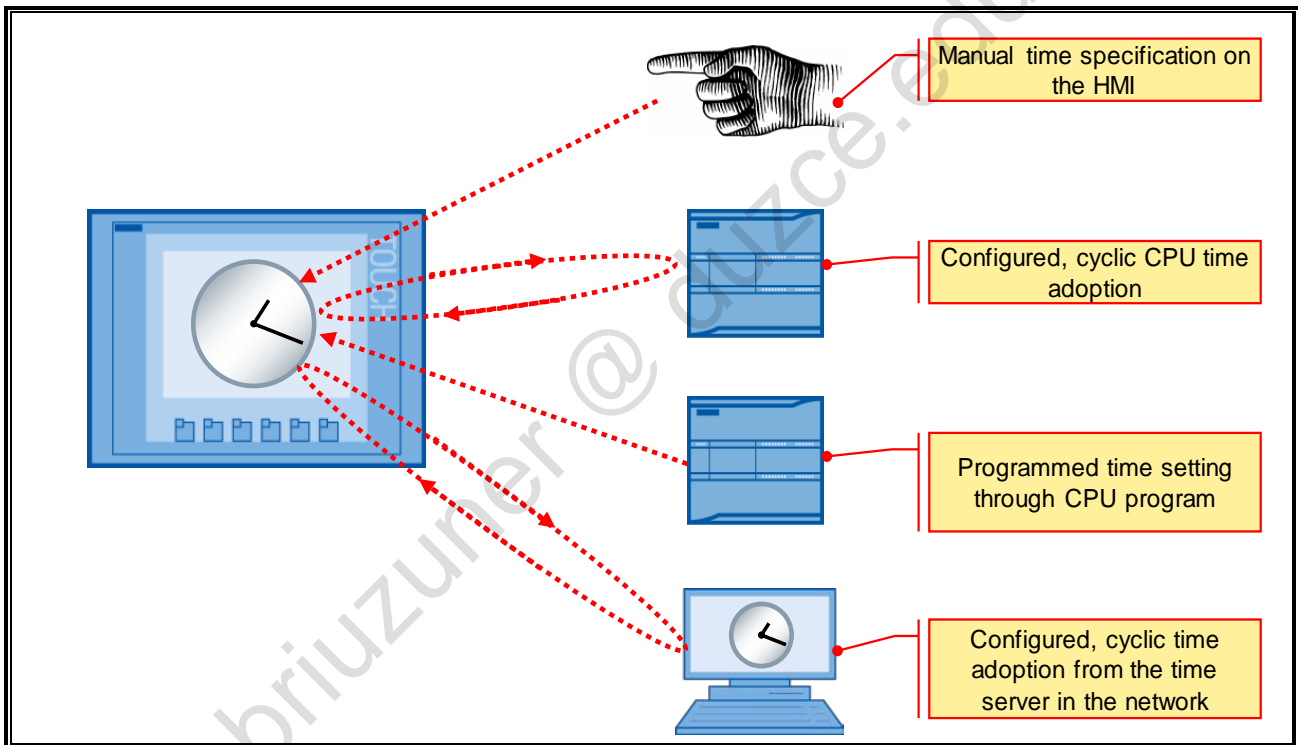
#### What to do

1. In the HMI project, open "HMI alarms"
2. Select the tab "Analog alarms"
3. Enter the alarm texts and configure the output of the process value:  
Alarm → Right-click in the text field → Insert tag field...
4. Enter the Alarm class and the Trigger tag "MW\_Weight"
5. Enter the limits (100 and 400) for controlling the alarm messages and the mode
6. Load the HMI project in the touchpanel
7. On the potentiometer, set a weight that is either too low or too high

#### Result

When the weight exceeds or falls below, the respective alarm message appears with the associated weight.

## 7.18. Possibilities for Time-of-day synchronization



### General

Since not all operating devices have a buffered realtime clock (in particular the lower performance range) the system time of the operating device must be set, since the operating device otherwise works with the initial time of the operating system after a Power OFF/ON.

Since the system time of the operating device is used as a time stamp for...

- all discrete alarms, analog alarms, system messages and diagnostic messages and
- logging (archiving) of process values within the operating device

...it must be set using one of the procedures listed below.

### Application

For the system times' synchronization of HMI and CPU, there are several procedures:

- **Manual time specification on the HMI**  
The system time is input via a configured Date/time field and doesn't represent a synchronization between CPU and HMI time in the proper sense, since the internal system clocks of CPU and HMI will inevitably deviate after the time has been input in the HMI.
- **Configured, cyclic CPU time adoption**  
The configured PLC time adoption is easy to configure and doesn't require any great program expansion on the PLC program side. For that reason, this procedure is usually used. → See next page.
- **Programmed time setting through CPU program**  
The programmed time setting through the CPU program utilizes a mechanism to start functions within the operating device from the CPU program. This can relieve the communication load of the operating device but it means programming effort in the CPU program.

- Configured, cyclic time adoption from the time server in the network (via NTP)  
The system clocks of HMI and PLC are synchronized by the same time server and thus run synchronously. In the HMI, this procedure is configured in the "Control Panel" in the PROFINET settings (NTP).

## 7.19. Cyclic Time-of-day synchronization

Pay attention to time zones as well as Daylight saving time / Standard time

Name	Communication driver	HMI time synchronization mode	Station	Partner	Node
TP-CPU	SIMATIC S7 1200	Slave	S7-1200-Station_1	PLC_1	CPU 1214C DC/DC/...
-<Add new>					

### Projected time-of-day synchronization

The S7-1200/1500 operating systems are capable of easily synchronizing the time cyclically with an operating device.

👉 Not every operating device supports this form of time-of-day synchronization. Basic-/Comfort-Panels, various Multipanels, PC-Runtime etc., support this.  
→ see online help or the operating device's technical data

### To be noted

- For several configured connections → only one connection can be "Slave"
- If a connection is activated as "Slave" → then the global area pointer "Date/time PLC" cannot also be used
- For the controller, if the type of protection "Complete protection" is activated for the connection → then the correct "password" must also be configured for the connection for time-of-day synchronization in the operating device

👉 For this method of time-of-day synchronization, the system time (world time) is synchronized. For that reason, the same time zone as well as daylight saving time and standard time must be set in the panel and the CPU

## 7.20. Cyclic Time-of-day synchronization by means of area pointer

The image illustrates the configuration for cyclic time-of-day synchronization. The top part shows the 'Connections' table in SIMATIC Manager, where the 'Area pointer' section is expanded to show a table of global area pointers. The bottom part shows a ladder logic diagram for the 'RD\_LOC\_T' function block, which is connected to a data block 'DB\_HMI\_sync'. The data block contains several fields, including 'Date\_Time' of type DTL, which is used for time synchronization.

Connection	Display name	PLC tag	Length	Acquisition...	Acquisition...
<Undefined>	Project ID	<Undefined>	1	Cyclic conti...	<Undefined>
<Undefined>	Screen number	<Undefined>	5	Cyclic conti...	<Undefined>
TP-CPU	Date/time PLC	DB_HMI_sync.Date_Time	6	Cyclic conti...	1 h

Name	Data type
Static	
Date_Time	DTL
YEAR	UInt
MONTH	USInt
DAY	USInt
WEEKDAY	USInt

### Time-of-day synchronization with area pointer

For the time-of-day synchronization by means of global area pointer, the realtime of the panel is cyclically updated.

### Principle

You store a time stamp of the CPU realtime clock in a tag of the data type DTL which is declared in a data block. The operating device cyclically reads out this tag from the controller via an area pointer and so synchronizes its time at the same time.



The CPU program must constantly or cyclically update the time stamp stored in the tag to ensure that the time stamp it contains is always current whenever the operating device reads out the tag.

### Data Format of the Time Stamp

The CPU has a buffered realtime clock that can be read out by means of the system function "RD\_SYS\_T" (system time) and "RD\_LOC\_T" (local time). This delivers the current time stamp in the "DTL" format which can also be interpreted by the operating device.

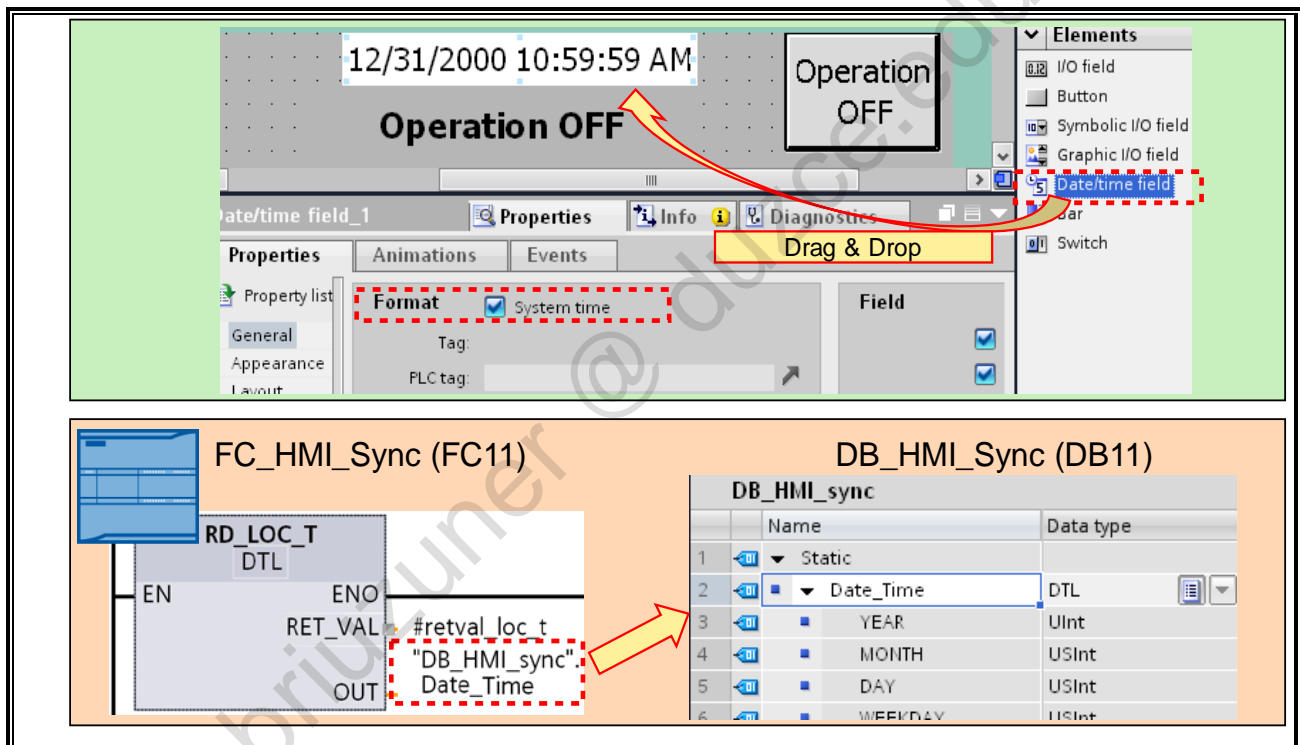
### HMI Configuration

The adoption of the relevant time from the CPU is configured via the area pointer "Date/time PLC". The configured time in the column "Acquisition cycle" determines the time interval in which the data stored in the CPU is read. This time should not be configured as short as possible, rather be configured as short as necessary in order to keep the communication load as small as possible.



Until the first acquisition cycle has elapsed, the operating device still works with the operating system's initial time.

### 7.20.1. Exercise 5: Configuring the Time-of-day synchronization CPU → TP by means of a global area pointer



#### Task

Configure the cyclic adoption of the CPU time. For this, read out the local CPU time with the function "RD\_LOC\_T" and configure the area pointer "Date/time PLC" in the HMI.

#### What to do PLC part

1. Create the new DB "DB\_HMI\_Sync"
2. In the new DB, create the tag "Date\_Time" of the type DTL
3. Create the new function "FC\_HMI\_Sync"
4. With the function "RD\_LOC\_T", read out the current CPU local time and save it in the previously created DB tag
5. Call the function "FC\_HMI\_Sync" in cyclic OB
6. Transfer the PLC program to the CPU

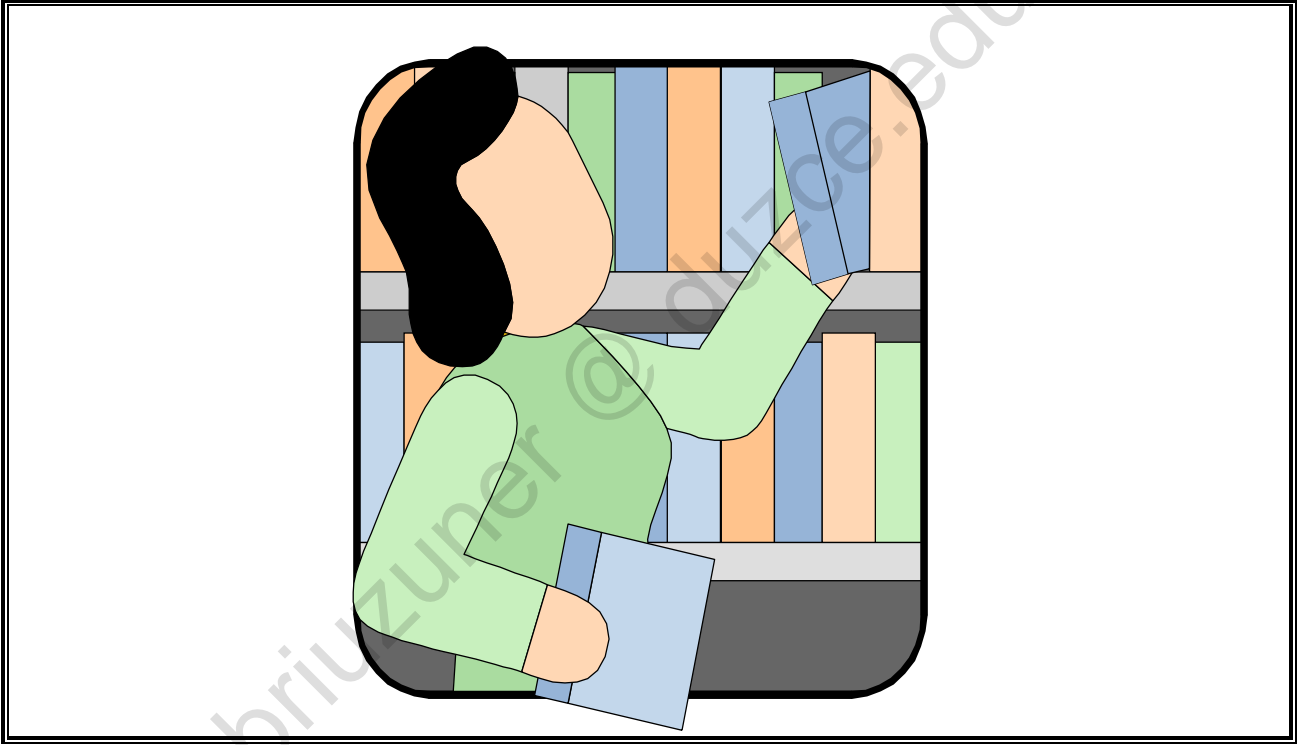
#### What to do HMI part

1. In the HMI project, open the "connections" and there the tab "area pointer"
2. Configure the "global area pointer" "Date/time PLC" by specifying the DB tag "DB\_HMI\_Sync".Date\_Time with an acquisition cycle of "1min"
3. In the screen "Conveyor", create a date/time field in the mode output and output the touchpanel's system time (see picture)
4. Transfer the project to the touchpanel and check whether the correct time is output



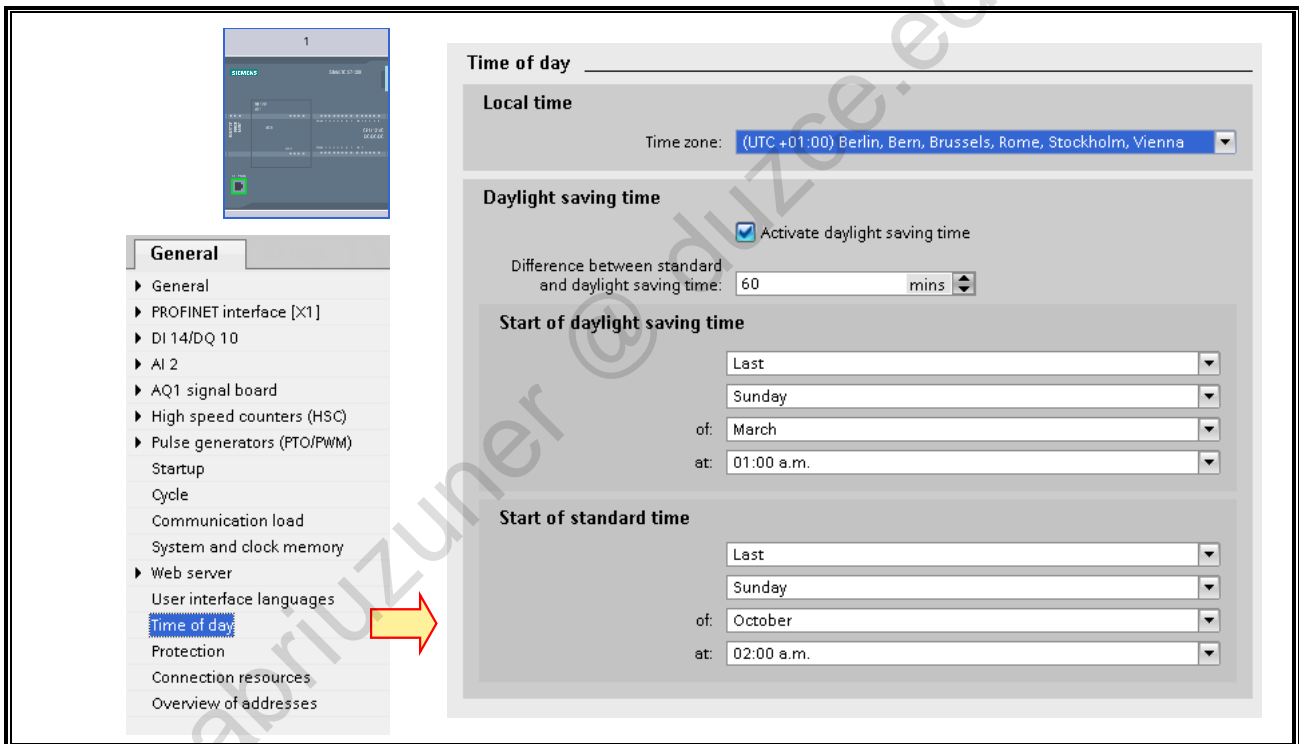
Until the first acquisition cycle has elapsed, the operating device still works with the operating system's initial time.

## 7.21. Additional information





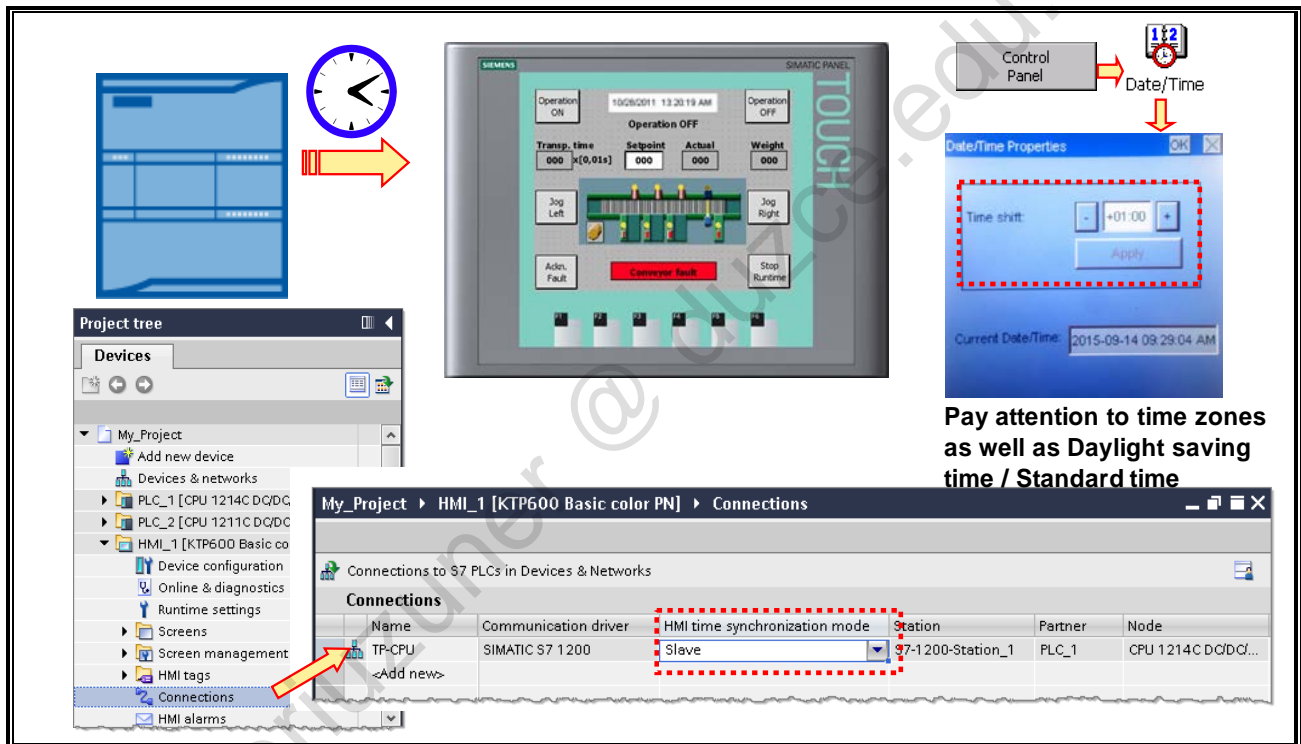
### 7.21.1. Daylight saving time / standard time change



#### Daylight saving time / standard time

In properties of the PLC an automatic activation of daylight saving time / standard time can be set.

## 7.21.2. Additional exercise: Adopting the time from the CPU



## Task description

In the Connections, use the setting Slave for "HMI time synchronization mode" for the time synchronization and not the global area point "Date/time PLC".

## What to do PLC part

1. Delete the call of the function "FC\_HMI\_Sync" in cyclic OB
2. Transfer the PLC program into the CPU

## What to do HMI part

1. In the HMI project, open the "connections" and there the tab "area pointer"
2. Delete the "global area pointer" "date/time PLC"
3. Set the HMI time synchronization mode to slave (see picture)
4. In the panel's settings, set the correct value for "time shift" "Stop Runtime > Control Panel > Date/Time". (pay attention to the time shift and +1h for daylight saving time)
5. Transfer the project to the touchpanel and check whether the correct time is output

# Contents

<b>8.</b>	<b>Technology objects.....</b>	<b>8-2</b>
8.1.	Objectives .....	8-2
8.2.	Task description: Commissioning a PID controller and controlling a stepper motor .....	8-3
8.3.	Introduction to pulse generators .....	8-4
8.3.1.	Pulse Width Modulation (PWM) .....	8-5
8.3.2.	Pulse Train Output (PTO) .....	8-6
8.3.3.	Configuring a pulse generator.....	8-7
8.4.	Introduction to the PID (Controller) .....	8-8
8.4.1.	Implementation of a PID controller in the S7-1200 .....	8-9
8.4.2.	Creating a "PID" technology object.....	8-10
8.4.3.	"PID_Compact" call.....	8-16
8.4.4.	Using the commissioning panel .....	8-17
8.5.	Task description: Controlling the capacitor voltage .....	8-18
8.5.1.	Exercise 1: Creating and configuring the "PID" technology object .....	8-19
8.5.2.	Exercise 2: Calling the "PID_Compact" block in the cyclic interrupt "Cyclic Interrupt" .....	8-21
8.5.3.	Exercise 3: Commissioning the PID controller.....	8-22
8.6.	Introduction to the "Axis" technology object (controlling the stepper motor) .....	8-24
8.6.1.	Principle of axis control .....	8-25
8.6.2.	Configuring a PTO output (1) .....	8-26
8.6.3.	Configuring a PTO output (2).....	8-27
8.7.	Creating a "Positioning Axis" technology object .....	8-28
8.7.1.	Properties of "Axis": Configuration .....	8-29
8.7.2.	Properties of "Axis": Commissioning.....	8-37
8.7.3.	Activating the commissioning panel.....	8-38
8.7.4.	Properties of "Axis": Diagnostics.....	8-40
8.7.5.	Blocks for axis control .....	8-43
8.8.	Task description: Controlling a stepper motor .....	8-44
8.8.1.	Exercise 4: Activating (enabling) PTO 1 of the CPU .....	8-45
8.8.2.	Exercise 5: Creating and configuring the technology object "Axis" .....	8-46
8.8.3.	Exercise 6: Commissioning "FB_Turntable" .....	8-50
8.8.4.	Exercise 7: Starting the axis and monitoring the statuses with the diagnostic panel .....	8-52

## 8. Technology objects

### 8.1. Objectives

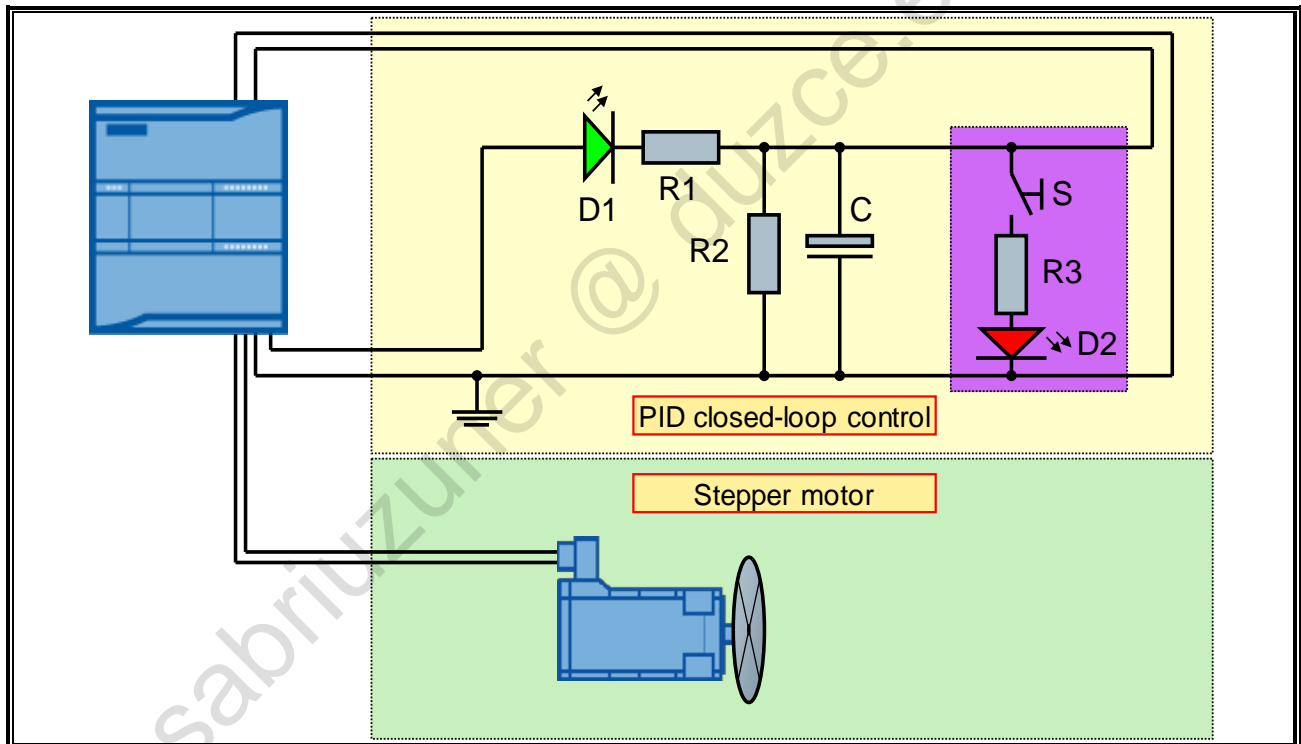
**At the end of the chapter the participant will ...**

- ... know the difference between PWM and PTO outputs.
- ... be familiar with the structure of a simple closed-loop control.
- ... be familiar with the procedure for creating a PID controller.
- ... be able to commission a PID controller with automatic optimization.
- ... understand the principle of controlling a stepper motor.
- ... be able to commission a stepper motor for positioning a turntable.

#### Objectives

In this chapter, the technology objects "PID (Control)" and "Axis" are dealt with. The necessary, theoretical basics and the procedure for configuring are presented.

## 8.2. Task description: Commissioning a PID controller and controlling a stepper motor



### Task description

In the first step, a PID controller is to be commissioned. It is to control the voltage at Capacitor C to a constant voltage of 10V, even when a disturbance, in the form of a load resistance R3 is switched in via the switch S.


The manipulated variable (PWM output) is thereby controlled by the "PID\_Compact" controller block, by evaluating the fed back capacitor voltage at the analog input of the CPU.

After the control loop has been commissioned with the technology object PID, the stepper motor of the training device is then to be commissioned. For this, the technology object "Positioning Axis" is to be used, which is to be configured by you. On the hardware side, the so-called "PTO" output of the CPU is used for this as well as a Boolean output for the direction setting.

There are Motion Control instructions for the programming of the axis control.

Your task is to commission a prefabricated function block and to monitor the movement sequences on an online screen.

### 8.3. Introduction to pulse generators



Description	Default output assignment		
		Pulse	Direction
PTO 1	Integrated in CPU	Q0.0	Q0.1
	Signal board	Q4.0	Q4.1
PWM 1	Integrated in CPU	Q0.0	--
	Signal board	Q4.0	--
PTO 2	Integrated in CPU	Q0.2	Q0.3
	Signal board	Q4.2	Q4.3
PWM 2	Integrated in CPU	Q0.2	--
	Signal board	Q4.2	--

#### Pulse generators

All CPU types of the SIMATIC S7-1200 series are equipped with two pulse generators. These can be used independent of each other either for Pulse Width Modulation (PWM) or pulse train (Pulse-Train-Output – PTO).

The two pulse generators are assigned specific digital outputs by default (see table above) and are activated in the device configuration of the respective CPU. Integrated CPU outputs or the outputs of an optional signal board can be used. If the addresses of the outputs were changed, the addresses correspond to the newly assigned ones.

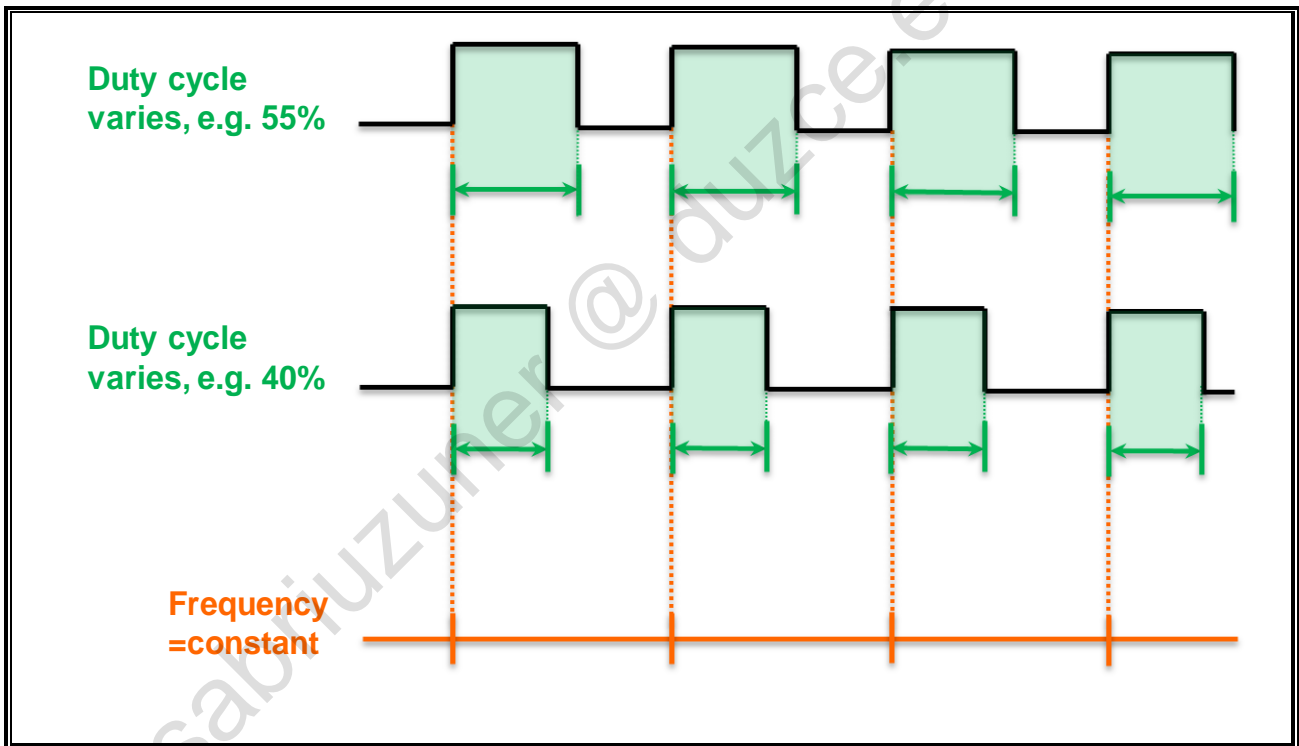
Regardless, PTO1/PWM1 always uses the first two digital outputs of the configured addresses and PTO2/PWM2 uses the next two digital outputs, either on the CPU or the inserted signal board. When an output is not required for a pulse function, it is available for other purposes.

The maximum pulse frequency of the pulse generators is 100 kHz for the digital outputs of the CPU and 20 kHz or 200 kHz for the digital outputs of the signal board.



STEP7 gives no warning when an axis is configured with a maximum speed or frequency that exceeds this hardware limitation. This can lead to problems in the application. You must always make sure that the maximum pulse frequency of the hardware is not exceeded.

### 8.3.1. Pulse Width Modulation (PWM)



#### Pulse Width Modulation

With the Pulse Width Modulation (PWM), the cycle time, that is, the time from one positive edge to the next, remains constant. The duty cycle (pulse width), however, represents the variable size of the modulation.

The duty cycle can be specified as hundredth of the cycle time (0 – 100), as thousandth (0 – 1000), as ten thousandth (0 – 10000) or as S7 analog format. The pulse duration can lie between 0 (no pulse, always off) and full scale (continuous pulse, always on).

Since the duty cycle can lie between 0 and full scale with the PWM, it provides a digital output that in many ways is the same as an analog output. The PWM output can, for example, serve to control the speed of a motor from standstill to full speed or it can be used to control the position of a valve from closed to fully open.

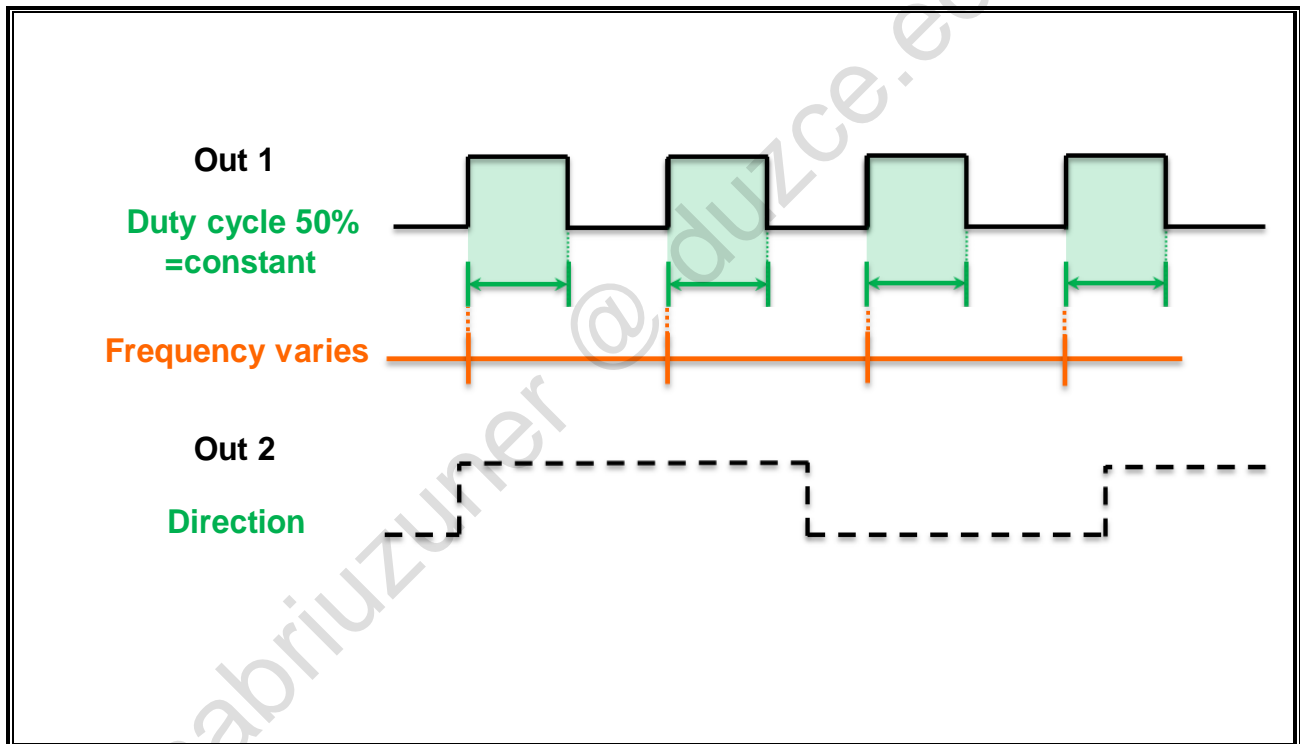
#### Controlling PWM outputs

The "CTRL\_PWM" block is used to control PWM outputs.



The first time the target system switches to RUN, the PWM duty cycle ratio is set to the start value specified in the device configuration. To change the pulse duration during program runtime, the desired values are written in the output addresses ("Start address") specified in the device configuration, for example, with the command "MOVE".

### 8.3.2. Pulse Train Output (PTO)



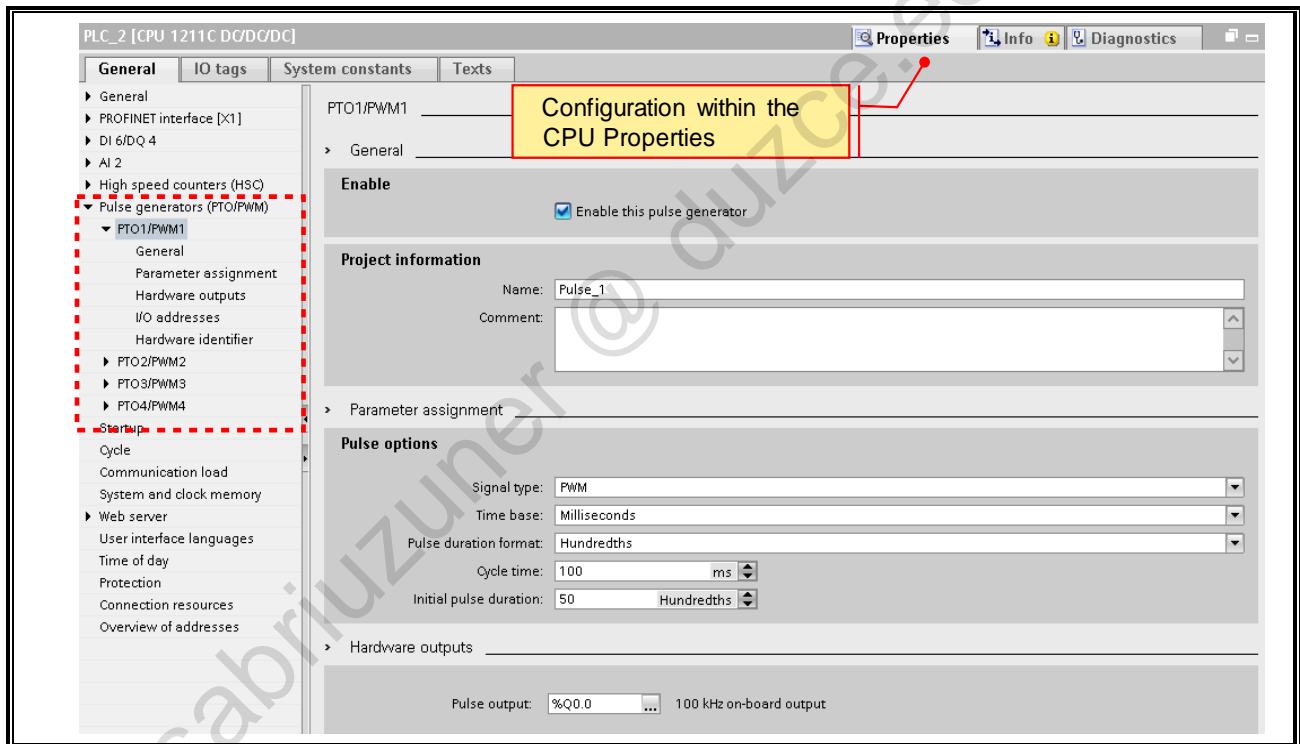
#### Pulse Train Output

Unlike the PWM, the Pulse Train Output has a fixed duty cycle of 50% and a variable frequency. Through this, the speed of the connected drive can be controlled.

The turning direction of the drive can be specified via the direction output.



### 8.3.3. Configuring a pulse generator

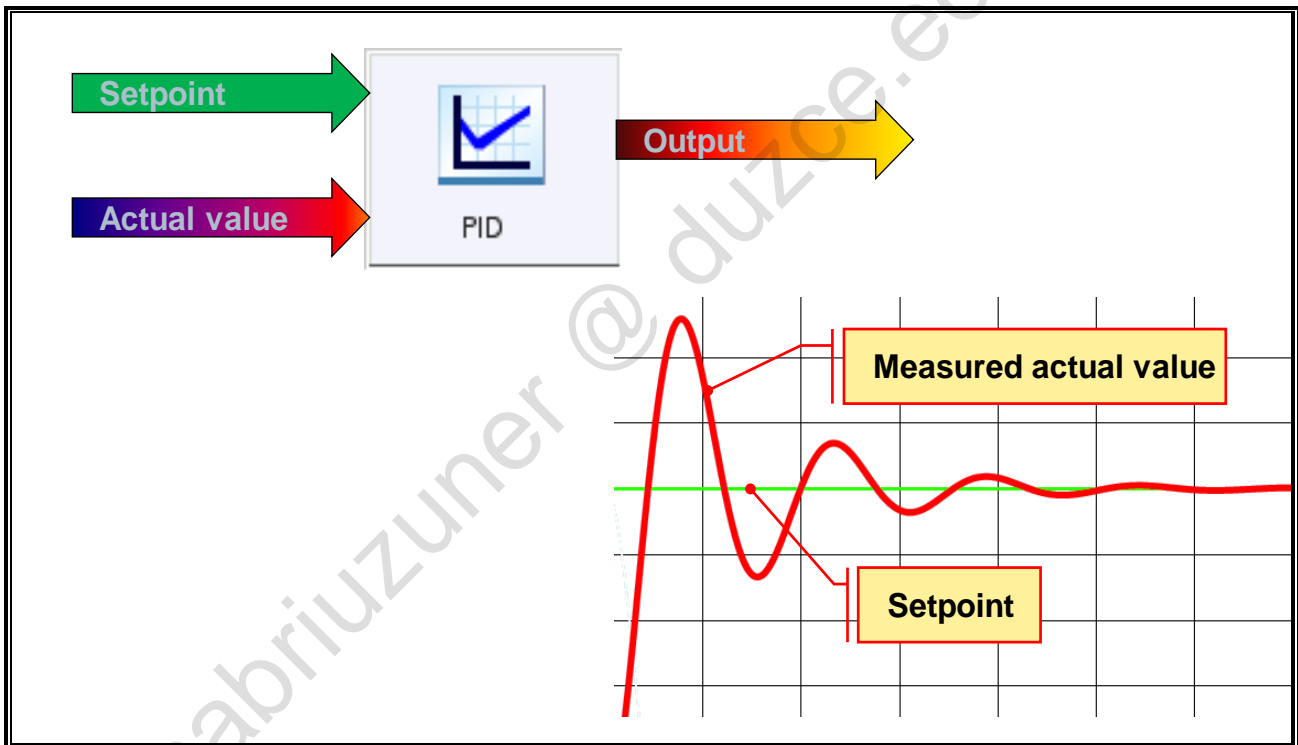


#### Configuring a Pulse generator

To activate the pulse generator, you need to proceed as follows:

1. First, the respective pulse generator must be activated
2. Another name than the default assigned name as well as a comment can be entered
3. Set pulse options
  - Use as PWM or PTO output
  - Time base
  - Format for the pulse duration
  - Cycle time specification
  - Initial pulse duration
4. The hardware outputs used by the pulse generator are displayed in the field "Pulse output"

## 8.4. Introduction to the PID (Controller)



### PID (Controller)

PID stands for "Proportional Integral Differential". A PID controller has a proportional component, an integrating component and a differentiating component. For each of the three components, a specific equation is in force:

- The equation of the proportional component results in a value that is proportional to the control deviation
- The result of the integral equation increases with the duration of the control deviation
- The speed of the control deviation influences the differential component; the steeper the increase or fall of the change, the greater the D-component is

The three equations are then combined and result in the output value (Output).

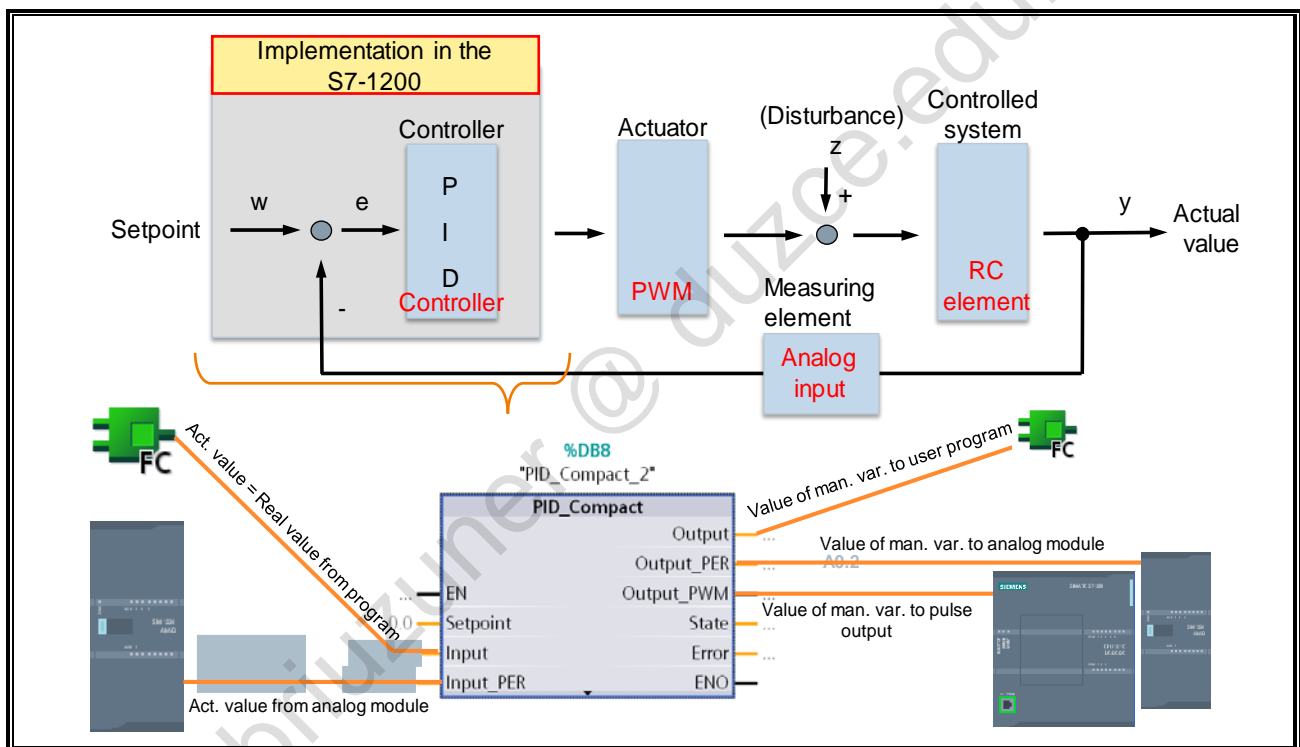
PID controllers are used in industry, for example, to control the temperature of welding systems when it is important to retain a constant temperature value despite possible disturbances.

Put very simply, a PID controller serves to align a changing, measured actual value with a setpoint value as quickly as possible and as exactly as possible.

This is done by readjusting the output variable whereby the overshoots and undershoots keep getting smaller until the actual value equals the setpoint value as exactly as possible.

For this, there is a wizard in STEP7 which, in conjunction with the S7-1200, enables you to configure the necessary settings of a controlled system quickly and easily as well as without extensive prior knowledge.

### 8.4.1. Implementation of a PID controller in the S7-1200



#### PID controller in the S7-1200

Based on the example of a complete control system, the picture above shows the PID controller implemented in the SIMATIC 1200 station in symbolic representation and the block that results from it.

In the S7-1200, the actual controller of a PID control system is implemented. For this, TIA Portal provides a prefabricated block "PID\_Compact" which can be inserted in the user program and then assigned.

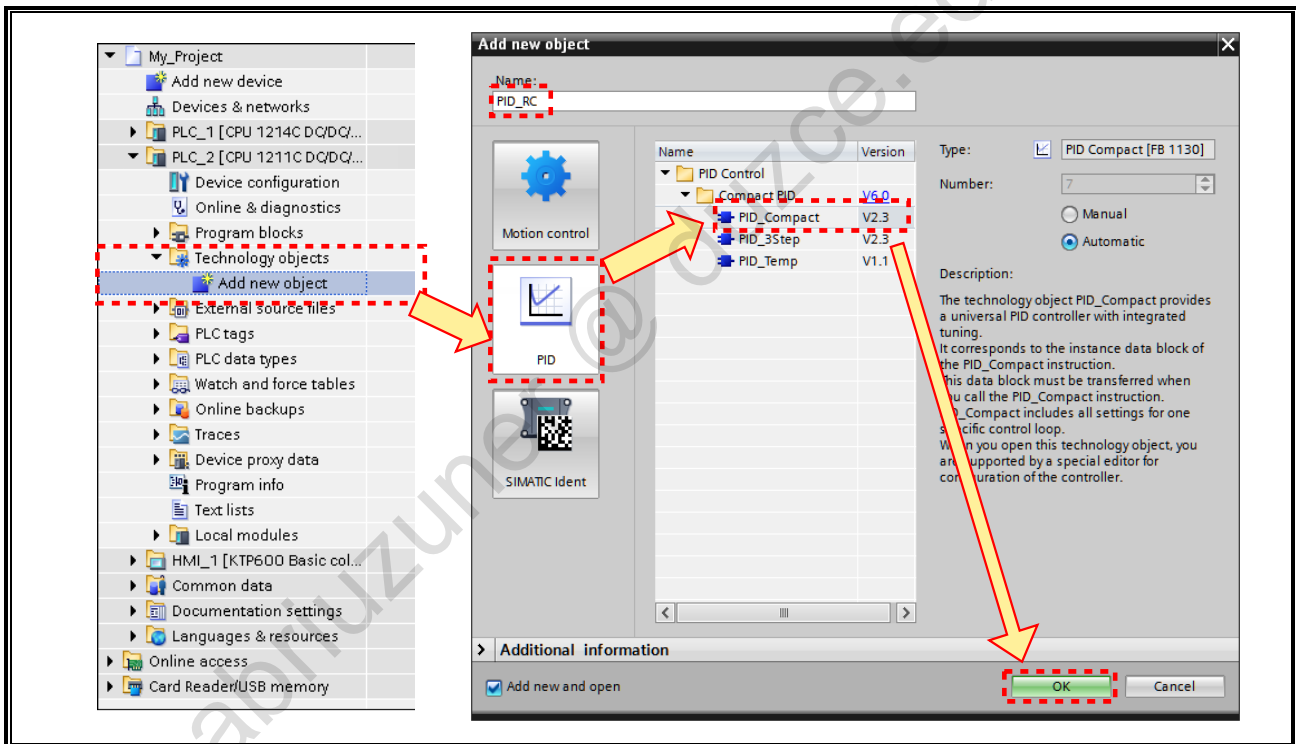
At the same time, a "PID" technology object is available with which the controller can be configured in the user program and then commissioned.

**! The switch output for the pulse width modulation is controlled by the instruction PID\_Compact. The pulse generators integrated in the CPU are not used.**

#### Closed-loop Control

- **Blocks:**  
In the simplest case, a PID control system consists of a PID controller, an actuator as well as the system to be controlled. The output signal of the controlled system is fed back via a measuring element on the PID controller.
- **Signals/Values:**  
In the simplest case, distinction is made in a PID control system between the setpoint value ( $w$ ), the input value ( $e$ ), the correcting variable that results from it. Together with an influencing disturbance ( $z$ ), the actual value ( $y$ ) results from this in the closed-loop control, which is once again fed back to the PID controller via the measuring element.

## 8.4.2. Creating a "PID" technology object



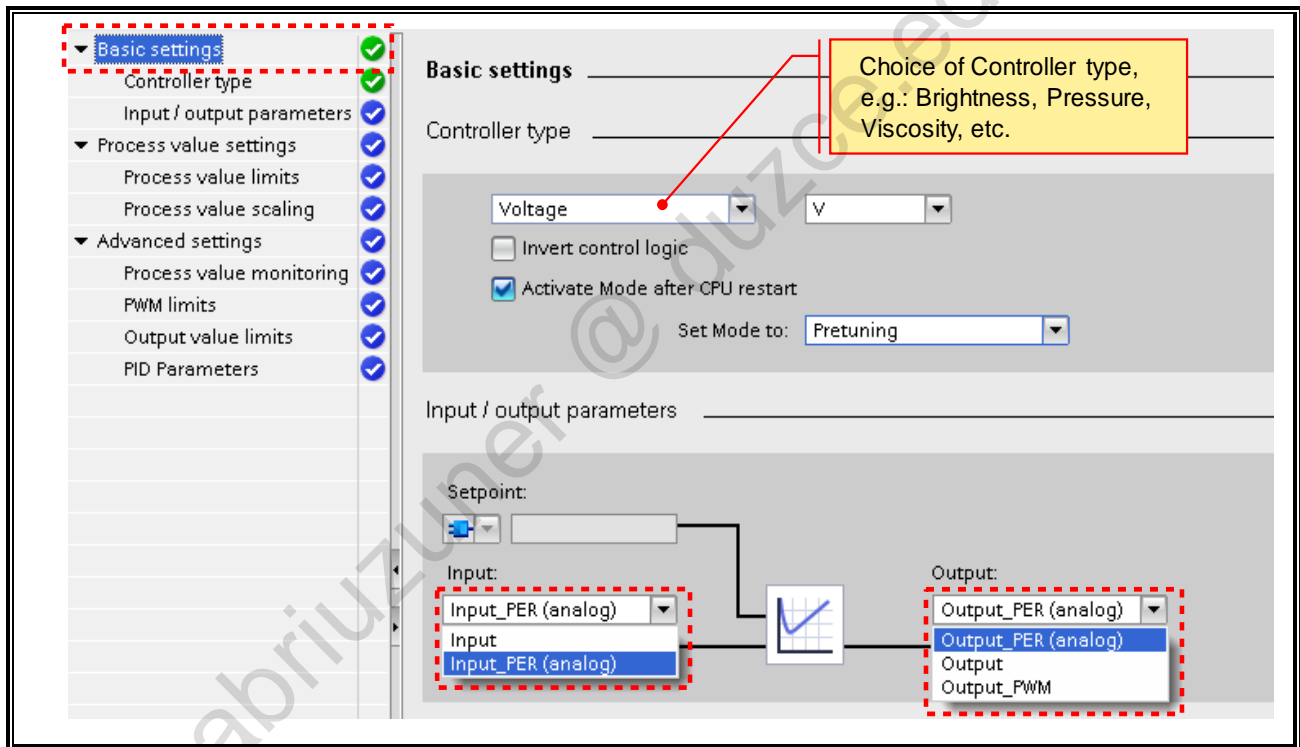
### Creating a PID controller

After a PID controller has been created under "Technology objects", the view is transferred to the wizard. It uses the following identifiers:

- The settings were configured successfully
- The settings are occupied only with default values, function is not hindered by this
- The settings are still faulty

As well, a function block for the PID controller is automatically created which contains all input values and output values in its interface.

## 8.4.2.1. Configuring a PID controller (1) - Basic settings

**Basic settings**

The configuration of the basic settings of the PID controller offers the following options.

**Type of controller**

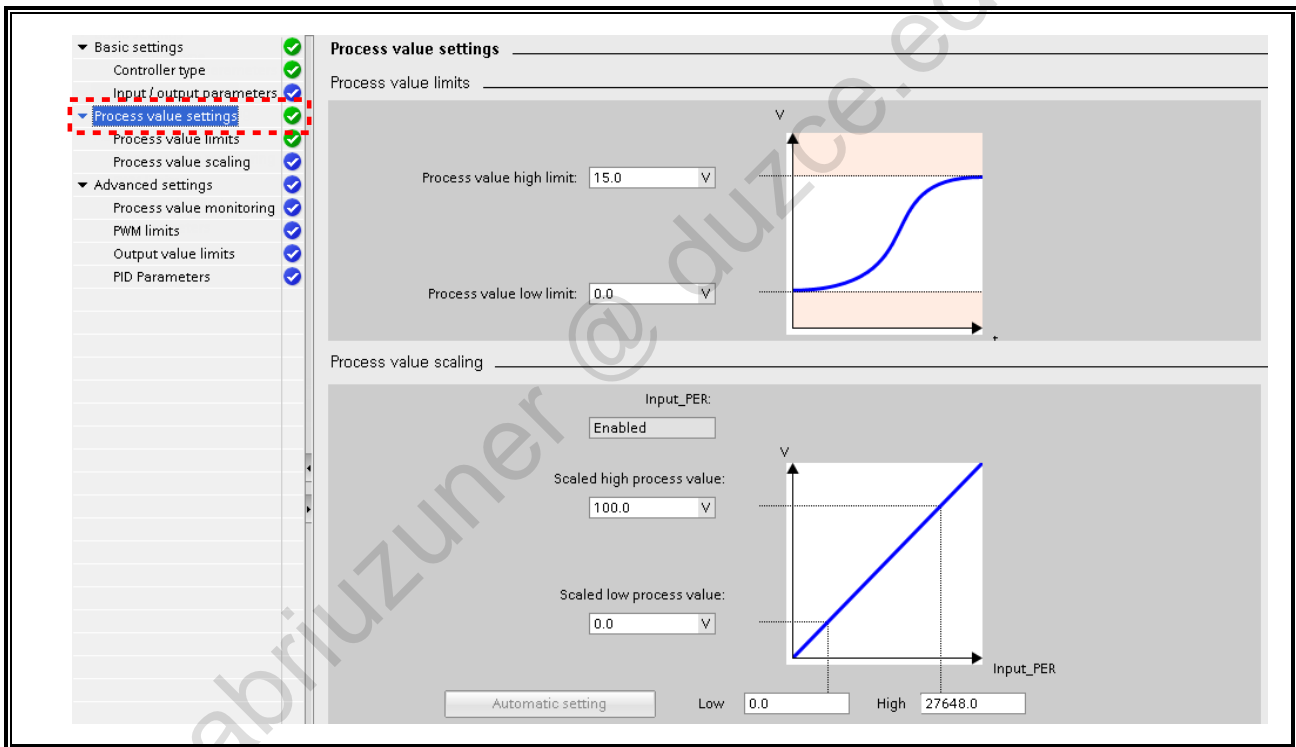
The preselection "controller type" sets the desired unit for the controller.

If the checkbox "invert the control logic" is activated (checked), it causes an increase of the manipulated value when a decrease of the actual value occurs (for example, falling water level through an increase of the valve position of the outlet valve or decreasing temperature through an increase of the cooling capacity).

**Setting the input / output parameters**

- **Setpoint:**  
Choose whether the value at the function block or the value of the instance DB is to be used (insofar as it exists and is only available in the Inspector window of the program editor)
- **Input:**  
Choose whether the input parameter "Input" or "Input\_PER" is to be used.
  - "Input" is used when an actual value from the user program is to be used.
  - "Input\_PER" is used when the actual value of an analog input is to be used.
- **Output:**  
Select the manipulated value output of the instruction "PID\_Compact". The following possibilities are available:
  - Output: uses a variable of the user program as the manipulated value output. (Real format)
  - Output\_PER: uses an analog output as the manipulated value output (analog output value)
  - Output\_PWM: uses a digital switch output and controls it via a pulse width modulation (the manipulated value is formed via variable switch-on and switch-off times)

## 8.4.2.2. Configuring a PID controller (2) - Process value settings

**Process value settings**

For the configuration of the process value settings, the following options are available.

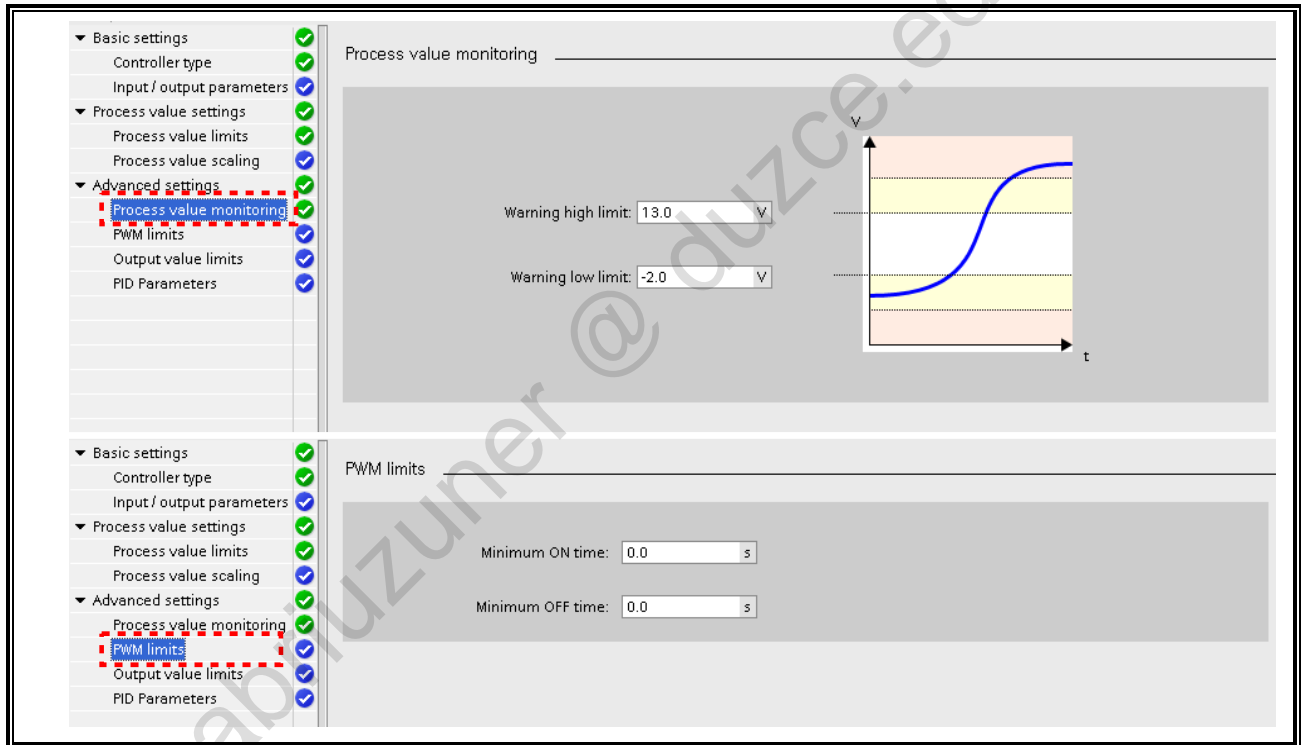
**High limit and low limit**

They define the absolute upper and lower limit of the process value. During operation, as soon as these limits are exceeded or fall below, the controller switches off and the value of the manipulated variable is set to 0%.

**Scaling**

Through scaling, the process values (actual values) are defined by a lower and an upper value pair. Each value pair consists of the value of the analog input and the physical value of the respective scaling point. Depending on the configuration of the basic setting, a process value of the user program can also be used instead of the analog value of the analog input.

## 8.4.2.3. Configuring a PID controller (3) - Process value monitoring and PWM limits

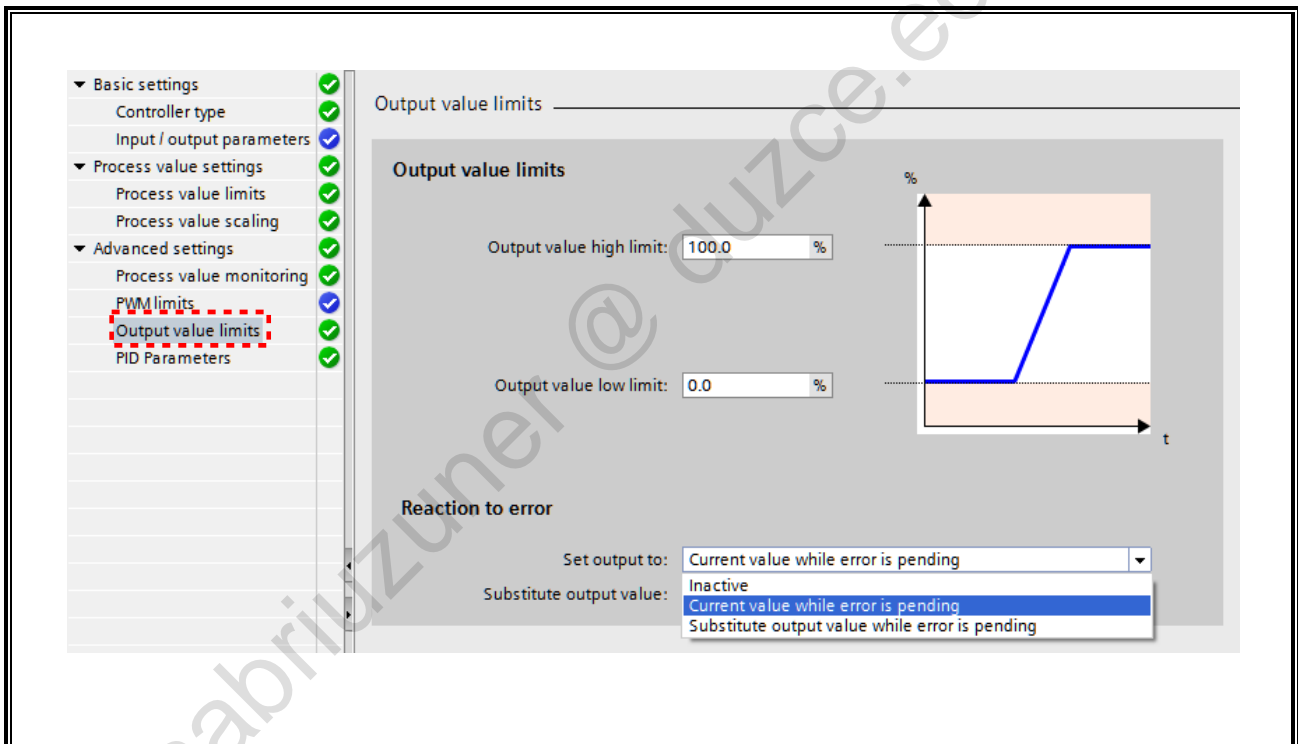
**Process value monitoring**

The monitoring of the process value is preset by two limits. If, during controller runtime, the process value exceeds the high limit or falls below the low limit, a message is output at the Boolean output parameters "InputWarning\_H" or "InputWarning\_L" of the block "PID\_COMPACT".

**PWM limits**

In the window "PWM limits", the minimum permitted switch ON and switch OFF times of the pulse width modulation are set. The minimum ON and OFF times can be extended when, for example, the number of switching cycles is to be reduced. This makes sense, for example, for the delayed control of a tank level when you want to avoid the valve reacting to every small change in the level.

#### 8.4.2.4. Configuring a PID controller (4) - Output value limits



#### Output value limits

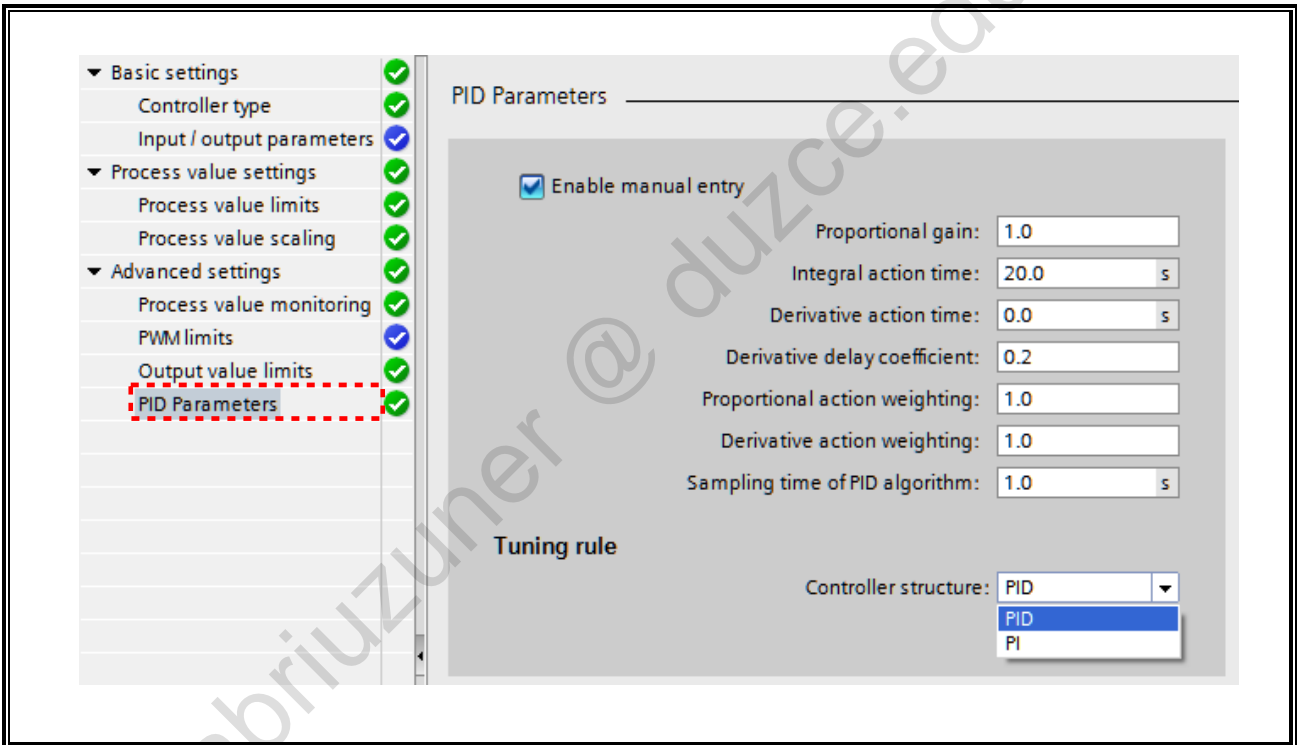
In the configuration window "Output value limits", the absolute limits of the manipulated value are specified. Neither in manual mode nor in automatic mode can absolute manipulated value limits be exceeded nor can they fall below. If in manual mode, a manipulated value is specified outside of the limits, the effective value in the CPU is limited to the configured limits.

#### Reaction to error

If an error occurs during processing (output parameter ERROR = TRUE), then a substitute value can be output, the old value can be held (pending) or the output can be deactivated (inactive) at OUTPUT.



8.4.2.5. Configuring a PID controller (5) - PID parameters

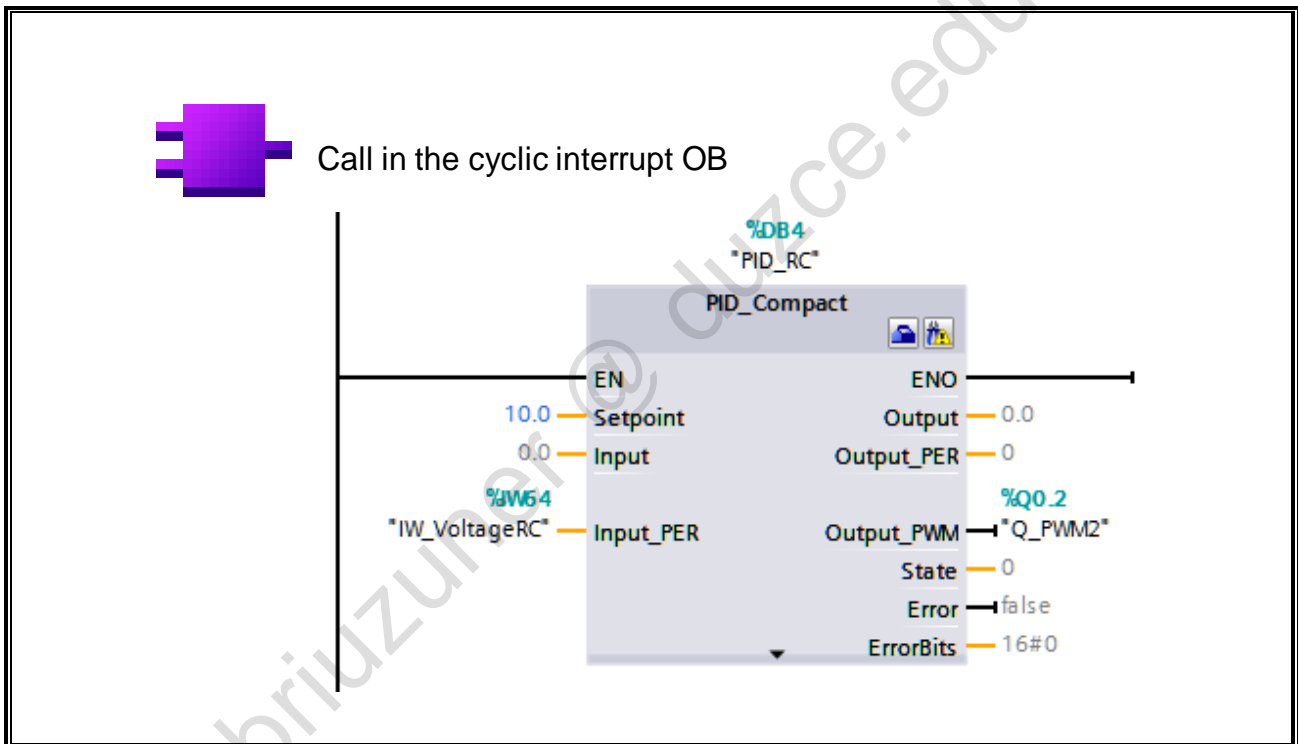


PID parameters

The PID parameters are grayed out by default; they can, if need be, be changed. This, however, is only recommended for users with experience in PID control.

The PID parameters are determined automatically when the automatic auto-tuning has been run through.

### 8.4.3. "PID\_Compact" call



#### PID\_Compact

The instruction PID\_Compact provides a PID controller with integrated optimization for automatic and manual mode operation.

#### Call

PID\_Compact is called in the time base of the cycle time of the calling OB. This must be constant to ensure that the PID controller can sample in equidistant intervals. For that reason, PID\_Compact is preferably called in a cyclic interrupt OB since the cycle time in the cyclic user program can vary significantly because of conditional program execution, for example.

#### Start-up behavior

When the CPU starts up, it starts PID\_Compact in the operating mode in which it was last active.

#### Monitoring the sampling time PID\_Compact

Ideally, the sampling time corresponds to the cycle time of the calling OB. The instruction PID\_Compact measures, in each case, the interval between two calls. That is the current sampling time. Every time the operating mode changes and in the first start-up, the mean value of the first 10 sampling times is formed. When the current sampling time deviates too greatly from this mean value, an error occurs (Error = 0800 hex) and PID\_Compact switches into the "inactive" mode.

During tuning (optimization), the following conditions put PID\_Compact in the "inactive" mode:

- New mean value  $\geq 1.1 \times$  old mean value
- New mean value  $\leq 0.9 \times$  old mean value

In automatic mode, the following conditions put PID\_Compact in the "inactive" mode:

- New mean value  $\geq 1.5 \times$  old mean value
- New mean value  $\leq 0.5 \times$  old mean value

### 8.4.4. Using the commissioning panel

The screenshot displays the SIMATIC Manager commissioning panel for a PID controller. The interface is divided into several sections:

- Measurement:** A graph showing the PID controller's response over time. The y-axis is labeled 'Setpoint' and ranges from 0.0 to 14.0. The x-axis is labeled '[s]' and ranges from 0.0 to 40.0. The graph shows a step change in the setpoint, with the controller output (red line) following the setpoint (green line) and the scaled input (blue line).
- Tuning mode:** A section with a 'Start' button and a 'Stop' button. A 'Sampling time: 0.3 s' is displayed. A red dashed box highlights the 'Start' and 'Stop' buttons.
- PID Parameters:** A table with the following data:
 

Name	Data t...	Addr...	Color	Scaling group	Min. y scale	Max. y scale
1	Setpoint		Real		0	15
2	ScaledInput		Real		0	15
3	Output		Real		0	100
- Tuning status:** A section with a 'Progress' bar and a 'Status: System tuned.' indicator with a green checkmark.
- Online status of controller:** A section with 'Setpoint: 10.0' and 'Input: 9.975766'. The 'Controller state' is 'Enabled'.
- PID Parameters:** A section with 'Upload PID parameters' and 'Go to PID parameters' buttons.

Annotations in yellow boxes highlight key features:

- Select Tuning (optimization) and Start:** Points to the 'Start' button in the 'Tuning mode' section.
- Monitor Setpoint, Scaled input and Output:** Points to the graph area.
- Adopting the optimized PID parameters in the project:** Points to the 'Upload PID parameters' button.
- Display of the current controller statuses -> Manual mode possible:** Points to the 'Controller state' indicator.

#### Commissioning panel

In the configuration of the PID controller, you can carry out an automatic tuning (optimization) and you can monitor the current measured values.

#### Commissioning

As soon as measurement is switched on through a click on "Start", actual value and setpoint value as well as manipulated value are graphically represented (see picture).

Under "tuning mode" the auto-tuning can be started. This must first occur in the first start-up. In the second step, you can then tune in the operating point. The status and the progress of the running tuning (optimization) can be read from the bar graph.

Required for automatic fine tuning:

- PID\_Compact is called in a cyclic interrupt OB
- "Manual mode" is deactivated
- The difference between current actual value and setpoint is >50%

The operation can take some minutes. During this time, you cannot work with the CPU.

Subsequently, the ascertained data must be adopted in the project via "Upload PID parameters".

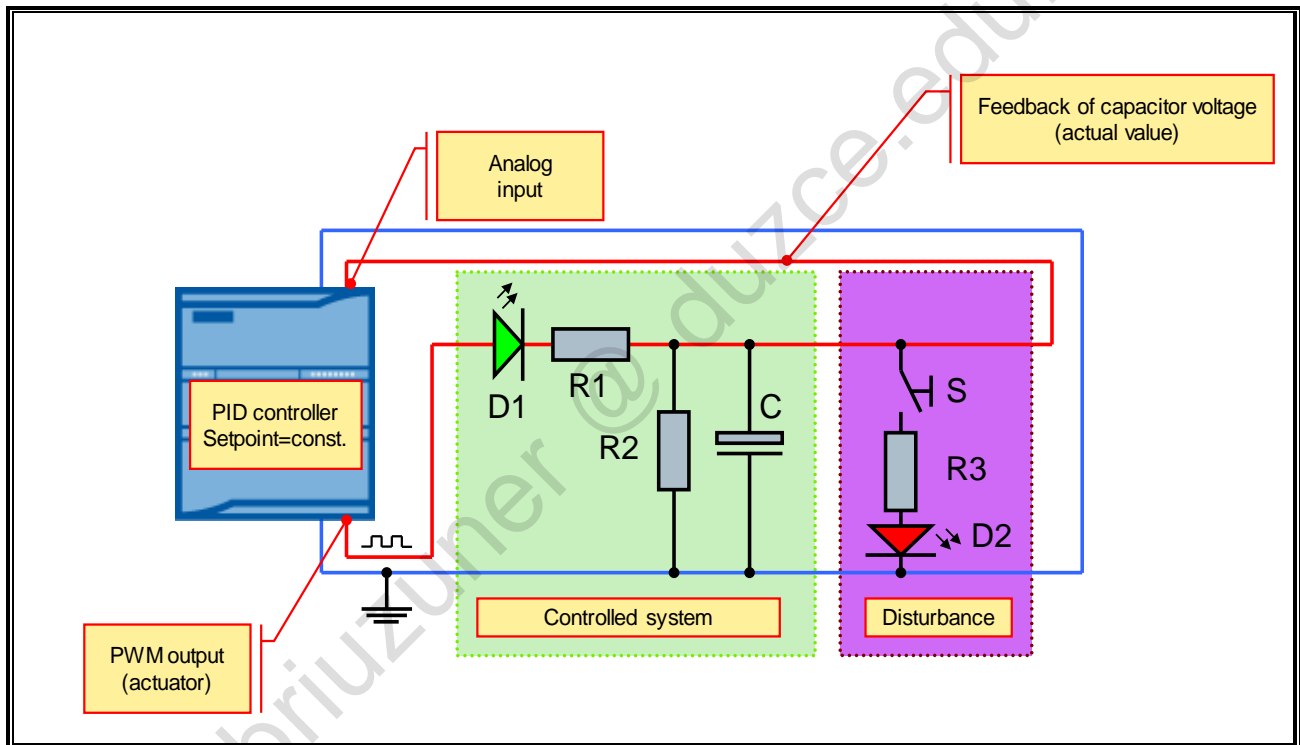
Through "Online status of controller" you can monitor the current actual value, the setpoint as well as the output in % and you can specify a manual manipulated value.

#### Representation

Using the following buttons, you can stretch or compress the value axes, select a type of representation for the value diagram, shift the view etc.



## 8.5. Task description: Controlling the capacitor voltage



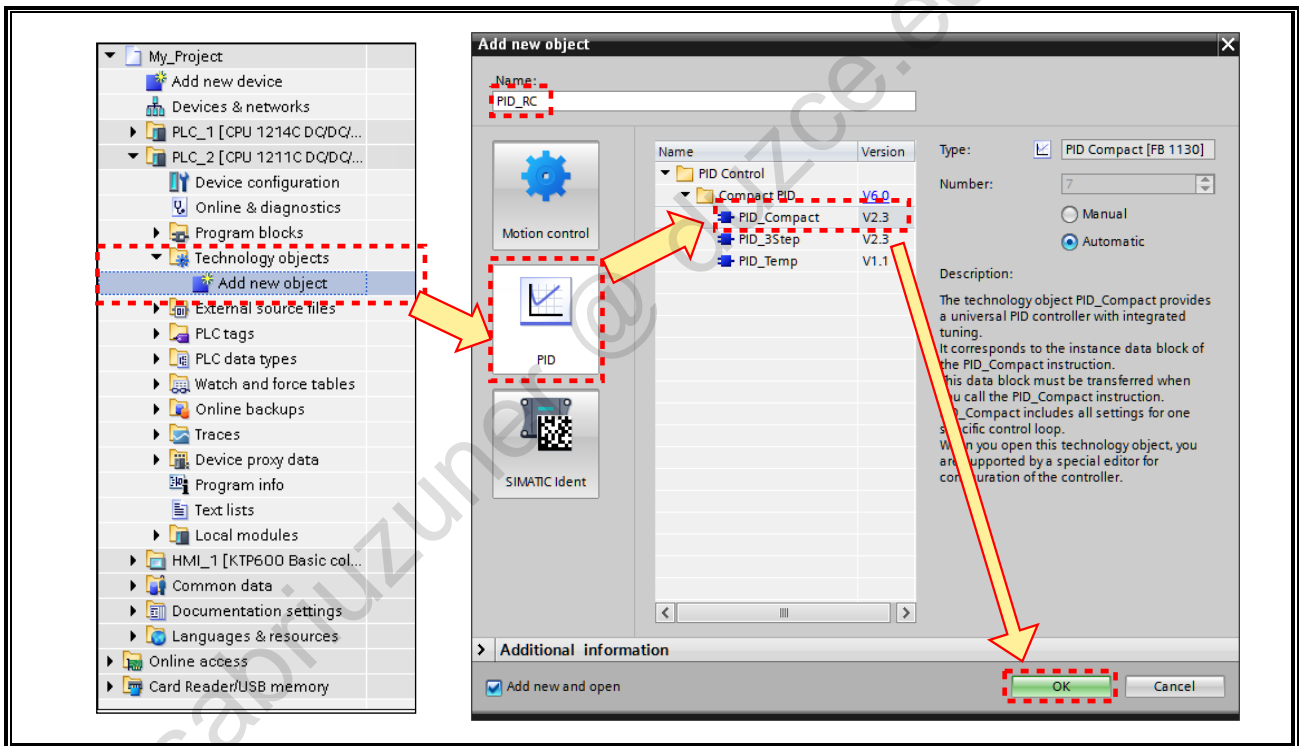
### Task description

In the first step, the PLC with CPU 1211C is to be commissioned.

Then, a PID controller is to be commissioned. This is to control the voltage at Capacitor C to a constant voltage of 10.0V, even when a fault in the disturbance of a load resistance R3 is switched in via the switch S.

The manipulated variable (PWM output) is controlled by the "PID\_Compact" controller block, by evaluating the fed-back capacitor voltage at analog input AI0.

### 8.5.1. Exercise 1: Creating and configuring the "PID" technology object

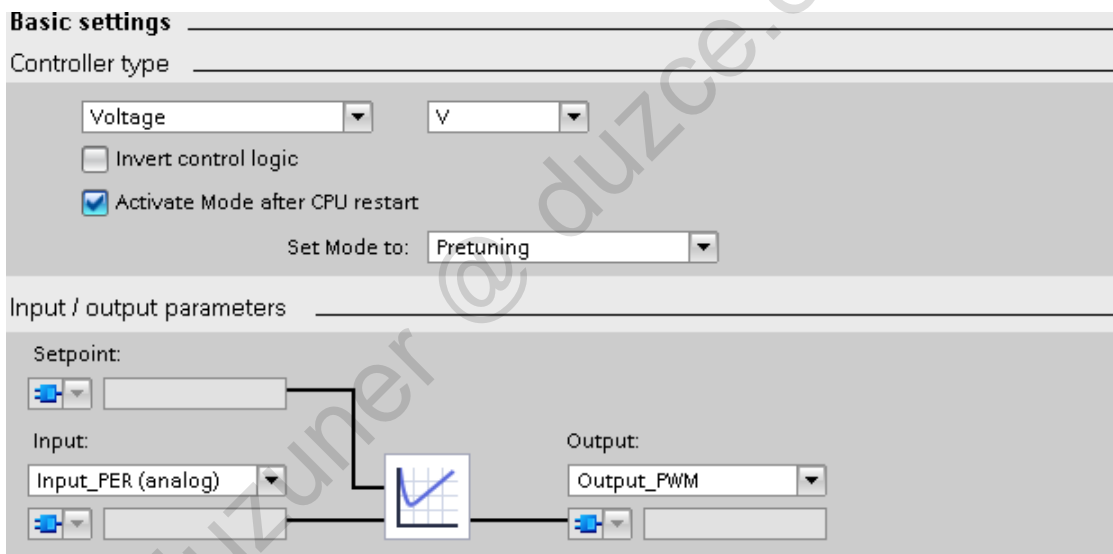


**Task**

In PLC\_2 (CPU 1211C), create a new technology object of the type "PID" and give it the name "PID\_RC".

**What to do**

1. Start the technology objects wizard and create a new PID controller.  
 PLC\_2 → Technology objects → Double-click on "add new object" → PID\_Compact → Name: PID\_RC
2. Implement the settings shown in the following:



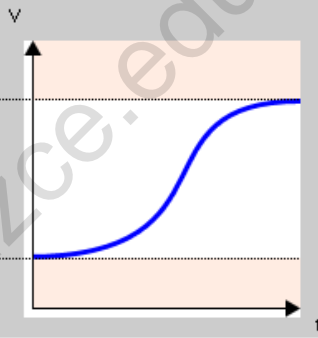
*Continued the next page →*

**Process value settings**

Process value limits

Process value high limit: 15.0 V

Process value low limit: 0.0 V



Process value scaling

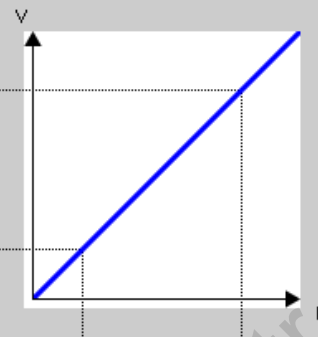
Input\_PER: Enabled

Scaled high process value: 10.0 V

Scaled low process value: 0.0 V

0.0 Low 27648.0 High

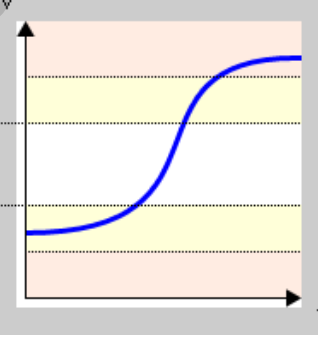
Automatic setting



Process value monitoring

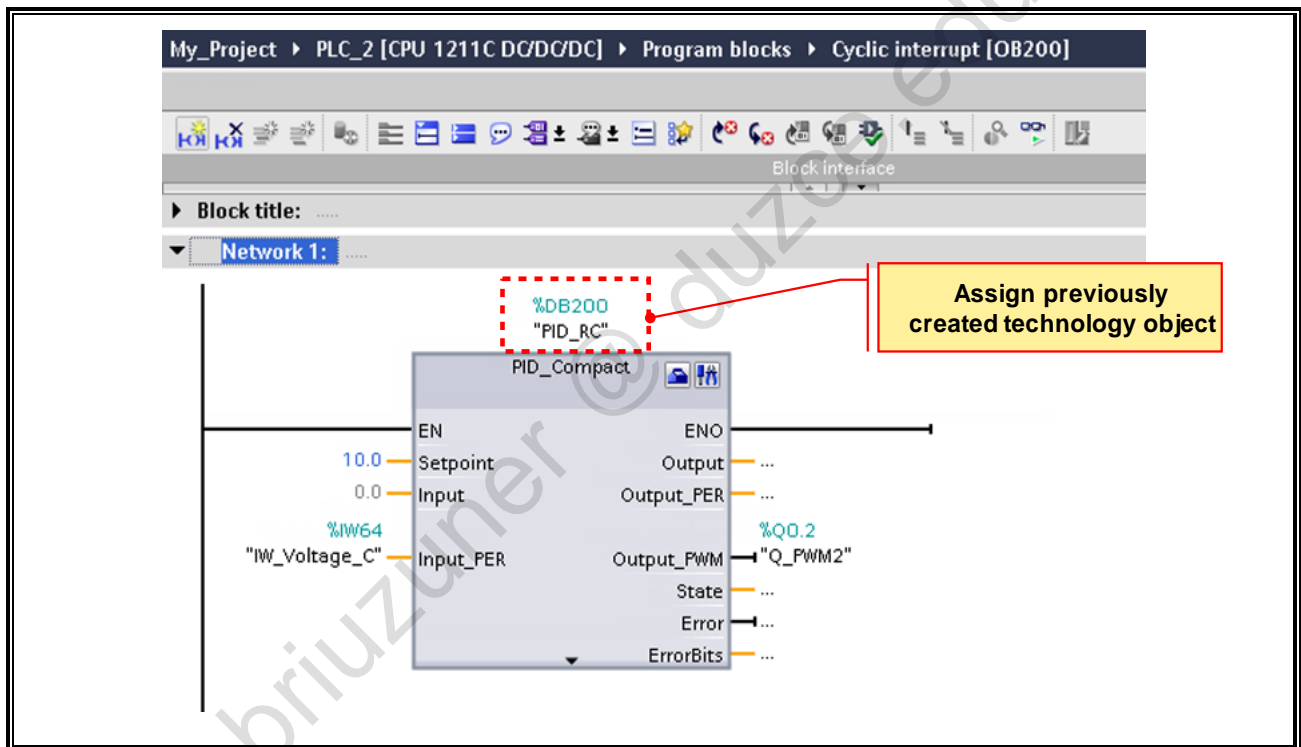
Warning high limit: 13.0 V

Warning low limit: 7.0 V



3. PWM limits, output value limits, reaction to error, and PID parameters remain unchanged.
4. Save your project.

## 8.5.2. Exercise 2: Calling the "PID\_Compact" block in the cyclic interrupt "Cyclic Interrupt"



### Task

Create "Cyclic Interrupt", call the block "PID\_Compact" and assign the previously created technology object "PID\_RC" to it.

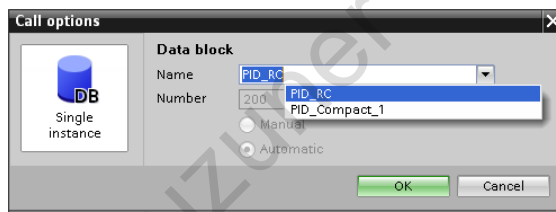
### What to do

1. Create the cyclic interrupt OB "Cyclic Interrupt". Cycle time 250ms.



In order to be able to react faster to disturbances, the sampling time of the closed-loop control (cycle time of the cyclic interrupt) can be reduced, for example, to 100ms or less. So that it is easier to monitor the control process, a relatively high value of 250ms is set for the exercise.

2. In this OB, call the block "PID\_Compact" from the instructions catalog.  
Instructions Task Card → Technology → PID Control → Compact PID → PID\_Compact
3. In the "Call options" dialog which opens, select "PID\_RC"



4. Assign the block as shown in the picture
  - Voltage\_RC (IW64) is the analog input 0 (0-10V) on the CPU.
  - PWM\_A02 (Q0.2) is the digital output for the PWM.
  - The setpoint is assigned constantly with 10.0 (V).
5. Save your project and transfer the complete user program into the CPU.

### 8.5.3. Exercise 3: Commissioning the PID controller



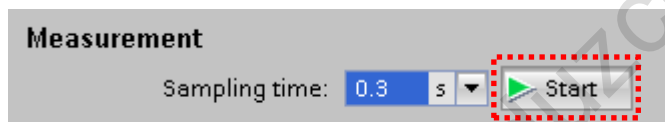
#### Task

Carry out a first commissioning of the PID controller and save the determined PID parameters in your project.

#### What to do

1. Open the commissioning panel  
PLC\_2 → Technology objects → PID\_RC → Commissioning

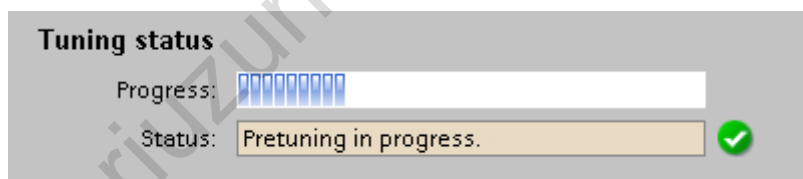
2. Start the measurement



3. Select the Tuning mode "Pretuning" and click on start

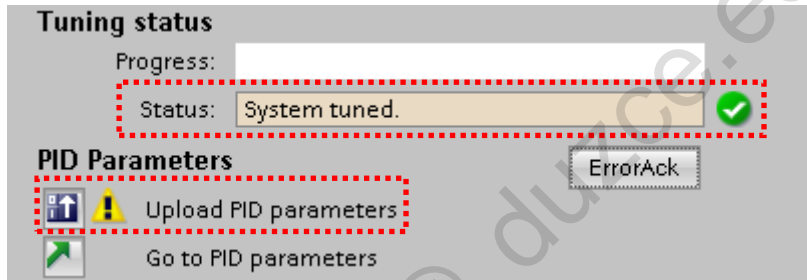


4. Monitor the progress of the tuning



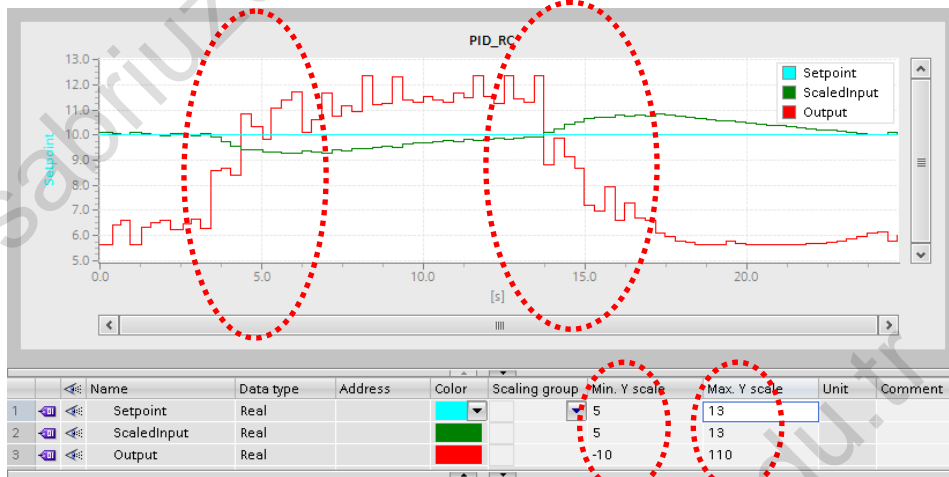


- After the system is tuned, click on "Upload PID parameters", in order to save the determined PID parameters in your project.



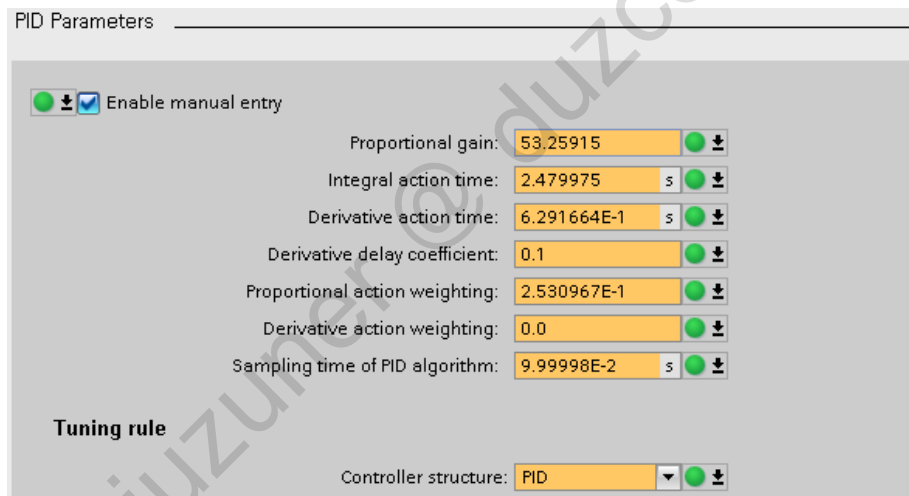
- Monitor the measured value trends of actual value (green) and manipulated value (red) and switch in the disturbance and after approximately 5 seconds, switch it off again.

Result: After switching on the disturbance, the manipulated value immediately shoots up to compensate for the dropping of the voltage at the capacitor. While the disturbance is pending, the manipulated value remains at a high level (approximately 85%). When the disturbance is switched off, an overshoot of the actual value develops, whereupon the manipulated value immediately drops.



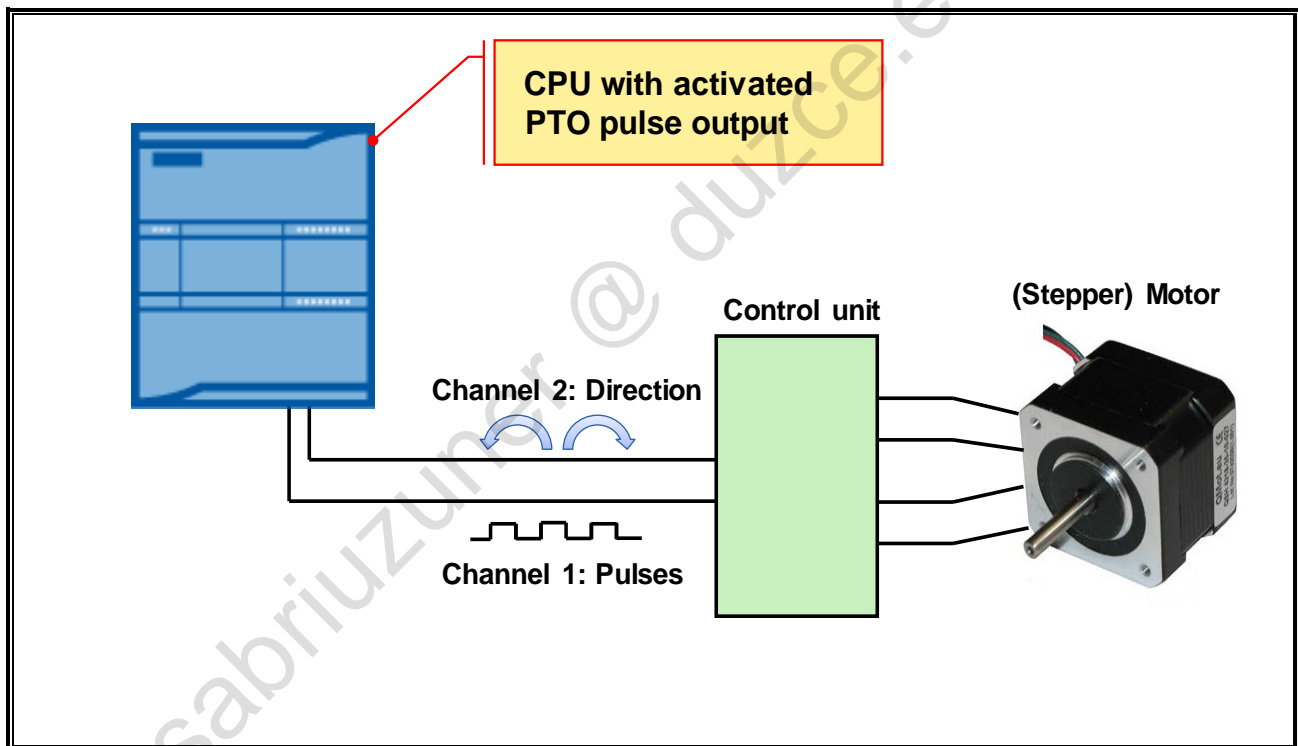
- Check the adoption of the PID parameters in the configuration of the technology object "PID\_RC"

PLC\_2 → Technology objects → PID\_RC → Configuration → PID Parameters



- Save your project.

## 8.6. Introduction to the "Axis" technology object (controlling the stepper motor)



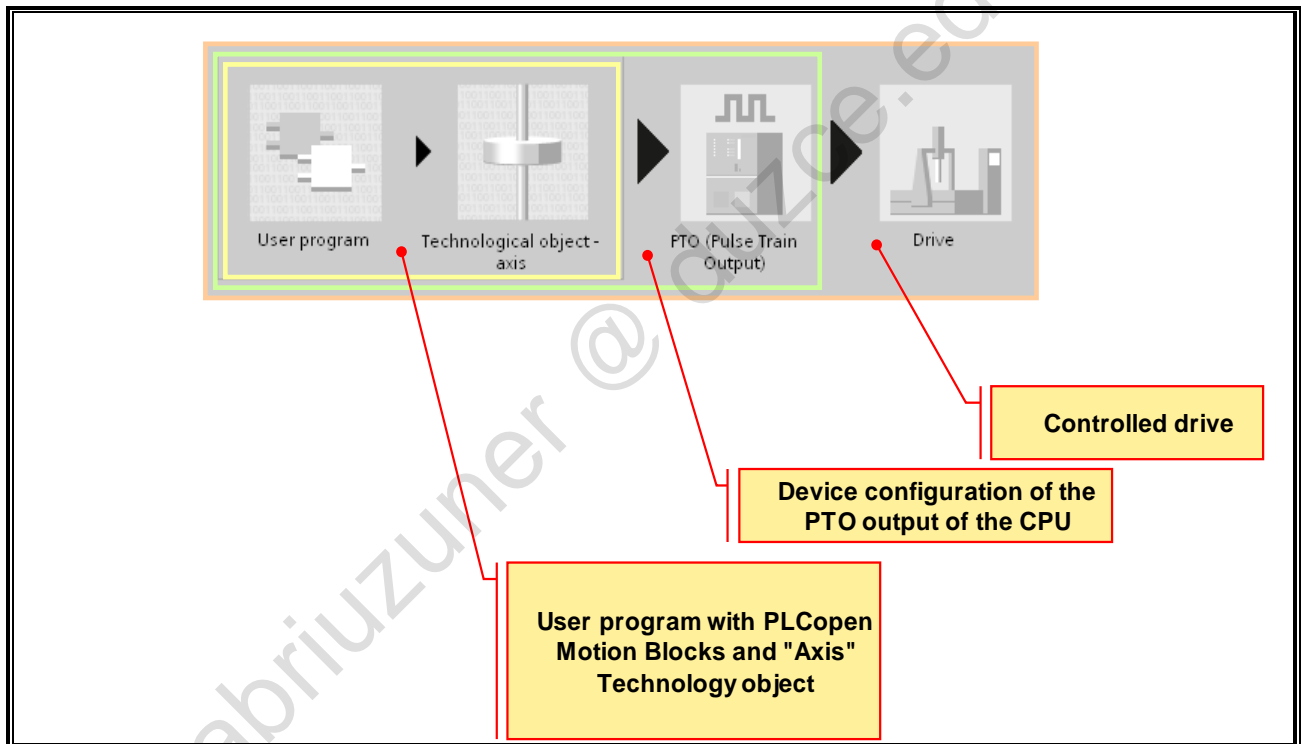
### "Axis" technology object

The "Axis" technology object represents an axis in the controller and is suitable for controlling stepper motors and servo motors with pulse interface. The "Axis" technology object is controlled via Motion Control instructions.

Suitable are all drives or control units which support a control via a pulse/direction interface.

Typical areas of use are adjustable axes and operating axes as well as feed axes and transport axes. These are used, for example, in the steel, automobile and food and beverage industry and are used, among others, in packaging machines.

### 8.6.1. Principle of axis control

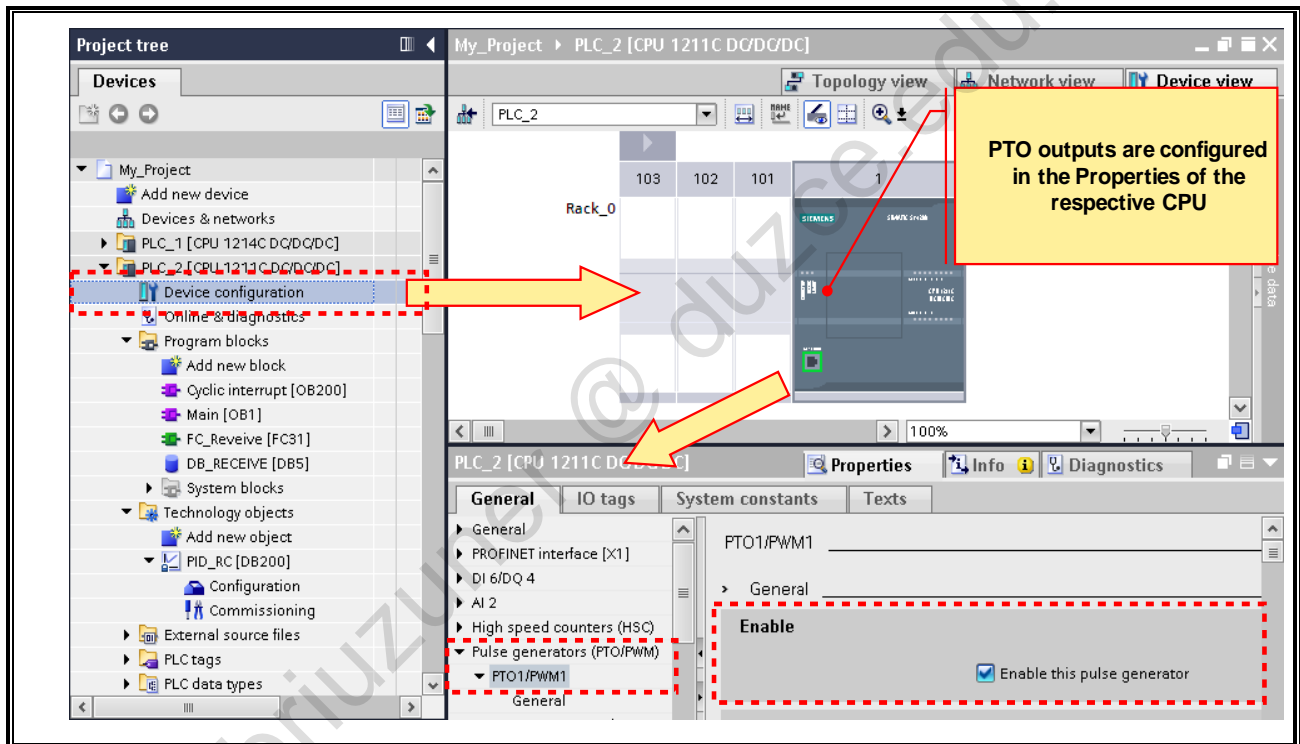


#### Principle of axis control

Within the S7-1200 there is a model grouped into four sections for the control of axes in which you are supported by wizards and diagnostic screens for the commissioning and diagnoses of axes.

- **User Program**  
The user program utilizes standardized Motion Control instructions for the control of the "Axis" technology object and thus the axis or the drive.
- **Technology object "Axis"**  
The "Axis" technology object represents an axis in the controller. Through dialog boxes (wizard), the parameters of the axis can very easily be specified. In the proper sense, the technology object is a data block with an exactly defined structure in which the parameters input by the user are entered. The specification of the DB is then required for the programming in the user program for the motion control instructions.
- **PTO (Output)**  
The PTO (output) is activated in the CPU device configuration and then assigned to the technology object "Axis". The output is subsequently controlled by the instructions in the user program.
- **Drive**  
Suitable are all drives or control units which support a control via a pulse/direction interface.

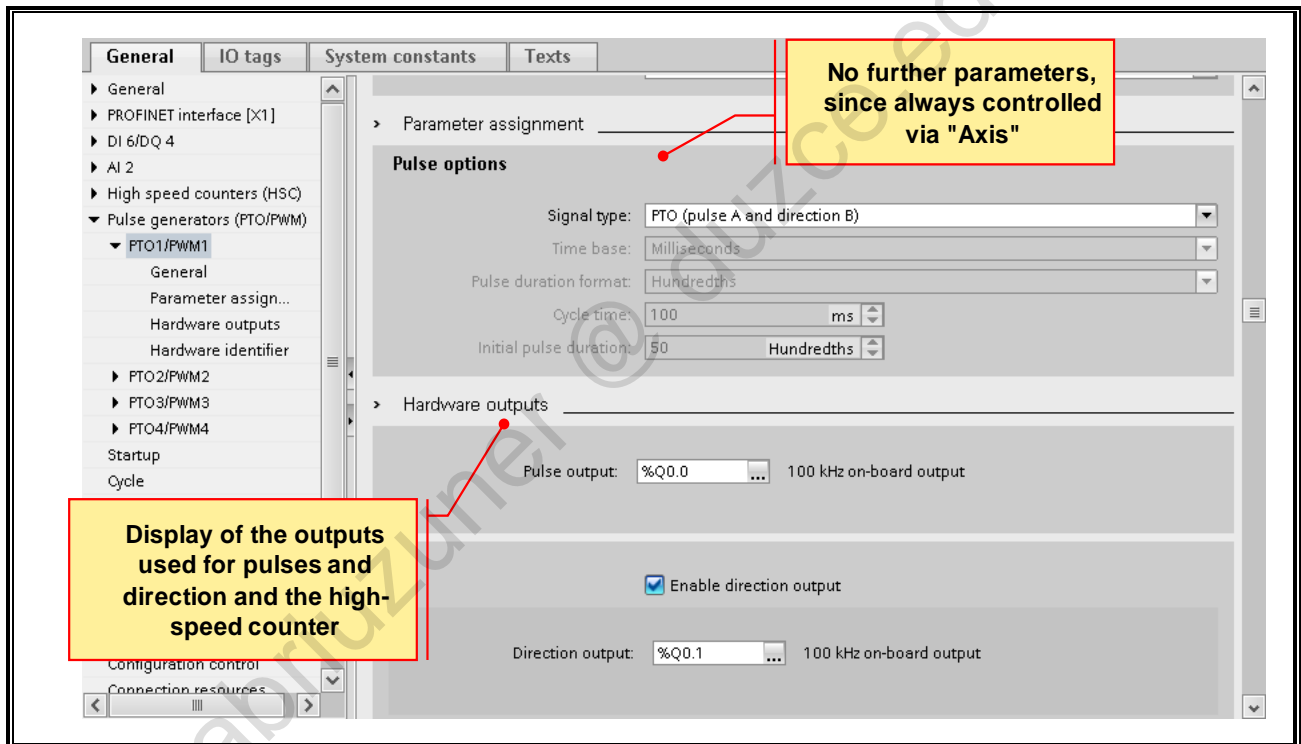
## 8.6.2. Configuring a PTO output (1)



### Configuring a PTO (output)

PTOs are activated via the device configuration of the respective CPU. In the properties window "Pulse generators", the respective pulse generator must first be activated.

### 8.6.3. Configuring a PTO output (2)



#### Configuring a PTO output

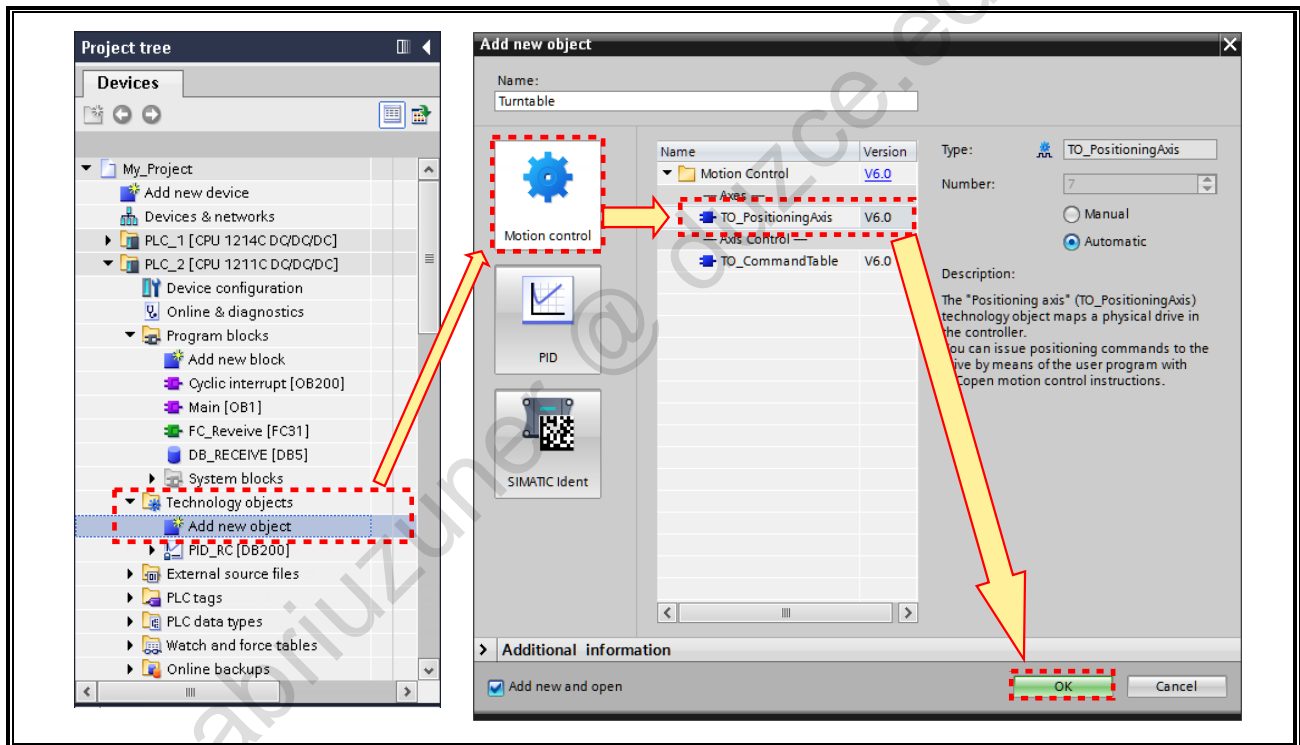
In the second step, the type of pulse generator must be selected. You can choose between "PTO" and "PWM".

To control an axis, you must choose "PTO". Since PTO outputs are always controlled via the "Axis" technology object, no further settings can be made in the device configuration.

#### There are 4 possibilities for controlling:

- PTO (pulse A and direction B)  
A pulse output and a direction output are used to control the stepper motor.
- PTO (count up A, count down B)  
One pulse output each for movement in positive direction and negative direction are used to control the stepper motor.
- PTO (A/B phase-shifted)  
Both pulse outputs for phase A and for phase B use the same frequency.  
On the drive side, the interval of the pulse outputs is evaluated as a step.  
The phase shifting between phase A and phase B determines the direction of movement.
- PTO (A/B phase-shifted four-fold)  
Both pulse outputs for phase A and for phase B use the same frequency.  
On the drive side, all rising and all falling edges of phase A and phase B are evaluated as a step.  
The phase shifting between phase A and phase B determines the direction of movement.

## 8.7. Creating a "Positioning Axis" technology object



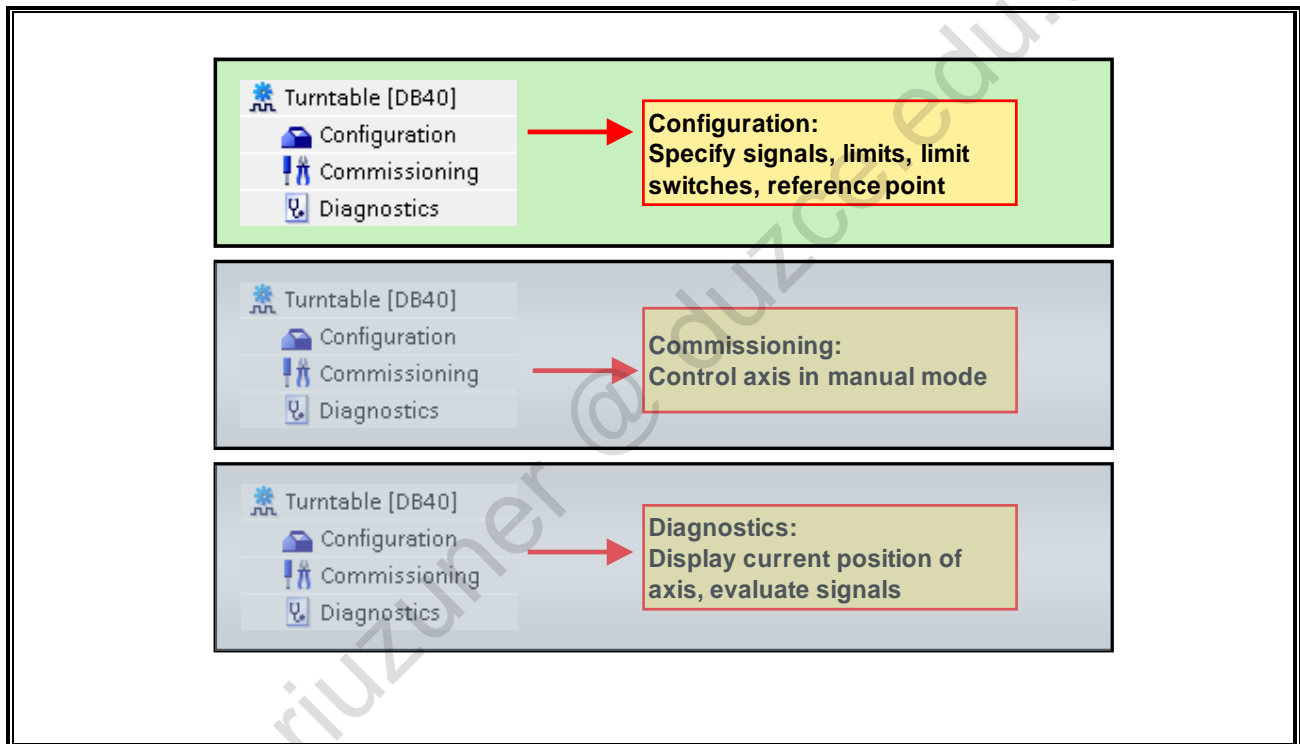
### Creating a "Positioning Axis" technology object

Just as with PID (controller), the "Add new object" dialog box (wizard) in the "Technology objects" folder is called for the creation.

First, "Motion" is selected and then the object "TO\_PositioningAxis".

With a click on "OK", the configuration wizard of the axis is started.

### 8.7.1. Properties of "Axis": Configuration



#### Properties of "Axis"

After the "Axis" technology object has been created, there are three selection possibilities available for handling:

- **Configuration**
  - Selection of the PTO (output) to be used and configuration of the drive interface
  - Properties of the mechanics and gear ratio of the drive (or the machine or plant)
  - Properties for position monitoring, for dynamic parameters and for referencing (homing)
  - The configuration is stored in the data block of the technology object
- **Commissioning**

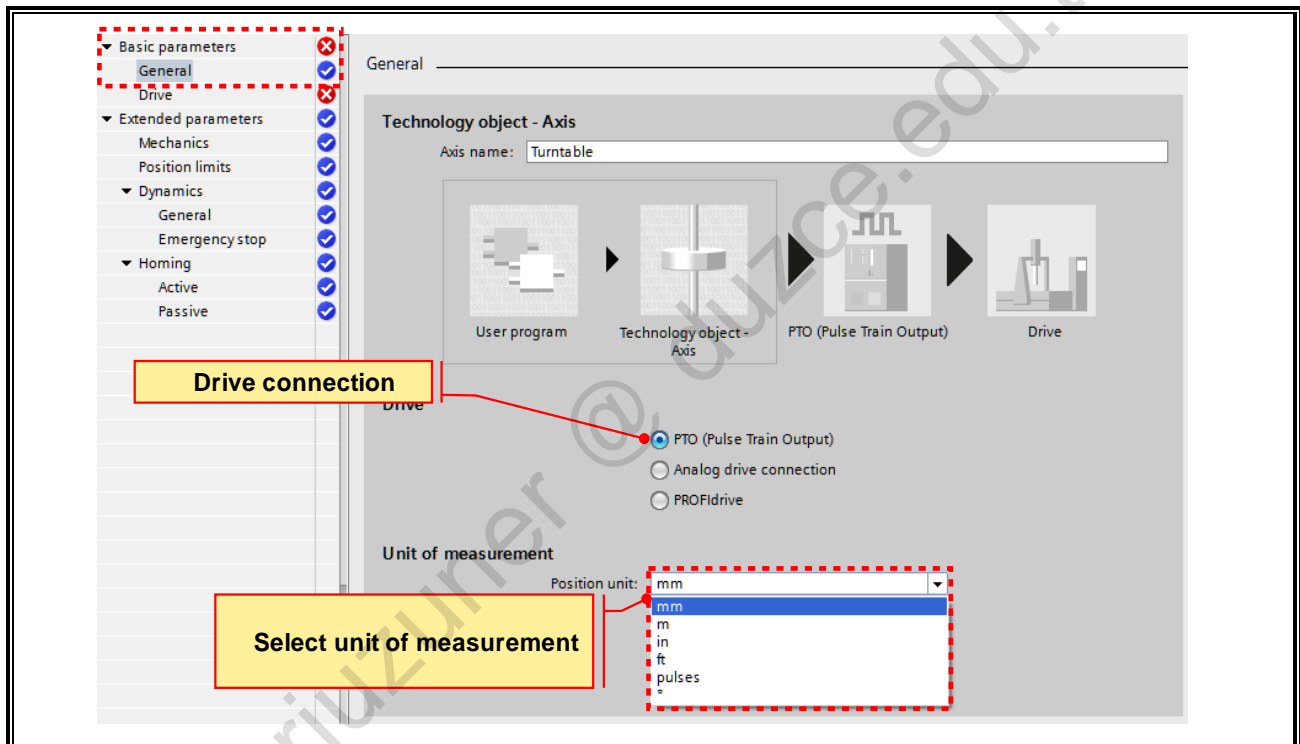
With the "Commissioning" tool, the functioning of the axis is tested without having to have created a user program. When you start this tool, the control panel opens. The following commands are available in the control panel:

  - Enable and disable the axis
  - Traversing the axis in jog mode
  - Absolute and relative positioning of the axis
  - Referencing (homing) the axis
  - Acknowledgement of errors
- **Diagnostic**

With the "Diagnostics" tool you check the current Status and Error information of axis and drive

In the following, the configuration of the axis is presented.

### 8.7.1.1. Configuring an "Axis" (1) - Basic parameters / general



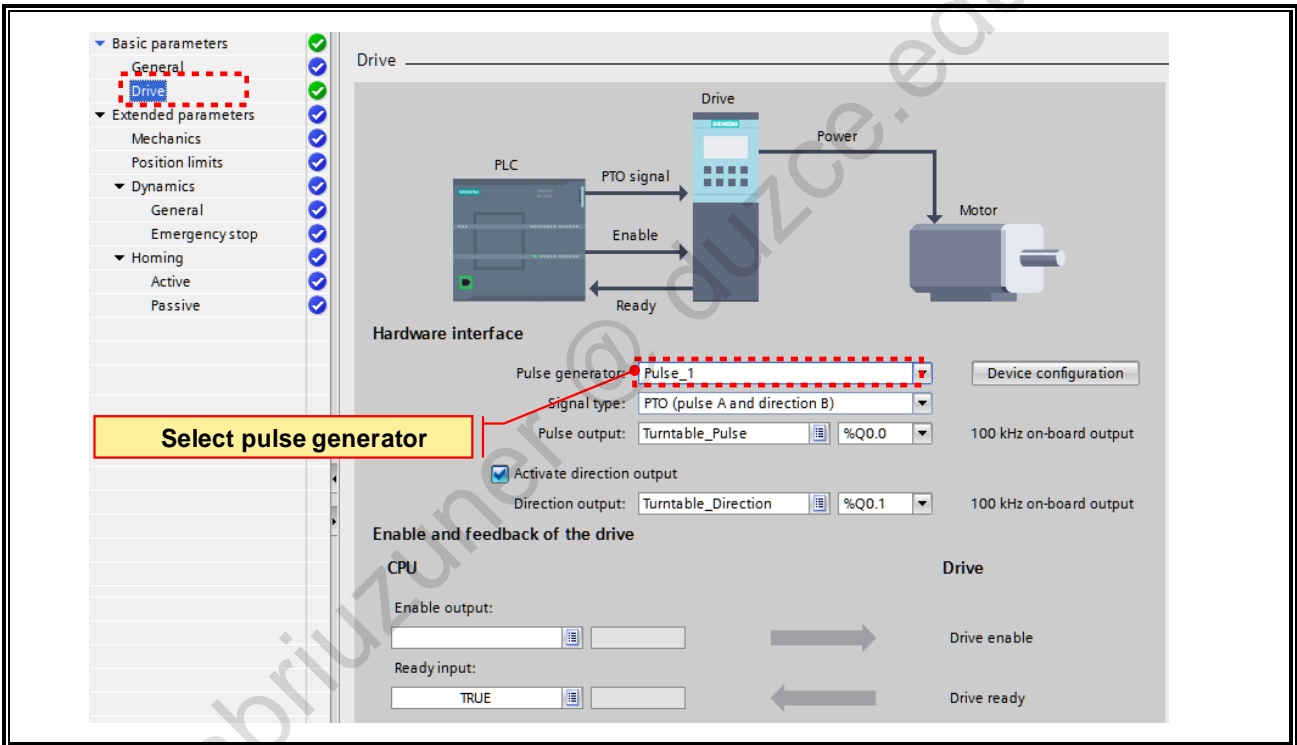
#### Basic parameters / general

The properties of "Axis" are divided into basic and extended parameters.

In the basic parameters, under general, the drive connection and the unit of measurement that is used are set.



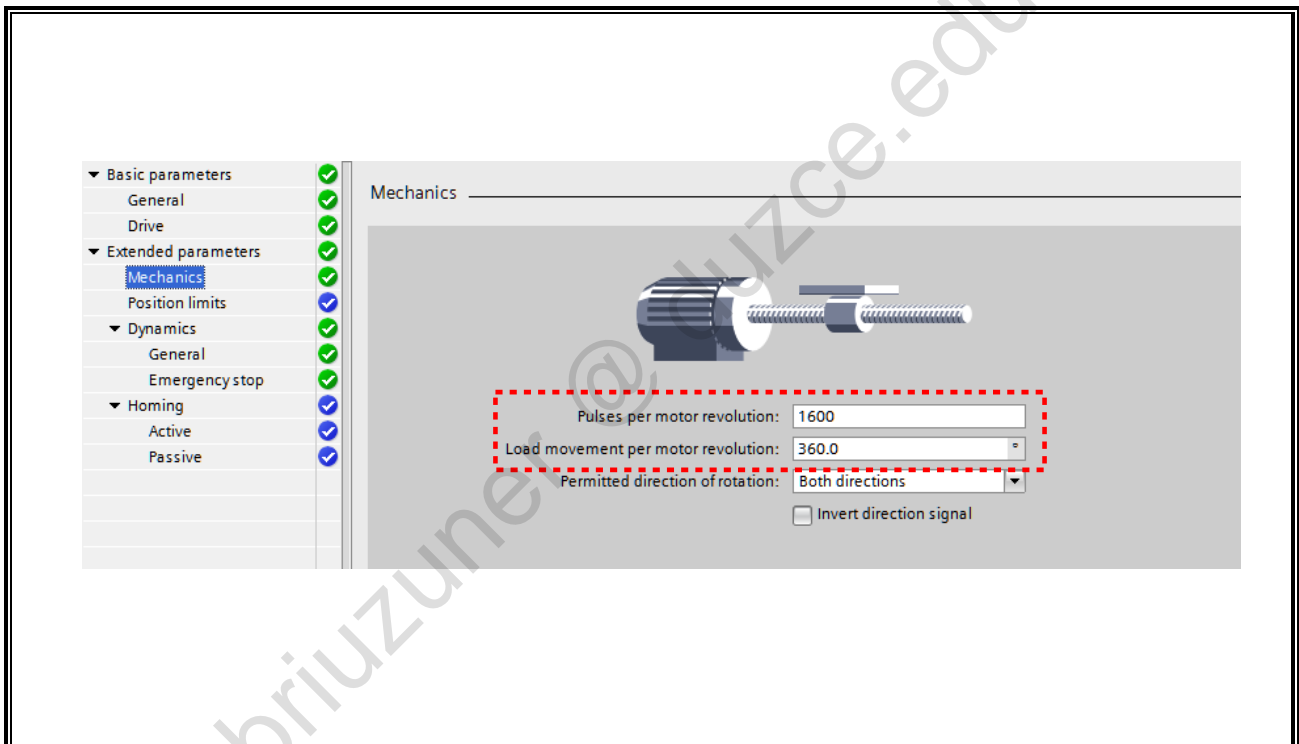
8.7.1.2. Configuring an "Axis" (2) - Drive



Basic Parameters – Drive

The properties of the "positioning axis" are subdivided into the basic and extended parameters. The hardware interface is set in the basic parameters.

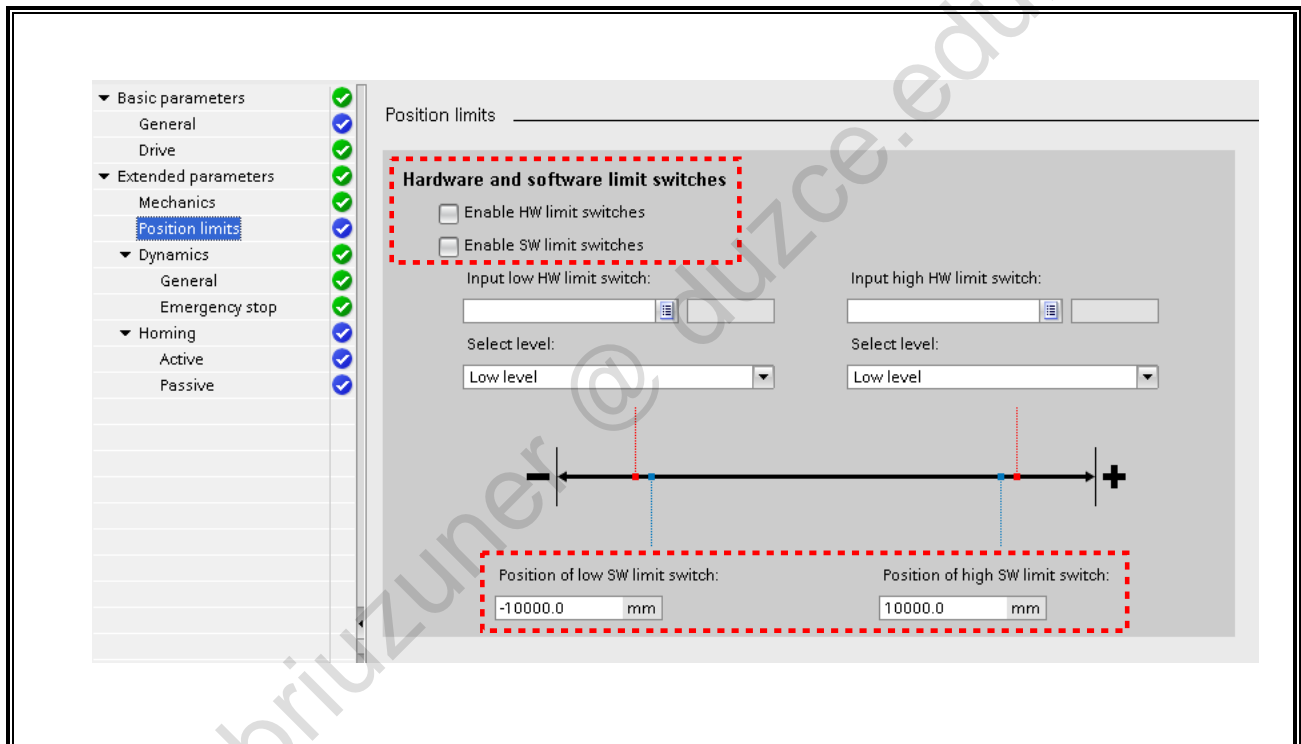
## 8.7.1.3. Configuring an "Axis" (3) – Mechanics

**Mechanics**

- Pulses per motor revolution  
In this field you specify how many pulses the motor requires for one revolution
- Distance (Load movement) per motor revolution  
In this field you specify what distance the mechanics of the system covers per motor revolution
- Invert direction signal  
Through the checkbox "Invert direction signal" you can adjust the direction output to the direction logic of the drive

Invert direction signal: deactivated	Invert direction signal: activated
0 V level = negative direction	0 V level = positive direction
5 V / 24 V level = positive direction	5 V / 24 V level = negative direction

## 8.7.1.4. Configuring an "Axis" (4) – Positioning limits

**Hardware limit switches**

Input low / high HW limit switch

Through the drop-down lists, you can set the digital hardware limit switches. The inputs must be interrupt-capable. As inputs for the HW limit switches, the digital on-board CPU inputs and the digital inputs of an inserted signal board are available.



A filter time of the digital inputs of the CPU is set to 6.4 ms by default. This can lead to undesired delays when used as HW limit switches. In this case, the delay time must be shortened accordingly.

The filter time can be set in the Devices configuration of the digital inputs under "Input filters".

**Software limit switches**

The software limit switches are activated via the checkbox "Enable SW limit switches". They are purely virtual and can be specified here as the distance travelled from the zero point.

The software limit switches are defined through the input fields "Position of low/high SW limit switch".



The value of the high software limit switch must be greater than or equal to the value of the low software limit switch.



Enabled software limit switches can only be operational with a referenced axis.

## 8.7.1.5. Configuring an "Axis" (5)- Dynamics/General

Note: Changes in the velocity limits affect acceleration and deceleration; the ramp-up time and ramp-down time stay the same.

Unit of velocity limits:  $^{\circ}/s$

Maximum velocity: 200.0  $^{\circ}/s$

Start/stop velocity: 20.0  $^{\circ}/s$

Acceleration: 360.0  $^{\circ}/s^2$

Deceleration: 360.0  $^{\circ}/s^2$

Ramp-up time: 0.5 s

Ramp-down time: 0.5 s

**Dynamics / General**

In the "Dynamics - General" configuration dialog you can specify the limits for the motion sequences:

- Velocity
  - In the field "Maximum velocity", the maximum allowable velocity of the axis is configured.
  - In the field "Start/stop velocity", the minimum allowable velocity of the axis is configured.
  - The value of the Maximum velocity must be greater than or equal to the value of the Start/stop velocity.
- Acceleration
 

The desired acceleration values can be specified either directly via the fields "Acceleration" and "Deceleration" or indirectly via the specification of acceleration or deceleration time.
- Jerk limit
 

The "Enable jerk limit" option can be used to set a non-abrupt acceleration or deceleration of the axis

## 8.7.1.6. Configuring an "Axis" (6) – Dynamics/Emergency stop

The screenshot shows the configuration interface for an axis emergency stop. The left sidebar lists various parameters, with 'Emergency stop' selected under 'Dynamics'. The main area displays two graphs: 'Velocity' and 'Deceleration' over time 't'. The Velocity graph shows a constant velocity of 200.0 °/s, followed by a linear ramp down to 20.0 °/s, and then a constant velocity of 20.0 °/s. The Deceleration graph shows a constant deceleration of 720.0 °/s² during the ramp-down phase. A parameter 'Emergency stop ramp-down time' is set to 0.25 s.

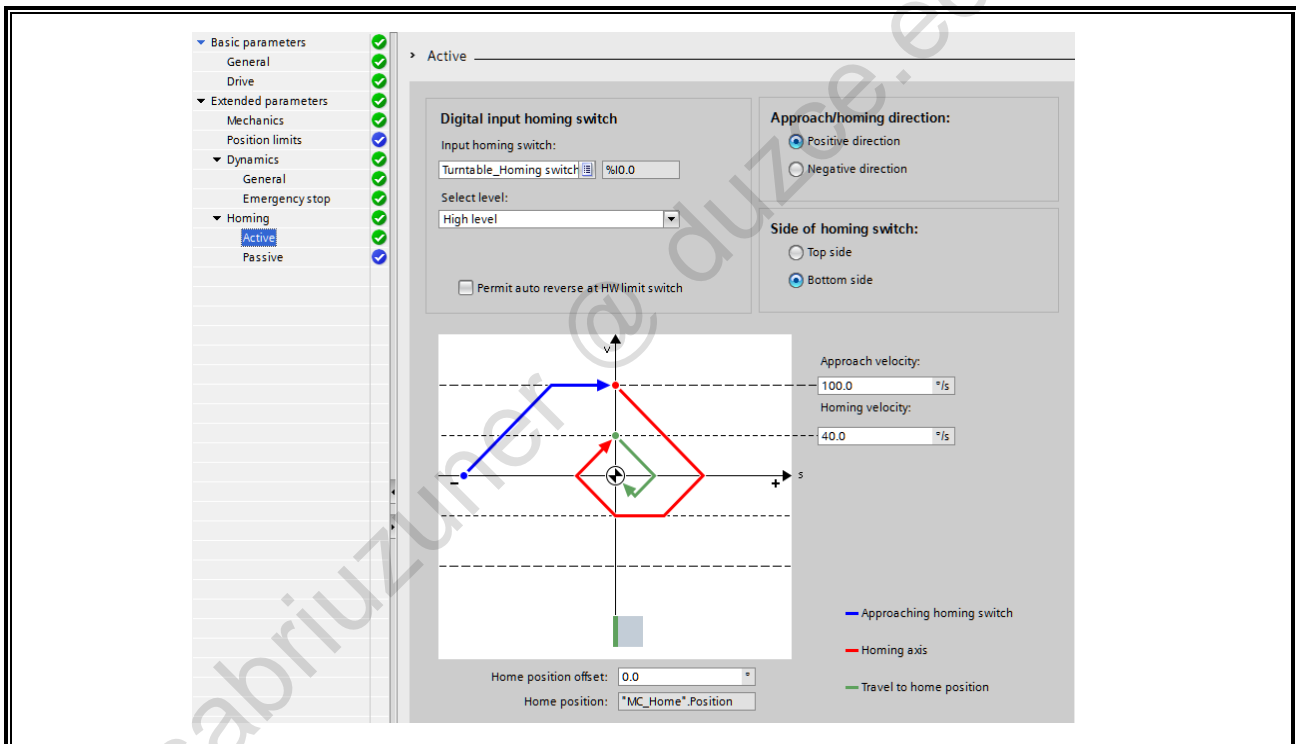
**Dynamics / Emergency stop**

In the "Dynamics - Emergency stop" configuration wizard dialog, the Emergency deceleration of the axis can be set. In case of failure and when blocking the axis with the Motion-Control instruction "MC\_Power" (Input parameter StopMode = 0), the axis is brought to a standstill with this deceleration.



The Emergency deceleration must be sufficiently large in order to bring the axis to a standstill in good time when there is an emergency (for example, when approaching the hardware limit switches, before reaching the mechanical stop).

## 8.7.1.7. Configuring an "Axis" (7) - Homing



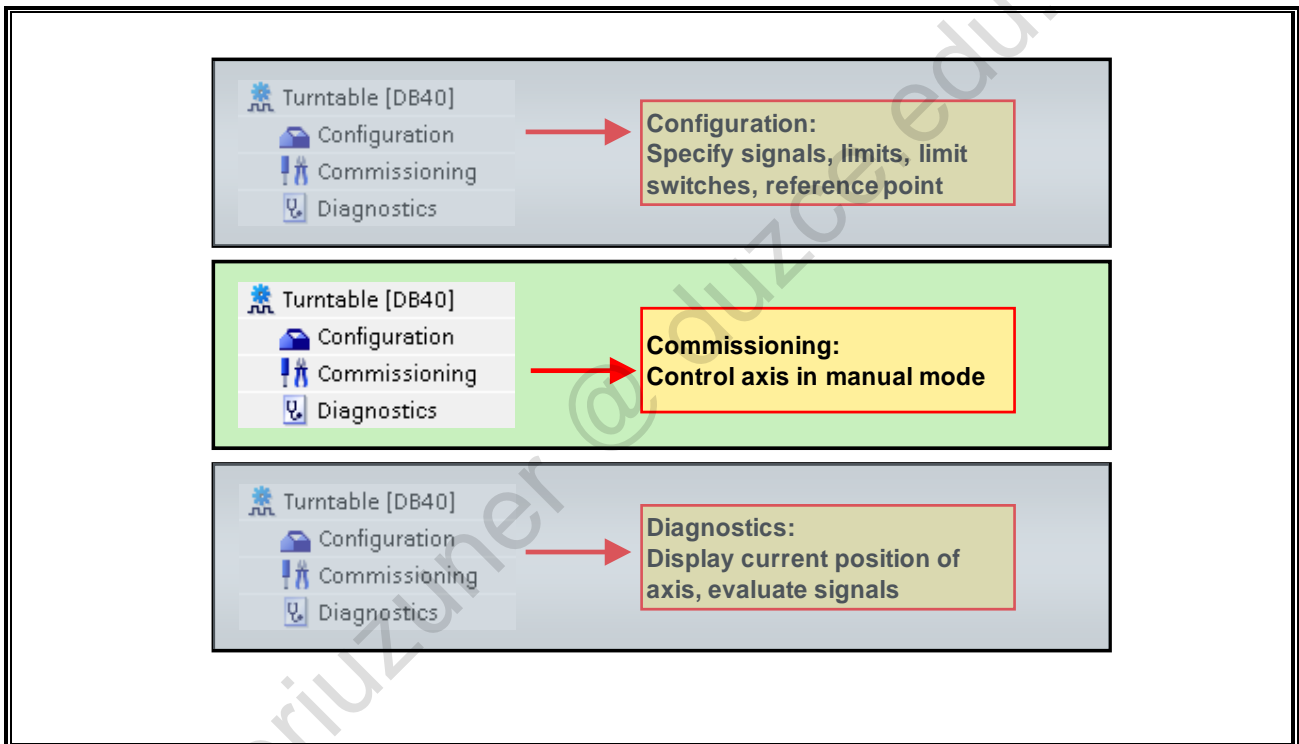
## Homing (referencing)

Compared to a closed-loop control system, you don't have any feedback signal during traversing from which you can derive conclusions on the current position of the axis. For that reason, it is necessary that the axis, for example, every time the system is switched on, references a defined point, the so-called reference point. This reference point (also reference cam) is configured as an interrupt-capable input and marks the zero point of the axis. When the position of the reference point switch and the reference point position (home position) is different, the appropriate reference point offset is entered in the field "Home position offset". The axis approaches the reference (home) position with the referencing velocity.

In order to approach the home position exactly there is the "active homing", whose sequence is represented in the picture above. The movement is divided into three steps:

- Seeking the referencing point (homing) switch (blue section of the graph)  
When active referencing is started, the axis accelerates to the configured "Approach velocity" and with it seeks the referencing point (homing) switch.
- Reference point travel (red section of the graph)  
When it detects the referencing point (homing) switch, the axis slows down in this example, turns around, in order to reference on the configured side of the referencing point (homing) switch with the configured "Homing velocity".
- Traversing the home position offset (green section of the graph)  
After referencing, the axis travels the distance of the home position offset with the referencing velocity. Arriving there, the axis finds itself in the home position which was specified at the input parameter "Position" of the Motion Control instruction "MC\_Home".

## 8.7.2. Properties of "Axis": Commissioning



### Properties of "Axis"

After the "Axis" technology object has been created, there are three selection possibilities available for handling:

- **Configuration:**
  - Selection of the PTO (output) to be used and configuration of the drive interface
  - Properties of the mechanics and gear ratio of the drive (or the machine or plant)
  - Properties for position monitoring, for dynamic parameters and for referencing (homing)
  - The configuration is stored in the data block of the technology object
- **Commissioning:**

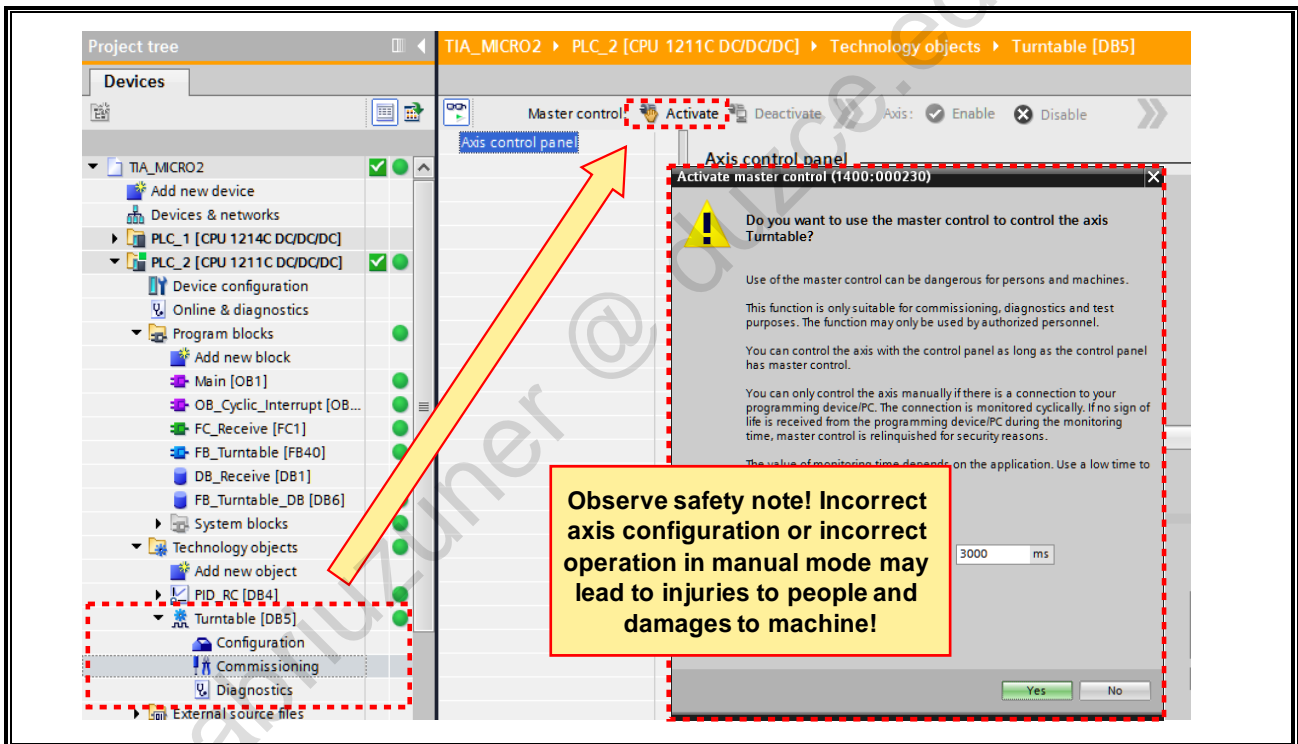
With the "Commissioning" tool, the functioning of the axis is tested without having to have created a user program. When you start this tool, the Control panel opens. The following commands are available in the Control panel:

  - Enable and Disable the axis
  - Traversing the axis in Jog mode
  - Absolute and relative positioning of the axis
  - Referencing (homing) the axis
  - Acknowledgement of errors
- **Diagnostics:**

With the "Diagnostics" tool you check the current Status and Error information of axis and drive.

In the following, the commissioning of the axis is presented.

### 8.7.3. Activating the commissioning panel



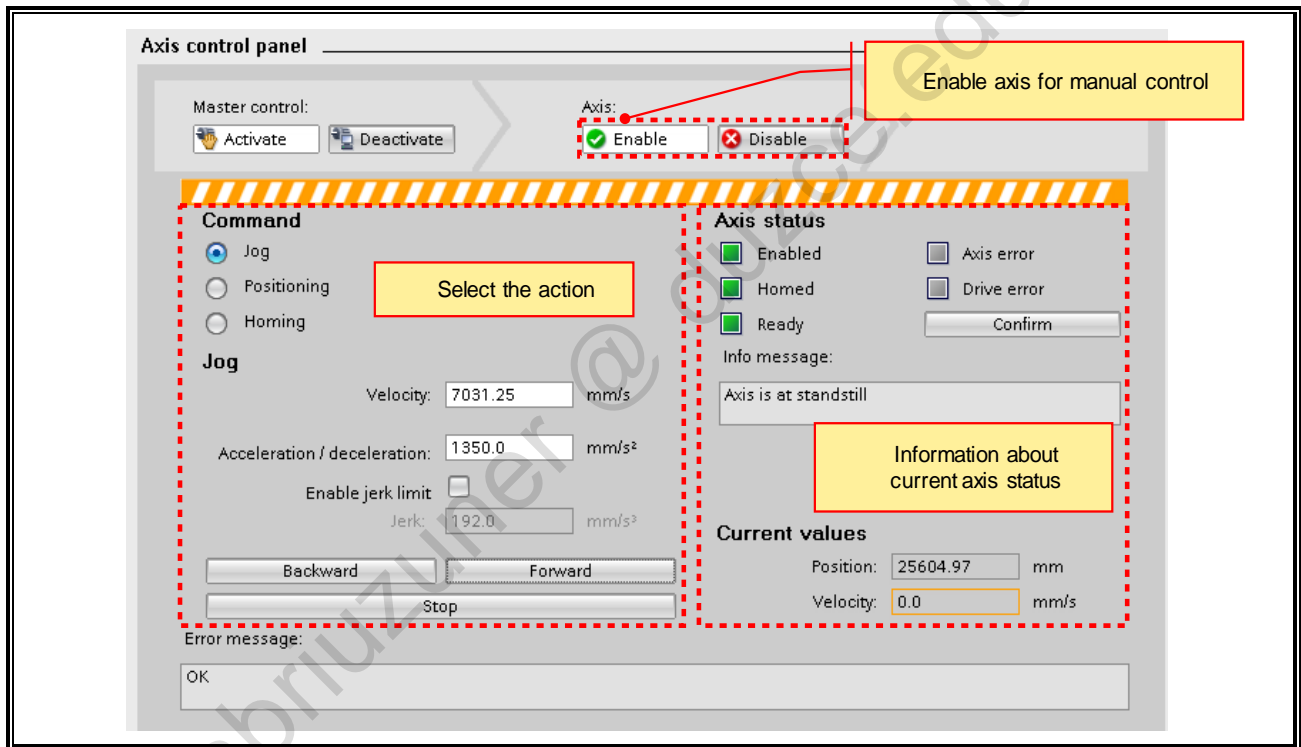
#### Commissioning panel

In automatic mode, the Control Panel offers the opportunity to get an overview of the current status of the axis. The most important bits such as Enabled, Homed or Axis error as well as the current values on Position and Velocity are represented.

If the axis is to be controlled in manual mode, a warning appears which points out that all necessary safety precautions are to be taken since you are about to actively intervene in the process.



## 8.7.3.1. Using the commissioning panel (manual control)

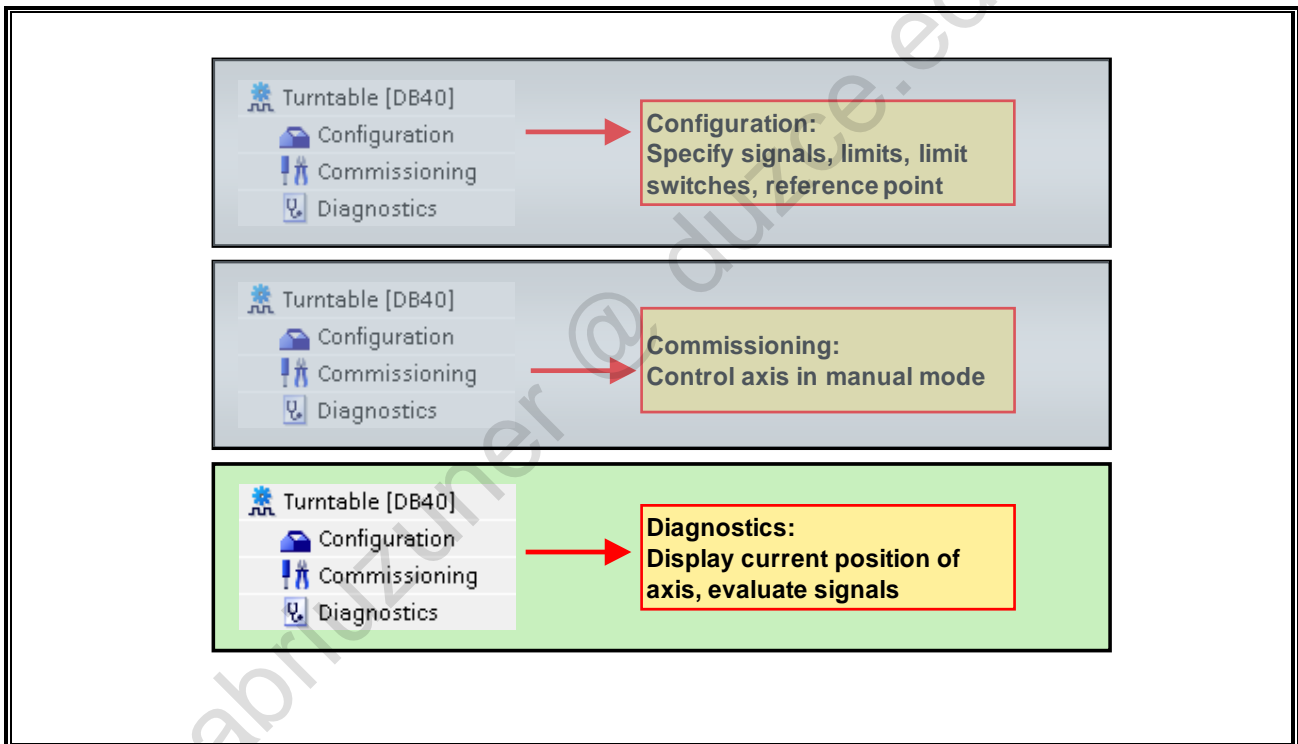


## Using the commissioning panel

If the axis is to change to manual control, the execution of the MC-Power command in the user program must be deactivated first. Then, the manual control can be activated via the button "Enable".

Thereupon, the control passes from the user program to the Control Panel and it is possible to enable and disable the axis, move with the selected velocity in jog mode or to acknowledge errors of the control panel as soon as their cause is eliminated.

### 8.7.4. Properties of "Axis": Diagnostics



#### Properties of "Axis"

After the "Axis" technology object has been created, there are three selection possibilities available for handling:

- **Configuration:**
  - Selection of the PTO (output) to be used and configuration of the drive interface
  - Properties of the mechanics and gear ratio of the drive (or the machine or plant)
  - Properties for position monitoring, for dynamic parameters and for referencing (homing)
  - The configuration is stored in the data block of the technology object
- **Commissioning:**

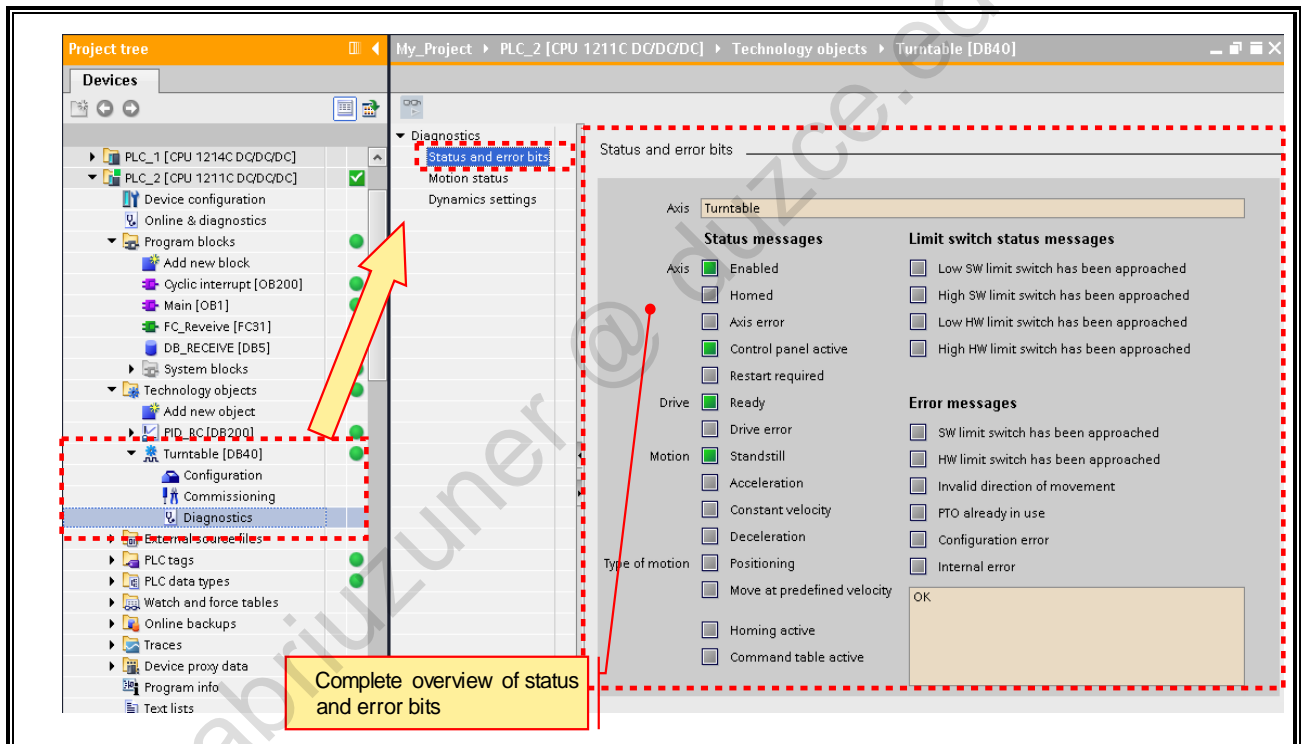
With the "Commissioning" tool, the functioning of the axis is tested without having to have created a user program. When you start this tool, the Control panel opens. The following commands are available in the Control panel:

  - Enable and Disable the axis
  - Traversing the axis in jog mode
  - Absolute and relative positioning of the axis
  - Referencing (homing) the axis
  - Acknowledgement of errors
- **Diagnostics:**

With the "Diagnostics" tool you check the current Status and Error information of axis and drive.

In the following, the diagnostics of the axis is presented.

## 8.7.4.1. Axis diagnostics (1)



## Status and error bits

After the axis diagnostics is started, the current statuses of the axis are displayed under "Status and error bits". Among other things, you can read out the current axis and motion status. In addition, error events, such as, the reaching of limit switches are displayed.

## 8.7.4.2. Axis diagnostics (2)

**Information on the current motion status**

**Display of the dynamics parameters**

Motion status	
Current position:	125178.0 mm
Current velocity:	2931.25 mm/s
Target position:	126542.1 mm
Remaining travel distance:	1364.063 mm

Dynamics settings	
Acceleration:	13500.0 mm/s <sup>2</sup>
Deceleration:	13500.0 mm/s <sup>2</sup>
Emergency deceleration:	7500.0 mm/s <sup>2</sup>
Jerk:	0.0 mm/s <sup>3</sup>

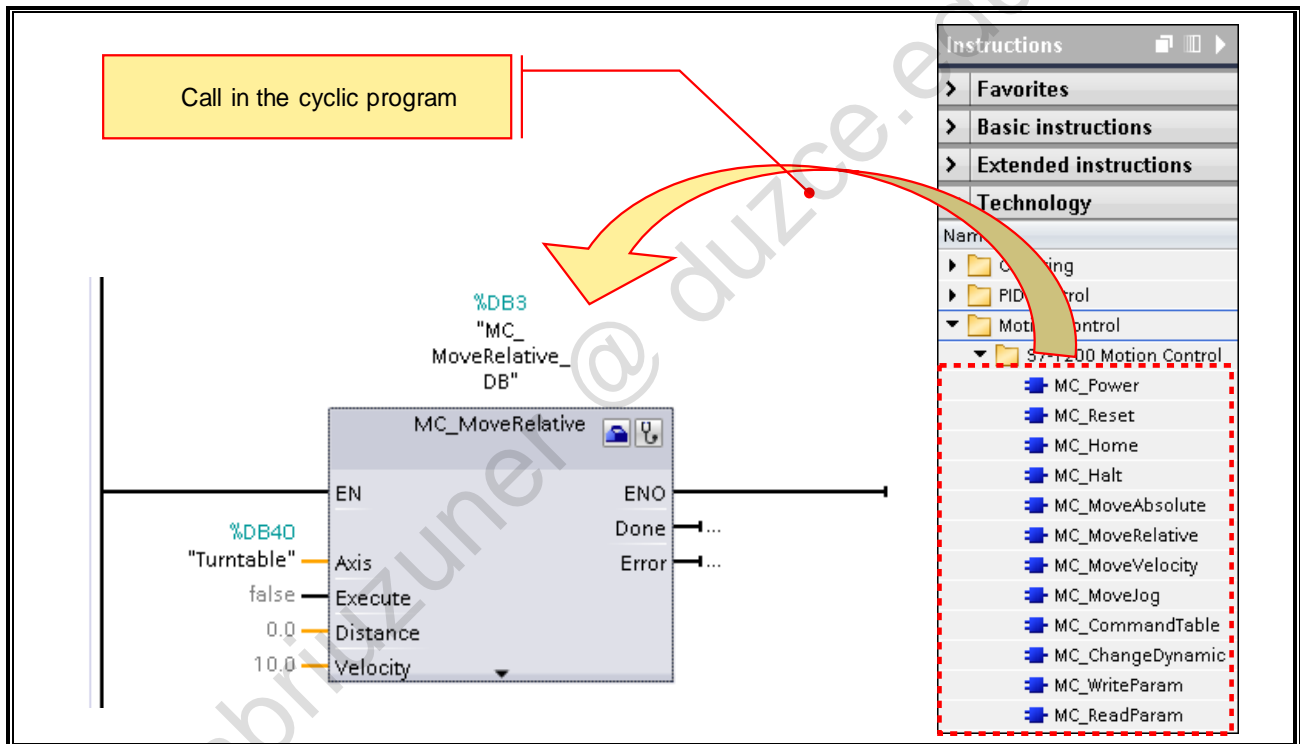
**Motion status and dynamics settings**

- Through the menu point "motion status" you can get information on the current movement. The values displayed are continuously updated.
- In the dynamics settings, you can read out the configured values for acceleration, deceleration and emergency deceleration.



All values are available as read-only.

### 8.7.5. Blocks for axis control



#### Motion control instructions

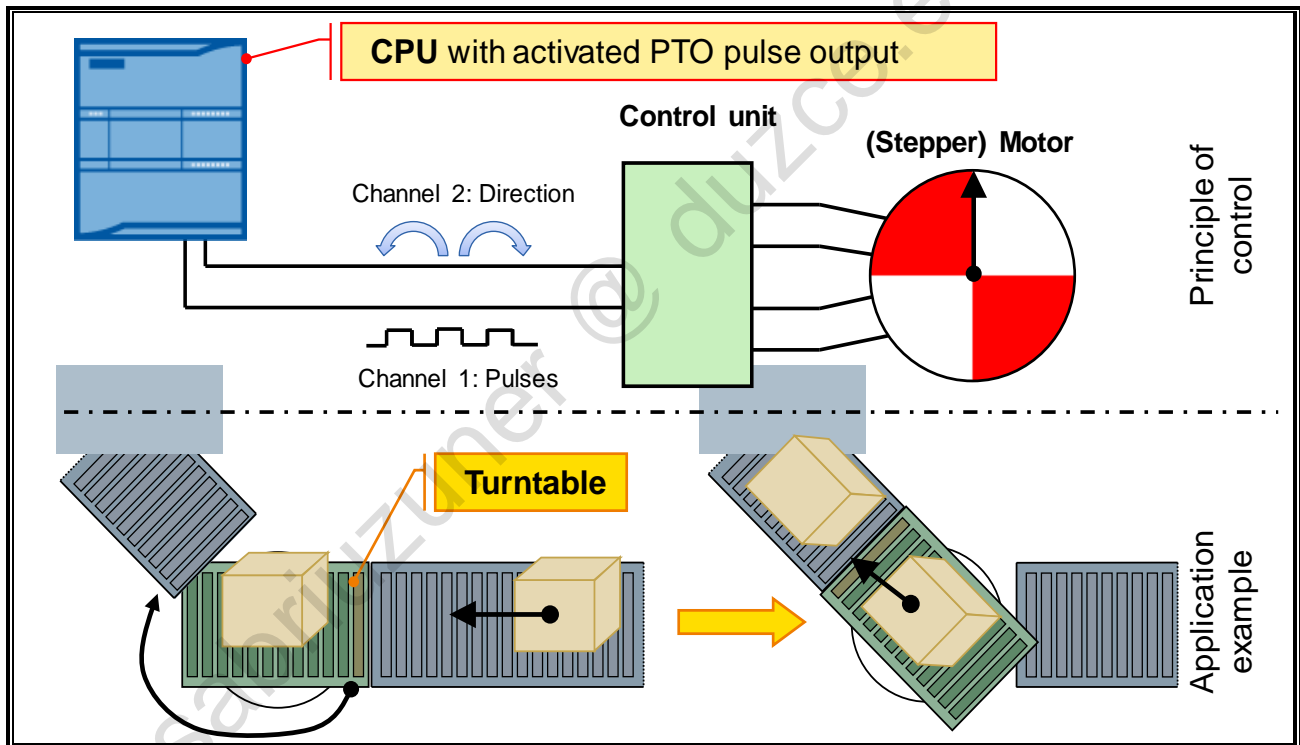
Through the motion control instructions, you control the axis from the user program. The instructions start motion control tasks which execute the desired functions.

The status of the motion control tasks as well as possible errors which occurred during processing can be queried at the output parameters of the motion control instructions. The following motion control instructions are available for selection:

Instruction	Function
MC_Power	Activate/deactivate axis
MC_Reset	Acknowledge error of the axis
MC_Home	Home (reference axis)
MC_Halt	Cancel all MC instructions (axis commands)
MC_MoveAbsolute	Move axis to an absolute position
MC_MoveRelative	Move axis to a position relative to the current one
MC_MoveVelocity	Move axis with a constant (defined) velocity
MC_MoveJog	Move axis with (manual) jog velocity
MC_CommandTable	Execute axis jobs as movement sequence
MC_ChangeDynamic	Change dynamic settings of the axis
MC_WriteParam	Write variables of the positioning axis
MC_ReadParam	Continuously read movement data of a positioning axis

All blocks can be called in the cyclic program.

## 8.8. Task description: Controlling a stepper motor



### Task description


The stepper motor of the training device is to be commissioned. For this, the "Turntable" technology object of the type "Axis" is to be created which is to be configured by you. On the hardware-side, the PTO (output) 1 of the CPU is used as well as a Boolean output for the specification of the direction.

The function block "FB\_Turntable" (FB40) takes over the control of the axis. You are to call this function block in the user program.

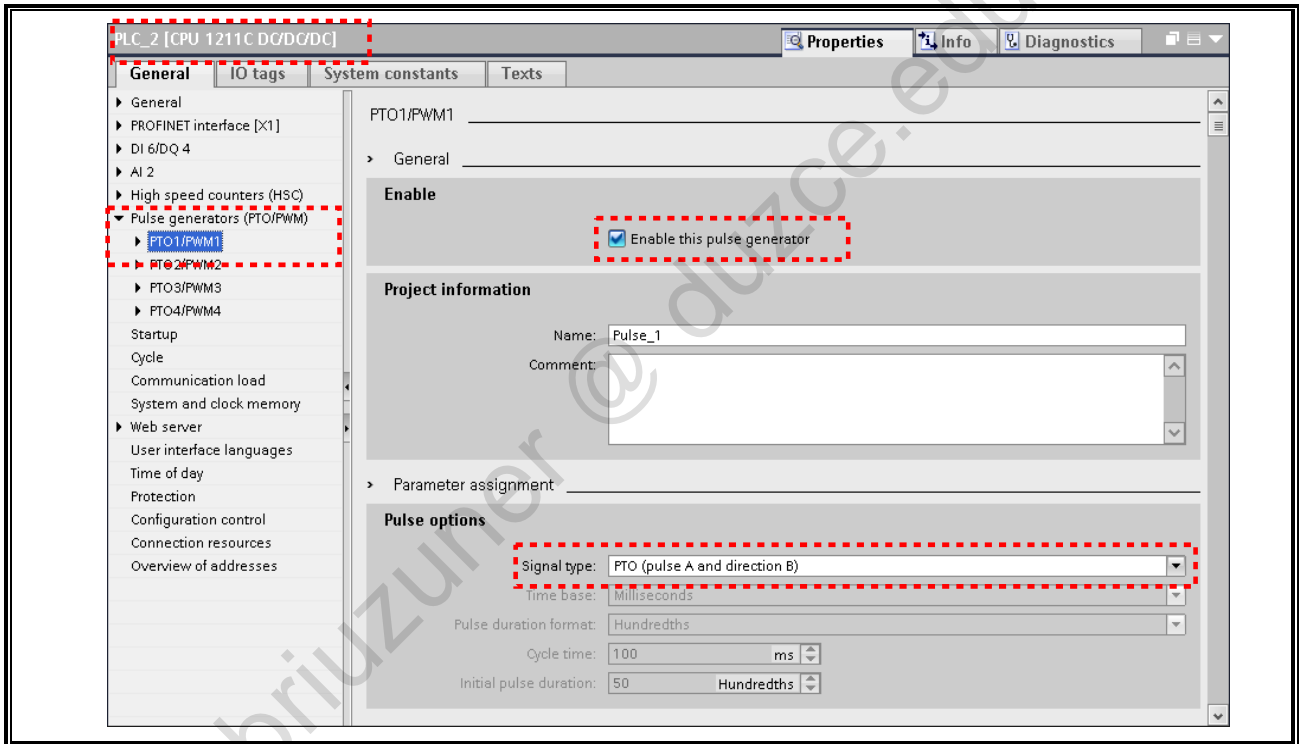
### Scenario

A production piece is transported via a turn-lift table. The production pieces arrive at the lower level and are transported onto the turn-lift table (Position 1: 90°). Then, the turn-lift table approaches the upper level and executes a 225° turn to Position 2 (315°). Having arrived at the upper level, the production piece is moved off the turn-lift table and is transported away. Subsequently, the turn-lift table travels back in the opposite direction to the starting point (Position 1).

The starting point is approached for the first time after the system is switched on and subsequent referencing (homing) is done.

 The vertical movement was not programmed.

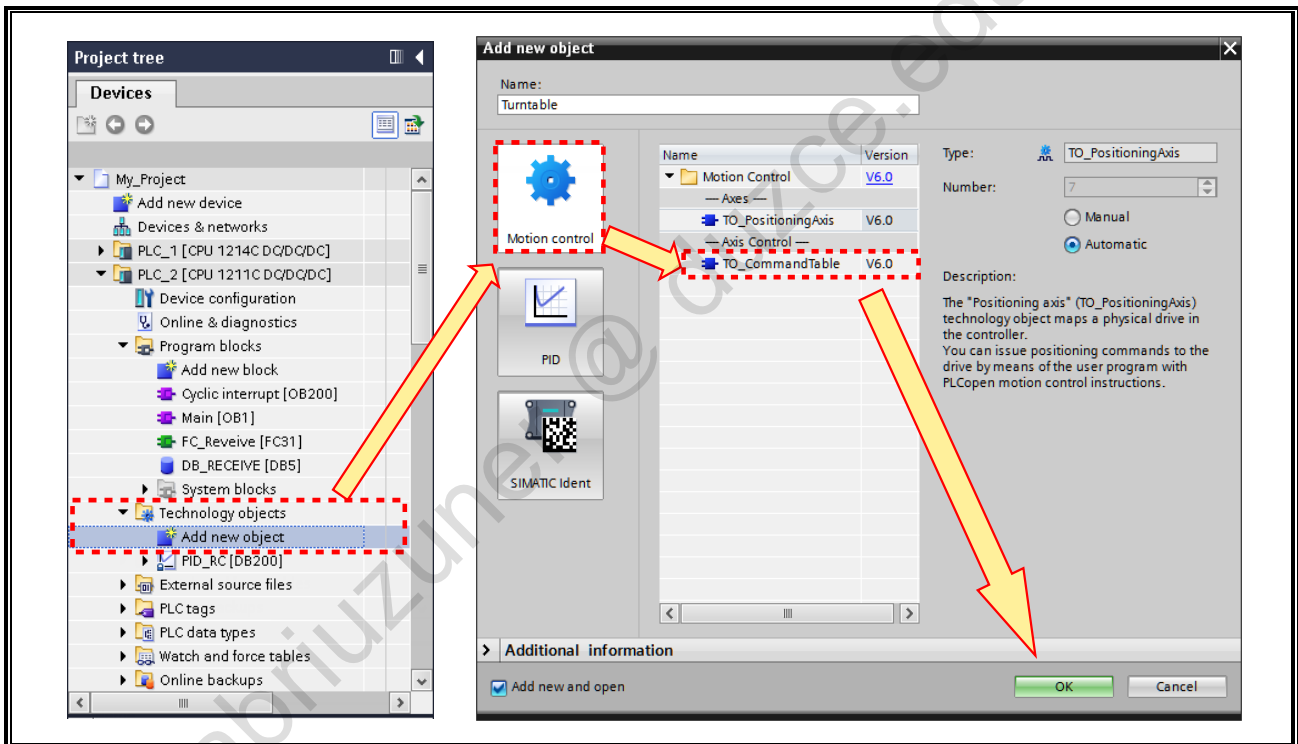
### 8.8.1. Exercise 4: Activating (enabling) PTO 1 of the CPU



#### Task

The "Axis" technology object configured in the following accesses a PTO output. For this, activate (enable) the PTO 1 of PLC\_2 as shown in the picture.

## 8.8.2. Exercise 5: Creating and configuring the technology object "Axis"

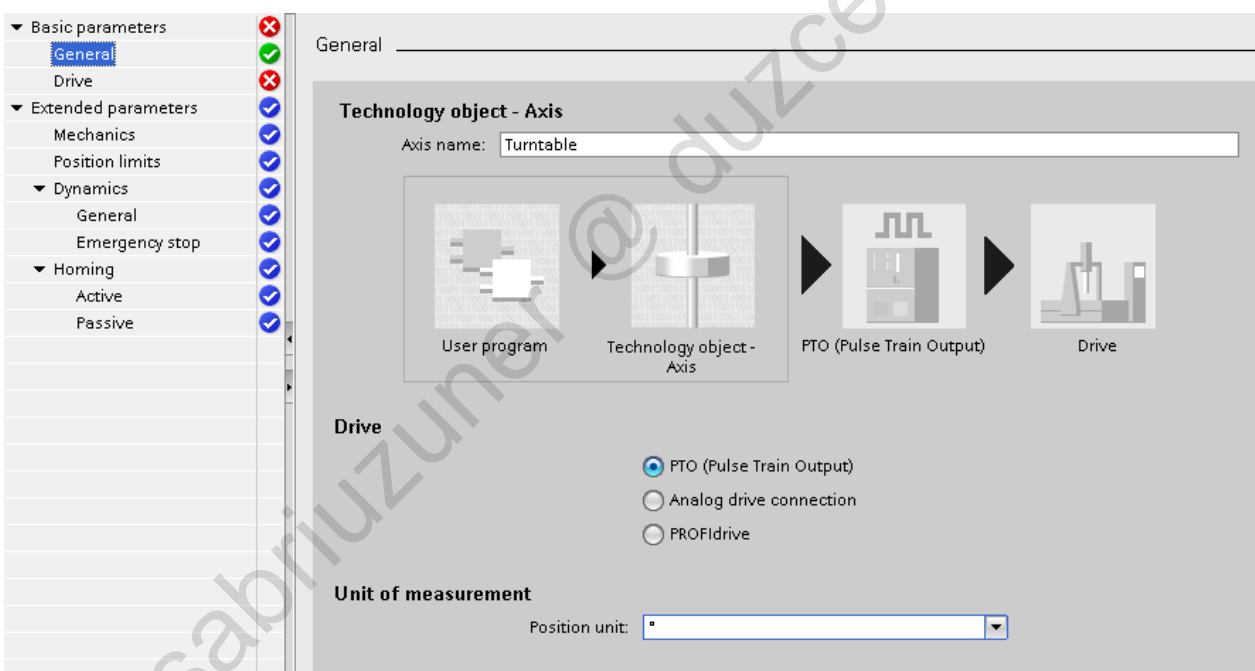


### Task

Create a technology object "Turntable" of the type "Axis" and configure it.

### What to do

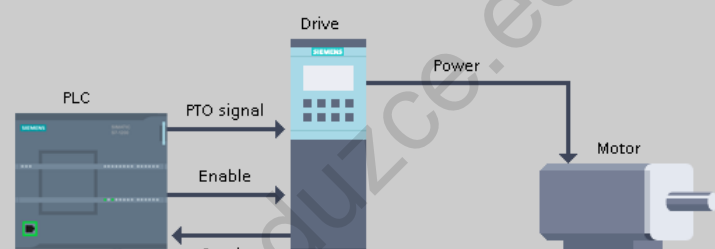
1. Create a new technology object in PLC\_2:  
PLC\_2 → Technology objects → Add new object → Motion Control → TO\_PositioningAxis;  
Name: Turntable
2. Implement the settings shown in the following:





- ▼ Basic parameters
  - General
  - Drive
- ▼ Extended parameters
  - Mechanics
  - Position limits
  - ▼ Dynamics
    - General
    - Emergency stop
  - ▼ Homing
    - Active
    - Passive

Drive



**Hardware interface**

Select pulse generator:  Device configuration

Signal type:

Pulse output:   100 kHz on-board output

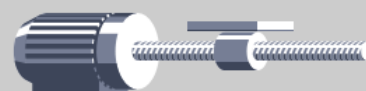
Activate direction output

Direction output:   100 kHz on-board output

**Enable and feedback of the drive**

<b>CPU</b>	<b>Drive</b>
Select enable output:	Drive enable
<input type="text"/>	←
Select ready input:	Drive ready
<input type="text" value="TRUE"/>	→

Mechanics



Pulses per motor revolution:

Load movement per motor revolution:

Permitted direction of rotation:

Invert direction signal

▼ Basic parameters

- General
- Drive

▼ Extended parameters

- Mechanics
- Position limits
- ▼ Dynamics
  - General
  - Emergency stop
- ▼ Homing
  - Active
  - Passive

> General

Note: Changes in the velocity limits affect acceleration and deceleration; the ramp-up time and ramp-down time stay the same.

Velocity

Unit of velocity limits: %/s

Maximum velocity: 200.0 %/s

200.0 %/s

Start/stop velocity: 20.0 %/s

20.0 %/s

Acceleration / deceleration

Acceleration: 360.0 %/s<sup>2</sup>

Deceleration: 360.0 %/s<sup>2</sup>

Ramp-up time: 0.5 s

Ramp-down time: 0.5 s

Enable jerk limit

Smoothing time:  $t_1$ : 0.0 s  $t_2$ : 0.0 s Jerk: 0.0 %/s<sup>3</sup>

Note: By enabling the jerk limit, the total time for acceleration and deceleration of the axis is increased.

▼ Basic parameters

- General
- Drive

▼ Extended parameters

- Mechanics
- Position limits
- ▼ Dynamics
  - General
  - Emergency stop
- ▼ Homing
  - Active
  - Passive

> Emergency stop

Velocity

Maximum velocity: 200.0 %/s

200.0 %/s

Start/stop velocity: 20.0 %/s

20.0 %/s

Deceleration

Emergency deceleration: 720.0 %/s<sup>2</sup>

Emergency stop ramp-down time: 0.25 s

▼ Basic parameters ✔

  General ✔

  Drive ✔

▼ Extended parameters ✔

  Mechanics ✔

  Position limits ✔

▼ Dynamics ✔

  General ✔

  Emergency stop ✔

▼ Homing ✔

  Active ✔

  Passive ✔

> Active

**Digital input homing switch**

Input homing switch:

Select level:

Permit auto reverse at HW limit switch

**Approach/homing direction:**

Positive direction

Negative direction

**Side of homing switch:**

Top side

Bottom side

Approach velocity:  %s

Homing velocity:  %s

— Approaching homing switch

— Homing axis

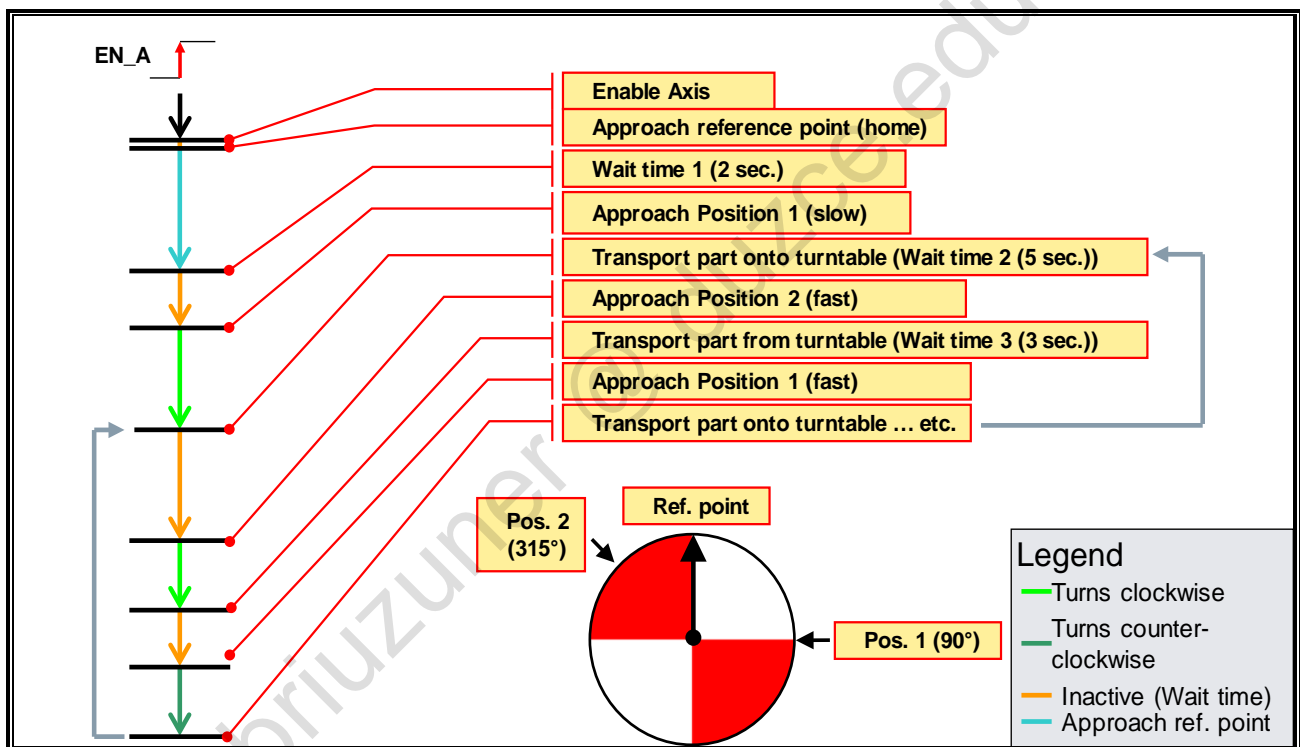
— Travel to home position

Home position offset:

Home position:

3. The settings "Enable and feedback of the drive", "Position limits" as well as "Homing Passive" remain unchanged.
4. Save your project.

## 8.8.3. Exercise 6: Commissioning "FB\_Turntable"

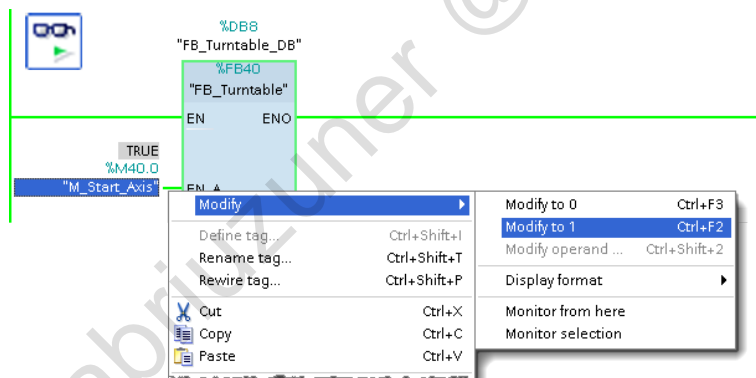


## Task

The control of the technology object "Turntable" is implemented by the function block "FB\_Turntable" (FB40). Insert the block from the Project library into your user program and call it in OB1.

## What to do

1. Using drag & drop, copy the block "FB\_Turntable" from the project library into the program folder of PLC\_2
2. Call "FB\_Turntable" in OB1
3. Connect the input "EN\_A" with the bit memory "M\_Start\_Axis" (M40.0)
4. Save your project and transfer both the hardware and the software to PLC\_2
5. Monitor the call of "FB\_Turntable" in OB1 and control the bit memory "M\_Start\_Axis" as shown in the following



## Result

The axis first carries out the active homing (referencing) and then begins with the motion sequence described in the task.

If the bit memory "M\_Start\_Axis" is reset, the axis ends the current execution and then stops at Position 1.

### 8.8.4. Exercise 7: Starting the axis and monitoring the statuses with the diagnostic panel

The screenshot shows the SIMATIC Manager interface. On the left, the project tree is expanded to show the 'Diagnostics' folder for the 'Turntable [DB40]' device. A red dashed box highlights the 'Diagnostics' folder, and a yellow arrow points to it. The main window displays the 'Status and error bits' for the 'Turntable' axis. The 'Status messages' section shows the axis is 'Enabled', 'Homed', and 'Ready'. The 'Limit switch status messages' section shows that no limit switches have been approached. The 'Error messages' section shows no errors. The 'Motion status' section shows the current position is 315.0 and the target position is 315.0. The 'Dynamics settings' section shows the acceleration and deceleration are both 360.0 %/s².

#### Task

Monitor the current status of the axis and track the motion sequences via the diagnostic panel.

# Contents

<b>9.</b>	<b>Troubleshooting .....</b>	<b>9-2</b>
9.1.	Objectives .....	9-2
9.2.	Categories of errors .....	9-3
9.3.	STEP7 - Test functions, overview .....	9-4
9.4.	System Diagnostics – Overview .....	9-5
9.5.	Online & Diagnostics – Functions .....	9-6
9.5.1.	Diagnostics: Diagnostics buffer .....	9-7
9.5.2.	Diagnostic buffer: Error Messages in the diagnostics buffer .....	9-8
9.5.3.	Diagnostic buffer: Opening a faulty block .....	9-9
9.5.4.	Call hierarchy (block stack) .....	9-10
9.6.	Monitor block .....	9-11
9.6.1.	Monitor block: Modify tags .....	9-12
9.6.2.	Monitoring structures .....	9-13
9.6.3.	Monitor block: Call environment .....	9-14
9.7.	"Monitor / modify variables": Watch tables .....	9-15
9.7.1.	"Monitor / modify variables": Trigger points .....	9-16
9.7.2.	"Enable peripheral outputs" .....	9-17
9.8.	Force variables .....	9-18
9.9.	Reference data: Cross-references of PLC tags .....	9-19
9.9.1.	Reference data: Cross-references of a tag .....	9-20
9.9.2.	Reference data: Go to → Point of use .....	9-21
9.9.3.	Reference data: Call structure .....	9-22
9.9.4.	Reference data: Dependency structure .....	9-23
9.9.5.	Reference data: Assignment of I, Q, M .....	9-24
9.9.6.	Reference data: Resources (memory utilization) .....	9-25
9.9.7.	Reference data: Overlapping accesses .....	9-26
9.10.	Compare (1) - Offline/online .....	9-27
9.10.1.	Compare (2) - Online/offline block detailed comparison .....	9-28
9.10.2.	Compare (3) – Software offline/offline .....	9-29
9.10.3.	Compare (4) - Offline/offline hardware .....	9-30
9.10.4.	Compare (5) - Block-quick compare .....	9-31
9.11.	Exercise 1: Downloading a faulty program in PLC_1 .....	9-32
9.12.	Exercise 2: Errors detected by the system: Reading out the diagnostics buffer .....	9-33
9.13.	Exercise 3: Testing the motor jog .....	9-34
9.14.	TRACE analyzer function .....	9-35
9.14.1.	Configuring a TRACE - Signals and sampling .....	9-36
9.14.2.	Configuring a TRACE – Trigger and saving measurement on device .....	9-37
9.14.3.	Downloading a TRACE configuration into the CPU and activating it .....	9-38
9.14.4.	Evaluating, Saving, Exporting a TRACE in STEP 7 .....	9-39
9.14.5.	Trace task card .....	9-40
9.14.6.	Additional exercise: Creating, viewing and saving a TRACE .....	9-41
9.15.	Additional information .....	9-43
9.15.1.	Monitor block: Display formats .....	9-44
9.15.2.	Monitor block: Call path .....	9-45

## 9. Troubleshooting

### 9.1. Objectives

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

- ... be able to classify errors that occur into the error categories "Errors detected by the system" and "Functional errors"
- ... be able to read out the diagnostics buffer, interpret it and use it for troubleshooting
- ... be able to read out the hardware diagnostics
- ... be able to apply the "Monitor/Modify Variables" test function
- ... be able to interpret the displays of the "Monitor" test function in the LAD/FBD Editor and use them for troubleshooting
- ... be able to read out the reference data, interpret it and use it for troubleshooting
- ... understand the "Enable peripheral outputs" and "Force" functions
- ... become familiar with the Trace function

#### Objectives

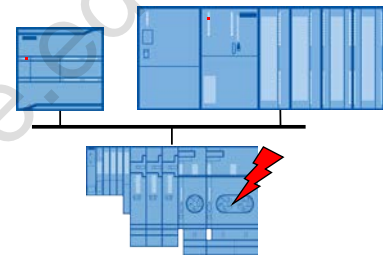
In this chapter, the tools for troubleshooting are presented. Tools for functional errors as well as tools for errors which the system itself detects are dealt with.



## 9.2. Categories of errors

### Errors Detected by the System

- Acquiring, evaluating and indicating errors within a PLC
  - Module failure
  - Short-circuit in signal cables
  - Scan time overrun
  - Programming error (accessing a non-existent block)



### Functional Errors

- Desired function is either not executed at all or is not correctly executed
  - Process fault (sensor/actuator, cable defective)
  - Logical programming error (not detected during creation and commissioning)



### Monitoring functions

Diagnosis is important in the operating phase of a system or machine. Diagnosis usually occurs when a problem (disturbance) leads to standstill or to the incorrect functioning of the system or machine. Due to the costs associated with downtimes or faulty functions, the associated cause of the disturbance must be found quickly and then eliminated.

### Categories of errors

Errors that occur can be divided into two categories, depending on whether they are detected by the PLC:

- Errors that are detected by the PLC's operating system.
- Functional errors, that is, the CPU executes the program as usual, but the desired function is either not executed at all or it is executed incorrectly. The search for these types of errors is much more difficult, since the cause of the error is initially hard to determine. Possible causes could be:
  - A logical programming error (software error) that was not detected during creation and commissioning of the user program and probably occurs only on extremely rare occasions
  - A process fault that was triggered by the faulty functioning of components directly associated with the process control, such as cables to sensors/actuators or by a defect in the sensor/actuator itself

### 9.3. STEP7 - Test functions, overview

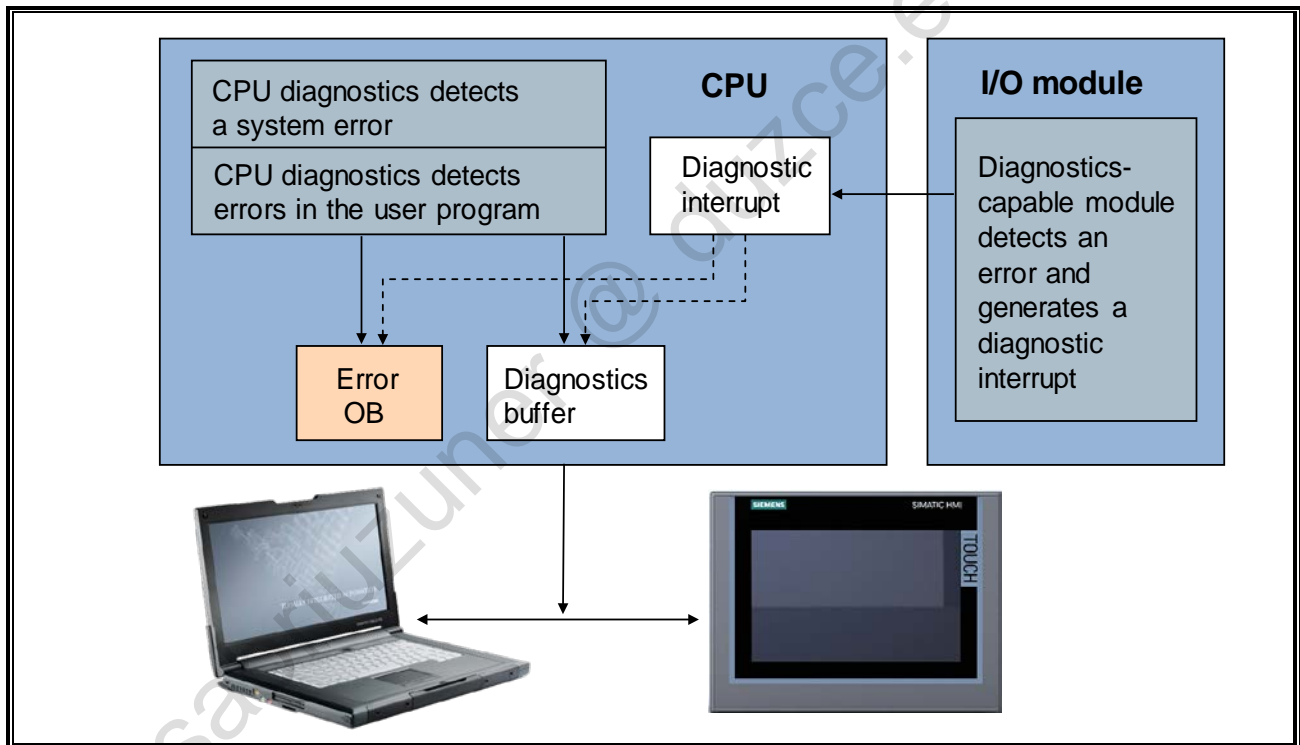
Errors Detected by the System General rule: CPU in <b>Run</b>	Functional Errors General rule: CPU in <b>RUN</b>
<ul style="list-style-type: none"> <li>• Asynchronous error (w/o OB82, 83, 86): <b>RUN</b></li>   <li>• <b>Online &amp; Diagnostics</b> <ul style="list-style-type: none"> <li>- Diagnostics buffer,</li> </ul> </li> <li>• <b>Task Card "Testing"</b> <ul style="list-style-type: none"> <li>- Call hierarchy / Block stack</li> <li>- Local data stack (in planning stage)</li> </ul> </li> <li>• <b>Diagnose Modules</b> <ul style="list-style-type: none"> <li>- Diagnostic status (all modules)</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Process fault (e.g. wire break)</li> <li>• Logical programming error (e.g. double assignment)</li>   <li>• <b>Monitor and Modify Variables</b> <ul style="list-style-type: none"> <li>→ Watch and Force tables</li> </ul> </li> <li>• <b>Monitor Blocks (Block Status)</b> <ul style="list-style-type: none"> <li>→ Monitor in the Blocks editor</li> <li>- with call environment</li> </ul> </li> <li>• <b>Tools</b> <ul style="list-style-type: none"> <li>- Cross references</li> <li>- Assignment list (I/Q/MT/C)</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li>• <b>"Trace" analyzer function</b></li> <li>• <b>Program/Block comparison</b></li> </ul>	

#### Test Functions

There are various STEP 7 test functions for troubleshooting, depending on the type of error caused:

- For errors that are detected by the system, the test functions diagnostics buffer and hardware diagnostics give detailed information on the cause of the error and the location of the interruption. By programming error OBs, information on the error that occurred can be evaluated by program and the transition of the CPU into the STOP state can be prevented. If the CPU has stopped, the use of the test functions monitor / modify variable and monitor blocks makes little sense since the CPU neither reads nor outputs process images while in the STOP state, and no longer executes the program.
- Vice versa, it makes little sense, as a rule, to use test functions such as module information when the CPU is in RUN. The Module Information test function merely provides general information on the CPU's operating status or on errors that occurred in the past. Functional errors can be diagnosed as follows:
  - Process fault (such as, wiring error)
    - Wiring test of the inputs: Monitor variable
    - Wiring test of the outputs: Enable peripheral outputs (only for CPU STOP) (in planning)
  - Logical programming error (such as, double assignment)
    - All test functions listed, except for Enable peripheral outputs, can be used for searching for logical program errors
  - Force
    - Forced control of operands regardless of the program logic

## 9.4. System Diagnostics – Overview



### System diagnostics

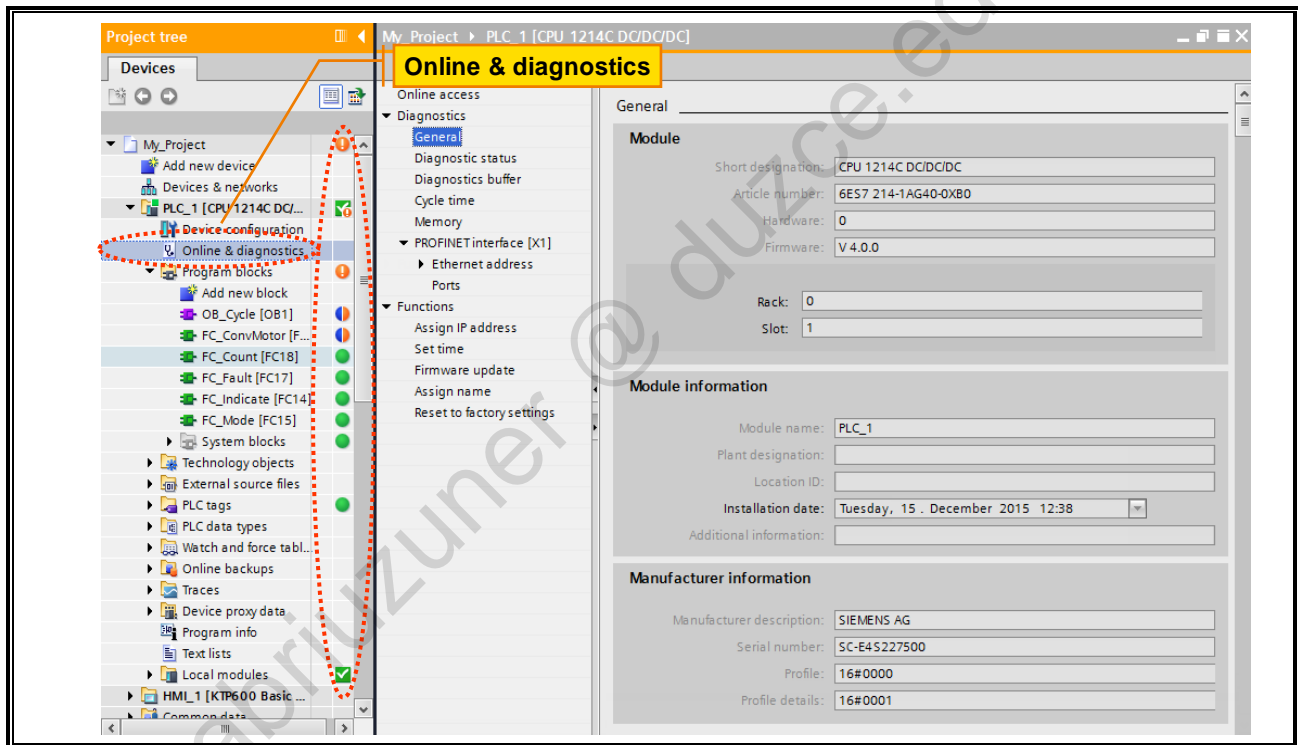
All those monitoring functions that deal with the correct functioning of the components of an automation system are grouped together under system diagnostics. All S7-CPU's have an intelligent diagnostics system. The acquisition of diagnostic data by the system diagnostics does not have to be programmed. It is integrated in the operating system of the CPU and in other diagnostics-capable modules and runs automatically. The CPU (temporarily) stores errors that occur in the diagnostics buffer and thus enables a fast and targeted error diagnosis by service personnel, even for sporadically occurring errors.

### System reaction

The operating system takes the following actions when it detects an error or a STOP event, such as an operating mode change (RUN → STOP):

- A message on the cause and the effect of the occurring error is entered in the diagnostics buffer, complete with the date and time. The diagnostics buffer is a FIFO (circular) buffer on the CPU module for storing error events. In the FIFO buffer structure, the most recently entered message overwrites the oldest diagnostics buffer entry. A CPU memory reset CANNOT delete the diagnostics buffer, only a "Reset to factory settings". The diagnostics buffer can be read out using the programming device or the panel.
- The Error OB associated with this error is called. This gives the user the opportunity of carrying out his own error handling.

## 9.5. Online & Diagnostics – Functions



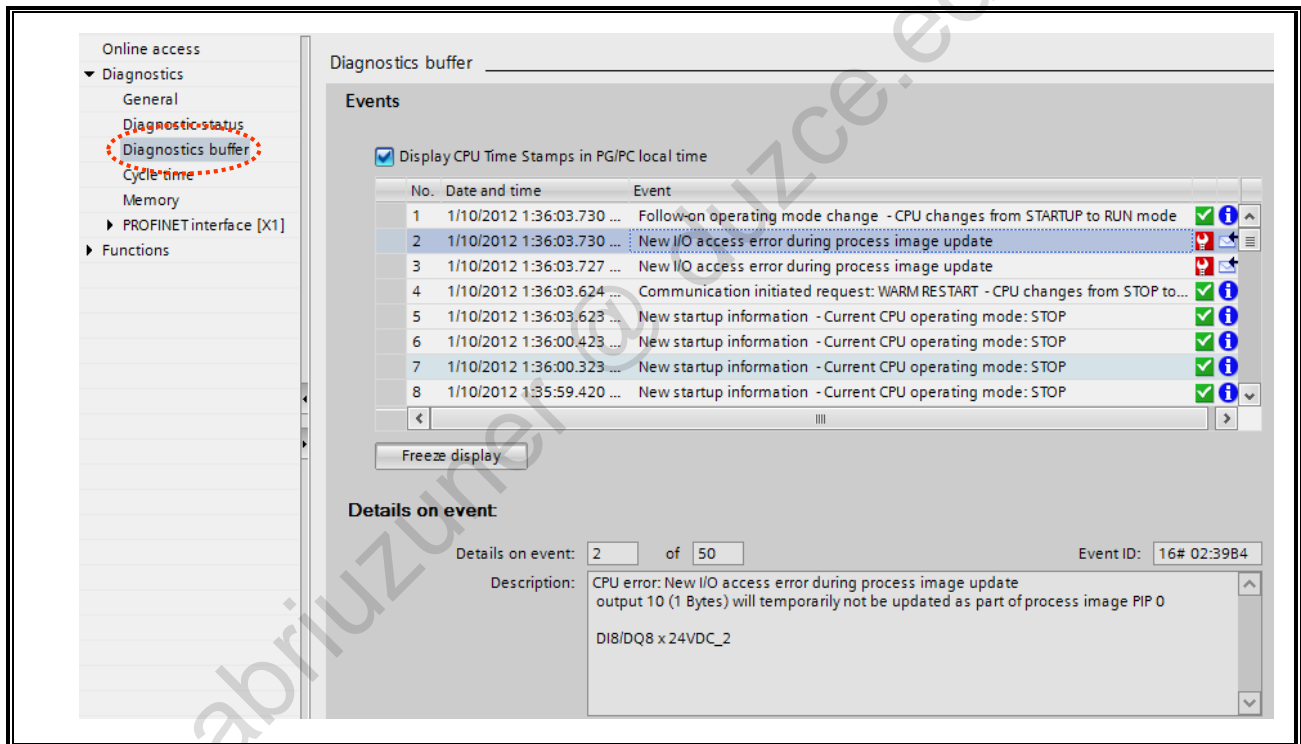
### Status display

After the online connection was successfully established, the user interface changes as follows:

- The title bar of the active window gets an orange colored background
- The title bars of the inactive windows of the associated station get an orange colored line at the lower edge
- In the project tree, operating state or diagnostic symbols are displayed for the objects of the associated station
- In the inspector window, the area "Diagnostics > Device Information" is brought to the foreground

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

### 9.5.1. Diagnostics: Diagnostics buffer



#### Diagnostic buffer

The diagnostic buffer is a buffered memory area of the CPU organized as a ring buffer. It contains all diagnostic events (error messages, diagnostic alarms, start-up information, etc.) of the CPU in the order in which they occurred. The topmost entry is the last event that occurred.

All events can be displayed on the programming device in plain text and in the order of their occurrence.



The size of the diagnostic buffer depends on the CPU. Likewise, the entire diagnostic buffer is not buffered when the power is OFF (only a part is retentive).

#### Event details

Some additional information about the selected event is provided in the field "Event details":

- Name of the event and event number
- Additional information, depending on the event, such as the address of the statement that caused the event, and so on.

## 9.5.2. Diagnostic buffer: Error Messages in the diagnostics buffer

The screenshot displays the 'Diagnostics buffer' window in SIMATIC Manager. On the left, a navigation tree shows 'Diagnostics' expanded to 'Diagnostics buffer'. The main area is divided into 'Events' and 'Details on event'. The 'Events' section contains a table with columns for 'No.', 'Date and time', and 'Event'. Five entries are visible, all with the same description: 'DB 5 not loaded, access in FC 18 - Processing will continue (no OB processing)'. The 'Details on event' section shows 'Event ID: 16# 02:253A' and a description: 'Temporary CPU error: DB 5 not loaded, access in FC 18 affecting OB 1 execution Processing will continue (no OB processing)'. Below the description, it lists 'PLC\_1' and 'Internal address details: Caddr=16#0000003B, area: DB 5 , addr: 0'. Buttons for 'Help on event', 'Open in editor', and 'Save as...' are at the bottom.

### Interpreting the diagnostics buffer

To interpret the diagnostics buffer, you must look at the events that belong together in the sequence in which they occurred, in other words, from bottom to top.

If errors occur in which the CPU does not change to STOP, an entry is made in the diagnostics buffer in every program cycle.

### Entries in the diagnostics buffer

The last error that occurred after this warm restart leads to the following entries in the diagnostics buffer: Example:

Event No. 1:

Temporary CPU error: Area length error in FC18 – Processing will continue

Description:

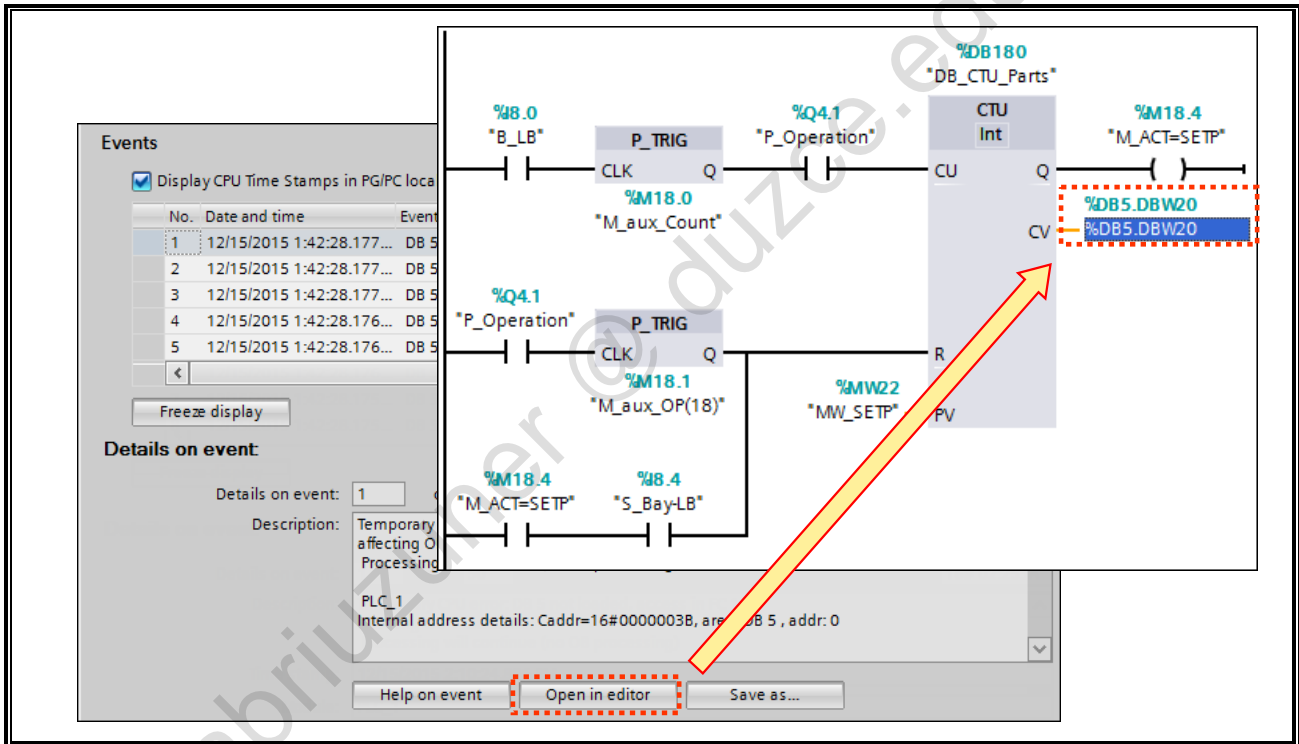
Temporary CPU error: Area length error in FC18  
affects OB1 execution  
Write-access DB area  
faulty address ignored (← system reaction)

Processing will continue (no OB processing)



If a runtime error occurs during the execution of the STEP 7 user program, the CPU as of Firmware 2.0 remains in RUN

### 9.5.3. Diagnostic buffer: Opening a faulty block



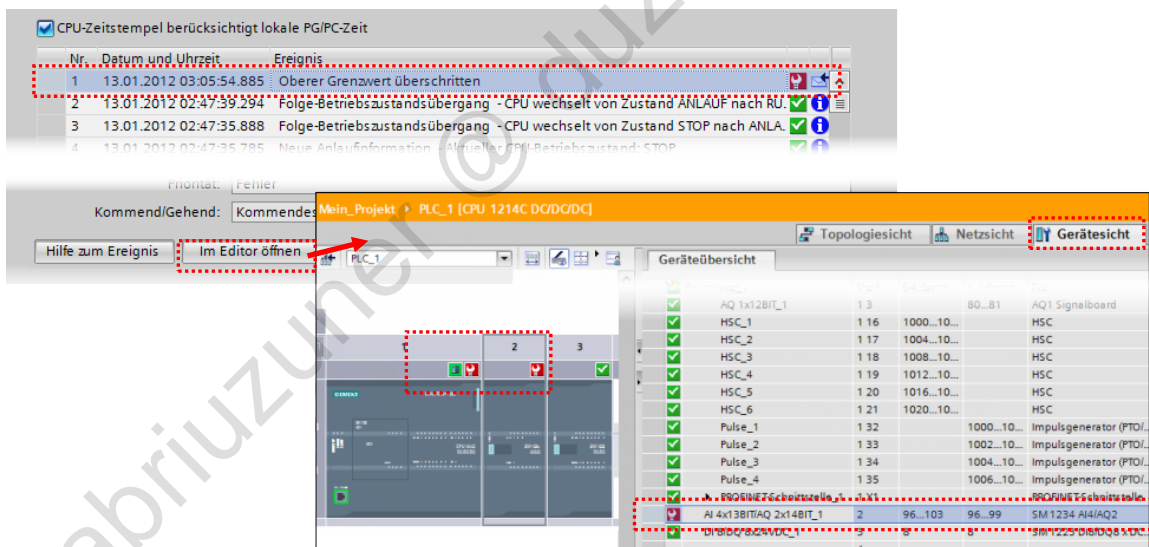
#### Opening a block

For synchronous errors, that is, for errors that were triggered by a faulty instruction in the user program, you can open the block in which the interruption occurred by clicking on the "Open in editor" button.

In LAD/FBD, the network causing the interruption is highlighted. In the example shown, the DBW 20 of DB5 is accessed which does not exist.

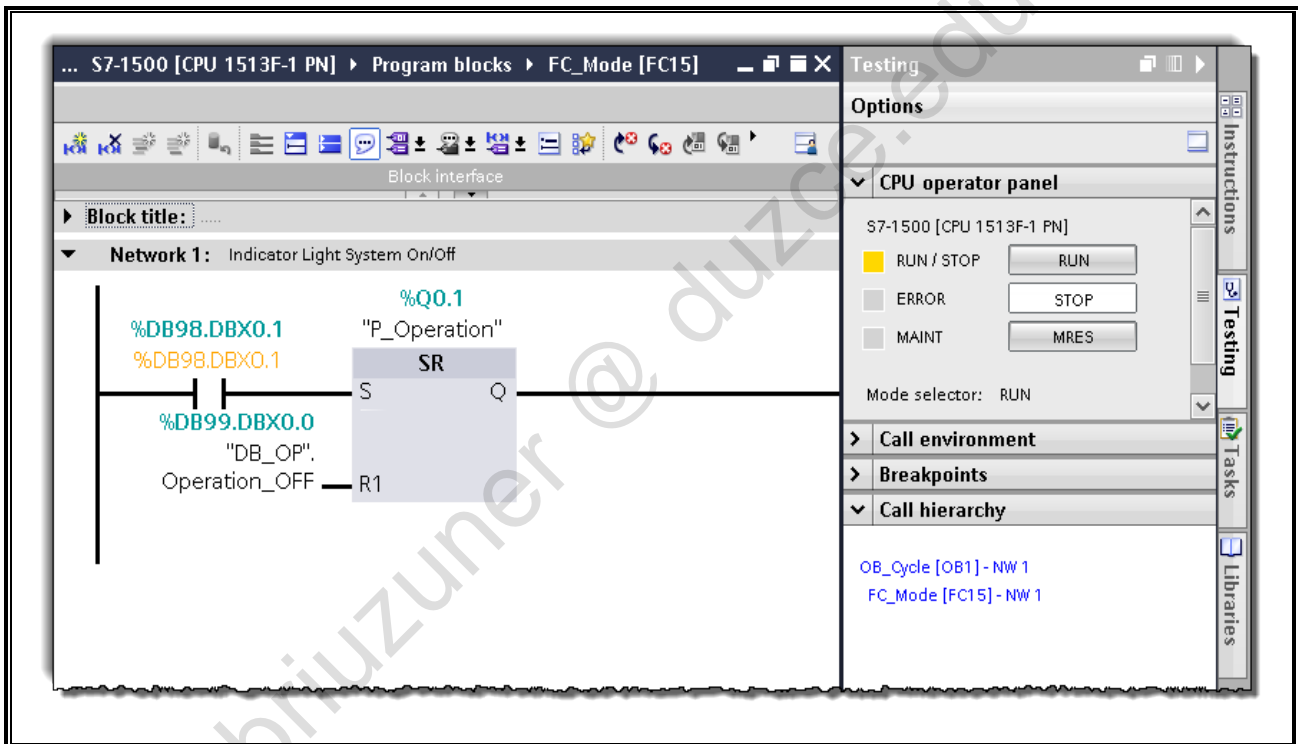
#### "Open in editor"

With the "Open in editor" button, the associated editor that is relevant for the entry (the event) is opened.





### 9.5.4. Call hierarchy (block stack)



#### Call hierarchy

The "Call hierarchy" gives you the information in which call path the block is opened.

If the block was opened from the diagnostic buffer via the button "Open in editor", then by looking at the entry in the Call hierarchy, you can see in which path the error occurred.

You can open the calling block by clicking on the appropriate link.

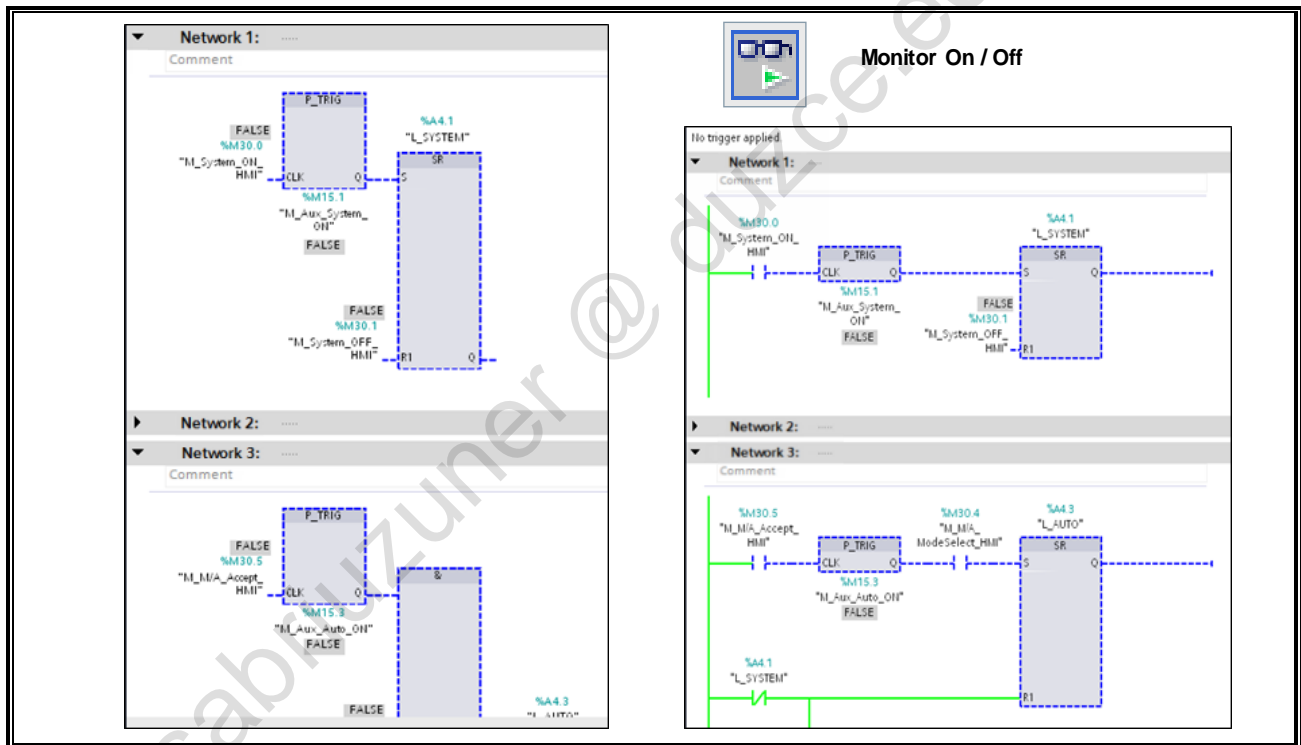
#### Note

If the CPU is in "STOP" mode, the current block stack at the time of the STOP transition can be read out via the call hierarchy. The block stack lists all blocks whose execution was not completed at the moment when the CPU went into "STOP". The blocks are listed in the order in which the execution was started.

With an existing online connection, open any block for this and switch into the Task Card "Test" > Call hierarchy.



## 9.6. Monitor block



### Application

The test function Monitor block is used to track the program processing within a block. The states or contents of the operands used in the block at the time of program processing are displayed on the screen. The test mode "Observe" ("Block status") for the block currently opened in the LAD/FUP/SCL editor is activated via the eyeglass symbol.

To start the test function, the block must be identical online and offline. If the opened block does not correspond to the block stored online in the CPU, the opened block must be loaded into the CPU or from the CPU and then observed before monitoring.

In the test mode the states of the operands and LAD / FBD elements are represented by different colors. The settings for this are to be made via Extras → Settings:

Example:

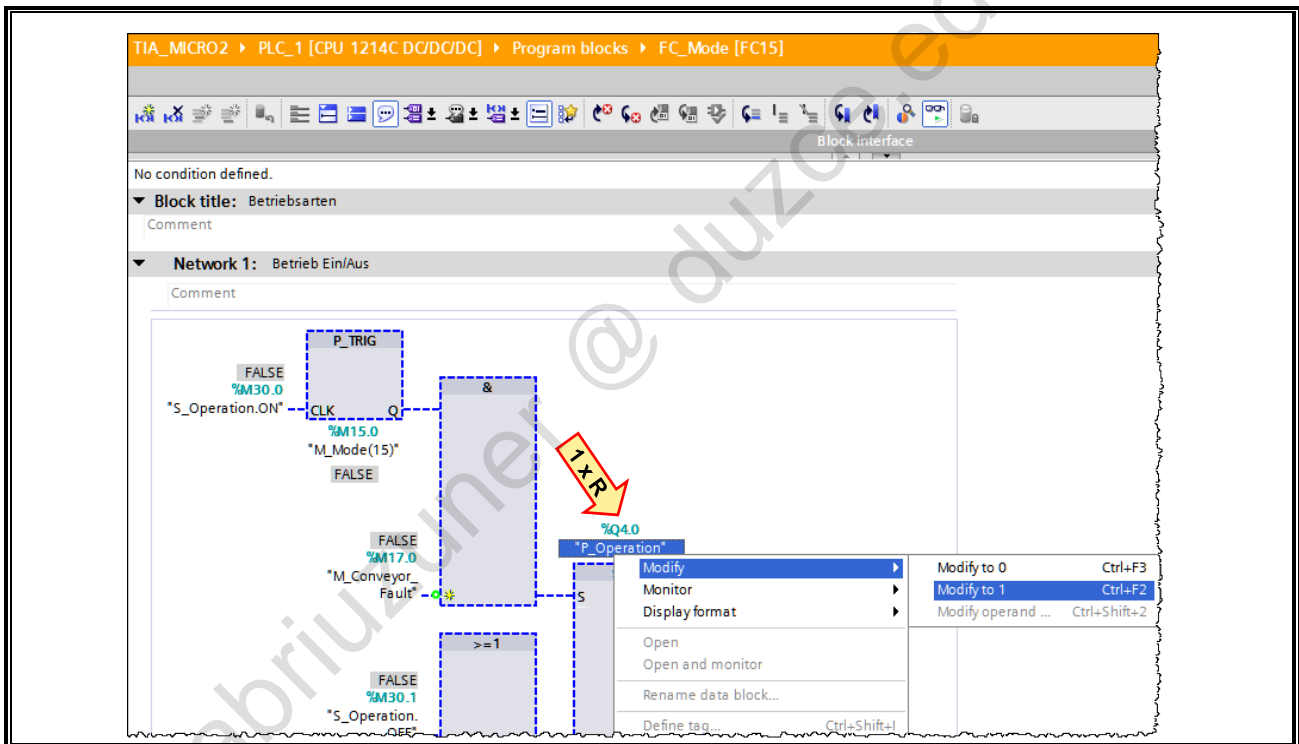
- Status fulfilled → "Element is displayed in green color"
- Status not fulfilled → "Element is displayed in blue color"

### Note

The observation values are only current if the CPU is in RUN state and the instructions to be observed are processed!

This is indicated by the progress bar "Online values are updated" in the upper right corner of the block.

### 9.6.1. Monitor block: Modify tags



#### Modify tags

If the test function "Monitor block" is activated, it is possible to control variables to status '0' or '1'. The assignment of the status is done once. With non-Boolean variables, the control can be carried out via the menu item "Modify operand...".

If the variable whose status was changed is not overwritten by the program, the variable remains at the assigned status. If, for example, an output is controlled to status '1' and this variable is not overwritten by the program, the output remains switched on or set to status '1'.

## 9.6.2. Monitoring structures

The screenshot shows the SIMATIC Manager interface. At the top, a variable declaration for `"DB_Weights_opt".WeightStore` is visible, with fields `SetpNo`, `Weight`, and `WeightStore`. A yellow box labeled "Monitor Structures" points to this declaration. Below, the "Diagnostics" window is open to the "Monitor value" tab. A yellow box labeled "For INOUT parameters Input value and Output value" points to the "Input value" and "Output value" columns. The table below shows the monitored values for the structure's elements.

Name	Data type	Input value	Output value
▼ "DB_Weights_opt".Wei...	"UDT_WeightStore"		
MaxNo	Int	10	10
ActNo	Int	0	1
▼ PartWeight	Array[0..9] of Int		
PartWeight[0]	Int	238	238
PartWeight[1]	Int	0	160
PartWeight[2]	Int	0	0
PartWeight[3]	Int	0	0

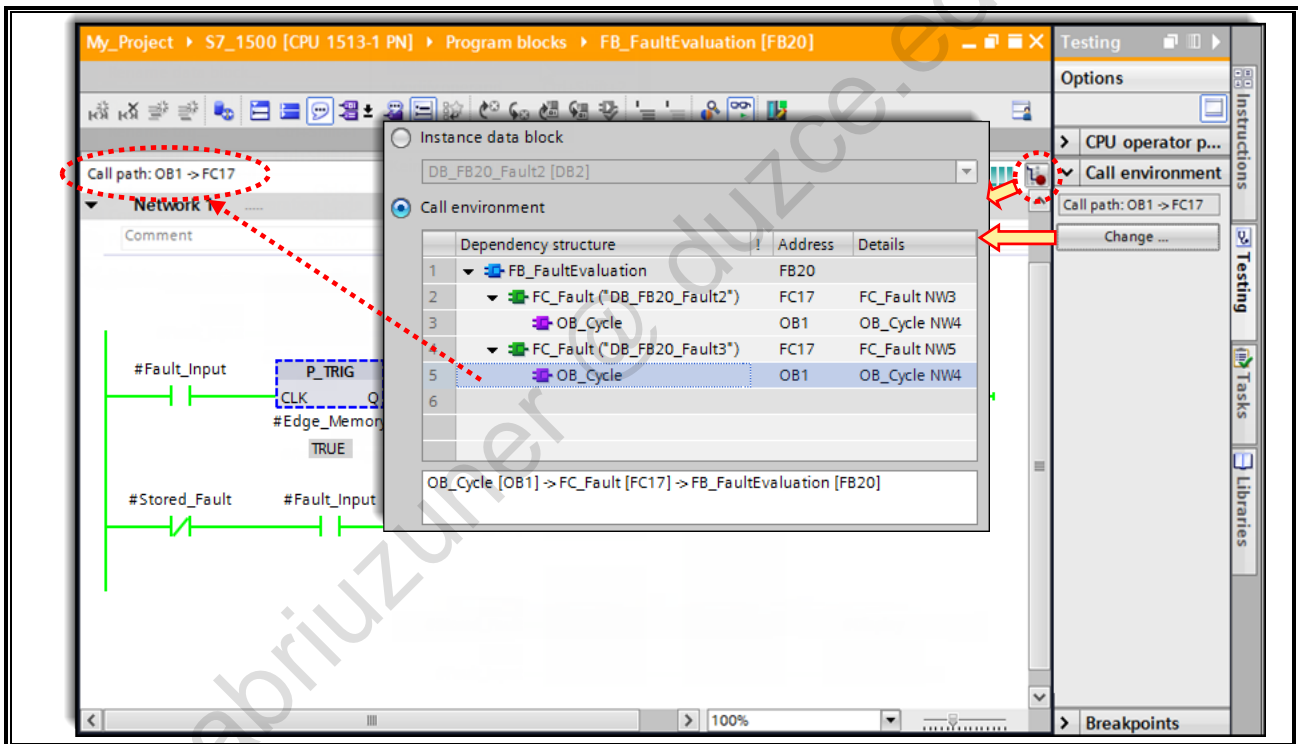
### Rules for Monitoring Structures (S7-1200/1500)

When monitoring structures, the values of a structured PLC tag are displayed in the Inspector window → Diagnostics → Monitor value, with the following exception:

- Structures, whose elements have adjustable retentive properties, cannot be monitored.

In order to display the Monitor values for a structure, you must first activate the monitoring via the Context menu.

### 9.6.3. Monitor block: Call environment

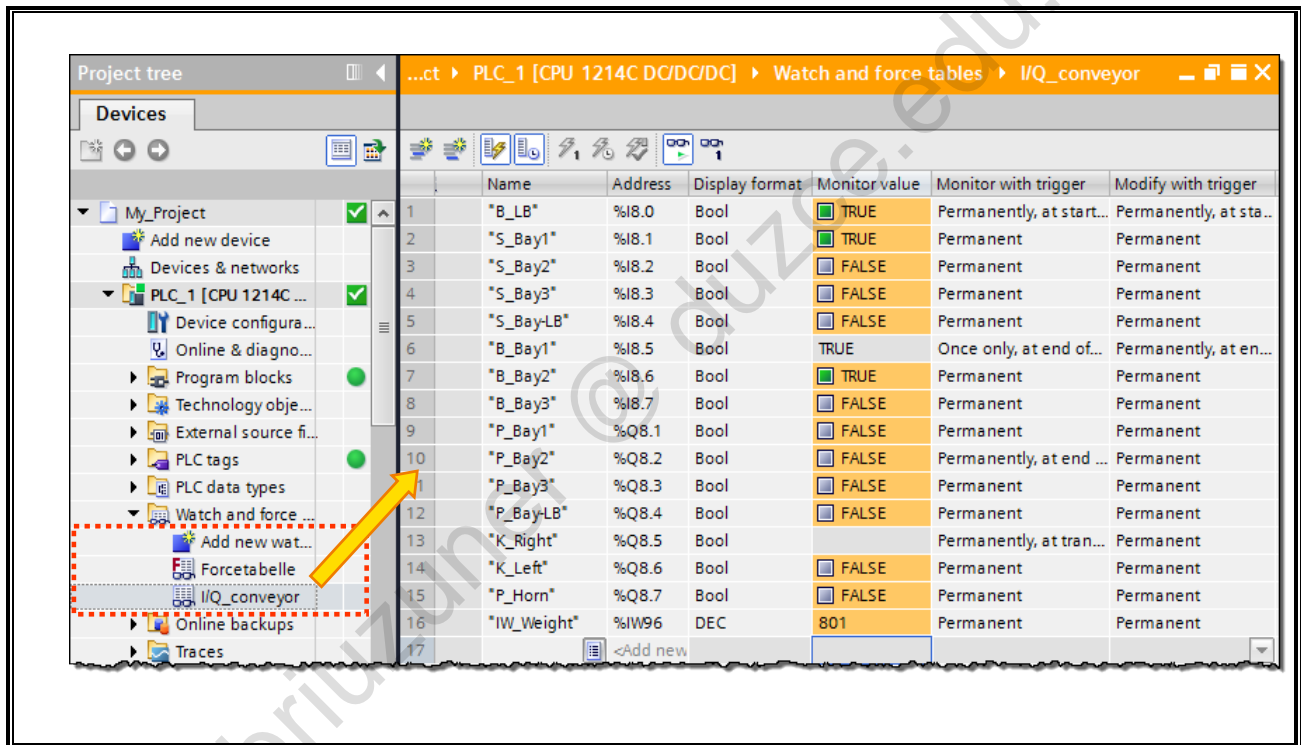


#### Function

The call conditions for blocks and for breakpoints can be defined. In this way we can determine under which conditions the program status of a block is displayed or the program execution is interrupted at a breakpoint. The following conditions can be selected:

- Instance data block  
The program status of a function block is only displayed when the function block is called with the selected instance data block.
- Call path  
The program status of a block is only displayed when the block is called by a specific block or from a specific path.

## 9.7. "Monitor / modify variables": Watch tables



### Area of use

The "Monitor/modify variables (Tags)" test function is used to monitor and / or modify variables (tags) in any format you choose. For this, the desired variables are entered in a watch table. Except for block-local, temporary variables, you can monitor and/or modify all variables (tags) or operands.

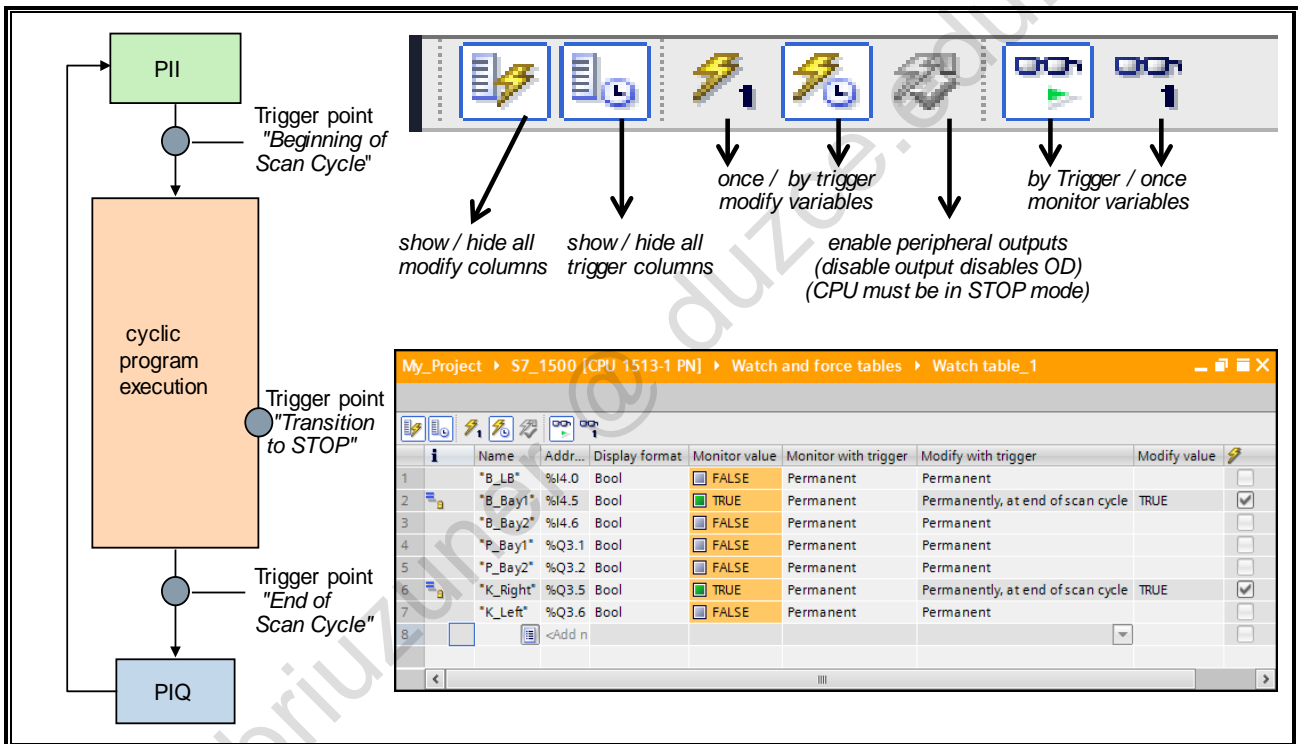
You can choose the columns displayed in the watch table via the menu 'view'. The columns have the following meanings:

- Name: symbolic name of the variable (tag)
- Address: absolute address of the variable (tag)
- Symbol comment: comment on the variable (tag) displayed
- Display format: a data format you can choose per mouse click (such as binary or decimal), in which the content of the variable (tag) is displayed
- Monitor value: variable (tag) value in the selected status format
- Modify value: value to be assigned to the variable (tag)

### Watch table

You can choose any name for the Watch table. Saved Watch tables can be reused to monitor and modify so that a renewed input of the variables to be monitored is no longer necessary.

### 9.7.1. "Monitor / modify variables": Trigger points



#### Trigger points

Through the "monitor with trigger or modify with trigger" columns, you can define the trigger points for monitoring and modifying. The "trigger point for monitoring" specifies when the values of the variables being monitored are to be updated on the screen. The "trigger point for modifying" specifies when the given modify values are to be assigned to the variables being modified.

#### Trigger condition

The "trigger condition for monitoring" specifies whether the values are to be updated on the screen once only when the trigger point is reached or permanently (when the trigger point is reached).

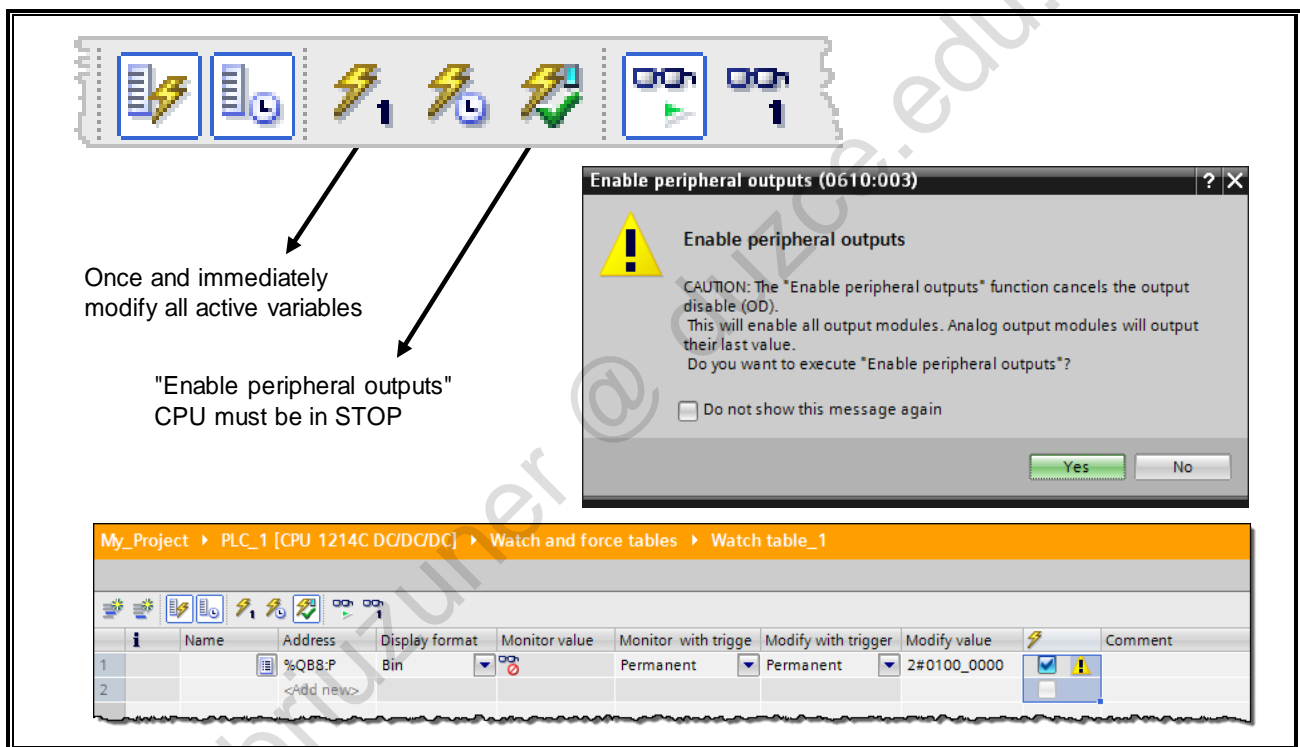
The "trigger condition for modifying" specifies whether the given modify values are to be assigned to the variables being modified once only or permanently (every time the trigger point is reached).

#### Area of use

The following tests, among others, can be implemented with the appropriate selection of trigger points and conditions:

- Wiring test of the inputs:  
Monitor variables, trigger point: Start of scan cycle, trigger condition: Permanent
- Simulate input states (user specified, independent of process):  
Modify variables, trigger point: Start of scan cycle, trigger condition: Permanent
- Differentiation between hardware / software errors (an actuator that should be activated in the process is not controlled)  
Monitor variables, in order to monitor the relevant output, trigger point: End of scan cycle, trigger condition: Permanent  
(output state = '1' > program logic OK > process error (hardware))  
(output state = '0' > program logic error (such as double assignment))
- Control outputs (independent of the program logic)  
Modify variables, trigger point: End of scan cycle, trigger condition: Permanent

## 9.7.2. "Enable peripheral outputs"



### The function "Enable peripheral outputs"

The "Enable peripheral outputs" function is used to check the functioning of the output modules, the wiring of the digital output modules or it can be used to continue to control actuators in the process even though the CPU finds itself in the STOP state because of an error that has occurred.

The "Enable peripheral outputs" function cancels the output disable of the peripheral outputs (PQ), which enables you to control the outputs despite the CPU's STOP state.

### Conditions

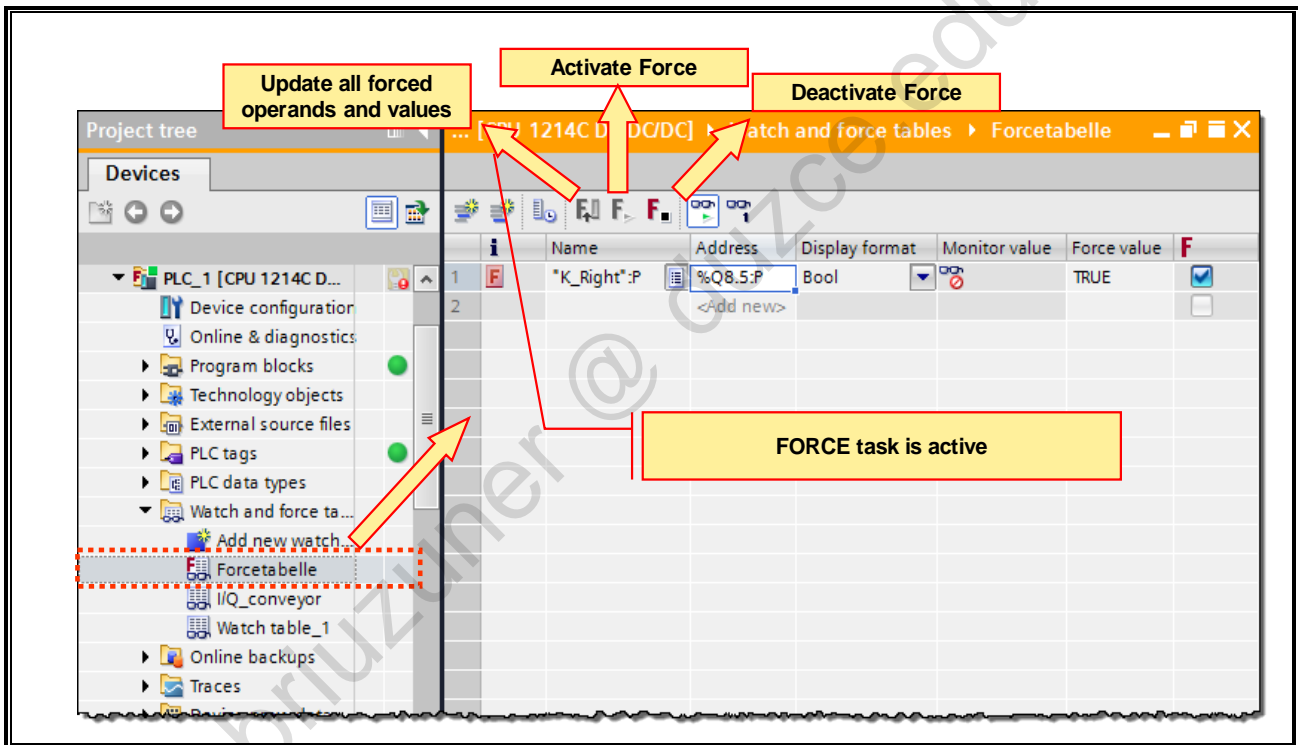
- The CPU must be in STOP mode
- A Force task must not be active in the CPU
- The Watch table must be displayed in "extended mode", in other words, displayed with trigger columns
- The peripheral outputs to be enabled are to be specified byte by byte, word by word or double-word by double-word with the suffix :P (for peripheral)
- After the peripheral outputs have been enabled, the modify values can be activated via the "Modify once only" button (not via "Modify with trigger")

### Note

When changing the CPU's operating status from STOP to RUN or STARTUP, enable peripheral outputs is deactivated and a message pops up.



## 9.8. Force variables



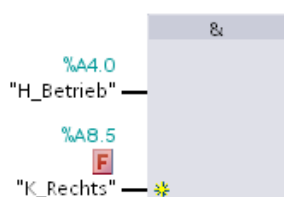
### Introduction

With the help of the force table, you can pre-assign individual variables of the user program with fixed values. These variables are then independently and continuously overwritten by the CPU. This process is called forcing. Requirement for forcing is that an online connection to the CPU exists and that the CPU used supports this function.

### Area of use

Through the fixed pre-assignment of variables with defined values, you can set certain specifications for your user program and thus test the programmed functions.

Forced variables and PLCs are identified with an F, as soon as you switch to the online view or the respective block is monitored.



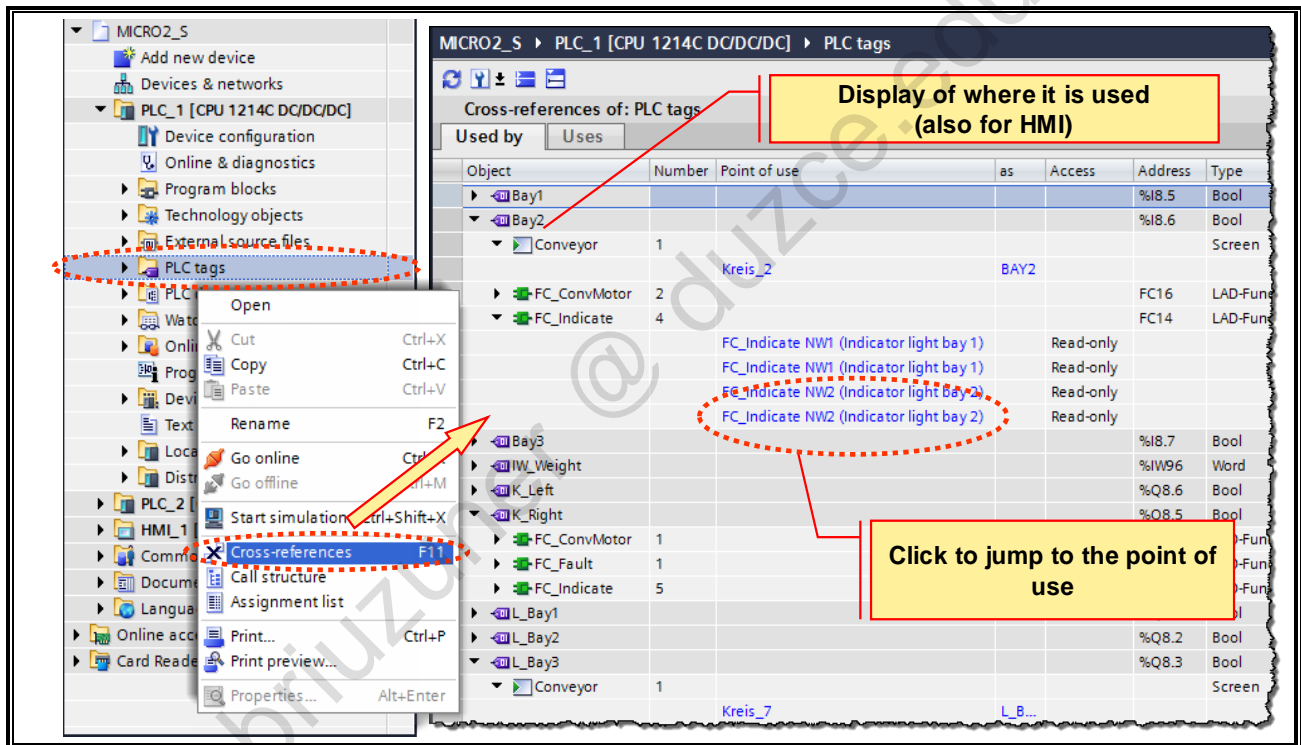
- With an active Force task, the MAINT-LED lights up on the CPU.
- Only physical inputs and outputs (in other words ":P") can be forced.

### Careful

Before forcing, you must become familiar with the safety precautions for this process.



## 9.9. Reference data: Cross-references of PLC tags



### Introduction

The cross-references list offers an overview of the use of operands and variables (tags) within the user program. From the cross-references list, you can jump directly to the point of use.

The cross-references list contains the following information:

- Which operand is used in which block with which instruction?
- Which tag is used in which HMI screen?
- Which block is called by which other block?

As part of the project documentation, the cross-references supply a comprehensive overview of all operands, memory areas, blocks, variables (tags) and screens used.

### Views

There are two views of the cross-references list which differentiate themselves by which objects are displayed in the first column:

- Used by:  
Displays the referenced objects  
Here, the reference location where the object is used are displayed.
- Used:  
Displays the referencing objects.  
Here, the users of the object are displayed.  
The associated tooltips give further information on the respective objects.

### Show Unused

This is a list of tags which are declared in the PLC tag table but are not used in the S7 user program.

### 9.9.1. Reference data: Cross-references of a tag

The screenshot displays a Ladder Logic network with the following components:

- Inputs: %M30.0 "S\_ON", %M15.0 "\*M\_aux\_Op(15)\*", %M17.7 "\*M\_max\_Fault\*" (via a normally open contact).
- Logic: A pulse trigger (P\_TRIG) block with a set coil (S) connected to %Q4.1 "\*P\_Operation\*".
- Output: %M17.0 "\*M\_Conv\_Fault\*" (via a normally open contact).

The Inspector window shows the following cross-reference information for the selected object "\*P\_Operation\*":

Object	Point of use	as	Access	Address	Type
*P_Operation*				%Q4.1	Bool
FC_Mode	FC_Mode NW1 (Operation ON/OFF)		Read and write	FC15	LAD-Function
FC_ConvMotor	FC_ConvMotor NW1 (Jog Right)		Read-only	FC16	LAD-Function
	FC_ConvMotor NW2 (Jog Left)		Read-only		
	FC_ConvMotor NW3 (Automatic transports)		Read-only		
FC_Count				FC18	LAD-Function
FC_Fault				FC17	LAD-Function
FC_Indicate				FC14	LAD-Function

#### Introduction

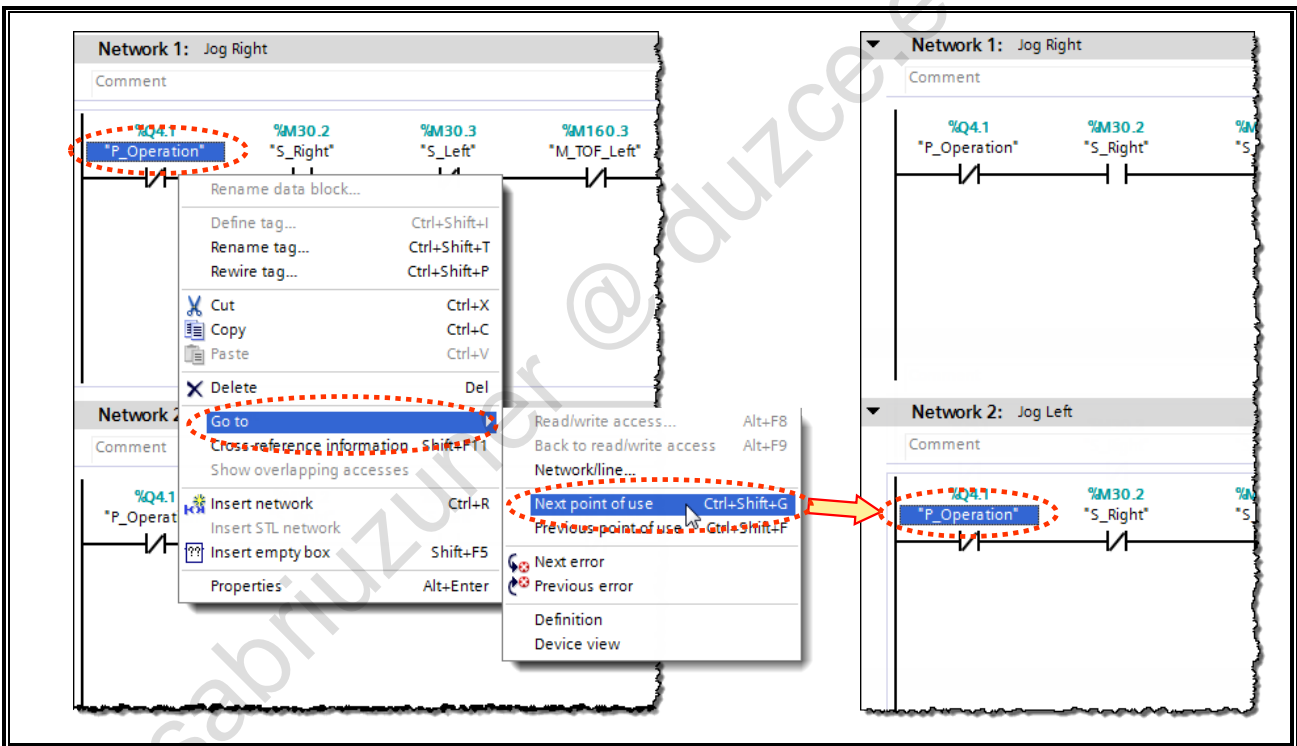
In the Inspector window, the cross-reference information for a selected object is displayed in the tabs "Info > Cross-references". In this tab, you will see at which locations (Point of use) and from which other objects every selected object is used.

In the Inspector window, cross-references are made even to those blocks which only exist online or cross-references from HMI accesses.

#### Structure

The cross-reference information is displayed in tabular form in the Inspector window. Each column contains specific detailed information on the selected object and its use.

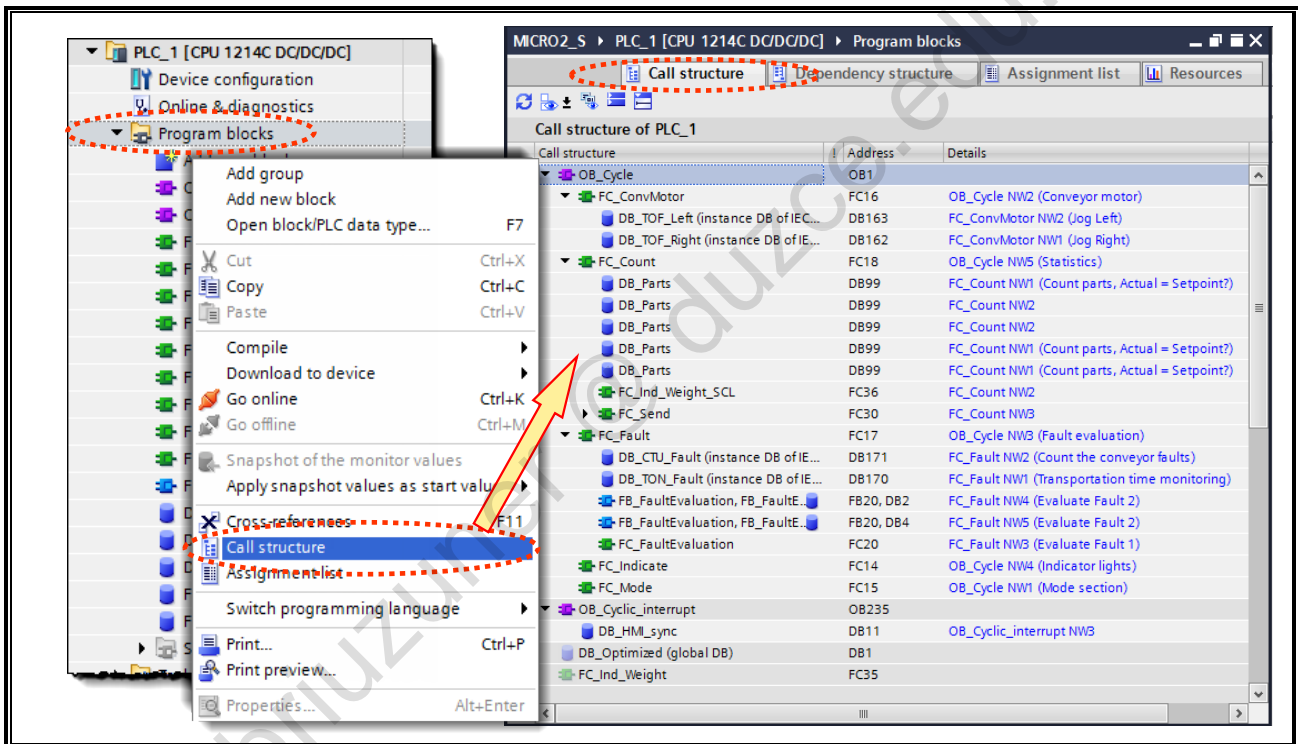
9.9.2. Reference data: Go to → Point of use



Go to → Next or previous point of use

During block processing, in order to navigate quickly within a block from one point of use to the next or previous point of use, the function "Go to > Next point of use" or "Go to > Previous point of use" is started via the context menu of the respective variable (tag).

### 9.9.3. Reference data: Call structure

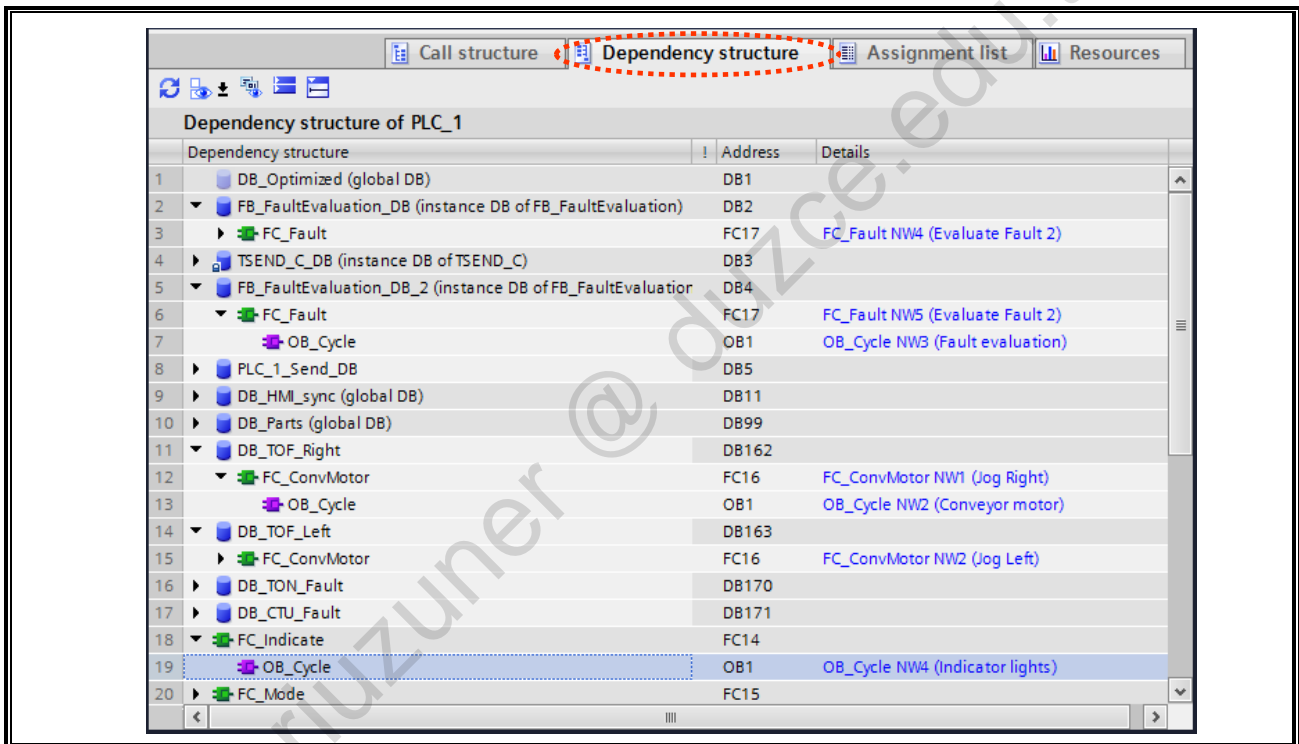


#### Call structure

The call structure shows (describes) the call hierarchy of the blocks within an S7 program. It can be opened via context menu or in menu "tools". It gives you an overview of:

- The blocks used
- Jumps to the points of use of the blocks
- Dependencies between the blocks
- Local data requirements of the blocks
- Status of the blocks

### 9.9.4. Reference data: Dependency structure



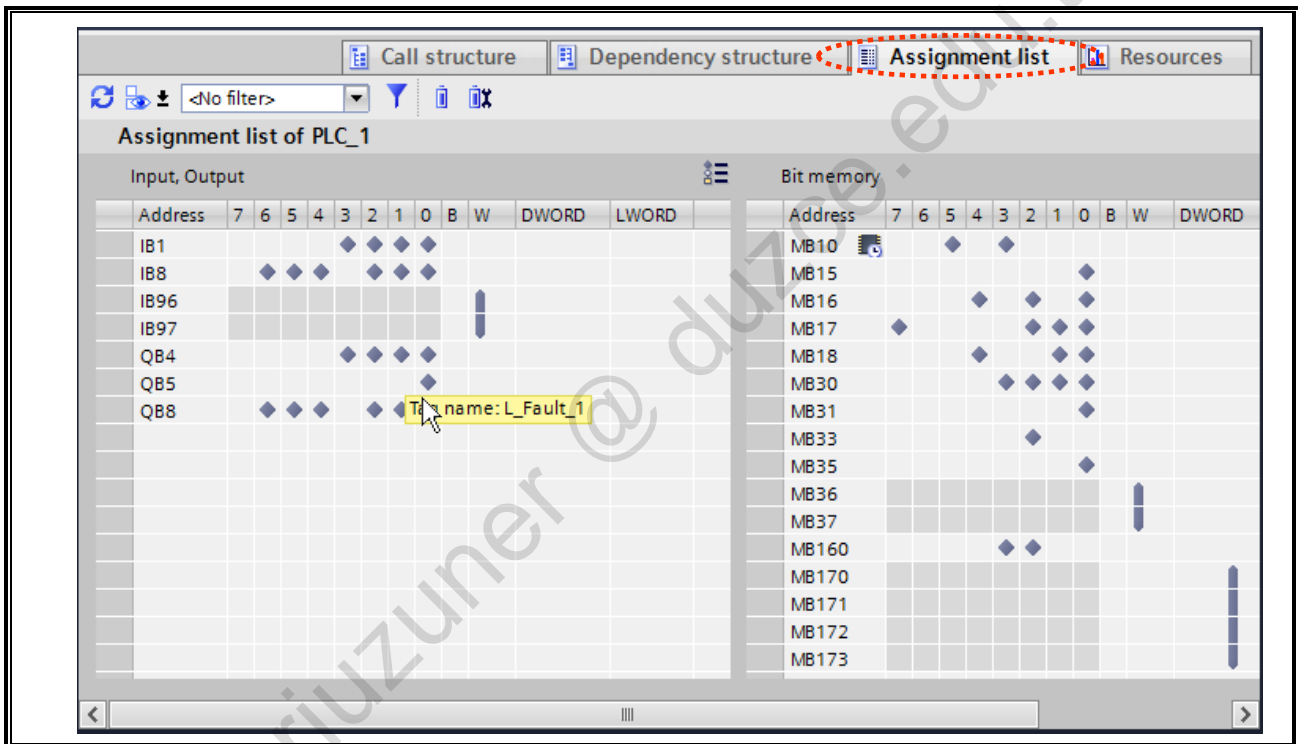
#### Dependency structure

The dependency structure can be shown via menu "tools" and it shows the list of blocks used in the user program. In the first level (to the very left) is the respective block and indented underneath it are the blocks which call this block or use it.

The dependency structure also shows the status of the individual blocks using symbols. Objects which cause a time stamp conflict, and which can lead to an inconsistency in the program are identified with different symbols.

The dependency structure represents an extension of the cross-references list for objects.

### 9.9.5. Reference data: Assignment of I, Q, M



#### Assignment I/Q/M

The assignment list for I/Q/M is opened via "Right-click on the device > Assignment list" or via Menu "Tools > Assignment list".

This assignment list gives you an overview of which bit is used from which byte of the memory areas input (I), output (Q) and bit memory (M) are used. The type of use (reading or writing) is not displayed.

The memory areas inputs (I), outputs (Q) and bit memories (M) are displayed byte-by-byte in lines.

- The bits identified with a small diamond, that is, binary operands (in the picture, for example, I 4.0 or M 16.4) are used explicitly in the program.
- The fields of the individual bits which have a gray background identify byte, word, double-word or long word operands that are used in the user program. The operand dimension (byte, word, double-word or long word) comes from the vertical line in one of the columns "B" (Byte), "W" (Word), "DWORD" (Double word) and "LWORD" (Long word).
- Bits that are marked with both a diamond and a gray background are used explicitly as a binary operand in the user program and are used via a byte, word, double-word or long word operand.

### 9.9.6. Reference data: Resources (memory utilization)

Resources of PLC_1											
Objects	Load memory	Work memory	Retain memory	I/O	DI	DO	AI	AQ			
1	2 %	3 %	0 %		33 %	35 %	17 %	0 %			
2											
3	Total:			4 MB	76800 bytes	10240 bytes	Configured:	30	26	6	3
4	Used:			83515 bytes	2116 bytes	12 bytes	Used:	10	9	1	0
5	Details										
6	▼ OB										
7	OB_Cycle [OB1]	8018 bytes	183 bytes								
8	OB_Cyclic interrupt [OB2...]	4462 bytes	130 bytes								
9	FC	3556 bytes	53 bytes								
10	▼ FB										
11	FB_FaultEvaluation [FB20]	38807 bytes	1221 bytes								
12	TSEND_C [FB1030]	11526 bytes	64 bytes								
13	DB	3363 bytes	64 bytes								
14	Objects for Motion Technology	8163 bytes	0 bytes								
15	▼ Data types										
16	UDT_1	19727 bytes	648 bytes	12 bytes							
17	PLC tags	-	-	0 bytes							

#### Resources (memory utilization)

The Resources is opened via Menu "Tools > Resources" and shows you which (how much) memory area is used by which objects in the CPU.

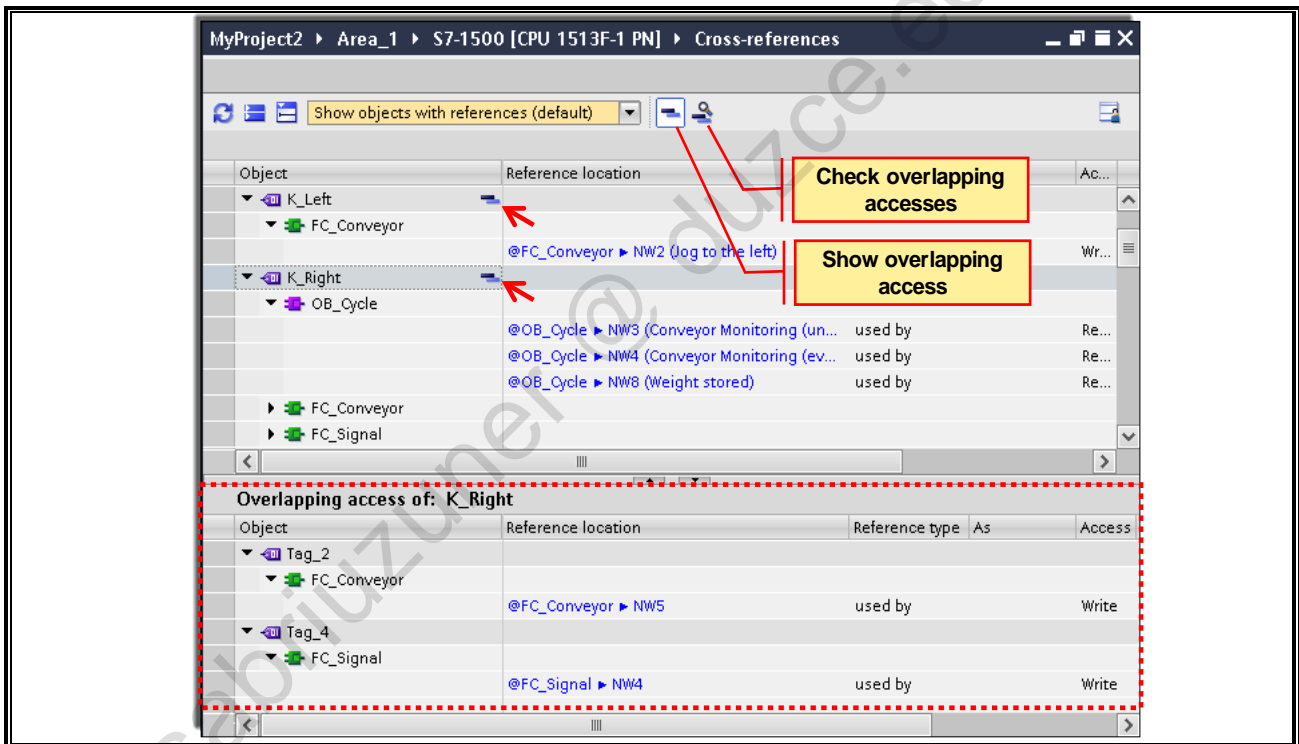
#### Note

Display of the 'Used' Load memory in the CPU:

Please note that the sum of the used load memory cannot be exactly determined if not all blocks have been compiled.

In this case, a ">" placed in front of the sum indicates that the value for the used memory area could be larger than displayed since blocks that are not compiled are not considered for the total formation.

### 9.9.7. Reference data: Overlapping accesses



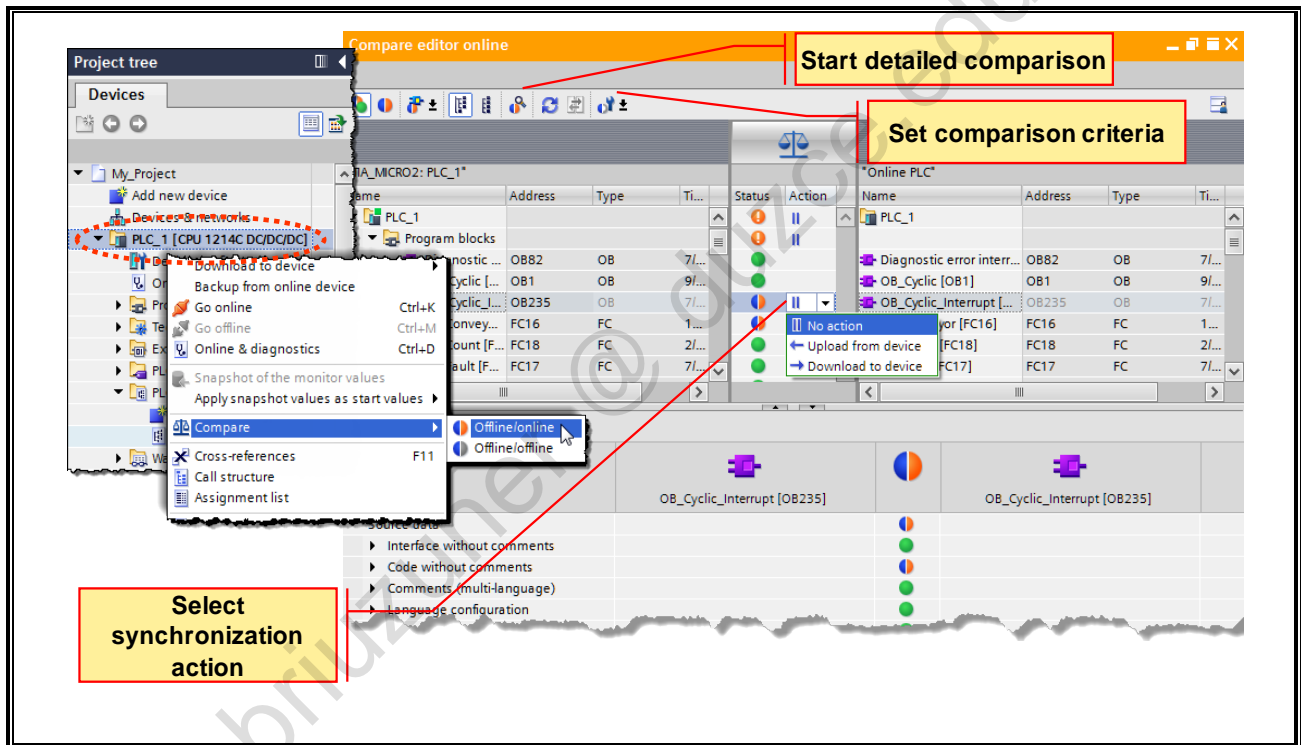
#### Overlapping accesses

With the help of the "Check overlapping accesses" button, you can check whether overlapping accesses exist for one of the variables (tags).

If this is the case, they can be displayed in a separate table with the help of the "Show overlapping accesses" button.



## 9.10. Compare (1) - Offline/online



### Types of comparison

In principle, there are two different types of comparison:

- **Online/Offline comparison:**  
The objects in the project are compared with the objects of the relevant device. For this, an online connection to the device is necessary.
- **Offline/Offline comparison:**  
Either the objects of two devices within a project or from different projects are compared.

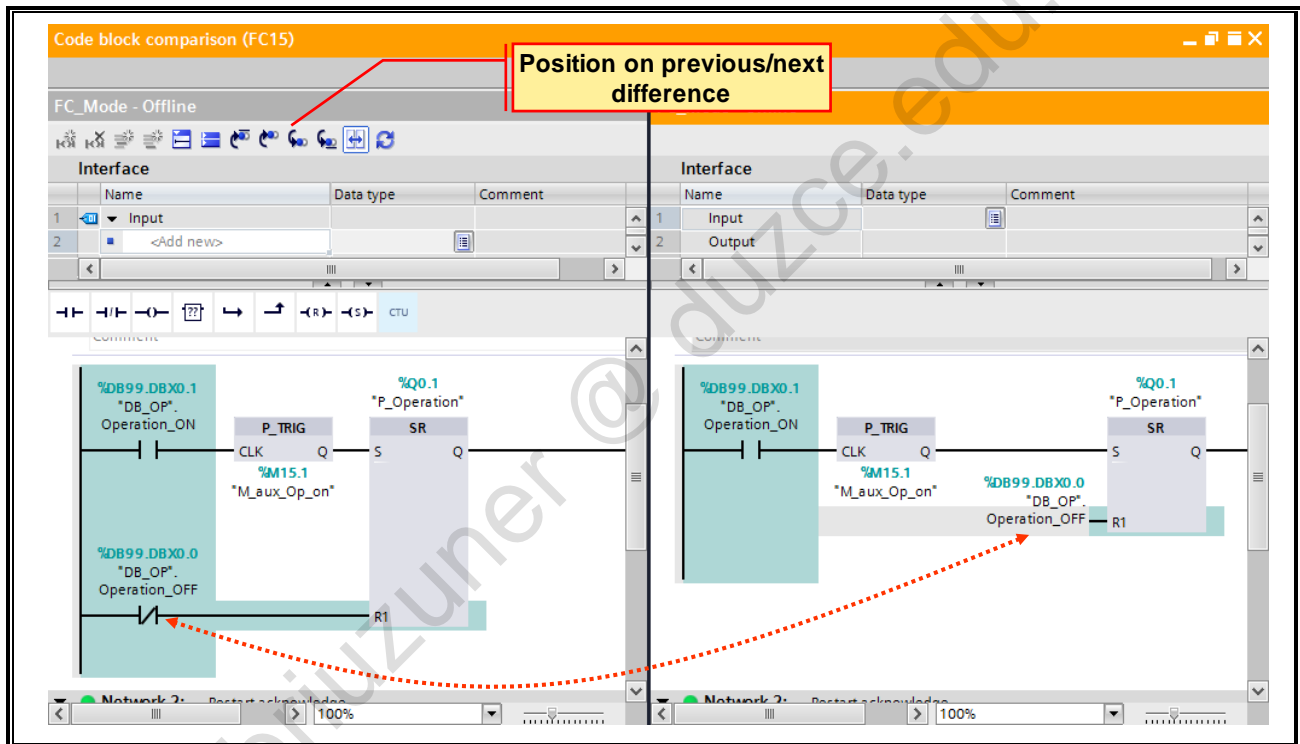
### Symbols of the result display

The result of the comparison is presented by means of symbols.

The following table shows the symbols for the comparison results of an Online/Offline comparison:

Symbol	Meaning
	Folder contains objects whose online and offline versions are different
	Comparison result is unknown
	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

### 9.10.1. Compare (2) - Online/offline block detailed comparison

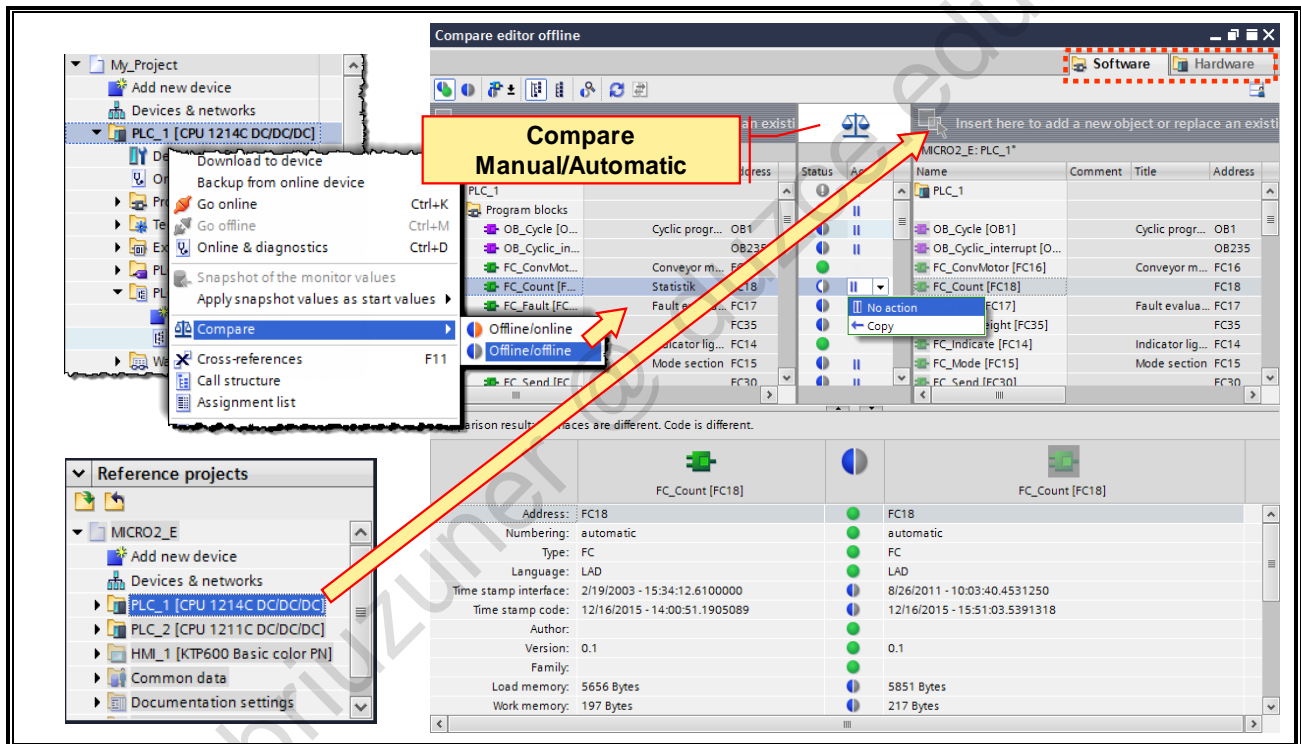


#### Detailed comparison

Through the detailed comparison you can identify exactly those locations that are different in the online and offline version of a block. So that you can find these locations as quickly as possible, the following identifiers are used:

- The lines in which there are differences are highlighted in grey
- The 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

## 9.10.2. Compare (3) – Software offline/offline





### Online/offline software comparison

Compared are either:

- the objects of two devices within a project
- the blocks of two devices within a project
- the blocks within one device
- the objects from different projects
- the blocks from different projects

By means of a mouse-click, you can toggle between

automatic  and manual  comparison.

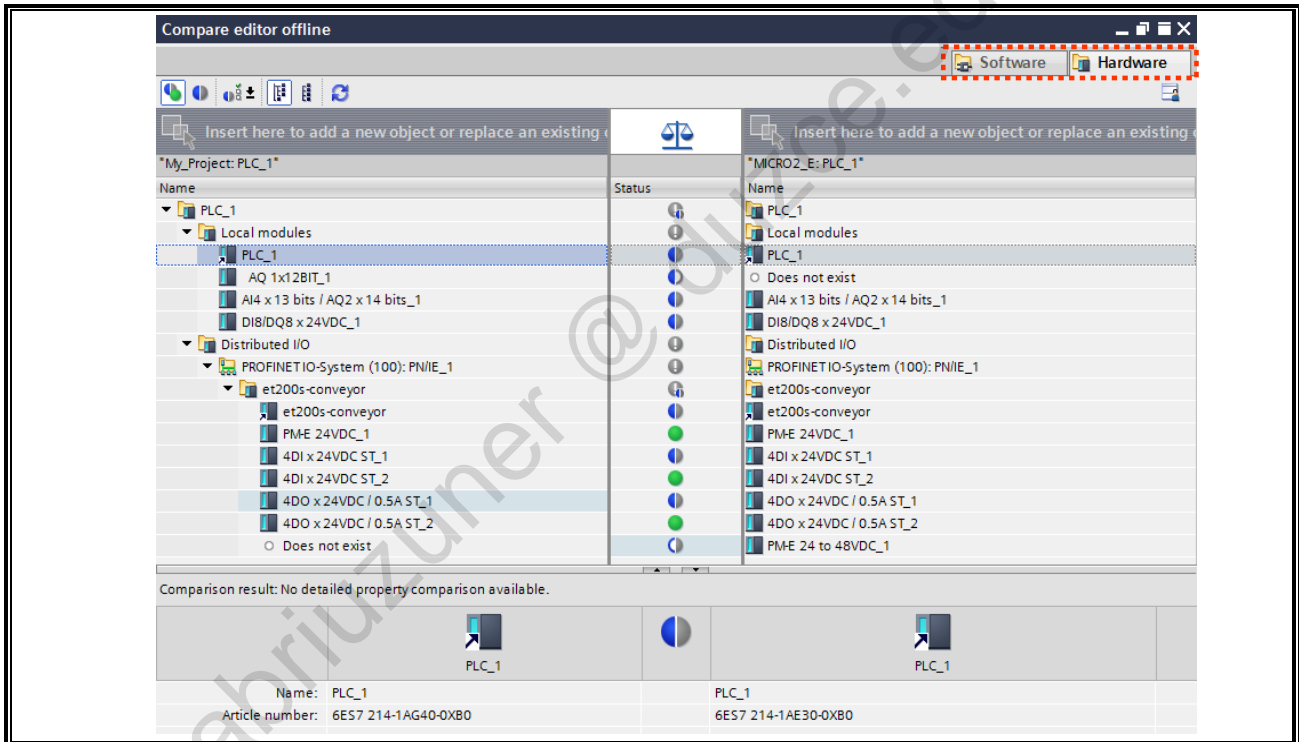
### Automatic Comparison

Blocks and objects of the same type and same name are compared with each other.

### Manual Comparison

You can select which blocks are compared with each other. That way, it is possible to compare all blocks with each other.

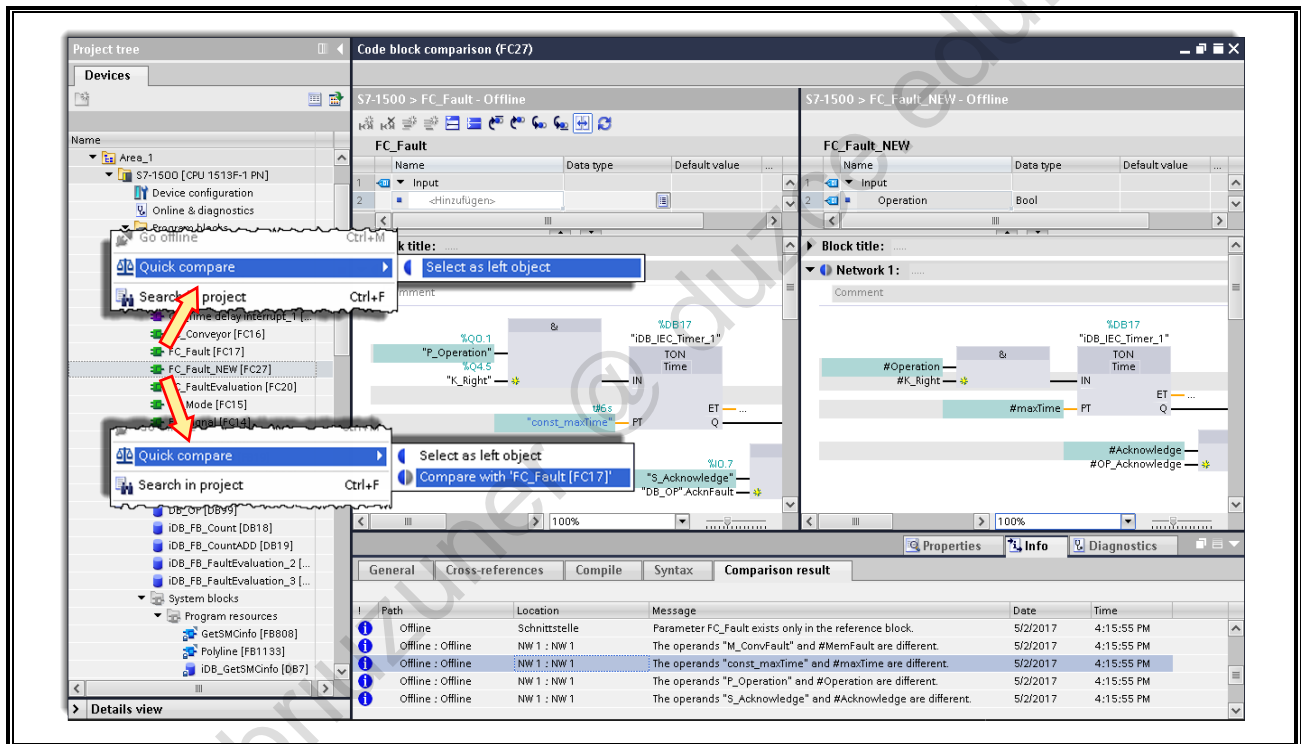
### 9.10.3. Compare (4) - Offline/offline hardware



#### Offline/offline hardware comparison

In addition, it is possible to compare the hardware between two devices or modules in one device with each other.

### 9.10.4. Compare (5) - Block-quick compare



#### Offline/Offline

To start an offline/offline detailed comparison for a block directly in the project tree, follow these steps:

- Right-click the block that you want to compare. This can also be a block from a reference project
- Select the command "Fast comparison > Select as left object" in the shortcut menu
- Right-click the block that you want to compare with the block that you previously selected as left object
- Select the command "Fast comparison > Compare with <selected object>" in the shortcut menu. "<selected object>" stands for the left comparison object

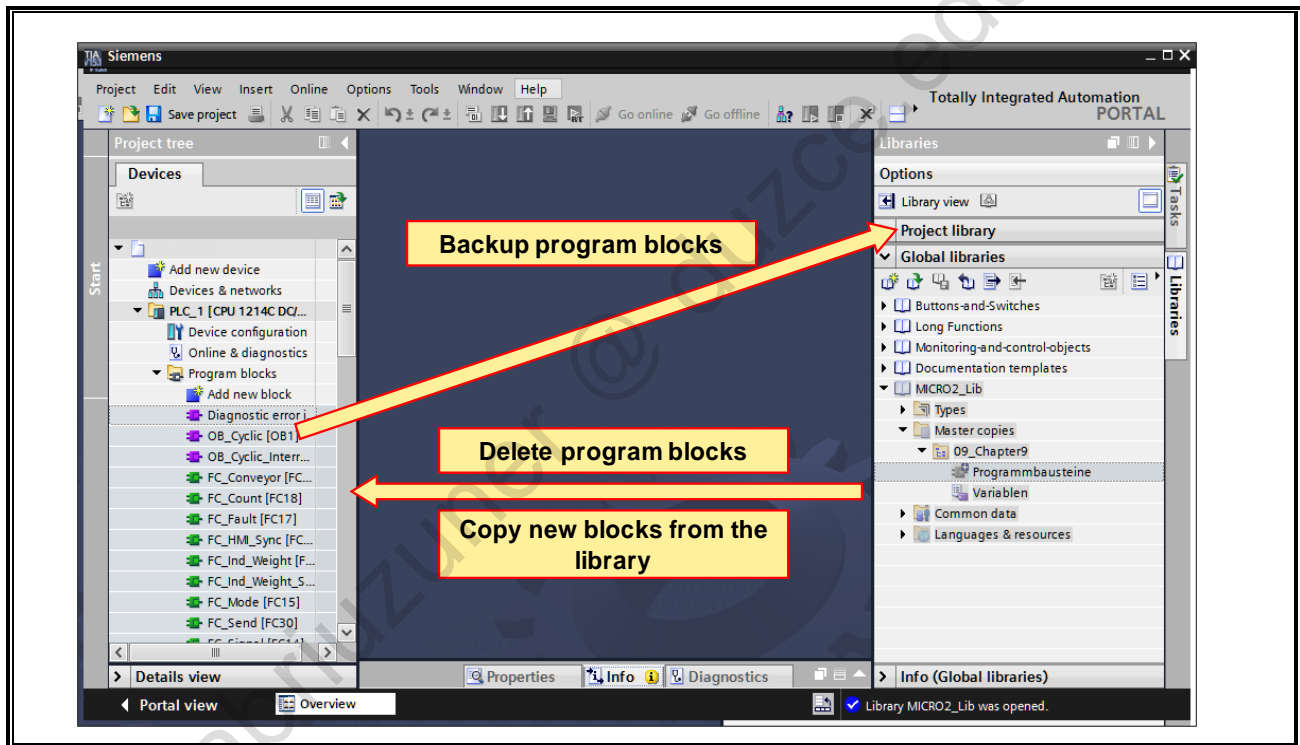
#### Offline/Online

To start an online/offline comparison for a block directly in the project tree, follow these steps:

- Establish an online connection to the device where the block is located
- Right-click the block that you want to compare with its online object
- Select the command "Fast comparison > Compare with the online object" in the shortcut menu



## 9.11. Exercise 1: Downloading a faulty program in PLC\_1



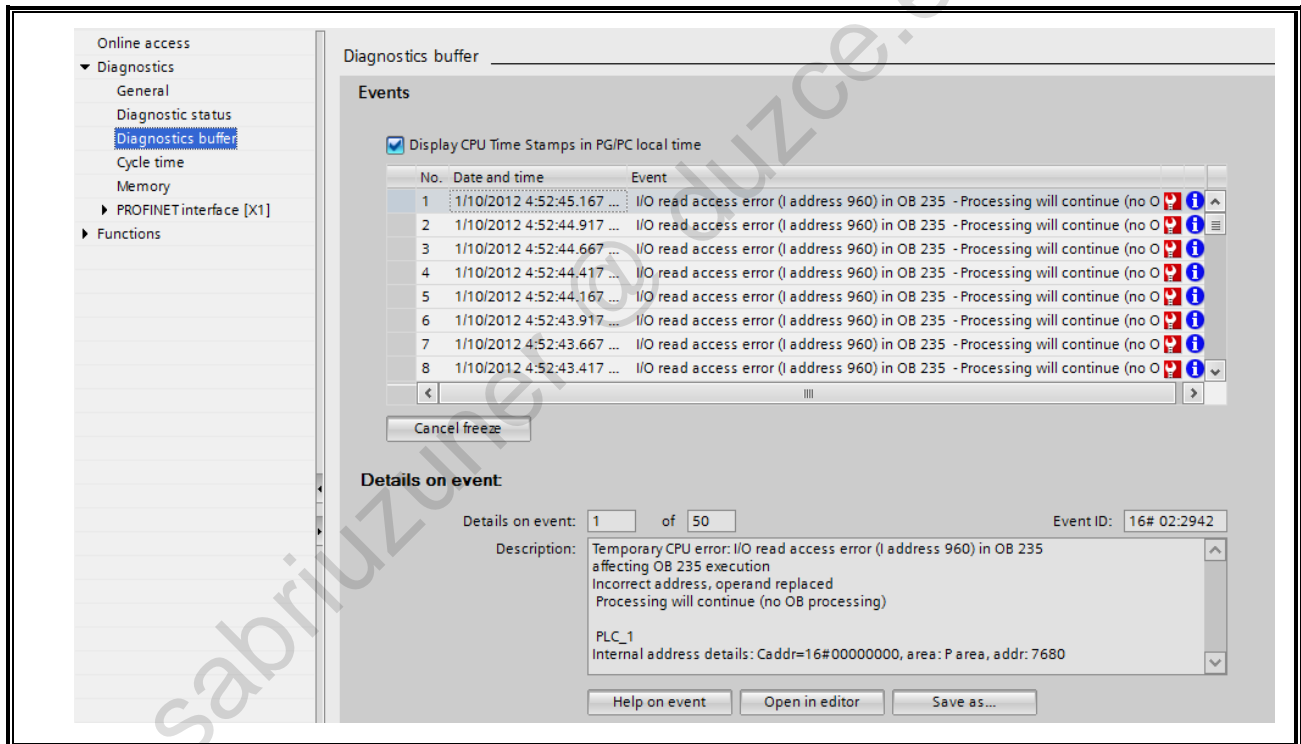
### Task

In the following exercises, you are to work with the program of the library "MICRO2\_Lib". This program fulfills the same tasks as your present project, but it contains errors which you are to look for and eliminate in the following.

### What to do

1. Backup all tag tables and program blocks in the project library, even the system blocks
2. Delete the tags and all blocks from the program blocks folder (even the system blocks)
3. Open the library "MICRO2\_Lib".  
<Drive>:\\_Archive\TIA-MICRO2[version]MICRO2\_Lib.al16
4. Copy the blocks from the folder master copies > chapter\_09 into the program blocks folder of your project and the tag table in the PLC tags
5. Compile the program blocks folder and transfer (download) the software from "PLC\_1" into the CPU 1214C

## 9.12. Exercise 2: Errors detected by the system: Reading out the diagnostics buffer



### Task

By reading out the diagnostics buffer, you are to find out why the ERROR-LED is flashing on the CPU and fix the error.

### What to do

1. Open the "Online & diagnostics" of PLC\_1
2. Select the diagnostics buffer
3. Select (highlight) the last error message and click on "Open in editor"
4. Fix the error in the block and download it into the CPU

### Result

After the error has been eliminated, the ERROR-LED on the CPU goes dark.

### 9.13. Exercise 3: Testing the motor jog

	Name	Address	Display format	Monitor value	Modify value	Corr
1	*P_Operation*	%Q4.0	Bool	<input type="checkbox"/> FALSE		<input type="checkbox"/>
2	*S_Right*	%M30.2	Bool	<input checked="" type="checkbox"/> TRUE		<input type="checkbox"/>
3	*S_Left*	%M30.3	Bool	<input type="checkbox"/> FALSE		<input type="checkbox"/>
4	*M_Jog_Lock_L*	%M150.3	Bool	<input type="checkbox"/> FALSE		<input type="checkbox"/>
5	*B_LB*	%B8.0	Bool	<input checked="" type="checkbox"/> TRUE		<input type="checkbox"/>
6	*K_Right*	%Q8.5	Bool	<input type="checkbox"/> FALSE		<input type="checkbox"/>
7						

#### Task

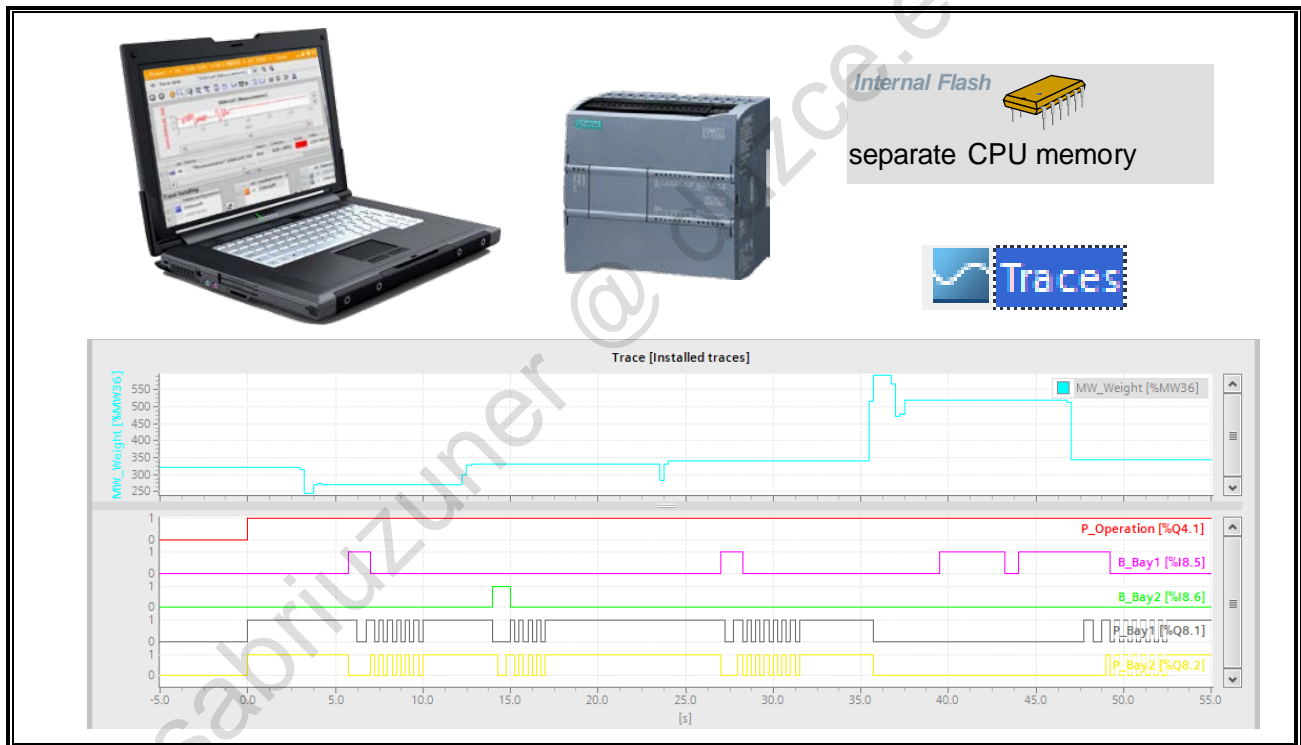
The function "Jog conveyor motor" does not work. The combined use of the PG functions "Monitor block" and "Watch table" (monitor tags [variables]) indicates that there must be a double assignment at output "K\_Right" (Q8.5). The task now is to find all instructions in the entire user program that write-access this output.

#### What to do

1. On the touchpanel, switch off operation
2. Open the "FC\_Conveyor" block and activate the "Monitor" test function
3. In the project tree, under "Watch and force tables" create a new watch table and in it monitor the output "K\_Right" (Q8.5)
4. Display the blocks editor with the opened "FC\_Conveyor" and the watch table one below the other by splitting the working area (see picture)
5. Interpret the different status displays of the two test functions
6. Localize the double assignment at output "K\_Right" (Q8.5) with the help of the reference data and correct the error
7. Download all modified blocks into the CPU and check how the program functions
8. Save your project



## 9.14. TRACE analyzer function



### "Trace" analyzer function

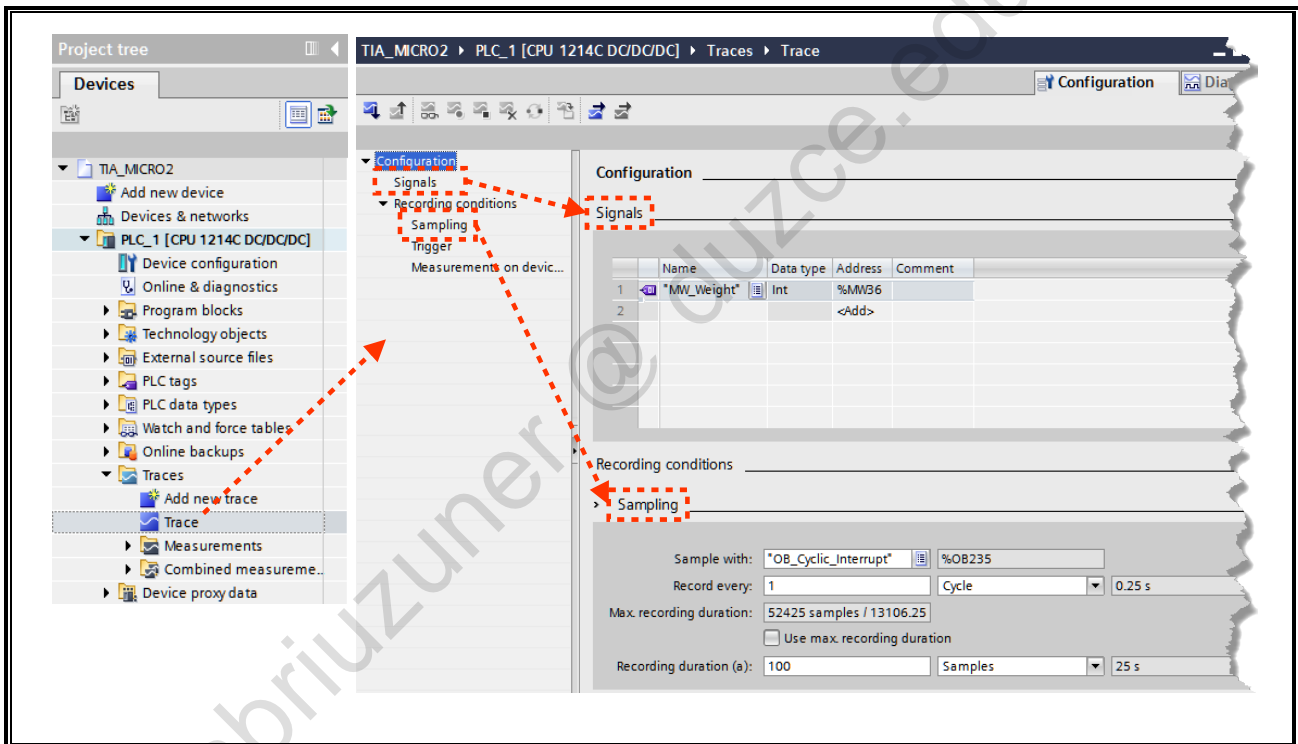
The value-over-time of one or several CPU tags (max. 16) can be stored in a TRACE. In STEP 7, a TRACE recording can be presented graphically.

The number of traces depends on the CPU.

Depending on the CPU, internal TRACE memories with 512 Kbyte each are available.

- S7-1200 2x TRACE (FW  $\geq$  V4.0)
- Up to S7-1517 4x TRACE, S7-1518 8x TRACE
- A maximum of 16 Trace signals or CPU tags (variables) can be recorded per Trace

### 9.14.1. Configuring a TRACE - Signals and sampling



#### Trace – Signals

All global PLC tags (variables) of an elementary data type can be recorded.

#### Trace – Sampling and recording duration

Here, you define how often or in which intervals the Trace signals are to be recorded. From these sampling intervals and the data type or the dimension of the Trace signals you get the maximum duration of a Trace recording since the memory space available for the recording is limited.

Max. memory per trace: 512 Kbytes – 30 bytes (for internal management) = 524,258 bytes

Each sample is saved with a time stamp (8 bytes).

This results in a number of bytes per sample = 8 bytes + number of bytes of a sample  
(for Boolean trace signals, the number of bytes of a measured value is 1 byte)

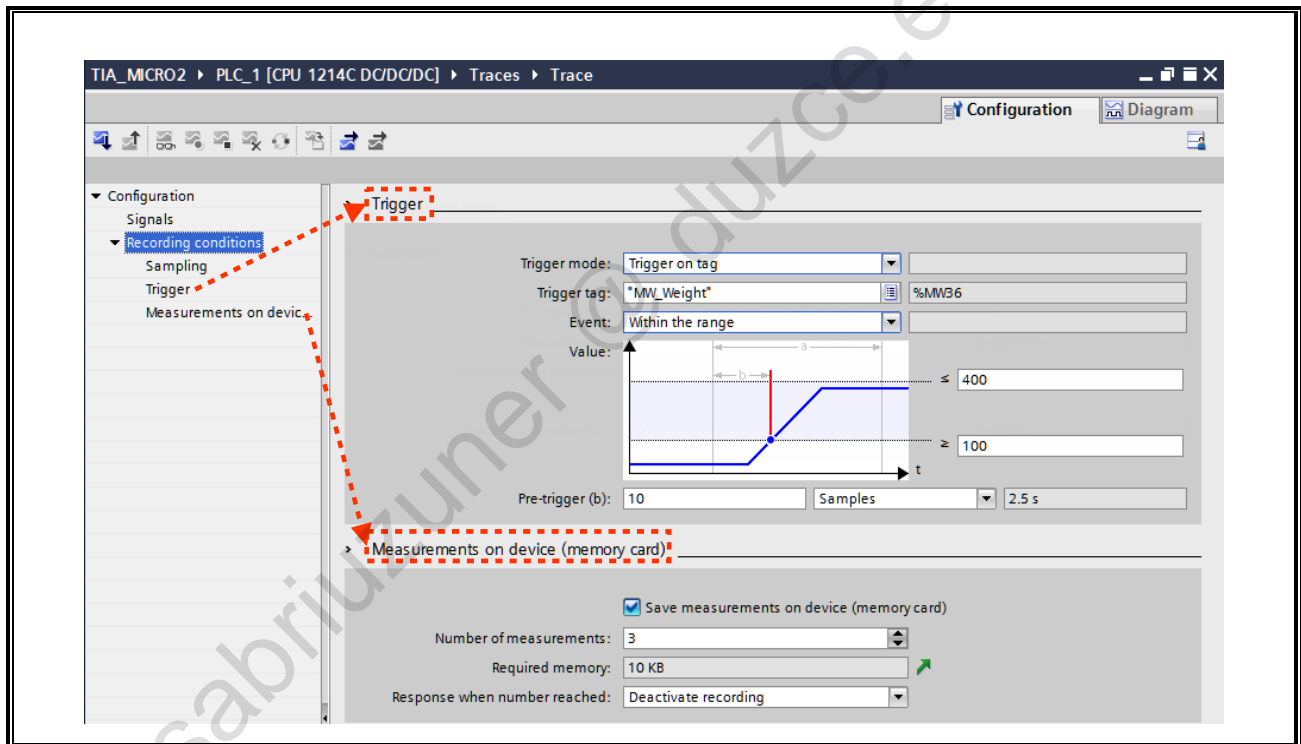
Example: Trace with 1x INT variable and a sampling interval of 100ms

Trace signal of the data type INT -> 8+2 bytes/sample -> 52,425 possible samples

Sampling interval = 100ms -> 10 samples per second

-> maximum recording duration: 52425 samples / 10 samples per second = 5242 seconds

## 9.14.2. Configuring a TRACE – Trigger and saving measurement on device



### Triggering the trace recording

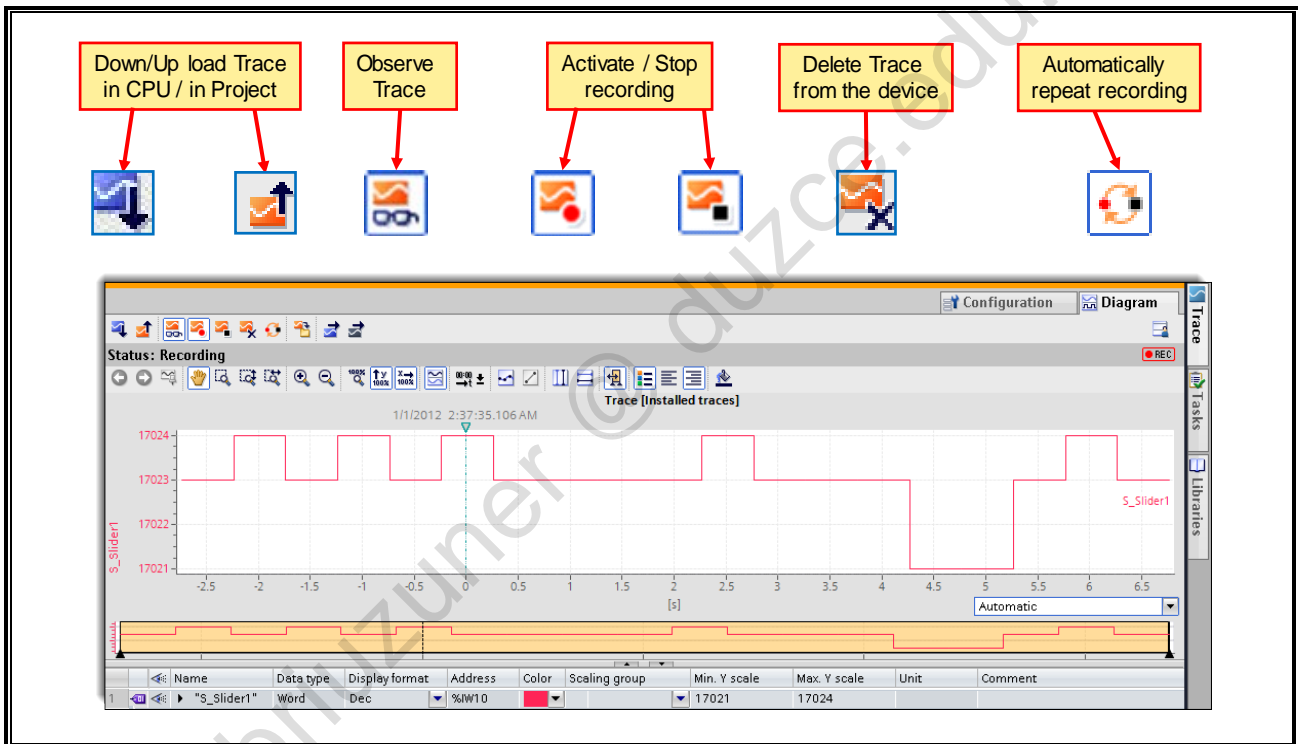
Activating the trace recording starts the measurement and recording of the trace signals, however, not the permanent saving of values since these are merely only temporarily saved in a ring buffer which is continuously overwritten with new values. Only when the configured trigger event is fulfilled, are the temporarily saved values permanently saved and no longer overwritten with new values, whereby the trigger event is dependent on the data type of the trigger variable. The Trace recording ends as soon as the maximum recording duration configured in Trace Sampling is reached.

By defining a pre-trigger, you determine how many of the samples recorded before the trigger event occurs are to remain stored.

### Measurement on device (memory card)

Completed measurements can be stored on the memory card in order to start a new measurement. In the item "Measurements on device", you can define if several and, if yes, how many measurements are to be made. In addition, you define whether the oldest measurement is to be deleted when the set number of measurements is reached or whether no more measurements are to be made. The measurements are stored on the memory card, if there is no memory card, the recording stops after one measurement.

### 9.14.3. Downloading a TRACE configuration into the CPU and activating it



#### Transfer trace configuration to device

#### Add trace configuration from the device to trace configurations

After the trace has been configured offline, i.e., in the project, the configuration must be downloaded into the CPU, since it is not the engineering tool that executed the trace but the CPU.

Only trace configurations that exist online in the CPU can be uploaded from the CPU into the project.

#### Observe trace



“Observe trace” displays the status of a trace on the CPU:

- inactive (Trace already loaded in the CPU, but not yet activated)
- wait for trigger (Trace activated in the CPU, but trigger event not yet fulfilled)
- recording running (Trace activated in the CPU and recording running)
- recording completed (Trace activated in the CPU and recording already completed)

#### Activate recording



/ deactivate recording



The trace is activated with "activate recording", that means that the measuring and recording of the Trace signals is started immediately, even if the possibly configured trigger event is not yet fulfilled. The recorded values are continuously displayed and stored in a ring buffer which is continuously overwritten with new values. Only when the trigger event is fulfilled, are the recorded values no longer overwritten and remain saved. As of this time, the recording is continued until the maximum recording duration is reached.

Through "deactivate recording", a trace with the status "wait for trigger" is deactivated (stopped) or an already running recording is aborted.

### 9.14.4. Evaluating, Saving, Exporting a TRACE in STEP 7

#### View and evaluate trace

When an online connection exists, the trace recording currently saved in the CPU is displayed in the diagram view of the trace editor.

You will find the measurements stored on the memory card in the "Measurements on device" folder.

Trace recordings saved offline in the project can be looked at by double-clicking on the trace recordings saved in the project tree in the "Measurements" folder.

Furthermore, in the "Combined measurements" folder, measurements can be simultaneously evaluated and compared.

#### Save trace in project

With this function, traces saved online in the CPU can be uploaded into the offline project (Add to measurements). A trace can only be saved in the project when it is full, or the recording has been stopped.

#### Trace configuration

This function exports the configuration of a trace which can then be imported into other projects. (TTCRX-file)

#### Trace measurement:

This function exports a trace recording in CSV-format which can then be further processed with MS Excel, for example, or, in TTCRX-format which can be imported into a TIA Portal-project.

### 9.14.5. Trace task card

The screenshot displays the SIMATIC Manager Trace task card interface. The main window shows a graph of a digital signal 'S\_Slider1' over time. The graph has a red signal line and a yellow shaded area. Two vertical dashed lines represent measuring cursors. A red box labeled 'Measuring cursor' points to the right pane, and another red box labeled 'Snapshots' points to the bottom pane. The right pane shows 'Options' for measuring cursors and snapshots. The bottom pane shows a table of snapshots.

Name	Data type	Display format	Address	Color	Scaling group
1	Word	Dec	%W10		
2	Bool	Bin	%I1.0		

#### "Measuring cursor" pane

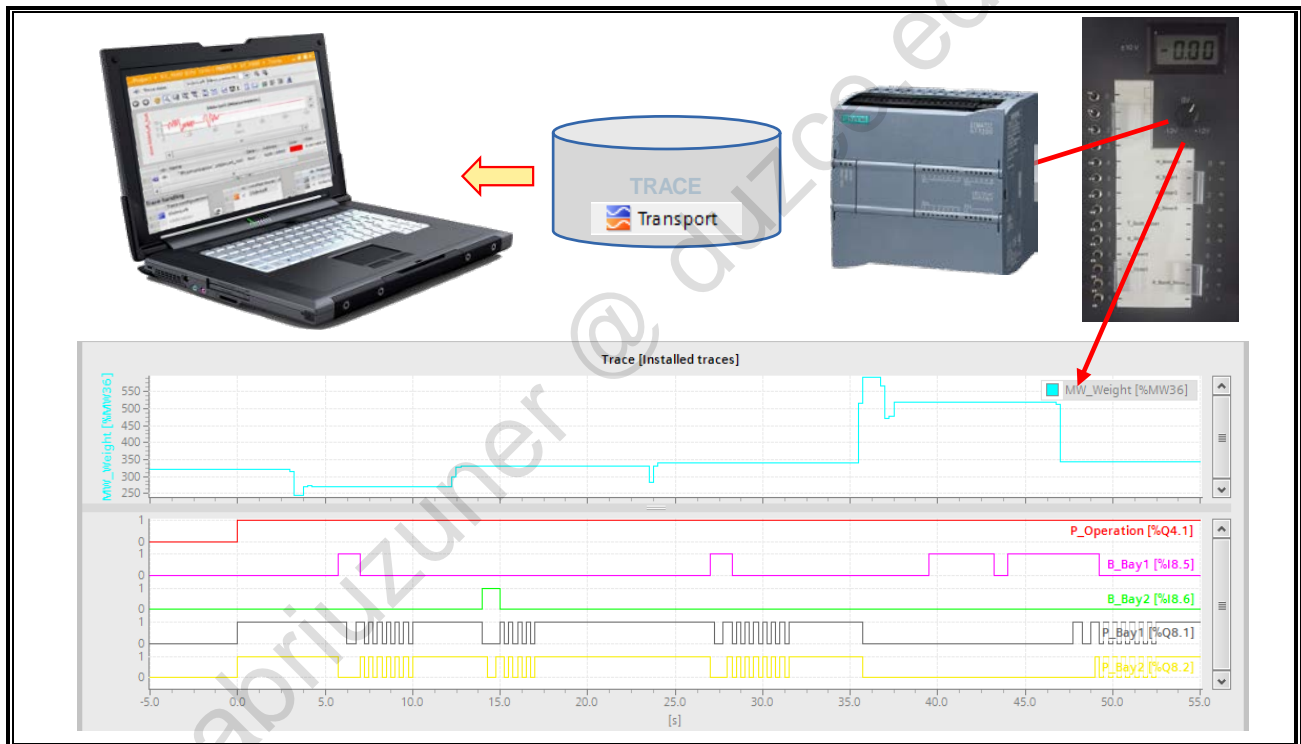
The "Measuring cursor" pane shows the position of the measuring cursor in the graph and the values at the intersections.

#### "Snapshots" pane

The "Snapshots" pane enables the saving and restoring of different views of a measurement.

A snapshot is created from the current view in the "Diagram" tab. The snapshots are saved in the measurement with the project.

### 9.14.6. Additional exercise: Creating, viewing and saving a TRACE



#### Task

When "P\_Operation" (Q4.0) is switched on, the recording of the values of the variables P\_Operation, MW\_Weight, B\_Bay1, B\_Bay2, P\_Bay1, P\_Bay2, S\_Bay1, S\_Bay2, K\_Right and B\_LB is to be started for 60 seconds with a pre-trigger of 5 seconds and then saved in the project.

#### What to do

1. Configure trace "Transport"  
Add a trace, give it the name "Transport", open the editor and switch to the "Configuration" tab
2. Recorded signals: all binary signals of the automatic transport:  
P\_Operation, MW\_Weight, B\_Bay1, B\_Bay2, P\_Bay1, P\_Bay2, S\_Bay1, S\_Bay2, K\_Right and B\_LS
3. The recording clock is to be clocked by the new OB "OB\_Cyclic\_Interrupt". The duration of the trace should be 60s.  
Cyclic time "OB\_Cyclic\_Interrupt" = 250ms = 0.25s  
Recording duration 60s
4. The recording is to be started with the start of the automatic mode with a pre-trigger of 5s
5. Transfer trace to the CPU and activate it  
Since a start trigger was configured in the properties of the trace, recording does not start immediately (status: Wait for trigger) → Recording waits for start of automatic transport
6. Observe trace and start recording  
By connecting online, trace monitoring is already active. The real-time signals are displayed in the diagram
7. Select the "Conveyor" screen in the Touchpanel, activate automatic mode and start several transports to the light barrier
8. In the diagram, you can see the values of the trace variables graphically.  
...wait until the trace is full (recording then stops automatically) or end the trace recording

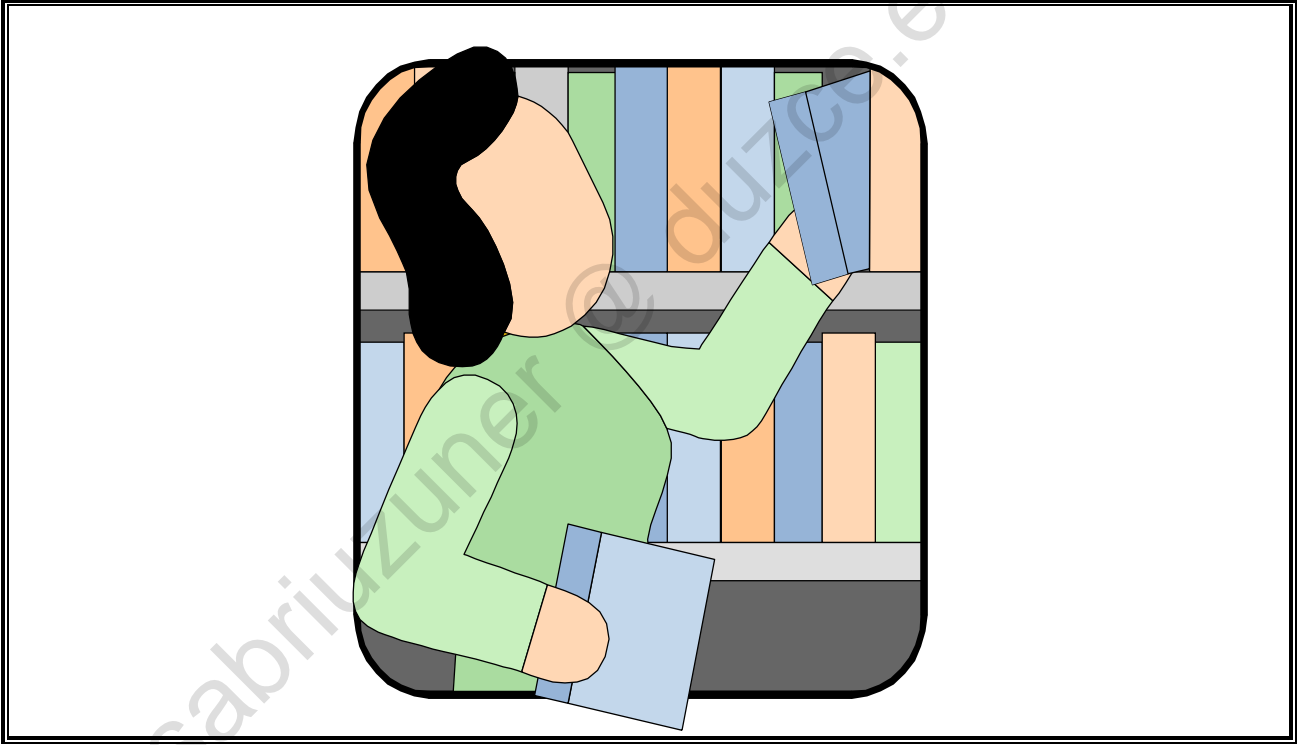
- 9. Save the trace in project**  
If the trace is full or the recording stopped, it can be saved in the project with (Add to measurement)
- 10. View the trace recording**
- 11. Save your project**

sabriuzuner @ duzce.edu.tr

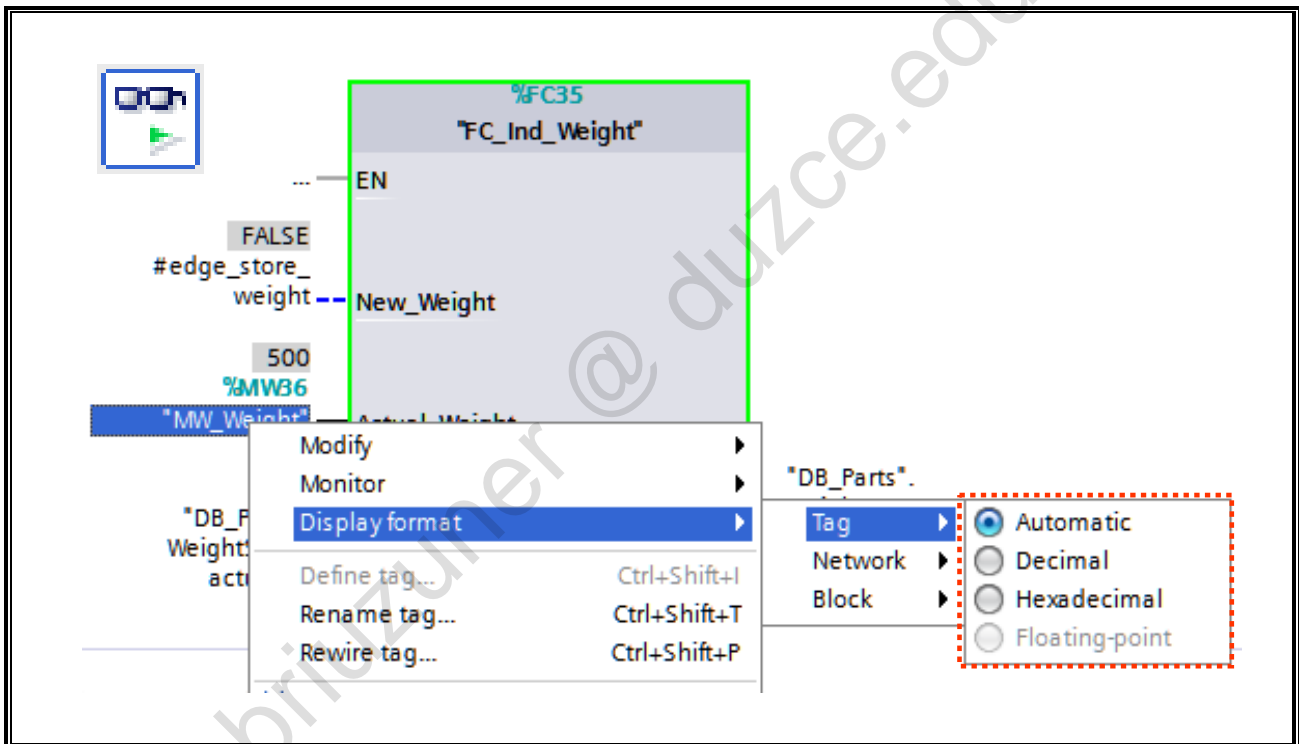
sabriuzuner @ duzce.edu.tr



## 9.15. Additional information



### 9.15.1. Monitor block: Display formats

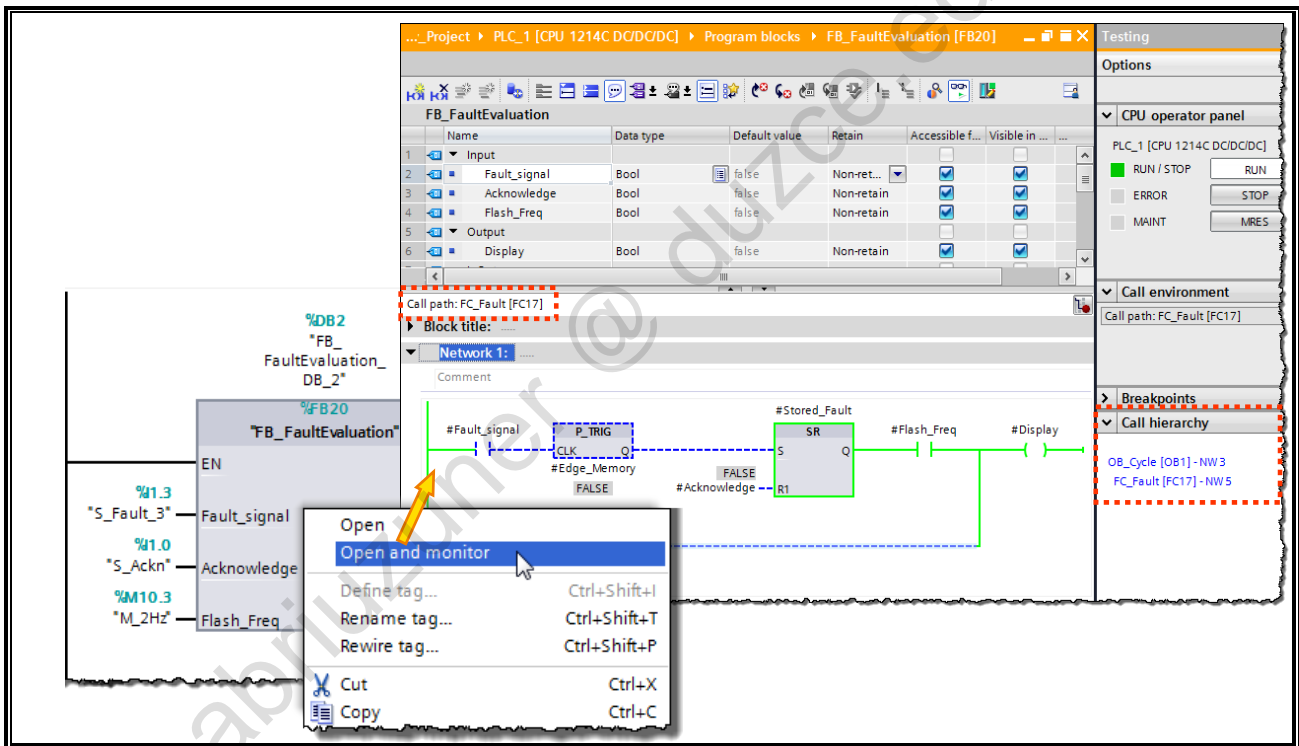


#### Displaying the program status

The display of the program status is updated cyclically and displayed with colored lines.

For the display of numerical values, the display format can be selected between "Automatic", "Decimal", "Hexadecimal" and "Floating point".

### 9.15.2. Monitor block: Call path



#### Monitor with call path

By means of the function "Open and monitor", a block is opened and directly monitored. This determines under which conditions the program status of a block is displayed.

# Contents

<b>10. Programming in SCL.....</b>	<b>10-2</b>
10.1. Objectives .....	10-2
10.2. Task description: Storing weight values in a DB variable.....	10-3
10.3. Program creation in SCL.....	10-4
10.4. Comparison of LAD and SCL.....	10-5
10.4.1. Comparison of different programming languages.....	10-6
10.5. Creating a new SCL block .....	10-7
10.6. Editing an SCL block.....	10-8
10.6.1. Operators .....	10-9
10.6.2. Control structures.....	10-10
10.6.3. Direct addressing (examples) .....	10-11
10.6.4. Indirect addressing (examples).....	10-12
10.6.5. Block calls in SCL .....	10-13
10.6.6. Monitoring an SCL block.....	10-14
10.7. Task description: Commissioning an SCL block and expanding it .....	10-15
10.7.1. Exercise 1: Copying an SCL block from the project library .....	10-16
10.7.2. Exercise 2: Commissioning the SCL block .....	10-17
10.7.3. Exercise 3: Expanding "FC_Ind_Weight_SCL" .....	10-18
10.7.4. Exercise 4: Updating and assigning the block call.....	10-19

## 10. Programming in SCL

### 10.1. Objectives

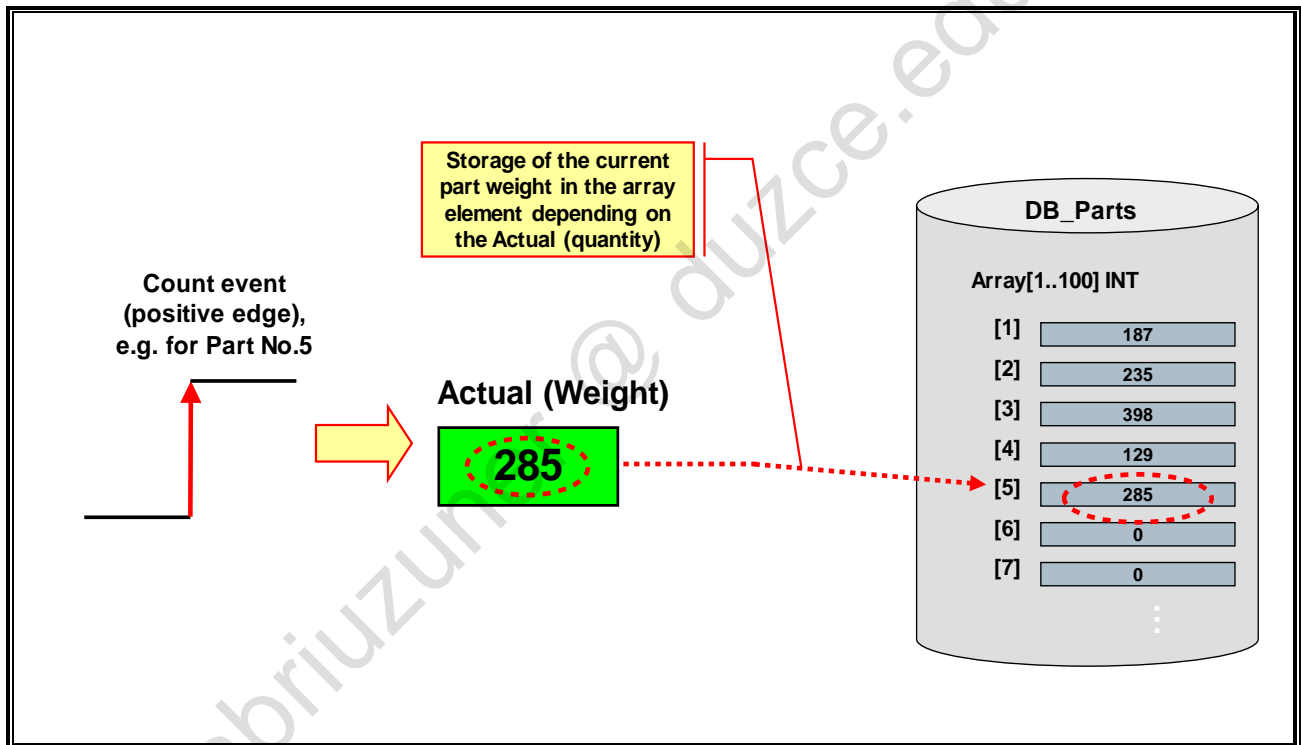
**At the end of the chapter the participant will ...**

- ... be familiar with the application areas of SCL
- ... be able to commission a given SCL block and expand it

#### Objectives

The programming language SCL is presented in this chapter.

## 10.2. Task description: Storing weight values in a DB variable



### Situation up until now

The weight values of transported parts are stored in the array variable "DB\_Parts".PartWeights in "FC\_Ind\_Weight" via indirect addressing using "FieldWrite".

Old weight values are overwritten with new ones when the ring buffer ("DB\_Parts".PartWeights) is rewritten.

### Task description

In the first step, the function "FC\_Ind\_Weight" and its call in "FC\_Count" is to be replaced by the function "FC\_Ind\_Weight\_SCL". This is copied out of the Project library and fulfills the same function.

Subsequently, the initializing of the DB variable "DB\_Parts".PartWeights is to be programmed: As soon as the setpoint quantity equals the actual quantity and the bay pushbutton at the light barrier bay is pressed, all array elements in "DB\_Parts".PartWeights are to be overwritten with the value "0".

## 10.3. Program creation in SCL

### Structured Control Language as text-based structured high-level language

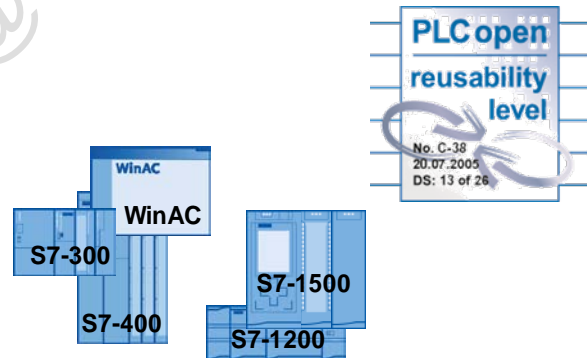
```

4 // Example FOR
5 FOR #index := 0 TO "Border" DO
6 // Statement section
7 #counter:=#counter + #index;

8 "Speed" := #counter;
9 END_FOR;
10 // Example IF
11 IF...
12 // Example CASE
16 CASE #trigger OF
17 1: // Statement section case 1
18 #trigger := 2;
19 2..4: // Statement section case 2 to 4
20 #trigger:= #trigger + 1 ;
21 ELSE
22 // Statement section ELSE
23 #trigger:=0;
24 END_CASE;
25 "Speed_Control" := #trigger;
26 (*...
28 IF #trigger = 0 THEN
29 // Statement section IF
30 #trigger := 1;
31 END_IF;

```

- Structured text
- Complex calculations & algorithms
- Database operations, for example, sorting data
- Program code exchangeable between S7-1200 and S7-1500 as well as between S7-300, S7-400 and WinAC



### S7-SCL

SCL (Structured Control Language) is a PASCAL-similar high-level textual language. It simplifies the programming of mathematical algorithms and complex data processing tasks for PLCs. SCL therefore also enables S7 PLCs to be used for more complex tasks such as closed-loop control or statistical evaluation.

SCL offers the functional scope of a high-level language such as:

- Loops
- Alternatives
- Branch distributors, etc.

combined with PLC-specific functions such as:

- Bit accesses to the I/O, bit memories, timers, counters etc.
- Access to the symbol table
- STEP 7 block accesses

### Advantages of SCL

- easy to learn programming language, especially for beginners
- easy to read (understand) programs are generated
- simpler programming of complex algorithms and processing of complex data structures
- System integration in STEP7 languages such as FBD and LAD
- relatively easy for PLC technicians to understand by reference to STEP7 languages

## 10.4. Comparison of LAD and SCL

The image shows a comparison between Ladder Logic (LAD) and Structured Control Language (SCL) for the same task. The LAD solution on the left consists of six networks:

- Network 1:** A normally open contact labeled #Request is connected to a coil labeled end (JMP).
- Network 2:** A coil labeled FieldWrite (Real) is connected to EN. The ENO output is connected to a coil labeled MEMBER (Store[1]). The IN/OUT is connected to #index (INDEX) and #Data (VALUE).
- Network 3:** A normally open contact labeled #index (Dint) with a value of 10 is connected to a coil labeled reset (JMP).
- Network 4:** A coil labeled INC (Dint) is connected to EN. The ENO output is connected to #index (IN/OUT).
- Network 5:** A coil labeled end (RET) is connected to EN.
- Network 6:** A coil labeled MOVE is connected to EN. The ENO output is connected to #index (OUT). The IN is connected to 1.

The SCL solution on the right is a compact 7-line code block:

```

1 IF #Request THEN
2   #Store[#Index]:= #Data;
3   IF #Index < 10 THEN
4     #Index:= #Index+1;
5     ELSE #Index:=1;
6   END_IF;
7 END_IF;

```

Red boxes labeled "Solution in LAD" and "Solution in SCL" are connected to their respective parts of the diagram.

### Comparison

The use of SCL is recommended for data processing and the programming of complex, mathematical functions. In the example shown, data is stored in an array variable using indirect addressing. It is obvious that the program code of the SCL solution is significantly more compact than the code in LAD or FBD.



### 10.4.1. Comparison of different programming languages

**Solution in STL for S7-300/400**

```

1 L #WeightStore.Act_No
2 + 1
3 T #WeightStore.Act_No
4 T #part_no
5
6 L P##WeightStore
7 LAR1
8 L W [ AR1 , P#0.0 ]
9 T #DBNo
10 OPN DB [ #DBNo]
11 L D [ AR1 , P#2.0 ]
12 LAR1
13
14 L #part_no
15 + -1
16 L P#2.0
17 *D
18 L P#4.0
19 +D
20 TAR1
21 +D
22 T #point
23 L #Weight
24 T DBW [ #point]

```

**Solution in SCL for S7-300/400 AND S7-1200/1500**

```

24 #WeightStore.Act_No:=#WeightStore.Act_No+1;
25 #part_no:=#WeightStore.Act_No;
26 #WeightStore.Part_Weight[#part_no]:= #Weight;

```

**Solution in STL for S7-1500**

```

1 L #WeightStore.Act_No
2 + 1
3 T #WeightStore.Act_No
4 T #part_no
5
6 L #Weight
7 T #WeightStore.Part_Weight[#part_no]

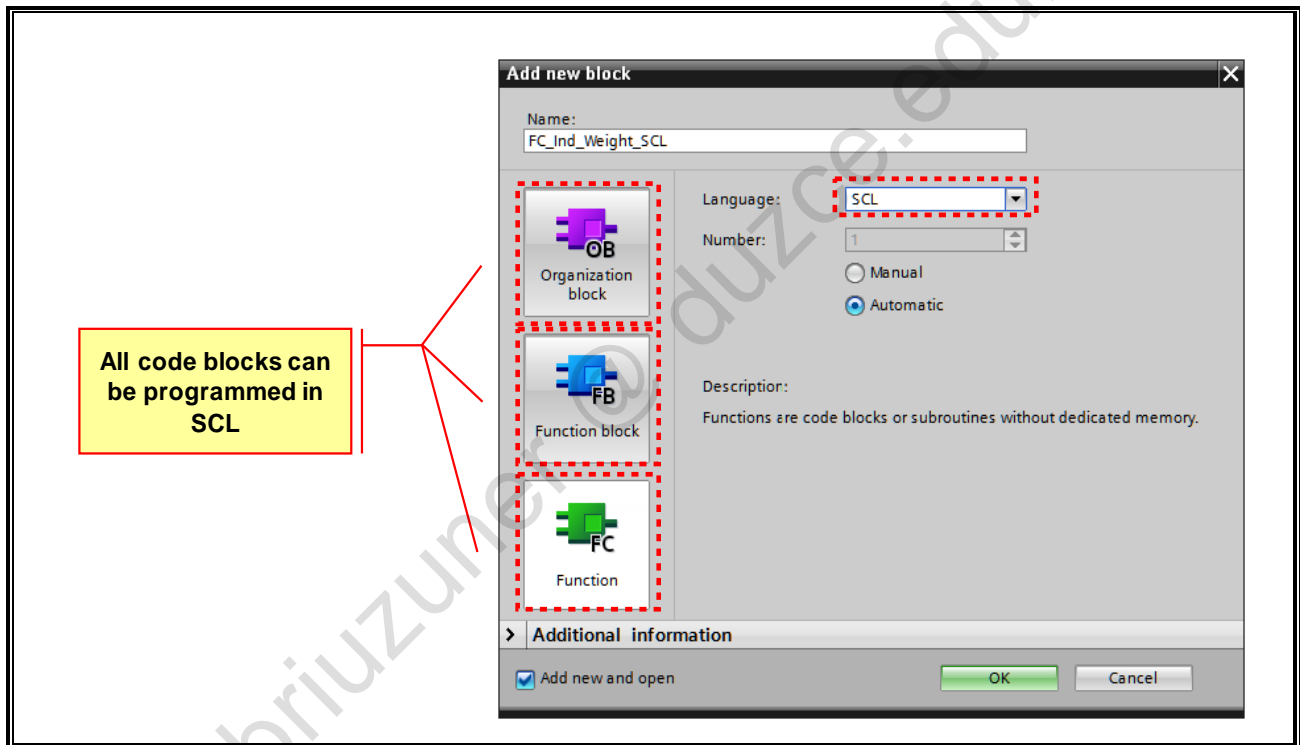
```

**Solution in FBD for S7-1200/1500**

#### Comparison

Compared to other programming languages, the programming is simple, compact and easy to understand.

## 10.5. Creating a new SCL block



### Creating an SCL block

SCL blocks are inserted in the same manner as blocks in the programming languages LAD/FBD/STL. Depending on the task, SCL blocks can be of the FC or FB block type.

Even the internal structure in the declaration and statement section is identical to the LAD/FBD/STL blocks:

- **Declaration Section:**  
The IN, OUT and INOUT parameters as well as the local temporary and local static variables of the block are declared in the declaration section of a block.
- **Statement Section:**  
The statement section contains the instructions that are executed after a logic block (OB, FB, FC) is called. These instructions are used to process data and operands.

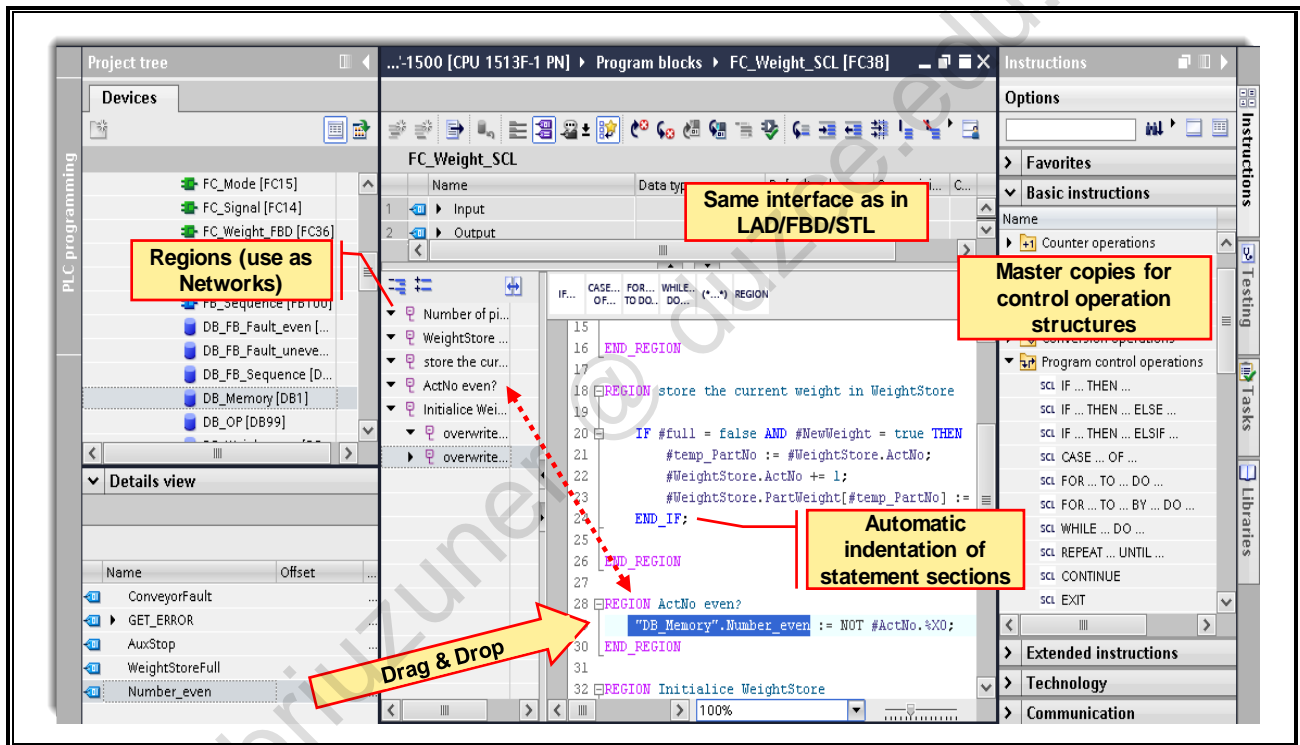
In SCL blocks, the interface can also be shown as textual interface. For this the standard for new blocks must be changed from table view to textual view in the settings (Options > Settings > PLC programming > SCL).

### Note

In programming instructions, the following points must be taken into consideration:

- Each instruction must be completed with a semicolon.
- All identifiers (names) used in the statement section must be declared.

## 10.6. Editing an SCL block



### Statement Section

The statement section contains the instructions that are executed after a code (logic) block (OB, FB, FC) is called. These instructions are used to process data and operands.

#### Subdivision

The individual instructions can basically be divided into three groups:

- Value assignments:
  - They are used to assign an expression or a value to a variable.
- Control instructions:
  - They are used to branch within a program or to repeat groups of instructions.
- Subroutine call:
  - They are used to process functions and function blocks.

### Note

In programming instructions, the following points must be taken into consideration:

- Each instruction must be completed with a semicolon.
- All identifiers (names) used in the statement section must be declared.

### Using Regions as Networks:

- Code structuring
- Greater clarity & readability
- Easy navigation even in large blocks
- In addition, there is a synchronized navigation column including the display of syntax errors..

### 10.6.1. Operators

Syntax: result := expression;		
Example: #Q_Average := (#Value1 + Value2) / 2;		
Logic operation	Description	Operator
Assignment	Assignment	:=
Parenthesis	(Expression)	(.)
Binary logic operation	Negation AND OR Exclusive-OR	NOT AND, & OR XOR
Comparison	Less than, less than or equal to, greater than, greater than or equal to, equal to, not equal to	<, <=, >, >= =, <>
Math	Plus, Minus (sign) Addition, Subtraction Multiplication, Division Exponentiation	+,- +,- *, /, MOD **

#### Expressions

Expressions consist of operands, operators and round brackets (parenthesis). Within an expression the operators (e.g. +, -, \*, /, etc.), that is, the active components of an expression, are linked with the passive elements, such as constants, variables and function values, in order to form a new value. An expression therefore stands for the value it represents. SCL permits the formation of standard expressions, that is, mathematical, logical and comparative expressions. Variables from data blocks, arrays, structures and CPU memory areas (inputs, outputs, etc.) can be enlisted for the formation.

#### Operators and Operands

Expressions consist of operators and operands. Most SCL operators link two operands (e.g. A + B) and are therefore termed binary operators. The others work with only one operand and are thus called unary operators.

The result of an expression can

- be assigned to a variable (e.g. A := B + C;)
- be used as a condition for a control statement (e.g. IF A < B DO ... )
- be used as an actual parameter for the call of a function or a function block (e.g. FB20 (Input := A + B))

#### Value Assignments

With their help, variables can be assigned new values. The old variable value is lost.

## 10.6.2. Control structures

There are master copies (so-called code snippets) for the control operation structures!		
	Keyword	Functionality
Program branching	IF	Program branching with Boolean value
	CASE	Program branching with INT value
Program loops (Abort possible)	FOR	Program loop with run variable
	WHILE	Program loop with execution condition
	REPEAT	Program loop with abort condition
Loop aborts	CONTINUE	Abort current loop pass
	EXIT	Exit program loop
Block abort	RETURN	Exit the block

### Control instructions

These are used for changing the sequence in which the instructions are normally processed.

A choice from the various alternatives in the program execution can be made with conditional instructions (IF and CASE instructions).

Loop instructions (FOR, WHILE and REPEAT instructions) are used to repeatedly execute instructions.

Jump instructions (CONTINUE, EXIT and GOTO instructions) permit the sequence of processing to be interrupted and to jump to a resumption point.

FB and FC Calls:

According to the principle of structured programming, other function blocks and functions can also be called from an SCL block.

Callable blocks are:

- Functions and function blocks that were created in SCL or in another STEP 7 language (STL, LAD, etc.)
- Standard functions and standard function blocks that are supplied with SCL
- Instructions with and without instance

### 10.6.3. Direct addressing (examples)

Direct addressing (absolute and symbolic) for inputs, outputs and memory bits is identical to LAD/FBD/STL!				
	Area	Notation examples		Examples
			SCL V5.x	TIA SCL (harmonized to STL...)
Absolute Addressing	Bit	%DBz.DXy.x, %Iy.x	%DBz.DBXy.x, %Iy.x	%DB5.DBX0.7, %I0.0
	Byte	%DBz.DBy, %MWy	%DBz.DBBy, %MWy	%DB5.DBB2, %QB2
	Word	%DBz.DWy, %MWy	%DBz.DBWy, %MWy	%DB5.DBW4, %MW20
	Double-word	%DBz.DDy	%DBz.DBDy	%DB5.DBD8
Symbolic Addressing		<DB-Symbol>.<Variable-Name>		"Motor".Setvalue
	Only for S7- 1200/1500	<Variable-Name>.%X<Bitnumber> <Variable-Name>.%B<Bytenumber> <Variable-Name>.%W<Wordnumber>		"Motor".Alarms.%X1 (Bit 1 of the variable "Alarms" in DB "Motor")

#### Absolute addressing

The instruction part of a block describes the actions to be executed with local operands (parameters or local variables) or global variables (PLC tag or variable in a global data block). The global variables can be addressed absolutely or symbolically.

The absolute addressing for SCL corresponds to that for the basic languages (LAD and FBD). Access to global data operands in SCL is only completely addressed. Inputs and outputs are automatically given a symbolic name (e.g. "Tag\_4") after input.



Note, that for SCL, there can be no separator (space or tabulator) between the operand notation and the operand address.

#### Symbolic addressing

Symbolic addressing addresses operands and variables with a name. The name is assigned in the symbol table for global data; for local data it is assigned in the declaration section of the block. The symbolic addressing for SCL corresponds to that of the basic languages.

### 10.6.4. Indirect addressing (examples)

Indirect addressing, both for I/Q/M/Peripheral access, and for variables in data blocks!			
	Area	Instruction	Examples
Indirect Addressing	Bit	PEEK_BOOL POKE_BOOL	#OUT1 :=PEEK_BOOL(area:=#Memory_area, dbNumber:=#DB_Number, byteOffset:=#Byte_addr, bitOffset:=Bit_addr);  POKE(area:= Memory_area, dbNumber:=#DB_Number, byteOffset:=#Byte_addr, value:=w#16#12);
	BYTE - LWORD	PEEK POKE	
	WORD	PEEK_WORD	
	DWORD	PEEK_DWORD	
	LWORD	PEEK_LWORD	
	Area	POKE_BLK	
Indexed Access	Array element	"<DB-Symbol>".<Array-Symbol>[Index]	"Motor".Value[#Index]

#### Indirect addressing

The instructions PEEK (reading) and POKE (writing) are used for indirect addressing.

Memory\_area, DB\_Number, Byte\_addr and Bit\_addr are constants or, at runtime, changeable variables or expressions.

The following operand areas (area) can be addressed in this way:

- 16#81: Input
- 16#82: Output
- 16#83: Memory bit
- 16#84: DB
- 16#1: Peripheral input (only S7-1500)

If the memory area is not 16#84, then the DB\_Number must be specified with 0.

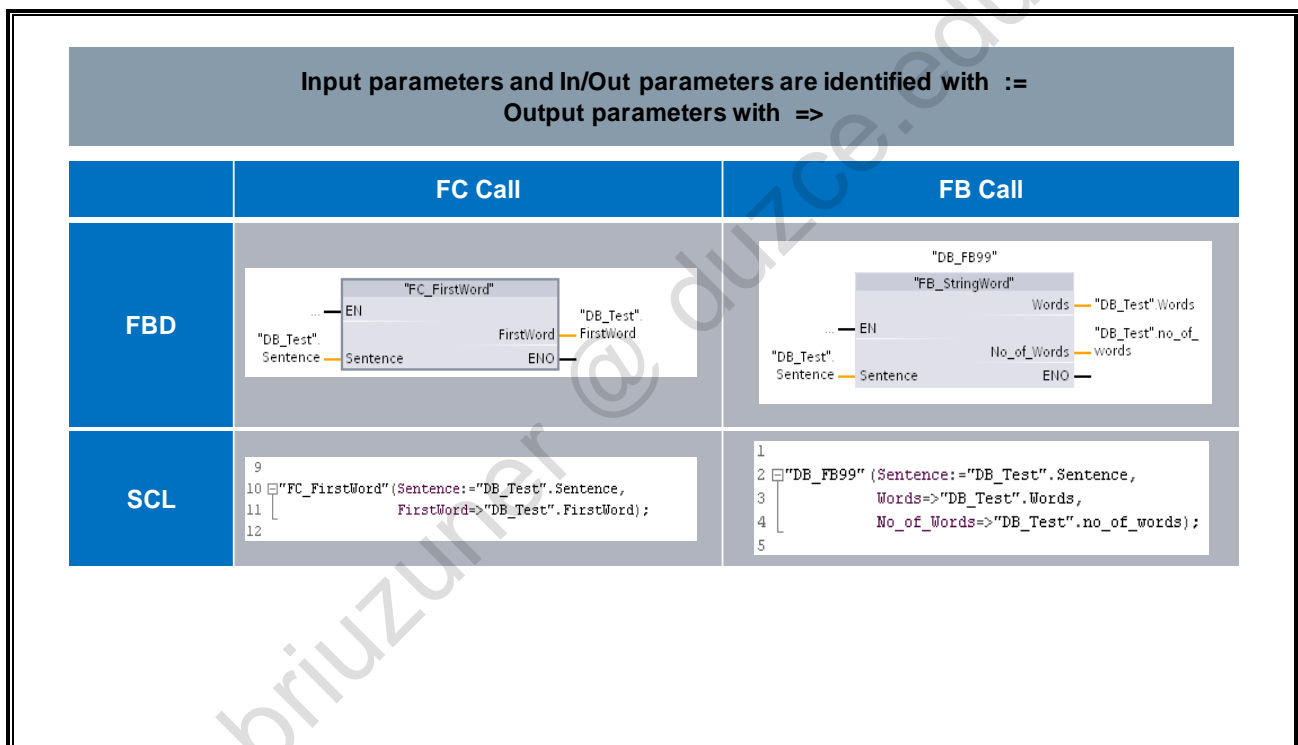
#### Example Poke\_BLK

```
POKE_BLK(AREA_SRC := "Tag_Source_Area",           //Memory area Source
         DBNUMBER_SRC := "Tag_Source_DBNumber",   //DB-Number Source
         BYTEOFFSET_SRC := "Tag_Source_Byte"),     //Byte number Source
         AREA_DEST := "Tag_Destination_Area",      // Memory area Destination
         DBNUMBER_DEST := "Tag_Destination_DBNumber", //DB-Number Destination
         BYTEOFFSET_DEST := "Tag_Destination_Byte", //Byte number Destination
         COUNT := "Tag_Count");                   //Number of bytes
```

#### Array Elements

Array elements can be addressed with a variable Index at runtime.

## 10.6.5. Block calls in SCL



## Calling an SCL block

Depending on the task, SCL blocks can be created as FC or FB. According to the principle of structured programming, other function blocks, functions as well as instructions with and without instance can also be called from an SCL block. In that way, other functions and function blocks that were created in SCL or in another STEP 7 language (STL, LAD, etc.) can be called.

Callable blocks are:

- Other functions and function blocks that were created in SCL or in another STEP7 language (FBD, LD, and so on)
- Simple and advanced instructions
- technology objects
- communication blocks

## Display of an FB call

For an FB call, only the instance is displayed. If you hover the mouse pointer over the instance, a tooltip displays the relevant FB.

## Note

When a function which supplies a return value (RET\_VAL) is called, this return value must be stored in an operand

<Operand>:=<Function name> (Parameter list);



### 10.6.6. Monitoring an SCL block

The screenshot displays the SCL code on the left and the monitoring interface on the right. The code includes comments and instructions for monitoring a weight store. The monitoring interface shows a table of variables and their values, with annotations explaining the display of variable values, instruction results, and variable values for selected lines.

**When a variable is selected, the value is displayed in the tooltip**

**Always the result of the instruction**

**Not executed instructions are greyed-out**

**Display all values of the instruction**

**When the line is selected, the values are presented in a popup**

#Max_No	2
#Max_No	10
Result	FALSE
#full	FALSE
#full	FALSE
Result	FALSE
Result	TRUE
#Init	TRUE
#WeightStore.Act_No	0
#count	1
#WeightStore.Part Weight [1]	0
#count	1

#### Monitoring an SCL block

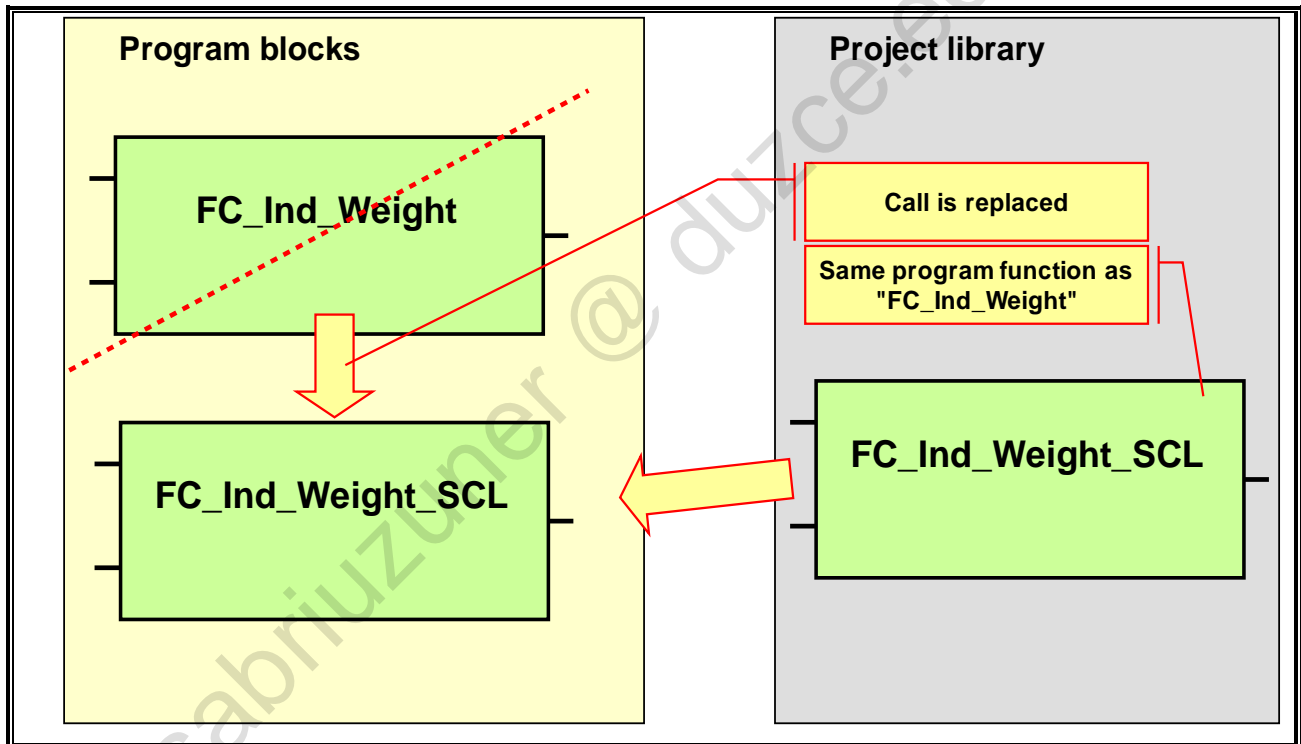
Just as in LAD/FBD/STL, SCL blocks can be monitored. All values of control operation instructions are displayed when these are "opened and revealed".

The values of not executed instructions are grayed out.

#### Special Features of Selected Variables

- In the online mode, a Tooltip displays the value of the variable.
- The value of IN-OUT variables is displayed before and after the call of the function.

## 10.7. Task description: Commissioning an SCL block and expanding it



### Situation up until now

The weight values of transported parts are stored in the array variable "DB\_Parts".Part\_Weight in "FC\_Ind\_Weight" via indirect addressing using "FieldWrite".

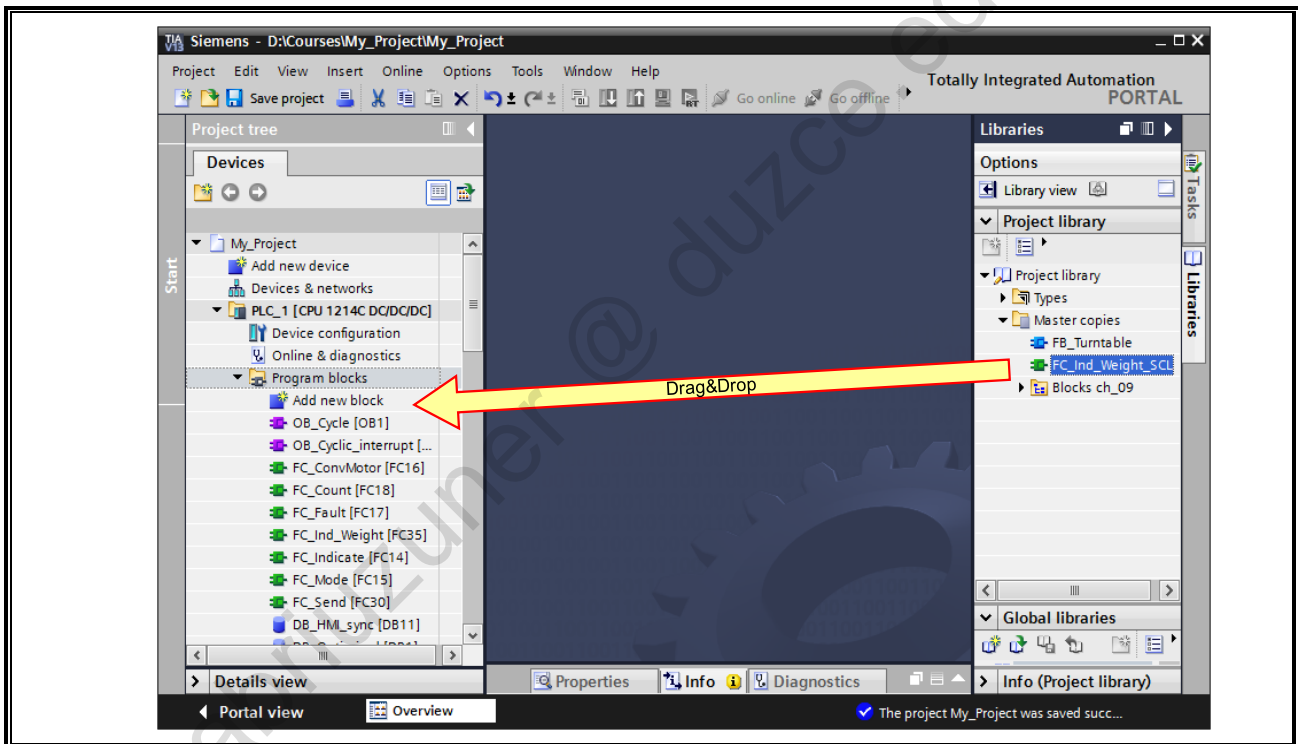
Old weight values are overwritten with new ones when the ring buffer ("DB\_Parts".Part\_Weight) is rewritten.

### Task description

In the first step, the function "FC\_Ind\_Weight" and its call in "FC\_Count" is to be replaced by the function "FC\_Ind\_Weight\_SCL". This is copied out of the Project library and fulfills the same function.

Subsequently, the initializing of the DB variable "DB\_Parts".PartWeights is to be programmed: As soon as the setpoint quantity equals the actual quantity and the bay pushbutton at the light barrier bay is pressed, all array elements in "DB\_Parts".PartWeights are to be overwritten with the value "0".

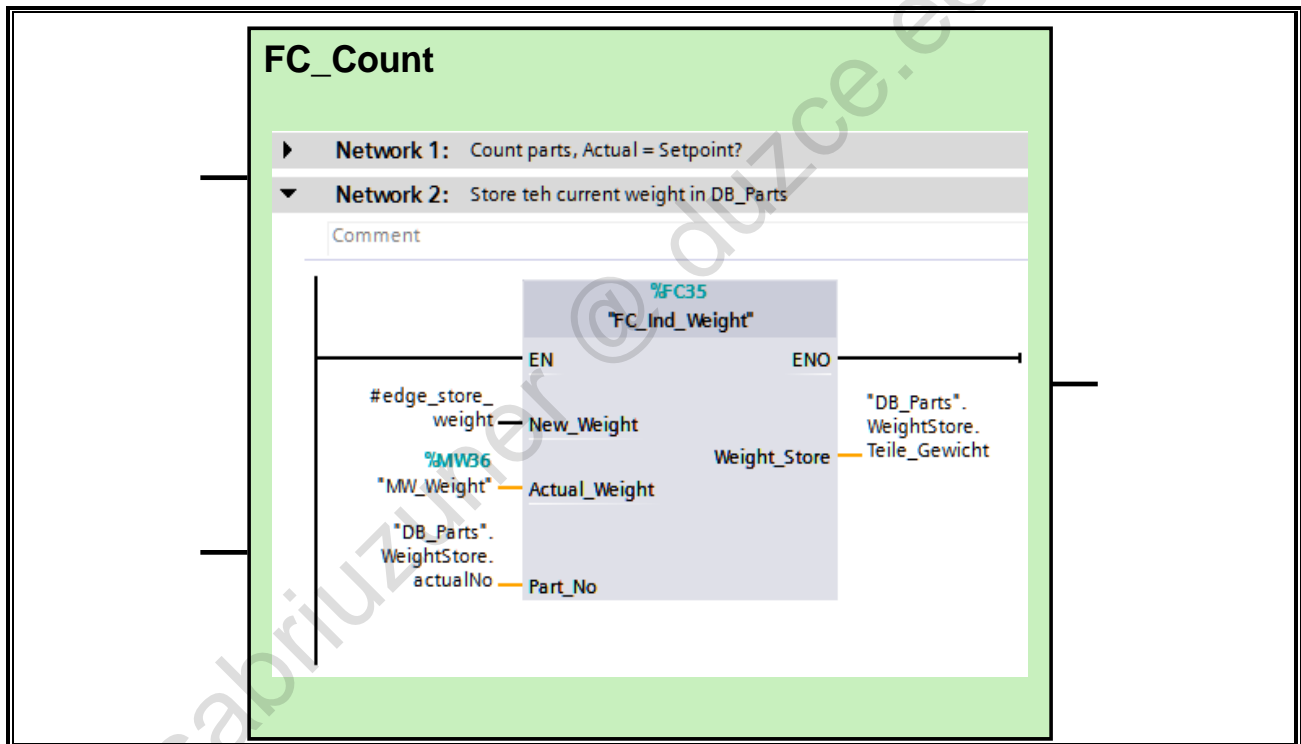
### 10.7.1. Exercise 1: Copying an SCL block from the project library



#### Task

Using drag & drop, copy the function "FC\_Ind\_Weight\_SCL" from the project library into the program blocks folder of "PLC1".

## 10.7.2. Exercise 2: Commissioning the SCL block



### Task

Replace the call of "FC\_Ind\_Weight" in "FC\_Count" with "FC\_Ind\_Weight\_SCL".

### What to do

1. Open "FC\_Count"
2. Replace the call of "FC\_Ind\_Weight" with "FC\_Ind\_Weight\_SCL"
3. Save your project and transfer the entire program into the CPU
4. Monitor "DB\_Parts" and produce at least (setpoint quantity) parts

### Result

The program behaves just as before: The weight values are written in the array variable "DB\_Parts".PartWeights according to the actual quantity. Old values are overwritten with new ones as soon as the ring buffer is full.

### 10.7.3. Exercise 3: Expanding "FC\_Ind\_Weight\_SCL"

```

1
2 IF #New_Weight THEN // if #New_Weight=TRUE, then...
3   #Weight_Store[#Part_No] :=#Actual_Weight; //...#Actual_Weight will be written...
4   //...to element with number #Part_No
5
6 END_IF;
7
8
9 IF #Init THEN // if #Init=true then...
10  FOR #index := 1 TO 100 DO //...#index is increased by 1 by ever loop ..
11    #Weight_Store[#index] := 0; //...0 ist written to...
12    //...the element with number #index
13  END_FOR;
14 END_IF;
15

```

#### Task

You are now to program the initialization of the DB variable "DB\_Parts".PartWeights:  
As soon as the actual quantity equals the setpoint quantity and the bay pushbutton at the light barrier bay ("S\_BayLB") is pressed, all array elements in "DB\_Parts".PartWeights are to be overwritten with the value "0".

#### What to do

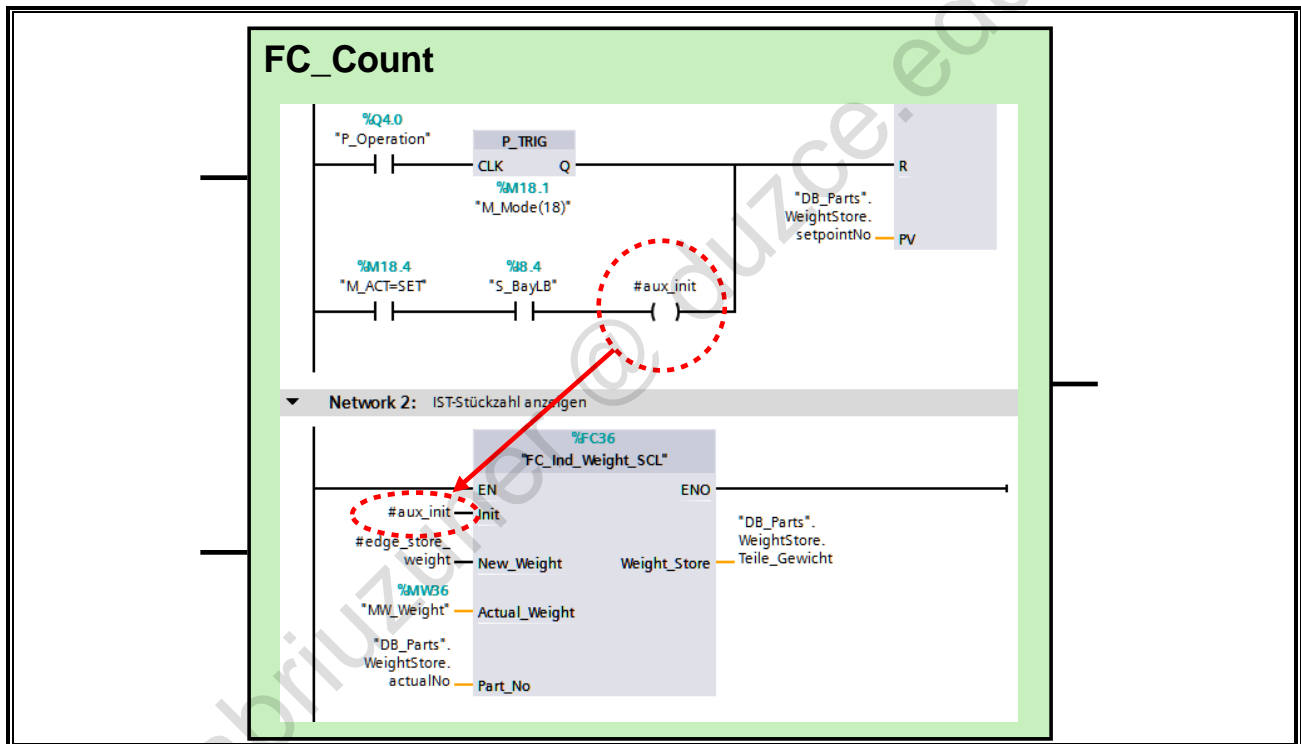
1. Open "FC\_Ind\_Weight\_SCL"
2. Declare the input parameter "init" of type BOOL
3. Declare the temporary variable "index" of type INT
4. Make the changes to the program code as shown in the picture
5. Save your project

#### Operating Principle

With the first IF statement instruction, it is checked whether a new weight is to be stored. All values are stored in the OUT-Array-Variable "Weight\_Store" with the Index "Part\_No".

In the second IF statement instruction, it is checked whether an initialization is to be executed. If this is the case, a For-loop with 100 executions is started. In every execution, one array element in "Weight\_Store" is overwritten with "0". The loop counter is automatically increased.

## 10.7.4. Exercise 4: Updating and assigning the block call



## Task

Due to the changes in the block interface, the call of "FC\_Ind\_Weight\_SCL" in "FC\_Count" must be updated.

The initialization of the DB is always to happen when the message "SETP=ACT" is pending and the bay pushbutton on the light barrier ("S\_Bay-LB") is pressed.

## What to Do

1. Open "FC\_Count"
2. Update the call of "FC\_Ind\_Weight\_SCL" (FC36)  
Right-click on "FC\_Ind\_Weight\_SCL" → Update → OK
3. Assign a positive signal edge to the input "init" as shown in the picture
4. Save your project and transfer the modified user program into the CPU
5. Monitor "DB\_Parts" and produce at least (setpoint quantity) parts in order to determine whether the DB is initialized when the init condition is fulfilled

# Contents

# 11

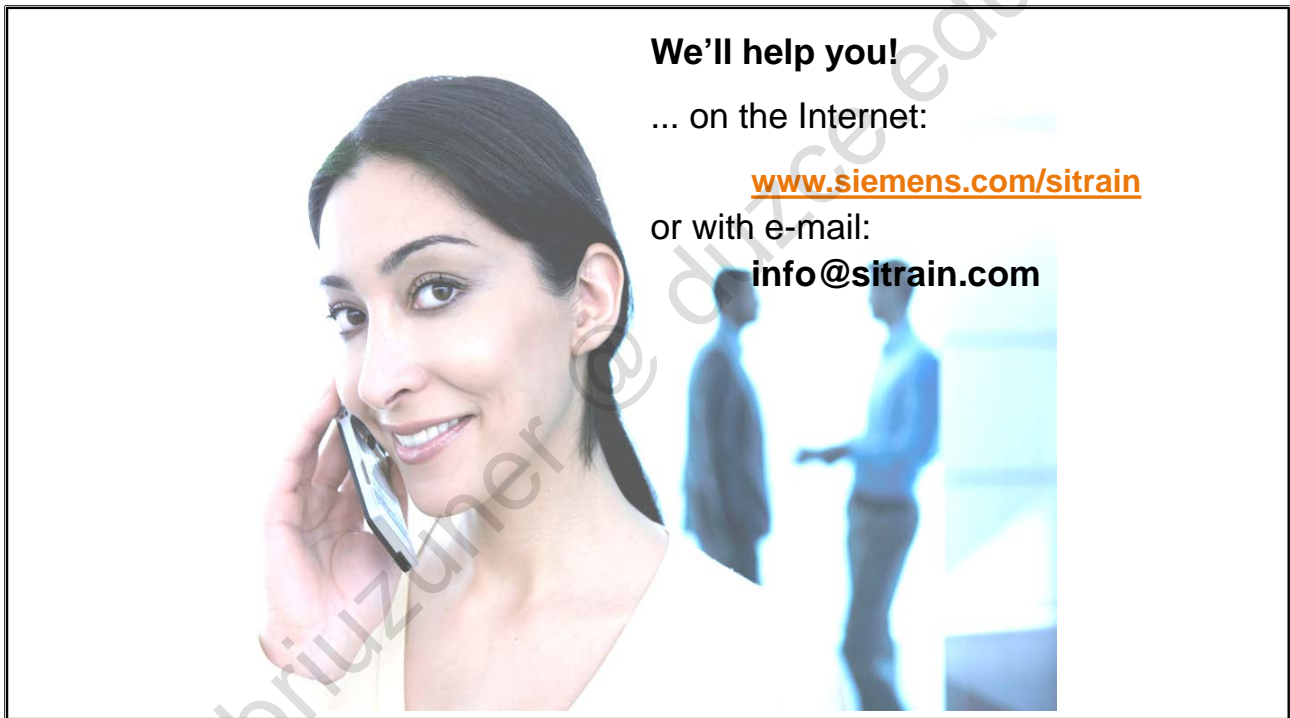
<b>11. Training and Support .....</b>	<b>11-2</b>
11.1. Any Questions on our Training Courses Offered?? .....	11-3
11.2. www.siemens.com/sitrain .....	11-4
11.3. Learning path: SIMATIC S7 Prgramming in the TIA Portal .....	11-6
11.4. Download the training documents.....	11-7
11.5. The Industry Online Support – the most important innovations.....	11-8
11.6. The Principle of Navigation .....	11-9
11.7. Complete product information.....	11-10
11.8. mySupport – Overview.....	11-11
11.9. Support Request .....	11-12
11.10. Support Request .....	11-13
11.11. Industry Online Support – wherever you go .....	11-14
11.11.1. Scanning product/EAN code.....	11-15
11.11.2. Scan functionality.....	11-16
11.12. Forum - the communication platform for Siemens Industry products .....	11-17
11.12.1. Conferences and Forum management .....	11-17
11.12.2. Interactions in the Forum .....	11-19
11.13. Task and Checkpoint .....	11-21

# 11. Training and Support





## 11.1. Any Questions on our Training Courses Offered??



**We'll help you!**  
... on the Internet:  
[www.siemens.com/sitrain](http://www.siemens.com/sitrain)  
or with e-mail:  
[info@sitrain.com](mailto:info@sitrain.com)

### General Information

We'll be glad to help you regarding any questions on our training courses offered.

## 11.2. [www.siemens.com/sitrain](http://www.siemens.com/sitrain)



The complete range of courses offered can be accessed via the following links:

[www.siemens.de/sitrain](http://www.siemens.de/sitrain) or

[www.siemens.com/sitrain](http://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.

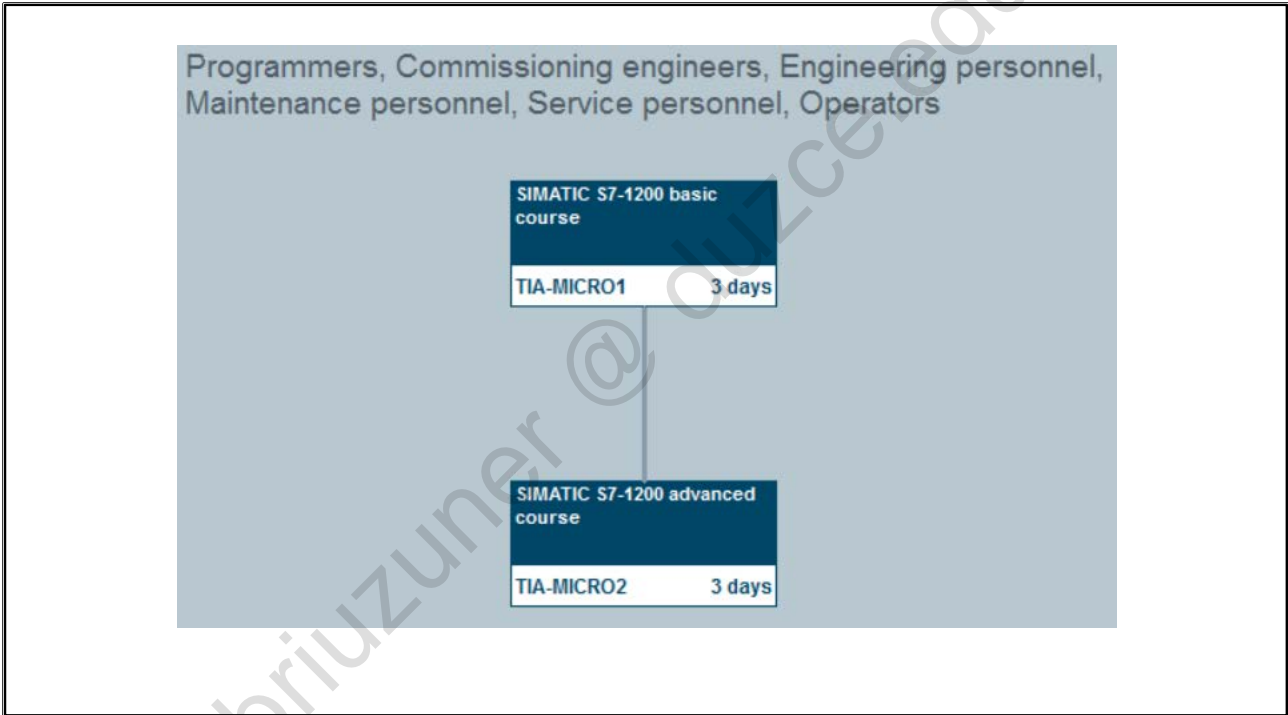
### SIMATIC S7 TIA Portal

- > 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

### 11.3. Learning path: SIMATIC S7 Prgramming in the TIA Portal



## 11.4. Download the training documents

Register with your access data

1

Chose "History" after the course.

Chose the course

Download your documents

Training Title (ID)	Type	Start Date	Duration	Complete by	Country	City	Language	Fee	Available Until	Details
SIMATIC S7 Sequence Control with S7...		Jan 05, 2016 08:30	5 days		EN	Hannover...	en	EUR		<ul style="list-style-type: none"> <li>Download documents</li> <li>Download certificate of participation</li> </ul>

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

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

Documents		
Name		Size
> SIMATIC S7 Sequence Control with ...		18,47 MB

### Hint:

Please note that the training documents may be used for personal purposes exclusively. You agree that you will not copy the training documents or make them accessible to third parties and that you will be liable for any damage resulting thereof.

## 11.5. The Industry Online Support – the most important innovations

The screenshot shows the Siemens Industry Online Support website. The layout is divided into several sections:

- 1**: The top navigation bar (Site Explorer) containing links for Home, Language, Contact, Help, and Support Request.
- 2**: The main content area, including a search bar for product information, a 'Quick guide' with links to various resources, and a 'TIA Portal - Topic Page' featuring a grid of images and a video player.
- 3**: The secondary navigation bar (Product Support, Services, Forum, mySupport) and the 'mySupport Cockpit' section, which displays personalized user information and support options.
- 4**: The 'Useful functions in the Online Support (videos)' section, which provides video tutorials for various support features.

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

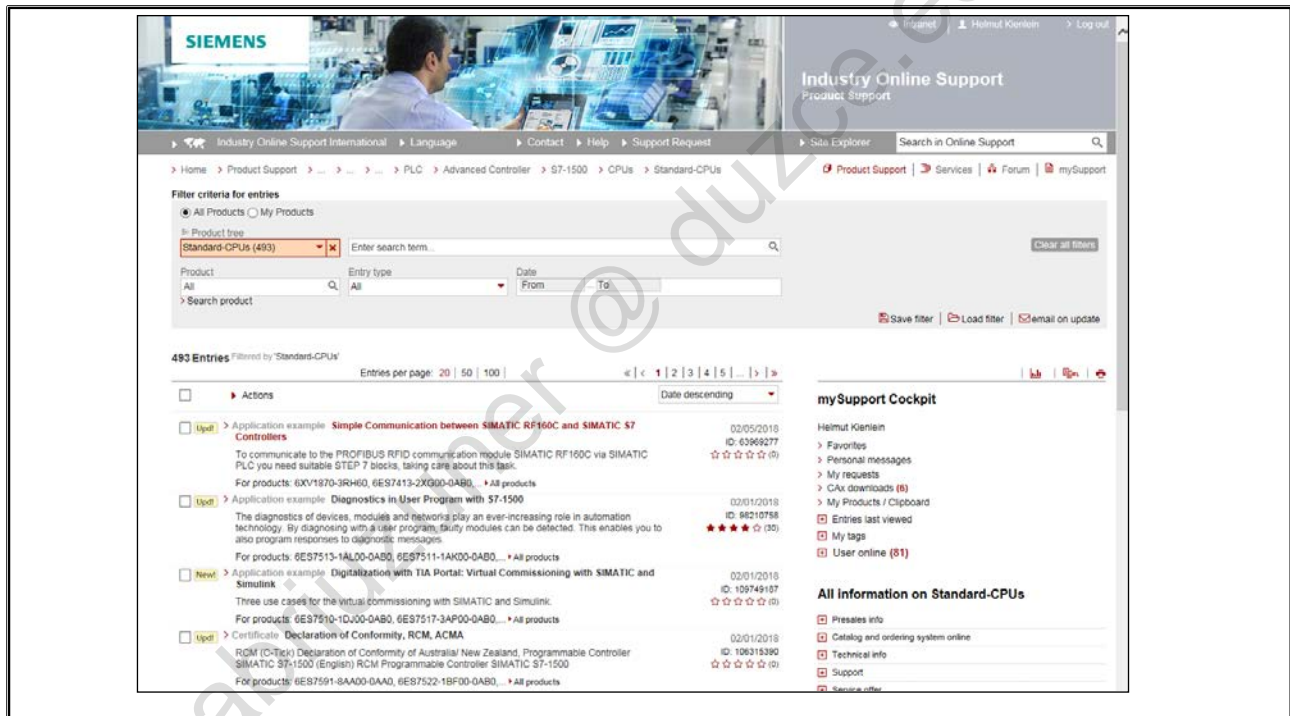
Links from the menu bar are repeated at the top of the page: Product Support, Services, Forum and mySupport.

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.



## 11.6. 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 through the product tree, but also through a central filter bar. The integration of various search filters will give you access to relevant information after only a few clicks. The product tree has been moved to an equivalent filter. This has the effect that several filter steps can be combined clearly and comprehensibly.

Based on the preview numbers you can see the expected set of results before using a filter. This makes finding relevant information considerably easier and more efficient.

For example, you can customize your search by combining the product tree, a search keyword and a document type in your search. There will be no hidden search parameters; all the settings and results will be clearly displayed.

## 11.7. Complete product information

The screenshot displays the Siemens Industry Online Support interface. At the top, there is a navigation bar with options like 'Home', 'Product Support', and 'Automation Technology'. A search bar is prominently featured. Below the search bar, there are filter criteria for entries, including 'All Products' and 'My Products'. A search term '6ES7513-1AL01-0AB0' is entered, and a list of suggested products is shown. The selected product, '6ES7513-1AL01-0AB0', is highlighted, and its technical specifications are displayed, including 'CPU 1513-1 PN, 300KB PROG., 1.5MB DATA'. The page also shows a 'mySupport Cockpit' section with user information and a list of recent entries.

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.

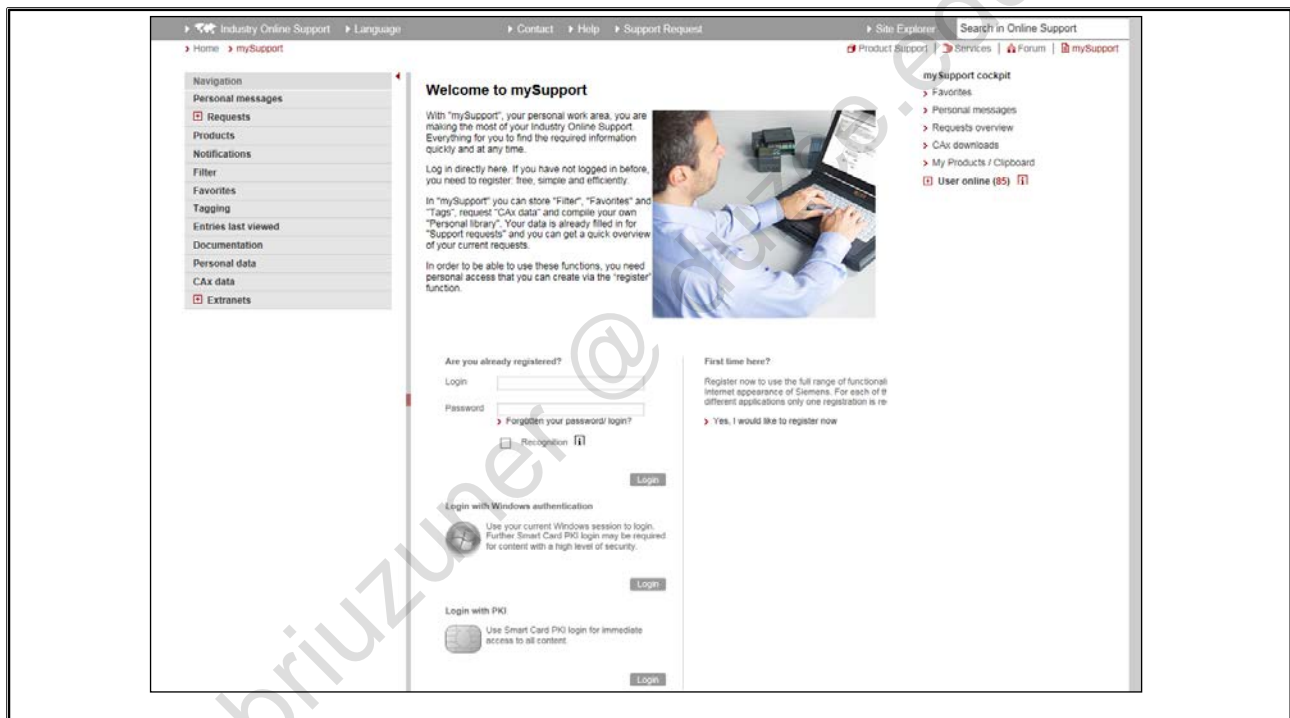
To select a product, simply select the filter "Product." Enter an order number or a product name here. You will be supported by a dynamic display of suitable products (list of suggestions).

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.



## 11.8. 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.

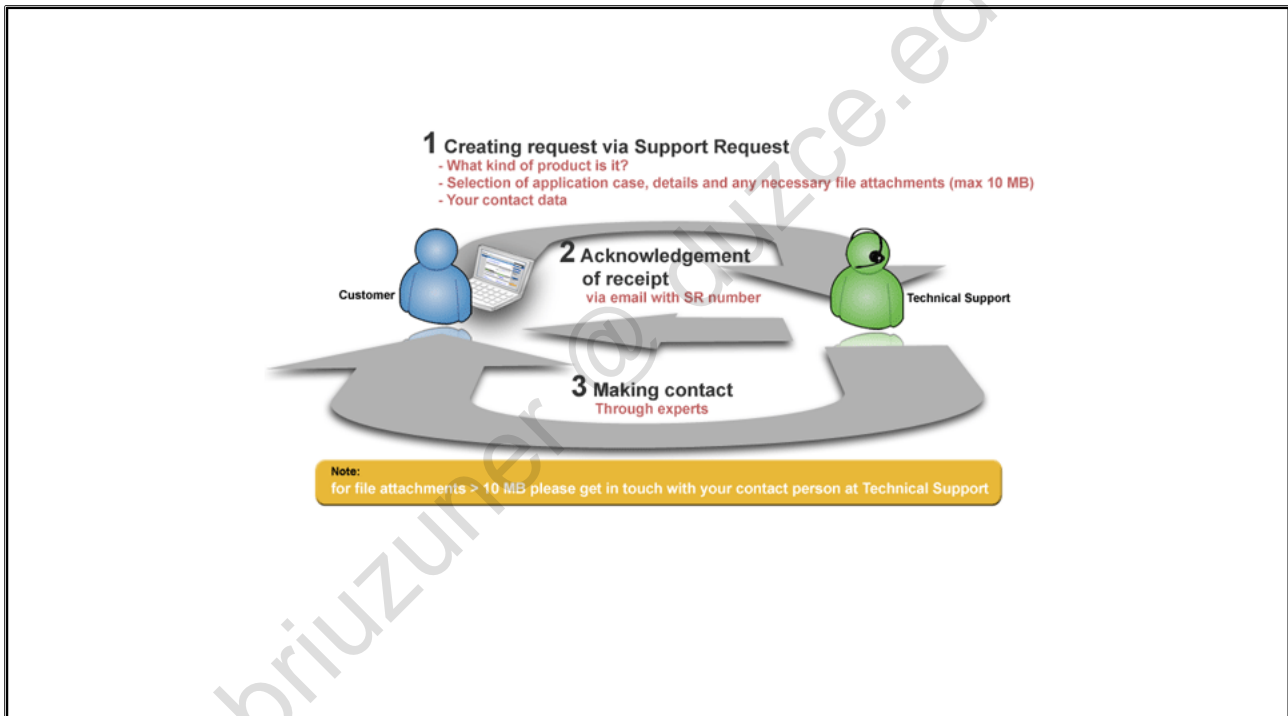
Moreover, you can now add notes, comments and tags (keywords). The system automatically creates a "Tag Cloud" based on your entries so you can access information quickly and easily by means of your own terms. The operation is consistent throughout mySupport so that you will easily find your way around.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

## 11.9. 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.

## 11.10. Support Request

Industry Online Support | Language | Contact | Help | Support Request | Sit

Home > mySupport > Requests

**Navigation**

- Personal messages
- Requests**
  - Overview
  - Create new request
- Products
- Notifications
- Filter
- Favorites
- Tagging
- Entries last viewed
- Documentation
- Personal data
- CAX data

**My requests**

Search in "My requests"  
for  search

Status: Everything

Actions: New request | Show details...

Items per page: 10 | 20 | 30

<input type="checkbox"/>	SR number	Product and subject	Status	Created on
<input type="checkbox"/>	1-3871916175	STEP7 Professional V13	Closed	2/12/2015 8:49 AM

Show details... | Add note

Items per page: 10 | 20 | 30

## 11.11. 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:

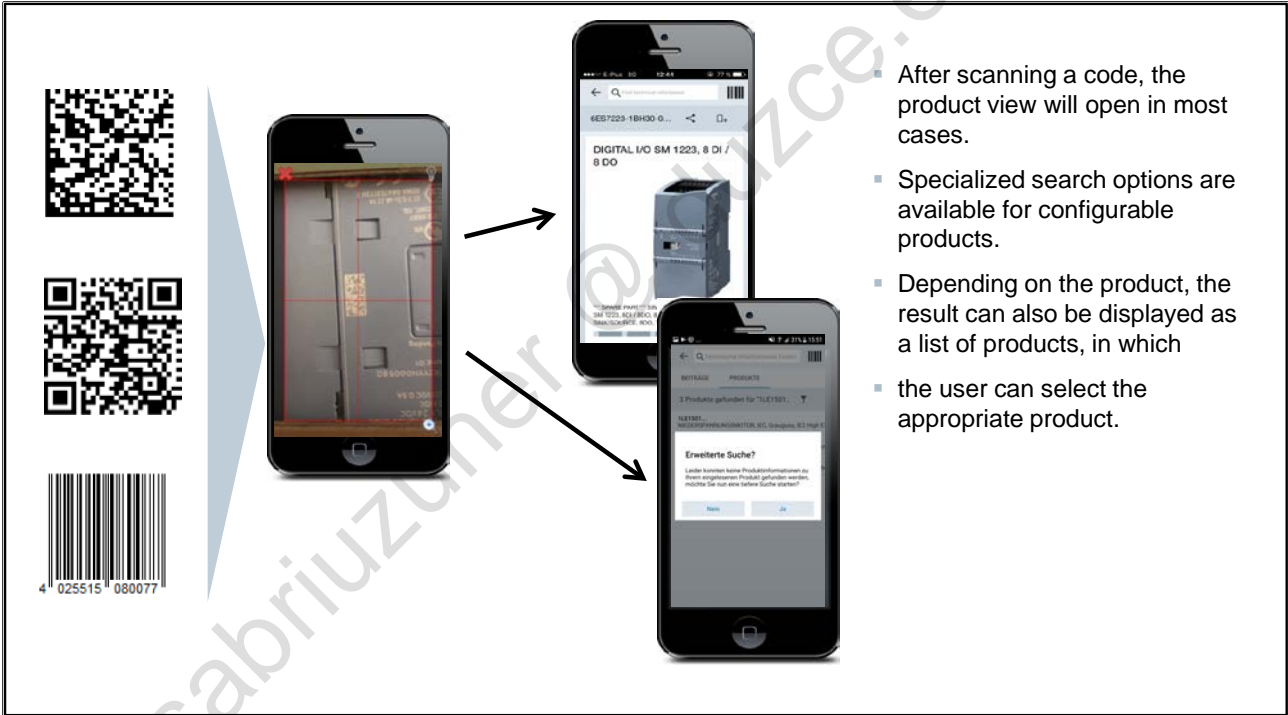
- FAQs
- Application examples
- Manuals
- Certificates
- Product notes and many others

### 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.

### 11.11.1. Scanning product/EAN code



### 11.11.2. Scan functionality

<p><b>Data matrix codes</b></p>  <p>on Siemens products as per standard SN60450</p>	<p><b>QR code</b></p>  <p>e.g.: in advertisements relating to Siemens content</p>	<p>The scan functionality in the Online Support app supports the following types of code:</p>
<p><b>EAN13 bar code</b></p>  <p>on Siemens products</p>	<p><b>Code39 bar code</b></p>  <p>(very hard to recognize / scan) on Siemens products as per standard SN60450</p>	<ul style="list-style-type: none"> <li>Data matrix code</li> <li>QR code</li> <li>EAN13 bar code</li> <li>Code39 bar code</li> </ul>
		<p>When one of these codes is recognized, the respective product view is called up in the app.</p>
		<p><b>Exception:</b> 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).</p>

## 11.12. Forum - the communication platform for Siemens Industry products

### 11.12.1. Conferences and Forum management

The screenshot displays the Siemens Industry Online Support Forum interface. The top navigation bar includes 'Industry Online Support', 'Deutsch', 'Contact', 'Help', 'Support Request', 'Site Explorer', and a search bar. The main content area is divided into three sections:

- Left Navigation Panel (1):** A vertical menu with options: 'Product Conferences', 'Solutions, Applications and Initiatives', 'General Conferences', 'Forum management', 'Quicklinks', and 'Forum Terms'. The 'Product Conferences' link is highlighted with a red circle labeled '1'.
- Central Conference Overview (2):** A section titled 'Conference overview' with a search bar and a list of product conferences. The title 'Conference overview' is highlighted with a red circle labeled '2'. The list includes items like 'LOGO!' and 'SIMATIC TDC, FM458, T400'.
- Right mySupport Cockpit:** A personal dashboard with sections for 'Favorites', 'Personal messages (0)', 'My requests (2)', 'CAx downloads (0)', 'Viewed item', 'My tags', and 'User online (54)'. Below this is an 'All about Product Information' section with links for 'Presales info', 'Catalog and ordering system online', 'Technical info', 'Support', and 'Training'.

At the bottom left of the navigation panel, the 'Forum management' link is highlighted with a red circle labeled '3'.

- 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.

> Home > Forum > Forum management > Conference filter Product

**Navigation**

- Product Conferences
- Solutions, Applications and Initiatives
- General Conferences
- Forum management
  - > Manage profile
  - > User filter
  - > Conference filter
  - > My topics
  - > My suggestions
  - > Received Feedback
- Quicklinks
- Forum Terms

**Manage conference filter**

Add or remove conferences to the filter

> Show topics in "My conferences"

**All conferences**

- Ankündigungen (de)
- Anregungen - Feedback (de)
- CNC Automatisierungssystem SINUMERIK (de)
- Code-Lesesysteme (de)
- DC Stromrichter SIMOREG (de)
- Der neue Industry Online Support (de)
- Desigo™ Gebäudeautomationssystem (de)
- Dezentrale Peripherie (de)
- Drive Tools (de)
- Energiemanagement mit SIMATIC (de)
- Gebäudetechnik allgemein (de)
- HLK Produkte (de)
- Industrie Software allgemein (de)
- Industrielle Schaltechnik (de)
- Kommunikation / Netzwerke (de)
- Logikmodul LOGO! (de)
- Meet & Talk (de)
- MICROMASTER (de)
- Motion Control System SIMOTION (de)
- Niederspannungs-Energieverteilung und Insta

**My conferences**

- Industrie-PC SIMATIC PC (de)
- Prozessleitsystem SIMATIC PCS 7 (de)

Enable notifications if there are new topics.

Set notifications

### Conference filter

Add conferences to your personal filter of preferred conferences.

This allows you to enable a notification that informs you when new topics are started in these conferences.

In Quicklinks, the Technical Forum additionally offers an overview page that contains all topics of your preferred conferences.

### Managing profile

Profile management provides interesting information and functions:

- You get an overview of your activities in the Technical Forum.
- You can view your rank, any special permissions and your ranking progress.
- You can store a signature and a personal description for your profile in the forum.
- You have direct access to the quick links to get an overview of all topics you have contributed to.

### User filter

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

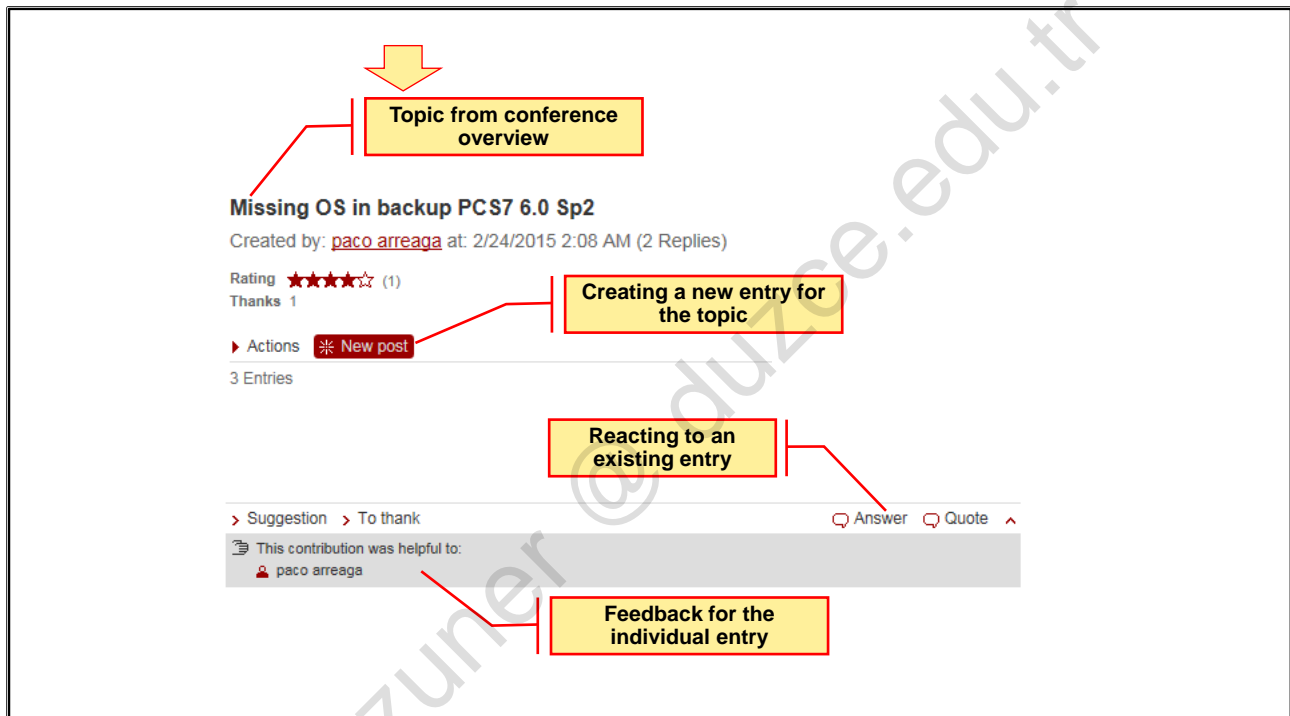
This allows you to enable a notification that informs you when the user has posted a new entry.

In Quicklinks, the Technical Forum additionally offers an overview page that contains all topics of your preferred users.



### 11.12.2. Interactions in the Forum

The screenshot shows a forum page titled "Process Control Systems SIMATIC PCS 7". At the top, there is a search bar with the text "Search in 'Process Control Systems SIMATIC PCS 7' for" and buttons for "search" and "reset all". Below the search bar, there are navigation links: "Actions", "Legend", "Experts", and "New topic" (highlighted with a red box and a callout "Creating a new topic in a conference"). There are also "Edit filter" and "Load filter" buttons. The page indicates "4748 Entries" and "Entries per page: 10 | 20 | 50". A list of forum entries is shown, with the first entry highlighted by a yellow arrow pointing to a "Status: solved" box. The entry text is "> Missing OS in backup PCS7 6.0 Sp2" and "from: paco arreaqa". To the right of the entry, it says "from: paco arreaqa" and "2/26/2015 3:41 AM". Further right, there are "2" and "81" (with a "(1)" below it) and a star rating of five stars (the last one is empty). A "Rating of the topic" box points to the star rating. A "Topic from conference overview" box points to the entry title. A "Status: solved" box points to a checkmark icon next to the entry title.



### 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.

## 11.13. 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?

sabriuzuner@duzce.edu.tr

sabriuzuner@duzce.edu.tr

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

[siemens.com/sitrain](https://www.siemens.com/sitrain)