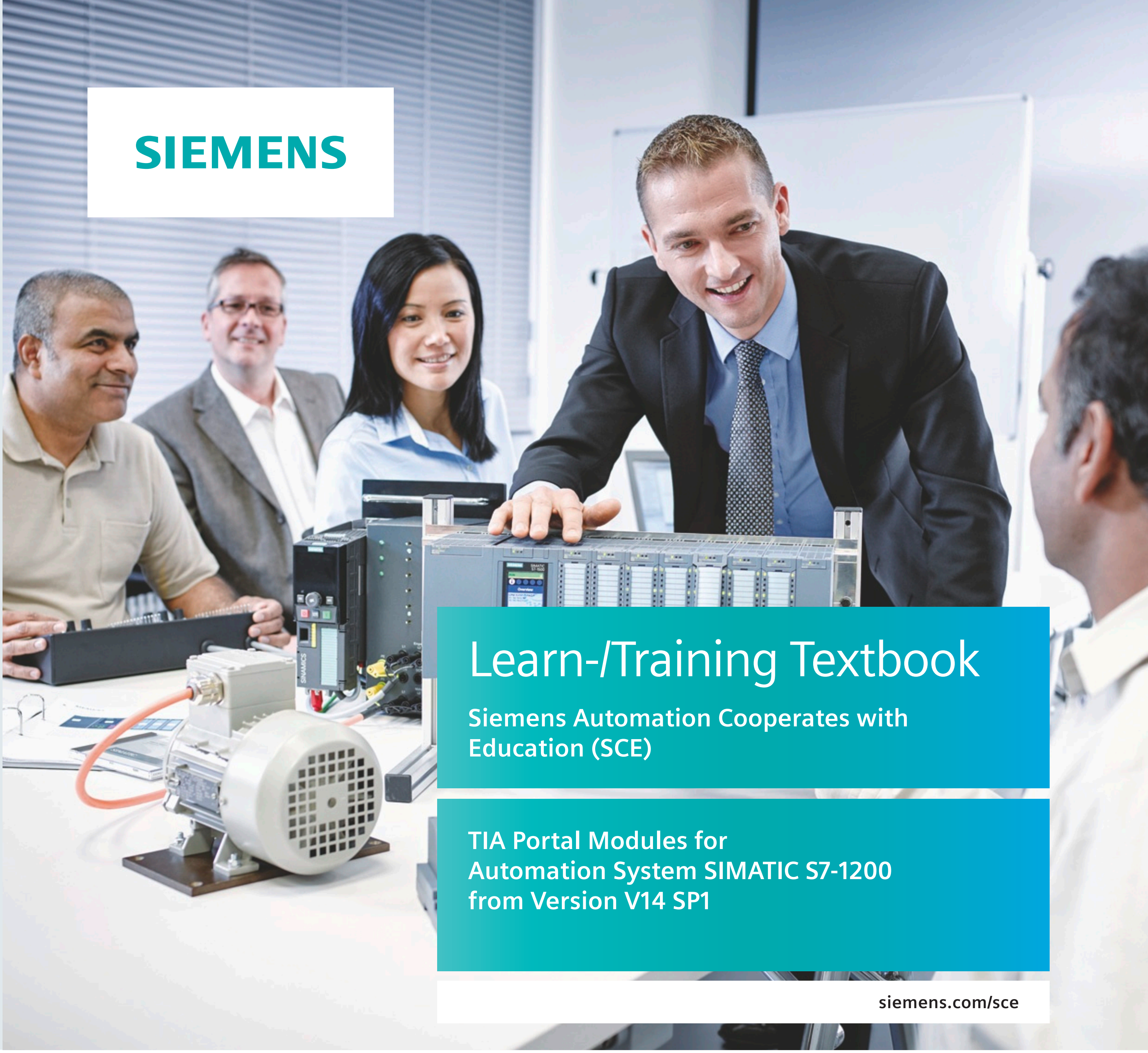


SIEMENS

Learn-/Training Textbook

Siemens Automation Cooperates with Education (SCE)

TIA Portal Modules for Automation System SIMATIC S7-1200 from Version V14 SP1

siemens.com/sce

SIEMENS

Global Industry Partner of WorldSkills International

worldskills

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Please note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is provided at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, contact your regional SCE representative: siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

Table of contents

Firmware Update with SIMATIC S7-1200

1 Goal

This chapter will show how the **firmware version of the CPU** of **SIMATIC S7-1200** can be checked and upgraded using the TIA Portal.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Requirement

No prerequisites have to be met for successful completion of this chapter.

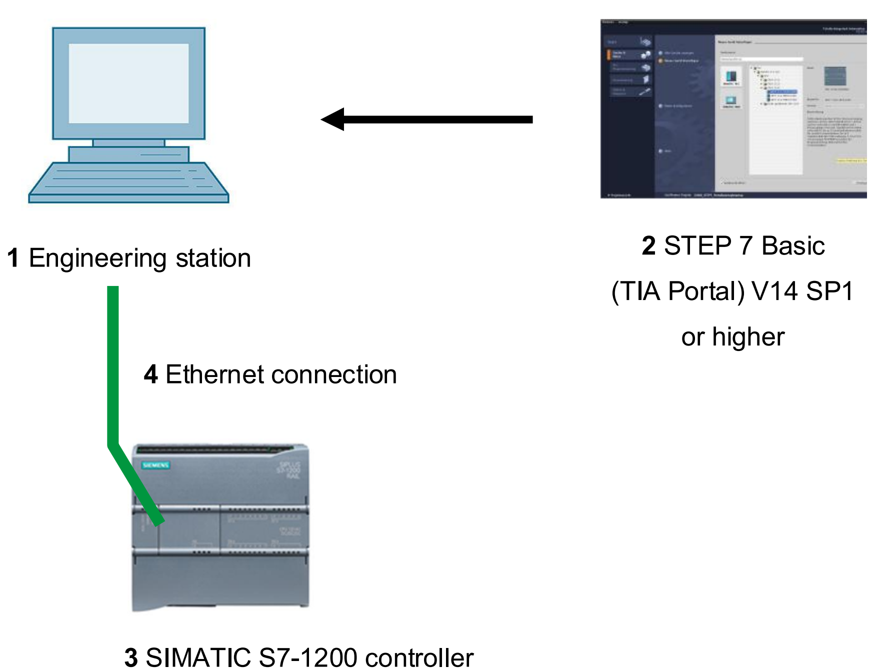
3 Required hardware and software

1 Engineering station: requirements for hardware and operating system
(for additional information, see Readme on the TIA Portal Installation DVD)

2 STEP 7 Basic software in TIA Portal – V14 SP1 or higher

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC – Firmware V4.2.1 or higher

4 Ethernet connection between engineering station and controller



4 Theory

4.1 SIMATIC S7-1200 automation system

The SIMATIC S7-1200 automation system is a modular microcontroller system for the lower performance range.

A comprehensive range of modules is available to optimally adapt the system to the automation task.

The S7 controller consists of a power supply and a CPU with integrated inputs and outputs or additional input or output modules for digital and analog signals.

If necessary, communication processors and function modules for special tasks such as stepper motor control are also used.

1

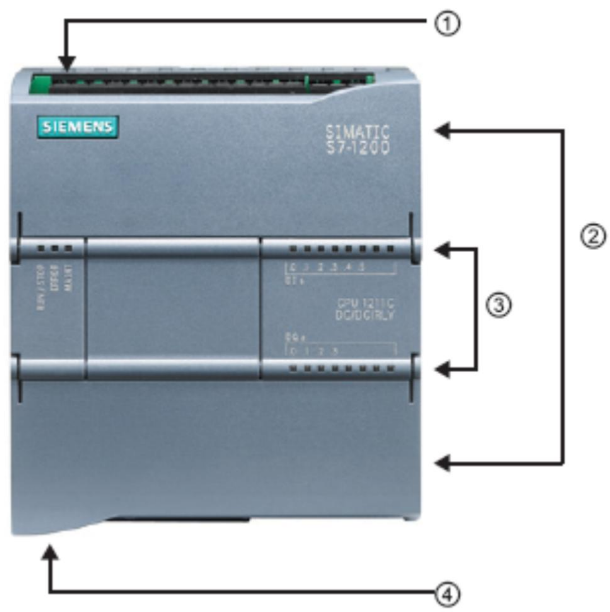
4.2 Operator controls and display elements of the CPU 1214C DC/DC/DC

4.2.1 Front view of the CPU 1214C DC/DC/DC

With an integrated power supply (24 V connection) and integrated inputs and outputs, the CPU 1214C DC/DC/DC is ready for immediate use without additional components.

For communication with a programming device, the CPU has an integrated TCP/IP connection.

The CPU can thus communicate with HMI devices or other CPUs over an Ethernet network.



① 24 V connection

② Plug-in terminal block for user wiring (behind the cover flaps)

③ Status LEDs for the integrated IO and the operating mode of the CPU

④ TCP/IP connection (on the bottom of the CPU)

4.2.2 SIMATIC Memory Card (MC)

The optional **SIMATIC Memory Card (MC)** stores the program, data, system data, files and projects. It can be used for:

– Transfer of a program to multiple CPUs

– Firmware update of CPUs, signal modules (SM) and communication modules (CM)

– Simple CPU replacement

4.2.3 Operating modes of the CPU

The CPU can have the following three operating modes:

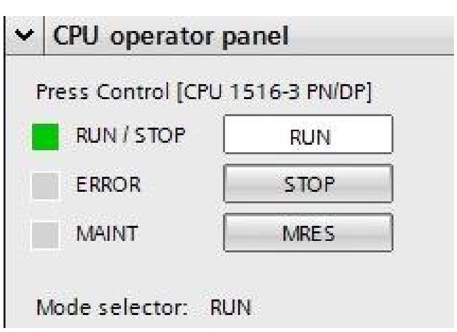
– In **STOP** mode, the CPU is not executing the program and you can download a project.

– In **STARTUP** mode, the CPU is starting up.

– In **RUN** mode, the program is being executed cyclically.

The CPU does not have a physical switch for changing the operating mode.

You use the button on the operator panel of the STEP 7 Basic software to change the operating mode (**STOP** or **RUN**). The operator panel also contains an **MRES** button for performing a memory reset and displays the status LEDs of the CPU.



1

4.2.4 Status and error displays

The **Status LED RUN/STOP** on the front of the CPU indicates the current operating mode of the CPU by the color indicated.

- **Yellow** light indicates **STOP** mode.
- **Green** light indicates **RUN** mode.
- A **flashing light** indicates **STARTUP** mode.

In addition, there is an **ERROR** LED for indicating errors and a **MAINT** LED for indicating a maintenance requirement.

4.3 STEP 7 Basic programming software (TIA Portal V14)

STEP 7 Basic V14 (TIA Portal V14) software is the programming tool for the following automation systems:

– SIMATIC S7-1200

– Basic Panels

STEP 7 Basic V14 provides the following functions for plant automation:

– Configuration and parameter assignment of the hardware

– Specification of the communication

– Firmware update

– Programming

– Testing, commissioning and service with operational/diagnostic functions

– Documentation

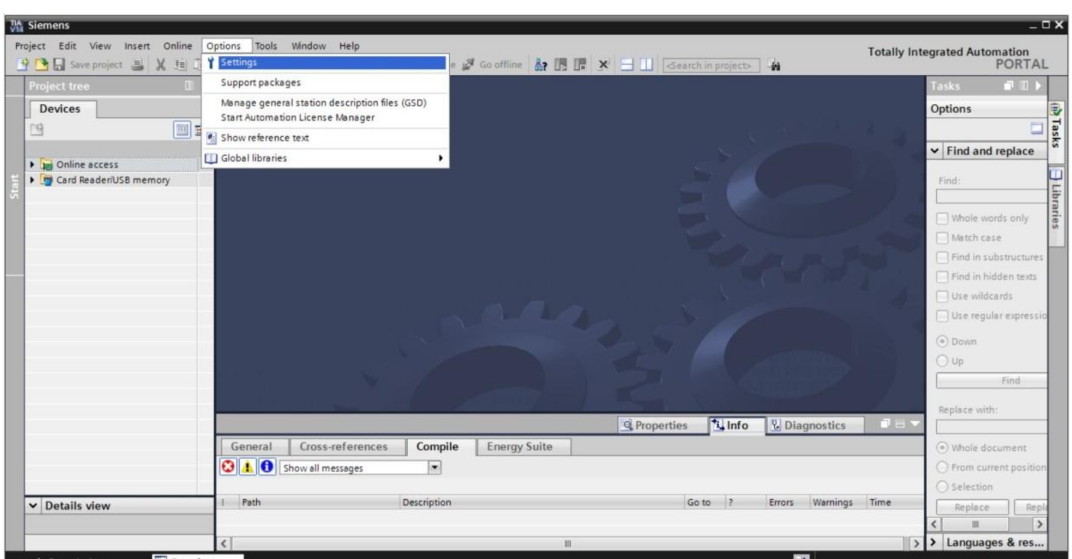
– Creation of visualizations for SIMATIC Basic Panels with the integrated WinCC Basic software

Support is provided for all functions through detailed online help.

4.3.1 Basic settings for the TIA Portal

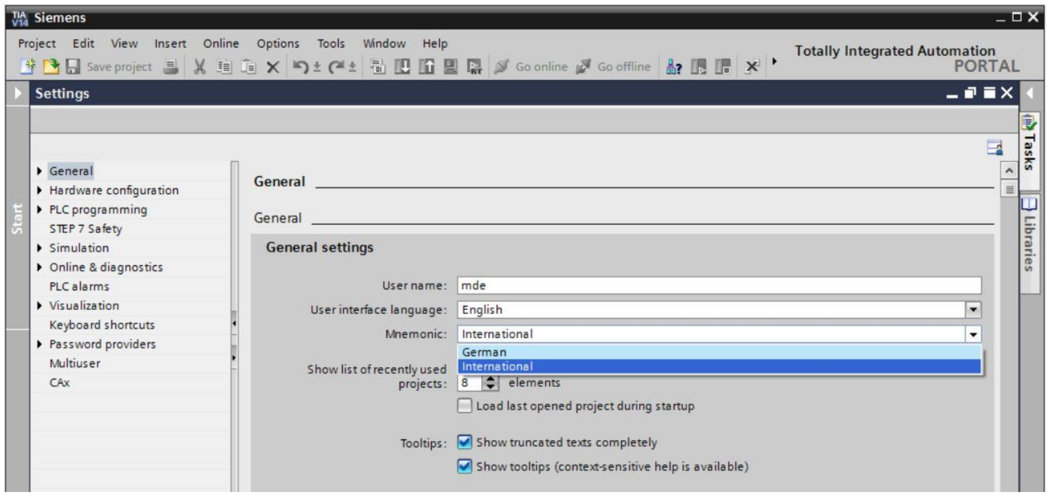
→ Users can specify their own default settings for certain settings in the TIA Portal. A few important settings are shown here.

→ In the project view, select the → "Options" menu and then → "Settings".



→ One basic setting is the selection of the user interface language and the language for the program display. In the curriculums to follow, "English" will be used for both settings.

→ Under → "General" in "Settings", select "User interface language → English" and "Mnemonic → International".



Note: *These settings can always be changed.*

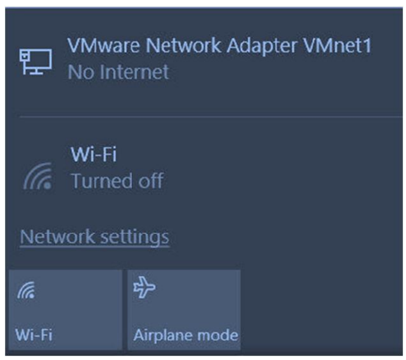
4.3.2 Setting the IP address on the programming device

You need a TCP/IP connection in order to upgrade the CPU of a SIMATIC S7-1200 controller from the PC, the programming device or a laptop.

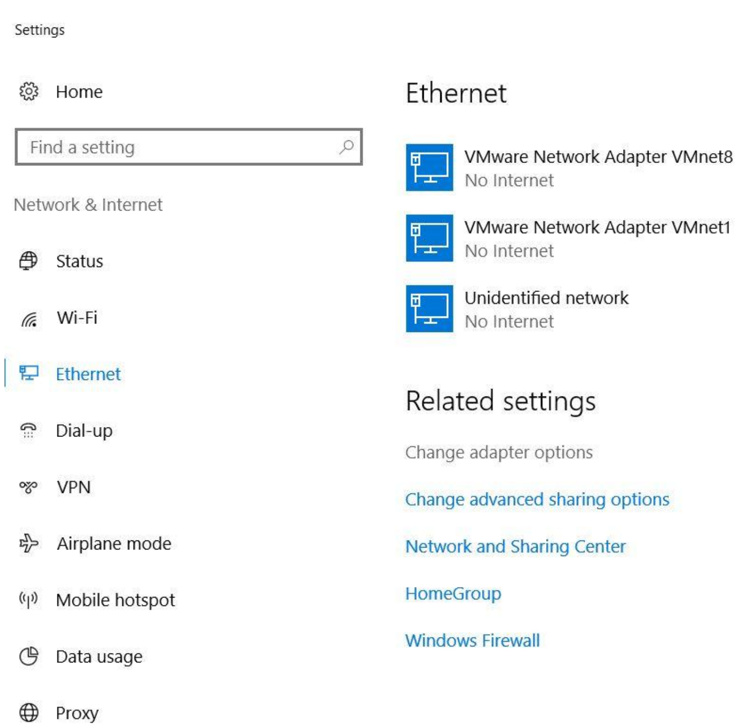
It is important that the IP addresses of both devices match for the computer and SIMATIC S7-1200 to communicate with each other via TCP/IP.

First, we will show you how to set the IP address of a computer with the Windows 10 operating system.

→ Select the network icon in the taskbar at the bottom  and click → "Network settings".

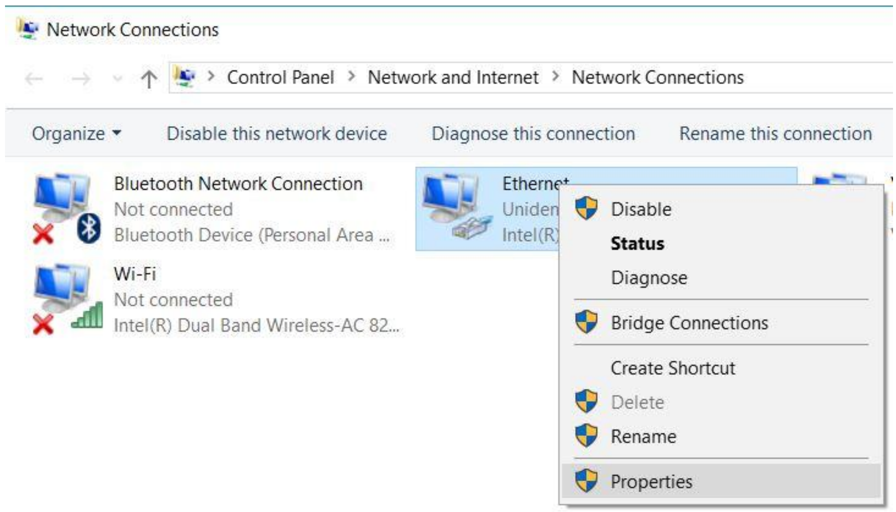


→ In the network settings window that opens, click → "Ethernet" and then on → "Change adapter options".

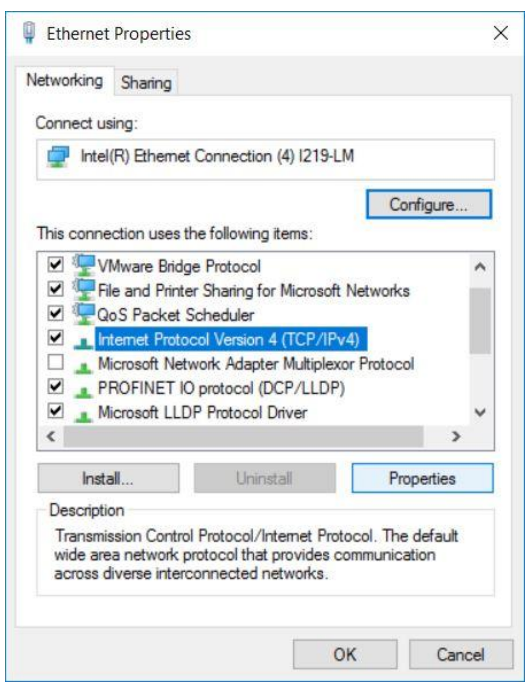


1

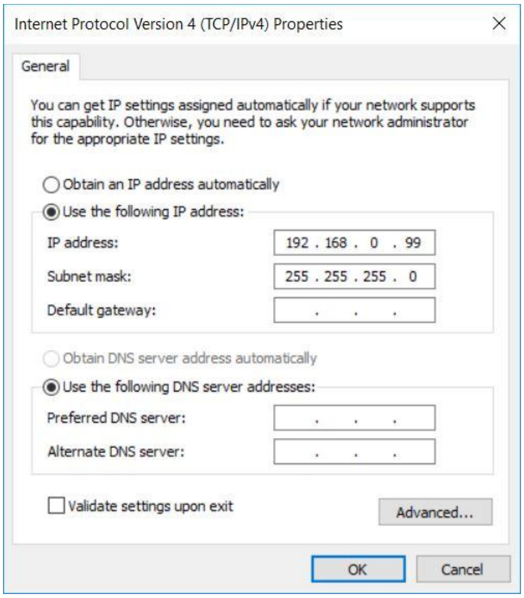
→ Select the desired → "LAN Connection" that you want to use to connect to the controller and click → "Properties".



→ Select → "Properties" for → "Internet Protocol Version 4 (TCP/IPv4)".



→ You can now use the following IP address, for example → IP address: 192.168.0.99 and enter the following → subnet mask 255.255.255.0. Accept the settings. (→ "OK")

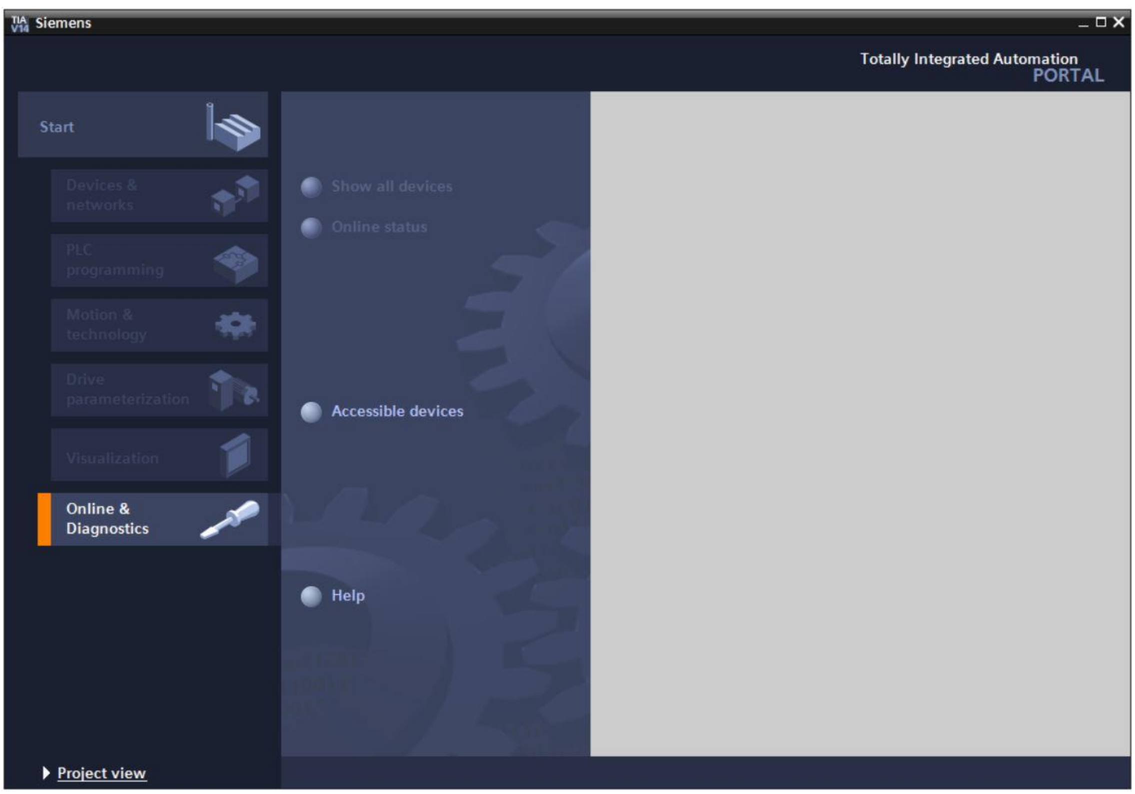


4.3.3 Setting the IP address in the CPU

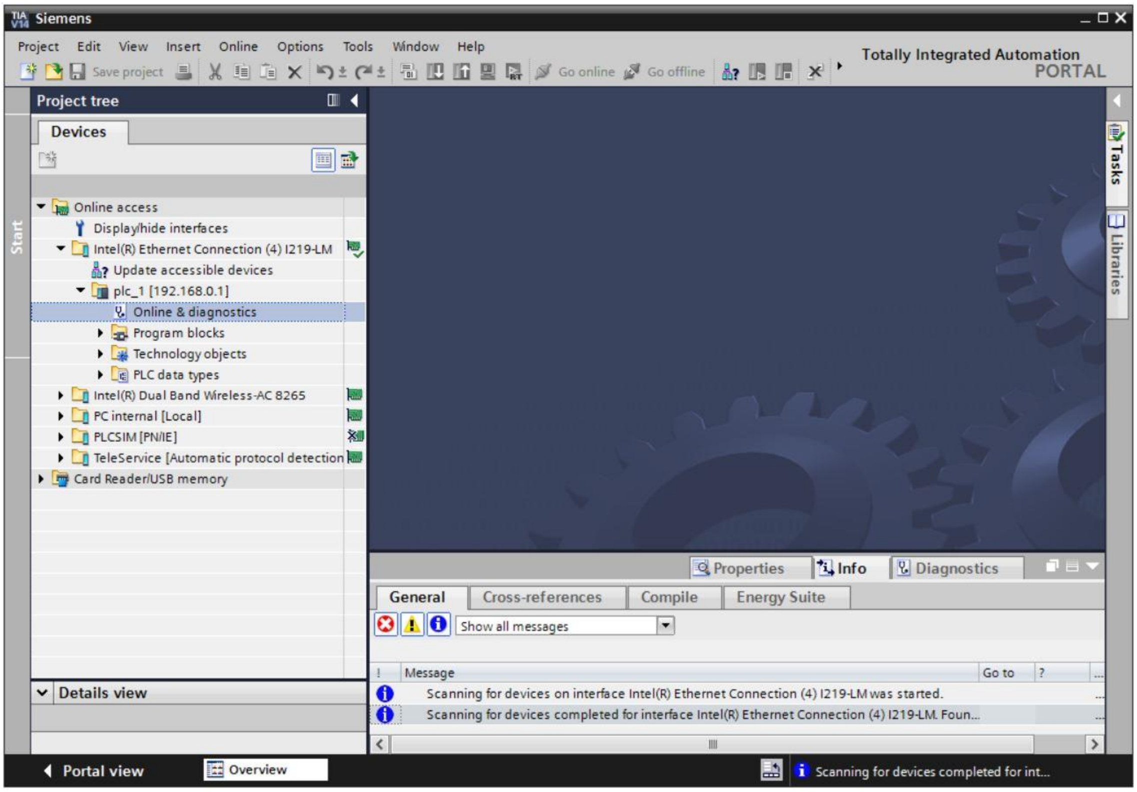
Before a firmware update of the CPU can be performed, set the IP address of the SIMATIC S7-1200 correctly so that the programming device can reach the CPU via TCP (IP communication). The IP address of the SIMATIC S7-1200 is set as follows:

→ Double-click the Totally Integrated Automation Portal to select it. (→ TIA Portal V14)

→ Click → "Online&Diagnostics" and open → "Project view".

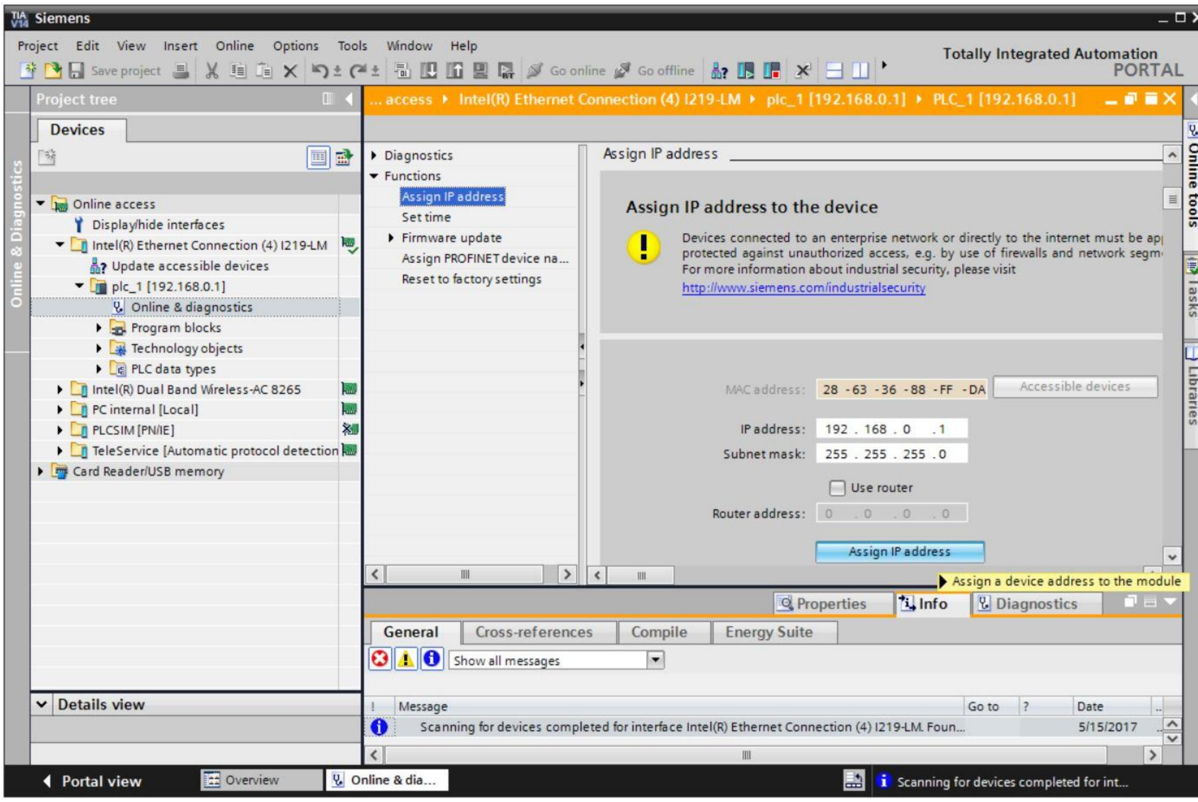


→ In the project tree under → "Online access", select the network adapter that was set previously. If you click → "Update accessible devices", you will see the IP address (if previously set) or the MAC address (if IP address has not yet been assigned) of the connected SIMATIC S7-1200. Select → "Online&Diagnostics".

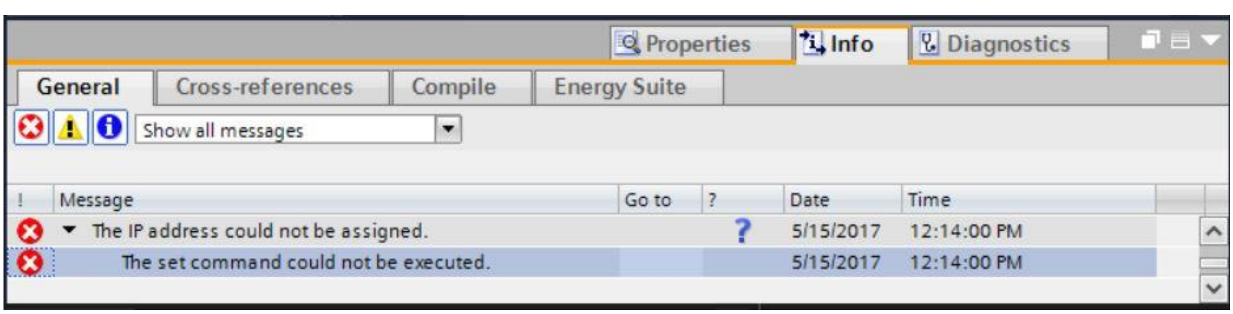


1

→ Under → "Functions", you now find the → "Assign IP address" item. Enter the following IP address here (example): → IP address: 192.168.0.1 → Subnet mask 255.255.255.0. Click → "Assign IP address" and this new address will be assigned to your SIMATIC S7-1200.

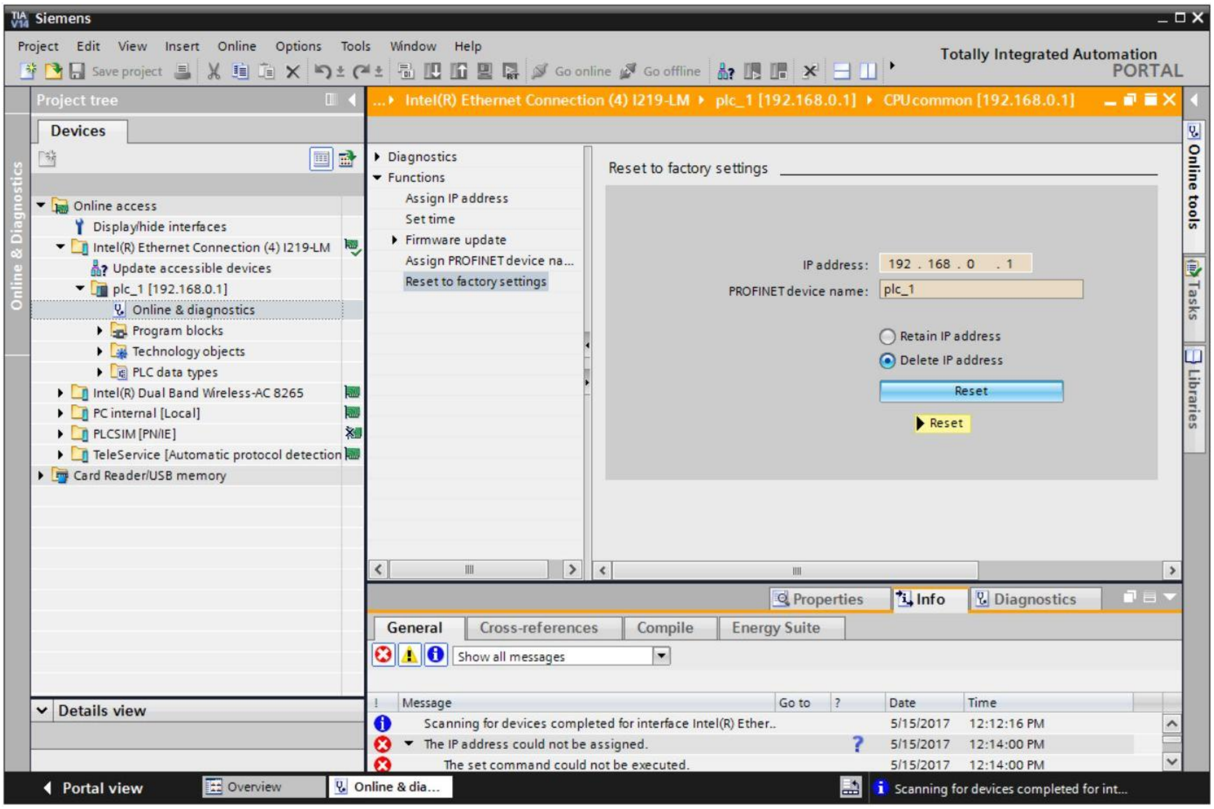


→ If the IP address was not successfully assigned, you will receive a message in the → "Info" window under → "General".

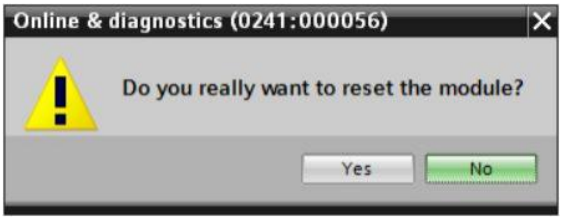


4.3.4 Restoring the factory settings of the CPU

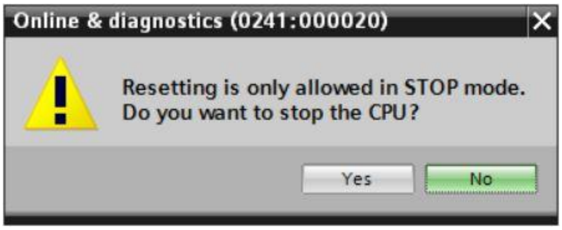
→ If the IP address could not be assigned, the program data on the CPU must be deleted. This is done by resetting the CPU. To reset the controller, select the → "Reset to factory settings" function and then → "Delete IP address" and click → "Reset".



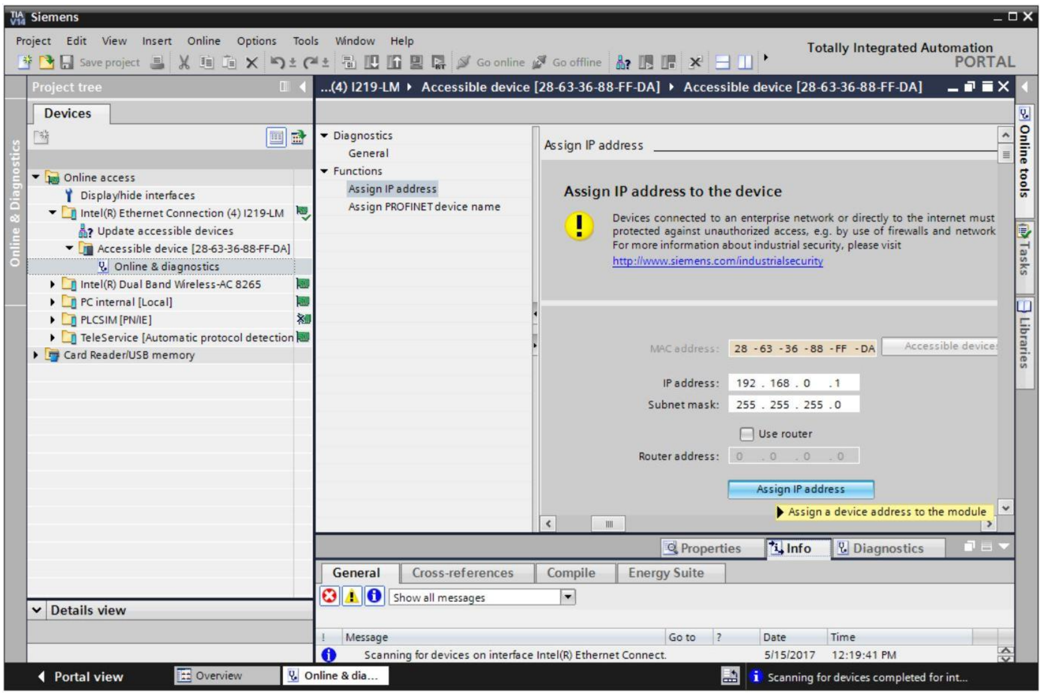
→ Confirm the prompt asking if you really want to reset the module with → "Yes".



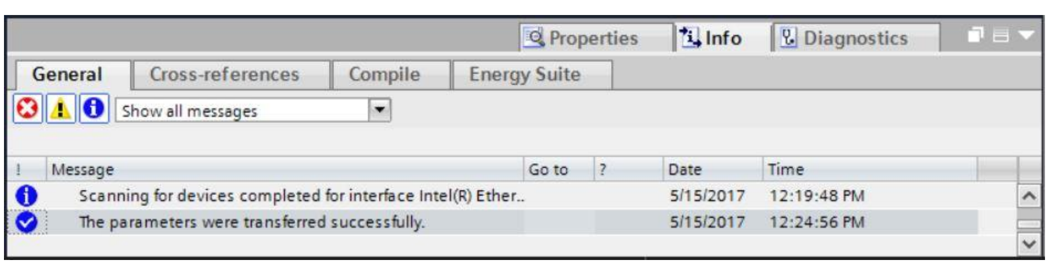
→ If necessary, stop the CPU. (→ "Yes")



→ Once the CPU has been reset, click → "Update accessible devices" again. The MAC address of the connected SIMATIC S7-1200 can now be seen. Select → "Online&Diagnostics" again. Select →"Assign IP address" under →"Functions". Enter the following IP address here (example): IP address: 192.168.0.1 Subnet mask 255.255.255.0. Click "Assign IP address" and this new address will be assigned to your SIMATIC S7-1200.



→ You will receive a message regarding successful transfer of parameters in the → "Info" → "General" window.



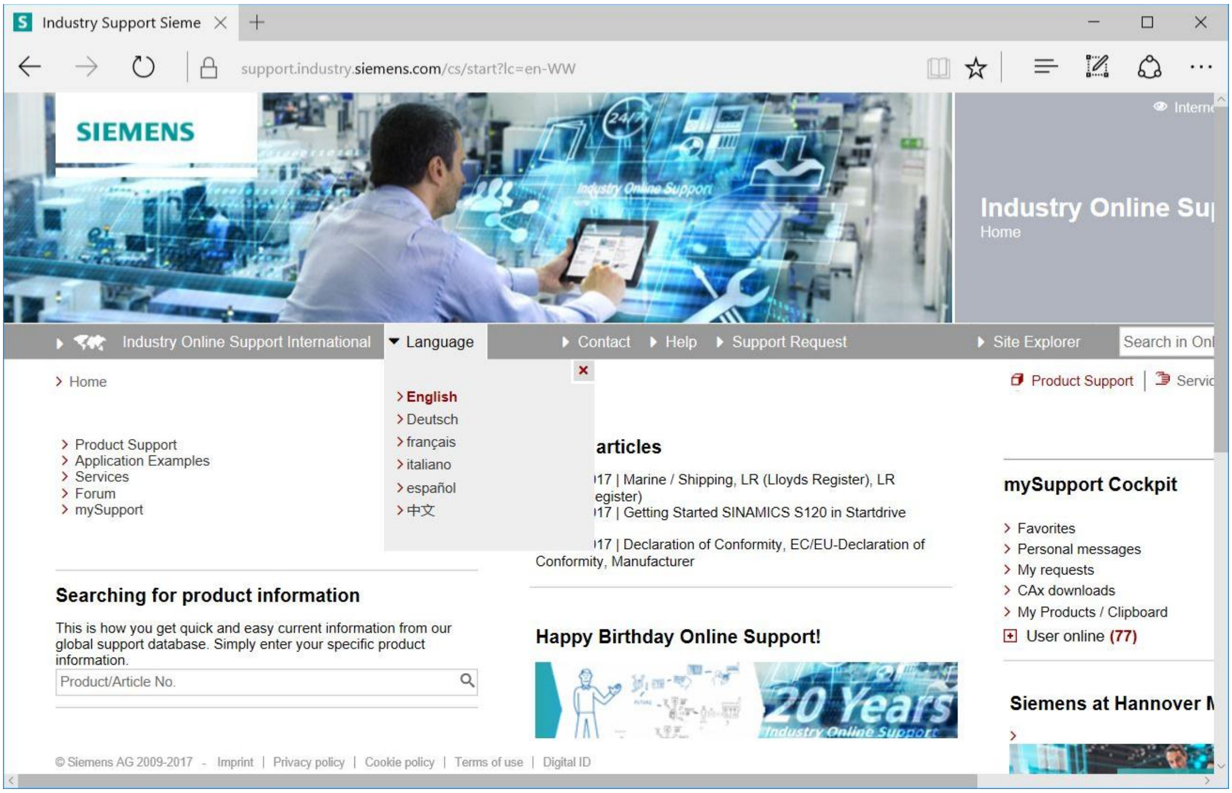
4.3.5 Downloading a firmware update from the SIEMENS Support website

You can download current firmware updates free of charge from the Industry Online Support of SIEMENS AG.

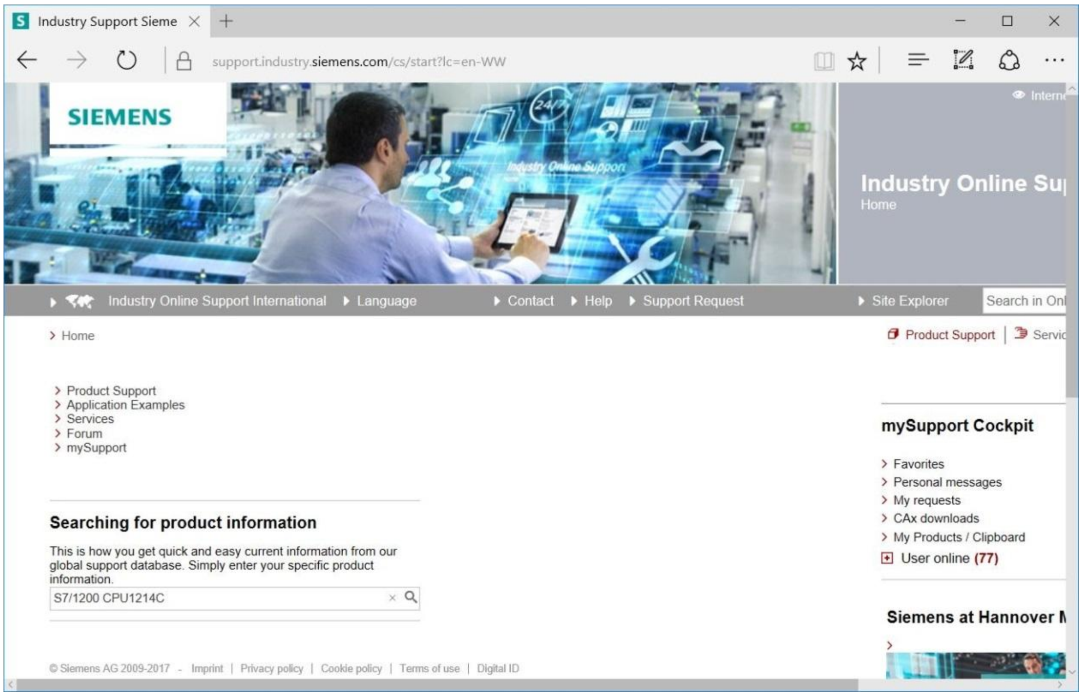
→ Open your choice of Internet browser and enter the address

→ support.automation.siemens.com

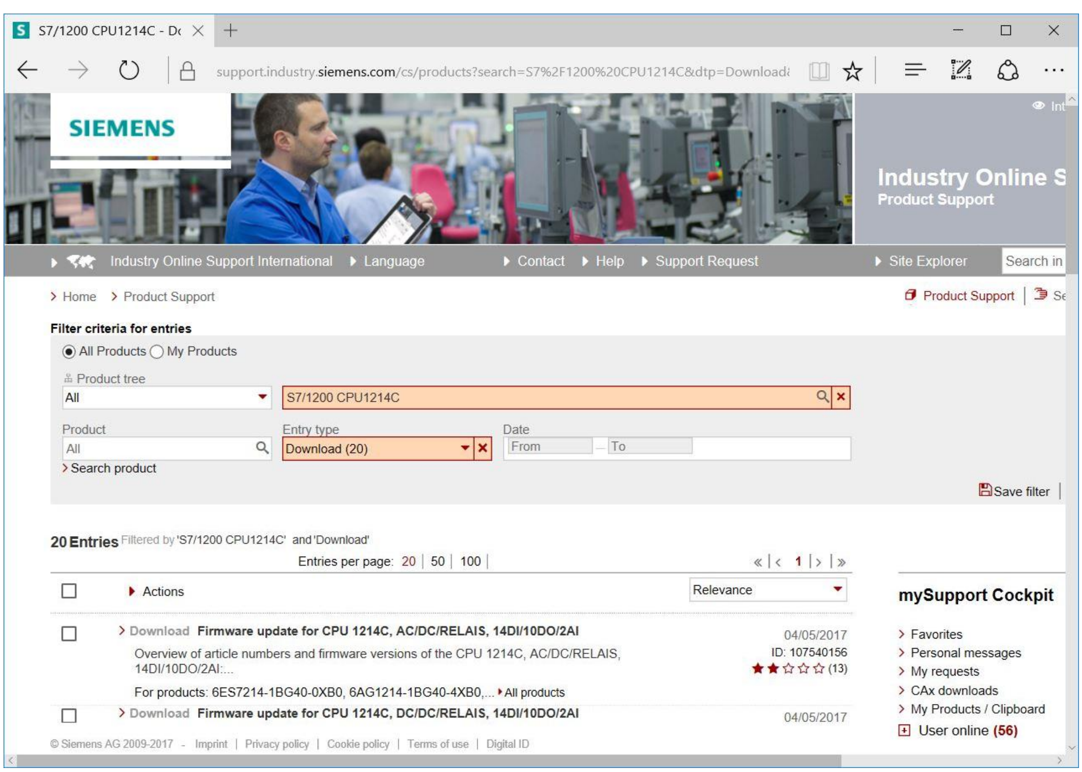
→ Select your desired language → "Language" → "English".



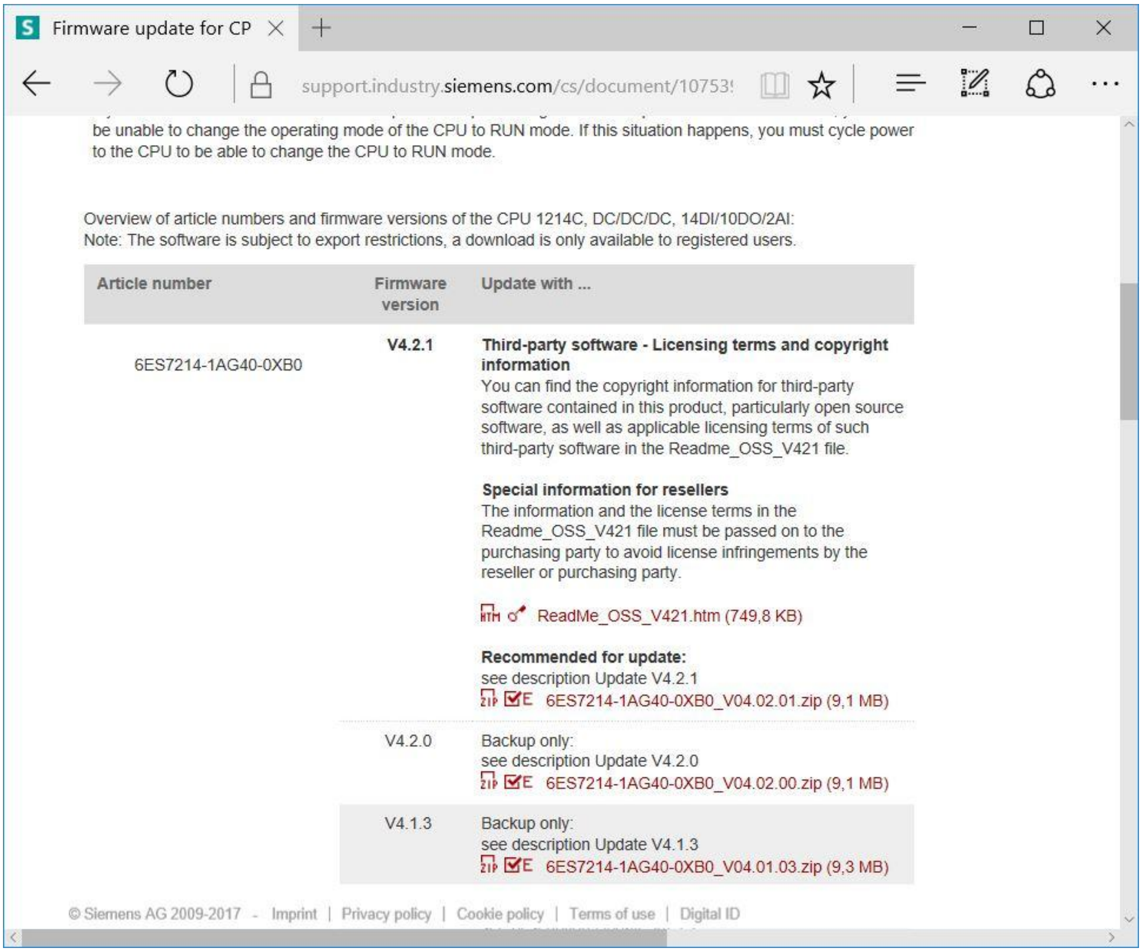
→ In "Searching for product information", enter the CPU for which you need a firmware update. For example: → "S7-1200 CPU1214C"



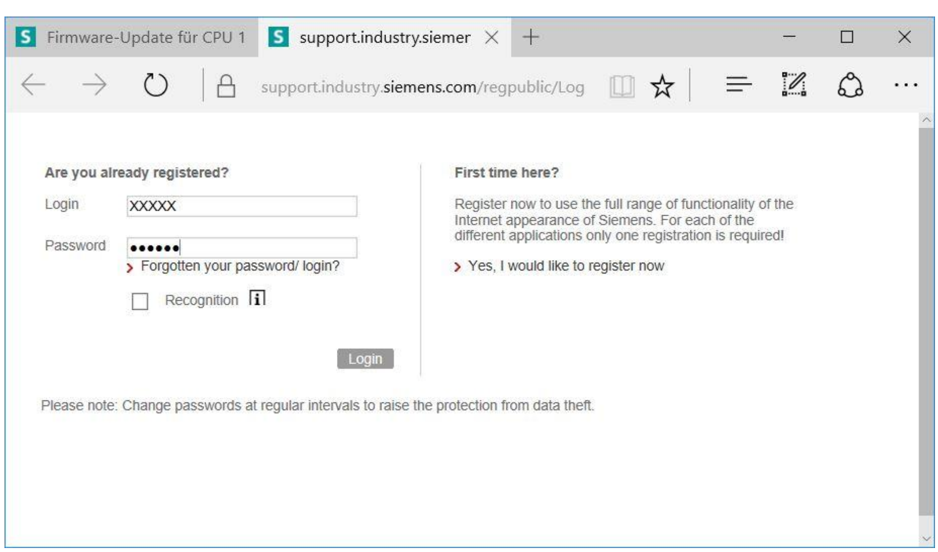
→ Under "Filter criteria for entries" select the "Entry type" → "Download" and click the entry with firmware updates for your CPU in the selection list.



→ Various updates are offered in the next window. Select the update recommended for the upgrade.

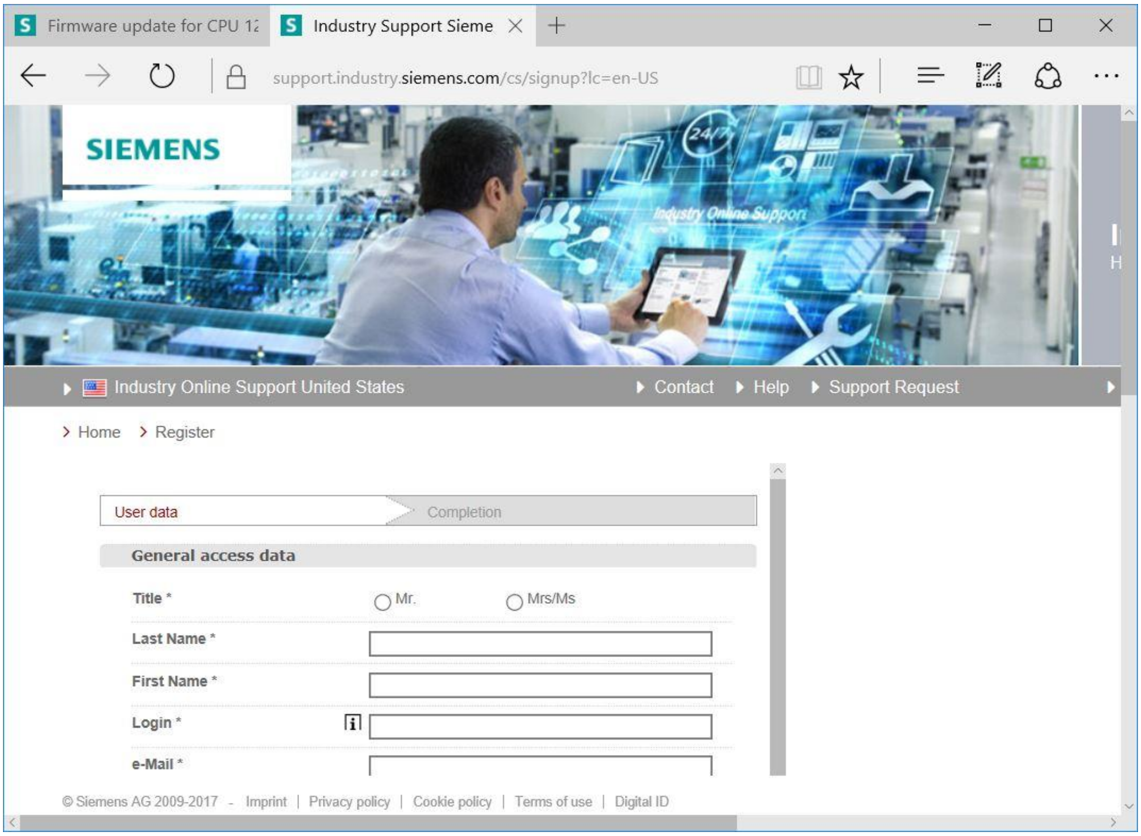


→ If you have not already registered, do so in the next window. (→ "Yes I would like to register now"), or if already registered – log in with your "Login" and "Password". (→ "Login")



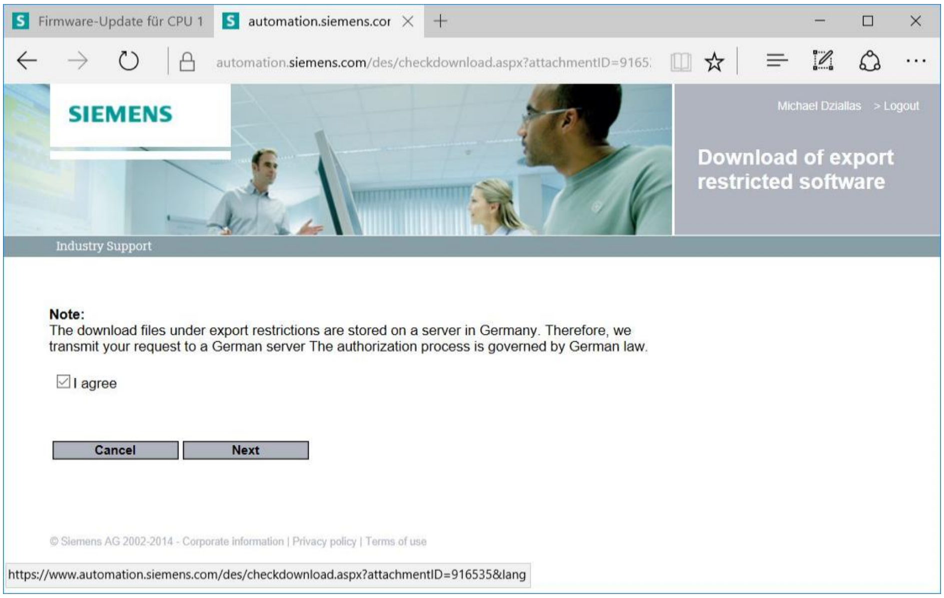
1

→ Enter the requested data for the registration, select the option "Download of export restricted software" and then save the registration. (→ ☑ → "Save")

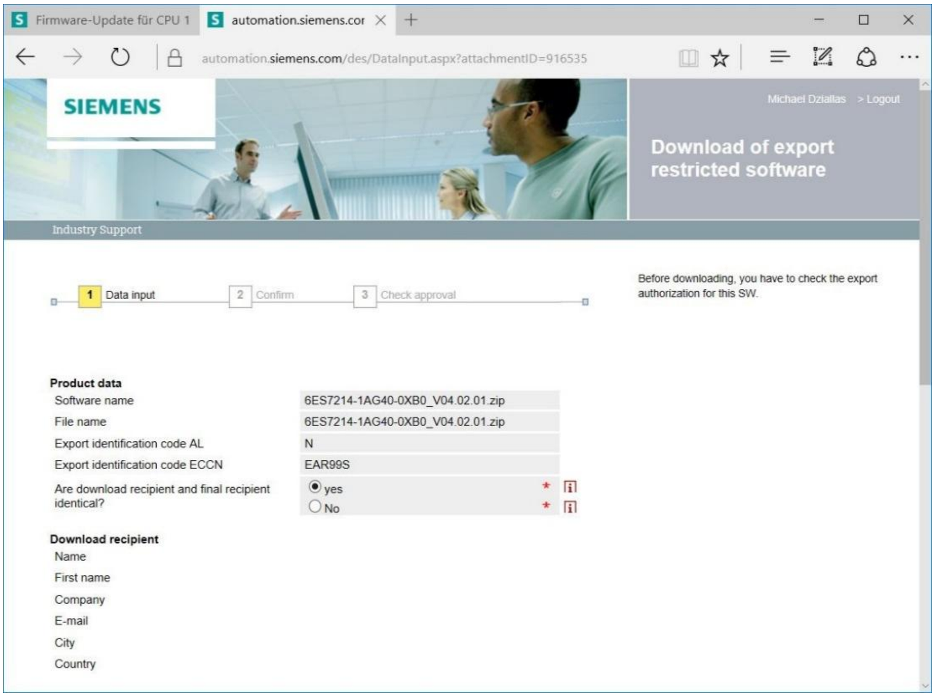


1

→ Go back to the Login. After login, consent to the procedure for export-restricted software. (→ ☑ I agree → "Next")

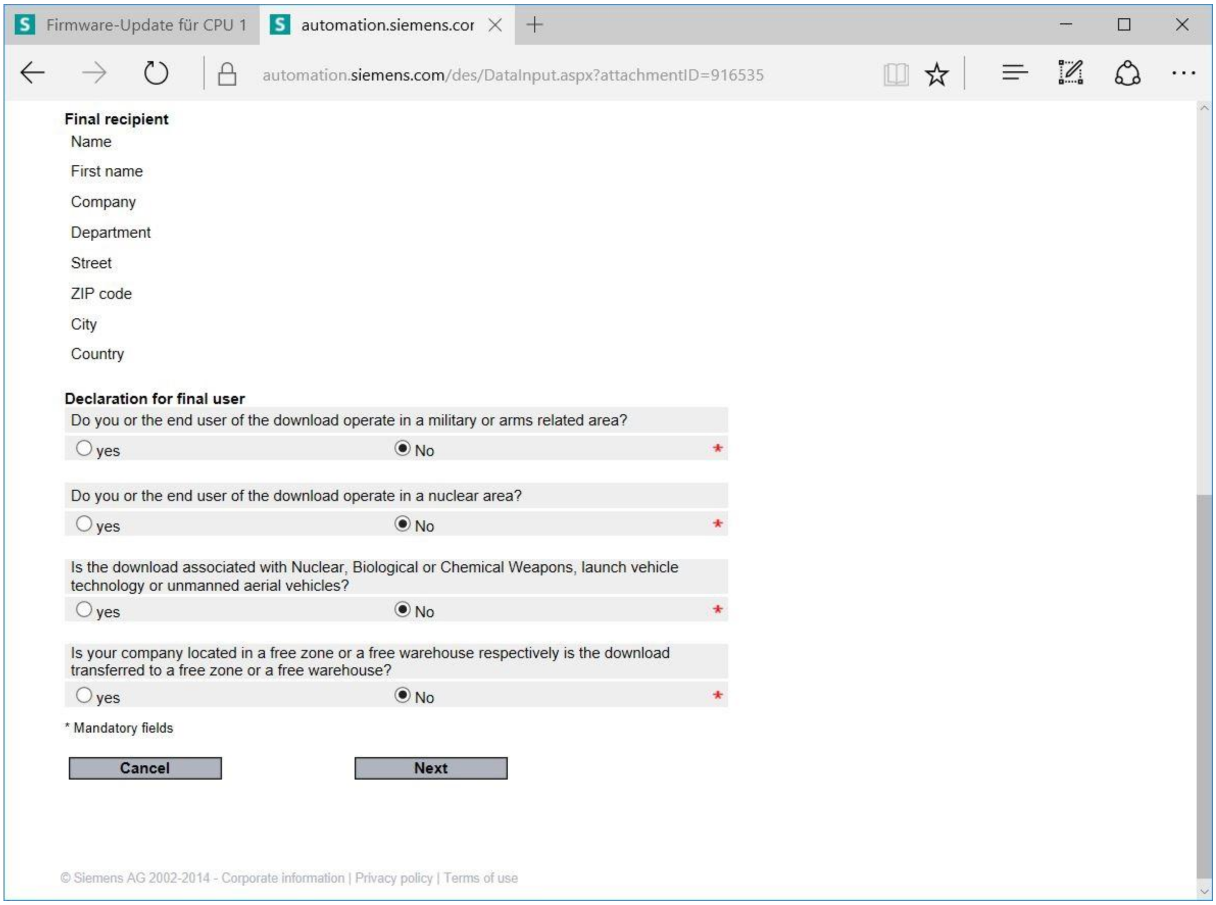


→ Enter the requested data for "Download recipient" and "Final recipient".

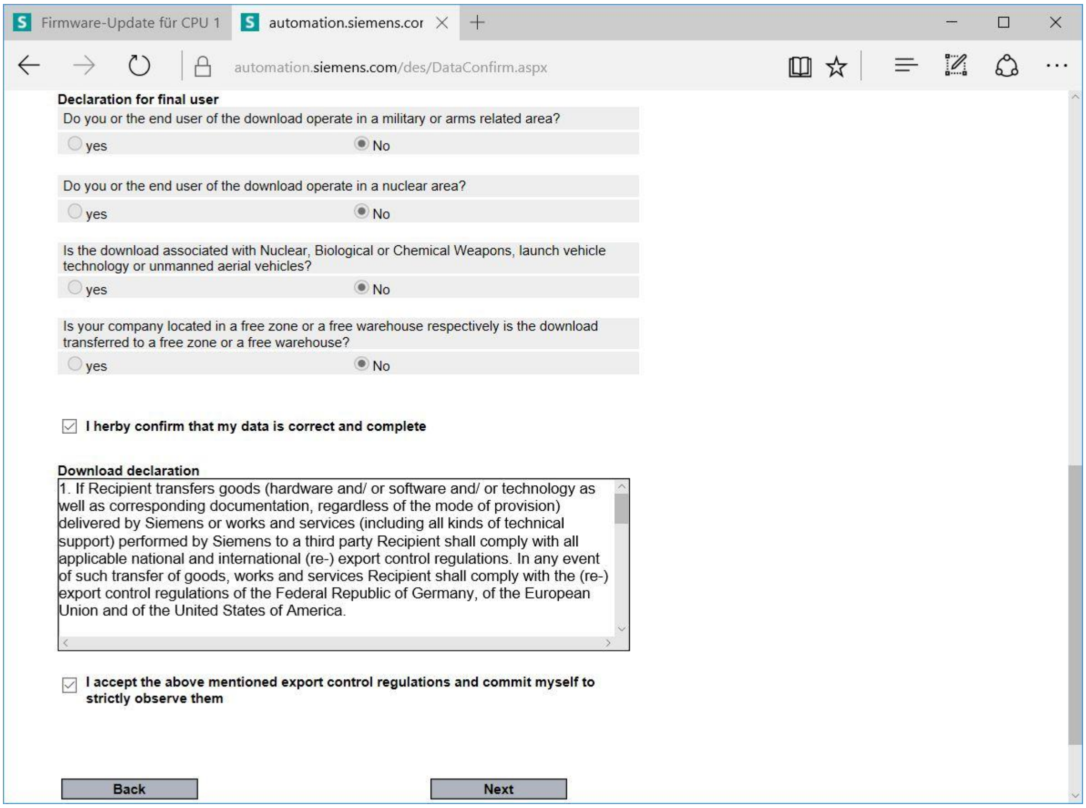


1

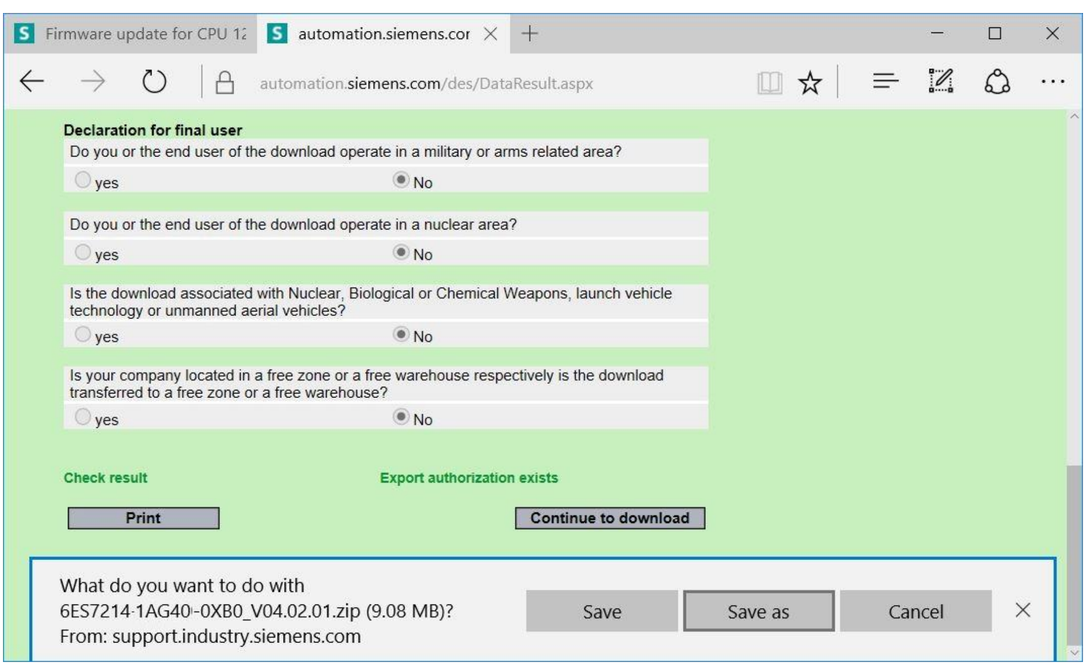
→ Enter the requested data for use of the software. (→ "Next")



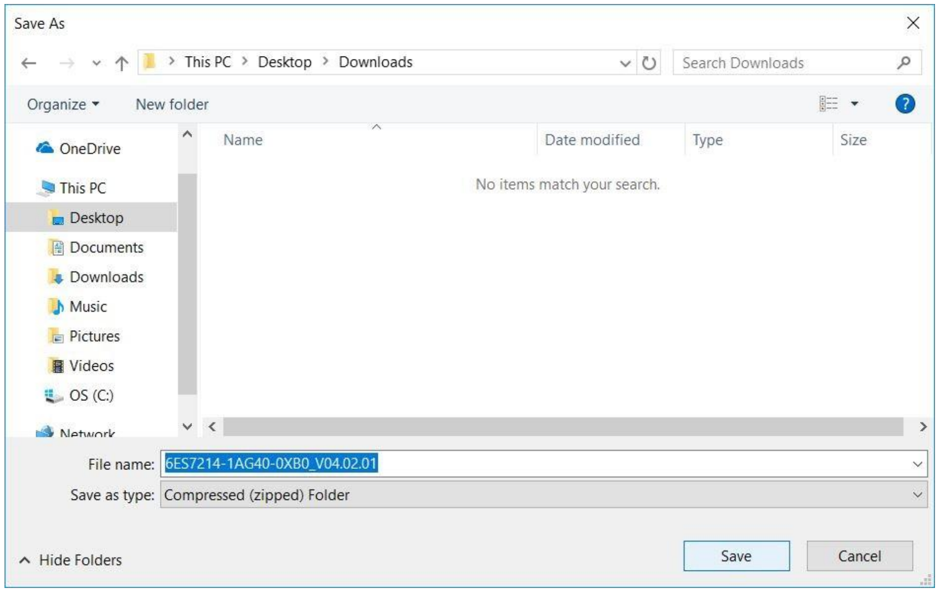
→ Confirm that your inputs are correct and accept the export regulations. (→ "Next")



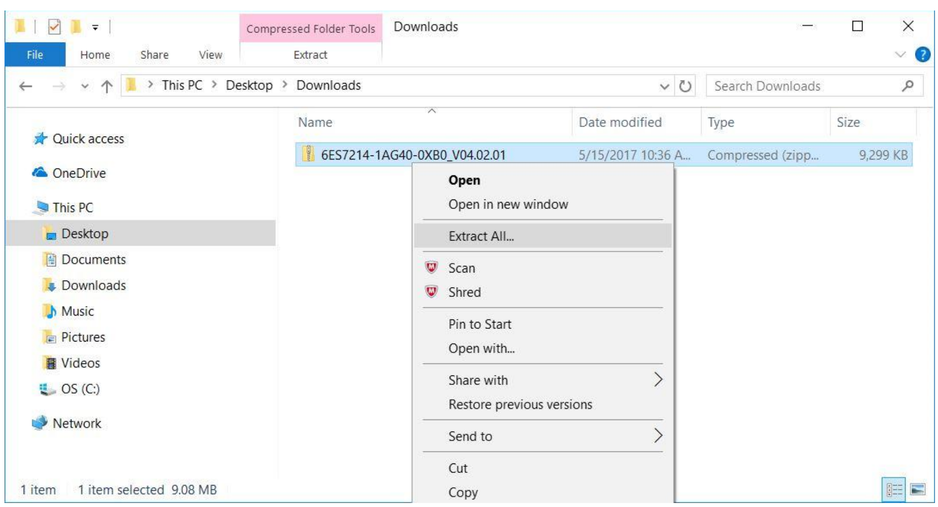
→ Once you have received confirmation of the export release, you can save the firmware update on your computer. (→ "Save as")



→ Enter the desired memory location on your computer and click → "Save".



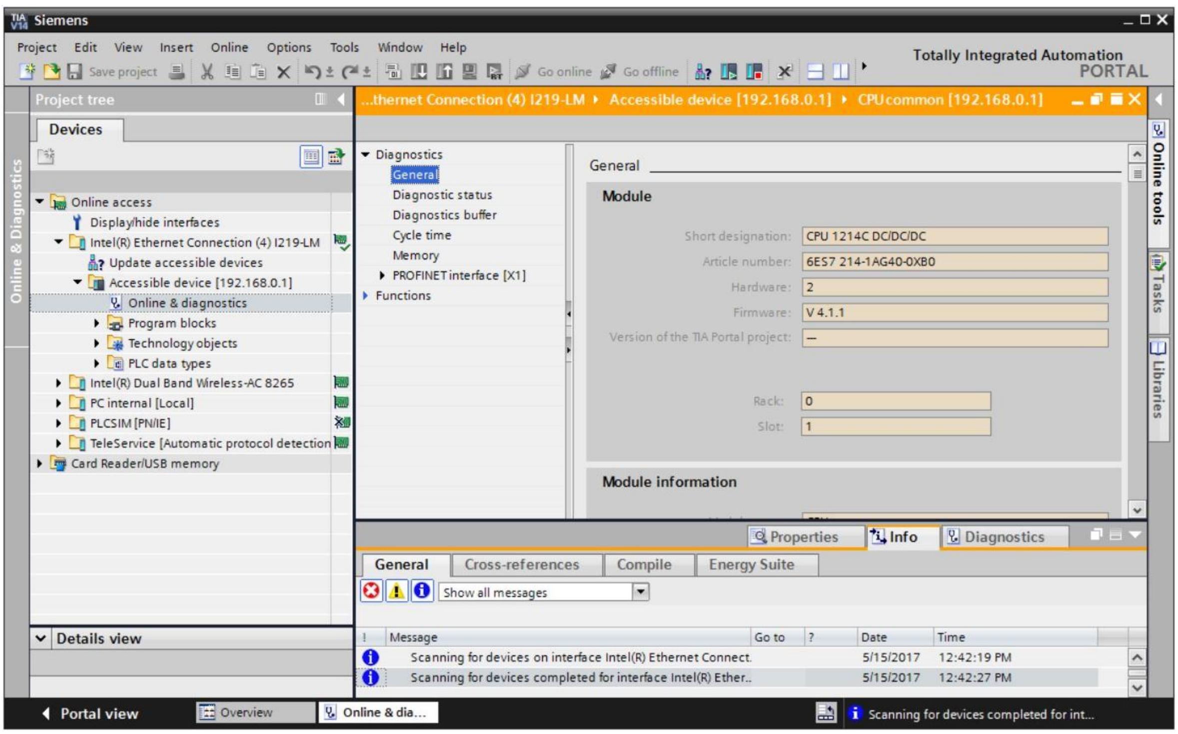
→ Click the downloaded compressed file in Windows Explorer and select → "Extract All".



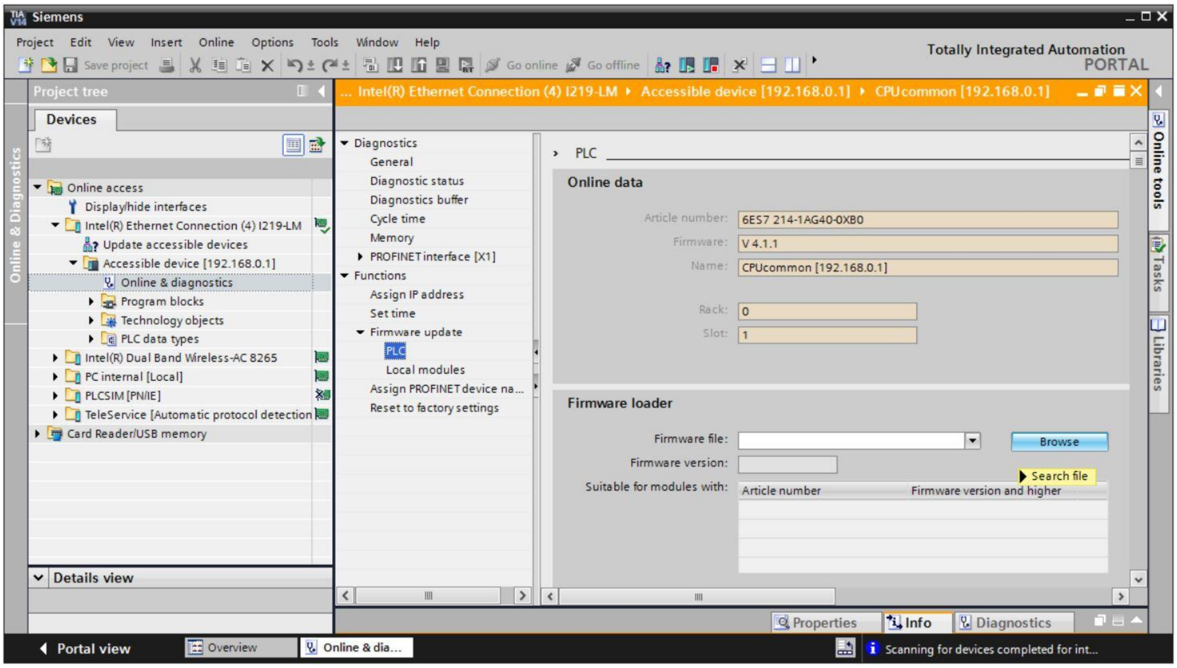
4.3.6 Firmware update of the CPU

The files with the firmware update can be imported into the "SIMATIC S7-1200" CPU as follows.

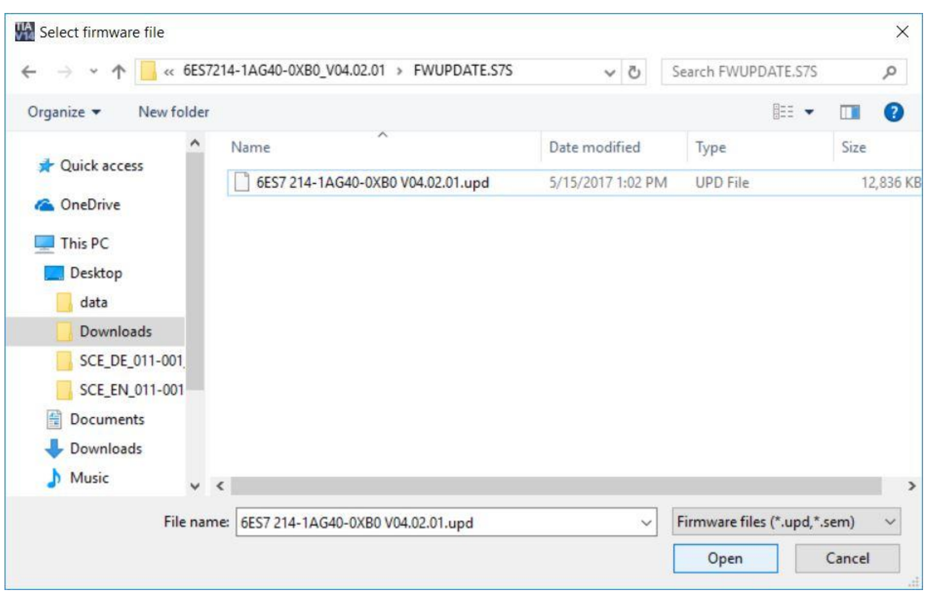
→ In the project tree under → "Online access", select the network adapter that was set previously. If you click → "Update accessible devices", you will see the IP address of the connected SIMATIC S7-1200. Select →"Online&Diagnostics". Under the "General" menu item, you can check the current firmware in your CPU in "Diagnostics".



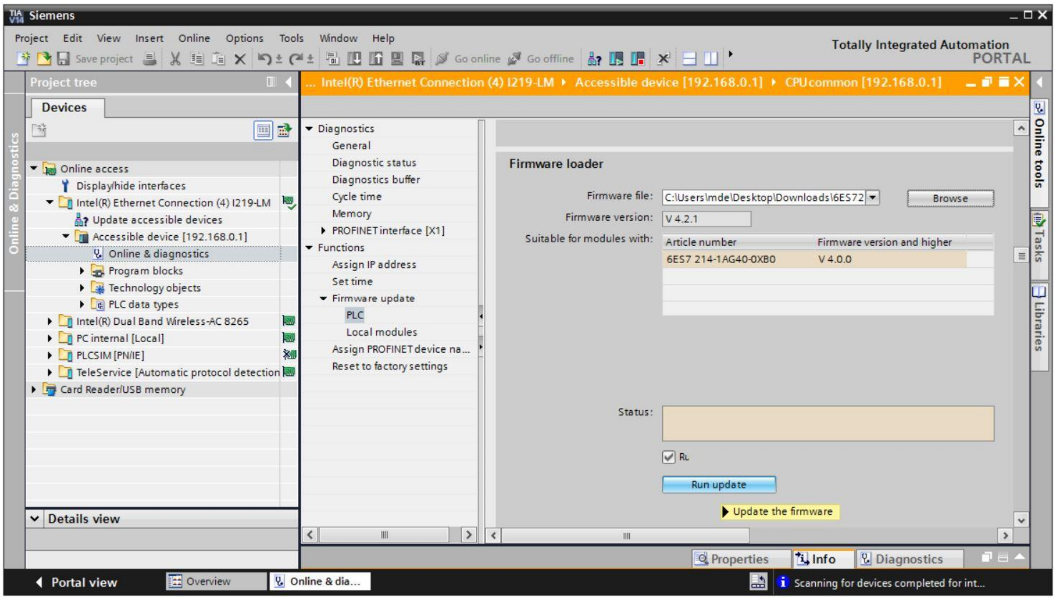
→ In the → "Functions" menu, change to → "Firmware update" → "PLC". In the → "Firmware loader" sub-item, click → "Browse".



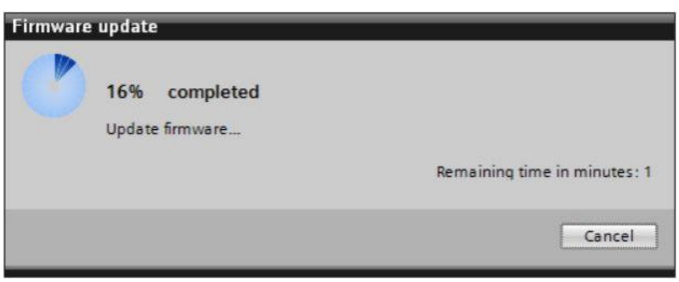
→ Select the previously downloaded and extracted firmware file → "6ES7 ***-*****-****.upd" on your computer and click → "Open".

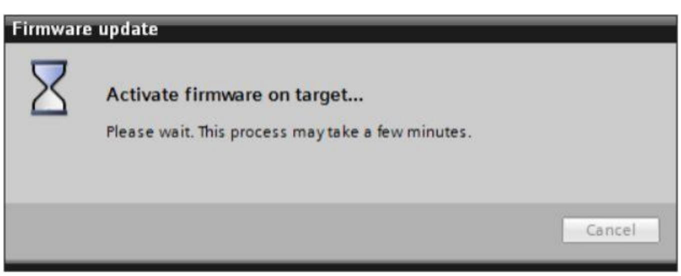


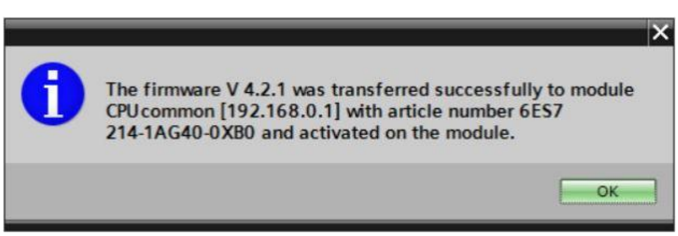
→ The following dialog indicates whether your firmware file is compatible with your CPU. Now start the update. (→ "Run update")



→ The progress of the update and its successful completion are indicated with the following dialogs. Click → "OK" to confirm.







5 Additional information

You can find additional information as an orientation aid for initial and advanced training, for example, Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software/firmware, at the following link:

siemens.com/sce/s7-1200

Preview "Additional information"

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- ↗ TIA Portal Videos
- ↗ TIA Portal Tutorial Center
- › Getting Started
- ↗ Programming Guideline
- ↗ Easy Entry in SIMATIC S7-1200
- › Download Trial Software/Firmware
- ↗ Technical Documentation SIMATIC Controller
- ↗ Industry Online Support App
- ↗ TIA Portal, SIMATIC S7-1200/1500 Overview
- ↗ TIA Portal Website
- ↗ SIMATIC S7-1200 Website
- ↗ SIMATIC S7-1500 Website

Notes

1

Notes

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

2

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is available at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, get in touch with your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

Table of contents

2

2

Unspecified Hardware Configuration – for a SIMATIC S7-1200

1 Goal

In this chapter, you will first learn how to ***create a project***. Next you will be shown how you can use the ***TIA Portal*** to detect ***hardware*** already installed and add it to a project. This hardware will then be configured.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

You do not need any previous knowledge from other chapters to successfully complete this chapter. You only need an S7-1200 controller and a PC with the STEP 7 Basic V14 (TIA Portal V14) software.

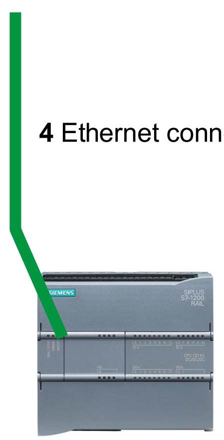
3 Required hardware and software

1. Engineering station: requirements include hardware and operating system
   (for additional information, see Readme on the TIA Portal Installation DVDs)
2. SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1
3. SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1
4. Ethernet connection between engineering station and controller

2 SIMATIC STEP 7 Basic (TIA Portal) as of V14 SP1

1 Engineering Station

4 Ethernet connection



3 SIMATIC S7-1200 controller

2

4 Theory

4.1 SIMATIC S7-1200 automation system

The SIMATIC S7-1200 automation system is a modular microcontroller system for the lower performance range.

A comprehensive range of modules is available to optimally adapt the system to the automation task.

The S7 controller consists of a power supply and a CPU with integrated inputs and outputs or additional input and output modules for digital and analog signals.

If necessary, communication processors and function modules are also used for special tasks such as stepper motor control.

The programmable logic controller (PLC) uses the S7 program to monitor and control a machine or process. In doing so, the IO modules are scanned in the S7 program using input addresses (%I) and addressed using output addresses (%Q).

The system is programmed with the TIA Portal Basic or Professional software.

4.1.1 Range of modules

The SIMATIC S7-1200 is a modular automation system and offers the following range of modules:

Central processing units (CPUs) with different performance, integrated inputs/outputs, and PROFINET interface (e.g. CPU 1214C)

Power supply module (PM) with input 120/230 V AC, 50 Hz / 60 Hz, 1.2 A / 0.7 A and output 24 V DC / 2.5 A

Signal boards (SBs) for adding analog or digital inputs/outputs, in which case the size of the CPU remains unchanged. (Signal boards can be used with CPUs 1211C / 1212C and 1214C.)

Signal modules (SMs) for digital and analog inputs and outputs (a maximum of 2 SMs can be used for CPU 1212C and a maximum of 8 SMs for CPU 1214C.)

Communication modules (CMs) for serial communication RS232 / RS 485 (Up to 3 CMs can be used for CPUs 1211C / 1212C and 1214C.)

Compact switch module (CSM) with 4x RJ45 sockets 10/100 Mbps

SIMATIC memory cards from 2 MB to 32 MB for storing program data and for easy exchange of CPUs during maintenance.

Note: *Only a single CPU (any type) with integrated digital inputs and digital outputs is needed for this module.*

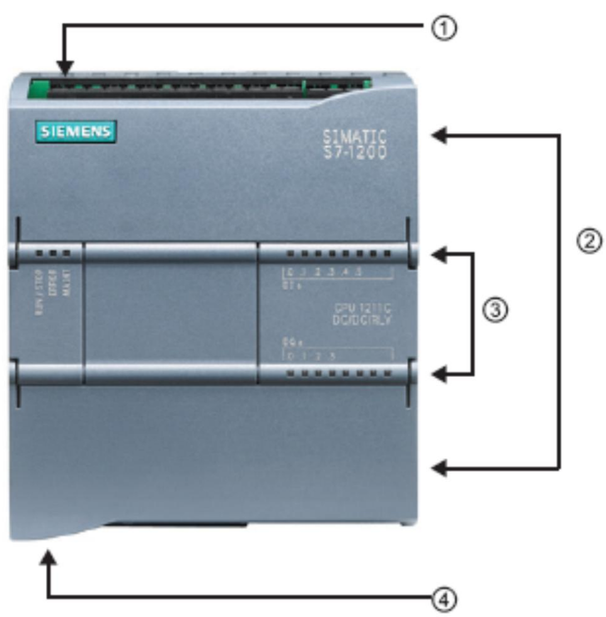
4.2 Operator control and display elements of the CPU 1214C DC/DC/DC

4.2.1 Front view of the CPU 1214C DC/DC/DC

With integrated power supply (24 V connection) and integrated inputs and outputs, the CPU 1214C DC/DC/DC is immediately ready for use without any other components.

The CPU has an integrated TCP/IP connection for communication with a programming device.

The CPU can thus communicate with HMI devices or other CPUs via an Ethernet network.



① 24 V connection

② Plug-in terminal block for user wiring (behind the cover flaps)

③ Status LEDs for the integrated IO and the operating state of the CPU

④ TCP/IP connection (on the underside of the CPU)

4.2.2 SIMATIC memory card (MC)

The optional **SIMATIC memory card (MC)** stores a program as well as data, system data, files and projects. It can be used for:

– Transferring a program to multiple CPUs

– Firmware update of CPUs, signal modules (SMs) and communication modules (CMs)

– Easy replacement of the CPU

4.2.3 Operating states of the CPU

The CPU can have the following three operating states:

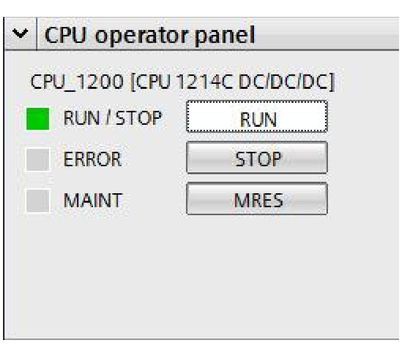
– In the **STOP** operating state, the CPU is not executing the program and you can download a project.

– In the **STARTUP** operating state, the CPU is starting up.

– In the **RUN** operating state, the program is cyclically executed.

The CPU does not have a physical switch for changing the operating state.

You use the button on the operator panel of the STEP 7 Basic software to change the operating state (**STOP** or **RUN**). The operator panel also contains an **MRES** button for performing a memory reset and displays the status LEDs of the CPU.



4.2.4 Status and error displays

The **RUN/STOP status LED** on the front side of the CPU indicates the current operating state of the CPU by the color of the display.

- **Yellow** light indicates **STOP** operating state.
- **Green** light indicates **RUN** operating state.
- A **flashing light** indicates **STARTUP** operating state.

There are two additional LEDs here: **ERROR** LED for indicating errors and **MAINT** LED for indicating that maintenance is required.

4.3 STEP 7 Basic V14 (TIA Portal V14) programming software

The STEP 7 Basic V14 (TIA Portal V14) software is the programming tool for the following automation systems:

– SIMATIC S7-1200

– Basic Panels

STEP 7 Basic V14 provides the following functions for automation of a system:

– Configuration and parameter assignment of the hardware

– Specification of the communication

– Programming

– Testing, commissioning and servicing with operational/diagnostic functions

– Documentation

– Creation of visualizations for SIMATIC Basic Panels using the integrated WinCC Basic software

– Support is provided for all functions through detailed online help.

4.3.1 Project

To implement a solution for an automation and visualization task, you create a project in the TIA Portal. A project in the TIA Portal contains the configuration data for the configuration and internetworking of devices as well as the programs and the configuration of the visualization.

4.3.2 Hardware configuration

The *hardware configuration* includes the configuration of the devices, consisting of the hardware of the automation system, the field devices on the PROFINET bus system and the hardware for visualization. The configuration of the networks specifies the communication between the various hardware components. Individual hardware components are *inserted in the hardware configuration* from catalogs.

The hardware of SIMATIC S7-1200 automation systems comprises the controller (CPU), the signal modules for input and output signals (SMs), the communication modules (CMs) and other special-purpose modules.

The signal modules and the field devices connect the input and output data of the process to be automated and visualized to the automation system.

The hardware configuration enables the downloading of automation and visualization solutions to the automation system and access to the connected signal modules by the controller.

4.3.3 Planning the hardware

Before you can configure the hardware, you must plan it (hardware planning). In general, you begin by selecting which controllers are needed and how many. You then select the communication modules and signal modules. The selection of signal modules is based on the number and type of inputs and outputs needed. As the final step, a power supply that ensures the necessary power supply must be selected for each controller or field device.

The functionality required and the ambient conditions are of vital importance for planning the hardware configuration. For example, the temperature range in the application area sometimes limits which devices are available for selection. Fail-safe operation might be another requirement.

The TIA Selection Tool (Select automation technology → TIA Selection Tool and follow the instructions) provides you support. Note: The TIA Selection Tool requires Java.

Note for online research: *If more than one manual is available, you should look for the description "Device Manual", "Product Manual" or simply "Manual" (as opposed to "Function Manual", "List Manual", "System Manual", etc.) in order to find the device specifications.*

4.3.4 TIA Portal – Project view and portal view

The TIA Portal has two important views. When started, the TIA Portal displays the portal view by default. This view makes getting started easier, especially for beginning users.

The portal view provides a task-oriented view of the tools for working on the project. Here, you can quickly decide what you want to do and open the tool for the task at hand. If necessary, a change to the project view takes place automatically for the selected task.

Figure 1 shows the portal view. At the bottom left, there is an option to switch between this view and the project view.

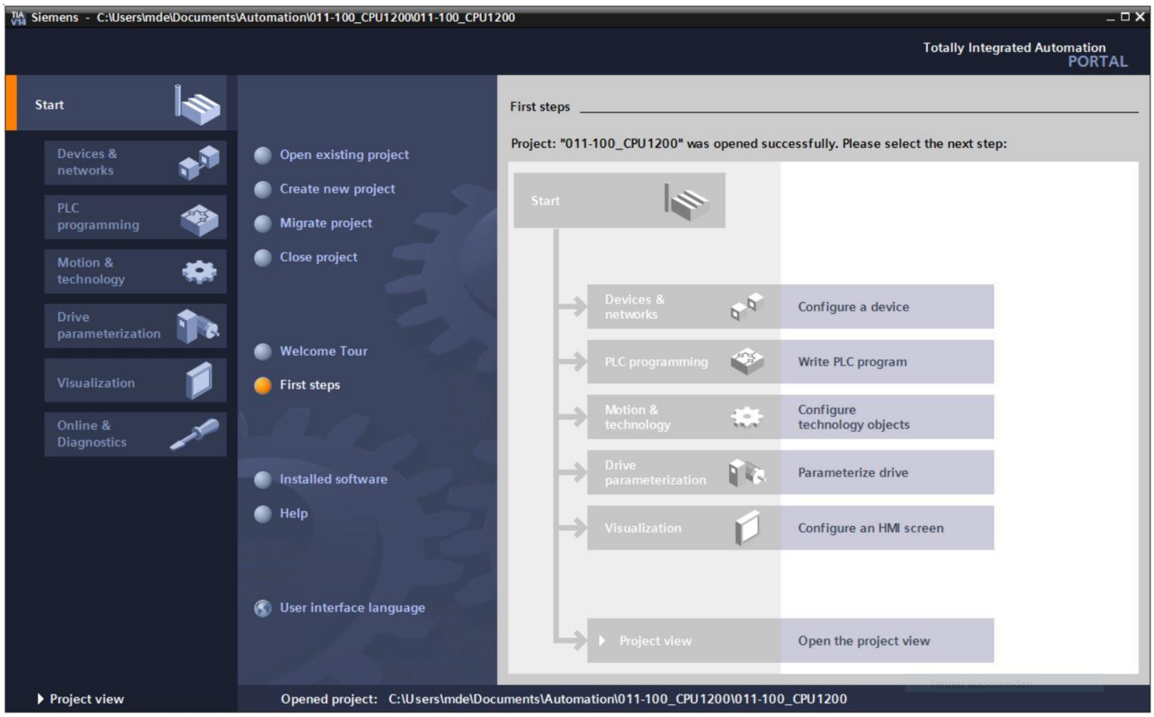


Figure 1: Portal view

The project view, as shown in Figure 2, is used for hardware configuration, programming, creation of the visualization and many other tasks.

By default, the project view displays the menu bar with the toolbars at the top, the project tree with all components of a project on the left and the so-called "task cards" with instructions and libraries, for example, on the right.

If an element (for example, the device configuration) is selected in the project tree, it is displayed in the center and can be worked on there.

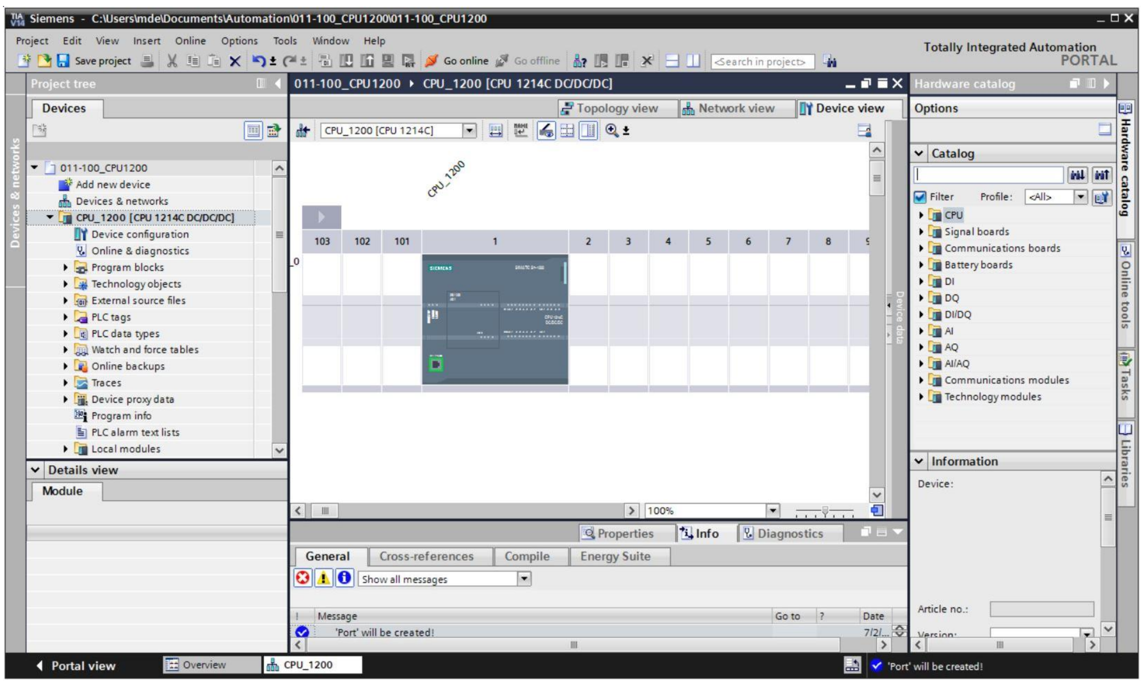
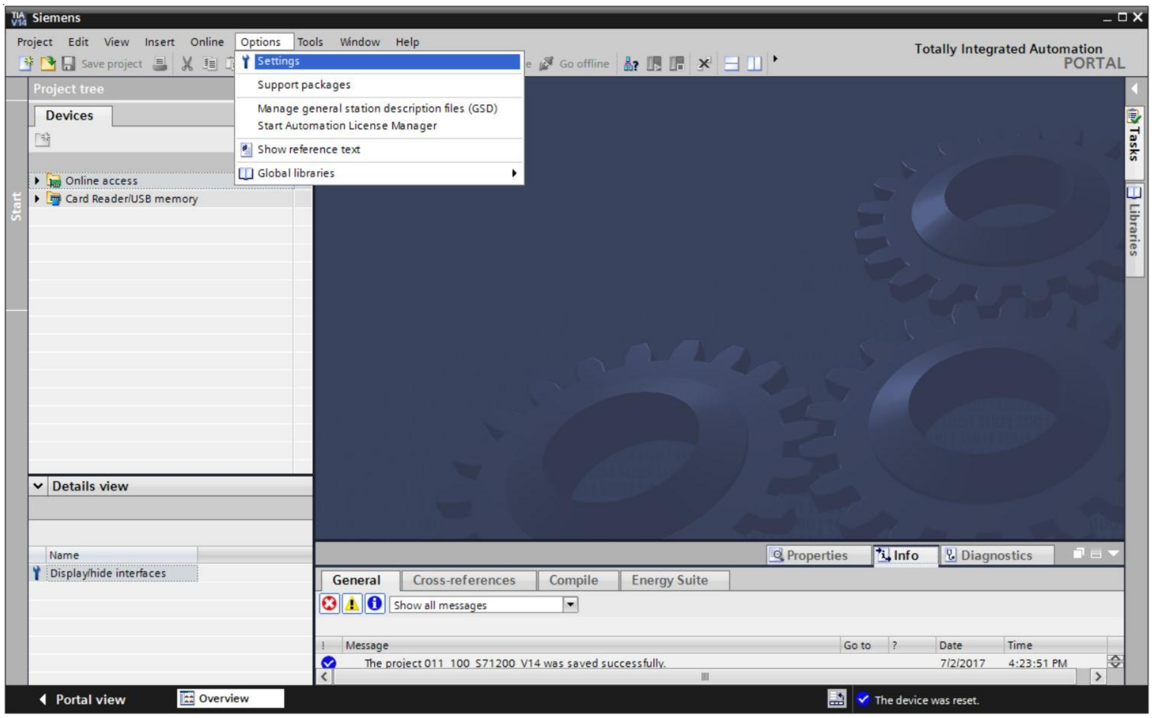


Figure 2: Project view

4.3.5 Basic settings for the TIA Portal

→ Users can specify their own default settings for certain settings in the TIA Portal. A few important settings are shown here.

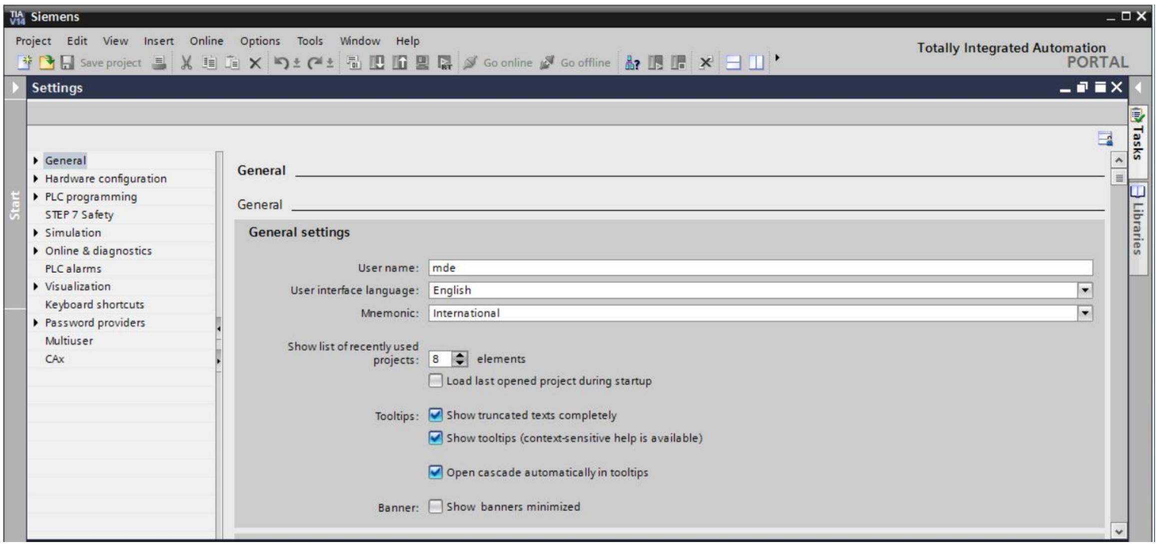
→ In the project view, select the → "Options" menu and then → "Settings".



2

→ One basic setting is the selection of the user interface language and the language for the program display. In the curriculums to follow, "English" will be used for both settings.

→ Under "General" in "Settings", select → "User interface language → English" and "Mnemonic → International".



Note: *These settings can always be changed.*

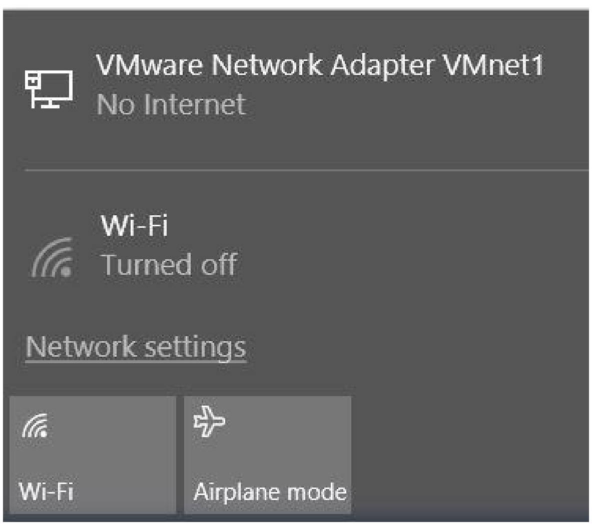
4.3.6 Set the IP address on the programming device

To program the SIMATIC S7-1200 controller from the PC, the programming device or a laptop, you need a TCP/IP connection or an optional PROFIBUS connection.

For the PC and SIMATIC S7-1200 to communicate with each other via TCP/IP, it is important that the IP addresses of both devices match.

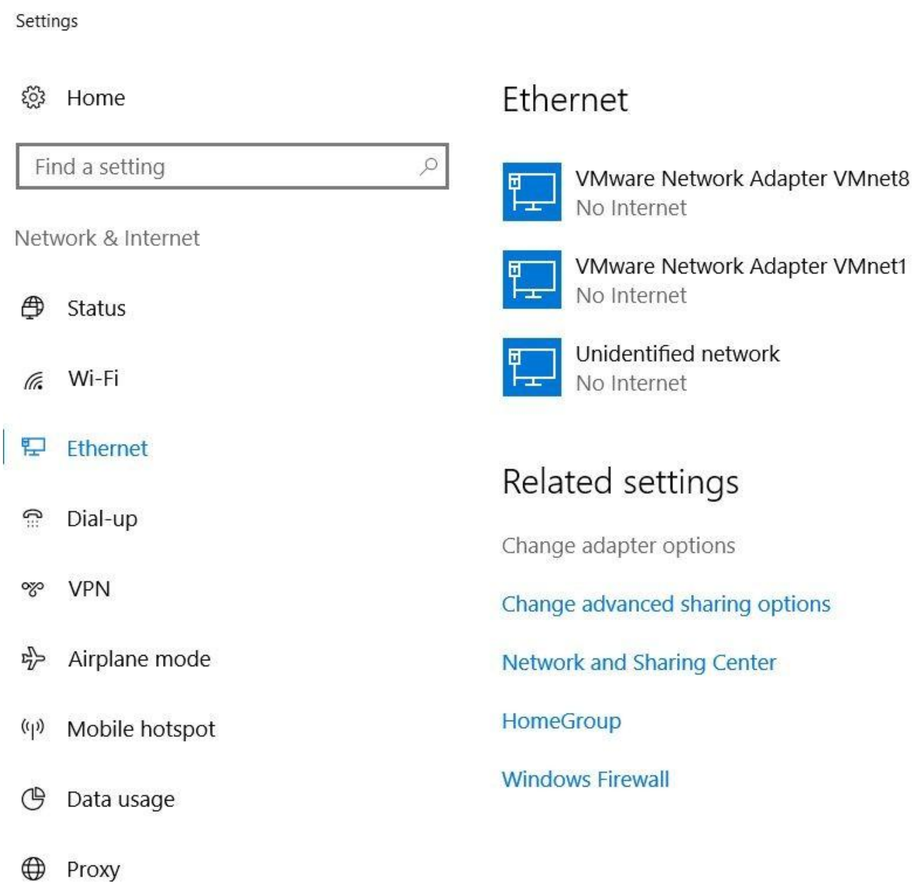
First, we show you how to set the IP address of a PC with Windows 7 operating system.

→ Locate the network icon in the taskbar at the bottom  and click → "Open Network and Sharing Center".

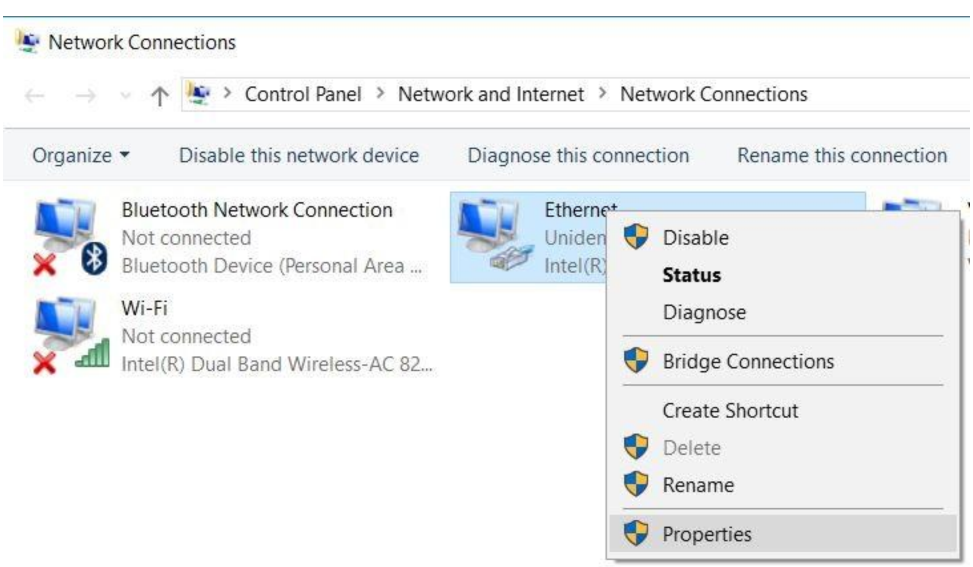


2

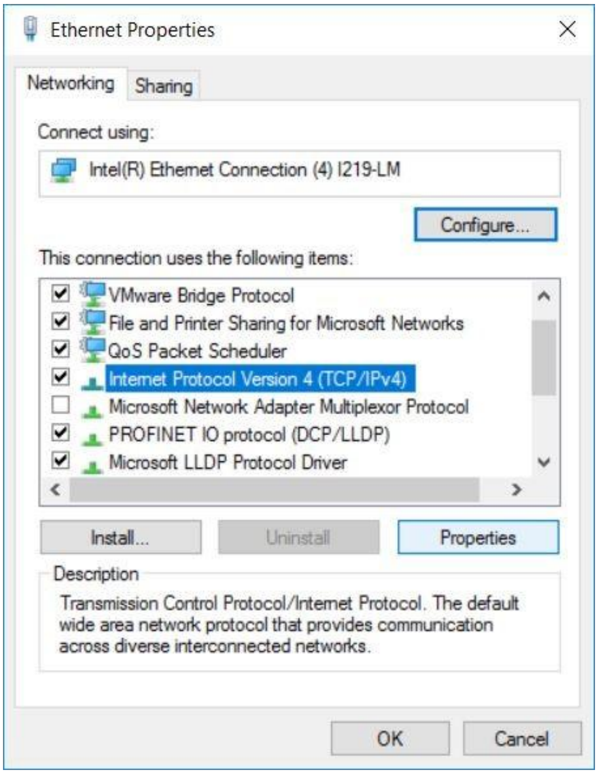
→ In the open Network and Sharing Center window, click → "Change adapter settings".



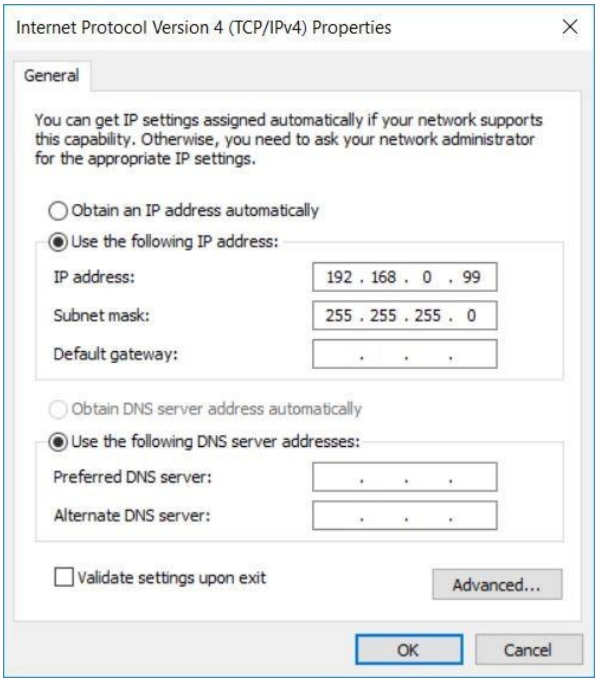
→ Select the desired → "Local Area Connection" that you want to use to connect to the controller and click → "Properties".



→ Next, select → "Properties" for → "Internet Protocol Version 4 (TCP/IP)".



→ You can use the following address, for example → IP address: 192.168.0.99 → Subnet mask 255.255.255.0 and accept the settings (→ "OK")



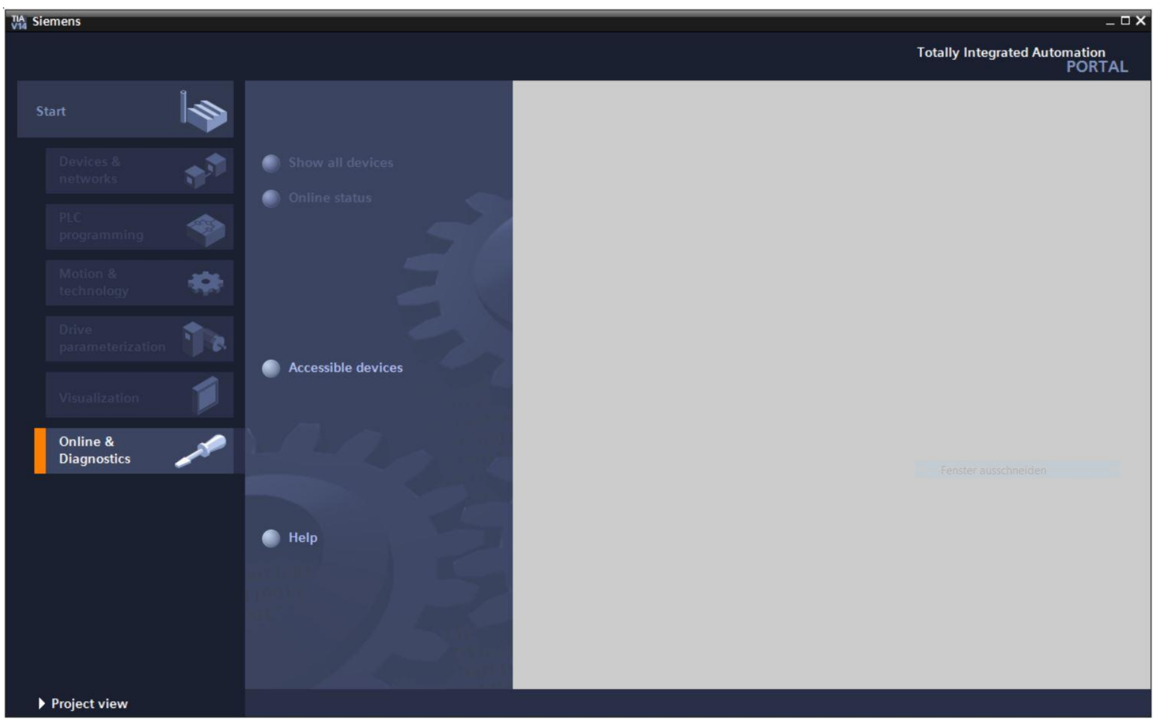
2

4.3.7 Set the IP address in the CPU

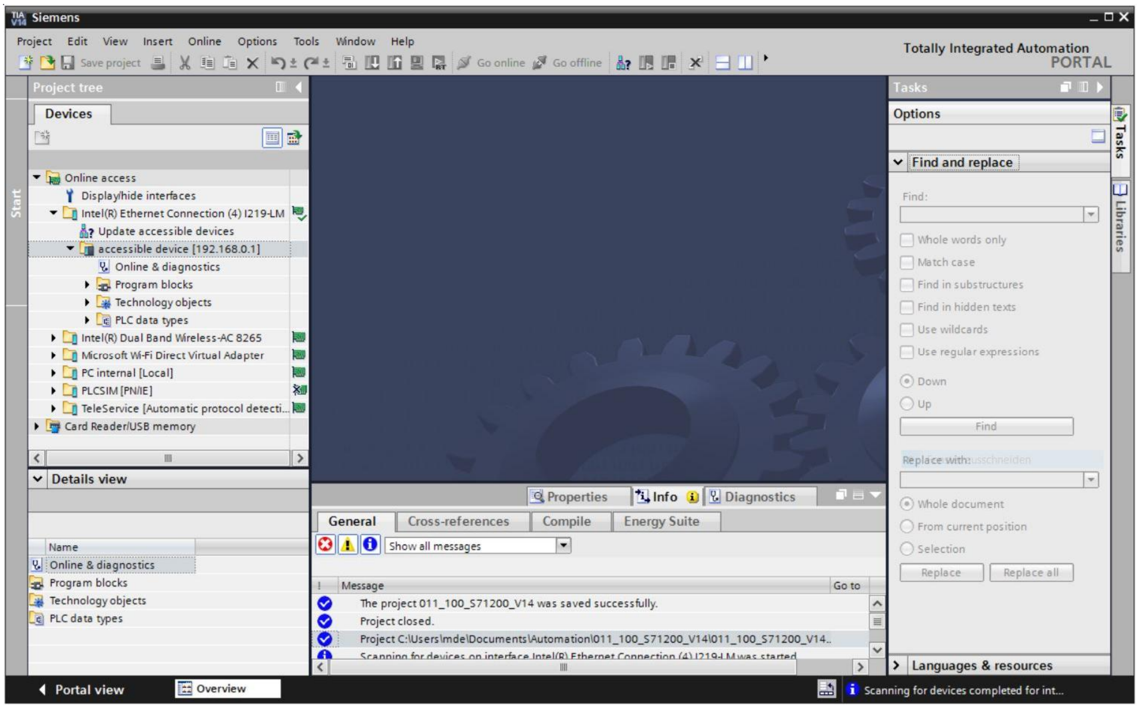
The IP address of SIMATIC S7-1200 is set as follows.

→ Select the Totally Integrated Automation Portal for this, which is opened here with a double-click (→ TIA Portal V14)

→ Select → "Online & diagnostics", and open the →"project view".

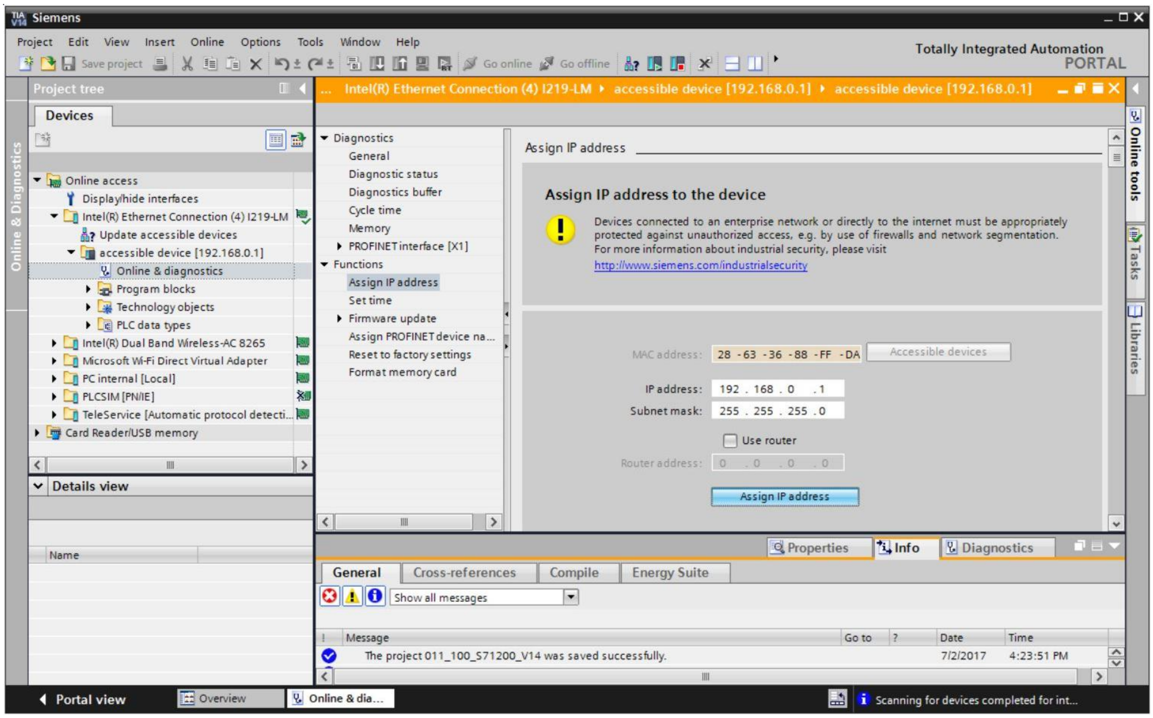


→ In the project tree under → "Online access", select the network adapter that was set previously. If you click → "Update accessible devices" here, you see the IP address (if previously set) or the MAC address (if IP address not yet assigned) of the connected SIMATIC S7-1200. Select → "Online & diagnostics" here.

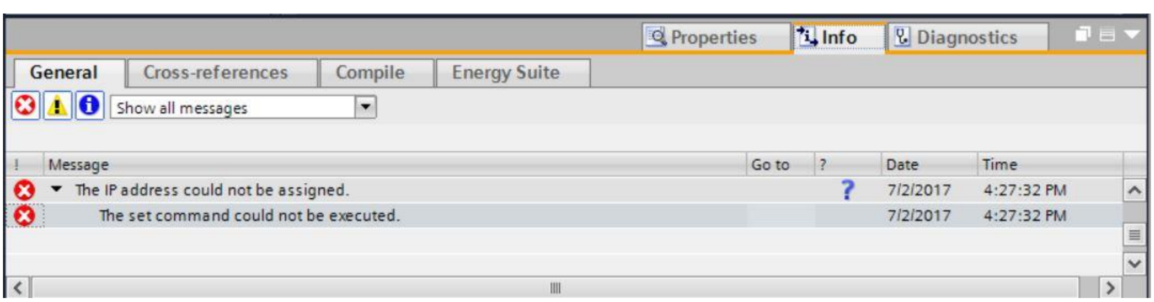


2

→ Under→ "Functions", you now find the → "Assign IP address" item. Enter the following IP address here (example): → IP address: 192.168.0.1 → Subnet mask 255.255.255.0. Click → "Assign IP address" and this new address will be assigned to your SIMATIC S7-1200.

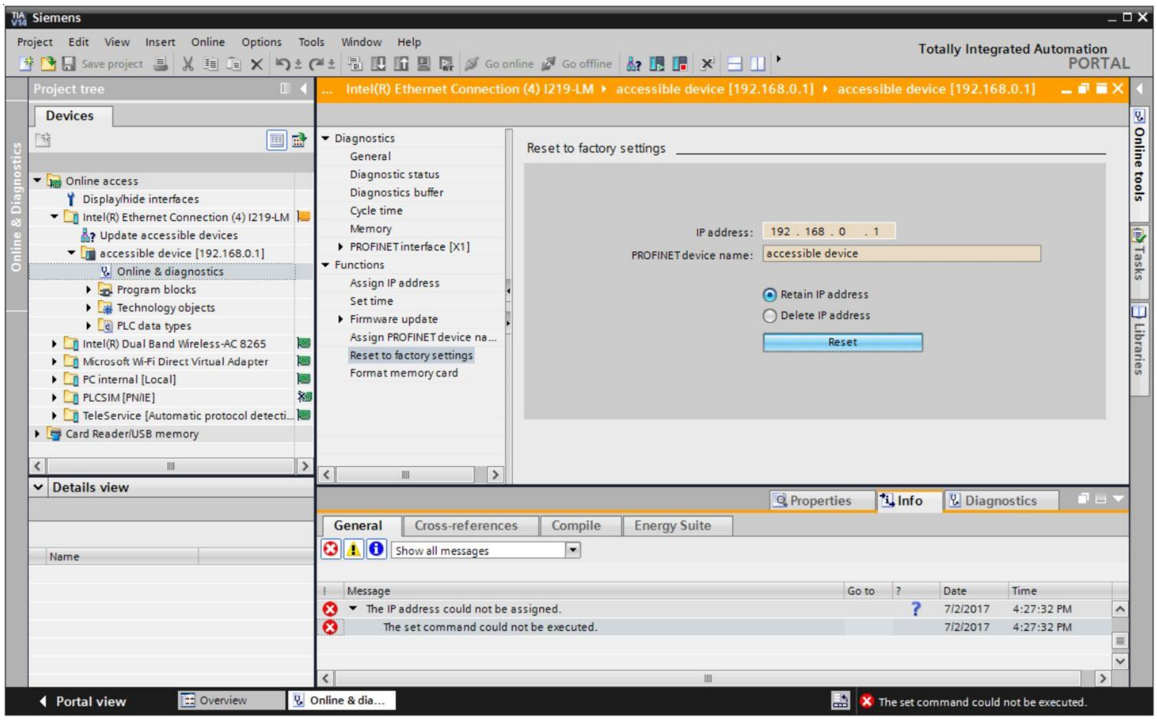


→ If the IP address was not successfully assigned, you will receive a message in the → "Info" window under → "General".

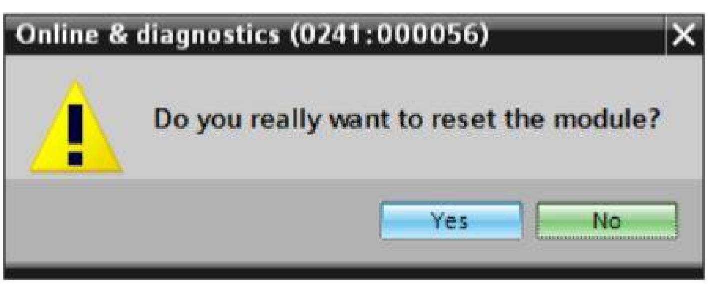


4.3.8 Restore the factory settings of the CPU

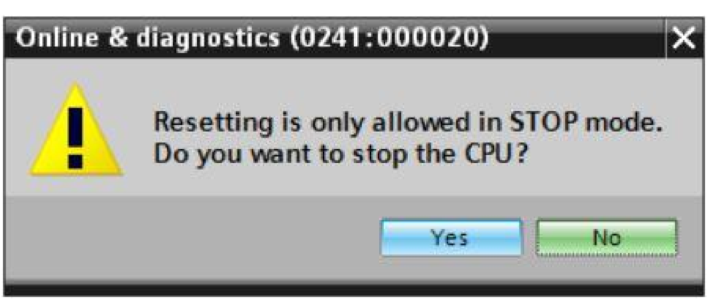
→ If the IP address could not be assigned, the program data on the CPU must be deleted. This is done by resetting the CPU. To reset the controller, select the → "Reset to factory settings" function and click → "Reset".



→ Confirm the prompt asking if you really want to reset the module with → "Yes".



→ If necessary, stop the CPU. (→ "Yes")



2

5 Task

Create a project and add the modules of the existing hardware (here: Trainer Package **SIMATIC S7-1200 with CPU 1214C)** by using the automatic hardware detection of the **TIA Portal**. The following modules must be detected:

– SIMATIC S7-1200, CPU 1214C DC/DC/DC
(Order number: 6ES7 214-1AG40-0XB0)

– 1X SIMATIC S7-1200, signal board ANALOG OUTPUT SB1232, 1 AO
(Order number: 6ES7 232-4HA30-0XB0)

6 Planning

Because this is a new system, a new project must be created.

The hardware for this project is already specified by the existing hardware (here: SIMATIC S7-1200 Trainer Package). Therefore, a selection does not have to be made. Instead, the listed modules of the Trainer Package are detected directly. The order numbers (see Task or Table 1) can be used for checking purposes.

The Ethernet interface must be set for the configuration of the CPU. For the digital and analog inputs and outputs, the address areas corresponding to Table 1 will be set.

Module	Order number	Slot	Address area
CPU 1214C DC/DC/DC	6ES7 214-1AG40-0XB0	1	DI 0.0 -1.5 DQ 0.0 - 1.1 AI 64 / 66
SB1232, 1 AO	6ES7 232-4HA30-0XB0		AO 64

Table 1: Overview of the planned configuration

As the final step, the hardware configuration must be compiled and downloaded. Any errors present can be detected during compiling and incorrect modules can be detected when the controller is started *(only possible when hardware is present and installed identically)*.

The tested project must be saved and archived.

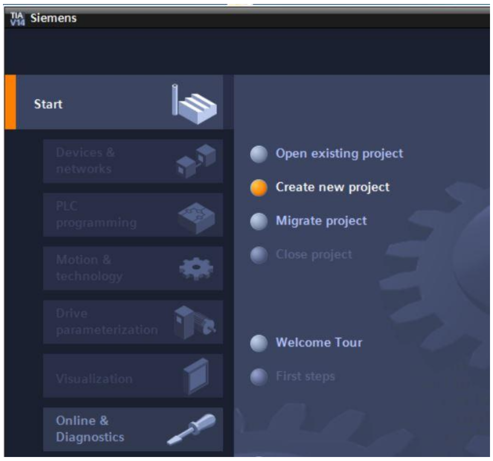
7 Structured step-by-step instructions

You can find instructions on how to carry out planning below. If you already have the relevant previous knowledge, it will be sufficient to focus on the numbered steps. Otherwise, simply follow the steps in the instructions.

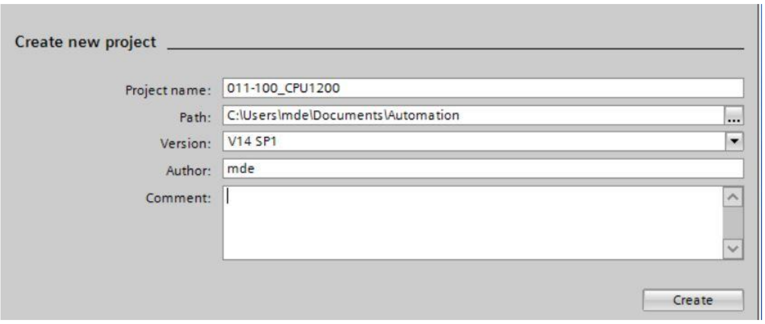
7.1 Create a new project

→ Select the Totally Integrated Automation Portal for this, which is opened here with a double-click (→ TIA Portal V14)

→ In the portal view under the "Start" menu, select the command → "Create new project".



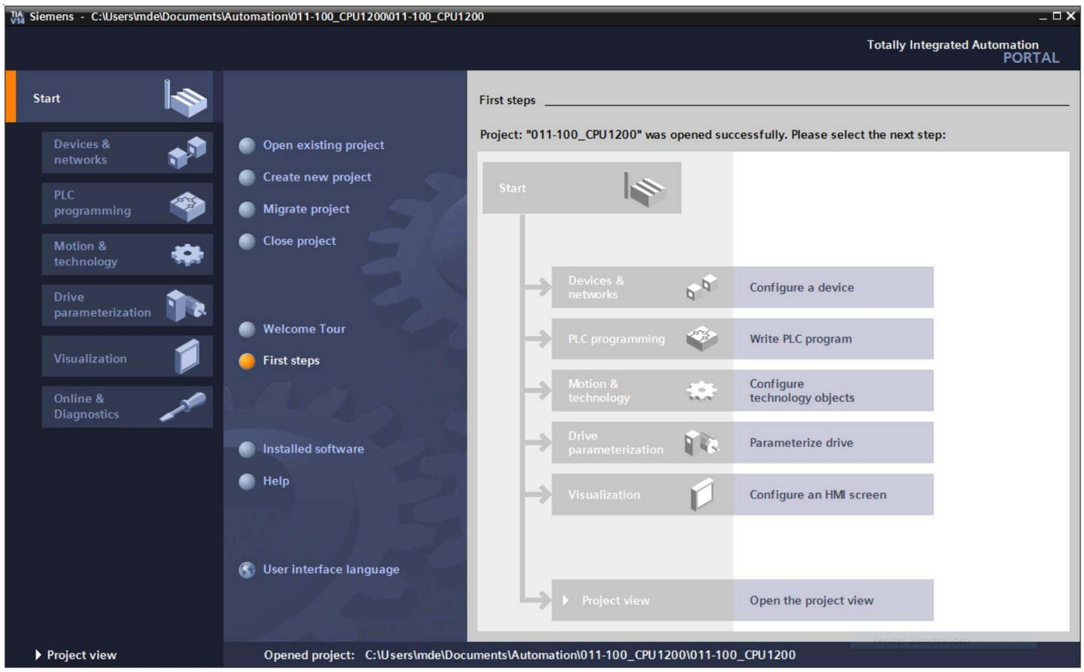
→ Modify Project name, Path, Author and Comment as appropriate and click → "Create".



→ The project will be created and opened and the menu "Start", "First steps" will open automatically.

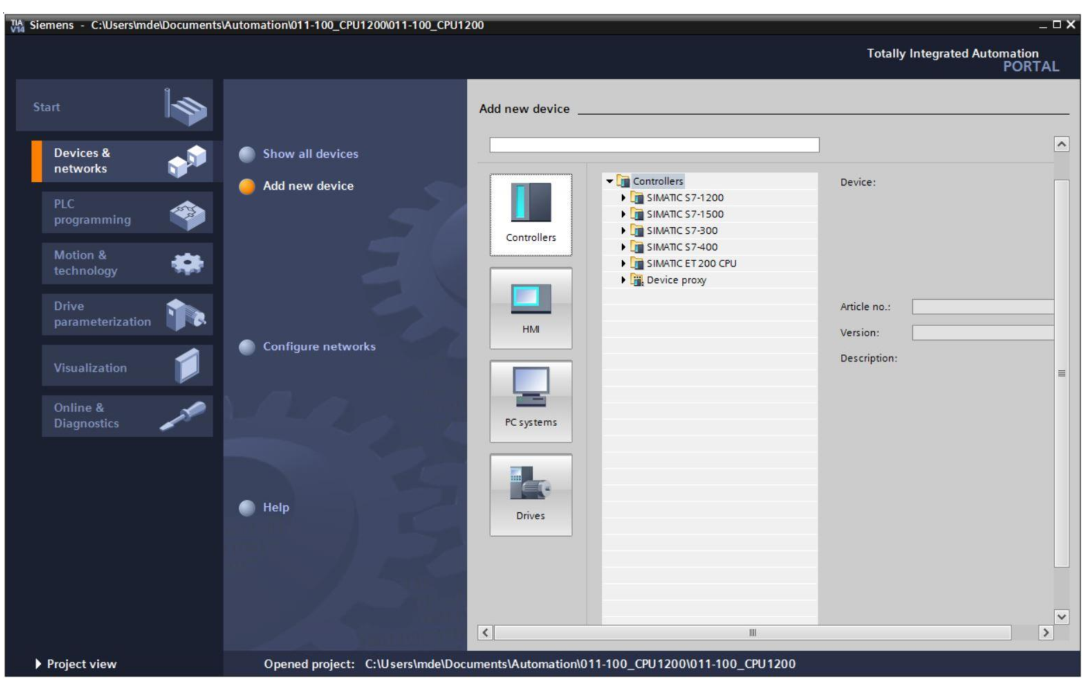
7.2 Read the hardware of SIMATIC S7-1200

→ In the → "Start" portal, select → "First steps" → "Devices & Networks" → "Configure a device".



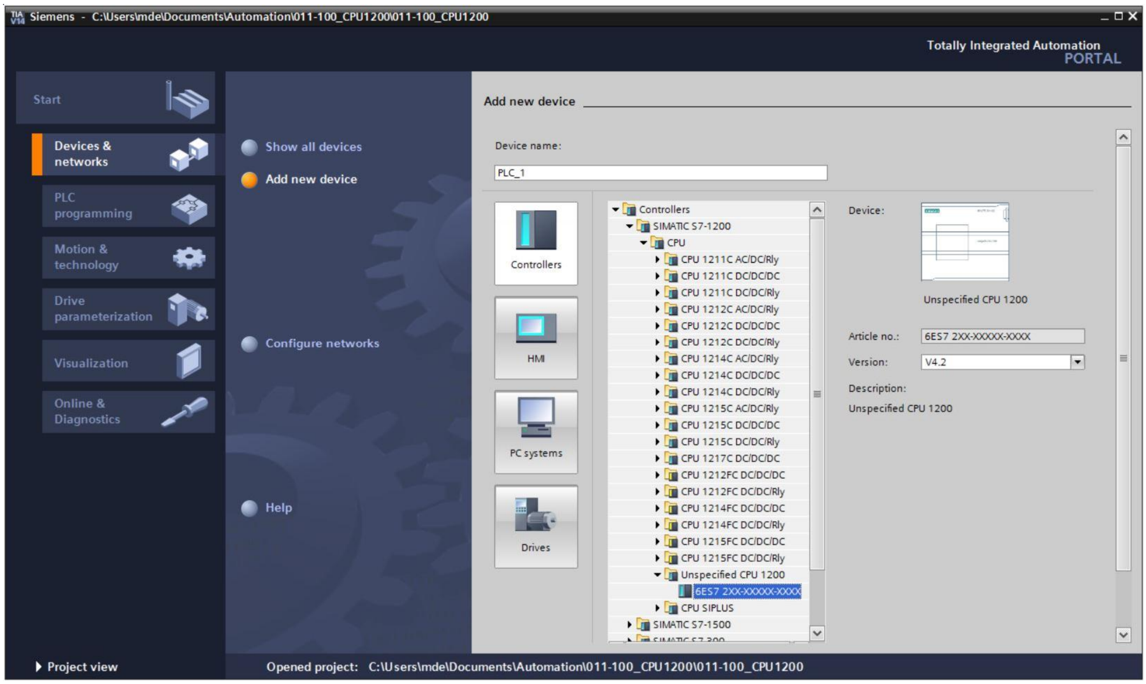
→ The "Show all devices" menu opens in the "Devices & Networks" portal.

→ Switch to the "Add new device" menu.



→ Create a new CPU. Use an unspecified model of the S7-1200 CPU with order number 6ES7 2XX-XXXXX-XXXX for this.

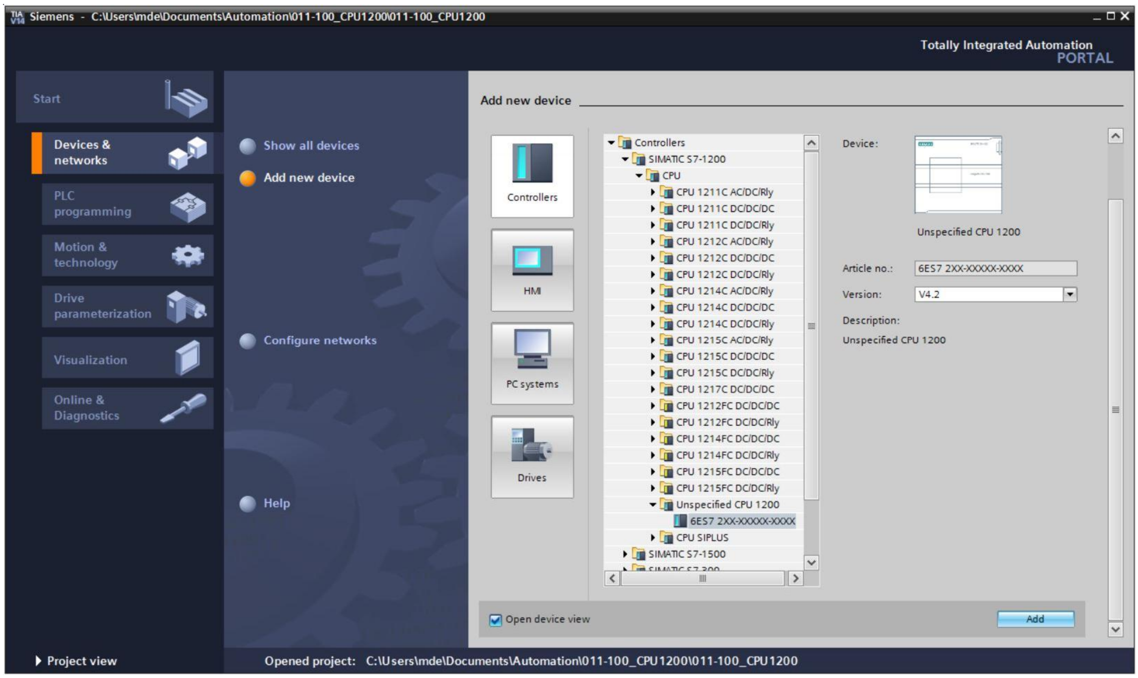
→ (Controllers → SIMATIC S7-1200 → CPU → Unspecified CPU 1200 → 6ES7 2XX-XXXXX-XXXX → V4.2)



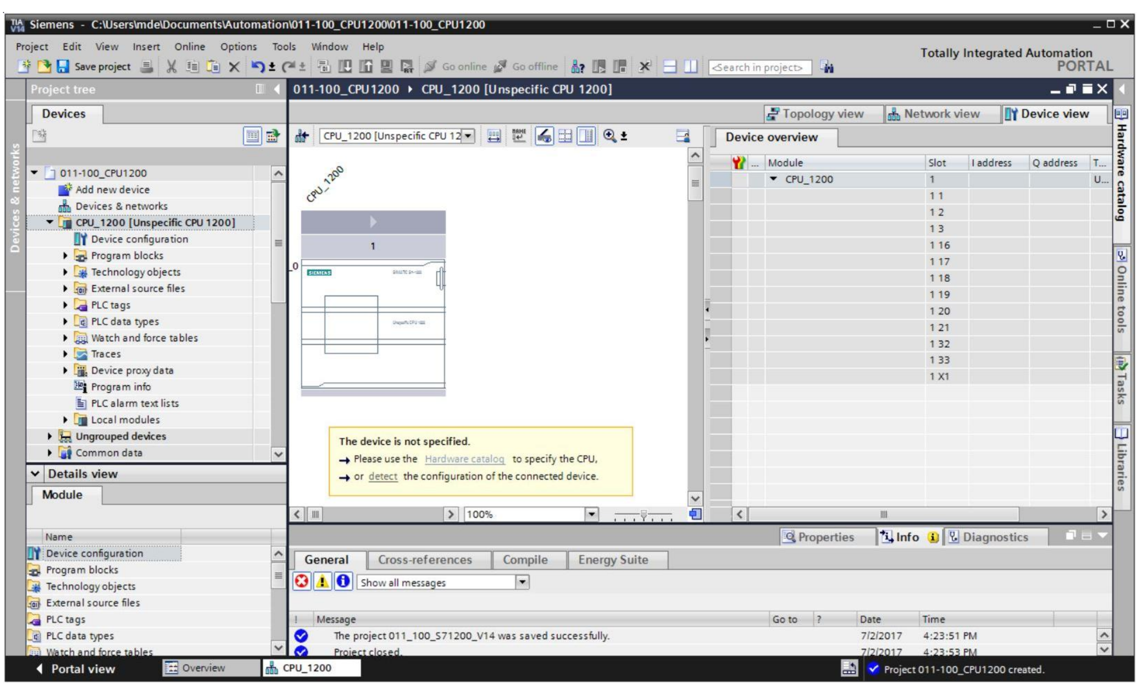
→ Assign a device name (Device name → "CPU_1200").

→ Select "Open device view".

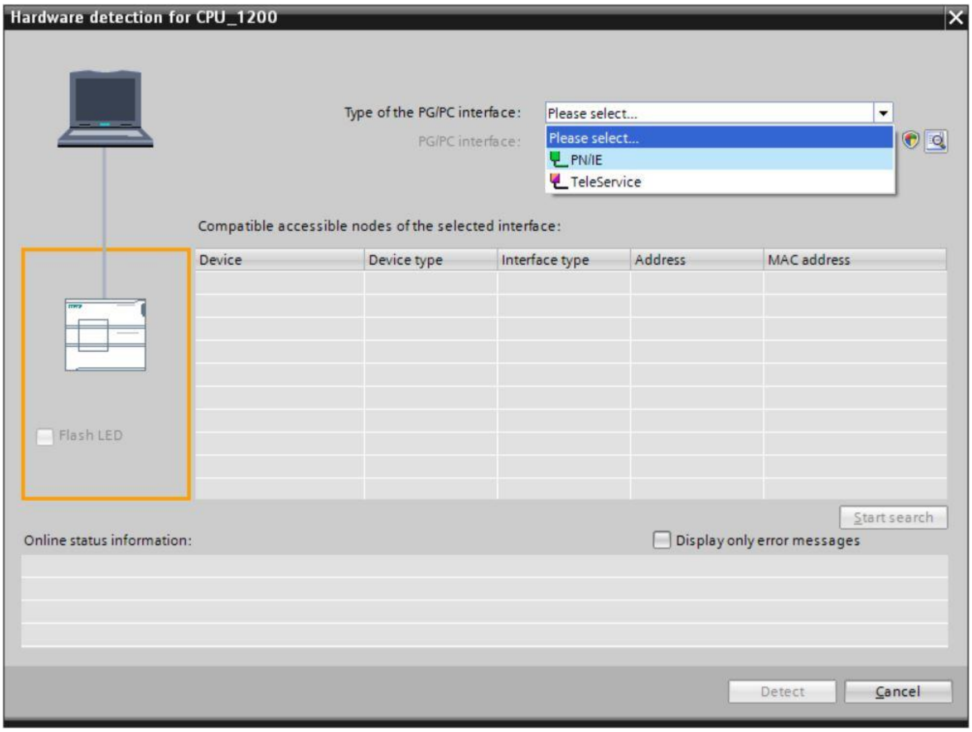
→ Click "Add".



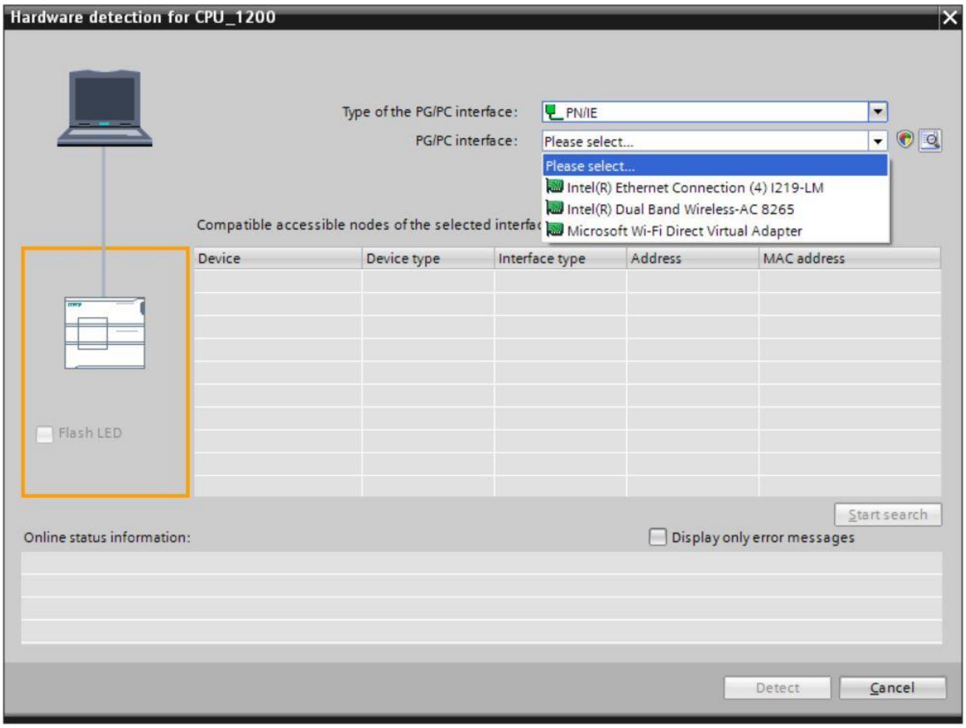
→ The TIA Portal now switches automatically to the project view and displays a notice there that this device is not specified. In order to have the hardware configuration automatically detected, start detection by clicking "detect" in the yellow information box (→ detect).



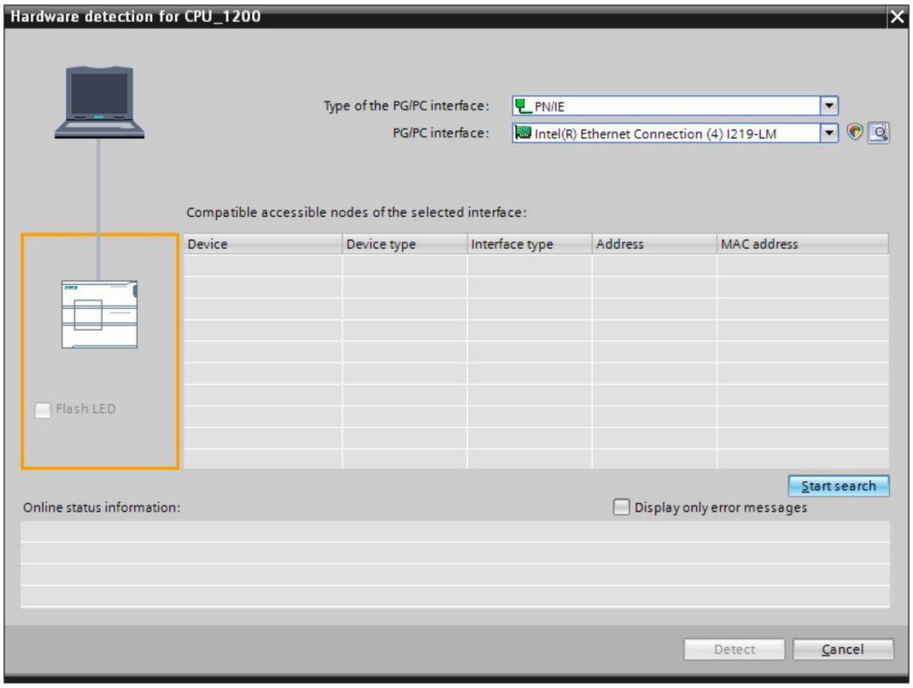
→ Select the type of your PG/PC interface (→ Type of the PG/PC interface: PN/IE).



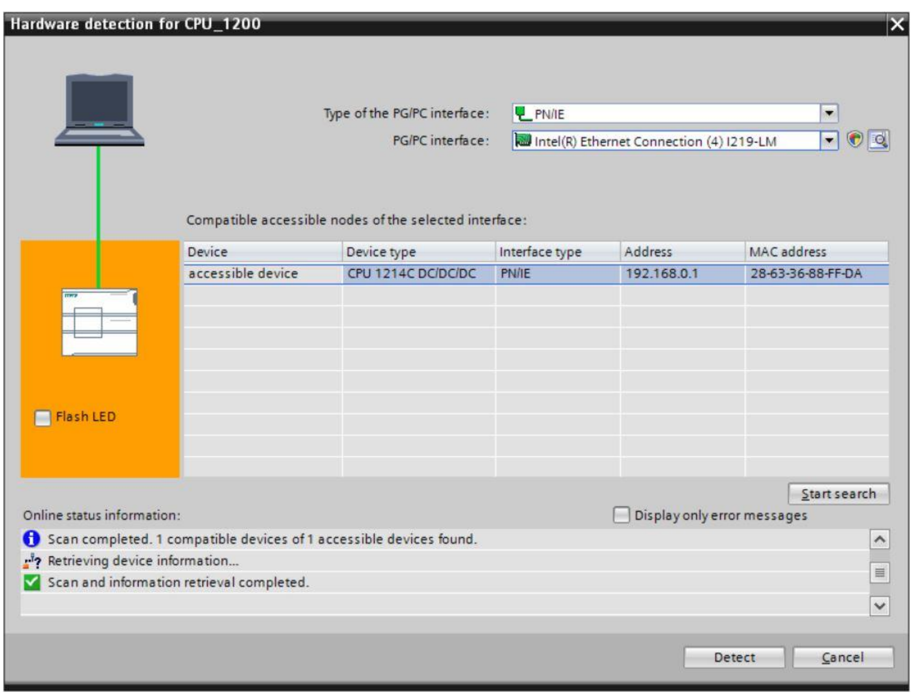
→ You can now select the network adapter you want to use to establish an Ethernet connection with the PLC (→ PG/PC interface: Intel(R) Ethernet Connection (4) I219-LM).



→ The search for devices in the network must be started by clicking the → Start search button.

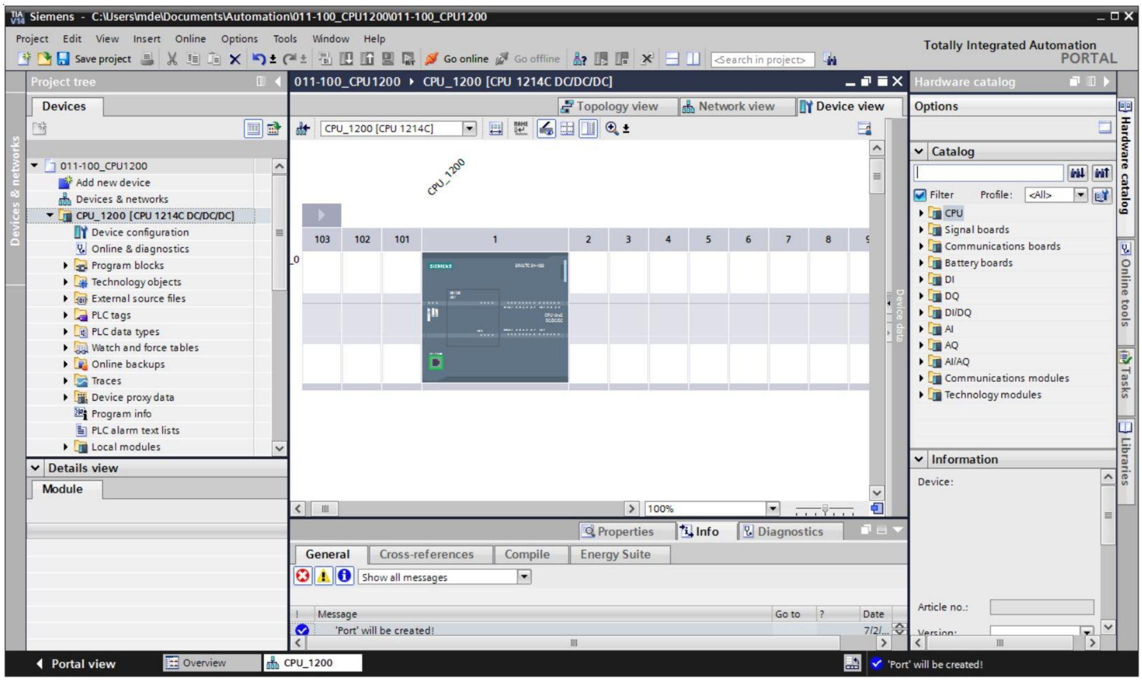


→ All accessible nodes are found and listed. If you have selected the correct CPU, the corresponding CPU and all the connected modules will be detected when you click "detect".



Note: *If the list does not contain your CPU, ensure that you have selected the correct network adapter and have established a connection between the laptop and CPU.*

→ The TIA Portal shows the complete device configuration of the selected CPU with signal board SB1232, 1 AO.



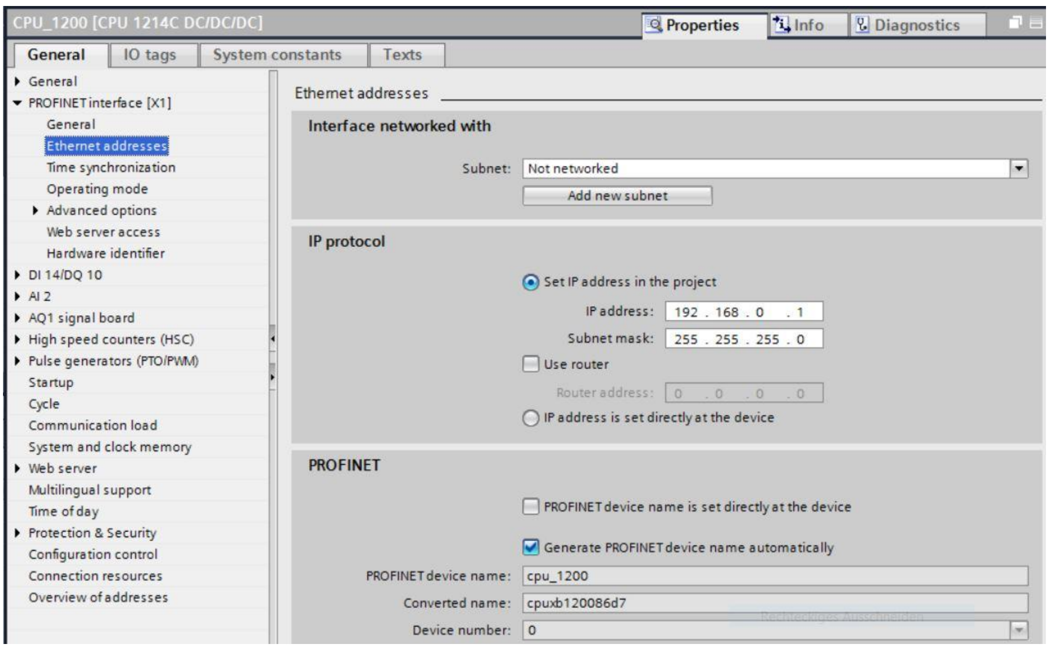
Note: *You can now configure the CPU according to your specifications there. Possible settings include the PROFINET interface, startup characteristics, cycle, password protection, communication load and many more.*

2

7.3 Configure the Ethernet interface of the CPU 1214C

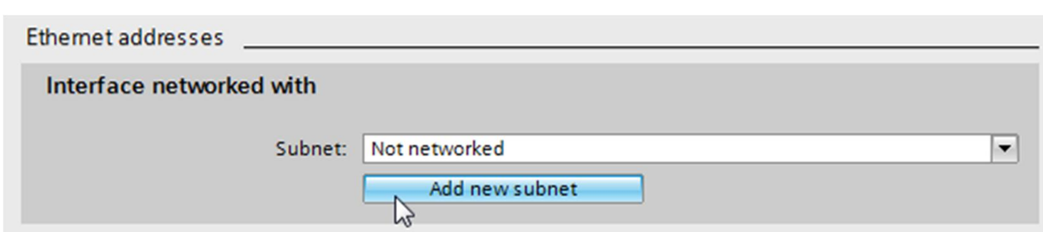
→ Select the CPU with a double-click

→ Under → "Properties", open the → "PROFINET interface [X1]" menu and select the → "Ethernet addresses" entry.



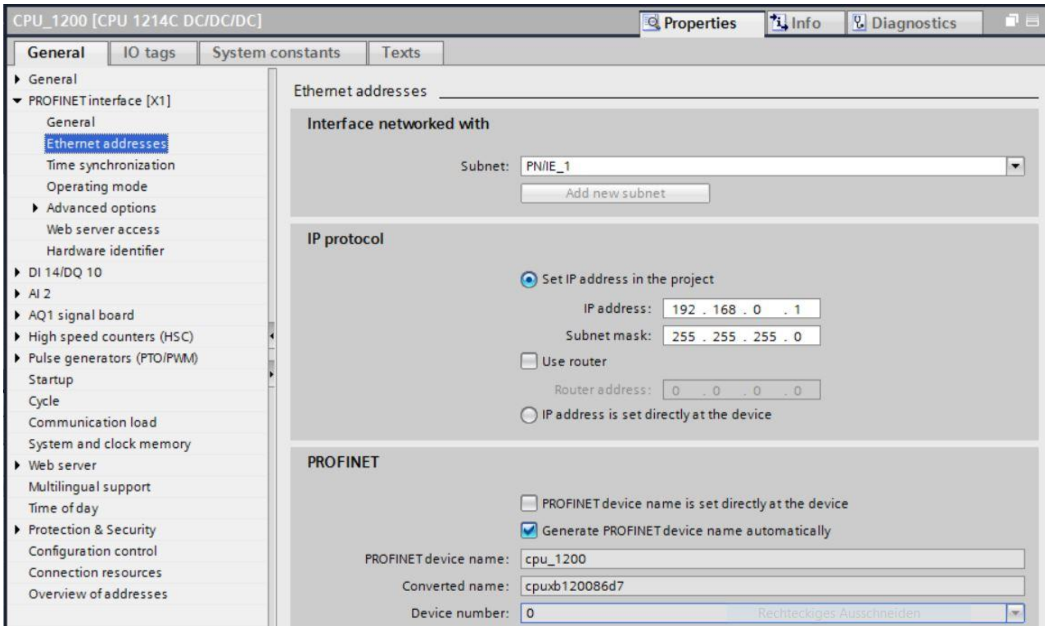
→ Under "Interface networked with", only the "Not networked" entry is available.

→ Add an Ethernet subnet with the → "Add new subnet" button.



2

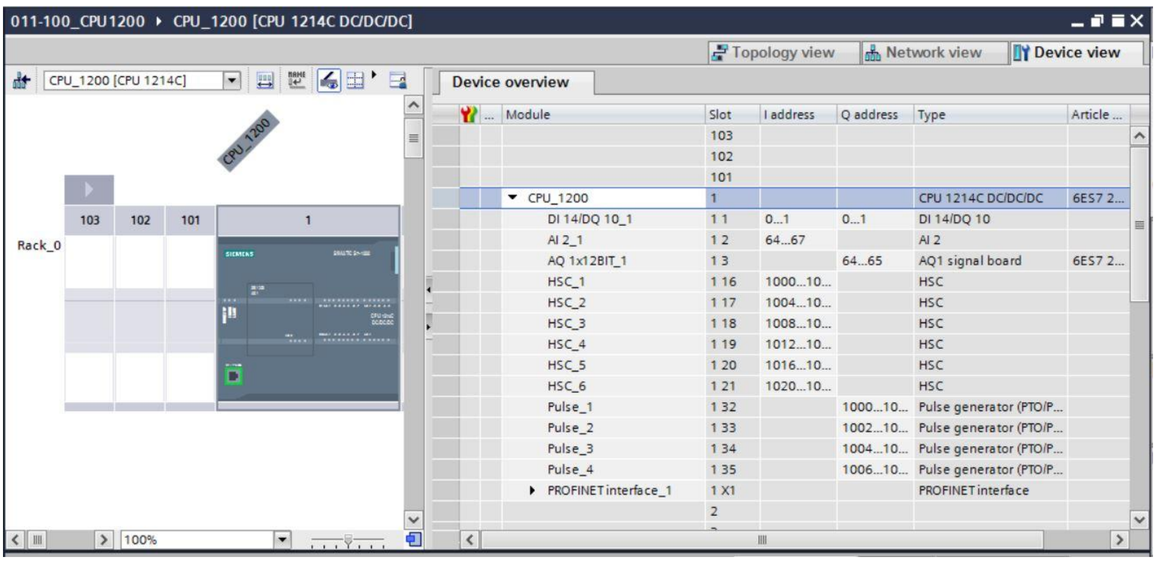
→ Keep the preassigned "IP address" and "Subnet mask".



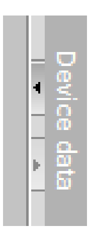
2

7.4 Configure the address areas

→ The next step is to check the address areas of the inputs and outputs and adapt them if necessary. DI/DO should have an address area of 0…1 and AI/AO should have an address area of 64…67 and 64…65, respectively. (→ Device overview → DI 14/DQ 10_1 → I address: 0..1→ Q address: 0...1 → AI 2_1 → I address: 64…67 → AQ 1x12BIT_1 → Q address: 64…65)

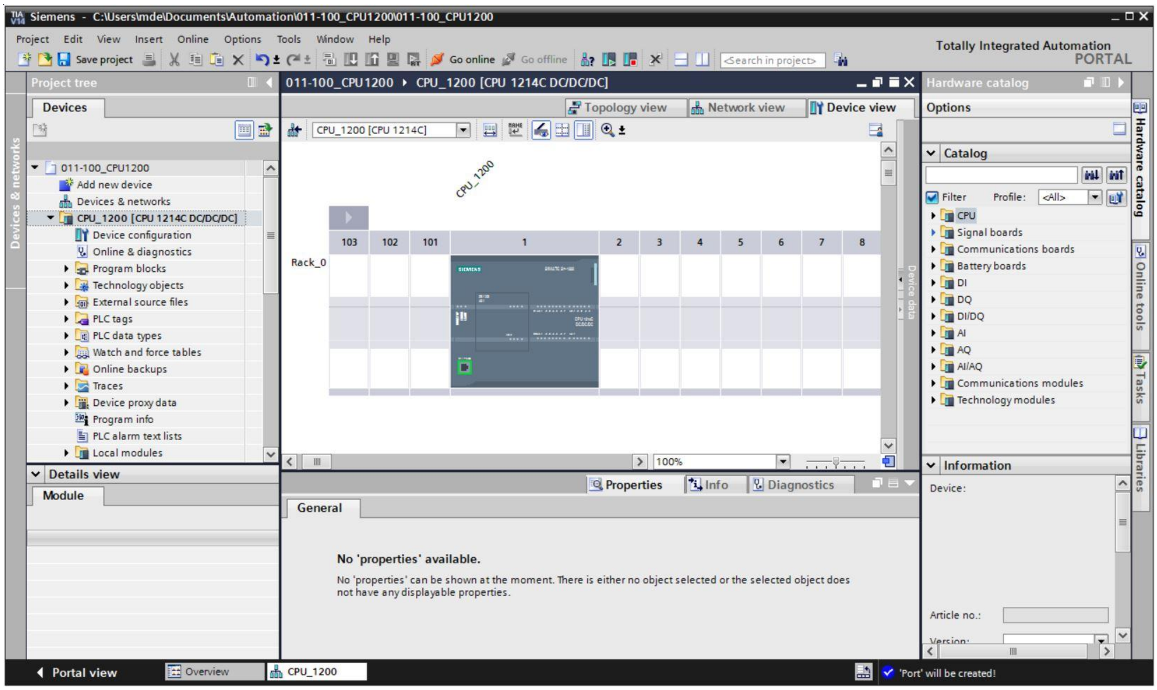


Note: *To show and hide the Device overview, you need to click the small arrow next to "Device data" on the right side of the hardware configuration.*



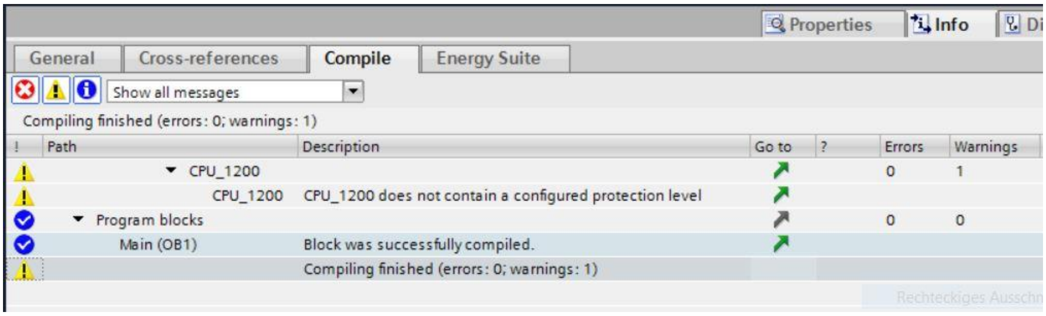
7.5 Save and compile the hardware configuration

→ Before you compile the configuration, you should save your project by clicking the → Save project button. To compile your CPU with the device configuration, first select the → "CPU_1200 [CPU1214C DC/DC/DC]" folder and click the → "Compile" icon.



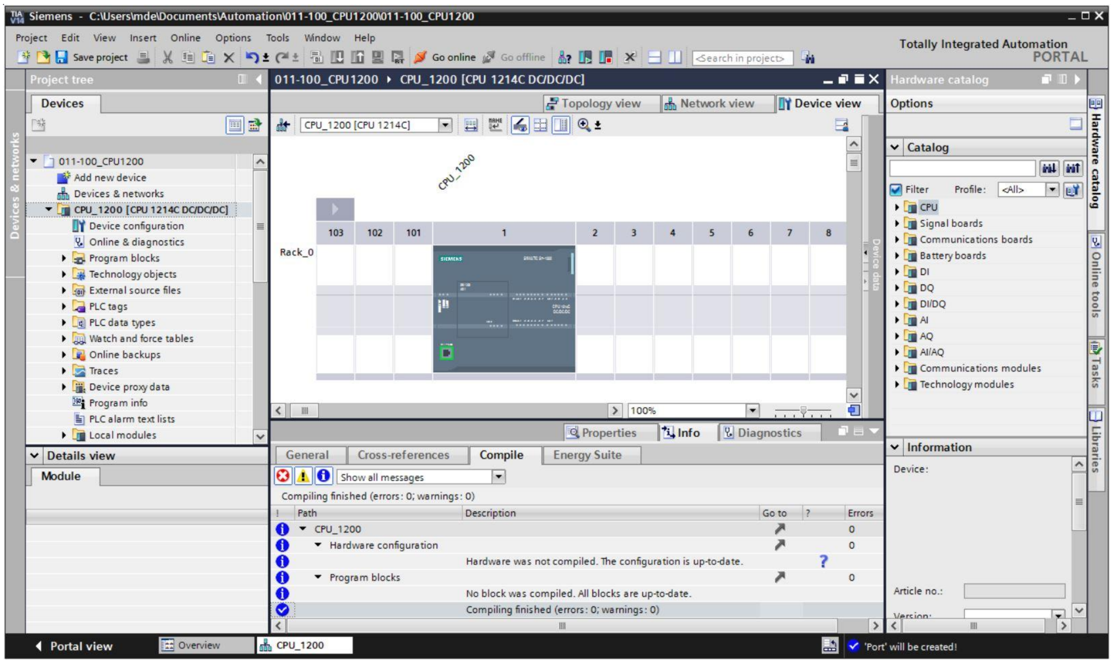
***Note:** "Save project" should be used again and again when working on a project since this does not happen automatically. A prompt to save the project only occurs when the TIA Portal is closed.*

→ If the project was compiled without errors, you see the following screen.

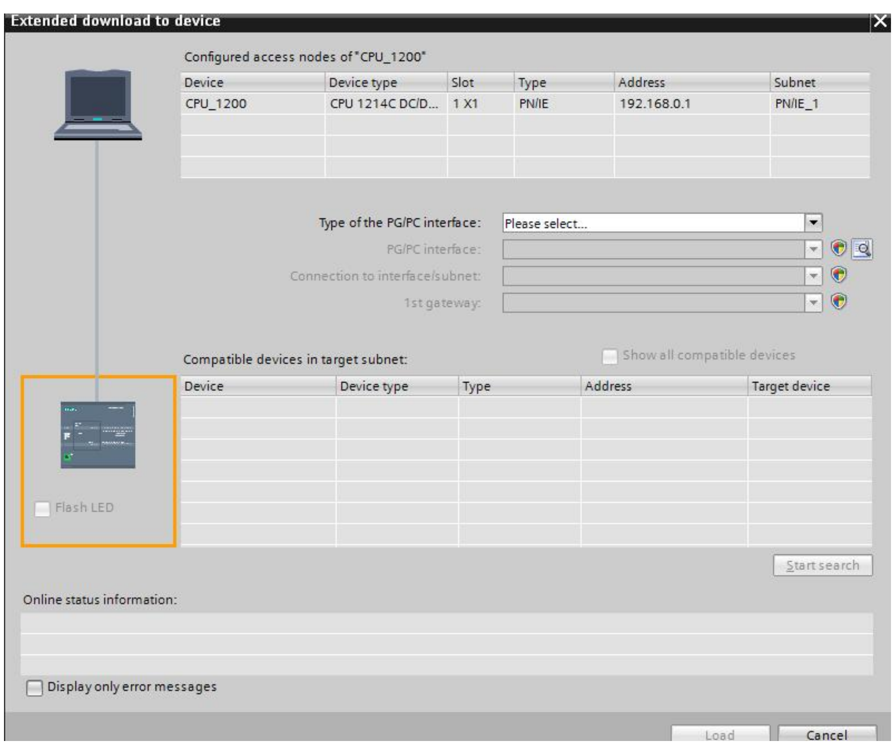


7.6 Download the hardware configuration to the device

→ To download your entire CPU, select the → "CPU_1200 [CPU1214C DC/DC/DC]" folder and click the → "Download" icon.

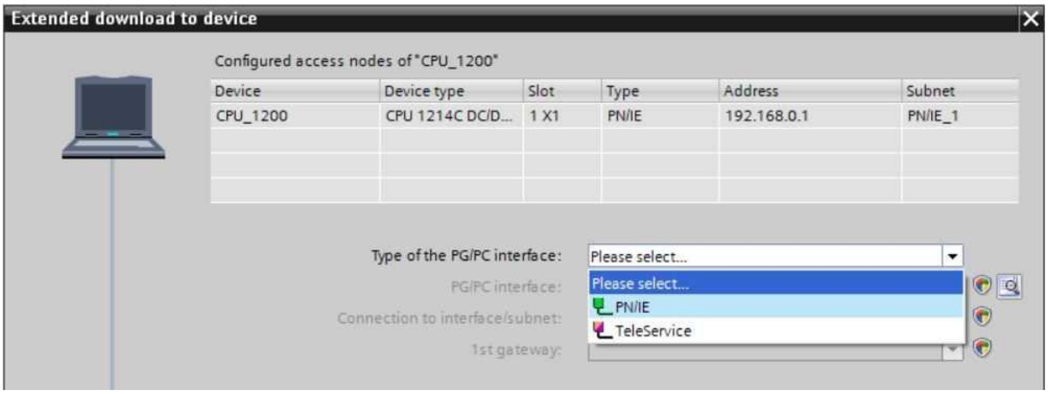


→ The manager for configuring the connection properties (extended download) opens.

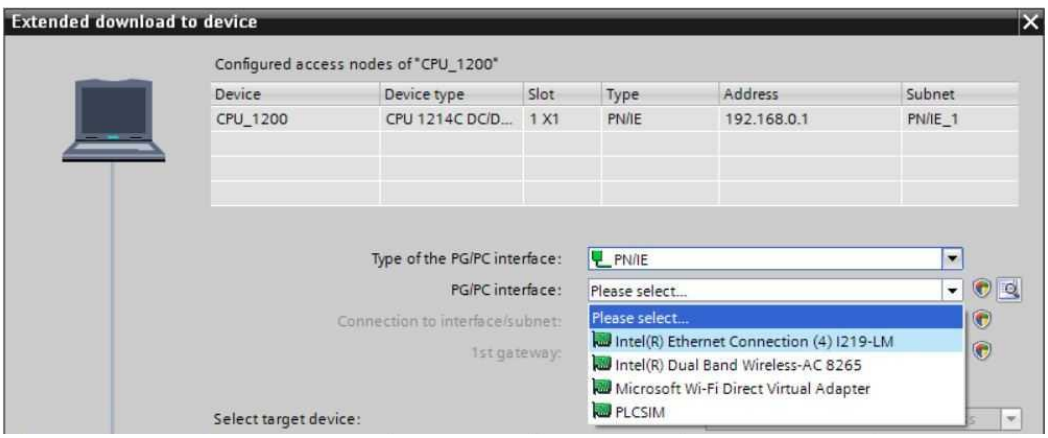


→ First, the interface must be correctly selected. This happens in three steps.

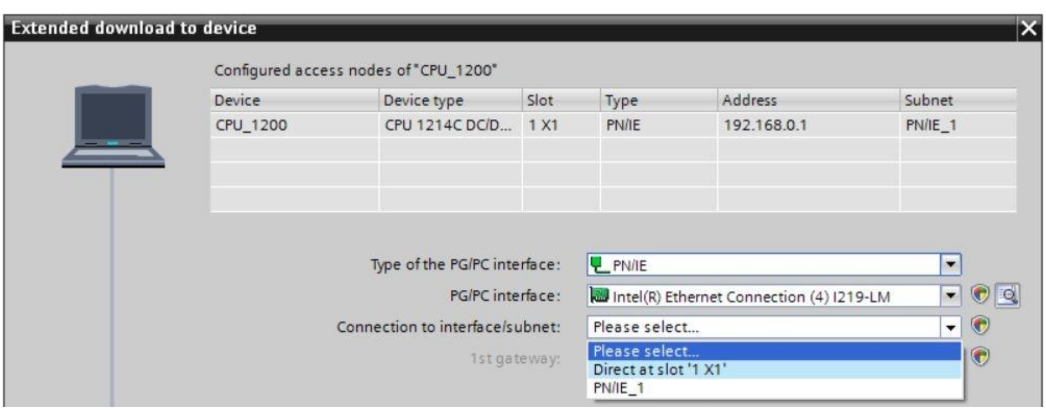
→ Type of the PG/PC interface → PN/IE



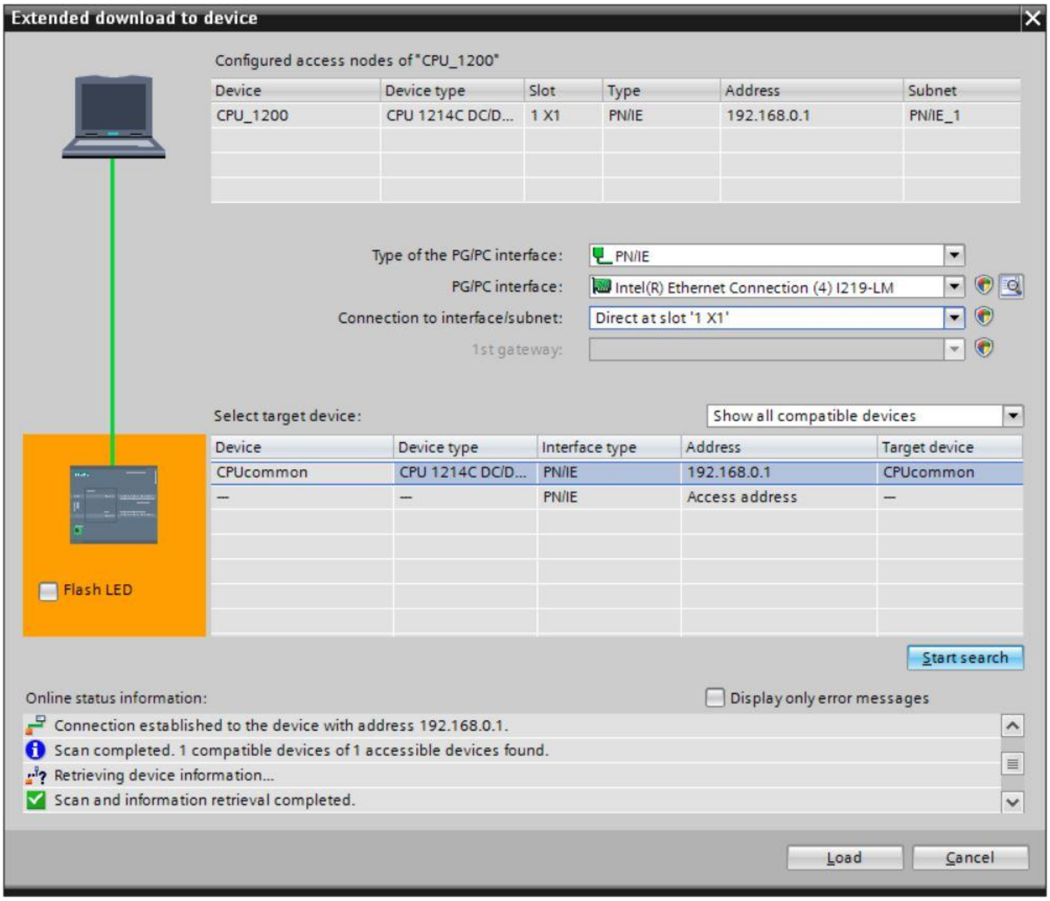
→ PG/PC interface → here: Intel(R) Ethernet Connection I217-LM



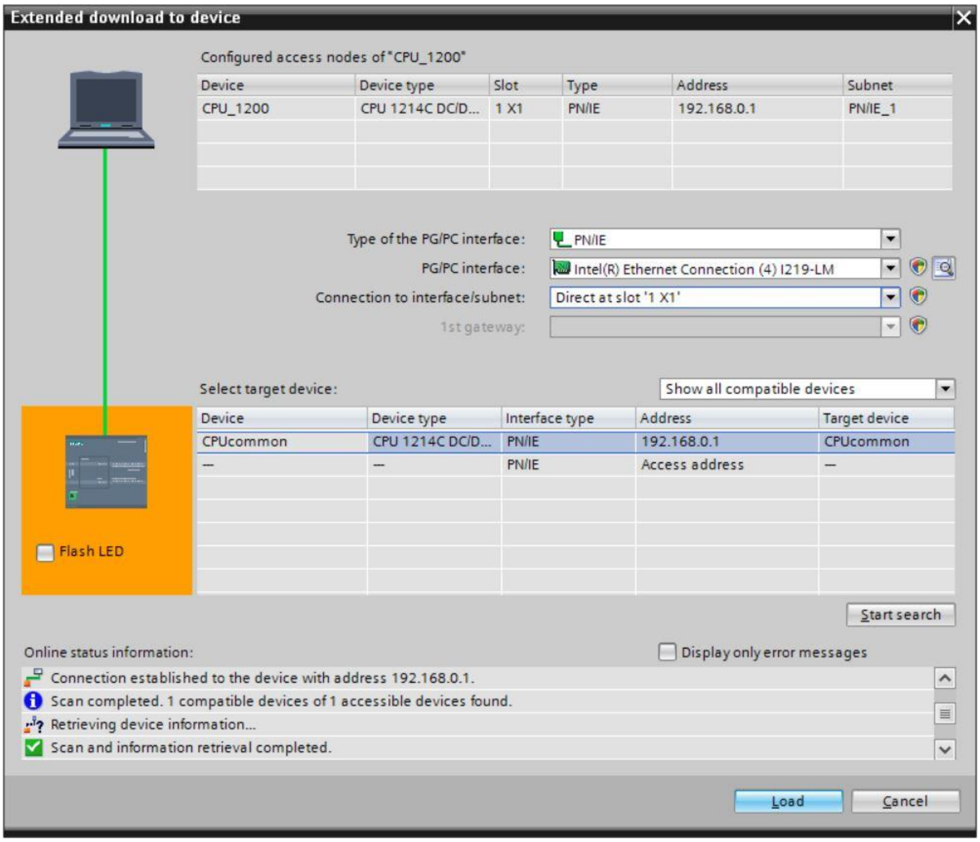
→ Connection to interface/subnet → "PN/IE_1"



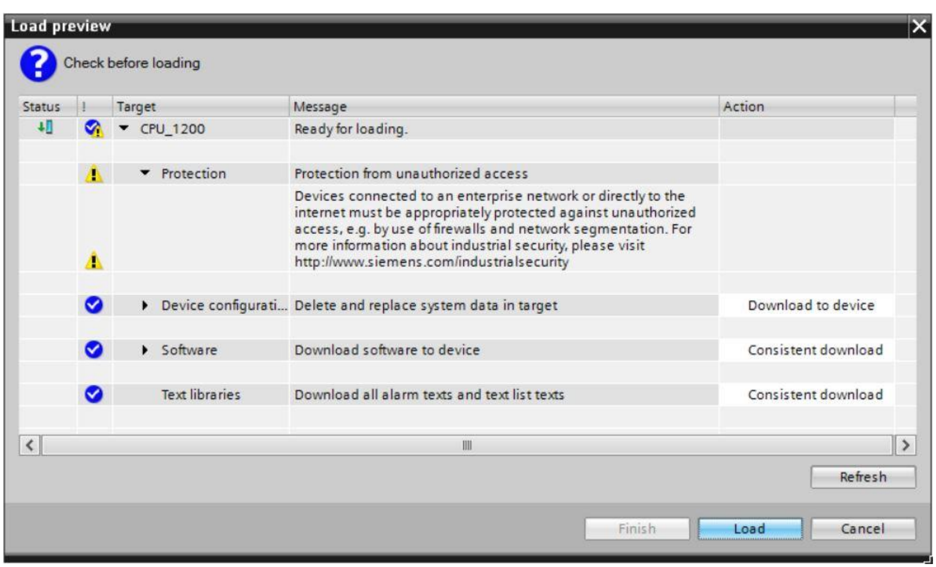
→ The → "Show all compatible devices" check box must be selected. The search for devices in the network is started by clicking the →  button.



→ If your CPU is shown in the "Compatible devices in target subnet" list, you must select it. The download can then be started (→ CPU 1214C DC/DC/DC → "Load").



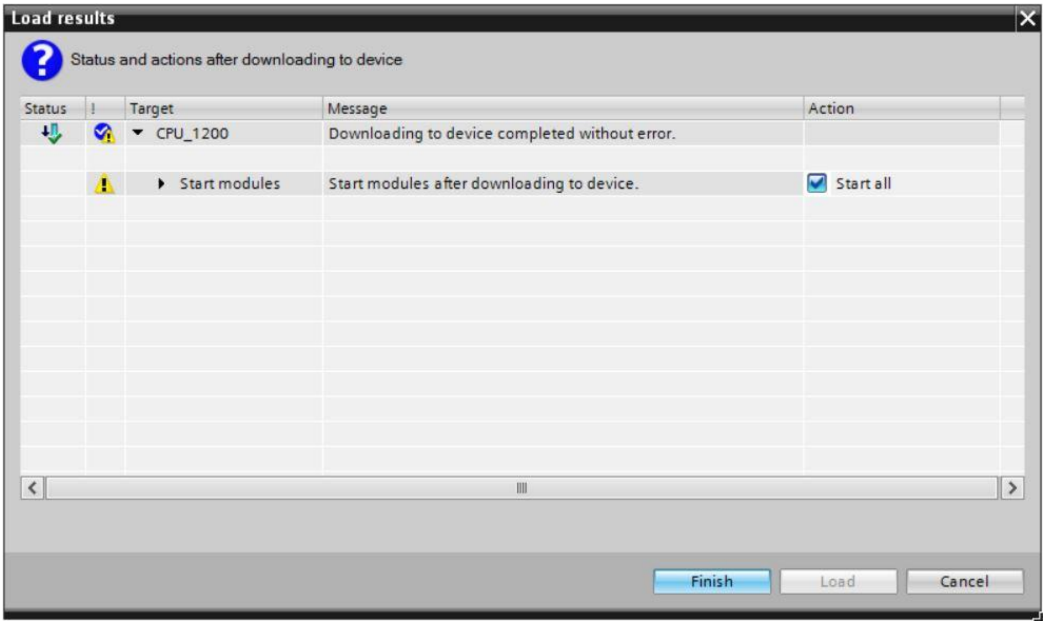
→ You first obtain a preview. Confirm the prompt → "Overwrite all" and continue with → "Load".



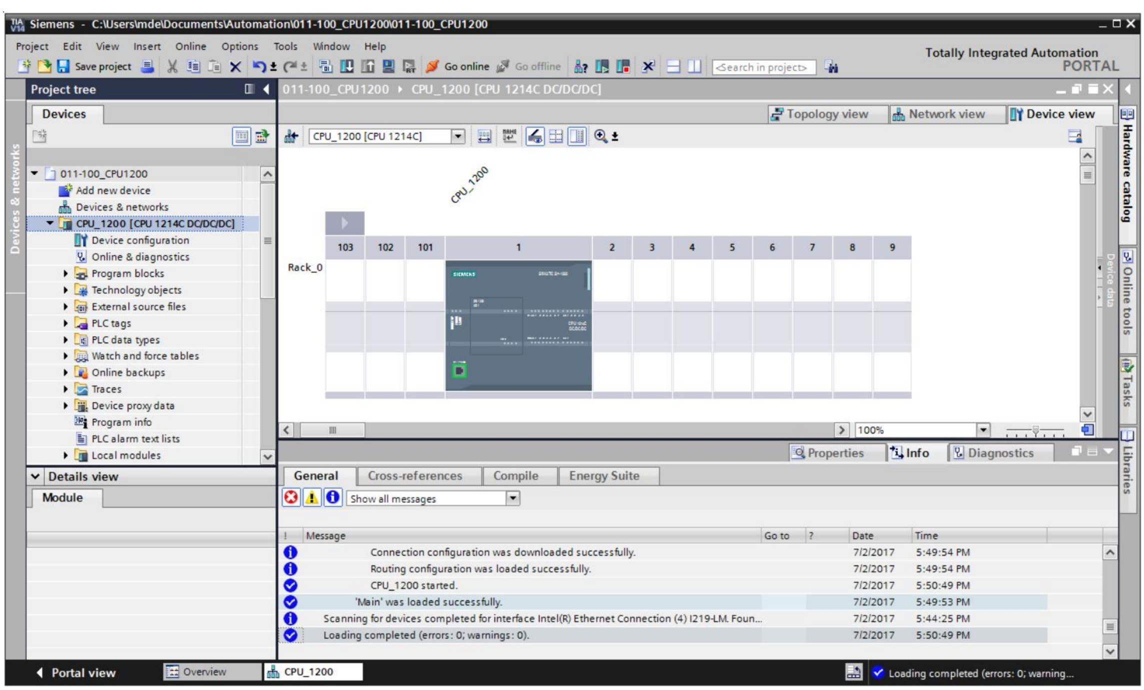
Note: *The symbol should be visible in every line of the "Load preview". You can find additional information in the "Message" column.*

2

→ The → "Start all" option will be selected next before the download operation can be completed with → "Finish".

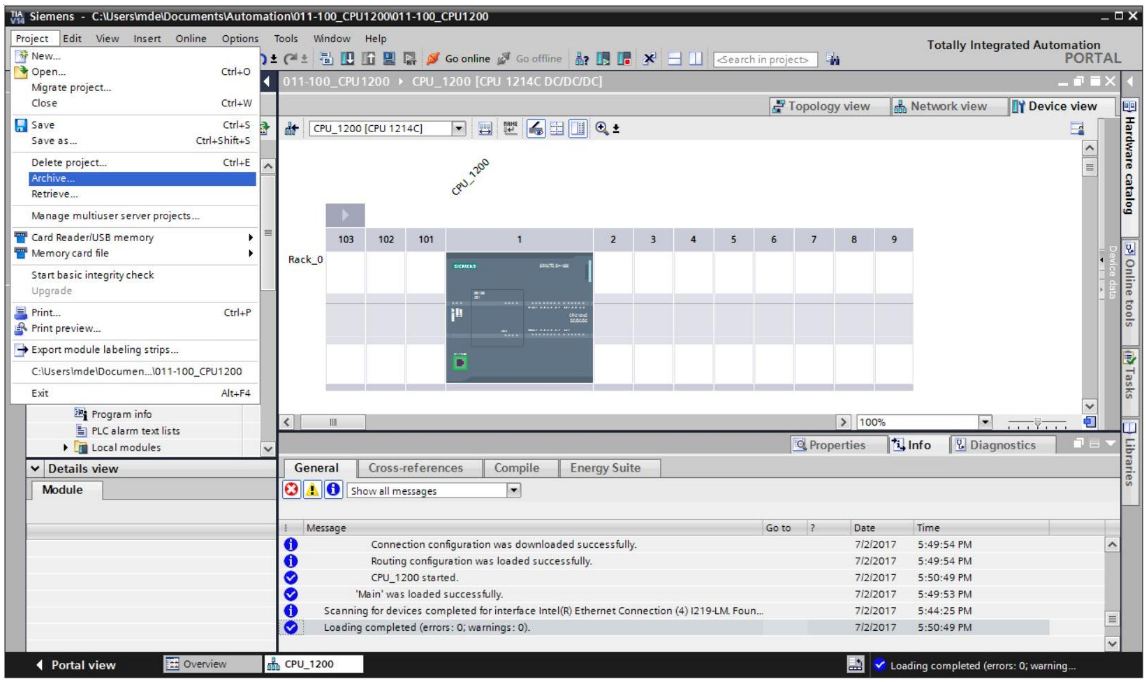


→ After a successful download, the project view will open again automatically. A loading report appears in the information field under "General". This can be helpful when troubleshooting an unsuccessful download.

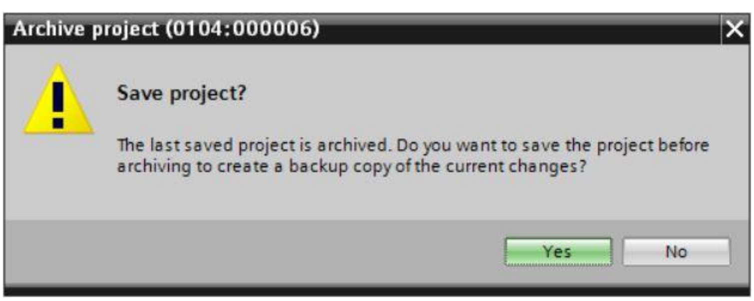


7.7 Archive the project

→ To archive the project, select the → "Archive ..." command in the → "Project" menu.



→ Confirm the prompt to save the project with → "Yes".



→ Select a folder where you want to archive your project and save it as file type "TIA Portal project archive" (→ "TIA Portal project archive" → "SCE_EN_011-100_Unspecified hardware configuration_S7-1200" → "Save").

2

7.8 Checklist

No.	Description	Completed
1	Project was created	
2	Slot 1: CPU with correct order number	
3	Slot 1: CPU with correct firmware version	
4	Signal board: Analog module AQ 1x12BIT with correct order number	
5	Signal board: Analog module AQ 1x12BIT with correct firmware version	
6	Address areas correct	
7	Hardware configuration was compiled without error message	
8	Hardware configuration was downloaded without error message	
9	Project was successfully archived	

8 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

3

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is available at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, get in touch with your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

Table of contents

3

Specified Hardware Configuration – SIMATIC S7-1200

1 Goal

In this chapter, you will first learn how to ***create a project***. You are then shown how the ***hardware is configured***.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

You do not need any previous knowledge from other chapters to successfully complete this chapter. You only need an S7-1200 controller and a PC with the STEP 7 Basic V14 (TIA Portal V14) software.

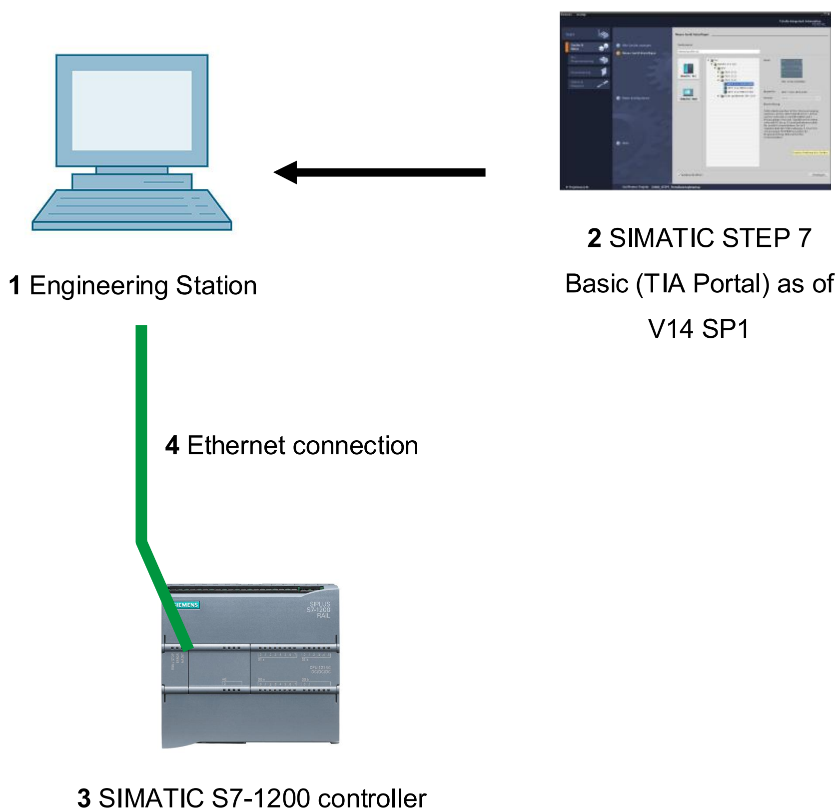
3 Required hardware and software

1 Engineering station: Requirements include hardware and operating system (for additional information, see Readme on the TIA Portal Installation DVDs)

2 SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1

4 Ethernet connection between engineering station and controller



4 Theory

4.1 SIMATIC S7-1200 automation system

The SIMATIC S7-1200 automation system is a modular microcontroller system for the lower performance range.

A comprehensive range of modules is available to optimally adapt the system to the automation task. The S7 controller consists of a power supply and a CPU with integrated inputs and outputs or additional input and output modules for digital and analog signals.

If necessary, communication processors and function modules are also used for special tasks such as stepper motor control.

The programmable logic controller (PLC) uses the S7 program to monitor and control a machine or process. In doing so, the IO modules are scanned in the S7 program using input addresses (%I) and addressed using output addresses (%Q).

The system is programmed with the TIA Portal Basic or Professional software.

4.1.1 Range of modules

The SIMATIC S7-1200 is a modular automation system and offers the following range of modules:

Central processing units (CPUs) with different performance, integrated inputs/outputs, and PROFINET interface (e.g. CPU 1214C)

Power supply module (PM) with input 120/230 V AC, 50 Hz / 60 Hz, 1.2 A / 0.7 A and output 24 V DC / 2.5 A

Signal boards (SBs) for adding analog or digital inputs/outputs, in which case the size of the CPU remains unchanged. (Signal boards can be used with CPUs 1211C / 1212C and 1214C.)

Signal modules (SMs) for digital and analog inputs and outputs (a maximum of 2 SMs can be used for CPU 1212C and a maximum of 8 SMs for CPU 1214C.)

Communication modules (CMs) for serial communication RS232 / RS 485 (Up to 3 CMs can be used for CPUs 1211C / 1212C and 1214C.)

Compact switch module (CSM) with 4x RJ45 sockets 10/100 Mbps

SIMATIC memory cards from 2 MB to 32 MB for storing program data and for easy exchange of CPUs during maintenance.

Note: *Only a single CPU (any type) with integrated digital inputs and digital outputs is needed for this module.*

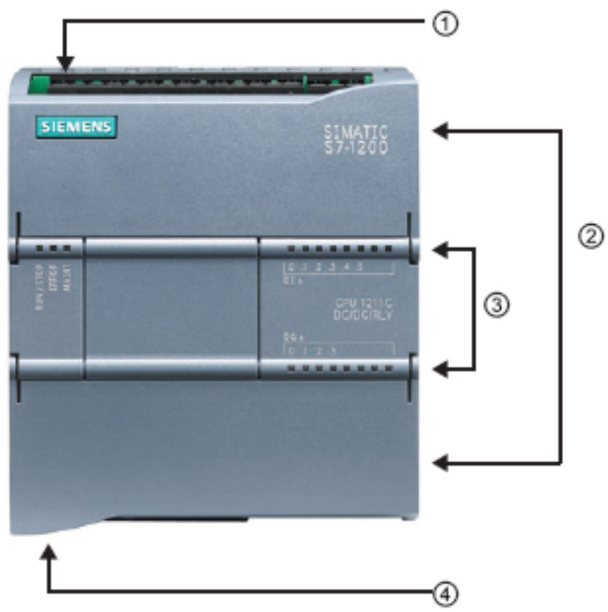
4.2 Operator control and display elements of the CPU 1214C DC/DC/DC

4.2.1 Front view of the CPU 1214C DC/DC/DC

With integrated power supply (24 V connection) and integrated inputs and outputs, the CPU 1214C DC/DC/DC is immediately ready for use without any other components.

The CPU has an integrated TCP/IP connection for communication with a programming device.

The CPU can thus communicate with HMI devices or other CPUs via an Ethernet network.



① 24 V connection

② Plug-in terminal block for user wiring (behind the cover flaps)

③ Status LEDs for the integrated IO and the operating state of the CPU

④ TCP/IP connection (on the underside of the CPU)

4.2.2 SIMATIC memory card (MC)

The optional **SIMATIC memory card (MC)** stores a program as well as data, system data, files and projects. It can be used for:

– Transferring a program to multiple CPUs

– Firmware update of CPUs, signal modules (SMs) and communication modules (CMs)

– Easy replacement of the CPU

4.2.3 Operating states of the CPU

The CPU can have the following three operating states:

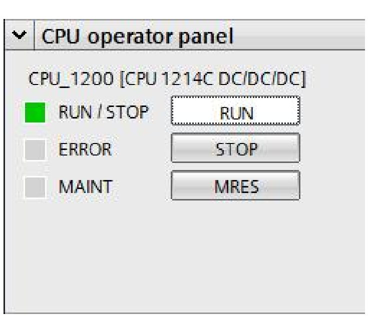
– In the **STOP** operating state, the CPU is not executing the program and you can download a project.

– In the **STARTUP** operating state, the CPU is starting up.

– In the **RUN** operating state, the program is cyclically executed.

The CPU does not have a physical switch for changing the operating state.

You use the button on the operator panel of the STEP 7 Basic software to change the operating state (**STOP** or **RUN**). The operator panel also contains an **MRES** button for performing a memory reset and displays the status LEDs of the CPU.



4.2.4 Status and error displays

The **RUN/STOP status LED** on the front side of the CPU indicates the current operating state of the CPU by the color of the display.

– **Yellow** light indicates **STOP** operating state.
– **Green** light indicates **RUN** operating state.
– A **flashing light** indicates **STARTUP** operating state.

There are two additional LEDs here: **ERROR** LED for indicating errors and **MAINT** LED for indicating that maintenance is required.

4.3 STEP 7 Basic V14 (TIA Portal V14) programming software

The STEP 7 Basic V14 (TIA Portal V14) software is the programming tool for the following automation systems:

– SIMATIC S7-1200

– Basic Panels

STEP 7 Basic V14 provides the following functions for automation of a system:

– Configuration and parameter assignment of the hardware

– Specification of the communication

– Programming

– Testing, commissioning and servicing with operational/diagnostic functions

– Documentation

– Creation of visualizations for SIMATIC Basic Panels using the integrated WinCC Basic software

Support is provided for all functions through detailed online help.

4.3.1 Project

To implement a solution for an automation and visualization task, you create a project in the TIA Portal. A project in the TIA Portal contains the configuration data for the configuration and internetworking of devices as well as the programs and the configuration of the visualization.

4.3.2 Hardware configuration

The *hardware configuration* includes the configuration of the devices, consisting of the hardware of the automation system, the field devices on the PROFINET bus system and the hardware for visualization. The configuration of the networks specifies the communication between the various hardware components. Individual hardware components are *inserted in the hardware configuration* from catalogs.

The hardware of SIMATIC S7-1200 automation systems comprises the controller (CPU), the signal modules for input and output signals (SMs), the communication modules (CMs) and other special-purpose modules.

The signal modules and the field devices connect the input and output data of the process to be automated and visualized to the automation system.

The hardware configuration enables the downloading of automation and visualization solutions to the automation system and access to the connected signal modules by the controller.

3

4.3.3 Planning the hardware

Before you can configure the hardware, you must plan it (hardware planning). In general, you begin by selecting which controllers are needed and how many. You then select the communication modules and signal modules. The selection of signal modules is based on the number and type of inputs and outputs needed. As the final step, a power supply that ensures the necessary power supply must be selected for each controller or field device.

The functionality required and the ambient conditions are of vital importance for planning the hardware configuration. For example, the temperature range in the application area sometimes limits which devices are available for selection. Fail-safe operation might be another requirement.

The TIA Selection Tool (Select automation technology → TIA Selection Tool and follow the instructions) provides you support. Note: The TIA Selection Tool requires Java.

Note for online research: *If more than one manual is available, you should look for the description "Device Manual", "Product Manual" or simply "Manual" (as opposed to "Function Manual", "List Manual", "System Manual", etc.) in order to find the device specifications.*

4.3.4 TIA Portal – Project view and portal view

The TIA Portal has two important views. When started, the TIA Portal displays the portal view by default. This view makes getting started easier, especially for beginning users.

The portal view provides a task-oriented view of the tools for working on the project. Here, you can quickly decide what you want to do and open the tool for the task at hand. If necessary, a change to the project view takes place automatically for the selected task.

Figure 1 shows the portal view. At the bottom left, there is an option to switch between this view and the project view.

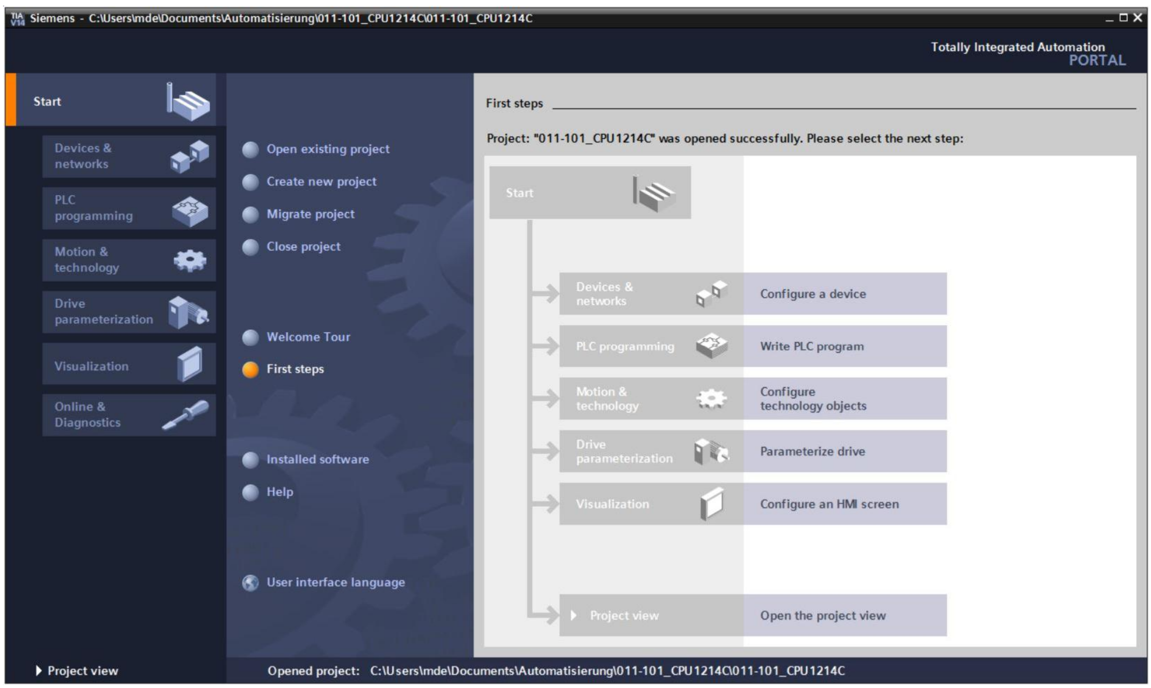


Figure 1: Portal view

The project view, as shown in Figure 2, is used for hardware configuration, programming, creation of the visualization and many other tasks.

By default, the project view displays the menu bar with the toolbars at the top, the project tree with all components of a project on the left and the so-called "task cards" with instructions and libraries, for example, on the right.

If an element (for example, the device configuration) is selected in the project tree, it is displayed in the center and can be worked on there.

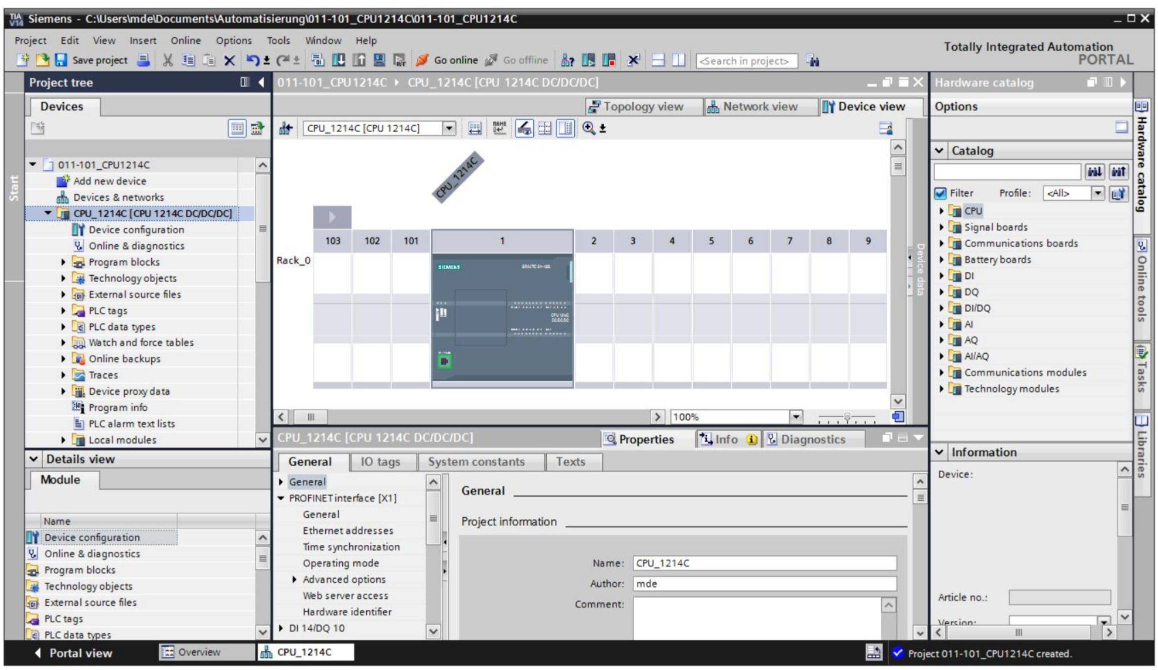
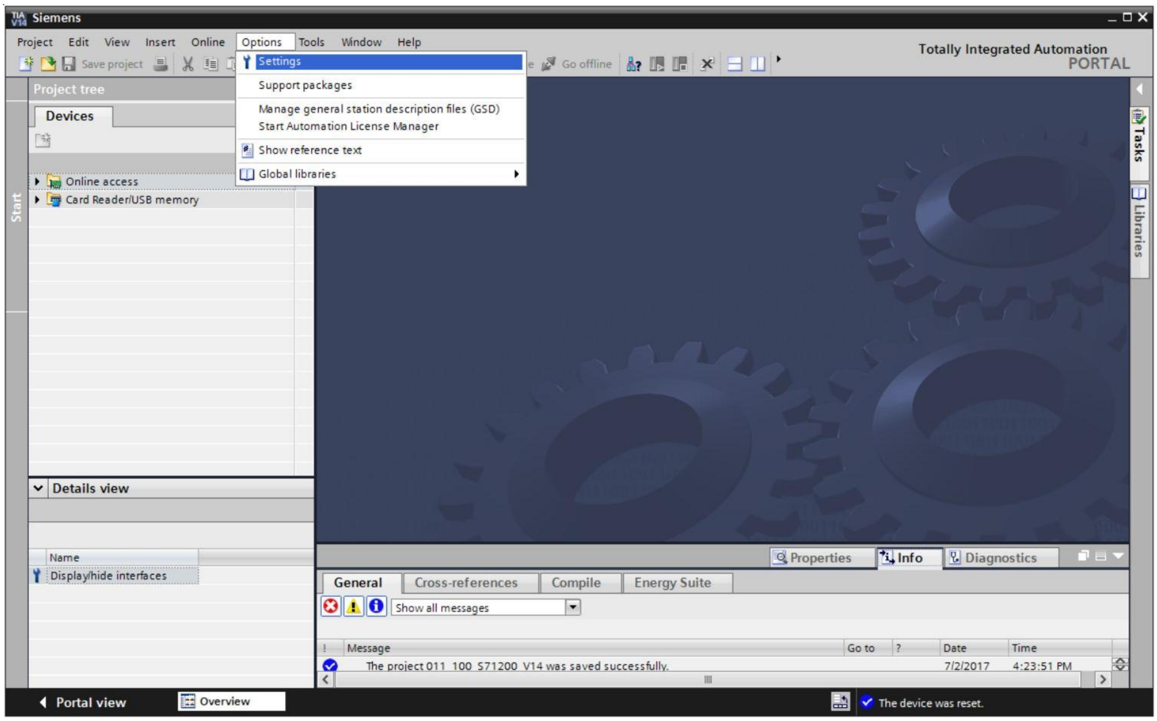


Figure 2: Project view

4.3.5 Basic settings for the TIA Portal

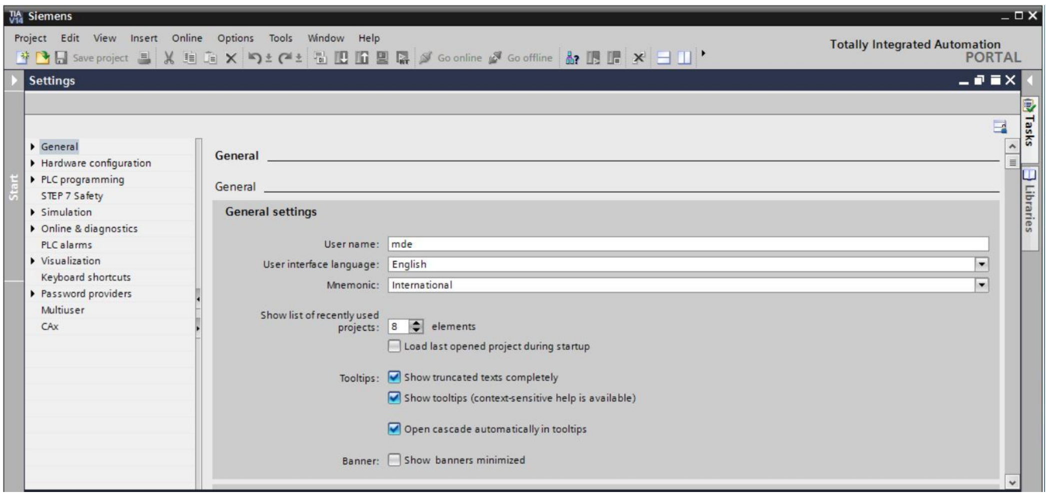
→ Users can specify their own default settings for certain settings in the TIA Portal. A few important settings are shown here.

→ In the project view, select the → "Options" menu and then → "Settings".



→ One basic setting is the selection of the user interface language and the language for the program display. In the curriculums to follow, "English" will be used for both settings.

→ Under "General" in "Settings", select → "User interface language → English" and "Mnemonic → International".



Note: *These settings can always be changed.*

4.3.6 Set the IP address on the programming device

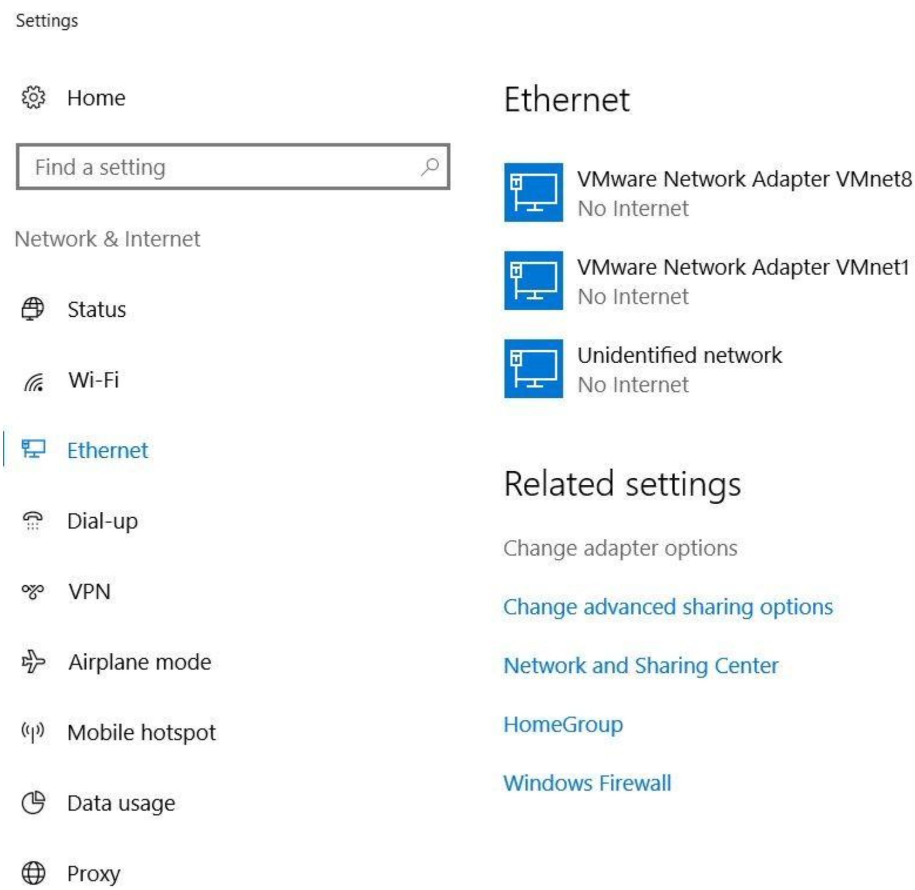
To program the SIMATIC S7-1200 controller from the PC, the programming device or a laptop, you need a TCP/IP connection or an optional PROFIBUS connection.

For the PC and SIMATIC S7-1200 to communicate with each other via TCP/IP, it is important that the IP addresses of both devices match.

First, we show you how to set the IP address of a PC with Windows 7 operating system.

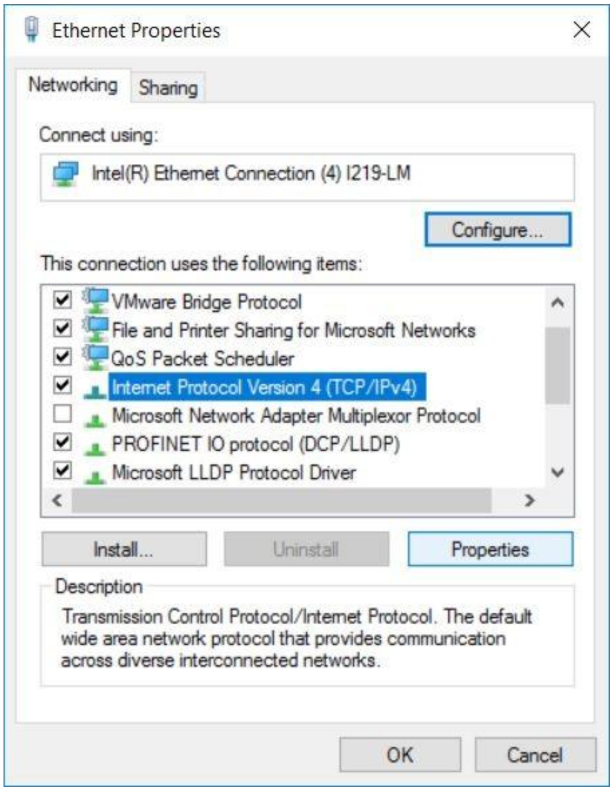
→ Locate the network icon in the taskbar at the bottom  and click → "Open Network and Sharing Center".

→ In the open Network and Sharing Center window, click → "Change adapter settings".

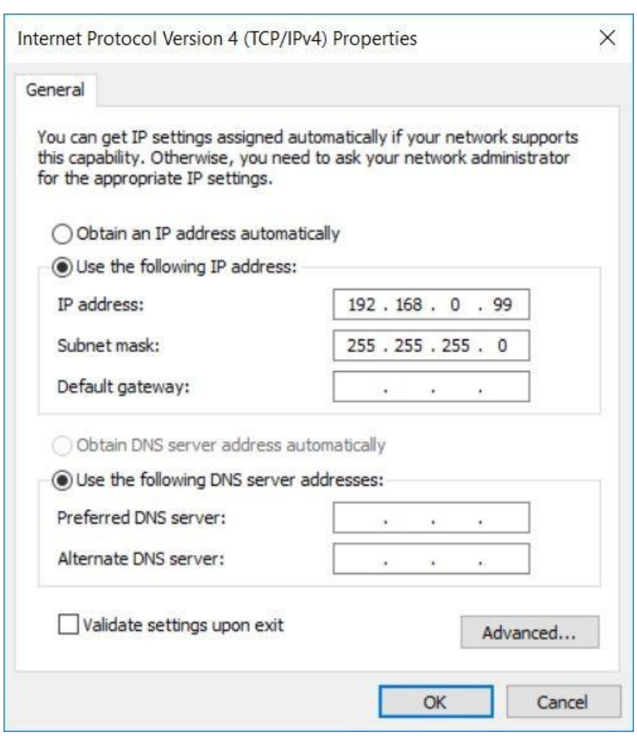


→ Select the desired → "Local Area Connection" that you want to use to connect to the controller and click → "Properties".

→ Next, select → "Properties" for → "Internet Protocol Version 4 (TCP/IP)".



→ You can use the following address, for example → IP address: 192.168.0.99 → Subnet mask 255.255.255.0 and accept the settings (→ "OK")

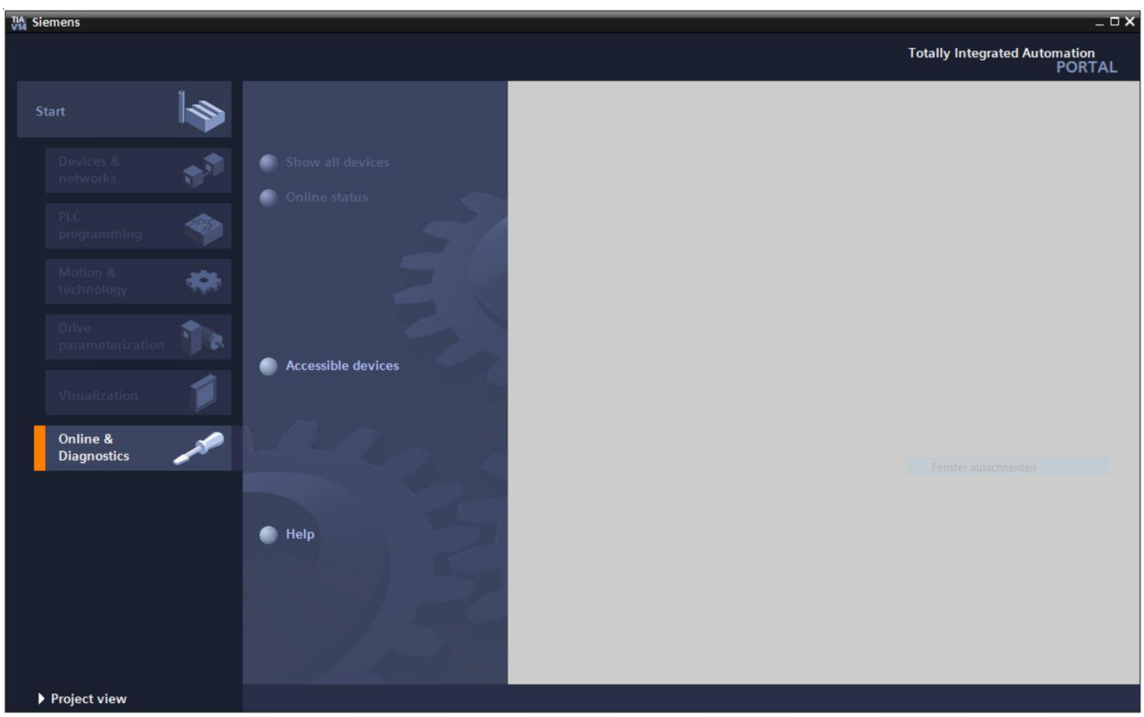


4.3.7 Set the IP address in the CPU

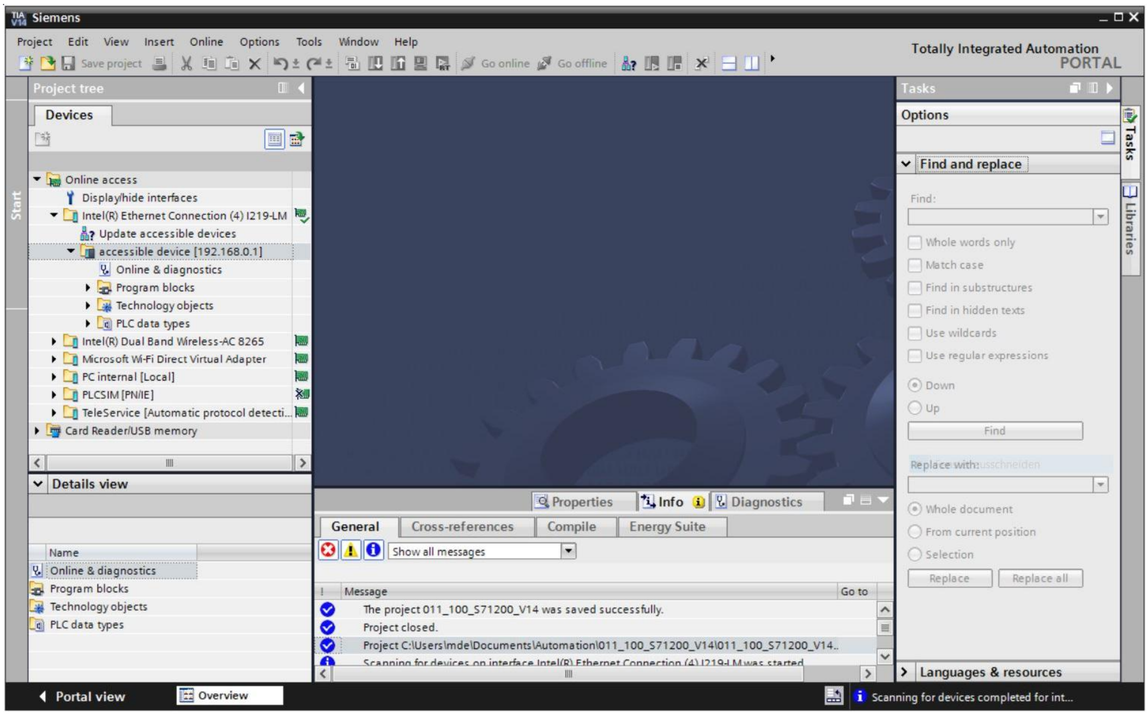
The IP address of SIMATIC S7-1200 is set as follows.

→ Select the Totally Integrated Automation Portal for this, which is opened here with a double-click (→ TIA Portal V14)

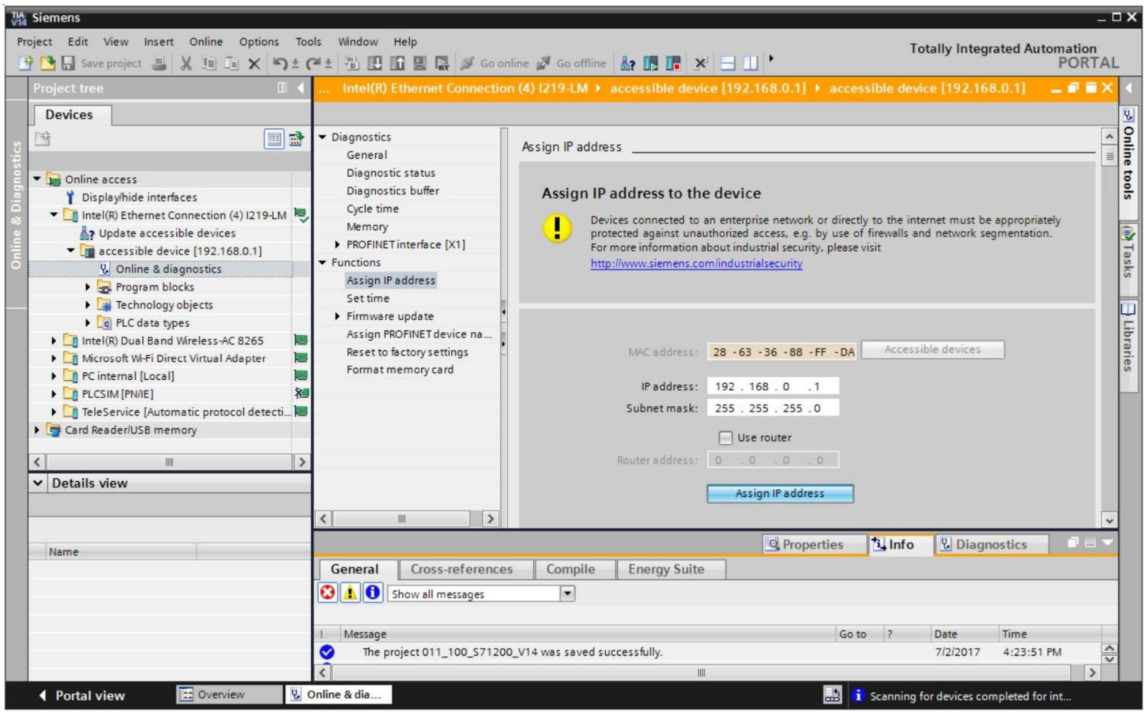
→ Select → "Online&Diagnostics", and open the →"project view".



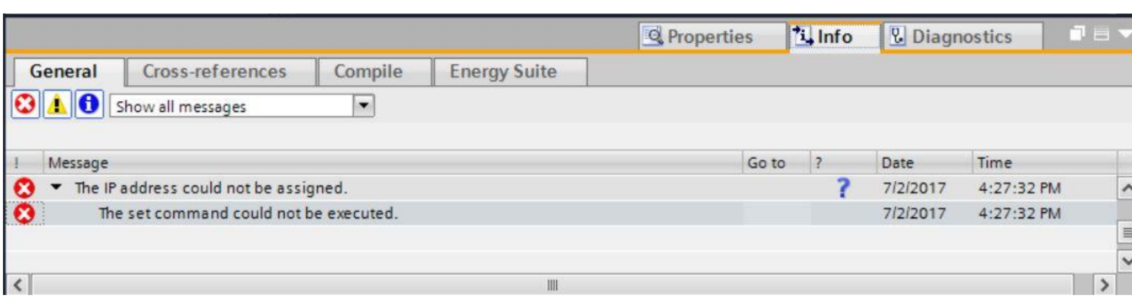
→ In the project tree under → "Online access", select the network adapter that was set previously. If you click → "Update accessible devices" here, you see the IP address (if previously set) or the MAC address (if IP address not yet assigned) of the connected SIMATIC S7-1200. Select → "Online&Diagnostics" here.



→ Under→ "Functions", you now find the → "Assign IP address" item. Enter the following IP address here (example): → IP address: 192.168.0.1 → Subnet mask 255.255.255.0. Click → "Assign IP address" and this new address will be assigned to your SIMATIC S7-1200.

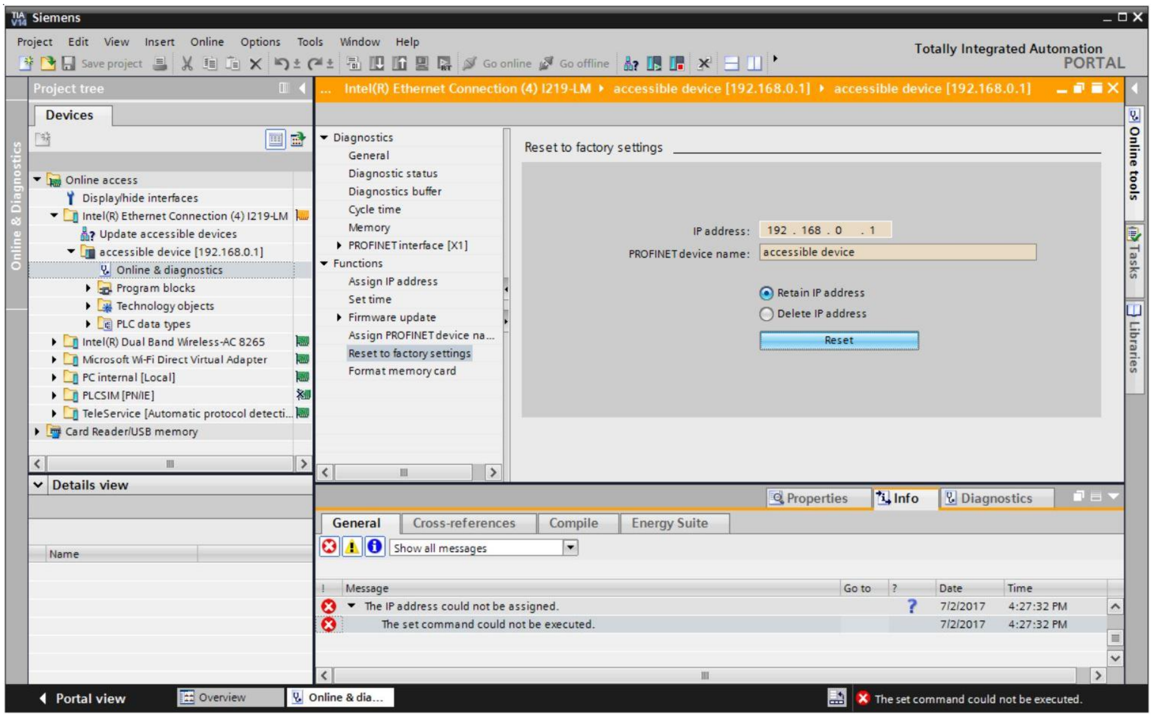


→ If the IP address was not successfully assigned, you will receive a message in the → "Info" window under → "General".

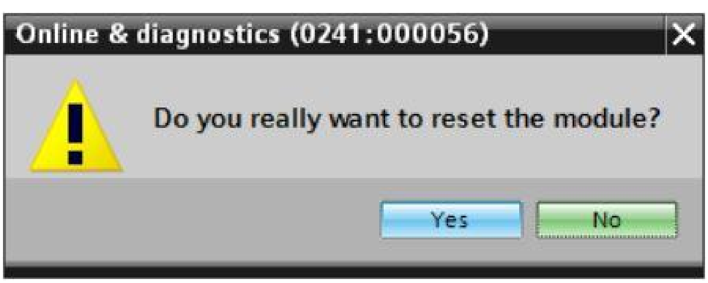


4.3.8 Restore the factory settings of the CPU

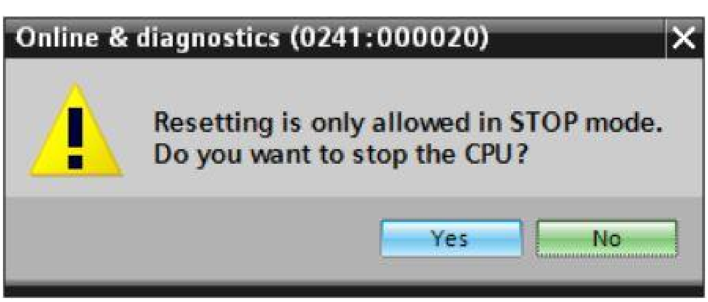
→ If the IP address could not be assigned, the program data on the CPU must be deleted. This is done by resetting the CPU. To reset the controller, select the → "Reset to factory settings" function and click → "Reset".



→ Confirm the prompt asking if you really want to reset the module with → "Yes".



→ If necessary, stop the CPU. (→ "Yes")



5 Task

Create a project und configure the compact CPU of your hardware, which corresponds to one part of the **SIMATIC S7-1200 with CPU 1214C** DC/DC/DC Trainer Package.

– SIMATIC S7-1200, CPU 1214C DC/DC/DC
  (Order number: 6ES7 214-1AG40-0XB0)

6 Planning

Because this is a new system, a new project must be created.

The hardware for this project is already specified with the SIMATIC S7-1200, CPU 1214C DC/DC/DC Trainer Package, Therefore, a selection does not have to be made. Instead, the listed CPU of the Trainer Package only has to be inserted in the project. To ensure that the correct module is inserted, the order number from the Task should be checked again directly on the installed device (see Table 1).

The Ethernet interface must be set for the configuration of the CPU. For the digital and analog inputs and outputs, the address areas corresponding to Table 1 will be set.

Module	Order number	Slot	Address area
CPU 1214C DC/DC/DC	6ES7 214-1AG40-0XB0	1	DI 0.0 -1.5 DQ 0.0 - 1.1 AI 64 / 66

Table 1: Overview of the planned configuration

As the final step, the hardware configuration must be compiled and downloaded. Any errors present can be detected during compiling and incorrect modules can be detected when the controller is started *(only possible when hardware is present and installed identically)*.

The tested project must be saved and archived.

3

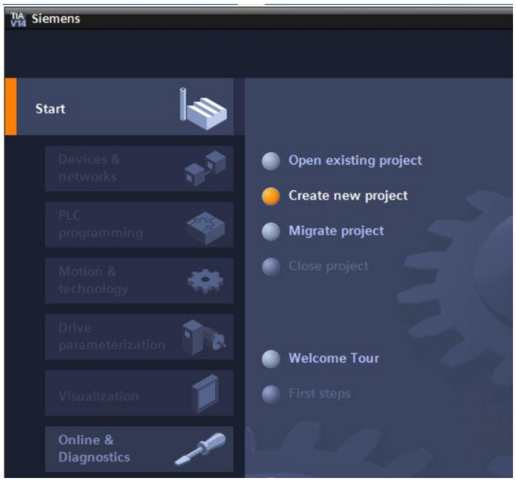
7 Structured step-by-step instructions

You can find instructions on how to carry out planning below. If you already have the relevant previous knowledge, it will be sufficient to focus on the numbered steps. Otherwise, simply follow the detailed steps in the instructions.

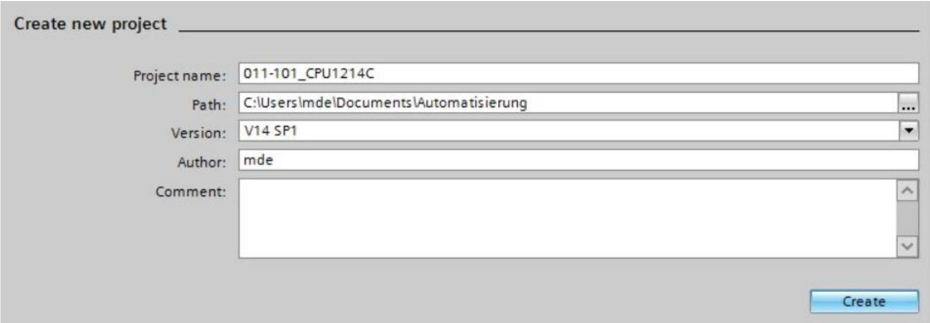
7.1 Create a new project

→ Select the Totally Integrated Automation Portal for this, which is opened here with a double-click (→ TIA Portal V14)

→ In the portal view under the "Start" menu, select the command → "Create new project".



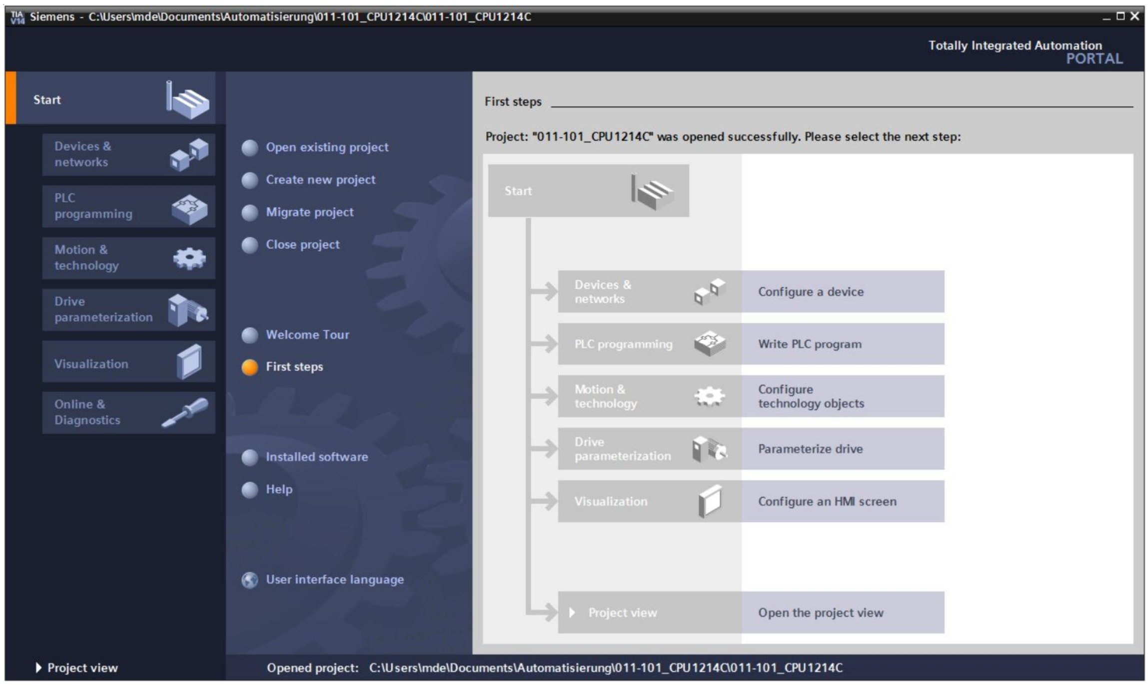
→ Modify Project name, Path, Author and Comment as appropriate and click → "Create".



→ The project will be created and opened and the "Start" menu, "First steps" will open automatically.

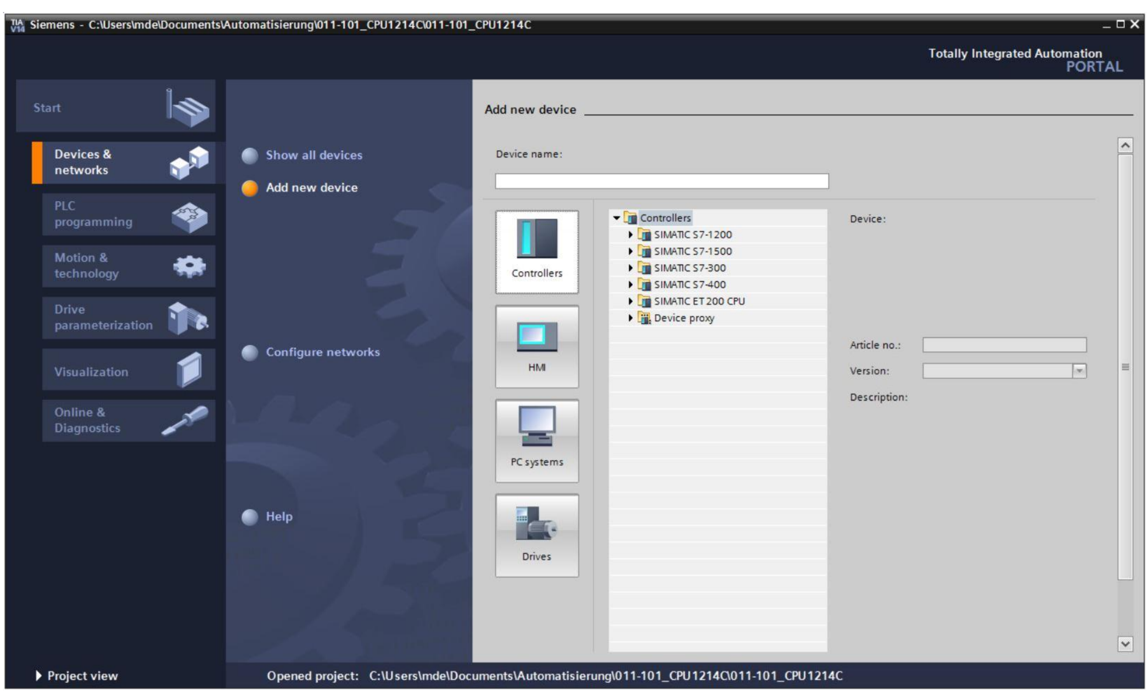
7.2 Insert the CPU 1214C DC/DC/DC

→ In the → "Start" portal, select → "First steps" → "Devices & Networks" → "Configure a device".



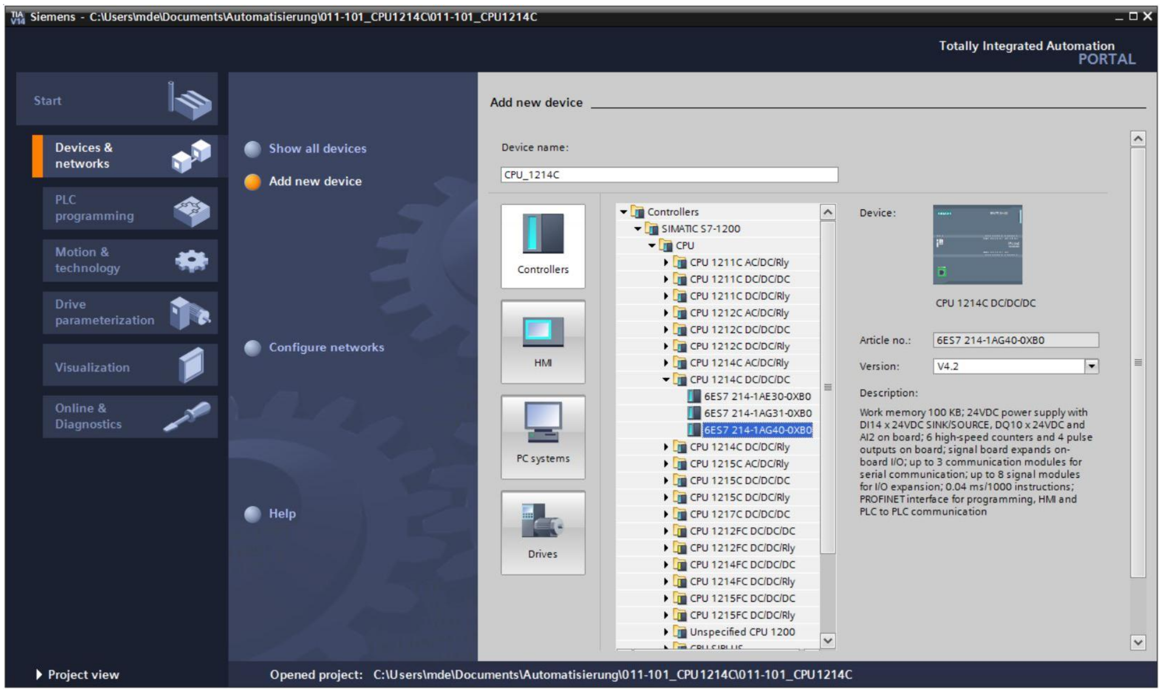
→ The "Show all devices" menu opens in the "Devices & Networks" portal.

→ Switch to the "Add new device" menu.



→ The specified model of the CPU will now be added as a new device.

→ (Controllers → SIMATIC S7-1200 → CPU → CPU 1214C DC/DC/DC→ 6ES7214-1AG40-0XB0 → V4.2)

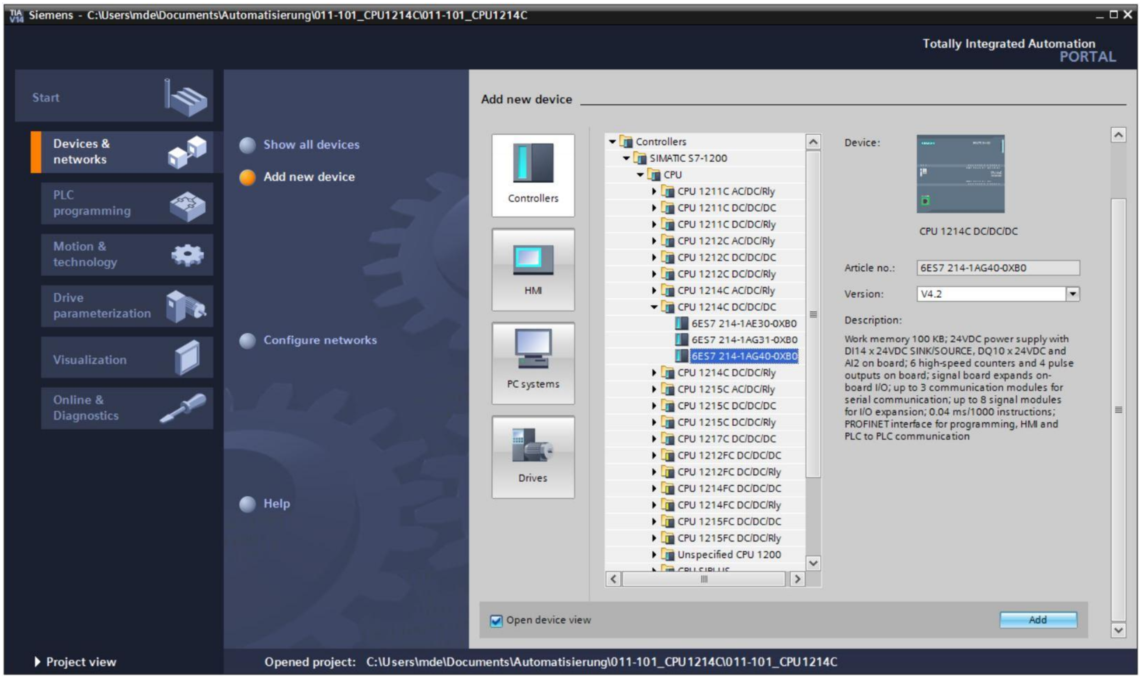


→ Assign a device name (Device name → "CPU_1214C").

→ Select "Open device view".

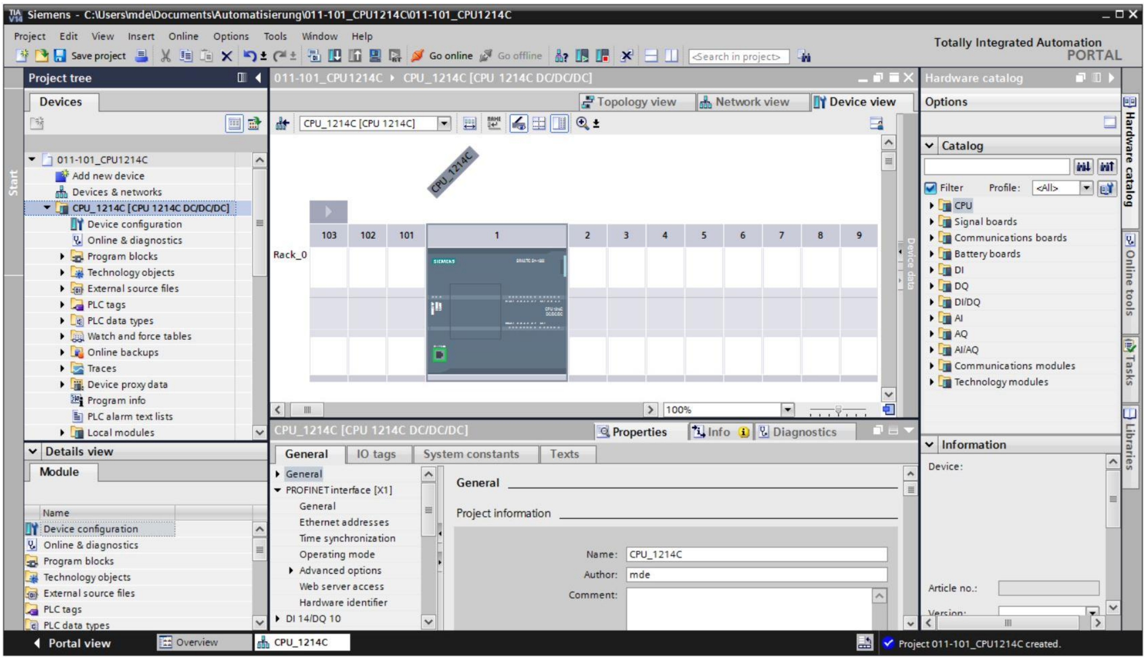
3

→ Click "Add".



Note: *The desired CPU may have multiple versions that differ in functionality (work memory, integrated memory, technology functions, etc.). In this case, you should ensure that the selected CPU meets the requirements placed on it.*

The TIA Portal now switches automatically to the project view and displays the selected CPU in the device configuration on slot 1 of a rail.

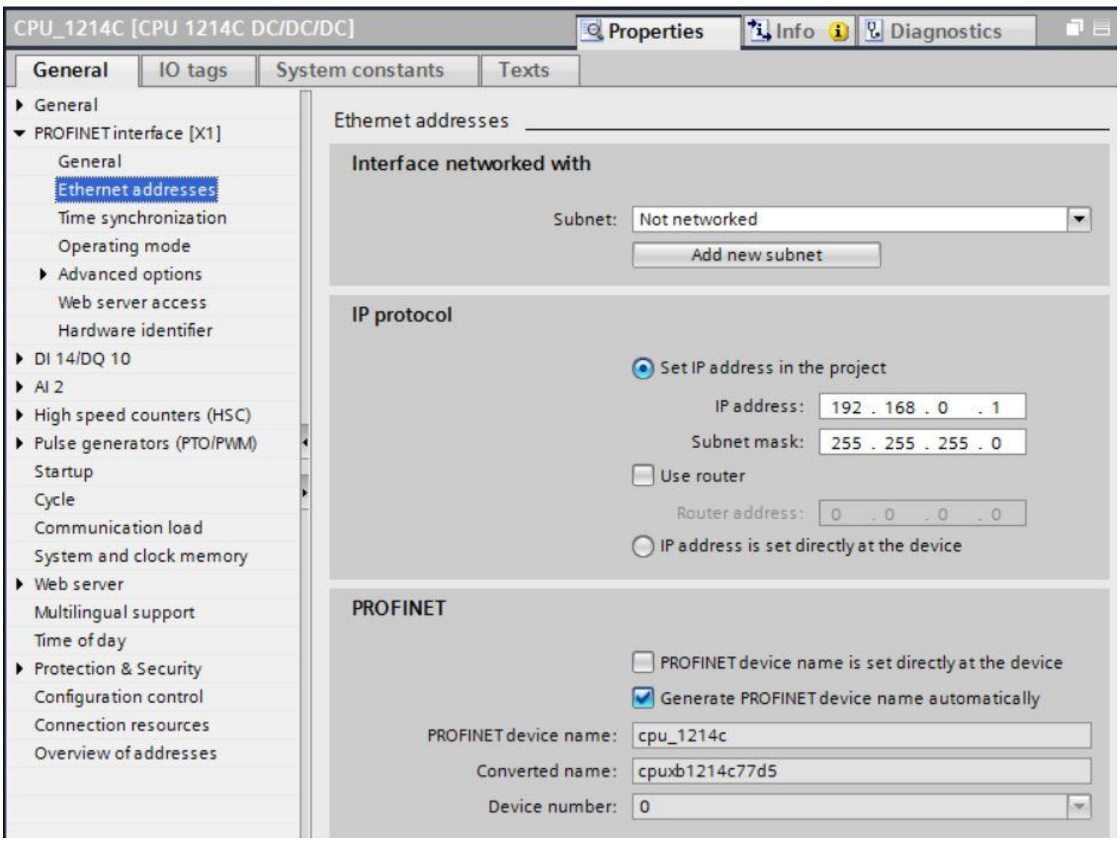


Note: *You can now configure the CPU according to your specifications there. Possible settings include the PROFINET interface, startup characteristics, cycle, password protection, communication load and many more.*

7.3 Configure the Ethernet interface of the CPU 1214C DC/DC/DC

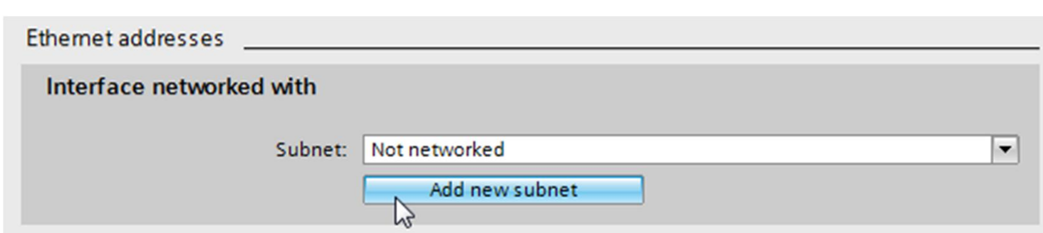
→ Select the CPU with a double-click

→ Under → "Properties", open the → "PROFINET interface [X1]" menu and select the → "Ethernet addresses" entry.

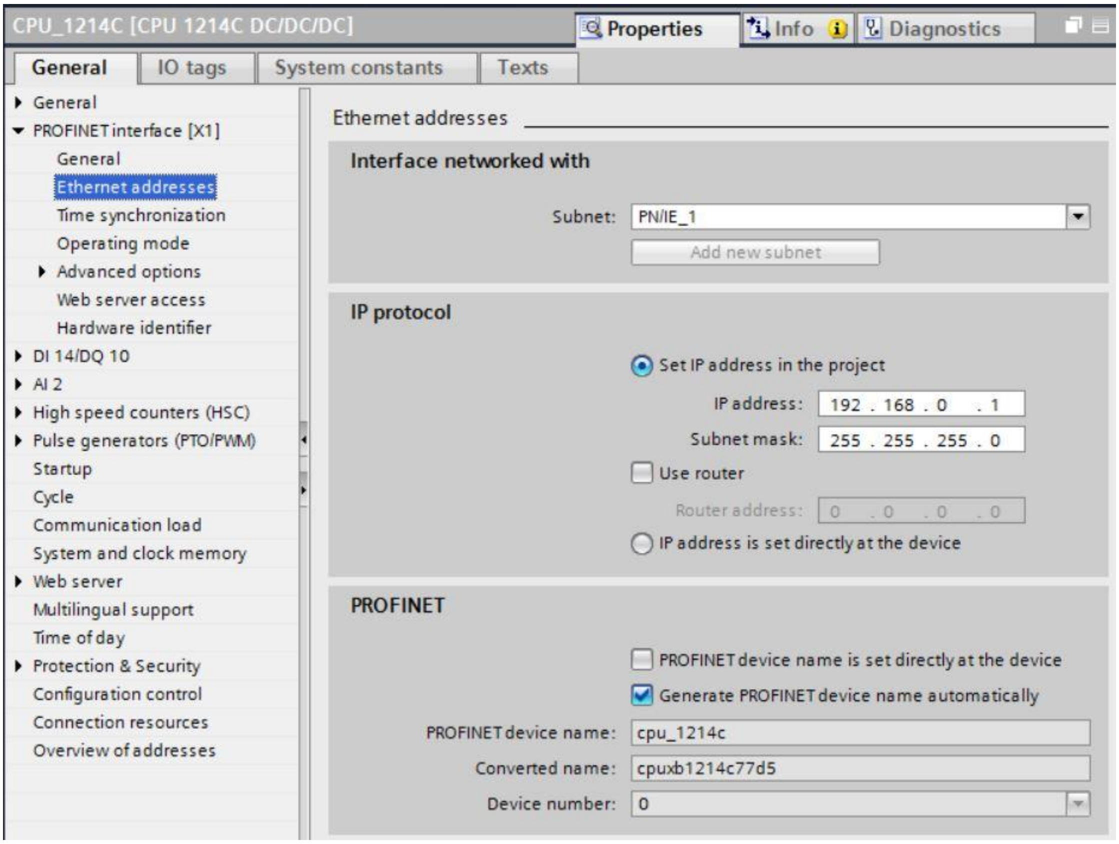


→ Under "Interface networked with", only the "Not networked" entry is available.

→ Add an Ethernet subnet with the → "Add new subnet" button.



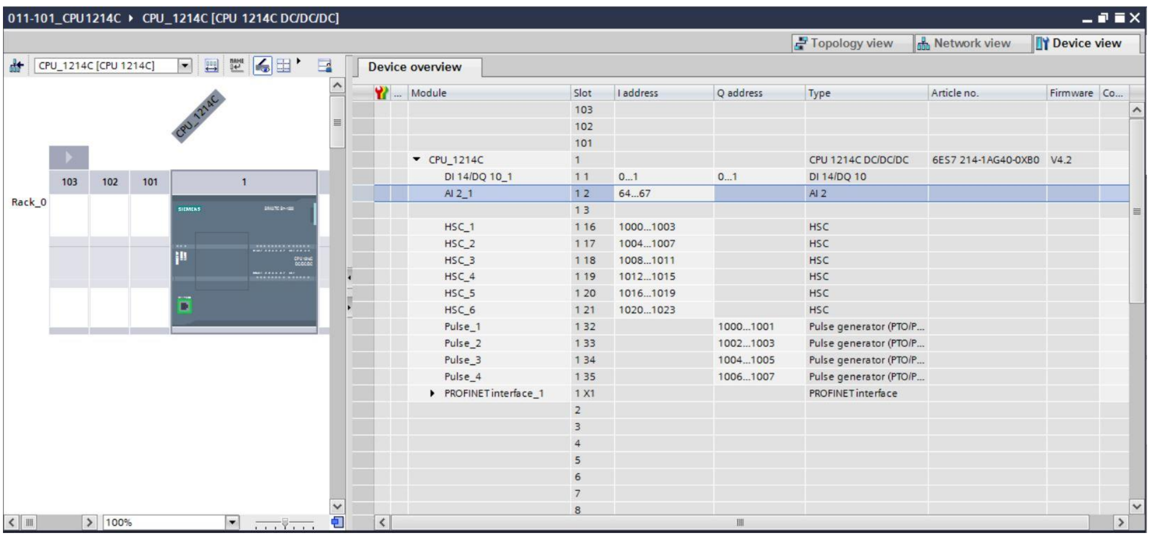
→ Keep the preassigned "IP address" and "Subnet mask".



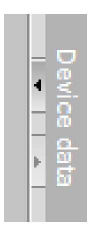
3

7.4 Configure the address areas

→ The next step is to check the address areas of the inputs and outputs and adapt them if necessary. DI/DO should have an address area of 0…1 and AI should have an address area of 64…67. (→ Device overview → DI 14/DQ 10_1 → I address: 0..1→ Q address: 0...1 → AI 2_1 → I address: 64…67)

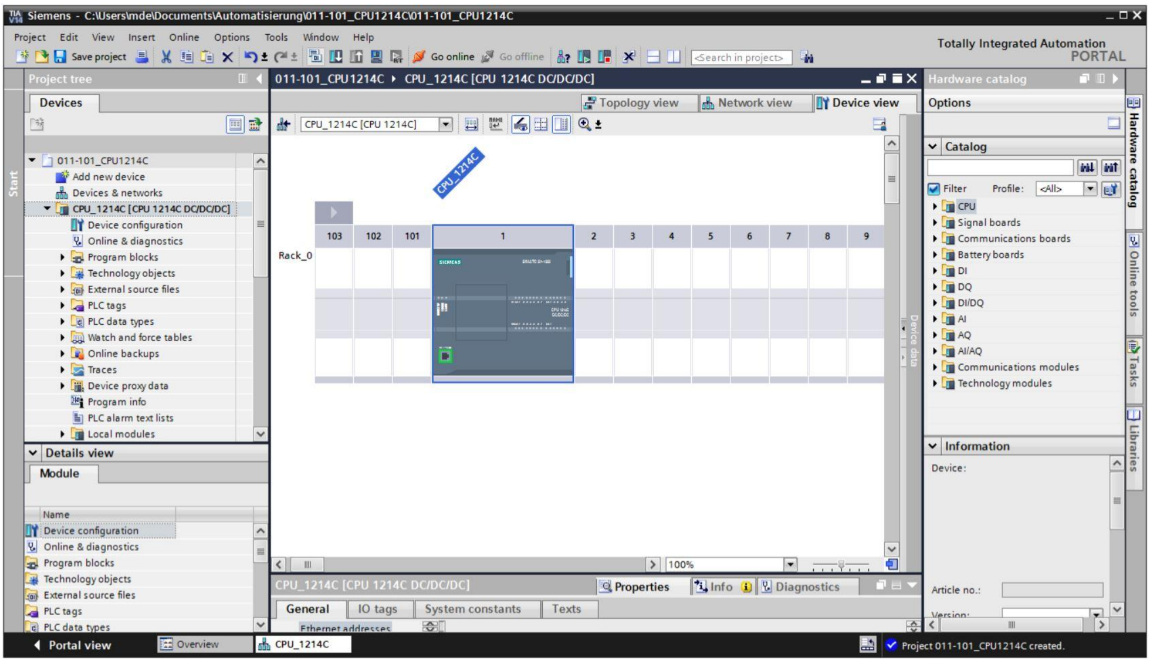


***Note:** To show and hide the Device overview, you need to click the small arrow next to "Device data" on the right side of the hardware configuration.*



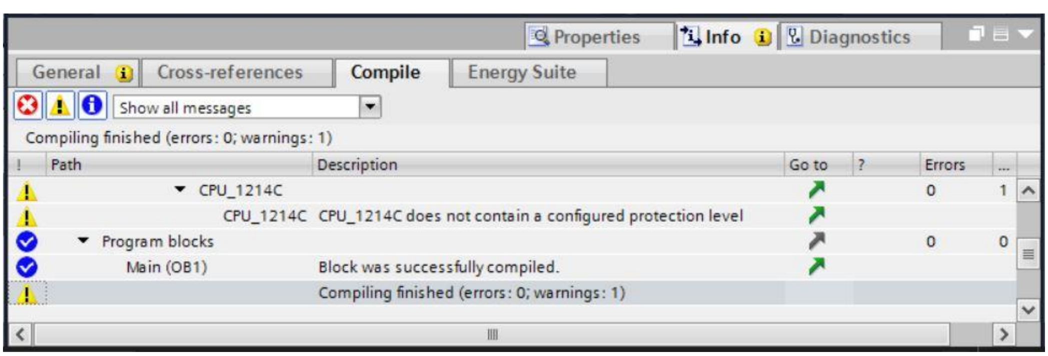
7.5 Save and compile the hardware configuration

→ Before you compile the configuration, you should save your project by clicking the → Save project button. To compile your CPU with the device configuration, first select the → "CPU_1214C [CPU1214C DC/DC/DC]" folder and click the → "Compile" icon.



***Note:** "Save project" should be used again and again when working on a project since this does not happen automatically. A prompt to save the project only occurs when the TIA Portal is closed.*

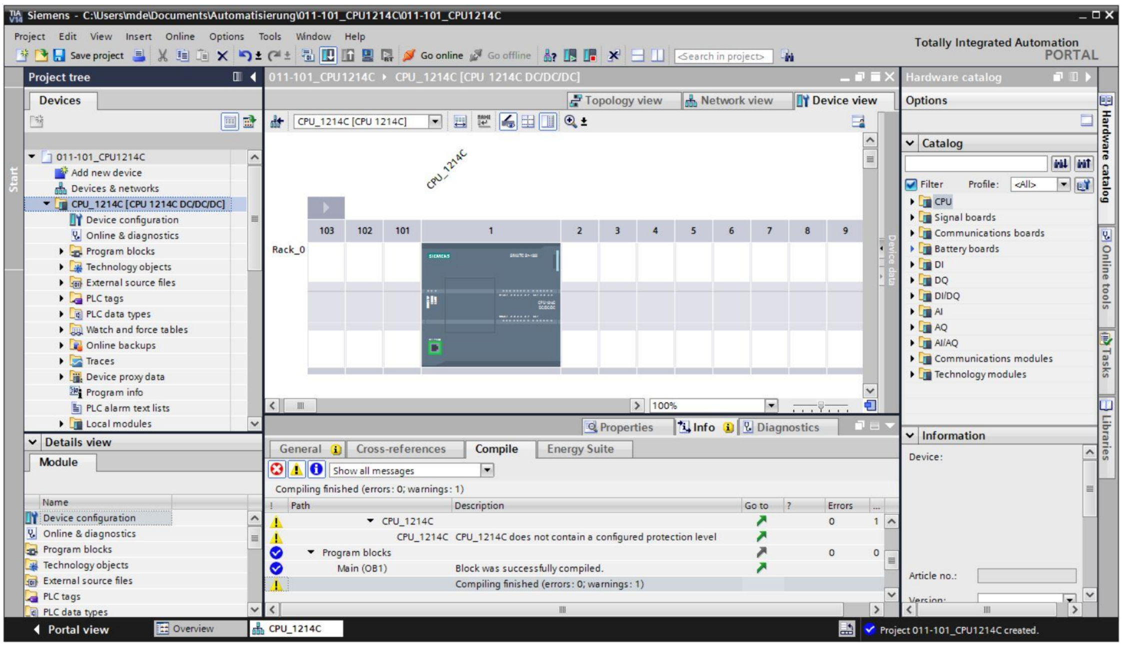
→ If the project was compiled without errors, you see the following screen.



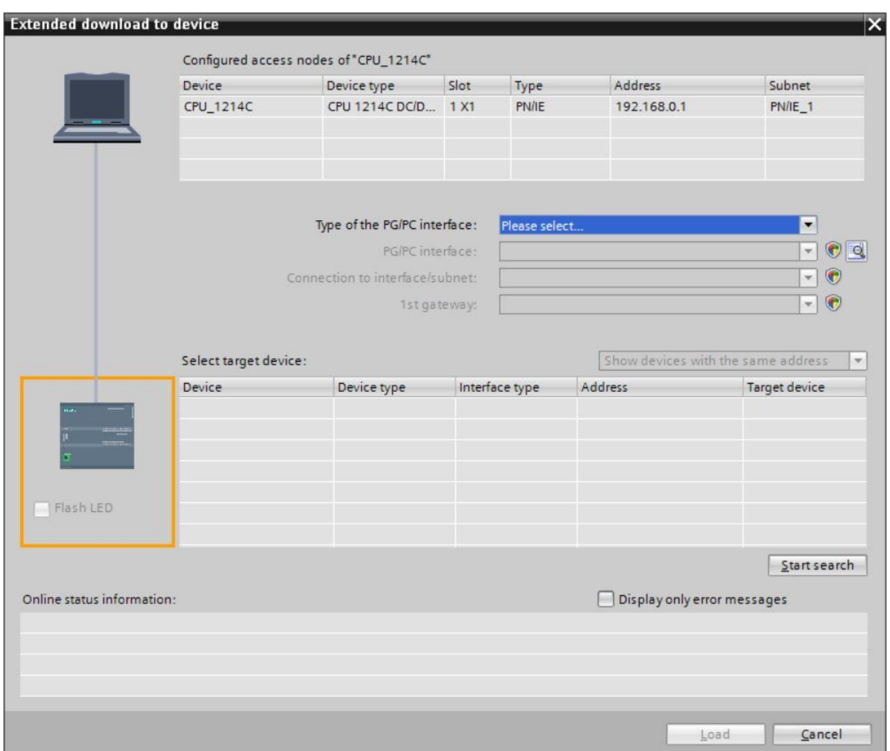
3

7.6 Download the hardware configuration to the device

→ To download your complete CPU, select the → "CPU_1214C [CPU1214C DC/DC/DC]" folder and click the → "Download" icon.

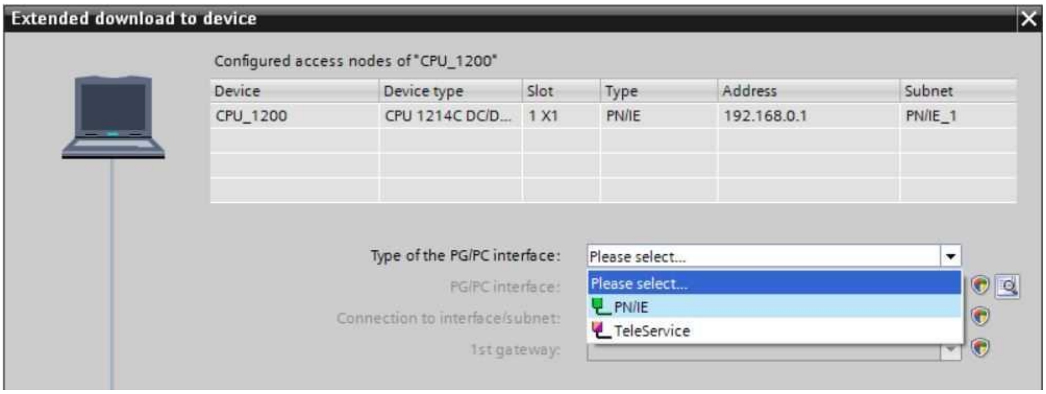


→ The manager for configuring the connection properties (extended download) opens.

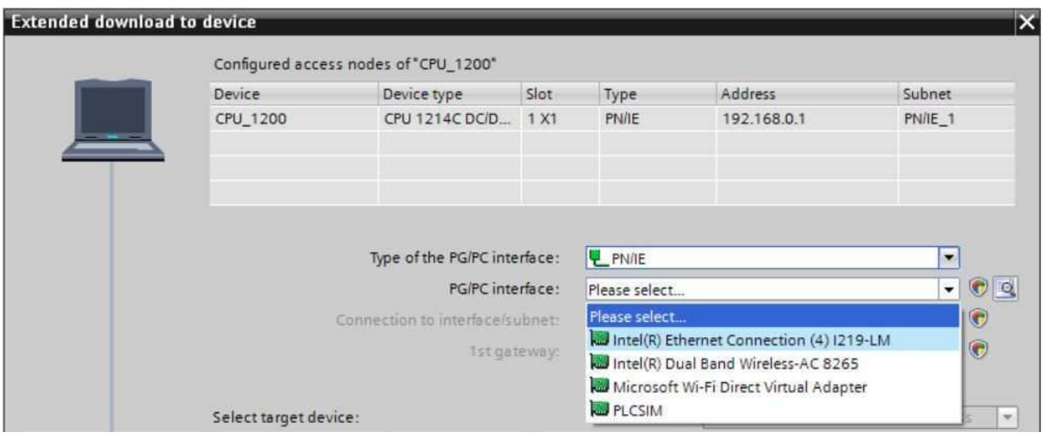


→ First, the interface must be correctly selected. This happens in three steps.

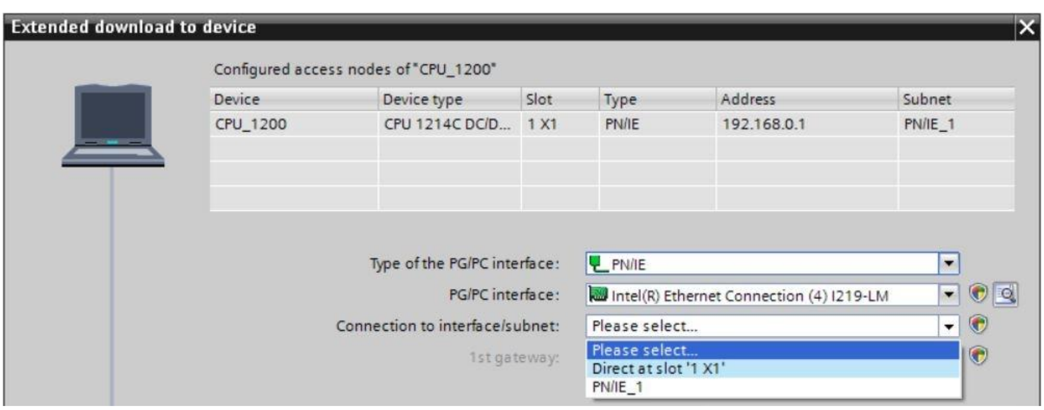
→ Type of the PG/PC interface → PN/IE



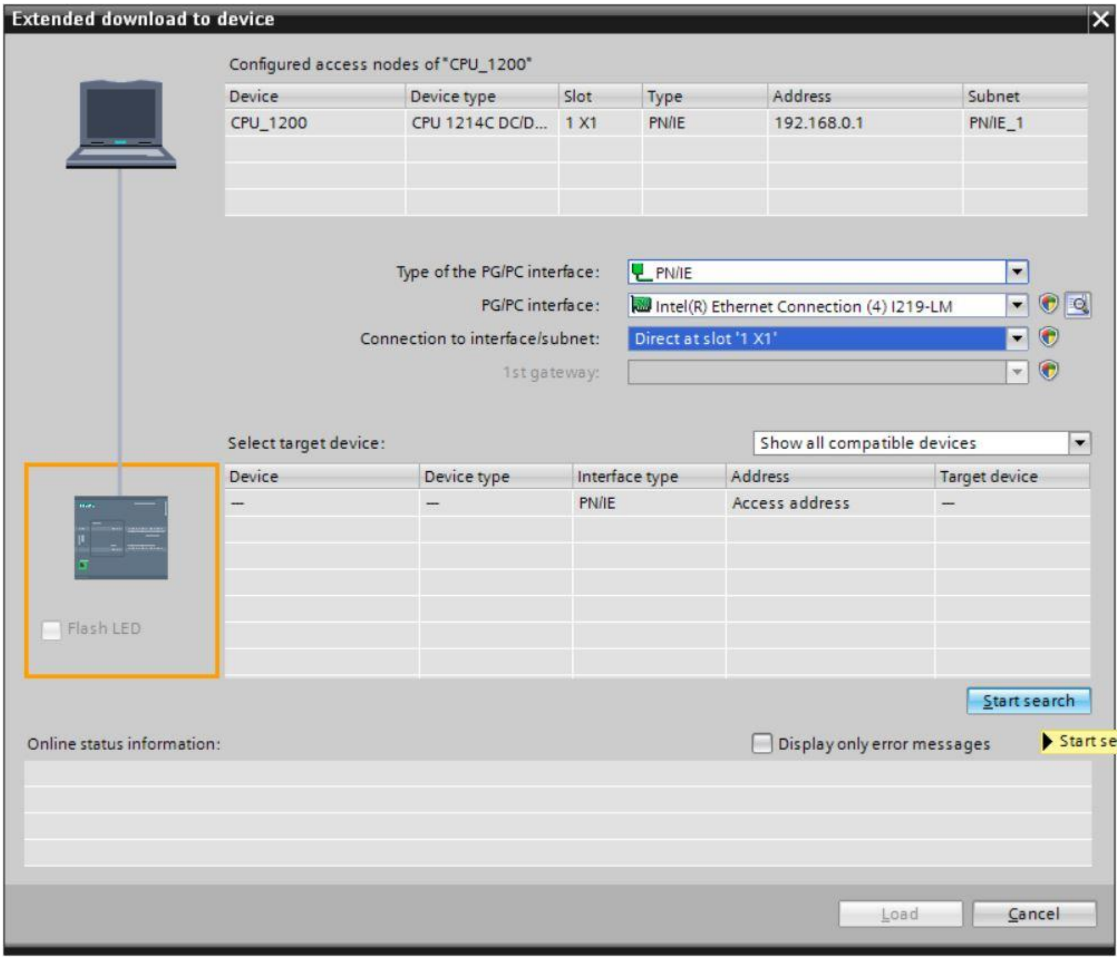
→ PG/PC interface → here: Intel(R) Ethernet Connection (4) I219-LM



→ Connection to interface/subnet → "PN/IE_1"

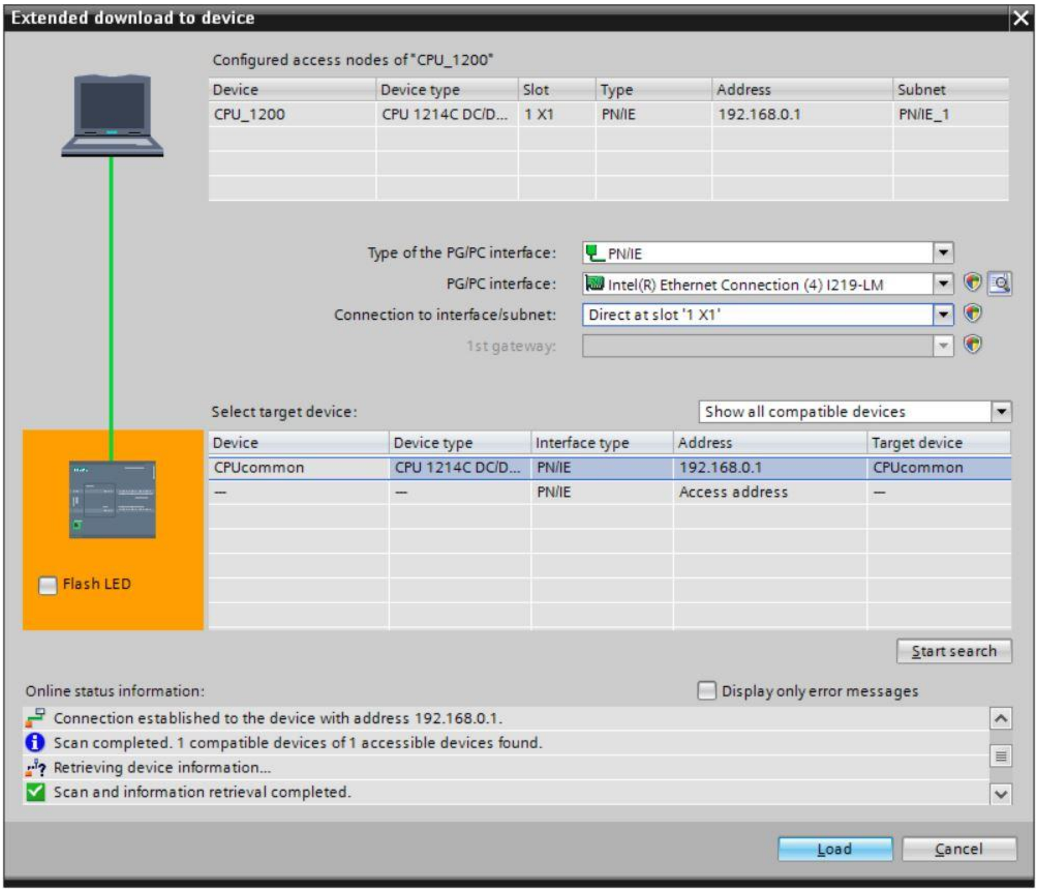


→ The → "Show all compatible devices" check box must be selected. The search for devices in the network is started by clicking the →  button.

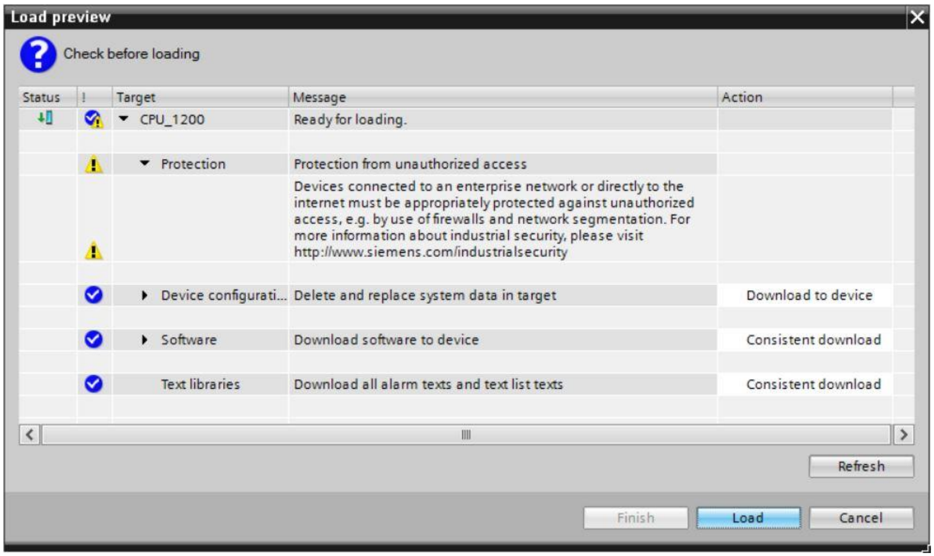


3

→ If your CPU is shown in the "Compatible devices in target subnet" list, you must select it. The download can then be started (→ CPU 1214C DC/DC/DC → "Load").

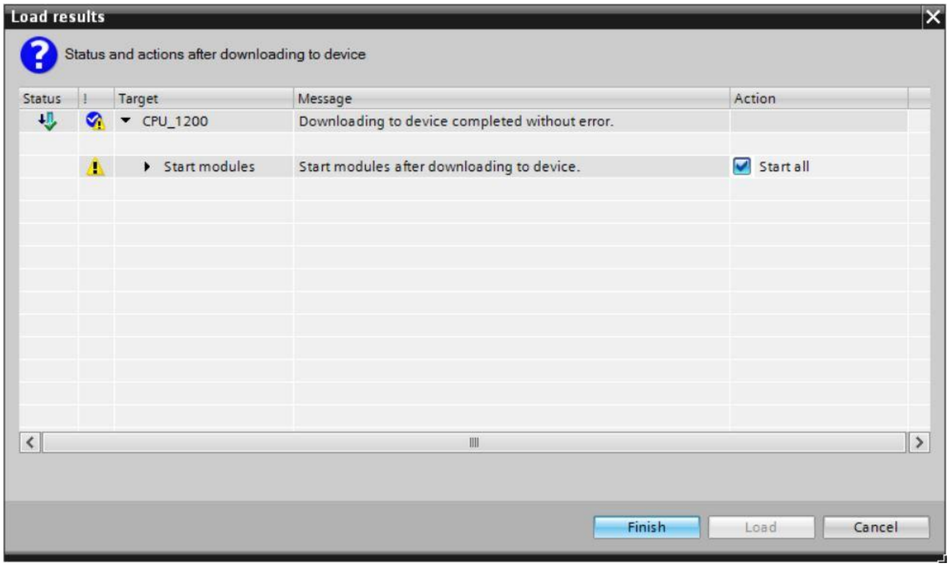


→ You first obtain a preview. Confirm the prompt → "Overwrite all" and continue with → "Load".

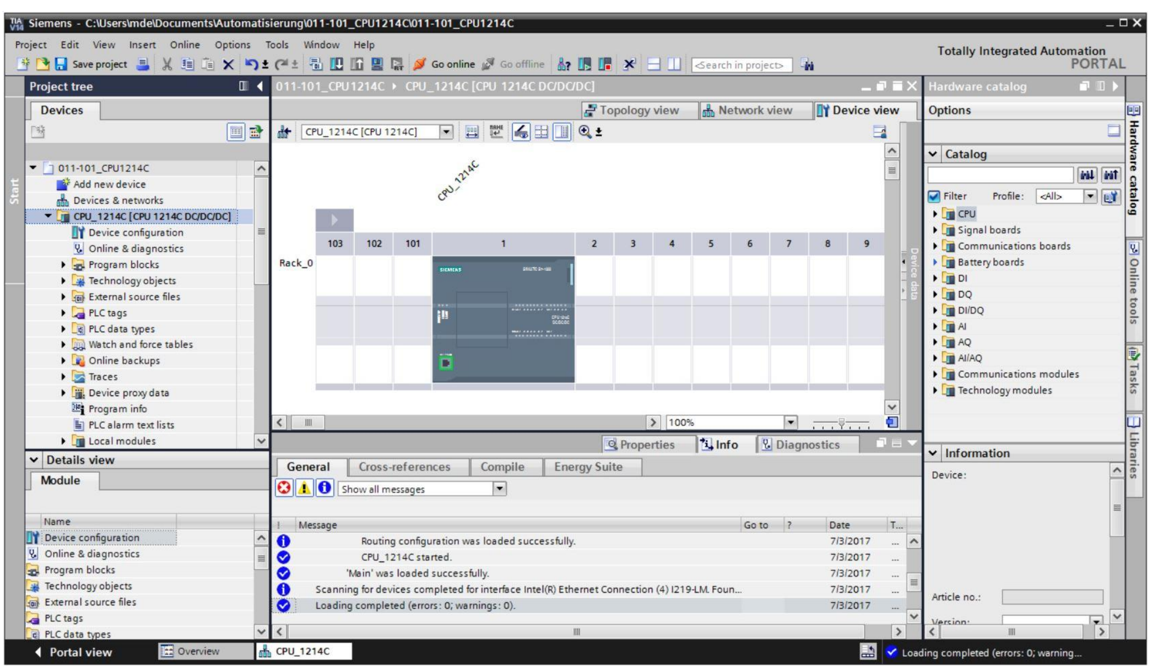


Note: *The symbol should be visible in every line of the "Load preview". You can find additional information in the "Message" column.*

The → "Start all" option will be selected next before the download operation can be completed with → "Finish".



→ After a successful download, the project view will open again automatically. A loading report appears in the information field under "General". This can be helpful when troubleshooting an unsuccessful download.

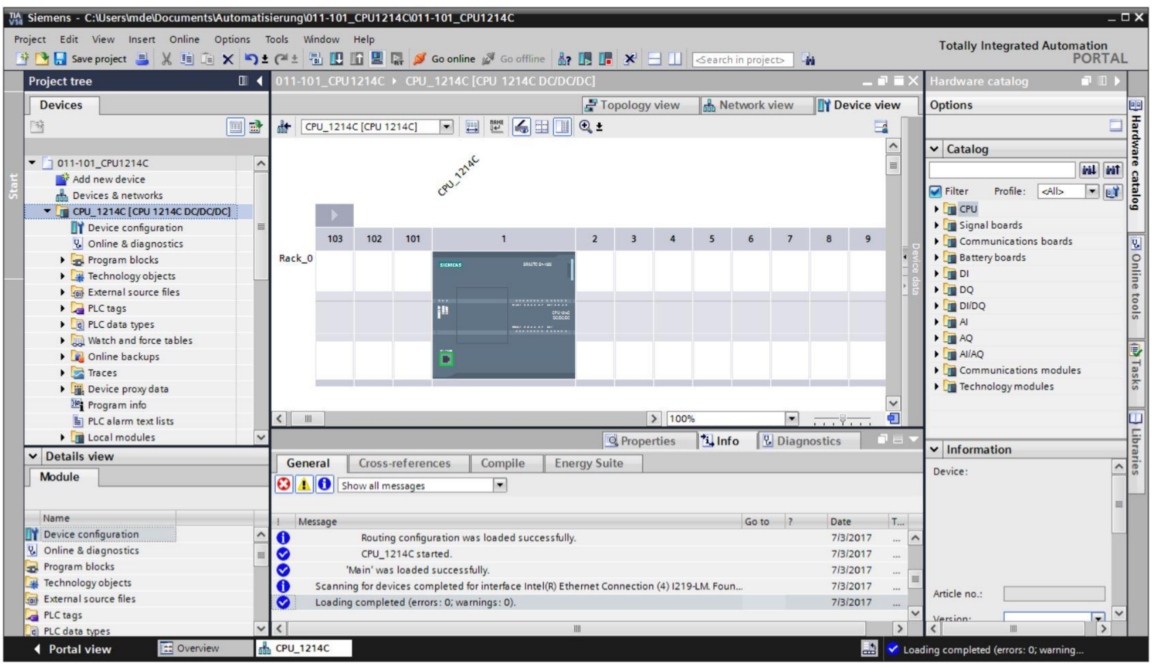


3

7.7 Download the hardware configuration to the PLCSIM simulation (optional)

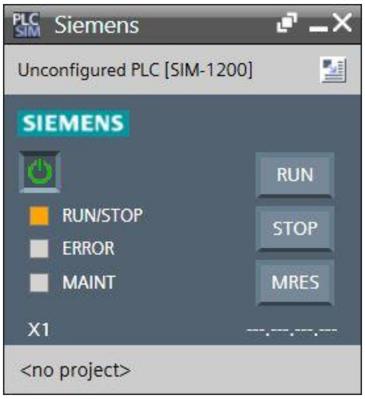
If no hardware is present, the hardware configuration can **alternatively** be downloaded to a PLC simulation (S7-PLCSIM).

→ To do so, you must first start the simulation by selecting the → "CPU_1214C [CPU1214C DC/DC/DC]" folder and clicking the → "Start simulation" icon.

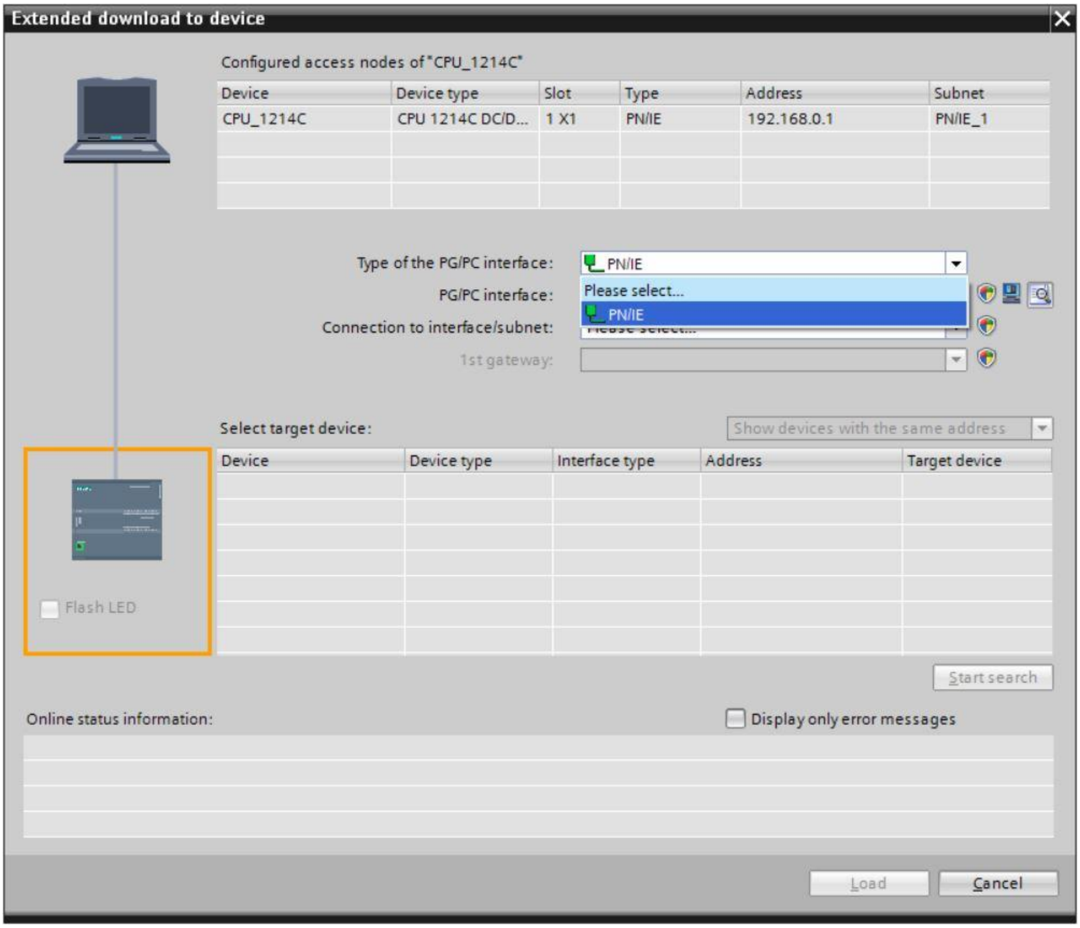


→ The prompt that all other online interfaces will be disabled is confirmed with → "OK".

→ The "S7 PLCSIM" software is started in a separate window in the compact view.

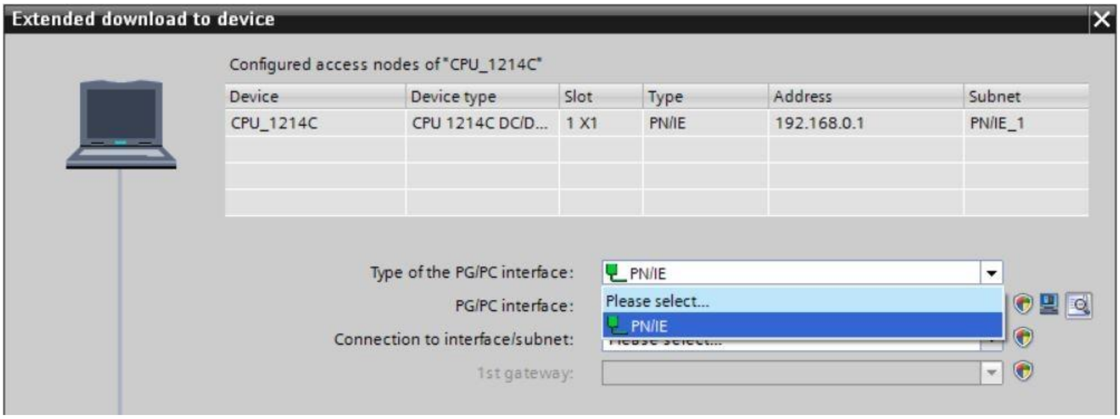


→ The manager for configuring the connection properties (extended download) opens shortly thereafter.

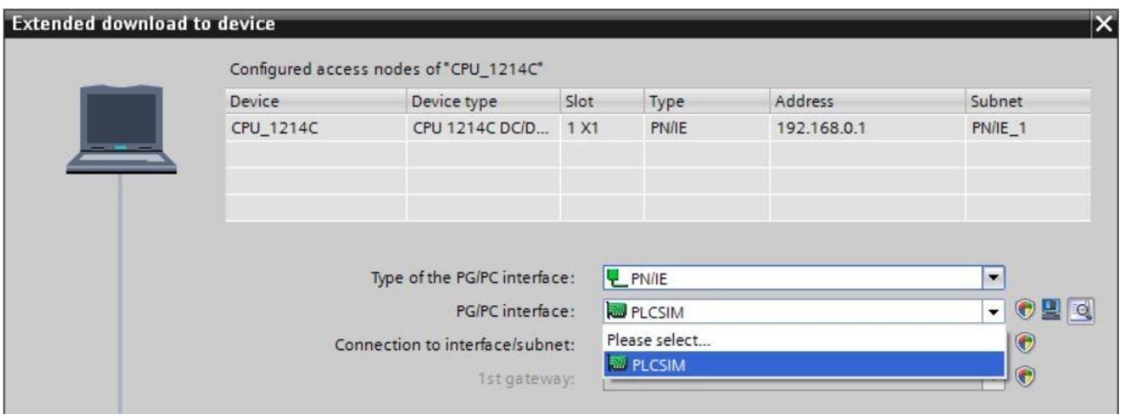


First, the interface must be correctly selected. This happens in three steps.

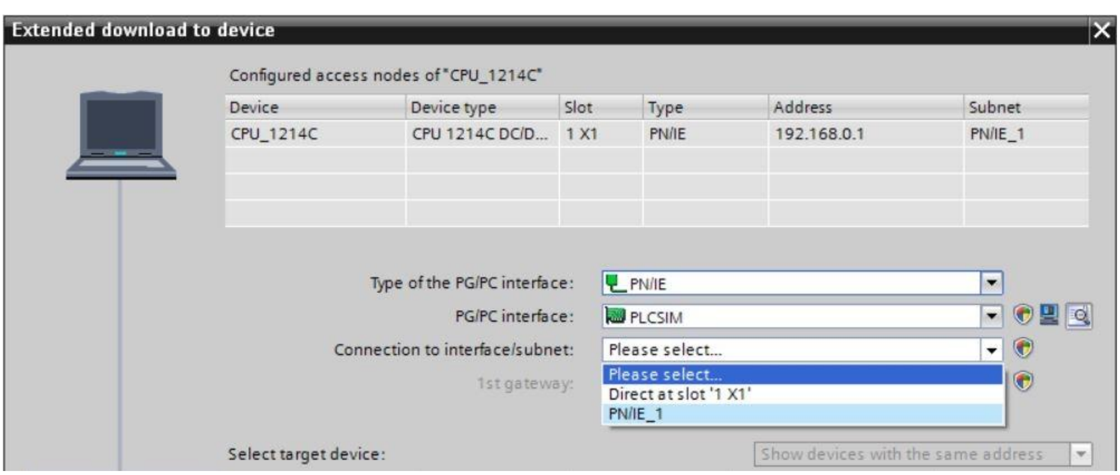
→ Type of the PG/PC interface → PN/IE



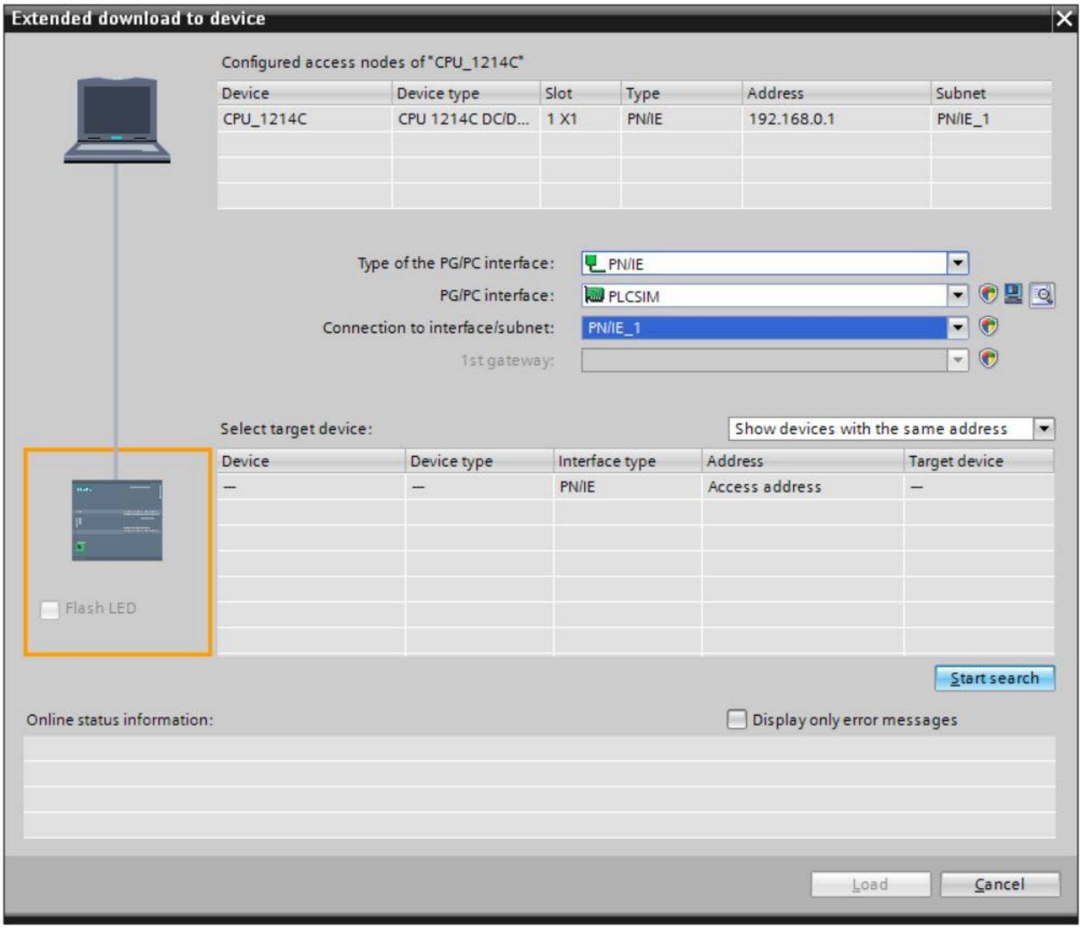
→ PG/PC interface → PLCSIM



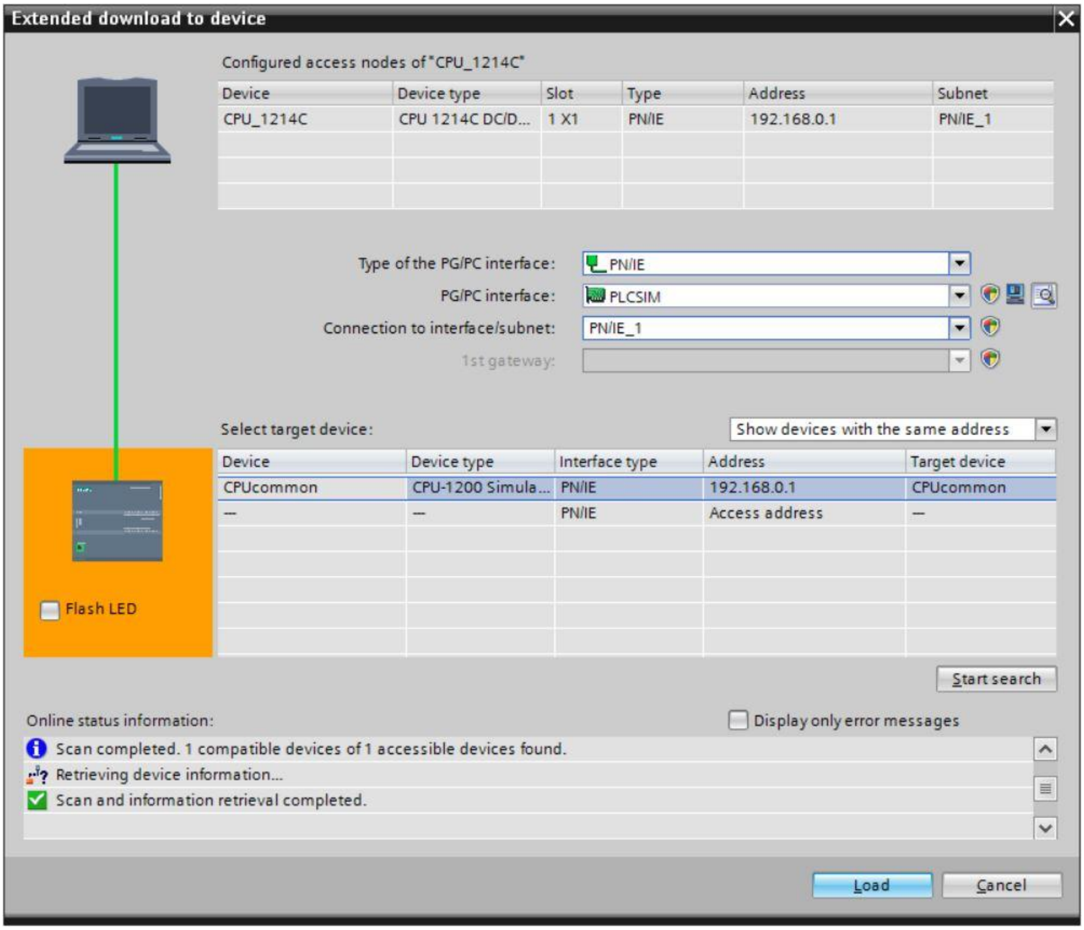
→ Connection to interface/subnet → "PN/IE_1"



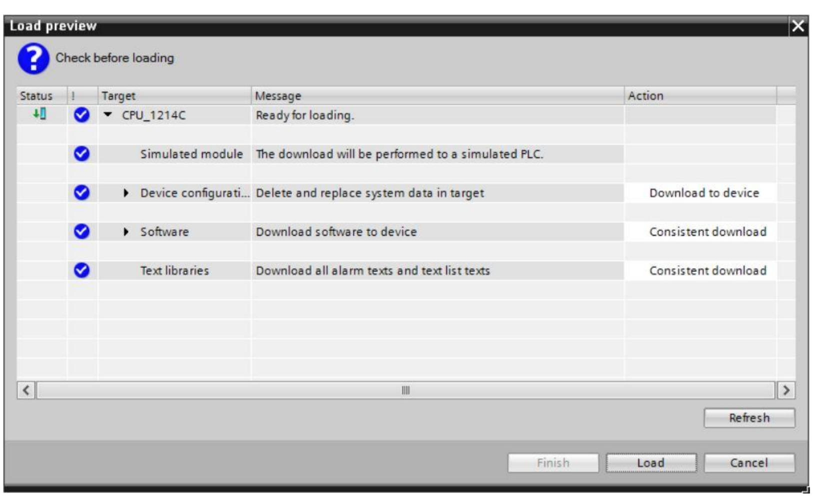
→ The → "Show all compatible devices" check box must be selected. The search for devices in the network is started by clicking the → Start search button.



→ If the simulation is shown in the "Compatible devices in target subnet" list, it must be selected before the download can be started (→ "CPU-1200 Simulation" → "Load")

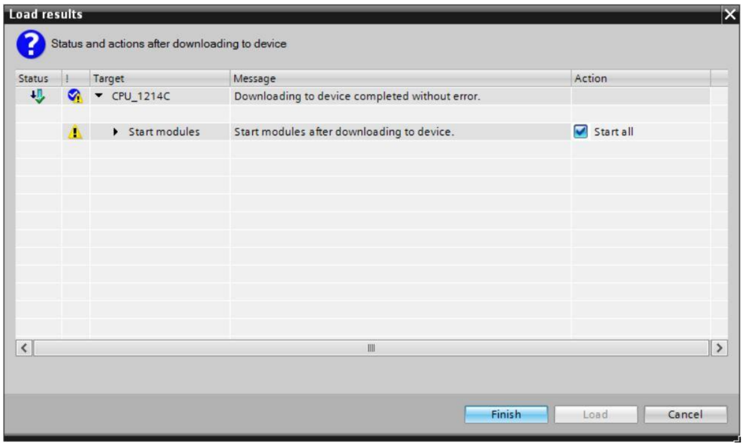


→ You first obtain a preview. Continue with → "Load".

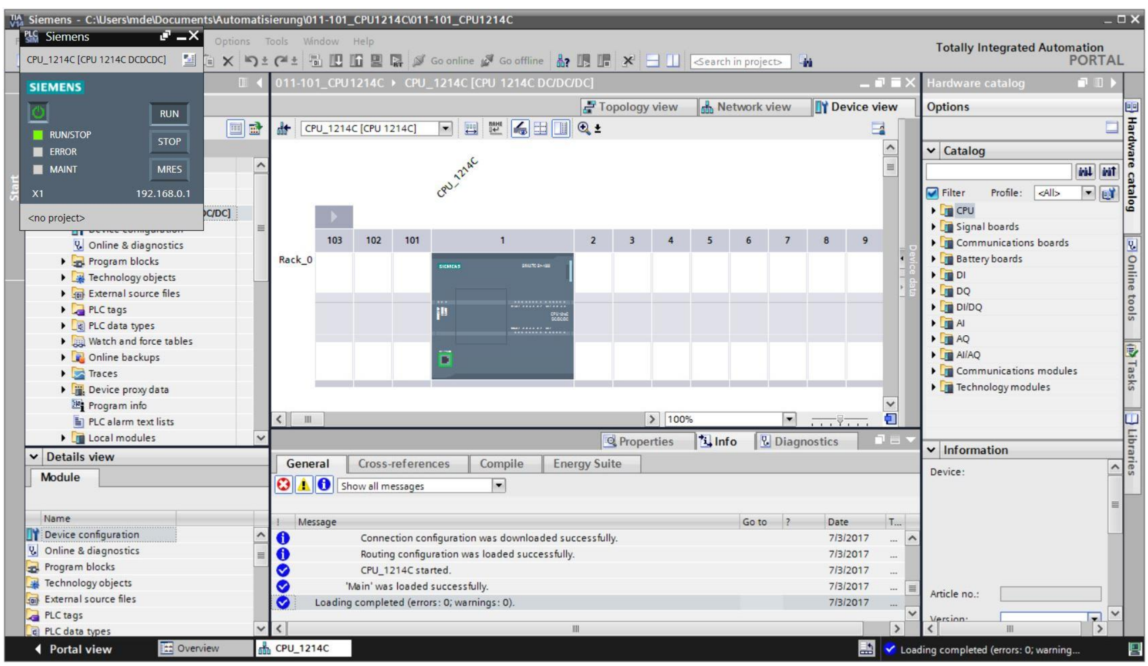


Note: *The symbol should be visible in every line of the "Load preview". You can find additional information in the "Message" column.*

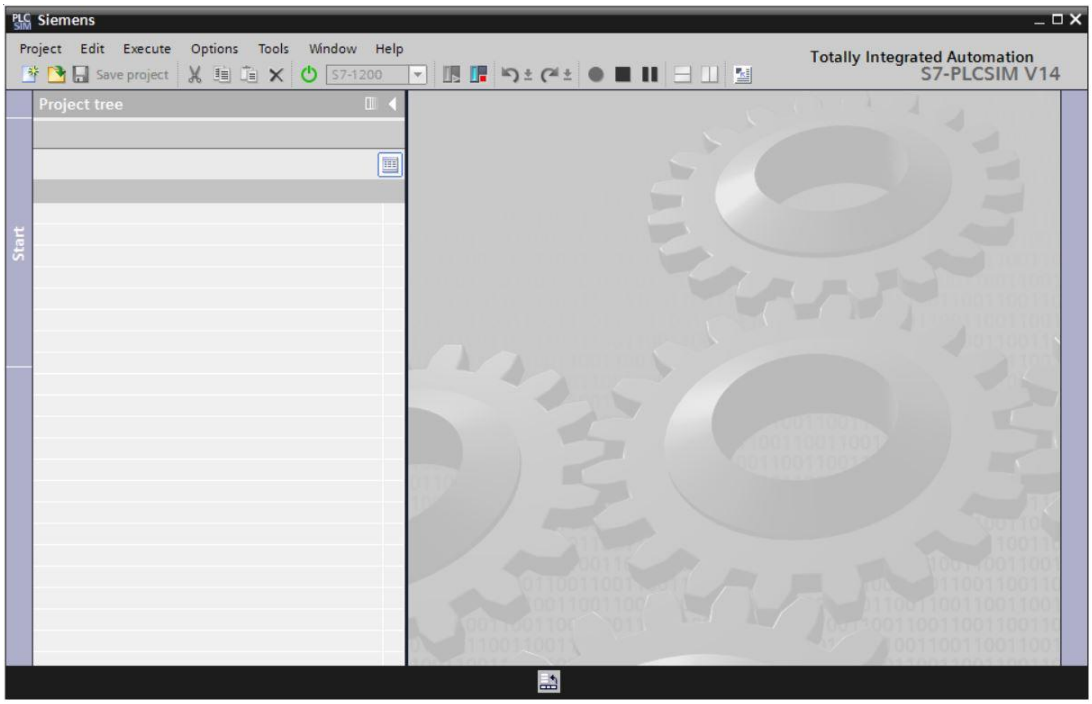
→ The → "Start all" option will be selected next before the download operation can be completed with → "Finish".



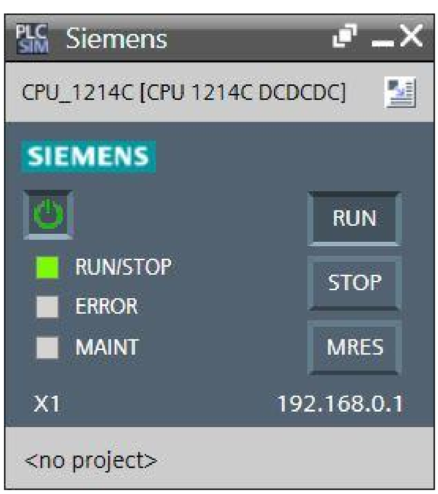
→ After a successful download, the project view will open again automatically. A loading report appears in the information field under "General". This can be helpful when troubleshooting an unsuccessful download.



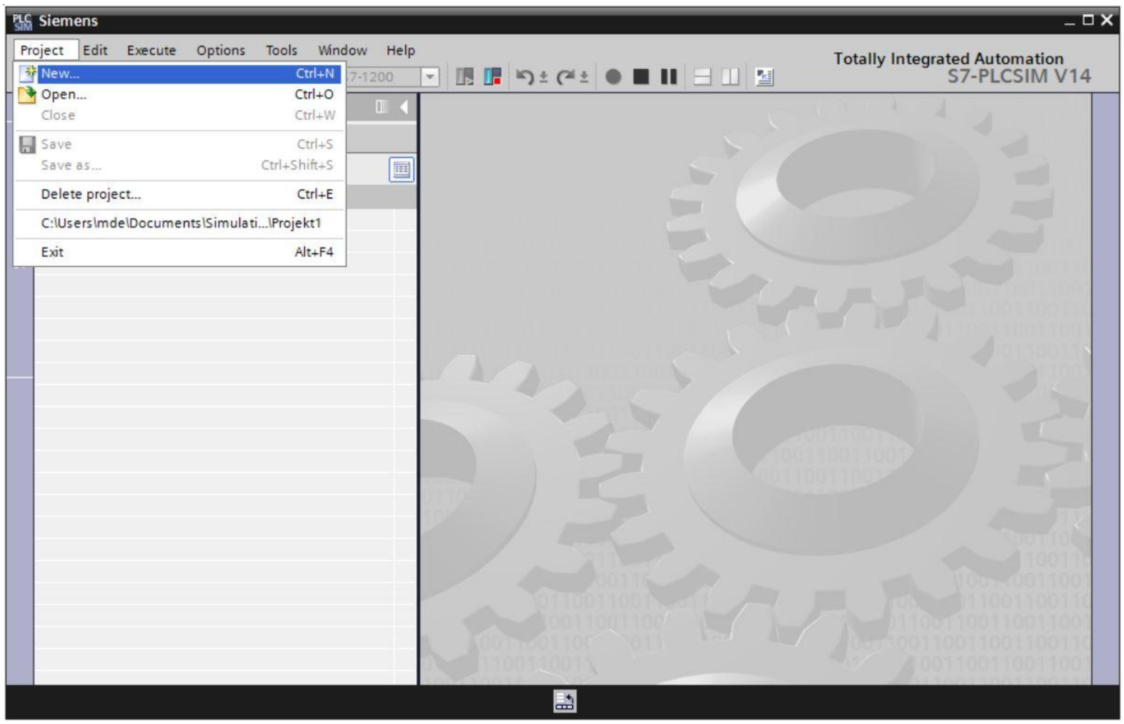
→ The PLCSIM simulation has the following appearance in the project view. You can switch to the compact view of the simulation by clicking the →  icon in the menu bar.



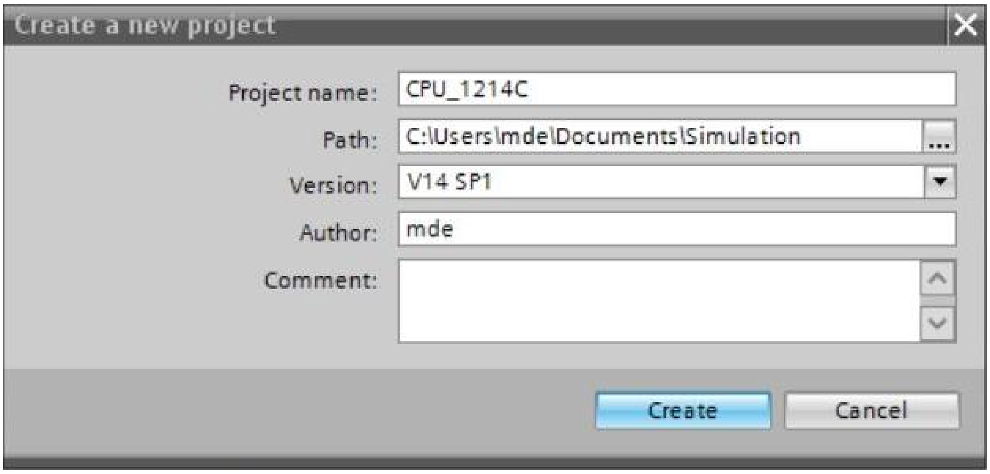
→ The compact view of the PLCSIM simulation has the following appearance. You can switch back to the project view by clicking the →  icon.



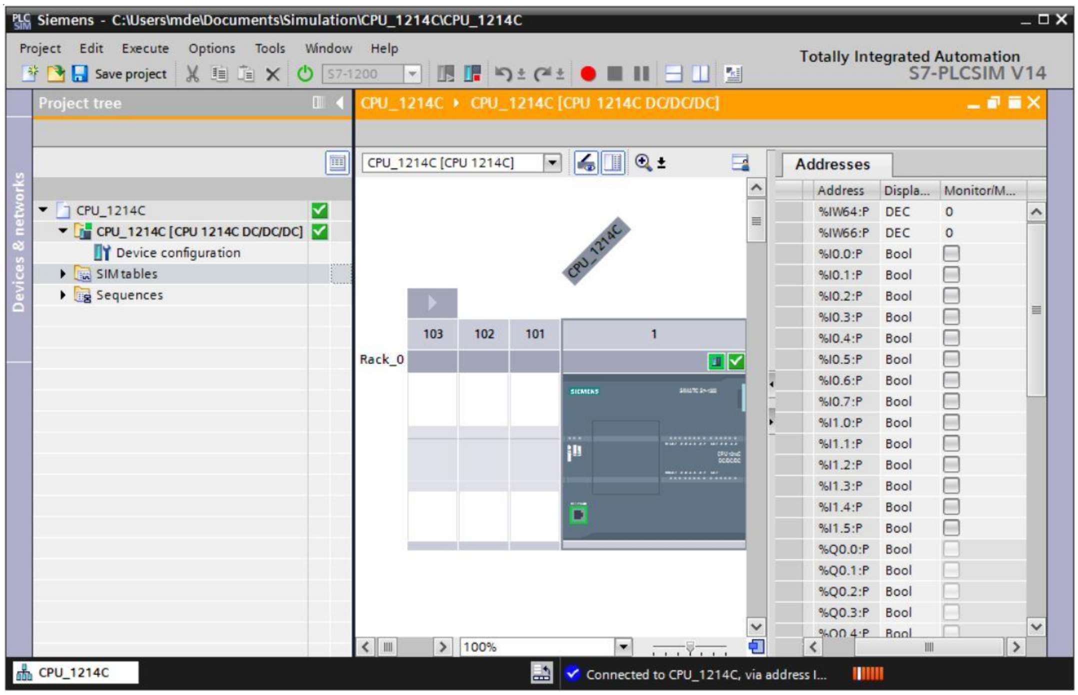
→ In the project view you have to create a new simulation project by clicking → "Project" → "New".



→ Assign a "Project name" → „CPU_1214C" and select a → "Path" where you want to create your project. Then click → „Create".



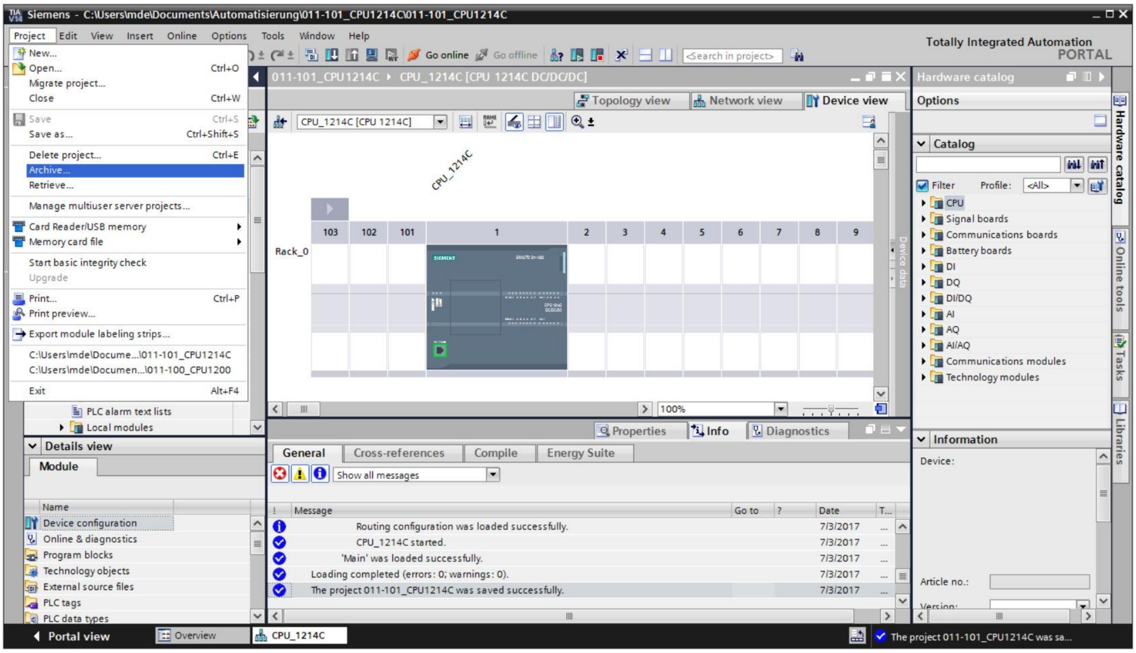
→ You can see the downloaded configuration with the status of all inputs and outputs in the project view by double-clicking → "Device configuration". Here you can also create your own → „Sim tables“ with selected input and output signals. You can modify the input signals used in your program to test the program in the PLCSIM simulation.



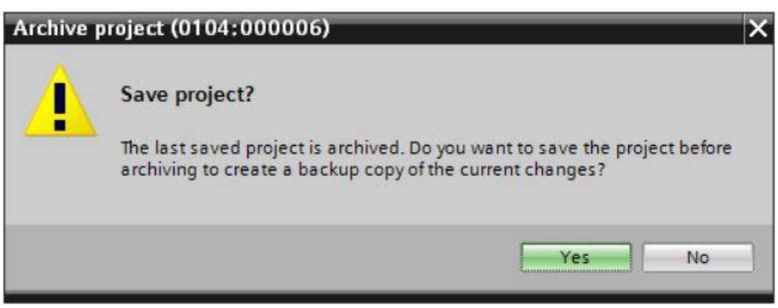
Note: *Because this is a simulation, you cannot detect errors in the hardware configuration in this case.*

7.8 Archive the project

→ To archive the project, select the → "Archive ..." command in the → "Project" menu.



→ Confirm the prompt to save the project with → "Yes".



→ Select a folder where you want to archive your project and save it as file type "TIA Portal project archive" (→ "TIA Portal project archive" → "SCE_EN_011-101_Hardware configuration_S7-1214C..." → "Save").

3

7.9 Checklist

No.	Description	Completed
1	Project was created	
2	Slot 1: CPU with correct order number	
3	Slot 1: CPU with correct firmware version	
4	Address area of the digital inputs correct	
5	Address area of the digital outputs correct	
6	Address area of the analog inputs correct	
7	Hardware configuration was compiled without error message	
8	Hardware configuration was downloaded without error message	
9	Project was successfully archived	

3

8 Exercise

8.1 Task – Exercise

The hardware configuration of the SIMATIC CPU 1214C DC/DC/DC Trainer Package is not quite complete. Insert the following missing signal board. Use the address area starting from 64 for the analog output.

– 1X SIMATIC S7-1200, signal board ANALOG OUTPUT SB1232, 1 AO (Order number: 6ES7 232-4HA30-0XB0)

8.2 Planning

Plan the implementation of the task on your own.

3

8.3 Checklist – Exercise

No.	Description	Completed
1	Signal board with correct order number	
2	Signal board with correct firmware version	
3	Signal board address area of analog output correct	
4	Hardware configuration was compiled without error message	
5	Hardware configuration was downloaded without error message	
6	Project was successfully archived	

9 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

3

Notes

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules from Version V14 SP1

Continued training

For regional Siemens SCE continued training, get in touch with your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

4

Information regarding use

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

Table of contents

4

Process Description - Sorting Station

The "Sorting station" example process is described in the following.

1 Description of functions

1.1 Brief description

The automated sorting station (see Figure 1) is used to separate plastic and metal components. A component is fed to the conveyor via a chute. The conveyor starts as soon as the component has been detected. If a component made of metal is on the conveyor, it is detected, transported up to the height of the metal magazine and pushed by a cylinder into the metal magazine. If no metal is detected, the component is made of plastic. The plastic component is transported to the end of the belt, where it falls into the plastic magazine. As soon as a component is sorted, the next component can be fed.

1.2 Technology diagram

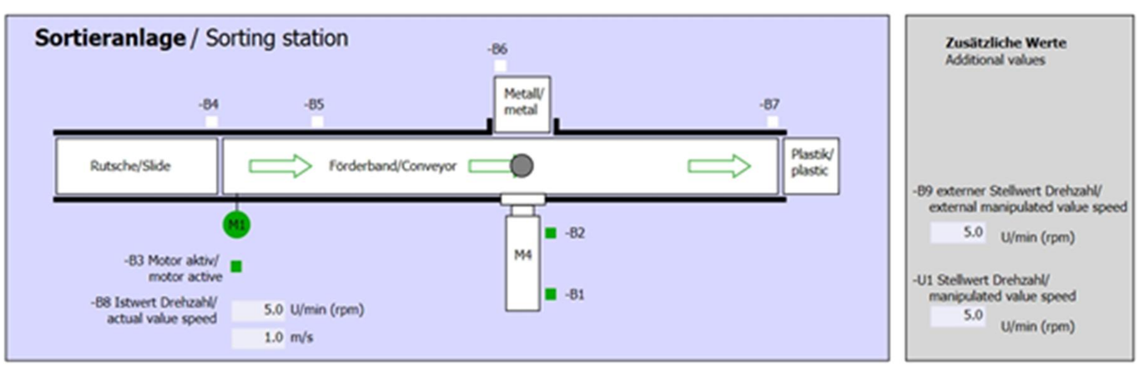


Figure 1: Technology diagram

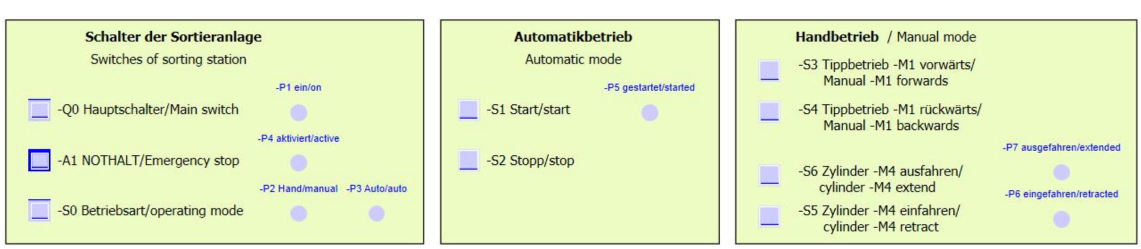


Figure 2: Control panel

4

1.3 Switching on

The station is switched on with the main switch Q0. Relay K0 (main switch "ON") is energized and provides the supply voltage for the sensors and actuators.

This operating state is indicated by indicator light P1 (main switch on)

1.4 Operating mode selection

Once the station has been switched on, two operating modes are possible: manual mode or automatic mode. The operating mode is selected using switch S0.

The selected operating mode is indicated by indicator lights P2 (manual mode) and P3 (automatic mode).

1.5 EMERGENCY STOP

In the absence of feedback from the EMERGENCY STOP (A1), all drives must be stopped immediately.

When feedback from the EMERGENCY STOP function is present again, the station may only start up again after another start signal.

Activation of the EMERGENCY STOP is indicated by indicator lights P4 (EMERGENCY STOP activated).

1.6 Manual mode

The station is set up in manual mode.

1.6.1 Retracting and extending the cylinder

After pushbutton S5 (cylinder M4 extend) is pressed, cylinder M4 is extended. When the front end position is reached (extended position), the cylinder pauses in this position. After pushbutton S4 is pressed, the cylinder retracts. A change of direction is to be possible at any time. When the two pushbuttons are pressed simultaneously, no motion should take place.

1.6.2 Conveyor motor in manual mode

With pushbutton S3 (pushbutton manual mode conveyor M1 forwards), motor Q1 (conveyor motor M1 forwards fixed speed) is moved forward in manual mode. With pushbutton S4 (pushbutton manual mode conveyor M1 backwards), motor Q2 (conveyor motor M1 backwards fixed speed) is moved backward in manual mode. When the two pushbuttons are pressed simultaneously, no motion should take place.

For safety reasons, only the preset speed may be used here. Output Q3 (conveyor motor M1 variable speed) must therefore be deactivated.

1.6.3 Initial state

At station start or after release of EMERGENCY STOP, the station must be moved in manual mode to a defined operating state (initial state). In the initial state, the conveyor is empty and stopped and the cylinder is retracted.

1.7 Automatic mode

In automatic mode, the station executes the process (see also Brief description).

1.7.1 Starting and stopping

If the station is in the initial state, automatic mode starts when pushbutton S1 (automatic start) is pressed. When pushbutton S2 (automatic stop) is pressed, automatic mode is ended again as soon as the initial state has been reached.

If EMERGENCY STOP has been tripped or the operating mode changed, automatic mode is ended immediately (without return to the initial state).

The current state is indicated by indicator light P6 (automatic mode started).

1.7.2 Conveyor control

If light sensor B4 (chute occupied) detects a component, the conveyor motor starts. The component slides onto the transport conveyor and is further conveyed.

If inductive sensor B5 detected a metal component, this is transported up to light sensor B6 (part in front of cylinder M4). The conveyor is then switched off. As soon as B3 (sensor motor M1 active) no longer supplies a signal, the Cylinder control (see below) is activated and moves the component into the metal magazine. As soon as the cylinder is retracted again, the sorting station is back in the initial state.

If a metal component was not detected by sensor B5, this is recognized when light sensor B6 (part in front of cylinder M4) is reached. The plastic component is then transported to the end of the conveyor. It is detected there by light sensor B7 and conveyed after a delay time into the plastic magazine at the end of the conveyor.

1.7.3 Cylinder control

If a metal component reaches light sensor B6 (part in front of cylinder M4) and the conveyor has stopped, cylinder M4 moves to the front end position B2 (cylinder M4 extended), thereby pushing the metal component from the conveyor into the metal magazine. Cylinder M4 then moves to the rear end position B1 (cylinder M4 retracted).

1.7.4 Speed control (conveyor speed)

In automatic mode, the motor can be moved at a fixed or variable speed.

Fixed speed requires signal "1" at Q1 "Conveyor motor M1 forwards fixed speed" or Q2 "Conveyor motor M1 backwards fixed speed". For variable speed, Q3 "Conveyor motor M1 variable speed" must be activated and a "manipulated value for motor speed" (analog value +/-10 V corresponds to +/- 50 rpm or 10 m/s) must be specified at U1. Signal "1" must not be present at Q1 "Conveyor motor M1 forwards fixed speed" or Q2 "Conveyor motor M1 backwards fixed speed". Otherwise, U1 has no effect on the speed of the conveyor.

1.7.5 Speed control

A speed control can be integrated for control of the conveyor speed. This uses the speed sensor for evaluating the current speed. A speed of 5 rpm corresponds to a conveyor belt speed of 1 m/s.

1.8 Indicator lights

As soon as relay K0 (main switch "ON") becomes energized, indicator light P1 (main switch on) lights up.

If switch S0 (mode selector manual/automatic) is set to Manual, the indicator light P2 (manual mode) lights up. If switch S0 is set to Automatic, the indicator light P3 (automatic mode) lights up.

If the EMERGENCY STOP function has tripped, P4 (EMERGENCY STOP activated) lights up.

If automatic mode has been selected and the station is in the initial state, P5 (automatic mode started) flashes to signal that automatic mode can be started. As soon as automatic mode has been started, P5 lights up.

Indicator light P6 (cylinder M4 retracted) lights up as soon as end position sensor B1 (sensor cylinder M4 retracted) has been reached. Indicator light P7 (cylinder M4 extended) lights up as soon as cylinder M4 has reached the front end position sensor B2 (sensor cylinder M4 extended). Indicator lights P6 and P7 are not lit if the cylinder is located in neither of the two end positions.

2 Reference list

By default, the S7-1200 has only 14 digital Inputs, 10 digital outputs, 2 analog inputs und 1 analog output. Therefore, the signals shown in the list with blue text are not available for it.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	A1	Return signal emergency stop OK	NC
I 0.1	BOOL	K0	Main switch "ON"	NO
I 0.2	BOOL	S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	S1	Pushbutton automatic start	NO
I 0.4	BOOL	S2	Pushbutton automatic stop	NC
I 0.5	BOOL	B1	Sensor cylinder M4 retracted	NO
I 0.6	BOOL	B2	Sensor cylinder M4 extended	NC
I 0.7	BOOL	B3	Sensor motor M1 active (pulse signal also suitable for positioning)	NO
I 1.0	BOOL	B4	Sensor at chute occupied	NO
I 1.1	BOOL	B5	Sensor metal part	NO
I 1.2	BOOL	B6	Sensor part in front of cylinder M4	NO
I 1.3	BOOL	B7	Sensor part at end of conveyor	NO
I 1.4	BOOL	S3	Pushbutton manual mode conveyor M1 forwards	NO
I 1.5	BOOL	S4	Pushbutton manual mode conveyor M1 backwards	NO
I 1.6	BOOL	S5	Pushbutton manual mode cylinder M4 retract	NO
I 1.7	BOOL	S6	Pushbutton manual mode cylinder M4 extend	NO

DO	Type	Identifier	Function	
Q 0.0	BOOL	Q1	Conveyor motor M1 forwards fixed speed	
Q 0.1	BOOL	Q2	Conveyor motor M1 backwards fixed speed	
Q 0.2	BOOL	Q3	Conveyor motor M1 variable speed	
Q 0.3	BOOL	M2	Cylinder M4 retract	
Q 0.4	BOOL	M3	Cylinder M4 extend	
Q 0.5	BOOL	P1	Display "main switch on"	
Q 0.6	BOOL	P2	Display "MANUAL" mode	
Q 0.7	BOOL	P3	Display "AUTOMATIC" mode	
Q 1.0	BOOL	P4	Display "emergency stop activated"	
Q 1.1	BOOL	P5	Display "automatic mode started"	
Q 1.2	BOOL	P6	Display "cylinder M4 retracted"	
Q 1.3	BOOL	P7	Display "cylinder M4 extended"	
AI	**Type**	**Identifier**	**Function**	
IW 64	INT	B8	Sensor actual value speed of motor +/- 10V	
IW 66	INT	B9	Setpoint specification via potentiometer +/- 10V	

AO	Type	Identifier	Function	
QW 64	INT	U1	Manipulated value speed of motor in 2 directions +/- 10V	

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

3 Description of components of the station

3.1 Manual operation

3.1.1 Pushbuttons

The utilized pushbuttons can supply either a "0" or "1" signal. Depending on whether you have planned them as normally-closed or normally-open contacts (see Reference list), they supply a "1" or "0" signal when not actuated. The signal changes to "0" or "1" only while the pushbutton is being pressed.

3.1.2 Switches

The utilized switches can also supply either a "0" or "1" signal. Depending on whether you have planned them as normally-closed or normally-open contacts (see Reference list), they supply a "1" or "0" signal when not actuated. The signal changes to "0" or "1" when the switch is actuated. This signal is present as long as the switch is not actuated again.

3.1.3 Feedback from EMERGENCY STOP pushbutton

EMERGENCY STOP pushbuttons are pushbuttons with an additional mechanical lock and are connected to a safety relay. They thus behave like a switch. The EMERGENCY STOP feedback from the safety relay is planned as a normally closed contact for safety reasons. If a wire break occurs, therefore, this feedback is no longer present and the station responds as if an EMERGENCY STOP has tripped.

3.2 Sensors

3.2.1 Position switches

A main switch is actuated to switch on the station. This energizes a relay and supplies the power to the station. A position switch provides feedback on the operation of the relay.

3.2.2 Limit switches

The limit switches supply a signal when the cylinder is either fully retracted or extended. The limit switches are implemented as normally-closed or normally-open contacts.

3.2.3 Light barriers / optical sensors

The light barriers supply a "1" signal as soon as an object is in the sensing range.

3.2.4 Metal detection / Inductive sensor

The inductive sensor supplies a "1" signal as soon as a metallic object enters its sensing range. In the case of non-metallic objects, the signal remains at "0".

3.2.5 Motor speed

The motor speed is recorded by an incremental encoder at the conveyor motor and provided as an analog value via a transducer. The speed falls within the range from -50 rpm to 50 rpm. That corresponds to a conveyor belt speed of -10 m/s to +10 m/s.

In addition, pulses are received at "Sensor conveyor motor M1 active" that can also be used for positioning. The resolution is 20 pulses per total conveyor belt length (10 m).

3.3 Actuators

3.3.1 Conveyor motor

The conveyor motor drives the conveyor belt. It has multiple signal combinations so that the conveyor belt can be moved at fixed or variable speed in both directions.

Fixed speed requires signal "1" at Q1 "Conveyor motor M1 forwards fixed speed" or Q2 "Conveyor motor M1 backwards fixed speed". For variable speed, Q3 "Conveyor motor M1 variable speed" must be activated and a "manipulated value for motor speed" (analog value +/-10 V corresponds to +/- 50 rpm or 10 m/s) must be specified at U1. Signal "1" must not be present at Q1 "Conveyor motor M1 forwards fixed speed" or Q2 "Conveyor motor M1 backwards fixed speed". Otherwise, U1 has no effect. Simultaneous activation of signals Q1 and Q2 causes the conveyor to stop and must be prevented by the control program.

3.3.2 Cylinders

Cylinder M4 is controlled using two separate signals. Activation of one signal (M3) causes the cylinder to extend and activation of the other signal (M2) causes the cylinder to retract. The signals must not be activated simultaneously, otherwise an undefined state occurs and the cylinder pauses at its position. This must be prevented by the control program.

3.3.3 Displays

All indicator lights are located on the control panel. If signal "1" is present, these indicator lights illuminate.

4 Brief description of the simulation

The simulation of the sorting station consists of 9 diagrams. The 01_Operating screen diagram is important for operation (see Figure 3), which contains the control panel and a representation of the station.

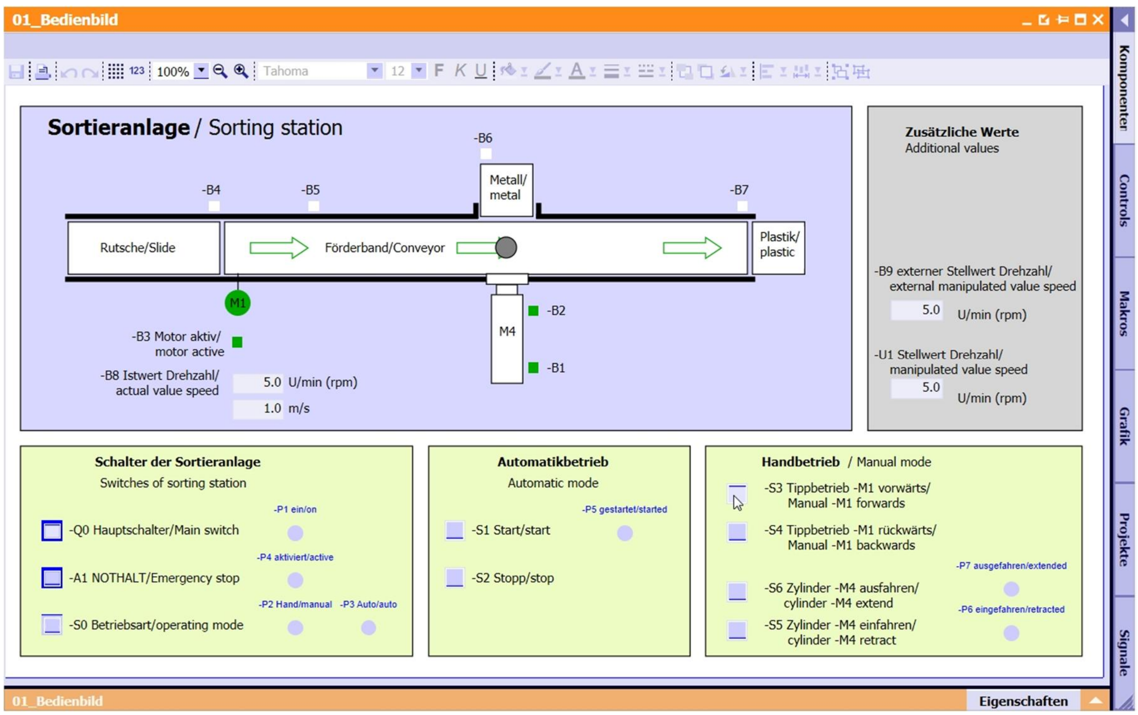


Figure 3: The operating screen

Figure 4 shows the 02_SimControl diagram. It allows important simulation settings to be made. The first settings affect the creation of the components. Here, you can select between automatic and manual creation of components. With automatic creation of components, a new component is always created and sent to the station when the previous component has been sorted. A single component is created with manual creation of components. The next settings allow you to specify whether a metal component or plastic component will be created. The following selections are available: Produce only metal components, Produce only plastic components and Randomly produce metal or plastic components Only one of the three options should be selected.

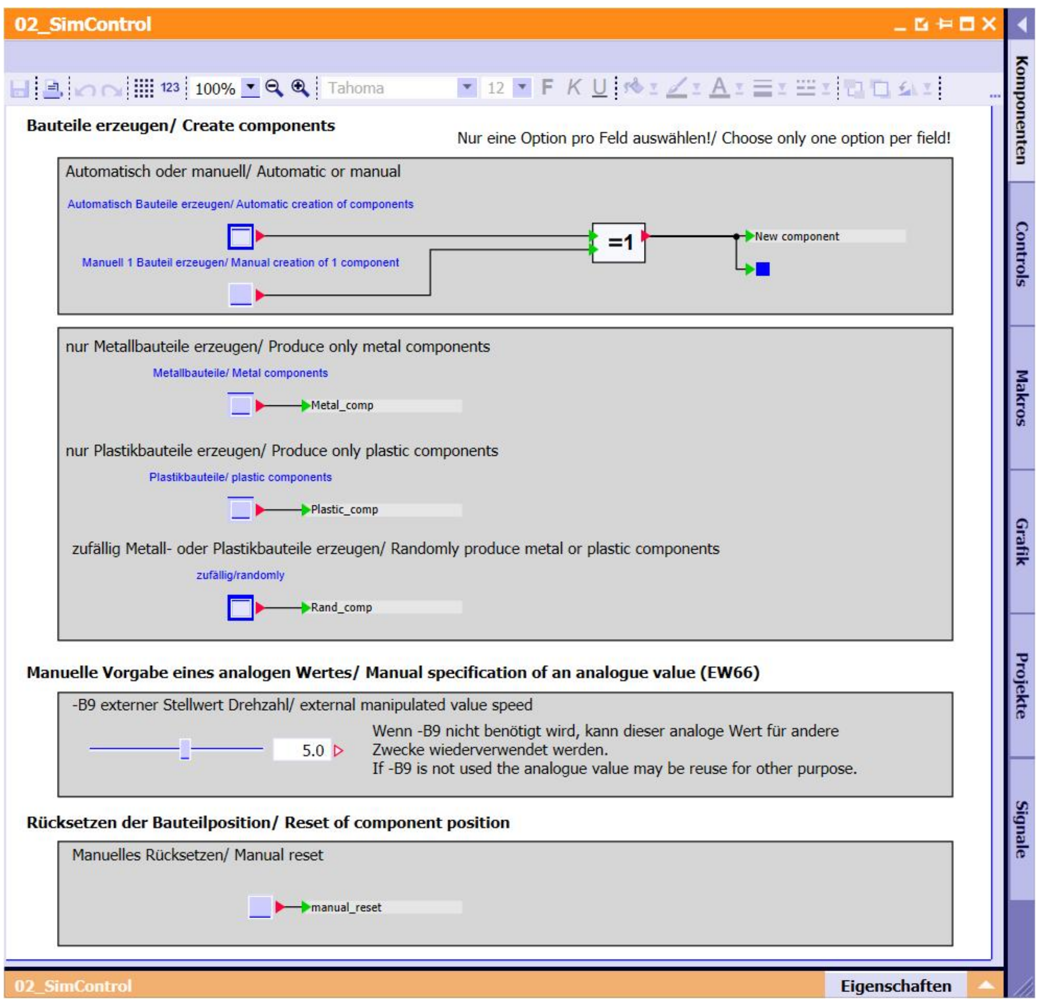


Figure 4: Simulation control

In the "Manual specification of an analog value" area, a value between -50 and +50 can be set for input word IW 66 (see Reference list). This corresponds to an input voltage of -/+10 V. This value is then converted to a digital value between -27648 and +27648 and is thus available as an analog input value.

The last setting concerns the manual resetting of the current component. This resets the position of the component and a new component can be created.

5 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

⊟ Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- ↗ TIA Portal Videos
- ↗ TIA Portal Tutorial Center
- › Getting Started
- ↗ Programming Guideline
- ↗ Easy Entry in SIMATIC S7-1200
- › Download Trial Software/Firmware
- ↗ Technical Documentation SIMATIC Controller
- ↗ Industry Online Support App
- ↗ TIA Portal, SIMATIC S7-1200/1500 Overview
- ↗ TIA Portal Website
- ↗ SIMATIC S7-1200 Website
- ↗ SIMATIC S7-1500 Website

Notes

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is available at: siemens.com/sce/tp

5

Continued training

For regional Siemens SCE continued training, get in touch with your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

Table of contents

5

5

Basics of FC Programming

1 Goal

In this chapter, you will get to know the basic elements of a control program – the ***organization blocks (OBs), functions (FCs)***, **function blocks (FBs)** and **data blocks (DBs)**. In addition, you will be introduced to ***library-compatible*** function und function block programming. You will get to know the ***Function Block Diagram (FBD)*** programming language and use it to program a function (FC1) and an organization block (OB1).

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

This chapter builds on the hardware configuration of SIMATIC S7 CPU1214C. However, other hardware configurations that have digital input and output cards can be used. For this chapter, you can use the following project, for example:

SCE_EN_011_101_Hardware_Configuration_CPU1214C.zap14

3 Required hardware and software

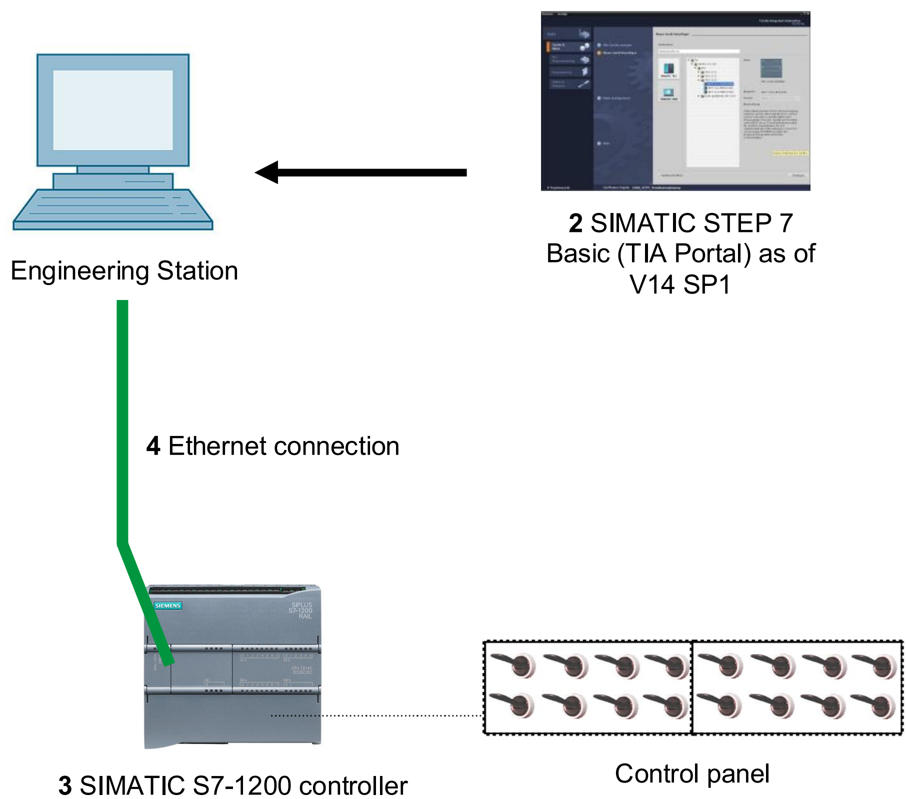
1 Engineering station: requirements include hardware and operating system (for additional information, see Readme on the TIA Portal Installation DVDs)

2 SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1

Note: The digital inputs should be fed out to a control panel.

4 Ethernet connection between engineering station and controller



4 Theory

4.1 Operating system and application program

Every controller (CPU) contains an ***operating system***, which organizes all functions and sequences of the CPU that are not associated with a specific control task. The tasks of the operating system include the following:

– Performing a warm restart

– Updating the process image of the inputs and output

– Cyclically calling the user program

– Detecting interrupts and calling interrupt OBs

– Detecting and handling errors

– Managing memory areas

The operating system is an integral component of the CPU and comes pre-installed.

The ***user program*** contains all functions that are necessary for executing your specific automation task. The tasks of the user program include the following:

– Checking the basic requirements for a warm restart using startup OBs

– Processing of process data, i.e. activation of output signals as a function of the input signal states

– Reaction to interrupts and interrupt inputs

– Error handling during normal program execution

4.2 Organization blocks

Organization blocks (OBs) form the interface between the operating system of the controller (CPU) and the application program. They are called from the operating system and control the following operations:

- Cyclic program processing (e.g. OB1)
- Startup characteristics of the controller
- Interrupt-driven program processing
- Error handling

A project must have, at a minimum, ***an organization block for cyclic program processing***. An OB is called by a ***start event*** as shown in Figure 1. In addition, the individual OBs have defined priorities so that, for example, an OB82 for error handling can interrupt the cyclic OB1.

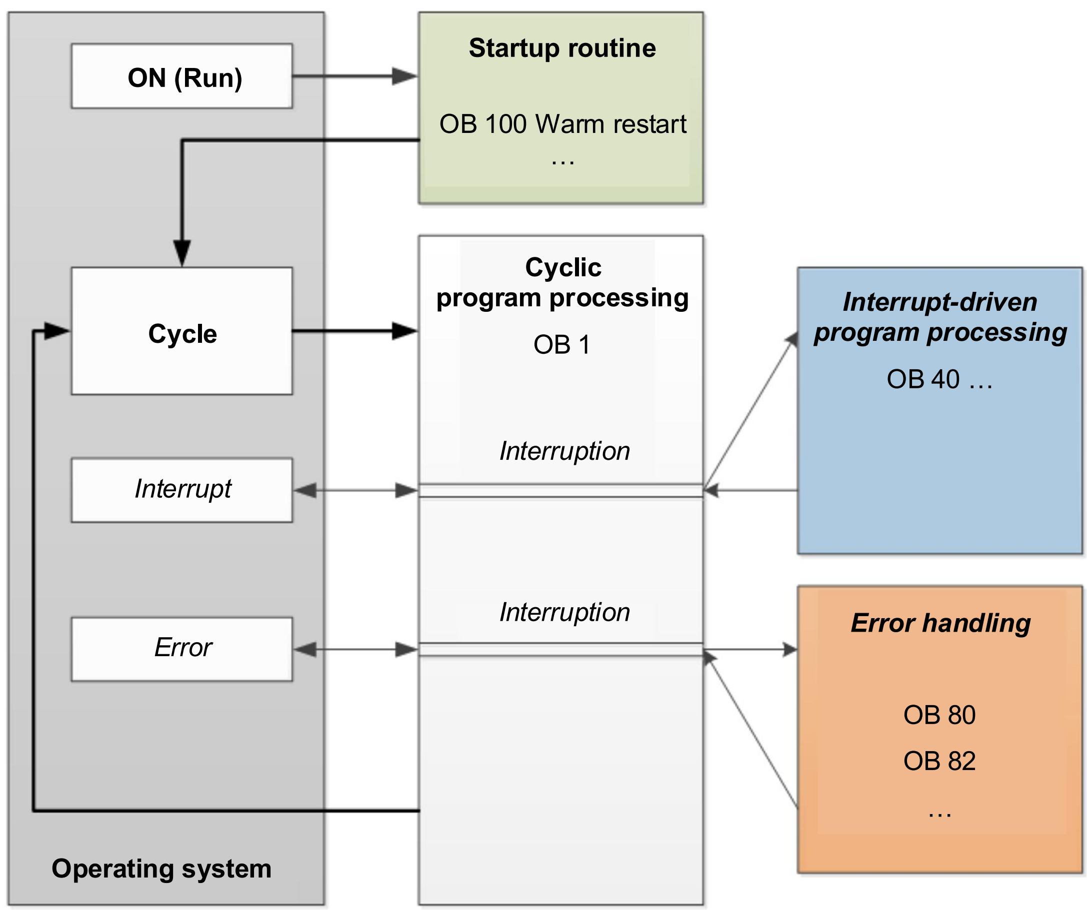


Figure 1: Start events in the operating system and OB call

When a start event occurs, the following reactions are possible:

– If an OB has been assigned to the event, this event triggers the execution of the assigned OB. If the priority of the assigned OB is greater than the priority of the OB that is currently being executed, it is executed immediately (interrupt). If not, the assigned OB waits until the higher-priority OB has been completely executed

– If you have not assigned an OB to the event, the default system reaction is performed.

Table 1 shows examples for various start events for a SIMATIC S7-1200. Also shown are the possible OB number(s) and the default system reactions that occur when the respective organization block (OB) is not present in the controller.

Start event	**Possible OB numbers**	**Default system reaction**
Startup	100, ≥ 123	Ignore
Cyclic program	1, ≥ 123	Ignore
Time-of-day interrupt	10 to 11	-
Update interrupt	56	Ignore
Scan cycle monitoring time exceeded once	80	Ignore
Scan cycle monitoring time exceeded twice	80	STOP
Diagnostic interrupt	82	Ignore

Table 1: OB numbers for various start events

4.3 Process image and cyclic program processing

When the cyclic user program addresses the inputs (I) and outputs (O), it does not query the signal states directly from the input/output modules. Instead, it accesses a memory area of the CPU. This memory area contains an image of the signal states and is called the **process image**.

The cyclic program processing sequence is as follows:

1. At the start of the cyclic program, a query is sent to determine whether or not the individual inputs are energized. This status of the inputs is stored in the **process image of the inputs (PII)**. In doing so, the information 1 or "High" is stored for energized inputs and the information 0 or "Low" for de-energized inputs.
2. The CPU then executes the program stored in the cyclic organization block. For the required input information, the CPU accesses the previously read **process image of the inputs (PII)** and the results of logic operation (RLOs) are written to a so-called **process image of the outputs (PIQ)**.
3. At the end of the cycle, the **process image of the outputs** (**PIQ**) is transferred as the signal state to the output modules and these are energized or de-energized. The sequence then continues again with Item 1.

5

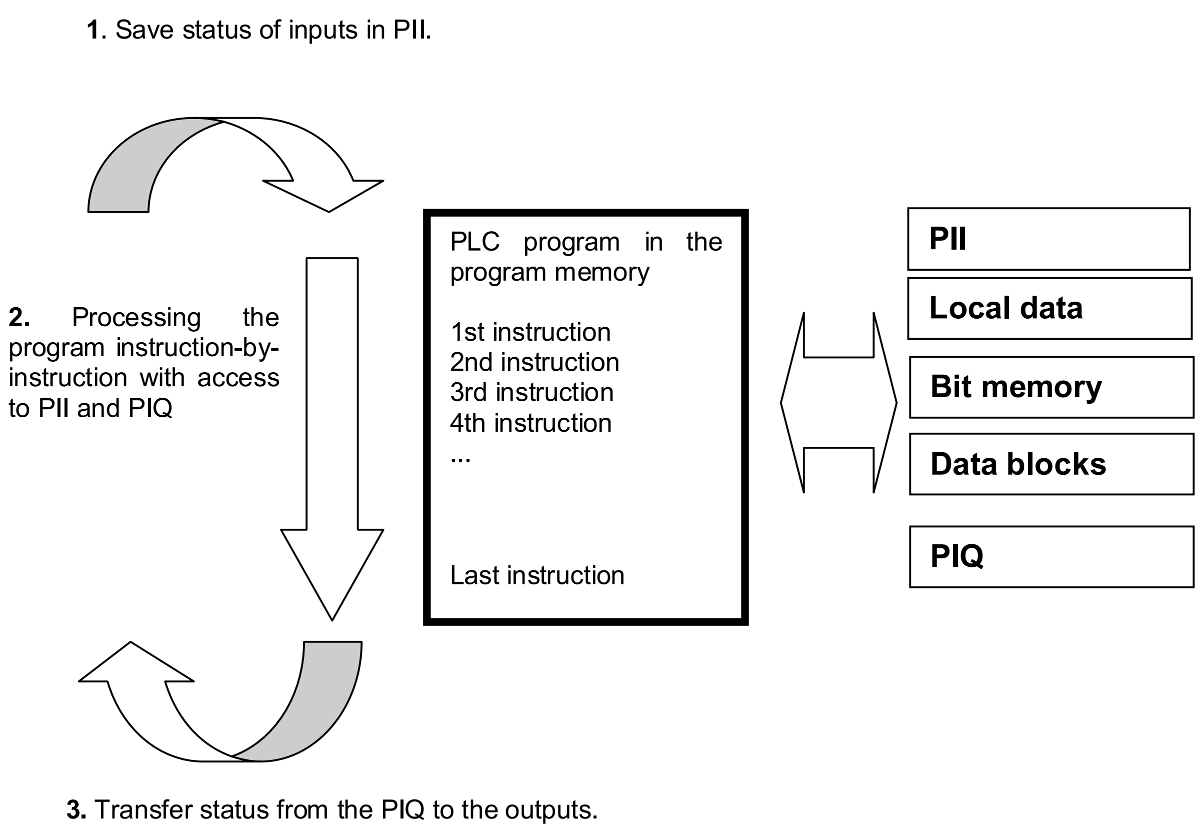


Figure 2: Cyclic program processing

Note: *The time the CPU needs for this sequence is called cycle time. This depends, in turn, on the number and type of instructions and the processor performance of the controller.*

4.4 Functions

Functions (FCs) are logic blocks without memory. They ***have no data memory*** in which values of block parameters can be stored. Therefore, all interface parameters must be connected when a function is called. To store data permanently, global data blocks must be created beforehand.

A function contains a program that is executed whenever the function is called from another logic block.

Functions can be used, for example, for the following purposes:

– Math functions – that return a result dependent on input values.

– Technological functions – such as individual controls with binary logic operations.

A function can also be called several times at different points within a program.

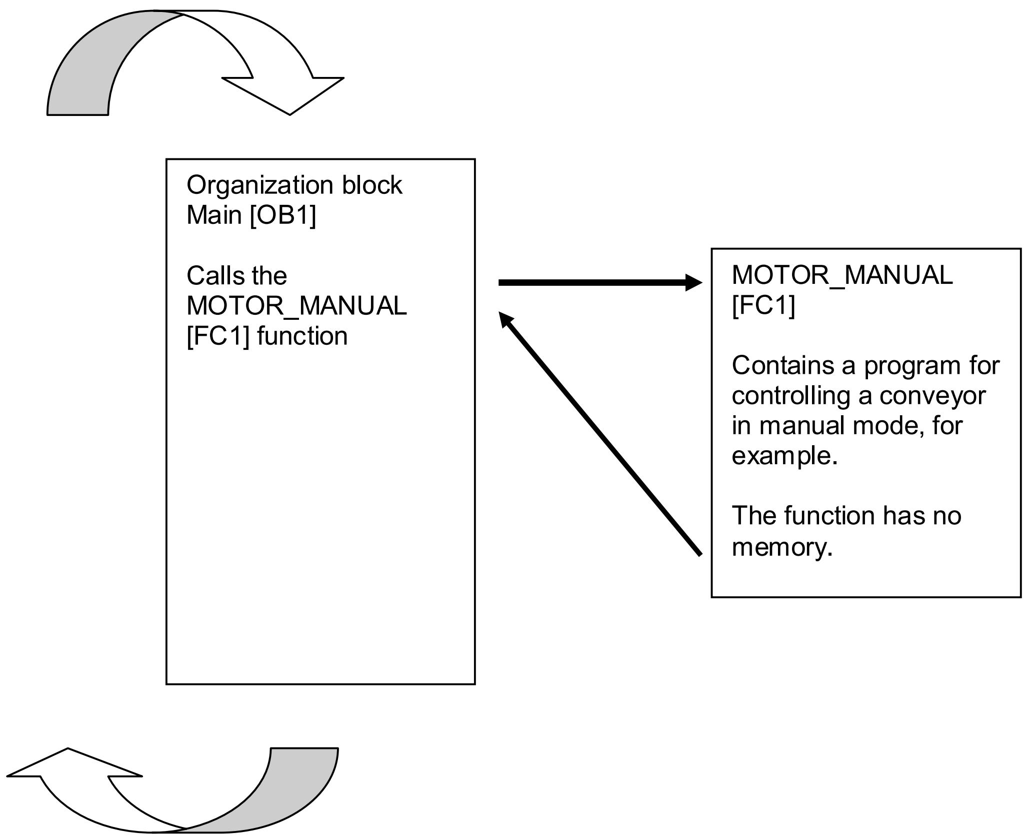


Figure 3: Function with call from organization block Main [OB1]

5

4.5 Function blocks and instance data blocks

Function blocks are logic blocks that store their input, output and in-out tags as well as static tags permanently in instance data blocks, so that they ***are available after the block has been executed***. For this reason, they are also referred to as blocks with "memory".

Function blocks can also operate with temporary tags. Temporary tags are not stored in the instance DB, however. Instead, they are only available for one cycle.

Function blocks are used for tasks that cannot be implemented with functions:

– Whenever timers and counters are required in the blocks, or

– When information must be stored in the program, such as preselection of the operating mode with a button.

Function blocks are always executed when called from another logic block. A function block can also be called several times at different points within a program. This facilitates the programming of frequently recurring complex functions.

A call of a function block is referred to as an instance. Each instance of a function block is assigned a memory area that contains the data that the function block uses. This memory is made available by data blocks created automatically by the software.

It is also possible to provide memory for multiple instances in one data block in the form of a **multi-instance**. The maximum size of instance data blocks varies depending on the CPU. The tags declared in the function block determine the structure of the instance data block.

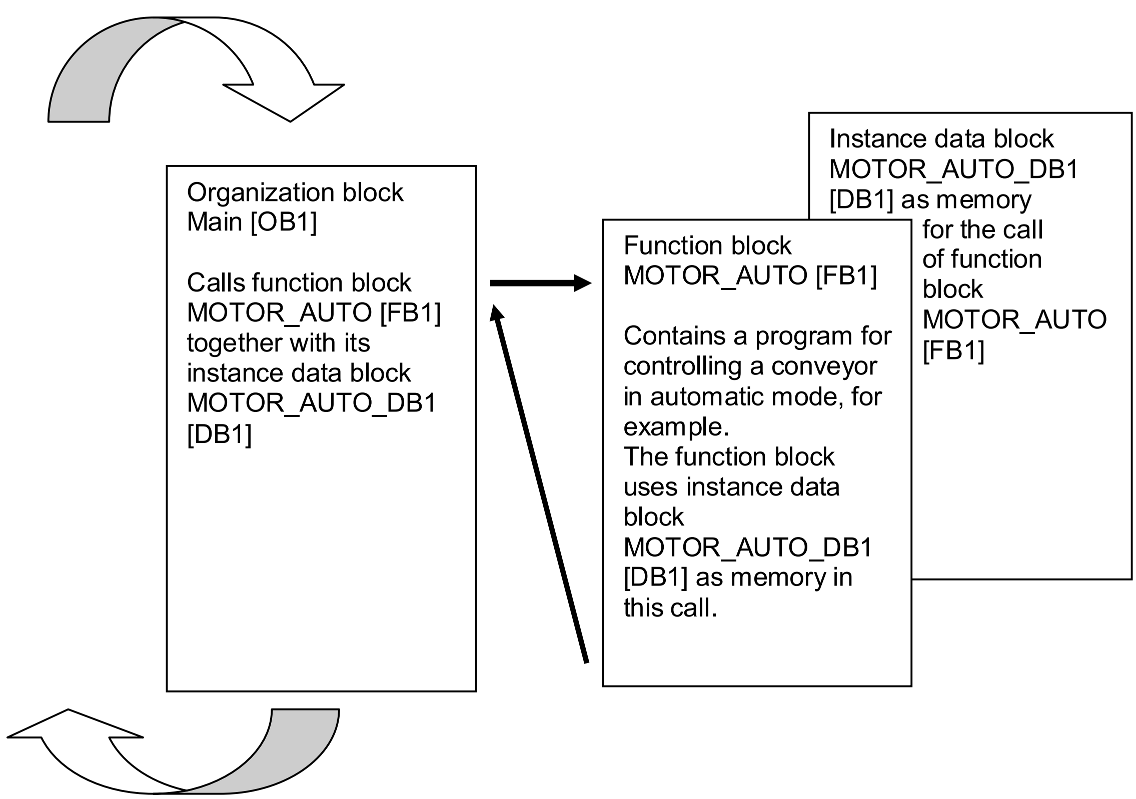


Figure 4: Function block and instance with call from organization block Main [OB1]

4.6 Global data blocks

In contrast to logic blocks, data blocks contain no instructions. Rather, they serve as memory for user data.

Data blocks thus contain variable data that is used by the user program. You can define the structure of global data blocks as required.

Global data blocks store data that can be used ***by all other blocks*** (see Figure 5). Only the associated function block should access instance data blocks. The maximum size of data blocks varies depending on the CPU.

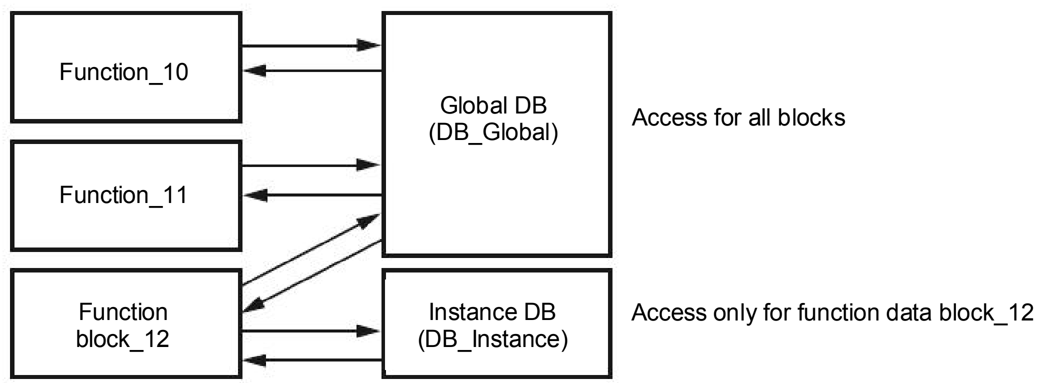


Figure 5: Difference between global DB and instance DB.

Application examples for **global data blocks** are:

– Saving of information about a storage system. "Which product is located where?"

– Saving of recipes for particular products.

4.7 Library-compatible logic blocks

A user program can be created with linear or structured programming. ***Linear programming*** writes the entire user program in the cycle OB, but is only suitable for very simple programs for which other less expensive control systems, such as LOGO!, can now be used.

For more complex programs, ***structured programming*** is always recommended. Here, the overall automation task can be broken down into small sub-tasks in order to implement a solution for them in functions and function blocks.

In this case, library-compatible logic blocks should be created preferentially. This means that the input and output parameters of a function or function block are defined generally and only supplied with the current global tags (inputs/outputs) when the block is used.

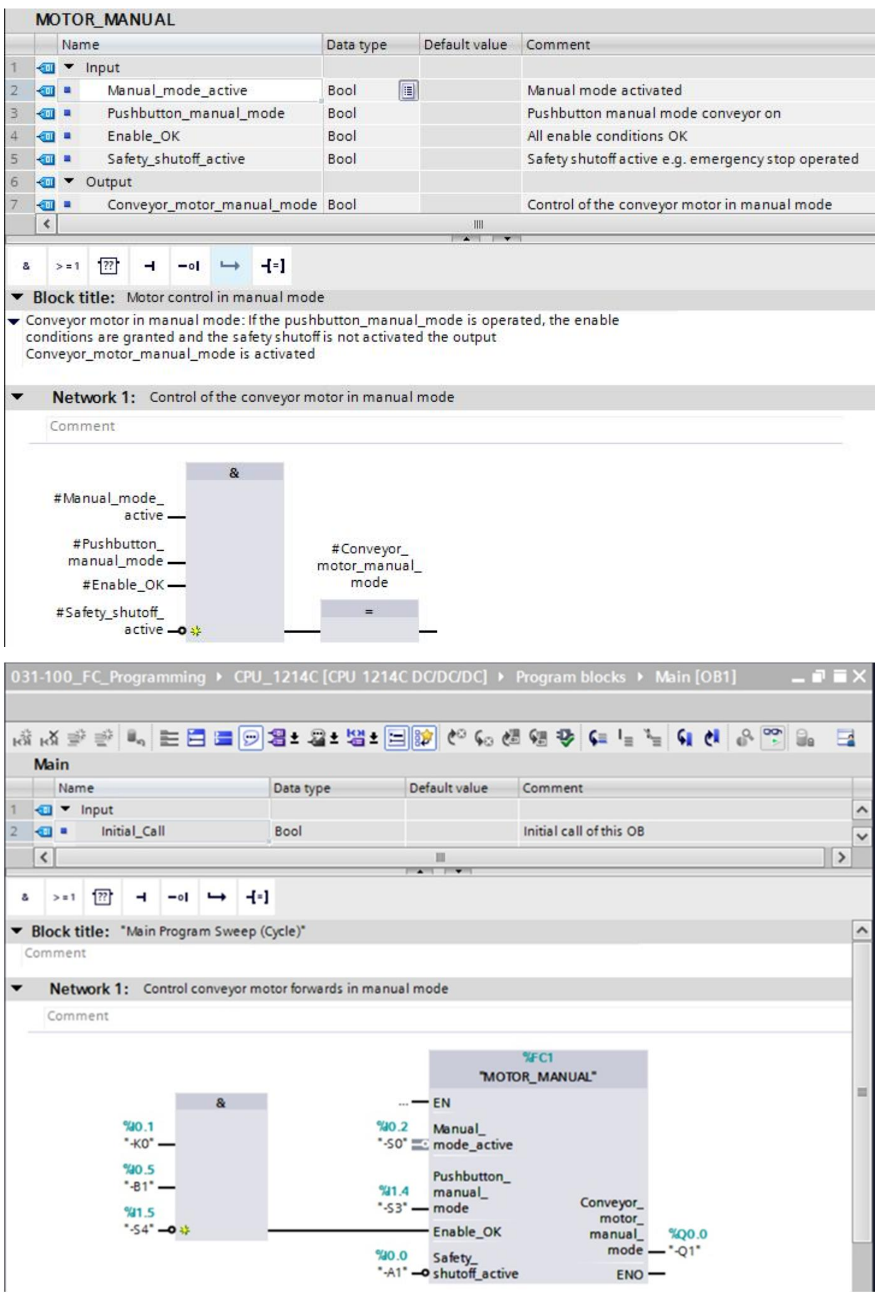


Figure 6: Library-compatible function with call in OB1

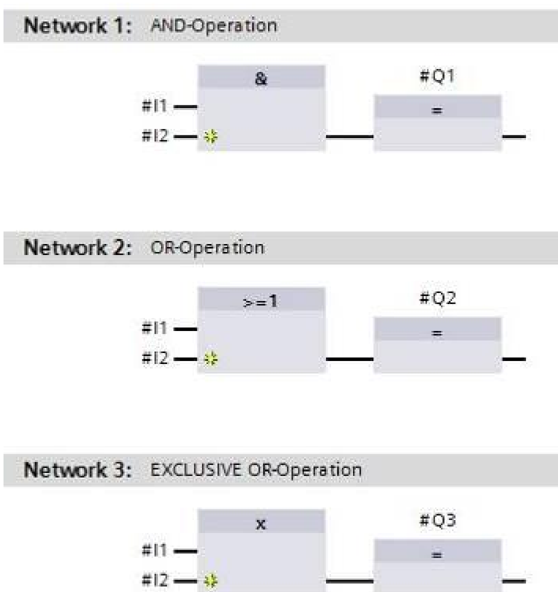
4.8 Programming languages

For SIMATIC S7-1200, the available programming languages for programming functions and function blocks are Function Block Diagram (FBD), Ladder Logic (LAD) and Structured Control Language (SCL).

The ***Function Block Diagram (FBD)*** programming language will be presented in the following.

FBD is a graphical programming language. The representation is based on electronic switching systems. The program is mapped in networks. A network contains one or more logic operation paths. Binary and analog signals are linked together by boxes. The graphical logic symbols known from Boolean algebra are used to represent the binary logic.

You can use binary functions to query binary operands and to logically combine their signal states. The following instructions are examples of binary functions: "AND operation", "OR operation" and "EXCLUSIVE OR operation". These are shown in Figure 7.



#e1	#e2	#a1
0	0	0
0	1	0
1	0	0
1	1	1

#e1	#e2	#a2
0	0	0
0	1	1
1	0	1
1	1	1

#e1	#e2	#a3
0	0	0
0	1	1
1	0	1
1	1	0

Figure 7: Binary functions in FBD and associated logic table

You can thus use simple instructions, for example, to control binary outputs, evaluate edges and execute jump functions in the program.

Program elements such as IEC timers and IEC counters provide complex instructions.

The empty box serves as a placeholder in which you can select the required instruction.

Enable input EN (enable) / Enable output ENO (enable output) mechanism:

– An instruction without EN/ENO mechanism is executed independent of the signal state at the box inputs.

– Instructions with EN/ENO mechanism are only executed if enable input "EN input has signal state "1". When the box is processed correctly, enable output "ENO" has signal state "1". If an error occurs during processing, the enable output "ENO" is reset. If enable input EN is not connected, the box is always executed.

5 Task

The following functions of the sorting station process description will be planned, programmed and tested in this chapter:

– Manual mode – Control of conveyor tracking forwards in manual/jog mode

6 Planning

The programming of all functions in OB1 is not recommended for reasons of clarity and reusability. The majority of the program code will therefore be moved into functions (FCs) and function blocks (FBs). The decision on which functions is to be moved to FCs and which is to run in OB 1 is planned below.

6.1 EMERGENCY STOP

The EMERGENCY STOP does not require a separate function. Just like the operating mode, the current state of the EMERGENCY STOP relay can be used directly at the blocks.

6.2 Manual mode – Conveyor motor in manual mode

Manual mode of the conveyor motor is to be encapsulated in a function (FC) "MOTOR_MANUAL". On the one hand, this preserves the clarity of OB1. On the other hand, it enables reuse if another conveyor belt is added to the station. Table 2 lists the planned parameters.

Input	Data type	Comment
Manual_mode_active	BOOL	Manual mode activated
Pushbutton_manual_mode	BOOL	Pushbutton to switch on conveyor in manual mode
Enable_OK	BOOL	All enable conditions OK
Safety_shutoff_active	BOOL	Safety shutoff active e.g. emergency stop operated
Output		
Conveyor_motor_manual_mode	BOOL	Control of the conveyor motor in manual mode

Table 2: Parameters for FC "MOTOR_MANUAL"

Output Conveyor_motor_manual_mode is ON as long as Pushbutton_manual_mode is pressed, manual mode is activated, the enable conditions are met and the safety shutoff is not active.

6.3 Technology diagram

Here, you see the technology diagram for the task.

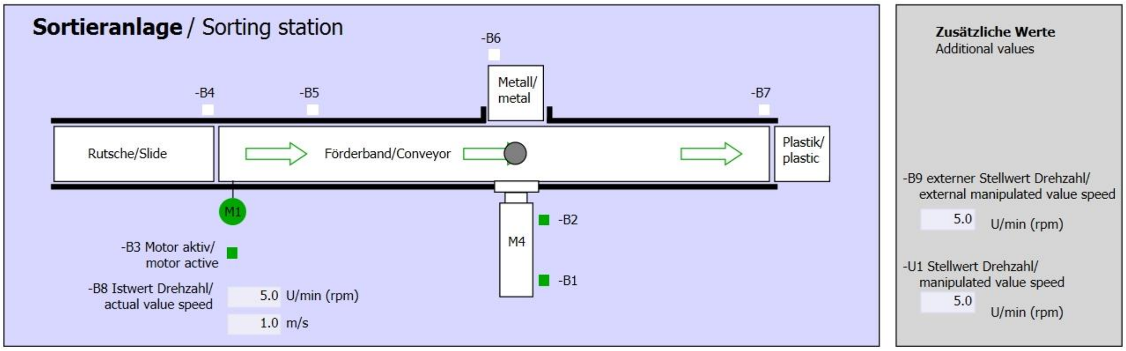


Figure 8: Technology diagram

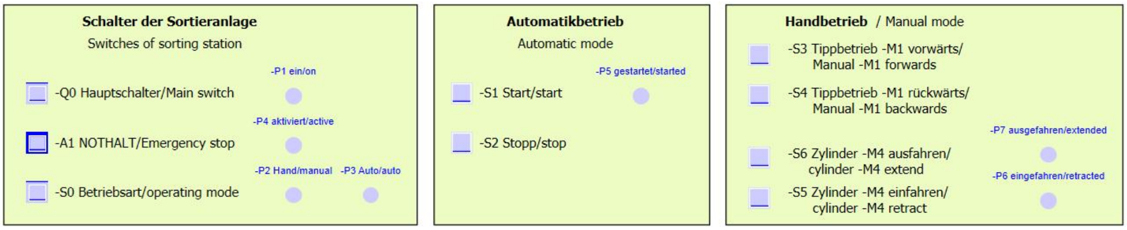


Figure 9: Control panel

5

6.4 Reference list

The following signals are needed as operands for this task.

DI	Type	Identifier	Function	NC/NO
I	BOOL	-A1	Return signal emergency stop ok	NC
I	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I	BOOL	-B1	Sensor cylinder M4 retracted	NO
I 1.4	BOOL	-S3	Pushbutton manual mode conveyor M1 forward	NO
I 1.5	BOOL	-S4	Pushbutton manual mode conveyor M1 reverse	NO

DO	Type	Identifier	Function	
Q	BOOL	-Q1	Conveyor motor M1 forwards fixed speed	

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

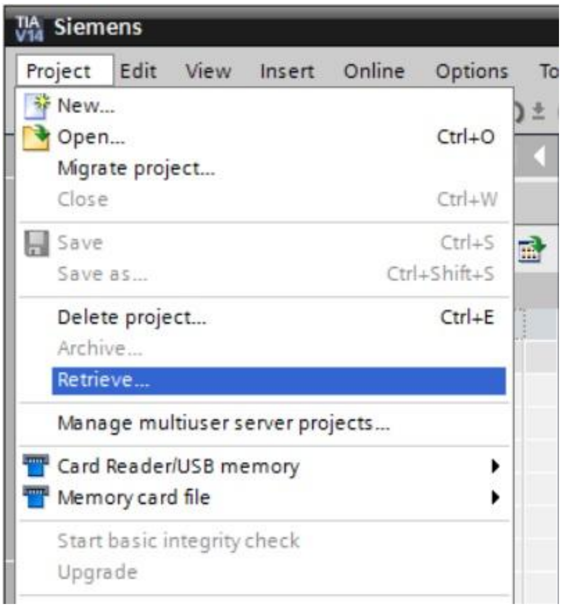
5

7 Structured step-by-step instructions

You can find instructions on how to carry out planning below. If you already have a good understanding of everything, it is sufficient to focus on the numbered steps. Otherwise, simply follow the detailed steps in the instructions.

7.1 Retrieve an existing project

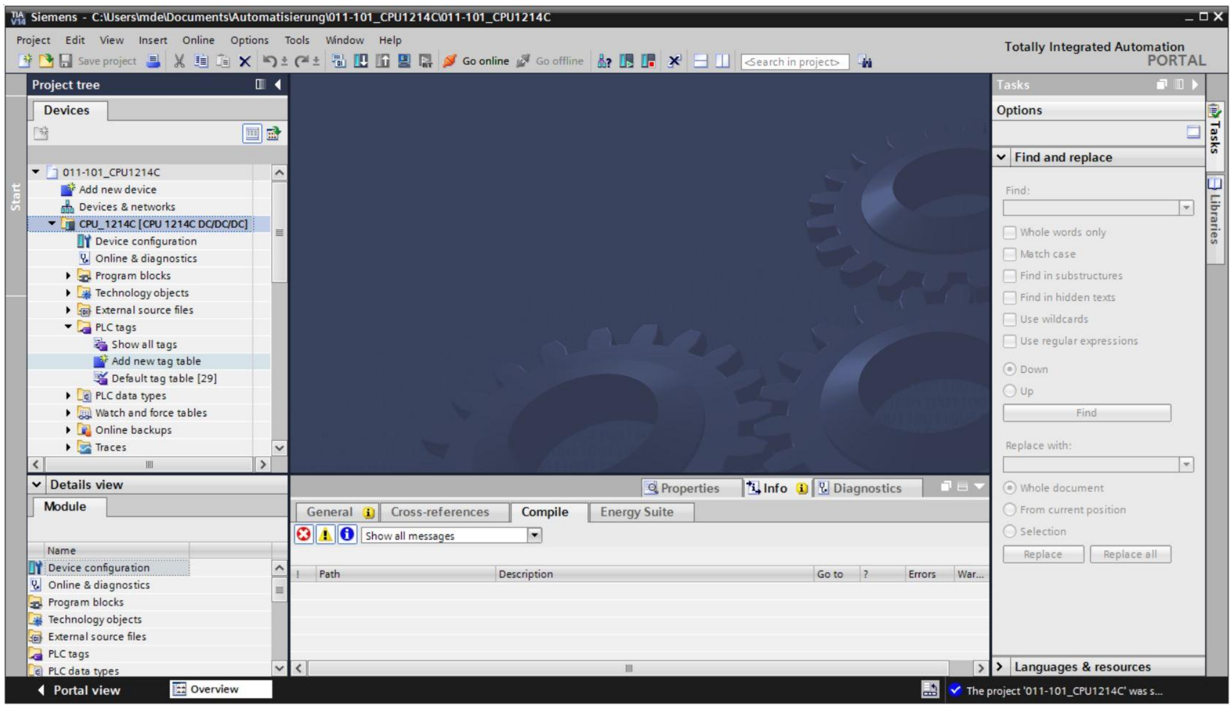
→ Before we can start programming the function (FC) "MOTOR_MANUAL", we need a project with a hardware configuration (e.g. SCE_EN_011_101_Hardware_Configuration_CPU1214C.zap14). To retrieve an existing project that has been archived, you must select the relevant archive with → Project → Retrieve in the project view. Confirm your selection with Open. (→ Project → Retrieve → Select a .zap archive → Open).



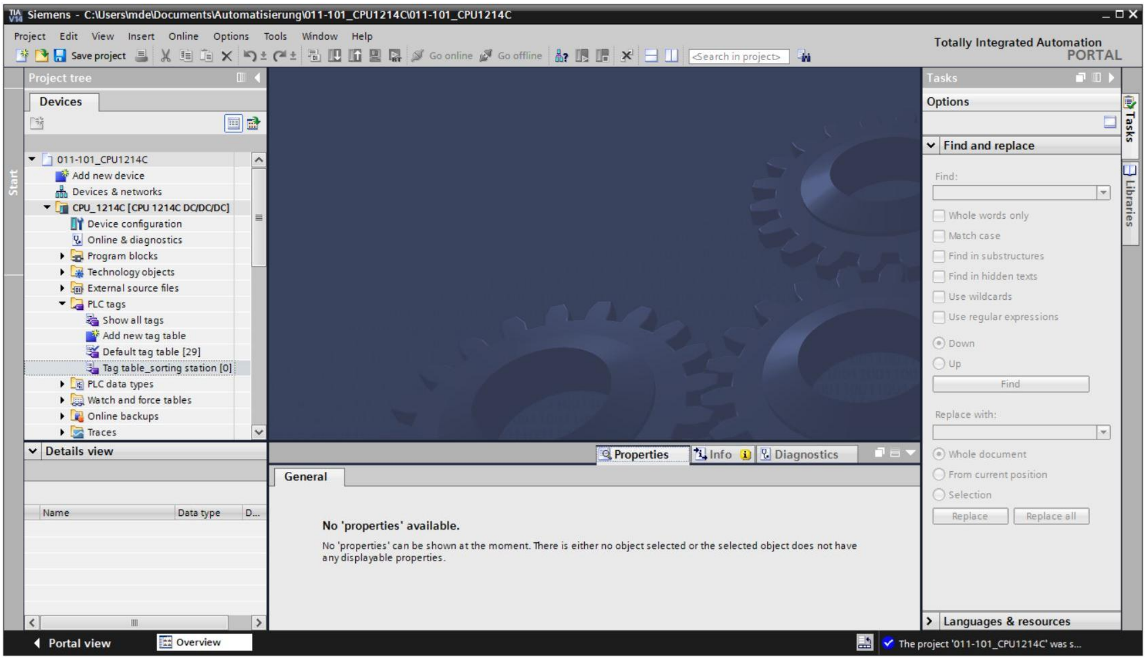
→ The next step is to select the target directory where the retrieved project will be stored. Confirm your selection with "OK". (→ Target directory → OK)

7.2 Create a new tag table

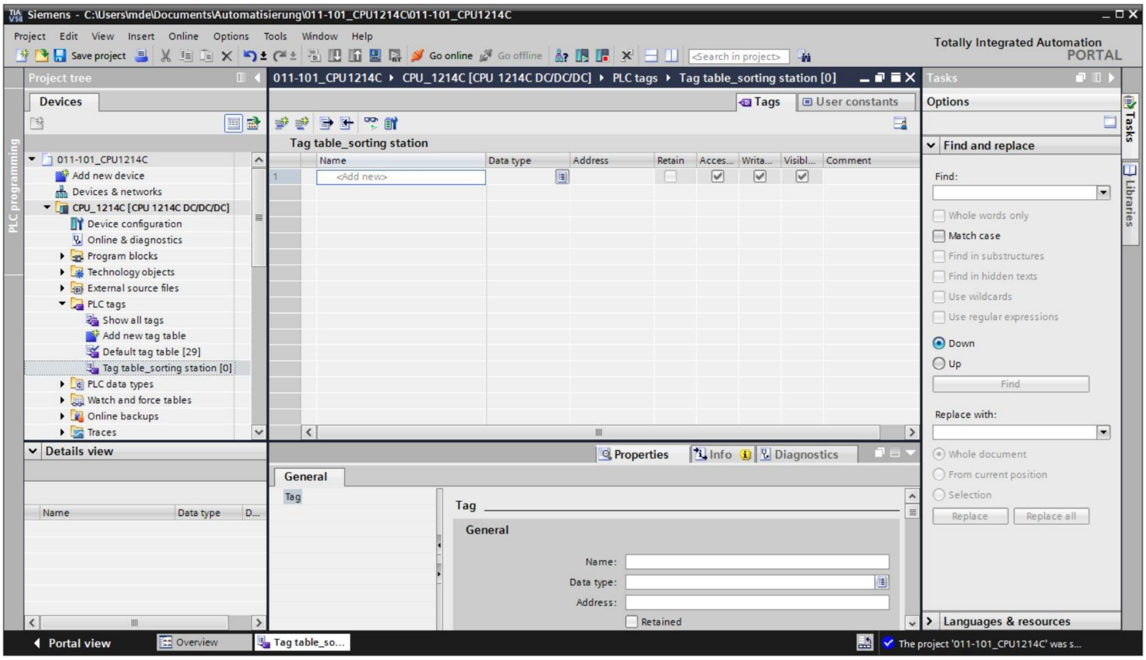
→ In the project view, navigate to the → PLC tags of your controller and create a new tag table by double-clicking → Add new tag table.



→ Rename the tag table you just created as "Tag_table_sorting_station" (→ right-click "Tag_table_1" → "Rename" → Tag_table_sorting_station).



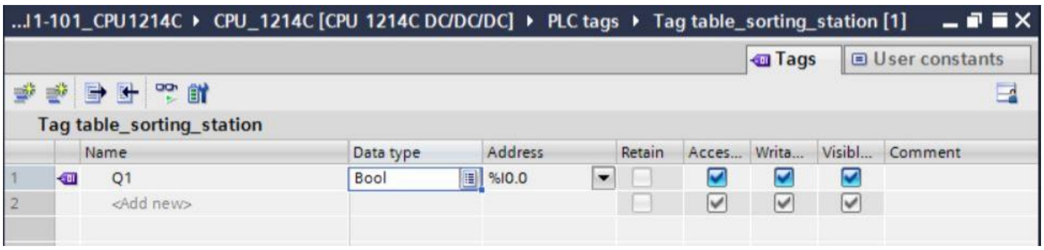
→ Open this tag table with a double-click. (→ Tag_table_sorting_station)



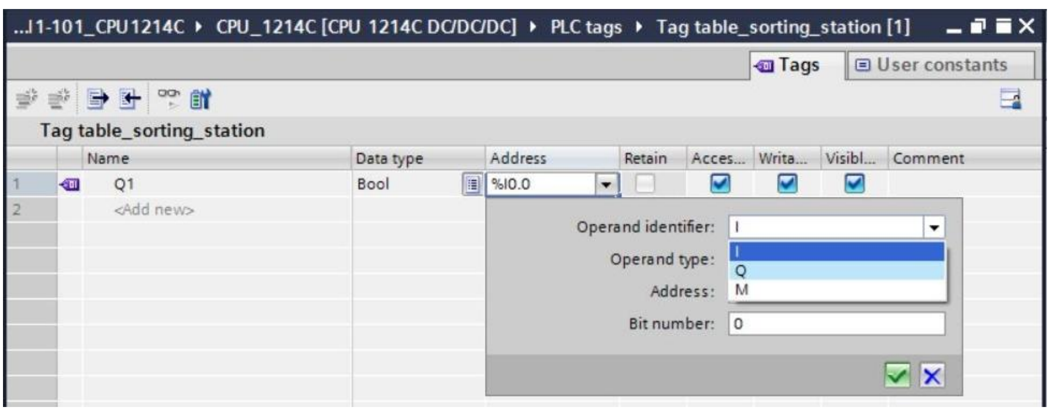
5

7.3 Create new tags within a tag table

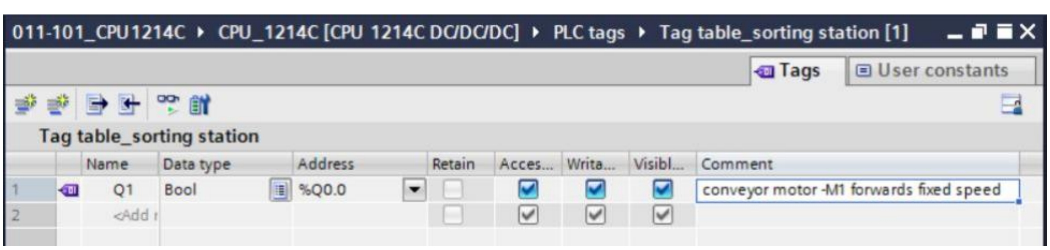
→ Add the name Q1 and confirm the entry with the Enter key. If you have not yet created additional tags, TIA Portal now automatically assigns data type "Bool" and address %I0.0 (I 0.0) (→ <Add> → Q1 → Enter).



→ Change the address to %Q0.0 (Q 0.0) by entering this directly or by clicking the drop-down arrow to open the Addressing menu. Change the operand identifier to Q and confirm with Enter or by clicking the check mark (→ %I0.0 → Operand identifier→ Q → ✓)

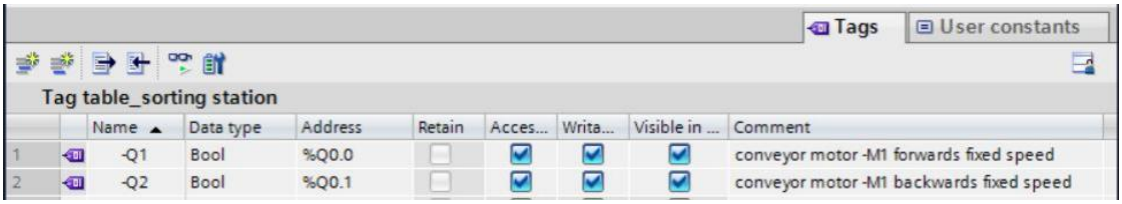


→ Enter the "Conveyor motor M1 forwards fixed speed" comment for the tag.



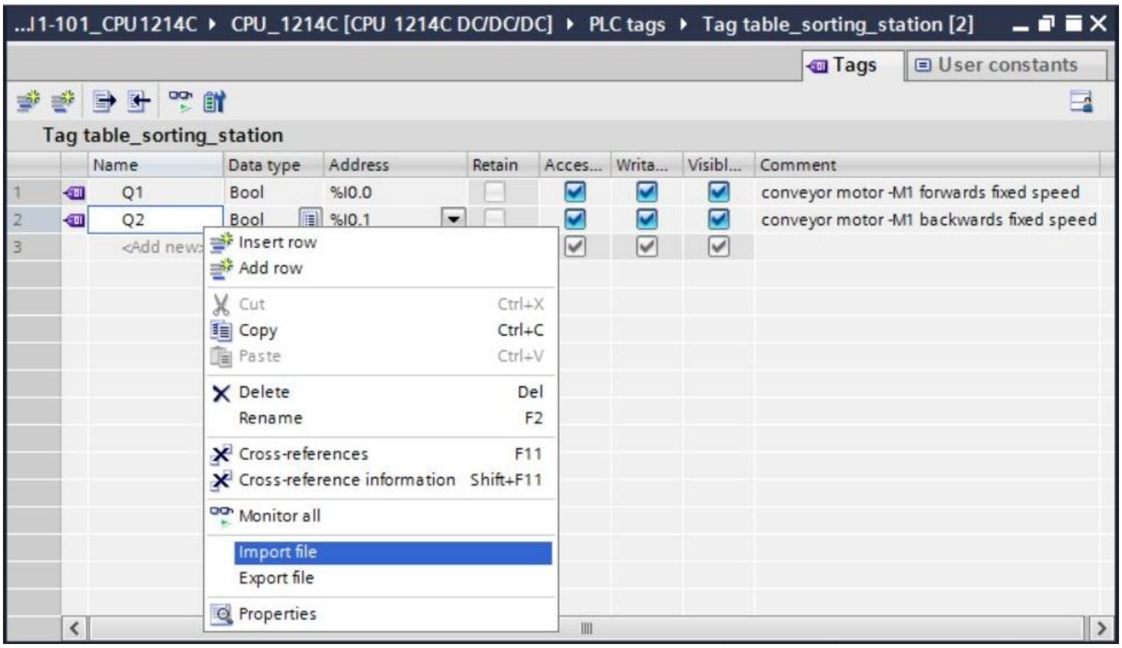
→ Add a new Q2 tag in line 2. TIA Portal has automatically assigned the same data type as the one in line 1 and has incremented the address by 1 to %Q0.1 (Q0.1). Enter the comment "Conveyor motor M1 backwards fixed speed".

(→ <Add> → Q2 → Enter → Comment → Conveyor motor M1 backwards fixed speed)



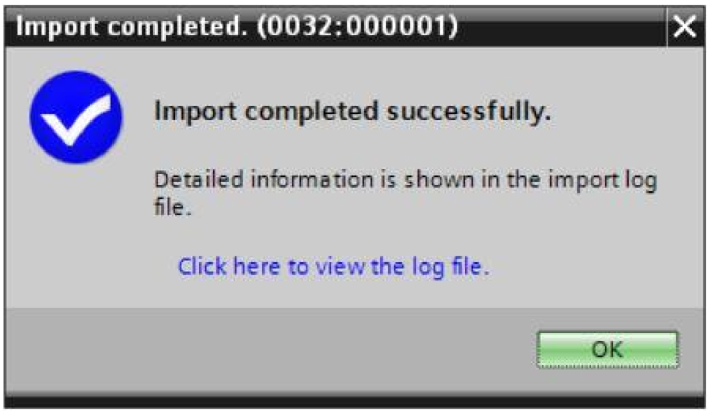
7.4 Import "Tag_table_sorting_station"

→ To insert an existing symbol table, right-click on an empty field of the created "Tag_table_sorting_station". Select "Import file" in the shortcut menu. (→ Right-click in an empty field of the tag table → Import file)



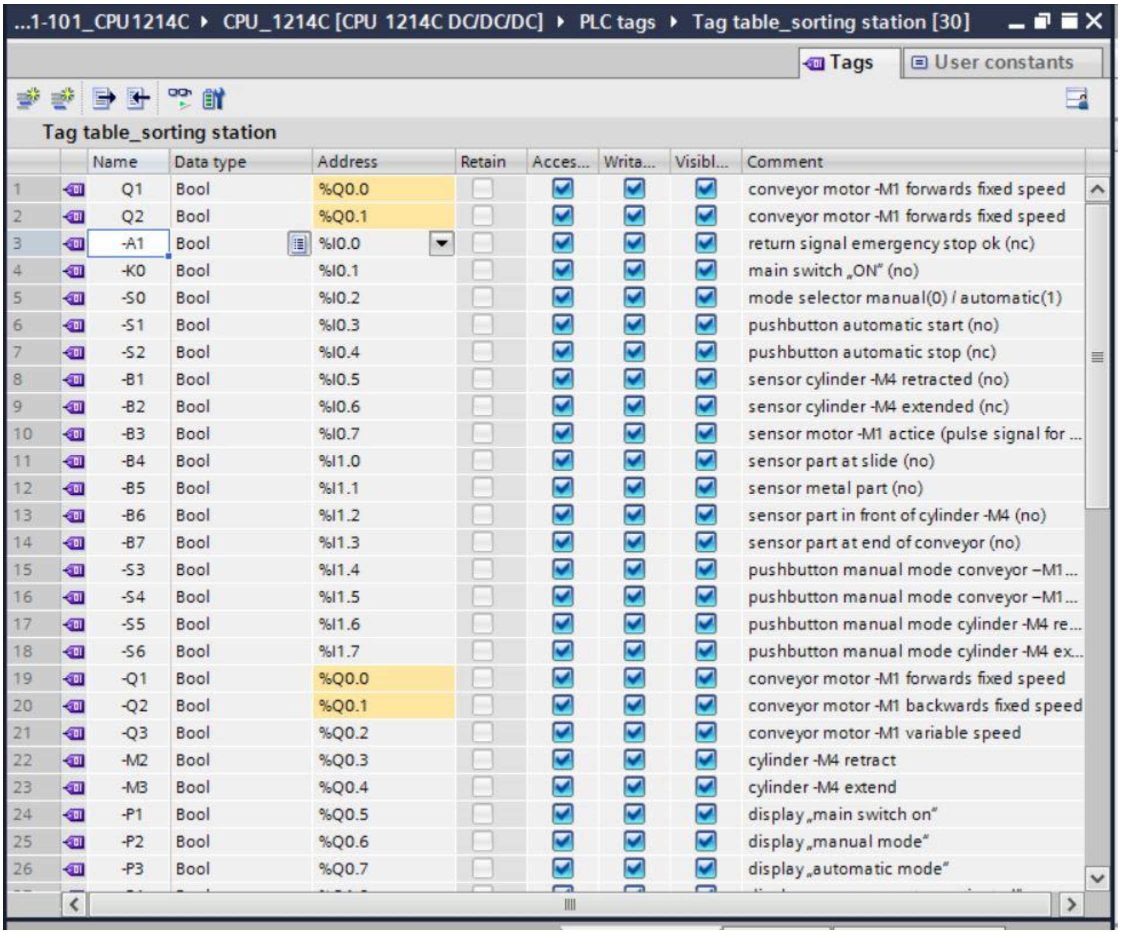
→ Select the desired symbol table (e.g. in .xlsx format) and confirm the selection with "Open". (→ SCE_EN_020-100_Tag_table_sorting_station… → Open)

→ When the import is finished, you will see a confirmation window and have an opportunity to view the log file for the import. Click → OK.



→ You can see that some addresses have been highlighted in orange. These are duplicate addresses and the names of the associated tags have been numbered automatically to avoid confusion.

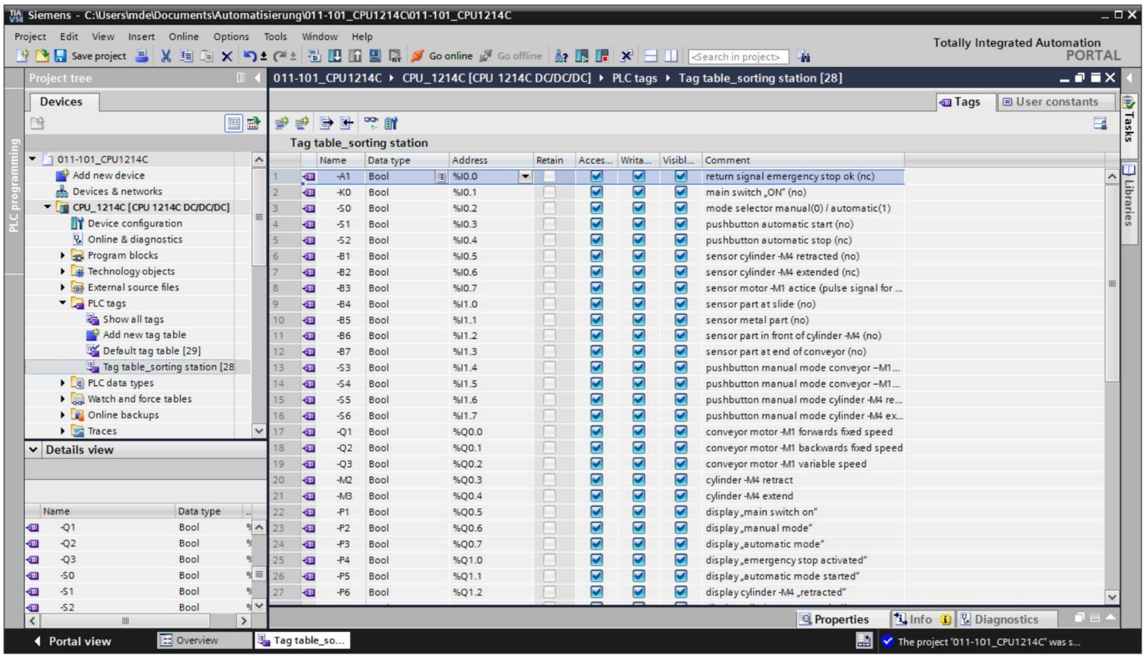
→ Delete the duplicate tags by selecting the lines and pressing the Del key on your keyboard or selecting "Delete" in the shortcut menu. (→ Right-click on selected tags → Delete)



	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	Q1	Bool	%Q0.0	☐	☑	☑	☑	conveyor motor -M1 forwards fixed speed
2	Q2	Bool	%Q0.1	☐	☑	☑	☑	conveyor motor -M1 forwards fixed speed
3	-A1	Bool	%I0.0	☐	☑	☑	☑	return signal emergency stop ok (nc)
4	-K0	Bool	%I0.1	☐	☑	☑	☑	main switch „ON" (no)
5	-S0	Bool	%I0.2	☐	☑	☑	☑	mode selector manual(0) / automatic(1)
6	-S1	Bool	%I0.3	☐	☑	☑	☑	pushbutton automatic start (no)
7	-S2	Bool	%I0.4	☐	☑	☑	☑	pushbutton automatic stop (nc)
8	-B1	Bool	%I0.5	☐	☑	☑	☑	sensor cylinder -M4 retracted (no)
9	-B2	Bool	%I0.6	☐	☑	☑	☑	sensor cylinder -M4 extended (nc)
10	-B3	Bool	%I0.7	☐	☑	☑	☑	sensor motor -M1 actice (pulse signal for ...
11	-B4	Bool	%I1.0	☐	☑	☑	☑	sensor part at slide (no)
12	-B5	Bool	%I1.1	☐	☑	☑	☑	sensor metal part (no)
13	-B6	Bool	%I1.2	☐	☑	☑	☑	sensor part in front of cylinder -M4 (no)
14	-B7	Bool	%I1.3	☐	☑	☑	☑	sensor part at end of conveyor (no)
15	-S3	Bool	%I1.4	☐	☑	☑	☑	pushbutton manual mode conveyor –M1 ...
16	-S4	Bool	%I1.5	☐	☑	☑	☑	pushbutton manual mode conveyor –M1 ...
17	-S5	Bool	%I1.6	☐	☑	☑	☑	pushbutton manual mode cylinder -M4 re...
18	-S6	Bool	%I1.7	☐	☑	☑	☑	pushbutton manual mode cylinder -M4 ex...
19	-Q1	Bool	%Q0.0	☐	☑	☑	☑	conveyor motor -M1 forwards fixed speed
20	-Q2	Bool	%Q0.1	☐	☑	☑	☑	conveyor motor -M1 backwards fixed speed
21	-Q3	Bool	%Q0.2	☐	☑	☑	☑	conveyor motor -M1 variable speed
22	-M2	Bool	%Q0.3	☐	☑	☑	☑	cylinder -M4 retract
23	-M3	Bool	%Q0.4	☐	☑	☑	☑	cylinder -M4 extend
24	-P1	Bool	%Q0.5	☐	☑	☑	☑	display „main switch on"
25	-P2	Bool	%Q0.6	☐	☑	☑	☑	display „manual mode"
26	-P3	Bool	%Q0.7	☐	☑	☑	☑	display „automatic mode"

→ You now have a complete symbol table of the digital inputs and outputs in front of you. Save your project under the name 031-100_FC Programming.

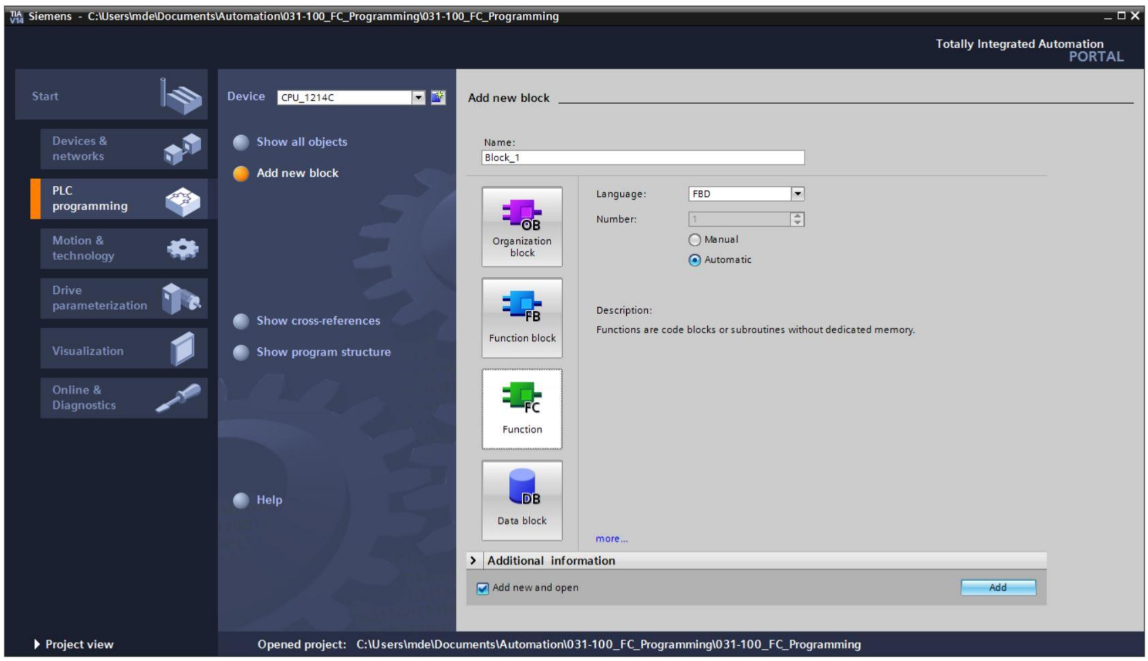
(→ Project → Save as ... →031-100_FC Programming → Save)



7.5 Create function FC1 "MOTOR_MANUAL" for the conveyor motor in manual mode

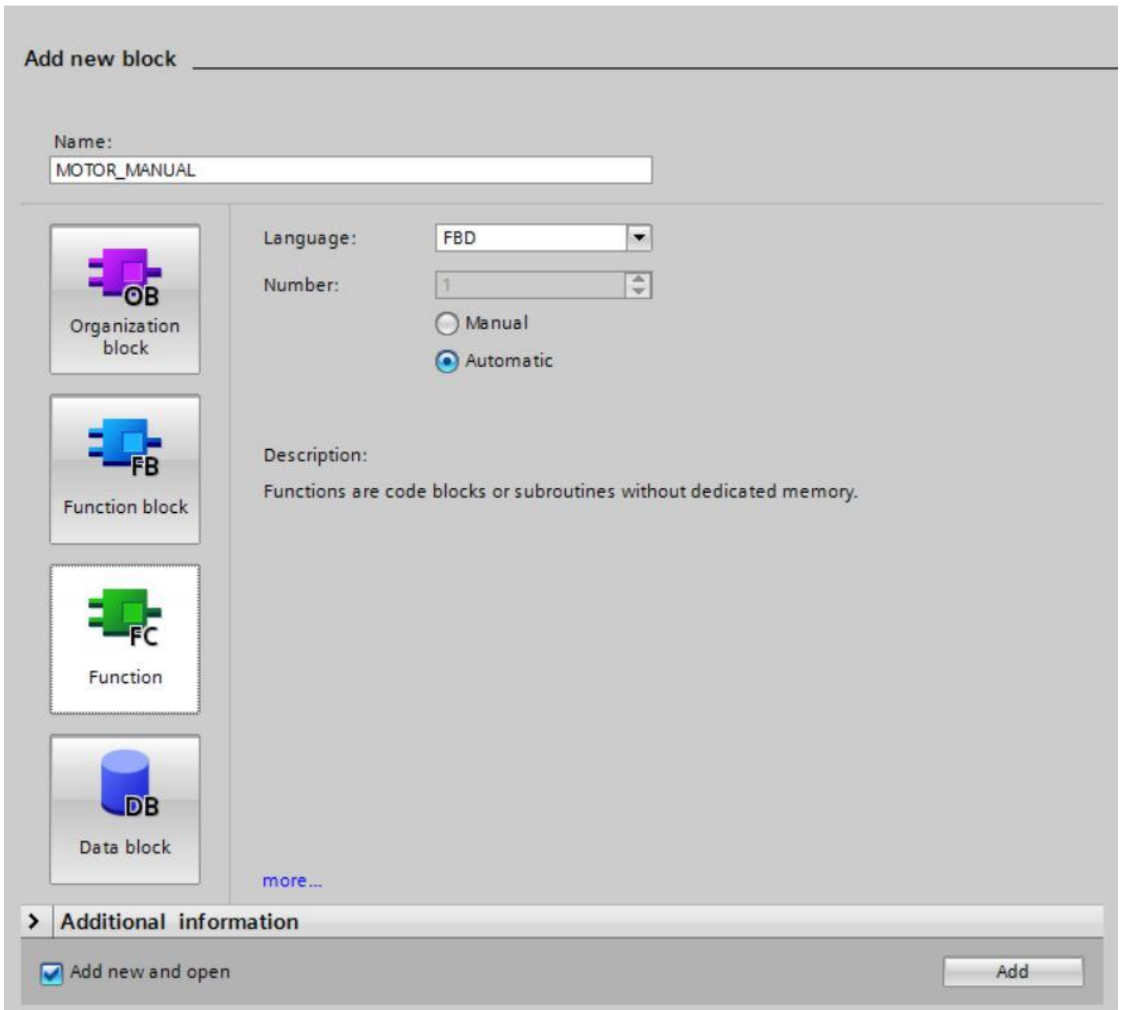
→ In the PLC programming section of the portal view, click "Add new block" to create a new function.

(→ PLC programming → Add new block → )



5

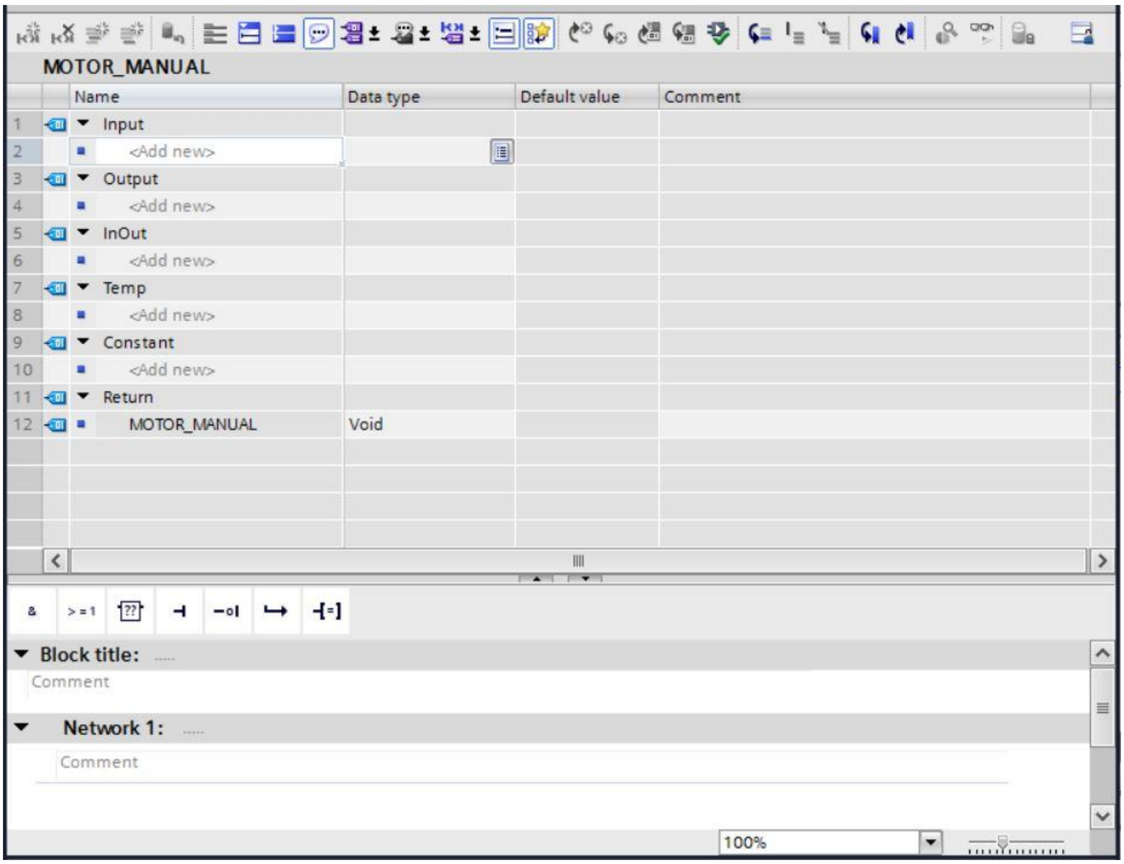
→ Rename your new block to: "MOTOR_MANUAL", set the language to FBD and keep automatic assignment of the number. Select the "Add new and open" check box. You will thus be taken automatically to your created function block in the project view.Click "Add". (→ Name: MOTOR_MANUAL→ Language: FBD → Number: Automatic → ☑ Add new and open → Add)



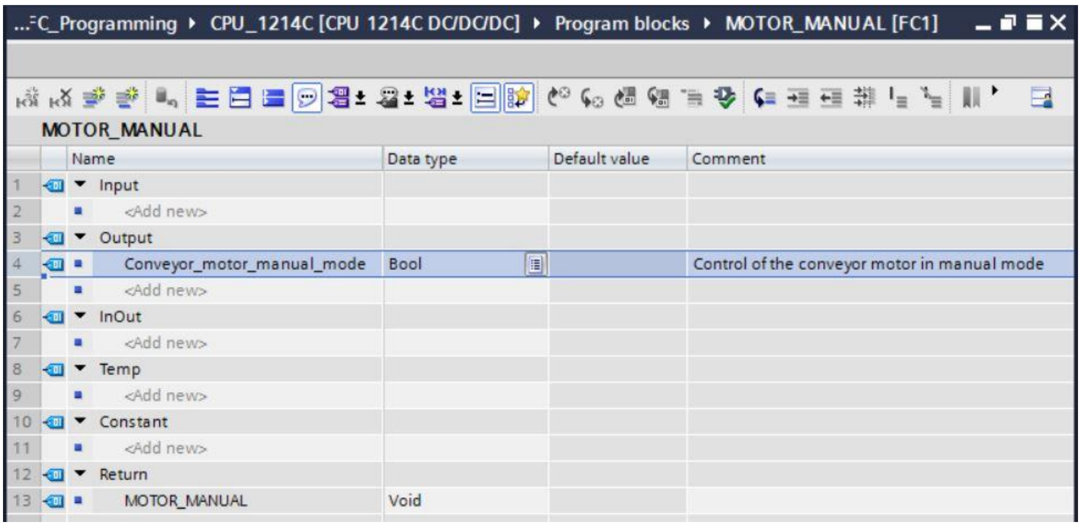
7.6 Define the Interface of function FC1 "MOTOR_MANUAL"

If you selected "Add new and open", the project view opens with a window for creating the block you just added.

→ You can find the interface description of your function in the upper section of your programming view.



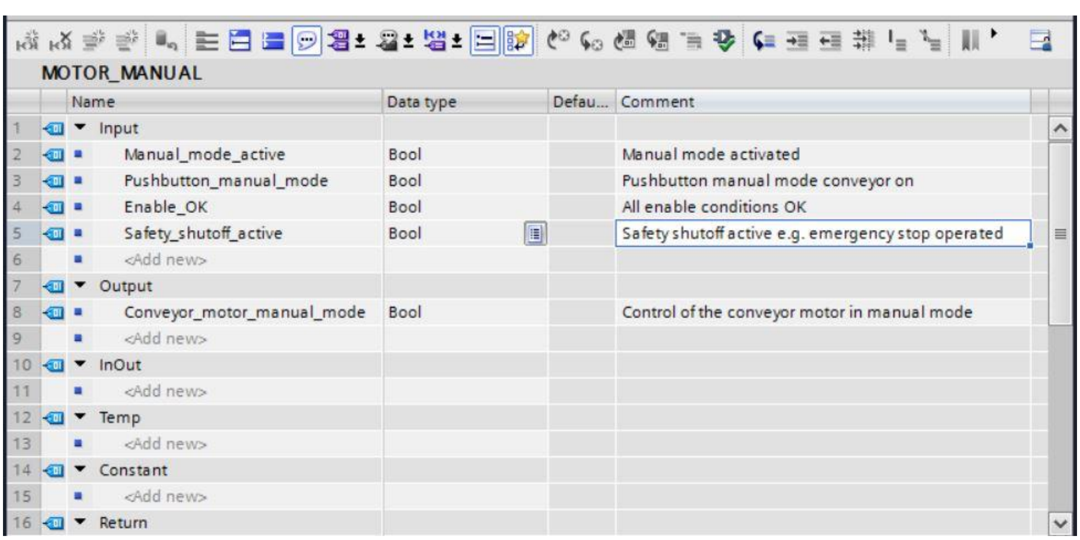
→ A binary output signal is needed for controlling the conveyor motor. For this reason, we first create local output tag #Conveyor_motor_manual_mode of the "Bool" type. Enter comment "Control of the conveyor motor in manual mode" for the parameter. (→ Output: Conveyor_motor_manual_mode → Bool → Control of the conveyor motor in manual mode)



→ Add parameter #Manual_mode_active as the input interface under Input and confirm the entry with the Enter key or by exiting the entry field. Data type "Bool" is assigned automatically. This will be retained. Next, enter the associated comment "Manual mode activated".

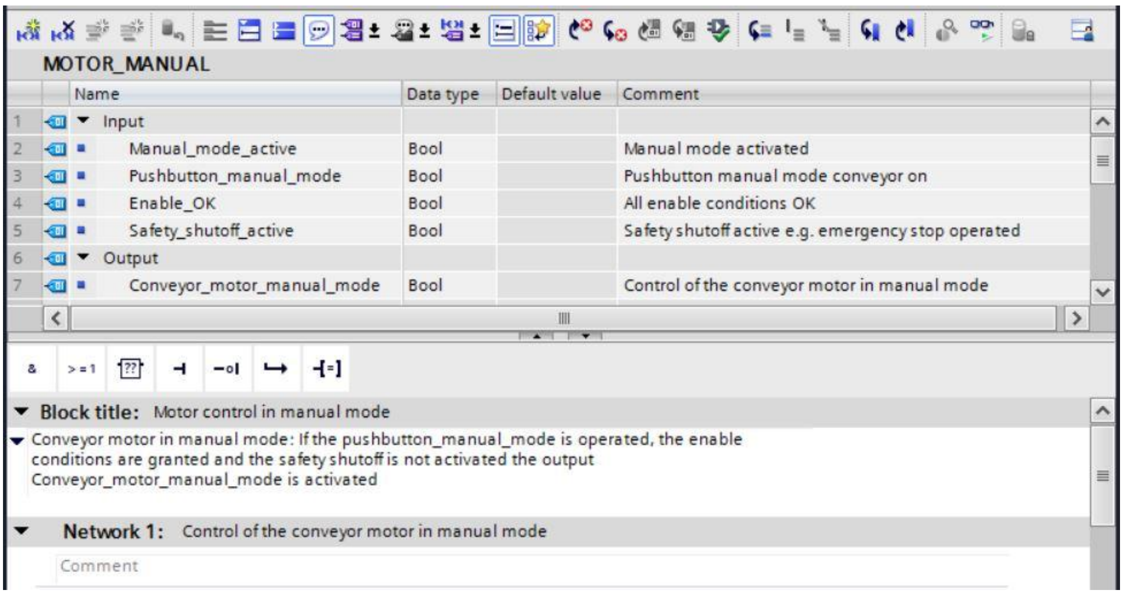
(→ Manual_mode_active → Enter → Bool → Manual mode activated)

→ Continue by adding parameters #Pushbutton_manual_mode, #Enable_OK and #Safety_shutoff_active as additional binary input parameters and check their data types. Add descriptive comments.



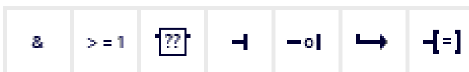
→ For purposes of program documentation, assign the block title, a block comment and a helpful network title for Network 1.

(→ Block title: Motor control in manual mode → Network 1: Control of the conveyor motor in manual mode)



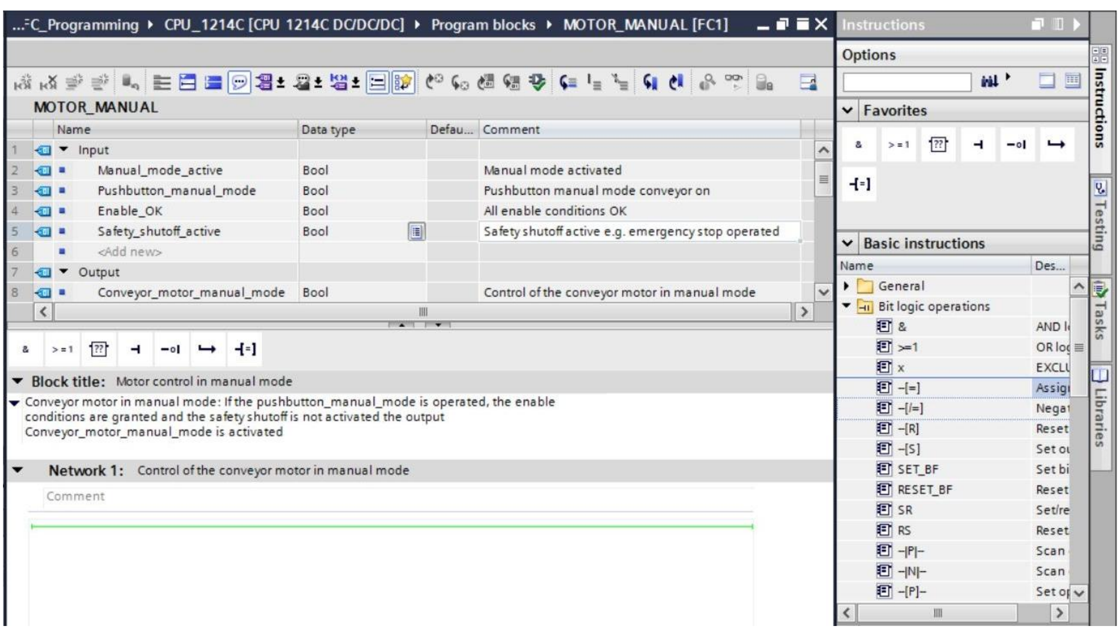
7.7 Program FC1: MOTOR_MANUAL

→ Below the interface description, you see a toolbar in the programming window with various logic functions and below that an area with networks. We have already specified the block title and the title for the first network there. Programming is performed within the networks using individual logic blocks. Distribution among multiple networks helps to preserve the clarity of the program. In the following, you will get to know the various ways you can insert logic blocks.



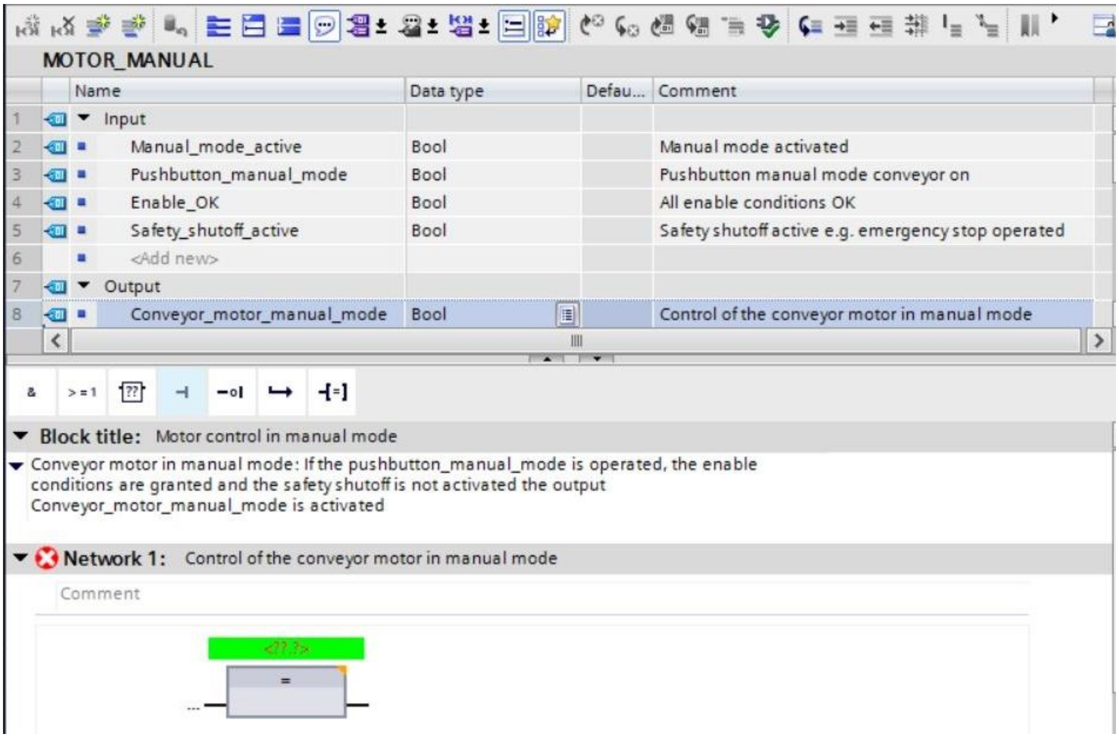
→ On the right side of your programming window is a list of instructions you can use in the program. Under → Basic instructions → Bit logic operations, find function –[=] (Assignment) and use a drag & drop operation to move it to Network 1 (green line appears, mouse pointer with + symbol).

(→ Instructions → Basic instructions → Bit logic operations → –[=])



→ Now use drag & drop to move your output parameter #Conveyor_motor_manual_mode onto <??.?> above the block you just inserted. The best way to select a parameter in the interface description is by "grabbing" it at the blue symbol .

(→  Conveyor_motor_manual_mode)

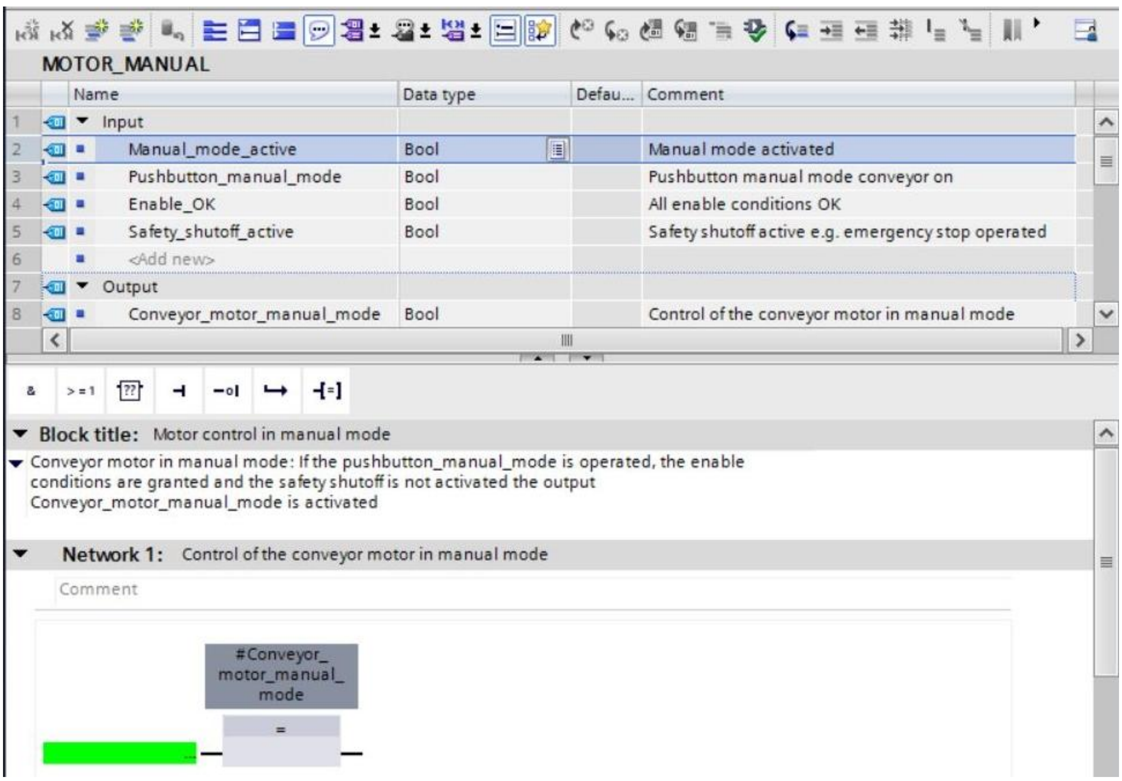


5

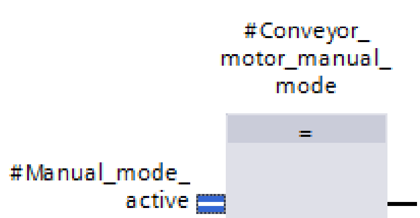
5

→ This determines that the #Conveyor_motor_manual_mode parameter is written by this block. Still missing, however, are the input conditions so that this actually happens. For this, use drag & drop to move input parameter #Manual_mode_active to the left side of the assignment block.

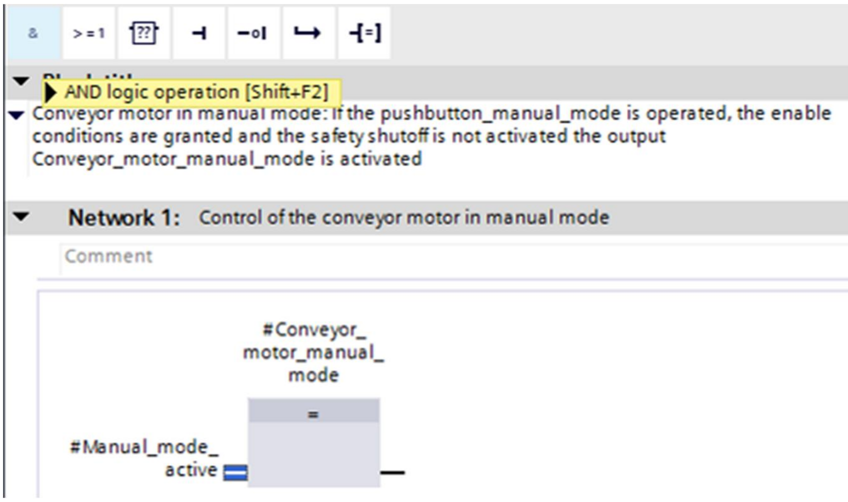
(→ Manual_mode_active)



→ The input of the assignment block will also be logically combined with other parameters by an AND logic operation. To do this, first click the input of the block to which #Manual_mode_active is already connected, so that the input line has a blue background.

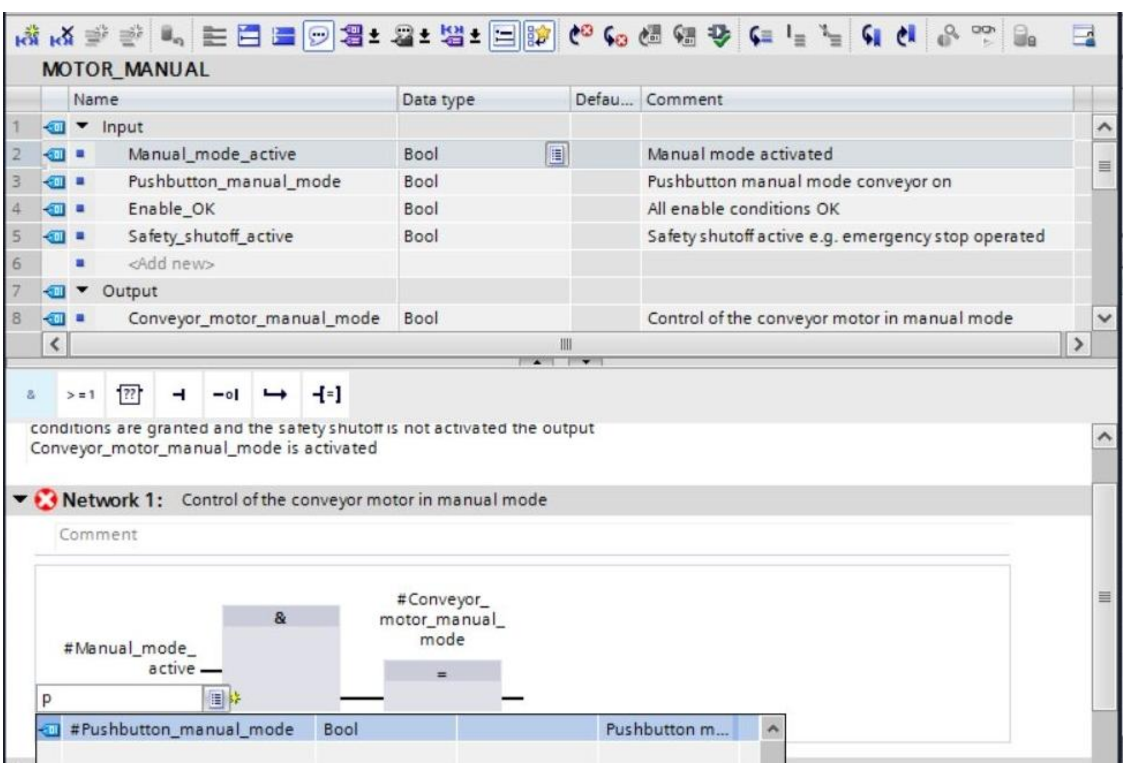


→ Click the & icon in your logic toolbar to insert an AND logic operation between the #Manual_mode_active tag and your assignment block.



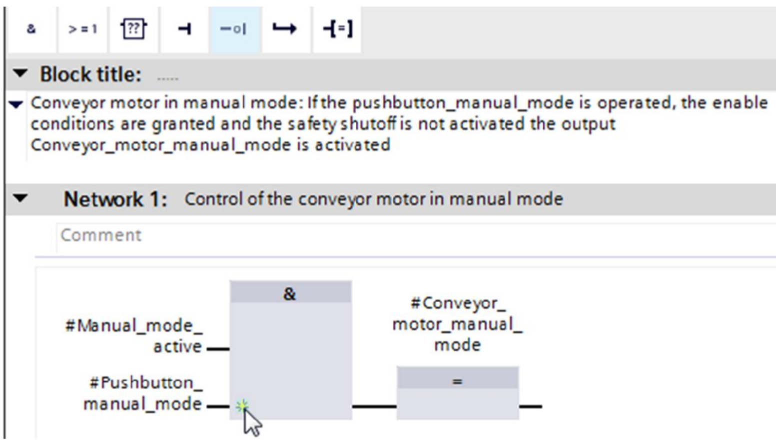
→ Double-click the second input of the & logic operation <??.?> and enter the letter "P" in the field that appears in order to see a list of available tags starting with "P". Click the #Pushbutton_manual_mode tag and apply with → Enter.

(→ & block → <??.?> → P → #Pushbutton_manual_mode → Enter)

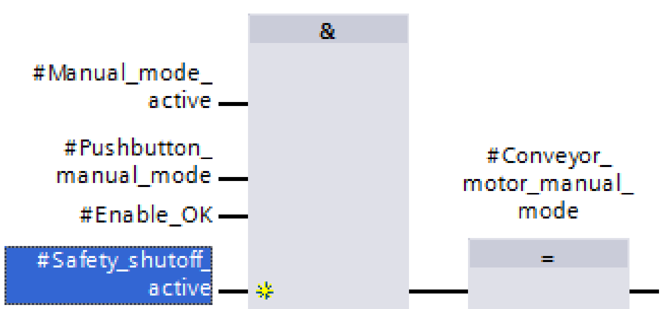


Note: *When assigning tags in this way, there is a risk of a mix-up with the global tags from the tag table. The previously presented procedure using drag & drop from the interface description should therefore be used preferentially.*

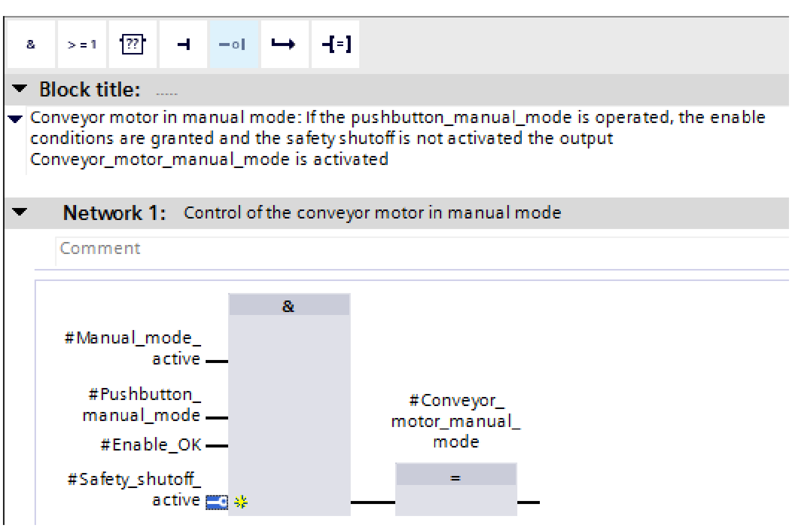
→ To ensure that the output can only be controlled when the enable conditions are met and the safety shutoff is not active, the #Enable_OK and #Safety_shutoff_active input tags are logically combined with the AND logic operation. To do this, click twice on the yellow star  of your AND block to add two additional inputs.



→ Add input tags #Enable_OK and #Safety_shutoff_active to your newly created inputs of the AND block.

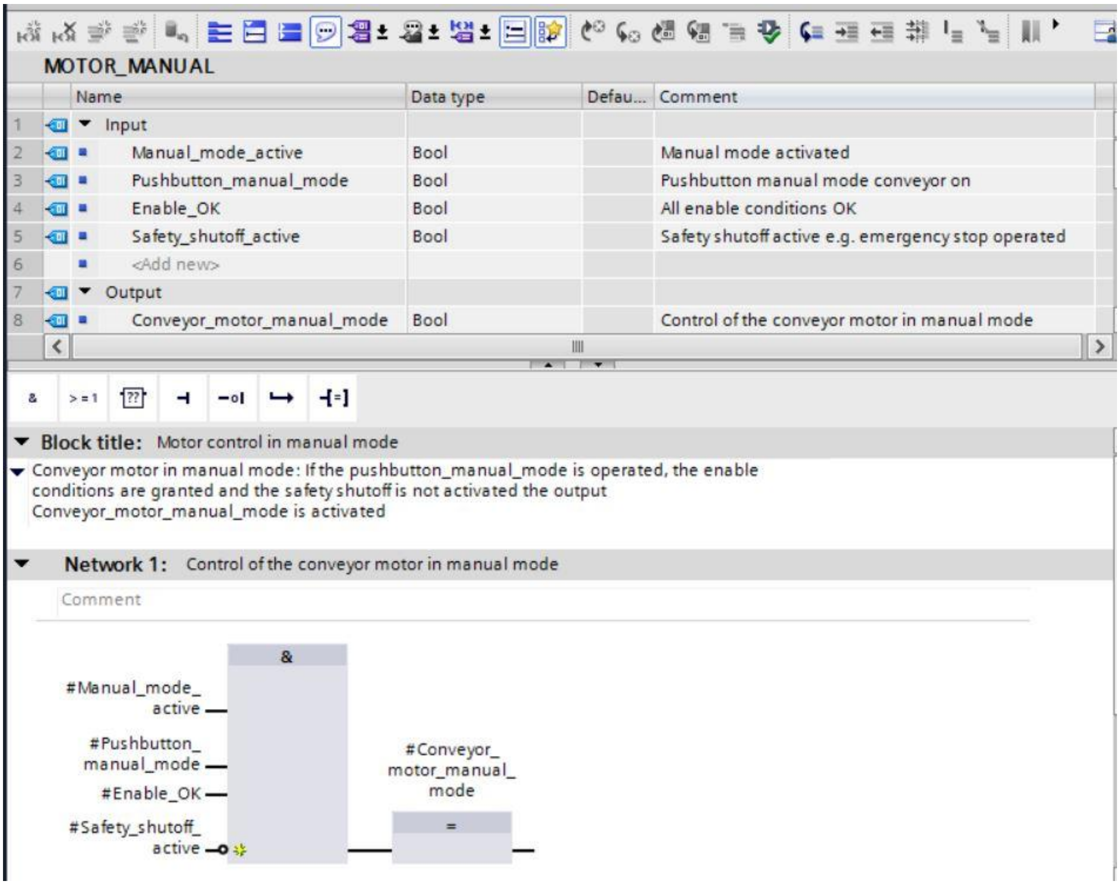


→ Negate the input connected to parameter #Safety_shutoff_active by selecting it and clicking .



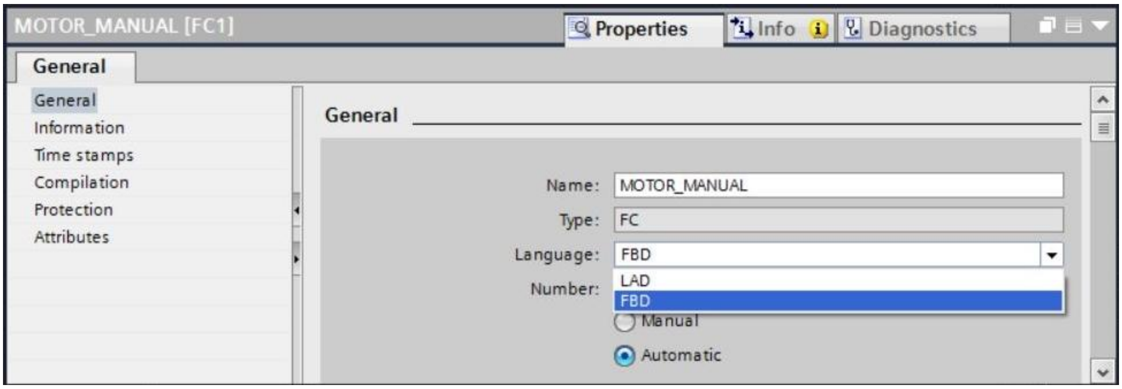
→ Do not forget to click 

regularly. The finished function "MOTOR_MANUAL" [FC1] in FBD is shown below.

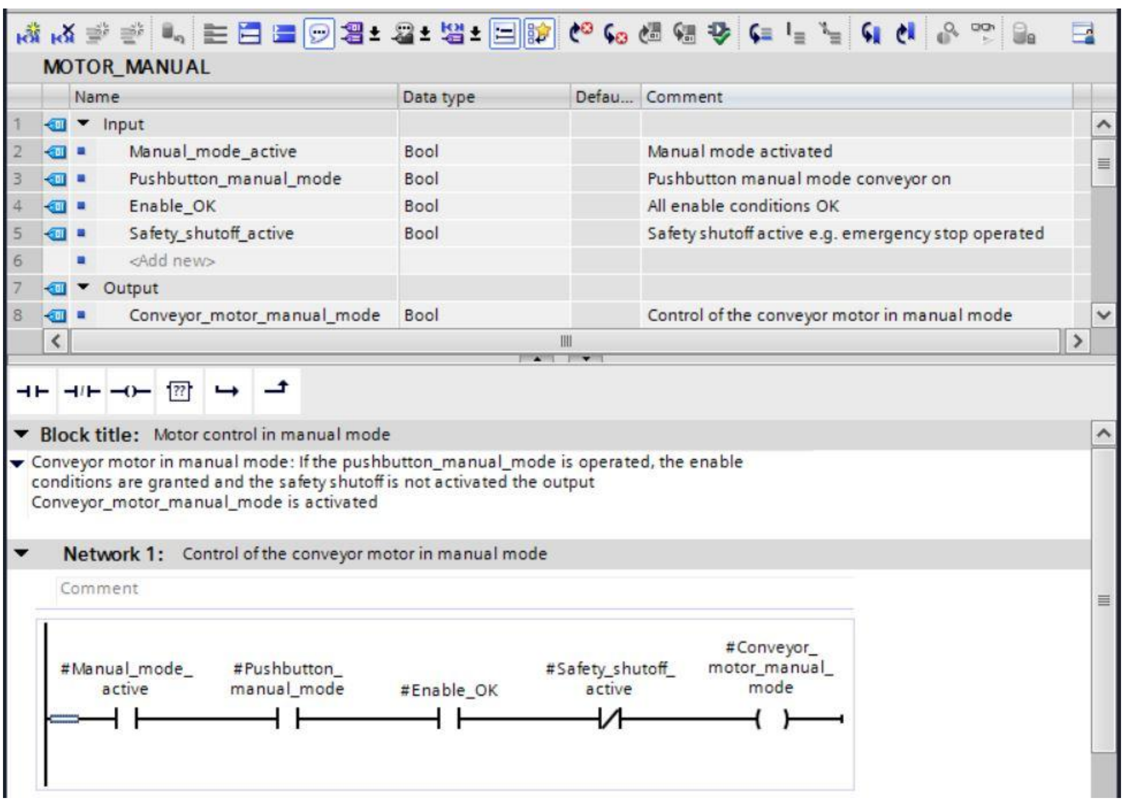


5

→ Under "General" in the properties of the block, you can change the "Language" to LAD (Ladder Logic) (→Properties → General → Language: LAD)



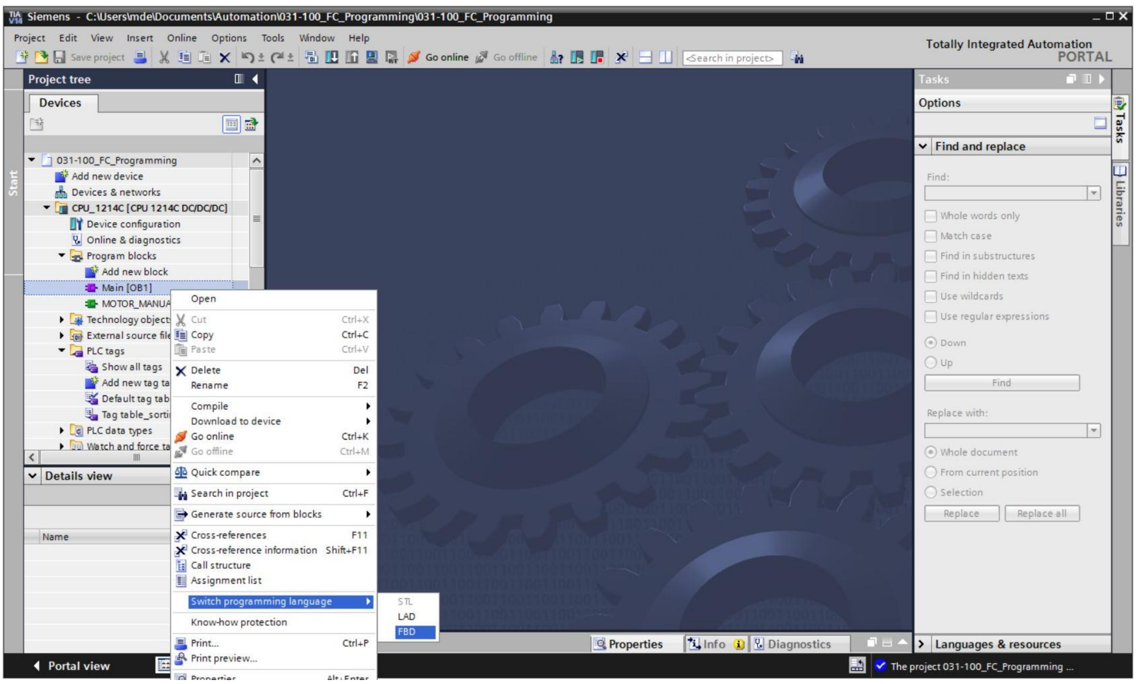
→ The program has the following appearance in LAD.



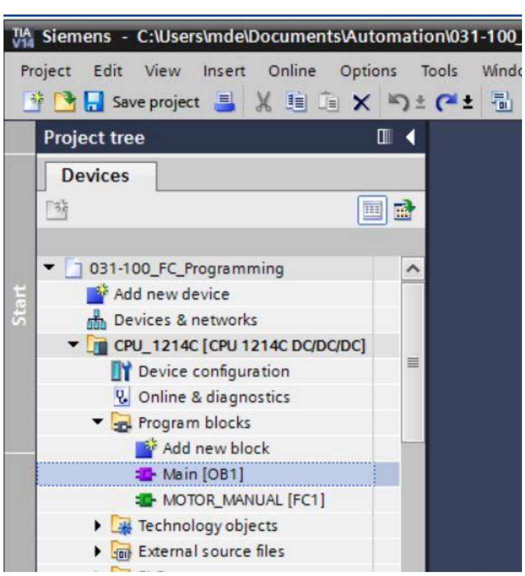
7.8 Program the organization block OB1 – Control conveyor tracking forwards in manual mode

→ Before programming organization block "Main [OB1]", we switch the programming language to FBD (Function Block Diagram). To do so, first click on "Main [OB1]" in the "Program blocks" folder.

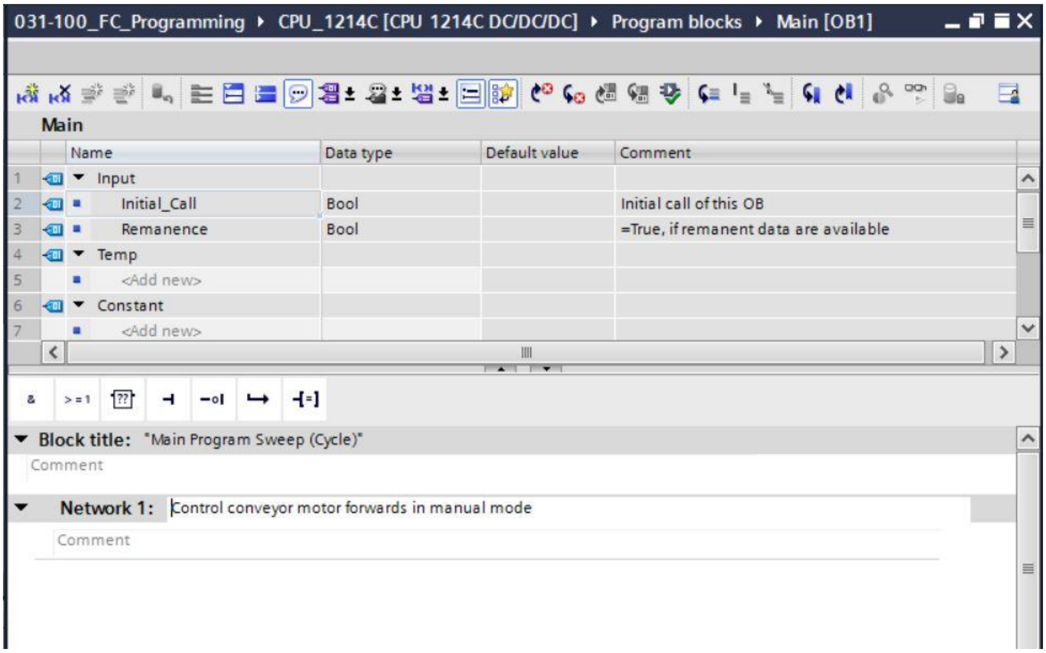
(→ CPU_1214C [CPU 1214C DC/DC/DC → Program blocks → Main [OB1] → Switch programming language → FBD)



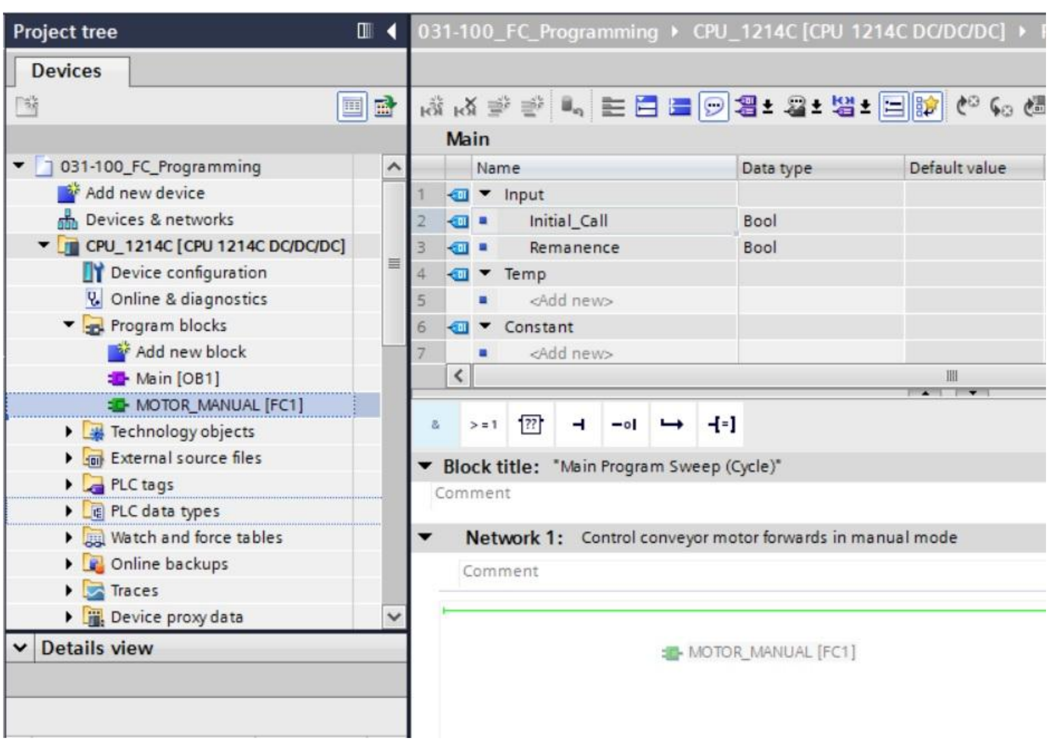
→ Open the "Main [OB1]" organization block with a double-click.



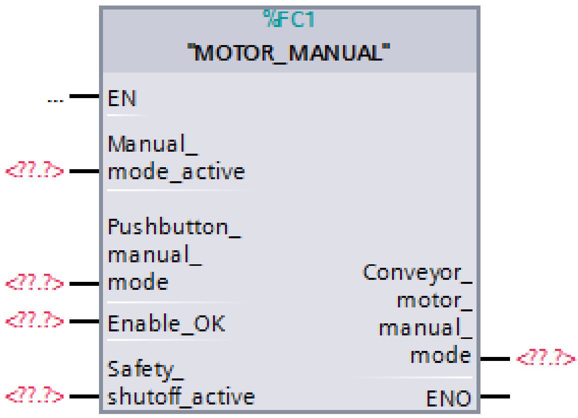
→ Assign Network 1 the name "Control conveyor tracking forwards in manual/jog mode" (→ Network 1:... →Control conveyor tracking forwards in manual/jog mode)



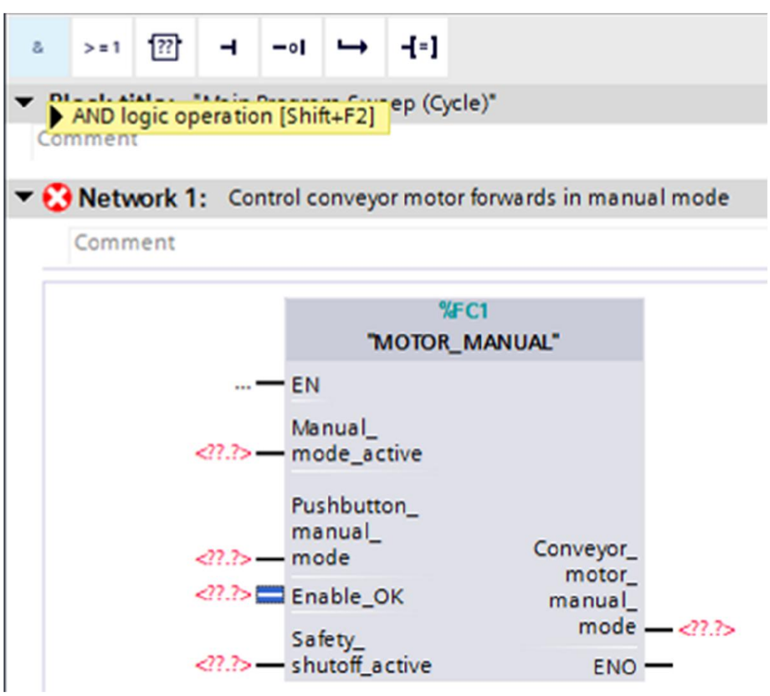
→ Use drag & drop to move your "MOTOR_MANUAL [FC1]" function onto the green line in Network 1.



→ A block with the interface you defined and connections EN and ENO are inserted in Network 1.

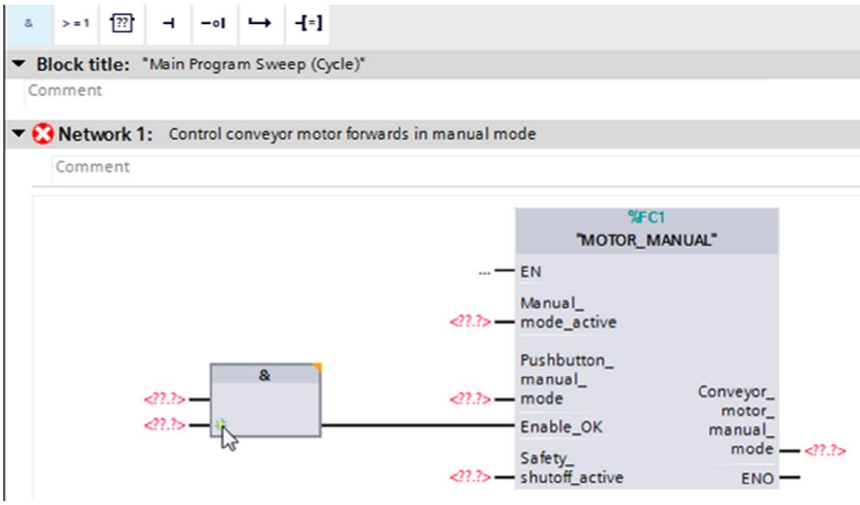


→ To insert an AND before input parameter "Enable_OK", select this input and insert the AND by clicking the & icon in your logic toolbar (→ &).



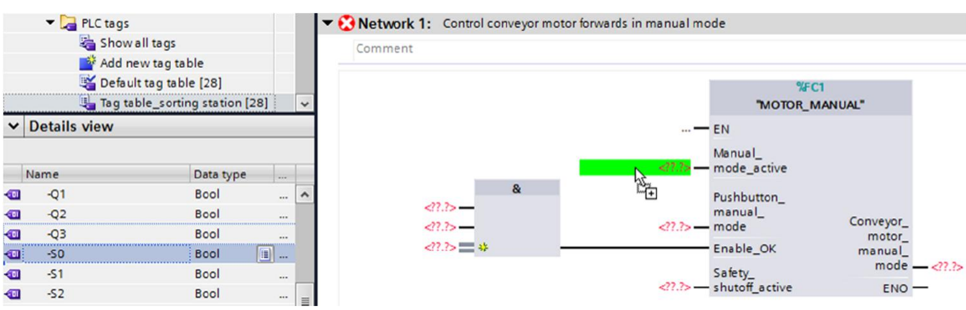
5

→ Click the yellow star of the AND block to add another input (→).

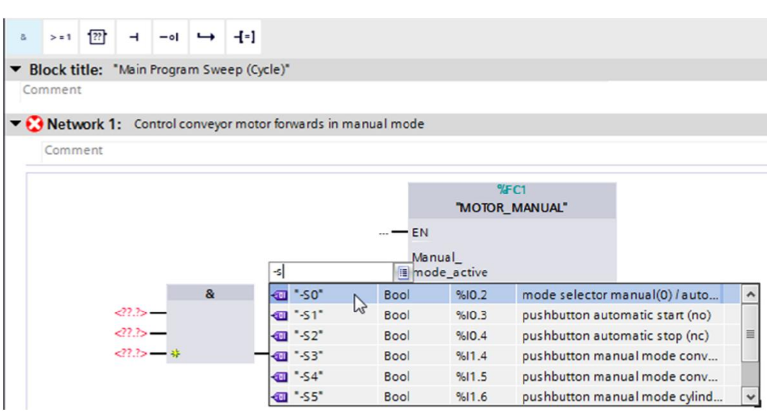


→ To connect the block to the global tags from "Tag_table_sorting_station", we have two options:

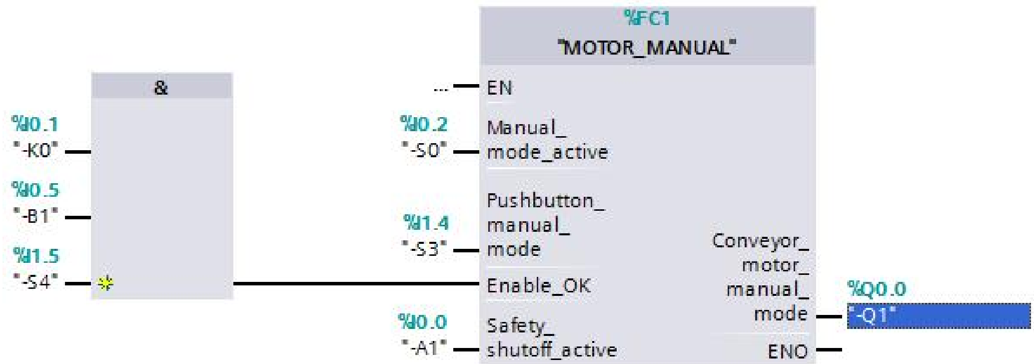
→ Either select the "Tag_table_sorting_station" in the project tree and use drag & drop to move the desired global tag from the Details view to the interface of FC1 (→ Tag_table_sorting_station → Details view. → -S0 → Manual_mode_active)



→ Or, enter the starting letters (e.g. "-S") of the desired global tag for <??.?> and select the global input tag "-S0" (%I0.2) from the displayed list (→ Manual_mode_active → -S → -S0).

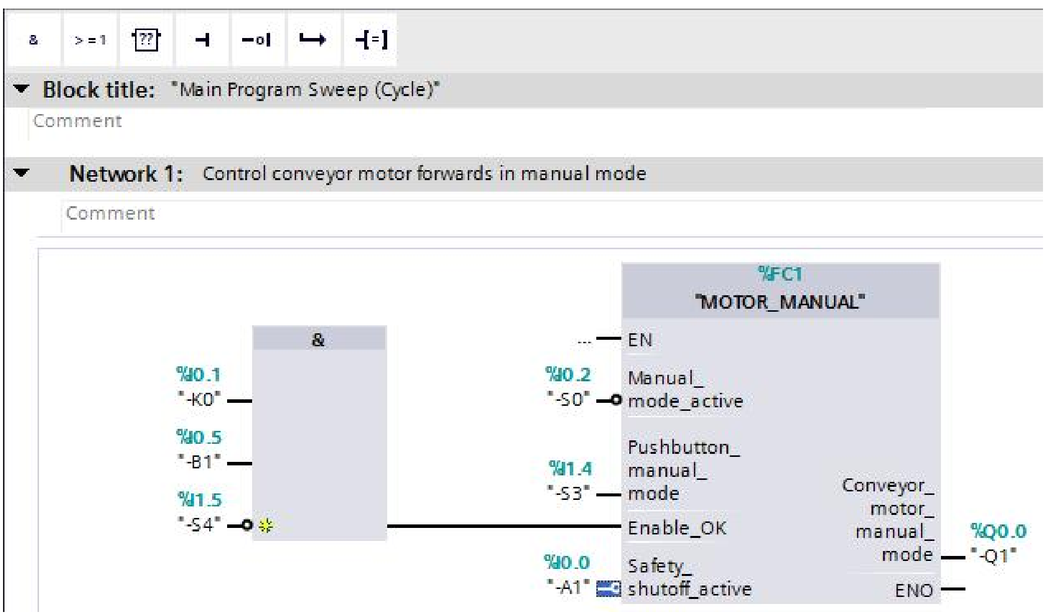


→ Insert the other input tags "-S3", "-K0", "-B1", "-S4" and "-A1" and then insert output tag "-Q1" (%Q0.0) at output "Conveyor_motor_manual_mode".



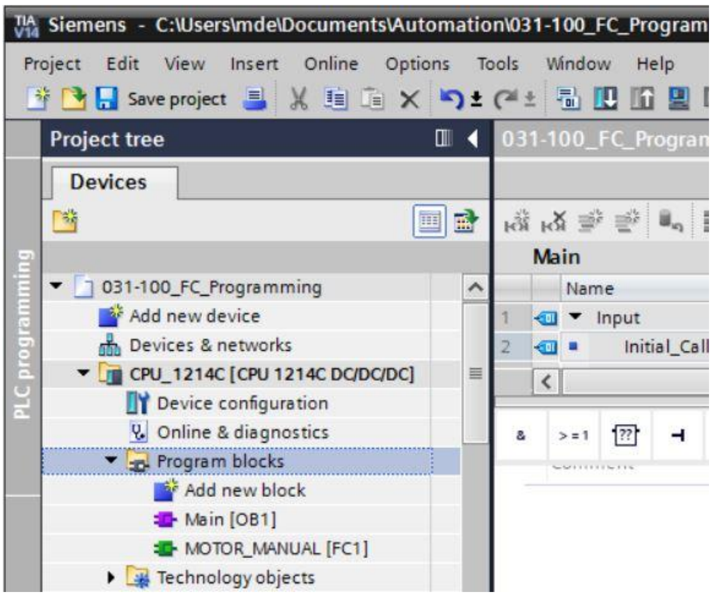
→ Negate the querying of input tags "-S0", "-S4" and "-A1" by selecting them and clicking .

(→ -S0 → → -S4 → → -A1 →)

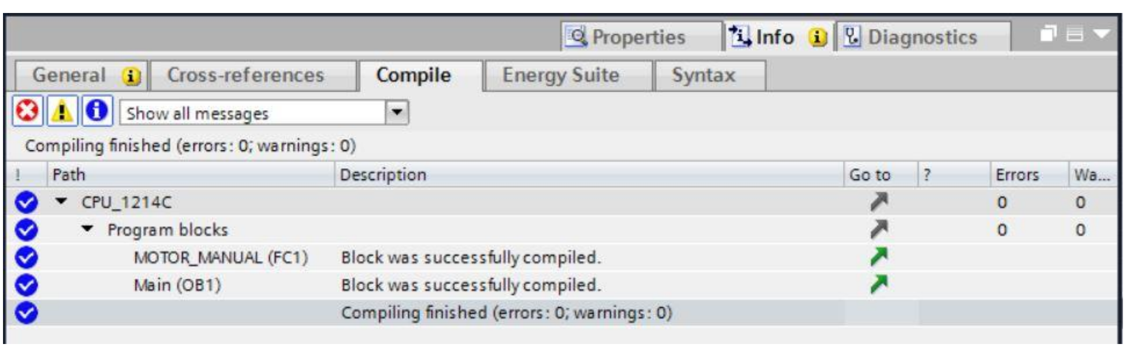


7.9 Save and compile the program

→ To save your project, select the Save project button in the menu. To compile all blocks, click the "Program blocks" folder and select the icon for compiling in the menu (→ Save project → Program blocks →).

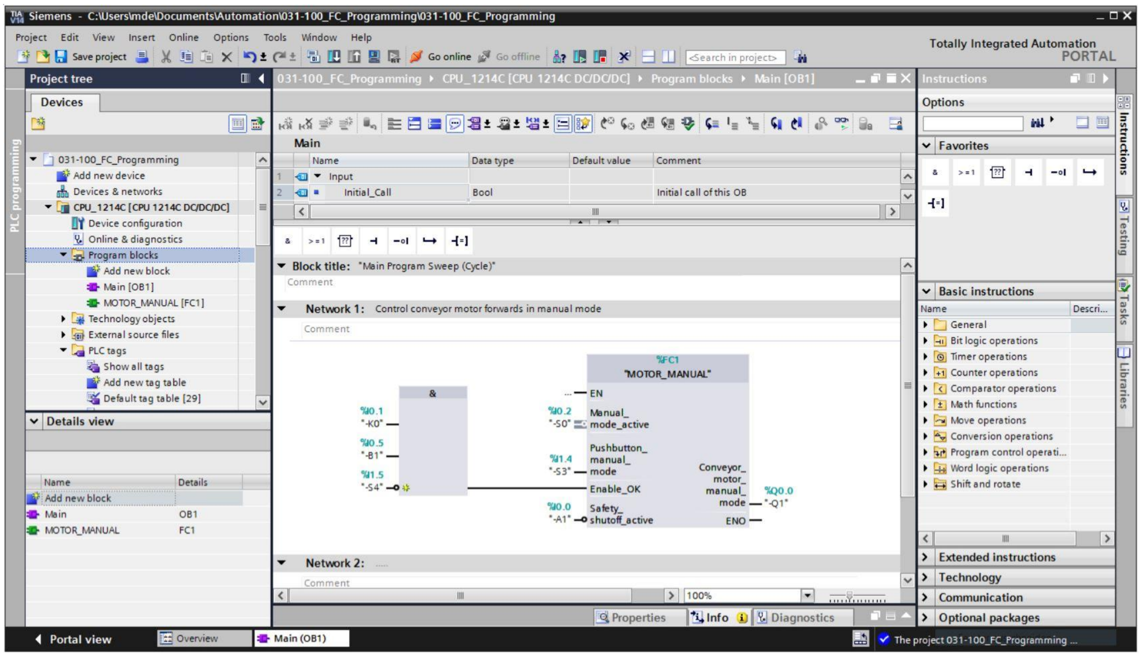


→ The "Info", "Compile" area shows which blocks were successfully compiled.



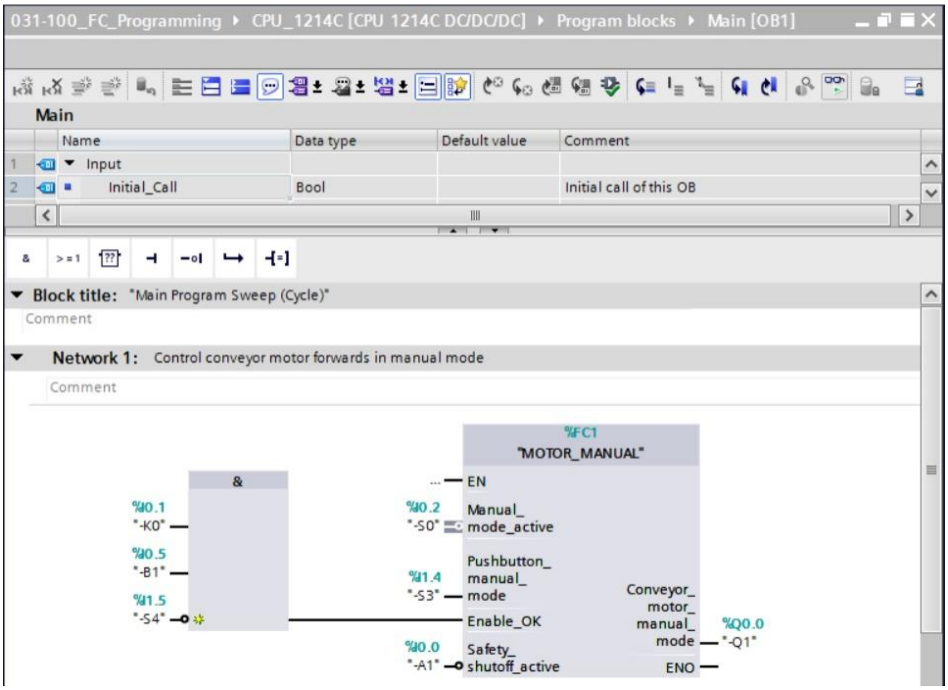
7.10 Download the program

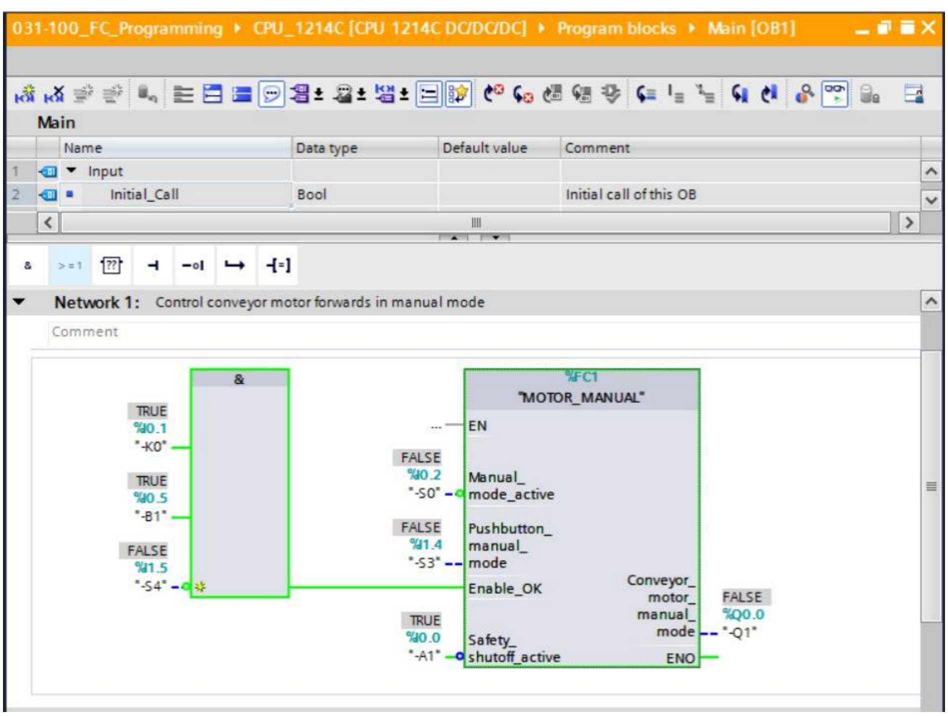
→ After successful compilation, the complete controller with the created program, as previously described in the modules for hardware configuration, can be downloaded (→ ).



7.11 Monitor program blocks

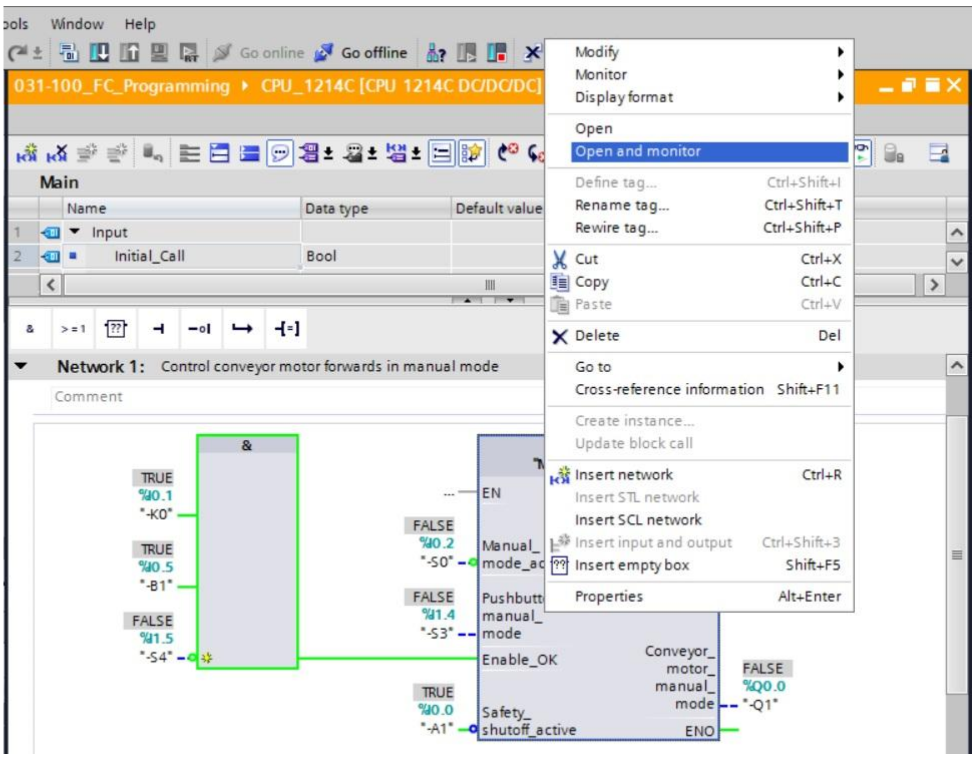
→ The desired block must be open for monitoring the downloaded program. The monitoring can now be activated/deactivated by clicking the icon (→ Main [OB1] →).

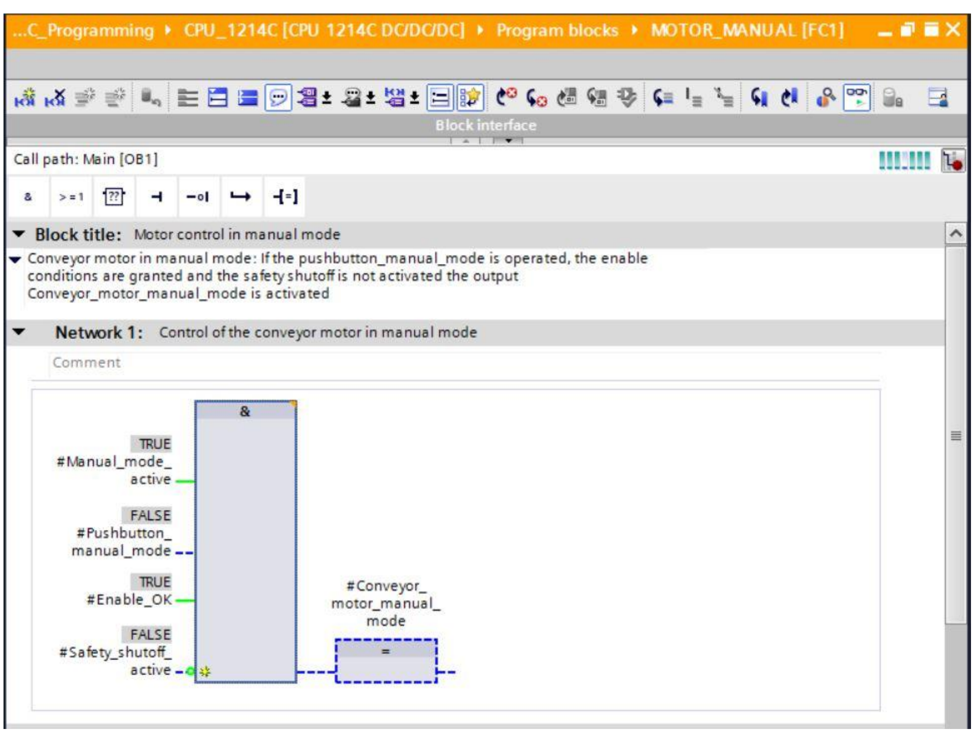




Note: *The monitoring here is signal-related and controller-dependent. The signal states at the terminals are indicated with TRUE or FALSE.*

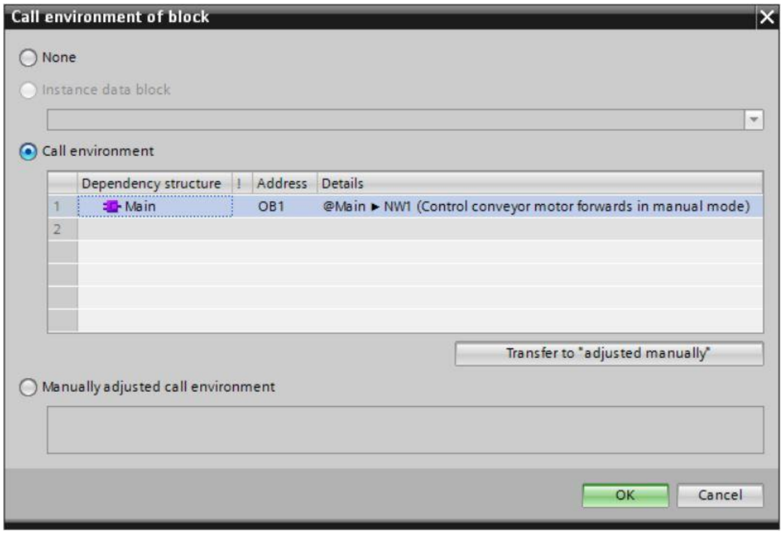
→ The "MOTOR_MANUAL" [FC1] function called in the "Main [OB1]" organization block can be selected directly for "Open and monitor" after right-clicking (→ "MOTOR_MANUAL" [FC1] → Open and monitor).





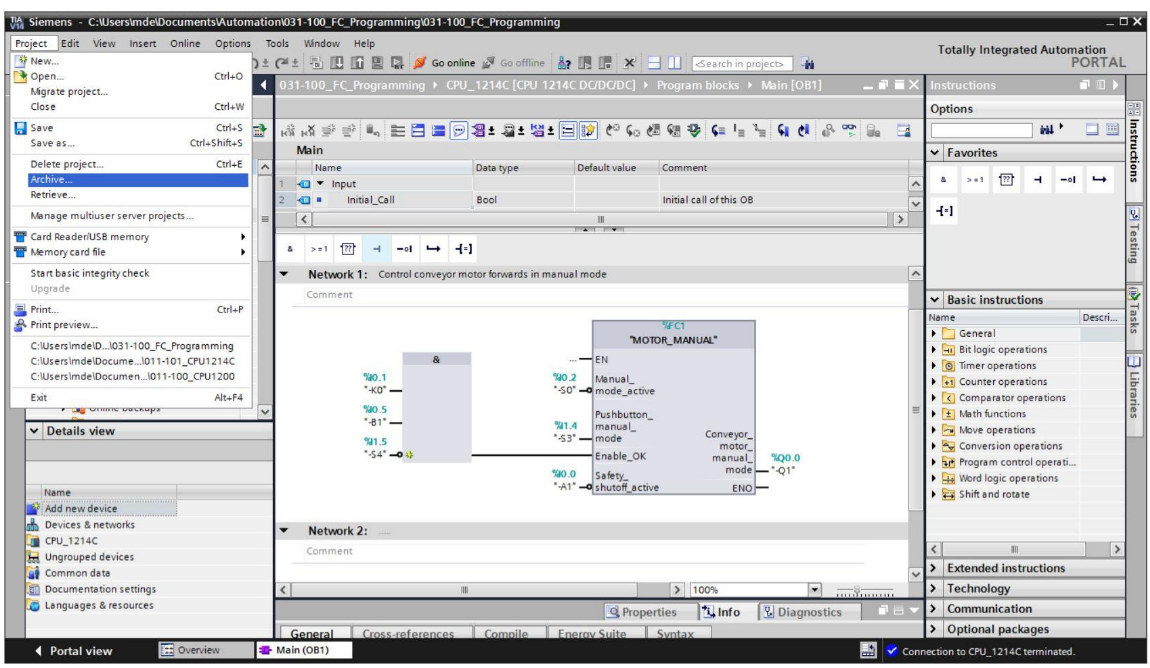
Note: *The monitoring here is function-related and controller-independent. The actuation of sensors and the station status are shown here with TRUE or FALSE.*

→ If a particular point of use of the "MOTOR_MANUAL" [FC1] function is to be monitored, the call environment can be selected using the icon (→ → Call environment → OK)



7.12 Archive the project

→ As the final step, we want to archive the complete project. Select the → "Archive ..." command in the → "Project" menu. Select a folder where you want to archive your project and save it with the file type "TIA Portal project archive". (→ Project → Archive → TIA Portal project archive → 031-100_FC Programming…. → Save)



7.13 Checklist

No.	Description	Completed
1	Compiling successful and without error message	
2	Download successful and without error message	
3	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated MANUAL mode (-S0 = 0) Activate manual mode conveyor forwards (-S3 = 1) Conveyor motor forwards fixed speed (-Q1 = 1)	
4	Same as 3 but activate EMERGENCY OFF (-A1 = 0) → -Q1 = 0	
5	Same as 3 but AUTO mode (-S0 = 1) → -Q1 = 0	
6	Same as 3 but switch off station (-K0 = 0) → -Q1 = 0	
7	Same as 3 but cylinder not retracted (-B1 = 0) → -Q1 = 0	
8	Same as 8 but also activate manual mode conveyor backwards (-S4 = 1) → -Q1 = 0	
9	Project successfully archived	

5

8 Exercise

8.1 Task – Exercise

The following functions of the sorting station process description will be planned, programmed and tested in this chapter:

– Manual mode – Control of conveyor tracking backwards in manual/jog mode

8.2 Technology diagram

Here, you see the technology diagram for the task.

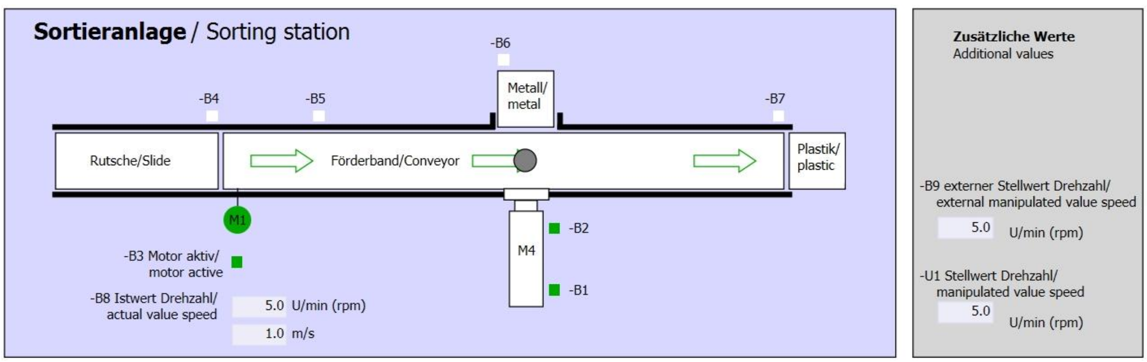


Figure 10: Technology diagram

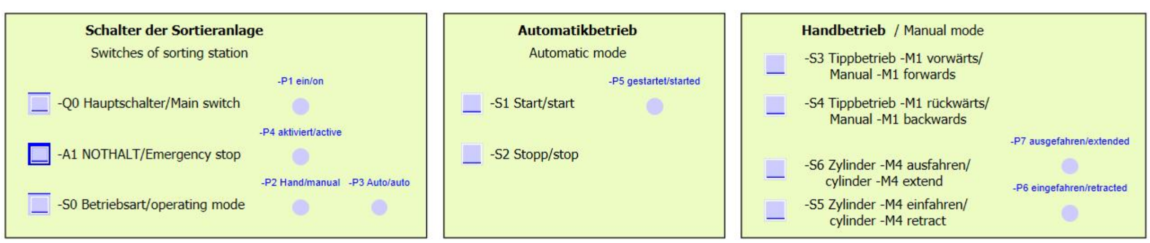


Figure 11: Control panel

8.3 Reference list

The following signals are needed as operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop ok	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.5	BOOL	-B1	Sensor cylinder M4 retracted	NO
I 1.4	BOOL	-S3	Pushbutton manual mode conveyor M1 forward	NO
I 1.5	BOOL	-S4	Pushbutton manual mode conveyor M1 reverse	NO

DO	Type	Identifier	Function	
Q 0.1	BOOL	-Q2	Conveyor motor M1 backwards fixed speed	

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

8.4 Planning

Plan the implementation of the task on your own.

5

5

8.5 Checklist – Exercise

No.	Description	Completed
1	Compiling successful and without error message	
2	Download successful and without error message	
3	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated MANUAL mode (-S0 = 0) Activate manual mode conveyor backwards (-S4 = 1) Conveyor motor backwards fixed speed (-Q2 = 1)	
4	Same as 8 but activate EMERGENCY OFF (-A1 = 0) → -Q2 = 0	
5	Same as 8 but AUTO mode (-S0 = 1) → -Q2 = 0	
6	Same as 8 but switch off station (-K0 = 0) → -Q2 = 0	
7	Same as 8 but cylinder not retracted (-B1 = 0) → -Q2 = 0	
8	Same as 8 but also activate manual mode conveyor forwards (-S3 = 1) → -Q1 = 0 and -Q2 = 0	
9	Project successfully archived	

9 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

5

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is available at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, get in touch with your regional SCE contact siemens.com/sce/contact

6

Additional information regarding SCE

siemens.com/sce

Information regarding use

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

Table of contents

6

6

Basics of FB Programming

1 Goal

In this chapter, you will get to know the basic elements of a control program – the ***organization blocks (OBs), functions (FCs)***, **function blocks (FBs)** and **data blocks (DBs).** In addition, you will be introduced to ***library-compatible*** function und function block programming. You will get to know the ***Function Block Diagram (FBD)*** programming language and use it to program a function block (FB1) and an organization block (OB1).

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

This chapter builds on the hardware configuration of SIMATIC S7 CPU1214C. However, other hardware configurations that has digital input and output board be used. For this chapter, you can use the following project, for example:

SCE_EN_011_101_Hardware_Configuration_CPU1214C.zap14

3 Required hardware and software

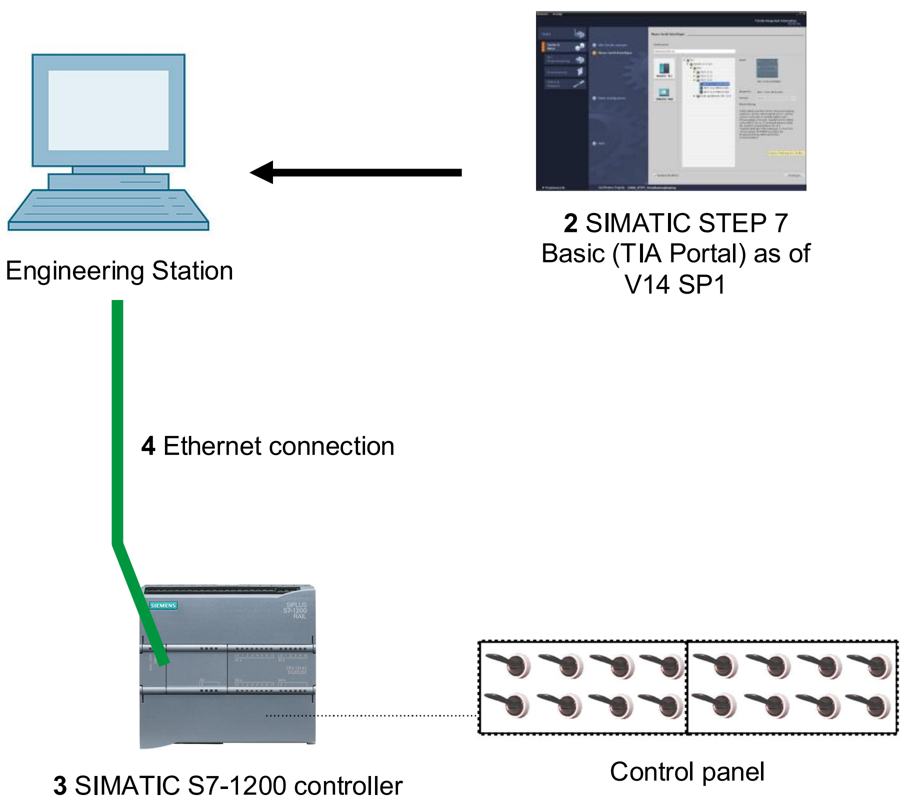
1 Engineering station: requirements include hardware and operating system
(for additional information, see Readme on the TIA Portal Installation DVDs)

2 SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1

Note: The digital inputs should be fed out to a control panel.

4 Ethernet connection between engineering station and controller



4 Theory

4.1 Operating system and application program

Every controller (CPU) contains an ***operating system***, which organizes all functions and sequences of the CPU that are not associated with a specific control task.

The tasks of the operating system include the following:

– Performing a warm restart

– Updating the process image of the inputs and output

– Cyclically calling the user program

– Detecting interrupts and calling interrupt OBs

– Detecting and handling errors

– Managing memory areas

The operating system is an integral component of the CPU and comes pre-installed.

The ***user program*** contains all functions that are necessary for executing your specific automation task. The tasks of the user program include the following:

– Checking the basic requirements for a warm restart using startup OBs

– Processing of process data, i.e. activation of output signals as a function of the input signal states

– Reaction to interrupts and interrupt inputs

– Error handling during normal program execution

4.2 Organization blocks

Organization blocks (OBs) form the interface between the operating system of the controller (CPU) and the application program. They are called from the operating system and control the following operations:

– Cyclic program processing (e.g. OB1)

– Startup characteristics of the controller

– Interrupt-driven program processing

– Error handling

A project must have, at a minimum, ***an organization block for cyclic program processing***. An OB is called by a ***start event*** as shown in Figure 1. In addition, the individual OBs have defined priorities so that, for example, an OB82 for error handling can interrupt the cyclic OB1.

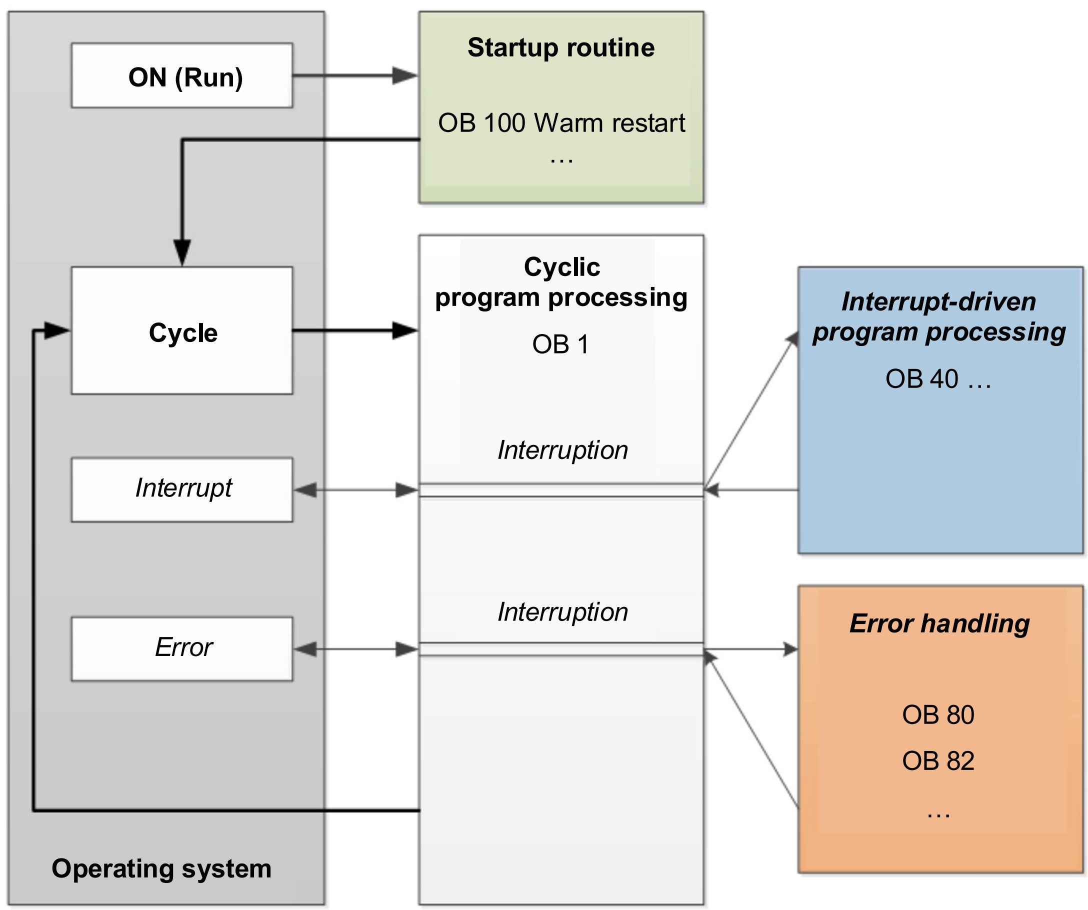


Figure 1: Start events in the operating system and OB calls

When a start event occurs, the following reactions are possible:

– If an OB has been assigned to the event, this event triggers the execution of the assigned OB. If the priority of the assigned OB is greater than the priority of the OB that is currently being executed, it is executed immediately (interrupt). If not, the assigned OB waits until the higher-priority OB has been completely executed

– If you have not assigned an OB to the event, the default system reaction is performed.

Table 1 shows examples for various start events for a SIMATIC S7-1200. Also shown are the possible OB number(s) and the default system reactions that occur when the respective organization block (OB) is not present in the controller.

Start event	**Possible OB numbers**	**Default system reaction**
Startup	100, ≥ 123	Ignore
Cyclic program	1, ≥ 123	Ignore
Time-of-day interrupt	10 to 11	-
Update interrupt	56	Ignore
Scan cycle monitoring time exceeded once	80	Ignore
Scan cycle monitoring time exceeded twice	80	STOP
Diagnostic interrupt	82	Ignore

Table 1: OB numbers for various start events

4.3 Process image and cyclic program processing

When the cyclic user program addresses the inputs (I) and outputs (O), it does not query the signal states directly from the input/output modules. Instead, it accesses a memory area of the CPU. This memory area contains an image of the signal states and is called the **process image**.

6

The cyclic program processing sequence is as follows:

1. At the start of the cyclic program, a query is sent to determine whether or not the individual inputs are energized. This status of the inputs is stored in the **process image of the inputs (PII)**. In doing so, the information 1 or "High" is stored for energized inputs and the information 0 or "Low" for de-energized inputs.
2. The CPU then executes the program stored in the cyclic organization block. For the required input information, the CPU accesses the previously read **process image of the inputs (PII)** and the results of logic operation (RLOs) are written to a so-called **process image of the outputs (PIQ)**.
3. At the end of the cycle, the **process image of the outputs** (**PIQ**) is transferred as the signal state to the output modules and these are energized or de-energized. The sequence then continues again with Item 1.

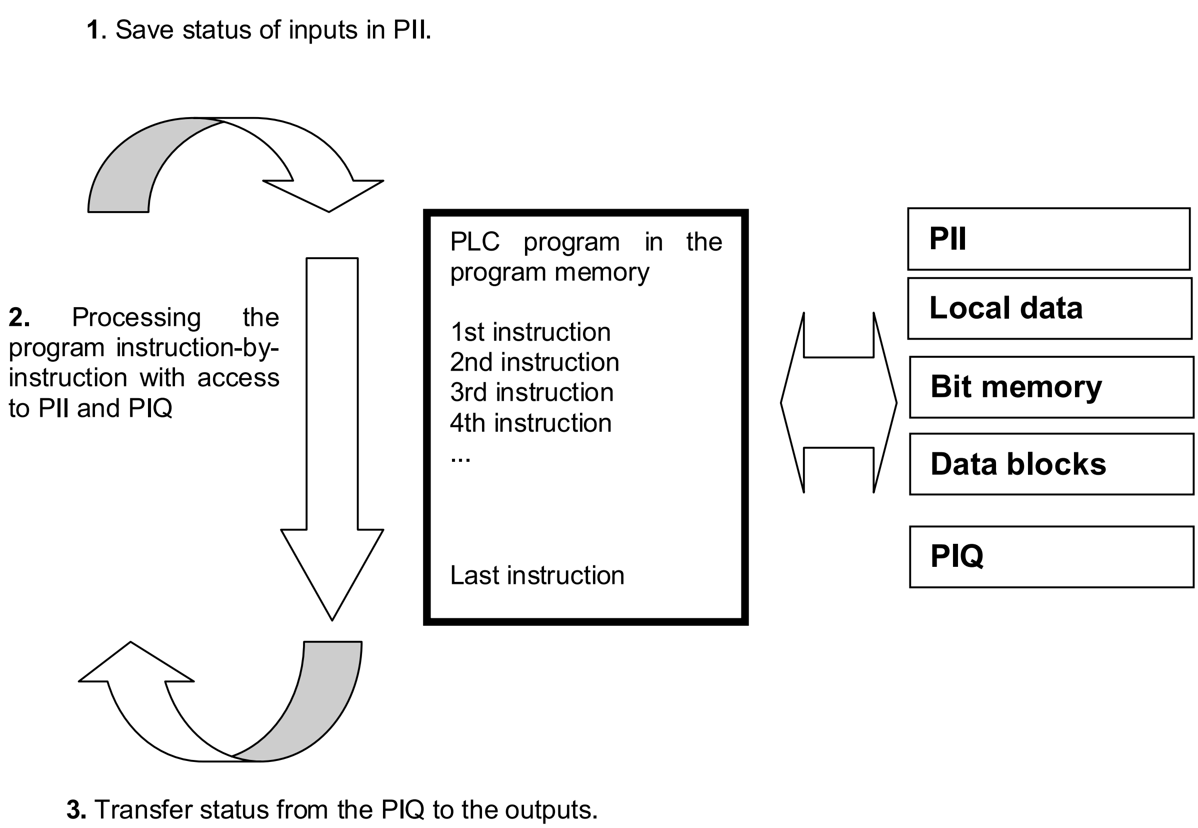


Figure 2: Cyclic program processing

Note: *The time the CPU needs for this sequence is called cycle time. This depends, in turn, on the number and type of instructions and the processor performance of the controller.*

4.4 Functions

Functions (FCs) are logic blocks without memory. They ***have no data memory*** in which values of block parameters can be stored. Therefore, all interface parameters must be connected when a function is called. To store data permanently, global data blocks must be created beforehand.

A function contains a program that is executed whenever the function is called from another logic block.

Functions can be used, for example, for the following purposes:

– Math functions – that return a result dependent on input values.

– Technological functions – such as individual controls with binary logic operations.

A function can also be called several times at different points within a program.

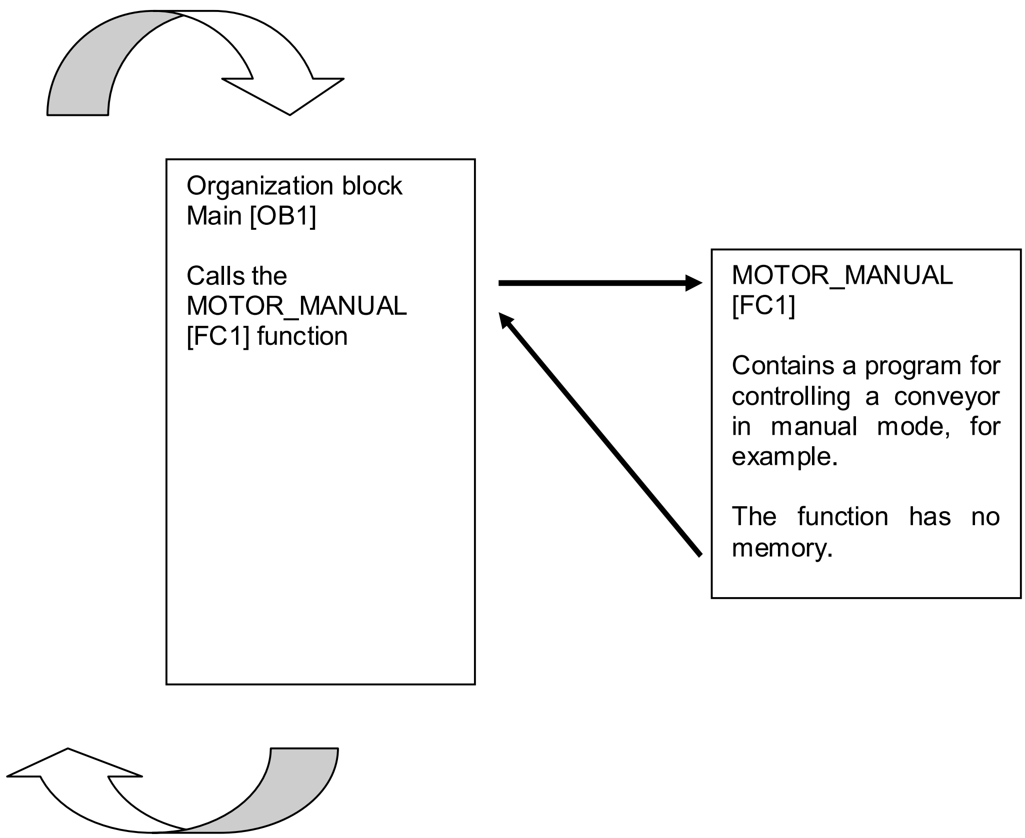


Figure 3: Function with call from organization block Main [OB1]

4.5 Function blocks and instance data blocks

Function blocks are logic blocks that store their input, output and in-out tags as well as static tags permanently in instance data blocks, so that they ***are available after the block has been executed***. For this reason, they are also referred to as blocks with "memory".

Function blocks can also operate with temporary tags. Temporary tags are not stored in the instance DB, however. Instead, they are only available for one cycle.

Function blocks are used for tasks that cannot be implemented with functions:

– Whenever timers and counters are required in the blocks, or

– When information must be stored in the program, such as preselection of the operating mode with a button.

Function blocks are always executed when called from another logic block. A function block can also be called several times at different points within a program. This facilitates the programming of frequently recurring complex functions.

A call of a function block is referred to as an instance. Each instance of a function block is assigned a memory area that contains the data that the function block uses. This memory is made available by data blocks created automatically by the software.

It is also possible to provide memory for multiple instances in one data block in the form of a **multi-instance**. The maximum size of instance data blocks varies depending on the CPU. The tags declared in the function block determine the structure of the instance data block.

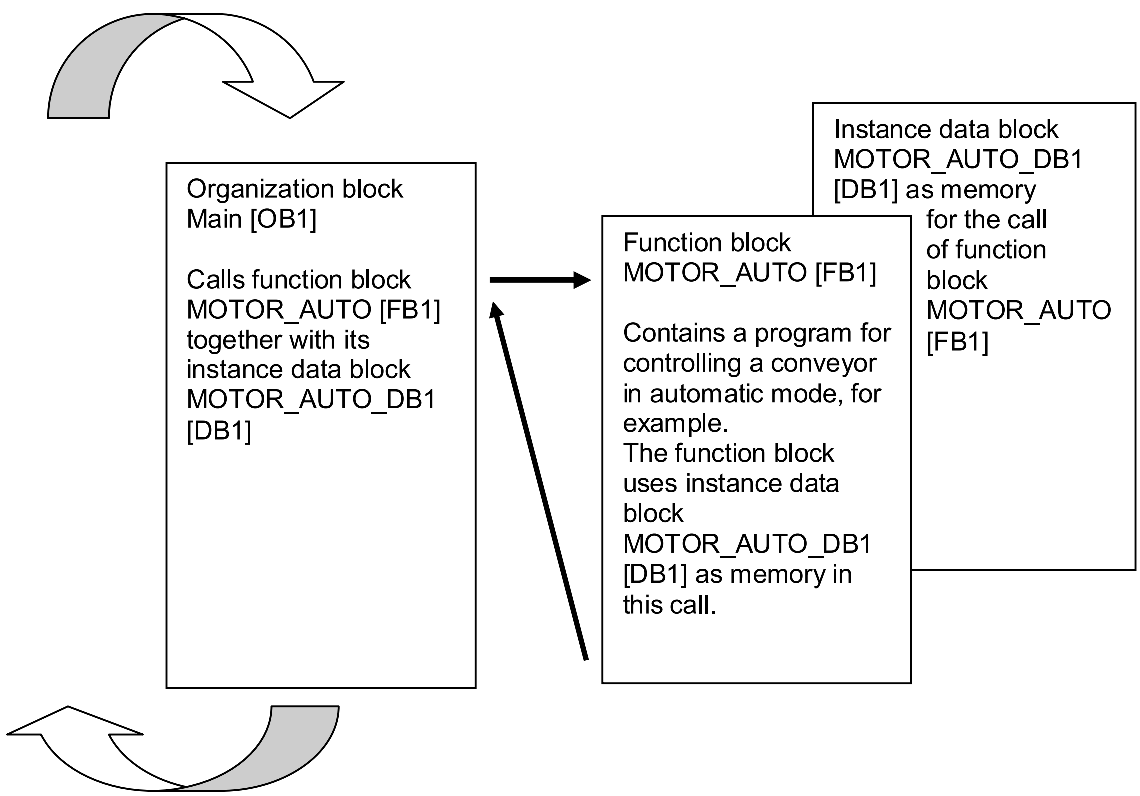


Figure 4: Function block and instance with call from organization block Main [OB1]

4.6 Global data blocks

In contrast to logic blocks, data blocks contain no instructions. Rather, they serve as memory for user data.

Data blocks thus contain variable data that is used by the user program. You can define the structure of global data blocks as required.

Global data blocks store data that can be used ***by all other blocks*** (see Figure 5). Only the associated function block should access instance data blocks. The maximum size of data blocks varies depending on the CPU.

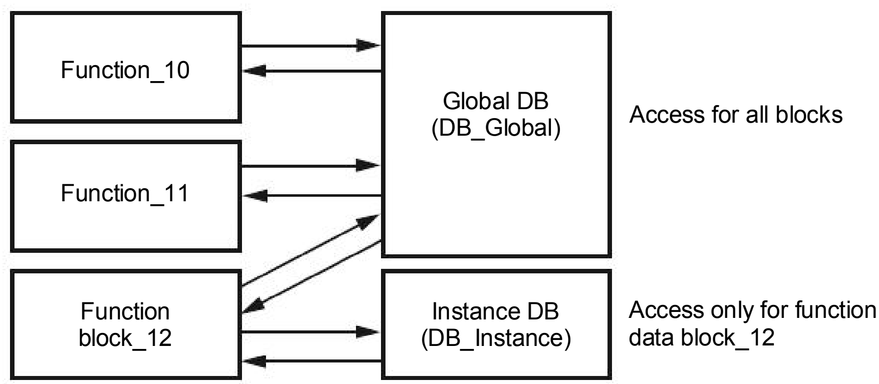


Figure 5: Difference between global DB and instance DB.

Application examples for **global data blocks** are:

– Saving of information about a storage system. "Which product is located where?"

– Saving of recipes for particular products.

6

4.7 Library-compatible logic blocks

A user program can be created with linear or structured programming. ***Linear programming*** writes the entire user program in the cycle OB, but is only suitable for very simple programs for which other less expensive control systems, such as LOGO!, can now be used.

For more complex programs, ***structured programming*** is always recommended. Here, the overall automation task can be broken down into small sub-tasks in order to implement a solution for them in functions and function blocks.

In this case, library-compatible logic blocks should be created preferentially. This means that the input and output parameters of a function or function block are defined generally and only supplied with the current global tags (inputs/outputs) when the block is used.

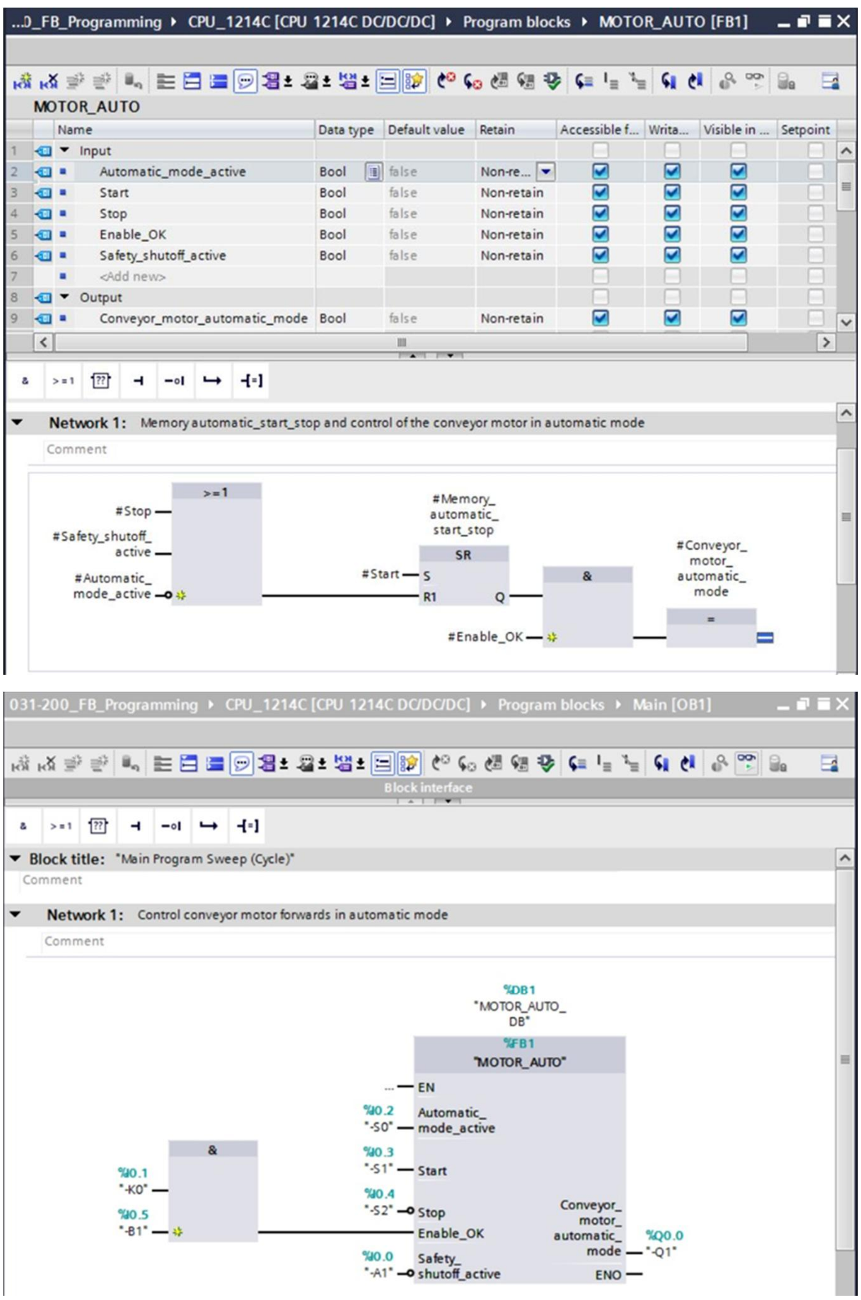


Figure 6: Library-compatible function bock with call in OB1

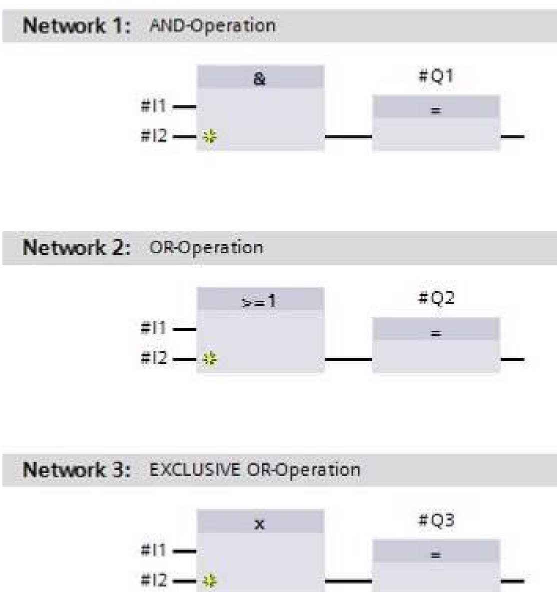
4.8 Programming languages

For SIMATIC S7-1200 controller, the available programming languages for programming functions and function blocks are Function Block Diagram (FBD), Ladder Logic (LAD) and Structured Control Language (SCL).

The ***Function Block Diagram (FBD)*** programming language will be presented in the following.

FBD is a graphical programming language. The representation is based on electronic switching systems. The program is mapped in networks. A network contains one or more logic operation paths. Binary and analog signals are linked together by boxes. The graphical logic symbols known from Boolean algebra are used to represent the binary logic.

You can use binary functions to query binary operands and to logically combine their signal states. The following instructions are examples of binary functions: "AND operation", "OR operation" and "EXCLUSIVE OR operation". These are shown in Figure 7.



#e1	#e2	#a1
0	0	0
0	1	0
1	0	0
1	1	1

#e1	#e2	#a2
0	0	0
0	1	1
1	0	1
1	1	1

#e1	#e2	#a3
0	0	0
0	1	1
1	0	1
1	1	0

Figure 7: Binary functions in FBD and associated logic table

You can thus use simple instructions, for example, to control binary outputs, evaluate edges and execute jump functions in the program.

Program elements such as IEC timers and IEC counters provide complex instructions.

The empty box serves as a placeholder in which you can select the required instruction.

Enable input EN (enable)/ Enable output ENO (enable output) mechanism:

– An instruction without EN/ENO mechanism is executed independent of the signal state at the box inputs.

– Instructions with EN/ENO mechanism are only executed if enable input "EN input has signal state "1". When the box is processed correctly, enable output "ENO" has signal state "1". If an error occurs during processing, the enable output "ENO" is reset. If enable input EN is not connected, the box is always executed.

6

5 Task

The following functions of the sorting station process description will be planned, programmed and tested in this chapter:

– Automatic mode - Conveyor motor

6 Planning

The programming of all functions in OB1 is not recommended for reasons of clarity and reusability. The majority of the program code will therefore be moved into functions (FCs) and function blocks (FBs). The decision on which functions are to be moved to the FB and which is to run in OB 1 is planned below.

6.1 EMERGENCY STOP

The EMERGENCY STOP does not require a separate function. Just like the operating mode, the current state of the EMERGENCY STOP relay can be used directly at the blocks.

6.2 Automatic mode - Conveyor motor

Automatic mode of the conveyor motor is to be encapsulated in a function block (FB) "MOTOR_AUTO". On the one hand, this preserves the clarity of OB1. On the other hand, it enables reuse if another conveyor belt is added to the station. Table 2 lists the planned parameters.

Input	Data	Comment
Automatic_mode_active	BOOL	Automatic mode activated
Start	BOOL	Pushbutton automatic start
Stop	BOOL	Pushbutton automatic stop
Enable_OK	BOOL	All enable conditions OK
Safety_shutoff_active	BOOL	Safety shutoff active e.g. emergency stop
Output		
Conveyor_motor_automatic_mode	BOOL	Control of the conveyor motor in automatic mode
Static		
Memory_automatic_start_stop	BOOL	Memory used for start/stop automatic mode

Table 2: Parameters for FB "MOTOR_AUTO"

The Memory_automatic_start_stop is latched with Start but only if the reset conditions are not present.

The Memory_automatic_start_stop is reset if Stop is present or safety shutoff is active or automatic mode is not activated (manual mode).

The Conveyor_motor_automatic_mode output is controlled when Memory_automatic_start_stop is set and the enable conditions are met.

6.3 Technology diagram

Here, you see the technology diagram for the task.

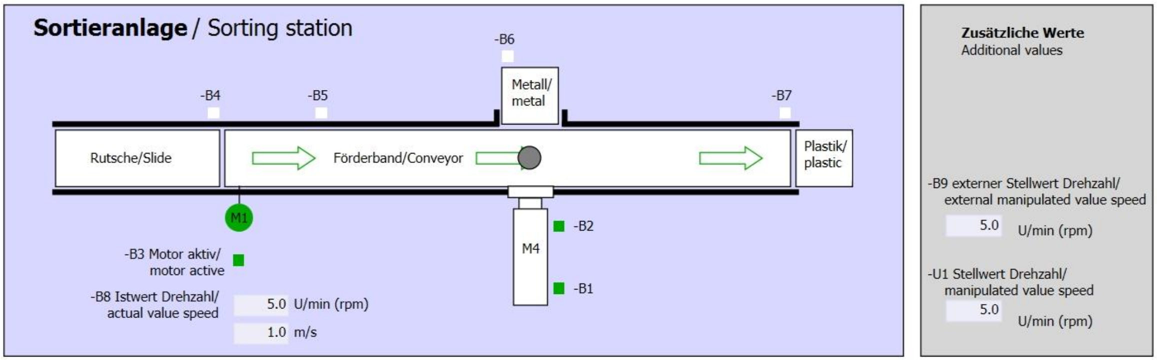


Figure 8: Technology diagram

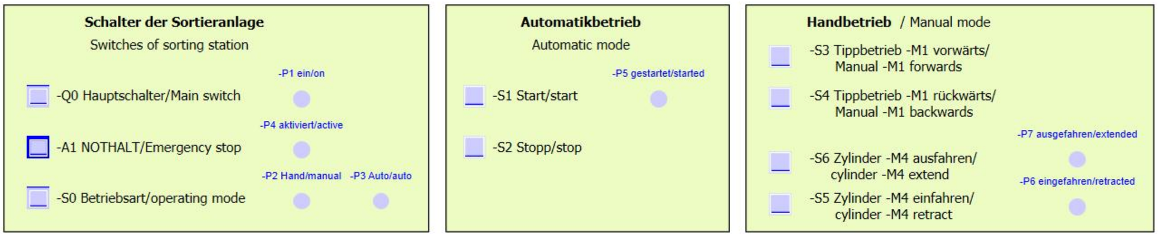


Figure 9: Control panel

6

6.4 Reference list

The following signals are needed as operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop ok	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	-S1	Pushbutton automatic start	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder M4 retracted	NO

DO	Type	Identifier	Function	
Q 0.0	BOOL	-Q1	Conveyor motor M1 forwards fixed speed	

Legend for reference list

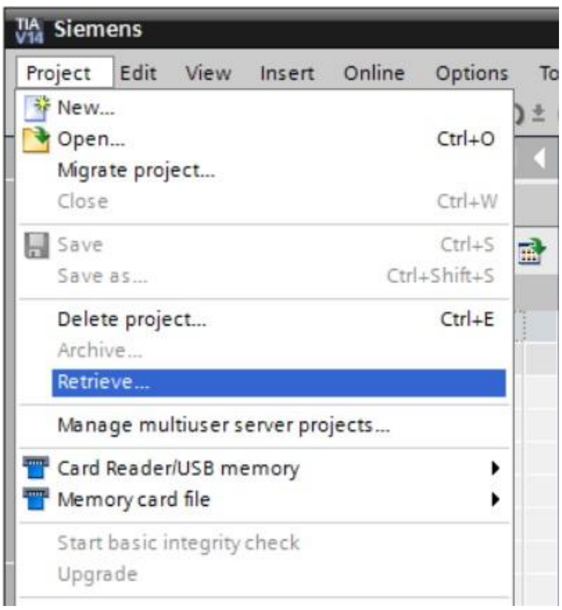
DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

7 Structured step-by-step instructions

You can find instructions on how to carry out planning below. If you already have a good understanding of everything, it will be sufficient to focus on the numbered steps. Otherwise, simply follow the detailed steps in the instructions.

7.1 Retrieve an existing project

→ Before we can start programming the function block (FB) "MOTOR_AUTO", we need a project with a hardware configuration (e.g. SCE_EN_011_101_Hardware_Configuration_S7-CPU1214C....zap). To retrieve an existing project that has been archived, you must select the relevant archive with → Project → Retrieve in the project view. Confirm your selection with Open (→ Project → Retrieve → Select a .zap archive → Open).

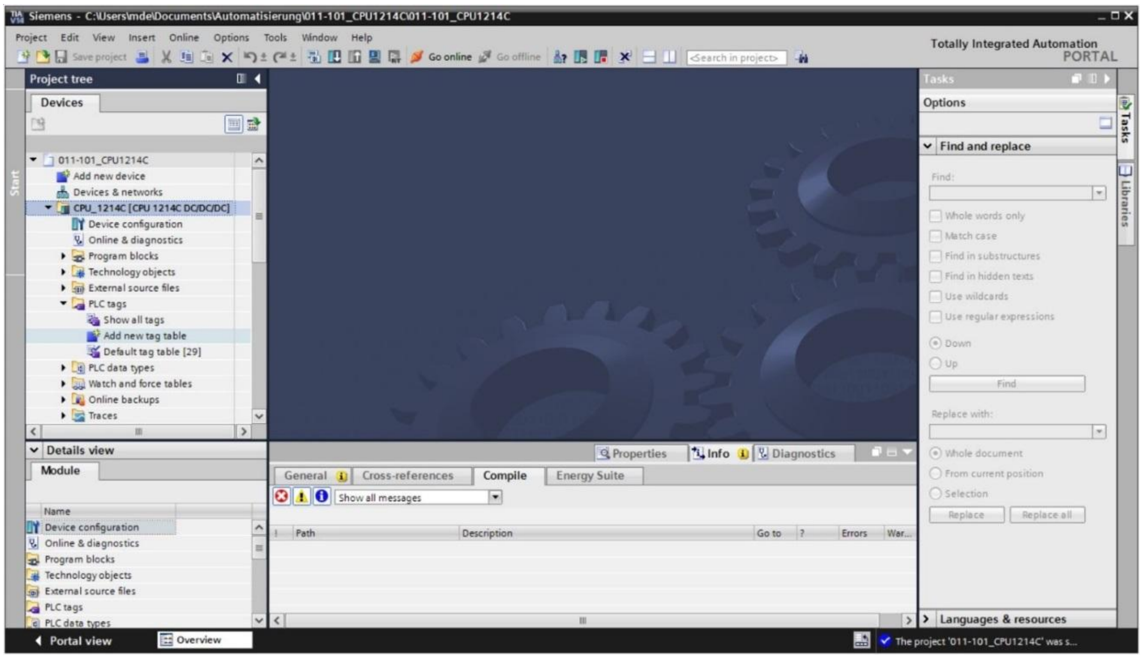


6

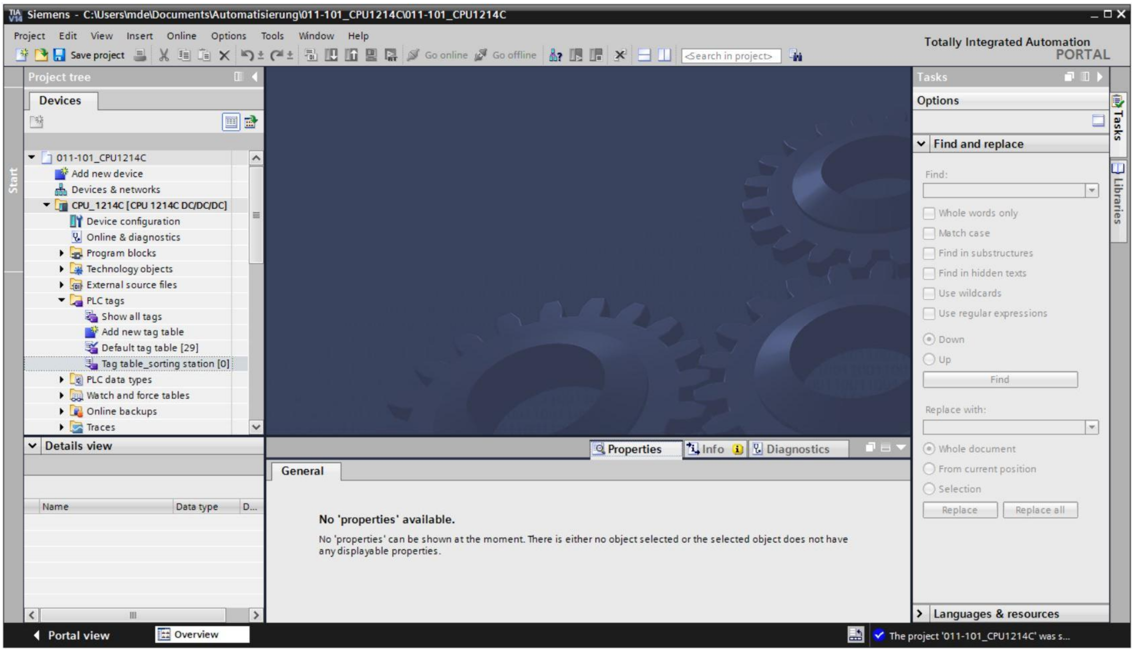
→ The next step is to select the target directory where the retrieved project will be stored. Confirm your selection with "OK". (→ Target directory → OK)

7.2 Create a new tag table

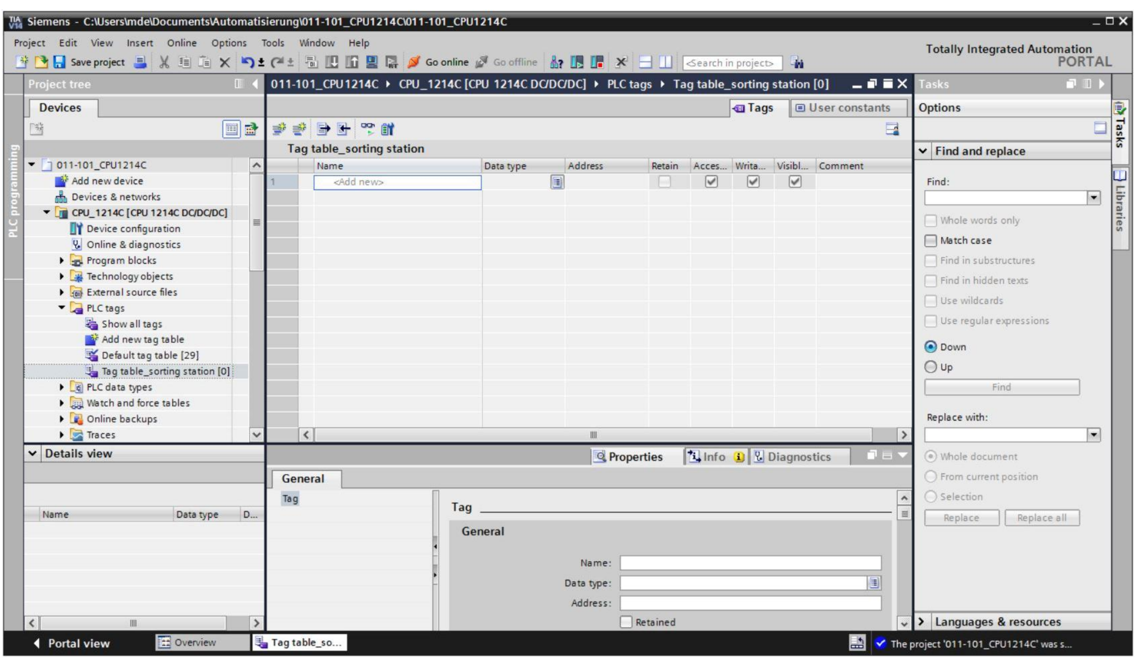
→ In the project view, navigate to the → PLC tags of your controller and create a new tag table by double-clicking → Add new tag table.



→ Rename the tag table you just created as "Tag_table_sorting_station" (→ right-click "Tag_table_1" → "Rename" → Tag_table_sorting_station).

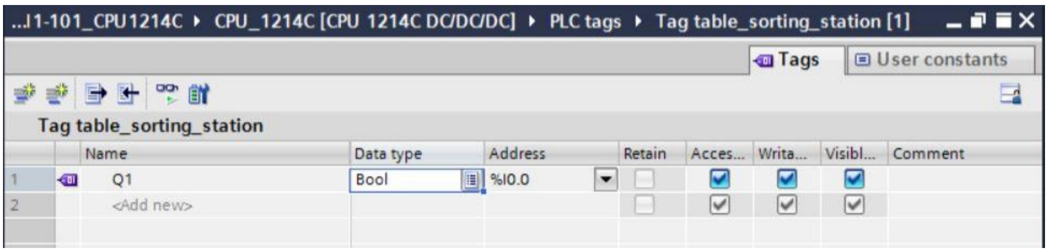


→ Open this tag table with a double-click. (→ Tag_table_sorting_station)

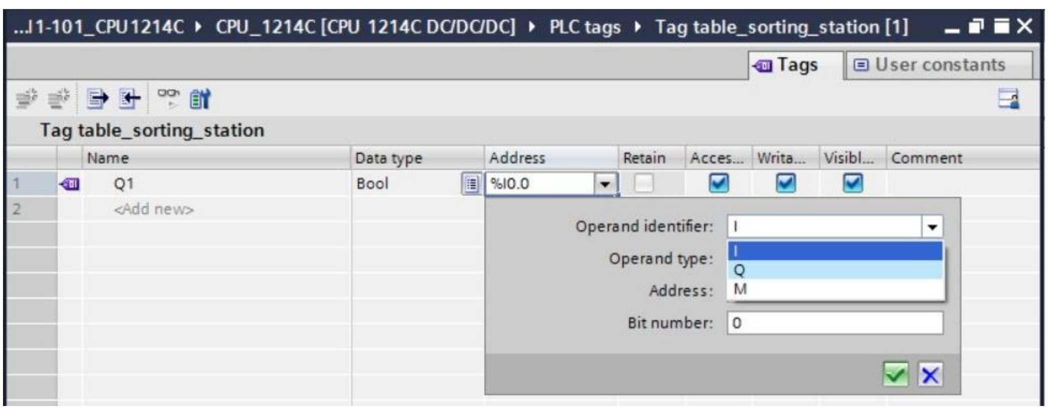


7.3 Create new tags within a tag table

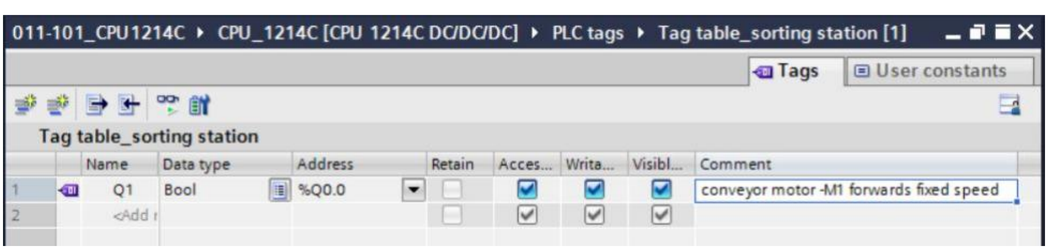
→ Add the name Q1 and confirm the entry with the Enter key. If you have not yet created additional tags, TIA Portal now automatically assigns data type "Bool" and address %I0.0 (I 0.0) (→ <Add> → Q1 → Enter).



→ Change the address to %Q0.0 (Q 0.0) by entering this directly or by clicking the drop-down arrow to open the Addressing menu. Change the operand identifier to Q and confirm with Enter or by clicking the check mark (→ %I0.0 → Operand identifier→ Q → ✓)



→ Enter the "Conveyor motor M1 forwards fixed speed" comment for the tag.



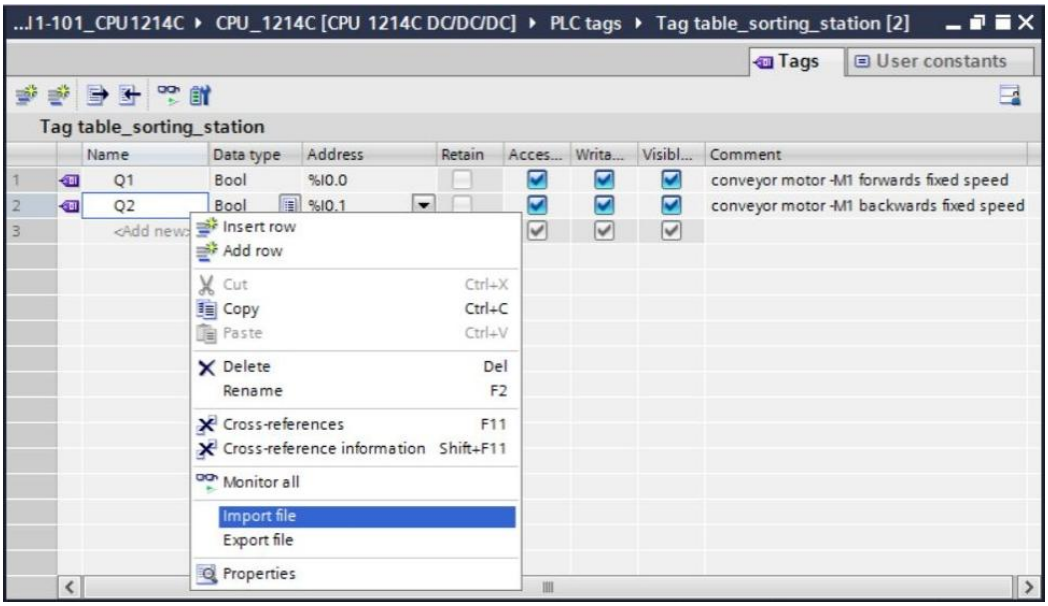
→ Add a new Q2 tag in line 2. TIA Portal has automatically assigned the same data type as the one in line 1 and has incremented the address by 1 to %Q0.1 (Q0.1). Enter the comment "Conveyor motor M1 backwards fixed speed".

(→ <Add> → Q2 → Enter → Comment → Conveyor motor M1 backwards fixed speed)

7.4 Import "Tag_table_sorting_station"

→ To insert an existing symbol table, right-click on an empty field of the created "Tag_table_sorting_station". Select "Import file" in the shortcut menu.

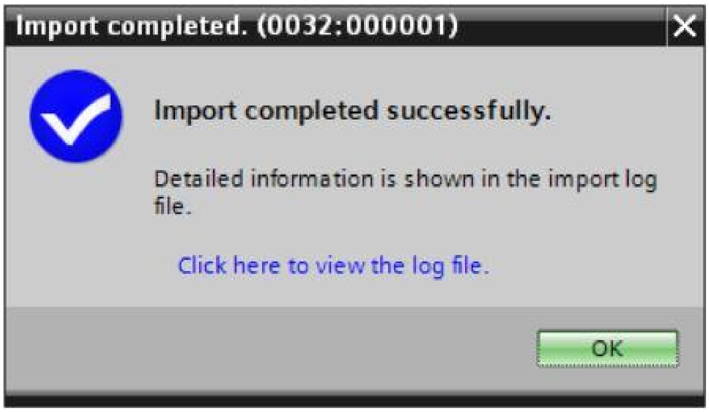
(→ Right-click in an empty field of the tag table → Import file)



6

→ Select the desired symbol table (e.g. in .xlsx format) and confirm the selection with "Open". (→ SCE_EN_020-100_Tag_table_sorting_station… → Open)

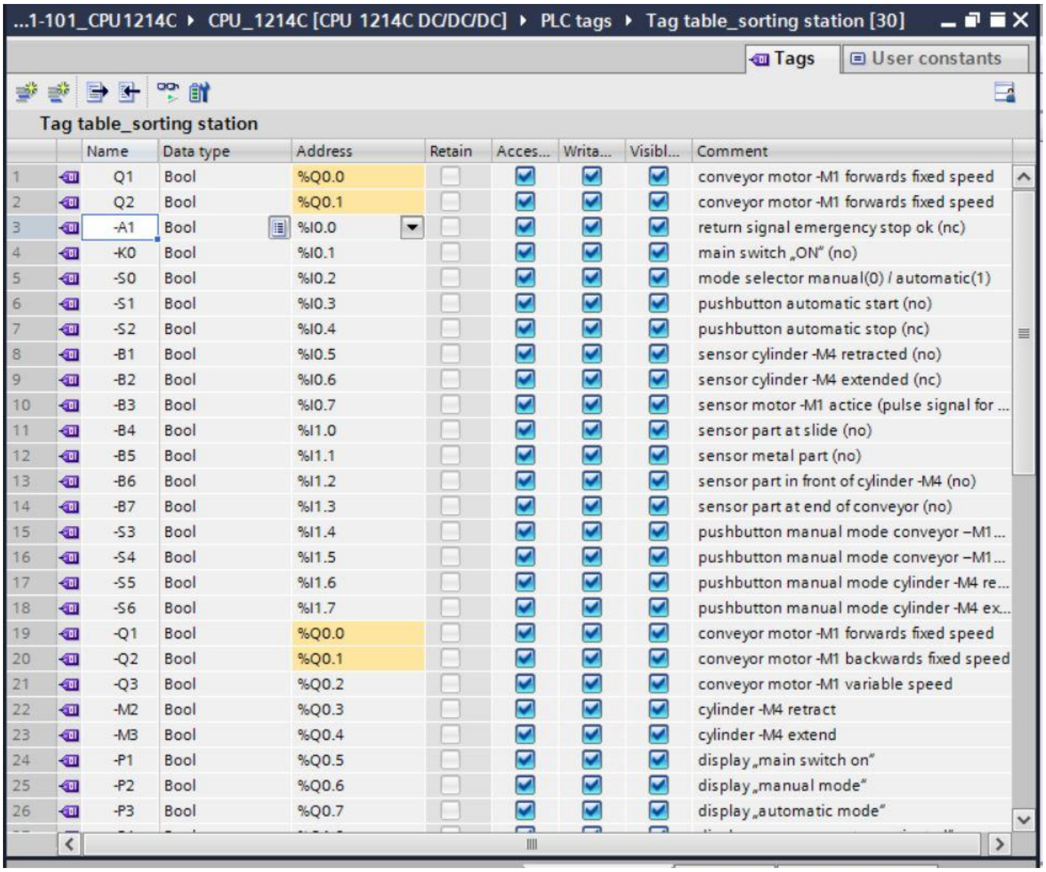
→ When the import is finished, you will see a confirmation window and have an opportunity to view the log file for the import. Click → OK.



You can see that some addresses have been highlighted in orange. These are duplicate addresses and the names of the associated tags have been numbered automatically to avoid confusion.

→ Delete the duplicate tags by selecting the lines and pressing the Del key on your keyboard or selecting "Delete" in the shortcut menu.

(→ Right-click on selected tags → Delete)



...1-101_CPU1214C ▸ CPU_1214C [CPU 1214C DC/DC/DC] ▸ PLC tags ▸ Tag table_sorting station [30]

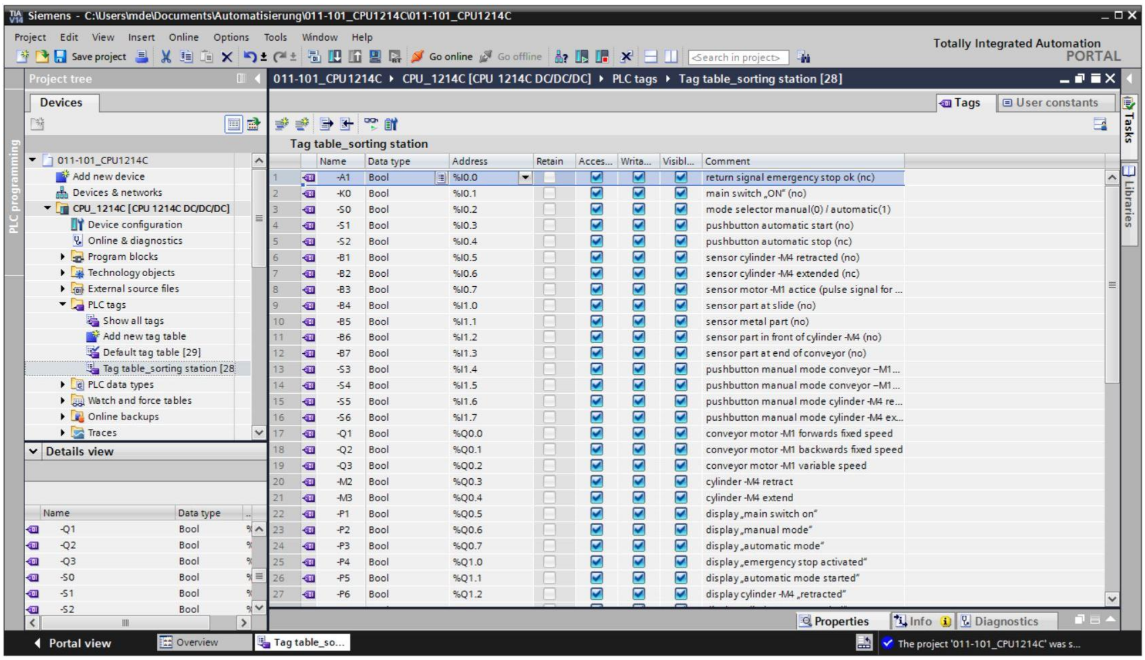
Tags | User constants

Tag table_sorting station

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	Q1	Bool	%Q0.0	☐	☑	☑	☑	conveyor motor -M1 forwards fixed speed
2	Q2	Bool	%Q0.1	☐	☑	☑	☑	conveyor motor -M1 forwards fixed speed
3	-A1	Bool	%I0.0	☐	☑	☑	☑	return signal emergency stop ok (nc)
4	-K0	Bool	%I0.1	☐	☑	☑	☑	main switch „ON" (no)
5	-S0	Bool	%I0.2	☐	☑	☑	☑	mode selector manual(0) / automatic(1)
6	-S1	Bool	%I0.3	☐	☑	☑	☑	pushbutton automatic start (no)
7	-S2	Bool	%I0.4	☐	☑	☑	☑	pushbutton automatic stop (nc)
8	-B1	Bool	%I0.5	☐	☑	☑	☑	sensor cylinder -M4 retracted (no)
9	-B2	Bool	%I0.6	☐	☑	☑	☑	sensor cylinder -M4 extended (nc)
10	-B3	Bool	%I0.7	☐	☑	☑	☑	sensor motor -M1 actice (pulse signal for ...
11	-B4	Bool	%I1.0	☐	☑	☑	☑	sensor part at slide (no)
12	-B5	Bool	%I1.1	☐	☑	☑	☑	sensor metal part (no)
13	-B6	Bool	%I1.2	☐	☑	☑	☑	sensor part in front of cylinder -M4 (no)
14	-B7	Bool	%I1.3	☐	☑	☑	☑	sensor part at end of conveyor (no)
15	-S3	Bool	%I1.4	☐	☑	☑	☑	pushbutton manual mode conveyor –M1 ...
16	-S4	Bool	%I1.5	☐	☑	☑	☑	pushbutton manual mode conveyor –M1 ...
17	-S5	Bool	%I1.6	☐	☑	☑	☑	pushbutton manual mode cylinder -M4 re...
18	-S6	Bool	%I1.7	☐	☑	☑	☑	pushbutton manual mode cylinder -M4 ex...
19	-Q1	Bool	%Q0.0	☐	☑	☑	☑	conveyor motor -M1 forwards fixed speed
20	-Q2	Bool	%Q0.1	☐	☑	☑	☑	conveyor motor -M1 backwards fixed speed
21	-Q3	Bool	%Q0.2	☐	☑	☑	☑	conveyor motor -M1 variable speed
22	-M2	Bool	%Q0.3	☐	☑	☑	☑	cylinder -M4 retract
23	-M3	Bool	%Q0.4	☐	☑	☑	☑	cylinder -M4 extend
24	-P1	Bool	%Q0.5	☐	☑	☑	☑	display „main switch on"
25	-P2	Bool	%Q0.6	☐	☑	☑	☑	display „manual mode"
26	-P3	Bool	%Q0.7	☐	☑	☑	☑	display „automatic mode"

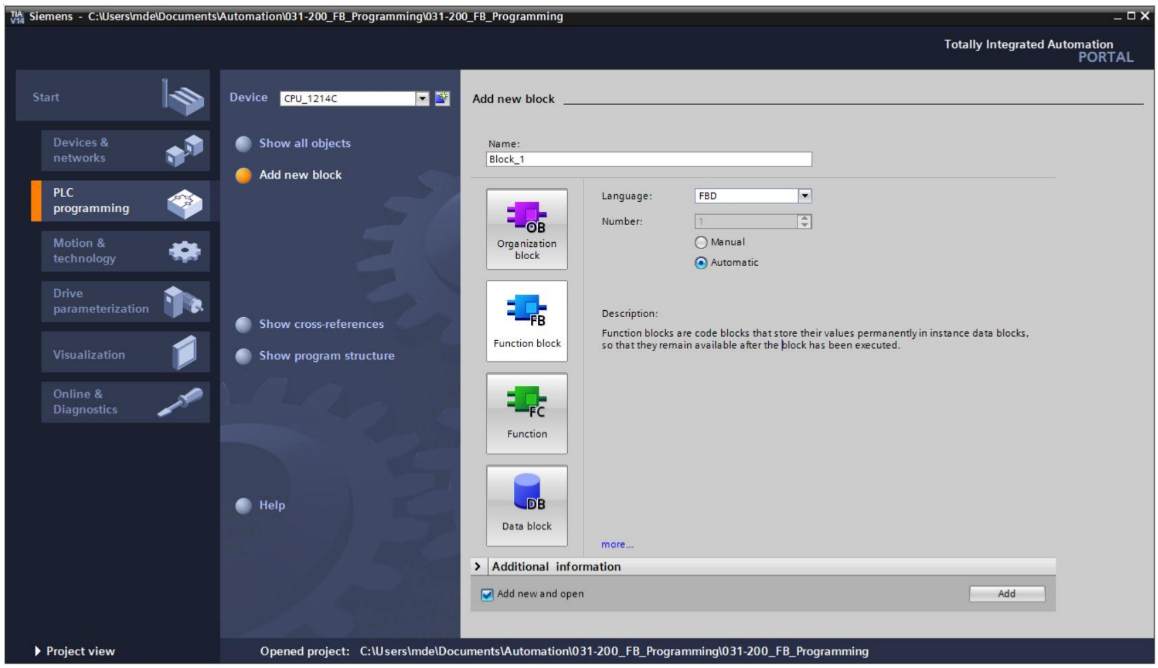
→ You now have a complete symbol table of the digital inputs and outputs in front of you. Save your project under the name 031-200_FB_Programming.

(→ Project → Save as ... → 031-200_FB_Programming → Save)

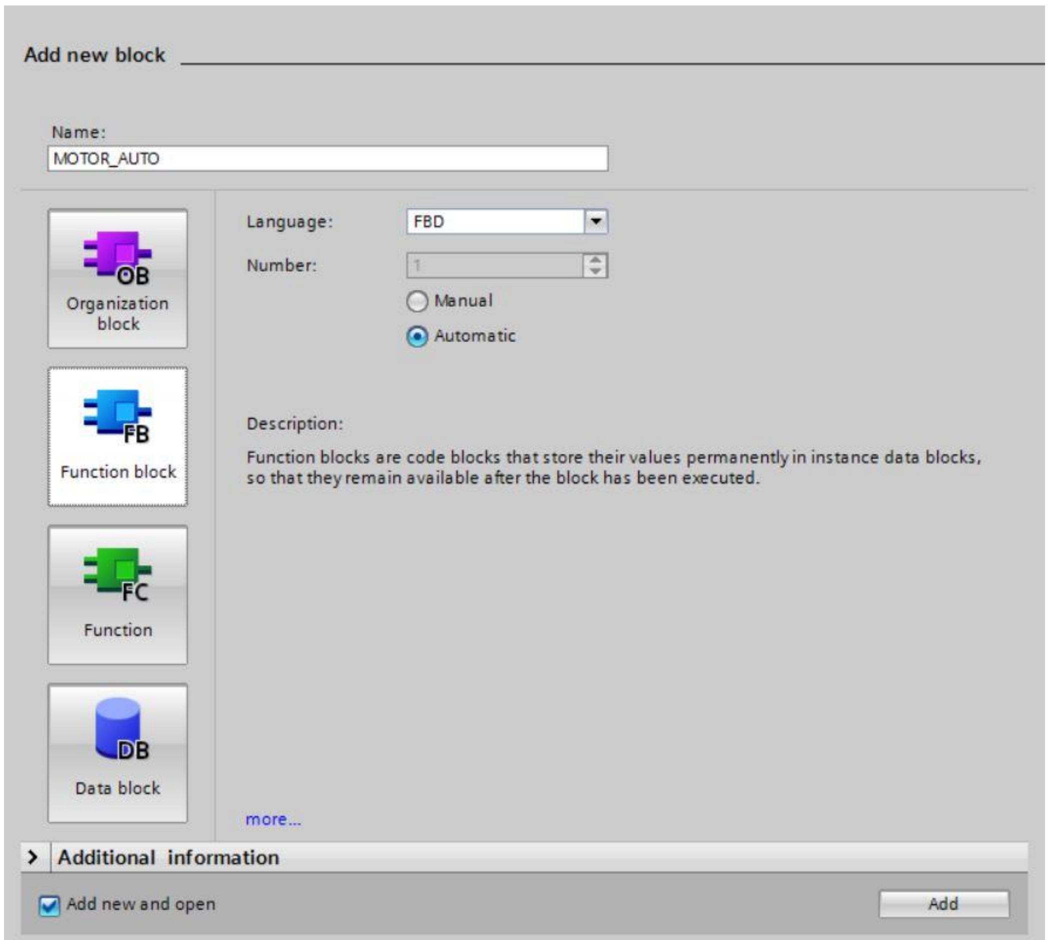


7.5 Create function block FB1 "MOTOR_AUTO" for the conveyor motor in automatic mode

→ In the PLC programming section of the portal view, click "Add new block" to create a new function block. (→ PLC programming → Add new block → )



→ Rename your new block to: "MOTOR_AUTO", set the language to FBD and keep automatic assignment of the number. Select the "Add new and open" check box. You will thus be taken automatically to your created function block in the project view.Click "Add". (→ Name: MOTOR_AUTO→Language: FBD → Number: Automatic → ☑ Add new and open → Add)

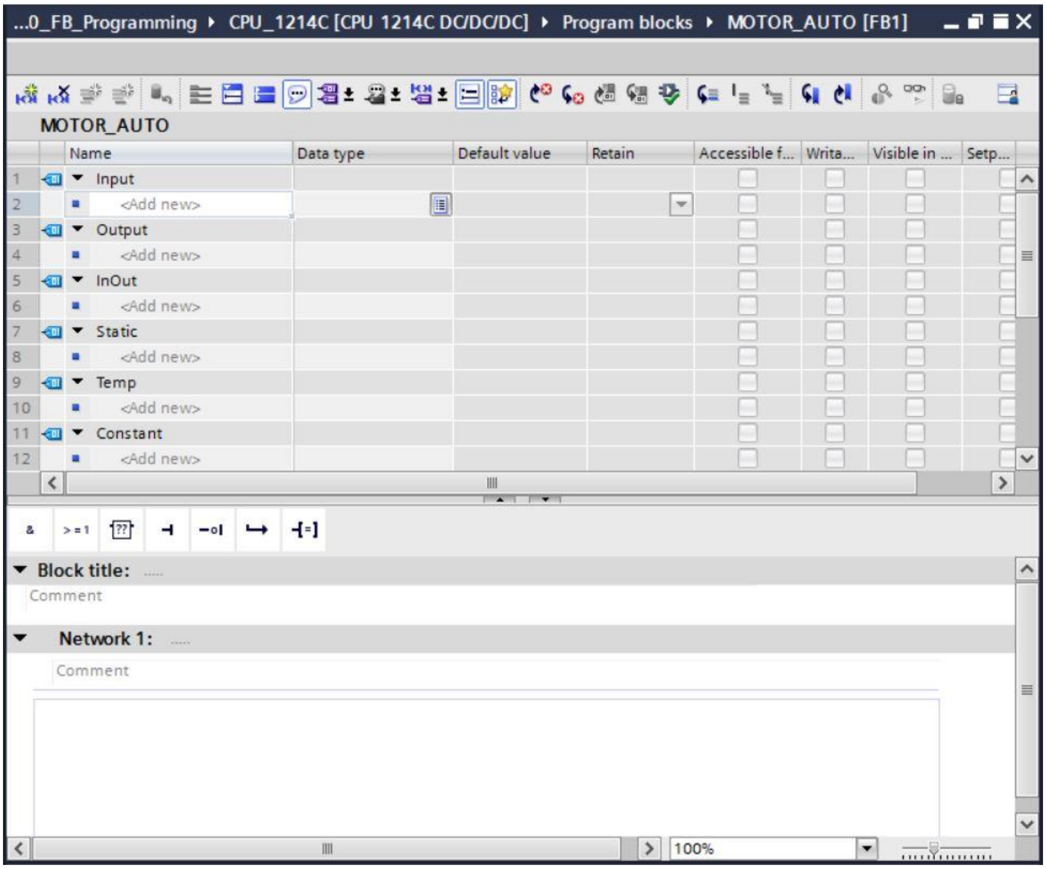


6

7.6 Define the interface of FB1 "MOTOR_AUTO"

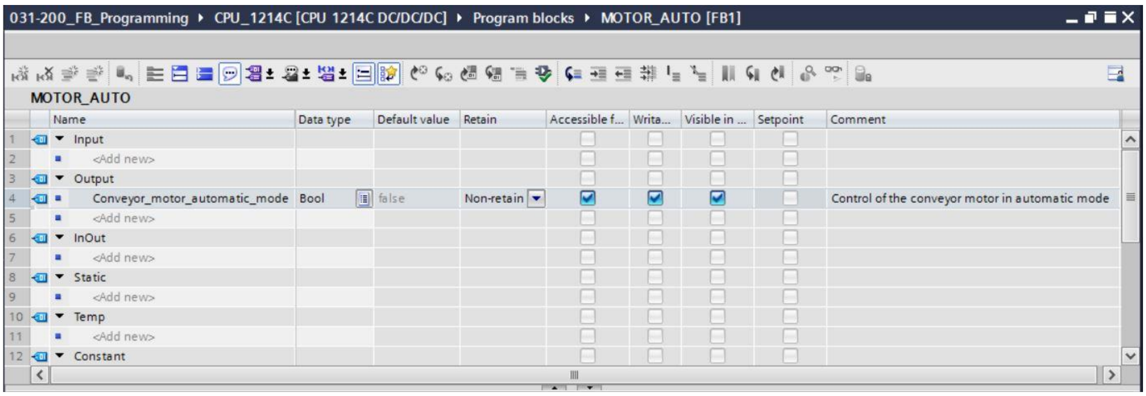
→ If you selected "Add new and open", the project view opens with a window for creating the block you just added.

→ You can find the interface description of your function block in the upper section of your programming view.



6

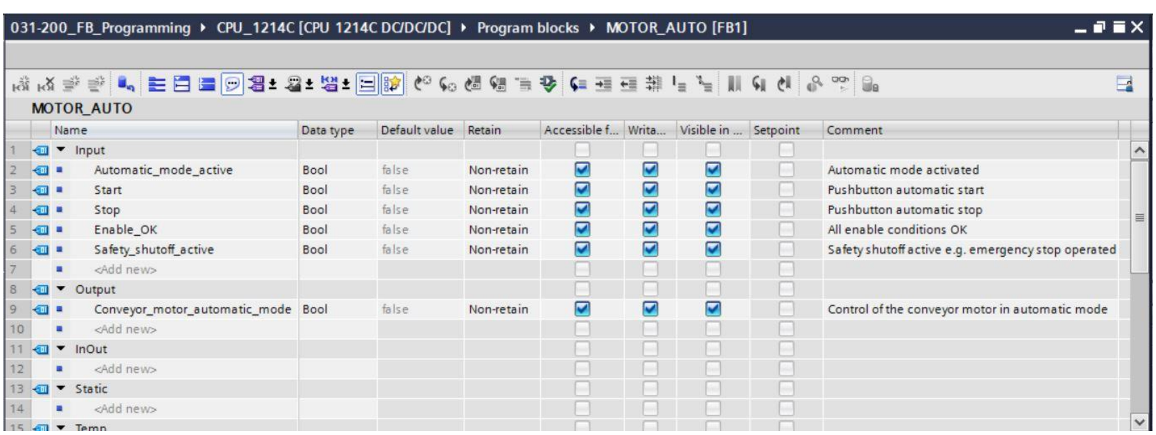
→ A binary output signal is needed for controlling the conveyor motor. For this reason, we first create local output tag #Conveyor_motor_automatic_mode of the "Bool" type. Enter the comment "Control of the conveyor motor in automatic mode" for the parameter. (→ Output→ Conveyor_motor_automatic_mode → Bool → Control of the conveyor motor in automatic mode)



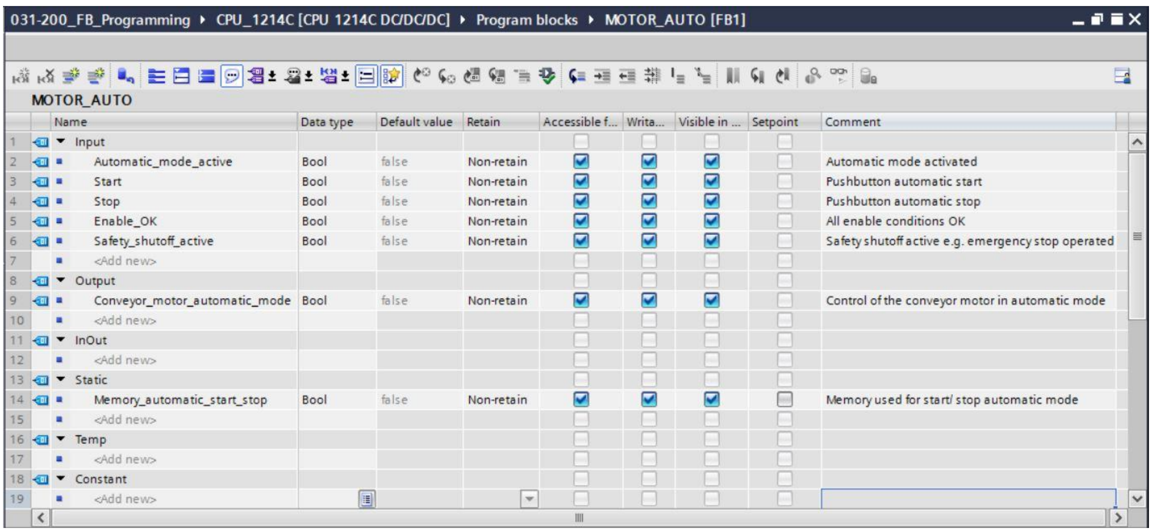
→ Add parameter #Automatic_mode_active as the input interface under Input and confirm the entry with the Enter key or by exiting the entry field. Data type "Bool" is assigned automatically. This will be retained. Next, enter the associated comment "Automatic mode activated".

(→ Input → Automatic_mode_active → Bool → Automatic mode activated)

→ Continue by adding parameters #Start, #Stop, #Enable_OK and #Safety_shutoff_active as additional binary input parameters and check their data types. Add descriptive comments.

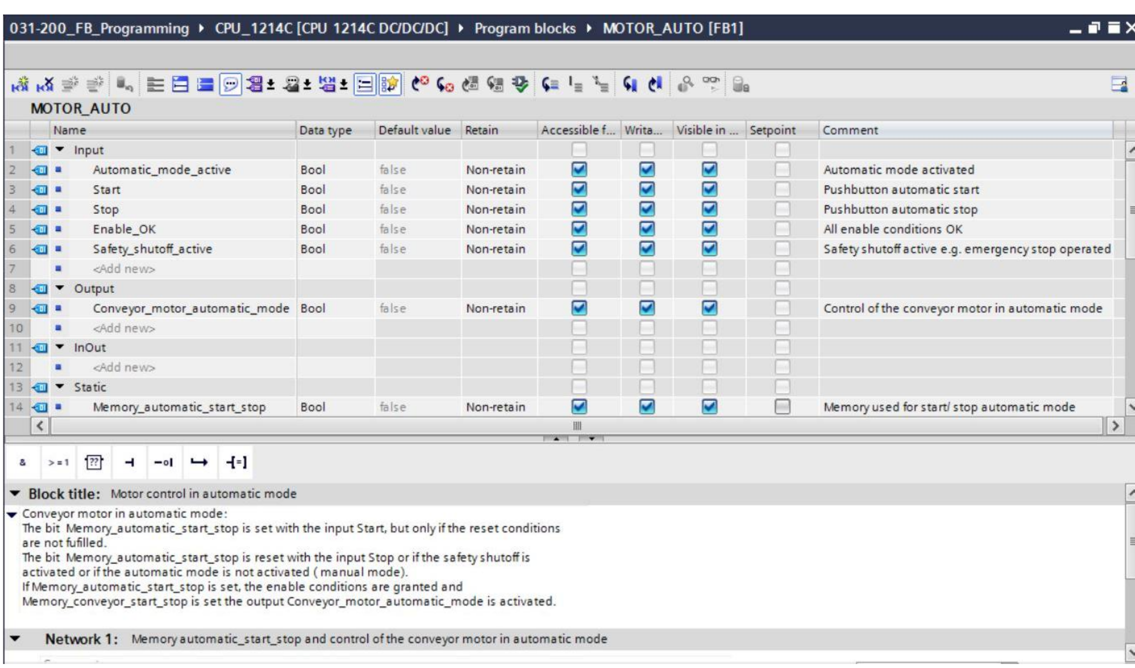


→ The conveyor is started and stopped with pushbuttons. We therefore need a "Static" tag as a memory. Under Static, add tag #Memory_automatic_start_stop and confirm the entry with the Enter key or by exiting the entry field. Data type "Bool" is assigned automatically. This will be retained. Enter the associated comment "Memory used for start_stop automatic mode". (→ Static → Memory_automatic_start_stop → Bool → Memory usedfor start/stop automatic mode)



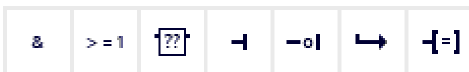
→ For purposes of program documentation, assign the block title, a block comment and a helpful network title for Network 1.

(→ Block title: Motor control in automatic mode → Network 1: Memory_automatic_start_stop and control of the conveyor motor in automatic mode)



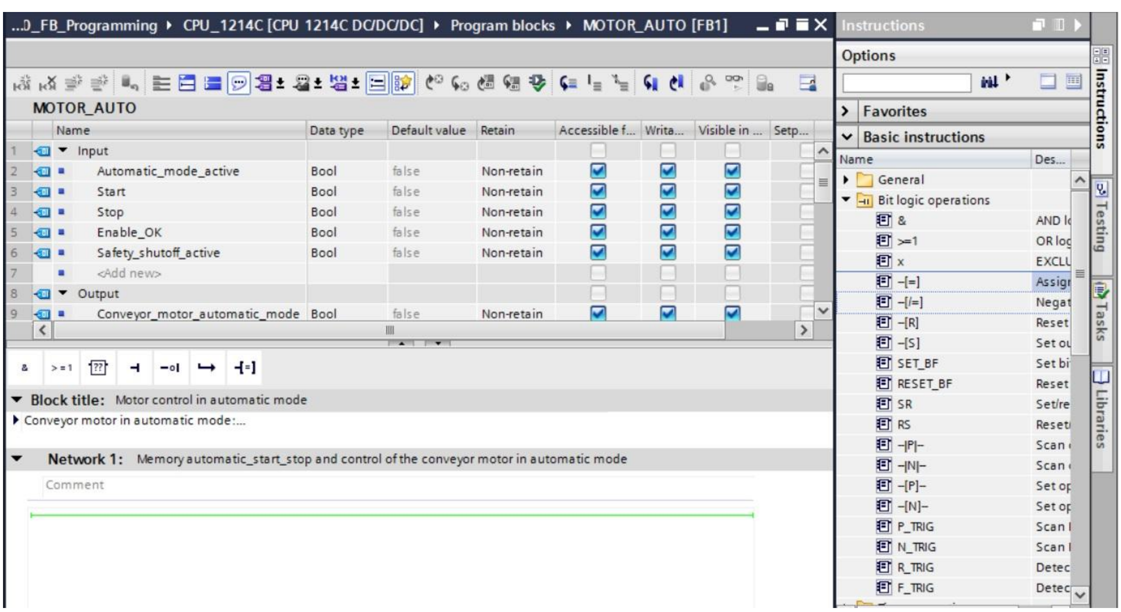
7.7 Program FB1: MOTOR_AUTO

→ Below the interface description, you see a toolbar in the programming window with various logic functions and below that an area with networks. We have already specified the block title and the title for the first network there. Programming is performed within the networks using individual logic blocks. Distribution among multiple networks helps to preserve the clarity of the program. In the following, you will get to know the various ways you can insert logic blocks.

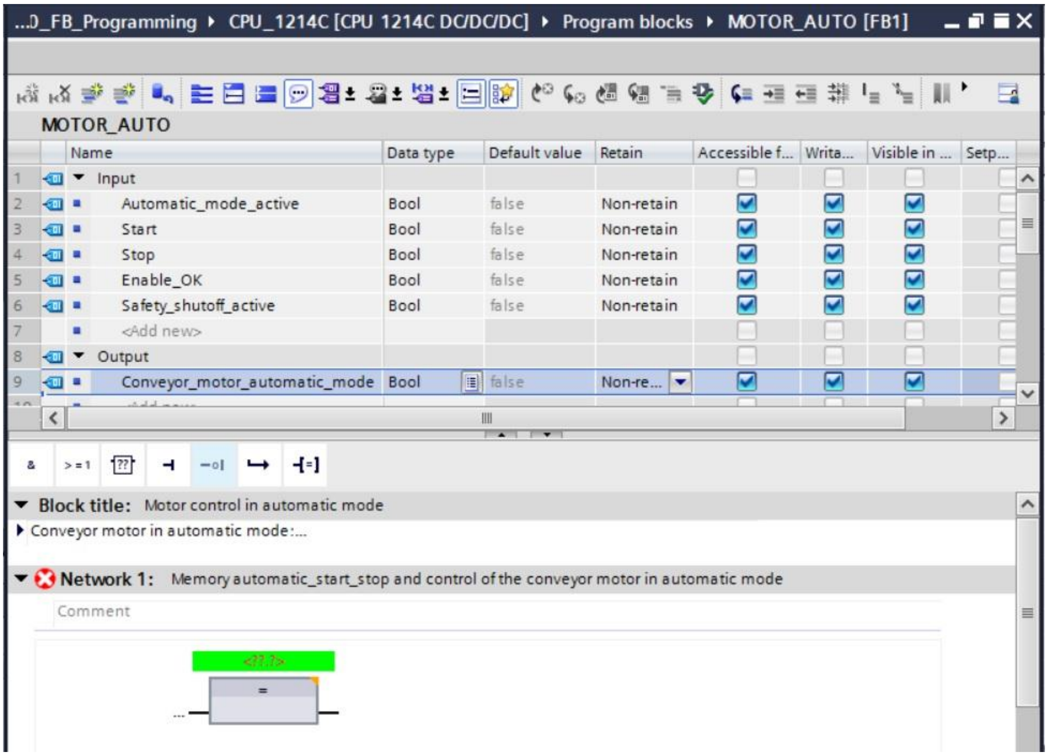


→ On the right side of your programming window is a list of instructions you can use in the program. Under → Basic instructions → Bit logic operations, find function --[=] (Assignment) and use a drag & drop operation to move it to Network 1 (green line appears, mouse pointer with + symbol).

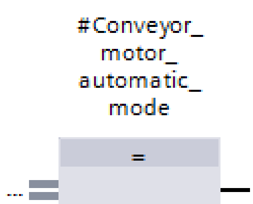
(→ Instructions → Basic instructions → Bit logic operations → --[=])



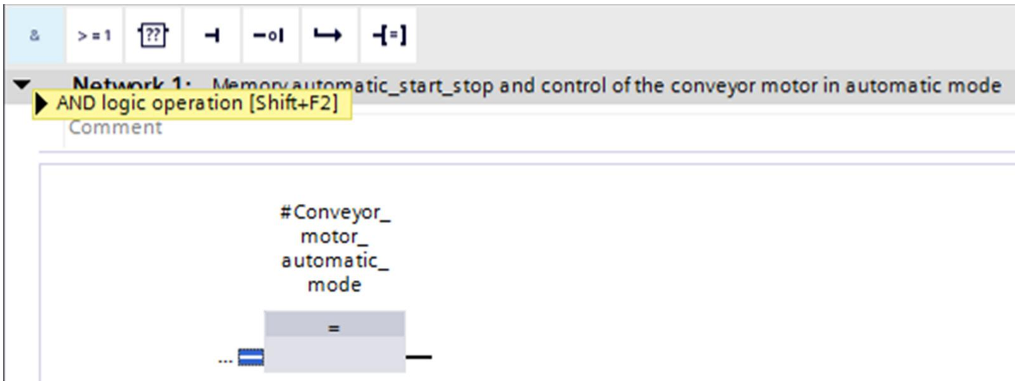
Now use drag & drop to move your output parameter #Conveyor_motor_automatic_mode onto <??.?> above the block you just inserted. The best way to select a parameter in the interface description is by "grabbing" it at the blue symbol . (→ Conveyor_motor_automatic_mode)



→ This determines that the #Conveyor_motor_automatic_mode parameter is written by this block. Still missing, however, are the input conditions so that this actually happens. An SR flip-flop and #Enable_OK parameter are logically combined with an AND logic operation at the input of the assignment block. To do this, first click the input of the block so that the input line has a blue background.

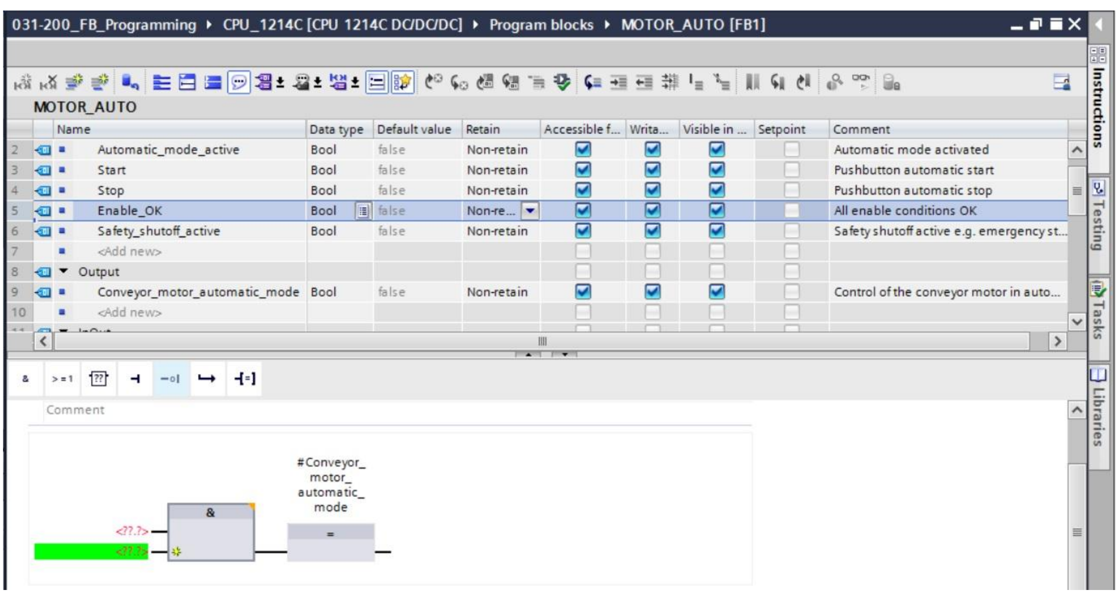


→ Click the & icon in your logic toolbar to insert an AND logic operation before your assignment block.



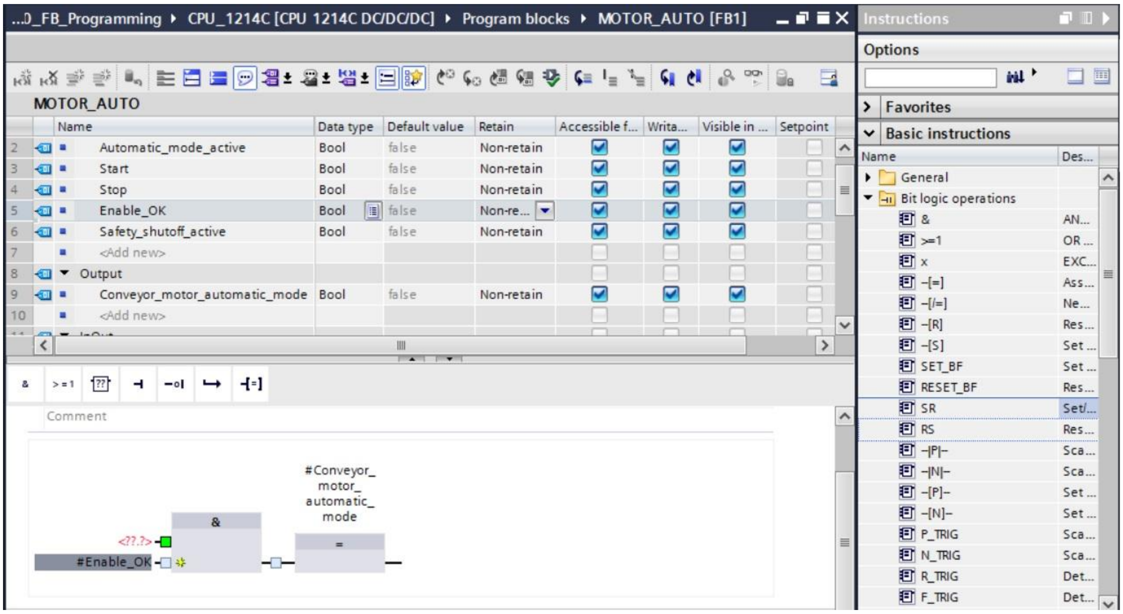
6

→ Use drag & drop to move input parameter #Enable_OK onto the second input of the & logic operation <??.?>. (→ Enable_OK)



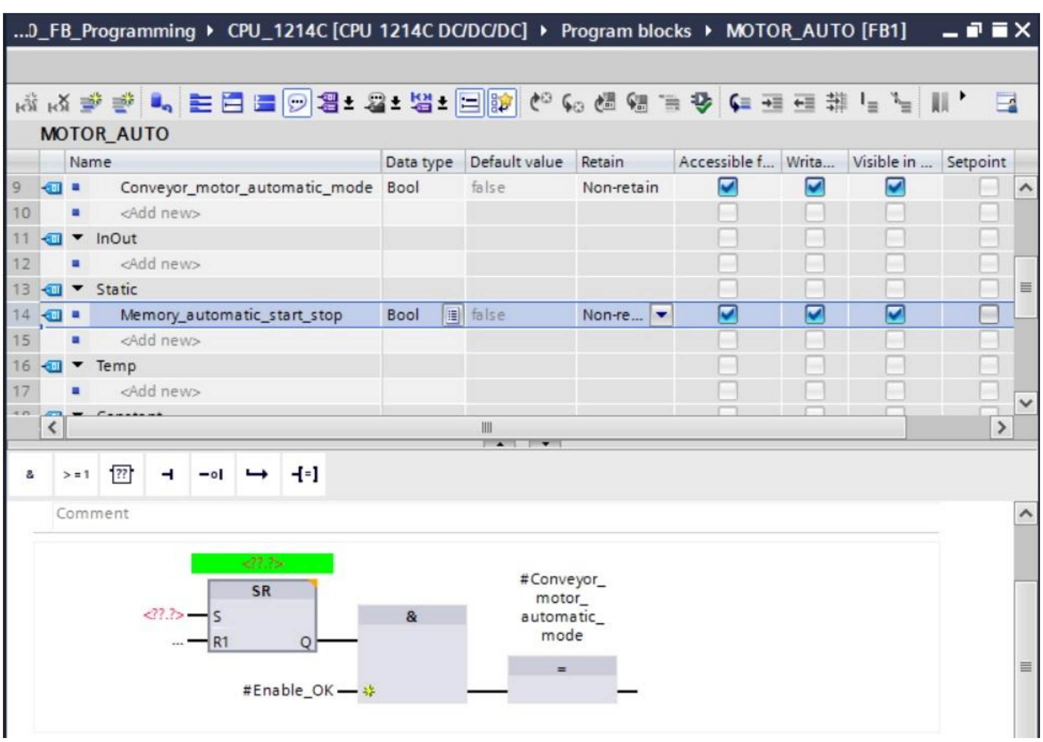
→ Use drag & drop to move the Set/reset flip-flop function SR from the list of instructions under → Basic instructions → Bit logic operations onto the first input of the & operation.

(→ Instructions → Basic instructions → Bit logic operations → SR →)



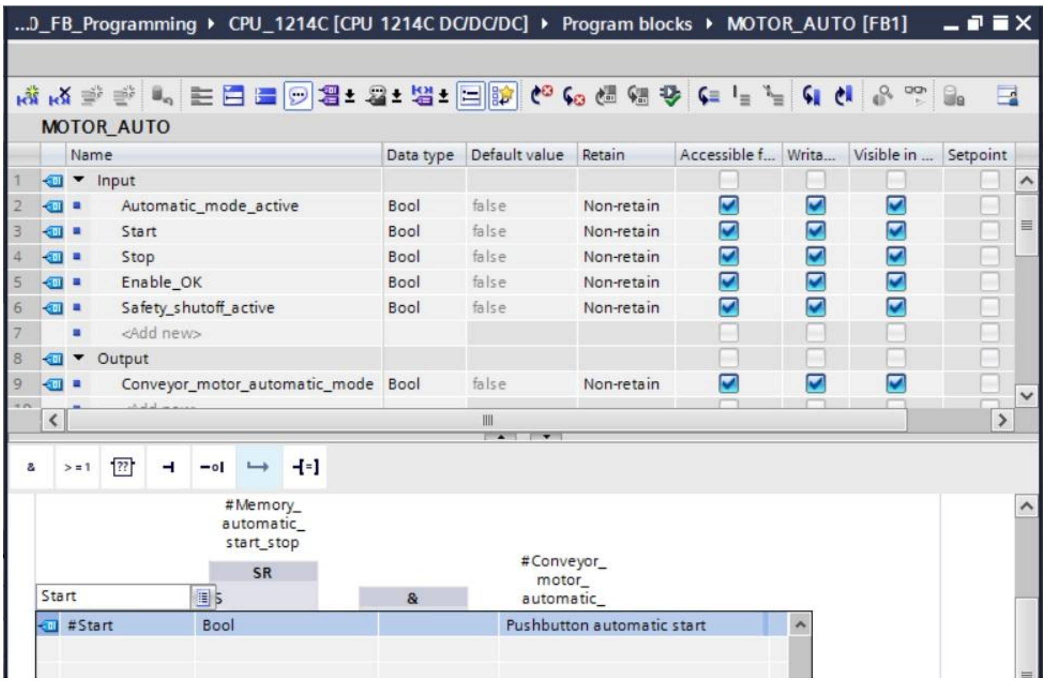
→ The SR flip-flop requires a memory tag. For this, use drag & drop to move static parameter #Memory_automatic_start_stop onto the <??.?> above the SR flip-flop.

(→ Memory_automatic_start_stop)



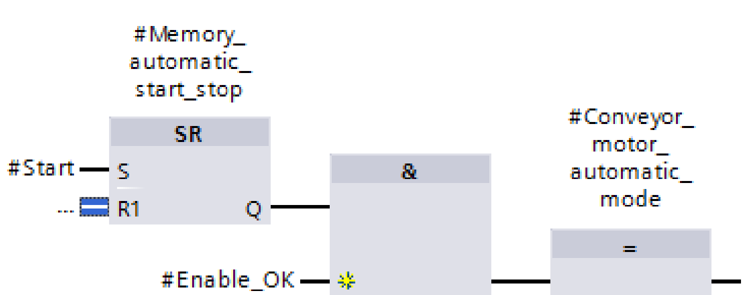
6

→ The #Memory_automatic_start_stop will be set with input tag #Start. Click twice on the S input of the SR flip-flop <??.?> and enter "Start" in the field that appears in order to see a list of available tags starting with "Start".Click the #Start tag and apply with → Enter. (→ SR flip-flop → <??.?> → Start → #Start → Enter)



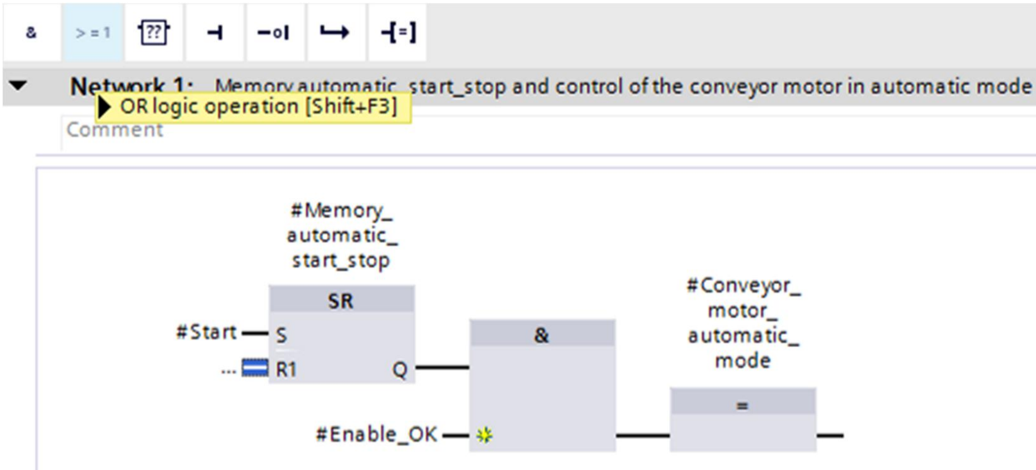
Note: *When assigning tags in this way, there is a risk of a mix-up with the global tags from the tag table. The previously presented procedure using drag & drop from the interface description should therefore be used preferentially.*

→ Multiple conditions are to be able to stop the conveyor. An OR block is therefore needed at the R1 input of the SR flip-flop. First, click the R1 input of the SR flip-flop so that the input line has a blue background.

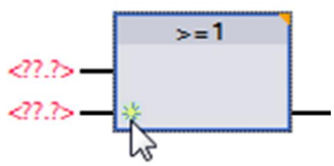


6

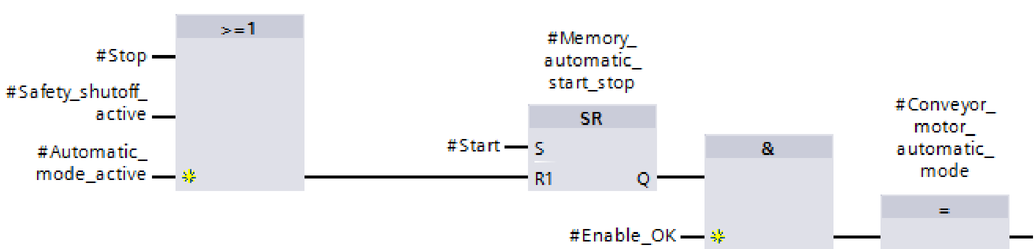
→ Click the icon in your logic toolbar to insert an OR logic operation.



→ The OR block has 2 inputs initially. In order to logically combine an additional input tag, click the yellow star of your OR block.

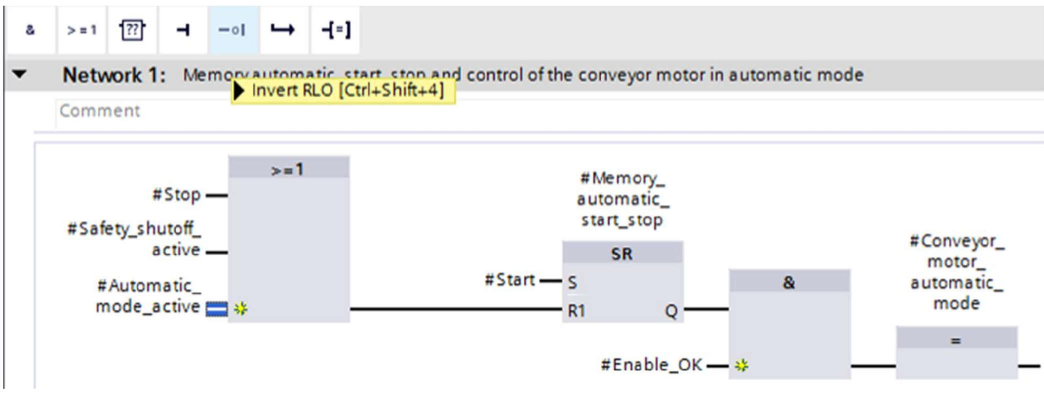


→ Add input tags #Stop, #Safety_shutoff_active and #Automatic_mode_active to the 3 inputs of the OR block.



6

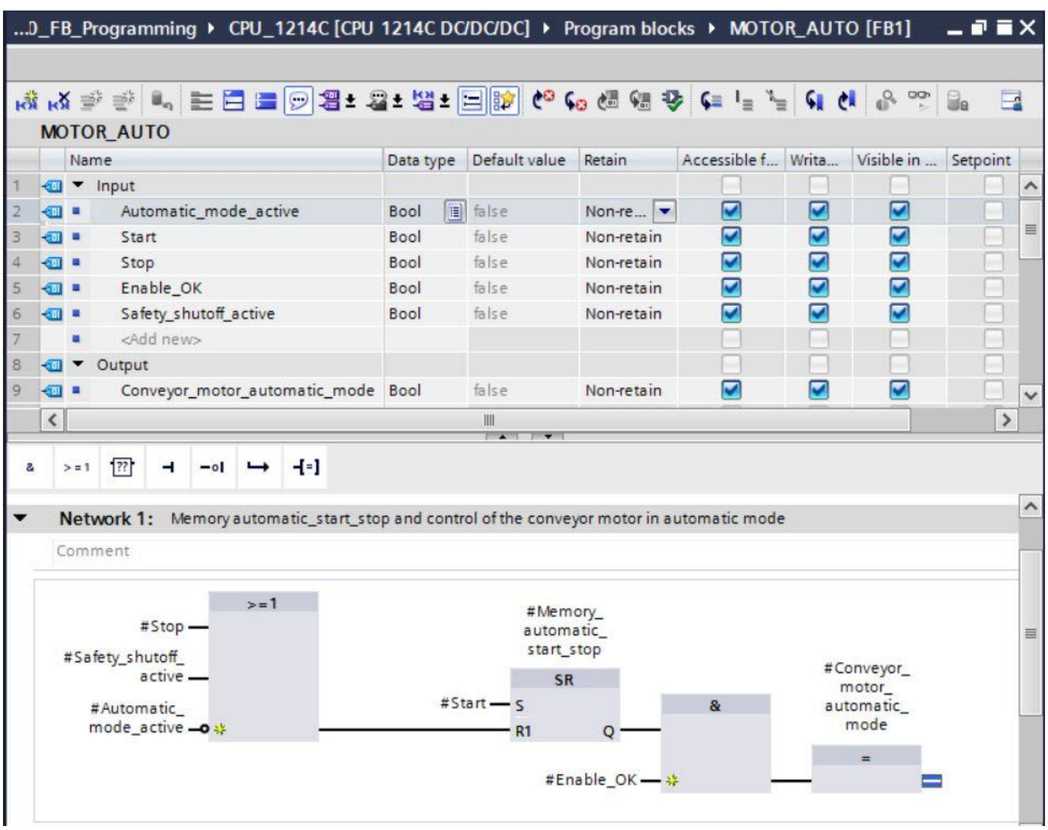
→ Negate the input connected to parameter #Automatic_mode_active by selecting it and clicking .



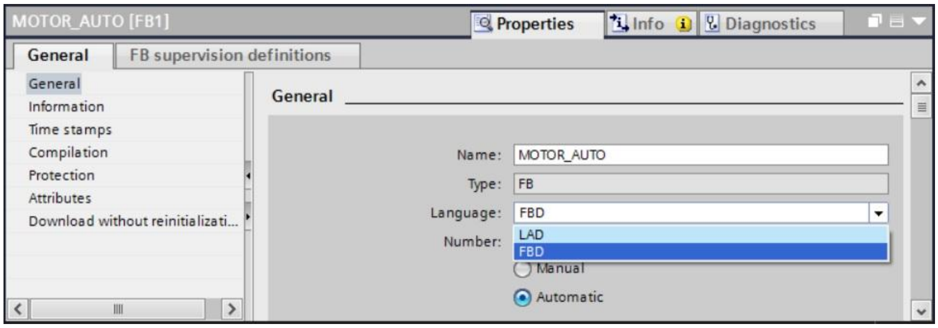
6

→ Do not forget to click 

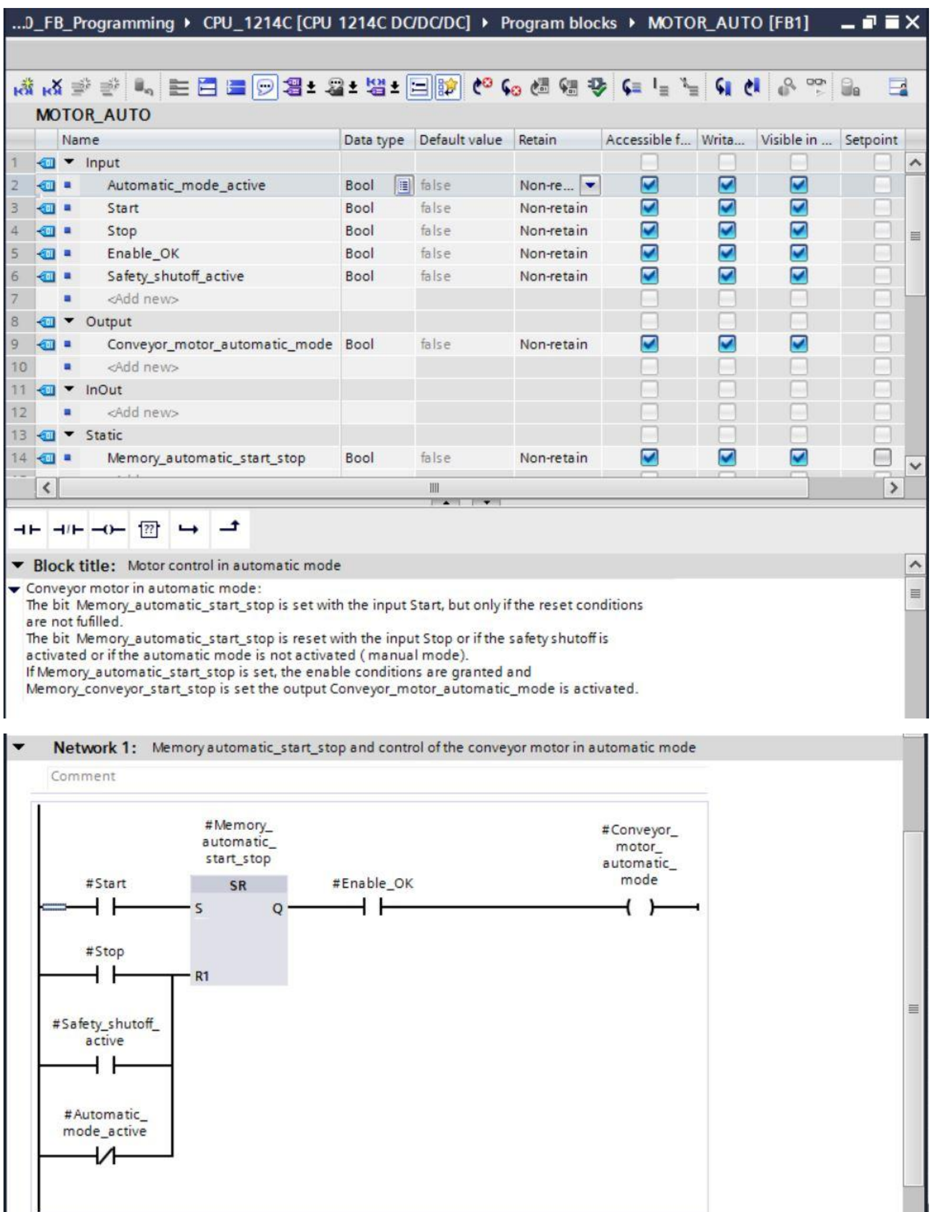
regularly. The finished function block "MOTOR_AUTO" [FB1] in FBD is shown below.



→ Under "General" in the properties of the block, you can change the "Language" to LAD (Ladder Logic) (→Properties → General → Language: LAD)



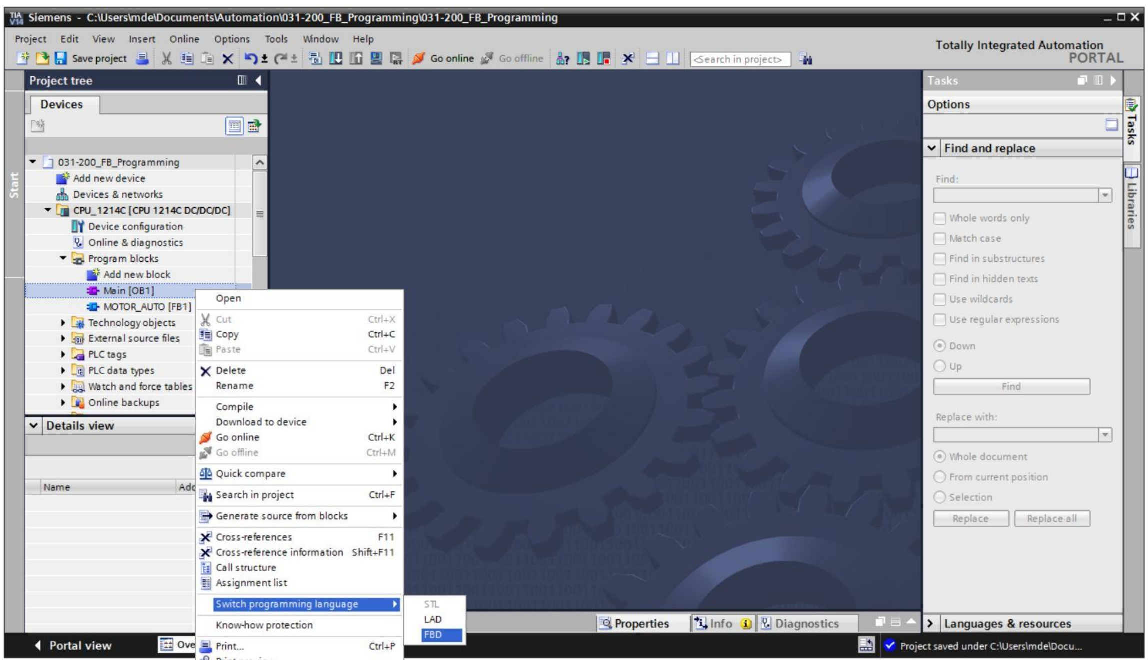
→ The program has the following appearance in LAD.



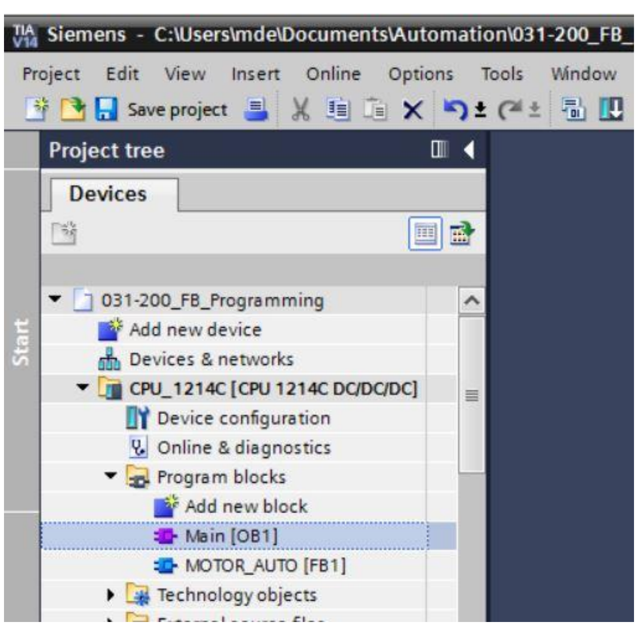
7.8 Program the organization block OB1 – Control conveyor tracking forwards in automatic mode

→ Before programming organization block "Main [OB1]", we switch the programming language to FBD (Function Block Diagram). To do so, first click on "Main [OB1]" in the "Program blocks" folder.

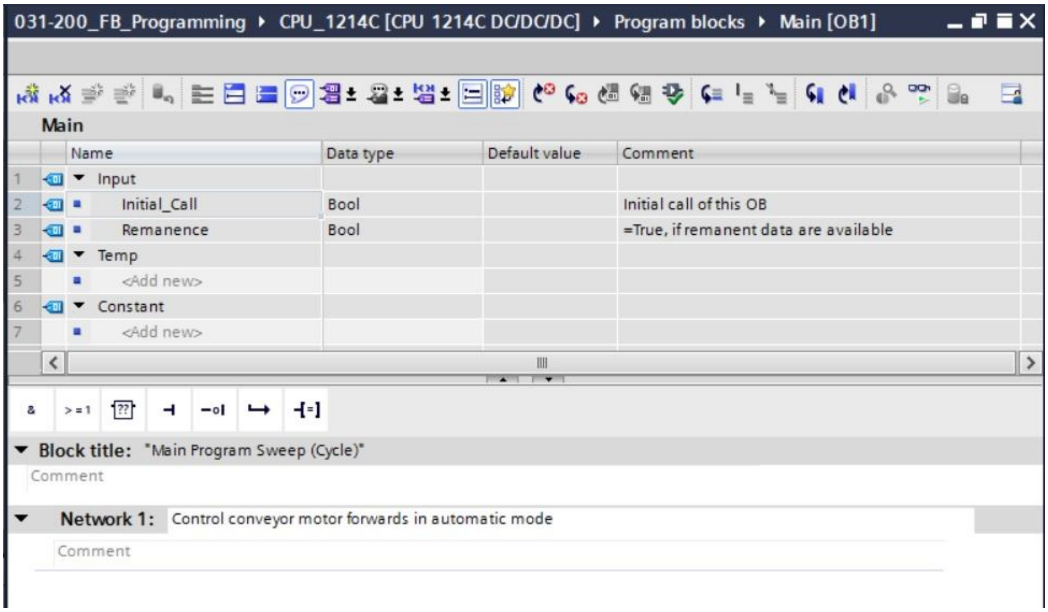
(→ CPU_1214C [CPU 1214C DC/DC/DC] → Program blocks → Main [OB1] → Switch programming language → FBD)



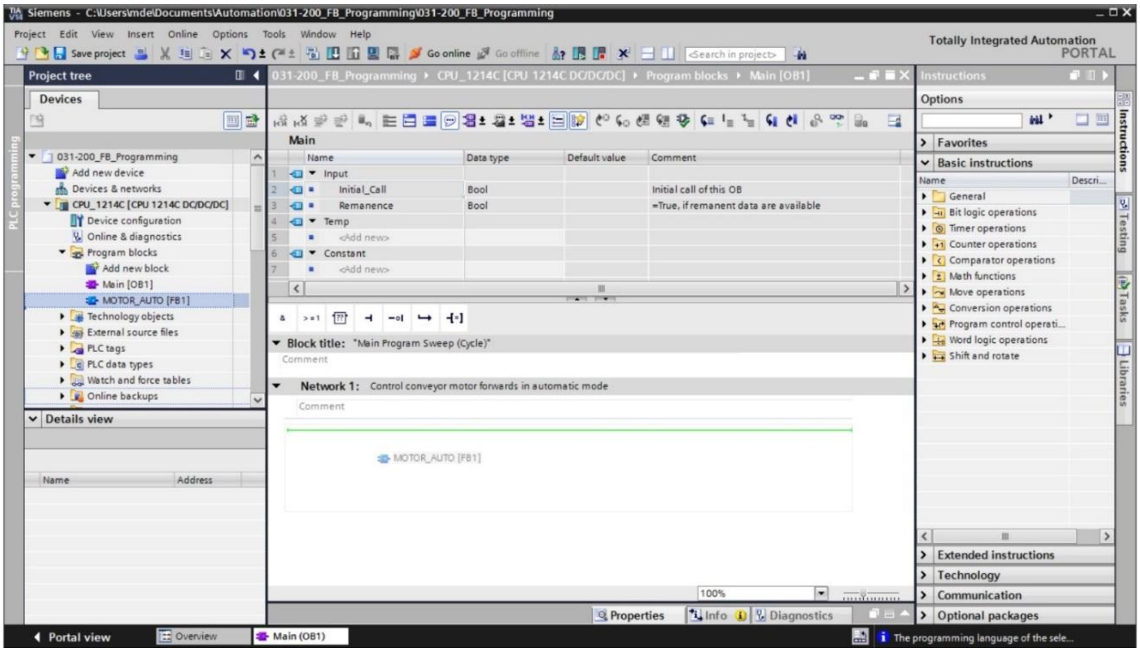
→ Open the "Main [OB1]" organization block with a double-click.



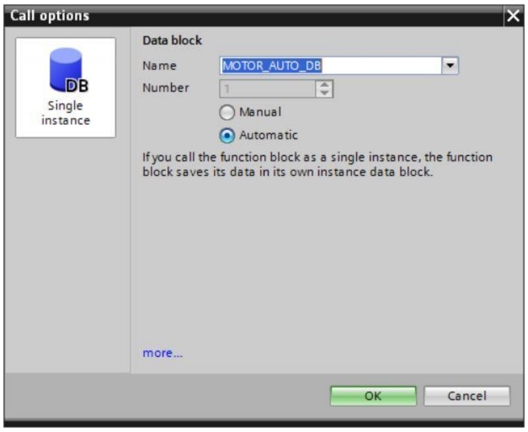
→ Assign Network 1 the name "Control conveyor tracking forwards in automatic mode" (→ Network 1:... →Control conveyor tracking forwards in automatic mode)



→ Use drag & drop to move your "MOTOR_AUTO [FB1]" function block onto the green line in Network 1.

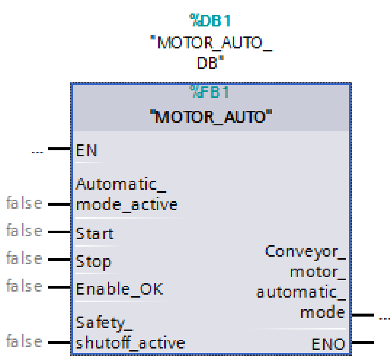


→ The instance data block for this call of FB1 is created automatically. Assign a name and apply it with OK. (→ MOTOR_AUTO_DB1 → OK)

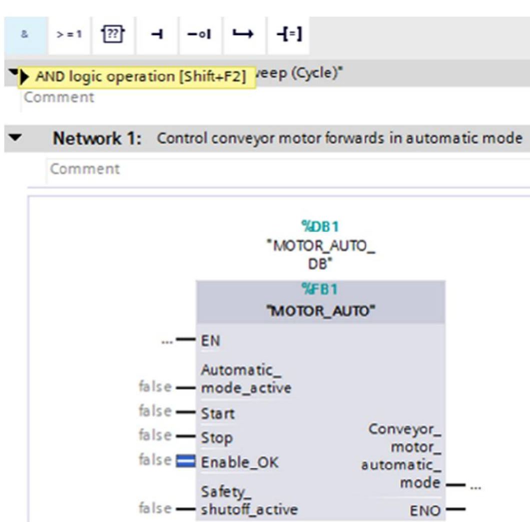


6

→ A block with the interface you defined, the instance data block and connections EN and ENO are inserted in Network 1.

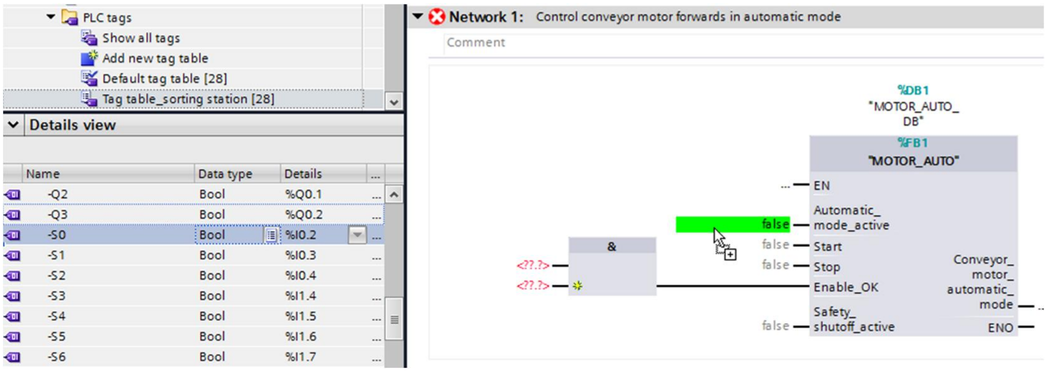


→ To insert an AND before input parameter "Enable_OK", select this input and insert the AND by clicking the  icon in your logic toolbar (→ ).



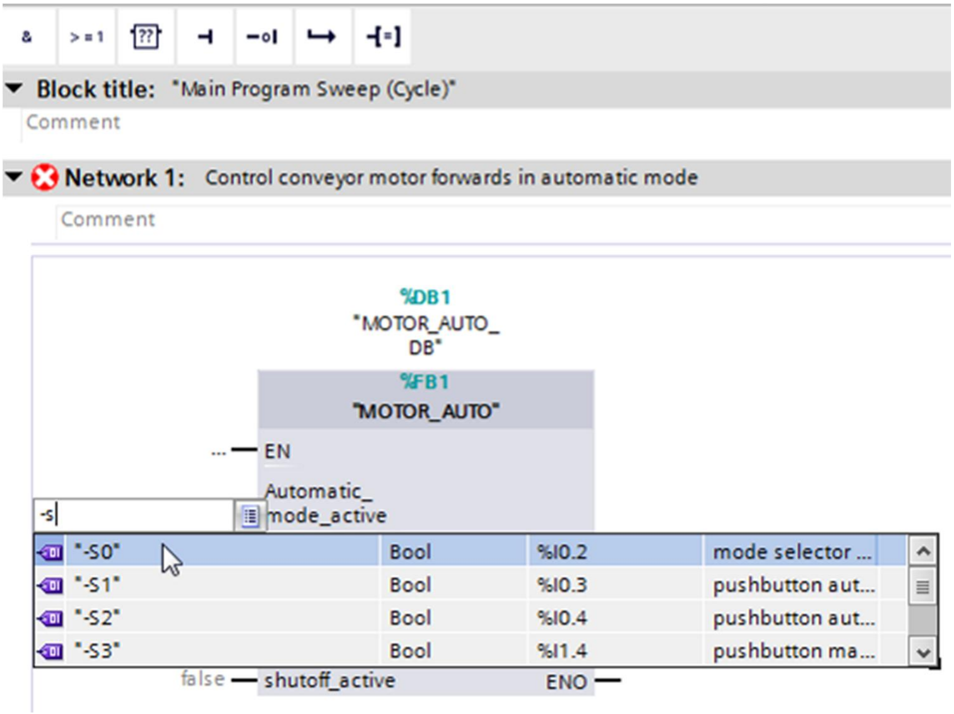
→ To connect the block to the global tags from "Tag_table_sorting_station", we have two options:

Either select the "Tag_table_sorting_station" in the project tree and use drag & drop to move the desired global tag from the Details view to the interface of FC1 (→ Tag_table_sorting_station → Details view → -S0 → Automatic_mode_active)



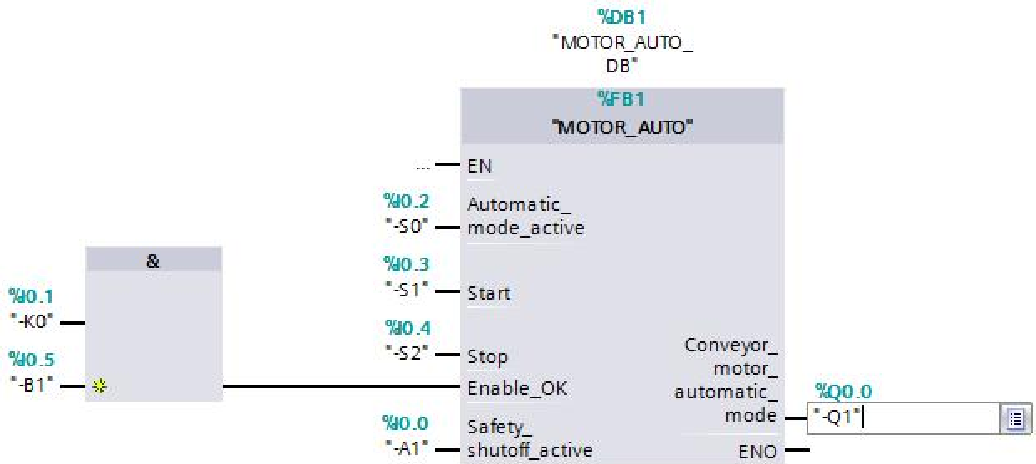
→ Or, enter the starting letters (e.g. "-S") of the desired global tag for <??.?> and select the global input tag "-S0" (%I0.2) from the displayed list

(→ Automatic_mode_active → -S → -S0).



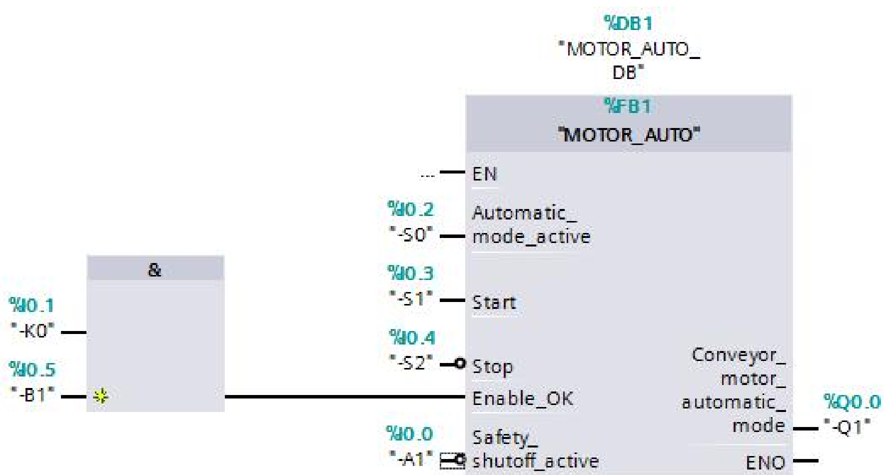
6

→ Insert the other input tags "-S1", "-S2", "-K0", "-B1" and "-A1" and then insert output tag "-Q1" (%Q0.0) at output "Conveyor_motor_automatic_mode".



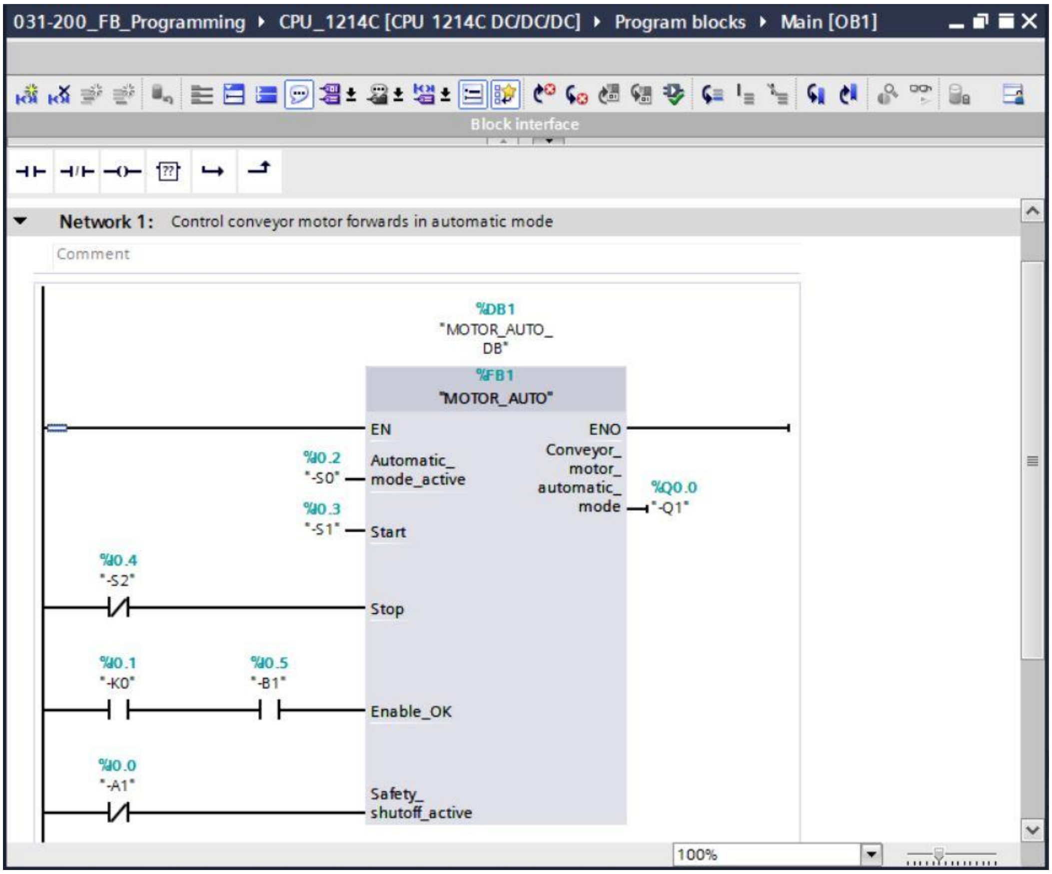
6

→ Negate the querying of input tags "-S2" and "-A1" by selecting them and clicking (→ -S2 → → -A1 →).



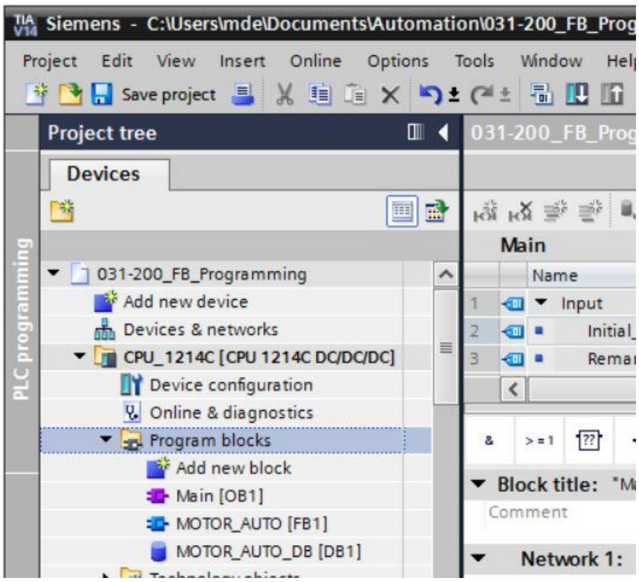
7.9 Result in the LAD (Ladder Logic) programming language

The result in the LAD (Ladder Logic) programming language has the following appearance.

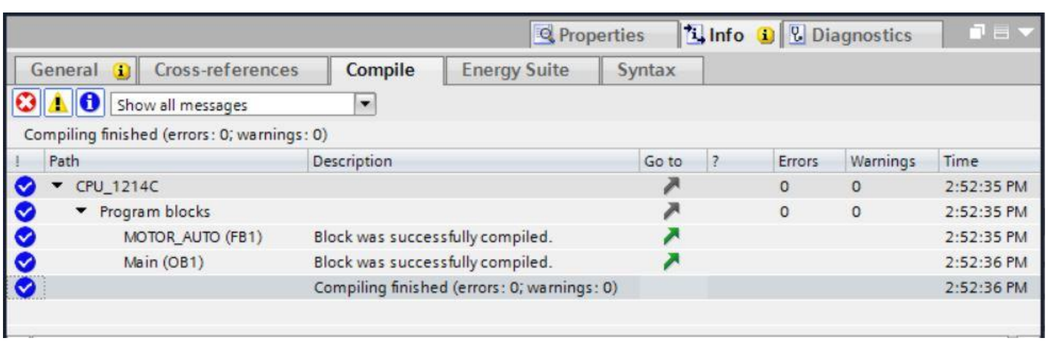


7.10 Save and compile the program

→ To save your project, select the Save project button in the menu. To compile all blocks, click the "Program blocks" folder and select the icon for compiling in the menu (→ Save project → Program blocks →).

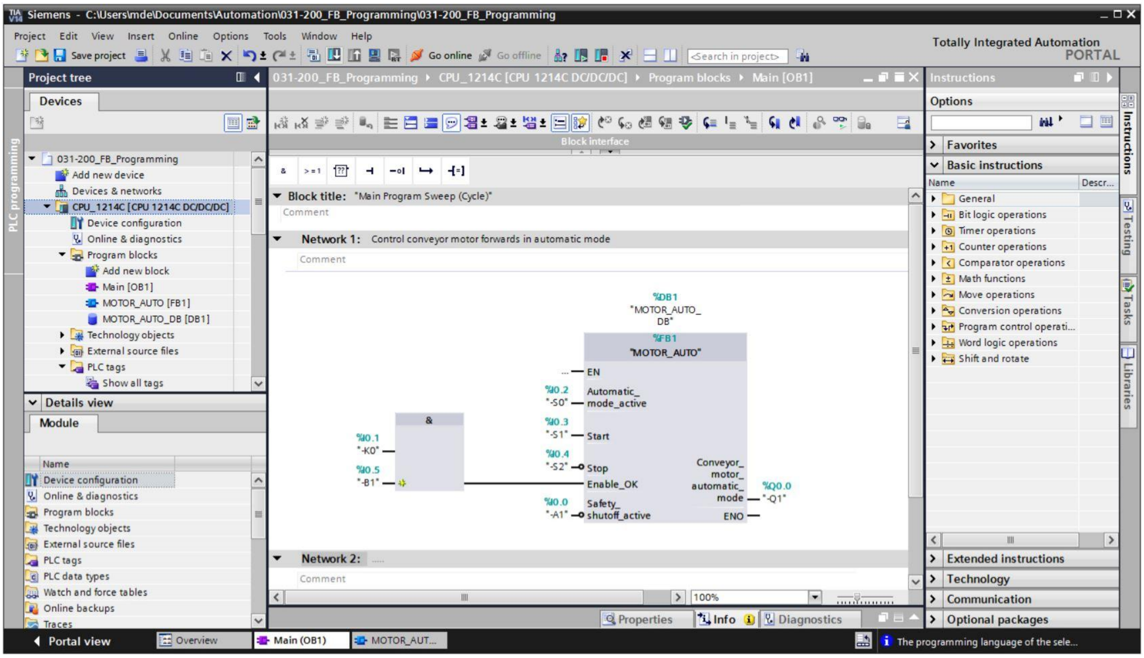


→ The "Info", "Compile" area shows which blocks were successfully compiled.



7.11 Download the program

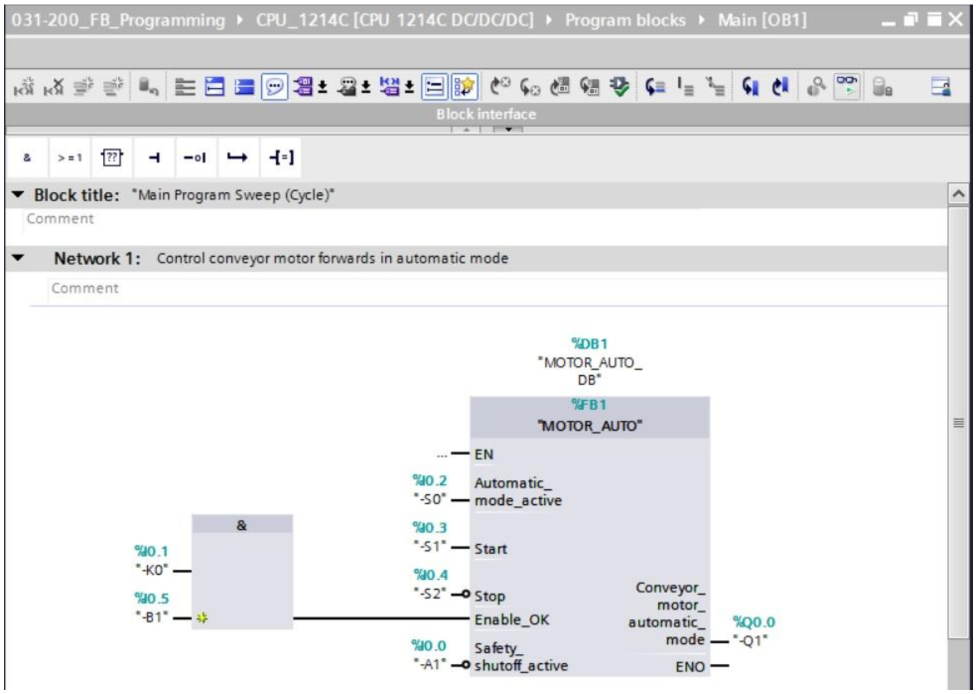
→ After successful compilation, the complete controller with the created program, as previously described in the modules for hardware configuration, can be downloaded (→ ).

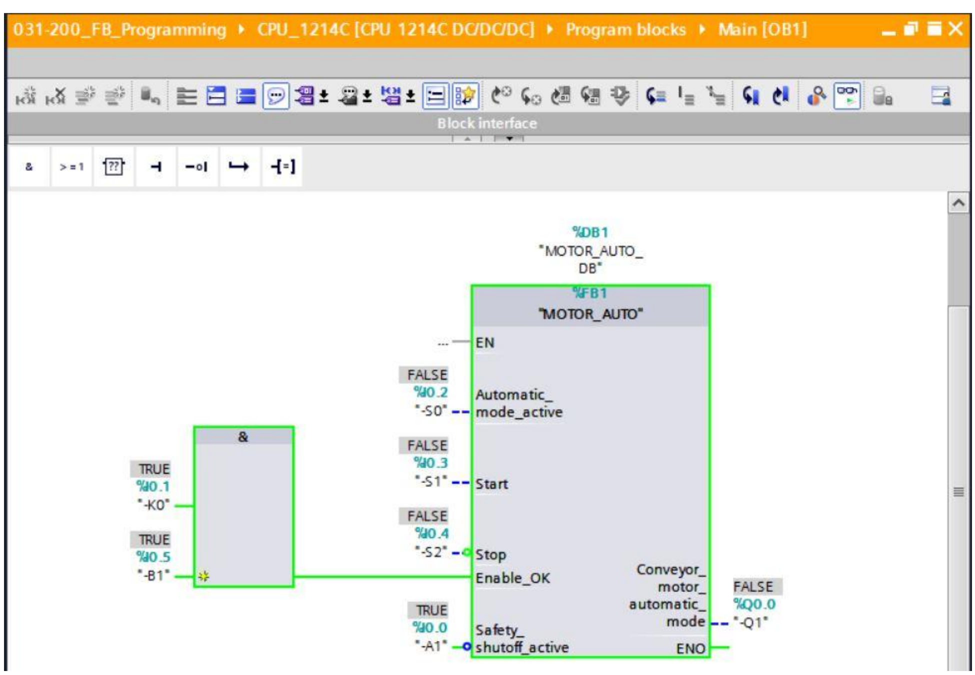


6

7.12 Monitor program blocks

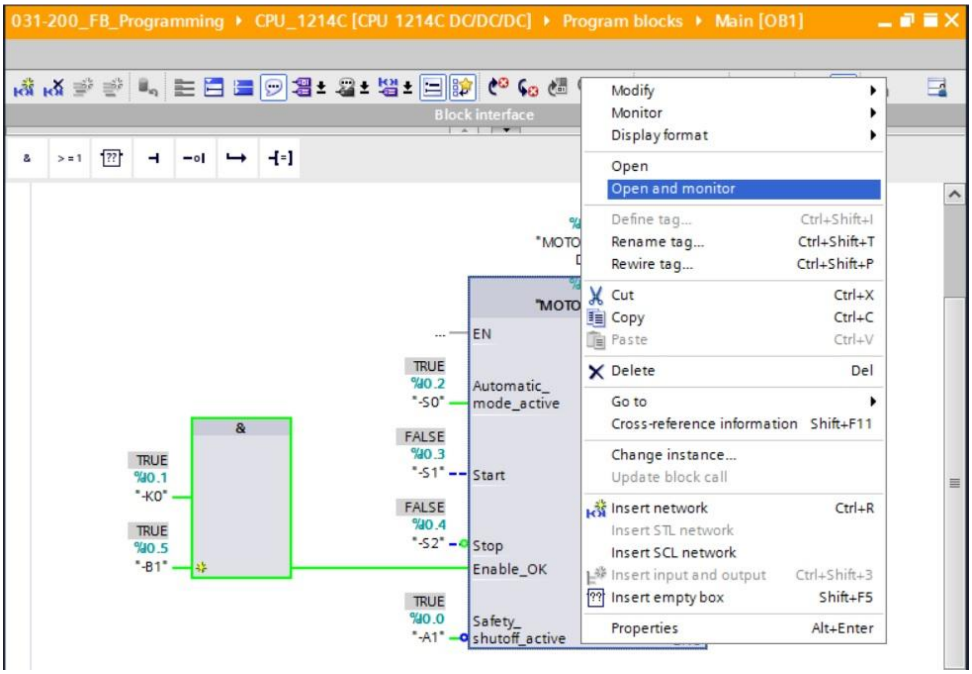
→ The desired block must be open for monitoring the downloaded program. The monitoring can now be activated/deactivated by clicking the icon (→ Main [OB1] →).

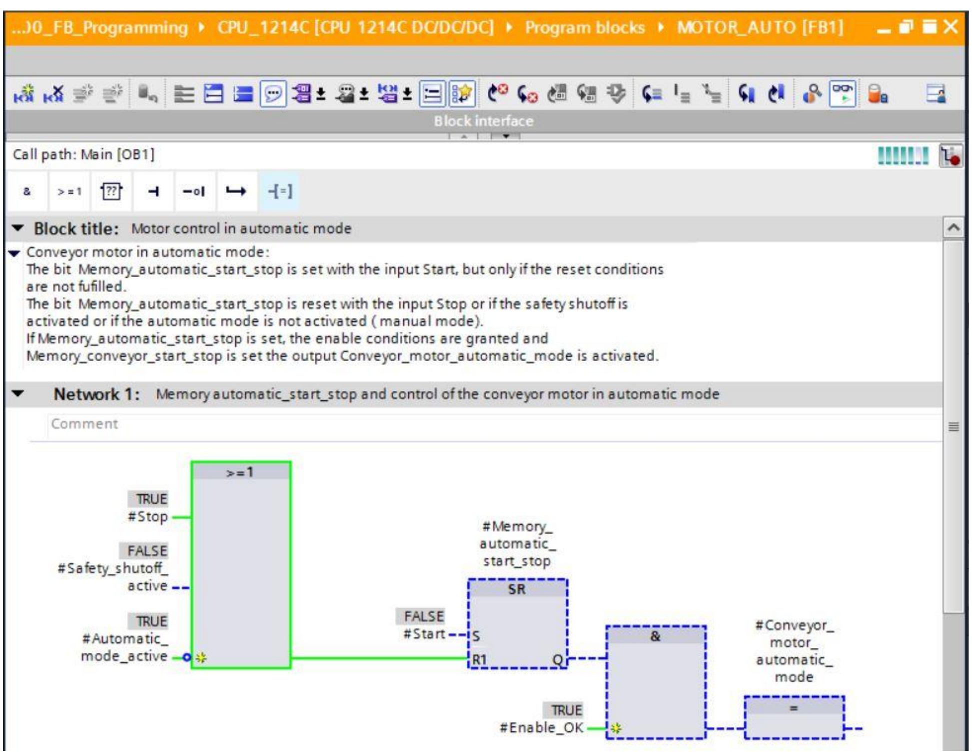




Note: *The monitoring here is signal-related and controller-dependent. The signal states at the terminals are indicated with TRUE or FALSE.*

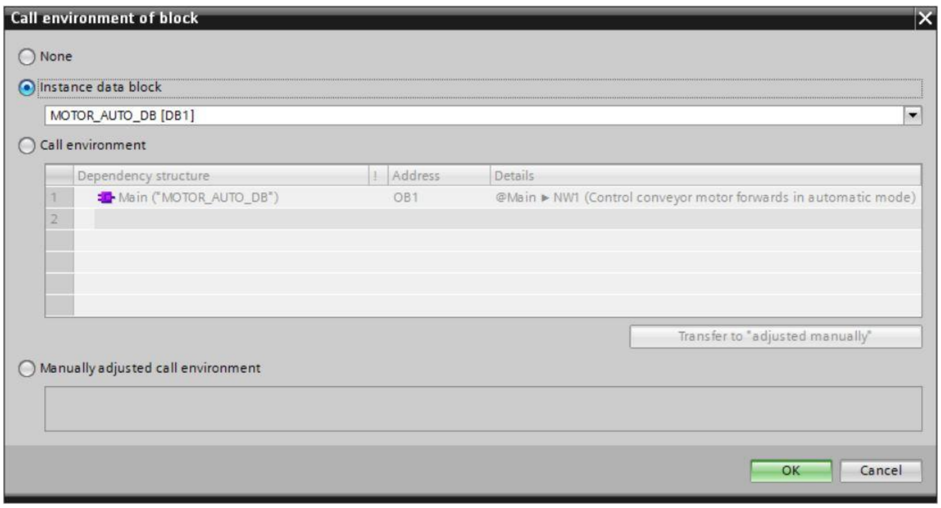
→ The "MOTOR_AUTO" [FB1] function block called in the "Main [OB1]" organization block can be selected directly for "Open and monitor" after right-clicking (→ "MOTOR_AUTO" [FB1] → Open and monitor).

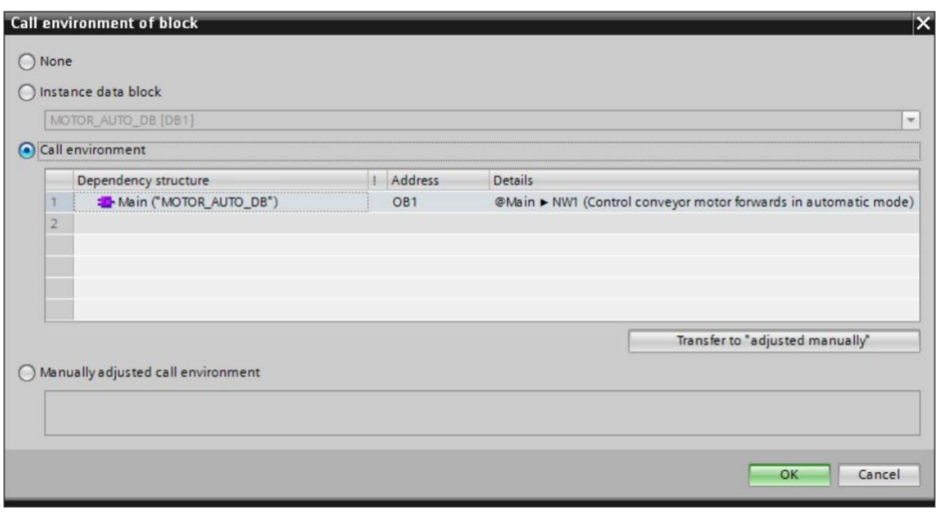




Note: *The monitoring here is function-related and controller-independent. The actuation of sensors and the station status are shown here with TRUE or FALSE.*

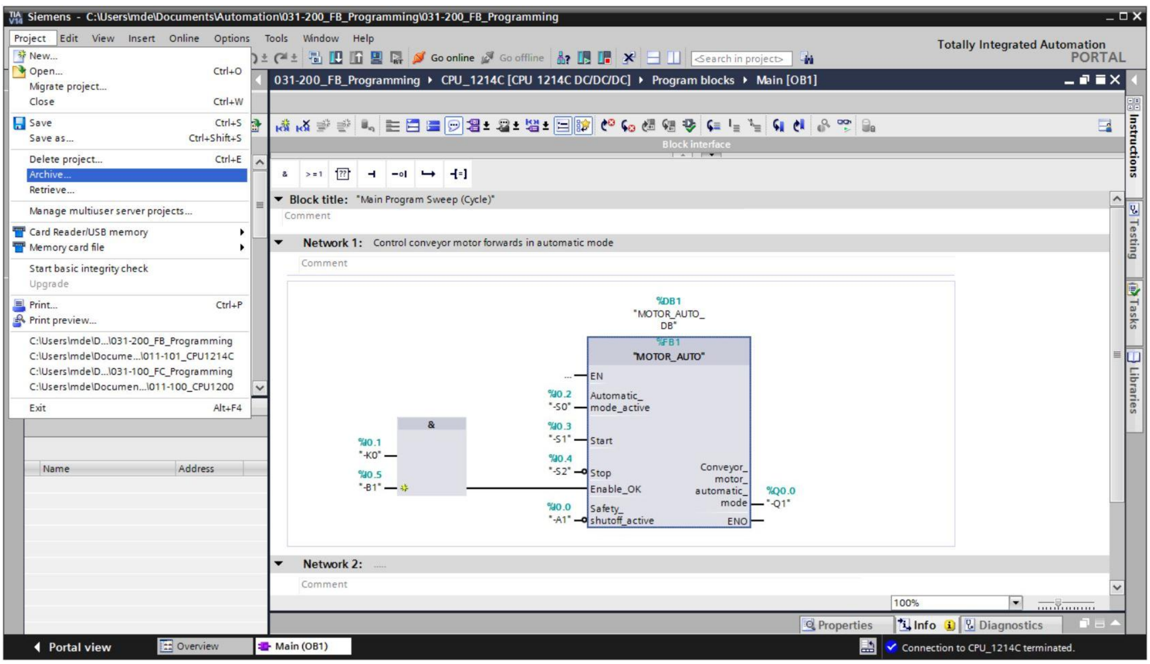
→ If a particular point of use of a "MOTOR_AUTO" [FB1] function block that is called multiple times is to be monitored, this can be performed using the icon. There are two alternatives available for specifying the call environment: using the call environment or the instance data block (→ → Instance data block → MOTOR_AUTO_DB1 [DB1] → Call environment → Address: OB1 → Details: Main NW1 → OK).





7.13 Archive the project

→ As the final step, we want to archive the complete project. Select the → "Archive ..." command in the → "Project" menu. Select a folder where you want to archive your project and save it with the file type "TIA Portal project archive". (→ Project → Archive → TIA Portal project archive → 031-200_FB Programming…. → Save)



7.14 Checklist

No.	Description	Completed
1	Compiling successful and without error message	
2	Download successful and without error message	
3	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated AUTOMATIC mode (-S0 = 1) Pushbutton automatic stop not actuated (-S2 = 1) Briefly press the automatic start pushbutton (-S1 = 1) Conveyor motor forwards fixed speed then switches on (-Q1 = 1) switches on and stays on.	
4	Briefly press the automatic stop pushbutton (-S2 = 0) → -Q1 = 0	
5	Activate EMERGENCY OFF (-A1 = 0) → -Q1 = 0	
6	Manual mode (-S0 = 0) → -Q1 = 0	
7	Switch off station (-K0 = 0) → -Q1 = 0	
8	Cylinder not retracted (-B1 = 0) → -Q1 = 0	
9	Project successfully archived	

6

8 Exercise

8.1 Task – Exercise

In this exercise, an energy saving function is to be added to the MOTOR_AUTO [FB1] function block. The expanded function block will be planned, programmed and tested:

To save energy, the conveyor should only run when a part is present.

The Conveyor_motor_automatic_mode output is therefore only activated when Memory_automatic_start_stop is set, the enable conditions are met and Memory_conveyor_start_stop is set.

The Memory_conveyor_start_stop is set when Sensor_chute_occupied signals a part and is reset when Sensor_end_of_conveyor produces a negative edge or safety shutoff is active or automatic mode is not activated (manual mode).

8.2 Technology diagram

Here, you see the technology diagram for the task.

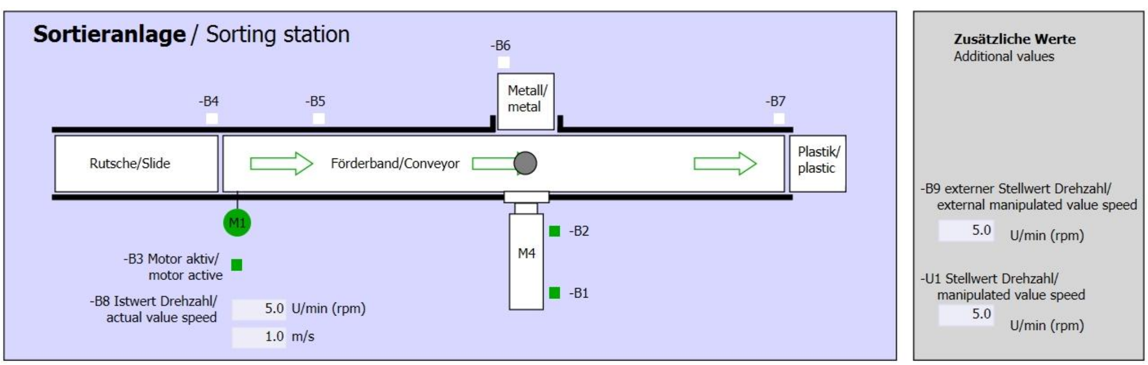


Figure 10: Technology diagram

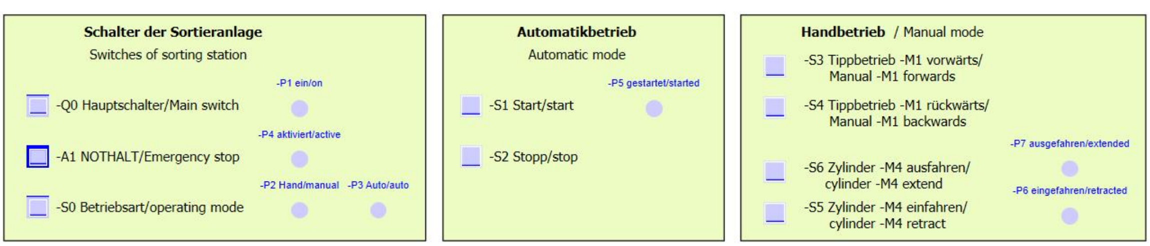


Figure 11: Control panel

8.3 Reference list

The following signals are needed as global operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop ok	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	-S1	Pushbutton automatic start	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder M4 retracted	NO
I 1.0	BOOL	-B4	Sensor at chute occupied	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO

DO	Type	Identifier	Function	
Q 0.0	BOOL	-Q1	Conveyor motor M1 forwards fixed speed	

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

8.4 Planning

Plan the implementation of the task on your own.

Note: *Learn about the use of the negative edge in SIMATIC S7-1200 in the online help.*

6

8.5 Checklist – Exercise

No.	Description	Completed
1	Compiling successful and without error message	
2	Download successful and without error message	
3	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated AUTOMATIC mode (-S0 = 1) Pushbutton automatic stop not actuated (-S2 = 1) Briefly press the automatic start pushbutton (-S1 = 1) Sensor at chute activated (-B4 = 1) Conveyor motor forwards fixed speed then switches on (-Q1 = 1) switches on and stays on.	
4	Sensor at end of conveyor activated (-B7 = 1) → -Q1 = 0	
5	Briefly press the automatic stop pushbutton (-S2 = 0) → -Q1 = 0	
6	Activate EMERGENCY OFF (-A1 = 0) → -Q1 = 0	
7	Manual mode (-S0 = 0) → -Q1 = 0	
8	Switch off station (-K0 = 0) → -Q1 = 0	
9	Cylinder not retracted (-B1 = 0) → -Q1 = 0	
10	Project successfully archived	

9 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

6

Notes

6

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is available at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, get in touch with your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

7

Information regarding use

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of

Table of contents

IEC Timers and IEC Counters Multi-instances for SIMATIC S7-1200

1 Goal

In this chapter, you will become acquainted with the use of single instances and multi-instances for programming of SIMATIC S7-1200 with the TIA Portal programming tool.

The module explains the various types of instance data blocks and shows step-by-step how to add IEC timers and IEC counters to a program block.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

This chapter builds on the FB programming for SIMATIC S7 CPU1214C. For this chapter, you can use the following project, for example:

031-200_FB-Programming_S7-1200….zap14

3 Required hardware and software

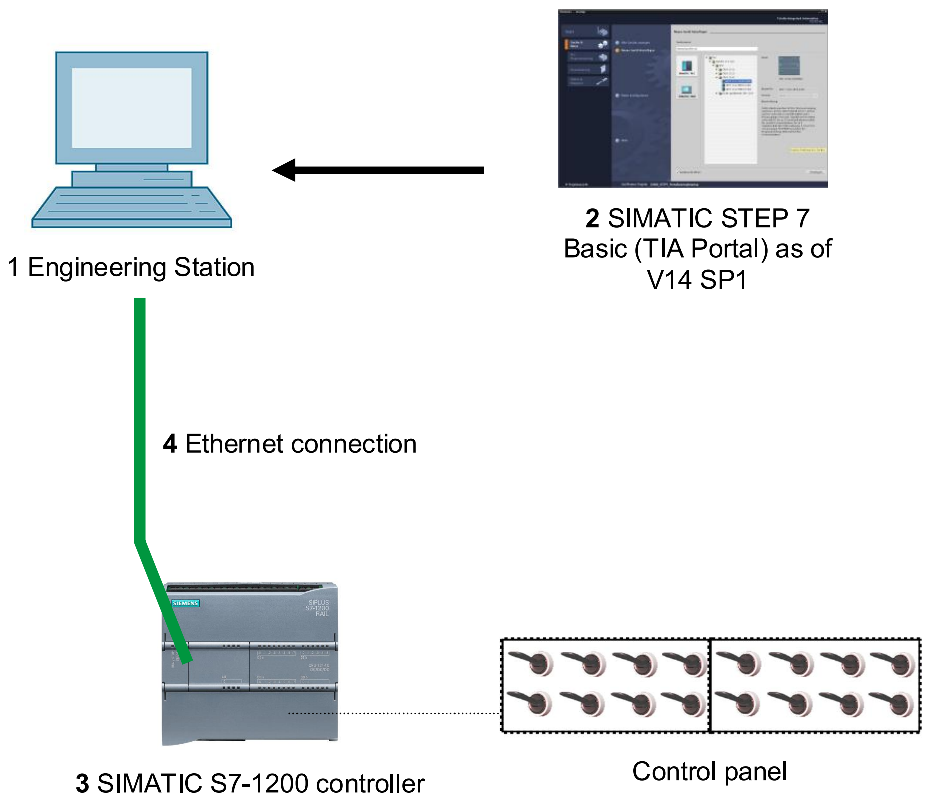
1 Engineering station: requirements include hardware and operating system (for additional information, see Readme on the TIA Portal Installation DVDs)

2 SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1

Note: The digital inputs should be fed out to a control panel.

4 Ethernet connection between engineering station and controller



4 Theory

4.1 Instances and multi-instances in SIMATIC S7-1200

The call of a function block is referred to as an **instance**. An **instance** is assigned to every call of a function block and serves as a data memory. It stores the actual parameters and the static data of the function block.

The tags declared in the function block determine the structure of the instance data block.

Use of single instances and multi-instances

You can assign instances as follows:

Call as a **single instance**:

– A separate instance data block for each instance of a function block

Call as a **multi-instance**:

– One instance data block for several instances of one or more function blocks

4.1.1 Instance data blocks / Single instances

The call of a function block that is assigned its own instance data block is called a **single instance**.

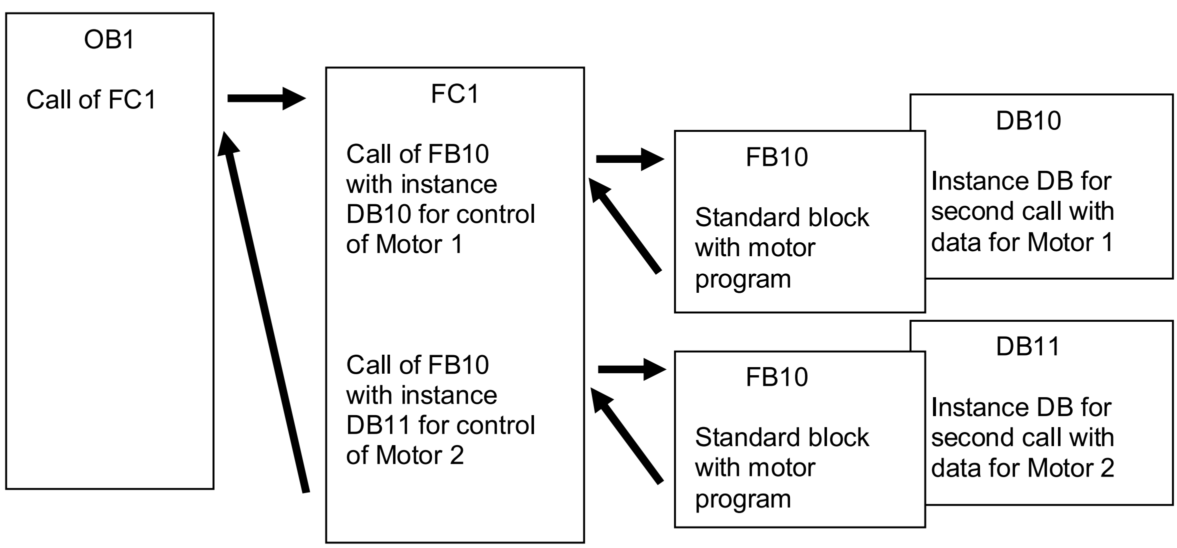
If the function block was created according to the rules for library-compatible standard blocks, it can also be called multiple times.

However, you must assign another instance data block for each call as a single instance.

Example of single instances:

The following figure shows the control of two motors using one function block FB10 and two different data blocks:

The different data for the individual motors, such as speed, acceleration time and total operating time, are saved in the instance data blocks DB10 and DB11.



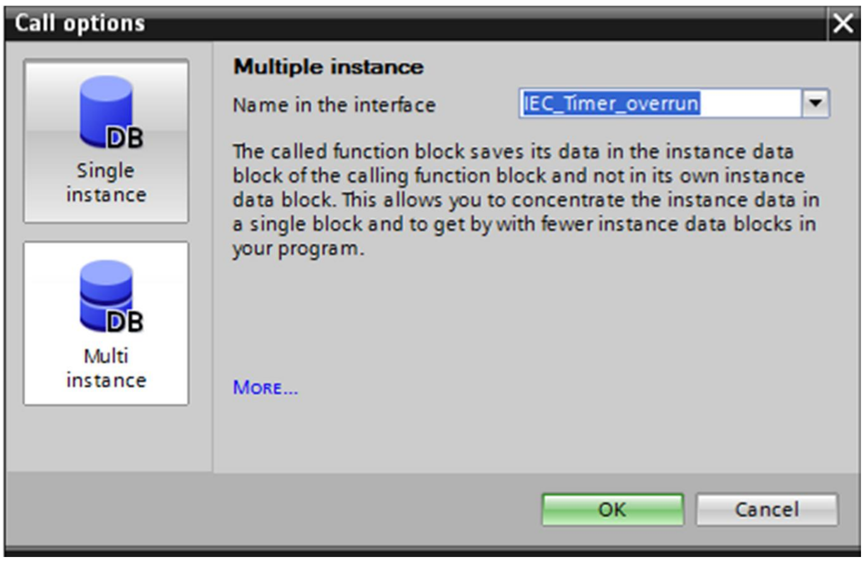
Note: *Some commands, such as timers and counters, behave like function blocks. When these are called, they also require an assigned memory area, e.g., in the form of an instance data block.*

4.1.2 Multi-instances

You may want to limit the number of data blocks used for instances or this may be necessary due to lack of memory in the utilized CPU.

If other function blocks, timers, counters, etc. that already exist are to be called in a function block in your user program, you can call these other function blocks without separate (i.e. additional) instance DBs.

Simply select '**Multi-instance**' for the call options:



7

Notes: *Multi-instances enable a called function block to store its data in the instance data block of the calling function block.*

In this case, the calling block must always be a function block.

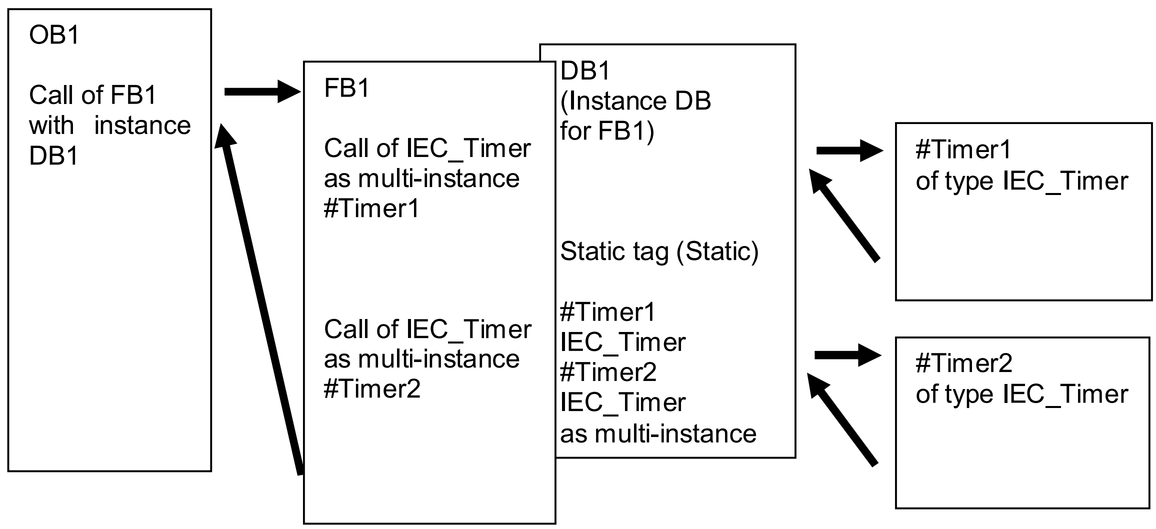
This allows you to concentrate the instance data in one instance data block and thus make better use of the number of DBs available.

Incidentally, this is always required when the calling block is to remain available for reuse as a standard block.

Example of multi-instances:

The following figure shows two calls of an IEC timer of type TP (pulse) within a function block.

The different data for the two counters is stored as different **multi-instances** in the instance data block DB1 of the calling function block FB1.



5 Task

In this chapter, an IEC timer will be added to the function block from chapter "SCE_EN_031-200 FB Programming S7-1200".

6 Planning

The IEC timer is programmed as an addition to the MOTOR_AUTO [FB1] function block from the "031-200_FB-Programming_S7-1200.zap13" project. This project must be retrieved in order to now add the IEC timer TP (latching pulse). A multi-instance will be created as a memory for the timer.

6.1 Automatic mode - Conveyor motor with time function

The Memory_automatic_start_stop is latched with Start but only if the reset conditions are not present.

The Memory_automatic_start_stop is reset if Stop is present or safety shutoff is active or automatic mode is not activated (manual mode).

The Conveyor_motor_automatic_mode output is activated when Memory_automatic_start_stop is set, the enable conditions are met and Memory_conveyor_start_stop is set.

To save energy, the conveyor should only run when a part is present.

For this reason, the Memory_conveyor_start_stop is set when Sensor_chute_occupied signals a part and reset when Sensor_end_of_conveyor produces a negative edge or safety shutoff is active or automatic mode is not activated (manual mode).

Addition of time function:

Because the Sensor_end_of_conveyor is not able to be mounted directly at the end of the conveyor, the Sensor_end_of_conveyor signal must be stretched.

To achieve this, a latching pulse will be inserted between Sensor_end_of_conveyor and the negative edge detection.

6.2 Technology diagram

Here, you see the technology diagram for the task.

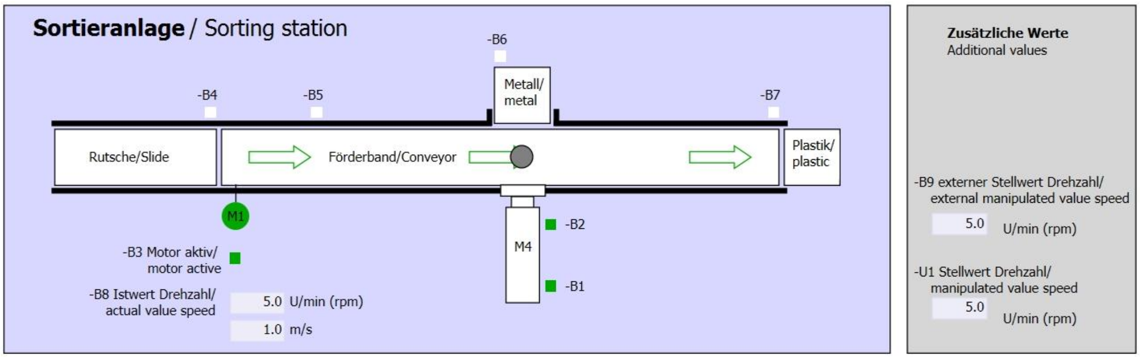


Figure 1: Technology diagram

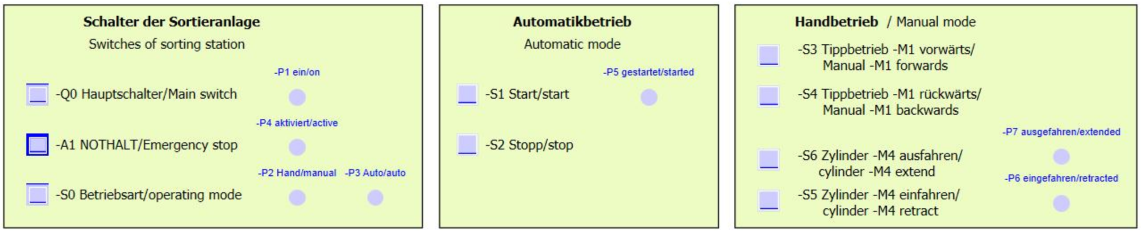


Figure 2: Control panel

7

6.3 Reference list

The following signals are needed as global operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop ok	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	-S1	Pushbutton automatic start	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder M4 retracted	NO
I 1.0	BOOL	-B4	Sensor at chute occupied	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO

7

DO	Type	Identifier	Function	
Q 0.0	BOOL	-Q1	Conveyor motor M1 forwards fixed speed	

Legend for reference list

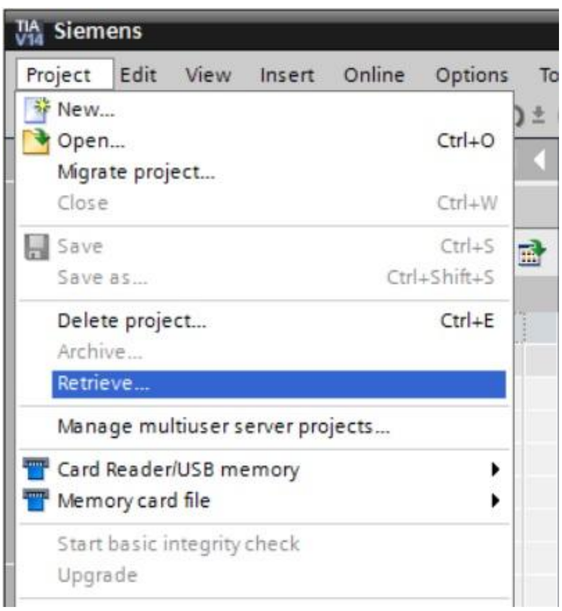
DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

7 Structured step-by-step instructions

You can find instructions on how to carry out planning below. If you already have a good understanding of everything, it will be sufficient to focus on the numbered steps. Otherwise, simply follow the detailed steps in the instructions.

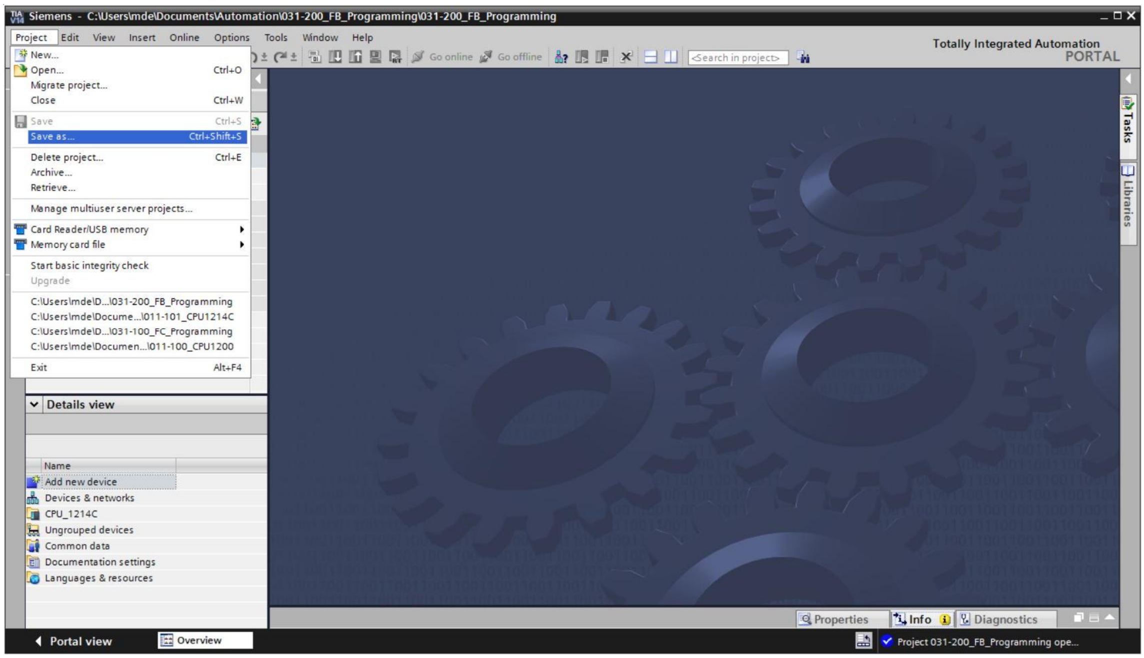
7.1 Retrieve an existing project

→ Before we can expand the "MOTOR_AUTO [FB1]" function block, we must retrieve the "031-200_FB-Programming_S7-1200.zap14" project from chapter "SCE_EN_031-200 FB Programming S7-1200". To retrieve an existing project that has been archived, you must select the relevant archive with → Project → Retrieve in the project view. Confirm your selection with Open (→ Project → Retrieve → Select a .zap archive → 031-200_FB-Programming_S7-1200.zap14 → Open).



→ The next step is to select the target directory where the retrieved project will be stored. Confirm your selection with "OK".

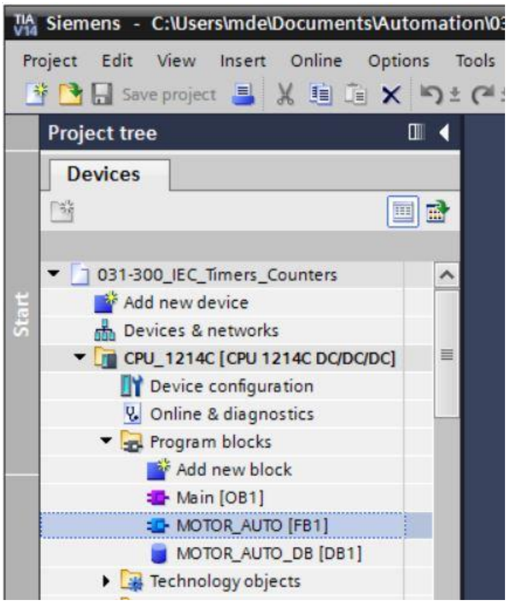
→ Save the opened project under the name 031-300_IEC_Timers_Counters (→ Project → Save as ... → 031-300_IEC_Timers_Counters → Save)



7

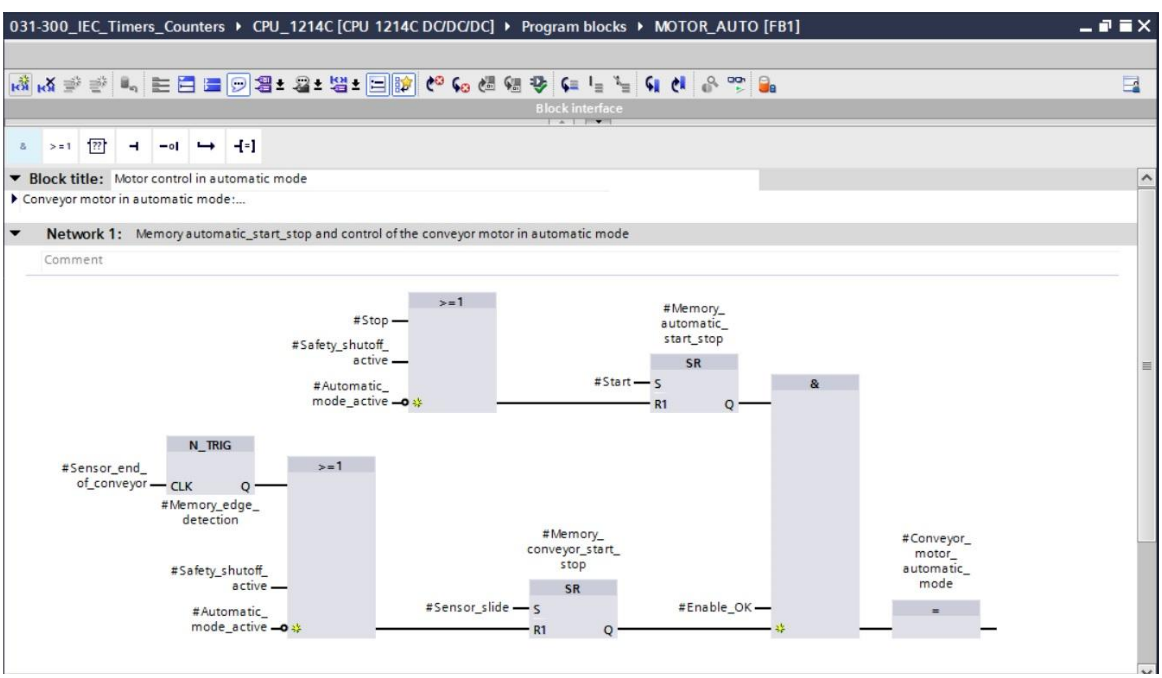
7.2 Addition of an IEC timer TP to function block FB1 "MOTOR_AUTO"

→ First, open the "MOTOR_AUTO [FB1]" function block with a double-click.

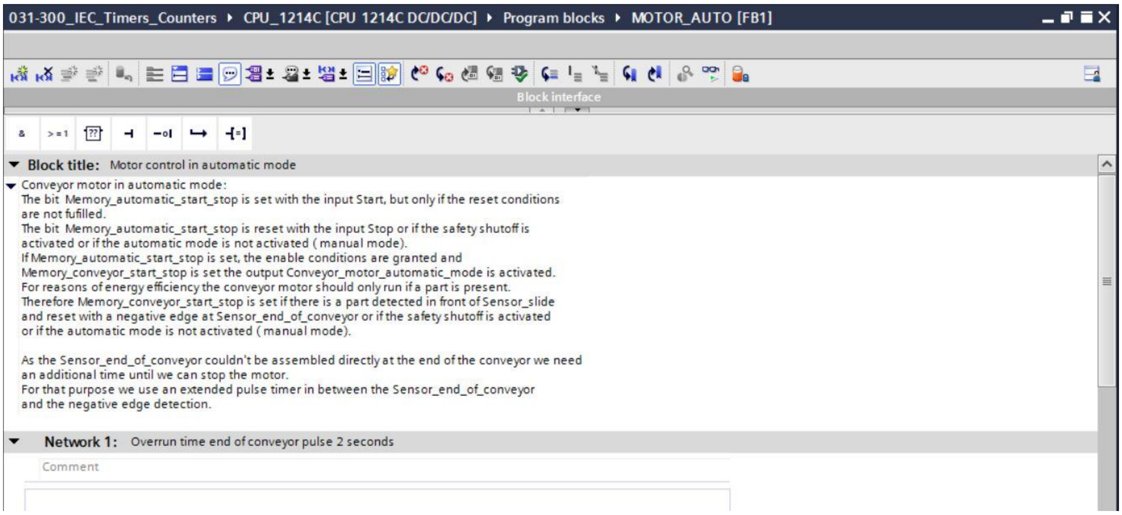


7

→ Insert another network at the beginning of the "MOTOR_AUTO [FB1]" function block by selecting the → "block title" and clicking the → icon for "Insert network".

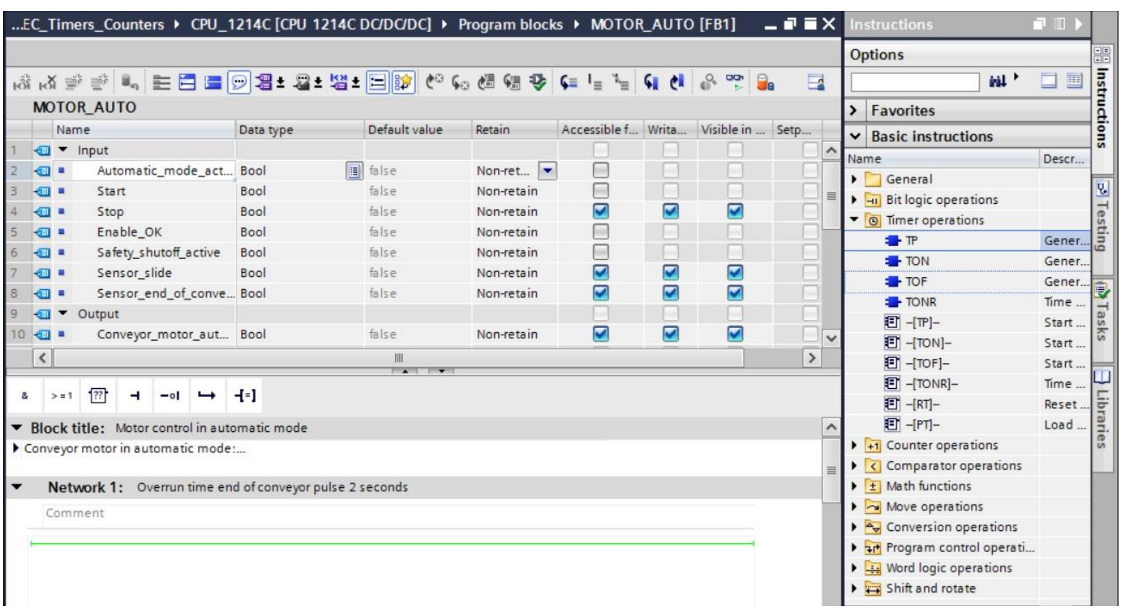


→ Add helpful information to the block comment and the network title of "Network 1:".

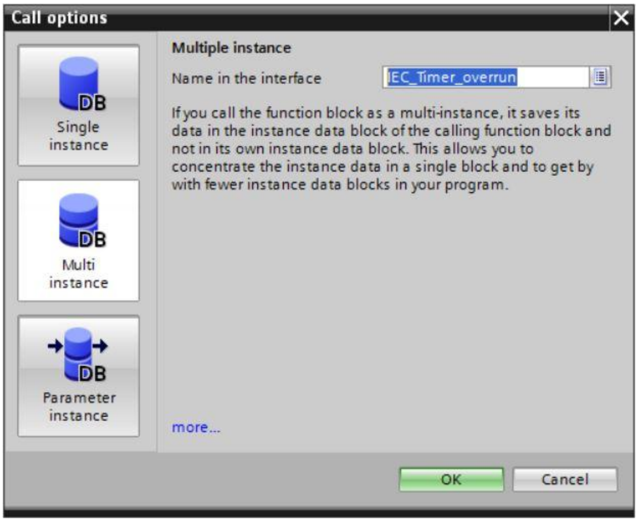


→ On the right side of your programming window, you will see the timer functions in the list of instructions. Under → Basic instructions → Timer operations, find function TP (Generate pulse) and use a drag & drop operation to move it to Network 1 (green line appears, mouse pointer with + symbol).

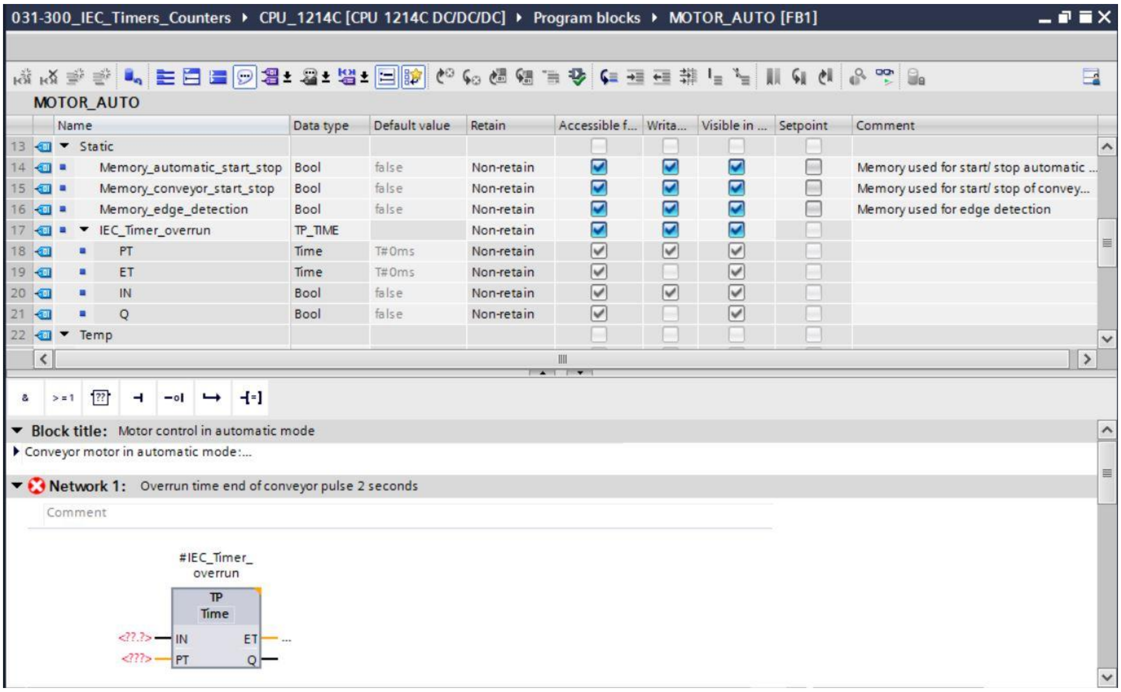
(→ Instructions → Basic instructions → Timer operations → TP)



→ The timer function requires a memory. This memory is provided in this case within the instance data block of the function block without creating a new instance data block. Select the →"Multi-instance" option for this. Enter a name for the multi-instance and confirm with → "OK". (→ Multi-instance → IEC_Timer_overrun → OK)

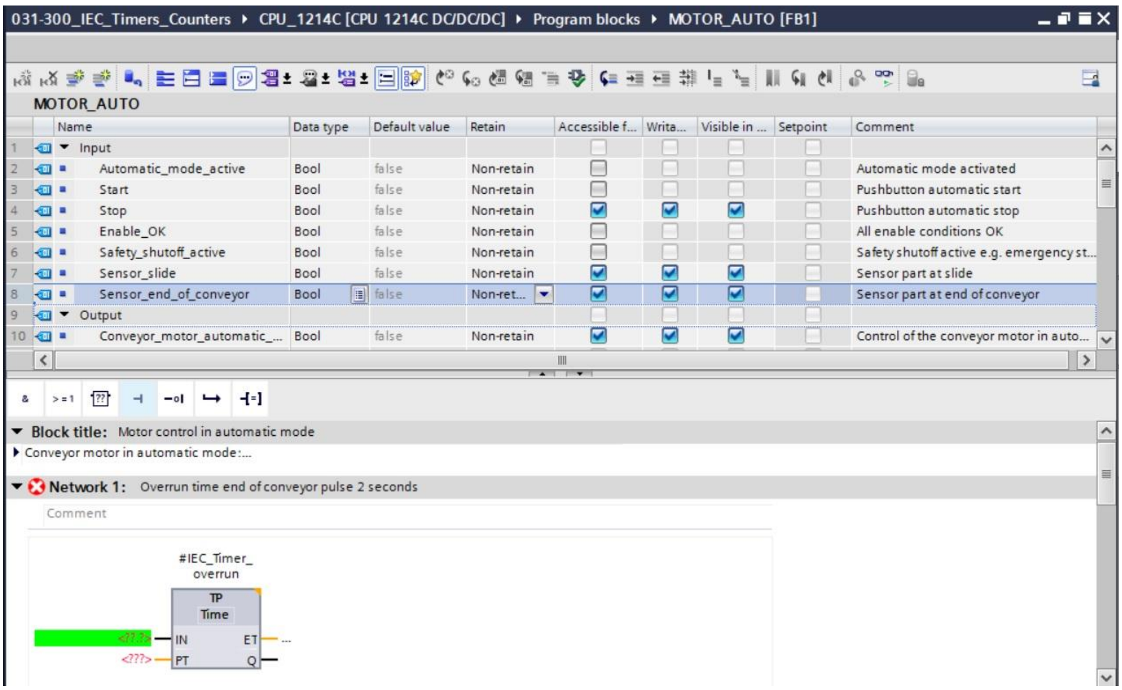


→ As a result, a tag structure of "Static" type suitable for TP Timer will be created in the interface description.

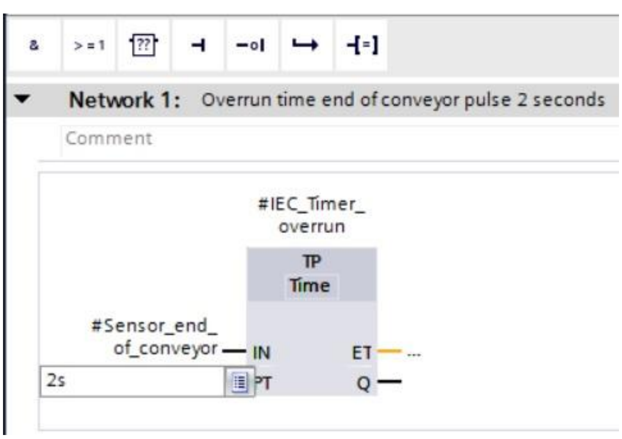


Note: *A multi-instance can only be used for programming within a function block because static tags are only available there.*

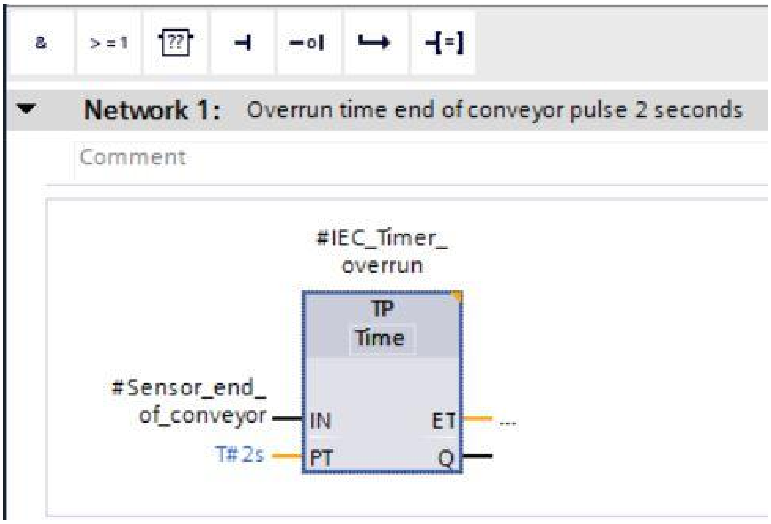
→ Use drag & drop to move input parameter #Sensor_end_of_conveyor to <??.?> in front of parameter "IN" of TP Timer so that this will be started at a positive edge at input #Sensor_end_of_conveyor. The best way to select a parameter in the interface description is by "grabbing" it at the blue symbol (→ Sensor_end_of_conveyor).



→ Enter the required pulse duration of 2 seconds in front of parameter "PT" (→2s).

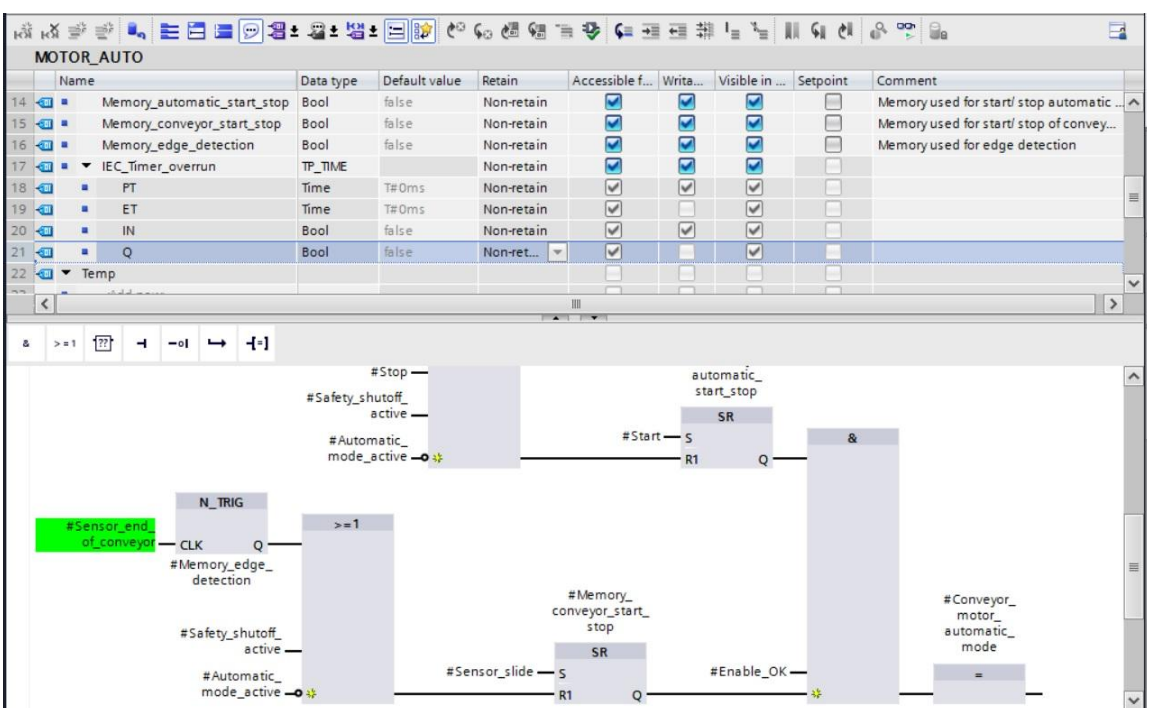


→ The entry of 2s is converted automatically to the IEC-Time format suitable for the IEC timer and is shown as constant "T#2s".

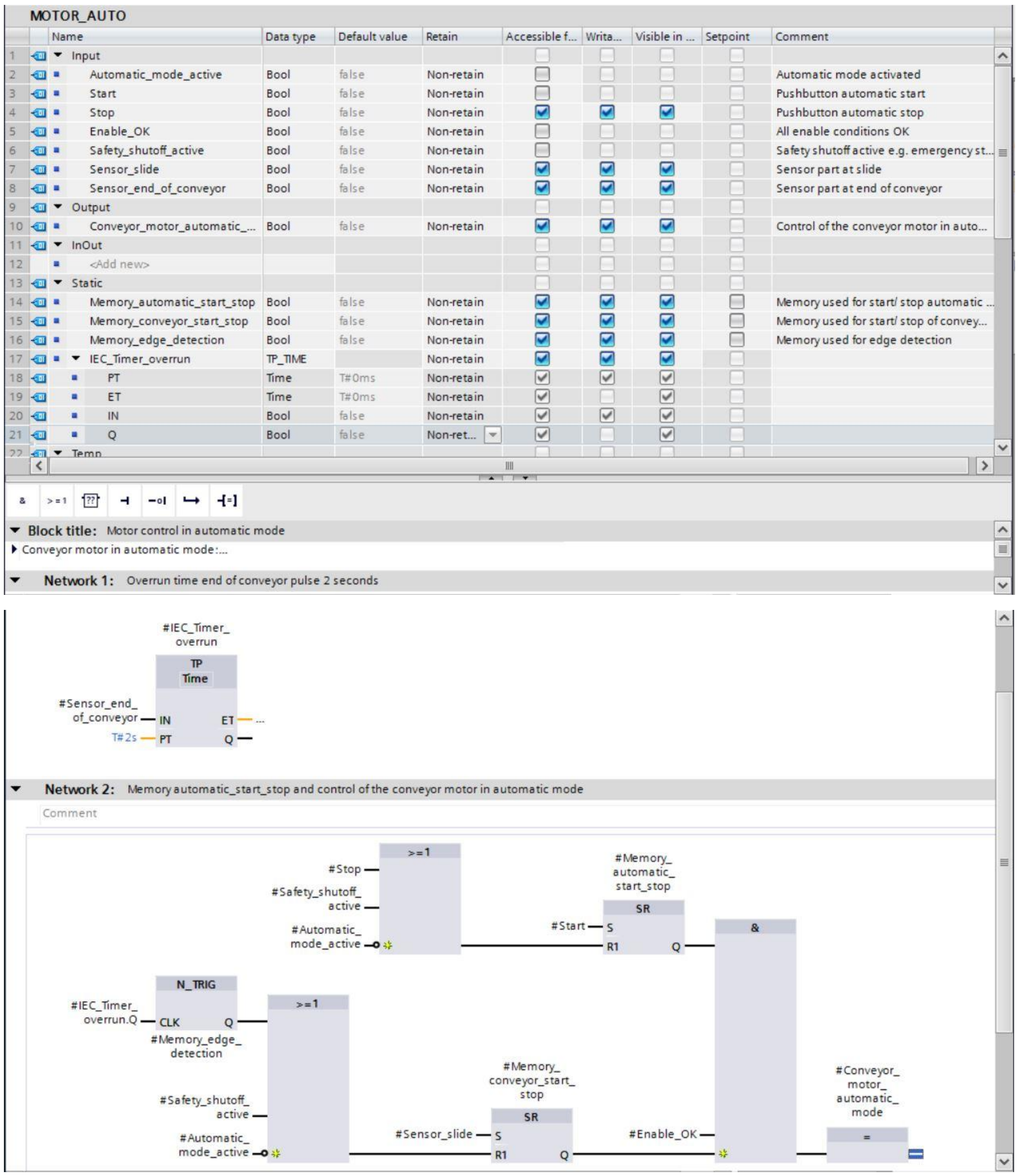


→ Now move output "Q" from tag structure "IEC_Timer_overrun" onto input "CLK" of negative edge "N_TRIG" in Network 2. This will replace the #Sensor_end_of_conveyor input tag previously entered there and the conveyor will be stopped by a negative edge of the IEC_Timer_overrun pulse.

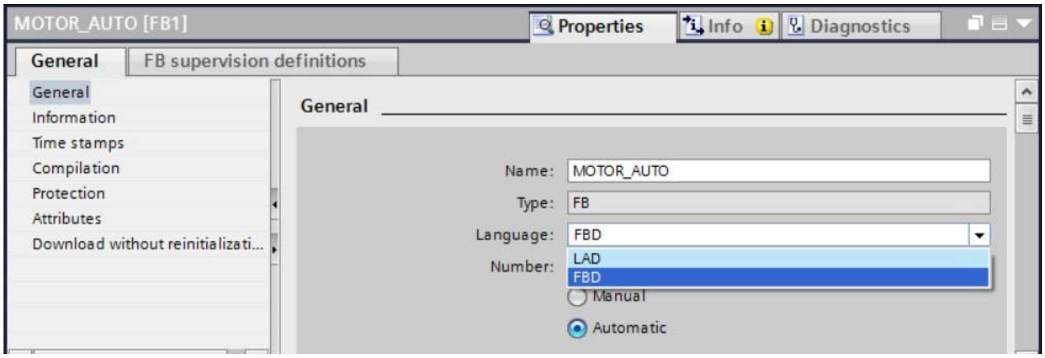
(→ Network 2 → IEC_Timer_overrun→ Q → #Sensor_end_of_conveyor)



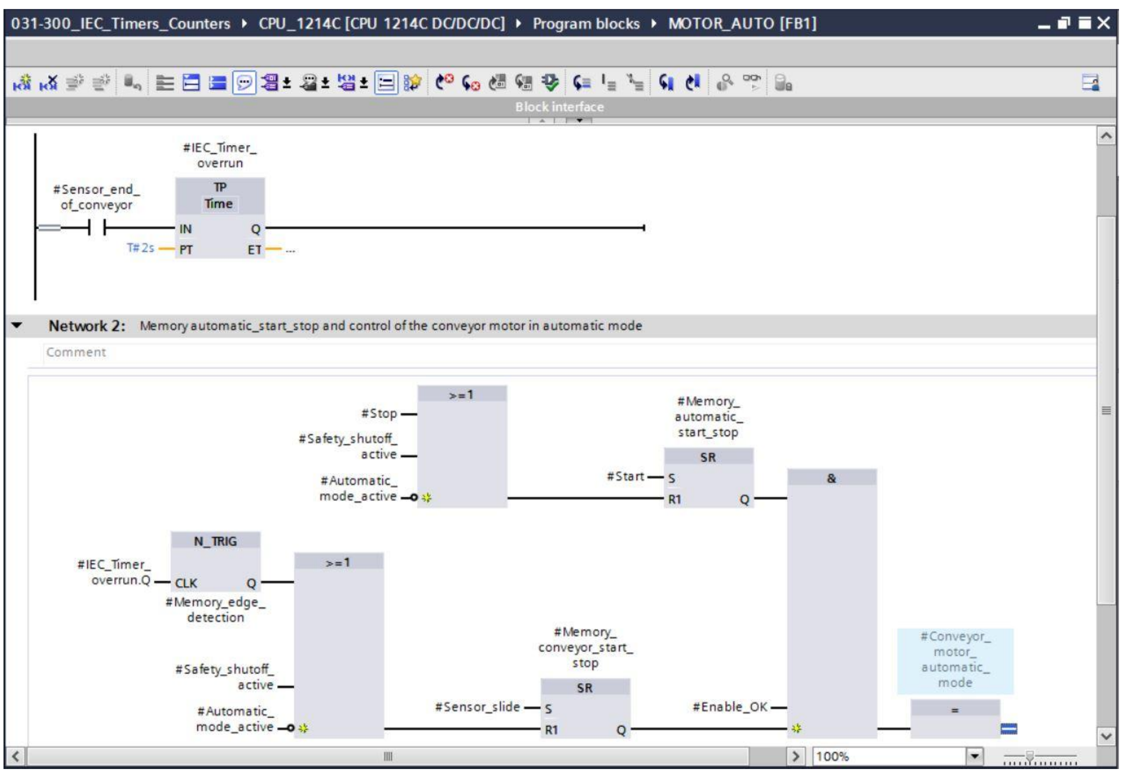
→ Do not forget to click Save project regularly. The finished function block "MOTOR_AUTO" [FB1] with the timer is shown in FBD below.



→ Under "General" in the properties of the block, you can change the "Language" to LAD (Ladder Logic) (→Properties → General → Language: LAD)

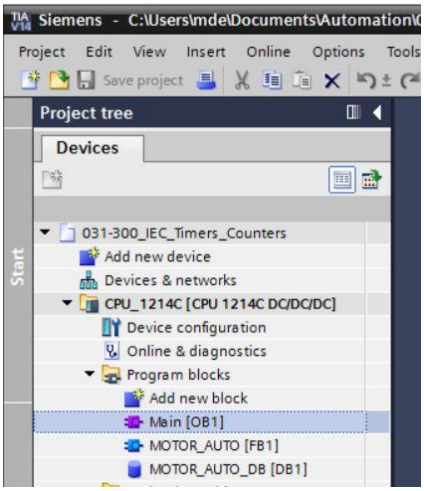


→ This is what networks 1 and 2 look like in LAD.

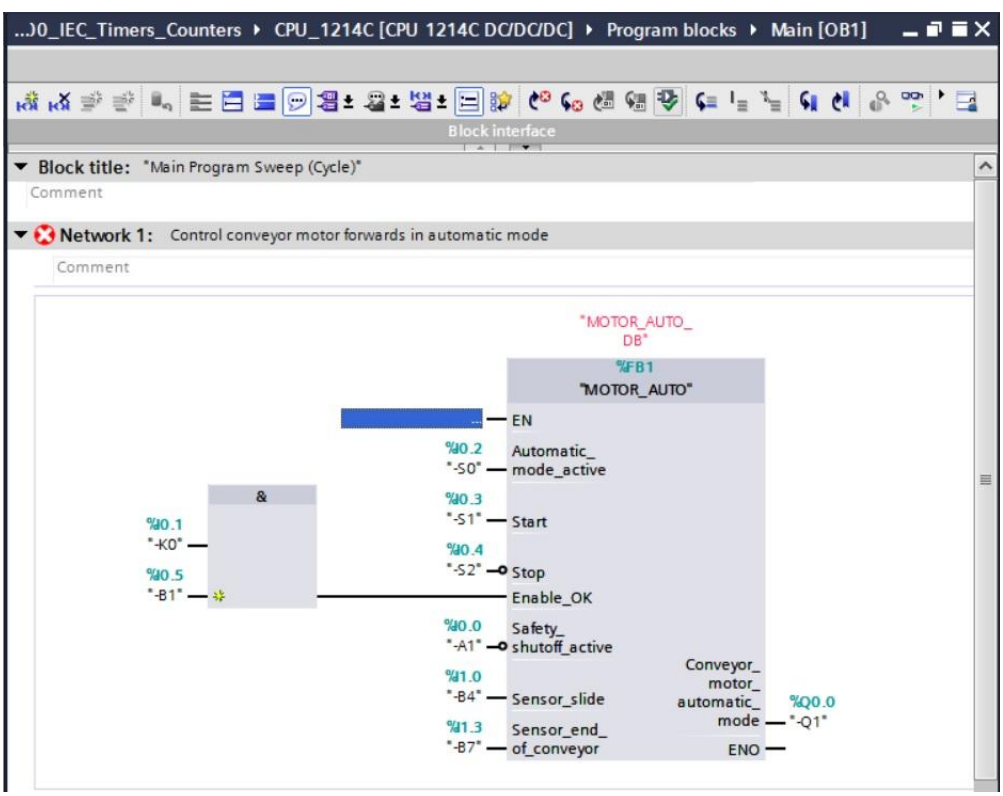


7.3 Update the block call in the organization block

→ Open the "Main [OB1]" organization block with a double-click.

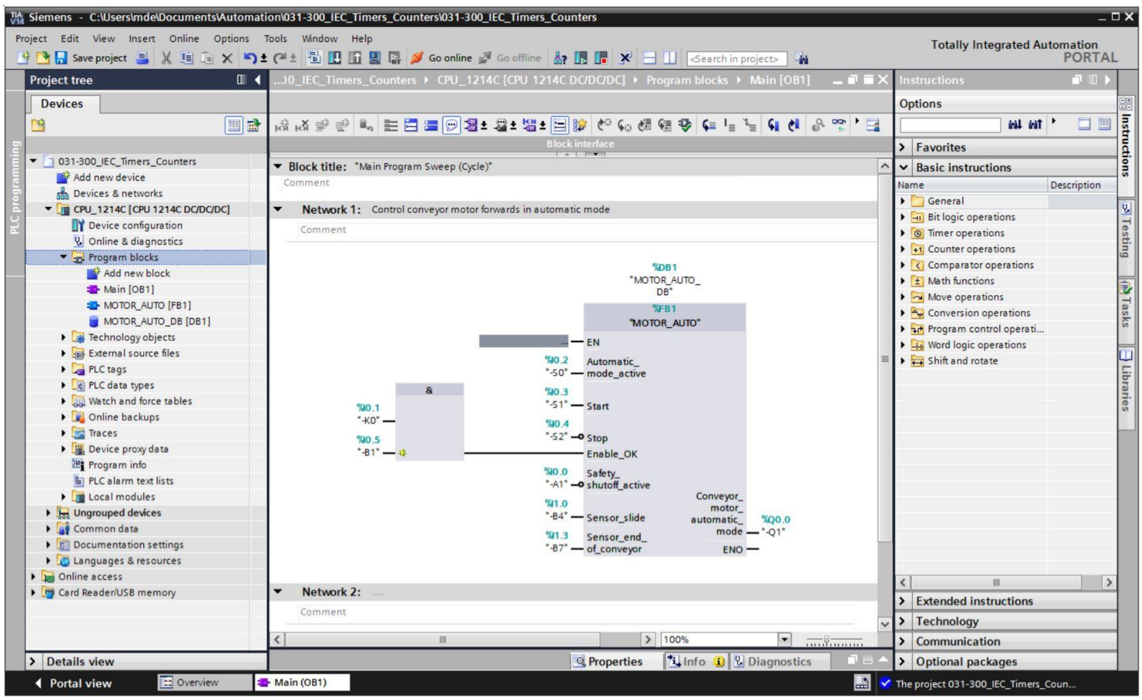


→ In Network 1 of the "Main [OB1]" organization block, instance data block "MOTOR_AUTO_DB1" for the "MOTOR_AUTO [FB1]" function block appears incorrect, because the additional memory for the TP Timer has not yet been added there. Click the → "" icon for "Update inconsistent block calls". This will add the "MOTOR_AUTO_DB1" instance data block correctly again (→).

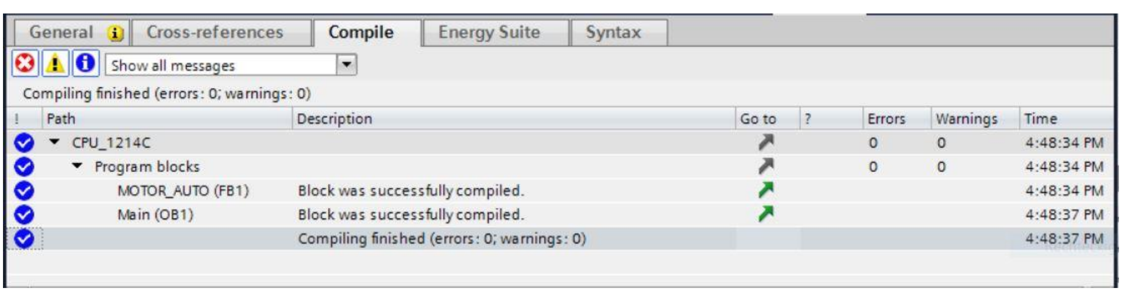


7.4 Save and compile the program

→ To save your project, select the Save project button in the menu. To compile all blocks, click the "Program blocks" folder and select the icon for compiling in the menu (→ Save project → Program blocks →).



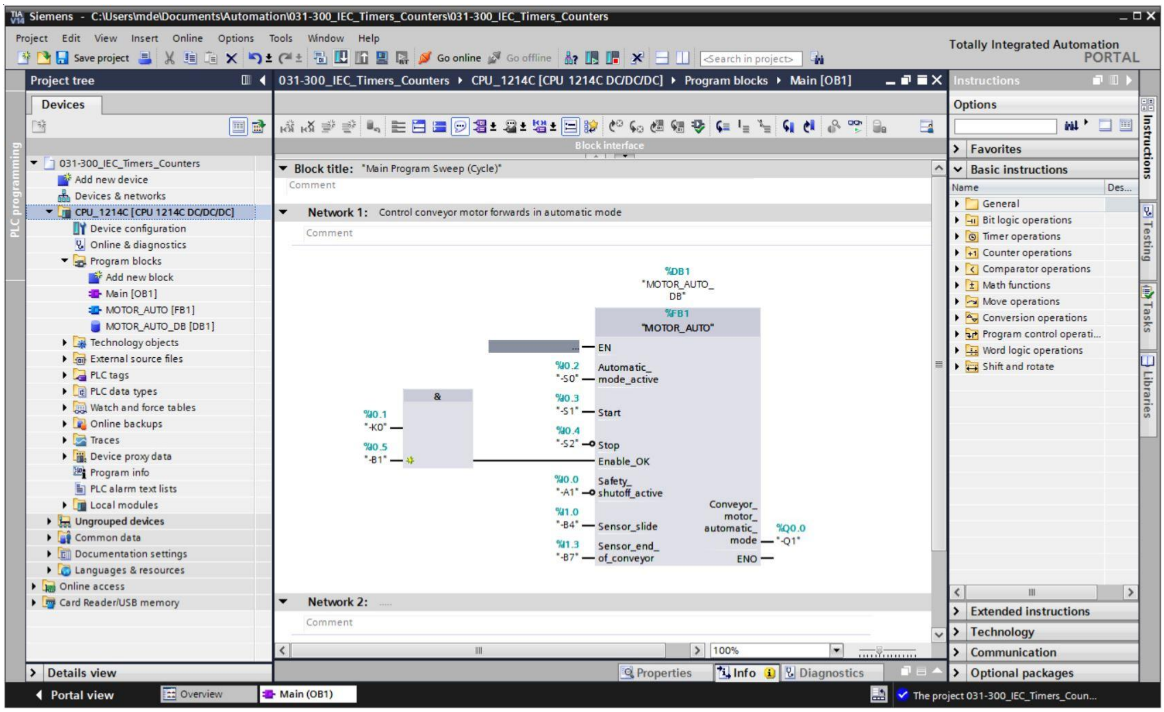
→ The "Info", "Compile" area shows which blocks were successfully compiled.



!	Path	Description	Go to	?	Errors	Warnings	Time
	CPU_1214C				0	0	4:48:34 PM
	Program blocks				0	0	4:48:34 PM
	MOTOR_AUTO (FB1)	Block was successfully compiled.					4:48:34 PM
	Main (OB1)	Block was successfully compiled.					4:48:37 PM
		Compiling finished (errors: 0; warnings: 0)					4:48:37 PM

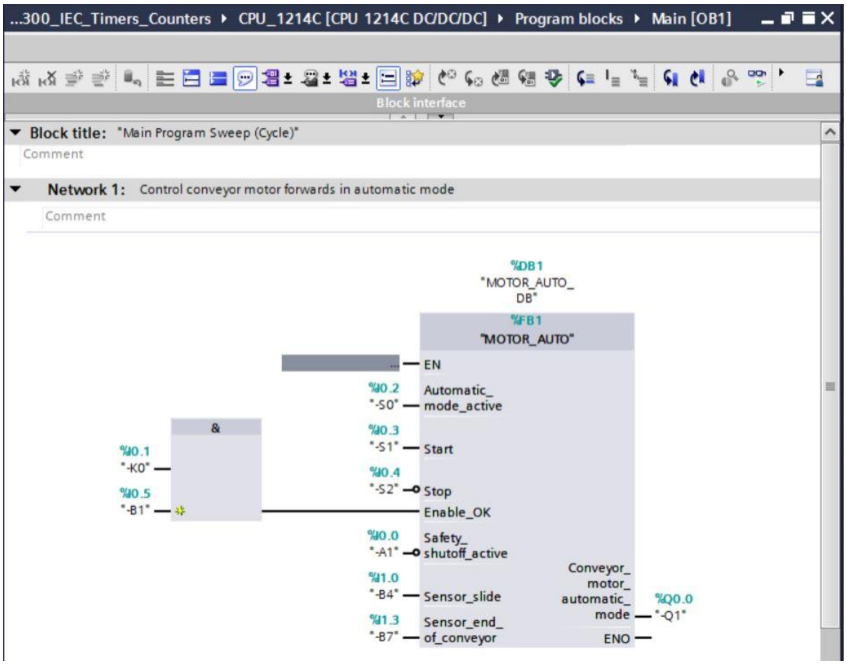
7.5 Download the program

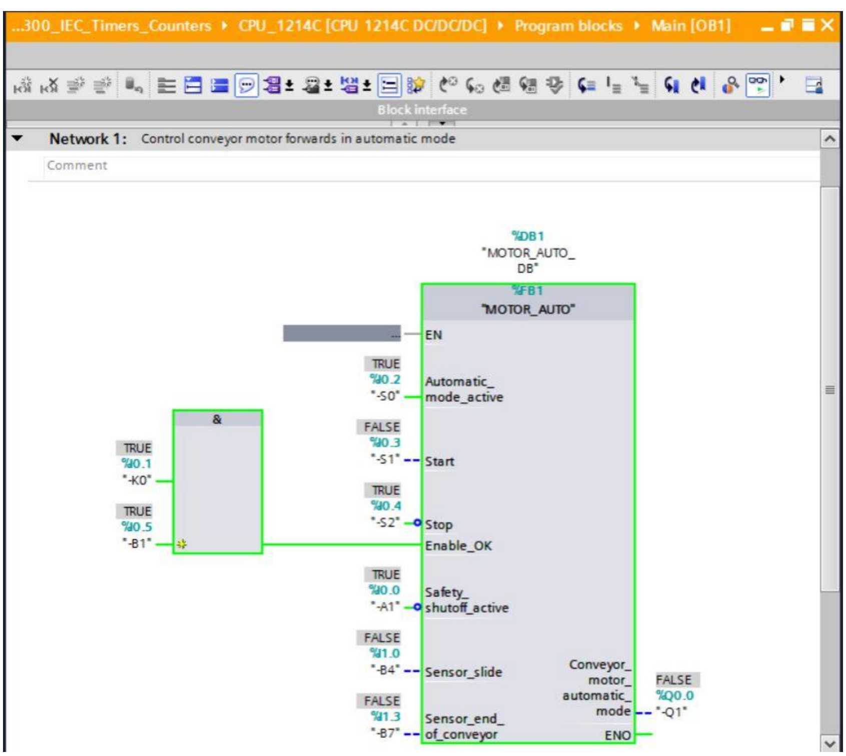
→ After successful compilation, the complete controller with the created program including the hardware configuration, as previously described in the modules, can be downloaded (→ ).



7.6 Monitor program blocks

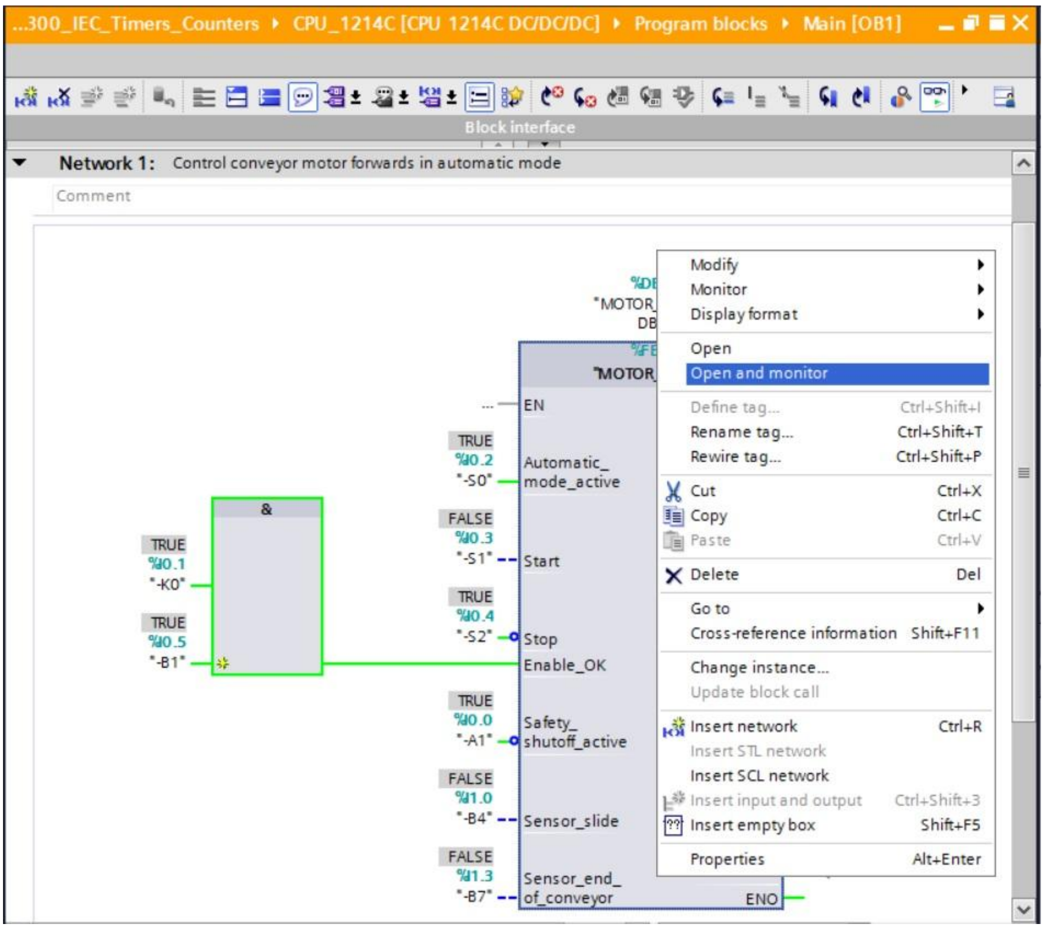
→ The desired block must be open for monitoring the downloaded program. Monitoring can now be activated/deactivated by clicking the  icon (→ Main [OB1] → ).

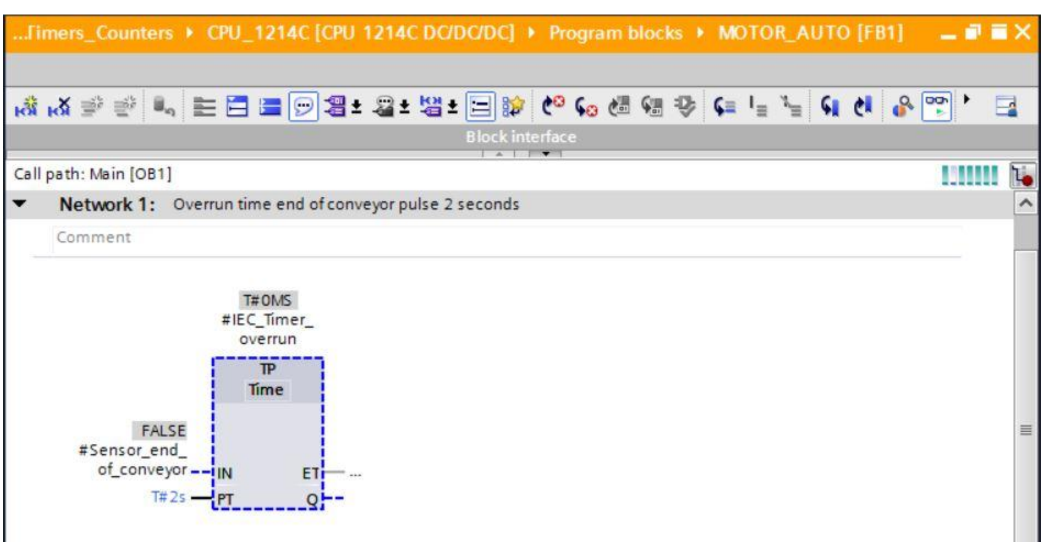




Note: *The monitoring here is signal-related and controller-dependent. The signal states at the terminals are indicated with TRUE or FALSE.*

→ The "MOTOR_AUTO" [FB1] function block called in the "Main [OB1]" organization block can be selected directly for "Open and monitor" after right-clicking, thereby allowing the program code in the function block with the TP Timer to be monitored
(→ "MOTOR_AUTO" [FB1] → Open and monitor).

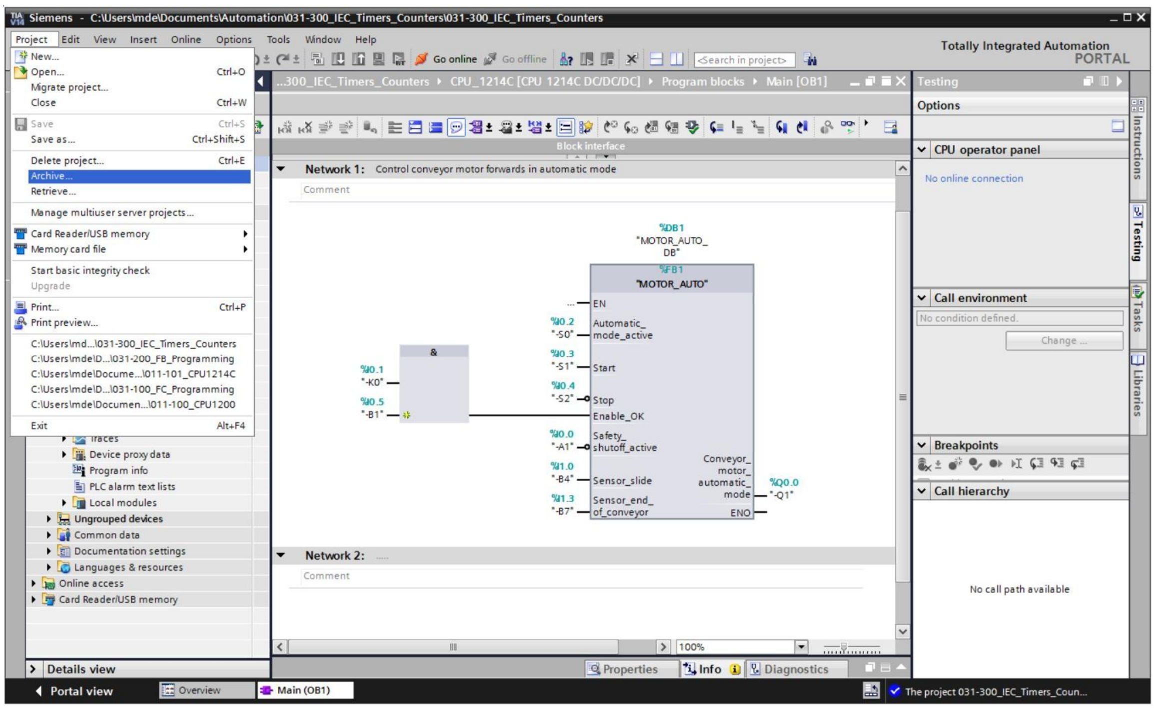




Note: *The monitoring here is function-related and controller-independent. The actuation of sensors and the station status are shown here with TRUE or FALSE.*

7.7 Archive the project

→ As the final step, we want to archive the complete project. Select the → "Archive ..." command in the → "Project" menu. Select a folder where you want to archive your project and save it with the file type "TIA Portal project archive". (→ Project → Archive → TIA Portal project archive → SCE_EN_031-300_IEC_Timers_Counters_S7-1200.... → Save)



7.8 Checklist

No.	Description	Completed
1	Compiling successful and without error message	
2	Download successful and without error message	
3	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated AUTOMATIC mode (-S0 = 1) Pushbutton automatic stop not actuated (-S2 = 1) Briefly press the automatic start pushbutton (-S1 = 1) Sensor at chute activated (-B4 = 1) Conveyor motor forwards fixed speed then switches on (-Q1 = 1) and stays on.	
4	Sensor at end of conveyor activated (-B7 = 1) → -Q1 = 0 (after 2 seconds)	
5	Briefly press the automatic stop pushbutton (-S2 = 0) → -Q1 = 0	
6	Activate EMERGENCY OFF (-A1 = 0) → -Q1 = 0	
7	Manual mode (-S0 = 0) → -Q1 = 0	
8	Switch off station (-K0 = 0) → -Q1 = 0	
9	Cylinder not retracted (-B1 = 0) → -Q1 = 0	
10	Project successfully archived	

7

8 Exercise

8.1 Task – Exercise

In this exercise, an IEC counter is to be added to the MOTOR_AUTO [FB1] function block. The expanded function block will be planned, programmed and tested:

The magazine for plastic holds only 5 parts. The parts are therefore be counted at the end of the conveyor.

When 5 parts are stored in the magazine, automatic mode is to be stopped.

Once the magazine has been emptied, automatic mode will be restarted with Start_command is started again and the counter is reset.

8.2 Technology diagram

Here, you see the technology diagram for the task.

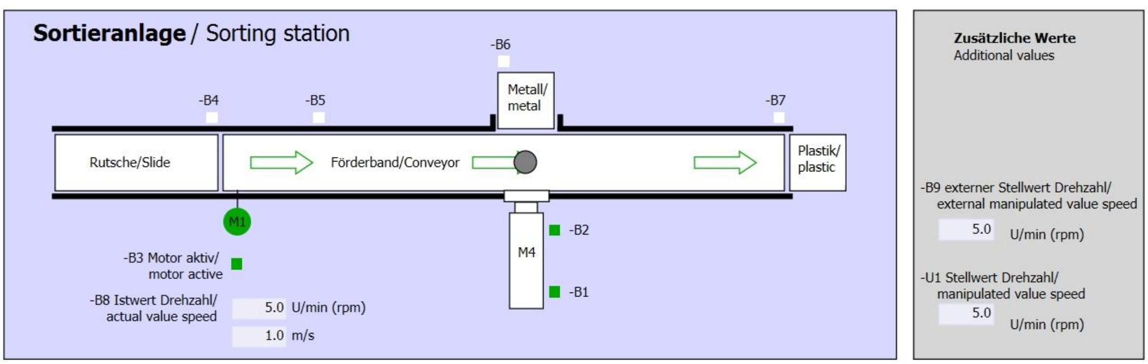


Figure 3: Technology diagram

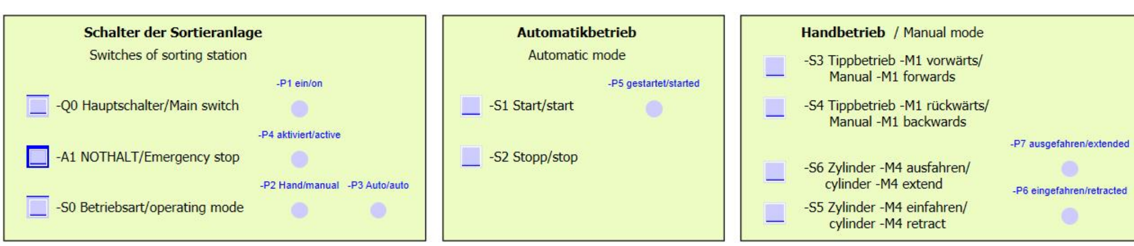


Figure 4: Control panel

8.3 Reference list

The following signals are needed as global operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop ok	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	-S1	Pushbutton automatic start	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder M4 retracted	NO
I 1.0	BOOL	-B4	Sensor at chute occupied	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO

DO	Type	Identifier	Function	
Q 0.0	BOOL	-Q1	Conveyor motor M1 forwards fixed speed	

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

8.4 Planning

Plan the implementation of the task on your own.

Note: *Learn about the use of IEC counters in SIMATIC S7-1200 in the online help.*

8.5 Checklist – Exercise

No.	Description	Completed
1	Compiling successful and without error message	
2	Download successful and without error message	
3	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated AUTOMATIC mode (-S0 = 1) Pushbutton automatic stop not actuated (-S2 = 1) Briefly press the automatic start pushbutton (-S1 = 1) Sensor at chute activated (-B4 = 1) Conveyor motor forwards fixed speed then switches on (-Q1 = 1) and stays on.	
4	Sensor at end of conveyor activated (-B7 = 1) → -Q1 = 0 (after 2 seconds)	
5	Briefly press the automatic stop pushbutton (-S2 = 0) → -Q1 = 0	
6	Activate EMERGENCY OFF (-A1 = 0) → -Q1 = 0	
7	Manual mode (-S0 = 0) → -Q1 = 0	
8	Switch off station (-K0 = 0) → -Q1 = 0	
9	Cylinder not retracted (-B1 = 0) → -Q1 = 0	
10	5th part in magazine → -Q1 = 0	
11	Project successfully archived	

9 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

Notes

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is available at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, contact your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

8

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

Table of contents

Basics of Diagnostic Functions

1 Goal

In this module, the reader will become acquainted with the tools that support troubleshooting.

This module will present diagnostic functions that, for example, you can test with the TIA project from the SCE_EN_031-100_FC-Programming with SIMATIC S7-1200 module.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

This chapter builds on the hardware configuration of SIMATIC S7 CPU1214C. However, other hardware configurations that have digital input and output boards can be used. For this chapter, you can use the following project, for example:

SCE_EN_031_100_FC-Programming_S7-1200_R1504.zap14

3 Required hardware and software

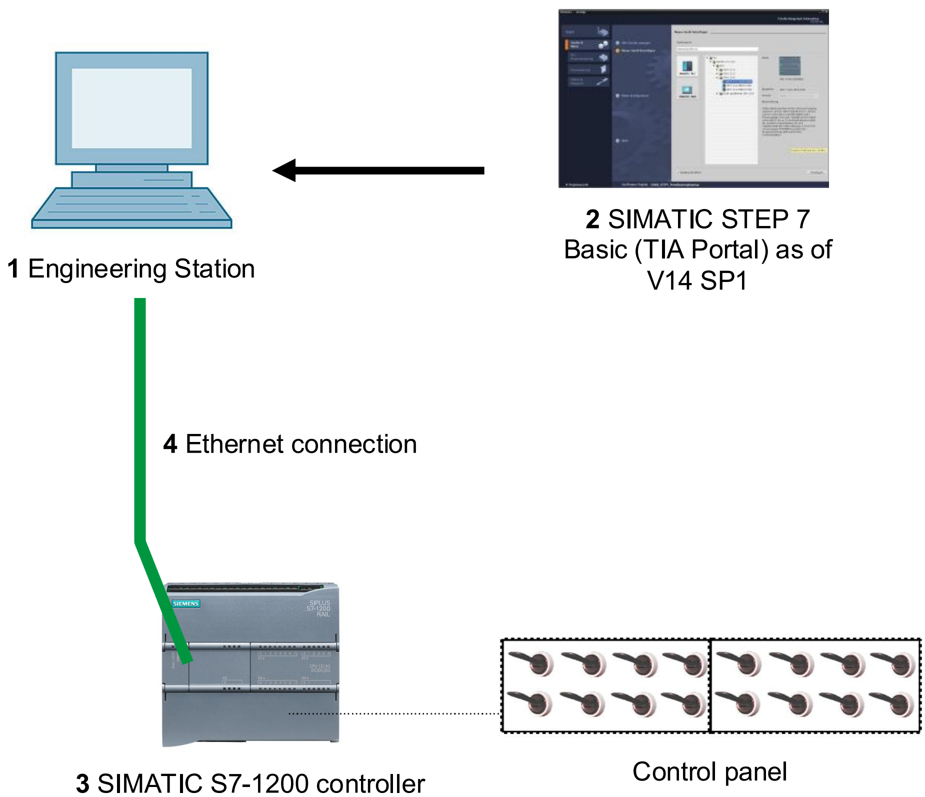
1 Engineering station: requirements include hardware and operating system (for additional information, see Readme on the TIA Portal Installation DVDs)

2 SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1

Note: The digital inputs should be fed out to a control panel.

4 Ethernet connection between engineering station and controller



4 Theory

4.1 Fault diagnostics and hardware faults

Faults can be caused by a variety of things.

For faults that occur after a changeover to RUN, there are two error patterns.

1. The CPU goes to or stays in the STOP operating state. The yellow STOP LED lights up and other indicator LEDs light up on the CPU, power supply unit, IO modules or bus modules.

   A CPU fault is present in this case. For example, a module in the automation system might be defective or have an incorrect parameter assignment or a bus system fault might be present.

   An interruption analysis will be performed in this case by evaluating the hardware diagnostics and by reading the module information from the diagnostic buffer of the CPU.

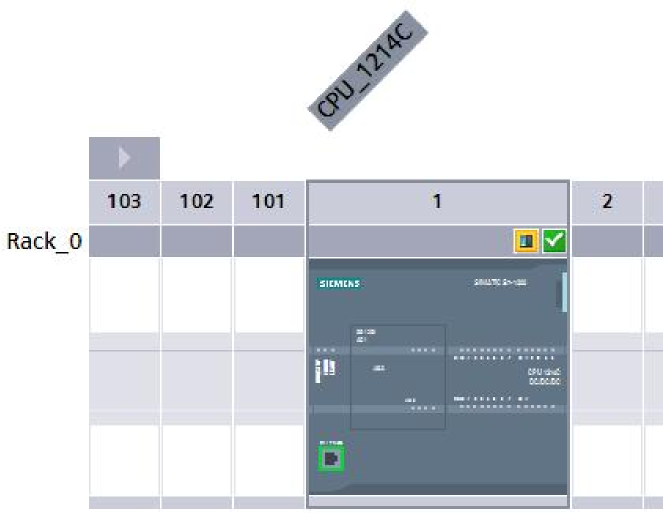
2. The CPU is in a faulty RUN operating state. The green RUN LED lights up and other indicator LEDs light up or flash on the CPU, power supply unit, IO modules or bus modules.

   In this case, a fault may be present in the IO devices or power supply.
   A visual check will be performed initially to narrow down the fault area. The indicator LEDs on the CPU and IO devices will be evaluated. The diagnostic data of the faulty IO and bus modules will be read from the hardware diagnostics. In addition, a fault analysis can be performed using a watch table on the programming device.

8

4.2 Hardware diagnostics

The device view in online mode of the TIA Portal gives you a quick overview of the configuration and system status of the automation system.



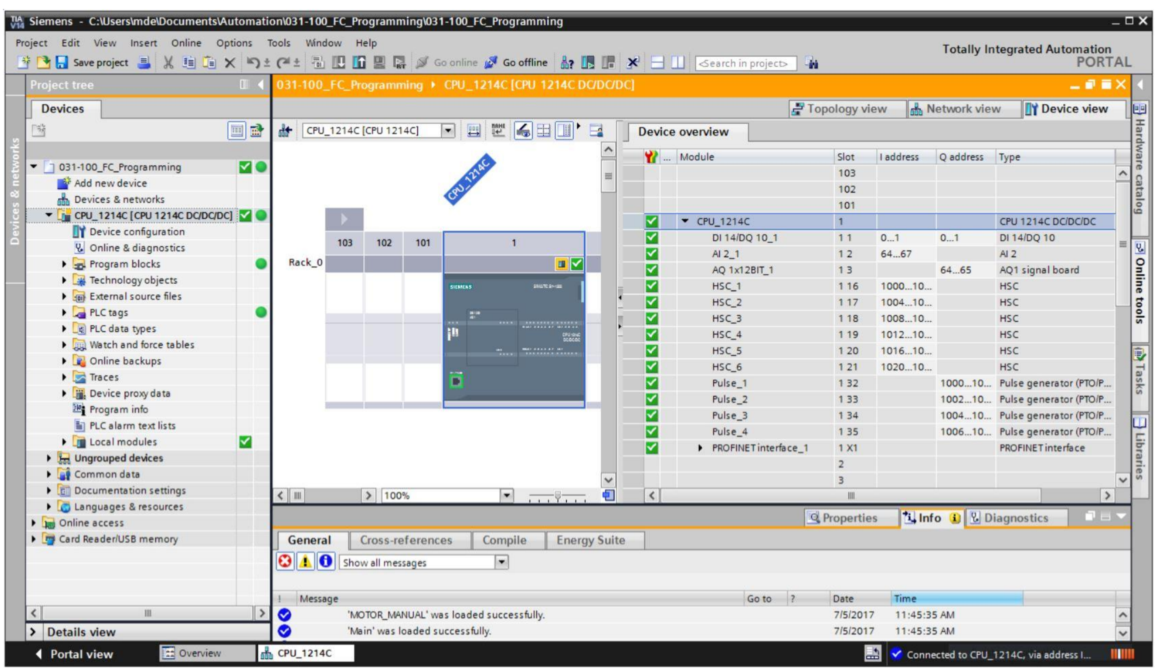


Figure 1: Online view of device configuration

4.3 Diagnostics for program blocks

The project tree window of the TIA Portal in online mode gives you an overview of the programmed blocks of the user program. A comparison of the program blocks used offline and online is displayed with the help of diagnostic symbols.

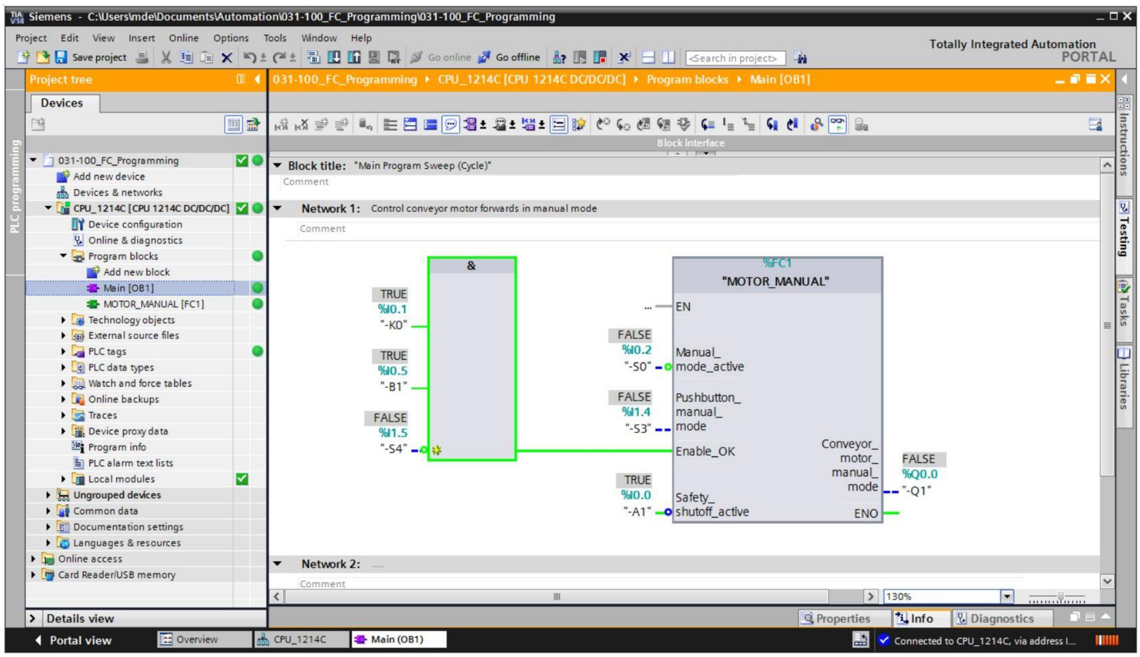


Figure 2: Online view of the Main [OB1] block

5 Task

The following diagnostic functions will be shown and tested in this chapter:

– Diagnostic symbols in the online view of the TIA Portal

– Device diagnostics with module information

– Offline/online comparison

– Monitoring and modifying tags

– Forcing tags

6 Planning

The diagnostic functions will be performed using a finished project as an example.

A project in the TIA Portal that was previously downloaded to the controller should be open for this.

In our case, once you have opened the TIA Portal, you will retrieve a previously created project that was archived and download it to the associated controller.

You can then start implementing the diagnostic functions in the TIA Portal.

6.1 Online interface

Online diagnostics can only be performed when the correct communication connection to the CPU has been established. We connect via Ethernet/PROFINET in this case.

When going online, you must therefore set the appropriate interfaces for your automation system.

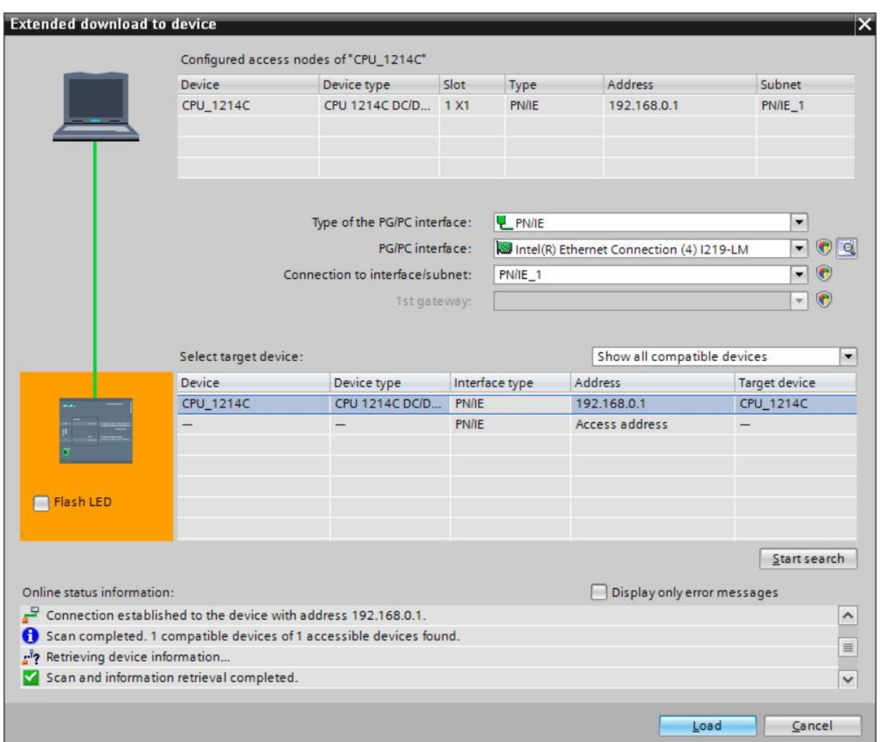


Figure 3: Connecting online

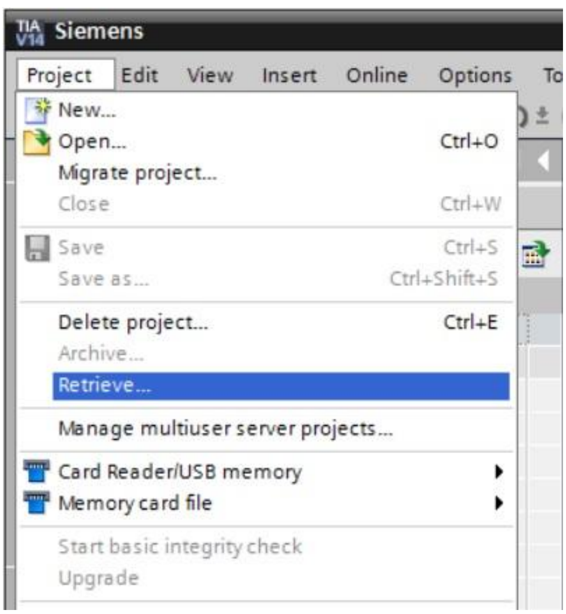
7 Structured step-by-step instructions

You will find instructions on how to carry out planning below. If you already have a good understanding of everything, it will be sufficient to focus on the numbered steps. Otherwise, simply follow the detailed steps in the instructions.

7.1 Retrieve an existing project

→ Before we can start the diagnostic functions, we need a project with programming and a hardware configuration (e.g., SCE_EN_031-100_FC-Programming_S7-1200....zap14). To retrieve an existing project that has been archived, you must select the relevant archive with → Project → Retrieve in the project view. Confirm your selection with "Open".

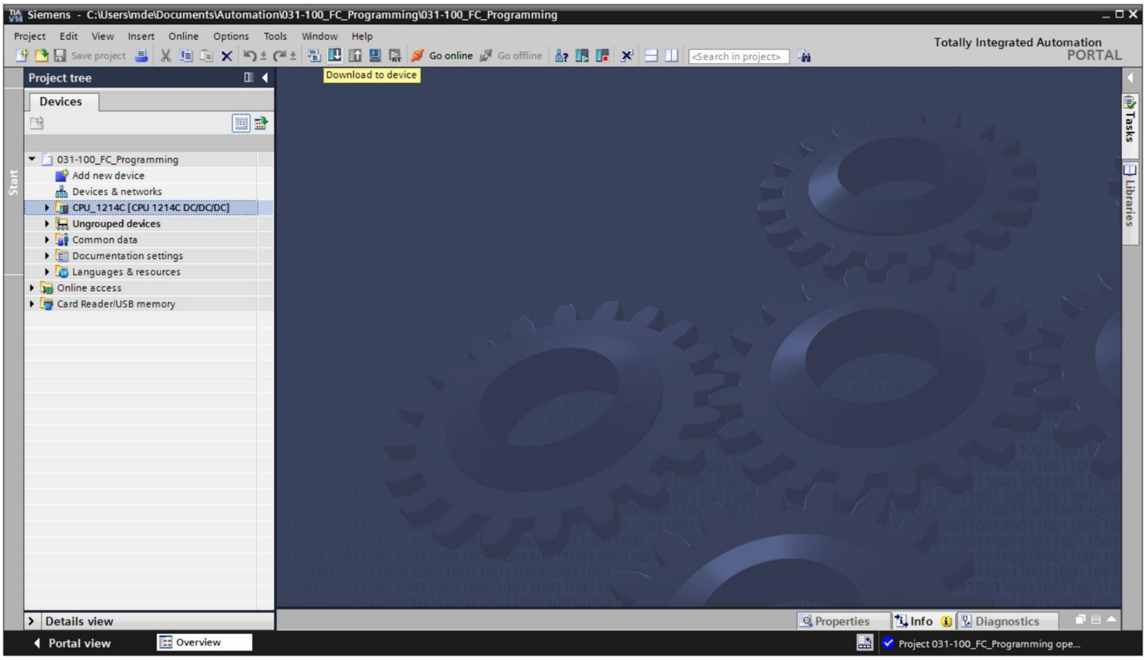
(→ Project → Retrieve → Select a .zap archive → Open)



→ The next step is to select the target directory where the retrieved project will be stored. Confirm your selection with "OK". (→ Target directory → OK)

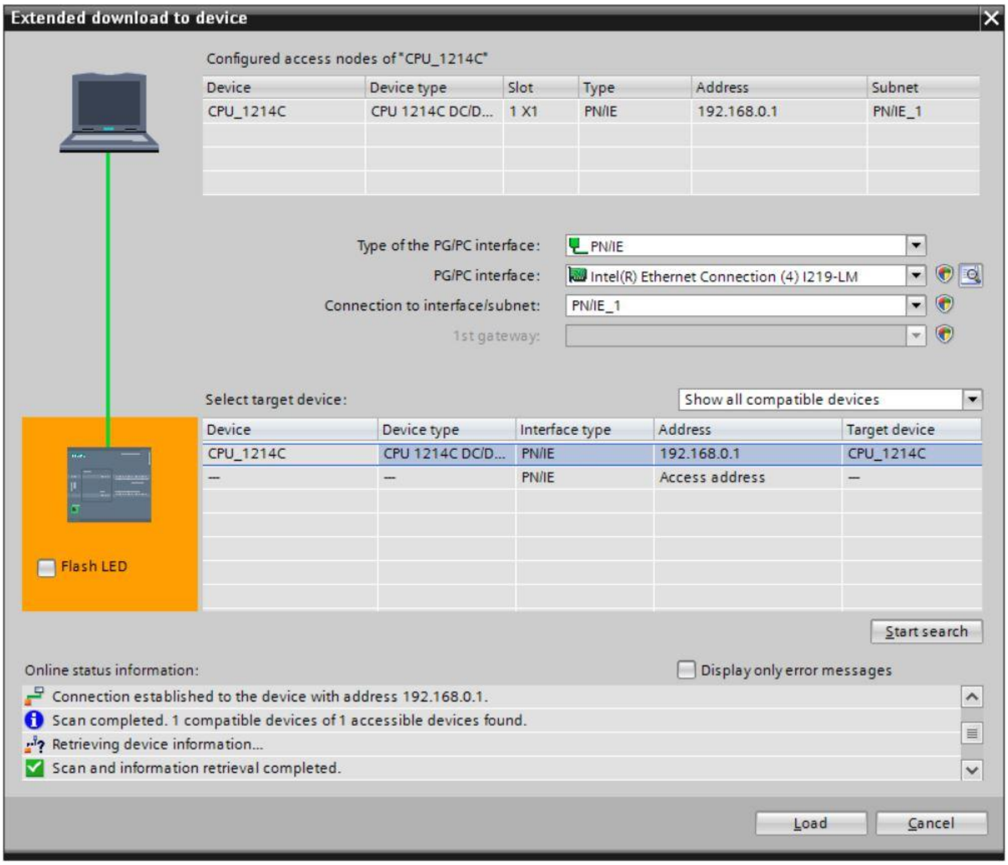
7.2 Download the program

→ After the project has been successfully retrieved, the controller can be selected and downloaded together with the created program. (→ )

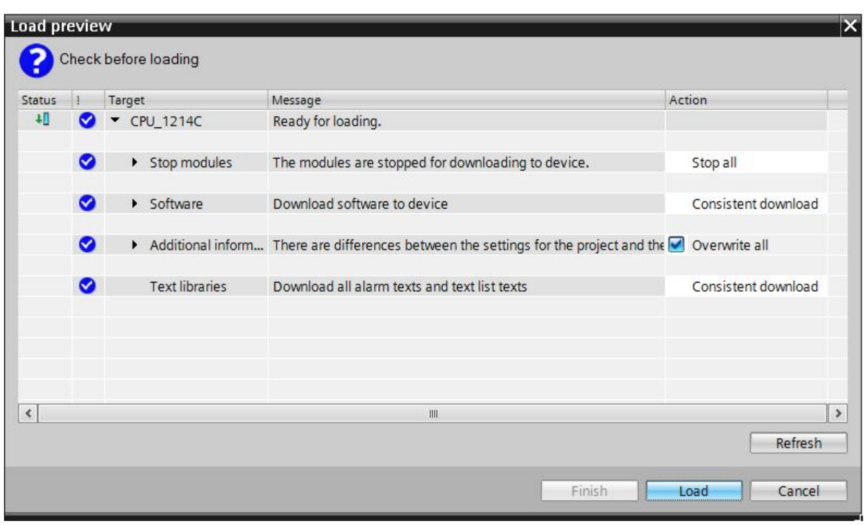


→ Select the correct interfaces and click "Start search". (→ "PN/IE" → Selection of the network adapter of the PG/PC → Direct at slot '1 X1'→ "Start search")

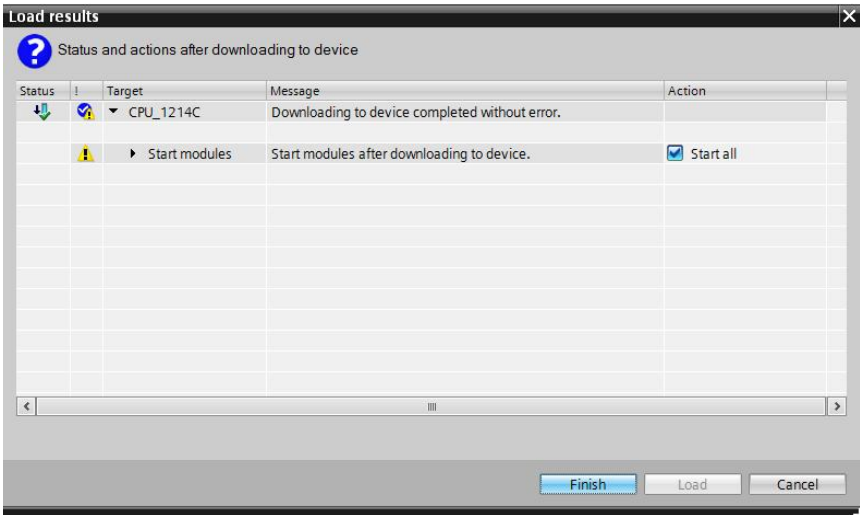
Once "Scan and information retrieval completed" appears, click "Load". (→ "Load")



→ Before downloading can be started, other actions may have to be set (pink marking). Then click "Load" again. (→ "Load").

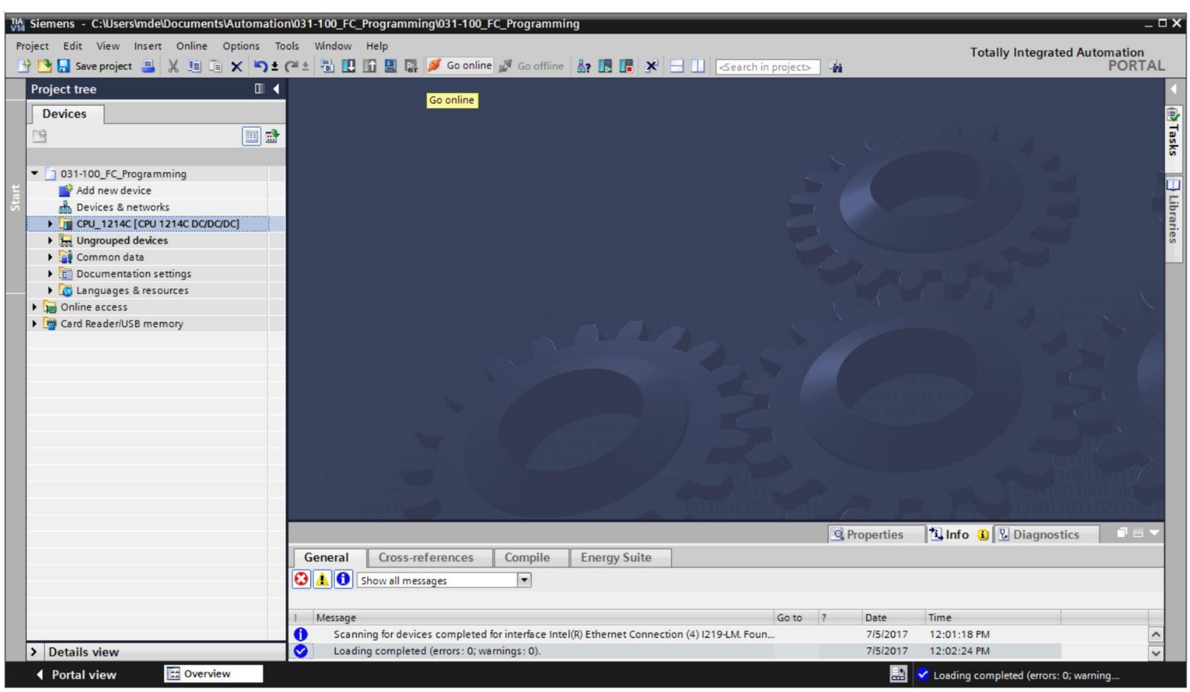


→ After loading, first select the "Start all" check box under Action. Then click "Finish". (→ select check box → "Finish")

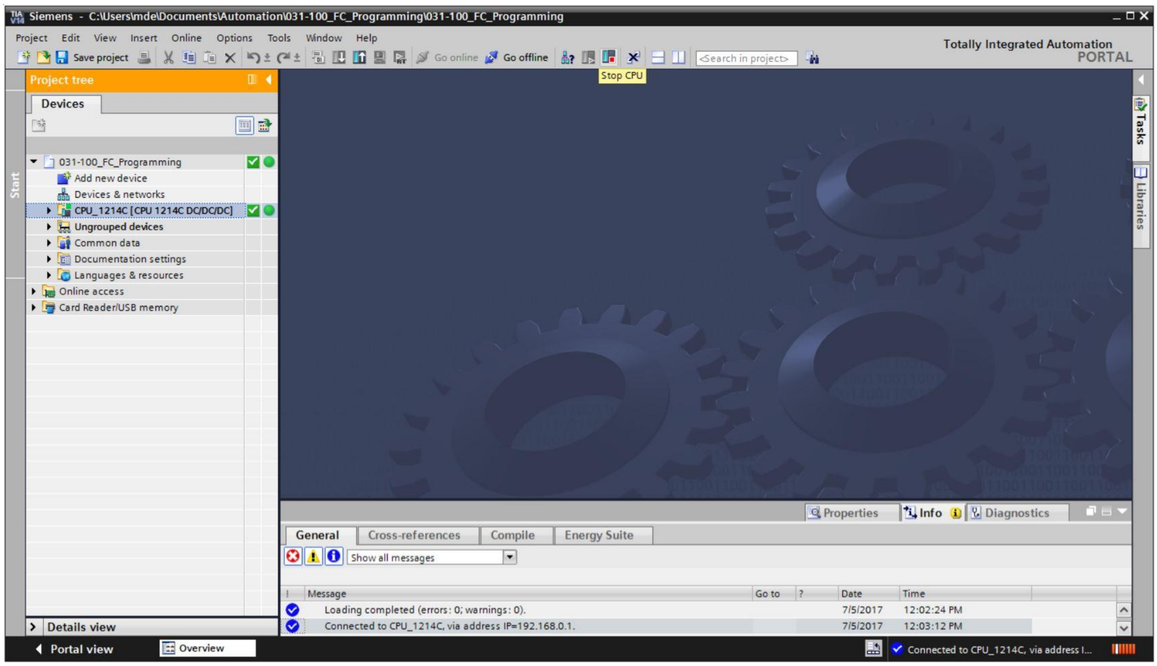


7.3 Connect online

→ To get started with the diagnostic functions, we will select our controller ("CPU_1214C") and click "Go online". (→ CPU_1214C → Go online)



→ Once the online connection to the "PLC_1" controller is established, the CPU can be started or stopped with the following buttons . Diagnostic information in the form of symbols will already be available in the project tree and in the diagnostics window.



8

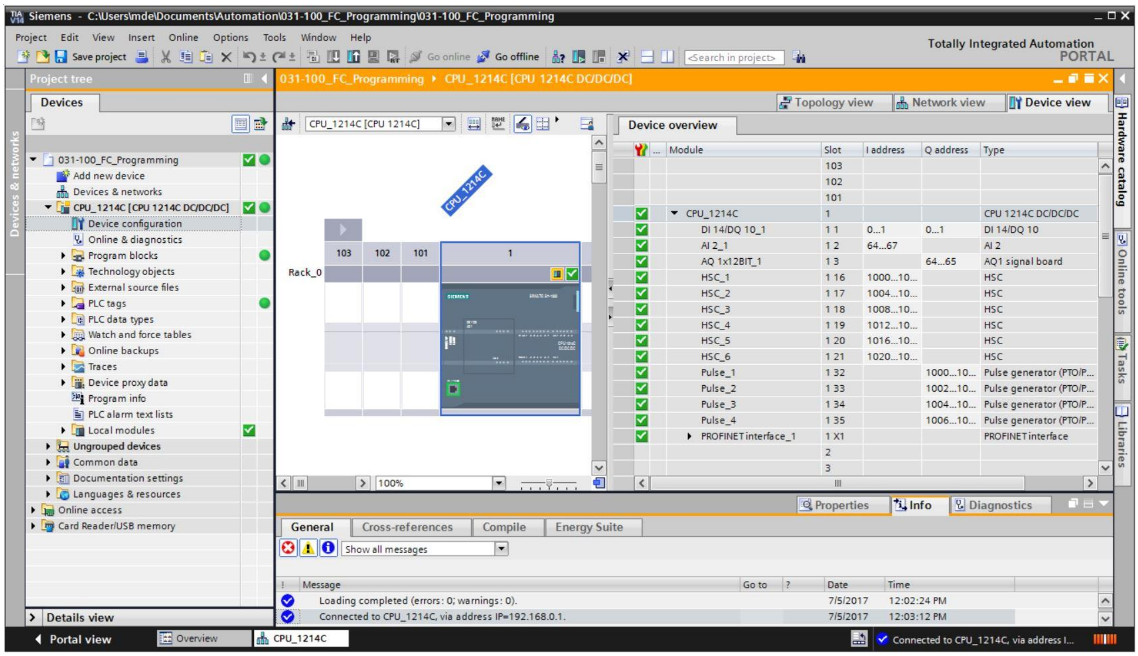
Symbols for the comparison status in the project tree

→ The diagnostic symbols in the project tree show a comparison status representing the online/offline comparison of the project structure.

Symbol	Meaning
	Folder contains objects with online and offline versions that different (only in the project tree)
	Online and offline versions of the object are different
	Object only exists online
	Object only exists offline
	Online and offline versions of the object are the same

→ Double-click the "Device configuration".

(→ Device configuration)



Operating state symbols for CPUs and CPs

→ The graphical representation and device information window show the various operating states of the CPU or communication processors (CPs).

Symbol	Operating state
	RUN
	STOP
	STARTUP
	HOLD
	DEFECT
	Unknown operating state
	The configured module does not support display of the operating state.

8

Diagnostic symbols for modules and devices in the device overview

→ The graphical representation and Device overview window show the operating states of the various modules, CPU or communication processors (CPs) using the following symbols.

Symbol	Meaning
	The connection to a CPU is currently being established.
	The CPU is not accessible at the configured address.
	The type of CPU configured and type of CPU actually present are incompatible.
	On establishment of the online connection to a protected CPU, the password dialog was terminated without entry of the correct password.
	No fault
	Maintenance required
	Maintenance demanded
	Fault
	The module or device is deactivated.
	The module or device cannot be accessed from the CPU (valid for modules and devices below a CPU).
	Diagnostic data is not available because the current online configuration data differs from the offline configuration data.
	The configured module or device and the module or device actually present are incompatible (valid for modules or devices below a CPU).
	The configured module does not support display of the diagnostic status (valid for modules below a CPU).
	The connection has been established, but the state of the module is currently still being determined.
	The configured module does not support display of the diagnostic status.
	Error in lower-level component: A fault is present in at least one lower-level hardware component.

Color coding of ports and Ethernet cables

→ The status of ports and Ethernet cables can be diagnosed in the network view and topology view.

→ The following table shows the possible colors and their respective meaning.

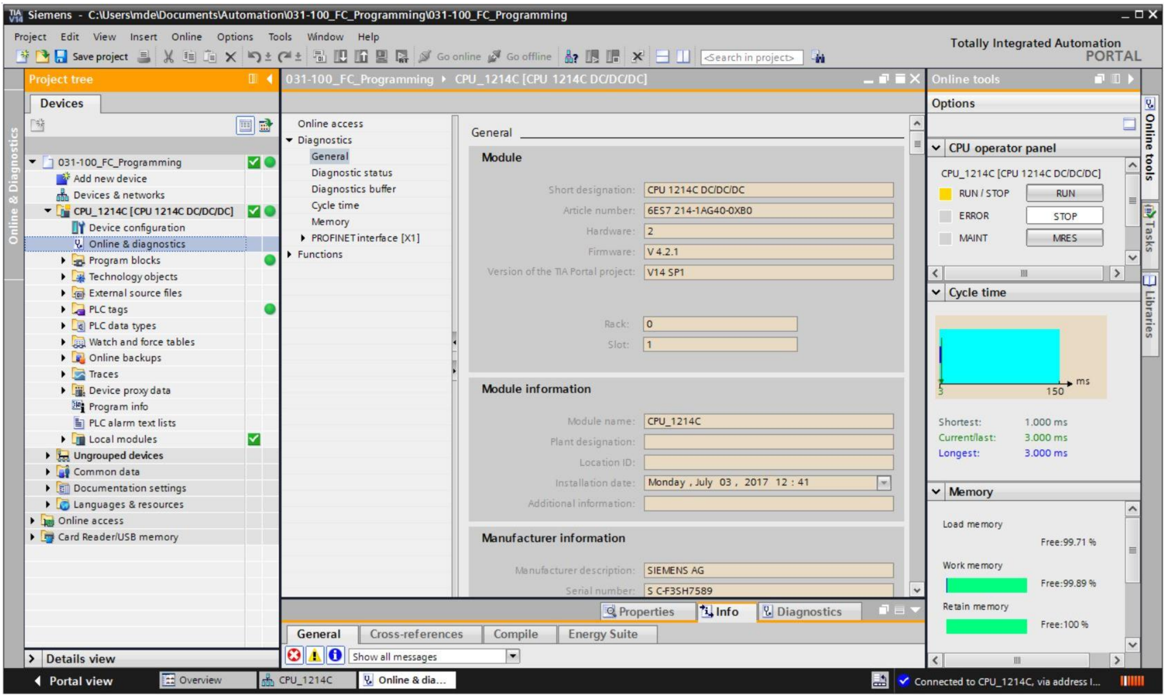
Color	Meaning
(green)	No fault or maintenance required
(yellow)	Maintenance demanded
(red)	Communication error

8

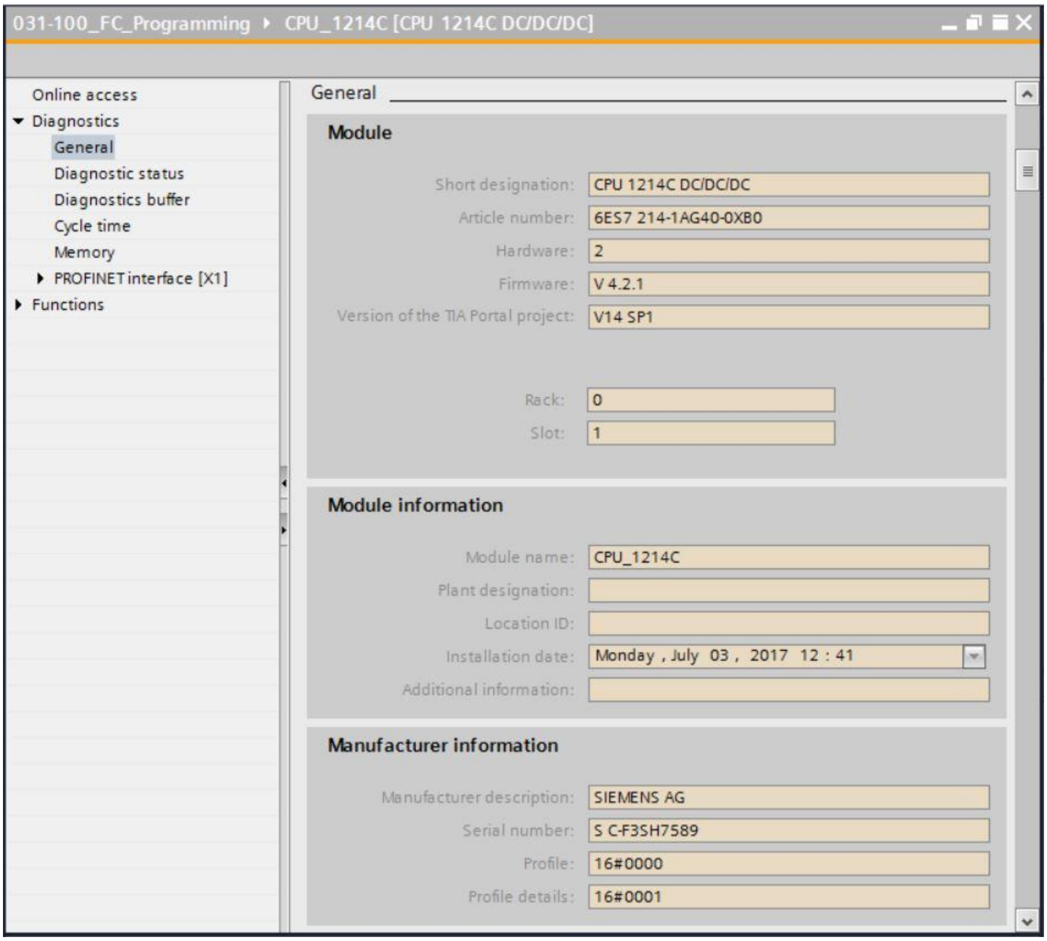
7.4 Online & diagnostics for SIMATIC S7 controller

→ Double-click "Online & diagnostics" in project tree. (→ Online&Diagnostics)

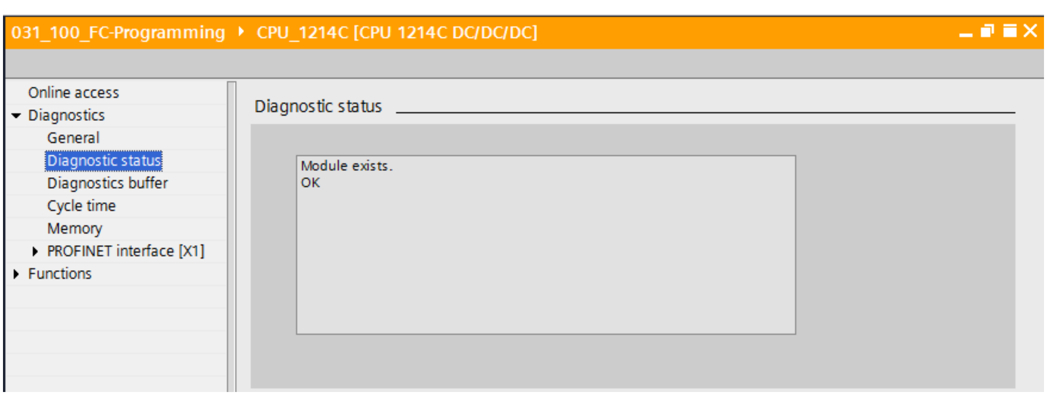
→ A CPU operating panel, the cycle time and the memory utilization are displayed in the online tools at the right. Switch the CPU to RUN here. (→ RUN)



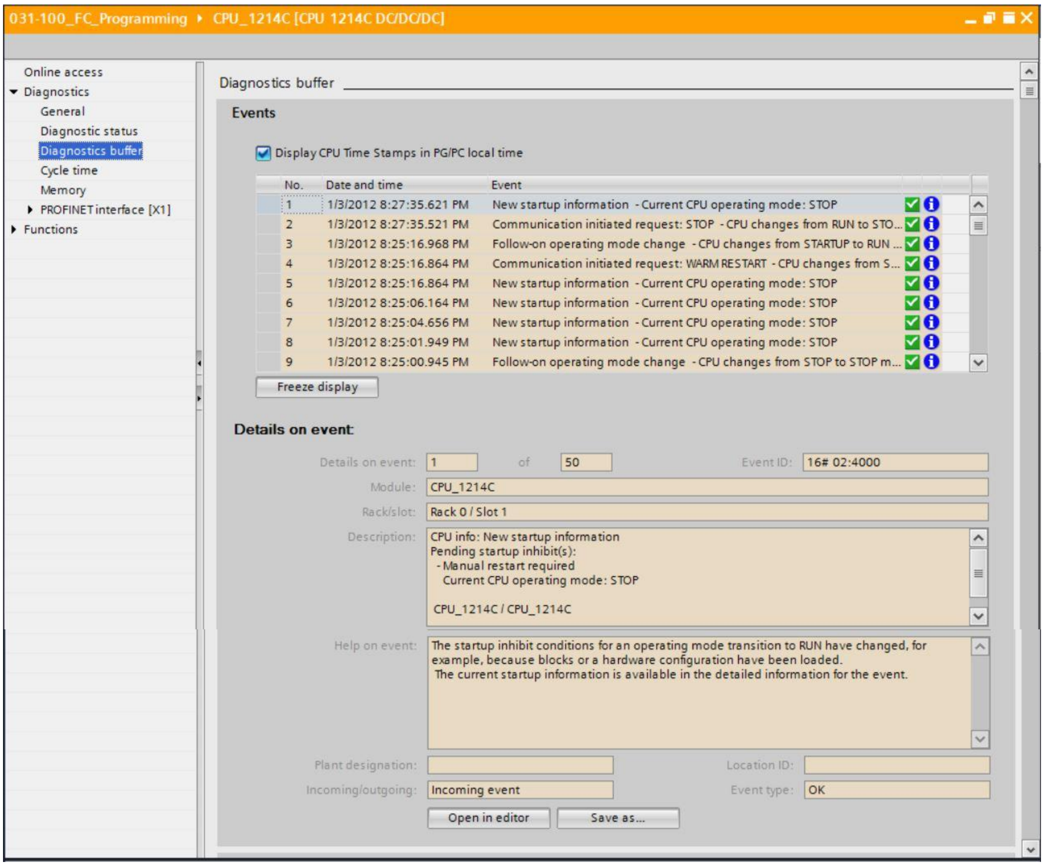
→ The working area window contains general information about the CPU. (→ General)



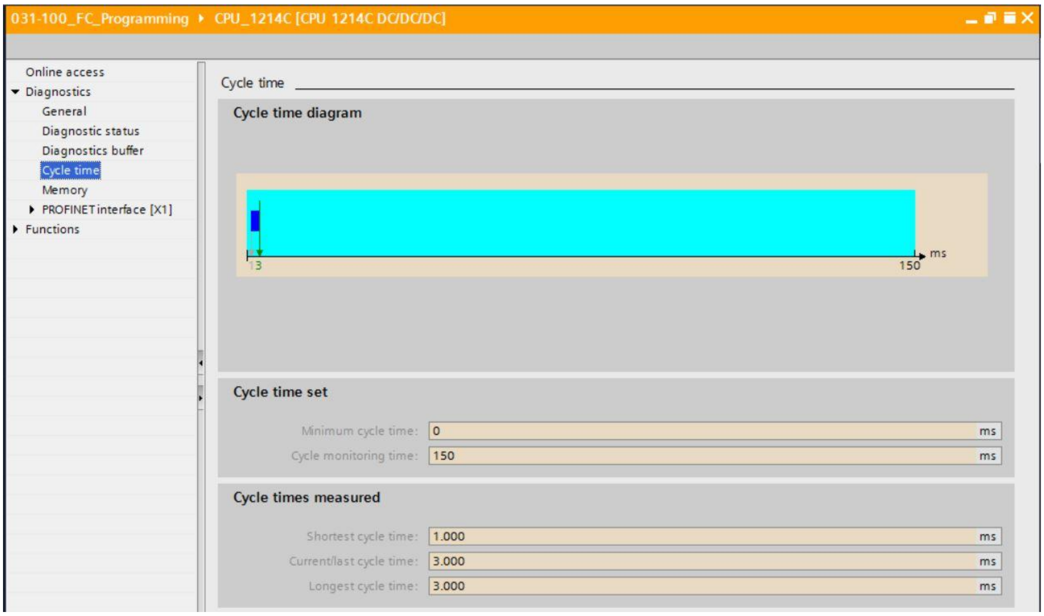
→ If diagnostic information is available, it is displayed in Diagnostic status. (→ Diagnostic status)



→ Detailed Information on the individual events is displayed in Diagnostics buffer. (→ Diagnostics buffer)

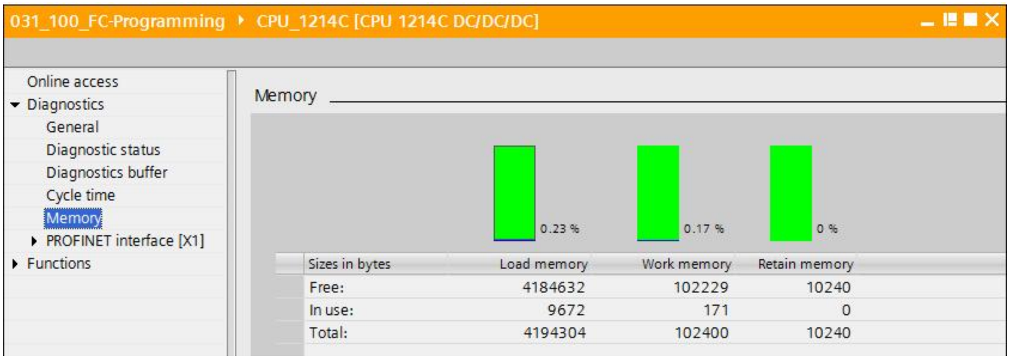


→ Next you receive information about the cycle time of the executed program. (→ Cycle time)

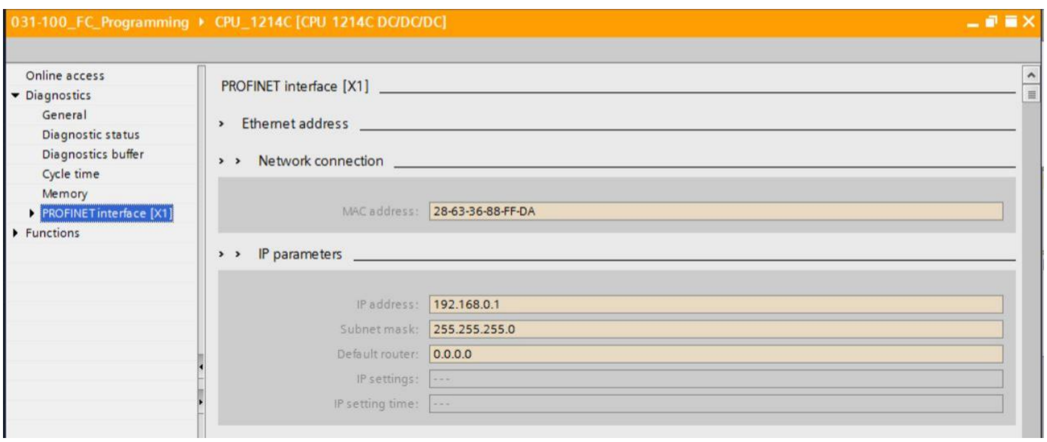


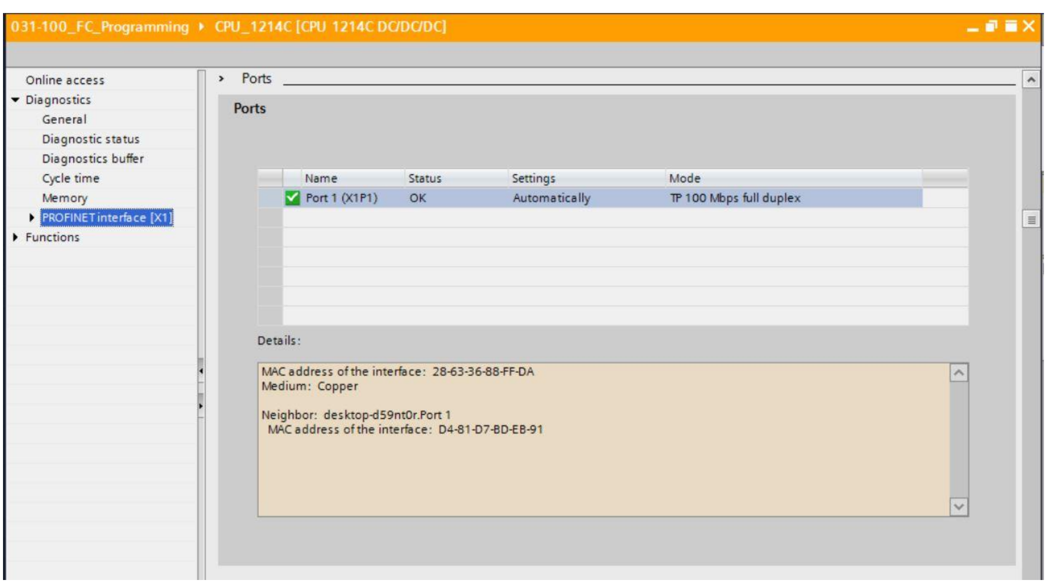
8

→ The memory utilization can be seen here in detail (→ Memory)



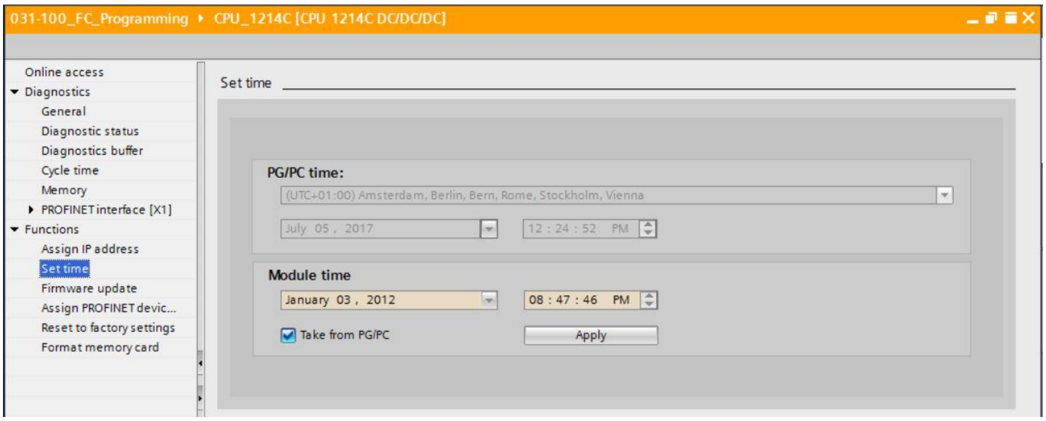
→ The network settings and the status of the PROFINET interface [X1] can also be displayed. (→ PROFINET interface [X1])



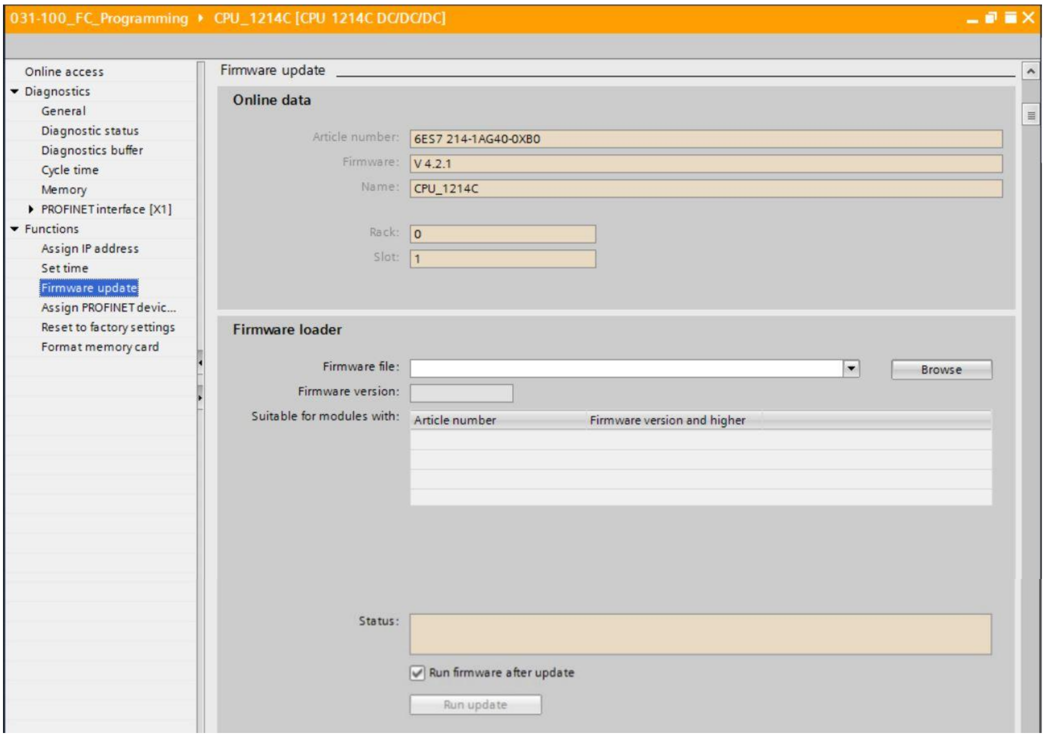


→ Under Functions, "Assign IP address", you can assign the IP address to a controller. However, this is only possible when no hardware has yet been downloaded to the CPU. (→ Functions → Assign IP address)

→ Under "Set time", you can set the time of the CPU. (→ Functions → Set time)



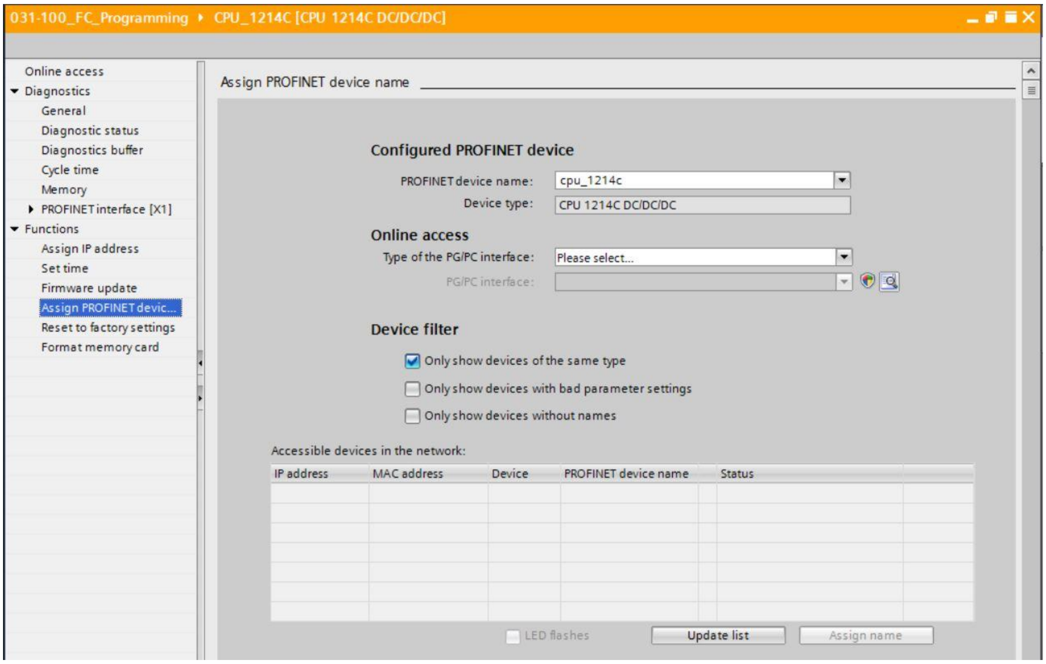
→ Under "Firmware update", you can update the firmware of the PLC. (→ Functions → Firmware update)



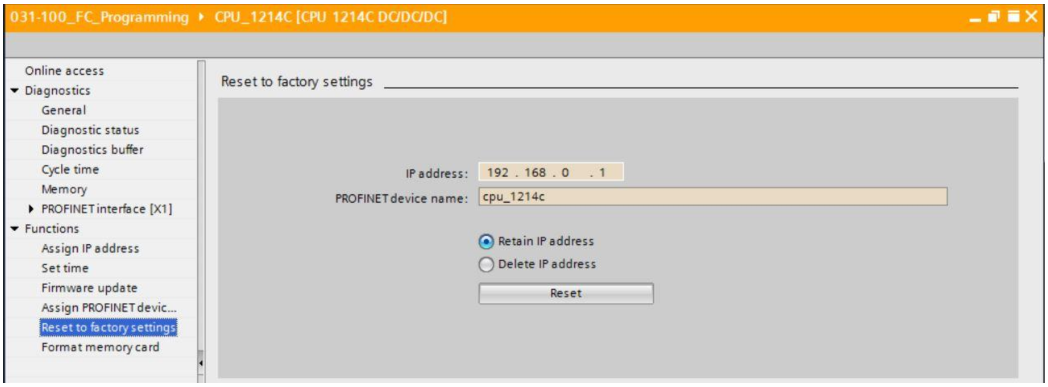
8

→ Under "Assign name", you can assign a PROFINET device name to the configured field devices on PROFINET. The device name of the CPU cannot be changed here. It can only be changed by downloading a modified hardware configuration.

(→ Functions → Assign name)

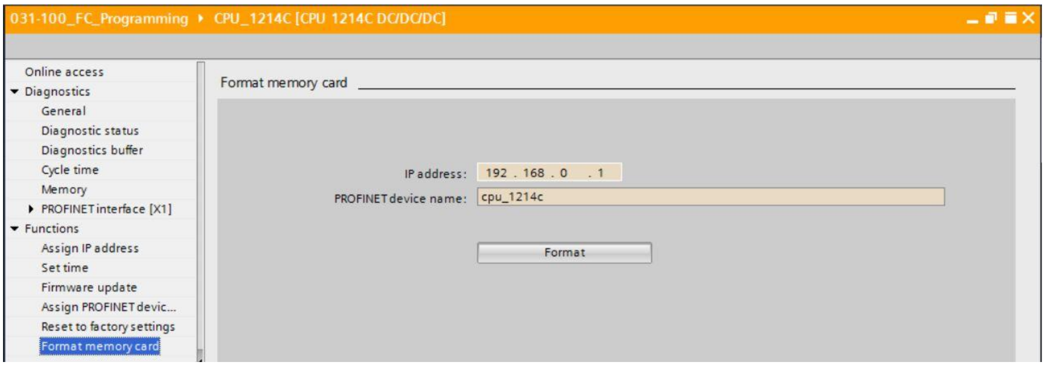


→ Under "Reset to factory settings", you can restore the factory settings of the CPU. (→Functions → Reset to factory settings → Retain or delete IP address → Reset)

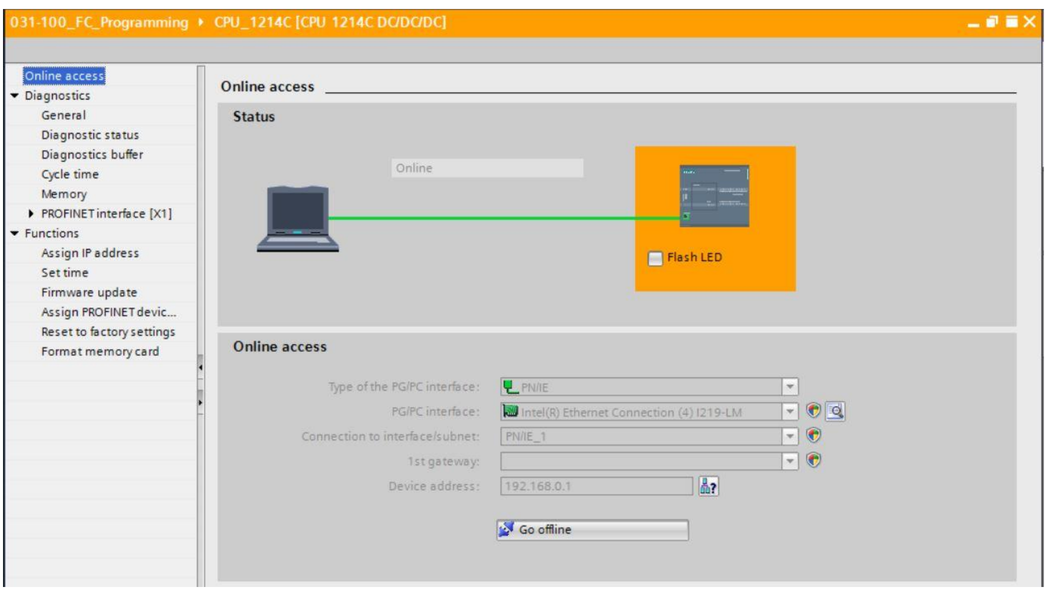


8

→ Under "Format memory card", you can format the optional memory card if it is inserted in the CPU. (→Functions → Format memory card → Format)



→ The online connection should be disconnected again before the next chapter. (→ Online access → Go offline)



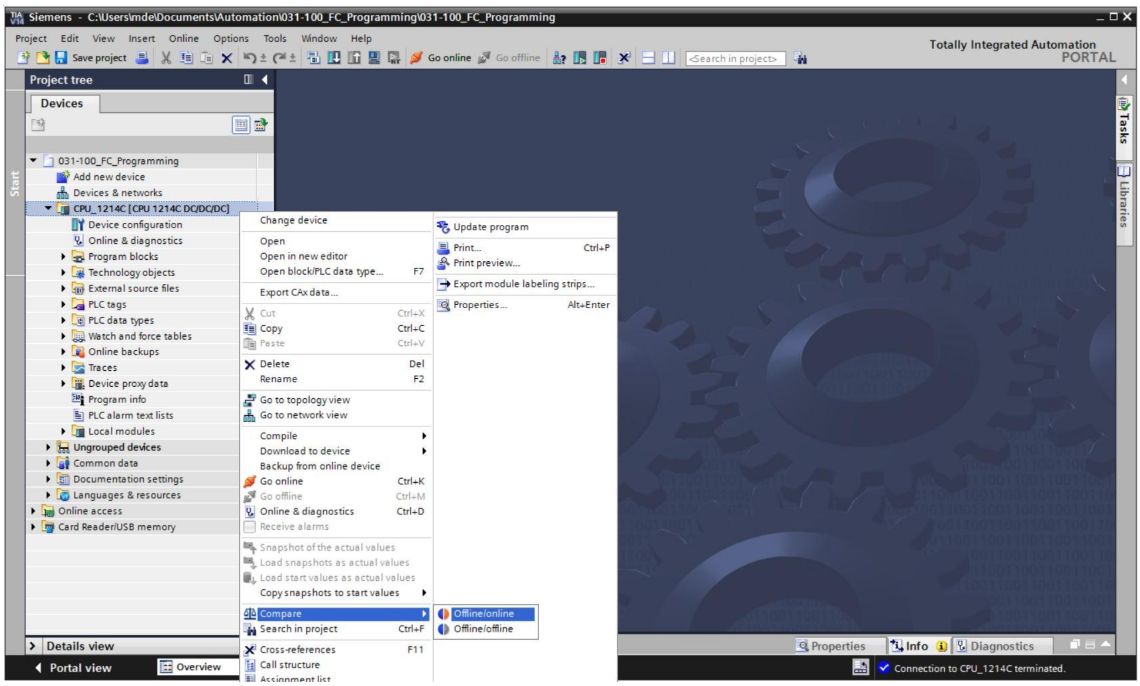
→ The TIA Portal is now back in offline mode. The orange-colored bars and the diagnostic symbols are no longer displayed.

7.5 Online/offline comparison

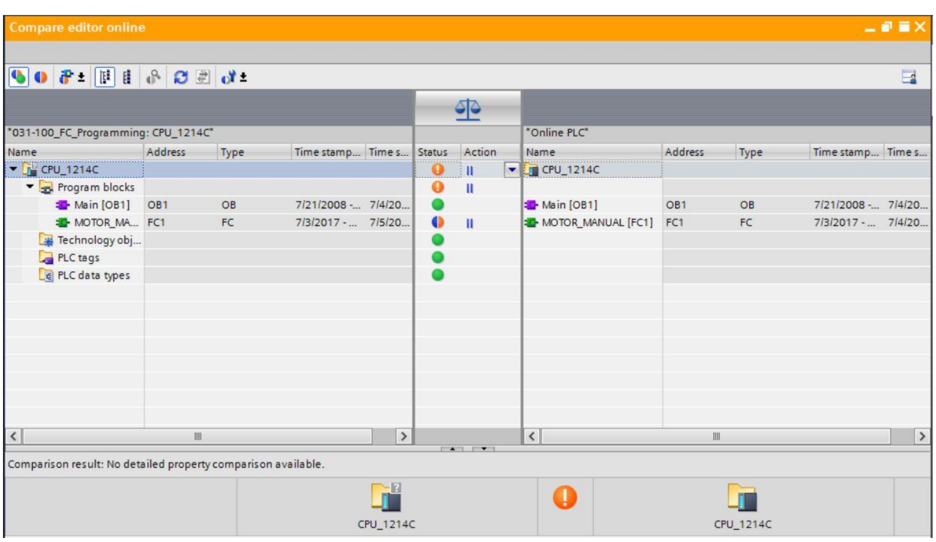
→ It is often important to know whether the saved data matches the data loaded in the controller. First, remove the negation from the "Safety_shutoff_active" tag at the AND function in the "MOTOR_MANUAL [FC1] block.

Then save the "MOTOR_MANUAL [FC1]" block, but do **NOT** download it to the controller. Close the "MOTOR_MANUAL [FC1] block again.

→ To compare, right-click the "PLC_1" controller and select "Compare", "Offline/online". (→ Select controller → Compare → Offline/online)



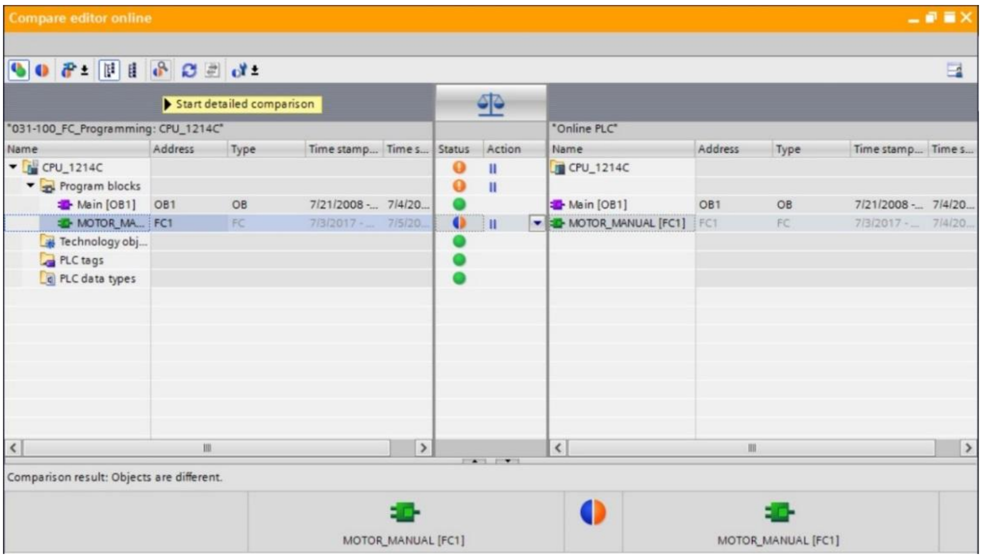
→ The Compare editor online opens.



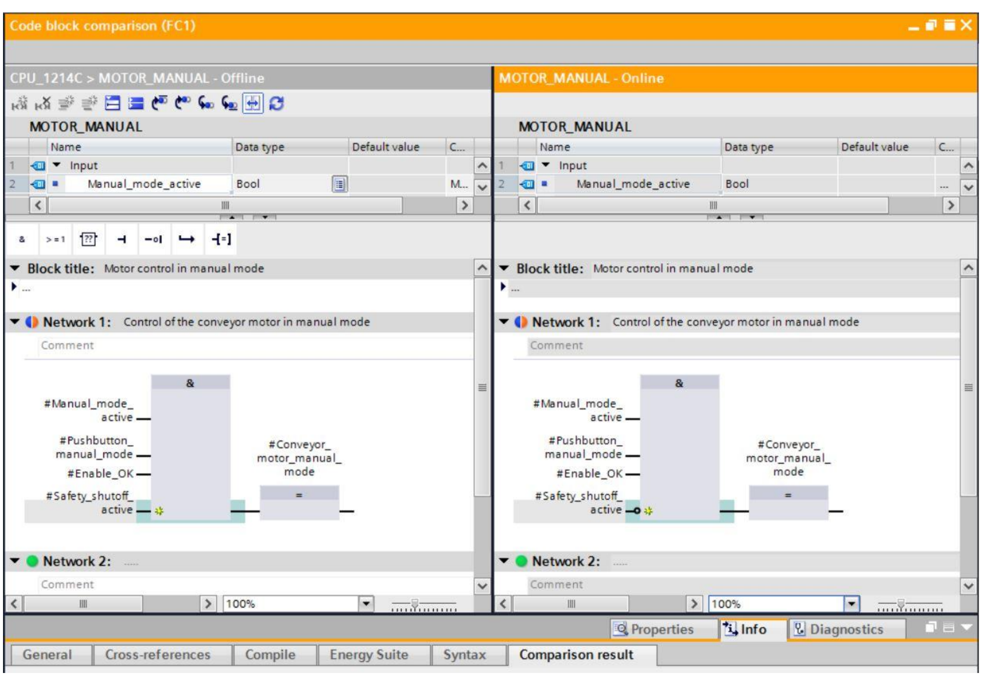
8

→ If, for example, block differences are indicated, first select the block involved. You can then click the button to "Start detailed comparison".

(→ MOTOR_MANUAL → Start detailed comparison).



→ The selected offline/online block will be compared in the code block comparison. A detailed description of the difference is shown in the comparison result.

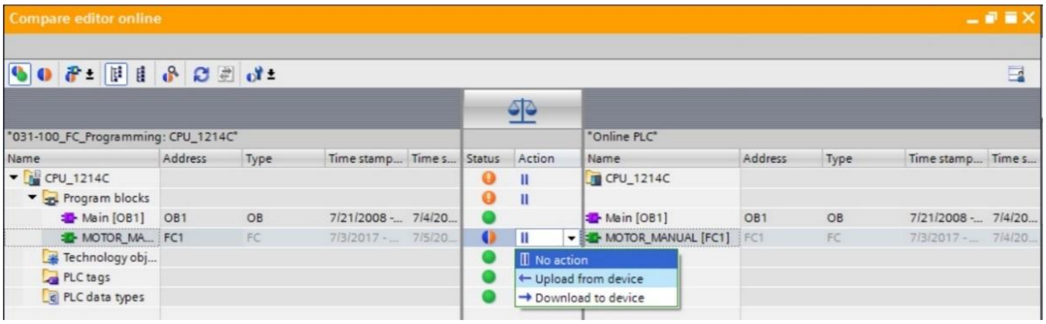


→ Close the window of the code block comparison.

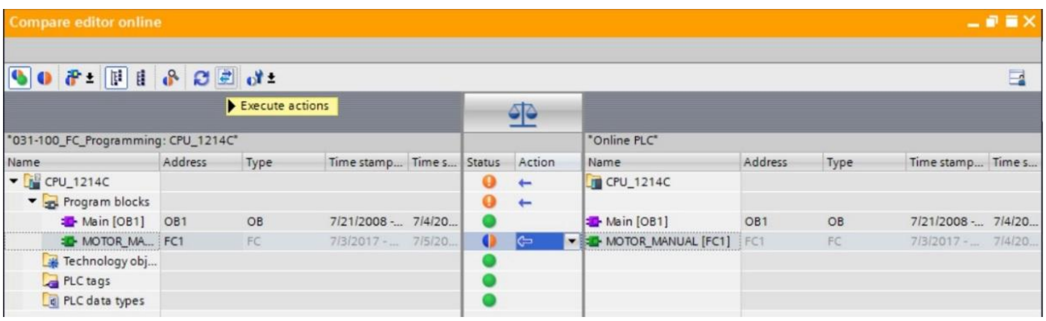
An action can be selected for the block involved in the Compare editor.

Either the "MOTOR_MANUAL" block will be downloaded from the programming device to the controller and overwritten there or the "MOTOR_MANUAL" block will be read in from the controller and overwritten in the TIA Portal.

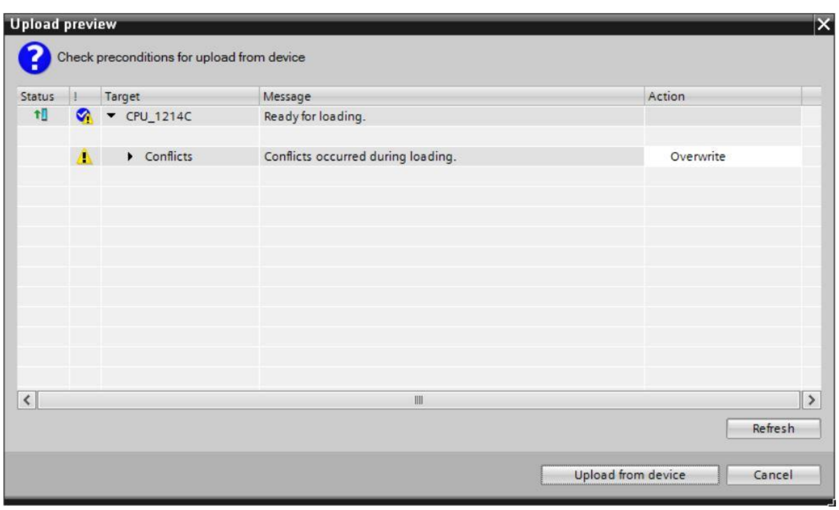
Select the "Upload from device" action (← Upload from device).



→ Click the "Execute actions" button (→ Execute actions)



→ Confirm "Upload from device" (→ Upload from device).



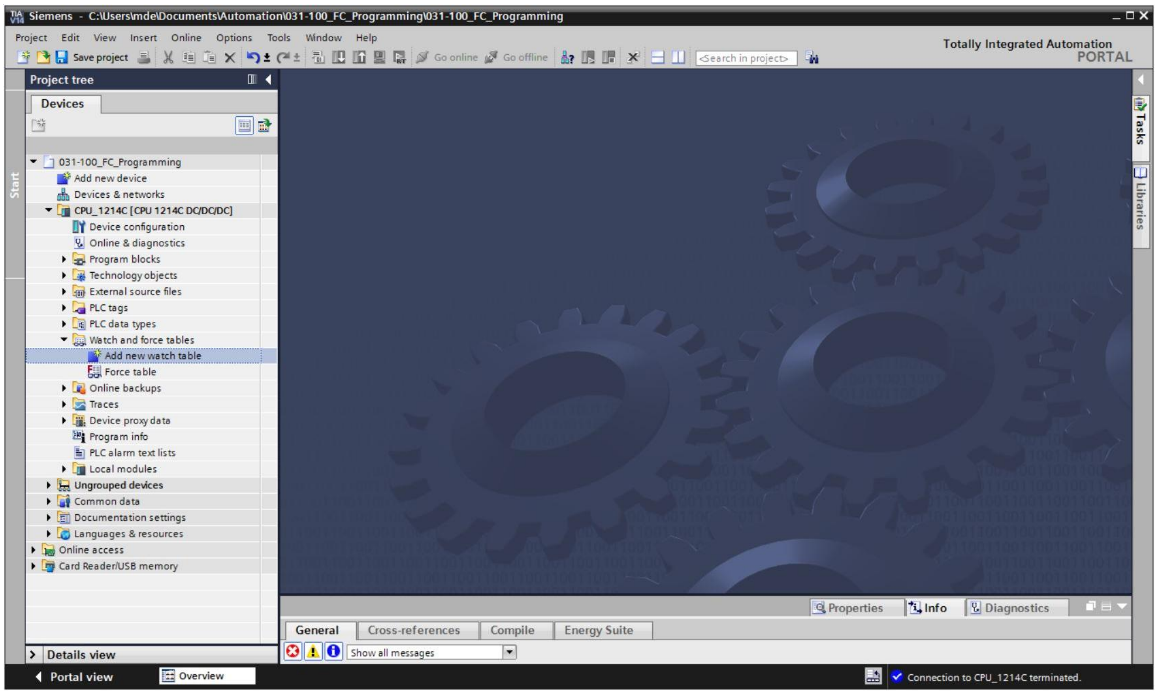
→ After the upload, there are no more differences. You should now save your project again and close the online connection.

8

7.6 Monitor and modify tags

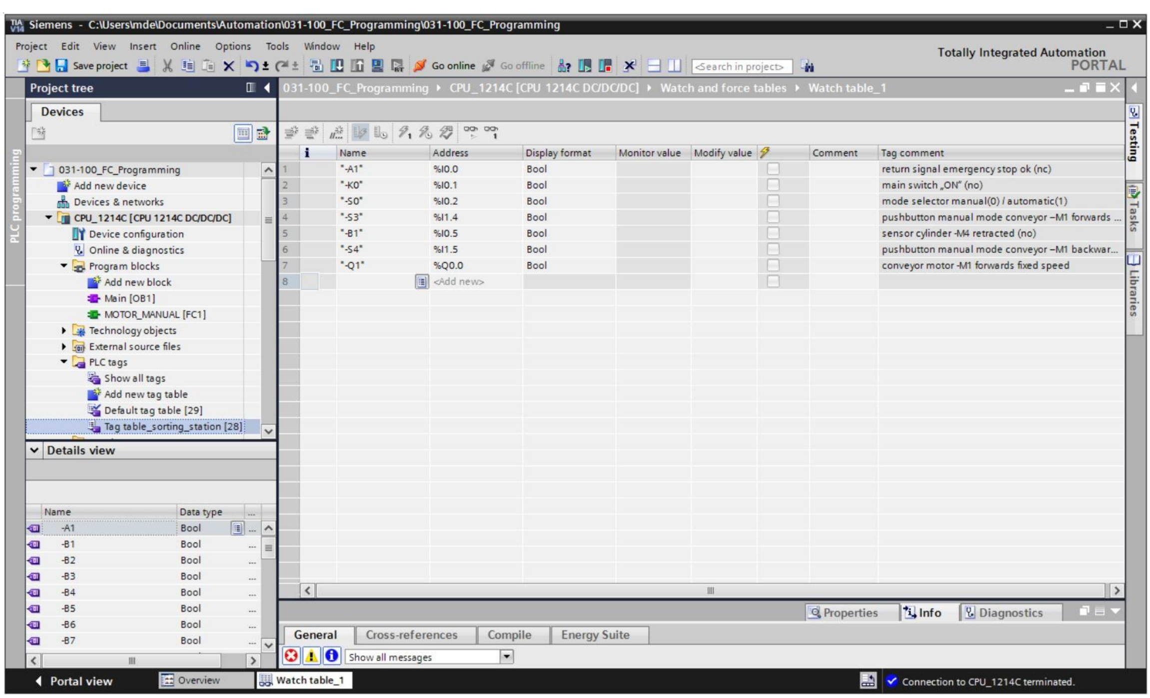
→ To monitor and modify tags, you need a watch table.

Double-click "Add new watch table" in the project tree. (→ Add new watch table).



→ Open the newly created "Watch table_1" by double-clicking it (→ "Watch table_1").

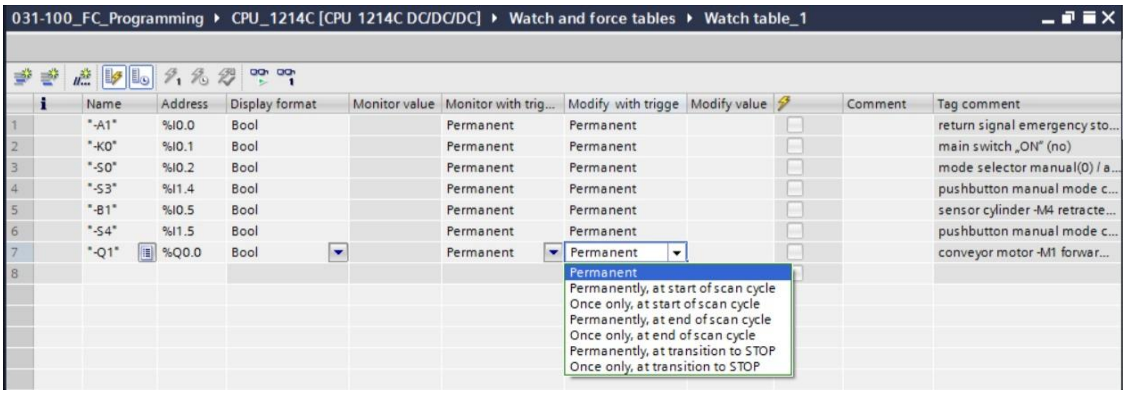
You can enter individual tags in the table or you can select the "Tag_table_sorting_station" and then select the tags to be monitored and drag them from the Details view to the watch table (→ Tag_table_sorting_station).



→ To have all monitoring and modifying functions available for selection, the following columns can be displayed:

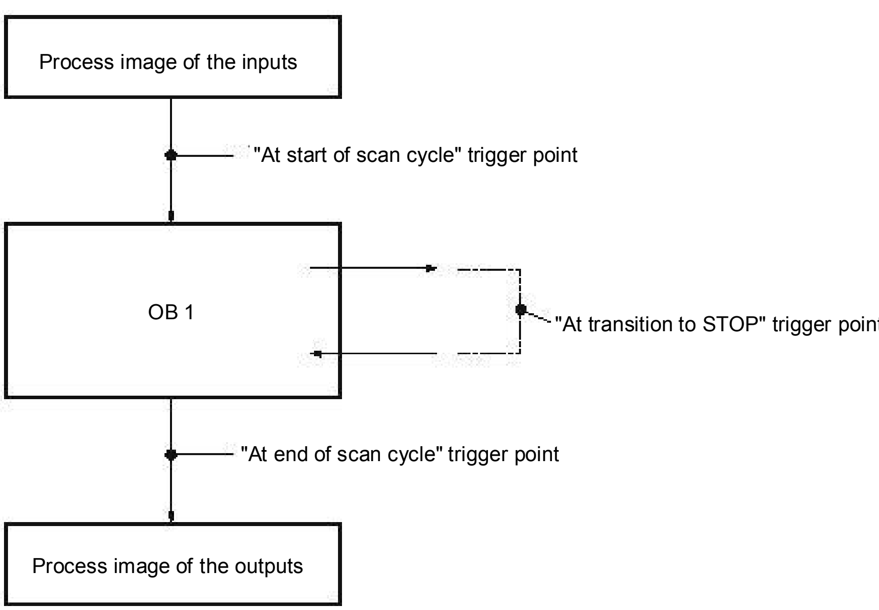
‘All modify columns’  and ‘All expanded mode columns’ .

Continue by selecting the trigger timing for the monitoring (→ Permanent).



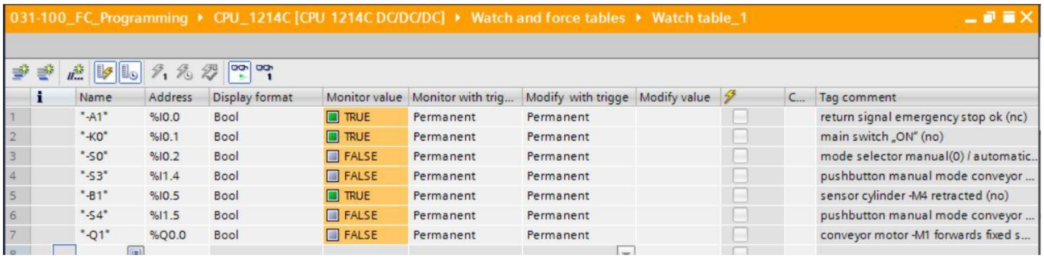
The following monitoring and modifying modes are available:

– Permanent (in this mode, the inputs are monitored/modified at the start of the cycle and the outputs at the end.)
– Once only, at start of scan cycle
– Once only, at end of scan cycle
– Permanently, at start of scan cycle
– Permanently, at end of scan cycle
– Once only, at transition to STOP
– Permanently, at transition to STOP



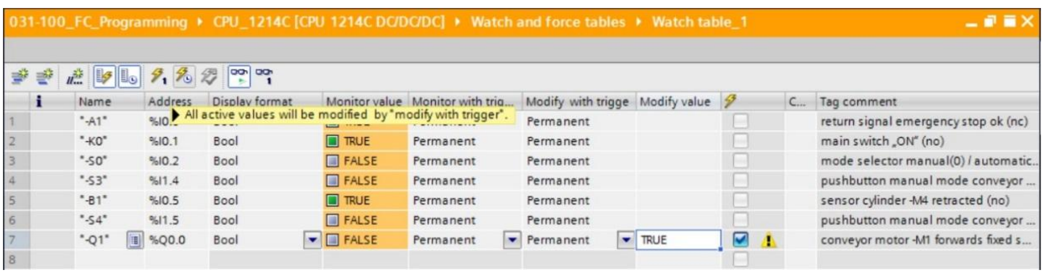
8

→ Next, click "Monitor all values once and now" or "Monitor all values according to trigger settings" (→ Monitor all).



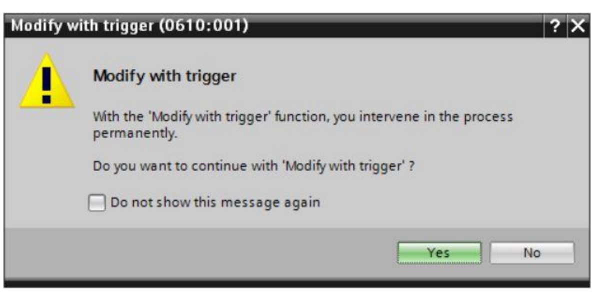
	i	Name	Address	Display format	Monitor value	Monitor with trig...	Modify with trigge	Modify value		C...	Tag comment
1		"-A1"	%I0.0	Bool	TRUE	Permanent	Permanent				return signal emergency stop ok (nc)
2		"-K0"	%I0.1	Bool	TRUE	Permanent	Permanent				main switch „ON" (no)
3		"-S0"	%I0.2	Bool	FALSE	Permanent	Permanent				mode selector manual(0) / automatic...
4		"-S3"	%I1.4	Bool	FALSE	Permanent	Permanent				pushbutton manual mode conveyor ...
5		"-B1"	%I0.5	Bool	TRUE	Permanent	Permanent				sensor cylinder -M4 retracted (no)
6		"-S4"	%I1.5	Bool	FALSE	Permanent	Permanent				pushbutton manual mode conveyor ...
7		"-Q1"	%Q0.0	Bool	FALSE	Permanent	Permanent				conveyor motor -M1 forwards fixed s...

→ To modify tags, enter the desired "Modify values". Then, click to "Modify all activated values once and now" or to "All active values will be modified by modify with trigger". (→ TRUE → "All active values will be modified by modify with trigger")

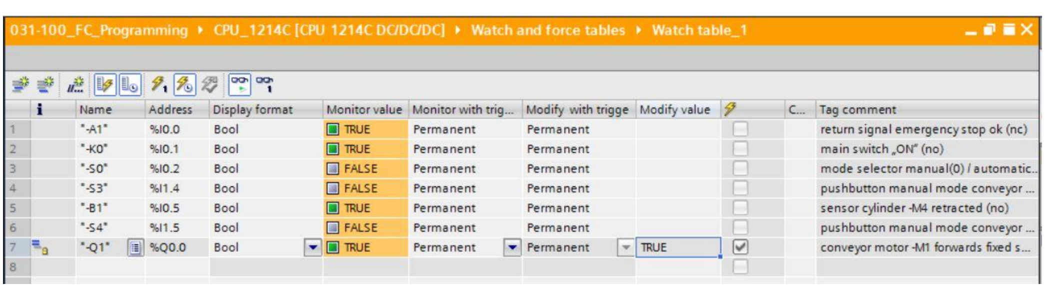


	i	Name	Address	Display format	Monitor value	Monitor with trig...	Modify with trigge	Modify value		C...	Tag comment
1		"-A1"	%I0.	All active values will be modified by "modify with trigger".			Permanent				return signal emergency stop ok (nc)
2		"-K0"	%I0.1	Bool	TRUE	Permanent	Permanent				main switch „ON" (no)
3		"-S0"	%I0.2	Bool	FALSE	Permanent	Permanent				mode selector manual(0) / automatic...
4		"-S3"	%I1.4	Bool	FALSE	Permanent	Permanent				pushbutton manual mode conveyor ...
5		"-B1"	%I0.5	Bool	TRUE	Permanent	Permanent				sensor cylinder -M4 retracted (no)
6		"-S4"	%I1.5	Bool	FALSE	Permanent	Permanent				pushbutton manual mode conveyor ...
7		"-Q1"	%Q0.0	Bool	FALSE	Permanent	Permanent	TRUE	☑		conveyor motor -M1 forwards fixed s...
8											

→ Confirm the warning with **'Yes'** (→ Yes).



→ The output becomes active even though the programmed conditions are not met.



	i	Name	Address	Display format	Monitor value	Monitor with trig...	Modify with trigge	Modify value		C...	Tag comment
1		"-A1"	%I0.0	Bool	TRUE	Permanent	Permanent				return signal emergency stop ok (nc)
2		"-K0"	%I0.1	Bool	TRUE	Permanent	Permanent				main switch „ON" (no)
3		"-S0"	%I0.2	Bool	FALSE	Permanent	Permanent				mode selector manual(0) / automatic...
4		"-S3"	%I1.4	Bool	FALSE	Permanent	Permanent				pushbutton manual mode conveyor ...
5		"-B1"	%I0.5	Bool	TRUE	Permanent	Permanent				sensor cylinder -M4 retracted (no)
6		"-S4"	%I1.5	Bool	FALSE	Permanent	Permanent				pushbutton manual mode conveyor ...
7		"-Q1"	%Q0.0	Bool	TRUE	Permanent	Permanent	TRUE	☑		conveyor motor -M1 forwards fixed s...
8											

Note: *If the watch table is closed or the connection to the PLC is lost, all modify commands are nullified.*

8

7.7 Force tags

→ The "Force" function can be used to assign a fixed value to tags. Force values are specified in a similar way as for the "Modify tags" function but, in contrast, are retained after the CPU is switched off or stopped. The "Modify tags" and "Force" functions essentially differ as follows:

In contrast to "Modify tags", it is not possible to assign values to data blocks, timers, counters and bit memory with the "Force" function.

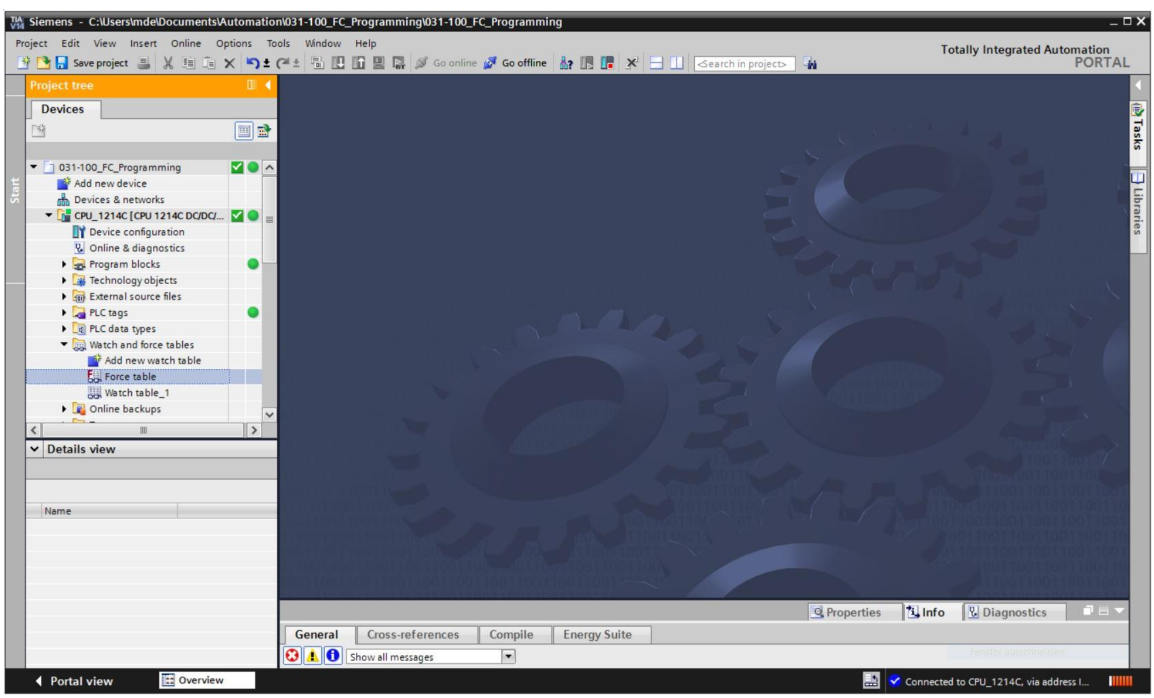
IO device inputs (e.g., IWxx:P) cannot be modified but can be pre-assigned by the "Force" function.

Unlike with the "Modify" function, values permanently assigned by the "Force" function cannot be overwritten by the user program.

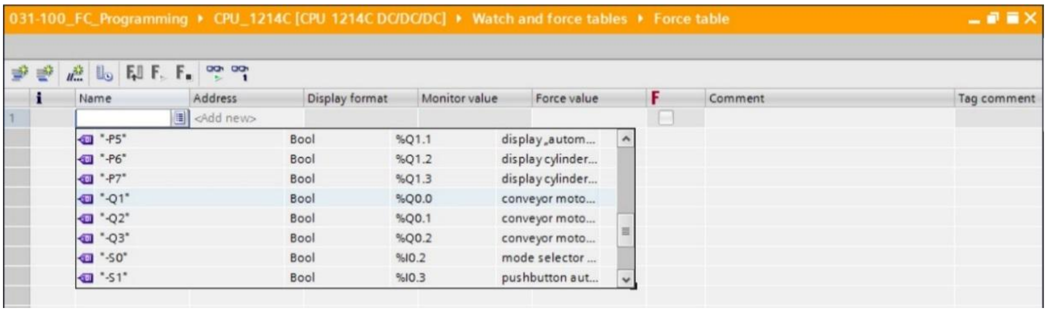
If you close the force table, the force values are retained. This is not the case with the "Modify" function.

If the online connection to the CPU is interrupted, the tags assigned with the "Force" function retain their value.

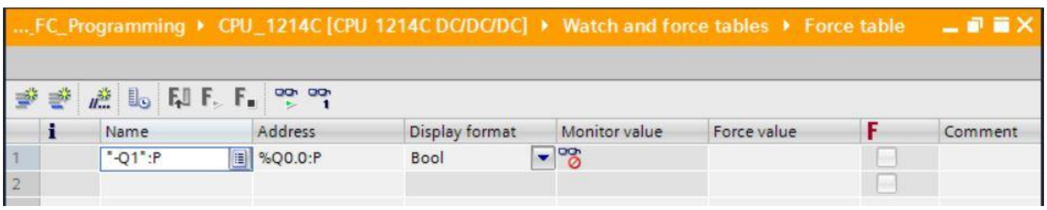
To force tags, you must first double-click the force table to open it. (→ Force table)



→ Select the "Q1" operand with address %Q0.0 from the list. (→ Q1)



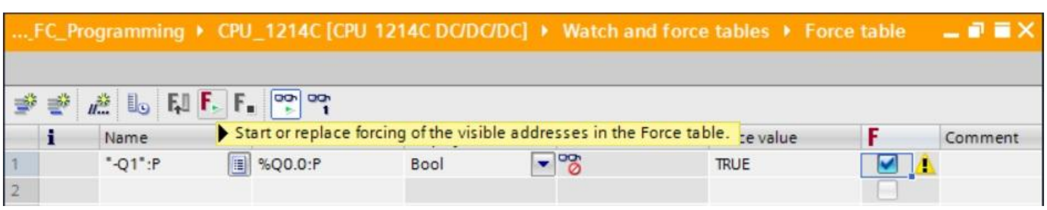
→ With forcing, the operands are entered with direct IO access (%Q0.0:P).



→ Enter the desired force value and activate it.

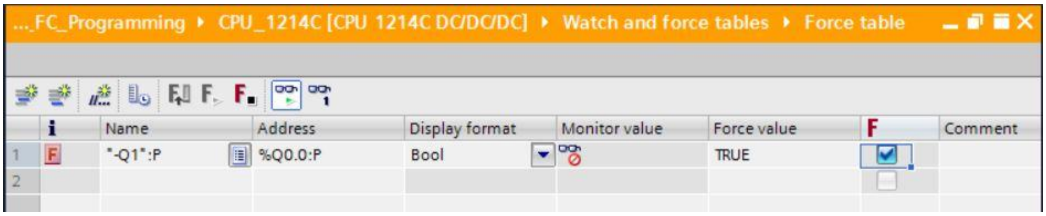
Click "Start or replace forcing". The new force request will be transferred to the CPU.

(→ %Q0.0:P → TRUE → → Start or replace forcing)



→ Confirm the warning with **'Yes'** (→ Yes).

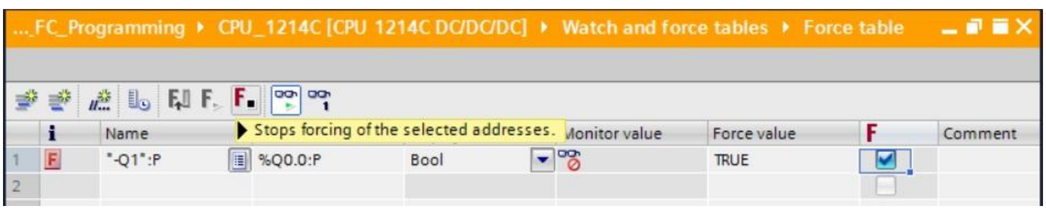
→ Forcing is activated and the yellow **MAINT LED** on the CPU lights up. In addition, an **F** on a red background is shown at the top right of the display of the S7-1200.



***Note:** If the watch table is closed or the connection to the PLC is lost, **forcing remains active** and the yellow **FORCE LED** on the CPU continues to be lit.*

→ If you want to '**Stop forcing**', simply click " Stop forcing" and confirm the next dialog with "Yes".

(→ Stop forcing) **'Yes'** (→ Yes)



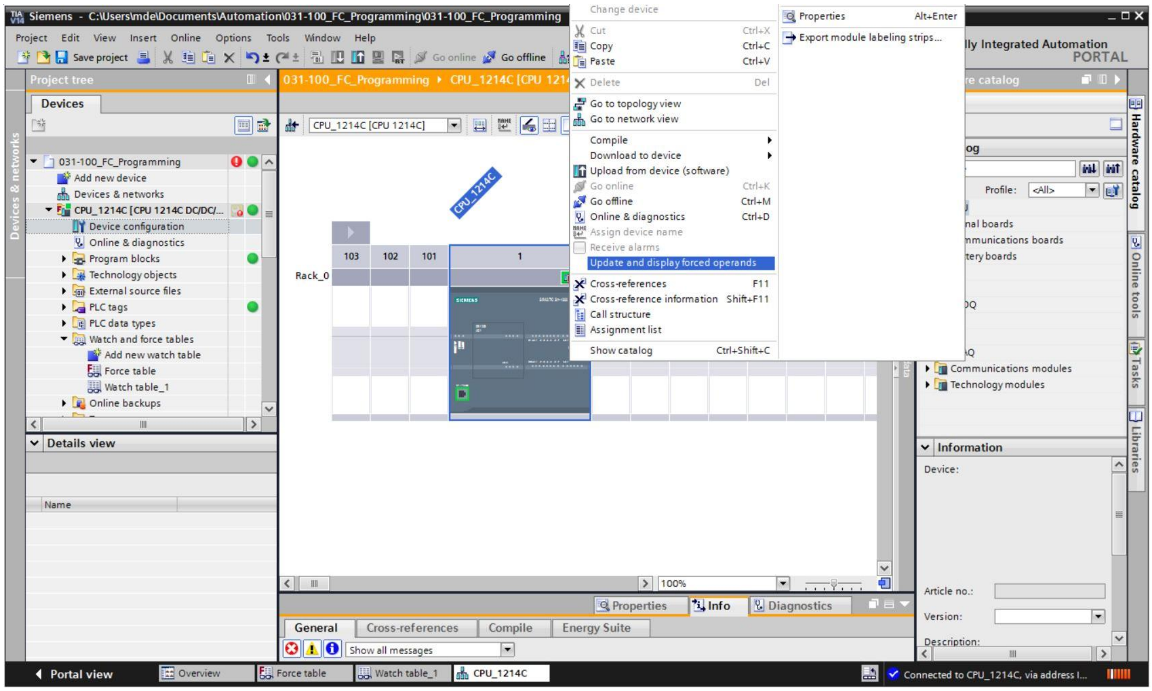
Forcing is stopped and the yellow **MAINT LED** on the CPU switches off.

→ If a force request already exists in the controller, this is indicated by the symbol in the watch table. If you then click , additional information will be displayed (→).

8

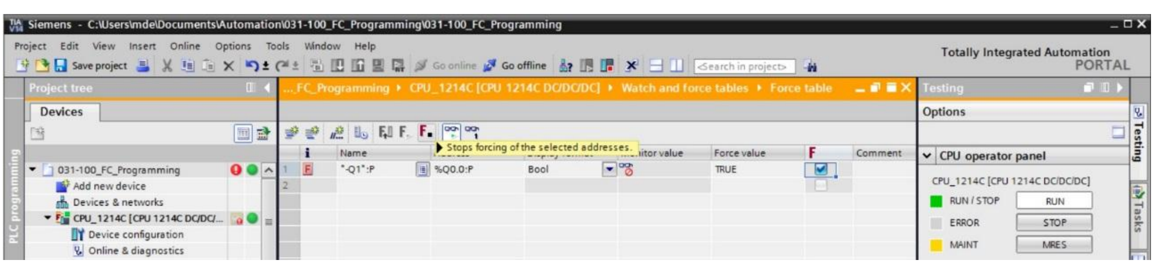
→ If a force request already exists in the controller, it can also be displayed and stopped via the online device view. To do this, you must right-click the CPU in online mode of the device view and select "Update and display forced operands".

(→ right-click the CPU → Update and display forced operands")



→ The force table with the current force requests will now be displayed and you can stop these.

(→  Stop forcing)



7.8 Checklist

No.	Description	Completed
1	Project 031-100_FC-programming… successfully retrieved.	
2	CPU 1214C from project 031-100_FC-Programming… successfully downloaded.	
3	CPU 1214C connected online.	
4	Status of the CPU 1214C checked with Online & Diagnostics.	
5	Offline/online comparison of blocks in the CPU 1214C performed.	
6	Watch table_1 created.	
7	Tags (-S0 / -S3 / -K0 / -B1 / -S4 / -A1 / -Q1) entered in watch table.	
8	Switch on conveyor motor forward by modifying the output (-Q1 = 1) in watch table.	
9	Switch off conveyor motor forward by modifying the output (-Q1 = 0) in watch table.	
10	Open force table	
11	Tag (-Q1:P) entered in force table.	
12	Switch on conveyor motor forward by forcing the output (-Q1 = 1) in force table.	
13	Force output -Q1 to switch off again.	

8 Exercise

8.1 Task – Exercise

In this exercise, the MOTOR_AUTO [FB1] function block from chapter SCE_EN_031-200_FB-Programming is to be tested.

The challenge here is that the cylinder is in the front end position and thus the enable conditions for switching on the conveyor are not met.

Using a watch table, the cylinder is to be moved to its rear end position so that the enable conditions for the MOTOR_AUTO [FB1] block are met.

8.2 Planning

Plan the implementation of the task independently using the step-by-instructions as an aid.

8.3 Checklist – Exercise

No.	Description	Completed
1	Project 031-200_FB-Programming… successfully retrieved.	
2	CPU 1214C from project 031-200_FB-Programming… successfully downloaded.	
3	Watch table created and renamed as "Watch_table_cylinder".	
4	Tags (-B1 / -B2 / -M2) entered in watch table.	
5	Retract cylinder by modifying the output (-M2 = 1) in watch table.	
6	Cylinder retracted (-B1 = 1)	
7	Reset output for Retract cylinder in watch table again (–M2 = 0).	

9 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

8

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Please note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is provided at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, please contact your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

9

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

Table of contents

9

Diagnostics via web server

1 Goal

In this module, the reader will become acquainted with the contents that can be displayed via the web server of the CPU 1214C.

This module will present the diagnostic functions in the web server that, for example, you can test with the TIA project from the SCE_EN_031-410_Basics Diagnostics with SIMATIC S7-1200 module.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

This chapter builds on the hardware configuration of the SIMATIC S7 CPU1214C DC/DC/DC. However, other hardware configurations can be used. You can use the following project for this chapter, for example:

SCE_EN_031-410_Basics_Diagnostics_S7-1200.zap14

9

3 Required hardware and software

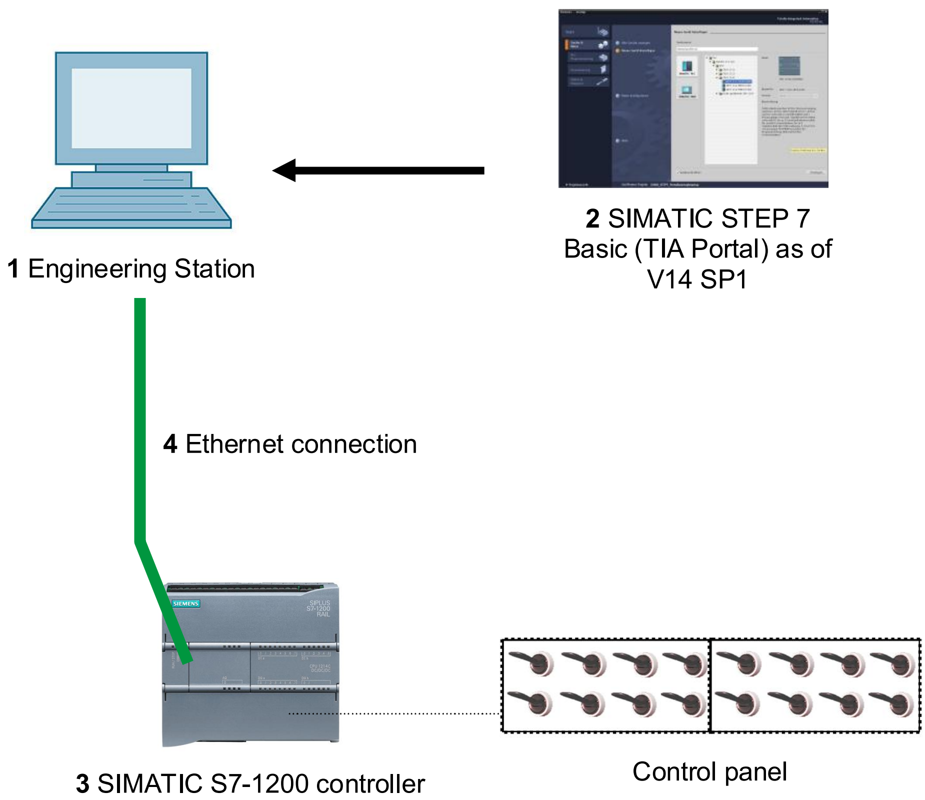
1 Engineering station: requirements include hardware and operating system (for additional information, see Readme on the TIA Portal Installation DVDs)

2 SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1

Note: The digital inputs should be fed out to a control panel.

4 Ethernet connection between engineering station and controller



4 Theory

4.1 Diagnostics via web server

The web server enables monitoring and administering of the CPU by authorized users over a network.

This permits evaluation and diagnostics over long distances. Monitoring and evaluation is possible without the TIA Portal; all you need is a web browser.

The web server is deactivated in the delivery state of the CPU. This means that you must load a project in which the web server is activated to enable access using the web browser.

The web server offers the following security functions:

– Access via secure "https" transmission protocol

– User authorization by means of a user list

– Restriction of access from certain interfaces

You need a web browser to access the HTML pages of the CPU.

The following web browsers have been tested for communication with the CPU:

– Internet Explorer (Version 8)

– Mozilla Firefox (Version 21)

– Mobile Safari (iOS5)

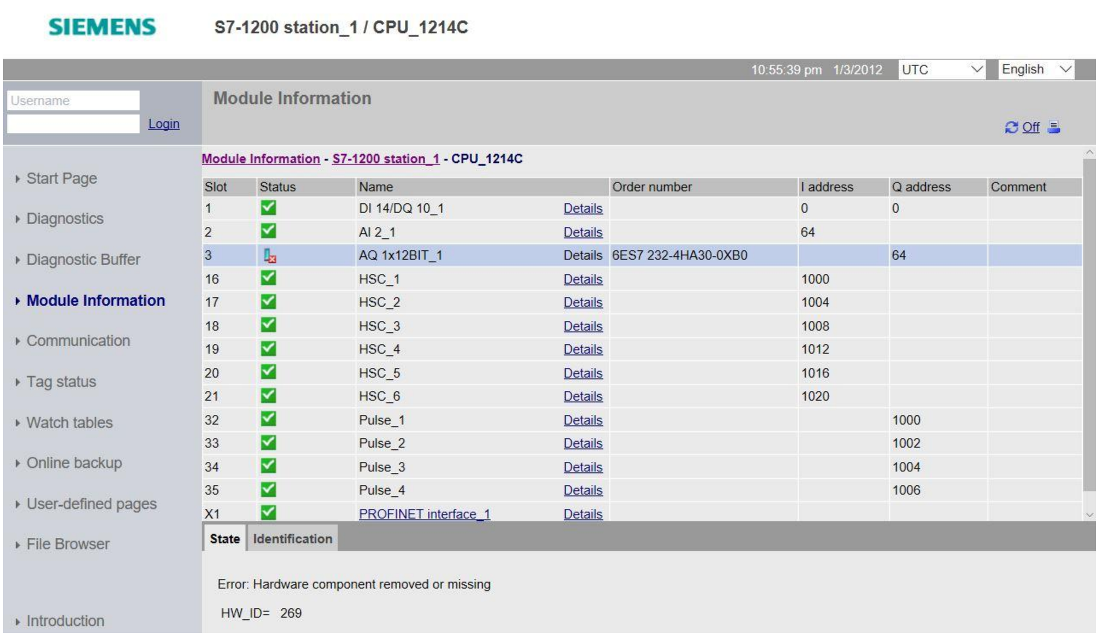


Figure 1: Web server of the CPU 1214C DC/DC/DC with Module Information

Note: *Make sure that you protect the CPU from manipulation and unauthorized access through the use of different methods (e.g., limiting network access, using firewalls).*

5 Task

The following advanced diagnostic functions will be shown and tested in this chapter:

– Configuration of the web server of the CPU 1214C DC/DC/DC

– Display messages via the web server of the CPU 1214C DC/DC/DC

6 Planning

The diagnostic functions will be performed using a finished project as an example.

A project in the TIA Portal that was previously downloaded to the controller should be open for this.

In our case, after starting the TIA Portal, a previously created project will be retrieved from the archive and downloaded to the associated controller.

You can then configure the web server in the TIA Portal.

To demonstrate the display of an error in the module information, the configured signal board AQ 1x12Bit, for example, can be removed. **Caution!** The PLC should be disconnected from the supply voltage beforehand.

9

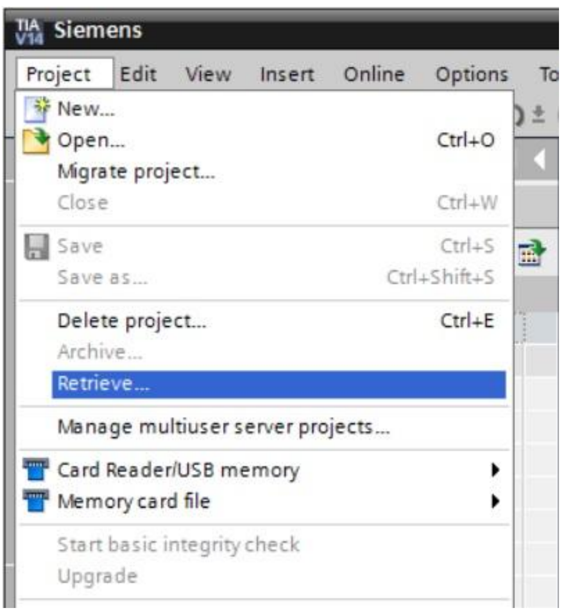
7 Structured step-by-step instructions

You can find instructions on how to carry out planning below. If you already have a good understanding of everything, it will be sufficient to focus on the numbered steps. Otherwise, simply follow the detailed steps in the instructions.

7.1 Retrieve an existing project

Before we begin with diagnostics via the web server, we need a project from the SCE_EN_031-410 Basics Diagnostics S7-1200 module. (e.g., SCE_EN_031-410_Basics Diagnostics_S7-1200_2.zap14)

To retrieve an existing project that has been archived, you must select the relevant archive with →Project →Retrieve in the project view. Confirm your selection with "Open". (→ Project → Retrieve → Select a .zap archive → Open)

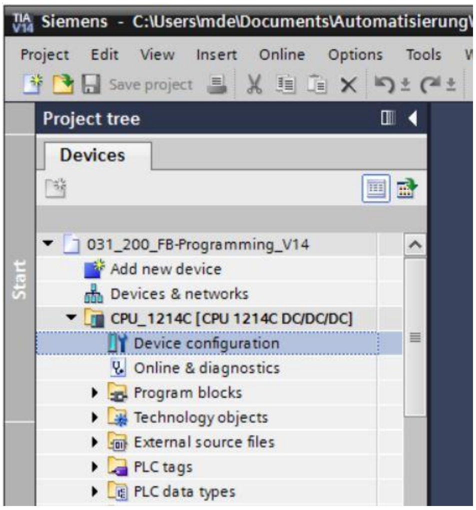


→ The next step is to select the target directory where the retrieved project will be stored. Confirm your selection with "OK". (→ Target directory → OK)

9

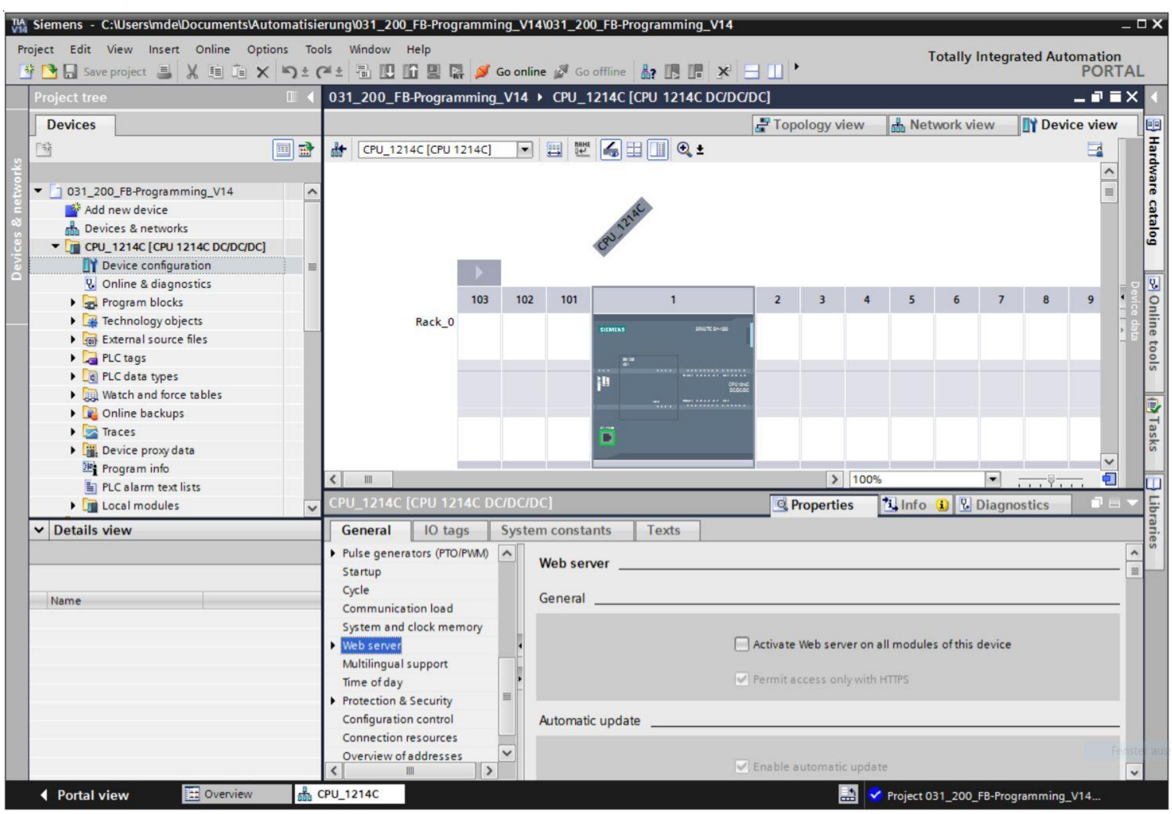
7.2 Configure the web server

→ To configure the web server, open the device configuration of the CPU 1214C DC/DC/DC. (→ CPU_1214C [CPU 1214C DC/DC/DC] → Device configuration)



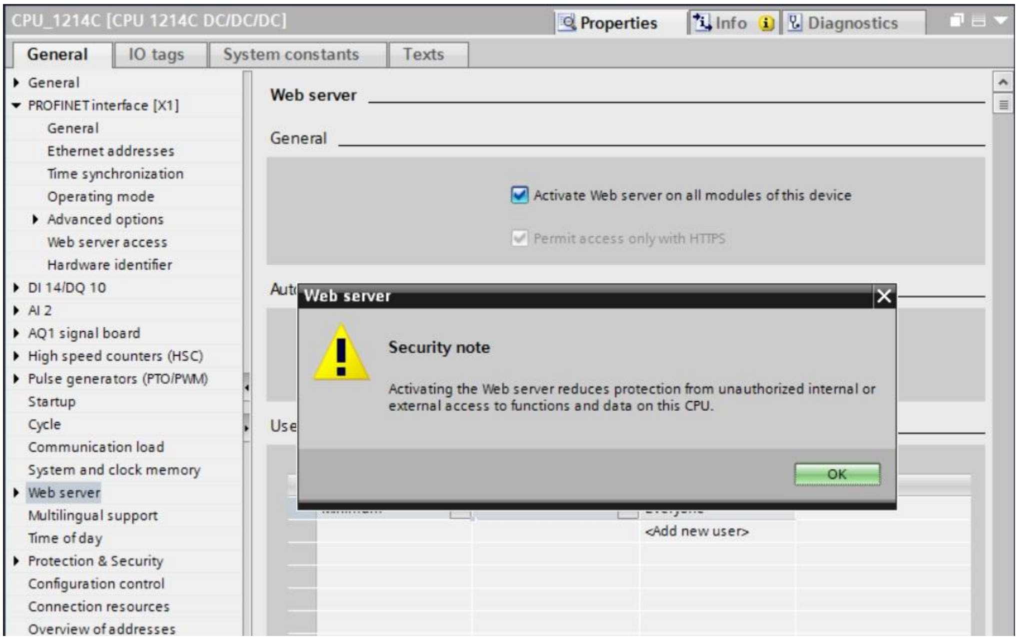
→ Select the CPU and choose the 'Web server' menu item in the properties.

(→ CPU_1214C → Properties → Web server)



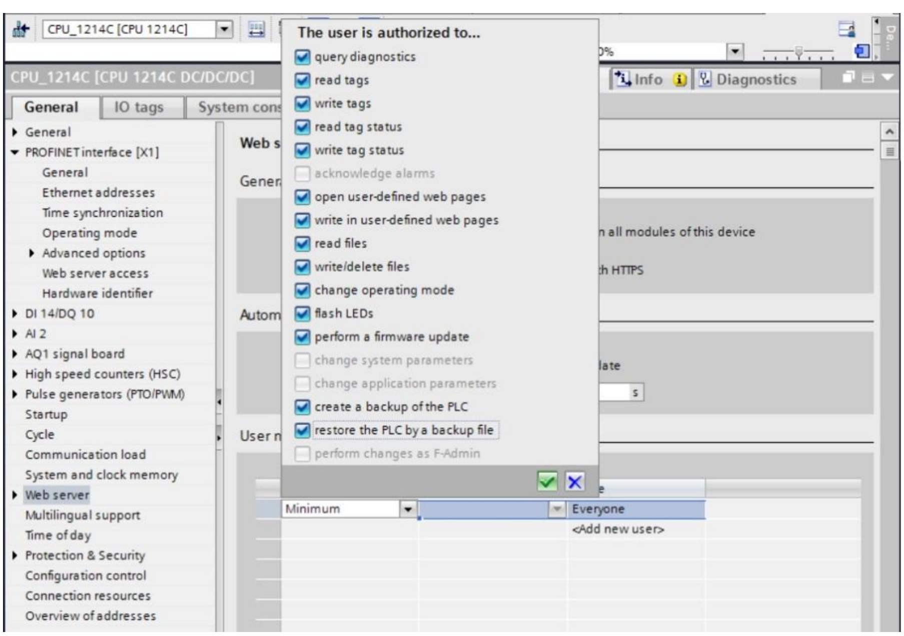
→ Activate the web server on this module and confirm the security note.

(→ Activate web server on this module → OK)



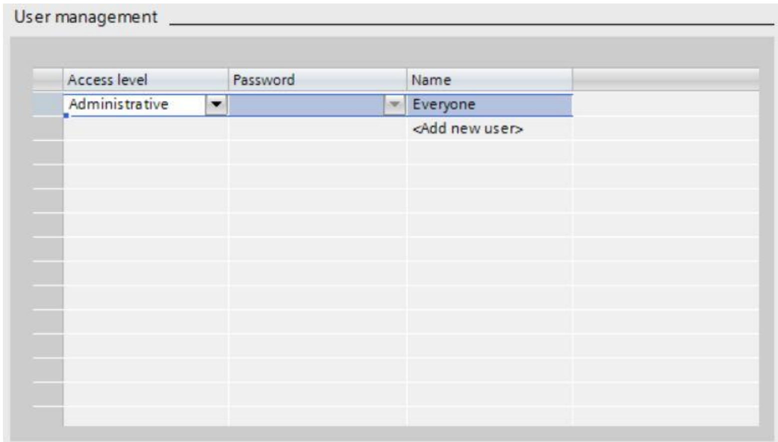
→ Leave the check mark for 'Enable automatic update', and select the security settings of the 'Everybody' user. Enable this user to carry out all possible actions and accept your settings.

(→ →)



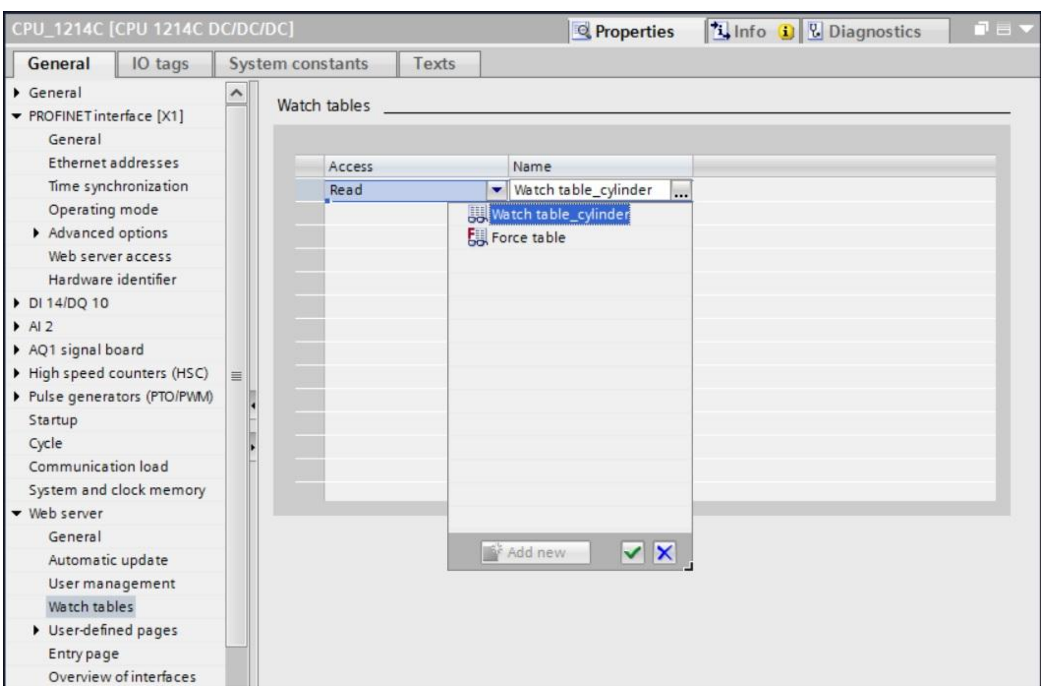
***Notes:** You can also create multiple users here with different authorizations. These users then require a password.*

→ As a result of these authorizations, the 'Everybody' user is now automatically assigned the access level 'Administrative'.

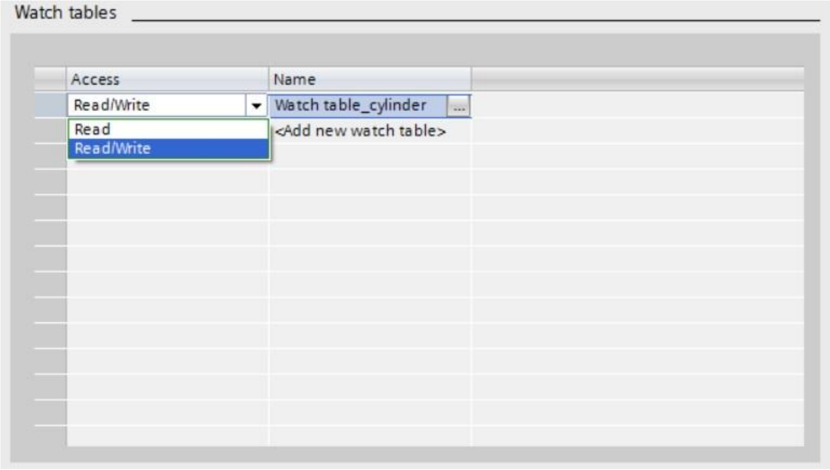


→ In the 'Watch tables' menu item, the 'Watch table_Cylinder' can now be entered in the web server.

(→Watch table_Cylinder → )

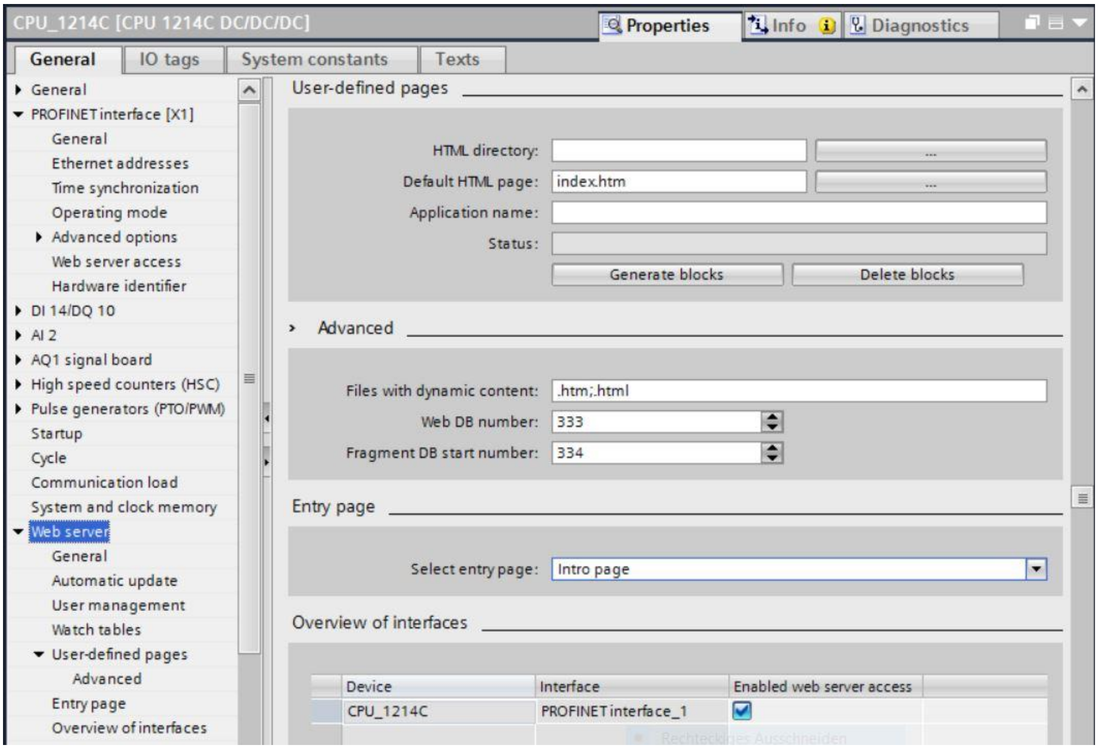


→ The access here is read/write access. (→ Read/Write)



→ User-defined web pages will not be created here. We must enable PROFINET interface_1 for access to the web server

(→ Enabled web server access → ☑ PROFINET interface_1)

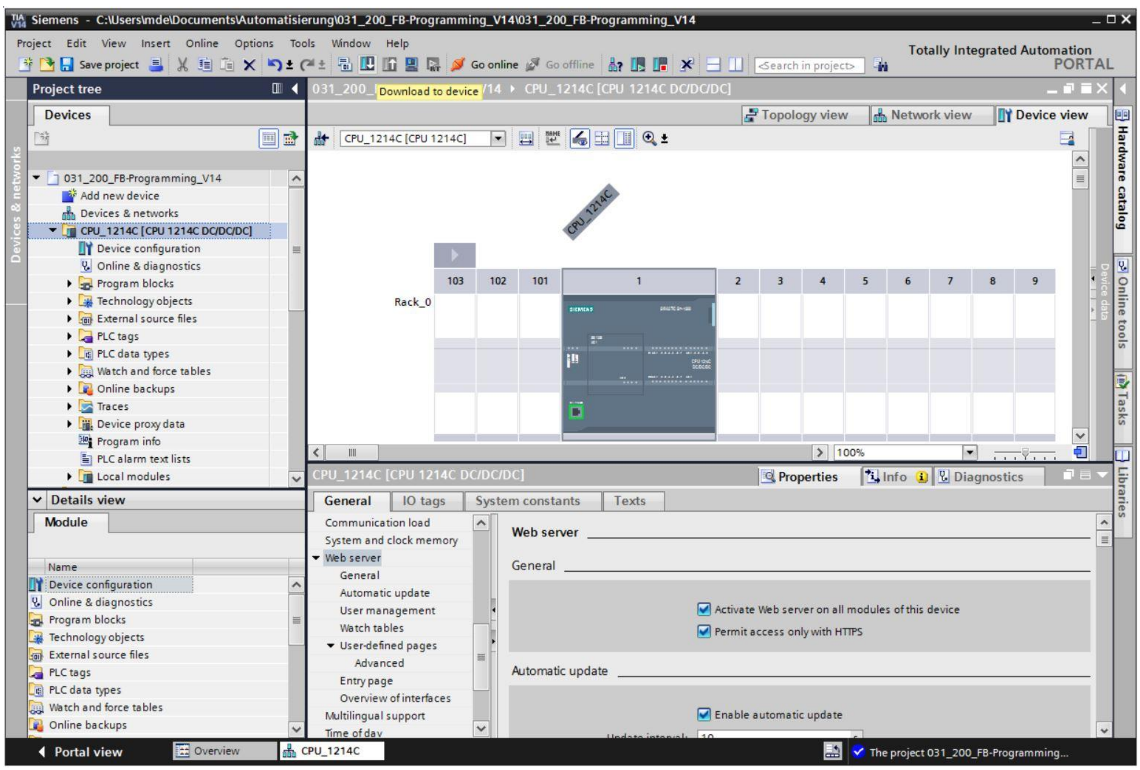


9

7.3 Save project and download CPU

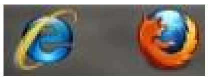
→ To save your project, click the Save project button in the menu. The complete controller with the modified configuration settings in the hardware configuration, as described in the previous modules, can be downloaded.

(→ Save project → )

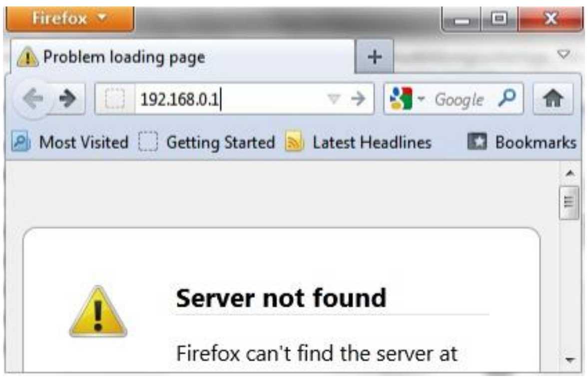


7.4 Diagnostics for the S7-1200 via the web

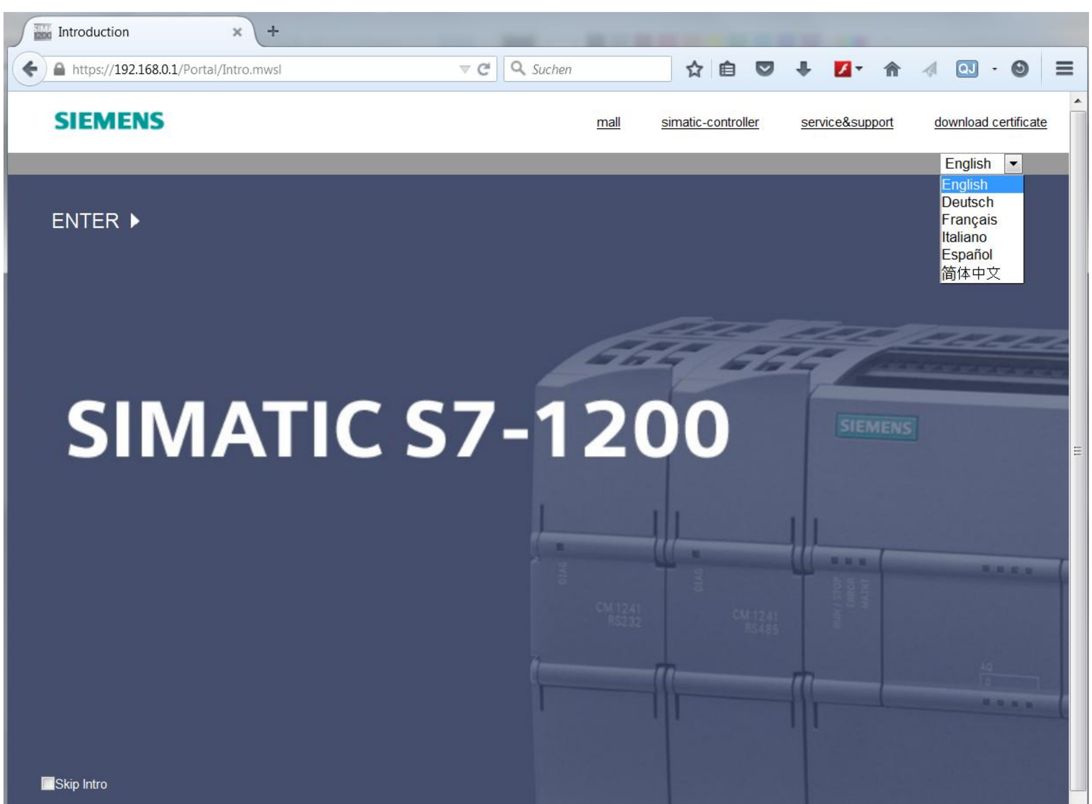
→ In order to access the web server of the CPU 1214C DC/DC/DC, we open any web browser on a PC that is connected to the CPU via TCP/IP.



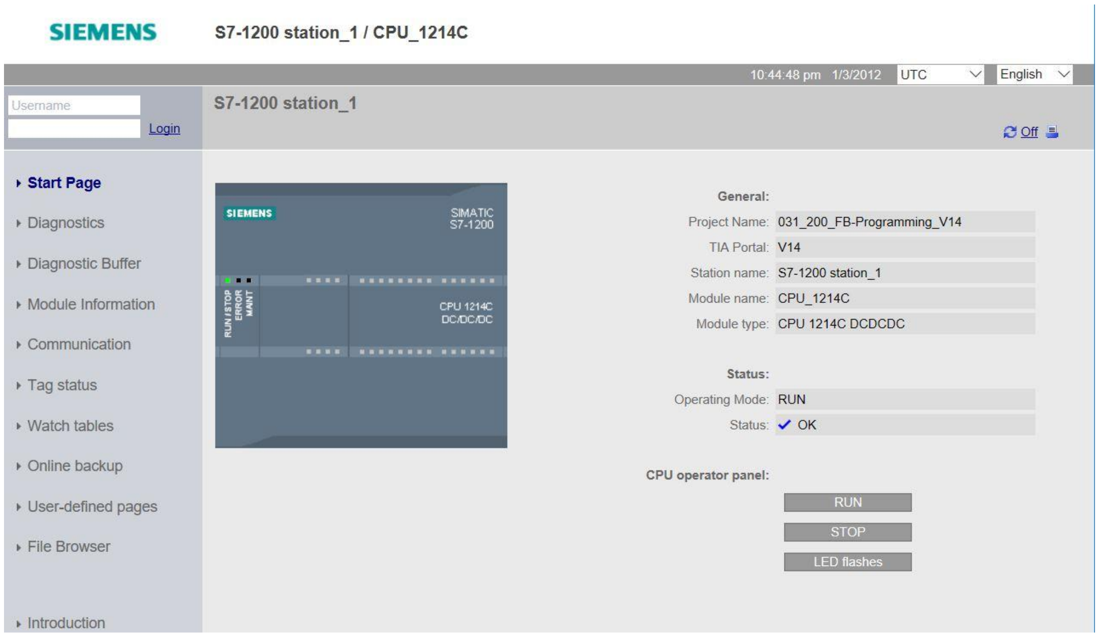
→ There we enter the IP address of the CPU 1214C DC/DC/DC. (→ 192.168.0.1)



→ On the displayed web page, we first select the language and then click '**ENTER**'. (→ English → ENTER)

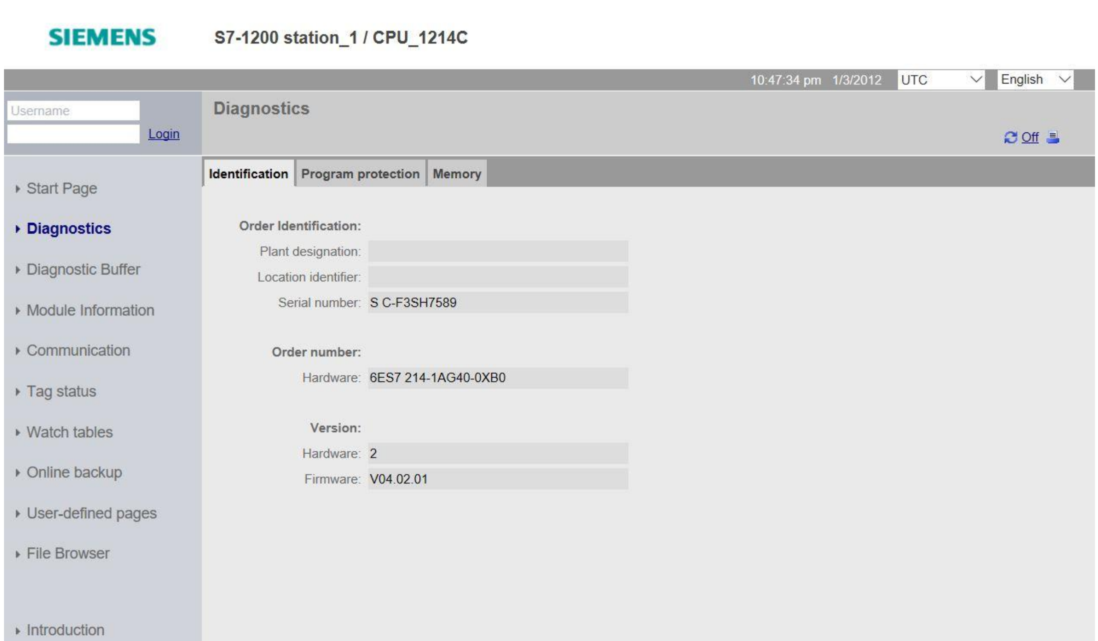


→ On the '**Home Page**' we see general information about the PLC and its status. (→ Home Page)

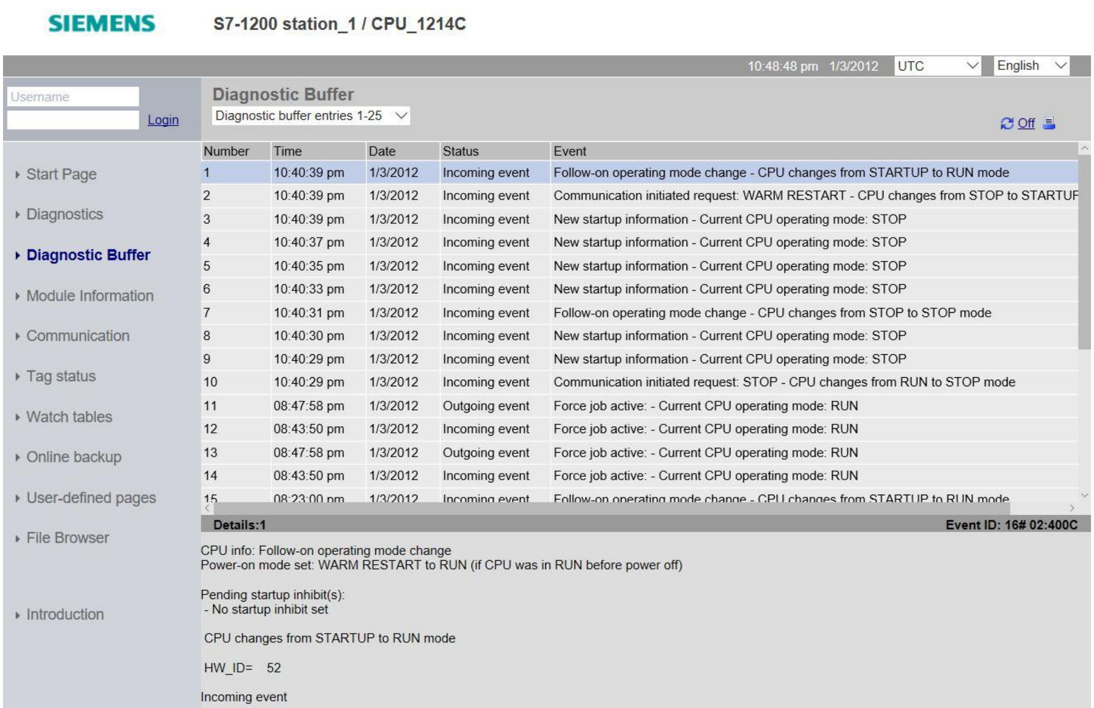


→ Hardware, Firmware Version and Serial number are displayed under 'Diagnostics'. (→ Diagnostics)

9

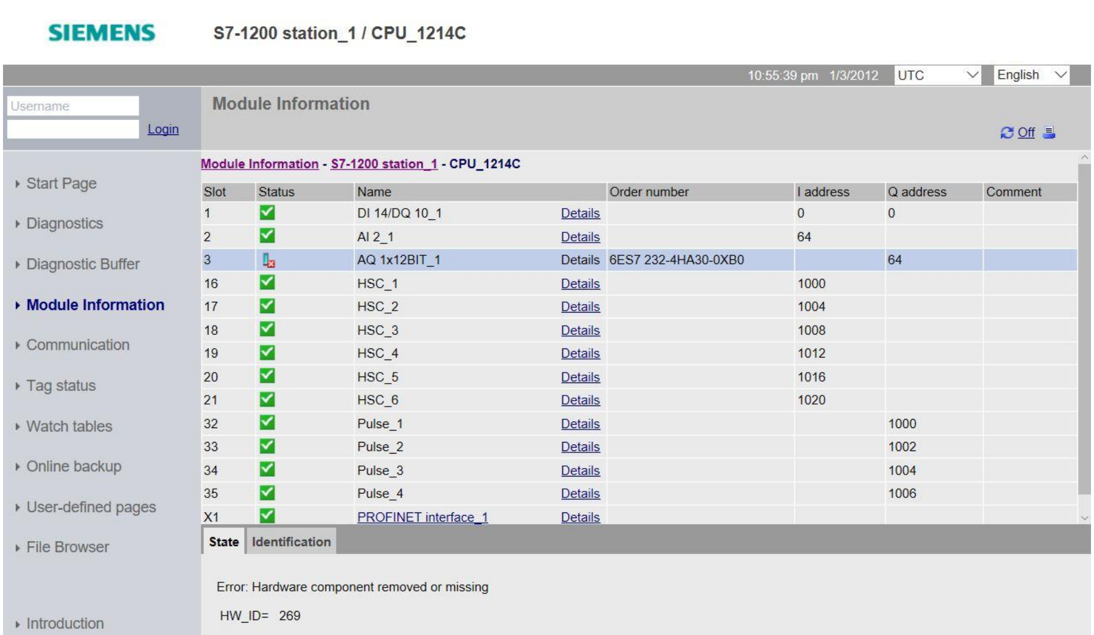


→ Under 'Diagnostics Buffer' we see descriptive information for all events in the CPU. Event information is recorded in a circular buffer. The most recent alarm is displayed in the top line. (→ Diagnostics Buffer)



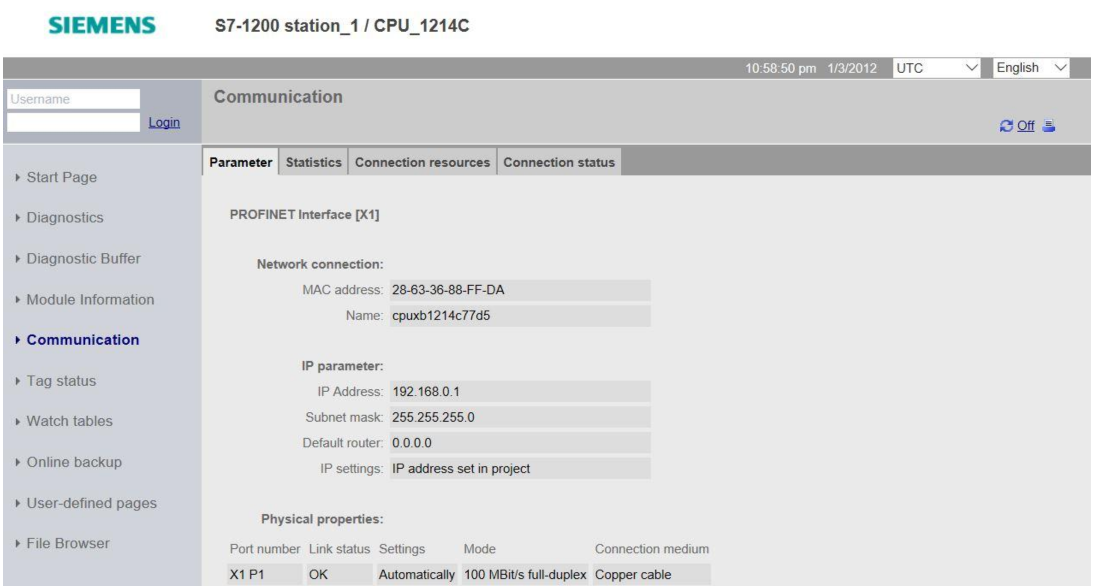
→ The status of the individual modules of our SIMATIC S7-1200 is displayed with additional details in the 'Module Information' view.

(→ Module Information)

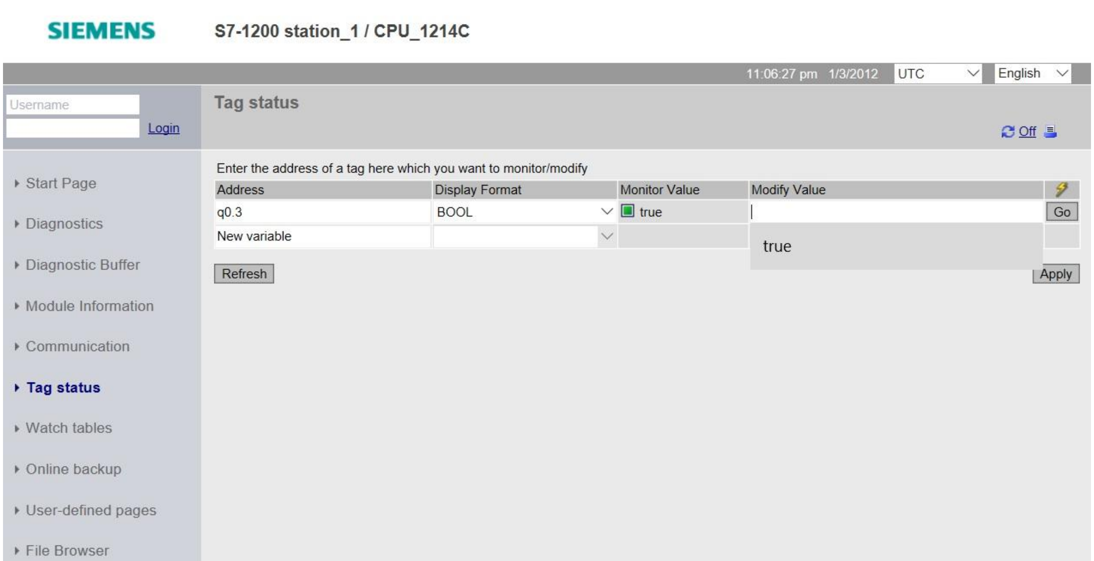


9

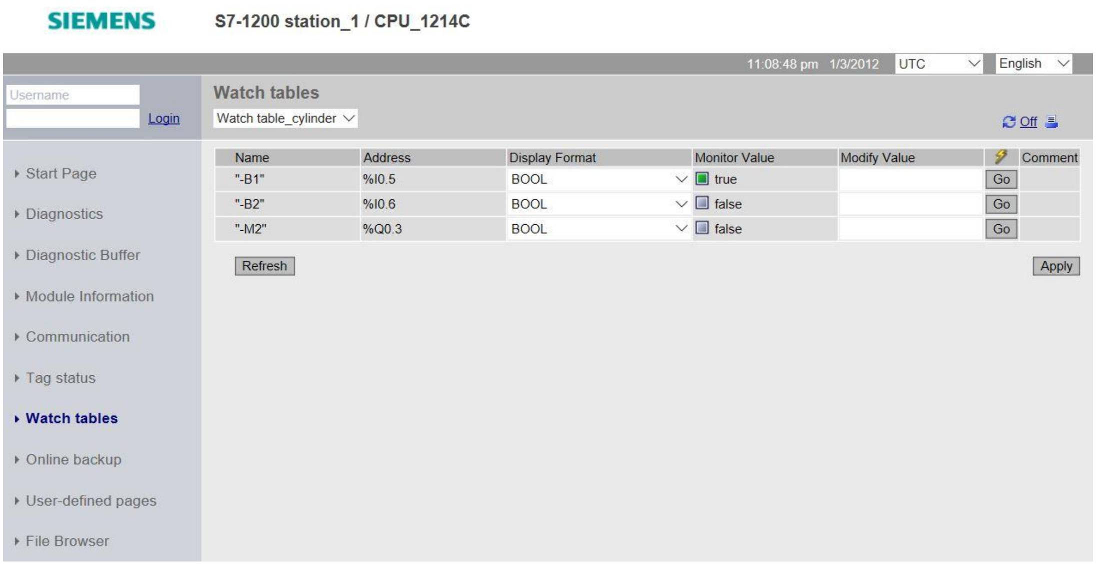
→ Details about communications settings are displayed under 'Communication'. (→ Communication)



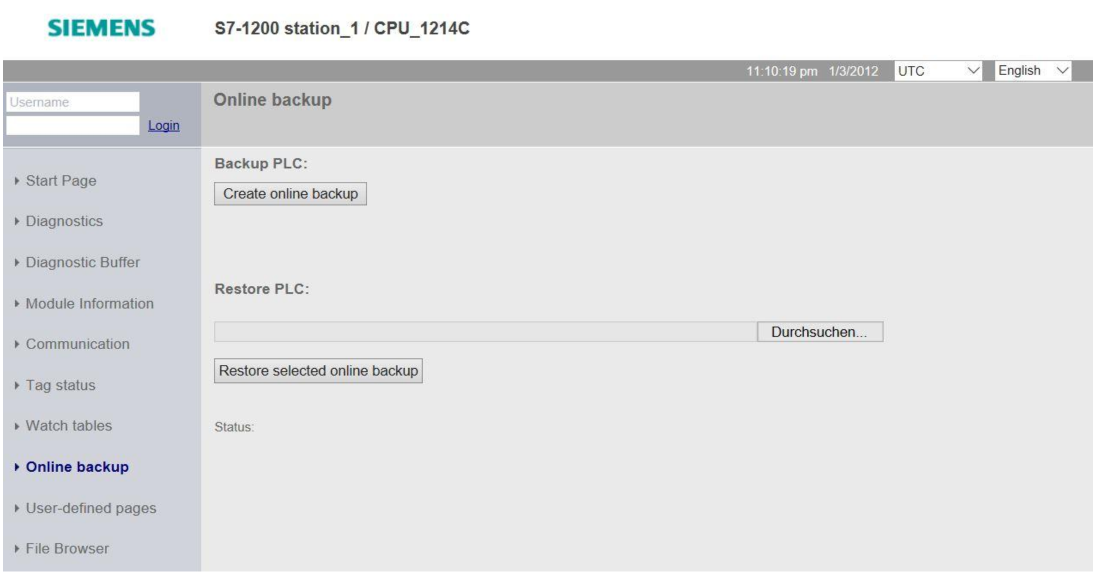
→ Values of the individual tags can be displayed and changed under 'Tag Status'. (→ Tag Status)



→ 'Watch tables' that are linked with the web server, such as the 'Watch table_cylinder', can also be displayed. (→ Watch tables → Watch table_cylinder)

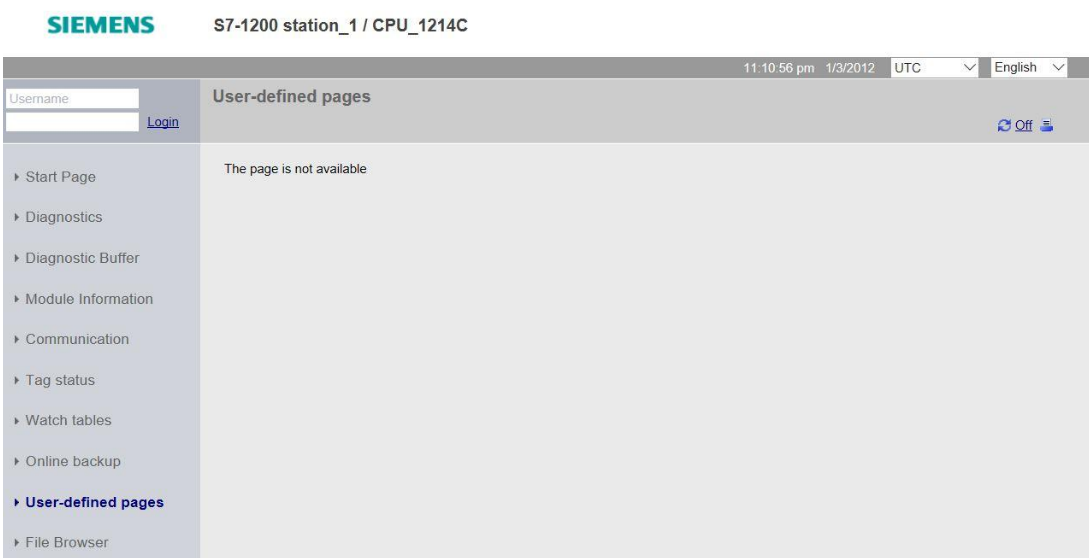


→ Under "Online backup" you can create a backup of the project in the PLC and restore this backup later. (→ Online backup → Create online backup → Restore selected online backup)

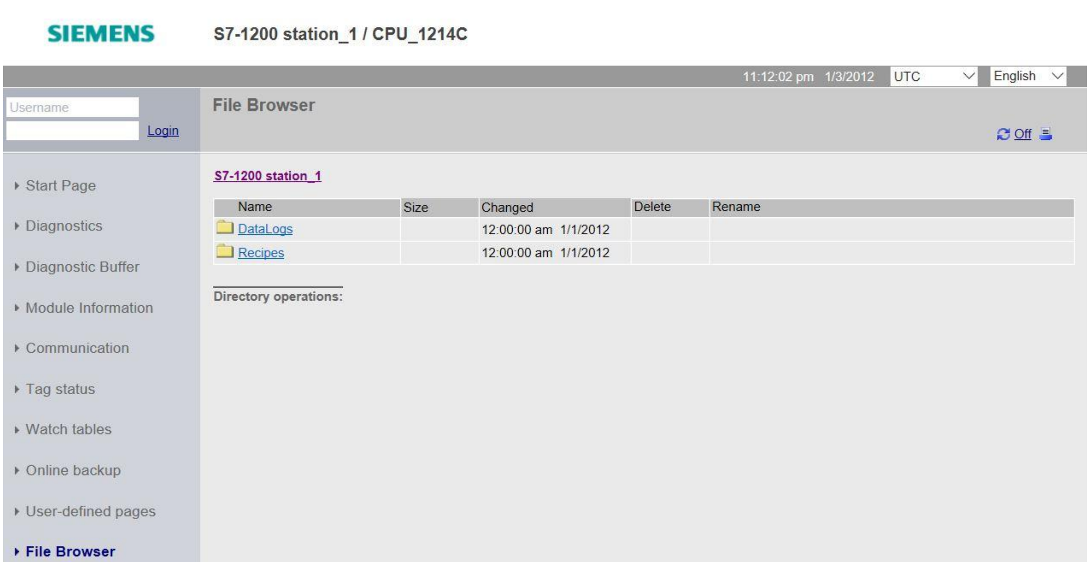


9

→ Individually created pages for the visualization and also for operator control of processes would be seen under 'User-defined pages'. (→ User-defined pages)



→ Data can be stored directly on the memory card in the CPU or loaded from there using the 'File Browser'. (→ File Browser)



7.5 Checklist

No.	Description	Completed
1	Project 031-410_Basics Diagnostics_S7-1200… successfully retrieved.	
2	Web server for the CPU 1214C from project 031-410_Basics Diagnostics_S7-1200… successfully configured.	
3	CPU 1214C from project 031-410_Basics Diagnostics_S7-1200… successfully downloaded.	
4	Voltage supply switched off.	
5	Signal board AQ 1x12Bit removed.	
6	Voltage supply switched on again.	
7	Web server of the CPU 1214C opened in one of the approved web browsers.	
8	Display checked for missing signal board AQ 1x12Bit in the Module Information menu item of the web server.	

8 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software/ firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

9

Notes

9

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Please note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is provided at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, please contact your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

10

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of

Table of contents

Analog Values for SIMATIC S7-1200

1 Goal

In this chapter, you will become acquainted with the analog value processing of the SIMATIC S7-1200 with the TIA Portal programming tool.

The module explains the acquisition and processing of analog signals and gives a step-by-step description of read and write access to analog values in the SIMATIC S7-1200.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

This chapter builds on the chapter IEC Timers and Counters with the SIMATIC S7 CPU1214C. You can use the following project for this chapter, for example: SCE_EN_031-300_IEC_Timers_Counters_S7-1200.zap14

3 Required hardware and software

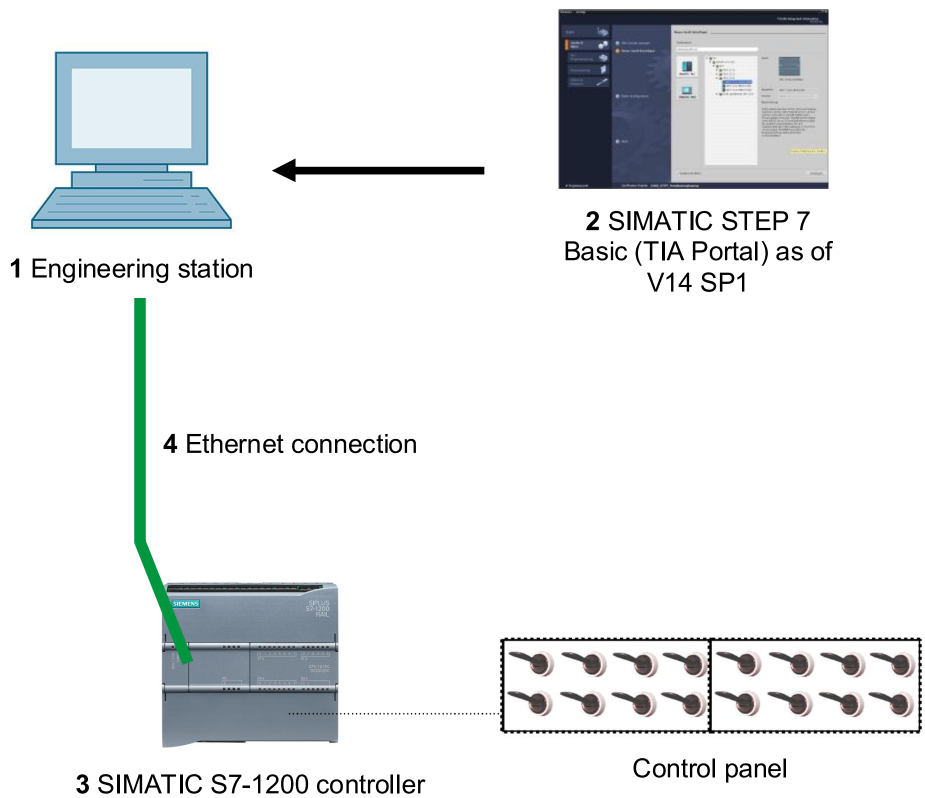
1 Engineering station: requirements include hardware and operating system (for additional information, see Readme on the TIA Portal Installation DVDs)

2 SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1

Note: The digital inputs and analog inputs and outputs should be fed out to a control panel.

4 Ethernet connection between engineering station and controller



10

4 Theory

4.1 Analog signals

In contrast to a binary signal, which can assume only two signal states ("Voltage present +24 V" and "Voltage not present 0 V"), analog signals can assume any value within a defined range. A typical example of an analog sensor is a potentiometer. Depending on the position of the knob, any resistance can be set, up to the maximum value.

Examples of analog quantities in control engineering:

– Temperature -50 to +150 °C

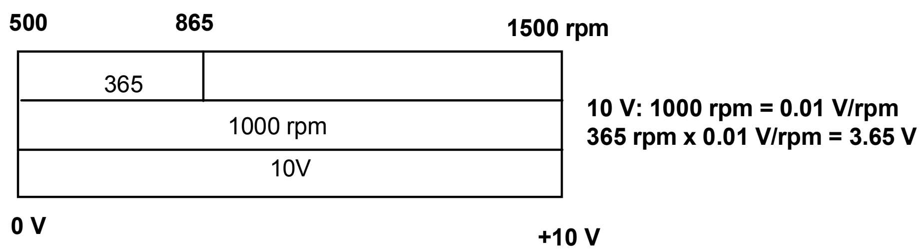
– Flow rate 0 to 200 l/min

– Speed -500 to +50 rpm

– etc.

4.2 Measuring transducers

These quantities are converted to electrical voltages, currents or resistances with the help of a measuring transducer. If, for example, a speed is to be measured, the speed range of 500 to 1500 rpm can be converted to a voltage range of 0 to +10 V using a measuring transducer. At a measured speed of 865 rpm, the measuring transducer would output a voltage value of +3.65 V.

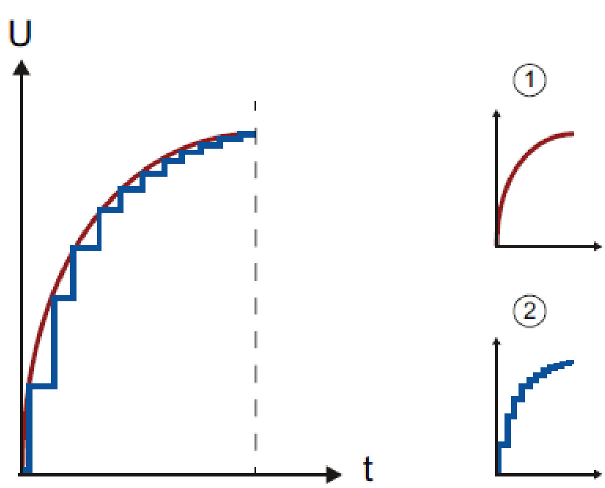


4.3 Analog modules – A/D converter

These electrical voltages, currents or resistances are then connected to an analog module that digitizes this signal for further processing in the PLC.

If analog quantities will be processed with a PLC, the read-in voltage, current or resistance value must be converted to digital information. The analog value is converted to a bit pattern. This conversion is referred to as analog-to-digital conversion (A/D conversion). This means, for example, that the voltage value of 3.65 V is stored as information in a series of binary digits.

The result of this conversion is always a 16-bit word for SIMATIC products. The integrated ADC (analog-to-digital converter) of the analog input module digitizes the analog signal being acquired and approximates its value in the form of a stepped curve. The most important parameters of an ADC are its resolution and conversion rate.

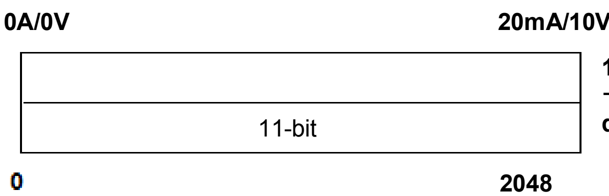


1: Analog value

2. Digital value

The more binary digits the digital representation uses, the finer the resolution is. For example, if only 1 bit was available for the voltage range of 0 to +10 V, you would only know whether the measured voltage is between 0 and +5 V or between +5 V and +10 V. With 2 bits, the range can be divided into 4 individual ranges, i.e., 0 to 2.5 / 2.5 to 5 / 5 to 7.5 / 7.5 to 10 V. Conventional A/D converters in control engineering use 8 bits, 11 bits or more for converting.

With 8 bits you have 256 individual ranges, while 11 bits provide a resolution of 2048 individual ranges.



10 V: 2048 = 0.0048828
→ Voltage differences of <5 mV can be detected

4.4 Data types of the SIMATIC S7-1200

The SIMATIC S7-1200 has many different data types for representing different numerical formats. A list of some of the elementary data types is given below.

Data type	Size (bits)	Range	Example of constant entry
Bool	1	0 to 1	TRUE, FALSE, O, 1
Byte	8	16#00 to 16#FF	16#12, 16#AB
Word	16	16#0000 to 16#FFFF	16#ABCD, 16#0001
DWord	32	16#00000000 to 16#FFFFFFFF	16#02468ACE
Char	8	16#00 to 16#FF	'A', 'r', '@'
Sint	8	-128 to 127	123,-123
Int	**16**	**-32,768 to 32,767**	**123, -123**
Dint	32	-2,147,483,648 to 2,147,483,647	123, -123
USInt	8	0 to 255	123
UInt	16	0 to 65,535	123
UDInt	32	0 to 4,294,967,295	123
Real	**32**	**+/-1.18 x 10^{-38} to +/-3.40 x 10^{38}**	**123.456, -3.4, -1.2E+12, 3.4E-3**
LReal	64	+/-2.23 x 10^{-308} to +/-1.79 x 10^{308}	12345.123456789 -1.2E+40
Time	32	T#-24d_20h_31 m_23s_648ms to T#24d_20h_31 m_23s_647ms Saved as: -2,147,483,648 ms to +2,147,483,647 ms	T#5m_30s 5#-2d T#1d_2h_15m_30x_45ms
String	Variable	0 to 254 characters in byte size	'ABC'

***Note:** The **'INT'** and **'REAL'** data types play a large role in analog value processing. This is because read-in analog values exist as 16-bit integers in the **'INT'** format, and in order to ensure exact further processing only **'REAL'** floating-point numbers should be used due to rounding errors in the case of **'INT'**.*

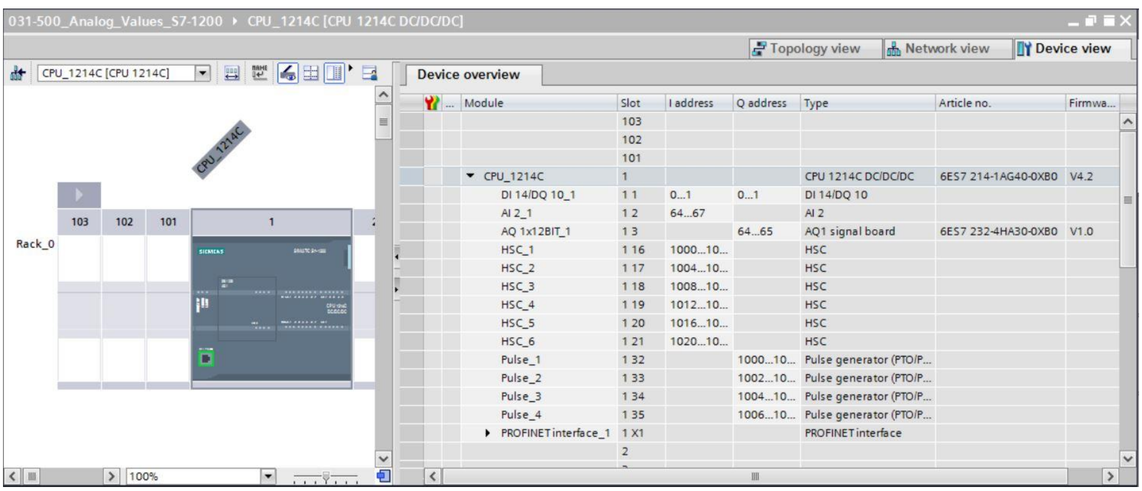
4.5 Reading/writing analog values

Analog values are read into the PLC or output from the PLC as word information. These words are accessed, for example, with the following operands:

%IW 64	Analog input word 64
%QW 64	Analog output word 64

Each analog value ("channel") occupies one input or output word. The format is **'Int'**, an integer.

The addressing of input and output words conforms to the addressing in the device overview. For example:



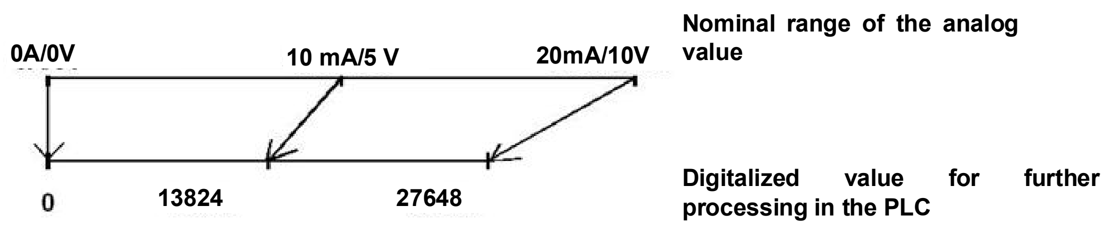
Here, the address of the first analog input would be %IW 64, and the address of the second analog input would be %IW 66.

The address of the analog output would be %QW 64.

10

The analog value transformation for further processing in the PLC is the same for analog inputs and analog outputs.

The digitized value ranges are as follows:



Often, these digitized values still have to be normalized by further processing them in the PLC in an appropriate manner.

4.6 Normalizing analog values

If an analog input value exists as a digitized value in the range +/- 27648, it must usually still be normalized so that the numerical values correspond to the physical quantities in the process.

Likewise, the analog output usually results from setting of a normalized value that then still has to be scaled to the output value +/- 27648.

In the TIA Portal, ready-made blocks or arithmetic operations are used for normalizing and scaling.

For this to be carried out as exactly as possible, the values for the normalizing must be converted to the REAL data type to minimize rounding errors.

5 Task

In this chapter, a function for analog control of the conveyor speed will be added to the program from chapter "SCE_EN_031-300 IEC Timers and Counters S7-1200".

6 Planning

The analog control of the conveyor speed will be programmed in the "MOTOR_SPEEDCONTROL" [FC10] function as an expansion of the "SCE_EN_031-300 IEC Timers and Counters S7-1200" project. This project must be retrieved from the archive in order to add this function. The "MOTOR_ SPEEDCONTROL" [FC10] function will be called in the "Main“ [OB1]" organization block and wired. The control of the conveyor motor must be changed to –Q3 (conveyor motor -M1 variable speed).

6.1 Analog control of the conveyor speed

The speed will be set at an input of the "MOTOR_SPEEDCONTROL" [FC10] function in revolutions per minute (range: +/- 50 rpm). The data type is 32-bit floating-point number (Real).

First, the function will be checked for correct entry of the speed setpoint in the range +/- 50 rpm.

If the speed setpoint is outside the range +/- 50 rpm, the value 0 with data type 16-bit integer (Int) will be output at the output. The return value of the function (Ret_Val) will then be assigned the value TRUE (1).

If the speed setting is within the range +/- 50 rpm, this value will first be normalized to the range 0…1 and then scaled to +/- 27648 with data type 16-bit integer (Int) for output as the speed manipulated value at the analog output.

10

The output will then be connected with signal U1 (manipulated value speed of the motor in 2 directions +/- 10V corresponds to +/- 50 rpm).

6.2 Technology diagram

Here you see the technology diagram for the task.

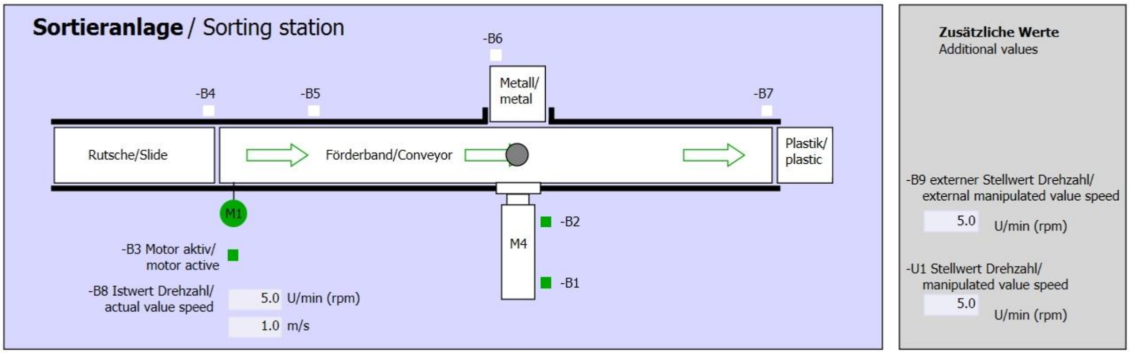


Figure 1: Technology diagram

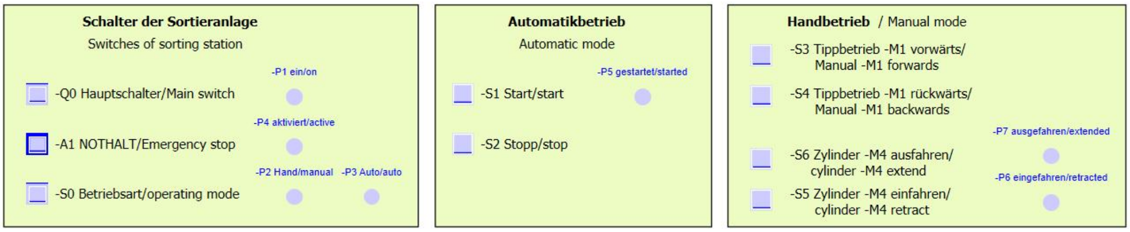


Figure 2: Control panel

10

6.3 Reference list

The following signals are required as global operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop OK	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	-S1	Pushbutton automatic start	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder -M4 retracted	NO
I 1.0	BOOL	-B4	Sensor part at slide	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO

DO	Type	Identifier	Function	
Q 0.2	BOOL	-Q3	Conveyor motor -M1 variable speed	
QW 64	BOOL	-U1	Manipulated value speed of the motor in 2 directions +/- 10V corresponds to +/- 50 rpm	

10

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

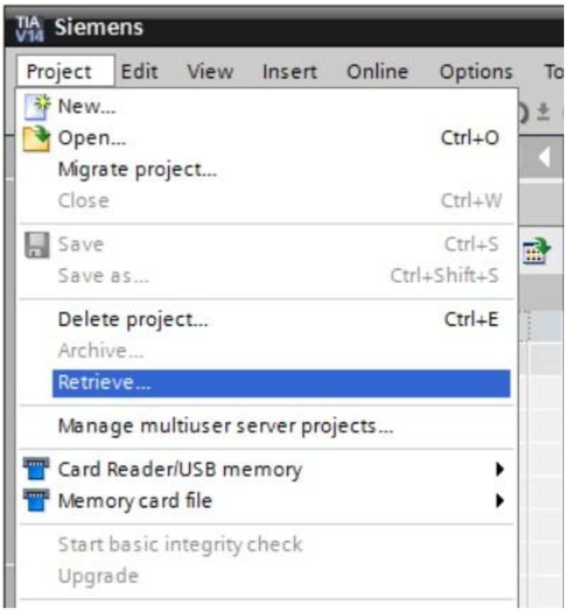
7 Structured step-by-step instructions

You can find instructions on how to carry out planning below. If you already have a good understanding of everything, it will be sufficient to focus on the numbered steps. Otherwise, simply follow the detailed steps in the instructions.

7.1 Retrieve an existing project

→ Before we can expand the "SCE_EN_031-300_IEC_Timers_Counters_S7-1200.zap14" project from chapter "SCE_EN_031-300_IEC_Timers_Counters_S7-1200", we must retrieve this project from the archive. To retrieve an existing project that has been archived, you must select the relevant archive with → Project → Retrieve in the project view. Confirm your selection with Open.

(→ Project → Retrieve → Select a .zap archive → Open)

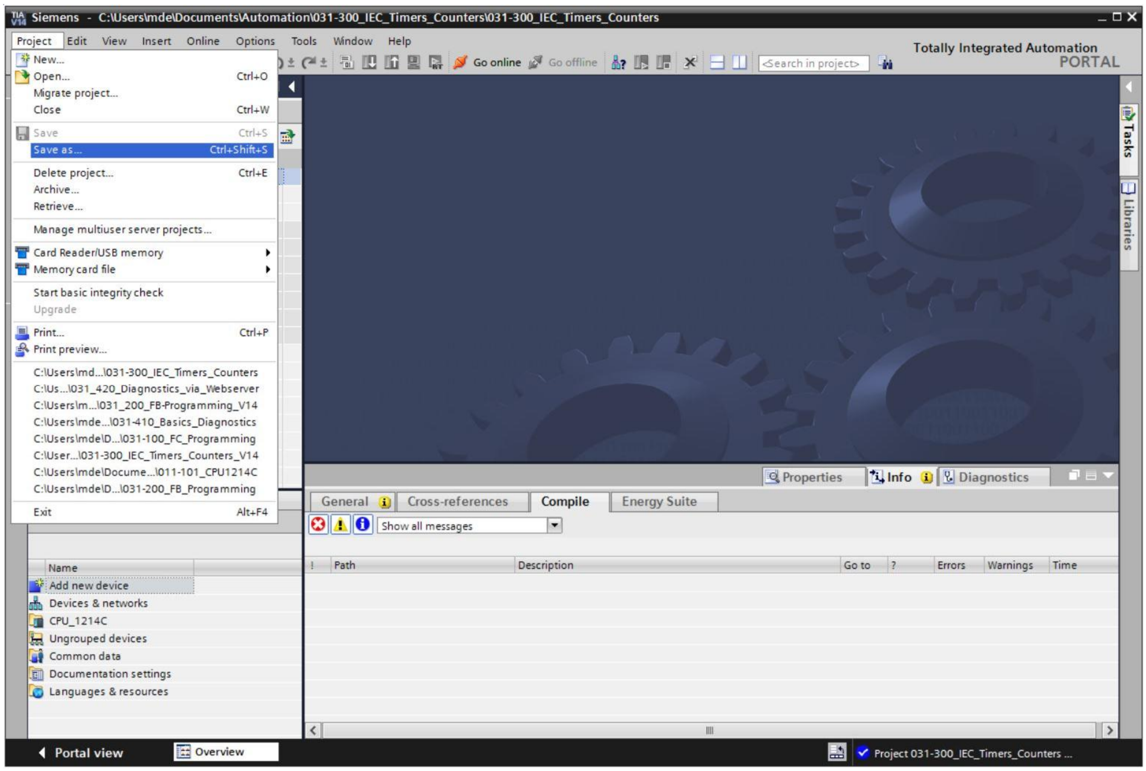


→ The next step is to select the target directory where the retrieved project will be stored. Confirm your selection with "OK".

(→ Target directory → OK)

10

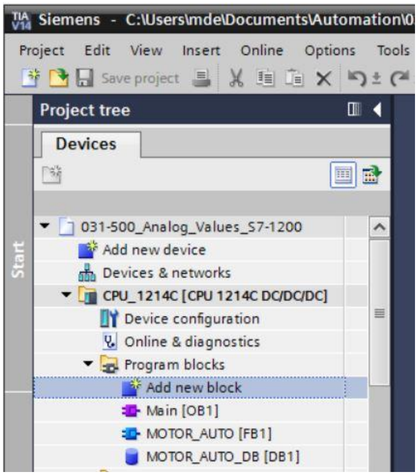
→ Save the opened project under the name 031-500_Analog_Values_S7-1200. (→ Project → Save as … → 031-500_Analog_Values_S7-1200 → Save)



7.2 Create the "MOTOR_SPEEDCONTROL" function

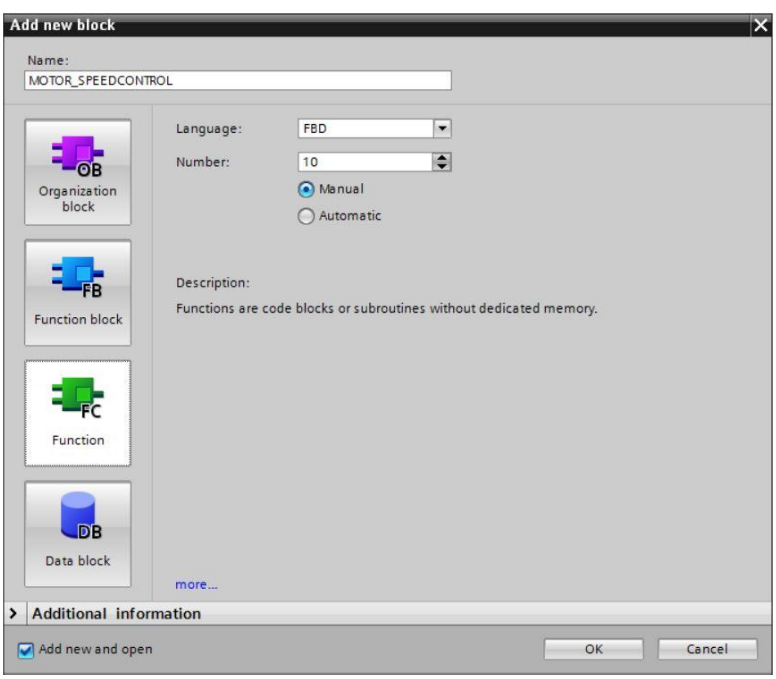
→ Select the 'Program blocks' folder of your CPU_1214C and then click "Add new block" to create a new function there.

(→ CPU_1214C [CPU 1214C DC/DC/DC] → Add new block)

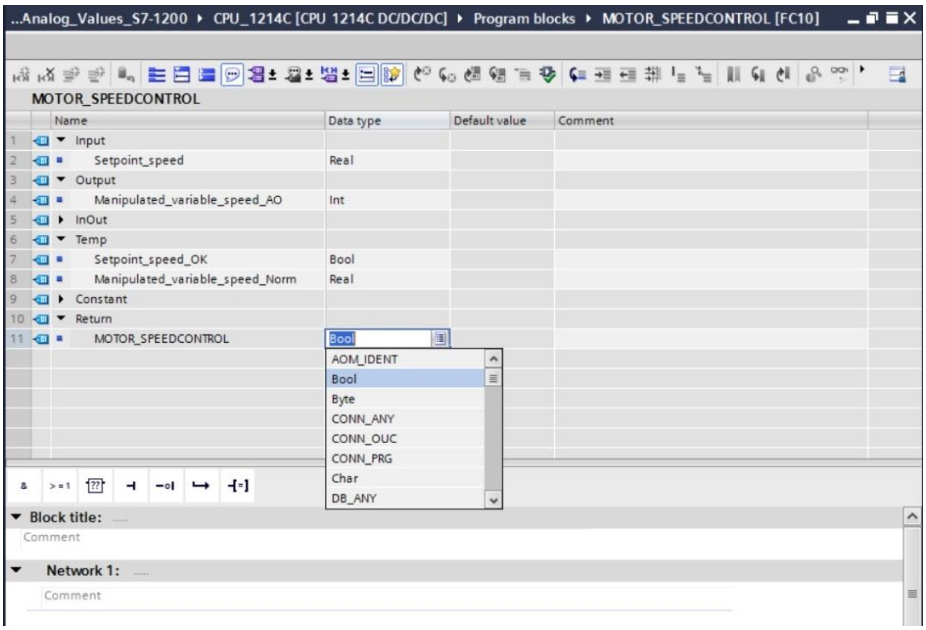


→ Select  in the next dialog and rename your new block to: "MOTOR_SPEEDCONTROL". Set the language to FBD and manually assign the number "10". Select the "Add new and open" check box. Click "OK".

(→  → Name: MOTOR_SPEEDCONTROL → Language: FBD → Number: 10 Manual → ☑ Add new and open → OK)



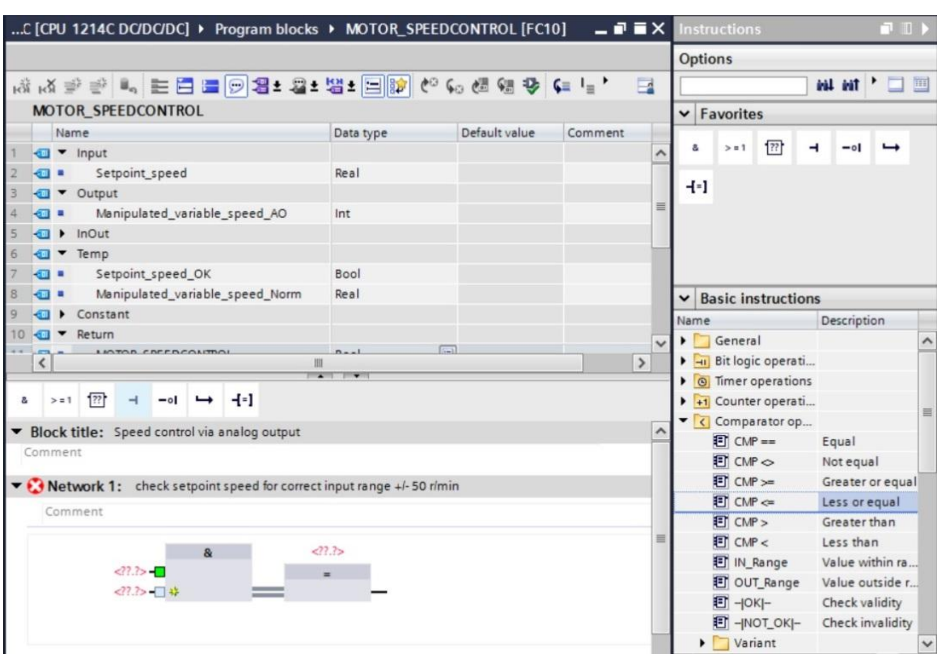
→ Create the local tags with their comments as shown here and change the data type of the 'Return' tag from 'Void' to 'Bool'. (→ Bool)



Note: *Be sure to use the correct data types.*

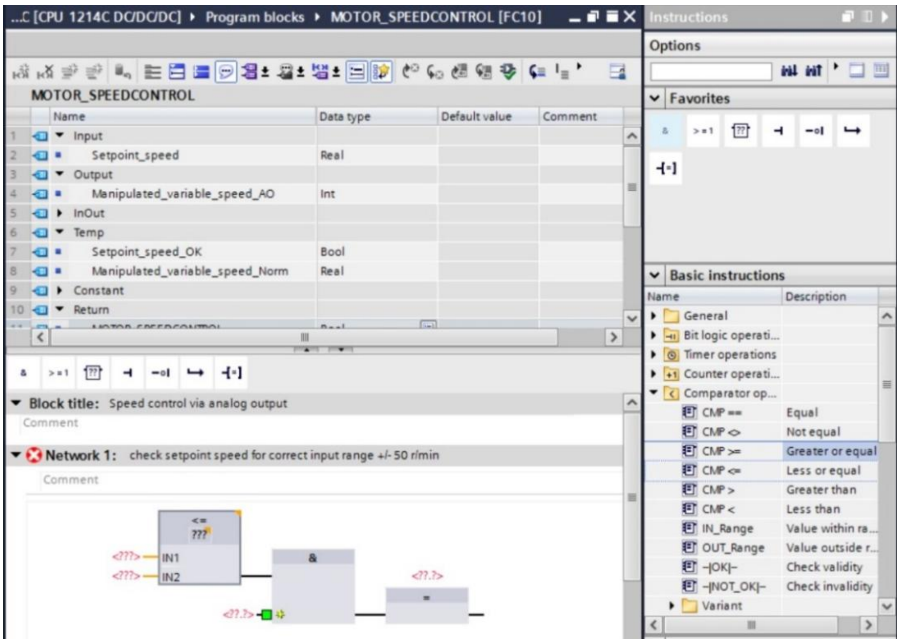
→ Insert an Assignment ' -[=] ' in the first network and an 'And' ' & ' in front of it. Then use drag & drop to move the 'Comparator operation' 'Less or equal' from the 'Basic instructions' onto the first input of the & AND logic operation.

(→ -[=] → & → Basic instructions → Comparator operations→ CMP<=)

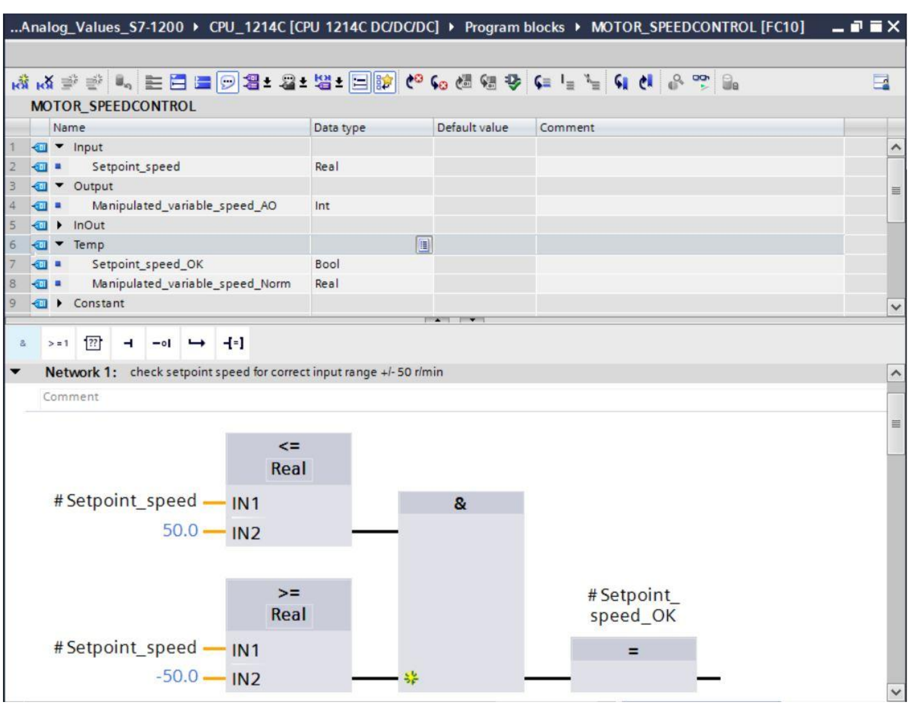


→ Next use drag & drop to move the 'Comparator operation' 'Greater or equal' onto the second input of the & AND logic operation.

(→ Basic instructions → Comparator operations→ CMP>=)

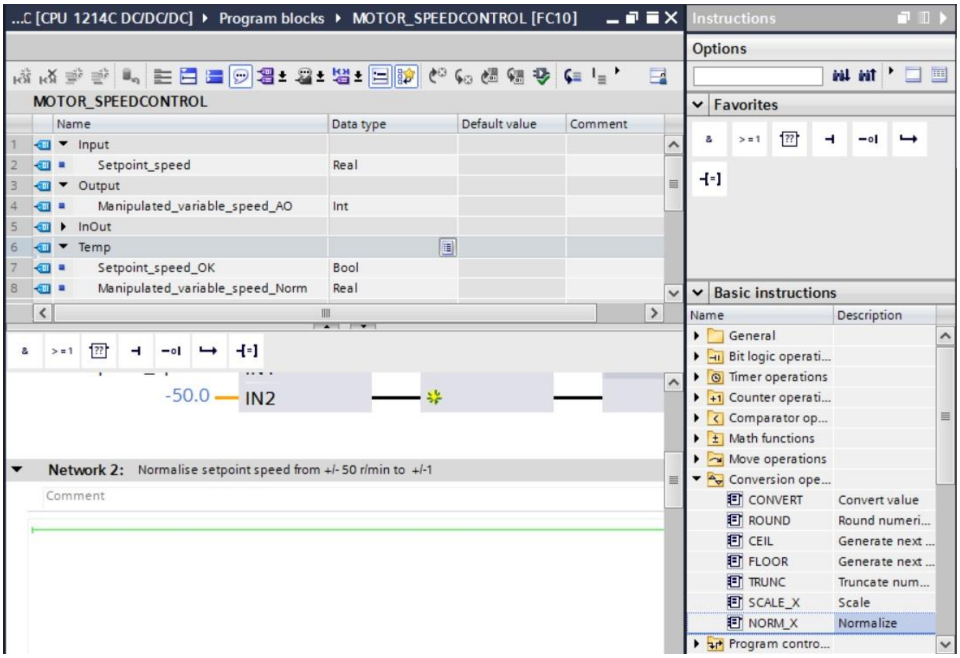


→ Connect the contacts in Network 1 with the constants and local tags as shown here. The data types in the comparator operations are automatically adapted to 'Real'.

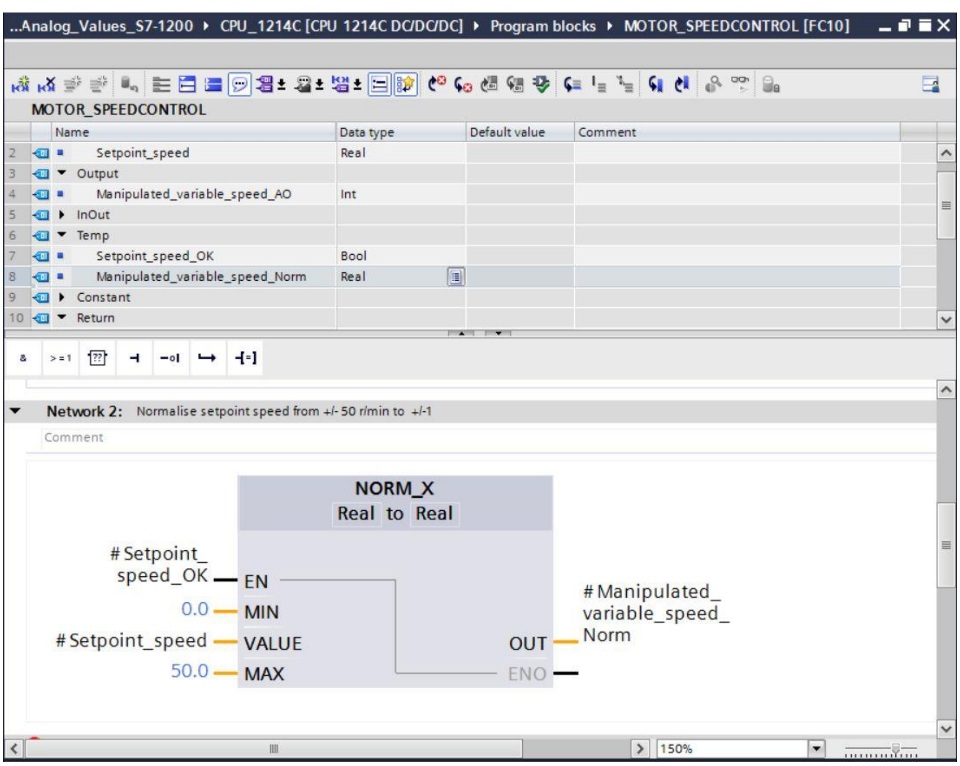


10

→ Use drag & drop to move the 'Conversion operation' 'NORM_X' into Network 2 in order to normalize the speed setpoint of +/- 50 rpm to +/- 1. (→ Basic instructions → Conversion operations → NORM_X)

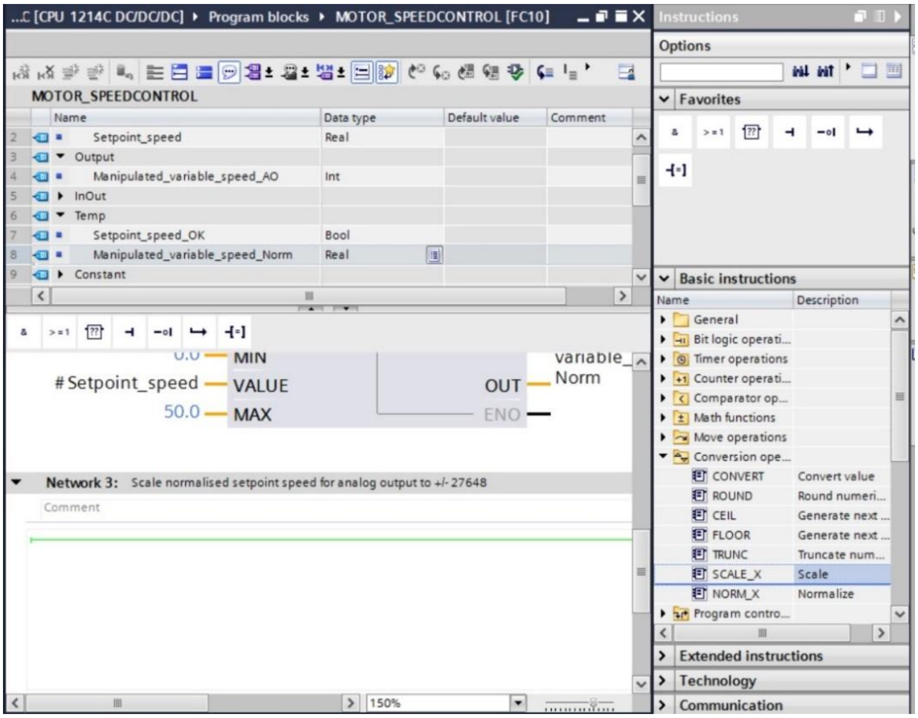


→ Connect the contacts in Network 2 with the constants and local tags as shown here. The data types in 'NORM_X' are automatically adapted to 'Real'.

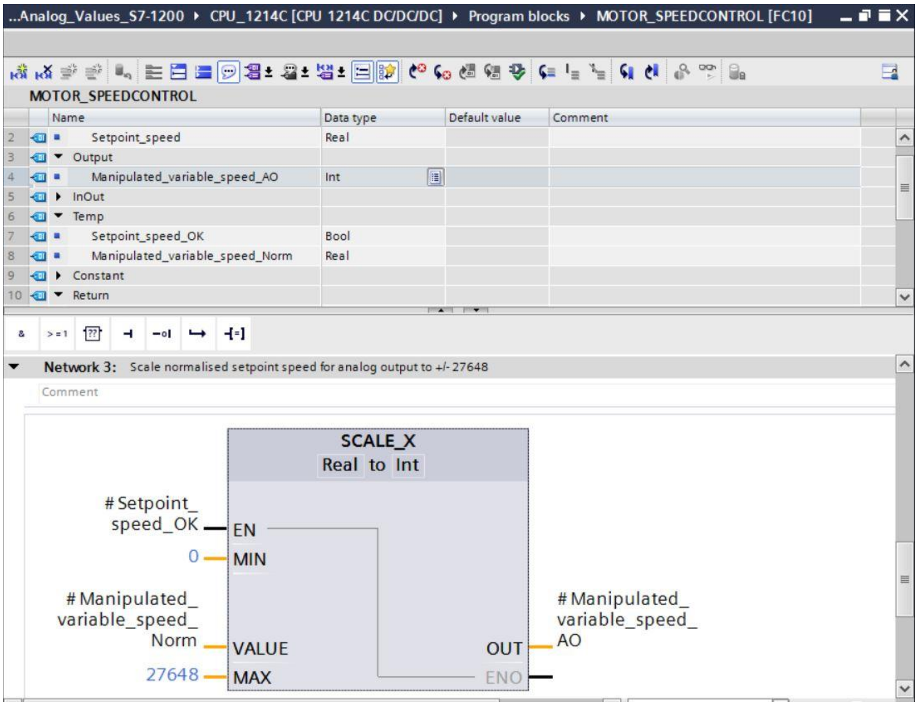


→ Use drag & drop to move the 'Conversion operation' 'SCALE_X' into Network 3 in order to scale the speed setpoint from the normalized +/- 1 onto the range for the analog output +/- 27468.

(→ Basic instructions → Conversion operations → SCALE_X)



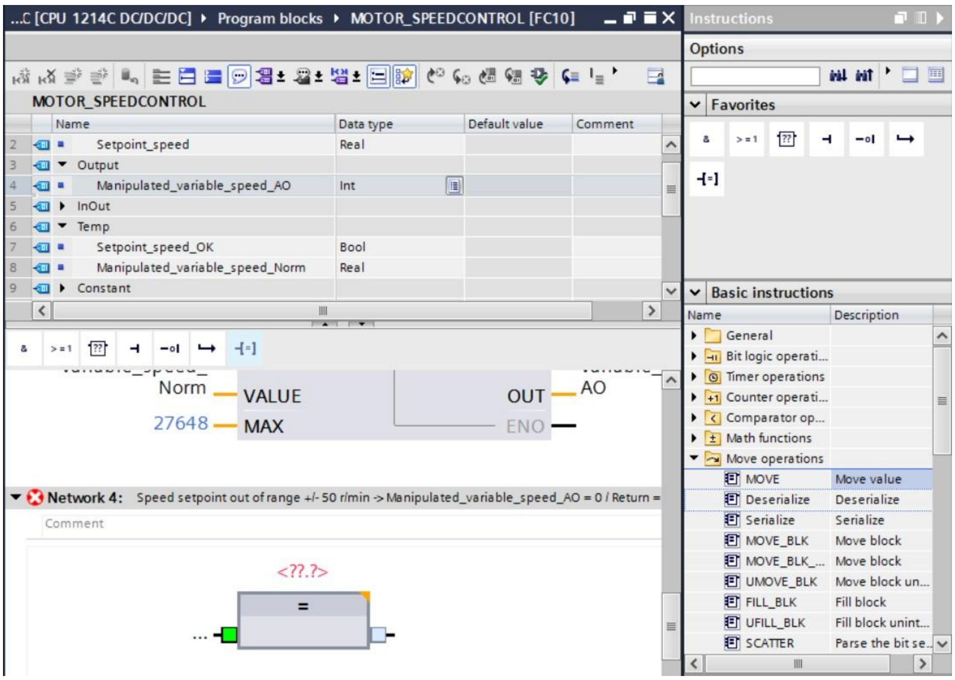
→ Connect the contacts with the constants and local tags in Network 3 as well, as shown here. The data types in 'SCALE_X' are automatically changed to 'Real' or 'Int'.



10

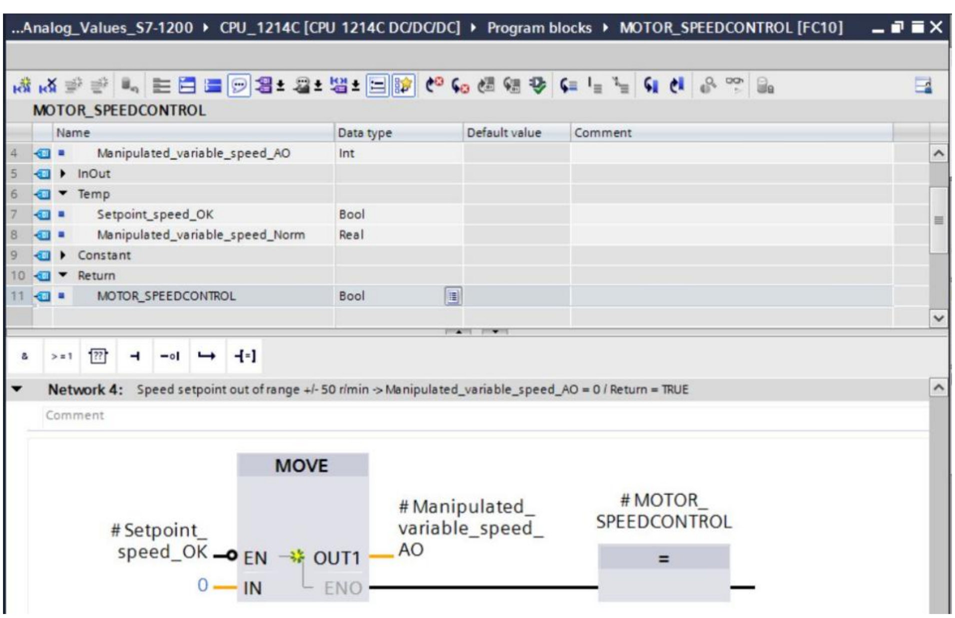
→ Insert an Assignment ' -[=] ' in the fourth network. Use drag & drop to move the 'Move' command from the 'Move operations' folder under 'Basic instructions' in front of the Assignment.

(→ -[=] → Basic instructions → Move operations → MOVE)



→ The contacts in Network 4 will now be connected with constants and local tags as shown here. If the speed setpoint is not within the range +/- 50 rpm, the value '0' is output at the analog output and the value TRUE is assigned to the return value (Return) of the "MOTOR_SPEEDCONTROL“ function.

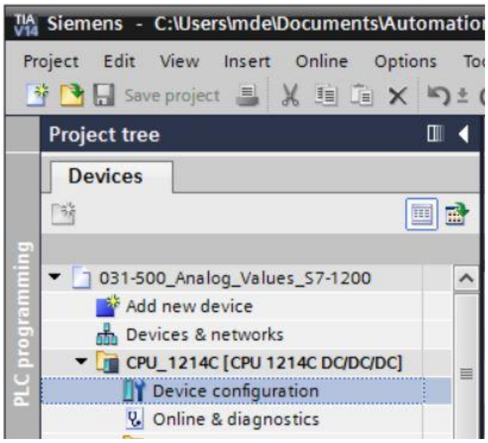
10



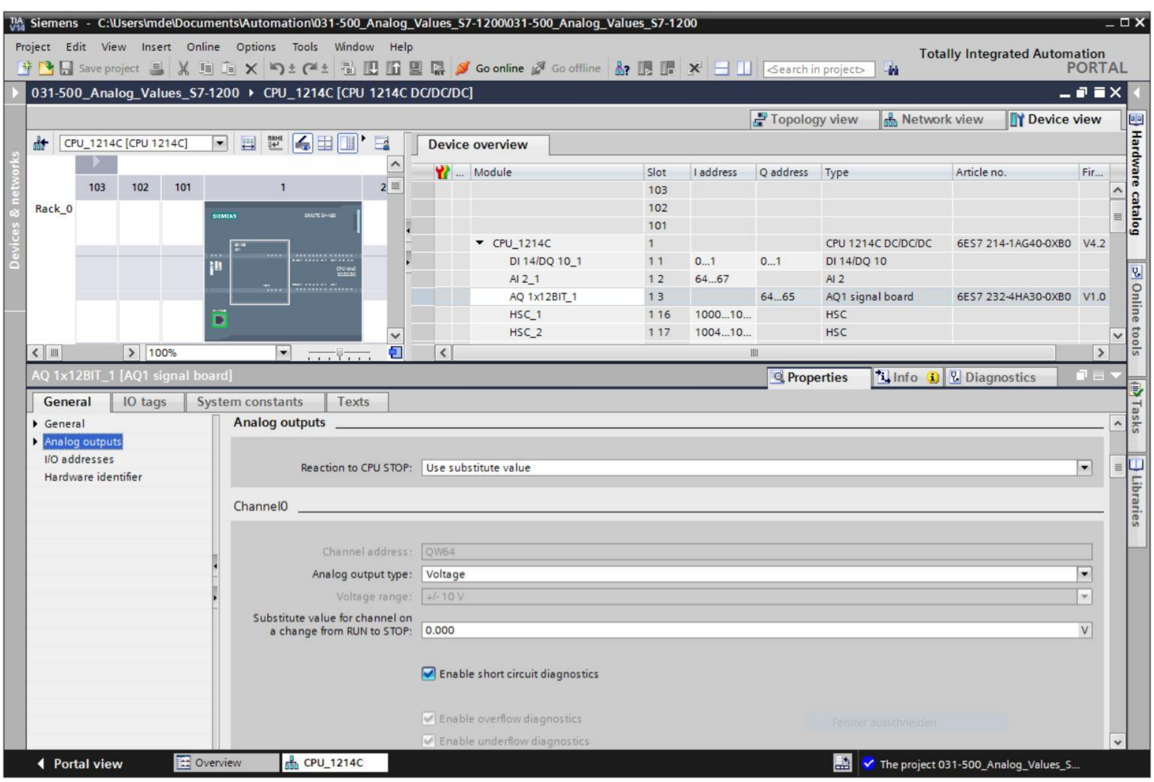
→ Do not forget to click Save project. The finished function "MOTOR_SPEEDCONTROL" [FC10] in FBD is shown below.

7.3 Configuration of the analog output channel

→ Double-click the 'Device configuration' to open it.

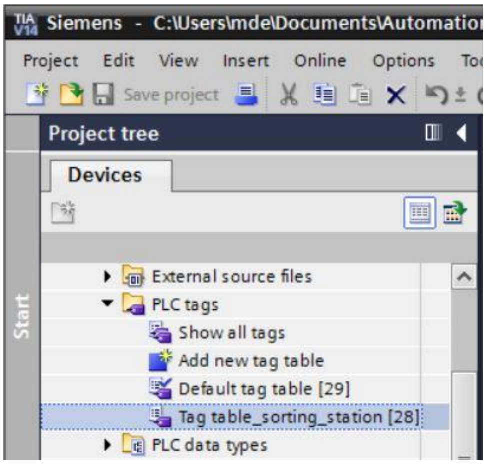


→ Check the address setting and the configuration of the analog output channel 0. (→ Q address: 64…65 →Properties → General → Analog outputs → Reaction to CPU STOP: Use substitute value → Channel 0 → Analog output type: Voltage → Substitute value for channel on a change from RUN to STOP: 0.000 V → ☑ Enable short circuit diagnostics)



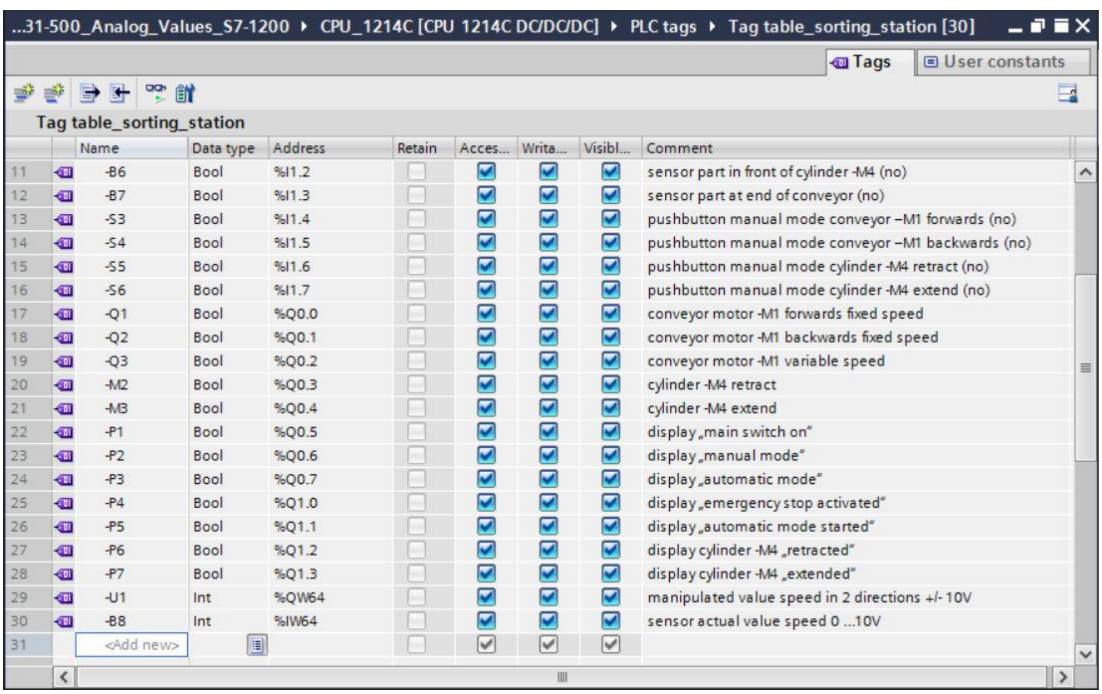
7.4 Expand the tag table to include analog signals

→ Double-click the 'Tag table_sorting station' to open it.



→ Add the global tags for the analog value processing to the "Tag table_sorting station". An analog input B8 and an analog output U1 must be added.

(→ U1 → %QW64 → B8 → %IW64)

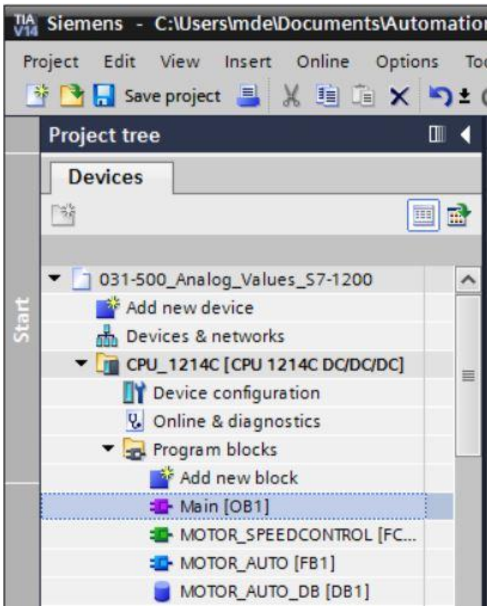


	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
11	-B6	Bool	%I1.2	☐	☑	☑	☑	sensor part in front of cylinder -M4 (no)
12	-B7	Bool	%I1.3	☐	☑	☑	☑	sensor part at end of conveyor (no)
13	-S3	Bool	%I1.4	☐	☑	☑	☑	pushbutton manual mode conveyor –M1 forwards (no)
14	-S4	Bool	%I1.5	☐	☑	☑	☑	pushbutton manual mode conveyor –M1 backwards (no)
15	-S5	Bool	%I1.6	☐	☑	☑	☑	pushbutton manual mode cylinder -M4 retract (no)
16	-S6	Bool	%I1.7	☐	☑	☑	☑	pushbutton manual mode cylinder -M4 extend (no)
17	-Q1	Bool	%Q0.0	☐	☑	☑	☑	conveyor motor -M1 forwards fixed speed
18	-Q2	Bool	%Q0.1	☐	☑	☑	☑	conveyor motor -M1 backwards fixed speed
19	-Q3	Bool	%Q0.2	☐	☑	☑	☑	conveyor motor -M1 variable speed
20	-M2	Bool	%Q0.3	☐	☑	☑	☑	cylinder -M4 retract
21	-M3	Bool	%Q0.4	☐	☑	☑	☑	cylinder -M4 extend
22	-P1	Bool	%Q0.5	☐	☑	☑	☑	display „main switch on"
23	-P2	Bool	%Q0.6	☐	☑	☑	☑	display „manual mode"
24	-P3	Bool	%Q0.7	☐	☑	☑	☑	display „automatic mode"
25	-P4	Bool	%Q1.0	☐	☑	☑	☑	display „emergency stop activated"
26	-P5	Bool	%Q1.1	☐	☑	☑	☑	display „automatic mode started"
27	-P6	Bool	%Q1.2	☐	☑	☑	☑	display cylinder -M4 „retracted"
28	-P7	Bool	%Q1.3	☐	☑	☑	☑	display cylinder -M4 „extended"
29	-U1	Int	%QW64	☐	☑	☑	☑	manipulated value speed in 2 directions +/- 10V
30	-B8	Int	%IW64	☐	☑	☑	☑	sensor actual value speed 0 ...10V
31	<Add new>			☐	☑	☑	☑	

10

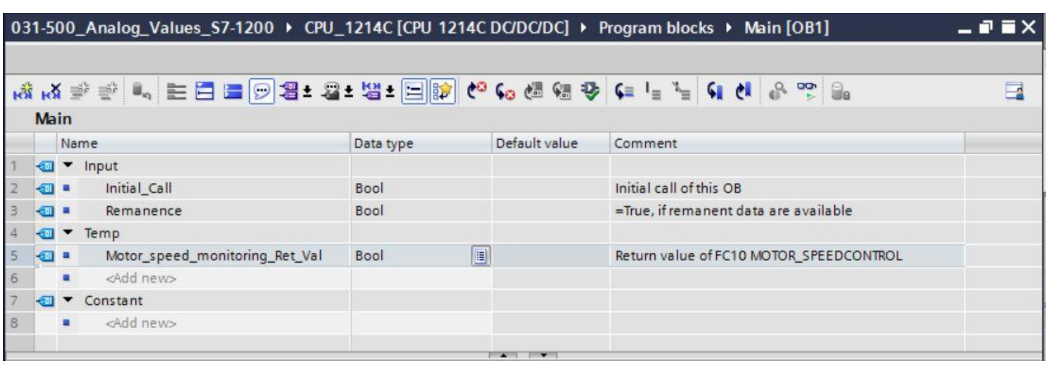
7.5 Call the block in the organization block

→ Open the "Main [OB1]" organization block with a double-click.

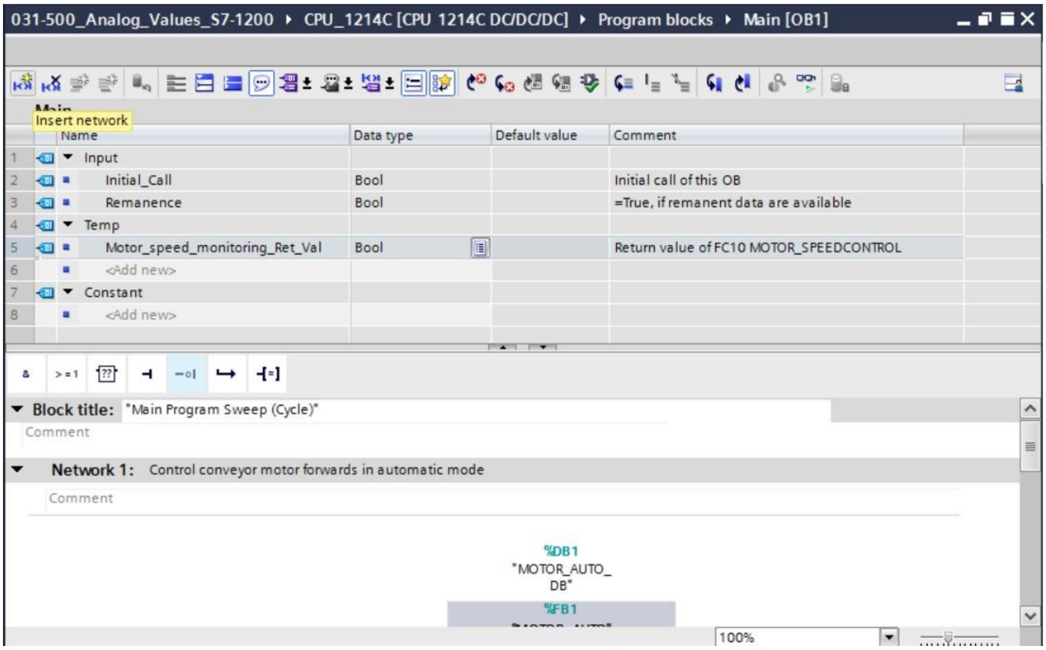


→ Add the temporary tag 'Motor_speed_monitoring_Ret_Val' to the local tags of OB1. These will be needed in order to interconnect the return value of the "MOTOR_SPEEDCONTROL" function.

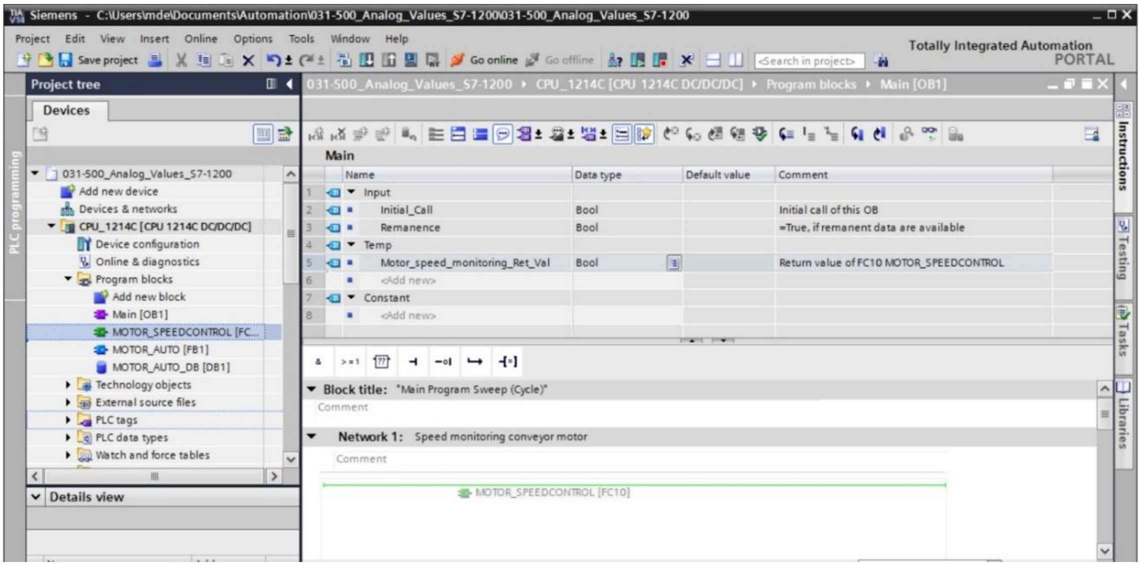
(→ Temp → Motor_speed_monitoring_Ret_Val → Bool)



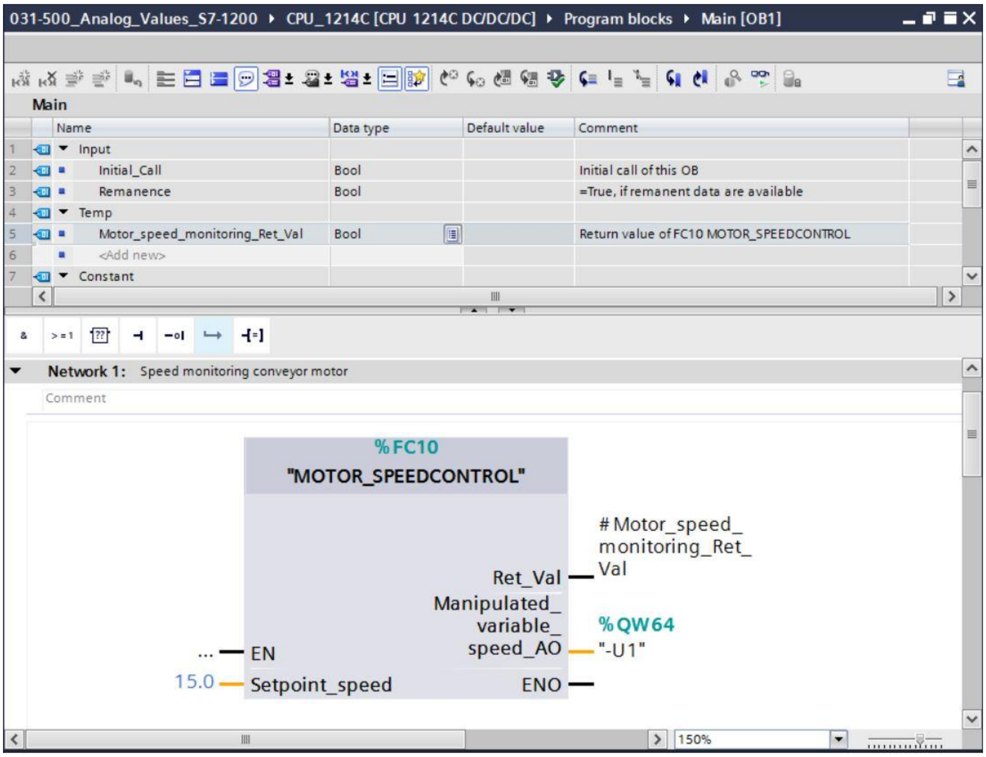
→ Select the block title of OB1 and then click '' to insert a new Network 1 in front of the other networks (→)



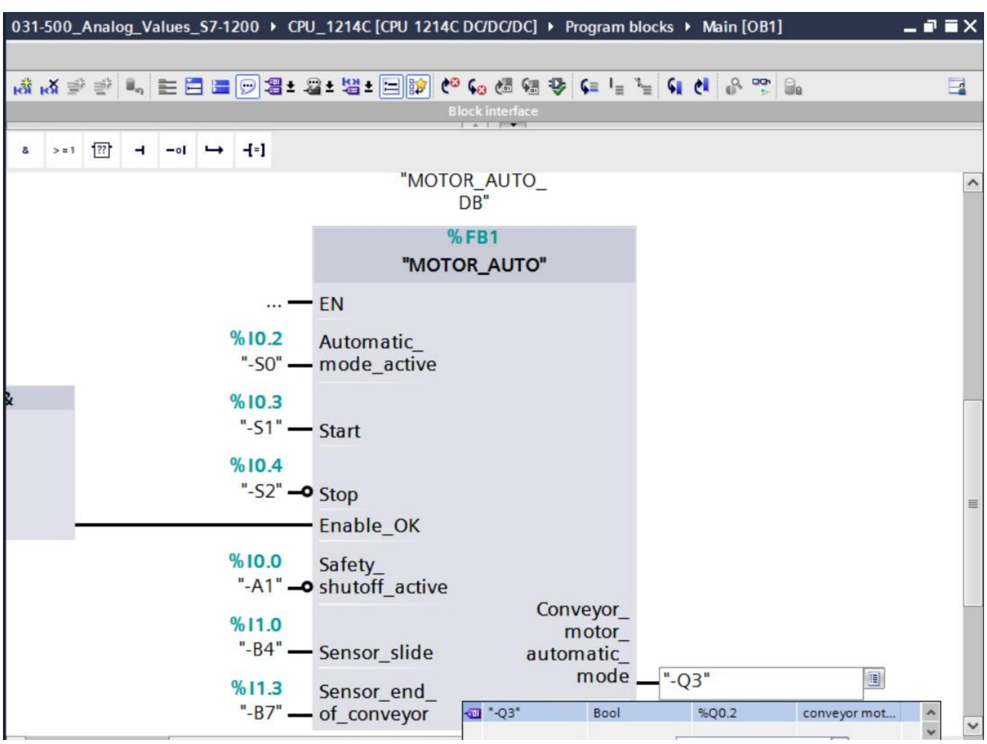
→ Use drag & drop to move your "MOTOR_SPEEDCONTROL [FC10]" function onto the green line in Network 1.



→ Connect the contacts with the constants and global and local tags here as shown.



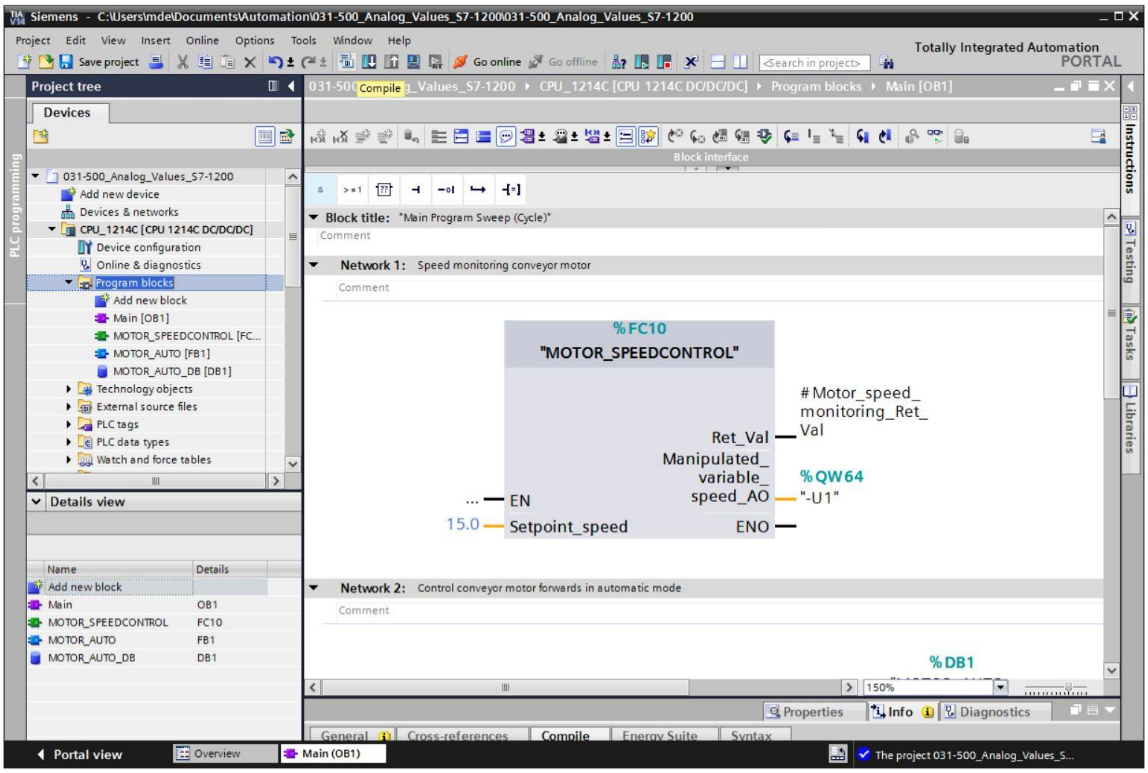
→ Change the connection of output tag "Conveyor_motor_automatic_mode" in Network 2 to '-Q3' (Conveyor motor -M1 variable speed) so that the conveyor motor is controlled taking the analog speed setting into consideration. (→ -Q3)



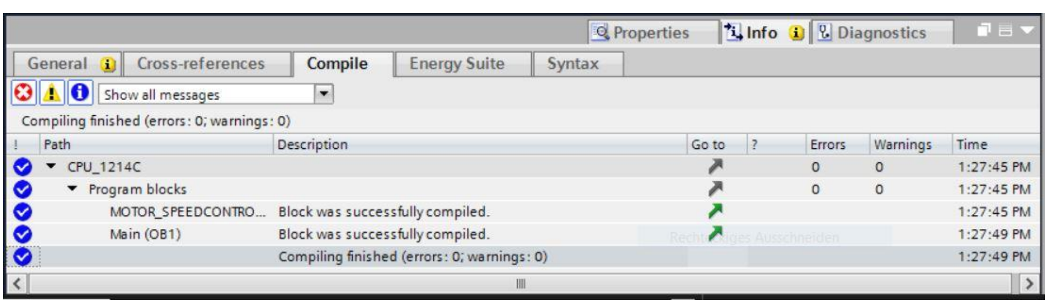
10

7.6 Save and compile the program

→ To save your project, select the Save project button in the menu. To compile all blocks, click the "Program blocks" folder and select the icon for compiling in the menu. (→ Save project → Program blocks →)



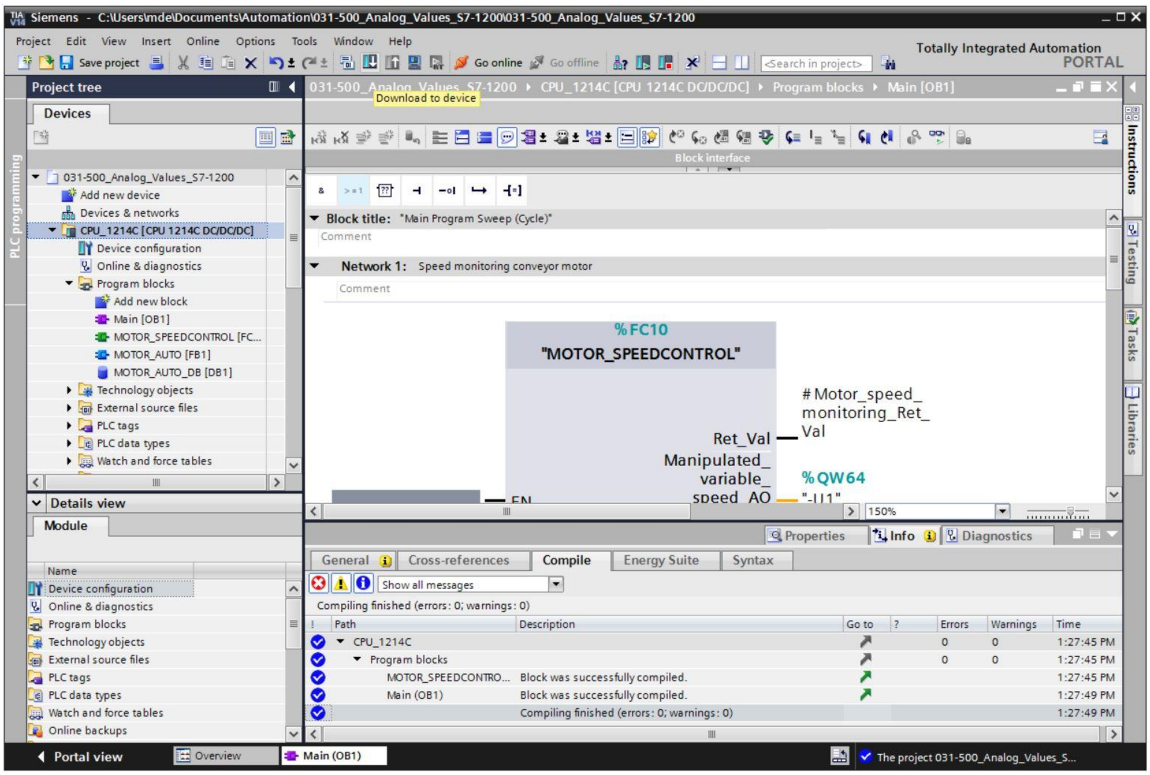
→ The "Info", "Compile" area shows which blocks were successfully compiled.



10

7.7 Download the program

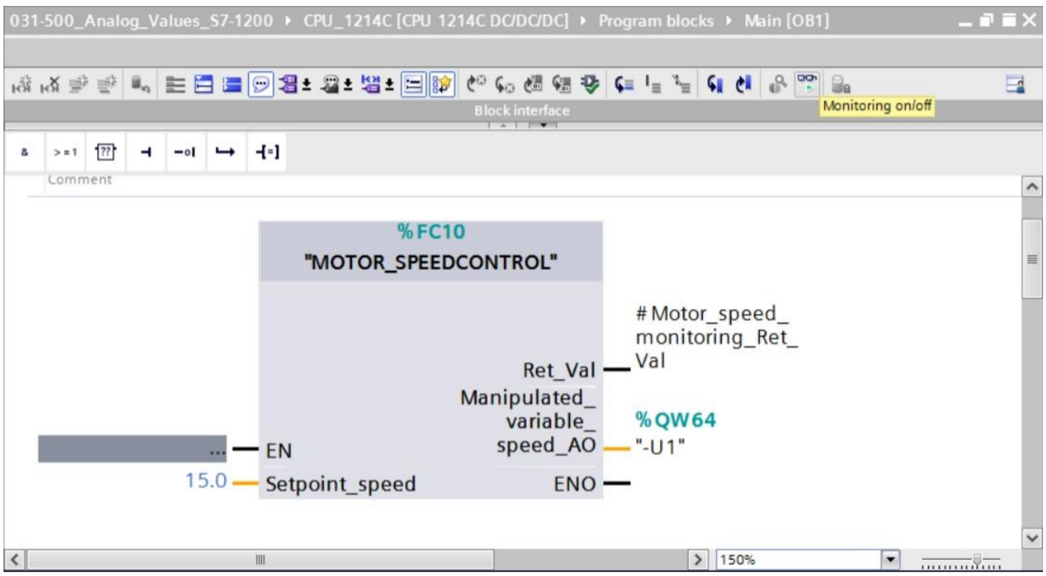
→ After successful compilation, the complete controller with the created program including the hardware configuration can, as described in the previous modules, be downloaded. (→ )

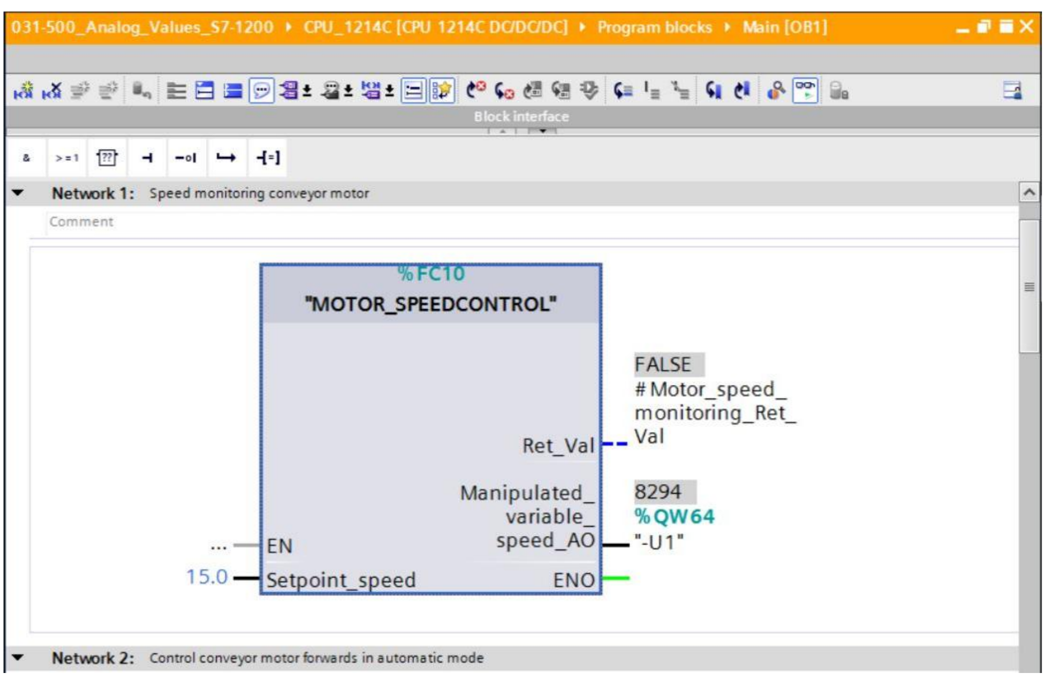


7.8 Monitor program blocks

→ The desired block must be open for monitoring the downloaded program. The monitoring can now be activated/deactivated by clicking the  icon.

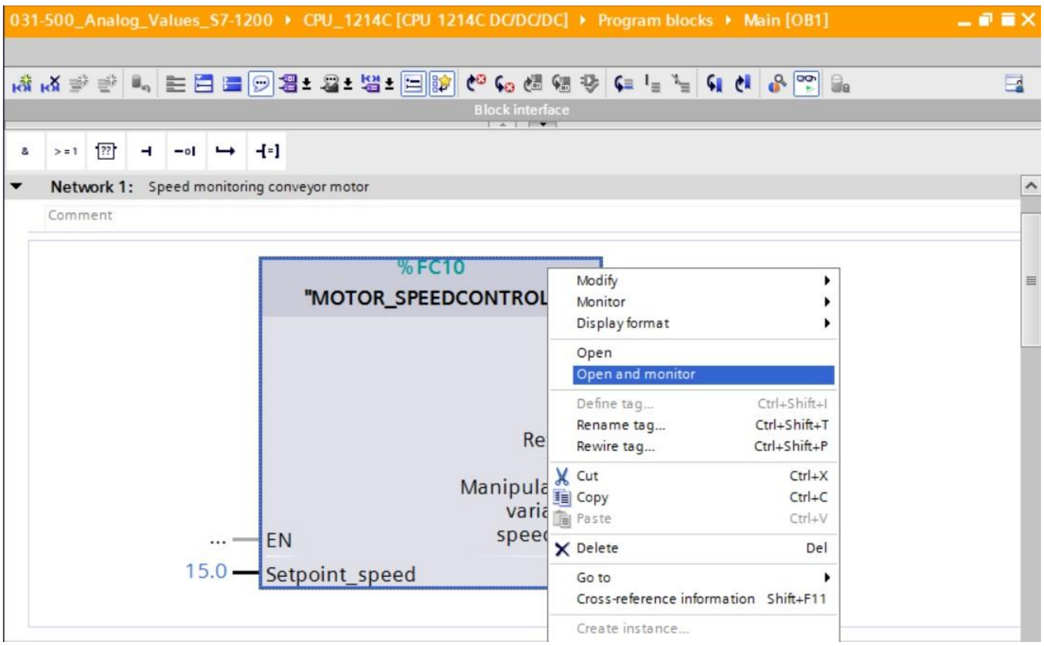
(→ Main [OB1] → )

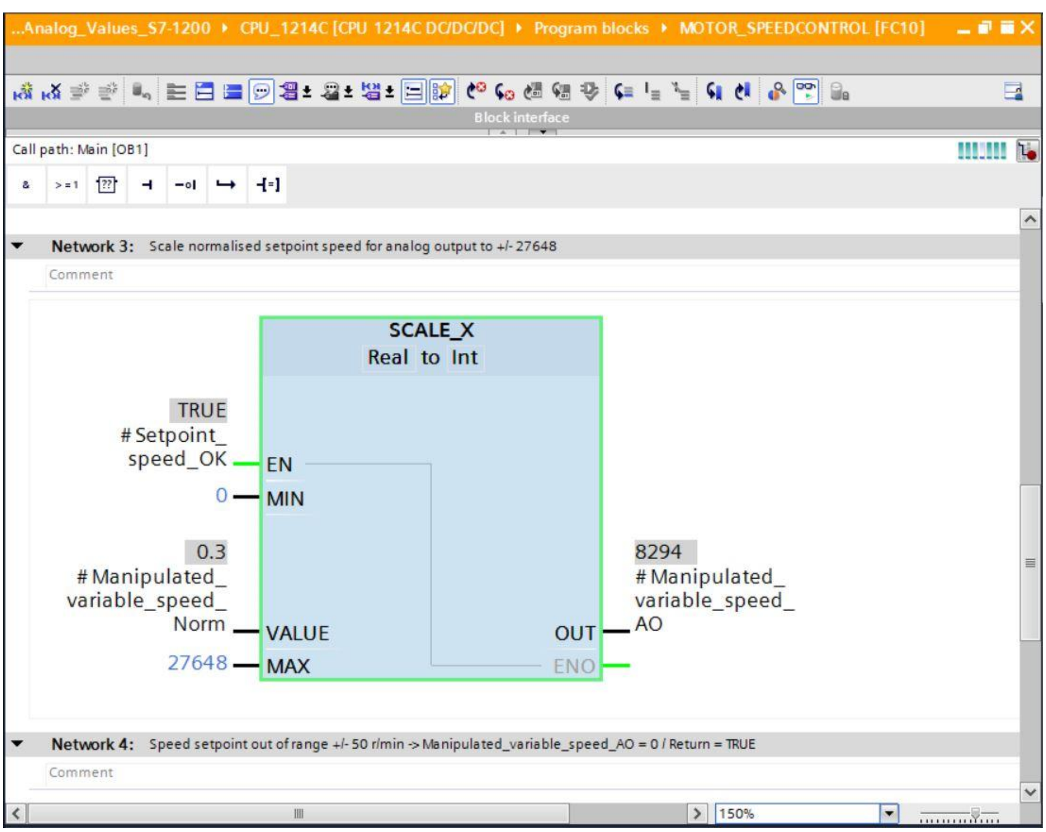




→ The "MOTOR_SPEEDCONTROL" [FC10] function called in the "Main [OB1]" organization block can be selected directly for "Open and monitor" after right-clicking and the program code in the function can thus be monitored.

(→ "MOTOR_SPEEDCONTROL" [FC10] → Open and monitor)

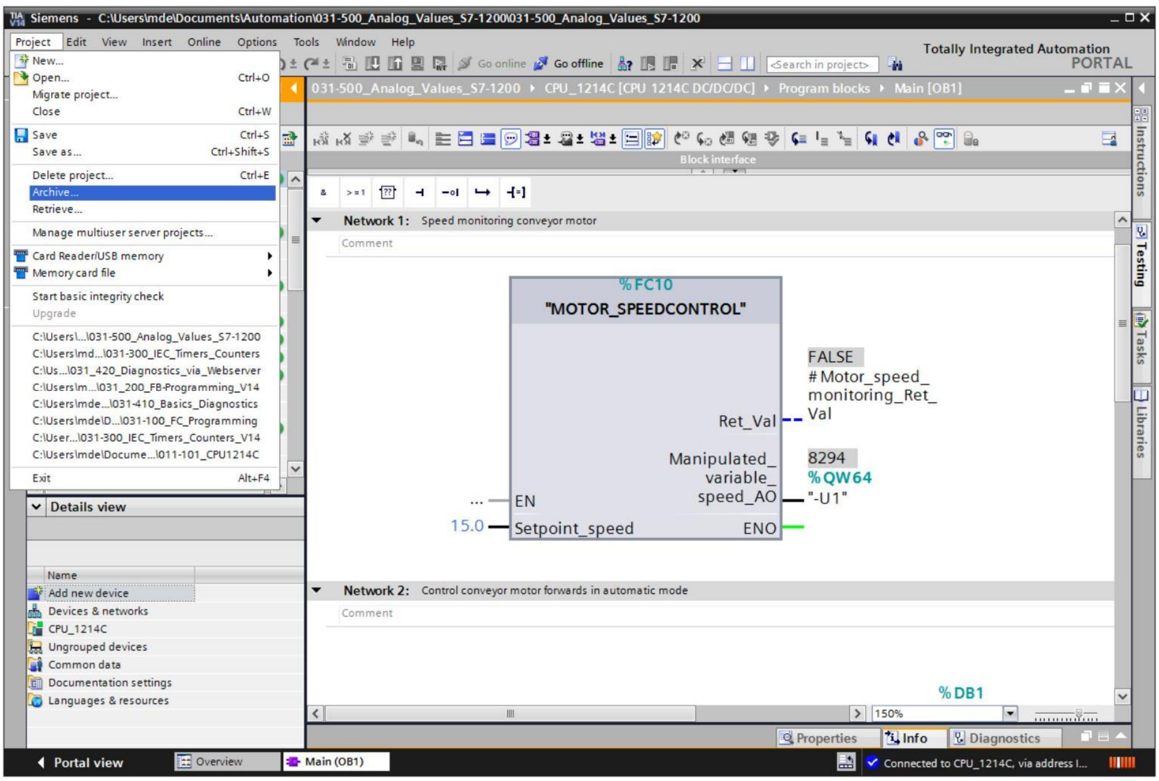




7.9 Archive the project

→ As the final step, we want to archive the complete project. Select the → 'Archive ...' command in the → 'Project' menu. Select a folder where you want to archive your project and save it with the file type "TIA Portal project archive".

(→ Project → Archive → TIA Portal project archive → 031-500_Analog_Values_S7-1200…. → Save)



10

8 Checklist

No.	Description	Completed
1	Compiling successful and without error message	
2	Download successful and without error message	
3	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated AUTOMATIC mode (-S0 = 1) Pushbutton automatic stop not actuated (-S2 = 1) Briefly press the automatic start pushbutton (-S1 = 1) Sensor part at slide activated (-B4 = 1) then Conveyor motor -M1 variable speed (-Q3 = 1) switches on and stays on. The speed corresponds to the speed setpoint in the range +/- 50 rpm	
4	Sensor part at end of conveyor activated (-B7 = 1) → -Q3 = 0 (after 2 seconds)	
5	Briefly press the automatic stop pushbutton (-S2 = 0) → -Q3 = 0	
6	Activate EMERGENCY OFF (-A1 = 0) → -Q3 = 0	
7	Manual mode (-S0 = 0) → -Q3 = 0	
8	Switch off station (-K0 = 0) → -Q3 = 0	
9	Cylinder not retracted (-B1 = 0) → -Q3 = 0	
10	Project successfully archived	

9 Exercise

9.1 Task – Exercise

In this exercise a "MOTOR_SPEEDMONITORING" [FC11] function will be created additionally.

The actual value will be made available to B8 (Sensor actual value speed of the motor +/-10V corresponds to +/- 50 rpm) as an analog value and queried at an input of the "MOTOR_SPEEDMONITORING" [FC11] function. The data type is 16-bit integer (Int).

This actual speed value will first be normalized to the range +/- 1 as 32-bit floating-point number (Real) in the function.

The normalized actual speed value will then be scaled to revolutions per minute (range: +/- 50 rpm) as 32-bit floating-point number (Real) and made available at an output.

The following 4 limit values can be specified as 32-bit floating-point numbers (Real) at the block inputs in order to monitor them in the function:

Speed > Motor_speed_monitoring_error_max

Speed > Motor_speed_monitoring_warning_max

Speed < Motor_speed_monitoring_warning_min

Speed < Motor_speed_monitoring_error_min

If a limit value is exceeded or fallen below, the value TRUE (1) is assigned to the corresponding output bit.

If a fault is present, the protective tripping of the "MOTOR_AUTO" [FB1] function block will be tripped.

10

9.2 Technology diagram

Here you see the technology diagram for the task.

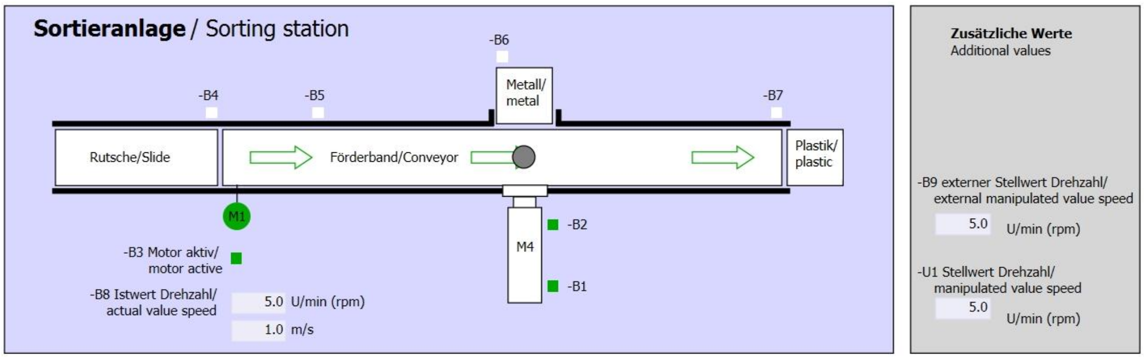


Figure 3: Technology diagram

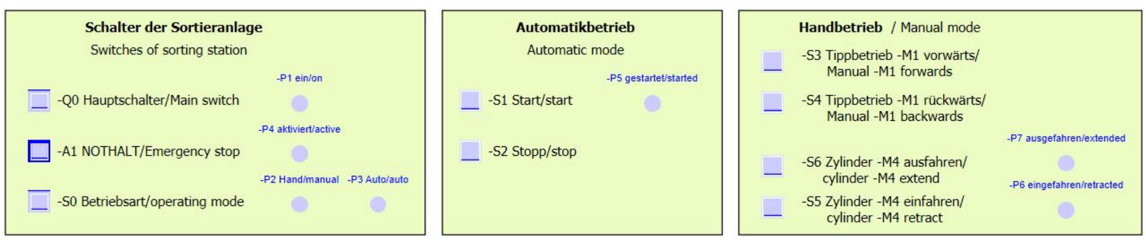


Figure 4: Control panel

10

9.3 Reference list

The following signals are required as global operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop OK	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	-S1	Pushbutton automatic start	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder -M4 retracted	NO
I 1.0	BOOL	-B4	Sensor part at slide	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO
IW64	BOOL	-B8	Sensor actual value speed of the motor +/-10V corresponds to +/- 50 rpm	

DO	Type	Identifier	Function	
Q 0.2	BOOL	-Q3	Conveyor motor -M1 variable speed	
QW 64	BOOL	-U1	Manipulated value speed of the motor in 2 directions +/- 10V corresponds to +/- 50 rpm	

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

9.4 Planning

Plan the implementation of the task on your own.

9.5 Checklist – Exercise

No.	Description	Completed
1	Compiling successful and without error message	
2	Download successful and without error message	
3	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated AUTOMATIC mode (-S0 = 1) Pushbutton automatic stop not actuated (-S2 = 1) Briefly press the automatic start pushbutton (-S1 = 1) Sensor part at slide activated (-B4 = 1) then Conveyor motor M1 variable speed (-Q3 = 1) switches on and stays on. The speed corresponds to the speed setpoint in the range +/- 50 rpm	
4	Sensor part at end of conveyor activated (-B7 = 1) → -Q3 = 0 (after 2 seconds)	
5	Briefly press the automatic stop pushbutton (-S2 = 0) → -Q3 = 0	
6	Activate EMERGENCY OFF (-A1 = 0) → -Q3 = 0	
7	Manual mode (-S0 = 0) → -Q3 = 0	
8	Switch off station (-K0 = 0) → -Q3 = 0	
9	Cylinder not retracted (-B1 = 0) → -Q3 = 0	
10	Speed > Motor_speed_monitoring_error_max → -Q3 = 0	
11	Speed < Motor_speed_monitoring_error_min → -Q3 = 0	
12	Project successfully archived	

10 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

10

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Please note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is provided at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, please contact your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

The SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

11

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

Table of contents

Global Data Blocks for the SIMATIC S7-1200

1 Goal

In this chapter, you will become acquainted with the use of global data blocks for the SIMATIC S7-1200 with the TIA Portal programming tool.

The module explains the structure and creation of and access to global data blocks for the SIMATIC S7-1200. It also shows the steps for creating a global data block in the TIA Portal and for accessing this data in the program with read and write access.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

This chapter builds on the chapter Analog Values with the SIMATIC S7 CPU1214C DC/DC/DC. You can use the following project for this chapter, for example: "SCE_EN_031-500_Analog_Values_S7-1200.zap14".

3 Required hardware and software

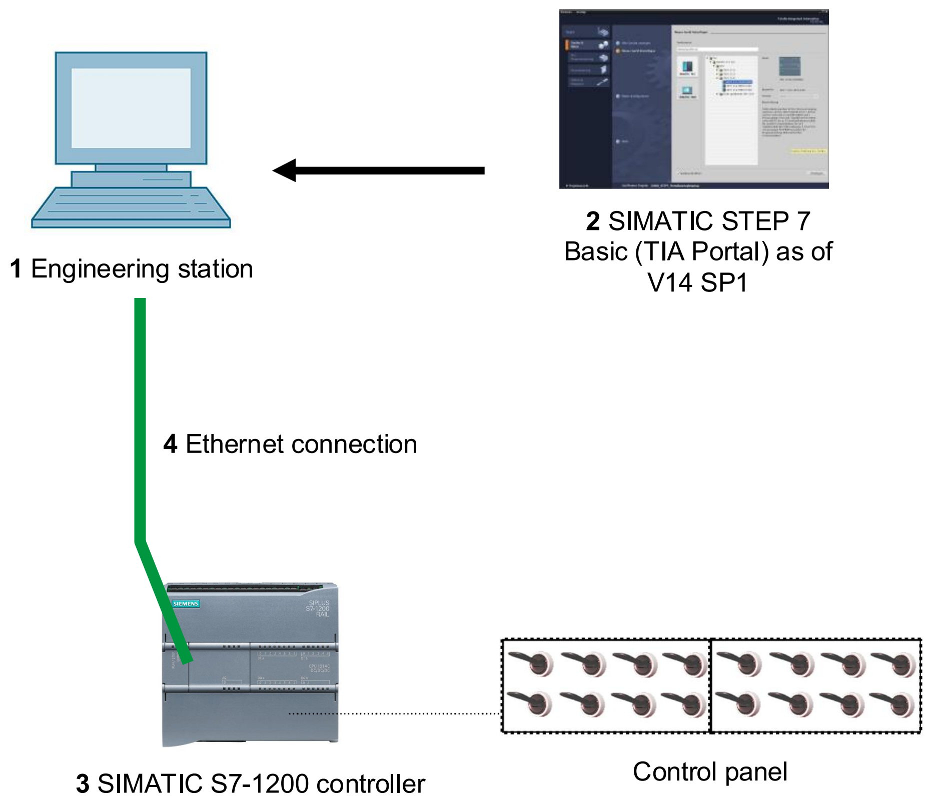
1 Engineering station: requirements include hardware and operating system
(for additional information, see Readme on the TIA Portal Installation DVDs)

2 SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1

Note: The digital inputs and analog inputs and outputs should be fed out to a control panel.

4 Ethernet connection between engineering station and controller



4 Theory

4.1 Data blocks

In contrast to logic blocks, data blocks contain no instructions. Rather, they serve as memory for user data.

Data blocks thus contain variable data that is used by the user program. You can define the structure of global data blocks as required.

Global data blocks store data that can be used ***by all other blocks*** (see Figure 1). Only the associated function block should access instance data blocks. The maximum size of data blocks varies depending on the utilized CPU.

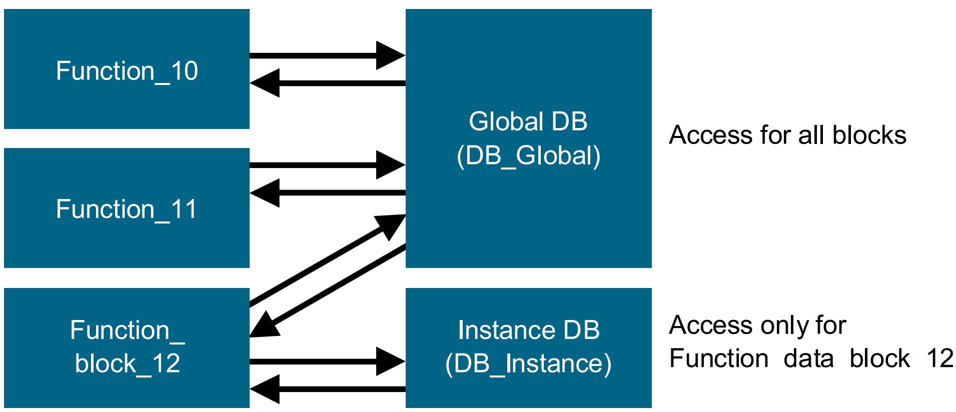


Figure 1: Difference between global DB and instance DB.

Application examples for **global data blocks** are:

– Saving of information about a storage system. "Which product is located where?"

– Saving of recipes for particular products.

The data in data blocks is stored retentively in most cases. This data is then retained in the event of a power failure or after a STOP/START of the CPU.

4.2 Data types of the SIMATIC S7-1200

The SIMATIC S7-1200 has many different data types for representing different numerical formats. A list of some of the elementary data types is given below.

Data type	Size (bits)	Range	Example of constant entry
Bool	1	0 to 1	TRUE, FALSE, O, 1
Byte	8	16#00 to 16#FF	16#12, 16#AB
Word	16	16#0000 to 16#FFFF	16#ABCD, 16#0001
DWord	32	16#00000000 to 16#FFFFFFFF	16#02468ACE
Char	8	16#00 to 16#FF	'A', 'r', '@'
Sint	8	-128 to 127	123,-123
Int	16	-32,768 to 32,767	123, -123
Dint	32	-2,147,483,648 to 2,147,483,647	123, -123
USInt	8	0 to 255	123
UInt	16	0 to 65,535	123
UDInt	32	0 to 4,294,967,295	123
Real	32	+/-1.18 x 10^{-38} to +/-3.40 x 10^{38}	123.456, -3.4, 1.2E+12 3.4E-3
LReal	64	+/-2.23 x 10^{-308} to +/-1.79 x 10^{308}	12345.123456789 -1.2E+40
Time	32	T#-24d_20h_31 m_23s_648ms to T#24d_20h_31 m_23s_647ms Saved as: -2,147,483,648 ms to +2,147,483,647 ms	T#5m_30s 5#-2d T#1d_2h_15m_30x_45ms
String	Variable	0 to 254 characters in byte size	'ABC'
Array		With arrays, data of a uniform data type is arranged one after the other and addressed consecutively in the address area. The properties of each array element are identical and are configured in the array tag.	
Struct		The STRUCT data type represents a data structure that consists of a fixed number of components of different data types. Components of STRUCT or ARRAY data type can also be nested in a structure.	
…		For other data types, refer to the online help.	

11

4.3 Optimized blocks

S7-1200 controllers have optimized data storage. In optimized blocks all tags are automatically sorted based on their data type. The sorting ensures that data gaps between the tags are minimized and the tags are stored in a manner that optimizes their access by the controller.

– The tags are always accessed as fast as possible because the file storage by the system is optimized and is independent of the declaration.

– There is no danger of inconsistencies due to incorrect, absolute accesses because symbolic access is generally used.

– Declaration changes do not result in access errors because accesses by process visualization systems, for example, occur symbolically.

– Individual tags can be selectively defined as retentive.

– No settings are needed or possible in the instance data block. Everything will be set in the assigned FB (e.g., retentivity).

– Memory reserves in the data block enable changes to be made without loss of actual values (download without reinitialization).

4.4 Downloading without reinitialization

To enable the subsequent editing of user programs that are already running in a CPU, the S7-1200 controllers support the option of expanding the interfaces of optimized function or data blocks during operation. You can download the modified blocks without switching the controller to STOP mode and without affecting the actual values of previously downloaded tags.

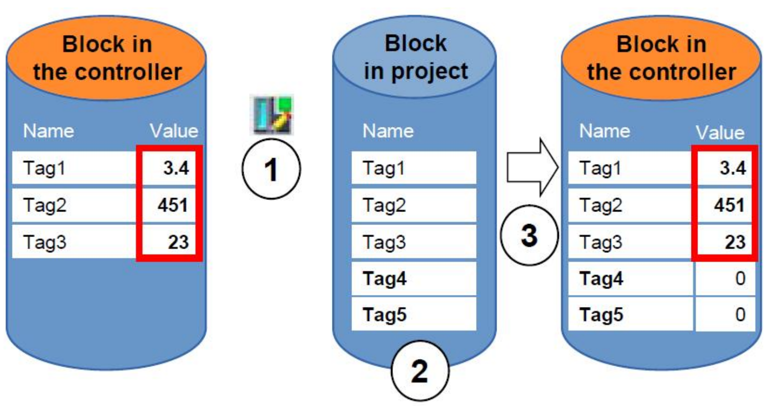


Figure 2: Download without reinitialization

The following steps can be performed while the controller is in RUN mode:

1. Activate "Download without reinitialization"

2. Insert newly defined tags in an existing block

3. Download expanded block to the controller

The newly defined tags are initialized. The existing tags retain their current value.

Prerequisite: a memory reserve must have been defined for the block beforehand and the block with this memory reserve must have downloaded to the CPU.

5 Task

In this chapter, the program from chapter "SCE_EN_031-500 Analog Values_S7-1200“ will be expanded to include a data block that centrally provides the parameters for the two functions "MOTOR_SPEEDCONTROL“ [FC10] and "MOTOR_ SPEEDMONITORING“ [FC11].

6 Planning

The data management and setpoint setting for the "MOTOR_SPEEDCONTROL" [FC10] and "MOTOR_SPEEDMONITORING" [FC11] functions will be carried out using the global data block "SPEED_MOTOR" [DB2].

This will be added to the "031-500_Analog_Values_S7-1200" project. This project must be retrieved from the archive beforehand.

In the "Main" [OB1] organization block, the two functions "MOTOR_SPEEDCONTROL" [FC10] and "MOTOR_SPEEDMONITORING" [FC11] must then be connected with the tags from global data block "SPEED_MOTOR" [DB2].

6.1 Global data block for speed control and speed monitoring of the motor

Speed setpoint and actual speed value will be created in Real data format (32-bit floating-point number) as the first tags in the "SPEED_MOTOR" [DB2] data block. The speed setpoint is thereby given the start value + 10 rpm.

A structure (Struct) 'Positive_Speed' will then be created for monitoring the positive speed limits.

This structure contains the 2 tags 'Threshold_Error' (start value + 15 rpm) and 'Threshold_Warning' (start value + 10 rpm) in Real data format (32-bit floating-point number) and the 2 tags 'Error' and 'Warning' in Bool data format (binary number).

The structure (Struct) 'Positive_Speed' will then be inserted again as a copy and renamed to 'Negative_Speed' for monitoring the negative speed limits.

The 'Threshold_Error' tag is given the start value - 16 rpm and the 'Threshold_Warning' tag the start value - 14 rpm.

6.2 Technology diagram

Here you see the technology diagram for the task.

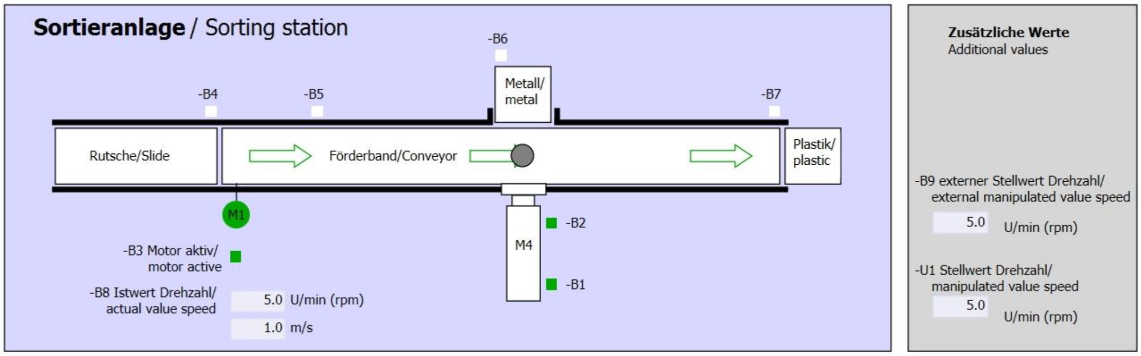


Figure 3: Technology diagram

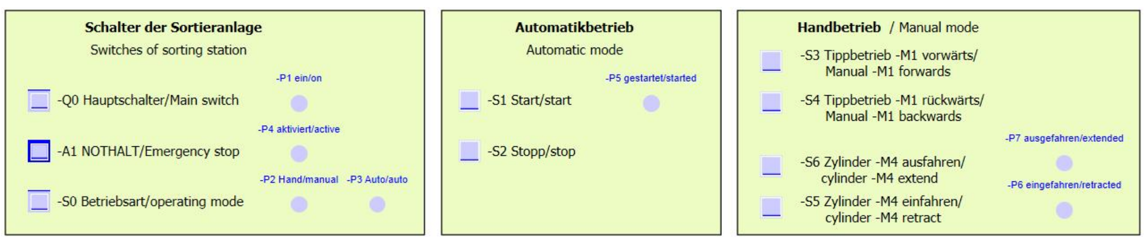


Figure 4: Control panel

11

6.3 Reference list

The following signals are required as global operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop OK	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	-S1	Pushbutton automatic start	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder -M4 retracted	NO
I 1.0	BOOL	-B4	Sensor part at slide	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO
IW64	BOOL	-B8	Sensor actual value speed of the motor +/-10V corresponds to +/- 50 rpm	

DO	Type	Identifier	Function	
Q 0.2	BOOL	-Q3	Conveyor motor -M1 variable speed	
QW 64	BOOL	-U1	Manipulated value speed of the motor in 2 directions +/- 10V corresponds to +/- 50 rpm	

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

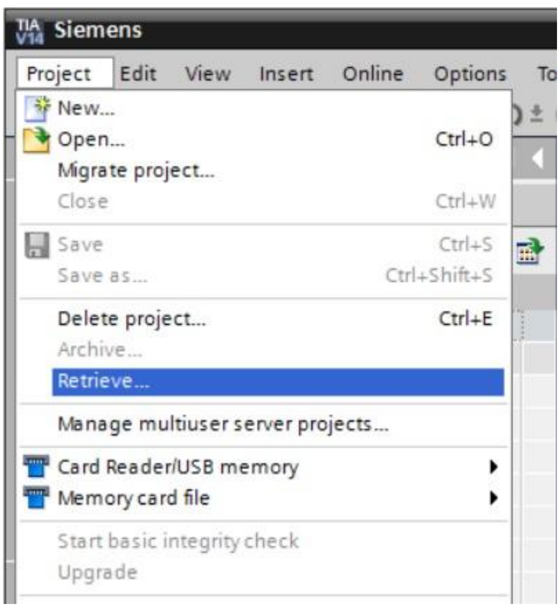
11

7 Structured step-by-step instructions

You can find instructions on how to carry out planning below. If you already have a good understanding of everything, it will be sufficient to focus on the numbered steps. Otherwise, simply follow the detailed steps in the instructions.

7.1 Retrieve an existing project

→ Before we can expand the "SCE_EN_031-500_Analog_Values_S7-1200.zap14" project from chapter "SCE_EN_031-500_Analog_Values_S7-1200", we must retrieve this project from the archive. To retrieve an existing project that has been archived, you must select the relevant archive with → Project → Retrieve in the project view. Confirm your selection with Open. (→ Project → Retrieve → Select a .zap archive → Open)

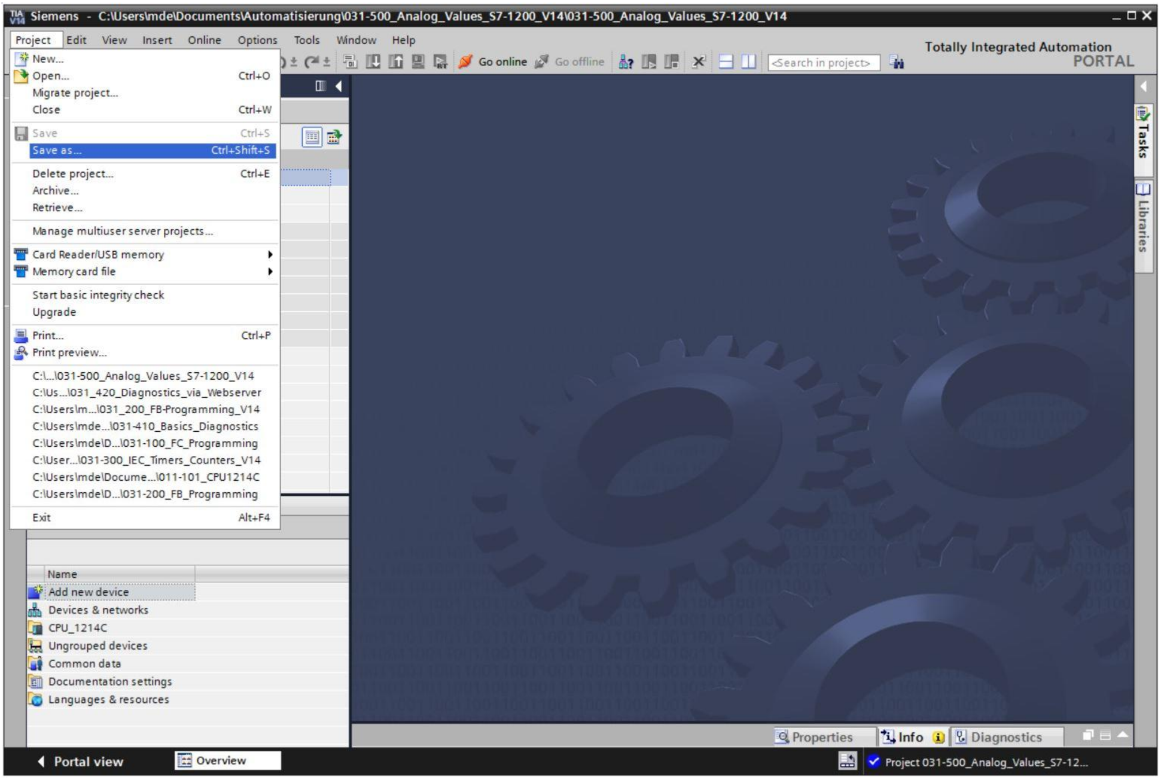


11

→ The next step is to select the target directory where the retrieved project will be stored. Confirm your selection with "OK".

(→ Target directory → OK)

→ Save the opened project under the name 031-600_Global_Data_Blocks_S7-1200. (→ Project → Save as … → 031-600_Global_Data_Blocks_S7-1200 → Save)

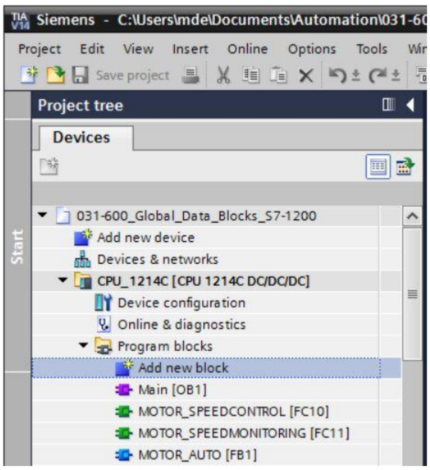


11

7.2 Create the global data block "SPEED_MOTOR"

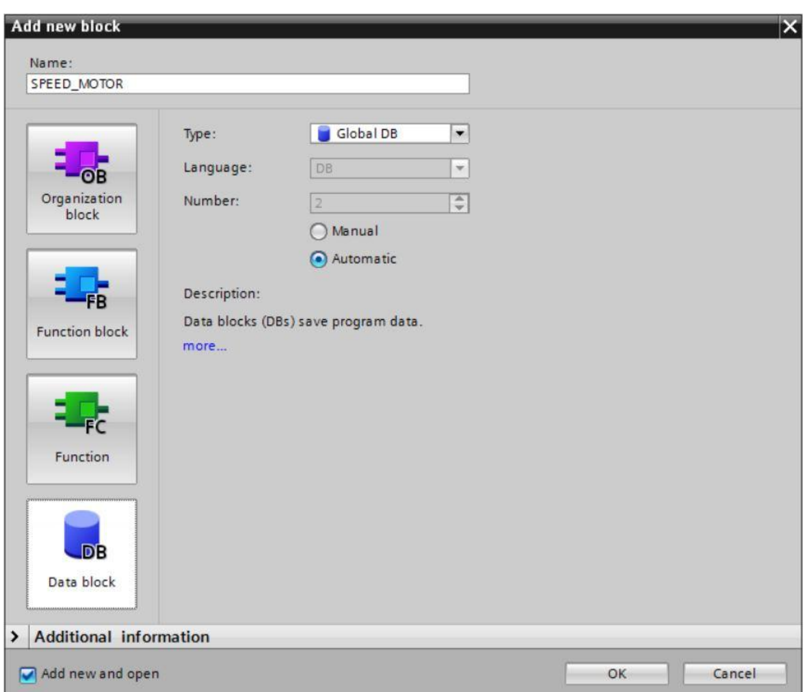
→ Select the 'Program blocks' folder of your CPU 1214C DC/DC/DC and then click "Add new block" to create a new global data block there.

(→ CPU_1214C [CPU 1214C DC/DC/DC] → Add new block)



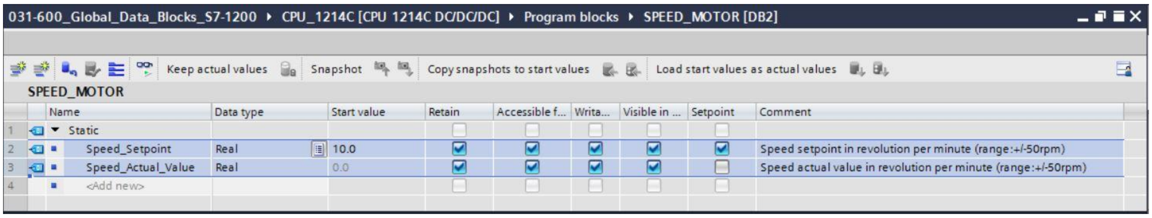
→ Select DB Data block in the next dialog and rename your new block to: "SPEED_MOTOR". Select 'Global DB' as the type. The number '2' will be automatically assigned. Select the "Add new and open" check box. Click "OK".

(→ DB Data block → Name: SPEED_MOTOR → Type: Global DB → ☑ Add new and open → OK)



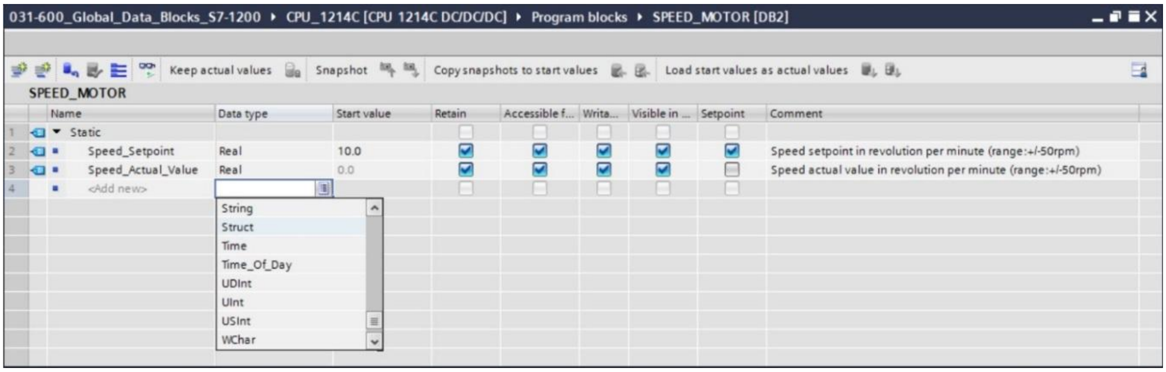
→ The "SPEED_MOTOR" data block is automatically displayed. Start by creating the 'Speed_Setpoint' and 'Speed_Actual_Value' tags shown here with their associated comments. Select 'Real' as the data type. Also set a start value of 10.0 rpm for the 'Speed_Setpoint'.

(→ Speed_Setpoint → Real → 10.0 → Speed_Actual_Value → Real)



Note: *Be sure to use the correct data types.*

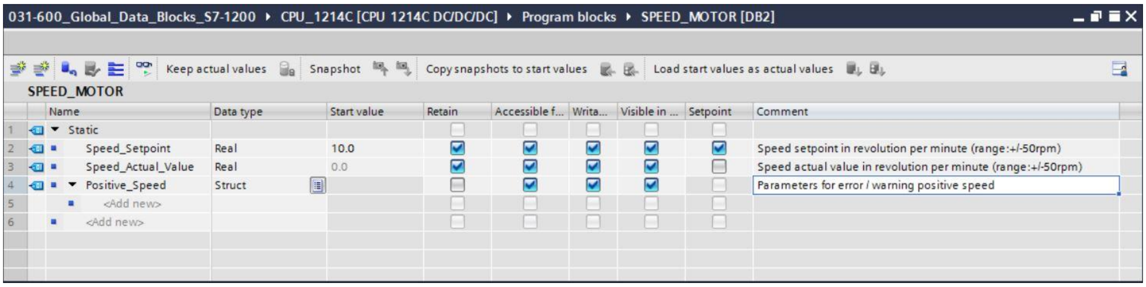
→ Next we create a tag structure 'Struct' so it can be duplicated later. (→ Struct)



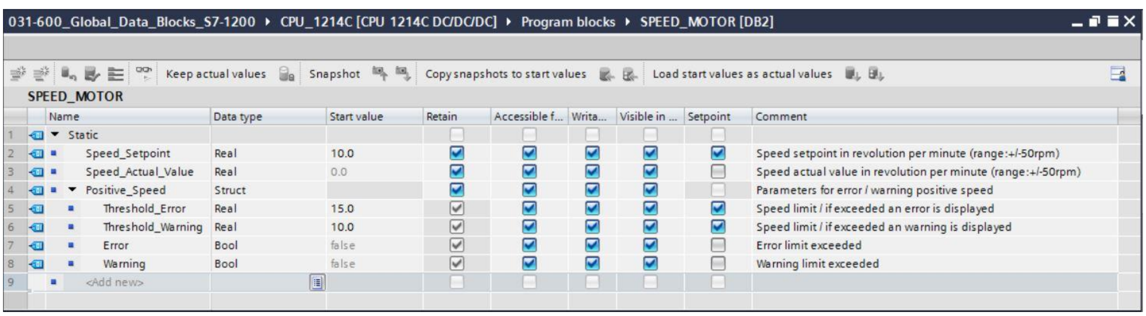
11

→ Name the structure 'Positive_Speed' and enter a comment.

(→ Positive_Speed)



→ Create the tags for the speed monitoring with the corresponding start values below the structure as shown here.

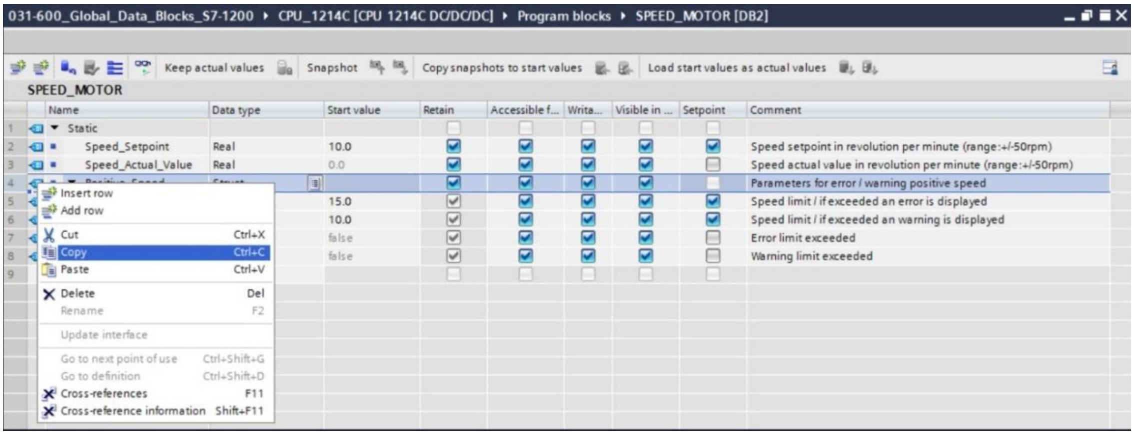


Note: *Be sure to use the correct data types.*

11

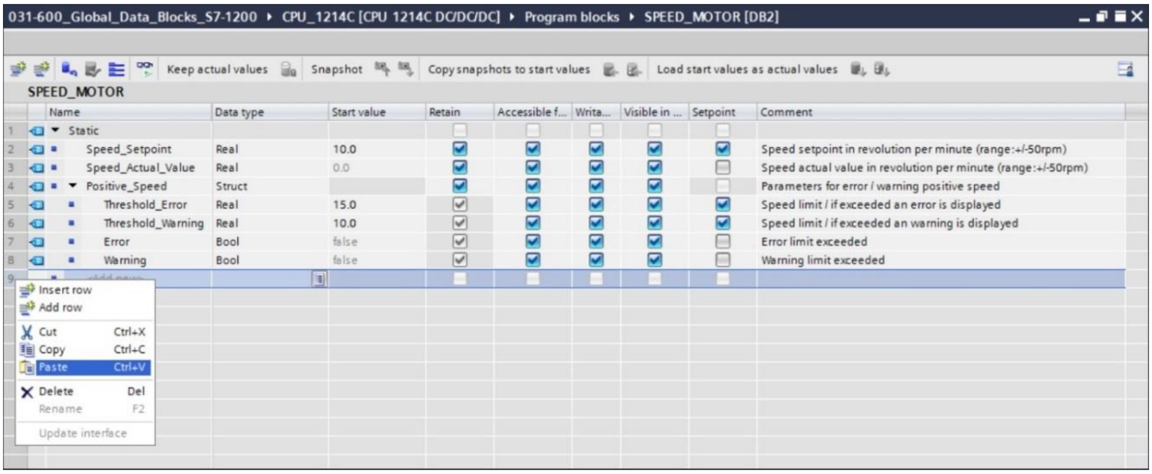
→ Then select the structure and copy it.

(→ Copy)



→ Paste the copied structure below the 'Positive_Speed' structure again.

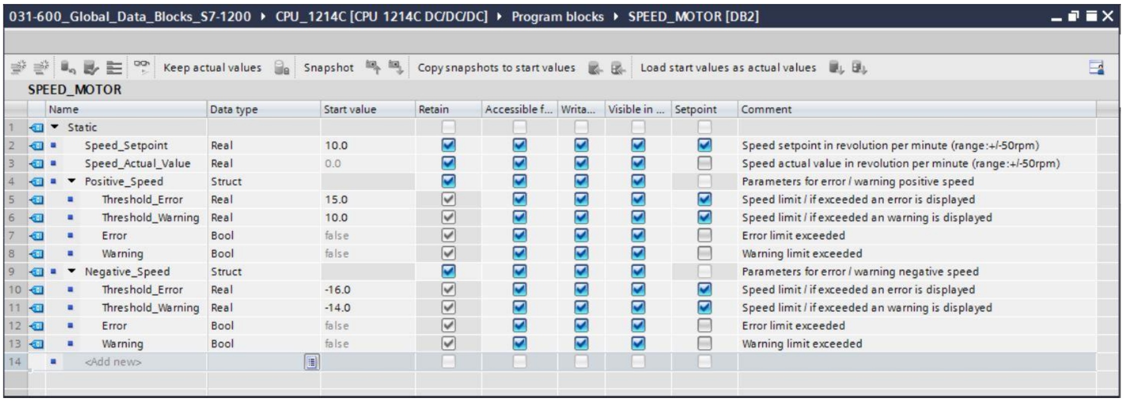
(→ Paste)



11

→ Rename the new structure to 'Negative_Speed' and enter a comment.

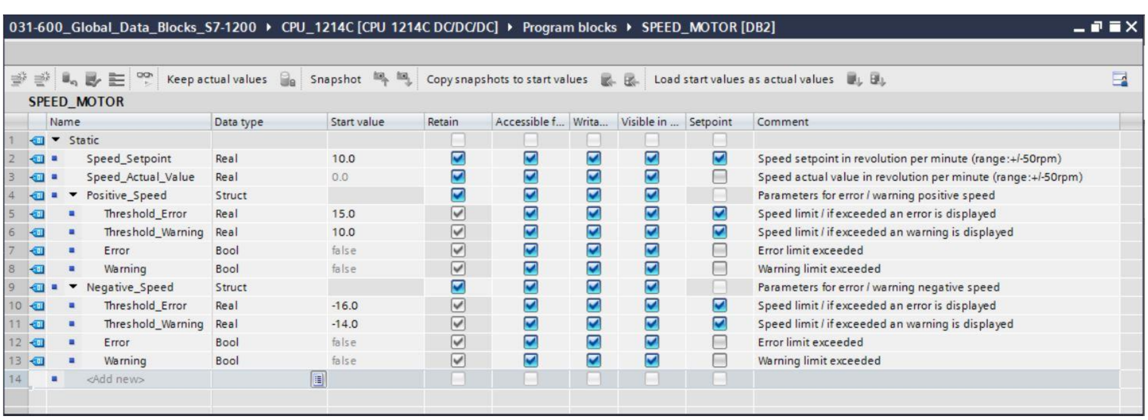
(→ Negative_Speed)



	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	▼ Static			☐	☐	☐	☐	☐	
2	Speed_Setpoint	Real	10.0	☑	☑	☑	☑	☑	Speed setpoint in revolution per minute (range:+/-50rpm)
3	Speed_Actual_Value	Real	0.0	☑	☑	☑	☑	☐	Speed actual value in revolution per minute (range:+/-50rpm)
4	▼ Positive_Speed	Struct		☑	☑	☑	☑	☐	Parameters for error / warning positive speed
5	Threshold_Error	Real	15.0	☑	☑	☑	☑	☑	Speed limit / if exceeded an error is displayed
6	Threshold_Warning	Real	10.0	☑	☑	☑	☑	☑	Speed limit / if exceeded an warning is displayed
7	Error	Bool	false	☑	☑	☑	☑	☐	Error limit exceeded
8	Warning	Bool	false	☑	☑	☑	☑	☐	Warning limit exceeded
9	▼ Negative_Speed	Struct		☑	☑	☑	☑	☐	Parameters for error / warning negative speed
10	Threshold_Error	Real	-16.0	☑	☑	☑	☑	☑	Speed limit / if exceeded an error is displayed
11	Threshold_Warning	Real	-14.0	☑	☑	☑	☑	☑	Speed limit / if exceeded an warning is displayed
12	Error	Bool	false	☑	☑	☑	☑	☐	Error limit exceeded
13	Warning	Bool	false	☑	☑	☑	☑	☐	Warning limit exceeded
14	<Add new>								

→ Do not forget to click Save project. The finished global data block "SPEED_MOTOR" [DB2] is shown below. Check to verify that Retain is selected ☑ and the corresponding start value is entered for all tags. The data will thus be retained in the data block even after a power failure or a STOP/START of the CPU. The check boxes for 'Accessible from HMI' ☑ and 'Visible in HMI' ☑ should also all have a check mark so that all tags in future expansions of this project will be accessible by the visualization systems (HMI). We will select the 'Setpoint' check box ☑ only for the default values in our data block.

(→ ☑ ☑ ☑ ☑)

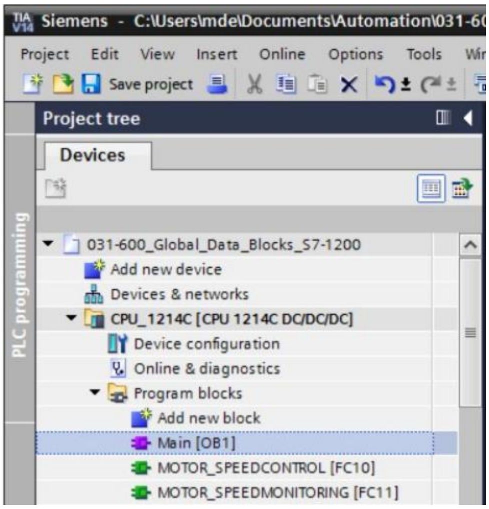


	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	▼ Static			☐	☐	☐	☐	☐	
2	Speed_Setpoint	Real	10.0	☑	☑	☑	☑	☑	Speed setpoint in revolution per minute (range:+/-50rpm)
3	Speed_Actual_Value	Real	0.0	☑	☑	☑	☑	☐	Speed actual value in revolution per minute (range:+/-50rpm)
4	▼ Positive_Speed	Struct		☑	☑	☑	☑	☐	Parameters for error / warning positive speed
5	Threshold_Error	Real	15.0	☑	☑	☑	☑	☑	Speed limit / if exceeded an error is displayed
6	Threshold_Warning	Real	10.0	☑	☑	☑	☑	☑	Speed limit / if exceeded an warning is displayed
7	Error	Bool	false	☑	☑	☑	☑	☐	Error limit exceeded
8	Warning	Bool	false	☑	☑	☑	☑	☐	Warning limit exceeded
9	▼ Negative_Speed	Struct		☑	☑	☑	☑	☐	Parameters for error / warning negative speed
10	Threshold_Error	Real	-16.0	☑	☑	☑	☑	☑	Speed limit / if exceeded an error is displayed
11	Threshold_Warning	Real	-14.0	☑	☑	☑	☑	☑	Speed limit / if exceeded an warning is displayed
12	Error	Bool	false	☑	☑	☑	☑	☐	Error limit exceeded
13	Warning	Bool	false	☑	☑	☑	☑	☐	Warning limit exceeded
14	<Add new>								

Note: *The use of setpoints is described further below in the step-by-step instructions.*

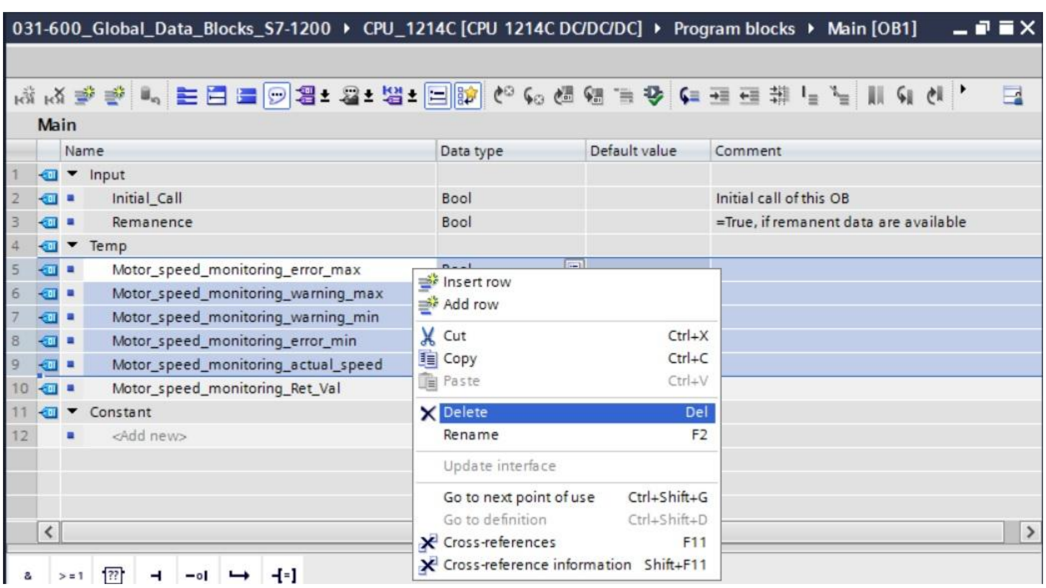
7.3 Access to data of the data block in the organization block

→ Open the “Main” [OB1] organization block with a double-click.

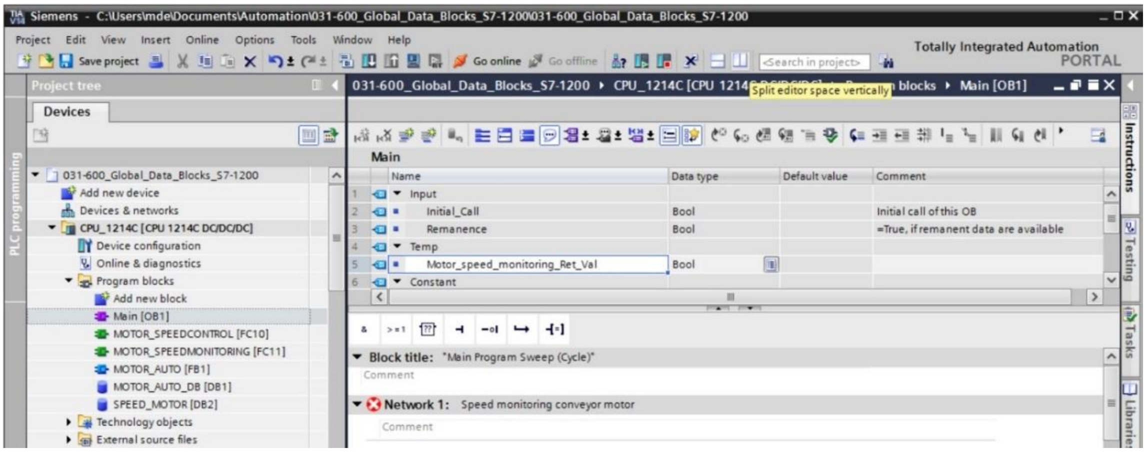


→ Delete the temporary tags in “Main” [OB1] that are no longer needed. Only the Boolean tag 'Motor_Speed_Control_Ret_Val' is still needed.

(→ Delete)

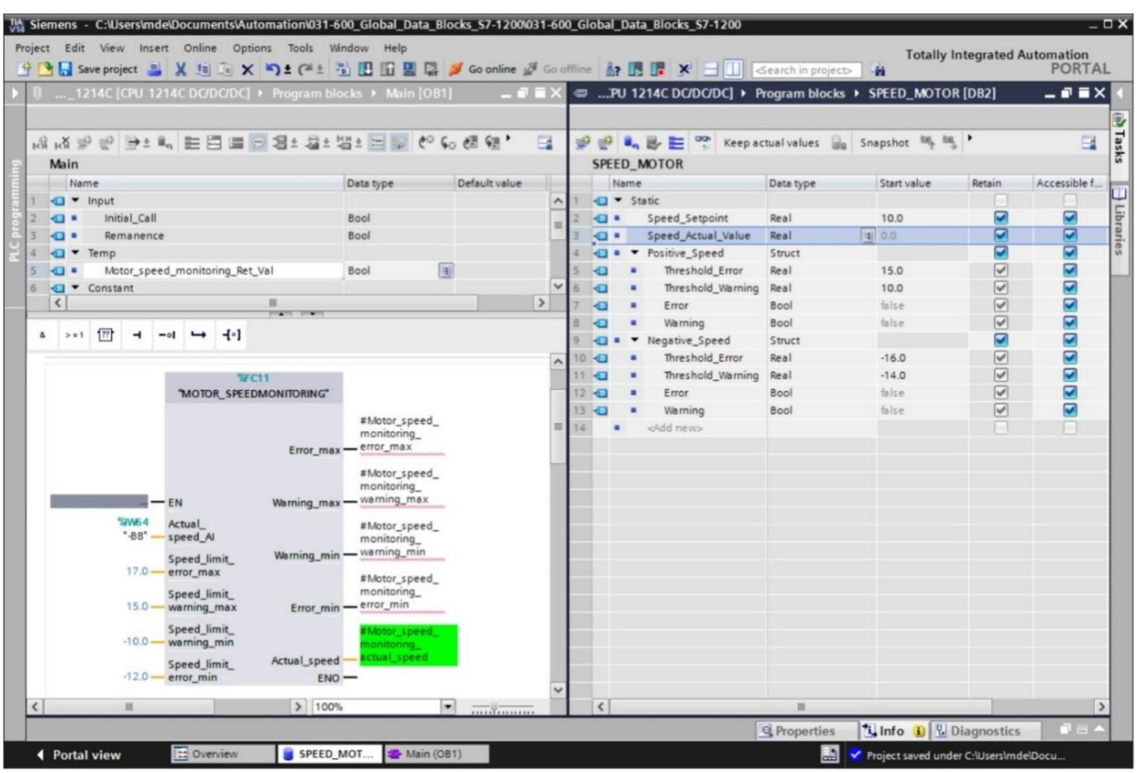


→ Have the "SPEED_MOTOR" [DB2] data block and the "Main" [OB1] organization block displayed side by side by clicking the '' icon to vertically split the editor area. (→)

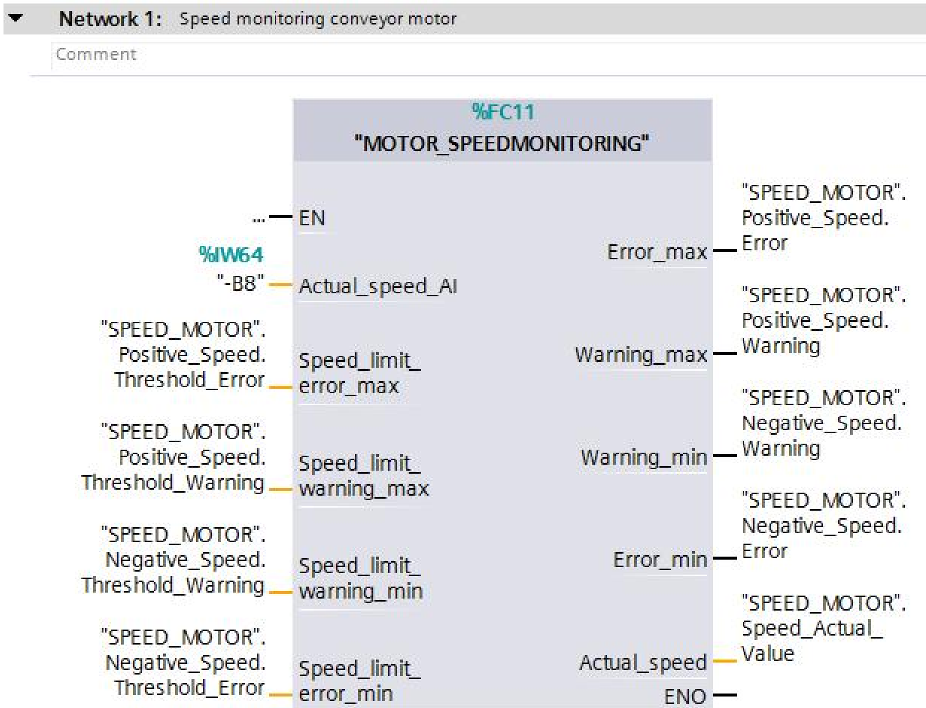


→ Use drag & drop to move the tags needed for the interconnection from the "SPEED_MOTOR" [DB2] data block onto the connections of the called functions and function blocks in the "Main" [OB1] organization block. First we move the 'Speed_Actual_Value' tag onto the 'Actual_speed' output of the "MOTOR_SPEEDMONITORING" [FC11] block.

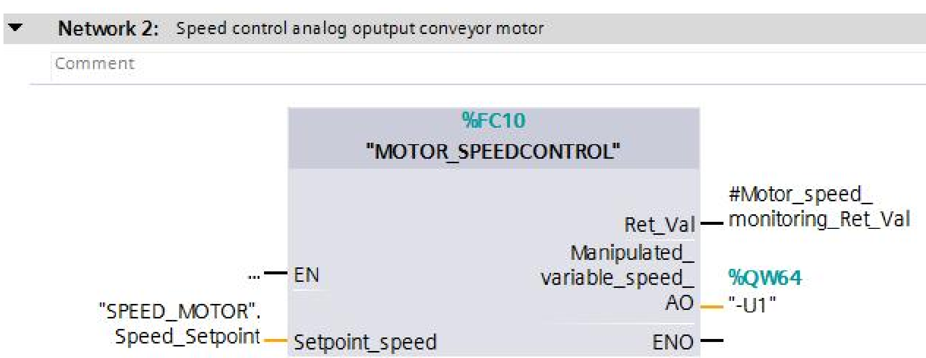
(→ Speed_Actual_Value)



→ Also connect the other contacts in Network 1 with tags from the "SPEED_MOTOR" [DB2] data block as shown here.

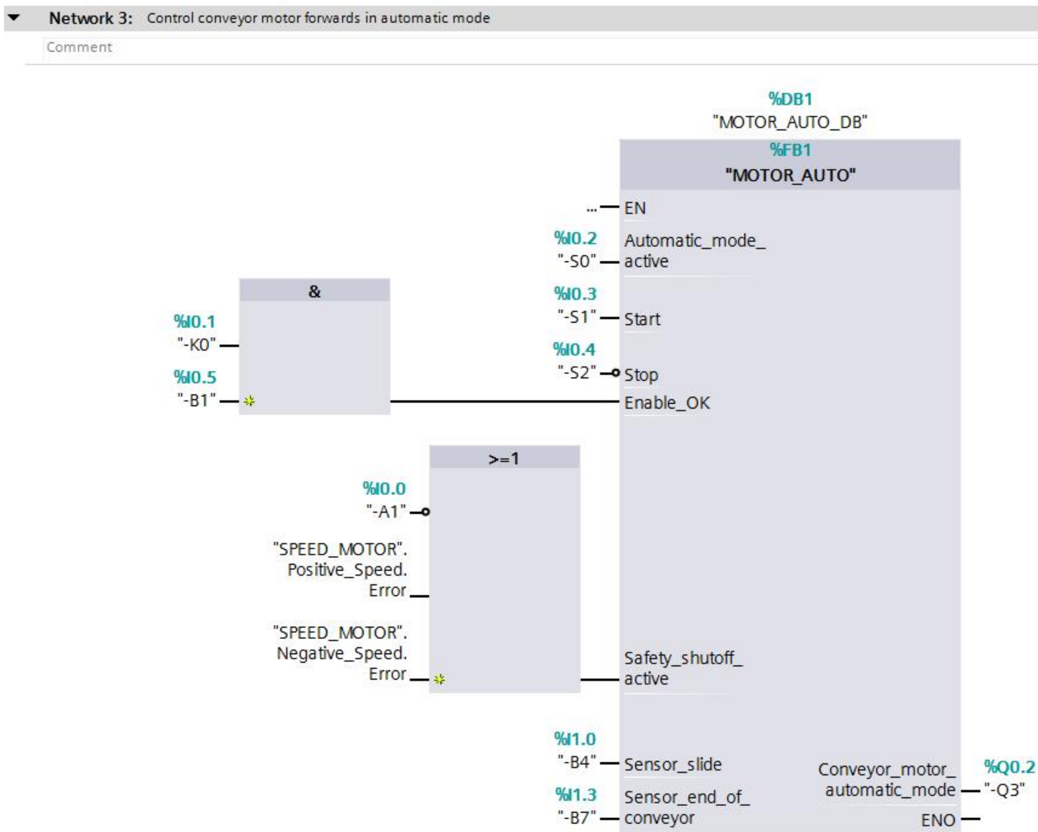


→ Connect the contacts in Network 2 with tags from the "SPEED_MOTOR" [DB2] data block as shown here.



11

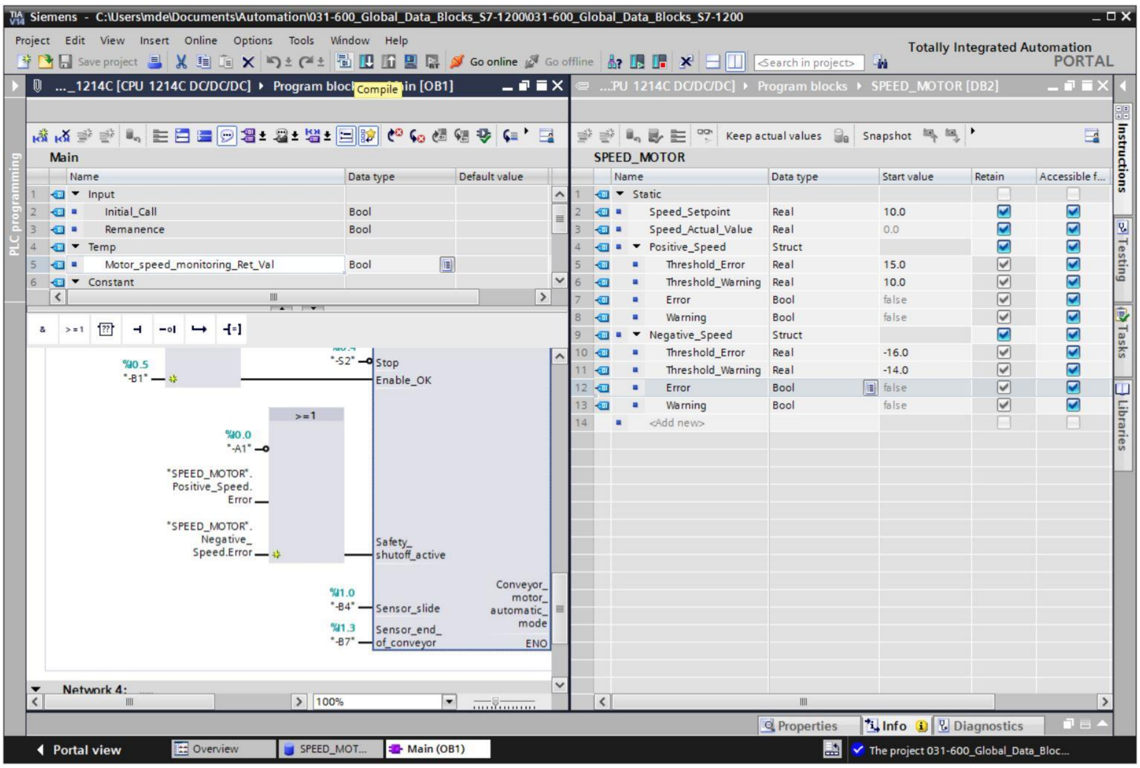
→ Connect the contacts in Network 3 with tags from the "SPEED_MOTOR" [DB2] data block as shown here.



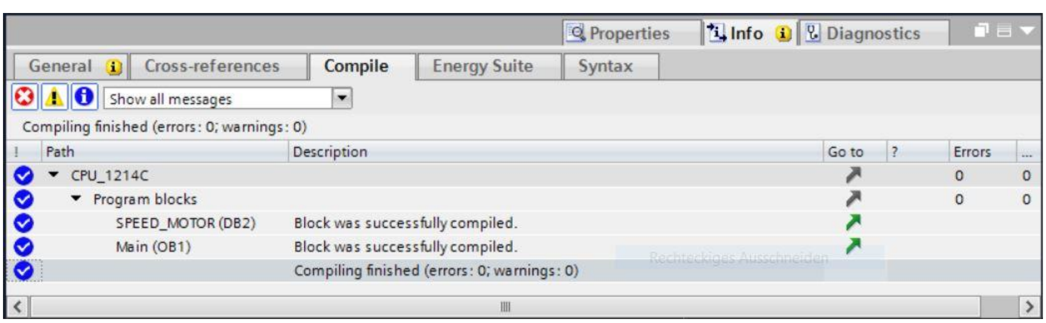
11

7.4 Save and compile the program

→ To save your project, click the Save project button in the menu. To compile all blocks, click the "Program blocks" folder and select the icon for compiling in the menu. (→ Save project → Program blocks →)

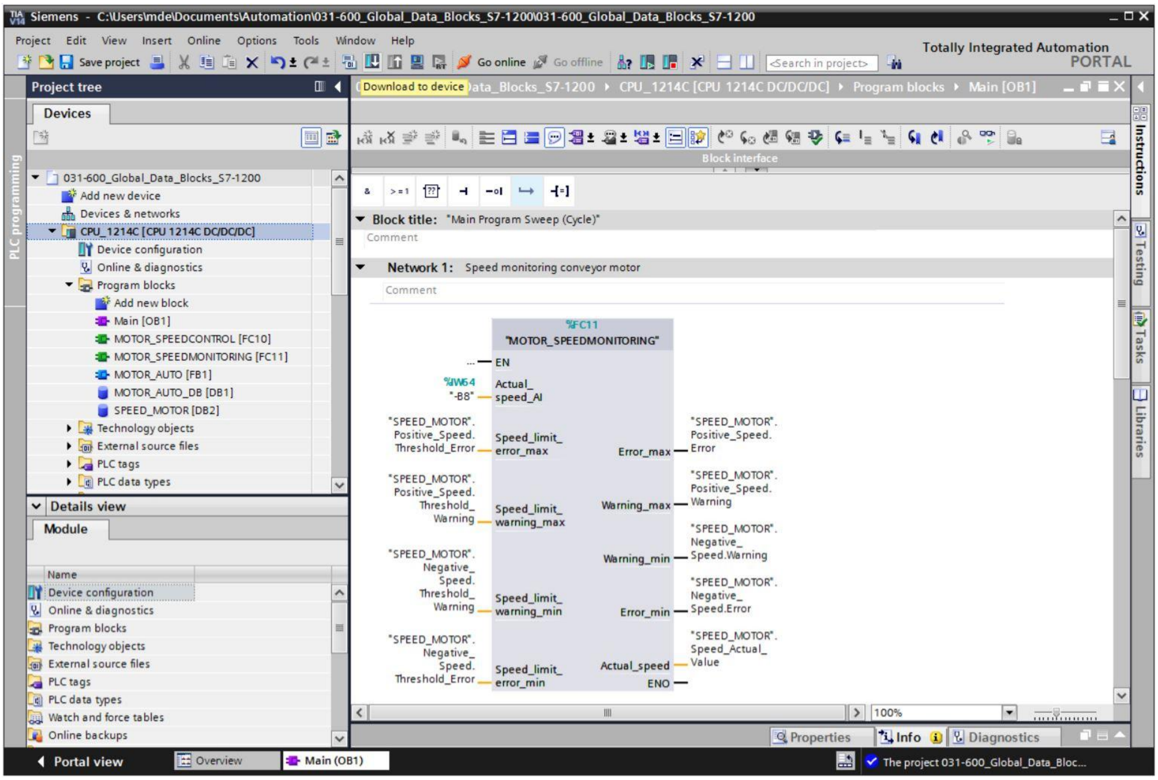


→ The "Info", "Compile" area shows which blocks were successfully compiled.



7.5 Download the program

→ After successful compilation, the complete controller with the created program including the hardware configuration can, as described in the previous modules, be downloaded. (→)

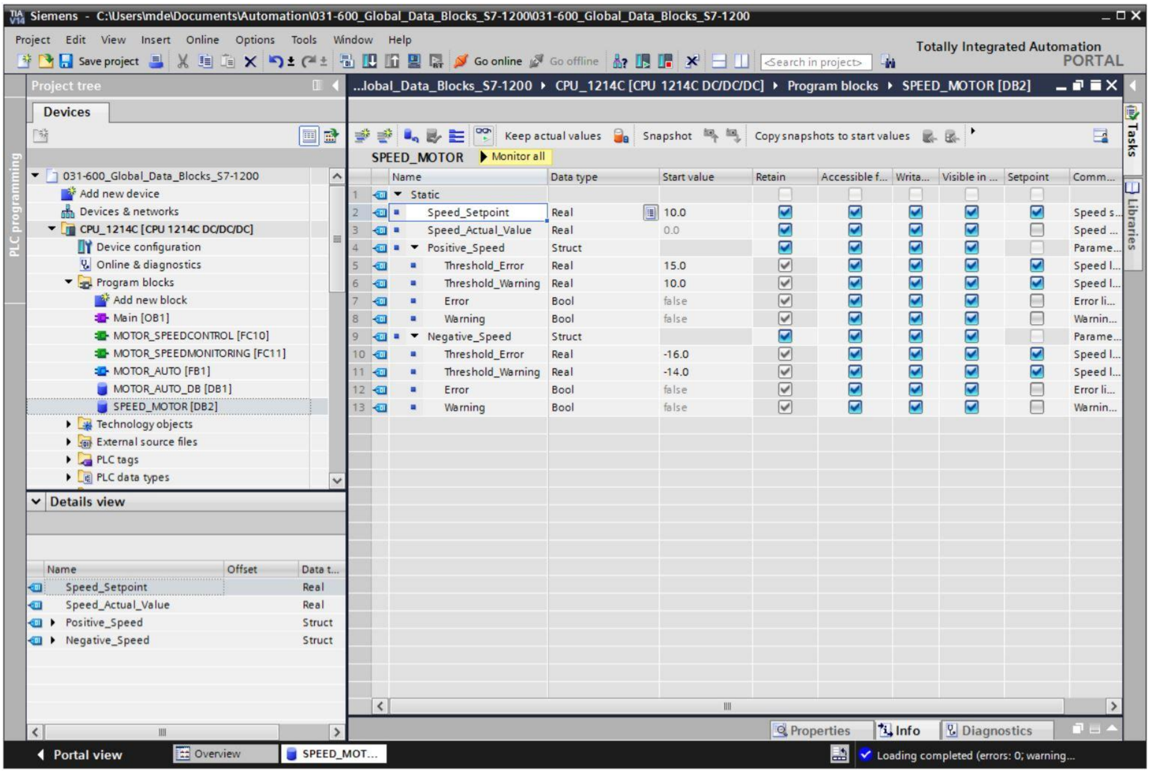


11

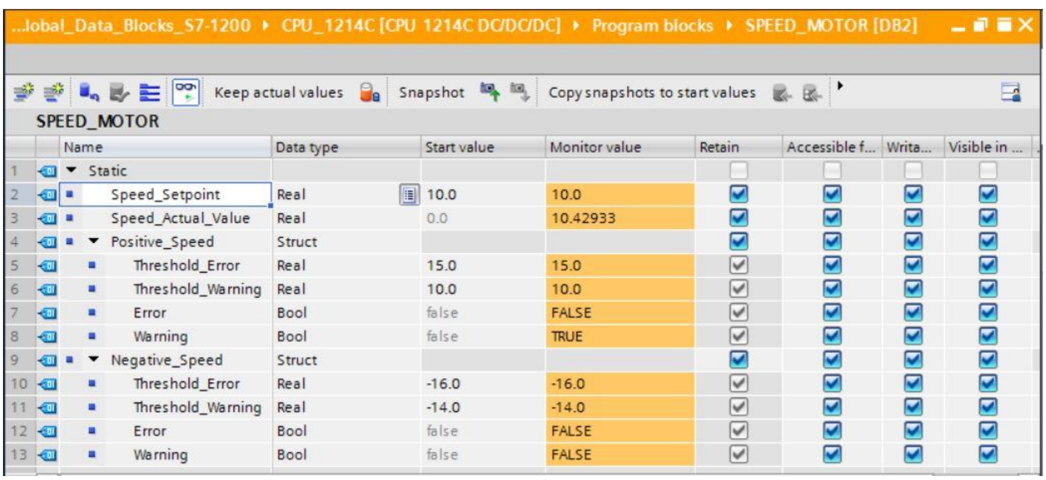
7.6 Monitor/modify values in data blocks

→ The desired block must be open for monitoring the tags of a downloaded data block. The monitoring can then be activated/deactivated by clicking the icon.

(→ SPEED_MOTOR [DB2] →)

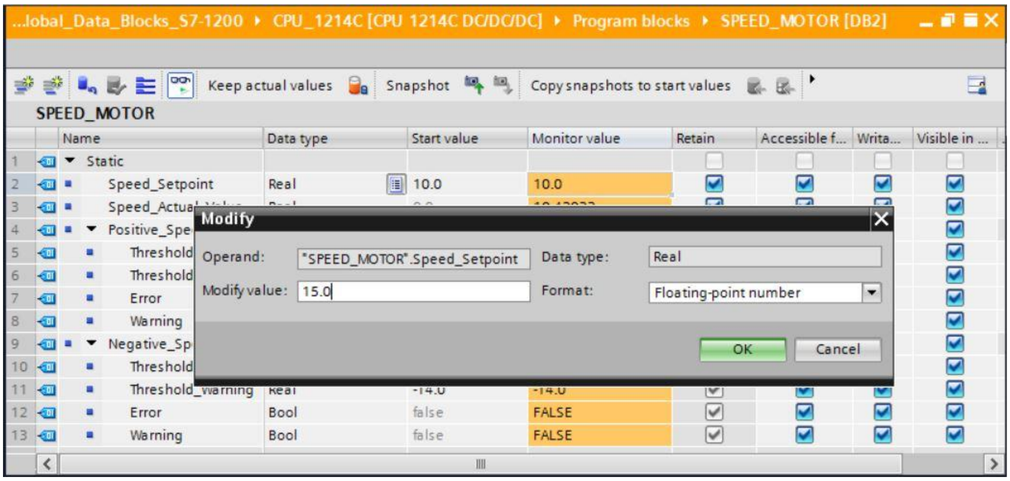


→ In the 'Monitor value' column, the values currently available in the CPU can be monitored.



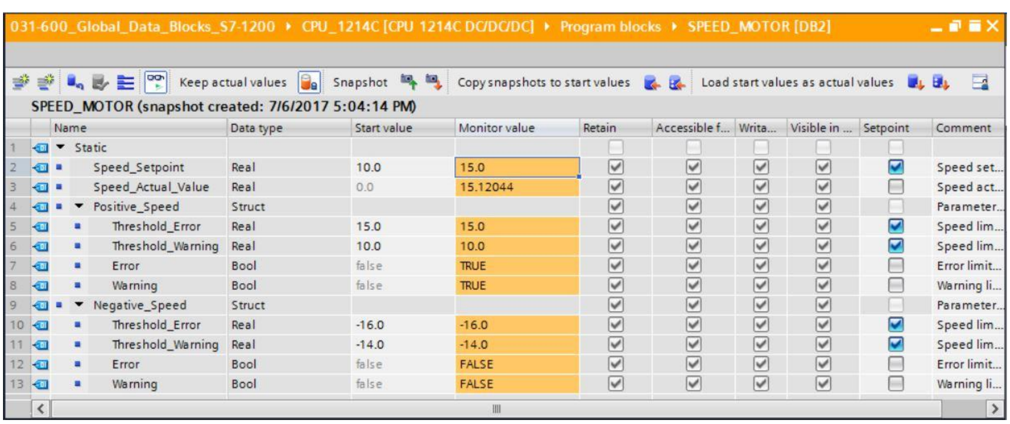
11

→ If you right-click on one of the values, the 'Modify' dialog for modifying this value opens (→ Modify → Modify value: 15.0 → OK)



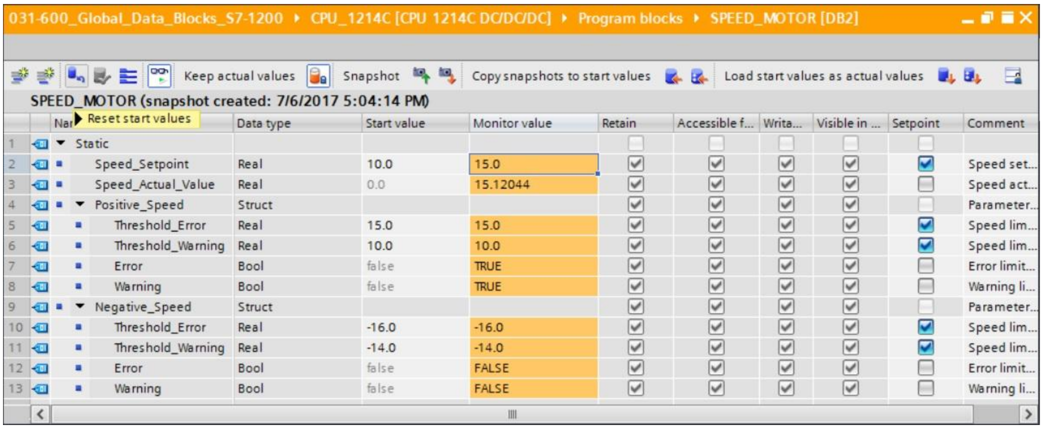
7.7 Initialize setpoints / reset start values

→ The setpoints can be initialized by clicking the '' icon. For the tags whose 'Setpoint' check box is selected, the start value will then be applied as the current value. (→)

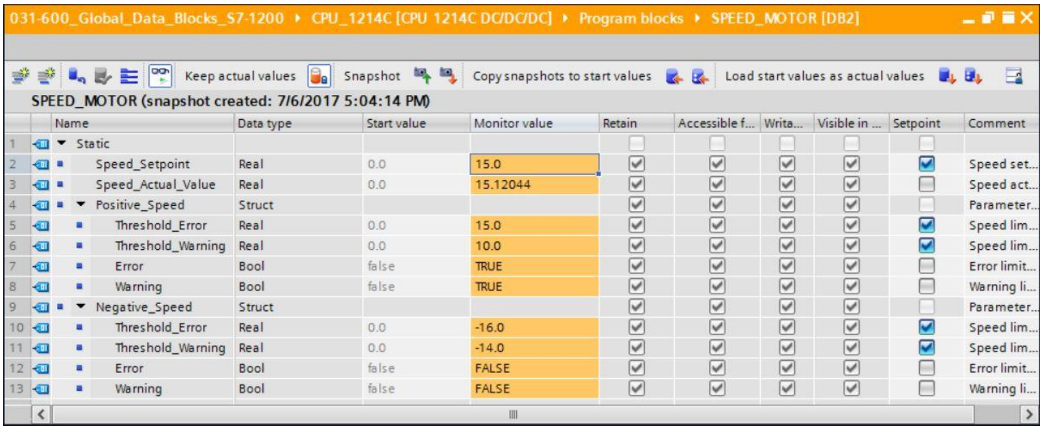


→ All start values can be reset by clicking the icon.

(→)



	Name	Data type	Start value	Monitor value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	▼ Static									
2	Speed_Setpoint	Real	10.0	15.0	✓	✓	✓	✓	✓	Speed set...
3	Speed_Actual_Value	Real	0.0	15.12044	✓	✓	✓	✓		Speed act...
4	▼ Positive_Speed	Struct			✓	✓	✓	✓		Parameter...
5	Threshold_Error	Real	15.0	15.0	✓	✓	✓	✓	✓	Speed lim...
6	Threshold_Warning	Real	10.0	10.0	✓	✓	✓	✓	✓	Speed lim...
7	Error	Bool	false	TRUE	✓	✓	✓	✓		Error limit...
8	Warning	Bool	false	TRUE	✓	✓	✓	✓		Warning li...
9	▼ Negative_Speed	Struct			✓	✓	✓	✓		Parameter...
10	Threshold_Error	Real	-16.0	-16.0	✓	✓	✓	✓	✓	Speed lim...
11	Threshold_Warning	Real	-14.0	-14.0	✓	✓	✓	✓	✓	Speed lim...
12	Error	Bool	false	FALSE	✓	✓	✓	✓		Error limit...
13	Warning	Bool	false	FALSE	✓	✓	✓	✓		Warning li...

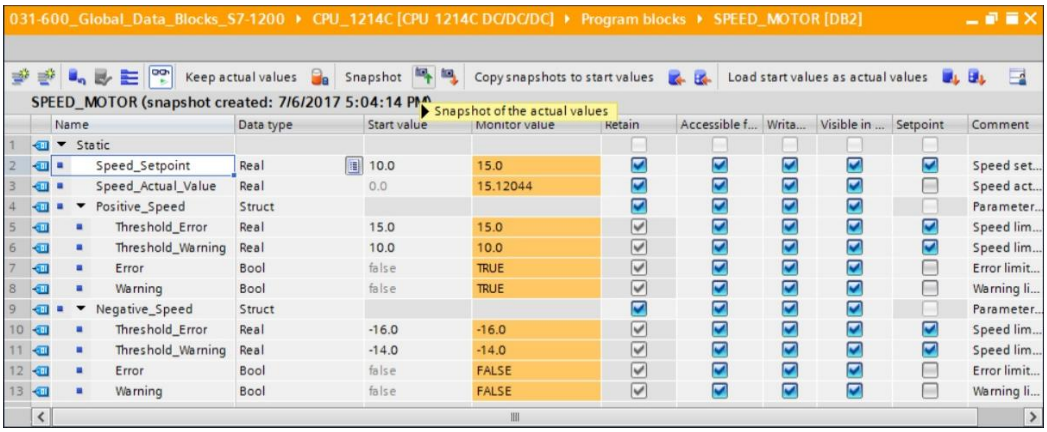


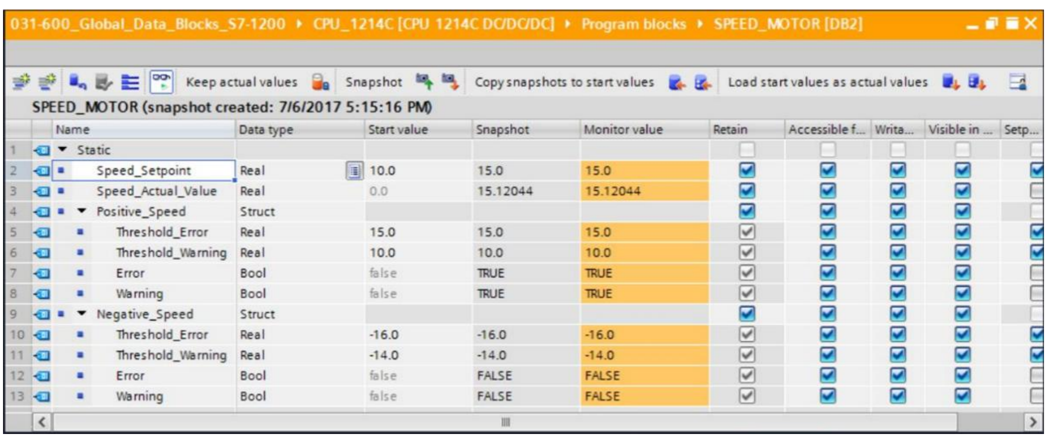
	Name	Data type	Start value	Monitor value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	▼ Static									
2	Speed_Setpoint	Real	0.0	15.0	✓	✓	✓	✓	✓	Speed set...
3	Speed_Actual_Value	Real	0.0	15.12044	✓	✓	✓	✓		Speed act...
4	▼ Positive_Speed	Struct			✓	✓	✓	✓		Parameter...
5	Threshold_Error	Real	0.0	15.0	✓	✓	✓	✓	✓	Speed lim...
6	Threshold_Warning	Real	0.0	10.0	✓	✓	✓	✓	✓	Speed lim...
7	Error	Bool	false	TRUE	✓	✓	✓	✓		Error limit...
8	Warning	Bool	false	TRUE	✓	✓	✓	✓		Warning li...
9	▼ Negative_Speed	Struct			✓	✓	✓	✓		Parameter...
10	Threshold_Error	Real	0.0	-16.0	✓	✓	✓	✓	✓	Speed lim...
11	Threshold_Warning	Real	0.0	-14.0	✓	✓	✓	✓	✓	Speed lim...
12	Error	Bool	false	FALSE	✓	✓	✓	✓		Error limit...
13	Warning	Bool	false	FALSE	✓	✓	✓	✓		Warning li...

11

7.8 Snapshots in data blocks

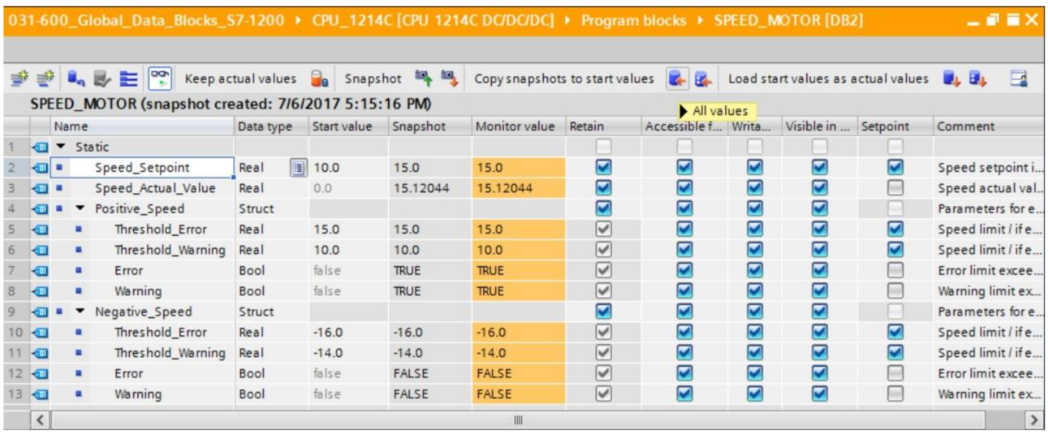
→ If you click the '' icon, a snapshot of the actual values can be taken in order to apply these values as start values or to transfer them back to the CPU later by clicking the icon '' (→ →).

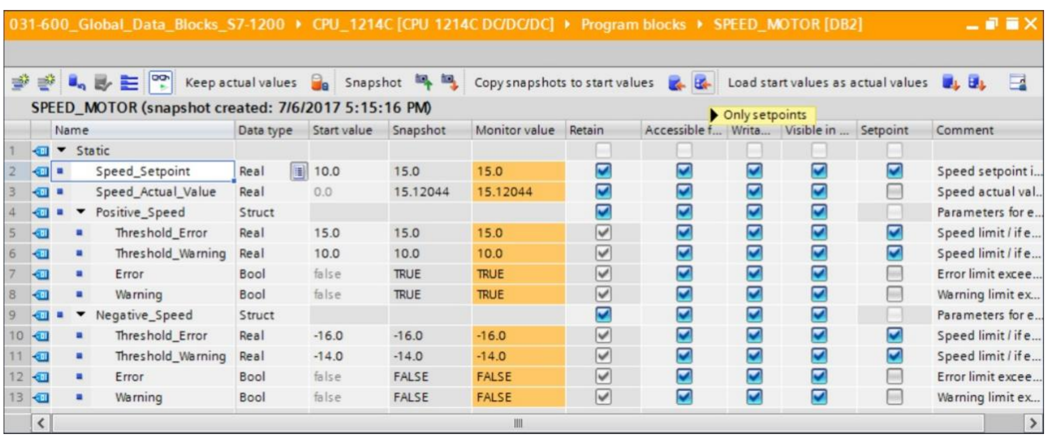




→ Alternatively, values from the snapshot can be copied to the start values by clicking the '' icon for all values or by clicking the '' icon for the setpoints only. Only the setpoints are needed here in most cases.

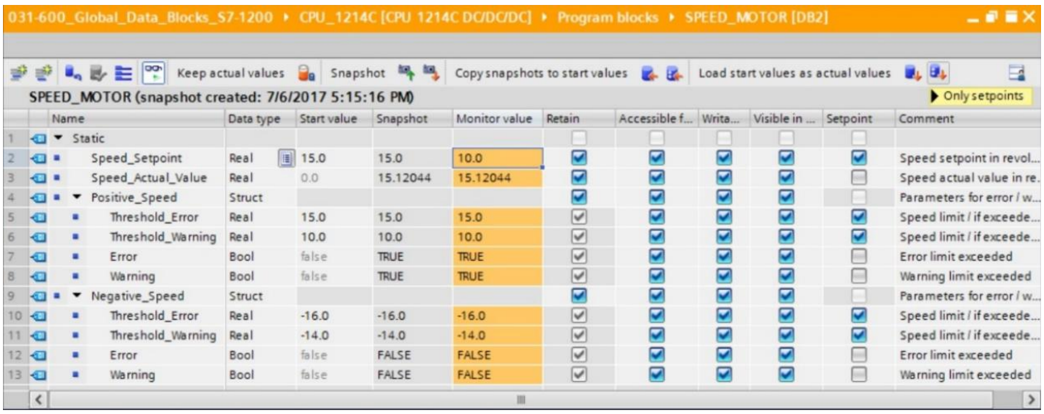
(→)

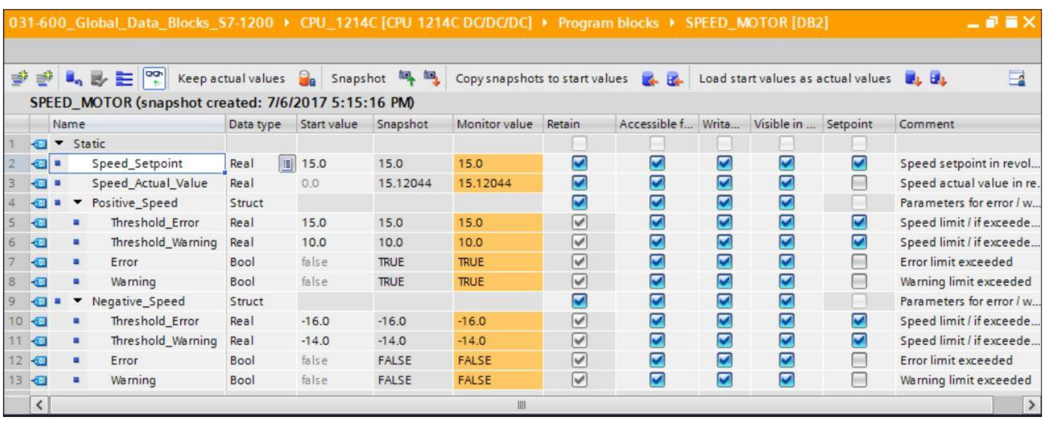




→ If you want to load the start values back into the actual values there are two possibilities. Alternatively all start values can be copied to the actual values by clicking the '' icon or only the setpoints by clicking the '' icon.

(→)

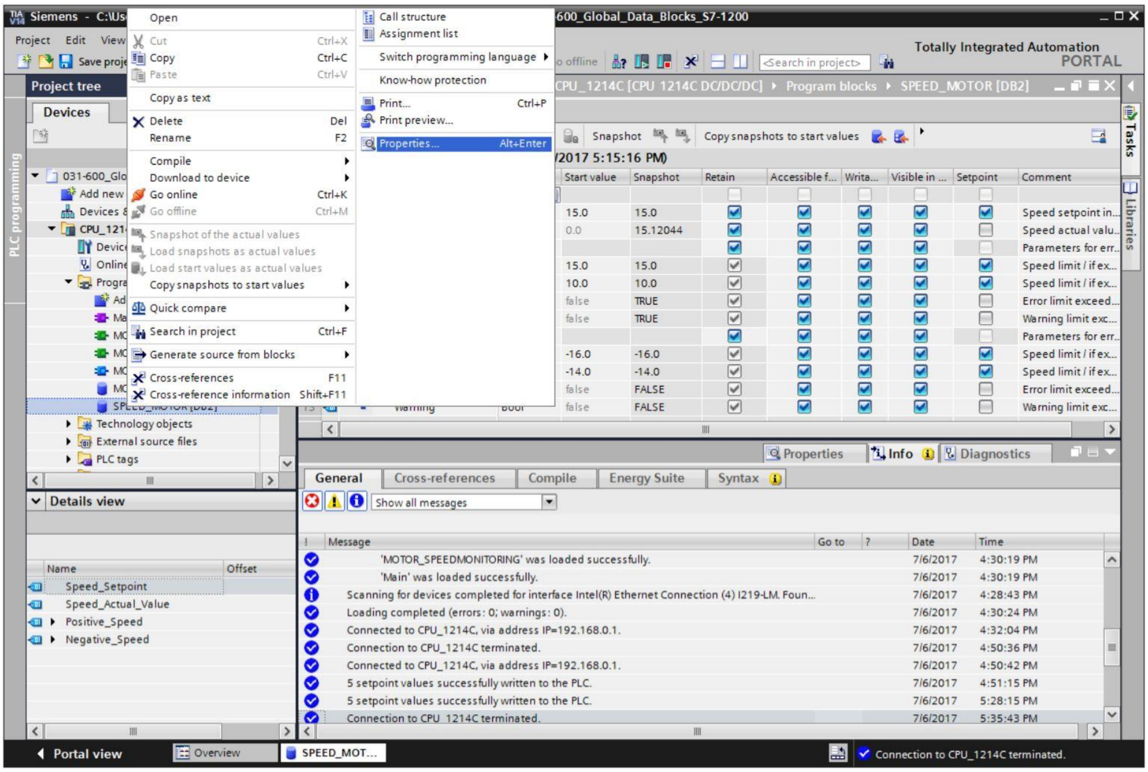




11

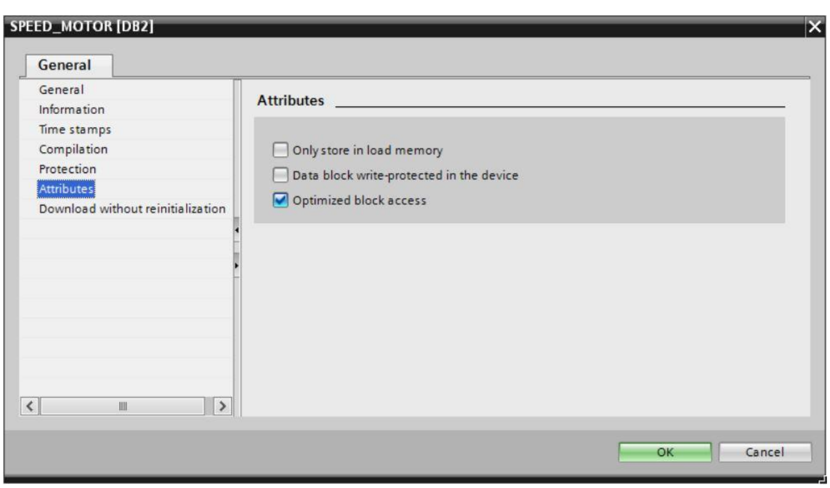
7.9 Expand data block and download it without reinitialization

→ To enable 'Download without reinitialization' for the "SPEED_MOTOR" [DB2] data block, you must go offline 'Go offline' and then open the properties of the data block. (→ Go offline → SPEED_MOTOR [DB2] → Properties)



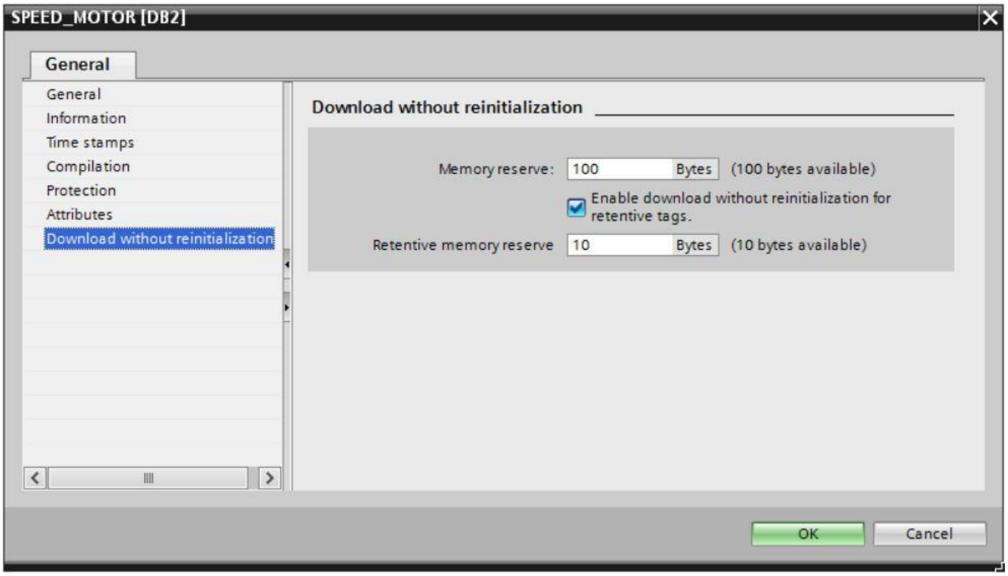
→ Select the 'Optimized block access' check box in the properties under 'General', 'Attributes'.

(→ General → Attributes → Optimized block access)



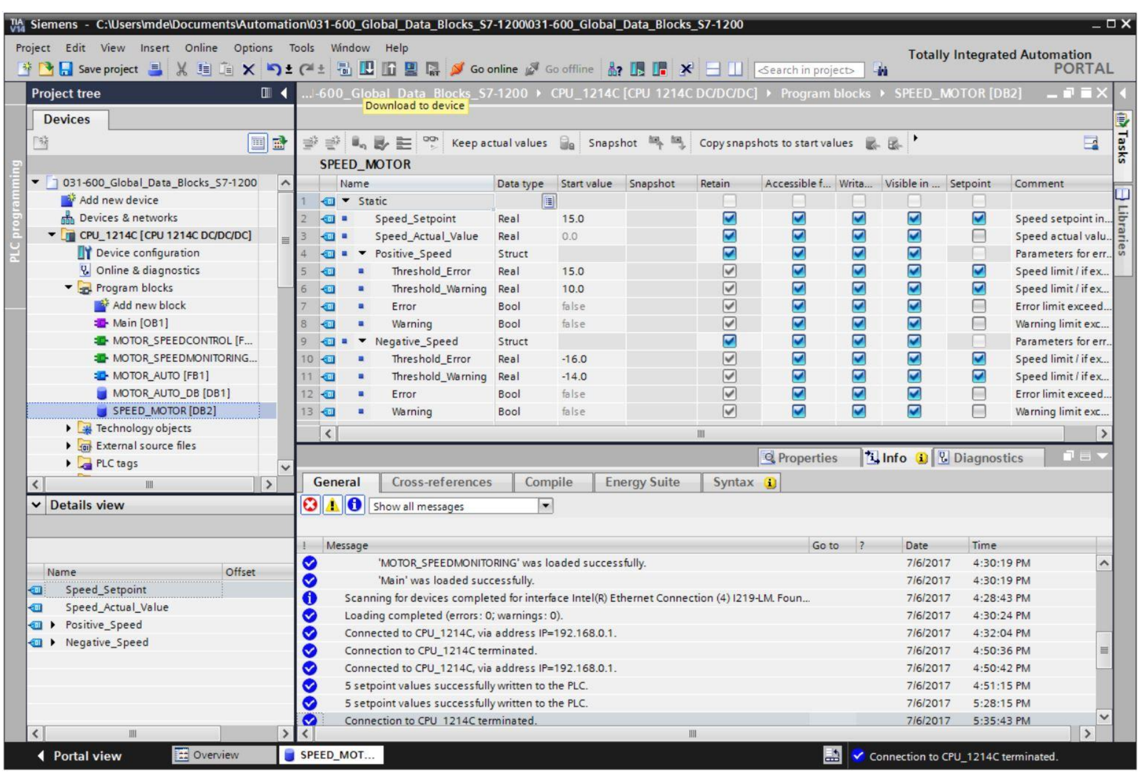
11

→ Assign a 'Retentive memory reserve' to the data block for 'Download without reinitialization'.
(→Download without reinitialization → Retentive memory reserve → 10 bytes → OK)

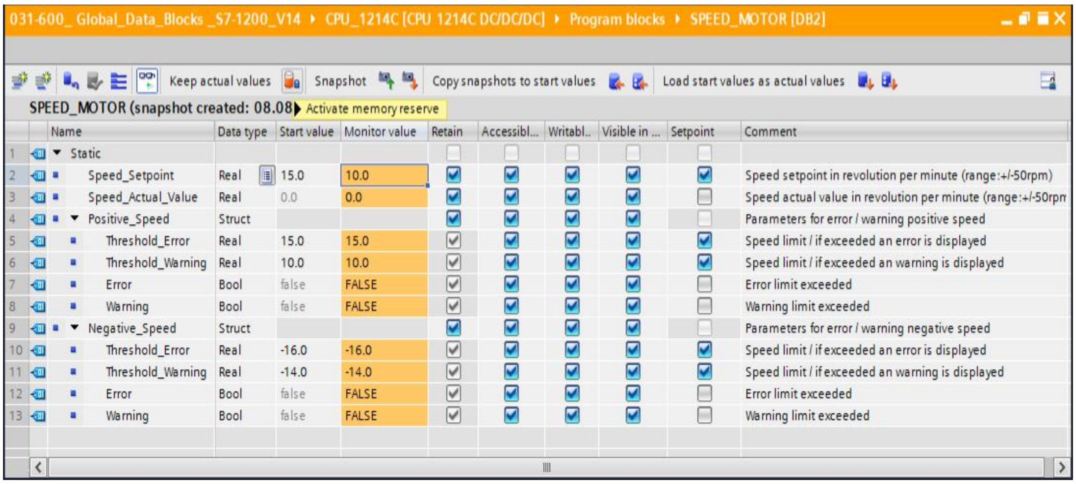


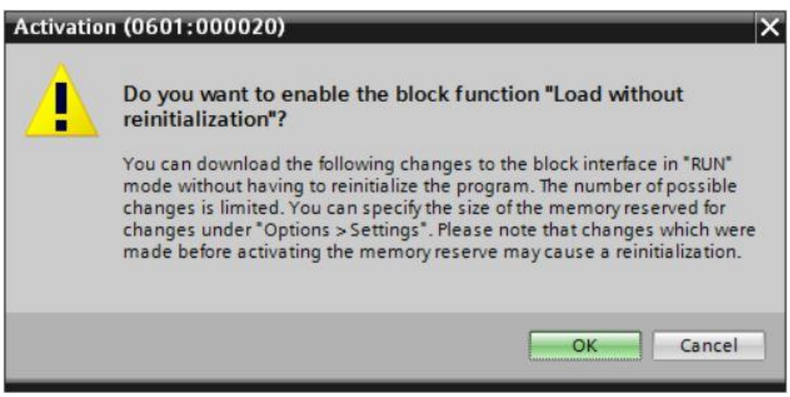
→ Download your "SPEED_MOTOR" [DB] data block to the controller again and select 'Go online'.

(→SPEED_MOTOR [DB] →  → Go online)



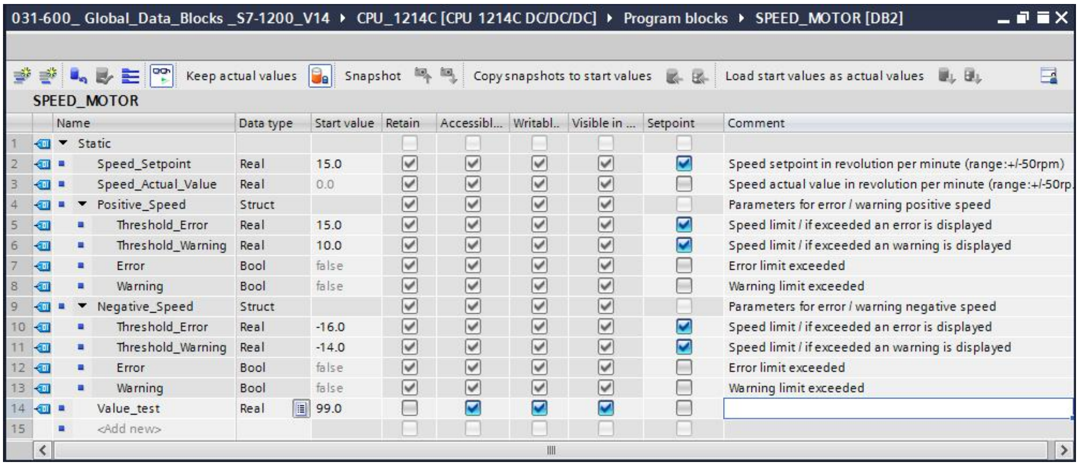
→ Then click the ' ' icon to activate memory reserve and thus activate downloading without reinitialization for keeping actual values. Confirm the safety prompt with 'OK'. (→ → OK)





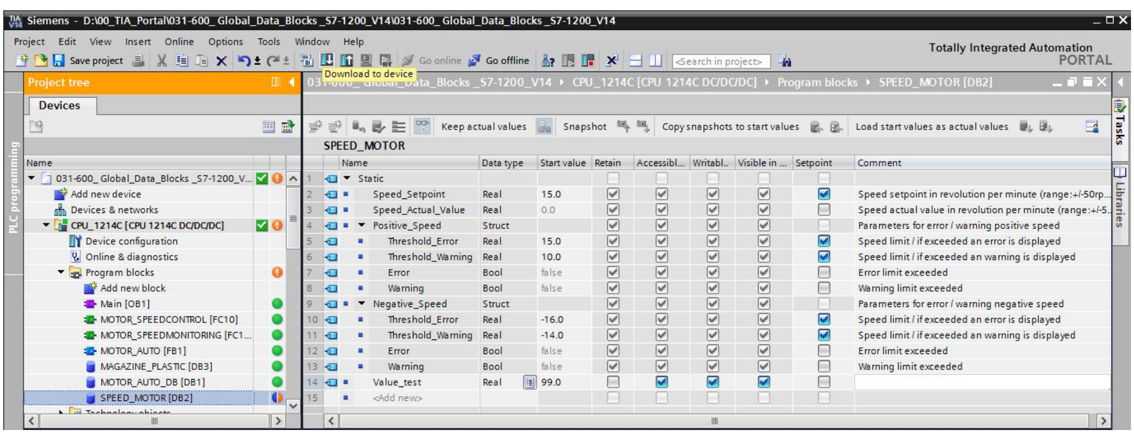
→ Next add any tag in your data block 99.0

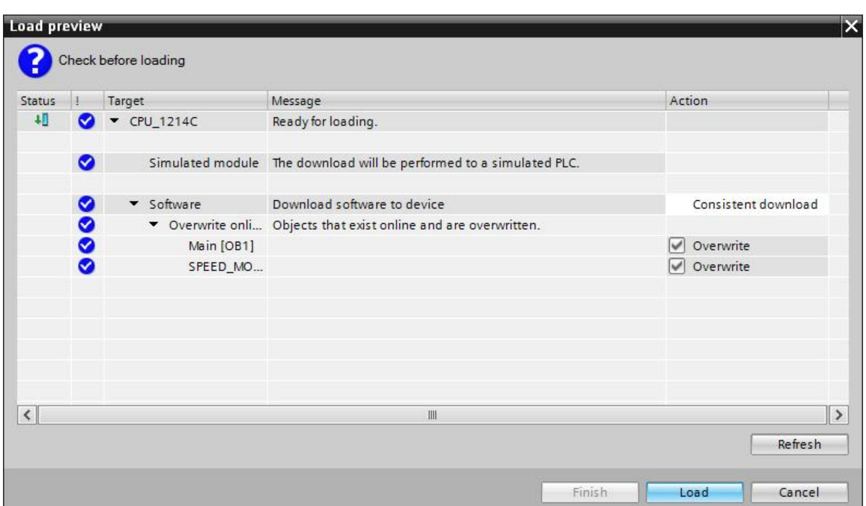
(→ Name: Value_test → Data type: Real → Start value: 99.0)



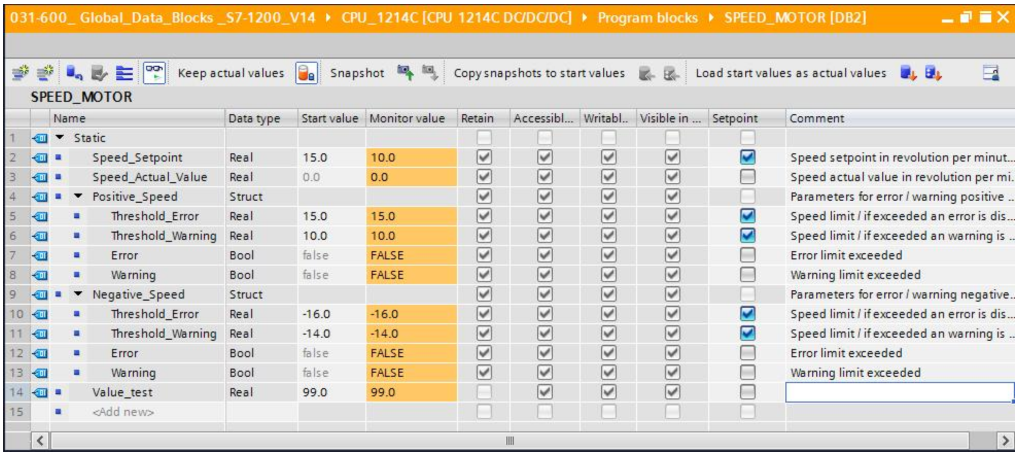
→ Download your "SPEED_MOTOR" [DB] data block to the controller again.

→SPEED_MOTOR [DB] →  → Download)





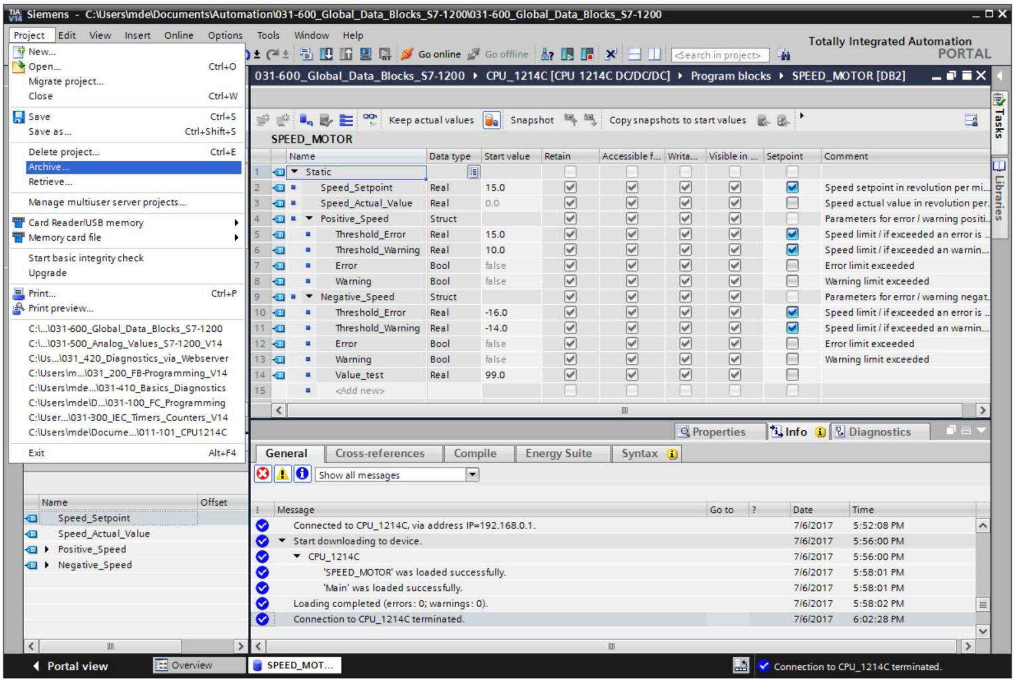
→ If you click ' ' to monitor the block again, you will see that the monitored values for the previously existing tags have not been overwritten with the start values. (→)



7.10 Archive the project

→ As the final step, we want to archive the complete project. Select the → 'Archive ...' command in the → 'Project' menu. Select a folder where you want to archive your project and save it with the file type "TIA Portal project archive".

(→ Project → Archive → TIA Portal project archive → 031-600_Global_Data_Blocks_S7-1200…. → Save)



8 Checklist

No.	Description	Completed
1	Data block SPEED_MOTOR [DB2] successfully created.	
2	Program changes made in Main [OB1].	
3	Compiling successful and without error message	
4	Download successful and without error message	
5	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated AUTOMATIC mode (-S0 = 1) Pushbutton automatic stop not actuated (-S2 = 1) Briefly press the automatic start pushbutton (-S1 = 1) Sensor part at slide activated (-B4 = 1) then Conveyor motor M1 variable speed (-Q3 = 1) switches on and stays on. The speed corresponds to the speed setpoint in the range +/- 50 rpm	
6	Sensor part at end of conveyor activated (-B7 = 1) → -Q3 = 0 (after 2 seconds)	
7	Briefly press the automatic stop pushbutton (-S2 = 0) → -Q3 = 0	
8	Activate EMERGENCY OFF (-A1 = 0) → -Q3 = 0	
9	Manual mode (-S0 = 0) → -Q3 = 0	
10	Switch off station (-K0 = 0) → -Q3 = 0	
11	Cylinder not retracted (-B1 = 0) → -Q3 = 0	
12	Speed > Motor_speed_monitoring_error_max → -Q3 = 0	
13	Speed < Motor_speed_monitoring_error_min → -Q3 = 0	
14	Project successfully archived	

9 Exercise

9.1 Task – Exercise

In this exercise a global data block "MAGAZINE_PLASTIC" [DB3] will be created additionally.

The setpoint and actual value of the counter for the plastic parts will be specified and displayed in this data block.

A connectable input for the setpoint setting and an output for displaying the actual value will also be added to the "MOTOR_AUTO" [FB1] function block.

9.2 Technology diagram

Here you see the technology diagram for the task.

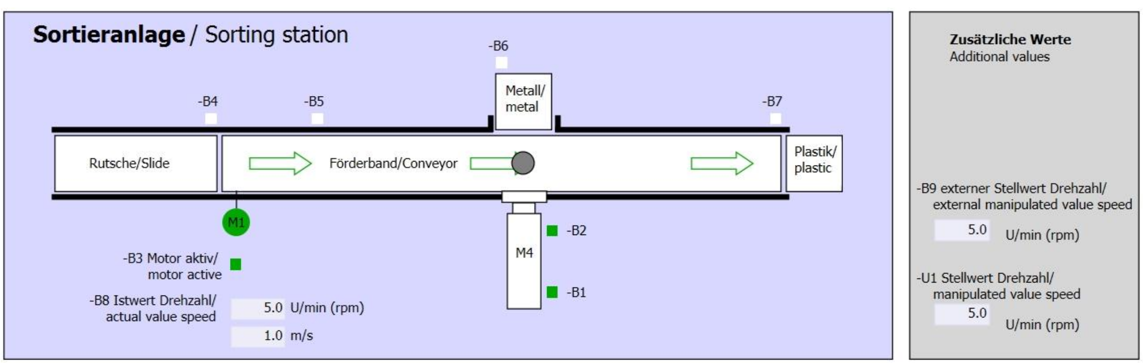


Figure 5: Technology diagram

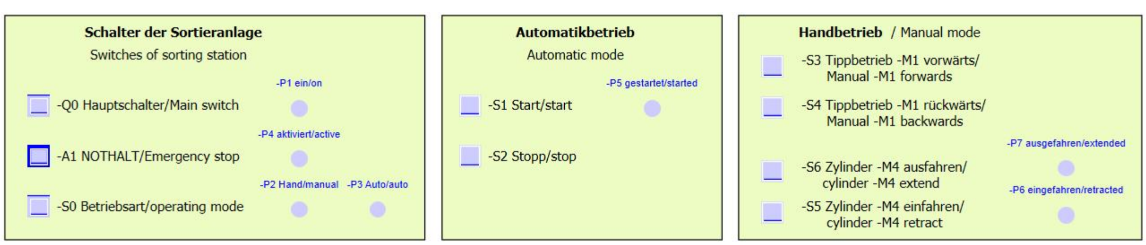


Figure 6: Control panel

11

9.3 Reference list

The following signals are required as global operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop OK	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	-S1	Pushbutton automatic start	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder -M4 retracted	NO
I 1.0	BOOL	-B4	Sensor part at slide	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO
IW64	BOOL	-B8	Sensor actual value speed of the motor +/-10V corresponds to +/- 50 rpm	

DO	Type	Identifier	Function	
Q 0.2	BOOL	-Q3	Conveyor motor -M1 variable speed	
QW 64	BOOL	-U1	Manipulated value speed of the motor in 2 directions +/- 10V corresponds to +/- 50 rpm	

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

11

9.4 Planning

Plan the implementation of the task on your own.

9.5 Checklist – Exercise

No.	Description	Completed
1	Data block MAGAZINE_PLASTIC [DB3] successfully created.	
2	Program changes made in MOTOR_AUTO [FB1].	
3	Program changes made in Main [OB1].	
4	Compiling successful and without error message	
5	Download successful and without error message	
6	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated AUTOMATIC mode (-S0 = 1) Pushbutton automatic stop not actuated (-S2 = 1) Briefly press the automatic start pushbutton (-S1 = 1) Sensor part at slide activated (-B4 = 1) then Conveyor motor M1 variable speed (-Q3 = 1) switches on and stays on. The speed corresponds to the speed setpoint in the range +/- 50 rpm	
7	Sensor part at end of conveyor activated (-B7 = 1) → -Q3 = 0 (after 2 seconds)	
8	Briefly press the automatic stop pushbutton (-S2 = 0) → -Q3 = 0	
9	Activate EMERGENCY OFF (-A1 = 0) → -Q3 = 0	
10	Manual mode (-S0 = 0) → -Q3 = 0	
11	Switch off station (-K0 = 0) → -Q3 = 0	
12	Cylinder not retracted (-B1 = 0) → -Q3 = 0	
13	Speed > Motor_speed_monitoring_error_max → -Q3 = 0	
14	Speed < Motor_speed_monitoring_error_min → -Q3 = 0	
15	Project successfully archived	

10 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

Notes

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

SIMATIC HMI Panels

- **1 SIMATIC HMI KTP700 BASIC COLOR PANEL for S7-1200**
  Order no.: 6AV2123-2GB03-0AA1
- **6x SIMATIC HMI KTP700 BASIC COLOR PANEL for S7-1200**
  Order no.: 6AV2123-2GB03-0AA0

SIMATIC controllers

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3

SIMATIC STEP 7 Software for Training

- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is available at: siemens.com/sce/tp

Continued training

For regional Siemens SCE Continued Training, get in touch with your regional SCE contact:
siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

This SCE Learn-/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learn-/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

12

Table of contents

12

12

Process Visualization with the SIMATIC HMI Panel KTP700 Basic and WinCC Basic

1 Objective

In this section, you will become familiar with the basics of process visualization and the use of a SIMATIC HMI Panel KTP700 Basic together with SIMATIC S7-1200 and the TIA Portal programming tool.

The module explains the configuring of a SIMATIC HMI Panel KTP700 Basic, the creation of a connection to the SIMATIC S7-1200 and the read and write access to CPU data of the SIMATIC HMI Panel KTP700 Basic.

The SIMATIC S7 controllers listed in section 3 SIMATIC S7 controllers can be used.

2 Requirements

This module builds on the Global Data Blocks for SIMATIC S7-1200 module. To implement this module, you can use the following project, for example:
"SCE_EN_031-600 Globale_Data_Blocks_S7-1200….zap14".

3 Hardware and software required

1 Engineering Station: Requirements for hardware and operating system
(for additional information, see Readme on the TIA Portal Installation DVD)

2 SIMATIC STEP 7 Basic software in the TIA Portal V14 SP1 or higher

3 WinCC Basic software in the TIA Portal V14 SP1 or higher

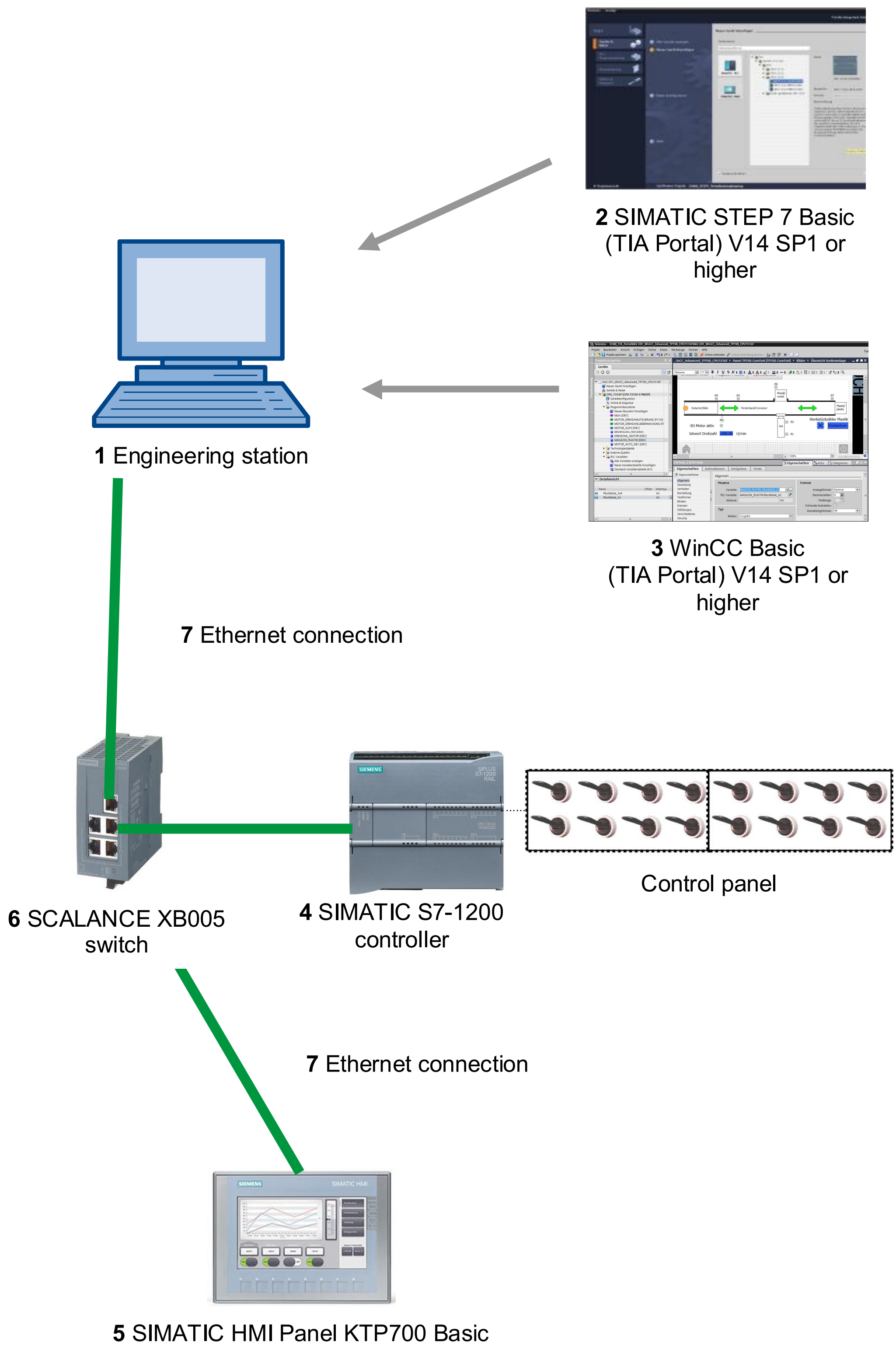
4 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with signal board ANALOG OUTPUT SB1232, 1 AO, firmware V4.2.1 or higher

Note: The digital inputs and the analog inputs and outputs should be fed out to a control panel.

5 SIMATIC HMI Panel KTP700 Basic

6 SCALANCE XB005 INDUSTRIAL ETHERNET Switch

7 Ethernet connection between the engineering station and switch, controller and switch and HMI Panel KTP700 Basic and switch


2 SIMATIC STEP 7 Basic (TIA Portal) V14 SP1 or higher
1 Engineering station
3 WinCC Basic (TIA Portal) V14 SP1 or higher
7 Ethernet connection
6 SCALANCE XB005 switch
4 SIMATIC S7-1200 controller
Control panel
7 Ethernet connection
5 SIMATIC HMI Panel KTP700 Basic

4 Theory

4.1 Process visualization

Due to production processes are becoming more and more complex and requirements for machine and plant functionality are increasing, operators need a powerful tool for controlling and monitoring production plants. An HMI system (human-machine interface) represents the interface between man (operator) and process (machine/plant). It is the controller that actually controls the process. Hence, there is an interface between the operator and WinCC (at the HMI device) and an interface between WinCC and the controller.

The SIMATIC HMI Basic Panels and WinCC perform the following tasks:

• Display processes with a straightforward screen structure

The process is mapped on the HMI device. The display on the HMI device is updated when a process state changes, for example. A process can be displayed in a clearly structured manner in multiple screens.

• Communicate with processes

The operator can communicate with the process via the graphical user interface For example, the operator can define a setpoint for the controller or start a motor.

• Output alarms

When critical process states occur, such as when a specified limit is exceeded, an alarm is automatically triggered.

• Archive process values and alarms

The HMI system can log alarms and process values. In this way, you can document the process history. As a result you can later access older production data.

• Document process values and alarms

The HMI system can print out alarms and process values as reports. This allows you to output production data at the end of a shift, for example.

• Manage process and machine parameters in recipes

The HMI system can store parameters for processes and machines in recipes. For example, you can transfer these parameters from the HMI device to the controller in one step in order to switch the production to a different product version.

• User management

Certain rights can be assigned to the devices, thereby limiting the possible operator inputs for particular users.

4.2 SIMATIC HMI Panel KTP700 Basic

4.2.1 Device description

The SIMATIC HMI Basic Panels product line features key and touch panels (operator input via keyboard and touch screen)

SIMATIC HMI Basic Panels cover all requirements described in the previous section.

These HMI devices are explained in this document using the KTP700 Basic as an example.

Figure 1: KTP700 Basic

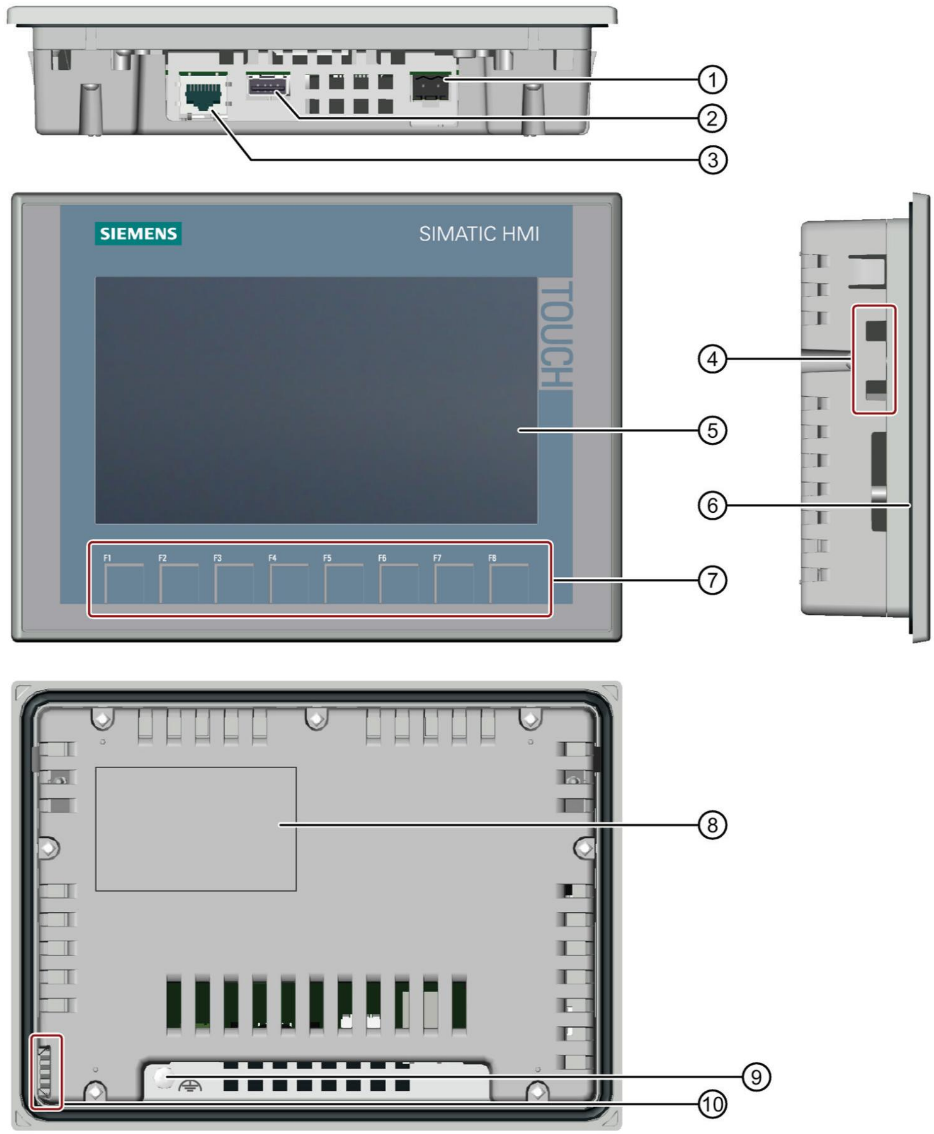
The WinCC Basic (TIA Portal) software is needed for configuration and programming. This software is included in the SCE Trainer Package **"SIMATIC HMI KTP700 BASIC COLOR PANEL for S7-1200"**!

Notes:

Because all the devices in this product series have similar functionality, it would also be possible to implement this section with another product device in this series.

The Touch Panel KTP700 Basic can also be displayed on the PC as Runtime simulation with WinCC Basic.

4.2.2 Design of the KTP700 Basic for PROFINET



① Connection for power supply

② USB interface for USB mass storage device or USB mouse

③ PROFINET interface

④ Recesses for a mounting clip

⑤ Display/touch screen

⑥ Mounting gasket

⑦ Function keys

⑧ Rating plate

⑨ Connection for functional ground

⑩ Guide for labeling strip

12

4.2.3 Memory concept

The HMI devices can use the following memory:

– *Internal memory*

– *USB mass storage on USB interface*

Internal memory

The following data is stored here:

– *Operating system*

– *Project file*

– *License keys*

– *User management*

– *Recipes*

USB mass storage on USB interface

The following data can be stored here:

– *Operating system for update*

– *Project file as backup*

– *User management as backup*

– *Recipes as backup*

– *Recovery software for resetting to factory settings via USB*

– *License keys for transfer to the panel*

– *Certificates for web-based communication*

12

4.2.4 Settings on Touch Panel KTP700 Basic/Start Center

Several important settings must be made directly on the Touch Panel KTP700 Basic.

The Touch Panel KTP700 Basic runs with the Windows CE operating system. Similar to all touch panels, operator inputs can be made directly on the screen. For better performance, you should use a special touch pen or connect a mouse to the panel's USB port.

After startup of the panel, the **'Start Center'** window appears.

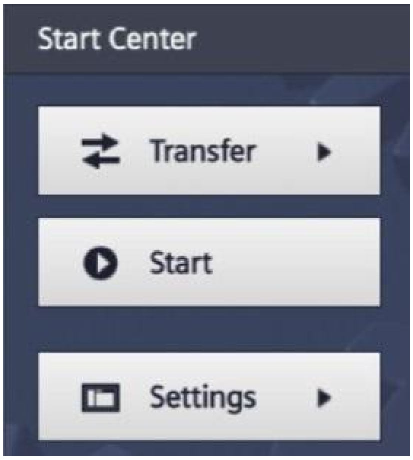
Buttons in the Start Center:

Transfer: Data transfer is activated and the panel waits for the configuration data to be downloaded from WinCC to the PC. "Transfer" mode can only be activated when at least one data channel is enabled for the transfer.

Start: Runtime is started and the process visualization appears on the panel. Often the panel is set in such a way that the start occurs automatically after a few seconds.

Settings: The Windows CE settings dialog is opened. Settings for the panel can be made here: You make various settings on this page, such as settings for the transfer.

→ Select → the "Settings" button in the "Start Center" immediately after switching on the power supply and starting the panel.



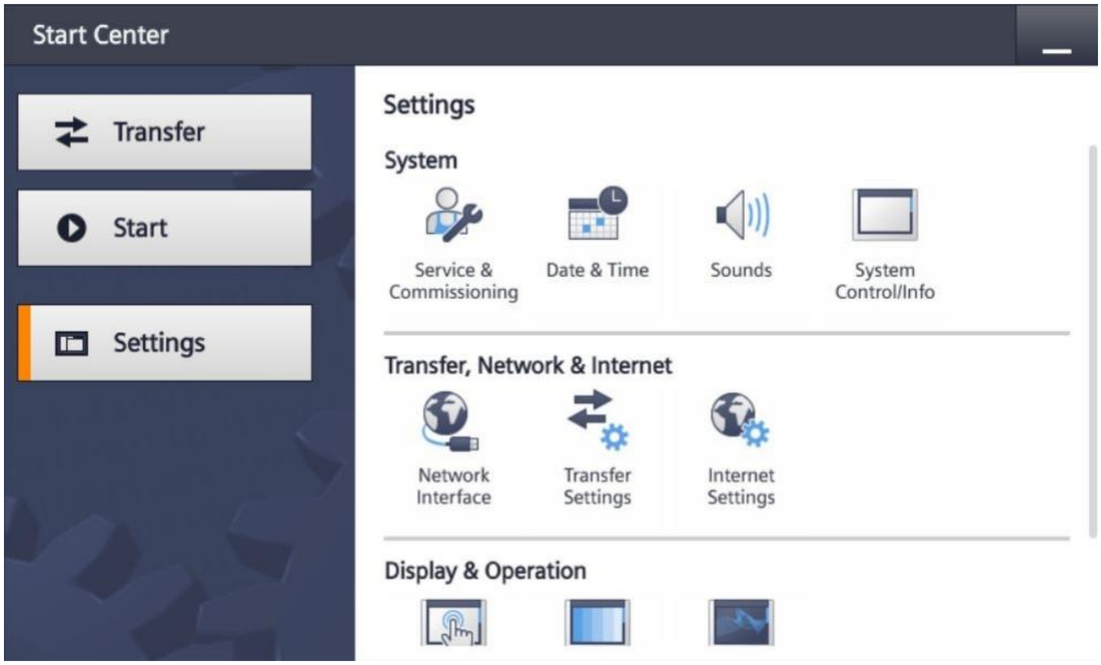
Note:
You must select "Settings" in the "Start Center" quickly before Runtime automatically starts.

12

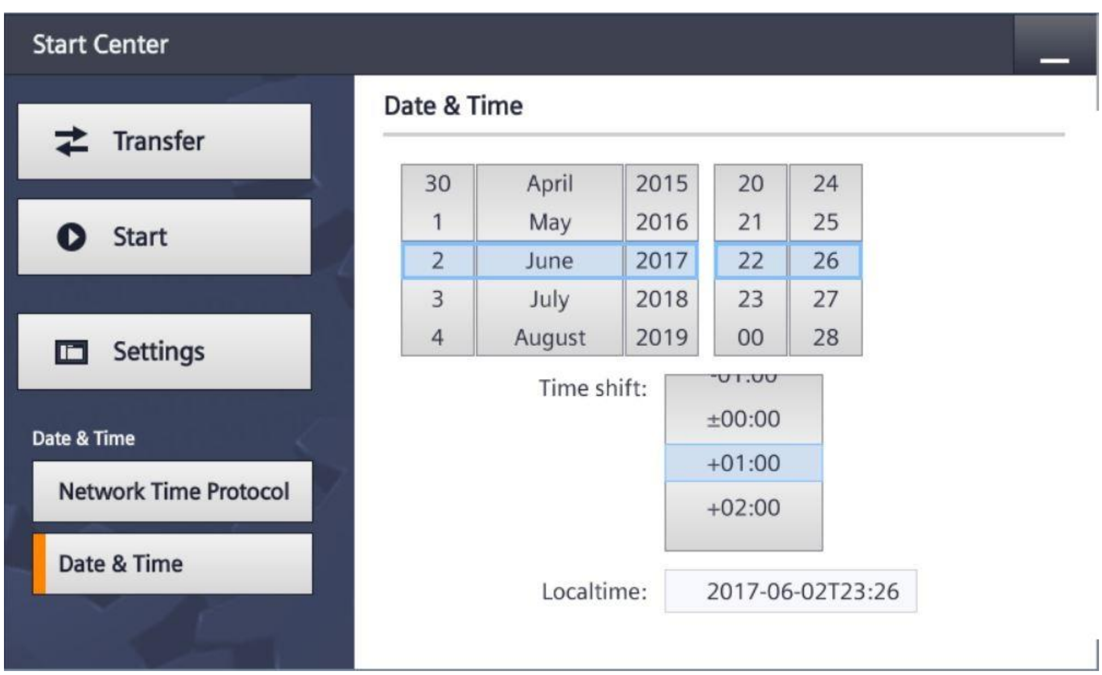
4.2.5 Setting the date and time

→ Under "System", select the 

icon to make the date and time settings.

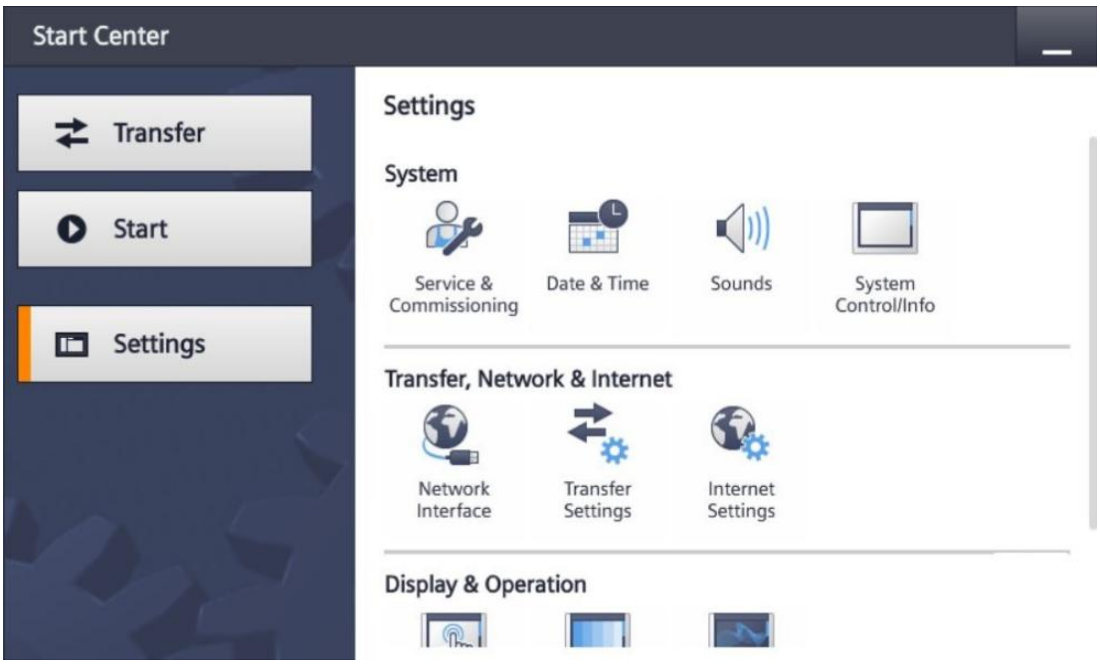


→ Under "Date & Time", set the time zone ("Time shift") and the date and time.

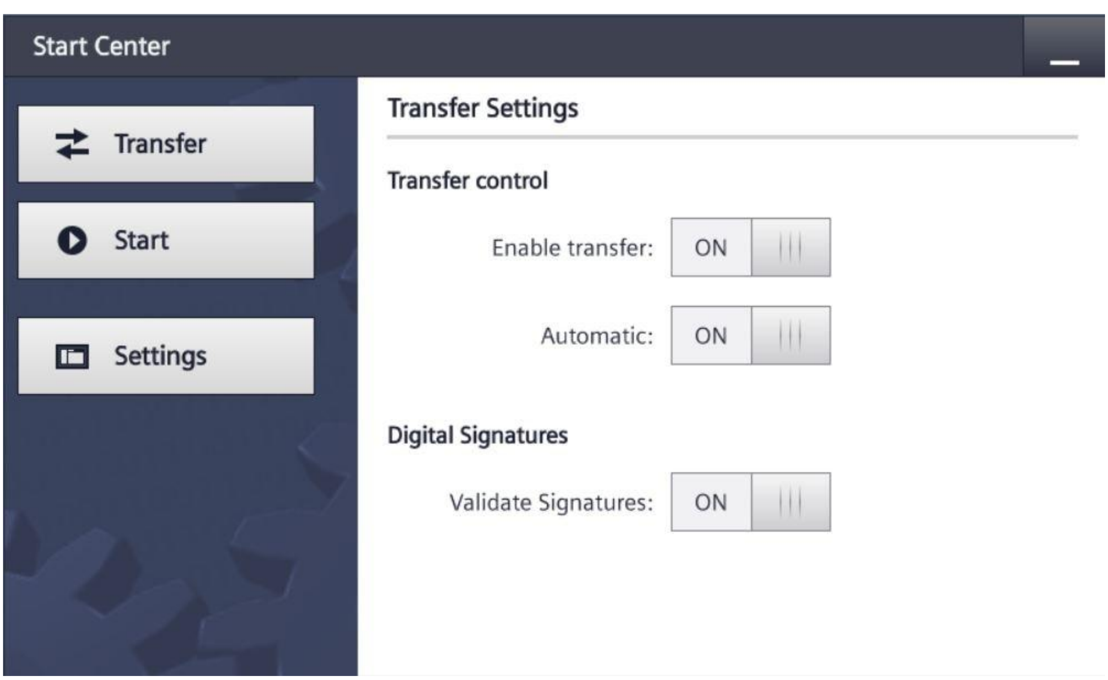


4.2.6 Setting the transfer properties and assigning the IP address

→ Under "Transfer, Network & Internet", select the  icon to navigate to the transfer properties.



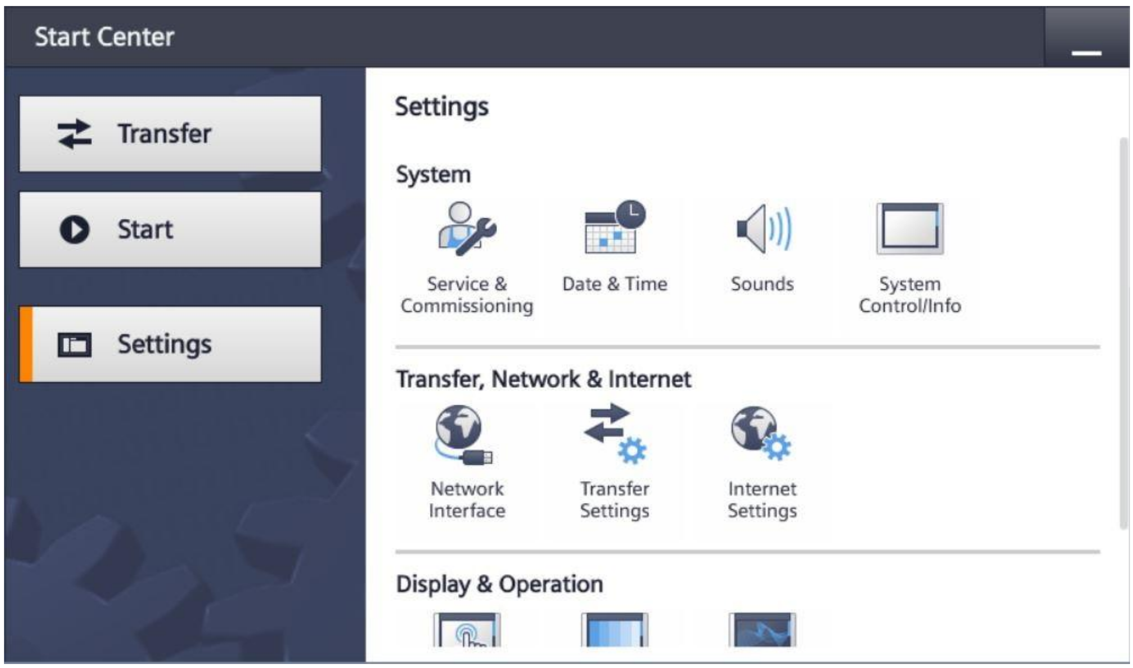
→ Select the following settings for the "Transfer Settings".



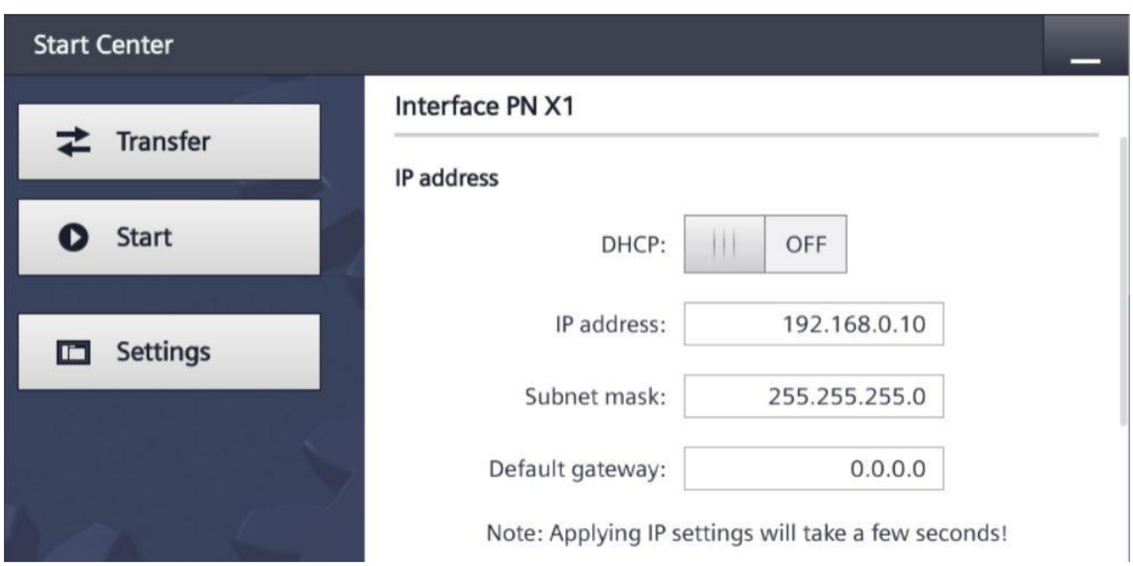
12

→ Under "Transfer, Network & Internet", select the 

icon to navigate to the network settings.



→ In menu item "Interface PN X1", set the IP address under "IP address" and the subnet mask under "Subnet mask".

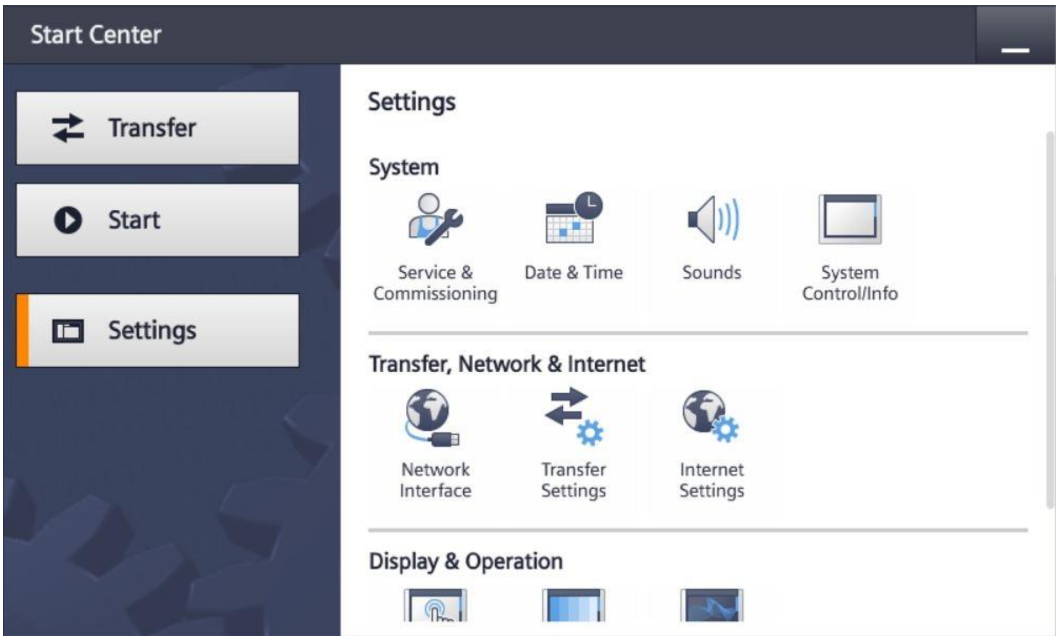


12

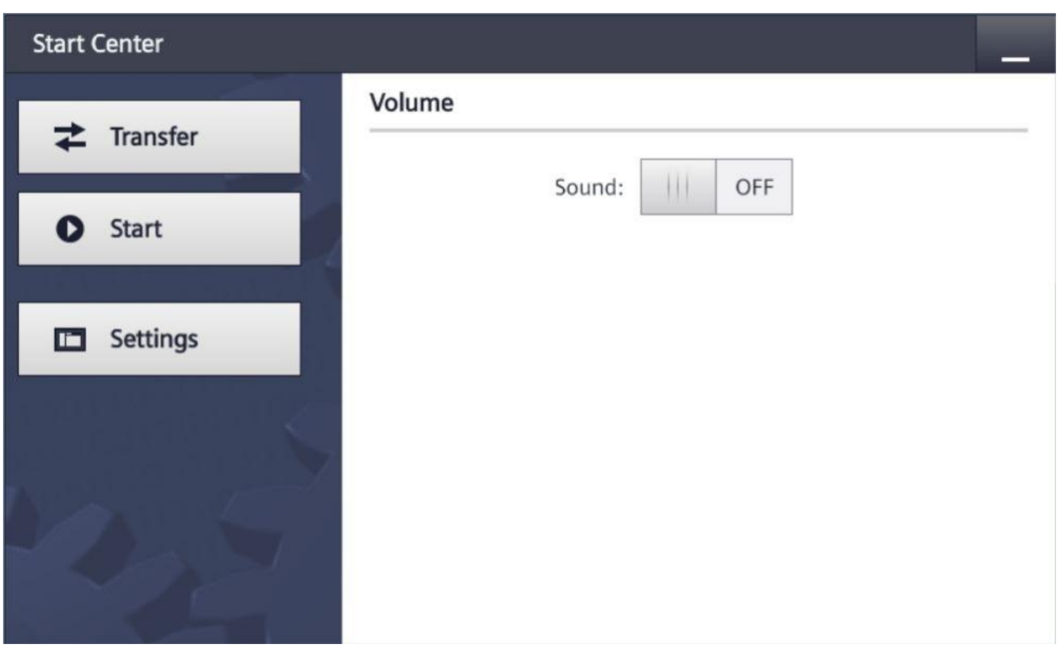
4.2.7 Switching off the sound on the touch panel

→ Under "System", select the 

icon to navigate to the sounds settings of the touch panel.



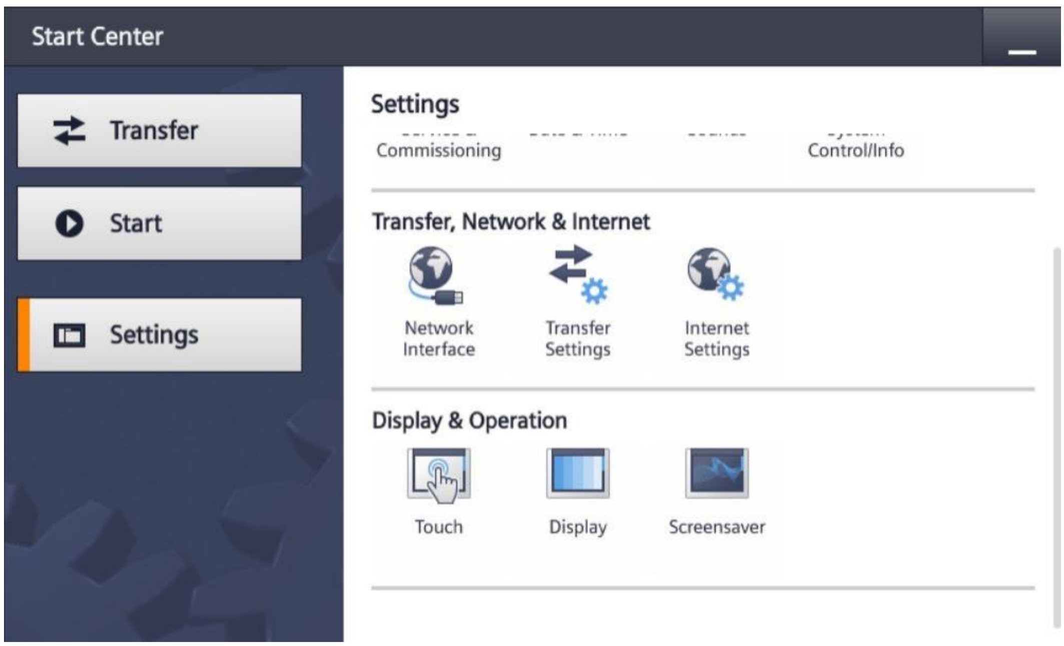
→ Under "Volume", switch → "Sound" to "OFF".



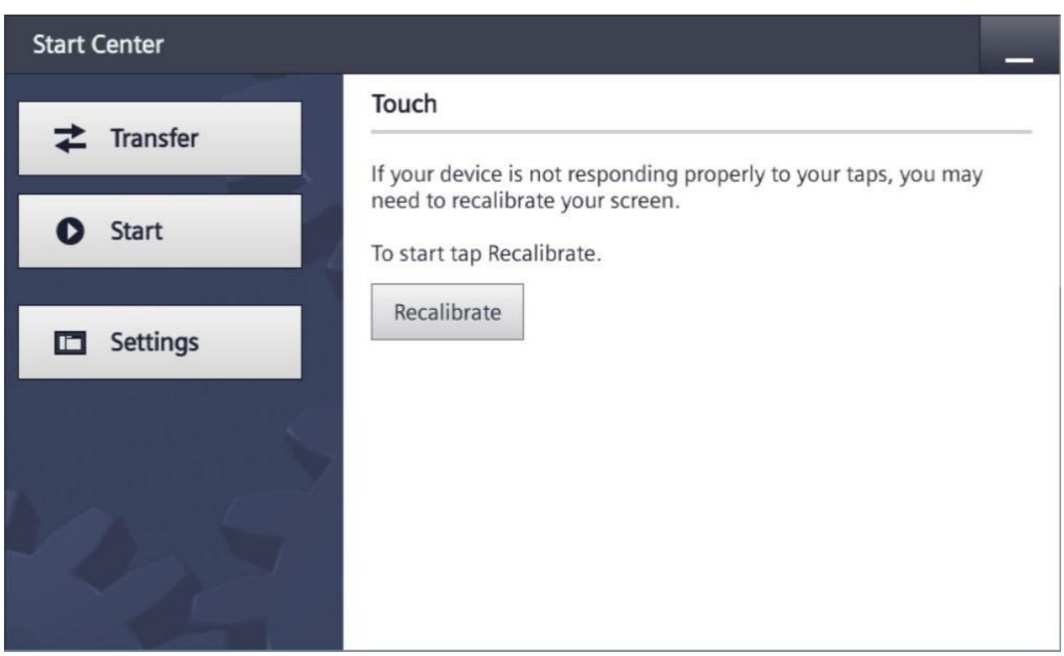
4.2.8 Calibrating the touch panel

→ Under "Display & Operation", select the 

icon to navigate to the calibration of the touch panel.

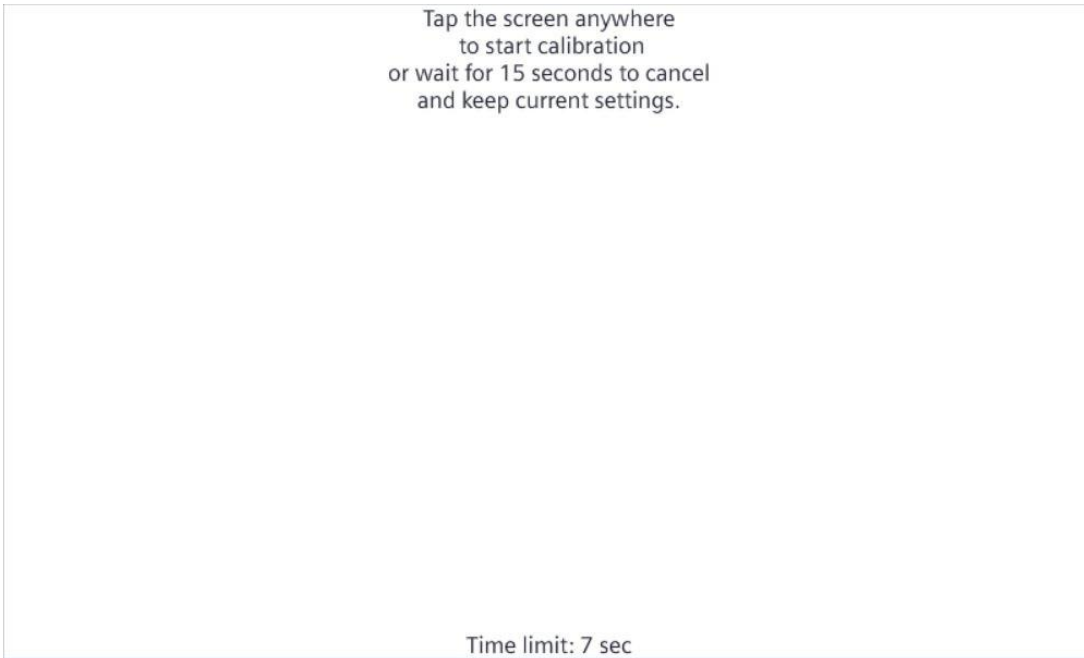


→ Select the "Touch" menu item. Start the calibration with → "Recalibrate".

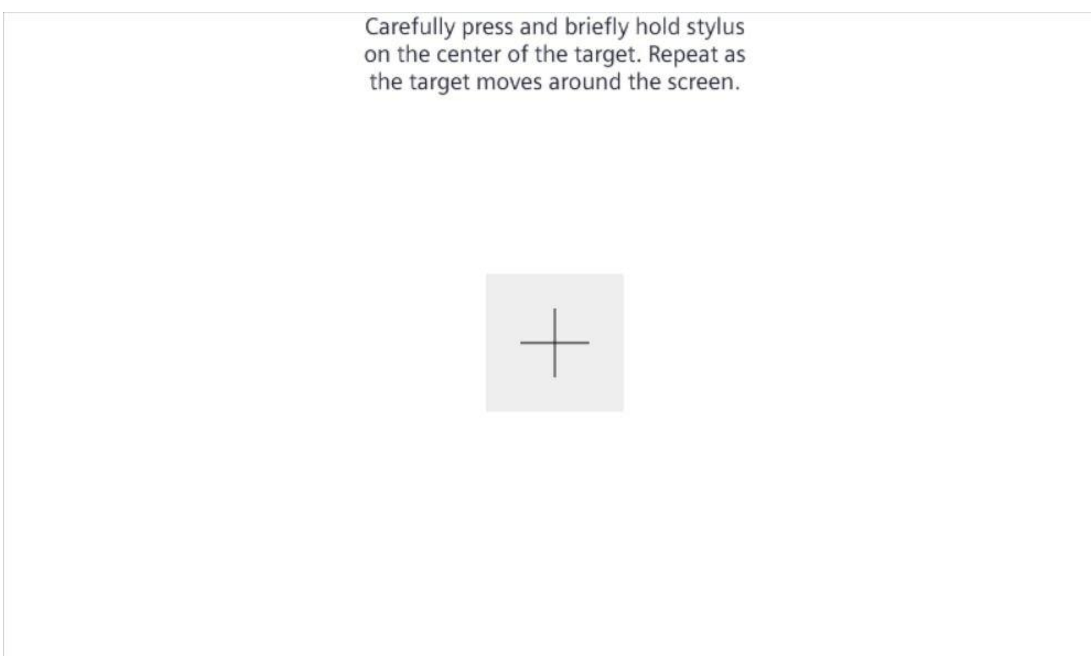


12

→ Touch anywhere on the screen within 15 seconds to start the calibration.



→ Follow the instructions on the touch panel and press as close to the center of the displayed cross as possible.



12

4.3 WinCC Basic programming software

The WinCC Basic software is included in the TIA Portal as an integral component of STEP 7 Basic or STEP 7 Professional and is the programming tool for the following visualization system:

– *SIMATIC Basic Panels*

With WinCC Basic you have the following functions for creating HMI systems:

– *Hardware configuration and parameter assignment*
– *Specification of communication and creation of a connection to a PLC*
– *Creation and layout of screens with hierarchical structure*
– *Creation of internal and external tags*
– *Creation of alarms and alarm displays*
– *Creation and display of logs as trends and in tables*
– *Creation of recipes and recipe displays*
– *Creation and printing of reports*
– *Testing, commissioning and service with operational/diagnostic functions*
– *Documentation*

All functions are supported by an extensive online help.

4.3.1 Project

For the solution to an automation and visualization task, you create a project in the TIA Portal. A project in the TIA Portal contains data on the configuration and networking of devices as well as the programs and the configuration of the visualization.

4.3.2 Hardware configuration

The *hardware configuration* contains the configuration devices, consisting of the automation system hardware, the field devices on the PROFINET bus system and the hardware for visualization. The configuration of the networks specifies the communication between the various hardware components. Individual hardware components are inserted from catalogs into the *hardware configuration.*

The hardware of SIMATIC S7-1200 automation systems consists of the controller (CPU), the signal modules for input and output signals (SM), the communication modules (CM) and other special modules.

The signal modules and field devices connect the input and output data of the process to be automated and visualized to the automation system.

The hardware configuration enables automation and visualization solutions to be downloaded to the automation system and the controller to have access to the connected signal modules.

4.3.3 Planning the hardware

Before you can configure the hardware, you must plan it. In general, you start by selecting the required controllers and the number needed. Next you select the communication modules and signal modules. The signal modules are selected based on the number and type of inputs and outputs needed. Finally, a power supply must be selected for each controller or each field device to ensure the required power is supplied.

The functionality needed and the environmental conditions are of critical importance for planning the hardware configuration. For example, the temperature range in the application area is one of the limiting factors for selecting possible devices. Fail-safe capability could be another requirement.

The TIA Selection Tool (select Automation System → TIA Selection Tool and follow the instructions) offers you support.

Notes:

– TIA Selection Tool requires Java

– Online research: To obtain the device specifications, look for the manual described as "Product Manual" or "Manual" among the various manuals listed.

There are centralized and distributed applications available for the visualization. For local, distributed operator input, panels are often used. These can communicate with the controller via Ethernet, WLAN or fieldbus. For central operator control and monitoring, PC can also be used. These are usually connected to the controller via Ethernet.

The TIA Selection Tool also supports you when selecting panels (select Automation System → TIA Selection Tool and follow the instructions).

12

4.3.4 Planning the screen structure

After selecting a device for the visualization, the screen structure must be planned. To do this, you should assemble, group and structure the information to be displayed. From this it should be possible to derive a screen structure like the one shown in Figure 2. The entry point to the screen structure is always ensured by a "root screen".

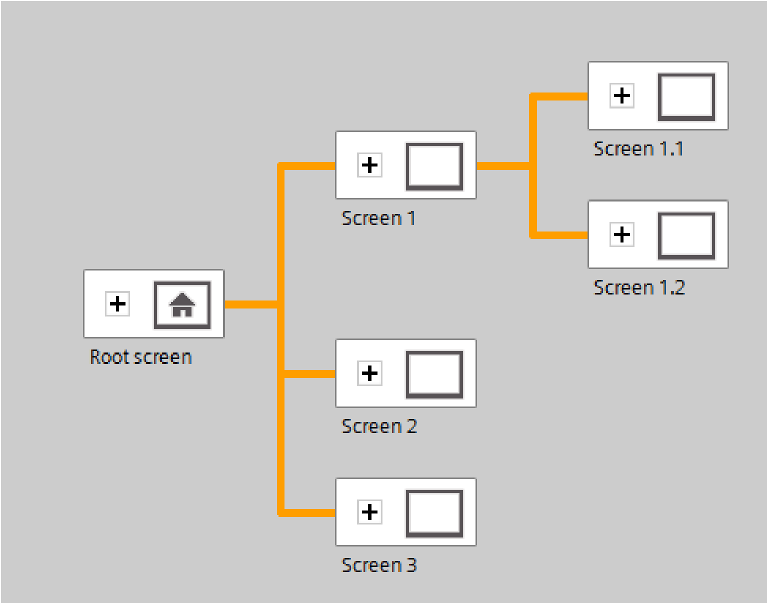


Figure 2: Example of screen structure

The determining factor for the screen structure should be its support of the operator when navigating through the information distributed among the screens for operator control and monitoring of the process.

The following questions may be helpful:

What conceptual model of the process must be followed for the information display?

What data belong together?

What data belong in a specific sequence?

What data belong to specific operations/processes?

Are there data and the like that apply across operations?

What data represent key information and what data are additional information?

12

4.3.5 Planning of screen design

Every individual screen must also be planned. Considerations regarding use by the operator are also necessary for the information display. It is helpful here to follow design principles such as the principles of proximity, similarity and symmetry. The following rules of thumb derived from design principles can also support the design of screens:

→ Form groups of data blocks

→ Uniform subdivision of the entire screen into work information, status or system information and controller information

→ Observe average distribution of attention on the screen as a function of reading direction

→ Use compactness as design principle (numbers, column headings same as column content)

→ Make appropriate use of 30-40 % of available space: Accommodate as little information as possible and as much as necessary

→ Economical coding (for example, color, bold text, lightness, shape, border, appearance, flashing)

→ Organize numbers: organize numbers with more than 4 digits in groups of 2, 3 or 4 (for example, 66 234)

→ Preferably choose numerals in listing of objects, properties, etc.

→ Use and position designations uniformly

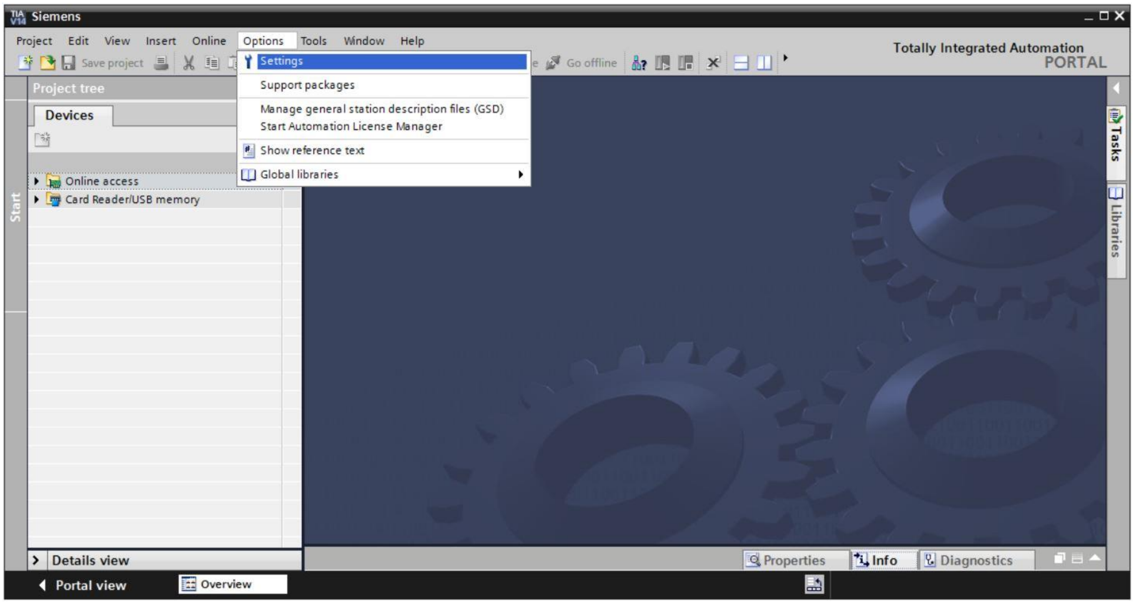
→ Use short words if possible

12

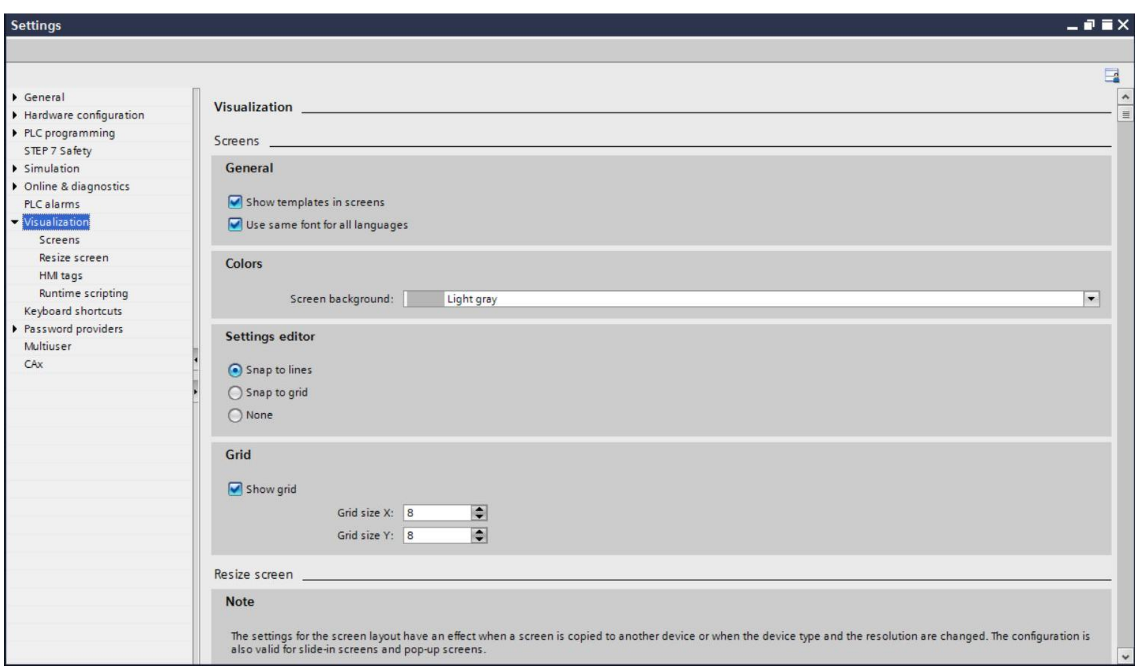
4.3.6 Basic settings for WinCC Basic in the TIA Portal

The user can make customized presets for certain settings in the TIA Portal. The method for making the settings for the visualization is shown here.

→ In the Project view, select menu command → "Options" and then → "Settings".



→ In the → "Visualization" item of Settings, select the desired presets for the design of the user interface.



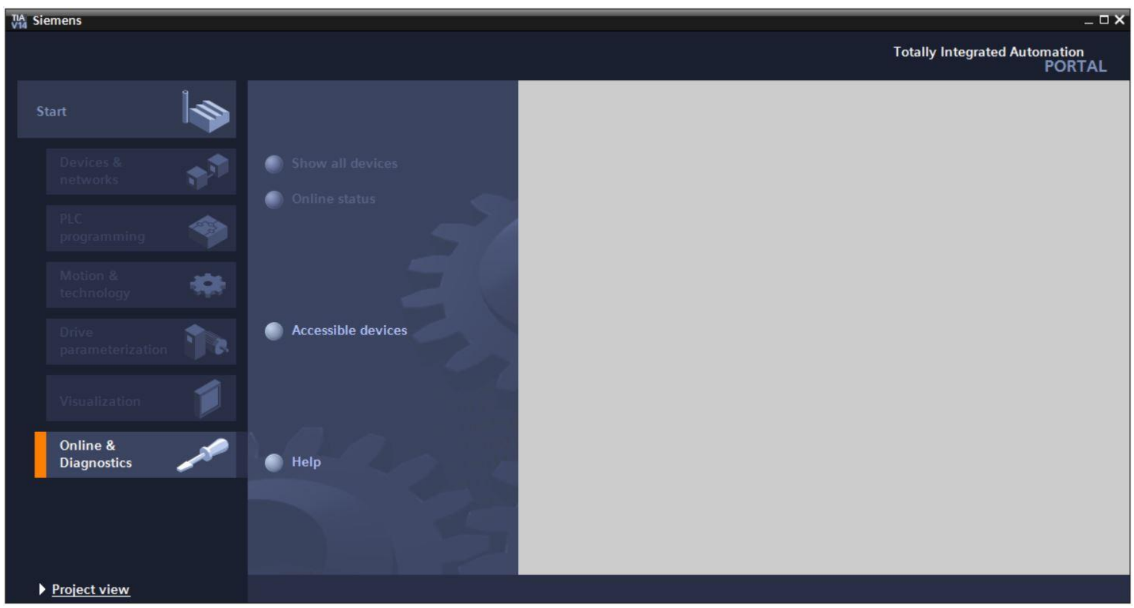
Note: Keep the default settings as the settings for the visualization here.

4.3.7 Resetting the SIMATIC HMI Panel KTP700 and setting the IP address

The HMI Panel KTP700 Basic can be reset directly in the TIA Portal. A new IP address can also be assigned to the panel there.

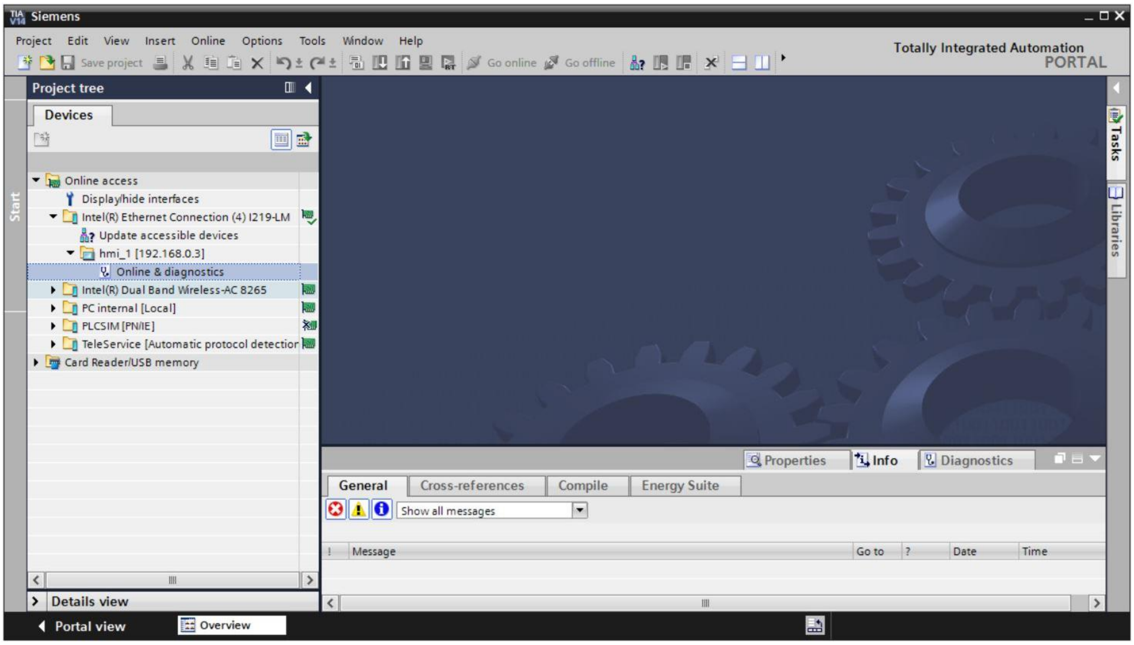
To do this, select the Totally Integrated Automation Portal, which is opened by a double-click. (→ TIA Portal V14)

→ Click the → "Online & Diagnostics" item and open the → "Project view".

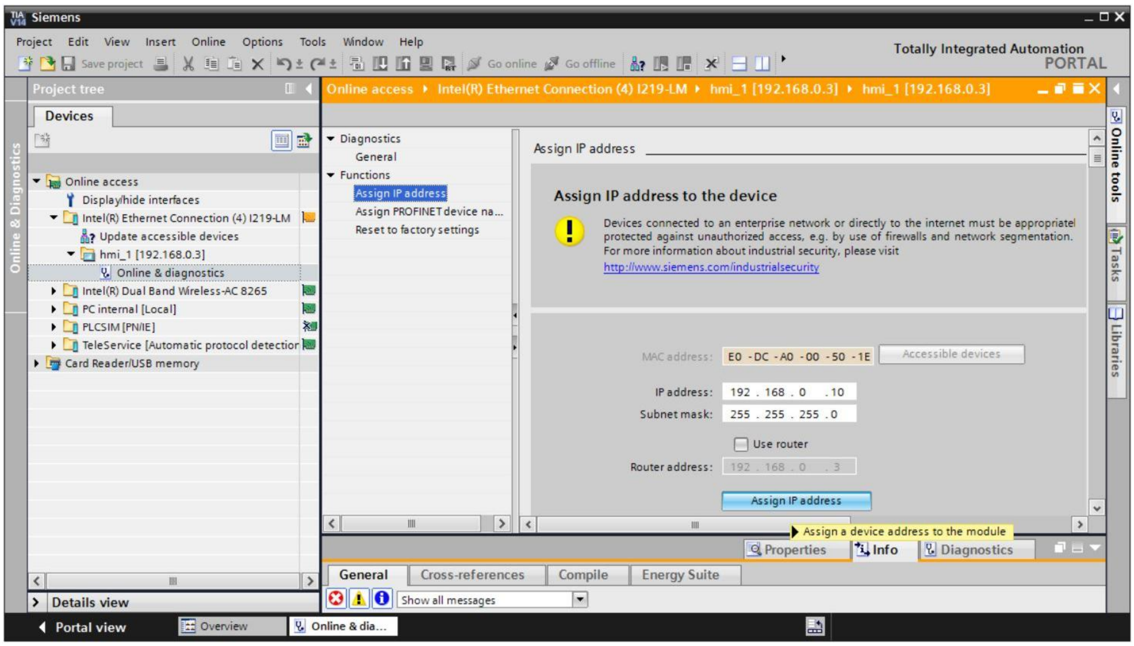


12

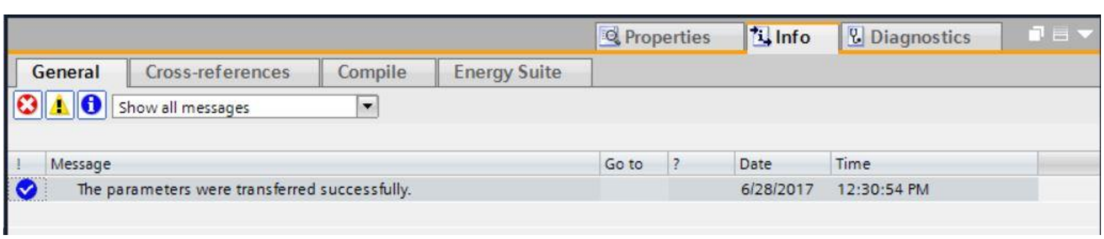
→ In the project tree, select the network adapter of your computer under → "Online access". When you click → "Update accessible devices", you see the IP address (if already set) or the MAC address (if the IP address has not yet been assigned) of the connected SIMATIC HMI Panel → Click → "Online & diagnostics".

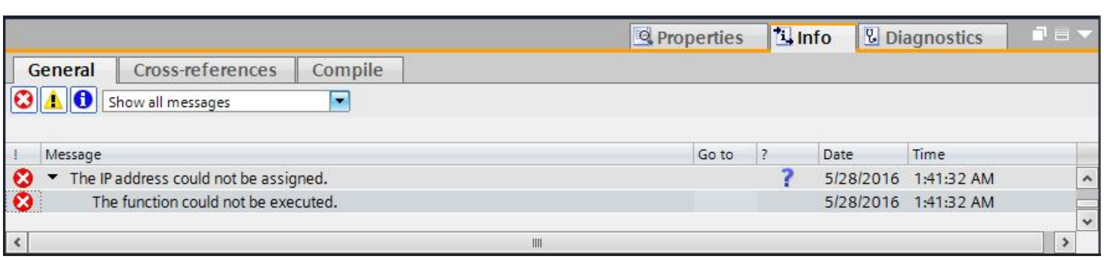


→ To assign the IP address, select the → "Assign IP address" function. For example, enter the following IP address and subnet mask here: → IP address: 192.168.0.10 → Subnet mask: 255.255.255.0. Next, click → "Assign IP address". The new address is assigned to your SIMATIC HMI Panel KTP700 Basic.



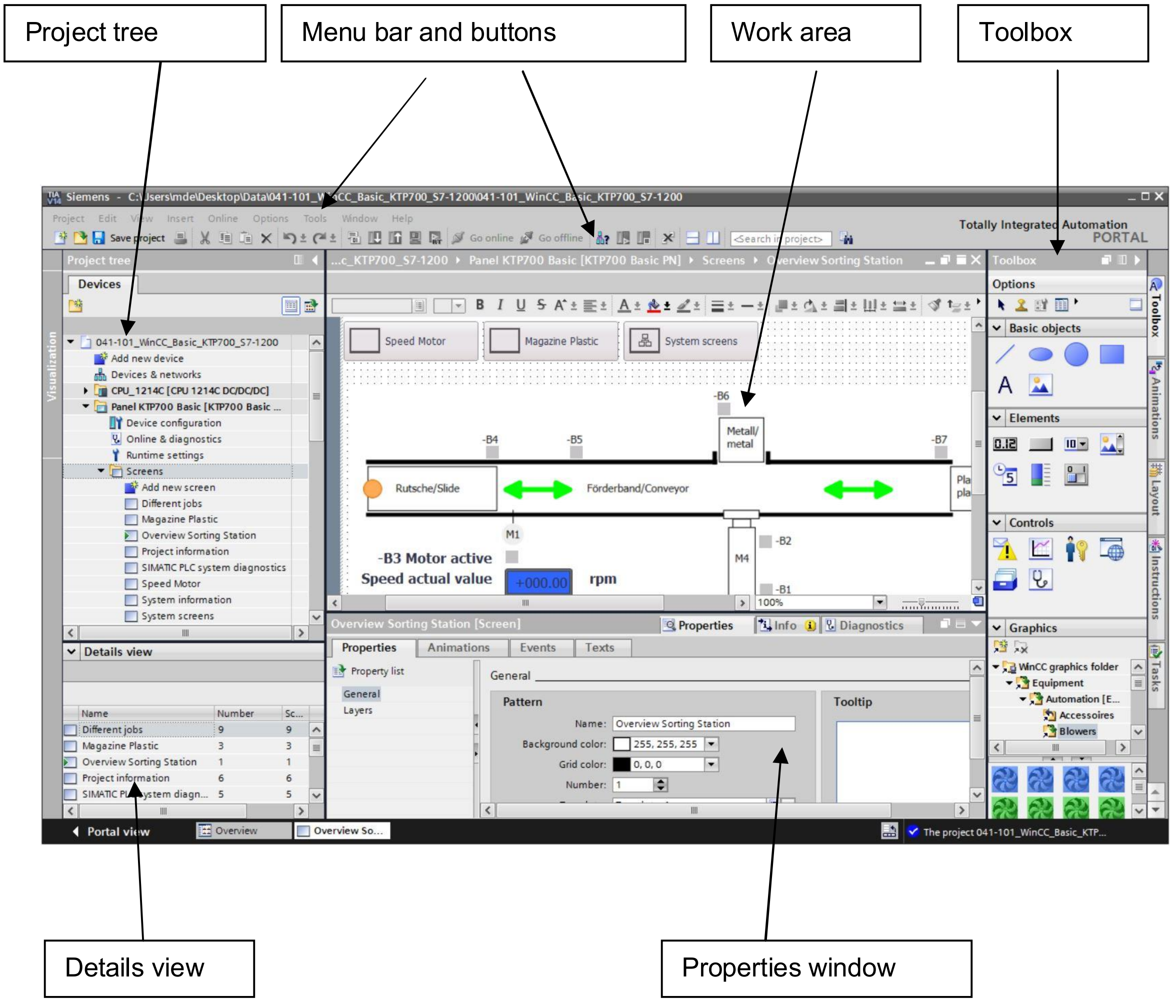
→ The successful (or unsuccessful) assignment of the IP address is shown as a message in the → "Info" window, → "General" tab.





Note: If there are problems assigning the IP address, the IP address of the SIMATIC HMI Panel KTP700 Basic can also be set via Windows CE of the panel.

4.3.8 WinCC user interface

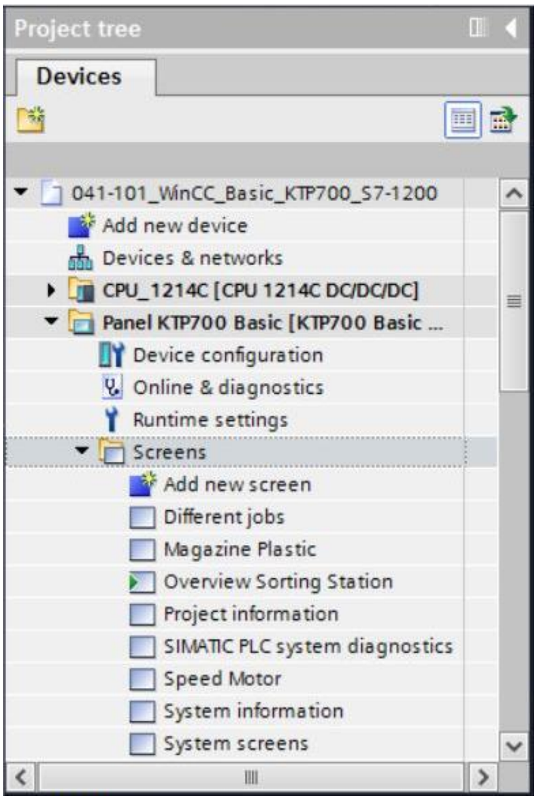


4.3.9 Project tree

The project tree is the central control point for the project handling. All component parts and all the available editors of a project are displayed in a tree structure in the project window and can be opened from there.

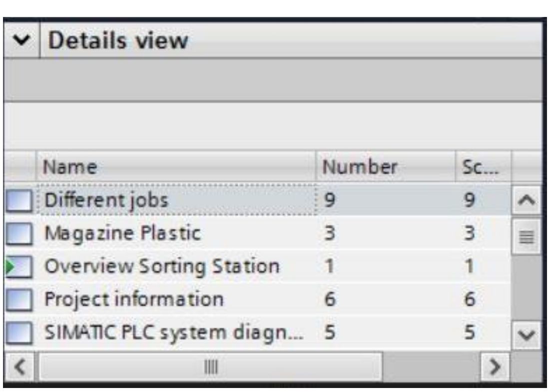
Each editor is assigned a symbol which you can use to identify the corresponding objects. Only elements that are supported by the selected HMI device are displayed in the project window.

In the project window, you have access to the device settings of the HMI device.



4.3.10 Details view

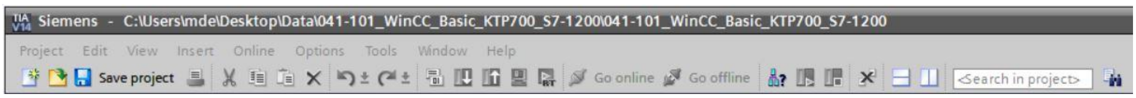
The Details view shows the contents or other information on the objects selected in the project tree.



4.3.11 Menu bar and buttons

The menus and toolbars provide access to all functions you need to configure your HMI device. When a corresponding editor is active, editor-specific menu commands and toolbars are visible.

When the mouse pointer is moved over a command, you receive a corresponding tooltip for each function.

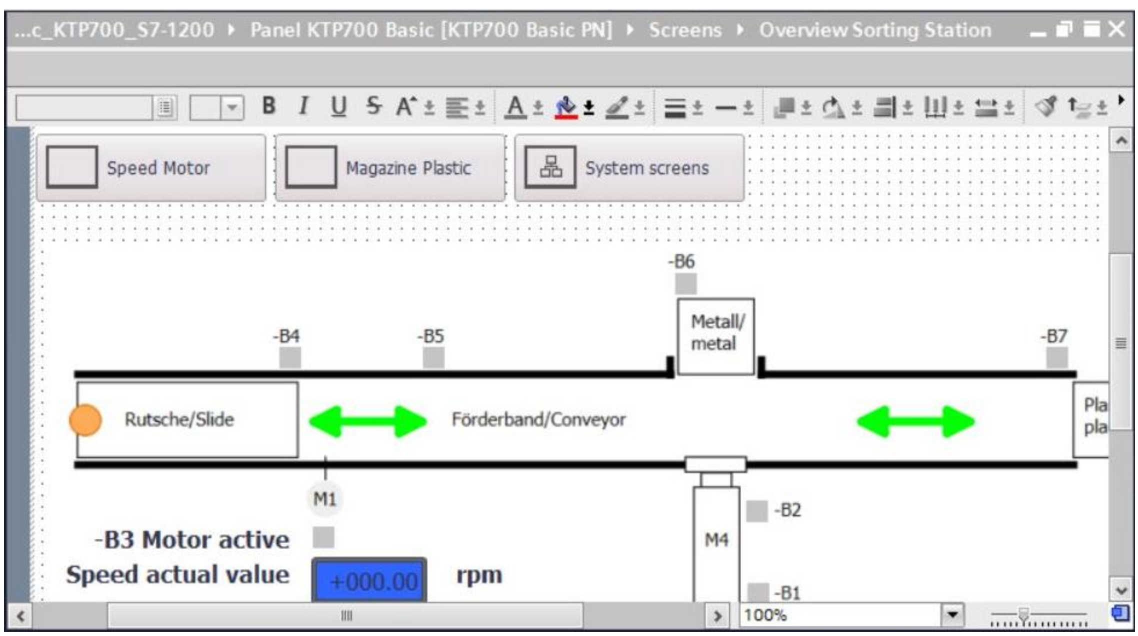


4.3.12 Work area

You edit the objects of the project in the work area. All other elements of WinCC are arranged on the borders of the work area.

Project data can also be edited here either in tabular form (e.g. tags) or graphically (e.g. process screens).

A toolbar is located at the top of the work area. For example, you can select formatting such as font and font color and functions such as rotate, align, etc. here.



4.3.13 Toolbox

In the toolbox window, you will find a selection of objects that you can insert in your screens, e.g. graphic objects and operator controls. In addition, the toolbox window also contains libraries with ready-made graphic objects and collections of faceplates.

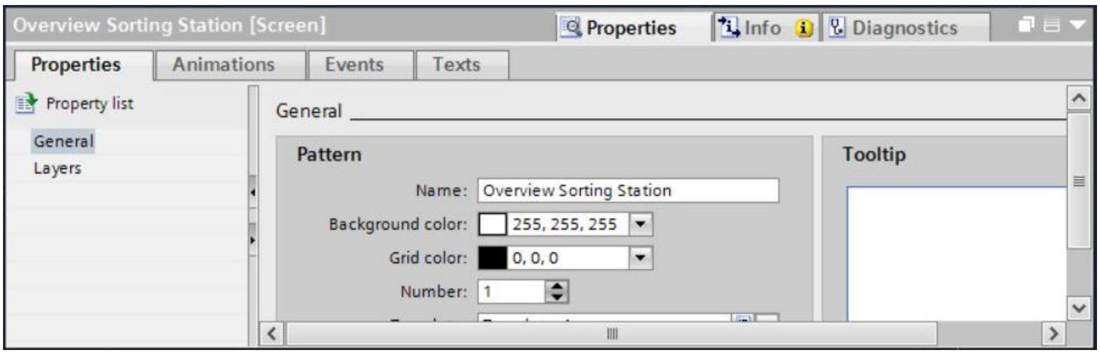
The objects are moved to the work area using drag & drop.

4.3.14 Properties window

In the properties window, you edit the properties of the objects selected in the work area, e.g. the color of screen objects. The window is only available in specific editors.

The properties window also shows the properties of the selected object, organized by category. The changed values take effect as soon as the input field is exited. If you enter an invalid value, it is highlighted in color. The tooltip gives you information about the valid value range, for example.

Animations (e.g. color change at a change of signal state in the PLC) and events (e.g. screen change when a button is released) are also configured for a selected object in the properties window. Multilingual texts can also be managed here.

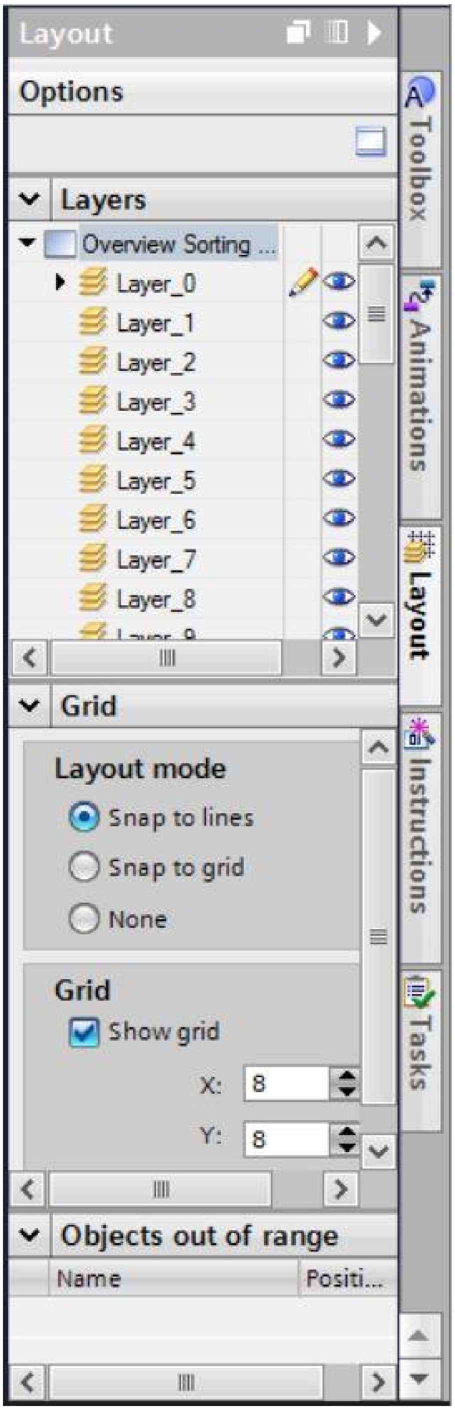


12

4.3.15 Additional tabs

The settings of the work area, such as the layer selection and the grid functions, can be made in the "Layout" window.

Animations, instructions, tasks and libraries of the selected object can be selected via other tabs.



5 Task

In this section, a process visualization is to be added to the program from section "SCE_EN_031-600 Global Data Blocks for S7-1200". This enables you to better monitor and process flow and control it more effectively.

6 Process visualization planning

A Touch Panel KTP700 Basic is to be used for the process visualization.

The programming device, a SIMATIC S7-1200 controller and the Touch Panel KTP700 Basic are connected to one another via the **Ethernet interface** using a SCALANCE XB005 UNMANAGED INDUSTRIAL ETHERNET SWITCH.

The wizard in the TIA Portal is to be used for the basic configuring. In doing this, all **system screens** are also to be created.

The process will be represented with the conveyor and sensors in an **"Overview Sorting Station"** overview screen. Conveyor speed and the plastic parts count will also be displayed here.

This operator is to be able to select the operating mode, start and stop in automatic mode and reset the counter in this screen.

In an additional **"Speed Motor"** screen, the actual speed of the motor will be graphically displayed. The speed setpoint can also be specified here.

The **"Magazine Plastic"** screen will be initially only created.

The screen name, date/time and system states "Emergency stop ok/released", "Main switch ON/OFF" and "Automatic started/stopped" are displayed in the **header**.

The **footer** has a button for returning to the root screen, a button for displaying the alarm window and a button for ending Runtime mode.

The **alarm system** is also to be configured.

12

System events of the panel are to be displayed as alarms and motor speed limit violations of the main switch are to be monitored.

In doing so, the alarms are automatically displayed in alarm windows when errors/warnings occur.

6.1 Program description for the sorting station with motor speed control and monitoring

The "MOTOR_AUTO" [FB1] function block controls a **conveyor in automatic mode**.

The Memory_Automatic_Start_Stop is switched on in a latching manner with Start, but only if the reset conditions are not present.

The Memory_Automatic_Start_Stop is to be reset if Stop is pending, the safety shutoff is active or automatic mode is not activated from the visualization.

The Automatic_Motor output is controlled when Memory_Automatic_Start_Stop is set, the enabling conditions are met and Memory_Conveyor_Start_Stop is set.

For energy saving reasons, the conveyor is to run only when a transport part is present. Therefore, Memory_Conveyor_Start_Stop is set if Sensor_slide signals a part and is reset if Sensor_end_of_conveyor generates a negative edge or the safety shutoff is active or automatic mode is not activated (manual mode).

Because Sensor_end_of_conveyor is not mounted directly at the end of the conveyor, a signal delay of the Sensor_end_of_conveyor signal is programmed.

The magazine for plastics holds only five parts. Therefore, the parts are counted at the end of the conveyor. If the magazine contains five parts, automatic mode is to be stopped. After the magazine is emptied, automatic mode is restarted with another Start, once the counter has been reset from the visualization.

The **speed is specified** with an input in the "MOTOR_SPEEDCONTROL" [FC10] function in revolutions per minute (range: +/- 50 rpm).

The speed setpoint is first checked in the function for correct input in the range +/- 50 rpm.

If the speed setpoint is outside the range +/- 50 rpm, the value 0 is output at the speed setpoint output. The value TRUE (1) is assigned to the return value of the function (Ret_Val).

If the specified speed is in the range +/- 50 rpm, this value is first scaled to the range 0…1 and then scaled for the output as manipulated speed value at the analog output to +/- 27648 with data type 16-bit integer (Int).

12

In the "MOTOR_SPEED_MONITORING" [FC11] function, the actual value is made available at -B8 as an analog value and is queried at an input of the "MOTOR_SPEEDMONITORING" [FC11] function.

The actual speed value is scaled to revolutions per minute (range: +/- 50 rpm) and made available at an output.

The following four limits can be specified at the block inputs in order to monitor them in the function:

Speed > Speed limit error max

Speed > Speed limit warning max

Speed < Speed limit warning min

Speed < Speed limit warning min

If a limit is exceeded or undershot, the value TRUE (1) is assigned to the corresponding output bit.

If a fault is present, the safety shutoff of the "MOTOR_AUTO" [FB1] function block is to be triggered.

Speed setpoint and actual speed value as well as the positive and negative error and warning limits are created in the "SPEED_MOTOR" [DB2] data block, as are the error and warning bits.

The setpoint and actual value of the counter for plastic parts will be specified and displayed in the global "MAGAZINE_PLASTIC" [DB3] data block. These values are connected with the "MOTOR_AUTO" [FB1] function block via an input for specification of the setpoint and via an output for display of the actual value.

6.2 Technology schematic diagram

Here you see the technology schematic diagram of the system for the task.

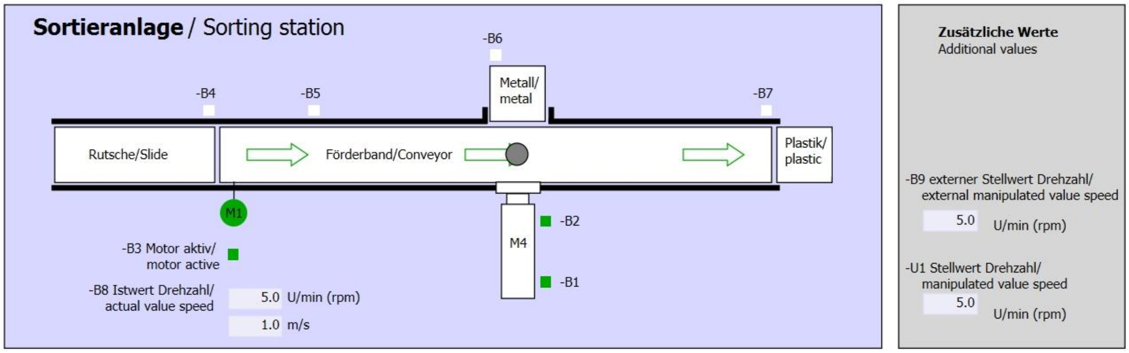


Figure 3: Technology schematic diagram

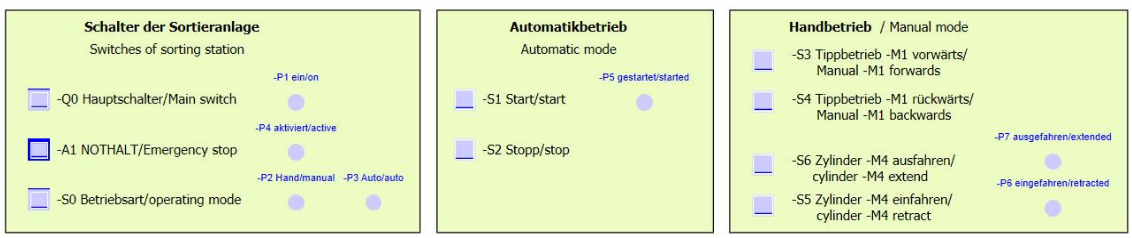


Figure 4: Operator panel

12

6.3 Reference table

The following signals are required as global operands for this task.

DI	Type	ID	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop ok	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0) / automatic (1)	Manual = 0 Auto=1
I 0.3	BOOL	-S1	"Automatic start" pushbutton	NO
I 0.4	BOOL	-S2	"Automatic stop" pushbutton	NC
I 0.5	BOOL	-B1	Sensor cylinder -M4 retracted	NO
I 1.0	BOOL	-B4	Sensor part at slide	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO
IW64	BOOL	-B8	Sensor actual motor speed +/-10V corresponds to +/- 50 rpm	

DO	Type	ID	Function	
Q 0.2	BOOL	-Q3	Conveyor motor -M1 variable speed	
QW 64	BOOL	-U1	Motor speed manipulated variable in both directions +/-10V corresponds to +/- 50 rpm	

Legend for reference list

DI	Digital input	DO	Digital output
AI	Analog input	AO	Analog output
I	Input	O	Output
NC	Normally Closed		
NO	Normally Open		

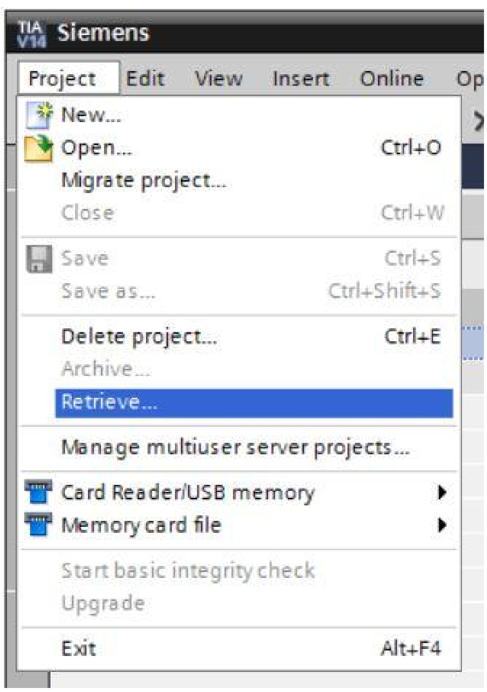
12

7 Structured step-by-step instructions

Here you will find an example of instructions for the planning task. If you already have a good understanding of everything, it is sufficient to focus on the numbered steps. Otherwise, follow the step-by-step instructions below.

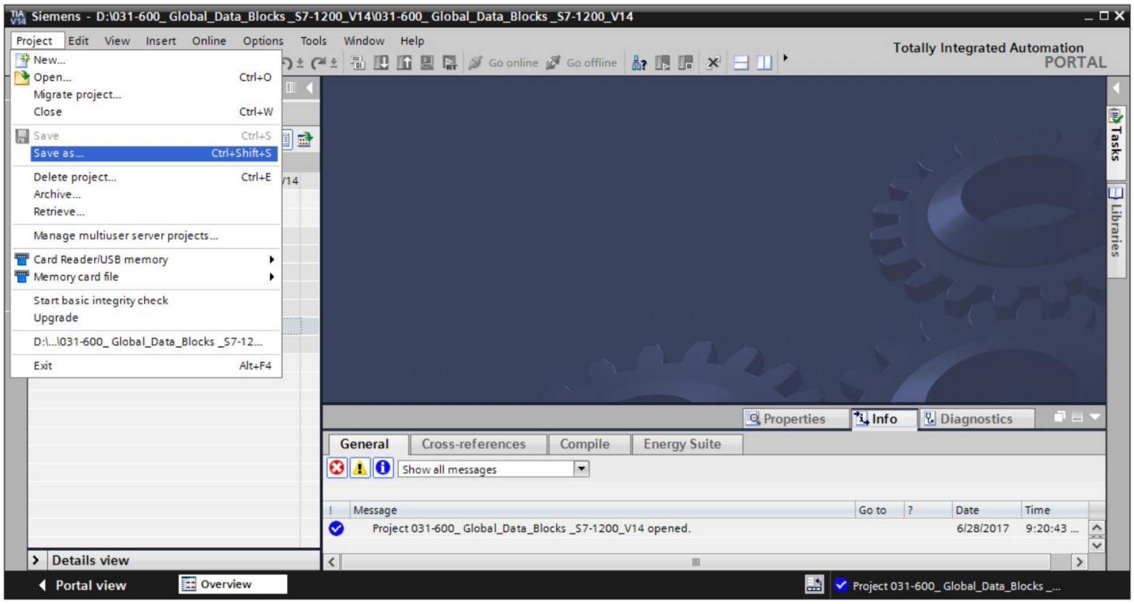
7.1 Retrieving an existing project

→ Before you can expand the "SCE_EN_031-600 Global_Data_Blocks_S7-1200…..zap14" project from section "SCE_DE_031-600 Global Data Blocks for S7-1200", you must retrieve it. To retrieve an existing project, you must select the respective archive from the Project view below → Project → Retrieve. Confirm your selection with Open.
(→ Project → Retrieve → Selection of a .zap archive … → Open)



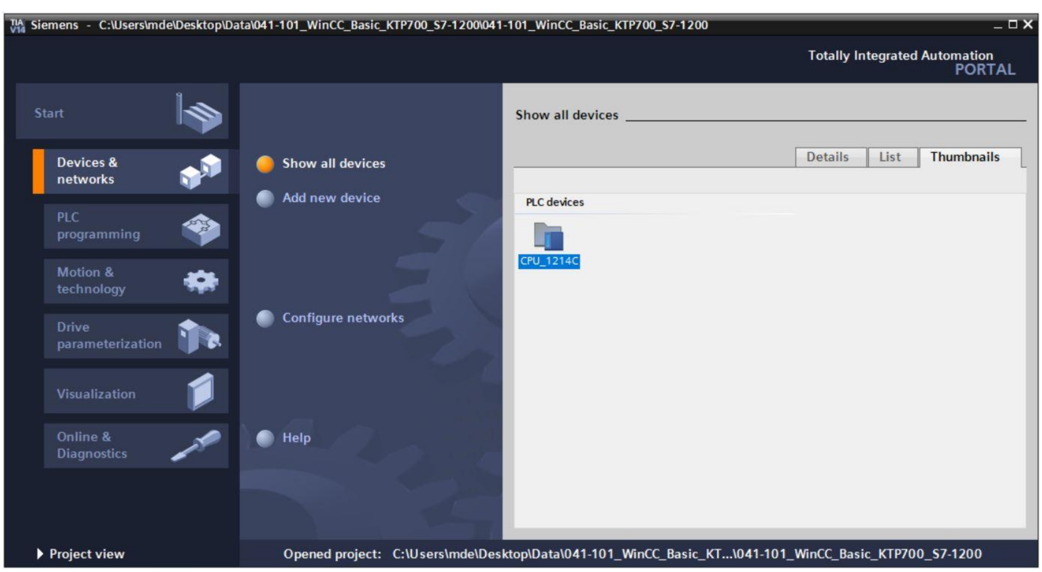
→ Next, the target directory in which the retrieved project is to be stored can be selected. Confirm your selection with "OK". (→ Target directory … → OK)

→ You save the opened project under the name 041-101_WinCC_Basic_KTP700_S7-1200. (→ Save → project as … → 041-101_WinCC_Basic_KTP700_S7-1200 → Save)



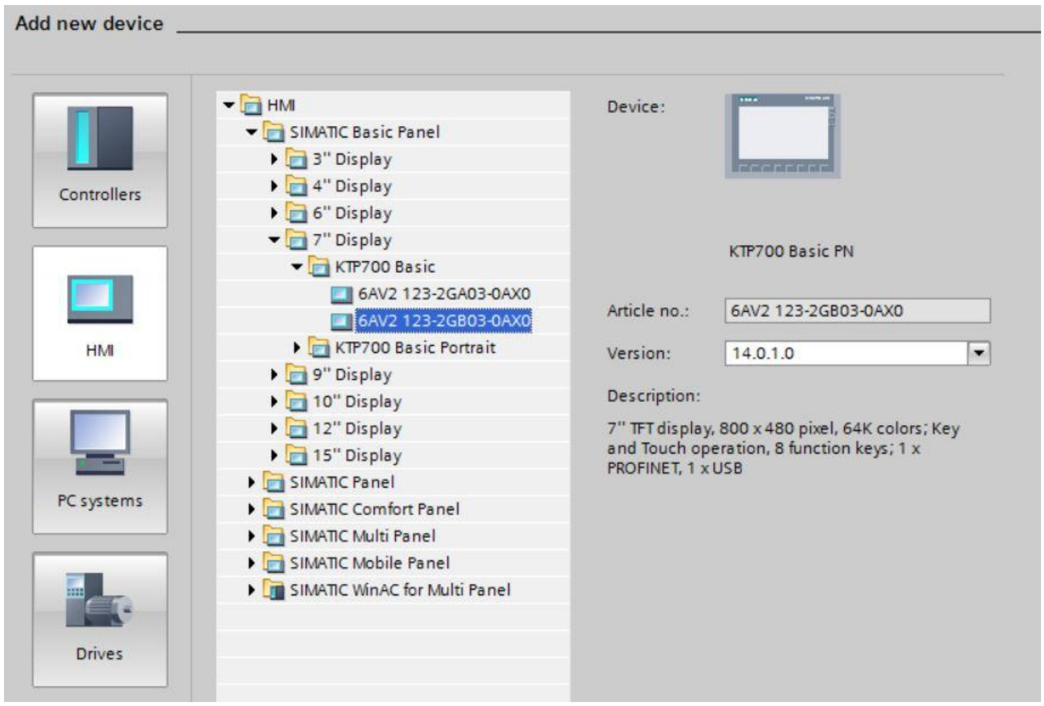
7.2 Adding SIMATIC HMI Panel KTP700 Basic

→ To create a new panel in the project, switch to the Portal view. Select menu item → "Devices & Networks" and → "Add new device" in the Portal.

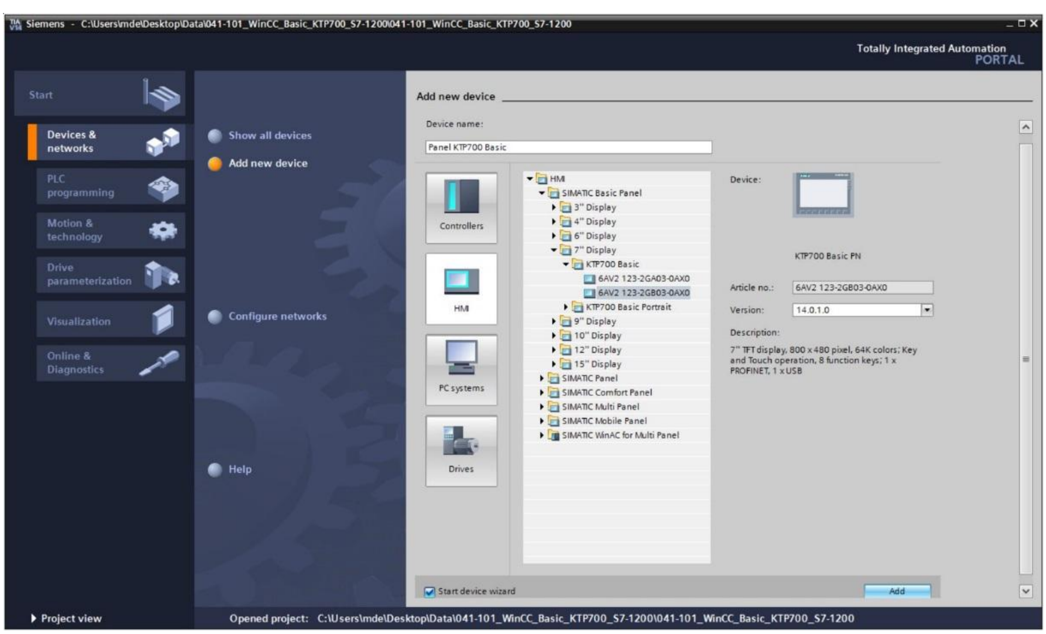


12

→ Next, select → "HMI" → "SIMATIC Basic Panel" → "7" Display" → "KTP700 Basic" as the device variant and the correct order number of your panel; here, e.g. → 6AV2 123-2GB03-0AX0.



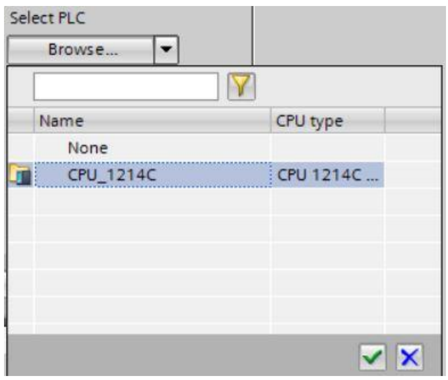
→ Enter the device name Panel KTP700 Basic and → select the "Start device wizard" check box. Click the Add button.



7.3 HMI device wizard for Panel KTP700 Basic

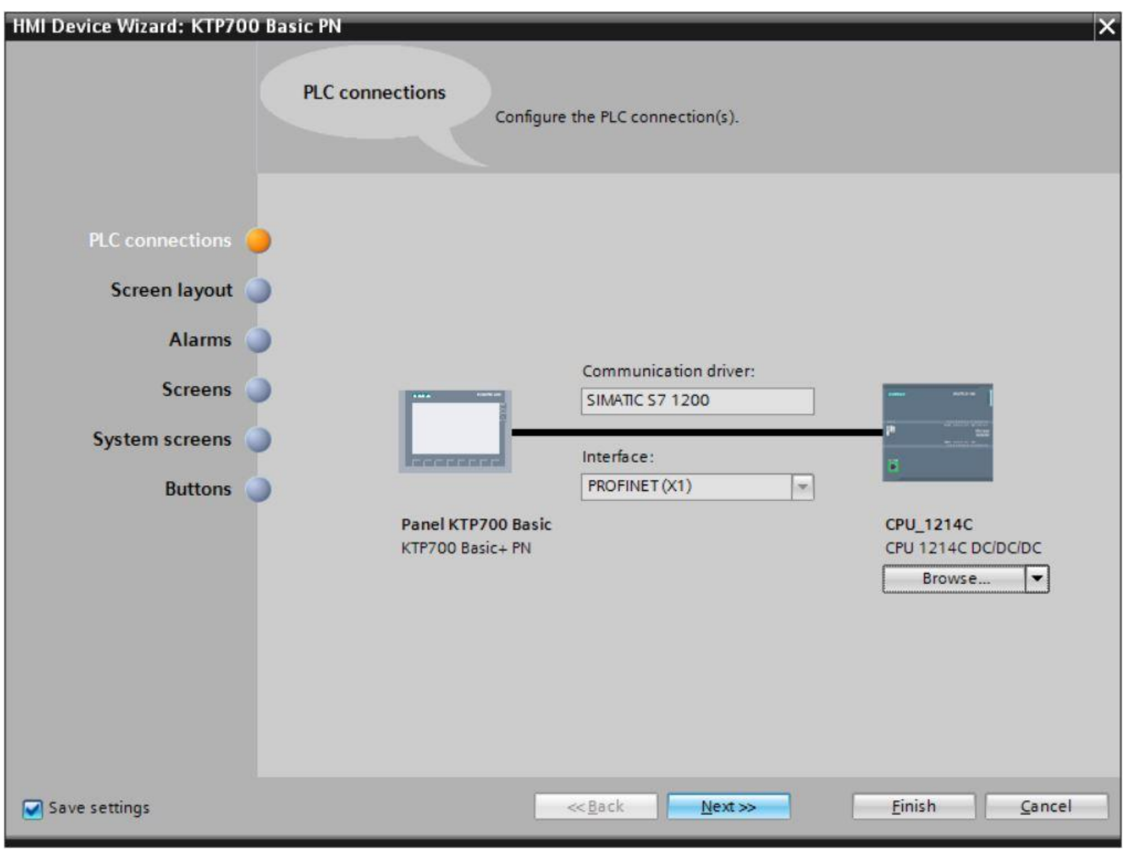
The TIA Portal creates the desired panel and automatically starts the HMI device wizard for Panel KTP700 Basic. This supports you in specifying some basic settings and functions for the panel.

→ You are first prompted for the PLC connections. Select your previously configured CPU 1214C as the communication partner.

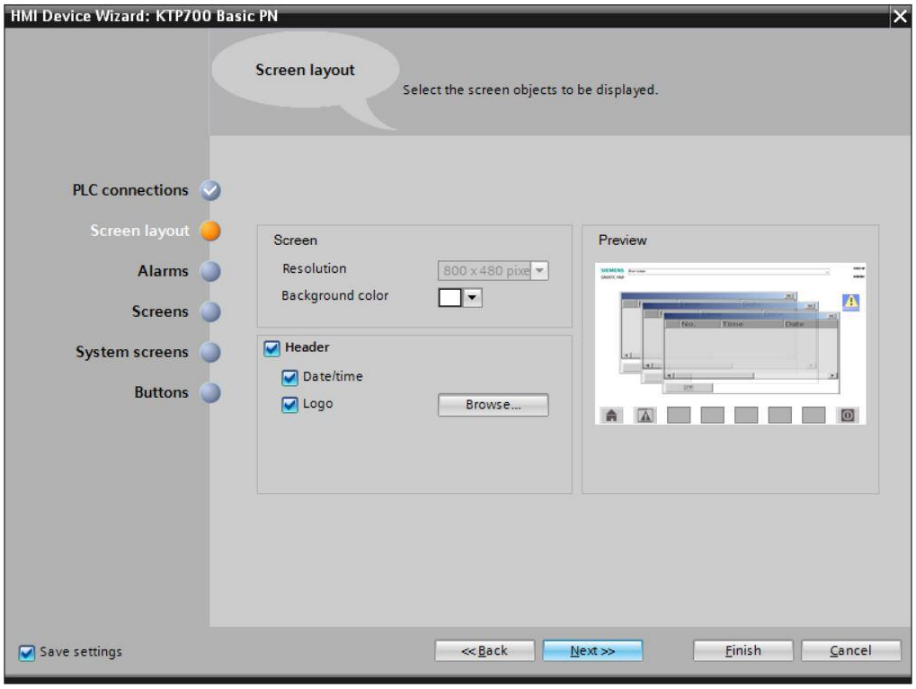


→ In order to connect your panel to the CPU, select the "PROFINET(X1)" interface.

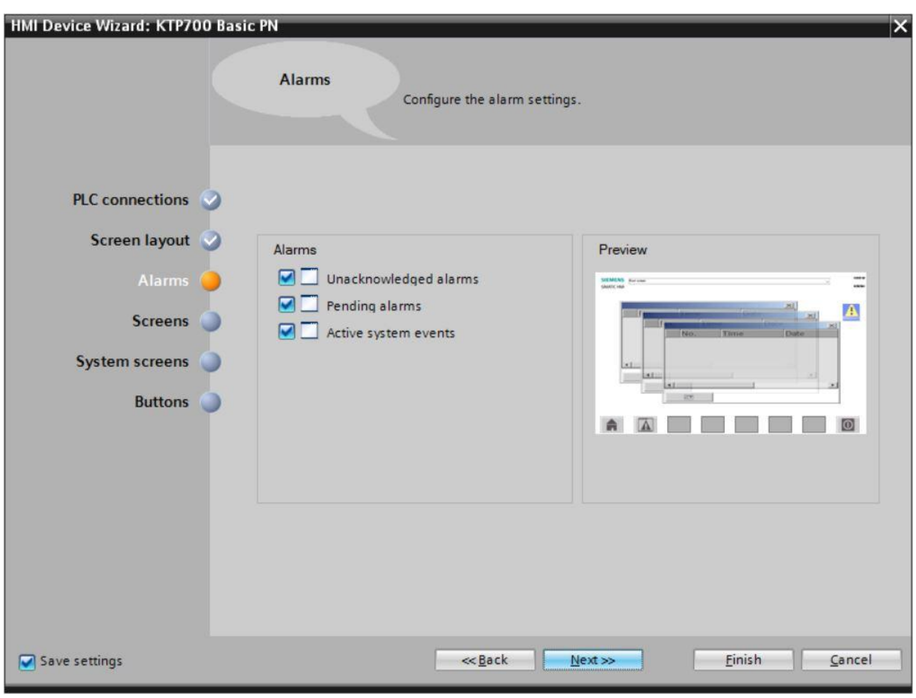
→ Confirm your selection by clicking on "Next >>".



→ You can change the default background color of your panel under "Screen layout". → Select the ☑ "Header", ☑ "Date/time" and ☑ "Logo" check boxes. → Confirm your selection by clicking on "Next >>".

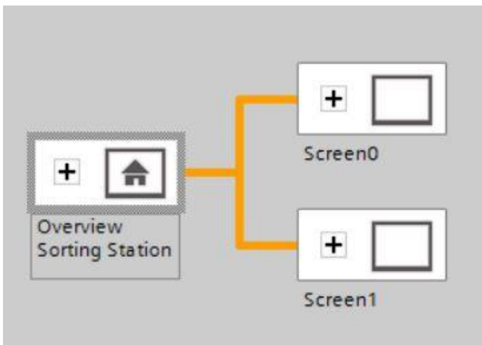


→ In the "Alarms" section, you can specify which of the alarms are to be displayed in a window. Select all three alarm types ☑ ☑ ☑ → Confirm your selection by clicking on "Next >>".

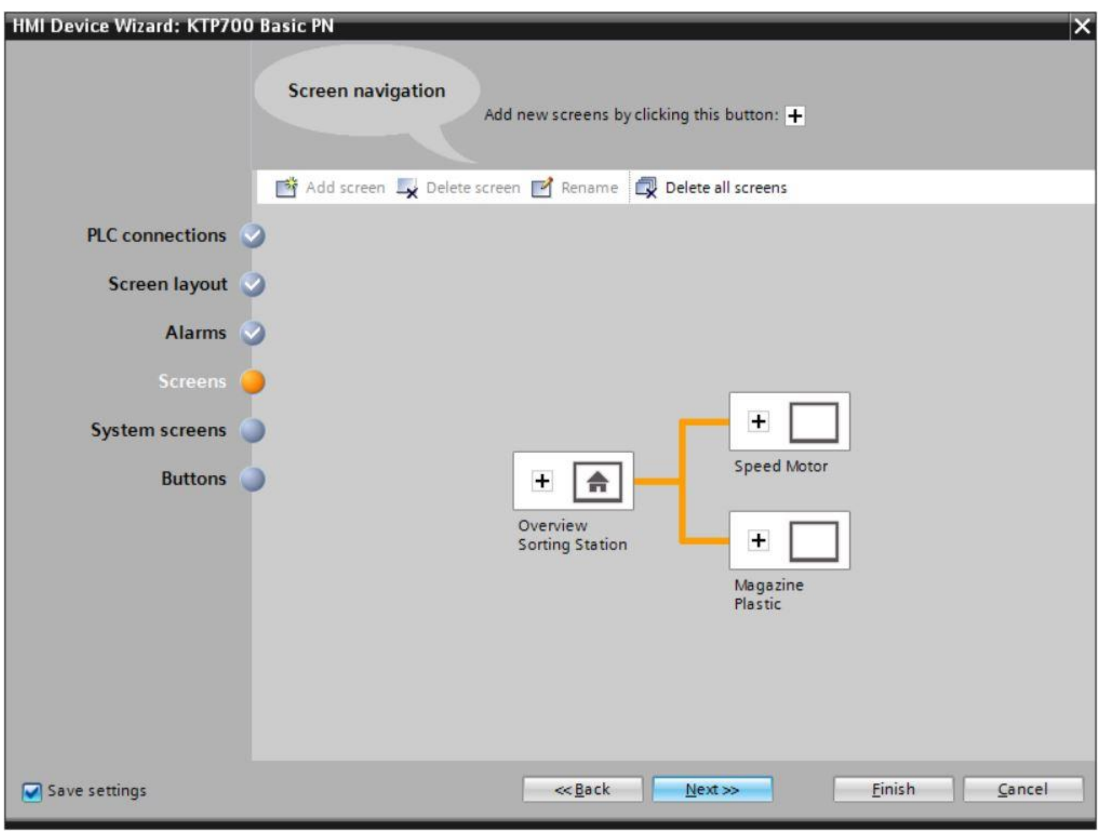


→ In the "Screen navigation" section, the screen structure is displayed with the screen name of the last created project, starting with the root screen on the far left.

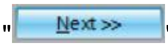
→ A new name can be assigned simply by clicking on a screen name. → If you click on + you can insert new screens in the hierarchy → and delete selected screens by clicking on " Delete screen ".

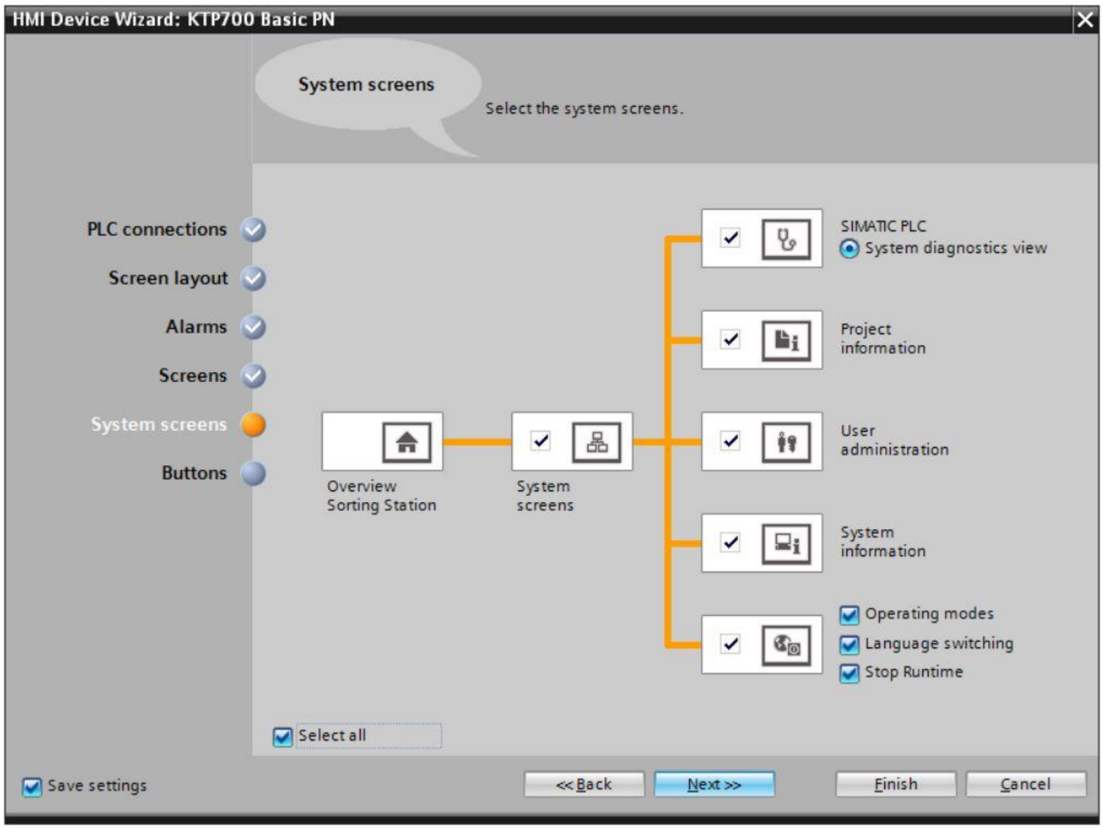


→ Use this approach to create the screen structure shown below with the corresponding screen names.→ Confirm your selection by clicking on " Next >> ".

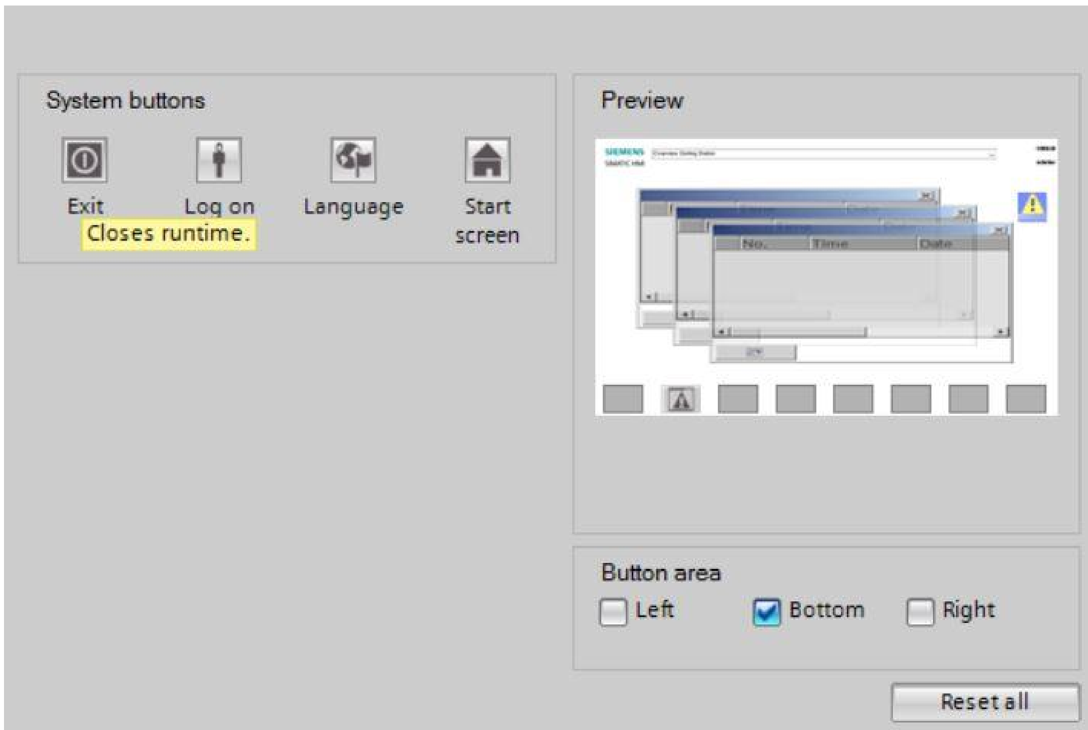


→ In the System screens section, you can select previously preset views for system functions and have them automatically added. → Select all system screens by clicking  "Select all".

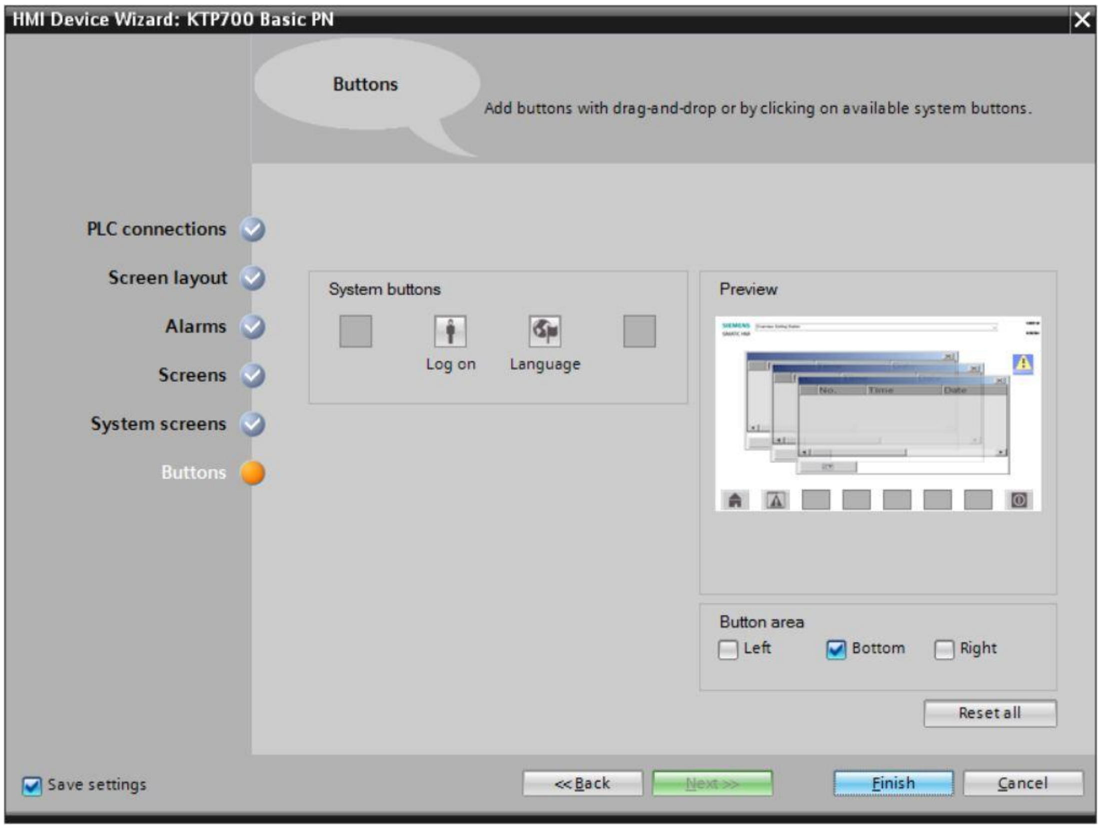
→ Confirm your selection by clicking on "".



→ In the System buttons section, you will find four user-selectable buttons for Exit (Runtime), Log on, Language and Root screen. You can place these buttons on the provided button areas "Left", "Bottom" or "Right" as desired. An "Open alarm window" button is already created.



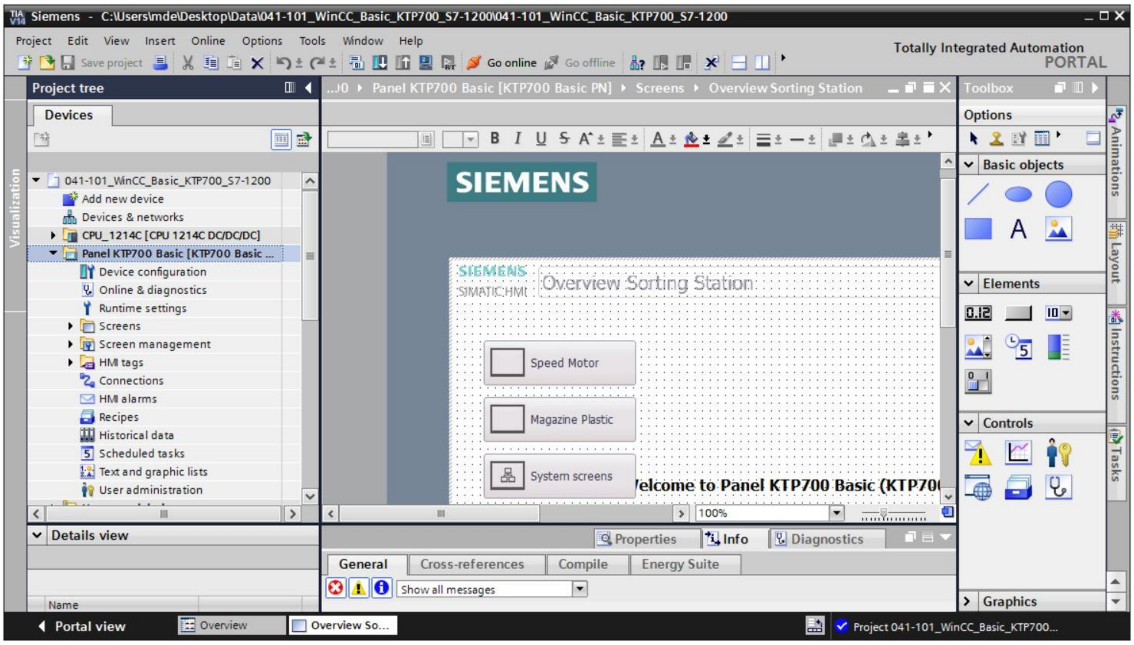
→ Select only the "Button area" "Bottom". → Insert the button for the "Root screen" on the left and the button for "Exit" Runtime on the right. → Confirm your selection by clicking on "Finish".



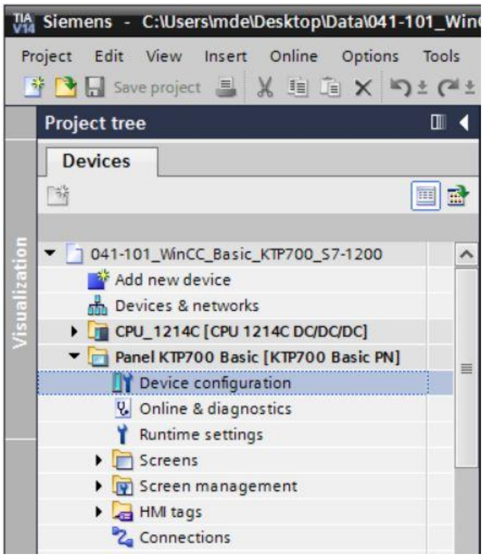
12

7.4 Device configuration of Panel KTP700 Basic

→ The TIA Portal now automatically changes to the Project view and displays the root screen of the visualization.



→ To configure the panel, select "Panel KTP700 Basic" in the project tree and open its "Device configuration" with a double-click.



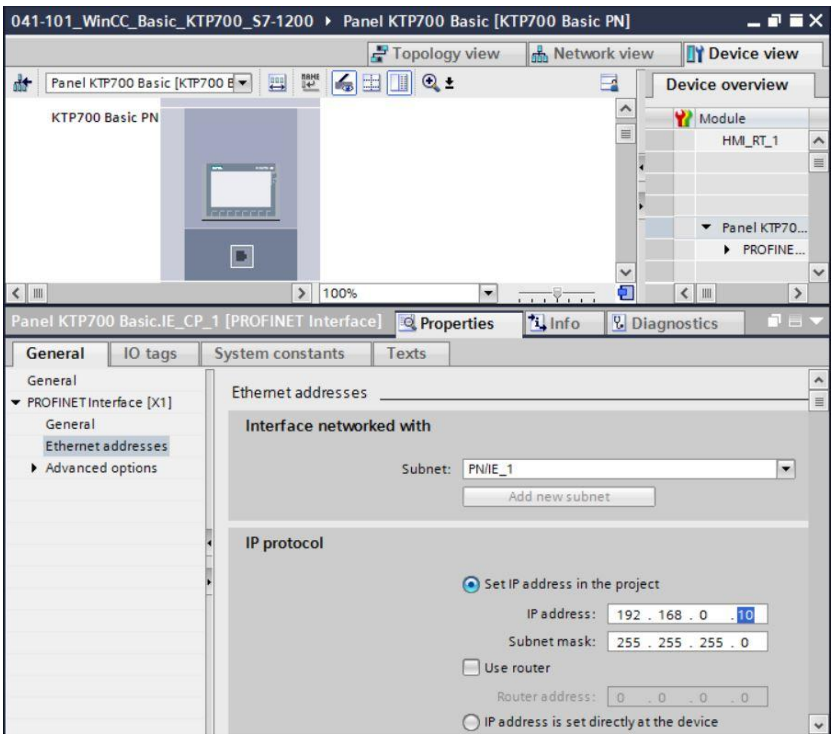
12

7.4.1 Setting the IP address

→ Select the Ethernet interface of the panel in the Device view with a double-click.

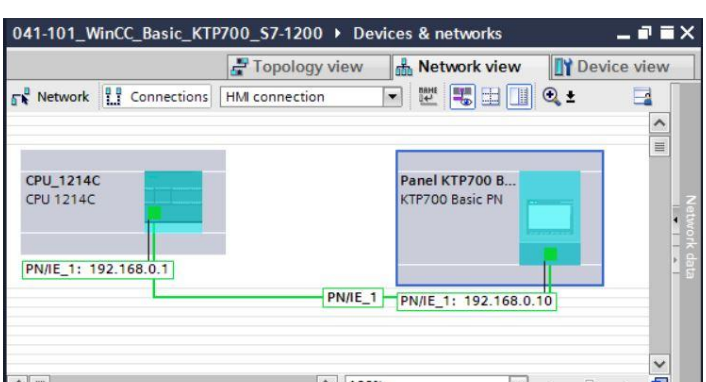
→ Under "General" in → "Properties", open menu item → "PROFINET interface [X1]" and select in the → "Ethernet addresses" entry.

→ Set the IP address "192.168.0.10" under IP protocol.



Note: The subnet mask was already set in the settings of the CPU 1214C and is automatically applied by the panel.

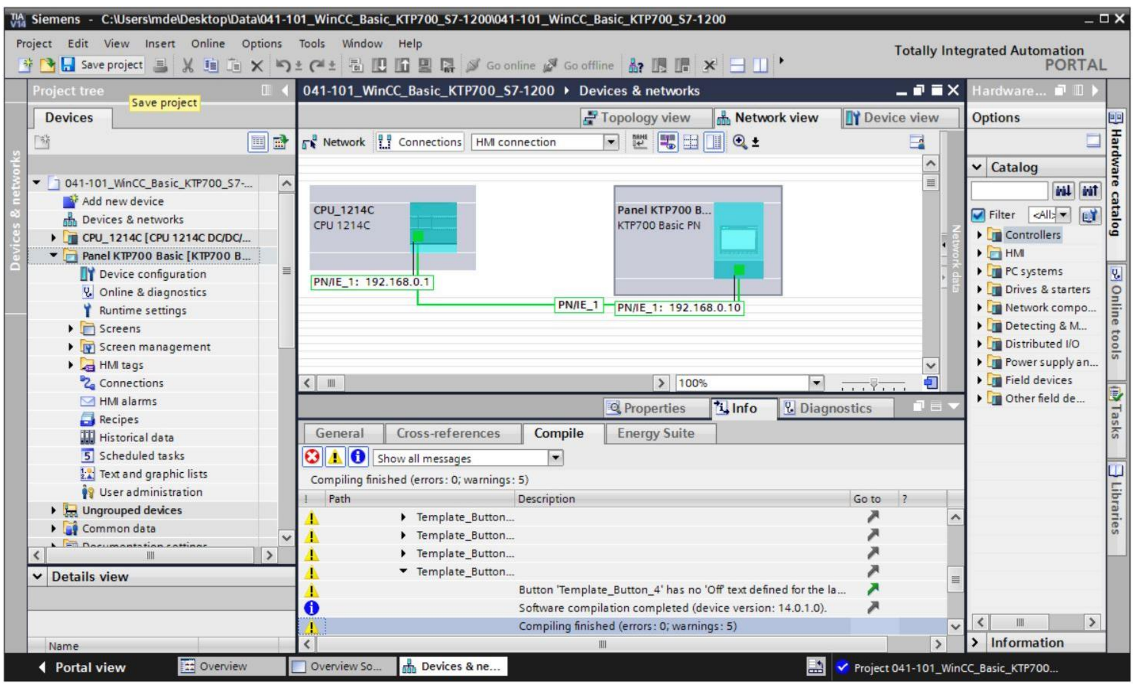
→ To obtain an overview of the assigned addresses within a project, you can click the → "" button in the → "Network view". If you click on → Connections here, the "HMI connection" between the CPU and panel that was created previously in the wizard is displayed.



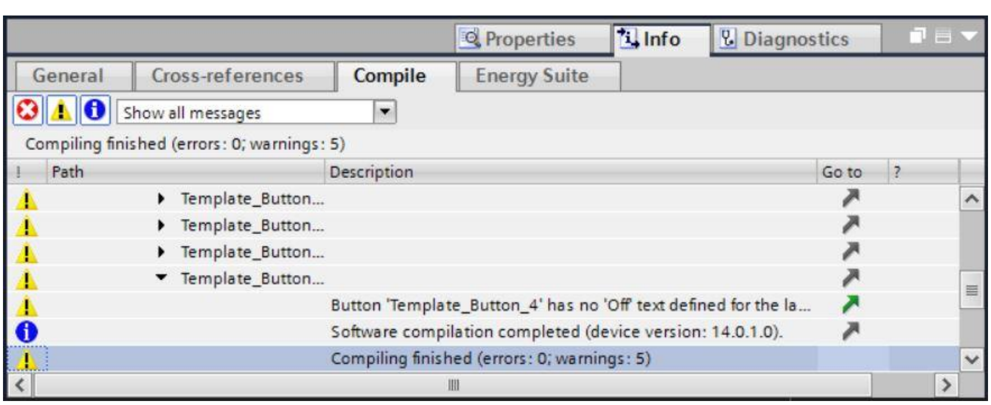
7.5 Compiling the CPU and panel and saving the project

→ To compile the CPU, click on the "CPU_1214C" folder, and select the button for compiling in the menu. To compile the panel, click on the "Panel KTP700 Basic" folder, and select the button for compiling in the menu. You can save your project by clicking on the Save project button in the menu.

(→ CPU_1214C → → Panel KTP700 Basic → → Save project).

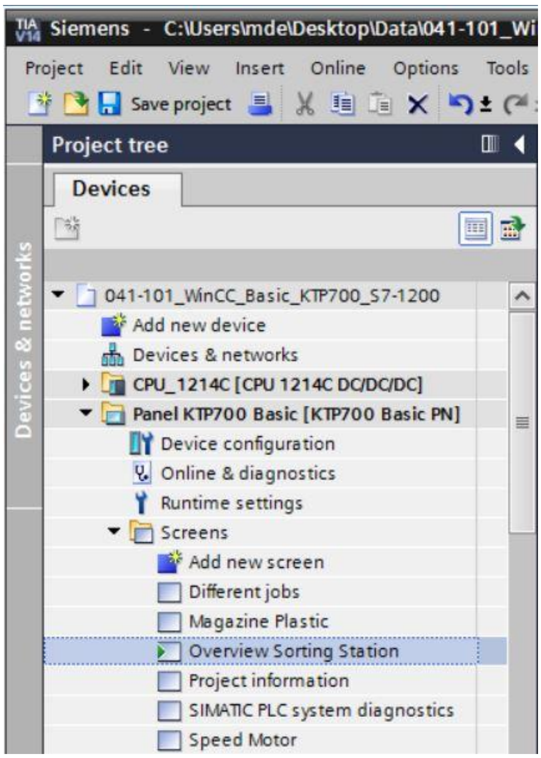


→ In the "Info" area under "Compile", it is then shown whether the compilation was successful or whether warnings or errors occurred.



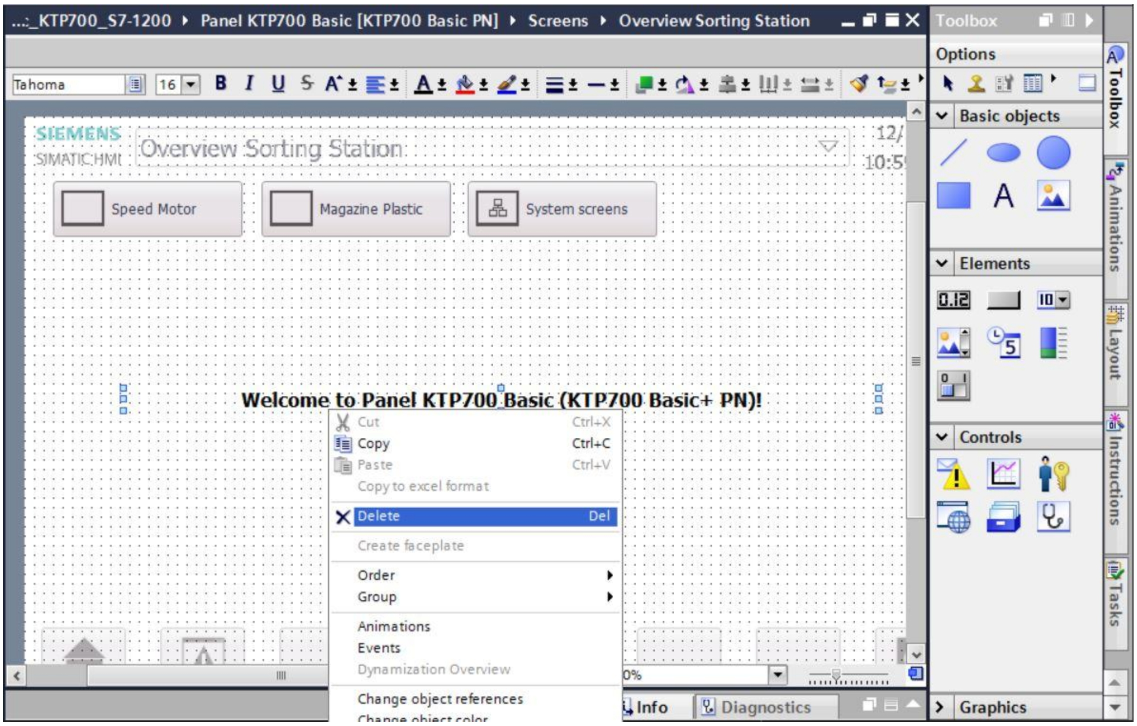
7.6 Configuring the Graphic view

→ After successful compilation, you want to design the first screen for the visualization. To do this, open the → "Overview Sorting Station" screen with a double-click:



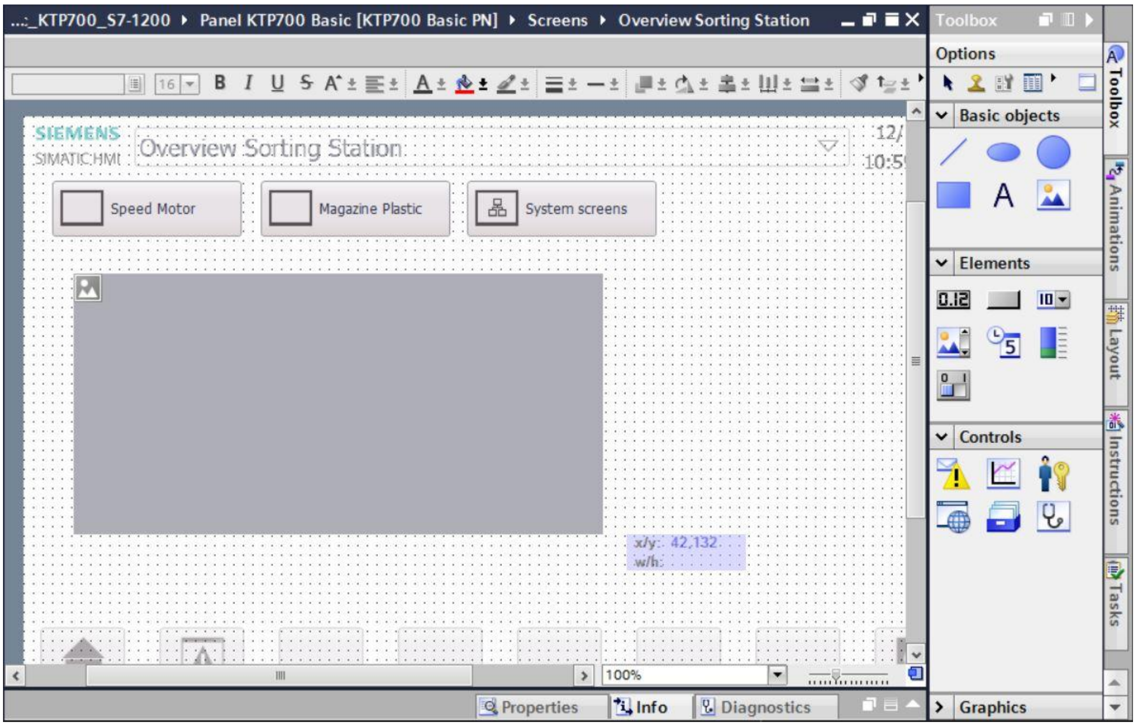
12

→ Several objects, such as the screen change buttons, have already been created by the wizard. The text box in the center of the screen is to be removed by right-clicking on it and selecting → "Delete" in the displayed dialog.

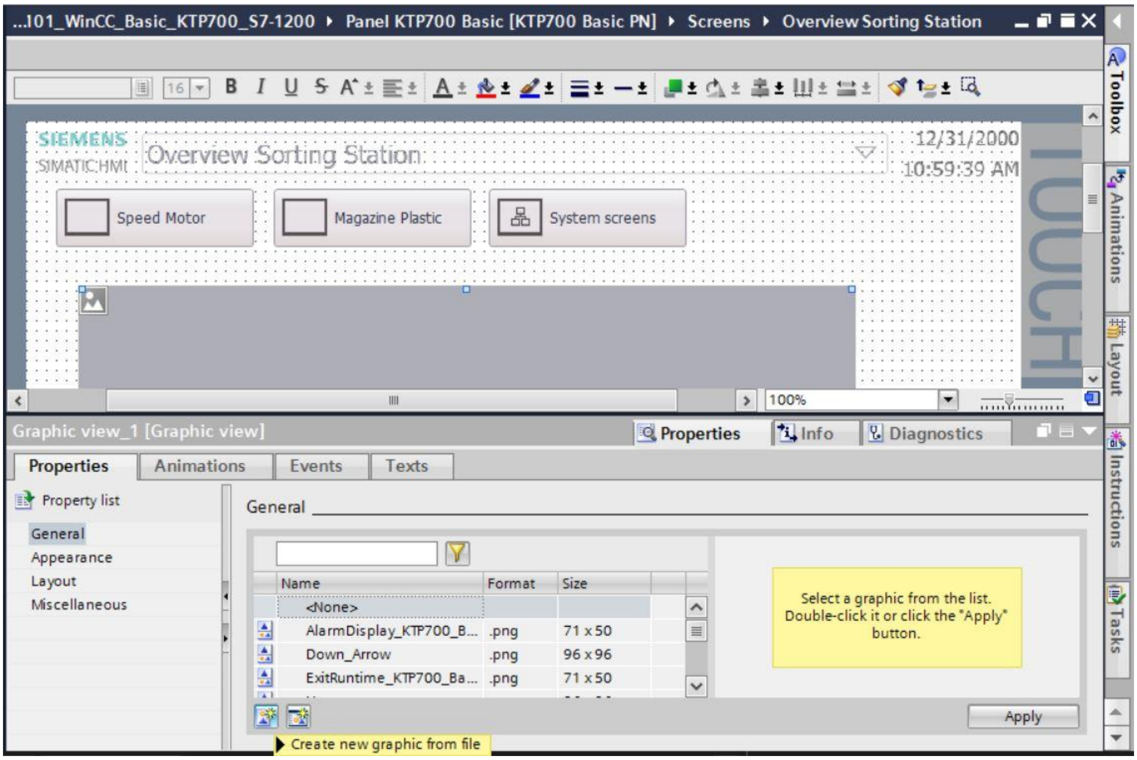


12

→ Select → "Graphic view"  from → "Basic objects" in Toolbox. The mouse pointer changes in such a way that it can now be used to draw an area for the display of a graphic in the work window.



→ You can double-click on the area of the Graphic view to have its properties displayed. In sub-item → "General", select → the symbol for → "Create new graphic from file".



Note: There are four sub-items for the properties of objects.

– *Properties for static settings of the object*

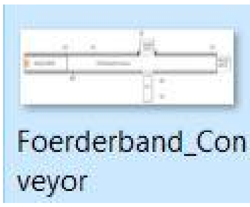
– *Animation for dynamic settings of the object*

– *Events when actions are to be triggered from an object*

– *Texts for multilingual display within an object*

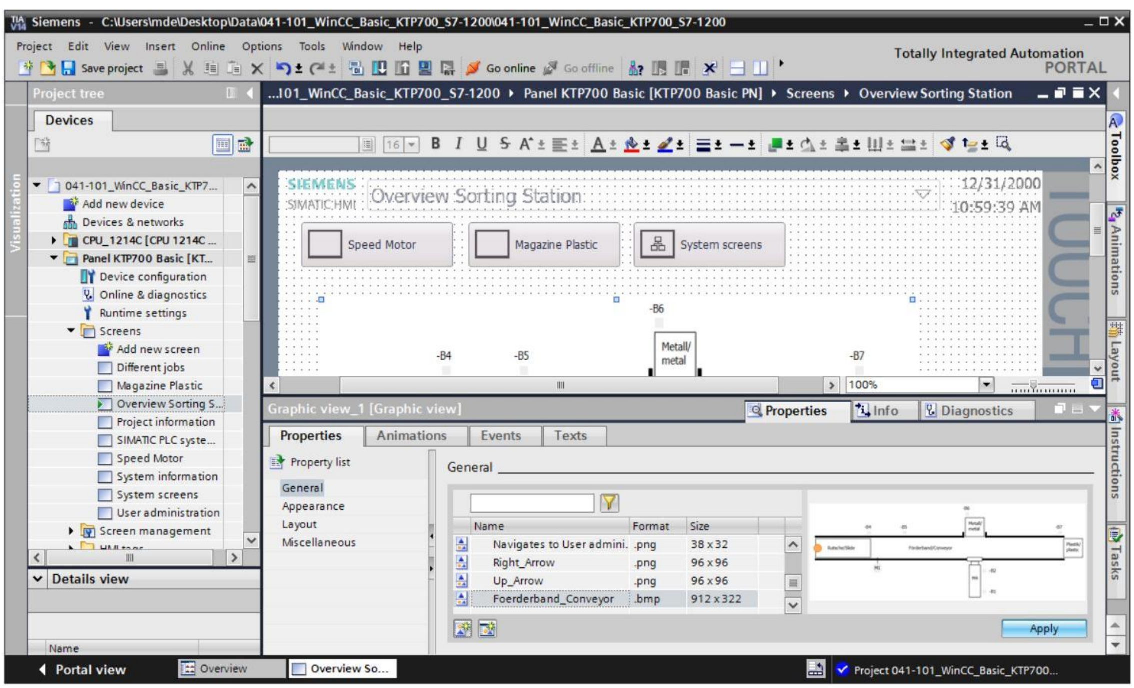
12

→ In the displayed dialog, select the "Foerderband_Conveyor.bmp" file from the "SCE_EN_041-101_Screens" folder and click → "Open".



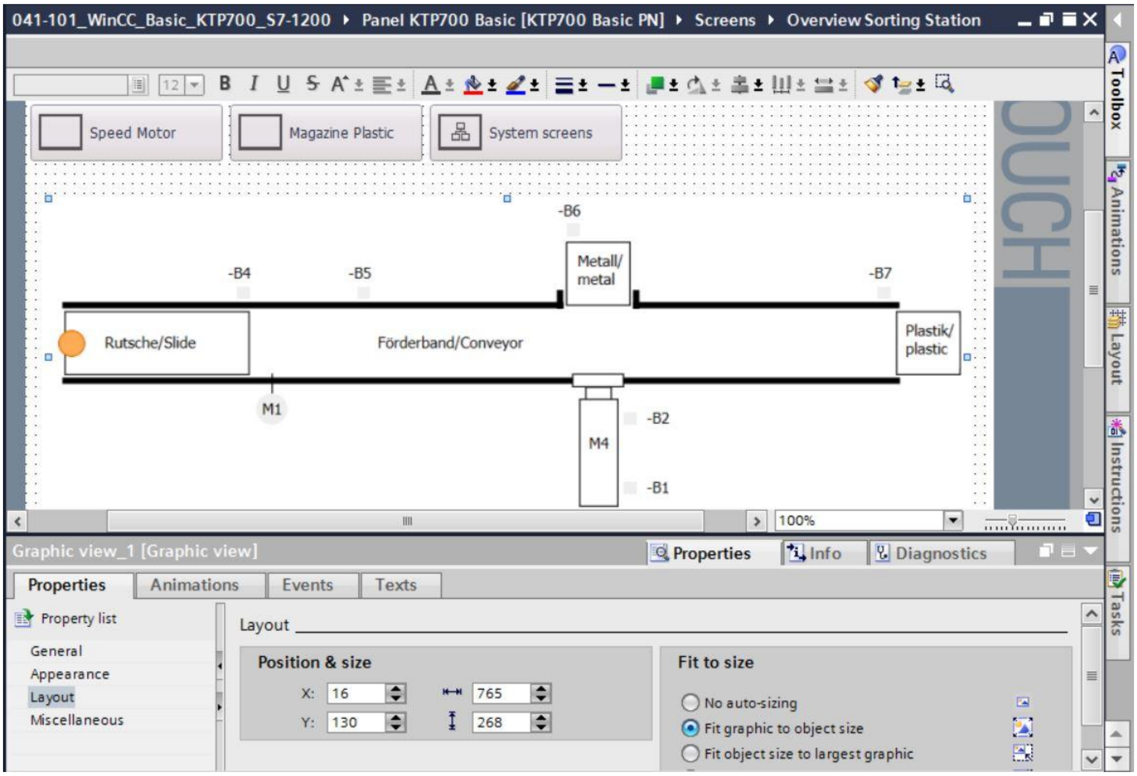
Note: You can draw the graphics used in this document yourself and save them in *.bmp format or download them from the Internet at www.siemens.com/sce/S7-1200 in module "SCE_EN_041-101 WinCC Basic with KTP700 and S7-1200" under "SCE_EN_041-101_Screens".

→ For the display, select the "Foerderband_Conveyor.bmp" graphic and click → "Apply".



Note: The created graphic is stored in the project in the "Languages & resources" path under "Graphic collection".

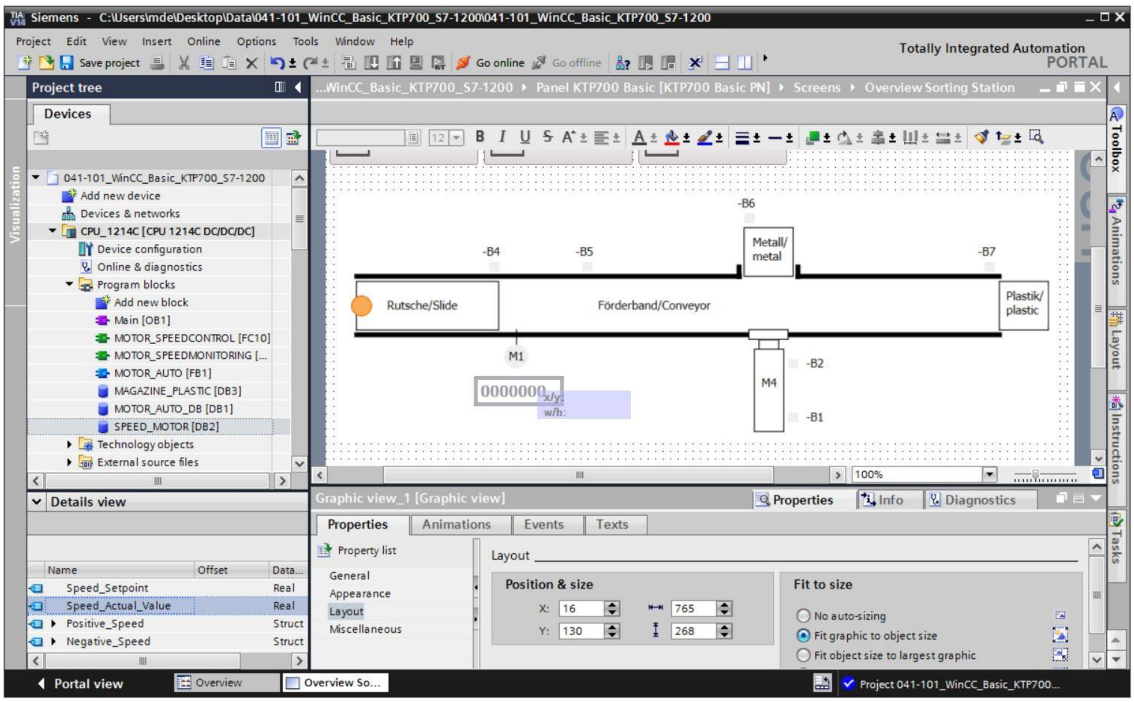
→ Use the mouse to position the graphic in such a way that the positions and sizes indicated below are entered → under "Layout" in → Properties. The → "Fit graphic to object size" option can be used for adjusting the size.



12

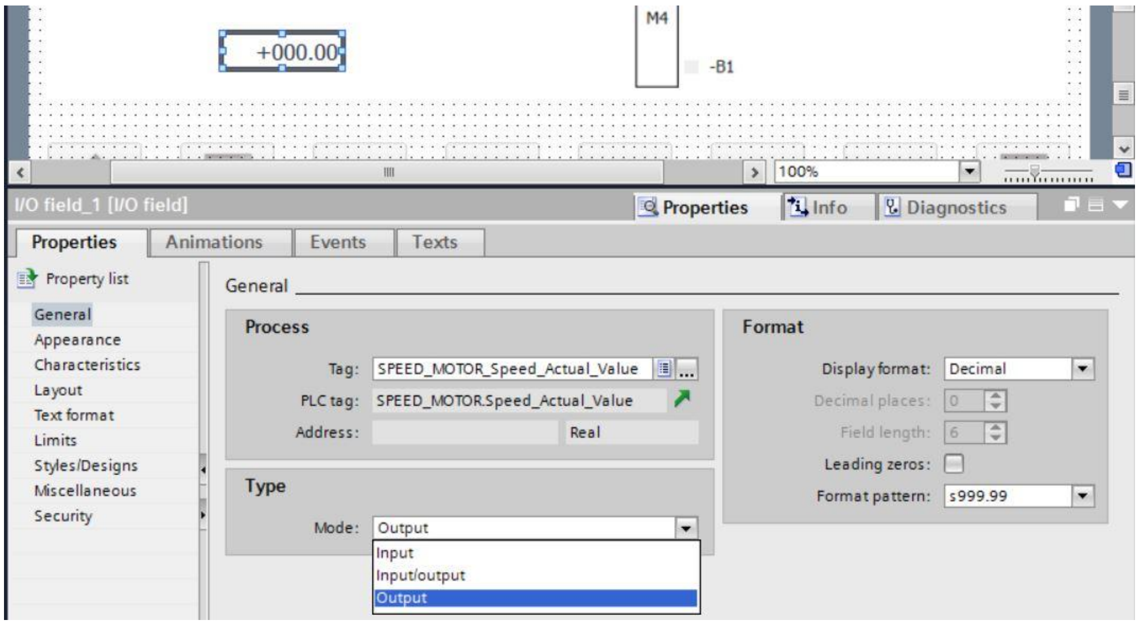
7.7 Displaying a process value in an IO field

→ First, insert a display of the actual value of the current speed below the conveyor motor. For this, select → "Program blocks" and the → "SPEED_MOTOR[DB2]" data block of → "CPU_1214C". Next, move the → "Speed_Actual_Value" tag from the → Details view to the "Overview Sorting Station" screen using drag & drop.



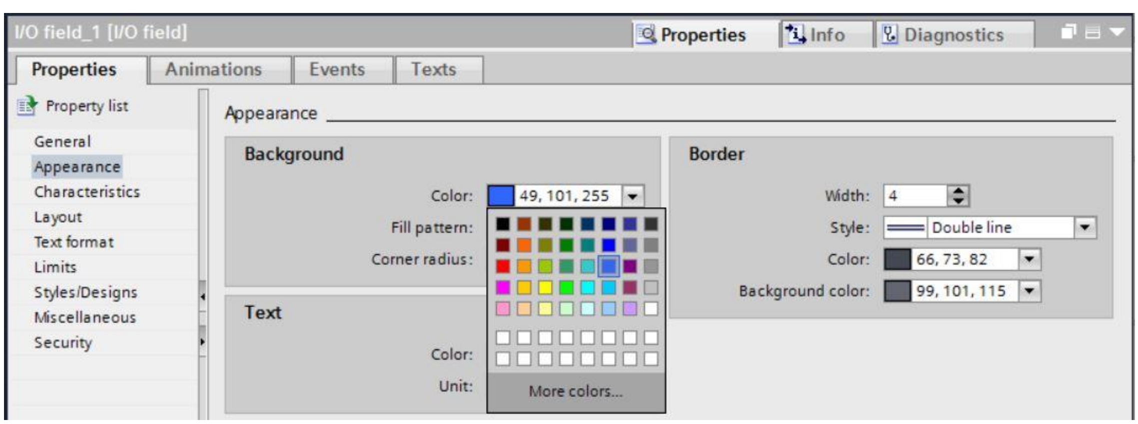
12

→ The connection to the tag in the PLC is already created in "Properties" of the IO field under "General" and "Process". The "Display format" is correctly set to "Decimal". Only the "Format pattern" and the "Type" of the field are changed here to → "s999.99" and → "Output", respectively.

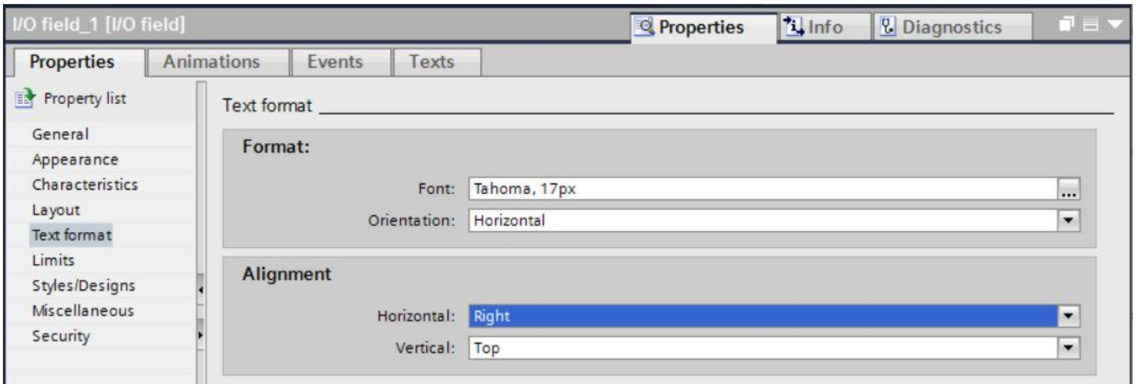


Note: The format pattern s999.99 means that the IO field is displayed with three places before decimal, two places after the decimal and a sign.

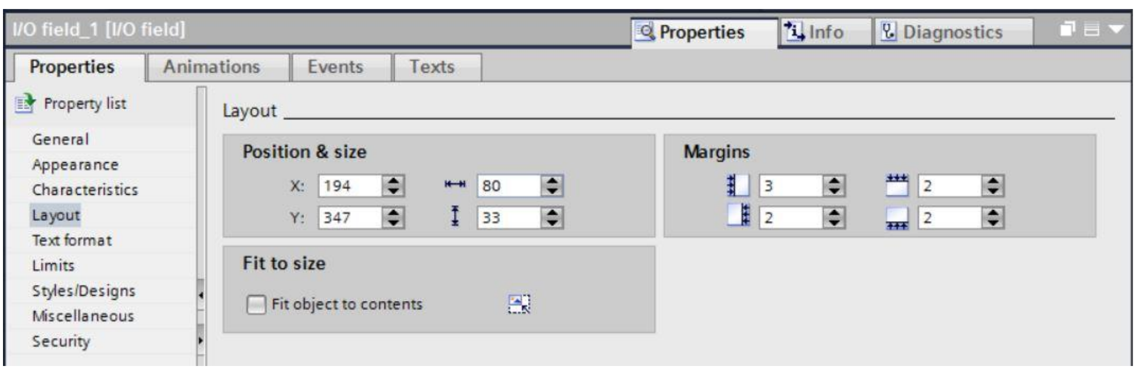
→ The "Color" of "Background" is changed to → "Blue" under "Appearance" in "Properties".



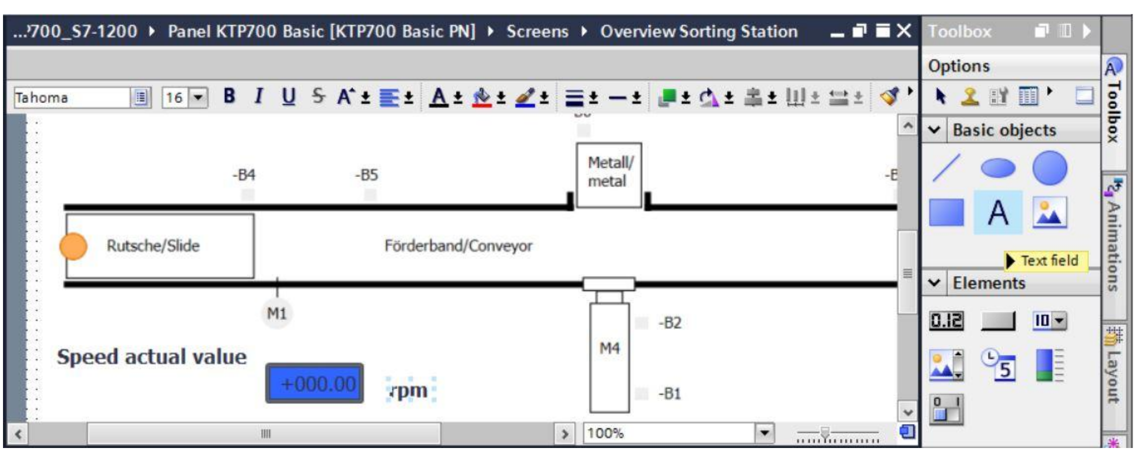
→ Under "Text format" in "Properties", change "Alignment" "Horizontal" to → "Right".

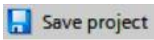


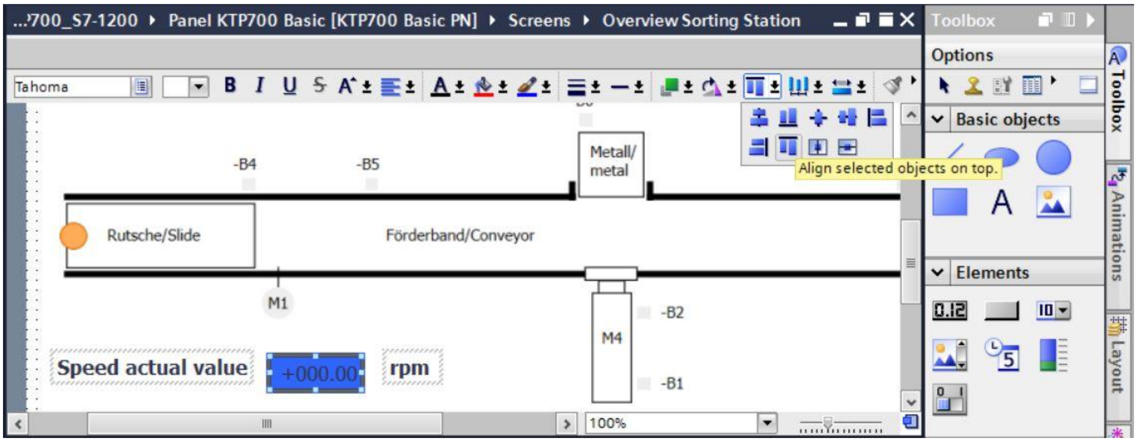
→ Under "Layout" in "Properties", you change → "Position & size" as illustrated here so that the IO field is displayed below the conveyor motor.



→ For the description, you insert a → "Text field"  from → "Basic objects" of Toolbox after the IO field using drag & drop. Type in the texts → "Speed actual value" and → "rpm".

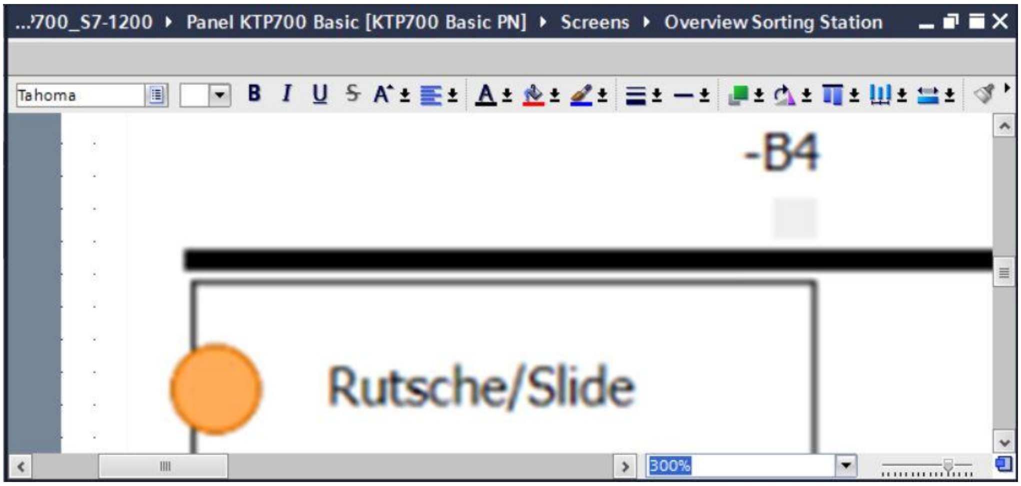


→ Select the three objects → IO field → text field "Speed actual value" → text field "rpm" in this order and click on the → "Align selected objects on top"  function in the toolbar of the work area. Save your project by clicking on .



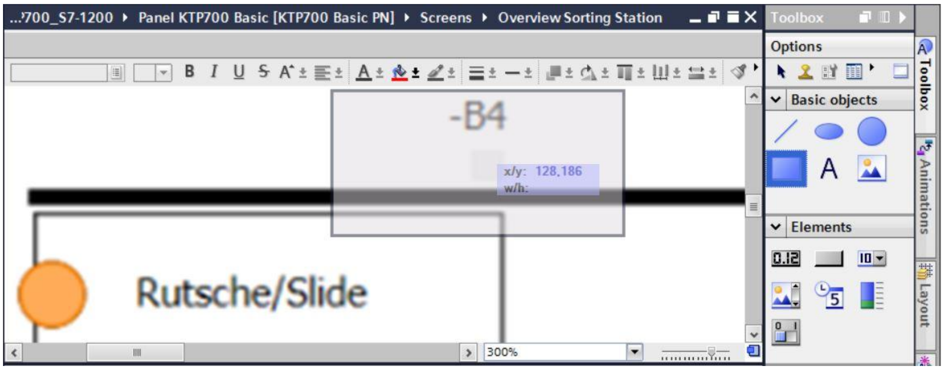
7.8 Visualizing binary signals with animated rectangles/lines

→ For visualization of the sensors, start with sensor "-B4" at the slide. To enable you to better draw and position the rectangle, change the zoom factor to → "300%".

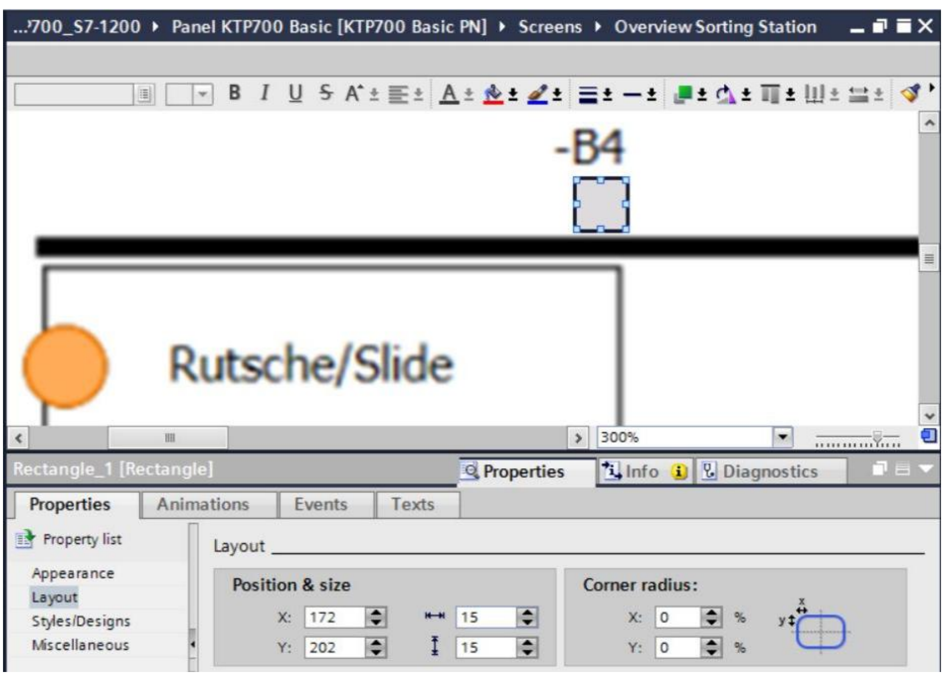


12

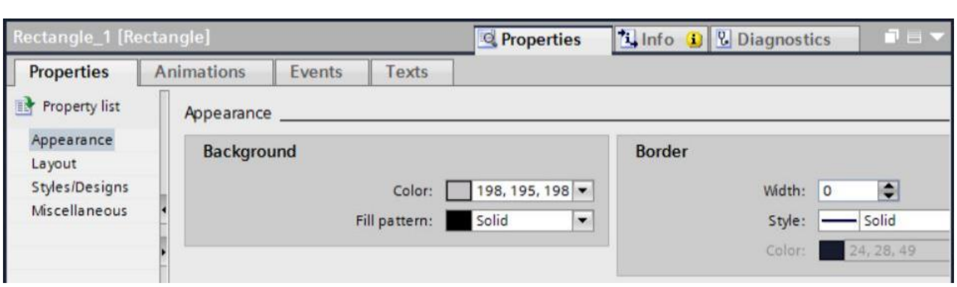
→ Then, use drag & drop to move a "Rectangle"  from → "Basic objects" of Toolbox to the position of sensor "-B4".



→ Next, use the mouse to change the rectangle to the appropriate position and size or set the → "Position & size" under "Layout" in "Properties" as shown here. As a result the sensor is displayed below the "-B4" label.

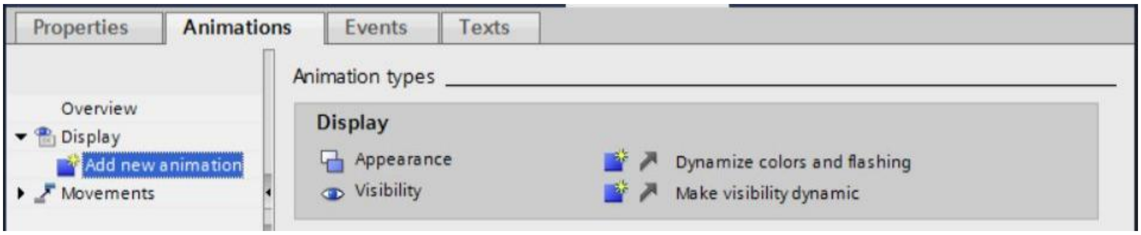


→ Under "Appearance" in "Properties", change the "Color" of "Background" to → "Gray" and the "Width" of "Border" to → "0".

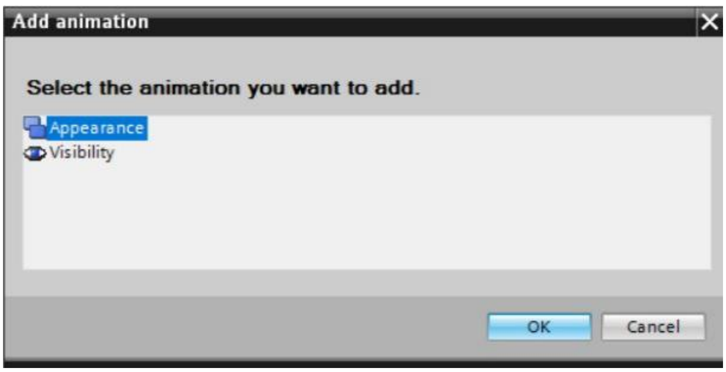


12

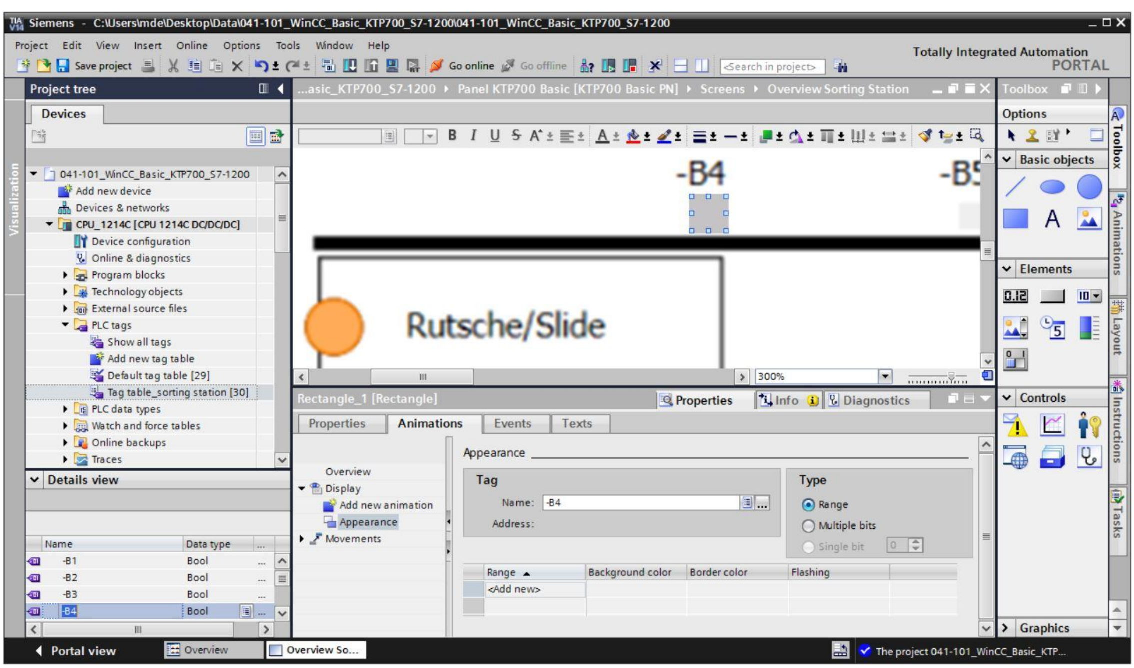
→ Now switch to the "Animations" tab, select "Display" and click → "Add new animation".



→ In the displayed dialog, select → "Appearance" and click → "OK".

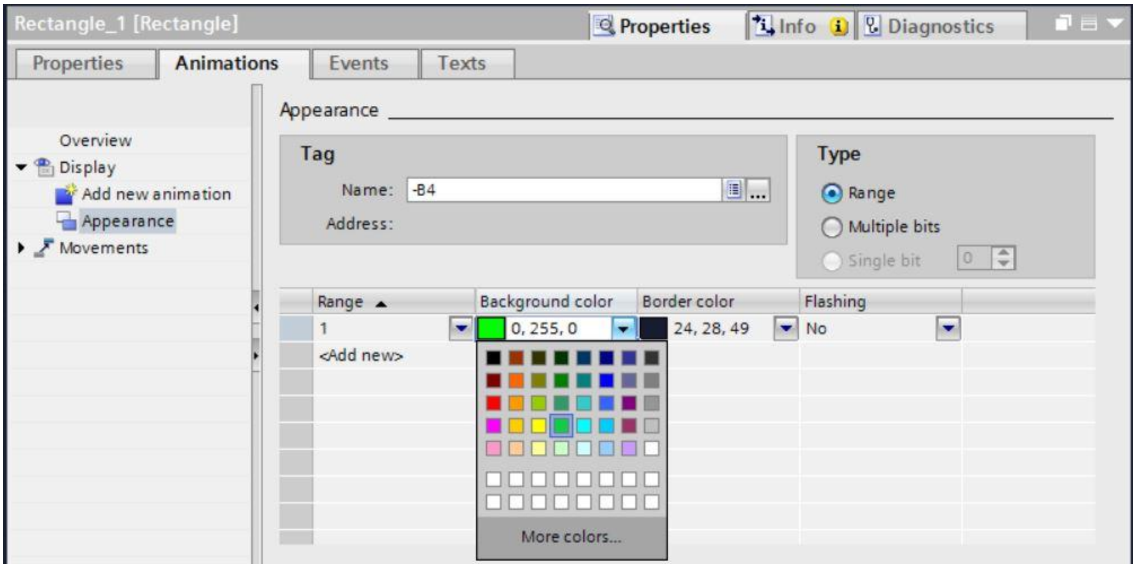


→ To establish the connection to the global tag in the CPU, select → "PLC tags" and → "Tag table_sorting station" below → "CPU_1214C". Next, move the → "-B4" tag from the Details view to the "Name" field for the tag using drag & drop.



12

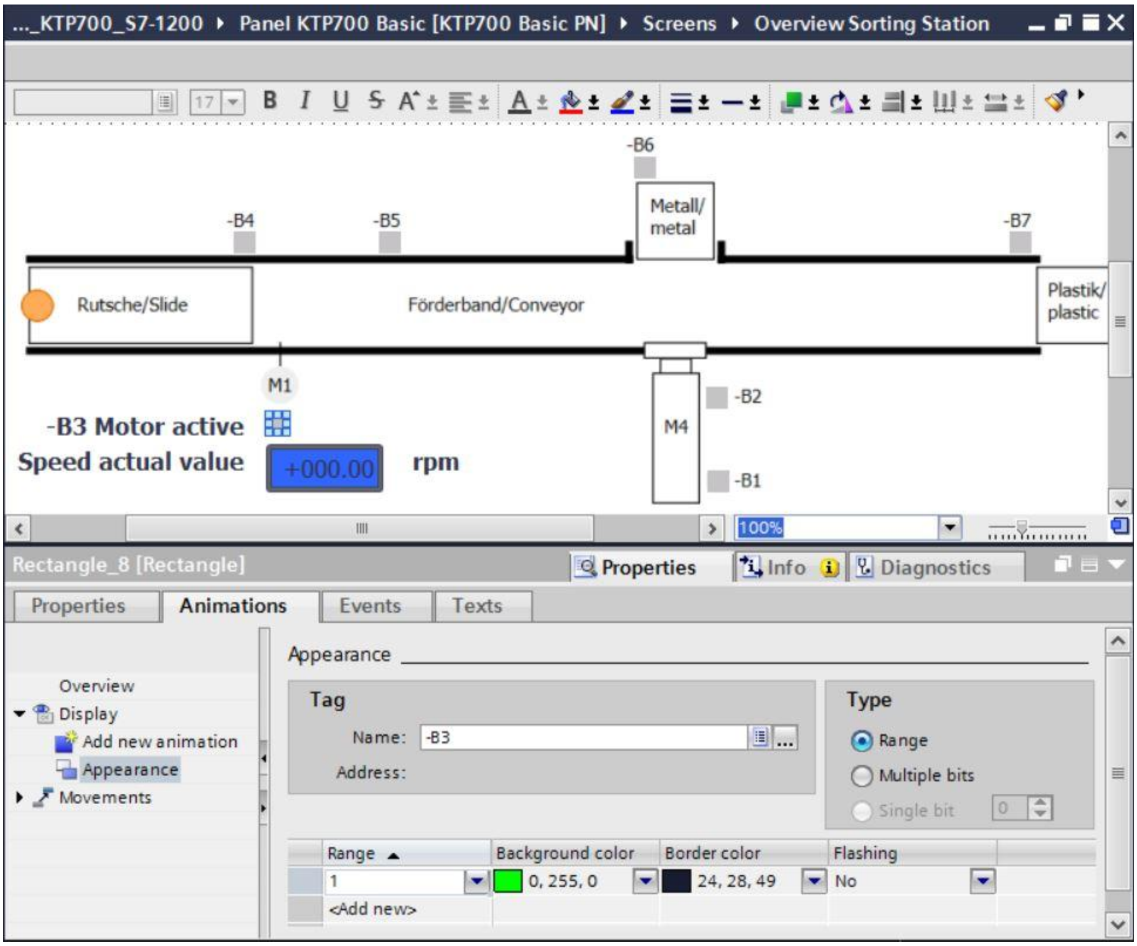
→ Under "Appearance" in "Display", add a range with value → "1" (signal state "High") and change the "Background color" to → "Green".



12

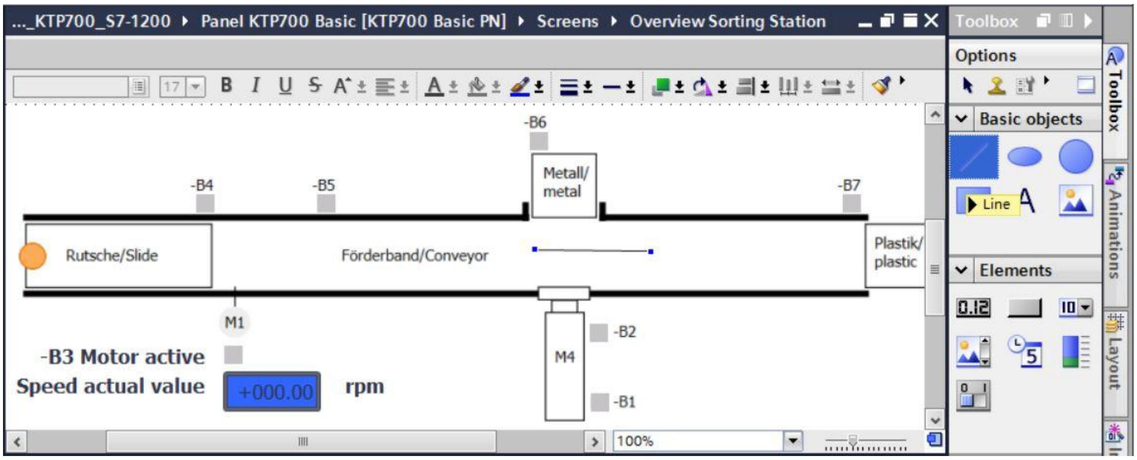
→ Next, use the steps shown previously to create a display for sensors → "-B1", → "-B2", → "-B5", → "-B6" and → "-B7".

→ Insert an additional binary display below motor M1 and connect it to the global tag → "-B3". For the description, insert a text field → "-B3 Motor active" in front.

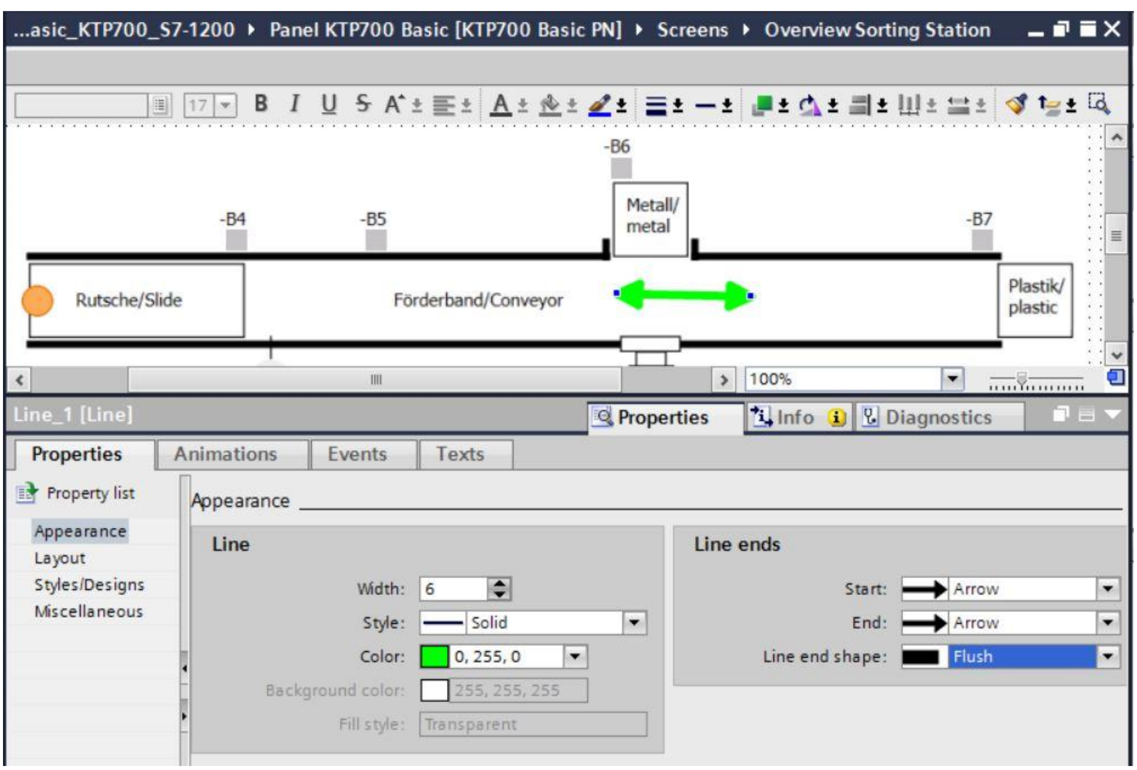


12

→ In order to display that the conveyor is being controlled, drag the "Line" object from → "Basic objects" of Toolbox onto the conveyor.

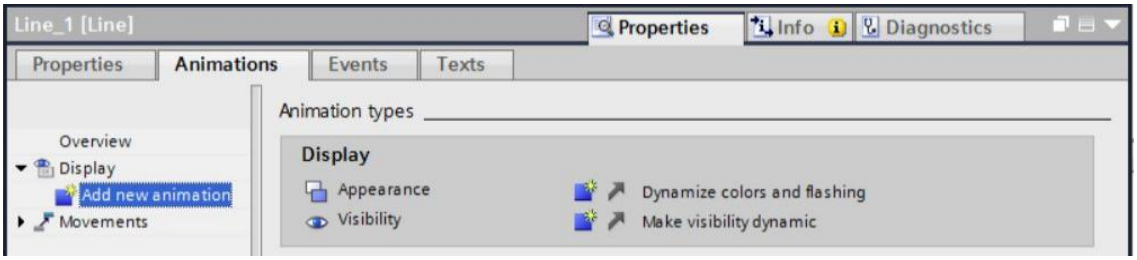


→ Under "Appearance" in "Properties", change "Style" of the line to → "Solid" and "Color" of "Foreground" to → "Green". Change the "Line ends" at "Start" and "End" to "Arrow".

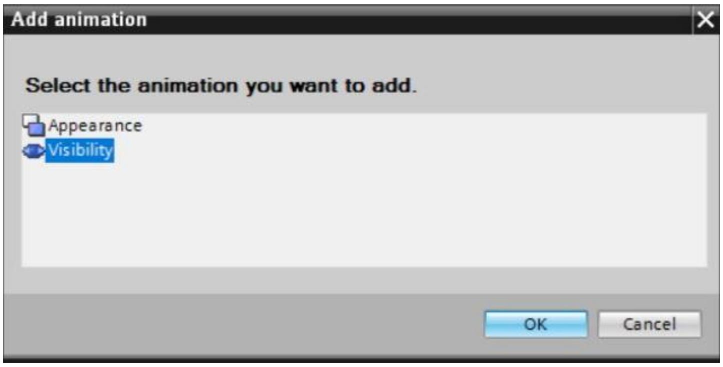


12

→ Now switch to the "Animations" tab, select "Display" and click → "Add new animation".

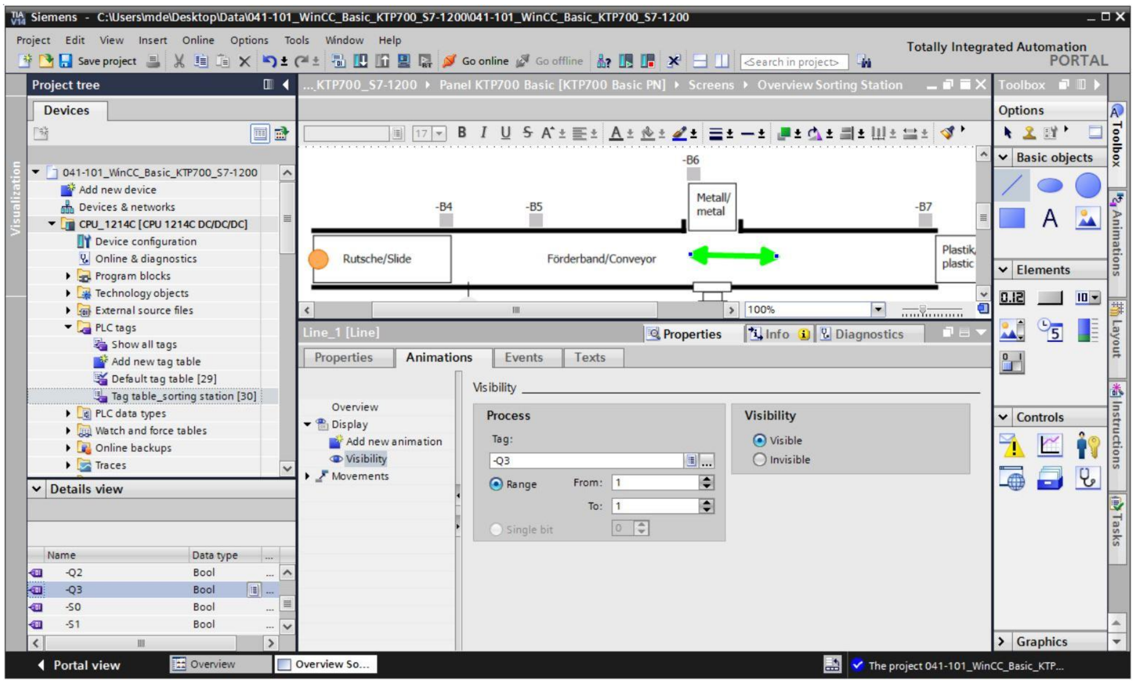


→ In the displayed dialog, select → "Visibility" and click → "OK".

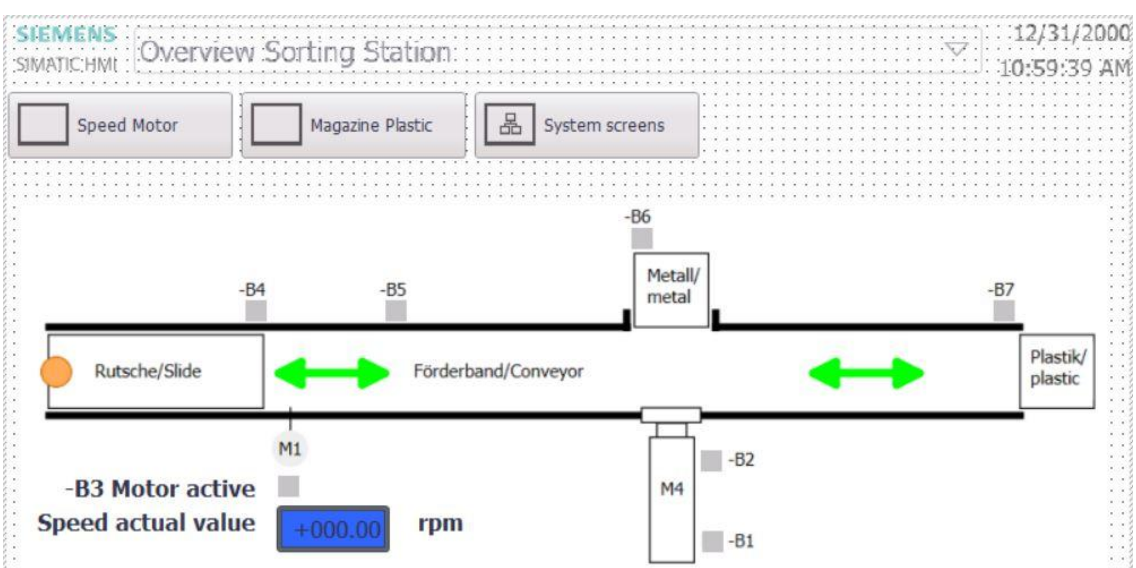


12

→ To establish the connection to the global tag in the CPU, select → "PLC tags" and → "Tag table_sorting station" below → "CPU_1214C". Next, drag the → "-Q3" tag from the Details view to the "Tag" field. In addition for the type of evaluation, select → "Range", enter "From" → 1 "To" → 1 and select → "Visible" for "Visibility".

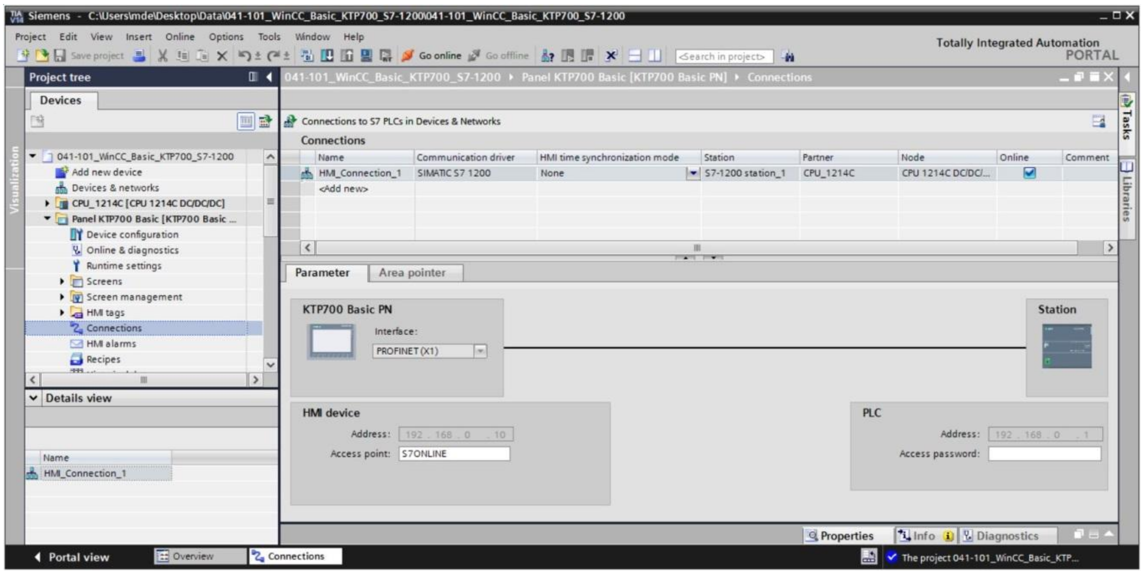


→ Copy the arrow from the symbol library with all its properties using → "Copy" and "Paste".



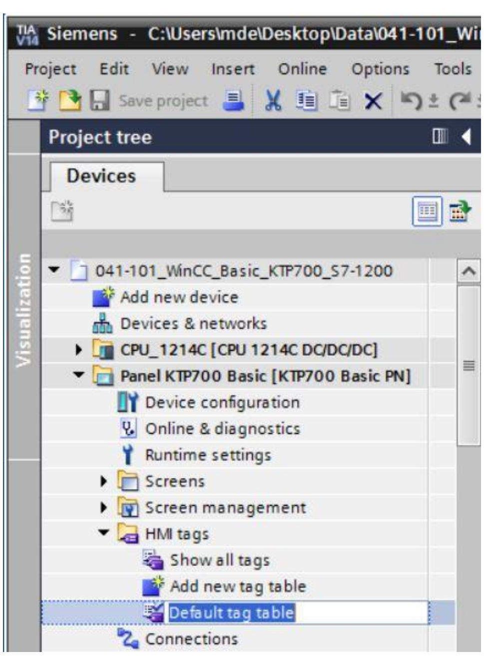
7.9 Connections and HMI tags

→ Before you download the configuration to the Panel KTP700 Basic, you should check the connection to the CPU 1214C. To do this, double-click → "Connections" in → "Panel KTP700 Basic". In the displayed view, you can check the IP addresses and connection settings again. It is also important that the ☑ Online check box is selected for the connection.

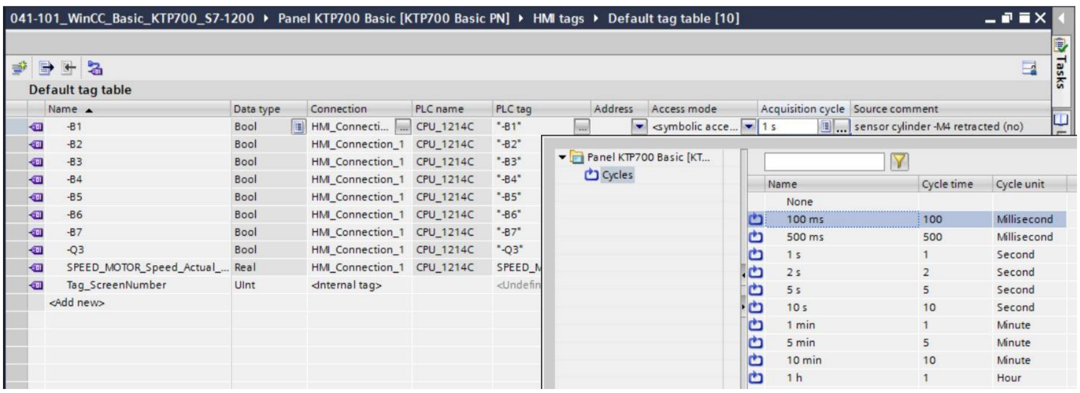


Note: If access protection has been enabled for the CPU 1214C, the access password can also be entered for the panel here.

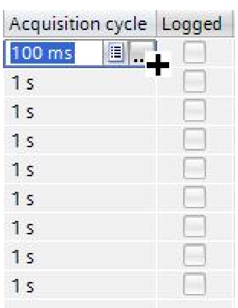
→ To go to the HMI tags, you must double-click the → "Default tag table" in the → "HMI tags" folder below → "Panel KTP700 Basic". All tags that were created with drag & drop have been entered here.

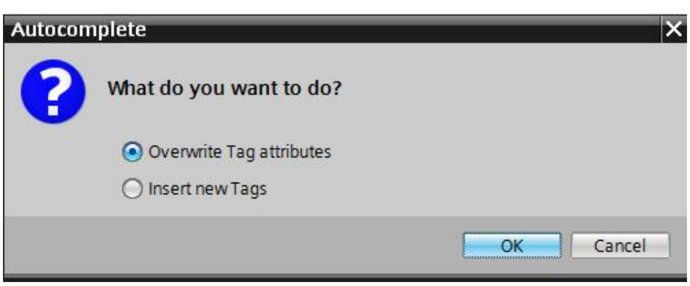


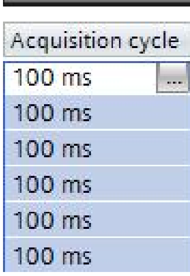
→ In the default tag table, you can check which tags are being accessed in the CPU 1214C. You can also make other settings.
The "Acquisition cycle" of the tags is to be accelerated from 1 second to 100 milliseconds. For this, click on the → ... selection field and double-click a new acquisition cycle → "100 ms" to select it.



→ You can make the settings of other tags using the "Auto complete" function by selecting the bottom right corner of the first entry with the mouse and dragging over the other entries.





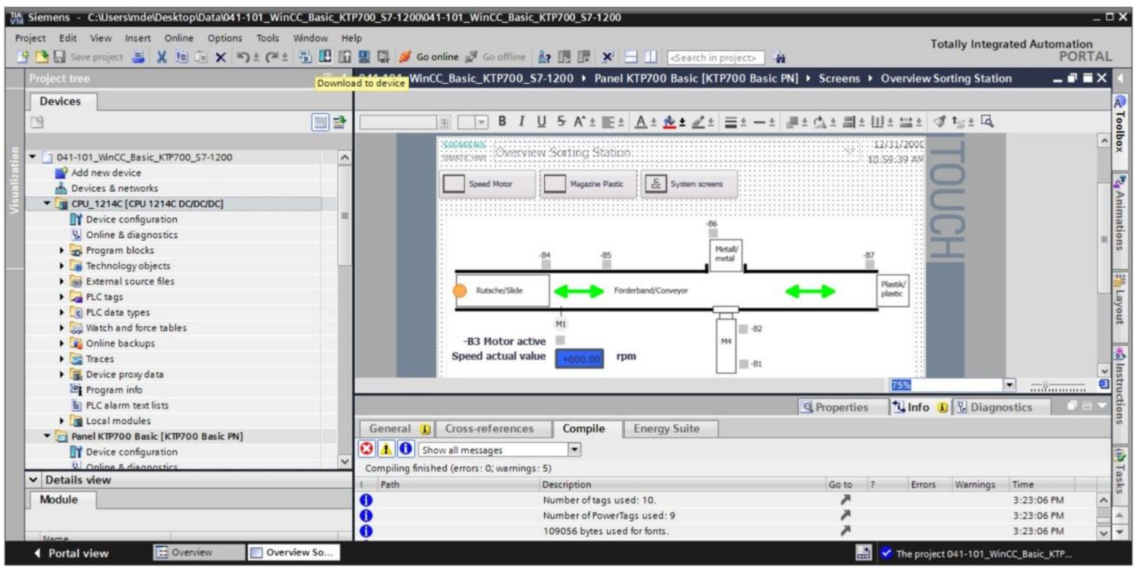


7.10 Downloading the CPU and panel

→ Before the project is downloaded to the CPU and the panel, compile the CPU and panel again and save the project.

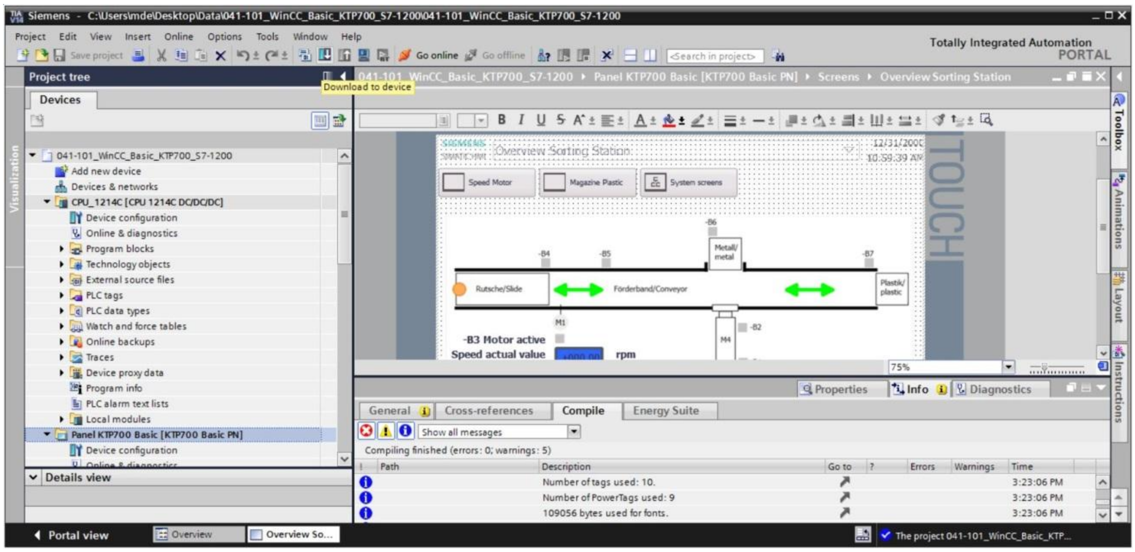
(→ CPU_1214C → → Panel KTP700 Basic → → Save project)

→ After successful compilation, the entire controller with the created program including the hardware configuration, as previously described in earlier modules, can be downloaded. (→)



12

→ To download the visualization to the panel, follow the same procedure. Select the → "Panel KTP700 Basic [KTP700 Basic PN]" folder and click the →  "Download to device" button.



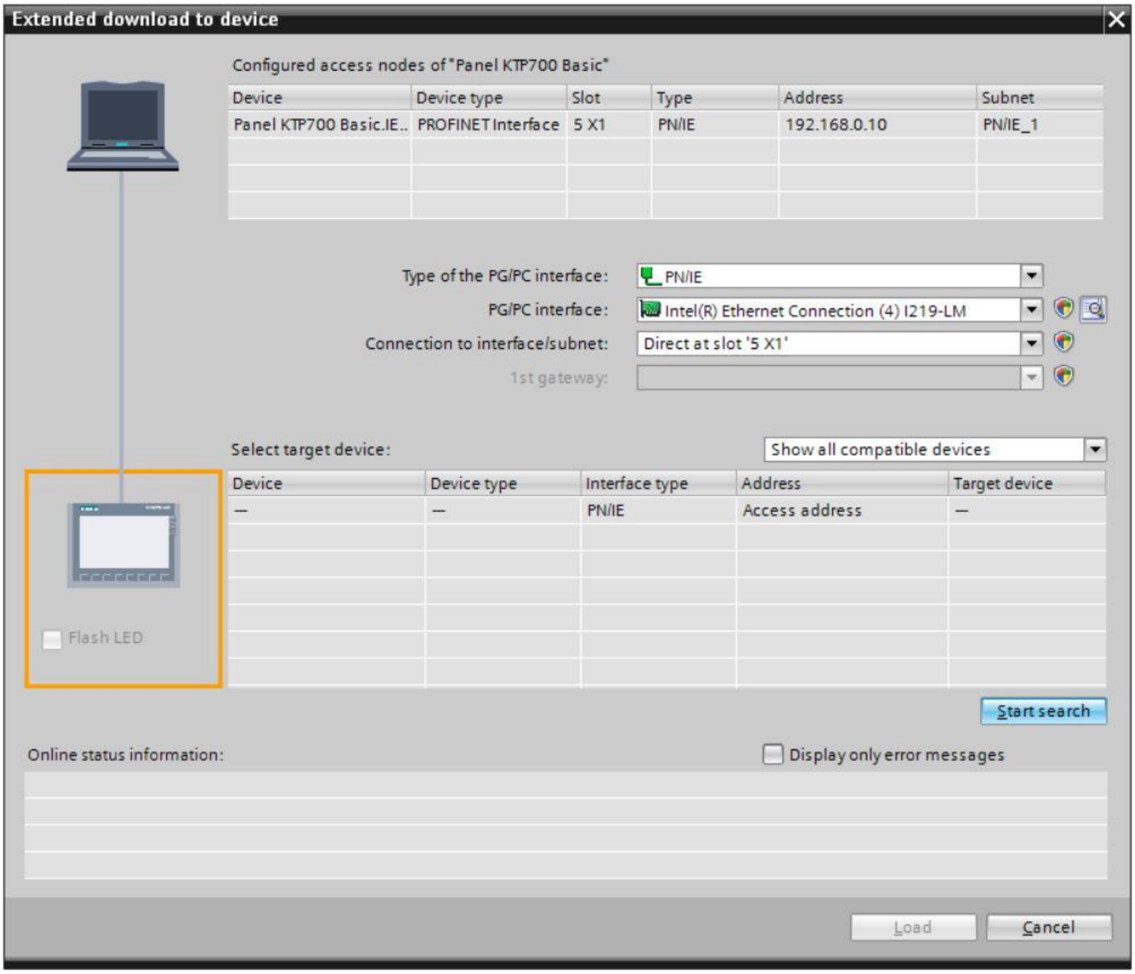
→ The manager for configuration of connection properties (Extended download) opens. First, the interface must be correctly selected. This is done in three steps:

→ Type of the PG/PC interface → PN/IE

→ PG/PC interface → here, e.g. Intel(R) Ethernet Connection I219-LM

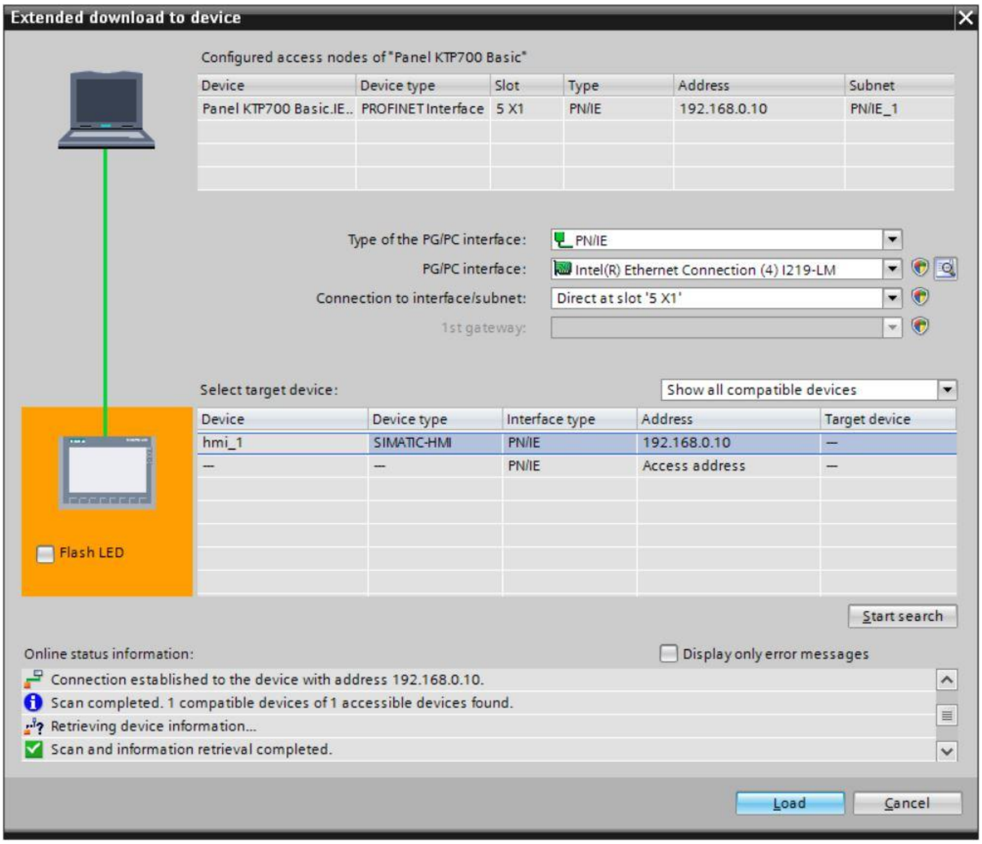
→ Connection to interface/subnet → "PN/IE_1"

The field → "Show all compatible devices" must be selected and the search for devices in the network must be started by clicking the → Start search button.

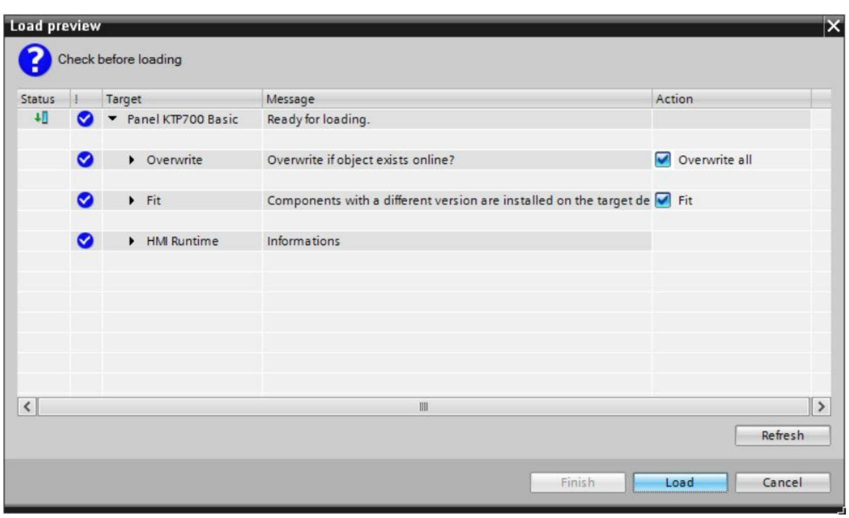


→ If your panel is displayed in the "Compatible devices in target subnet" list, it must be selected and the download must be started.

(→ Device type SIMATIC HMI → "Load")



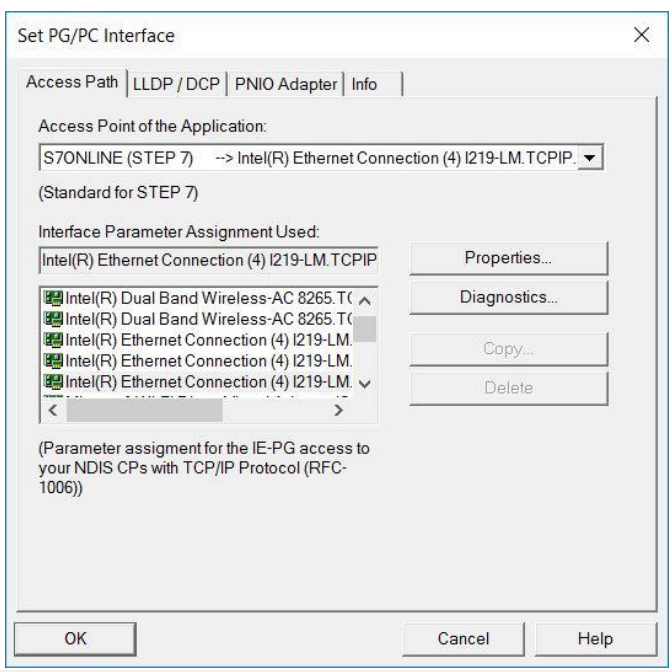
→ You first receive a preview. Confirm the prompt → "Overwrite all" and continue with → "Load".



Note: In the "Load preview", you should see the symbol in each line, in which actions will be performed. You will see additional information in the "Message" column.

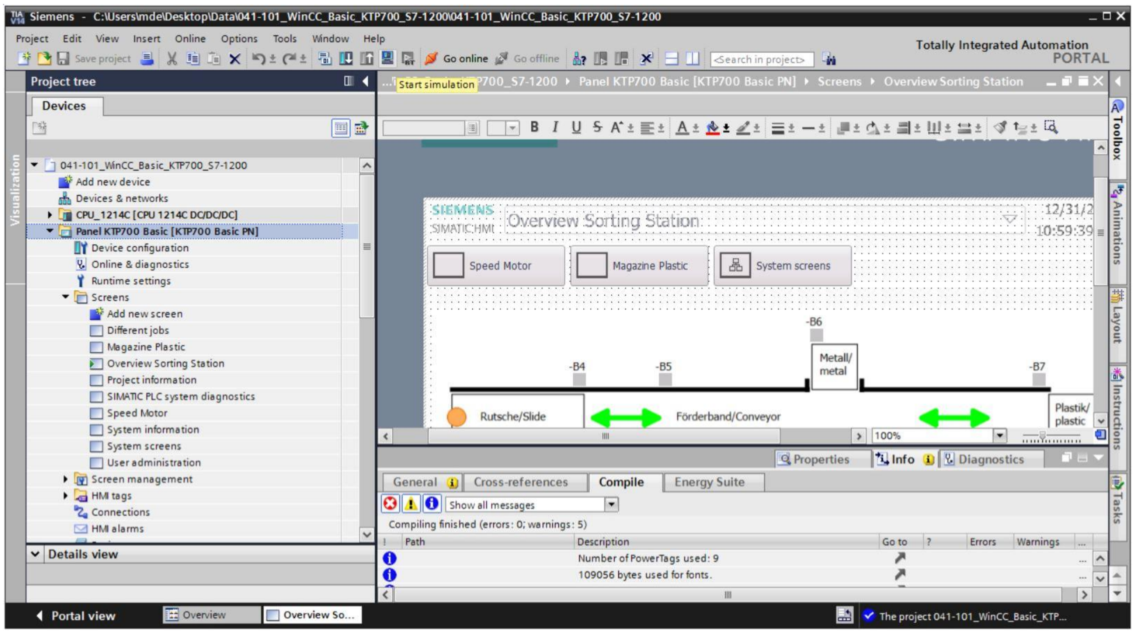
7.11 Testing the process visualization in the simulation

So that a connection can be established between the Runtime Simulation on the PG/PC and the S7-1200 CPU, the PG/PC interface must first be set to TCP/IP.

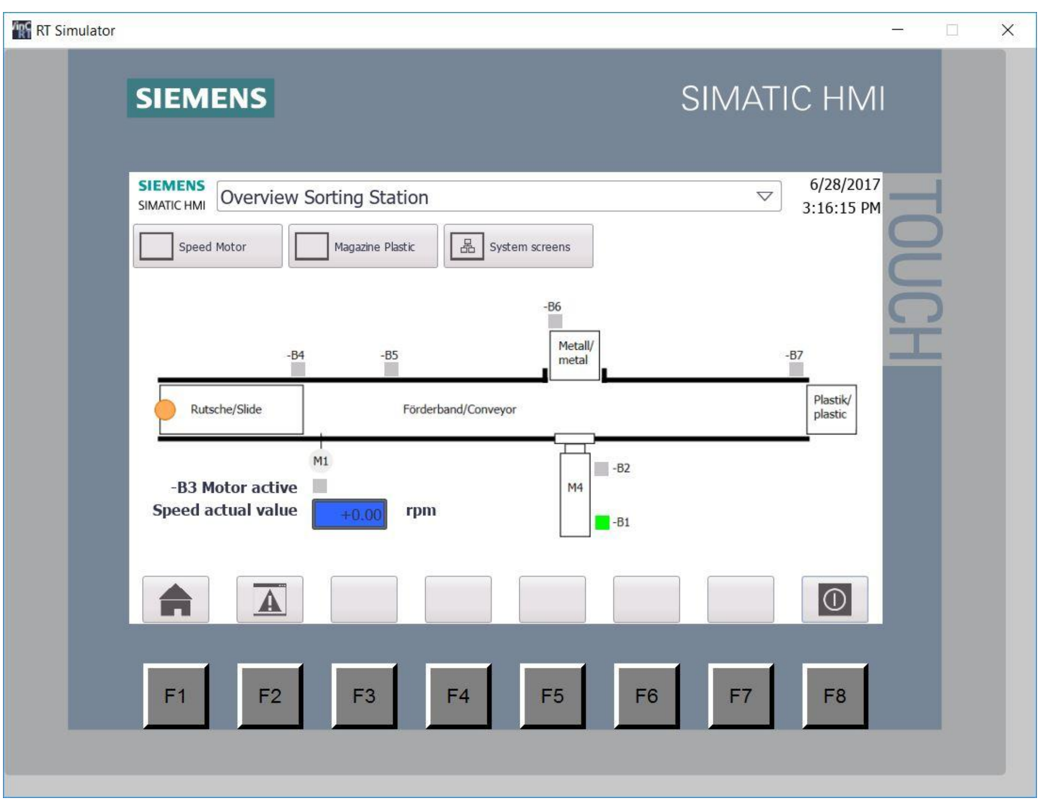
No.	Procedure:
1	Open the Control Panel • Using "Start > Control Panel" • Or using "Start > Settings > Control Panel" (in the classical start menu as in earlier Windows versions)
2	Double-click the "Set PG/PC interface" icon in the Control Panel. 
3	On the "Access Path" tab, set the following parameters: 1. For "Access Point of the Application", select S7ONLINE (STEP 7). 2. From the "Interface Parameter Assignment Used" list, select the interface "TCP/IP(Auto) -> with your network adapter that is connected directly to the panel and controller, e.g. Intel® Ethernet Connection. 3. Click OK and confirm the subsequent prompt with OK. 

12

→ Select "Panel KTP700 Basic" and click the → "Start simulation" button.



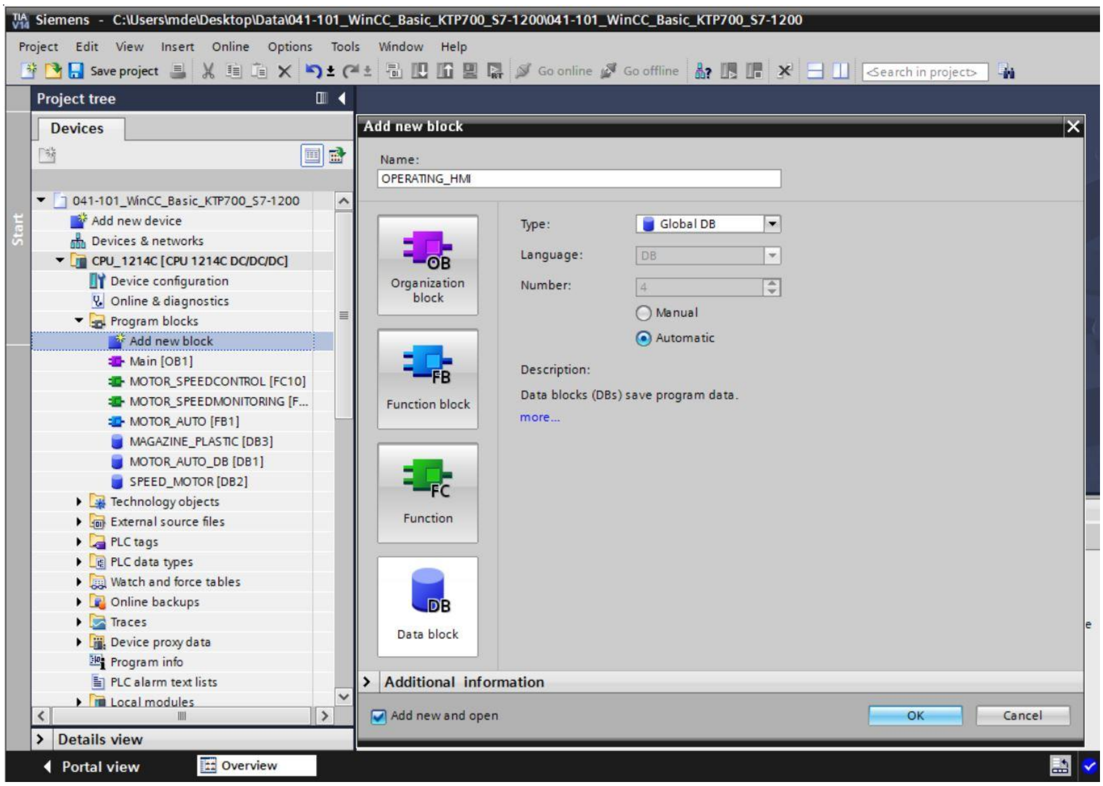
→ The process visualization is performed on the PC in its entirety with connection to the process data in the CPU 1214C. To close the simulation, you can select the → button for "End Runtime" or close the window by clicking on → "X".



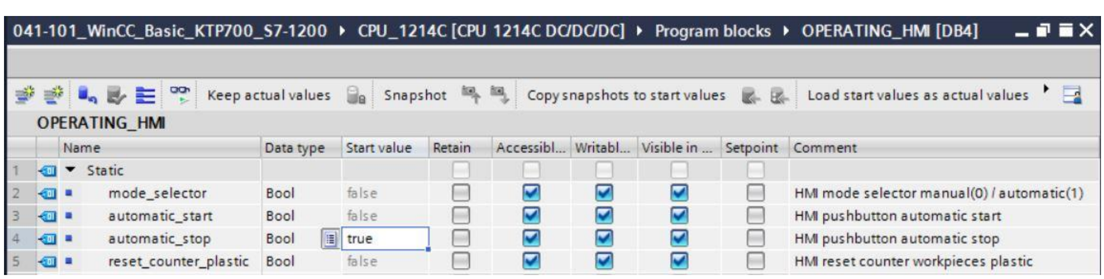
12

7.12 Switches and buttons for process operation

→ To have an interface for process operation available in the PLC, select → "Add new block" in the "Program blocks" folder below "CPU_1214C" and create a global data block "OPERATING_HMI".

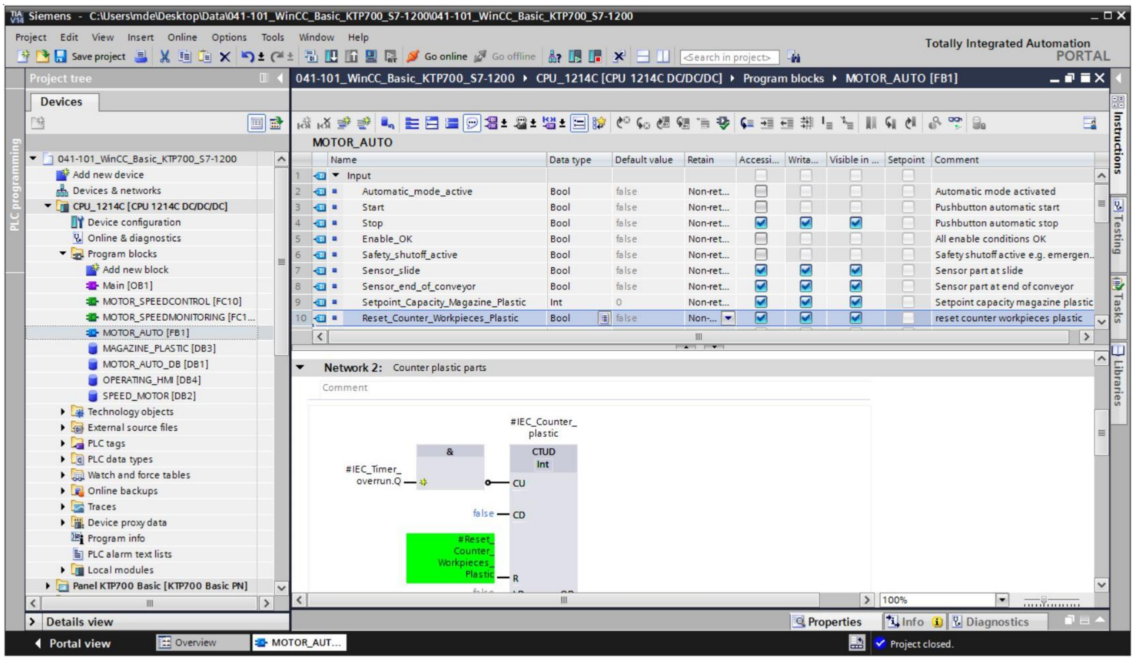


→ In the "OPERATING_HMI" data block, create four tags of data type Bool: → "mode_selector", → "automatic_start", → "automatic_stop" and → "reset_counter_plastic". The start value of the "automatic_stop" is also preassigned with → "true".

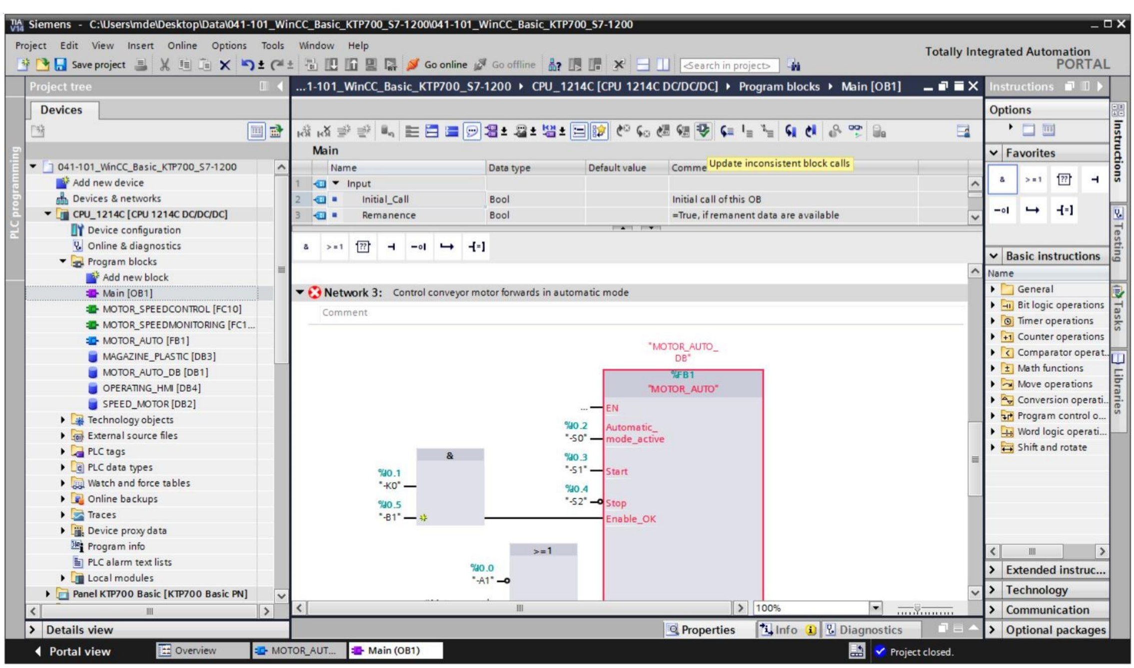


12

→ The "MOTOR_AUTO[FB1]" function block is now expanded to include an input tag → "Reset_Counter_Workpieces_Plastic" of type → "Bool". This tag is move onto the → "R" input of the "CTUD" counter in Network 2 using drag & drop.

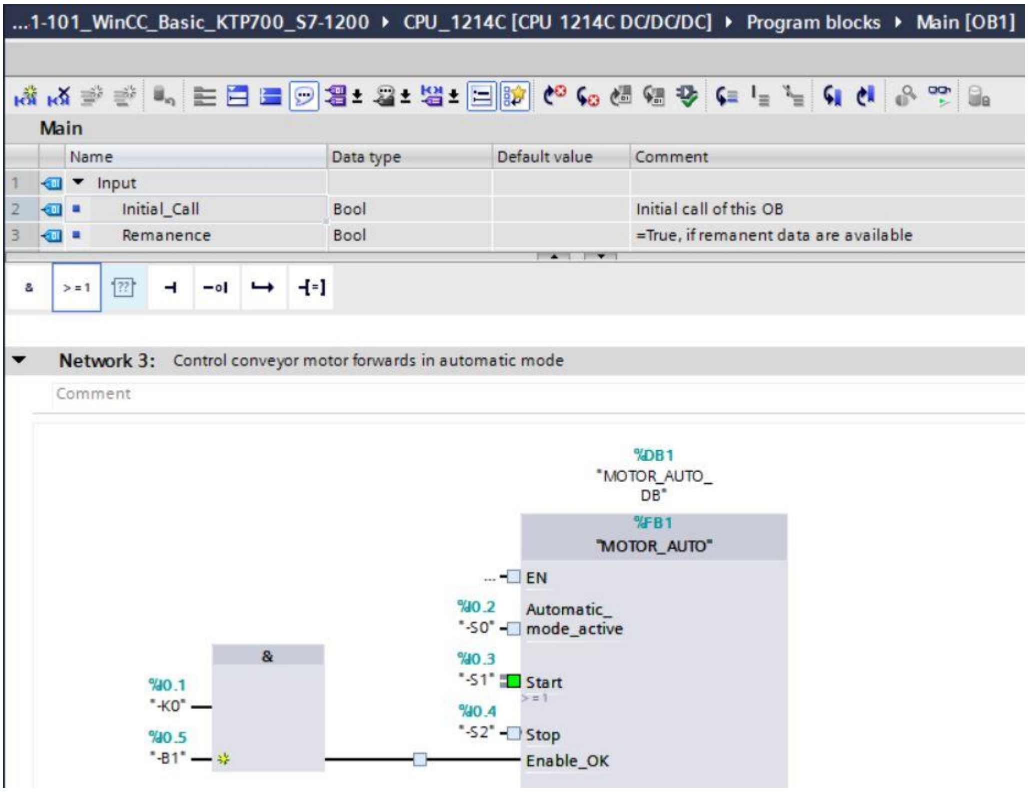


→ Next, the call of the "MOTOR_AUTO[FB1]" function block must be updated in the "Main[OB1]" block. This is done by clicking the → "Update inconsistent block calls" button .

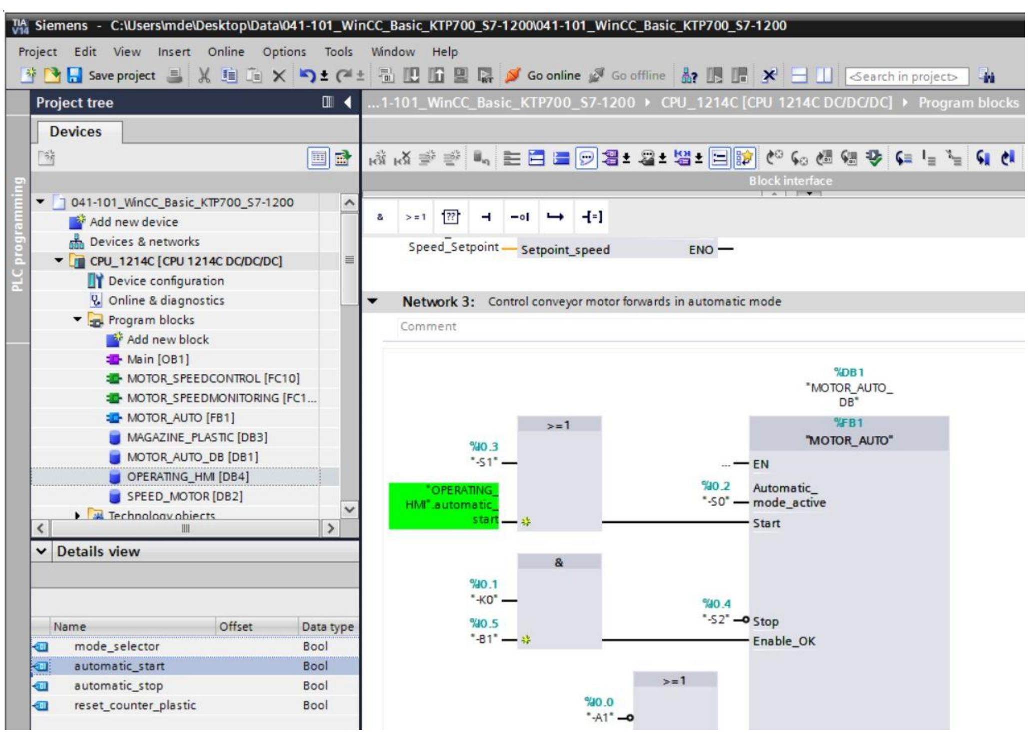


12

→ In Network 3 of the "Main[OB1]" block, drag an → "OR" in front of input tag → "Start".

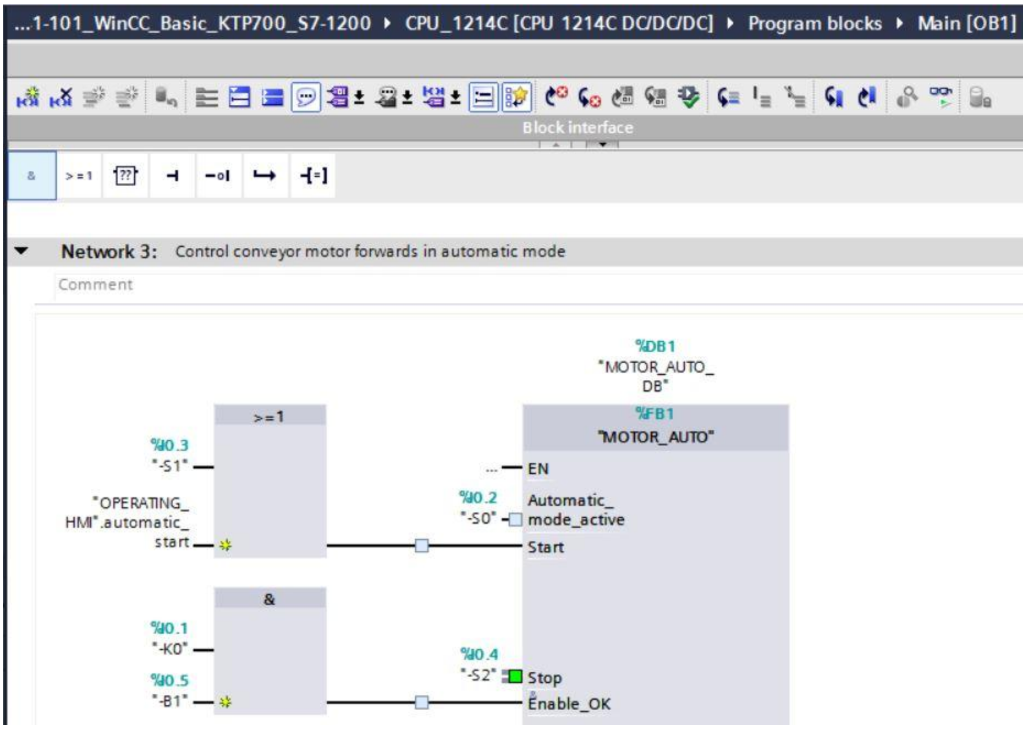


→ The second free input of the → "OR" is connected to the → "automatic_start" tag from data block "OPERATING_HMI".

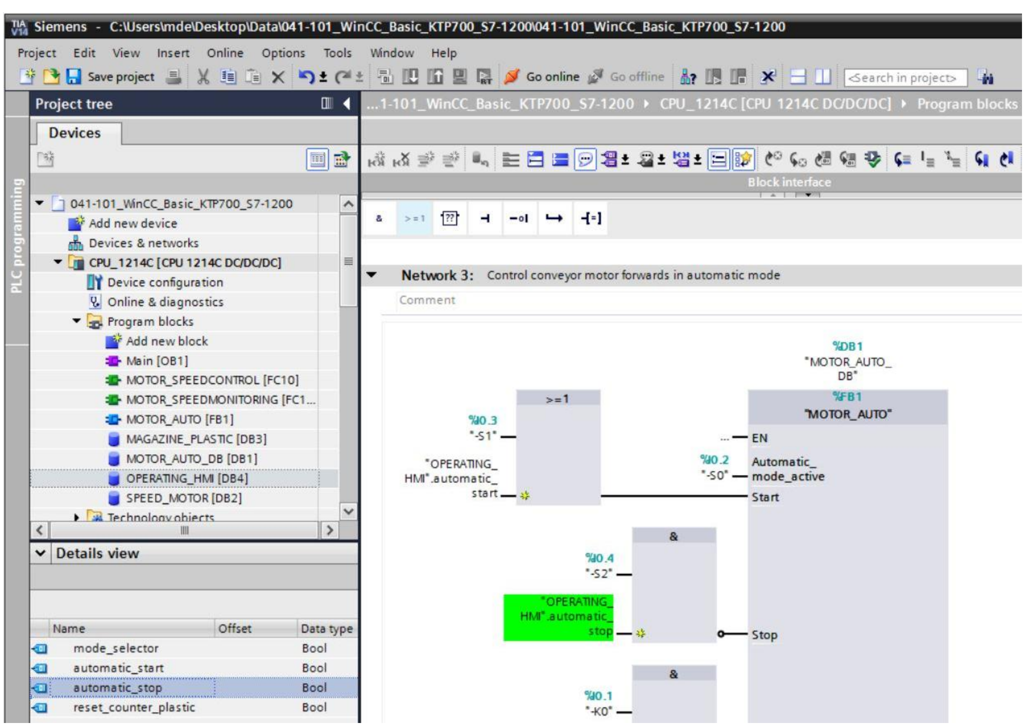


12

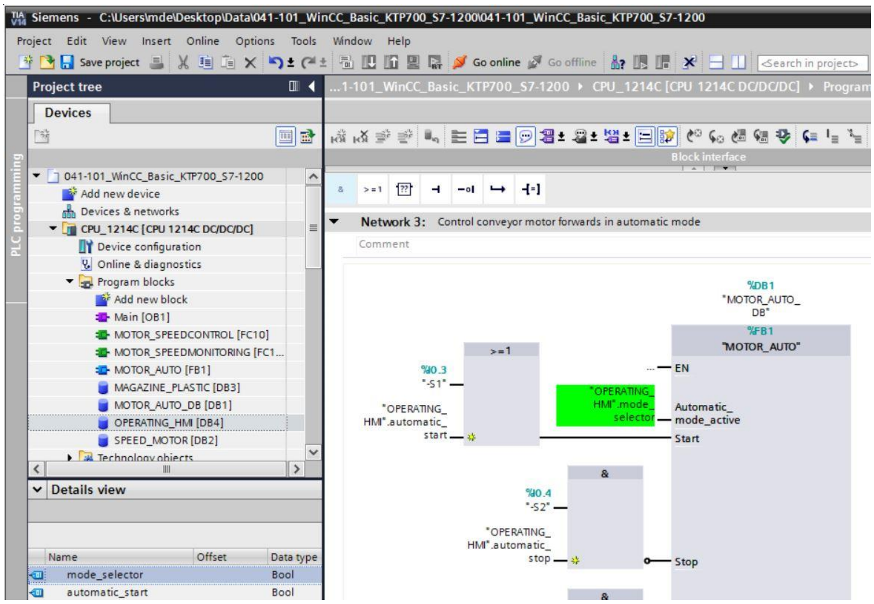
→ In Network 3 of the "Main[OB1]" block, drag an → "AND" in front of input tag → "Stop".



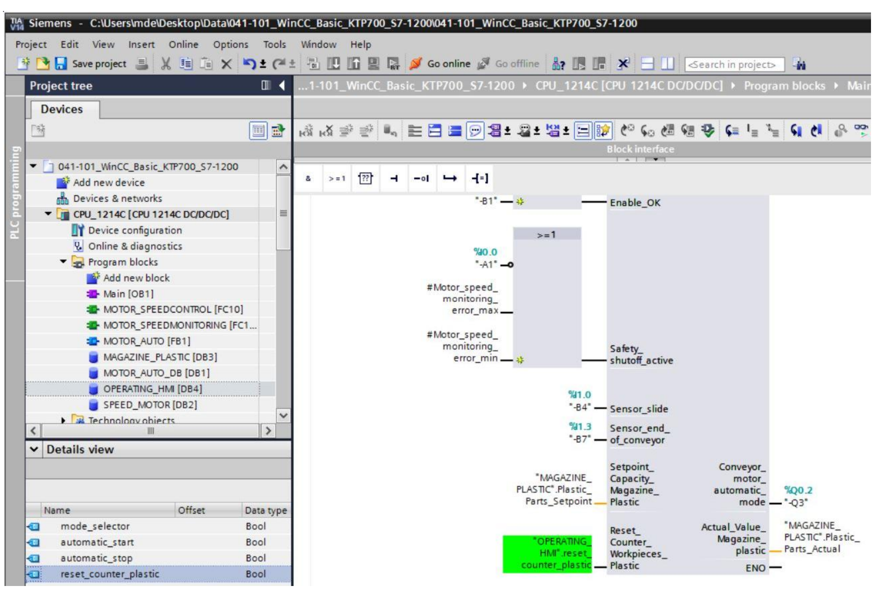
→ The second free input of the → "AND" is connected to the → "automatic_stop" tag from data block "OPERATING_HMI".



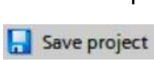
→ The input tag → "Automatic_mode_active" is connected to the → "mode_selector" tag from data block "OPERATING_HMI".



→ The input tag → "Reset_Counter_Workpieces_Plastic" is connected to the → "reset_counter_plastic" tag from data block "OPERATING_HMI".



→ Compile the CPU again and save the project.

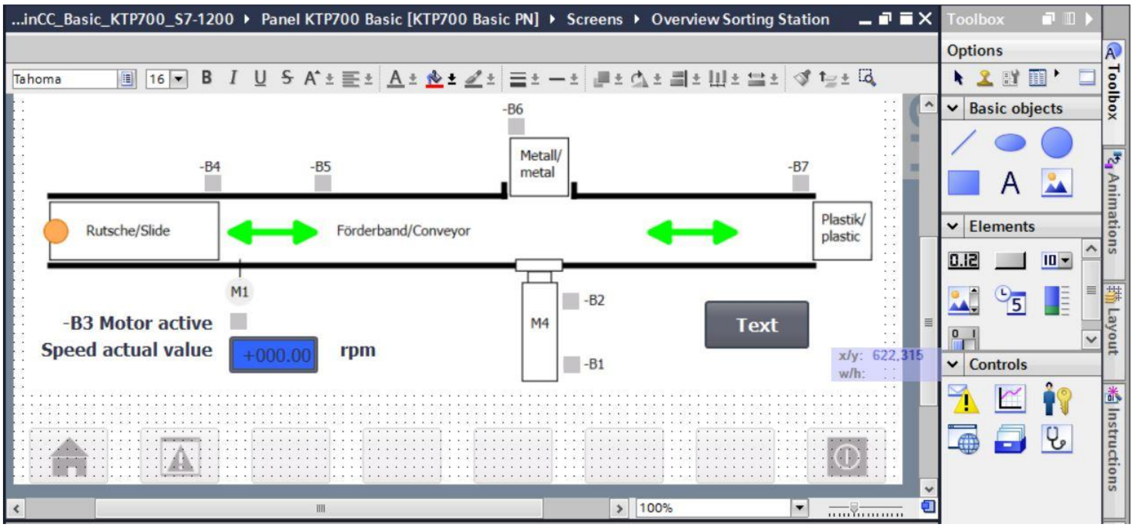
(→ CPU_1214C →  → )

→ Download the changed program including the hardware configuration to the CPU 1214C.

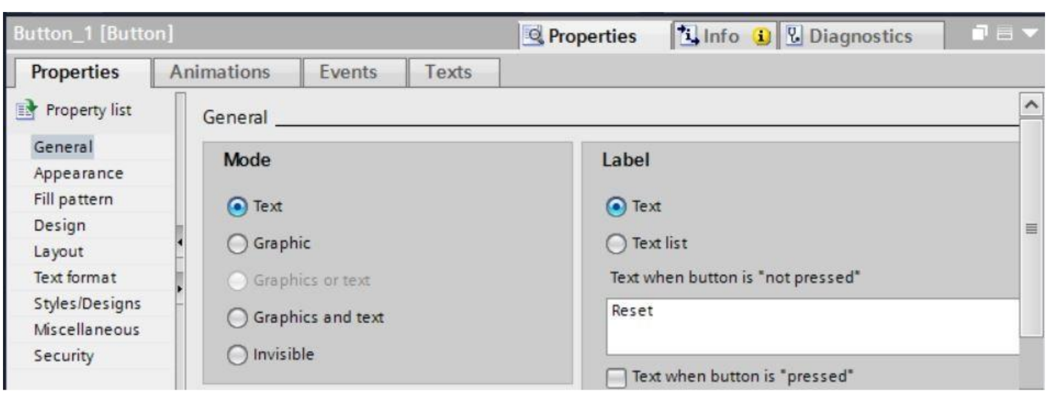
(→ )

12

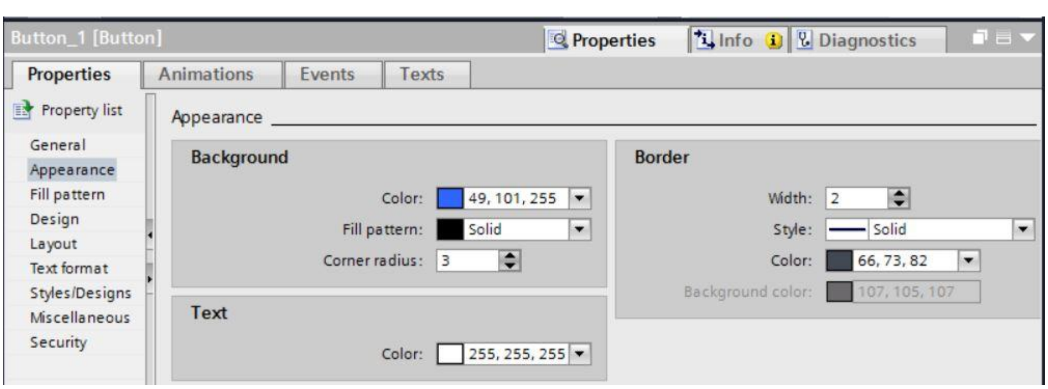
→ To implement a pushbutton that resets the workpiece counter for the plastic parts, use drag & drop to move the → "Button" object from → "Elements" in Toolbox to the "Overview Sorting Station" screen below the plastic parts storage.



→ Under "General" in "Properties", enter → "Reset" for "Label".

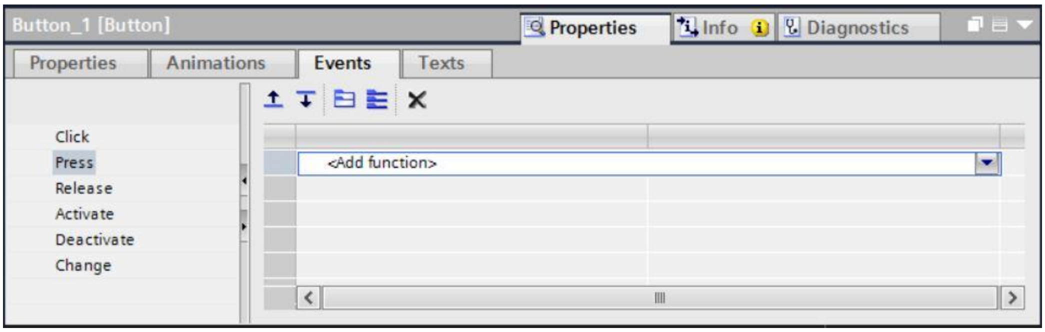


→ Under "Appearance" in "Properties", change the fill pattern to "Solid" and "Color" of "Background" to → "Blue".

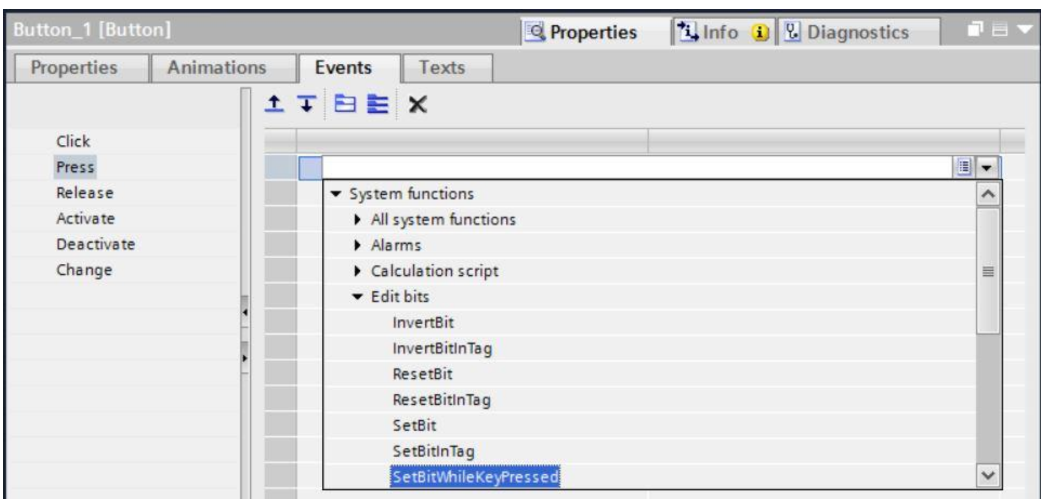


12

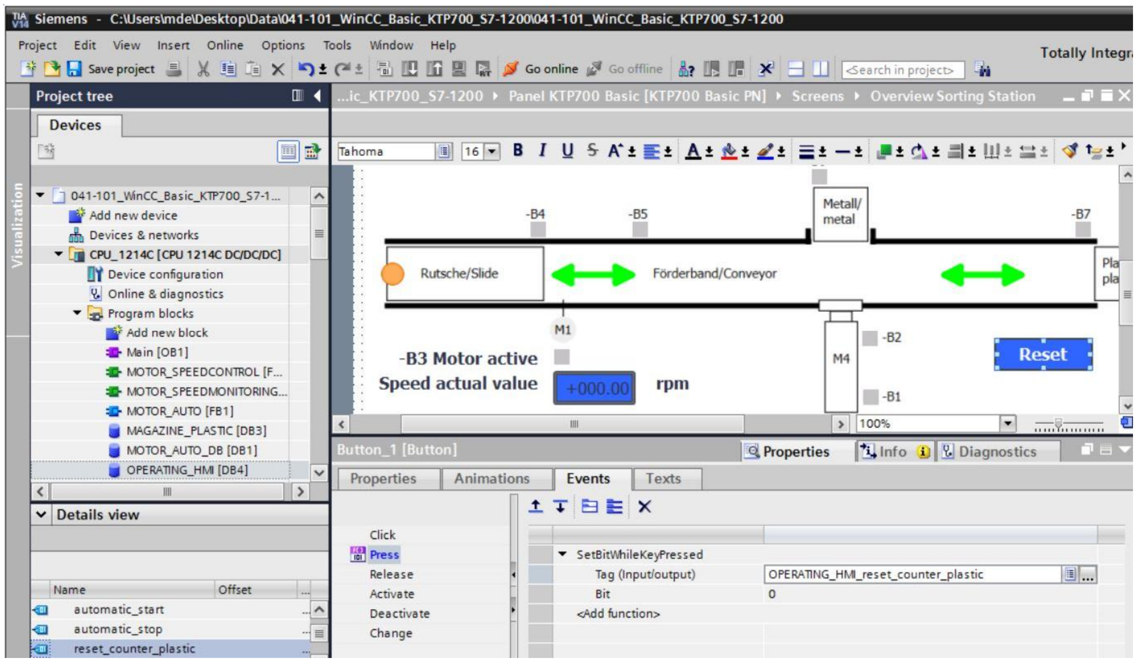
→ The functionality must also be configured as a pushbutton. To do this, go to the "Events" menu, select the → "Press" event and → "<Add function>".



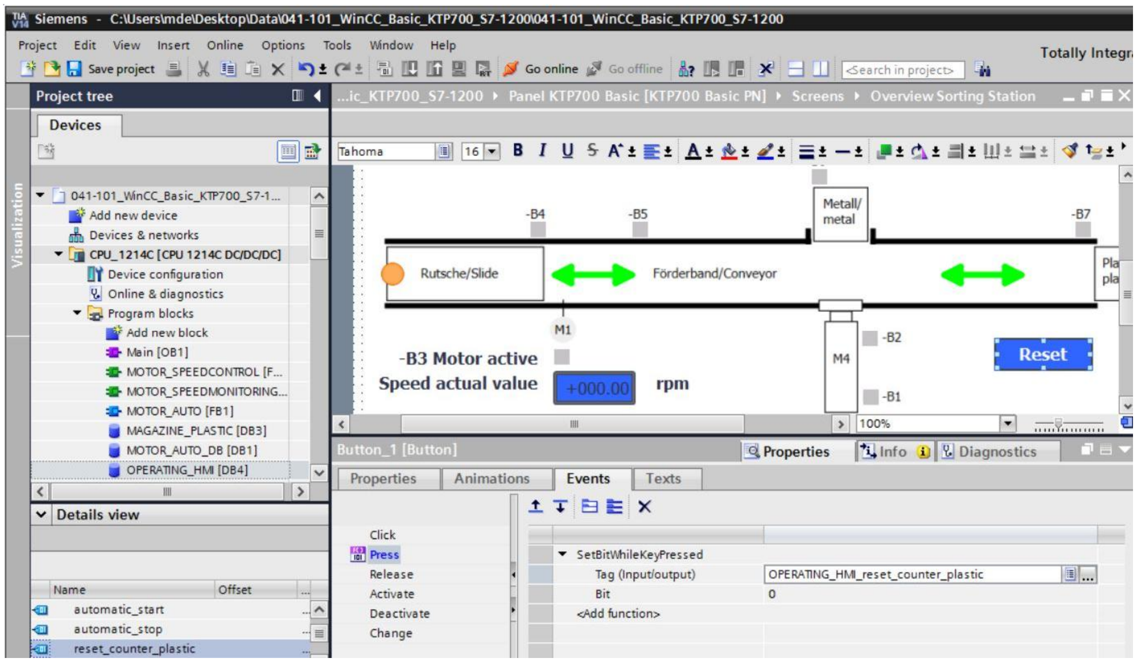
→ Under "System functions", select "Edit bits" and → "SetBitWhileKeyPressed" as the function.



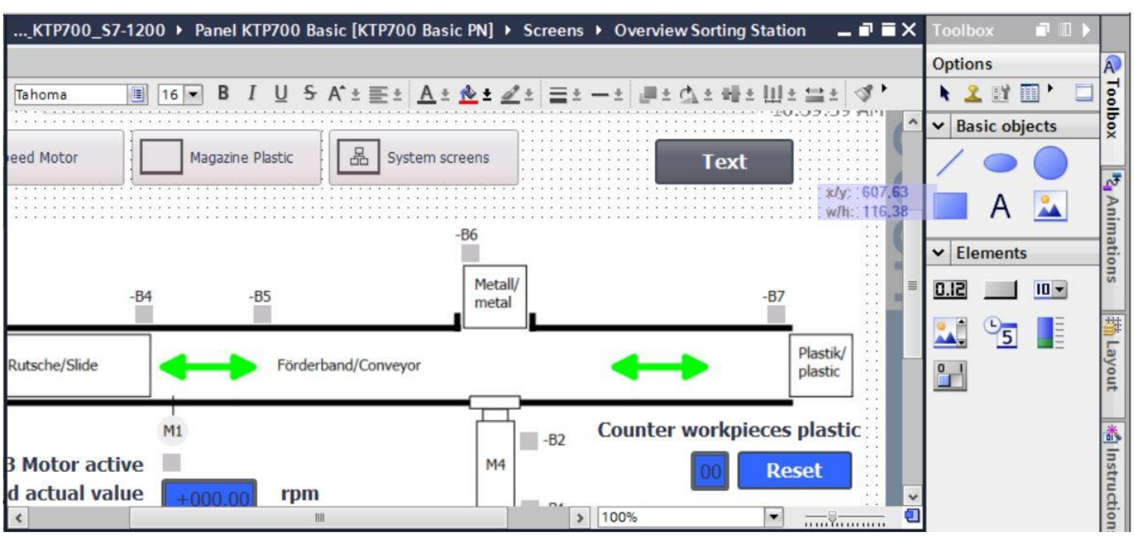
→ For the process connection, select → "Program blocks" and the → "OPERATING_HMI[DB4]" data block of → "CPU_1214C". Next, drag the → "reset_counter_plastic" tag from the → Details view to the "Tag (input/output)" field.



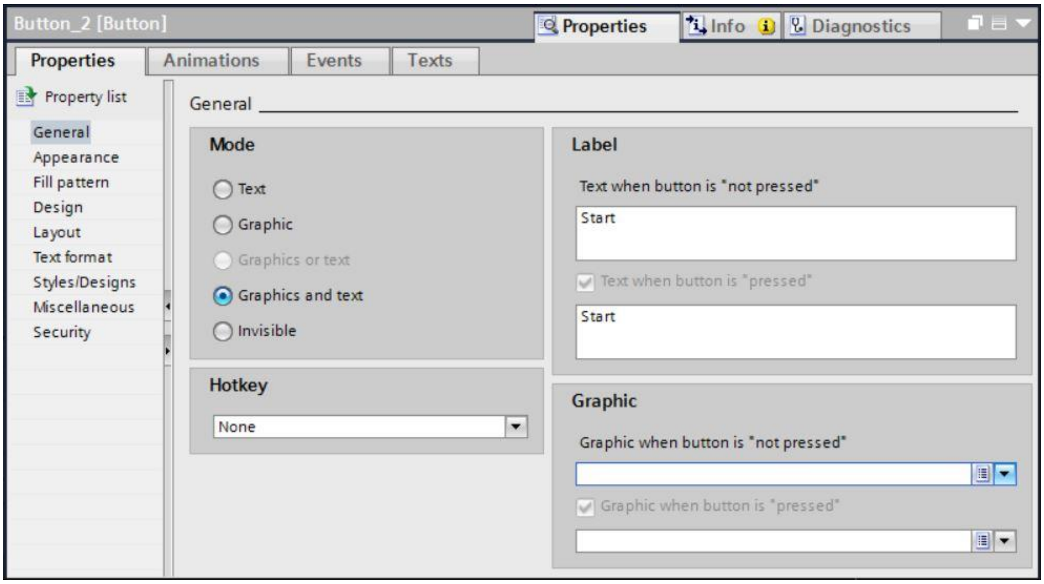
→ As previously shown in the document, insert a text → "Counter workpieces plastic" above the button and a display of the → "actual_plastic_parts" tag from the "MAGAZINE_PLASTIC[DB3]" block to the left of the button.



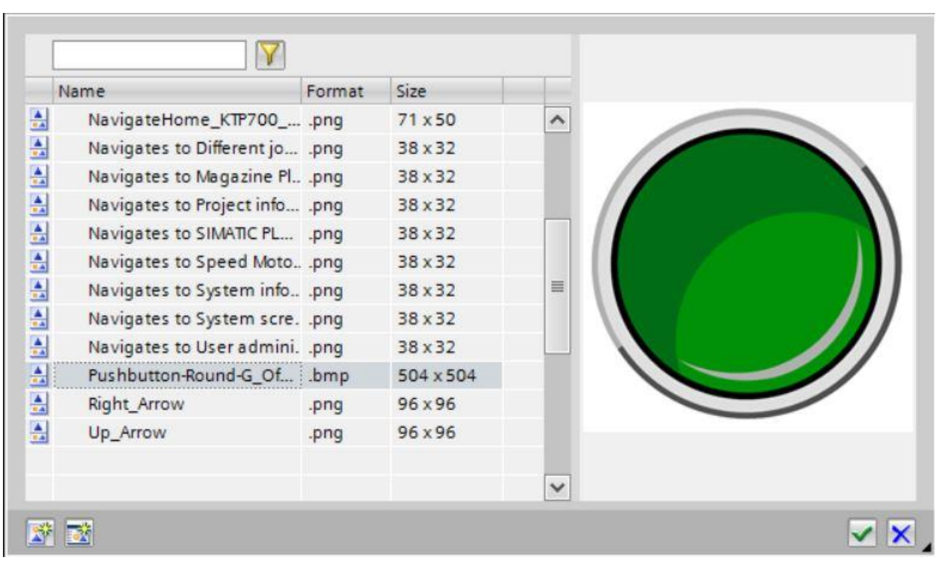
→ To implement the button, use drag & drop to move the → "Button" object from → "Elements" in Toolbox to the top next to the buttons for the screen change.



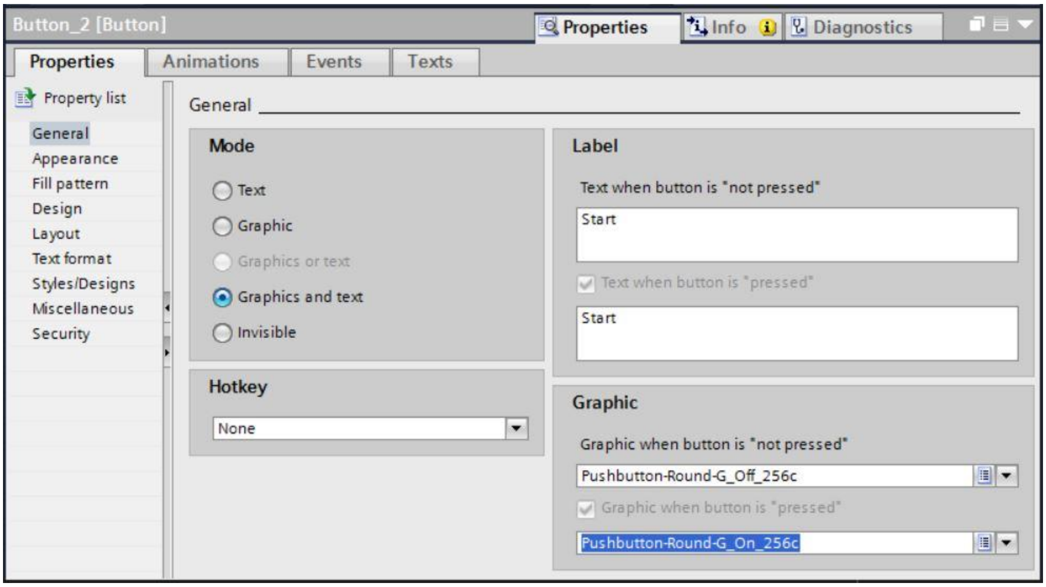
→ Under "General" in "Properties", change "Mode" to → "Graphics and text". Click the symbol to open the selection dialog for the → "Graphic when button is not pressed".



→ Next, click the symbol for "Create graphic from file" and double-click the "Pushbutton-Round-G_Off_256c.bmp" file in the "SCE_EN_041-101_Screens" folder in the displayed dialog.

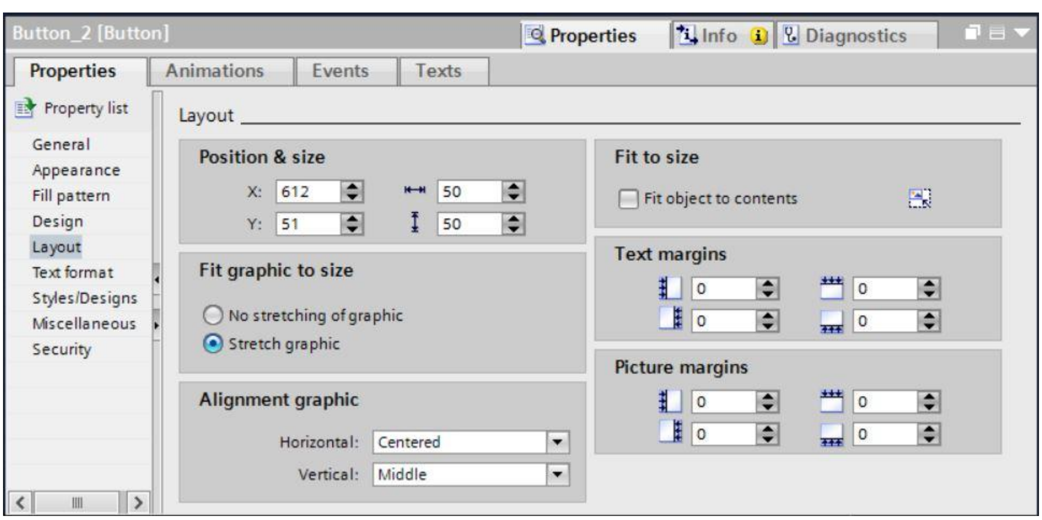


→ Similarly, select the "Pushbutton-Round-G_On_256c.bmp" file in the "SCE_EN_041-101_Screens" folder for the "Graphic when button is pressed".



Note: The created graphics are stored in the project in the "Languages & resources" path under "Graphic collection".

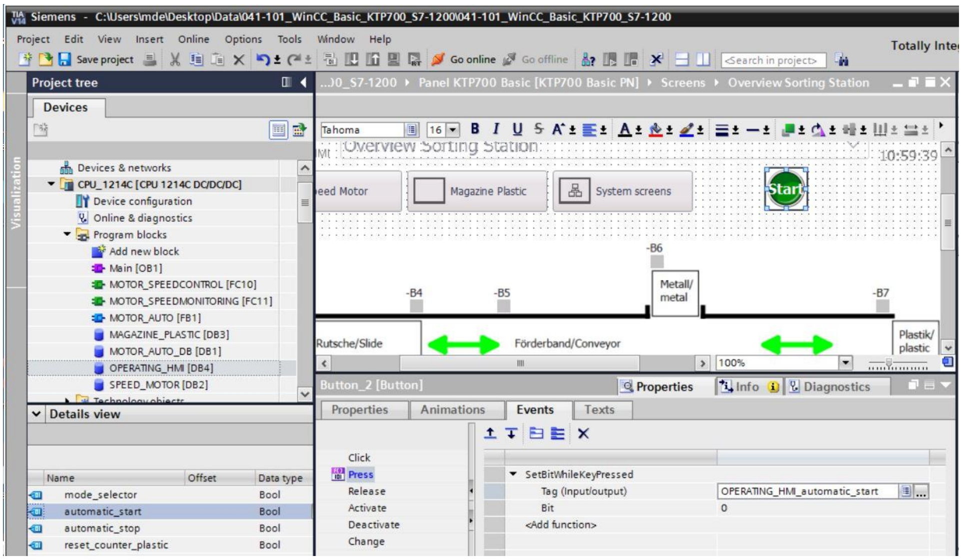
→ Under "Layout" in "Properties", change the size of the button under → "Position & size".



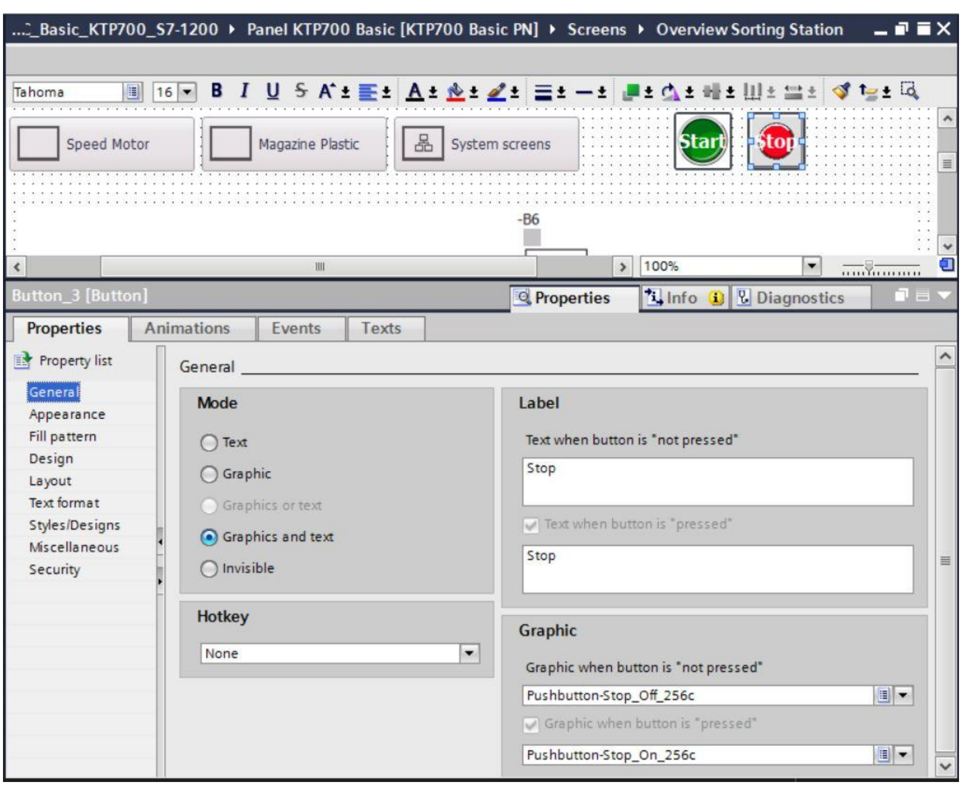
12

→ The functionality as a pushbutton is implemented again as a → "Press" event with "System function" → "SetBitWhileKeyPressed".

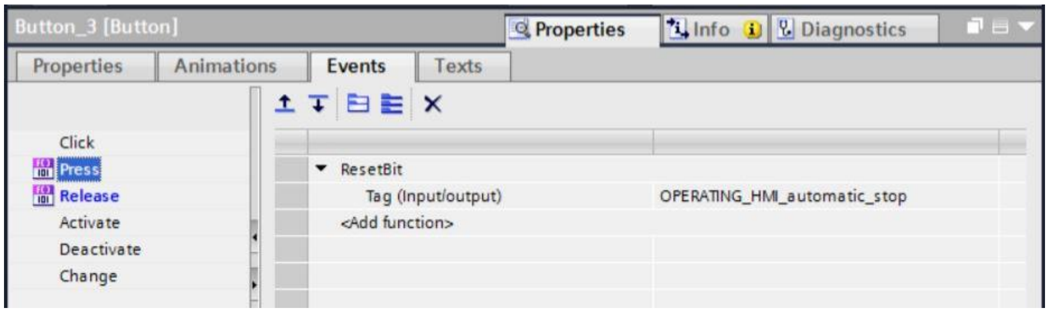
The → "automatic_start" tag from the → "OPERATING_HMI[DB4]" data block is used for the process connection.

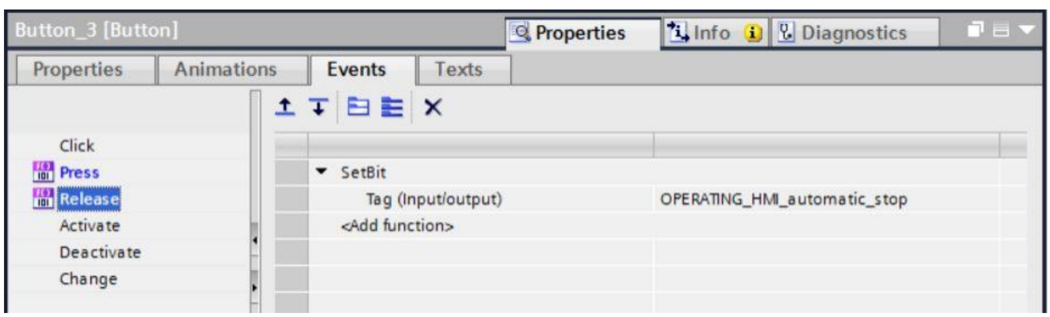


→ Next, a "button" for the Stop pushbutton is inserted, as shown in the last steps. The "Pushbutton-Stop_Off_256c.bmp" and "Pushbutton-Stop_On_256c" files in the "SCE_EN_041-101_Screens" folder are used as graphics.

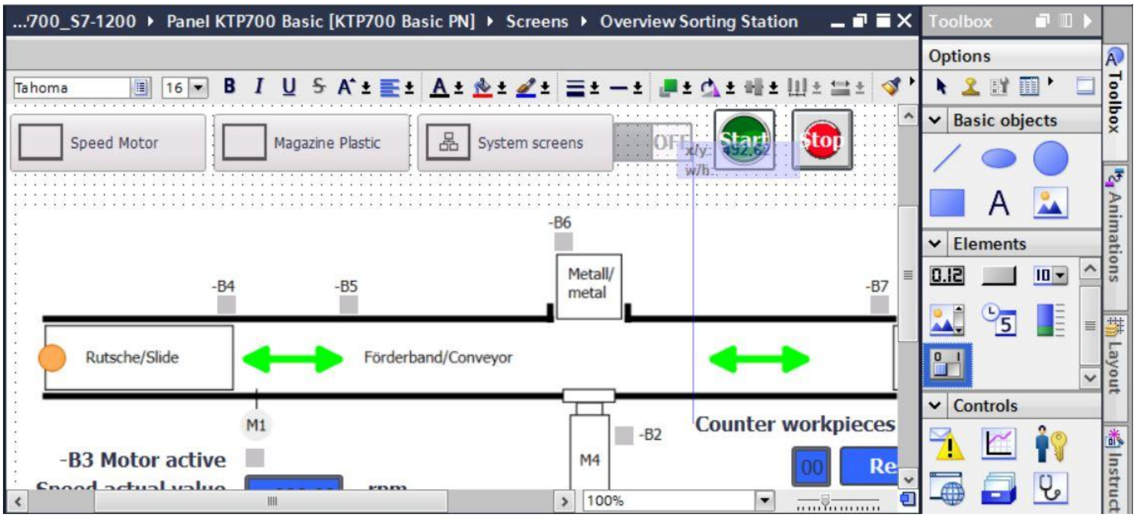


→ The functionality as a "normally closed" pushbutton is implemented here with two events. The first event is the → "Press" event with "System function" → "ResetBit" and the second event is the → "Release" event with "System function" → "SetBit". In both cases, the → "automatic_stop" tag from the → "OPERATING_HMI[DB4]" data block is used for the process connection.

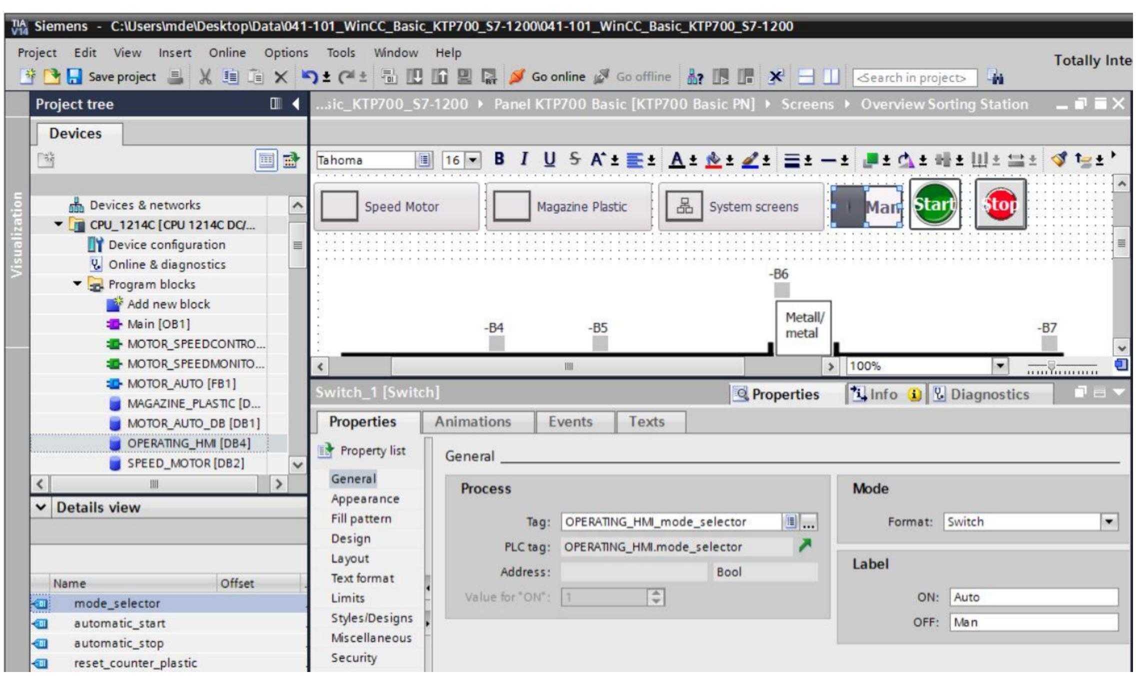




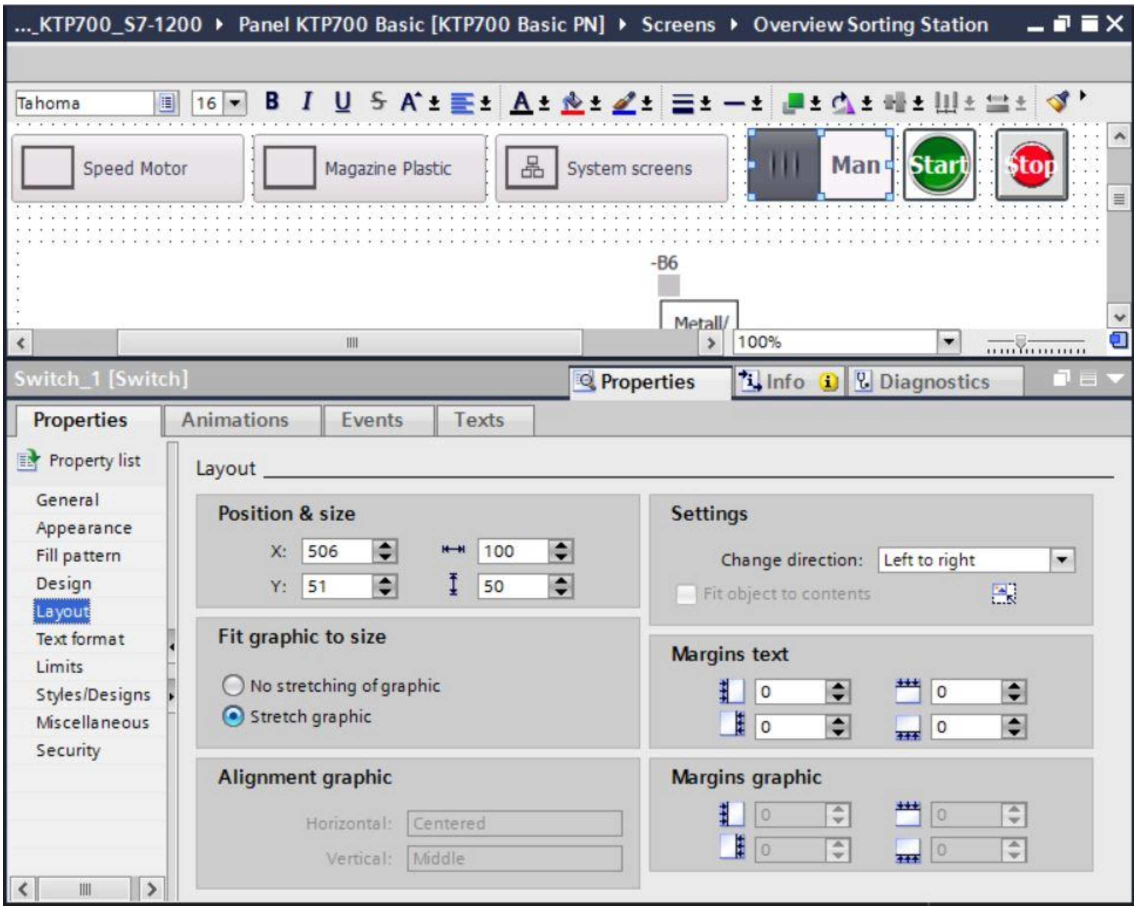
→ To implement the mode selector, use drag & drop to move the → "Switch" object from → "Elements" in Toolbox to the top between the buttons for the screen change and the Start pushbutton.



→ Under "General" in "Properties", enter the texts → "Auto" for the "ON" state and → "Man" for the "OFF" state. The → "mode_selector" tag from the → "OPERATING_HMI[DB4]" data block is used for the process connection.



→ Under "Layout" in "Properties", change the size of the mode selector under → "Position & size".



→ Compile the panel and save the project. (→ Panel KTP700 Basic → → Save project)

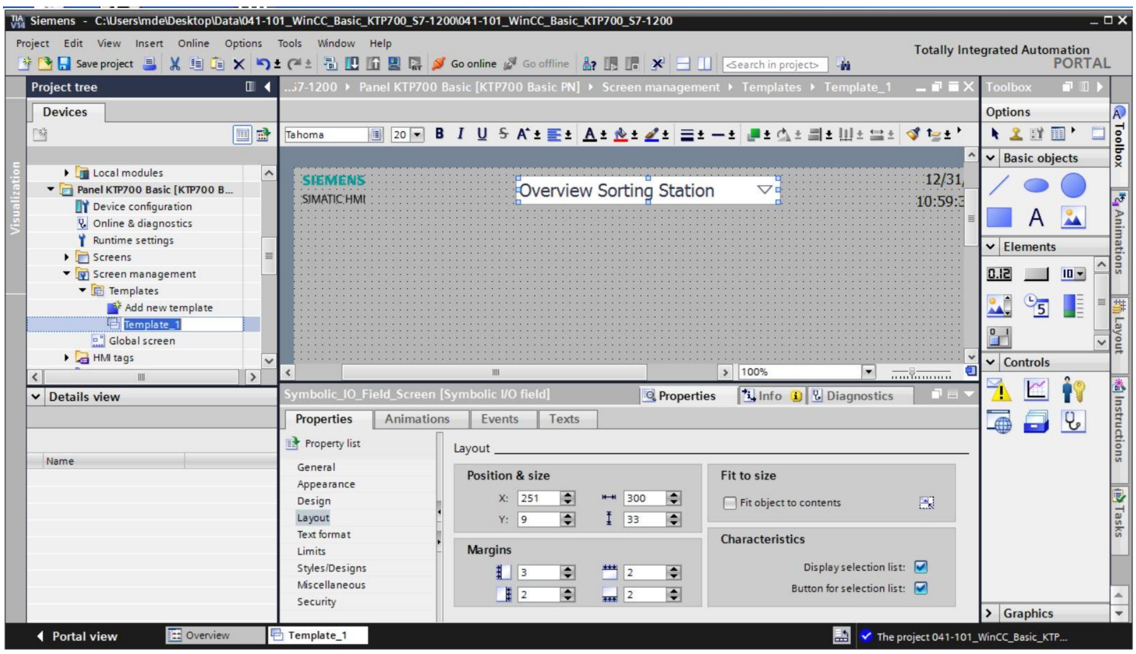
→ Download the modified visualization to the panel. (→)

12

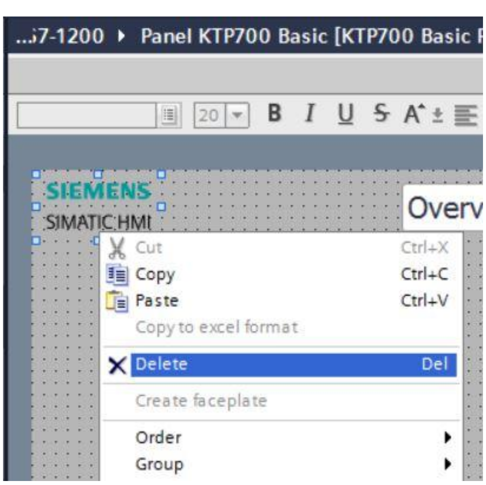
7.13 Changing the header and footer in the template

→ The plant states should be displayed universally in the header. "Template_1" was created for the header and footer by the wizard when the panel was created. The footer contains the system buttons. The logo, date and time and the symbolic IO field for selection and display of screens have already been created in the header.

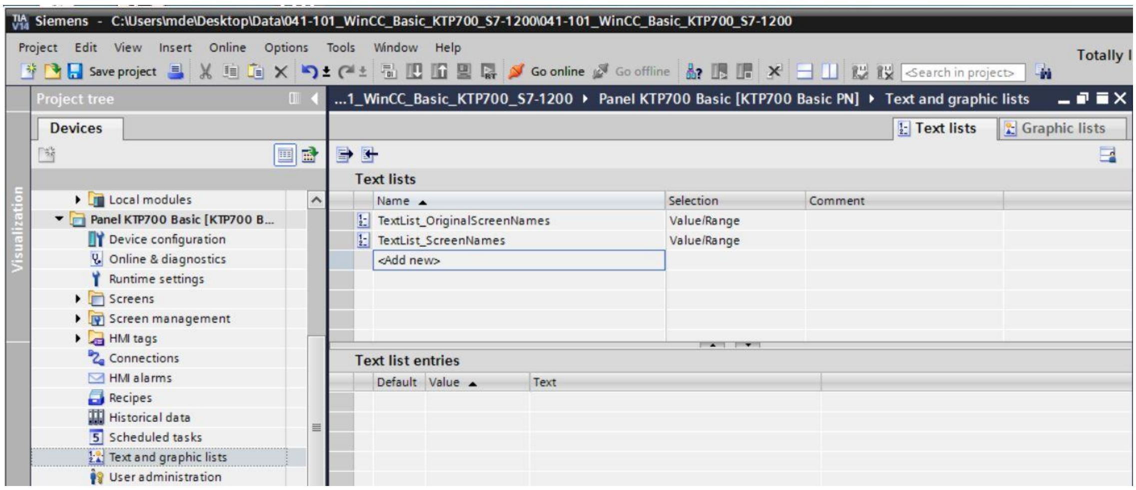
→ First, the "Symbolic_IO_Field_Screen" is changed to the size specified here under "Layout" in "Properties" in → "Position & size".



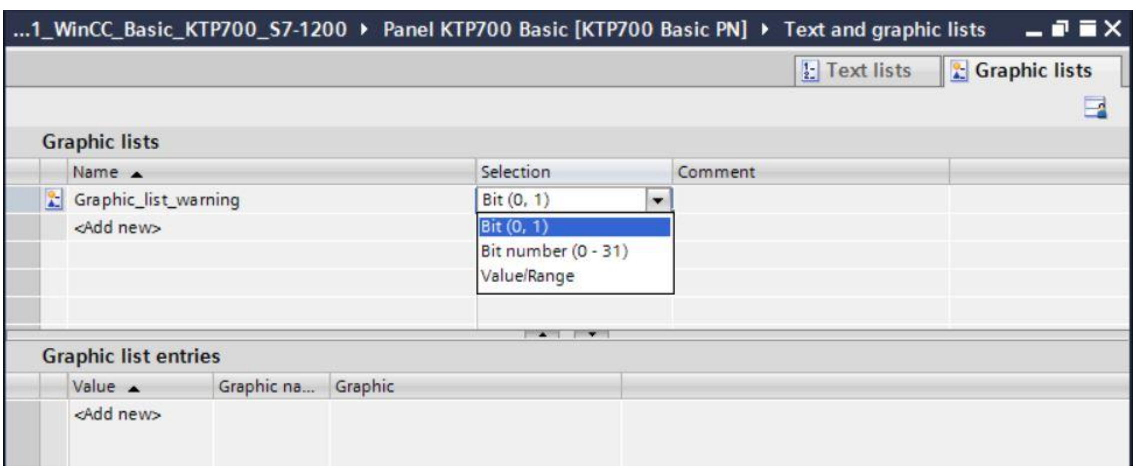
→ Delete the logo on the left side of the header by right-clicking the → Graphic view for the LOGO and clicking → "Delete".



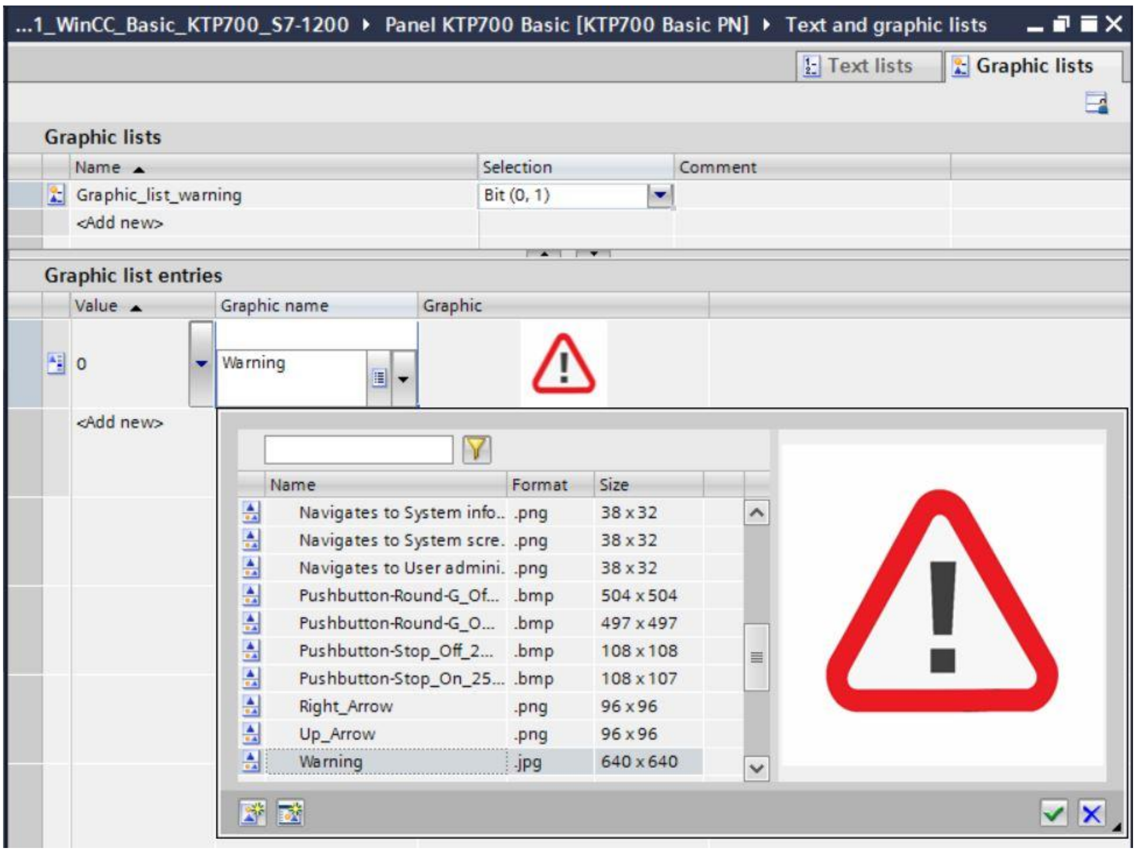
→ Open the → "Text and graphic lists" folder below "Panel KTP700 Basic".



→ Under "Graphic lists", create an additional → "Graphic_list_warning" with → Selection "Bit (0,1)".

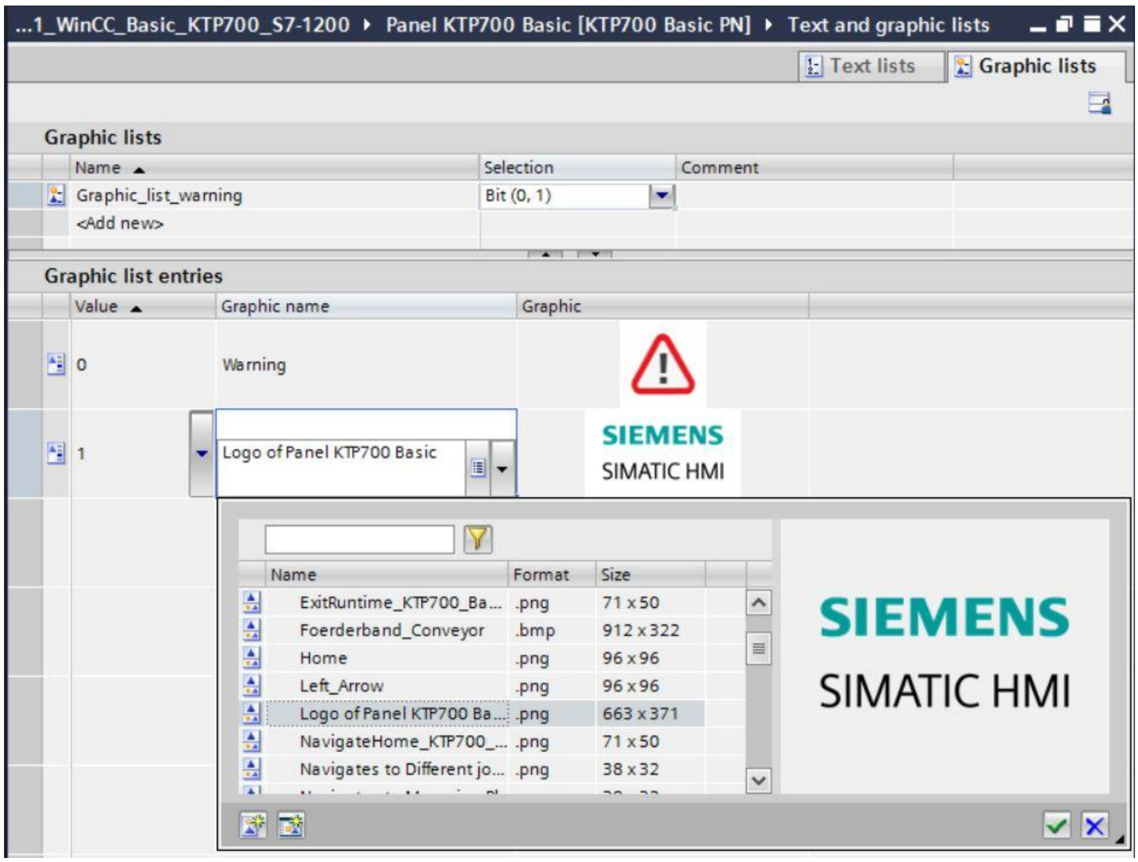


→ Click the symbol next to "Value 0" to open the selection dialog for the graphics stored in the "Graphic collection" in the "Languages & resources" path. Next, click the symbol for "Create graphic from file" and double-click the "Warning.bmp" file in the "SCE_EN_041-101_Screens" folder in the displayed dialog. This file is now stored in the "Languages & resources" path under "Graphic collection".

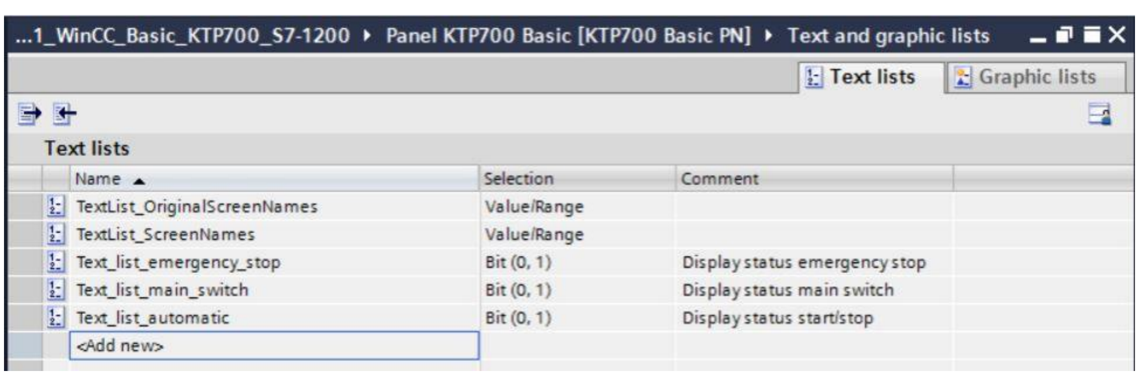


12

→ The graphic that you want to assign to "Value 1" is already stored in the "Languages & resources" path under "Graphic collection". After you click the → symbol, you can select the → "Logo of Panel KTP700 Basic" file directly here.

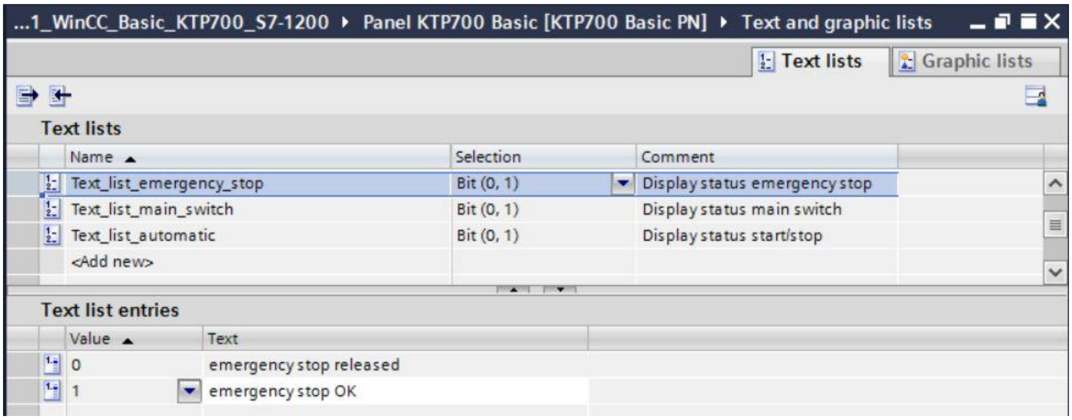


→ Change to the "Text lists" and create three text lists → "Text_list_emergency_stop" → "Text_list_main_switch" and → "Text_list_automatic", each with → Selection "Bit (0,1)".

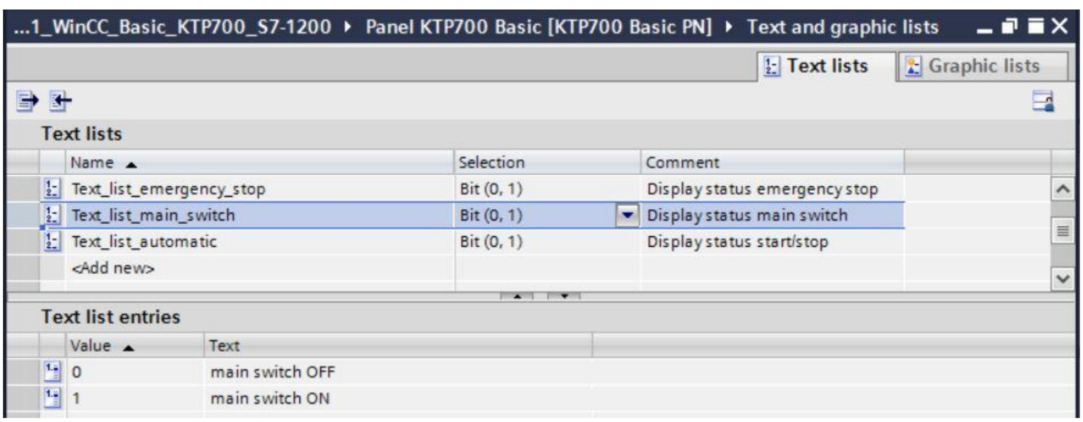


12

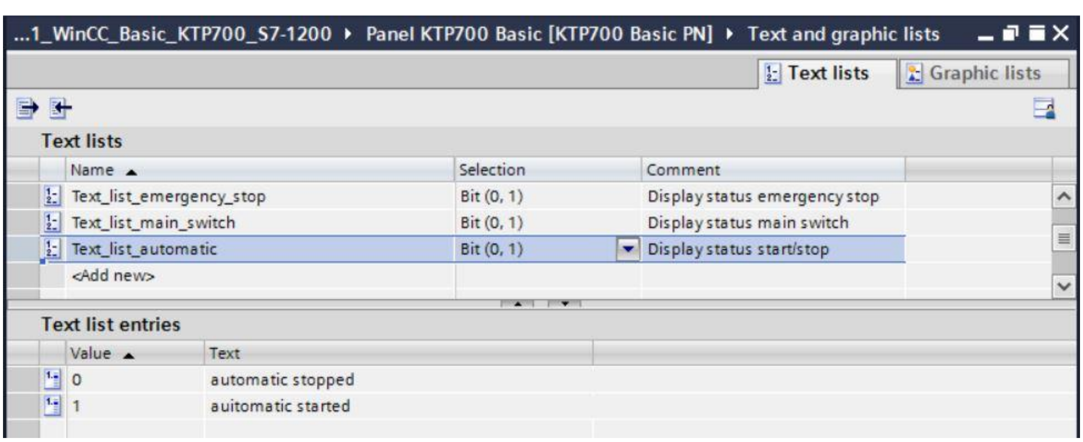
→ Specify the following assignments in "Text_list_emergency_stop": "Value 0" → "emergency stop released" and → "Value 1" → "emergency stop OK".



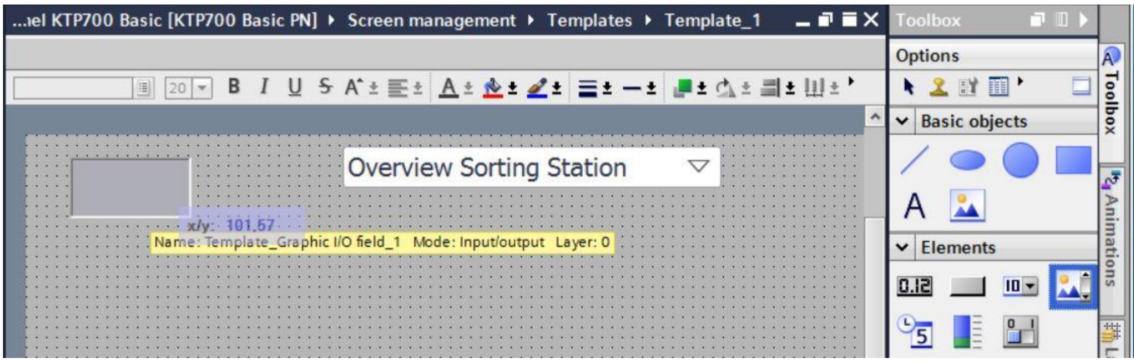
→ Specify the desired assignments in "Text_list_main_switch". "Value 0" → "main switch OFF" and → "Value 1" → "main switch ON".



→ Specify the following assignments in "Text_list_automatic". "Value 0" → "automatic stopped" and → "Value 1" → "automatic started"

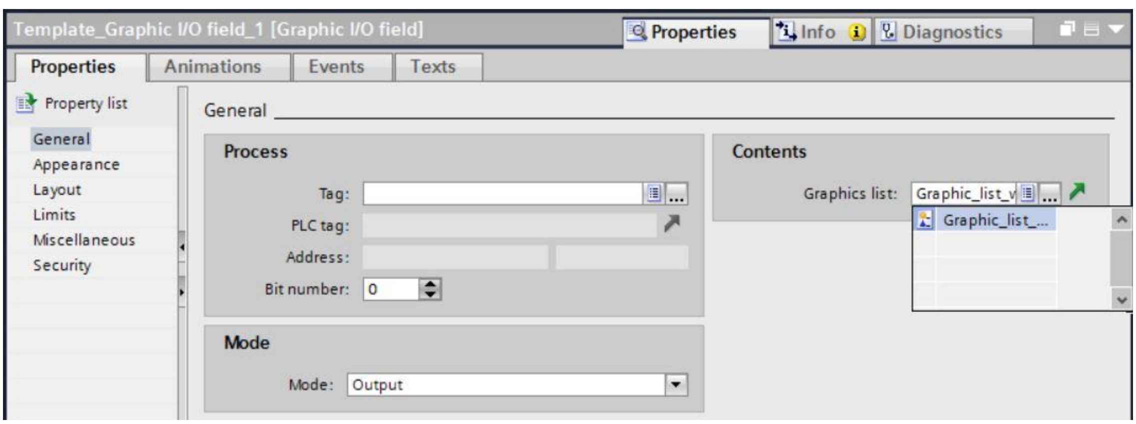


→ Back in "Template_1" for the header, use drag & drop to move the → "Graphic IO field" object from → "Elements" in Toolbox to the upper left corner.

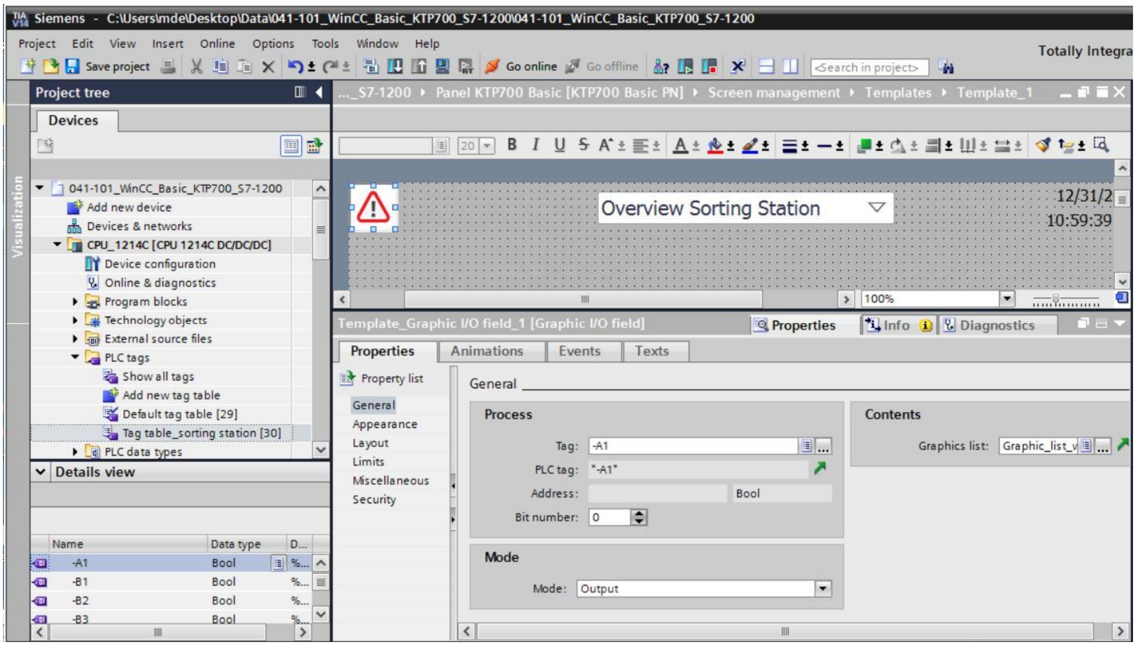


→ Under "General" in "Properties", change "Mode" to → "Output".

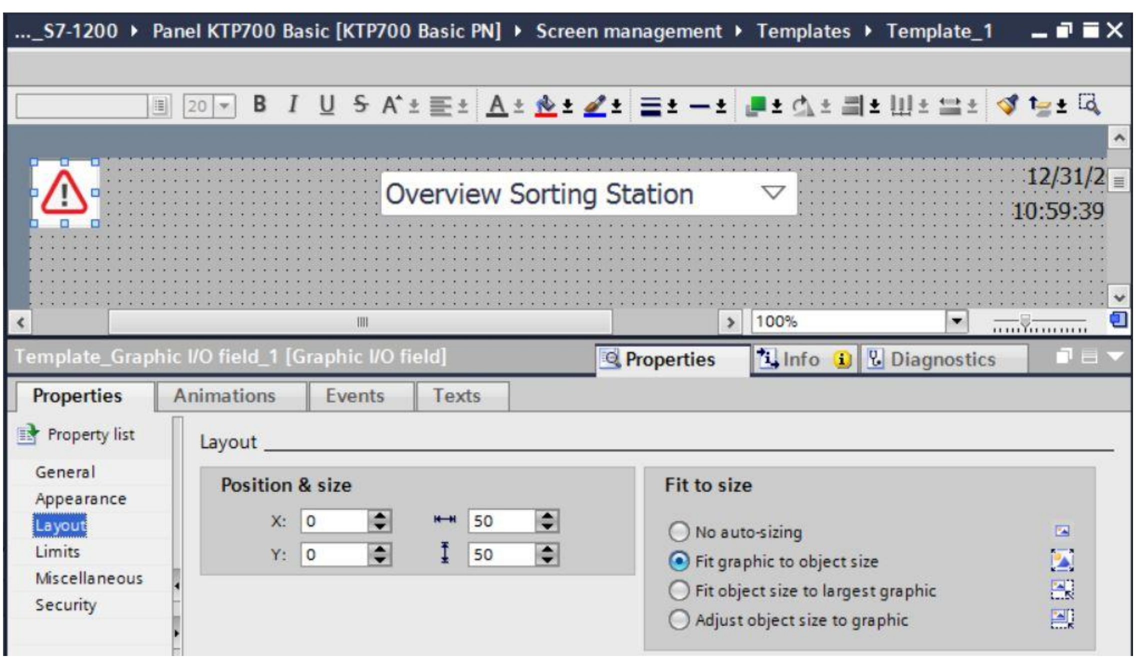
Click the symbol [...] to open the selection dialog for the → "Graphic list" and select the "Graphic_list_warning" you just created.



→ To establish the connection to the global tag in the CPU, select → "PLC tags" and → "Tag table_sorting station" below → "CPU_1214C". Next, move the → "-A1" tag from the "Details view" to the "Tag" field. Also select → "Bit number 0".

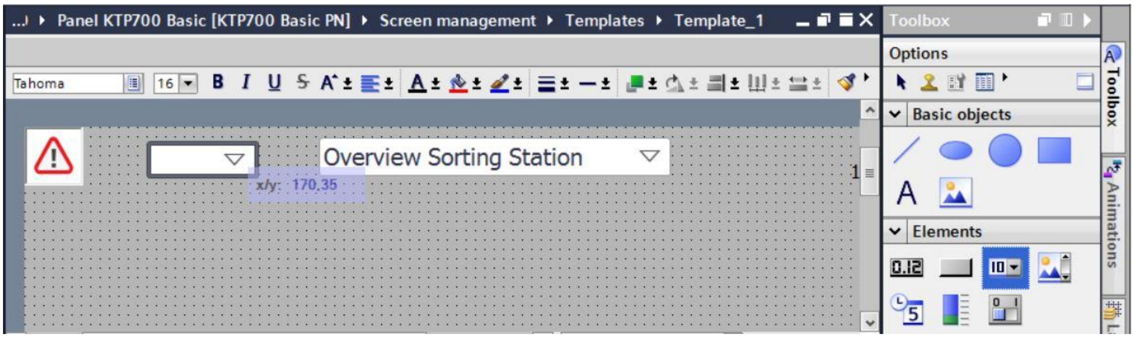


→ Under "Layout" in "Properties", change the size of the "Graphic IO field" under → "Position & size".

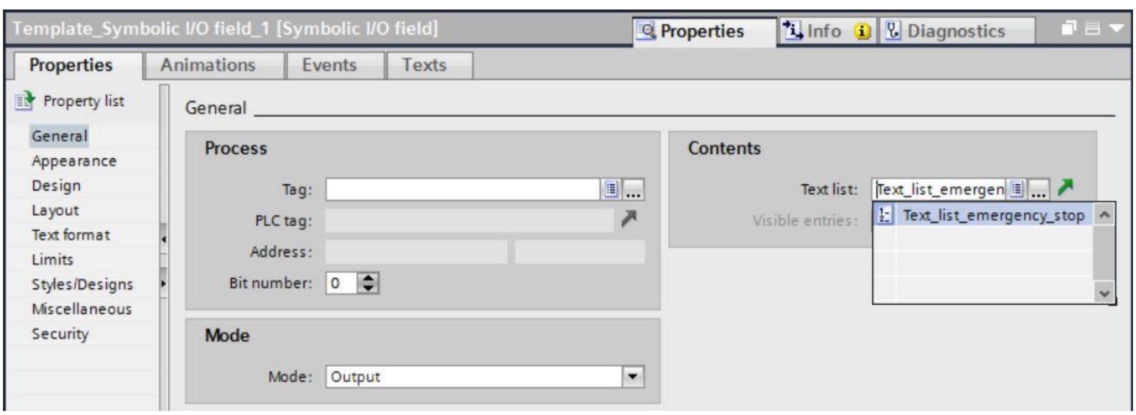


12

→ To display the Emergency Stop status in the header as text, use drag & drop to move the → "Symbolic IO field" object from → "Elements" in Toolbox to the right of the "Graphic IO field".

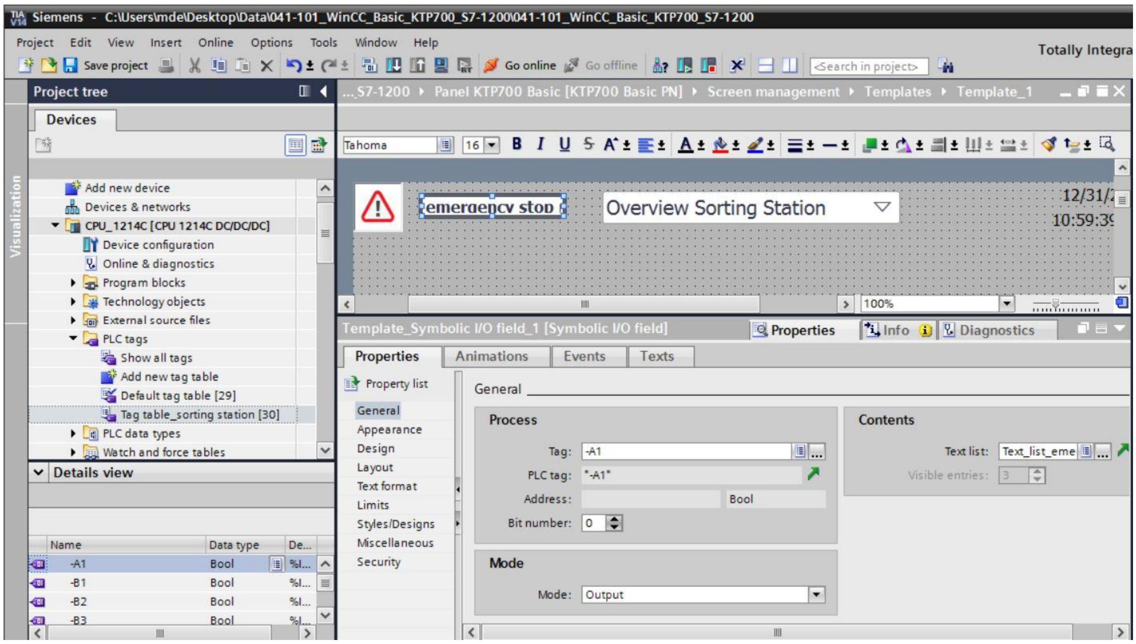


→ Under "General" in "Properties", change "Mode" to → "Output". Click the symbol to open the selection dialog for the → "Text list" and select the "Text_list_emergency_stop" you just created.

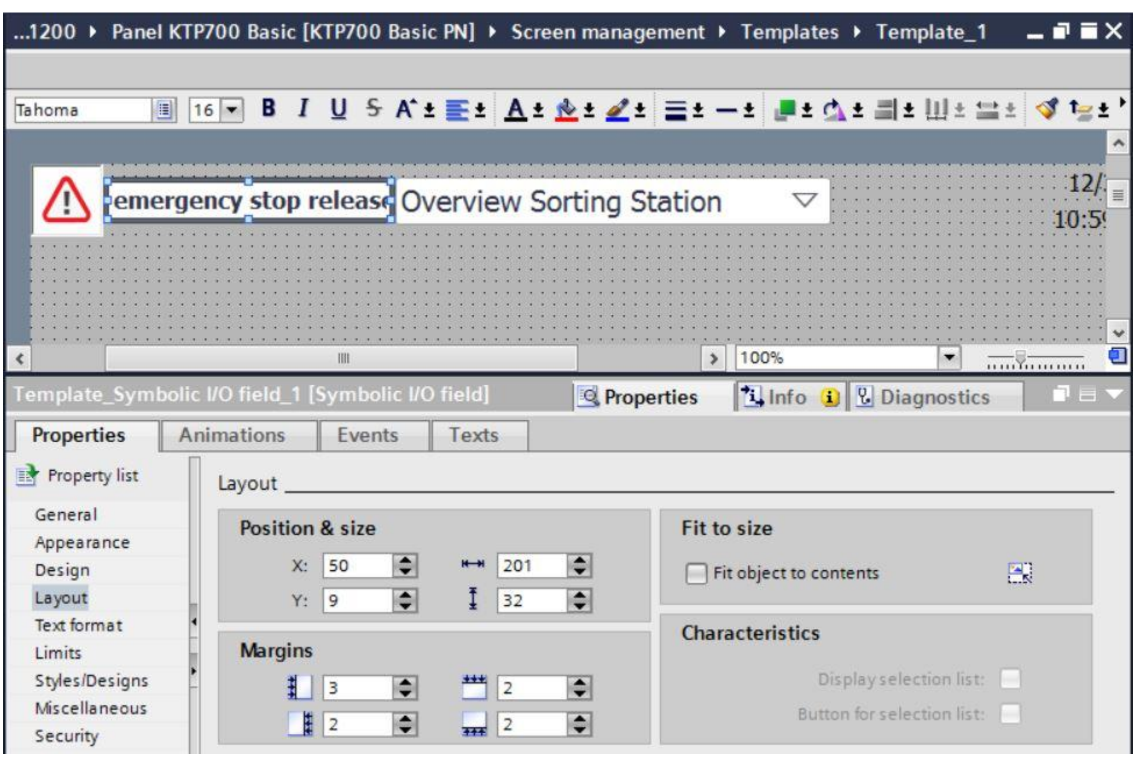


12

→ To establish the connection to the global tag in the CPU, select → "PLC tags" and → "Tag table_sorting station" below → "CPU_1214C". Next, move the → "-A1" tag from the "Details view" to the "Tag" field and also select → "Bit number 0".



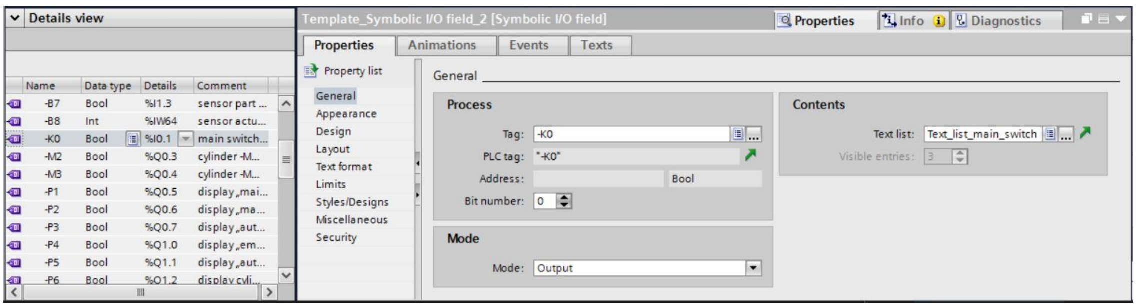
→ Under "Layout" in "Properties", change the size of the "Graphic IO field" under → "Position & size".



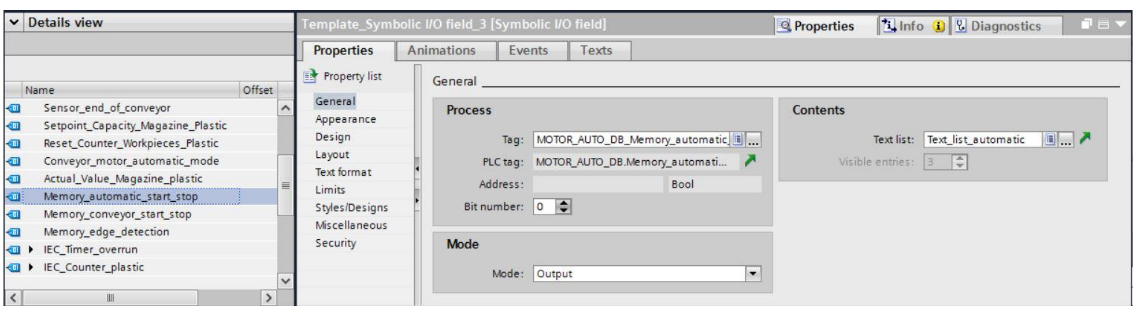
12

→ Repeat the previous steps for the text lists → "Text_list_main_switch" and → "Text list_automatic" to insert them directly one below the other to the left of the date and time. Change the size and font so that the text has enough space.

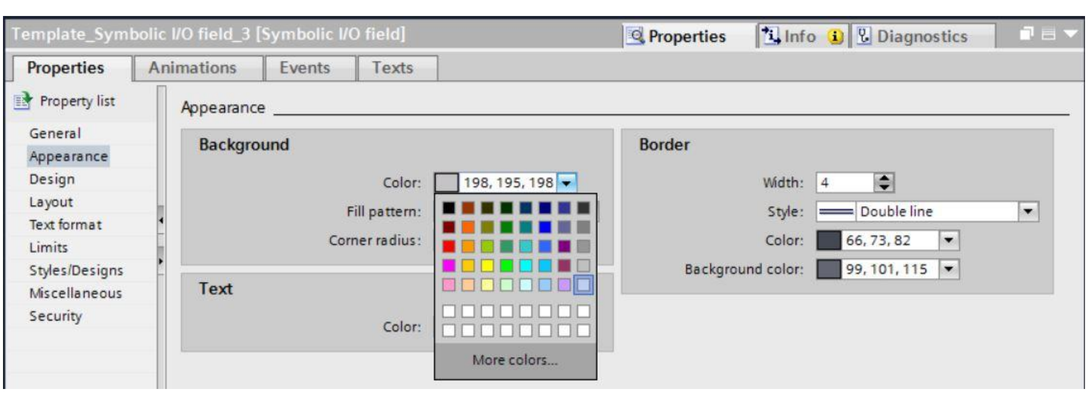
→ The connection of the "Text_list_main_switch" is made using the → "-K0" tag from the "Tag table_sorting station".



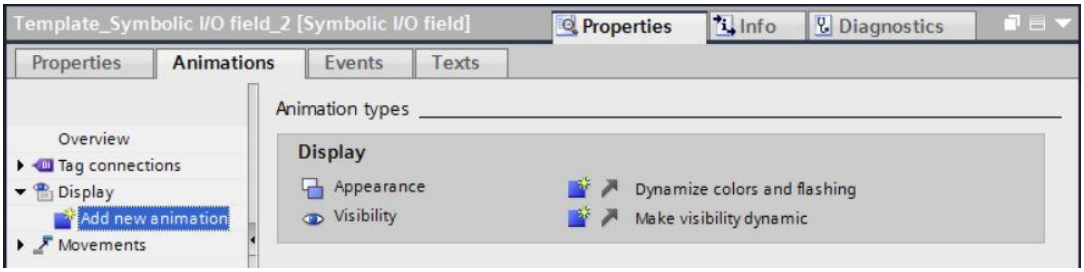
→ The connection of the "Text_list_automatic" is made using the → "Memory_Automatic_Start_Stop" tag from "MOTOR_AUTO_DB1[DB1]".



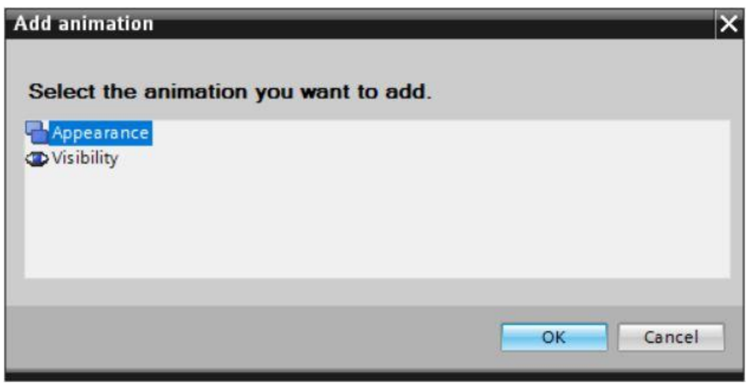
→ Under "Appearance" in "Properties", change the "Color" of "Background" to → "Gray" for → "Text_list_main_switch" and → "Text_list_automatic".



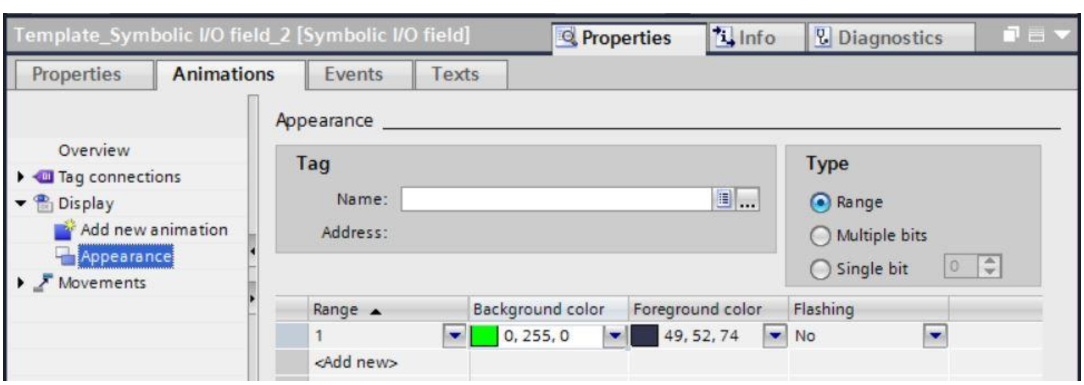
→ Now switch to the "Animations" tab for → "Text_list_main_switch" and → "Text_list_automatic", select "Display" and click → "Add new animation".



→ In the displayed dialog, select → "Appearance" and click → "OK".

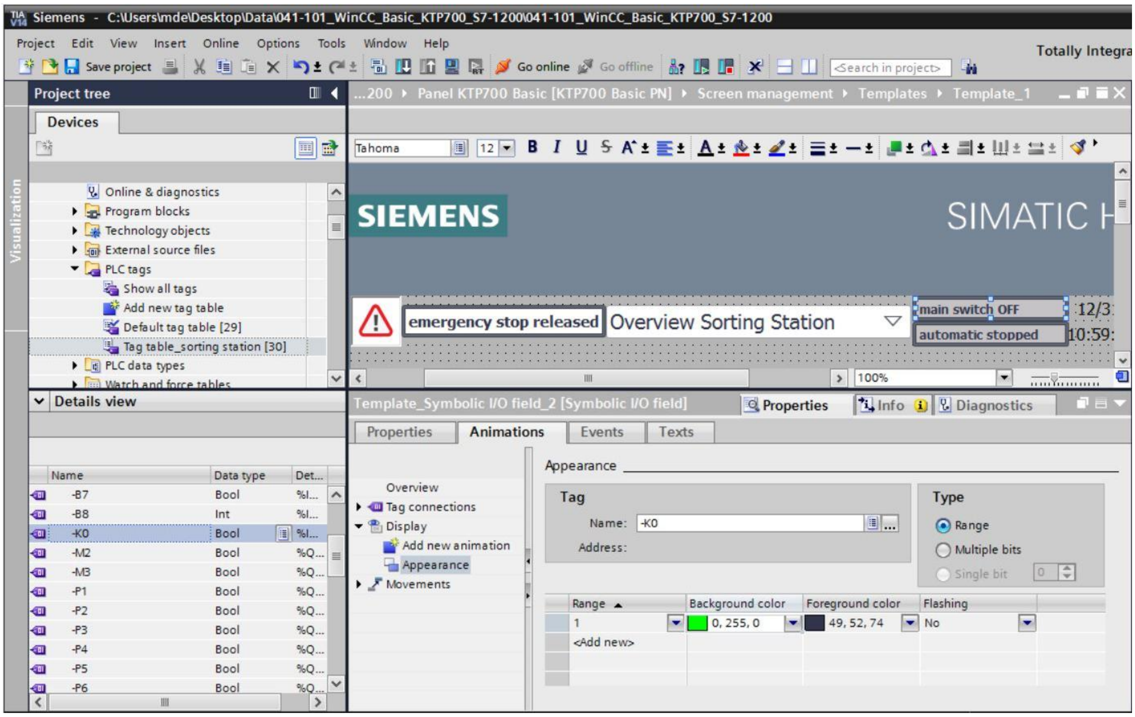


→ Under "Appearance" of both "Symbolic IO fields", add a range with value → "1" (signal state "High") and change the "Background color" to → "Green".

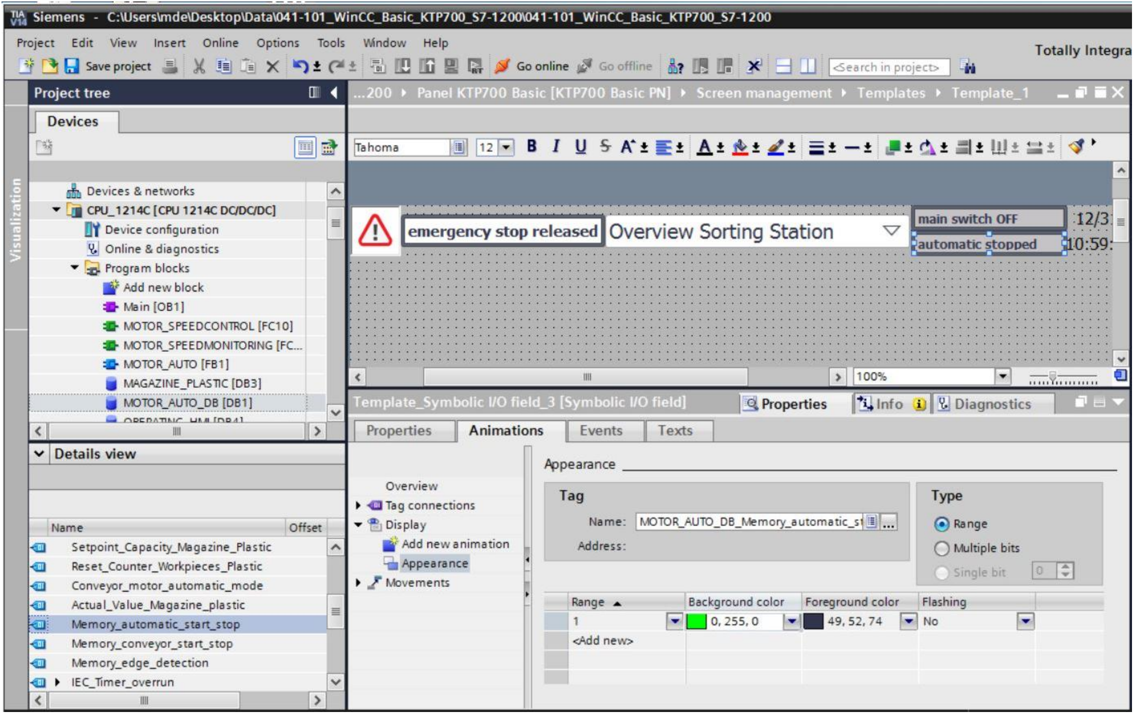


12

→ The connection of the "Text_list_main_switch" is made again using the → "-K0" tag from the "Tag table_sorting station".

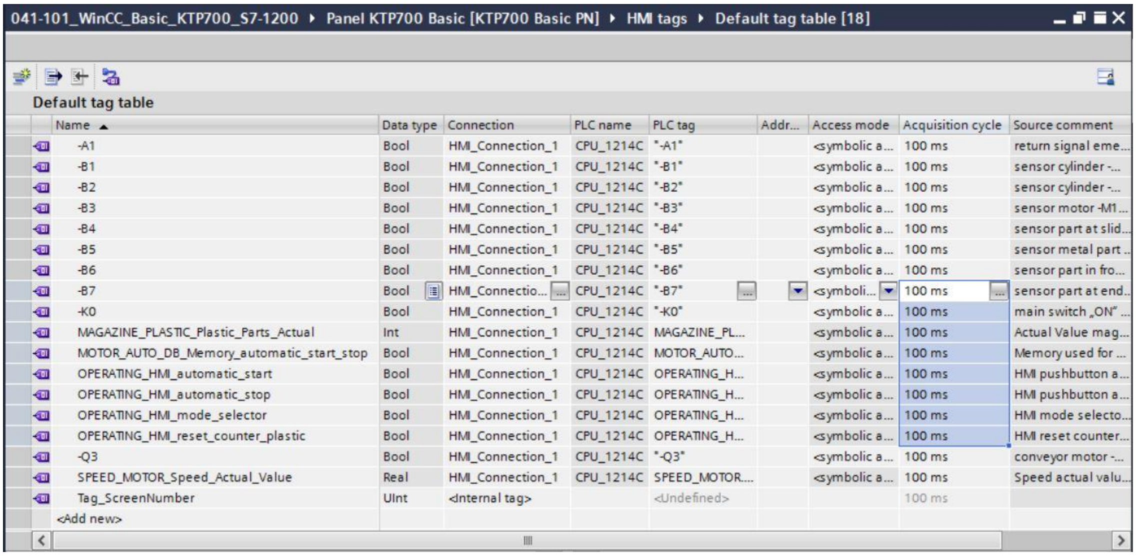


→ The connection of the "Text_list_automatic" is made using the
→ "Memory_Automatic_Start_Stop" tag from "MOTOR_AUTO_DB1[DB1]".



12

→ The "Acquisition cycle" of all tags is also to be accelerated from 1 second to 100 milliseconds in the default tag table.


041-101_WinCC_Basic_KTP700_S7-1200 ▸ Panel KTP700 Basic [KTP700 Basic PN] ▸ HMI tags ▸ Default tag table [18]

Default tag table

Name	Data type	Connection	PLC name	PLC tag	Addr...	Access mode	Acquisition cycle	Source comment
-A1	Bool	HMI_Connection_1	CPU_1214C	"-A1"		<symbolic a...	100 ms	return signal eme...
-B1	Bool	HMI_Connection_1	CPU_1214C	"-B1"		<symbolic a...	100 ms	sensor cylinder -...
-B2	Bool	HMI_Connection_1	CPU_1214C	"-B2"		<symbolic a...	100 ms	sensor cylinder -...
-B3	Bool	HMI_Connection_1	CPU_1214C	"-B3"		<symbolic a...	100 ms	sensor motor -M1...
-B4	Bool	HMI_Connection_1	CPU_1214C	"-B4"		<symbolic a...	100 ms	sensor part at slid...
-B5	Bool	HMI_Connection_1	CPU_1214C	"-B5"		<symbolic a...	100 ms	sensor metal part ...
-B6	Bool	HMI_Connection_1	CPU_1214C	"-B6"		<symbolic a...	100 ms	sensor part in fro...
-B7	Bool	HMI_Connectio...	CPU_1214C	"-B7"		<symboli...	100 ms	sensor part at end...
-K0	Bool	HMI_Connection_1	CPU_1214C	"-K0"		<symbolic a...	100 ms	main switch „ON" ...
MAGAZINE_PLASTIC_Plastic_Parts_Actual	Int	HMI_Connection_1	CPU_1214C	MAGAZINE_PL...		<symbolic a...	100 ms	Actual Value mag...
MOTOR_AUTO_DB_Memory_automatic_start_stop	Bool	HMI_Connection_1	CPU_1214C	MOTOR_AUTO...		<symbolic a...	100 ms	Memory used for ...
OPERATING_HMI_automatic_start	Bool	HMI_Connection_1	CPU_1214C	OPERATING_H...		<symbolic a...	100 ms	HMI pushbutton a...
OPERATING_HMI_automatic_stop	Bool	HMI_Connection_1	CPU_1214C	OPERATING_H...		<symbolic a...	100 ms	HMI pushbutton a...
OPERATING_HMI_mode_selector	Bool	HMI_Connection_1	CPU_1214C	OPERATING_H...		<symbolic a...	100 ms	HMI mode selecto...
OPERATING_HMI_reset_counter_plastic	Bool	HMI_Connection_1	CPU_1214C	OPERATING_H...		<symbolic a...	100 ms	HMI reset counter...
-Q3	Bool	HMI_Connection_1	CPU_1214C	"-Q3"		<symbolic a...	100 ms	conveyor motor -...
SPEED_MOTOR_Speed_Actual_Value	Real	HMI_Connection_1	CPU_1214C	SPEED_MOTOR....		<symbolic a...	100 ms	Speed actual valu...
Tag_ScreenNumber	UInt	<Internal tag>		<Undefined>			100 ms	
<Add new>								

→ Before the visualization is downloaded to the panel, compile the CPU and panel again and save the project. (→ CPU_1214C → → Panel KTP700 Basic → → Save project)

→ After successful compilation, the entire controller with the created program including the hardware configuration, as previously described in earlier modules, can be downloaded.

(→)

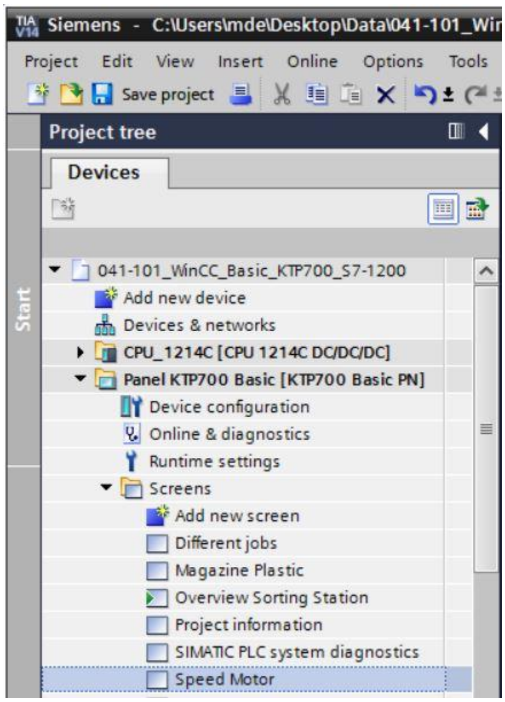
→ To download the visualization to the panel, follow the same procedure. Select the → "Panel KTP700 Basic [KTP700 Basic]" folder and click the

→ "Download to device" button.

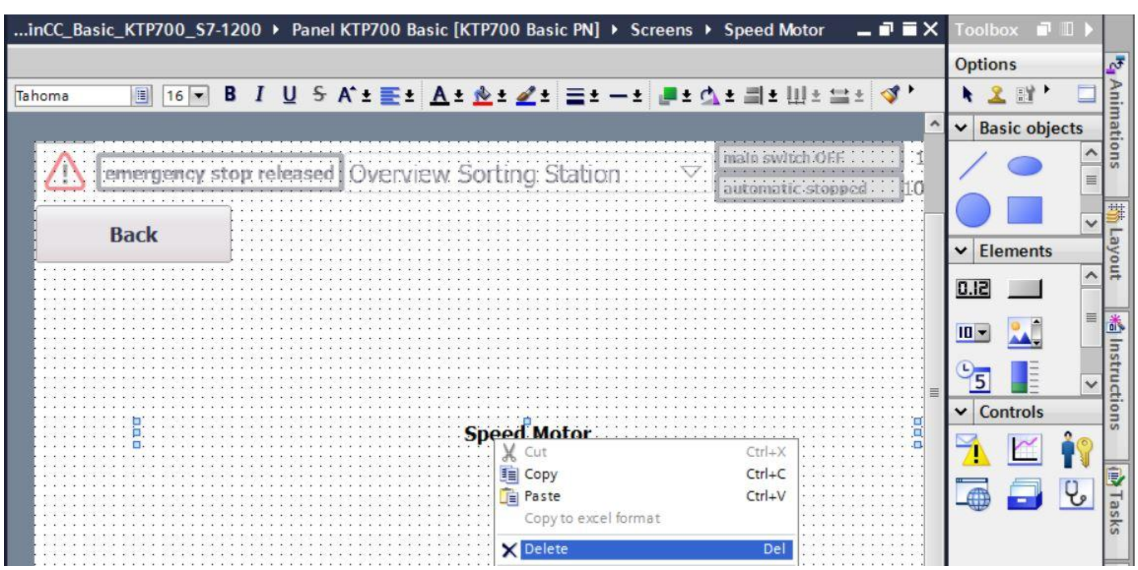
12

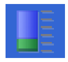
7.14 Bar graph display

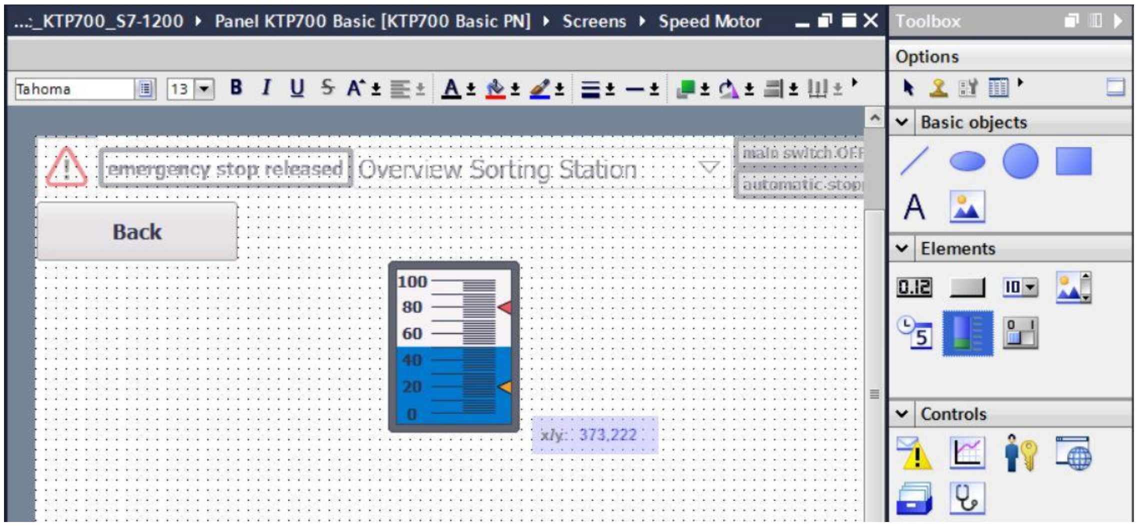
→ Next, the setpoint is to be specified for the motor speed control and the actual value is to be displayed. To do this, open the → "Speed Motor" screen with a double-click.



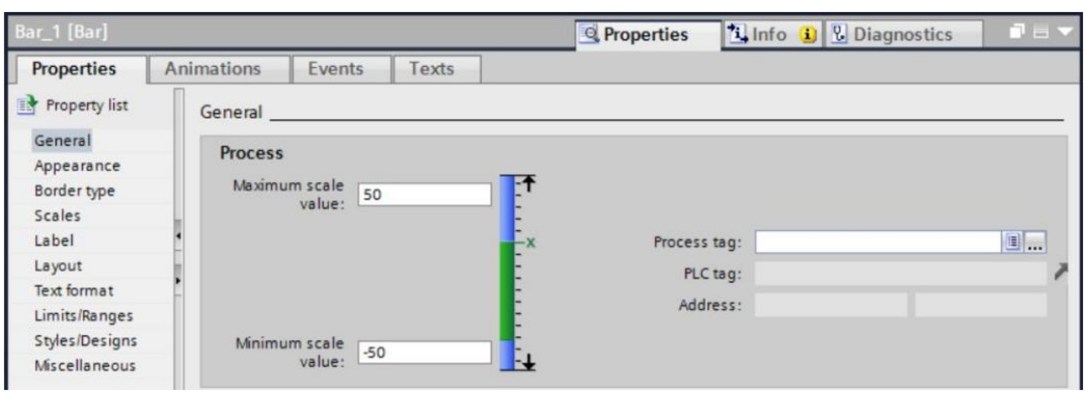
→ The text box in the center of the screen is to be removed by right-clicking on it and selecting → "Delete" in the displayed dialog.



→ To display the actual speed value graphically, use drag & drop to move the → "Bar graph" object  from → "Elements" in Toolbox to the middle of the screen.

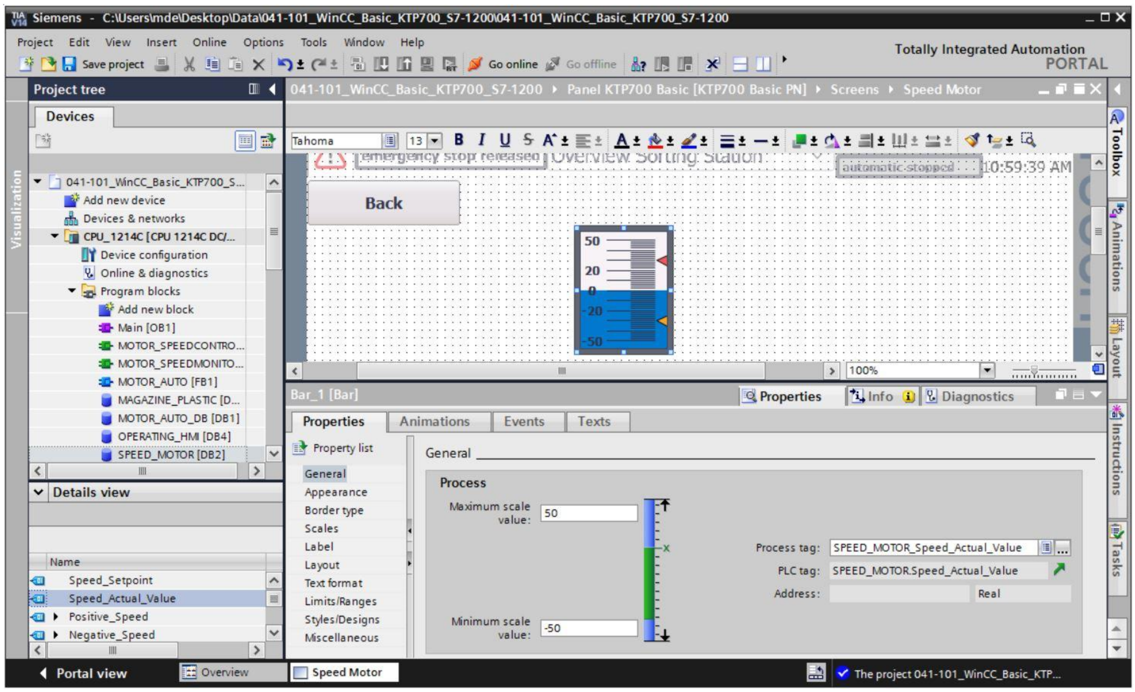


→ Under "General" in "Properties", change "Maximum scale value" to → 50 and "Minimum scale value" to → -50.

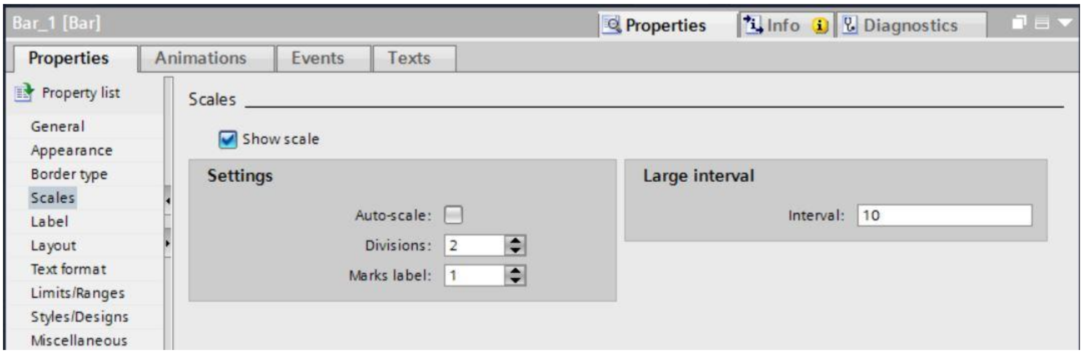


12

→ For the process connection, select → "Program blocks" and the → "SPEED_MOTOR[DB2]" data block below → "CPU_1214C". Next, drag the → "Speed_Actual_Value" tag from the → Details view to the "Process tag" field.

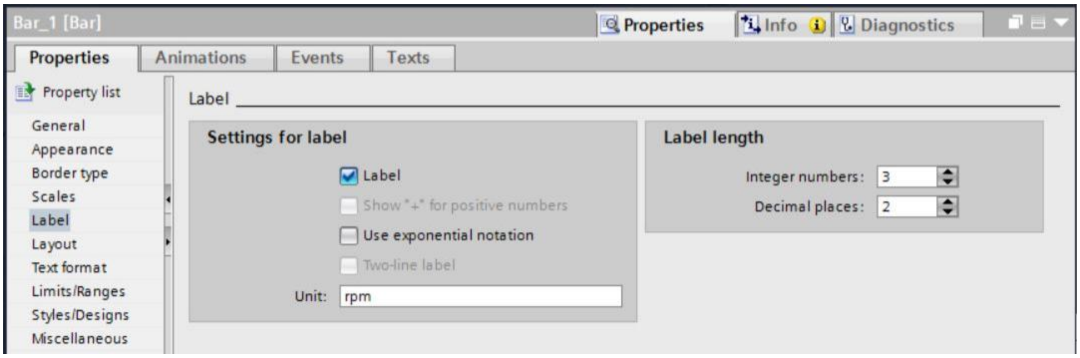


→ Under "Scales" in "Properties", select → ☑ "Show scale" and set "Divisions" → 2, "Marks label" → 1 and "Interval" → 10.

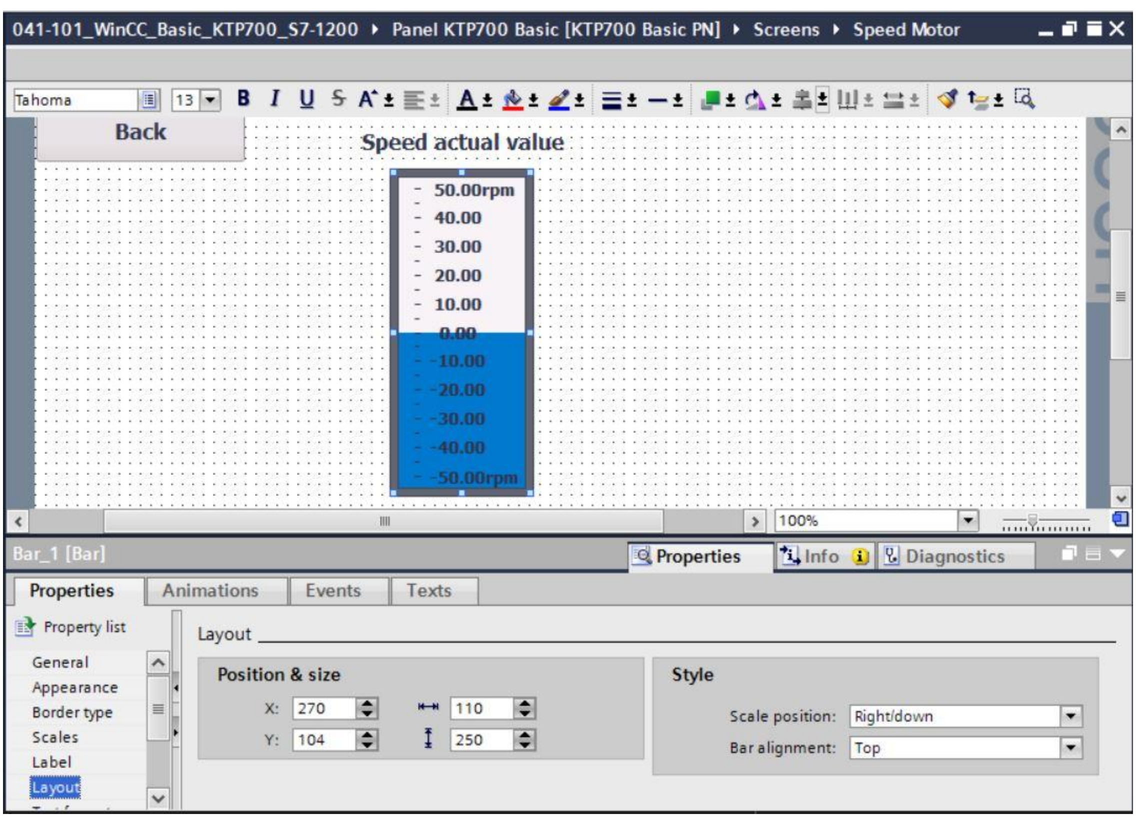


12

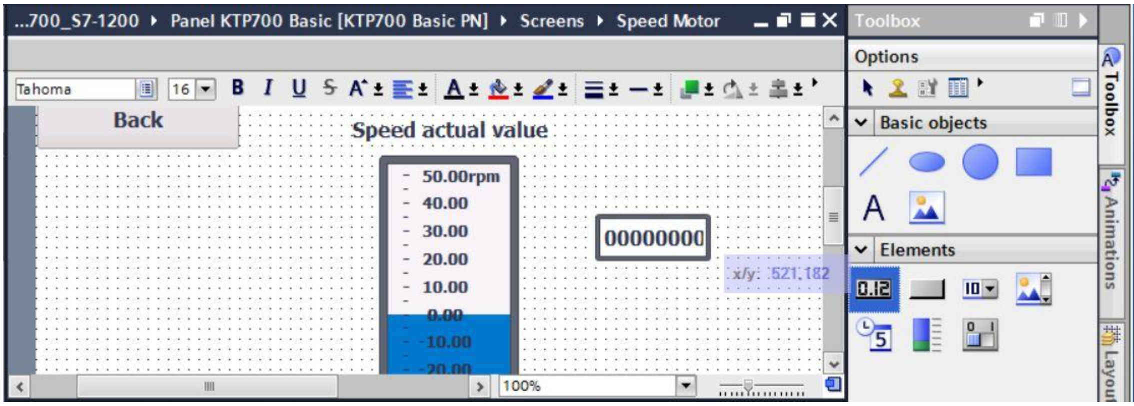
→ Under "Label" in "Properties", select → ☑ "Label" and set "Unit"→ rpm and "Decimal places" → 2.



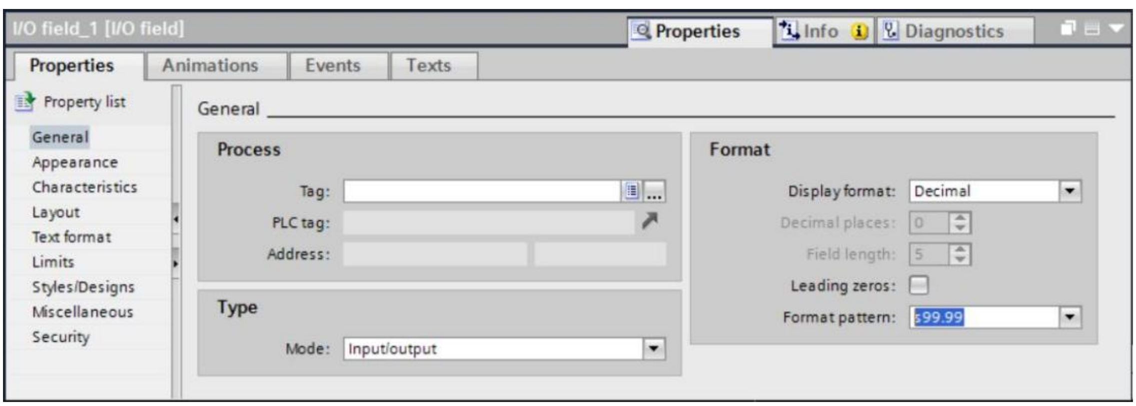
→ Under "Layout" in "Properties", change the position and size of the bar graph under → "Position & size". Above the bar graph, insert a → "Text field" A with text → "Speed actual value" for the description.



→ To enable the speed setpoint to be specified, use drag & drop to move the → "IO field" object  from → "Elements" in Toolbox to the right of the bar graph.

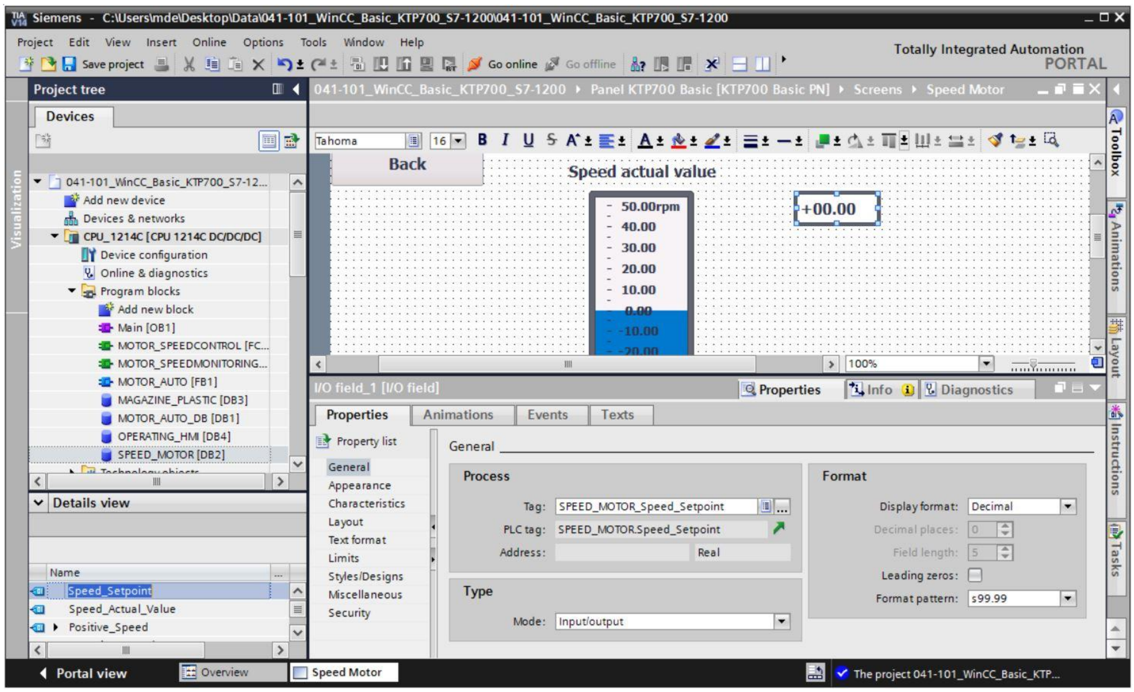


→ Under "General" in "Properties", keep "Type" → "Input/Output" and change the "Format pattern" to → s99.99.

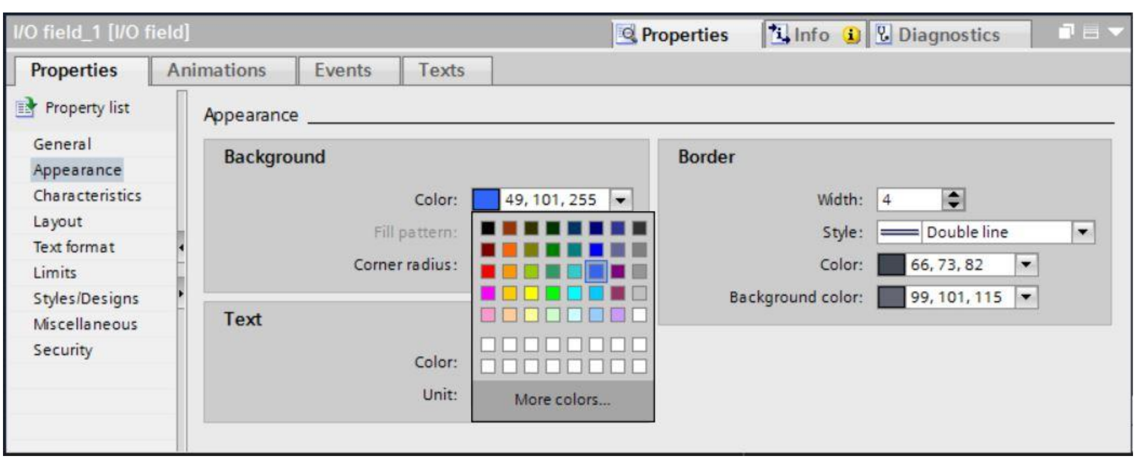


→ For the process connection, select → "Program blocks" and the → "SPEED_MOTOR[DB2]" data block below → "CPU_1214C".

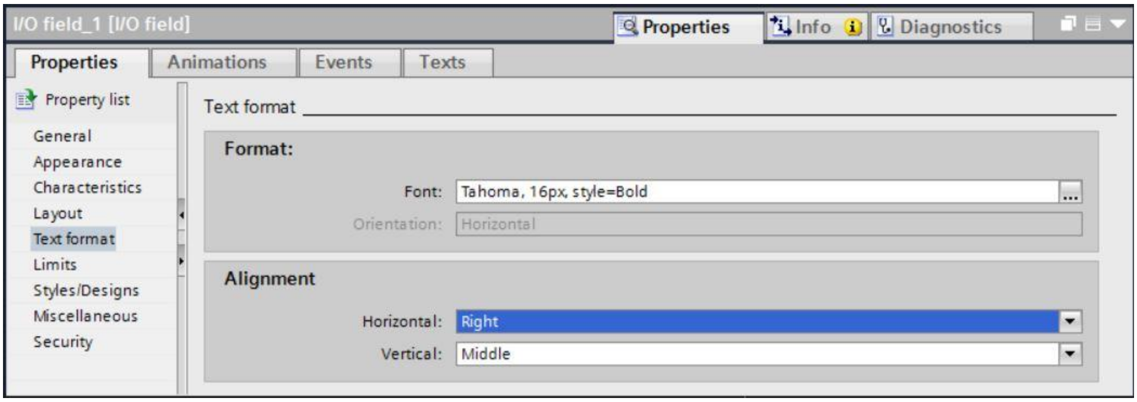
Next, drag the → "Speed_Setpoint" tag from the → Details view to the "Tag" field.



→ Under "Appearance" in "Properties", change the "Color" of "Background" to → "Blue".

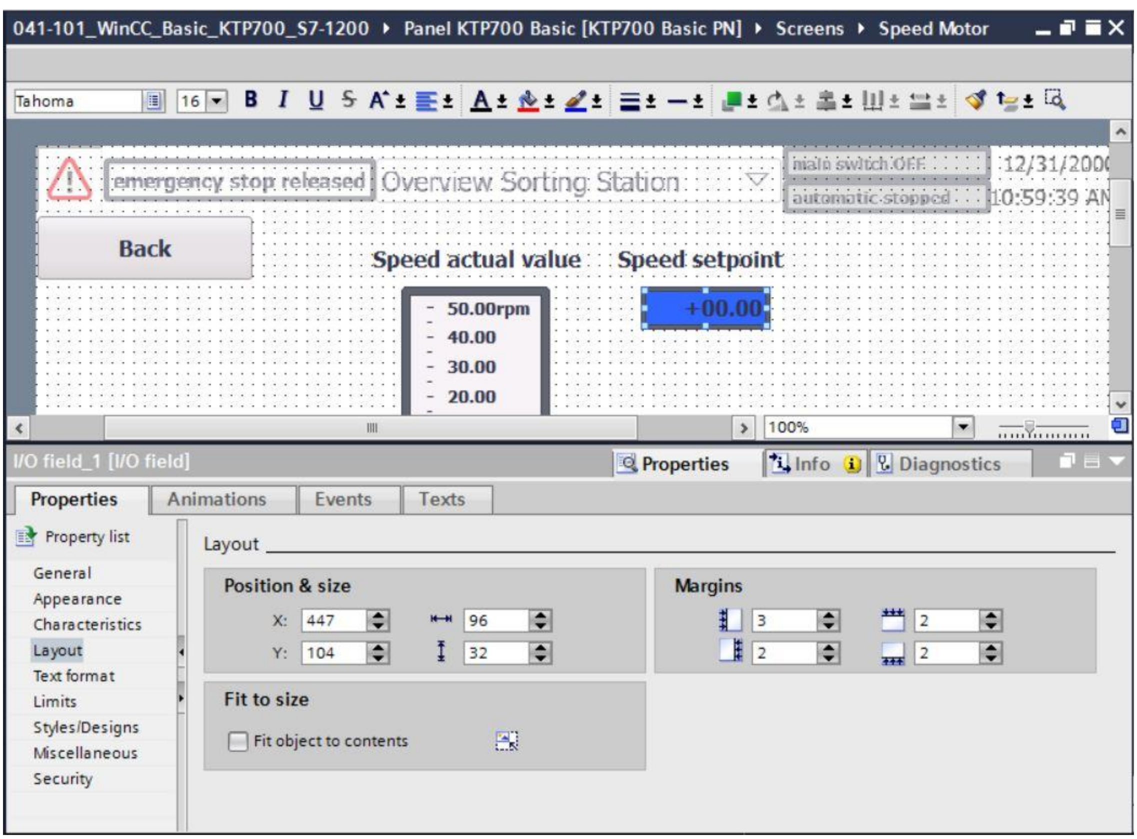


→ Under "Text format" in "Properties", change "Alignment" "Horizontal" to → "Right".



→ Under "Layout" in "Properties", change the position and size of the IO field under → "Position & size".

→ Above the bar graph, insert a → "Text field"  with text → "Speed setpoint" for the description.



12

→ The "Acquisition cycle" of the newly create tag is also to be changed from 1 second to 100 milliseconds in the default tag table.

→ Before the visualization is downloaded to the panel, compile the CPU and panel again and save the project. (→ Panel KTP700 Basic → → Save project)

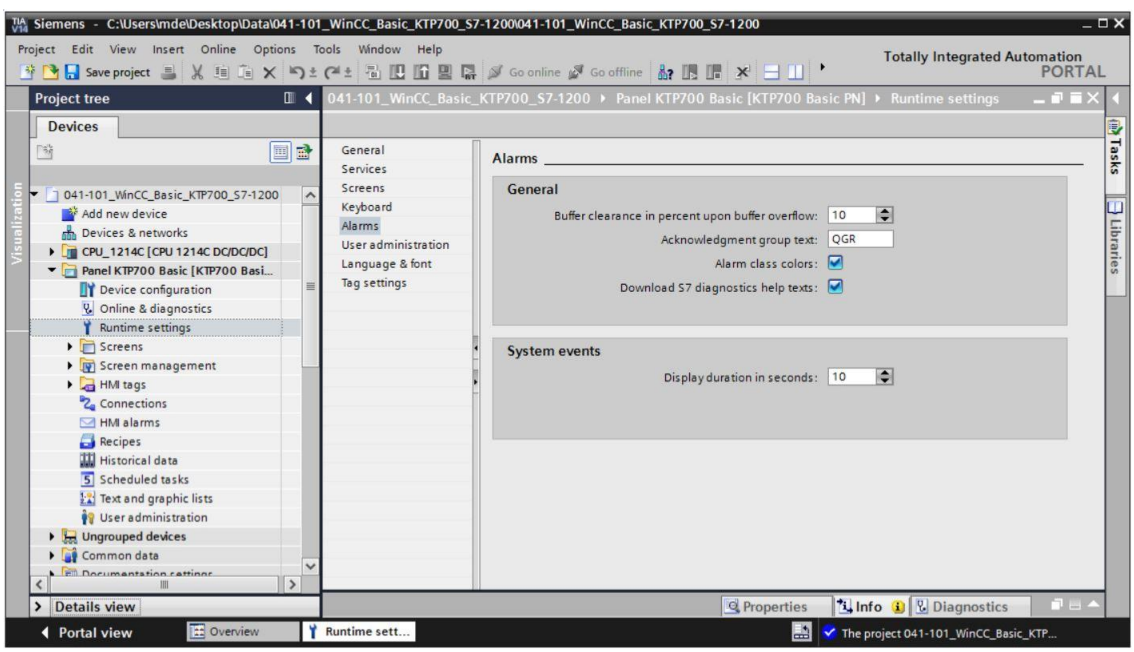
→ To download the visualization to the panel, select the → "Panel KTP700 Basic [KTP700 Basic]" folder and click the → "Download to device" button.

7.15 Messages

A couple of alarm windows were already created when you used the wizard to create the Panel KTP700 Basic. We will now take a closer look at them.

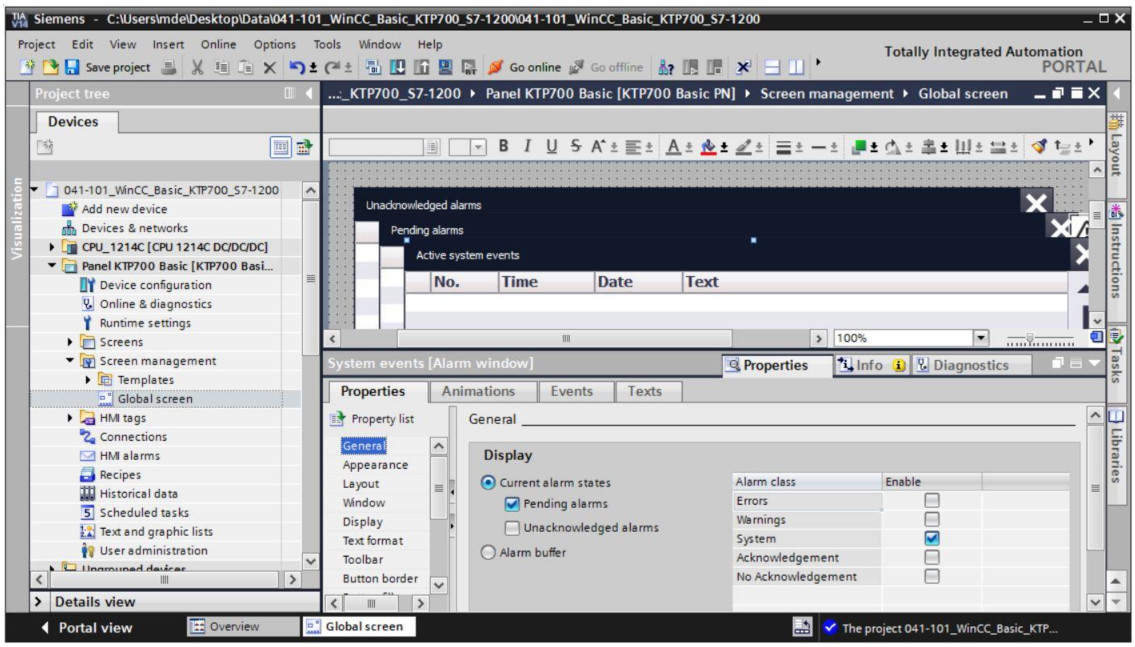
7.15.1 General alarm settings

→ The first step is to make several settings for the display of alarms in Runtime. To do this, double-click the → "Runtime settings" folder in → "Panel KTP700 Basic". Under "General" in "Alarms", select → "Alarm class colors", and under "System events", change → Display duration in seconds to "10".



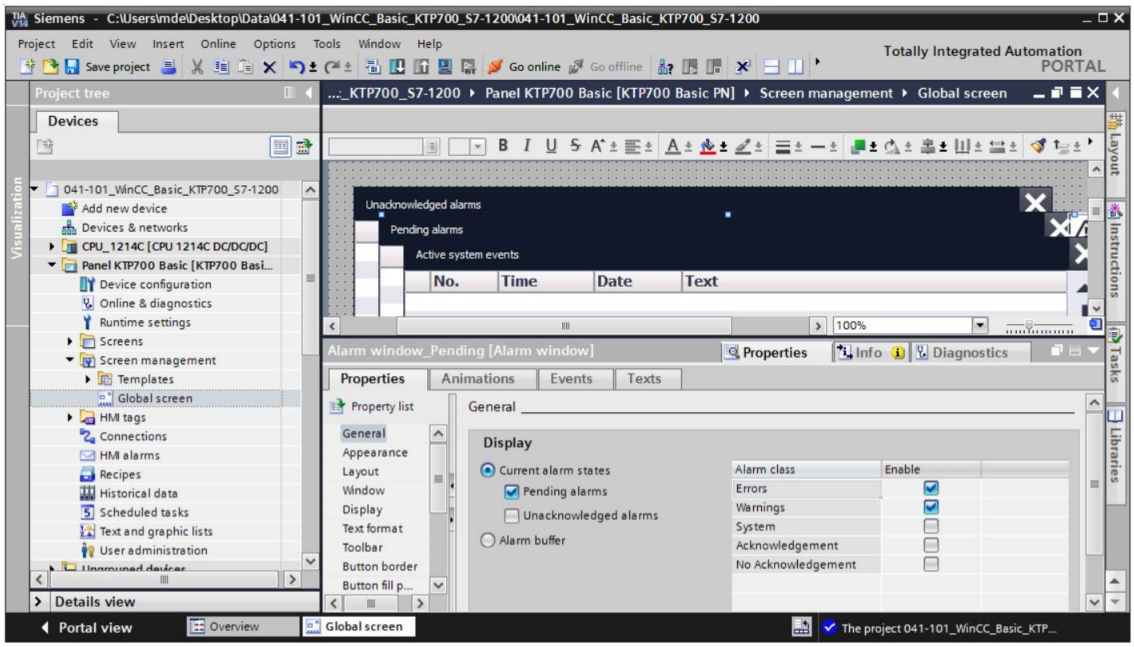
7.15.2 Alarm window

→ For the alarm windows to be displayed in the foreground in every screen, a → "Global screen" is available in the → "Screen management" folder of → "Panel KTP700 Basic". Open this with a double-click. Three alarm windows have already been created in this screen. In the first alarm window→ "System events", ☑ "Pending alarms" and the alarm class ☑ "System" are already selected under "General" in "Properties".



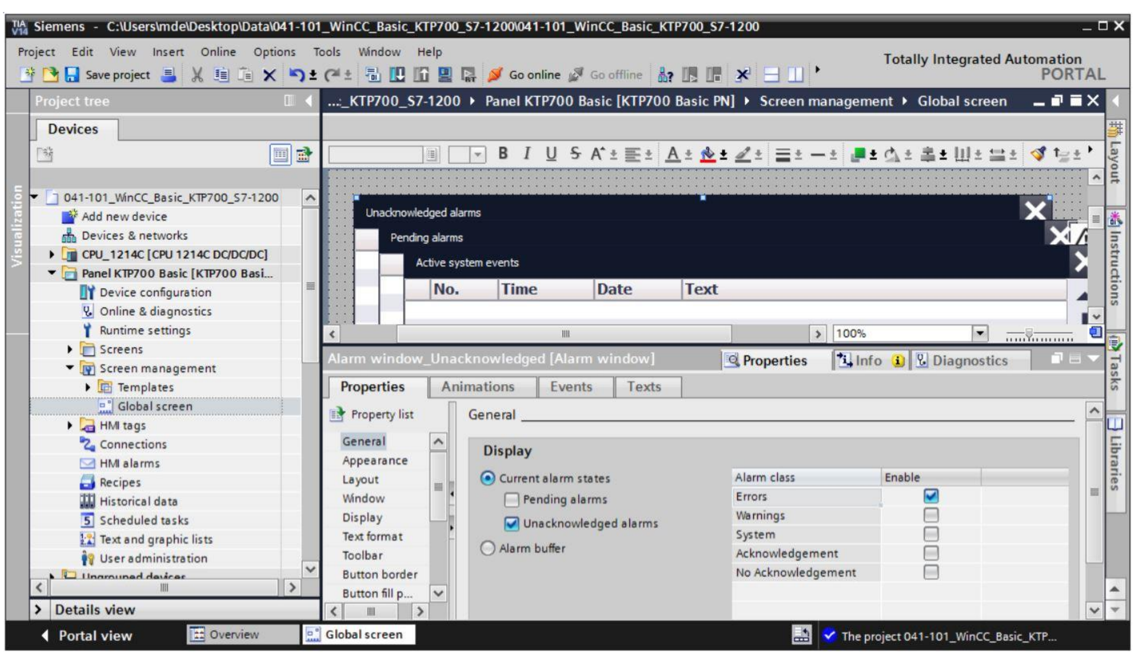
Note: System events will be automatically displayed in Runtime for ten seconds.

→ The → "Pending alarms" alarm window is the second alarm window in the "Global screen" screen. Select ☑ "Pending alarms" under "General" in "Properties". Select ☑ "Errors" and ☑ "Warnings" as alarm classes.



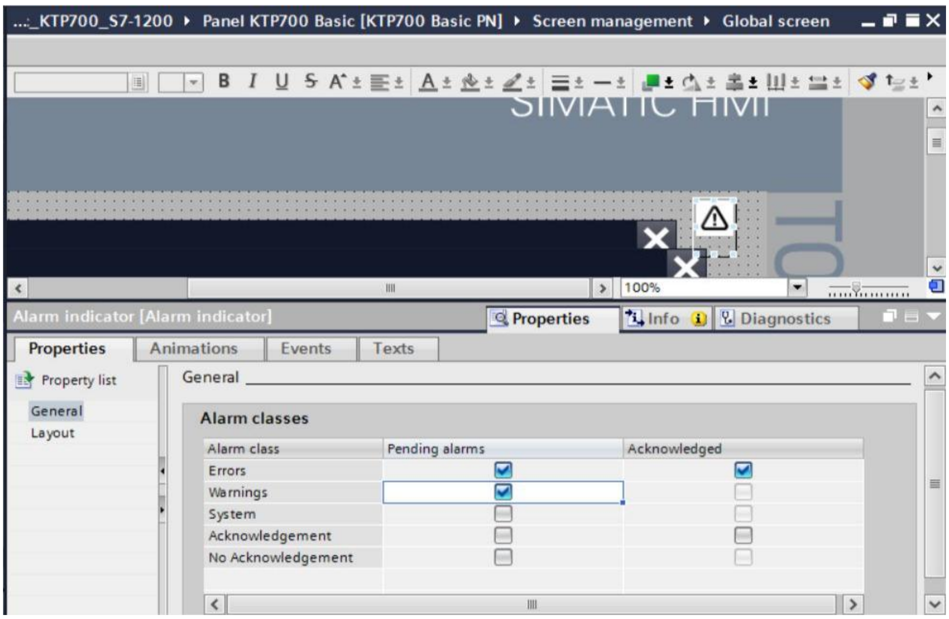
Note: You will create alarm classes of types "Errors" and "Warnings" in the panel itself in subsequent steps.

→ The → "Unacknowledged alarms" alarm window is the third alarm window in the "Global screen" screen. Select ☑ "Unacknowledged alarms" under "General" in "Properties". Select only ☑ "Errors" here as the alarm classes.

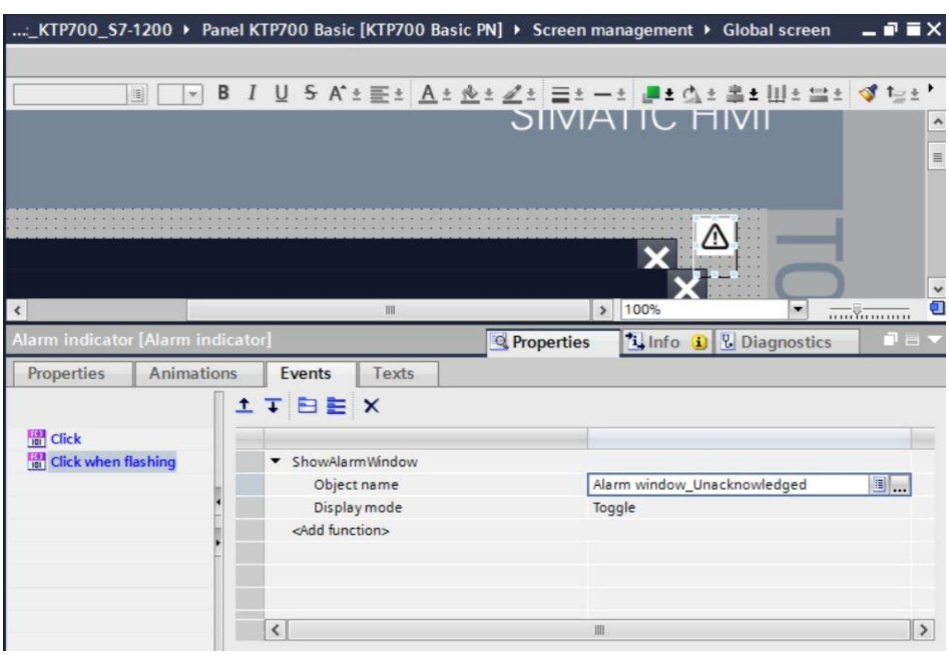


7.15.3 Alarm indicator

→ In addition to the alarm windows, the "Global screen" screen also contains an → "Alarm indicator". Its purpose is to display an alarm window again, which the user hid by clicking it away. Under "General" in "Properties", select ☑ "Errors: Pending messages", ☑ "Errors: Acknowledged" and ☑ "Warnings: Pending alarms" as the message classes.



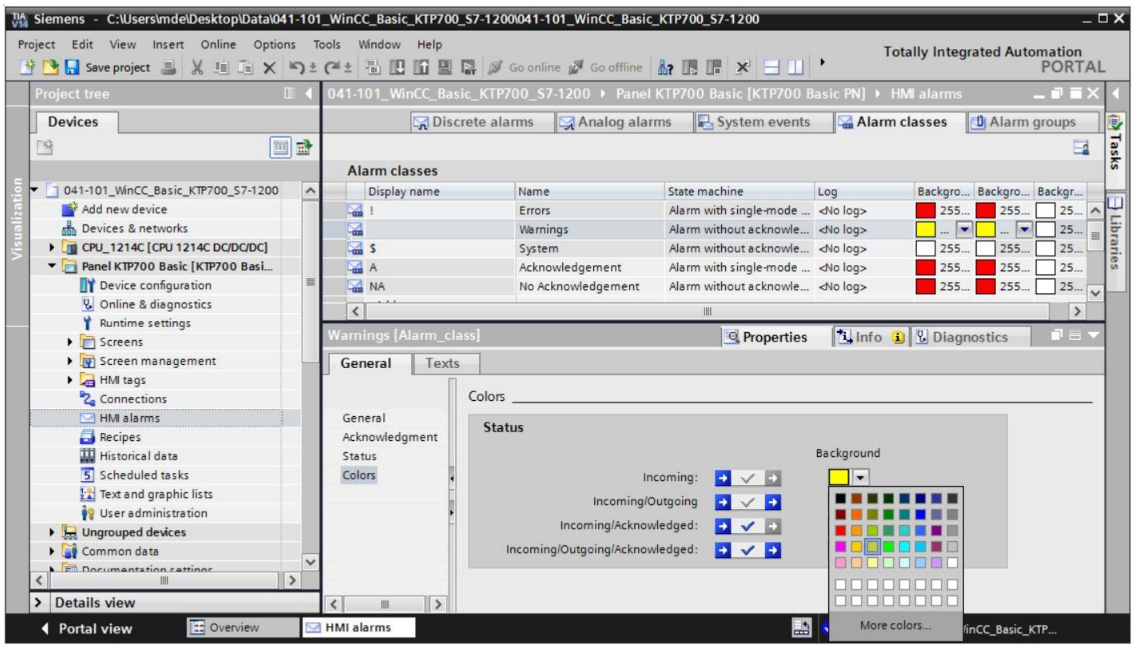
→ Under → "Events", the "ShowAlarmWindow" function is already entered for the "Click" event. Change the → "Object name" for "Click when flashing" to "Alarm window_Unacknowledged" so that this alarm window will be opened here.



12

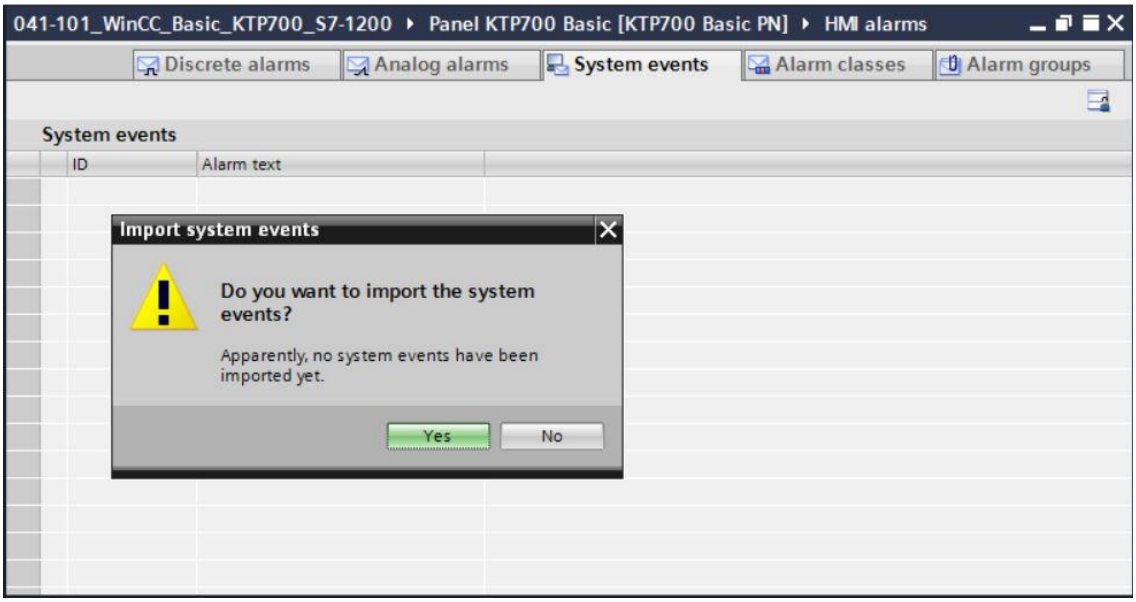
7.15.4 Settings of alarm classes

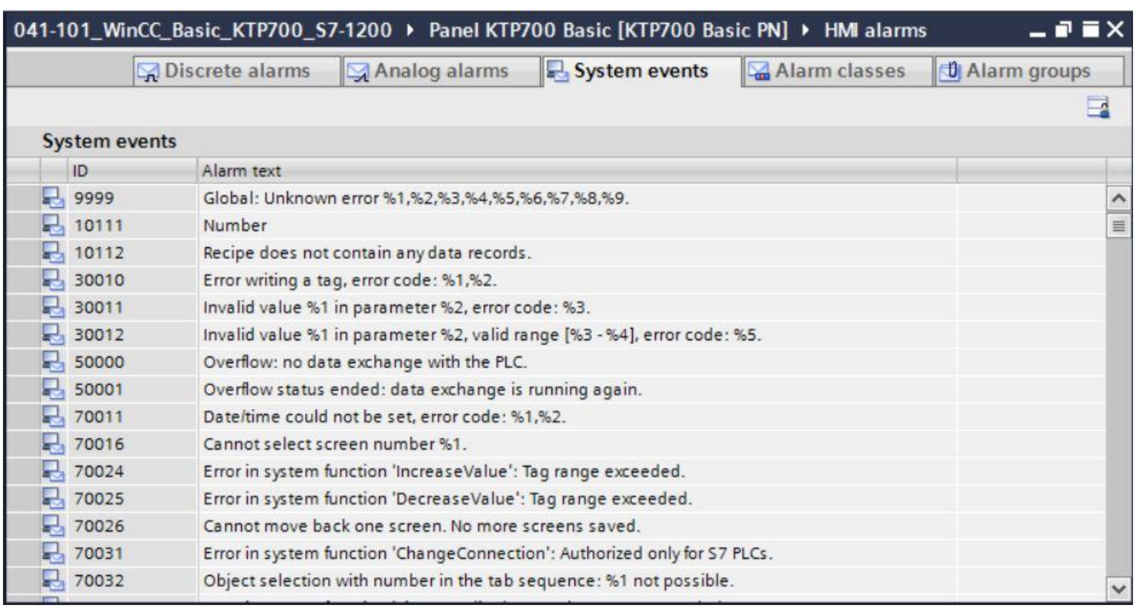
→ The → "HMI alarms" menu item is available in → "Panel KTP700 Basic" for configuration of the alarm system and creation of customized alarms. Open this with a double-click. The alarm classes to be used are already created in the "Alarm classes" menu item. These can still be changed, however. For the → "Warnings" alarm class, change the background color for the "Incoming" and "Incoming/Outgoing" statuses to → "Yellow".



7.15.5 System events

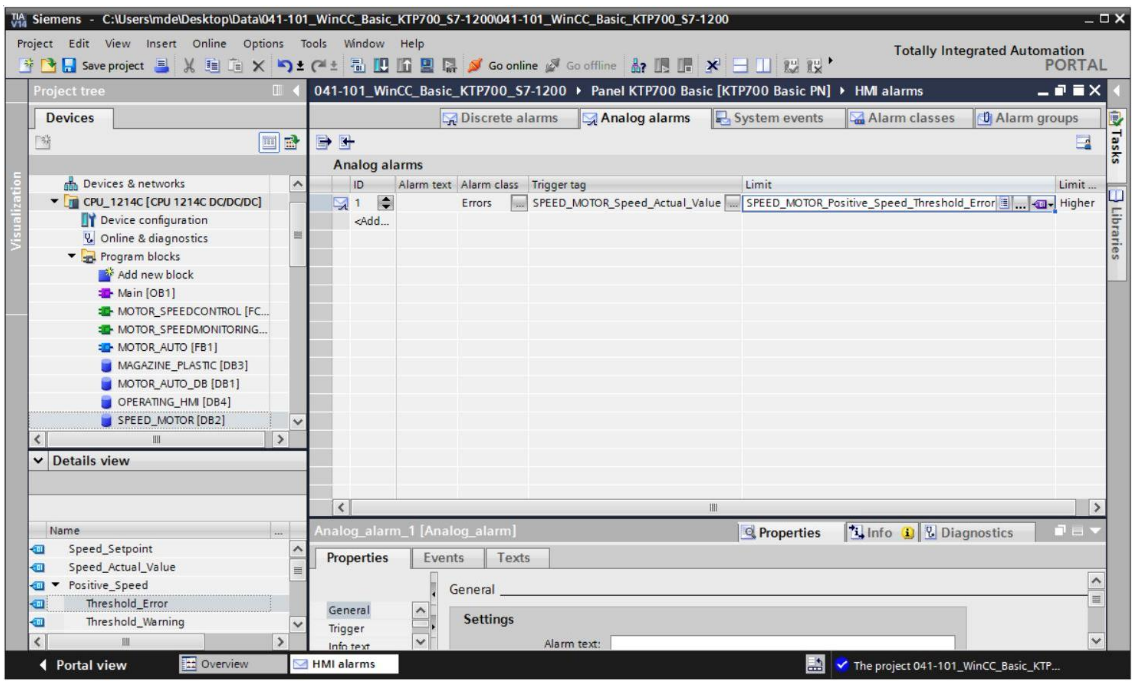
→ You can have system events automatically imported in the "System events" menu item by clicking → "Yes".



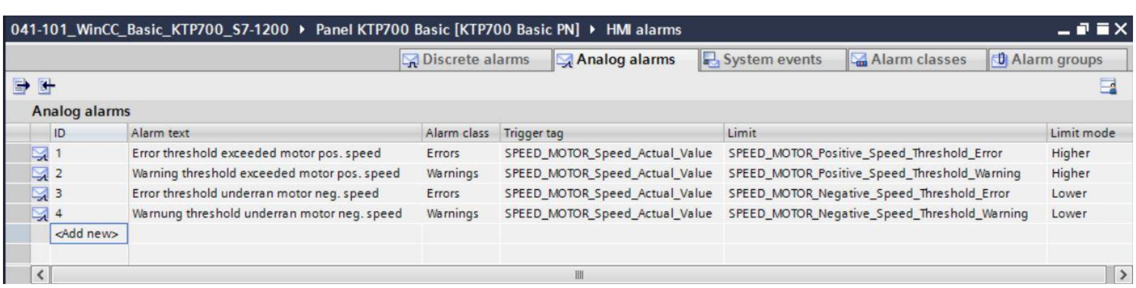


7.15.6 Analog alarms

→ Tags can be monitored for limits in "Analog alarms". Click "Add" to create a new alarm. For monitoring, select the → "SPEED_MOTOR [DB2]" data block in → "CPU_1214C" and drag the → "Speed_Actual_Value" tag from the → "Details view" to the "Trigger tag" field. Next, drag the → "Positive_Speed_Threshold_Error" tag from the → Details view to the "Limit" field.



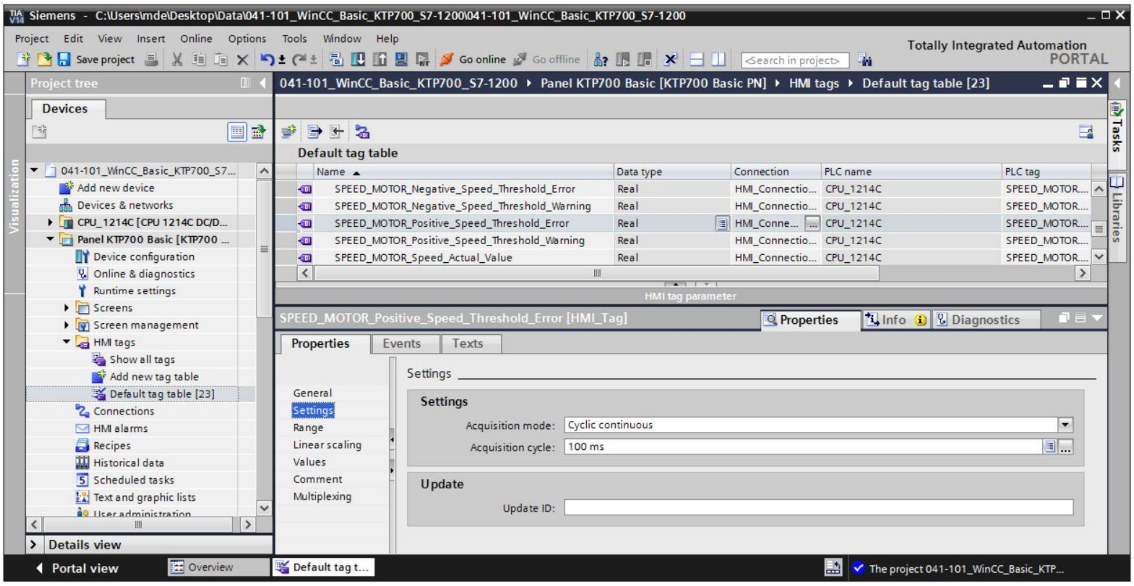
→ Enter the text → "Error threshold exceeded motor pos. speed" in the "Alarm text" column, select the "Alarm class" → "Errors" and → "Higher" for mode. Follow the same procedure to create the three other alarms of alarm classes "Warnings" and "Errors" shown below.



→ The tags relevant for the alarm system must be continuously updated cyclically.
For this, open the → "Default tag table" of the panel and select the "SPEED_MOTOR_Positive_Speed_Threshold_Error" tag.

You can change the → "Acquisition mode" under "Settings" in "Properties" to → "Cyclic continuous". Follow the same procedure to change and check the following tags: "SPEED_MOTOR_Speed_Actual_Value",

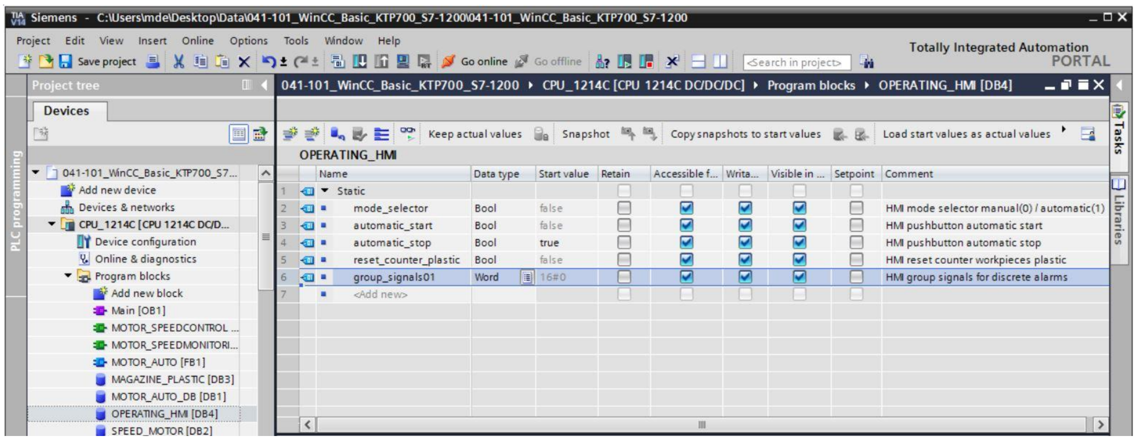
"SPEED_MOTOR_Positive_Speed_Threshold_Warning",
"SPEED_MOTOR_Negative_Speed_Threshold_Error"
"SPEED_MOTOR_Negative_Speed_Threshold_Warning".



12

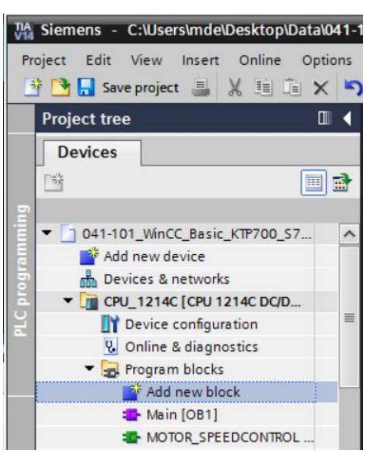
7.15.7 Discrete alarms

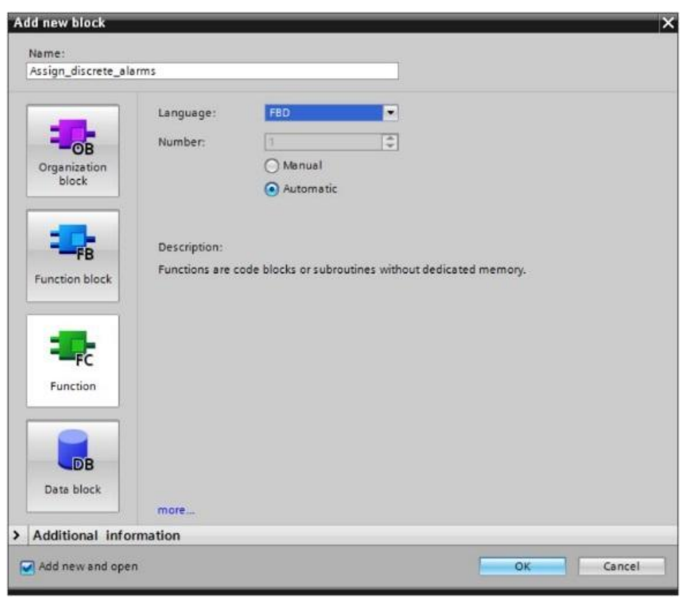
→ Before you can create discrete alarms in the panel, you need a global tag with at least 16 bits in the CPU 1214C, which you will use to trigger the discrete alarms from the PLC. Here, you open the → "OPERATING_HMI[DB4]" data block in the → "Program blocks" folder of "CPU 1214C" and create a global tag → "Group signals01" of data type → "Word".



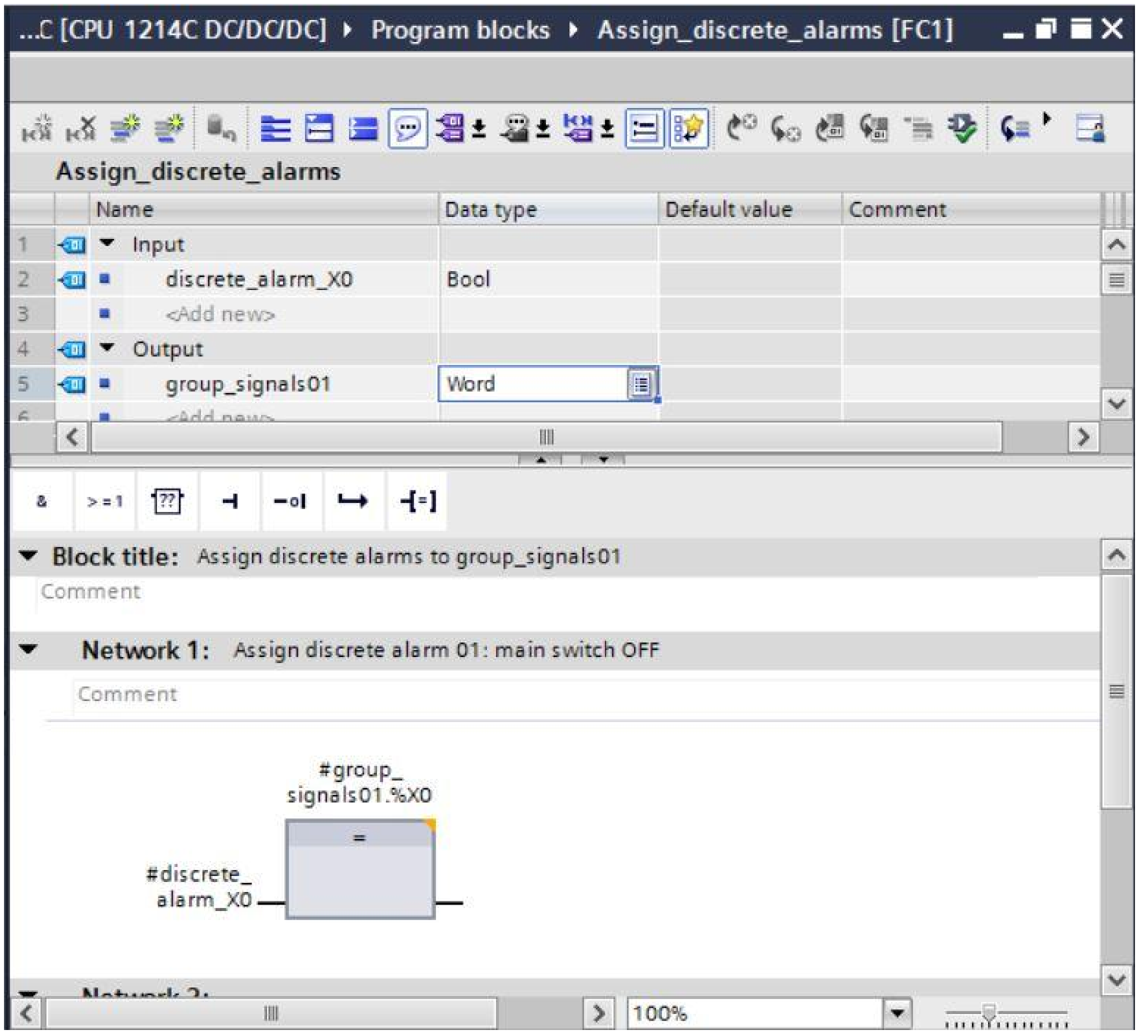
→ In the → "Program blocks" folder, click → "Add new block" to create

→ "Assign_discrete_alarms".





→ In the "Assign_discrete_alarms" function, create a local input tag → "discrete_alarm_X0" of data type → "Bool" and a local output tag → "group_signals01" of data type → "Word". In the first network, program a simple -[=] assignment of the → "discrete_alarm_X0" tag to the → "group_signals01" tag.

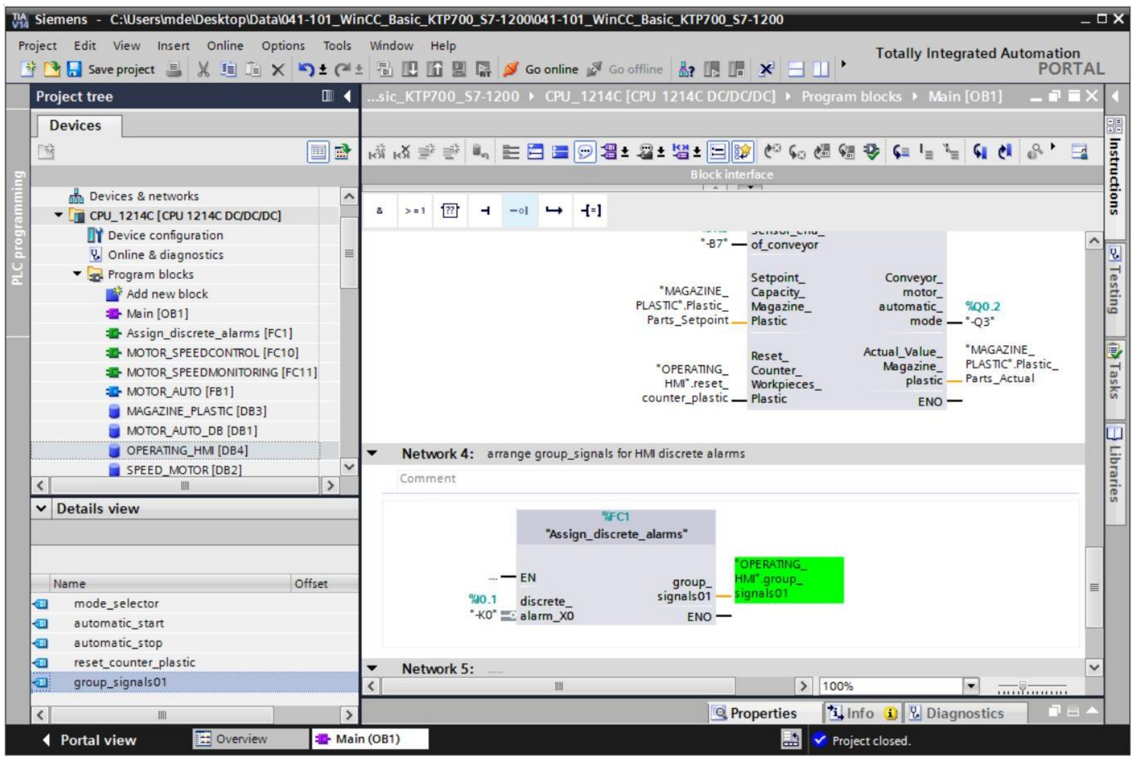


Note: The "Tag1.%X0" syntax is referred to as slice access in the TIA Portal. This enables, for example, bit-by-bit access to a tag of data type Byte, Word or DWord. If you need additional information on this, you can search the term "slice" in the STEP 7 online help.

12

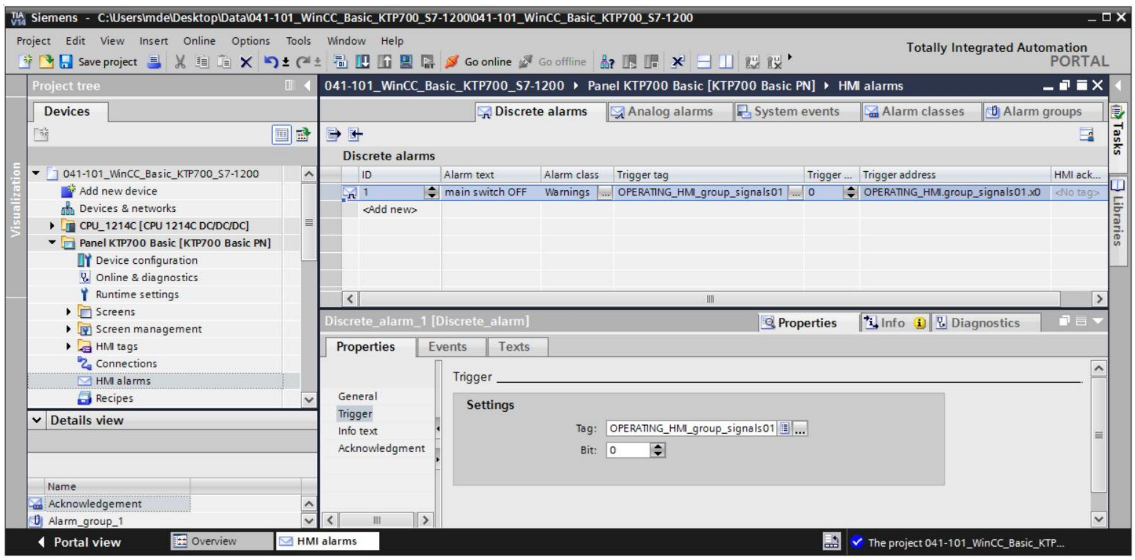
→ Next, open the → "Main[OB1]" block in the "Program blocks" folder and call the → "Assign_discrete_alarms[FC1]" function → in "Network 4". Connect the input of the "Assign_discrete_alarms[FC1]" function with the **negated** global tag → "-K0" / %I0.1 / System "ON" (no) from the "Tag table_Sorting station".

Connect the output of the "Assign_discrete_alarms[FC1]" function to the global tag → "group_signals01" from the "OPERATING_HMI[DB4]" data block.



→ Return to → "HMI alarms" → "Discrete alarms" in "Panel KTP700 Basic". Click → "Add" to create a new alarm. Select the → "group_signals01" tag you just created from the "OPERATING_HMI[DB4]" data block.

Enter the text → "Main switch OFF" in the "Alarm text" column, select the "Alarm class" → "Warnings" and → "0" for "Trigger bit". The "Trigger address" now displays "OPERATING_HMI_group_signals01.x0".



→ Before the visualization is tested, the "Acquisition cycle" of all newly created tags is to be accelerated again from 1 second to 100 milliseconds in the default tag table.

→ Before the visualization is downloaded to the panel, compile the CPU and panel again and save the project. (→ CPU_1214C → → Panel KTP700 Basic → → Save project)

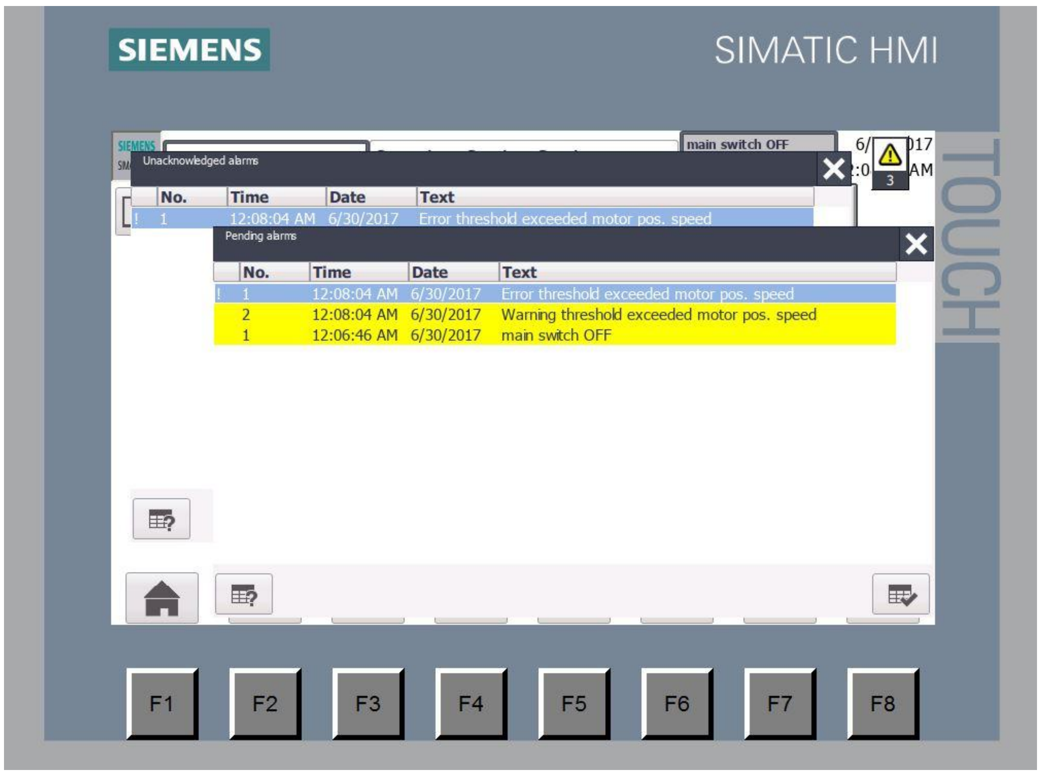
→ After successful compilation, the entire controller with the created program including the hardware configuration, as previously described in earlier modules, can be downloaded.

(→ CPU_1214C →)

→ To download the visualization to the panel, follow the same procedure. Select the → "Panel KTP700 Basic [KTP700 Basic]" folder and click the → "Download to device" button.

12

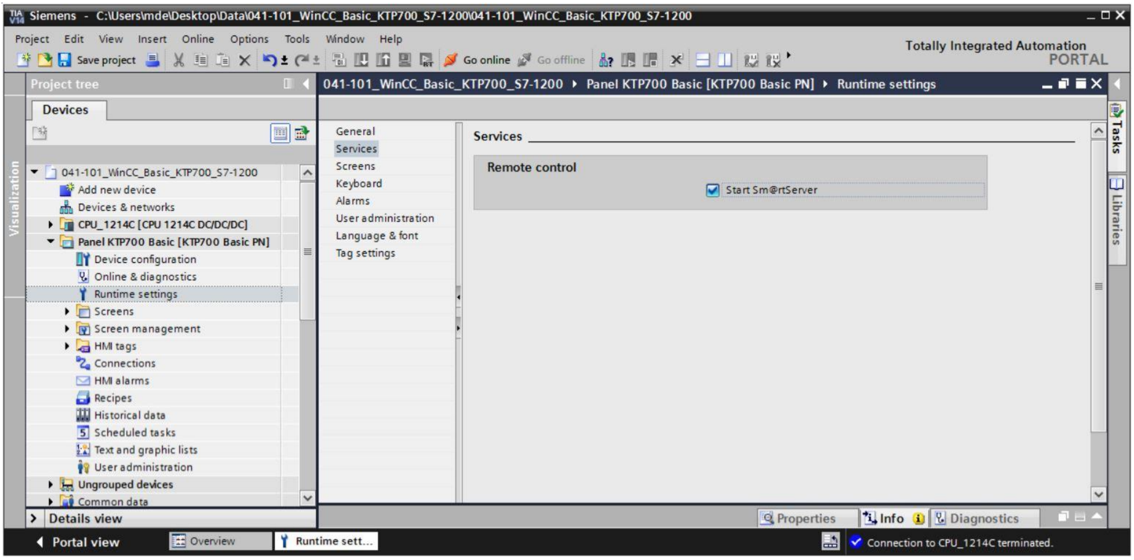
→ Analog alarms and discrete alarms will now be automatically displayed in the "Pending/Unacknowledged alarms" alarm window and in the "Alarm line" in Runtime. Details and help texts can be displayed in the alarm window and alarms can be acknowledged if necessary. If the alarm window has been closed, it can be displayed again by clicking the displayed alarm indicator. Various alarm classes appear in different colors.



7.16 Remote operation of Panel KTP700 Basic

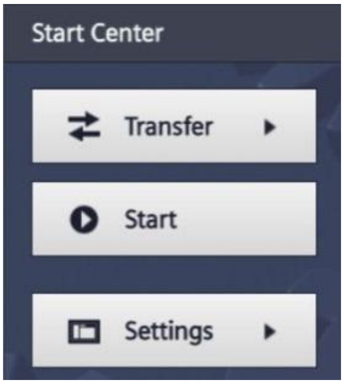
7.16.1 Activating web services for Runtime

→ To enable remote control, the → "Runtime settings" must be opened with a double-click in the configuration for → Panel KTP700 Basic. Under → "Remote control" in "Services", select the → ☑ "Start Sm@rtServer" option.



7.16.2 WinCC Internet settings on Panel KTP700 Basic

→ Settings must also be made directly on the panel. Select → the "Settings" button in the "Start Center" immediately after switching on the power supply and starting the panel.

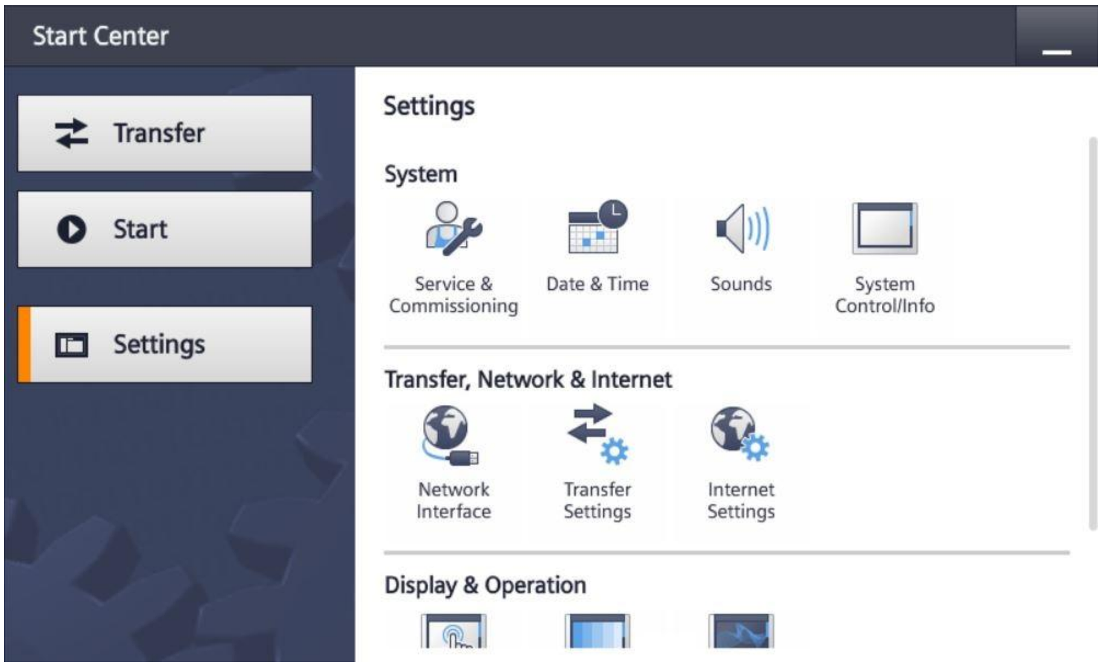


Note: You must select "Settings" in the "Start Center" quickly before Runtime automatically starts.

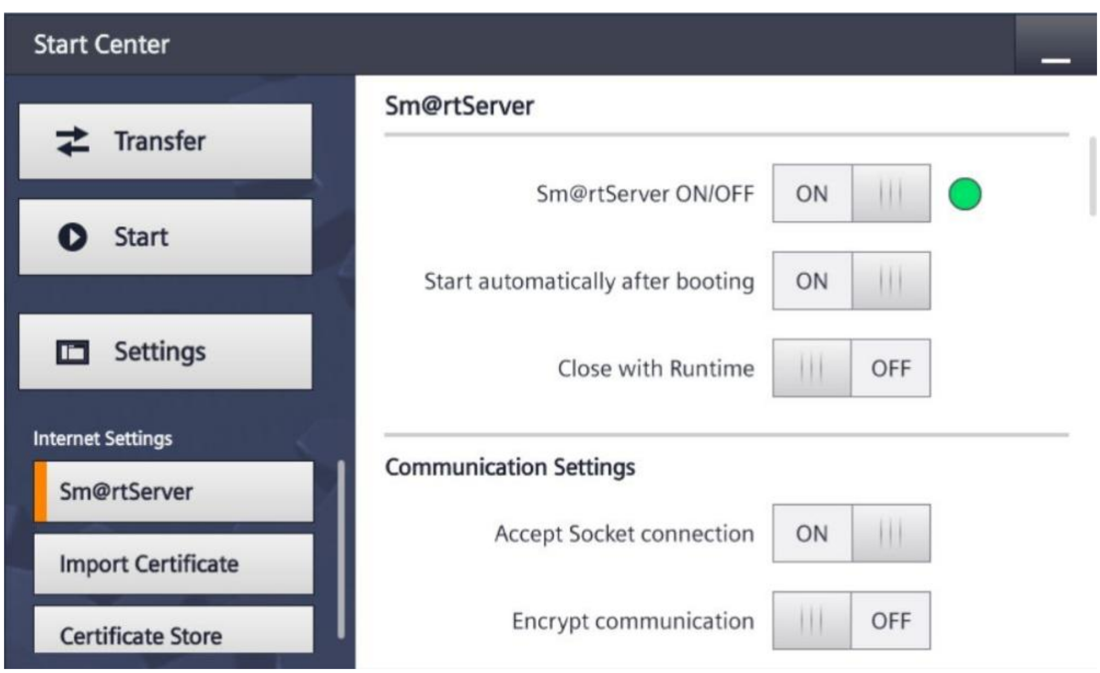
12

→ Under "Transfer, Network & Internet" click the 

icon to navigate to the web server settings.

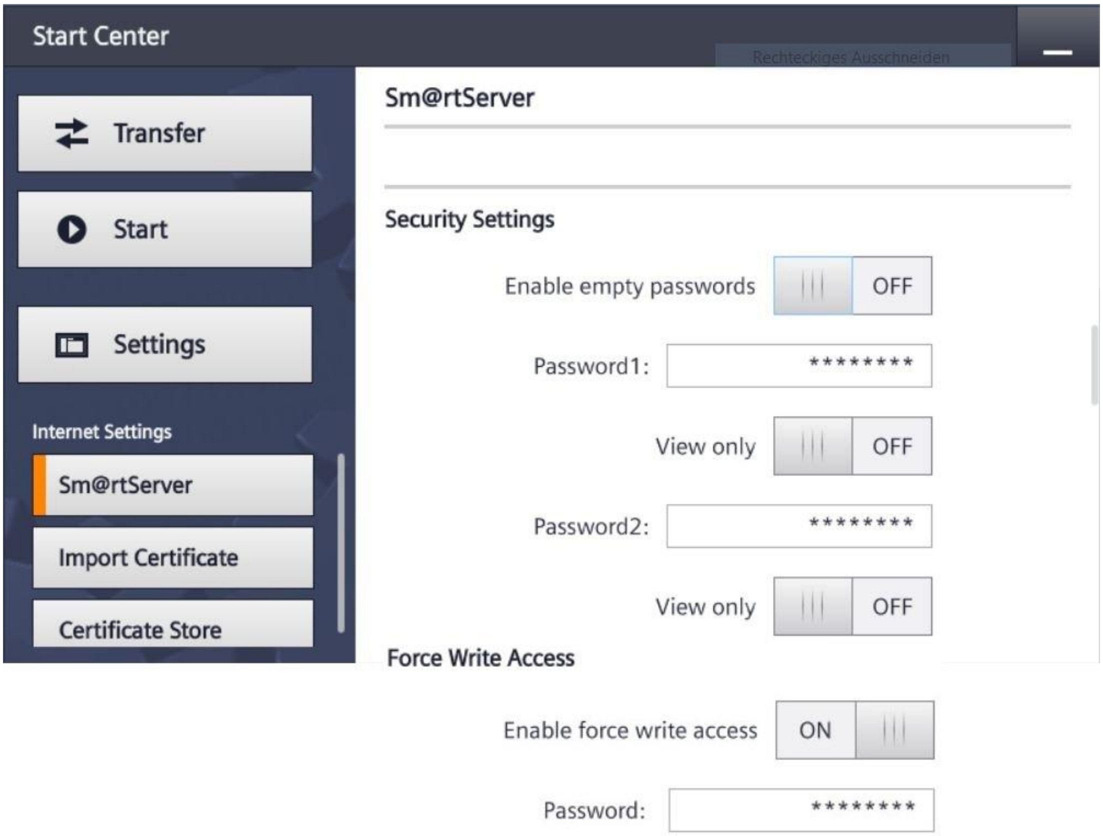


→ Select the following settings in the "Sm@rtServer" menu item.



12

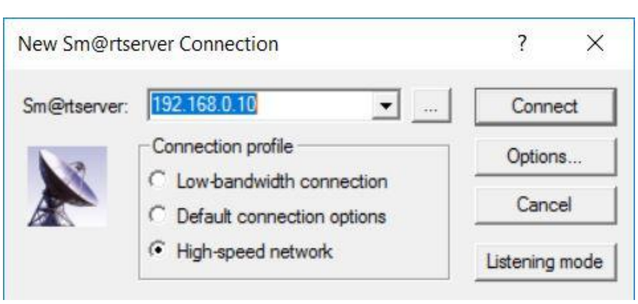
→ Under "Security Settings" and "Force Write Access", assign passwords (e.g. "sce") and select the settings shown here.



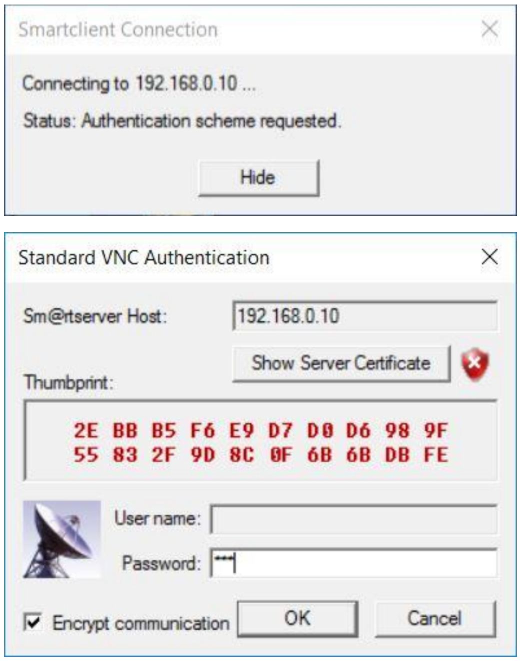
7.16.3 Starting remote access to Panel KTP700 Basic

→ To use remote access to your panel, you can start the → "Sm@rtClient" tool installed with the TIA Portal.

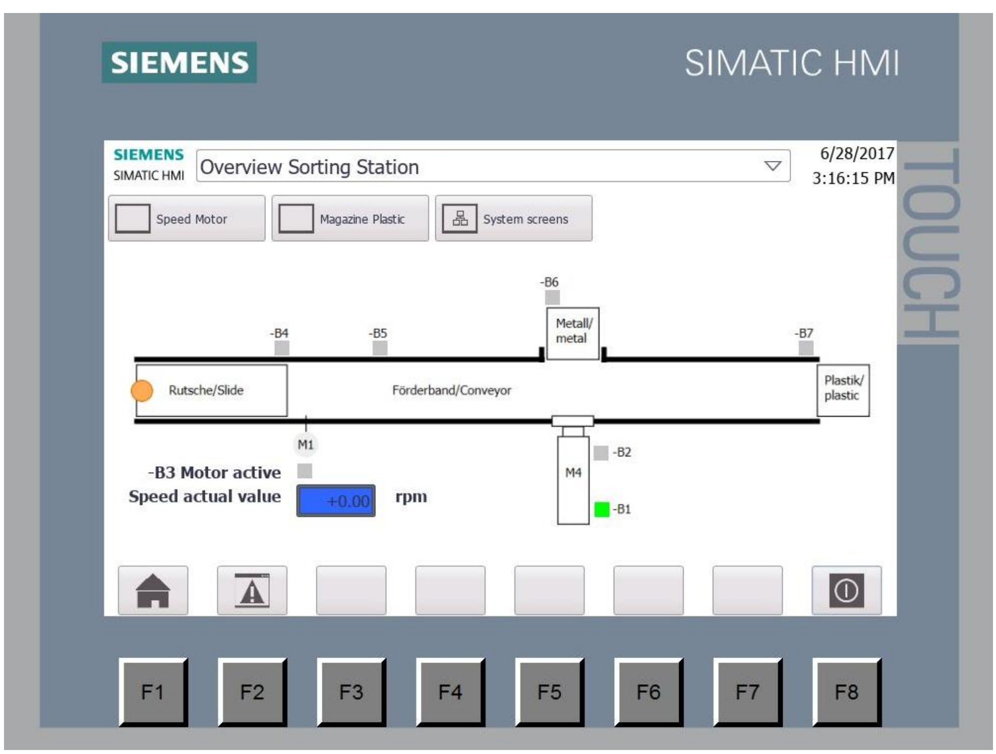
→ Enter the IP address of the device → "192.168.0.10" and click on → "Connect".



→ A window indicating the status of the connection is displayed as well as another window where you must enter the password you set previously in the panel → "sce" → "OK".



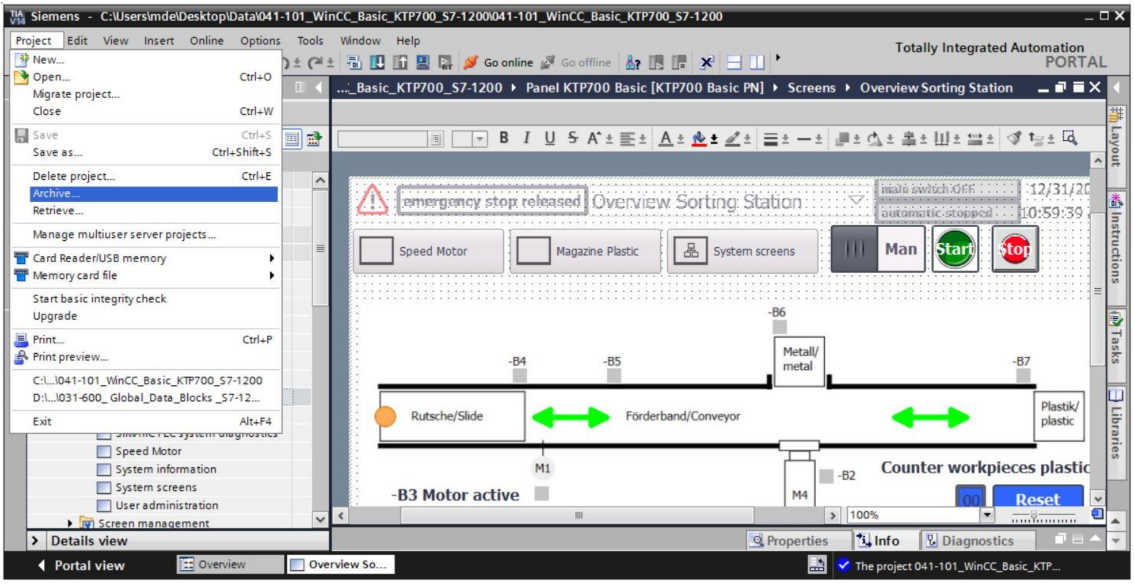
→ You then have the option of monitoring and operating the panel remotely and even changing the settings in Windows CE of the device.



12

7.17 Archiving the project

Finally, you should archive the complete project. Select → "Project" → "Archive …" in the menu. Create a folder in which you want to archive your project and save your project as file type 'TIA Portal project archive'. (→ Project → Archive … → SCE_EN_041-101 WinCC Basic with KTP700 and S7-1200…. → Save)



12

8 Checklist

No.	Description	Checked
1	Program successfully changed in the CPU 1214C	
2	Successful compilation of the CPU 1214C without error message	
3	Successful download of the CPU 1214C without error message	
4	Process visualization successfully created for Touch Panel KTP700 Basic	
5	Successful compilation of the Touch Panel KTP700 Basic without error message	
6	Successful download of the Touch Panel KTP700 Basic without error message	
7	Switch on system (-K0 = 1) Cylinder retracted / feedback activated (-B1 = 1) EMERGENCY STOP (-A1 = 1) not activated AUTOMATIC mode (in panel) Automatic stop pushbutton not actuated (-S2 = 1) Briefly press the automatic start pushbutton (in panel) Sensor part at slide activated (-B4 = 1) Conveyor motor -M1 variable speed (-Q3 = 1) turns on and remains active The speed corresponds to the speed setpoint in the range of +/- 50 rpm	
8	Sensor at conveyor end activated (-B7 = 1) → -Q3 = 0 (after 2 seconds)	
9	Briefly press the automatic stop pushbutton (-S2 = 0 or in panel) → -Q3 = 0	
10	Activate EMERGENCY STOP (-A1 = 0) → -Q3 = 0	
11	Manual mode (in panel)→ -Q3 = 0	
12	Switch off system (-K0 = 0) → -Q3 = 0	
13	Cylinder not retracted (-B1 = 0) → -Q3 = 0	
14	Speed > Speed limit fault max. → -Q3 = 0	
15	Speed < Speed limit fault min. → -Q3 = 0	
16	Values and alarms are displayed on the panel	
17	Project successfully archived	

9 Exercise

9.1 Task description – Exercise

The following functions are to be added to the process visualization in this exercise:

The **"Overview Sorting Station"** overview screen will display the "Setpoint" and "Actual" "Plastic" workpieces count.

The **"Speed Motor"** screen will display the current speed and speed setpoint of the motor graphically and in IO fields. The speed setpoint can also continue to be specified here.

The error and warning thresholds for positive and negative motor speed are to be displayed and set here in IO fields. A red box in front of the IO fields indicates when a limit has been exceeded.

In the **"Magazine Plastic"** screen, the "Setpoint" and "Actual" count will be displayed graphically and in IO fields. The setpoint for the plastic parts can be specified in the range 0 to 20 in the IO field. The counter can also be reset here.

The emergency stop and the status of automatic mode is also to be monitored in the **alarm system.** If the emergency stop is triggered or automatic mode is stopped, a warning is to be displayed.

9.2 Technology schematic diagram

Here you see the technology schematic diagram for the task.

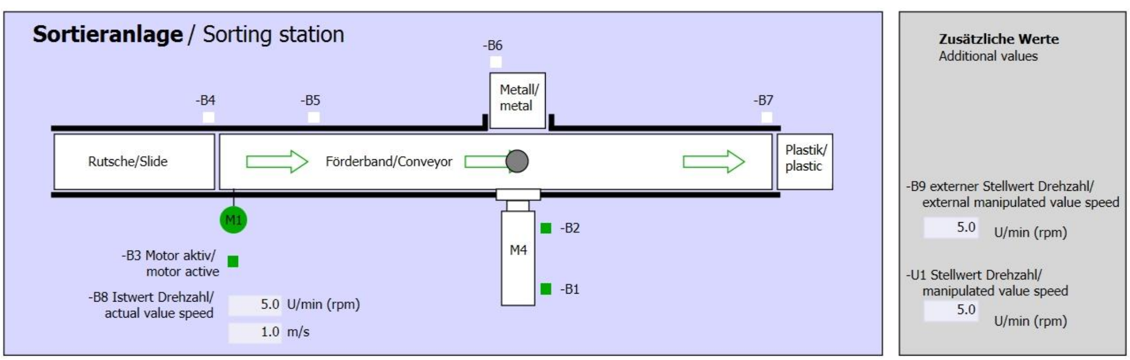


Figure 5: Technology schematic diagram

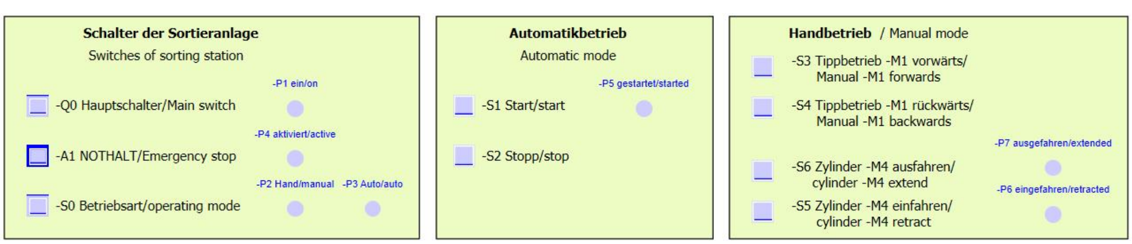


Figure 6: Operator panel

9.3 Reference table

The following signals are required as global operands for this task.

DI	Type	ID	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop ok	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0) / automatic (1)	Manual = 0 Auto=1
I 0.3	BOOL	-S1	"Automatic start" pushbutton	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder -M4 retracted	NO
I 1.0	BOOL	-B4	Sensor part at slide	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO
IW64	BOOL	-B8	Sensor actual motor speed +/-10V corresponds to +/- 50 rpm	

DO	Type	ID	Function	
Q 0.2	BOOL	-Q3	Conveyor motor -M1 variable speed	
QW 64	BOOL	-U1	Motor speed manipulated variable in both directions +/-10V corresponds to +/- 50 rpm	

Legend for reference list

DI	Digital input	DO	Digital output
AI	Analog input	AO	Analog output
I	Input	O	Output
NC	Normally Closed		
NO	Normally Open		

12

9.4 Planning

Plan the implementation of the task by yourself.

9.5 Checklist – Exercise

No.	Description	Checked
1	Program successfully changed in the CPU 1214C	
2	Successful compilation of the CPU 1214C without error message	
3	Successful download of the CPU 1214C without error message	
4	Process visualization successfully created for Touch Panel KTP700 Basic	
5	Successful compilation of the Touch Panel KTP700 Basic without error message	
6	Successful download of the Touch Panel KTP700 Basic without error message	
7	Switch on system (-K0 = 1) Cylinder retracted / return signal activated (-B1 = 1) EMERGENCY STOP (-A1 = 1) not activated AUTOMATIC mode (in panel) "Automatic stop" pushbutton not pressed (-S2 = 1) Briefly press "Automatic start" pushbutton (in panel) Sensor part at slide activated (-B4 = 1) Conveyor motor -M1 variable speed (-Q3 = 1) turns on and remains active The speed corresponds to the speed setpoint in the range of +/- 50 rpm	
8	Sensor at conveyor end activated (-B7 = 1) → -Q3 = 0 (after 2 seconds)	
9	Briefly press "Automatic stop" pushbutton (-S2 = 0 or in panel) → -Q3 = 0	
10	Activate EMERGENCY STOP (-A1 = 0) → -Q3 = 0	
11	Manual mode (in panel)→ -Q3 = 0	
12	Switch off system (-K0 = 0) → -Q3 = 0	
13	Cylinder not retracted (-B1 = 0) → -Q3 = 0	
14	Speed > Speed limit fault max. → -Q3 = 0	
15	Speed < Speed limit fault min. → -Q3 = 0	
16	Values and alarms are displayed on the panel	
17	Project successfully archived	

12

10 Additional information

More information for further practice and consolidation is available as orientation, for example, Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

siemens.com/sce/s7-1200

Preview "Additional information"

⊟ Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- ↗ TIA Portal Videos
- ↗ TIA Portal Tutorial Center
- › Getting Started
- ↗ Programming Guideline
- ↗ Easy Entry in SIMATIC S7-1200
- › Download Trial Software/Firmware
- ↗ Technical Documentation SIMATIC Controller
- ↗ Industry Online Support App
- ↗ TIA Portal, SIMATIC S7-1200/1500 Overview
- ↗ TIA Portal Website
- ↗ SIMATIC S7-1200 Website
- ↗ SIMATIC S7-1500 Website

12

Notes

Notes

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

**TIA Portal Modules
from Version V14 SP1**

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Note that these trainer packages are replaced with successor packages when necessary. An overview of the currently available SCE packages is available at: siemens.com/sce/tp

Continued training

For regional Siemens SCE Continued Training, get in touch with your regional SCE contact: siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

This SCE Learning/Training Document for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public educational facilities and R&D institutions. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems, which means it can be copied in whole or part and given to those being trained for use within the scope of their training. Circulation or copying this Learning/Training Document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written consent from the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

13

Table of contents

High-Level Language Programming with SCL and S7-1200

1 Objective

In this section, you will become familiar with the basic functions of the SCL high-level language. Test functions for eliminating logical programming errors will also be presented.

The SIMATIC S7 controllers listed in section 3 can be used.

2 Requirements

This section builds on the hardware configuration of a SIMATIC S7-1200. It can be implemented with any hardware configurations that have digital input and output cards. To implement this section, you can use the following project, for example:

"SCE_EN_011_101_Hardware_Configuration_CPU1214C…..zap14"

You should also be familiar with high-level language programming, such as Pascal.

13

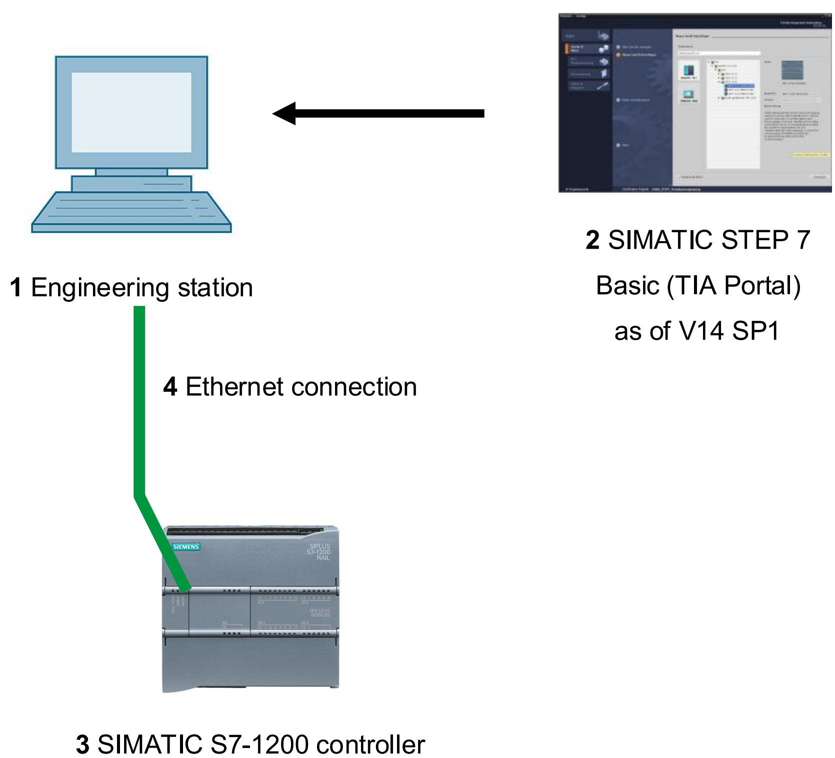
3 Hardware and software required

1 Engineering Station: The requirements are hardware and operating system (for additional information, see Readme on the TIA Portal Installation DVD)

2 SIMATIC STEP 7 Basic software in the TIA Portal - as of V14 SP1

3 SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC – Firmware V4.2.1 or higher

4 Ethernet connection between the engineering station and controller



4 Theory

4.1 SCL programming language

SCL (Structured Control Language) is a high-level, Pascal-based programming language that enables structured programming. The language corresponds to the "Structured Text" (ST) programming language specified in DIN EN-61131-3 (IEC 61131-3). In addition to high-level language elements, SCL contains typical elements of the PLC as language elements such as inputs, outputs, timers, block calls, etc. It supports the STEP 7 block concept and enables block programming in compliance with standards in addition to programming with Ladder Logic (LAD) and Function Block Diagram (FBD). This means SCL supplements and expands the STEP 7 programming software with its LAD and FBD programming languages.

You do not have to create every function yourself but can use pre-compiled blocks, such as system functions and system function blocks that are present in the CPU's operating system.

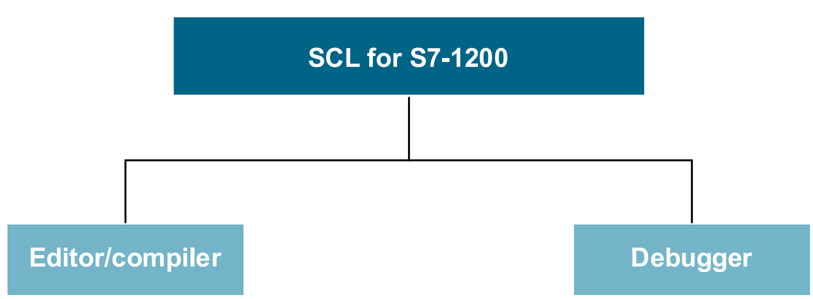
Blocks that are programmed with SCL can be mixed with LAD and FBD blocks. This means that a block programmed with SCL can call another block that is programmed in LAD or FBD. Accordingly, SCL blocks can also be called in LAD and FBD programs.

SCL networks can also be inserted in LAD and FBD blocks.

The SCL test functions can be used to find logical programming errors in an error-free compilation.

4.2 SCL development environment

There is a development environment that is tailored to the specific properties of both SCL and STEP 7 for use and application of SCL. This development environment consists of an editor/compiler and a debugger.



13

Editor/compiler

The SCL editor is a text editor that can be used to edit any text. The main task of the SCL editor is the creation and editing of blocks for STEP 7 programs. A basic syntax check is performed during the input which makes it easier to avoid errors during programming. Syntax errors are displayed in different colors.

The editor offers the following options:

– Programming of an S7 block in the SCL language

– Convenient insertion of language elements and block calls using drag & drop

– Direct syntax check during programming

– Customization of the editor to meet your needs, e.g. color-coding for the different language elements according to syntax

– Checking of the finished block through compiling

– Display of all errors and warnings that occur during compiling

– Localization of error locations in the block, optionally with error description and information on troubleshooting

Debugger

The SCL debugger enables you to check a program while it is running in the automation system (AS) and thus find potential logical errors.

SCL provides two different test modes:

– Continuous monitoring

– Step-by-step monitoring

With "Continuous monitoring" you can test a group of instructions within a block. During the test, the values of the tags and parameters are displayed in chronological order and – if possible – updated cyclically.

With "Step-by-step monitoring" the logical program sequence is followed. You can run the program algorithm instruction-by-instruction and observe how the contents of the processed tags change in a result window.

The type of CPU you are using determines whether or not you can use "Step-by-step monitoring". The CPU must support the use of breakpoints. The CPU used in this document does not support breakpoints.

13

5 Task

5.1 Example task – Tank volume

In the first part, you are to program the calculation of the tank volume.

5.2 Expansion of the sample task

In the second part, the task is expanded and you are to program an error evaluation.

6 Planning

The tank is in the shape of a vertical cylinder. The filling level is measured with an analog sensor. For the first test, the filling level value should be available as a scaled value (in meters).

Global parameters, such as the diameter and height of the tank, are to be stored in a structured manner in a global data block "Data_Tank".

The program for calculation of the volume should be written in a "Calculate_Volume" function and the parameters are to use the unit 'meter' or 'liter'.

6.1 Global data block "Data_Tank"

The global parameters are stored in multiple structures in a global data block.

Name	Data type	Start value	Comment
Dimensions	STRUCT		
Height	REAL	12.0	in meter
Diameter	REAL	3.5	in meter
measured_data	STRUCT		
filling_level_per	INT	0	value between 0...27648
filling_level_scal	REAL	0.0	range 0…12.0.
Volume	REAL	0.0	Volume of tank in liter
fault_flags	STRUCT		
calculate_volume	BOOL		fault == true
Scaling	BOOL		fault == true

Table 1: Parameters in the "Data_Tank" data block

13

6.2 "Calculate_Volume" function

This block calculates the volume of the tank in liters.

In the first step, there is to be no check of the transferred parameters for reasonableness.

The following parameters are required for this step:

Input	**Data type**	**Comment**
Diameter	REAL	Diameter of cylindric tank in meter
Filling_level	REAL	Filling level of liquid in meter
Output		
Volume	REAL	Volume of liquid in the tank in liter

Table 2: Parameters for "Calculate_Volume" function in the first step

The formula for calculating the volume of a vertical cylinder is used to solve the task. The conversion factor 1000 is used to calculate the result in liters.

$$V = \frac{d^2}{4} \bullet \pi \bullet h \qquad \Rightarrow \qquad \#Volume = \frac{\#Diameter^2}{4} \bullet 3.14159 \bullet \#Filling\_level \bullet 1000$$

6.3 Expansion of the "Calculate_Volume" function

The second step checks whether the diameter is greater than zero. In addition, a test is to be performed to determine whether the filling level is greater than or equal to zero and less than or equal to the height of the tank.

In case of an error, the new parameter "er" is set to TRUE, and the "Volume" parameter is set to the value -1.

For this purpose, add the "er" and "Height" parameters to the interface.

Input	**Data type**	**Comment**
Height	REAL	Height of cylindric tank in meter
Diameter	REAL	Diameter of cylindric tank in meter
Filling_level	REAL	Filling level of liquid in meter
Output		
er	BOOL	fault flag; fault == true
Volume	REAL	Volume of liquid in the tank in liter

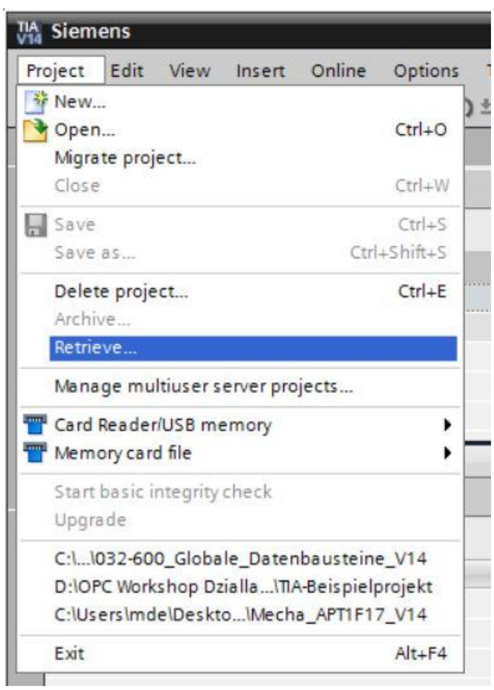
Table 3: Parameters for "Calculate_Volume" function in the second step

7 Structured step-by-step instructions

You can find instructions on how to implement the planning below. If you already have a good understanding of everything, it is sufficient to focus on the numbered steps. Otherwise, simply follow the steps of the instructions explained below.

7.1 Retrieving an existing project

→ Before you can start programming, you need a project with a hardware configuration.
(e.g. SCE_EN_011-101_Hardware_Configuration_CPU1214C_....zap14).
To retrieve an existing project, you must select the respective archive from the Project view under → Project → Retrieve. Confirm your selection with "Open".
(→ Project → Retrieve → Selection of a .zap archive → Open)

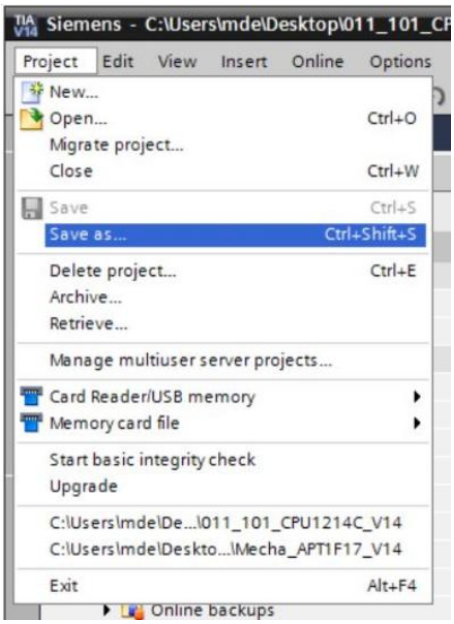


→ Next you can select the target directory to which you want to save the retrieved project. Confirm your selection with "OK". (→ Project → Save as... → OK)

13

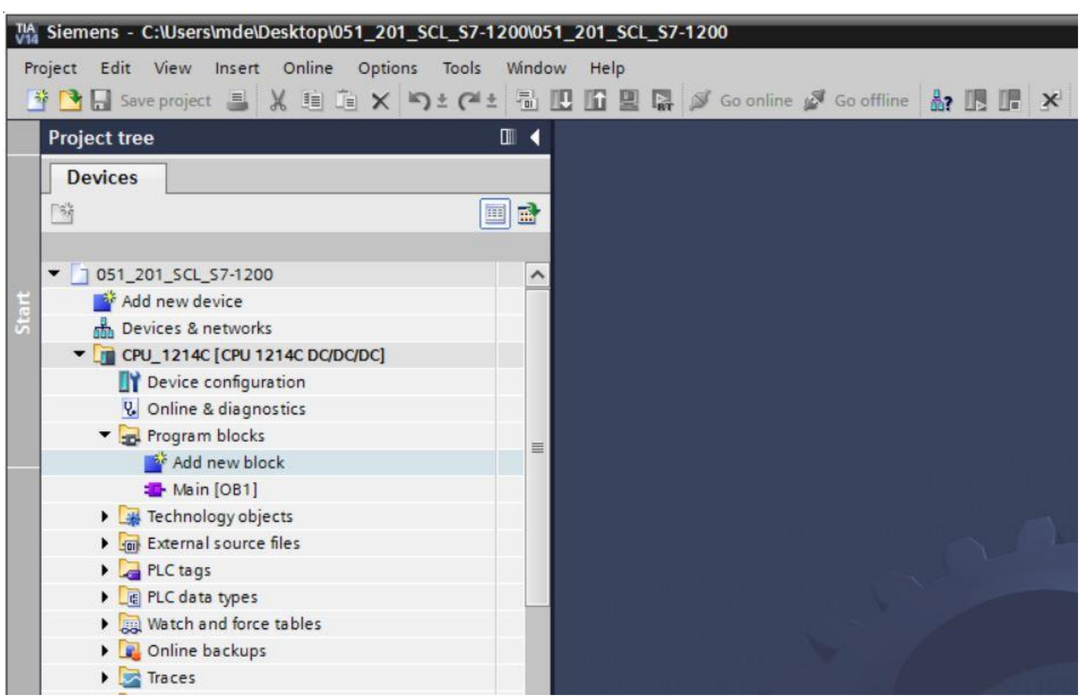
7.2 Saving the project under a new name

→ You save the opened project under the name 051-201_SCL_S7-1200. (→ Project → Save as … → 051-201_SCL_S7-1200 → Save)



7.3 Creating the "Data_Tank" data block

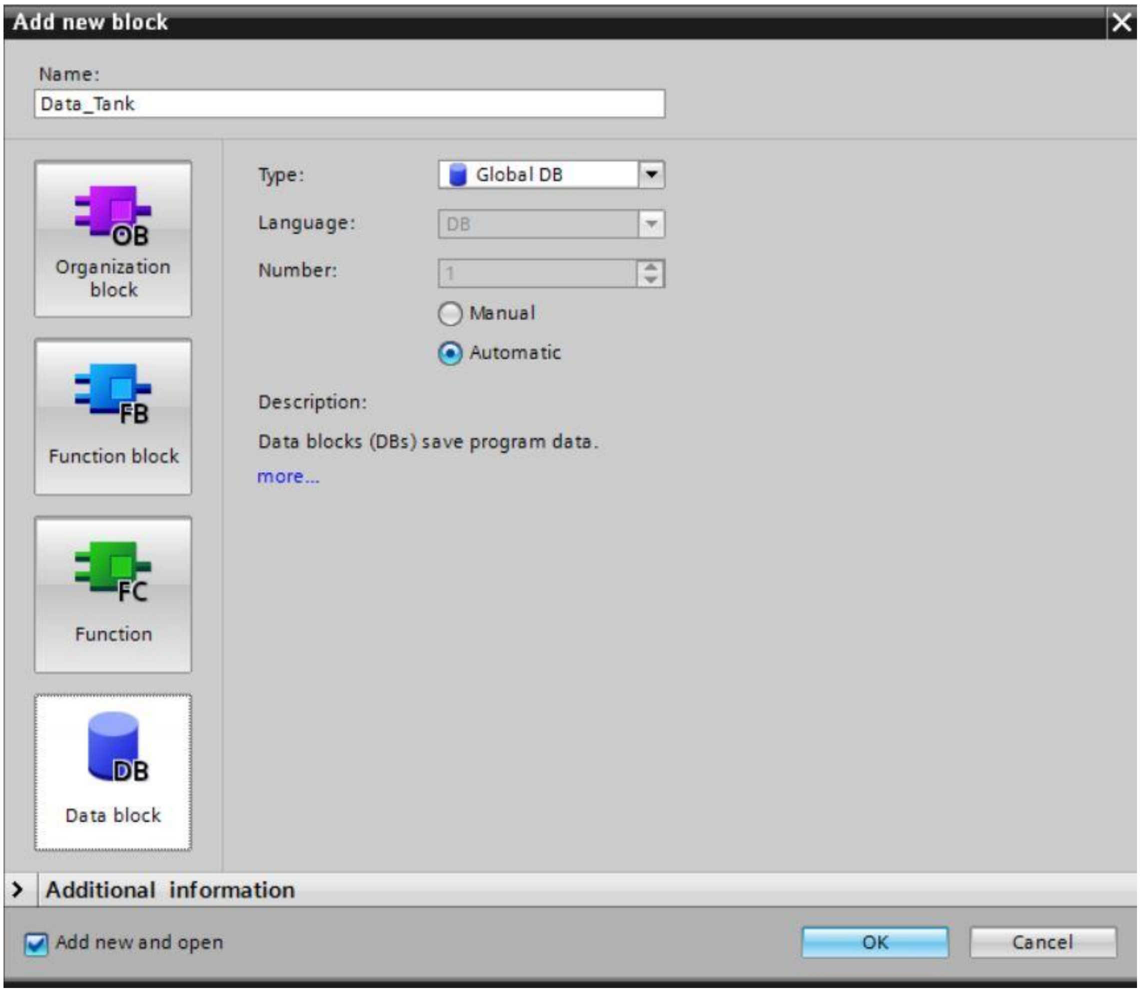
→ In the Project view, navigate to → the program blocks and create a new block by double-clicking → Add new block.



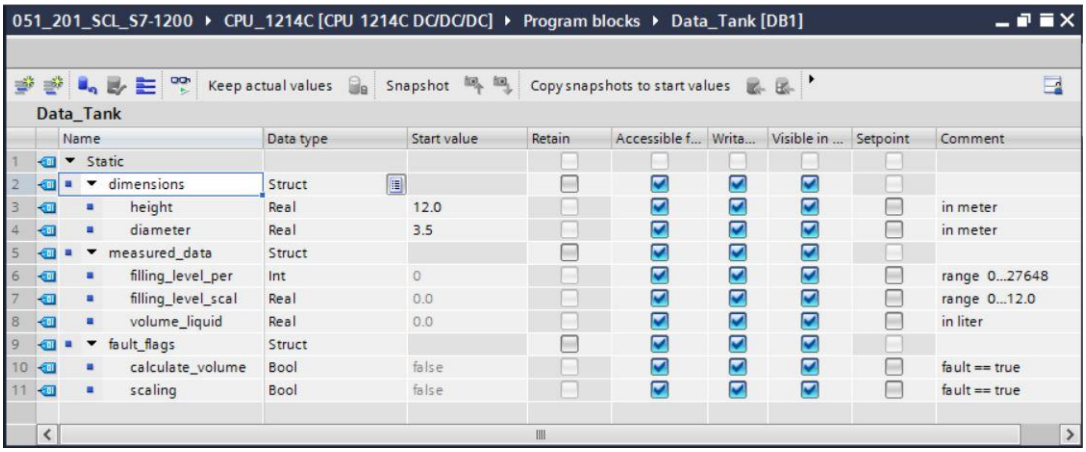
13

→ Now select a data block and enter the name.

(→  → "Data_Tank" → OK)



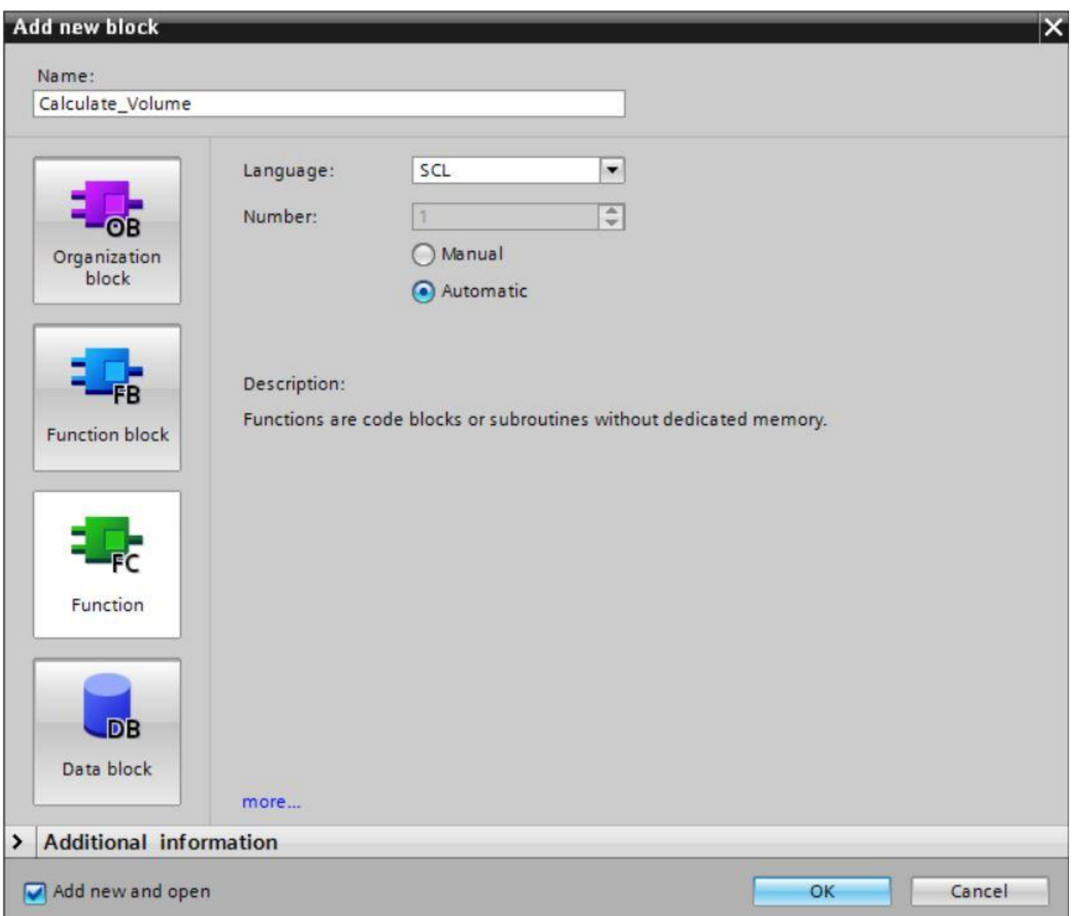
→ Next, enter the names of the tags listed below with data type, start value and comment.



7.4 Creating the "Calculate_Volume" function

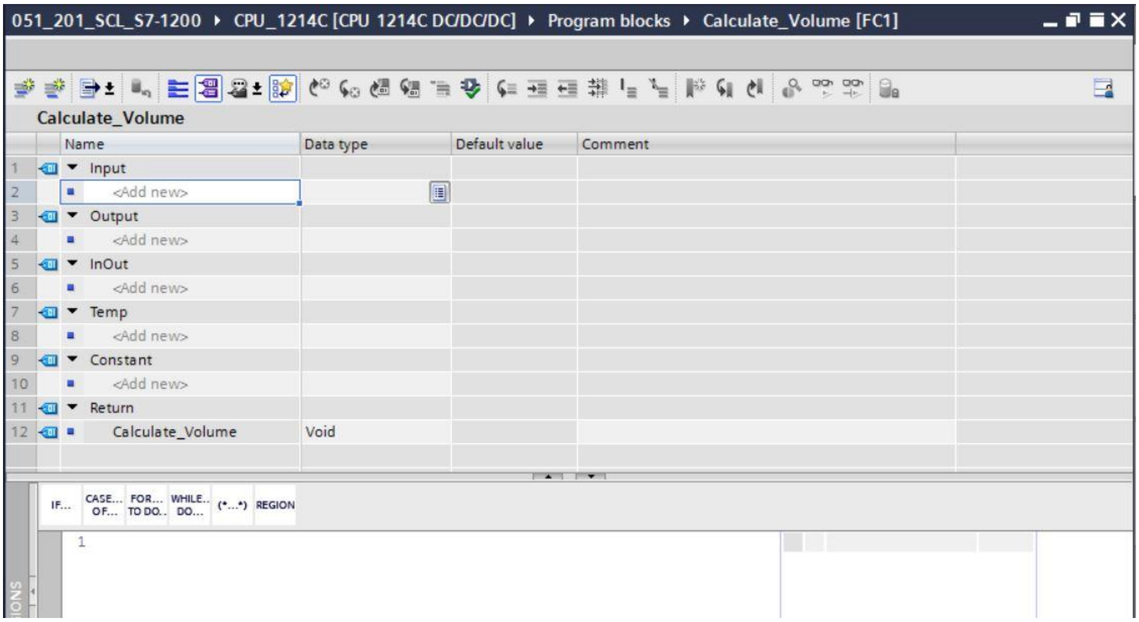
→ Next, add a function, enter the name and select the language.

(→ Add new block →  Function → "Calculate_Volume" → SCL → OK)

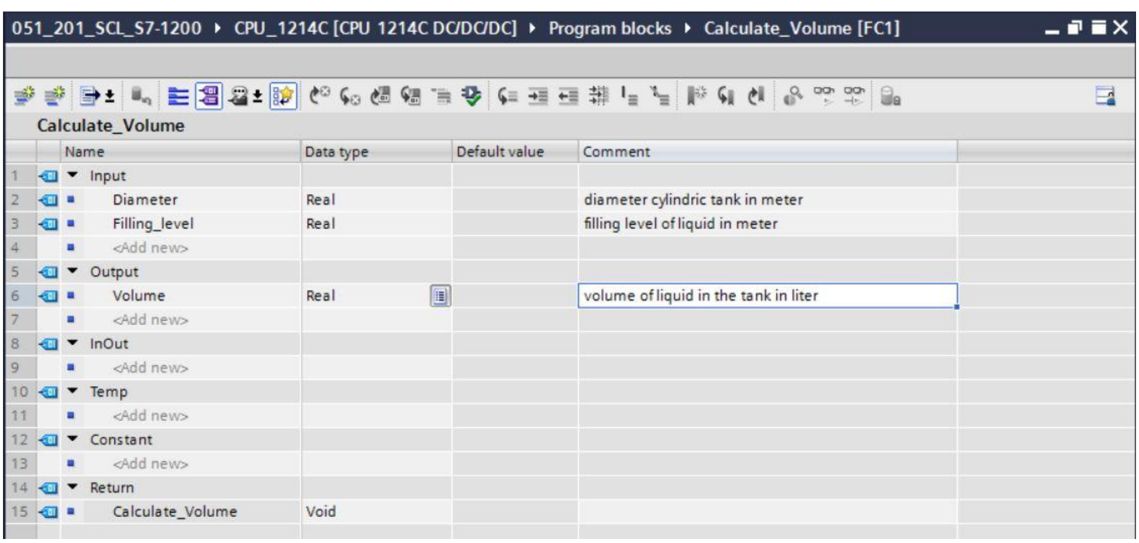


7.5 Specifying the interface of the "Calculate_Volume" function

→ The top section of your programming view shows the interface description of your function.



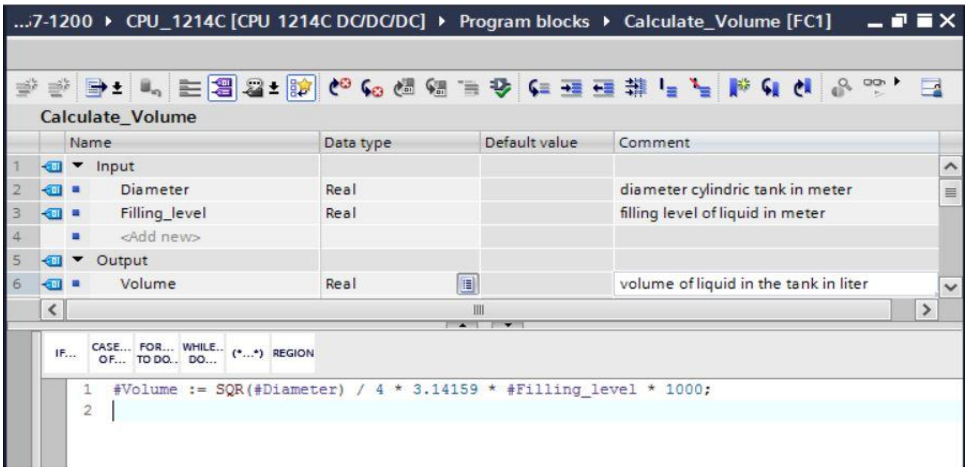
→ Create the following input and output parameters.
(→ Name → Data type → Comment)



13

7.6 Programming the "Calculate_Volume" function

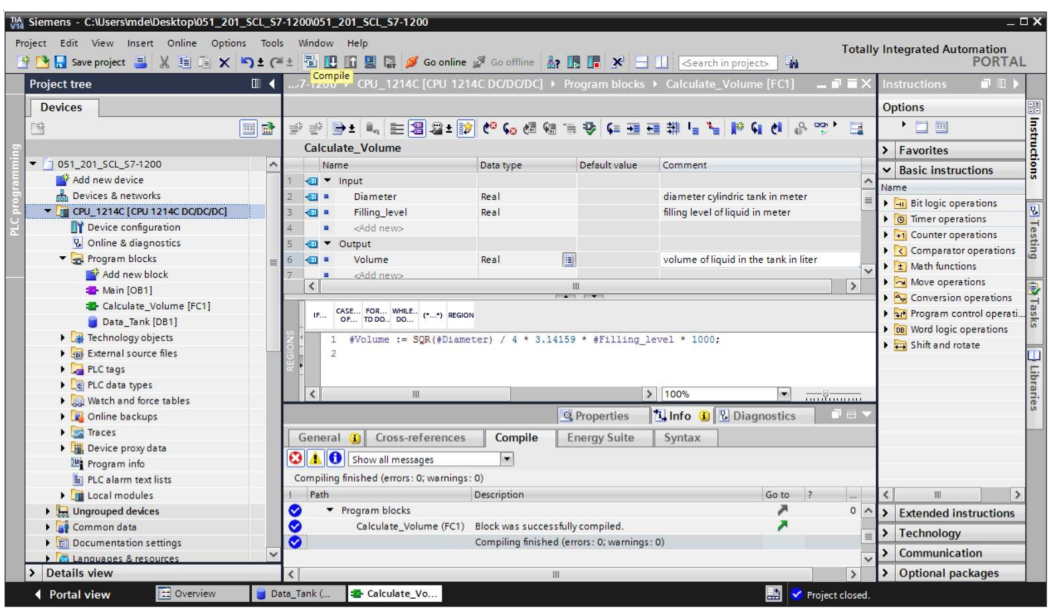
→ Enter the program shown below. (→ Enter program)

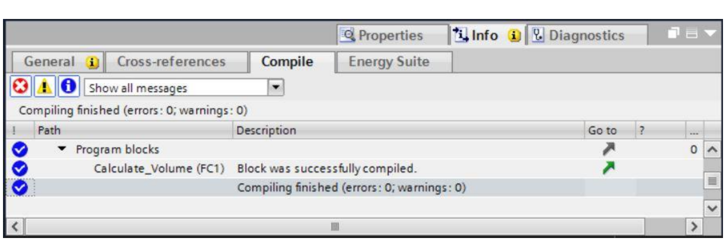


→ Now compile your program and check it for syntax errors. These are displayed in the Inspector window below the programming. Correct any errors and compile the program again.

Then save your program. (→  → Eliminate errors → 

)

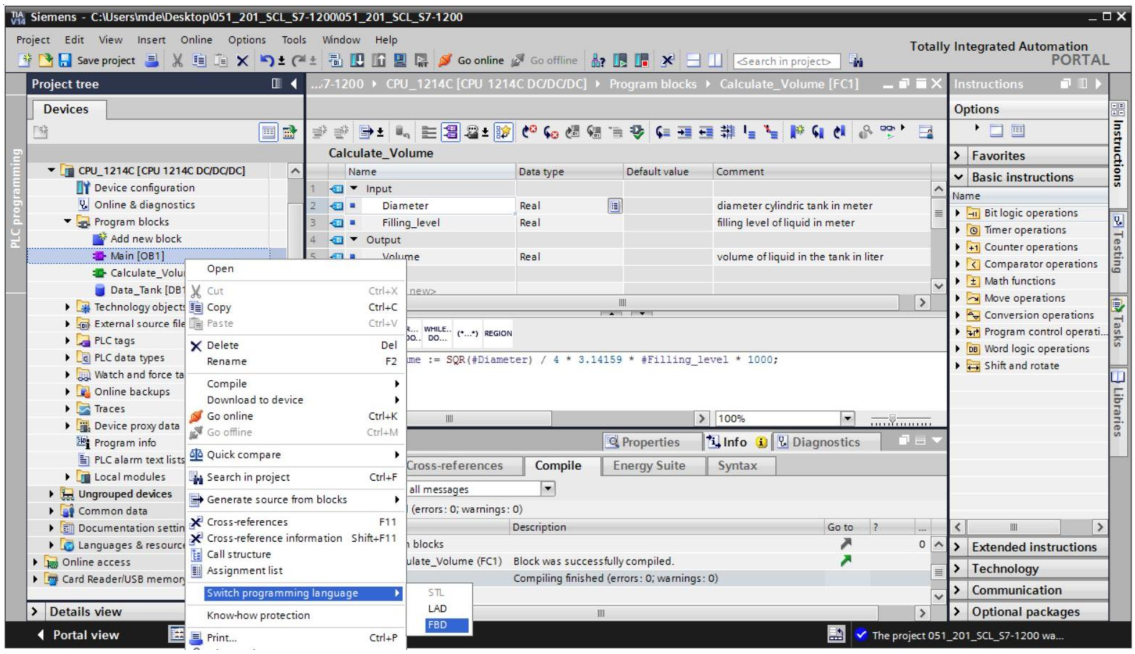




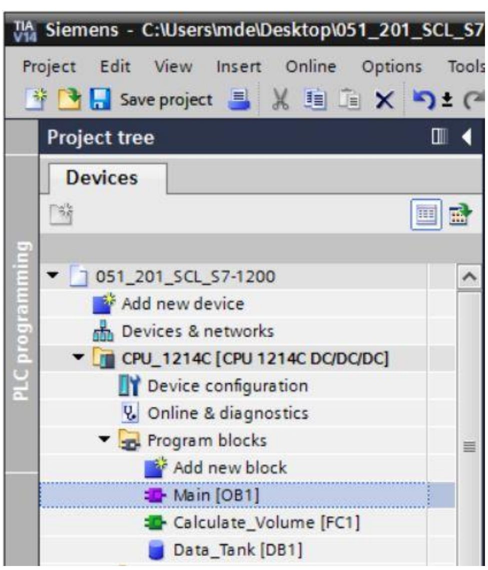
13

7.7 Programming the "Main [OB1]" organization block

→ Before programming the "Main [OB1]" organization block, switch the programming language to FBD. To do this, click on "Main [OB1]" in the "Program blocks" folder. (→ CPU_1214C[CPU 1214C DC/DC/DC] → Program blocks → Main [OB1] → Switch programming language → FBD)

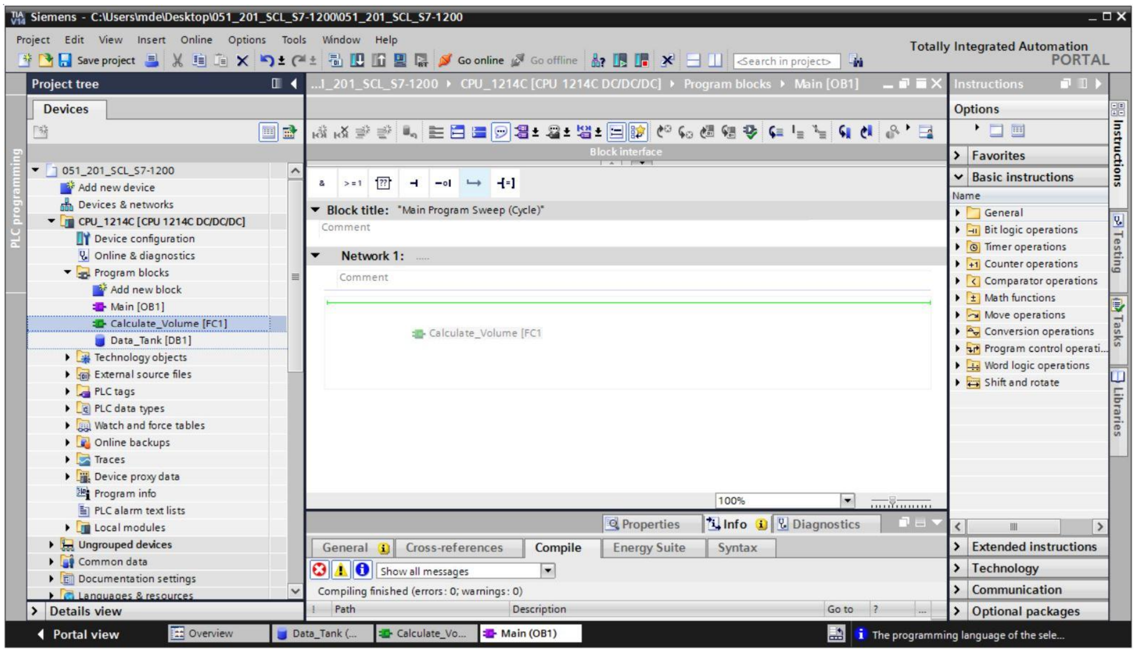


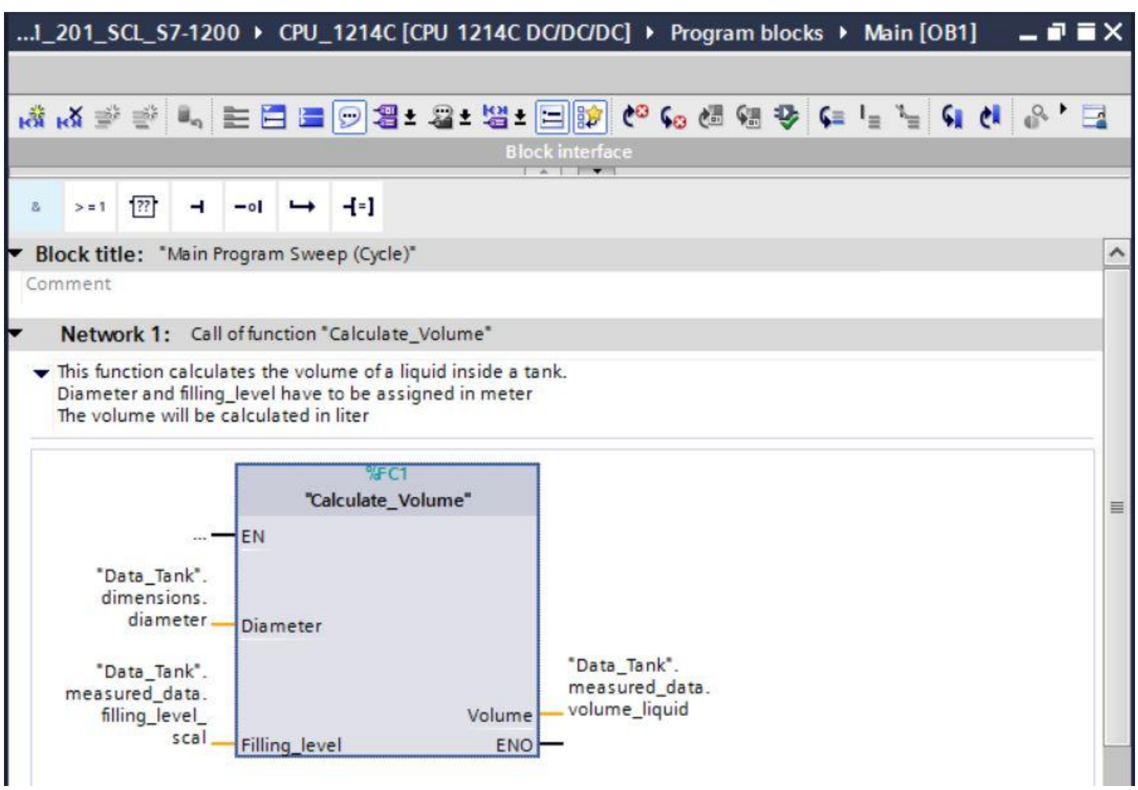
→ Now double-click the "Main [OB1]" organization block to open it.



13

→ Call the "Calculate_Volume" function in the first network. Assign network title, comment and connect the parameters. (→ Call "Calculate_Volume" → Assign network title → Write network comment → Connect parameters)





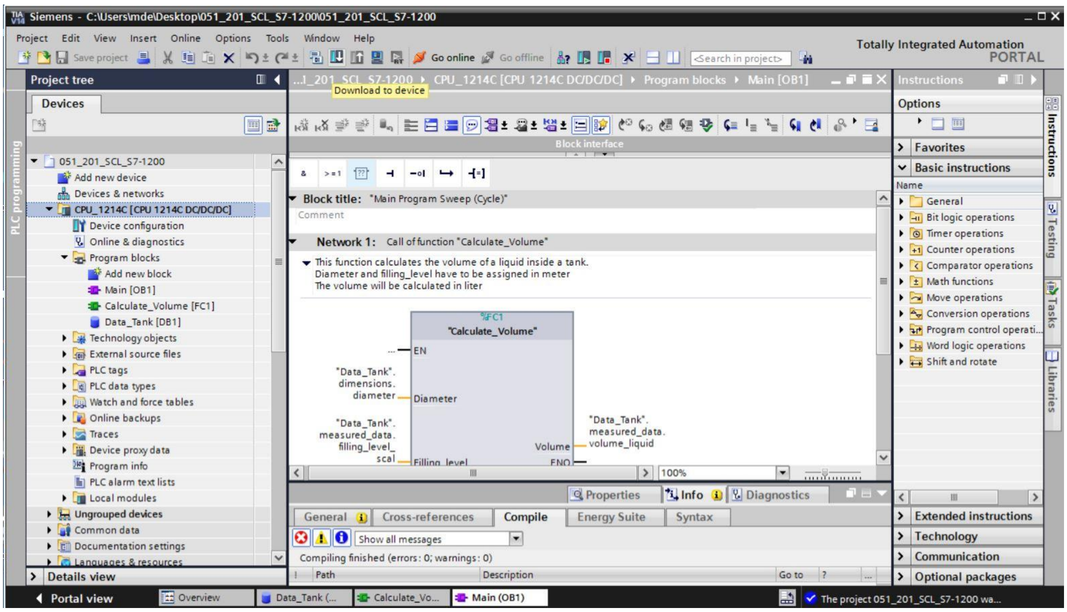
13

7.8 Compiling and downloading the program

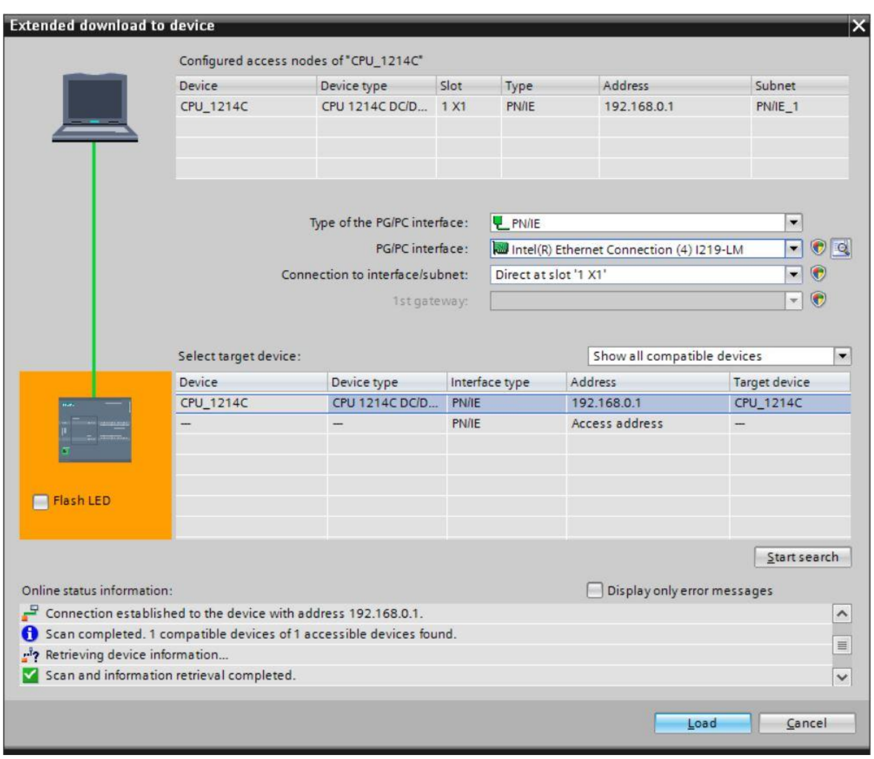
→ Click the "Program blocks" folder and compile the entire program. After successful compilation, save your project and download it to the controller.

(→  → 

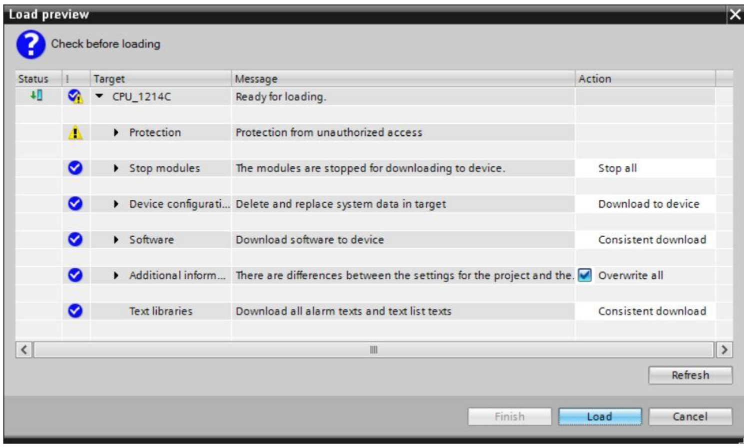
→ )



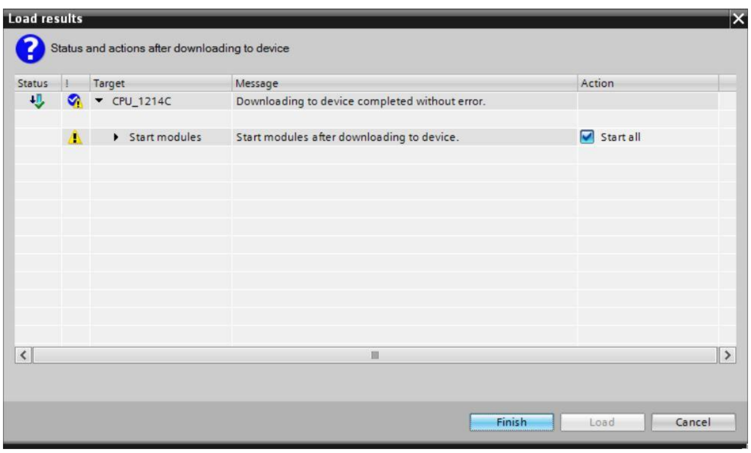
→ Select PG/PC interface → Select subnet → Start search → Load



→ Make a selection, if necessary → Load

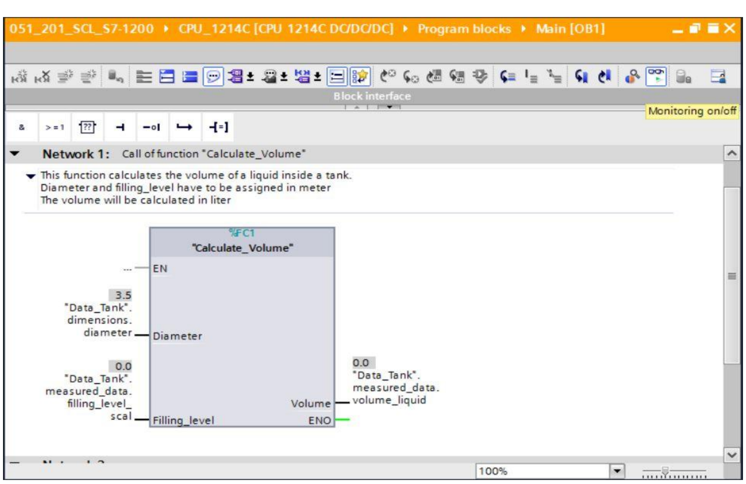


→ Finish

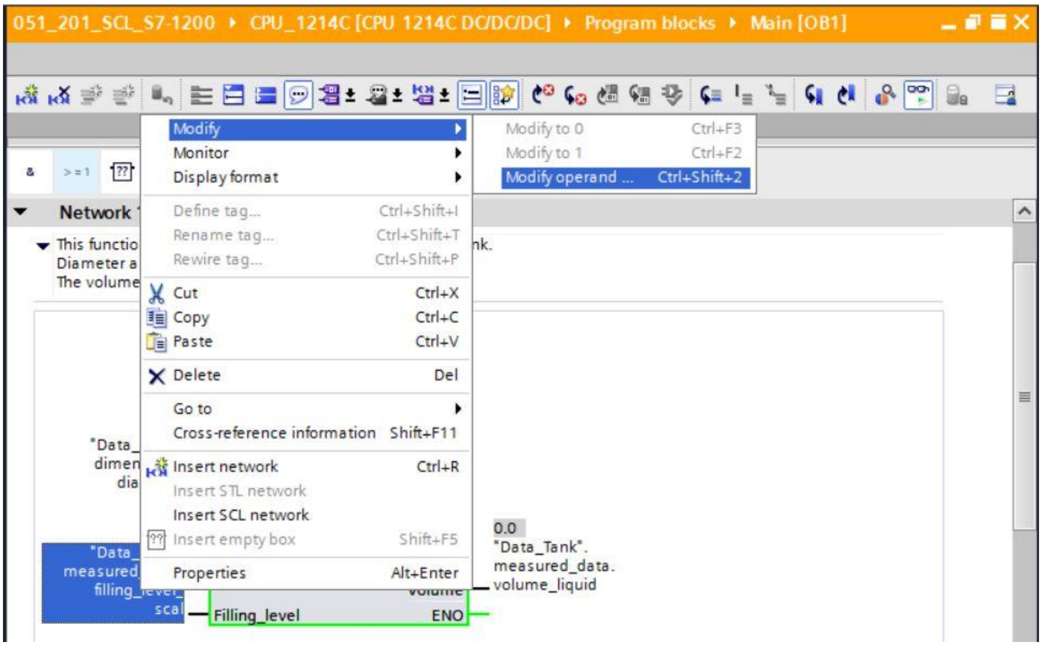


7.9 Monitoring and testing the organization block

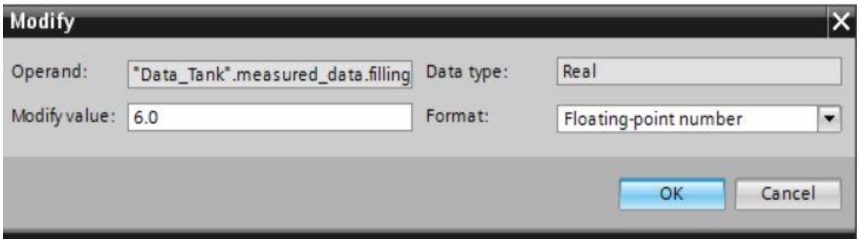
→ In the open OB1, click the  icon to monitor the block.



→ Test your program by writing a value to the "Filling_level_scal" tag in the data block. (→ Right-click on "Filling_level_scal" → "Modify" menu → Modify operand)

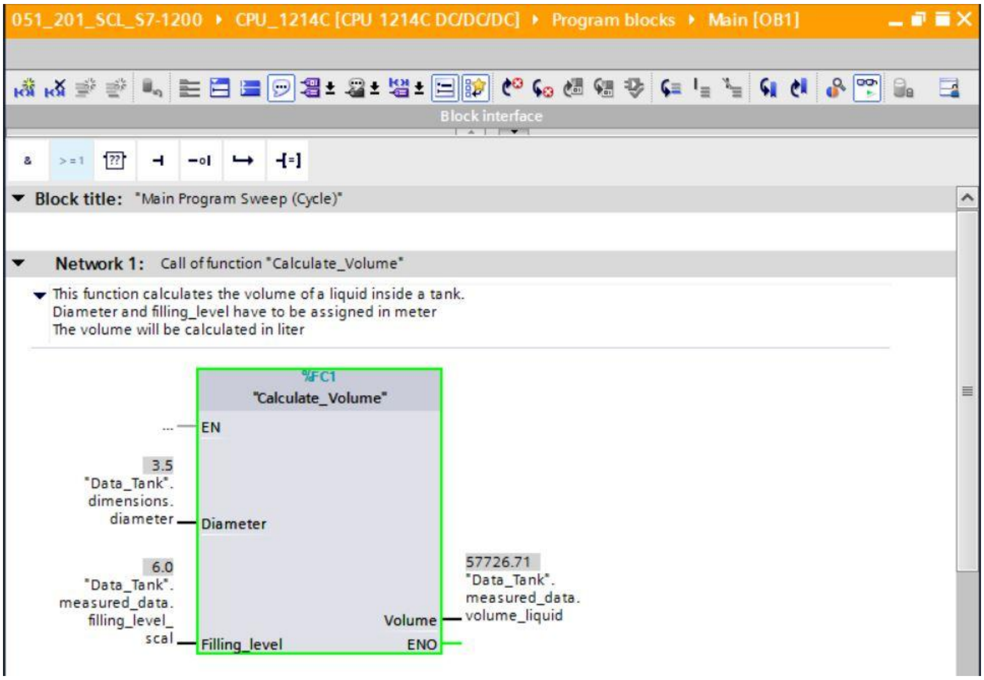


→ Enter value 6.0 → OK



13

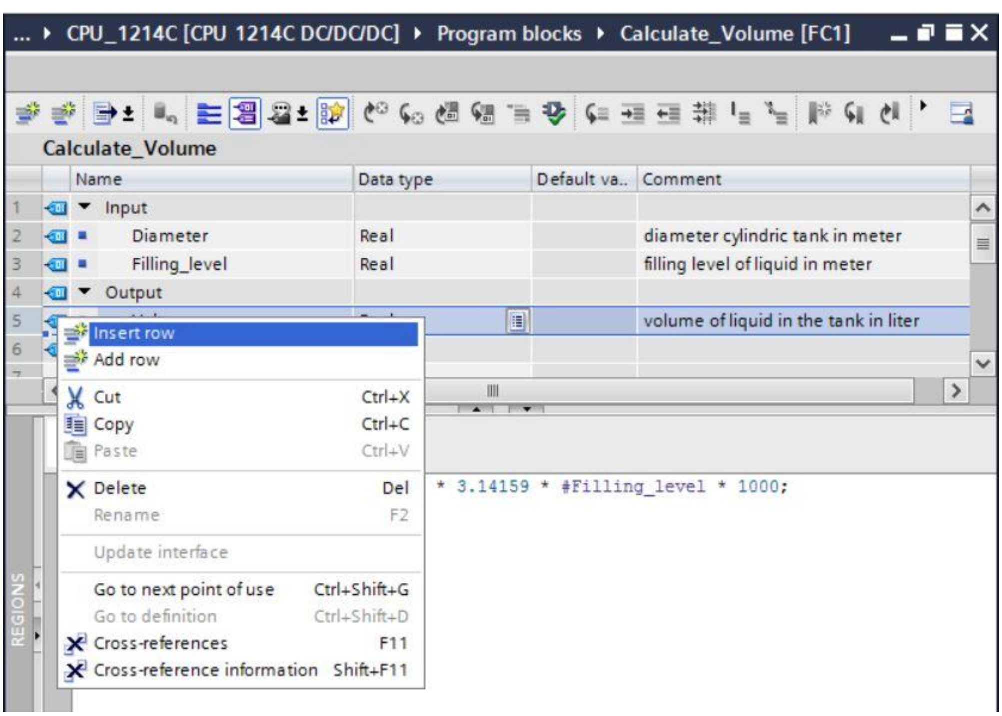
→ Check the result for correctness.



7.10 Expansion of the "Calculate_Volume" function

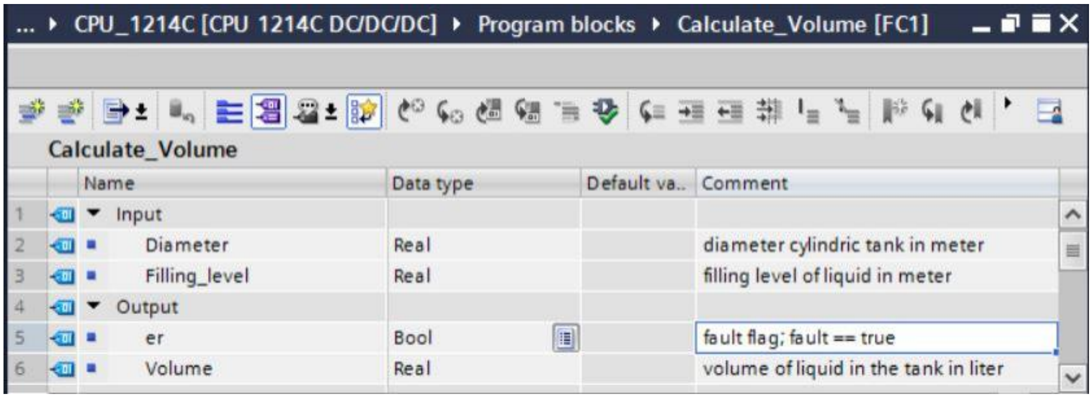
→ Open the "Calculate_Volume" function, and insert a row in the output parameters by right-clicking the row in the interface.

(→ Open "Calculate_Volume" → Right-click on row 5 → Insert row)

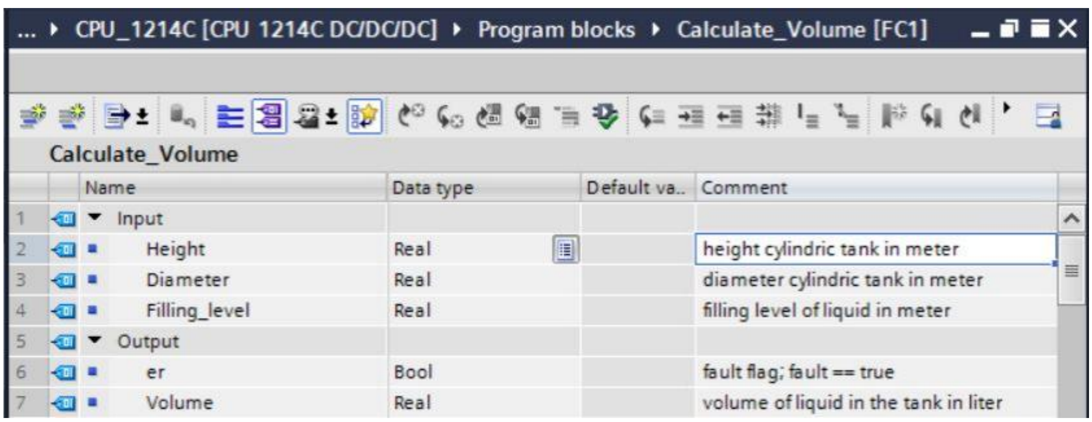


13

→ Enter the parameter "er" with data type BOOL and comment.

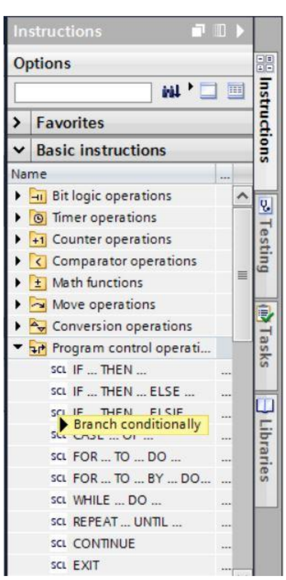


→ Follow the same steps to add the "Height" tag with data type Real and comment.



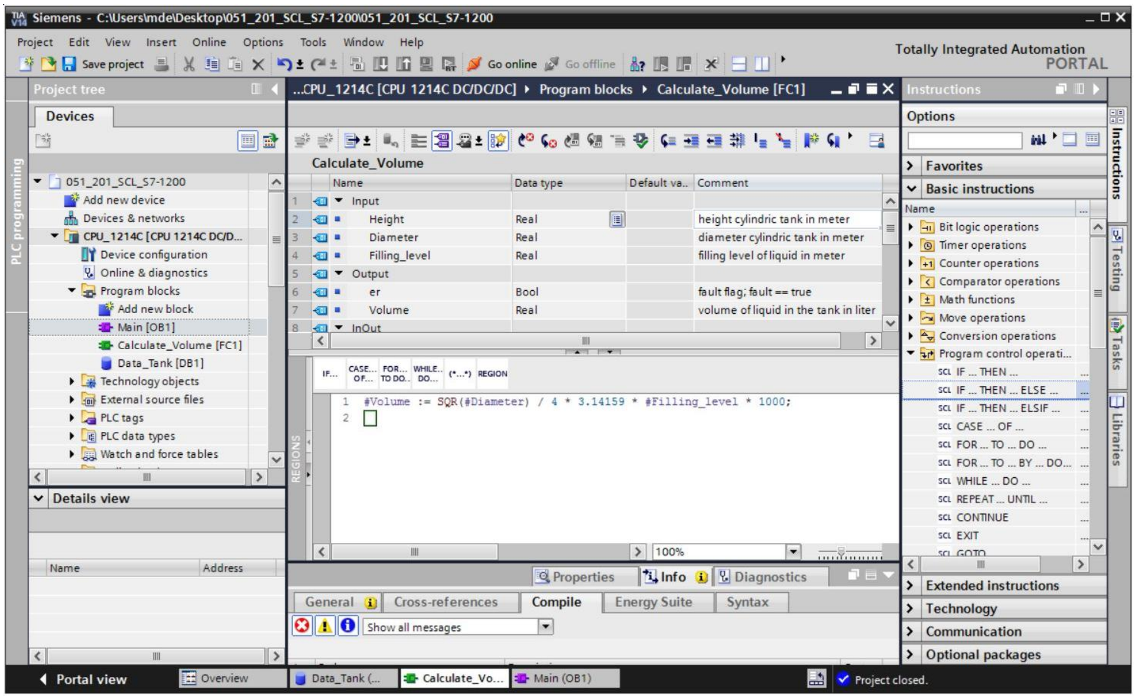
→ Then go to the "IF…THEN…ELSE" control statement from the "Program control operations" folder of Basic instructions.

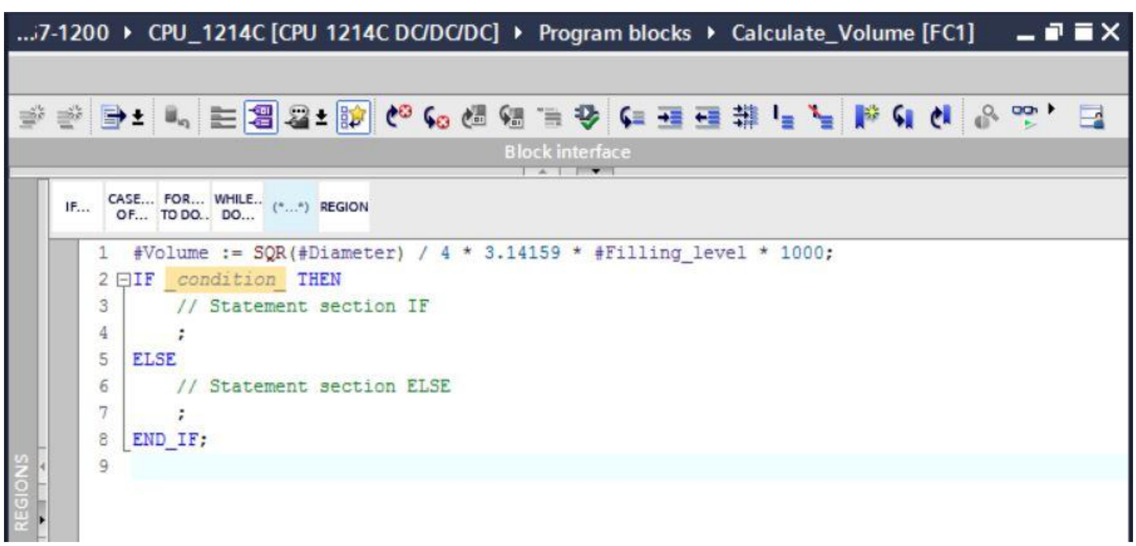
(→ Instructions → Basic instructions → Program control operations → "IF...THEN…ELSE")



13

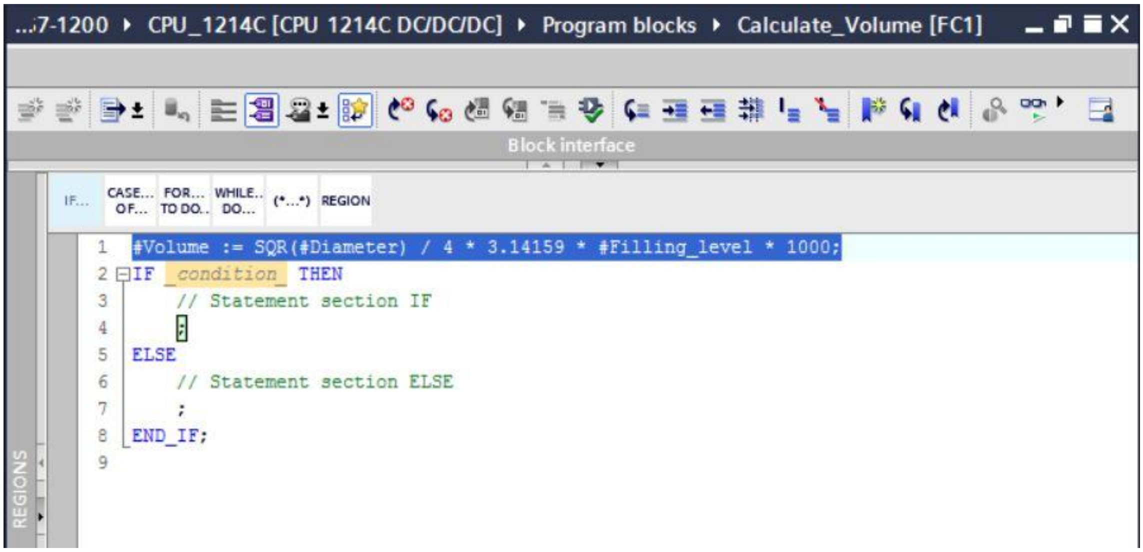
→ Then drag the "IF...THEN...ELSE" control statement to the second row of the program. (→ "IF…THEN…ELSE" → drag & drop)

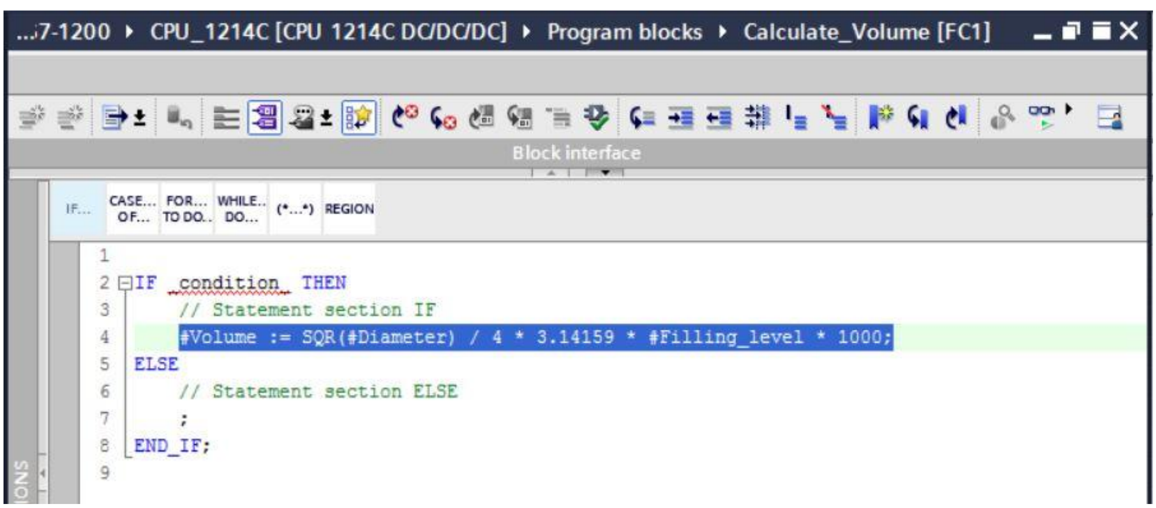




→ Highlight the mathematical formula and move it onto the semicolon in front of the ELSE using drag & drop.

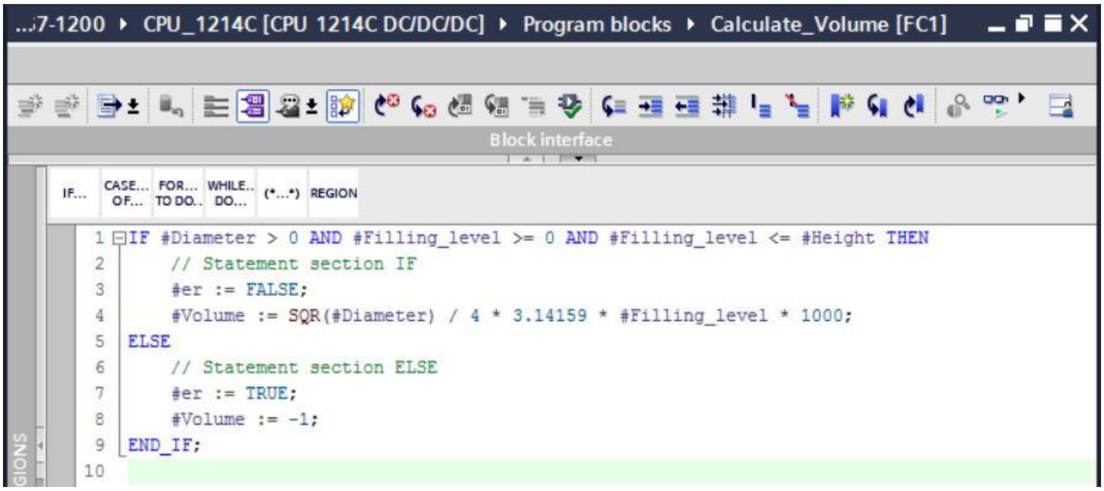
(→ Select → drag & drop)





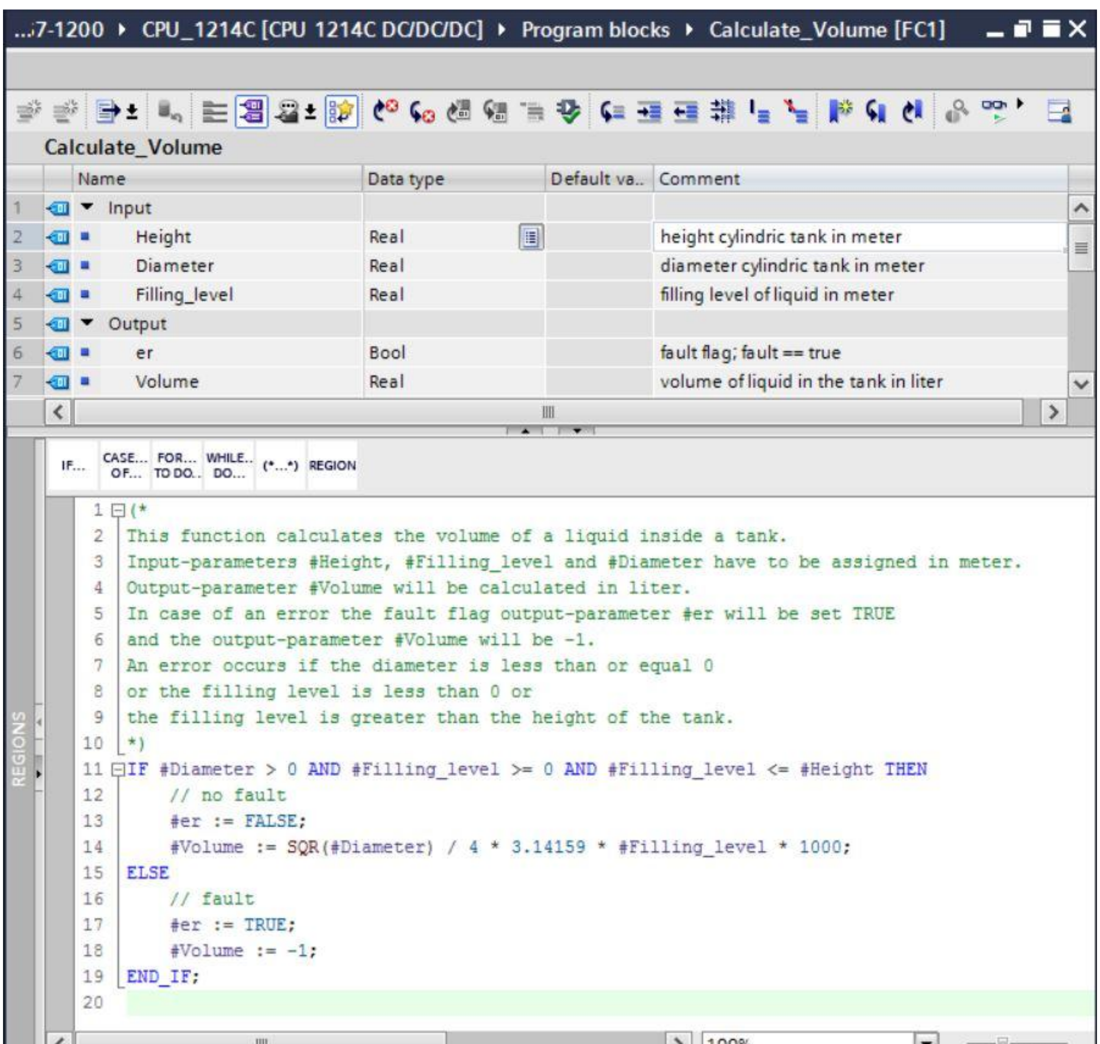
→ Complete the function and check your program by compiling it.

(→ Complete program → )



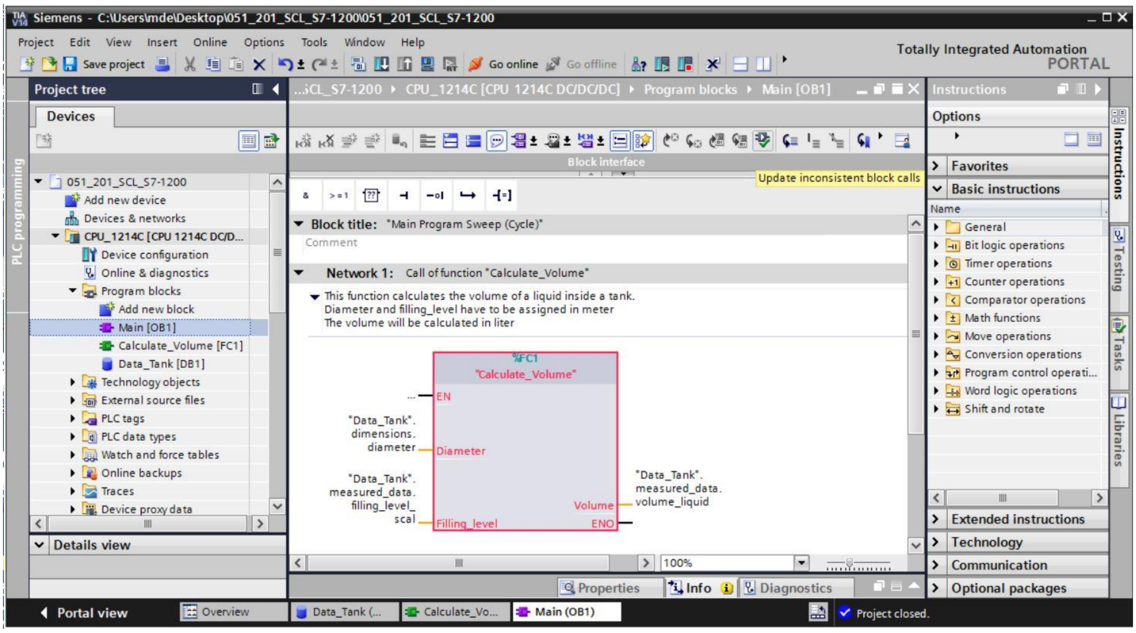
→ Comments can be added with "(**)" as block comment and with "//" as row comment. You can now add comments to your program.

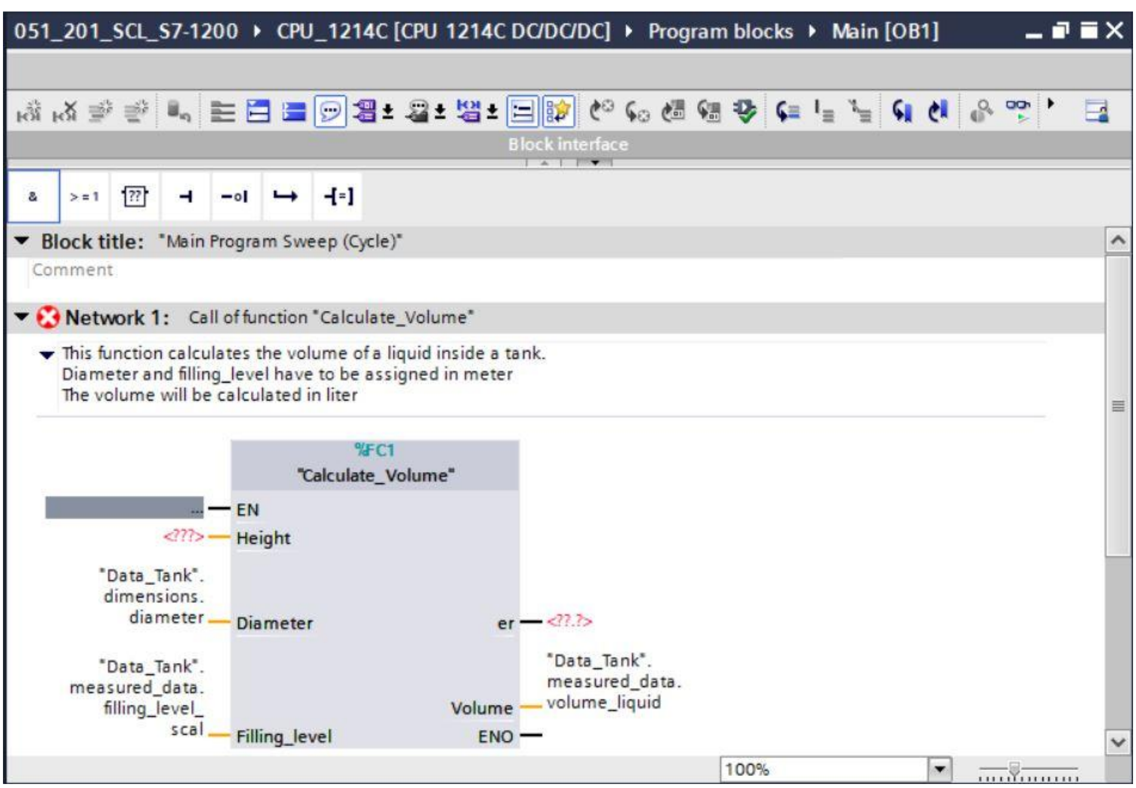
(→ Insert block comment starting with row 1 → Insert row comments in rows 12/16)



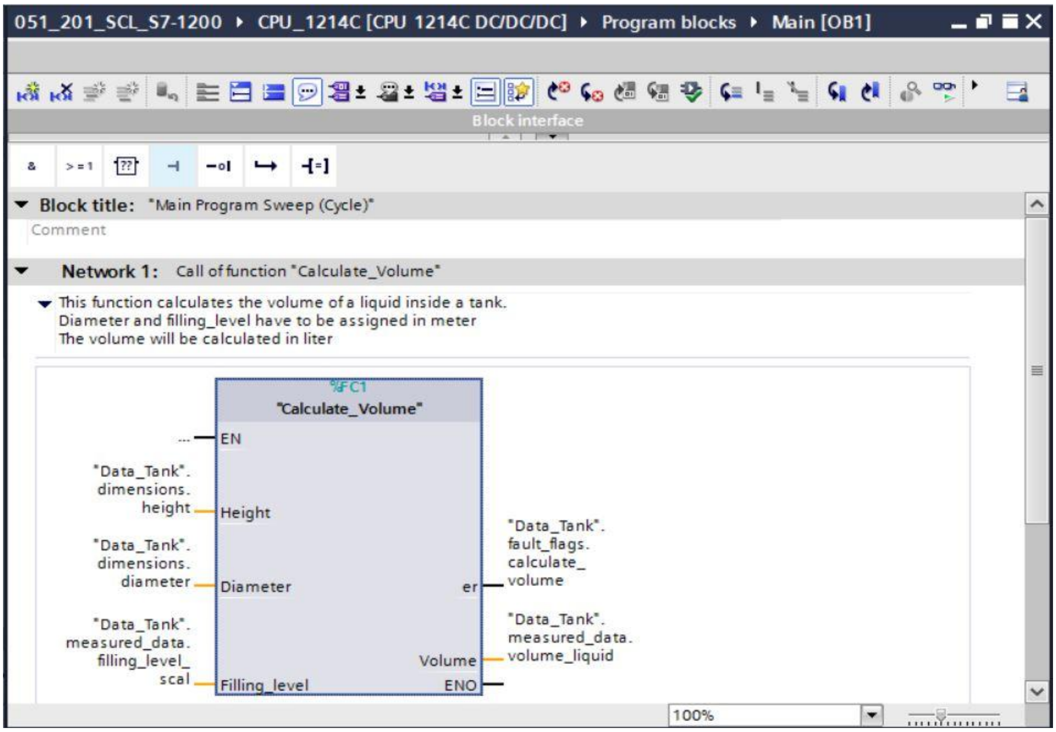
7.11 Customizing the organization block

→ Open OB1 and update the inconsistent block calls by clicking . (→ Open OB1 →)



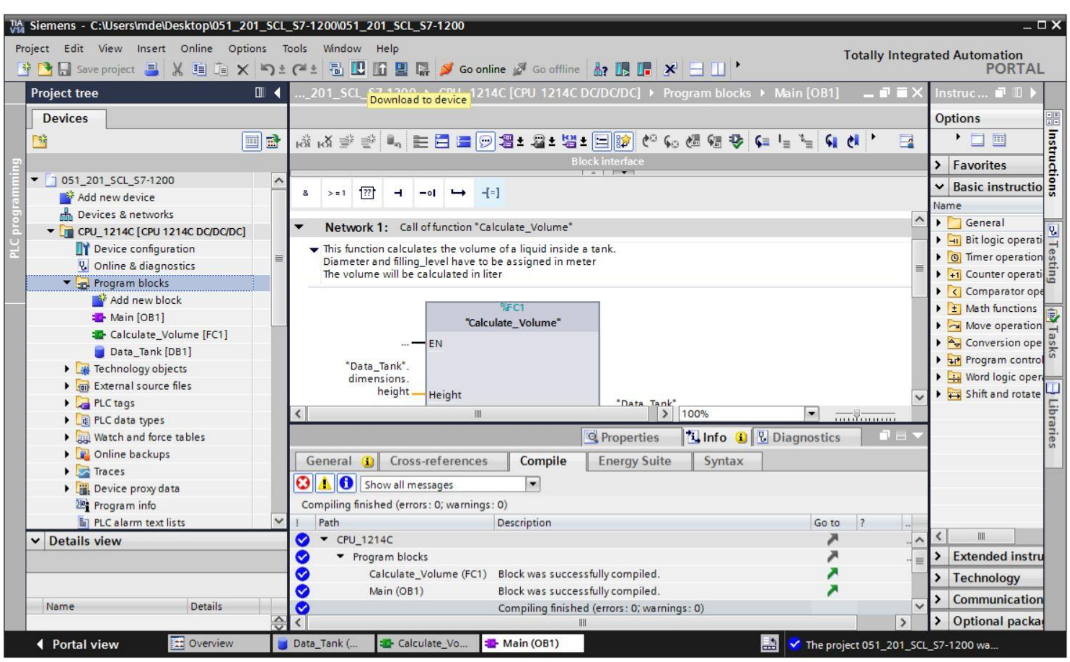


→ To do this, add the parameters "er" and "Height".



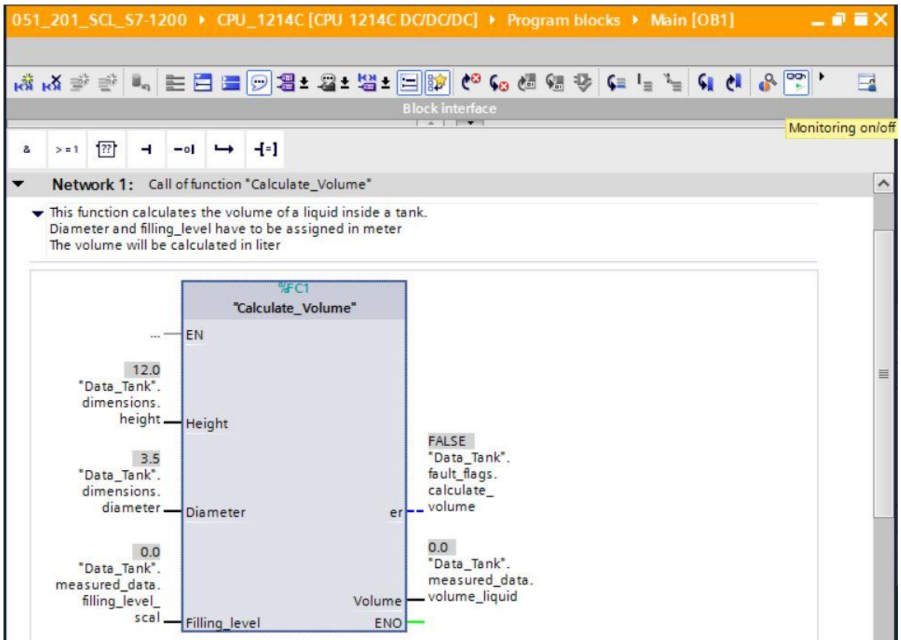
7.12 Compiling, saving and downloading the program

→ Click the "Program blocks" folder, compile the entire program and then save it. After successful compilation and saving, download the project to the controller. (→ Program blocks →  →  → )

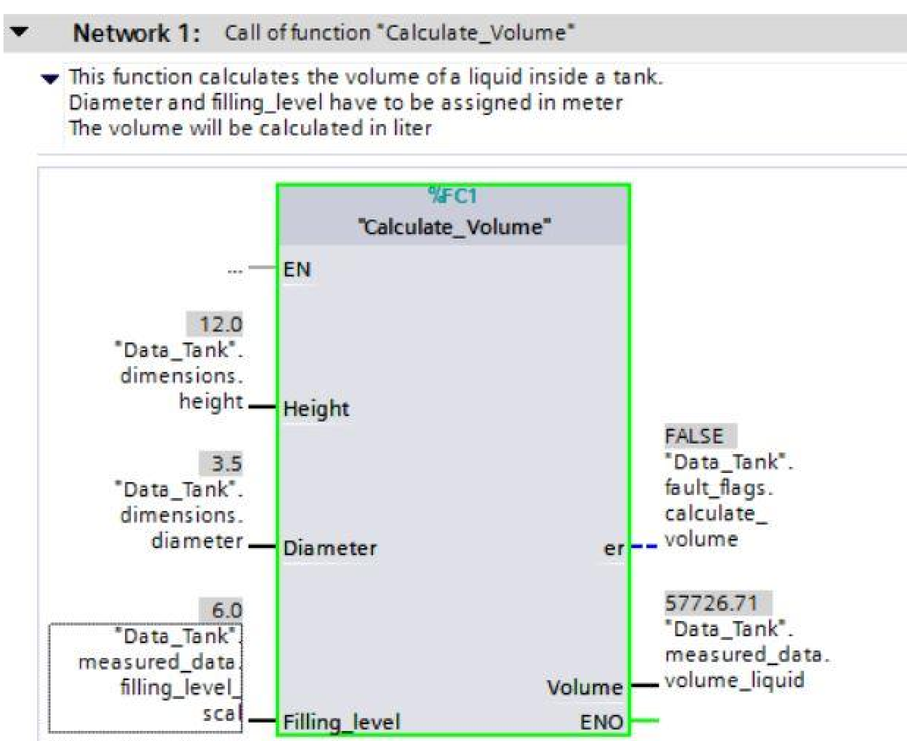


7.13 Monitoring and testing the organization block

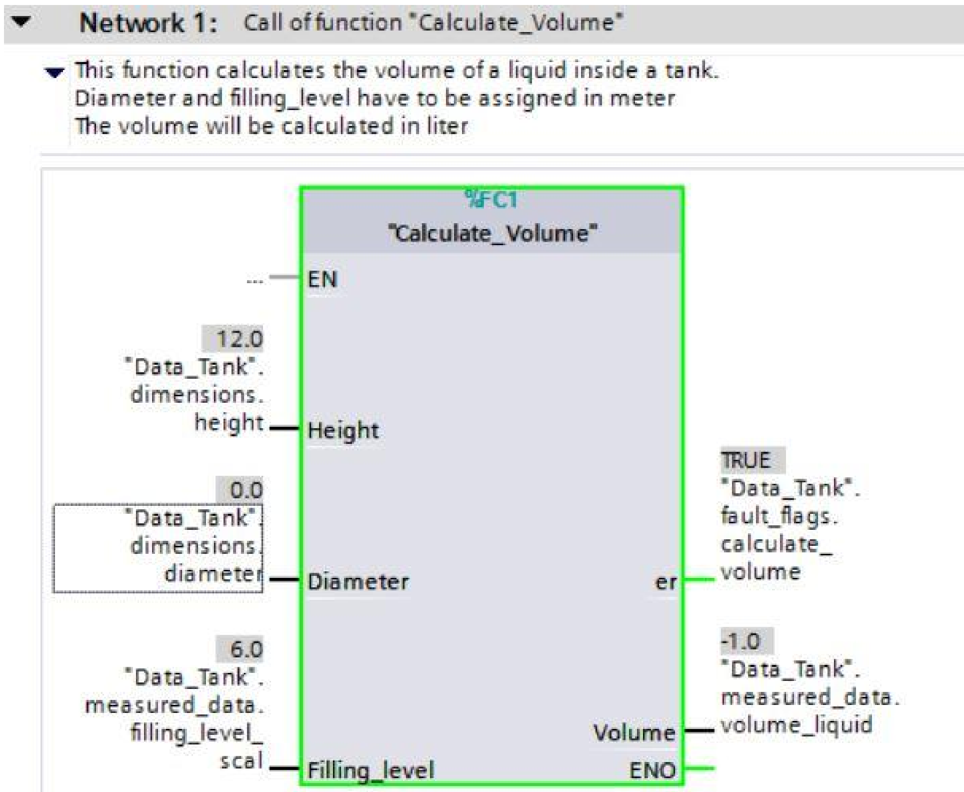
→ In the open OB1, click the icon to monitor the block.



→ Test your program by writing a value to the "Filling_level_scal" tag in the data block. (→ Right-click on "Filling_level_scal" → "Modify" menu → Modify operand → Enter value 6.0 → OK → Check)

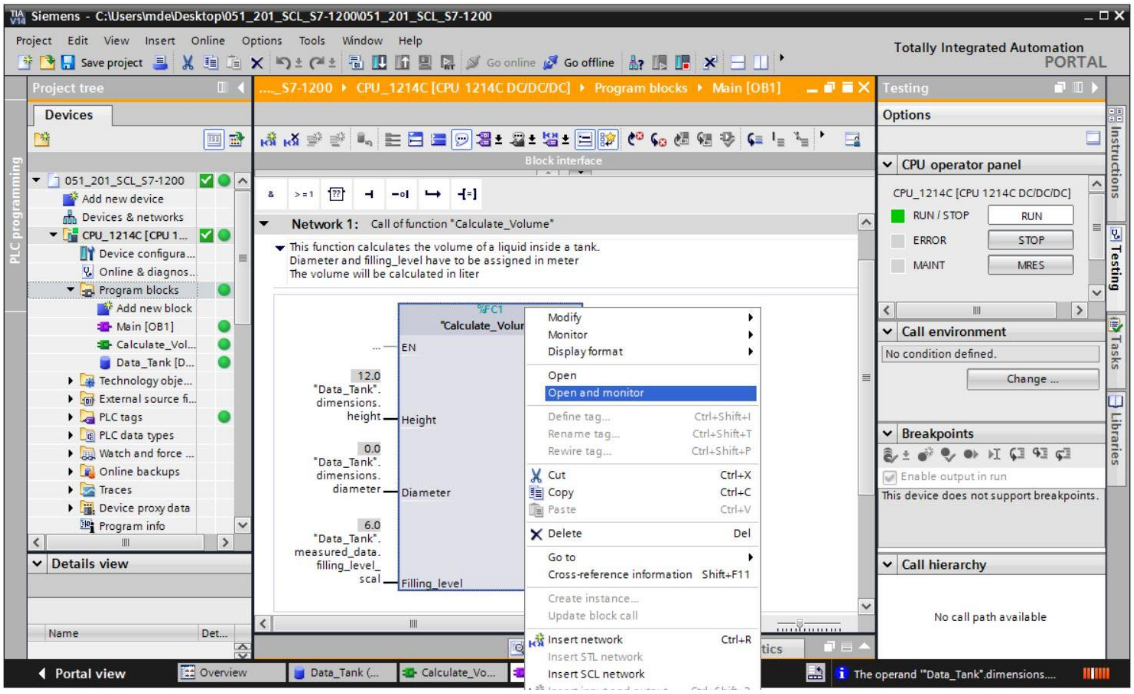


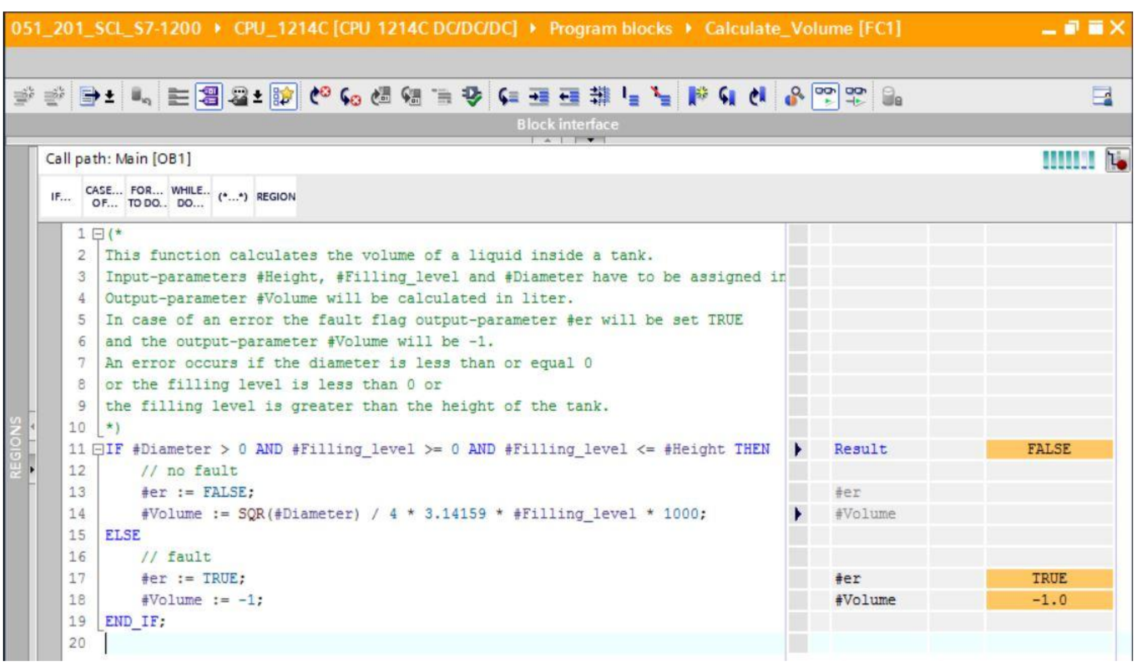
→ Now test if an error is output by setting the diameter to zero.
(→ Right-click on "Diameter" → "Modify" menu → Modify operand → Enter value 0.0 → OK → Check)



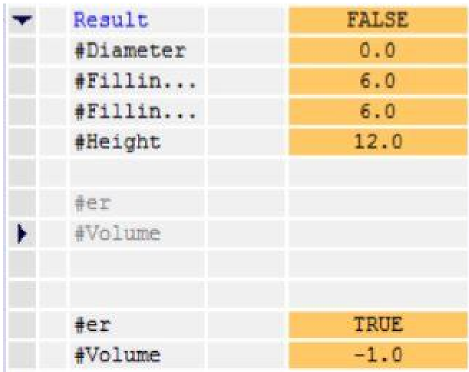
7.14 Monitoring and testing the "Calculate_Volume" function

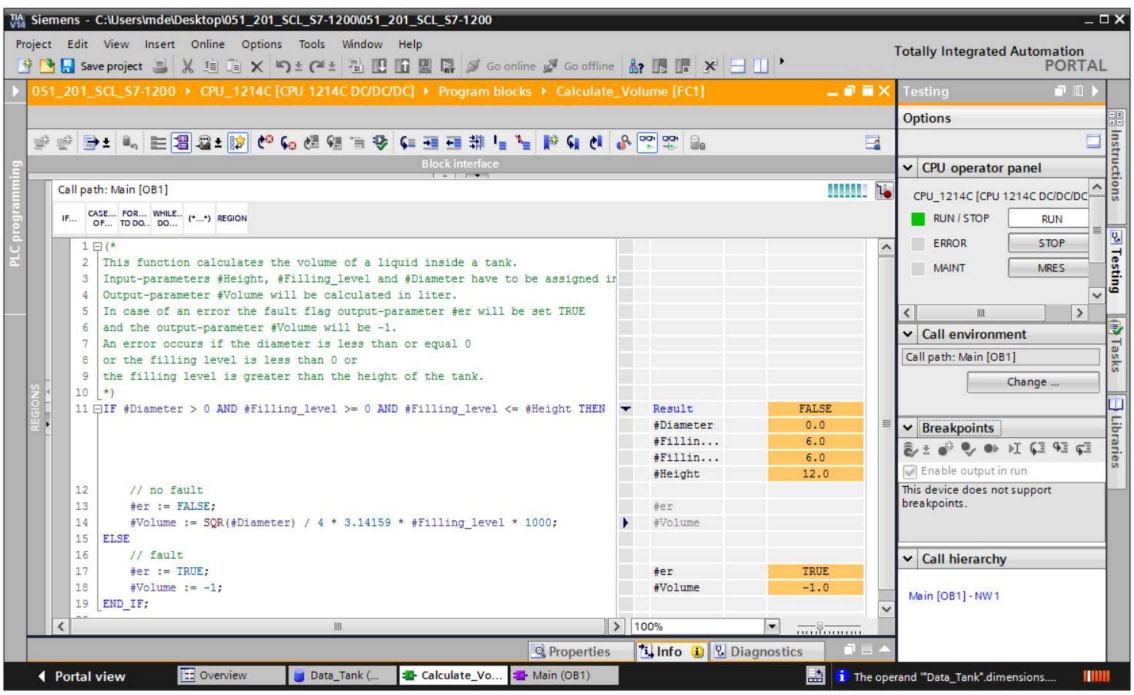
→ Finally, open and monitor the "Calculate_Volume" function by right-clicking the function and selecting the "Open and monitor" menu command. (→ Right-click on function → Open and monitor)



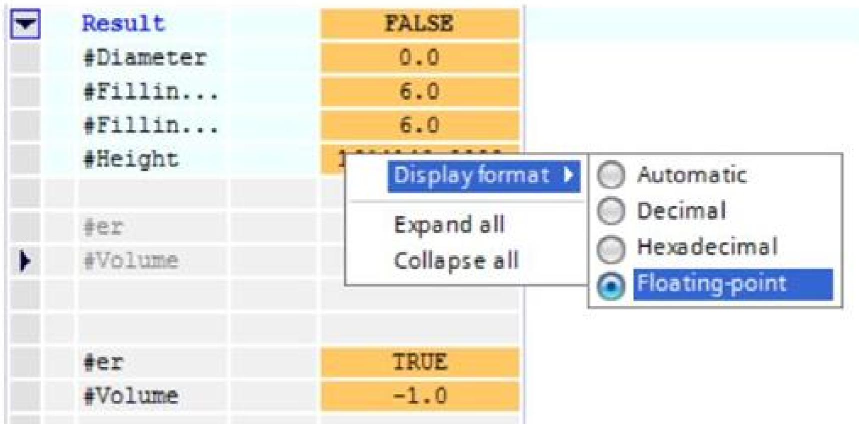


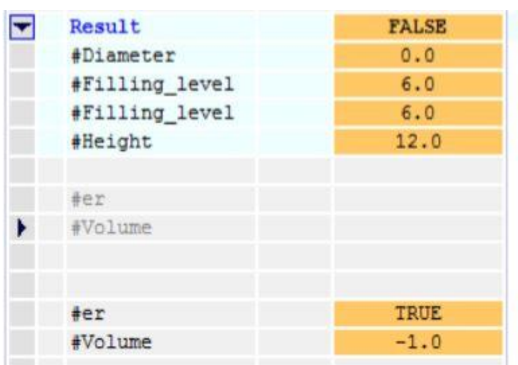
→ You can show the values of the individual tags of the IF query by clicking the black arrow ▼. (→ ▼)



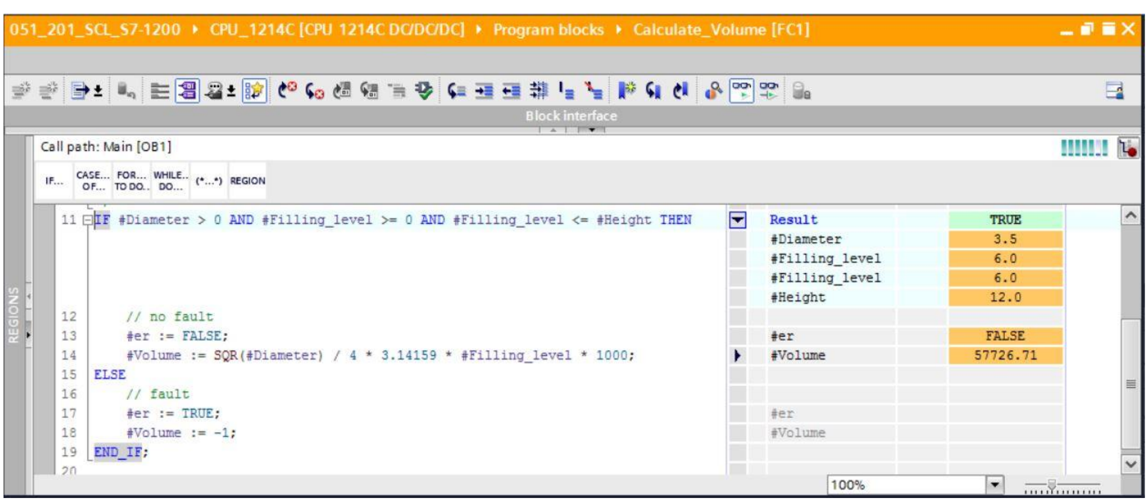


→ Right-click the tag to adjust the display format. (→ Right-click tag → Display format → Floating point)





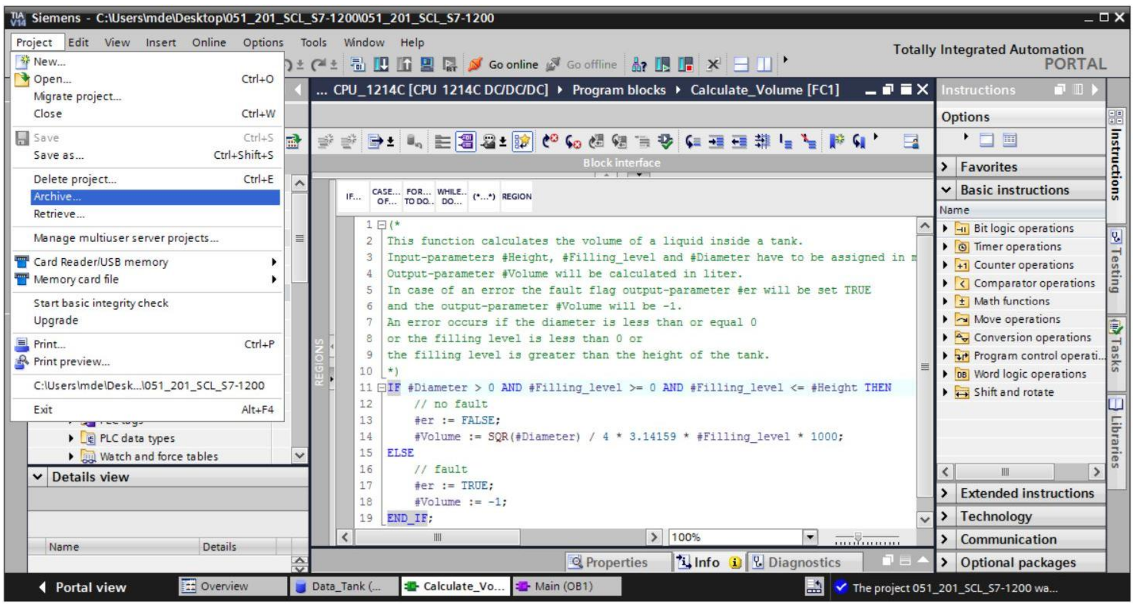
→ Now test the other branch of the IF branch by modifying the diameter in OB1 back to 3.5 meters. (→ Open OB1 → Modify diameter to 3.5 → Open and monitor function)



13

7.15 Archiving the project

→ Finally, the complete project is to be archived. Select → "Project" → "Archive …" in the menu. Open the folder in which you want to archive your project and save it as file type "TIA Portal Project archive". (→ Project → Archive → TIA Portal Project archive → File name: SCE_EN_051-201 SCL_S7-1200… → Archive)



8 Checklist

No.	Description	Checked
1	Successful compilation without error message	
2	Successful download without error message	
3	Modify operand (Diameter = 0.0) Result tag Volume= -1 Result tag "er" = TRUE	
4	Modify operand (Diameter = 3.5 and Level_scal = 0) Result Volume = 0 Result tag "er" = FALSE	
5	Modify operand (Filling_level_scal= 6.0) Result Volume = 57726.72 Result tag "er" = FALSE	
6	Modify operand (Filling_level_scal= 12.0) Result Volume = 115453.4 Result tag "er" = FALSE	
7	Modify operand (Filling_level_scal= 14.0) Result Volume = -1 Result tag "er" = TRUE	
8	Project successfully archived	

9 Exercise

9.1 Task description – Exercise

In this exercise you are going to program a "Scaling" function. The program is to be generally applicable to any positive analog values. In our example task "Tank", the filling level is read by an analog sensor and stored as a scaled value in the data block using this function.

In case of an error, the block is to set the error flag "er" to TRUE and set the parameter "Analog_scal" to zero as a result. An error exists when the "mx" parameter is less than or equal to "mn".

The function must contain the following parameters.

Input	**Data type**	**Comment**
Analog_per	INT	Analog value of the IO between 0..27648
mx	REAL	Maximum of the new scale
mn	REAL	Minimum of the new scale
Output		
er	BOOL	Error flag, no error = 0, error = 1
Analog_scal	REAL	Analog value scaled between mn..mx In case of an error = 0

The following formula is used to solve the task:

$$\#\text{Analog\_scal} = \frac{\#\text{Analog\_per}}{27648} \bullet (\#\text{mx} - \#\text{mn}) + \#\text{mn}$$

An analog signal is required for this task. The operand used for this task must be entered in the PLC tag table.

13

Name	**Data type**	**Address**	**Comment**
B1	INT	%IW64	Filling level between 0..27648

9.2 Planning

Now solve this task on your own.

9.3 Checklist – Exercise

No.	Description	Checked
1	Operand added to PLC tag table	
2	Function FC: "Scaling" created	
3	Interface defined	
4	Function programmed	
5	"Scaling" function added to network 1 of OB1	
6	Input tags connected	
7	Output tags connected	
8	Successful compilation without error message	
9	Successful download without error message	
10	Analog value for filling level set to zero Result Filling_level_scal = 0 Result er = FALSE	
11	Analog value for filling level set to 27648 Result Filling_level_scal = 12.0 Result er = FALSE	
12	Analog value for filling level set to 13824 Result Filling_level_scal = 6.0 Result er = FALSE	
13	Modify operand (mx = 0.0) Result Filling_level_scal = 0 Result tag er = TRUE	
14	Project successfully archived	

13

10 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

siemens.com/sce/s7-1200

Preview "Additional information"

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

Notes

13

SCE Learn-/Training Textbook

Automation System
SIMATIC S7-1200

TIA Portal Modules
from Version V14 SP1

Matching SCE Trainer Packages for these Learn-/Training Document

- **SIMATIC S7-1200 AC/DC/RELAY (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC (set of 6) "TIA Portal"**
  Order no.: 6ES7214-1AE30-4AB3
- **Upgrade SIMATIC STEP 7 BASIC V14 SP1 (for S7-1200) (set of 6) "TIA Portal"**
  Order no.: 6ES7822-0AA04-4YE5

Please note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is provided at: siemens.com/sce/tp

Continued training

For regional Siemens SCE continued training, please contact your regional SCE contact siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding use

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

14

Table of contents

14

PID Controller for the SIMATIC S7-1200

1 Goal

In this chapter, you will become acquainted with the use of software PID controllers for the SIMATIC S7-1200 with the TIA Portal programming tool.

The module explains the call-up, connection, configuration and optimization of a PID controller for the SIMATIC S7-1200. It also shows the steps for calling the PID controller in the TIA Portal and integrating it into a user program.

The SIMATIC S7 controllers listed in Chapter 3 can be used.

2 Prerequisite

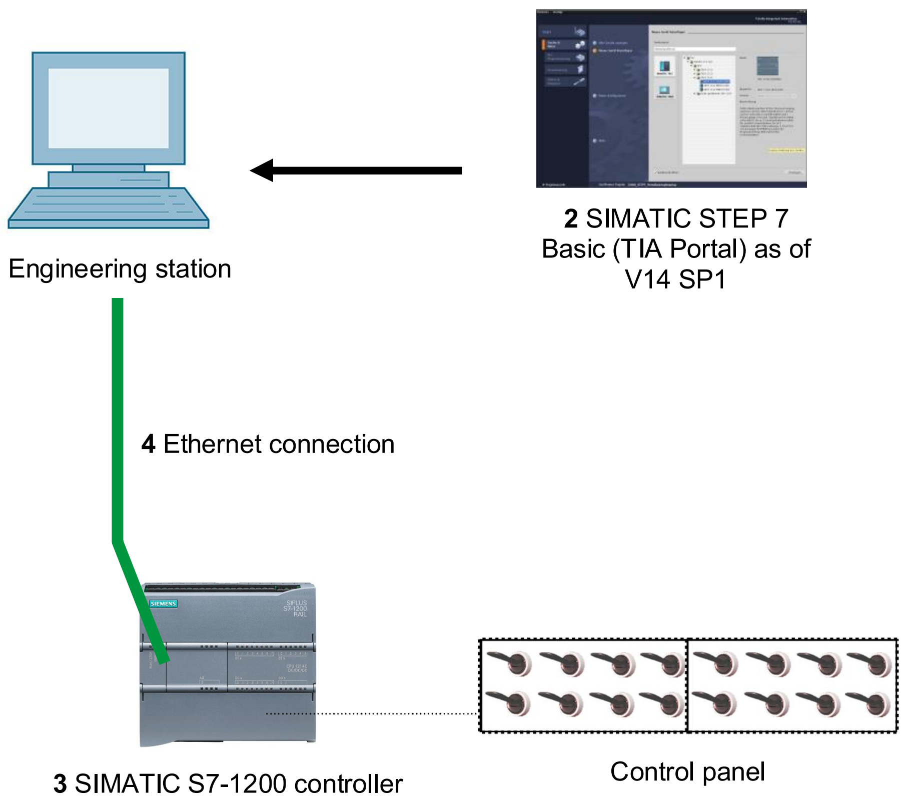
This chapter builds on the chapter Analog Values with the SIMATIC S7 CPU1214C DC/DC/DC. You can use the following project for this chapter, for example: "SCE_EN_031-500_Analog_Values_S7-1200.zap14“.

14

3 Required hardware and software

1. Engineering station: requirements include hardware and operating system
(for additional information, see Readme on the TIA Portal Installation DVDs)
2. SIMATIC STEP 7 Basic software in TIA Portal – as of V14 SP1
3. SIMATIC S7-1200 controller, e.g. CPU 1214C DC/DC/DC with ANALOG OUTPUT SB1232 signal board, 1 AO – Firmware as of V4.2.1

   Note: The digital inputs and analog inputs and outputs should be fed out to a control panel.
4. Ethernet connection between engineering station and controller



4 Theory of closed loop controls

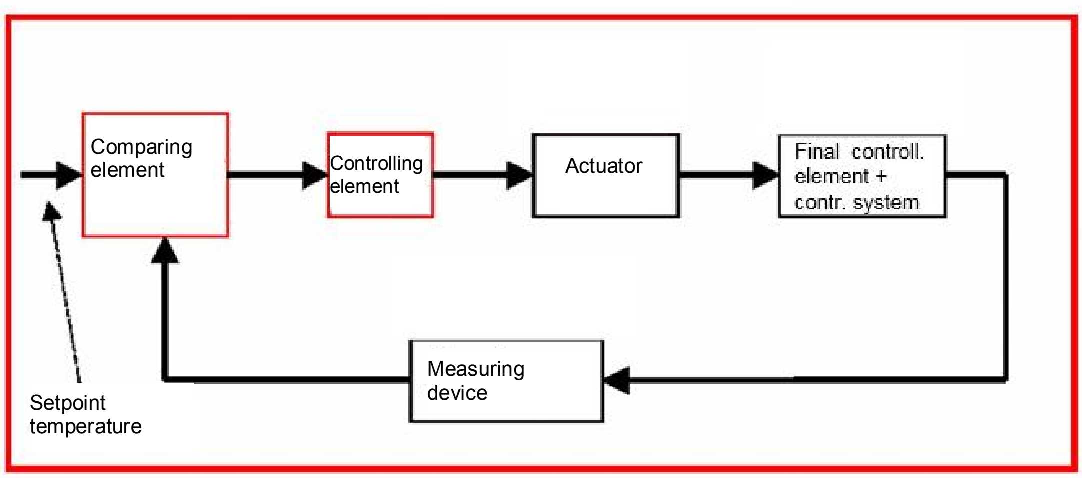
4.1 Tasks of closed loop controls

Closed loop control is a process in which the value of a variable is generated and maintained continuously through an intervention based on measurements of this variable.

This produces an action path that takes place in a closed loop – the control loop – because the process runs based on measurements of a variable that is, in turn, influenced by itself.

The variable to be controlled is continuously measured and compared with another preset variable of the same type. Depending on the result of this comparison, an adjustment of the variable to be controlled to the value of the preset variable is made.

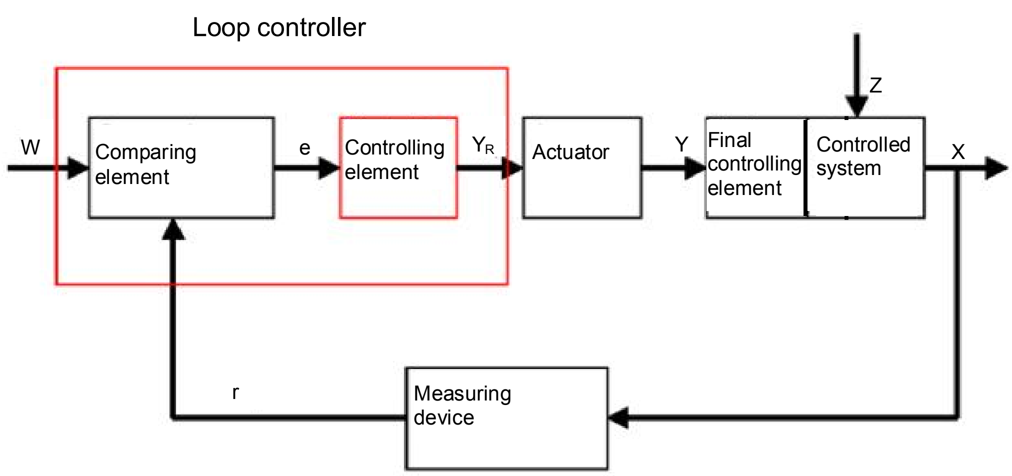
Diagrammatic representation of a closed loop control



4.2 Components of a control loop

The fundamental concepts of closed loop controls are explained in detail in the following.

An overview based on a diagram is presented here to start.



1. The controlled variable x

This is the actual "target" of the closed-loop control, namely the variable that is to be influenced or kept constant. In our example, this would be the room temperature. The instantaneous value of the controlled variable at a particular time is called the "actual value" at this time.

2. The feedback variable r

In a control loop, the controlled variable is continuously checked to enable a response to unwanted changes. The measured quantity proportional to the controlled variable is called the feedback variable. In the "Heating" example, it would correspond to the measured voltage of the inside thermometer.

3. The disturbance variable z

The disturbance variable is the variable that influences the controlled variable in an unwanted way and moves it away from the current setpoint. In the case of fixed setpoint control, this control is only necessary in the first place due to the existence of the disturbance variable. In the examined heating system, this would be, for example, the outside temperature or any other variable that causes the room temperature to move away from its ideal value.

4. The setpoint w

The setpoint at a given time is the value that the controlled variable should ideally have at this time. Note that the setpoint may vary continuously in a slave control. In our example, the setpoint would be the currently desired room temperature.

5. The comparing element

This is the point at which the current measured value of the controlled variable and the instantaneous value of the reference variable are compared. In most cases, both variables are measured voltages. The difference between the two variables is the "system error" e. This is passed to the controlling element and evaluated there (see below).

6. The controlling element

The controlling element is the actual heart of a closed loop control. It evaluates the system error, thus the information regarding whether, how and how much the controlled variable deviates from the current setpoint, as an input variable and derives from this the **"Controller output variable" $\mathbf{Y_R}$**, which is ultimately used to influence the controlled variable. In the heating system example, the controller output variable would be the voltage for the mixer motor.

The manner in which the controlling element determines the controller output variable from the system error is the main criterion of the closed-loop control.

7. The actuator

14

The actuator is, so to speak, the "executive organ" of the closed loop control. It receives information from the controlling element in the form of the controller output variable indicating how the controlled variable is to be influenced and translates this into a change of the "manipulated variable". In our example, this would be the mixer motor controller.

8. The final controlling element

This is the element of the control loop that influences the controlled variable (more or less directly) as a function of the **manipulated variable Y**. In the example, this would be the combination of the mixer, heating lines and radiators. The adjustment of the mixer (the manipulated variable) is made by the mixer motor (actuator) and influences the room temperature by means of the water temperature.

9. The controlled system

The controlled system is the system containing the variable to be controlled, thus the living space in the heating example.

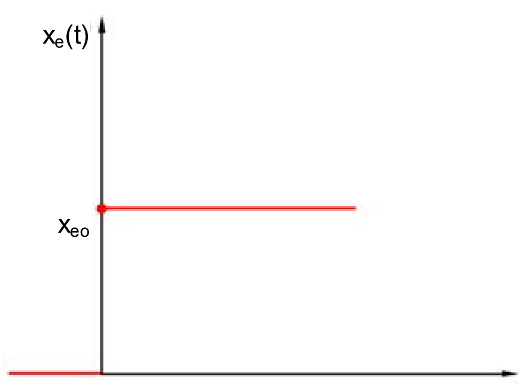
10. The dead time

The dead time refers to the time that elapses from a change in the controller output variable until there is a measurable response in the controlled system. In the example, this would be the time between a change in the voltage for the mixer motor and a measurable change in the room temperature resulting from this.

4.3 Step function for analysis of controlled systems

To analyze the response of controlled systems, controllers and control loops, a uniform function for the input signal is used – the step function.

Depending on whether a control loop element or the entire control loop is being analyzed, the controlled variable x(t), the manipulated variable y(t), the reference variable w(t) or the disturbance variable z(t) can be assigned the step function. The input signal is often designated xe(t) and the output signal xa(t).



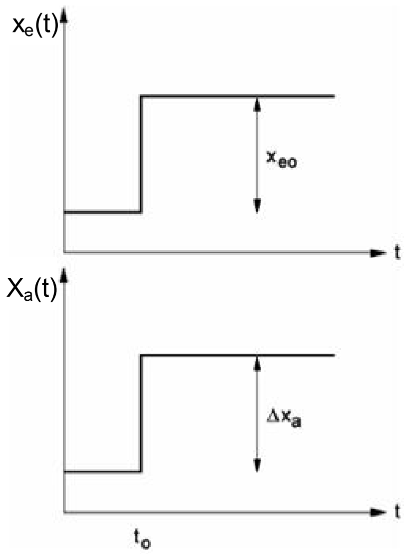
$$x_e(t) = \begin{cases} 0 & \text{for } t < 0 \\ x_{eo} & \text{for } t \geq 0 \end{cases}$$

14

4.4 Controlled systems with self-regulation

4.4.1 Proportional system without time delay

This controlled system is called a P system for short.



sudden change of the input variable at i t_0

Controlled variable/manipulated variable:

$$x = K_{ss} \bullet y$$

Kss: Proportional coefficient for a manipulated variable change:

$$K_{ss} = \frac{\Delta x}{\Delta y} = \tan\alpha$$

Controlled variable/disturbance variable:

$$x = K_{sz} \bullet z$$

Ksz: Proportional value for a disturbance variable change

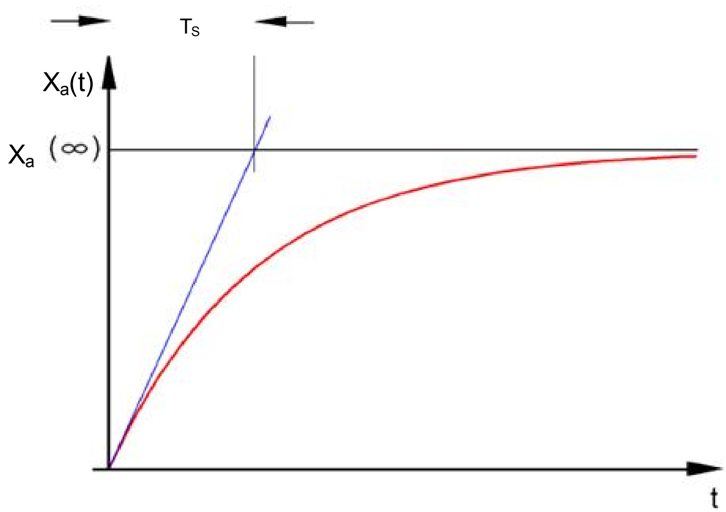
<u>Range</u>: $y_h = y_{max} - y_{min}$

<u>Control range</u>: $x_h = x_{max} - x_{min}$

14

4.4.2 Proportional system with time delay

This controlled system is called a P-T1 system for short.



Differential equation for a general input signal $x_e(t)$:

$T_S \bullet x_a(t) + x_a(t) = K_{PS} \bullet x_e(t)$

Solution of the differential equation for a step function at the input (step response)

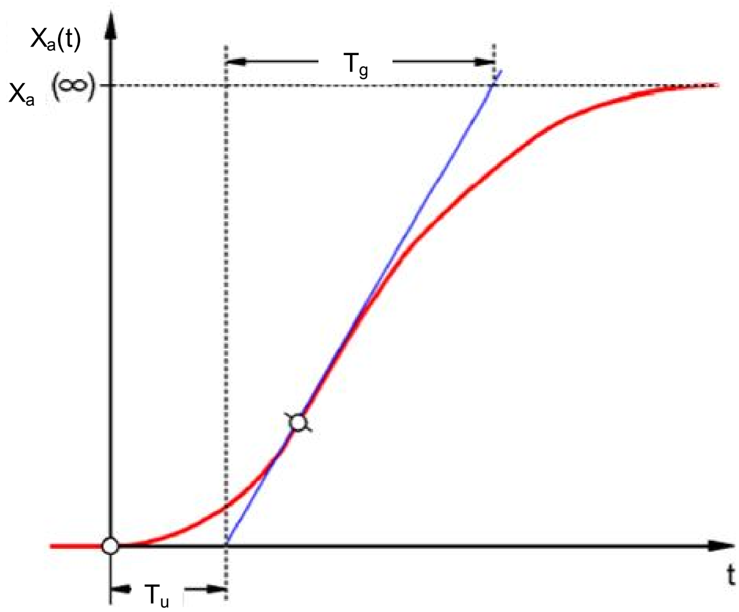
$x_a(t) = K_{PS}\ (1\text{-}e^{-t/TS}) \bullet x_{eo}$

$x_a\ (t = \infty) = K_{PS} \bullet x_{eo}$

T_S: Time constant

4.4.3 Proportional system with two time delays

This system is called a P-T2 system for short.



Tu: Delay time Tg: Compensation time

The system is generated through the reaction-free series connection of two P-T1 systems that have the time constants TS1 and TS2.

Controllability of P-Tn systems:

$\frac{T_u}{T_g} < \frac{1}{10} \rightarrow$ easily controllable

$\frac{T_u}{T_g} \approx \frac{1}{6} \rightarrow$ still controllable

$\frac{T_u}{T_g} > \frac{1}{3} \rightarrow$ difficult to control

With the increasing ratio Tu/Tg, the system becomes less and less controllable.

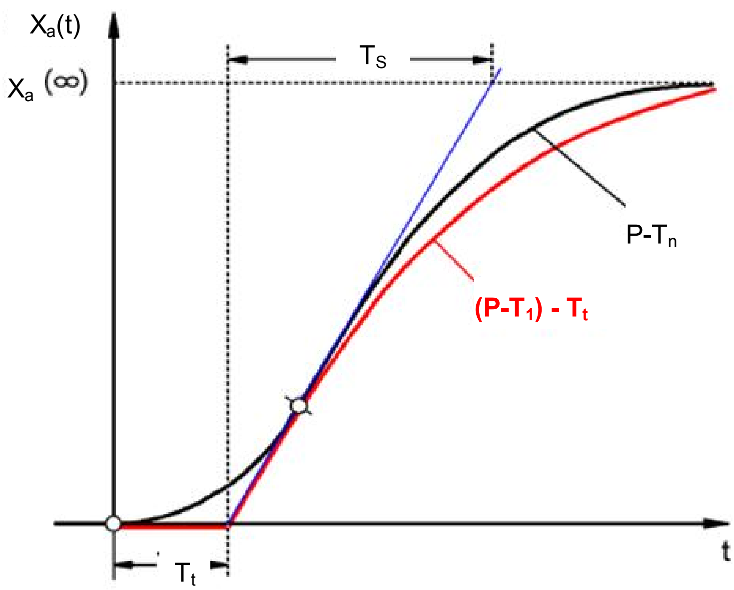
4.4.4 Proportional system with n time delays

This controlled system is called a P-Tn system for short.

The time response is described by an nth order differential equation. The step response characteristic is similar to that of the P-T2 system. The time response is described by Tu and Tg.

Substitute: An approximate substitution for the system with many delays is the series connection of a P-T1 system with a dead time system.

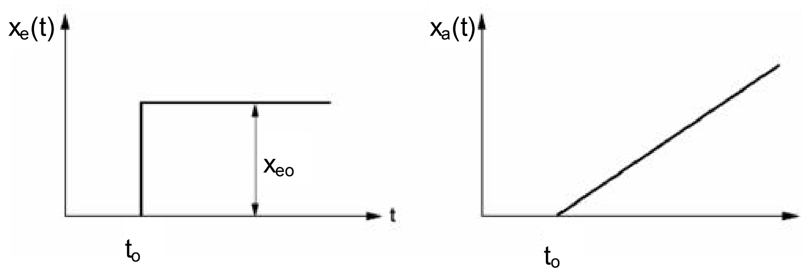
The following applies: Tt » Tu and TS » Tg.



4.5 Systems without self-regulation

This controlled system is called an I system for short.

After a disturbance, the controlled variable continues increasing steadily without striving for a fixed final value.

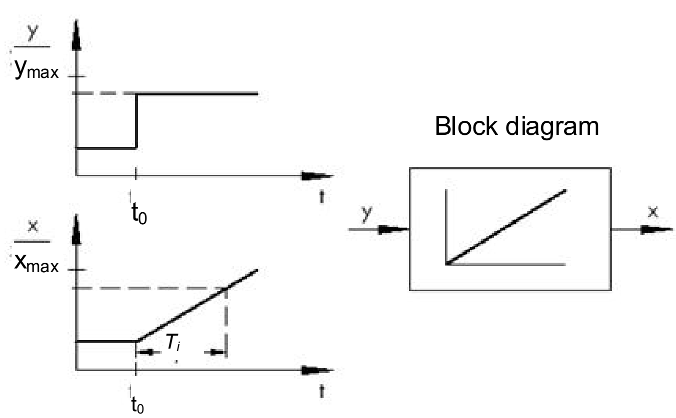


Example: Level control

For a tank with discharge outlet, whose incoming and outgoing flow rates are the same, there is a constant fill height. If the incoming or outgoing flow rate changes, the liquid level rises or falls. The level changes faster as the difference between the incoming flow rate and outgoing flow rate increases.

It is clear from this example that, in practice, the integral action has a limit in most cases. The controlled variable increases or decreases only until a system-inherent limit value is reached. A tank runs over or drains dry, pressure reaches the system maximum or minimum, etc.

The figure shows the time response of an I system to a step change in the input variable as well as the derived block diagram:



If the step function at the input changes to a function xe(t), then

$$x_a(t)=K_{IS}\int x_e(t)\,dt \Rightarrow \text{integrating controlled system}$$

K_{IS}: Integral coefficient of the controlled system

* Figure from SAMSON Technical Information - L102 Controllers and Controlled Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

4.6 *Basic types of continuous controllers*

Discrete controllers that only switch one or two manipulated variables on and off have the advantage of simplicity. Both the controller itself and the actuator and final controlling element are simpler in nature and thus less expensive than continuous controllers.

Discrete controllers have several disadvantages, however. For one thing, when large loads such as large electric motors or cooling units must be switched, high load peaks may occur at switch-on and overload the power supply, for example. For this reason, these often do not switch between "Off" and "On" but instead between full power ("full load") and a significantly lower power of the actuator or final controlling element ("base load"). Still, even with this improvement, a discrete closed-loop control is unsuitable for numerous applications. Consider an automobile engine whose speed is discreetly controlled. There would then be nothing between idle and full throttle. Apart from the fact that it would probably be impossible to properly transfer the forces from a sudden full-throttle to the road via the tires, such a vehicle would probably be unsuitable for road traffic.

Continuous controllers are therefore used for such applications. Theoretically, hardly any limits are placed on the mathematical relationship that establishes the controlling element between the system error and controller output variable. In practice, however, three classic basic types are differentiated. These will be described in more detail in the following.

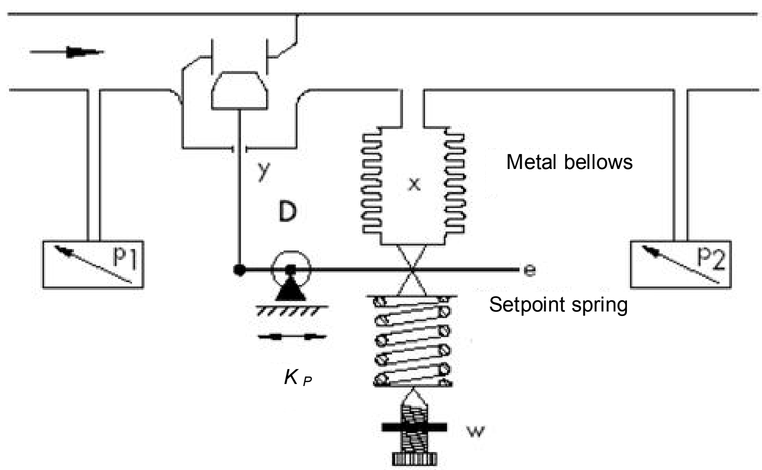
4.6.1 The proportional controller (P controller)

The manipulated variable y of a P controller is proportional to the measured error e. From this can be deducted that a P controller reacts to any deviation without lag and only generates a manipulated variable in case of system deviation.

The proportional pressure controller illustrated in the figure compares the force FS of the setpoint spring with the force FB created in the elastic metal bellows by the pressure p2. When the forces are off balance, the lever pivots about point D. This changes the position of the valve plug –and, hence, the pressure p2 to be controlled –until a new equilibrium of forces is restored.

The dynamic behavior of the P controller after a step change in the error variable is shown in the figure. The amplitude of the manipulated variable y is determined by the error e and the proportional-action coefficient Kp:

To keep the control deviation as small as possible, as large a proportional-action coefficient as possible must be selected. An increase in the factor causes the controller to react faster, but if the value is too high there is a risk of overshooting and a large "hunting" tendency of the controller.

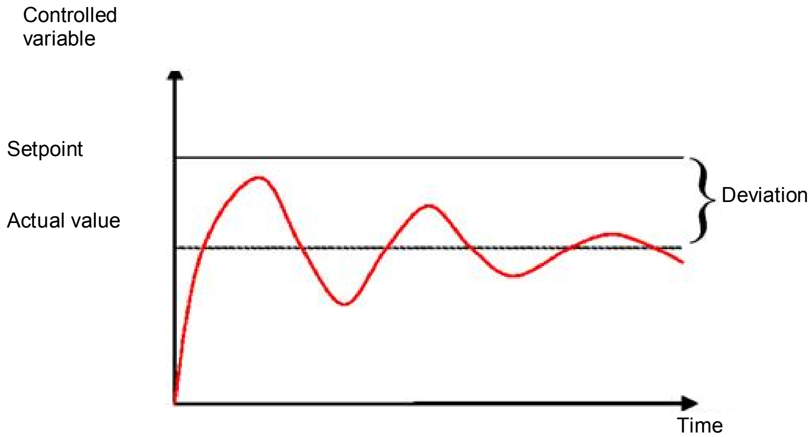


$$y = K_P \cdot e$$

14

* Figure and text from SAMSON Technical Information - L102 Controllers and Controlled Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

You see the response of the P controller in the diagram.



The advantages of this controller type lie, on the one hand, in its simplicity (in the simplest case, it can be implemented electronically with just a resistor) and, on the other hand, in its very prompt reaction compared to other controller types.

The main disadvantage of the P controller is its permanent system deviation. That is, the setpoint is never fully reached even over the long term. This disadvantage as well as the not yet ideal response speed cannot be minimized to a satisfactory extent through a larger proportional-action coefficient, because this leads to overshooting by the controller, or in other words an overreaction. In the worst case, the controller goes into a permanent oscillation in which the controlled variable is periodically moved away from the setpoint by the controller itself instead of by the manipulated variable.

The problem of permanent control deviation is best solved by an additional integral controller.

14

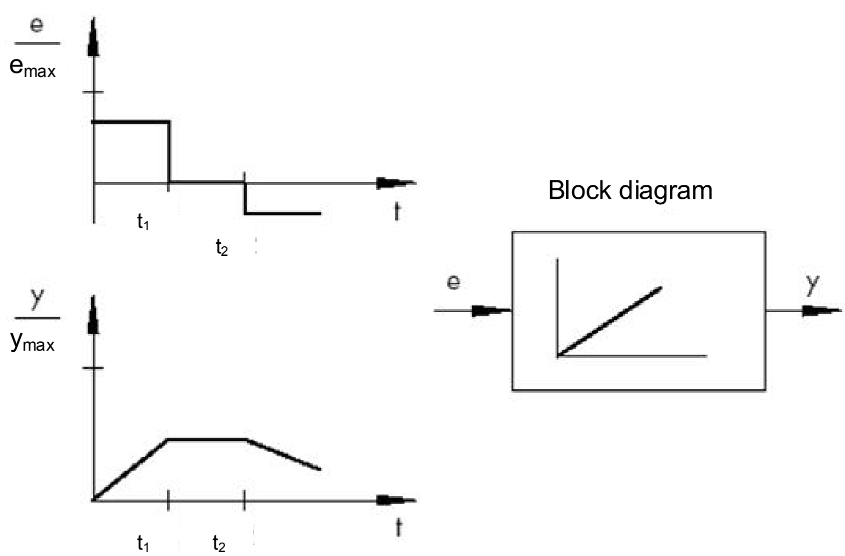
4.6.2 The integral controller (I controller)

Integral control action is used to fully correct system deviations at any operating point. As long as the error is nonzero, the integral action will cause the value of the manipulated variable to change. Only when reference variable and controlled variable are equally large –at the latest, though, when the manipulated variable reaches its system specific limit value (Umax, pmax, etc.)– is the control process balanced.

Mathematics expresses integral action as follows: the value of the manipulated variable is changed proportional to the integral of the error e.

$$y = K_i \int e \, dt \qquad \text{with} \qquad K_i = \frac{1}{T_n}$$

How rapidly the manipulated variable increases/decreases depends on the error and the integral time.

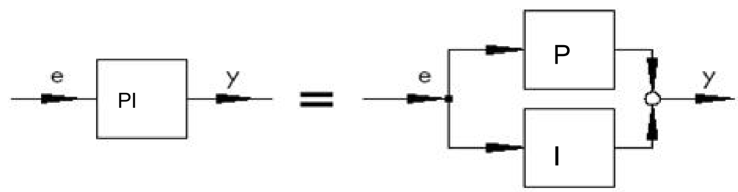


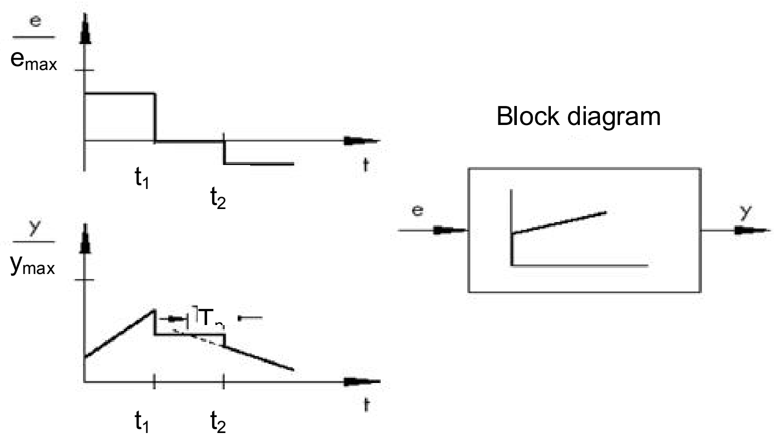
* Figure and text from SAMSON Technical Information - L102 Controllers and Controlled Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

4.6.3 The PI controller

PI controllers are often employed in practice. In this combination, one P and one I controller are connected in parallel.

If properly designed, they combine the advantages of both controller types (stability and rapidity; no steady-state error), so that their disadvantages are compensated for at the same time.





The dynamic behavior is marked by the proportional-action coefficient Kp and the reset time Tn. Due to the proportional component, the manipulated variable immediately reacts to any error signal e, while the integral component starts gaining influence only after some time. Tn represents the time that elapses until the I component generates the same control amplitude that is generated by the P component (Kp) from the start. As with I controllers, the reset time Tn must be reduced if the integral-action component is to be amplified.

Controller dimensioning:

By adjusting the Kp and Tn values, oscillation of the controlled variable can be reduced, however, at the expense of control dynamics.

PI controller applications: Fast control loops allowing no steady-state error

Examples: pressure, temperature. ratio control, etc.

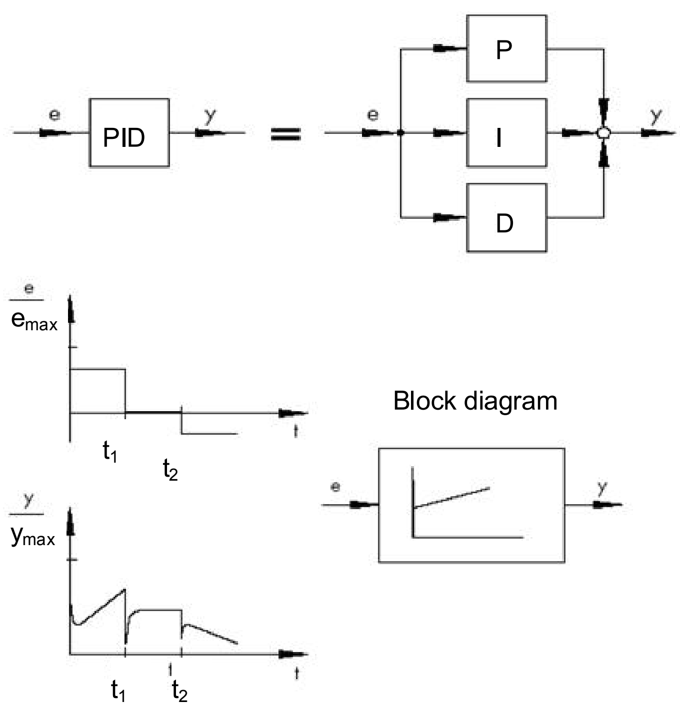
* Figure and text from SAMSON Technical Information - L102 Controllers and Controlled Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

4.6.4 The derivative controller (D controller)

D controllers generate the manipulated variable from the rate of change of the error and not – – as P controllers — from their amplitude. Therefore, they react much faster than P controllers: even if the error is small, derivative controllers generate– by anticipation, so to speak –large control amplitudes as soon as a change in amplitude occurs. A steady-state error signal, however, is not recognized by D controllers, because regardless of how big the error, its rate of change is zero. Therefore, derivative-only controllers are rarely used in practice. They are usually found in combination with other control elements, mostly in combination with proportional control.

4.6.5 The PID controller

If a D component is added to PI controllers, the result is an extremely versatile PID controller. As with PD controllers, the added D component –if properly tuned –causes the controlled variable to reach its setpoint more quickly, thus reaching steady state more rapidly.



$$y = K_P \cdot e + K_i \int e\,dt + K_D \frac{de}{dt} \quad \text{with} \quad K_i = \frac{K_P}{T_n};\ K_D = K_P \cdot T_V$$

* Figure and text from SAMSON Technical Information - L102 Controllers and Controlled Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

14

4.7 Controller tuning using the oscillation test

For a satisfactory control result, the selection of a suitable controller is an important aspect. It is even more important that the control parameters Kp, Tn and TV be appropriately adjusted to the system response. Mostly, the adjustment of the controller parameters remains a compromise between a very stable, but also very slow control loop and a very dynamic, but irregular control response which may easily result in oscillation, making the control loop instable in the end.

For nonlinear systems that should always work in the same operating point, e.g. fixed setpoint control, the controller parameters must be adapted to the system response at this particular operating point. If a fixed operating point cannot be defined, such as with follow-up control systems ñ, the controller must be adjusted to ensure a sufficiently rapid and stable control result within the entire operating range.

In practice, controllers are usually tuned on the basis of values gained by experience.

Should these not be available, however, the system response must be analyzed in detail, followed by the application of several theoretical or practical tuning approaches in order to determine the proper control parameters.

One approach is a method first proposed by Ziegler and Nichols, the so-called ultimate method. It provides simple tuning that can be applied in many cases. This method, however, can only be applied to controlled systems that allow sustained oscillation of the controlled variable.

For this method, proceed as follows:

– At the controller, set Kp and Tv to the lowest value and Tn to the highest value (smallest possible influence of the controller).

– Adjust the controlled system manually to the desired operating point (start up control loop).

– Set the manipulated variable of the controller to the manually adjusted value and switch to automatic operating mode.

– Continue to increase Kp (decrease Xp) until the controlled variable encounters harmonic oscillation. If possible, small step changes in the setpoint should be made during the Kp adjustment to cause the control loop to oscillate.

– Take down the adjusted Kp value as critical proportional-action coefficient Kp,crit. Determine the time span for one full oscillation amplitude as Tcrit, if necessary by taking the time of several oscillations and calculating their average.

– Multiply the values of Kp,crit and Tcrit by the values according to the table and enter the determined values for Kp, Tn and Tv at the controller.

	K_p	T_n	T_v
P	$0.50 \times K_{p.\ crit.}$	-	-
PI	$0.45 \times K_{p.\ crit.}$	$0.85 \times T_{crit.}$	-
PID	$0.59 \times K_{p.\ crit.}$	$0.50 \times T_{crit.}$	$0.12 \times T_{crit.}$

* Figure and text from SAMSON Technical Information - L102 Controllers and Controlled Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

14

4.8 Controller tuning with T_u-T_g approximation

The tuning of the controlled systems will be performed here using the example of a P-T2 system.

T_u-T_g approximation

The Ziegler-Nichols method and the Chien, Hrones and Reswick method are based on the T_u-T_g approximation in which the transfer coefficient of the system K_s, delay time T_u and balancing time T_g parameters are determined from the system step response.

The tuning rules, which are described below, are the result of experiments using analog computer simulations.

P-T_N systems can be described with sufficient accuracy with a so-called T_u-T_g approximation, that is, through approximation using a P-T_1-T_L system.

The starting point is the system step response with input step height K. The required parameters (transfer coefficient of the system K_s, delay time T_u and balancing time T_g) are determined as shown in the figure.

The transfer function must be measured up to the final steady-stated value (K*Ks) so that the transfer coefficient of the system Ks required for the calculation can be determined.

The main advantage of this method is that the approximation can also be used when an analytical description of the system is not possible.

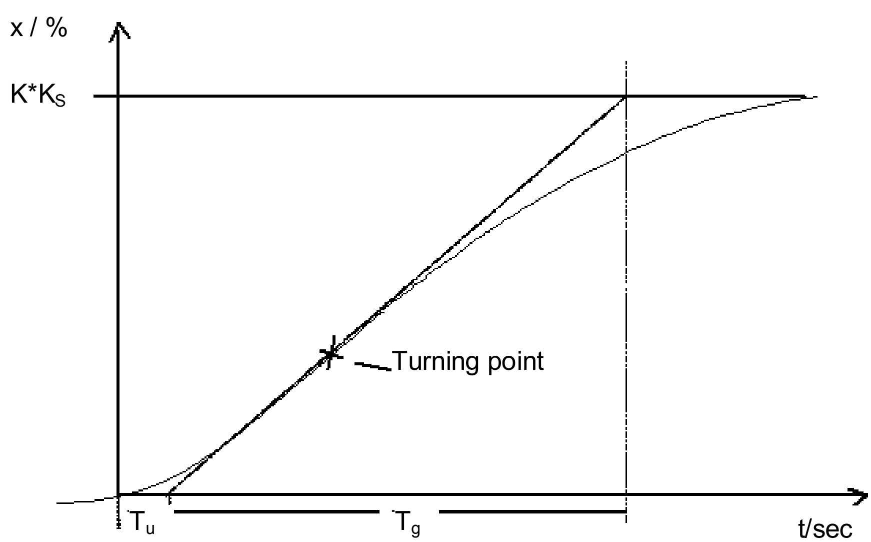


Figure: **T_u-T_g-Approximation**

14

4.8.1 Tuning the PI controller according to the Ziegler-Nichols method

Based on experiments on P-T_1-T_L systems, Ziegler and Nichols have identified the following optimal controller adjustments for fixed setpoint control:

$$K_{PR} = 0.9 \frac{T_g}{K_S T_u}$$

$$T_N = 3.33\ T_u$$

Use of these tuning values generally results in very good response to disturbances.

4.8.2 Tuning the PI controller according to the Chien, Hrones and Reswick method

Both the response to disturbances and response to setpoint changes were examined in order to achieve the most favorable controller parameters. Different values are yielded for the two cases. In addition, two different adjustments are specified in each case that meet different control performance requirements.

This resulted in the following adjustments:

- For response to disturbances:

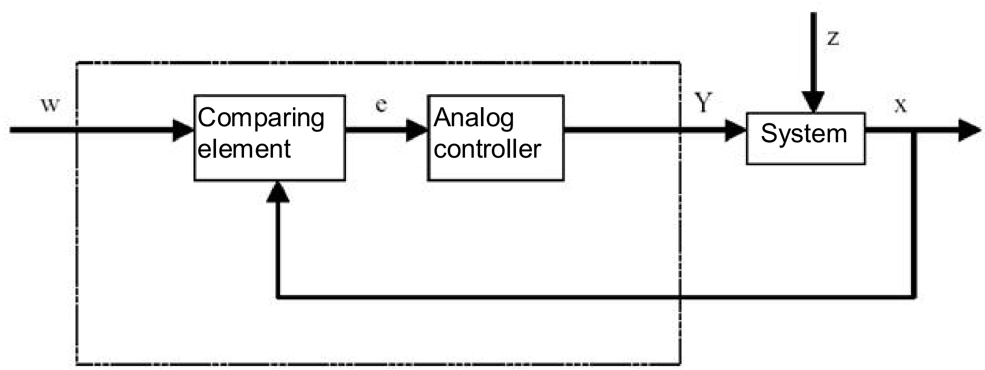
Aperiodic transient reaction with the shortest duration	20% overshoot minimum oscillation period
$K_{PR} = 0.6 \frac{T_g}{K_S T_u}$	$K_{PR} = 0.7 \frac{T_g}{K_S T_u}$
$T_N = 4\ T_u$	$T_N = 2{,}3\ T_u$

- For response to setpoint changes:

Aperiodic transient reaction with the shortest duration	20% overshoot minimum oscillation period
$K_{PR} = 0.35 \dfrac{T_g}{K_S T_u}$	$K_{PR} = 0.6 \dfrac{T_g}{K_S T_u}$
$T_N = 1.2\ T_g$	$T_N = T_g$

4.9 Digital controllers

Up to now, the main focus was on analog controllers, in other words, controllers that use the system error, which exists as an analog value, to derive the controller output variable in an analog manner. The diagram of this type of control loop is now well-known:

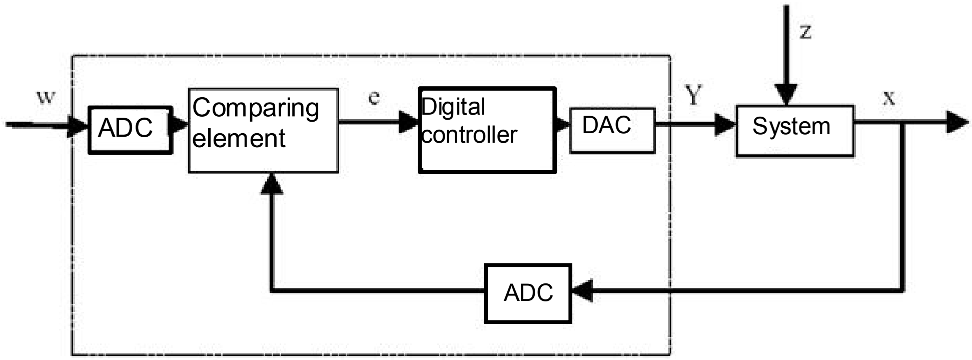


Often, however, it is advantageous to perform the actual evaluation of the system error digitally. For one thing, the relationship between the system error and controller output variable can be defined much more flexibly when it can be defined by an algorithm or formula that can be used in each case to program a computer than when it has to be implemented in the form of an analog circuit. For another, digital technology enables significantly greater integration of circuits so that multiple controllers can be accommodated in the smallest space. Finally, by dividing the computing time when there is a sufficient amount of computing capacity, it is even possible to use an individual computer as a controller for multiple control loops.

To enable digital processing of the variables, both the reference variable and the feedback variable are first converted to digital values in an analog-to-digital converter (ADC). These are then subtracted from one another by a digital comparing element and the difference is passed to the digital controlling element. Its controller output variable is then converted back to an analog value in a digital-to-analog converter (DAC). From the outside, the combined unit of converters, comparing element and controlling element resembles an analog controller.

14

We will examine the structure of a digital controller based on a diagram:



The advantages resulting from digital implementation of the controller are accompanied by various problems. For this reason, the size of some variables related to the digital controller must be chosen large enough to prevent the accuracy of the closed loop control from suffering too much from digitization.

Quality criteria for digital computers are:

– The quantization resolution of the digital-to-analog converter

  This specifies how fine the continuous value range is digitally mapped. The chosen resolution must be high enough that none of the finer points important for the closed loop control are lost.

– The sampling rate of the analog-to-digital converter.

  This is the frequency at which the analog values present at the converter are measured and digitized. This must be high enough that the controller can also still respond to step changes in the controlled variable in a timely manner.

– The cycle time

  Unlike an analog closed-loop controller, each digital computer works in clock cycles. The speed of the utilized computer must be high enough that a significant change of the controlled variable cannot occur during a single clock cycle (in which the output value is calculated and no input value is queried).

The performance of the digital controller must be high enough that its response is apparently as prompt and precise as an analog controller.

5 Task

In this chapter, a PID controller for speed control will be added to the program from chapter "SCE_EN_031-500 Analog Values_S7-1200". The call-up of the "MOTOR_SPEEDCONTROL" [FC10] function must be deleted for this.

6 Planning

The PID_Compact technology object is available in the TIA Portal for closed loop controls.

For closed-loop control of the motor speed, this technology object replaces the "MOTOR_SPEEDCONTROL" [FC10] block.

This will be carried out as an expansion of the "031-500_Analog_Values_S7-1200" project. This project must be retrieved from the archive beforehand.

The call-up of the "MOTOR_SPEEDCONTROL" [FC10] function must be deleted in the "Main" [OB1] organization block before the technology object can be called and connected in a cyclic interrupt OB.

The PID_Compact technology object must then be configured and commissioned.

6.1 PID_Compact closed-loop control block

The PID_Compact technology object provides a PID controller with integrated tuning for proportional-action final controlling elements.

The following operating modes are possible:

– Inactive

– Pretuning

– Fine tuning

– Automatic mode

– Manual mode

– Substitute output value with error monitoring

Here, the connection, parameter assignment and commissioning of this controller will be for automatic mode

During commissioning we will use the integrated tuning algorithms and record the control response of the controlled system.

The PID_Compact technology object is always called from a cyclic interrupt OB whose fixed set cycle time is 50 ms here.

The speed setpoint is set as a constant at the "Setpoint" input of the PID_Compact technology object in revolutions per minute (range: +/- 50 rpm). The data type is 32-bit floating-point number (Real).

The actual speed value -B8 (sensor actual value speed of the motor +/-10V corresponds to +/- 50 rpm) will be entered at the "Input_PER" input.

The output of the controller "Output_PER" will then be connected directly with signal -U1 (manipulated value speed of the motor in 2 directions +/- 10V corresponds to +/- 50 rpm).

The controller will only be active as long as output -Q3 (conveyor motor -M1 variable speed) is set. If this is not set, the controller will be deactivated by connection of the "Reset" input.

6.2 Technology diagram

Here you see the technology diagram for the task.

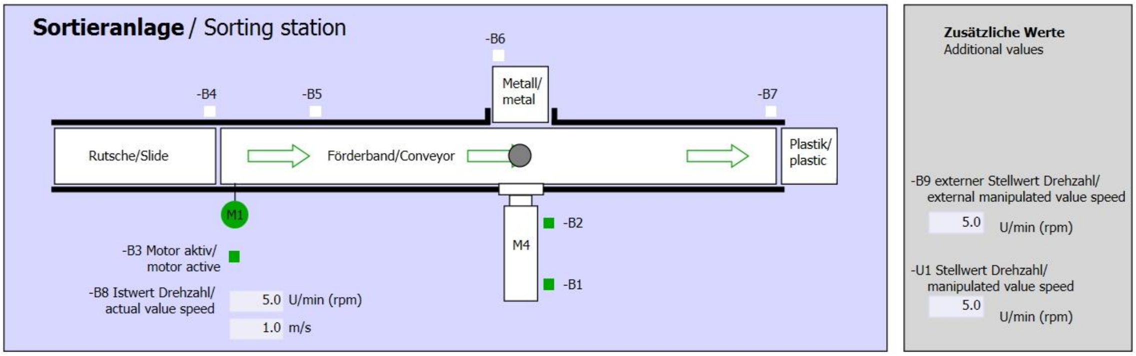


Figure 1: Technology diagram

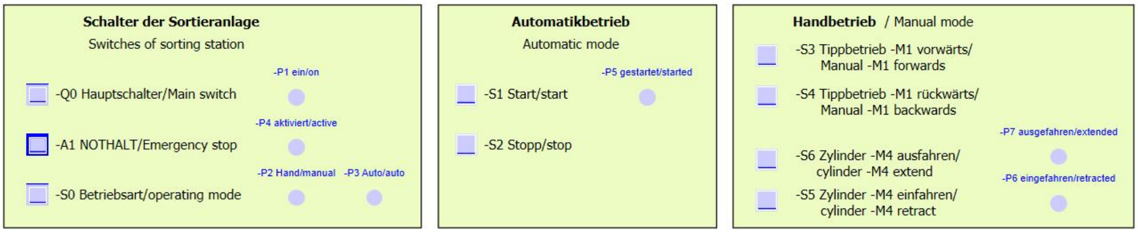


Figure 2: Control panel

14

6.3 Reference list

The following signals are required as global operands for this task.

DI	Type	Identifier	Function	NC/NO
I 0.0	BOOL	-A1	Return signal emergency stop OK	NC
I 0.1	BOOL	-K0	Main switch "ON"	NO
I 0.2	BOOL	-S0	Mode selector manual (0)/ automatic (1)	Manual = 0 Auto = 1
I 0.3	BOOL	-S1	Pushbutton automatic start	NO
I 0.4	BOOL	-S2	Pushbutton automatic stop	NC
I 0.5	BOOL	-B1	Sensor cylinder -M4 retracted	NO
I 1.0	BOOL	-B4	Sensor part at slide	NO
I 1.3	BOOL	-B7	Sensor part at end of conveyor	NO
IW64	BOOL	-B8	Sensor actual value speed of the motor +/-10V corresponds to +/- 50 rpm	

DO	Type	Identifier	Function	
Q 0.2	BOOL	-Q3	Conveyor motor -M1 variable speed	
QW 64	BOOL	-U1	Manipulated value speed of the motor in 2 directions +/- 10V corresponds to +/- 50 rpm	

Legend for reference list

DI	Digital Input	DO	Digital Output
AI	Analog Input	AO	Analog Output
I	Input	Q	Output
NC	Normally Closed		
NO	Normally Open		

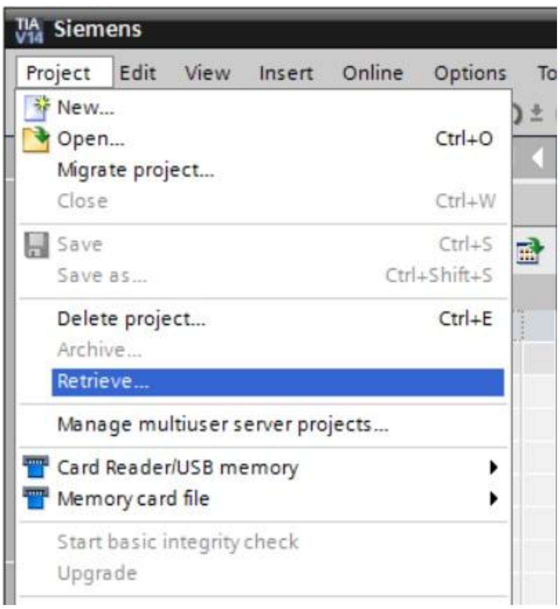
14

7 Structured step-by-step instructions

You can find instructions on how to carry out planning below. If you already have a good understanding of everything, it will be sufficient to focus on the numbered steps. Otherwise, simply follow the detailed steps in the instructions.

7.1 Retrieve an existing project

→ Before we can expand the "SCE_EN_031-500_Analog_Values_S7-1200.zap14" project from chapter "SCE_EN_031-500 Analog Values_S7-1200", we must retrieve this project from the archive. To retrieve an existing project that has been archived, you must select the relevant archive with → Project → Retrieve in the project view. Confirm your selection with Open. (→ Project → Retrieve → Select a .zap archive → Open)

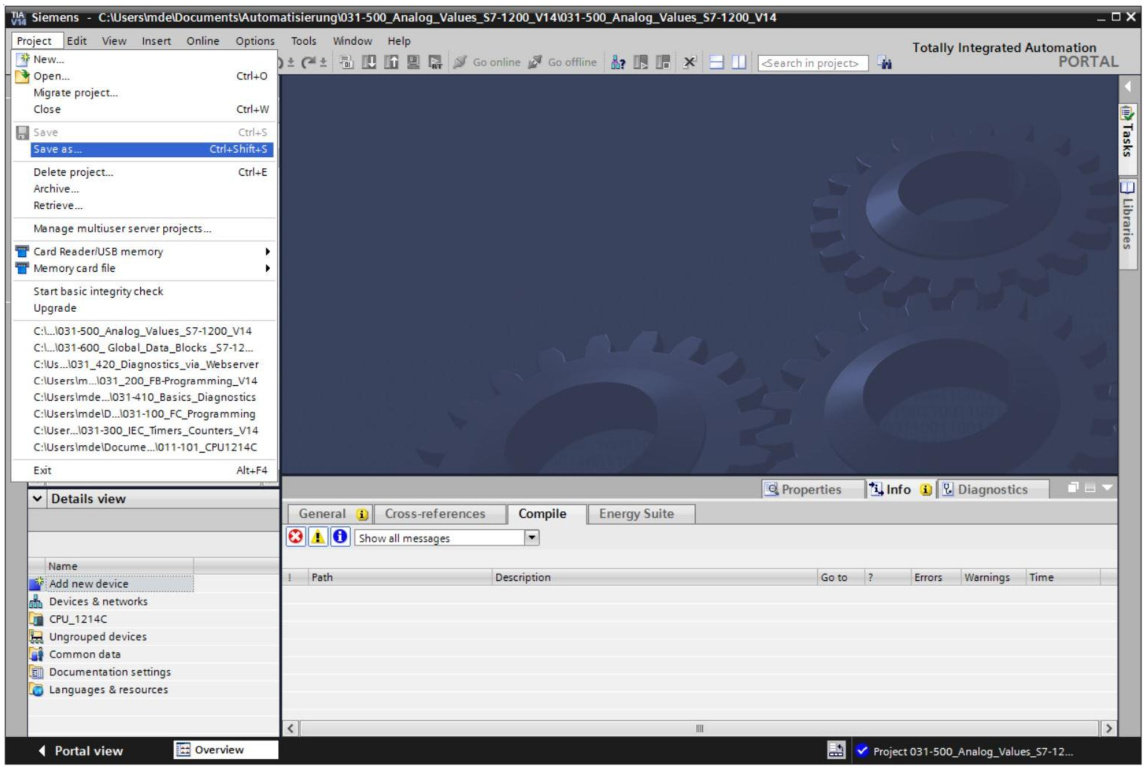


→ The next step is to select the target directory where the retrieved project will be stored. Confirm your selection with "OK".

(→ Target directory → OK)

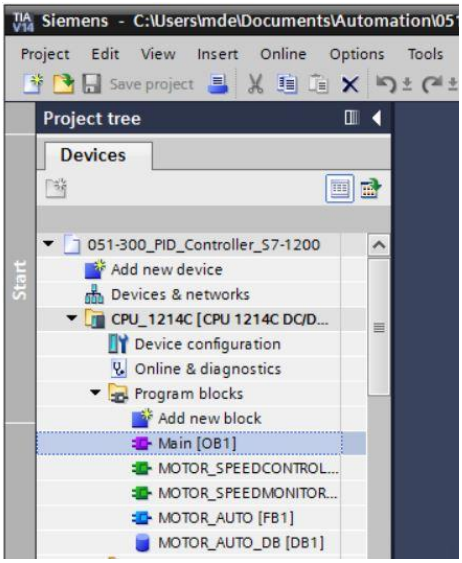
14

→ Save the opened project under the name 051-300_PID_Controller_S7-1200.
(→ Project → Save as … → 051-300_PID_Controller_S7-1200 → Save)



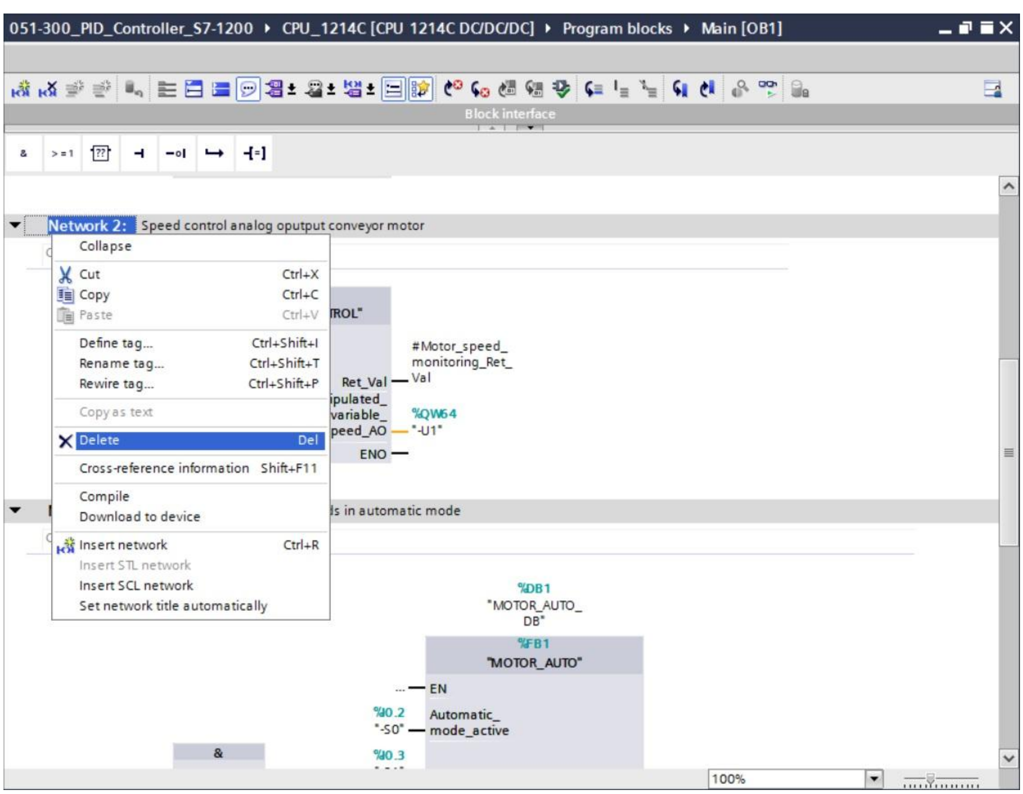
7.2 Call PID_Compact controller in a cyclic interrupt OB

→ Open the “Main” [OB1] organization block with a double-click.



→ Delete Network 2 with the no longer needed call-up of the "MOTOR_SPEEDCONTROL" [FC10] function.

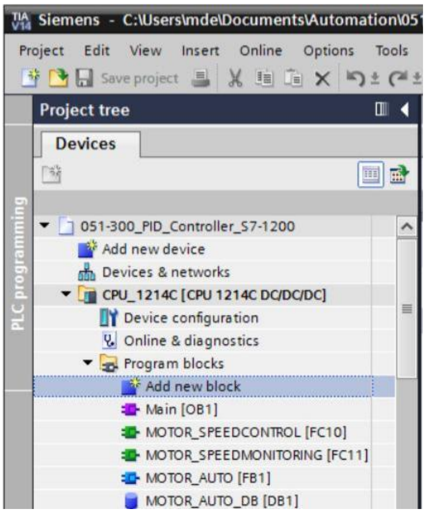
(→ Network 2 → Delete)



14

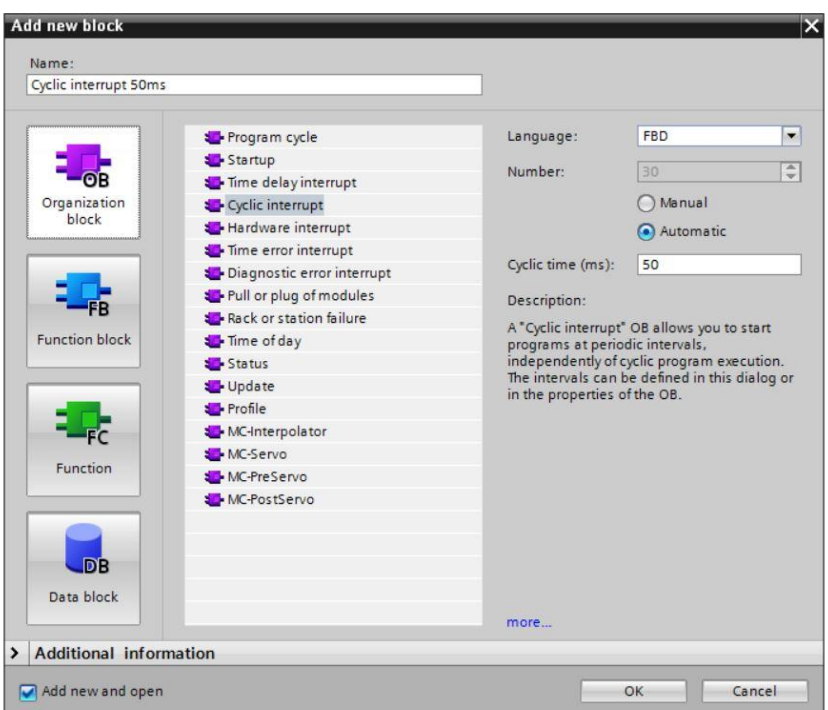
→ We need a cyclic interrupt OB for calling the PID_Compact controller. Therefore, select the 'Add new block' item in the Program blocks folder.

(→ Program blocks → Add new block)



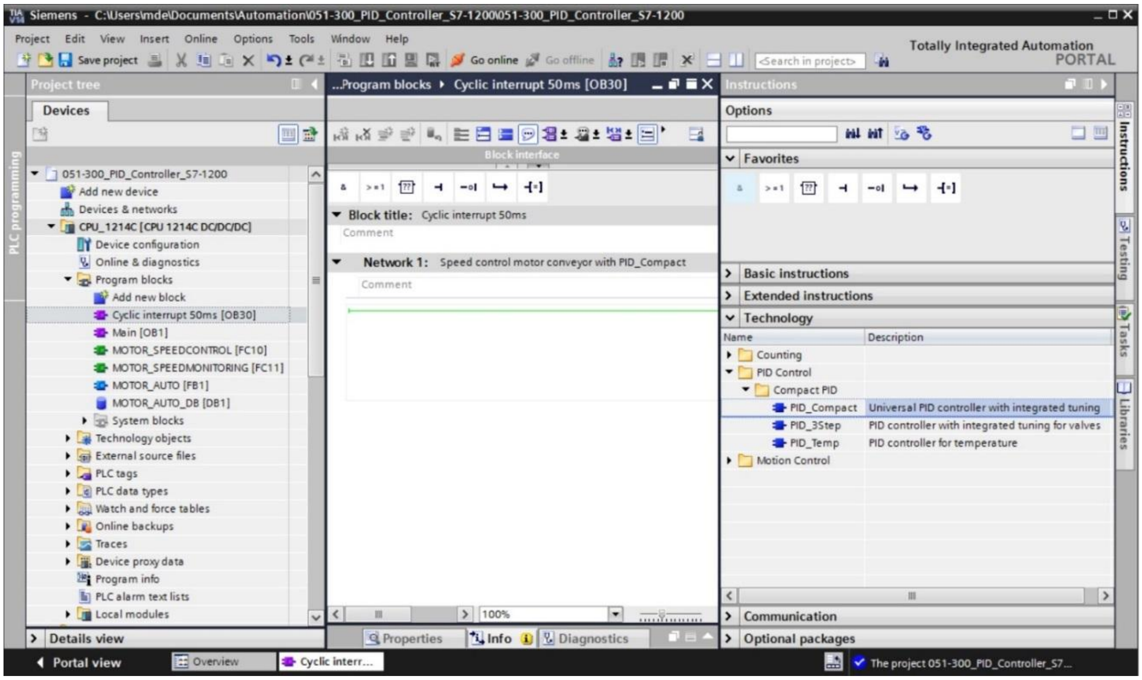
→ Select OB in the next dialog and rename the cyclic interrupt OB to: "Cyclic interrupt 50ms". Set the language to FBD and assign "50 ms" as the cyclic time. Select the "Add new and open" check box. Click "OK".

(→ OB → Name: Cyclic interrupt 50ms → Language: FBD → Cyclic time (ms): 50 → Add new and open → OK)



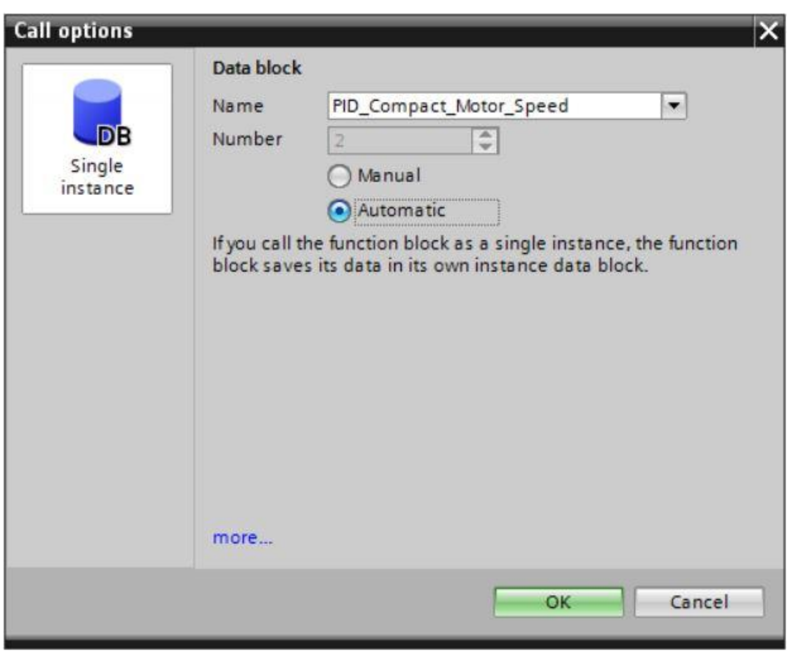
→ The block is then directly opened. Enter meaningful comments and move the 'PID_Compact' technology object to Network 1 using drag & drop.

(→ Technology → PID Control → Compact PID → PID_Compact)



→ Assign a name for the instance data block and apply it with OK.

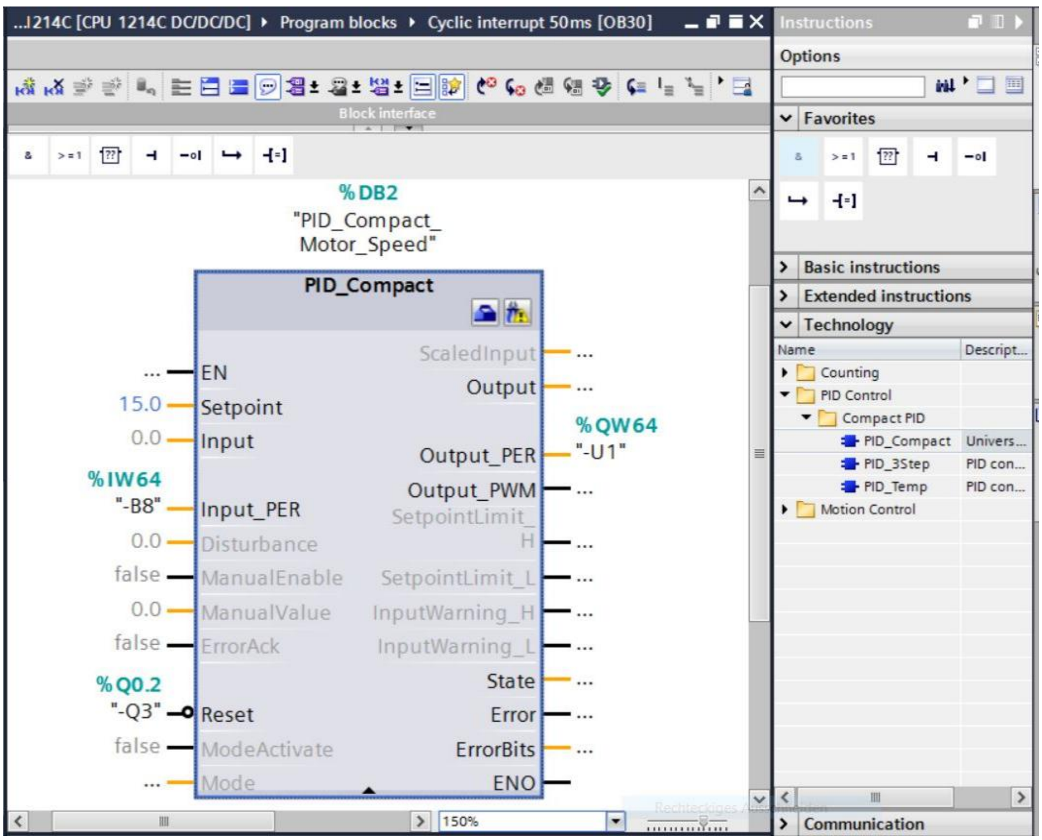
(→ PID_Compact_Motor_Speed → OK)



14

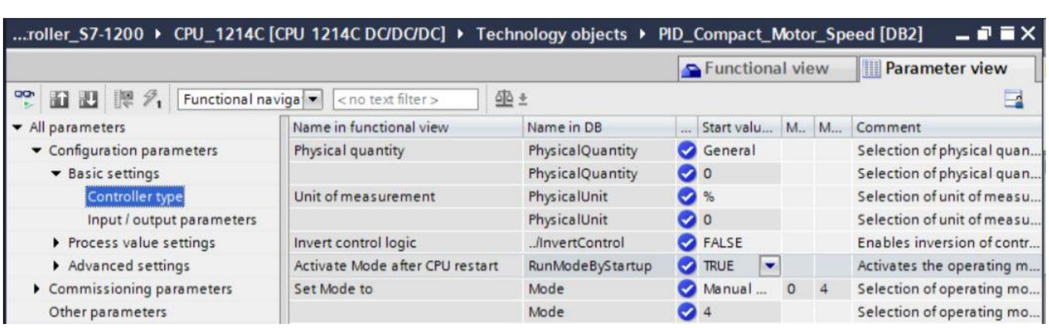
→ Expand the view of the block by clicking the '▲' arrow. Interconnect this block as shown here with setpoint (constant: 15.0), actual value (global tag "-B8"), manipulated variable (global tag "-U1") and Reset input for deactivating the controller (global tag "-Q3"). Negate the 'Reset' input. The configuration mask '' of the controller can then be opened.

(→ ▲ → 15.0 → "-B8" → "-U1" → -Q3 → –o| →)



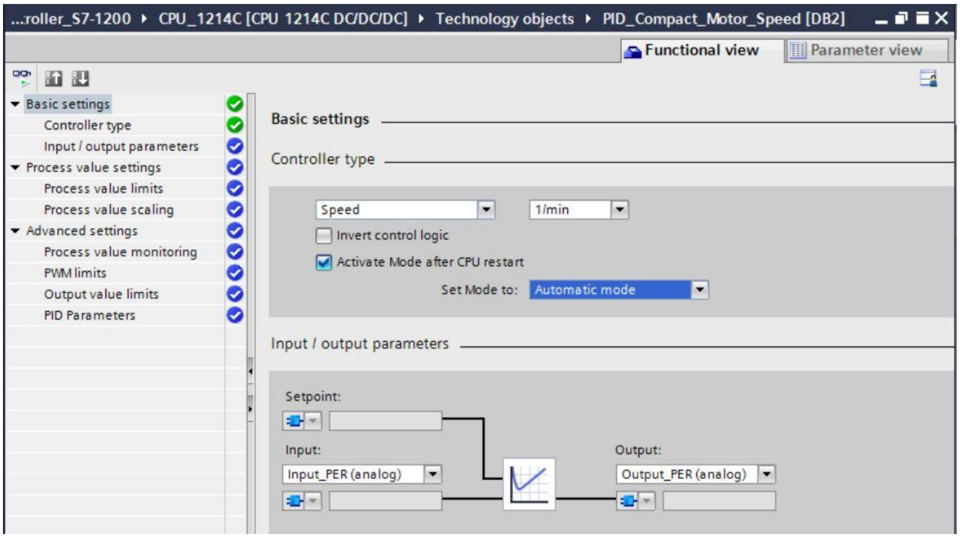
→ There are 2 views for configuration of the controller: Parameter view and Functional view. Here we will use the easier-to-understand 'Functional view'.

(→ Functional view)



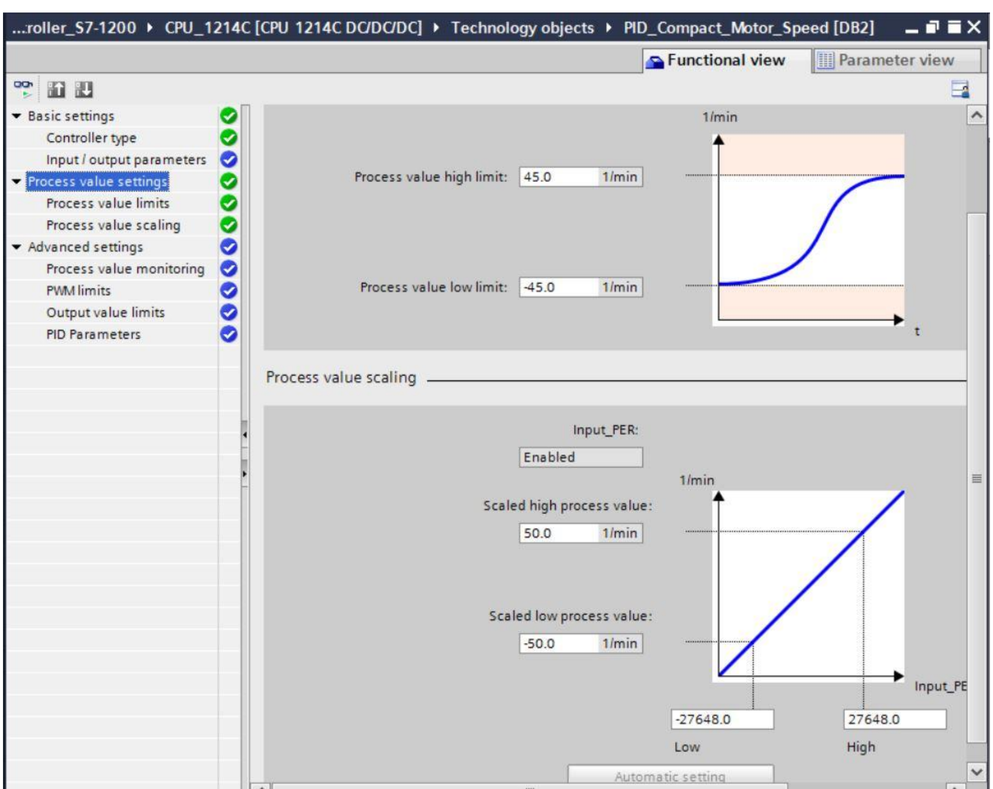
→ In the 'Basic settings', the 'Controller type' and the interconnection of the 'Input / output parameters' are entered. Set the values as shown here.

(→ Basic settings→ Controller type → Input / output parameters)



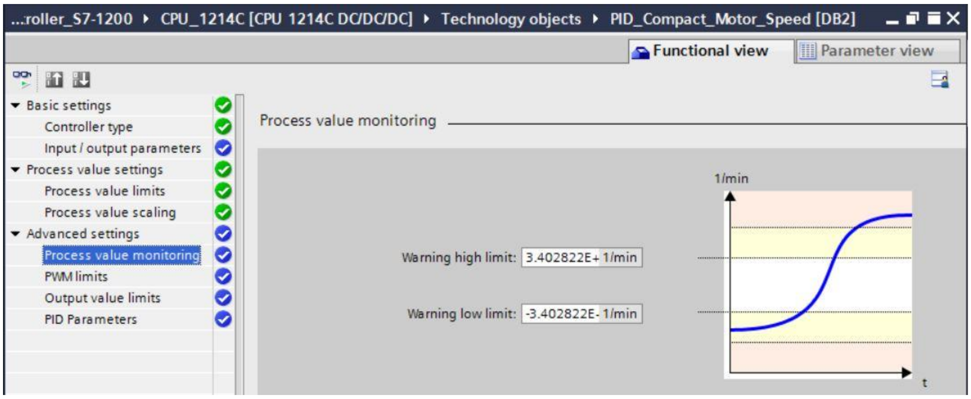
→ In 'Process value settings' we scale to the range +/- 50 rpm and define the 'Process value limits' of +/- 45 rpm.

(→ Process value settings → Process value limits → Process value scaling)



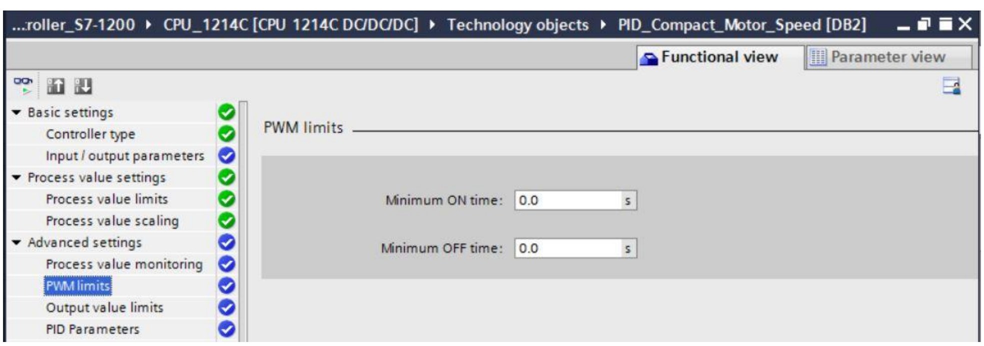
→ In the 'Advanced settings', a process value monitoring would be possible but we don't want to deal with that here.

(→ Advanced settings → Process value monitoring)

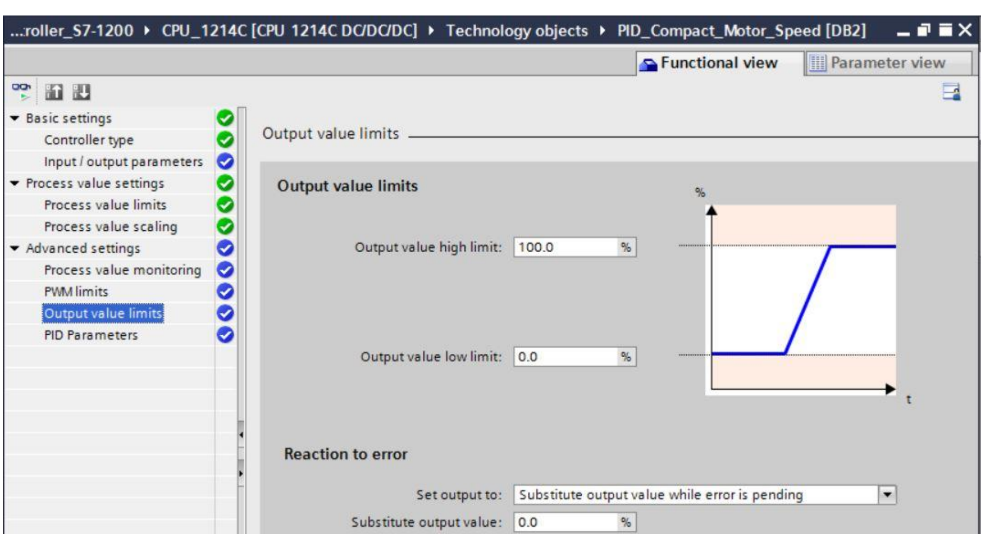


→ In the 'Advanced settings' for 'PWM' (pulse width modulation), we will leave the default values since the output for this is not needed in our project.

(→ Advanced settings → PWM)

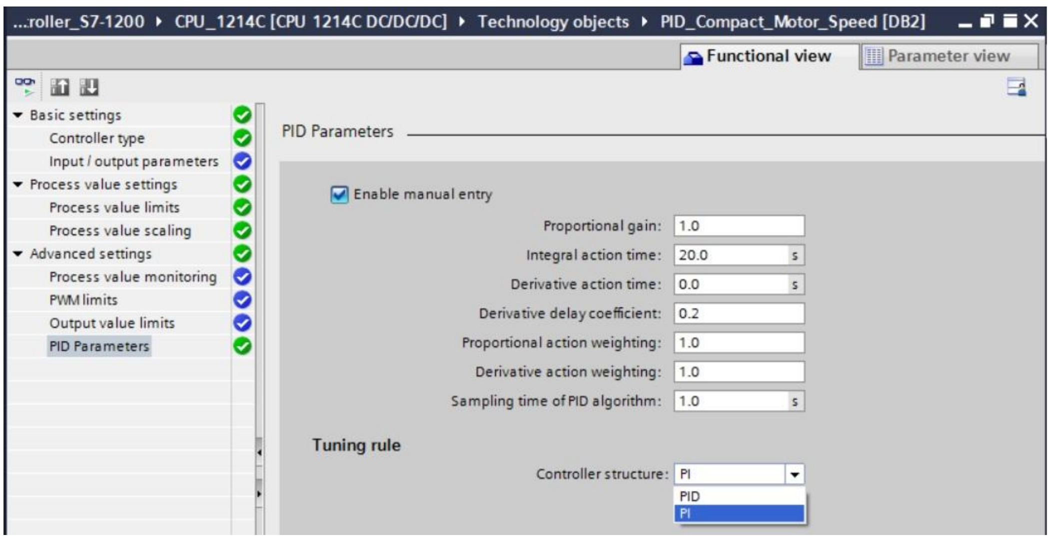


→ In the 'Advanced settings', we define the 'Output value limits' of 0.0% to 100.0%. (→ Advanced settings → Output value limits)



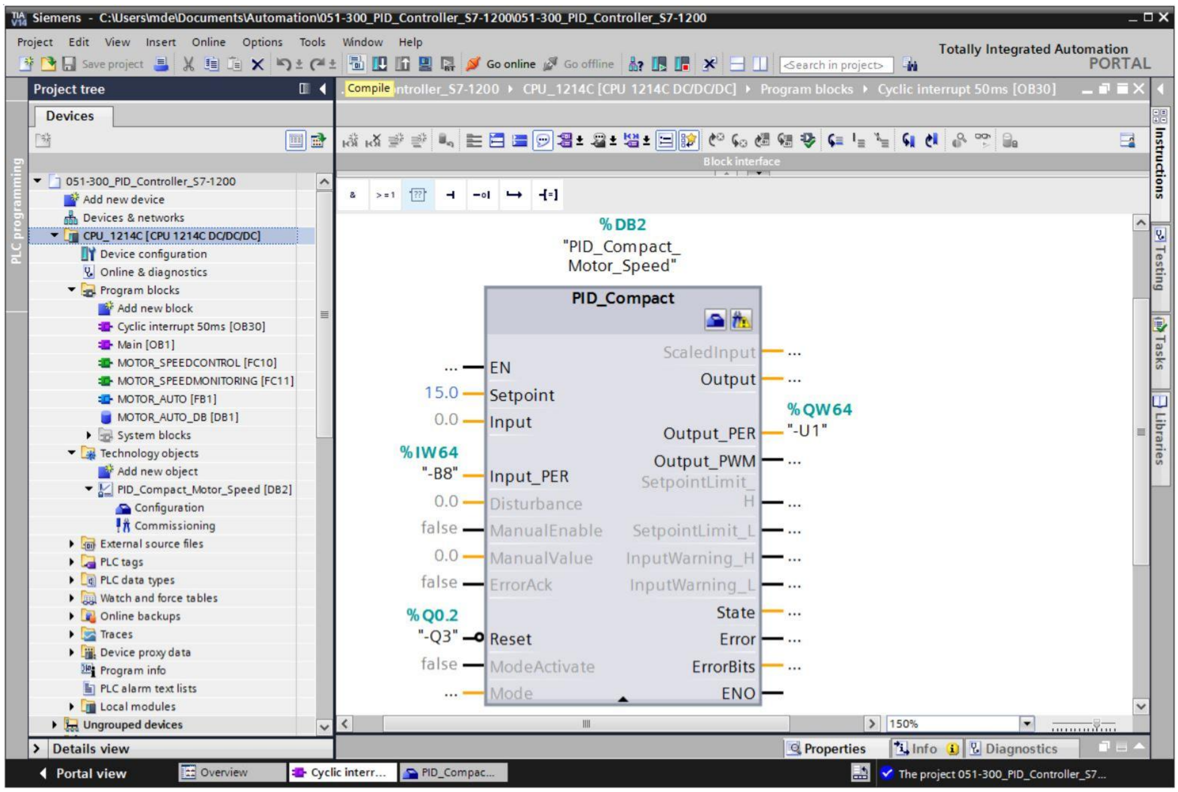
→ In the 'Advanced settings', you will now also find a manual setting of the 'PID parameters'. Once we have changed the controller structure to 'PI', the configuration window is closed by clicking ✕ and we receive a finished product with a functional PID controller. This should, however, still be commissioned and tuned online during operation.

(→ Advanced settings → PID Parameters → Controller structure: PI → ✕)

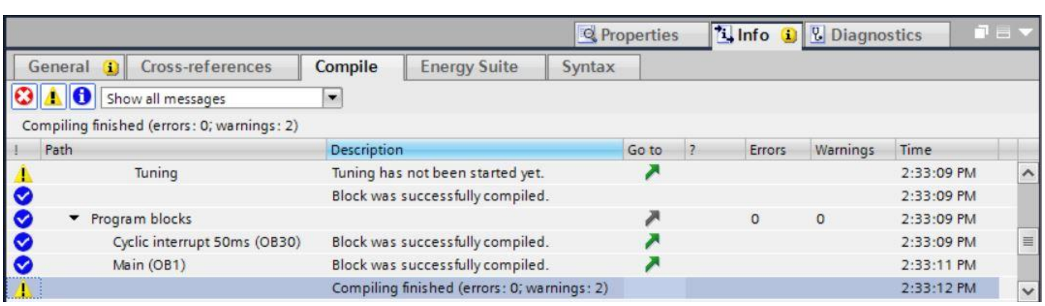


7.3 Save and compile the program

→ To save your project, click the Save project button in the menu. To compile all blocks, click the "Program blocks" folder and select the icon for compiling in the menu. (→ Save project → Program blocks →)



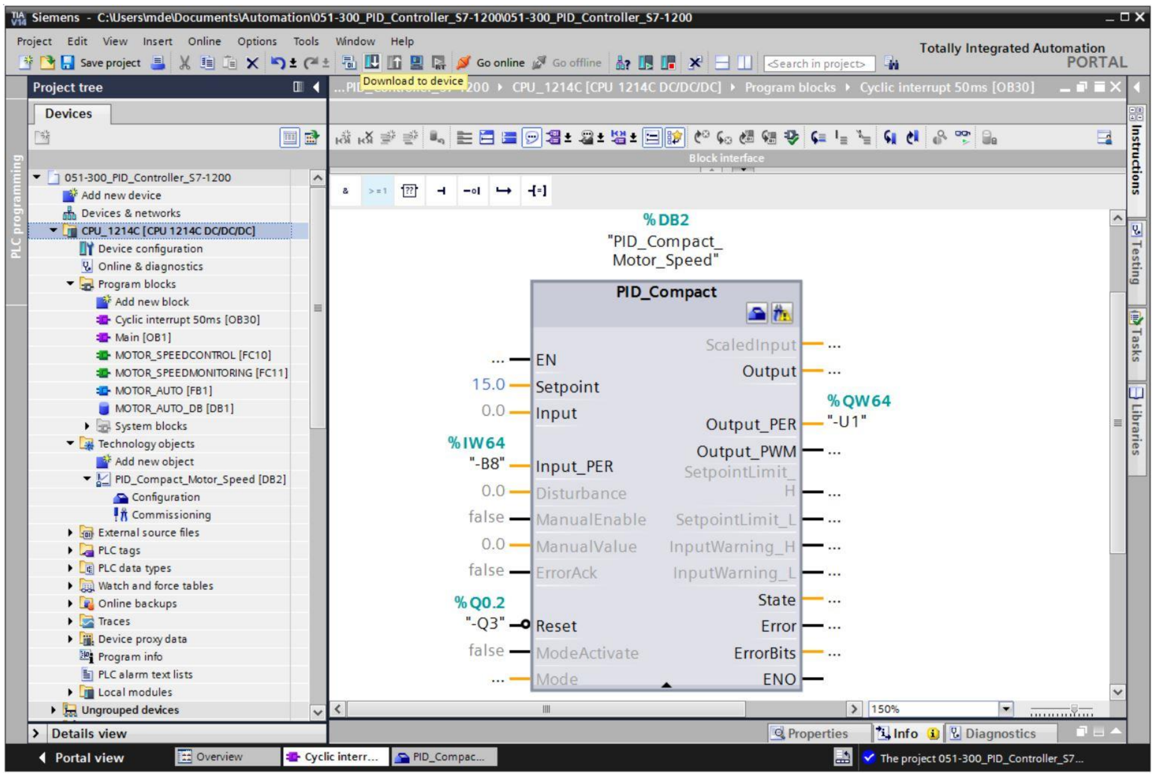
→ The "Info", "Compile" area shows which blocks were successfully compiled.



14

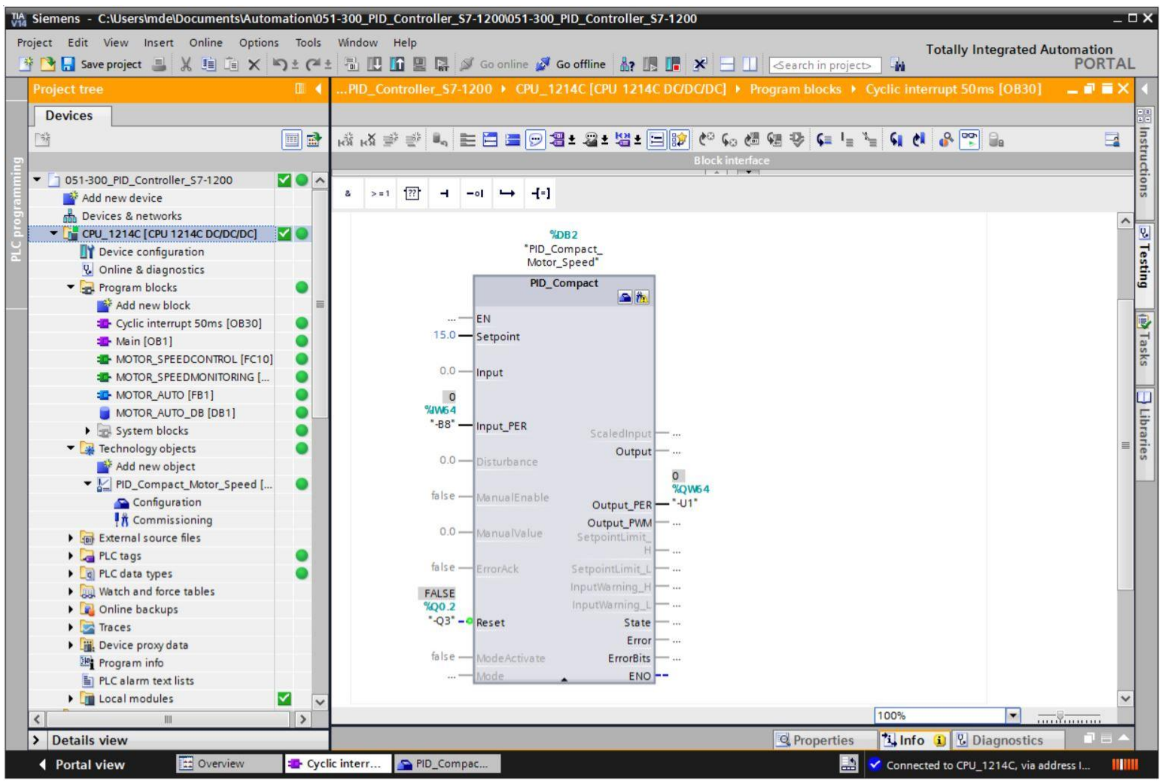
7.4 Download the program

→ After successful compilation, the complete controller with the created program including the hardware configuration can, as described in the previous modules, be downloaded. (→ )

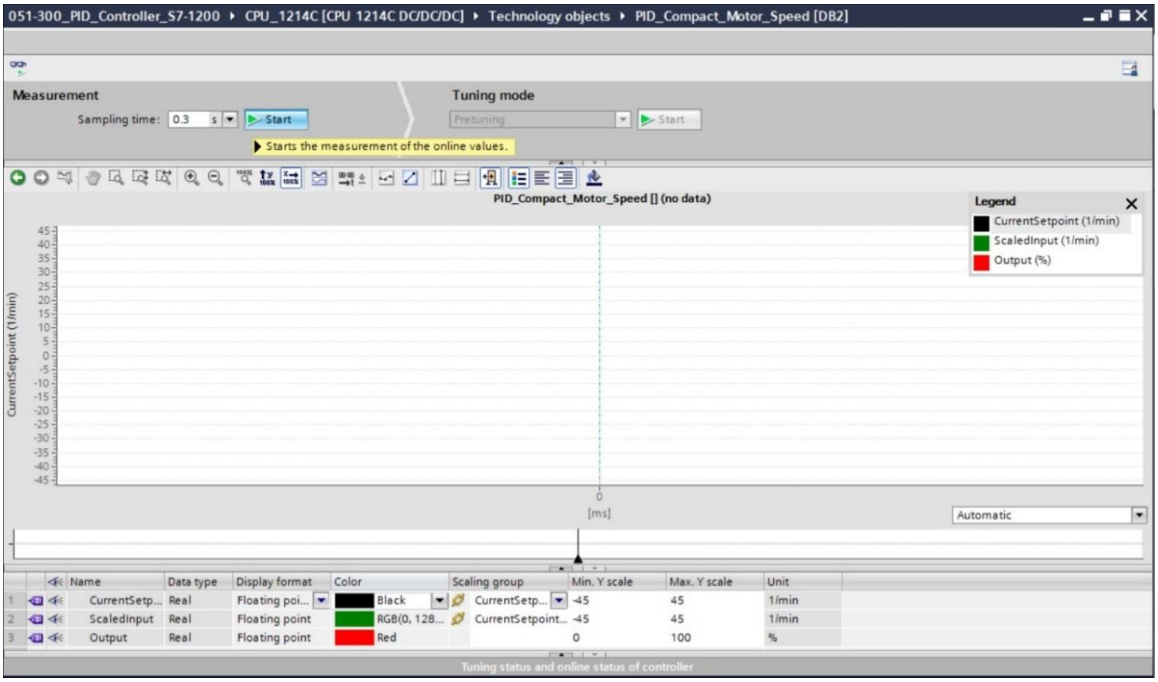


7.5 Monitor PID_Compact

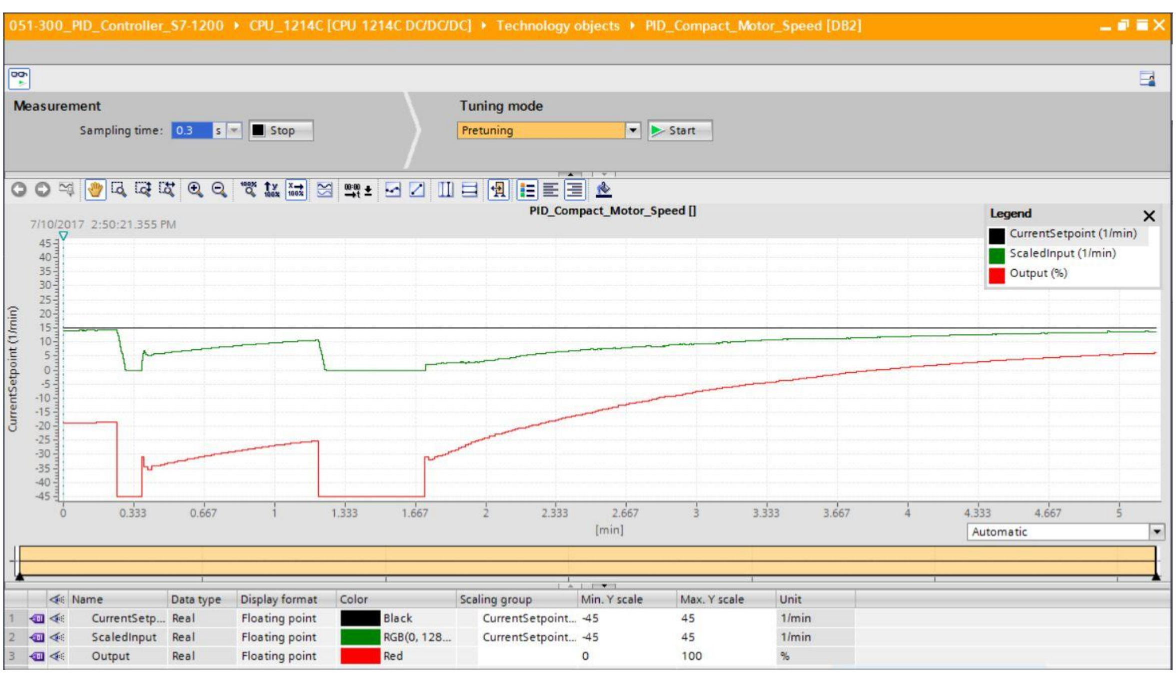
→ Click the Monitoring on/off icon to monitor the state of the blocks and tags when testing the program. At the first start of the CPU, however, the 'PID_Compact' controller is not yet tuned. We still have to start the tuning by clicking the 'Commissioning' icon. (→ Cyclic interrupt 50ms [OB30] → → PID_Compact → Commissioning)



→ If we click ▶ Start under 'Measurement', the values of the setpoint (Setpoint), actual value (ScaledInput) and manipulated variable (Output) can be displayed and monitored in a diagram. (→ ▶ Start)



→ The measurement can be stopped again by clicking '■ Stop'.
(→ ■ Stop)



14

7.6 PID_Compact pretuning

The pretuning determines the process response to a step change of the output value and searches for the turning point. The PID parameters are calculated from the maximum slope and the dead time of the controlled system. The optimal PID parameters are obtained when you perform pretuning and fine tuning.

The more stable the actual value is, the easier and more accurately the PID parameters can be determined. Actual value noise is acceptable as long as the actual value rise is significantly greater than the noise. This is most likely the case in "Inactive" or "Manual mode" operating mode. The PID parameters are backed up before they are recalculated.

The following requirements must be met:

– The "PID_Compact" instruction is called in a cyclic interrupt OB.

– ManualEnable = FALSE

– Reset = FALSE

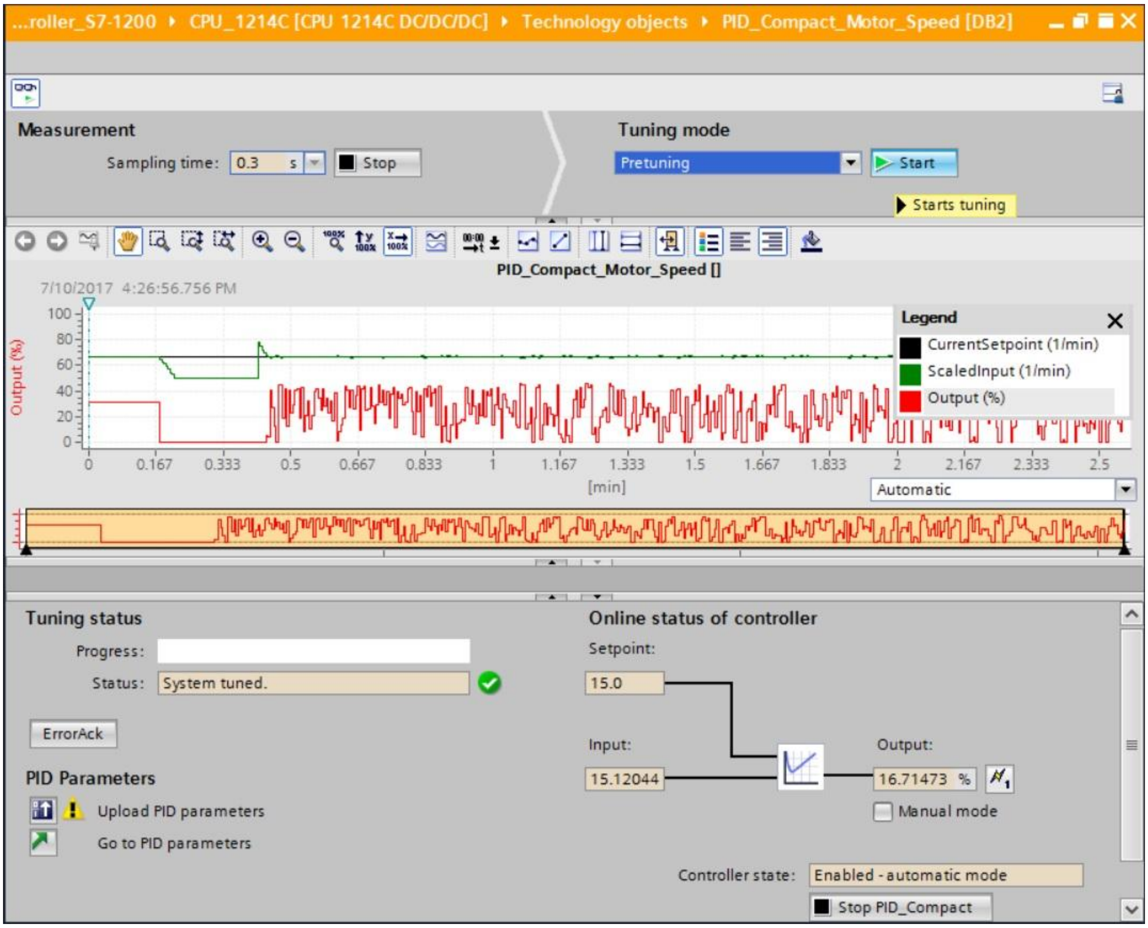
– PID_Compact is in "Manual mode", "Inactive" or "Automatic mode" operating mode.

– The setpoint and actual value are within the configured limits (see "Process value monitoring" configuration).

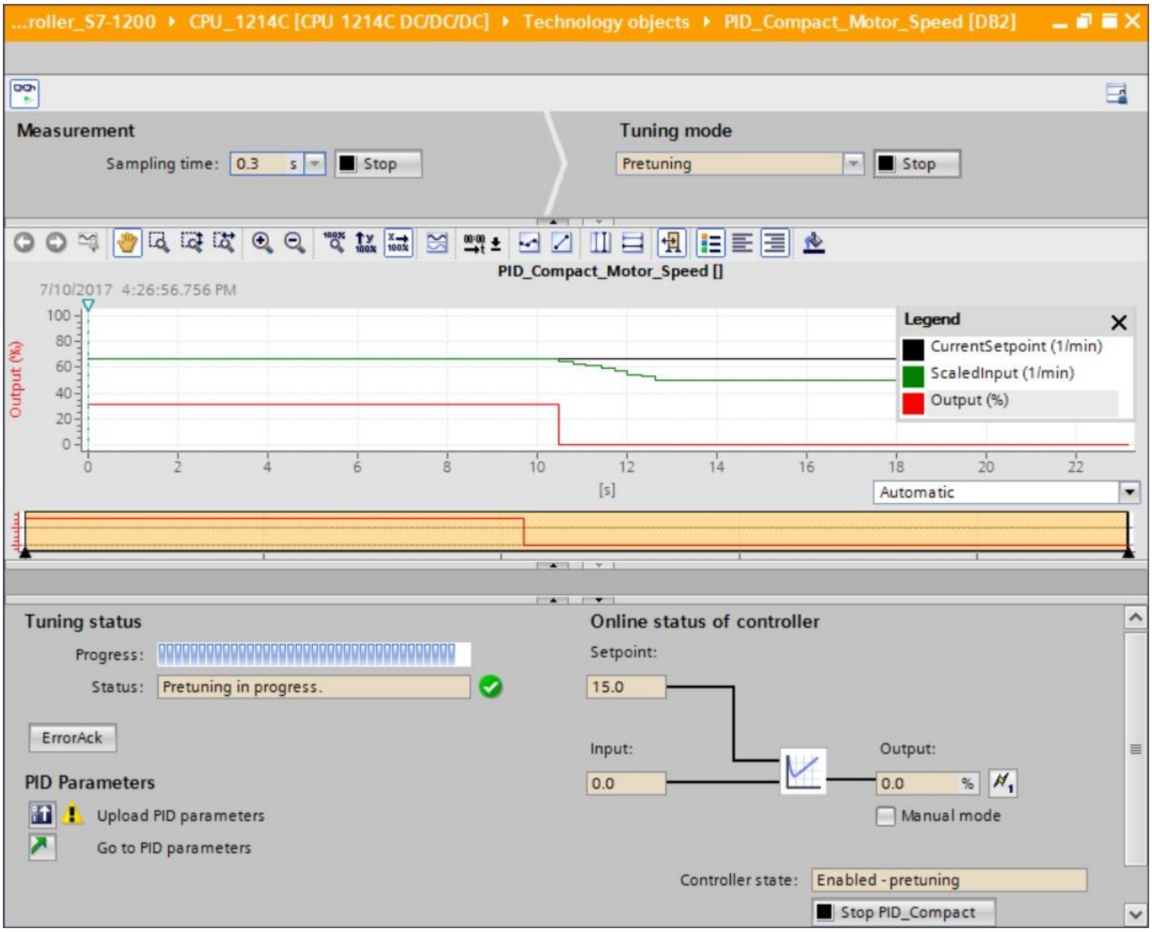
– The difference between setpoint and actual value is greater than 30 % of the difference between the process value high limit and low limit.

– The difference between setpoint and actual value is > 50 % of the setpoint.

→ 'Pretuning' is selected as the 'Tuning mode' and this is then started. (→ Tuning mode → Pretuning → Start)



→ The pretuning starts. The current work steps and any errors that occur are shown in the "Tuning status" field. The progress bar shows the progress of the current work step.



7.7 PID_Compact fine tuning

The fine tuning generates a constant, limited oscillation of the actual value. The PID parameters are optimized for the operating point based on the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. The PID parameters resulting from fine tuning generally produce a better response to setpoint changes and disturbances than the PID parameters from pretuning. The optimal PID parameters are obtained when you perform pretuning and fine tuning.

PID_Compact automatically attempts to generate an oscillation that is greater than the actual value noise. The fine tuning is influenced only slightly by the stability of the actual value. The PID parameters are backed up before they are recalculated.

The following requirements must be met:

– The "PID_Compact" instruction is called in a cyclic interrupt OB.

– ManualEnable = FALSE

– Reset = FALSE

– The setpoint and actual value are within the configured limits.

– The control loop is stable at the operating point. The operating point is reached when the actual value is equal to the setpoint.

– No disturbances are expected.

– PID_Compact is in "Manual mode", "Inactive" or "Automatic mode" operating mode.

The fine tuning runs as follows when started in automatic mode:

When you want to improve the existing PID parameters by tuning them, start the fine tuning from automatic mode.

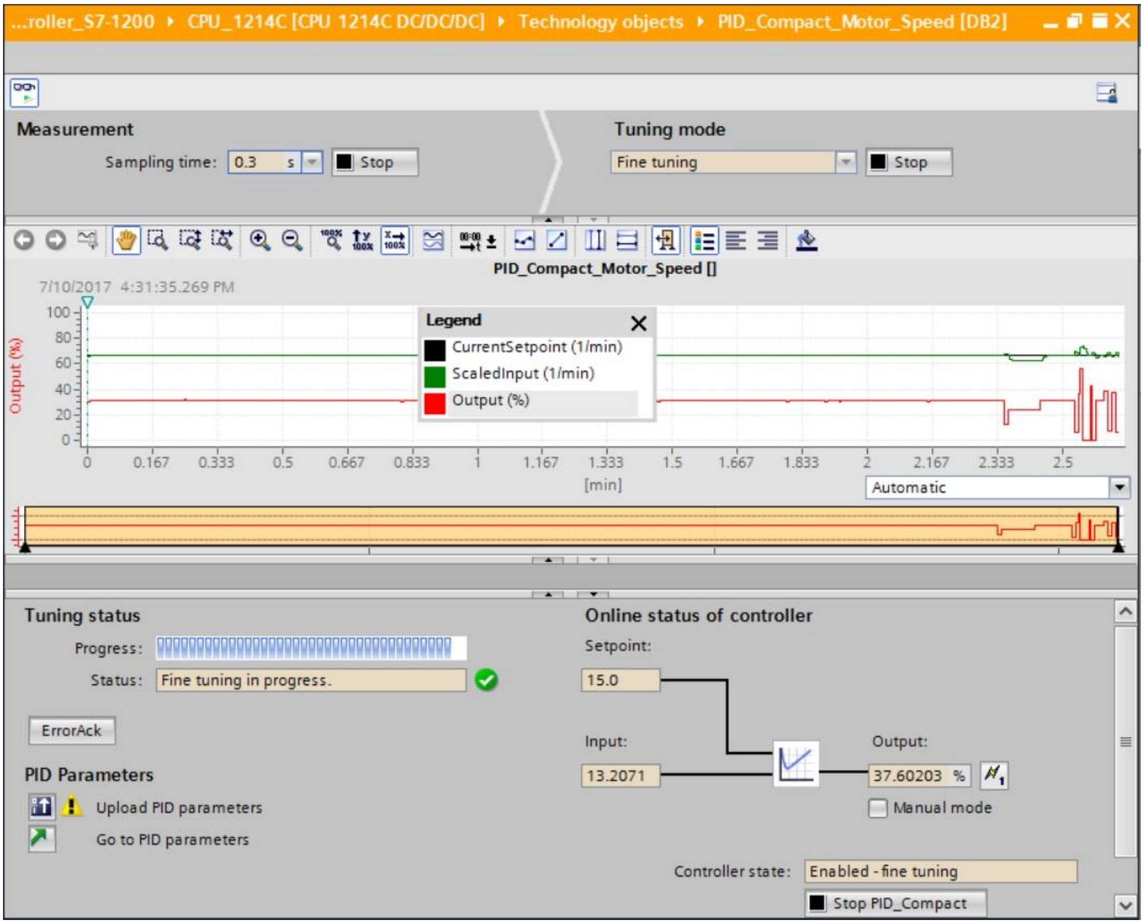
PID_Compact uses the existing PID parameters for controlling until the control loop is stable and the requirements for fine tuning are met. Only then does the fine tuning start.

The fine tuning runs as follows when started in inactive or manual mode:

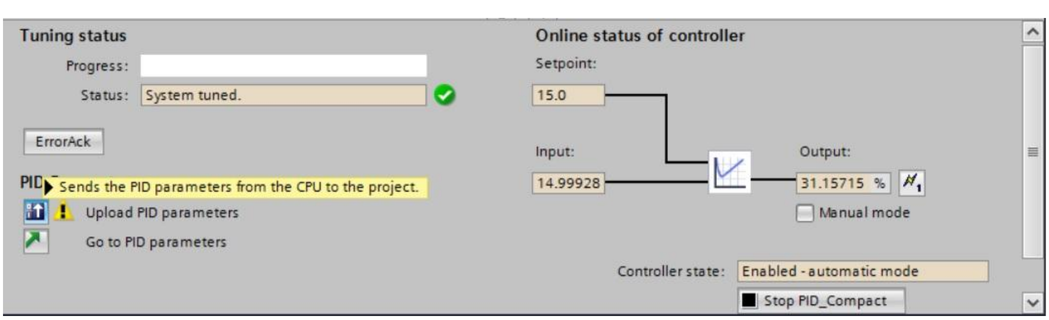
When the requirements for pretuning are met, pretuning is started. PID_Compact uses the determined PID parameters for controlling until the control loop is stable and the requirements for fine tuning are met. Only then does the fine tuning start. If pretuning is not possible, PID_Compact responds as configured in Response to error.

If the actual value is already too close to the setpoint for pretuning, an attempt is made to reach the setpoint with minimum or maximum output value. This can cause increased overshoot.

→ 'Fine tuning' is selected as the 'Tuning mode' and this is then started. (→ Tuning mode → Fine tuning → Start)

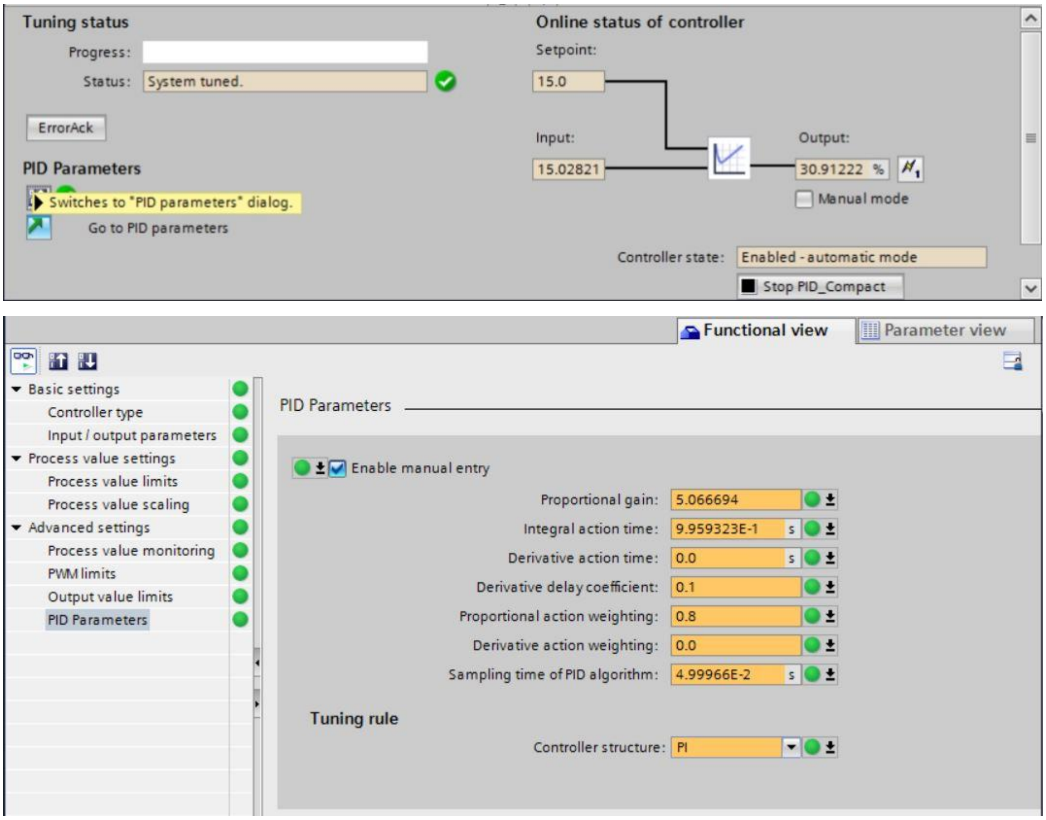


→ The fine tuning starts. The current work steps and any errors that occur are shown in the "Tuning status" field. If the self-tuning was completed without error message, the PID parameters have been tuned. The PID controller switches to automatic mode and uses the tuned parameters. The tuned PID parameters are retained at a Power ON and restart of the CPU. You can download the PID parameters from the CPU to your project with the ' ' button. (→)



14

→ The PID parameters in the configuration can be displayed by clicking ''. (→)



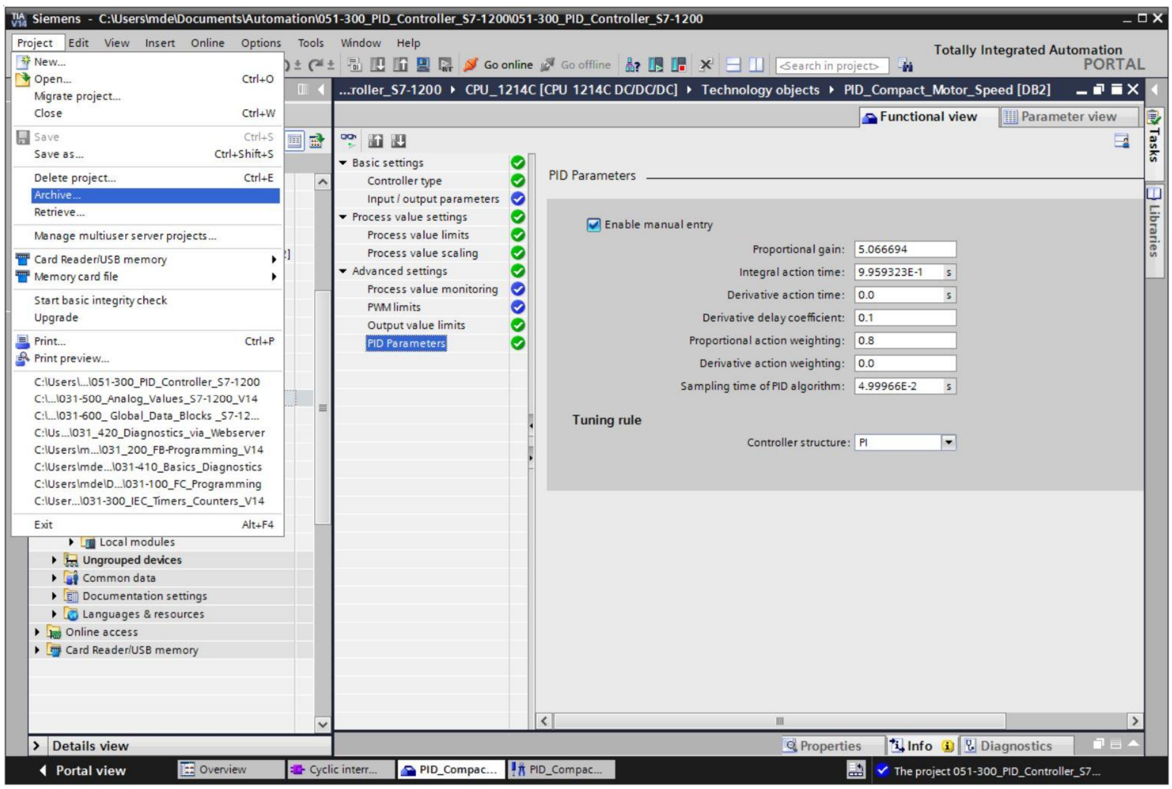
→ As the final step, the online connection should be disconnected and the complete project should be saved.

(→ Go offline → Save project)

7.8 Archive the project

→ Now we want to archive the complete project. Select the → 'Archive ...' command in the → 'Project' menu. Select a folder where you want to archive your project and save it with the file type "TIA Portal project archive".

(→ Project → Archive → TIA Portal project archive → 051-300_PID_Control_S7-1200.... → Save)



8 Checklist

No.	Description	Completed
1	Cyclic interrupt OB Cyclic interrupt 50ms [OB30] successfully created.	
2	PID_Compact controller in cyclic interrupt OB Cyclic interrupt 50ms [OB30] called and connected.	
3	Configuration of the PID_Compact controller performed.	
4	Compiling successful and without error message	
5	Download successful and without error message	
6	Pretuning successful and without error message	
7	Fine tuning successful and without error message	
8	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated AUTOMATIC mode (-S0 = 1) Pushbutton automatic stop not actuated (-S2 = 1) Briefly press the automatic start pushbutton (-S1 = 1) Sensor part at slide activated (-B4 = 1) then Conveyor motor M1 variable speed (-Q3 = 1) switches on and stays on. The speed corresponds to the speed setpoint in the range +/- 50 rpm	
9	Sensor part at end of conveyor activated (-B7 = 1) → -Q3 = 0 (after 2 seconds)	
10	Briefly press the automatic stop pushbutton (-S2 = 0) → -Q3 = 0	
11	Activate EMERGENCY OFF (-A1 = 0) → -Q3 = 0	
12	Manual mode (-S0 = 0) → -Q3 = 0	
13	Switch off station (-K0 = 0) → -Q3 = 0	
14	Cylinder not retracted (-B1 = 0) → -Q3 = 0	
15	Speed > Motor_speed_monitoring_error_max → -Q3 = 0	
16	Speed < Motor_speed_monitoring_error_min → -Q3 = 0	
17	Project successfully archived	

9 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software / firmware, under the following link:

www.siemens.com/sce/s7-1200

Preview „Additional information“

Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- TIA Portal Videos
- TIA Portal Tutorial Center
- Getting Started
- Programming Guideline
- Easy Entry in SIMATIC S7-1200
- Download Trial Software/Firmware
- Technical Documentation SIMATIC Controller
- Industry Online Support App
- TIA Portal, SIMATIC S7-1200/1500 Overview
- TIA Portal Website
- SIMATIC S7-1200 Website
- SIMATIC S7-1500 Website

Notes

14

Notes


SIEMENS
Ingenuity for life
SIEMENS
Global Industry Partner of WorldSkills International
worldskills
Simplify your education in automation
Unique support for educators and students in educational institutions
Siemens Automation Cooperates with Education (SCE)
Curriculums
Courses
Trainer Packages
Partnerships
siemens.com/sce

Further Information

Siemens Automation Cooperates with Education
siemens.com/sce

SCE Learn-/Training Documents
siemens.com/sce/documents

SCE Trainer Packages
siemens.com/sce/tp

SCE Contact Partners
siemens.com/sce/contact

Digital Enterprise
siemens.com/digital-enterprise

Industry 4.0
siemens.com/
future-of-manufacturing

Totally Integrated Automation (TIA)
siemens.com/tia

TIA Portal
siemens.com/tia-portal

SIMATIC Controller
siemens.com/controller

SIMATIC Technical Documentation
siemens.com/simatic-docu

Industry Online Support
support.industry.siemens.com

Product catalogue and online ordering system Industry Mall
mall.industry.siemens.com

Siemens AG
Digital Factory Division
P.O. Box 48 48
90026 Nuremberg
Germany

Subject to change without prior notice
Article No. 6ZB5310-0QL01-0BA0
SO 1017 PDF 668 En
Produced in Germany

Industrial Security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit
https://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under
https://www.siemens.com/industrialsecurity.