

SIEMENS



Cooperates
with Education

Automation

SIEMENS

SCE Training Curriculum for Integrated Automation Solutions Totally Integrated Automation (TIA)

Automation System SIMATIC S7-1200

Answers for industry.

SIEMENS



Engineered
with
TIA Portal

SIMATIC S7-1200

It's the interplay that makes the difference

siemens.com/s7-1200



**Intuitive, efficient, proven:
TIA Portal redefines engineering.**

SIMATIC
S7-1200 –
take a look!



Answers for industry.

Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Modules 010 Edition 09/2012



© Siemens AG 2013

TIA Portal Module 010-010

Startup Programming of the
SIMATIC S7-1200

1

TIA Portal Module 010-020

Block Types for the SIMATIC S7-1200

2

TIA Portal Module 010-030

IEC Timers and IEC Counters at the
SIMATIC S7-1200

3

TIA Portal Module 010-040

Diagnosis and Error Search at the
SIMATIC S7-1200

4

TIA Portal Module 010-050

Analog Value Processing with
SIMATIC S7-1200

5

TIA Portal Module 010-060

Control Engineering with
SIMATIC S7-1200

6

TIA Portal Module 010-070

Communication between two
SIMATIC S7-1200

7

TIA Portal Module 010-080

HMI Panel KTP600 with SIMATIC
S7-1200

8

TIA Portal Module 010-090

Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

9

Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Module 010-010



TIA Portal Module 010-010

Startup Programming of the
SIMATIC S7-1200

TIA Portal Module 010-020

Block Types for the SIMATIC S7-1200

TIA Portal Module 010-030

IEC Timers and IEC Counters at the
SIMATIC S7-1200

TIA Portal Module 010-040

Diagnosis and Error Search at the
SIMATIC S7-1200

TIA Portal Module 010-050

Analog Value Processing with
SIMATIC S7-1200

TIA Portal Module 010-060

Control Engineering with
SIMATIC S7-1200

TIA Portal Module 010-070

Communication between two
SIMATIC S7-1200

TIA Portal Module 010-080

HMI Panel KTP600 with SIMATIC
S7-1200

TIA Portal Module 010-090

Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

Matching SCE training packages for these documents

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
Order number: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
Order number 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
Order number 6ES7822-0AA01-4YE0

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

Continued Training

For regional Siemens SCE continued training, please contact your regional SCE contact person
siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding Usage

The training curriculum 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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by 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.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the documents being used commercially.

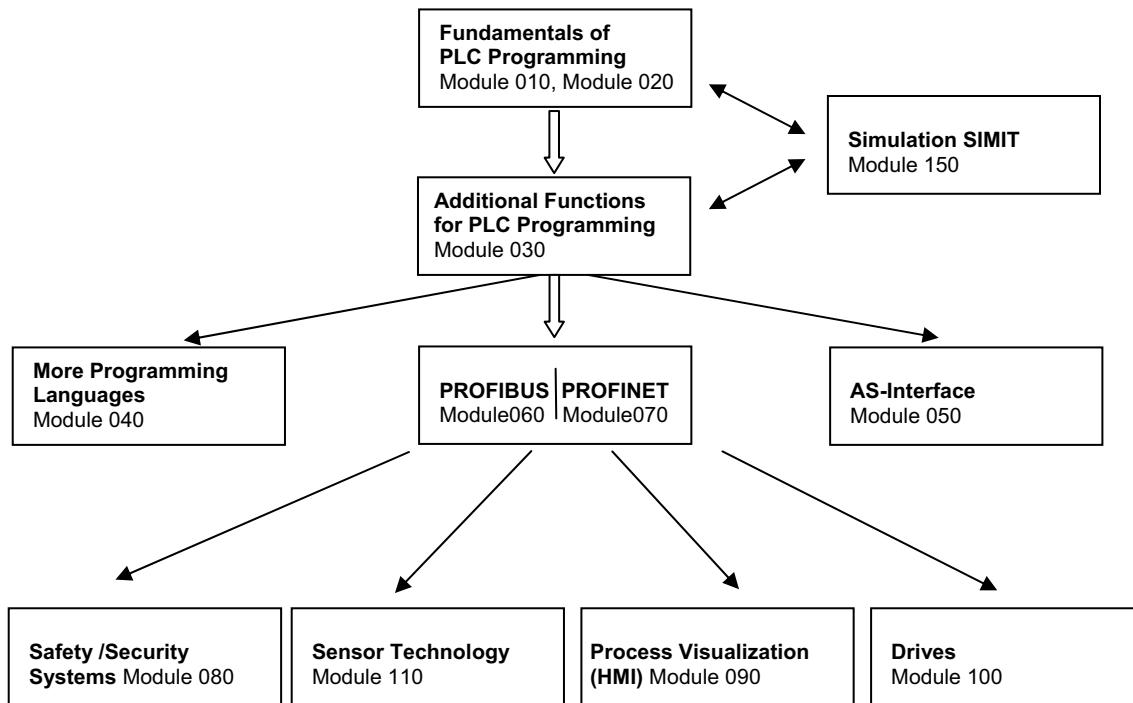
We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this document.

Contents

1.	Preface	4
2.	Notes on Programming the SIMATIC S7-1200	6
2.1	Automation System SIMATIC S7-1200	6
2.2	Programming Software STEP Professional V11 (TIA Portal V11)	6
3.	Installing the Software STEP 7 Professional (TIA Portal V11)	7
3.1	Installing current Updates for STEP 7 Professional V11 (TIA-Portal V11)	7
3.2	Update the hardware catalog in STEP 7 Professional V11 (TIA-Portal V11 with Hardware Support Packages (HSP))	8
4.	Connecting to the CPU by means of TCP/IP, and Resetting to Factory Setting	12
5.	What is a PLC and what are PLCs Used For?	19
5.1	What does the term PLC mean?	19
5.2	How does the PLC Control the Process?	19
5.3	How does the PLC Get the Information about the Process States?	20
5.4	What is the Difference between Break Contact Elements and Make Contact Elements?	20
5.5	How does the SIMATIC S7-1200 Address Individual Input/Output Signals?	21
5.6	How is the Program Processed in the PLC?	22
5.7	What do Logic Operations Look Like in the PLC Program?	23
5.7.1	AND Operation	23
5.7.2	OR Operation	25
5.7.3	Negation	26
5.8	How is the PLC Program generated? How does it get to the PLC's memory?	27
6.	Configuring and Operating the SIMATIC S7-1200	28
7.	Sample Task: Controlling a Press	32
8.	Programming the Press for the SIMATIC S7-1200	33
8.1.	Portal View	33
8.2.	Project View	34

1. Preface

Regarding its content, module SCE_DE_010-010 is part of the training unit '**Basics of PLC Programming**' and represents a **fast entry point** for programming the SIMATIC S7-1200 with the TIA Portal.



Training Objective:

In this module 010-010, the reader learns how to program the programmable logic controller (PLC) SIMATIC S7-1200, using the programming tool TIA Portal. This module provides the fundamentals and demonstrates with the steps listed below how this is done, using a detailed example.

- Installing the software and setting the programming interface
- Explanation: What is a PLC and how does it work
- Structure and operation of the SIMATIC S7-1200 PLC
- Generating, loading and testing a sample program

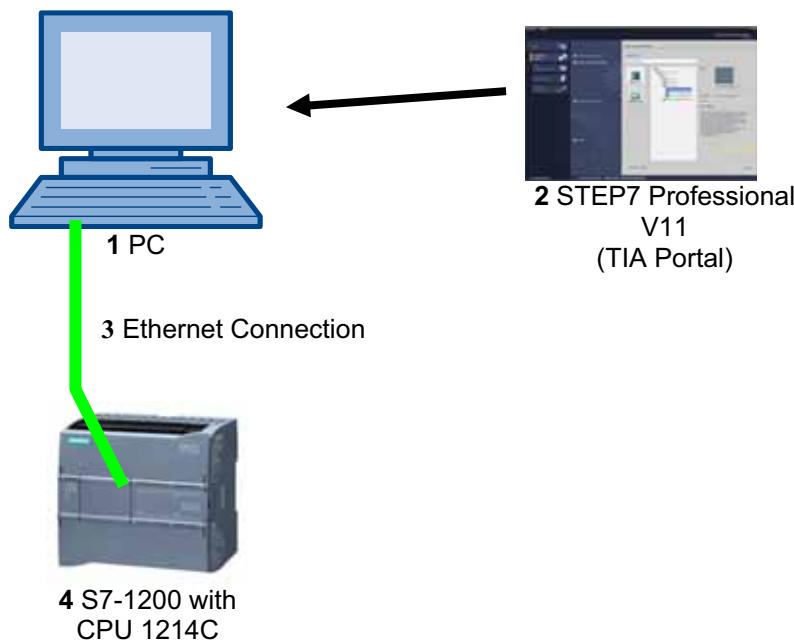
Preconditions:

To successfully work through this module M01, the following knowledge is assumed:

- How to operate Windows

Hardware and software needed

- 1** PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB; operating system Windows XP Professional SP3/Windows 7 Professional/Windows 7 Enterprise/Windows 7 Ultimate/Windows 2003 Server R2/Windows Server 2008 Premium SP1, Business SP1, Ultimate SP1
- 2** Software STEP7 Professional V11 SP1 (Totally Integrated Automation (TIA) Portal V11)
- 3** Ethernet connection between PC and CPU 315F-2 PN/DP
- 4** PLC SIMATIC S7-1200; for example, CPU 1214C.
The inputs have to be brought out to a panel.



2. Notes on Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The automation system SIMATIC S7-1200 is a modular mini-controller system for the lower and medium performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU and input/output modules for digital and analog signals. .

If needed, communication processors and function modules are added for special tasks such as step motor control.

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process, whereby the IO modules are polled in the S7 program by means of the input addresses (%I) and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7.

2.2 Programming Software STEP Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 (TIA Portal V11) is the programming tool for the following automation systems

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining the communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating the visual displays for the SIMATIC basic panels with the integrated WinCC Basic
- With additional WinCC packages, visual display solutions for PCs and other panels can be prepared

All functions are supported with detailed online help.

3. Installing the Software STEP 7 Professional (TIA Portal V11)

STEP 7 Professional is supplied on a DVD.

To install STEP 7 Professional, do the following:

1. Insert the DVD of STEP 7 Basic Professional in the DVD drive.
2. The setup program is started automatically. If not, start it by double clicking on the file '**START.exe**'.
3. The setup program guides you through the entire installation of STEP 7 Professional.
4. To utilize STEP 7 Professional, a license key is needed on your computer.

This license key can be transferred in the course of installation from a USB stick that is included to your computer. Subsequently it is possible to move this license key to another data carriers using the software '**Automation License Manager**'. This license key may also be located on another computer and polled by means of a network.

3.1 Installing current Updates for STEP 7 Professional V11 (TIA-Portal V11)

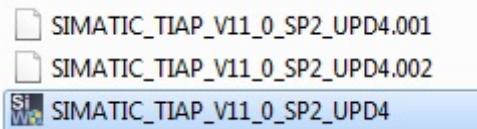
STEP 7 Professional V11 (TIAPortal V11) is updated frequently to optimally integrate new products; the respective updates are made available on the Internet.

Here the link to the corresponding webpage:

<http://support.automation.siemens.com/WW/view/en/59604410>

From there, you can download upgrades to save them on your PC or laptop.
Installation then takes place as follows:

1. First you start the "**Application**" with a double-click from the saved files. Here update 4 for version 11 Service Pack 2, **SIMATIC TIAP_V11_0_SP2_UPD4**. (→ SIMATIC TIAP_V11_0_SP2_UPD4)



Note:

You should check the software requirements before you install the update. In this case, the requirement is an installed software package STEP 7 Professional V11 (TIA Portal V11) SP2.

2. Next you select the setup language. (→ Setup language: English → Next)
3. Then you select the “Temp” folder to unzip the packet. (→ Next)
4. In a fourth step, you can see detailed product information. (→ Read product information → Next)
5. In step 5, you accept the license agreement and acknowledge the security information. (→ → → Next)
6. In step 6, you accept the security information. (→ → Next)
7. You start the installation in step 7. (→ Install)
8. You have to restart your computer to finish installation of the update. (→ Restart →)

3.2 Update the hardware catalog in STEP 7 Professional V11 (TIA-Portal V11 with Hardware Support Packages (HSP))

Hardware Support Packages (HSP) are offered on the Internet in the STEP 7 Professional V11 (TIA Portal V11) hardware catalog to integrate new modules.

Here the link to the corresponding webpage:

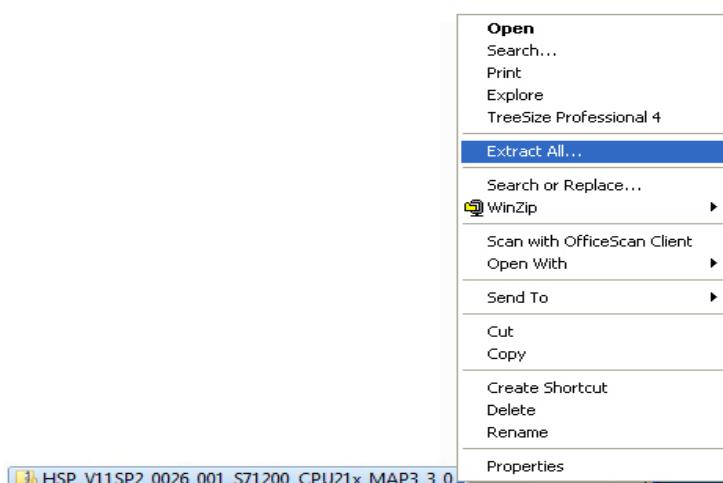
<http://support.automation.siemens.com/WW/view/en/54164095>

A description for installation of the Hardware Support Packages (HSP) is available under:

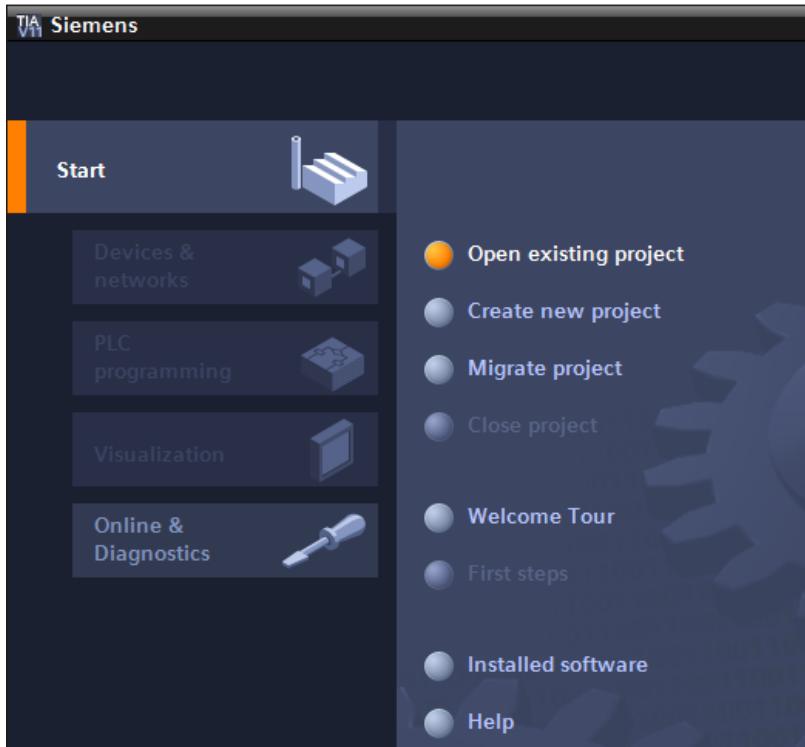
<http://support.automation.siemens.com/WW/view/en/54163659>

Here is a description of how to install new modules in STEP 7 Professional V11 (TIA Portal V11):

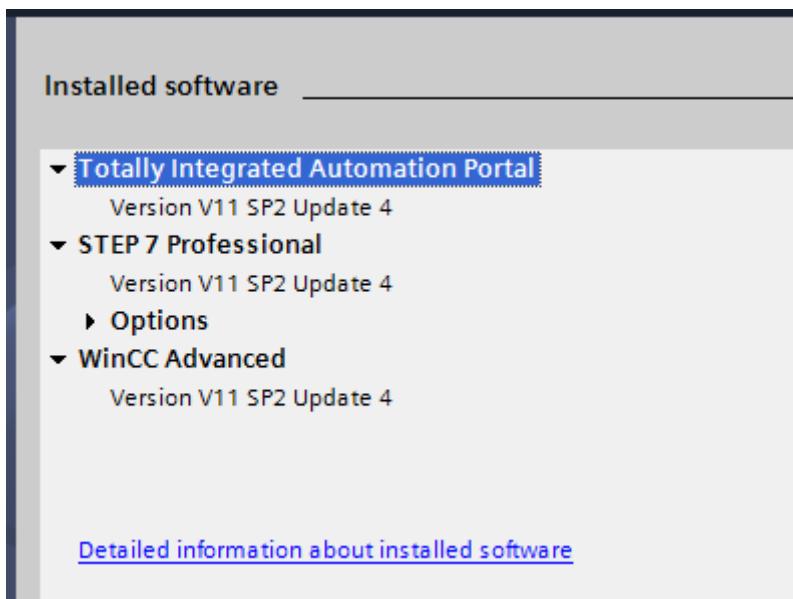
1. First you must “extract” the downloaded file in Windows Explorer. (→ HSP_V11SP2_000....)



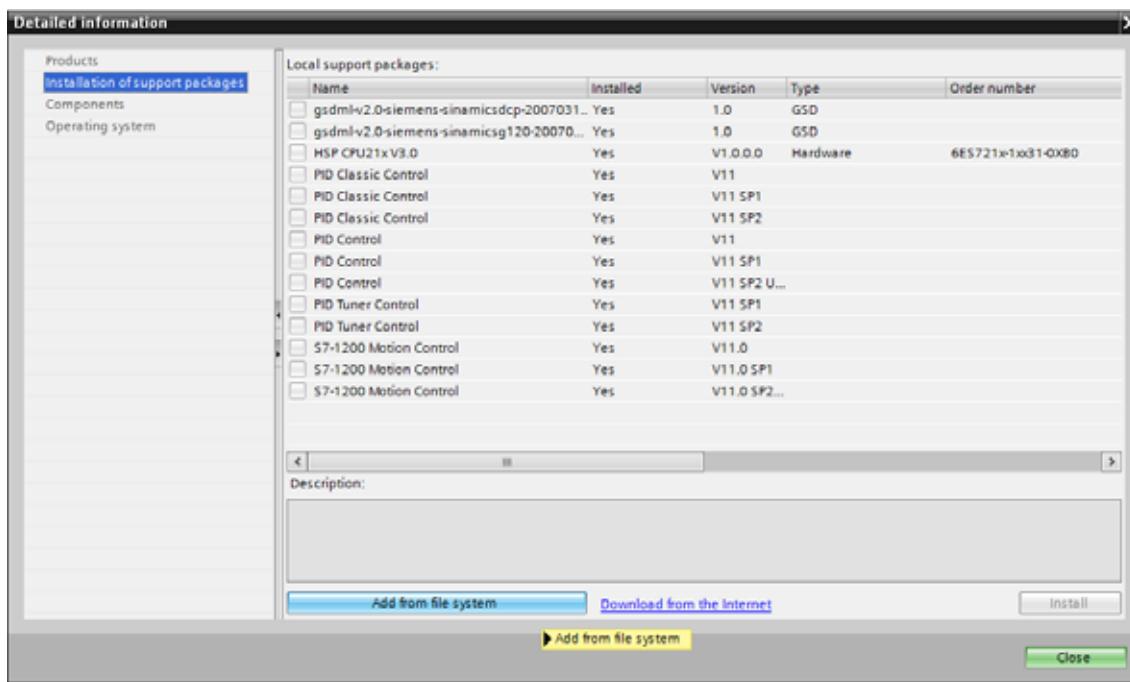
2. Then you select the item "Installed software" in the TIA Portal. (→ Installed software)



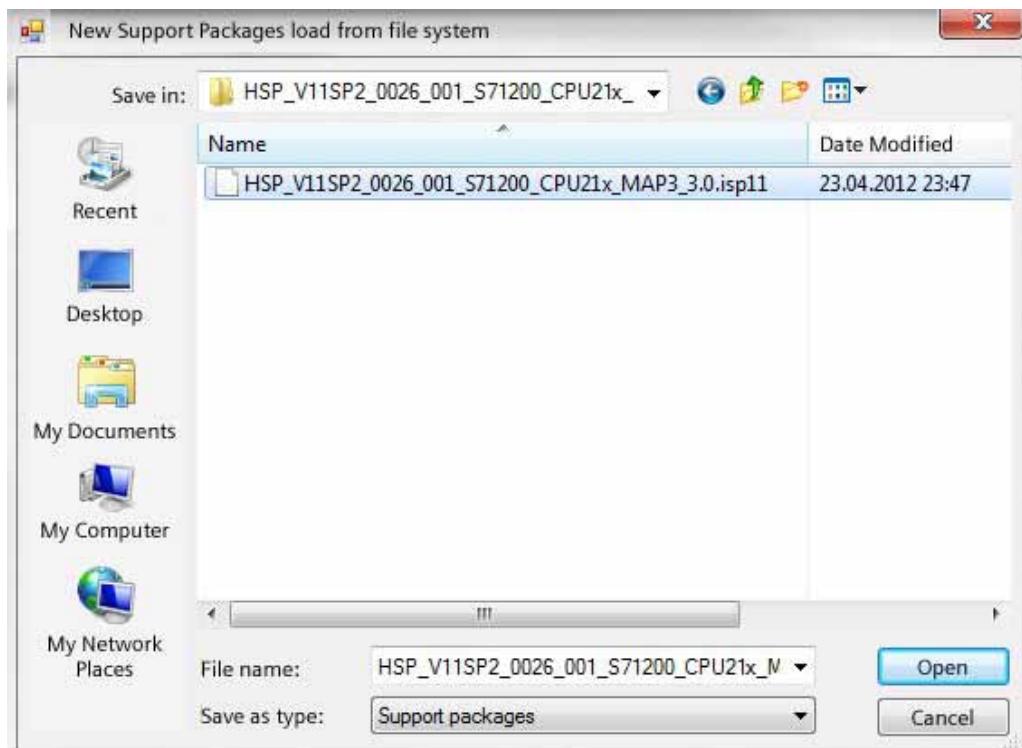
3. In the next step, you select "Detailed information about installed software". (→ Detailed information about installed software)



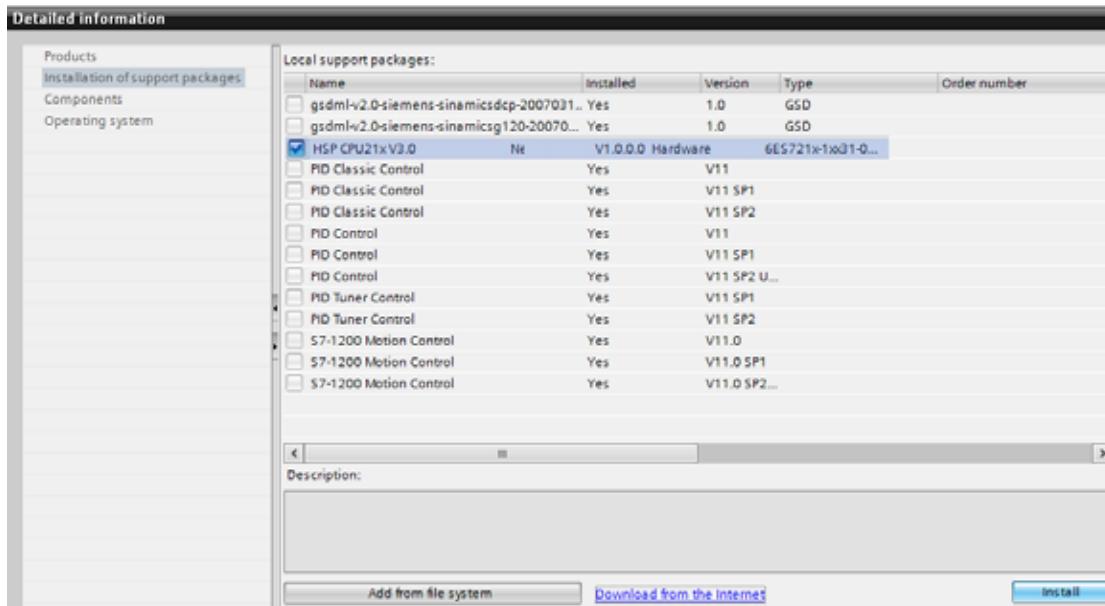
4. In this step, select the button “**Add from file system**” under “**Installation of support packages**”.
 (→ Installation of support packages → Add from file system)



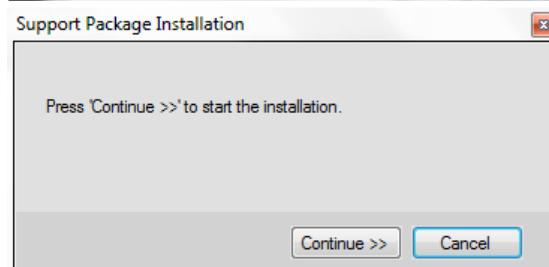
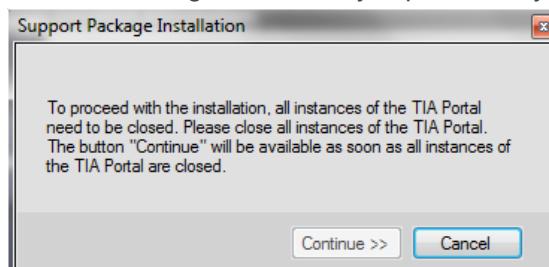
5. In step 5, you select the required Hardware Support Package in the format “*.isp11” and open it.
 (→ *.isp11 → Open)



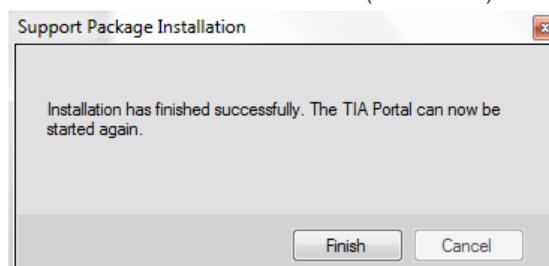
6. In step 6, you select the required Hardware Support Package under “**Installation of support packages**” and install it. (→ Installation of support packages → → Install)



7. If you see the following error message, the background in the TIA Portal must be closed, whereby the message window stays open. Then you can continue with the installation. (→ Continue)



8. The status of the installation is displayed until you see the final message to inform you that the TIA Portal must be restarted. (→ Finish)



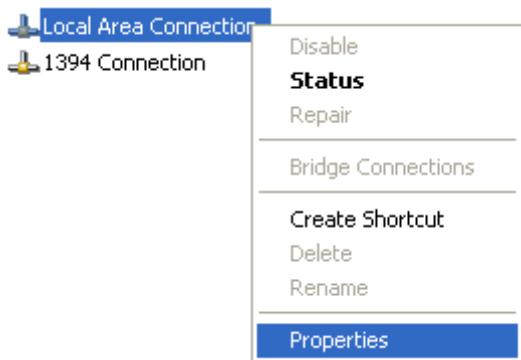
4. Connecting to the CPU by means of TCP/IP, and Resetting to Factory Setting

To program the SIMATIC S7-1200 from the PC, the PG or a laptop, you need a TCP/IP connection.

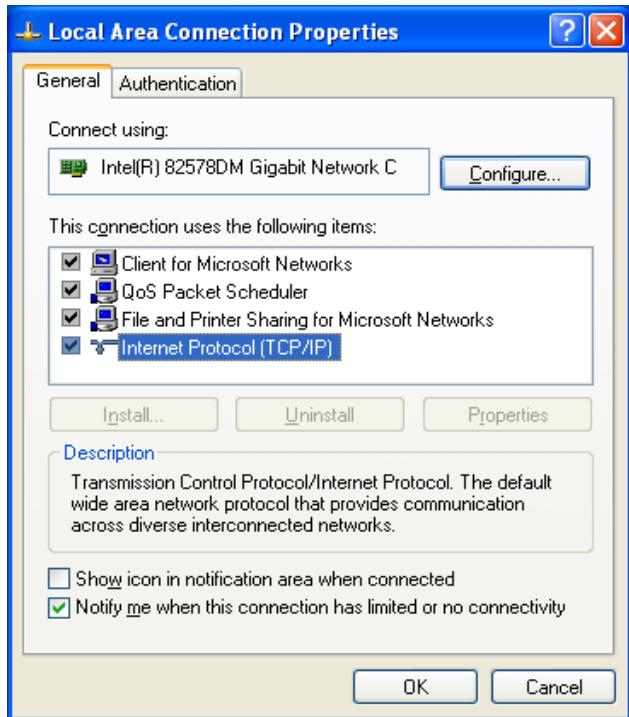
For the PC and the SIMATIC S7-1200 to communicate with each other, it is important also that the IP addresses of both devices match.

First, we show you how to set the computer's IP address.

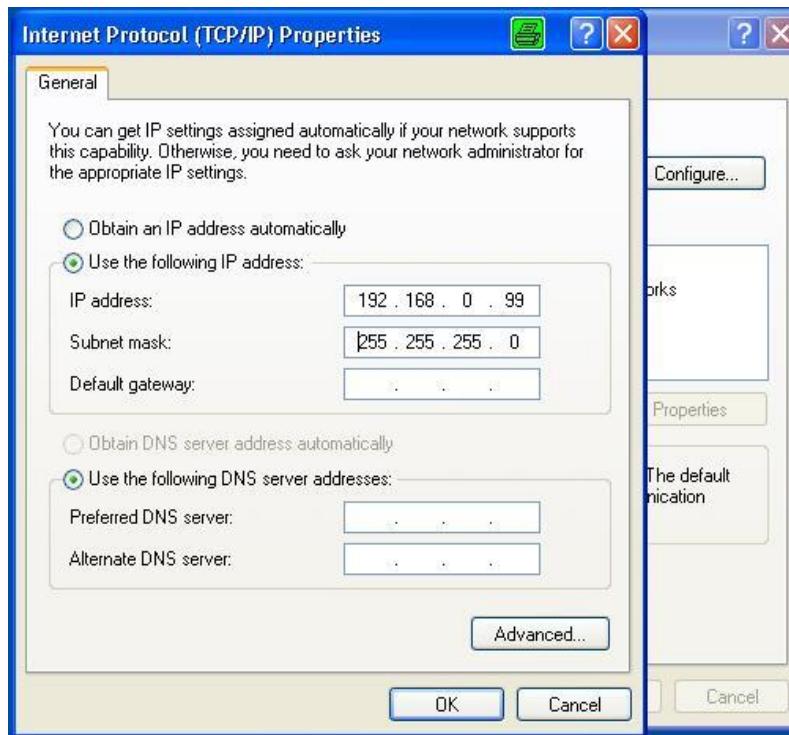
1. From the '**System control**', call the '**Network connections**'. Then, select the '**Properties**' of the LAN connection (→ Start → Settings → System control → Network connections → Local Area Connection → Properties)



2. Select the '**Properties**' from the '**Internet Protocol (TCP/IP)**' (→ Internet Protocol (TCP/IP) → Properties)



3. You can now set the '**IP address**' and the '**Subnet screen form**', and accept with '**OK**' (→ Use the following IP address → IP address: 192.168.0.99 → Subnet screen form 255.255.255.0 → OK → Close)



Notes on networking on the Ethernet (additional information is provided in Appendix V of the training curriculum):

MAC address:

The MAC address consists of a permanent and a variable part. The permanent part ("Basic MAC Address") identifies the manufacturer (Siemens, 3COM, etc.). The variable part of the MAC address differentiates the various Ethernet stations and should be assigned uniquely world-wide. On each module, a MAC address is imprinted specified by the factory.

Value range for the IP-address:

The IP address consists of 4 decimal numbers from the value range 0 to 255, separated by a period. For example, 141.80.0.16

Value range for the subnet screen form:

This screen form is used to recognize whether a station or its IP address belongs to the local subnetwork, or can be accessed only by means of a router.

The subnet screen form consists of four decimal numbers from the value range 0 to 255, separated by a period. For example, 255.255.0.0

In their binary representation, the 4 decimal numbers of the subnet screen form have to contain -from the left- a series of gapless values "1" and from the right a series of gapless values "0".

The values "1" specify the area of the IP address for the network number. The values "0" specify the area of the IP address for the station address.

Example:

Correct values: 255.255.0.0 Decimal = 1111 1111.1111 1111.0000 0000.0000 0000 binary
 255.255.128.0 Decimal = 1111 1111.1111 1111.1000 0000.0000 0000 binary
 255.254.0.0 Decimal = 1111 1111.1111 1110.0000 0000.0000.0000 binary

Wrong value: 255.255.1.0 Decimal = 1111 1111.1111 1111.0000 0001.0000 0000 binary

Value range for the address of the gateway (Router):

The address consists of 4 decimal numbers from the value range 0 to 255, separated by a period. For example, 141.80.0.1.

Relationship of IP addresses, router address, and subnet screen form:

The IP address and the gateway address are to differ only at positions where a "0" is located in the subnet screen form.

Example: You entered the following: for the subnet screen form 255.255.255.0, for the IP address 141.30.0.5 and for the router address 141.30.128.1.

The IP address and the gateway address must have a different value only in the 4th decimal number. However, in the example, the 3rd position already differs.

That means, in the example you have to change alternatively:

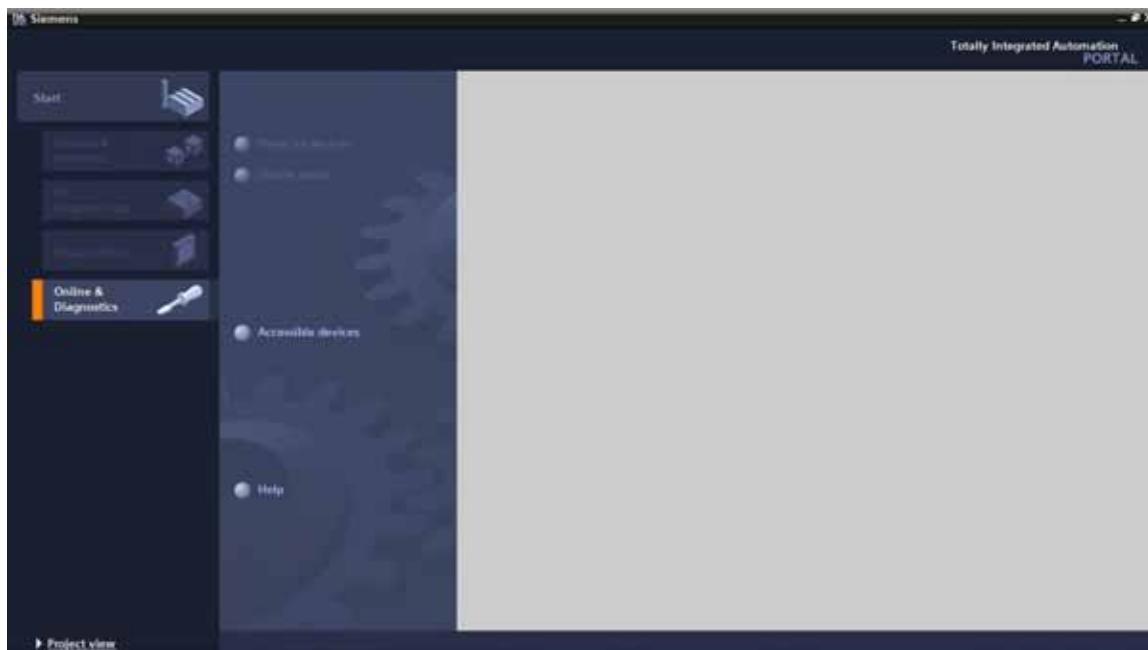
- the subnet screen form to: 255.255.0.0 or
- the IP address to: 141.30.128.5 or
- the gateway address to: 141.30.0.1

The SIMATIC S7-1200 IP address is set as follows:

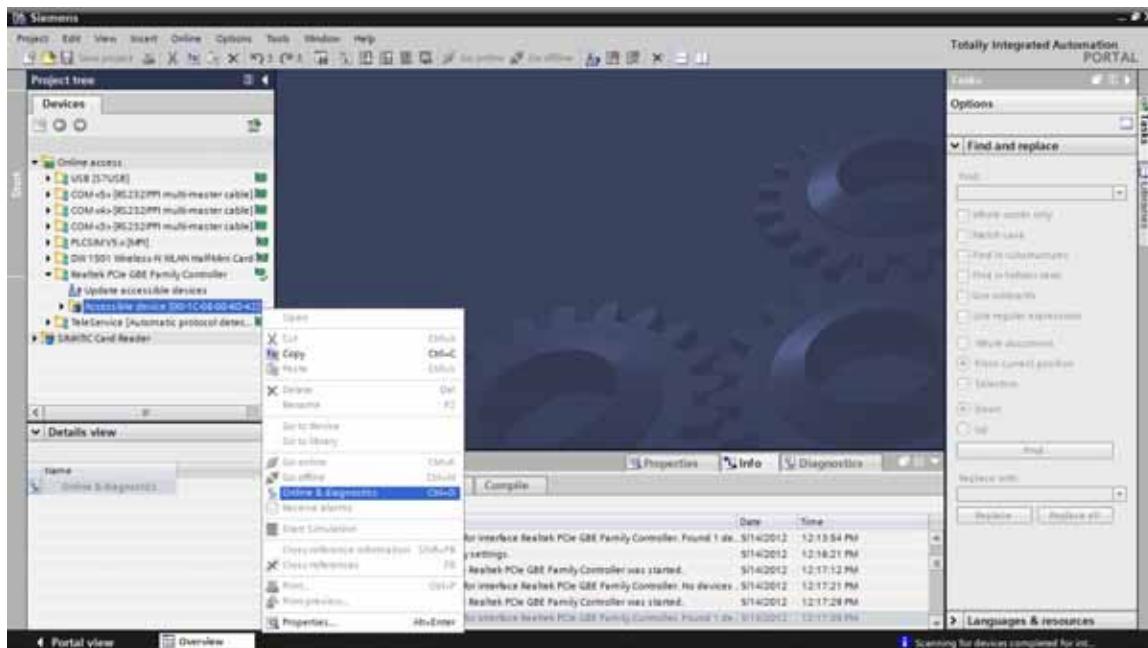
4. Select the '**Totally Integrated Automation Portal**'; it is called here with a double click (→ TIA Portal V11)



5. Then, select '**Online & Diagnostics**' and open '**Project View**' (→Online & Diagnosis → Project view)

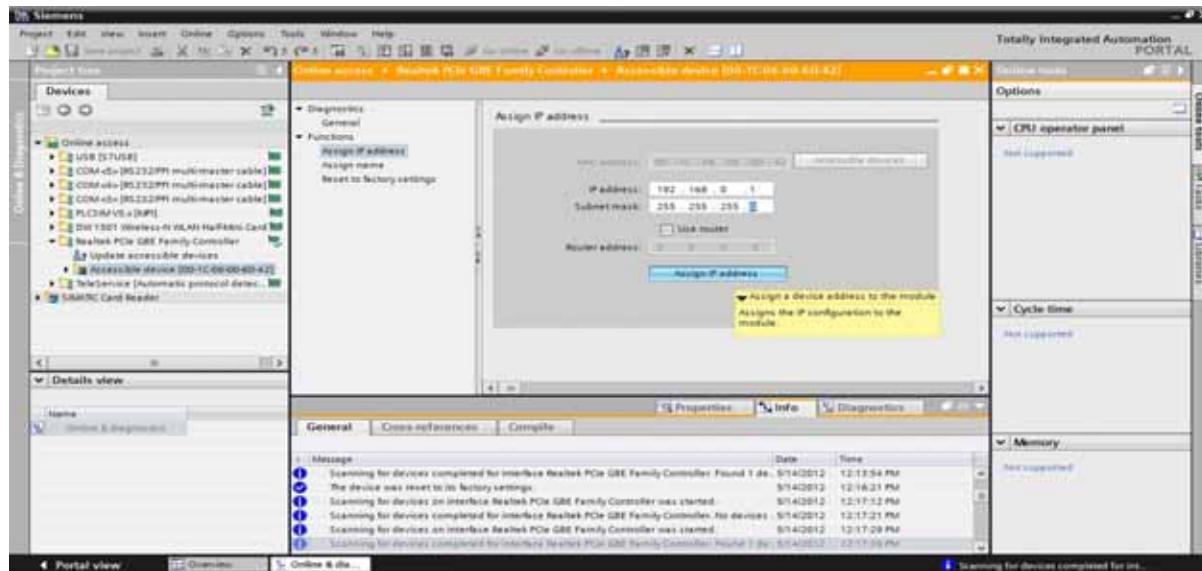


6. Next, in project navigation, select under '**Online accesses**' the network card that was already set beforehand. If you click here on '**Update accessible devices**', you will see the MAC address of the connected SIMATIC S7-1200. Select '**Online & Diagnostics**'. (→ Online accesses → ... Network Connection → Update accessible devices → MAC= → Online & diagnostics)

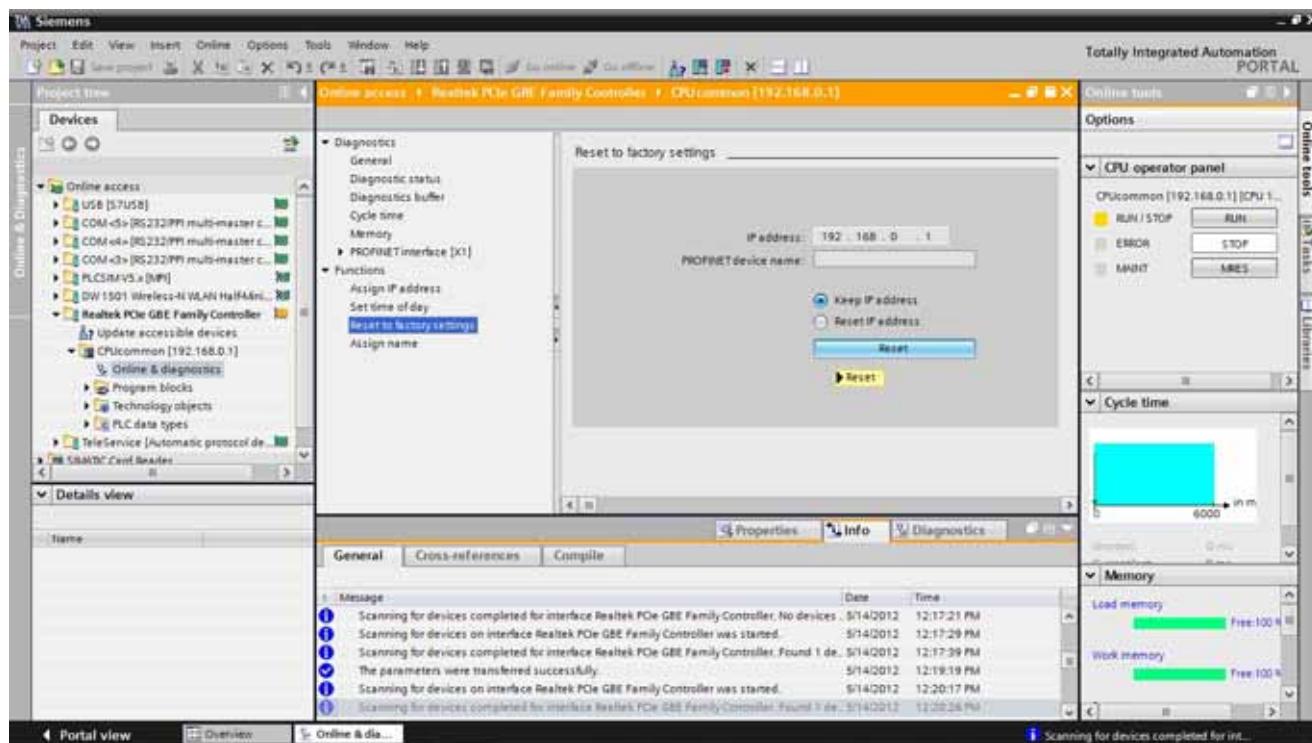


Note: If an IP address was set previously at the CPU, you will see this address instead of the MAC address.

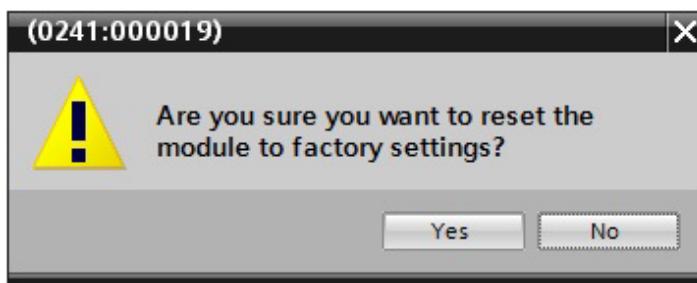
7. Under '**Functions**' you will see the item '**Assign IP address**'. Here, enter '**IP address**' and '**Subnet mask**'. Then, click on '**Assign IP address**', and your SIMATIC S7-1200 will be assigned this new address (→ Functions → Assign IP address → IP address: 192.168.0.1 → Subnet mask: 255.255.255.0 → Assign IP address)



8. Under '**Functions**', select '**Reset to factory settings**'. Keep this setting on '**Keep IP address**' and click on '**Reset**'. (→ Functions → Reset to factory settings → Keep IP address → Reset)



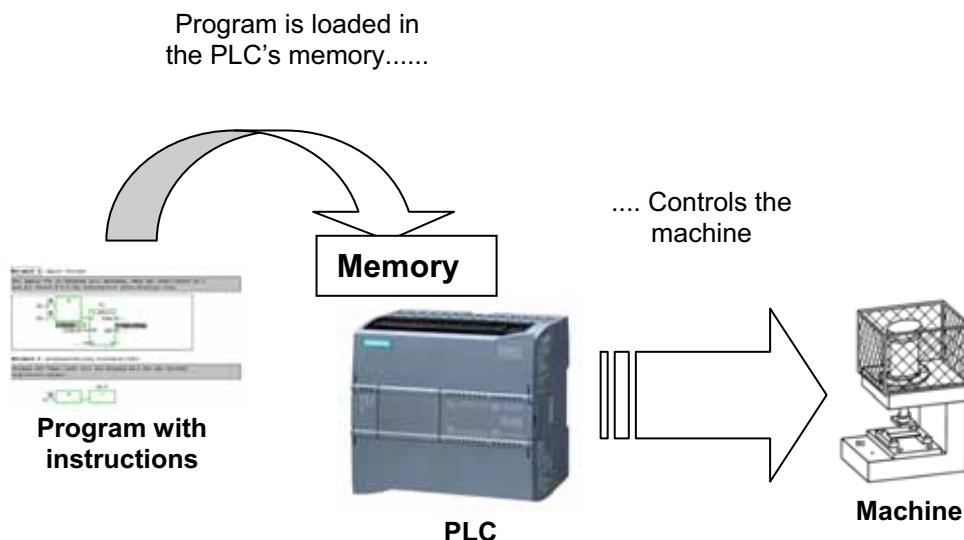
9. Confirm the prompt whether you want to go through with a reset to the factory setting with '**YES**' (→ OK)



5. What is a PLC and what are PLCs Used For?

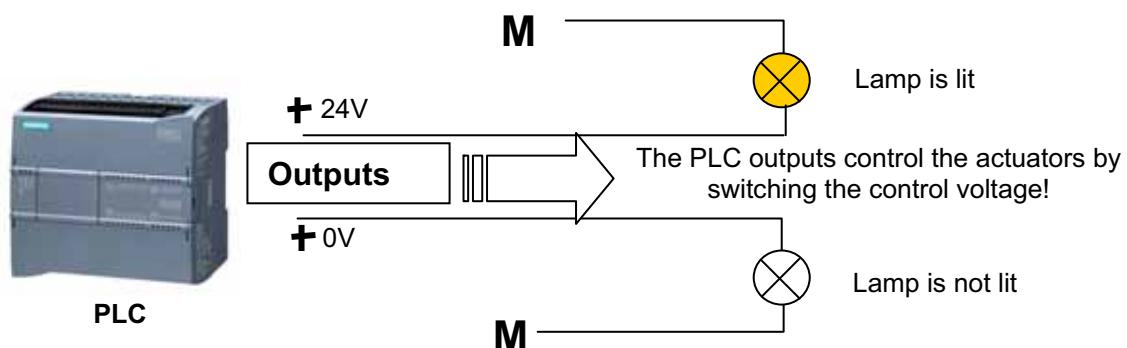
5.1 What does the term PLC mean?

PLC is the abbreviation for **Programmable Logic Controller**. It describes a device that controls a process (for example, a printing press for printing newspapers, a filling plant for filling cement in bags, a press for forming plastic shapes, etc....). This is performed according to the instructions of a program that is located in the memory of the device.



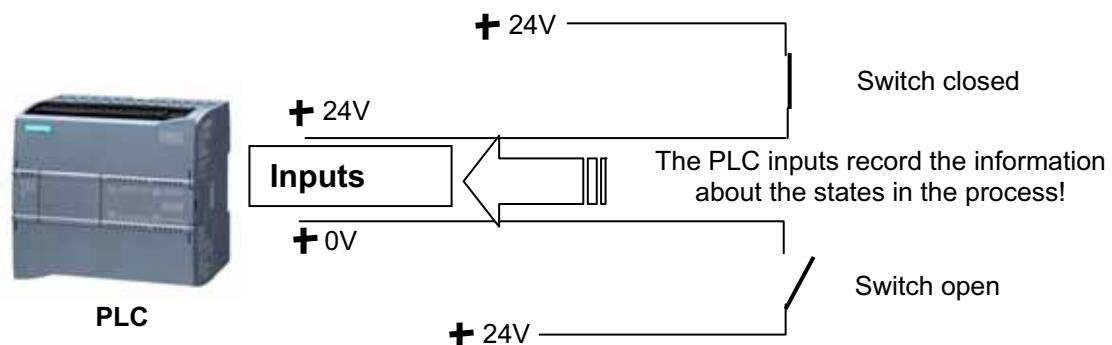
5.2 How does the PLC Control the Process?

The PLC controls the process as follows: through the PLC connections called **outputs**, so-called **actuators** are wired with a control voltage of 24V, for example. This allows for switching motors on and off, opening and closing valves, turning lamps on and off.



5.3 How does the PLC Get the Information about the Process States?

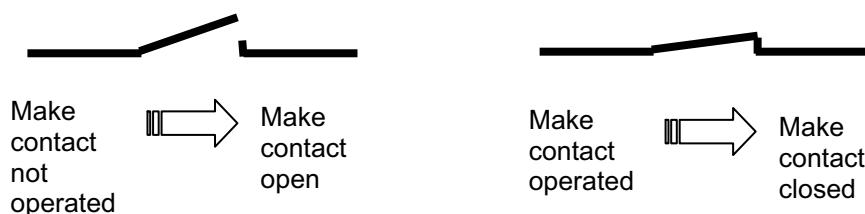
The PLC receives information about the process from the so-called **signal transmitters** that are wired to the **inputs** of the PLC. These signal transmitters can be, for example, sensors that recognize whether a work piece is in a certain position, or they can be simple switches and pushbuttons that may be open or closed. Here, we differentiate between **break contact elements** that are closed if not operated, and **make contact elements** that are open if not activated.



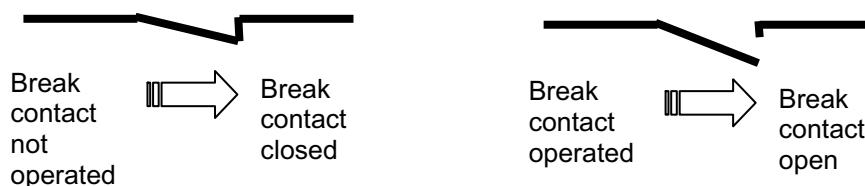
5.4 What is the Difference between Break Contact Elements and Make Contact Elements?

As to the signal transmitters, we differentiate between **break contacts** and **make contacts**.

The switch shown below is a make contact; i.e., it is closed exactly when it was operated



The switch shown below is a break contact; i.e., it is closed exactly when it was not operated.



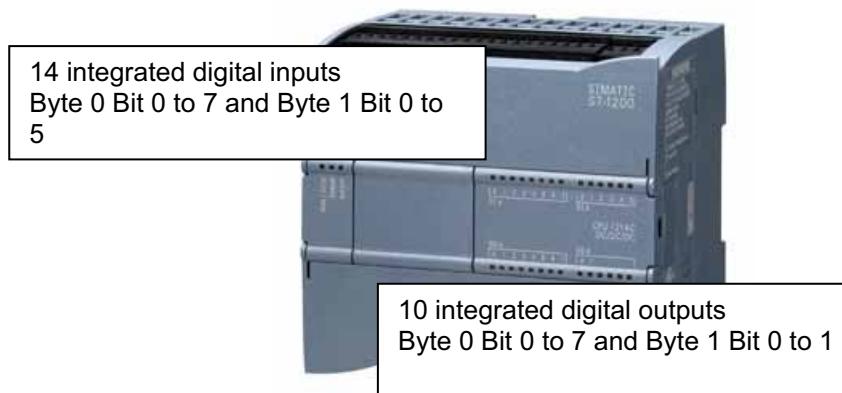
5.5 How does the SIMATIC S7-1200 Address Individual Input/Output Signals?

Specifying a certain input or output within the program is called addressing.

The PLC inputs and outputs are usually combined into groups of 8 on digital input modules and digital output modules. This unit of 8 is called a **byte**. Each such group receives a number as the so-called **byte address**.

In order to address a single input or output within a byte, each byte is broken down into 8 individual **bits**. These are numbered Bit 0 to Bit 7. This is how we arrive at the **bit address**.

The PLC shown here has the input bytes 0 and 1 as well as the output bytes 0 and 1.



To address the fifth digital input, for example, we specify the following address:

%I 0 . 4

%I here indicates the address type as input, **0** the byte address, and **4** the bit address.
Byte address and bit address are always separated by a period.

Note: For the bit address, a **4** is shown for the fifth input, because we start counting with 0.

To address the 10th output, for example, we specify the following address:

%Q 1 . 1

%Q here indicates the address type as output, **1** the byte address, and **1** the bit address.
Byte address and bit address are always separated by a period.

Note: For the bit address, a **1** is shown at the 10th output, because we start counting with 0.

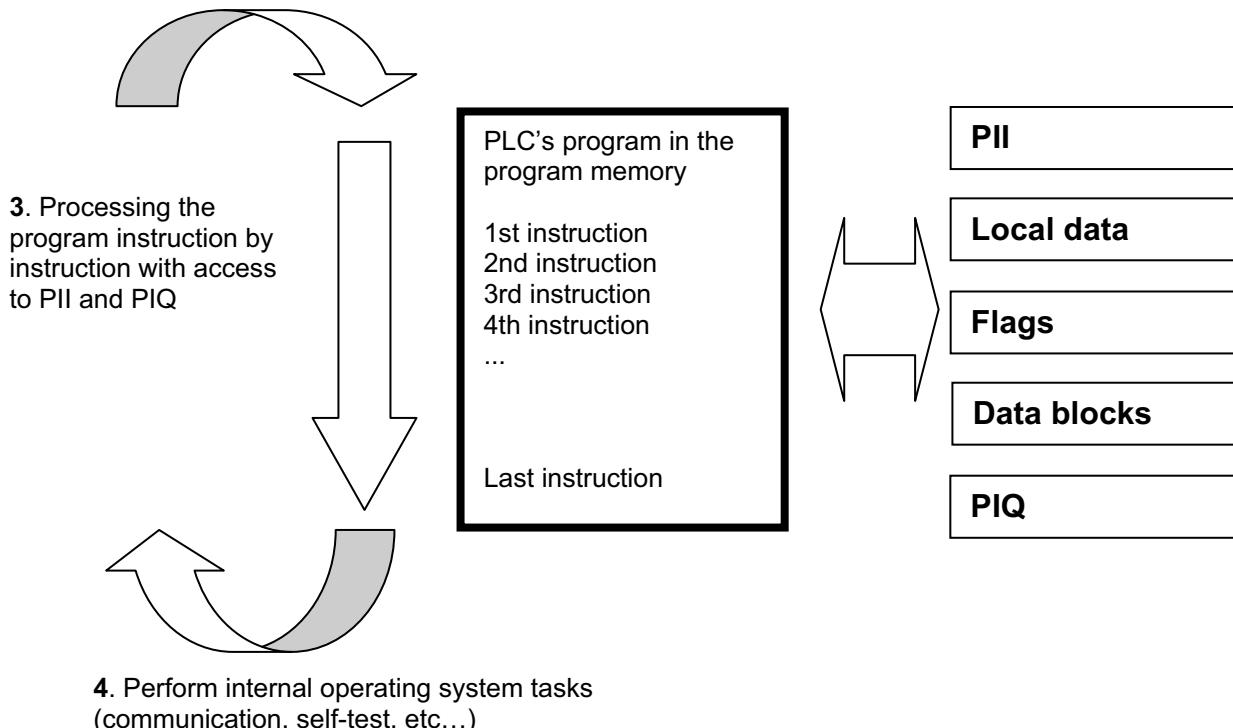
5.6 How is the Program Processed in the PLC?

The program is processed in the PLC cyclically, in the following sequence:

1. First, the status is transferred from the process image of the outputs (**PIQ**) to the inputs, and switched on or off.
2. Then the **processor** -which is practically the PLC's brain- inquires whether the individual inputs are carrying voltage. This status of the inputs is stored in the process image of the inputs (**PII**). For the inputs that carry voltage, the information 1 or "High" is stored, for those that don't the information 0 or "Low".
3. The processor then processes the program stored in the program memory. The program consists of a list of logic operations and instructions that are processed one after the other. For the required input information, the processor accesses the PII that was entered previously, and the result of the logic operation (RLO) is written into a process image of the outputs (**PIQ**). If necessary, the processor also accesses other memory areas during program processing; for example, for local data of sub-programs, data blocks and flags.
4. Then, internal operating system tasks such as self tests and communication are performed.

Then we continue with Item 1.

1. Transfer the status from the PIQ to the outputs.
2. Store the status of the inputs in the PII.



Note: The time the processor needs for this sequence is called cycle time. In turn, the cycle time depends on the number and type of instructions and the processor capacity.

5.7 What do Logic Operations Look Like in the PLC Program?

Logic operations are used to specify conditions for switching an output.

In the PLC program, these can be programmed in the programming languages Ladder Diagram (**LAD**) or Function Block Diagram (**FBD**).

For illustrative purposes, we will use **FBD** here.

There are a large number of logic operations that can be used in PLC programs.

However, **AND** as well as **OR** operations and the **NEGATION** of an input are used most frequently and are explained briefly below, using examples.

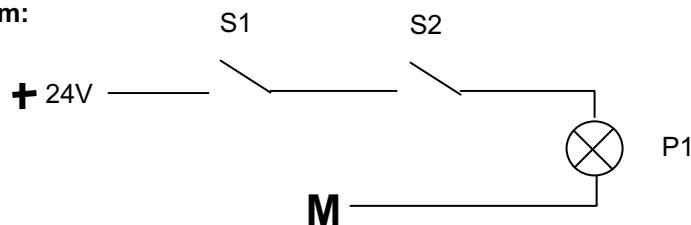
Note: Clearly arranged information about additional logic operations can be obtained quickly in online help.

5.7.1 AND Operation

Example of an AND operation:

A lamp is to light up when two switches are operated simultaneously as make contacts.

Diagram:



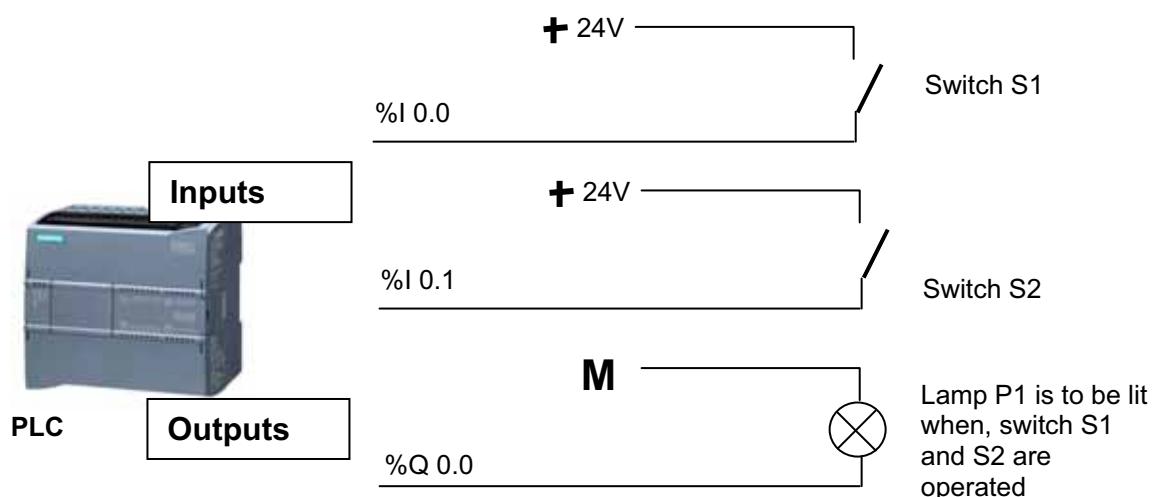
Explanation:

The lamp lights up exactly when both switches are operated.
That is, when switches S1 **and** S2 are operated, lamp P1 is lit.

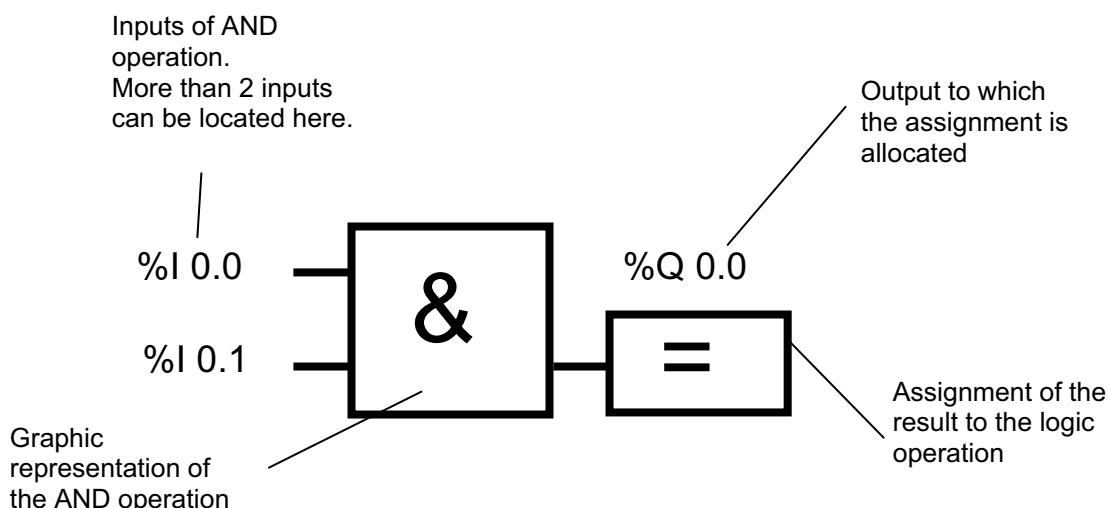
Wiring the PLC:

To apply this logic to a PLC program, both switches have to be connected to inputs of the PLC. Here, S1 is wired to input I 0.0 and S2 to input I 0.1.

In addition, lamp P1 has to be connected to an output; for example Q 0.0.

**AND logic in the FBD:**

In the function block diagram FBD, the AND operation is programmed using a graphic representation, and looks like this:

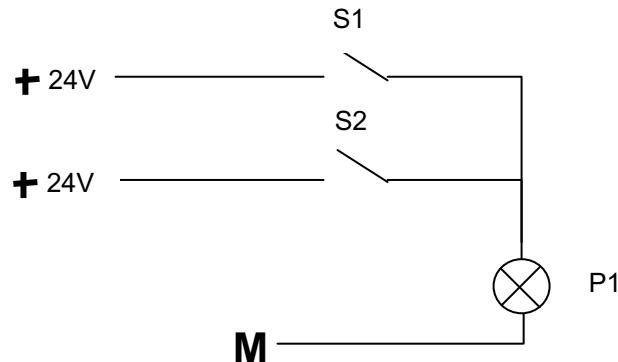


5.7.2 OR Operation

Example of an OR operation:

A lamp is to light up when one or both of two switches are operated as make contacts.

Diagram:



Explanation:

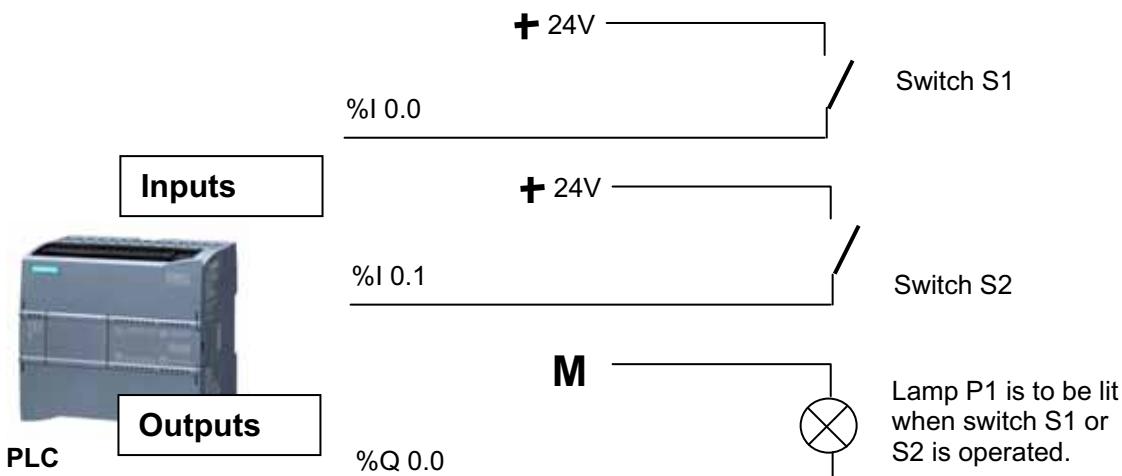
The lamp lights up exactly when one or both switches are operated.

That is, if switch S1 **or** S2 is operated, lamp P1 is lit.

Wiring the PLC:

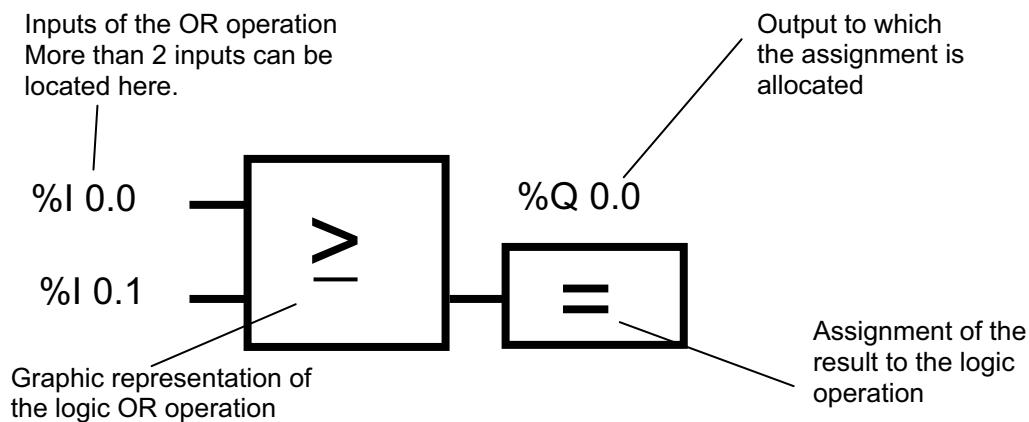
To apply this logic to a PLC program, both switches have to be connected to inputs at the PLC, of course. Here, S1 is connected to input I 0.0 and S2 to input I 0.1.

In addition, lamp P1 has to be connected to an output; for example, Q 0.0.



OR operation in the FBD:

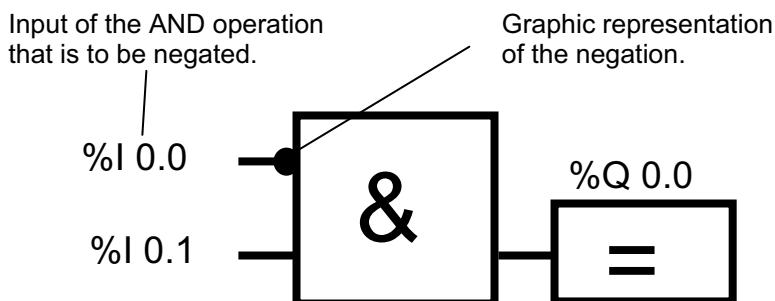
In the function plan FBD, the OR operation is programmed using a graphic representation, and looks like this:



5.7.3 Negation

In logic operations, it is often necessary to inquire whether a **make contact** was **NOT operated** or whether a **break contact** was **operated** and thus no voltage is applied to the corresponding input. This happens when we program a **Negation** at the input of the AND or OR operation.

In the function block diagram FBD, the negation of an input at an AND operation is programmed with the following graphic representation:

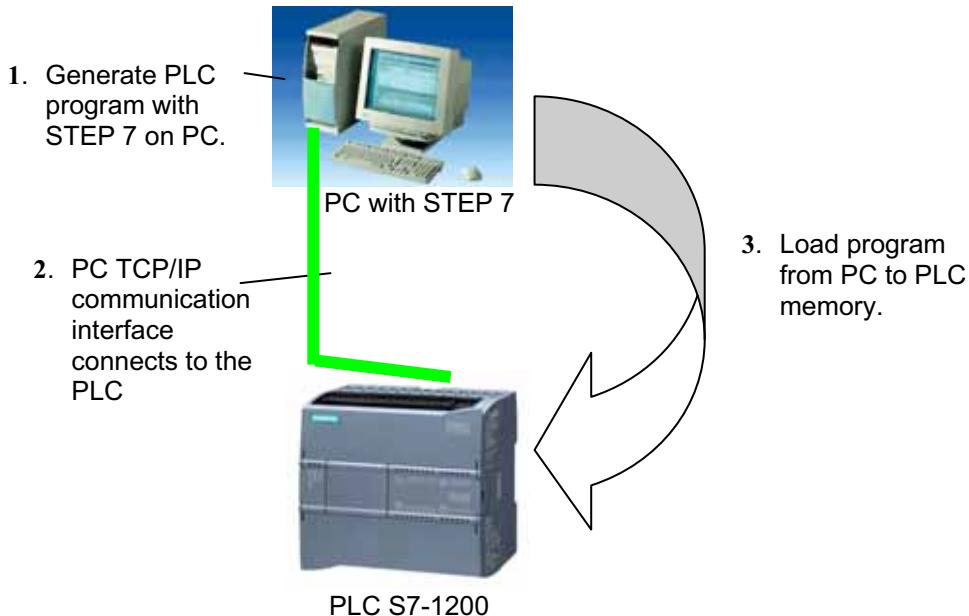


This means voltage is applied to the output %Q 0.0 exactly when %I 0.0 is not connected and %I 0.1 is connected.

5.8 How is the PLC Program generated? How does it get to the PLC's memory?

The PLC program is generated on a PC using the software STEP 7, and temporarily stored there. After the PC is connected with the TCP/IP interface of the PLC, the program can be transferred with a load function to the PLC's memory.

The PC is no longer needed for further program processing in the PLC.



Note: The exact sequence is described step by step in the chapters below.

6. Configuring and Operating the SIMATIC S7-1200

Module Spectrum:

The SIMATIC S7-1200 is a modular automation system and offers the following module spectrum:

- Central modules CPU with different capacity, integrated inputs/outputs and PROFINET interface (for example, CPU1214C)



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



- Signal boards SB for adding analog or digital inputs/outputs; whereby the size of the CPU does not change.
(signal boards can be used with the CPUs 1211C/1212C and 1214C)



- Signal modules SM for digital and analog inputs and outputs
(for CPUs 1212C a maximum of 2 SMs can be used, for 1214C a maximum of 8)



- Communication modules CM for serial communication RS 232/RS 485
(for CPUs 1211C/1212C and 1214C, up to 3 CMs can be used)



- Compact Switch Module CSM with 4x RJ45 socket connectors 10/100 Mbps



- SIMATIC memory cards 2MB or 24MB for storing program data and simple CPU replacement for maintenance

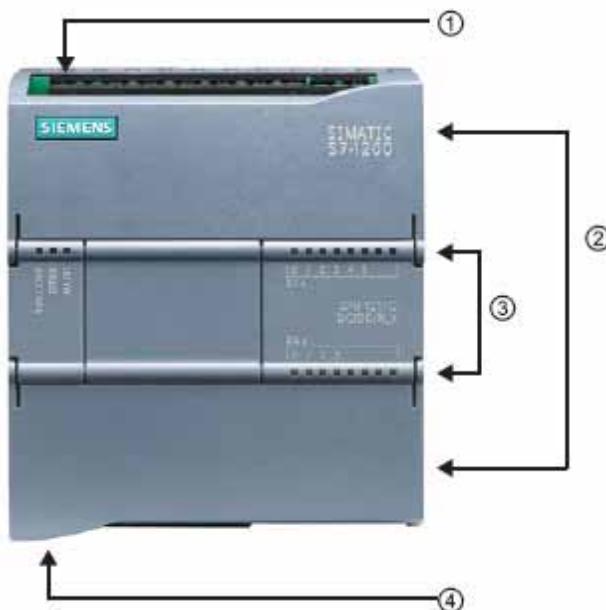


Note: For this module 010-010, any CPU with integrated digital inputs and digital outputs is sufficient.

Important CPU elements:

With an integrated voltage supply (24V connection) and integrated inputs and outputs, the S7-1200 CPU is ready, without additional components.

To communicate with a programming device, the CPU is equipped with an integrated TCP/IP port. By means of an ETHERNET network, the CPU is able to communicate with HMI operator devices or other CPUs.



- ① Connection 24V
- ② Plug-in terminal block for user wiring (behind the cover plates)
- ③ Status LEDs for the integrated I/O and the CPU's operating mode
- ④ TCP/IP connection (on the lower side of the CPU)

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

- Transferring a program to several CPUs
- Firmware update of CPUs, signal modules SM and communication modules CM

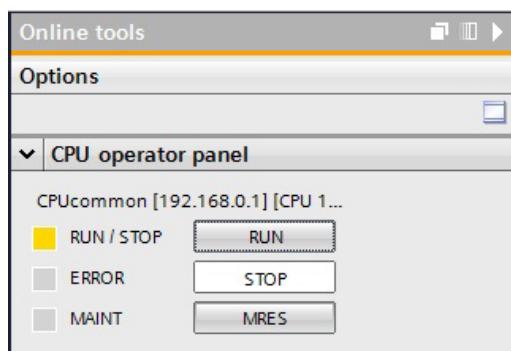


Operating Modes of the CPU

The CPU has the following operating modes:

- In the operating mode **STOP**, the CPU does not execute the program, and you can load a project.
- In the operating mode **STARTUP**, the CPU performs a startup.
- In the operating mode **RUN**, the program is executed cyclically. Projects cannot be loaded in the CPU's RUN mode.

The CPU does not have a physical switch for changing the operating mode. The operating mode (**STOP** or **RUN**) is changed by using the button on the operator panel of the software STEP7 Basic. In addition, the operator panel is provided with the button **MRES** to perform a general memory reset and displays the status LEDs of the CPU.



The color of the **status LED RUN/STOP** on the front of the CPU indicates its current operating mode.



- **Yellow** light indicates the **STOP** mode.
- **Green** light indicates the **RUN** mode.
- **Flashing** light indicates the **STARTUP** mode.

In addition, there are LEDs for **ERROR** to indicate errors and **MAINT** to indicate that maintenance is required.

7. Sample Task: Controlling a Press

Our first program consists of programming a press control.

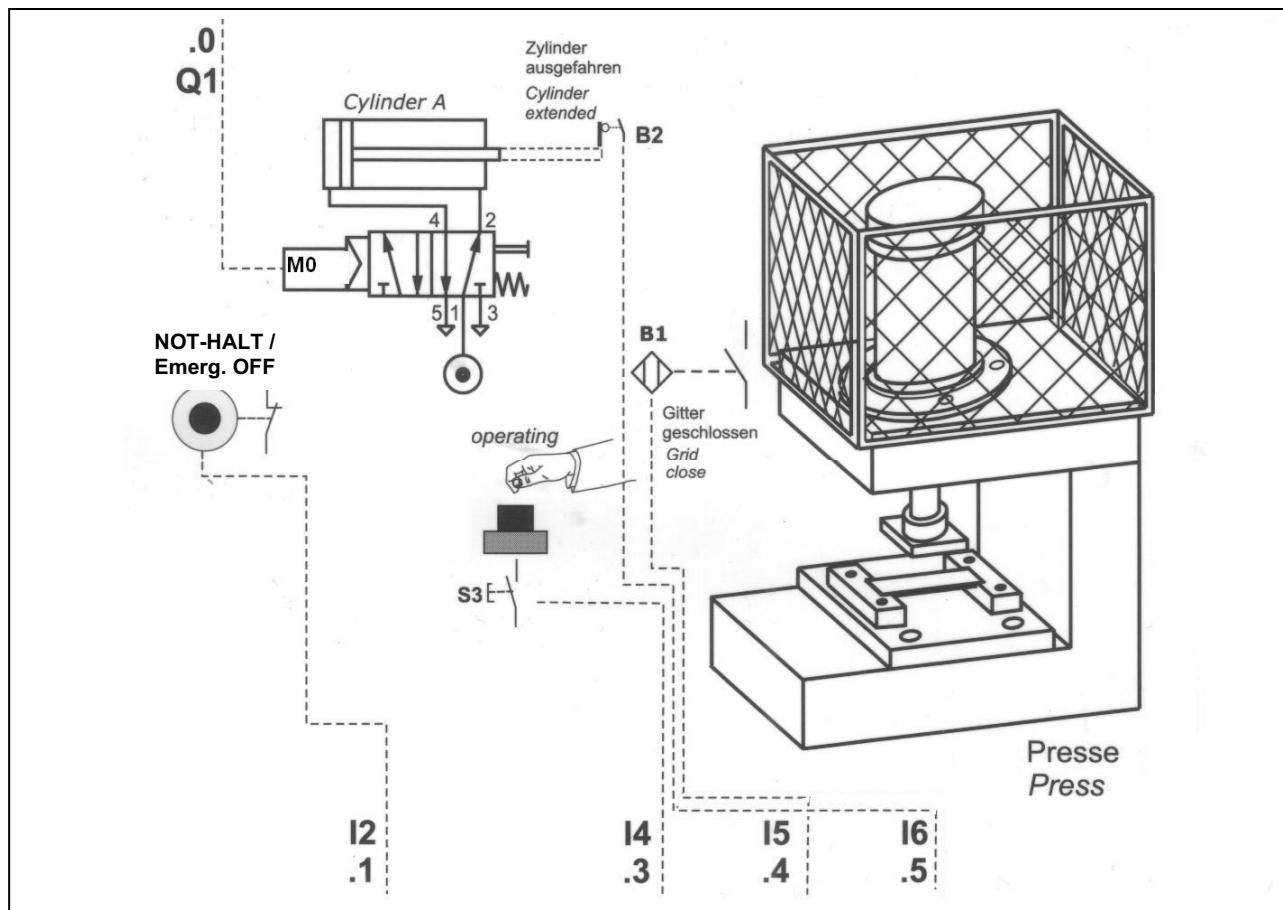
A press with a safety fence is to be activated with a START button S3 only if the safety fence is closed. This state is monitored with a sensor Safety Fence Closed B1.

If this is the case, a 5/2 way valve M0 for the press cylinder is set so that a plastic shape can be pressed.

The press is to retract again when the EMERGENCY OFF button (NC) is activated, or the sensor safety fence B1 no longer responds, or the sensor Cylinder B2 responds Extended.

Assignment list:

Address	Symbol	Comment
%I 0.1	EMERGENCY OFF	EMERGENCY OFF button NC
%I 0.3	S3	Start button S3 NO
%I 0.4	B1	Sensor safety fence closed NO
%I 0.5	B2	Sensor Cylinder extended NO
%Q 0.0	M0	Extend Cylinder A



8. Programming the Press for the SIMATIC S7-1200

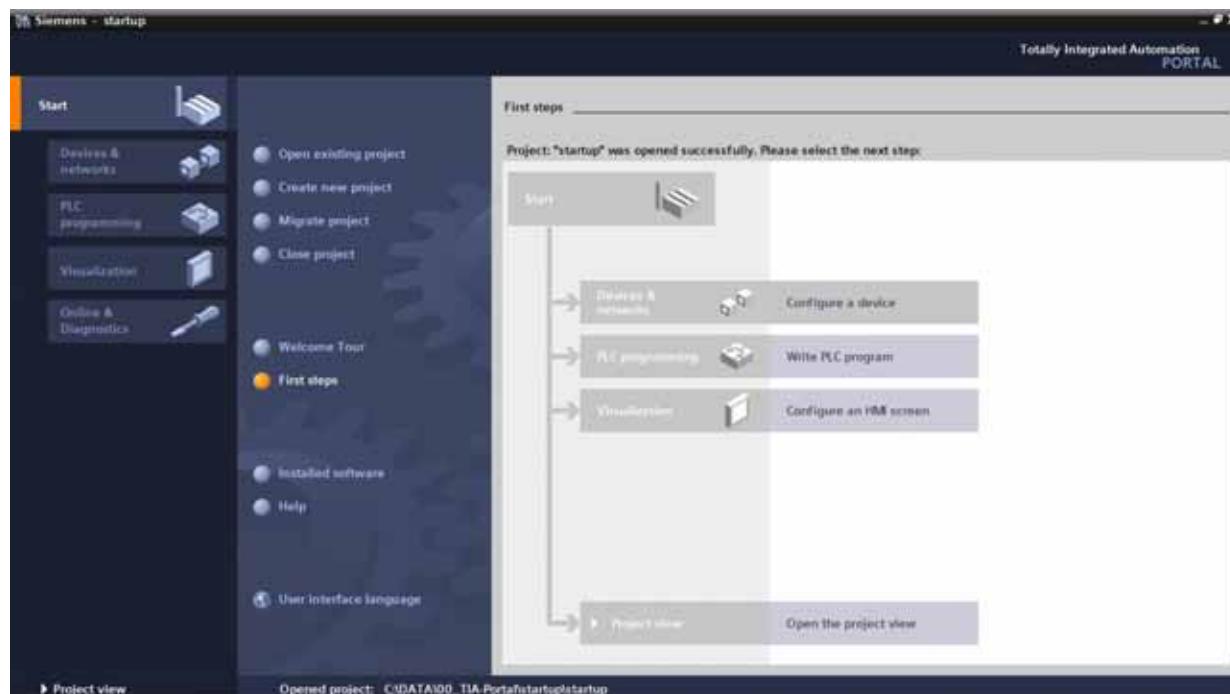
The software '**Totally Integrated Automation Portal**' manages the project and does the programming.

Here, under a uniform interface, the components such as the controller, visualization and networking the automation solution are set up, parameterized and programmed.
Online tools are provided for error diagnosis.

The software '**Totally Integrated Automation Portal**' has two different views: the portal view and the project view.

8.1. Portal View

The portal view provides a task oriented view of the tools for processing the project. Here, you can quickly decide what you want to do, and call the tool for the respective task. If necessary, a change to the project view takes place automatically for the selected task. Primarily, getting started and the first steps are to be facilitated here.

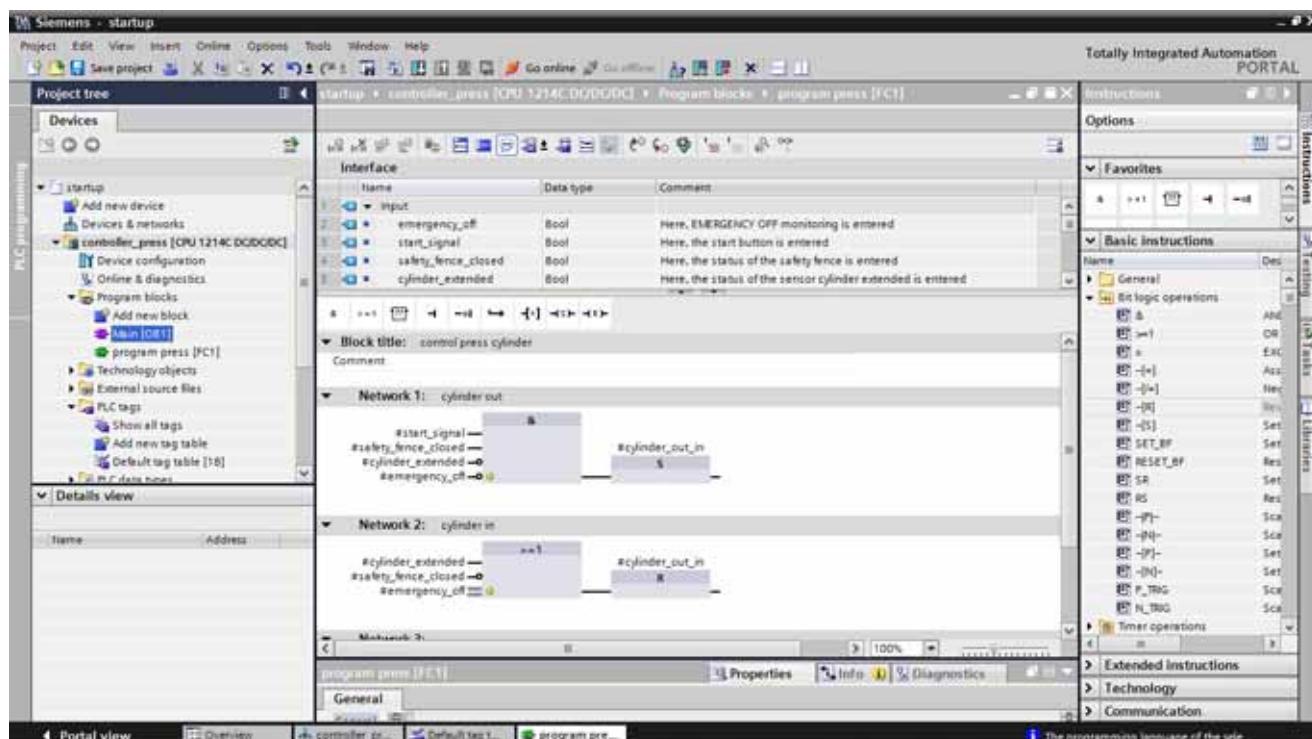


Note: On the lower left, you can jump from the portal view to the project view.

8.2. Project View

The project view is a structured view to all constituent parts of the project. As a matter of standard, the menu bar with the function bars is located on top, project navigation with all the parts of a project on the left, and the task cards -with instructions and libraries, for example, on the right.

If an element (here, for example, program block FC1) is selected in project navigation, it is displayed in the center and can be processed there.



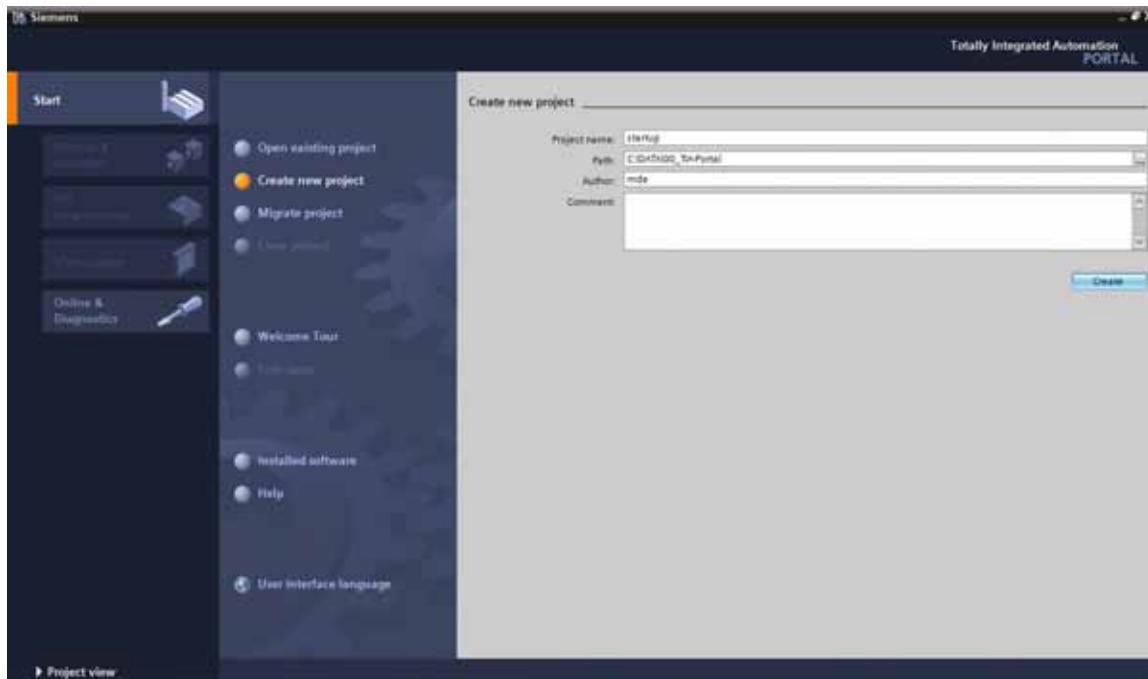
Note: On the lower left, you can jump from the project view to the portal view

With the following steps, we are setting up a project for the SIMATIC S7-1200 and we are programming the solution of the task:

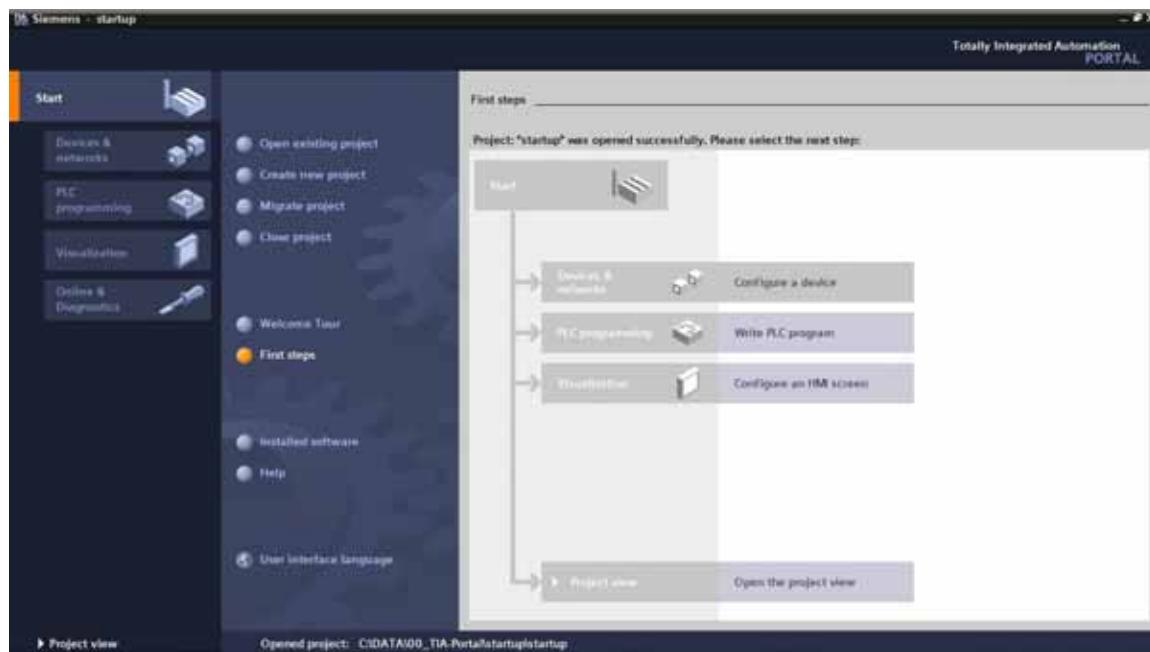
1. The central tool is the '**Totally Integrated Automation Portal**' which is called here with a double click.
(→ Totally Integrated Automation Portal V11)



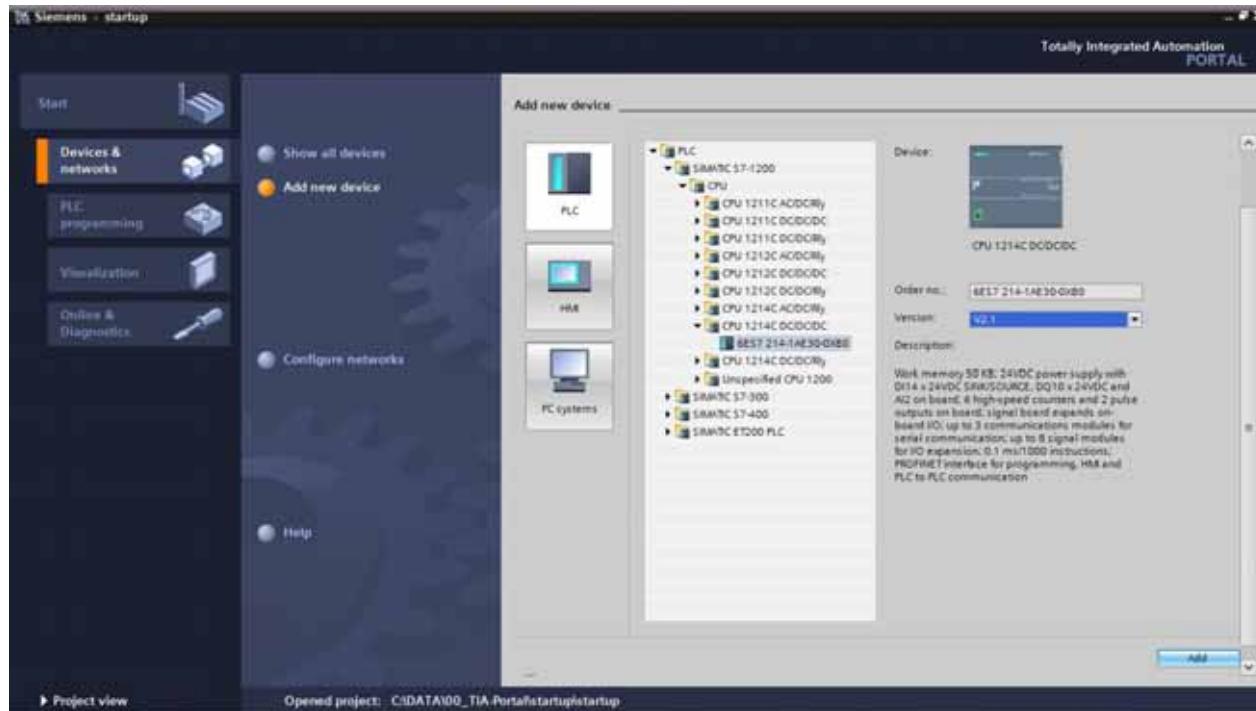
2. Programs for the SIMATIC S7-1200 are managed in projects. Such a project is now set up in the portal view. (→ Create new project → startup → Create)



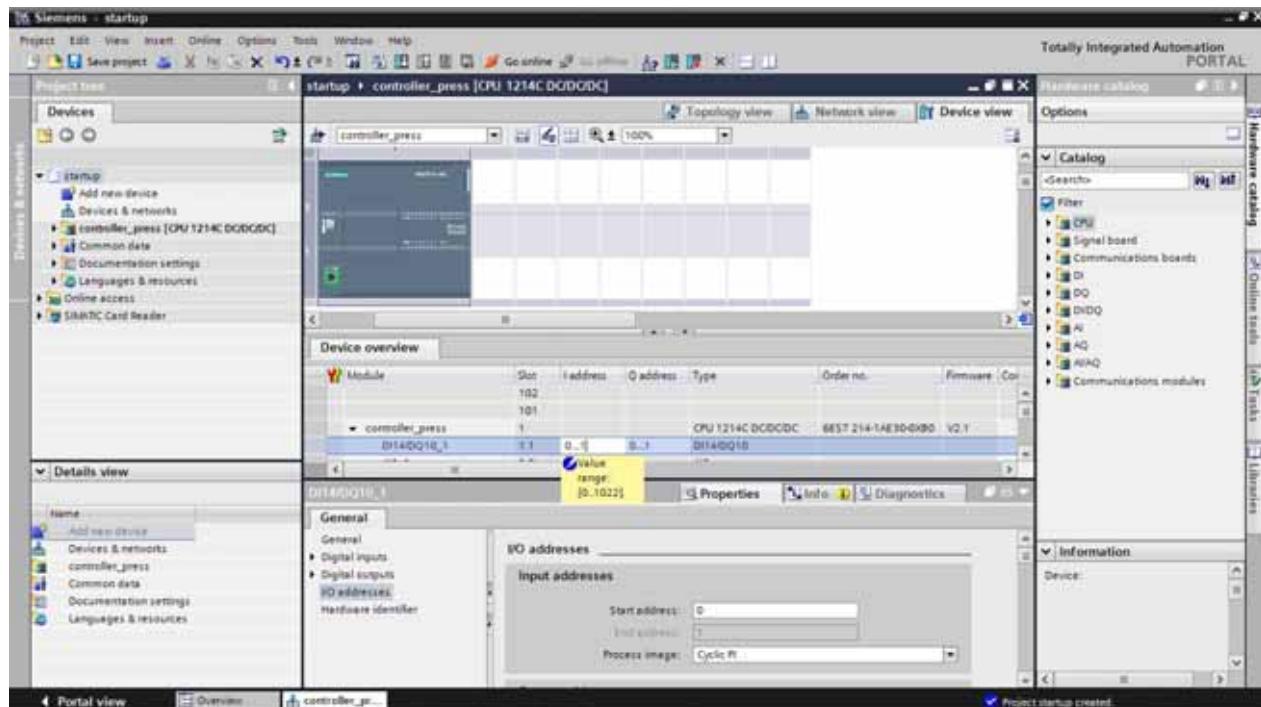
3. Now '**First Steps**' are suggested for the configuration. First, we want to '**Configure a device**'. (→ First steps → Configure a device)



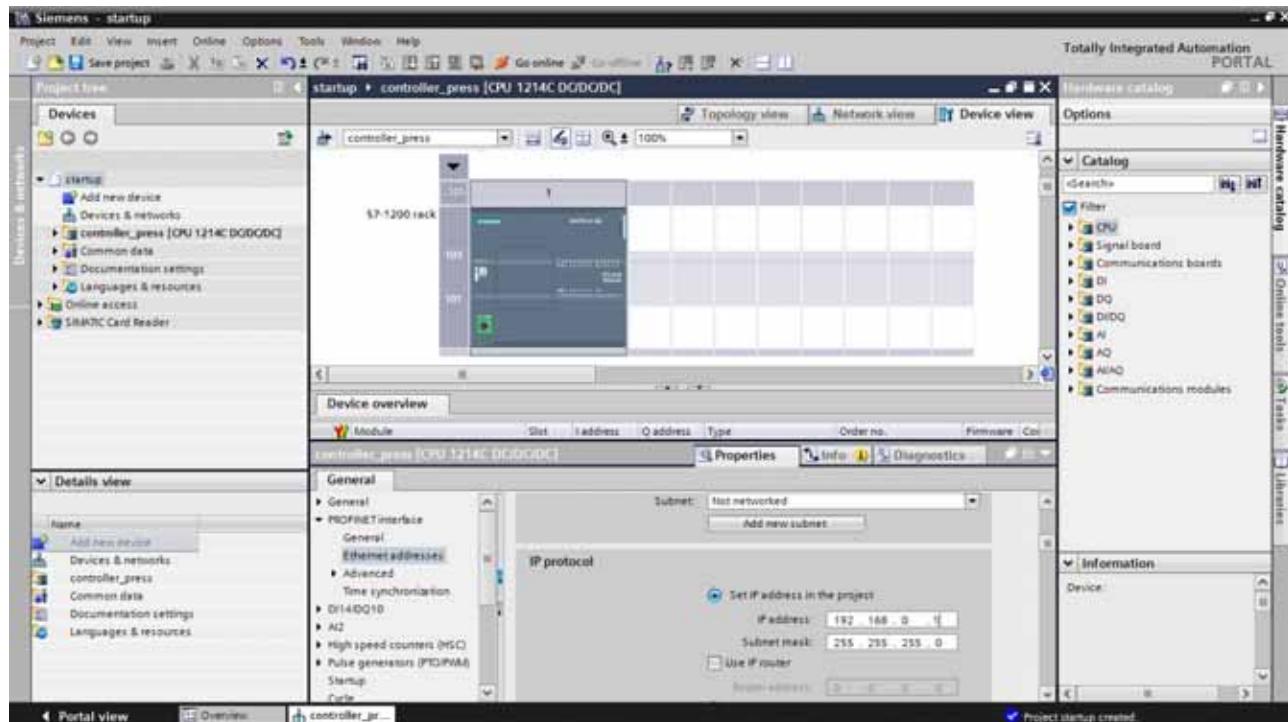
4. Then, we '**Add new device**' with the '**Device name Controller Press**'. To this end, we select from the catalog the '**CPU1214C**' with the matching order number (→ Add new device → Controller press → CPU1214C → 6ES7 → Insert)



5. Now, the software automatically changes to the project view with the opened hardware configuration. Here, additional modules can be inserted from the hardware catalog (to the right!), and in the 'Device overview', the input/output addresses can be set. Here, the integrated inputs of the CPU have the addresses %I0.0 to %I1.5 and the integrated outputs the addresses %Q0.0 to %Q1.1
 (→ Device overview → DI14/DO10 → 0...1)



6. So that the software later accesses the correct CPU, its IP address and the subnet screen form have to be set (→ Properties → General → PROFINET interface → IP address: 192.168.0.1 → Subnet screen form: 255.255.255.0)
 (refer also to Chapter 3 for setting the programming interface)



7. Since for modern programming, we don't program with absolute addresses but with tags, we have to specify the **global PLC tags** here.

These global PLC tags are descriptive names with comments for those inputs and outputs that are used in the program. Later, during programming, the global PLC tags can be accessed by means of this name.

These global tags can be used in the entire program in all blocks.

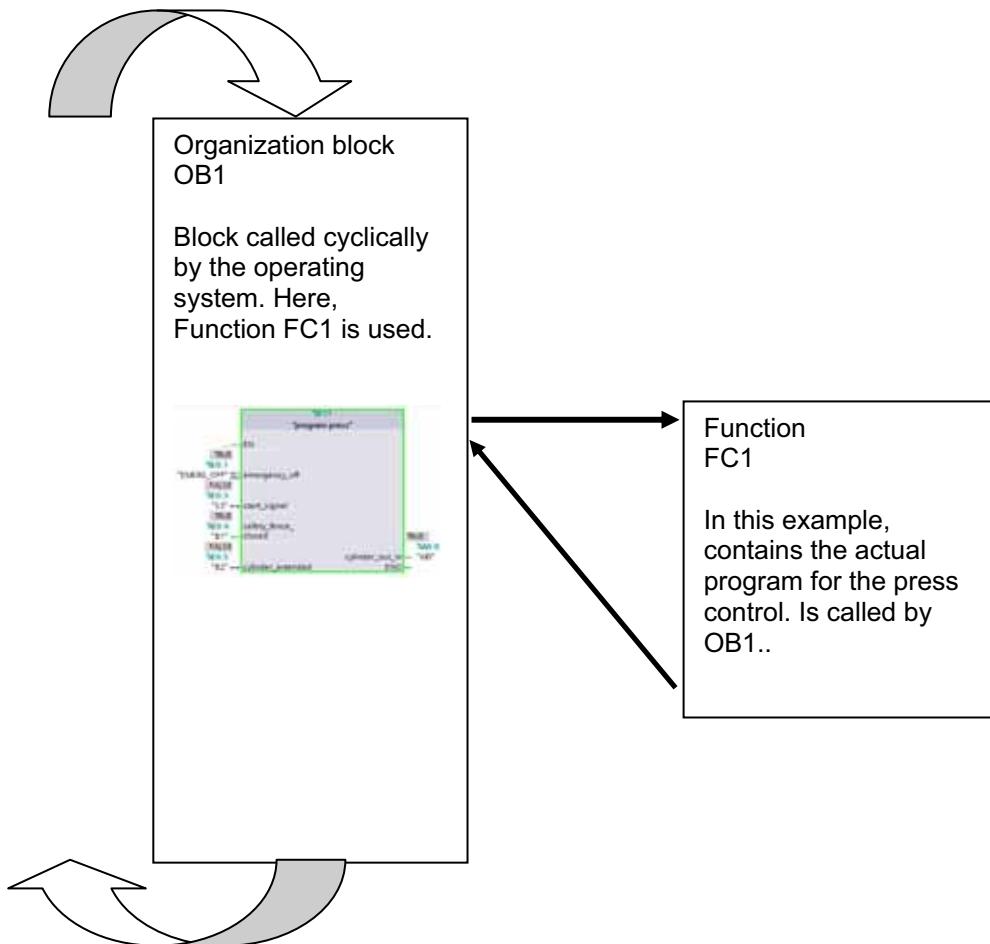
To this end, first select in project navigation the '**Controller Press[CPU1214C DC/DC/DC]**' and then '**PLC tags**'. With a double click, open the table '**Default tag table**' and as shown below, enter the names for the inputs and outputs

(→ Controller Press[CPU1214C DC/DC/DC] → PLC tags → Default tag table)

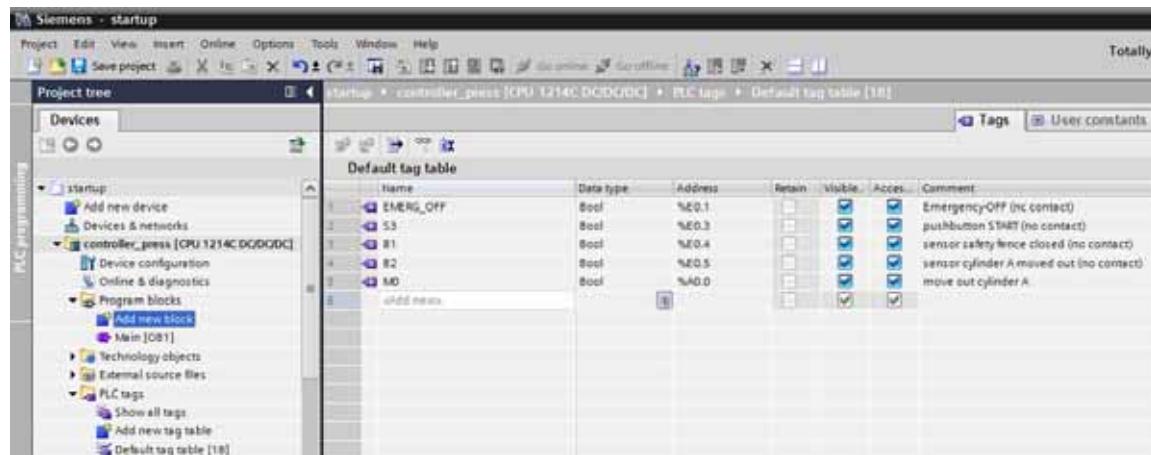
Name	Data type	Address	Retain	Visible	Access	Comment
EMERG_STOP	Bool	%E0.1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Emergency-OFF (no contact)
S3	Bool	%E0.3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	pushbutton START (no contact)
I1	Bool	%E0.4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	sensor safety fence closed (no contact)
I2	Bool	%E0.5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	sensor cylinder A moved out (no contact)
M0	Bool	%D0.0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	move out cylinder A

8. The program sequence is written in so-called blocks. As a matter of standard, organization block OB1 already exists.
 It represents the interface to the CPU's operating system, is called by it automatically, and processed cyclically.
 From this organization block, additional blocks can be called in turn for structured programming, such as the function FC1.
 The purpose is to break down an overall task into partial tasks. These can be solved more easily and tested in their functionality.

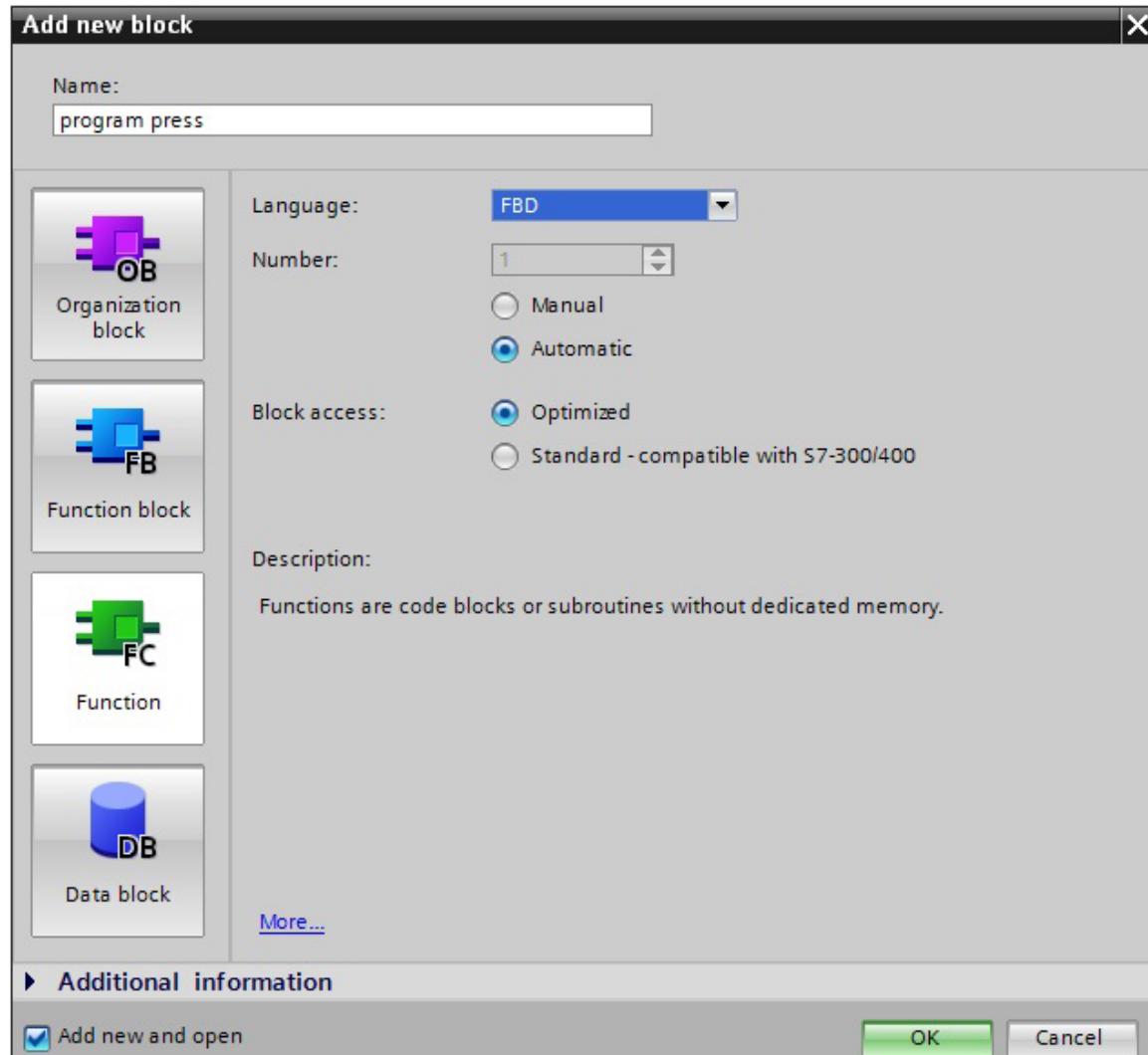
Program structure of the example:



9. To generate the function FC1, in Project Navigation first select '**Controller Press[CPU1214C DC/DC/DC]**' and then '**Program blocks**'. Next, double click on '**Add new block**'
 (→ Controller Press[CPU1214C DC/DC/DC] → Program blocks → Add new block)



10. In the selection, select '**Function (FC)**' and assign the name '**Program press**'. As programming language, '**FBD**' is entered. Enumeration is automatic. Since this FC1 is called later with the symbolic name anyhow, the number is no longer that important. Accept your input with '**OK**'.
 (→ Function (FC) → Program Press → FBD → OK)



Note

Normally, a function block with the associated instance data block would have to be used for a latching function. Since in our program example, however, saving takes place outside the block by means of an OUT variable and thus takes place in the PIQ, a function (FC) can be used here.

11. The block '**Program Press[FC1]**' will be opened automatically. However, before the program can be written, the block's interface has to be declared.
When the interface is declared, the local variables known only in this block are specified.

The variables consist of two groups

- Block parameters that generate the interface of the block for the call in the program.

Type	Name	Function	Available in
Input parameters	Input	Parameters whose values the block reads	Functions, function blocks and some types of organization blocks
Output parameters	Output	Parameters whose values the block writes	Functions and function blocks
InOut parameters	InOut	Parameters whose value the block reads when it is called and after processing, writes again to the same parameter	Functions and function blocks

- Local data used for storing intermediate results.

Type	Name	Function	Available in
Temporary local data	Temp	Variables that are used for storing temporary intermediate results. Temporary data is retained for one cycle only.	Functions, function blocks and organization blocks
Static local data	Static	Variables that are used for storing static intermediate results in the instance data block. Static data is retained until it is written anew, even over several cycles.,	Function blocks

12. When declaring the local variables, in our example the following variables are needed.

Input:

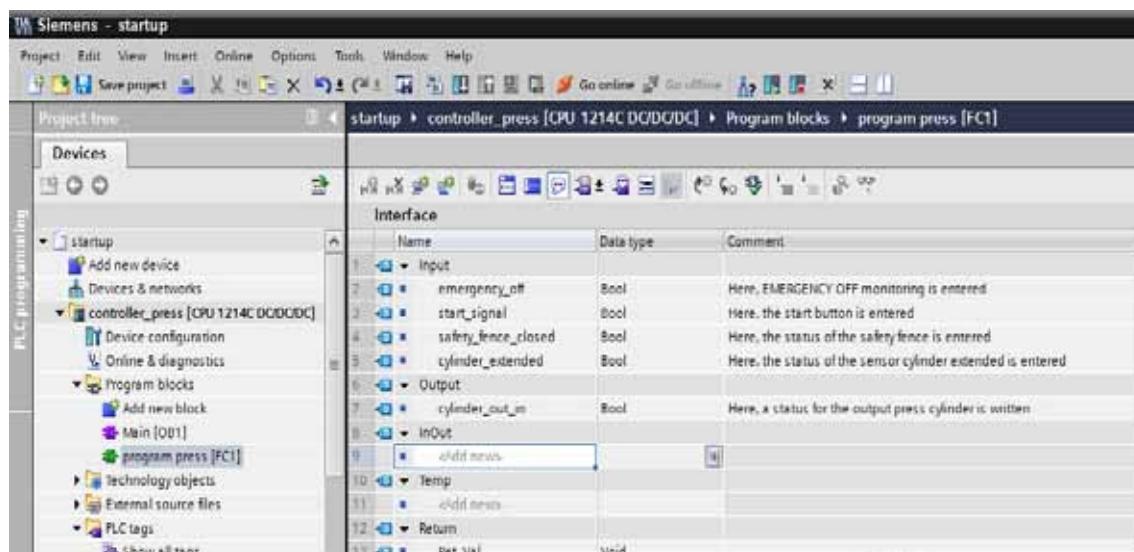
emergency_off Here, the EMERGENCY OFF function is entered
 start_signal Here, the start button is entered
 safety_fence_closed Here, the status of the safety fence is entered
 cylinder_extended Here, the status of the sensor Cylinder Extended is entered

Output:

cylinder_out_in Here, a status for the output Press Cylinder is written

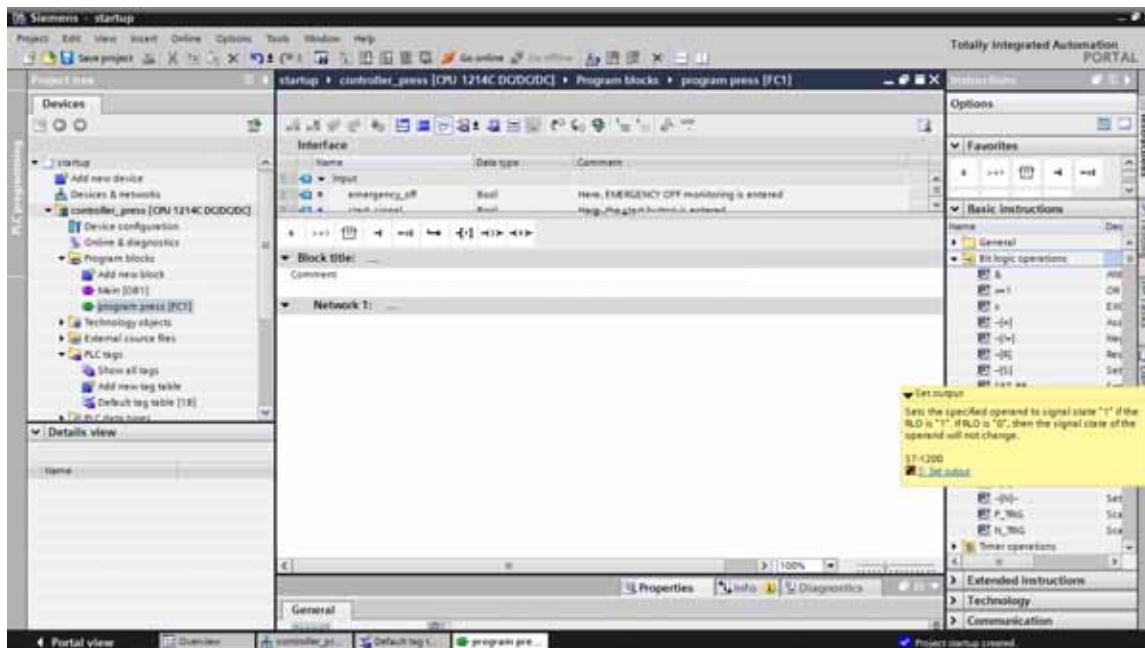
All variables in this case are of the type 'Bool'; which means variables that only can have the status '0' (false) or '1' (true).

To make it easier to follow them, all local variables should be written in the lower case and also be provided with a sufficient comment.



13. After having declared the local variables, we can now start programming. To provide a better overview, we program in networks. A new network can be inserted by clicking on the symbol  'Insert network'. Like the block itself, each network should be documented in the title line. If a longer text is needed for the description, the 'Comment' field can be used. (→ 

To generate our solution, we need an '**S**' to set the output. It is located under '**Instructions**' in the folder '**Bit combinations**'. If you point with the mouse to an object such as the **S**, for example, detailed information about this object will be displayed. (→ Instructions → Bit combination → S)



14. If you highlight an object and then press '**F1**' on your PC, you will be provided with online help about this object in a window to the right. (→ F1)

S: Set output



Description

You can use the "Set output" instruction to reset the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is set to "1". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

Executing the instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

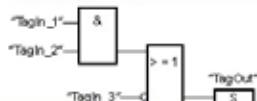
Parameters

The following table shows the parameters of the "Set output" instruction:

Parameters	Declaration	Data type	Memory area	Description
«Operand»	Output	BOOL	I, Q, M, D, L	Operand that is set when RLO = "1".

Example

The following example shows how the instruction works:

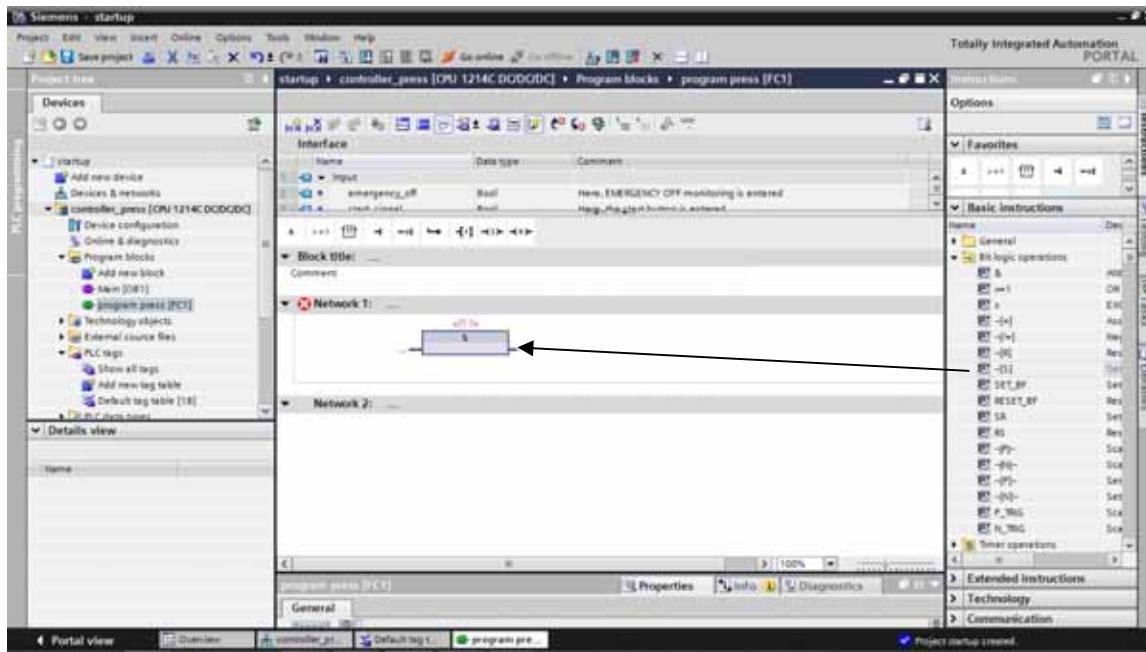


The operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

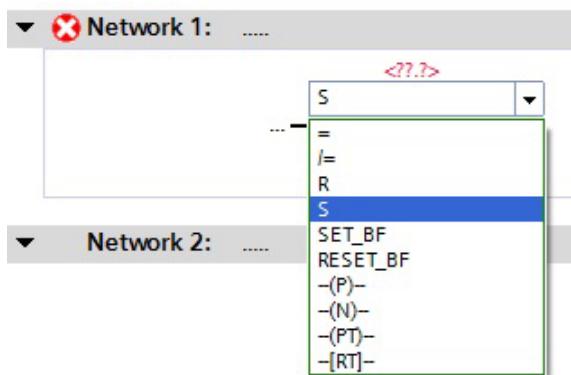
Note: Here, extensive information is provided in online help regarding the function and the wiring of the **S** (set output).

15. Now, with the mouse drag the **S** below the comment in Network 1. (→ S)

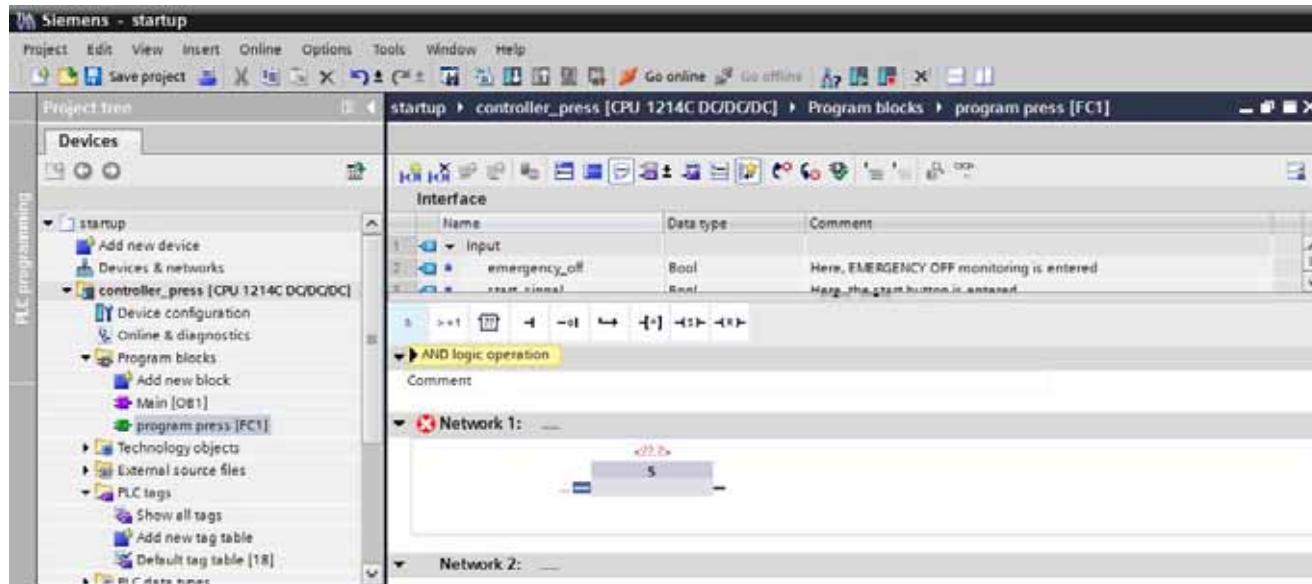


Note:

By clicking on the orange colored upper right corner, the function of the block can be changed subsequently with a menu.



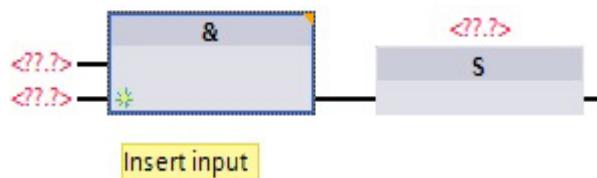
16. Next, to the left, highlight the input of the **S** block and double click on  AND in the favorites
 (→ left input → Favorites →  AND)



Note:

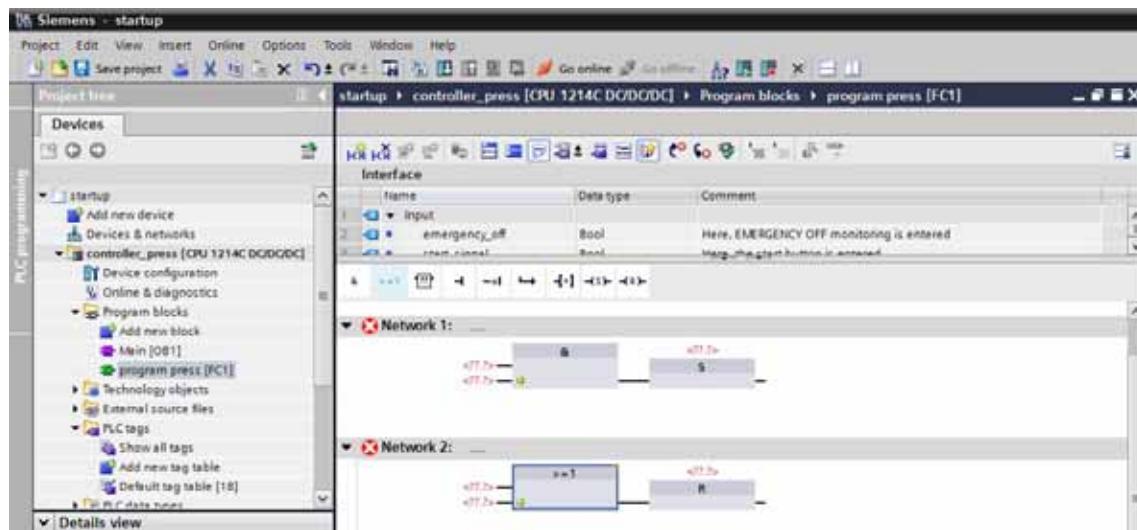
By clicking on the yellow star on the lower left, more inputs can be added.

By clicking on the orange colored upper right corner, the function of the block can be changed here also with a menu.

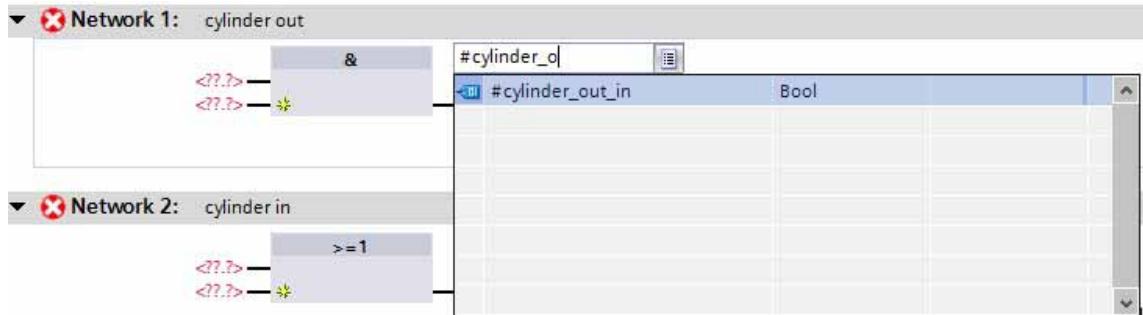


17. Exactly in the same way we place in **Network 2** the  OR at the **R** for Reset output

(→ R →  OR)



18. Now, we enter the local variables. It suffices to enter the first letter of the local variables in the fields at the commands. Then we can select the desired variable from a list. Local variables are always identified with the symbol '#' preceding the name ($\rightarrow \#cylinder_in_out$)



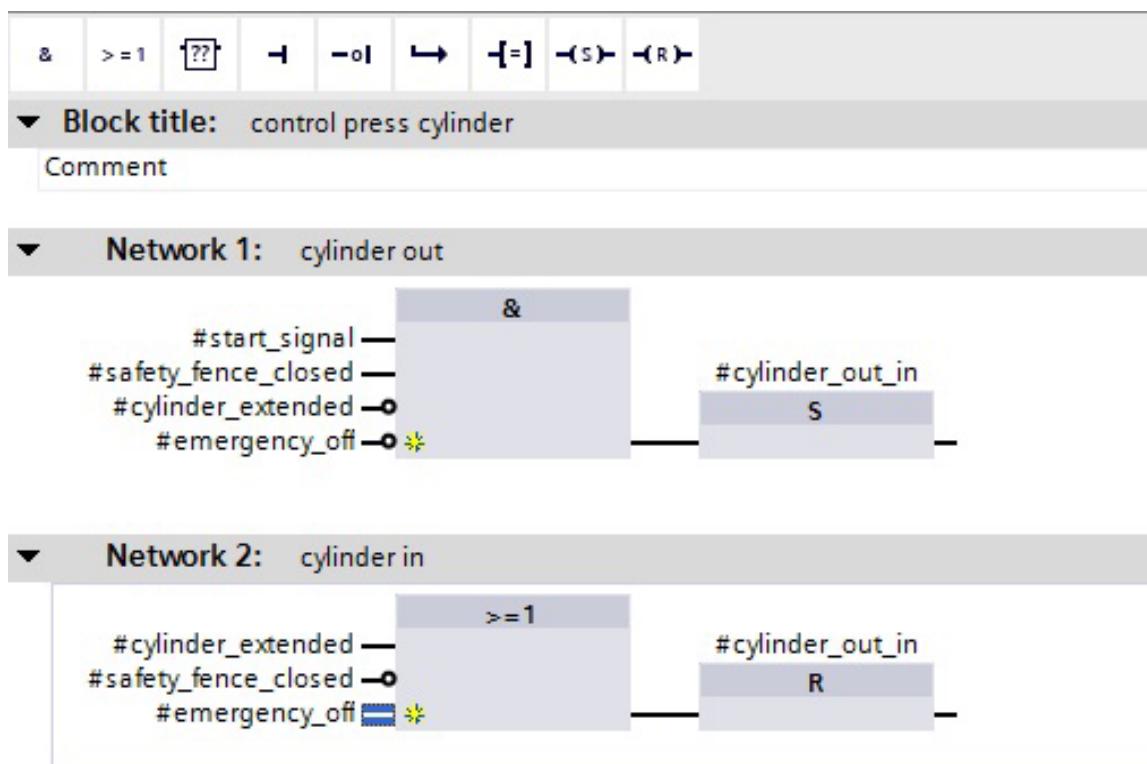
The local variables can also be dragged from the interface.

	Name	Data type	Comment
1	Input		
2	emergency_off	Bool	Here, EMERGENCY OFF monitoring is entered
3	start_signal	Bool	Here, the start button is entered
4	safety_fence_closed	Bool	Here, the status of the safety fence is entered
5	cylinder_extended	Bool	Here, the status of the sensor cylinder extended is entered
6	Output		
7	cylinder_out_in	Bool	Here, a status for the output press cylinder is written

19. Likewise, add the other local variables. At the AND or the OR, block, additional inputs have to be inserted. Click on the yellow star or highlight the lowest input with the right mouse key and select 'Add input' (→ Add input)

20. If an input is to be inverted, highlight the input and click on  or simply drag the symbol  negation from the 'Favorites' to the input.

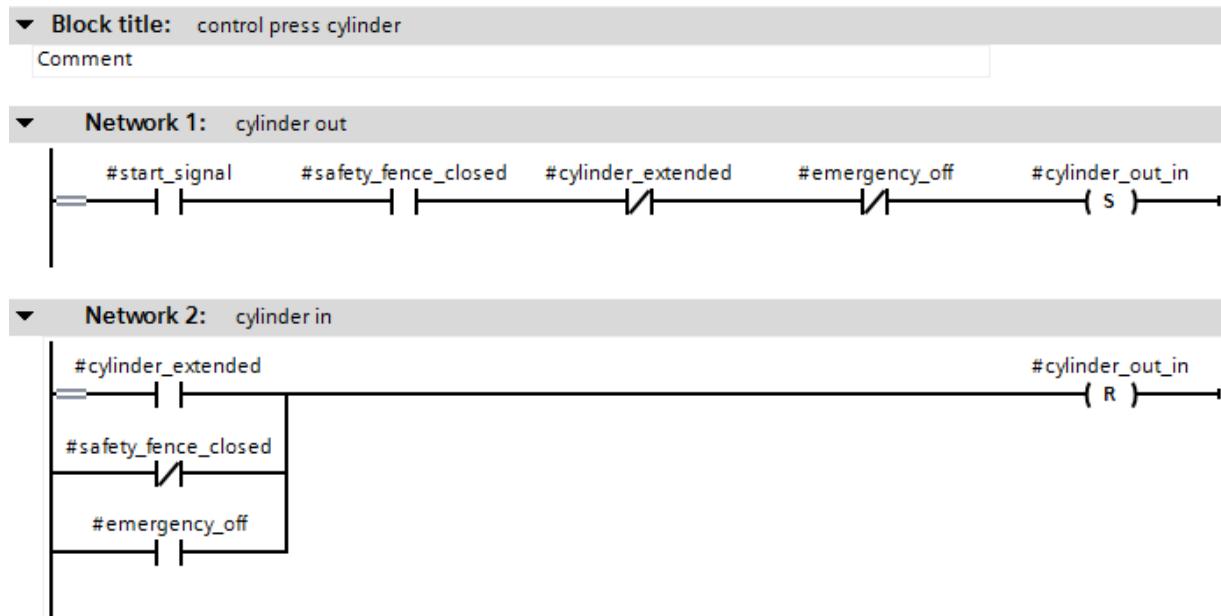
Program in function block diagram (FBD):



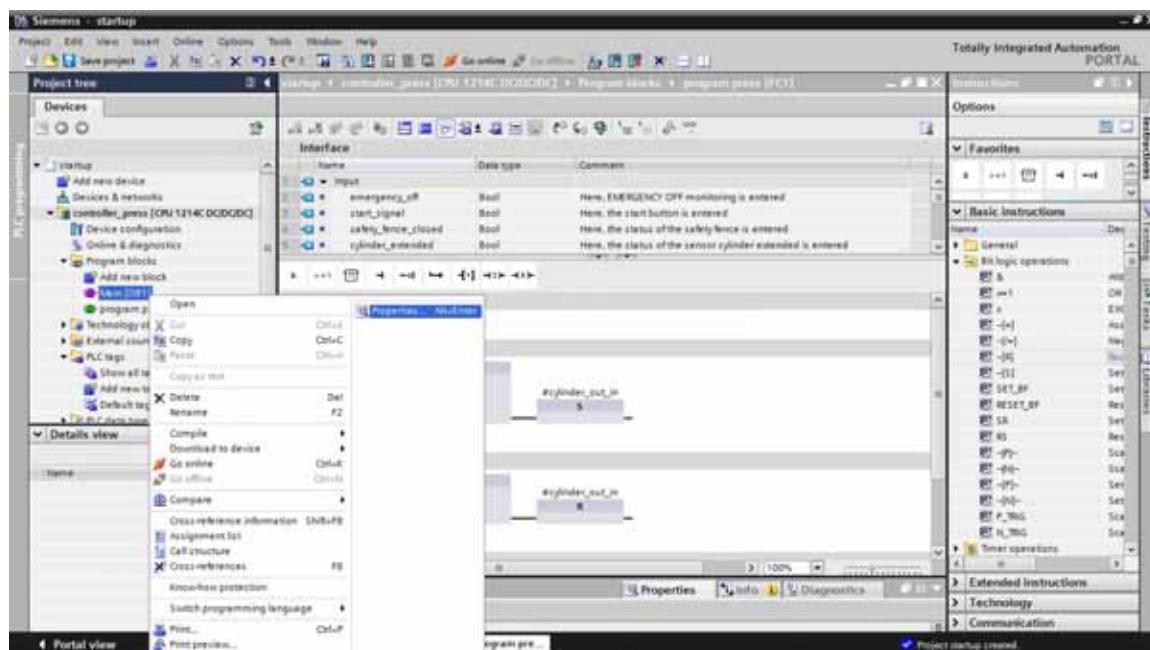
Function Description of the FC Block

The cylinder is to be extended when the start command is given and the safety fence is closed **and** the cylinder is **not** extended **and** emergency off is **not** pending.

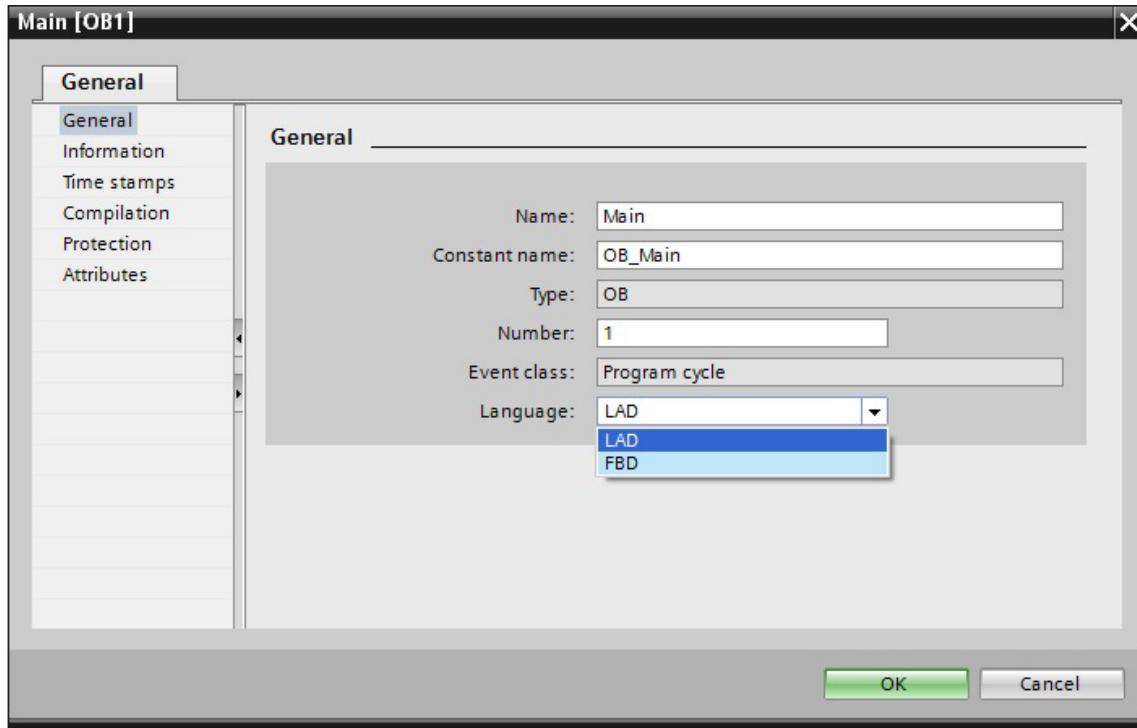
The cylinder is to be retracted when the cylinder is extended **or** when the safety fence is **no** longer closed **or** when emergency off is pending.

Program in ladder diagram (LAD):

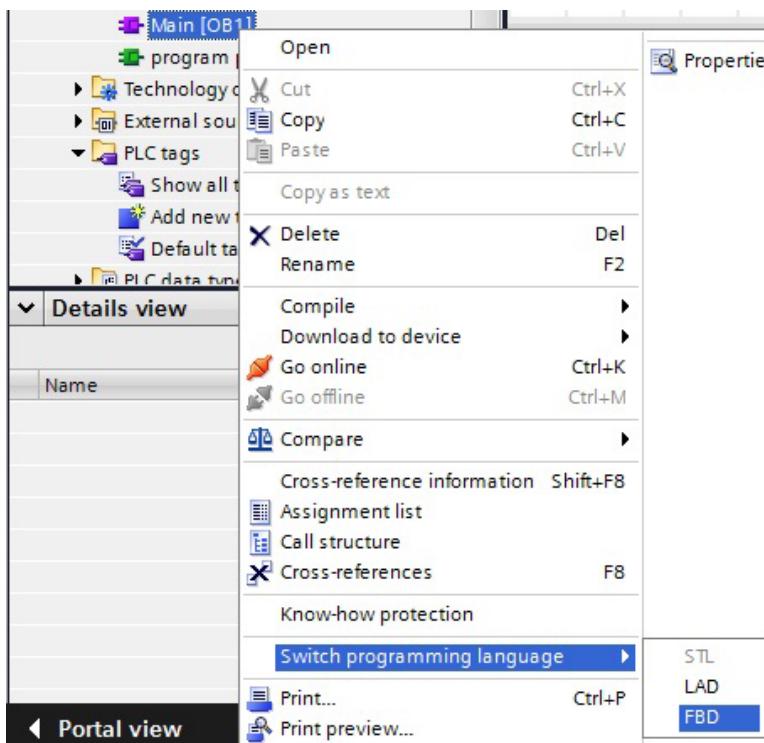
21. Next, the '**Properties**' of the cyclically processed block '**Main[OB1]**' are selected. Block properties can be modified (→ Properties → Main[OB1])



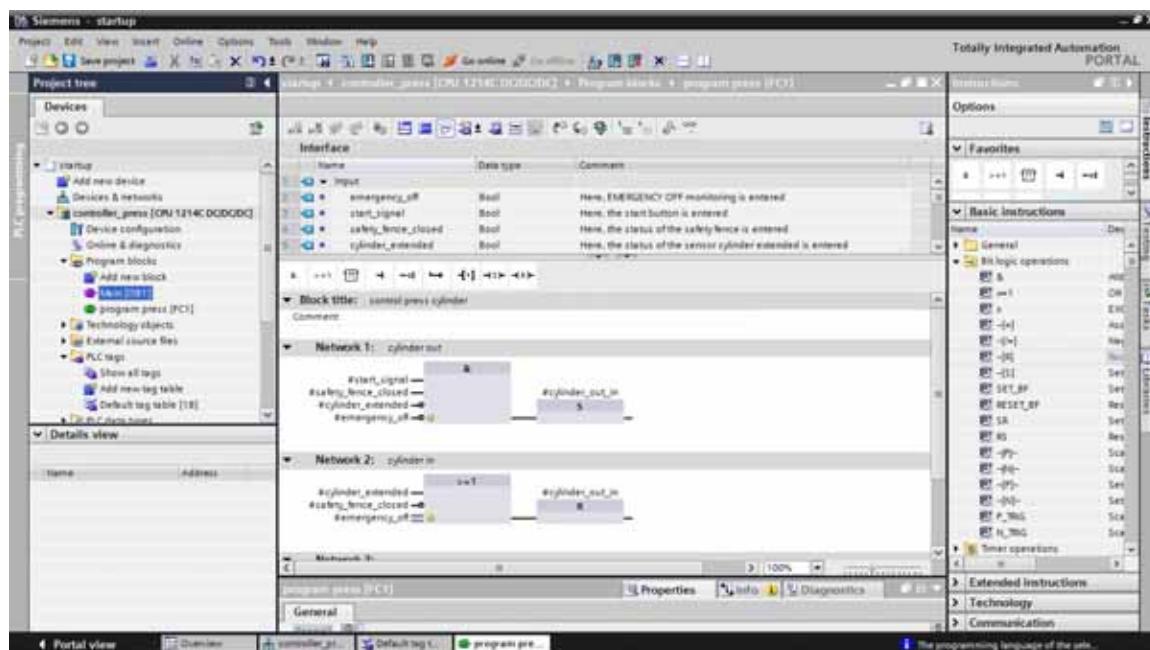
22. In the properties, select the programming-'**Language**' function block diagram '**FBD**'. (→ FBD → OK)



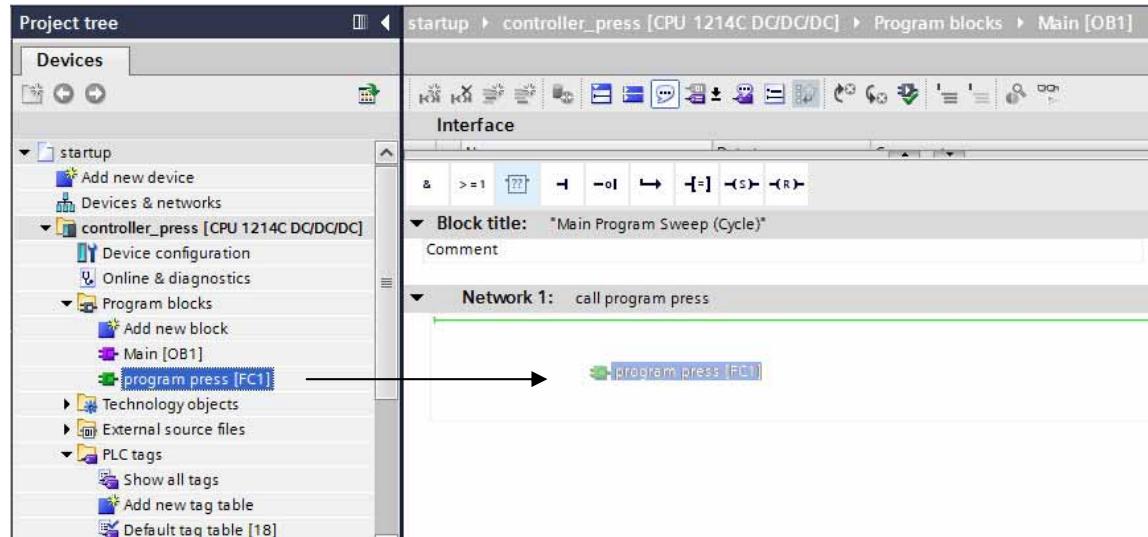
It is also possible to **Switch programming language**.



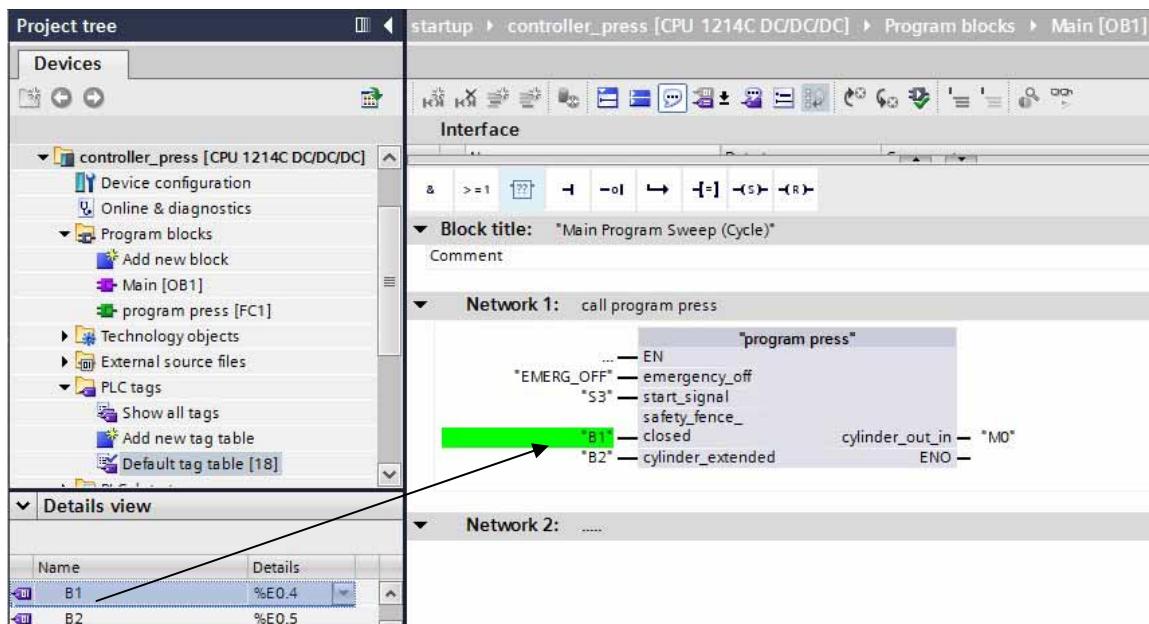
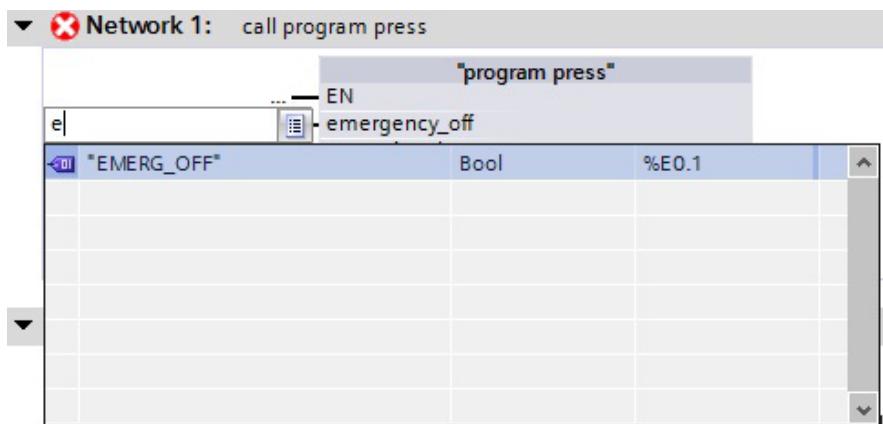
23. As mentioned previously, the block "Program Press" has to be called from the program block Main[OB1]. Otherwise, the block would not be processed at all. Open this block by double clicking on '**Main[OB1]**' (→ Main[OB1])



24. The block "Program Press" can then simply be dragged with Drag&Drop to Network 1 of the block Main[OB1]. Don't forget to document the networks also in block Main[OB1]. (→ Program Press)



25. Next, the interface parameters of the block "Program Press" have to be connected to global PLC variables. It is sufficient to enter the first letter and select from the list the desired operand (→ "EMERGENCY OFF") or to select the tag table and drag the tags from the detail view to the block.

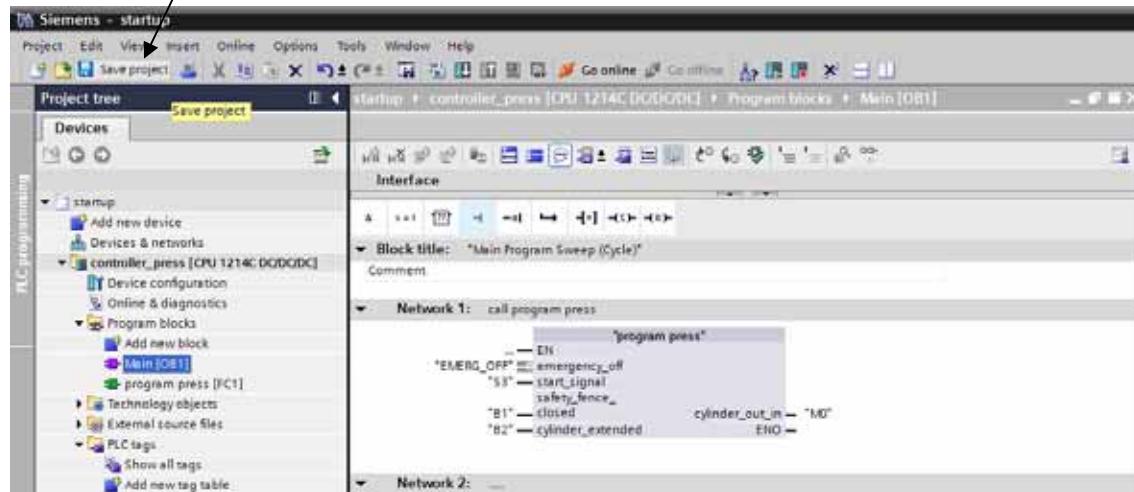


Note:

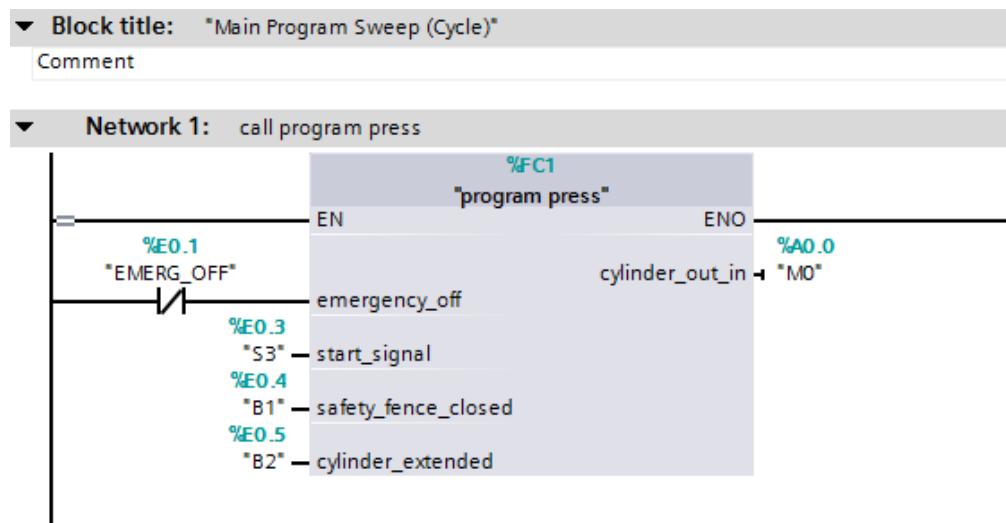
EMERGENCY STOP is a break contact (NC) and has to be negated for that reason; i.e. if EMERGENCY OFF is operated there is no longer a signal at input I0.1. When the blocks are wired with genuine addresses, the sensors (NO, NC) have to be noted.

26. Clicking on saves the project.
 (→

Program in function block diagram (FBD):

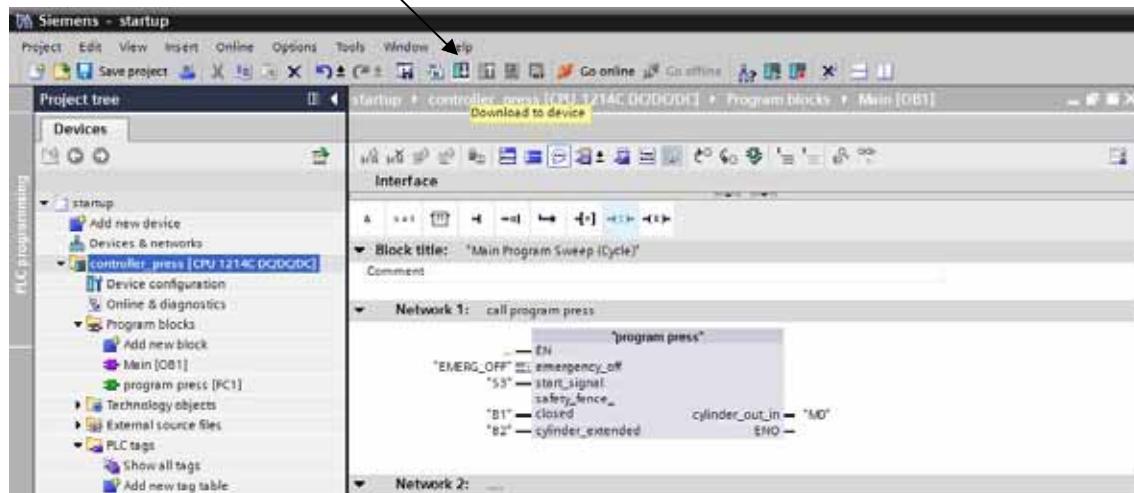


Program in ladder diagram (LAD):

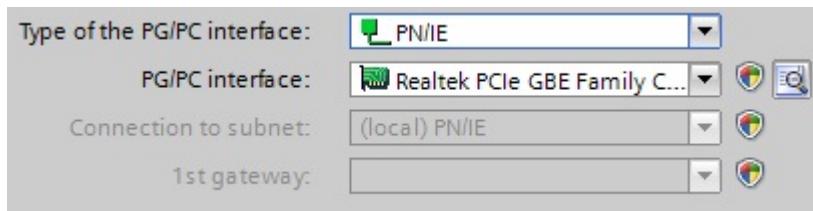


27. To load your entire program into the CPU, first highlight the folder '**Controller Press**' and then click on the symbol  **Load to device**. (→ Controller Press → 

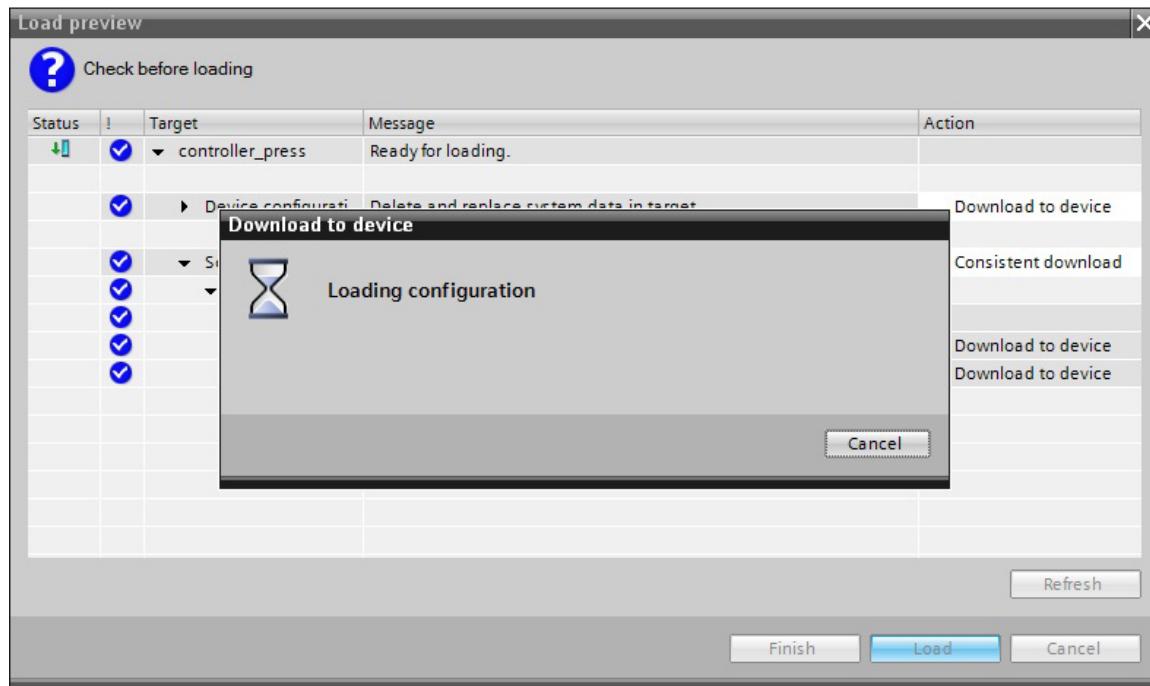
Program in function block diagram (FBD):



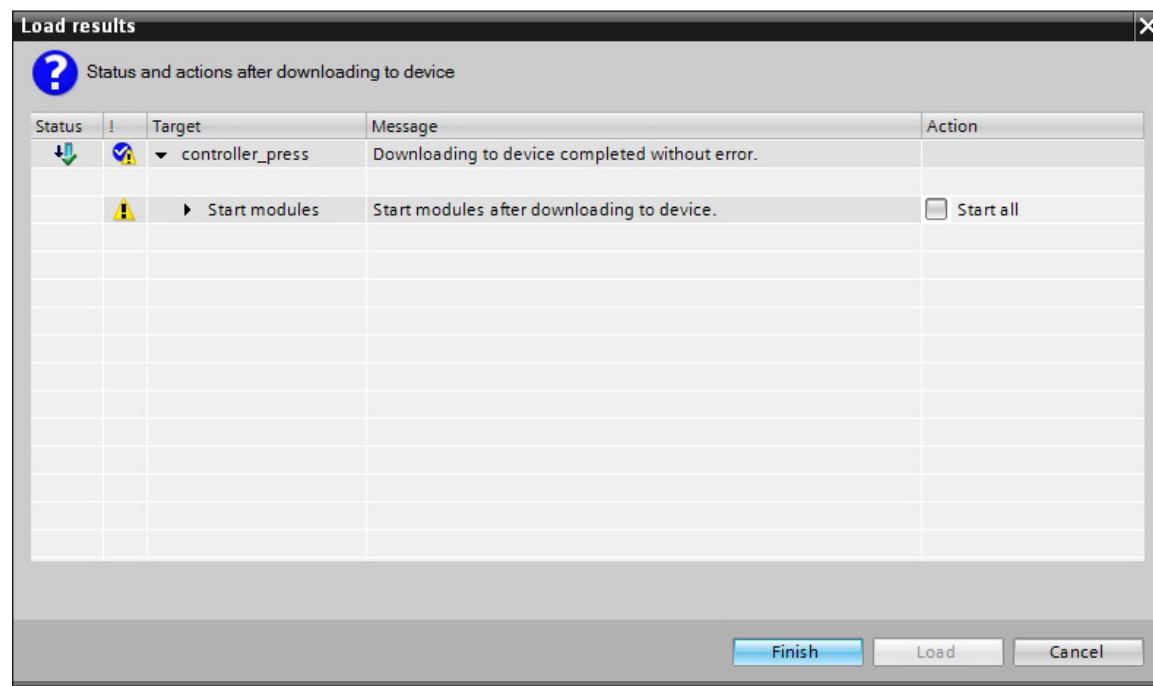
28. In the following dialog, select "**PN/IE**" as type of the PG/PC interface and then the previously set network card as PG/PC interface. After "**Update**" of the available devices, you should now see your CPU with the address 192.168.0.1 and be able to select it as target device. Then click ,  (→ Type of PG/PC interface: PN/IE → PG/PC interface: → Update → 



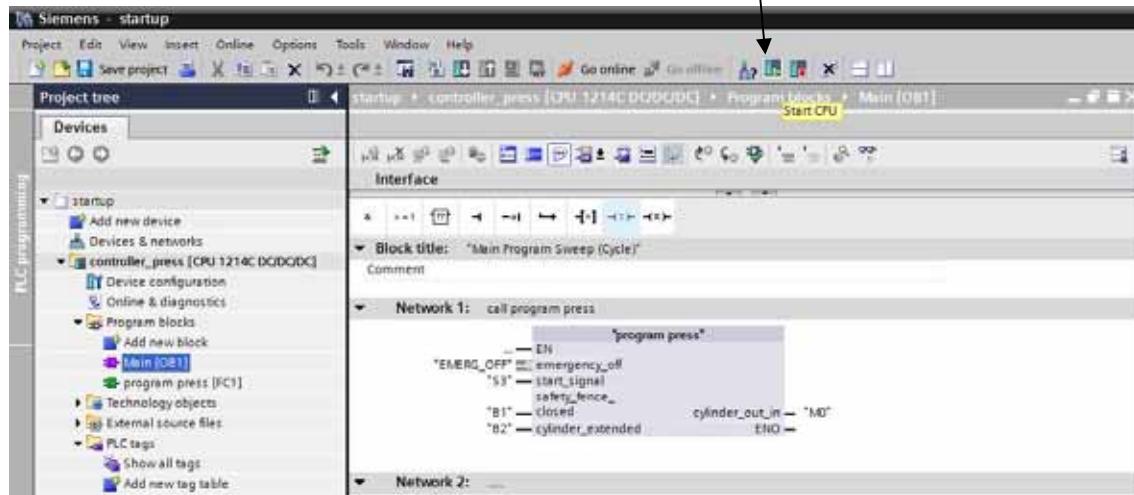
29. During loading, the status is displayed in a window.



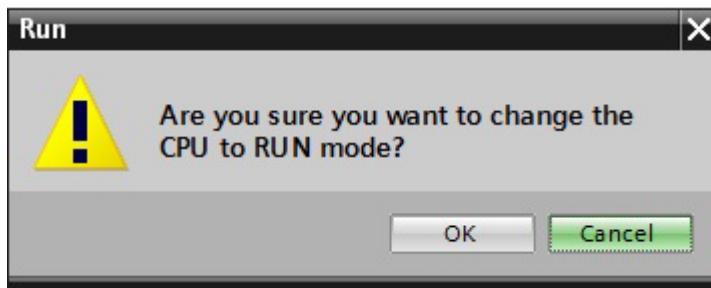
30. If loading was successful, it is displayed in a window. Now click on '**Finish**' (→ Finish)



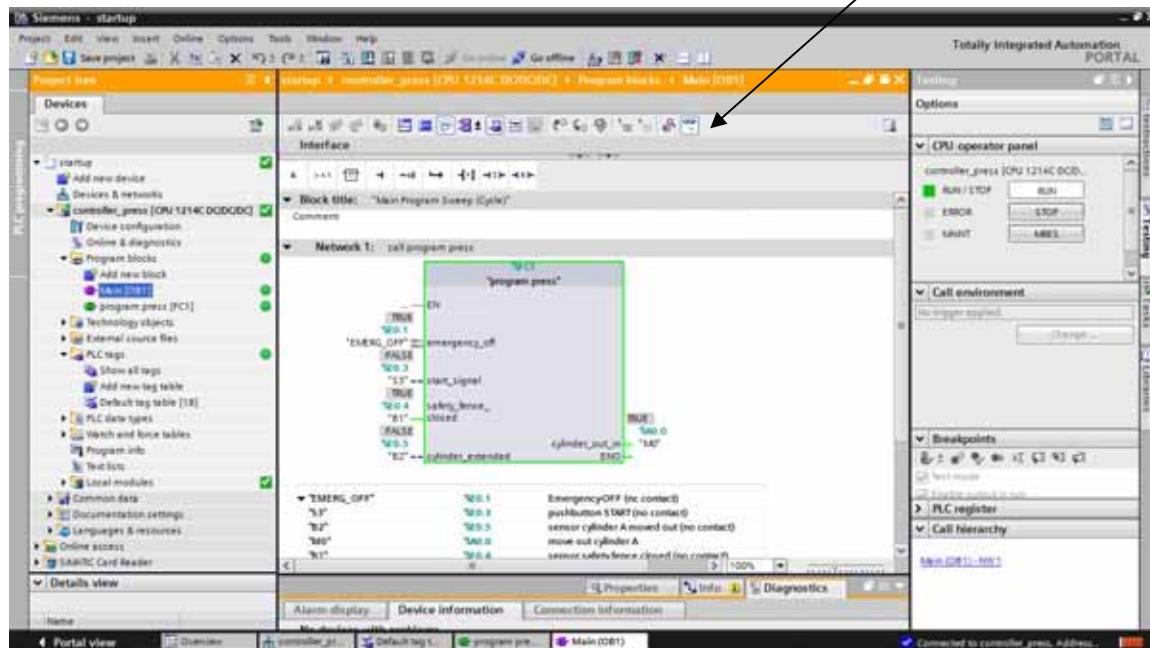
31. Now, start the CPU by clicking on the symbol (→)



32. With 'OK', confirm the question whether you actually want to start the CPU (→ OK)



33. By clicking on the symbol Monitoring on/off, you can monitor the status of the input and output variables during program testing at the block "Program Press". (→



Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Module 010-020



TIA Portal Module 010-010

Startup Programming of the
SIMATIC S7-1200

TIA Portal Module 010-020

Block Types for the SIMATIC S7-1200

TIA Portal Module 010-030

IEC Timers and IEC Counters at the
SIMATIC S7-1200

TIA Portal Module 010-040

Diagnosis and Error Search at the
SIMATIC S7-1200

TIA Portal Module 010-050

Analog Value Processing with
SIMATIC S7-1200

TIA Portal Module 010-060

Control Engineering with
SIMATIC S7-1200

TIA Portal Module 010-070

Communication between two
SIMATIC S7-1200

TIA Portal Module 010-080

HMI Panel KTP600 with SIMATIC
S7-1200

TIA Portal Module 010-090

Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

Matching SCE training packages for these documents

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
Order number: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
Order number 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
Order number 6ES7822-0AA01-4YE0

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

Continued Training

For regional Siemens SCE continued training, please contact your regional SCE contact person
siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding Usage

The training curriculum 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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by 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.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the documents being used commercially.

We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this document.

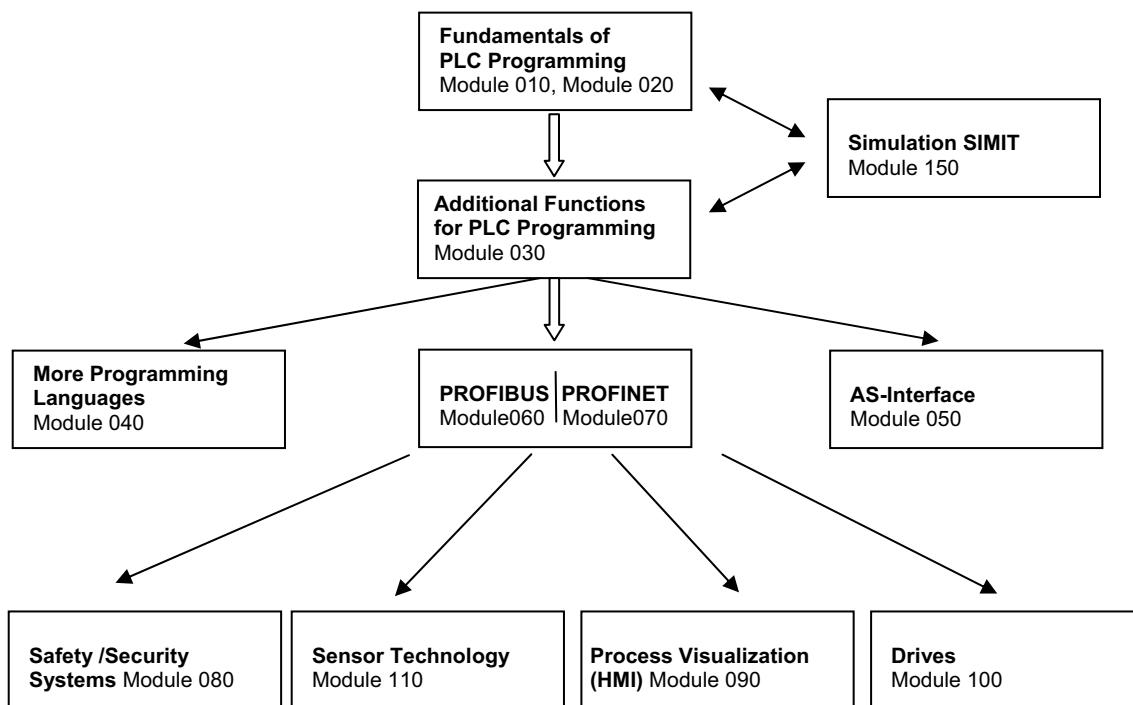
PAGE**2**

Contents

	PAGE
1. Preface	4
2. Notes on Programming the SIMATIC S7-1200	6
2.1 Automation System SIMATIC S7-1200	6
2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)	6
3. Block Types for the SIMATIC S7-1200	7
3.1 Linear Programming	7
3.2 Structured Programming	8
3.3 User Blocks for the SIMATIC S7-1200	9
3.3.1 Organization Blocks	10
3.3.2 Functions	11
3.3.3 Function Blocks	11
3.3.4 Data Blocks	12
4. Sample Task Function Block for Conveyor Control	13
5. Programming the Conveyor Control for the SIMATIC S7-1200	14

1. Preface

Regarding its content, module SCE_EN_010-020 is part of the training unit '**Basics of PLC Programming**' and represents a **fast entry point** for programming the SIMATIC S7-1200 with the TIA Portal.



Training Objective:

In module SCE_EN_010-020, the reader will become acquainted with the different blocks used for programming the SIMATIC S7-1200 with the programming tool TIA Portal. The module explains the various block types and shows in the steps listed below how to generate a program in a function block.

- Generating the function block
- Defining internal variables
- Programming with internal variables in the function block
- Calling and parameterizing the function block in OB1

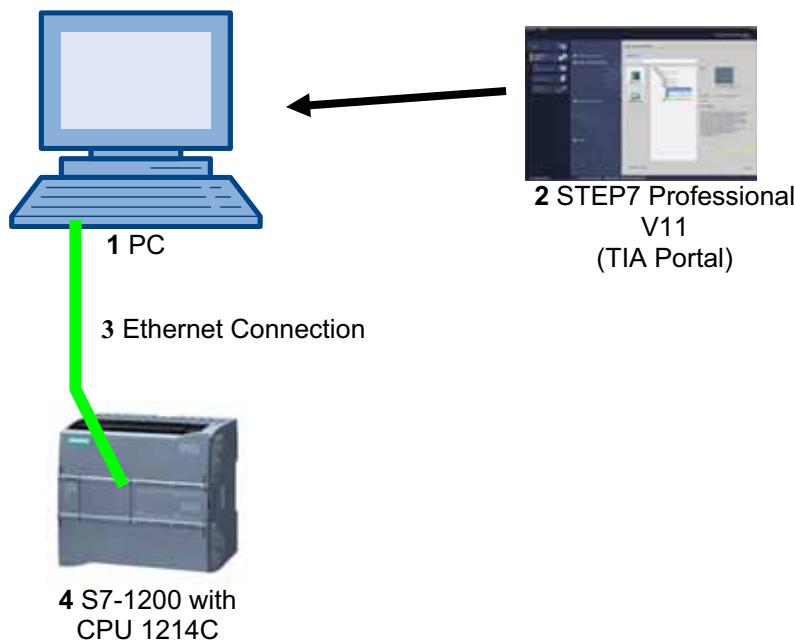
Prerequisites:

To successfully process this module, the following knowledge is assumed:

- How to operate Windows
- Basics of PLC programming with the TIA Portal (for example, Module 010-010 'Startup' Programming of the SIMATIC S7-1200 with TIA Portal V11)

Hardware and software required

- 1** PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB operating system Windows XP Professional SP3/Windows 7, Professional/Windows 7 Enterprise/Windows 7 Ultimate/Windows 2003 Server R2/Windows Server 2008 Premium SP1, Business SP1, Ultimate SP
- 2** Software STEP7 Professional V11 SP1 (Totally Integrated Automation (TIA) Portal V11)
- 3** Ethernet connection between PC and CPU 315F-2 PN/DP
- 4** PLC SIMATIC S7-1200; for example, CPU 1214C.
The inputs have to be brought out to a panel.



2. Notes on Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The SIMATIC S7-1200 automation system is a modular mini-control system for the lower and medium performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU, and input and output modules for digital and analog signals.

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

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process; the IO modules are polled in the S7 program by means of the input addresses (%I), and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7.

2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 (TIA Portal V11) is the programming tool for the automation systems

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating visual displays for the SIMATIC basic panels with integrated WinCC Basic
- With additional WinCC packages, visualization solutions for PCs and other panels can be generated

All functions are supported with detailed online help.

3. Block Types for the SIMATIC S7-1200

For the SIMATIC S7-1200, the program is written in so-called blocks.

As a matter of standard, the organization block Main[OB1] is already provided.

It represents the interface to the CPU's operating system, is called by it automatically, and processed cyclically.

If the control task is extensive, we subdivide the program into small program blocks that are arranged according to functions, and easy to follow.

These blocks in turn are called from the organization blocks. At the end of the block, we jump back to the organization block that performed the call, exactly to the line behind the call.

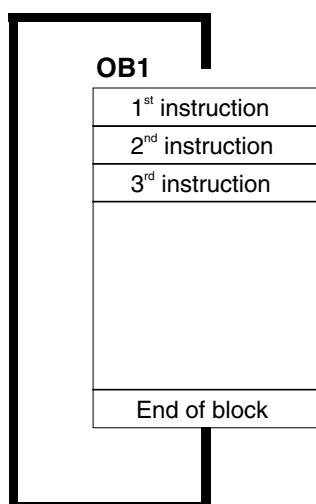
3.1 Linear Programming

For linear programming, the instructions are stored in a block and processed in the sequence in which they are stored in the program memory. When the end of the program (end of block) is reached, program processing begins again from the start.

This is called cyclical processing.

The time a device needs to process all instructions once is called cycle time.

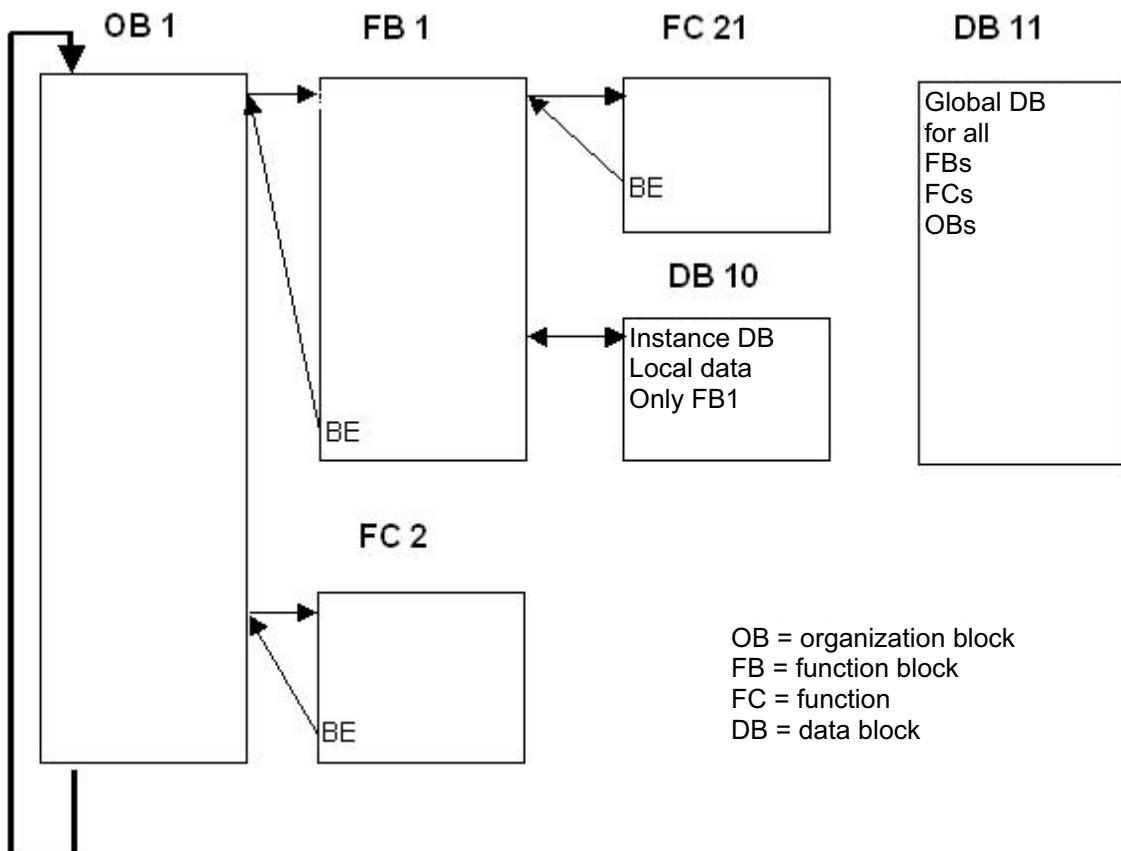
Linear program processing is usually used for simple control tasks that are not too extensive; it can be implemented in one OB.



3.2 Structured Programming

If the control task is extensive, we subdivide it into small program blocks arranged according to functions and that are easy to follow. The advantage: program parts can be tested individually and if they work, they can be merged into a total function.

The primary block has to call the program blocks. If the end of the block (BE) is recognized, the program continues to be processed in the calling block behind the call.



3.3 User Blocks for the SIMATIC S7-1200

The following user blocks are provided for structured programming:

- OB (organization block):

OBs are called by the operating system cyclically and are the interface between the user program and the operating system. In this OB, the PLC's control unit is informed by means of block call commands as to which program blocks it has to process.

- FB (function block):

For each call (instance), the FB needs an assigned memory area. When an FB is called, a data block (DB) for example can be assigned to it as instance DB.

The data in this instance DB is then accessed by means of the variables of the FB.

Different memory areas have to be assigned to an FB if it is called several times.

Additional FBs and FCs can be called in a function block in turn.

- FC (function):

A FC does not have an assigned memory area. The local data of a function is lost after the function is processed.

Additional FBs and FCs can be called in a function in turn.

- DB (data block):

DBs are used to provide memory for the data variables. There are two types of data blocks: global DBs where all OBs, FBs and FCs can read the stored data or themselves can write data into the DB, and instance DBs that are assigned to a certain FB.

Note:

If during FC and FB programming only internal variables were used, they can be used multiple times in the form of standard blocks. They can then be called any number of times. However, the FBs have to be assigned a memory area, a so-called instance (for example, a DB), for each call.

3.3.1 Organization Blocks

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

They are called by the operating system, and control the following processes:

Startup behavior of the automation system

- Cyclical program processing

- Alarm-controlled program processing

- Error handling

You can program the organization blocks as desired, and thus determine the CPU's behavior.

You have various options for using organization blocks in your program:

- **Startup OB, Cycle OB, Timing Error OB and Diagnosis OB:**

You can simply insert and program these organization blocks in your project. You don't have to assign parameters to them, nor do you have to call them.

- **Process Alarm OB and Time Interrupt OB:**

These organization blocks have to be parameterized after you inserted them in your program. In addition, process alarm OBs can be assigned to an event at execution time using the instruction ATTACH, or separated again with DETACH.

- **Time Delay Interrupt OB:**

The time delay interrupt OB can be inserted in your project and programmed. In addition, you have to call it in the user program with the instruction SRT_DINT. Parameterization is not necessary.

Start Information

When some organization blocks are started, the operating system reads out information that can be evaluated in the user program.

This can be very helpful for diagnosing errors.

Whether and which information is read out is provided in the descriptions of the organization blocks.

3.3.2 Functions

A function contains a program that is executed when another code block calls the function.

Functions (FCs) are code blocks without memory. The data of the temporary variables is lost after the function is processed. Global data blocks can be used to store FC data.

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

- Returning function values to the calling block; for example, in the case of mathematical functions
- Executing technological functions; for example, individual controls with binary operations

Also, a function can be called several times at different locations within a program. This facilitates programming complicated repetitious functions.

3.3.3 Function Blocks

Function blocks contain subprograms that are always executed when a function block is called by another code block.

Function blocks are codes blocks that store their values in instance data blocks so that these values are available also after the block is processed.

Store your input, output and in/out parameters permanently in instance data blocks. They will still be available after the block is processed. For that reason, they are also called blocks with 'memory'.

Function blocks are used for tasks that can't be implemented with functions:

- Always when timers and counters are needed in the blocks (refer to module M3)
- Always when information has to be stored in the program; for example, when preselecting an operating mode with a button.

A function block can also be called multiple times at different locations within a program. This facilitates programming repetitious, complicated functions.

Instances of function blocks

The call of a function block is called an instance.

To each instance of a function block, a memory area is assigned that contains the data the function block uses for processing. This memory is provided by data blocks that the software generates automatically. It is also possible to provide memory for several instances in one data block as **multi-instance**.

3.3.4 Data Blocks

In contrast to code blocks, data blocks do not contain instructions but are used to store user data. That means, the data blocks contain variable data that the user program uses for processing.

Global data blocks store data that can be used by all other blocks.

The maximum size of data blocks varies, depending on the CPU. The structure of global data blocks can be specified as required.

Application examples are:

- Storing the information of a warehouse system. "Which product is located where"
- Storing recipes for certain products

Every function block, every function or every organization block can read data from a global data block, or write data into a global data block. This data is retained in the data block even when the data block is exited.

The call of a function block is referred to as an instance. To each call of a function block with parameter transfer, an **instance data block** is assigned that serves as data storage. In it, the actual parameters and the static data of the function blocks are stored.

The maximum size of the instance data blocks varies, depending on the CPU. The variables declared in the function block determine the structure of the instance data block.

A global data block and an instance data block can be open at the same time.

4. Sample Task Function Block for Conveyor Control

When blocks are to be generated that are working in any program like a "Black Box" as it were, they have to be programmed by using variables. In this case, the following rule applies: that in these blocks, no absolute-addressed inputs/outputs, flags etc. must be used. Within the block, only variables and constants are used.

In the example below, a function block is to be generated with a variable declaration containing a conveyor control that is dependent on the operating mode.

With button 'S1', the operating mode 'Manual' and with button 'S2' the operating mode 'Automatic' can be selected.

In the operating mode 'Manual', the motor is switched on as long as button 'S3' is operated, whereby button 'S4' must not be operated.

In the operating mode 'Automatic', the conveyor motor is switched on with button 'S3' and switched off with button 'S4' (break contact).

Assignment list:

Address	Symbol	Comment
%I 0.0	S1	Button operating mode Manual S1 NO
%I 0.1	S2	Button operating mode Automatic S2 NO
%I 0.2	S3	On button S3 NO
%I 0.3	S4	Off button S4 NC
%Q 0.2	M1	Conveyor motor M1

Note: The Off button S4 is a break contact here in order to ensure wire break safety. That means: if there is a wire break at this button, the system stops automatically. Otherwise, it could not stop if there were a wire break. For that reason, in control engineering all Stop buttons, Off buttons/switches have to be designed as break contacts.

5. Programming the Conveyor Control for the SIMATIC S7-1200

The project is managed and the components are programmed with the software '**Totally Integrated Automation Portal**'.

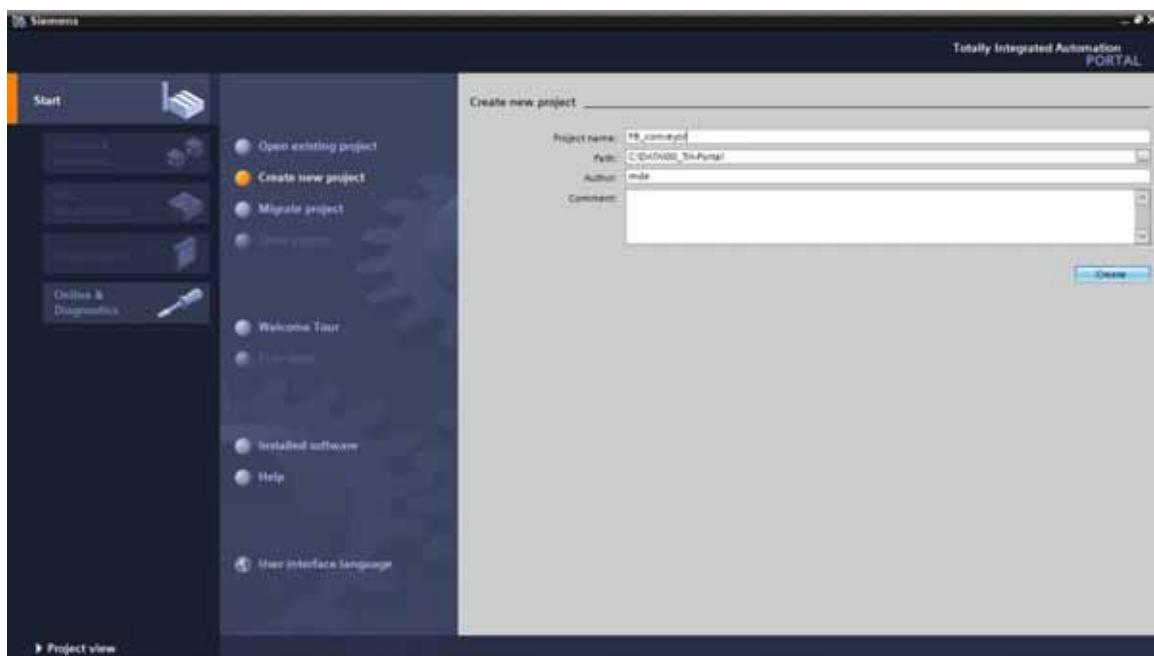
Here, under a uniform interface, the components such as controller, visual display and networking of the automation solution are set up, parameterized and programmed. Online tools are provided for error diagnosis

In the steps below, a project can be set up for the SIMATIC S7-1200 and the solution for a task can be programmed:

1. The central tool is the '**Totally Integrated Automation Portal**' that we call here with a double click (→ Totally Integrated Automation Portal V11)

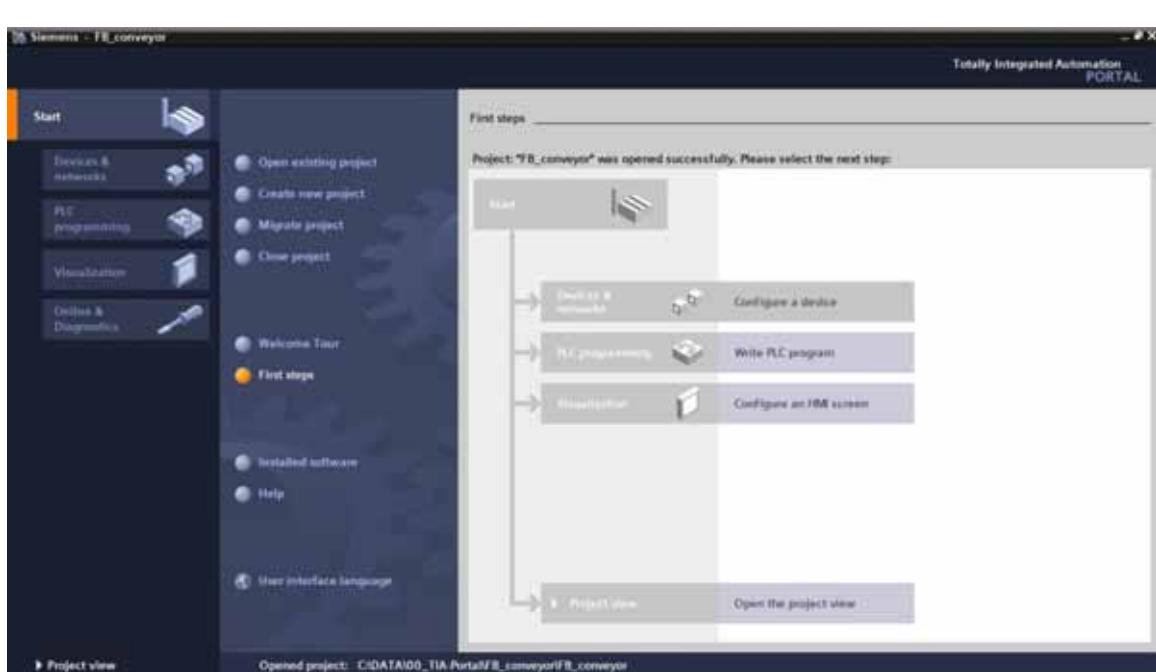


2. Programs for the SIMATIC S7-1200 are managed in projects. Such a project is now set up in the portal view → Create new project → FB_conveyor → Create)

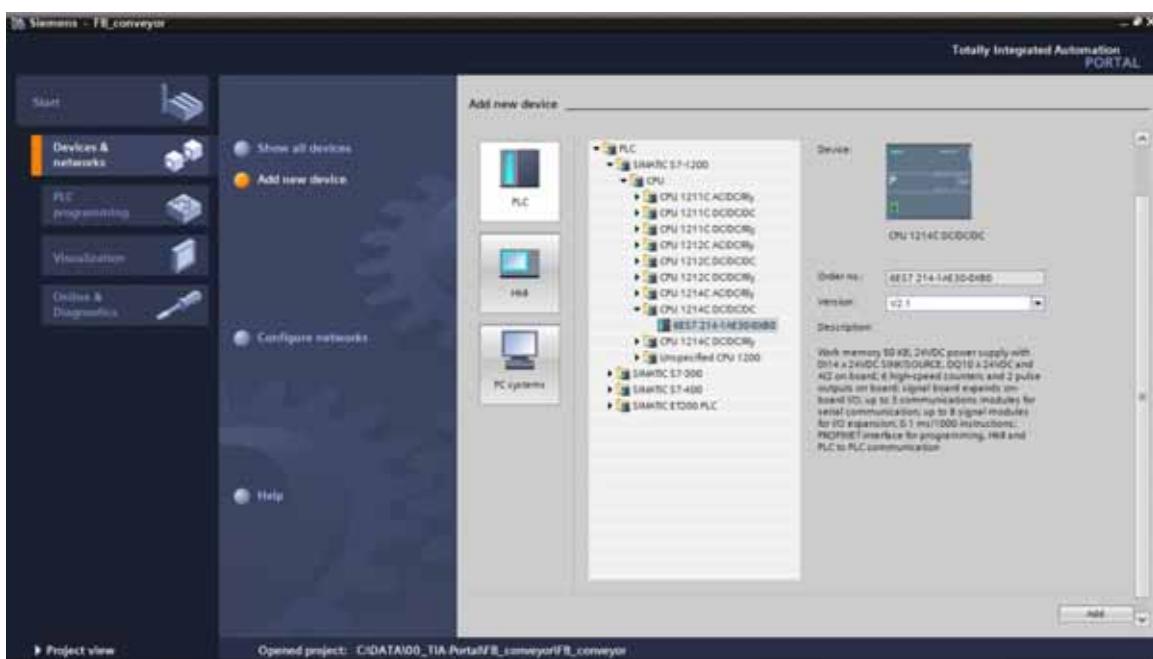


2

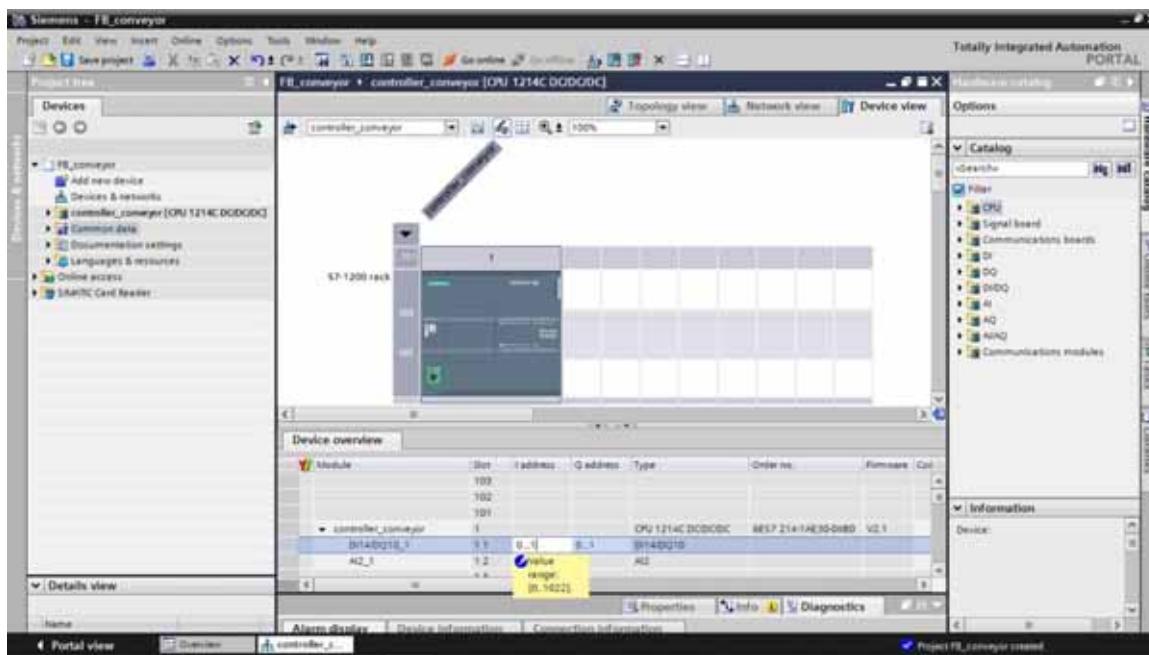
3. 'First Steps' are suggested regarding the configuration. First, we want to '**Configure a device**'.
(→ First Steps → Configure a device)



4. Next, we 'Add new device' with the 'Device name conveyor control'. To this end, we select from the catalog the '**CPU1214C**' with the matching order number. (→ Add new device → conveyor control → CPU1214C → 6ES7 → Add)

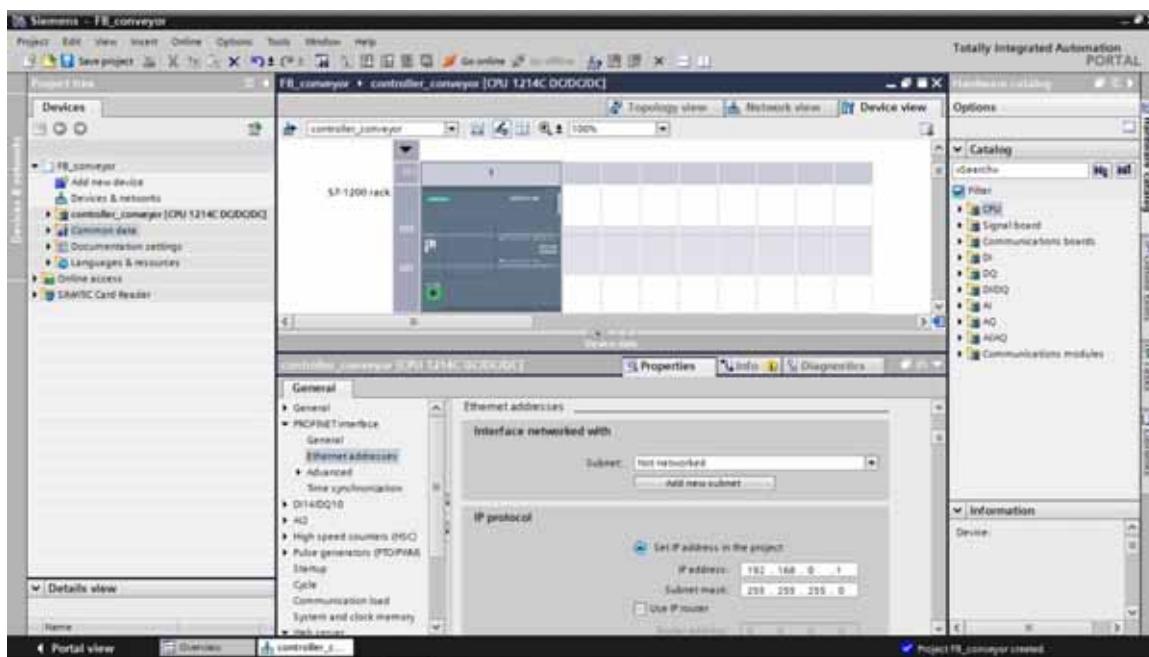


5. Now, the software automatically switches to the project view with the opened hardware configuration. Here, additional modules from the hardware catalog (to the right) can be added and in the '**Device overview**', the addresses of the inputs and outputs can be set. The integrated inputs of the CPU have the addresses %I0.0 to %I1.5 and the integrated outputs have the addresses %Q0.0 to %Q1.1 (→ Device overview → DI14/DO10 → 0...1)



6. So that the software later accesses the correct CPU, its IP address and the subnet mask have to be set.

(→ Properties → General → PROFINET interface → IP address: 192.168.0.1 → Subnet mask: 255.255.255.0)

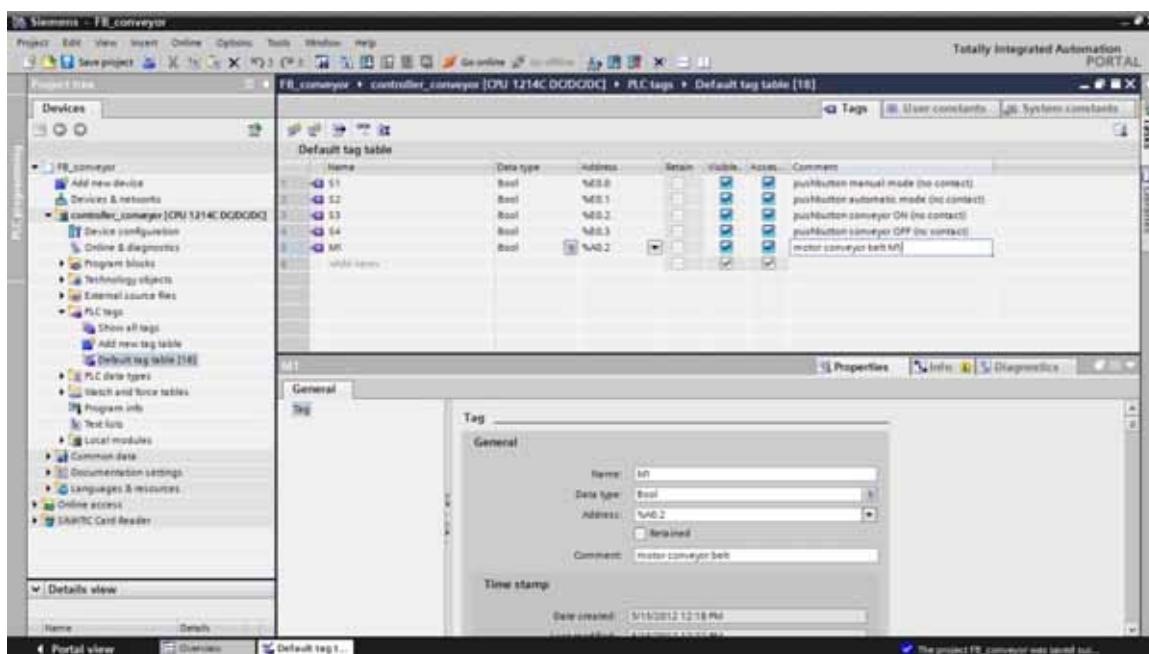


7. Since modern programming is not carried out with absolute addresses but with variables, the **global PLC tags** have to be specified here.

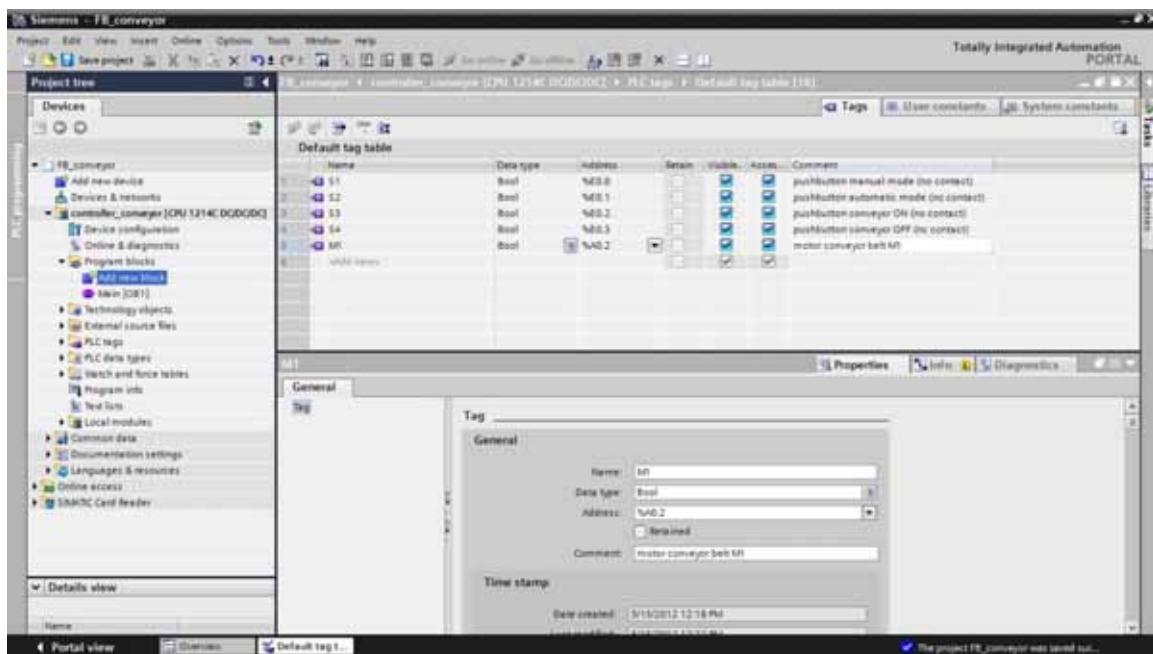
These global PLC variables are descriptive names with comments for those inputs and outputs that are used in the program. Later, during programming, this name is used to access the global PLC tags.

These global tags can be used in the entire program, in all blocks.

To this end, select in Project Navigation the '**controller_conveyorCPU1214C DC/DC/DC**' and then '**PLC tags**'. Open the table '**PLC tags**' with a double click and enter the names for the inputs and outputs as shown below (→ controller_conveyor[CPU1214C DC/DC/DC] → PLC tags → PLC tags)

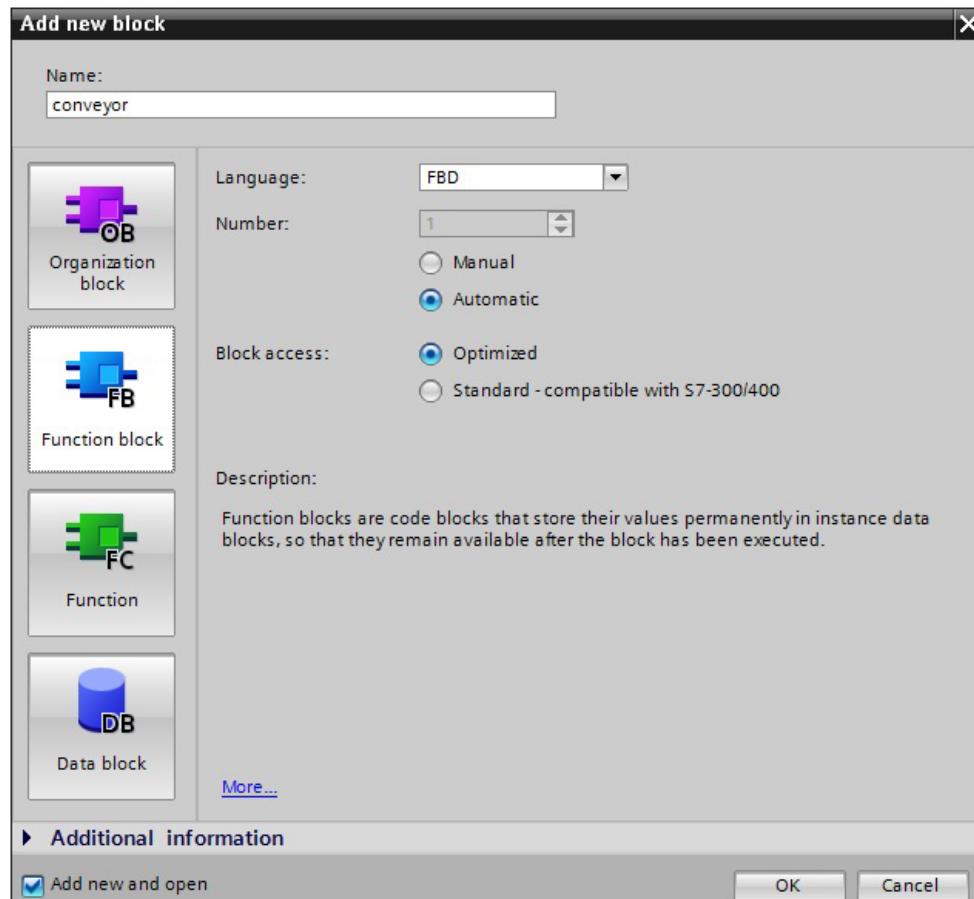


8. To generate the function block FB1, first select '**controller_conveyor[CPU1214C DC/DC/DC]**' in project navigation, and then '**Program blocks**'. Now, double click on '**Add new block**' (→ **controller_conveyor[CPU1214C DC/DC/DC]** → Program blocks → Add new block)



9. In the selection, select '**Function block (FB)**' and assign the name '**conveyor**'. As programming language, we specify function block diagram '**FBD**'. Enumeration is automatic. Since this FB1 is called later with the symbolic name anyhow, this number is no longer that important. Accept the input with '**OK**'. (→ Function block (FB1) → conveyor → FBD → OK)

2



10. The block '**conveyor[FB1]**' will be opened automatically. But before we can write the program, we have to declare the block's interface.

When the interface is declared, the local variables -known only in this block- are specified.

2

The variables are divided into two groups:

- Block parameters that generate the interface of the block for the call in the program.

Type	Name	Function	Available in
Input parameters	Input	Parameters whose values the block reads	Functions, function blocks and some types of organization blocks
Output parameters	Output	Parameters whose values the block writes	Functions and function blocks
In/out parameters	InOut	Parameters whose value the block reads when called, and after processing writes to the same parameter	Functions and function blocks

- Local data that is used for storing intermediate results

Type	Name	Function	Available in
Temporary local data	Temp	Variables that are used for storing temporary intermediate results. Temporary data is retained for one cycle only.	Functions, function blocks and organization blocks
Static local data	Static	Variables that are used for storing static intermediate results in instance data blocks. Static data is retained -also over several cycles-.until it is written anew.	Function blocks

11. To declare local variables, the following variables are needed for our example.

Input:

manual	Here, the signal for selecting the operating mode Manual is entered
automatic	Here, the signal for selecting the operating mode Automatic is entered
on	Here, the start signal is entered
off	Here, the stop signal is entered

Output:

motor	Here, the output signal for the output conveyor motor is written
-------	--

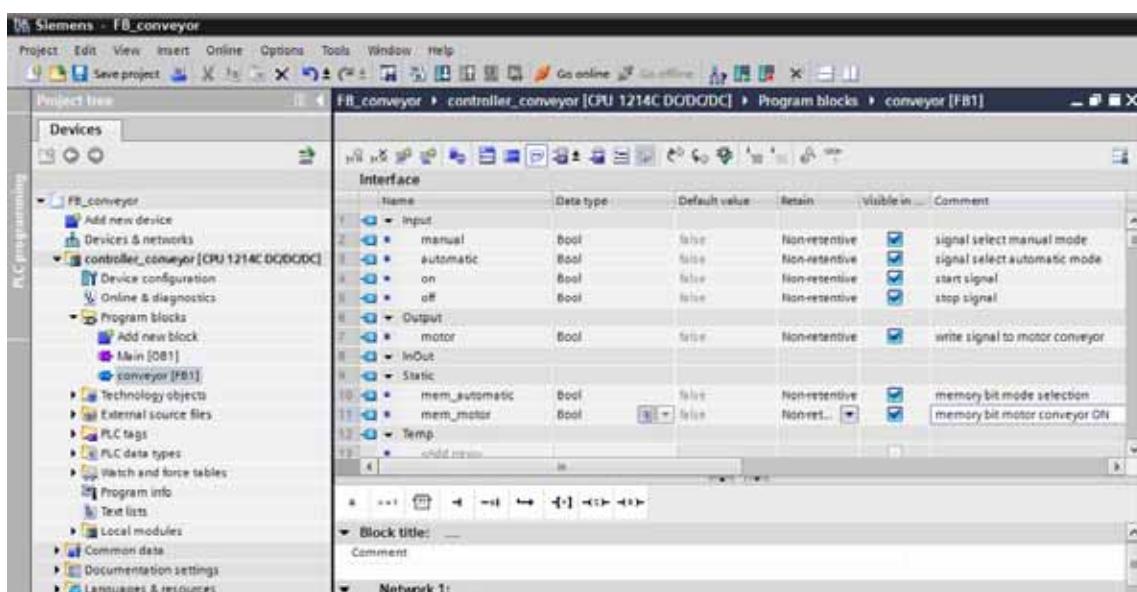
Static (exists only in the function blocks FB):

memory_automatic	Here, the preselected operating mode is stored
memory_motor	Here, we store when the motor was started in the Automatic mode

All variables are of the type 'Bool'; that means binary variables that only can have the status '0' (false) or '1' (true).

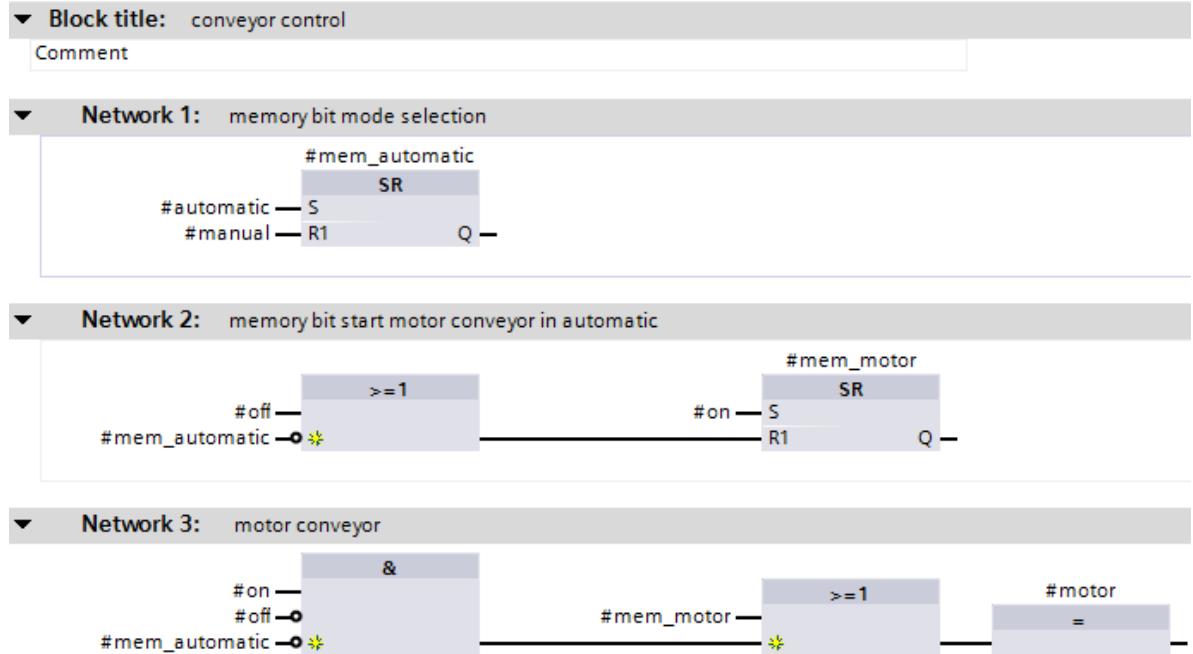
In this example, it is important to note that the status of the two variables 'memory_automatic' and 'memory_motor' has to be stored over a longer period of time. For that reason, the variable type '**Static**' has to be used here. This variable type in turn exists only in a function block FB.

For the sake of clarity, all local variables should also be provided with a sufficient comment.

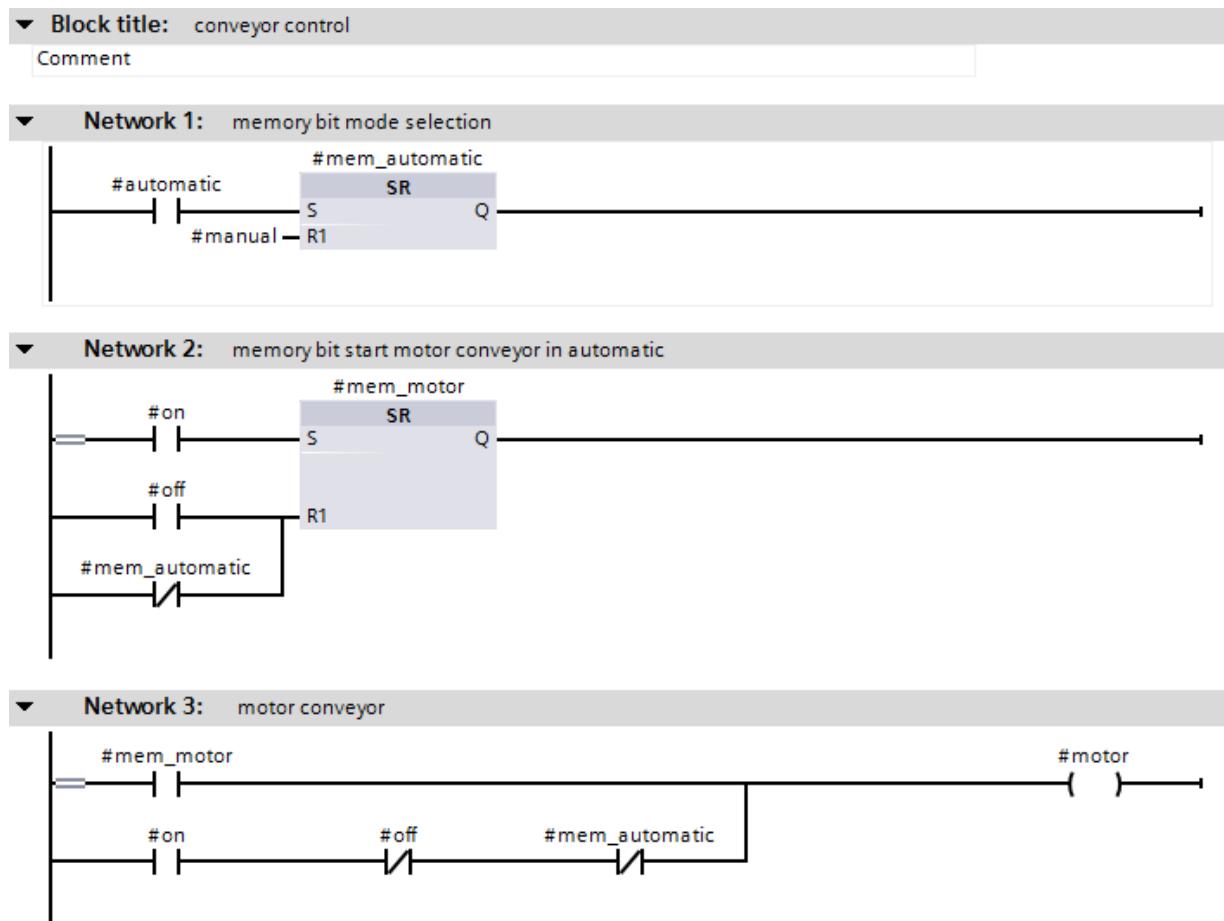


12. After the local variables have been declared, the program can now be entered by using the variable names (variables are identified with the symbol '#'). For the example in FBD, it could look like this:

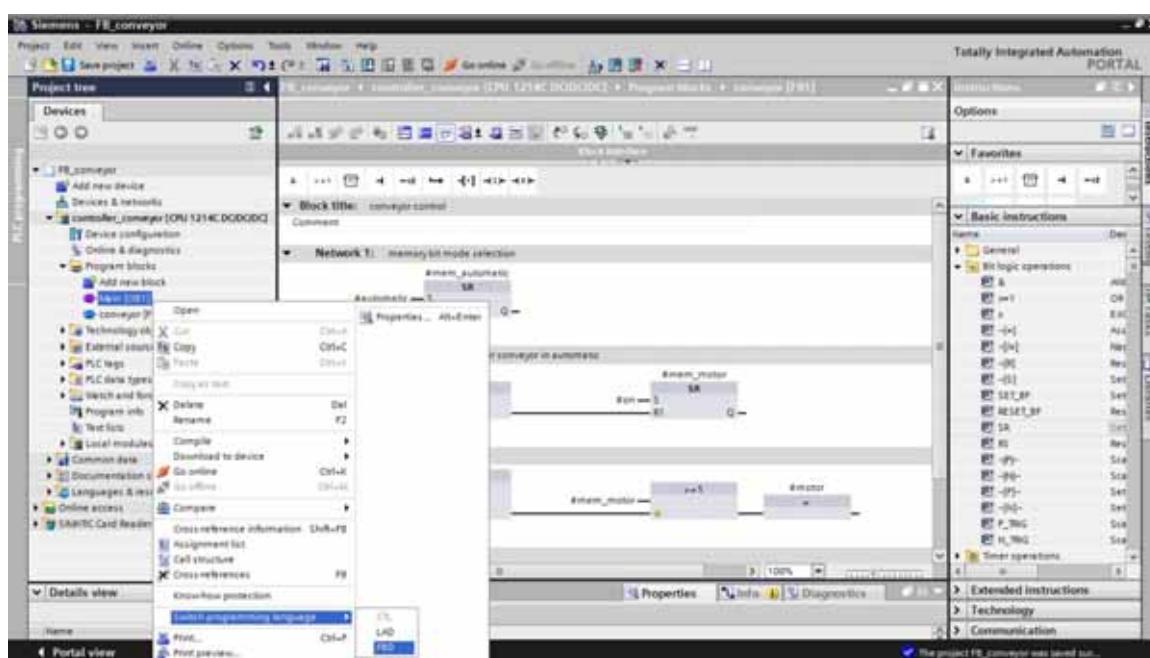
Program in function block diagram (FBD):



Program in ladder diagram (LAD):

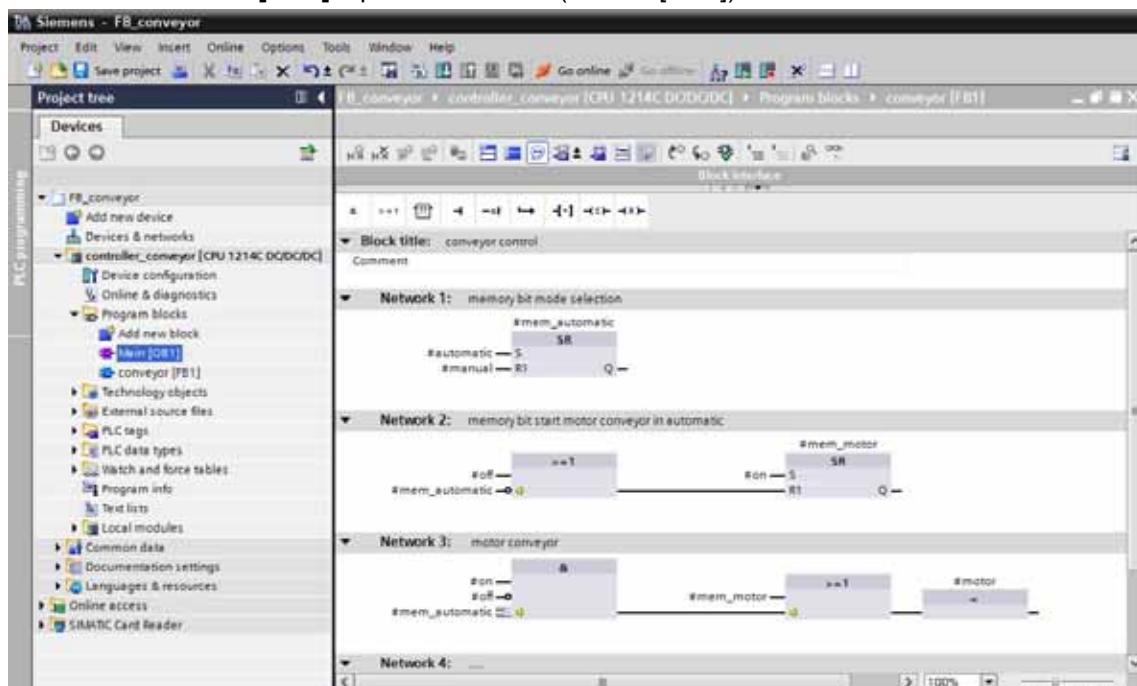


13. Next, right-click on the block 'Main[OB1]'.
 Then, under 'Switch programming language', select the function block diagram 'FBD'.

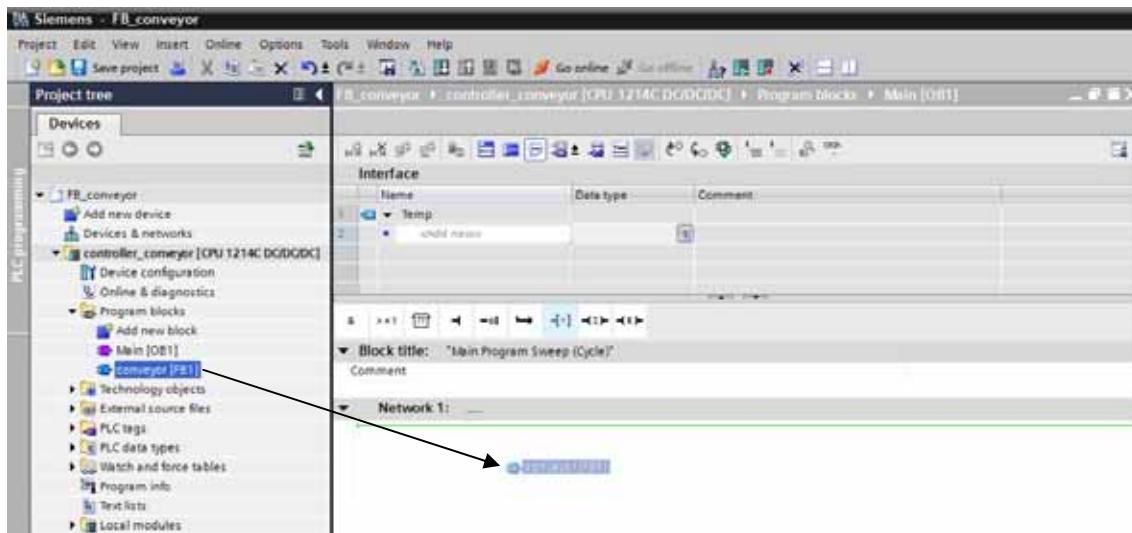


14. Now, the block "conveyor" has to be called from the program block Main[OB1]. Otherwise, the block would not be processed.

A double click on '**Main[OB1]**' opens this block. (→ Main[OB1])



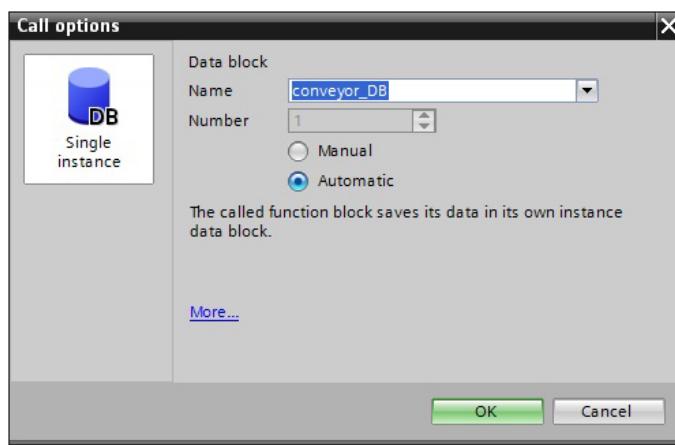
15. Now, you can drag the block "conveyor[FB1]" with Drag&Drop to Network 1 of the block Main[OB1]. (→ conveyor[FB1])



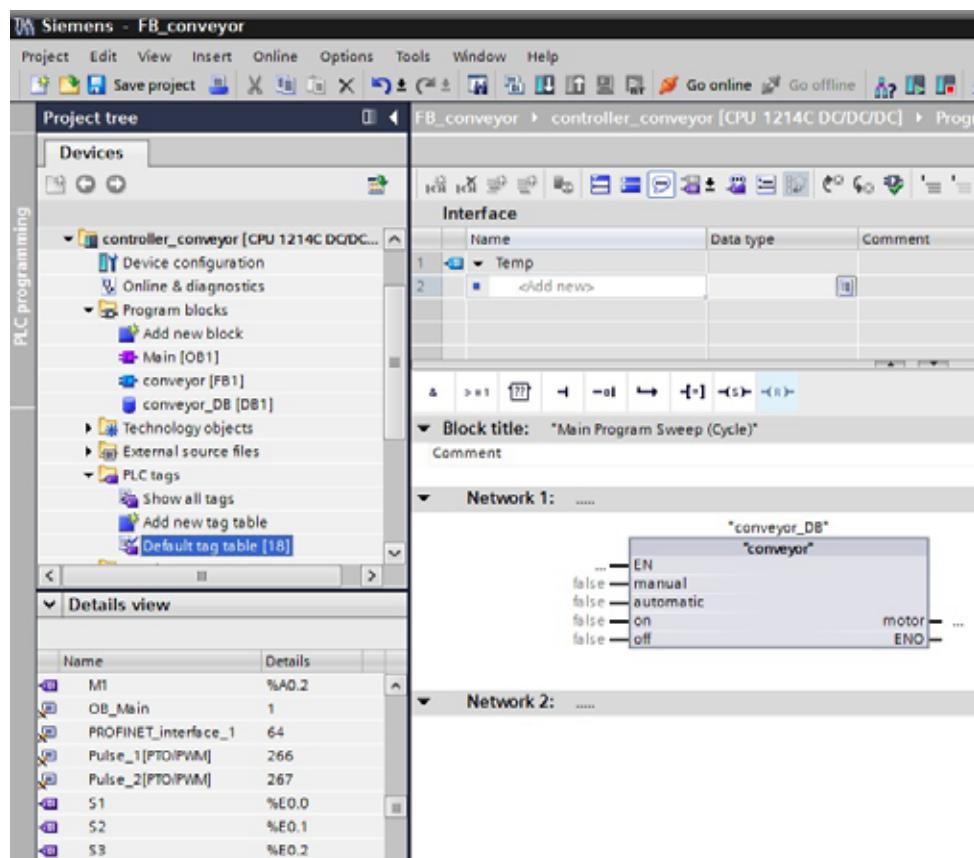
2

16. Since we are dealing with a function block, it has to be provided with memory. In SIMATIC S7-1200, data blocks are provided as memory. Such an assigned data block is called **Instance Data block**.

Here, it is to be specified and generated '**automatically**'. (→ Automatic → OK)



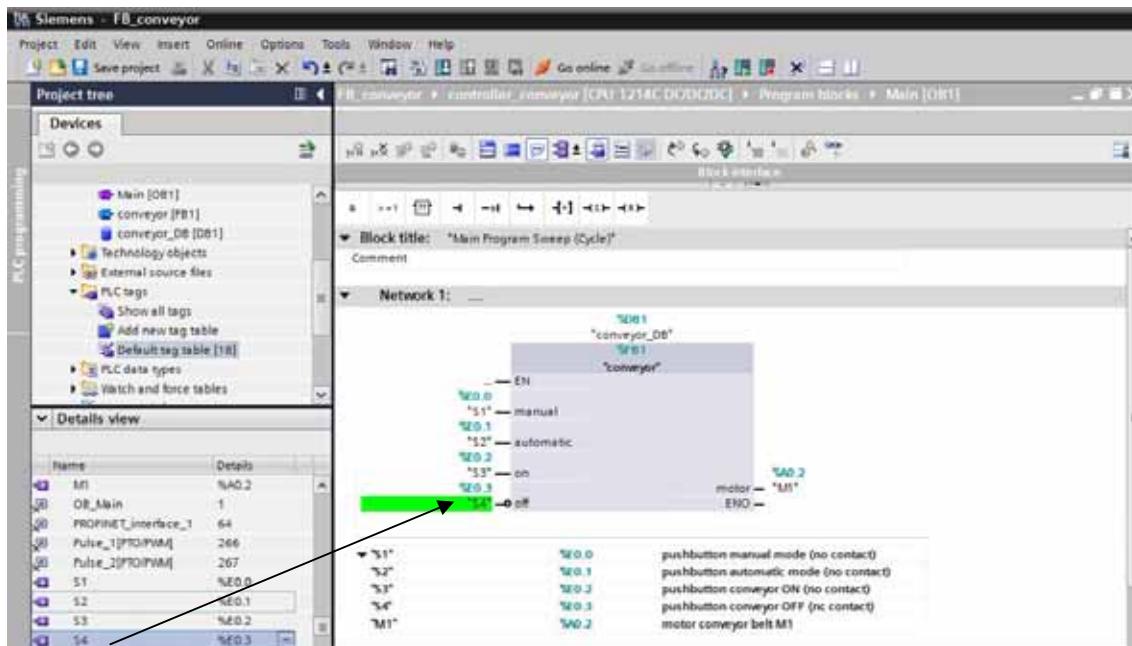
Highlight the default tag table.



17. In OB1, we now connect the input variables and the output variable with the PLC tags shown here.

To this end, just drag the PLC tags to the block variables.

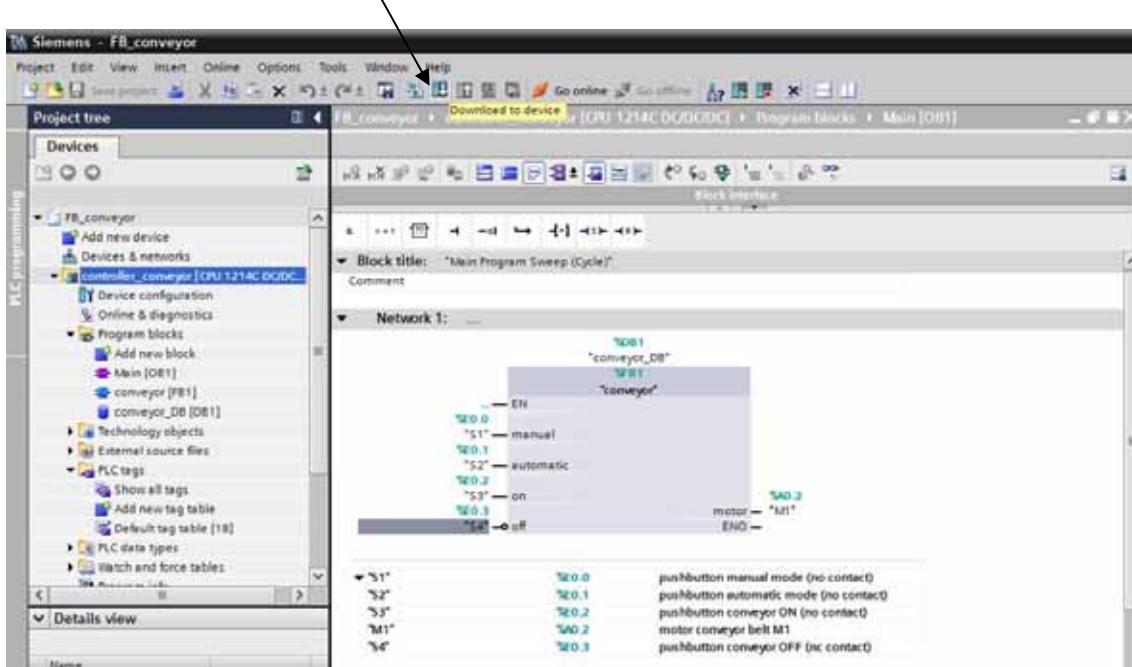
By clicking on Save project, the project is saved. (→ "S1" → "S2" → "S3" → "S4" → "M1" → Save project)



Important!

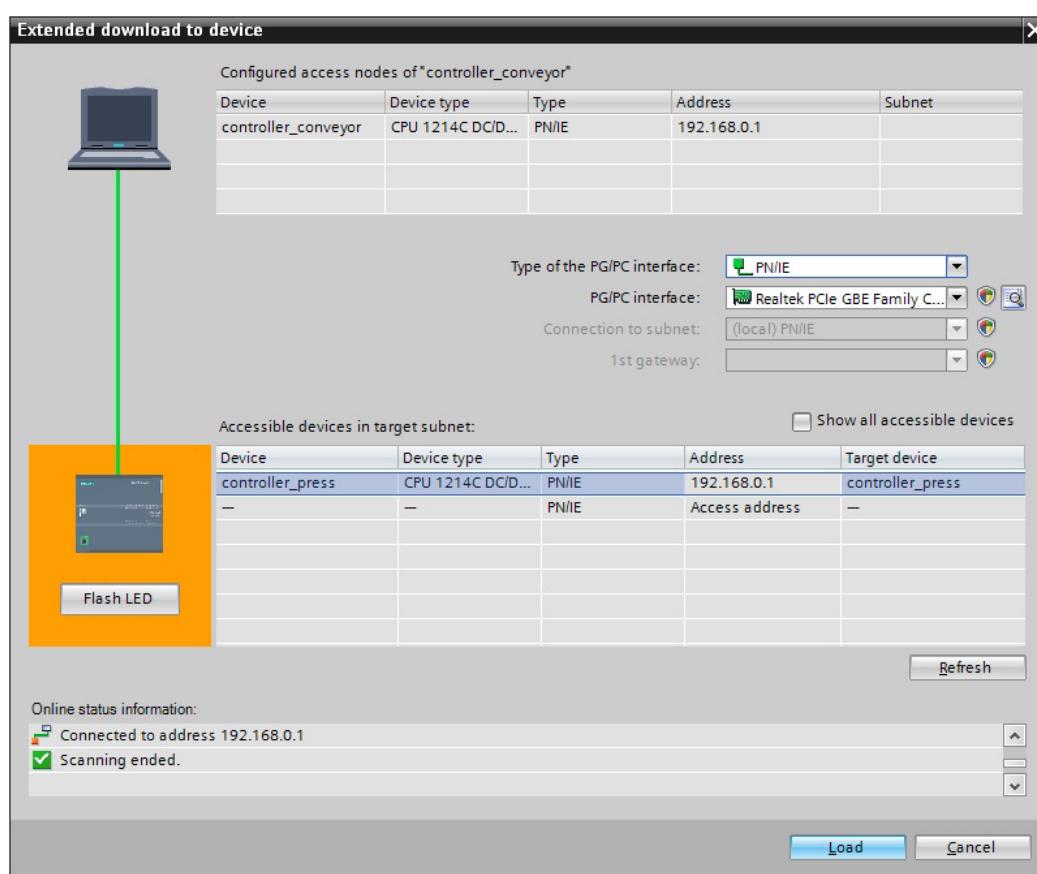
The Off button S4 is a break contact (NC) and has to be negated at the block during wiring; i.e., the Off function in the block is pending when the Off button S4 is operated and thus no signal is pending at terminal %I0.3.

18. To load your entire program to the CPU, first highlight the folder '**controller conveyor**' and then click on the symbol  Load to device. (→ controller_conveyor → 

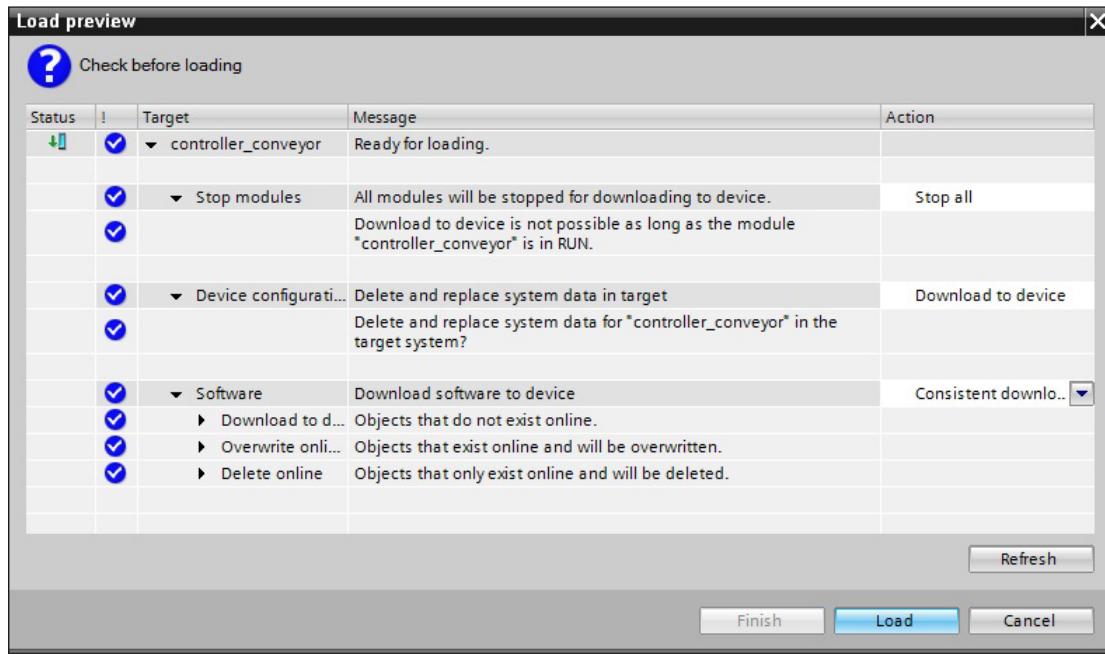


19. If you omitted specifying the PG/PC interface beforehand, a window is displayed where you can still do this.

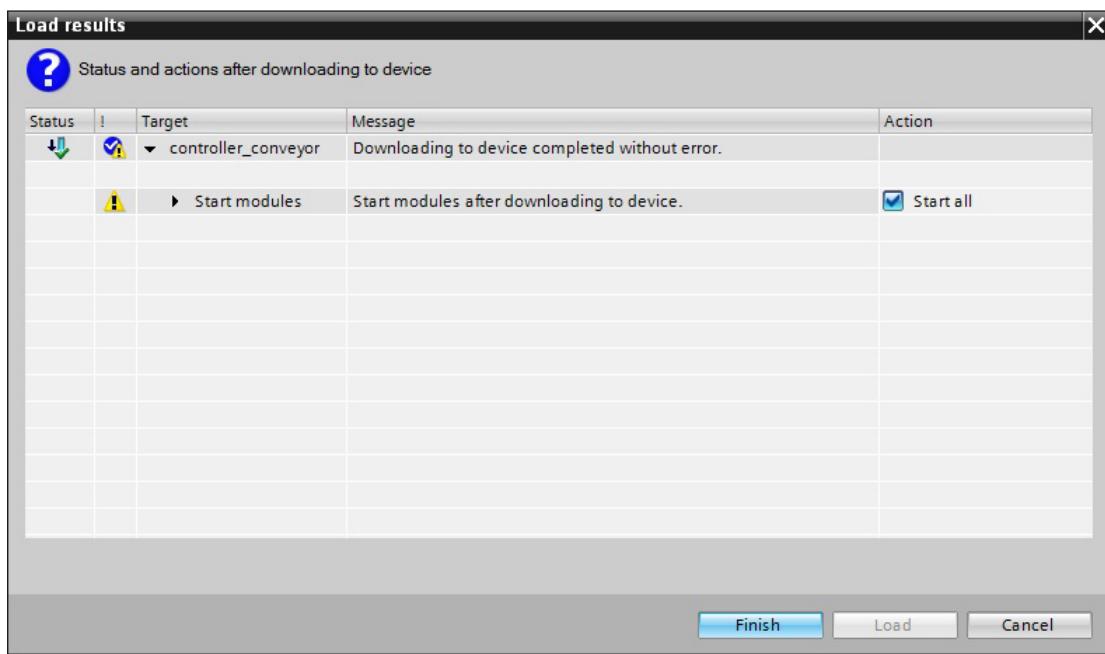
(→ PG/PC interface for loading → Load)



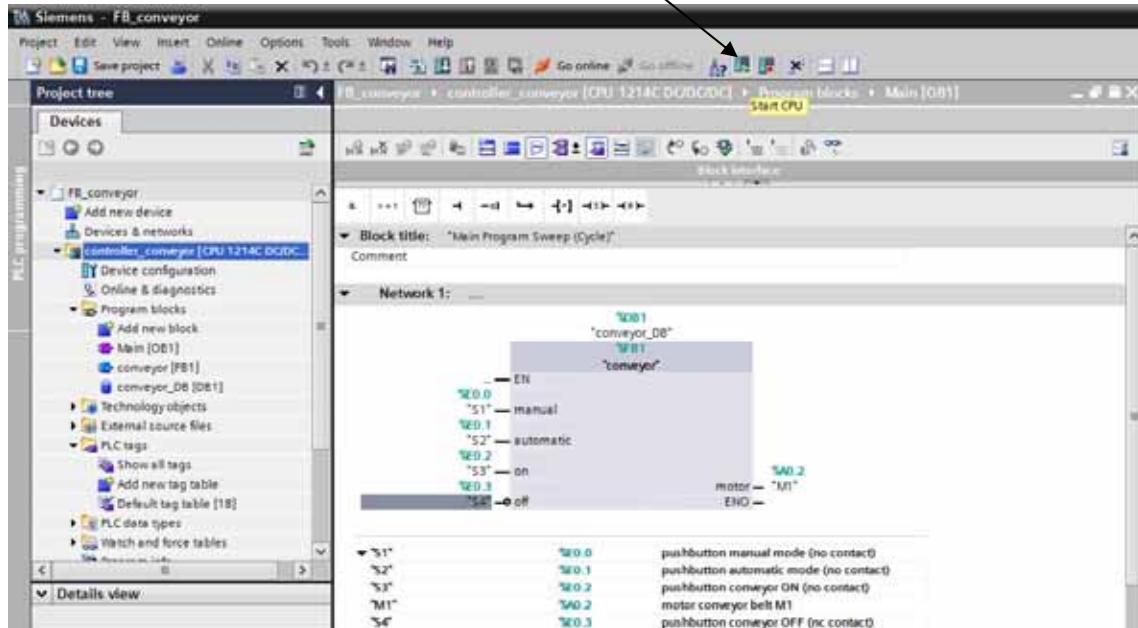
20. Click on 'Load' once more. In a window, the status is indicated during loading. (→ Load)



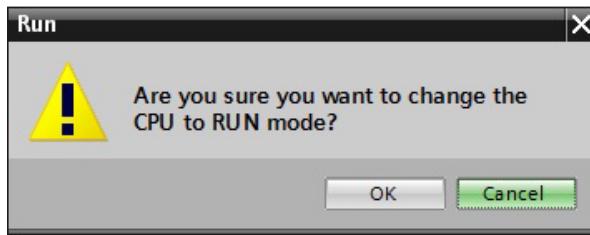
21. If loading was successful, it is shown in a window. Click on 'Finish'. (→ Finish)



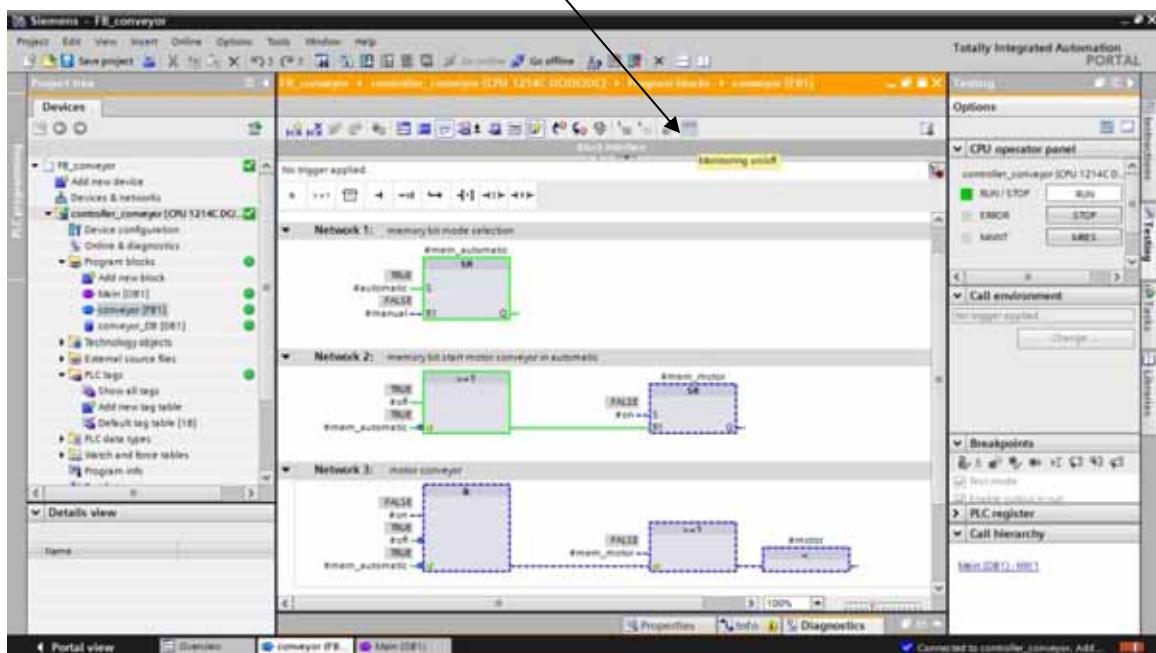
22. Now, start the CPU by clicking on the symbol .



23. With 'OK', confirm the question whether you actually want to start the CPU. (→ OK)



24. By clicking on the symbol  Monitoring On/Off, you can, during the program test, observe the status of the input and output variables at the block "conveyor", but also the program execution in the block "conveyor". (\rightarrow conveyor[FB1] \rightarrow 



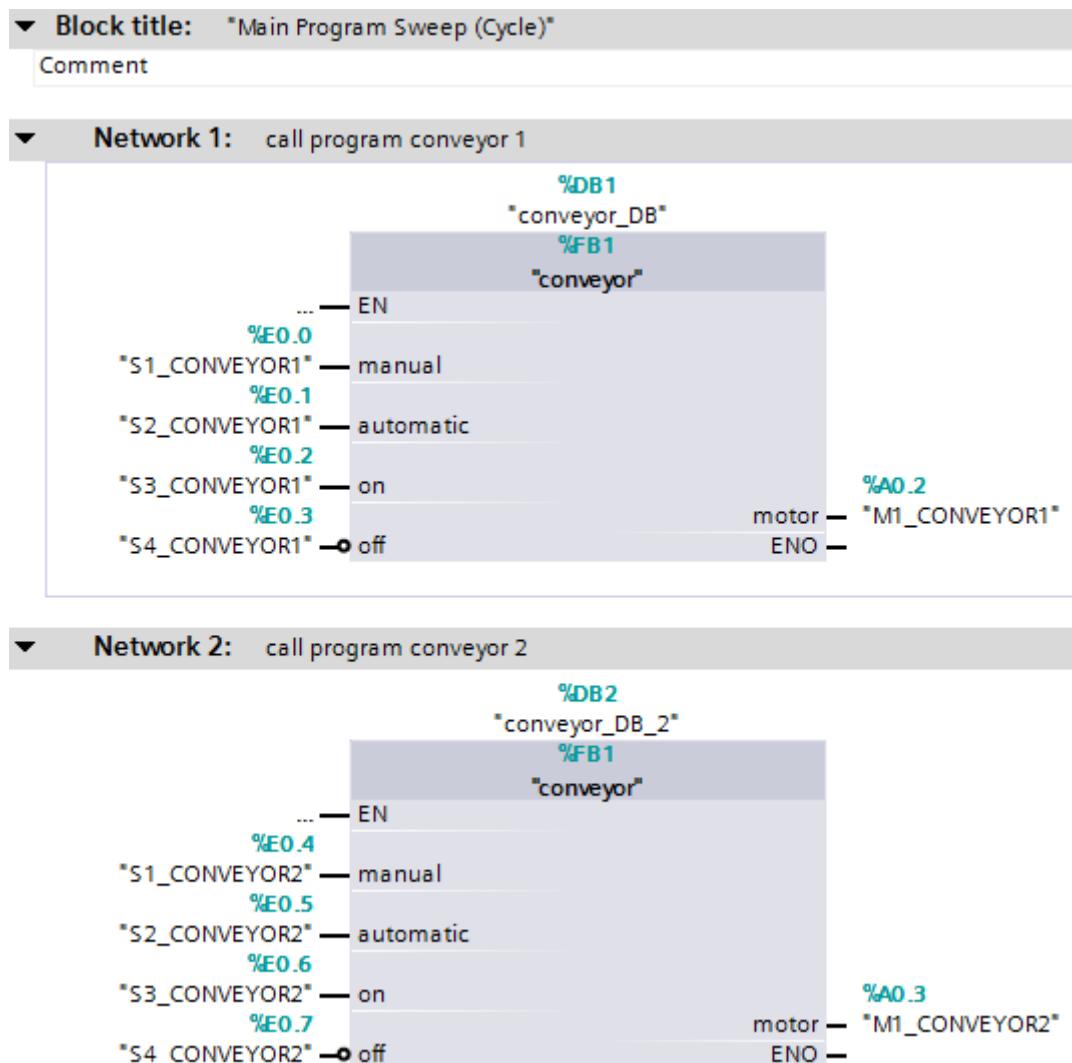
25. Since our block “conveyor” was generated according to the rules for standard blocks (no use of global variables within the block!!!!), it can now be used and called any number of times.

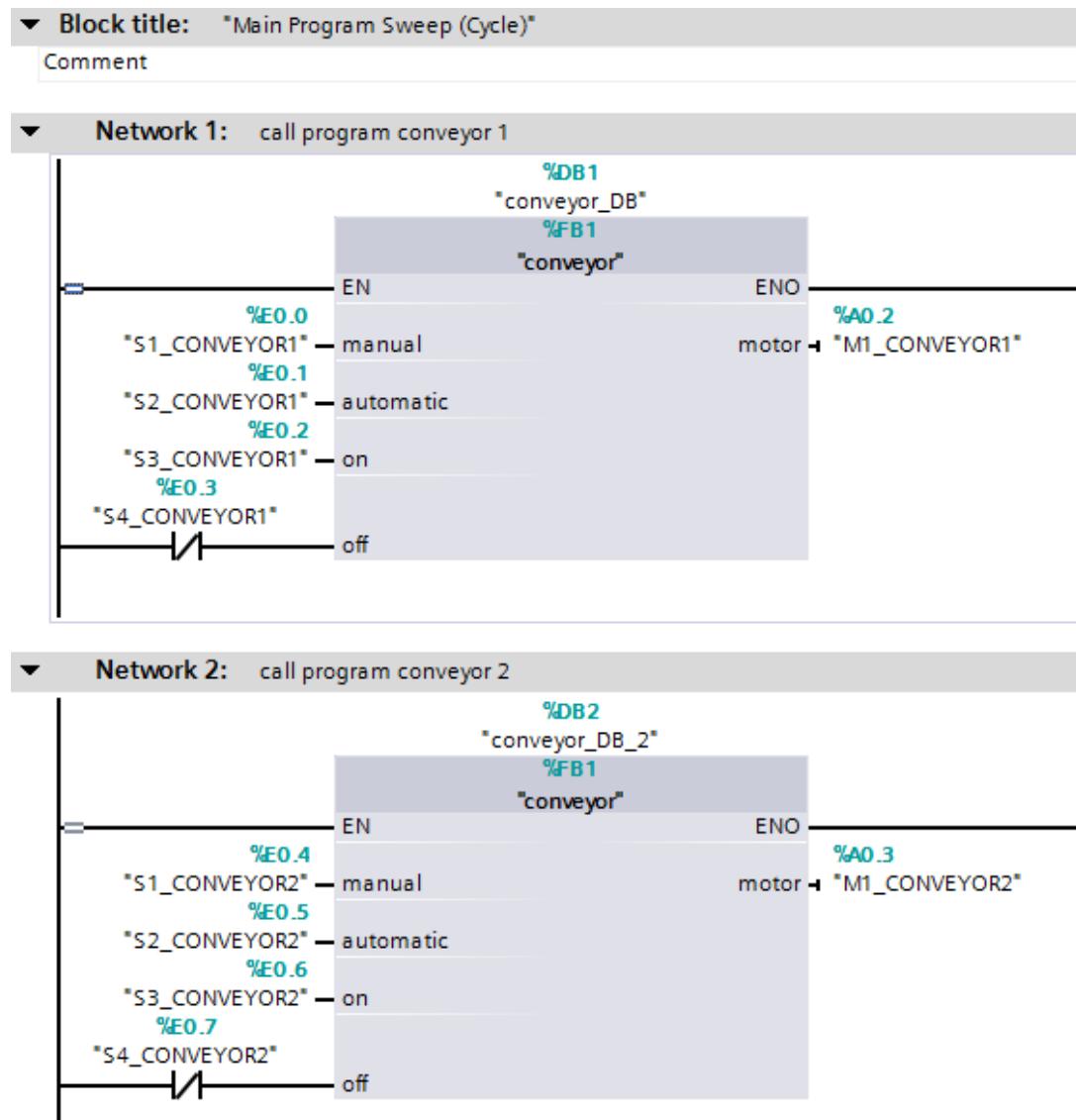
Below, an expanded PLC tag table is shown, with the inputs and outputs for two conveyors.

Default tag table							
	Name	Data type	Address	Retain	Visible	Access	Comment
1	S1_CONVEYOR1	Bool	%E0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 pushbutton manual mode (no contact)
2	S2_CONVEYOR1	Bool	%E0.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 pushbutton automatic mode (no contact)
3	S3_CONVEYOR1	Bool	%E0.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 pushbutton conveyor ON (no contact)
4	S4_CONVEYOR1	Bool	%E0.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 pushbutton conveyor OFF (nc contact)
5	M1_CONVEYOR1	Bool	%A0.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 motor conveyor belt M1
6	S1_CONVEYOR2	Bool	%E0.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor2 pushbutton manual mode (no contact)
7	S2_CONVEYOR2	Bool	%E0.5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor2 pushbutton automatic mode (no contact)
8	S3_CONVEYOR2	Bool	%E0.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor2 pushbutton conveyor ON (no contact)
9	S4_CONVEYOR2	Bool	%E0.7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor2 pushbutton conveyor OFF (nc contact)
10	M1_CONVEYOR2	Bool	%A0.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor2 motor conveyor belt M1
11	Add new...			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

26. Now, the block "**conveyor**" can also be called twice in OB1, with different wiring respectively. For each call, another instance data block is specified.

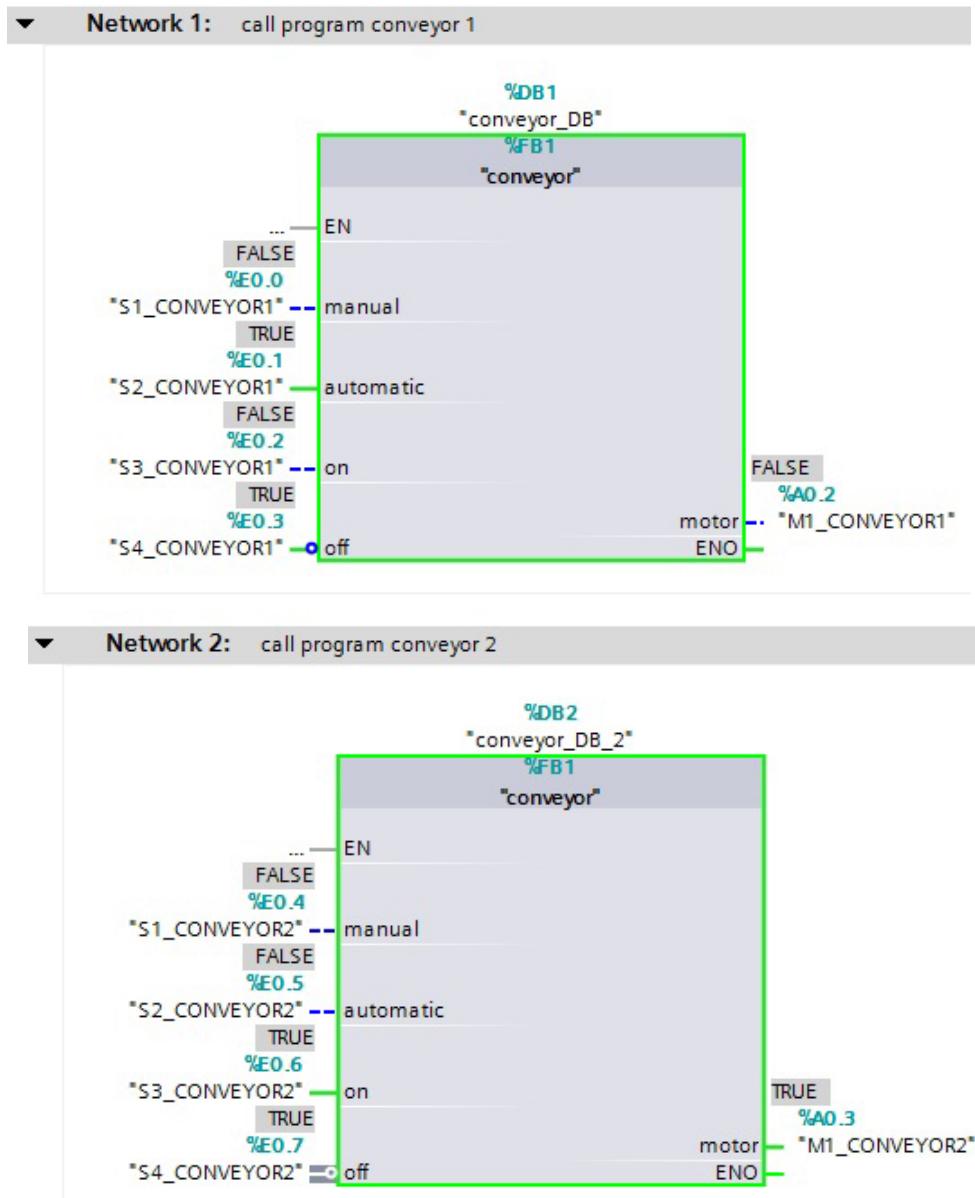
Program in function block diagram (FBD):



Program in ladder diagram (LAD):

Two conveyors separated from each other can now be controlled by means of the same conveyor block. Only another instance data block has to be assigned for each call.

2



Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Module 010-030



TIA Portal Module 010-010

Startup Programming of the
SIMATIC S7-1200

TIA Portal Module 010-020

Block Types for the SIMATIC S7-1200

TIA Portal Module 010-030

IEC Timers and IEC Counters at the
SIMATIC S7-1200

TIA Portal Module 010-040

Diagnosis and Error Search at the
SIMATIC S7-1200

TIA Portal Module 010-050

Analog Value Processing with
SIMATIC S7-1200

TIA Portal Module 010-060

Control Engineering with
SIMATIC S7-1200

TIA Portal Module 010-070

Communication between two
SIMATIC S7-1200

TIA Portal Module 010-080

HMI Panel KTP600 with SIMATIC
S7-1200

TIA Portal Module 010-090

Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

Matching SCE training packages for these documents

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
Order number: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
Order number 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
Order number 6ES7822-0AA01-4YE0

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

3

Continued Training

For regional Siemens SCE continued training, please contact your regional SCE contact person
siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding Usage

The training curriculum 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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by 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.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the documents being used commercially.

We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this document.

PAGE

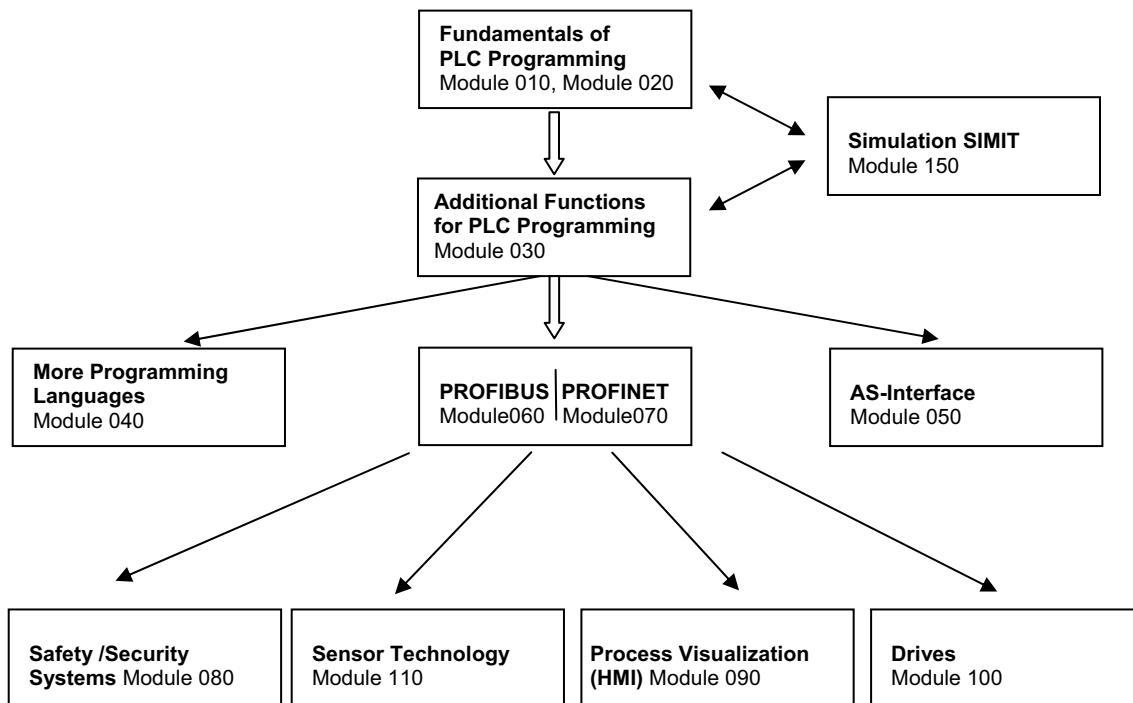
Contents

	PAGE
1. Preface	4
2. Notes on Programming the SIMATIC S7-1200	6
2.1 Automation System SIMATIC S7-1200	6
2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)	6
3. Instances and Multi-Instances when Programming the SIMATIC S7-1200	7
3.1 Instance Data Blocks/Single Instances	7
3.2 Multi-Instances	9
4. Sample Task: Press Control with Timer and Instance DB	11
5. Programming the Press with a Time Delay using the SIMATIC S7-1200	12
6. Sample Task for Conveyor Control with Counter and Multi-Instance	29
7. Programming the Conveyor with the SIMATIC S7-1200	30

3

1. Preface

Regarding its content, module SCE_EN_010-030 is part of the training unit '**Basics of PLC Programming**' and represents a fast **entry point** for programming the SIMATIC S7 1200 with TIA Portal.



Training Objective:

In this module 010-030, the reader learns how to program the programmable logic controller (PLC) SIMATIC S7-1200 using the programming tool TIA Portal. Module 010-030 provides the fundamentals and shows in the steps below how it is done based on a detailed example. .

- Installing the software and setting the program interface
- Explanation of what a PLC is and how it processes
- Configuration and operation of the PLC SIMATIC S7-1200
- Creating, loading and testing a sample program

Prerequisites:

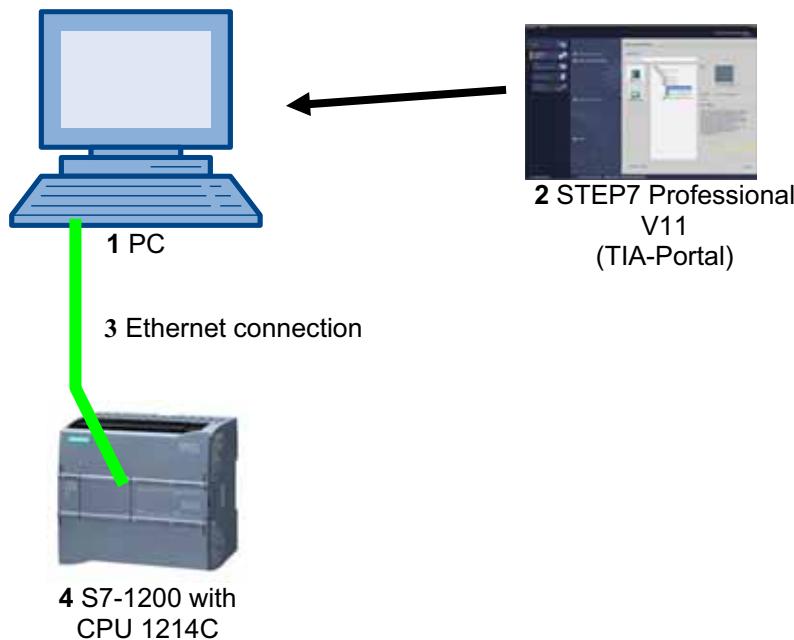
To successfully work through module 010-030, the following knowledge is assumed:

- How to operate Windows
- Basics of PLC programming with the TIA Portal (for example, Module 010-010 'Startup' Programming the SIMATIC S7-1200 with TIA Portal V11)
- Blocks for the SIMATIC S7-1200 (for example, Module 010-020 Block Types at the SIMATIC S7-1200)

Hardware and software required

- 1 PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB
Operating system Windows XP Professional SP3/Windows 7 Professional/Windows 7 Enterprise/Windows 7 Ultimate/Windows 2003 Server R2/Windows Server 2008 Premium SP1, Business SP1, Ultimate SP1
- 2 Software STEP7 Professional V11 SP1 (Totally Integrated Automation (TIA) Portal V11)
- 3 Ethernet connection between PC and CPU 315F-2 PN/DP
- 4 PLC SIMATIC S7-1200; for example CPU 1214C.
The inputs have to be brought out to a panel.

3



2. Notes on Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The SIMATIC S7-1200 automation system is a modular mini-control system for the lower and medium performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU, and input and output modules for digital and analog signals.

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

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process; the IO modules are polled in the S7 program by means of the input addresses (%I), and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7.

2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 (TIA Portal V11) is the programming tool for the automation systems

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating visual displays for the SIMATIC basic panels with integrated WinCC Basic
- With additional WinCC packages, visualization solutions for PCs and other panels can be generated

All functions are supported with detailed online help.

3. Instances and Multi-Instances when Programming the SIMATIC S7-1200

Calling a function block is referred to as **instance**. To each call of a function block, an **instance data block** is assigned that is used for data storage. The actual parameters and the static data are stored here.

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

Applying single and multi-instances

3

Instance data blocks can be assigned as follows:

- Call as **single instance**:
 - A separate instance data block for each instance of a function block
- Call as **multi-instance**:
 - One instance data block for several instances of one or several function blocks

3.1 Instance Data Blocks/Single Instances

The call of a function block to which its own instance data block is assigned is referred to as **single instance**.

If the function block was generated according to the rules for standard blocks (refer to Module 010-020), it can be called multiple times.

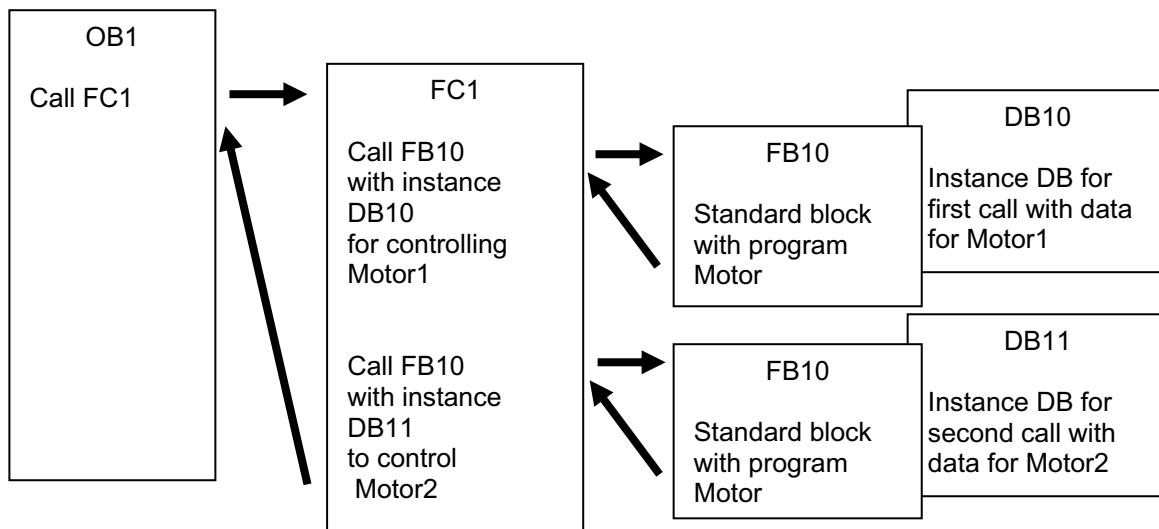
However, for each call as single instance, you have to assign a different instance data block.

Example of single instances:

The figure below shows two motors being controlled with a function block FB10 and two different data blocks:

The different data for the individual motors -for example, speed, power-up time, total operating time- is stored in the different instance data blocks DB10 and DB11.

3

**Note**

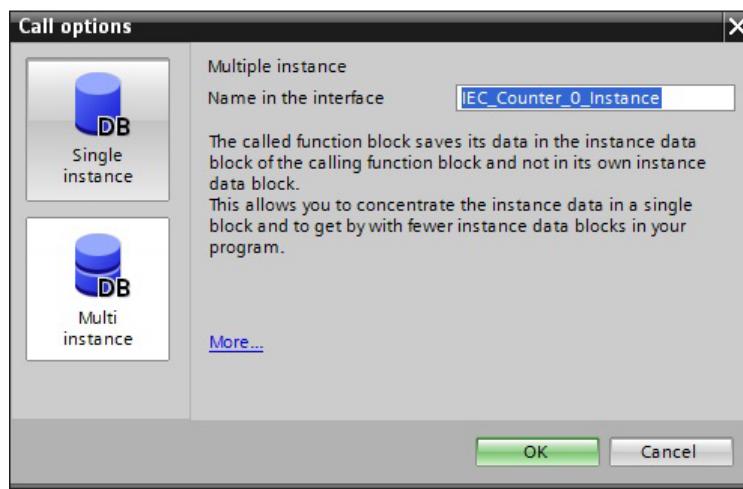
Some instructions such as timers and counters behave like function blocks. If they are called, they also represent instances and need an assigned memory area; in the form of an instance data block, for example.

3.2 Multi-Instances

Because of the memory capacity of the CPUs used, it is possible that you want to or you can allocate only a limited number of data blocks for instance data.

If in your user program, additional already existing function blocks, timers, counters, etc. are called in a function block, it is possible to call these additional function blocks without their own (that is, additional) instance DBs.

Simply select the call options '**Multi-Instance**':



Notes:

For a function block that was called, multi-instances make it possible to place its data in the instance data block of the function block that is calling.

The block that is calling always has to be a function block in this case.

In this way, you concentrate the instance data in one instance data block; i.e., you can utilize the available number of DBs more efficiently.

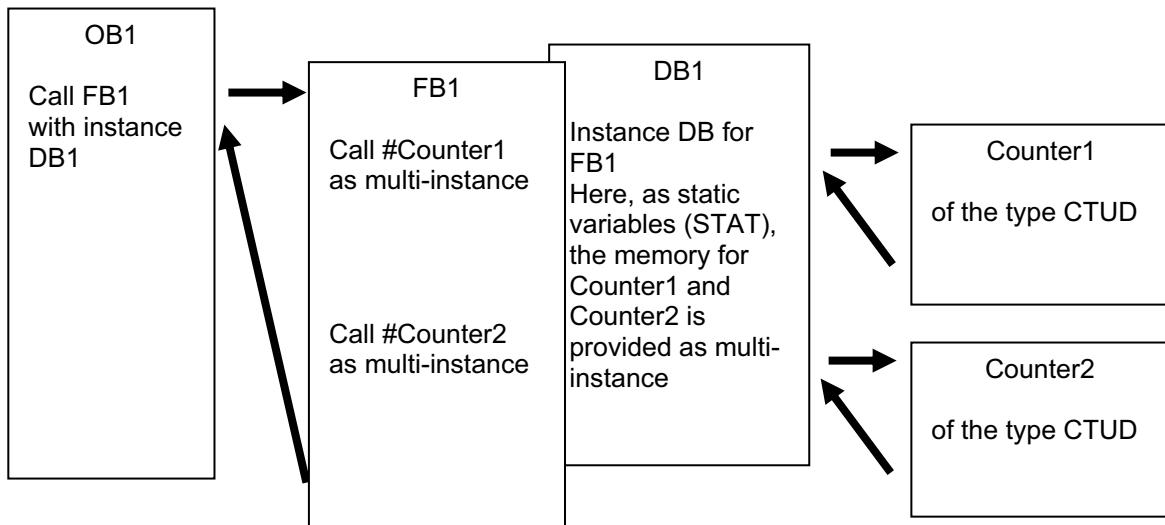
This, by the way, always has to be done if the block that is calling is to be reusable as a standard block.

Example for Multi-Instances:

The figure below shows a counter of the type CTUD (up and down counter) being called twice..

The different data for the two counters is stored as different **multi-instances** in instance data block DB1 of the calling function block FB1.

3



4. Sample Task: Press Control with Timer and Instance DB

For our program, a timer will be added to the press control in Module 010-010.

The task to be performed is as follows:

A press with a safety fence is to be started with a START button S3 only if the safety fence is closed. This state is monitored with a sensor Safety fence closed B1.

If this is the case, a 5/2 way valve M0 for the press cylinder is activated so that a plastic shape can be pressed.

The press is to retract again when the EMERGENCY OFF button (NC) is operated, or the sensor Safety Fence B1 no longer responds.

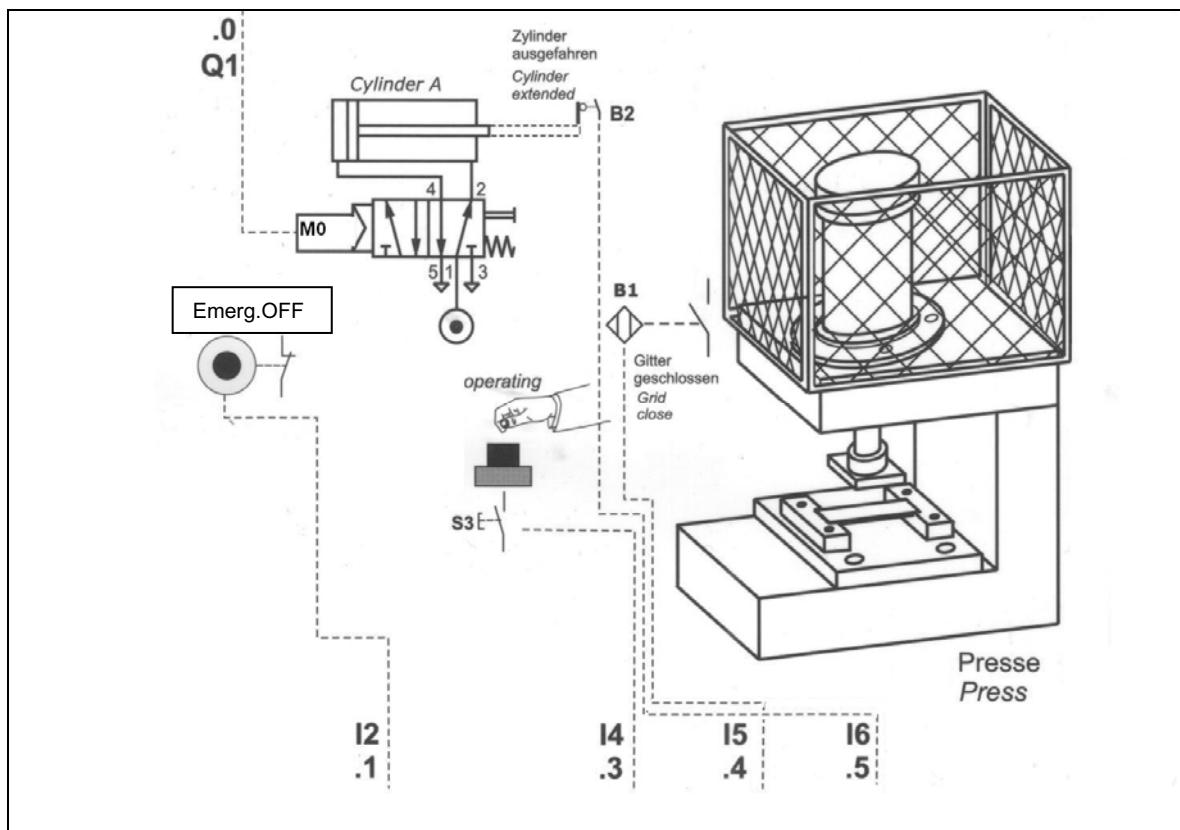
If the sensor Cylinder extended B2 responds, the press is to retract again after a press time of 5 seconds.

An instance DB is used as the memory for the timer.

3

Assignment list:

Address	Symbol	Comment
%I 0.1	EMERGENCY OFF	EMERGENCY OFF button NC
%I 0.3	S3	Start button S3 NO
%I 0.4	B1	Sensor Safety fence closed NO
%I 0.5	B2	Sensor Cylinder extended NO
%Q 0.0	M0	Extend Cylinder A



5. Programming the Press with a Time Delay using the SIMATIC S7-1200

The software '**Totally Integrated Automation Portal**' manages the project and does the programming.

Here, under a uniform interface, the components such as the controller, visualization and networking the automation solution are set up, parameterized and programmed.

Online tools are provided for error diagnosis.

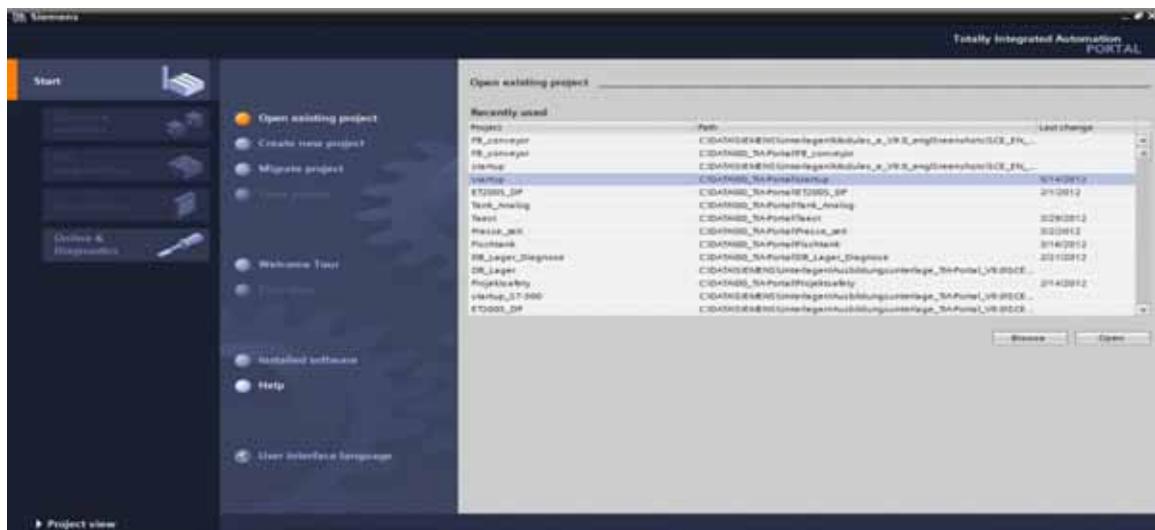
3

In the steps that follow, a project can be opened for the SIMATIC S7-1200, it can be stored under a different name and adapted to the new requirement.

1. The central tool is the '**Totally Integrated Automation Portal**'. Here, we call it with a double click. (→ Totally Integrated Automation Portal V11)

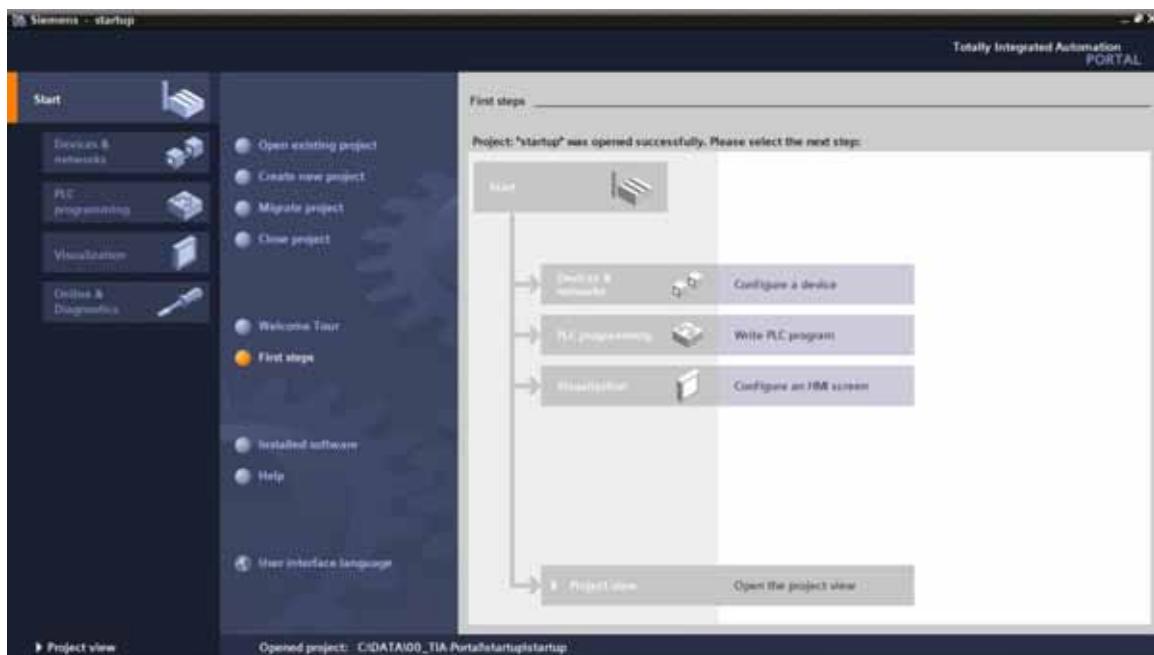


2. The project "**startup**" from Module 010-010 is now opened in the portal view as the basis for this program. (→ Open existing project → **startup** → open)

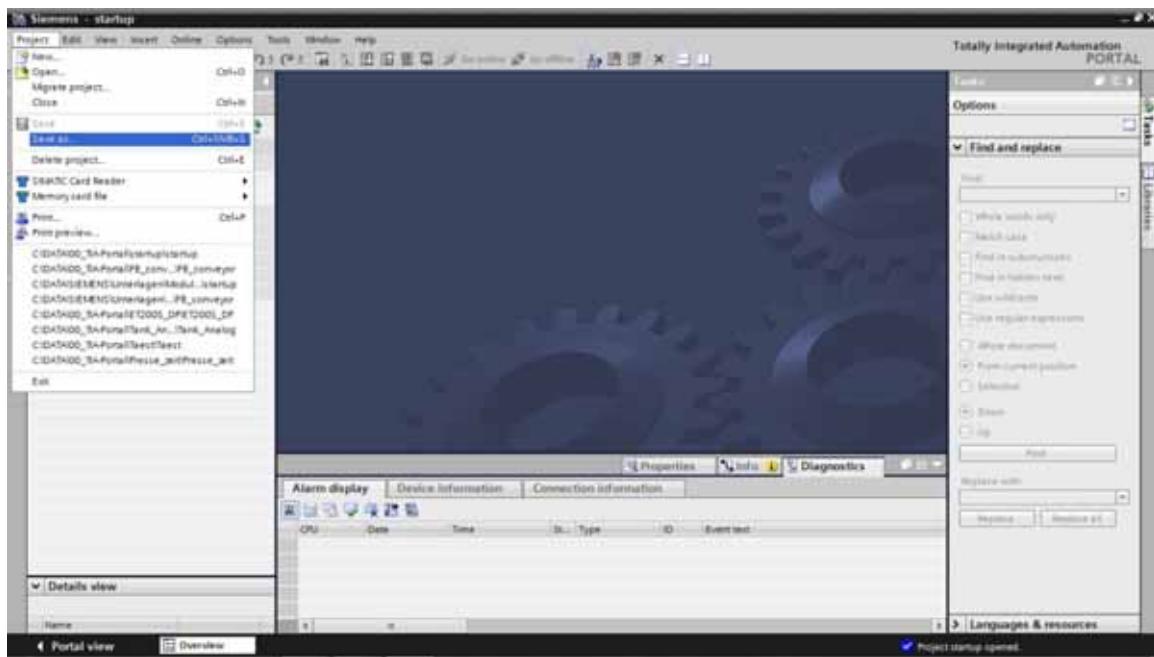


3

3. Next, '**First Steps**' for the configuration are suggested. We want to '**Open project view**'. (→ Open project view)

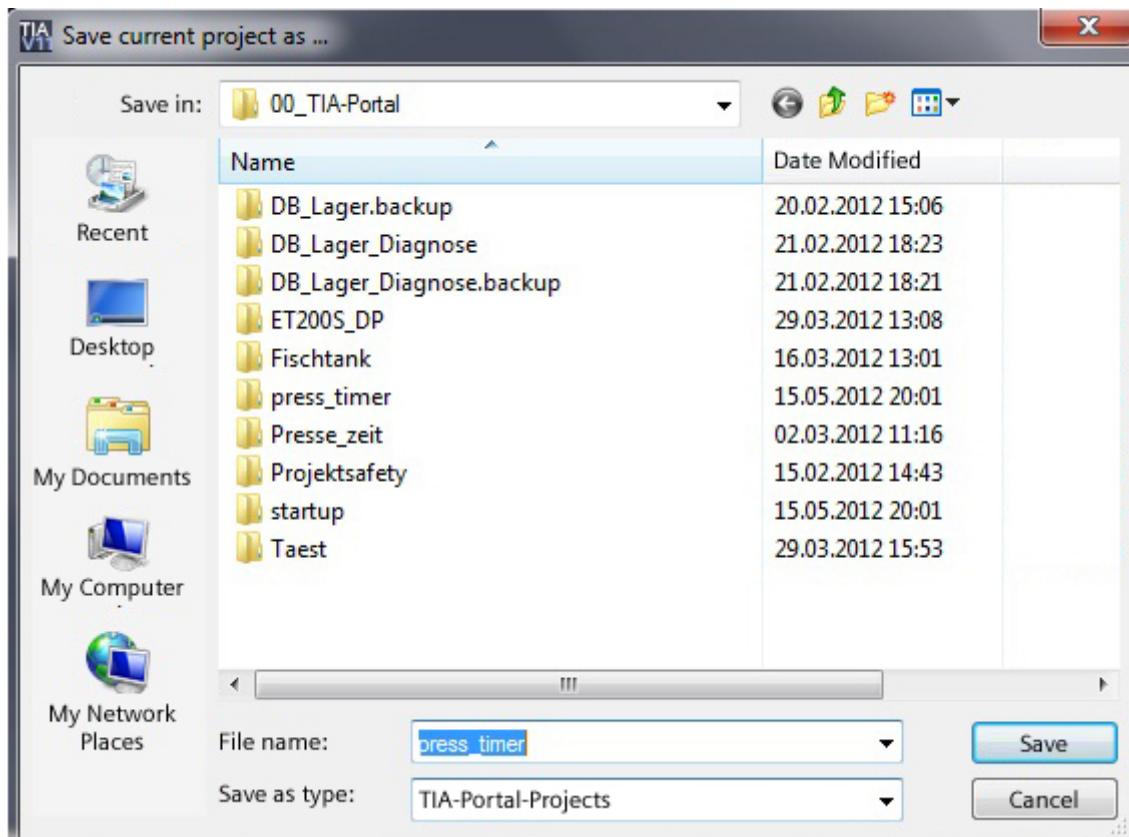


4. First, we want to save the project under another name. (→ Project → Save as)

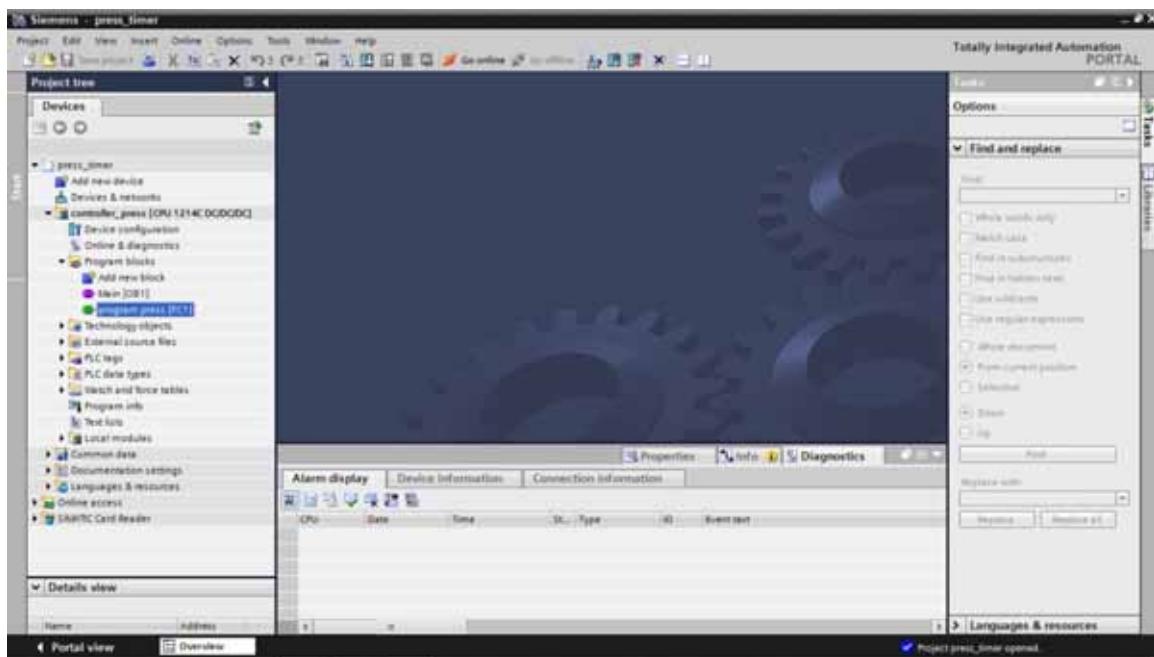


3

5. Now, 'Save' the project under the new name '**press_timer**'. (→ press_timer → Save)



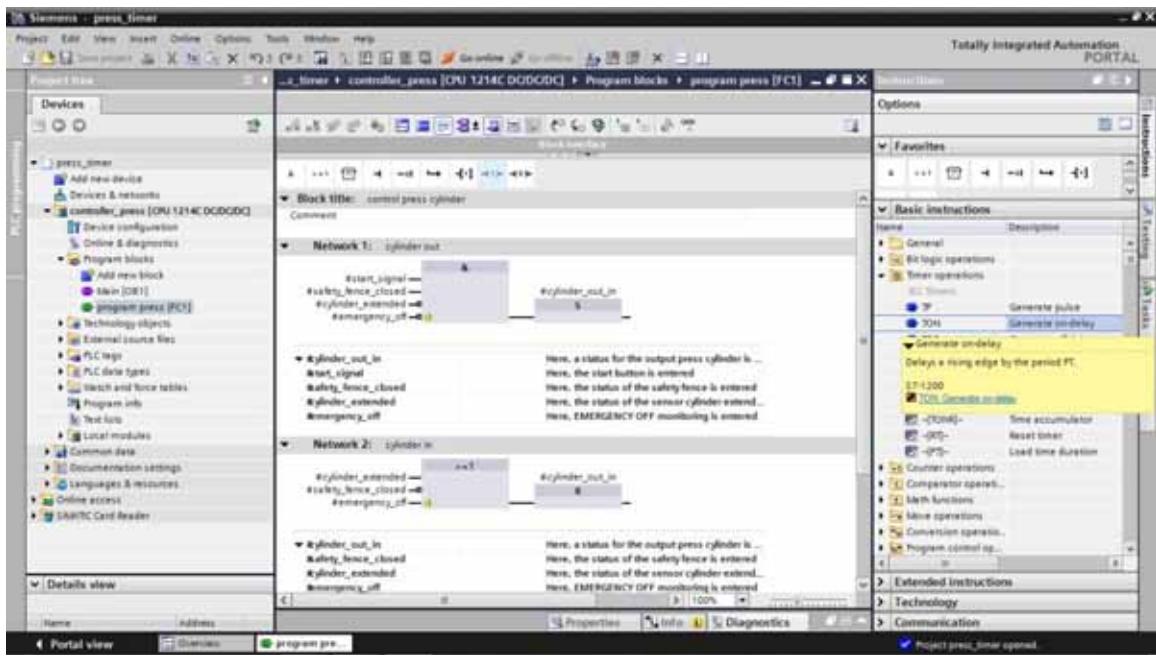
6. To make the changes, open the block '**program press[FC1]**' with a double click. (→ program press[FC1])



3

7. Now we can start changing the program.

When generating our solution with the delay, we need an ON delay '**TON**'. It is located under '**Instructions**' in the folder '**Timer operations**'. If you point with the mouse to an object such as the time TON, details about this object will be provided. (→ Instructions → Timer operations → TON)



8. If you highlight an object and then press the button '**F1**' on your PC, online help regarding this object is displayed in a window to the right. (→ F1)

The screenshot shows the SIMATIC Manager Information System interface. On the left is a tree view of documentation topics, and on the right is a detailed help page for the 'TON: Generate on-delay' instruction.

TON: Generate on-delay

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	input	BOOL	I, Q, M, D, L	Start input
PT	input	TIME	I, Q, M, D, L or constant	Duration of the on-delay. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

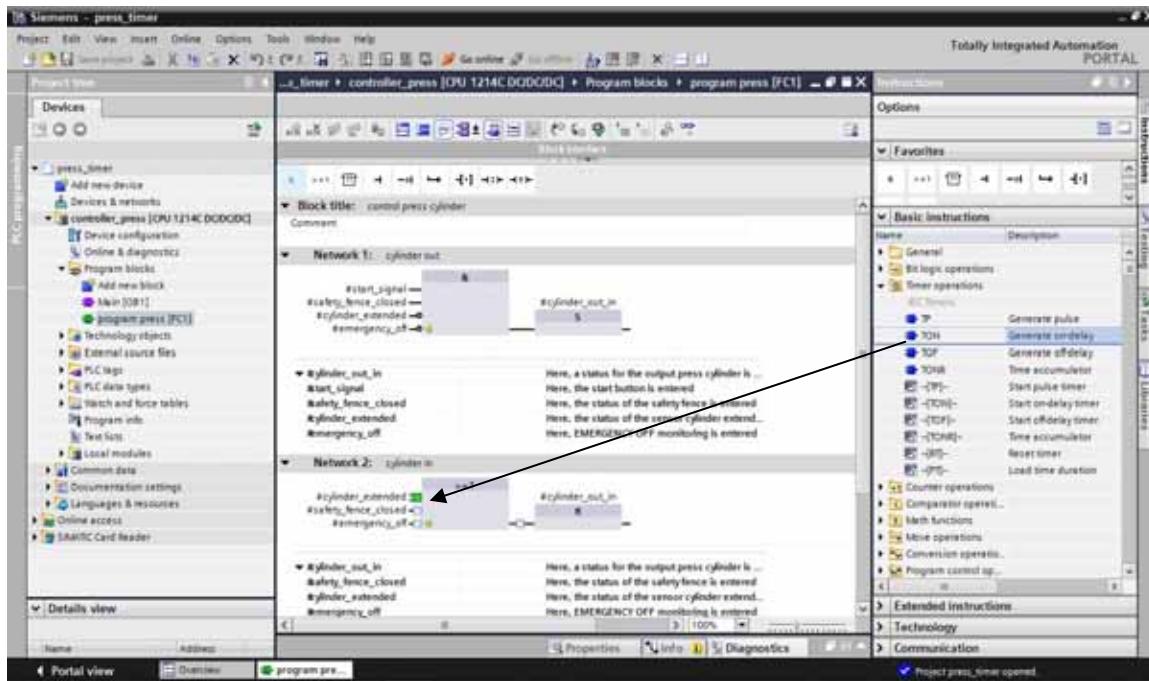
The following figure shows the pulse diagram of the "Generate on-delay" instruction:

The pulse diagram illustrates the behavior of the TON instruction. The IN signal is a square wave. The Q output signal is active whenever the IN signal is active and the timer has not yet expired. The ET signal shows the current time value.

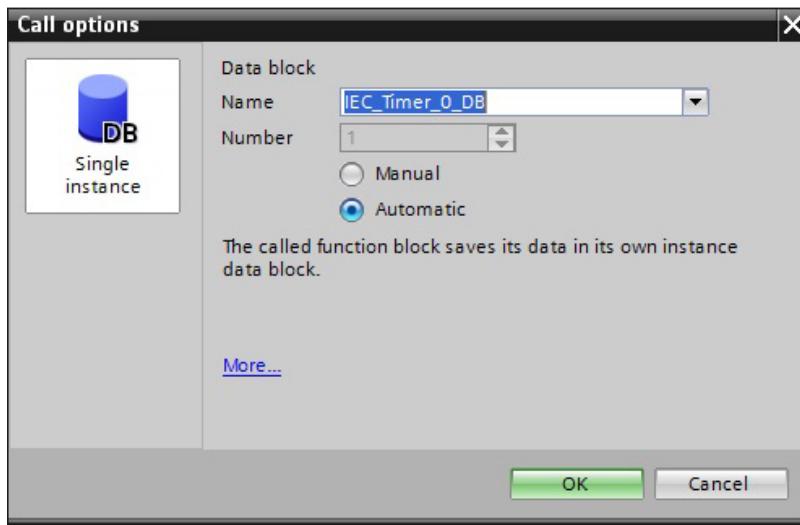
Note

Here, go to online help where details are provided about all timer functions.

9. Next, drag the timer 'TON' with the mouse to the third contact of the OR function behind the variable '#cylinder_extended' ($\rightarrow \text{TON} \rightarrow \#cylinder_extended$)



10. For the timing function we need memory. Here, it can be made available only by generating a new instance data block as a '**Single instance**'. (→ OK)



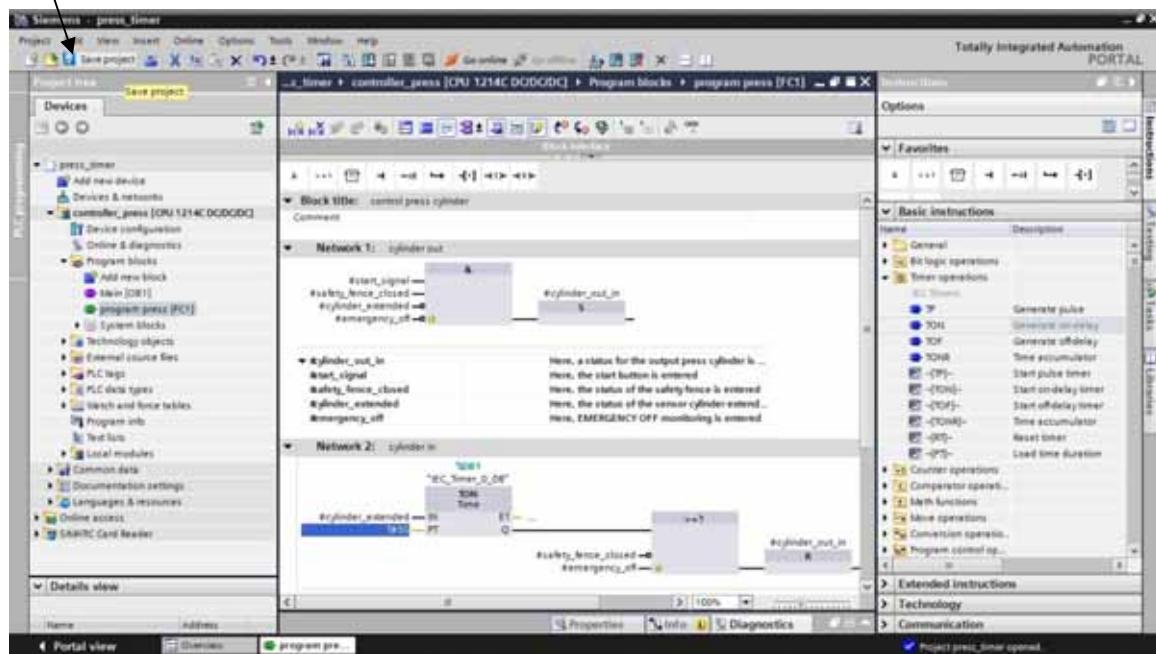
3

Note

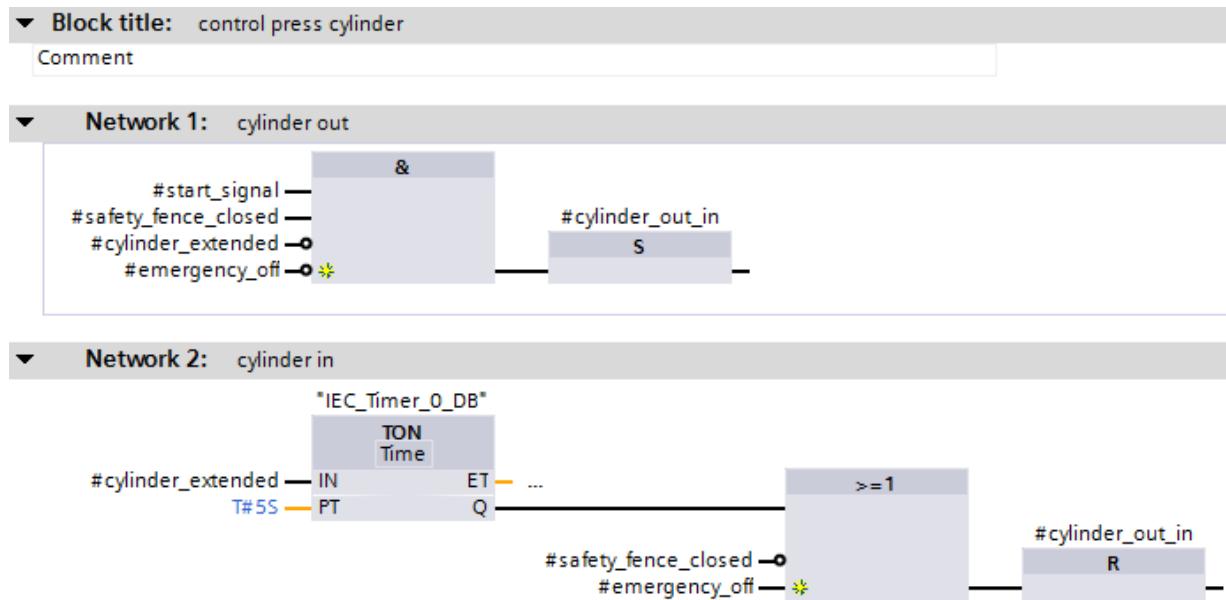
Multi-instances can be used only when programming within a function block. This will be shown below in the example for the IEC counter.

11. Now, connect the time delay 'TON' with the time base 't#5s' for 5 seconds. By clicking on

Save project, the project is saved. ($\rightarrow t\#5s \rightarrow$ Save project)

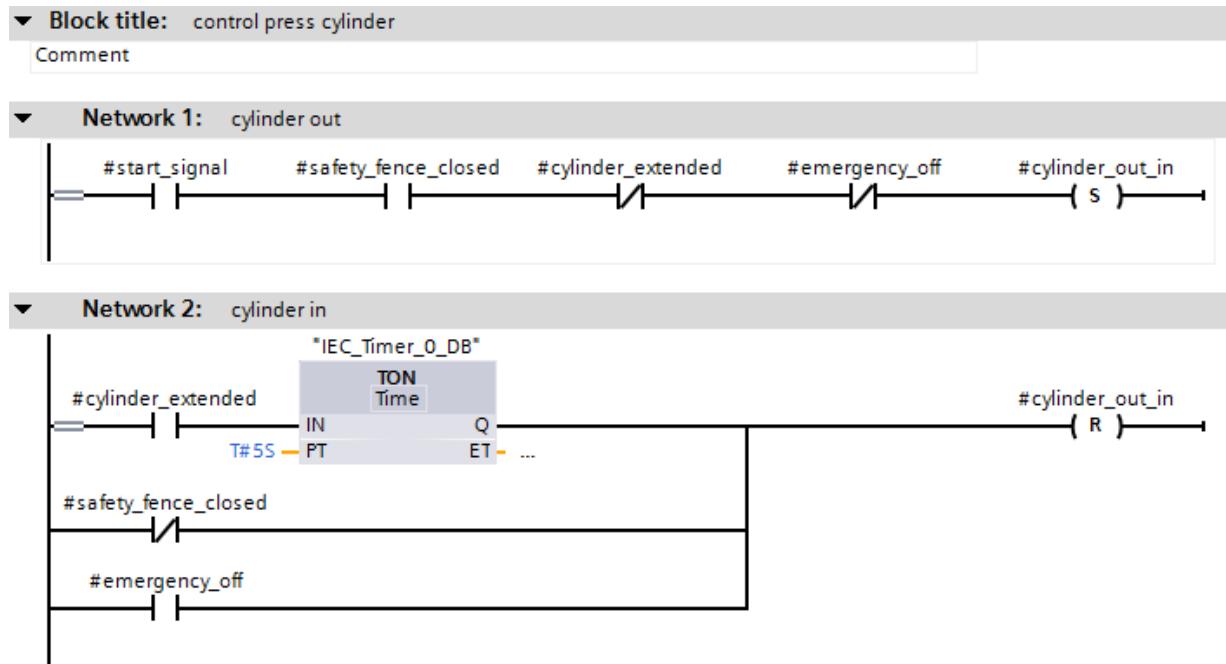


Program in function block diagram (FBD)



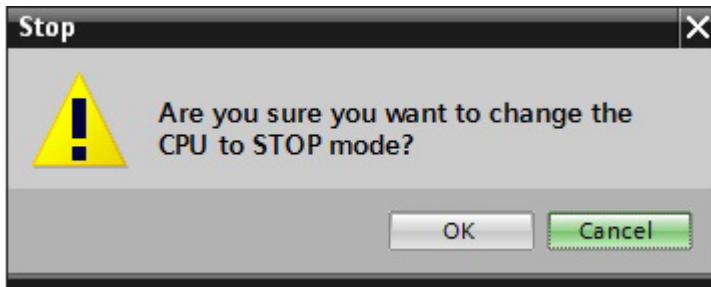
3

Program in ladder diagram (LAD)



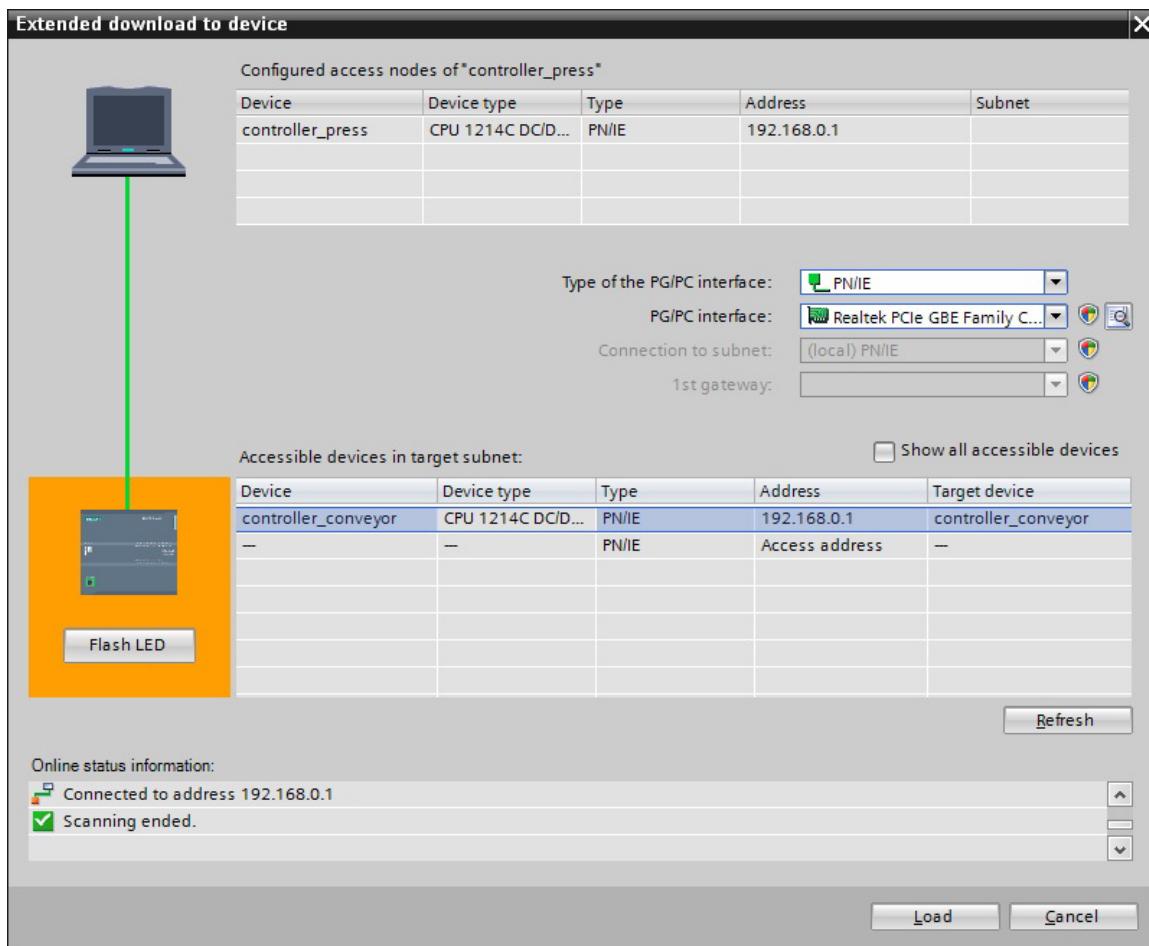
12. To load your entire program into the CPU, highlight the folder '**controller_press**' and then click on the symbol  Load to device. (→ controller_press → 

13. If the CPU is in the '**RUN**' mode, you will be asked whether you want to take it to the '**STOP**' mode. Confirm with '**OK**'. (→ OK)

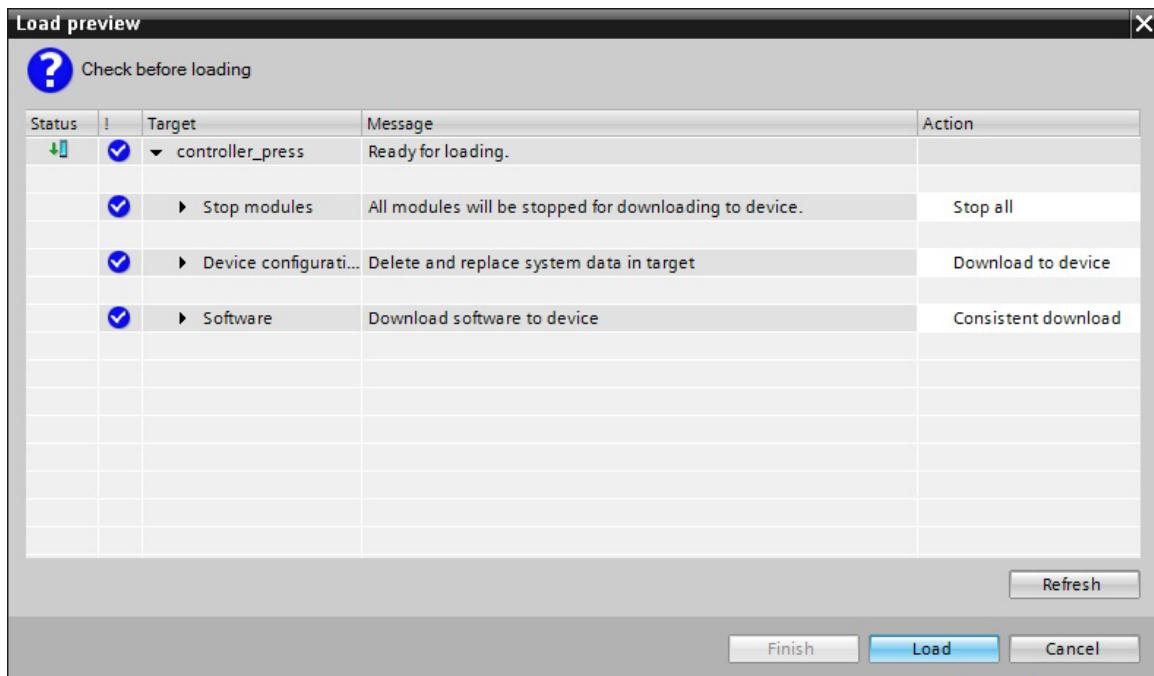


3

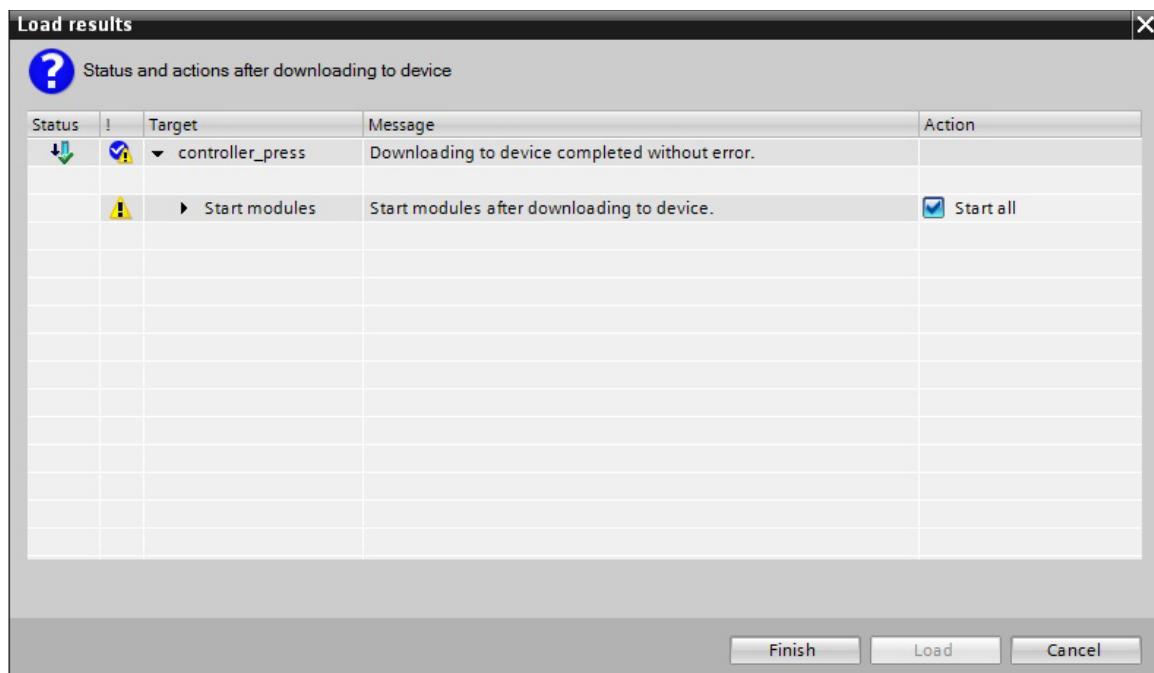
14. If you omitted to specify the PG/PC interface beforehand, a window is displayed where you can do this now. (→ PG/PC interface for loading → Load)



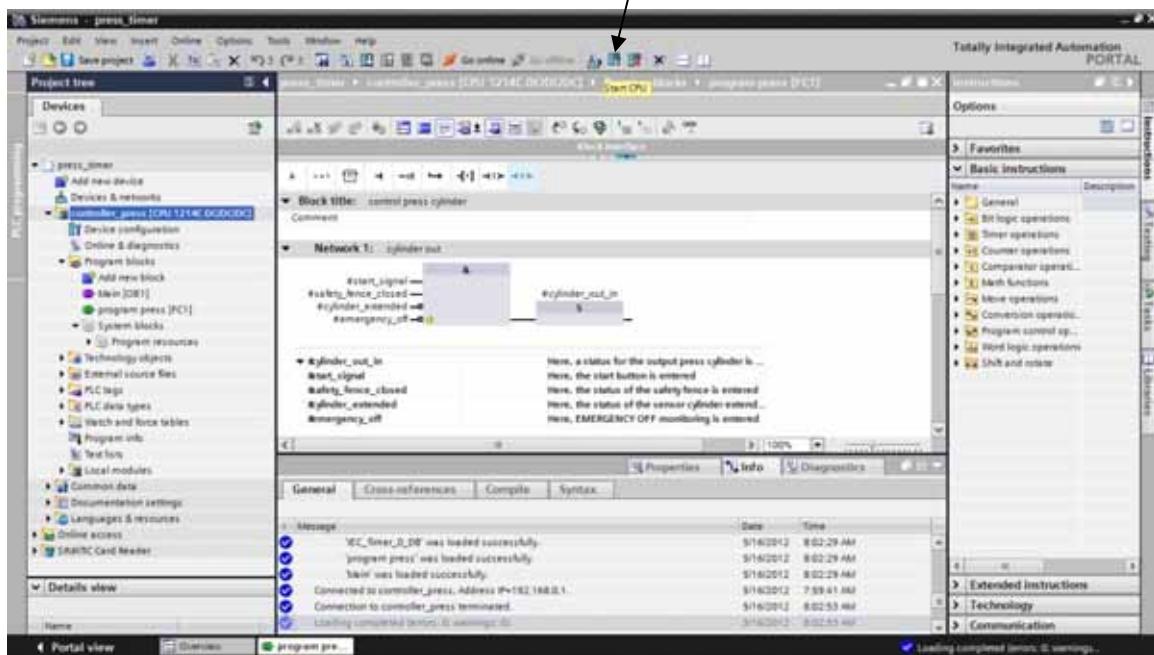
15. Confirm '**Load**' once more. During loading, the status is shown in a window. (→ Load)



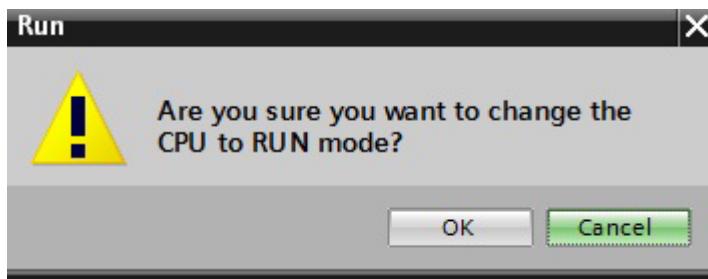
16. If loading was successful, it is displayed in a window. Now click on '**Finish**'. (→ Finish)



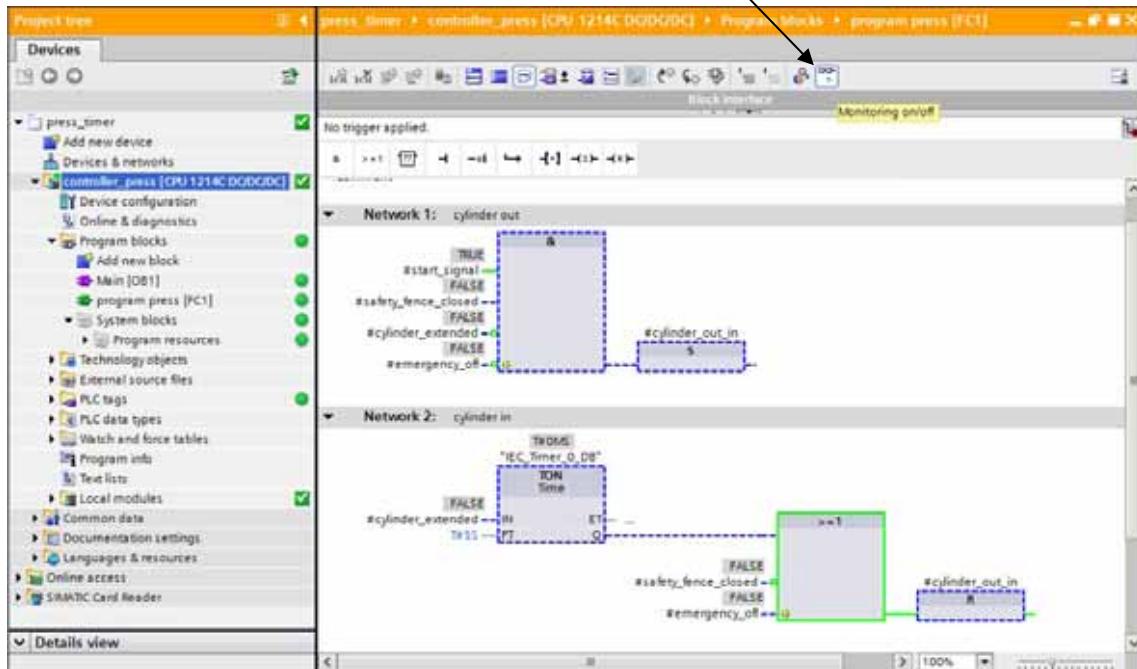
17. Next, start the CPU by clicking on the symbol  . (→ 



18. Confirm the question whether you actually want to start the CPU with 'OK'. (→ OK)



19. By clicking on the symbol Monitoring on/off you can, while the program is tested, observe the status of the timer as well as the time that expired. (→ 



6. Sample Task for Conveyor Control with Counter and Multi-Instance

When blocks are to be generated that are working in any program like a "Black Box" as it were, they have to be programmed by using variables. In this case, the following rule applies: that in these blocks, no absolute-addressed inputs/outputs, flags etc. must be used. Within the block, only variables and constants are used.

If secondary function blocks -or timers/counters- are called from a block that can be used multiple times, they must not be assigned their own data block.

The required memory is provided as **multi-instance** within the instance DB that is assigned to the function block doing the calling.

3

In the example below, we add a bottle counter to the function block that already contains a conveyor control dependent on the operating mode.

With this conveyor, 20 bottles are to be transported to a case. When the case is full, the conveyor is stopped and the case has to be exchanged.

With the button 'S1', we want to select the operating mode 'Manual', and with the button 'S2' the operating mode 'Automatic'.

In the operating mode 'Manual', the motor is switched on as long as the button 'S3' is operated; button 'S4' must not be operated.

In the operating mode 'Automatic', the conveyor motor is switched on with button 'S3' and switched off with button 'S4' (break contact).

In addition, there is a sensor 'B0' that counts the bottles into the case. After counting 20 bottles, the conveyor is stopped.

When a new case is put in place, this has to be confirmed with the button 'S5'.

Assignment list:

Address	Symbol	Comment
%I 0.0	S1	Button operating mode Manual S1 NO
%I 0.1	S2	Button operating mode Automatic S2 NO
%I 0.2	S3	On button S3 NO
%I 0.3	S4	Off button S4 NC
%I 0.6	S5	Button S5 NO Reset counter/new case
%I 0.7	B0	Sensor B0 NO bottle counter
%Q 0.2	M1	Conveyor motor M1

7. Programming the Conveyor with the SIMATIC S7-1200

The '**Totally Integrated Automation Portal**' software manages the project and does the programming.

Here, under a uniform interface, the components such as controller, visual display and networking of the automation solution are set up, parameterized and programmed. Online tools are provided for error diagnosis.

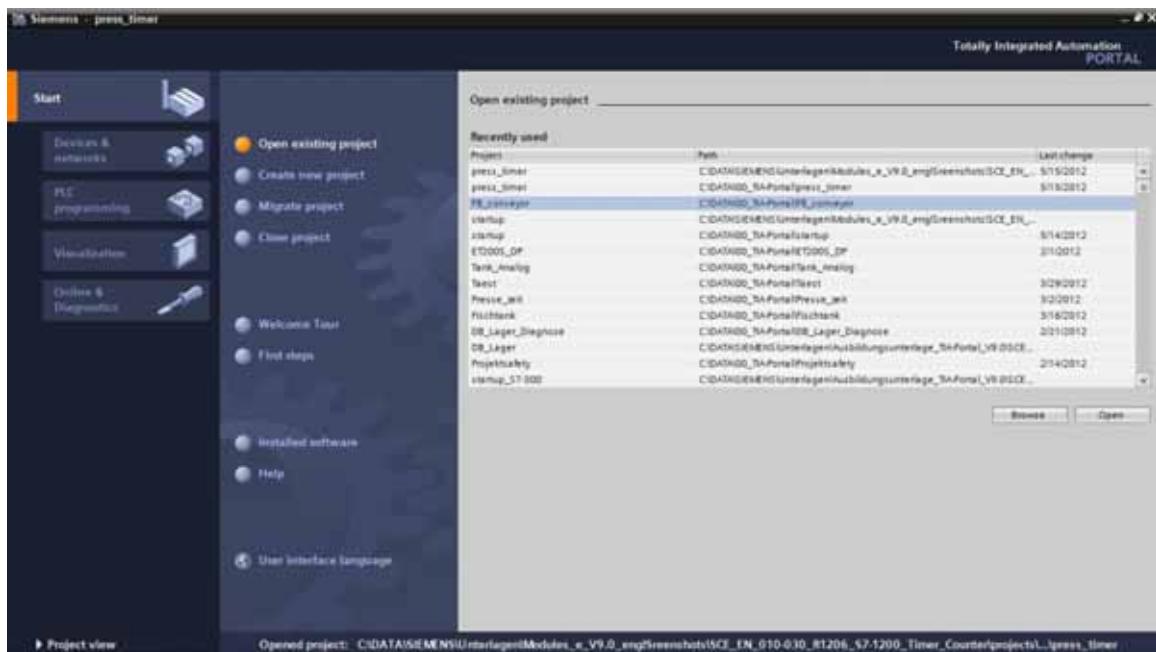
3

In the steps below, for the SIMATIC S7-1200 a project can be opened, stored under a different name and adapted to the new requirement.

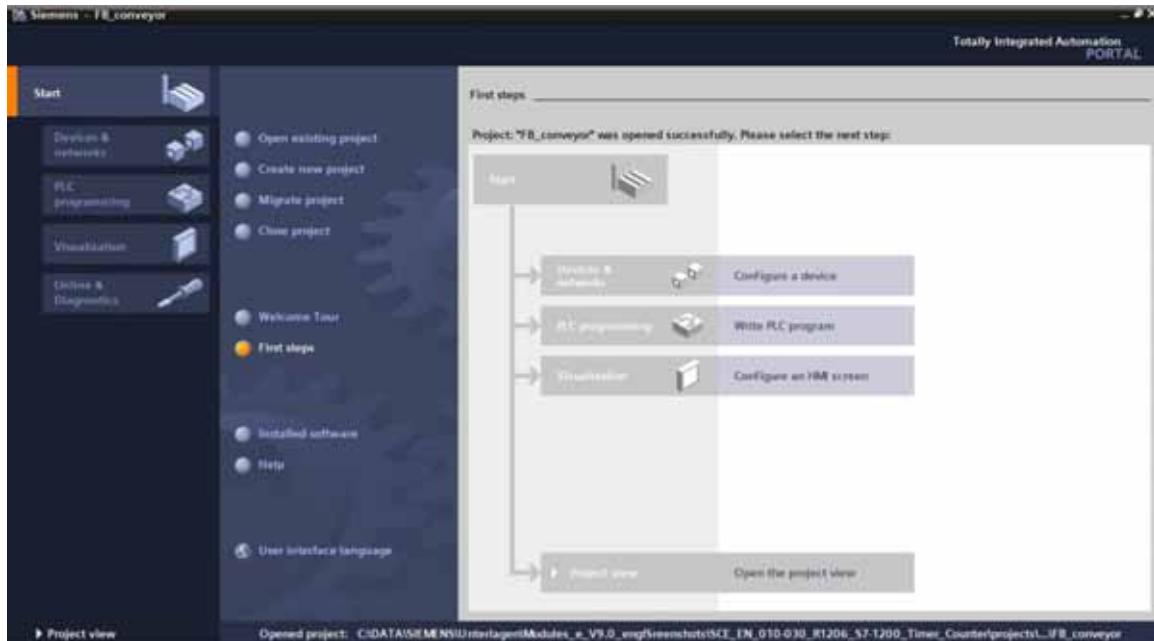
1. The central tool is the '**Totally Integrated Automation Portal**'. Here, we call it with a double click. (→ Totally Integrated Automation Portal V11)



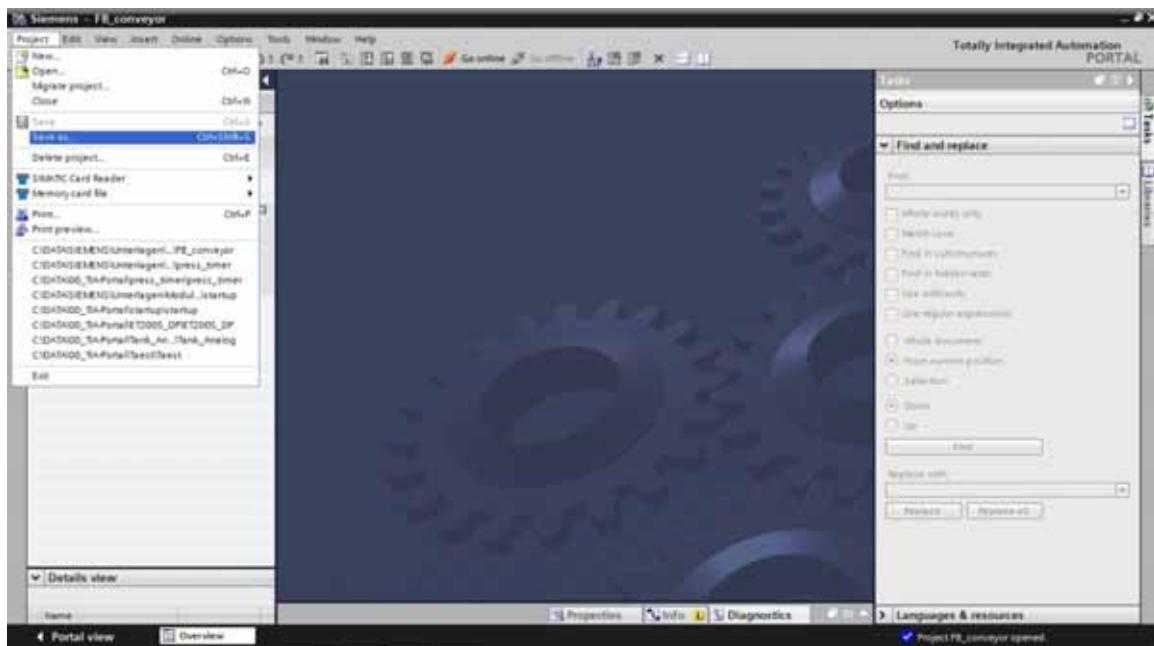
2. We now open the project "**FB_conveyor**" from Module 010-020 in the portal view as the basis for this program. (→ Open existing project → FB_conveyor → Open)



3. Next, '**First Steps**' are suggested for the configuration. We want to '**Open project view**'. (→ Open project view)

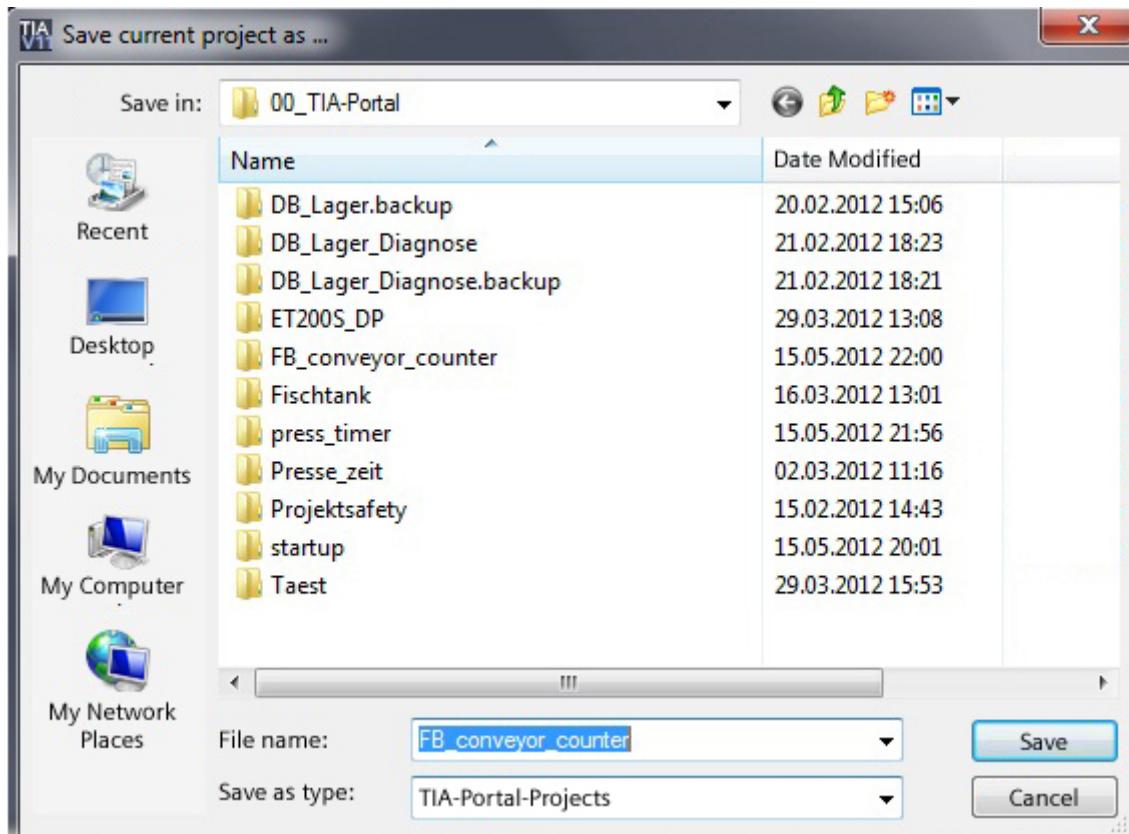


4. Now, we first save the project under a different name. (→ Project → Save as)



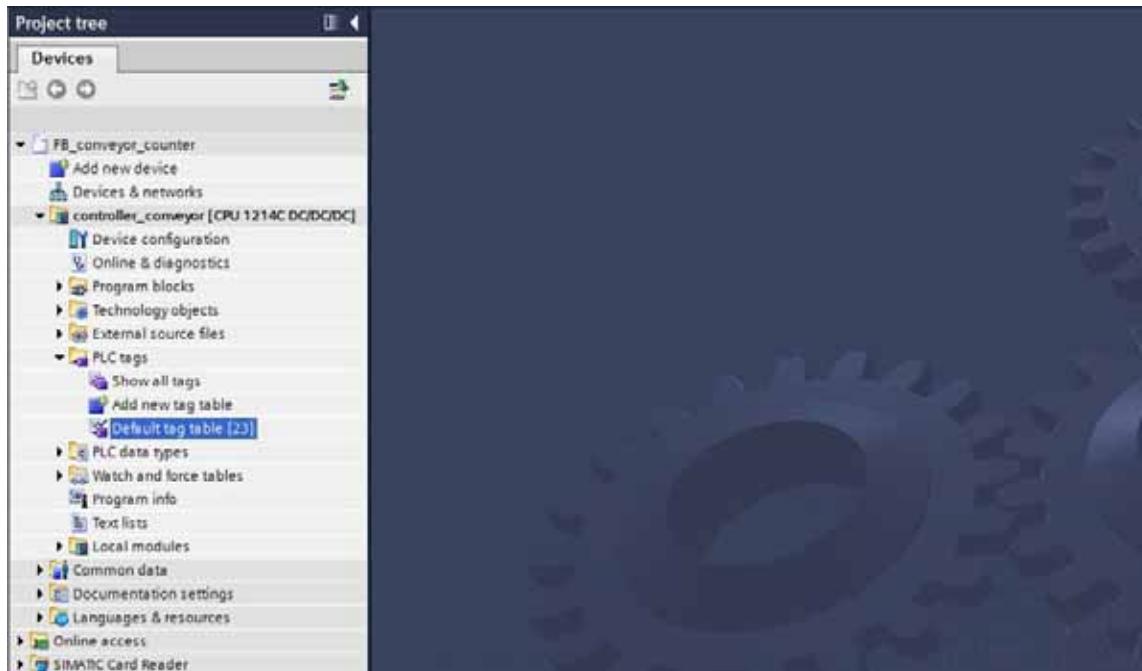
3

5. Next, '**Save**' the project under the new name '**FB_conveyor_counter**'. (→ FB_conveyor_counter → Save)



3

6. To set up new global variables, open with a double click on '**PLC tags**' under the '**controller_conveyor**' '**Default tag table**'. (→ controller_conveyor → PLC tags → Default tag table)



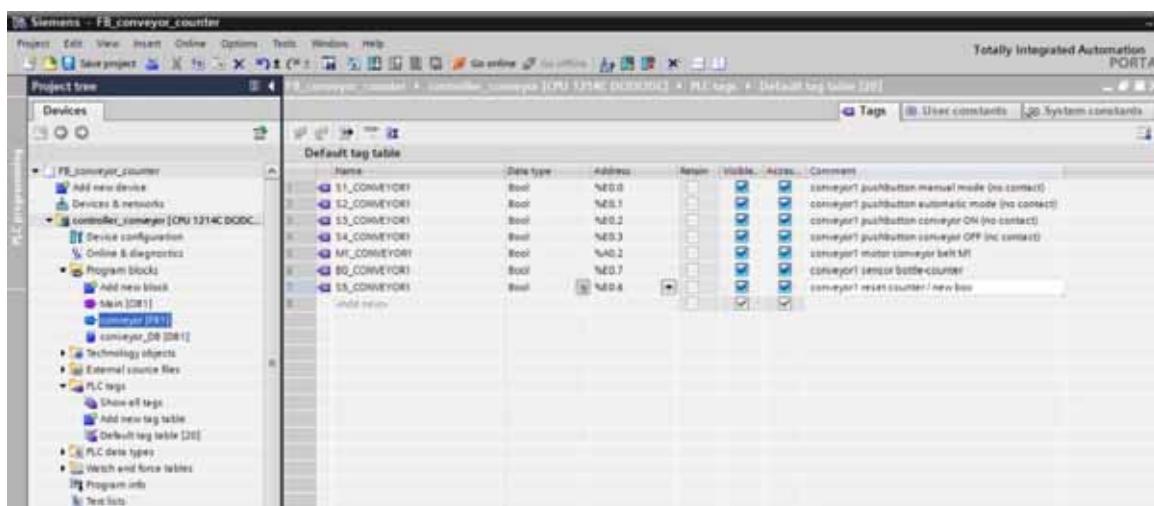
3

7. Change the tag table according to the default setting.

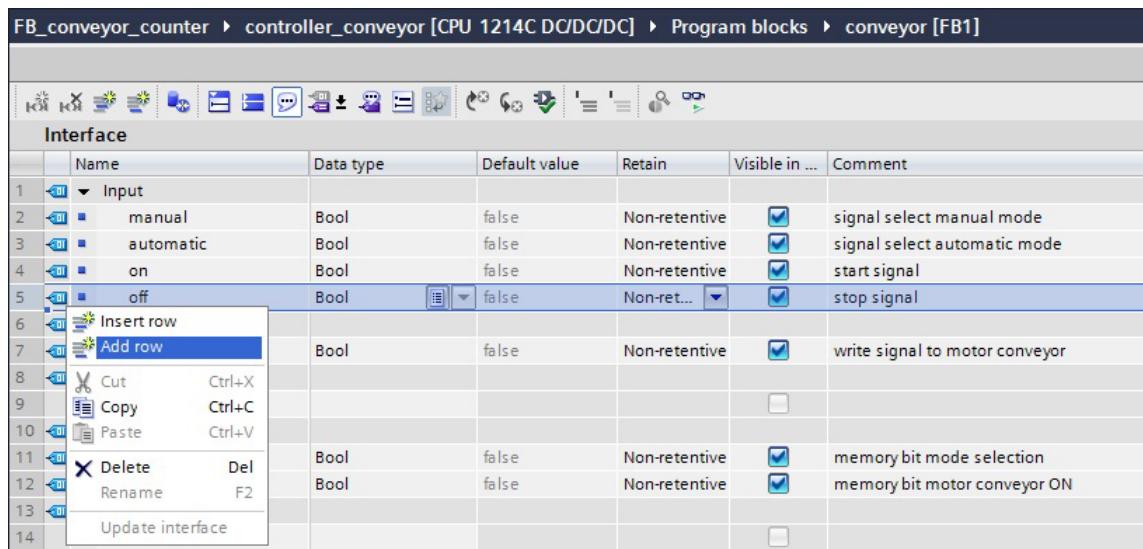
Next, set up the two global variables '**B0**' and '**S5**'. (→ B0/Bool/%I0.7/sensor bottle counter → S5/Bool/%I0.6/reset counter/new box)

	Name	Data type	Address	Retain	Visible..	Access...	Comment
1	S1_CONVEYOR1	Bool	%E0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 pushbutton manual mode (no contact)
2	S2_CONVEYOR1	Bool	%E0.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 pushbutton automatic mode (no contact)
3	S3_CONVEYOR1	Bool	%E0.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 pushbutton conveyor ON (no contact)
4	S4_CONVEYOR1	Bool	%E0.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 pushbutton conveyor OFF (nc contact)
5	M1_CONVEYOR1	Bool	%A0.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 motor conveyor belt M1
6	B0_CONVEYOR1	Bool	%E0.7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 sensor bottle-counter
7	S5_CONVEYOR1	Bool	%E0.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	conveyor1 reset counter / new box

8. To make the changes in the program, open the block 'conveyor[FB1]**' with a double click.
(→ conveyor[FB1])**



9. First, add two rows under Interface for the input variables. (→ Interface → Input → Add row)



The screenshot shows the SIMATIC Manager interface with the title bar "FB_conveyor_counter > controller_conveyor [CPU 1214C DC/DC/DC] > Program blocks > conveyor [FB1]". Below the title bar is a toolbar with various icons. The main area is a table titled "Interface" with columns: Name, Data type, Default value, Retain, Visible in ..., and Comment. Rows 1 through 5 represent input variables: "manual" (Bool, false, Non-retentive, checked), "automatic" (Bool, false, Non-retentive, checked), "on" (Bool, false, Non-retentive, checked), and "off" (Bool, false, Non-retentive, checked). Row 6 is a separator row with a plus sign icon. Row 7 is currently selected and has a blue background, showing "Add row" in the context menu. Rows 8 through 14 are empty rows for memory bit mode selection and motor conveyor ON. A context menu is open over row 6, listing options: Insert row, Add row, Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V), Delete (Del), Rename (F2), and Update interface.

	Name	Data type	Default value	Retain	Visible in ...	Comment
1	Input					
2	manual	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	signal select manual mode
3	automatic	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	signal select automatic mode
4	on	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	start signal
5	off	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	stop signal
6	Insert row					
7	Add row					
8	Cut	Ctrl+X				
9	Copy	Ctrl+C				
10	Paste	Ctrl+V				
11	Delete	Del				
12	Rename	F2				
13	Update interface					
14						

10. When declaring the local variables, we are adding the following variables.

Input:

sensor_bottle
reset_counter

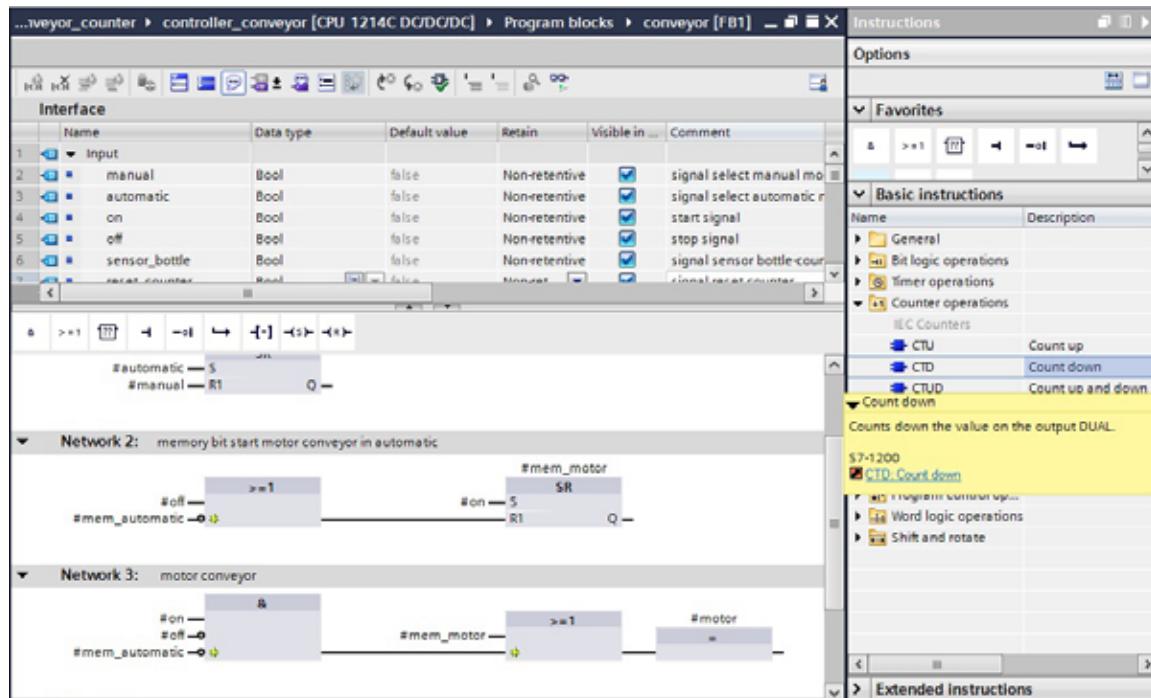
Here, the sensor of the bottle counter is polled
Here, the signal for resetting the counter is entered

3

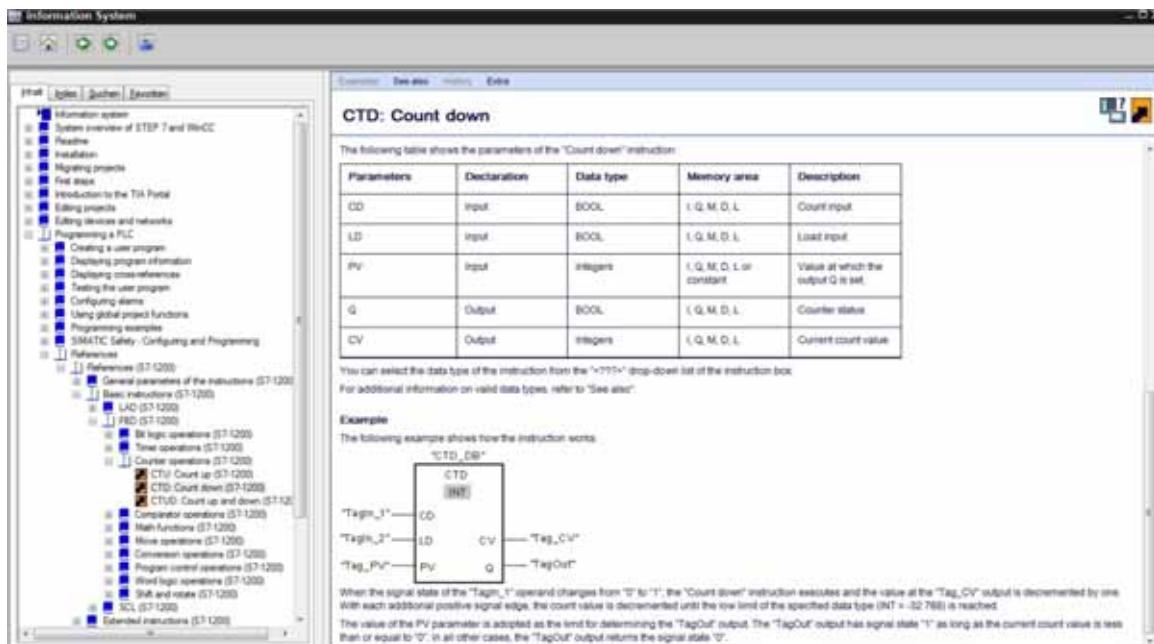
Interface						
	Name	Data type	Default value	Retain	Visible in ...	Comment
1	Input					
2	manual	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	signal select manual mode
3	automatic	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	signal select automatic mode
4	on	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	start signal
5	off	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	stop signal
6	sensor_bottle	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	signal sensor bottle-counter
7	reset_counter	Bool	<input style="width: 20px; height: 20px; border: none; background-color: #f0f0f0; font-size: small; font-weight: bold; padding: 0 5px;" type="button" value="..."/> false	Non-retentive	<input checked="" type="checkbox"/>	signal reset counter
8	<Add new>				<input type="checkbox"/>	
9	Output					
10	motor	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	write signal to motor conveyor
11	InOut				<input type="checkbox"/>	
12	<Add new>				<input type="checkbox"/>	
13	Static					
14	mem_automatic	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	memory bit mode selection
15	mem_motor	Bool	false	Non-retentive	<input checked="" type="checkbox"/>	memory bit motor conveyor ON
16	Temp				<input type="checkbox"/>	
17	<Add new>				<input type="checkbox"/>	

11. Now we can start changing the program.

As we generate our solution with the counter, we need a down counter 'CTD'. It is located under '**Instructions**' in the folder '**Counters**'. If you point with the mouse to an object such as the counter CTD, you will be provided with detailed information about this object. (→ Instructions → Counters → CTD)



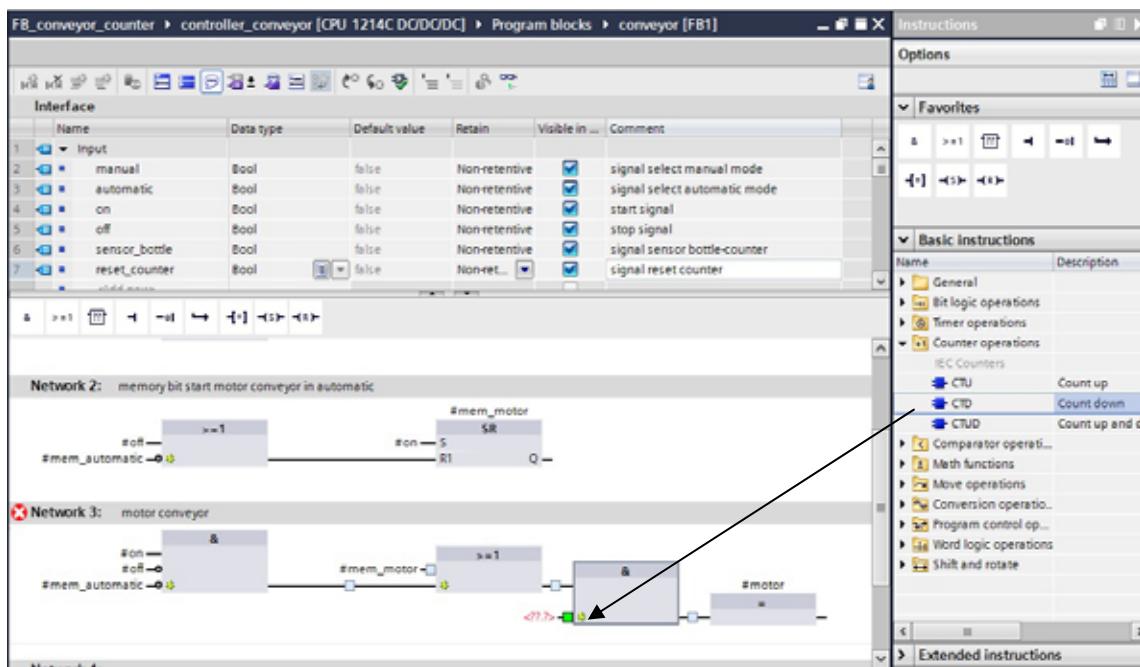
12. If you highlight an object and then press the '**F1**' key on your PC, online help regarding this object is displayed in a window to the right. (→ F1)



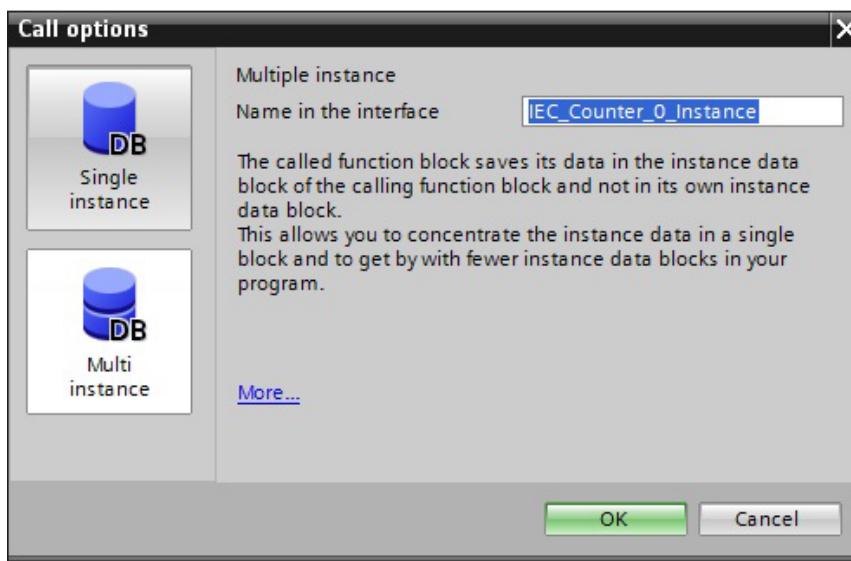
Note

Here, go to online help to obtain the details about all counters.

13. Now, first insert an AND between OR and assignment, and then drag the counter 'CTD' to the second contact of the AND function.
- (→ & → CTD)



14. We need memory for the counter function. Here, the function block makes it available within the instance data block as '**Multi-Instance**', without generating a new instance data block. (→ Multi-Instance → OK)

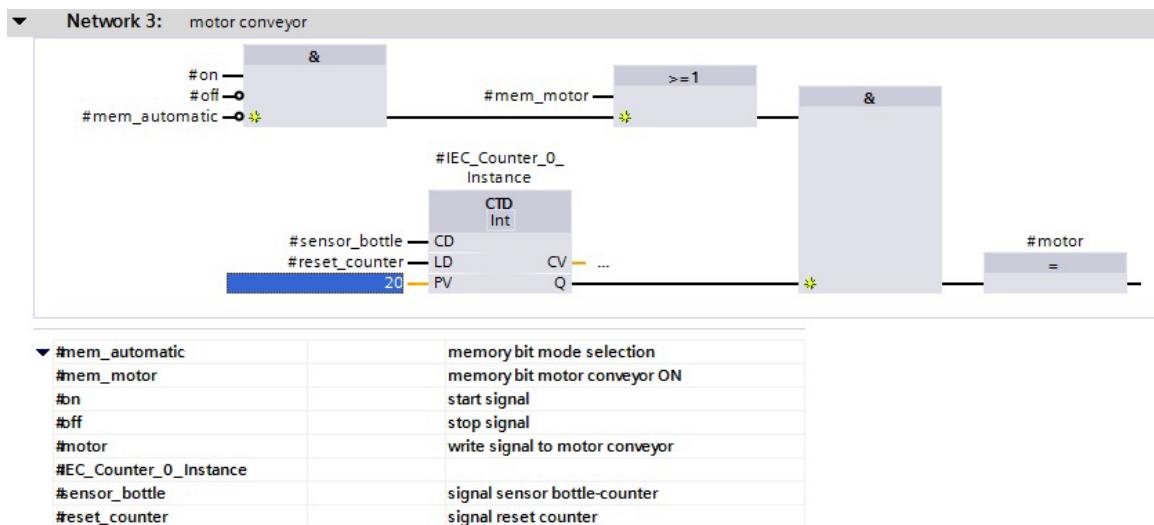


Note

A multi-instance can be used only when programming within a function block.

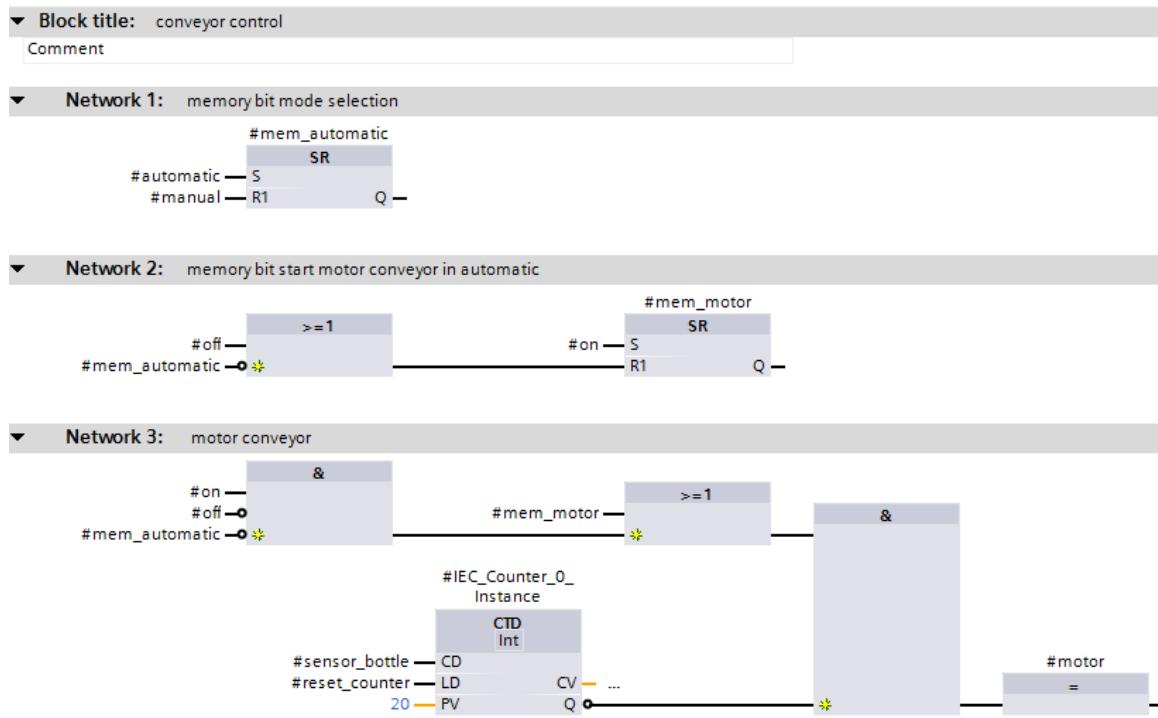
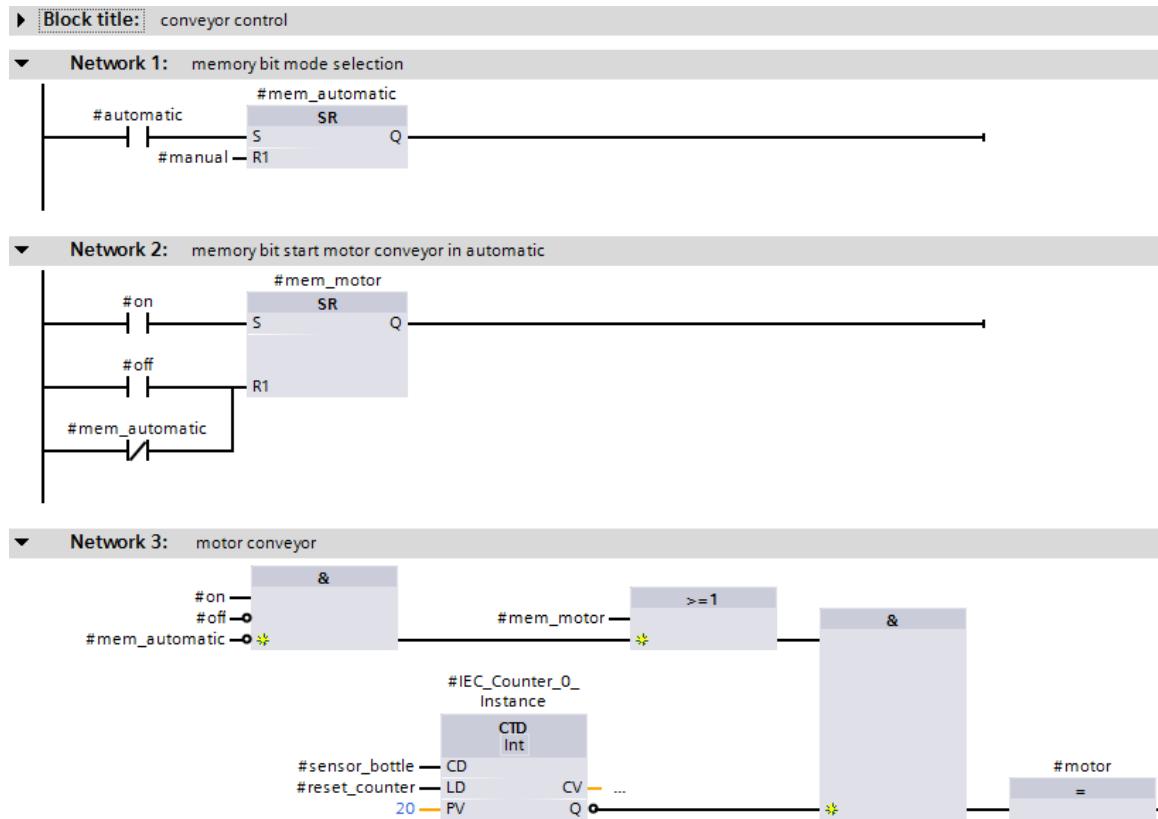
15. Now, connect the down counter '**CTD**' to the specified value '**PV**' for the 20 bottles and connect the input '**CD**' to '**#sensor_bottle**', and the input '**LD**' to '**#reset_counter**'. Next, negate the second contact of the AND function.

Click on  **Save project** and the project will be saved. ($\rightarrow 20 \rightarrow \#sensor_bottle \rightarrow \#reset_counter$)

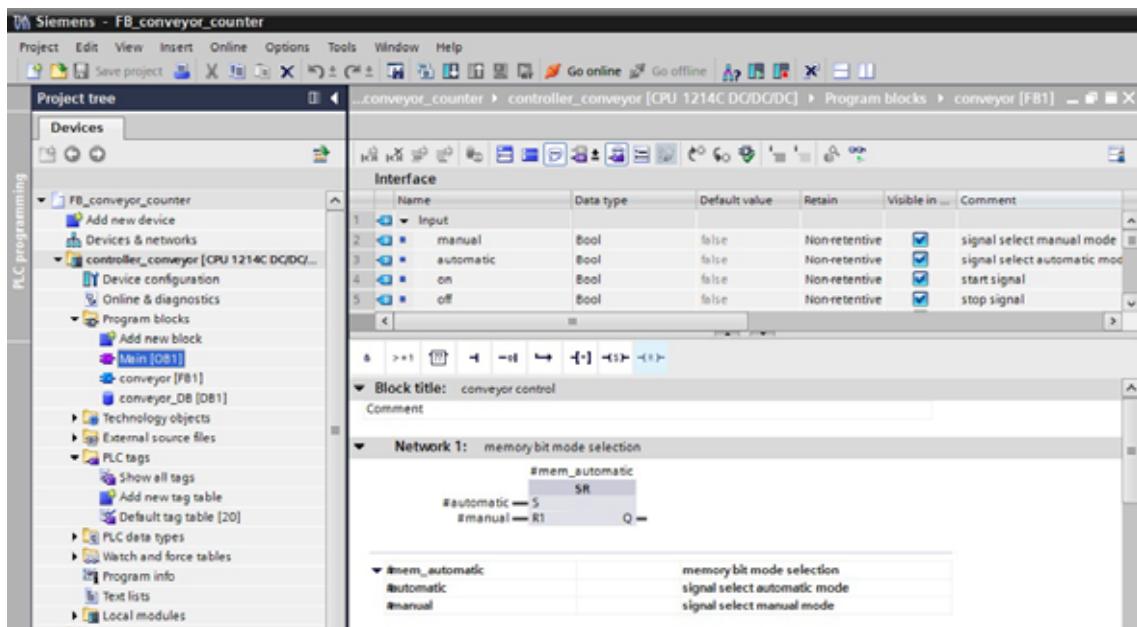


Note

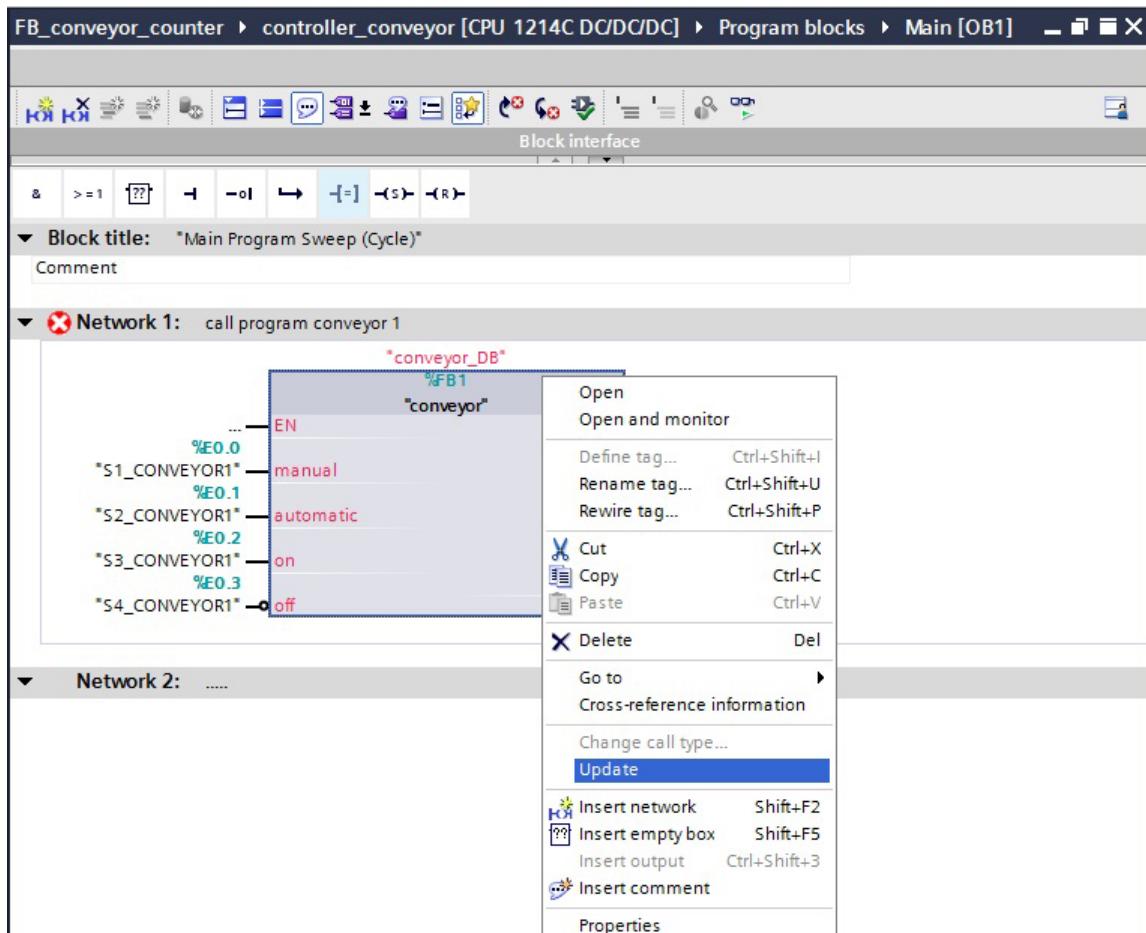
The down counter is most suitable for counting specified quantities, since simply the binary output '**Q**' can be used for further connections. Otherwise, a comparator would have to be programmed.

Program in function block diagram (FBD)**Program in ladder diagram (LAD)**

16. Now, open the block '**Main[OB1]**' to update the call of block '**conveyor[FB1]**' (→ Main[OB1])

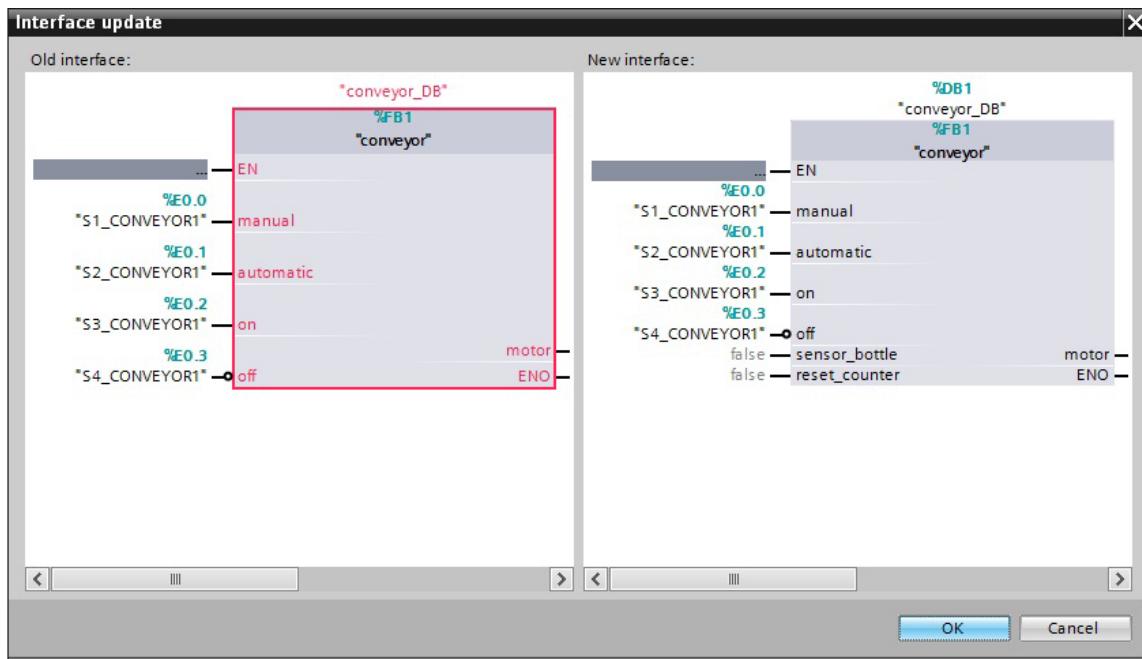


17. In the block '**Main[OB1]**', click with the right mouse key on "conveyor" and then on 'Update'.
 (→ Main[OB1] → Update)

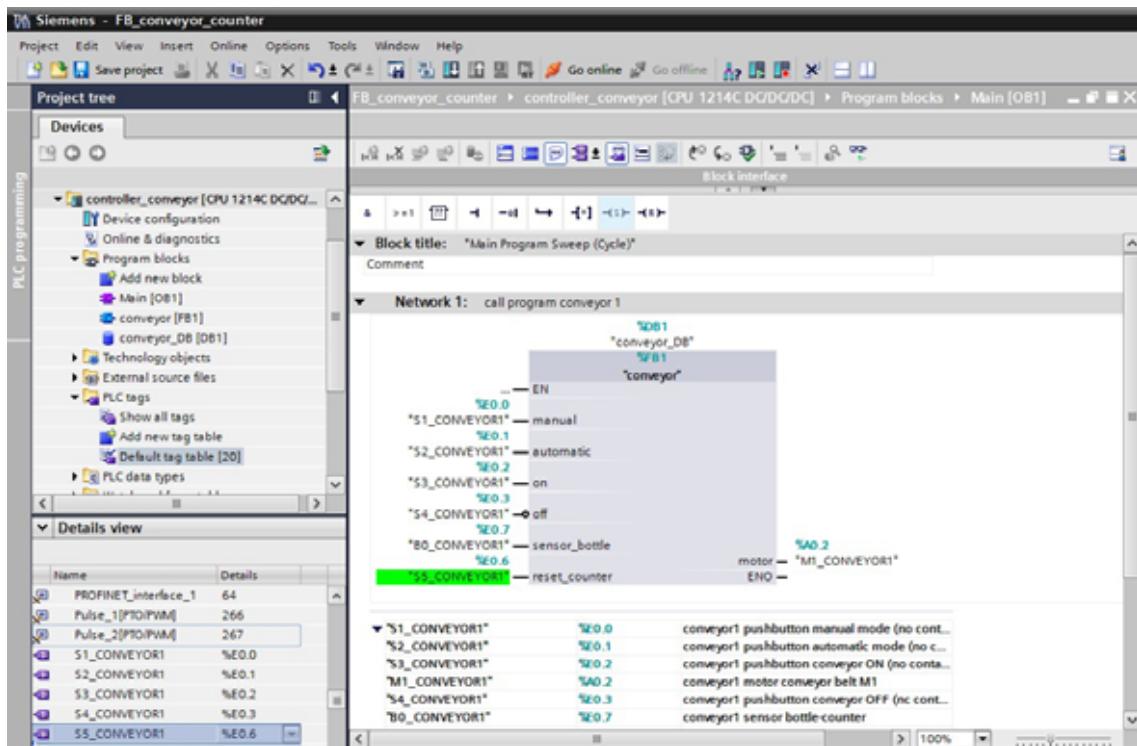


3

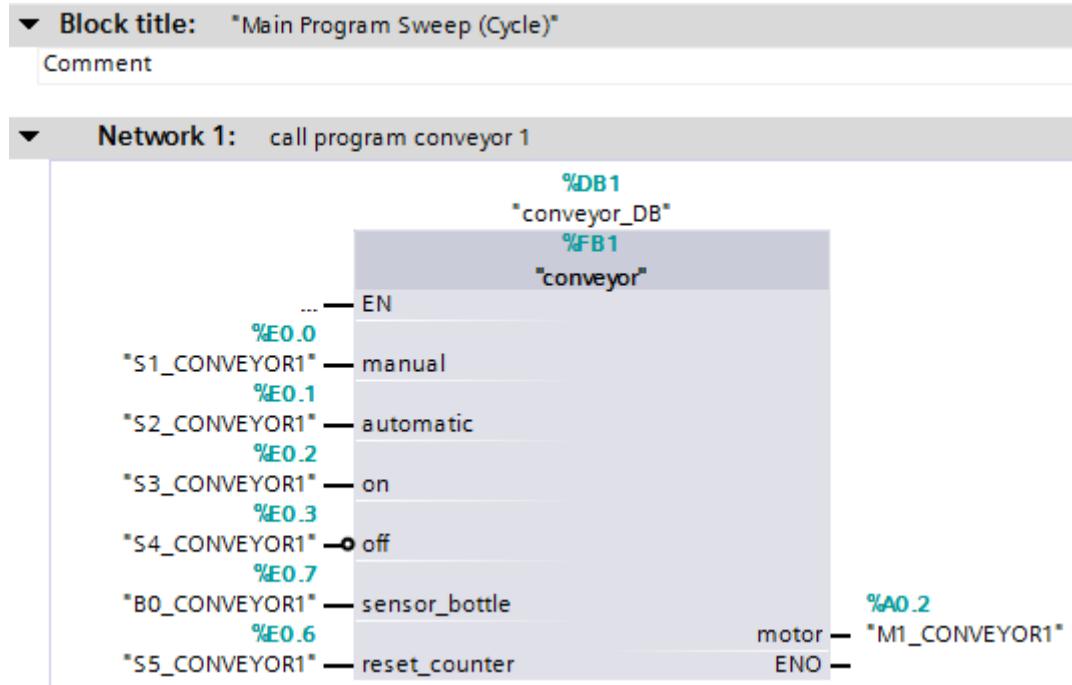
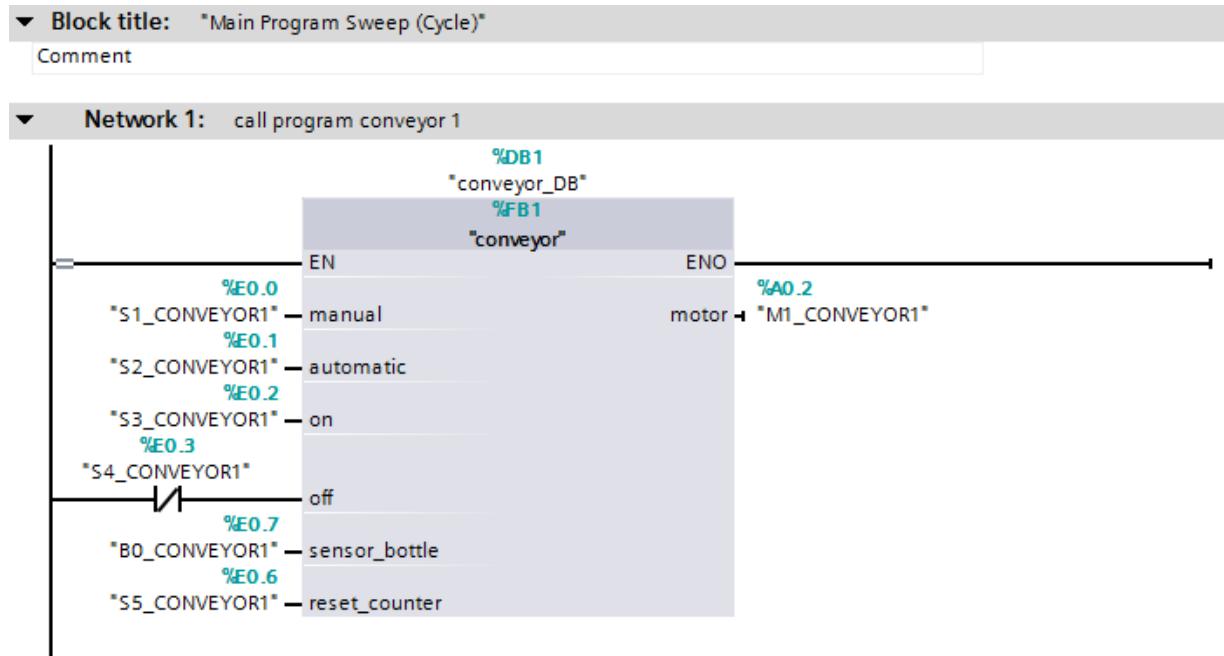
18. Next, select the '**New Interface**' and confirm with '**OK**'. (→ New interface → OK)



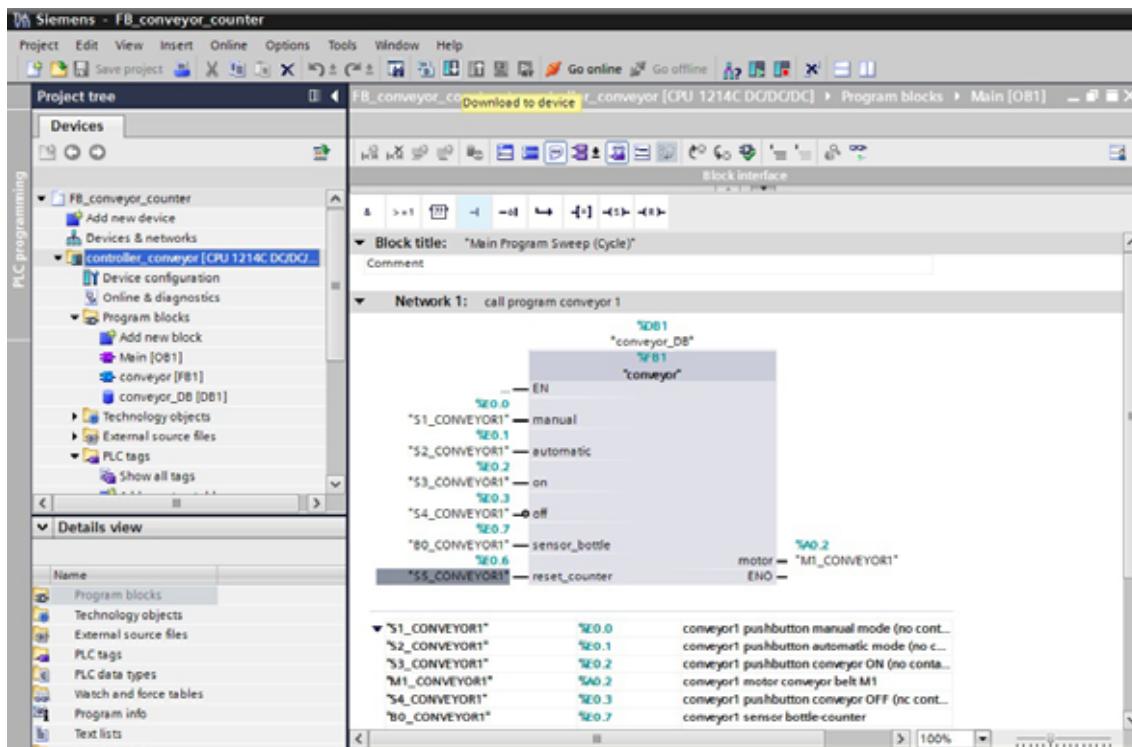
19. Now connect the two new input variables to the PLC tags “B0” and “S5” shown here. Then click on Save project, and the project will be saved. (→ Save project)



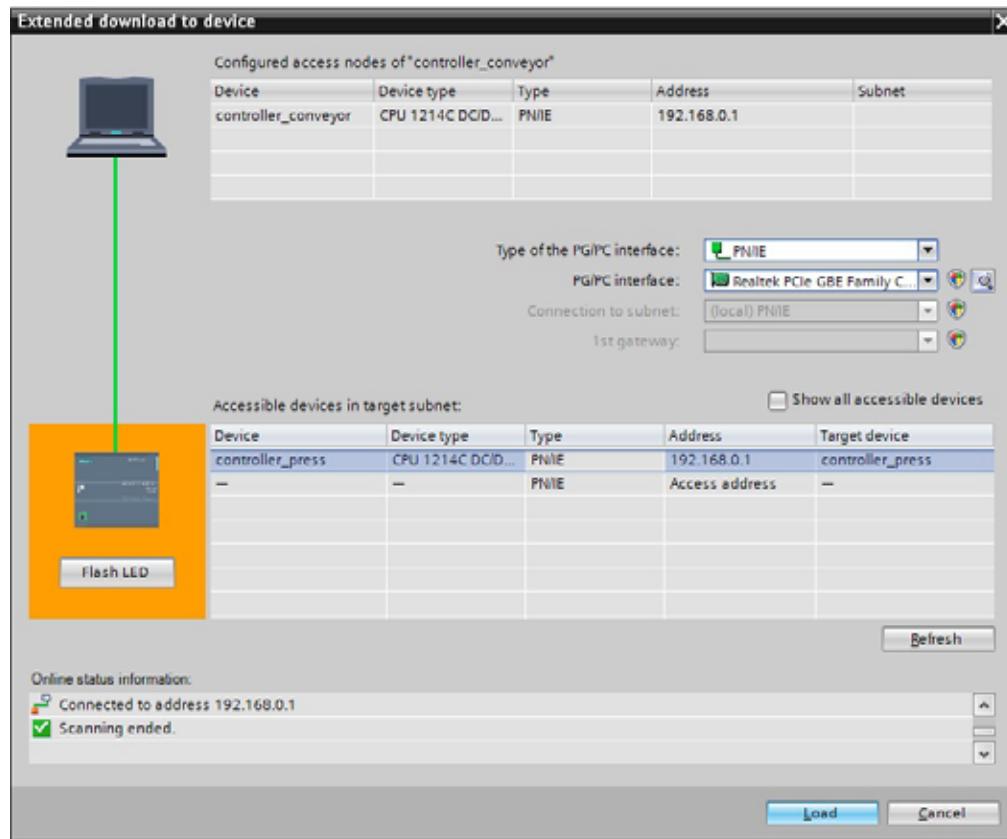
3

Program in function block diagram (FBD)**Program in ladder diagram (LAD)**

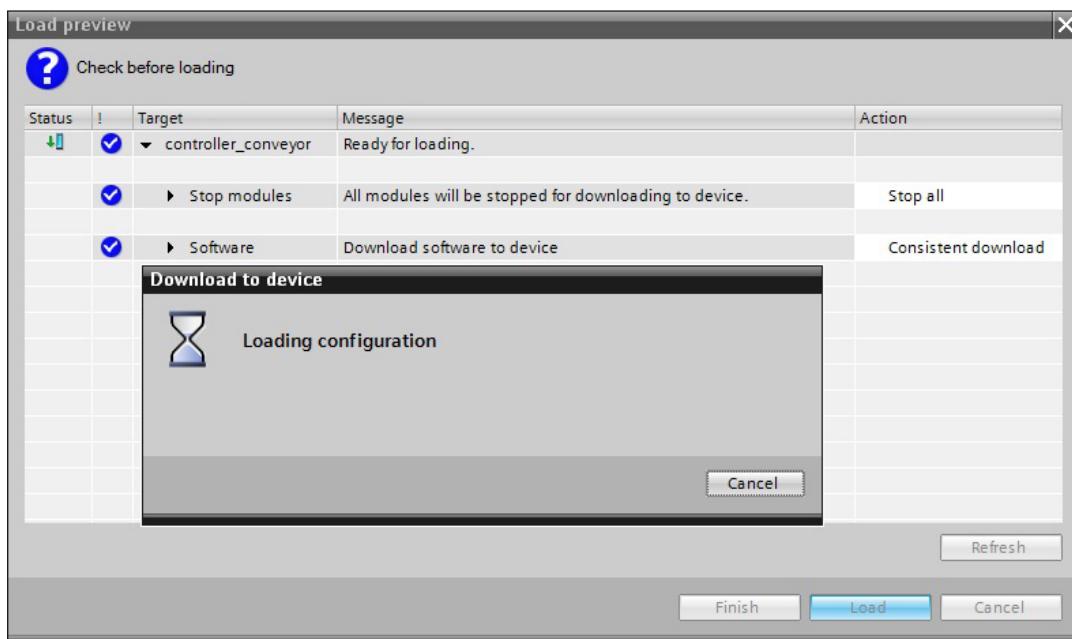
20. To load your entire program into the CPU, highlight the folder '**controller_conveyor**', and then click on the symbol Load to device. (→ controller_conveyor →



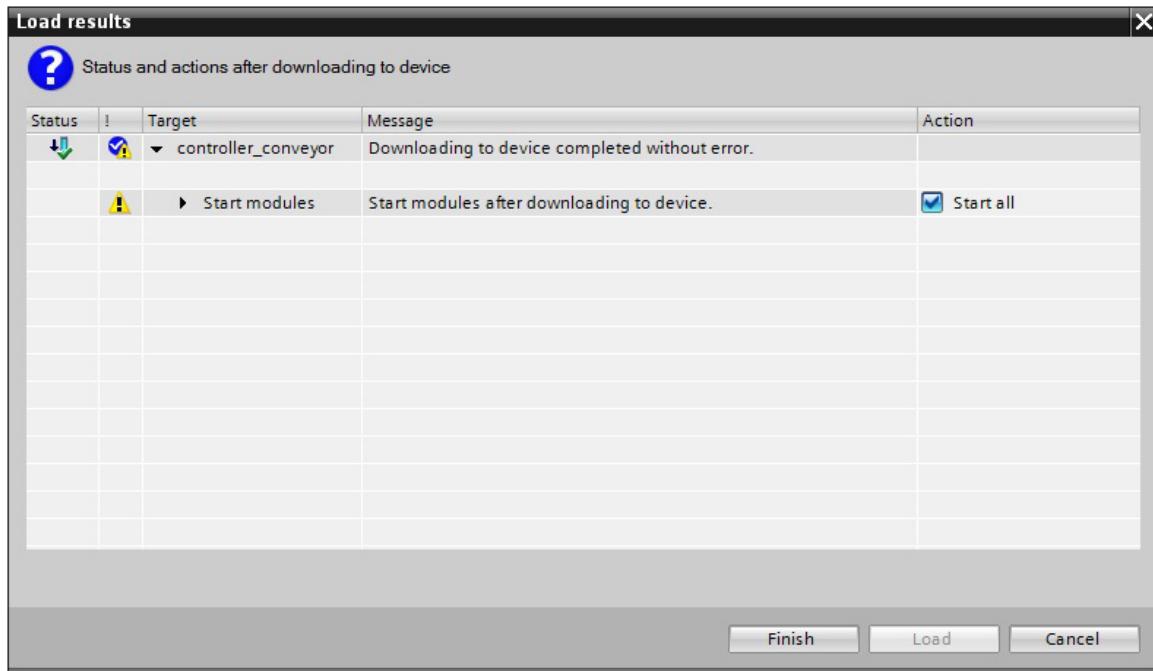
21. Setting the Interface



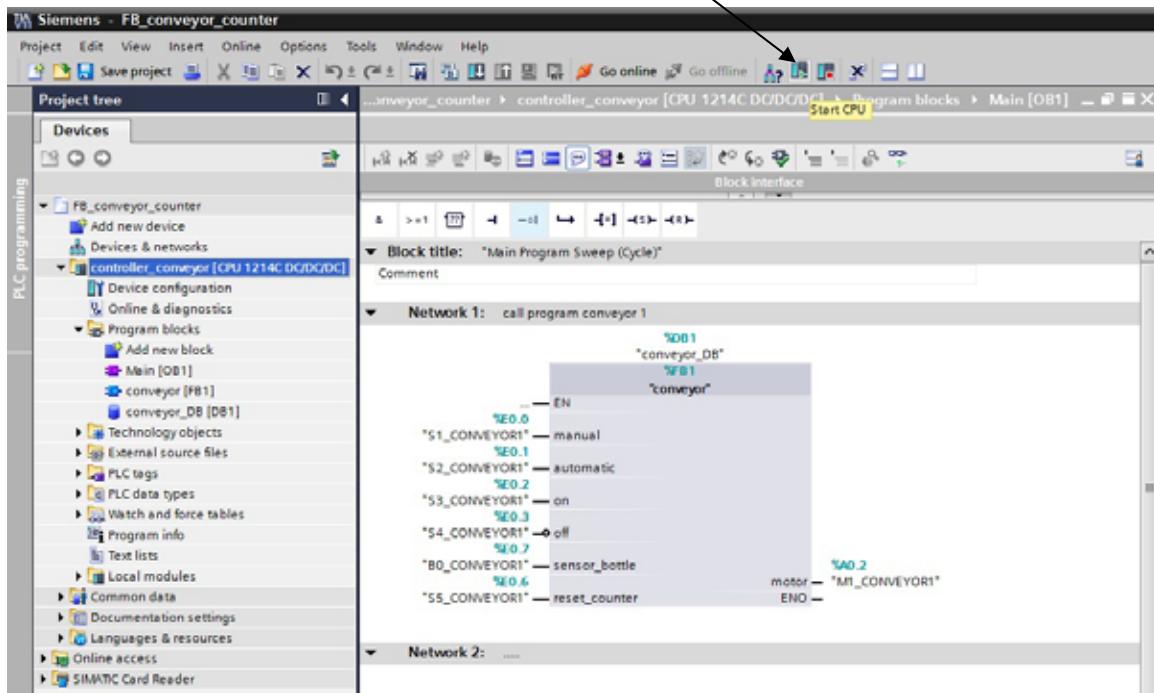
22. Confirm 'Load' once more. During loading, the status is displayed in a window. (→ Load)



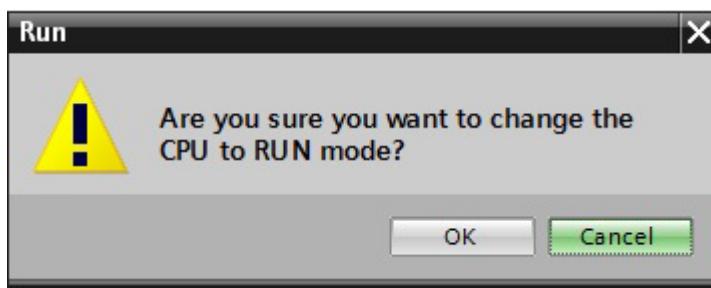
23. If loading was successful, it is displayed in a window. Click on '**Finish**'. (→ Finish)



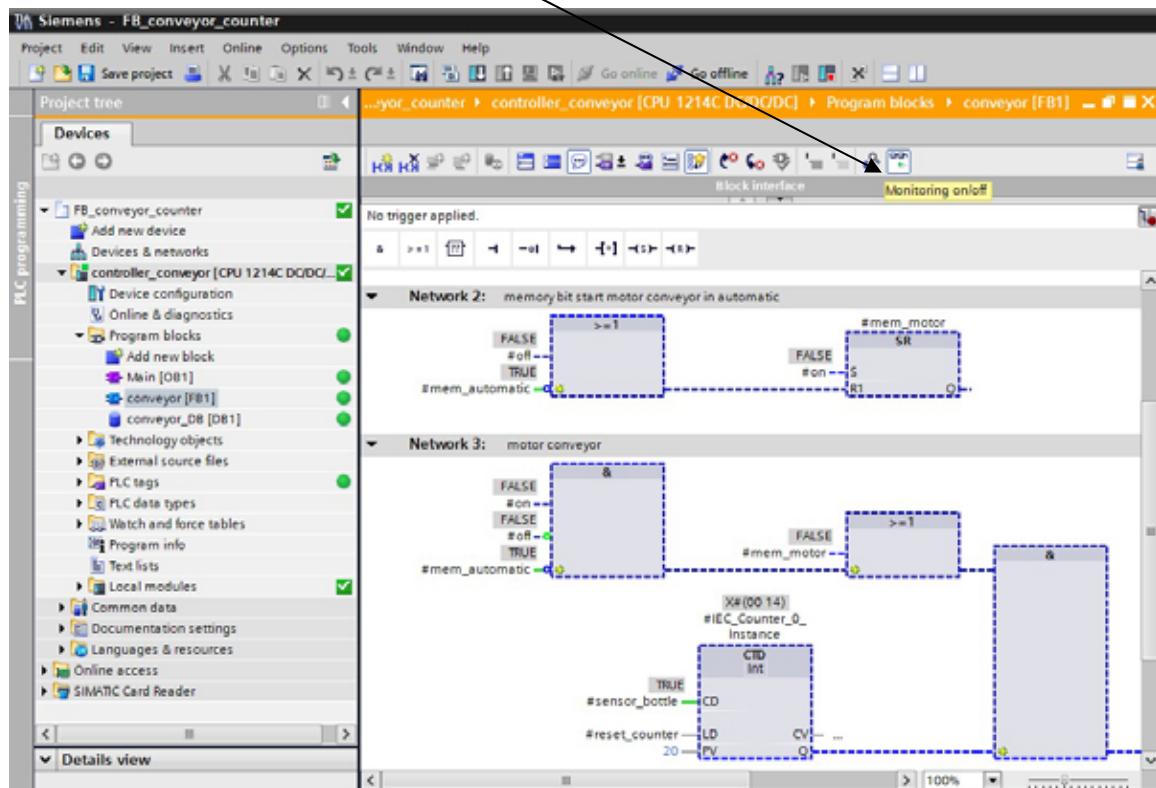
24. Next, start the CPU by clicking on the symbol .



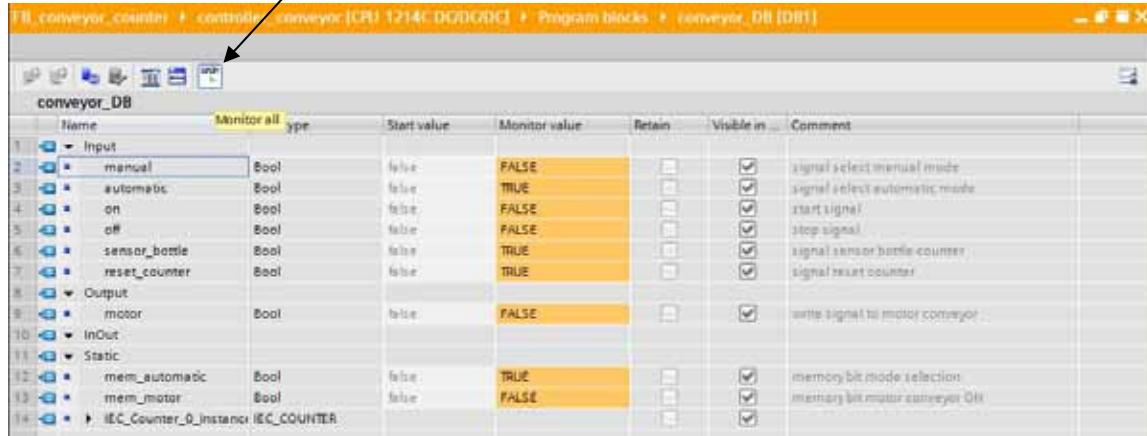
25. Confirm the question whether you actually want to start the CPU with '**OK**'. (→ OK)



26. By clicking on the symbol  Monitoring On/Off, you can observe the counter status while the program is tested.



27. By clicking on the symbol  Monitoring On/Off, you can observe the open data block while the program is tested.



Name	Monitor all	Type	Start value	Monitor value	Retain	Visible in...	Comment
1 Input							
2 manual	Bool	false	FALSE			<input checked="" type="checkbox"/>	signal select manual mode
3 automatic	Bool	false	TRUE			<input checked="" type="checkbox"/>	signal select automatic mode
4 on	Bool	false	FALSE			<input checked="" type="checkbox"/>	start signal
5 off	Bool	false	FALSE			<input checked="" type="checkbox"/>	stop signal
6 sensor_bottle	Bool	false	TRUE			<input checked="" type="checkbox"/>	signal sensor bottle counter
7 reset_counter	Bool	false	TRUE			<input checked="" type="checkbox"/>	signal reset counter
8 Output							
9 motor	Bool	false	FALSE			<input checked="" type="checkbox"/>	write signal to motor conveyor
10 InOut							
11 Static							
12 mem_automatic	Bool	false	TRUE			<input checked="" type="checkbox"/>	memory bit mode selection
13 mem_motor	Bool	false	FALSE			<input checked="" type="checkbox"/>	memory bit motor conveyor On
14 IEC_Counter_0_Instance IEC_COUNTER							

Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Module 010-040



TIA Portal Module 010-010

Startup Programming of the
SIMATIC S7-1200

TIA Portal Module 010-020

Block Types for the SIMATIC S7-1200

TIA Portal Module 010-030

IEC Timers and IEC Counters at the
SIMATIC S7-1200

TIA Portal Module 010-040

Diagnosis and Error Search at the
SIMATIC S7-1200

TIA Portal Module 010-050

Analog Value Processing with
SIMATIC S7-1200

TIA Portal Module 010-060

Control Engineering with
SIMATIC S7-1200

TIA Portal Module 010-070

Communication between two
SIMATIC S7-1200

TIA Portal Module 010-080

HMI Panel KTP600 with SIMATIC
S7-1200

TIA Portal Module 010-090

Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

Matching SCE training packages for these documents

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
Order number: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
Order number 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
Order number 6ES7822-0AA01-4YE0

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

4

Continued Training

For regional Siemens SCE continued training, please contact your regional SCE contact person
siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding Usage

The training curriculum 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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by 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.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the documents being used commercially.

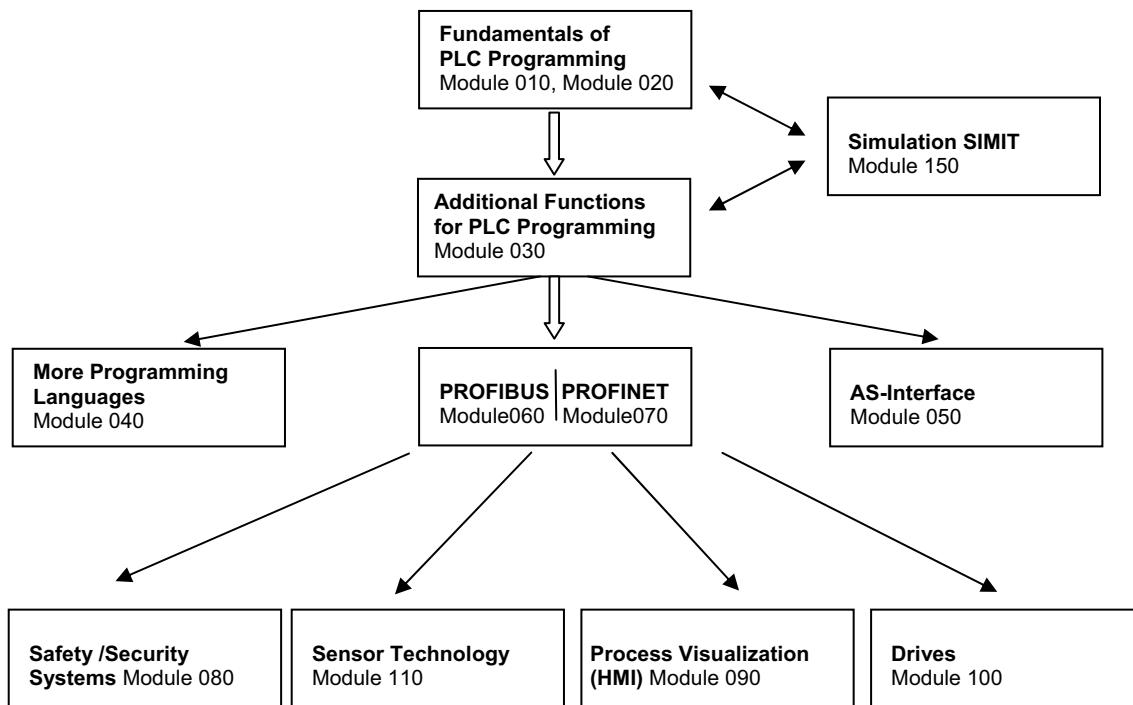
We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this document.

PAGE

1.	Preface	4
2.	Notes on Programming the SIMATIC S7-1200	6
2.1.	Automation System SIMATIC S7-1200	6
2.2.	Programming Software STEP 7 Professional V11 (TIA Portal V11)	6
3.	Diagnosis Functions for the SIMATIC S7-1200	7
3.1.	Device Diagnosis	13
3.2.	Offline/Online Comparison	22
3.3.	Monitoring and Controlling Variables	25
3.4.	Forcing Variables	29
3.5.	Enabling IO Outputs	33

1. PREFACE

Regarding its content, module SCE_EN_010-040 is part of the training unit '**Basics of PLC Programming**', and represents a **fast entry point** for programming the SIMATIC S7 1200 with TIA Portal.



Training Objective:

This module SCE_EN_010-040 acquaints the reader with the tools that are helpful when searching for errors.

To this end, test and online functions are introduced that can be tested with the TIA project 'Startup' in Module 010-010_R1201, for example

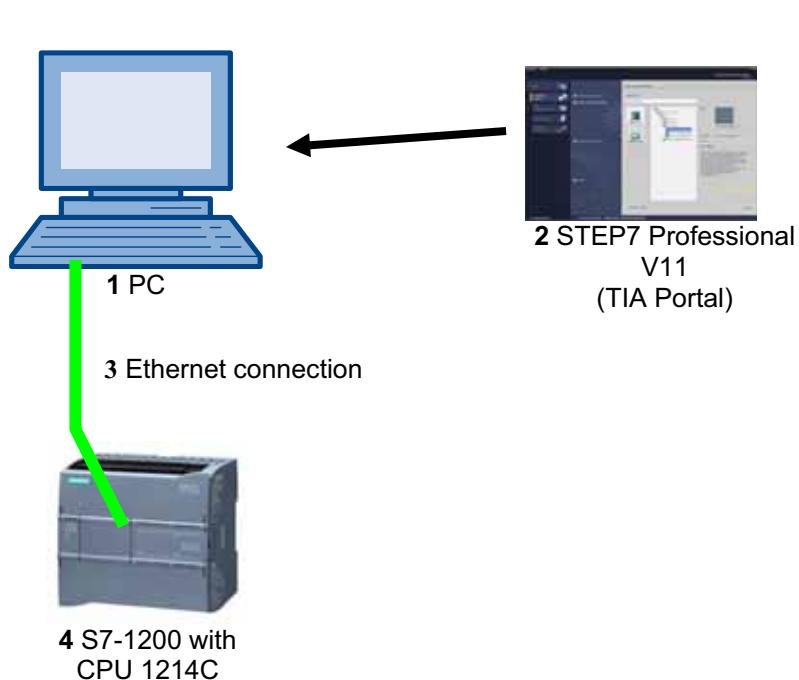
Prerequisites:

To successfully work through this module, the following knowledge is assumed:

- How to operate Windows
- Basics of PLC programming with the TIA Portal (for example, module 010-010 – 'Startup' Programming the SIMATIC S7-1200 with TIA Portal V11)

Hardware and software required

- 1 PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB, operating system Windows XP Professional SP3//Windows 7 Professional/Windows 7 Enterprise/Windows 7 Ultimate/Windows 2003 Server R2/Windows Server 2008
- 2 Software STEP7 Professional V11 SP1 (Totally Integrated Automation (TIA) Portal V11)
- 3 Ethernet connection between PC and CPU 315F-2 PN/DP
- 4 PLC SIMATIC S7-1200; for example, CPU 1214C.
The inputs have to be brought out to a panel.



2. NOTES ON PROGRAMMING THE SIMATIC S7-1200

2.1. Automation System SIMATIC S7-1200

The SIMATIC S7-1200 automation system is a modular mini-control system for the medium and lower performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU and input and output modules for digital and analog signals..

If necessary, communication processors and function modules for special tasks such as step motor control are added.

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process; the IO modules are polled in the S7 program by means of the input addresses (%I), and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7 .

2.2. Programming Software STEP 7 Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 is the programming tool for the automation systems

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating visual displays for the SIMATIC basic panels with integrated WinCC Basic
- With additional WinCC packages, visualization solutions for PCs and other panels can be generated

All functions are supported with detailed online help.

3. DIAGNOSIS FUNCTIONS FOR THE SIMATIC S7-1200

Below, test and online functions are introduced that you can test with the TIA_ Portal project 'Startup' in module 010-010, for example.

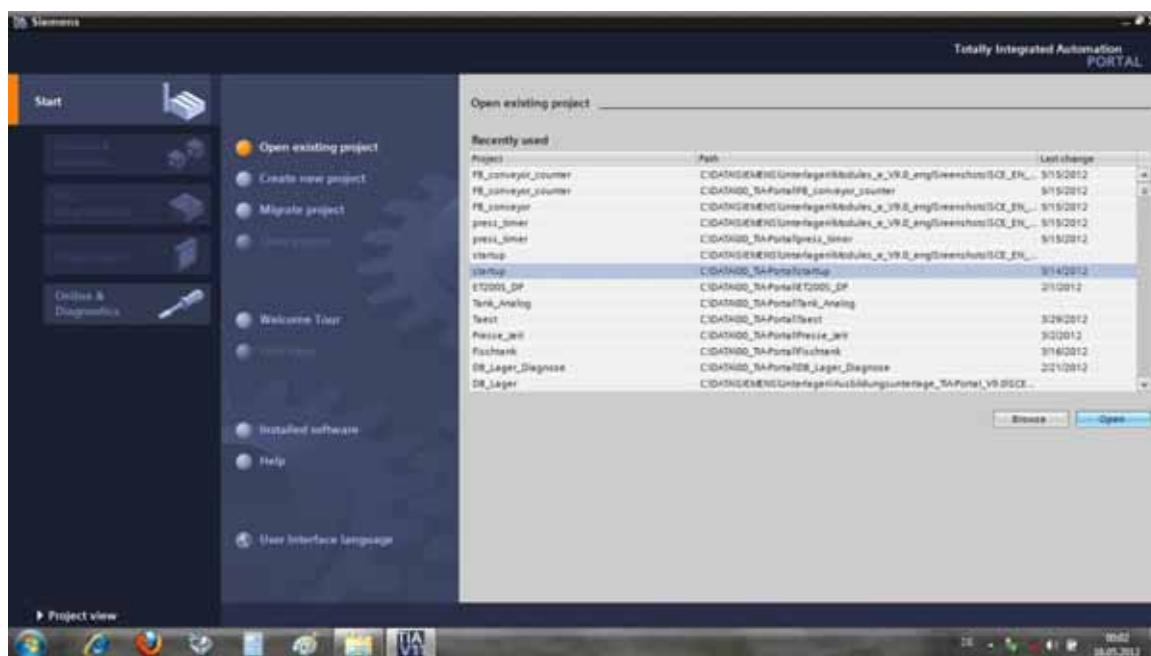
To this end, first load the program '**startup**' to the controller.

1. The central tool is the '**Totally Integrated Automation Portal**'. Here, it is called with a double click. (→ Totally Integrated Automation Portal V11)

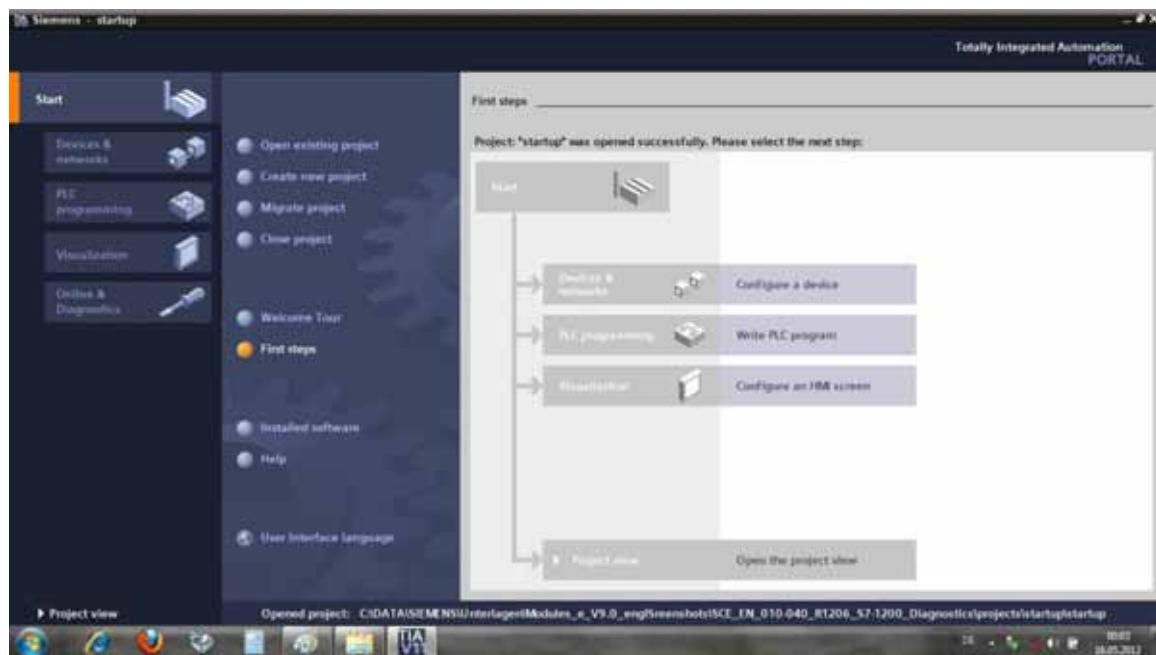


4

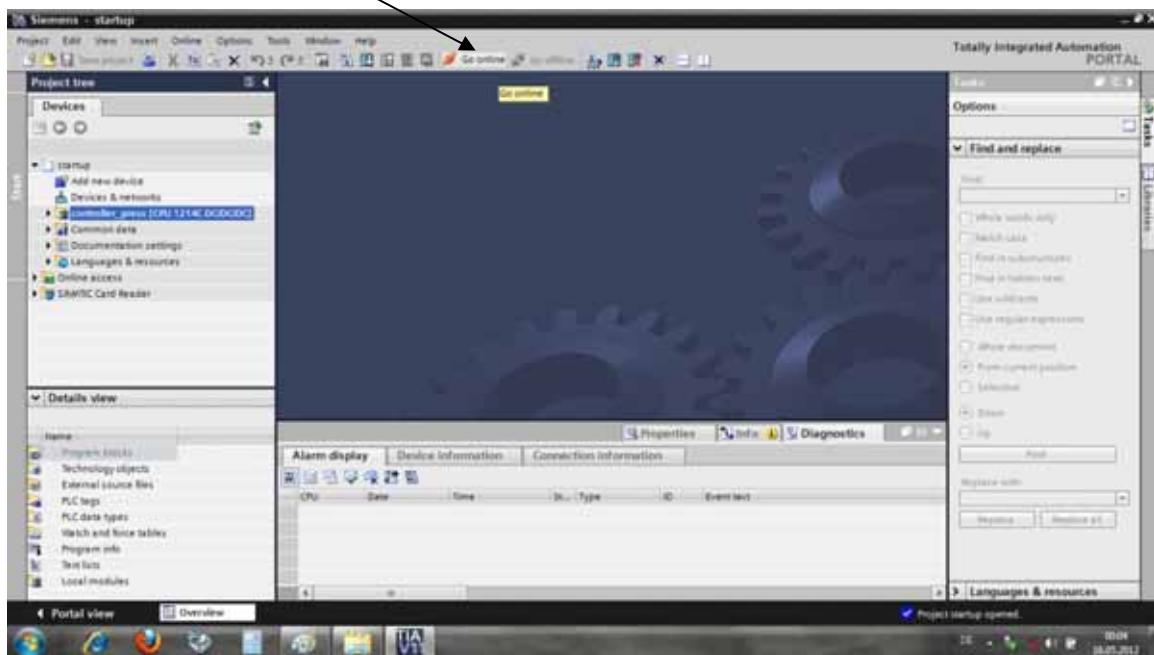
2. Now, we open the project "**startup**" from Module 010-010 in the portal view as the basis for this program. (→ Open existing project → startup → Open)



3. Next, 'First Steps' for the configuration are suggested. We are going to '**Open project view**'. (→ Open project view)

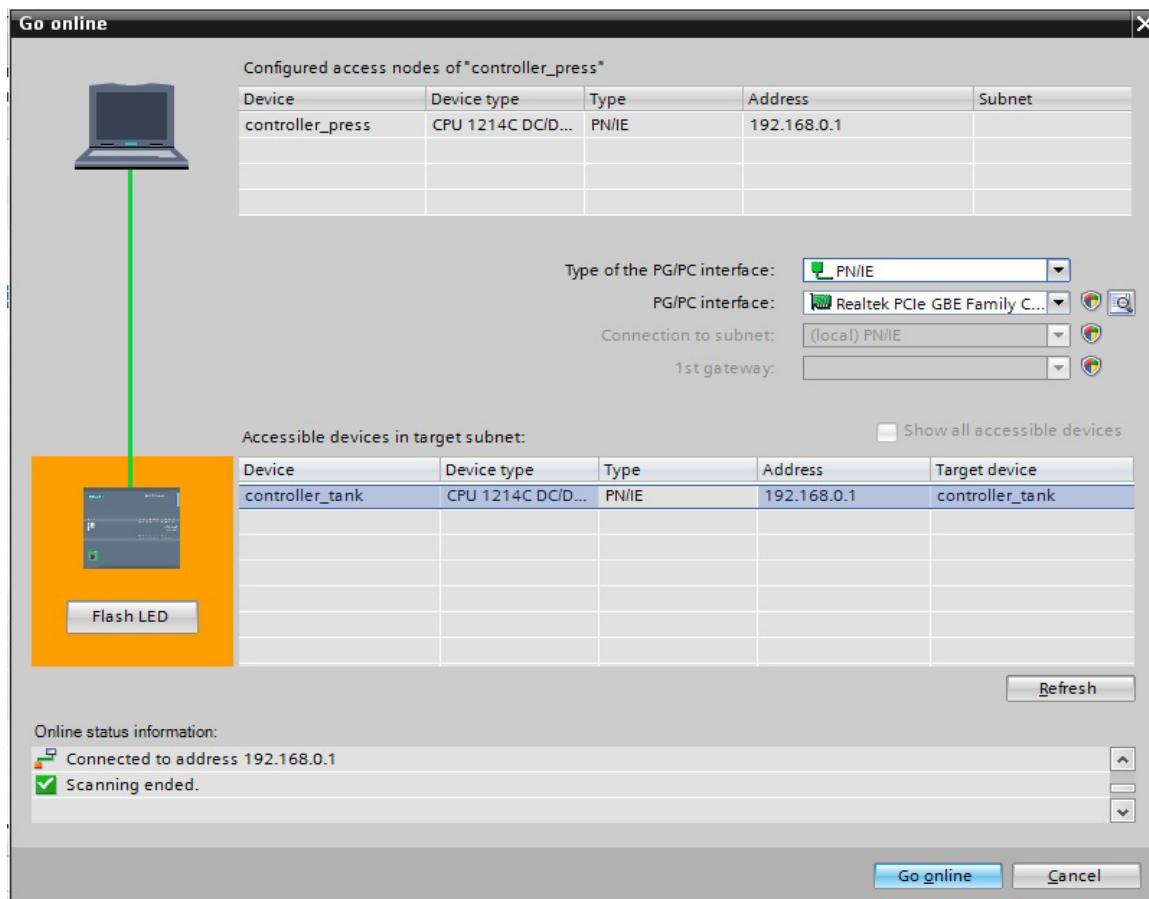


4. To get started with the error search, we now select our '**controller press**' and then '**Go online**'.
 (→ controller press → Go online)



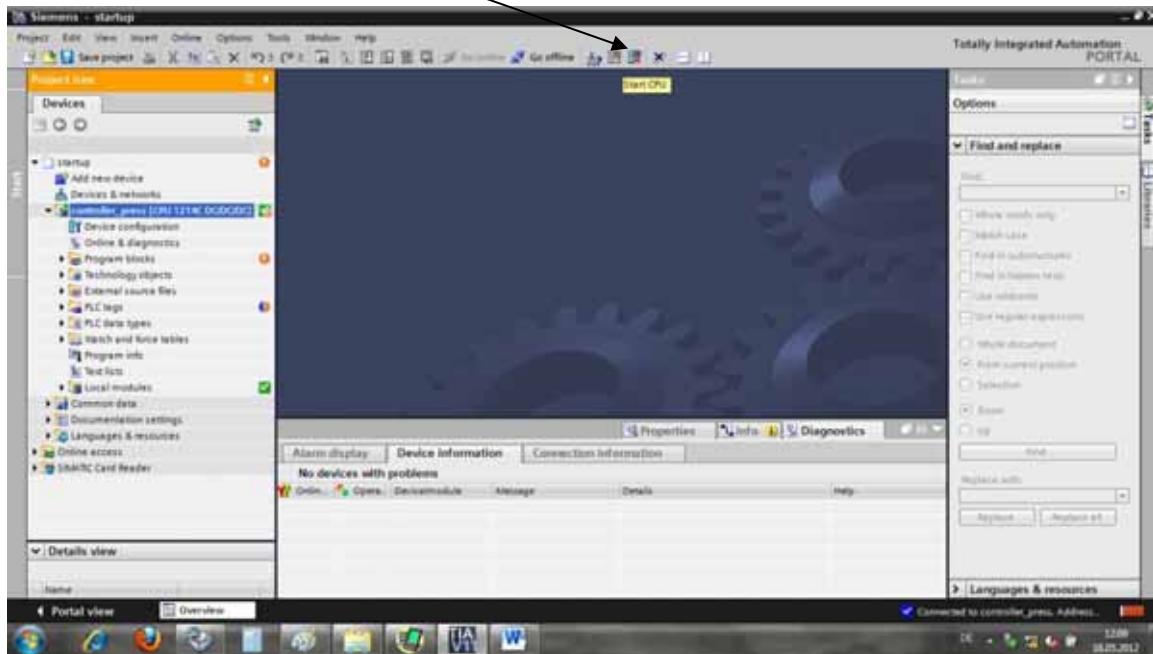
4

5. Next, select the '**PG/PC interface for online access**' and then click on '**Go online**' (→ PG/PC interface for online access → Go online)



6. Once we are connected to the CPU online, we can start and stop the CPU with the following

buttons  . In project navigation, references regarding the diagnosis are already provided symbolically.



4

Symbols for the comparison status

On the bottom to the right, the diagnosis symbols can be combined with smaller supplementary symbols that indicate the result of the online/offline comparison.

Symbol	Description
!	Folder contains objects whose online and offline versions differ
?	Comparison results are not known
■	Online and offline versions of the object are identical
●	Online and offline versions of the object are different
○	Object only exists offline
○	Object only exists online

Combined Diagnosis and Comparison Symbols

Icon	Meaning
	Folder contains objects whose online and offline versions differ (only in the project tree)
	Object only exists online

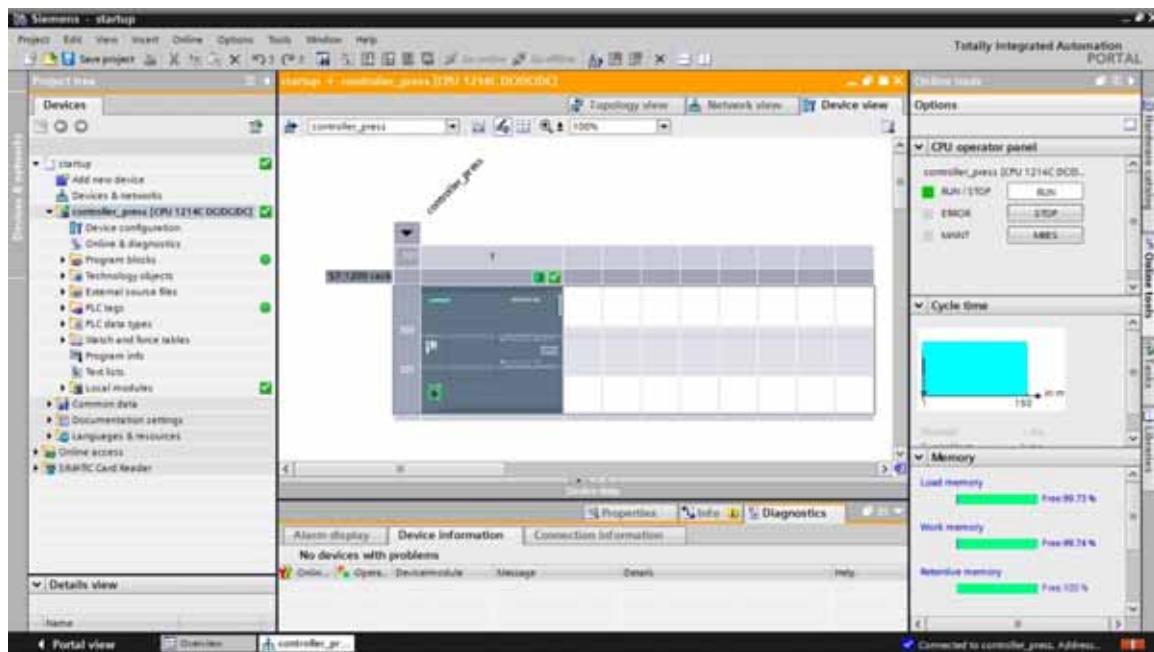
Diagnosis Symbols for Modules and Devices

4

Icon	Meaning
	The connection with a CPU is currently being established.
	The CPU is not reachable at the set address.
	The configured CPU and the CPU actually present are of incompatible types.
	On establishment of the online connection to a protected CPU, the password dialog was terminated without specification of the correct password.
	No fault
	Maintenance required
	Maintenance demanded
	Error
	The module or device is deactivated.
	The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
	Diagnostics data are not available because the current online configuration data differ from the offline configuration data.
	The configured module or device and the module or device actually present are incompatible (valid for modules or devices under a CPU).
	The configured module does not support display of the diagnostics status (valid for modules under a CPU).
	The connection is established, but the state of the module has not yet been determined.
	The configured module does not support display of the diagnostics status.
	Error in subordinate component: An error is present in at least one subordinate hardware component.

3.1. Device Diagnosis

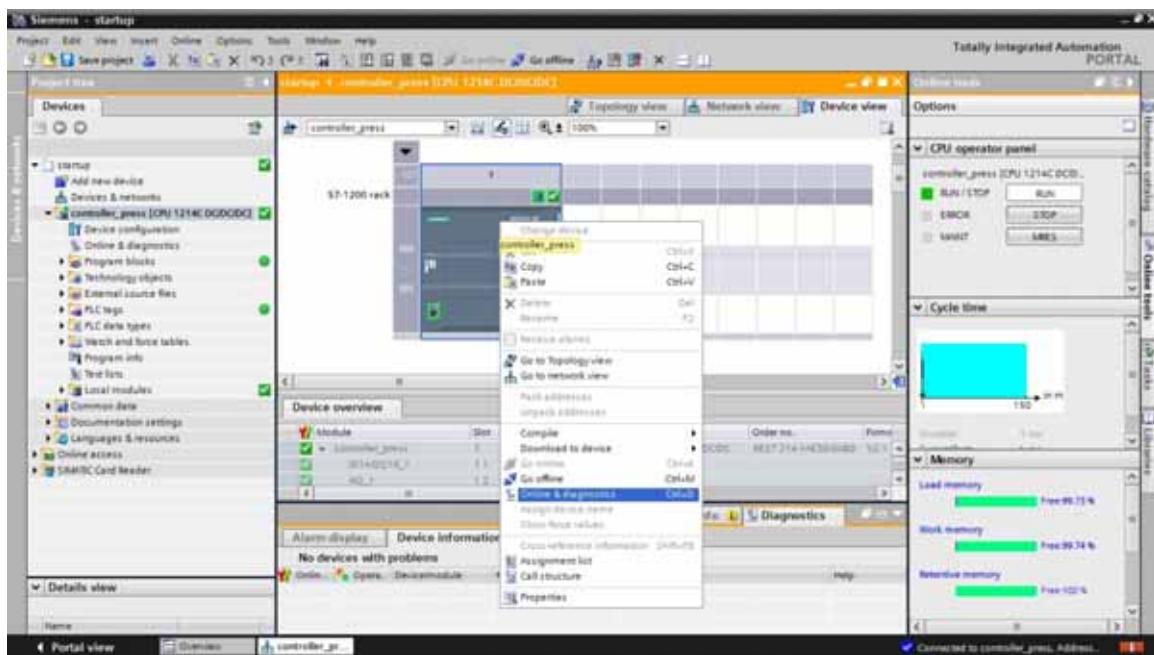
7. If we open the '**Device configuration**', under '**Diagnosis**' we are informed about the status of the individual components. (→ Device configuration → Diagnosis)



Operating Mode Symbols for CPUs and CPs

Icon	Operating mode
	RUN
	STOP
	STARTUP
	HOLD
	DEFECTIVE
	Unknown operating mode
	The configured module does not support display of the operating mode.

8. You can view details about the CPU if you click on it with the right mouse key, and then select '**Online & diagnosis**'. (→ Online & diagnosis)

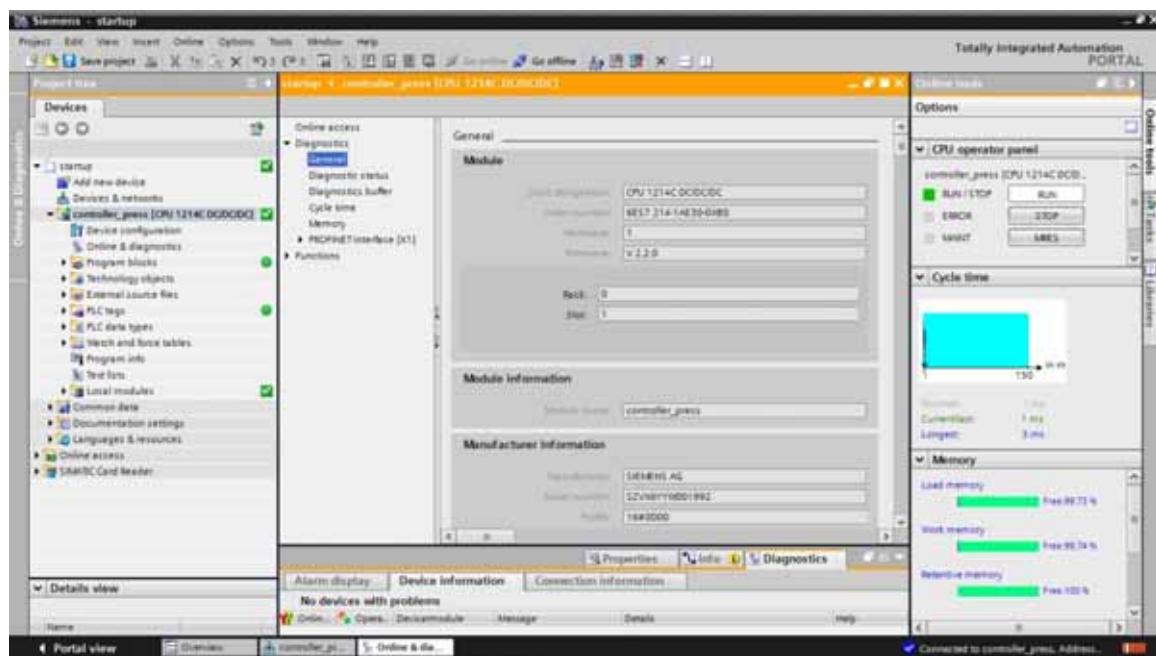


Color code for ports and Ethernet cables

The table below shows the possible colors and their respective meaning.

Color	Meaning
	No fault or maintenance required
	Maintenance demanded
	Communication error

9. On the right side, the following is displayed: an operator panel for the CPU, the cycle time and the memory capacity.



10. General information about the CPU is available. (→ General)

4

General

Module

Short designation: **CPU 1214C DC/DC/DC**

Order number: **6ES7 214-1AE30-0XB0**

Hardware: **1**

Firmware: **V 2.2.0**

Rack: **0**

Slot: **1**

Module information

Module name: **controller_press**

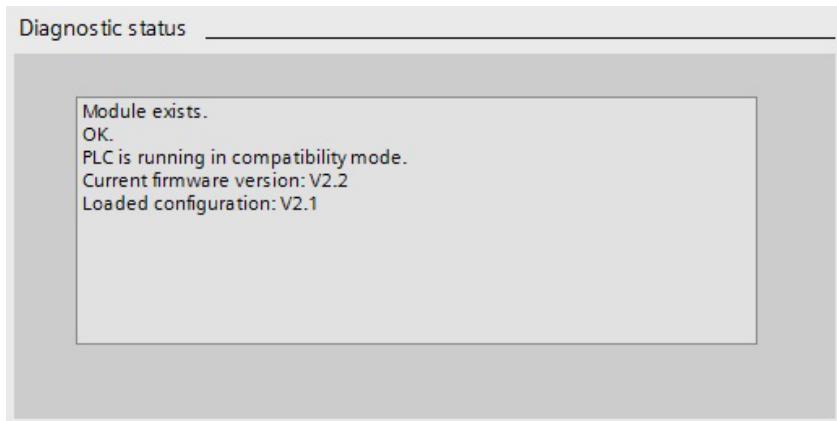
Manufacturer information

Manufacturer: **SIEMENS AG**

Serial number: **SZVX6YYW001992**

Profile: **16#0000**

11. If there is information regarding the diagnosis, it is displayed in the diagnosis status and in the diagnostics buffer. (→ Diagnostic status → Diagnostics buffer)



4

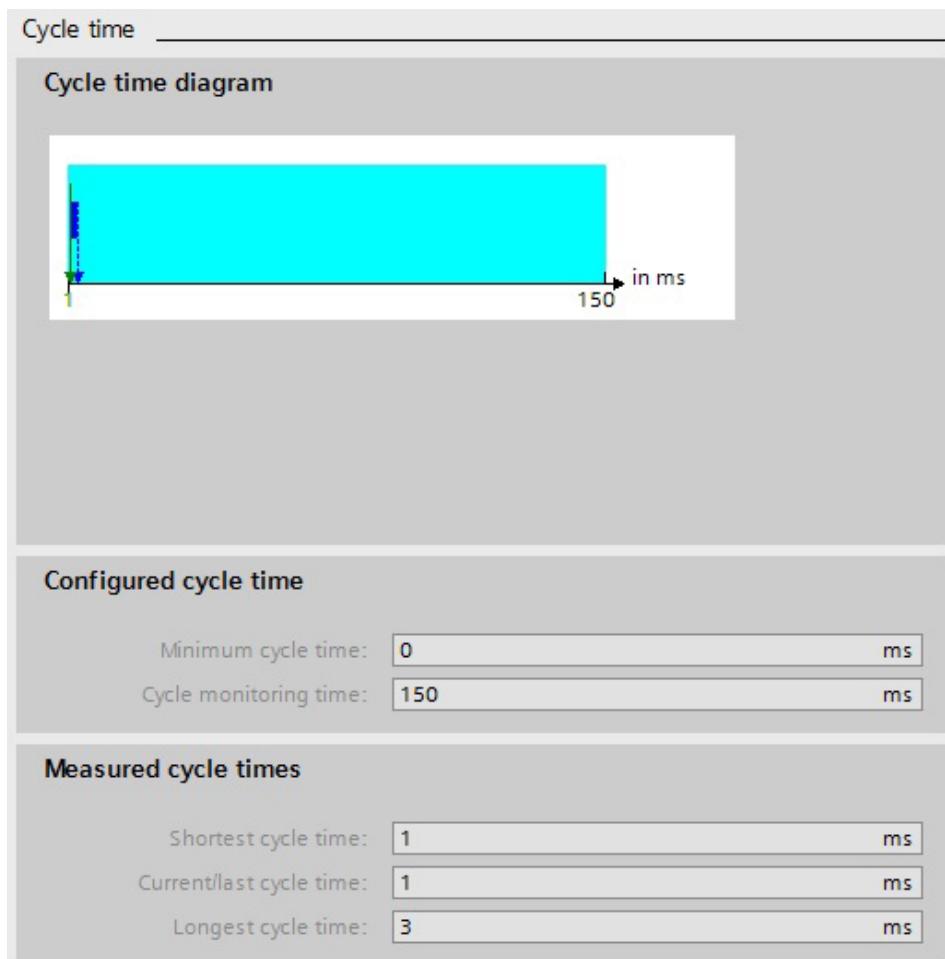
12. The most important aid for error diagnosis is the diagnostics buffer.

The screenshot shows the 'Diagnostics buffer' section of a SIMATIC management interface. It includes the following components:

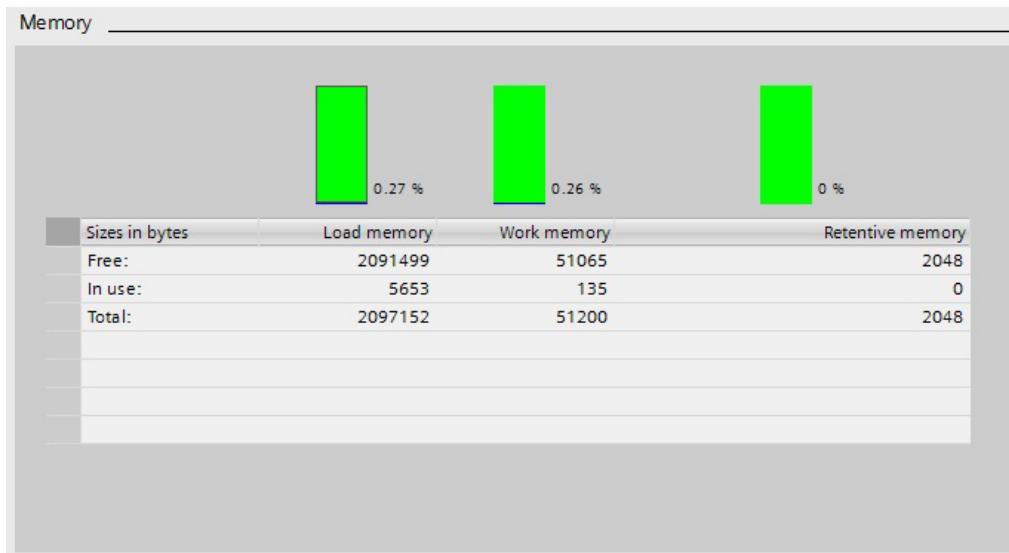
- Events:** A table listing 9 events. The first event is highlighted in blue. The table has columns: No., Date and time, Event, and two small icons.
- Details on event:** A detailed view for the first event (No. 1). It shows the event number (1), date (1/3/1970 12:49:42.310 AM), event ID (16# 02:400C), and a description box containing text about CPU mode changes and startup inhibit settings.
- Buttons:** 'Freeze display' button at the bottom left of the event table, and scroll bars for the event list and details view.

No.	Date and time	Event
1	1/3/1970 12:49:42.310 ...	Follow-on operating mode change - CPU changes from STARTUP to RUN mode
2	1/3/1970 12:49:42.303 ...	Communication initiated request: WARM RESTART - CPU changes from STOP to ...
3	1/3/1970 12:49:42.303 ...	New startup information - Current CPU operating mode: STOP
4	1/3/1970 12:49:08.203 ...	New startup information - Current CPU operating mode: STOP
5	1/3/1970 12:49:08.103 ...	New startup information - Current CPU operating mode: STOP
6	1/3/1970 12:49:02.302 ...	New startup information - Current CPU operating mode: STOP
7	1/3/1970 12:49:00.554 ...	New startup information - Current CPU operating mode: STOP
8	1/3/1970 12:49:00.453 ...	Communication initiated request: STOP - CPU changes from RUN to STOP mode
9	1/2/1970 9:51:52.504 PM	Follow-on operating mode change - CPU changes from STARTUP to RUN mode

13. In addition, information is provided about the cycle time of the processed program. (→ Cycle time)

**4**

14. Here, memory capacity is provided in detail. (→ Memory)



4

15. In addition, information about the CPU's interface is provided.

The figure shows a PROFINET interface configuration interface. It includes sections for Ethernet address, Network connection, and IP parameters. The IP parameters section displays MAC and IP addresses.

PROFINET interface [X1]

- > Ethernet address: _____
- > > Network connection: _____

MAC address: 00-1C-06-00-6D-42

- > > IP parameters: _____

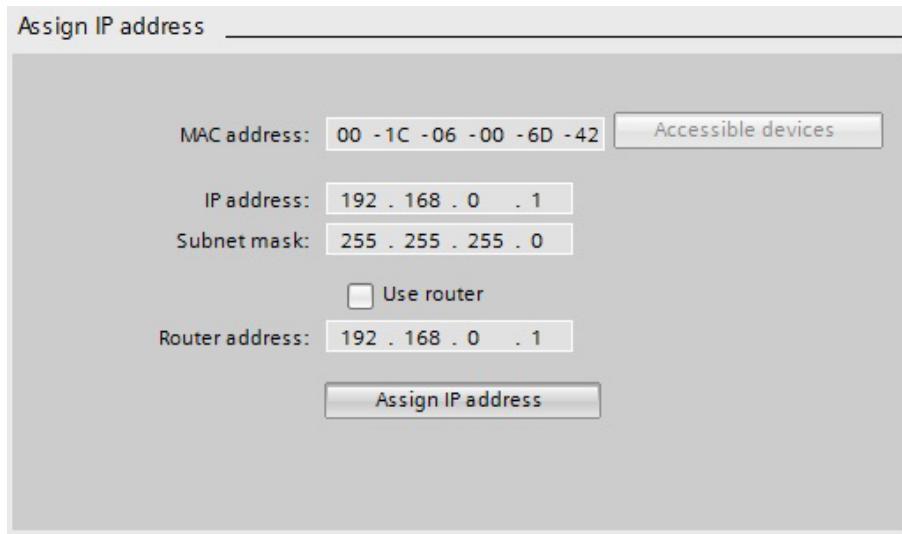
IP address:	192.168.0.1
Subnet mask:	255.255.255.0
Default router:	0.0.0.0
IP settings:	---
IP setting time:	---

- > Ports: _____

Ports

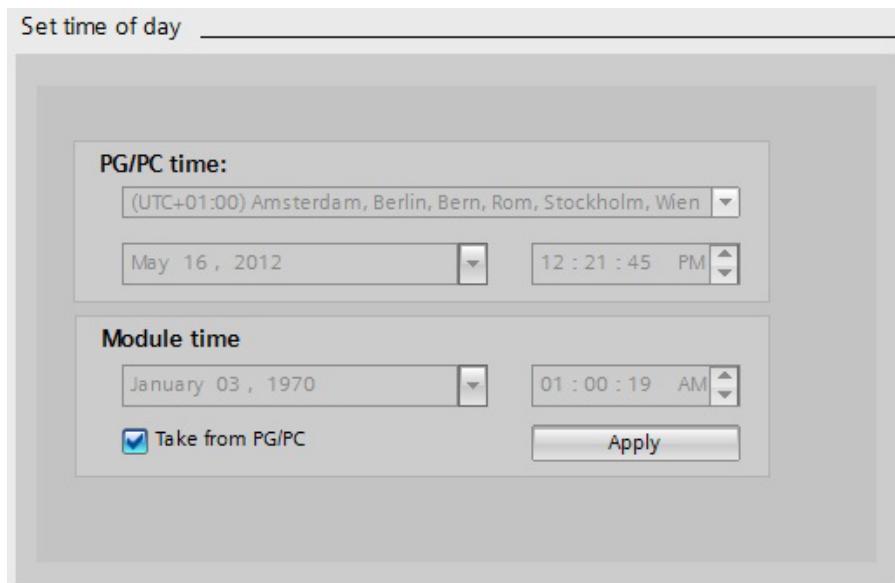
Port no.	Status	Settings	Mode

16. Under Functions, the IP address is assigned. (→ Functions → Assign IP address)

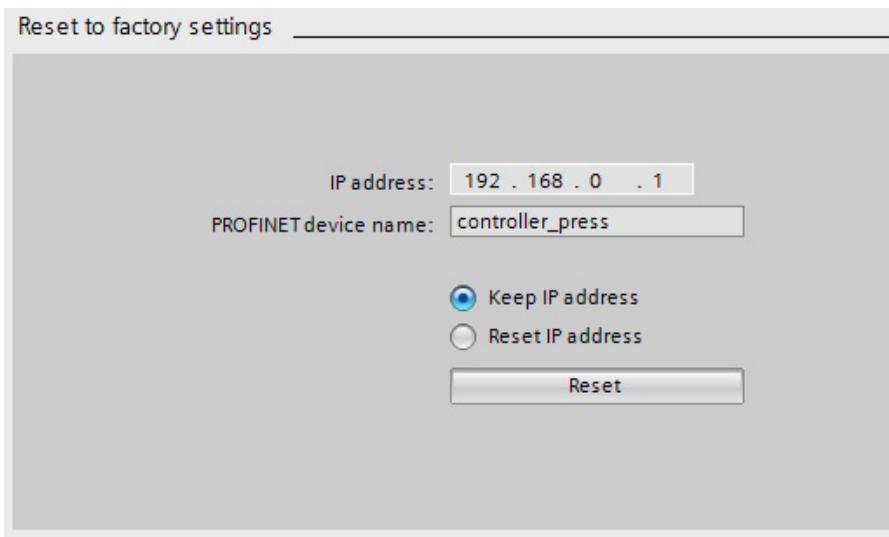


4

17. Under Functions, the time of day can be set. (→ Set time of day)

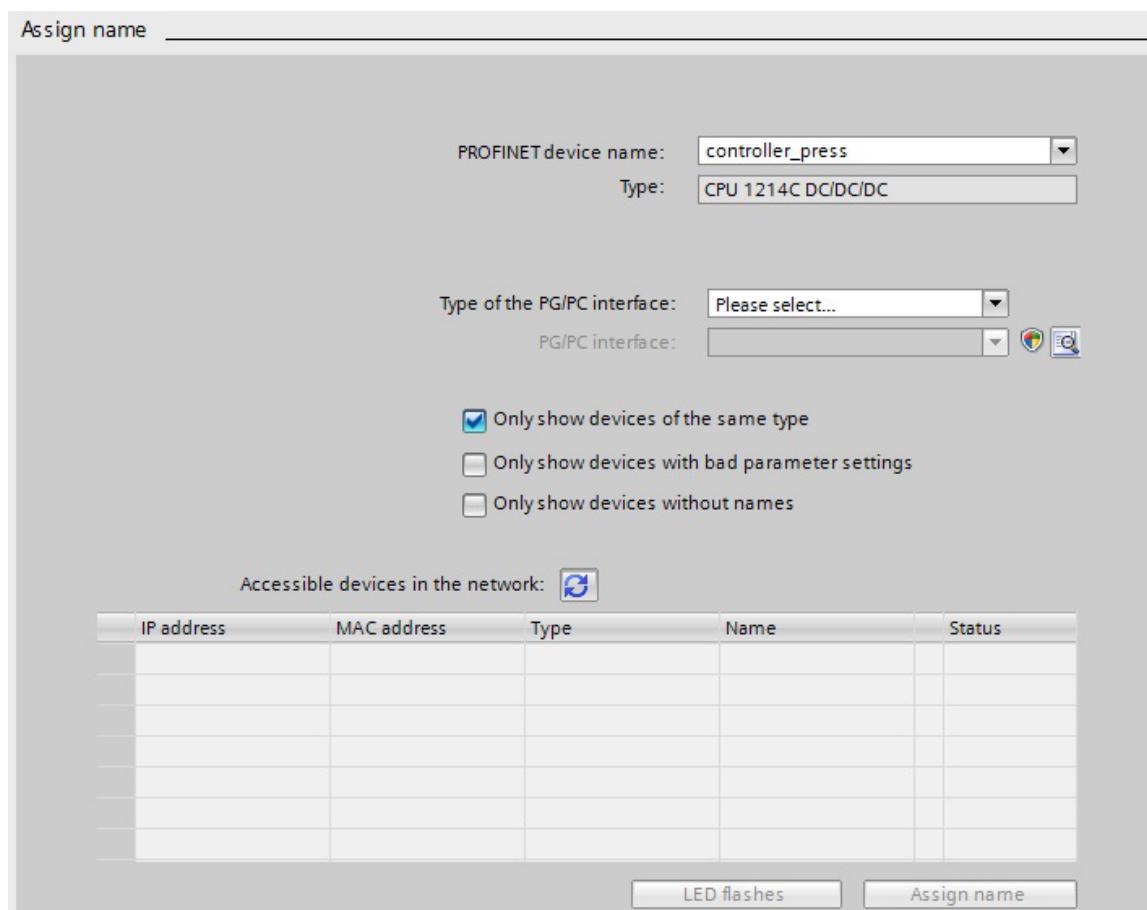


18. Here, we can also perform a reset to the factory setting. (→ Reset to factory settings)



4

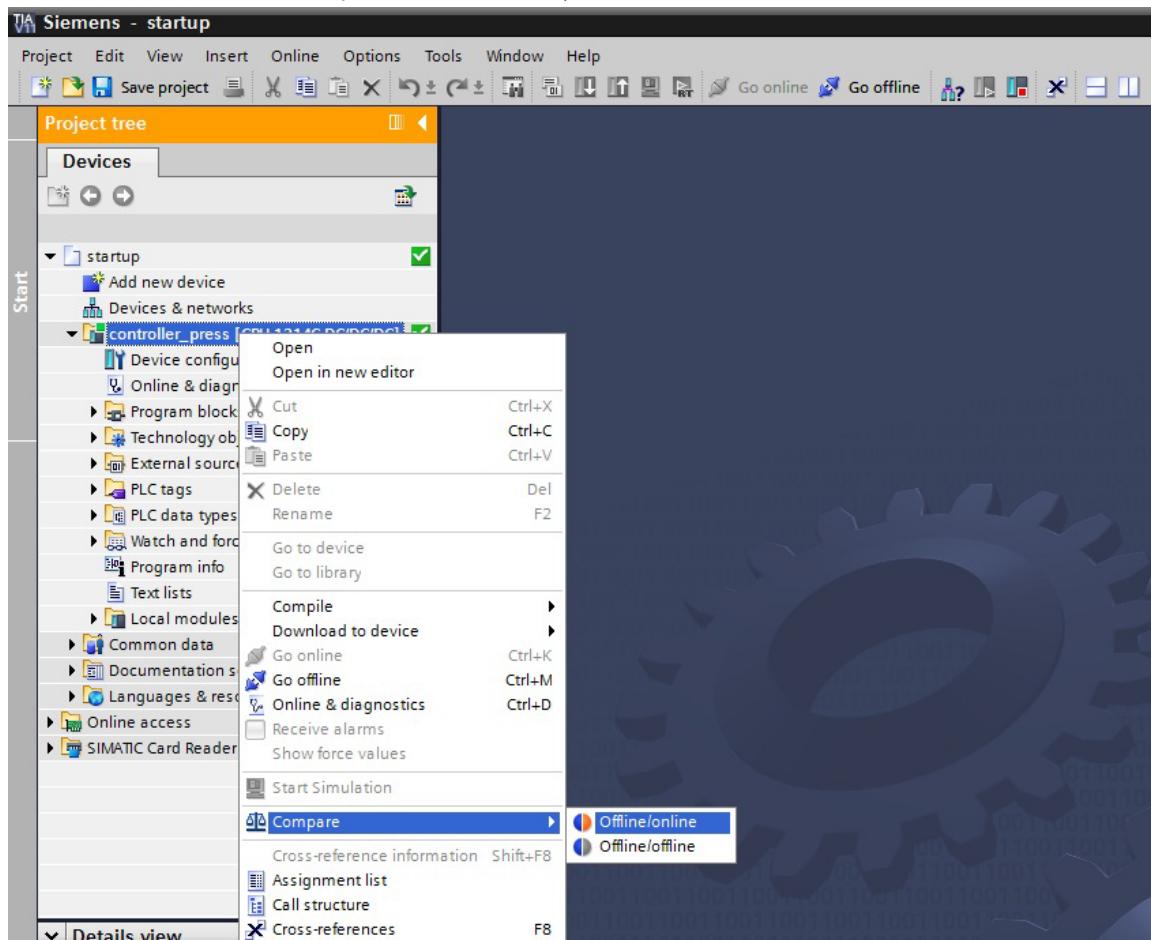
19. Here, the PROFINET device name can be checked and assigned.



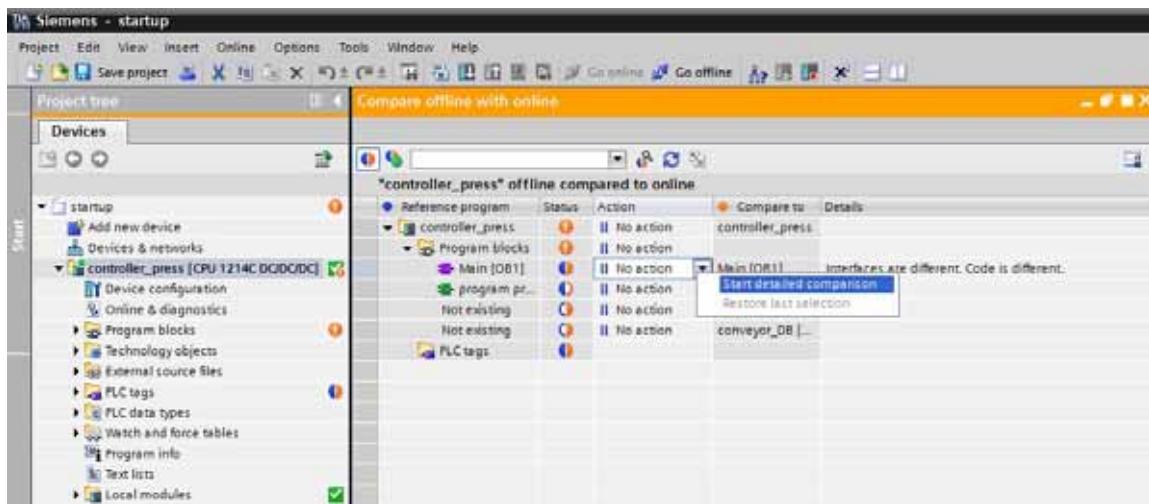
3.2. Offline/Online Comparison

Often it is important to know whether the stored data matches the data loaded to the controller. First, remove the negation at Emergency OFF at the AND in block **Program Press [FC1]B**. Save block **Program Press**, but **do not** load to the controller.

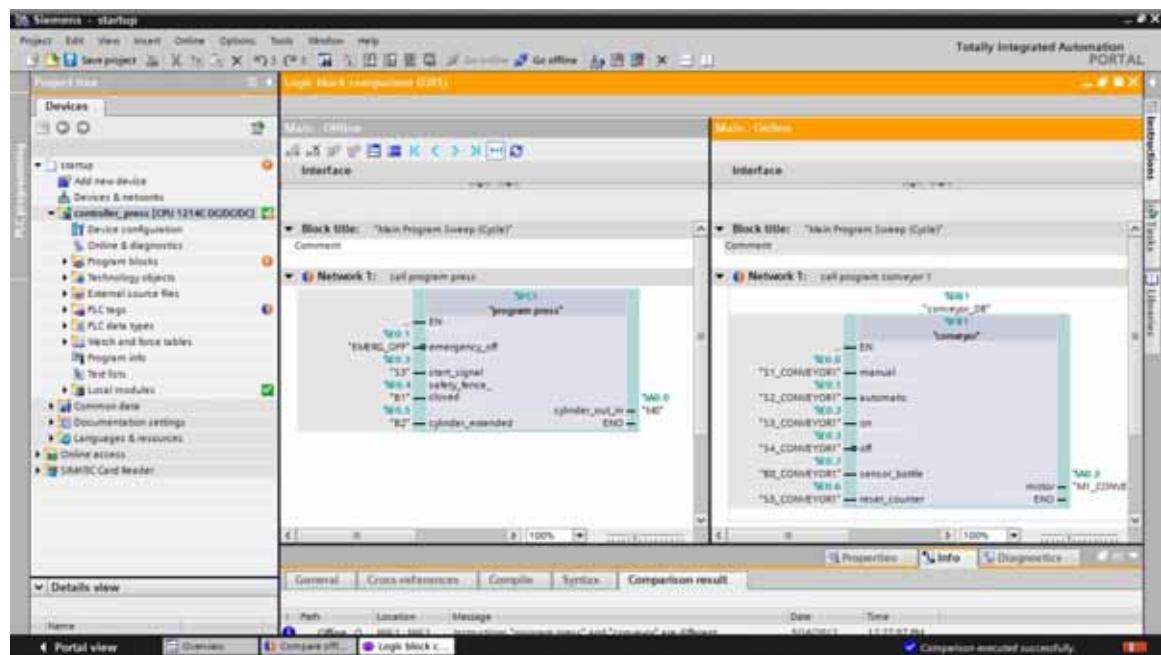
To make the comparison, first select '**controller press**' and then '**Compare Offline/Online**'. (→ Controller Press → Compare Offline/Online)



If block differences are displayed , you can **'Start detailed comparison.** (→ Program Press → Start detailed comparison)

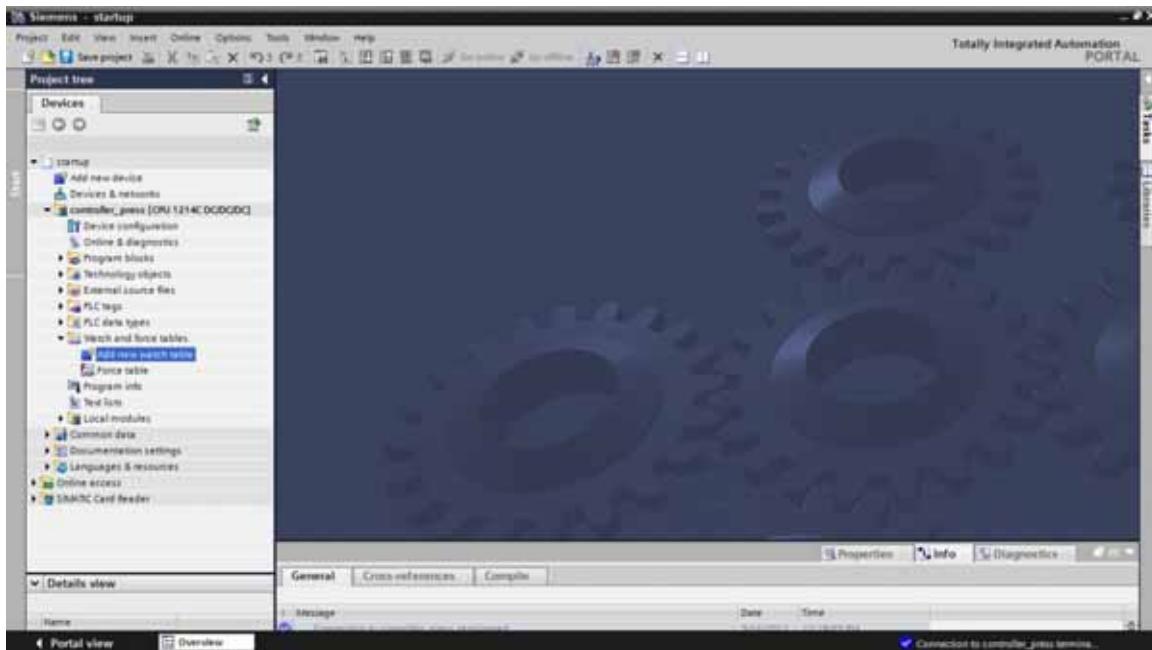


The block is displayed online and offline, and references to differences are highlighted. (→ Comparison result)



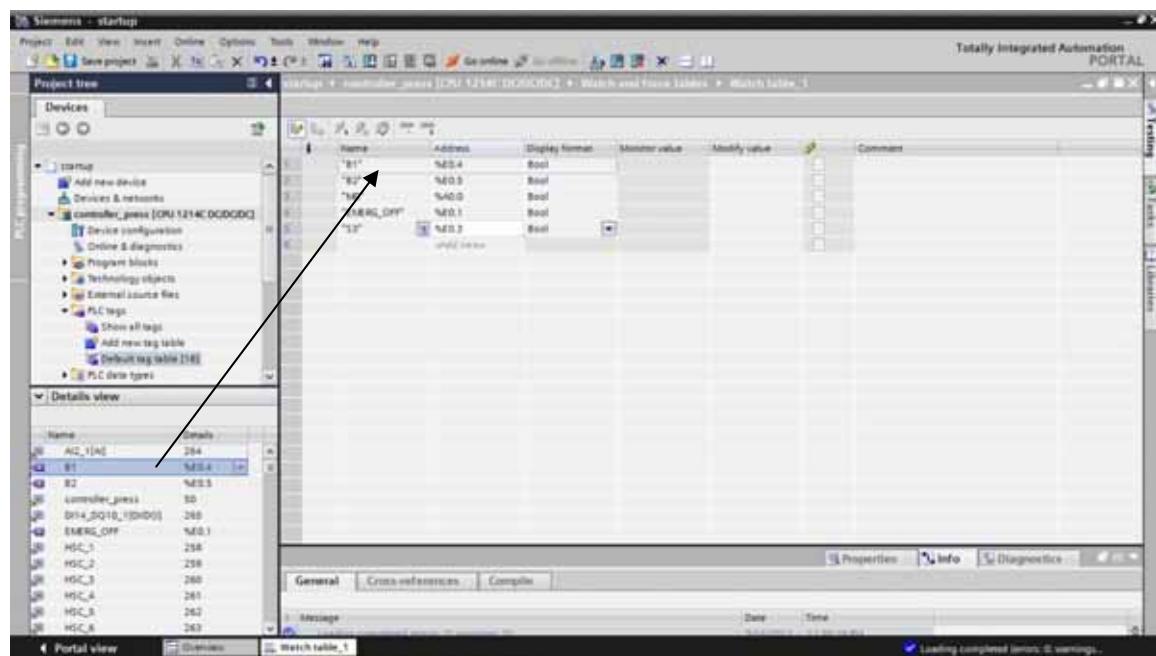
3.3. Monitoring and Controlling Variables

1. To monitor and control variables, '**Add new watch table**'. (→ Add new watch table)



4

You can enter individual tags in the table or, after selecting the default tag table, highlight the tags to be watched and drag them from the detail view to the watch table. (→ Default tag table)



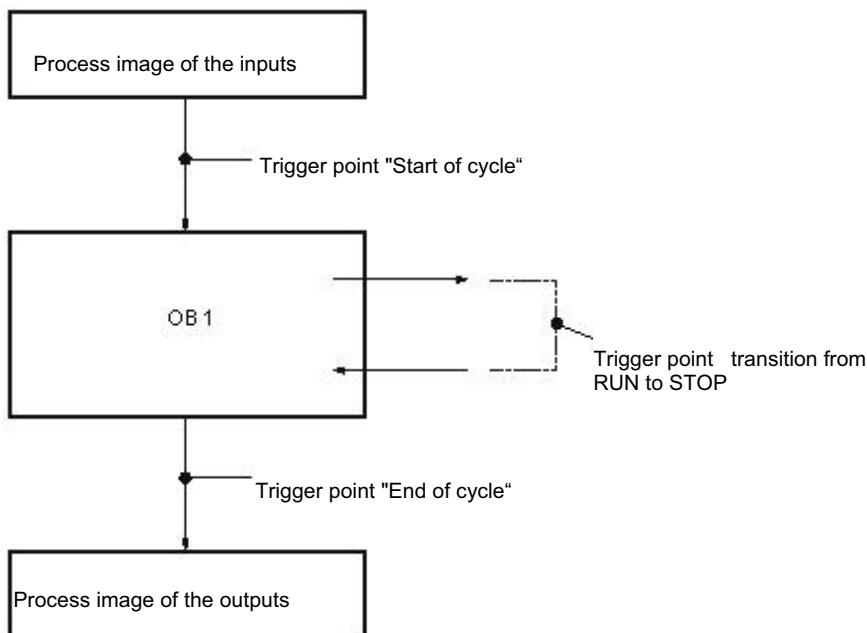
The following columns can be displayed: 'All control columns' and 'All columns of the expanded mode'. Now, select the trigger time for monitoring. (→ Permanent)

	Name	Address	Display format	Monitor value	Monitor with trig...	Modify with trig...	Modify value	Comment
1	"B1"	%E0.4	Bool		Permanent	Permanent		
2	"B2"	%E0.5	Bool		Permanent	Permanent		
3	"M0"	%A0.0	Bool		Permanent	Permanent		
4	"EMERG_OFF"	%E0.1	Bool		Permanent	Permanent		
5	"S3"	%E0.3	Bool		Permanent	Permanent		
6	<Add new>				Permanent Permanently, at start of scan cycle Once only, at start of scan cycle Permanently, at end of scan cycle Once only, at end of scan cycle Permanently, at transition to STOP Once only, at transition to STOP			

4

The following monitoring and control modes are provided:

- Permanent (in this mode, the inputs can be monitored/controlled at the beginning of the cycle, and the outputs at the end of the cycle)
- Cycle start once
- Cycle end once
- Cycle start permanent
- Cycle end permanent
- Transition to STOP once
- Transition STOP permanent



Next, click on 'Monitor all values once and immediately' or on 'Monitor all'. (→ Monitor all)



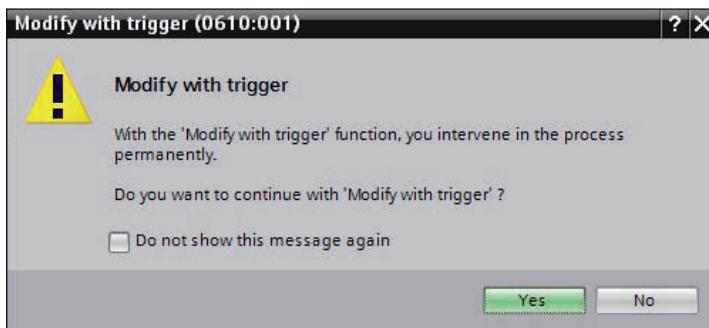
	Name	Address	Display format	Monitor value	Monitor with trig	Modify with trigger	Modify value	Comment
1	"B1"	%E0.4	Bool	FALSE	Permanent	Permanent		
2	"B2"	%E0.5	Bool	FALSE	Permanent	Permanent		
3	"M0"	%A0.0	Bool	FALSE	Permanent	Permanent		
4	"EMERG_OFF"	%E0.1	Bool	TRUE	Permanent	Permanent		
5	"S3"	%E0.3	Bool	TRUE	Permanent	Permanent		
<Add new>								

Enter the desired 'Control values' and activate them. Then, click on to 'modify all activated values once and immediately', or on to 'modify all activated values through "modify with trigger"'. (→ → Modifies all activated values through 'Modify with trigger')

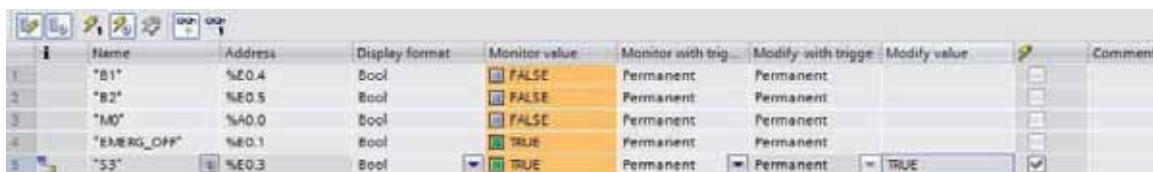


	Name	Address	Display format	Monitor value	Monitor with trig	Modify with trigger	Modify value	Comment
1	"B1"	%E0.4	Bool	FALSE	Permanent	Permanent		
2	"B2"	%E0.5	Bool	FALSE	Permanent	Permanent		
3	"M0"	%A0.0	Bool	FALSE	Permanent	Permanent		
4	"EMERG_OFF"	%E0.1	Bool	TRUE	Permanent	Permanent		
5	"S3"	%E0.3	Bool	FALSE	Permanent	Permanent	TRUE	
<Add new>								

Confirm the warning with 'Yes'. (→ Yes)



The input becomes active although no voltage is applied to the terminal.



	Name	Address	Display format	Monitor value	Monitor with trig	Modify with trigger	Modify value	Comment
1	"B1"	%E0.4	Bool	FALSE	Permanent	Permanent		
2	"B2"	%E0.5	Bool	FALSE	Permanent	Permanent		
3	"M0"	%A0.0	Bool	FALSE	Permanent	Permanent		
4	"EMERG_OFF"	%E0.1	Bool	TRUE	Permanent	Permanent		
5	"S3"	%E0.3	Bool	TRUE	Permanent	Permanent	TRUE	
<Add new>								

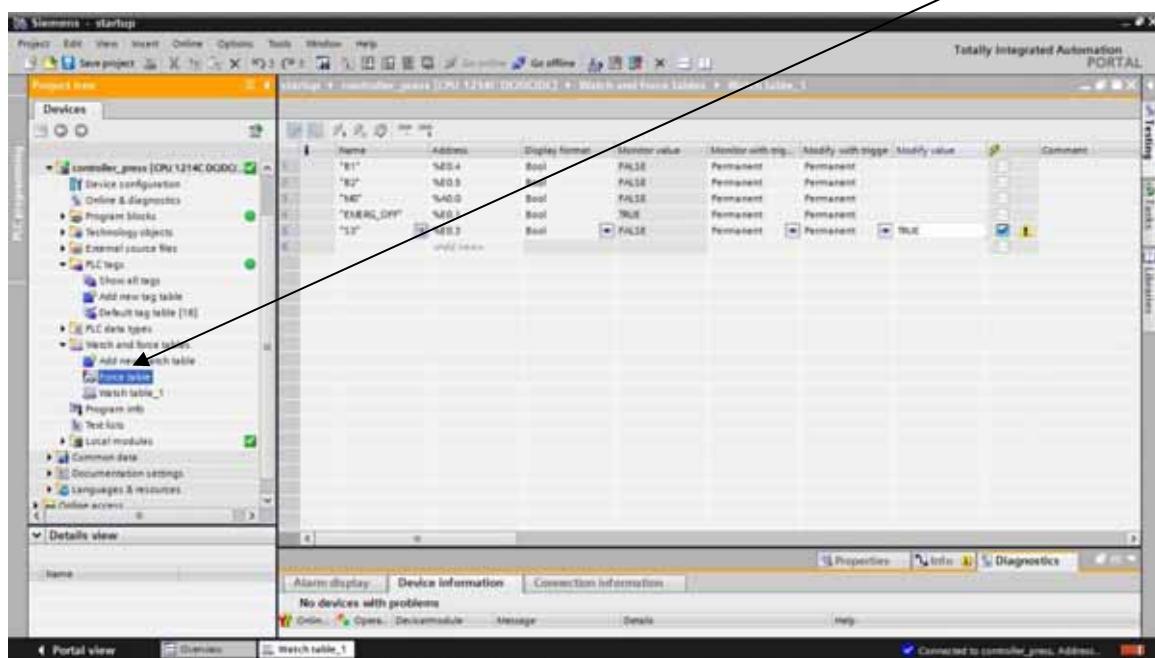
Note

If the watch table is closed or the connection to the PLC is lost, all modify commands become ineffective.

3.4. Forcing Variables

With the function 'Force', variables can be assigned a permanent value. Force values are specified similar to "Modify variables" but in contrast are retained after the CPU is switched off or stopped.

To force, first the force table has to be opened with a double click. (→ Force table)



1. Copy the operand "M0" with the address %A0.0 from the watch table. (→ M0)

i	Name	Address	Display format	Monitor value	Monitor with trig...	Modify with trigge...	Modify value	Comment
1	"B1"	%E0.4	Bool	FALSE	Permanent	Permanent	<input type="checkbox"/>	
2	"B2"	%E0.5	Bool	FALSE	Permanent	Permanent	<input type="checkbox"/>	
3	"M0"	%A0.0	Bool	FALSE	Permanent	Permanent	<input type="checkbox"/>	
4	"EMERG_OFF"	%E0.1	Bool	TRUE	Permanent	Permanent	<input type="checkbox"/>	
5	"S1"	%E0.3	Bool	FALSE	Permanent	Permanent	<input checked="" type="checkbox"/> TRUE	

Add the operand M0 to the force table. In the case of forcing, the operands are entered with direct IO access. (Syntax: %A0.0:P)

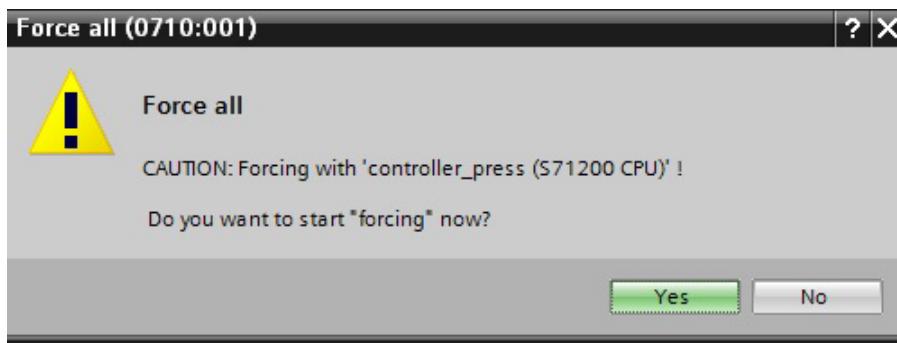
i	Name	Address	Display format	Monitor value	Force value	F	Comment
1	*M0*:P	%A0.0:P	Bool	0	FALSE	<input type="checkbox"/>	
2		<Add new>				<input type="checkbox"/>	

Enter the desired '**values**' and activate them. Then, click on 'Start or replace forcing'.

This writes the new force request to the CPU. ($\rightarrow \%A0.0:P \rightarrow \text{TRUE} \rightarrow \checkmark \rightarrow$ Start or replace forcing)

i	Name	Address	Display format	Monitor value	Force value	F	Comment
1	*M0*:P	%A0.0:P	Bool	0	TRUE	<input checked="" type="checkbox"/>	
2		<Add new>				<input type="checkbox"/>	

2. Confirm the warning with 'Yes'. (→ Yes)

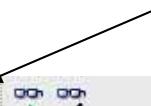


	i	Name	Address	Display format	Monitor value	Force value	F	Comment
1		"MO":P	%A0.0:P	Bool		TRUE	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2			<Add new>				<input type="checkbox"/>	

Note:

If the watch table is closed or if the connection to the PLC is lost, **forcing remains active** and the yellow **FRCE LED** on the CPU is lit.

If you want to '**End forcing**' just click on: . (→ End forcing)



		Name	Address	Display format	Monitor value	Force value		Comment
1		"M0":P	%AO.0:P	Bool		TRUE	<input checked="" type="checkbox"/>	
2		<Add new>					<input type="checkbox"/>	

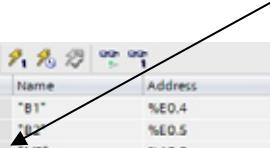
Forcing is terminated and the yellow **FRCE LED** on the CPU turns off.

If there is already a force request on the controller, it is indicated in the watch table with the symbol .



		Name	Address	Display format	Monitor value	Monitor with trig...	Modify with trigge	Modify value		Comment
1		"B1"	%E0.4	Bool	FALSE	Permanent	Permanent		<input type="checkbox"/>	
2		"B2"	%E0.5	Bool	FALSE	Permanent	Permanent		<input type="checkbox"/>	
3		"M0"	%AO.0	Bool	FALSE	Permanent	Permanent		<input type="checkbox"/>	
4		"EMERG_OFF"	%E0.1	Bool	TRUE	Permanent	Permanent		<input checked="" type="checkbox"/>	
5		"S3"	%E0.3	Bool	FALSE	Permanent	Permanent	TRUE	<input checked="" type="checkbox"/>	

Now, if you click on , additional information is displayed. (→)



		Name	Address	Display format	Monitor value	Monitor with trig...	Modify with trigge	Modify value		Comment
1		"B1"	%E0.4	Bool	FALSE	Permanent	Permanent		<input type="checkbox"/>	
2		"B2"	%E0.5	Bool	FALSE	Permanent	Permanent		<input type="checkbox"/>	
3		"M0"	%AO.0	Bool	FALSE	Permanent	Permanent		<input type="checkbox"/>	
4		"EMERG_OFF"	%E0.1	Bool	TRUE	Permanent	Permanent		<input checked="" type="checkbox"/>	
5		"S3"	%E0.3	Bool	FALSE	Permanent	Permanent	TRUE	<input checked="" type="checkbox"/>	

3.5. Enabling IO Outputs

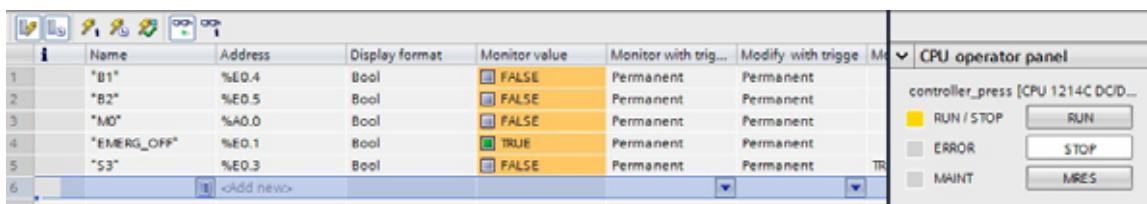
With the function “Enable peripheral outputs”, the command output disable -which in the CPU’s stop mode prevents the output of output signals- can be switched off.

Activated outputs are active in the CPU’s stop mode after the IO output enable. Analog modules read out their last value.

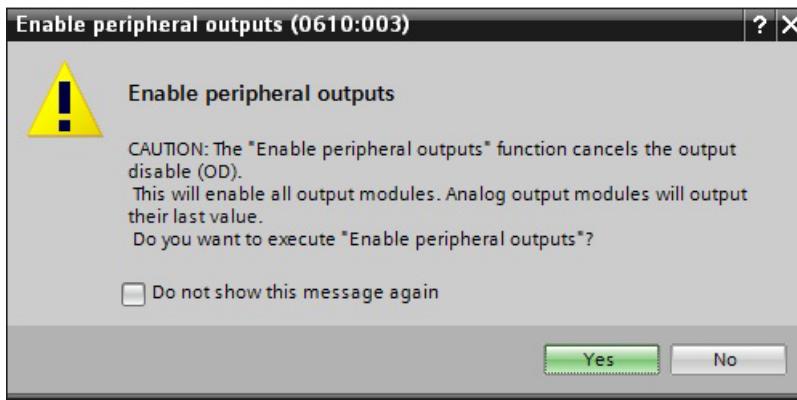
If you want to use this function, in the watch table enter the operand in the correct display format and control value.

1. Switch the CPU to the **STOP mode**

and click on the button 'Enable peripheral outputs'.



2. Confirm the warning with 'Yes'. (→ Yes)



The operand has to be addressed as periphery in the byte, word or DWord (%AB0:P, %AW0:P, %AD0:P) format.

	Name	Address	Display format	Monitor value	Monitor with trig...	Modify with trigge...	Modify value	Comment
1	"B1"	%E0.4	Bool	FALSE	Permanent	Permanent		
2	"B2"	%E0.5	Bool	FALSE	Permanent	Permanent		
3	"M0"	%A0.0	Bool	TRUE	Permanent	Permanent		
4	"EMERG_OFF"	%E0.1	Bool	TRUE	Permanent	Permanent		
5	"S3"	%E0.3	Bool	TRUE	Permanent	Permanent	TRUE	
6		%AB0:P	Hex	16#01	Permanent	Permanent	16#01	

Click on force variable immediately

Although periphery outputs cannot be monitored, it is possible to monitor the process image of the outputs (PIQ) in the CPU's memory (also in the STOP mode).



By again clicking on the button or by switching the CPU to the RUN mode, enable peripheral outputs can be terminated.

Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Module 010-050



TIA Portal Module 010-010

Startup Programming of the
SIMATIC S7-1200

TIA Portal Module 010-020

Block Types for the SIMATIC S7-1200

TIA Portal Module 010-030

IEC Timers and IEC Counters at the
SIMATIC S7-1200

TIA Portal Module 010-040

Diagnosis and Error Search at the
SIMATIC S7-1200

TIA Portal Module 010-050

Analog Value Processing with
SIMATIC S7-1200

TIA Portal Module 010-060

Control Engineering with
SIMATIC S7-1200

TIA Portal Module 010-070

Communication between two
SIMATIC S7-1200

TIA Portal Module 010-080

HMI Panel KTP600 with SIMATIC
S7-1200

TIA Portal Module 010-090

Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

Matching SCE training packages for these documents

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
Order number: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
Order number 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
Order number 6ES7822-0AA01-4YE0

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

Continued Training

For regional Siemens SCE continued training, please contact your regional SCE contact person
siemens.com/sce/contact

5

Additional information regarding SCE

siemens.com/sce

Information regarding Usage

The training curriculum 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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by 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.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the documents being used commercially.

We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this document.

PAGE

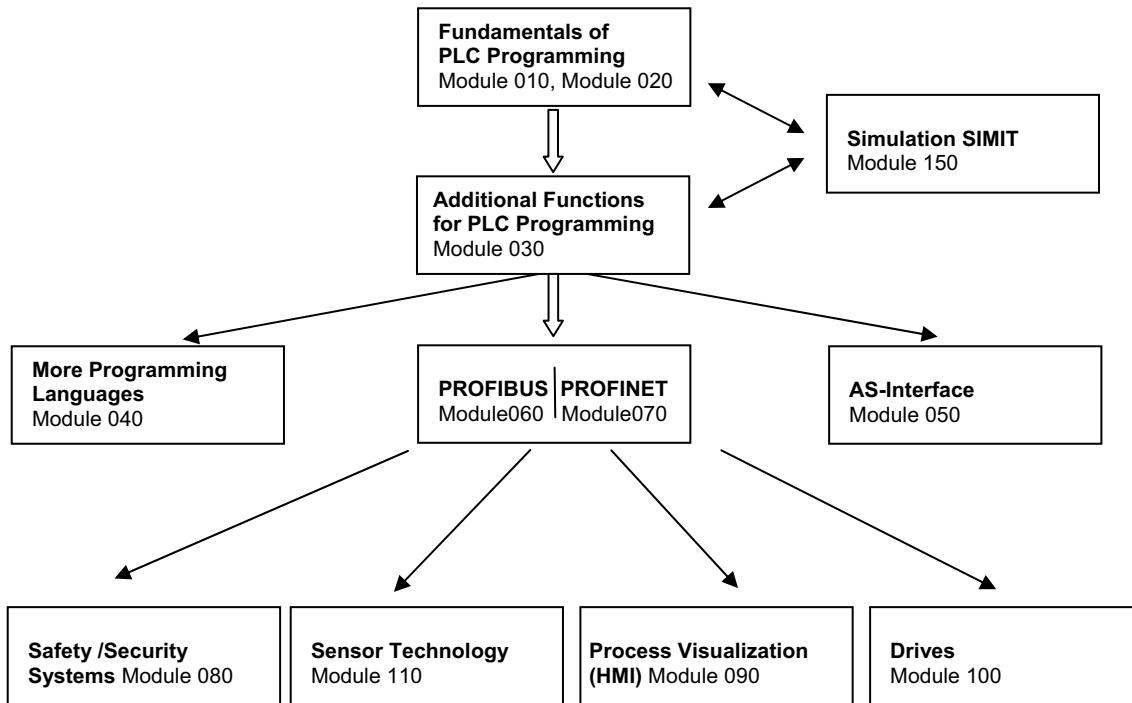
Contents

1.	Preface	4
2.	Notes on Programming the SIMATIC S7-1200.....	6
2.1	Automation System SIMATIC S7-1200.....	6
2.2	Programming Software STEP 7 Professional V11 (TIA Portal V11).....	6
3.	Analog Signals	7
4.	Data Types at the SIMATIC S7-1200	8
5.	Reading In/Reading Out Analog Values	9
5.1	Normalizing Analog Values	10
6.	Sample Task – Monitoring the Tank Level.....	10
6.1.	Programming Level Monitoring for the SIMATIC S7-1200.....	11

5

1. Preface

Regarding its content, module SCE_EN_010-050 is part of the training unit **Basics of PLC Programming** and represents a fast a **fast entry point** for programming the SIMATIC S7-1200 with the TIA Portal.



Training Objective

In this module SCE_EN_010-050, the reader learns how to program limit value monitoring for an analog value. The controller (PLC) in our case is the SIMATIC S7-1200 and the program is created with the programming tool TIA Portal. Module SCE-DE-010-1050 provides the fundamentals and shows how it's done using a detailed example.

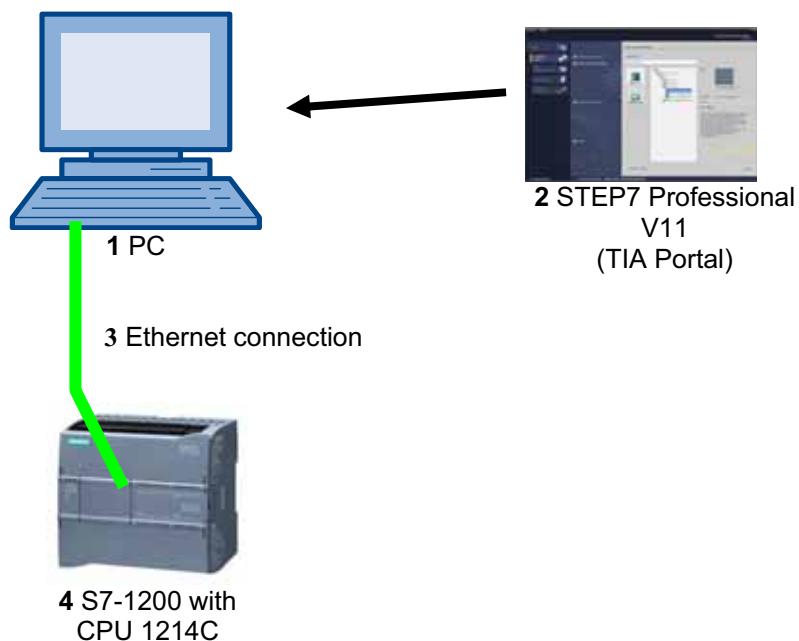
Prerequisites

To successfully work through module SCE_EN_010-050, the following knowledge is assumed

- How to operate Windows
- Fundamentals of PLC programming with the TIA Portal (for example, Module_010-010 'Startup' Programming the SIMATIC S7-1200 with TIA- Portal V11)

Required Hardware and Software

- 1 PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB, operating system Windows XP Professional SP3/Windows 7 Professional/Windows 7 Enterprise/Windows 7 Ultimate/Windows 2003 Server R2/Windows Server 2008 Premium SP1, Business SP1, Ultimate SP1
- 2 Software STEP7 Professional V11 SP1 (Totally Integrated Automation (TIA) Portal V11)
- 3 Ethernet connection between PC and CPU 315F-2 PN/DP
- 4 PLC SIMATIC S7-1200; for example, CPU 1214C.
The inputs have to be brought out to a panel.



5

2. Notes on Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The automation system SIMATIC S7-1200 is a modular mini-controller system for the lower and medium performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU and input/output modules for digital and analog signals.

If needed, communication processors and function modules are added for special tasks such as step motor control.

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process, whereby the IO modules are polled in the S7 program by means of the input addresses (%I) and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7.

2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 (TIA Portal V11) is the programming tool for the following automation systems

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining the communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating the visual displays for the SIMATIC basic panels with the integrated WinCC Basic
- With additional WinCC packages, visual display solutions for PCs and other panels can be prepared

All functions are supported with detailed online help.

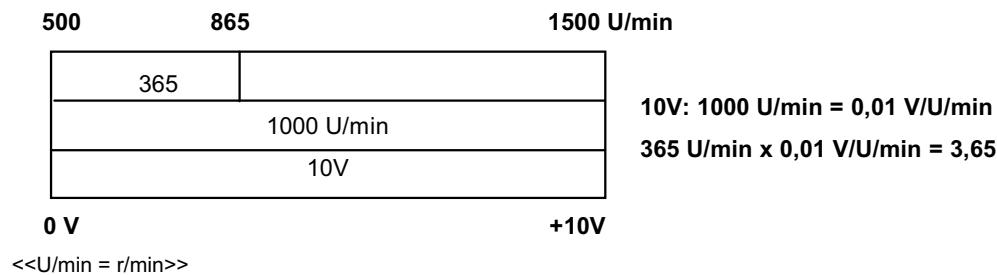
3. Analog Signals

In contrast to binary signals that can assume only the two signal states "Voltage present +24V" and "Voltage not present 0V", analog signals within a certain range can assume any number of values. A typical example of an analog sensor is a potentiometer. Depending on the position of the rotary button, any resistance can be set, up to the maximum value.

Below are some examples of analog variables in control engineering:

- Temperature -50 ... +150°C
- Flow 0 ... 200l/min
- Speed 500 ... 1500 r/min
- etc.

Using a transducer, these variables are converted into electrical voltages, currents or resistances. If, for example, a certain speed is to be recorded, the speed range of 500 ... 1500 r/min can be converted by means of a transducer into a voltage range of 0 ... +10V. If a speed of 865 rpm is measured, the transducer would read out a voltage of + 3.65 V.

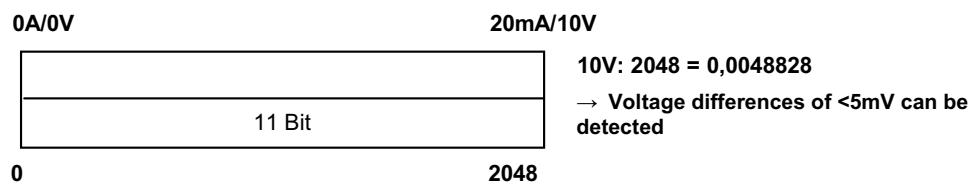


These electrical voltages, currents or resistances are then connected to an analog module that digitalizes this signal.

If analog variables are processed with a PLC, the voltage, current or resistance value that was read in has to be converted into digital information. This conversion is called analog/digital conversion (A/D conversion).

This means, for example, that the voltage 3.65V is stored in a series of binary digits as information. The more binary digits are used for digital representation, the finer is the resolution. If, for example, there were only 1 bit available for the voltage range 0 ... +10V, only one statement could be made whether the measured voltage is in the range 0... +5V or in the range +5V ... +10V. With 2 bits, the range can be divided into 4 individual ranges; i.e. 0 ... 2.5/2.5 ... 5/5 ... 7.5/7.5 ... 10V. A/D converters commonly used in control engineering convert with 8 or 11 bits.

With 8 bits, we have 256 individual ranges and with 11 bits a resolution of 2048 individual ranges.



4. Data Types at the SIMATIC S7-1200

The SIMATIC S7-1200 is provided with a large number of data types that can be used to represent different numerical formats. The list below shows the elementary data types.

Data Type	Size (Bit)	Range	Example of a Constant Entry
Bool	1	0 to 1	TRUE, FALSE, 0,1
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', 't', '@'
Sint	8	-128 to 127	123, -123
Int	16	-32768 to 32767	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, 3.4E3
LReal	64	+/-2.23 x 10-308 to +/-1.79 x 10 308	12345.123456789 -1.2E+40
Time	32	T#-24d_20h_31m_23s_648ms_to T#-24d_20h_31m_23s_647ms Stored 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:

For analog value processing, the data types '**INT**' and '**REAL**' play an important part since entered analog values are present as integers in the format '**INT**' and for accurate further processing, only floating point numbers '**REAL**' can be used because of the rounding off error in the case of '**INT**'.

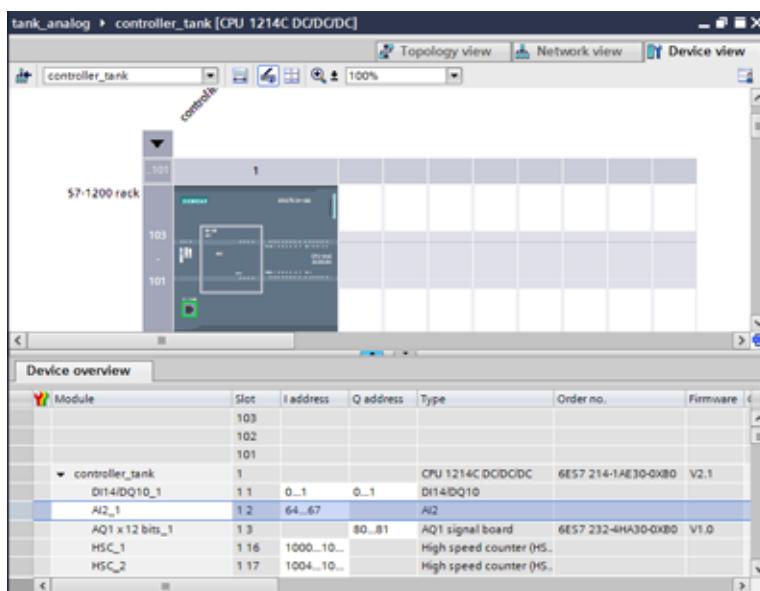
5. Reading In/Reading Out Analog Values

Analog values are entered/read out in the PLC as word information. These words are accessed with the operands

%IW 64	Analog input word 64
%QW 80	Analog output word 80
for example.	

Each analog value ("channel") is assigned one input or output word. The format is '**Int**', an integer.

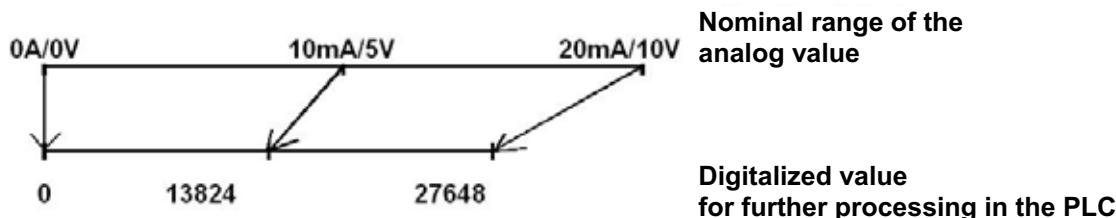
Addressing the input or output word depends on the addressing in the device overview. For example:



The address of the first analog input would be here %IW 64, the address of the second analog input %IW 66, and the address of the analog output %QW 80.

The analog value transformation for further processing in the PLC is the same for analog inputs and analog outputs.

The digitalized values look like this:



Often, these digitalized values have to be normalized through corresponding further processing in the PLC.

5.1 Normalizing Analog Values

If an analog input value is present as digitalized value, it usually has to be normalized so that the numerical values correspond to the physical values in the process.

Likewise, the analog output to the IO output word usually takes place only after the output value is normalized.

In STEP7 programs, computing operations are used for normalizing. For this to be done as accurately as possible, the values have to be converted to the data type REAL normalizing, to keep the rounding off errors to a minimum.

In the chapters below, an example is provided using level monitoring of a tank as an illustration.

6. Sample Task – Monitoring the Tank Level

We are going to program monitoring the level in a tank.

A sensor measures the level in a tank and converts it into the voltage signal 0 to 10V. 0V corresponds to a level of 100 liters and 10V to a level of 1000 liters.

This sensor is connected to the analog input of the SIMATIC S7-1200.

Now, this signal is to be entered in a function FC1 and normalized.

Next, the following is to be programmed: monitoring and displaying the maximum permissible level of 990 liters and monitoring the minimum permissible level of 110 liters.

Assignment list:

Address	Symbol	Data Type	Comment
%IW 64	AI_level_tank1	Int	Analog input level Tank1
%Q 0.0	Tank1_max	Bool	Display level > 990 liters
%Q 0.1	Tank1_min	Bool	Display level < 110 liters

6.1. Programming Level Monitoring for the SIMATIC S7-1200

The project is managed and programmed with the software '**Totally Integrated Automation Portal**'.

Here, under a uniform interface, the components such as the control system, visualization and networking the automation solution are set up, parameterized and programmed.

For error diagnosis, online tools are available.

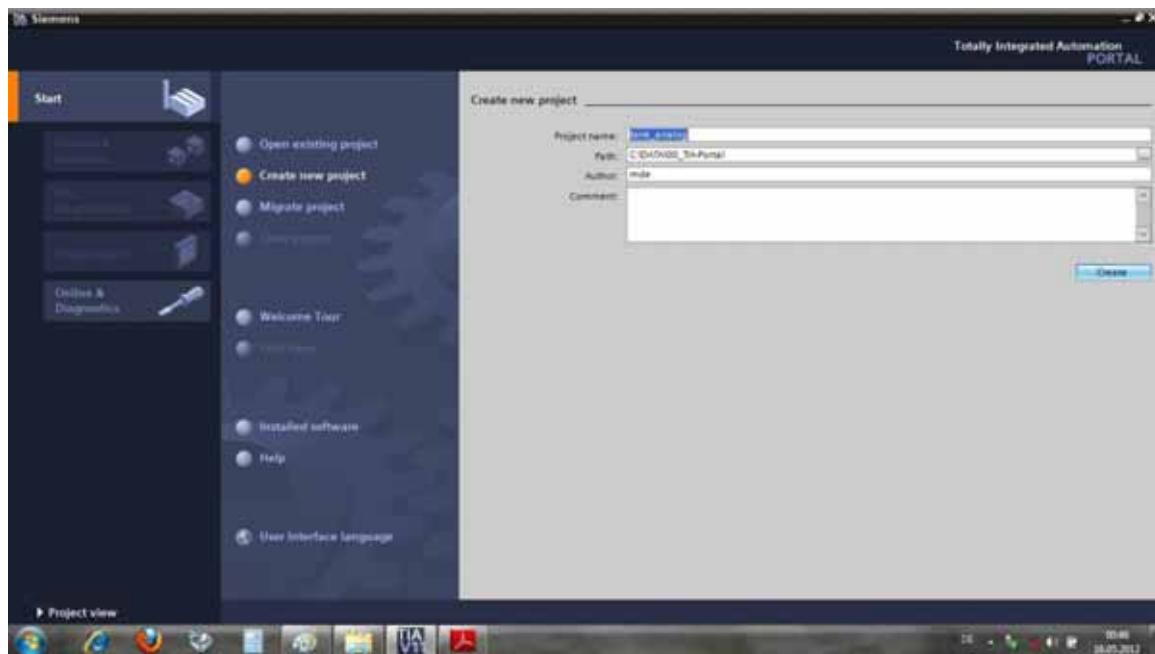
In the steps below, a project can be set up for the SIMATIC S7-1200 and the solution of the task can be programmed.

1. The central tool is the '**Totally Integrated Automation Portal**'. It is called here with a double click.
(→ **Totally Integrated Automation Portal V11**)



5

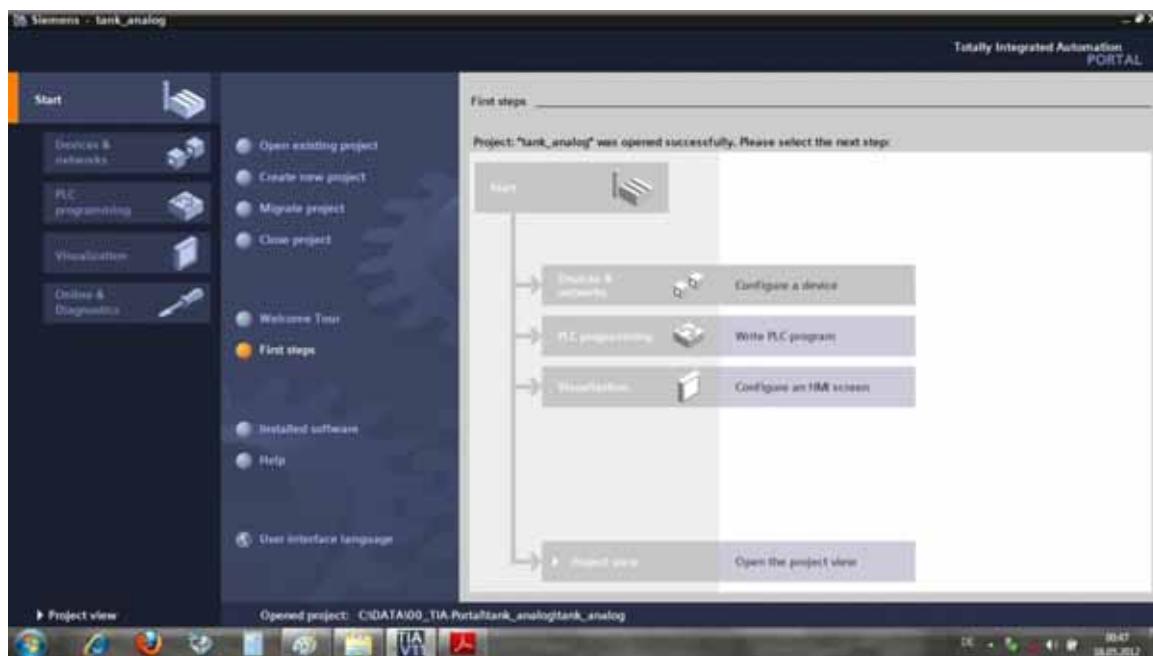
2. Programs for the SIMATIC S7-1200 are managed in projects. Such a project is now set up in the Portal View. (→ Create new project → Tank_Analog → Create)



5

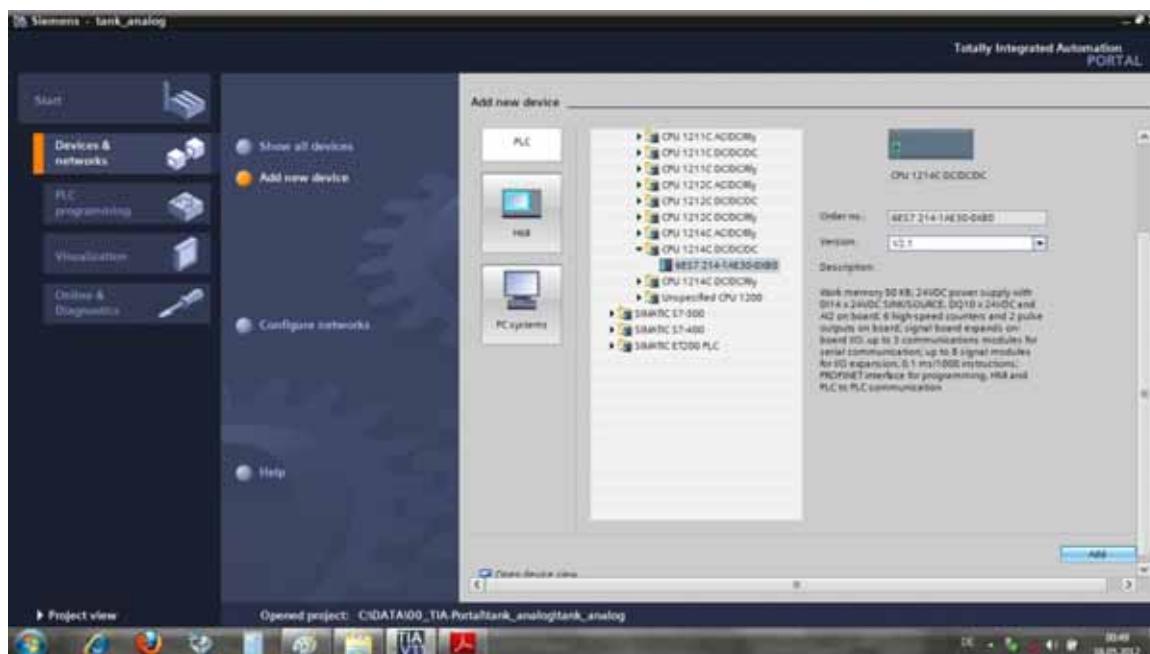
3. Now, '**First Steps**' are recommended for the configuration.

First, we want to '**Configure a device**'. (→ First steps → Configure a device)



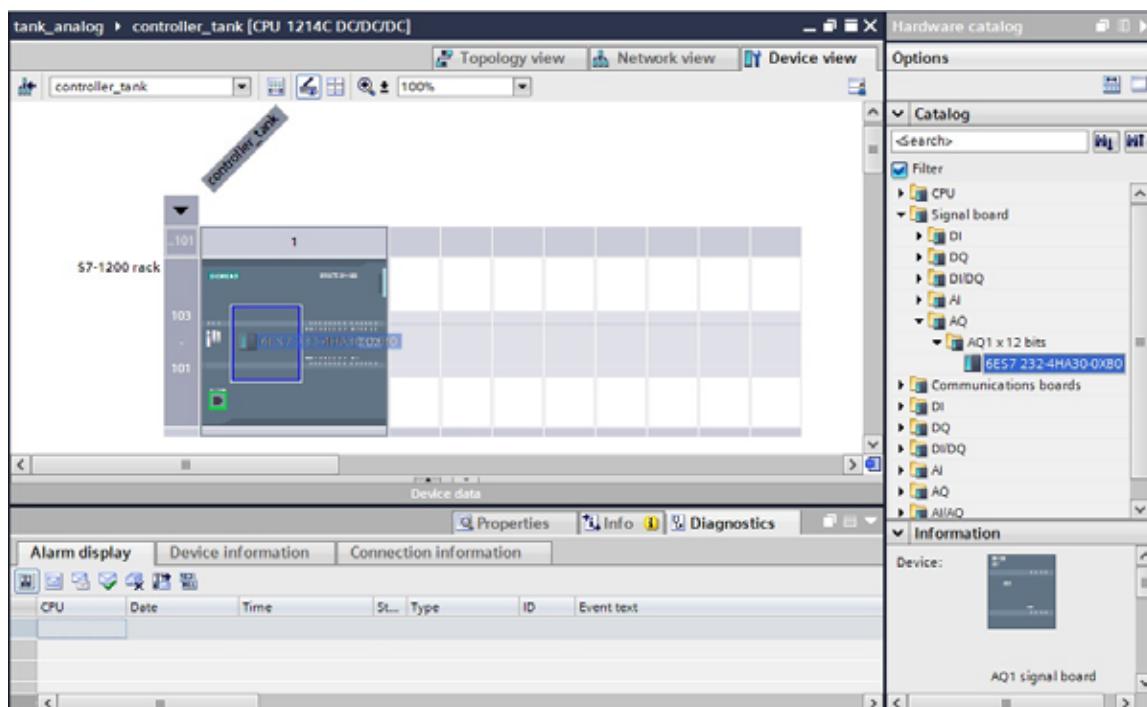
5

4. Then we '**Add new device**' with the device name "**controller_tank**". To this end, we select from the catalog '**CPU1214C**' with the matching order number. (→ Add new device → controller_tank → CPU1214C → 6ES7 → Add)

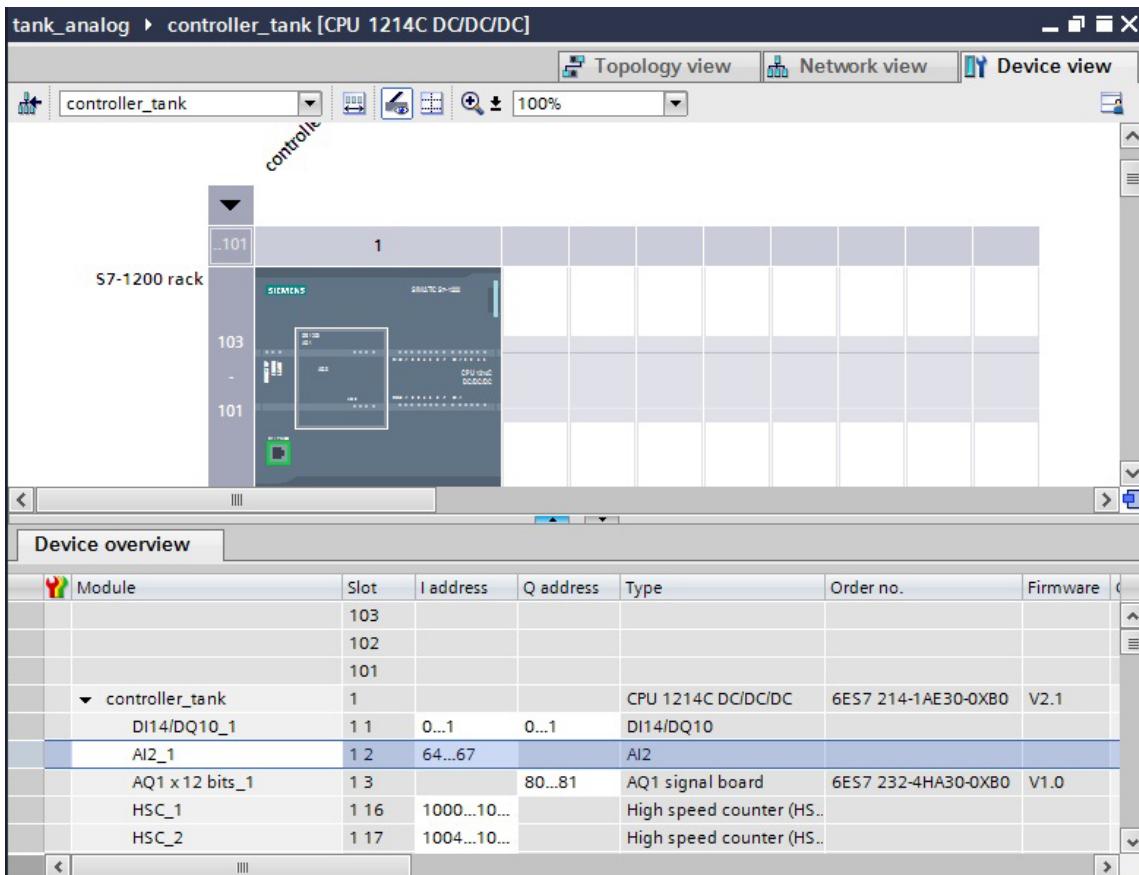


5

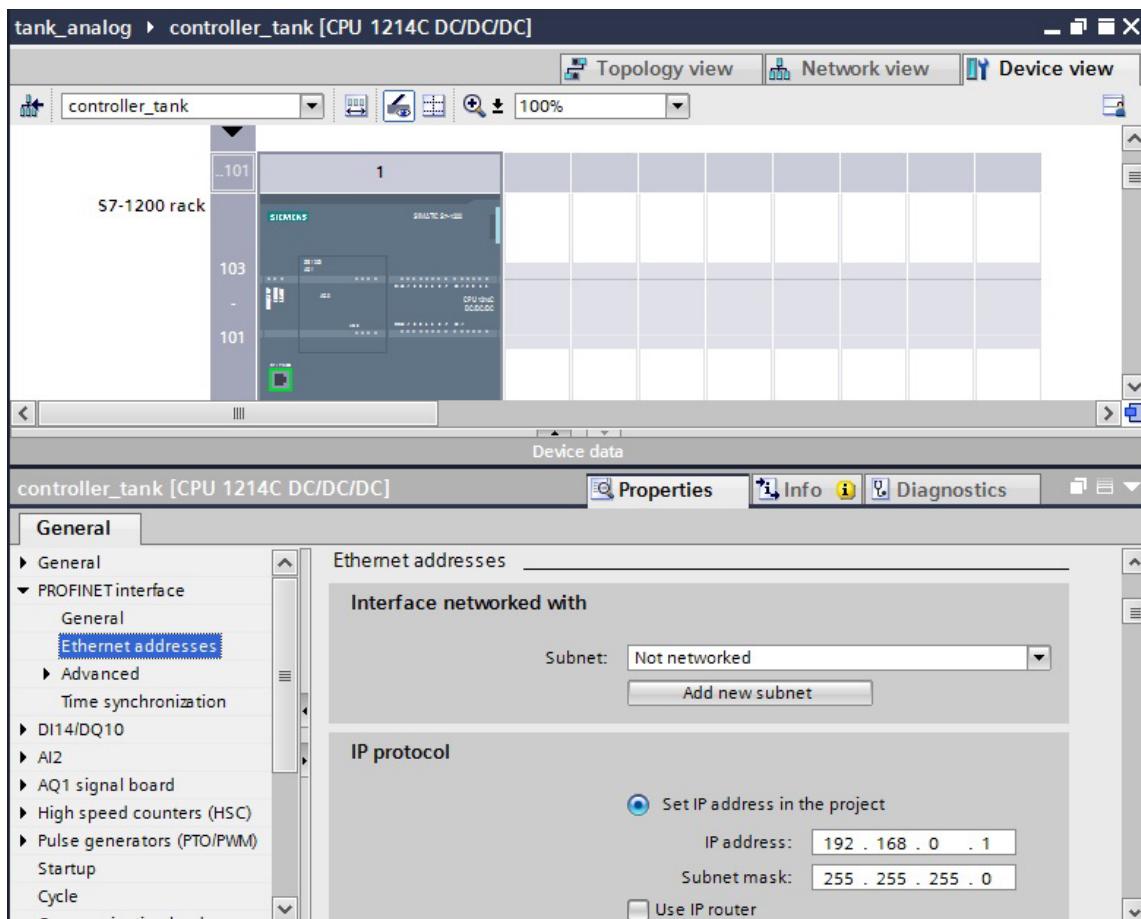
5. The software now changes automatically to the Project View with the opened hardware configuration. Here, more modules can be added from the hardware catalog (on the right) Here, the signal board for an analog output is to be inserted from the catalog using drag&drop. (→ Catalog → Signal board → AO1 x 12Bit → 6ES7 232-...)



6. In the '**Device view**', the addresses of the inputs and outputs can be checked or reset.
 Here, the integrated analog outputs of the CPU have the addresses %IW64 to %IW66 and the
 integrated digital outputs the addresses %Q0.0 to %Q1.1
 The analog output at the signal board has the address %QW80



7. So that the software later accesses the correct CPU, its IP address and the subnet mask have to be set. (→ Properties → General → PROFINET interface → Ethernet addresses →IP address: 192.168.0.1 → subnet mask: 255.255.255.0)



5

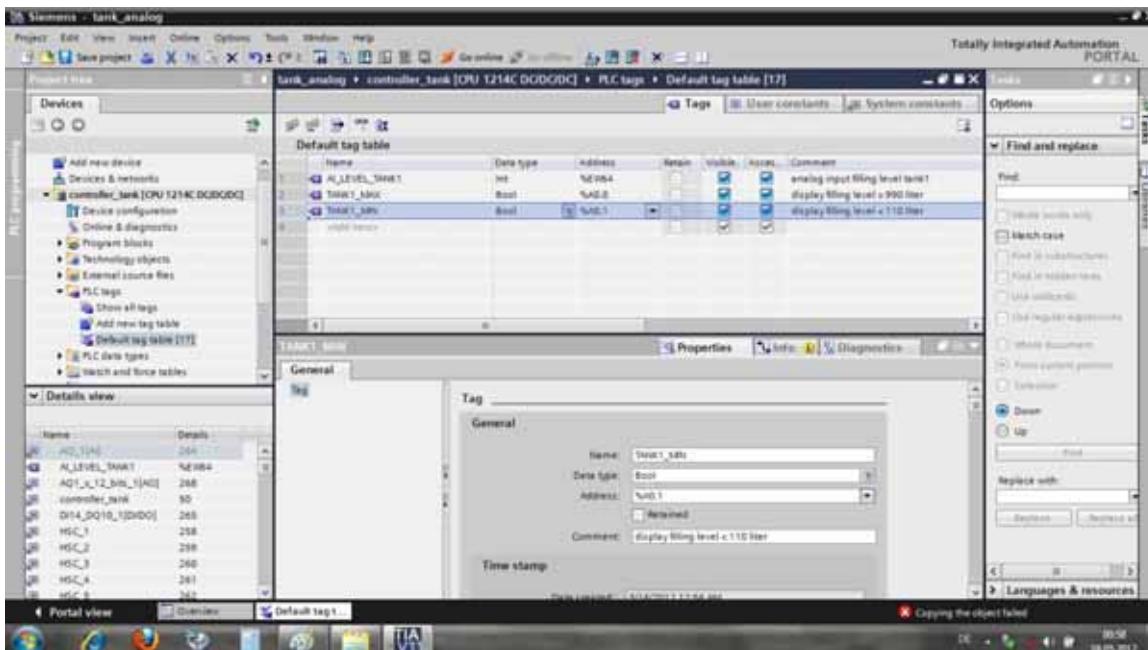
8. Since modern programming does not program with absolute addresses but with tags, the **global PLC tags** have to be specified here.

These global PLC tags are descriptive names with a comment for those inputs and outputs that are used in the program. Later, during programming, the global PLC tags can be accessed by means of this name.

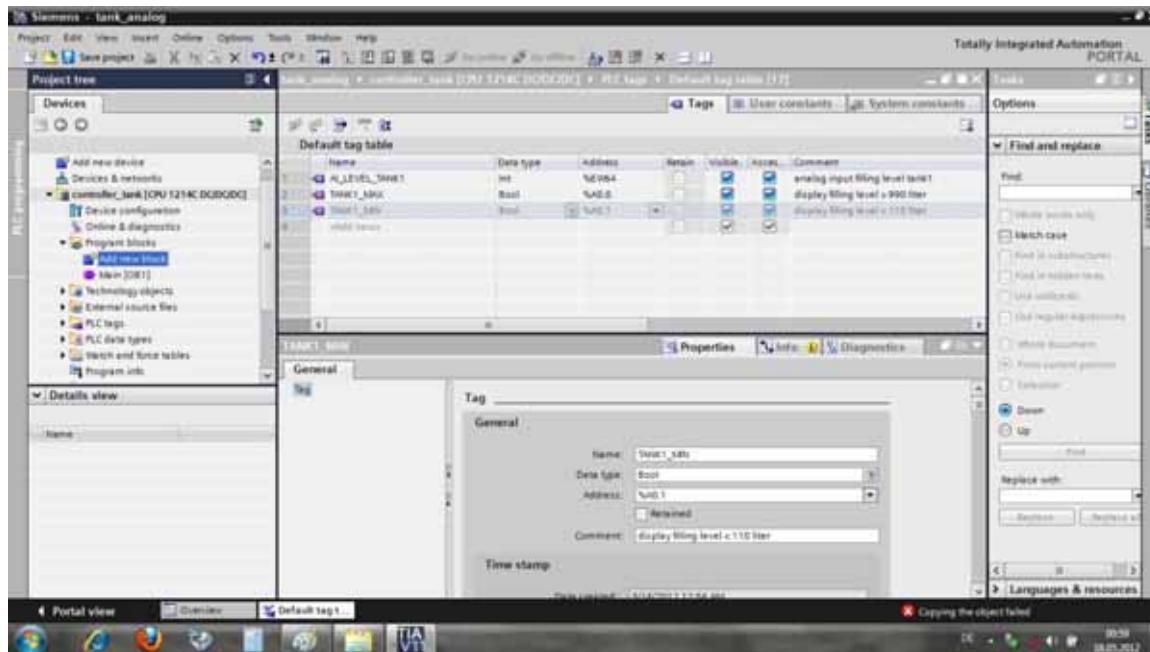
These global tags can be used in the entire program in all blocks.

To this end, select in project navigation '**controller_tank [CPU1214C DC/DC/DC]**' and then '**PLC tags**'. With a double click, open the table '**PLC tags**' and enter the names for the inputs and the outputs, as shown below.

(→ controller_tank[CPU1214C DC/DC/DC] → PLC tags → Default tag table)

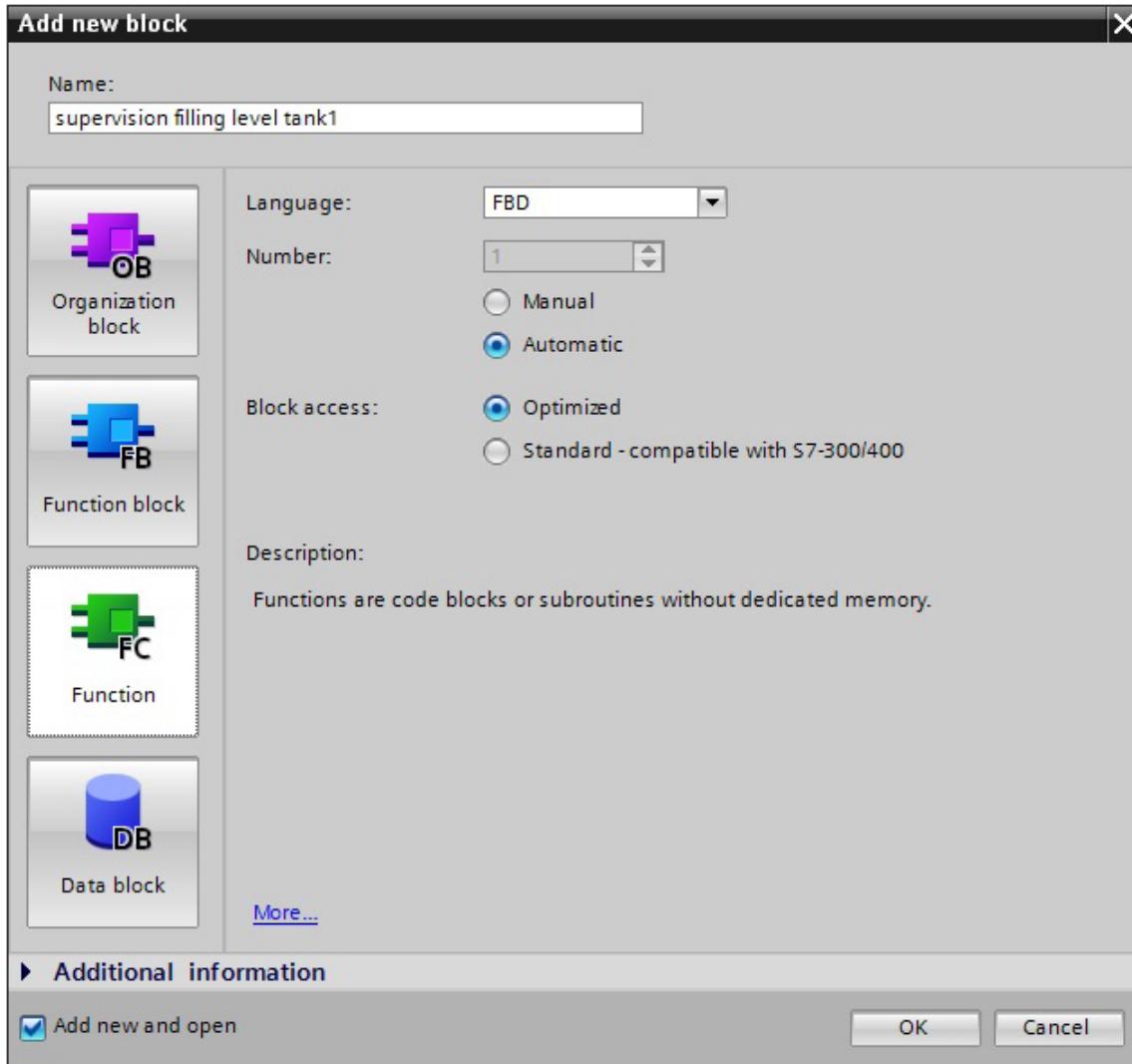


9. To create function block FC1, select in Project navigation **controller_tank [CPU1214C DC/DC/DC]** and then '**Program blocks**'. Next, double click on '**Add new block**'. (→ controller_tank[CPU1214C DC/DC/DC] → Program block → Add new block)



5

10. Select '**Function(FC)**' and assign the name '**supervision filling level tank1**'. As programming language, '**FBD**' (function block diagram) is specified. Numbering is automatic. Since this FC1 is called later by its symbolic name, the number is no longer that important. Accept the inputs with '**OK**'. (→ Function (FC1) → supervision filling level tank1 → FBD → OK)



5

11. The block ‘**supervision filling level tank1**’ [FC1]’ is opened automatically. Before writing the program, the interface of the block has to be declared. .
When the interface is declared, the local variables are specified that are known in this block. .

The variables consist of two groups:

- Block parameters that are the interface of the block for calls in the program.

Type	Name	Function	Available in
Input parameter	Input	Parameters whose values the block reads.	Functions, function blocks and some types of organization blocks
Output parameter	Output	Parameters whose values the block writes.	Functions and function blocks
InOut parameter	InOut	Parameters whose values the block reads when called, and after processing writes again to the same parameter.	Functions and function blocks

- Local data used for storing interim results

Type	Name	Function	Available in
Temporary local data	Temp	Variables used to store temporary interim results. Temporary data is retained for one cycle only.	Functions, function blocks and organization blocks
Static local data	Static	Variables used to store static interim results in the instance data block. Static data is retained -even over several cycles- until it is rewritten.	Function blocks

12. When local variables are declared, the following variables are needed for our example.

Input:

tank_level_AI Here, the level sensor enters the analog value

Output:

tank_max Here, the status of the maximum display is written to the output

tank_min Here, the status of the minimum display is written to the output

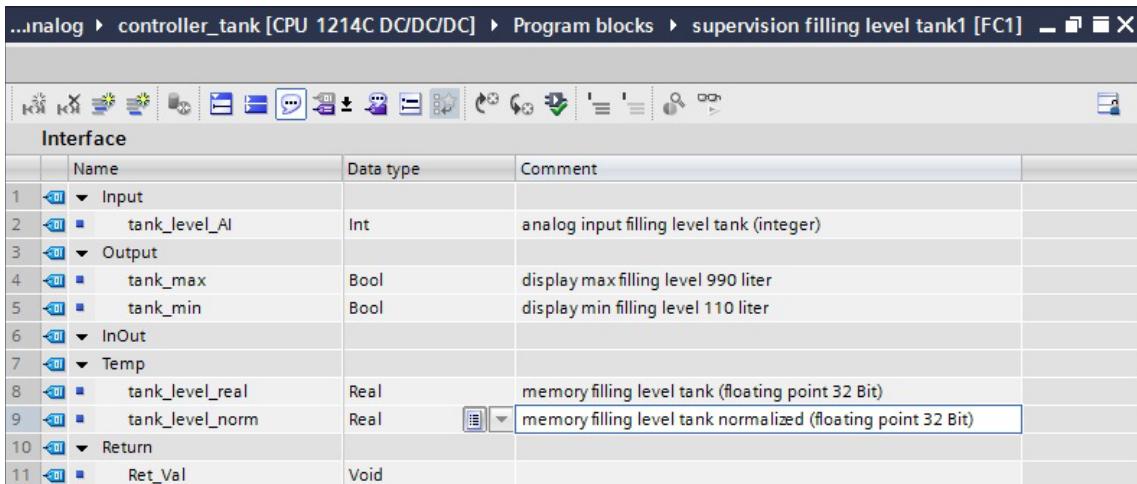
Temp:

tank_level_real This variable is needed to store an interim value

tank_level_norm Here, a value for the level, normalized in the floating point format to the range of 100 to 1000 liters is provided.

In this example, it is particularly important to use the correct data types since otherwise they would not be compatible in the following program with the conversion functions that are used. .

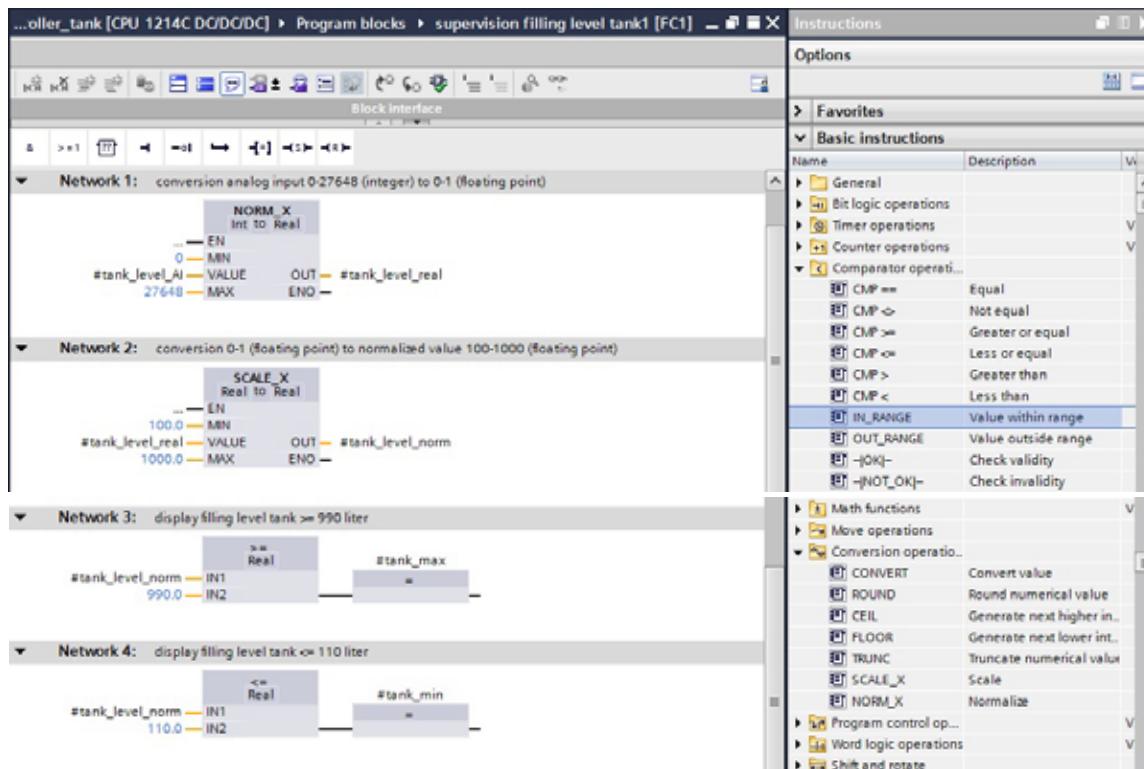
For the sake of clarity, all local variables should be provided with a sufficient comment.



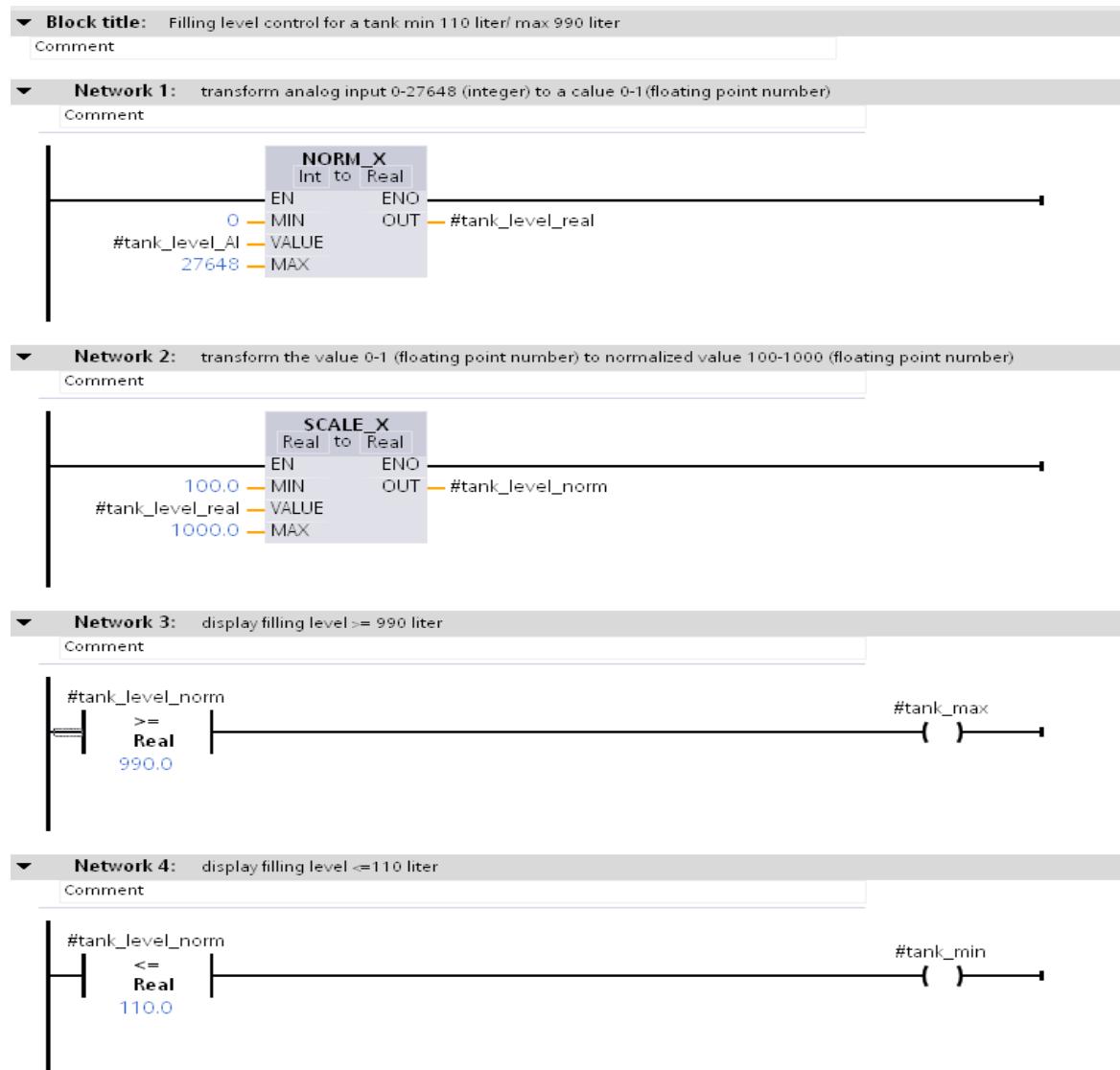
	Name	Data type	Comment
1	Input		
2	tank_level_AI	Int	analog input filling level tank (integer)
3	Output		
4	tank_max	Bool	display max filling level 990 liter
5	tank_min	Bool	display min filling level 110 liter
6	InOut		
7	Temp		
8	tank_level_real	Real	memory filling level tank (floating point 32 Bit)
9	tank_level_norm	Real	memory filling level tank normalized (floating point 32 Bit)
10	Return		
11	Ret_Val	Void	

13. After the local variables were declared, the program can be entered only by using the names of the variables (identified with the symbol '#'). For the example in **FBD**, it could look like this:

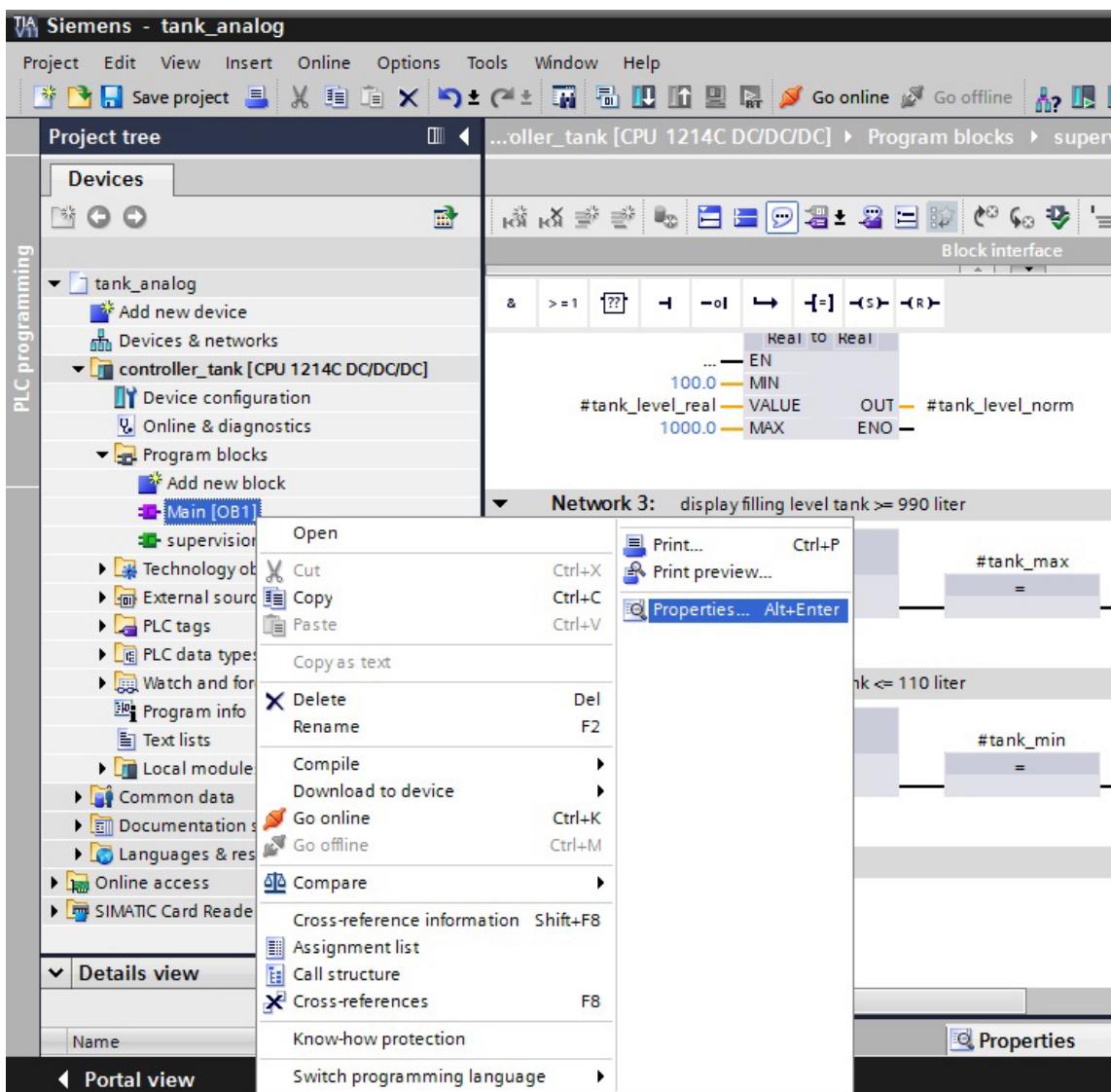
Program in function block diagram (FBD):



Program in ladder diagram (LAD):

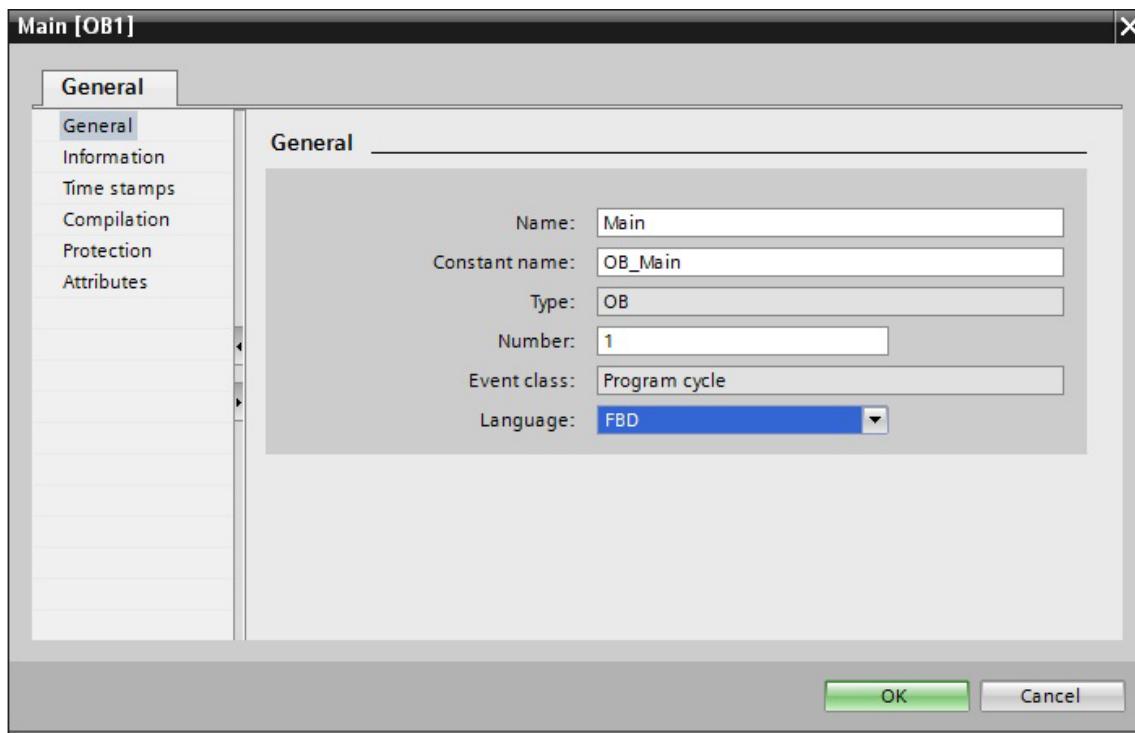


14. Next, the '**Properties**' of the block '**Main[OB1]**' that is processed cyclically are selected. Block properties can be changed. (→ Properties → Main[OB1])



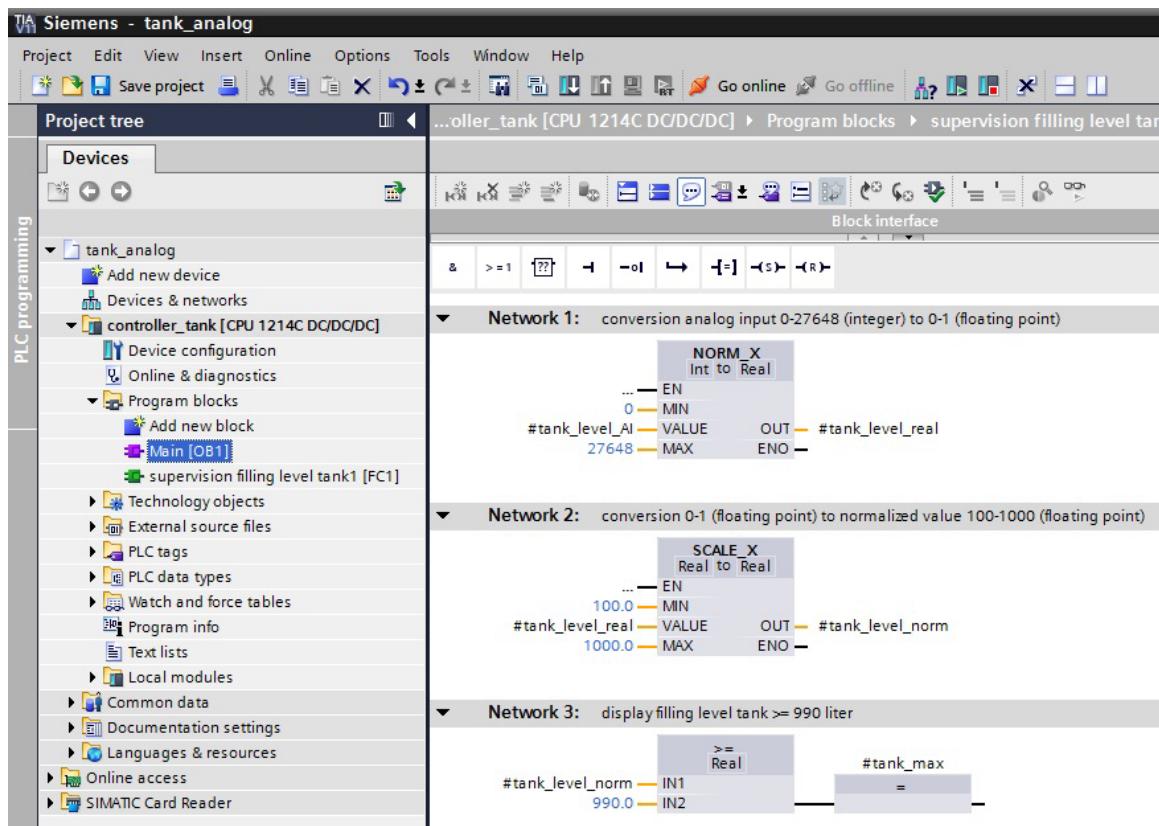
5

15. In the properties, select the programming ‘**Language**’ function block diagram ‘**FBD**’. (→ FBD → OK)



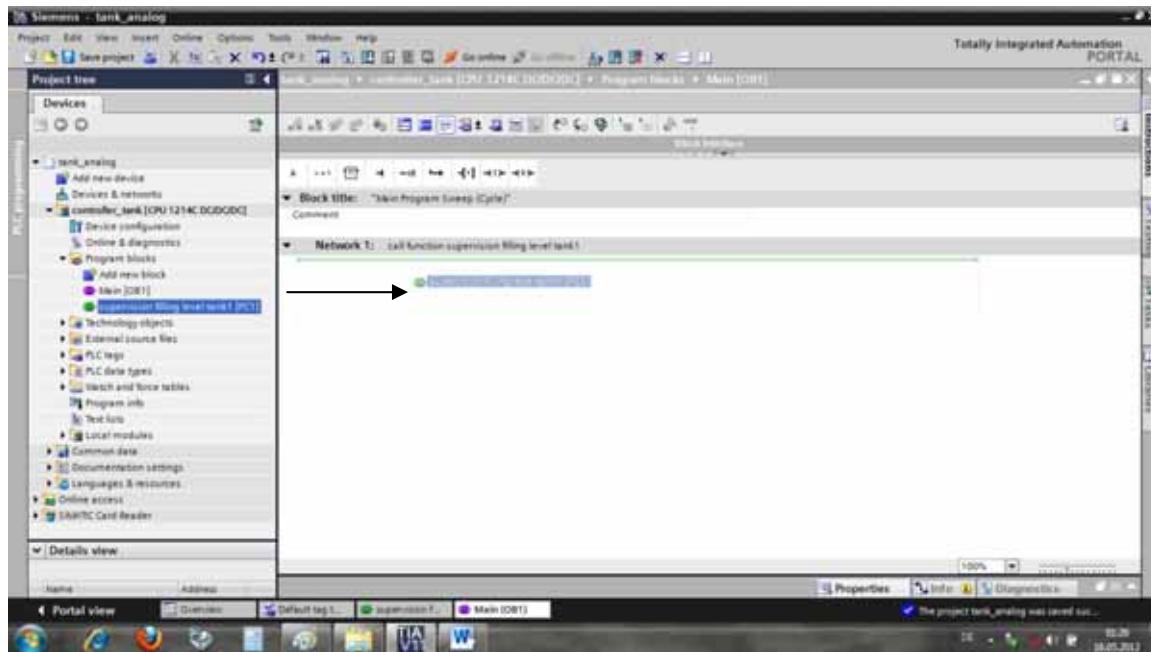
5

16. Now, the block "supervision filling level tank1 [FC1]" has to be called from the program block Main [OB1]. Otherwise, the block would not be processed. With a double click on 'Main [OB1]' open this block. (→ Main [OB1])



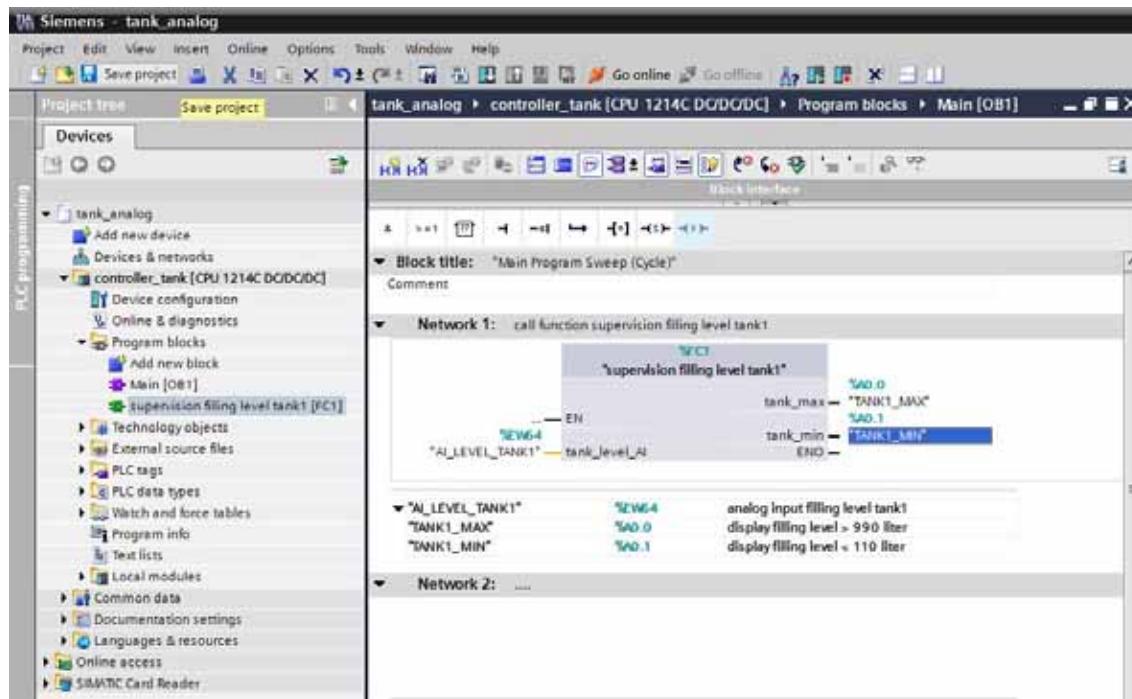
5

17. With drag&drop, the block "**supervision filling level tank1 [FC1]**" can then be dragged to Network 1 of the block Main [OB1]. Don't forget to document the networks also in the block Main [OB1].
 (→ supervision filling_level_tank1 [FC1])

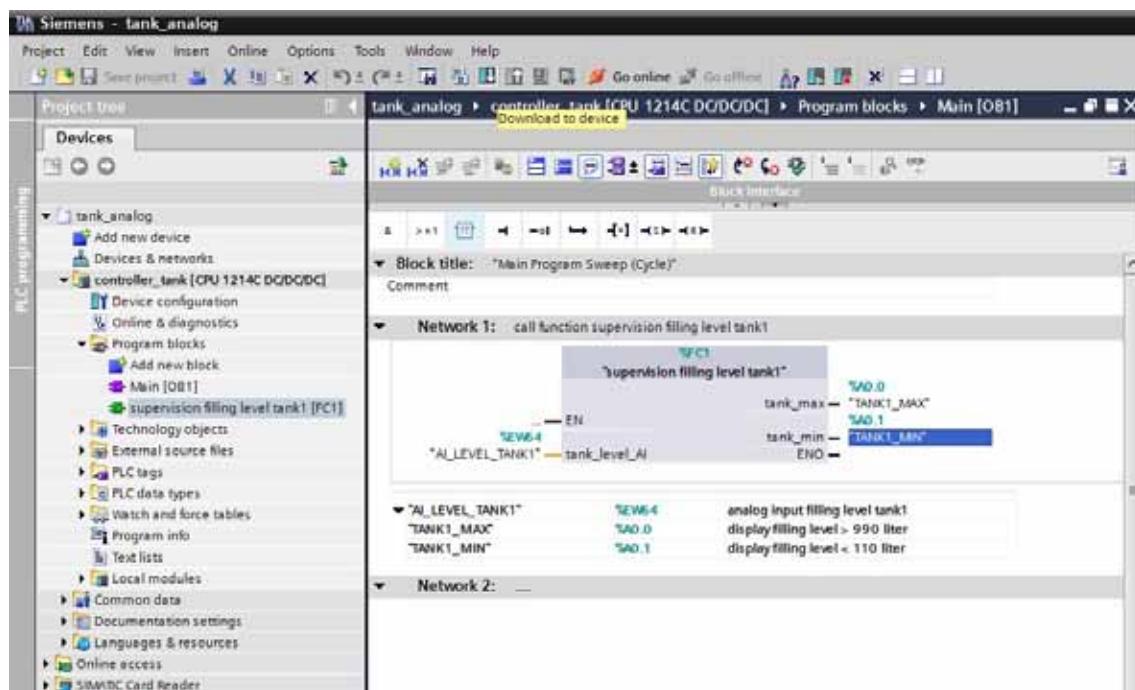


5

18. Next, the input variable as well as the output variable is wired in OB1 with the PLC tags shown here. Clicking on saves the project. (→ „AI_LEVEL_TANK1“ → "TANK1_MAX" → "TANK1_MIN" →

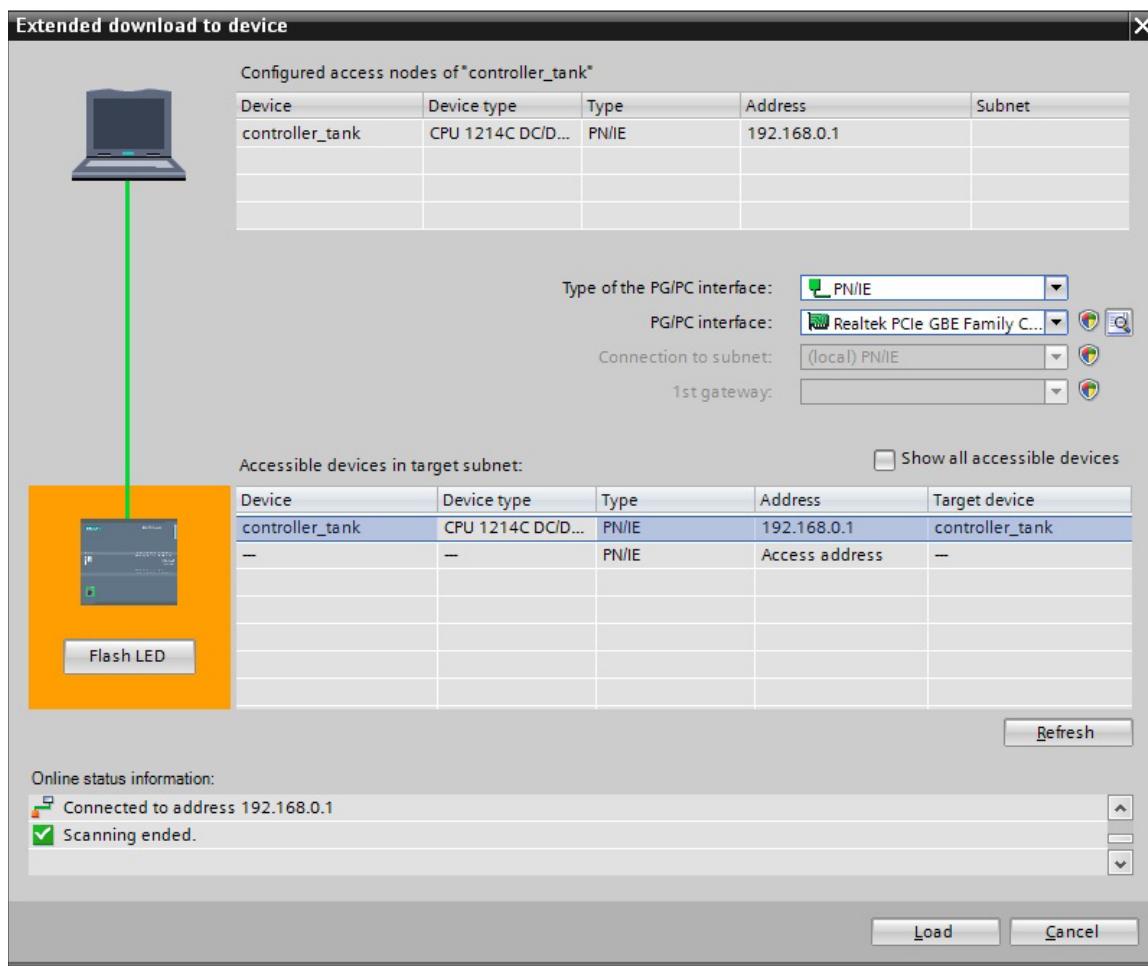


19. To load your entire program to the CPU, first highlight the folder '**controller_tank**' and then click on the symbol Load to device. (→ controller_tank →



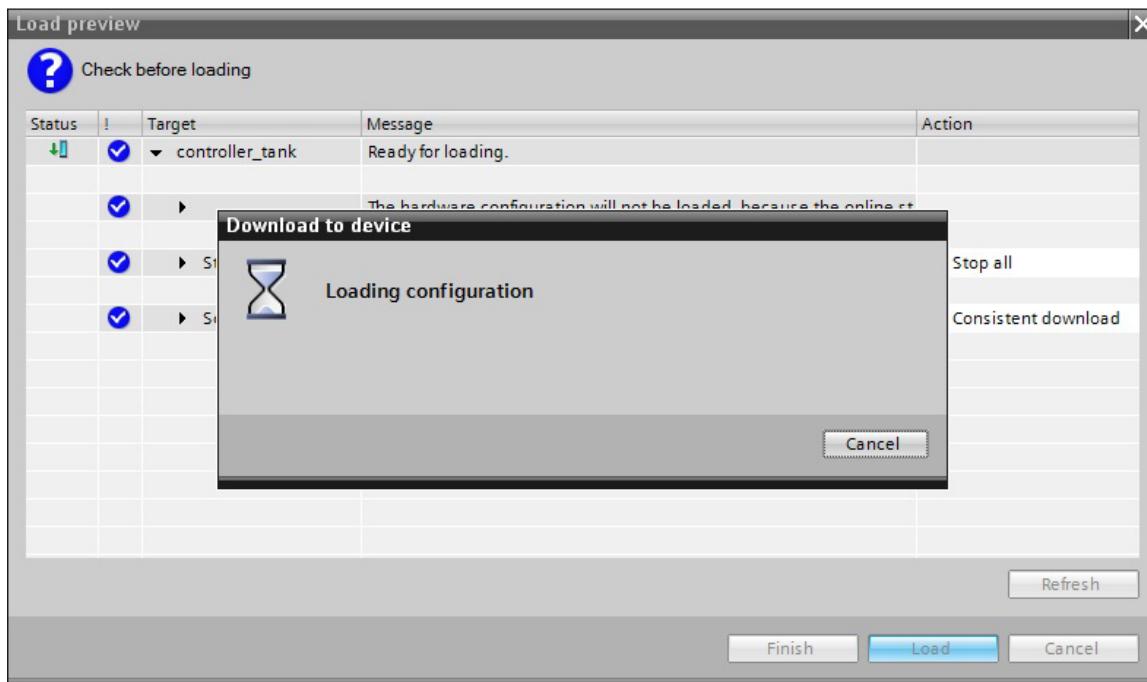
5

20. If the PG/PC interface was not specified previously, a window is displayed where this can still be done. (→ PG/PC interface for loading → load)

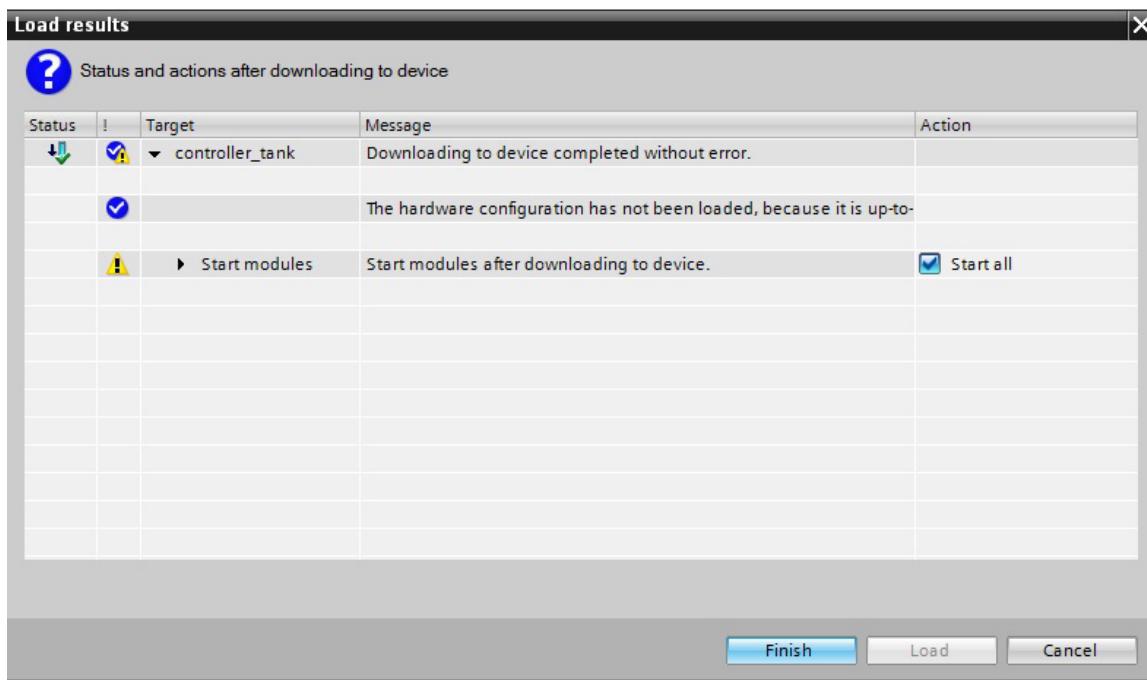


5

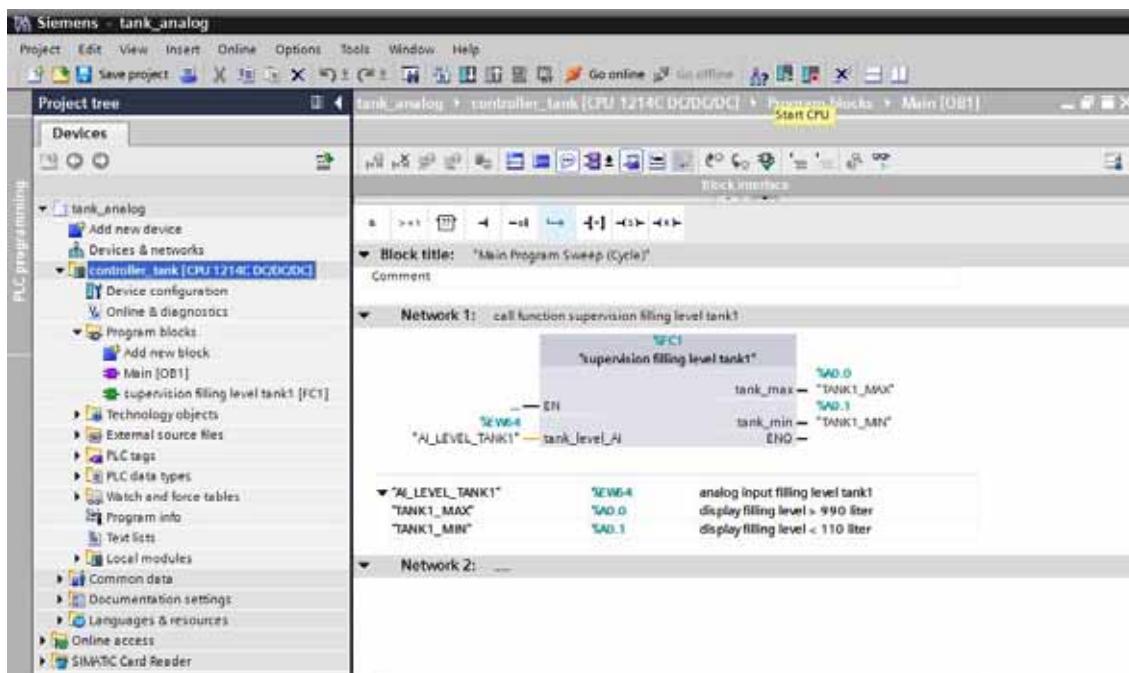
21. Then, click on 'Load' once more. During downloading, the status is displayed in a window. (→ Load)



22. Successful downloading is indicated in a window. Now click on 'Finish'. (→ Finish)

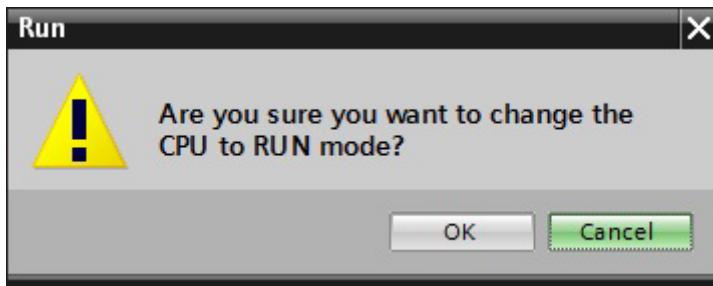


23. Next, start the CPU by clicking on the symbol . (→

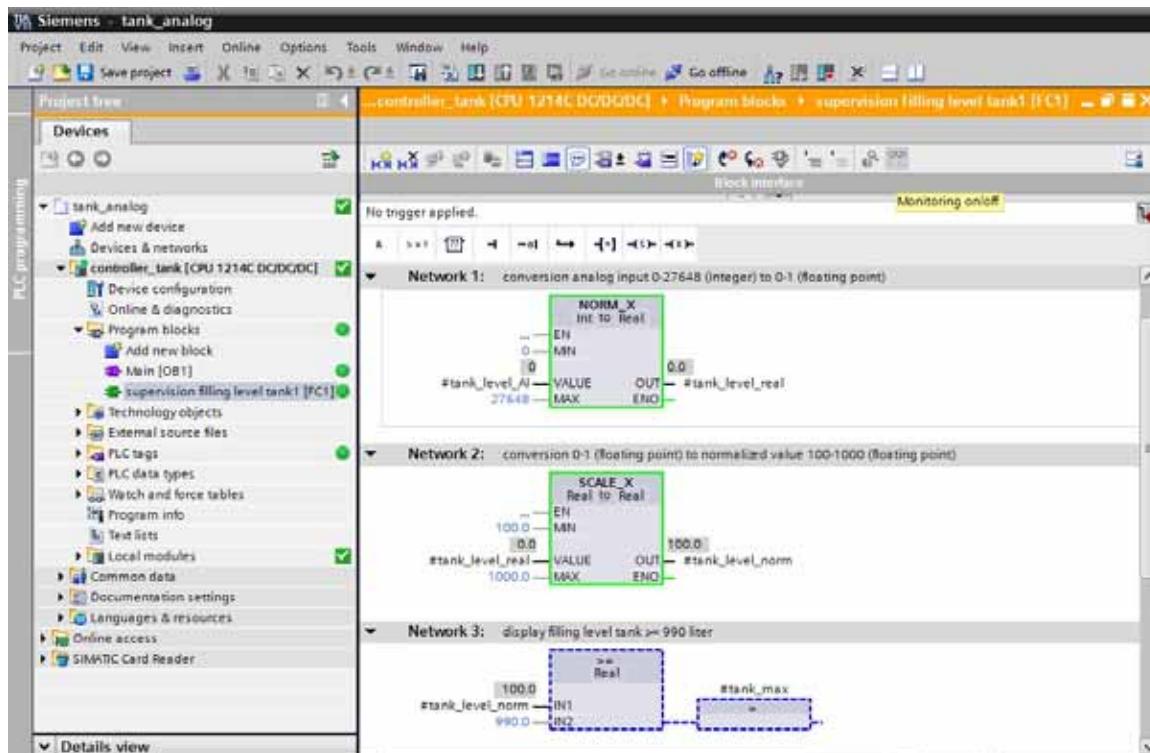


5

24. Confirm the question whether you actually want to start the CPU with 'OK'. (→ OK)



25. By clicking on the symbol  Monitoring on/off, the status of the variables can be monitored while the program is tested. (→ supervision filling level tank1[FC1] → 



Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Module 010-060



TIA Portal Module 010-010

Startup Programming of the
SIMATIC S7-1200

TIA Portal Module 010-020

Block Types for the SIMATIC S7-1200

TIA Portal Module 010-030

IEC Timers and IEC Counters at the
SIMATIC S7-1200

TIA Portal Module 010-040

Diagnosis and Error Search at the
SIMATIC S7-1200

TIA Portal Module 010-050

Analog Value Processing with
SIMATIC S7-1200

TIA Portal Module 010-060

Control Engineering with
SIMATIC S7-1200

TIA Portal Module 010-070

Communication between two
SIMATIC S7-1200

TIA Portal Module 010-080

HMI Panel KTP600 with SIMATIC
S7-1200

TIA Portal Module 010-090

Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

Matching SCE Training Packages for these documents

- **SIMATIC S7-1200 AC/DC/RELAIS 6er "TIA Portal"**
Order No.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
Order No.: 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
Order No. 6ES7822-0AA01-4YE0

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

Continued Training

For regional Siemens SCE continued training, please contact your regional SCE contact person
siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

6

Information regarding Usage

The training curriculum 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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by 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.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the documents being used commercially.

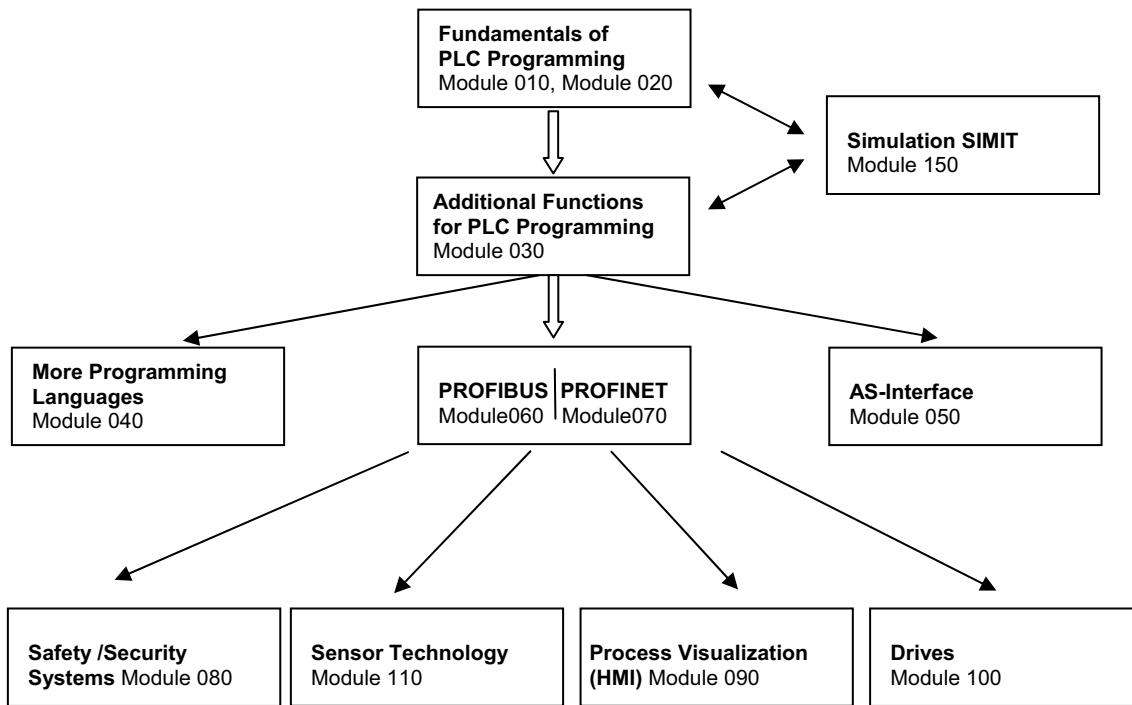
We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this document.

PAGE

1.	Preface	4
2.	Instructions for Programming the SIMATIC S7-1200	6
2.1	Automation System SIMATIC S7-1200.....	6
2.2	Programming Software STEP 7 Professional V11 (TIA Portal V11)	6
3.	Basics of Control Engineering.....	7
3.1	Tasks of Control Engineering.....	7
3.2	Components of a Closed Loop	8
3.3	Step Function to Examine Controlled Systems	11
3.4	Controlled Systems with Compensation	12
3.4.1	Proportional Controlled System without Time Delay.....	12
3.4.2	Proportional Controlled System without Time Delay.....	13
3.4.3	Proportional Controlled System with Two Time Delays	14
3.4.4	Proportional Controlled System with n Time Delays.....	15
3.5	Controlled Systems without Compensation	16
3.6	Basic Types of Continuous Action Controllers.....	17
3.6.1	The Proportional Controller (P-Controller).....	18
3.6.2	The Integral Controller (I-Controller)	20
3.6.3	The PI Controller	21
3.6.4	Differential Action Controller (D-Controller).....	22
3.6.5	The PID Controller.....	22
3.7	Objectives for Setting the Controller	23
3.8	Setting Controlled Systems.....	25
3.8.1	Setting the PI Controller according to Ziegler-Nichols	26
3.8.2	Setting the PI Controller according to Chien, Hrones and Reswick.....	26
3.9	Digital Controllers.....	28
4.	Sample Task - Level Control for a Tank	30
5.	Programming the Level Control for the SIMATIC S7-1200.....	31

1. Preface

Regarding its content, module 010-060 is part of the training unit '**Basics of PLC Programming**' and explains how to program a PID controller with the SIMATIC S7-1200 with the TIA Portal.



Training Objective

In this module 010-060, the reader learns how to program a PID controller with the SIMATIC S7-1200 with the programming tool TIA Portal. The module provides the fundamentals and shows how it is done using a detailed example.

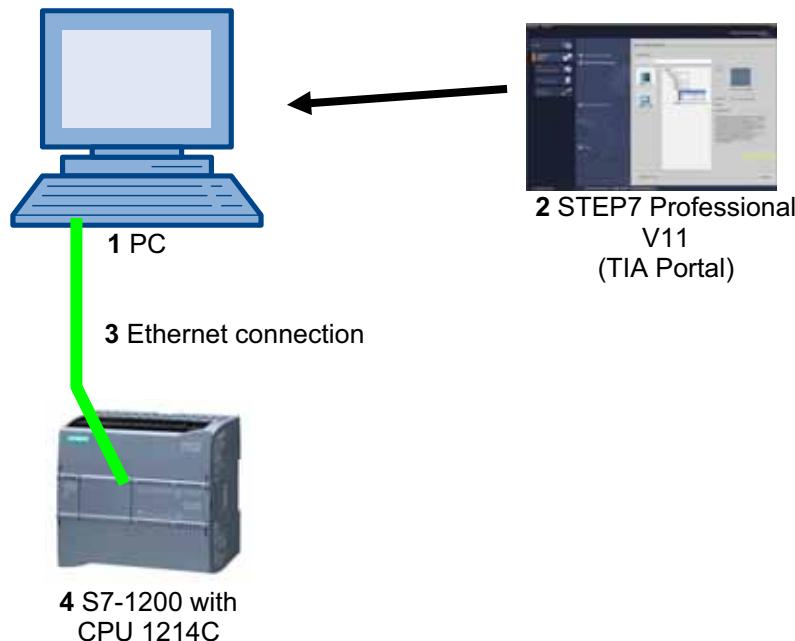
Prerequisites

To successfully work through this module 010-060, the following knowledge is assumed:

- How to handle Windows
- Basics of PLC programming with the TIA Portal
(for example, Module 010-010 – 'Startup' Programming of the SIMATIC S7-1200 with TIA- Portal V11)
- Blocks for the SIMATIC S7-1200
(for example, Module 010-020 - Block Types for the SIMATIC S7-1200)
- Analog value processing with the SIMATIC S7-1200
(for example, Module 010-050 - Analog Value Processing with the SIMATIC S7-1200)

Hardware and software required

- 1** PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB
Operating system Windows XP Professional SP3/Windows 7 Professional/Windows 7 Enterprise/
Windows 7 Ultimate/Windows 2003 Server R2/Windows Server 2008
Premium SP1, Business SP1, Ultimate SP1
- 2** Software STEP7 Professional V11 SP1 (Totally Integrated Automation (TIA) Portal V11)
- 3** Ethernet connection between PC and CPU 315F-2 PN/DP
- 4** PLC SIMATIC S7-1200; for example, CPU 1214C.
The inputs have to be brought out to a panel.



2. Instructions for Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The automation system SIMATIC S7-1200 is a modular mini-controller system for the lower and medium performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU and input/output modules for digital and analog signals.

If needed, communication processors and function modules are added for special tasks such as step motor control.

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process, whereby the IO modules are polled in the S7 program by means of the input addresses (%I) and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7.

2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 (TIA Portal V11) is the programming tool for the following automation systems

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining the communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating the visual displays for the SIMATIC basic panels with the integrated WinCC Basic
- With additional WinCC packages, visual display solutions for PCs and other panels can be prepared also.

All functions are supported with detailed online help.

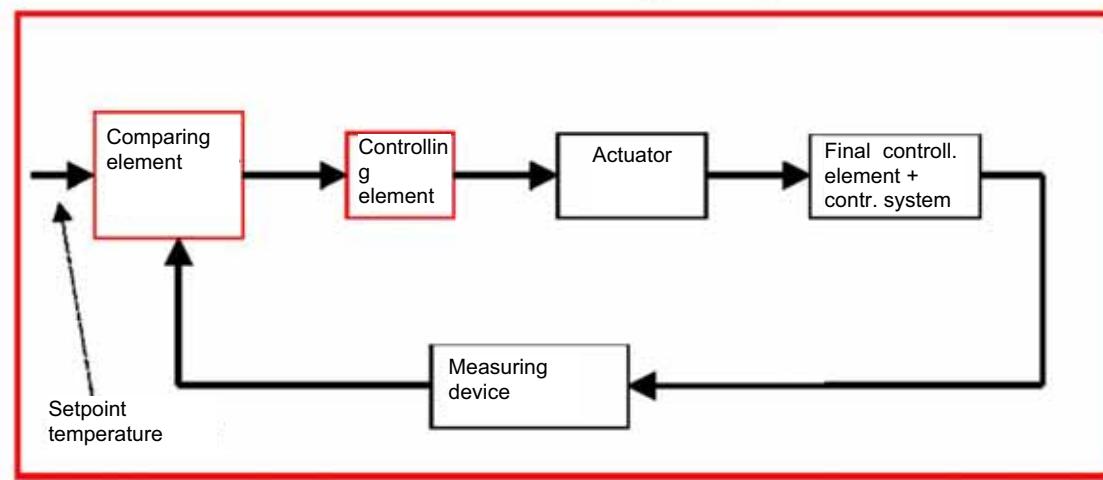
3. Basics of Control Engineering

3.1 Tasks of Control Engineering

"Controlling is a process where the value of a variable is established and maintained through the intervention of measurements of this variable. An action sequence occurs that takes place in a closed loop, because the process transpires based on measurements of a variable that in turn is influenced by itself."

The variable to be controlled is measured continuously and compared with another specified variable of the same kind. Depending on the result of this comparison, the control process adjusts the variable to be controlled to the value of the specified variable.

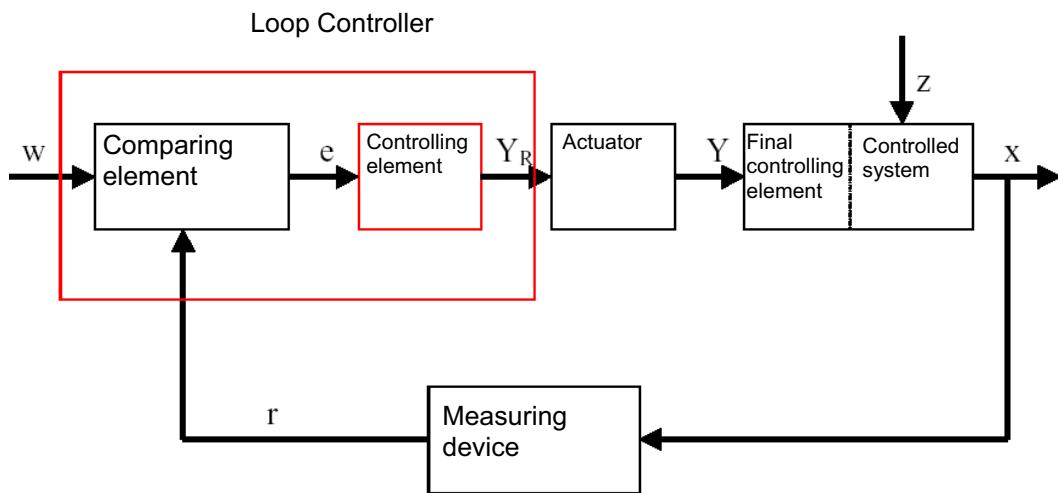
Diagrammatic Representation of a
Closed Loop Control



3.2 Components of a Closed Loop

Below, the basic terms of control engineering are explained in detail.

The diagram below provides an overview:



1. The controlled variable x

It is the actual "target" of the control system: to influence the variable and to keep it constant is the purpose of the entire system. In our example, this would be the room temperature. The current value of the controlled variable existing at a certain point in time is called the "actual value" at that time.

2. The feedback variable r

In a closed loop, the controlled variable is constantly checked in order to respond to unintended changes. The measured value proportional to the controlled variable is called feedback variable. In the example "heater" it would correspond to the measured voltage of the inside thermometer.

4. Disturbance Variable z

The disturbance variable is the variable that influences the controlled variable unintentionally and moves it away from the current setpoint. If there is a fixed setpoint control, it is necessary only because of the existence of the disturbance variable. In our 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.

5. Setpoint w

The setpoint at this time is the value that the controlled variable should ideally be at this time. It should be noted that the setpoint in the case of a slave value control may change continuously under certain circumstances. The measured value that would be ascertained by the measuring device if the controlled variable had exactly the setpoint is the current value of the reference variable. In our example, that would be the setpoint of the room temperature that is presently desired.

6. Comparing element

This is the point where the measured value of the controlled variable and the current value of the reference variable are compared. In most cases, the two variables are measurement voltages. The difference of the two variables is the "control deviation" e . It is passed on to the controlling element and evaluated there (see below).

7. Controlling element

The controlling element is the core of a control system. It evaluates the control deviation -i.e., the information as to whether, how and to what extent the controlled variable deviates from the current setpoint- as input variable and derives from it the "**controller output variable**" Y_R that ultimately influences 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 control deviation is the main criterion of the control system. In Part II, this topic is discussed in greater detail.

8. Actuator

The actuator is the "executing agent" of the control system. In the form of the controller output variable, the controlling element provides the sensor with information as to how the controlled variable is to be influenced and implements this information as a change of the "manipulated variable". In our example, the actuator would be the mixer motor. Depending on the voltage provided by the controlling element, (i.e., the controller output variable), it influences the position of the mixer (which here represents the manipulated variable).

9. Final controlling element

This is the element of the control loop that influences the controlled variable (more or less directly) depending on the **manipulated variable Y**. In the example, this would be a combination consisting of the mixer, heating lines and heater. The mixer motor (actuator) sets the mixer (manipulated variable) which influences the room temperature by means of the water temperature.

10. Controlled system

The controlled system is the system where the variable to be controlled is located; in the example of the heater: the room.

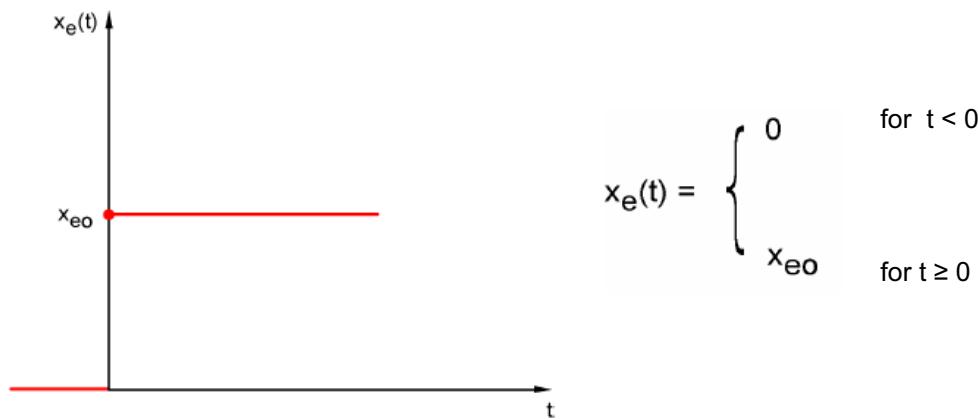
6

11. Dead time

Dead time is the time that passes starting with the change of the controller output variable until the measurable reaction of the controlled system. In our example, this would be the time between a voltage change for the mixer motor and a measurable change of the room temperature caused by this.

3.3 Step Function to Examine Controlled Systems

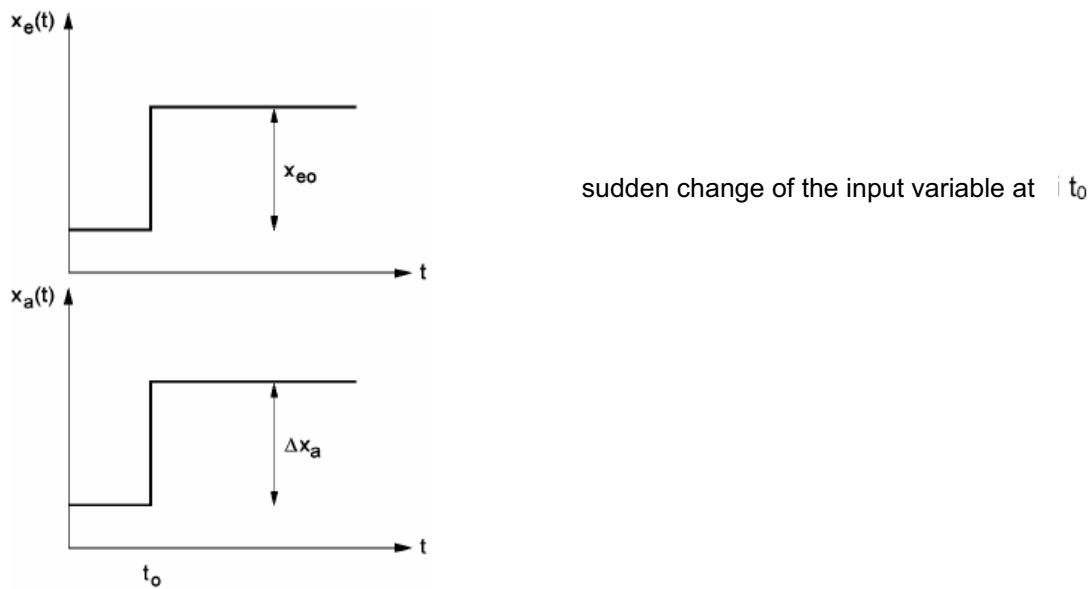
To examine the behavior of controlled systems, controllers and control loops, a uniform function is used for the input signals: the step function. Depending on whether a control loop element or the entire control loop is examined, the controlled variable $x(t)$, the manipulated variable $y(t)$, the reference variable $w(t)$ or the influencing variable $z(t)$ can be assigned the step function. For that reason, the input signal, the step function, is often designated $x_e(t)$ and the output signal $x_a(t)$.



3.4 Controlled Systems with Compensation

3.4.1 Proportional Controlled System without Time Delay

The controlled system is called P-system for short.



6

Controlled variable/manipulated variable:

$$x = K_{ss} \cdot y$$

K_{ss} : Proportional coefficient for a manipulated variable change:

$$K_{ss} = \frac{\Delta x}{\Delta y} = \tan \alpha$$

Controlled variable/disturbance variable:

$$x = K_{sz} \cdot z$$

K_{sz} : Proportional value for a disturbance variable change

Range:

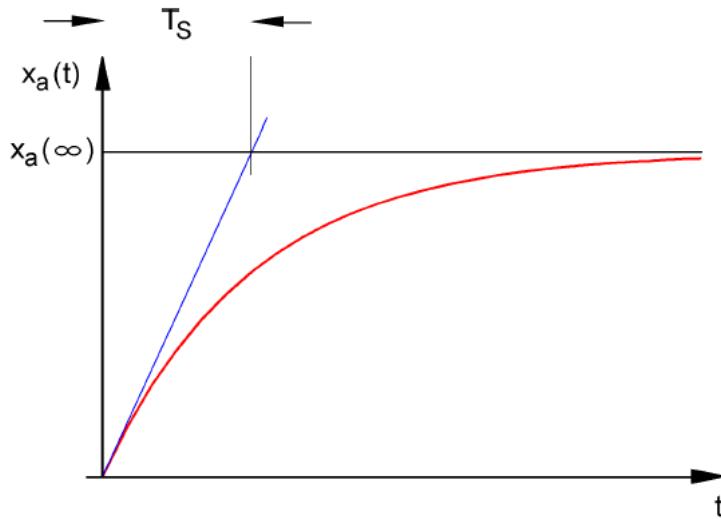
$$y_h = y_{\max} - y_{\min}$$

Control range:

$$x_h = x_{\max} - x_{\min}$$

3.4.2 Proportional Controlled System without Time Delay

The controlled system is called P-T1 system for short.



6

Differential equation for a general input signal $x_e(t)$:

$$T_S \cdot \dot{x}_a(t) + x_a(t) = K_{PS} \cdot x_e(t)$$

Solution of the differential equation for a step function at the input (step response)

$$x_a(t) = K_{PS} (1 - e^{-t/T_S}) \cdot x_{eo}$$

$$x_a(t = \infty) = K_{PS} \cdot x_{eo}$$

T_S : Time constant

3.4.3 Proportional Controlled System with Two Time Delays

The controlled system is called P-T2 for short.

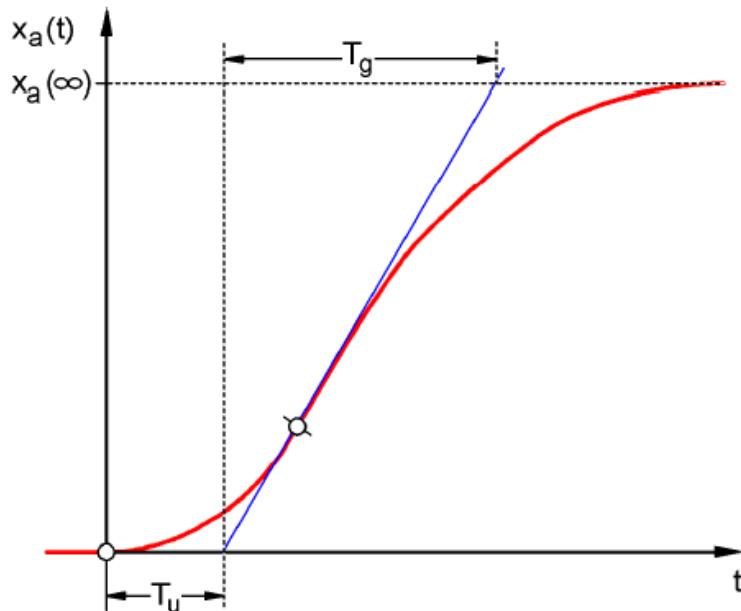


Figure: Step Response of the P-T2 System

T_u: Delay time T_g: 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:

$$\frac{T_u}{T_g} < \frac{1}{10} \rightarrow \boxed{\text{easily controllable}} \quad \frac{T_u}{T_g} \approx \frac{1}{6} \rightarrow \boxed{\text{still controllable}} \quad \frac{T_u}{T_g} > \frac{1}{3} \rightarrow \boxed{\text{difficult to control}}$$

With the increasing ratio T_u/T_g , the system becomes less and less controllable.

3.4.4 Proportional Controlled System with n Time Delays

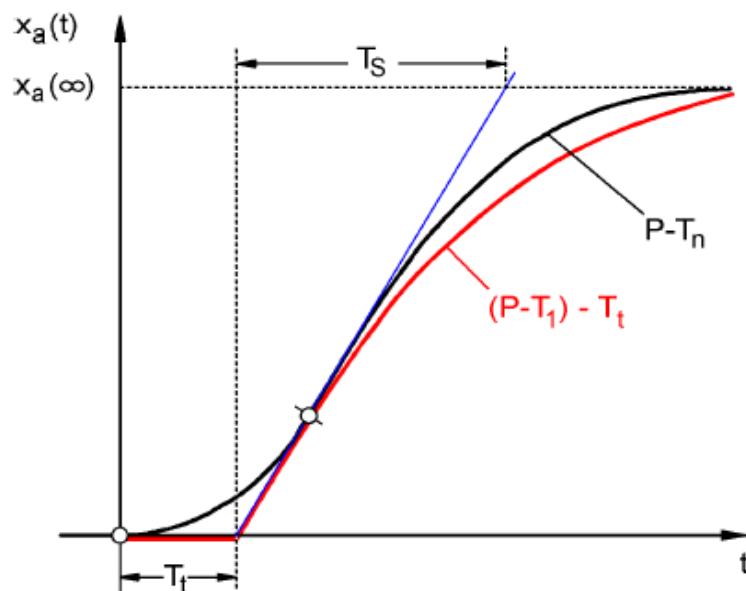
The controlled system is called P-T_n system for short.

The time response is described with a differential equation of the nth degree.

The characteristic of the step response is similar to the P-T₂ system. The time response is described with T_u and T_g.

Substitution: The controlled system with many delays can be replaced approximately with the series connection of a P-T₁ system with a dead time system.

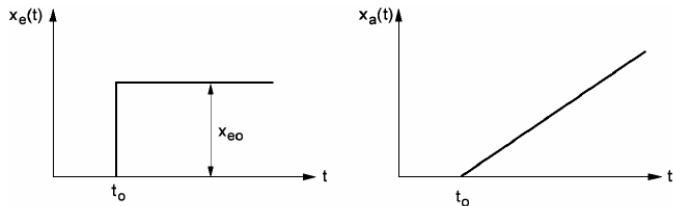
The following applies: T_t » T_u and T_S » T_g.



Substitute step response for the P-T_n system

3.5 Controlled Systems without Compensation

After a disturbance, the controlled variable continues to increase without trying to reach a fixed final value.

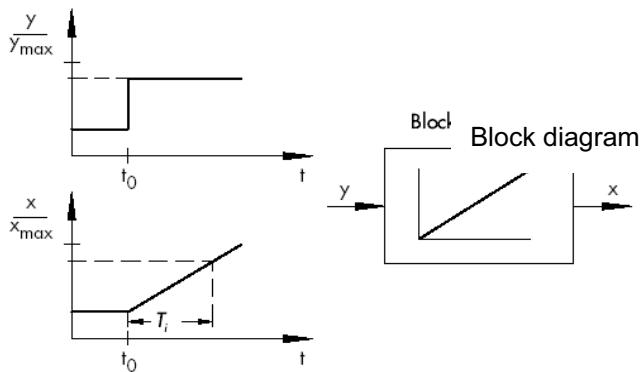


Example: Level control

A container with a drain whose inflow and outflow volume is equal will reach a constant level. If the inflow or the outflow changes, the level of the liquid rises or drops. The level changes the faster, the larger the difference between inflow and outflow.

The example shows that in practice, the integral action usually has a limit. The controlled variable rises or drops only until it reaches a limit depending on the system: the container overflows or it empties, the pressure reaches the plant maximum or minimum, etc..

The figure shows the time behavior of an I-system whose input variable changes abruptly, and the diagram derived from this.



If the step function transitions at the input to any function $x_e(t)$, 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 taken from SAMSON Technical Information - L102 – Controllers and Controlled Systems, Edition August 2000
[\(http://www.samson.de/pdf_en/l102en.pdf\)](http://www.samson.de/pdf_en/l102en.pdf)

3.6 Basic Types of Continuous Action Controllers

The controllers discussed above have -as mentioned previously- the advantage of simplicity.

The controller itself as well as the actuator and the final controlling element are of a simple nature and therefore less expensive than for stepless controllers. However, discrete controllers have a number of disadvantages. For example, if large loads such as large electrical motors or cooling units have to be switched on, peak loads may stress the power supply.

For this reason, often we do not switch between "Off" and "On", but between full (full load), and an appreciably lower performance of the actuator/final controlling element (basic load). But even with this improvement, continuous control is not suitable for many applications. Imagine a car engine whose speed is controlled directly. There would be nothing between idle and full throttle.

Besides it being impossible to transfer the power of a sudden full throttle suitably via the tires to the road, such a car would probably be highly unsuitable for street traffic. For such applications continuous action controllers are used for that reason.

In that case, the mathematical relationship that the controlling element establishes between the system deviation and the controller output variable has virtually no limits theoretically. In practice, however, we differentiate among three classical basic types that are discussed in greater detail below.

3.6.1 The Proportional Controller (P-Controller)

In the case of a P-controller, the manipulated variable y is always proportionate to the recorded control deviation ($y \sim e$). The result is that a P-controller responds to a system deviation without delay and generates a manipulated variable only if there is a deviation e .

The proportionate pressure regulator shown in the figure below compares the force FS of the setpoint spring with the force FB that the pressure p_2 generates in the spring-elastic metal bellows

If the forces are not in balance, the lever rotates around the pivot point **D**. In the process, the valve position \bar{n} changes and correspondingly the pressure p_2 to be controlled until a new balance of forces is established.

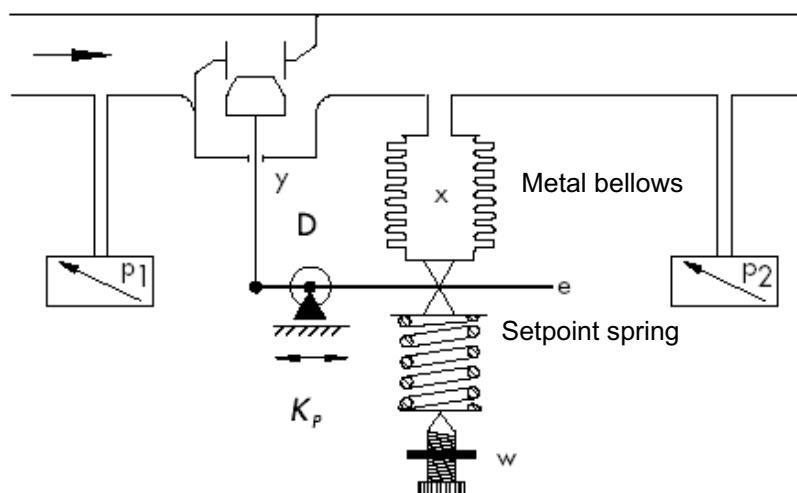
The figure shows the behavior of the P-controller as a sudden control deviation occurs.

The amplitude of the manipulated variable step y depends on the value of the control deviation e and the amount of the proportional coefficient K_p :

To keep the deviation low, a proportional factor that is as large as possible has to be selected.

Enlarging the factor causes the controller to respond faster. However, a value that is too high can cause overshoot and a pronounced hunting tendency on the part of the controller.

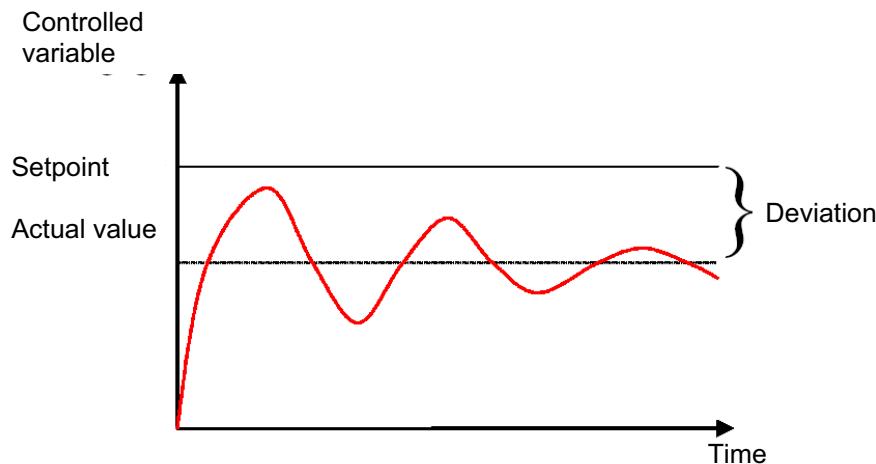
6



$$y = K_p \cdot e$$

* Figure and text from SAMSON Technical Information - L102 – Controllers and Controlled Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

The diagram below shows the behavior of the P-controller:



In comparison to other controller types, the advantages of this type of controller are its simplicity (in the simplest case, the electronic realization can consist of a mere resistor) on the one hand, and on the other hand its rather prompt response.

The main disadvantage of the P-controller is its permanent control deviation. Even over long periods of time, the setpoint is never reached completely. This disadvantage as well as a response speed that is as yet not ideal can be minimized only insufficiently with a larger proportional factor, since otherwise, the controller will overshoot; i.e. there will be an overreaction as it were.

In the most disadvantageous case, the controller will be in continuous oscillation; this moves the controlled variable away from the setpoint: not through the influencing variable, but through the controller itself.

The problem of permanent control deviation is best solved with the integral controller.

3.6.2 The Integral Controller (I-Controller)

Integrating controllers are used to completely compensate control deviations at every operating point. As long as the control deviation is not equal to zero, the amount of the manipulated variable changes. Only if the reference variable and the controlled variable are the same -at the latest when the manipulated variable has reached its system-related limit (Umax, Pmax etc.)- is the controller in a steady state.

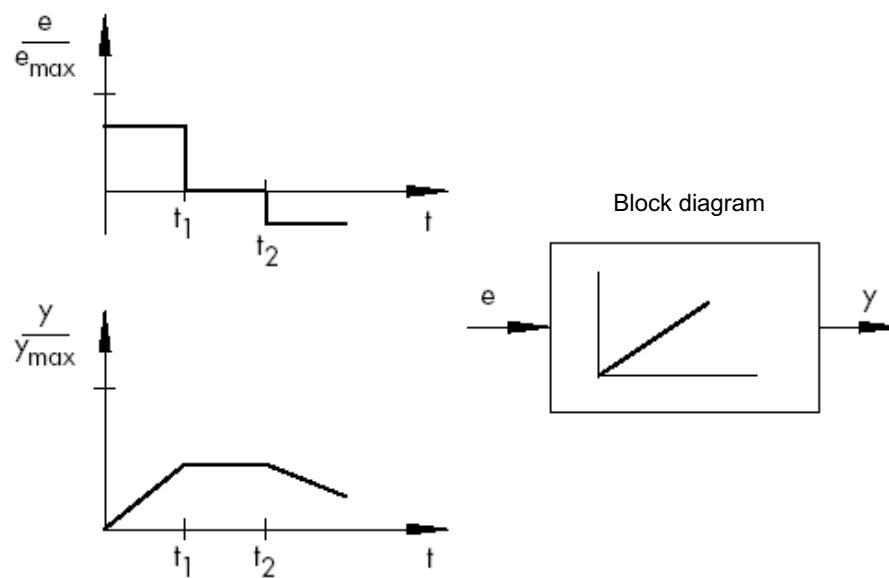
The mathematical formulation of this integral behavior is as follows:

The manipulated variable is proportional to the time integral of the control deviation e :

$$y = K_i \int e dt \quad \text{with: } K_i = \frac{1}{T_n}$$

How fast the manipulated variable rises (or drops), depends on the control deviation and the integration time.

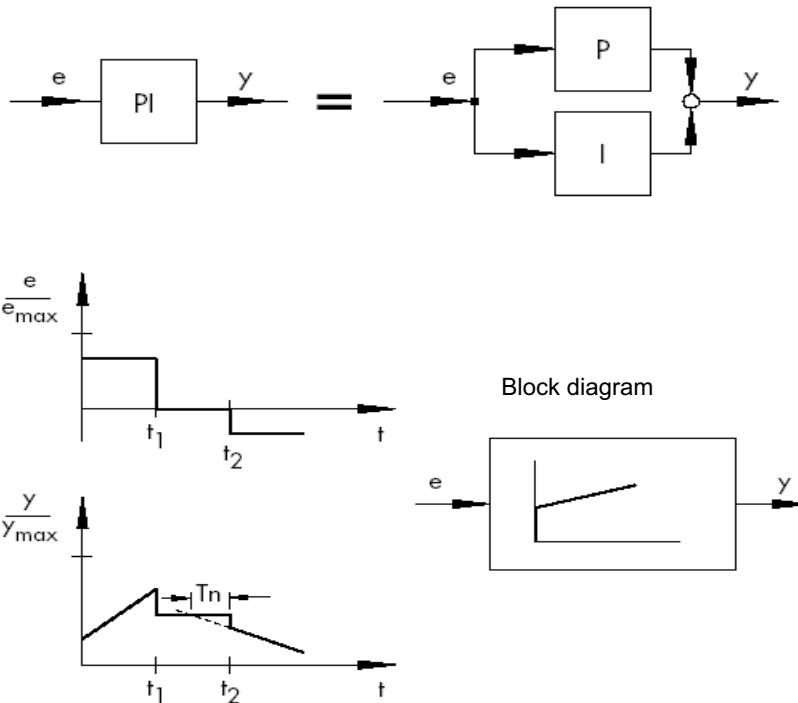
6



* Figure and text from SAMSON Technical Information - L102 – Controllers and Control Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

3.6.3 The PI Controller

The PI controller is a controller type much used in practice. It is the result of connecting a P-controller and an I-controller in parallel. If laid out correctly, it combines the advantages of both controller types (stable and fast, no permanent steady state deviation), so that their disadvantages are compensated at the same time.



The behavior with respect to time is marked with the proportional coefficient K_p and the integrator time T_n . Because of the proportional component, the manipulated variable immediately responds to any control deviation e , while the integral component will have an effect only in the course of time. T_n represents the time that elapses until the I-component generates the same amplitude of flow as is generated immediately because of the P-component (K_p). As for the I-controller, the integrator time T_n has to be reduced if the integral component is to be increased.

Controller design:

Depending on K_p and T_n dimensioning, overshooting the controlled variable can be decreased at the expense of control dynamics.

Application area for the PI controller: fast control loops that don't permit steady state deviation.

Examples: pressure, temperature and ratio controls.

* Figure and text from SAMSON Technical Information - L102 – Controller and Control Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

3.6.4 Differential Action Controller (D-Controller)

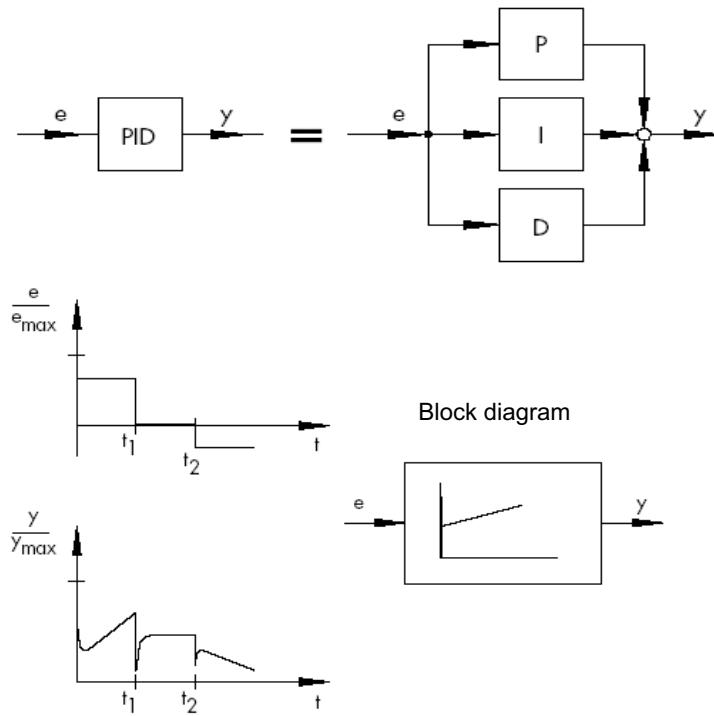
The D-controller generates its manipulated variable from the time derivative of the control deviation and not, like the P-controller, from its amplitude. For that reason, it responds considerably faster than the P-controller. Even if the control deviation is low, it generates -looking ahead, as it were- large amplitudes of flow as soon as an amplitude change occurs.

However, the D-controller does not recognize steady state deviations, because no matter how large it is, its time derivative is equal to zero. For that reason, the D-controller is rarely used by itself in practice. Rather, it is used jointly with other control elements, usually in connection with a proportional component.

3.6.5 The PID Controller

If we expand the PI controller with a D-component, the universal PID controller is boosted.

As in the case of the PD controller, adding the D-component has the effect that -if laid out correctly- the controlled variable reaches its setpoint sooner and enters a steady state faster.



$$y = K_p \cdot e + K_i \int e dt + K_D \frac{de}{dt} \quad \text{with} \quad K_i = \frac{K_p}{T_n}; \quad K_D = K_p \cdot T_V$$

* Figure and text from SAMSON Technical Information - L102 – Controllers and Control Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

3.7 Objectives for Setting the Controller

For the control result to be satisfactory, the selection of a suitable controller is important. But even more important is the setting of the suitable controller parameters K_p , T_n and T_v , which have to be in tune with the controlled system behavior. Usually we have to compromise between very stable but slow controlling or a very dynamic more uneven control behavior that under certain circumstances has a tendency to oscillate and can become unstable.

In the case of non-linear systems that are always to work in the same operating point -for example, fixed setpoint control- the controller parameters have to be adapted to the system behavior in this operating point. If, as in the case of cascade controls ñ, no fixed operating point can be defined, a controller setting has to be found that provides a sufficiently fast and stable control result over the entire work area.

In practice, controllers are usually set based on empirical values. If they are not available, the control system behavior has to be exactly analyzed in order to specify -with the aid of theoretical or practice based design procedures- suitable controller parameters later.

One possibility is the oscillation trial based on the Ziegler-Nichols method. It provides for a simple and in many cases suitable layout. However, this setting method can be used only for controlled systems that allow for a controlled variable to oscillate autonomously. The method used is then as follows

- Set K_p and T_v at the controller to the lowest value and T_n to the highest value (the lowest possible controller effect).
- Take the controlled system manually to the desired operating point (start controller)
- Set the manipulated variable manually to the specified value and switch to the automatic mode
- Increase K_p (decrease X_p) until harmonic oscillations of the controlled variable are recognized. If possible, the control loop should be induced to oscillate during the K_p setting by using small sudden setpoint changes.
- Store the K_p value that was set as critical proportional coefficient $K_{p,krit}$
- Determine the duration of an entire oscillation as T_{krit} , if necessary with a stop watch, by generating the arithmetical mean over several oscillations.
- Multiply the values of $K_{p,krit}$ and T_{krit} using the multipliers according to the table and then set the values ascertained for K_p , T_n and T_v at the controller.

* Text from SAMSON Technical Information - L102 – Controllers and Controlled Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

	K_p	T_n	T_v
P	$0,50 \cdot K_{P,krit.}$	-	-
PI	$0,45 \cdot K_{P,krit.}$	$0,85 \cdot T_{krit.}$	-
PID	$0,59 \cdot K_{P,krit.}$	$0,50 \cdot T_{krit.}$	$0,12 \cdot T_{krit.}$

6

* Figure and text from SAMSON Technical Information - L102 – Controllers and Control Systems, Edition: August 2000 (http://www.samson.de/pdf_en/l102en.pdf)

3.8 Setting Controlled Systems

To set the controlled systems, the example of a PT2 system is used.

T_u-T_g Approximation

The basis for the methods according to Ziegler-Nichols and Chien, Hrones and Reswick is the T_u-T_g approximation, where from the system step response, the parameters transfer coefficient of the controlled system K_S, delay time T_u and compensation time T_g are determined.

The setting rules described below were ascertained experimentally, using analog computer simulations.

P-T_N systems can be described with sufficient accuracy using the so-called T_u-T_g approximation; i.e., through approximation by means of a P-T₁-T_L system..

The starting point is the system step response with the input step height K. The required parameters transfer coefficient of system K_S, delay time T_u and recovery time T_g are determined as shown in the figure below.

Measuring the transition function up to the stationary end value (K*K_S) is necessary so that the transfer coefficient of the system K_S needed for the calculation can be determined.

The essential advantage of these methods is that the approximation can also be used if the controlled system cannot be described analytically.

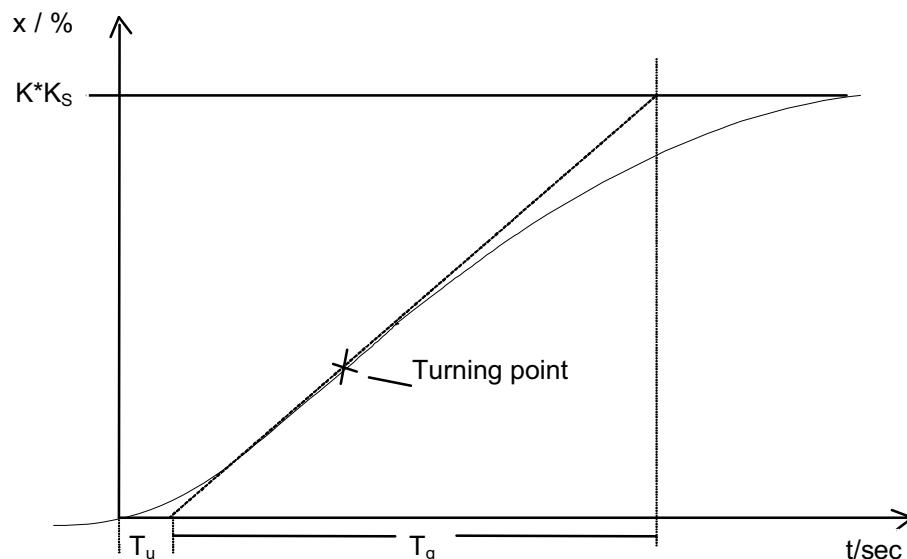


Figure: **T_u-T_g-Approximation**

3.8.1 Setting the PI Controller according to Ziegler-Nichols

By examining P-T₁-T_L systems, Ziegler and Nichols discovered the following optimum controller settings for fixed setpoint control:

$$K_{PR} = 0,9 \frac{T_g}{K_S T_u}$$

$$T_N = 3,33 T_u$$

These settings generally provide a disturbance characteristic that is quite good [7]

3.8.2 Setting the PI Controller according to Chien, Hrones and Reswick

For this method, command as well as the disturbance behavior were examined to obtain the most favorable controller parameters. The result is different values for both cases. In addition, two different settings are specified respectively that meet different requirements regarding the control quality.

The following settings resulted:

- For disturbance behavior:

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 command behavior:

Aperiodic transient reaction
with the shortest duration

20% overshoot
minimum oscillation period

$$K_{PR} = 0.35 \frac{T_g}{K_S T_u}$$

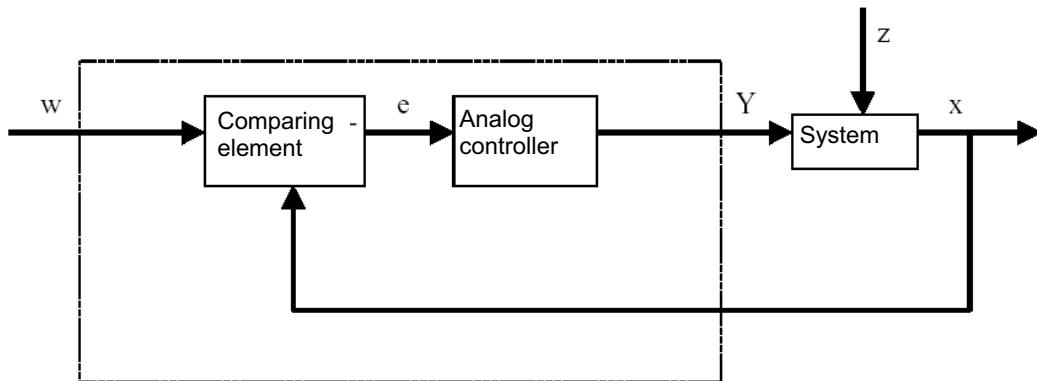
$$K_{PR} = 0.6 \frac{T_g}{K_S T_u}$$

$$T_N = 1,2 T_g$$

$$T_N = T_g$$

3.9 Digital Controllers

So far, primarily analog controllers were discussed; i.e., such controllers that -from the system deviation that is present as analog value- derive the controller output variable analogously also. The schematic of such a loop is known:

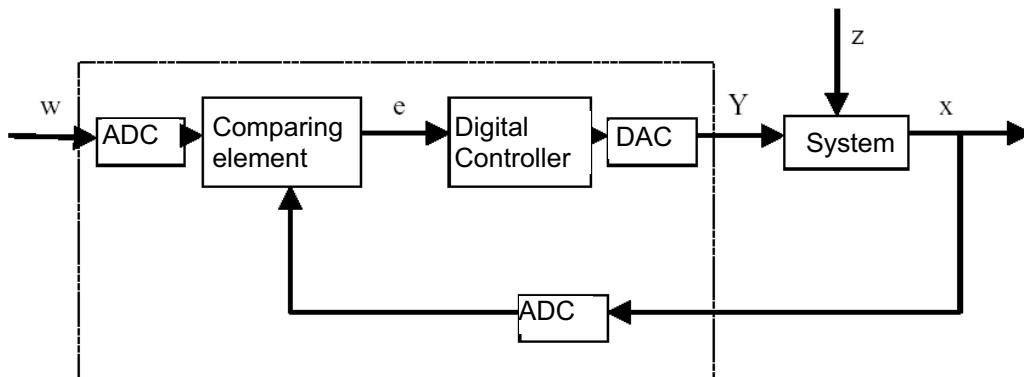


6 However, often it is advantageous to evaluate the system deviation digitally. On the one hand, the relationship between the system deviation and the controller output variable can be specified much more simply if it is defined with an algorithm or a formula that can be used to program a computer rather than implementing it in the form of an analog circuit. On the other hand, a clearly higher integration of circuits is possible in digital technology, so that several controllers can be accommodated in a small space. And finally, by distributing the computing time it is even possible -if the computing capacity is large enough- to use one single controller as controlling elements of several loops.

To make digital processing of the variables possible, first the reference variable as well as the feedback variable are converted in an analog-digital converter (ADC) into digital variables.

A digital comparison element subtracts them from each other and the difference is transferred to the digital controlling element. Its controller output variable is then converted again to an analog variable in a digital-analog converter (DAC). That is, the unit consisting of converters, comparing element and controlling element appear to the outside to be an analog converter.

We are using a diagram to look at the structure of a digital controller:



In addition to the advantages that the digital conversion of a controller has, it also has as a consequence various problems. For that reason, some variables have to be selected large enough for the digital controller so that control accuracy does not suffer too much under digitalization. Quality criteria for digital computers are:

- Quantization resolution of the digital-analog converters

It indicates how fine the continuous value range is digitally rasterized. The resolution has to be selected such that no resolutions important to the control system are lost.

- The sampling rate of the analog-digital converters

That is the frequency with which the analog values pending at the converter are measured and digitalized. It has to be high enough that the controller can also react to sudden changes in the controlled variable in time.

- Cycle time

In distinction to analog computers, every digital computer processes in clock cycles. The speed of the computer used has to be so fast that during a clock cycle (where the output value is calculated and no input value is polled) the controlled variable cannot change significantly.

The quality of the digital controller has to be high enough so that to the outside, it comparably reacts as promptly and precisely as an analog controller would.

4. Sample Task - Level Control for a Tank

We are programming a level control for our program.

A sensor measures the level in a tank and converts it into a voltage signal 0 to 10V: 0V corresponds to a level of 0 liters and 10V to a level of 1000 liters.

The sensor is connected to the first analog input of the SIMATIC S7-1200.

Now, this level is to be controlled alternatively for 0 liters ($S1 == 0$) or 700 liters ($S1 == 1$).

To this end, a "**PID_Compact**" controller integrated in STEP 7 Basic V10.5 is used. This PID controller in turn controls a pump as manipulated variable by means of 0 to 10V.

Assignment list:

Address	Symbol	Data Type	Comment
%IW 64X_Level_Tank1		Int	analog input actual level filling level tank1
%QW 80	Y_Level_Tank1	Int	analog output manipulated value pump1
%I 0.0	S1	Bool	sudden setpoint change filling level 0 (0) or 700 liters (1)

5. Programming the Level Control for the SIMATIC S7-1200

For project management and programming, the software '**Totally Integrated Automation Portal**' is used.

Here, under a uniform interface, components such as the controller, visualization and networking of the automation solution are set up, parameterized and programmed.

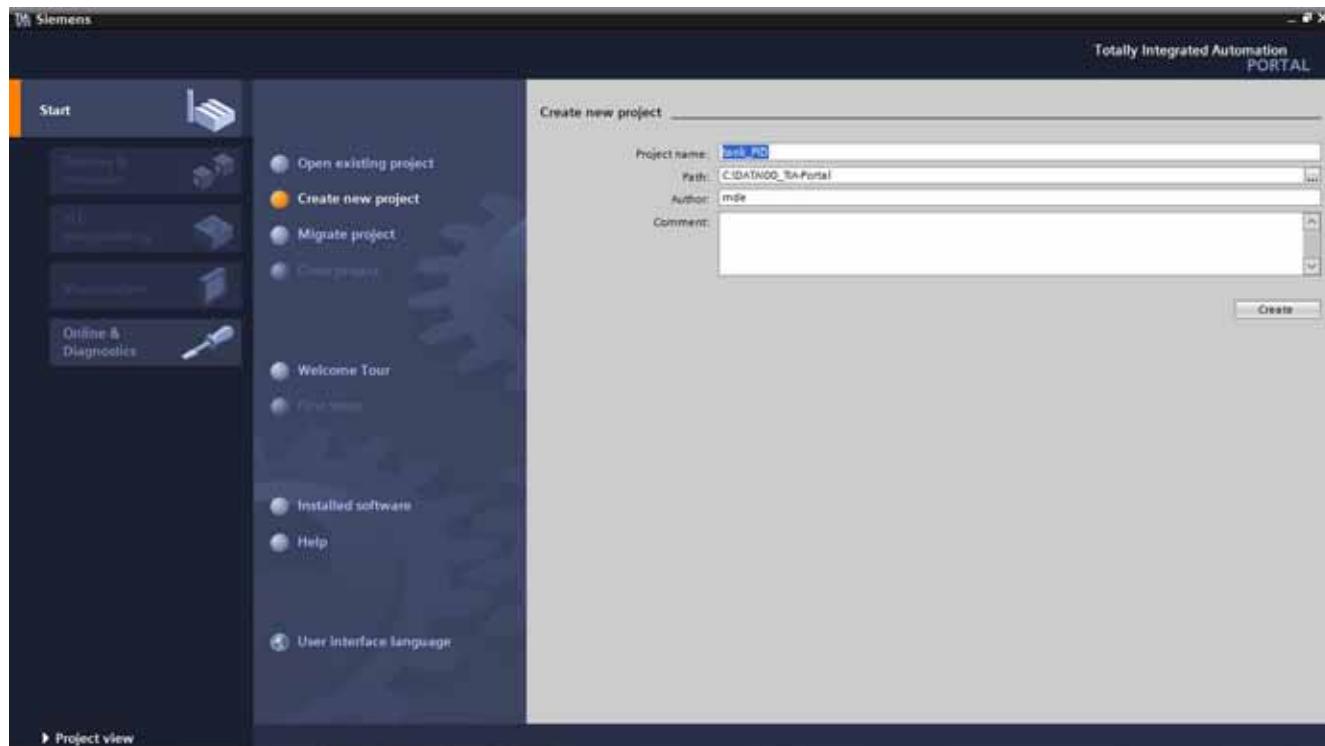
Online tools are available for error diagnosis.

In the steps below, a project can be set up for the SIMATIC S7-1200 and the solution for the task can be programmed.

1. The central tool is the '**Totally Integrated Automation Portal**'. It is called here with a double click.
(→ Totally Integrated Automation Portal V11)

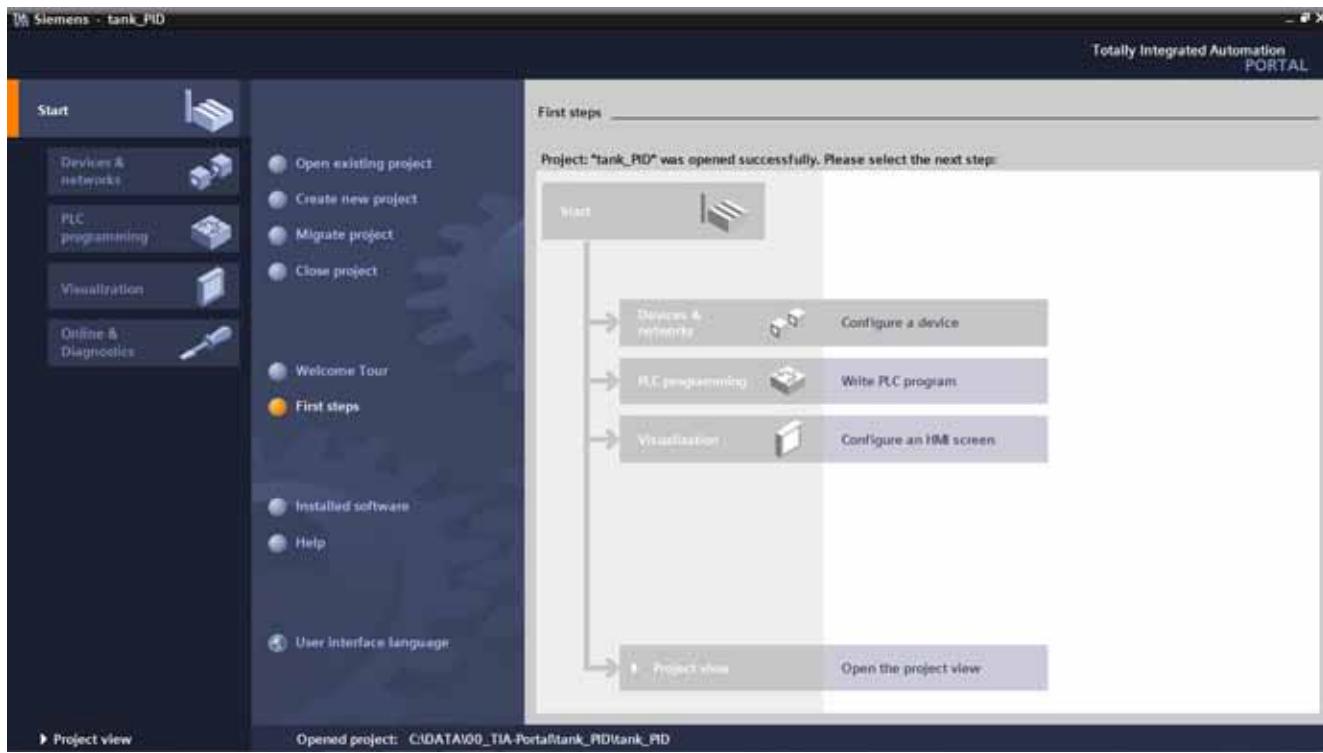


2. Programs for the SIMATIC S7-1200 are managed in projects. We are now setting up such a project. (→ Create new project → Tank_PID → Create)

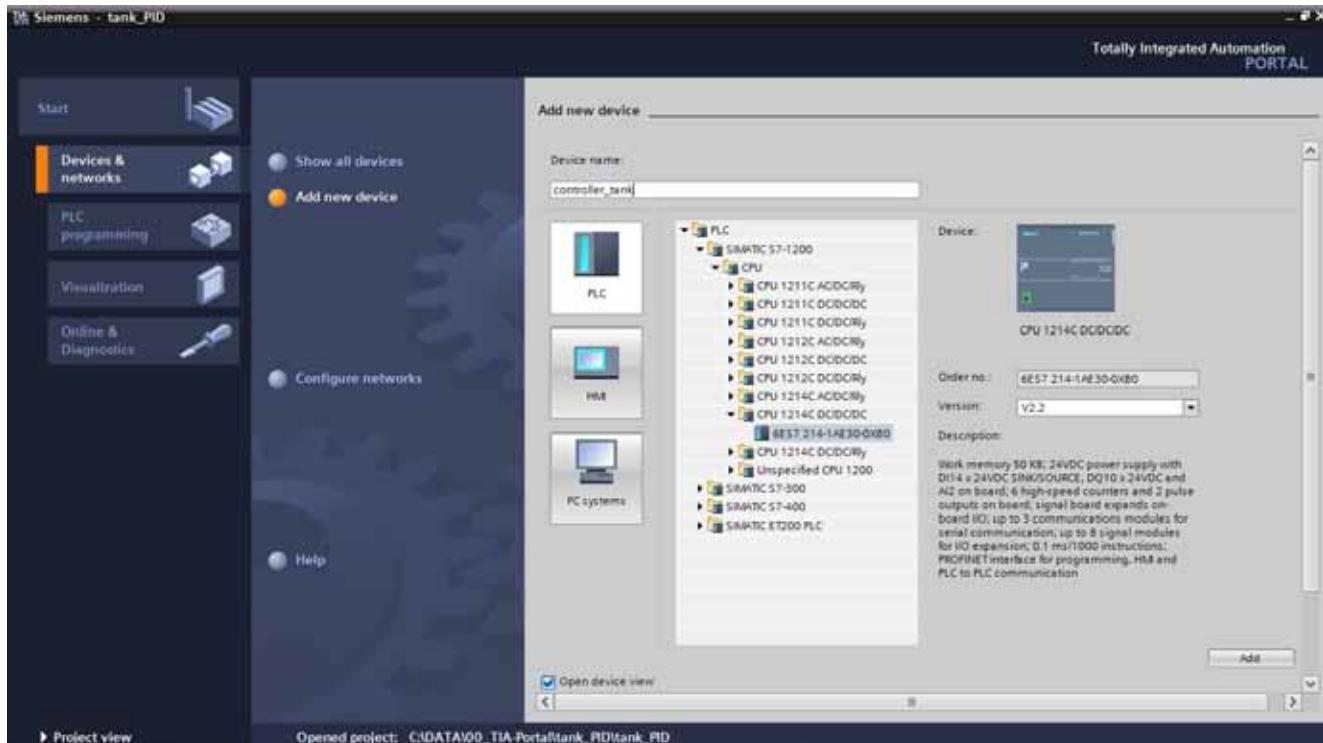


6

3. Next, 'First steps' are suggested for configuration. First, we want to '**Configure a device**'. (→ First steps → Configure a device)

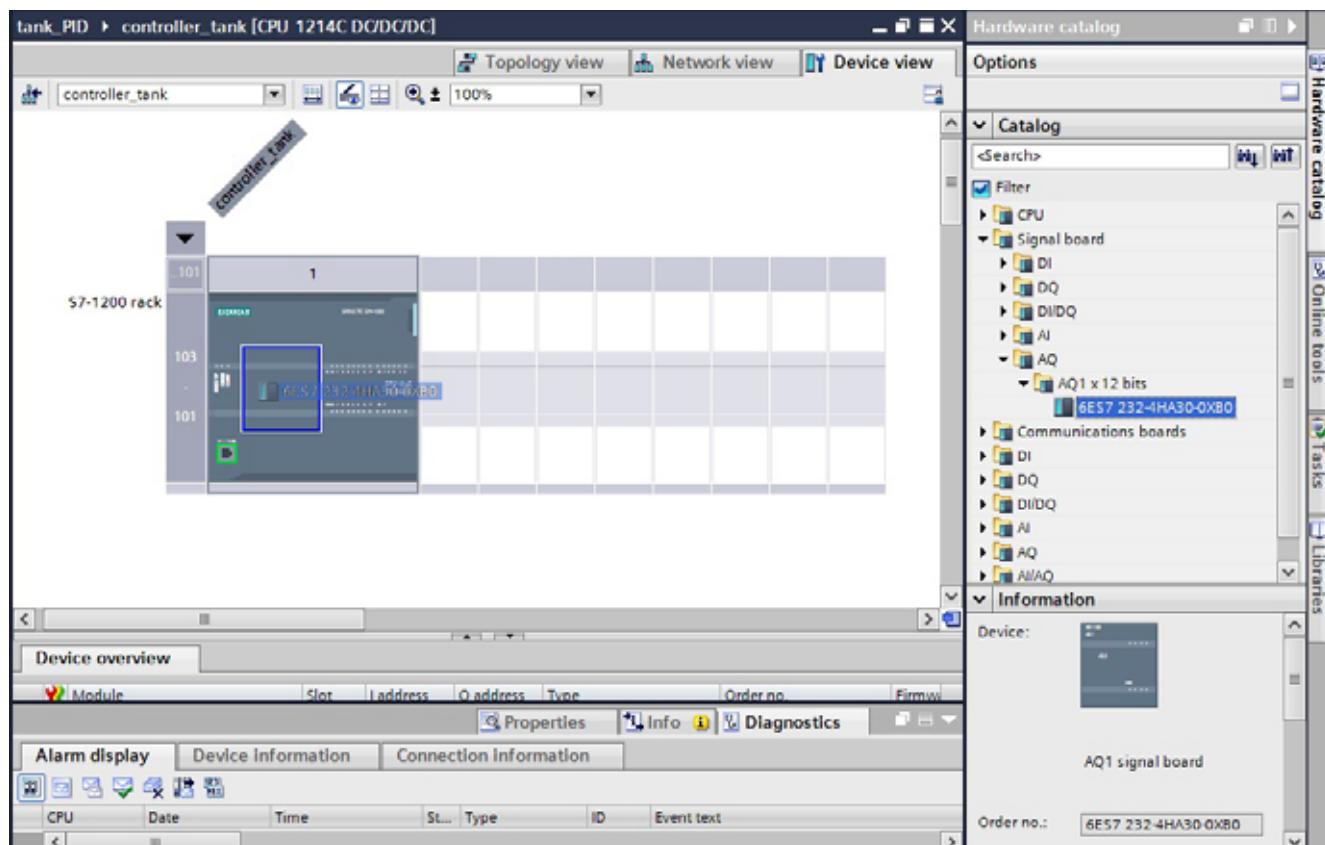


4. Then, we '**Add new device**' with the '**Device name controller_tank**'. To this end, we select '**CPU1214C**' from the catalog, with the matching order number.
 (→ Add new device → controller_tank → CPU1214C → 6ES7 → Add)

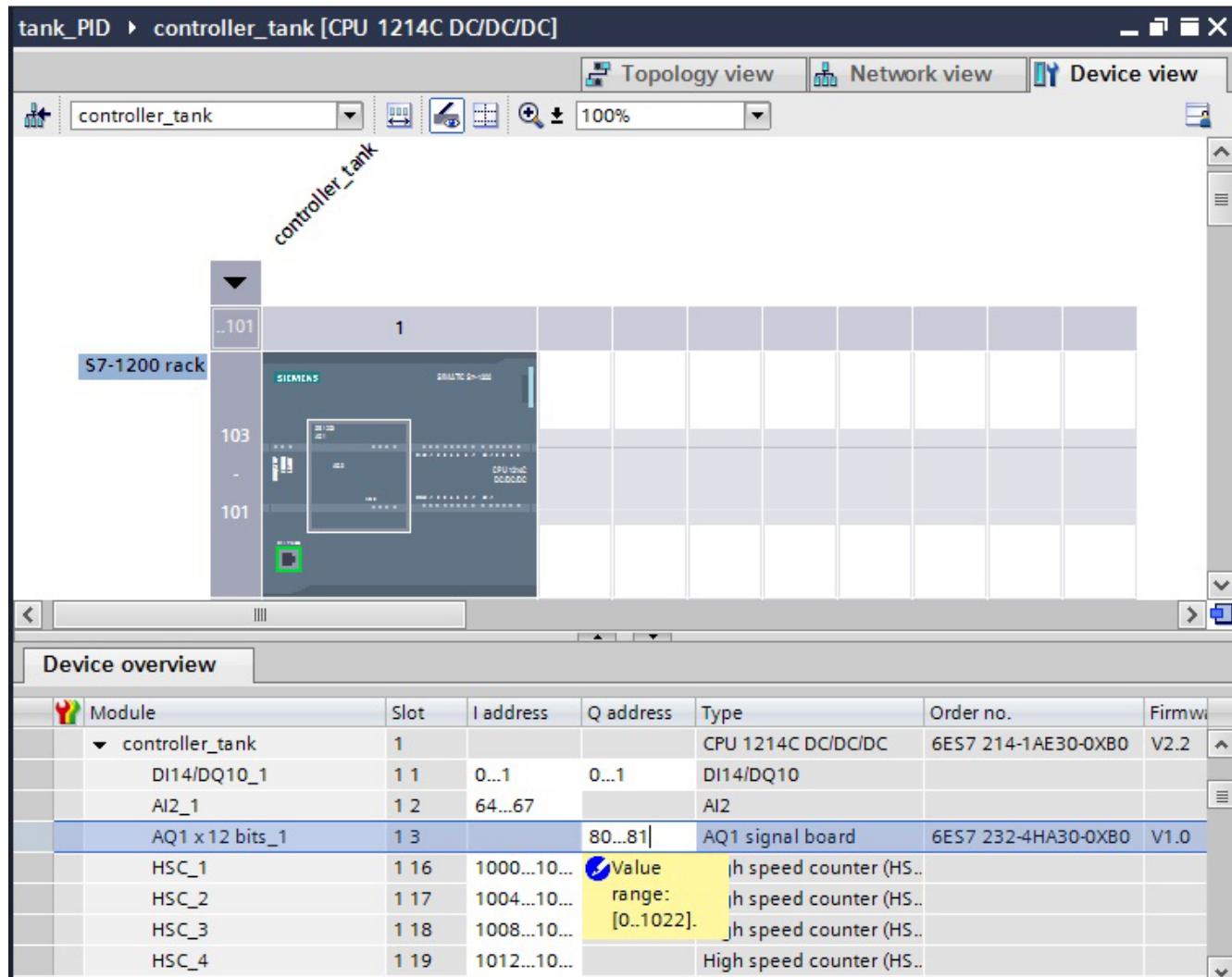


6

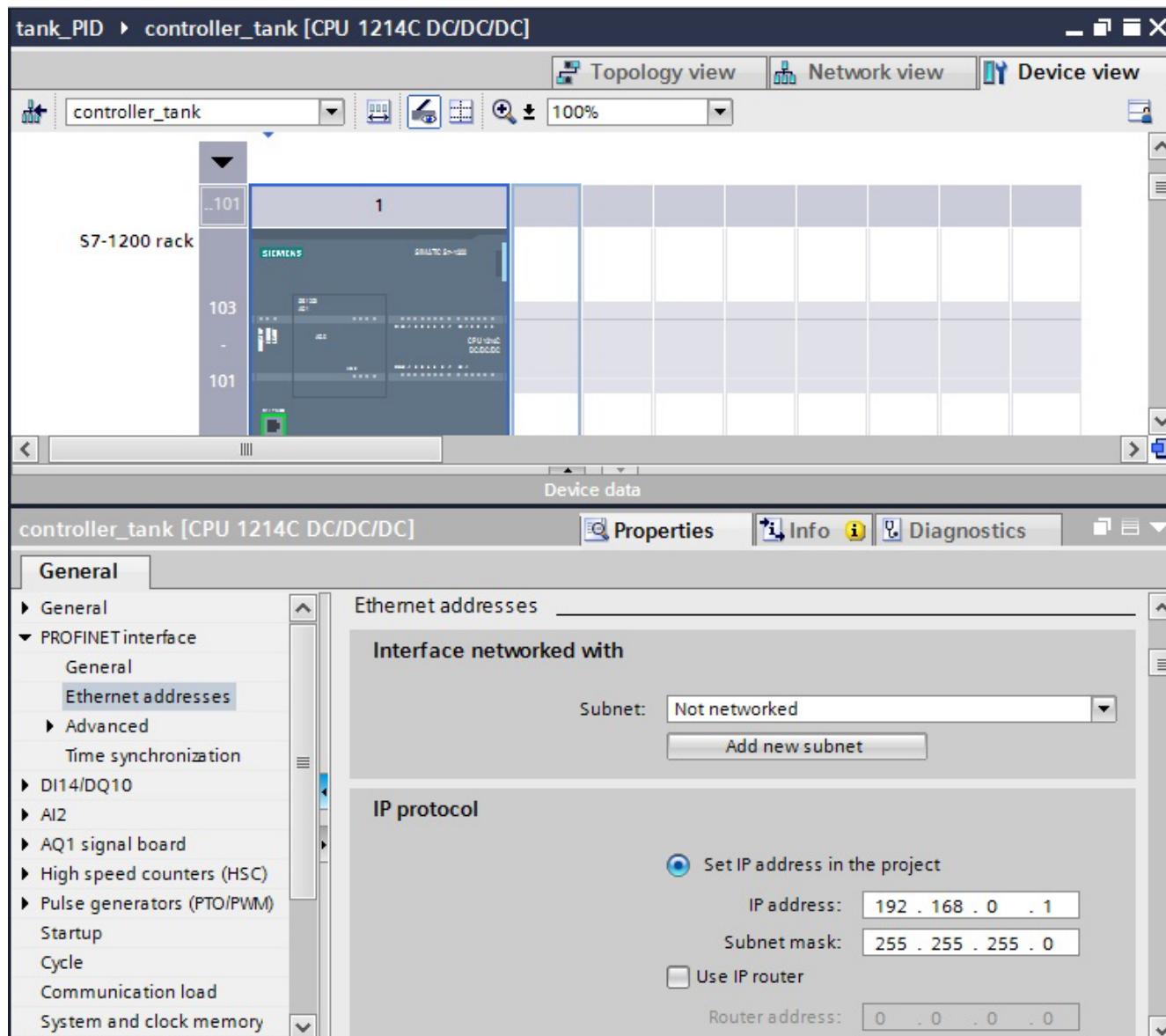
5. The software now changes automatically to the project view with the open hardware configuration. Additional modules can be added from the hardware catalog (to the right!). Here, the signal board for an analog output is to be inserted using drag&drop. (→ Catalog → Signal board → AO1 x 12Bit → 6ES7 232-...)



6. In 'Device overview', input and output addresses can be set. Here, the CPU's integrated analog inputs have the addresses %IW64 to %IW66, and the integrated digital inputs the addresses %I0.0 to %I1.3. The address of the analog output on the signal board is OW80. (→ Device overview → AQ1 x 12Bit → 80...81)



7. For the software to later access the correct CPU, its IP address and subnet mask have to be set. (→ Properties → General → PROFINET interface → Ethernet addresses →IP address: 192.168.0.1 → subnet mask: 255.255.255.0)



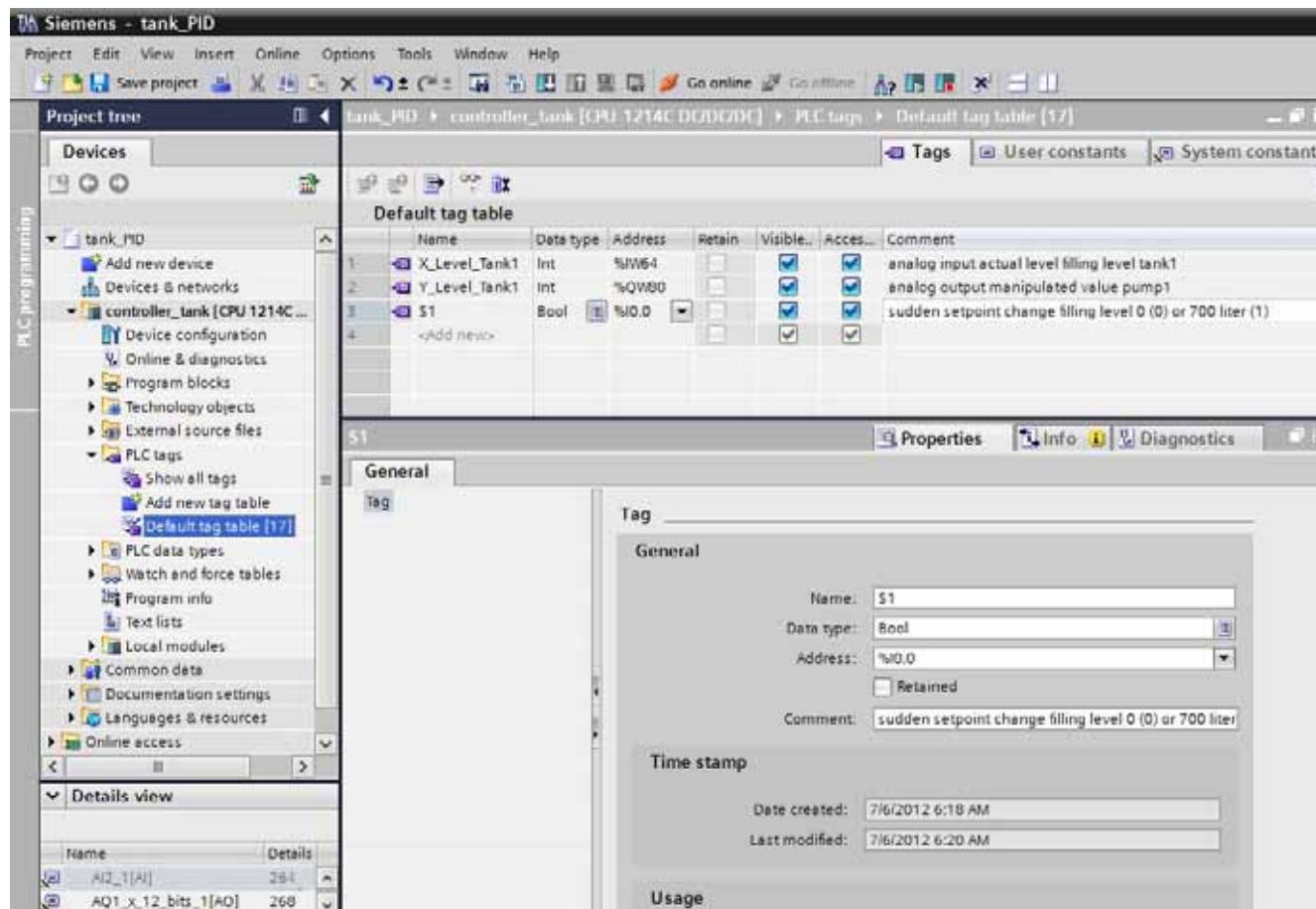
6

8. Since modern programming uses tags instead of absolute addresses, the **global PLC tags** have to be specified here.

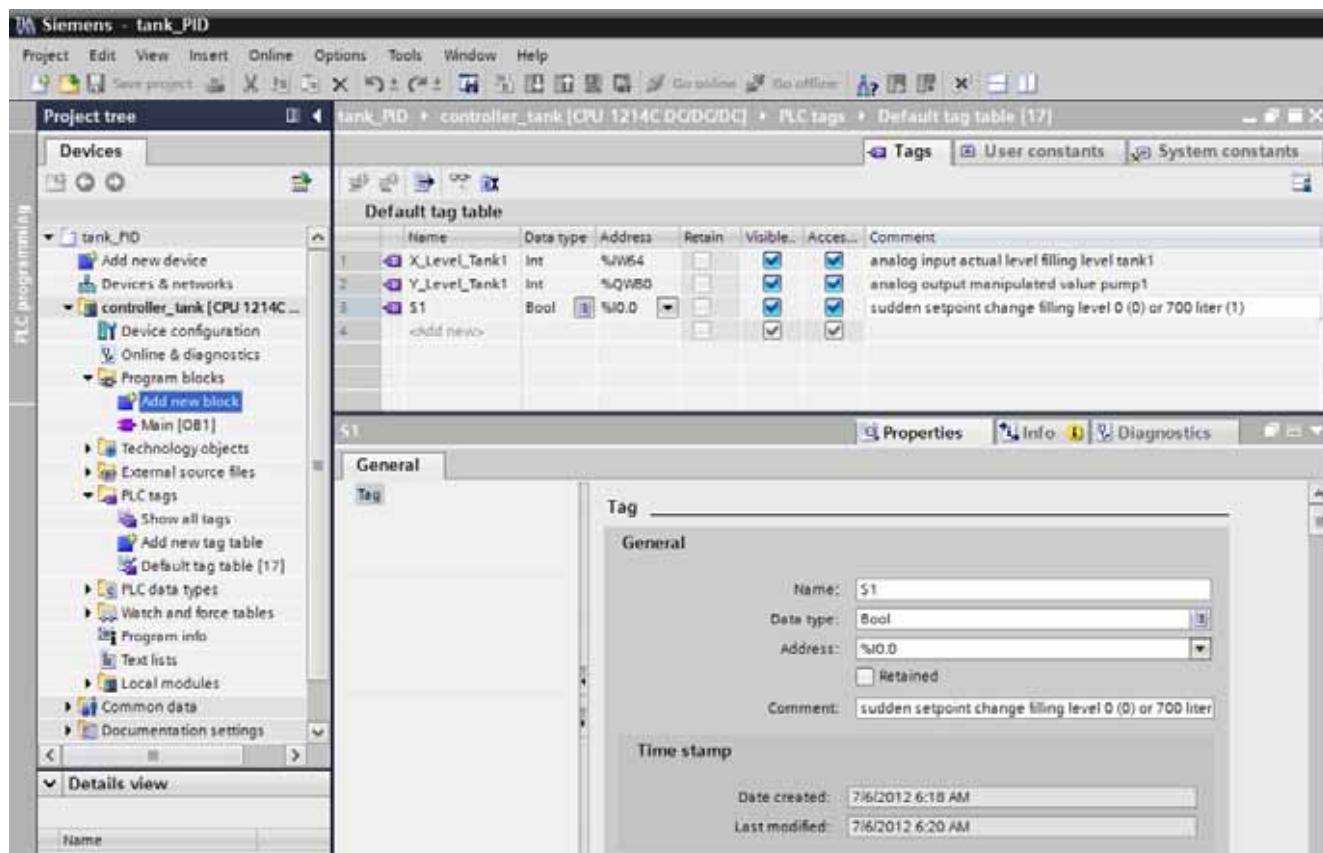
These global PLC tags are descriptive names with a comment for those inputs and outputs that are used in the program. Later, global PLC tags can be accessed by means of this name. These global tags can be used in the entire program in all blocks.

To this end, in program navigation select '**controller_tank[CPU1214C DC/DC/DC]**' and then '**PLC tags**'. With a double click, open the table '**PLC tags**' and enter, as shown below, the names for the inputs and outputs.

(→ controller_tank[CPU1214C DC/DC/DC] → PLC tags → Default tag table)

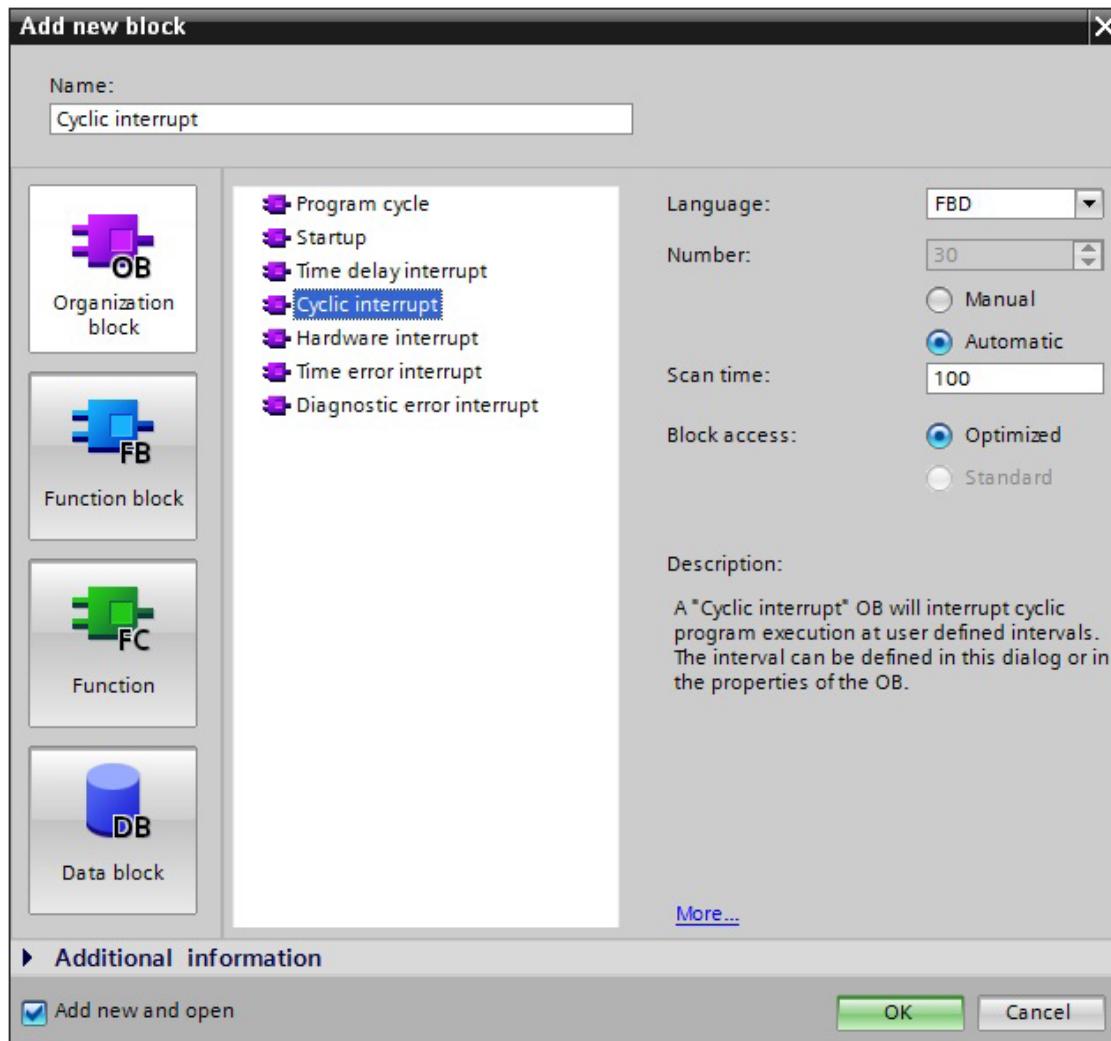


9. To create the function block FC1, in project navigation select '**controller_tank[CPU1214C DC/DC/DC]**' and then '**Program blocks**'. Next, double click on '**Add new block**'. (→ controller_tank[CPU1214C DC/DC/DC] → Program blocks → Add new block)



6

10. Select '**Organization block(OB)**', and as type '**Cyclic interrupt**'. As programming language, specify '**FBD**'. Numbers are assigned automatically (OB200). We leave the permanent scan time at 100ms. Accept the inputs with '**OK**'. (→ Organization block (OB) → Cyclic interrupt → FBD → Scan time 100 → OK)



Note:

The PID controller has to be called with a permanent scan time (here 100ms) since its processing is critical with respect to time. The controller could not be optimized if it would not be called accordingly.

11. The organization block '**Cyclic interrupt**'[OB200] will be opened automatically. Before the program can be written, its local tags have to be specified.

In the case of this block, only one tag type is used:

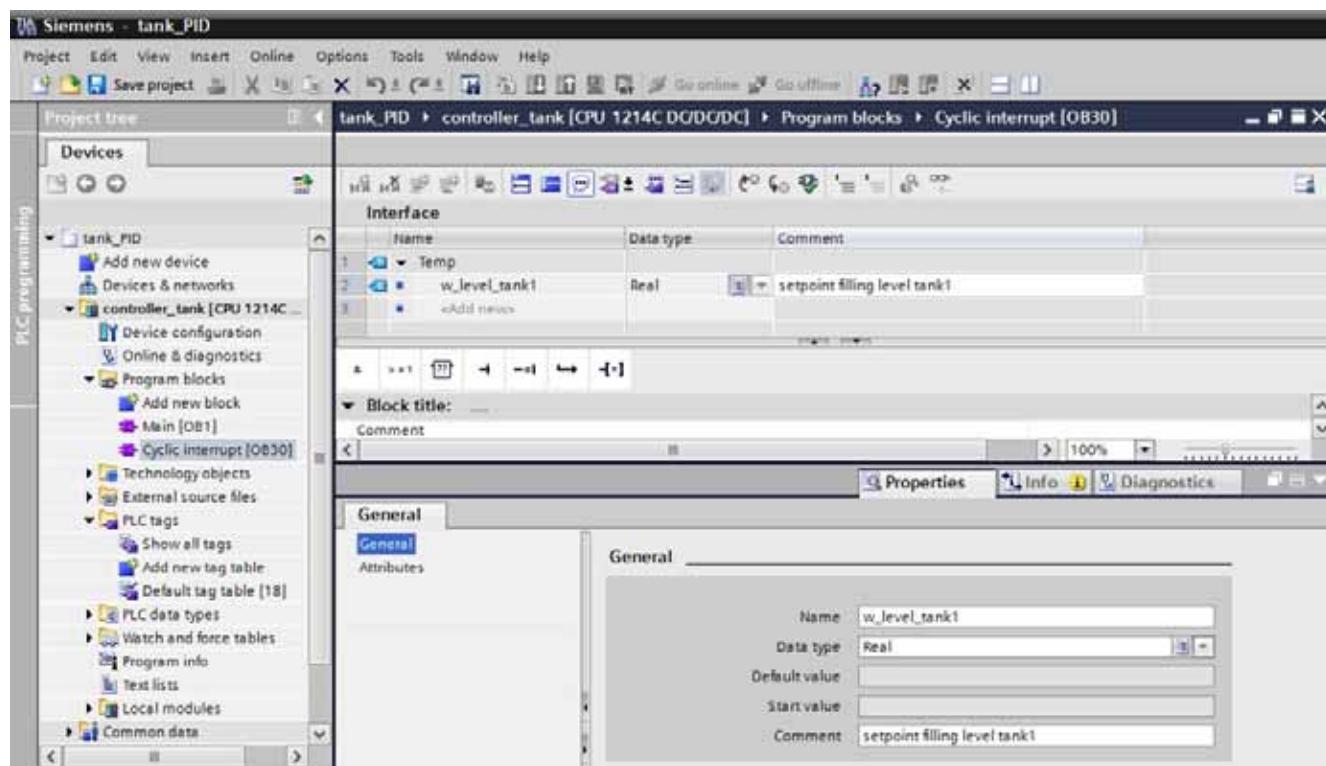
Type	Name	Function	Available in
Temporary		Tags that are used to store temporary intermediate results.	Functions, function blocks and
Local data	Temp	Temporary data is retained only for one cycle.	organization blocks

12. In our example, only the following local tag is needed.

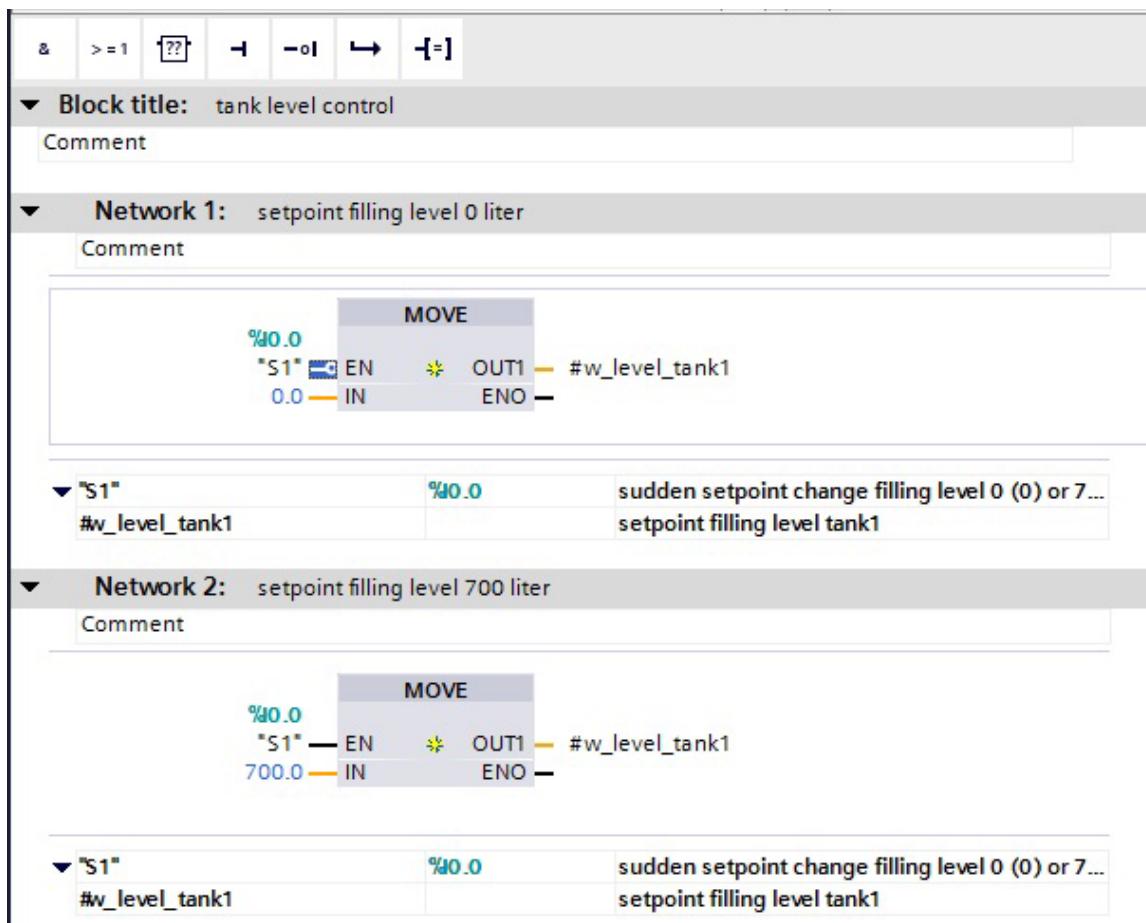
Temp:

w_level_tank1 Real This tag stores the setpoint for Tank1 as intermediate value

- Again, important in this example is the use of the correct data type Real; otherwise, it is not compatible in the following program with the PID controller block used.
For the sake of clarity, all local variables should also be provided with sufficient commentary.

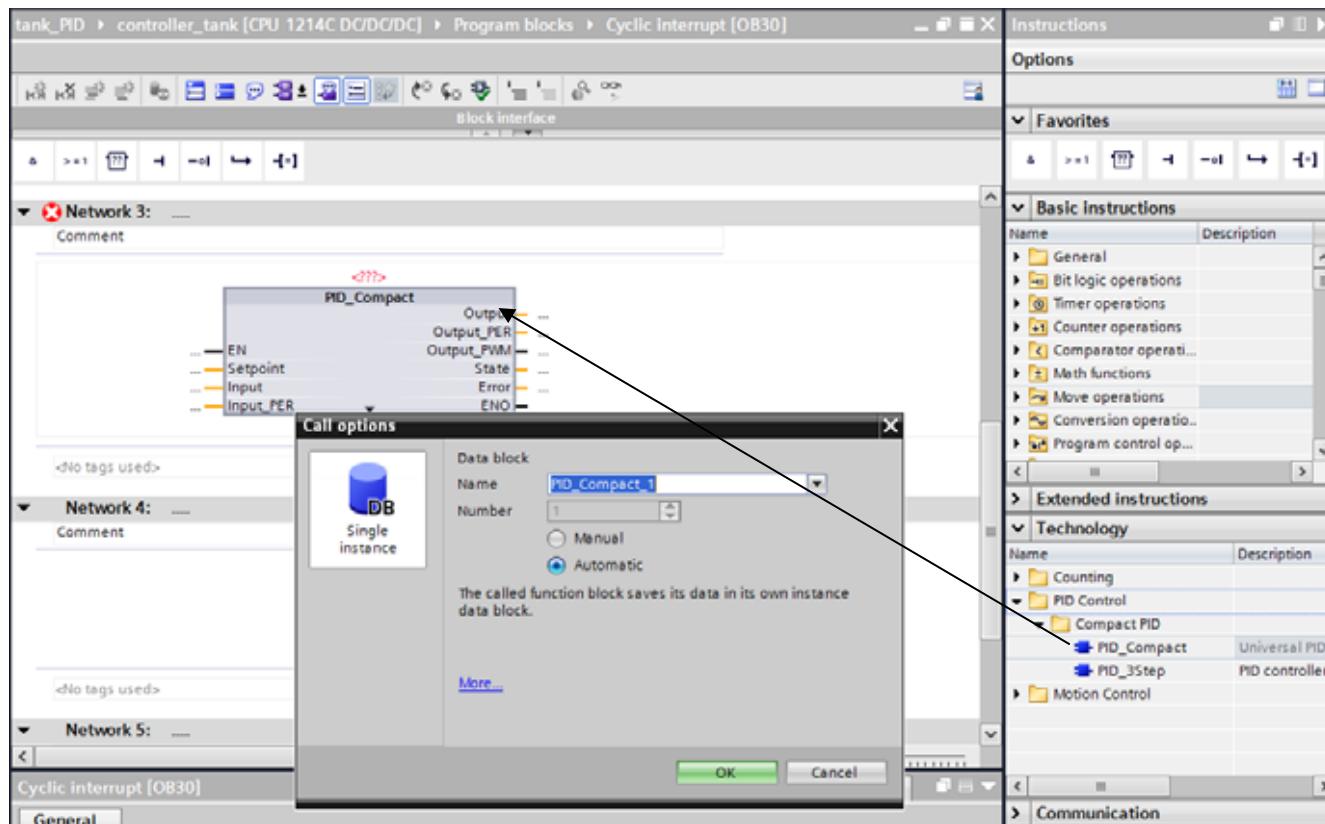


13. After the local tags are declared, the program can be entered by using the tag names. (Tags are marked with the symbol '#'.) Here, in the first two networks - with a 'MOVE' instruction respectively- either the floating point number 0.0 (S1 == 0) or 700.0 (S1 == 1) is copied to the local tag #w_fuell_tank1. (→ Instructions → Move → MOVE)

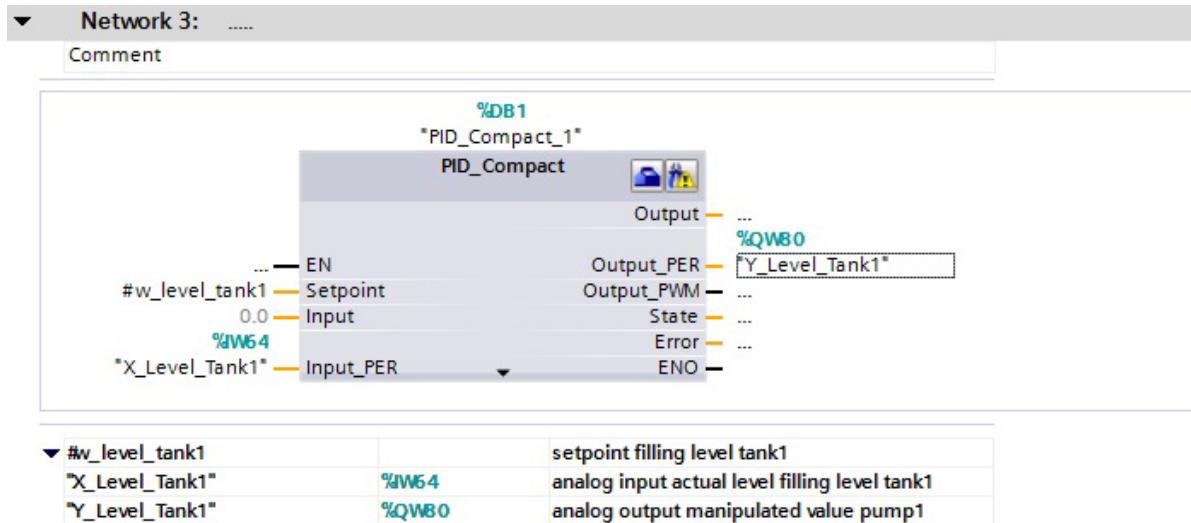


14. The controller block '**PID_Compact**' is moved to the third network. Since this block does not have multi-instance capability, it has to be assigned a data block as single instance. It is generated automatically by STEP7.

(→ Advanced instructions → PID → PID_Compact → OK)

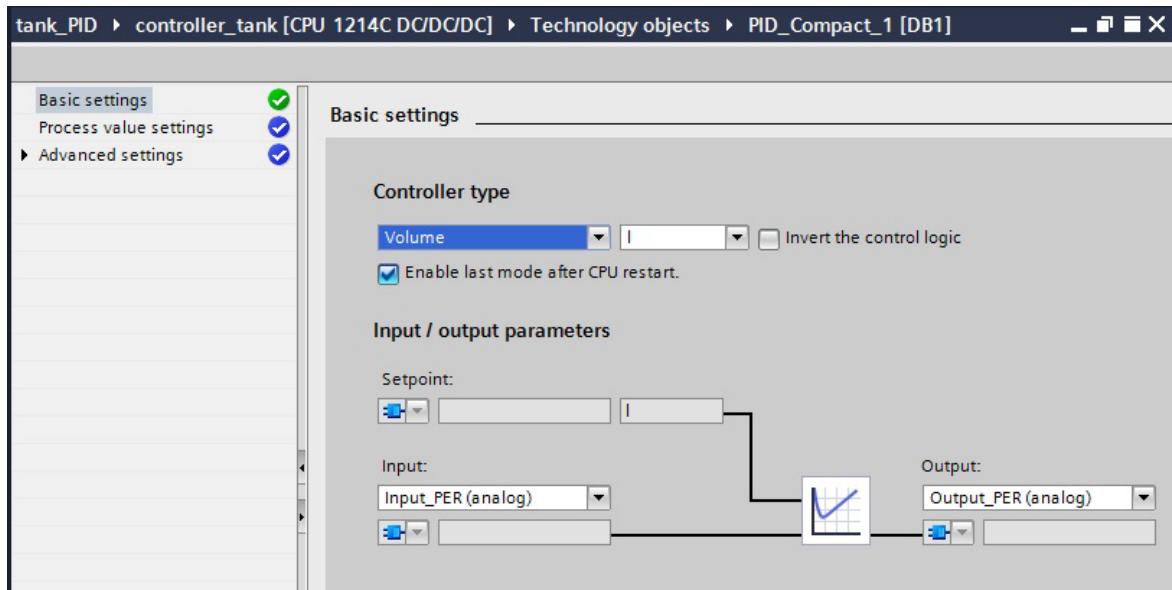


15. Wire this block, as shown here, with the setpoint (local tag #w_fuell_tank1), actual value (global tag "X_Level_Tank1") and manipulated variable (global tag "Y_Level_Tank1"). Then the configuration screen '  ' of the controller block can be opened. (→ #w_level_tank1 → "X_Level_Tank1" → "Y_Level_Tank1" → 



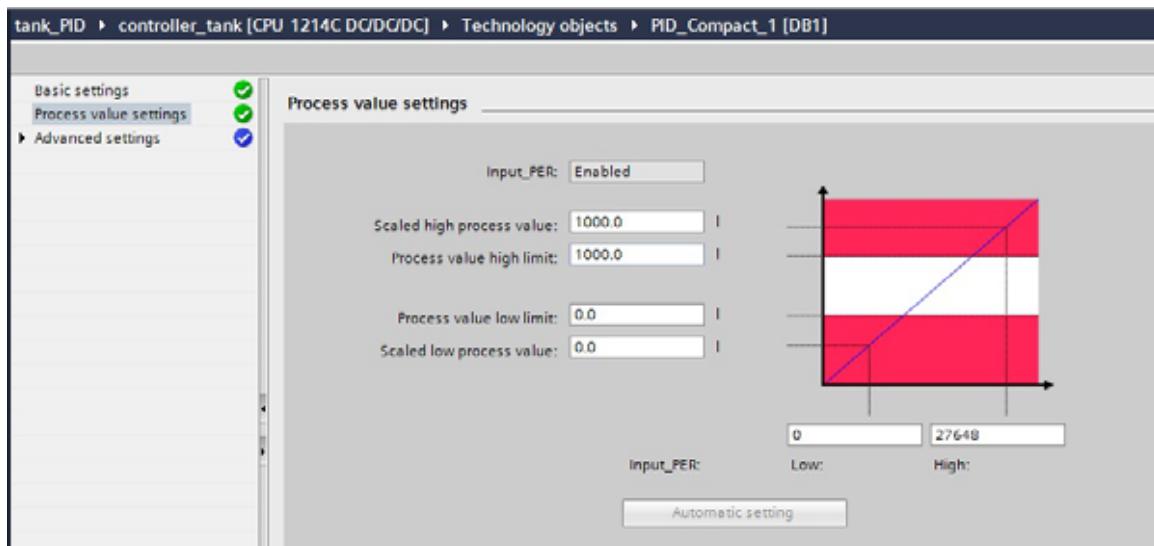
6

16. Here, we have to make the '**Basic settings**' such as controller type, and wire the internal controller structure. (→ Basic settings → Controller type Volume → I → actual value: Input_PER(analog) → manipulated value: Output_PER)



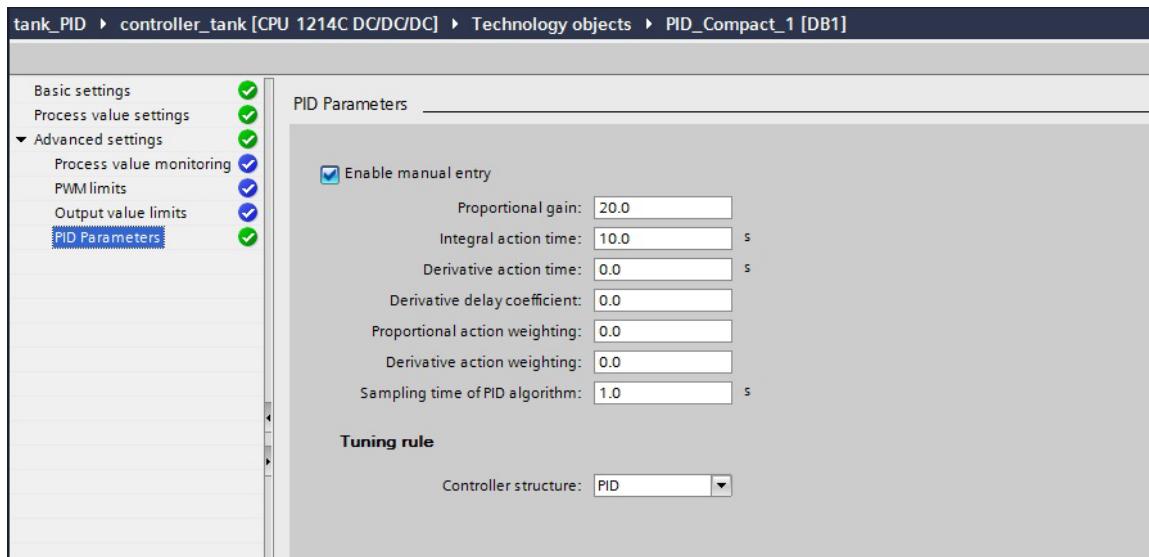
6

17. At 'Process value settings' we set the measuring range from 0 liters to 1000 liters. The limits also have to be adjusted. (→ Process value settings → Scaled high 1000.0 l → high limit 1000.0 l → Low limit 0.0 l → scale low 0.0 l)



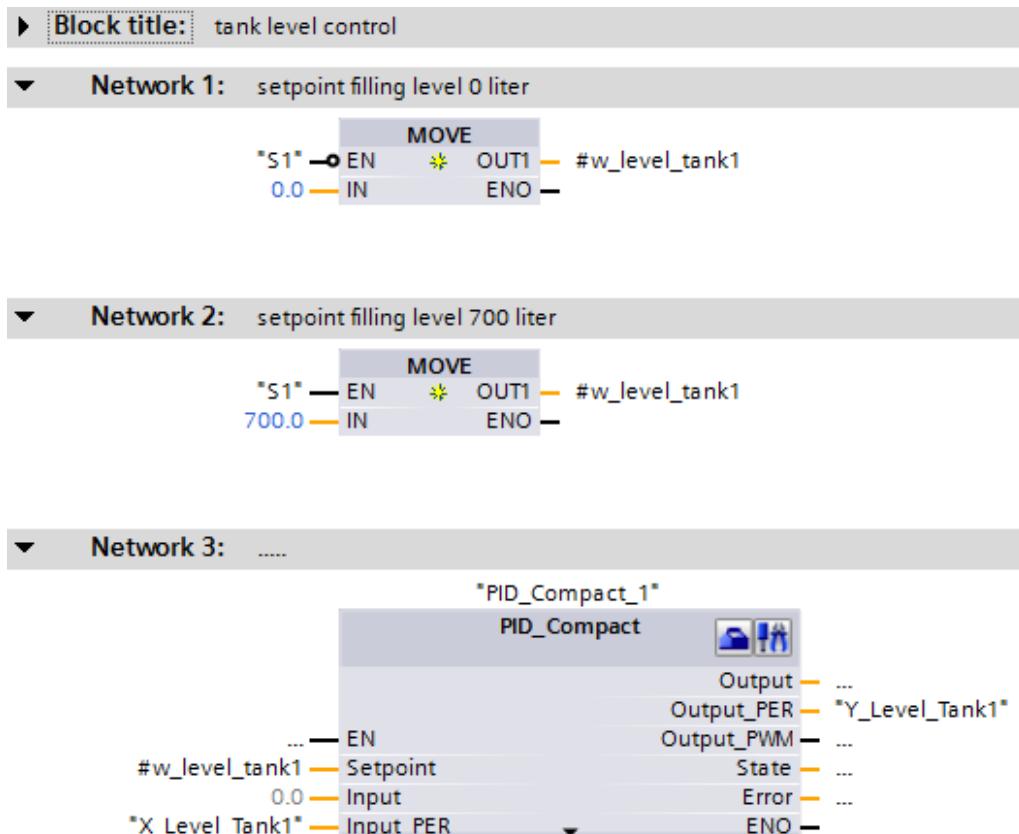
6

18. At '**Advanced settings**', there are also '**Process value monitoring**' and a manual setting for the '**PID Parameters**'. By clicking on **Save project** the project is saved. (→ Advanced settings → Process value monitoring → PID Parameters → **Save project**)



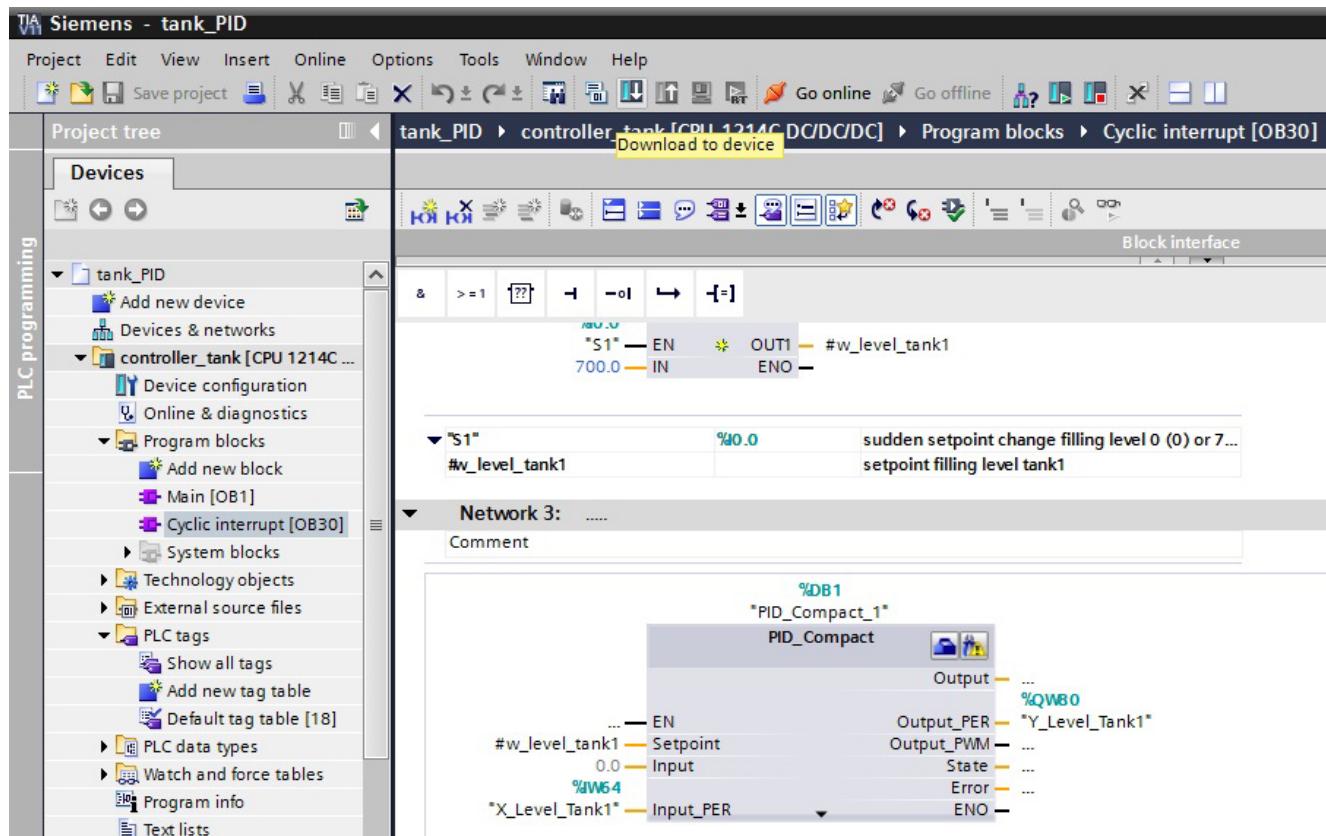
6

Program in function block diagram (FBD):

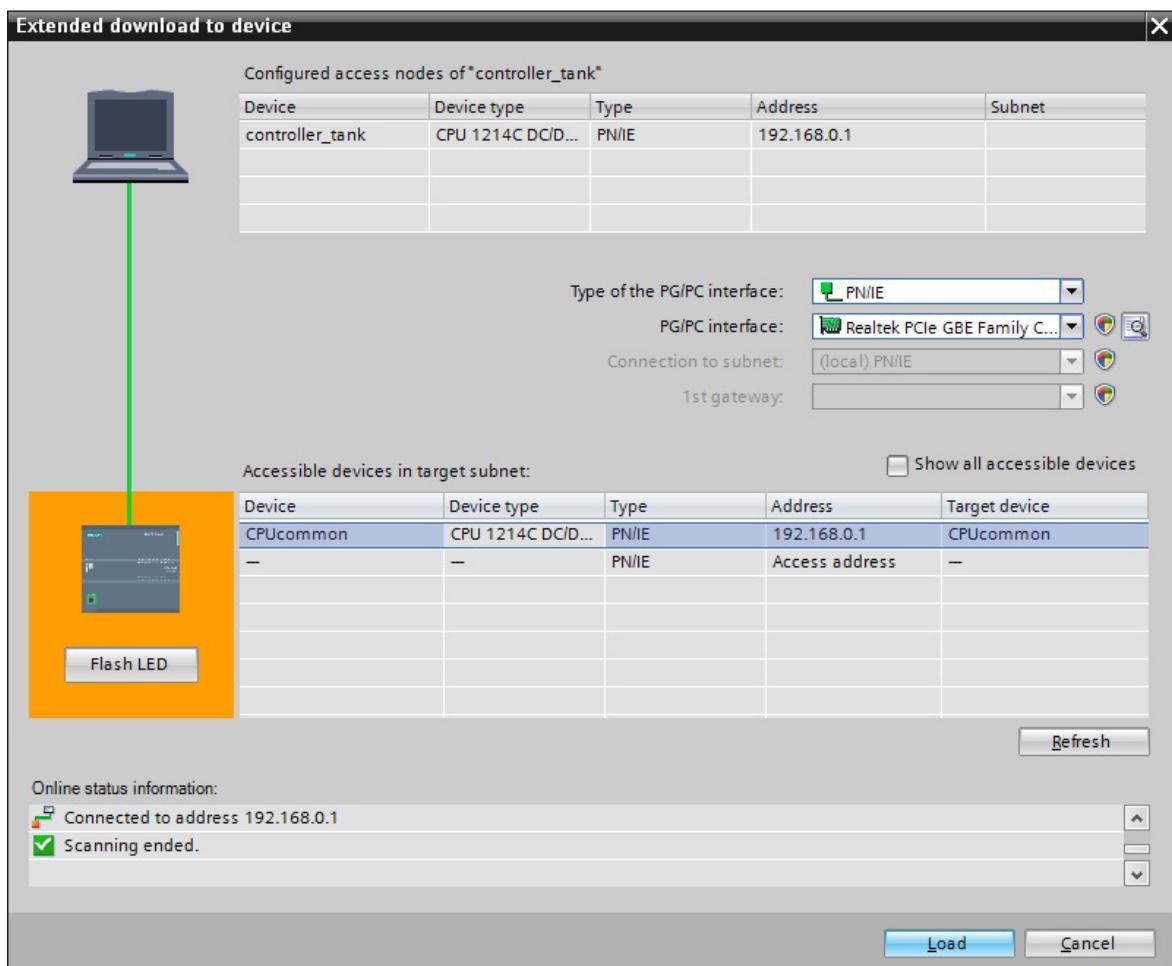


Program in ladder diagram (LAD):

19. To load the entire program to the CPU, first select the folder '**controller_tank**' and then click on the symbol  Download to device.
 (→**controller_tank** → 

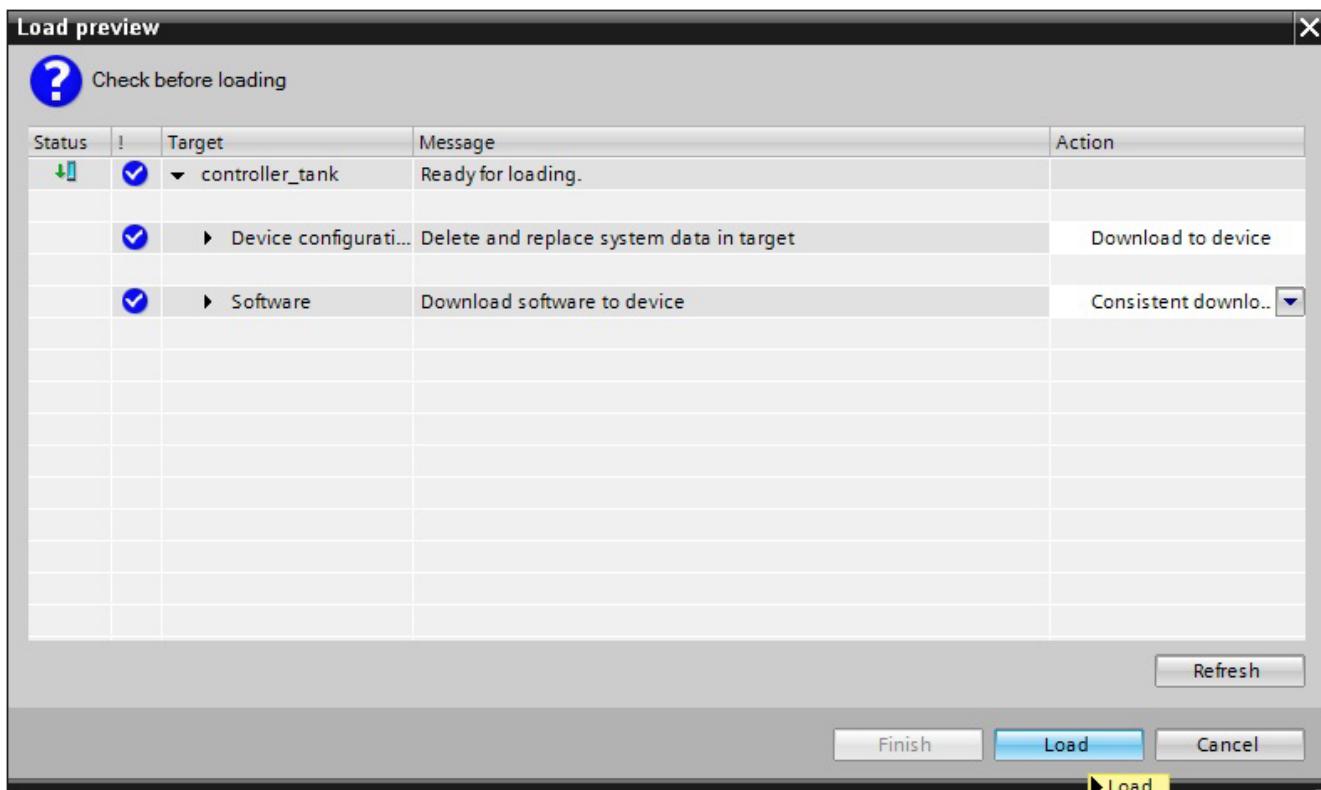


20. If the PG/PC interface was not specified previously (refer to Module M1, Chapter 4), a window is displayed where this can still be done. (→ PG/PC- interface for download → Load)



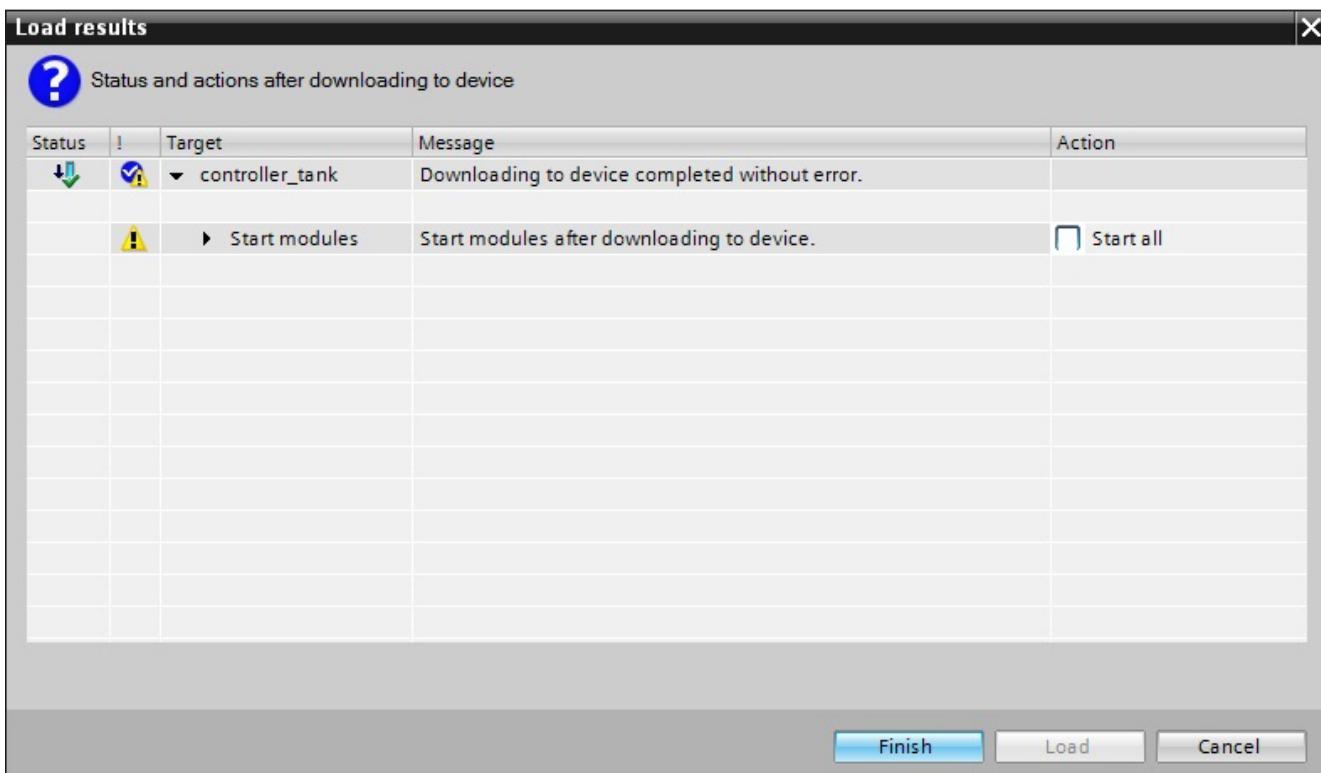
6

21. Now, click on 'Load' once more. During downloading, the status is displayed in a window. (→ Load)

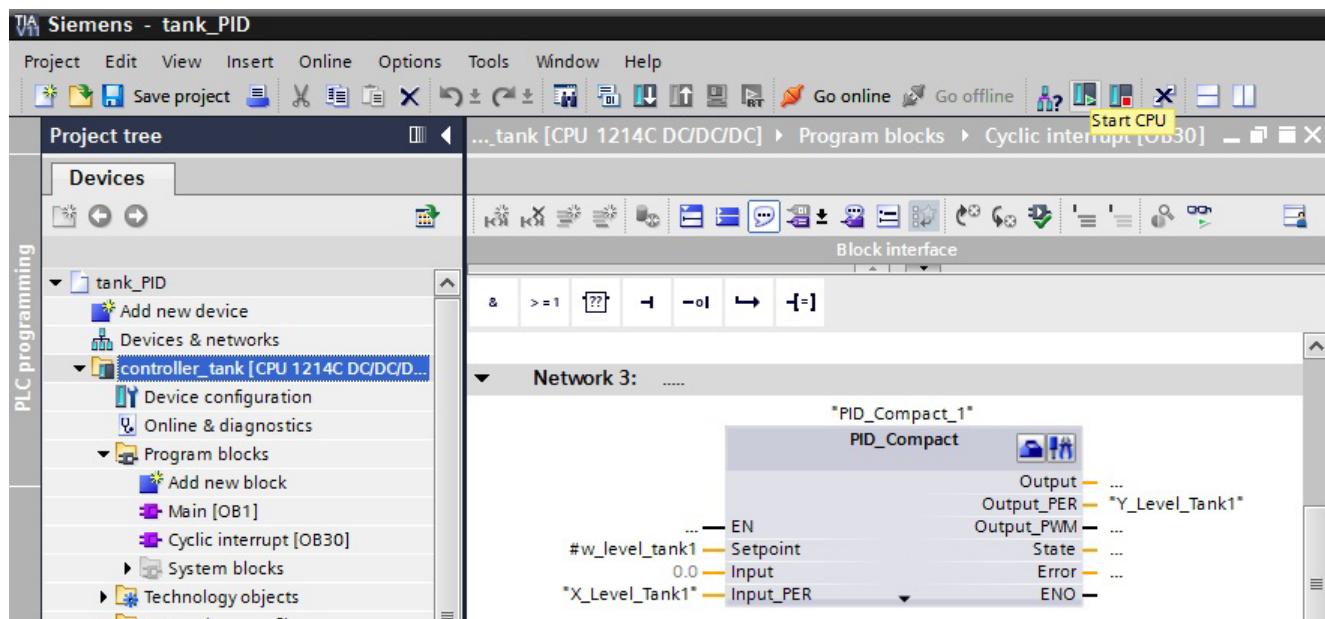


6

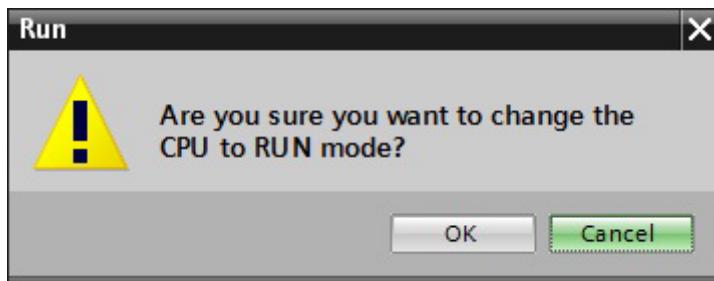
22. The successful download is now displayed in a window. Click on 'Finish'. (→ Finish)



23. Next, start the CPU by clicking on the symbol . (→ 

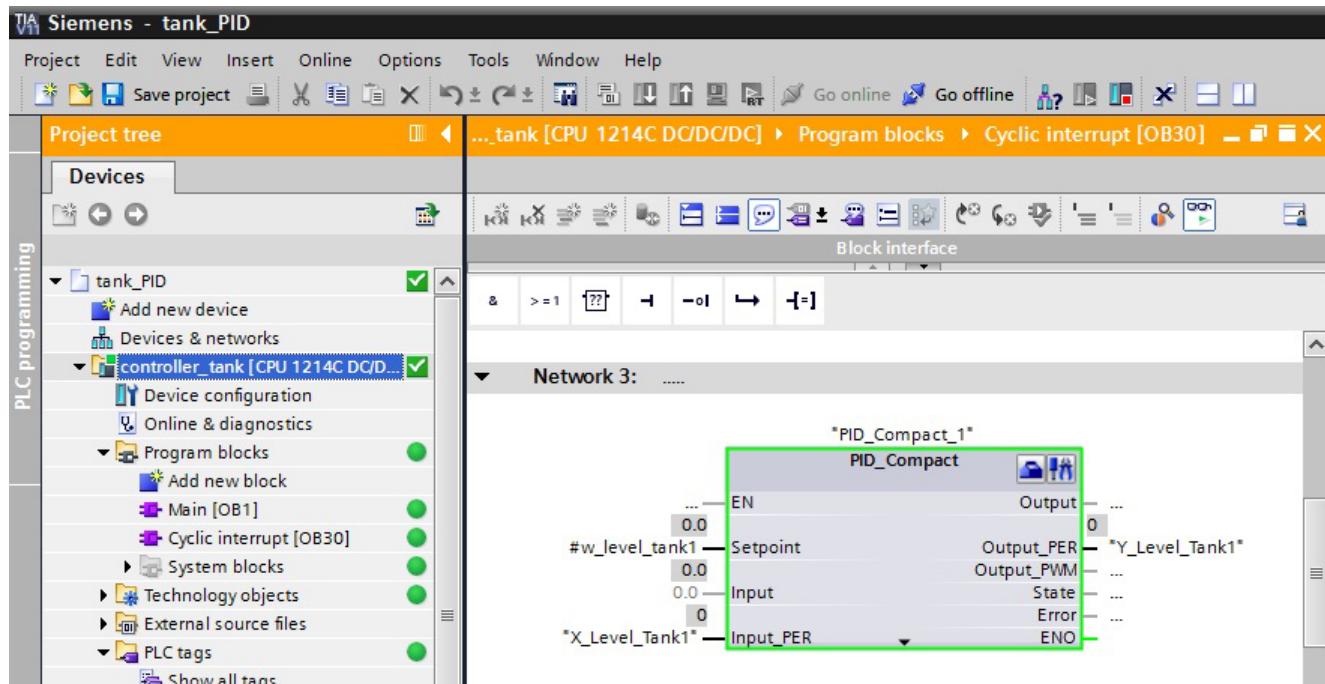


24. Confirm the question whether you are sure you want to start the CPU with 'OK'. (→ OK)



25. By clicking on the symbol  Monitoring on/off, the status of the blocks and the tags can be monitored while the program is tested.

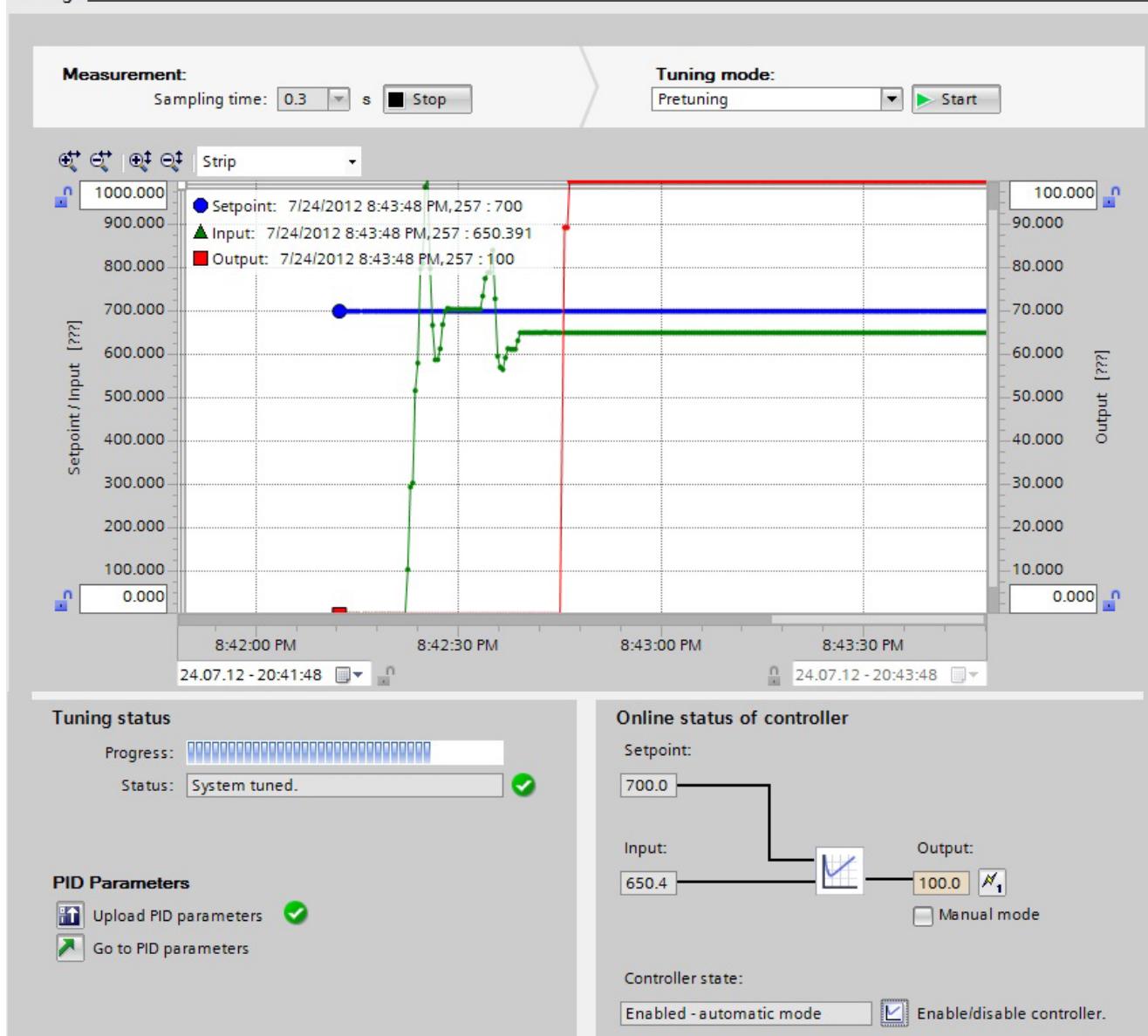
However, when initially starting the CPU, the controller '**PID_Compact**' is not activated. To this end, start commissioning by clicking on the symbol . (→ Cyclic interrupt[OB200] →  → PID_Compact →  Commissioning)



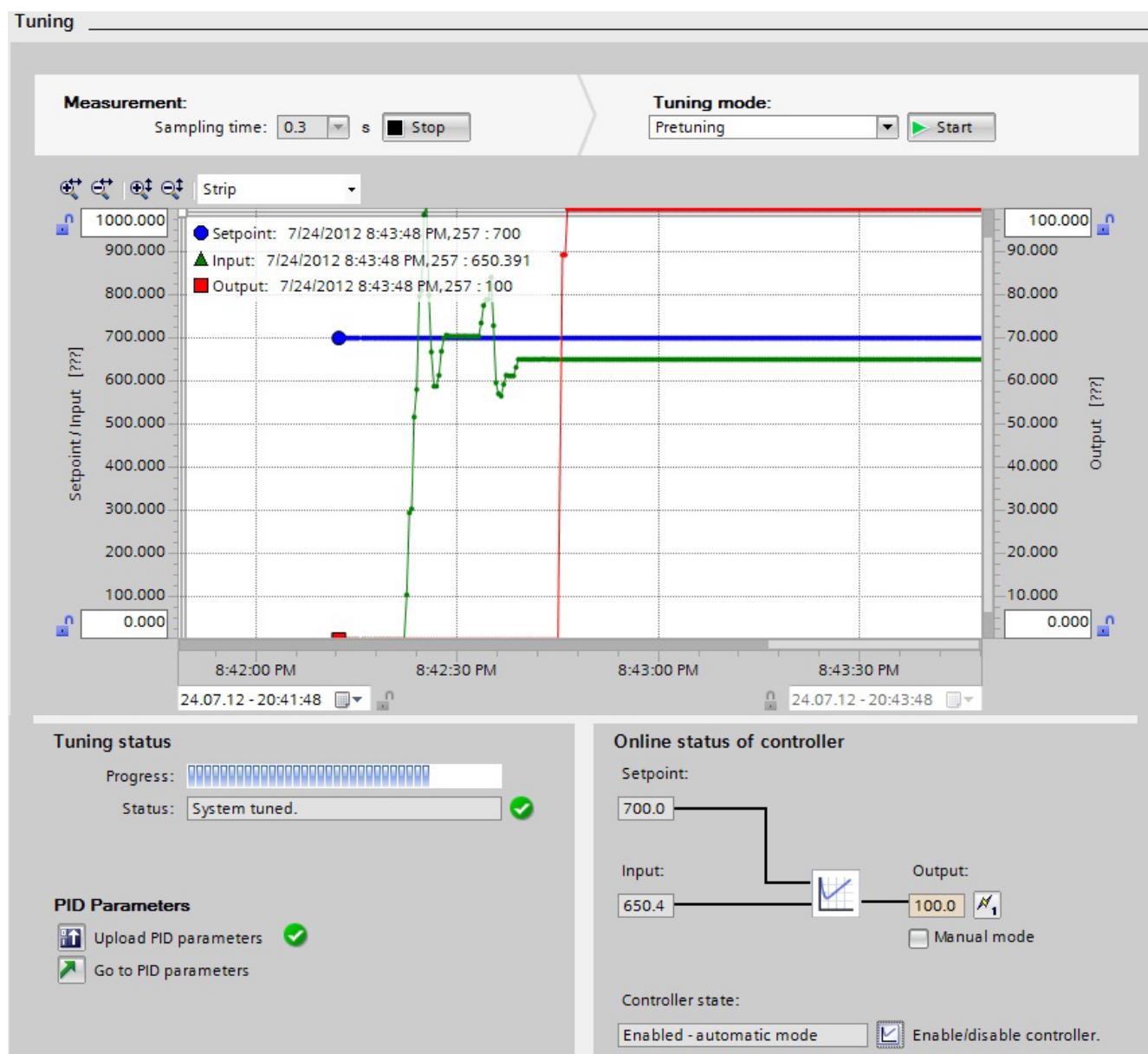
26. On an operating screen, the following can be displayed with 'Measurement Start/Stop' in a diagram: setpoint, actual value and manipulated variable.

After the controller is initially loaded to the control system, it is still inactive. That means that the manipulated variable remains at 0%. Now, select 'Tuning mode' 'Pretuning' and then 'Pretuning' 'Start'. (→ Measurement → Start → Tuning mode → Pretuning → Start)

Tuning

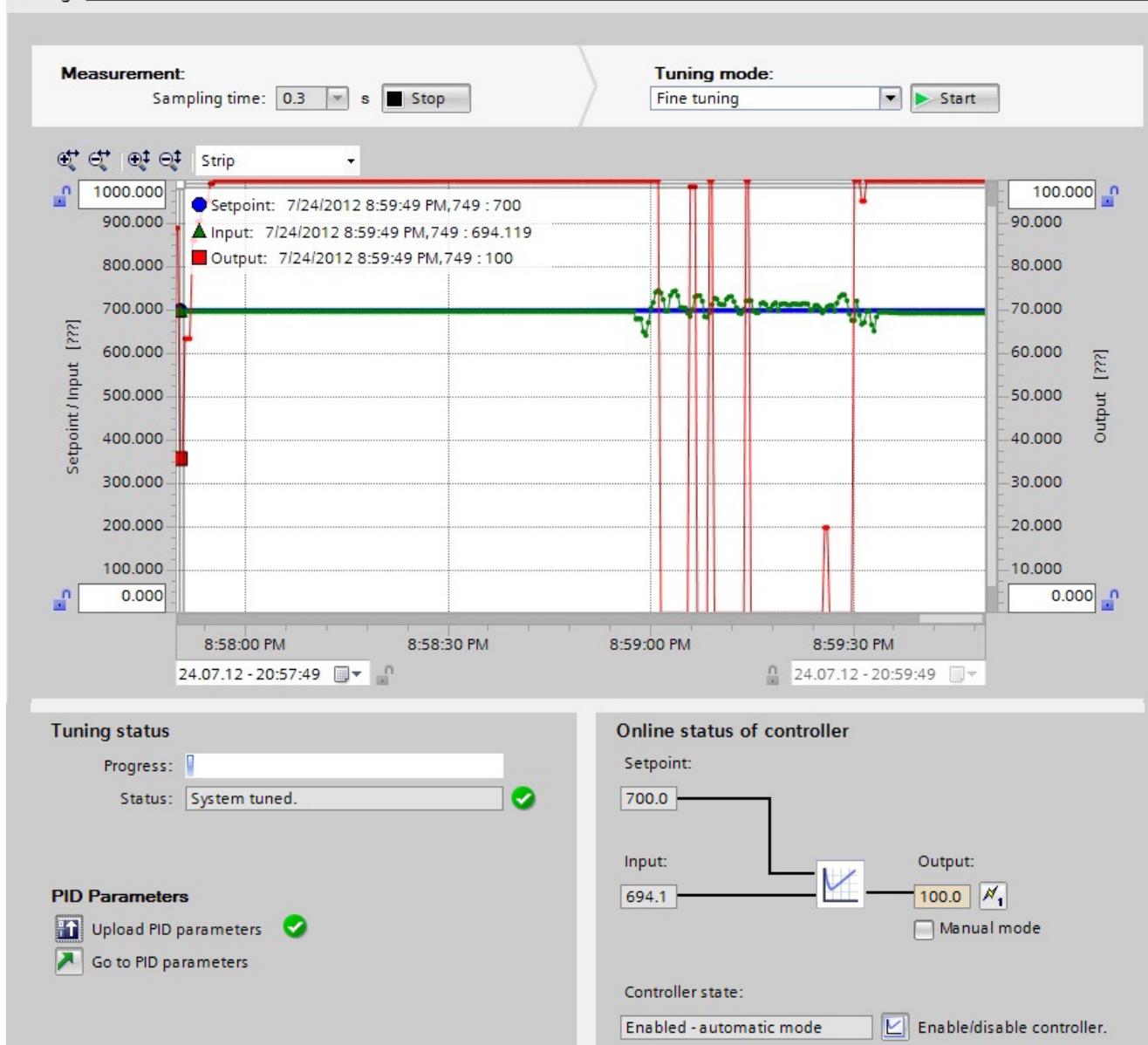


27. Now, self tuning starts. In the field '**Tuning status**' the current work steps and errors that occur are displayed. The progress bar shows the progress of the current work step.



28. If self tuning was executed without error messages, the PID parameters have been optimized. The PID controller changes to the automatic mode and uses the optimized parameters. The optimized parameters are retained at power ON and a CPU restart. With the button '' the PID parameters can be loaded to the project. (→ )

Tuning



Note:

For faster processes, such as speed control, **fine tuning** should be selected for optimization. A cycle is executed here that lasts several minutes where all PID parameters are determined and set.

The parameters values can be monitored in the data block after the project is loaded.

Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Module 010-070



TIA Portal Module 010-010

Startup Programming of the
SIMATIC S7-1200

TIA Portal Module 010-020

Block Types for the SIMATIC S7-1200

TIA Portal Module 010-030

IEC Timers and IEC Counters at the
SIMATIC S7-1200

TIA Portal Module 010-040

Diagnosis and Error Search at the
SIMATIC S7-1200

TIA Portal Module 010-050

Analog Value Processing with
SIMATIC S7-1200

TIA Portal Module 010-060

Control Engineering with
SIMATIC S7-1200

TIA Portal Module 010-070

Communication between two
SIMATIC S7-1200

TIA Portal Module 010-080

HMI Panel KTP600 with SIMATIC
S7-1200

TIA Portal Module 010-090

Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

Matching SCE training packages for these documents

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
Order number: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
Order number 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
Order number 6ES7822-0AA01-4YE0

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

Continued Training

For regional Siemens SCE continued training, please contact your regional SCE contact person
siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

7

Information regarding Usage

The training curriculum 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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by 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.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the documents being used commercially.

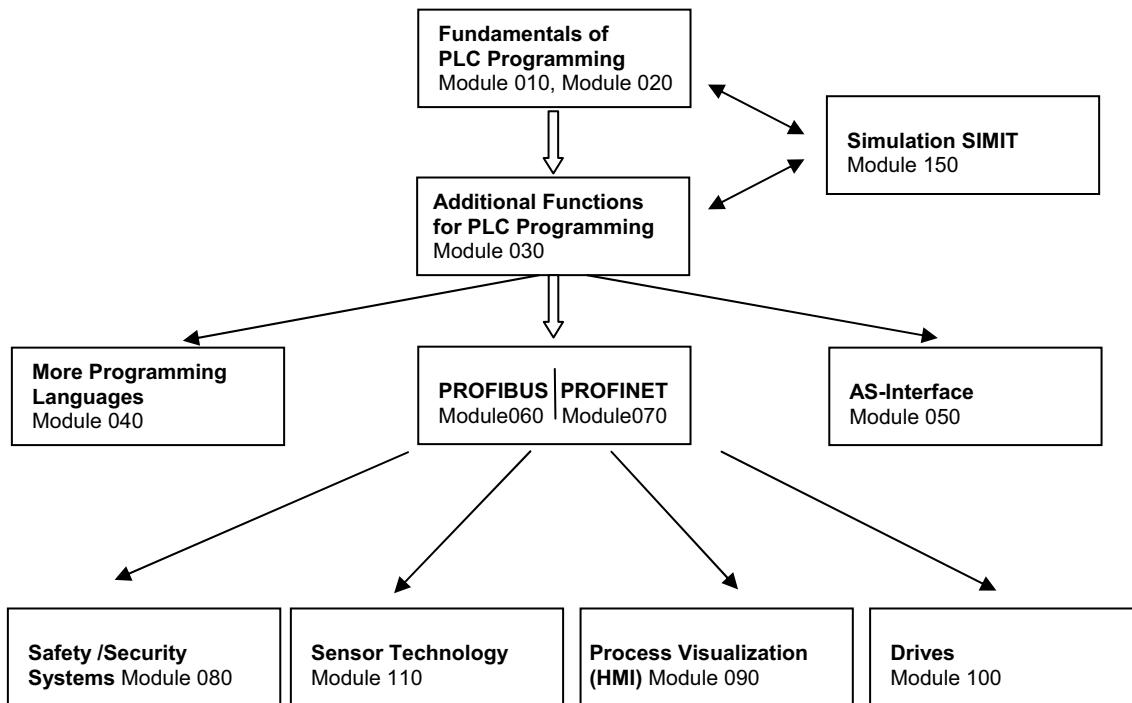
We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this document.

Page:

1.	Preface	4
2.	Instructions on Programming the SIMATIC S7-1200.....	6
2.1	Automation System SIMATIC S7-1200.....	6
2.2	Programming Software STEP 7 Professional V11 (TIA Portal V11)	6
2.3	SIMATIC NET Switch CSM 1277.....	7
3.	Conveyor Control with Counter and Multi-Instance	8
3.1	Task.....	8
4.	Expanding the Hardware Configuration in the Project Conveyor Control	9
4.1	Loading the Sample Project and Re-Storing It	9
4.2	Adding the Second CPU	13
4.3	Connecting the Controllers	15
4.4	Loading the Hardware Configuration to the CPUs.....	16
5.	Programming the Program Blocks	21
5.1	Send Block TSEND_C	21
5.2	Control Program for the Controller_Conveyor	24
5.3	Receive Block TRCV_C	30
5.4	Control Program for controller_data.....	35

1. Preface

Regarding its content, module 010-070 is part of the training unit '**Basics of PLC Programming**' and explains the communication between two SIMATIC S7-1200 controllers.



7

Training Objective

In this module 010-070, the reader learns how to configure an ISO on TCP communication connection between two SIMATIC S7-1200 controllers, and how to program the data exchange in the control program using the communication blocks TSEND_C and TRCV_C.

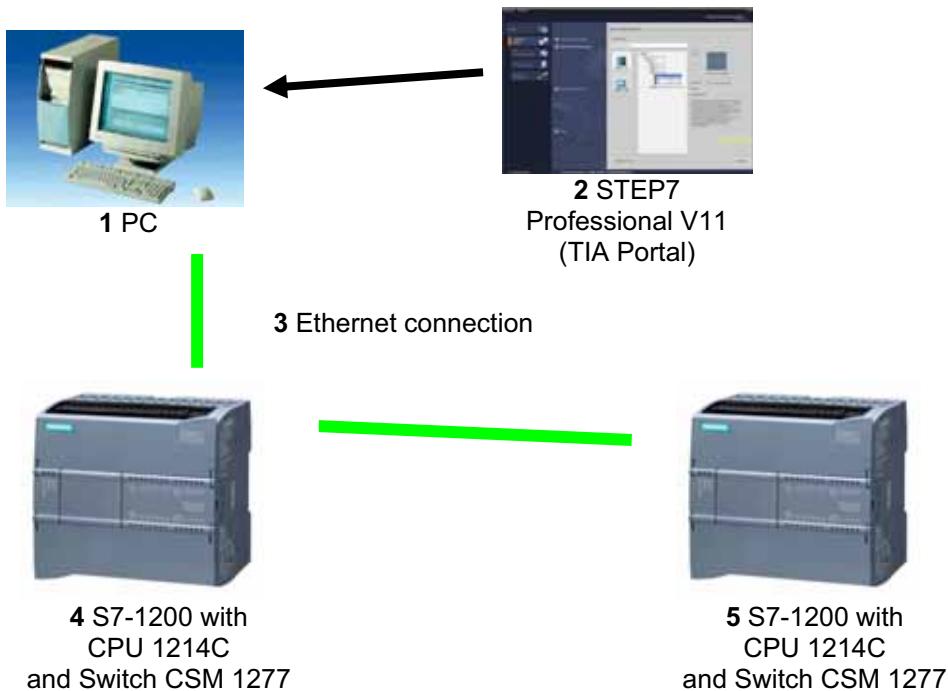
Prerequisites

To successfully work through this module 010-070, the following knowledge is assumed:

- How to handle Windows
- Basics of PLC programming with the TIA Portal
(for example, Module 010-010 – 'Startup' programming of the SIMATIC S7-1200 with TIA- Portal V11)
- Blocks for the SIMATIC S7-1200
(for example, Module 010-020 – block types for the SIMATIC S7-1200)
- Timer and counter blocks for the SIMATIC S7-1200
(for example, Module 010-030 – Multi-instances at the SIMATIC S7-1200)

Hardware and software needed

- 1** PC Pentium 4, 1.7 GHz, 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB
Operating system Windows XP (Home SP3, Professional SP3)/Windows Vista (Home Premium SP1, Business SP1, Ultimate SP1)
- 2** Software STEP7 Basic V11 SP2 (Totally Integrated Automation (TIA) Portal V11)
- 3** Ethernet connection between PC and CPU 1214C and panel
- 4** PLC 1 SIMATIC S7-1200; for example, CPU 1214C with Switch CSM 1277.
The inputs have to be brought out to a panel.
- 5** PLC 2 SIMATIC S7-1200; for example, CPU 1214C with Switch CSM 1277.
The inputs have to be brought out to a panel.



2. Instructions on Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The automation system SIMATIC S7-1200 is a modular mini-controller system for the lower and medium performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU and input/output modules for digital and analog signals.

If needed, communication processors and function modules are added for special tasks such as step motor control.

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process, whereby the IO modules are polled in the S7 program by means of the input addresses (%I) and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7.

2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 (TIA Portal V11) is the programming tool for the following automation systems

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining the communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating the visual displays for the SIMATIC basic panels with the integrated WinCC Basic
- With additional WinCC packages, visual display solutions for PCs and other panels can be prepared also.

All functions are supported with detailed online help.

2.3 SIMATIC NET Switch CSM 1277

System Description



The CSM 1277 is equipped with four RJ45 socket connectors to connect terminal equipment or additional network segments.

The TP interfaces are laid out as RJ45 socket connector with MDI-X assignment (MediumDependent InterfaceAutocrossover) of a network component. If the receive cable pair is connected wrong (RD+ and RD-reversed), the polarity is adjusted automatically. The MDI/MDIX autocrossover function offers the advantage of integrated cabling without external crossed Ethernet cables being needed. This prevents malfunctions if send and receive lines should be inadvertently switched.

This considerably simplifies the installation for the user.

The CSM 1277 is a plug&play device that does not require setting to be started.

CSM 1277 Displays

Power display 'L' (green LED).

The status of voltage supply is signaled by means of a green LED

State	Description
LED is green	Voltage supply is connected
LED not lit	Voltage supply is not connected, or the applied voltage is too low

Port status display 'P1' to 'P4' (green LED)

The status of the interfaces is signaled by means of the 4 green LEDs. They are located below the upper cover. Refer also to Figure 4-4.

State	Description
Port 1 to Port 4 LED is lit	Existing connection via port to Industrial Ethernet (LINK status)
Port 1 to Port 4 LED flashes	Port is sending /receiving via Industrial Ethernet
Port 1 to Port 4 LEDs are flashing/running light	Test phase during power on

3. Conveyor Control with Counter and Multi-Instance

Below, the example for conveyor control is to be expanded with a counter and a multi-instance for the communication of two S7-1200 controllers. <>or: Below, the example for conveyor control with a counter and a multi-instance is to be expanded for the communication of two S7-1200 controllers?>>

With the conveyor, 20 bottles are always to be transported in a case. When the case is full, the conveyor is stopped, and the case has to be exchanged.

With button 'S1' the operating mode 'Manual' and with button 'S2' the operating mode 'Automatic' can be selected.

In the operating mode 'Manual', the motor is switched on as long as button 'S3' is operated; button 'S4' must not be operated.

In the 'Automatic' mode, the conveyor motor is switched on with button 'S3' and switched off with button 'S4' (NC).

In addition, there is a sensor 'B0' that counts the bottles into the case. After 20 bottles are counted, the conveyor is stopped...

When a new case is put in place, it has to be confirmed with 'S5'.

Assignment list

Address	Symbol	Comment
%I 0.0	S1	Button manual mode S1 NO
%I 0.1	S2	Button automatic mode S2 NO
%I 0.2	S3	On button S3 NO
%I 0.3	S4	Off button S4 NC
%I 0.6	S5	Button S5 NO reset counter/new case
%I 0.7	B0	Sensor B0 NO bottle counter
%O 0.2	M01	Conveyor motor M01

3.1 Task

The states of the input signals are to be sent to the second CPU for further processing and displayed there.

The inputs of the first CPU are to be written to the Send buffer.

With the send block "TSEND_C", the content of the Send buffer is to be sent to the second CPU by means of an ISO on TCP connection.

Inputs of CPU1 from %I0.0 to %I0.7 (%IB0) to the – %MB10 (send buffer) – "TSEND_C" – CPU2.

At the second CPU, the received data is to be written to the receive buffer by means of the receive block "TRCV_C". Then the content of the receive buffer is to be displayed at the CPU's outputs.

CPU2 "TRCV_C" – %MB12 (receive buffer) – to the outputs of %O0.0 to %O0.7 (%OB0).

4. Expanding the Hardware Configuration in the Project Conveyor Control

For project management and programming, the software '**Totally Integrated Automation Portal**' is used.

Here, on a uniform interface, components such as controlling, visualizing and networking the automation solution are set up, parameterized and programmed.

Online tools are available for error diagnosis.

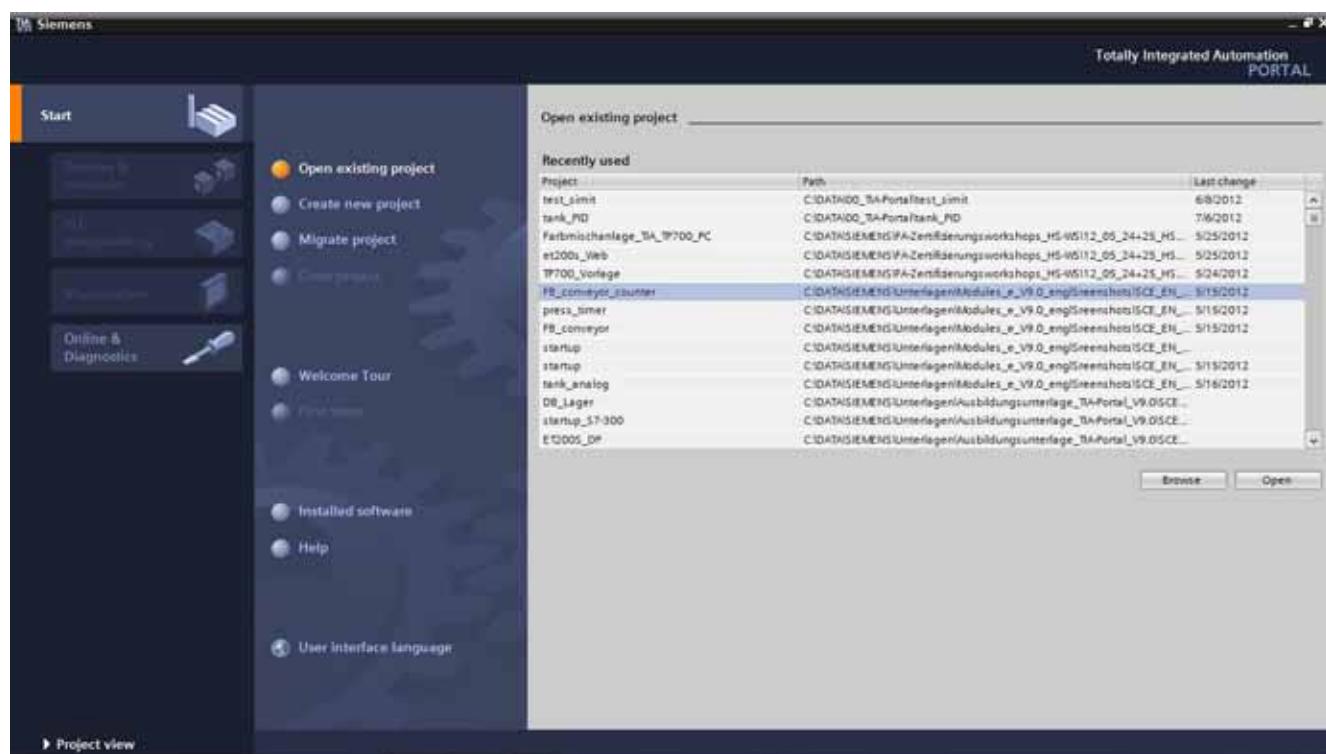
In the steps below, the following is to be done: for the SIMATIC S7-1200, a project is to be opened, stored under a different name and adapted to the new requirement:

The central tool is the '**Totally Integrated Automation Portal**'. It is called here with a double click.

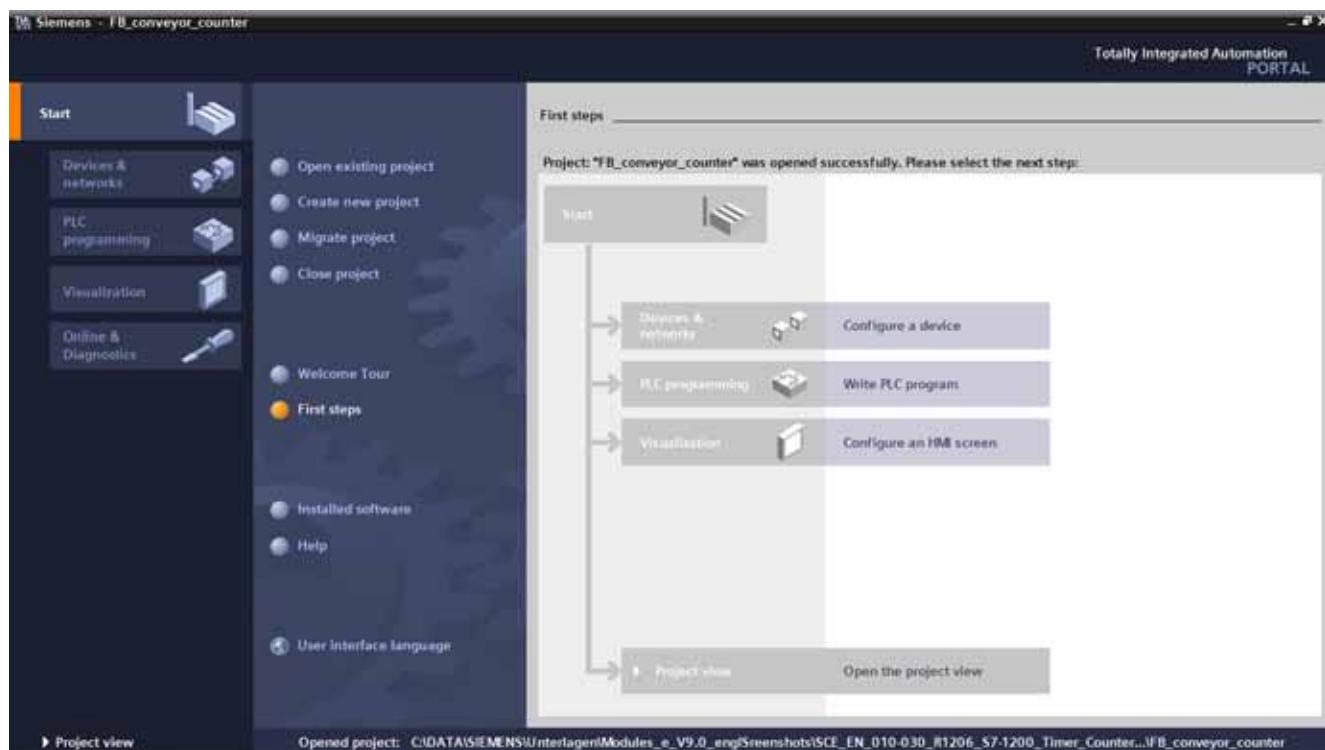


4.1 Loading the Sample Project and Re-Storing It

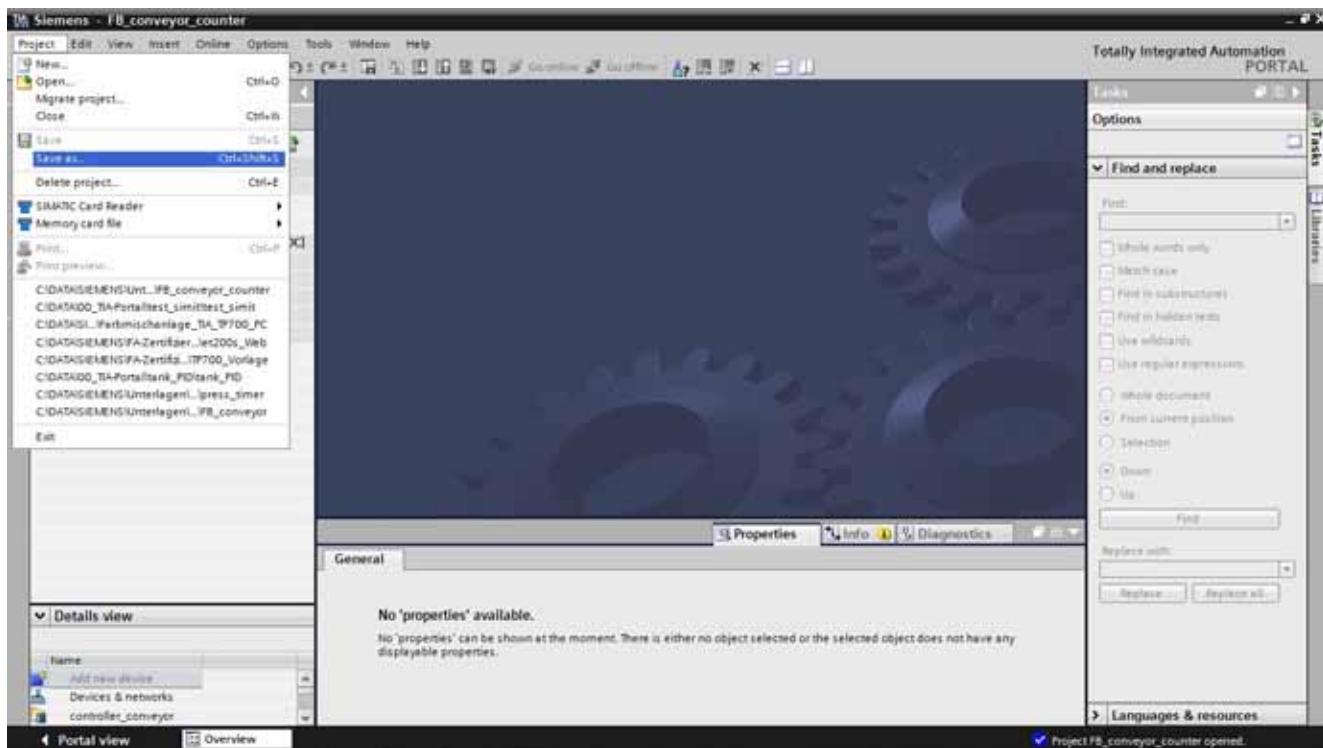
The project "**FB_conveyor_counter**" from Module 010-030 is now opened as a pattern for this program.



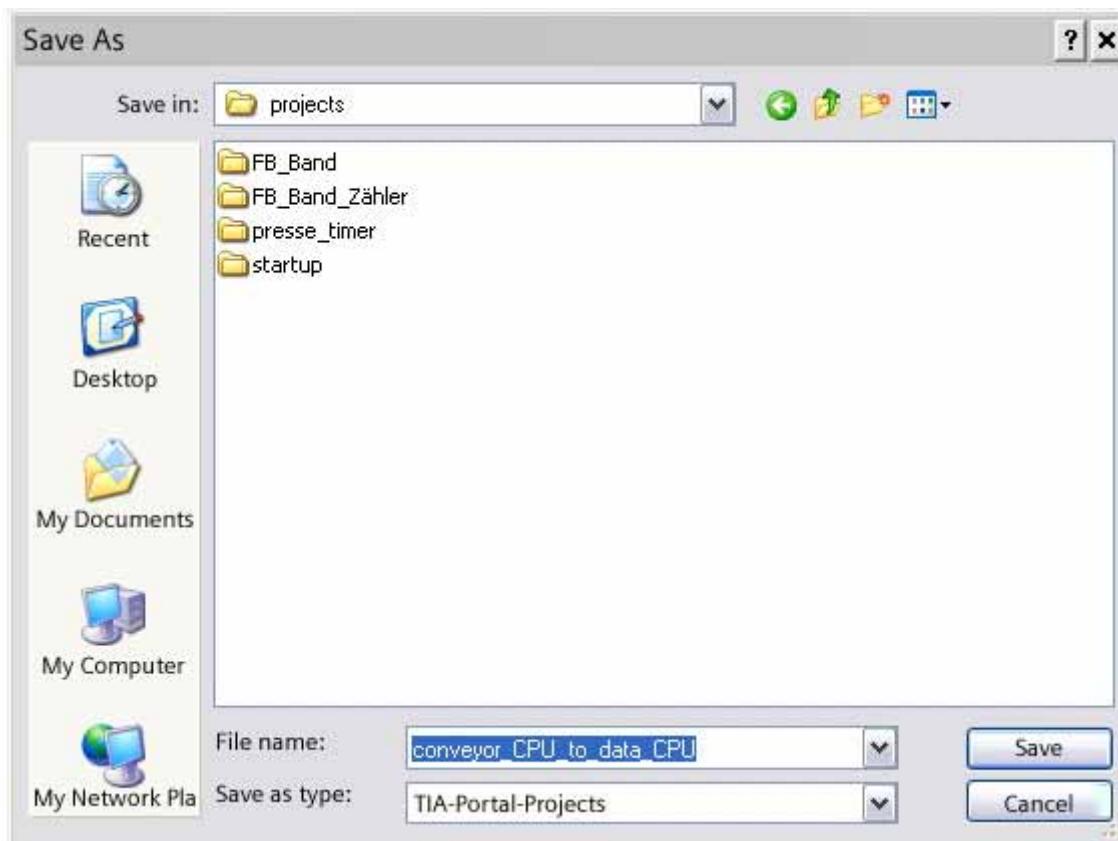
Next, 'First steps' are suggested. Click on **Open the project view**.



Now, the project is to be saved under another name. In the menu **Project** click on "**Save As**".



Now, 'Save' the project under the new name '**conveyor_CPU_to_data_CPU**'.



4.2 Adding the Second CPU

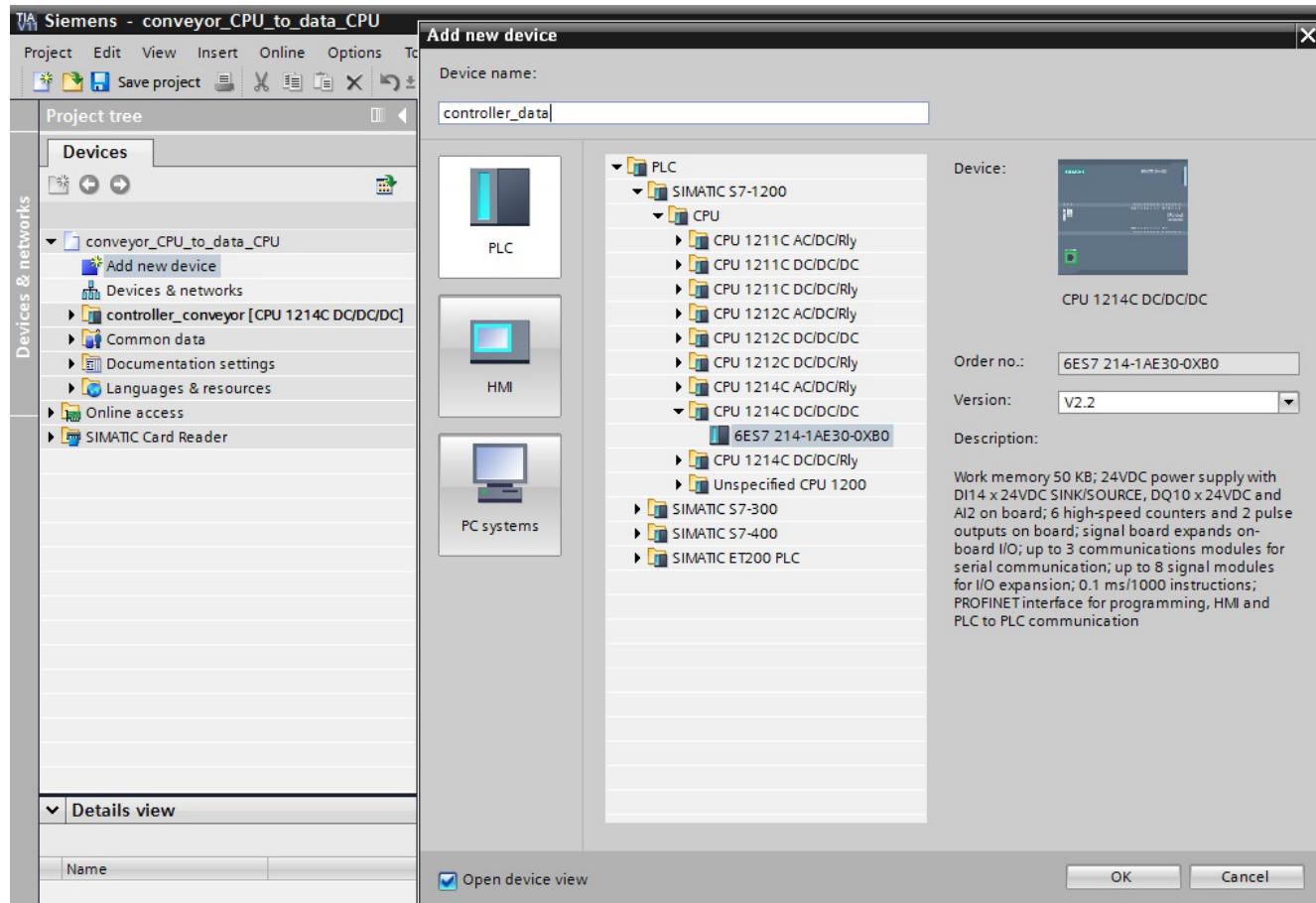
To set up a second CPU in the project, open the list box with a double click on 'Add new device'.

Under SIMATIC S7-1200, select CPU 1214C with the order number "6ES7 214-1AE30-0XB0".

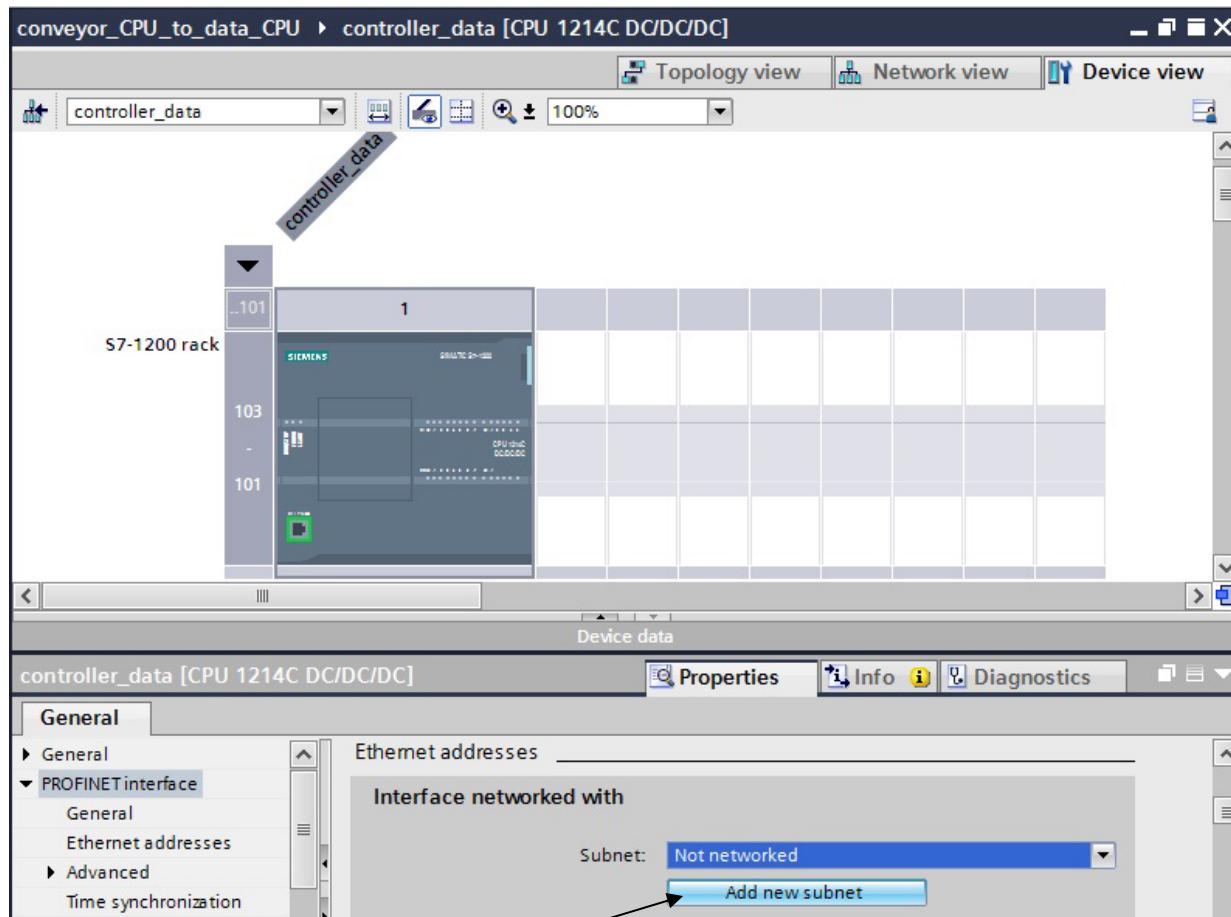
As device name, assign "**controller_data**"

Place a check mark at "**Open device view**"

Click "**OK**".

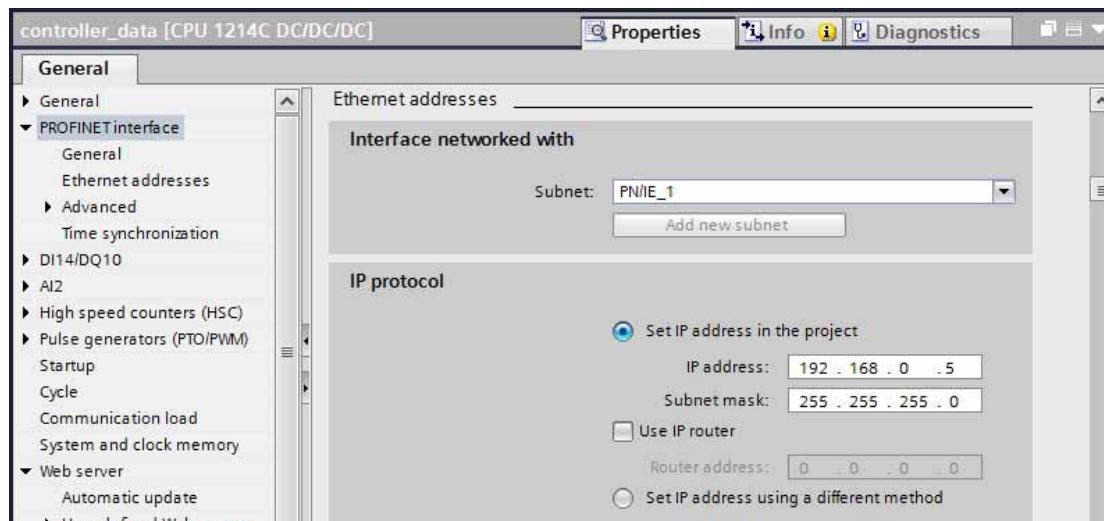


In the lower window of the device view, under the tab **Properties** select the **PROFINET interface**.



Then click on "**Add new subnet**".

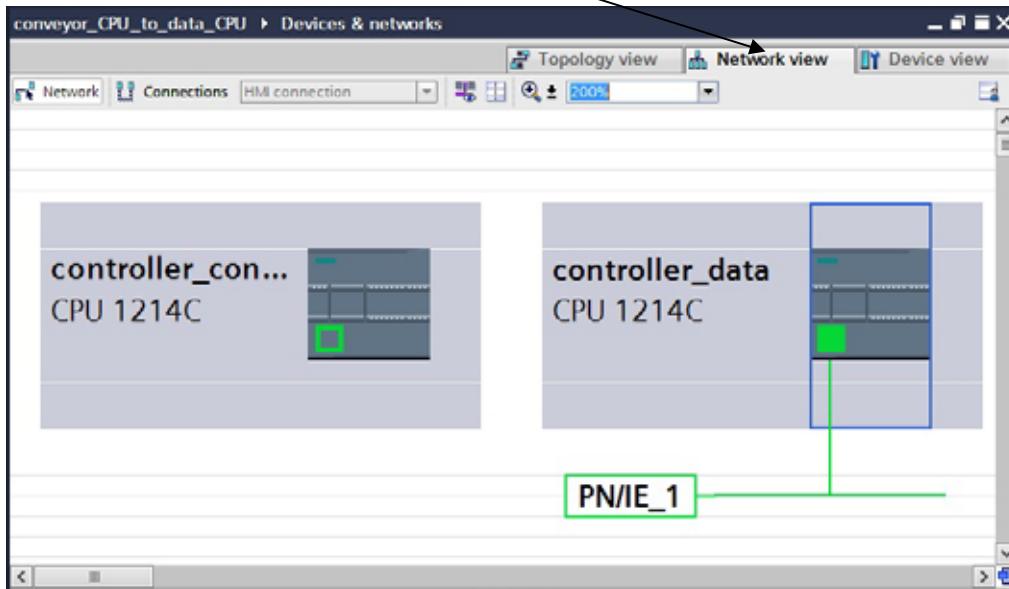
Assign the IP address **192.168.0.5** and the subnet mask **255.255.255.0**



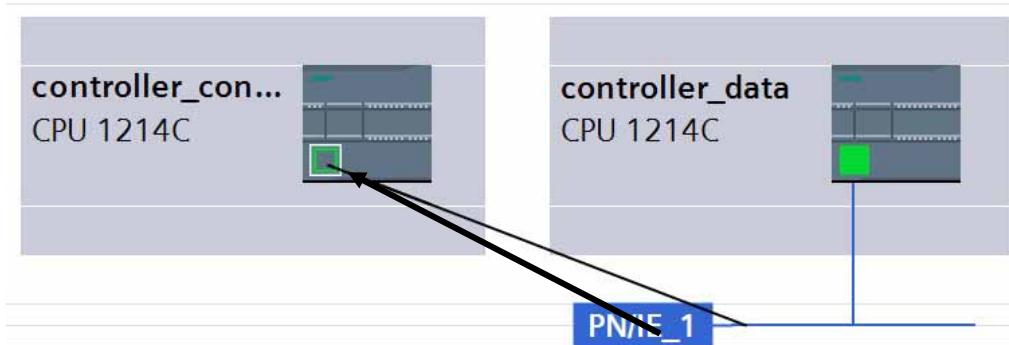
Now click on "**Save project**".

4.3 Connecting the Controllers

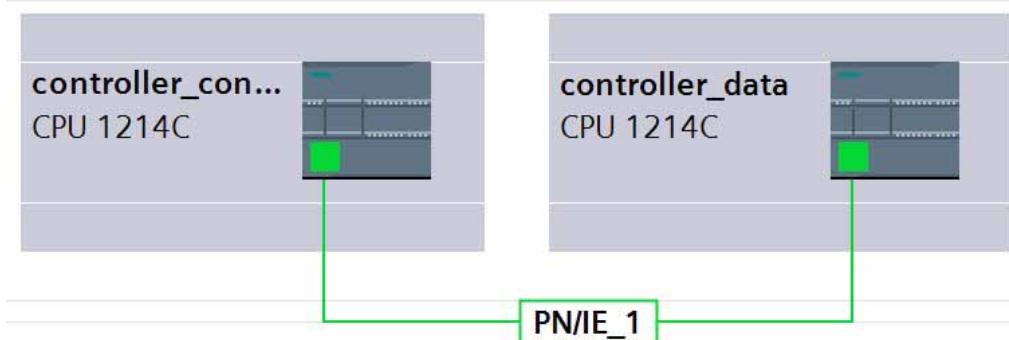
Change to the Network view.



Drag the line of the PN/IE_1 network to the green square of the controller_conveyor.



The two CPUs are now connected.

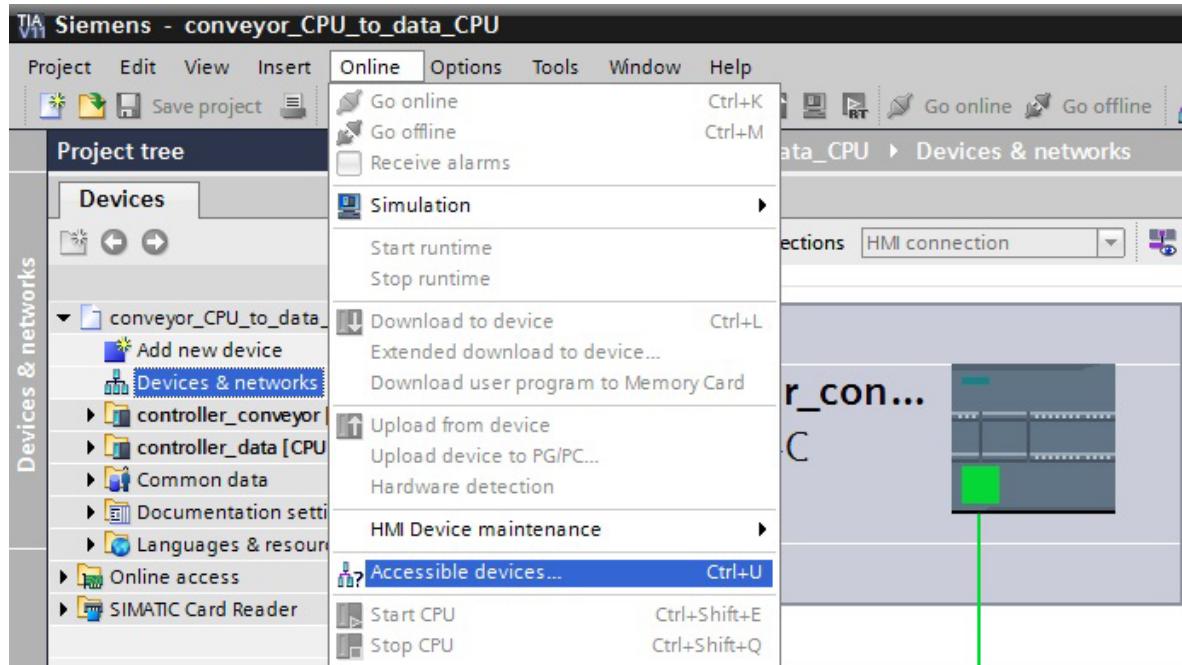


Then, click on "Save project".

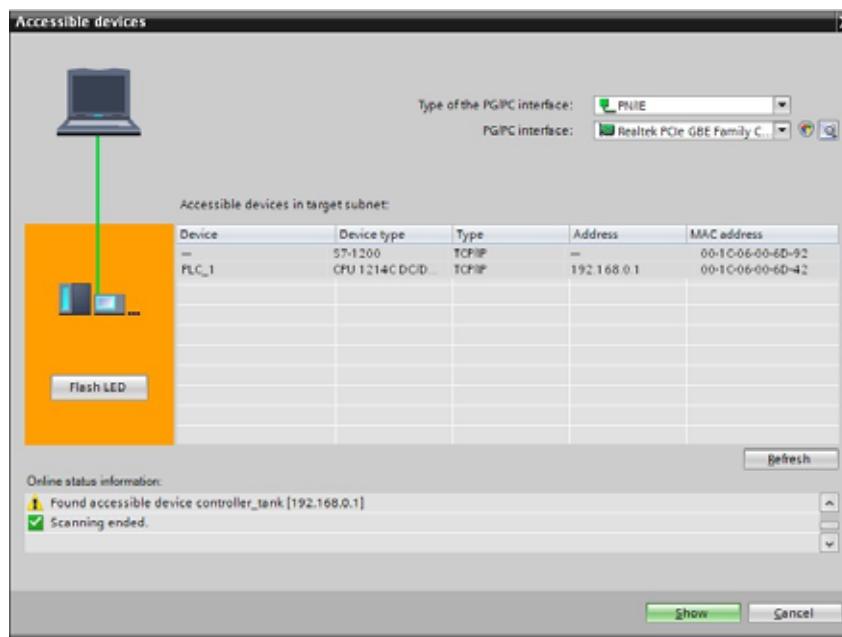
4.4 Loading the Hardware Configuration to the CPUs

Prior to the steps below, we have to establish the network wiring between the two controllers and the programming device by means of the switch CSM 1277.

In the menu **Online**, click on “**Accessible devices**”



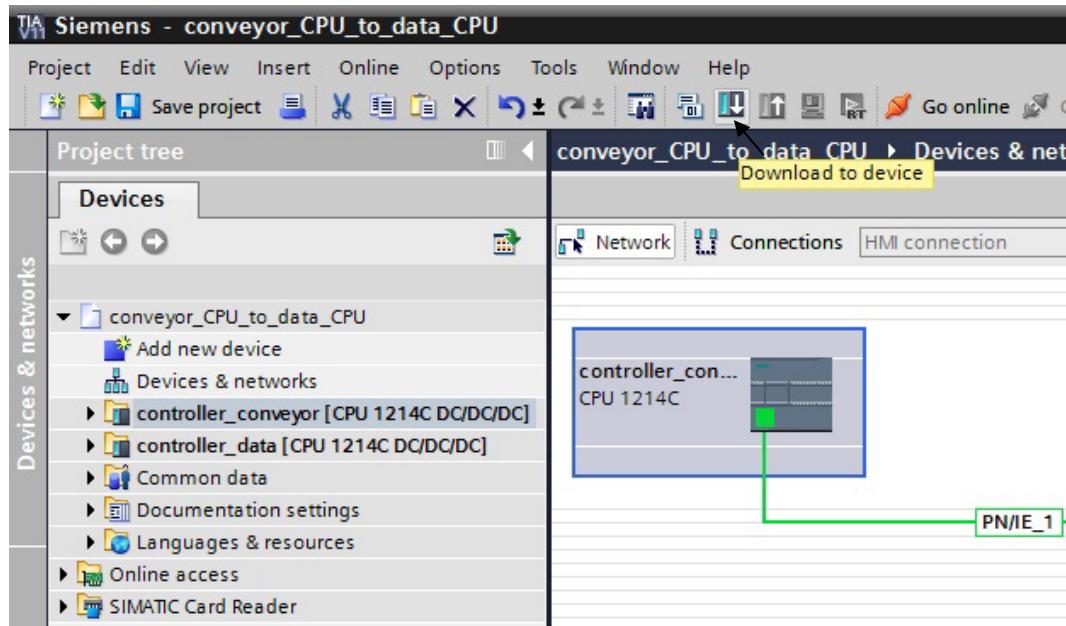
The second controller still has the factory settings and no IP address.



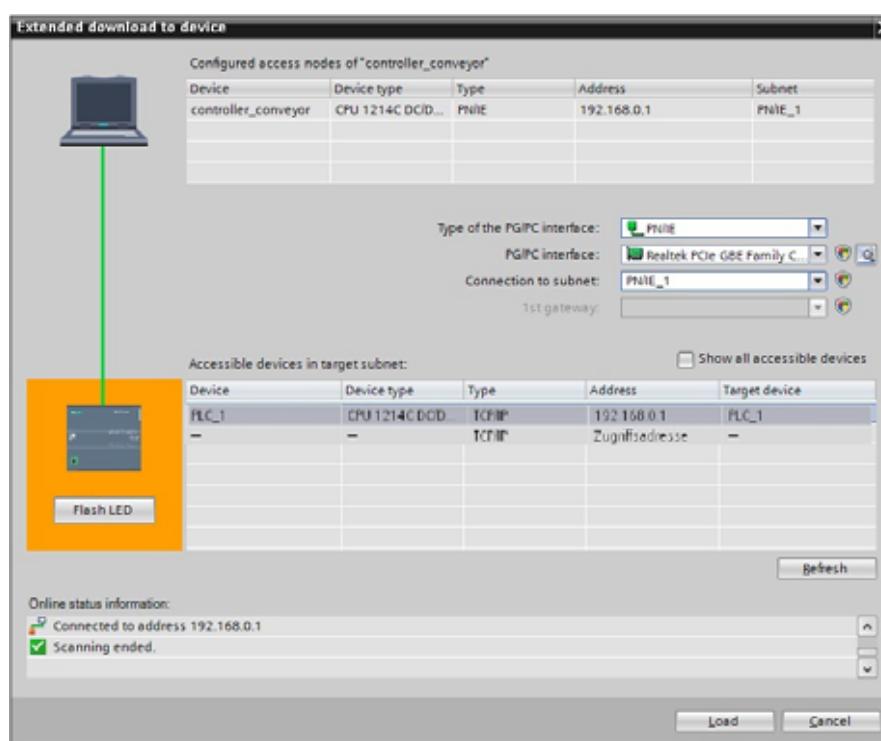
Now click on "Cancel".

Loading the first controller conveyor.

In the project navigation window, select “**controller_conveyor**” and click on the button **Download to device**.

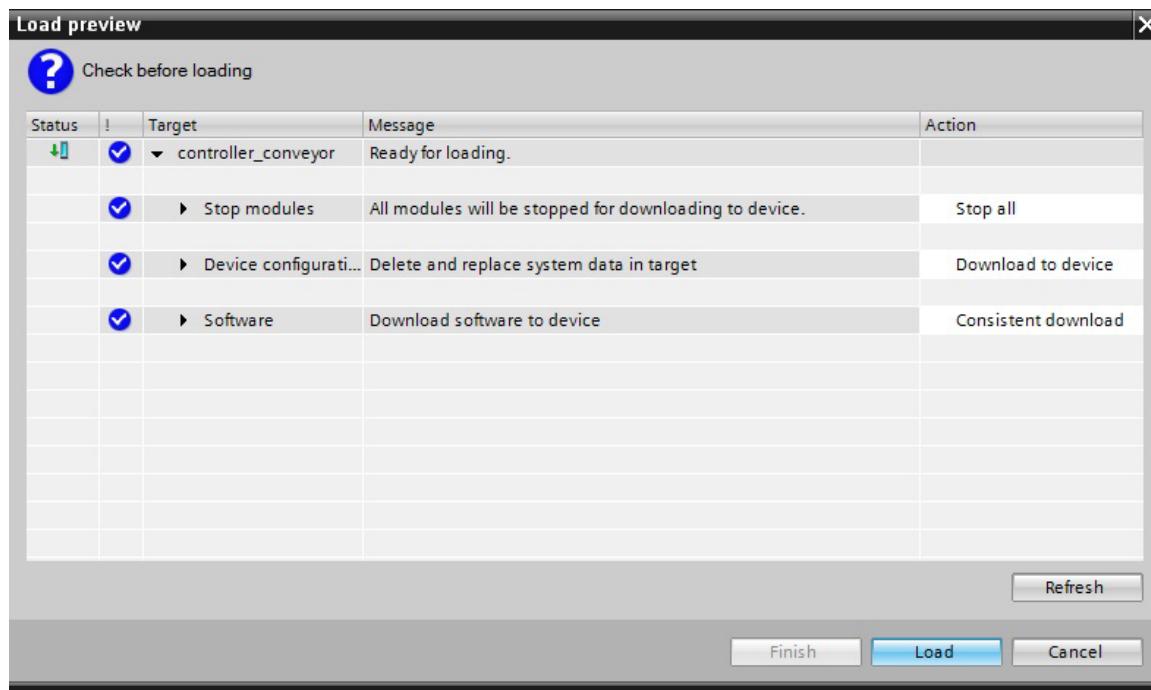


If the CPU is in the RUN mode, it has to be set to STOP prior to the download.
Select the PG/PC interface for the download and the connection with subnet.

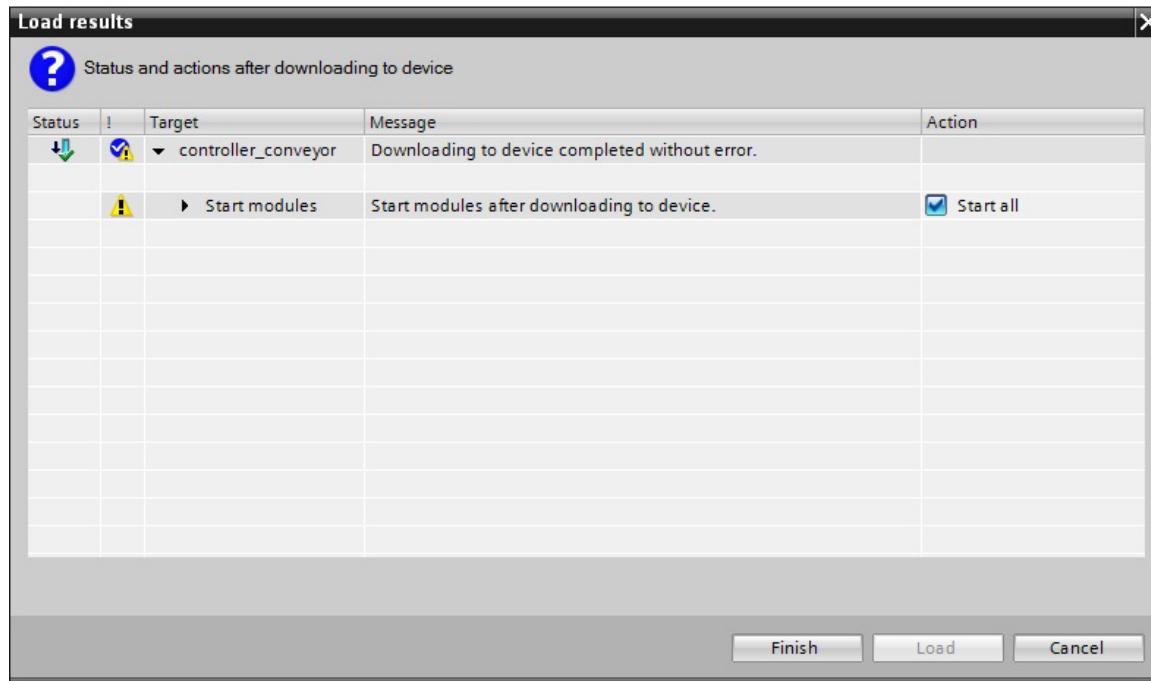


Now click on “**Load**“.

Prior to downloading, the project data is checked and compiled.



Then click on "Load".

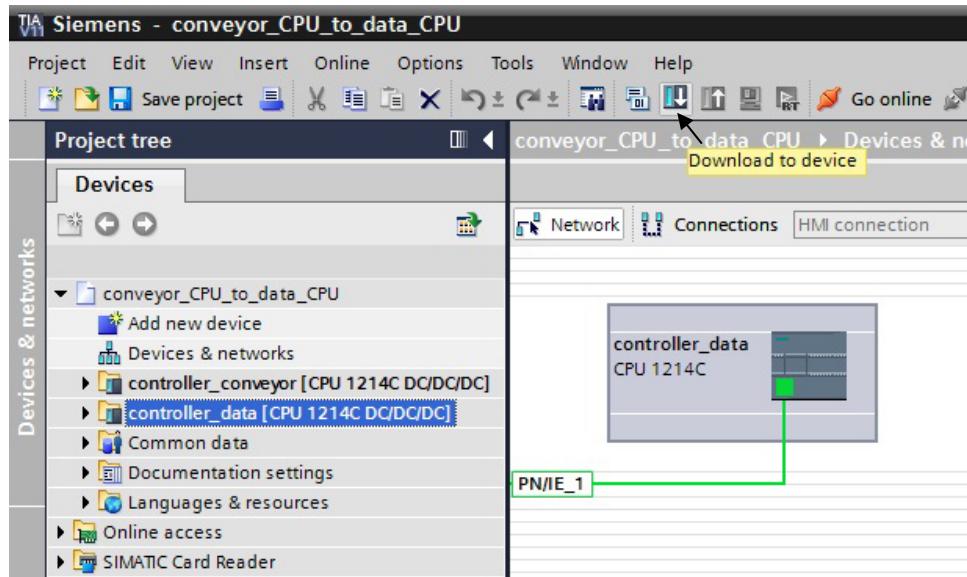


Now click on "Finish".

The CPU returns to the RUN mode.

Downloading Data to Second Controller.

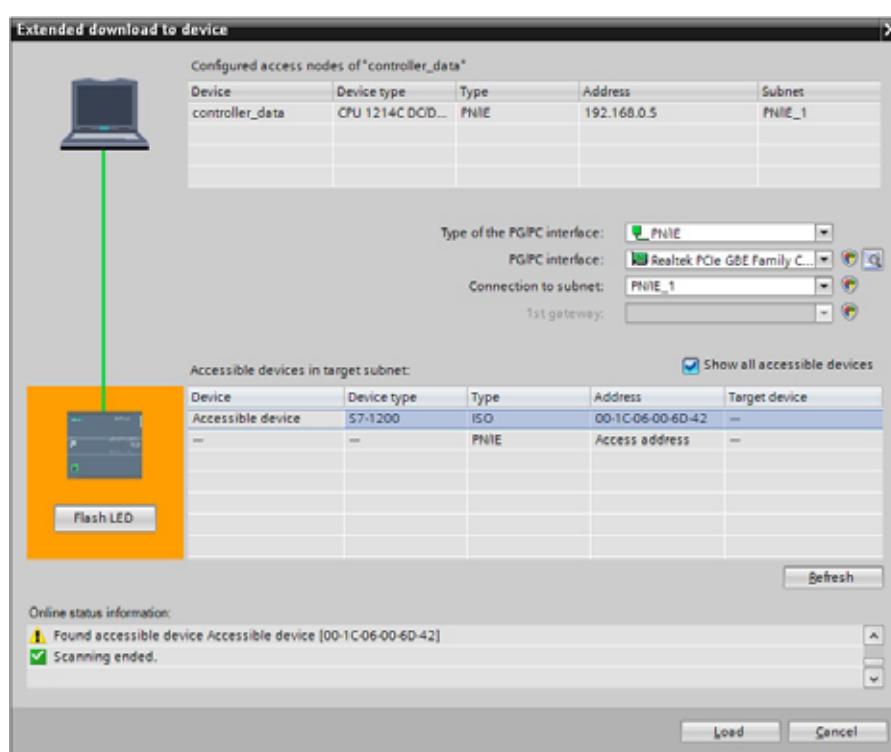
In the project navigation window, select "**controller_data**" and click on the button **Download to device**.



Select the PG/PC interface for the download and the connection with subnet.

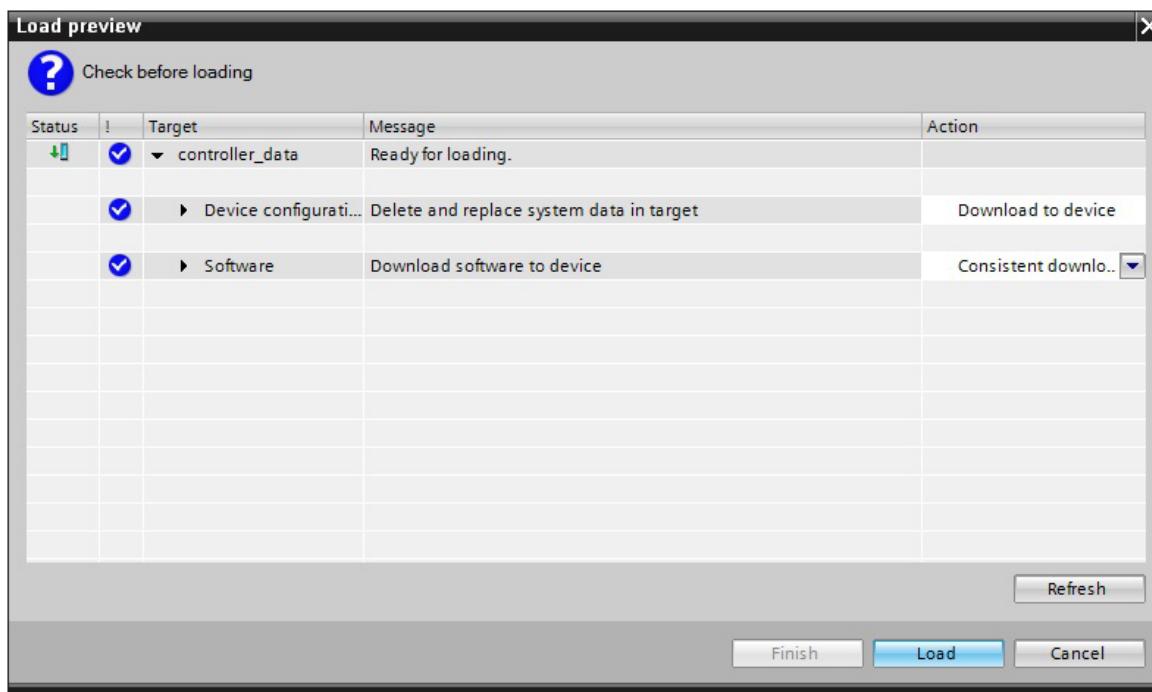
Place a checkmark at **Show all accessible devices**.

Select the S7-1200 CPU.



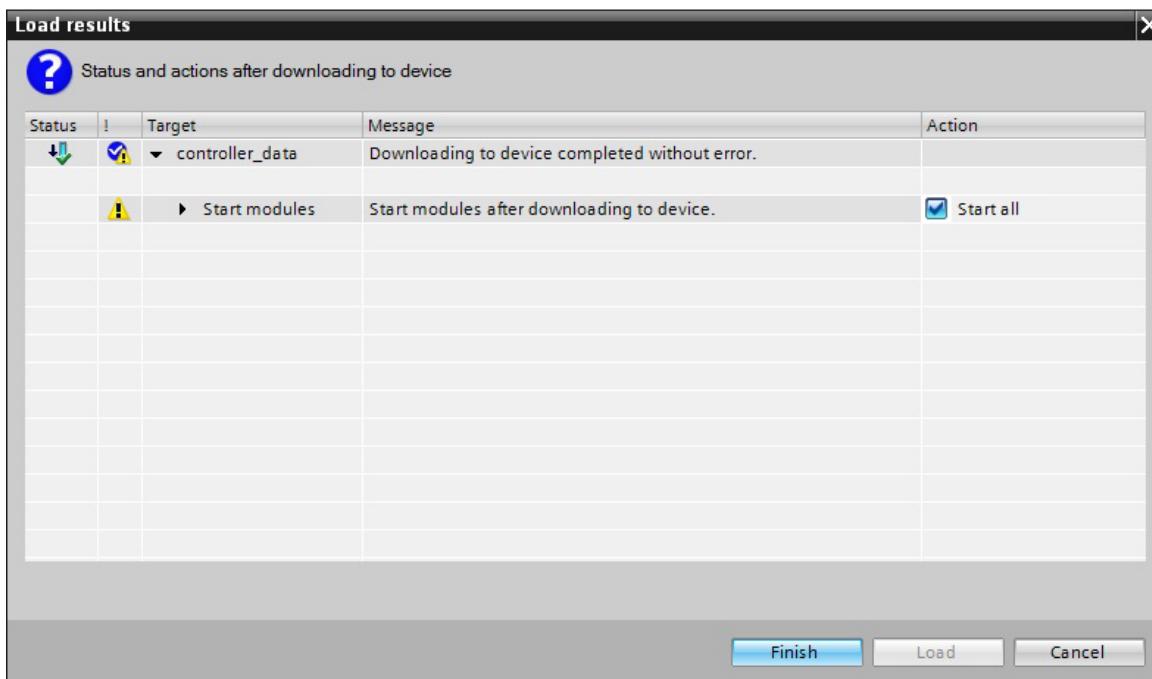
Then click on "**Load**".

Prior to the download, the project data is checked and compiled.



Then, click on “**Load**“.

Loading writes the IP address to the CPU.



Then, click on “**Finish**“.

The CPU is taken to the RUN mode.

5. Programming the Program Blocks

Now, the required program blocks for the communication between the two CPUs are to be called and parameterized.

By calling the transmission blocks TSEND_C and TRCV_C, a connection is established automatically between the two CPUs. The connection is set up, established, and monitored automatically until an operation or CPU STOP separates the connection.

5.1 Send Block TSEND_C

Description

TSEND_C is executed asynchronously and has the following functions:

Setting up and establishing a communication connection:

TSEND_C sets up a TCP or ISO-on-TCP communication connection and establishes it. After the connection is set up and established, the CPU maintains and monitors it automatically.

To set up the communication connection, the connection description specified at the parameter CONNECT is used. To establish the connection, the parameter CONT has to be set to "1". If the connection was established successfully, the parameter DONE is set to "1" for 1 cycle.

When the CPU enters the STOP mode, an existing connection is cancelled and the connection that was set up is removed. To again set up and establish the connection, TSEND_C has to be executed again. The number of possible communication connections is provided in the technical data of your CPU.

Sending data over an existing communication connection:

The transmitting range is specified at the parameter DATA. It includes the address and the length of the data to be sent.

The send request is executed when a rising edge is recorded at the parameter REQ. The maximum number of bytes that is sent with a send request is specified at the parameter LEN. The data to be sent must not be edited until the send request is executed completely. If the send request was executed successfully, the parameter DONE is set to "1". The signal status "1" at parameter DONE is not a confirmation, however, that the communication partner has read the data that was sent.

Disconnecting the communication connection:

The communication connection is disconnected when the parameter CONT is set to "0".

When setting the parameter COM_RST to "1", TSEND_C is executed again. The existing communication connection is disconnected and a new connection established. Data that is transmitted while the connection is re-established may be lost.

Parameters of TSEND_C

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
CONT	Input	BOOL	I, Q, M, D, L	Controls the communications connection: <ul style="list-style-type: none">• 0: Disconnect the communications connection• 1: Establish and maintain the communications connection When sending data (rising edge at the REQ parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum number of bytes to be sent with the job. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Parameters of communication connections
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent.
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: <ul style="list-style-type: none">• 0: Irrelevant• 1: Complete restart of the instruction causing an existing connection to be terminated and a new connection to be established.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: <ul style="list-style-type: none">• 0: Job not yet started or is still executing• 1: Job completed error-free
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: <ul style="list-style-type: none">• 0: Job not yet started or already completed• 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: <ul style="list-style-type: none">• 0: No error• 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

BUSY, DONE and ERROR

With the parameters BUSY, DONE, ERROR and STATUS, we can check the execution status. The parameter BUSY shows the processing status.

The parameter DONE checks whether a request was executed successfully.

The parameter ERROR is set when errors occur while TSEND_C is executed.

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was stopped with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

The error information is provided at the parameter STATUS.

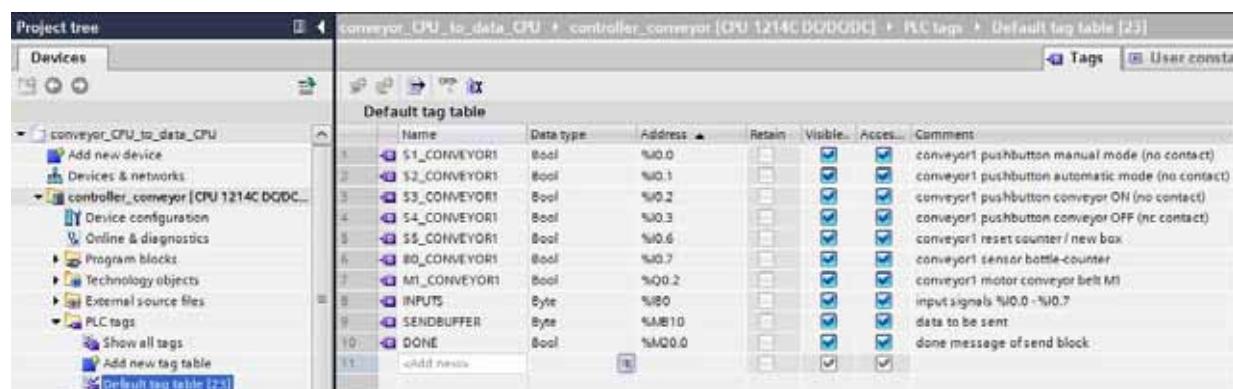
ERROR	STATUS (W#16#...)	Description
0	0000	Job completed error-free
0	7000	No job processing active
0	7001	<ul style="list-style-type: none"> • Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data are being sent
0	7003	Connection is terminated
0	7004	Connection established and monitored, no job processing active.
1	80A0	Group error for error codes 80A1 and 80A2.
1	80A1	<ul style="list-style-type: none"> • Connection or port already being used by user. • Communications error: <ul style="list-style-type: none"> ◦ The specified connection has not yet been established. ◦ The specified connection is being terminated. A transfer over this connection is not possible. ◦ The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.
1	80A3	Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communications error: You called the COM_RST = 1 instruction before the send job was complete.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute Only store in load memory.
1	80B3	Inconsistent parameter assignment: Group error for error codes 80A0 to 80A2, 80A4, 80B4 to 80B9.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80B8	The LEN parameter is higher than the highest permitted value.
1	80B9	The ID parameter within the CONNECT parameter is outside the permitted range.
1	80B7	Maximum number of connections reached; no additional connection possible.
1	80B8	The value at the LEN parameter does not match the receive area set at the DATA parameter.
1	80B9	The CONNECT parameter does not point to a data block.
1	80B1	Maximum nesting depth exceeded.
1	80B2	The CONNECT parameter points to a field that does not match the length of the connection description.
1	80B3	<ul style="list-style-type: none"> • All connection resources are in use. • A block with this ID is already being processed in a different priority group.

1	80C4	Temporary communications error: <ul style="list-style-type: none">The connection cannot be established at this time.The interface is receiving new parameters or the connection is being established.The configured connection is being removed by a "TDISCON" instruction.The connection used is being terminated by a call with COM_RST= 1.
1	8722	CONNECT parameter: The source area is invalid. The area does not exist in the DB.
1	873A	CONNECT parameter: Access to the connection description is not possible (for example, DB does not exist).
1	877F	CONNECT parameter: Internal error.
1	8822	DATA parameter: Invalid source area, the area does not exist in the DB.
1	8824	DATA parameter: Area error in the VARIANT pointer.
1	8832	DATA parameter: The DB number is too large.
1	883A	CONNECT parameter: Access to entered connection data not possible (e.g. because the DB does not exist).
1	887F	DATA parameter: Internal error, e.g. illegal VARIANT reference.
1	893A	DATA parameter: Access to entered transmission range not possible (e.g. because the DB does not exist).

5.2 Control Program for the Controller_Conveyor

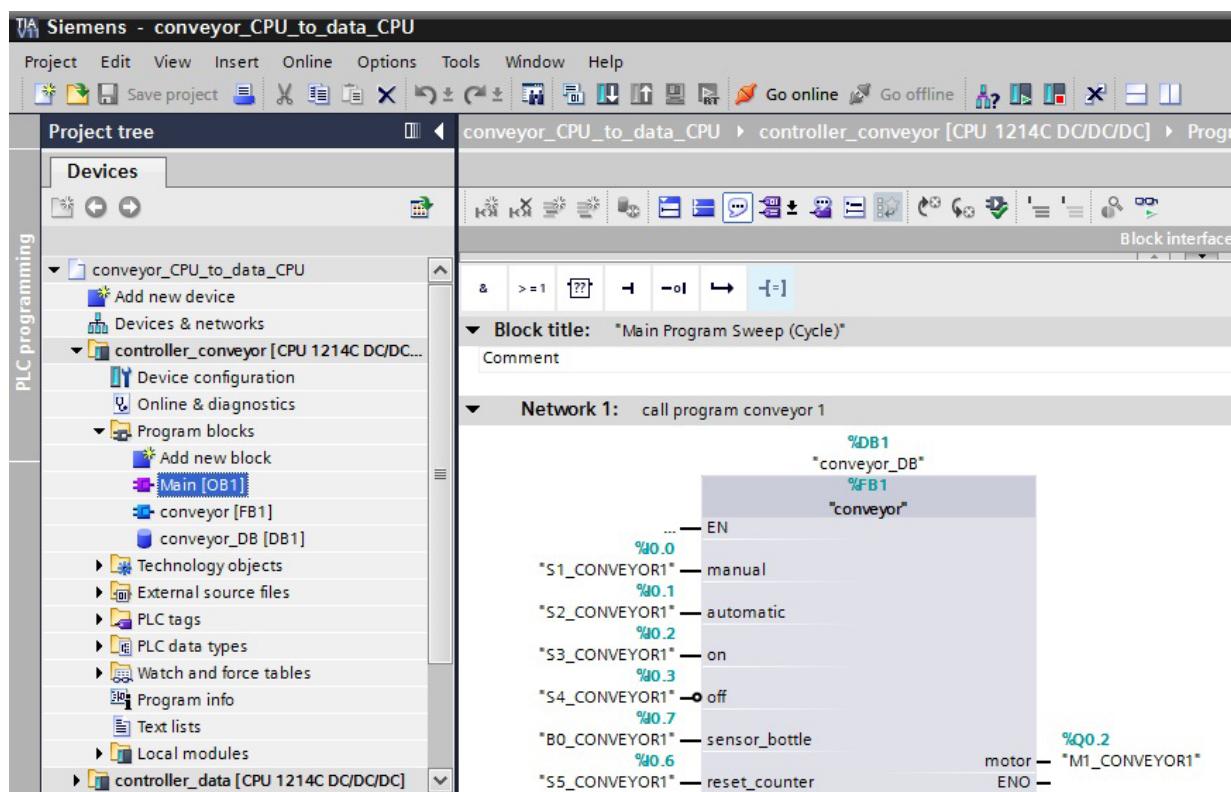
Supplementing the default tag table

Under PLC tags, open the default tag table of the controller_conveyor
Supplement the table.



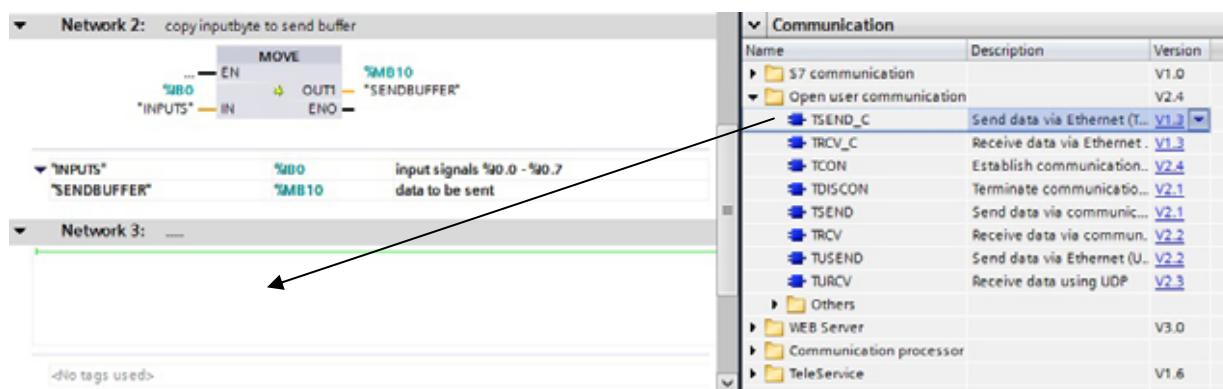
The TSEND_C is called in OB1 of the control program.

Open the Main [OB1] of controller_conveyor

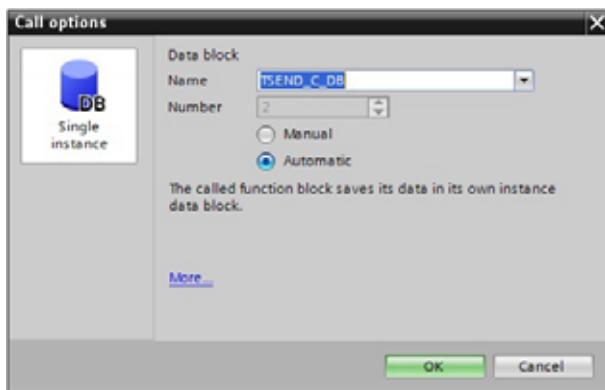


Generate the **MOVE** command in Network 2.

Drag the block "TSEND_C" to Network 3.



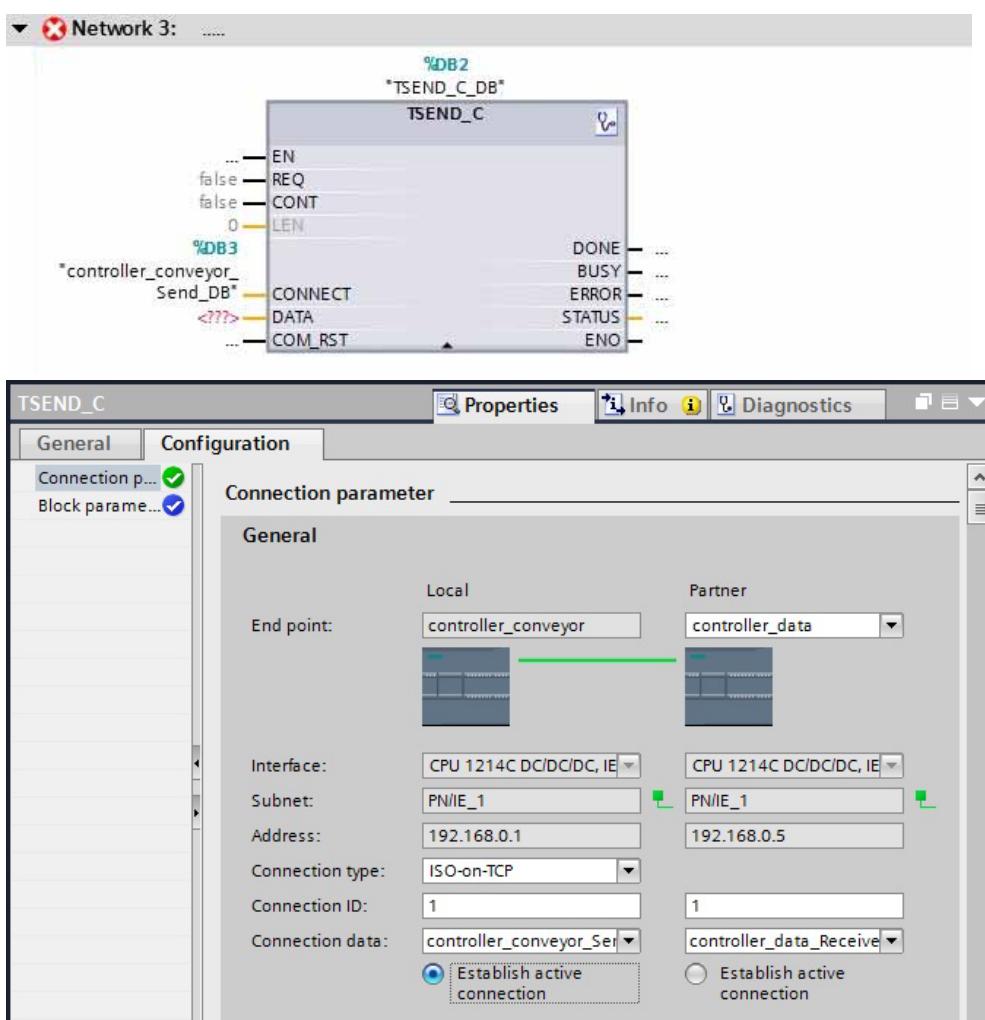
In the window that follows, accept the suggested data block. Click on **Single instance** and then on **"OK"**.



The data block is generated and incorporated automatically.

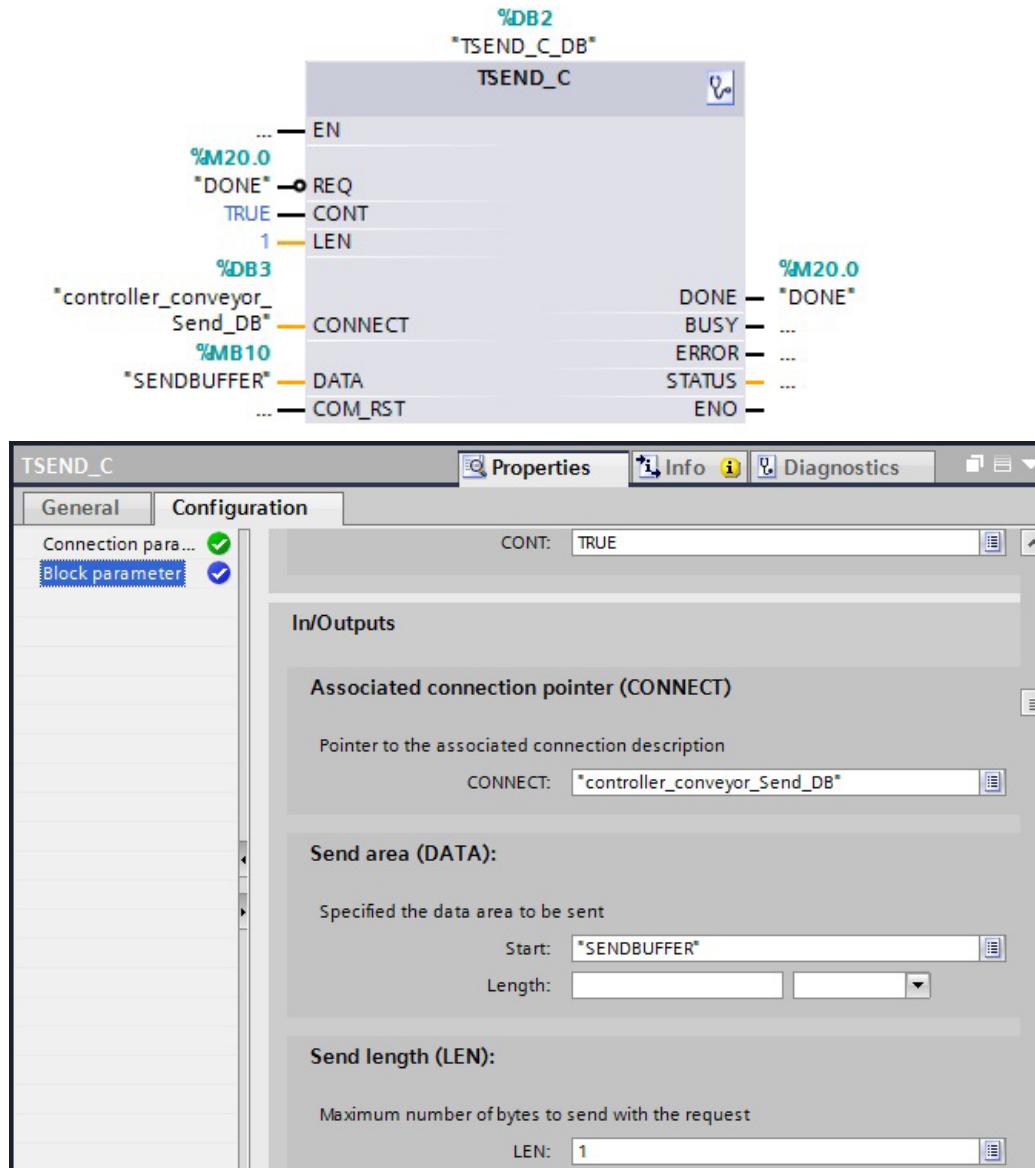
Under Properties, select the connection parameters.

At the connection data, select the partner controller and make a selection in the colored fields. For each controller, a new data block has to be selected for the connection, and the connection ID "1" has to be entered directly on both sides.

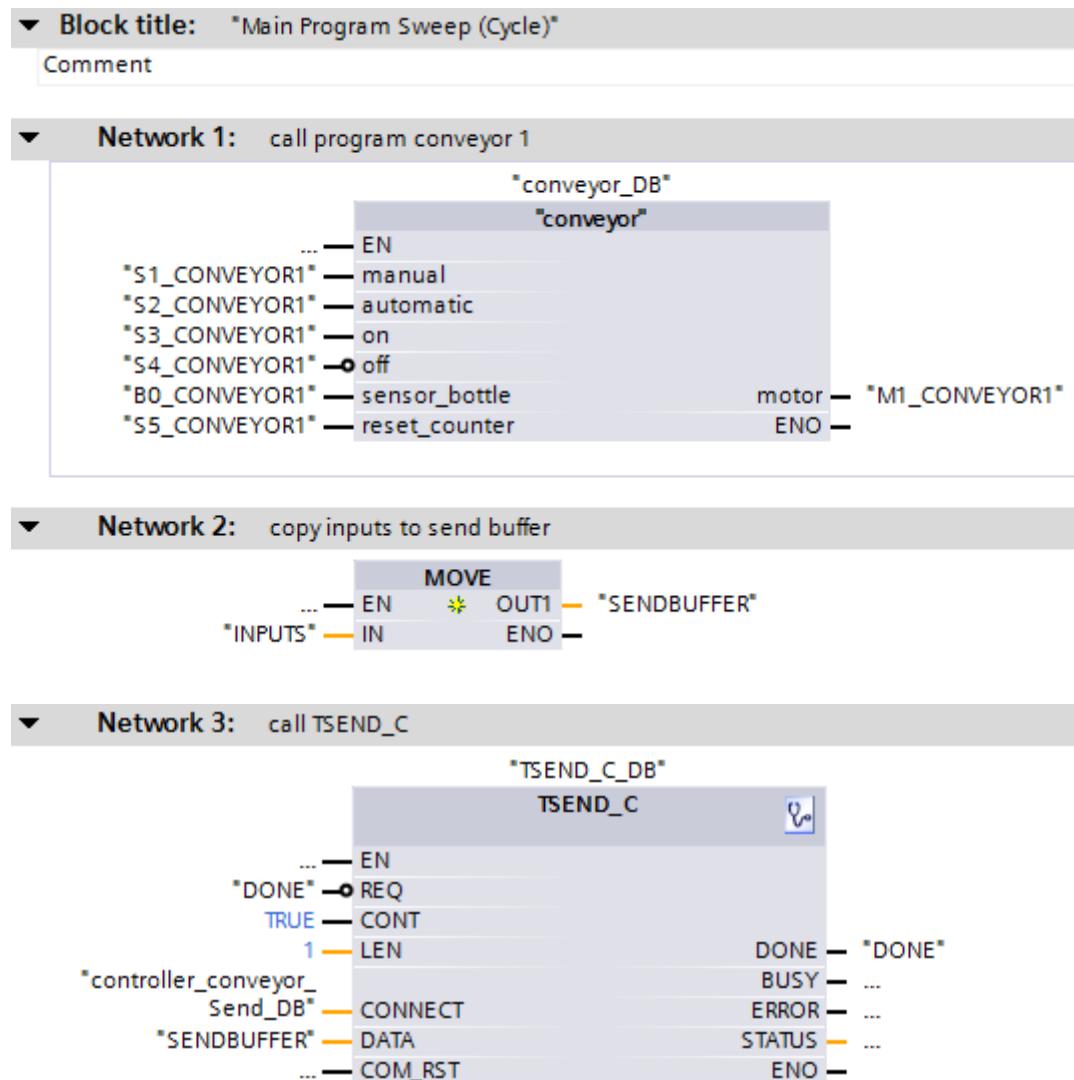


The send request (**REQ**) of the block TSEND_C is to take place cyclically –which explains the linking with the output parameter (**DONE**).

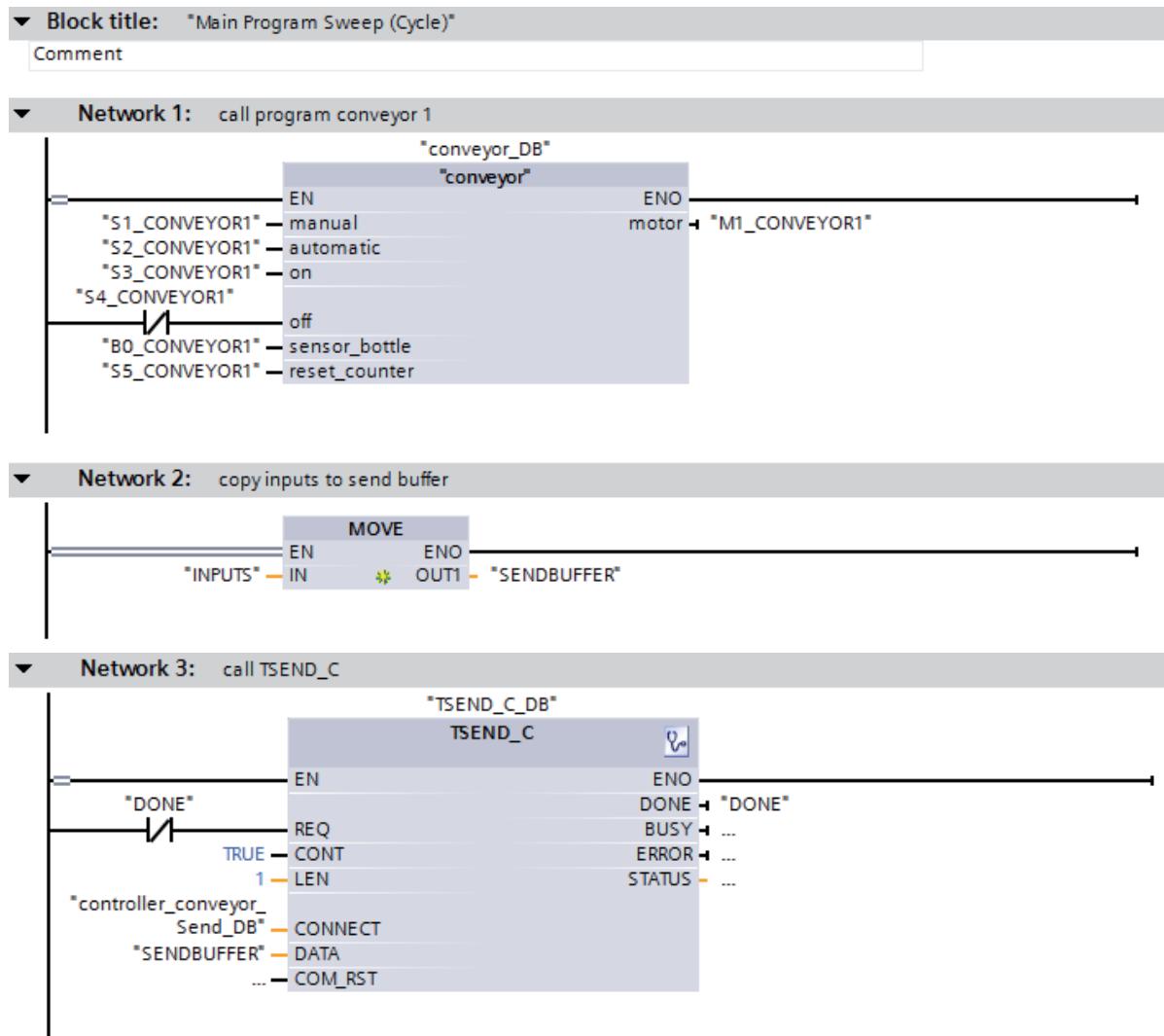
Now, expand the block to its full size and enter the block parameters either at the block or in the property window.



This completes the programming at the controller_conveyor

Program in function block diagram (FBD):

7

Program in ladder diagram (LAD):

Click on **Save project**.

7

In the project navigation window, select “**controller_conveyor**” and click on the button **Download to device** .

5.3 Receive Block TRCV_C

Description

TRCV_C is executed asynchronously and has the following functions:

Setting up and establishing a communication connection:

TRCV_C sets up a TCP or ISO-on-TCP communication connection and establishes it. After the connection is set up and established, the CPU maintains and monitors it automatically.

To set up the communication connection, the connection description specified at the parameter CONNECT is used. To establish the connection, the parameter CONT has to be set to "1". If the connection was established successfully, the parameter DONE is set to "1".

When the CPU enters the STOP mode, an existing connection is cancelled and the connection that was set up is removed. To again set up and establish the connection, TRCV_C has to be executed again. The number of possible communication connections is provided in the technical data of your CPU.

Receiving data over an existing communication connection:

If the parameter EN_R is set to "1", data receipt is activated. The data that is received is entered in the receive area. The length of the receive area is specified depending on the protocol variant used; either with the parameter LEN (if LEN <> 0) or the length indication of the parameter DATA (if LEN = 0).

If data receipt was successful, the signal status at parameter DONE is "1". If errors occur during data transmission, the parameter DONE is reset to "0".

Disconnecting the communication connection:

The communication connection is disconnected when the parameter CONT is set to "0".

When setting the parameter COM_RST, TRCV_C is executed again. The existing communication connection is disconnected and a new connection established. Data received while the connection is re-established may be lost.

Receive Modes of TRCV_C

The table below shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type parameter of the connection description	LEN parameter	RCVD_LEN parameter
TCP (Ad-hoc mode)	The data are immediately available.	B#16#11	65535	1 to 1472
TCP (Data receipt with specified length)	The data are available as soon as the data length specified at the LEN parameter was fully received.	B#16#11	1 to 8192	Identical to the value at the LEN parameter
ISO on TCP (protocol-controlled data transfer)	The data are available as soon as the data length specified at the LEN parameter was fully received.	B#16#12	1 to 8192	Identical to the value at the LEN parameter

TCP (ad hoc mode)

The ad hoc mode exists only for the protocol variant TCP. The ad hoc mode is set by assigning the value 0 to the parameter LEN. The length of the receive area is defined by the pointer at the parameter DATA. The data length actually received at the parameter RCVD_LEN has to be identical with the length that was defined at the parameter DATA. A maximum of 8192 bytes can be received.

TCP (data volume with specified length)

The length of data volume is specified with the value of the parameter LEN. The data specified at the parameter DATA is available at the receive area as soon as the length specified at the parameter LEN was received completely.

ISO on TCP (protocol controlled data transmission)

In the case of the protocol variant ISO on TCP, the data is transmitted protocol-controlled. The receiving area is defined by the parameters LEN and DATA.

Parameters of the TRCV_C

Parameters	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Receive enable
CONT	Input	BOOL	I, Q, M, D, L	Controls the communications connection: <ul style="list-style-type: none"> • 0: Automatically disconnect communications connection after data have been sent • 1: Establish and maintain the communications connection When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the data to be received. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Parameters of communication connections
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: <ul style="list-style-type: none"> • 0: Irrelevant • 1: Complete restart of the instruction causing an existing connection to be terminated
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: <ul style="list-style-type: none"> • 0: Job not yet started or is still executing • 1: Job completed error-free
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: <ul style="list-style-type: none"> • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: <ul style="list-style-type: none"> • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UINT	I, Q, M, D, L	Amount of data actually received in bytes

BUSY, DONE and ERROR

The parameters BUSY, DONE, ERROR and STATUS are used to check the execution status. The parameter BUSY shows the processing status.

The parameter DONE checks whether a request was executed successfully.

The parameter ERROR is set when errors occur while TRCV_C is executed.

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was stopped with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

The error information is read out at the parameter STATUS.

ERROR	STATUS (W#16#...)	Description
0	0000	Job completed error-free
0	7000	No job processing active
0	7001	<ul style="list-style-type: none"> • Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data is being received
0	7003	Connection is being terminated
0	7004	<ul style="list-style-type: none"> • Connection established and monitored • No job processing active
1	8085	<ul style="list-style-type: none"> • The LEN parameter is higher than the highest permitted value. • The value at the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8088	The value at the LEN parameter does not match the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not match the length of the connection description.
1	809B	The ID of the local device (local_device_id) in the connection description does not match the CPU.
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.

1	80A1	<ul style="list-style-type: none"> • Connection or port already being used by user. • Communications error: <ul style="list-style-type: none"> ◦ The specified connection has not yet been established. ◦ The specified connection is being terminated. Transfer over this connection is not possible. ◦ The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.
1	80A3	<ul style="list-style-type: none"> • Attempt being made to re-establish an existing connection. • Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communications error: You called the COM_RST = 1 instruction before the send job was complete.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute Only store in load memory.
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80C3	<ul style="list-style-type: none"> • All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	<p>Temporary communications error:</p> <ul style="list-style-type: none"> • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST= 1.
1	8722	Error in the CONNECT parameter: Invalid source area (area not declared in data block).
1	873A	Error in the CONNECT parameter: Access to connection description is not possible (no access to data block).
1	877F	Error in the CONNECT parameter: Internal fault
1	8922	DATA parameter: Invalid target area, the area does not exist in the DB.
1	8924	DATA parameter: Area error in the VARIANT pointer.
1	8932	DATA parameter: The DB number is too large.
1	893A	CONNECT parameter: Access to entered connection data not possible (e.g. because the DB does not exist).
1	897F	DATA parameter: Internal error, e.g. illegal VARIANT reference.
1	8A3A	DATA parameter: No access to the data area because, for example, the data block does not exist.

5.4 Control Program for controller_data

Supplementing the default tag table

Under PLC tags, open the default tag table of the controller_data.

Supplement the table.

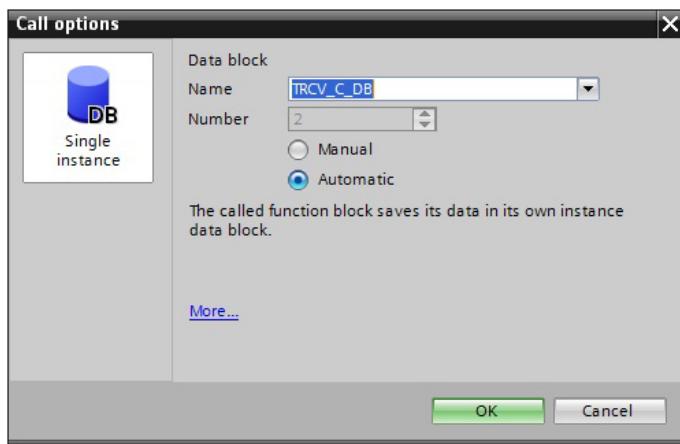
Name	Data type	Address	Retain	Visible	Access	Comment
1 RECEIVEBUFFER	Byte	%MB12	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	receive data
2 OUTPUTS	Byte	%Q80	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	outputs %Q0.0 - %Q0.7
3 <Add new...>						

In OB1 of the control system program, the receive block TRCV_C is called.

Open the Main [OB1] of controller_data
Drag the block "TRCV_C" to Network 1.



In the following window, accept the suggested data block. First click on **Single instance** and then on **"OK"**.



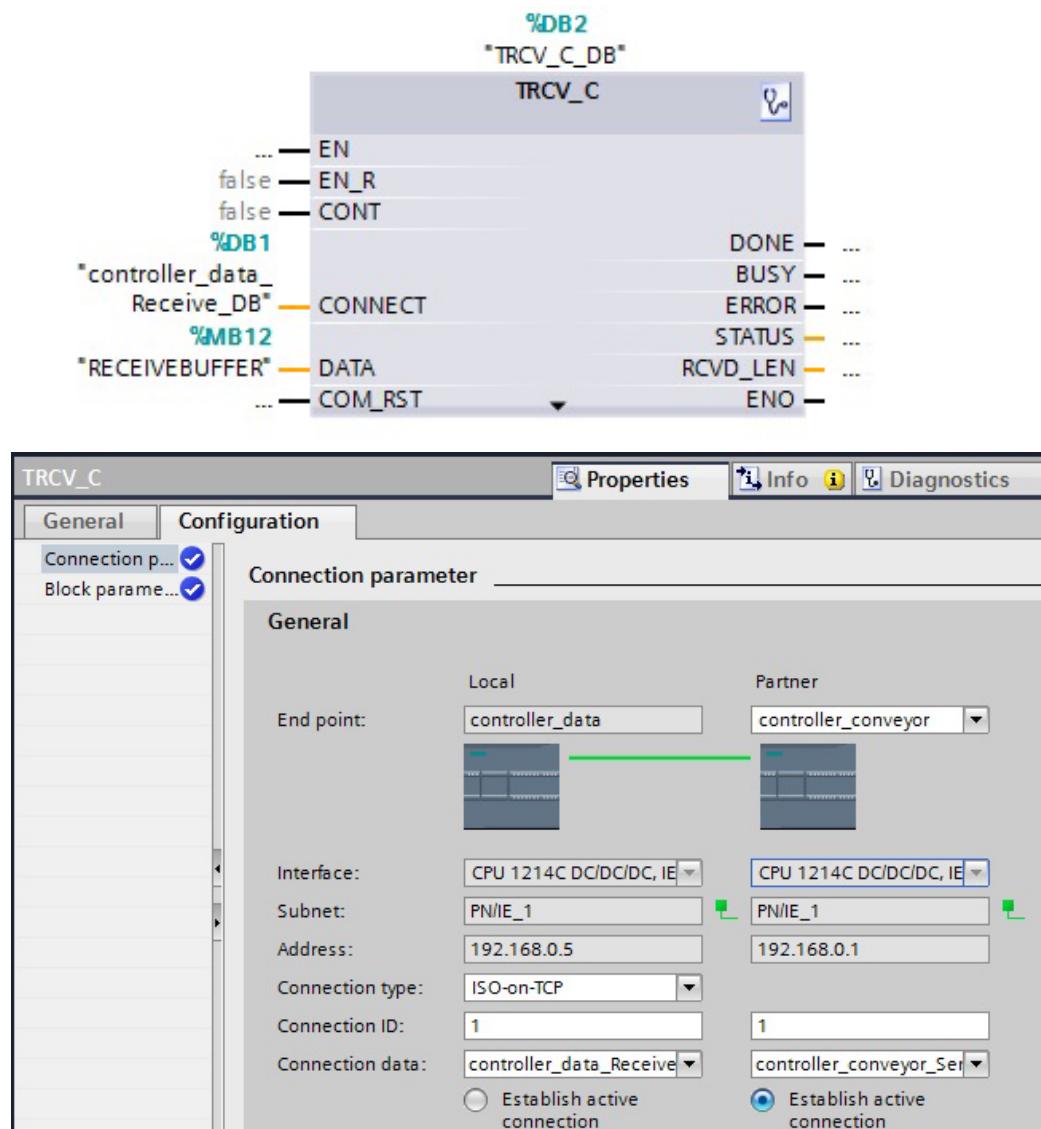
The data block is generated and incorporated automatically.

Under Properties, select the connection partners.

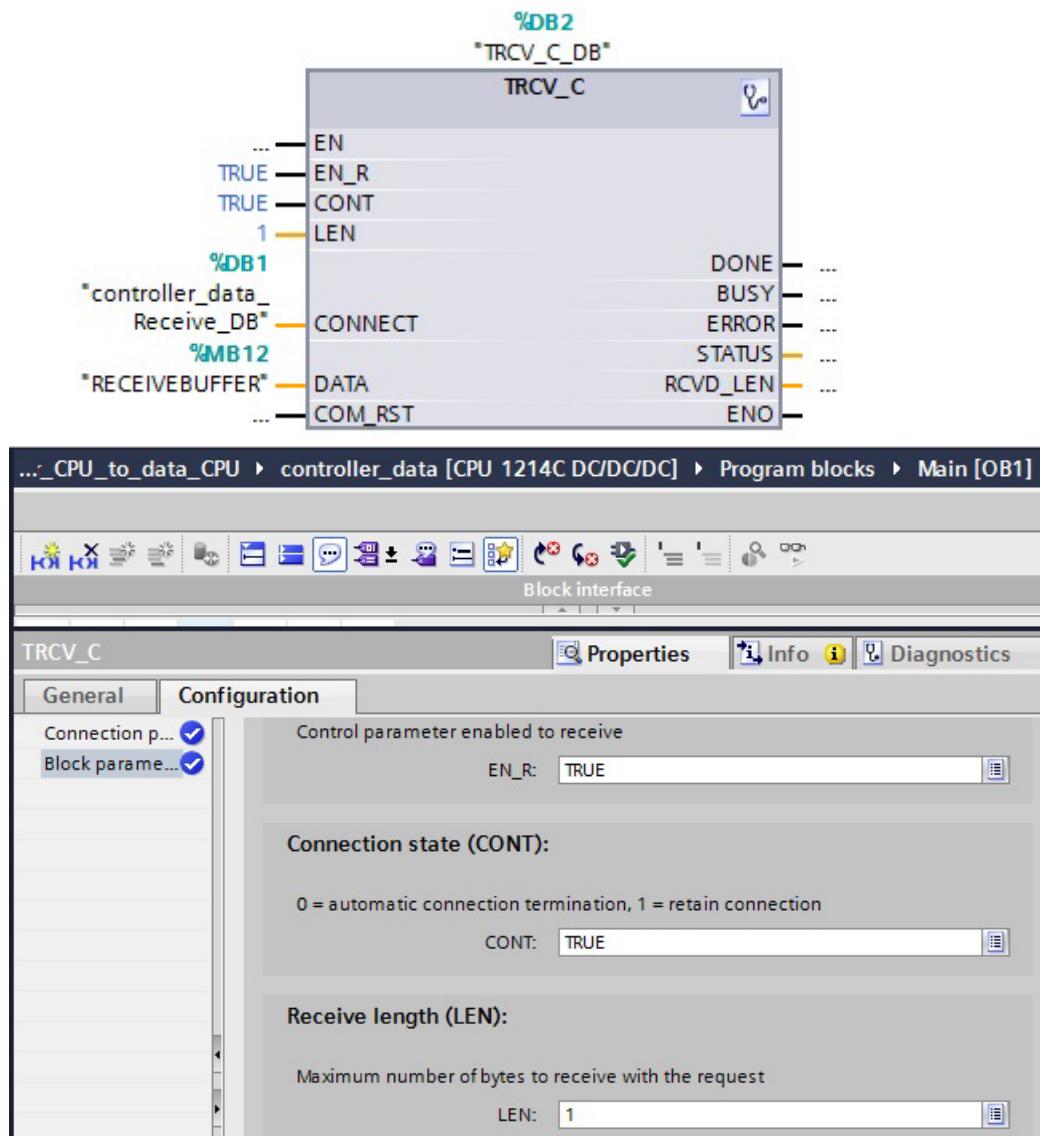
First, at the connection data of the local controller, select the existing data block

"**controller_data_connection_DB**", otherwise, a new data block is generated.

Then, select the partner controller and the associated data block.



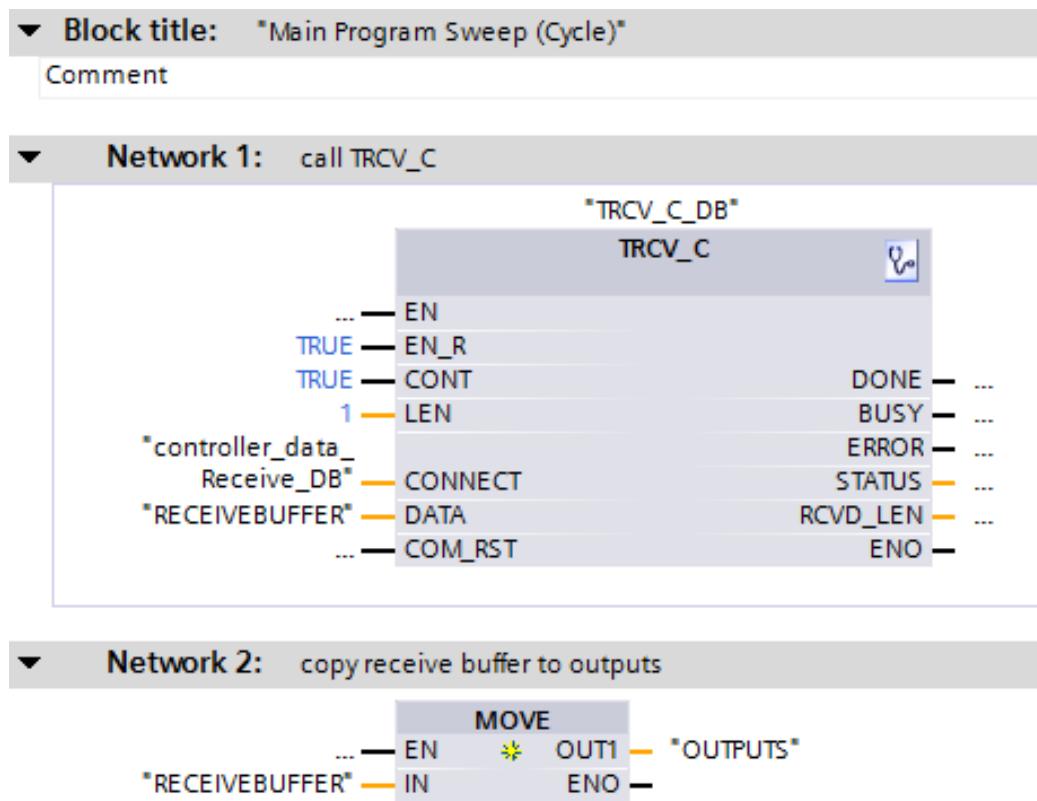
Expand the block to its full size and enter the block parameters either at the block or at the property window.



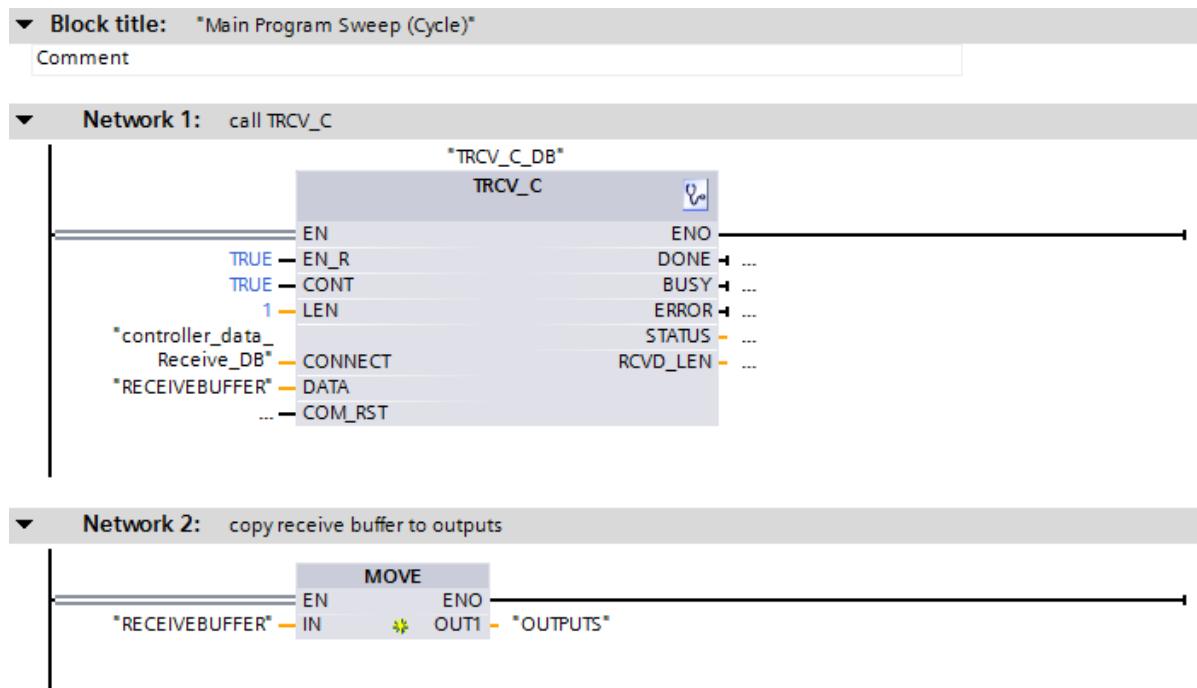
Generate the **MOVE** command in Network 2.

This completes the programming at the controller_data.

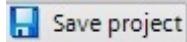
Program in function block diagram (FBD):



Program in ladder diagram (LAD):



1. Click on “Save project”.



2. In the project navigation window, select “controller_data” and click on the button **Downloading to device**.

After successful loading to the CPU, the program can be tested.

The input signals I0.0 to I0.7 of the controller_conveyor are displayed at outputs O0.0 to O0.7 of the controller_data.

Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Module 010-080



TIA Portal Module 010-010

Startup Programming of the
SIMATIC S7-1200

TIA Portal Module 010-020

Block Types for the SIMATIC S7-1200

TIA Portal Module 010-030

IEC Timers and IEC Counters at the
SIMATIC S7-1200

TIA Portal Module 010-040

Diagnosis and Error Search at the
SIMATIC S7-1200

TIA Portal Module 010-050

Analog Value Processing with
SIMATIC S7-1200

TIA Portal Module 010-060

Control Engineering with
SIMATIC S7-1200

TIA Portal Module 010-070

Communication between two
SIMATIC S7-1200

TIA Portal Module 010-080

HMI Panel KTP600 with SIMATIC
S7-1200

TIA Portal Module 010-090

Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

Matching SCE training packages for these documents

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
Order No.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
Order No.: 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
Order No.: 6ES7822-0AA01-4YE0

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

Continued Training

For regional Siemens SCE continued training, please contact your regional SCE contact person
siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding Usage

The training curriculum 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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by 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.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the documents being used commercially.

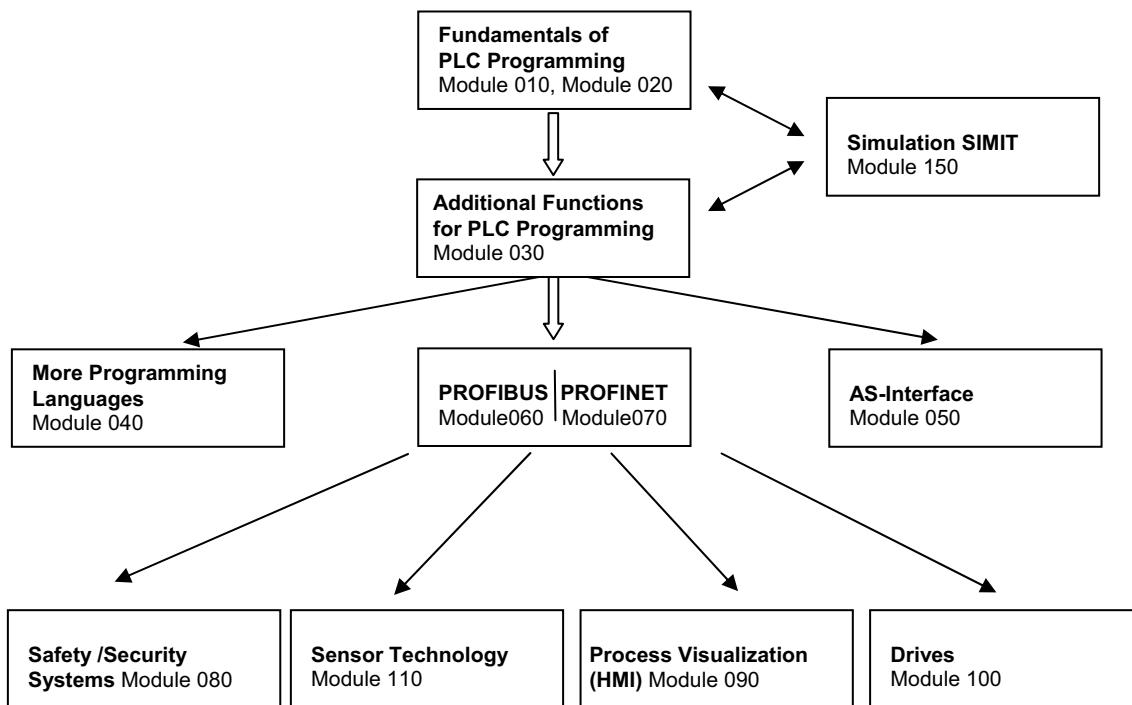
We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this document.

Page

1.	Preface	4
2.	Instructions for Programming the SIMATIC S7-1200	6
2.1	Automation System SIMATIC S7-1200.....	6
2.2	Programming Software STEP 7 Professional V11 (TIA Portal V11)	6
2.3	Operator Control and Monitoring with WinCC	7
3.	Conveyor Control with Counter and Multi-Instance	8
3.1	Task.....	8
3.2	Configuration.....	9
4.	Inserting Panel KTP600 PN in the Project of Conveyor Control.....	9
4.1	Load and Re-Store Existing Project.....	10
5.	WinCC Operator Interface	22
5.1	Project Navigation.....	23
5.2	Menu Bar and Buttons	23
5.3	Work Area	24
5.4	Tools.....	24
5.5	Property Window.....	25
5.6	Detail View	25
6.	Operating Screens and Connections	26
6.1	Root Screen or Start Screen.....	26
6.2	Connections to S7 Controllers	27
6.3	Assigning the IP Address	28
7.	Configuring the Root Screen.....	29
7.1	Function Key F6	30
7.2	Configuring the Automatic and Manual Buttons	31
7.3	Changes in the Step7 Program.....	35
7.4	Setting the PG/PC Interface for Runtime Simulation	36
7.5	Starting the Configuration in Runtime	37
7.6	Loading the Configuration to the Panel and Testing It.....	38
7.7	Start and Stop Button	39
7.8	Adding Graphics from the Graphics Folder	42
7.9	Control Program for Simulating Bottle Movement	44
7.10	Activating the Clock Memory and Assigning the MB100	46
7.11	Calling FB2 (Simulation) in OB1	47
7.12	Configuring the Bottle Movement.....	49
7.13	Resetting the Bottle Counter.....	53
7.14	Drawing the Beer Case	55
7.15	Drawing the Bottles in the Case.....	60

1. Preface

Regarding its content, module 010-080 is part of the training unit '**Basics of PLC Programming**' and demonstrates how to configure WinCC visualization between the panel KTP600 and the SIMATIC S7-1200 controller.



Training Objective

In this module 010-080 the reader learns how to configure process visualization of a panel by means of WinCC, in connection with the SIMATIC S7-1200.

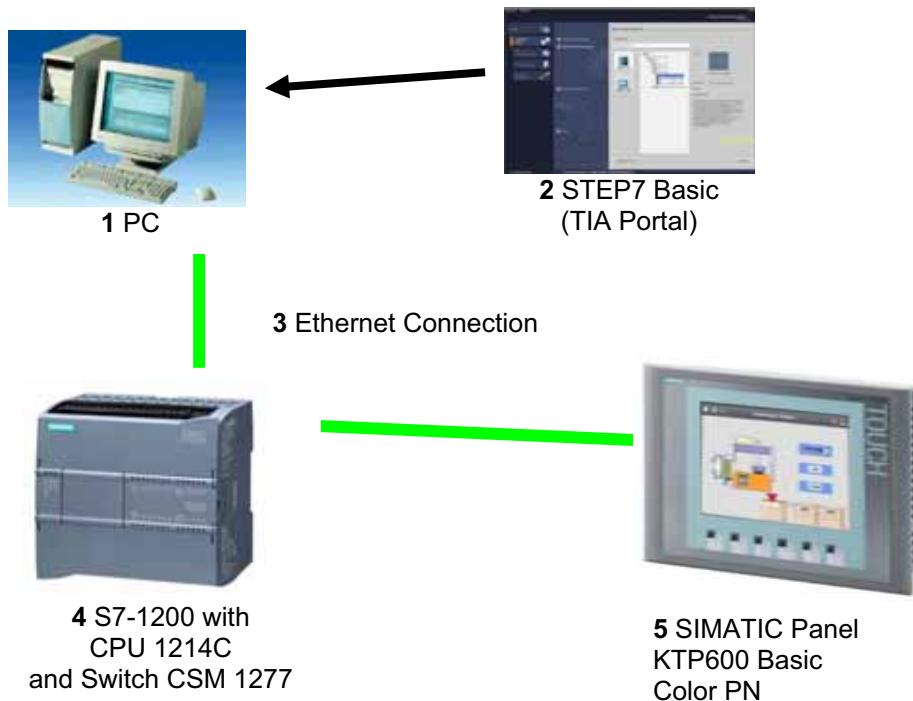
Prerequisites

To successfully work through this module 010-080, the following knowledge is assumed:

- How to operate Windows
- Basics of PLC programming with the TIA Portal
(for example, Module 010-010 'Startup' Programming of the SIMATIC S7-1200 with TIA- Portal V11)
- Blocks for the SIMATIC S7-1200
(for example, Module 010-020 – Block Types for the SIMATIC S7-1200)
- Timer and Counter Blocks for the SIMATIC S7-1200
(for example, Module 010-030 – Multi-Instances at the SIMATIC S7-1200)

Hardware and Software required

- 1** PC Pentium 4, 1.7 GHz, 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB
Operating system Windows XP (Home SP3, Professional SP3)/Windows Vista (Home Premium SP1, Business SP1, Ultimate SP1)
- 2** Software STEP7 Basic V11 SP2 (Totally Integrated Automation (TIA) Portal V11)
- 3** Ethernet connection between PC and CPU 1214C and panel
- 4** PLC SIMATIC S7-1200; for example, CPU 1214C und switch CSM 1277.
The inputs have to be brought out to the panel.
- 5** SIMATIC Panel KTP600 Basic Color PN



2. Instructions for Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The automation system SIMATIC S7-1200 is a modular mini-controller system for the lower and medium performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU and input/output modules for digital and analog signals. .

If needed, communication processors and function modules are added for special tasks such as step motor control.

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process; the IO modules are polled in the S7 program by means of the input addresses (%I) and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7.

2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 (TIA Portal V11) is the programming tool for the following automation systems:

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be used to automate a plant:

- Configuring and parameterizing the hardware
- Specifying communication
- Programming
- Test, commissioning and service with the operational/diagnostic functions
- Documentation
- Generating visualizations for the SIMATIC basic panels with the integrated WinCC Basic.
- With additional WinCC packages, visualization solutions for PCs and other panels can be created

All functions are supported with detailed online help.

2.3 Operator Control and Monitoring with WinCC

System Description

Since processes are becoming more and more multi-layered and the demands regarding the functionality of machines and plants are rising, the operator needs a powerful tool to control and monitor production systems. A HMI system (Human Machine Interface) represents the interface between the human being (operator) and the process (machine/system). The controller actually controls the process. That is, there is an interface between the operator and WinCC (at the operator panel) and an interface between WinCC and the controller. WinCC is the software we use to handle all necessary configuring tasks. WinCC Runtime is the software for process visualization. In runtime, the project is executed in the process mode.

WinCC handles the following tasks:

- **Representing the process**

The process is mapped to the operator panel. If, for example, a state changes in the process, the display is updated on the operator panel.

- **Operating the process**

The operator can operate the process by means of the graphic operator interface. For example, the operator can enter a setpoint for the controller, or start a motor. .

- **Reading out alarms**

If critical process states occur in the process, an alarm is triggered automatically; for example, if a specified limit is exceeded,

- **Archiving process values and alarms**

The HMI system is able to archive alarms and process values. Thus, the process can be documented and older production data can later be accessed.

- **Documenting process values and alarms**

Alarms and process values can be read out by the HMI system as a log. After a shift, for example, you can have the production data read out.

- **Managing process and machine parameters**

The HMI system is able to store parameters for processes and machines in recipes. These parameters can, for example, be transferred from the operator panel to the controller in one step in order to change production to another product type.

3. Conveyor Control with Counter and Multi-Instance

For our process visualization with WinCC, the example for conveyor control will be expanded with a counter and a multi-instance.

With the conveyor, 20 bottles are always to be transported in a case. When the case is full, the conveyor belt is stopped, and the case has to be exchanged. .

With button 'S1' the operating mode 'Manual' and with button 'S2' the operating mode 'Automatic' can be selected.

In the operating mode 'Manual', the motor is switched on as long as button 'S3' is operated; button 'S4' must not be operated.

In the 'Automatic' mode, the conveyor motor is switched on with button 'S3' and switched off with button 'S4' (NC).

In addition, there is a sensor 'B0' that counts the bottles into the case. After 20 bottles are counted, the conveyor is stopped...

When a new case is put in place, is hast o be confirmed with 'S5'.

Assignment list

Address	Symbol	Comment
%I 0.0	S1	Button manual mode S1 NO
%I 0.1	S2	Button automatic mode S2 NO
%I 0.2	S3	On button S3 NO
%I 0.3	S4	Off button S4 NC
%I 0.6	S5	Button S5 NO reset counter/new case
%I 0.7	B0	Sensor B0 NO bottle counter
%O 0.2	M01	conveyor motor M01

8

3.1 Task

The conveyor control is to be operated and monitored by means of the panel.

Using the panel, the following requirements are to be met:

The operating mode is switched by means of the panel, and the respective operating mode is to be displayed on the panel.

Starting and stopping the conveyor motor is controlled from the panel.

Case replacement is acknowledged on the panel.

Transport of the bottles and filling the case is to be shown graphically.

3.2 Configuration

With the configuring software STEP7 V11, process visualization for conveyor control is set up, using the integrated WinCC version. The process values are represented with pictures and picture objects. With operating elements, default values are transferred to the controller. By means of the controller, **tags** handle the communication between the operator panel and the machine or the process. The value of a tag is written to a memory area (address) in the controller where it is read by the operator panel.

**The process visualization is stored and loaded to the panel KTP600 Basic color PN.
After the panel is powered up, the conveyor control can be monitored and operated.**

4. Inserting Panel KTP600 PN in the Project of Conveyor Control

For project management and programming, the software '**Totally Integrated Automation Portal V11**' is used.

Here, under a uniform interface, components such as controller, visualization and networking for the automation solution are set up, parameterized and programmed.
Online tools are available for error diagnosis.

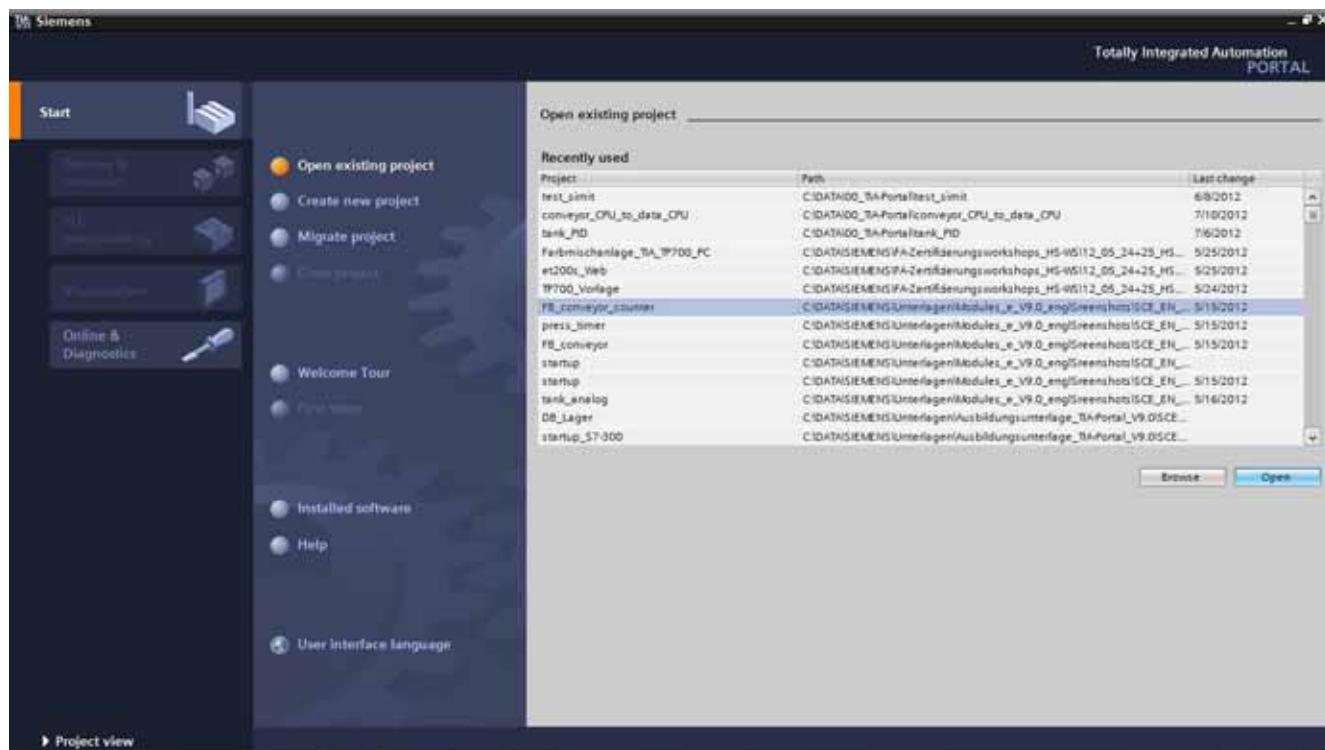
In the steps below, the following is done: for the SIMATIC S7-1200 a project is opened, stored under a different name and adapted to the new requirements.

The central tool is the '**TIA Portal V11**'. It is called here with a double click.

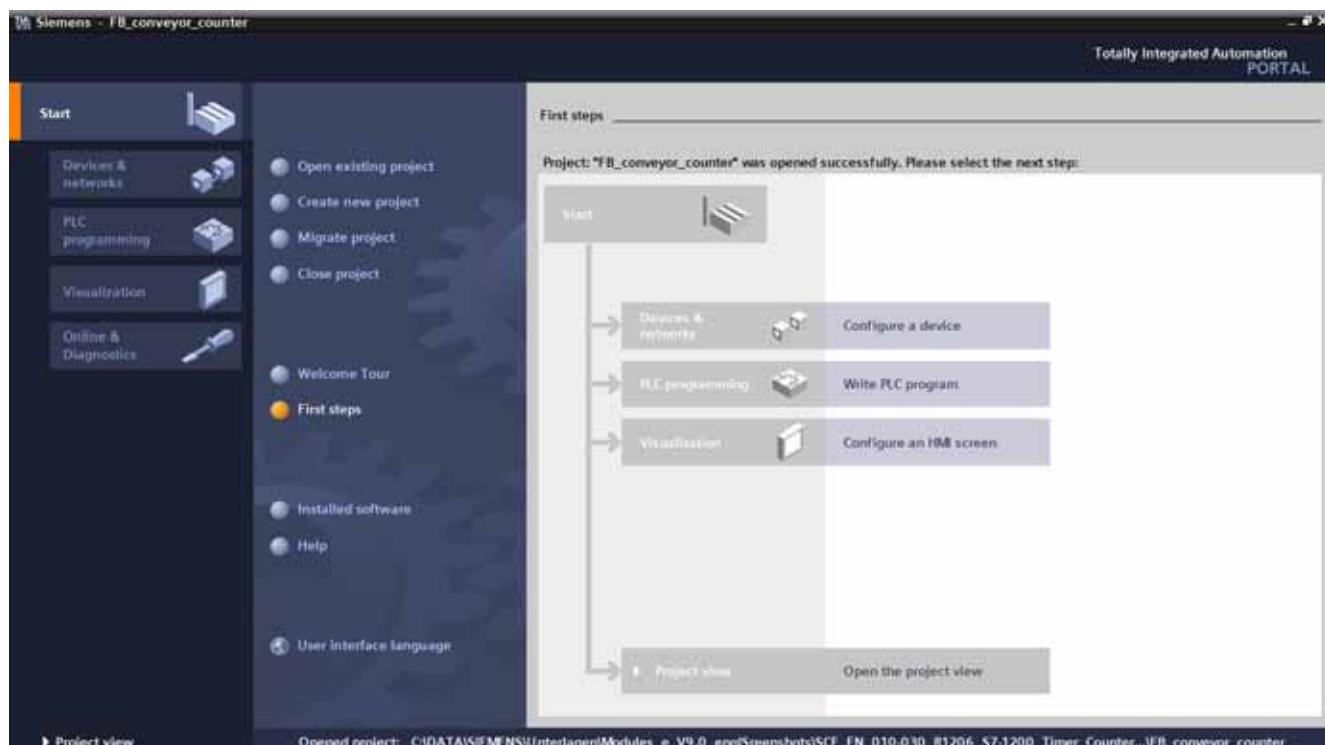


4.1 Load and Re-Store Existing Project

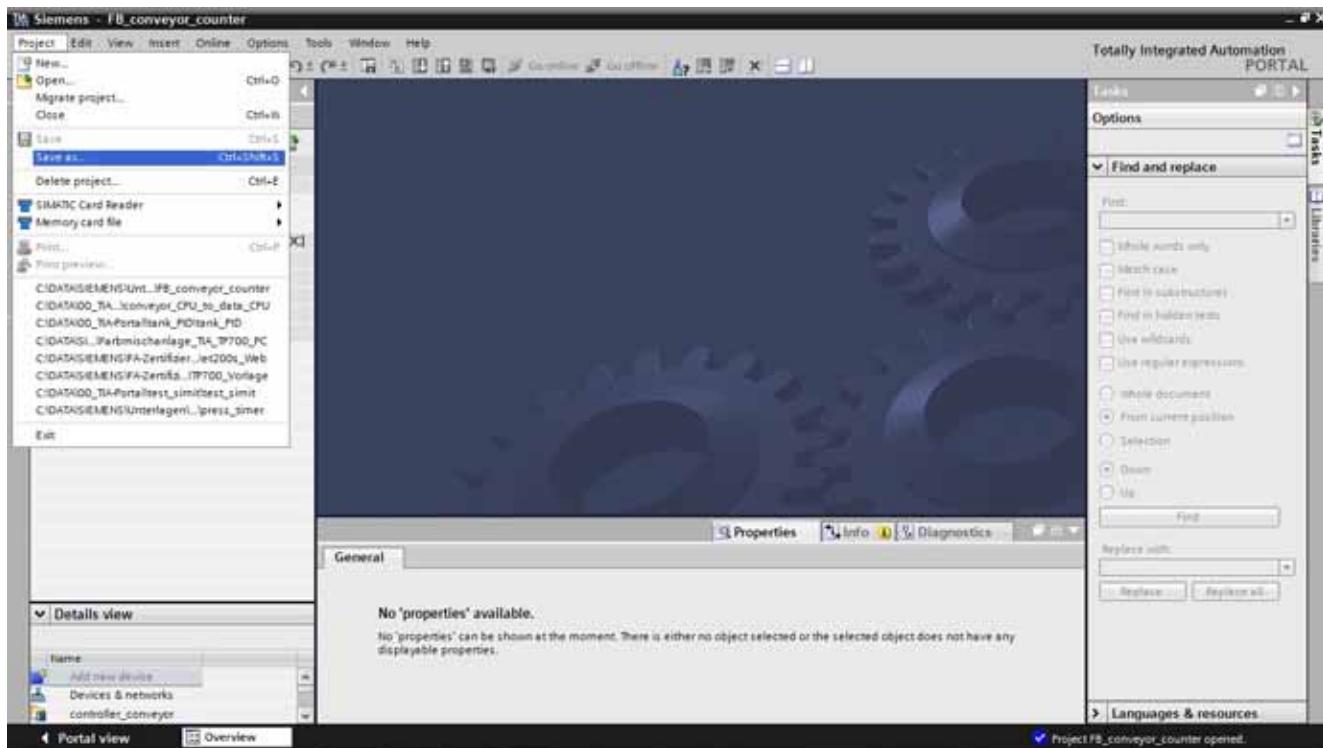
The project "**FB_conveyor_counter**" from Module 010-030 will now be opened as the model for this program.



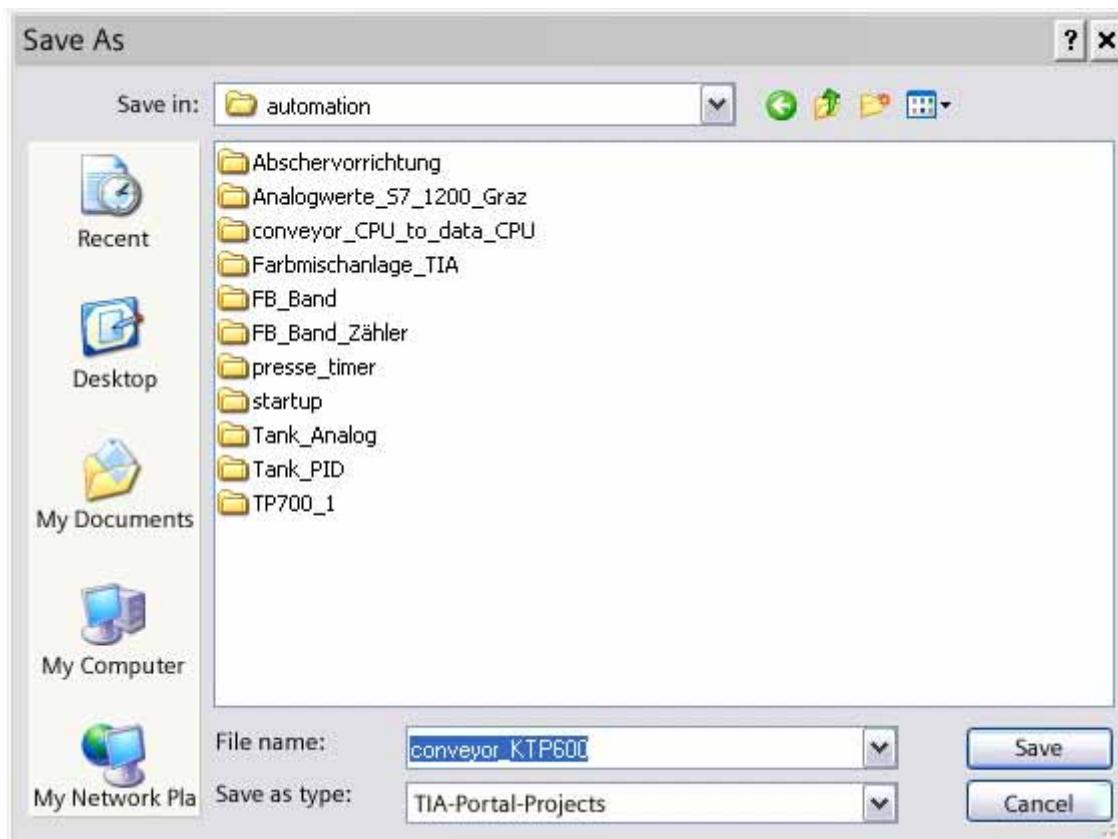
Now , 'First steps' is offered for configuring.
Click on 'Open the project view'.



First, we are going to store the project under another name.
In the menu **Project** click on “**Save As**“.



Now, 'Save' the project under the new name '**conveyor_KTP600**'.

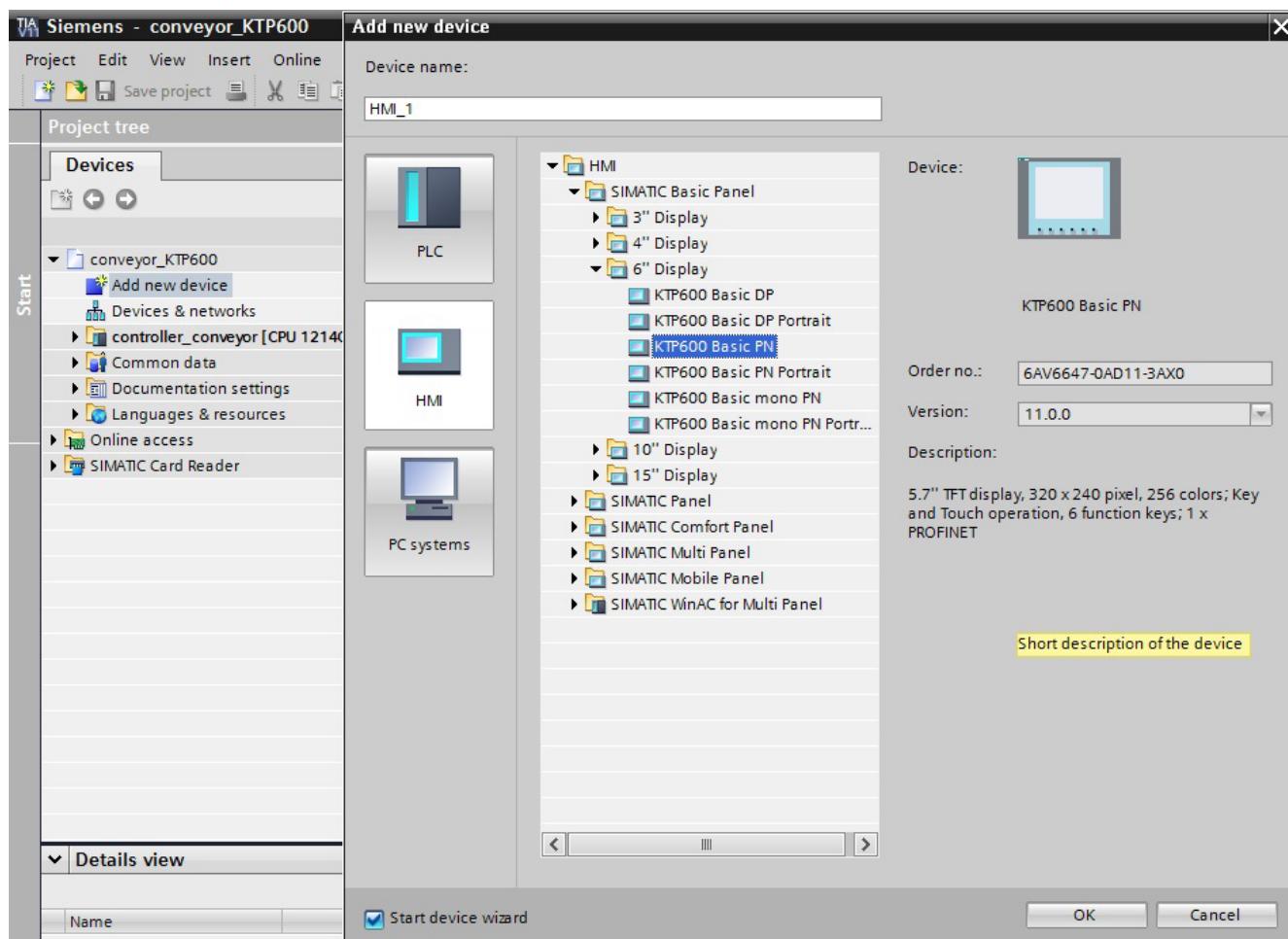


To set up a new project, select the list box by double clicking on
'Add new device'.

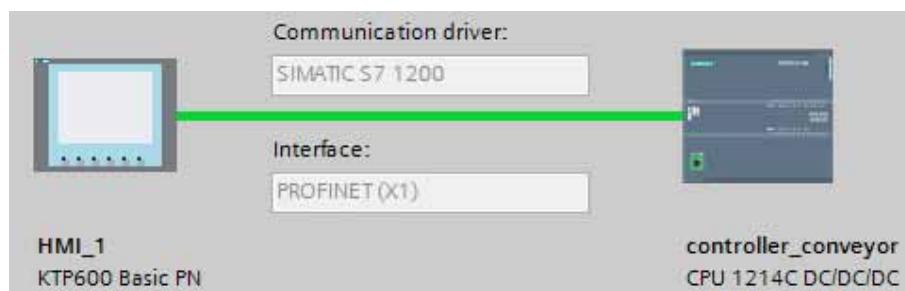
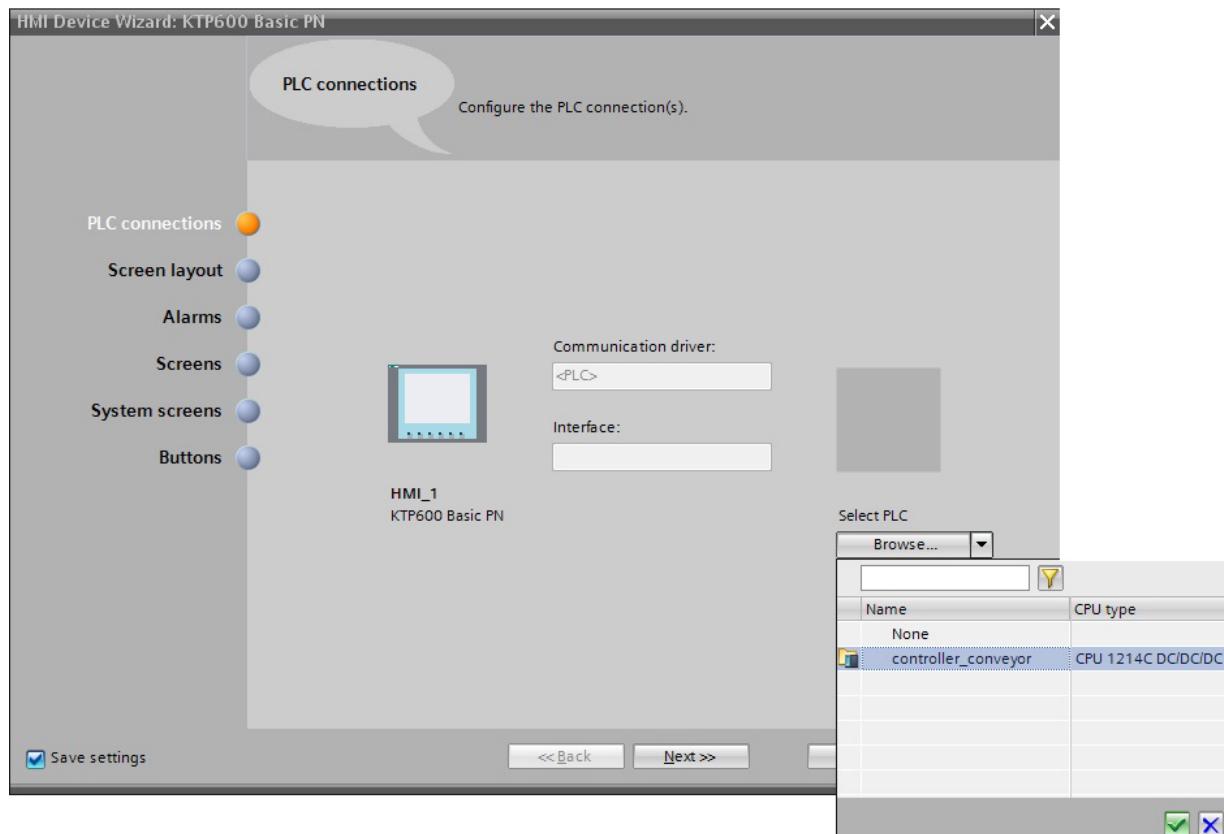
Under SIMATIC HMI, select the 6" display panel "**KTP600 Basic PN**".

Set the check mark at "**Start device wizard**".

Click "**OK**".



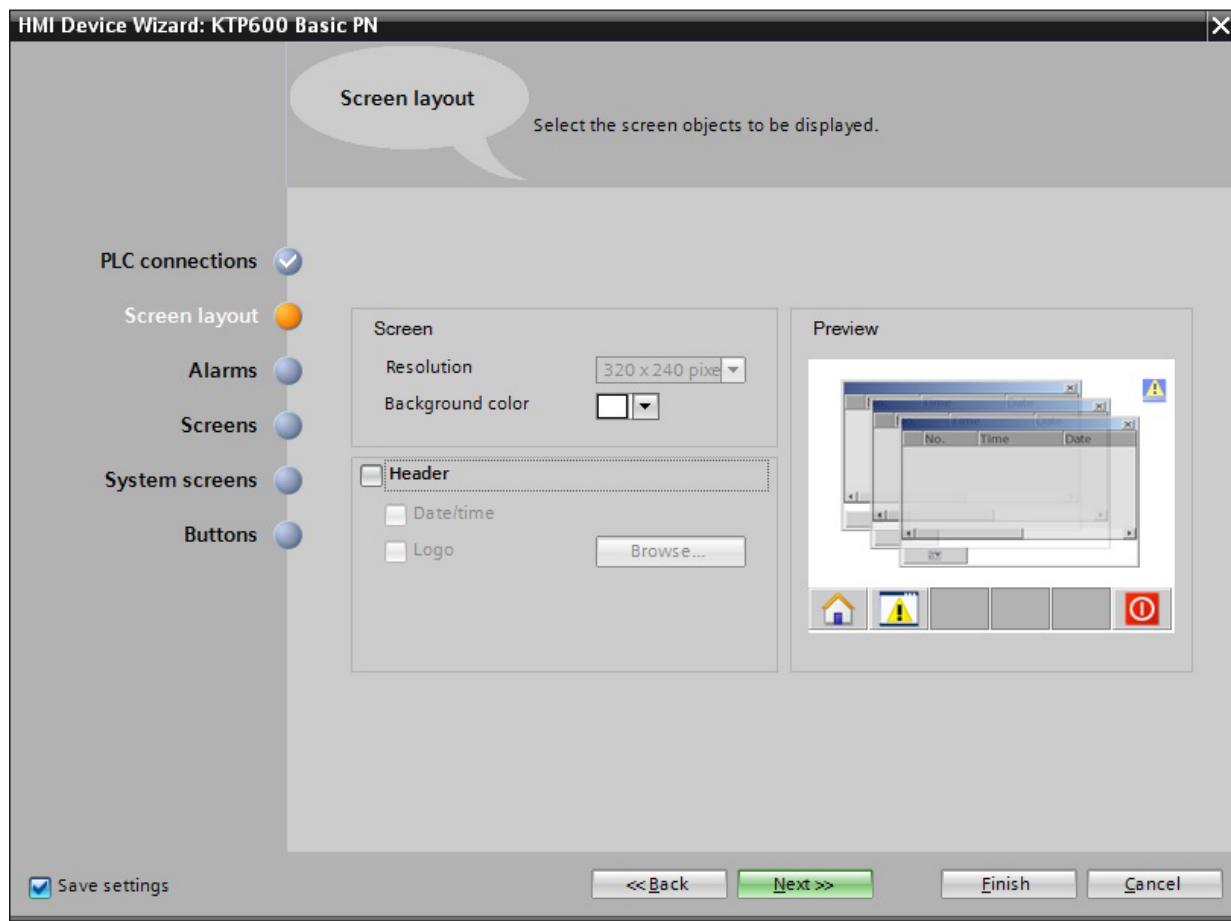
Under Select PLC, first select "**controller_conveyor**".



8

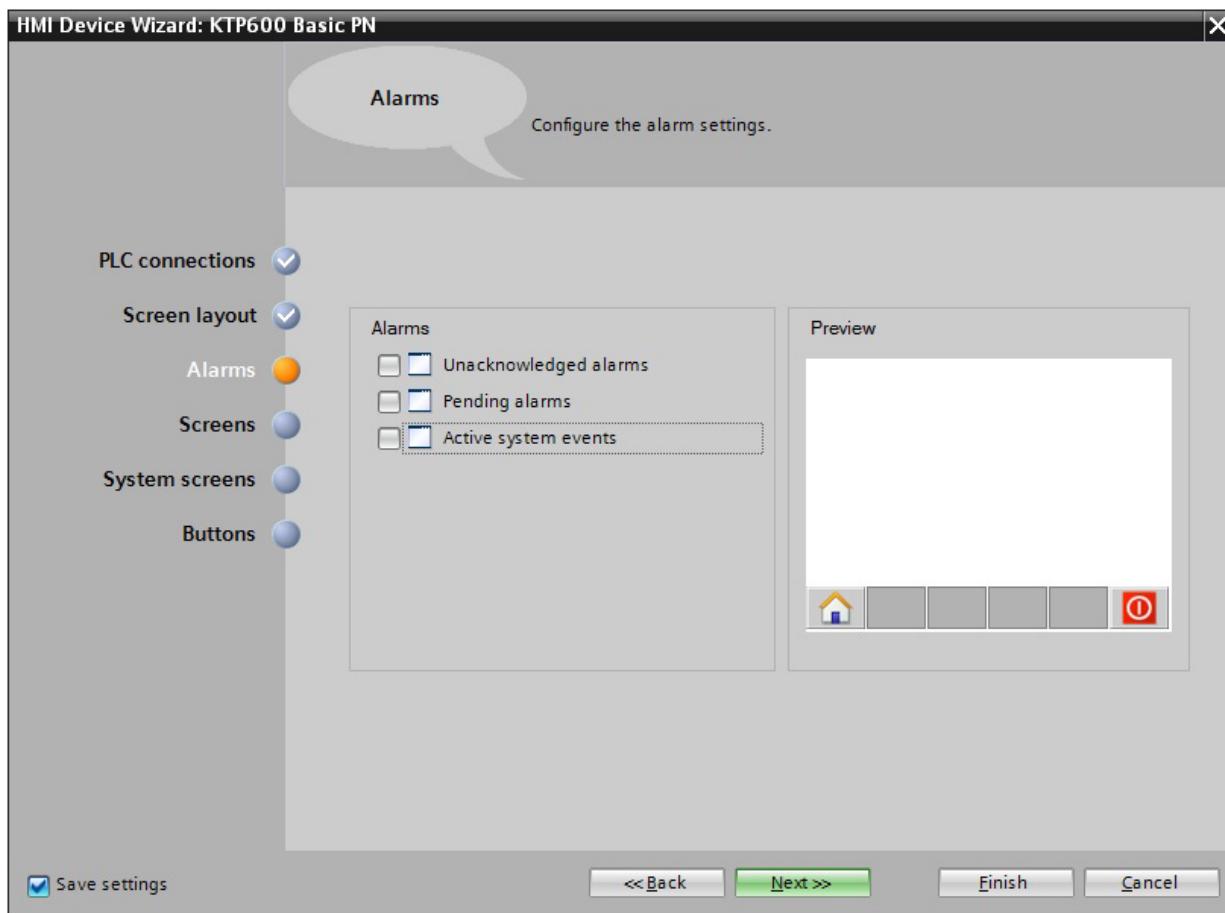
Then, click on "**Next**".

Under Screen layout, change the background color to “White”.
And remove the check mark at “Header”.



8 Then click on “Next“.

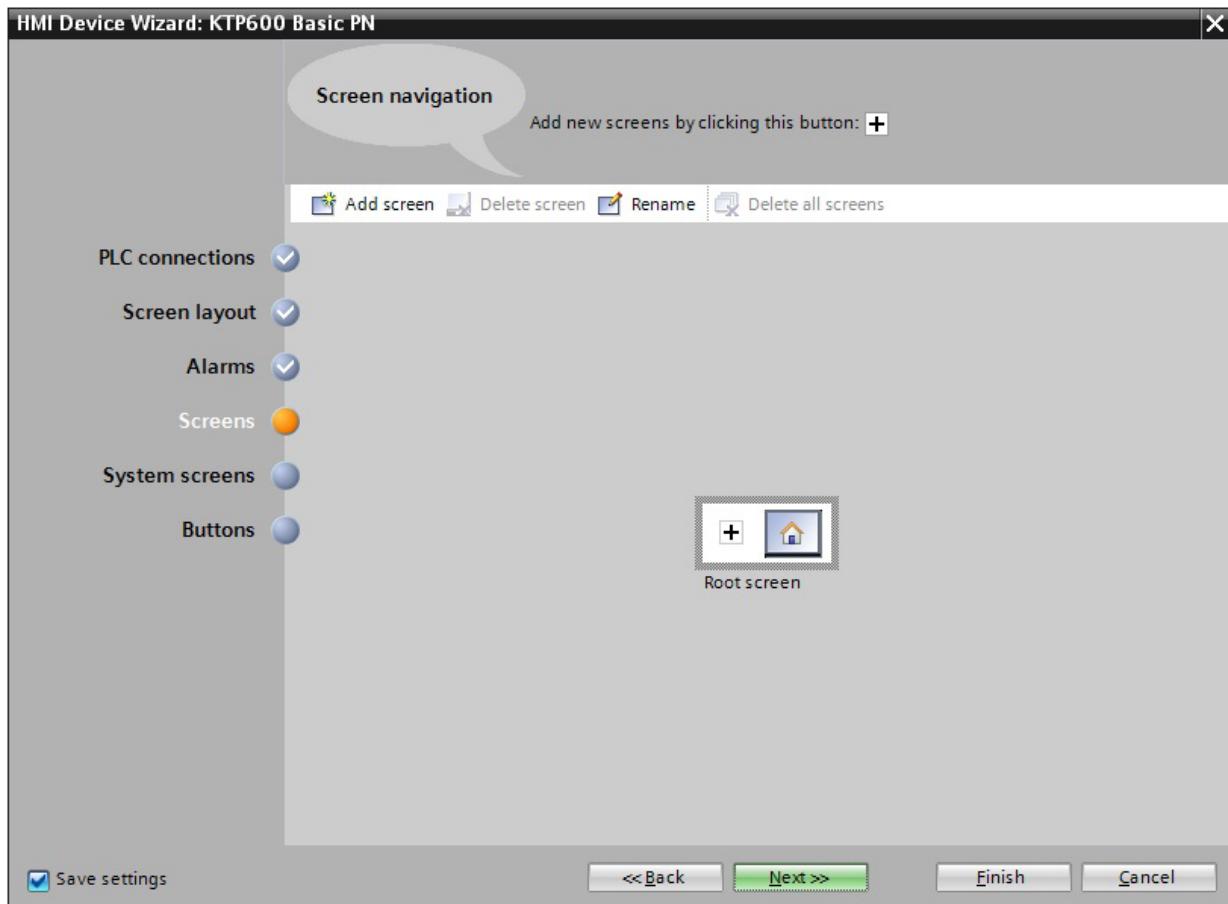
Remove all check marks at Alarms.



Then click on “**Next**“.

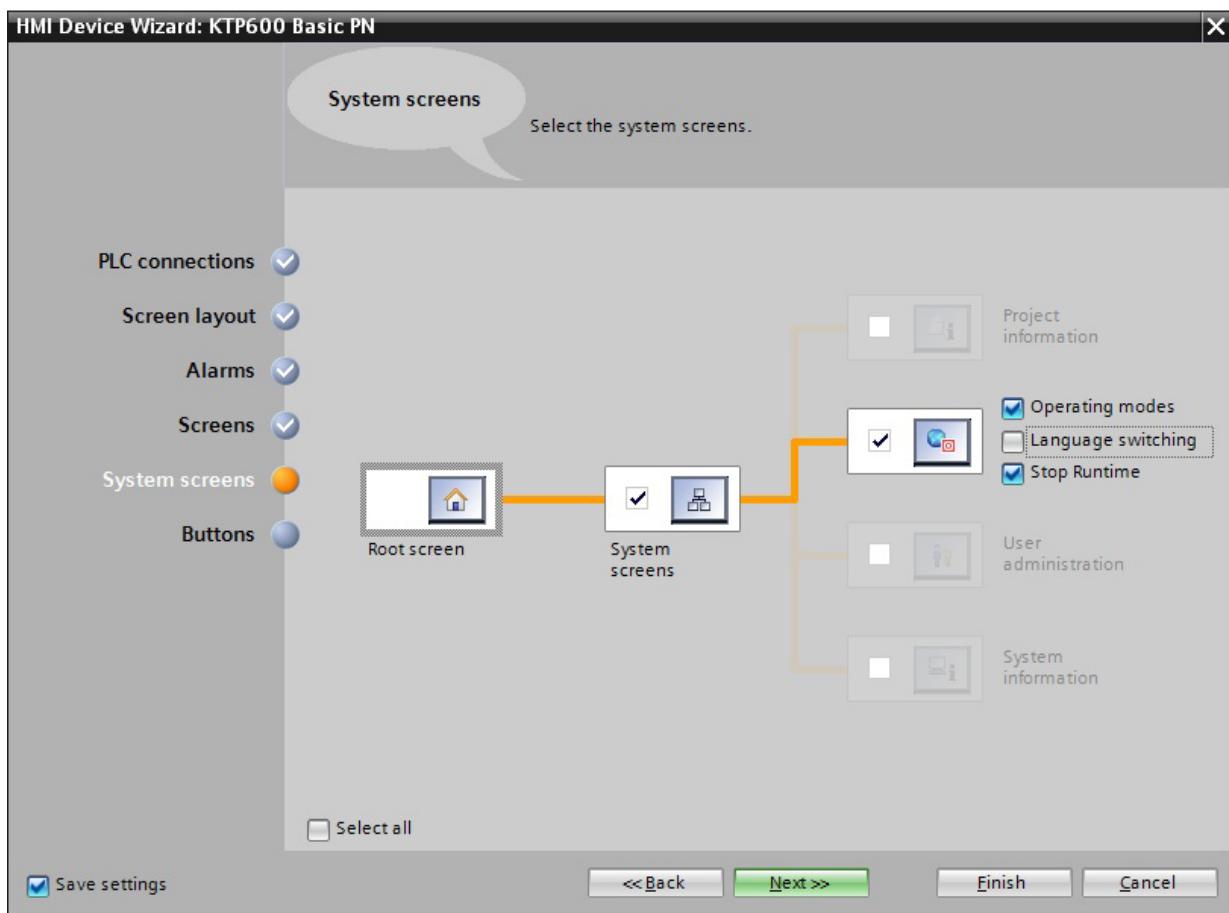
8

Under Screen navigation, a screen menu structure could be set up.
For our example, the “Root screen” is sufficient for the time being.



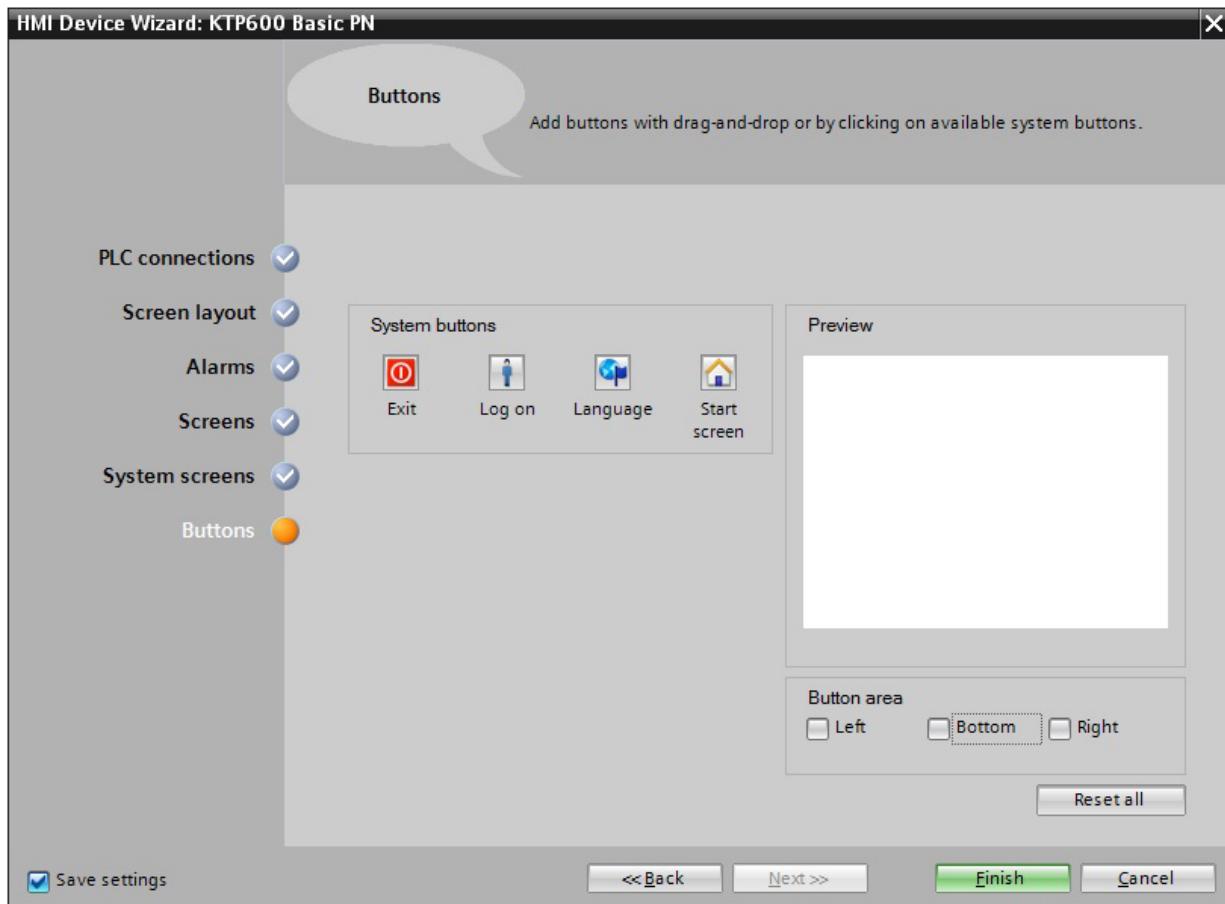
8 Then click on “**Next**“.

As System screen, select the switch-over “**Operating modes**” and “**Stop Runtime**“.



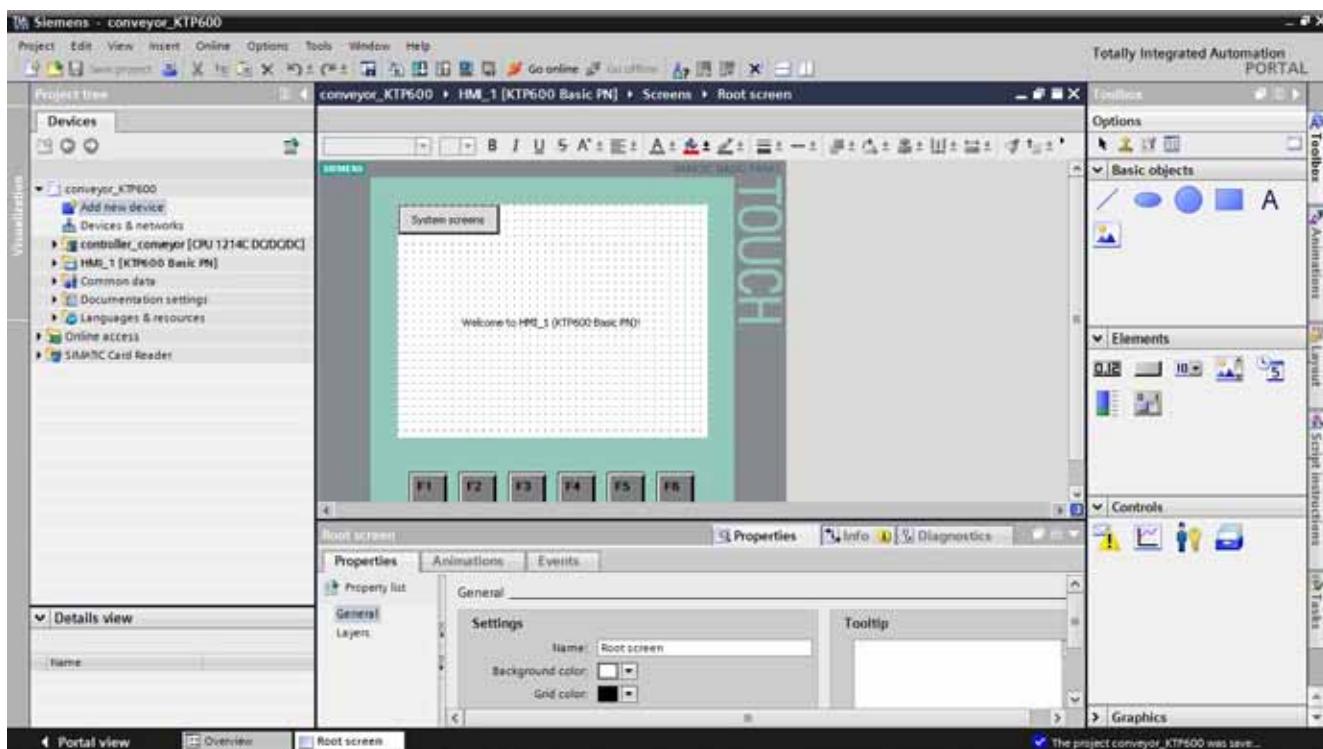
Then click on “**Next**“.

Finally, predefined system buttons can be added.
Remove all checkmarks.

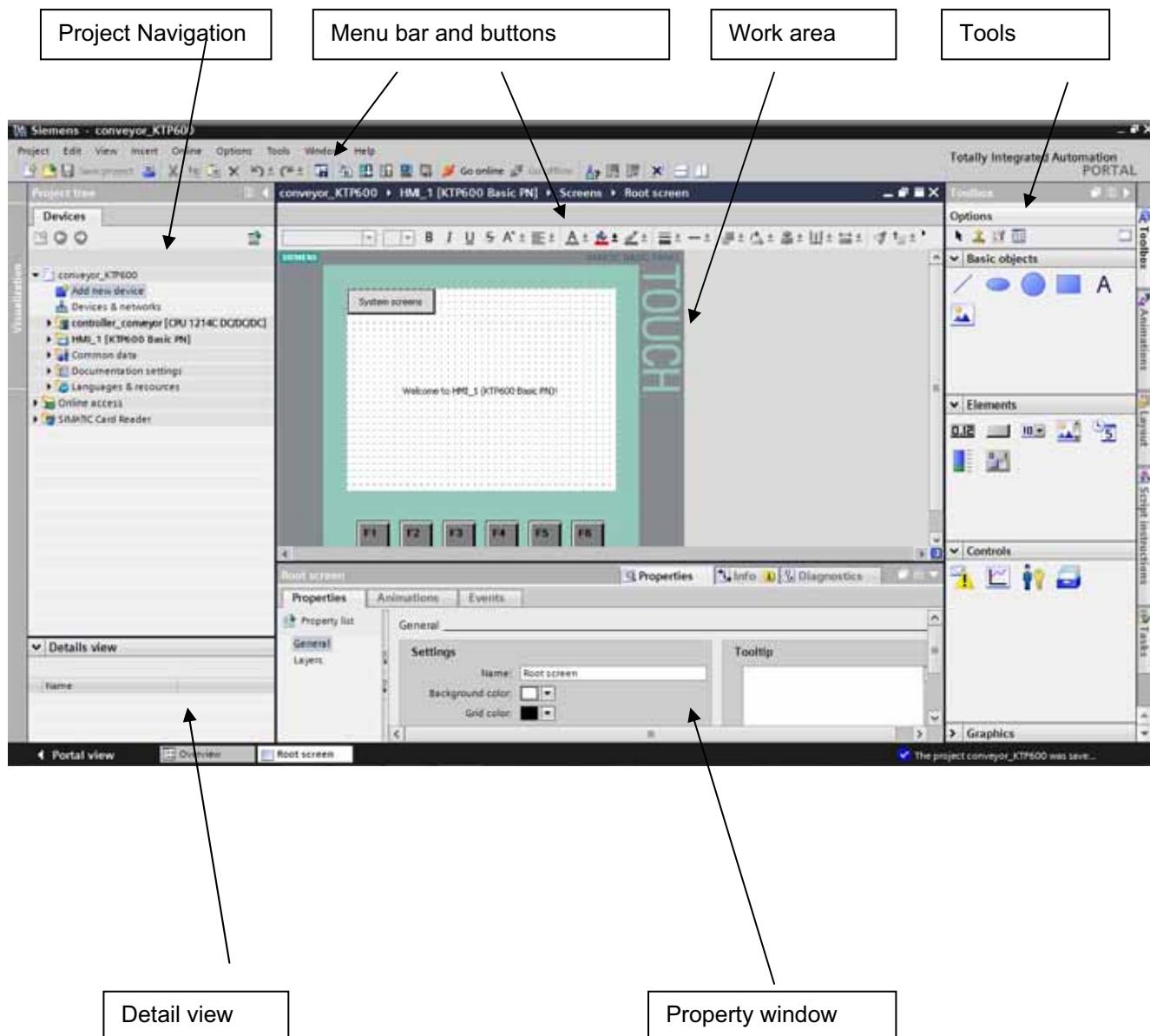


Then click on “Finish“.

The WinCC interface is now opened with the root screen.



5. WinCC Operator Interface



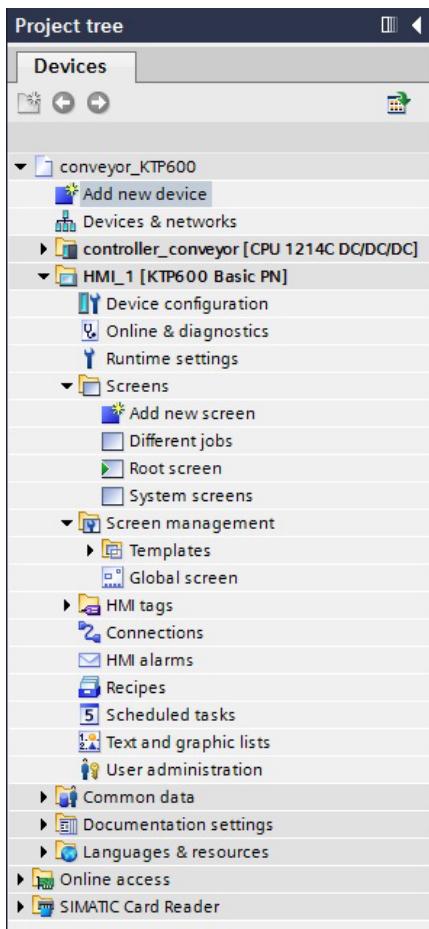
5.1 Project Navigation

The project navigation window is the central connection point for project editing.

All components and all available editors for a project are displayed in a project window in a tree structure, and can be opened from there.

Each editor is assigned a symbol with which the associated objects can be identified. Only those elements are displayed in the project window that the selected operator panel supports.

In the project window, you can access the basic settings of the operator panel.



8

5.2 Menu Bar and Buttons

The menus and the symbol bars provide all functions needed to program the operator panel. If a corresponding editor is active, editor-specific menu commands or symbol bars are displayed.

If you point to a command with the mouse pointer, a corresponding QuickInfo is provided for each function.

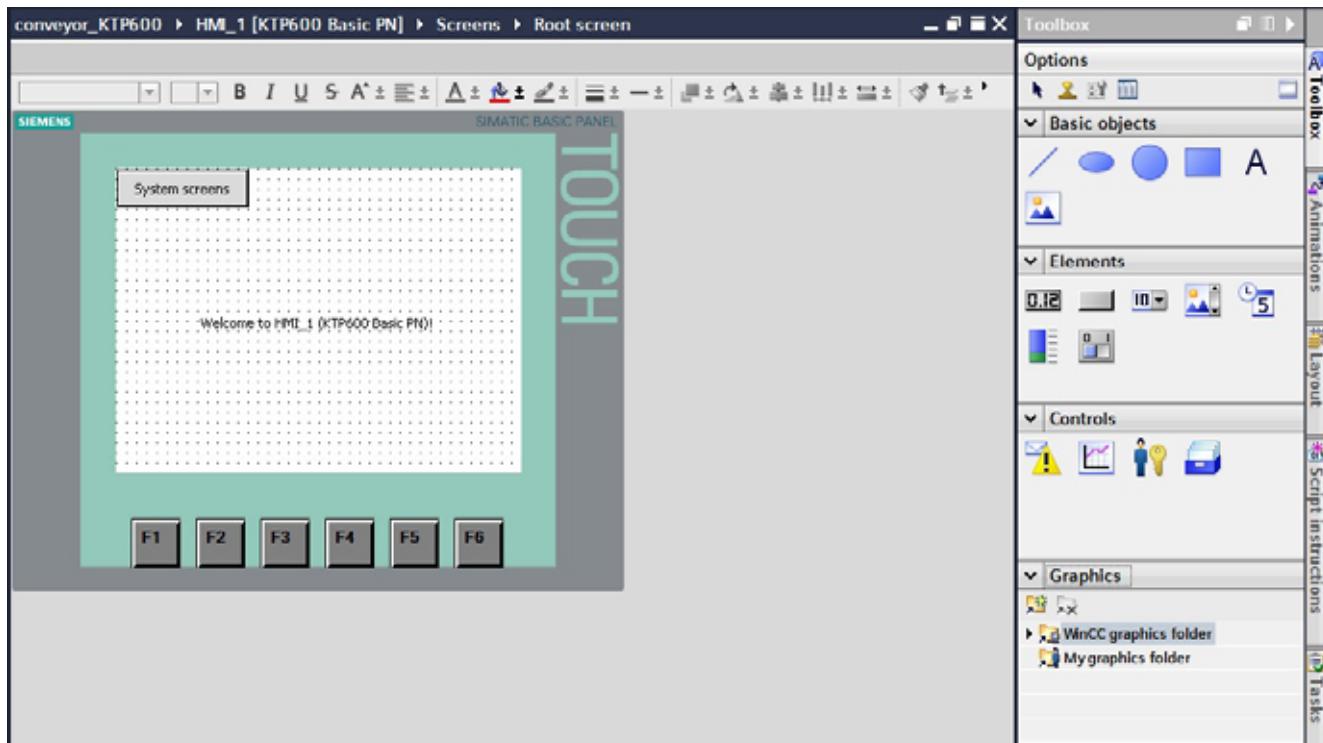


5.3 Work Area

In the work area, we edit objects of the project. All WinCC elements are arranged around the work area. In the work area, we edit the project data either in table form (for example, tags), or graphically (for example, a process display).

A symbol bar is located on the upper part of the work area.

Here, fonts, colors or functions such as Rotate, Align, etc. can be selected.



8

5.4 Tools

In the tool window, a selection of objects is provided that can be inserted in pictures, such as graphic objects and operating elements. In addition, the tool window includes libraries with preassembled library objects and collections of picture blocks.

Objects are moved to the work area with drag&drop.

5.5 Property Window

In the property window, we edit the properties of objects; for example, the color of picture objects.

The property window is available only in certain editors.

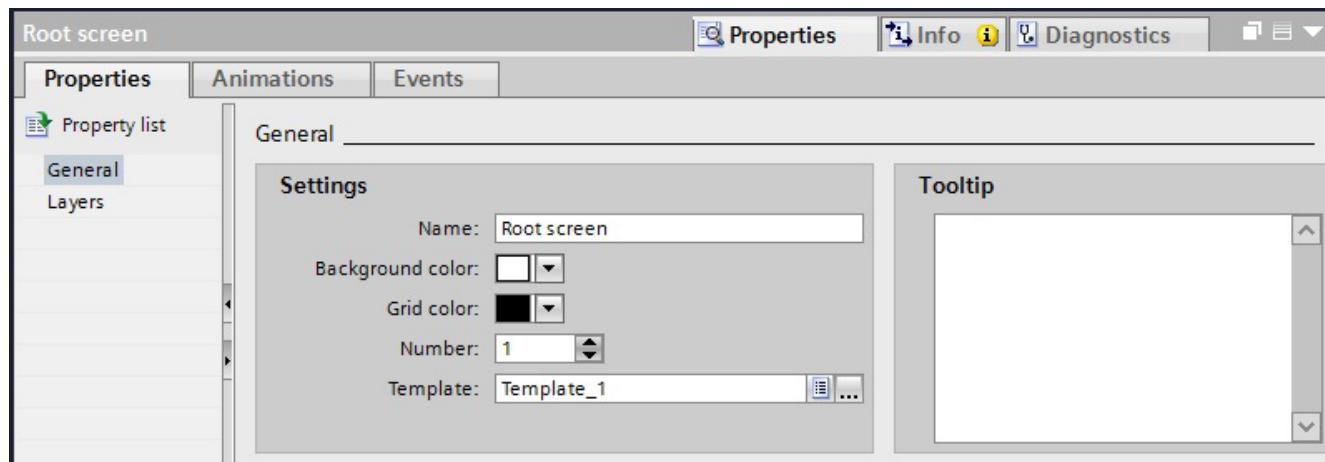
In the property window, the properties of the selected object are displayed arranged according to categories.

As soon as you exit an input field, the value changes become effective.

If you enter an invalid value, its background is colored.

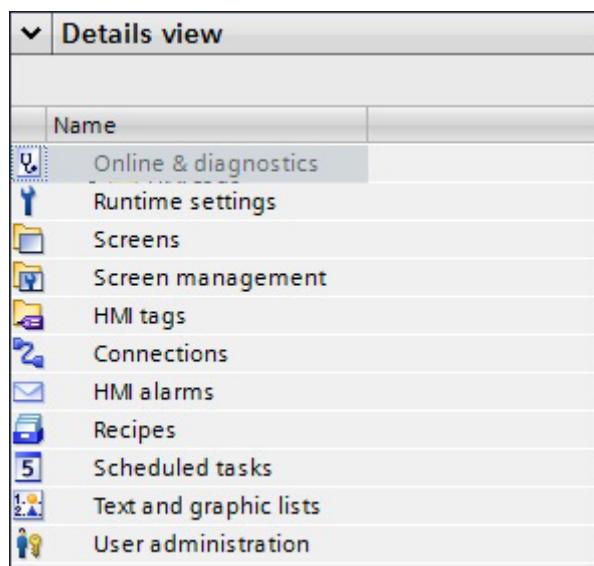
Via the QuickInfo, you are then provided with information regarding the valid value range, for example.

In addition, animations and events of the selected object are configured in the property window; here, for example, a display change when releasing the button.



5.6 Detail View

In the Details view, additional information is displayed regarding the object marked in project navigation.



6. Operating Screens and Connections

A screen can consist of static and dynamic elements. .

Static elements such as text and graphics are not updated by the controller.

Dynamic elements are connected to the controller and visualize current values from the controller's memory. Visualization can be in the form of alpha-numerical displays, curves and bars. Dynamic elements are also inputs on the operator panel that are written to the controller's memory. **Tags** provide for the interfacing with the controller.

First, we are going to create a screen for our conveyor control.

6.1 Root Screen or Start Screen

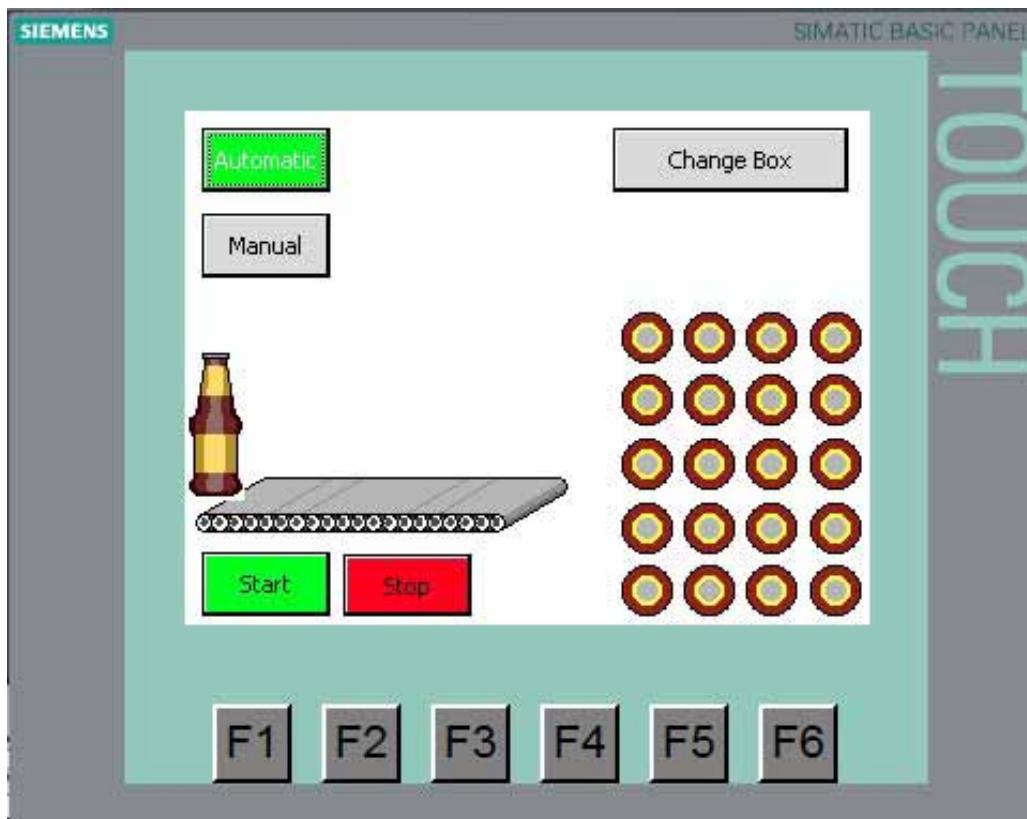
This screen is already set up automatically and defined as start screen.

Here, the entire system is represented.

Buttons are provided for switching between the manual and the automatic mode, for starting and stopping the conveyor motor and for exchanging the case.

The movement of the bottle on the conveyor belt and the number of bottles in the case are shown graphically.

By operating the F6 key, we are jumping to the system screen.



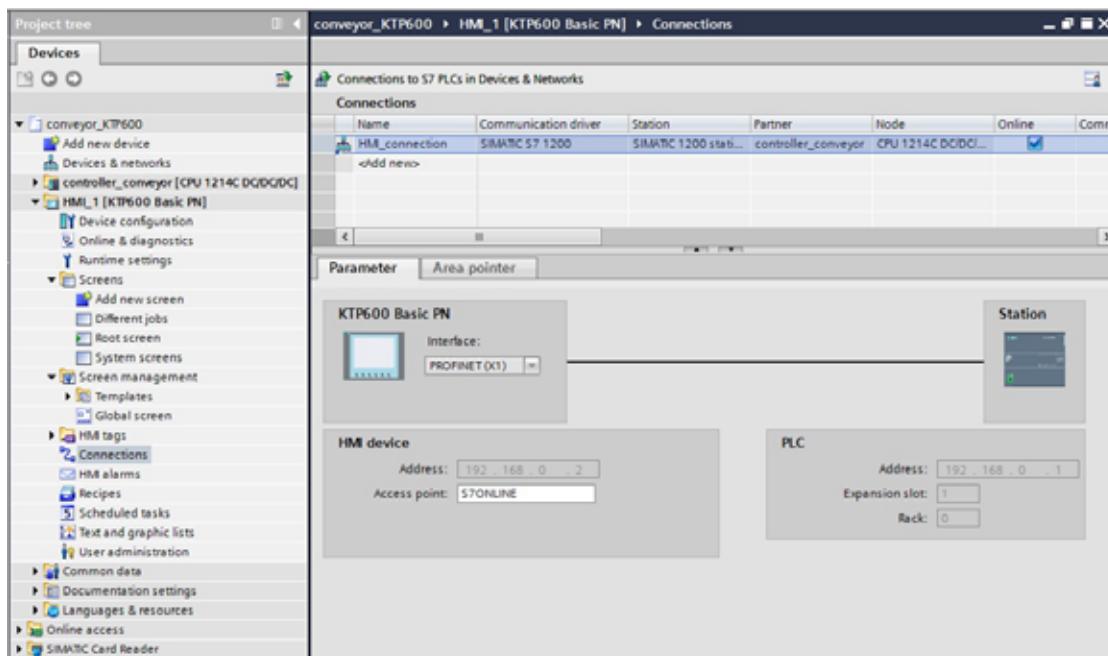
6.2 Connections to S7 Controllers

For operating elements and display objects that access the process values of a controller, first a connection to the controller has to be configured.

Here, we specify how and over which interface the panel communicates with the controller.

In Project navigation, click on **Connections**.

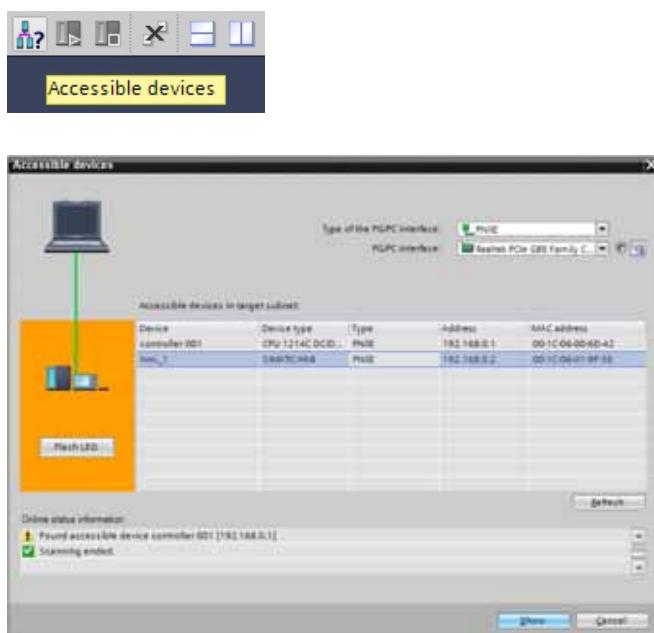
Through the settings in the hardware configuration, all parameters are already set.



The IP address still has to be assigned to the panel.

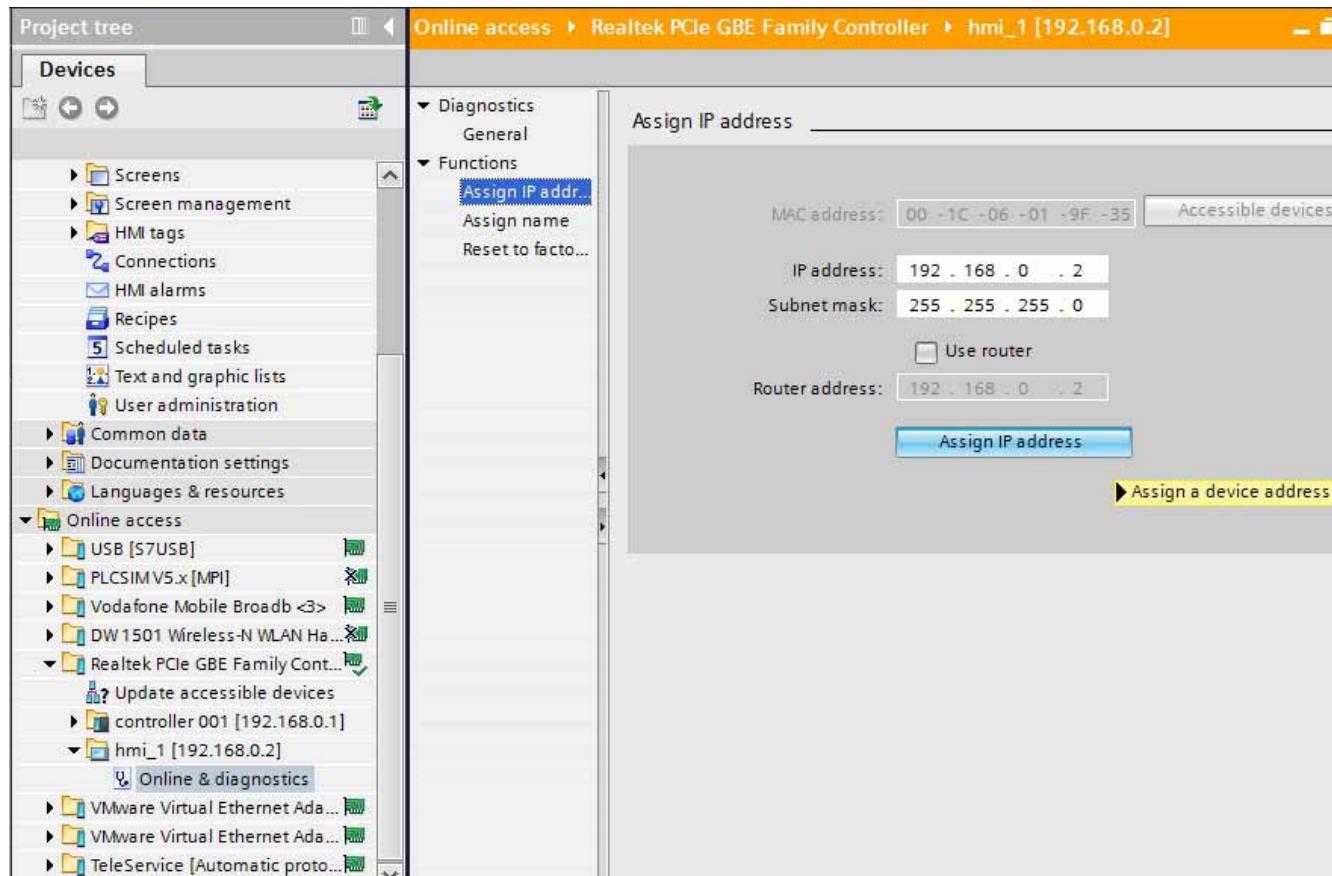
With **Accessible devices**, read out the panel's MAC address

Then click on the button "**Show**".



6.3 Assigning the IP Address

After the MAC address is entered, the IP address can be assigned under **Online & Diagnostics**. The panel has to be in the transfer mode in that case.



Note

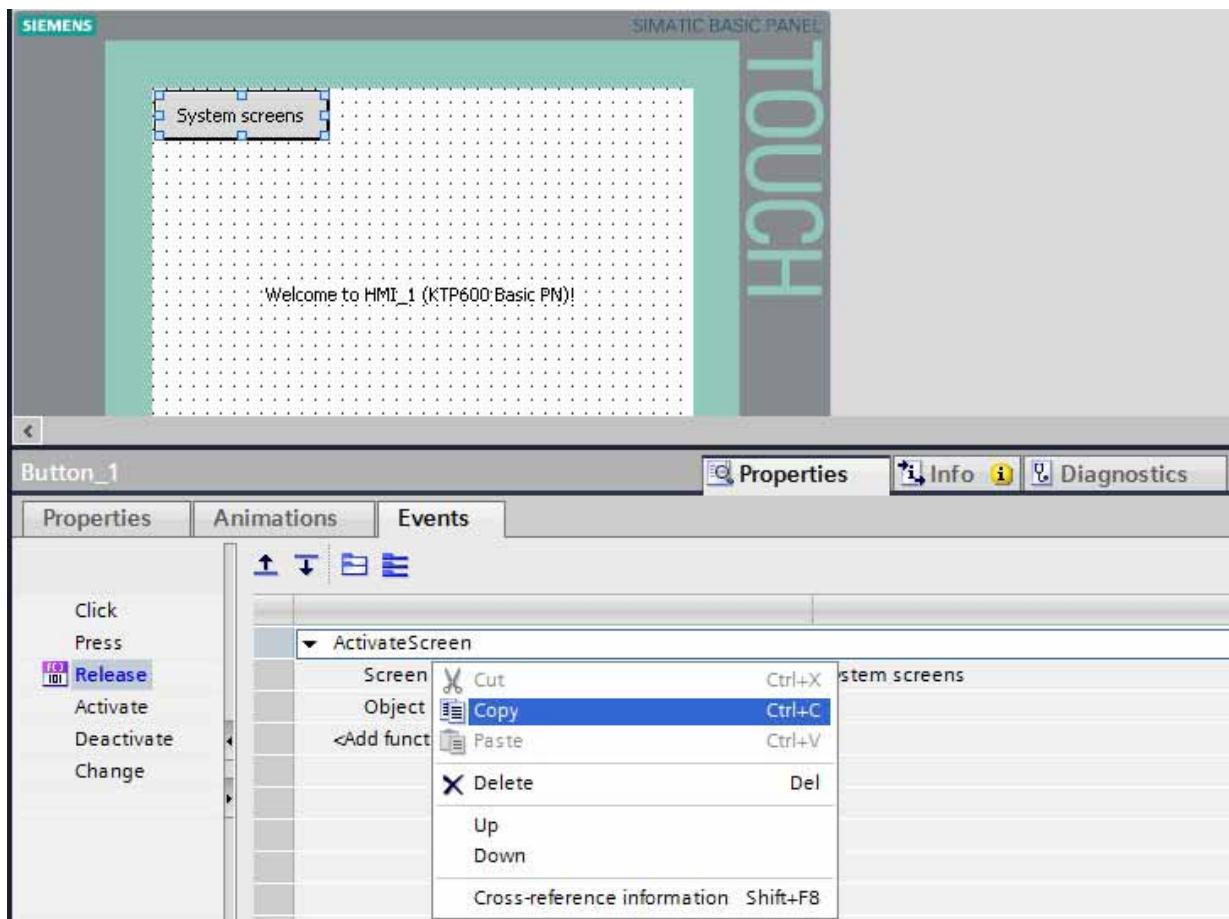
The IP address can also be checked or entered on the panel in the system control under **Control Panel at Profinet**.

7. Configuring the Root Screen

The system screen is called using the button “System screens”.

The function of the button “**System screens**“ is to be transferred to the function key “**F6**“.

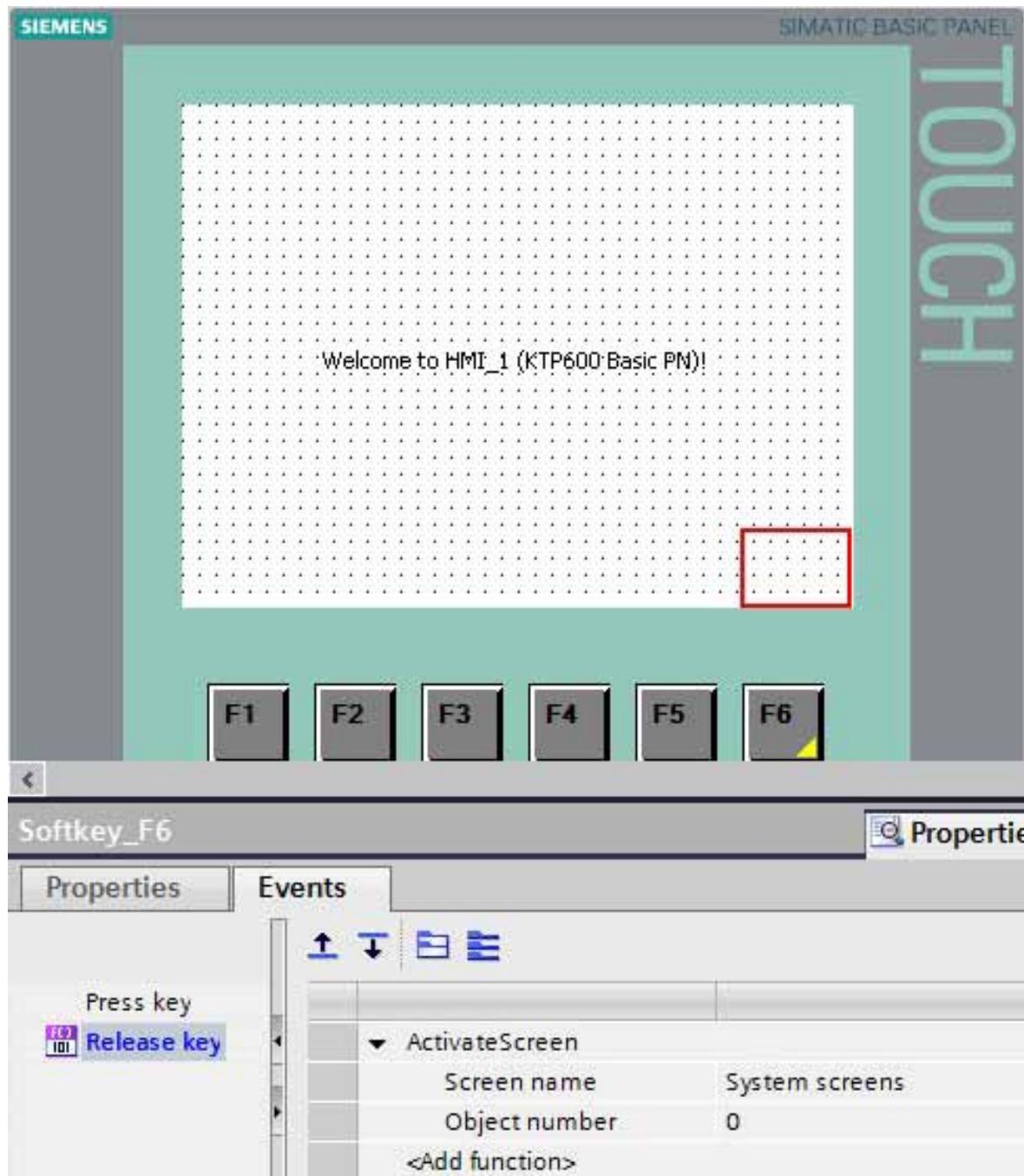
Select the button “**System screens**“ and below, in the property window, copy the function “**ActivateScreen**“ at “**Events**“ “**Release**“.



7.1 Function Key F6

Select function key "F6" and below, in the Properties window, insert the function "ActivateScreen" at "Events" "Release key".

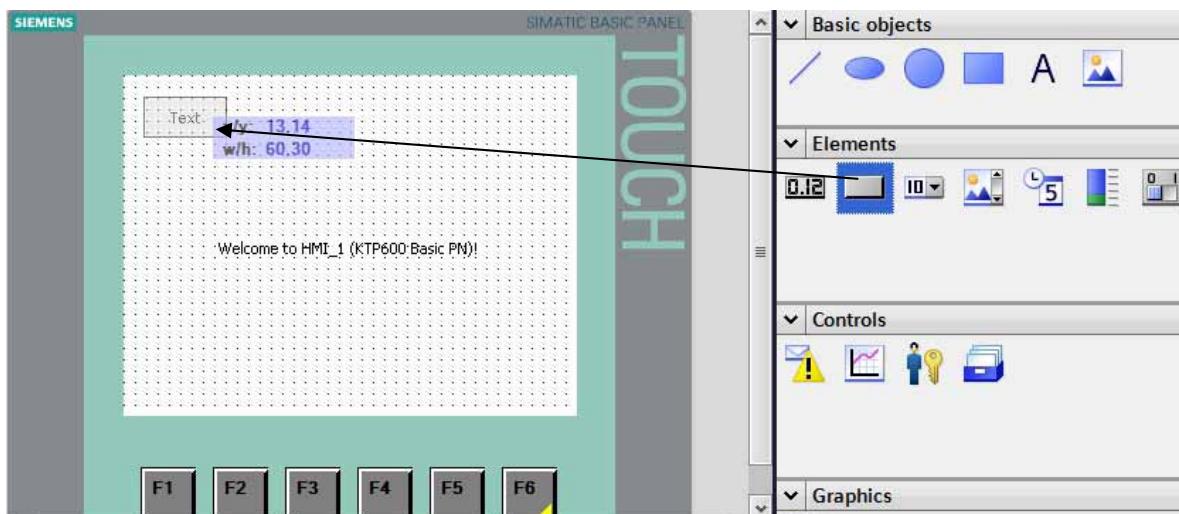
Then, delete or remove the text field in the center and the button "System screens".



The yellow triangle on the function key F6 refers to the key having been configured.

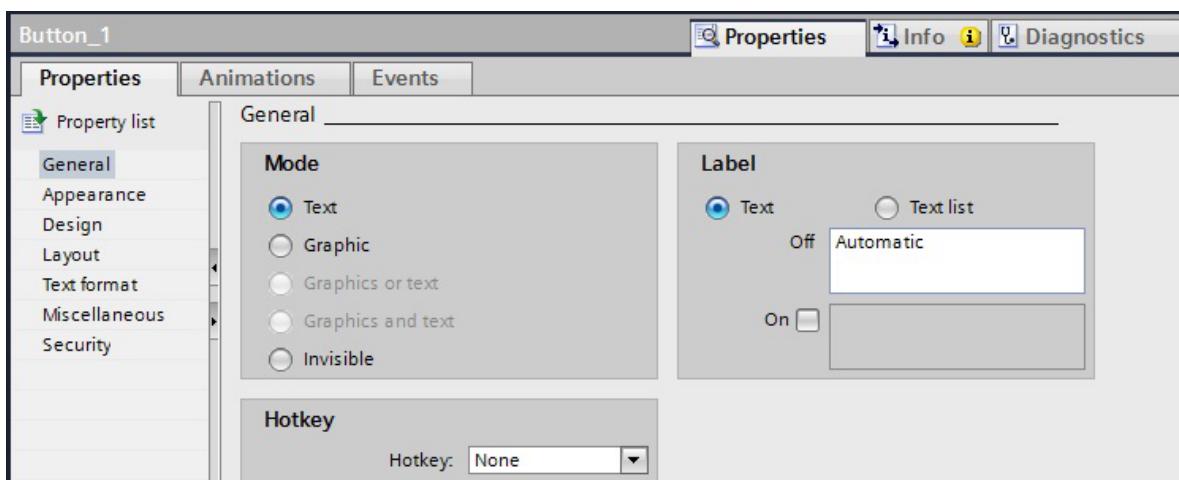
7.2 Configuring the Automatic and Manual Buttons

Drag a button into the work area of the root screen.

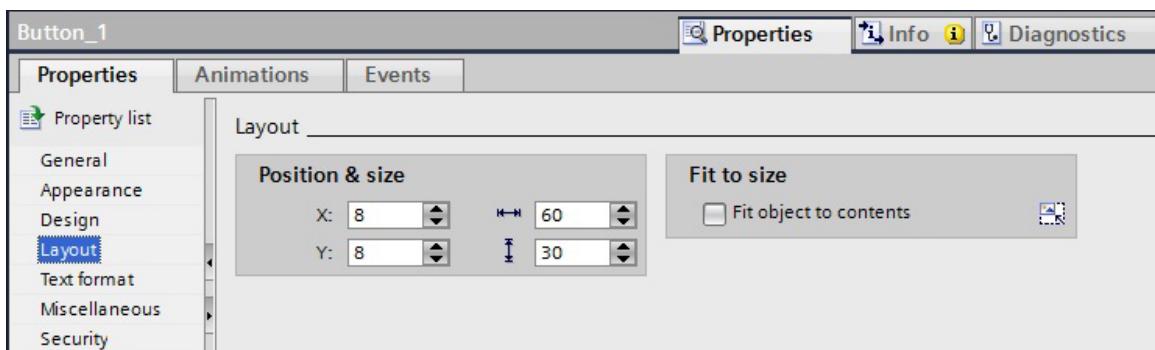


As text, enter **Automatic**.

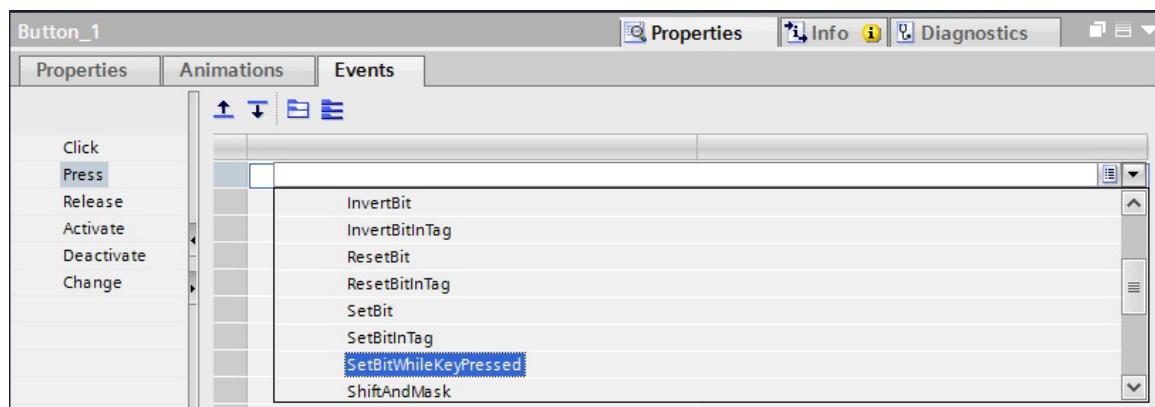
Caution! Don't press the input key; otherwise, a second line is generated.



Under Layout, enter position & size.



Under events, at Press, select the function under bit editing
"SetBitWhileKeyPressed".

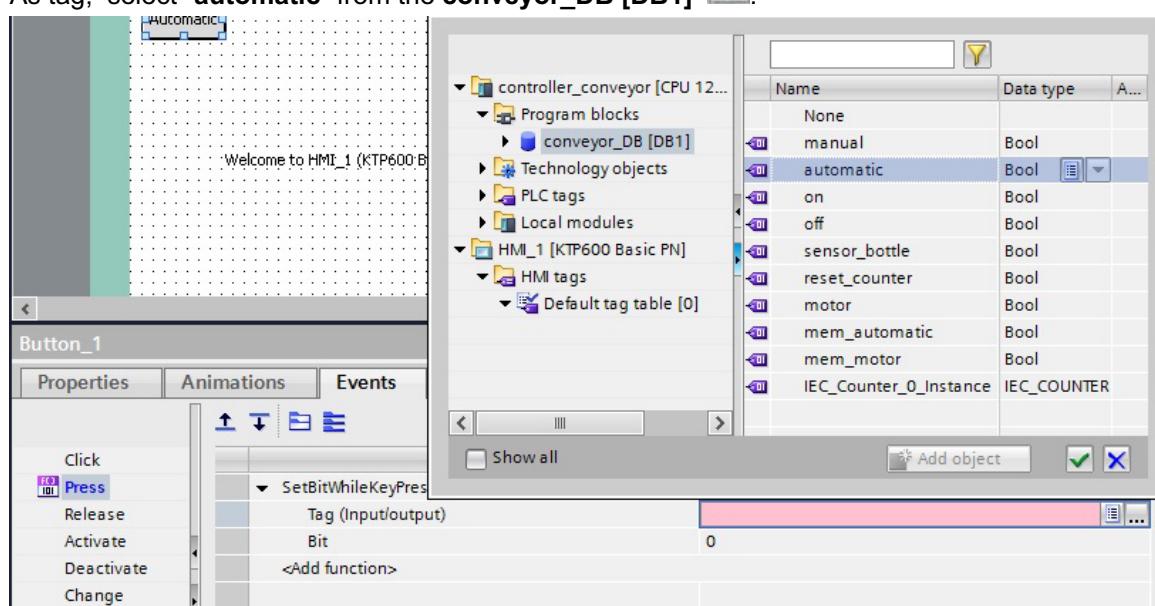


Then, click on the field Tag (Input/Output) and open the tag window with the button "...".



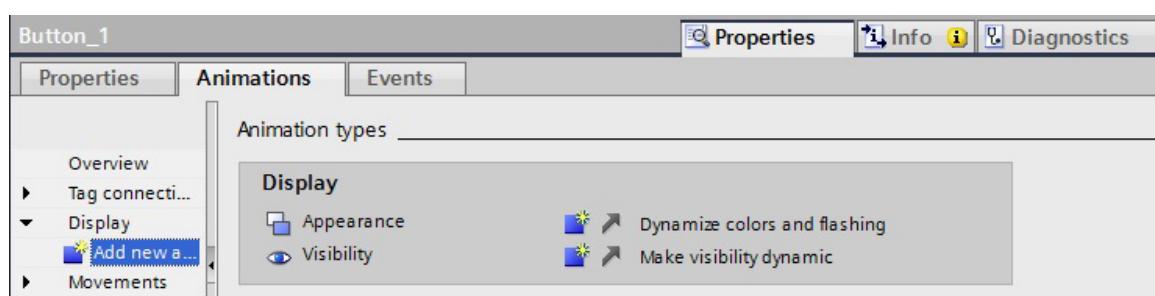
Here, you can also access the interface declaration of data blocks.

As tag, select "automatic" from the conveyor_DB [DB1]

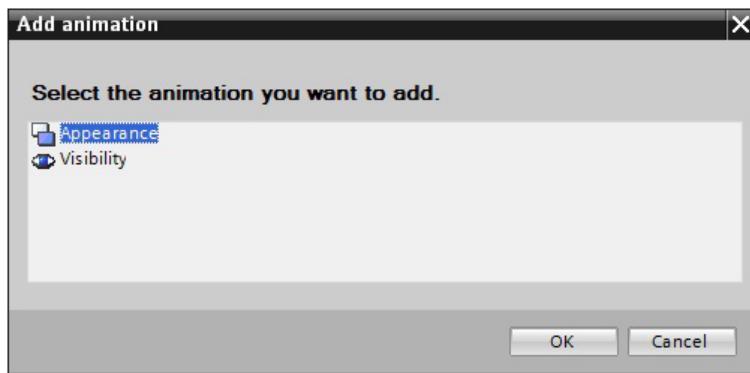


In the automatic mode, the button flashes and changes color.

Under Animations, select Add new animation.



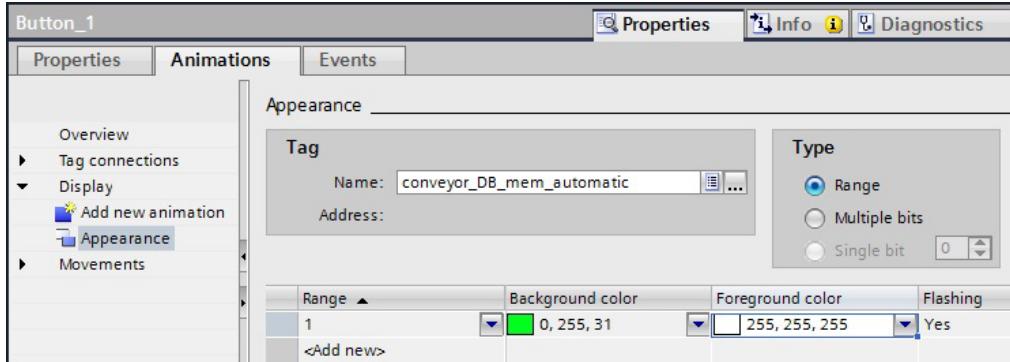
Select **Appearance** and confirm your selection with **OK**.



As tag, select “**memory_automatic**” from the **conveyor_DB [DB1]**.

The button changes color in the automatic mode; i.e., when the tag “**memory_automatic**” has the value **1**. For the color change to be visible, at **Appearance** change the foreground color to **white** and the background color to **green**.

At **Flashing**, set to **yes**.

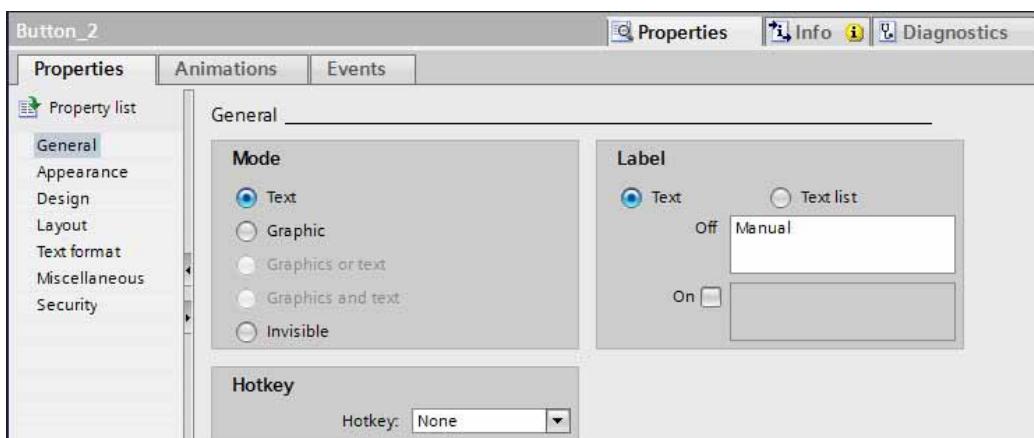


Copy and add the button “**Automatic**”.

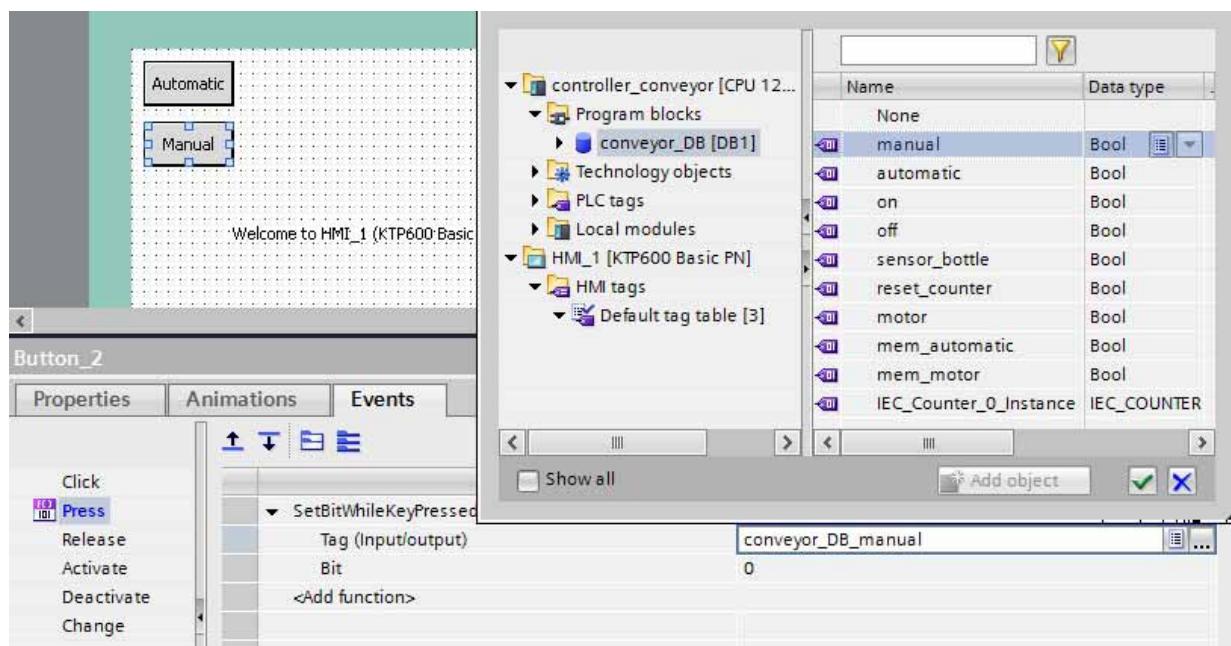
Place the added button below the Automatic button.

As text, enter **Manual** at Label.

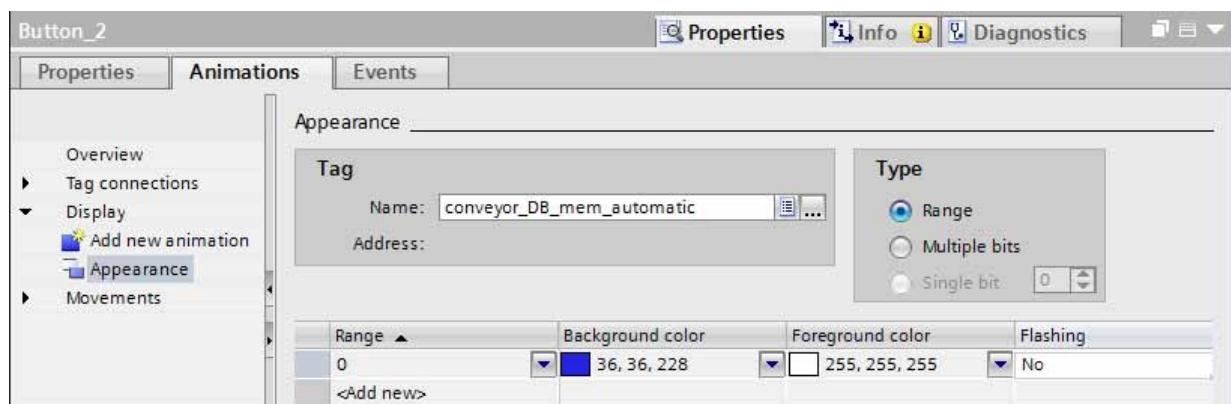
Caution! Don't press the input key; otherwise, a second line is generated.



Under Events Press, select as tag “**manual**” from the **conveyor_DB [DB1]**.
 The tag has to be selected, because only in this way a new HMI tag generated.



The button changes color in the manual mode; i.e. when the tag “**memory_automatic**“ has the value **0**.
 For the color change to be visible, at **Appearance** change the foreground color to **white** and the background color to **blue**.
 At Flashing, set **No**.



Save the project.

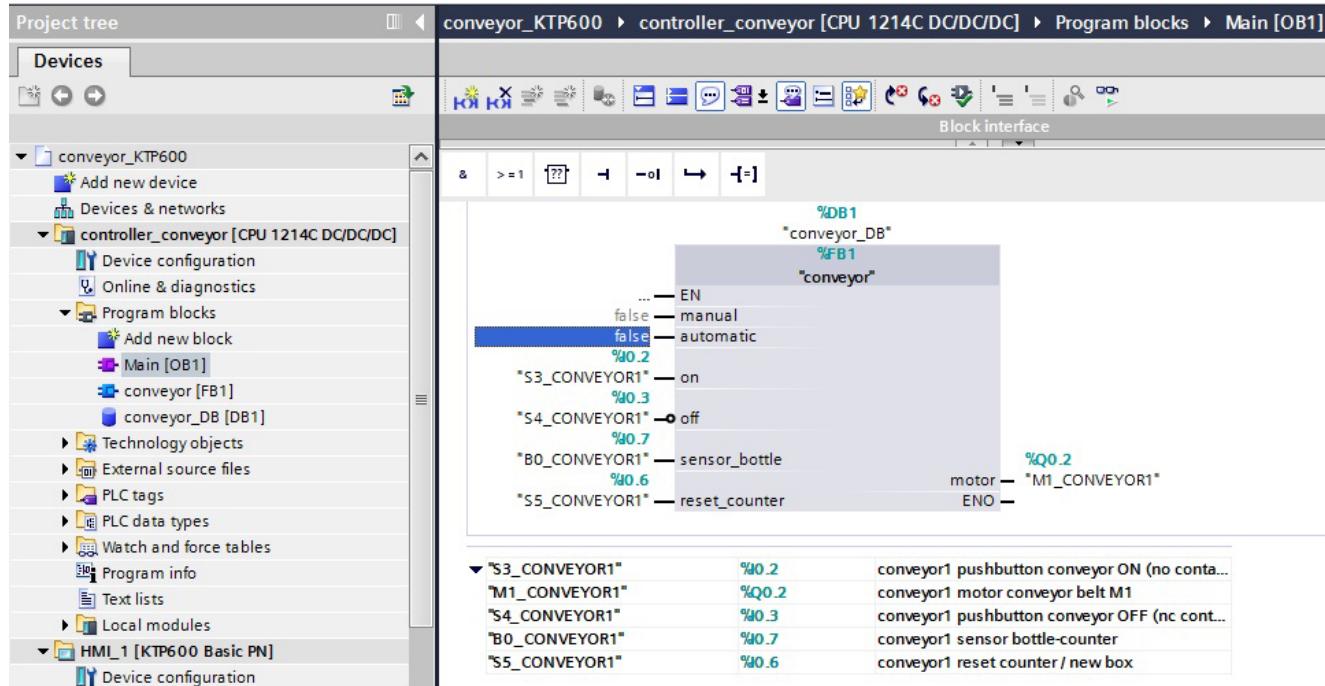
7.3 Changes in the Step7 Program

Before we are testing the visualization, we first have to make a change in the Step7 program.

In OB1, remove the assignment S1_conveyor1 and S2_conveyor1 when calling FB1.

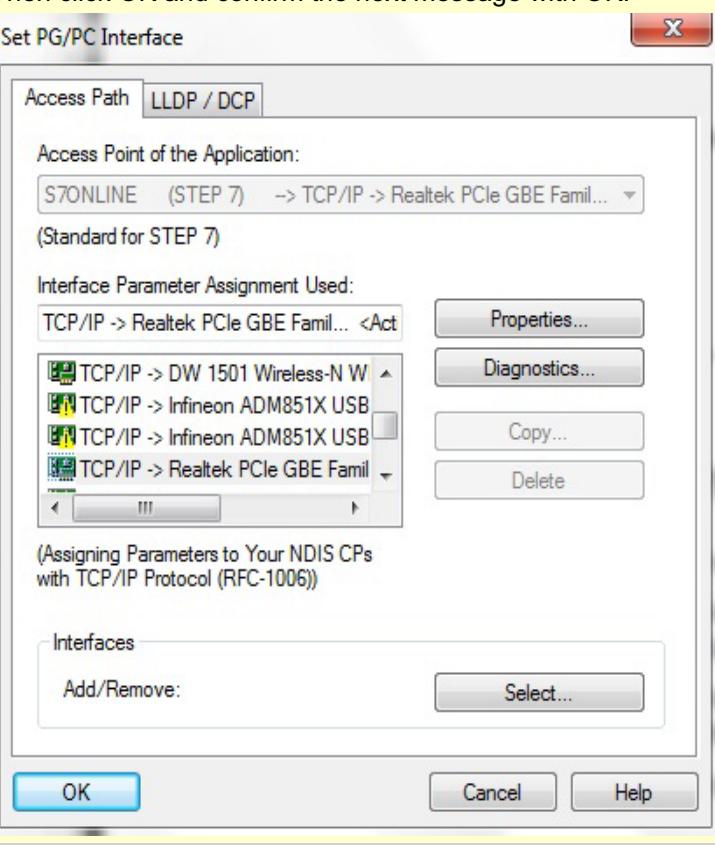
This is necessary since otherwise, the panel signals are overwritten by the process image of the inputs.

Save and load the modified program.



7.4 Setting the PG/PC Interface for Runtime Simulation

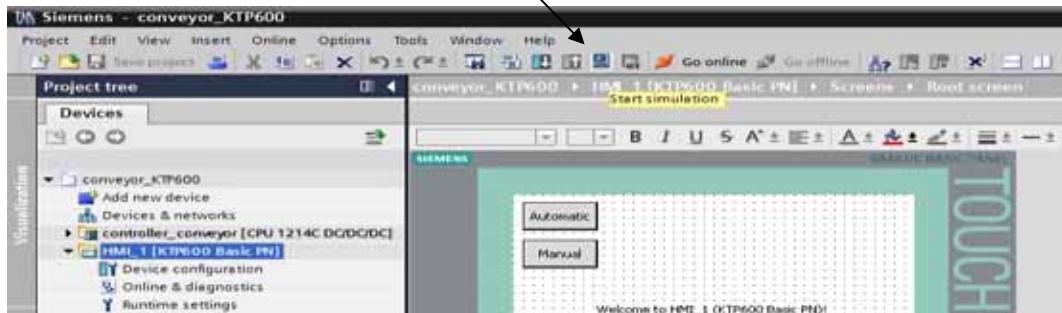
For a connection to be set up between the runtime simulation on the PG/PC and the S7-1200 CPU, first the PG/PC interface has to be set to TCP/IP.

No.	Procedure:
1	<p>Open the Control Panel</p> <ul style="list-style-type: none"> • with "Start > Control Panel" (Start menu for easy access to programs under Windows XP), • or via "Start > Settings > Control Panel" (with class Start menu as is the case in earlier Windows versions).
2	<p>In the Control Panel, double-click the icon "Set PG/PC Interface".</p>  PG/PC-Schnitt stelle einstellen
3	<p>Set the following parameters in the "Access Path" tab:</p> <ol style="list-style-type: none"> 1. For the access point of the application, select "S7ONLINE [STEP 7]" from the drop-down list. 2. From the list of interface parameter assignment used, select the interface "TCP/IP(Auto) -> with your network adapter that is connected directly with the Panel and the controller, e.g., 3Com EtherLink XL. 3. Then click OK and confirm the next message with OK. 

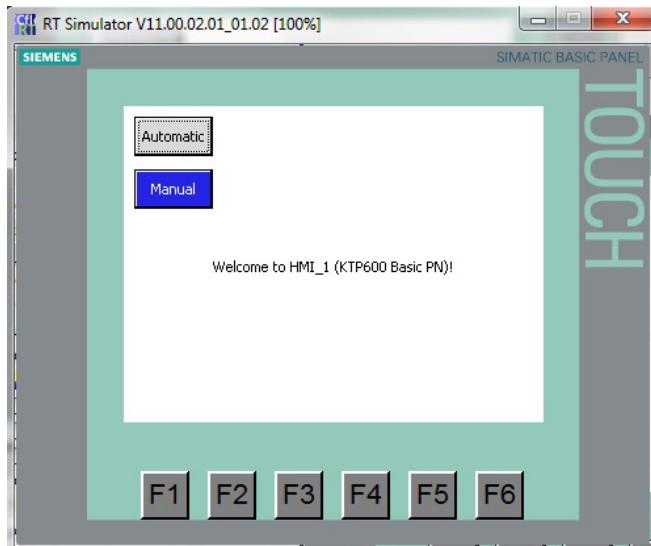
7.5 Starting the Configuration in Runtime

In the project window, select the **HMI_1 [KTP600 Basic PN]** panel

Click in the button “**Start simulation**”.

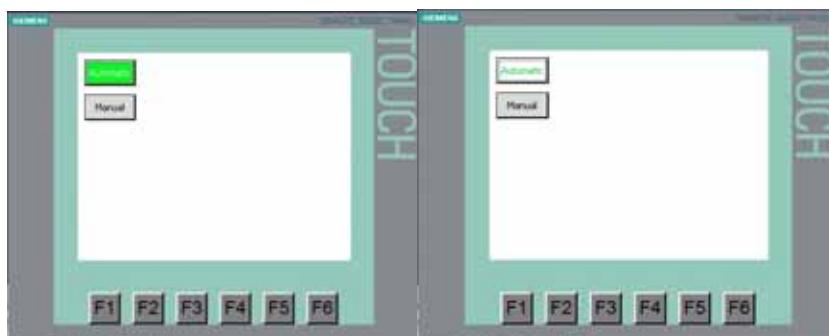


Visualization is opened in the RT simulator.



Test the conveyor control project.

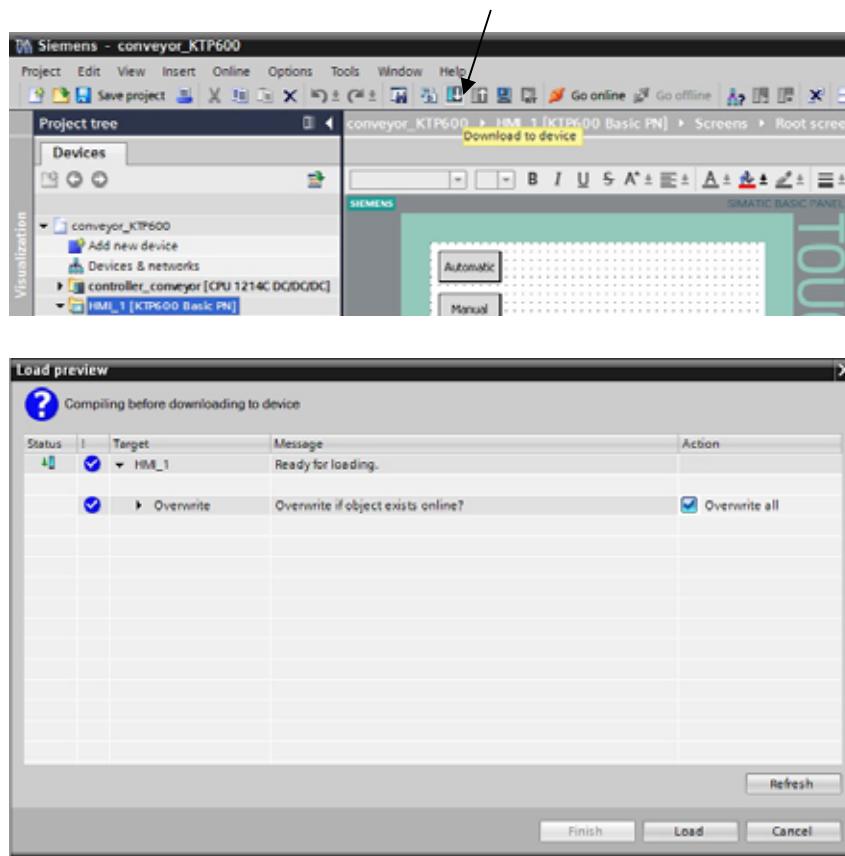
Either the automatic or the manual mode is now preselected on the panel.



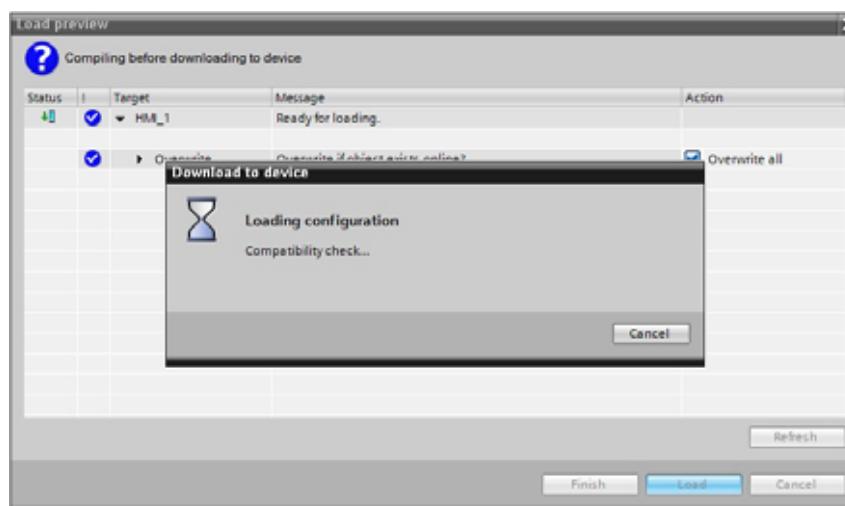
7.6 Loading the Configuration to the Panel and Testing It

In the project window, select the **HMI_1 [KTP600 Basic PN]** panel

Click on the button “**Load to device**”.



Click on the button “**Load**”



If the operating system on the panel is not current, an additional window is displayed for updating the operating system.

Test function key **F6** also.

7.7 Start and Stop Button

Now we are going to configure the start and stop buttons.

The button "Start" is created in the same manner as the Automatic and Manual buttons.

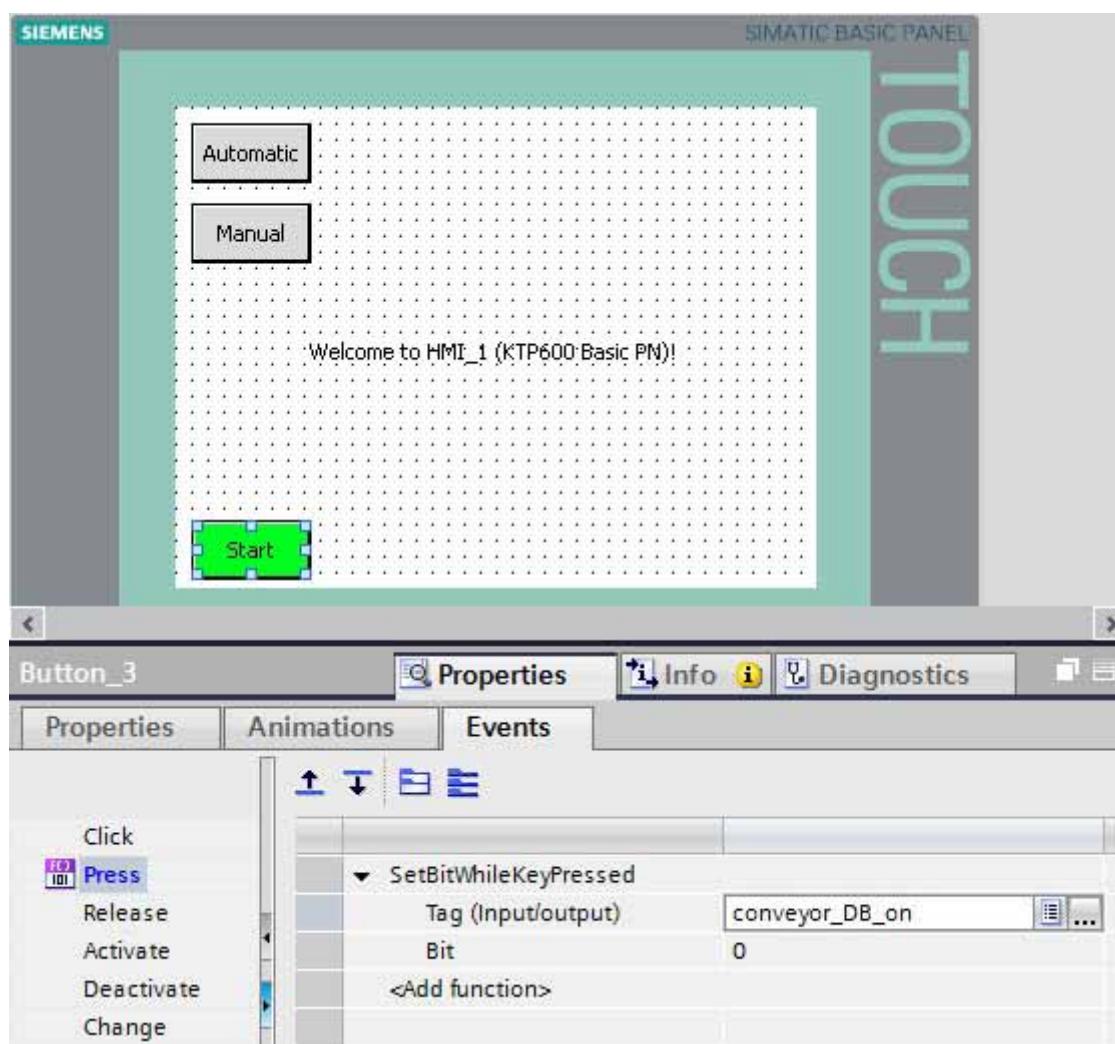
The button "Stop" has a break contact function and has to remove the signal when operated.

Generate the "Start" button.

Set the background color to green.

Under Events Press, select the function **SetBitWhileKeyPressed** under bit editing.

Select the tag "on" from **conveyor_DB [DB1]**.

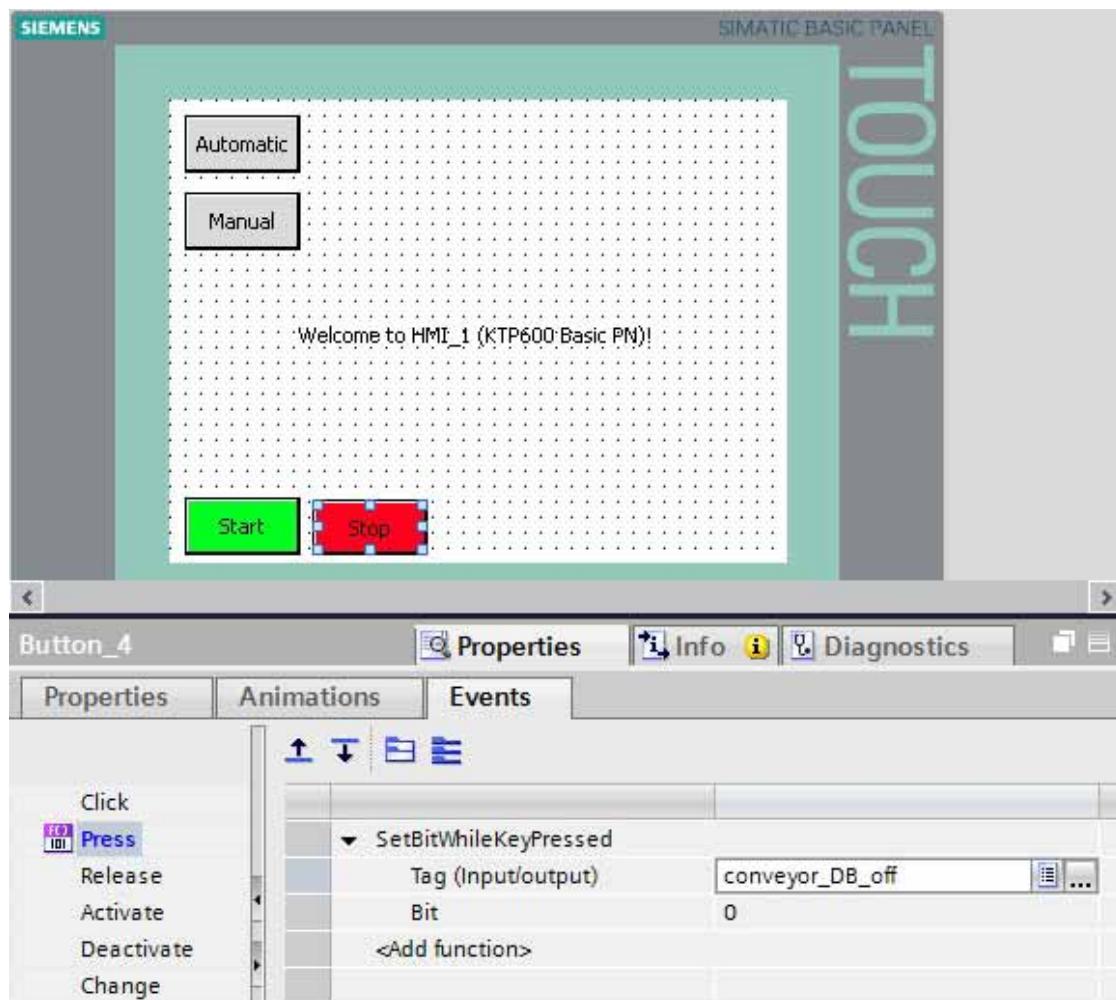


Generate the "Stop" button.

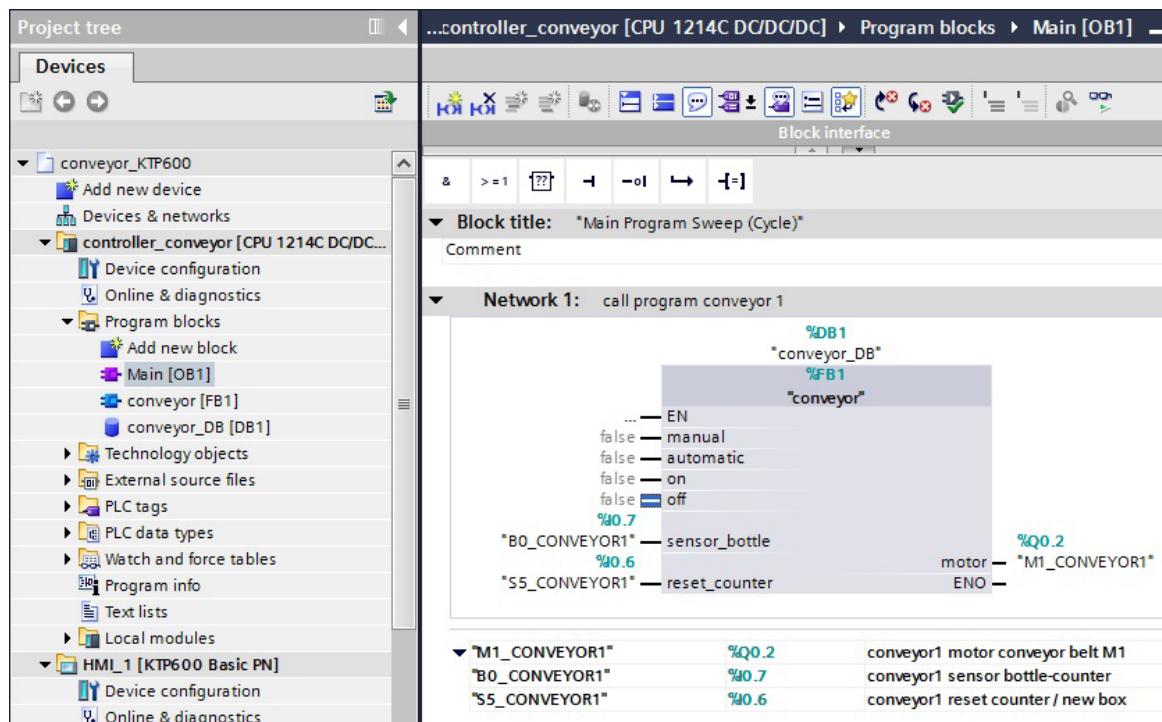
Set the background color to red.

Under Events Press, select the function **SetBitWhileKeyPressed** under bit editing.

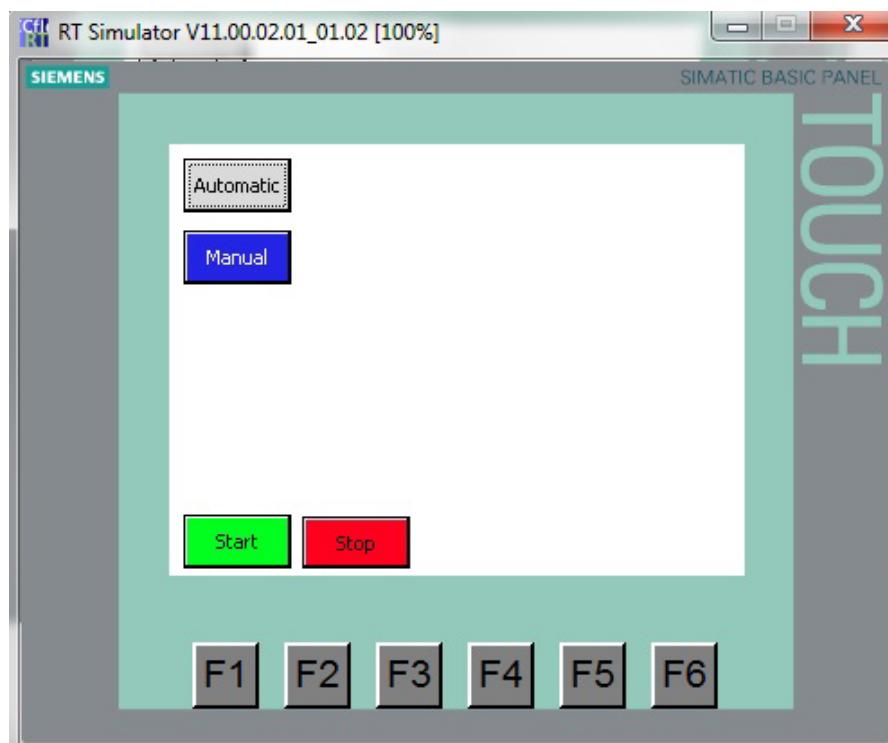
Select the tag "off" from **conveyor_DB [DB1]**.



Before we test visualization, another change has to be made in the Step7 program.
 In OB1, remove the assignment S3_Conveyor1 and S4_Conveyor1 when calling FB1.
 Delete the negation at the **off** input of the block.
 Save and load the modified program.



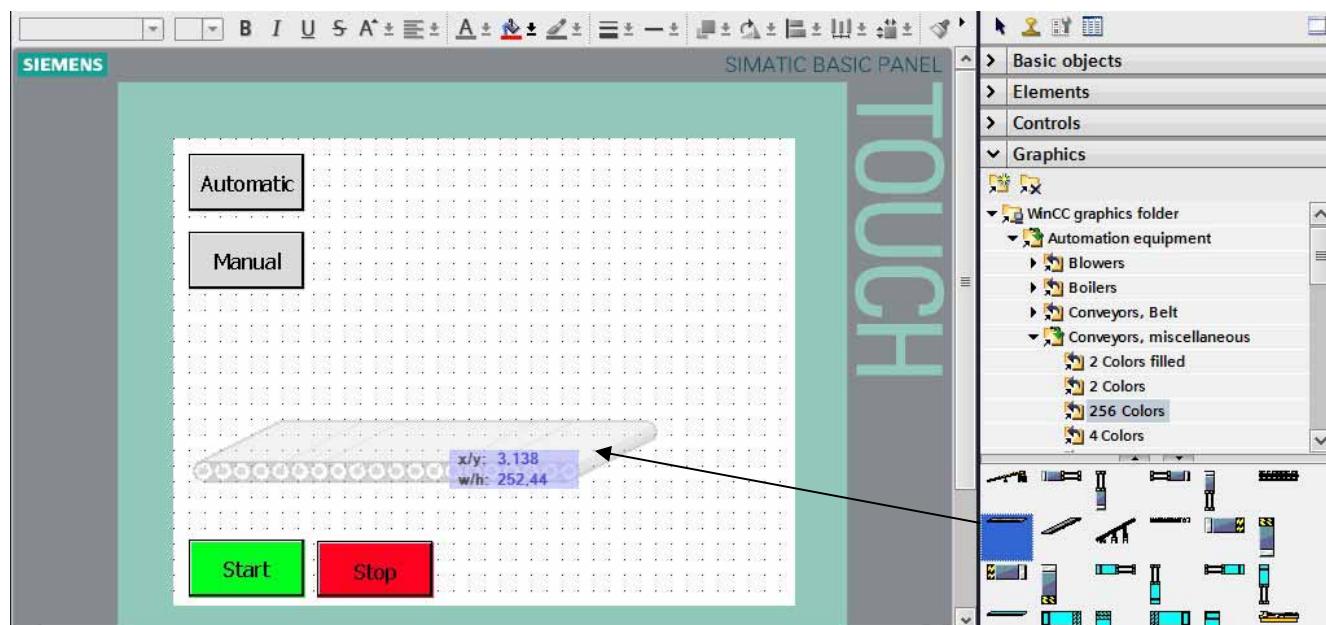
Load the configuration to the panel and test the **Start** and **Stop** buttons.



8

7.8 Adding Graphics from the Graphics Folder

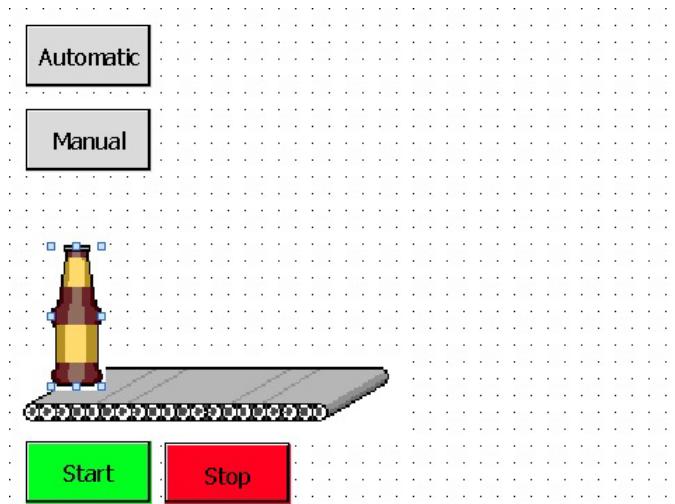
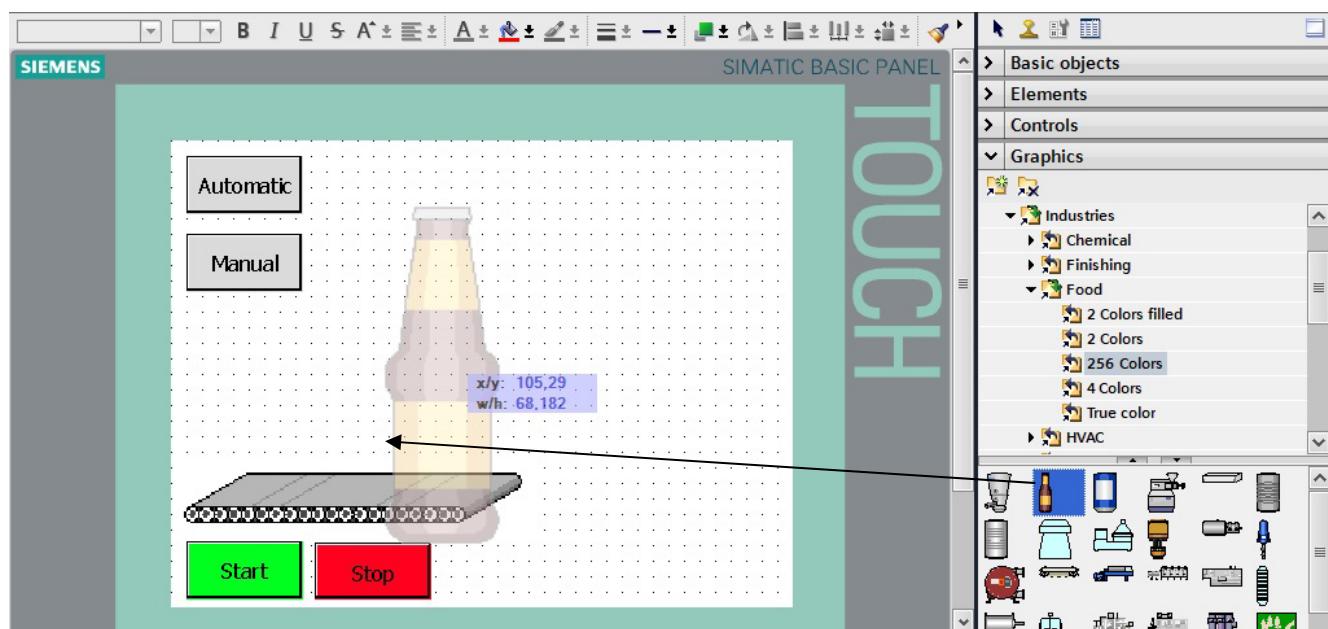
In the tool box under Graphics, open the directory tree **WinCC graphics folder**
Drag the conveyor belt graphic to the root screen and drop it. .



In the tool box under Graphics, open the directory tree **WinCC Graphics folder**.

Drag the graphic of the beer bottle to the basic screen and drop it.

Change the size and the position of the bottle.



Note

All graphic objects have to be located **within** the work area (320x240 pixels).

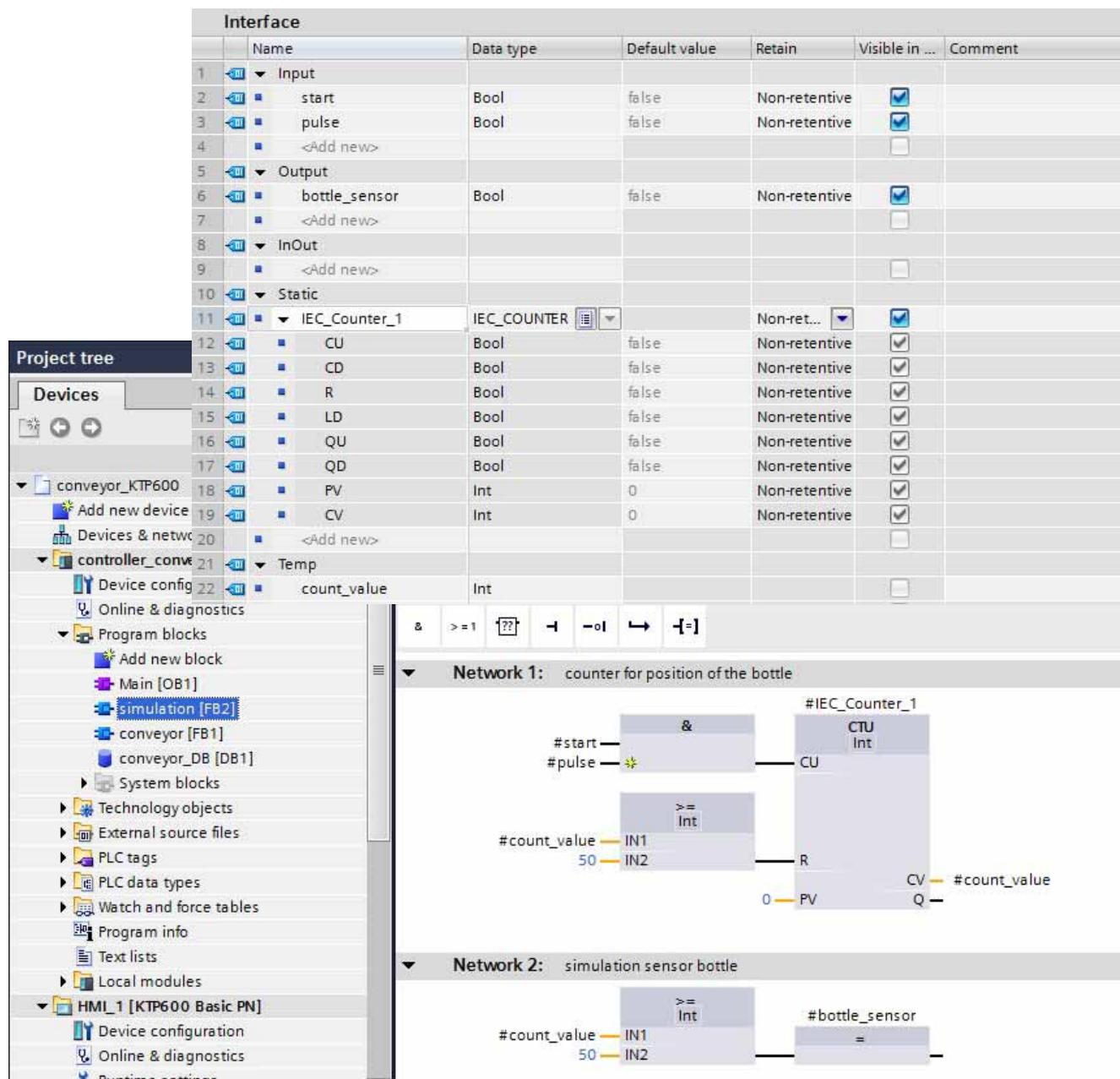
7.9 Control Program for Simulating Bottle Movement

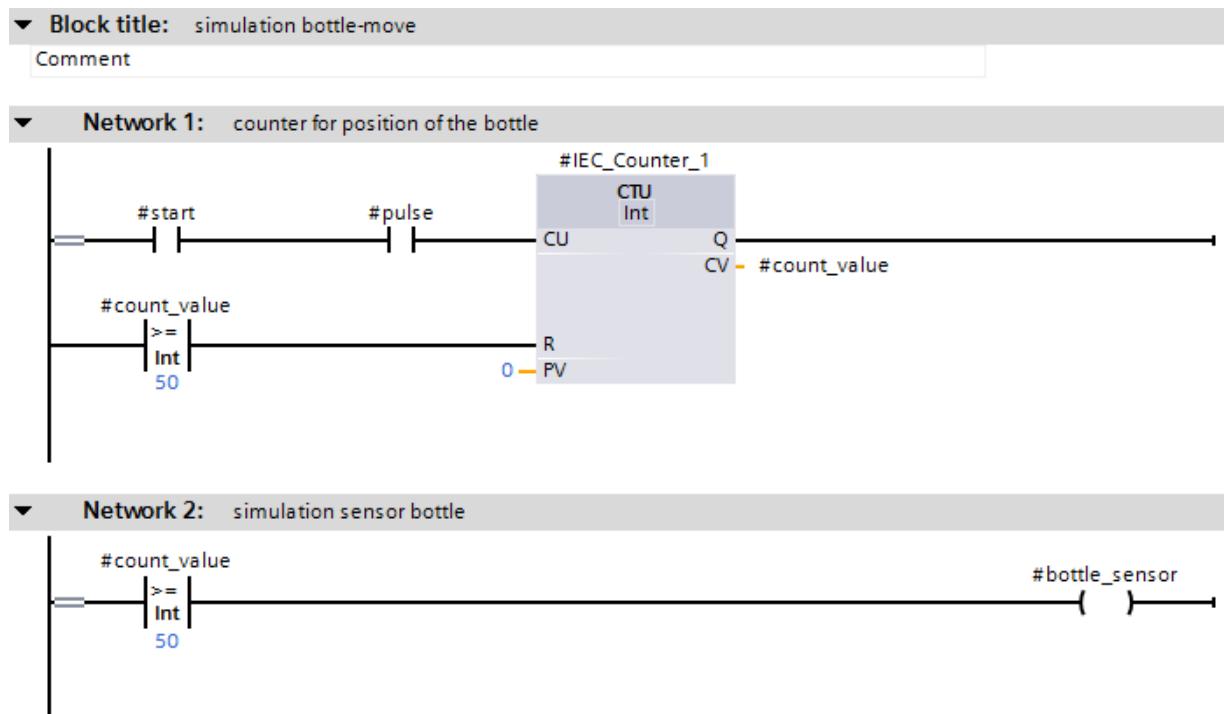
To simulate bottle movement and the bottle sensor, we are creating a new block. The FB2 (simulation) below with tag declaration and network consists of a counter that, activated by a start signal, always counts up from 1 to 51.

In Network 1, the CTU (up counter) is added as multi-instance.

In Network 2, a bottle sensor pulse signal is read out when the count reaches 50.

This simulates when a bottle leaves the conveyor.

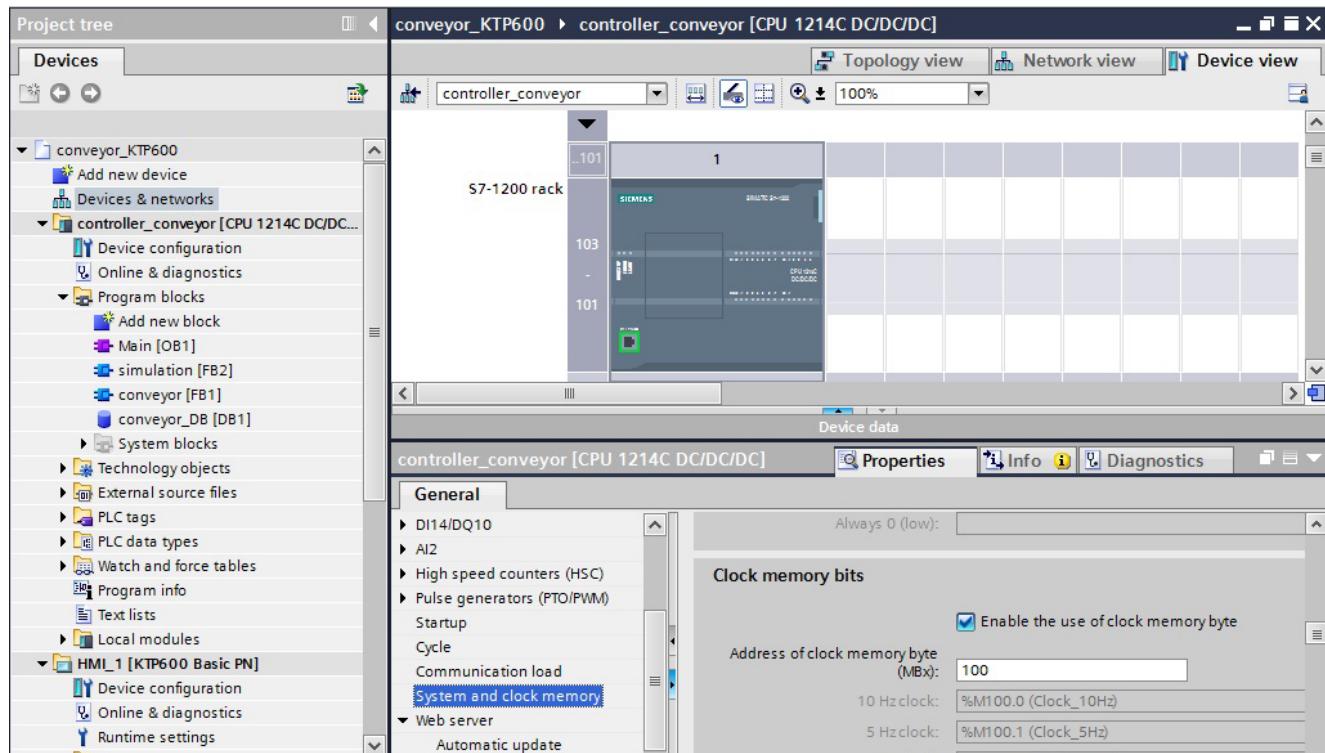


Program in ladder diagram (LAD):

7.10 Activating the Clock Memory and Assigning the MB100

As clock memory, an internal CPU clock memory bit is used.

Activate the clock memory bits, and assign MB100 as address.



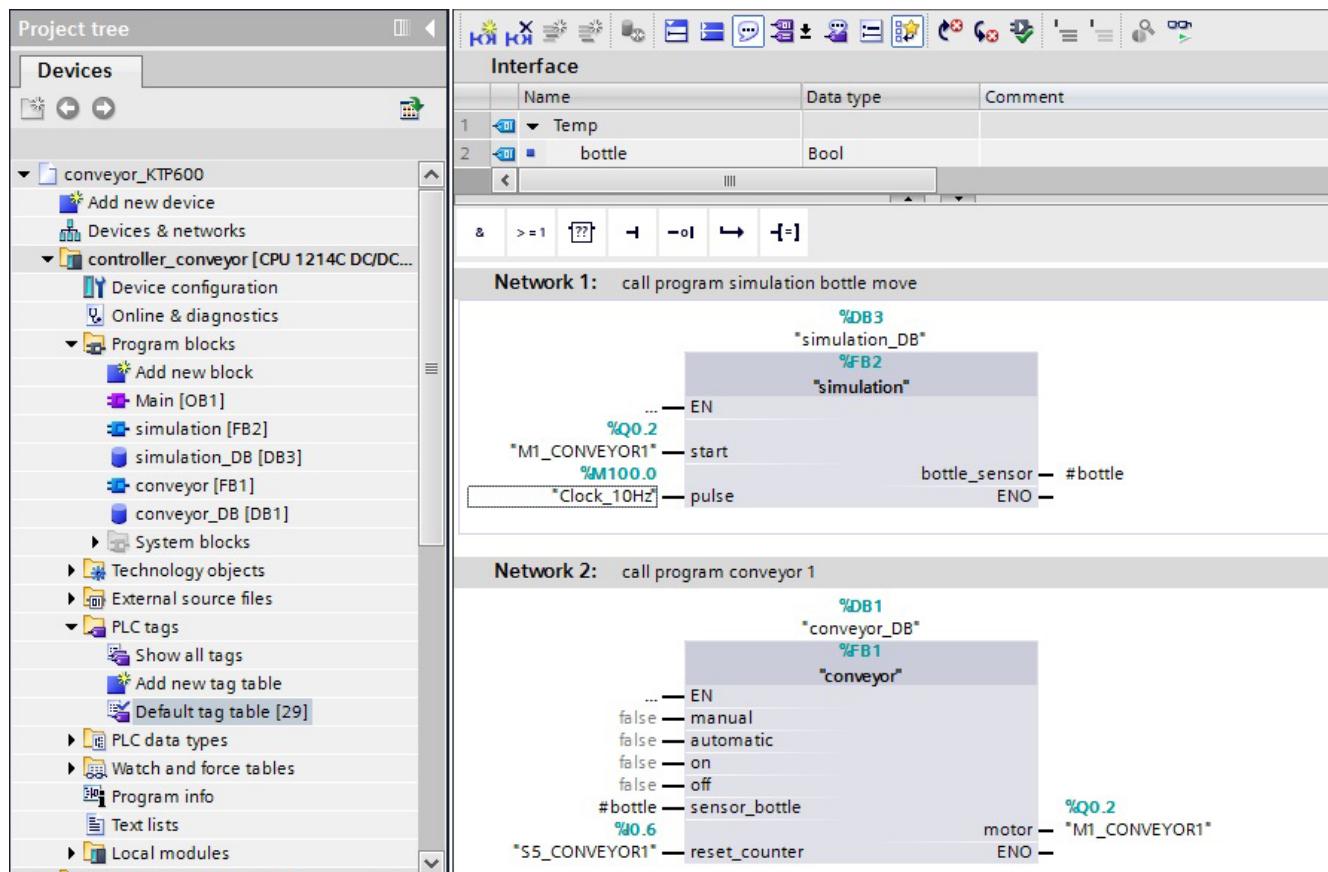
7.11 Calling FB2 (Simulation) in OB1

Before calling FB1 (conveyor), add a new network.

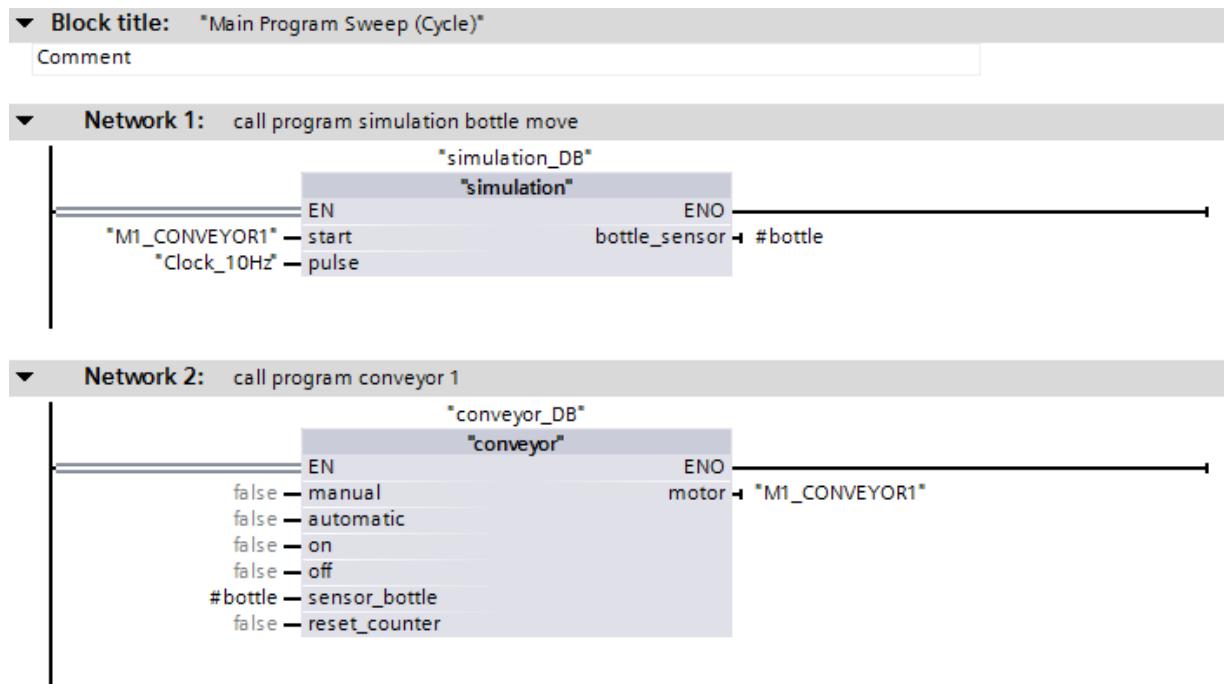
Call the simulation block (FB2) **before** the conveyor block (FB1).

In OB1, set up the "Temp tag" "bottle" and wire the blocks.

Then, save the project and load it to the controller.

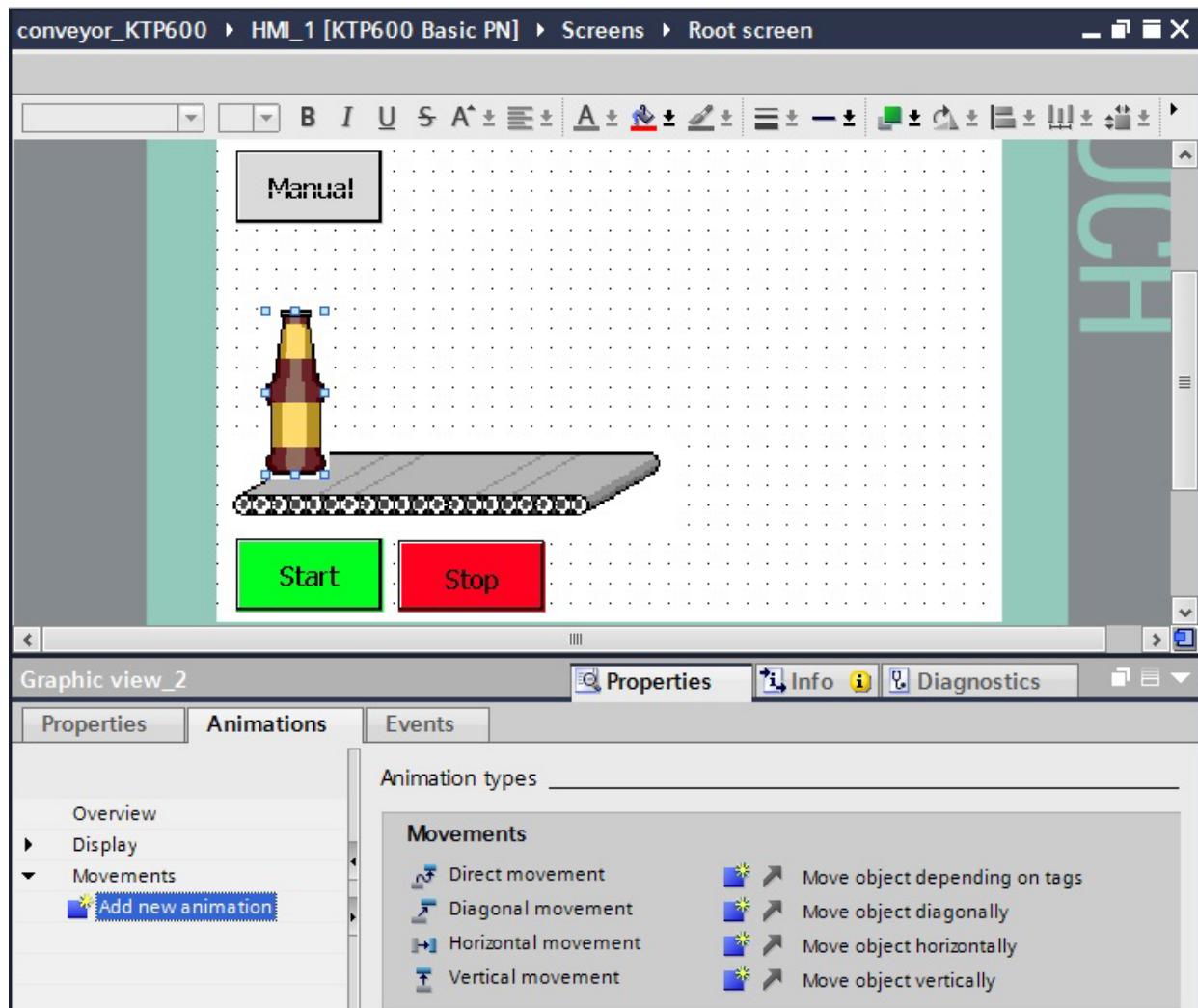


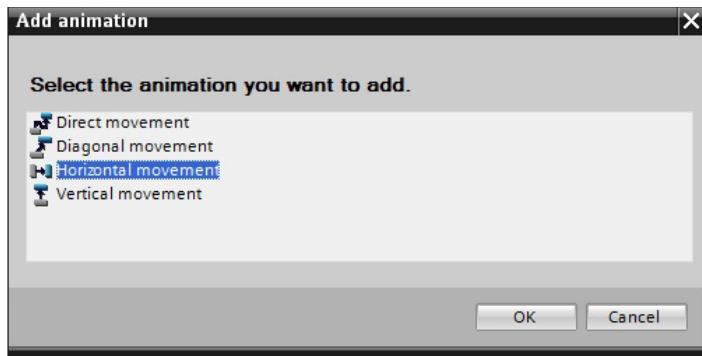
Program in ladder diagram (LAD):



7.12 Configuring the Bottle Movement

Select the bottle and under the tab “Properties/Animations“, select under Movements “Add new animation“.

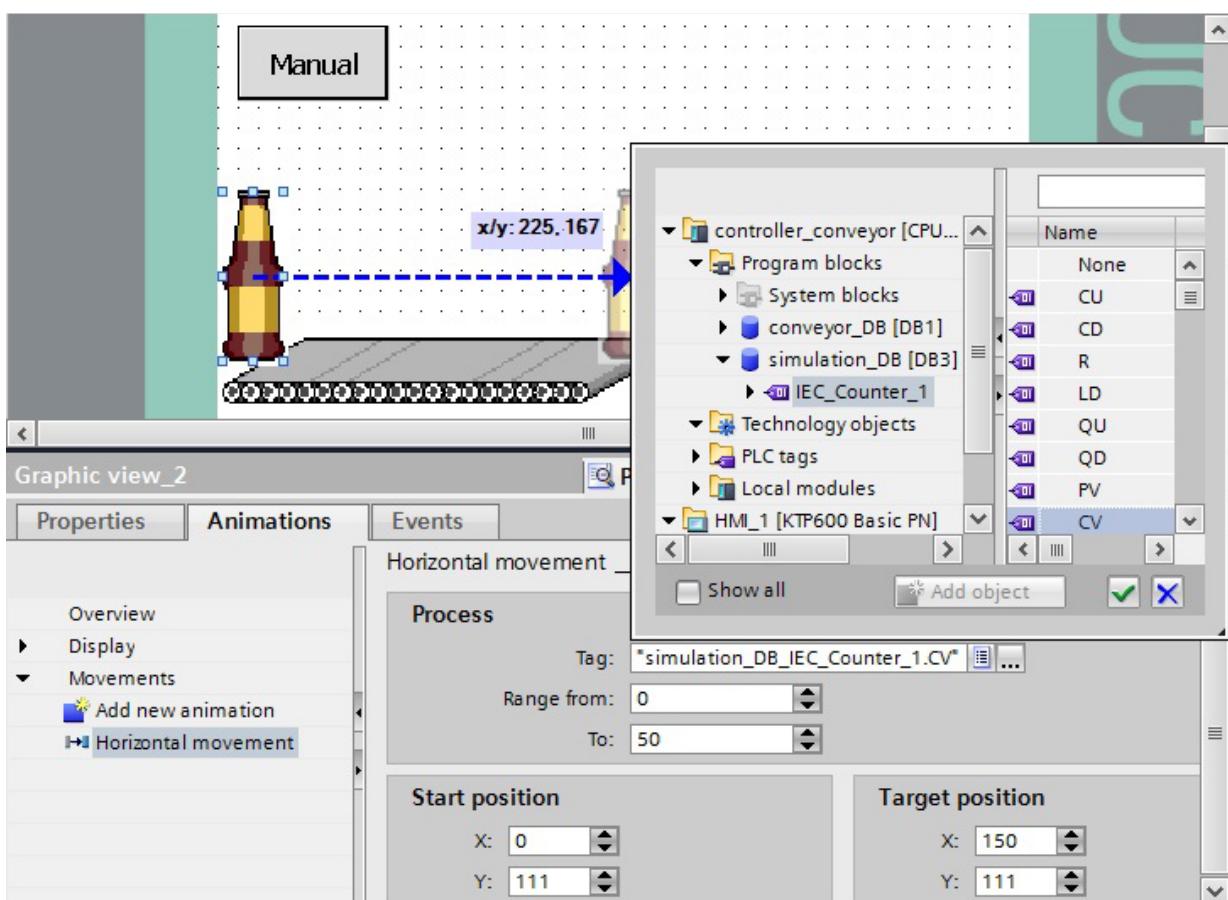




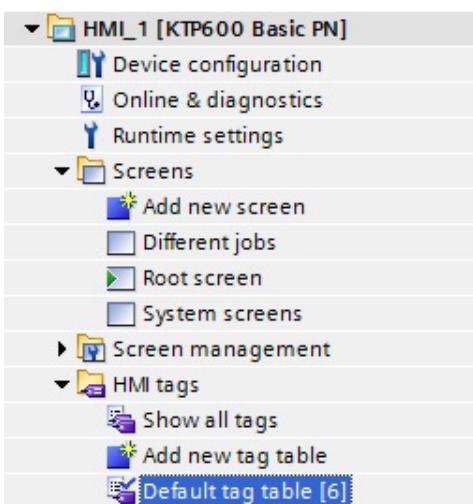
As tag, select "**CV**" of IEC_Counter_1 in the simulation_DB (DB2).

Under range, enter **0 to 50**.

Change the target position to the end of the conveyor **X150**.



In the project window, select the **HMI tags**.



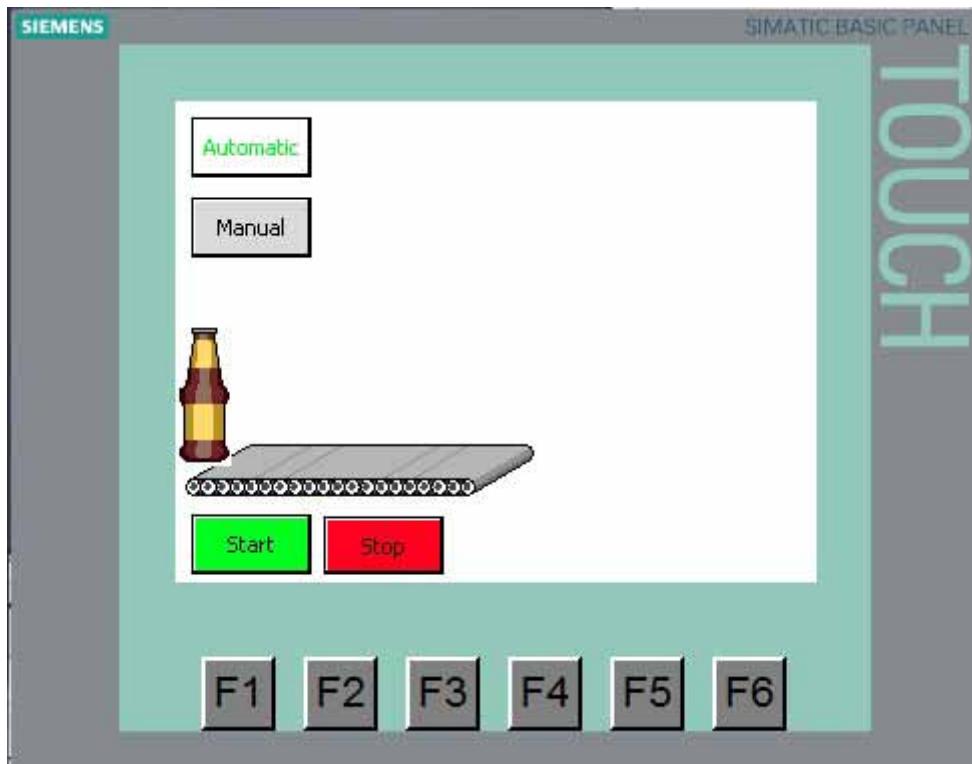
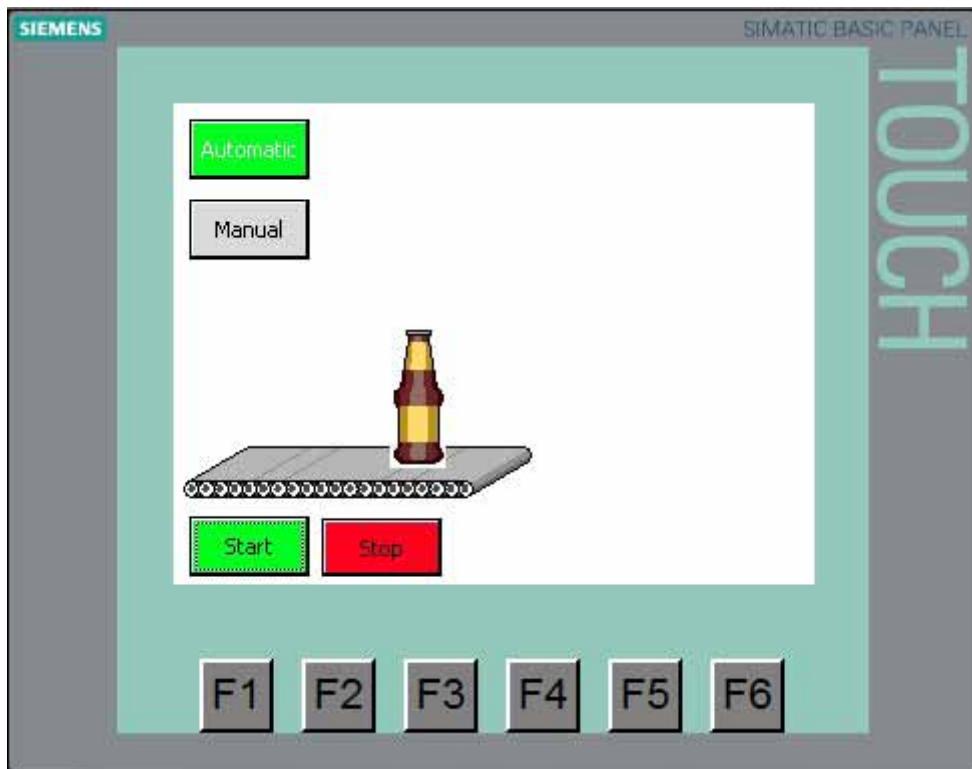
Default tag table

Name	Access mode	Acquisition cycle	Comment
conveyor_DB_automatic	<symbolic access>	100 ms	
conveyor_DB_manual	<symbolic access>	100 ms	
conveyor_DB_mem_automatic	<symbolic access>	100 ms	
conveyor_DB_off	<symbolic access>	100 ms	
conveyor_DB_on	<symbolic access>	100 ms	
simulation_DB_IEC_Counter_1...	<symbolic access>	100 ms	
<Add new>			

A small window is overlaid on the table, showing the 'Cycles' configuration for the acquisition cycle. The slider is set to 100 ms.

Drag the slider in the window towards the right to get to the column Acquisition cycle.
Set the acquisition cycle of the **HMI tags** to **100ms**.

Then, save the project, load it to the panel and test it.

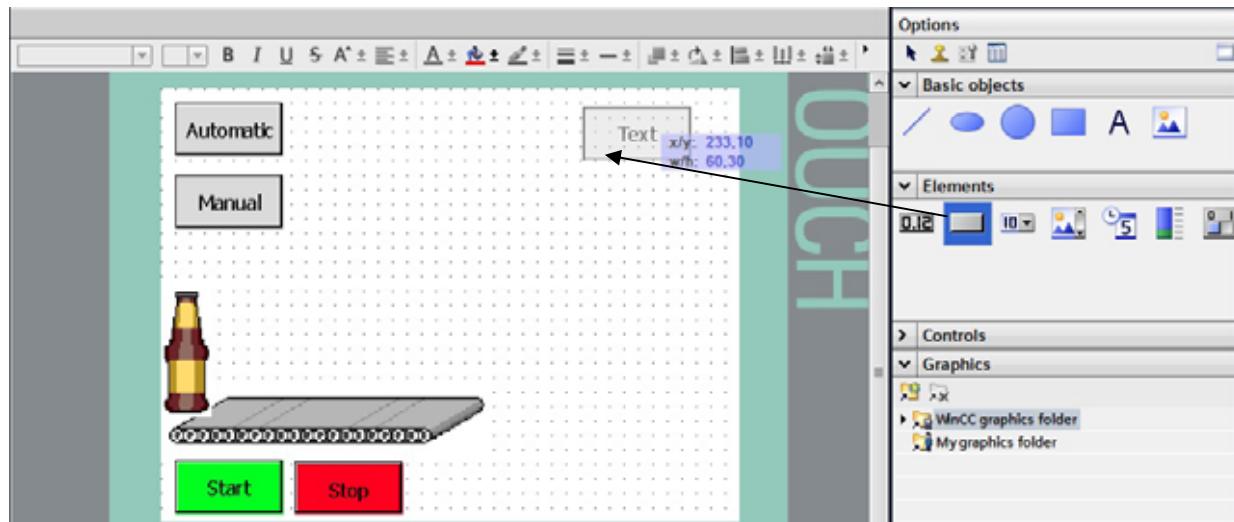


After 20 bottles, the conveyor motor stops.

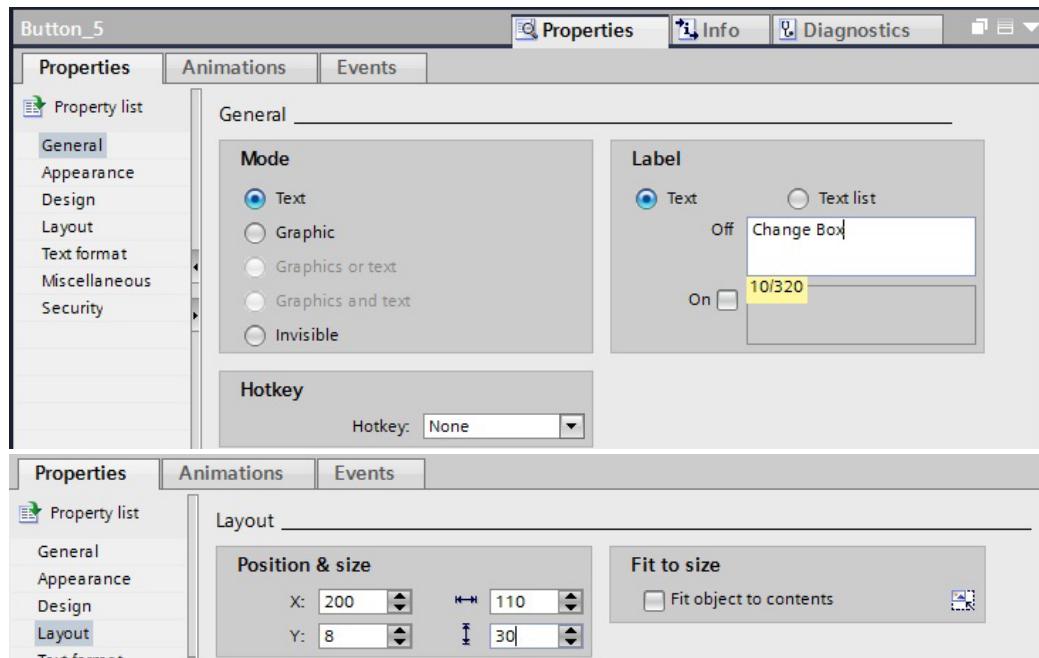
To restart, the bottle counter has to be reset.

7.13 Resetting the Bottle Counter

Drag a button into the basic screen.



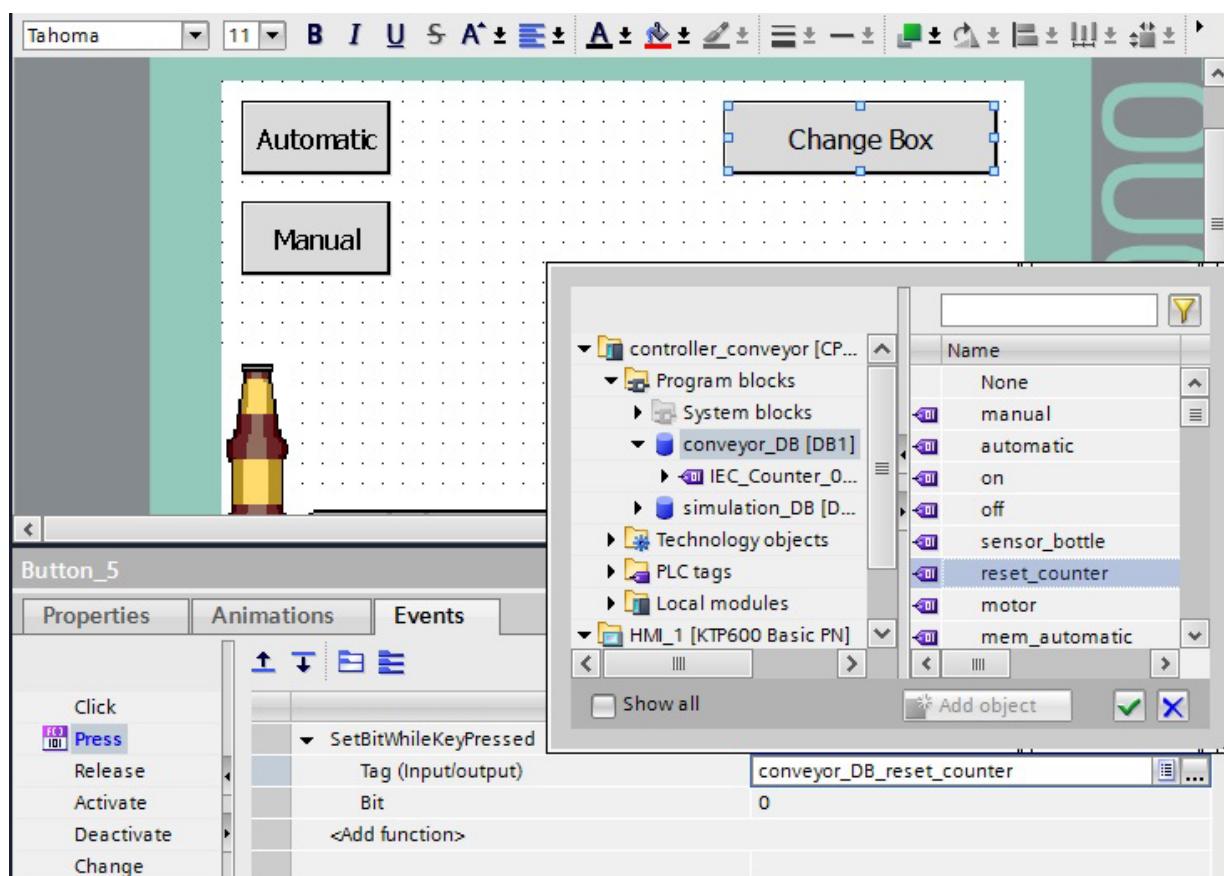
As text, enter “**Change beer case**“ and adjust the color, position and size of the button.



Under Events Press, select under bit editing the function

"SetBitWhileKeyPressed".

Select the tag **"reset_counter"** from **conveyor_DB [DB1]**.



Set the acquisition cycle of the new HMI tag to 100ms.

In OB1, remove the wiring **reset_counter** when calling the conveyor_FB.

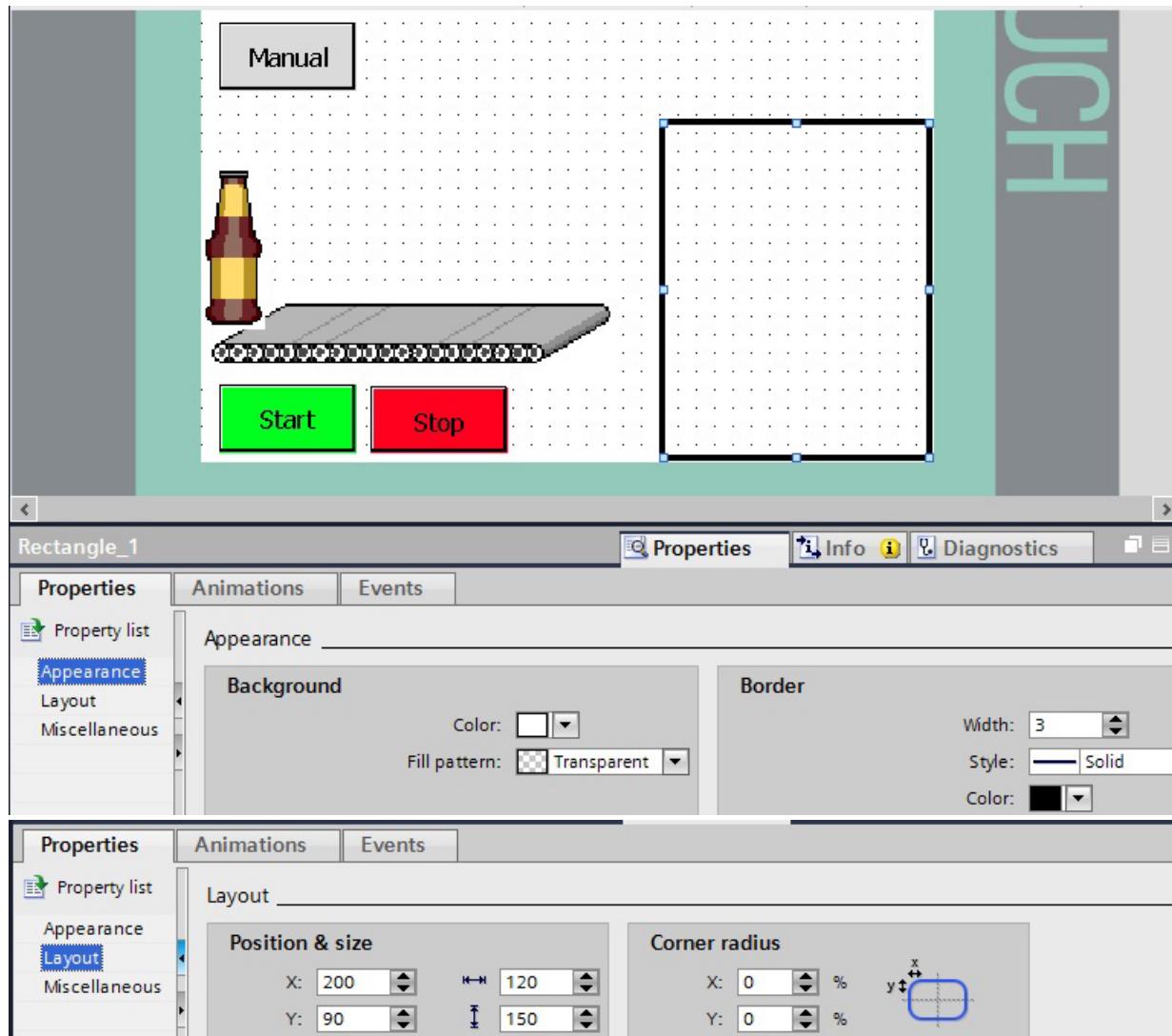
Then save the project, load it to the CPU and the panel, and test it.



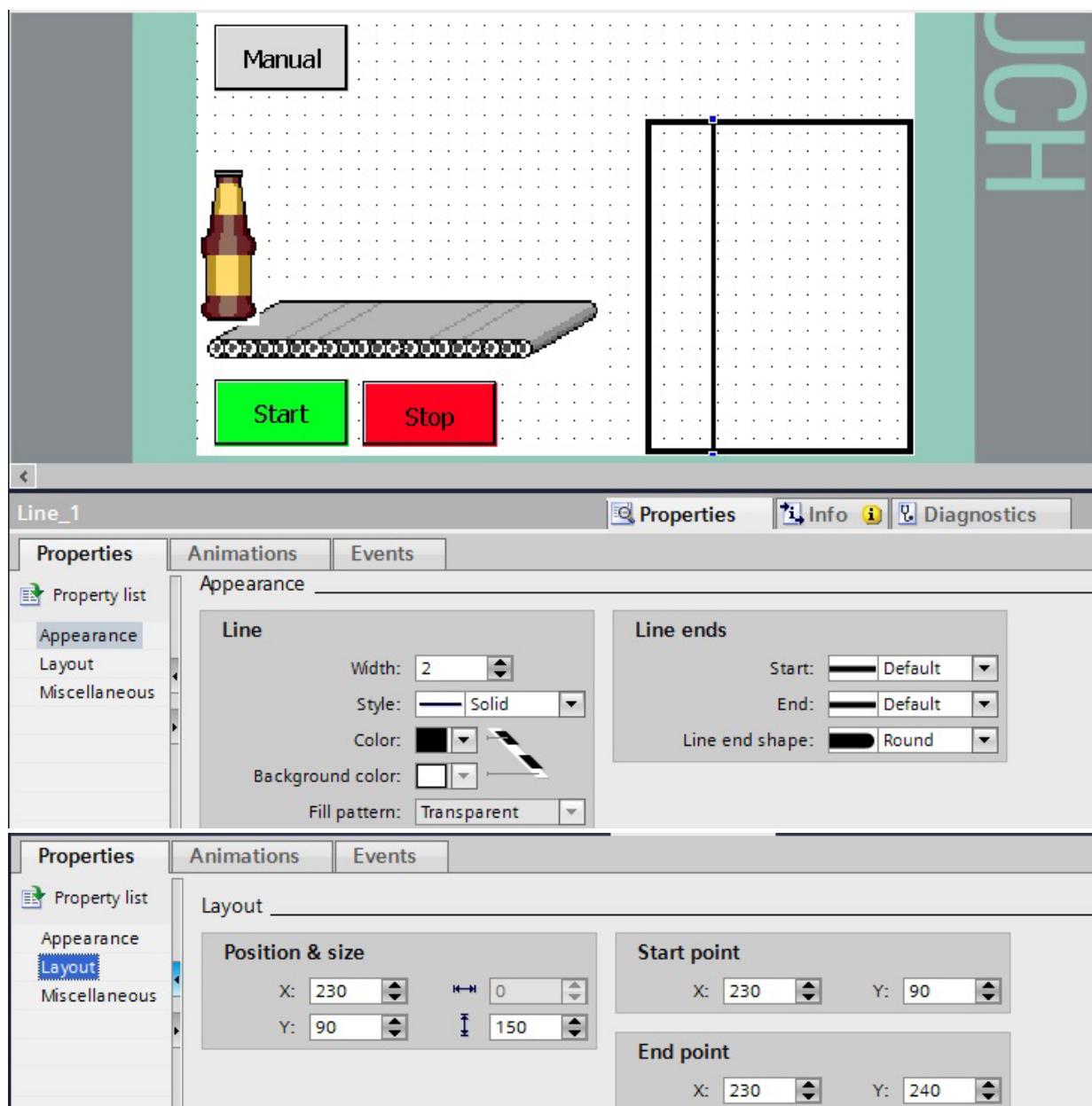
7.14 Drawing the Beer Case

Draw a rectangle with a transparent background.

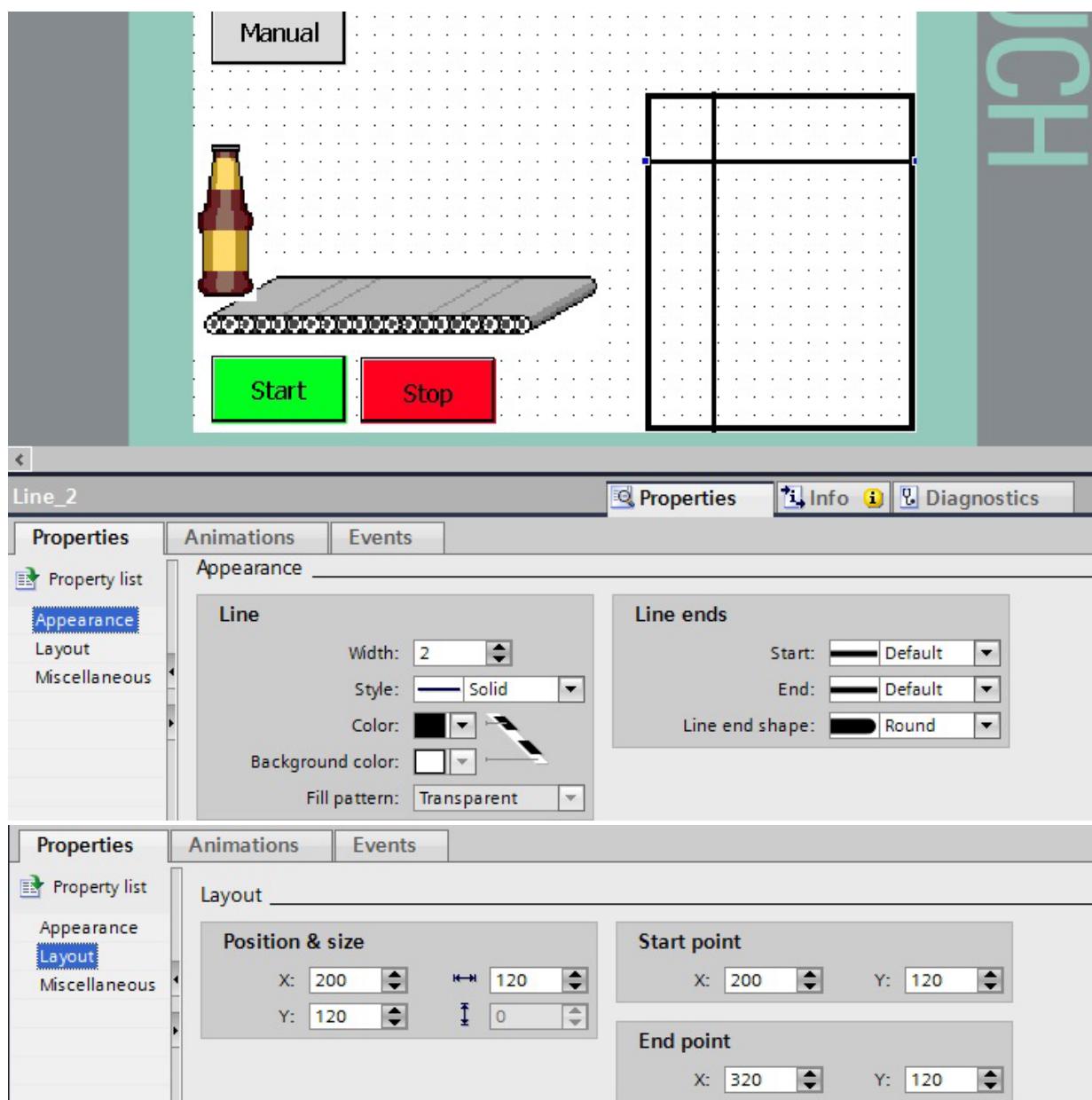
Enter the frame width, the position and the size.



Draw a vertical line with a spacing of 30 pixels.

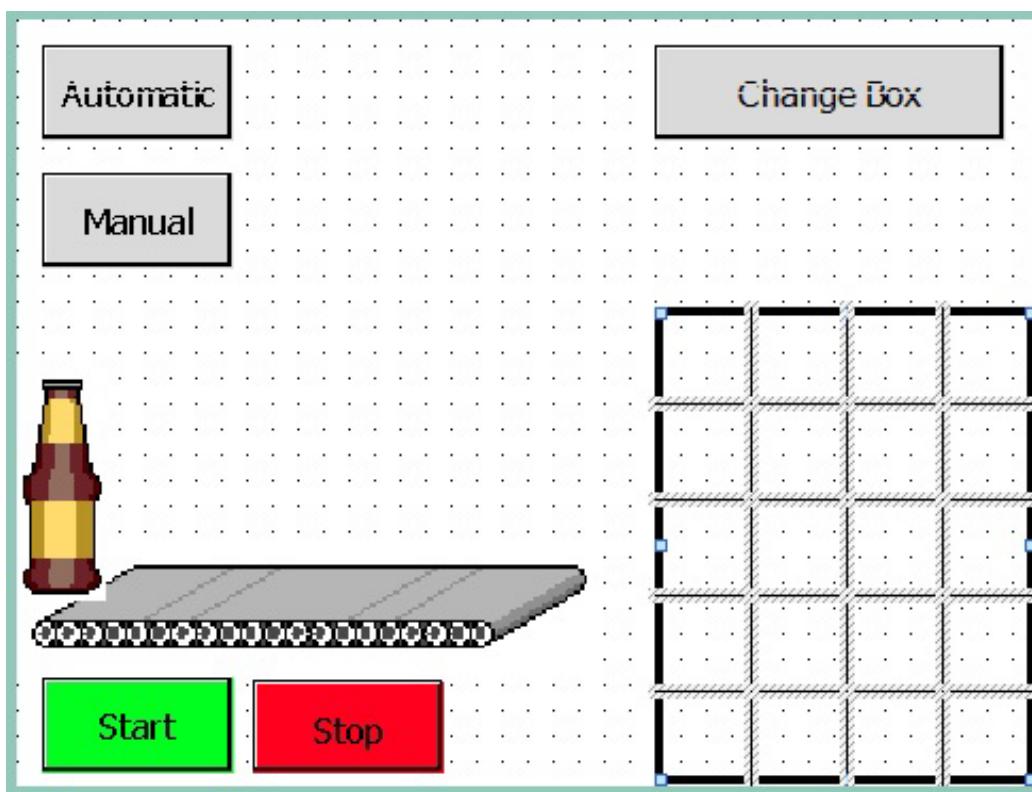


Draw a horizontal line with a spacing of 30 pixels.

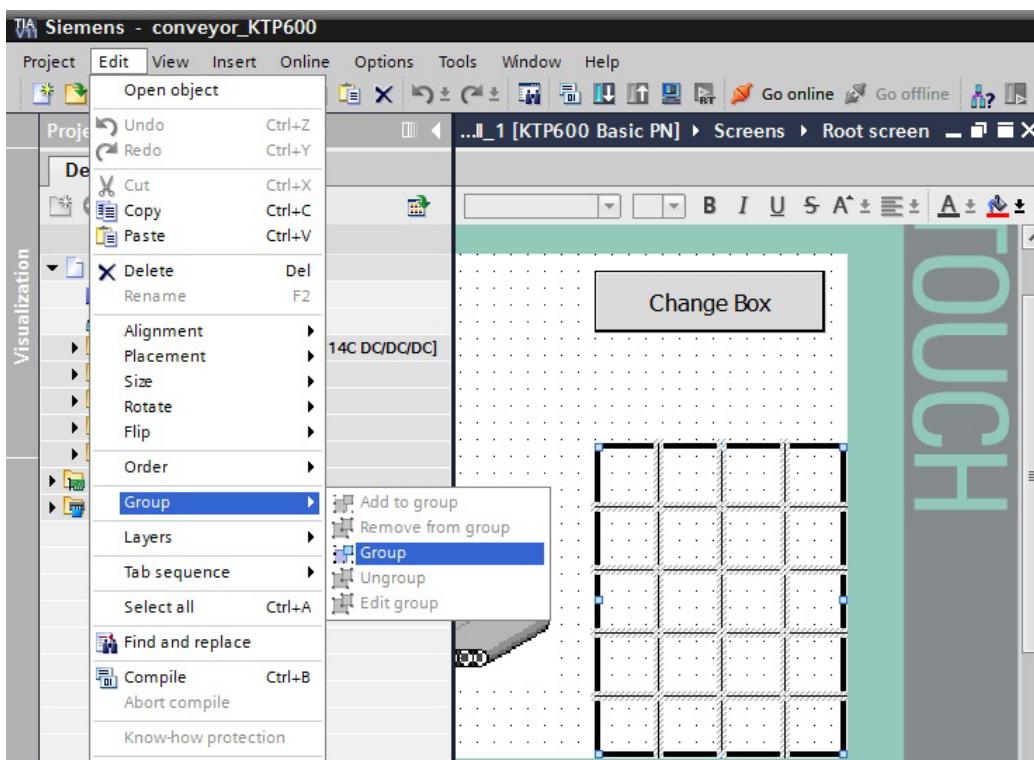


With copying and inserting, add the remaining lines with a spacing of 30 pixels.

Select the beer case by drawing a frame around the case with the mouse.



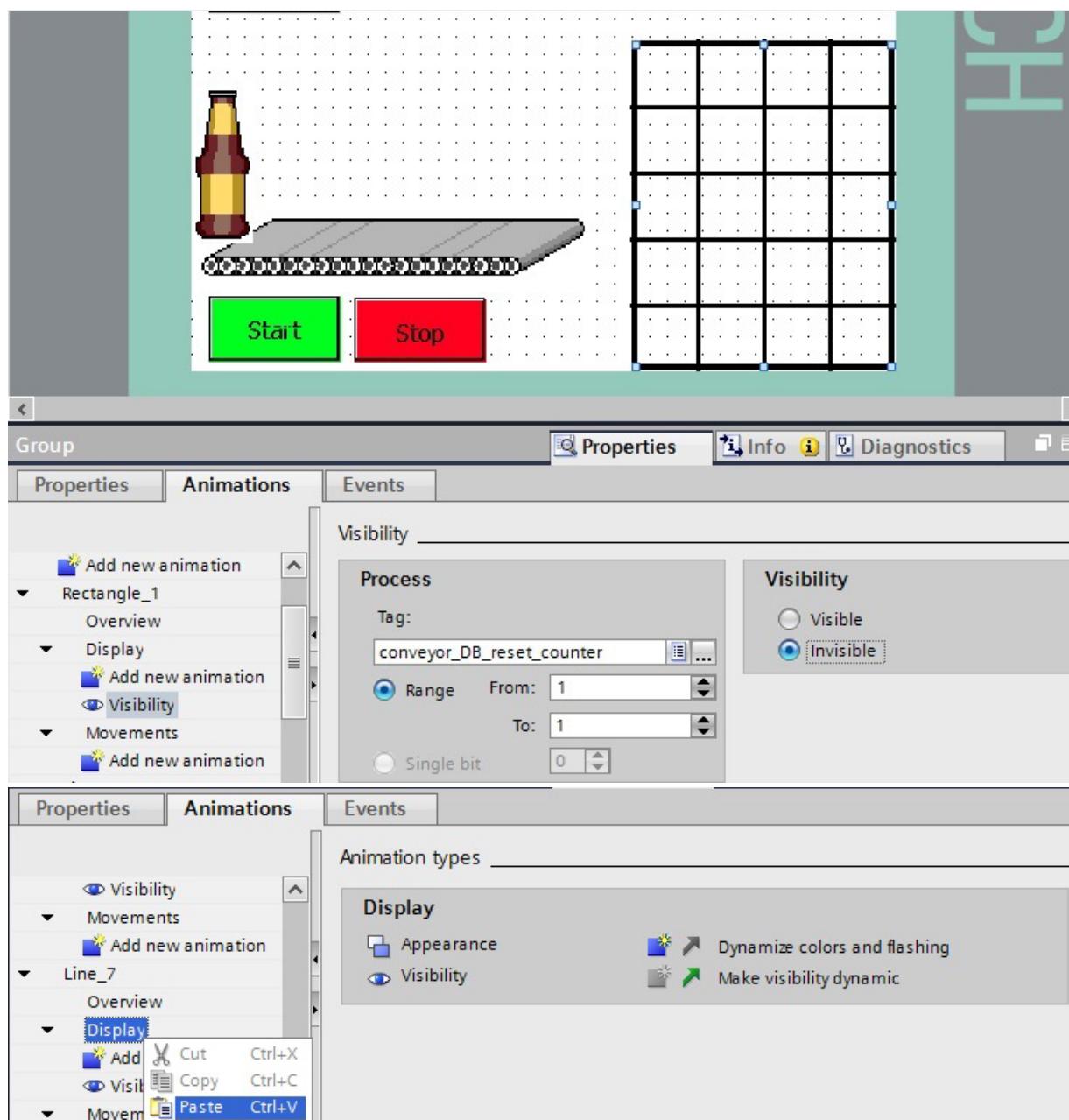
In the menu "Edit", select the function "Group".



The rectangle and the lines are not to be displayed when the beer case is replaced.

At Rectangle_1 and at the lines, create the animation "Visibility" with the tag "conveyor_DB_reset_counter" at Value 1 Invisible.

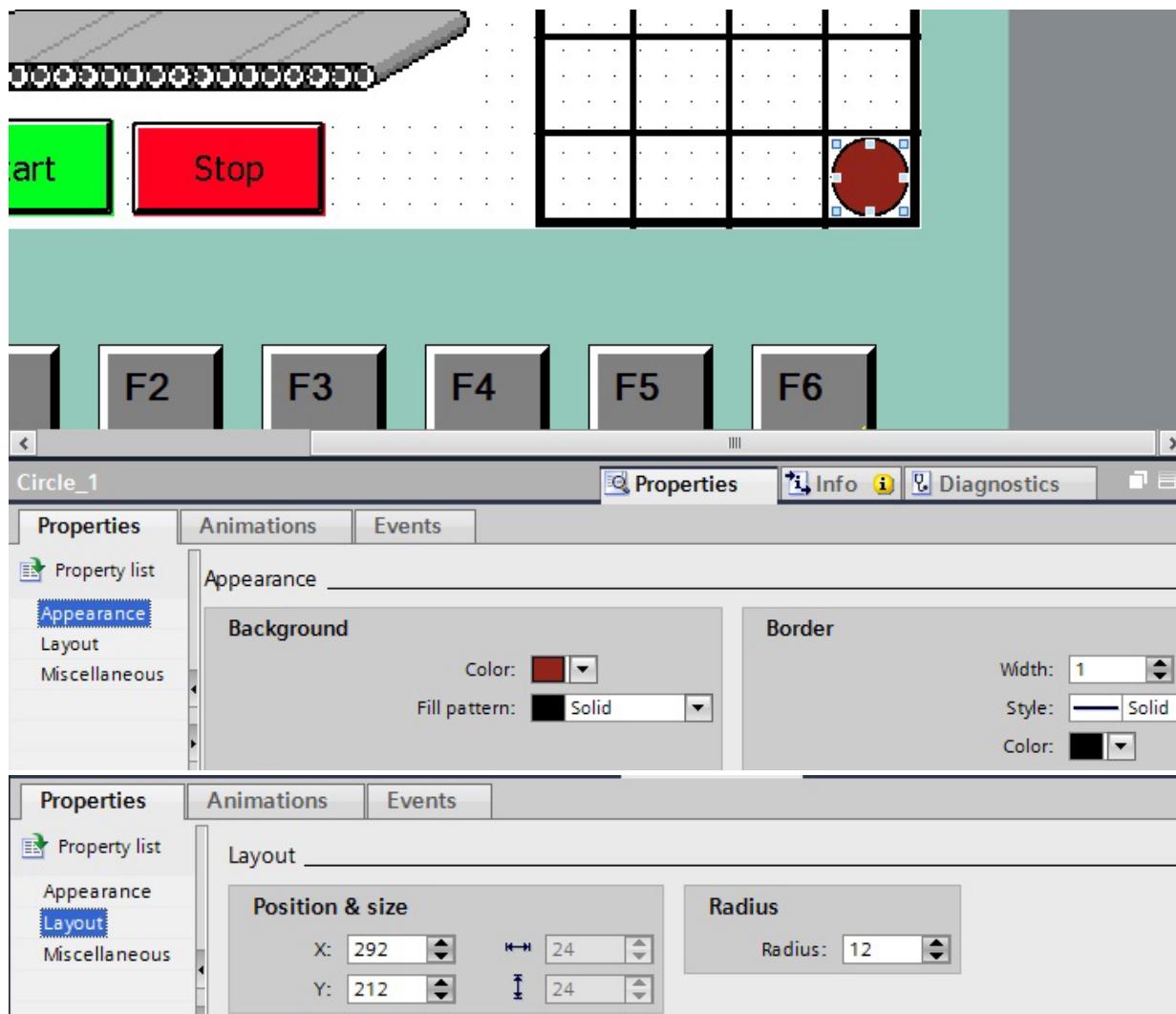
At the lines, the animation can also be copied and inserted.



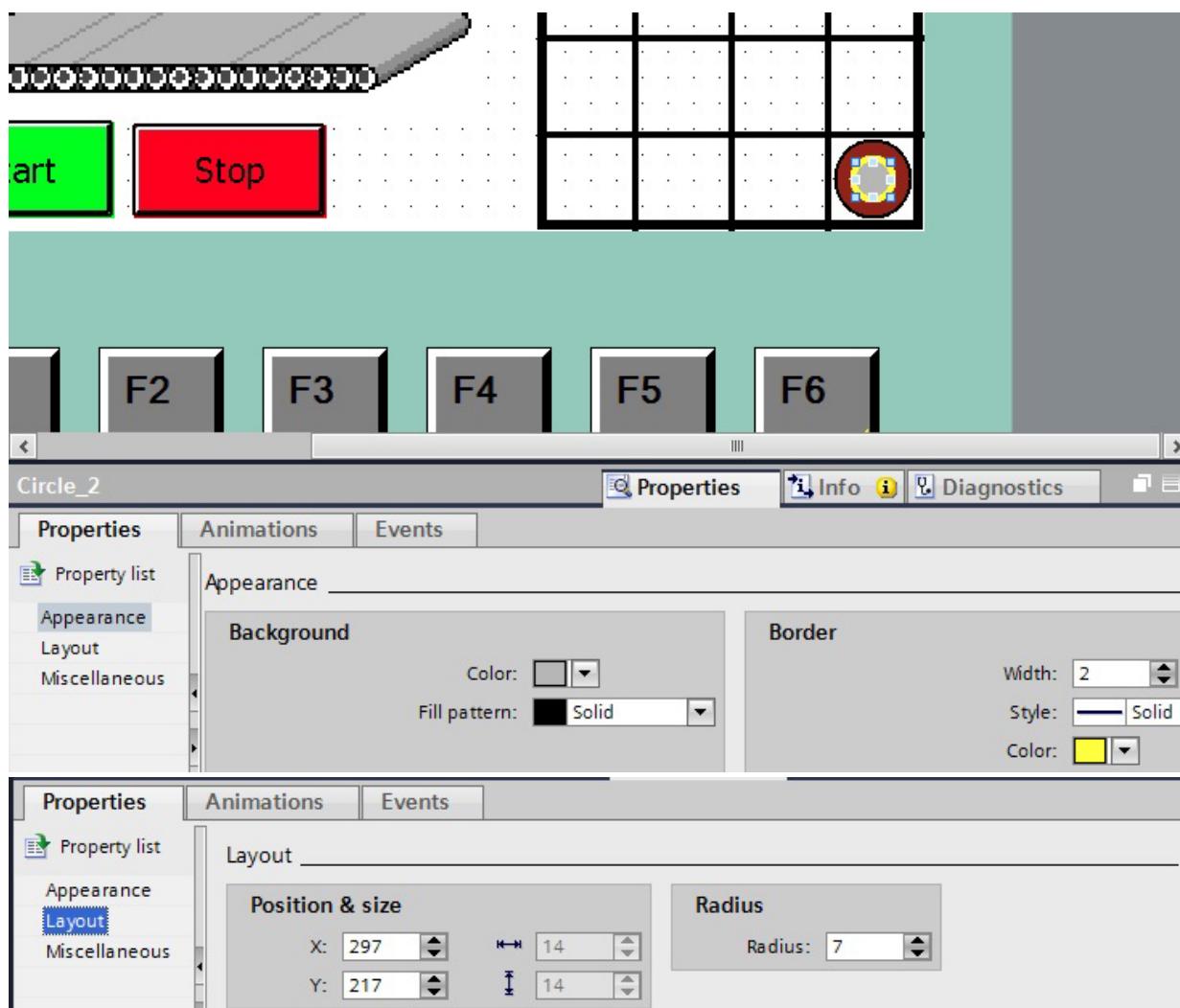
Then, save the project, load it to the panel and test it.

7.15 Drawing the Bottles in the Case

Enlarge the view and draw a circle in the lower right field of the case.

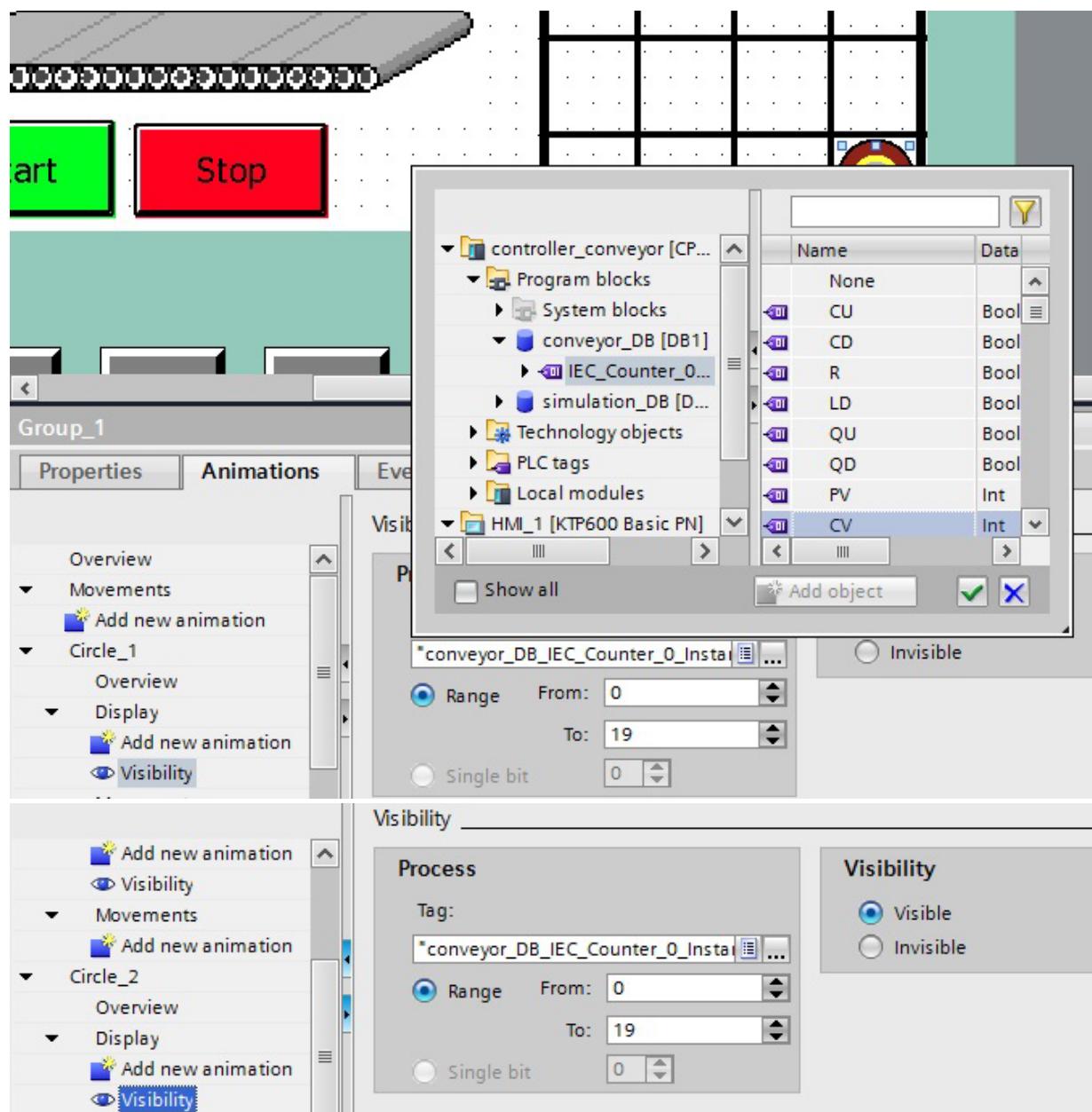


Draw a second circle.



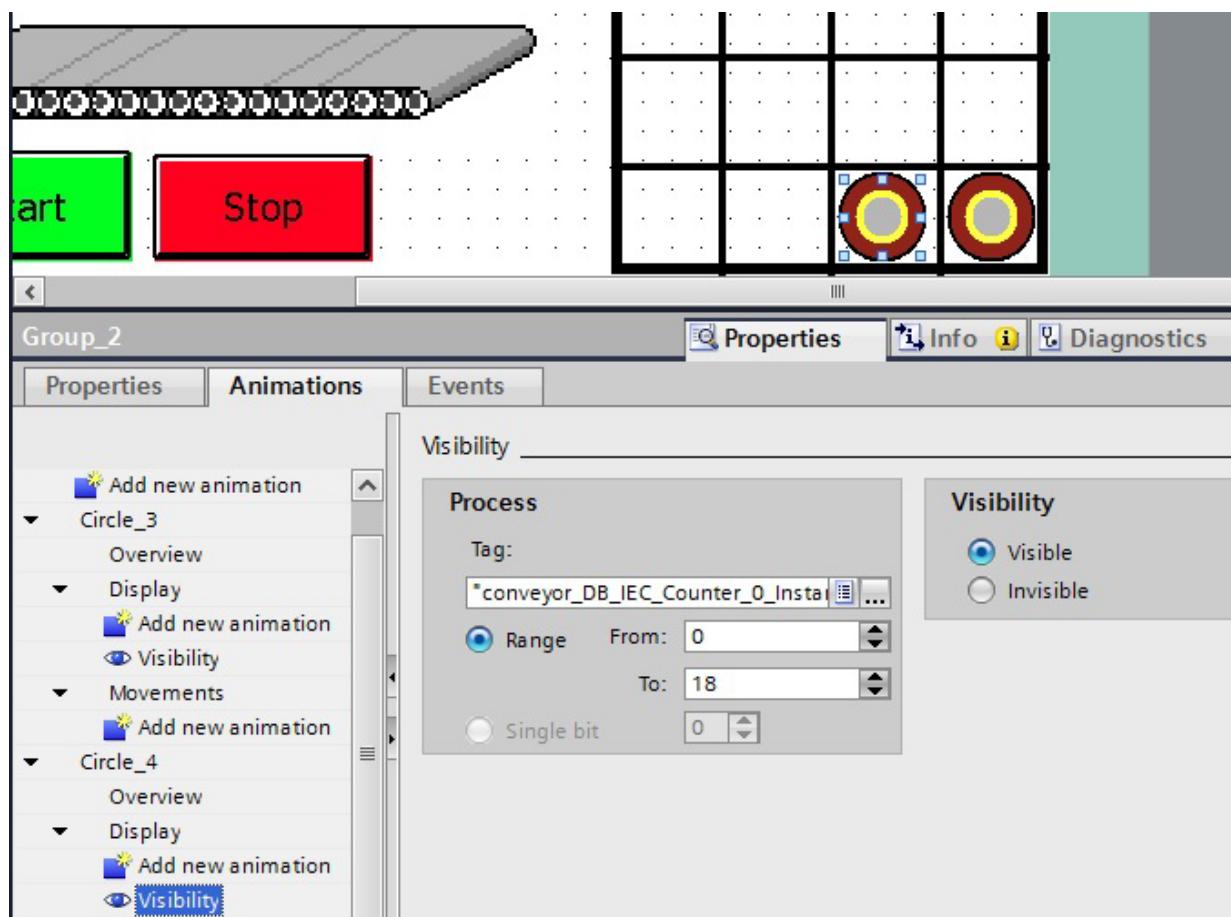
Group the two inserted circles.

At Circle_1 and Circle_2, generate the animation “**Visibility**“ with the tag “**Conveyor_DB_IEC_Counter_0_Instance_CV**“ value range **0 to 19** Visible.



Copy and insert the bottle.

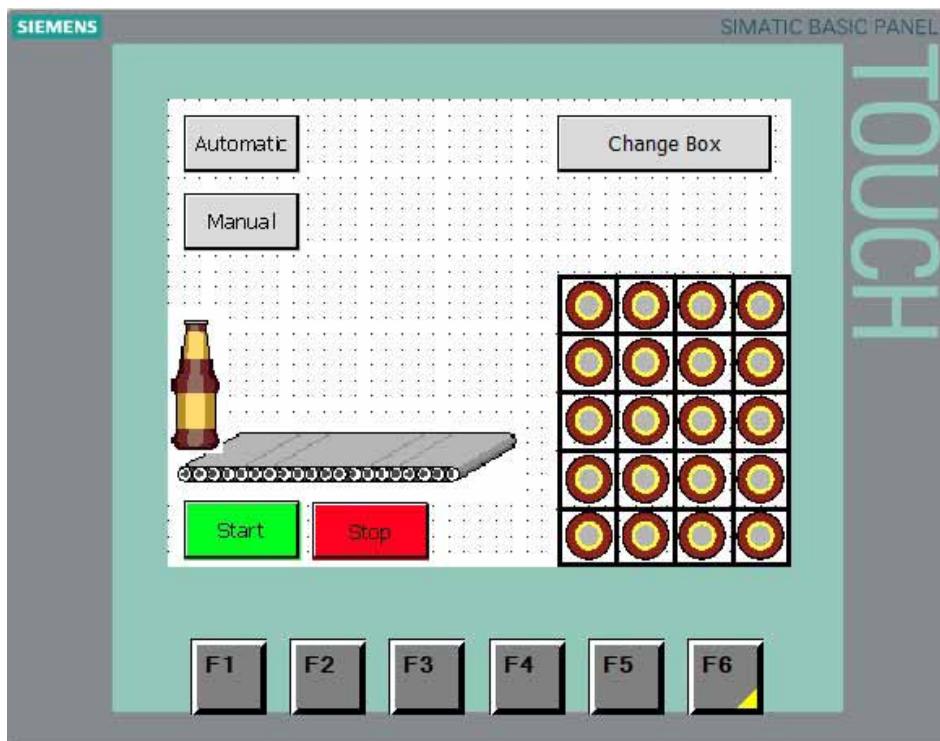
At the two circles under Visibility, change the value range of the tag
"Conveyor_DB_IEC_Counter_0_Instance_CV" to **0 to 18** Visible.



Copy and insert the individual bottles.

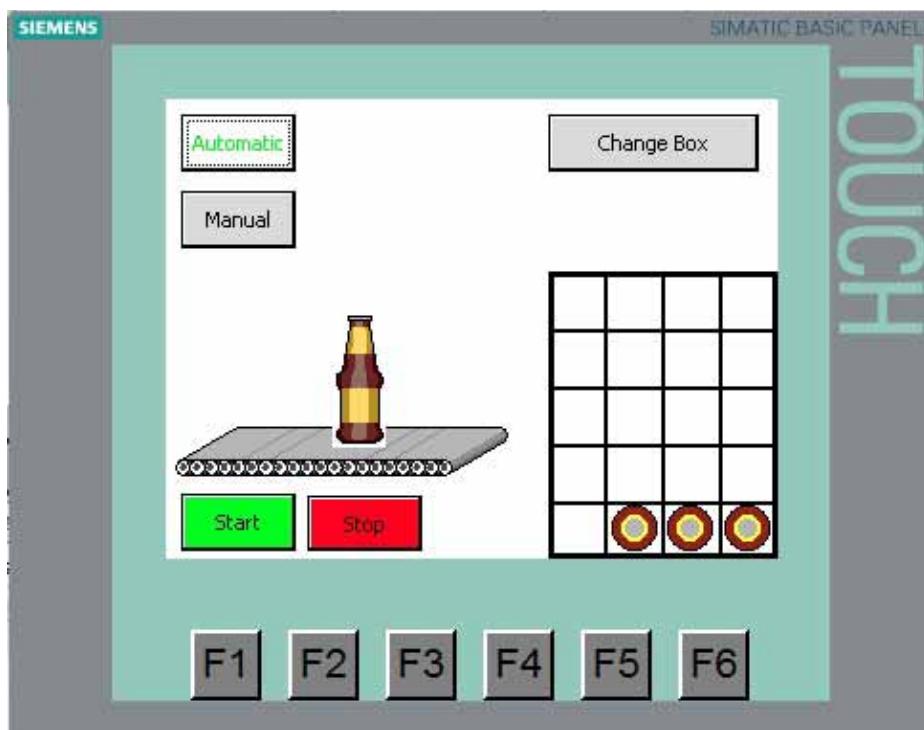
At the animation “**Visibility**” of the two circles, decrease the value at “**to**” by 1.

The last bottle has the value range **0 to 0**.



Set the acquisition cycle of the new **HMI tag** to **100ms**.

Then save the project, load it to the panel and test it.



Automation System SIMATIC S7-1200

SCE Training Curriculum for Integrated Automation Solutions

Totally Integrated Automation (TIA)

TIA Portal Module 010-090



TIA Portal Module 010-010
Startup Programming of the
SIMATIC S7-1200

TIA Portal Module 010-020
Block Types for the SIMATIC S7-1200

TIA Portal Module 010-030
IEC Timers and IEC Counters at the
SIMATIC S7-1200

TIA Portal Module 010-040
Diagnosis and Error Search at the
SIMATIC S7-1200

TIA Portal Module 010-050
Analog Value Processing with
SIMATIC S7-1200

TIA Portal Module 010-060
Control Engineering with
SIMATIC S7-1200

TIA Portal Module 010-070
Communication between two
SIMATIC S7-1200

TIA Portal Module 010-080
HMI Panel KTP600 with SIMATIC
S7-1200

TIA Portal Module 010-090
Startup – High Level Language
Programming with S7-SCL and
SIMATIC S7-1200

Matching SCE Training Packages for these Manuals

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
Order No.: 6ES7214-1BE30-4AB3
- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
Order No.: 6ES7214-1AE30-4AB3
- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
Order No.: 6ES7822-0AA01-4YE0

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

Continued Training

For regional Siemens SCE continued training, please contact your regional SCE contact person
siemens.com/sce/contact

Additional information regarding SCE

siemens.com/sce

Information regarding Usage

The training curriculum 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 education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this document and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by 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.

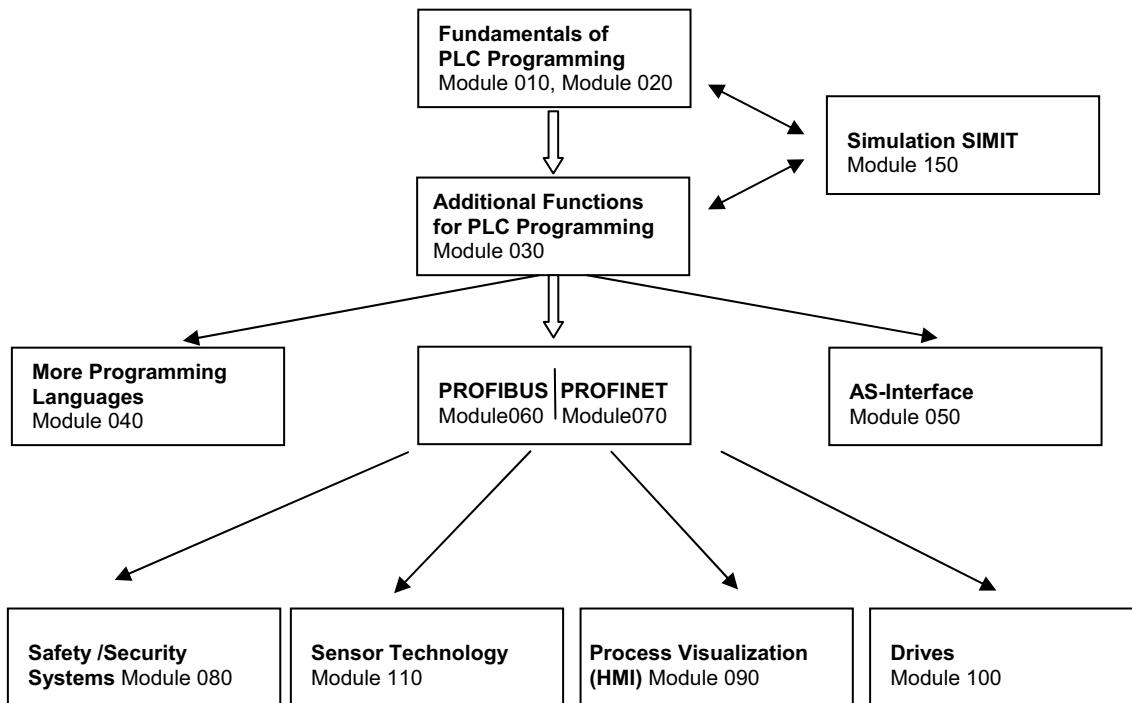
Usage for industrial customer courses is explicitly not permitted. We do not consent to the documents being used commercially.

We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this document.

	PAGE
1. Preface	4
2. Instructions regarding the Programming Language S7 SCL	6
3. S7-SCL Development Environment	7
4. Sample Task Tank Content	8
4.1 Task Description	8
4.2 Assignment List/Tag Table.....	8
4.3 Program Structure.....	9
4.4 Interface of the Block calculate_volume_tank [FC140].....	10
4.5 Note on the solution	11
5. Programming the Tank Content Calculation for the SIMATIC S7-1200 in S7-SCL.....	12
5.1 Setting Up the Project and Configuring the Hardware.....	12
5.2 Setting Up the Program.....	15
5.3 Testing the Program.....	23
5.4 Expanding the Program	26

1. Preface

Regarding its content, module 010-090 is part of the training unit '**Basics of PLC Programming**' and provides a **quick entry point** for programming the SIMATIC S7-1200 using the programming language **S7 SCL** with the TIA portal.



Training Objective

In this module 010-090, the reader is introduced to the basic functions of the S7 SCL development environment. In addition, test functions for the removal of logical programming errors are shown.

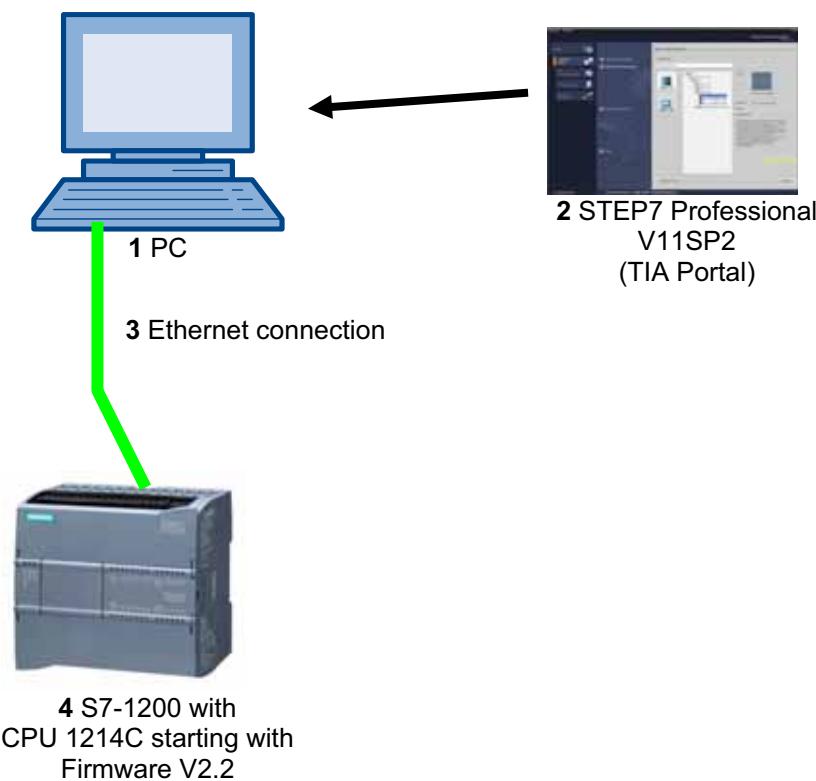
Prerequisites

To successfully work through this module 010-090, the following knowledge is assumed:

- How to handle Windows
- Basics of PLC programming with STEP7 Professional V11
(for example, Modules 010-010 to 010-040)
- Basic knowledge of high level language programming such as Pascal

Required hardware and software

- 1** PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB
Operating system Windows XP Professional SP3/Windows 7 Professional/Windows 7 Enterprise/
Windows 7 Ultimate/Windows 2003 Server R2/Windows Server 2008
Premium SP1, Business SP1, Ultimate SP1
- 2** Software STEP7 Professional V11 SP2 (Totally Integrated Automation (TIA) Portal V11)
- 3** Ethernet connection between PC and CPU 1214C
- 4** PLC SIMATIC S7-1200 starting with Firmware V2.2; for example, CPU 1214C with 14DI/10DO
signal module.
The inputs have to be brought out to a panel.



2. Instructions regarding the Programming Language S7 SCL

S7 SCL (Structured Control Language) is a higher level programming language that is based on PASCAL and makes structured programming possible. The language corresponds to the sequential language SFC "Sequential Function Chart" specified in the standard DIN EN-61131-3 (IEC 61131-3). In addition to high level language elements, S7-SCL also includes typical PLC elements such as inputs, outputs, timers, flags, block calls, etc. as language elements. It supports the STEP 7 block concept and -in addition to STL, LAD and FBD- allows for standard-conforming programming of blocks. That means, S7-SCL supplements and expands the programming software STEP 7 with its programming languages LAD, FBD and STL.

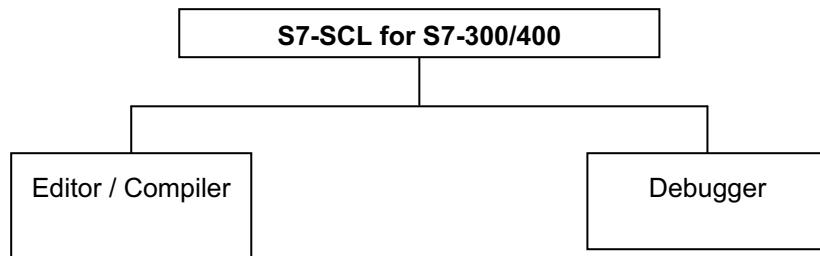
You don't have to generate each function yourself; preassembled blocks such as system functions or system function blocks are provided that are available in the CPU's operating system.

Blocks that are programmed with S7-SCL can be mixed in with STL, LAD and FBD blocks. This means that a block programmed with S7-SCL is able to call another block that is programmed in STL, LAD or FBD. Correspondingly, S7-SCL blocks can be called in STL, LAD and FBD programs.

The S7-SCL test functions make possible the search for logical programming errors in a faultless compilation.

3. S7-SCL Development Environment

To use S7-SCL, a development environment is provided that is harmonized with specific S7-SCL characteristics as well as with STEP 7. This development environment consists of an Editor/Compiler and a Debugger.



Editor/Compiler

The S7-SCL editor is a text editor that can be used to edit any text. The central task that you will be performing is the generation and editing of blocks for STEP 7 programs. During the input, a basic syntax check is performed which simplifies faultless programming. Syntax errors are shown in different colors.

The Editor offers the following:

- Programming an S7 block in the language S7-SCL
- Convenient insertion of language elements and block calls with drag&drop
- Direct syntax check during programming.
- Setting the editor according to your requirements; for example, through syntax oriented coloring of the different language elements
- Checking the completed block through compilation
- Display of all errors and warnings that occur during compilation
- Identifying the faulty location in the block, optionally with error description and information regarding error removal

Debugger

The S7-SCL debugger allows for testing a program during its run in the AS and thus locates possible logical errors.

To this end, the S7-SCL provides two different test modes:

- Step by step monitoring
- Continuous monitoring

In the case of “step by step monitoring“, the logical program sequence is traced. You can execute the program algorithm instruction by instruction, and monitor in a result window how the tag contents that are being edited change.

With “continuous monitoring“ we test a group of instructions within a block. During the test run, the values of the tags and the parameters are shown in a chronological sequence and -to the extent possible- updated cyclically.

4. Sample Task Tank Content

4.1 Task Description

For our first program, we are programming the calculation of the contents of a tank.

The tank has the shape of an upright cylinder. The level of the content is measured with an analog sensor. In the task, the value of the level is already available, standardized in the unit 'Meter'.

The program is to be programmed in a function FC140 'calculate_volume_tank'. Transfer parameters are the diameter and the level in the unit Meter. The result is the content of the tank in the unit Liter.

4.2 Assignment List/Tag Table

Since for modern programming, tags are used rather than absolute addresses, first the **global PLC tags** have to be specified.

These global PLC tags are descriptive names with comments for those inputs and outputs that are used in the program. Later, during programming, we can access the global PLC tags by means of this name. These global tags can be used in the entire program in all blocks.

Default Tag Table

Name	Data type	Address	Comment
filling_level_tank1	REAL	%MD40	filling level of tank1 (meter)
diameter_tank1	REAL	%MD44	diameter tank1 (meter)
volume_tank1	REAL	%MD48	volume tank1 (liter)

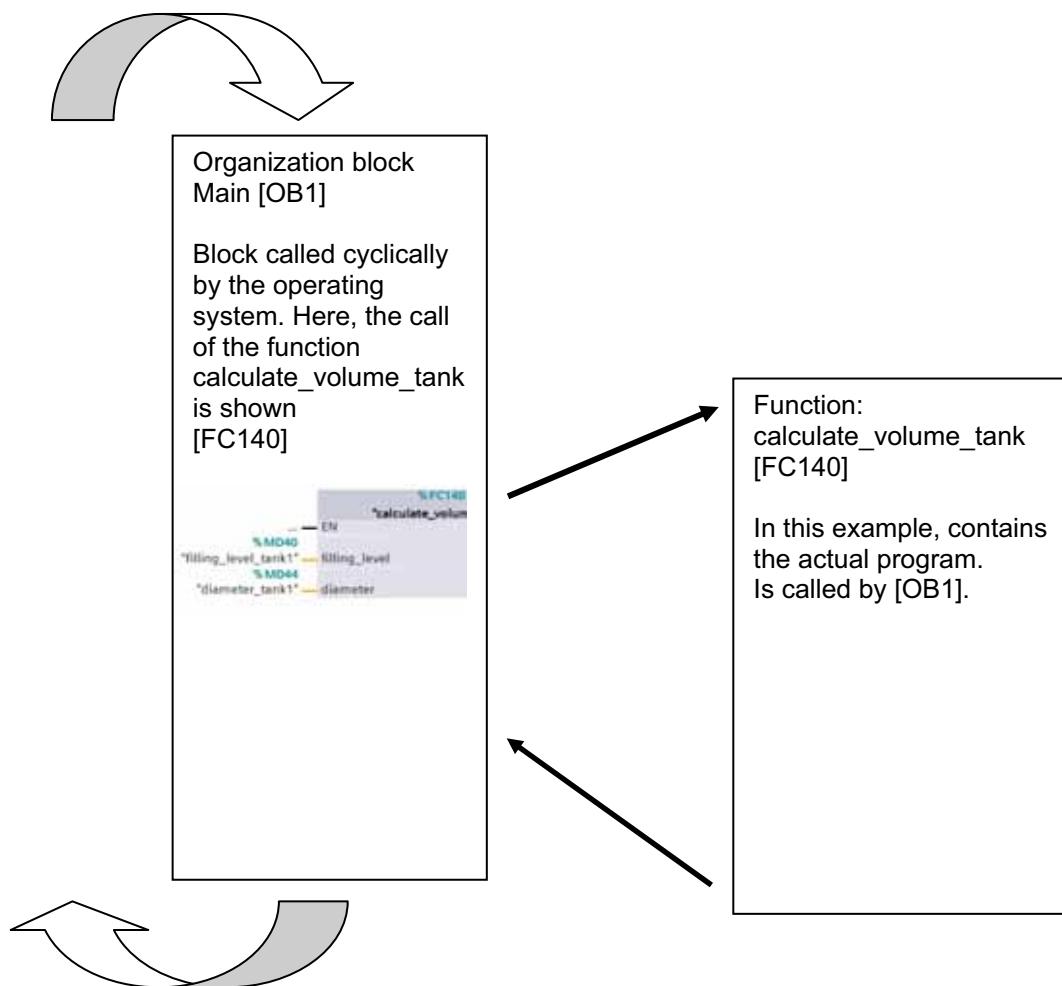
4.3 Program Structure

The program structure is written in socalled blocks. As standard, the organization block Main [OB1] is already there. It represents the interface to the CPU's operating system which calls this block automatically and processes it cyclically.

From this organization block, additional blocks can be called in turn for structured programming, such as the function calculate_volume_tank [FC140].

The purpose is breaking down an overall task into partial tasks; they can then be solved and tested more easily in their functionality.

Structure of the Sample Task Tank Content



4.4 Interface of the Block calculate_volume_tank [FC140]

Before the program can be written, the interface of the block has to be declared. When the interface is declared, only the local tags that are known in this block are specified.

The tags or interface parameters consist of two groups:

- Block parameters that generate the interface of the block for the call in the program.

Type	Name	Function	Available in
Input parameter	Input	Parameters whose values the block reads.	Functions, function blocks and some types of organization blocks
Output parameter	Output / Return	Parameters whose values the block writes.	Functions and function blocks
In/out parameter	InOut	Parameters whose value the block reads when called, and after processing writes to the same parameter.	Functions and function blocks

- Local data used for storing intermediate results.

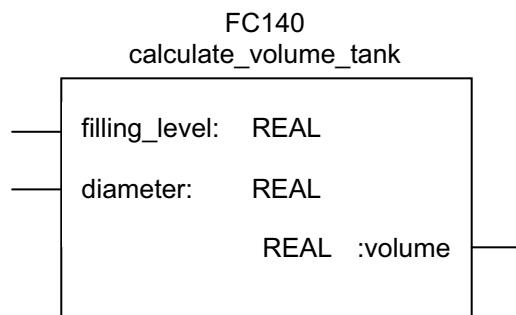
Type	Name	Function	Available in
Temporary local data	Temp	Tags that are used to store temporary intermediate results. Temporary data is maintained for one cycle only.	Functions, function blocks and organization blocks
Static local data	Static	Tags that are used for storing static intermediate results in the instance data block. Static data is retained -also over several cycles- until it is written again.	Function blocks

The interface parameters used in our sample program for the block 'calculate_volume_tank [FC140] are:

Interface of Function FC140:
calculate_volume_tank

Type	Name	Data type	Comment
IN	filling_level	REAL	filling level tank (meter)
IN	diameter	REAL	diameter tank (meter)
OUT	volume	REAL	volume of liquid in the tank (liter)/ -1 if error

The function call in a block programmed in FBD looks like this:

Function call: Representation in FBD


4.5 Note on the solution

To solve the task, the formula for volume calculation of an upright cylinder is used. The conversion factor 1000 is used to calculate the result in liters.

$$V = \frac{d^2}{4} \cdot \pi \cdot h \quad => \quad \text{volume} = \frac{\text{diameter}^2}{4} \cdot 3.14159 \cdot \text{filling_level} \cdot 1000$$

5. Programming the Tank Content Calculation for the SIMATIC S7-1200 in S7-SCL

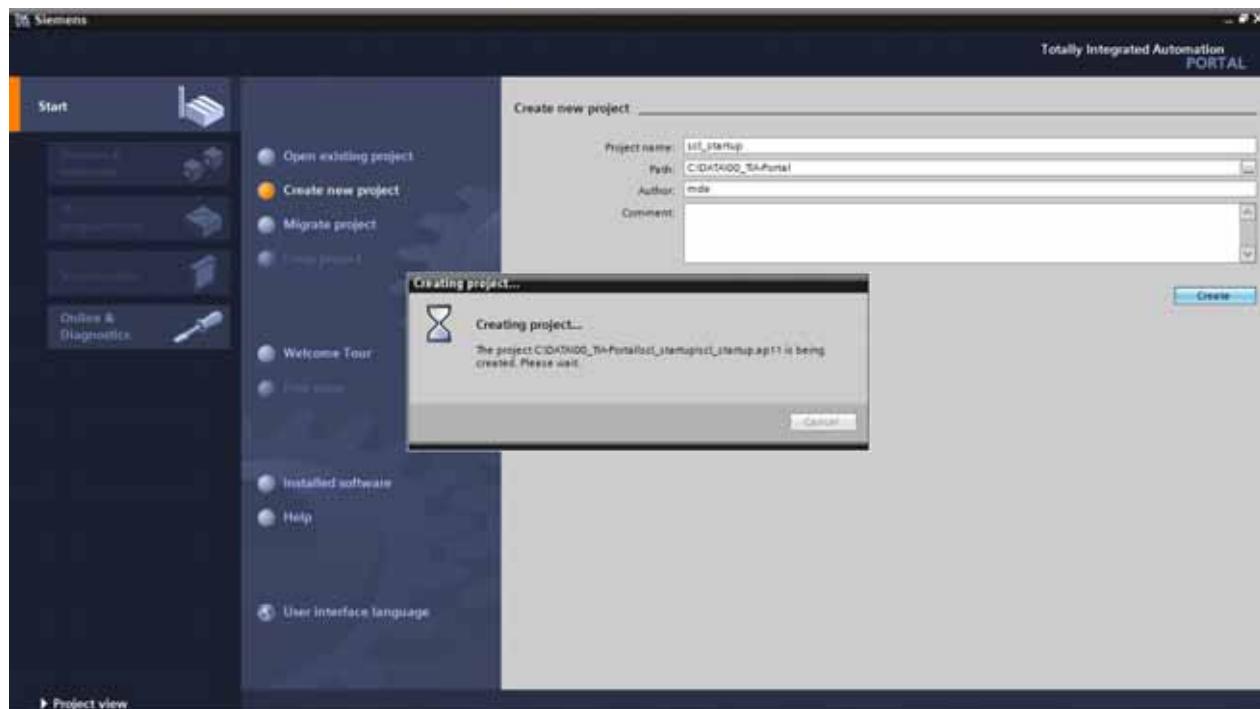
In the steps below, a project can be set up for the SIMATIC S7-1200 and the solution to the task can be programmed.

5.1 Setting Up the Project and Configuring the Hardware

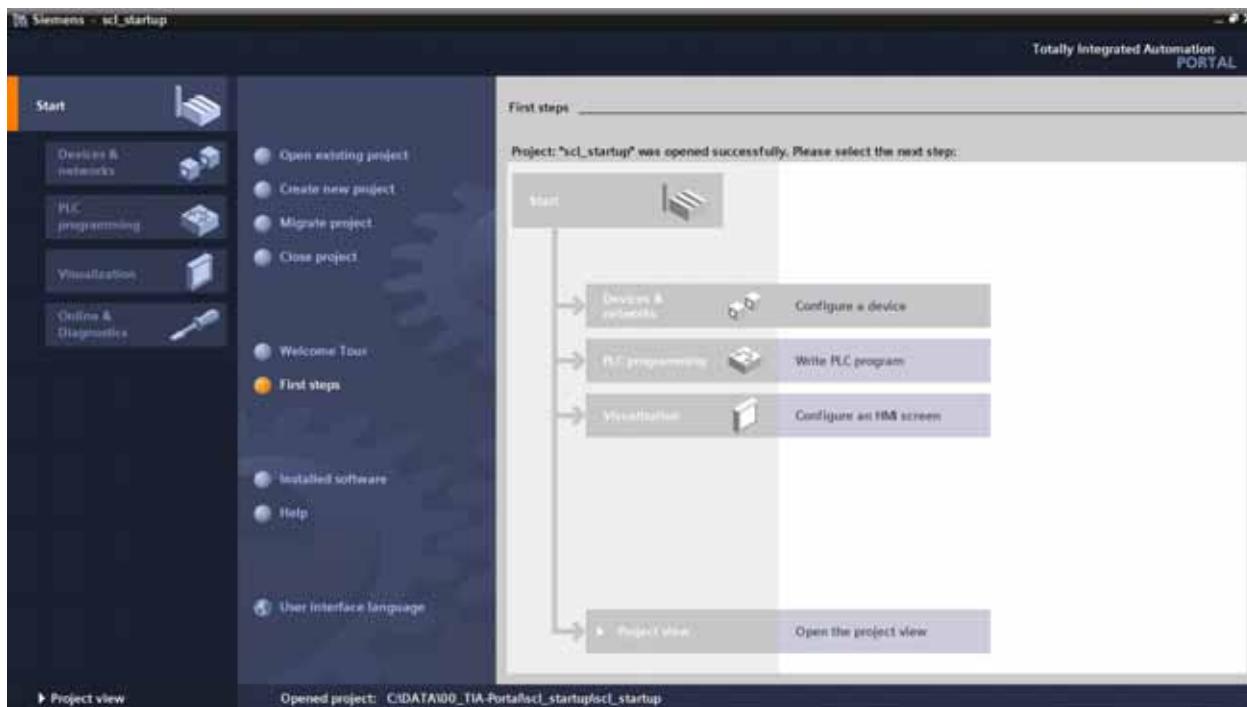
1. The central tool is the '**Totally Integrated Automation Portal**'. It is called here with a double click.
(→ TIA Portal V11)



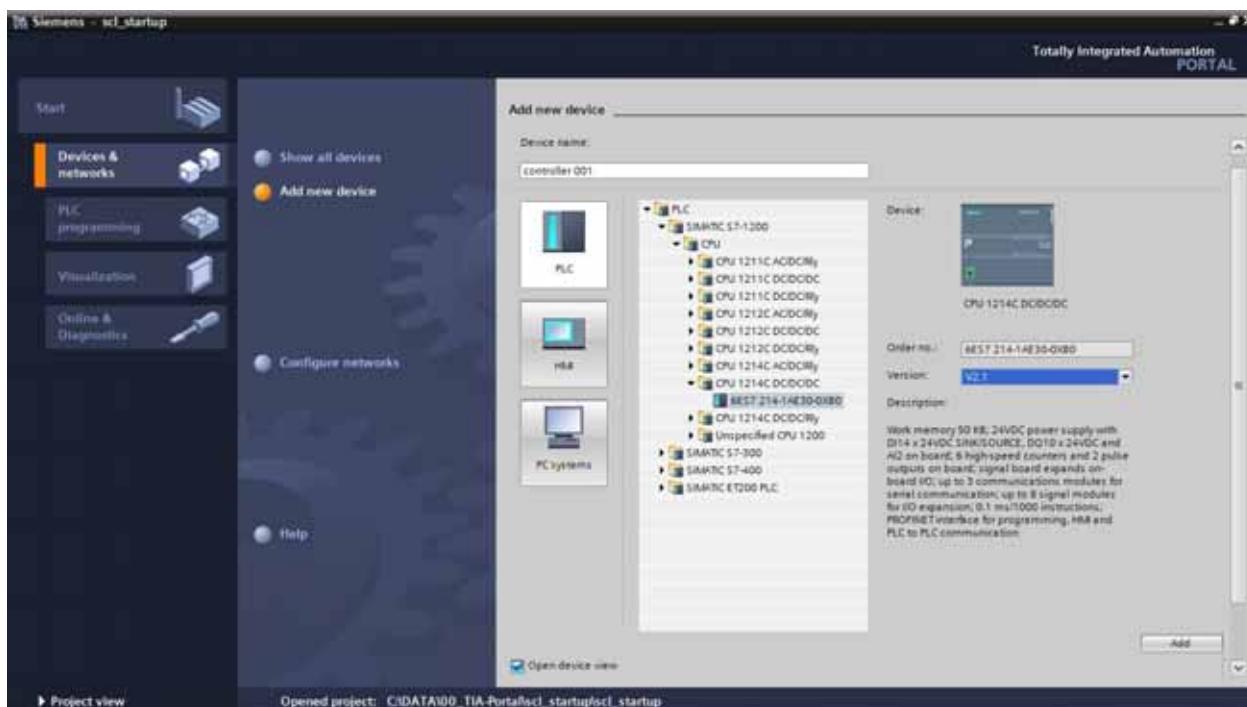
2. Programs for the SIMATIC S7-1200 are managed in projects. We are now setting up such a project in the portal view. (→ Create new project → scl_startup → Create)



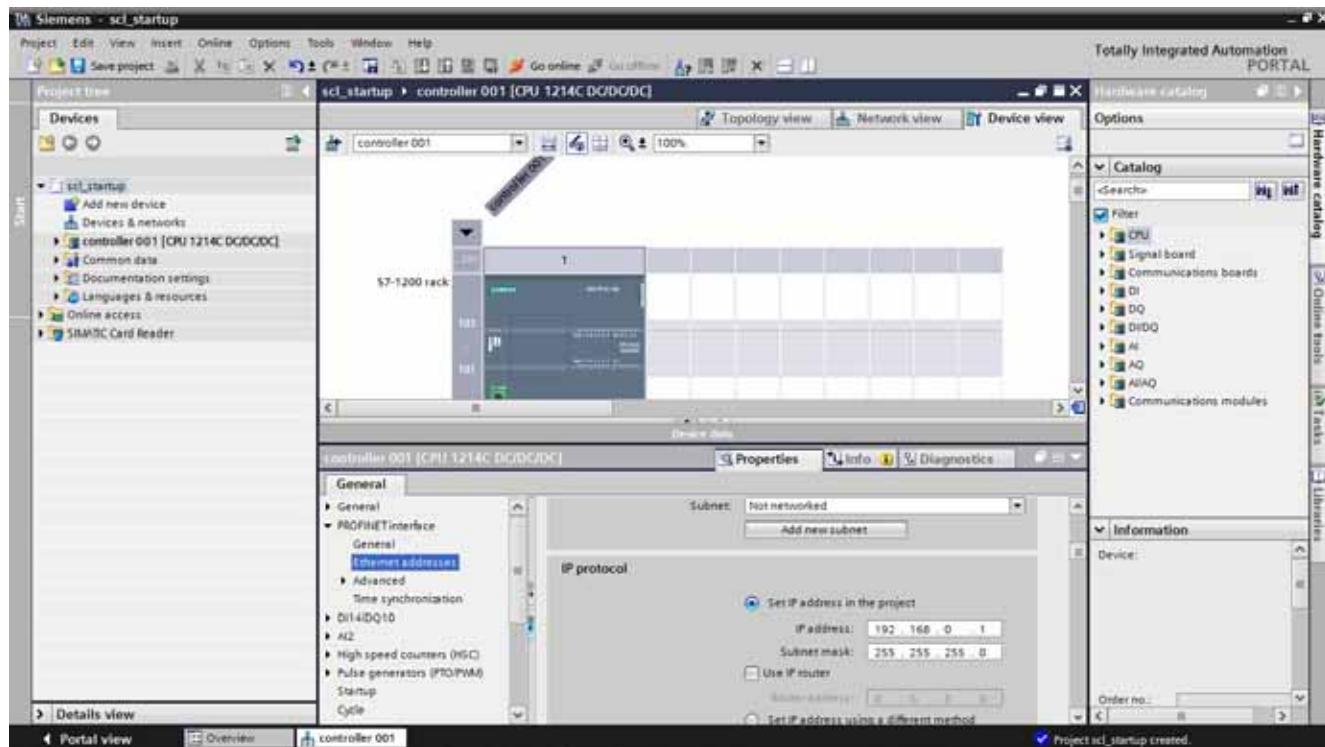
3. Now, 'First steps' are suggested for configuration. We first want to 'Configure a device'. (→ First steps → Configure a device)



4. Then we 'Add new device' with the 'Device name controller 001'. To this end, we select from the catalog 'CPU1214C AC/DC/Rly' with the matching order number. (→ Add new device → controller 001 → PLC → SIMATIC S7-1200 → CPU → CPU1214 AC/DC/Rly → 6ES7 214-1BE30-0XB0 → V2.2 → Add)

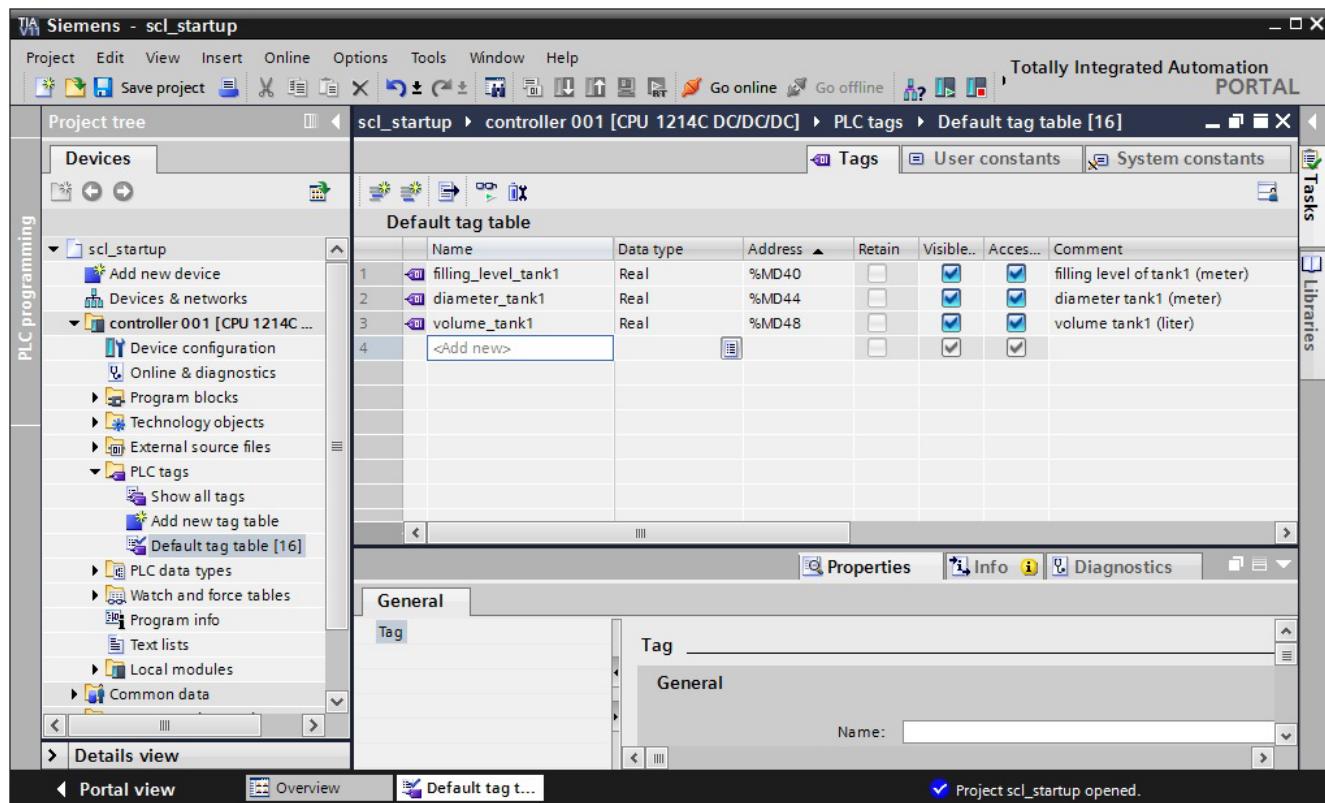


5. The software now changes automatically to the project view, with the opened hardware configuration in the device view. For the CPU to later access the correct CPU, its '**ETHERNET address**' has to be set.
 (→ Properties → General → ETHERNET address → IP address: 192.168.0.1 → Subnet mask: 255.255.255.0)

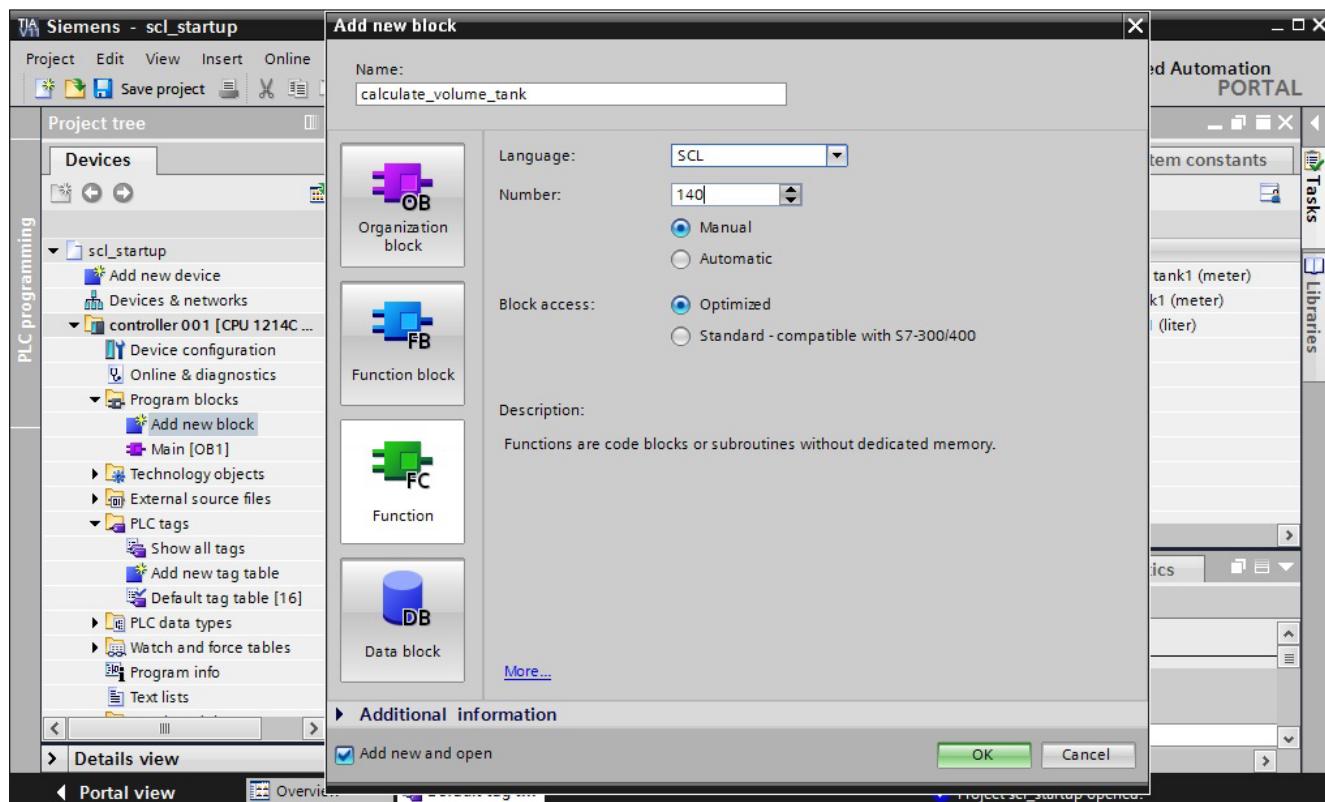


5.2 Setting Up the Program

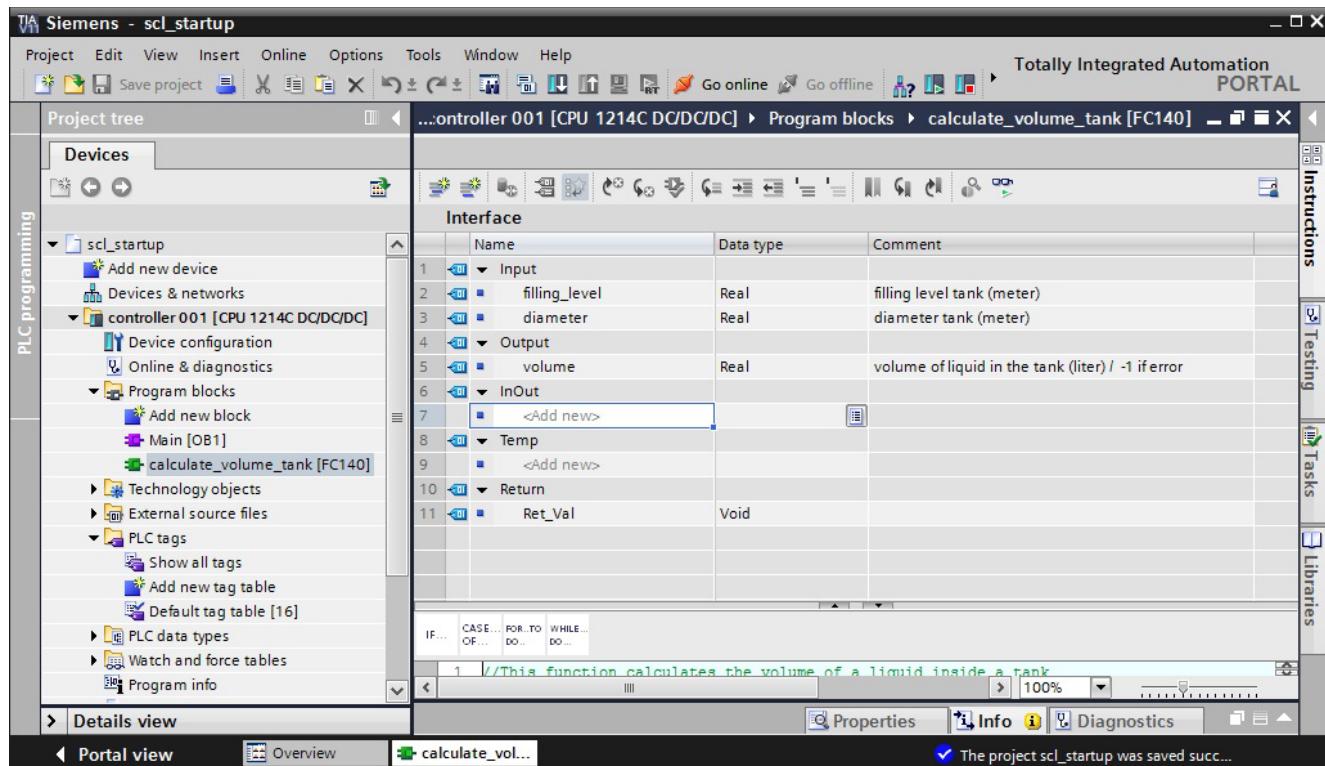
1. Open the 'Default tag table' and enter the 'Operands' with their names and data types.
 (→ controller 001[CPU1214 AC/DC/Rly] → PLC tags → Default tag table → Add new)



2. In order to generate the function calculate_volume_tank [FC140], in project navigation select '**controller 001[CPU1214 AC/DC/Rly]**' and then '**Program blocks**'. Next, double click on '**Add new block**'. In the selection, select '**Function (FC)**' and assign the name '**calculate_volume_tank**'. Change the programming language to '**SCL**'. The numbering can be changed by switching from automatic to manual. Enter the number 140. Accept the inputs with '**OK**'.
 (→ controller 001[CPU1214 AC/DC/Rly] → Program blocks → Add new block → Function (FC) → calculate_volume_tank → SCL → manual → 140 → OK)



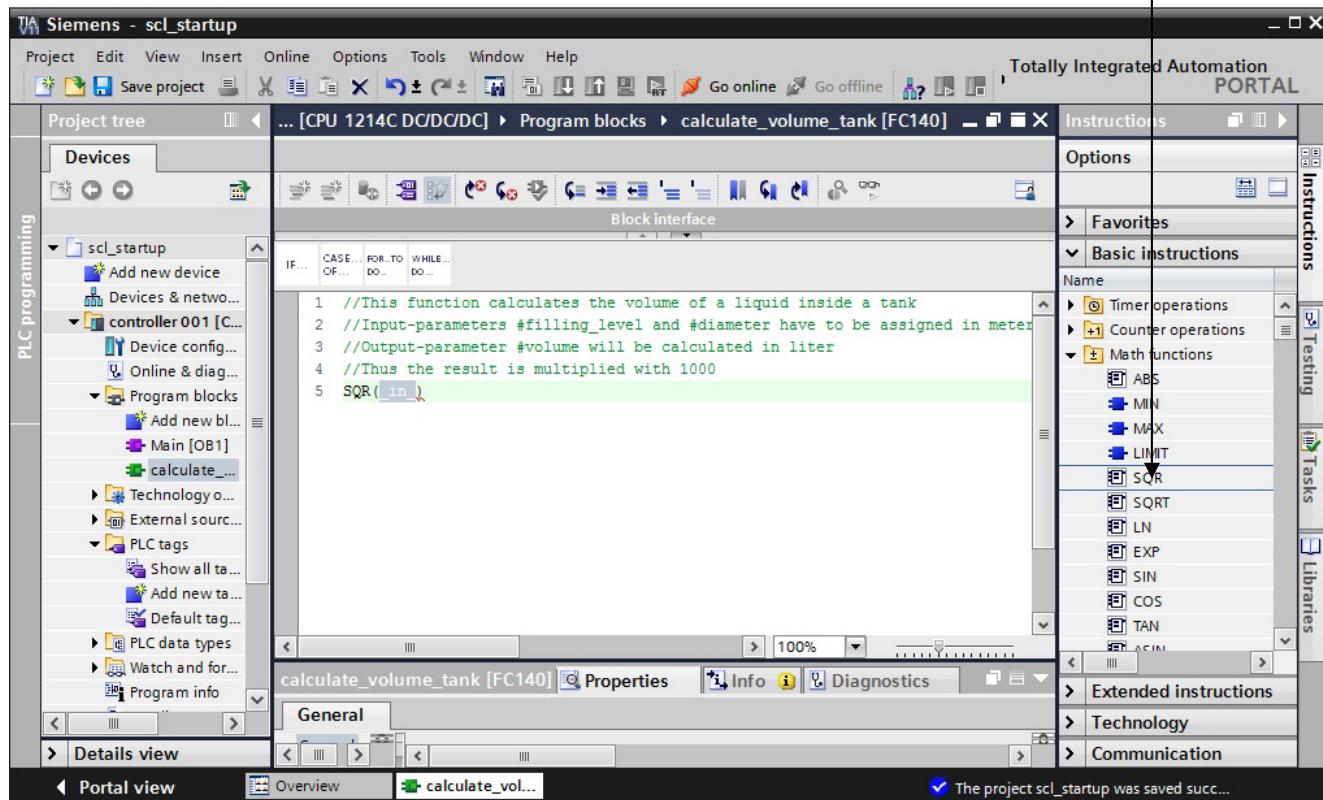
3. The block 'calculate_volume-tank [FC140]' is opened automatically. Now, enter '**Input and Output parameter**' of the block as specified. All local tags should be provided with '**sufficient comment**' for better understanding.
 (→ Enlarge interface of block FC140 → enter interface parameters)



Note:

To avoid a confusion with the PLC tags, it is helpful to write the local tags in lowercase letters.

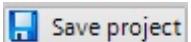
4. After the local tags were declared, we can start creating the program shown here. The square function '**SQR**' integrated in SCL is used to square a function. The number to be squared is in parentheses. To insert the function, drag it to the location in the program where it is to be used (→ Instructions (menu to the right!) → Simple instructions → Math functions → SQR)

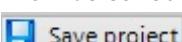


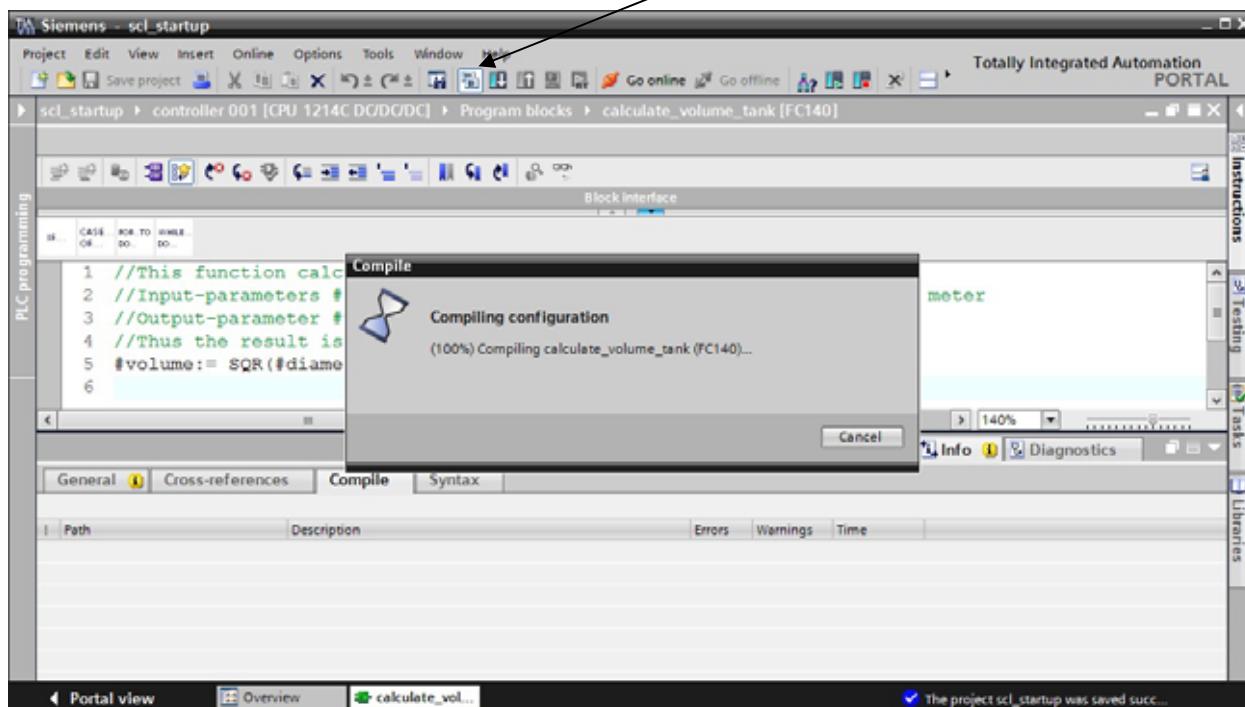
5. Supplement the program as shown below.

```

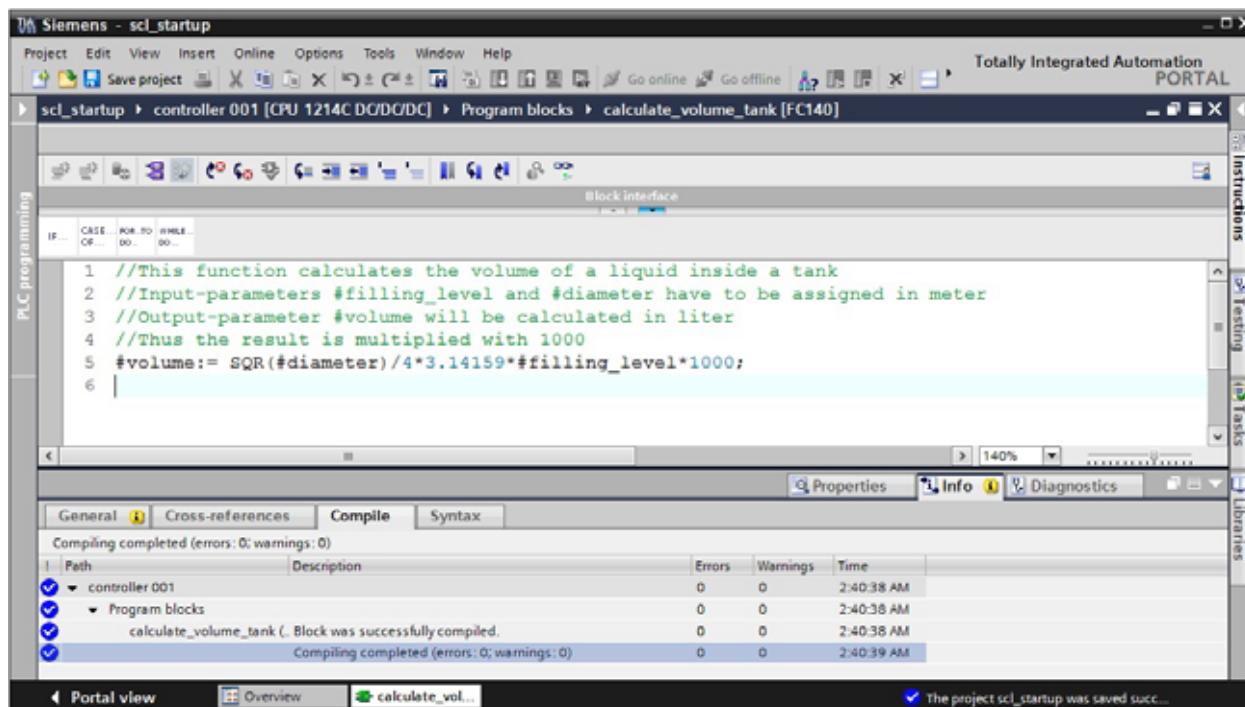
1 //This function calculates the volume of a liquid inside a tank
2 //Input-parameters #filling_level and #diameter have to be assigned in meter
3 //Output-parameter #volume will be calculated in liter
4 //Thus the result is multiplied with 1000
5 #volume:= SQR(#diameter)/4*3.14159*#filling_level*1000;
  
```

6. The program can now be saved  and compiled .

(→ Save project  → Compile 

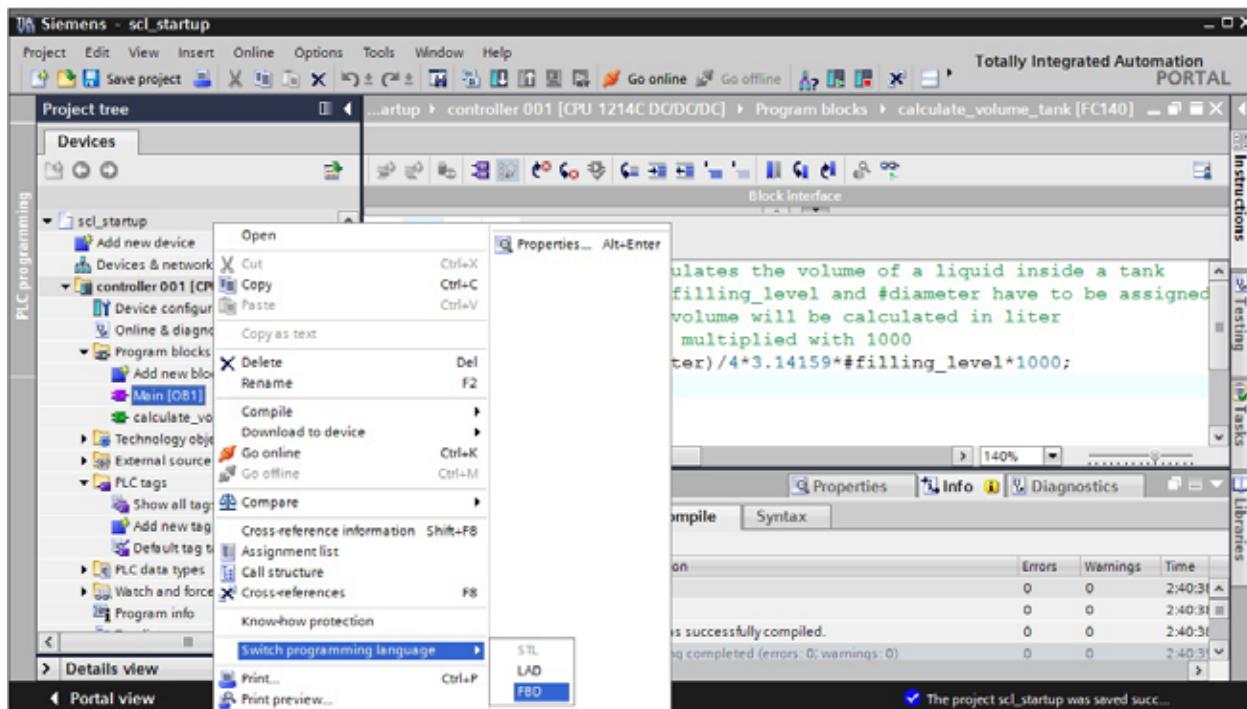


7. Syntax errors that occurred are detected during compilation and displayed in 'Menu info/Compile'.
(→ Info → Compile)

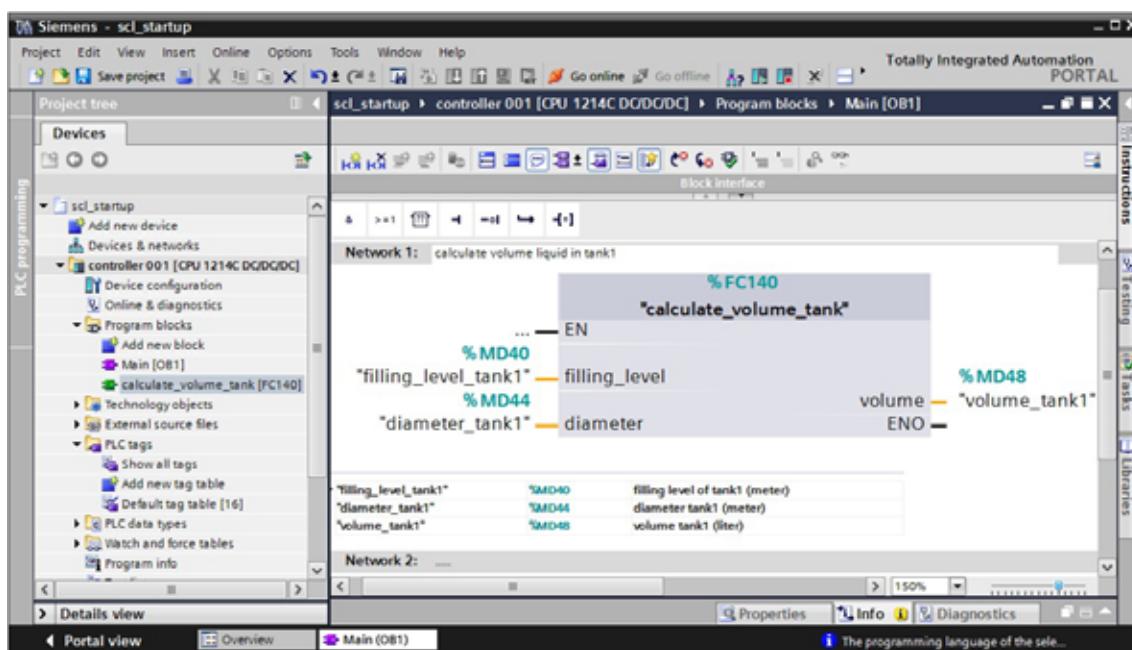


9

8. Now, the function can be called in the '**Main[OB1]**'. Before we open the block '**Main[OB1]**' with a double click, we change its programming language to '**FBD**'.
 (→ Main[OB1] → Switch programming language → FBD)

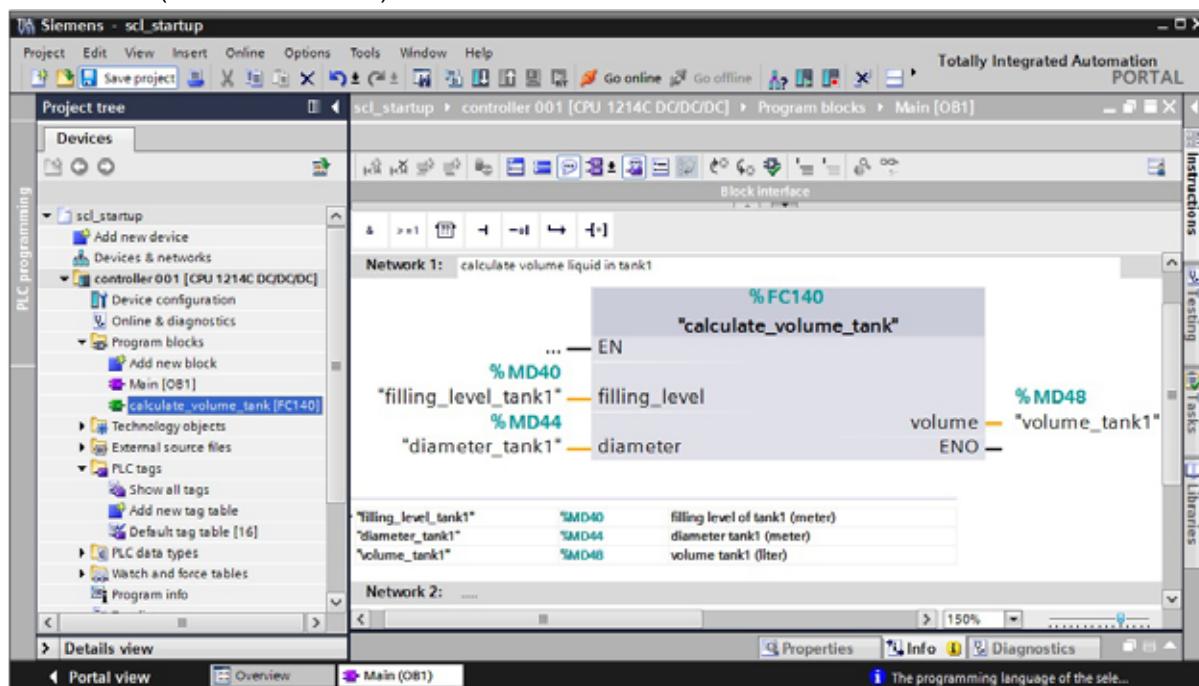


9. Next, we simply drag the function '**calculate_volume_tank[FC140]**' into Network 1 of the Main[OB1]. The interface parameters of the function 'calculate_volume_tank[FC140]' have to be wired to the global PLC tags, as shown here. Remember to document the networks in the Main[OB1]. (→ Main[OB1] → Program blocks→ calculate_volume_tank[FC140])

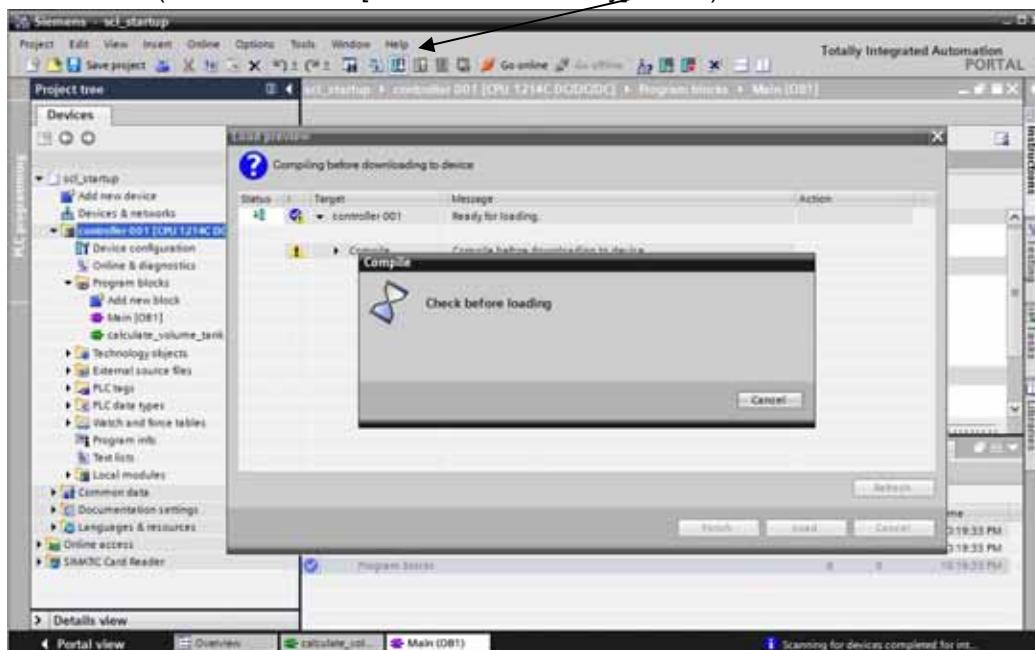


10. Save the project again with  Save project

(→  Save project)



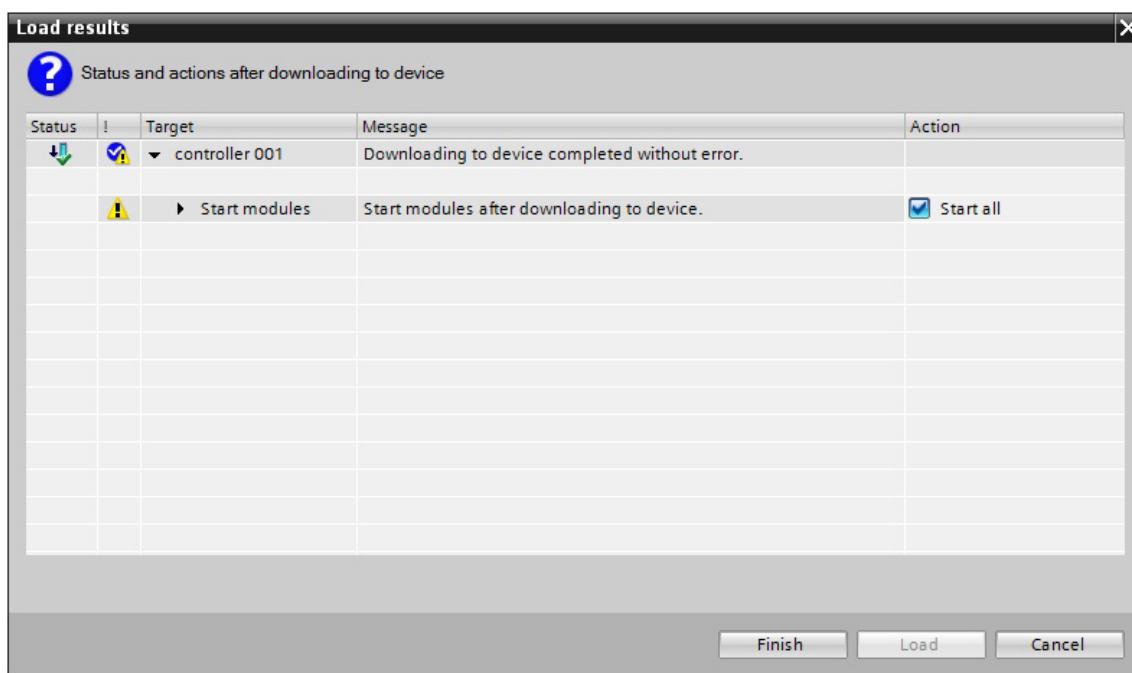
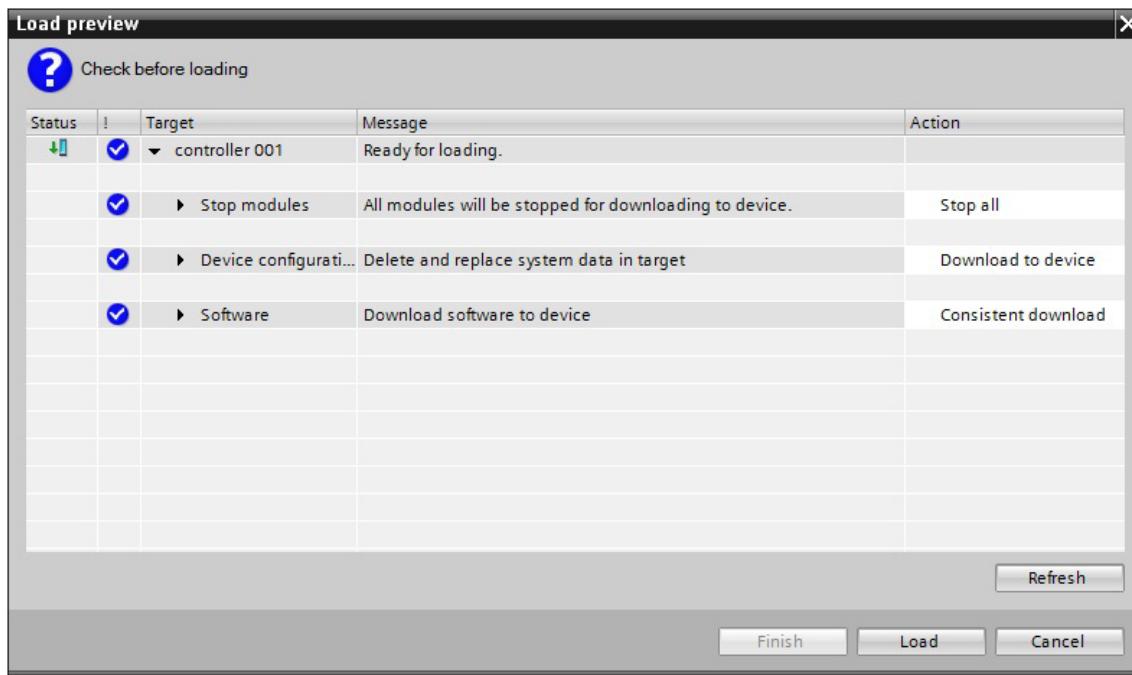
11. Now, in order to load the program blocks and the device configuration to the CPU, first select the folder '**controller 001[CPU1214 AC/DC/Rly]**' and then click on the symbol  Download to device. (→ controller 001[CPU1214 AC/DC/Rly] → 



Note:

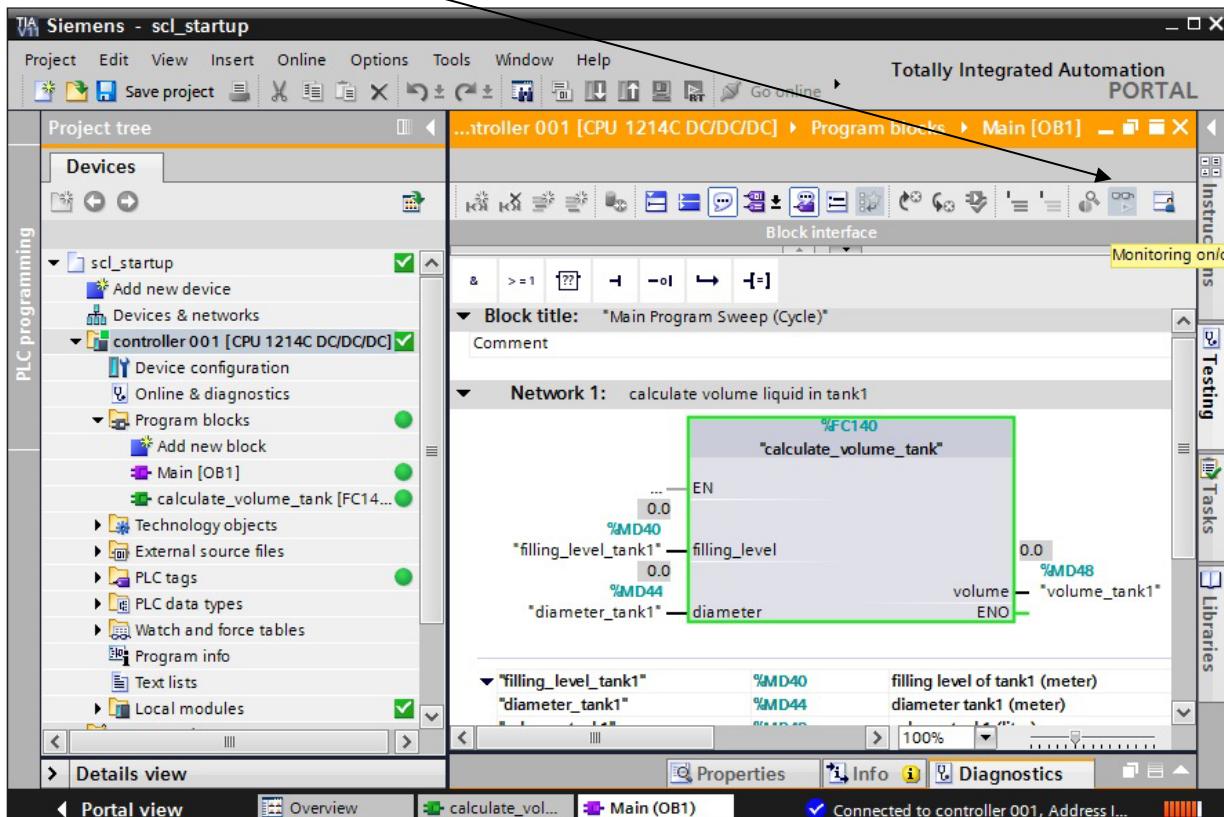
Through the loading process, the project is automatically compiled once more and errors are searched for.

12. Prior to the download, an overview is displayed to check the steps that are to be performed. Start these steps with '**Load**', and start the CPU with '**Start all**' '**Finish**'. (→ Load → Start all → Finish)



5.3 Testing the Program

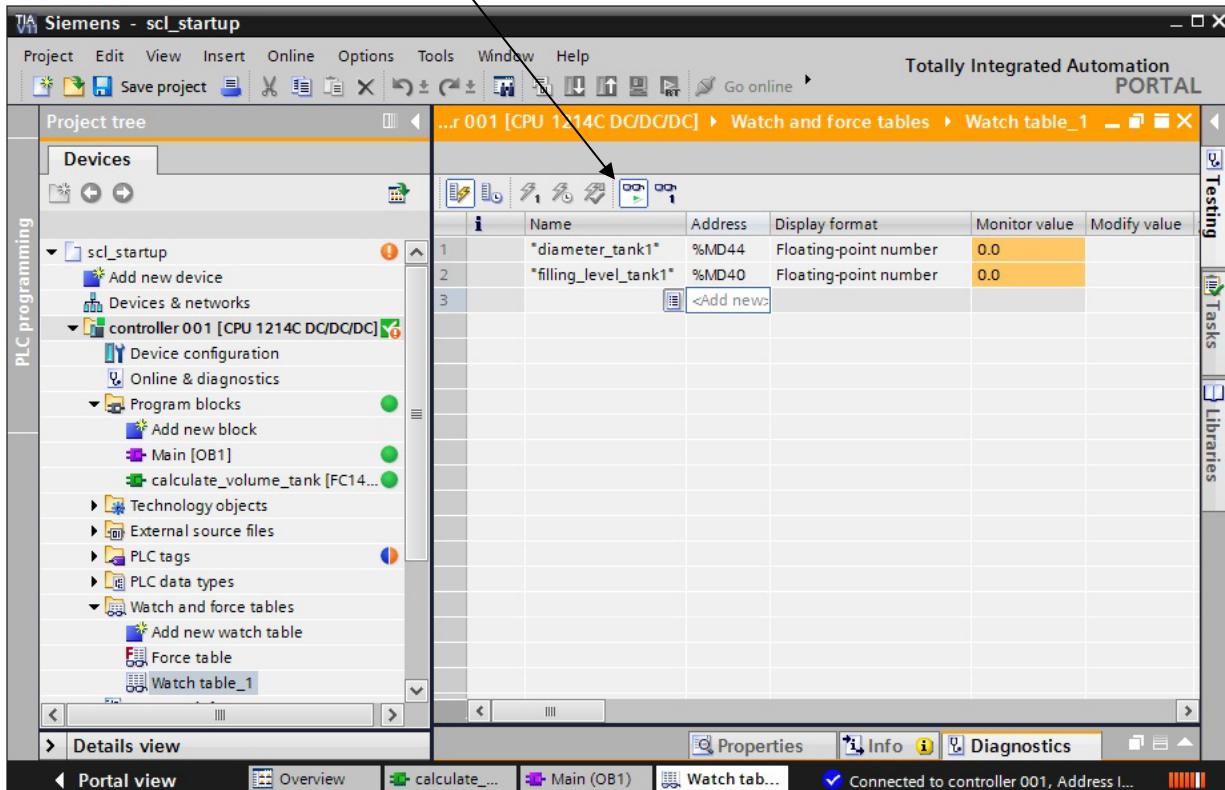
1. By clicking on the symbol  Monitoring on/off, you can observe the status of the input and output tags at the block 'calculate_volume_tank' while the program is tested.



2. Since we don't have an analog sensor and therefore no corresponding process value available, we have to specify the values '**diameter_tank1**' and '**filling_level_tank1**' by using a monitoring table. Set up a '**new monitoring table**' and enter the two values. Switch on the '**Monitoring mode**' to view the current values.

(→ controller 001 → Monitoring and force table → New monitoring table → diameter_tank1,

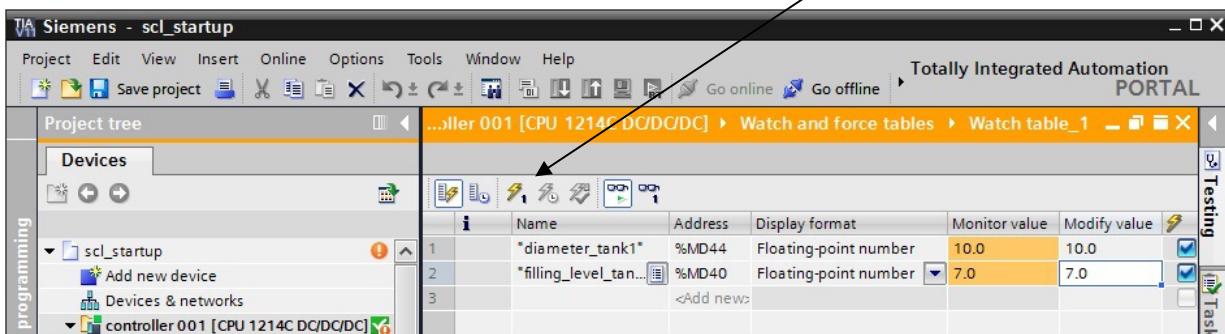
filling_level_tank1 →



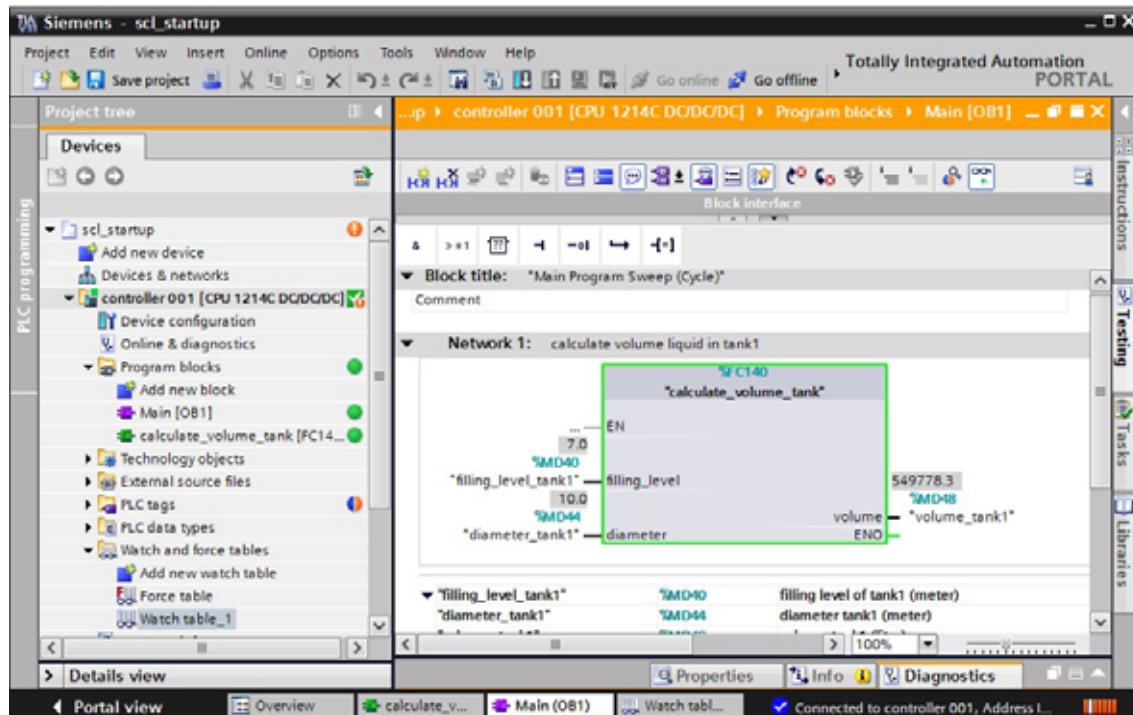
3. To specify the values, we now enter a control value in the column '**Control value**'. With the button

'Write control values once and immediately' , the values are transferred to the CPU.

(→ diameter_tank1 = 10.0 → filling_level_tank1 = 7.0 →

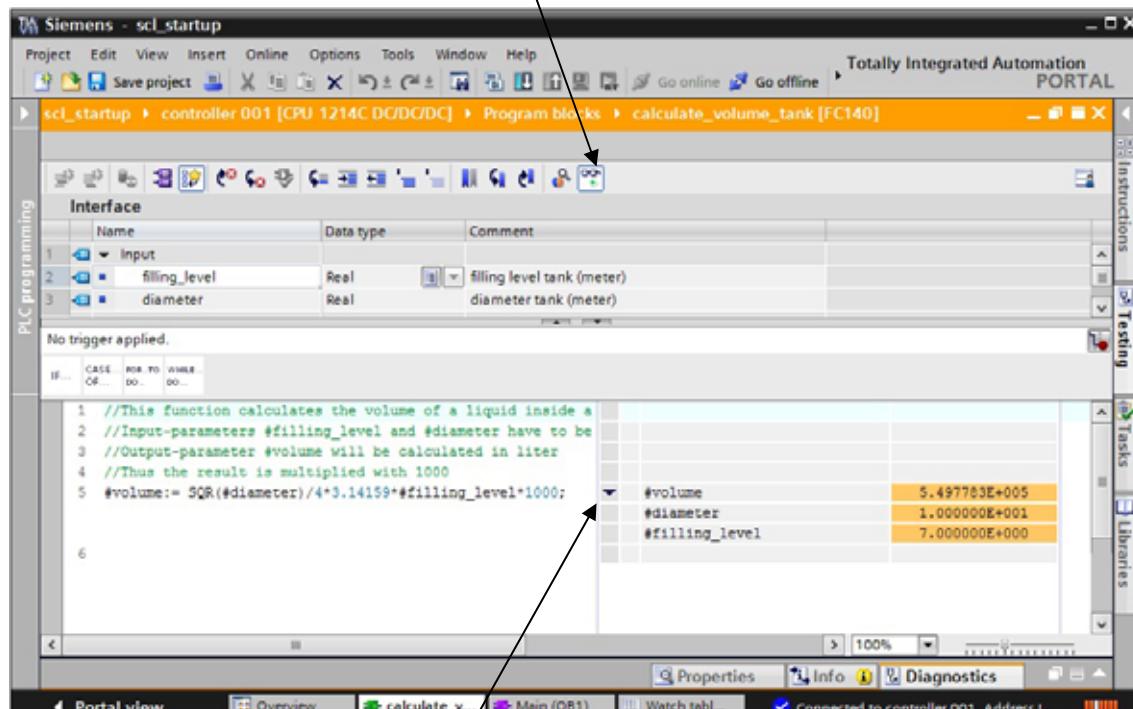


4. Now, the program can be checked in OB1. (→ 



5. We can monitor the values of the individual tags in the SCL editor. To this end, switch the

'Monitoring mode on'. (→ 



Note:

If you click on this button  , the current values of the tags programmed in this line are displayed.

5.4 Expanding the Program

Now, the block '**calculate_volume_tank**' is to check whether faulty data was entered at the input parameters. In addition, the value '**height_max**' is transferred to the block. It specifies the height of the tank.

The block is to evaluate whether the level of the tank is less than zero or more than the specified height of the tank. In addition, it is to be checked whether the diameter was specified as less than zero.

If there is an error, a boolean output parameter is to return '**er**' TRUE, and the value of the parameter is to be '**volume**' -1.

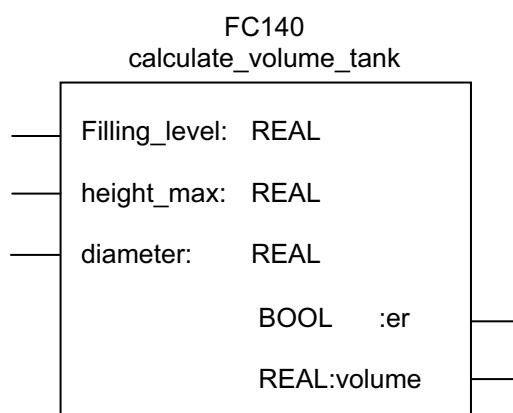
Expanding the assignment list/tag table:

Address	Symbol	Data Type	Comment
%Q1.7	error_bit	BOOL	error, calculation not possible

Expanding the interface of function FC140: calculate_volume_tank

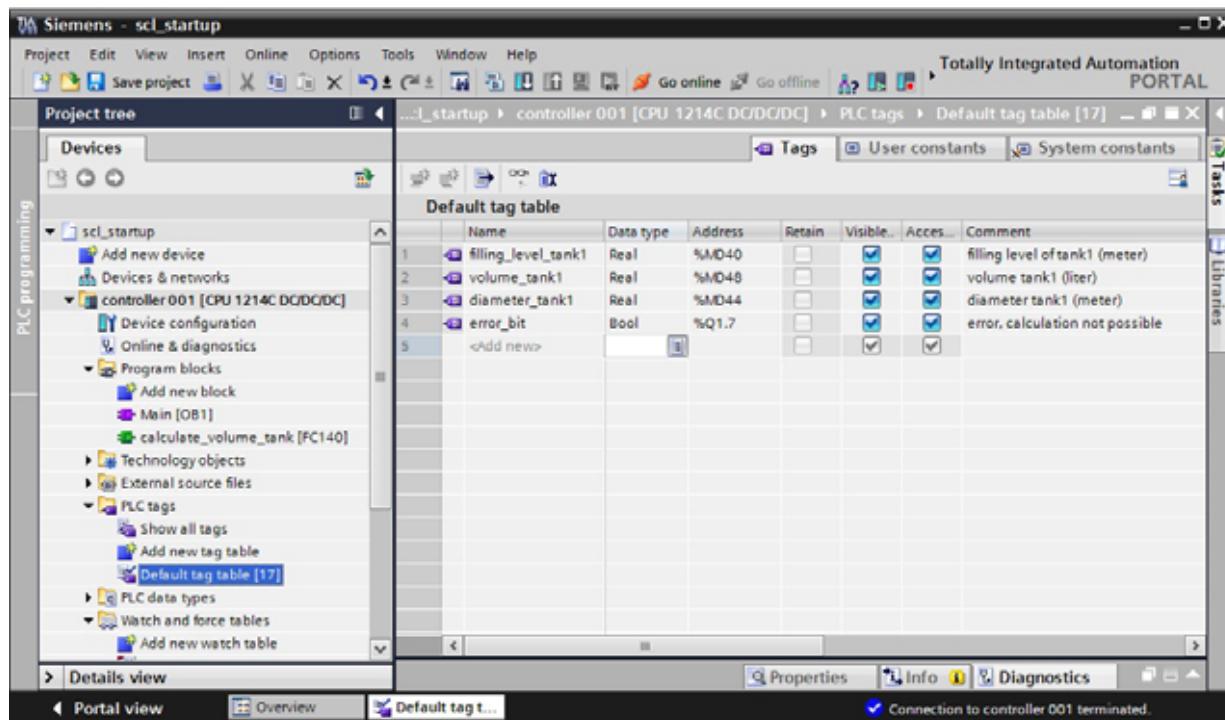
Type	Symbol	Data Type	Comment
IN	height_max	REAL	maximum height (meter)
OUT	er	BOOL	er = true -> error

Function call: representation in FBD



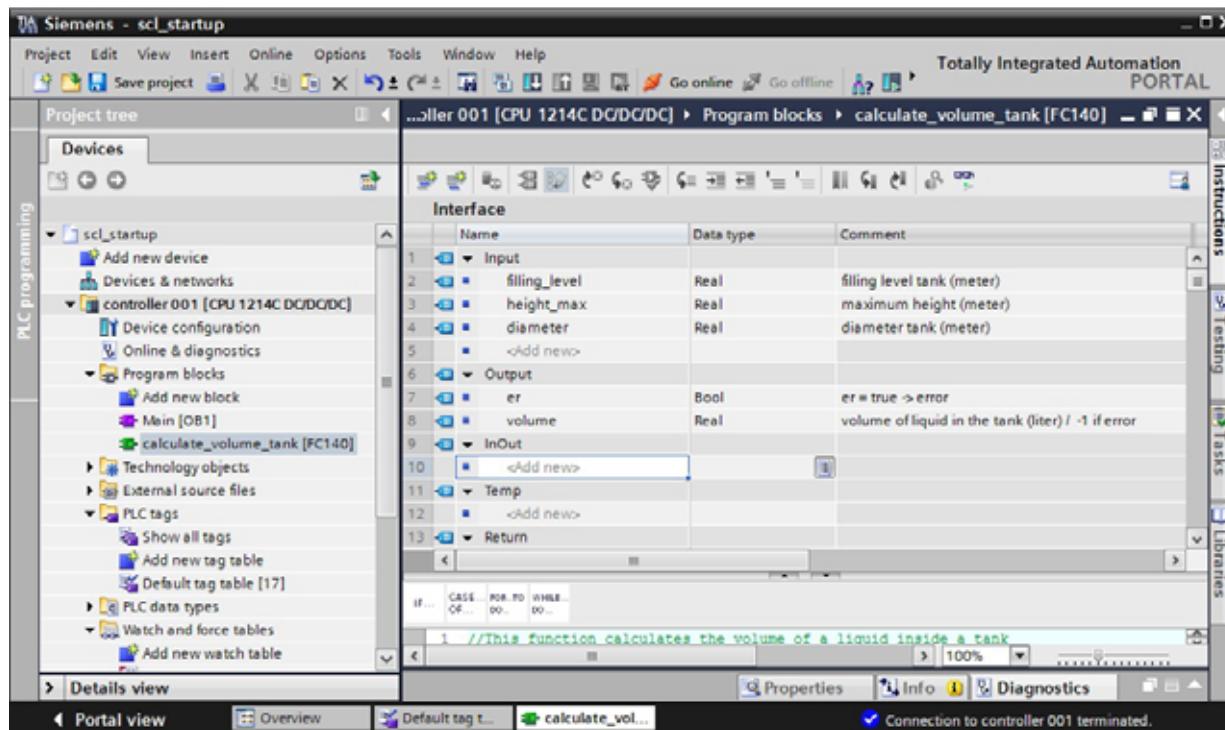
1. Expand the block's '**tag table**' as specified above.

(→ controller 001[CPU1214 AC/DC/Rly] → PLC tags → default tag table → enter operands)



2. Expand the block's '**Interface parameters**' as specified above.

(→ controller 001[CPU1214 AC/DC/Rly] → Program blocks → calculate_volume_tank) → enter parameters)



3. Now, supplement the program as indicated below and search for syntax errors by compiling it.
Save the program and load it to the controller.

(Write program → →

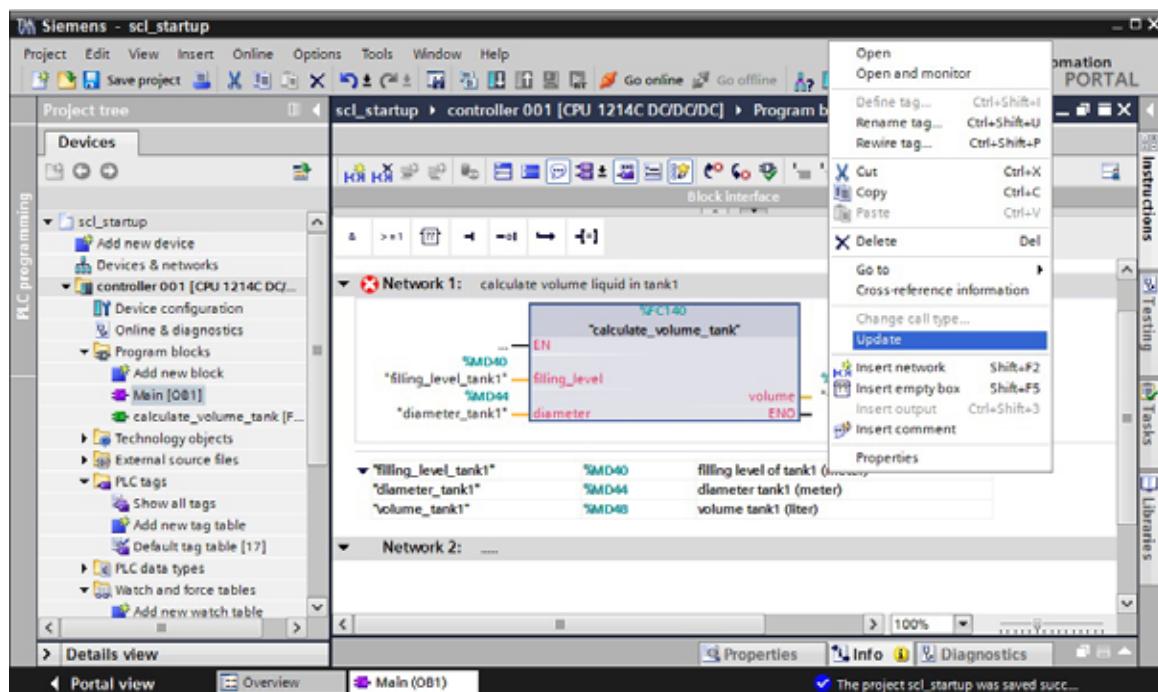
The screenshot shows the SIMATIC Manager software interface. The title bar reads "...controller 001 [CPU 1214C DC/DC/DC] > Program blocks > calculate_volume_tank [FC140]". The main area is titled "Block interface" and contains a code editor. The code editor has tabs for IF..., CASE..., FOR..TO, WHILE..., and DO... at the top. The code itself is as follows:

```

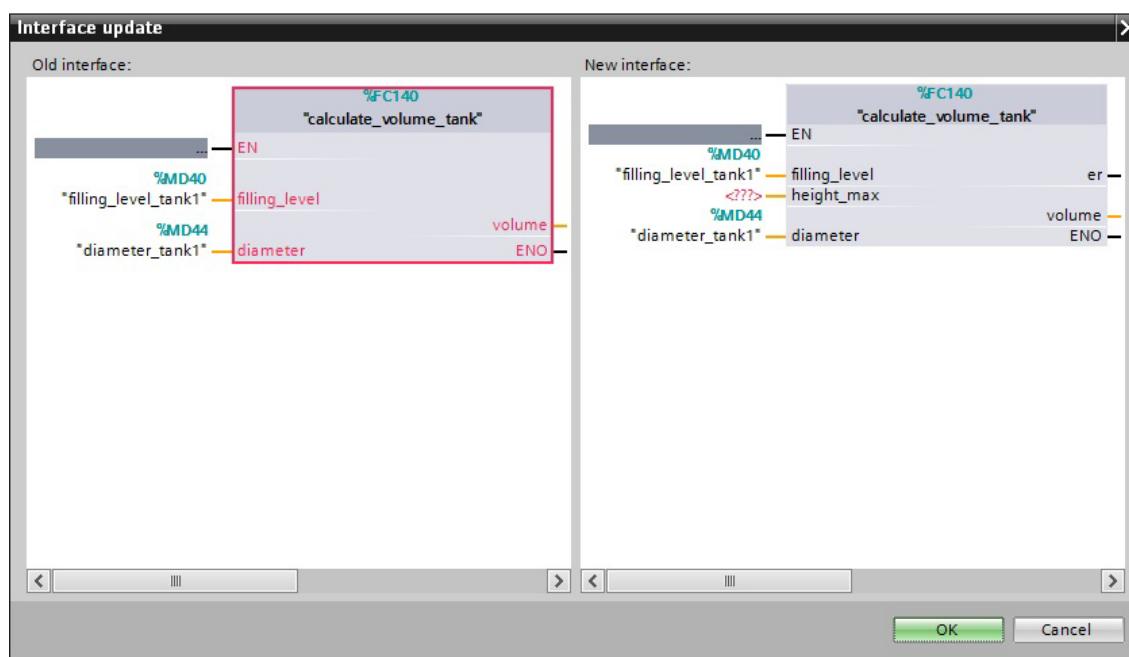
1 //This function calculates the volume of a liquid inside a tank
2 //Input-parameters #filling_level and #diameter have to be assigned in meter
3 //Output-parameter #volume will be calculated in liter
4 //Thus the result is multiplied with 1000
5
6 IF #filling_level>= 0 AND #filling_level<= #height_max AND #diameter> 0 THEN
7     #volume:= SQR(#diameter)/4*3.14159*#filling_level*1000;
8     #er:= false;
9 ELSE
10    #volume:= -1;
11    #er:= true;
12 END_IF;

```

4. Since the block's parameters were changed, the call has to be updated in OB1. Open OB1 and scroll to the location of the block call. With the right mouse key, open the context menu and select 'Update'.
 (→ controller 001[CPU1214 AC/DC/Rly] → Program blocks → Main [OB1]
 → right mouse key → Update)

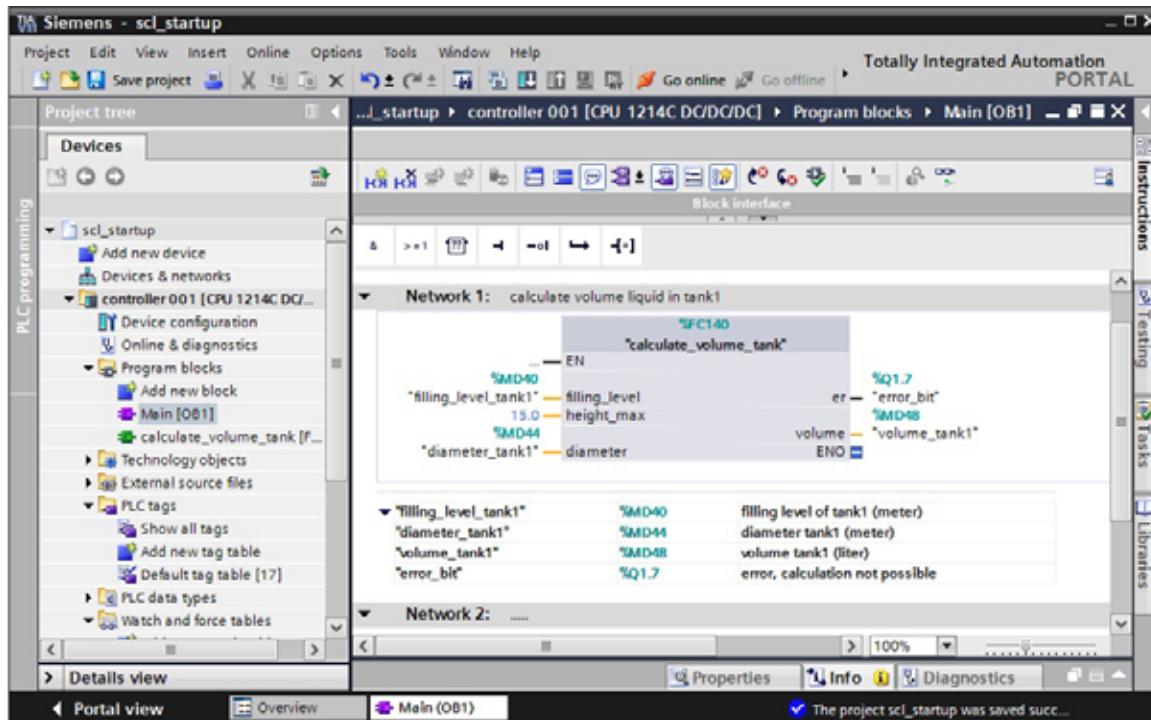


5. Now, the old and the new interface are shown. Confirm with 'OK'.
 (→ OK)



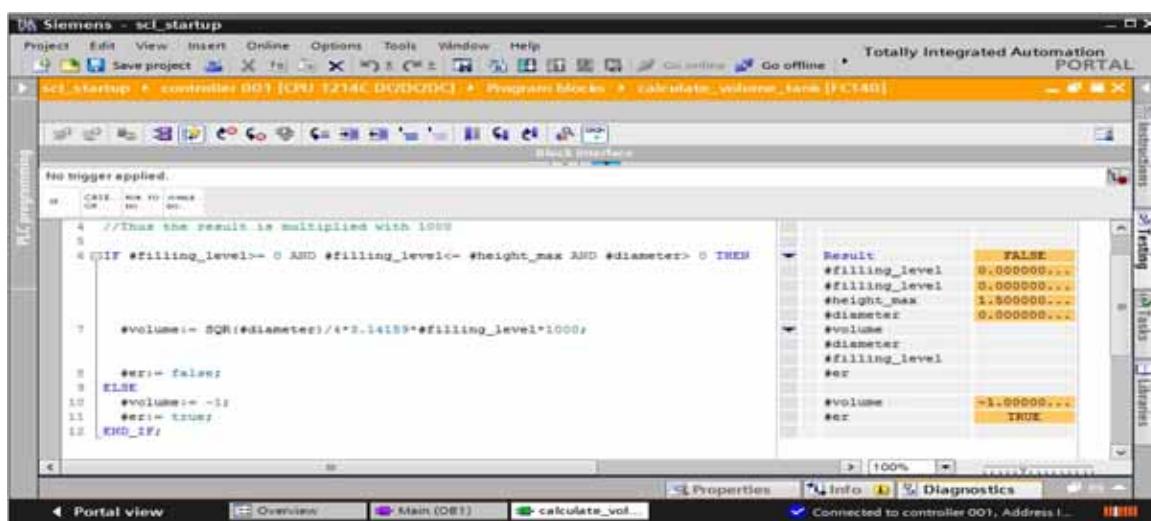
6. Supplement the values at the input parameter 'height_max' and at the output parameter 'er' as shown below. Compile, save and load the program to the controller.

(Supplement parameters → → → →



7. Check the changes in the 'Monitoring mode' of the block 'calculate_volume_tank'.

(→ controller 001[CPU1214 AC/DC/Rly] → Program block → calculate_volume_tank →



Note:

The display format of the current values can be changed by right clicking the value and opening the context menu.



Applicable practical know-how

Comprehensive teaching support for educational institutions

Siemens Automation Cooperates with Education

Siemens Automation Cooperates with Education (SCE) focuses on the needs of the students by providing educators the tools and training to build confidence and applied expertise. The SCE program delivers value to learning institutions with instructor training, learning curriculum and exceptional hardware and software Trainer Packages. Through partnerships we are driven to share knowledge, resources and tools for teaching automation and drive system technologies.

Scan the
QR code
for further
information.



Get more information

Siemens Automation Cooperates with Education:

www.siemens.com/sce

SCE training curriculums:

www.siemens.com/sce/modules

SCE trainer packages:

www.siemens.com/sce/tp

SCE contact partners:

www.siemens.com/sce/contact

SCE Support Finder:

www.siemens.com/sce/supportfinder

LOGO! Logic Module:

www.siemens.com/logo

SIMATIC Controller:

www.siemens.com/controller

Distributed IO:

www.siemens.com/et200

Totally Integrated Automation:

www.siemens.de/totally-integrated-automation

Totally Integrated Automation Portal:

www.siemens.com/tia-portal

SIMATIC Manual Guide:

www.siemens.com/simatic-docu

Information material for SIMATIC:

www.siemens.com/simatic/printmaterial

Siemens Industry Online Support – Technical Forum:

www.automation.siemens.com/WW/forum

Service&Support:

www.siemens.com/automation/support

Other contact partners:

www.siemens.com/automation/partners

Industry Mall for electronic ordering:

www.siemens.com/industrymall

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit
<http://www.siemens.com/industrialsecurity>.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit
<http://support.automation.siemens.com>.



Scan this
QR code to visit
our website!
siemens.com/SCE



The information provided in this brochure contains merely general descriptions or characteristics of performance which in case of actual use do not always apply as described or which may change as a result of further development of the products. An obligation to provide the respective characteristics shall only exist if expressly agreed in the terms of contract. Availability and technical specifications are subject to change without notice. All product designations may be trademarks or product names of Siemens AG or supplier companies whose use by third parties for their own purposes could violate the rights of the owners.

Siemens AG
Industry Sector
Industrial Automation Systems
Postfach 48 48
90026 NÜRNBERG
GERMANY

Subject to change without prior notice
Article No. 6ZB5310-0QL01-0BA0
MP.R1.AS.0000.36.3.06 / Dispo 06321
SO 0213 0.5 DPG 418 En
0613 1. DPG / MP.R1.AS.0000.36.3.13
0314 1. DPG / DR.PN.AS.14.XXBR.95.15
Printed in Germany
© Siemens AG 2013