

# SIEMENS

## SIMATIC

## S7-1500

### Getting Started

Welcome

---

Automation task

---

1

Hardware section

---

2

Software section

---

3

Security

---

4






Under the following link you find the Getting Started in multimedia form. Clear videos show you with the help of an automation task the project engineering, programming and visualisation of the S7-1500 with the TIA Portal.

[http://www.automation.siemens.com/salesmaterial-as/interactive-manuals/getting-started\\_simatic-s7-1500/\\_content/EN/content\\_en.html](http://www.automation.siemens.com/salesmaterial-as/interactive-manuals/getting-started_simatic-s7-1500/_content/EN/content_en.html)

## Legal information

### Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 <b>DANGER</b>
indicates that death or severe personal injury <b>will</b> result if proper precautions are not taken.
 <b>WARNING</b>
indicates that death or severe personal injury <b>may</b> result if proper precautions are not taken.
 <b>CAUTION</b>
indicates that minor personal injury can result if proper precautions are not taken.
<b>NOTICE</b>
indicates that property damage can result if proper precautions are not taken.


If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

### Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

### Proper use of Siemens products

Note the following:

 <b>WARNING</b>
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

### Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

### Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

# Welcome

## Welcome

Welcome to the Getting Started "TIA Portal V13".

In this Getting Started, we show you an example of how to use the CPU SIMATIC S7-1500 with the TIA Portal to create an automation solution for a "color mixing plant". Video clips will illustrate the approach for creating a solution for the automation task.

In the first part, you assemble the hardware and prepare your configuration PC.

In the second part, you configure the CPU and HMI visualization using the example of a color mixing plant.

In addition, you can find options and extensions for your automation solutions.

## 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. You can find more information about industrial security on the Internet (<http://www.siemens.com/industrialsecurity>).

To stay informed about product updates as they occur, sign up for a product-specific newsletter. You can find more information on the Internet (<http://support.automation.siemens.com>).

# Table of contents

	<b>Welcome .....</b>	<b>3</b>
<b>1</b>	<b>Automation task .....</b>	<b>7</b>
	1.1 Introduction.....	7
	1.2 Sample project .....	8
<b>2</b>	<b>Hardware section .....</b>	<b>15</b>
	2.1 Introduction.....	15
	2.1.1 Requirements .....	15
	2.1.2 Additional information.....	16
	2.2 Installing the assembly.....	17
	2.2.1 Overview .....	17
	2.2.2 Installing the assembly.....	17
	2.3 Wiring .....	20
	2.3.1 Overview .....	20
	2.3.2 Wiring rules .....	21
	2.3.3 Wiring the mains connection plug.....	22
	2.3.4 Wiring the load current supply (PM) to the CPU.....	24
	2.3.5 Potential bridge circuits.....	25
	2.3.6 Wiring the digital input module.....	25
	2.3.7 Wiring the digital output module.....	27
	2.3.8 Wiring front connectors .....	29
	2.4 Power on .....	31
	2.4.1 Overview .....	31
	2.4.2 Power on .....	31
	2.4.3 Assign IP address via the display .....	33
<b>3</b>	<b>Software section .....</b>	<b>34</b>
	3.1 Creating the project and hardware.....	34
	3.1.1 Introduction to the TIA Portal .....	34
	3.1.2 Creating a project.....	36
	3.1.3 Creating an S7-1500 CPU .....	38
	3.1.4 Running the hardware detection .....	40
	3.1.5 Creating ET 200 interface modules .....	41
	3.1.6 Networking ET 200 interface modules .....	42
	3.1.7 Creating input and output modules and a server module for ET 200SP .....	44
	3.1.8 Creating input and output modules for ET 200MP.....	46
	3.1.9 Assigning names for ET 200 .....	47



3.2	Creating the program .....	48
3.2.1	Loading the block library .....	48
3.2.2	Deleting program block Main [OB1] .....	50
3.2.3	Copying program blocks .....	51
3.2.4	Cyclic interrupt OB .....	52
3.2.4.1	Cyclic interrupt OB – Cycle time and phase .....	52
3.2.4.2	Changing the cycle time .....	53
3.2.5	Copying tag tables .....	54
3.2.6	Compiling a project .....	55
3.2.7	Load project into the CPU .....	57
3.2.8	Optimized block access .....	59
3.2.8.1	Introduction .....	59
3.2.8.2	Expanding and reloading the optimized "Filling" data block .....	60
3.2.9	Versioning a block .....	65
3.2.10	Setting retentivity .....	68
3.2.11	Activating the EN/ENO mechanism .....	71
3.2.12	Using the comment function .....	73
3.2.13	Local error handling .....	74
3.2.13.1	Handle errors within block .....	74
3.2.13.2	Loading blocks for local error handling .....	76
3.2.13.3	Generating errors without local error handling .....	78
3.2.13.4	Generating errors with local error handling .....	79
3.3	Configure visualization .....	81
3.3.1	Present sample project .....	81
3.3.2	HMI configuration .....	81
3.3.2.1	Overview .....	81
3.3.2.2	SIMATIC HMI Comfort Panels .....	82
3.3.2.3	HMI screens .....	83
3.3.2.4	Additional control elements .....	84
3.3.2.5	Recipes .....	85
3.3.2.6	Archives .....	86
3.3.2.7	User-defined functions .....	87
3.3.2.8	User Management .....	88
3.3.2.9	Multilingualism .....	89
3.3.2.10	Reports .....	91
3.3.3	Insert HMI device from libraries .....	93
3.3.3.1	Storing an object in a library .....	93
3.3.4	Configuring HMI connection .....	94
3.3.4.1	Communication between devices .....	94
3.3.4.2	Configuring HMI connection .....	95
3.3.4.3	Connecting HMI tags .....	97
3.3.5	Configuring system diagnostics .....	99
3.3.5.1	System diagnostics basics .....	99
3.3.5.2	System diagnostics views .....	100
3.3.5.3	Configuring the system diagnostic view .....	103
3.3.6	Simulating an HMI device .....	105
3.3.6.1	Simulation basics .....	105
3.3.6.2	Operating the panel in simulation .....	106

3.4	Loading the project into the programming device .....	110
3.4.1	Load CPU to project.....	110
3.5	Team engineering via Inter Project Engineering.....	112
3.5.1	Basics of "Inter Project Engineering" .....	112
3.5.2	Creating an IPE file .....	113
3.5.3	Importing an IPE file.....	114
<b>4</b>	<b>Security .....</b>	<b>117</b>
4.1	Overview of the protective functions of the CPU .....	117
4.2	Using the display to configure additional access protection .....	118
4.3	Know-how protection.....	119
4.4	Copy protection .....	122
4.5	Protection by locking the CPU .....	123
4.6	Configuring access protection for the CPU.....	123
4.7	Configuring protection of the HMI connection.....	126

# Automation task

## 1.1 Introduction

### Introduction

In the following section, you will become familiar with the automation task.

You can find out more about the application example, the hardware configuration and the components of the sample project.

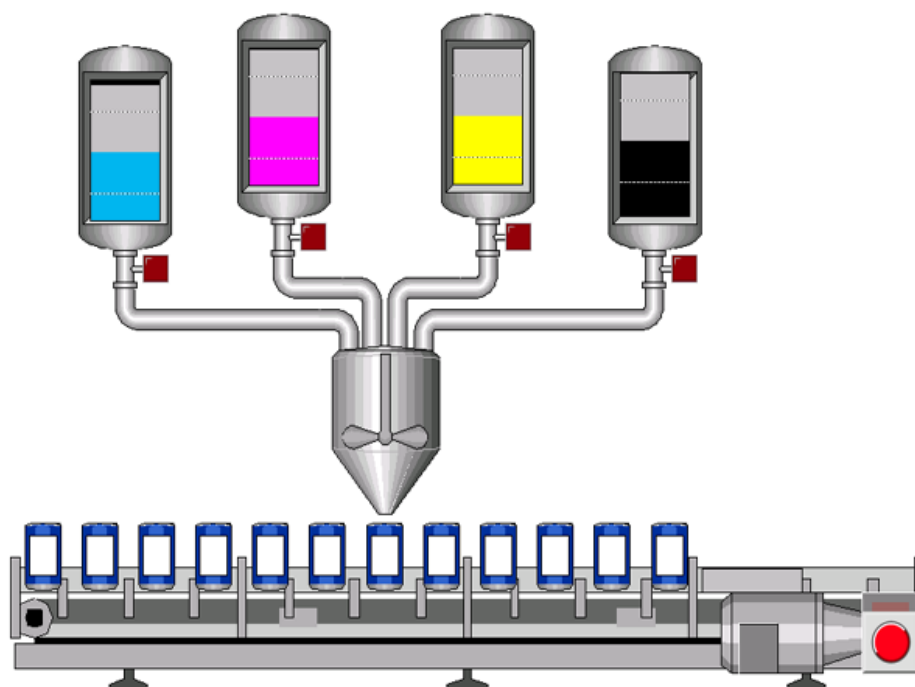
### Application example

The application example for this Getting Started is a color mixing plant for mixing and filling a previously selected color recipe.

There are four color components for the recipe, cyan, magenta, yellow and black, which means the colors of the CMYK color space.

Filling takes place in four steps.

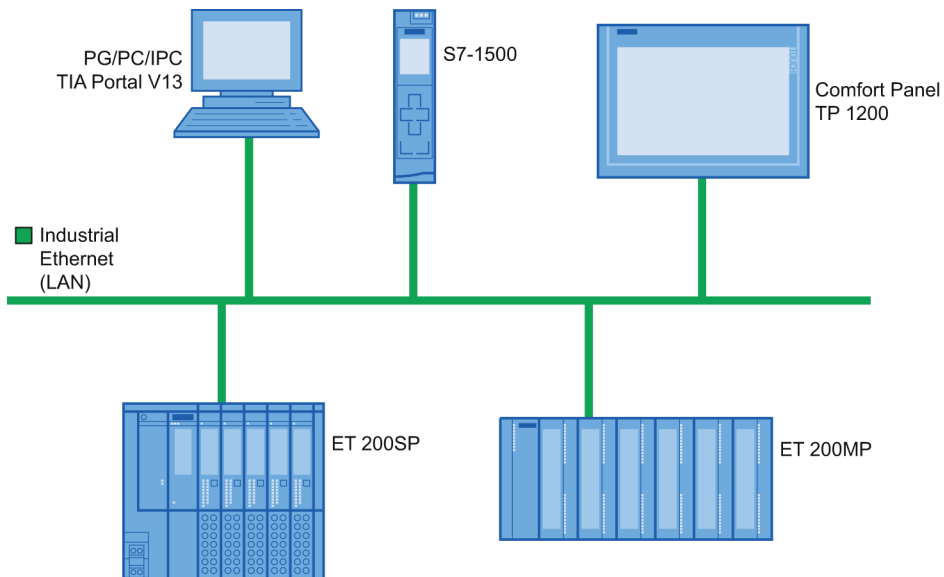
- Selection of the color mixture using the HMI recipe function.
- Filling the recipe components or the four basic colors by opening the respective tank valves.
- Mixing the colors.
- Filling the finished color mixture into tins and transportation by a conveyor belt.



### Design of the hardware configuration

The hardware configuration consists of the following devices:

- The CPU 1511-1 PN with an S7-1500 load current supply, a digital input module and a digital output module.
- HMI Panel TP1200 Comfort that can also be simulated with the TIA Portal.
- Distributed ET 200MP I/O system with IM 155-5 PN ST interface module and digital input and digital output modules.
- Distributed ET 200SP I/O system with IM 155-6 PN ST interface module, digital input modules, digital output modules and server module.



## 1.2 Sample project

### Sample project for the application

To configure the color mixing system with the TIA Portal, create the sample project "Color\_Filling\_Station".

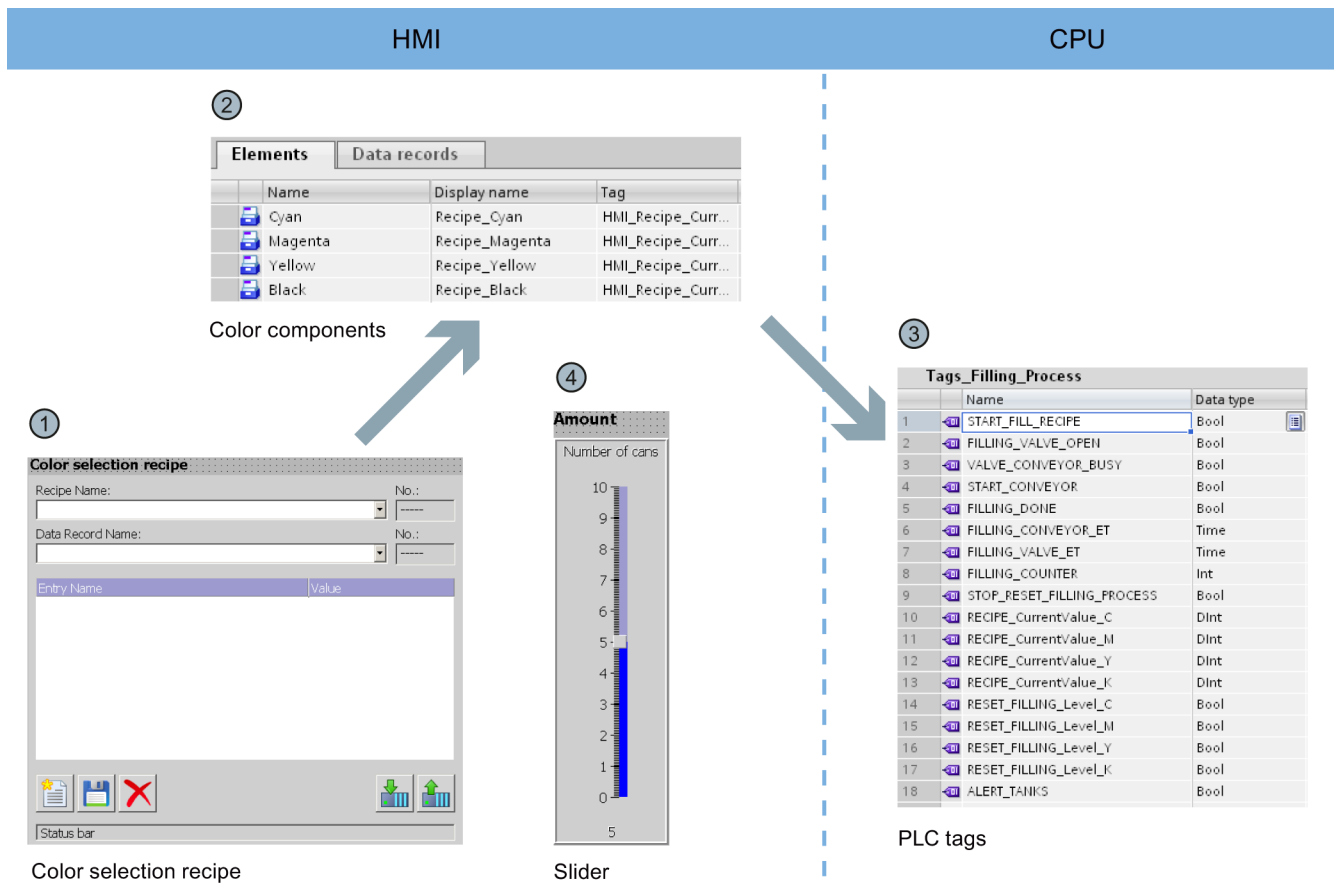
The following project components already exist for the sample project:

- The program blocks of the CPU
- The visualization of the HMI on a Comfort Panel

In this section, we will explain the relationships between the individual project components of the sample project.

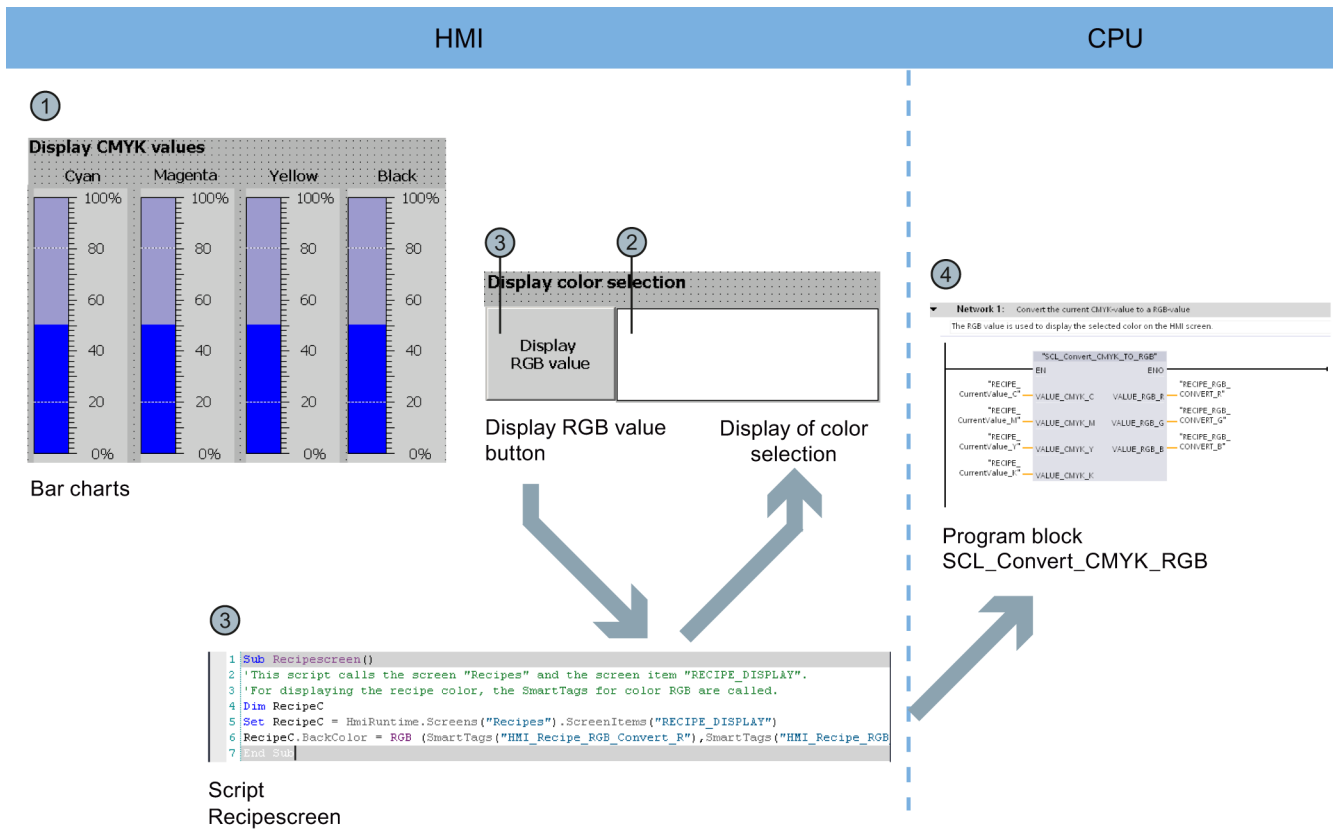
You will carry out the necessary configuration steps yourself at a later point in time.

## Selecting the recipe



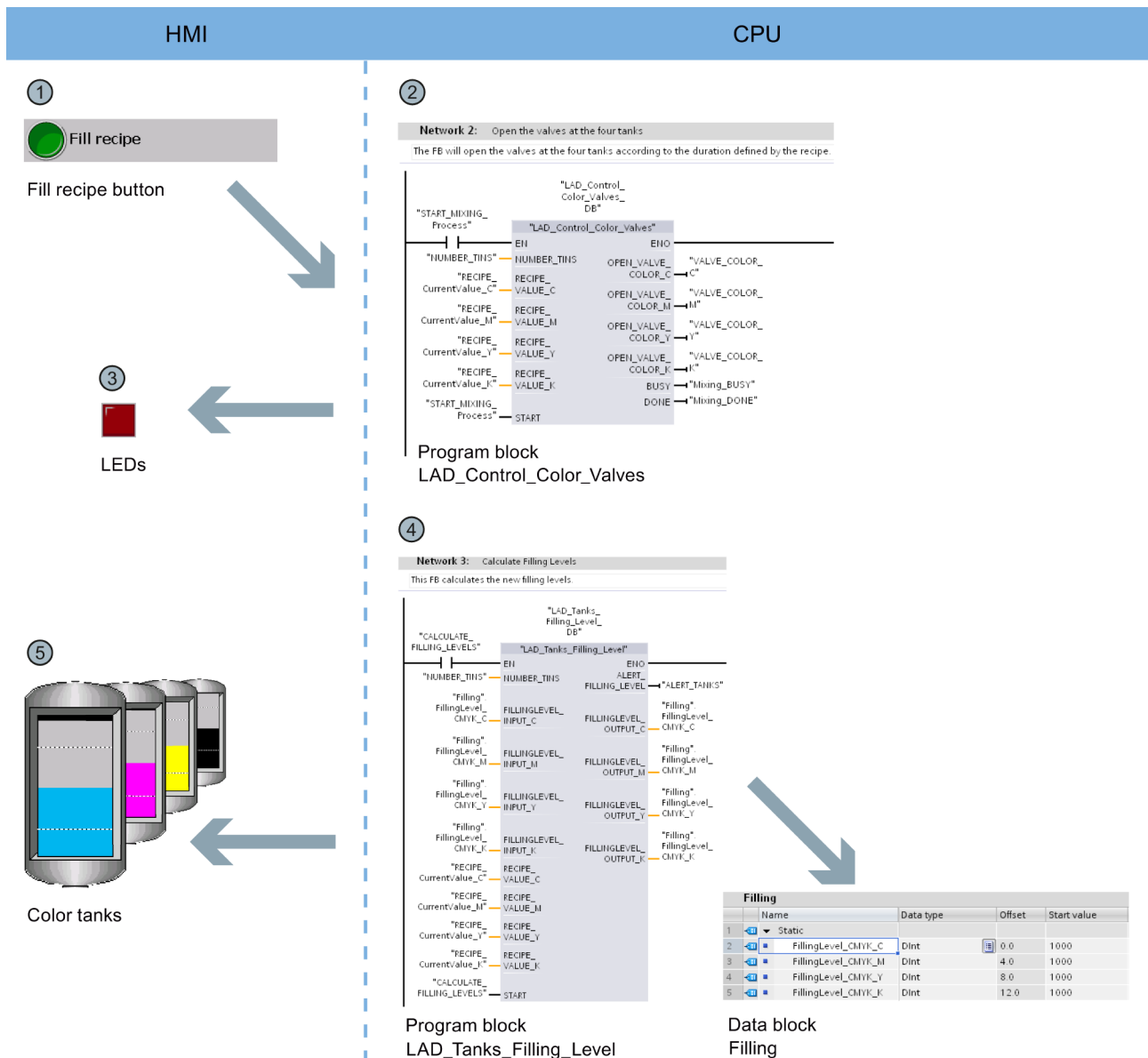
- ① The HMI screen "Recipes" includes the "Color selection recipe". This is a prefabricated object from the library of the TIA Portal. You can use this object to select data records and to create new data records.
- ② The data records (color mixtures) and elements (color components) are stored in the "Recipes" HMI editor. Each color mixture consists of the four color components cyan (C), magenta (M), yellow (Y), black (K). The proportion of each of the four color components in a color mixture is stored in the "Recipes" editor.
- ③ The values for the respective color components are written in PLC tags when you load a color mixture. The PLC tags are stored in the "Tags\_Filling\_Process" tag table of the CPU.
- ④ The HMI screen "Recipes" also includes a slider. You use this slider to specify the number of tins to be filled.

### Displaying the CMYK and RGB values



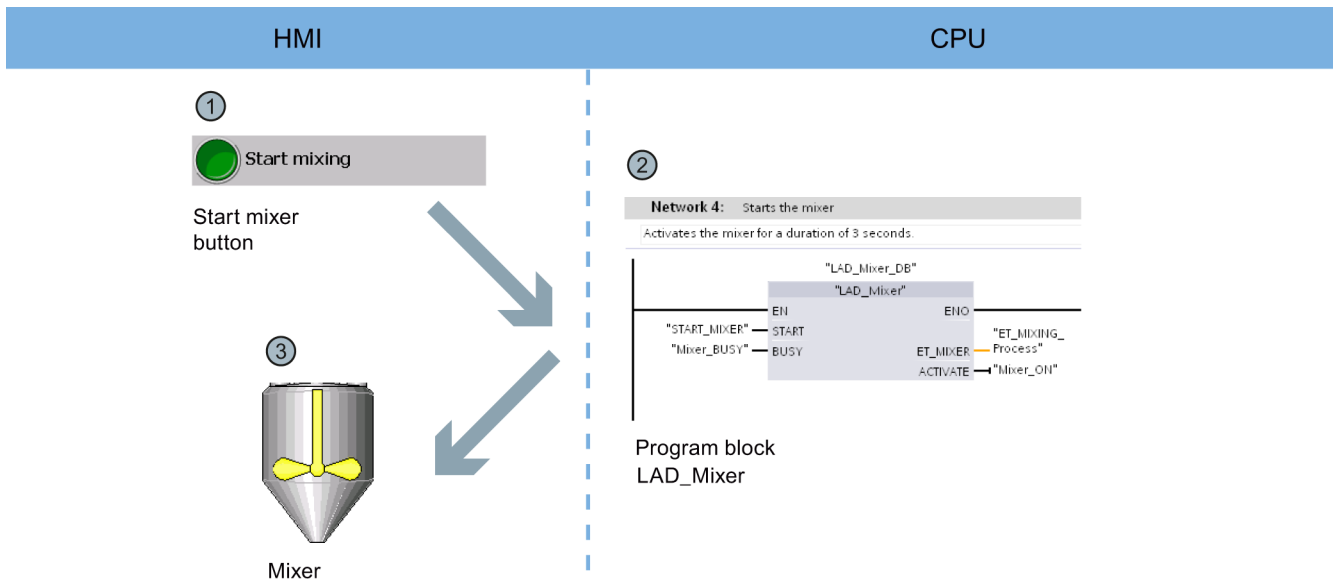
- ① When the required color mixture is selected in the HMI screen "Recipes", the values are displayed in the CMYK color space by means of a bar diagram.
- ② The color mixture can be shown with an additional display. This requires the "Recipescreen" script to be run.
- ③ You run the "Recipescreen" script by clicking the "Display RGB Value" button. The script assigns the RGB value assigned to the CMYK value to the display, because CMYK values cannot be output directly on screens.
- ④ The required RGB value is calculated by the "SCL\_Convert\_CMYK\_TO\_RGB" program block.

## Filling the recipe



- ① The "Fill recipe" button starts the filling of the color components in the HMI screen "Start screen". The button activates the "LAD\_Control\_Color\_Valves" program block.
- ② The program block calculates how long each of the four valves needs to stay open for the color mixture based on the specified recipe and the number of tins that have to be filled.
- ③ LEDs below the tanks indicate that the valves are opened.
- ④ The "LAD\_Tanks\_Filling\_Level" program block is executed at the same time as the filling. The program block calculates the quantity remaining in the tank for the tank fill level. The fill levels of the tanks are stored in the global data block "Filling".
- ⑤ The fill level indicators in the HMI screen are directly linked with the global data block and are updated with each runtime acquisition cycle.

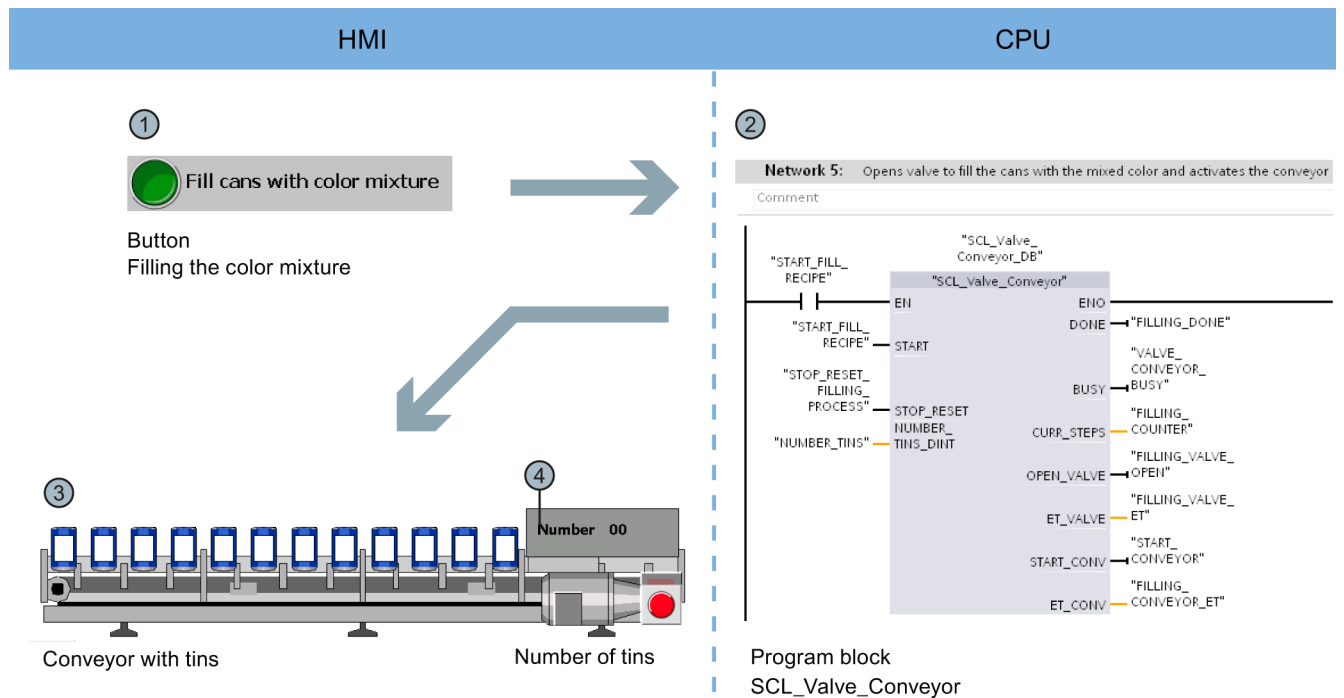
Starting the mixing process



- ① The "Start mixing process" button starts the mixer of the color mixing plant in the HMI screen "Start screen".
- ② The "LAD\_Mixer" program block is called for this purpose at the CPU end. It activates the mixer for three seconds.
- ③ The activation of the mixer is indicated by flashing in the HMI screen.

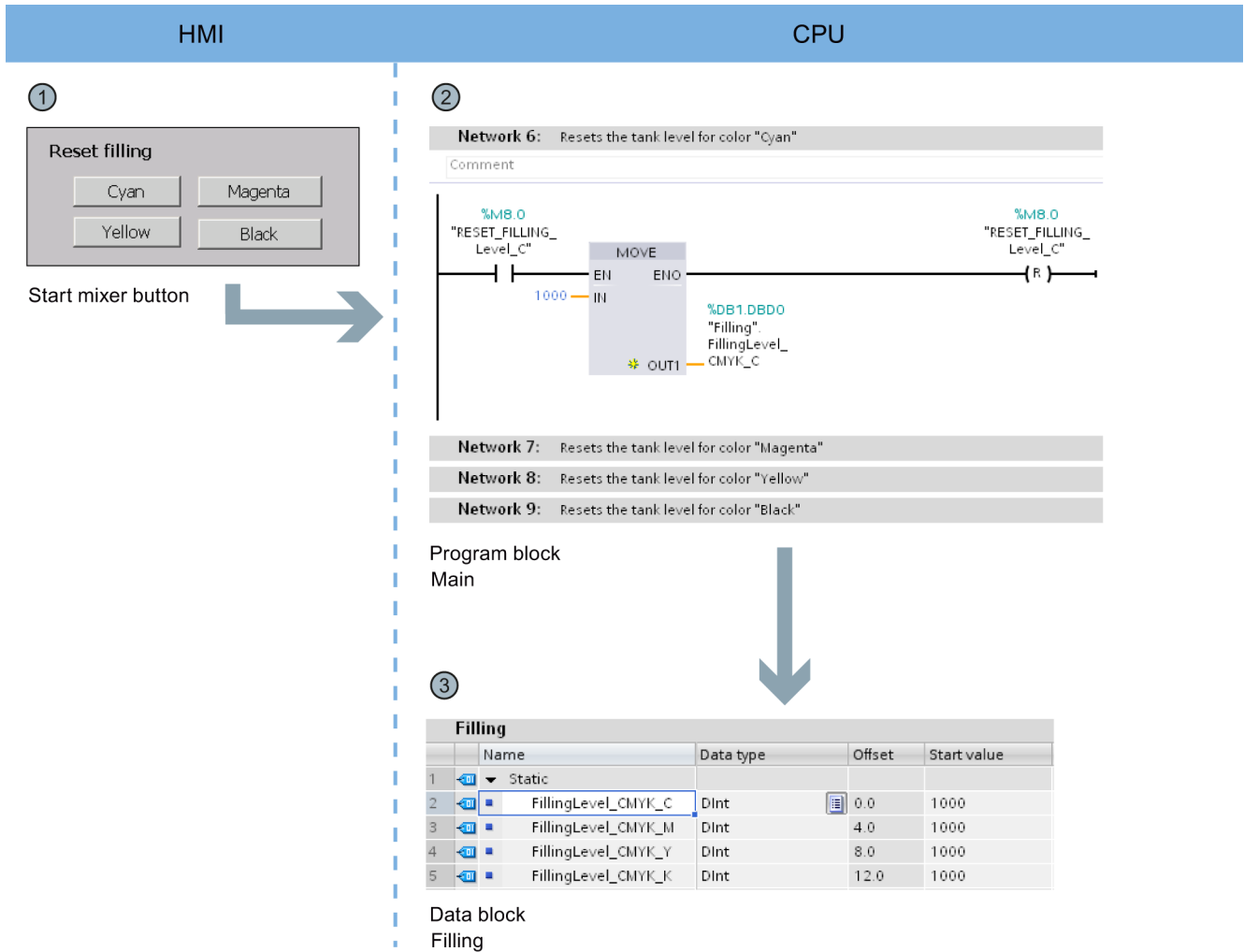


## Filling the color mixture



- ① The "Fill Color Mixture" button starts the filling of the tins in the HMI screen "Start screen".
- ② The "SCL\_Valve\_Conveyor" program block is activated for this purpose at the CPU end; it controls the valves and the conveyor belt.
- ③ The tins are animated according to the movement of the conveyor belt in the HMI screen.
- ④ A counter indicates the number of tins that have already been filled.

### Resetting the fill level



- ① There is a button to reset the fill level of each of the four color tanks in the HMI screen "Start screen".
- ② The reset of the respective fill level has been implemented in the "Main" program block in networks 6 to 9.
- ③ Networks 6 to 9 reset the values to the Start value in the global data block "Filling".

## Hardware section

### 2.1 Introduction

The new SIMATIC S7-1500 controller family with the Totally Integrated Automation Portal (TIA Portal) offers you numerous new options to further increase the productivity of your machines and to make the engineering process even more efficient. Explore the options in this Getting Started.

In the first basic steps, you will get to know the new hardware better. We will also show you how to configure and program the SIMATIC S7-1500 with SIMATIC STEP 7 V13 (TIA Portal). The connection of a SIMATIC HMI Comfort Panel with SIMATIC WinCC Advanced V13 (TIA Portal) or SIMATIC WinCC Professional V13 (TIA Portal) completes the basic steps.

#### 2.1.1 Requirements

##### Hardware requirements

To implement the hardware section of this Getting Started, you will need:

- 1 × CPU 1511-1 PN (6ES7511-1AK00-0AB0)
- 1 × S7-1500 load current supply PM 70W 120/230VAC (6EP1332-4BA00)
- 1 × Mounting rail (6ES7590-1AB60-0AA0)
- 1 × digital input module DI 16x24VDC SRC BA (6ES7521-1BH50-0AA0)
- 1 × digital output module DQ 16x24VDC/0.5A ST (6ES7522-1BH00-0AB0)
- 2 × Front connectors (6ES7592-1AM00-0XB0)
- 1 × SIMATIC Memory Card with at 4 MB (e.g. 6ES7954-8LBxx-0AA0)
- 1 × Ethernet cable

The hardware mentioned above is also part of the following starter package:

Starter package S7-1500 with software: 6ES7511-1AK00-4YB5

## Software requirements

To implement the software section of this Getting Started, you will need:

- SIMATIC STEP 7 Professional V13
- SIMATIC WinCC Advanced V13 or SIMATIC WinCC Professional V13



### WARNING

#### Severe personal injury may result

The S7-1500 automation system in plants or systems is governed by specific standards and regulations, based on the relevant field of application. Please observe the applicable safety and accident prevention regulations such as IEC 60204-1 (general machine safety requirements).

Failure to observe these regulations can result in serious injuries and damages to machinery and facilities.

## 2.1.2 Additional information

Detailed information on the hardware used is available here:

- CPU 1511-1 PN (6ES7511-1AK00-0AB0)  
(<http://support.automation.siemens.com/WW/view/en/68020492>)
- S7-1500 load current supply PM 70W 120/230VAC (6EP1332-4BA00)  
(<http://support.automation.siemens.com/WW/view/en/68036174>)
- DI 16x24 V DC SRC BA digital input module (6ES7521-1BH50-0AA0)  
(<http://support.automation.siemens.com/WW/view/en/59191844/>)
- DQ 16x24 V DC/0.5A ST digital output module (6ES7522-1BH00-0AB0)  
(<http://support.automation.siemens.com/WW/view/en/59193401>)

## 2.2 Installing the assembly

### 2.2.1 Overview

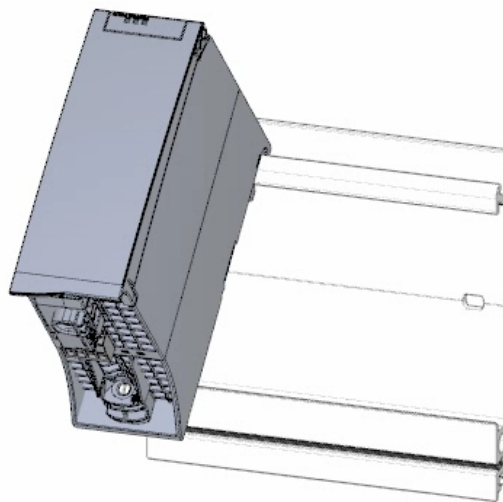
#### Mounting the assembly

You mount the structure in this section.

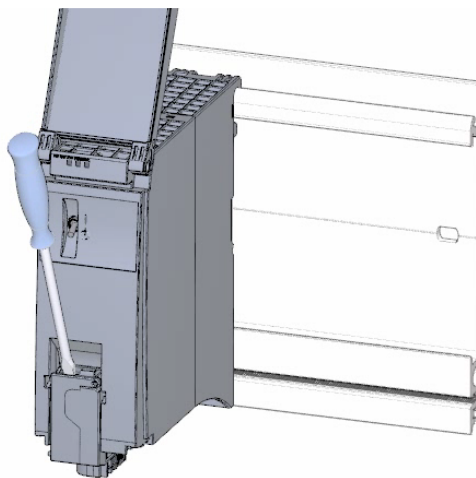
### 2.2.2 Installing the assembly

#### Procedure

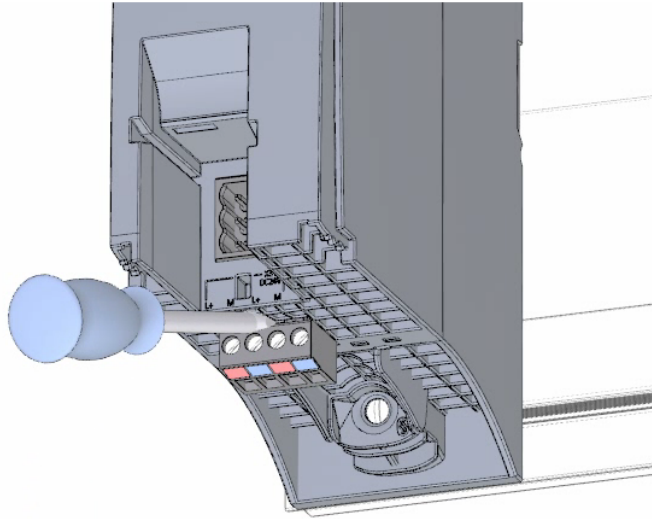
1. Mount the load current supply (PM) on the mounting rail.



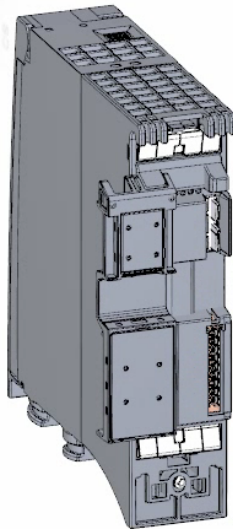
2. Open the front cover and pull out the mains connection plug.



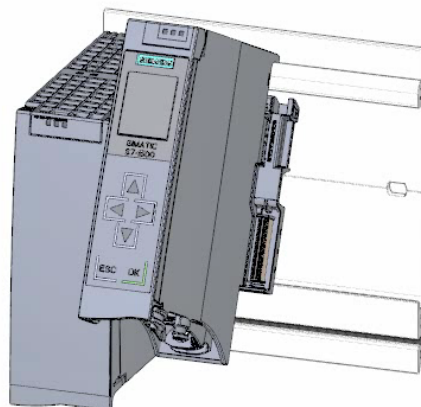
3. Remove the 4-pole connection plug and screw the load current supply (PM) tight.



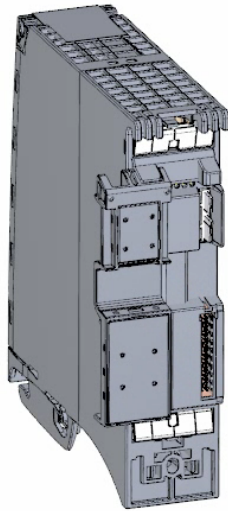
4. Insert the U-connector into the back of the CPU.



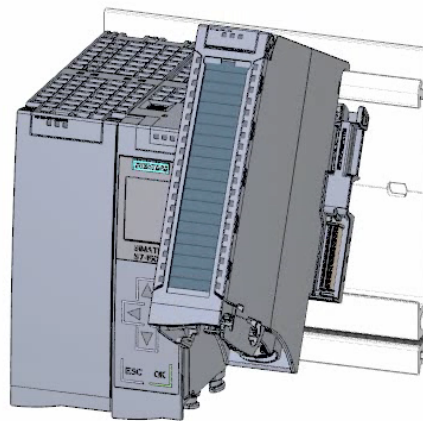
5. Mount the CPU on the mounting rail and screw tight.



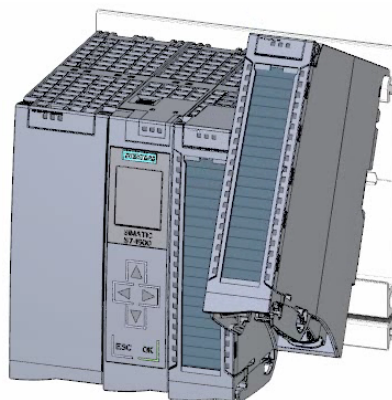
6. Insert the U-connector into the back of the digital input module.



7. Attach the digital input module to the mounting rail and screw tight.



8. Attach the digital output module to the mounting rail and screw tight.



## Result

The assembly has been mounted.

## 2.3 Wiring

### 2.3.1 Overview

#### Wiring the assembly

You mount the assembly in this section.



The mains cable for the load current supply must not be connected to the power supply during wiring.



## 2.3.2 Wiring rules

Operation of an S7-1500 CPU in plants or systems is defined by special set of rules and regulations, based on the relevant field of application.

You can find the general rules and regulations for operating the S7-1500 in the S7-1500 system description (<http://support.automation.siemens.com/WW/view/en/59191792>).

### Wiring rules for the CPU

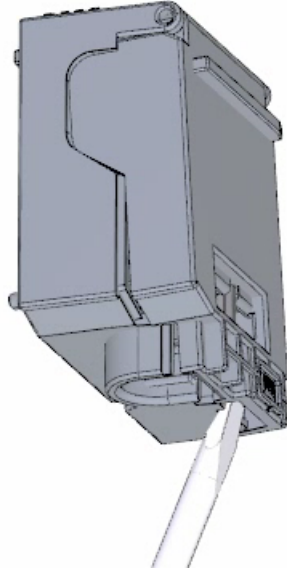
Wiring rules...		CPU	40-pin front connector (screw-type connection)	Load power supply
Connectible wire cross-sections for solid wires		—	up to 0.25 mm <sup>2</sup>	—
		—	AWG*: 24	—
Connectible wire cross-sections for stranded wires	Without wire end ferrule	0.25 to 2.5 mm <sup>2</sup>	0.25 to 1.5 mm <sup>2</sup>	1.5 mm <sup>2</sup>
		AWG*: 24 to 16	AWG*: 24 to 16	AWG*: 16
	With wire end ferrule	0.25 to 2.5 mm <sup>2</sup>	0.25 to 1.5 mm <sup>2</sup>	1.5 mm <sup>2</sup>
		AWG*: 24 to 16	AWG*: 24 to 16	AWG*: 16
Number of wires per connection		1	1 or a combination of 2 cables up to 1.5 mm <sup>2</sup> (total) in the same wire end ferrule	1
Length of stripped wires		10 to 11 mm	10 to 11 mm	7 to 8 mm
End sleeves according to DIN 46228	Without plastic sleeve	Design A, 10 mm long	Design A, 10 mm and 12 mm long	Design A, 7 mm long
	with plastic sleeve 0.25 to 1.5 mm <sup>2</sup>	Design E, 10mm long	Design E, 10 mm and 12 mm long	Design A, 7 mm long
Sheath diameter		—	—	8.5 mm
Tool		3 to 3.5 mm Phillips screwdriver, conic design	3 to 3.5 mm Phillips screwdriver, conic design	3 to 3.5 mm Phillips screwdriver, conic design
Connection method		Push-in terminal	Screw terminal	Screw terminal
Tightening torque		—	from 0.4 Nm to 0.7 Nm	from 0.5 Nm to 0.6 Nm

\* AWG: American Wire Gauge

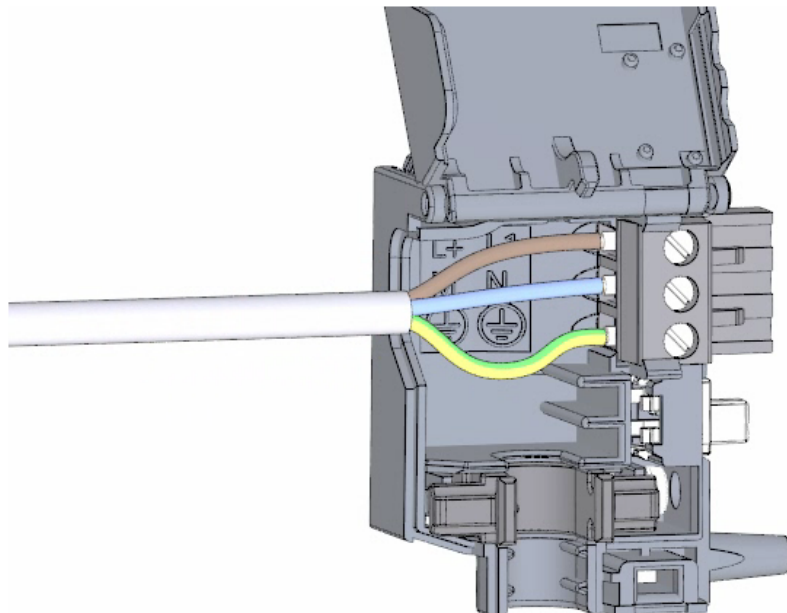
### 2.3.3 Wiring the mains connection plug

#### Procedure

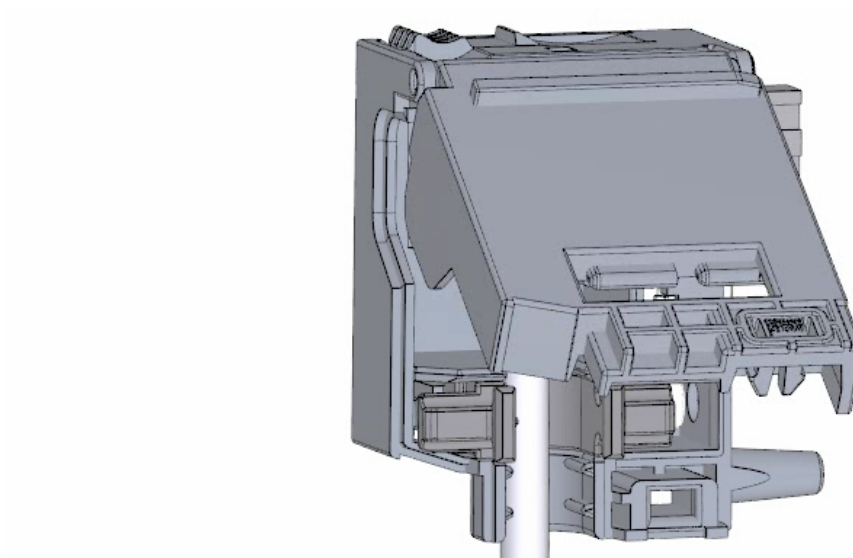
1. Pry off the connector cover using a suitable tool.



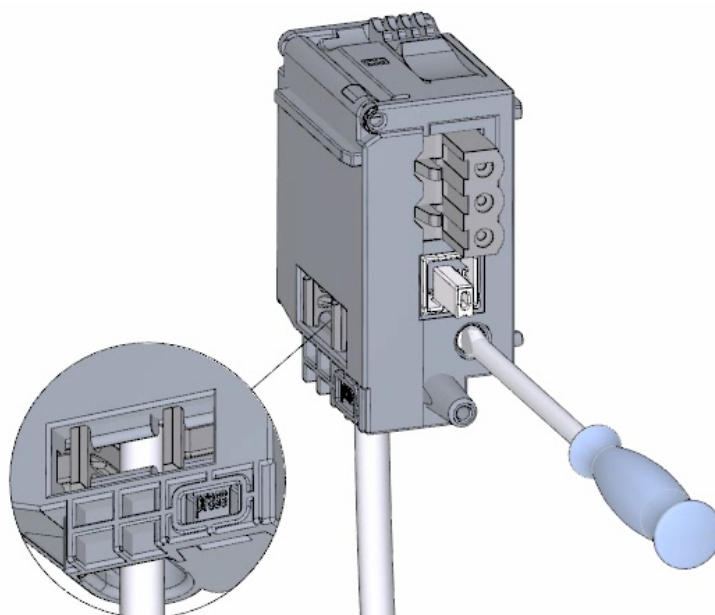
2. Connect the mains cable in the plug according to the connection diagram. You will find information on which voltage the plug is approved for on the side of the plug. You select the voltage by inserting the coding element accordingly on the back of the plug.



3. Close the cover.



4. Tighten the screw on the front of the mains connection plug.



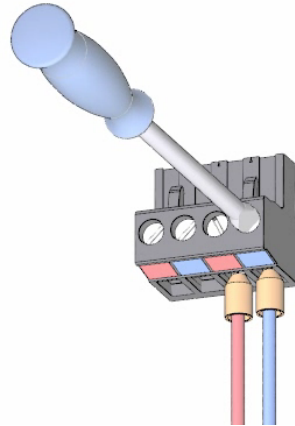
## Result

The mains connection plug is now wired.

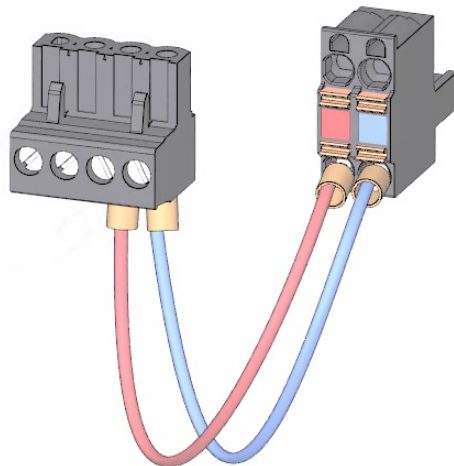
### 2.3.4 Wiring the load current supply (PM) to the CPU

#### Procedure

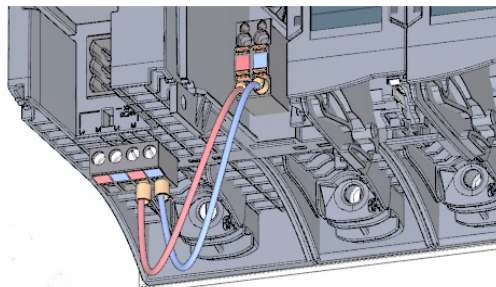
1. Wire the 4-pin connector plug of the load current supply (PM).



2. Wire the 4-pin connector plug with the 4-pin mains connection plug of the CPU.



3. Connect the load current supply (PM) to the CPU.



#### Result

The load current supply is now wired to the CPU.

## 2.3.5 Potential bridge circuits

### Application of the potential bridge circuits

If you want to supply the load groups with the same potential (non-isolated), use the potential circuit bridges supplied for the front connector. This means that you avoid having to wire a clamping unit with two wires.

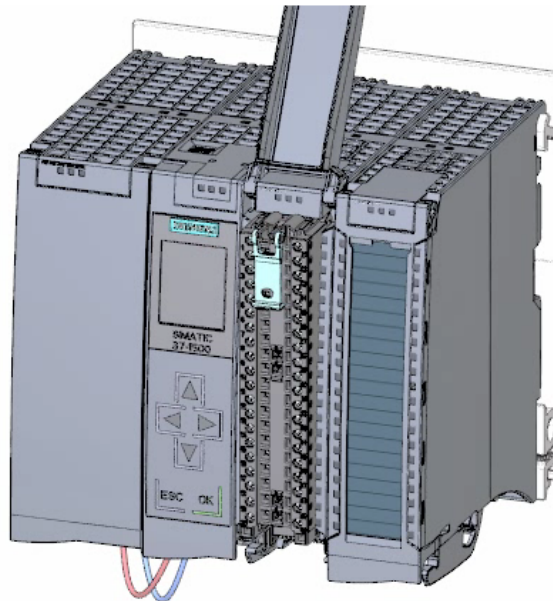
### Tip

Use the terminals 40 (M) and 39 (L+) on the front connector to loop the potential to the next module.

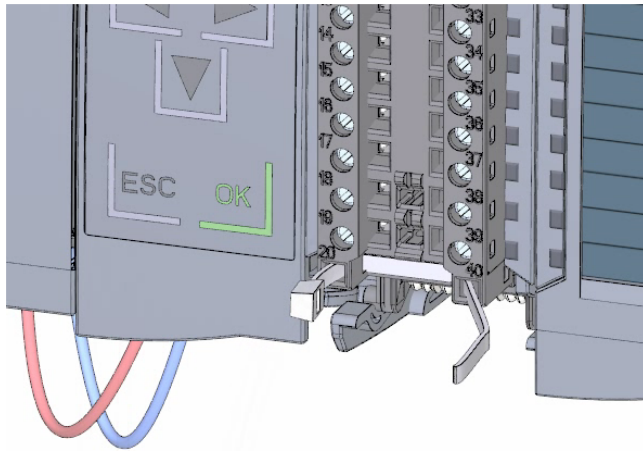
## 2.3.6 Wiring the digital input module

### Procedure

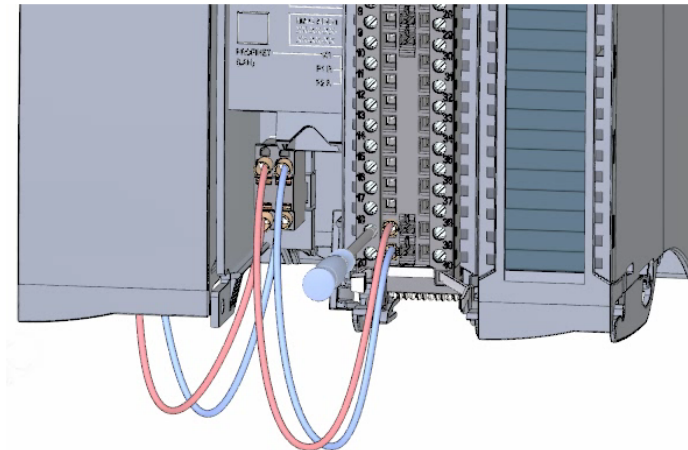
1. Insert the front connector into the pre-wiring position. There is no electrical connection between the front connector and the module in the pre-wiring position.



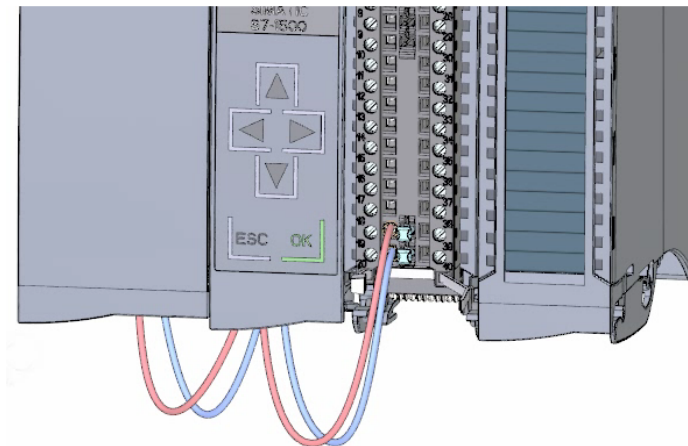
2. Thread in the cable tie.



3. Connect the supply voltage 24 V DC to the terminals 20 (M) and 19 (L+).



4. Insert the potential circuit bridges between the two bottom terminals.



## Result

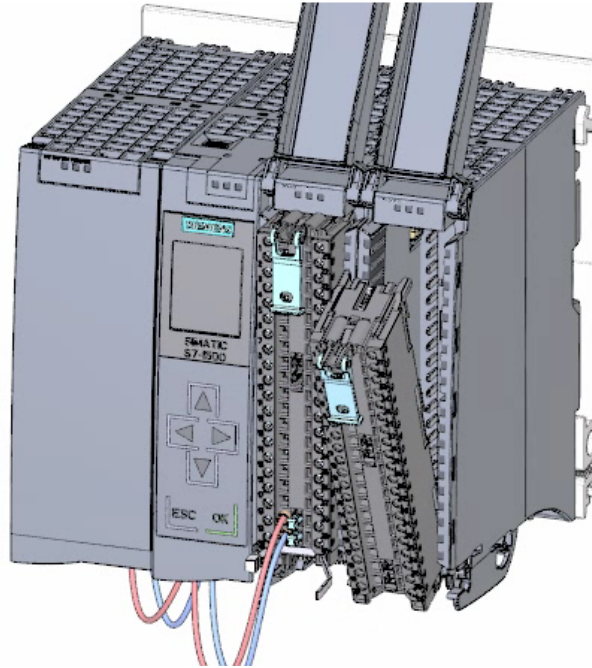
The digital input module is now wired.



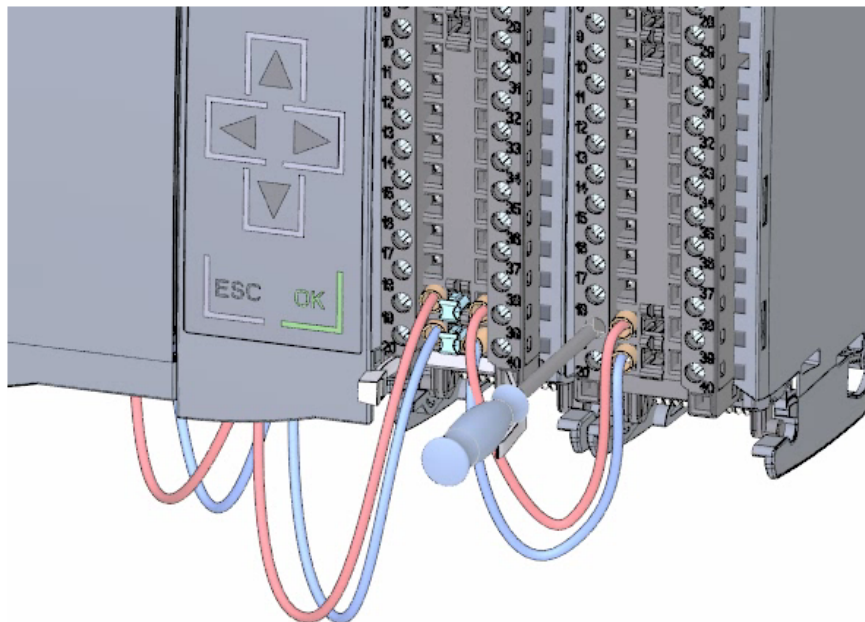
## 2.3.7 Wiring the digital output module

### Procedure

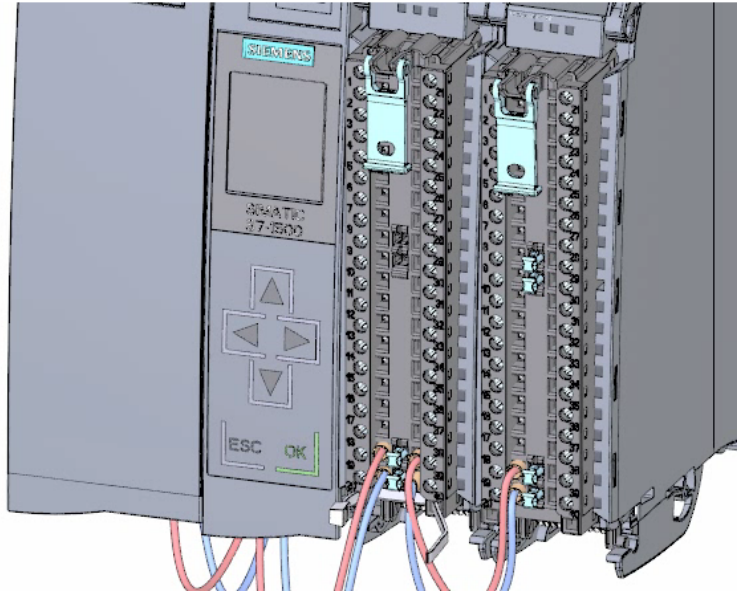
1. Insert the front connector into the pre-wiring position.



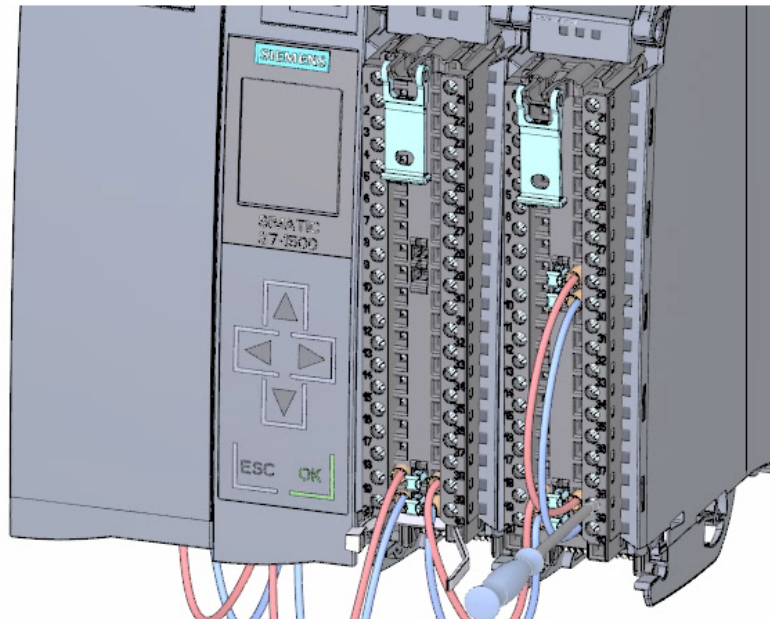
2. Use terminals 40 (M) and 39 (L+) from the digital input module to feed the supply voltage DC 24 V from the digital input module to terminals 20 (M) and 19 (L+).



3. Connect the four potential circuit bridges.



4. Connect the terminals 30 and 40, as well as 29 and 39 to each other.



## Result

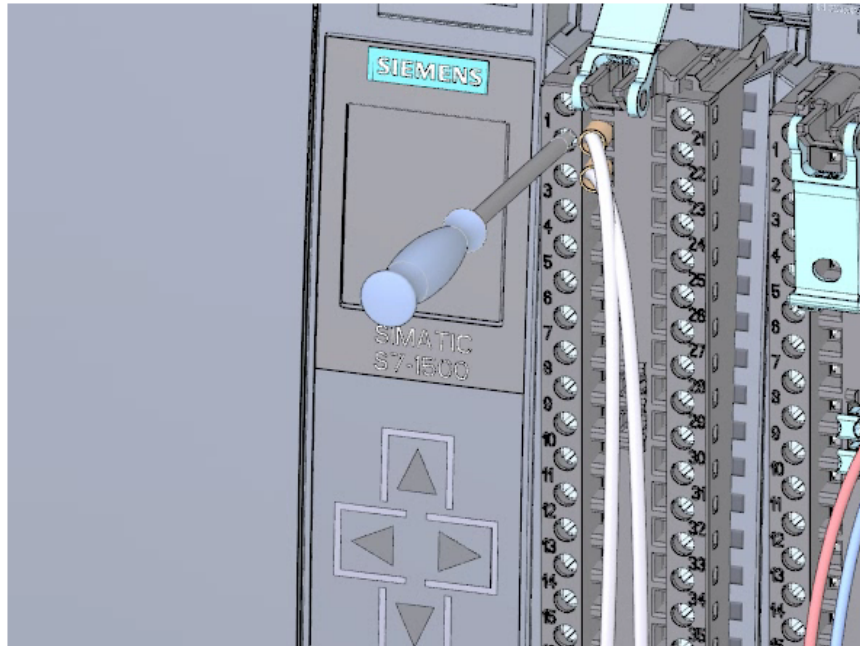
The digital output module is now wired.



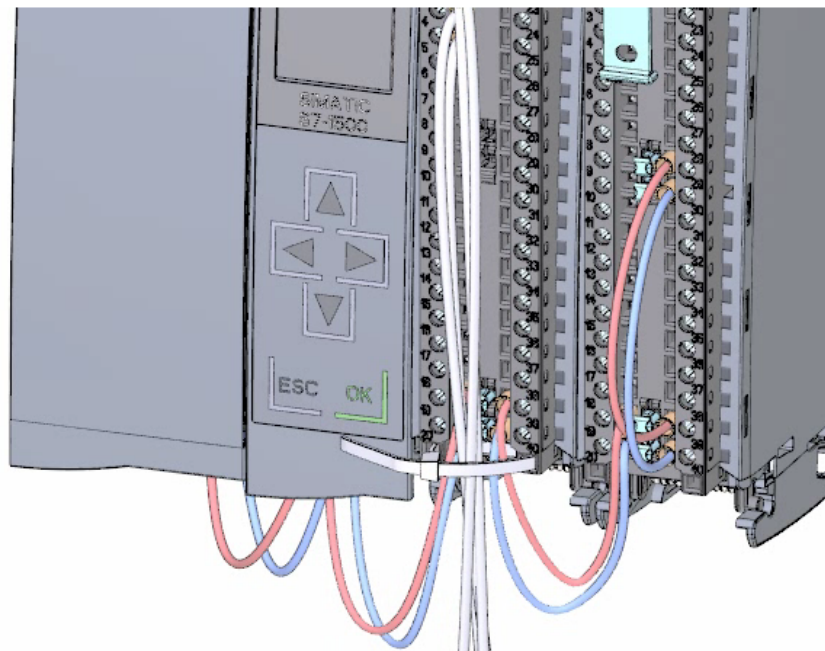
## 2.3.8 Wiring front connectors

### Procedure

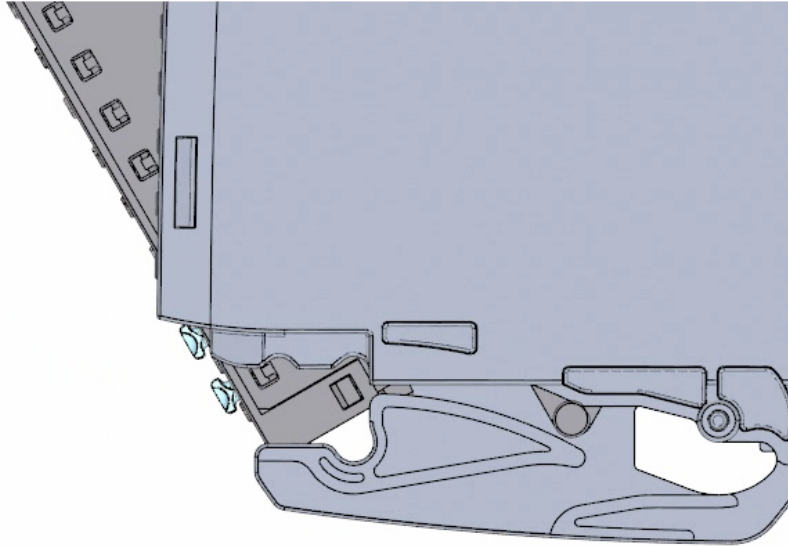
1. Connect the individual wires according to the connection diagram on the inner side of the front cover in the terminal and screw tight.



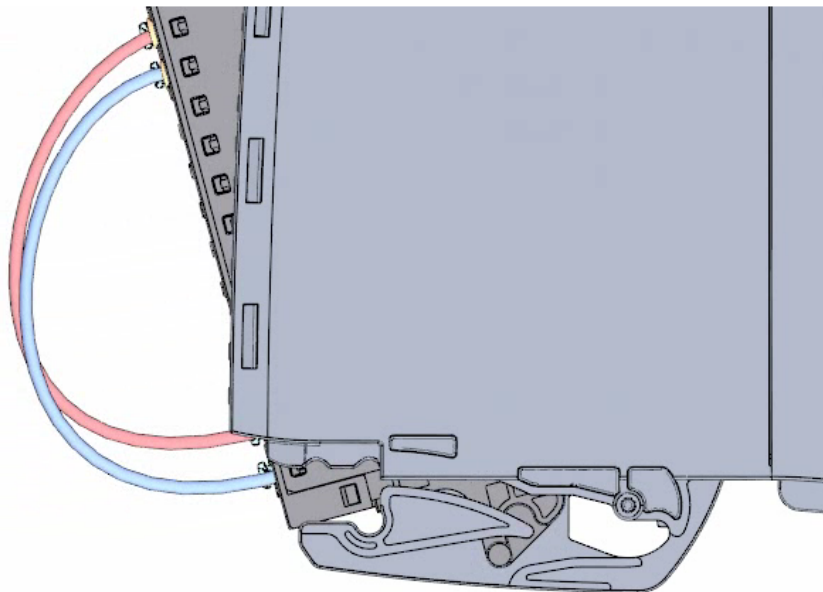
2. For strain relief, run the cable tie around the cable harness and pull tight.



3. Move the front connector from the pre-wiring position to its final position. By doing this, you create an electrical connection between the front connector and the module.



4. **Tip:** Pre-wired front connectors, e.g. for replacing modules, can be inserted directly.



## Result

The front connectors are now wired.

## 2.4 Power on

### 2.4.1 Overview

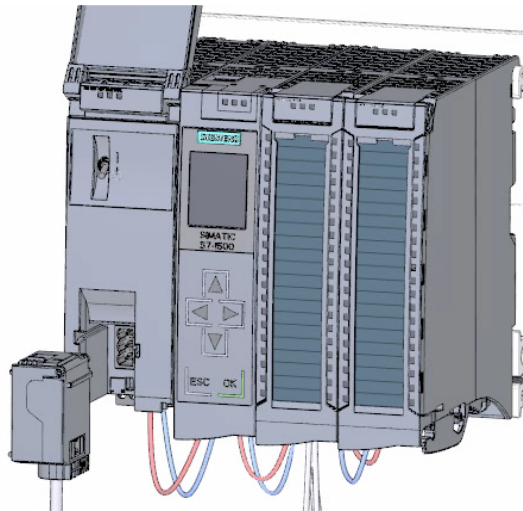
#### Turning on the CPU for the first time

You turn on the CPU for the first time in this section.

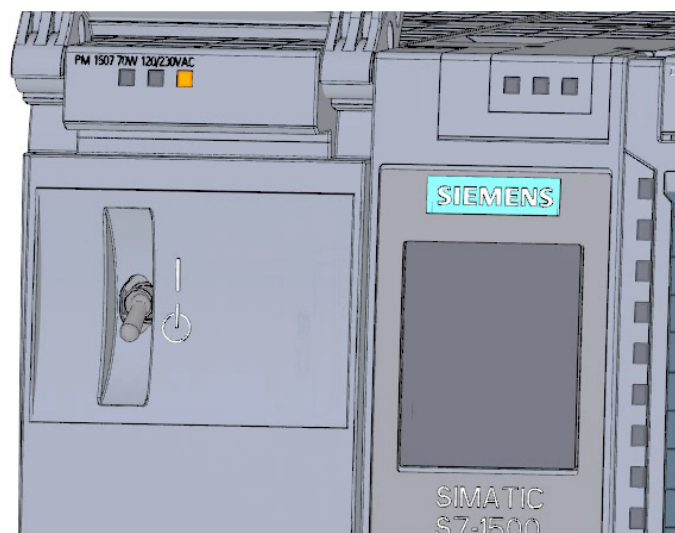
### 2.4.2 Power on

#### Procedure

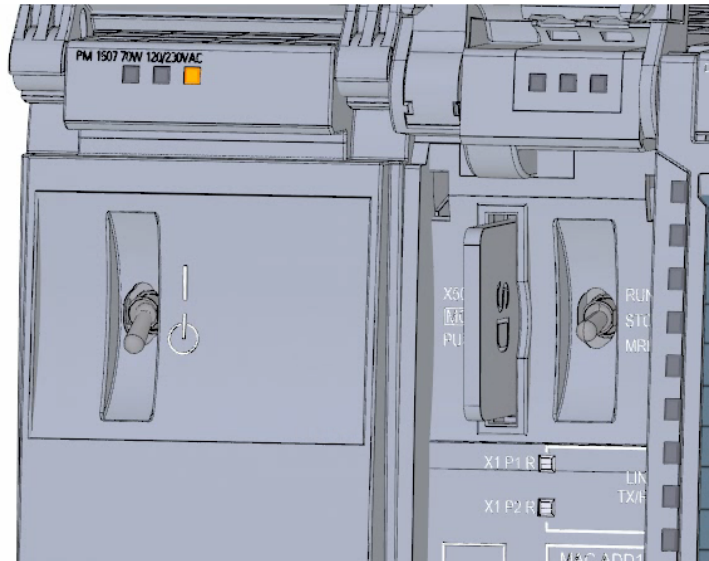
1. Insert mains connection plug of the load current supply (PM).



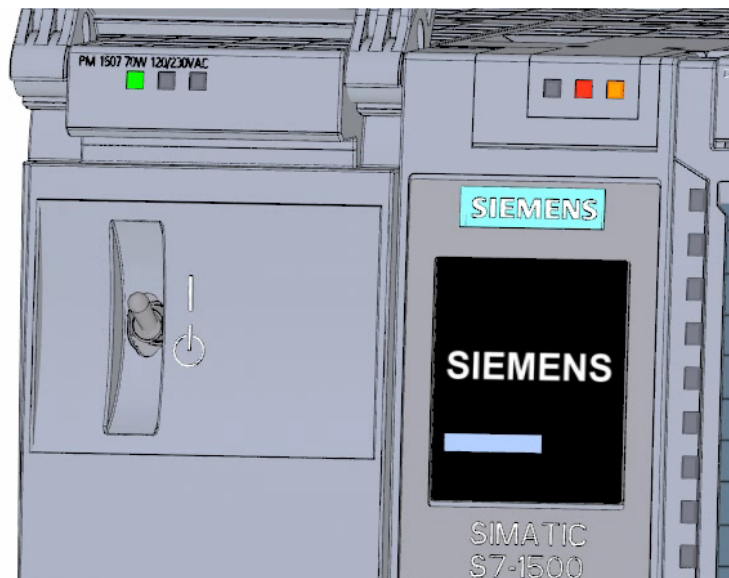
2. Connect the mains connection plug to the power supply.



3. Insert a blank SIMATIC memory card into the CPU.



4. Move the switch for the load current supply (PM) to the position RUN.  
The CPU starts up.



## Result

The CPU starts up and is in STOP mode.

### 2.4.3 Assign IP address via the display

In this step you set the IP address and the subnet mask for the CPU.

#### Procedure

1. Navigate to "Settings".
2. Select "Addresses".
3. Select the interface "X1 (IE/PN)".
4. Select the menu item "IP Addresses".
5. Set the IP address 192.168.0.10.
6. Press the "right" arrow key on the module.
7. Set the subnet mask 255.255.255.0.
8. Press the "down" arrow key on the module to select the menu item "Apply" and confirm the setting with "OK"

#### Result

You have now assigned an IP address and the subnet mask for the interface "X1 (IE/PN)".

## Software section

### 3.1 Creating the project and hardware

#### 3.1.1 Introduction to the TIA Portal

##### Introduction

The Totally Integrated Automation Portal, referred to as TIA Portal in the following, offers all the functions you need for implementing your automation task assembled in a single, cross-software platform.

The TIA Portal is the first shared working environment for integrated engineering with the various SIMATIC systems made available within a single framework. The TIA Portal therefore also enables reliable, convenient cross-system collaboration for the first time.

All required software packages, from hardware configuration and programming to visualization of the process are integrated in a comprehensive engineering framework.



## Advantages of working with the TIA Portal

The following features provide efficient support during the realization of your automation solution when working with the TIA Portal:

- **Integrated engineering with a uniform operating concept**  
Process automation and process visualization go "hand-in-hand".
- **Consistent, centralized data management with powerful editors and universal symbols**  
Data created once is available in all editors. Changes and corrections are automatically applied and updated within the entire project.
- **Comprehensive library concept**  
Use the ready-made instructions and pre-existing parts of the project again and again.
- **Multiple programming languages**  
Five different programming languages are available for implementing your automation task.

### 3.1.2 Creating a project

#### Introduction

In the following step, you will create a new project.

All data which is generated during the creation of an automation solution is saved in the project file. The data is stored in the form of objects. Within the project, the objects are arranged in a tree structure (project hierarchy).

The project hierarchy is based on the devices and stations along with the configuration data and programs belonging to them.

#### Requirement

You need the following hardware and software equipment to create the project:

- Hardware:
  - The CPU 1511-1 PN that was installed and wired in the hardware section of the Getting Started.
  - An Ethernet connection to your programming device/PC.
- Software:

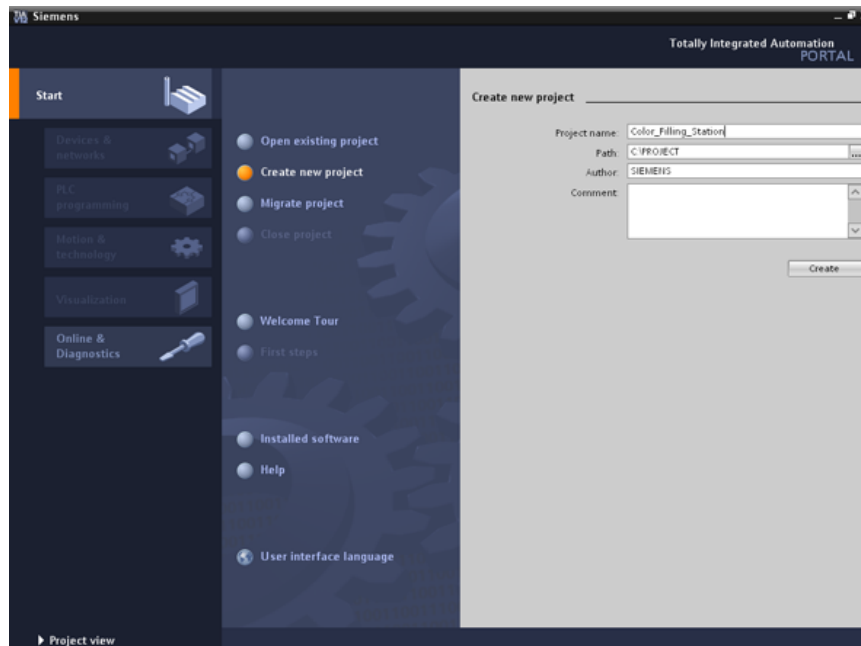
The following software packages must be installed and executable on your programming device/PC:

  - SIMATIC STEP 7 Professional V13
  - SIMATIC WinCC Advanced V13 or SIMATIC WinCC Professional V13

#### Creating a new project

To create a new project, follow these steps:

1. Click "Create new project".
2. Enter a name for your project.

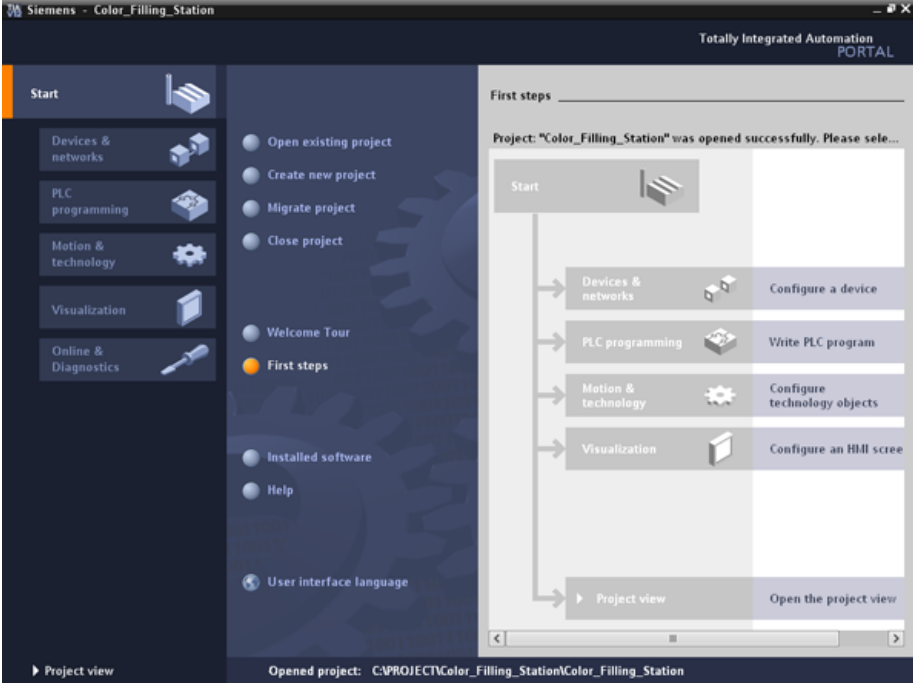


3. Click "Create" to create the new project.



Result

The project has been created. All data, such as the hardware configuration, the CPU programming and the visualization in HMI, is saved in the project.



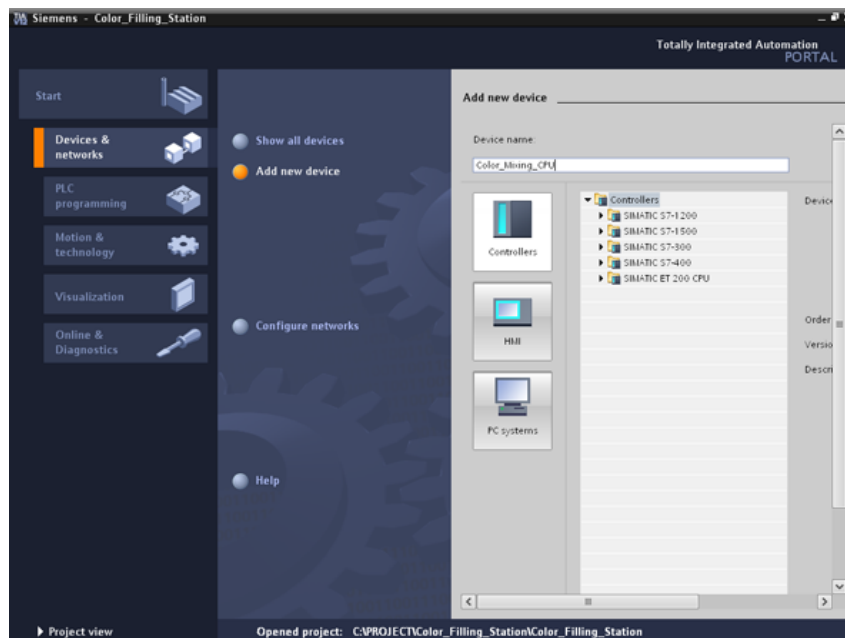
### 3.1.3 Creating an S7-1500 CPU

#### Introduction

In the following step, you will create an unspecified CPU. Unspecified CPUs are placeholders for specific CPUs from the hardware catalog which will be defined later.

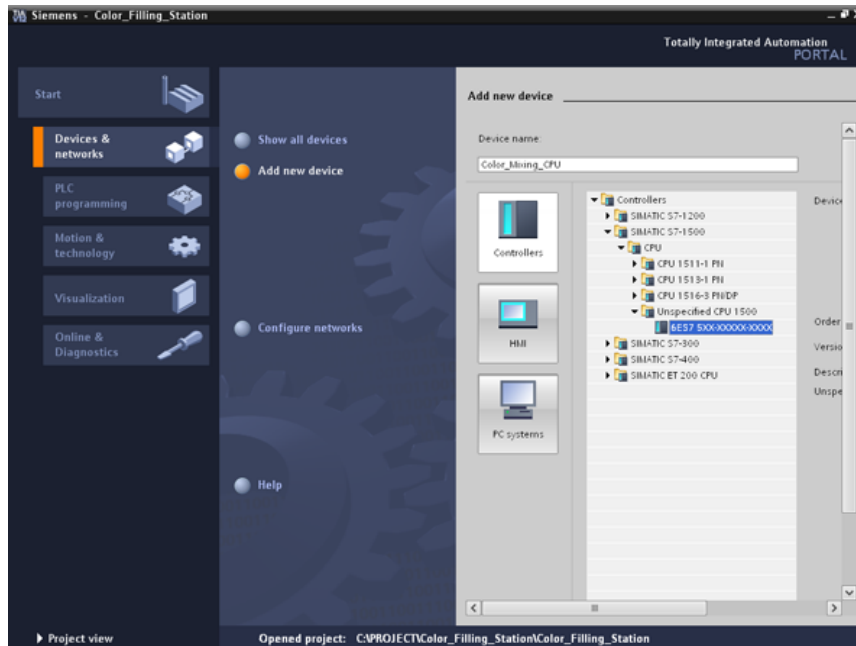
#### Procedure

1. Open the "Devices & Networks" portal.
2. Insert a new device.
3. Enter "Color\_Mixing\_CPU" as the name for the CPU.



4. Open the "SIMATIC S7-1500" folder.

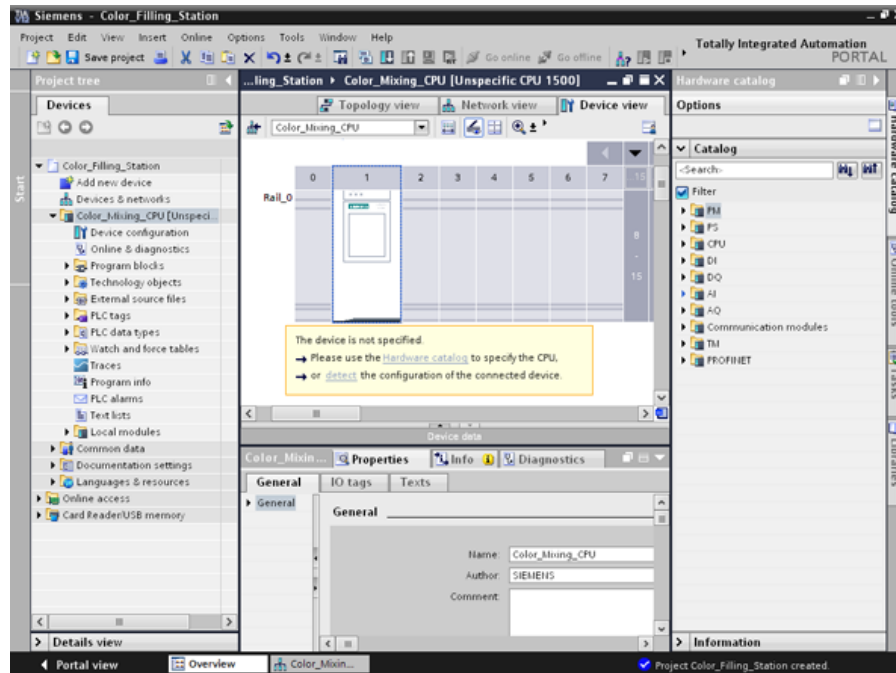
5. Select the CPU which has not yet been specified.



6. Create the CPU with a double-click.

## Result

The unspecified CPU is created in the project file. Contents of the user program can already be created at this point for this CPU.



### 3.1.4 Running the hardware detection

#### Introduction

In the following section, you will use the hardware detection function to read the CPU type. Run an LED flashing test during hardware detection. The LED flashing test activates the LEDs on a detected device. You may also use this function to verify that the correct device was selected in a hardware configuration consisting of several devices.

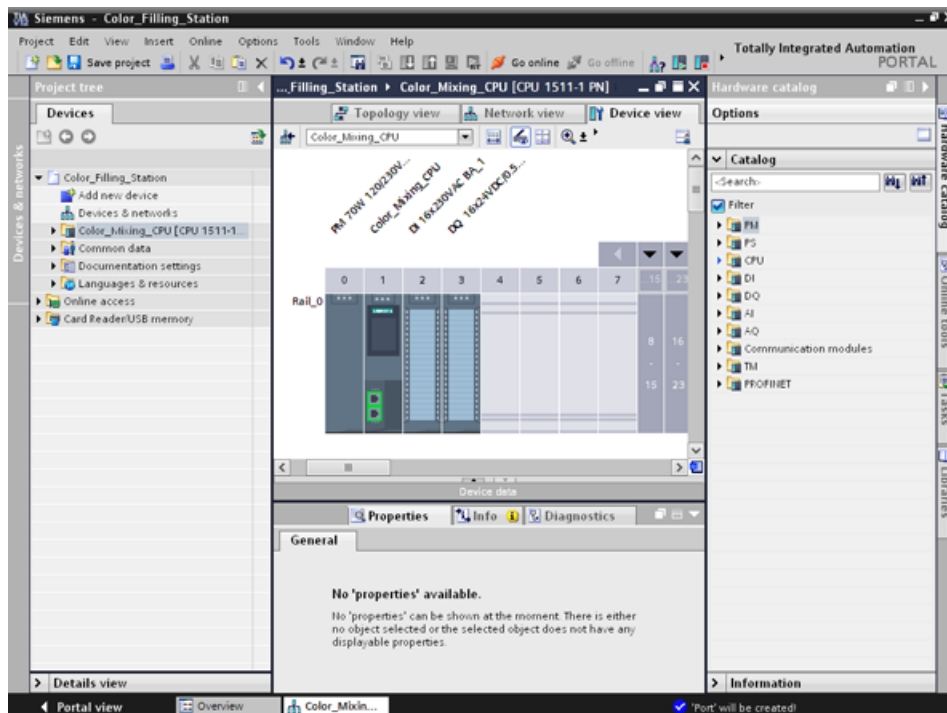
#### Procedure

1. Select the unspecified CPU in the project tree.
2. Select the "Hardware detection" function from the "Online" menu.  
Option 2: Click on the yellow framed alarm in the device view.
3. Select the "PN/IE" entry as the type of PG/PC interface.
4. Select the PG/PC interface.
5. Click the "Show all compatible devices" option.
6. Select the CPU from the compatible devices in the subnet.
7. Select the "Flash LED" check box to run a flashing test.
8. Click "Detect" to replace the unspecified CPU with the necessary CPU type.

#### Result

The CPU type is read out. The correct device name is appended in brackets to your CPU name in the project tree.

The CPU and modules used are displayed in the hardware configuration.



### 3.1.5 Creating ET 200 interface modules

#### Introduction

In the following section, you will create two distributed I/O systems in the hardware configuration:

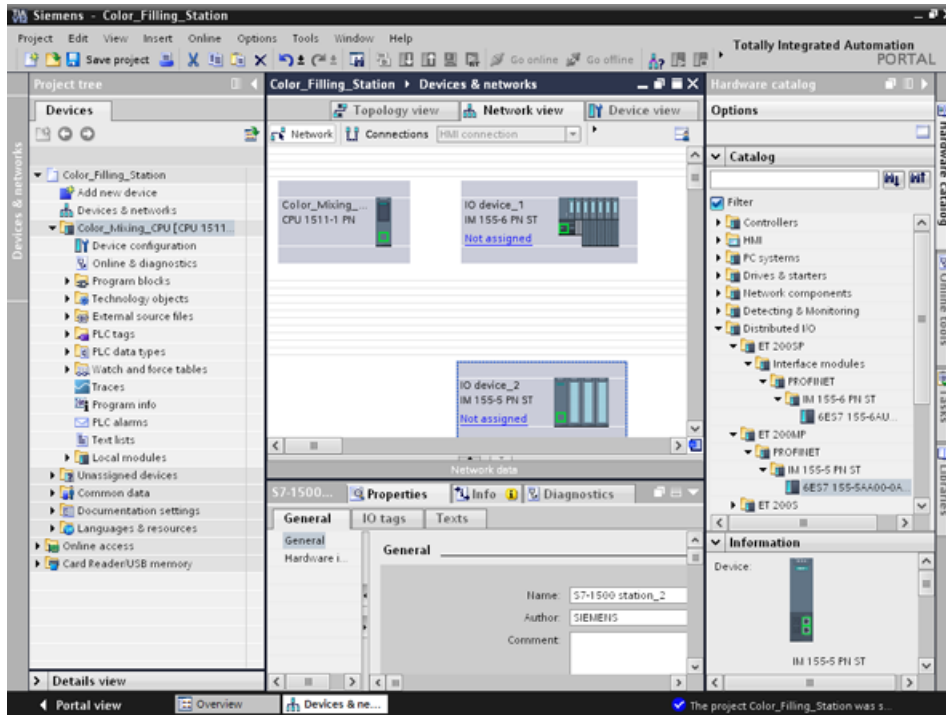
- An ET 200SP distributed I/O system, which basically consists of the following components:
  - An interface module for communication with the CPU.
  - Up to 32 modules that can be inserted in any combination.
  - A server module that completes the configuration.
- An ET 200MP distributed I/O system, which consists of the following components:
  - The interface module for communication with the CPU.
  - Up to 30 modules, each one providing up to 32 channels.

#### Procedure

1. Open the "Hardware catalog".
2. Change to the "Network view".
3. Open the "Distributed I/O" and "ET 200SP" folders.
4. Open the "IM 155-6 PN ST" folder.
5. Drag-and-drop the "6ES7 155-6AU00-0BN0" interface module to the network view.
6. Open the "ET 200MP" folder.
7. Open the "IM 155-5 PN ST" folder.
8. Drag-and-drop the "6ES7 155-5AA00-0AB0" interface module to the network view.

## Result

The I/O systems have been created in the hardware configuration, but not yet assigned to the CPU 1511-1 PN. They are both displayed under "Unassigned devices" in the project view.



## Additional information

The SIMATIC ET 200 product family offers different scalable I/O systems to suit your specific application.

You will find more information about the SIMATIC ET 200 distributed I/O on the Internet at (<http://www.automation.siemens.com/mcms/distributed-io/en/>).

### 3.1.6 Networking ET 200 interface modules

#### Introduction

In the following section, you will create a PROFINET I/O system.

A PROFINET I/O system consists of the PROFINET IO controller and its assigned PROFINET IO devices:

- The CPU 1511-1 PN you already created is used as PROFINET IO controller.
- The two distributed I/O systems are used as PROFINET IO devices.

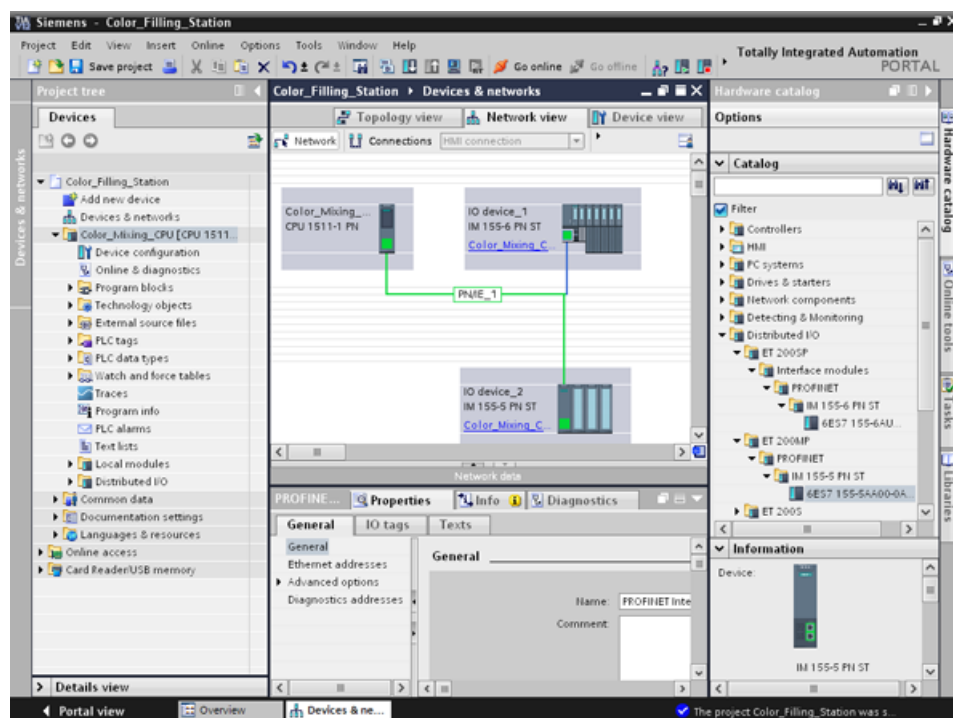
## Procedure

1. Drag-and-drop a connection from the interface of the IM 155-5 PN ST interface module to the CPU interface.
2. Create a second connection between the IM 155-6 PN ST interface module and the CPU.

## Result

The interface modules are assigned to the CPU as IO devices. Both distributed IO systems are displayed in the project tree in the "Distributed I/O" folder below the CPU.

A PROFINET I/O system was created automatically in the networking process and its properties are displayed in the network view.



### 3.1.7 Creating input and output modules and a server module for ET 200SP

#### Introduction

In the following section, you will create input and output modules for the ET 200SP.

---

#### Note

You need the server module to operate the input and output modules. These modules will fail if the server module is missing.

---

#### Maximum configuration per potential group

The number of I/O modules that can be used per potential group depends on the following factors:

1. Total power requirement of all I/O modules operated on this potential group
2. Total power requirement of all loads connected externally to this potential group

The sum of the total power calculated from 1. and 2. cannot exceed the current carrying capacity of the employed BaseUnit and the load supply voltage.

Set the "Potential group" parameter for a module as follows:

Parameters	Value range	Usage
Potential group	Use potential group of the left module (default setting)	if the total power consumption of all modules from the left + power consumption of the module is less than the current carrying capacity of the BaseUnit
	Enable new potential group	if the total power consumption of all modules from the left + power consumption of the module is greater than the current carrying capacity of the BaseUnit

You can find additional information on potential groups in the module manuals such as SIMATIC ET 200SP DI 8x24VDC HF digital input module (<http://support.automation.siemens.com/DE/view/en/66912542>).

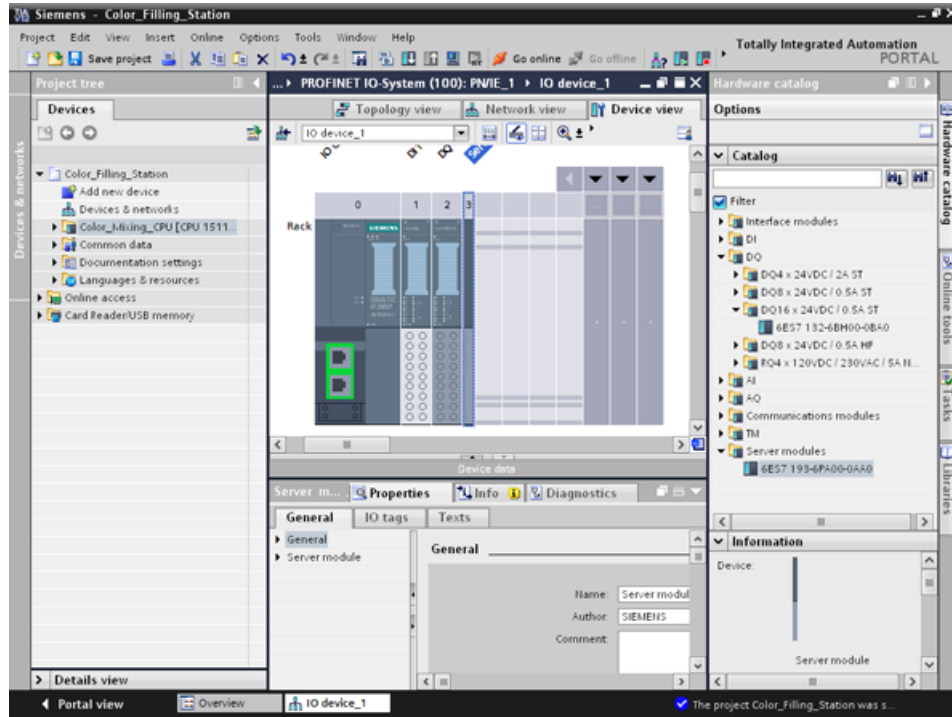
#### Procedure

1. Open the device view of ET 200SP.
2. Open the "DI" and "DI16 x DC24V ST" folders in the hardware catalog.
3. Drag-and-drop input module "6ES7 131-6BH00-0BA0" to slot 1 of the rail.
4. Open the "DQ" and "DQ16 x DC24V / 0.5A ST" folders.
5. Drag-and-drop output module "6ES7 132-6BH00-0BA0" to slot 2 of the rail.
6. Open the "Server modules" folder.
7. Drag-and-drop the server module "6ES7 193-6PA00-0AA0" to slot 3 of the rail.



## Result

You have created the input and output modules and the server module.



### 3.1.8 Creating input and output modules for ET 200MP

#### Introduction

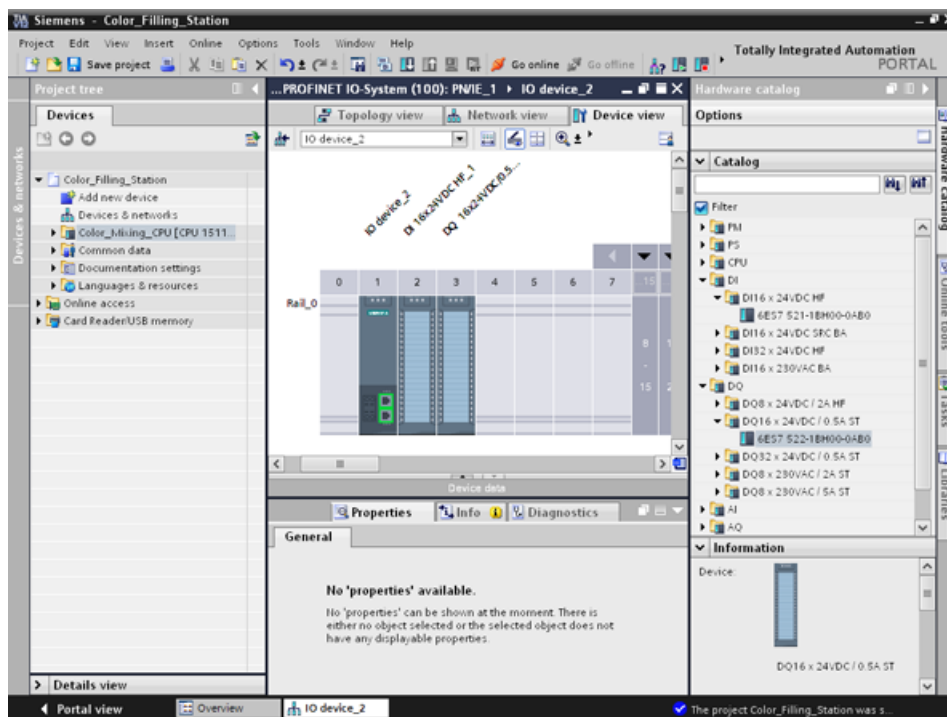
In the following section, you will create the input and output modules for ET 200MP.

#### Procedure

1. Open the device view of ET 200MP.
2. Open the "DI" and "DI16 x DC24V HF" folders in the hardware catalog.
3. Drag-and-drop input module "6ES7 521-1BH00-0AB0" to slot 2 of the rail.
4. Open the "DQ" and "DQ16 x DC24V / 0.5A ST" folders.
5. Drag-and-drop output module "6ES7 522-1BH00-0AB0" to slot 3 of the rail.

#### Result

You have created the input and output modules.



### 3.1.9 Assigning names for ET 200

#### Introduction

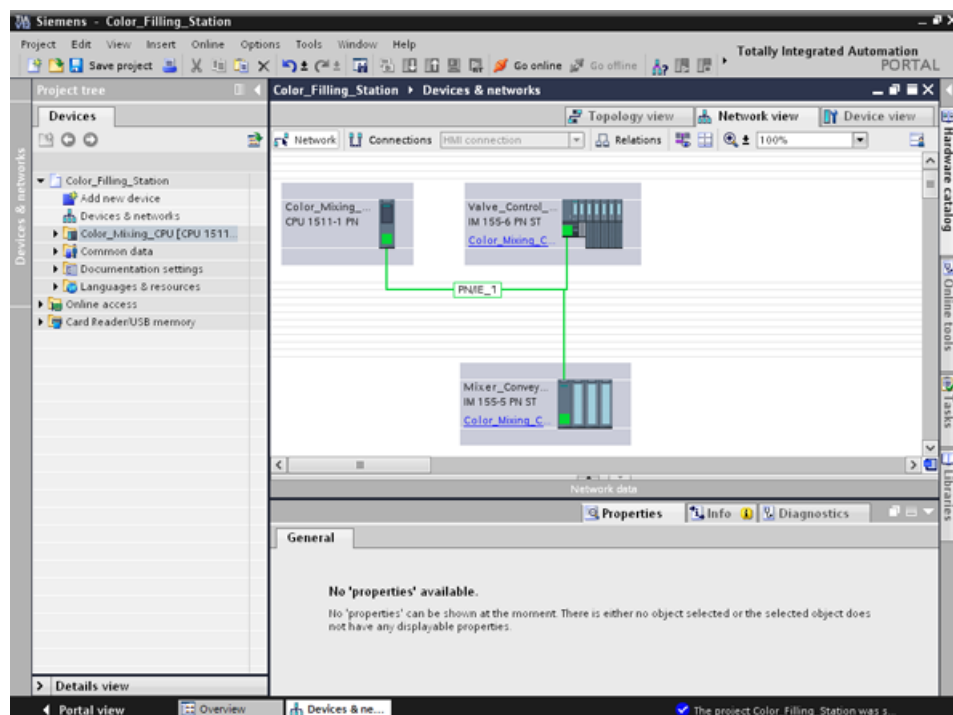
In the following section, you will assign project-specific names to the distributed I/O.

#### Procedure

1. Select ET 200SP.
2. Under **Properties > General** in the inspector window, enter the name "Valve\_Control\_Unit" in the "Name" field.
3. Select ET 200MP and enter the new name "Mixer\_Conveyor\_Control\_Unit".

#### Result

You have assigned the project-specific names.



## 3.2 Creating the program

### 3.2.1 Loading the block library

#### Introduction

In the following section, you will load the global library "ProgLib\_ColorFillingStation". This library contains the blocks and tag tables that you need for the example project. This library is available as a ZIP file under "Getting Started S7-1500 / TIA V13 ([http://www.automation.siemens.com/salesmaterial-as/interactive-manuals/getting-started\\_simatic-s7-1500/project/color\\_filling\\_station.zip](http://www.automation.siemens.com/salesmaterial-as/interactive-manuals/getting-started_simatic-s7-1500/project/color_filling_station.zip))". You need to unzip this library before you import it to your project.

#### Global libraries

Global libraries are used to store elements that you want to reuse in other projects. You must create global libraries explicitly.

The following libraries are provided in the standard package:

- "Buttons and Switches"

They offer a large selection of switches and buttons. The folders organize switches and buttons into categories. You can find the "System diagnostics indicator" object in the "DiagnosticsButtons" folder, for example. You use the "System diagnostics indicator" object for system diagnostics in your plant.

- "Monitoring and Control objects"

This provides complex operator control and display objects in several designs as well as suitable control lights, buttons and switches.

---

#### Note

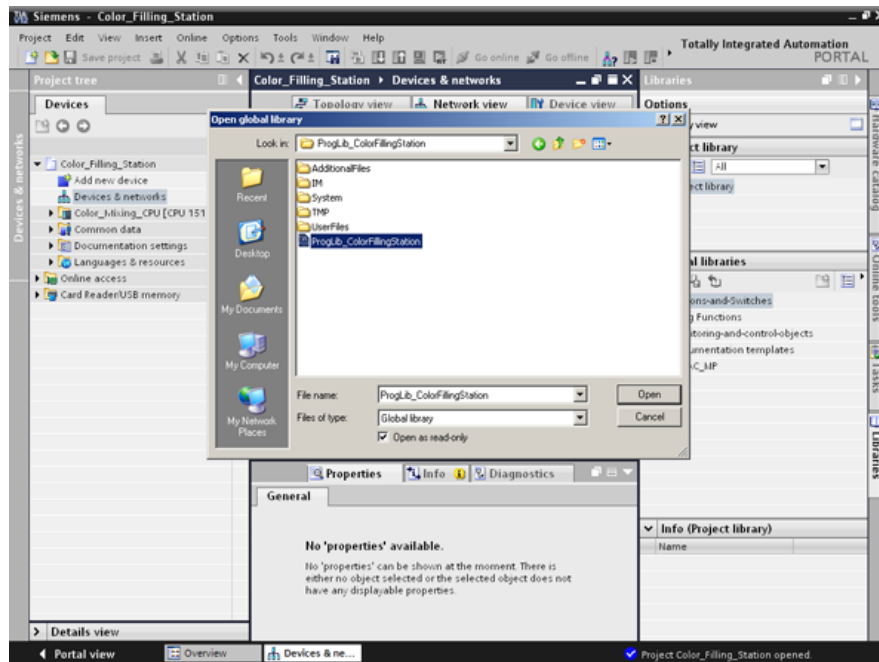
##### Library is write-protected

The "Open read-only" option is activated by default in the "Open global library" dialog. Click in the check box to open the library without write protection.

---

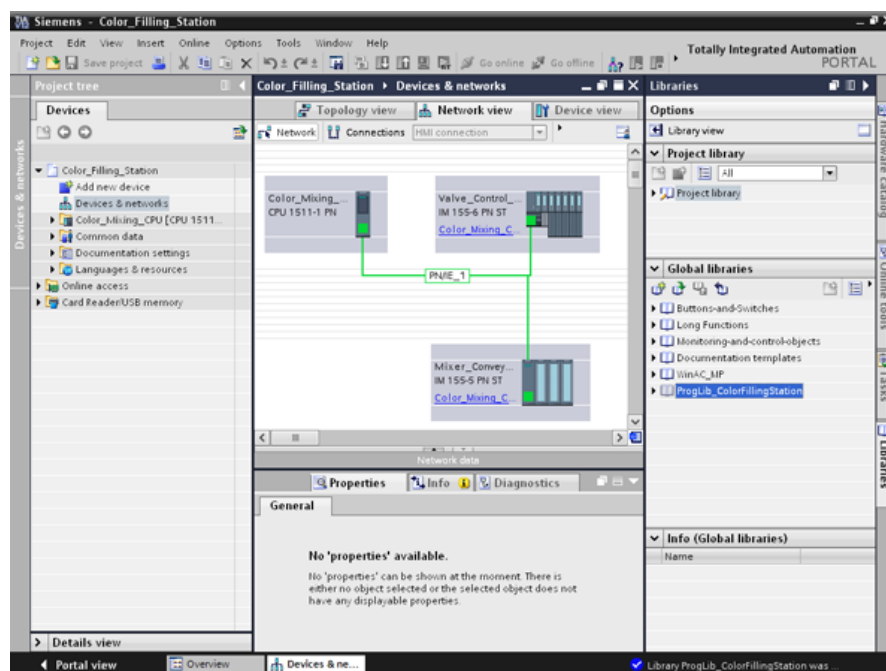
## Procedure

1. Click on the "Libraries" tab.
2. Click "Open global library".
3. Select the "ProgLib\_ColorFillingStation" file from the directory that contains the unzipped library folder and click "Open".



## Result

The "ProgLib\_ColorFillingStation" global library is open.

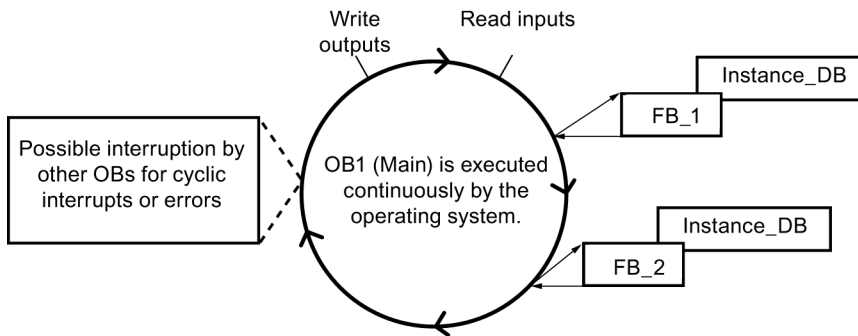


### 3.2.2 Deleting program block Main [OB1]

#### Introduction

In the following section, you will delete the automatically generated "Main [OB1]" program block from the project folder. A "Main [OB1]" program block is included in the program blocks of the example project.

Organization blocks (OBs) form the interface between the CPU operating system and the user program. These blocks are called by the operating system. At least one cycle OB must be available in an automation project.

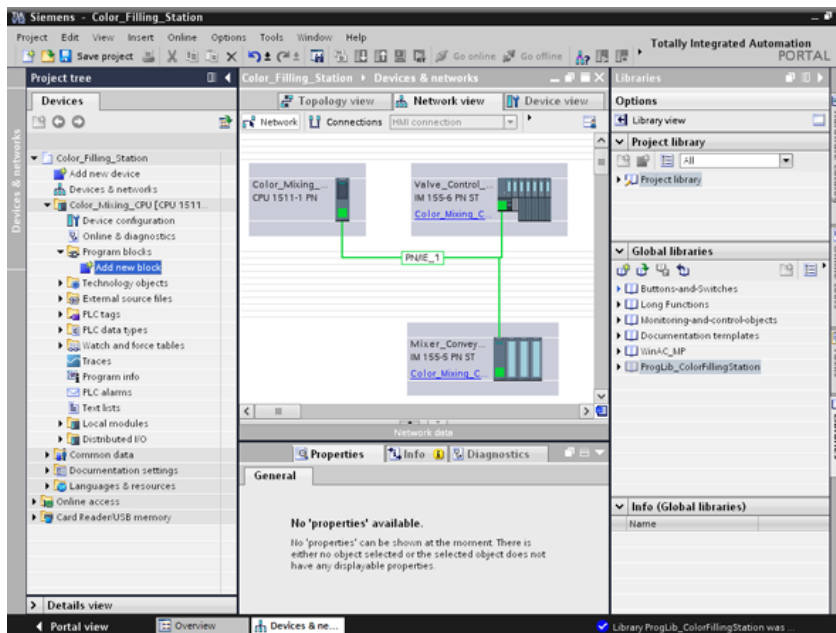


#### Procedure

1. Open the "Program blocks" folder in the project tree and then click the "Main [OB1]" program block.
2. Right-click to open the shortcut menu and then click "Delete".
3. Click "Yes" to confirm deletion of the block.

#### Result

The automatically generated "Main [OB1]" program block is deleted.



### 3.2.3 Copying program blocks

#### Introduction

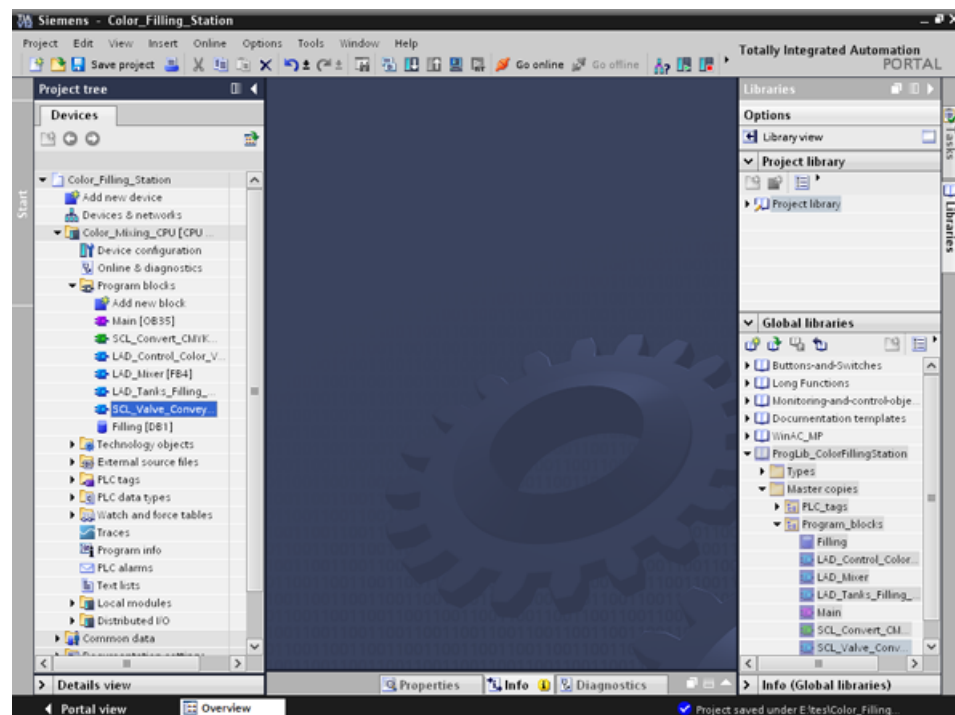
In the following section, you will insert the program blocks from the "ProgLib\_ColorFillingStation" global library into your project.

#### Procedure

1. Click on global library "ProgLib\_ColorFillingStation".
2. Click the "Master copies" folder and then on "Programm\_blocks".
3. Drag-and-drop the program block to be imported from the global library to the "Program blocks" folder.
4. Proceed as described in steps 2 and 3 for the other blocks.

#### Result

The program blocks are inserted in the project folder of the same name.



### 3.2.4 Cyclic interrupt OB

#### 3.2.4.1 Cyclic interrupt OB – Cycle time and phase

##### Cycle time and phase offset can be changed

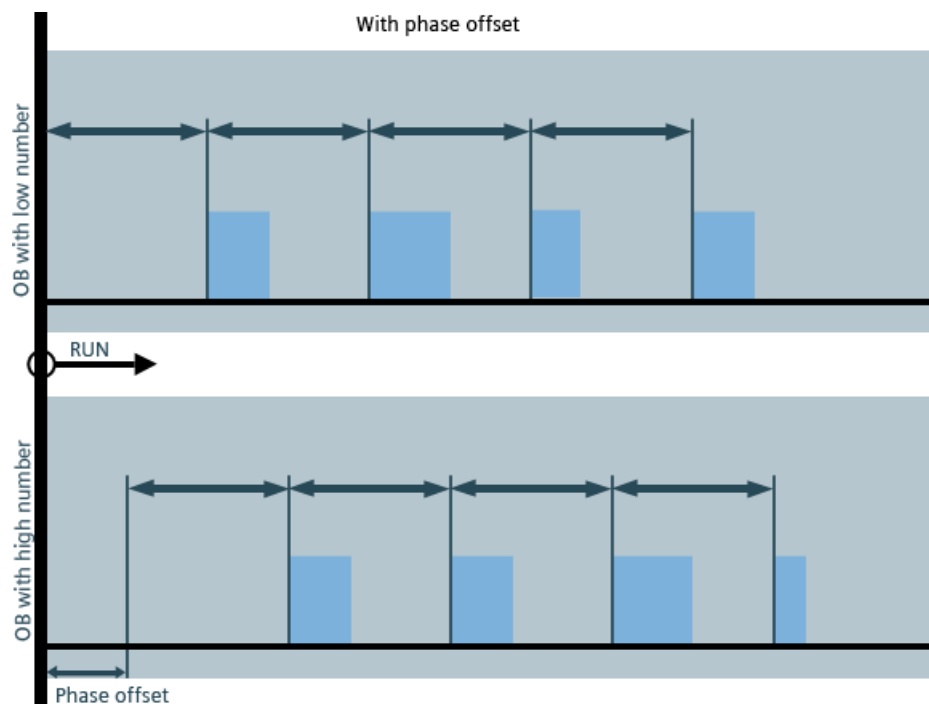
Main [OB35] is located below the program blocks inserted into the project. Main [OB35] is a cyclic interrupt organization block (cyclic interrupt OB). Cyclic interrupt OBs serve to start programs in periodic time intervals independently of the cyclic program execution. The start times of a cyclic interrupt OB are specified using the cycle time and the phase offset.

##### Cycle time

The cycle time determines the interval at which an OB is called. The cyclic interrupt OB has a cycle time of 100000  $\mu$ s by default.

##### Phase offset

The phase offset is used to increase the accuracy of the processing intervals of cyclic interrupt programs. If an OB has the same or a common multiple clock pulse of another OB, both can be operated at a precise interval by a phase offset.





### 3.2.4.2 Changing the cycle time

#### Introduction

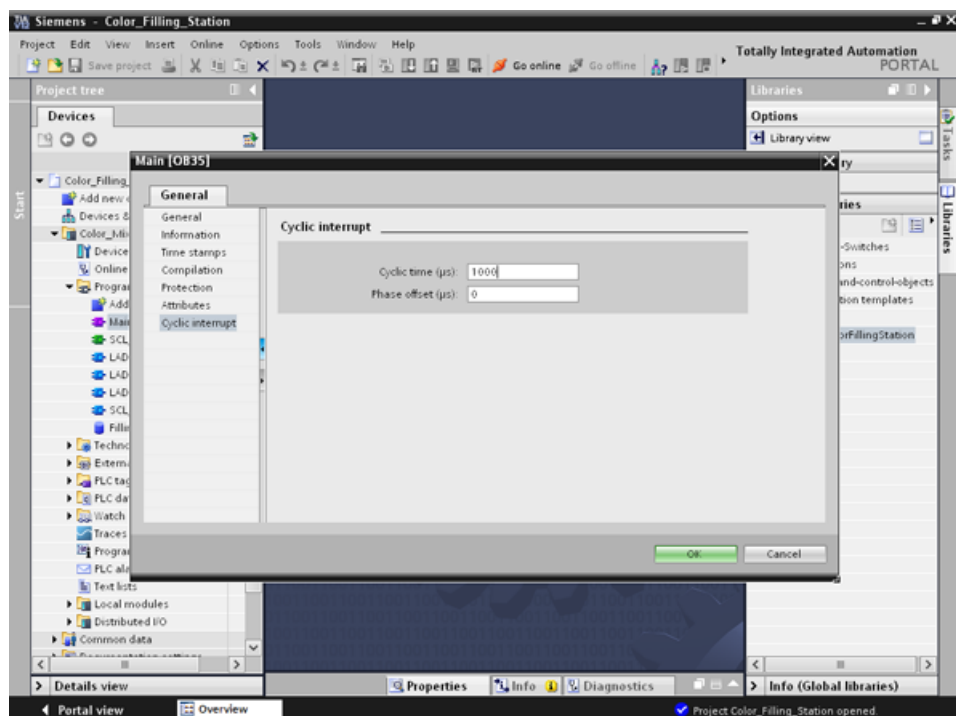
In the following section, you will change the cycle time for the "Main" program block.

#### Requirement

- The program block "Main" [OB35] is contained in the library
- The FB/FC calls exist

#### Procedure

1. Open the properties of the "Main" program block.
2. Select the "Cyclic interrupt" option under "General".
3. Enter the new value for the "Cycle time" and click "OK".



#### Result

The cycle time is changed.

### 3.2.5 Copying tag tables

#### Introduction

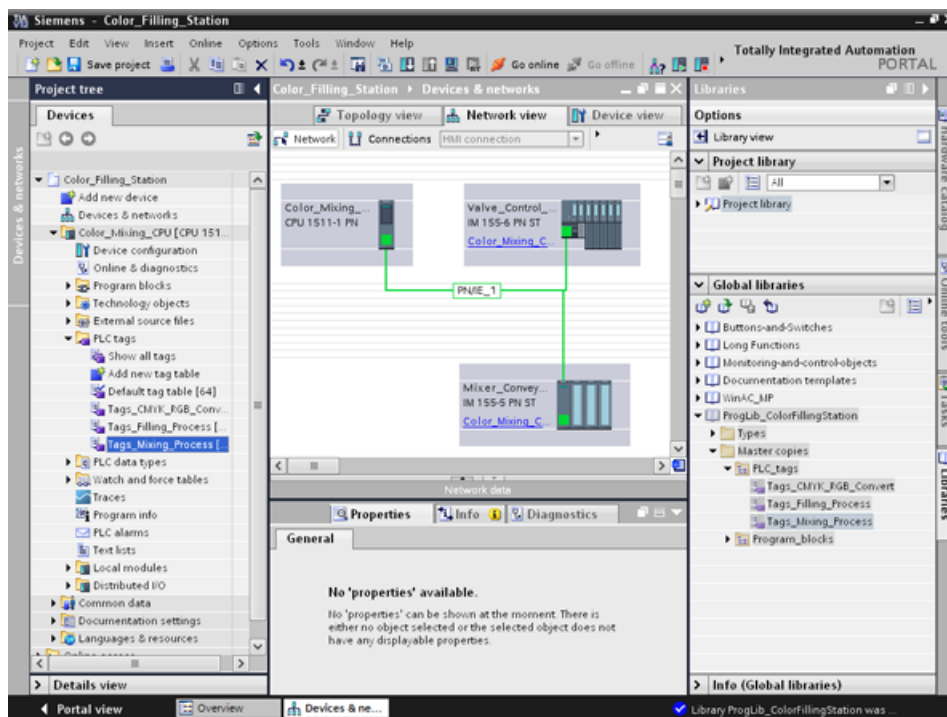
In the following section, you will insert the tag tables from the "ProgLib\_ColorFillingStation" global library into your project.

#### Procedure

1. Open the "PLC tags" folder in the project navigation.
2. Open the "PLC\_tags" folder.
3. Drag-and-drop the tag table to be imported from the global library to the "PLC tags" folder.
4. Proceed as described in step 3 for the other tag tables.

#### Result

The tag tables are inserted in the project folder of the same name.



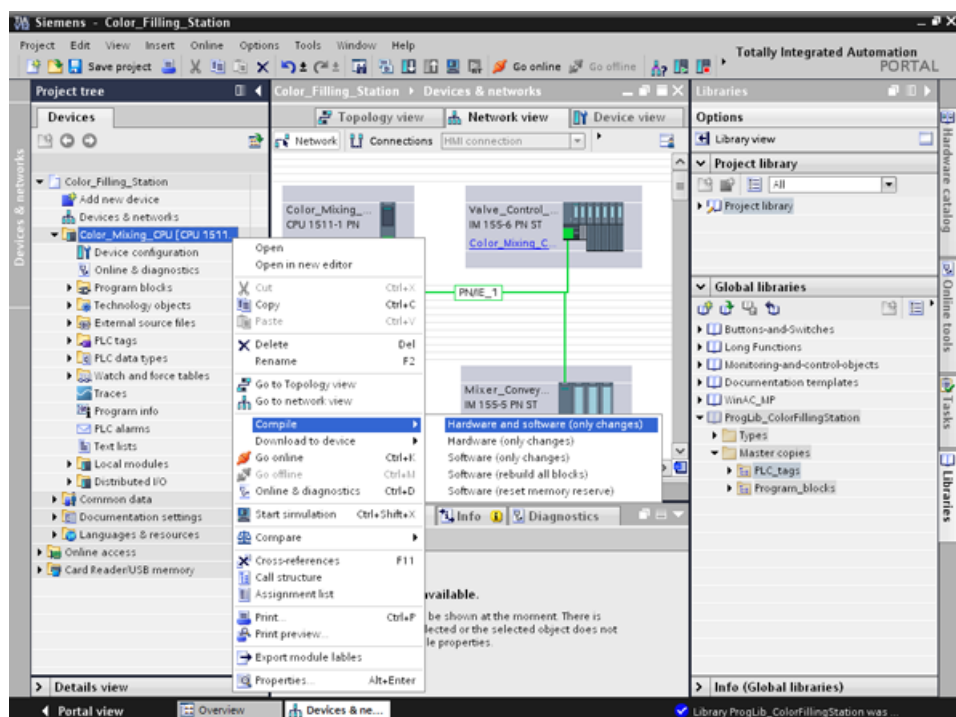
## 3.2.6 Compiling a project

### Introduction

In the next section, you will compile the "Color\_Filling\_Station" project.

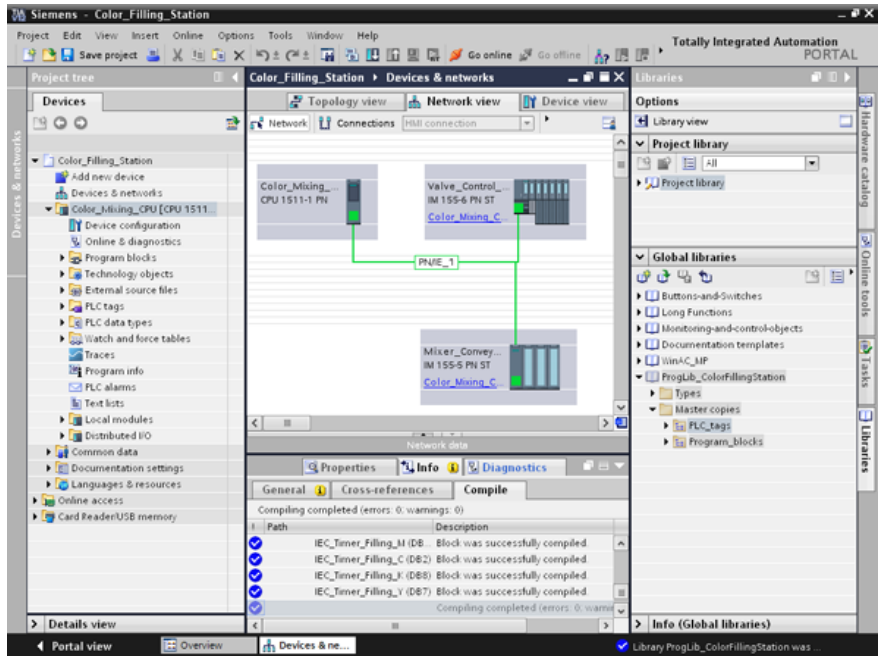
### Procedure

1. Select the "Color\_Mixing\_CPU" CPU in the project tree.
2. Right-click to open the shortcut menu and then select "Compile" > "Hardware and software (only changes)".



### Result

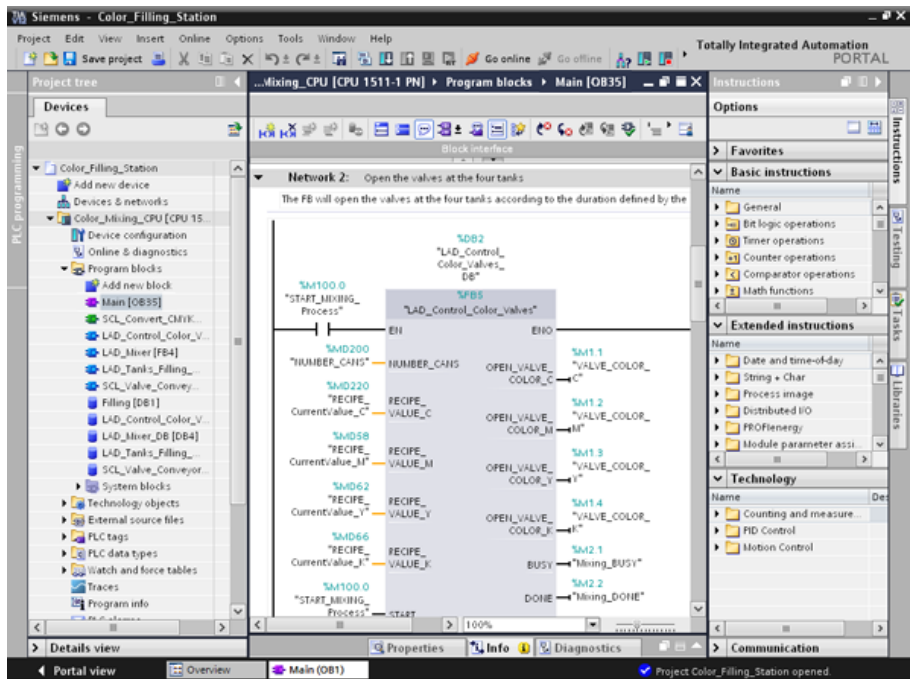
The project is compiled and ready for downloading.



### Note

"Main" program block is updated

Open the "Main" program block after compilation. All instance data blocks have been created and the data blocks are updated.



## 3.2.7 Load project into the CPU

### Introduction

In the next section, you will download the "Color\_Filling\_Station" project to the CPU.

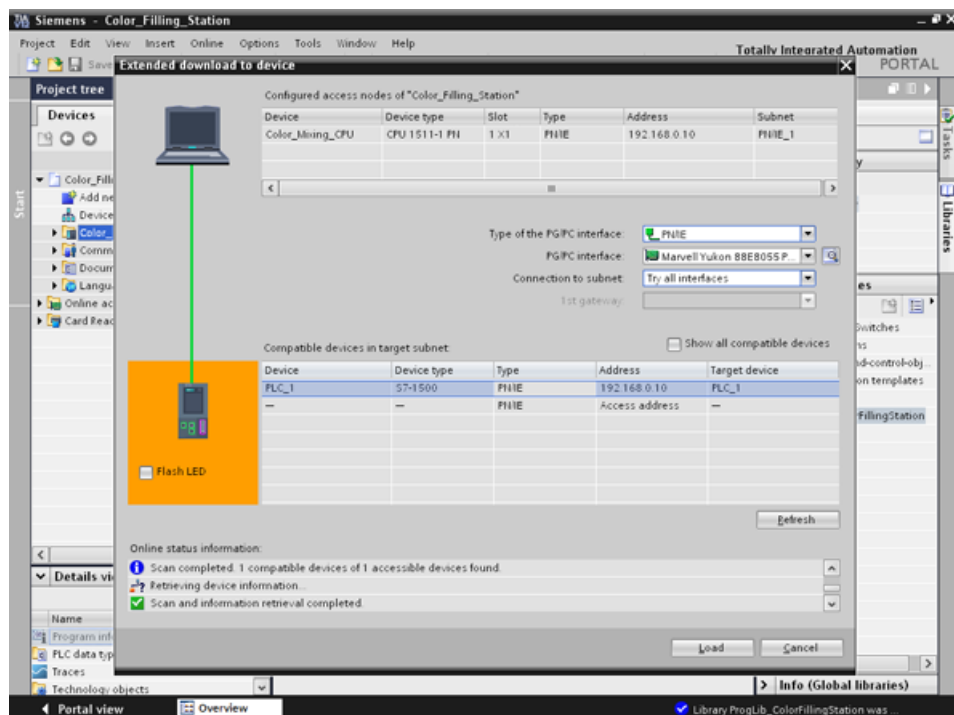
### Note

#### Displaying all compatible devices

If the desired CPU is not displayed after you have made the settings in the "Extended download to device" dialog, click the option "Show all compatible devices".

### Procedure

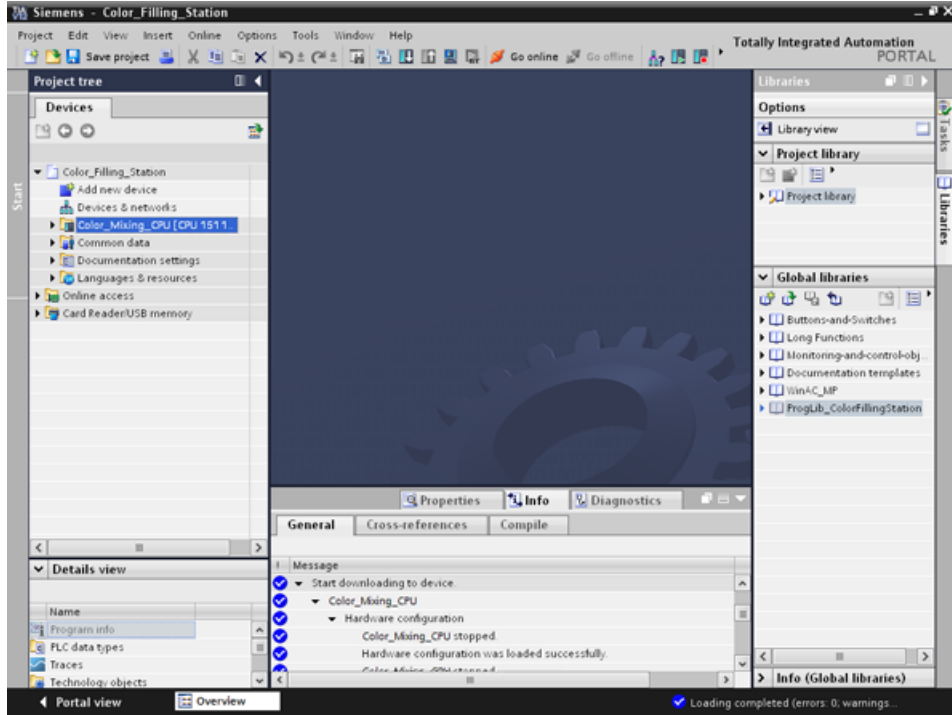
1. Open the CPU shortcut menu and select "Download to device" > "Hardware and software (only changes)".
2. From the drop-down lists, select the PG/PC interface type, the interface and the connection with the subnet.
3. Select the CPU from the compatible devices in the subnet and click "Load".



4. Confirm the two "Assign IP address" dialogs with "Yes" and "OK".
5. In the "Load preview" dialog, select the alternative entry for all entries set to "No action" in the drop-down list and confirm open options.
6. Click "Load".
7. Confirm the "Start all" option and click "Finish".

## Result

The project is downloaded to the CPU.



## 3.2.8 Optimized block access

### 3.2.8.1 Introduction

#### Operating principle

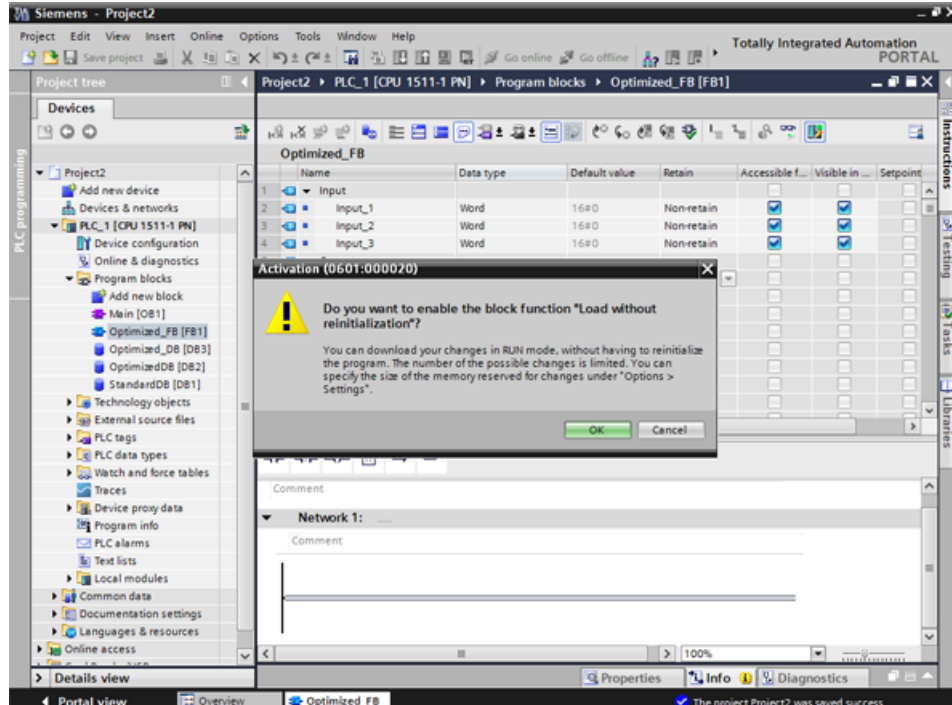
The "optimized data blocks" of the CPUs of the S7-1500 series are optimized for performance and are only programmed symbolically. By using the optimized data blocks, you make your program more efficient, because the declared tags are given symbolic names and no longer a fixed address.

You can create data blocks with any structure without paying attention to the physical arrangement of the individual tags. Quick access to the optimized data is always available because the data storage is optimized and managed by the system.

Changing data types increases the risk of error in the standard block. In the optimized block, changes lead to a reorganization of the data storage. Addressing remains unique.

To enable the subsequent editing of user programs that are already running in a CPU, the S7-1500 CPUs support the option of extending the interfaces of function or data blocks during runtime. You can download the modified blocks without setting the CPU to STOP and without affecting the actual values of tags already loaded.

In addition: You can define in the data block itself, which the values in the CPU are read-only for an HMI device ("Visible in HMI") or which can be written ("Accessible from HMI").



### 3.2.8.2 Expanding and reloading the optimized "Filling" data block

#### Introduction

In the following section, you will supplement the "Filling" data block with the date and time of the last filling and reload the data block. To do this, create a block for recording the date and time and enable the function "Download without reinitialization".

**Note:** The "Download without reinitialization" function protects the actual parameters of the data block from being overwritten during download to the CPU.

**Advantages of symbolic addressing:** The use of universally applied and meaningful symbols in the entire project makes the program code easier to read and understand. This gives you the following advantages:

- You do not have to write detailed comments.
- Data access is faster.
- No errors occur when accessing data.
- You no longer have to work with absolute addresses.
- The assignment of the symbol to the memory address is monitored by STEP 7, which means that all points of use are automatically updated when the name or the address of a tag changes.

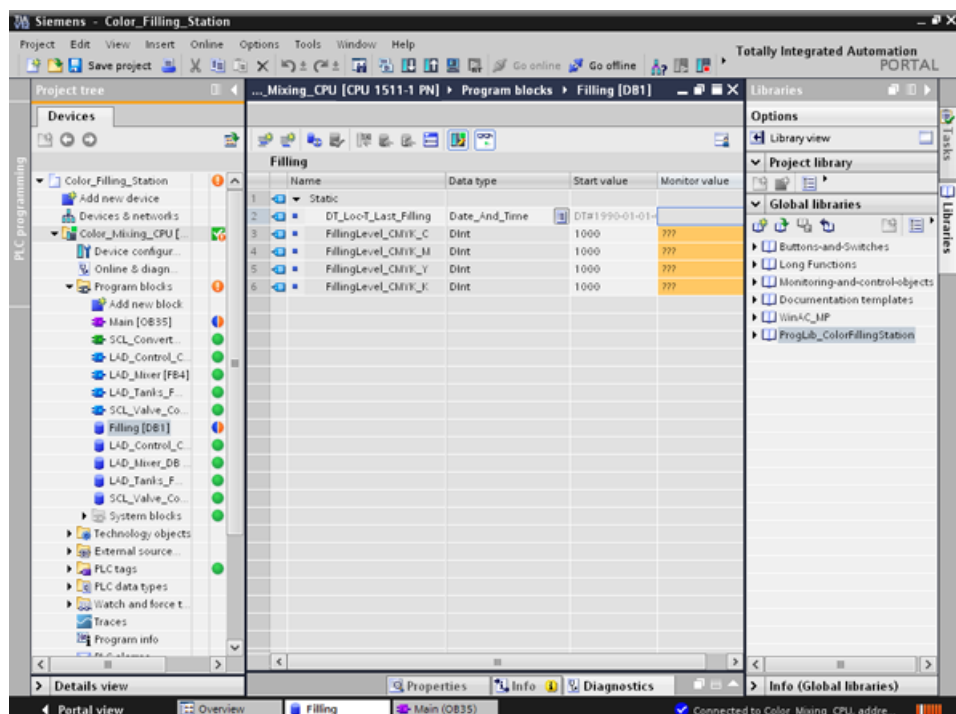
#### Requirement

- The library has been loaded
- The project has been compiled and loaded into the CPU

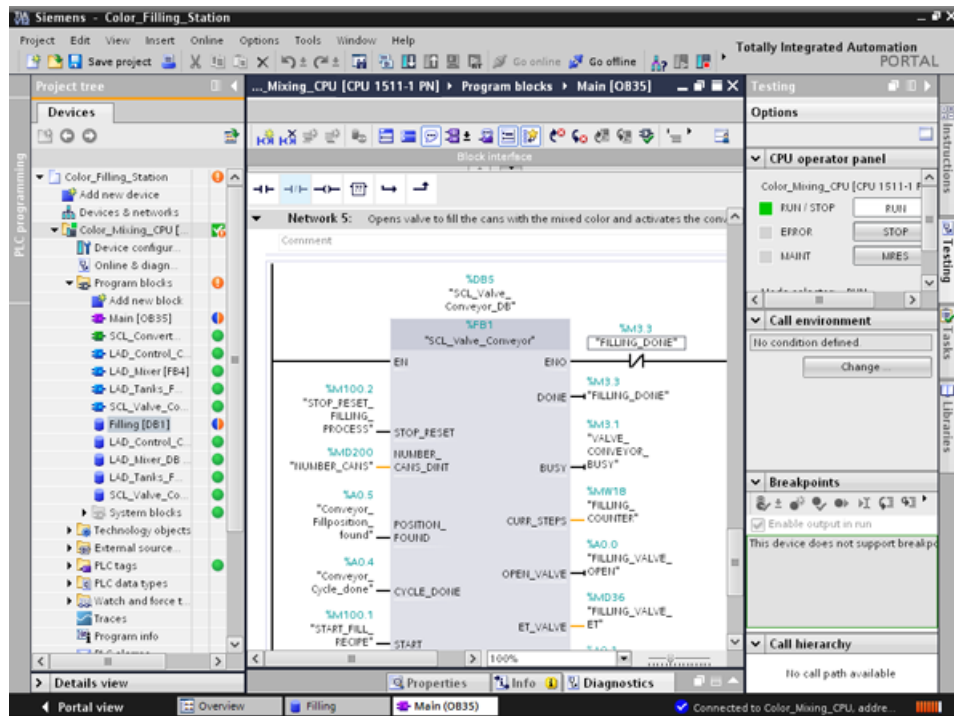


## Procedure

1. Open the "Filling" data block and the "Main" program block.
2. Enable the "Monitoring on/off" function for the "Main" program block.
3. In the "Main" program block, open the shortcut menu of the "'FILLING' FillingLevel\_CMYK\_C" I/O in the 3 network with a right-click and select "Modify > Modify operand".
4. Enter a new value and click "OK".
5. Enable the "Download without reinitialization" function and the "Monitor all" function in the "Filling" data block.
6. Create a new parameter named "DT\_Loc-T\_Last\_Filling" and select "Date\_And\_Time" as the data type.

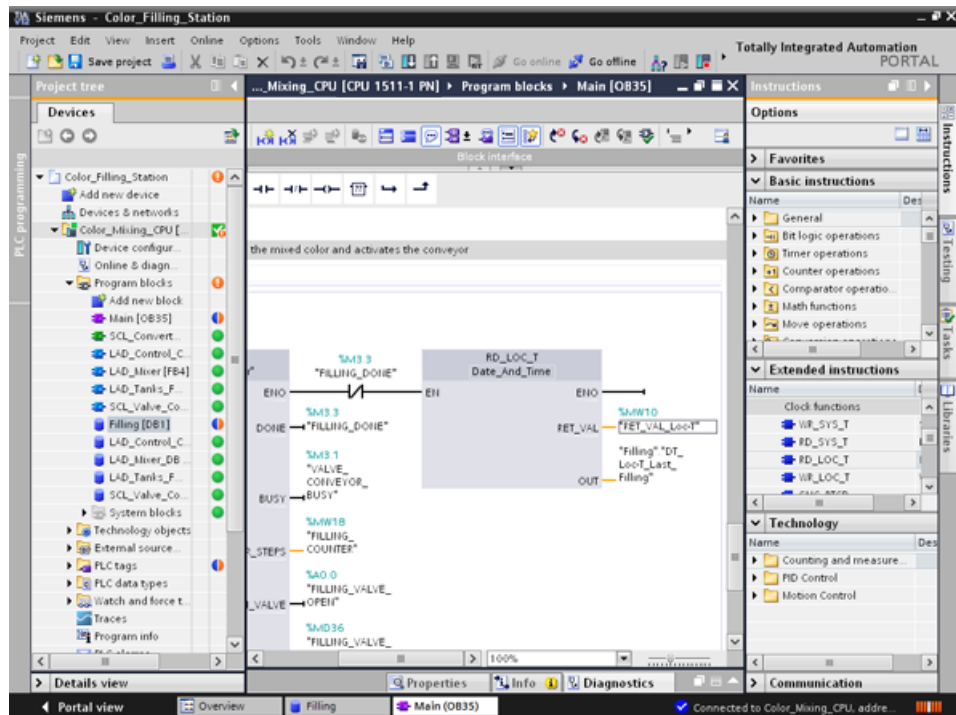


7. Insert a normally closed contact into the "Main" program block in the 5 network, and interconnect it with the "FILLING\_DONE" parameter.



8. Open the "Date & time" folder from the "Instructions" tab and insert the "RD\_Loc\_T" block in the "Main" program block.

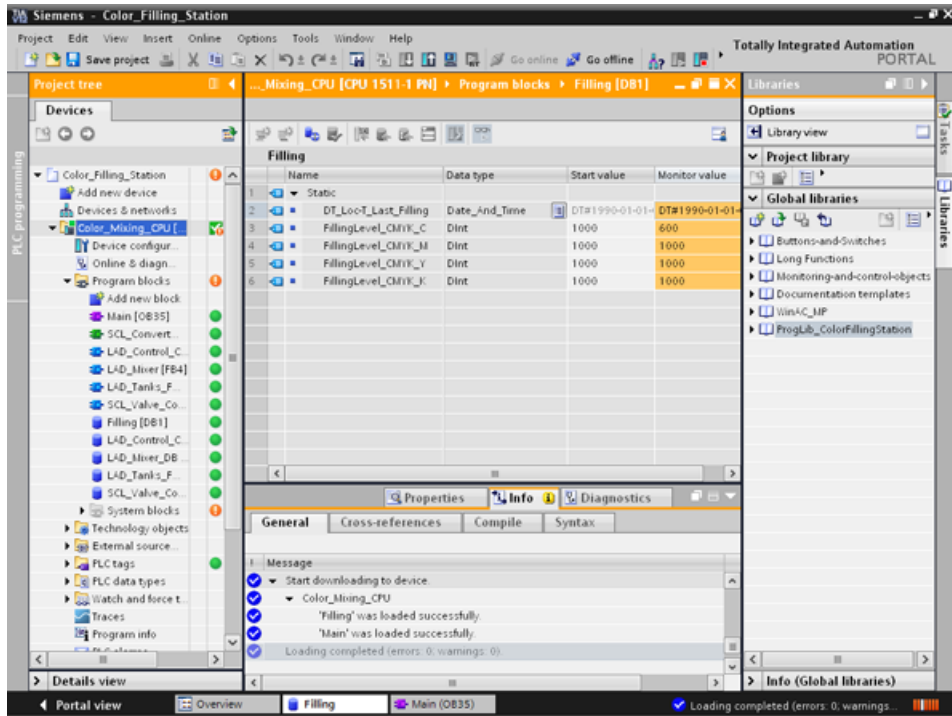
- Interconnect the "OUT" output with the "DT\_Loc-T\_Last\_Filling" parameter and the "RED\_VAL" output with the newly created "RED\_VAL\_Loc-T" parameter. Use the "LAD\_Tanks\_Filling\_Process" data block as the storage location for the "RED\_VAL\_Loc-T" parameter.



- Compile and download the project.

### Result

The date and time of the last filling are reloaded. The actual parameters of the "Filling" data block are not overwritten.



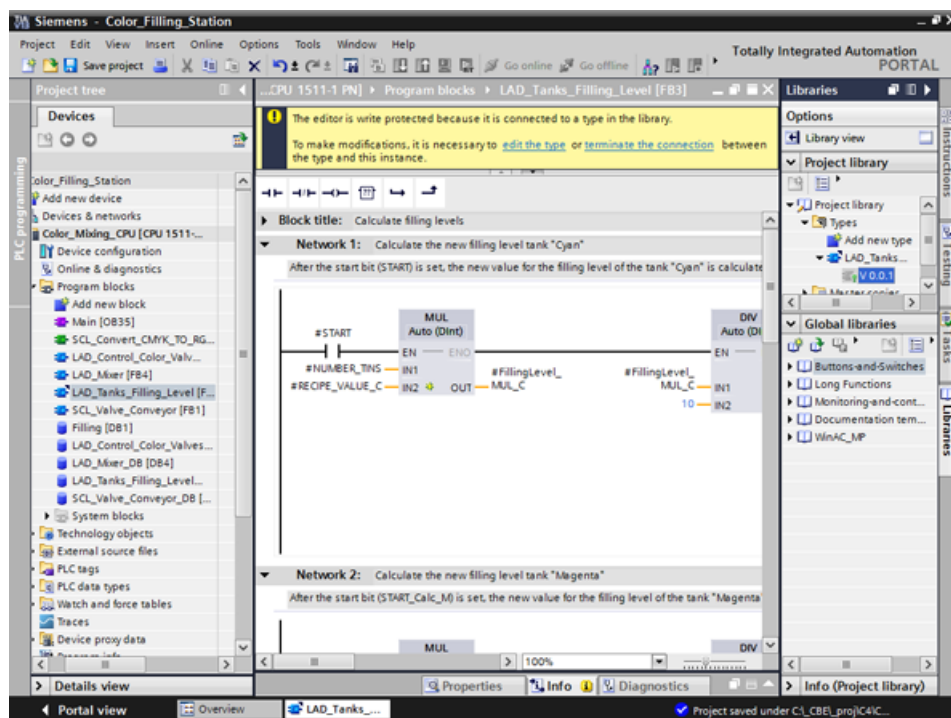
## 3.2.9 Versioning a block

### Introduction

The use of block types ensures a high degree of standardization in your projects. You can easily integrate function extensions to the block type into existing projects. Change tracking is ensured by versioning. In this example, you create a "LAD\_Tanks\_Filling" block as a type in the project library. As a function extension, replace the three instructions for the level calculation with CalculateBox, which performs all arithmetic functions. This optimization means that fewer temporary tags are required and that the switch between blocks with various programming languages is no longer necessary.

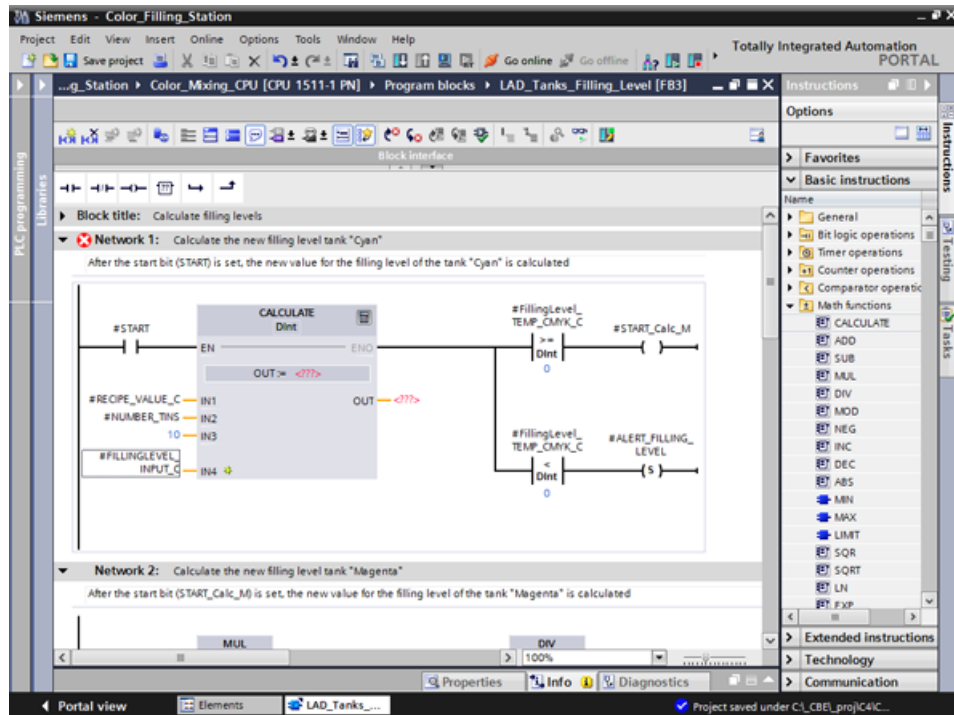
### Procedure

1. Compile the "LAD\_Tanks\_Filling" block and then insert it in the project library under "Types".
2. Create a new block version with "Edit type".

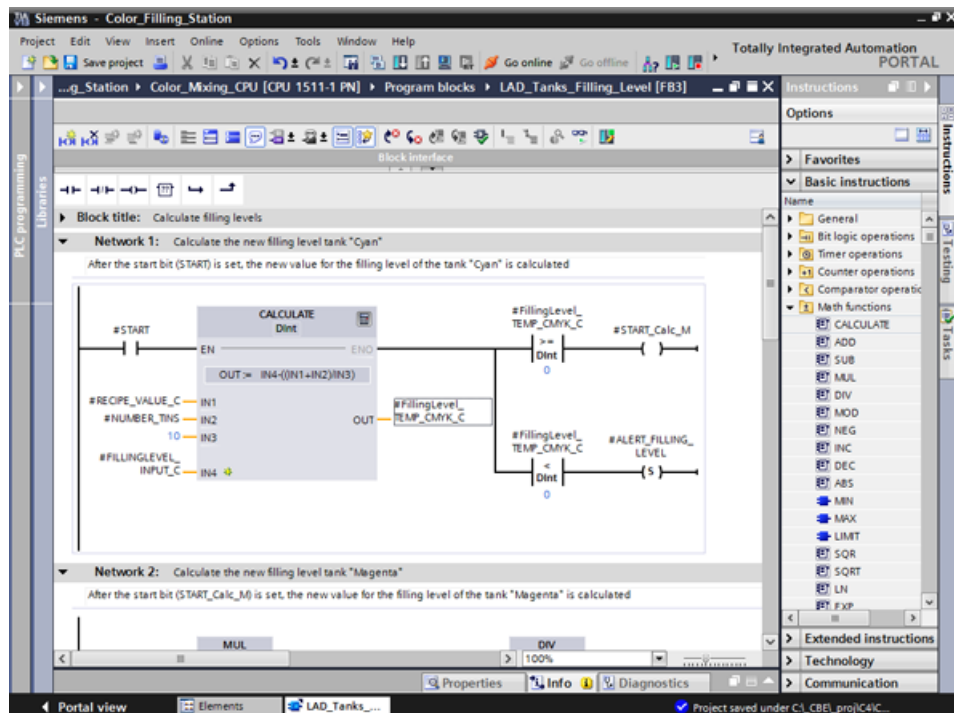


3. Insert the CALCULATE instruction from the "Basic instructions > Mathematical functions" library.
4. Delete the MUL, DIV and SUB instructions from the block.

5. Insert two inputs into the CALCULATE instruction and interconnect the inputs.



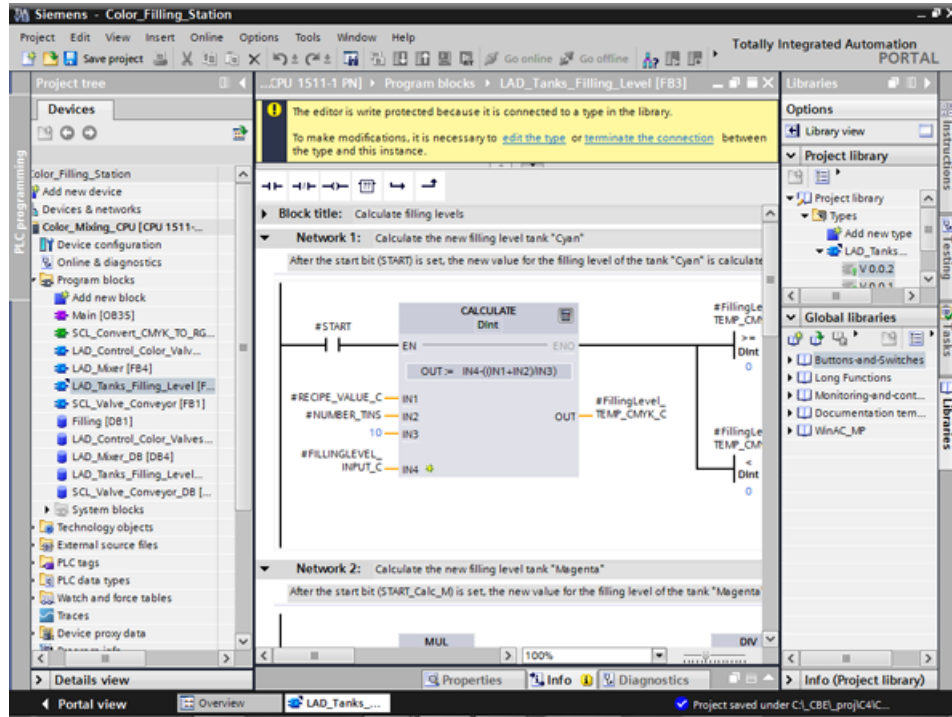
6. Define the calculation formula and then interconnect the output.



7. Release the block version.

## Result

The revised version of the block type is saved in the library with a new version number.



### 3.2.10 Setting retentivity

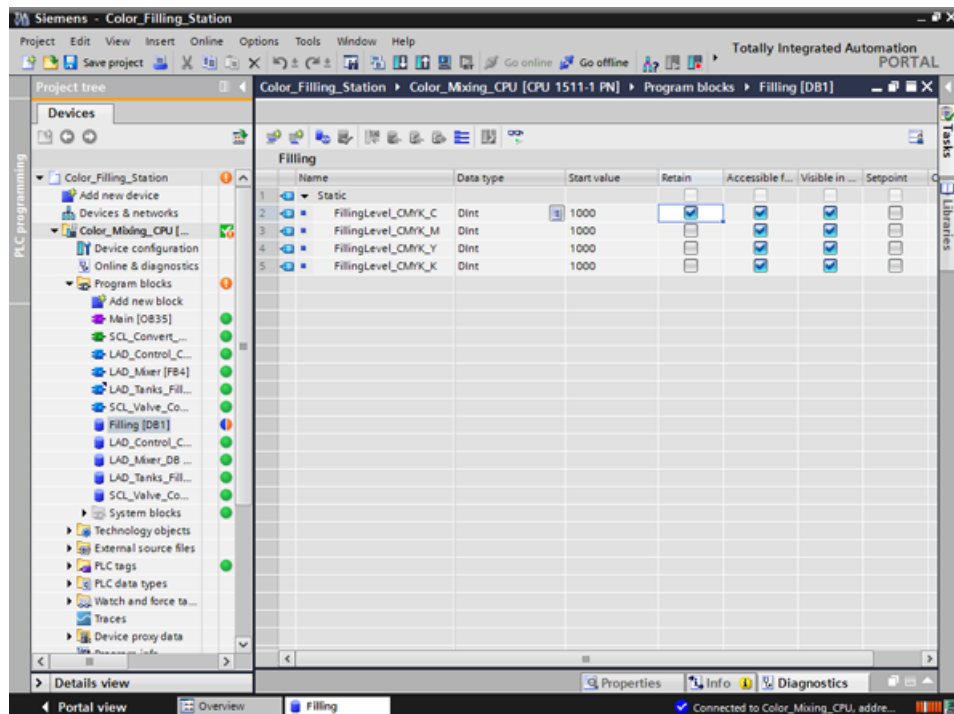
#### Introduction

All tags are initialized with their configured start values during CPU startup, for example, after a power failure. The most recent values the tags had immediately before the interruption are overwritten with the initial values. To prevent this, define the tag as retentive. Retentive tags retain their values even after a restart.

In this example, the levels of paint storage tanks are backed up in the retentive memory area of the CPU.

#### Procedure

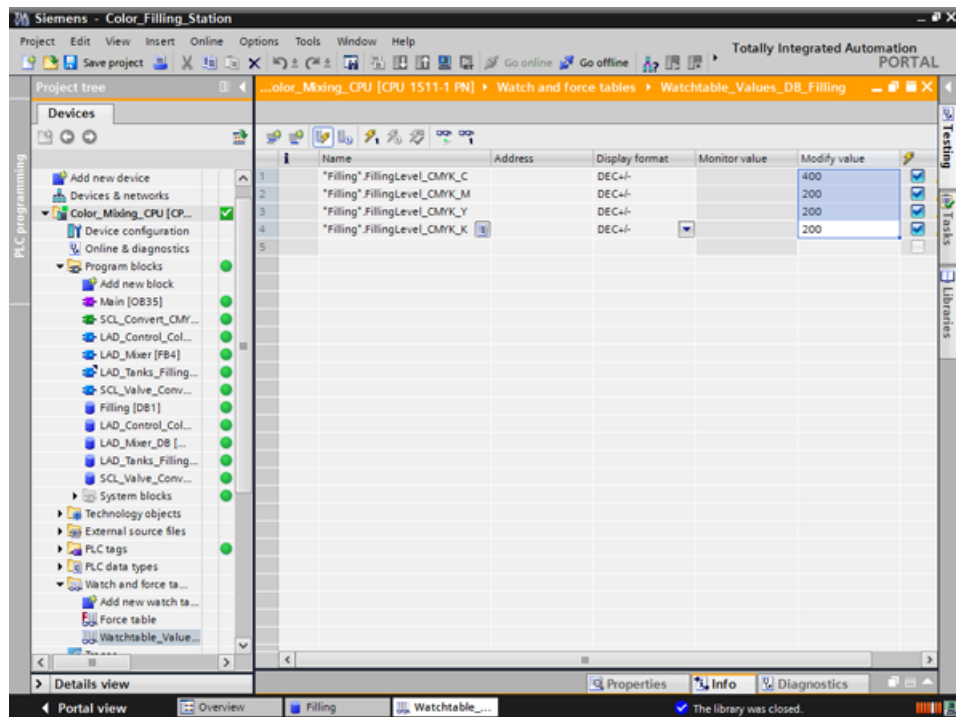
1. Connect to the CPU online.
2. Enable the retentivity for the "Cyan" entry in the "Filling" data block.



3. Load the change to the CPU.
4. Drag the "Watchtable" object from the library into the project. This object contains the fill level tags included a control value.



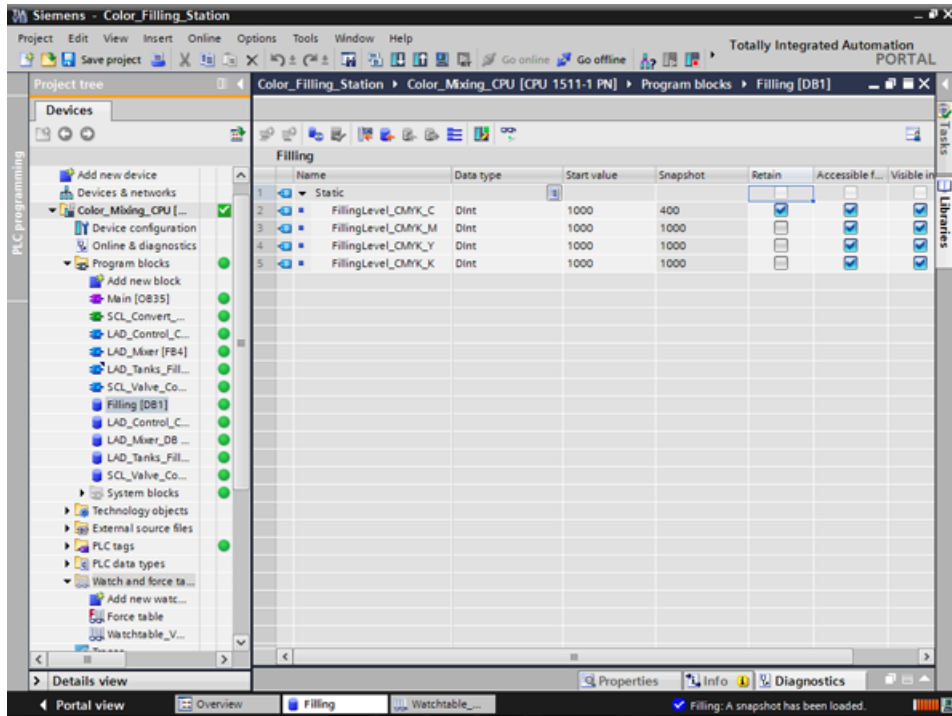
5. Transfer the control values to the CPU with "Modify now".



6. Close the online connection to the CPU. To simulate a power failure, disconnect the power supply to the CPU.
7. Reconnect the power supply and go online to the CPU. Enable "Monitor all" for the "Filling" DB.

### Result

The fill level for "Cyan" is read from the retentive memory area. All other fill levels are re-initialized with their start value.



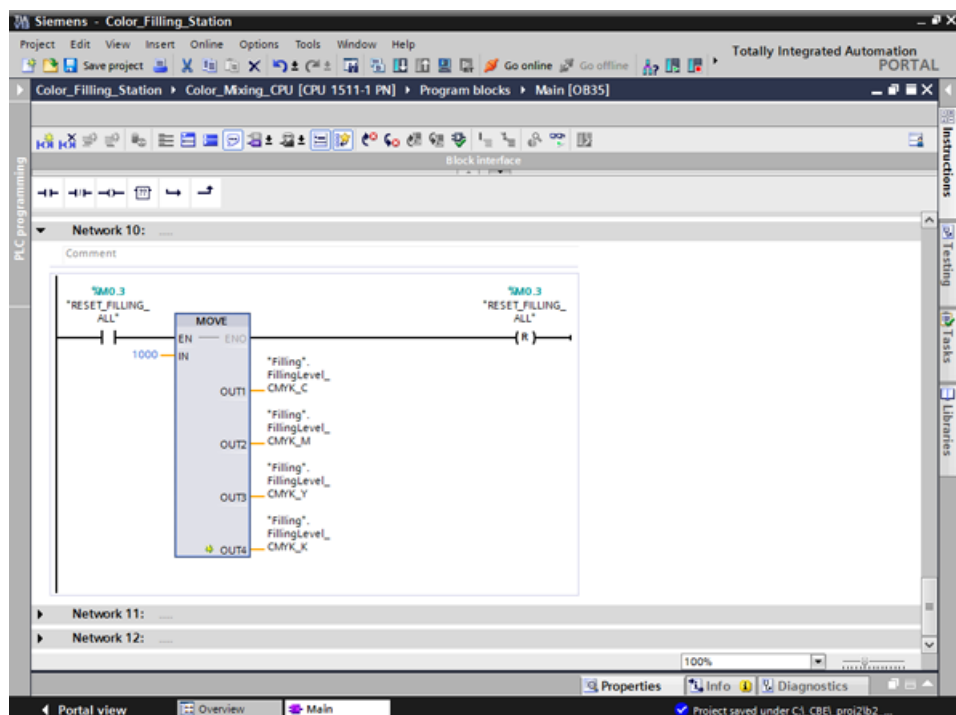
### 3.2.11 Activating the EN/ENO mechanism

#### Introduction

The EN/ENO mechanism in various instructions enables you to detect runtime errors and avoid a program crash. Newly inserted ENO instructions are disabled by default. You can then activate the ENO enable output. You can use this in a new network that has the fill level of all paint storage tanks reset to the start value (1000) at the same time.

#### Procedure

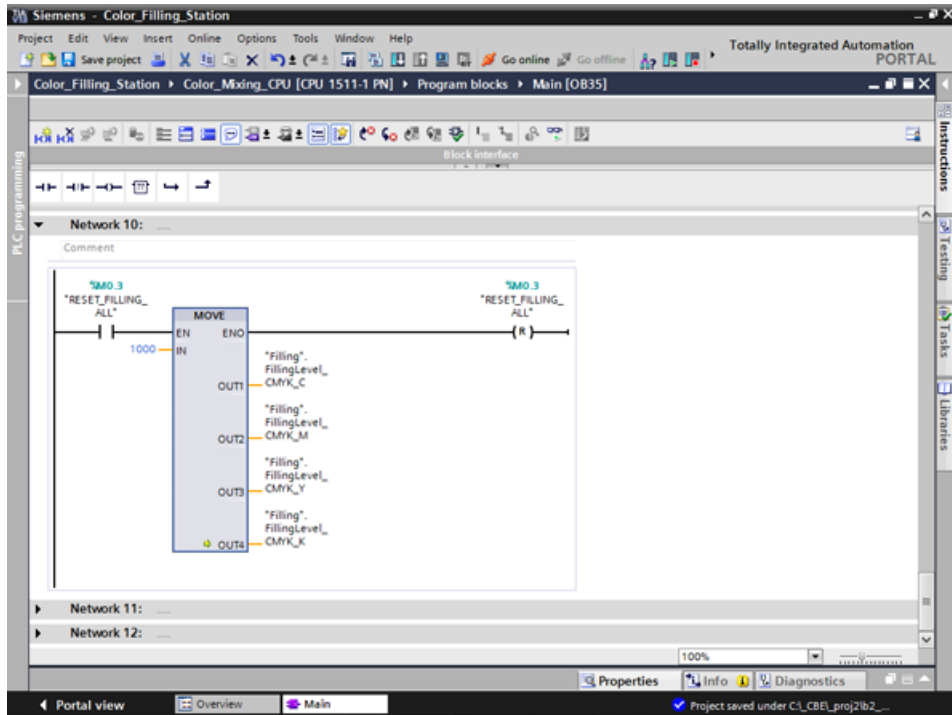
1. Open the Main[OB35] program block and insert the MOVE instruction into network 10.
2. Expand the instruction to a total of four outputs.
3. Insert a normally open contact before the MOVE instruction.
4. Insert a reset coil after the MOVE instruction.
5. Interconnect the inputs and outputs of the MOVE instruction.



6. Generate the instruction with the ENO shortcut menu.

### Result

The EN/ENO mechanism is interconnected for this block. If there are no errors during execution, the ENO enable output has the signal state "1". If there are errors during execution, the ENO enable output has the signal state "0".



### 3.2.12 Using the comment function

#### Introduction

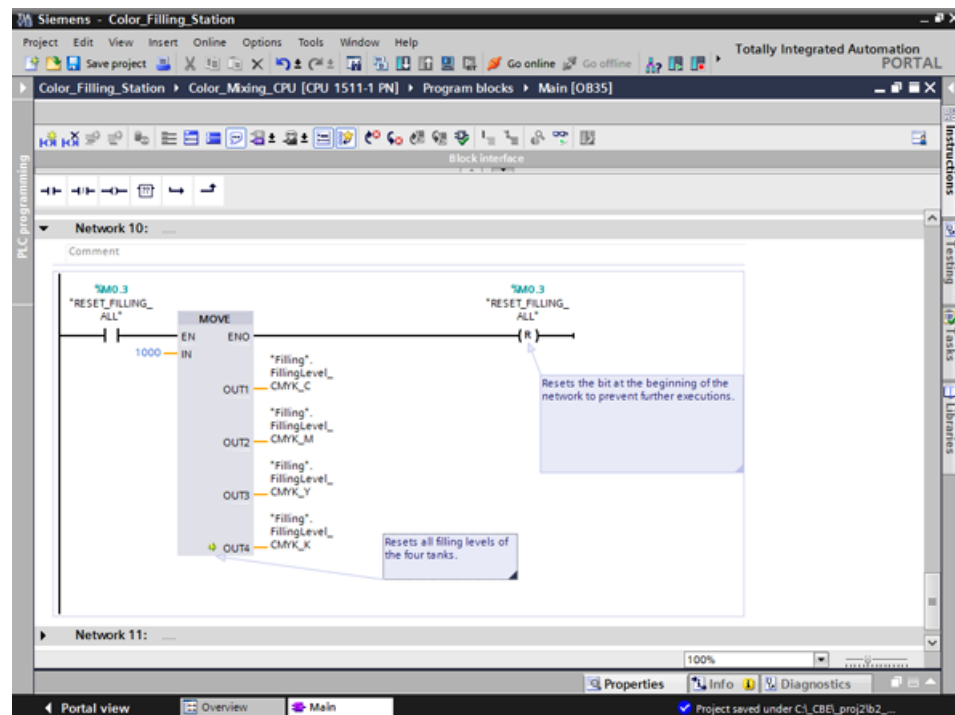
The MOVE and Reset instructions should be expanded with detailed commentary.

#### Procedure

1. Insert a comment using the shortcut menu.
2. Enter the comment text.

#### Result

The comments for the instruction and the coil are entered.

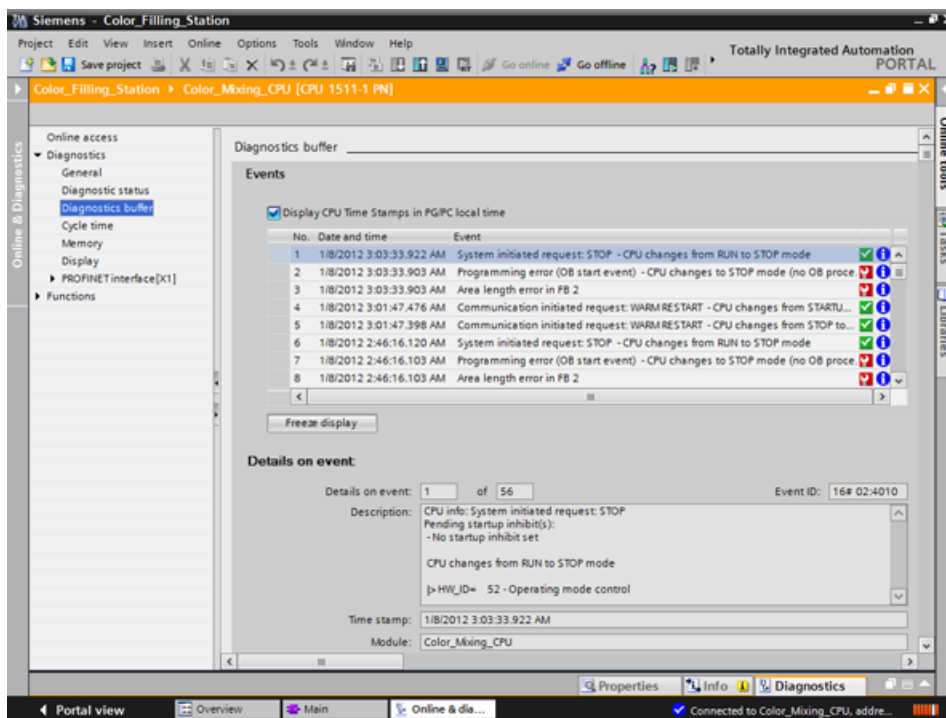


### 3.2.13 Local error handling

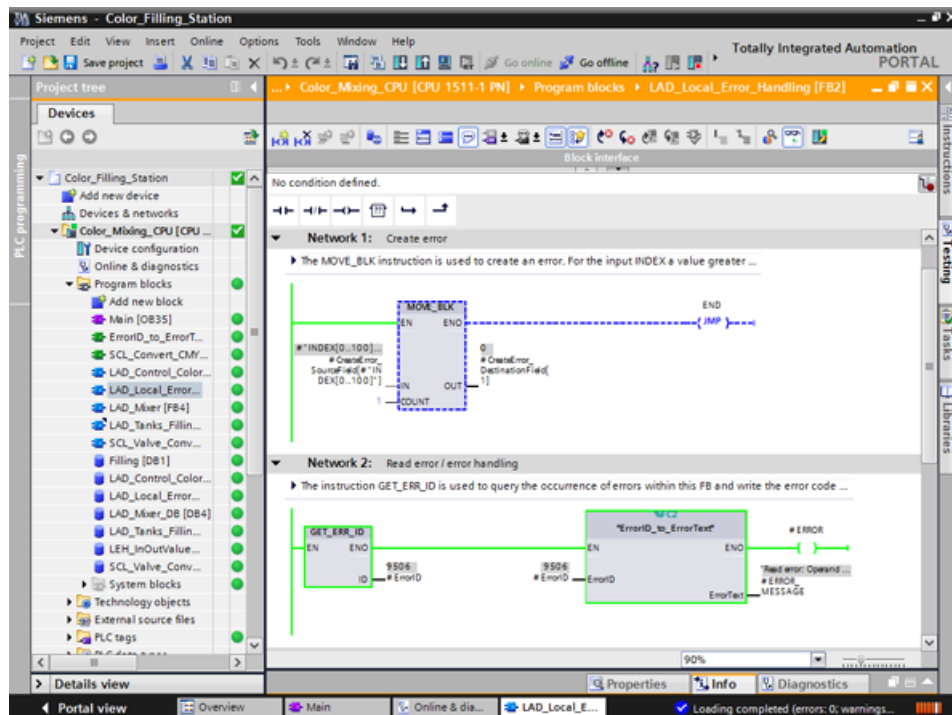
#### 3.2.13.1 Handle errors within block

#### Procedure

Unlike the CPUs of the S7-300/400, CPUs of the S7-1500 go to STOP with errors much less often. If an error occurs, it is entered in the diagnostics buffer of the CPU. You avoid the CPU STOP by using local error handling at each block. You should preferably enable local error handling during development of the user program.



You can precisely evaluate the information and, for example, program the error handling in the block with STL/FBD/LAD and SCL programs. The block generates an error ID that is evaluated by the "GET\_ERROR\_ID" instruction. You can call the "GET\_ERROR\_ID" instruction in both the MAIN block and in the function blocks. The CPU remains in RUN mode.



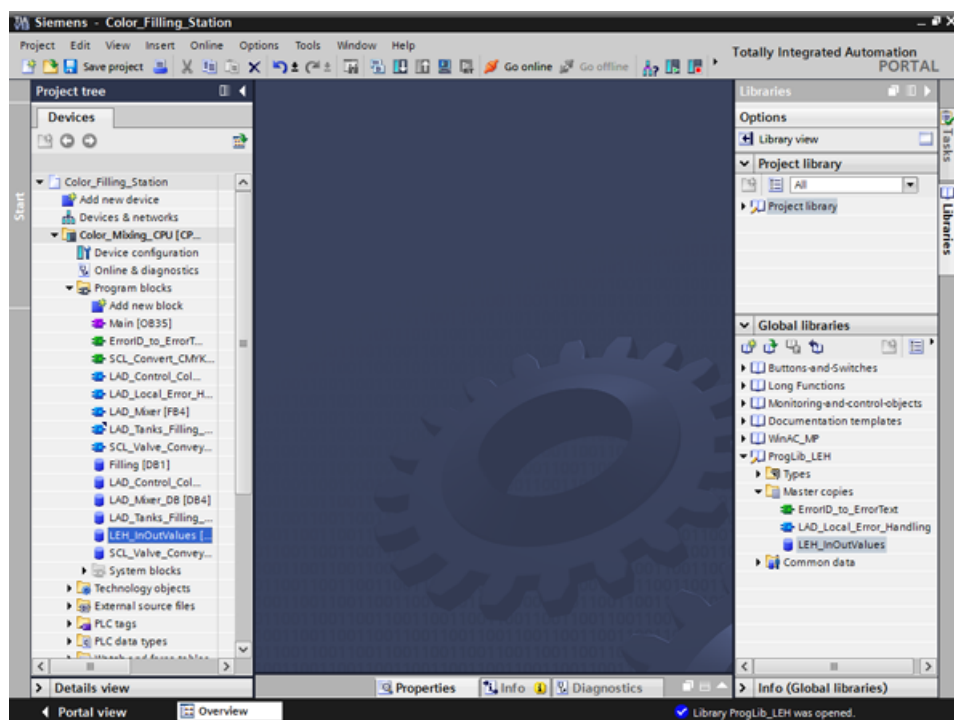
### 3.2.13.2 Loading blocks for local error handling

#### Introduction

To illustrate the local error handling, load the blocks of the "ProgLib\_LEH" library in the project. The blocks are used only to demonstrate the local error handling and are otherwise not used in the project.

#### Procedure

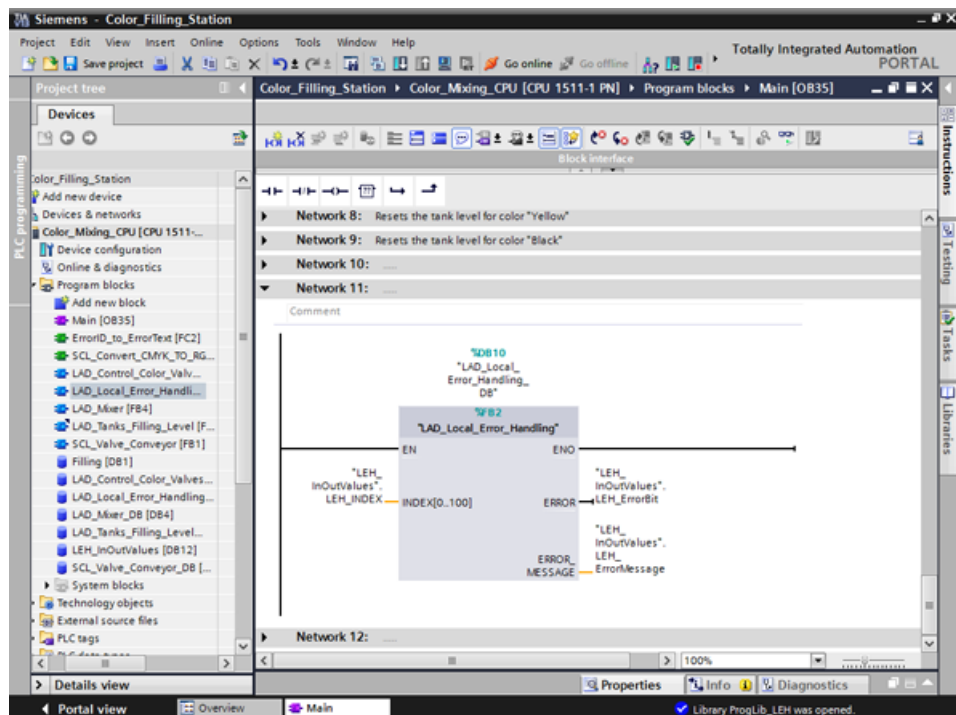
1. Open the global library, "ProgLib\_LEH".
2. Copy the blocks from the master copies into the project.



3. Call the "LAD\_Local\_Error\_Handling" function block in an empty network of the "Main" block.



- Interconnect the parameters of the "LAD\_Local\_Error\_Handling" function block with tags of the "LEH\_InOutValues" data block.



- Connect to the CPU online.
- Compile and load the changes to the CPU.

## Result

Use the "LEH\_INDEX" tag at the "INDEX[0..100]" input parameter to trigger a programming error in the following. For example, if you set the input parameter to "101", an error at the output parameters is reported.

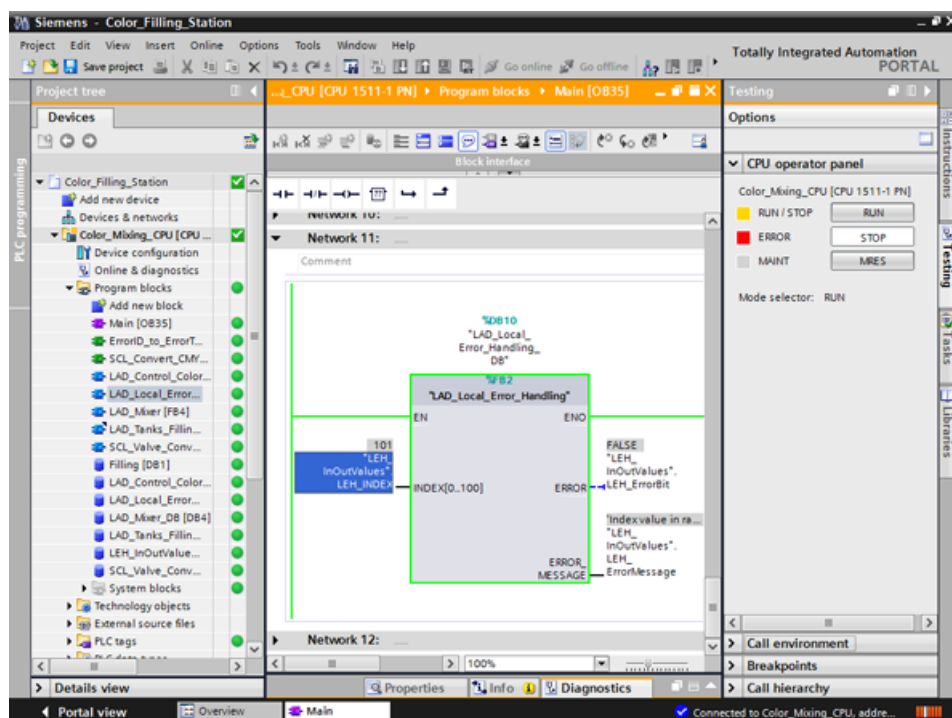
### 3.2.13.3 Generating errors without local error handling

#### Introduction

Perform the following steps to trigger a programming error without using the local error handling or creating a corresponding OB.

#### Procedure

1. Activate the "Monitor" function.
2. Set the value of the "LEH\_INDEX" tag to an invalid value, for example, "101". In the Testing dialog, the ERROR LED flashes briefly and the CPU goes from RUN to STOP.



3. Switch to the diagnostics buffer. The error and the error response is displayed in the diagnostics buffer.
4. Set the CPU back to RUN.

#### Result

The transition from STOP in RUN resets the "LEH\_INDEX" tag to the start value "0". This automatically solves the problem.

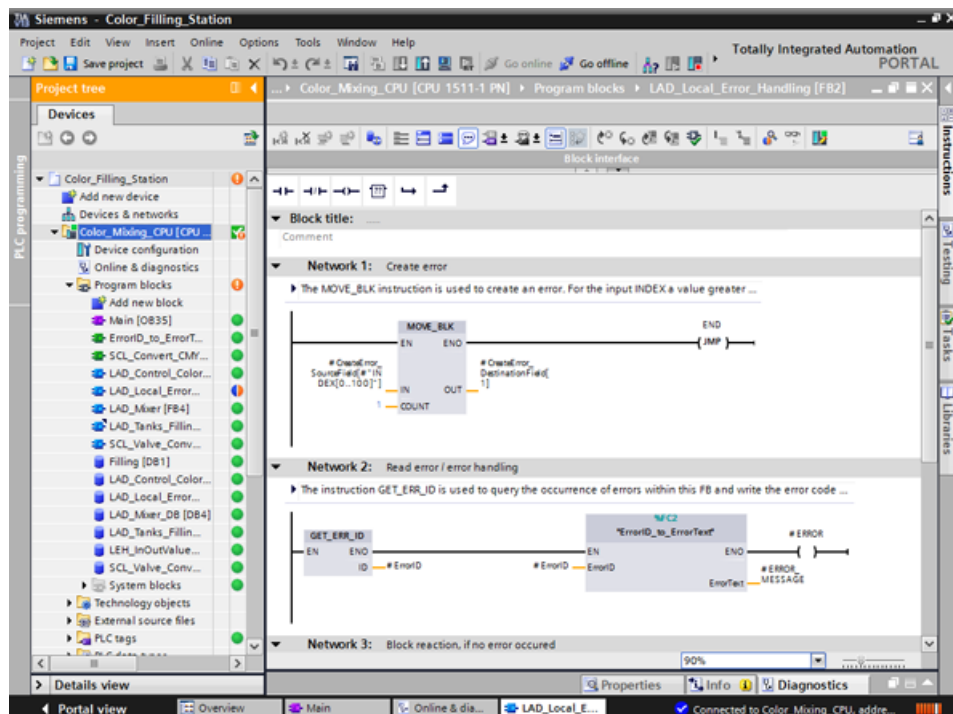
### 3.2.13.4 Generating errors with local error handling

#### Introduction

Perform the following steps to use "GET\_ERR\_ID" instruction and its ENO bit for the local error handling to respond to the error with an error message. This means the CPU remains in RUN mode.

#### Procedure

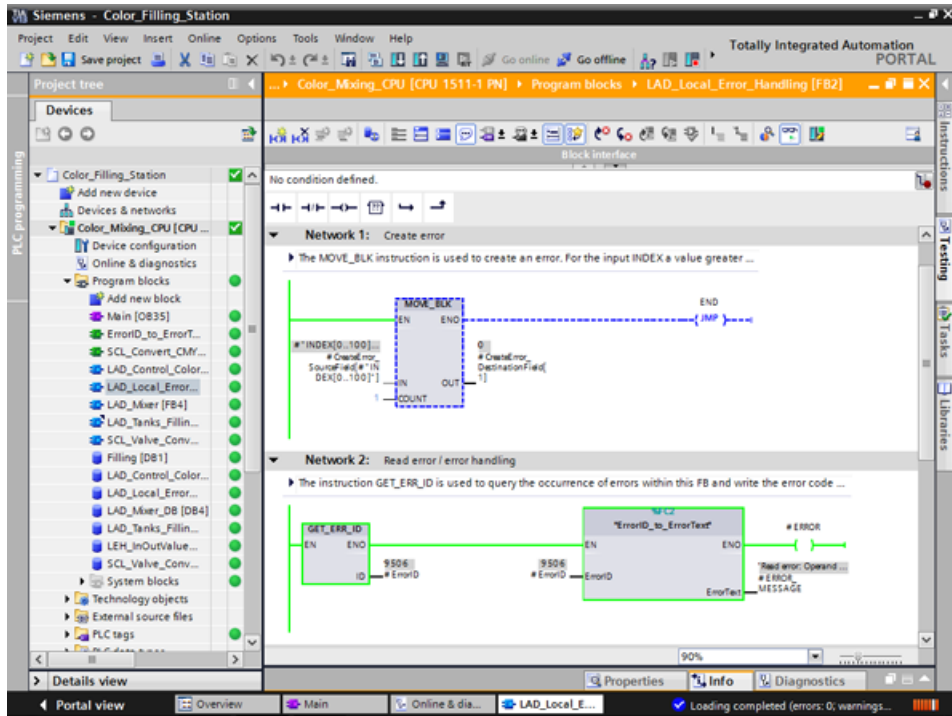
1. Open the "LAD\_Local\_Error\_Handling" function block.
2. Insert the "GET\_ERR\_ID" instruction in the second network and interconnect the "ID" output.



3. Call the "ErrorID\_to\_ErrorText" function from the project tree.
4. Interconnect the parameters of the "ErrorID\_to\_ErrorText" function so that they can convert the error code into an error message.
5. Load the changes to the CPU.
6. Trigger an error in the "Main" organization block by entering an invalid value, for example, "101". An error message is output at the "ERROR\_MESSAGE" parameter.

### Result

The error message is output as long as the error is not corrected. To correct the error, assign the "LEH\_INDEX" a valid value or restart the CPU.



## **3.3 Configure visualization**

### **3.3.1 Present sample project**

#### **Sample project for the application**

To configure the color mixing system with the TIA Portal, create the sample project "Color\_Filling\_Station". The following project components already exist for the sample project: The program blocks and tag tables of the CPU user program and a configured Comfort Panel with the necessary HMI screens, HMI tags and scripts.

In this section, we will explain the relationships between the individual project components of the sample project. You will carry out the necessary configuration steps yourself at a later point in time.

### **3.3.2 HMI configuration**

#### **3.3.2.1 Overview**

##### **Introduction of HMI configuration**

The supplied project includes the programmed CPU and the preconfigured HMI device in the "global library".

##### **HMI configuration**

In this section, we will introduce the HMI device and HMI configuration.

##### **Additional information**

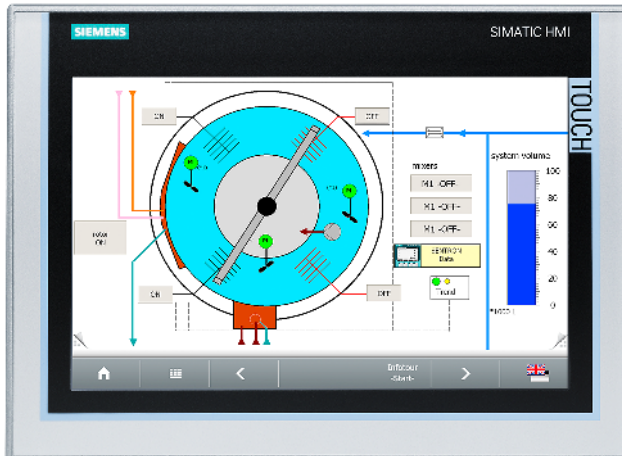
For detailed instructions on HMI configuration see:

Getting Started WinCC V13 Comfort Panels, Runtime Advanced

### 3.3.2.2 SIMATIC HMI Comfort Panels

#### SIMATIC HMI Comfort Panels

The TP1200 Comfort HMI device from the Comfort Panel series is used to operate the color mixing system.



Comfort Panels are particularly suitable for challenging HMI tasks in PROFINET and PROFIBUS environments and are characterized by the following features:

- High-quality housing and numerous interfaces
- Industrial widescreen displays with large visualization area, optimum viewing angle stability and maximum brightness
- Installation either in horizontal or vertical format
- Exact diagnostics with system diagnostics viewer

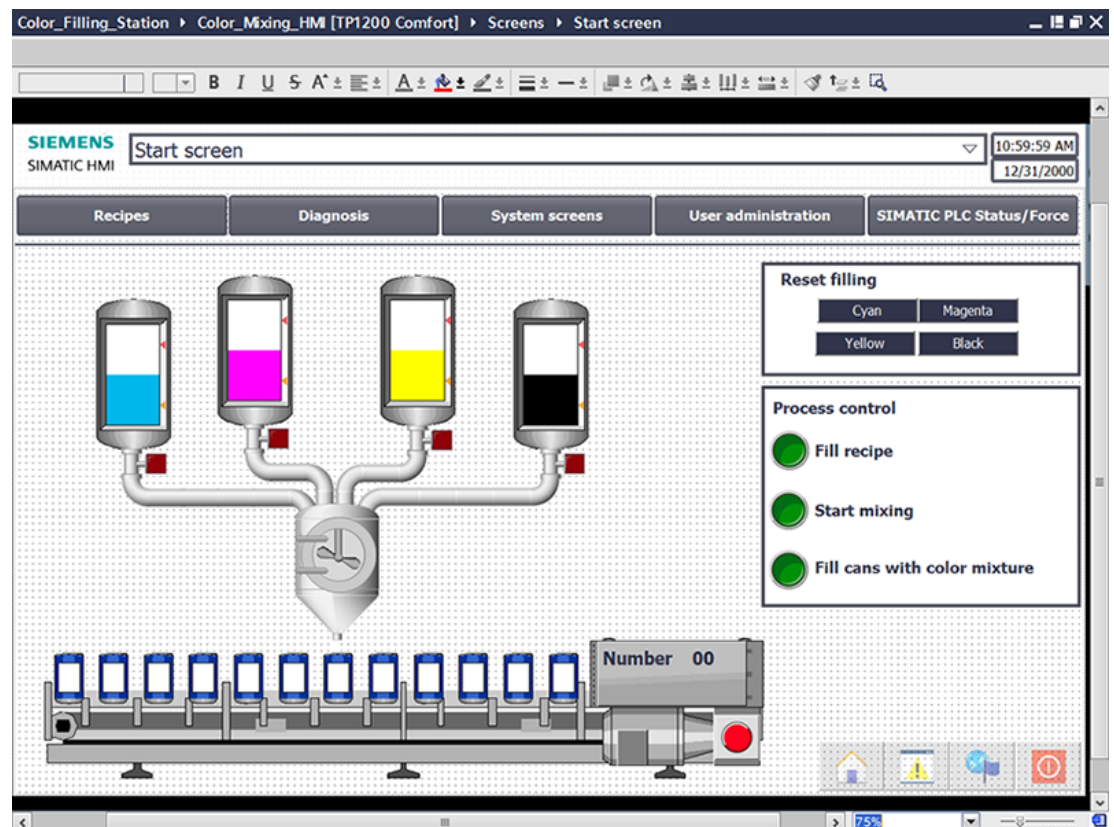
### 3.3.2.3 HMI screens

#### HMI screens

You use the screens loaded onto the respective HMI device to operate and monitor machines and plants in runtime.

You manage the screens in WinCC under "Screens" in the project navigation.

The start screen of the HMI device is used to visualize the color mixing system as well as the most important status information and key figures.



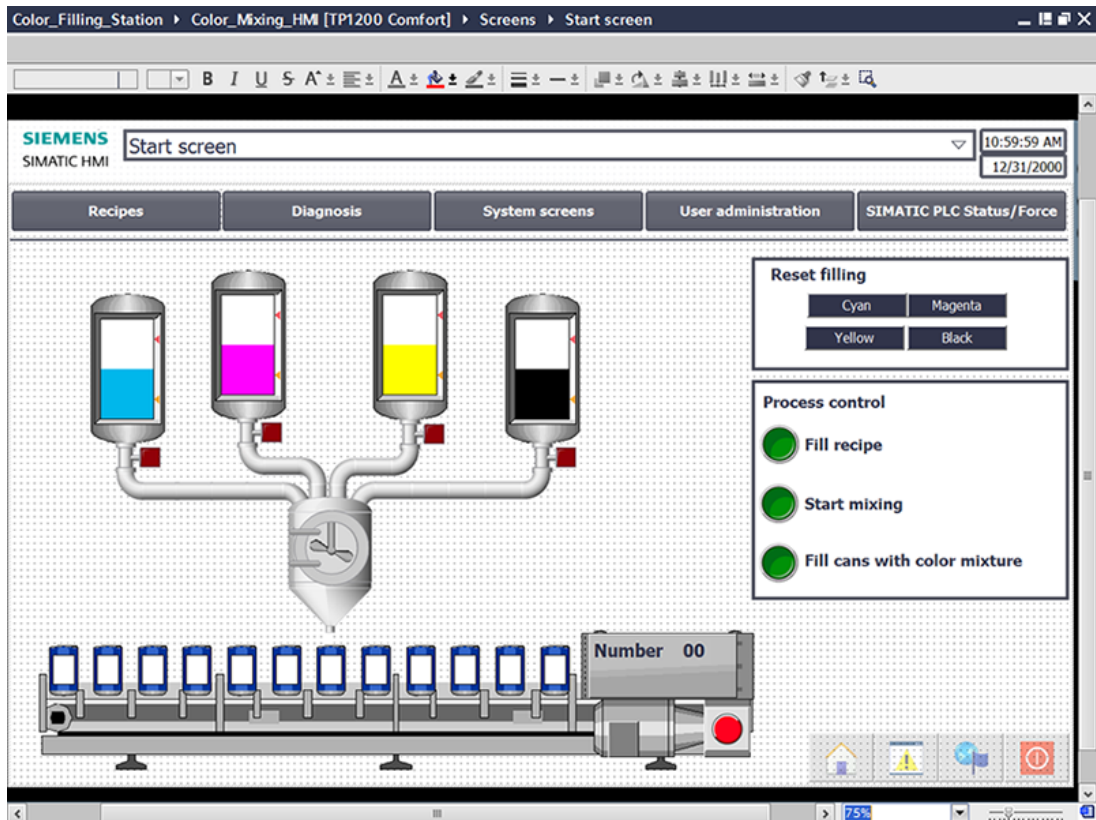
The color mixing system includes the following elements:

- A color reservoir for each print color with fill level display
- Mixer
- Feeder pipes to the mixer
- Conveyor belt with emergency stop switch

### 3.3.2.4 Additional control elements

#### Additional control elements

The process steps "Mixing color" and "Filling color" are to be displayed as animations with dynamic visualization objects.



The start screen of the example project includes additional control objects:

- Buttons for screen changes
- Buttons to reset the fill levels
- Buttons for operating and monitoring the system: Fill recipe, start mixing process, fill color mixture

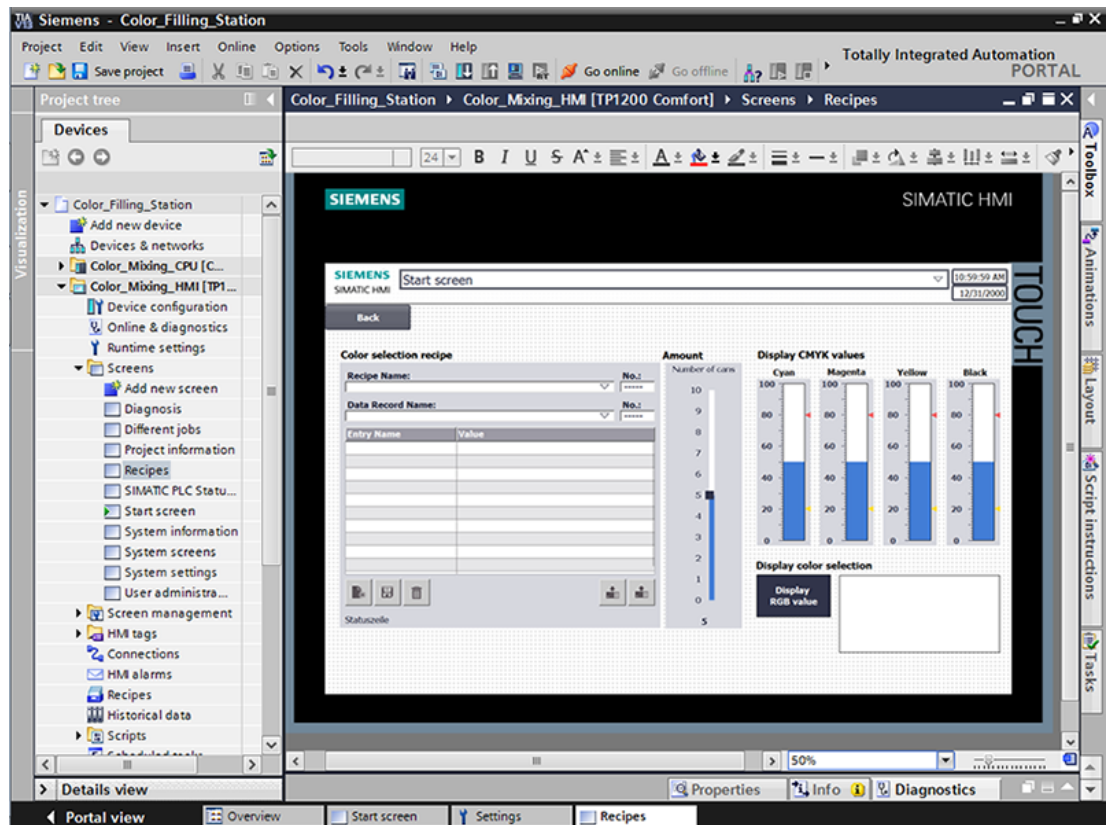


### 3.3.2.5 Recipes

#### Recipes

A recipe contains related production parameters, such as mixing ratios.

The required mixing ratio can be transferred from the HMI device to the color mixing system in a single step, for example, to switch production from dark orange to signal yellow.



The color mixing system can produce the mixed colors "Orange", "Amber", "Green" and "Red".

A recipe data record is created for each color. The recipe data record includes the percentage of basic colors which result in the respective mixed color.

The recipe consists of relevant parameters and the recipe data records in which the mixing ratios for the individual shades of color are stored.

### 3.3.2.6 Archives

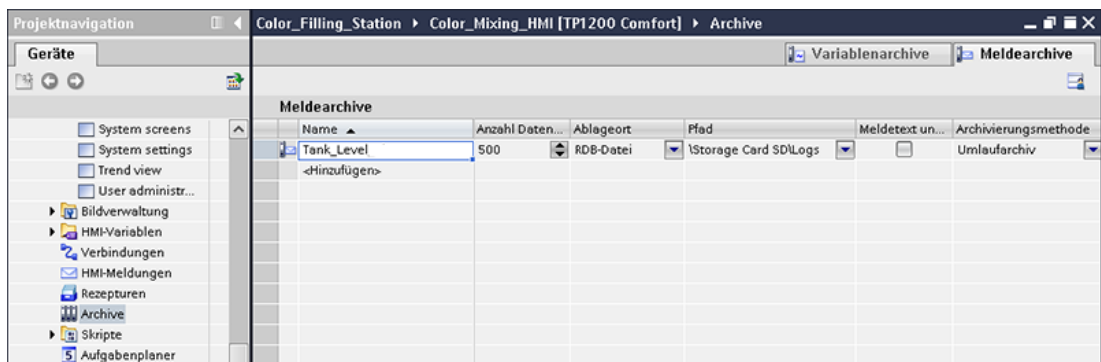
#### Archives

To record operational events of a system, the alarms and process values generated during production are saved to logs.

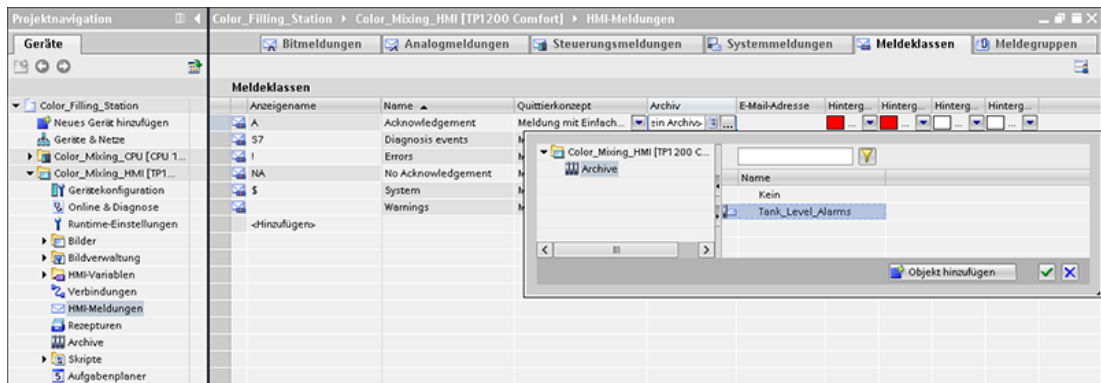
You can then evaluate the alarms and process data logs.

The fill levels of the color reservoirs are to be documented for the color mixing system.

You have configured the alarm log "Tank\_Level" for this purpose.



This log stores alarms for fill levels that were too low and fill levels that were too high during a shift.

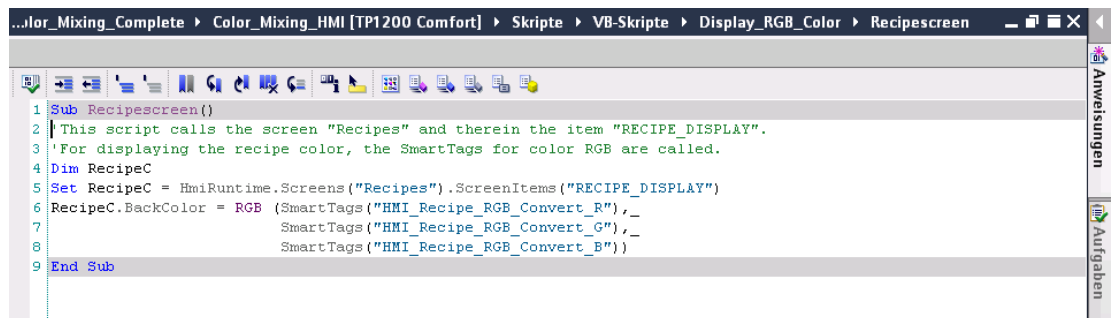


### 3.3.2.7 User-defined functions

#### Scripts

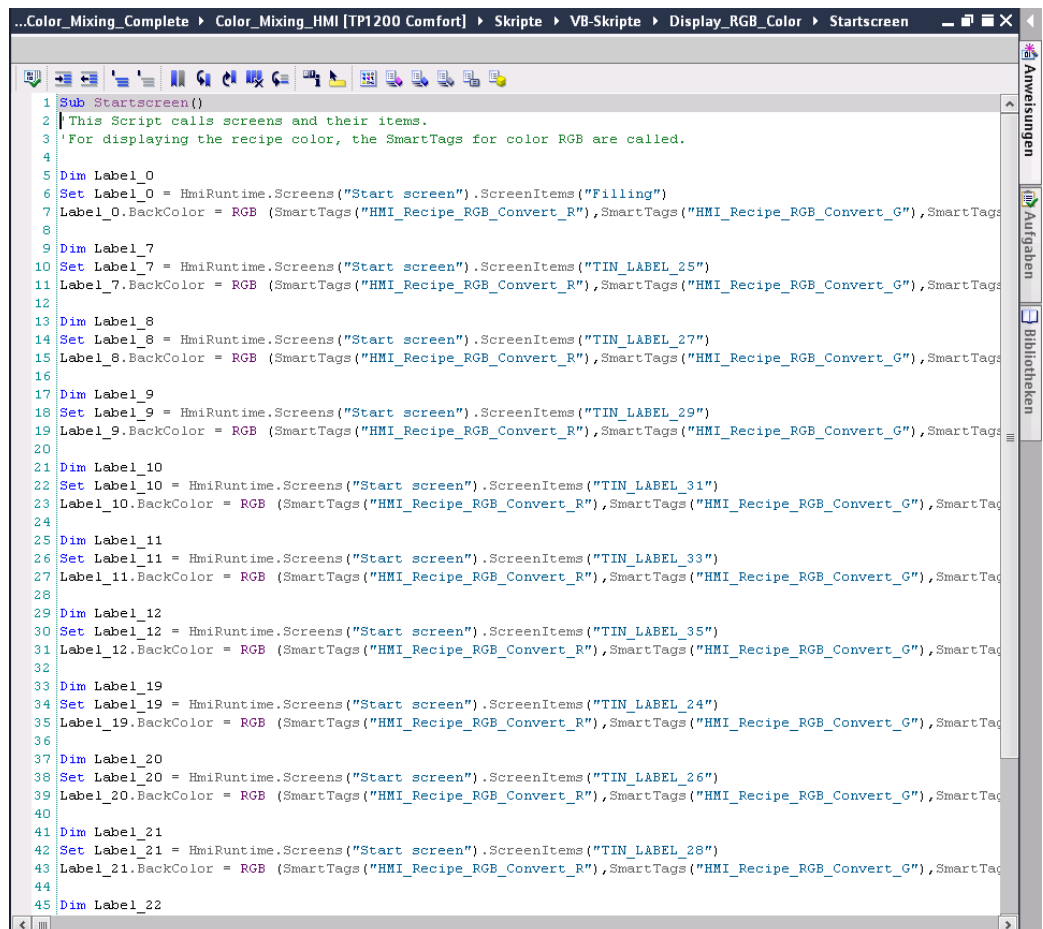
You use the user-defined functions to program additional functionality for the HMI device. WinCC offers a VBS programming interface to create user-defined functions. The example project uses two user-defined functions to display the mixed print color on the monitor in different screens.

- "Recipescreen" displays a rectangle in the selected color in the "Recipes" screen.



```
1 Sub Recipescreen()  
2 | This script calls the screen "Recipes" and therein the item "RECIPE_DISPLAY".  
3 | For displaying the recipe color, the SmartTags for color RGB are called.  
4 Dim RecipeC  
5 Set RecipeC = HmiRuntime.Screens("Recipes").ScreenItems("RECIPE_DISPLAY")  
6 RecipeC.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"),_  
7 SmartTags("HMI_Recipe_RGB_Convert_G"),_  
8 SmartTags("HMI_Recipe_RGB_Convert_B"))  
9 End Sub
```

- "Startscreen" displays the label of the filled cans in the currently mixed color in the system overview of the start screen.

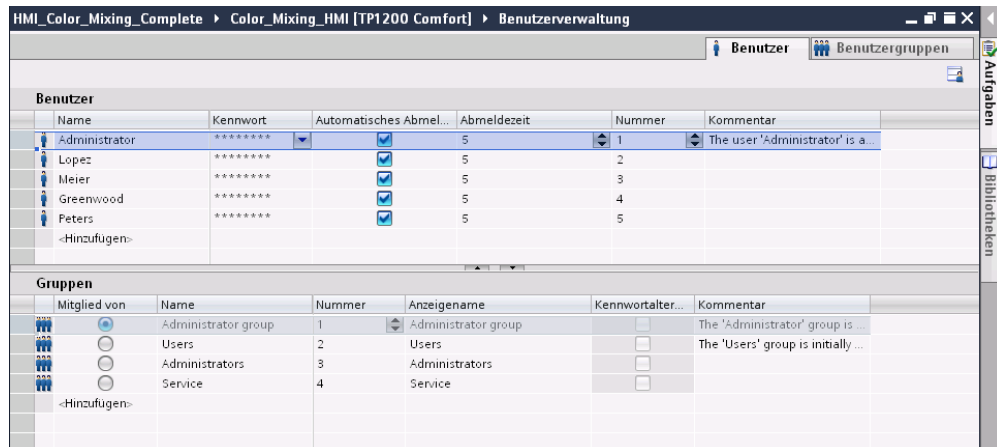


```
1 Sub Startscreen()  
2 | This Script calls screens and their items.  
3 | For displaying the recipe color, the SmartTags for color RGB are called.  
4  
5 Dim Label_0  
6 Set Label_0 = HmiRuntime.Screens("Start screen").ScreenItems("Filling")  
7 Label_0.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
8  
9 Dim Label_7  
10 Set Label_7 = HmiRuntime.Screens("Start screen").ScreenItems("TIN_LABEL_25")  
11 Label_7.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
12  
13 Dim Label_8  
14 Set Label_8 = HmiRuntime.Screens("Start screen").ScreenItems("TIN_LABEL_27")  
15 Label_8.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
16  
17 Dim Label_9  
18 Set Label_9 = HmiRuntime.Screens("Start screen").ScreenItems("TIN_LABEL_29")  
19 Label_9.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
20  
21 Dim Label_10  
22 Set Label_10 = HmiRuntime.Screens("Start screen").ScreenItems("TIN_LABEL_31")  
23 Label_10.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
24  
25 Dim Label_11  
26 Set Label_11 = HmiRuntime.Screens("Start screen").ScreenItems("TIN_LABEL_33")  
27 Label_11.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
28  
29 Dim Label_12  
30 Set Label_12 = HmiRuntime.Screens("Start screen").ScreenItems("TIN_LABEL_35")  
31 Label_12.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
32  
33 Dim Label_19  
34 Set Label_19 = HmiRuntime.Screens("Start screen").ScreenItems("TIN_LABEL_24")  
35 Label_19.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
36  
37 Dim Label_20  
38 Set Label_20 = HmiRuntime.Screens("Start screen").ScreenItems("TIN_LABEL_26")  
39 Label_20.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
40  
41 Dim Label_21  
42 Set Label_21 = HmiRuntime.Screens("Start screen").ScreenItems("TIN_LABEL_28")  
43 Label_21.BackColor = RGB (SmartTags("HMI_Recipe_RGB_Convert_R"), SmartTags("HMI_Recipe_RGB_Convert_G"), SmartTags("HMI_Recipe_RGB_Convert_B"))  
44  
45 Dim Label_22
```

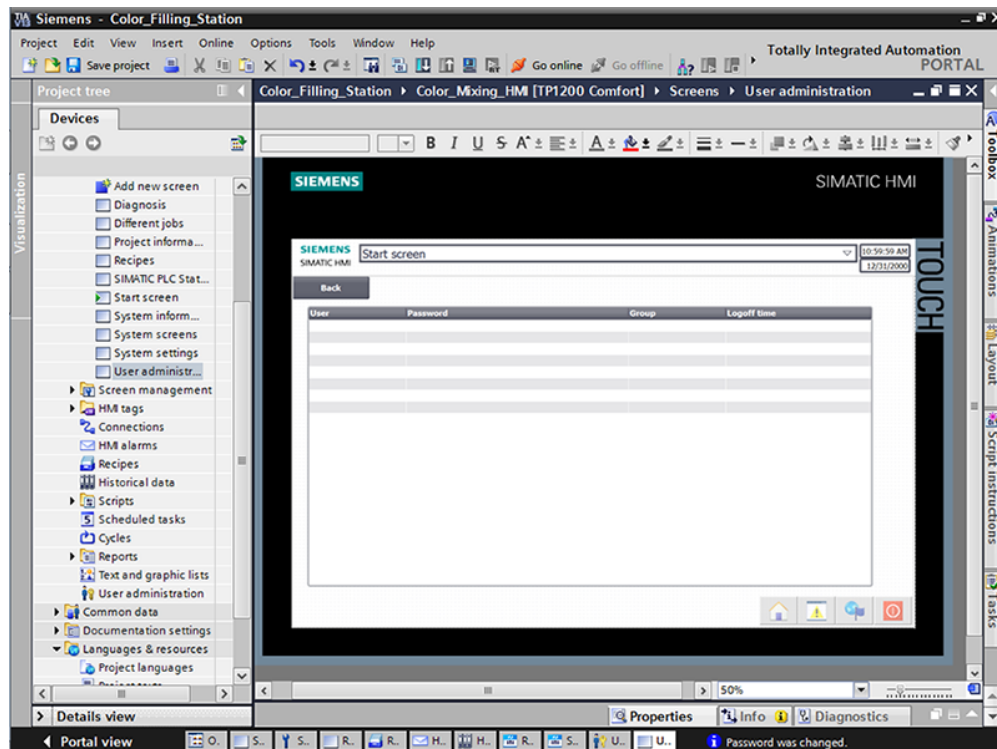
### 3.3.2.8 User Management

#### User Management

WinCC gives you the option to restrict safety-related operations to special user groups and thus protect data and functions from unauthorized access in Runtime.



The "User view" object offers management of users and passwords on the HMI device.



Users with user management authorization have access to the full range of functions in the user view.

They can create and delete users and change their own password or that of other users.

### 3.3.2.9 Multilingualism

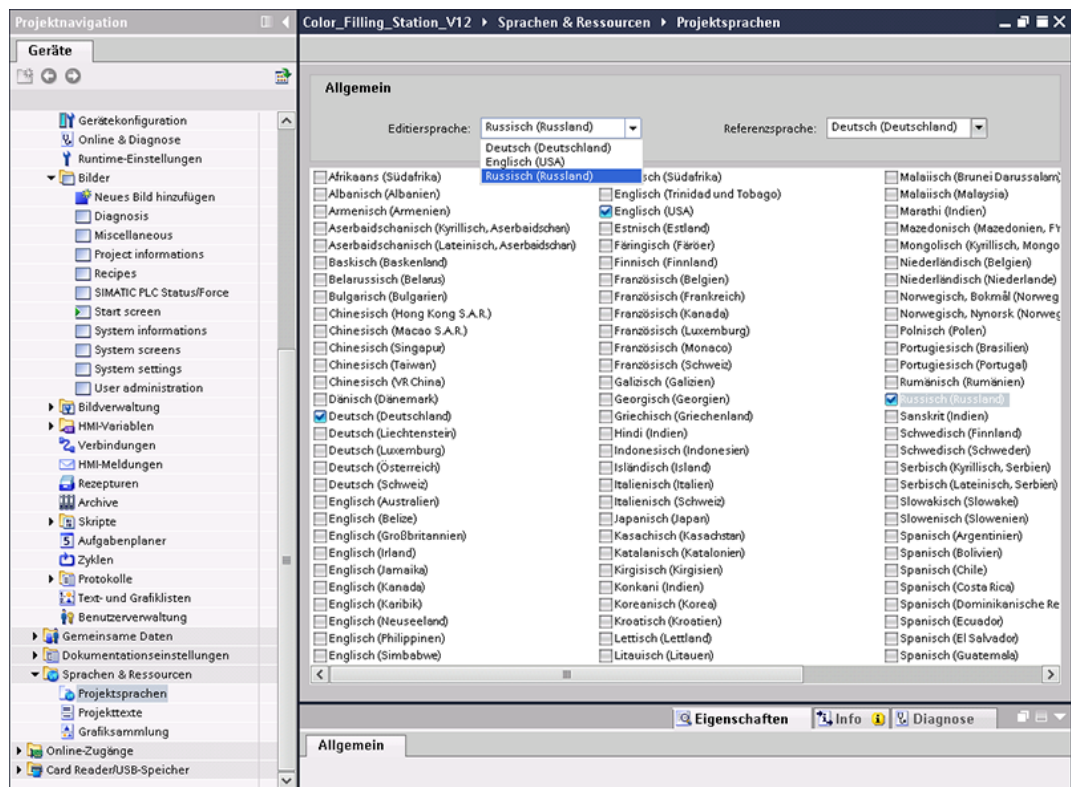
#### Multilingualism

WinCC supports multilingual user interfaces.

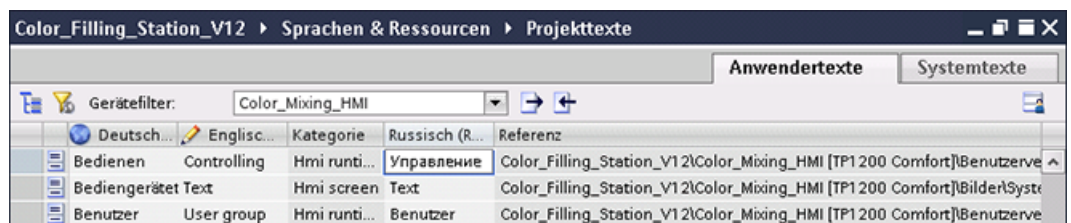
The color mixing system is operated in a new subsidiary in Russia.

A Russian user interface is required for maintenance and service technicians.

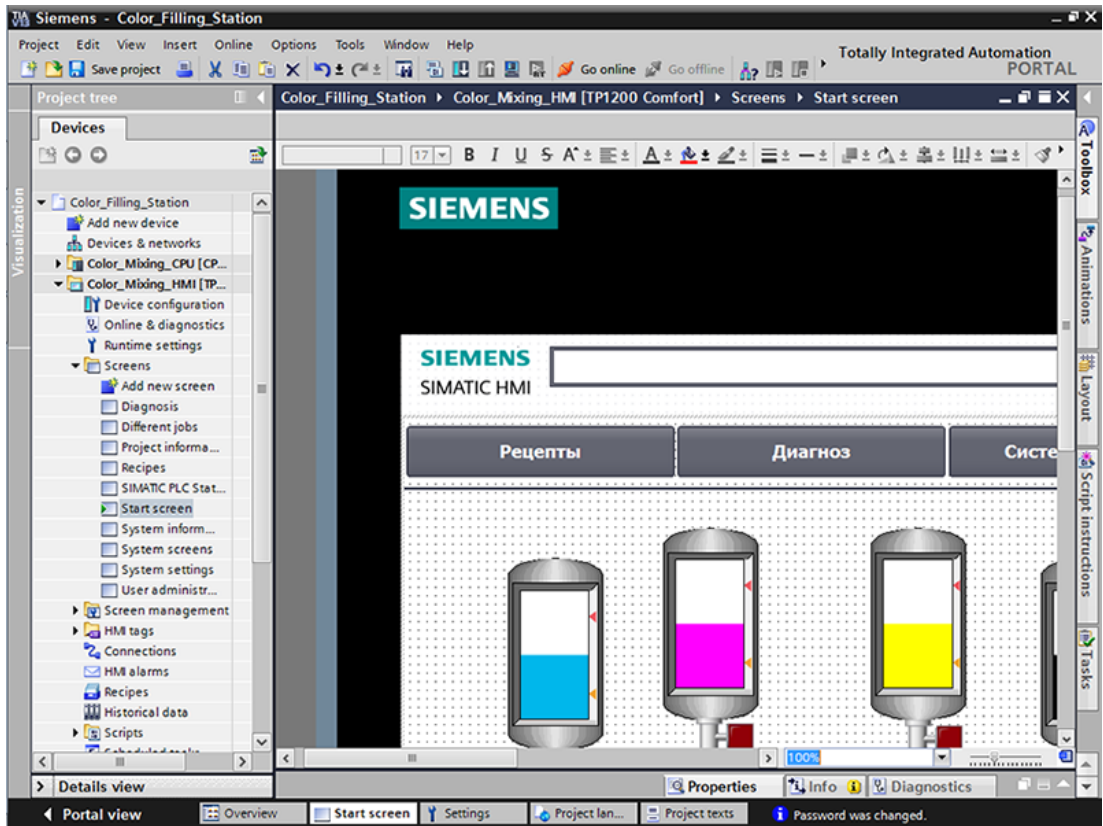
The example project has been expanded by another language for this purpose.



The texts are imported again after they have been exported and translated into Russian.



The Russian texts are displayed in Runtime in case of a language selection.



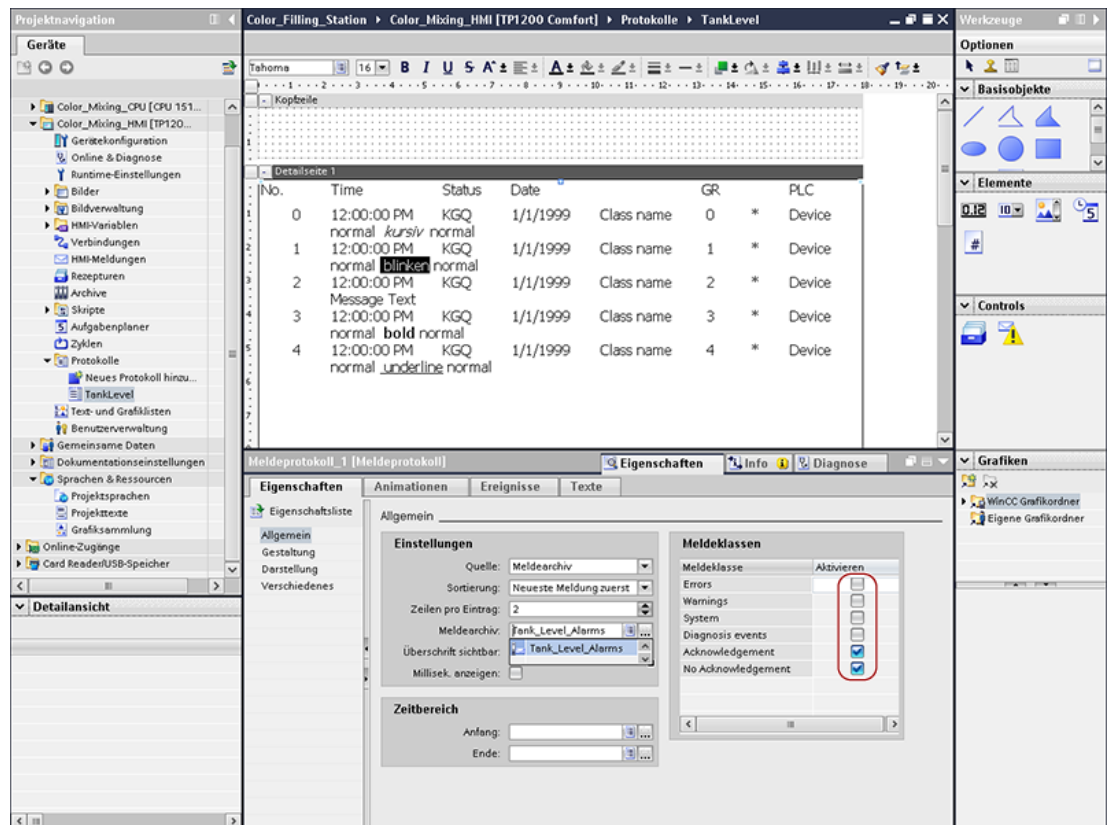


### 3.3.2.10 Reports

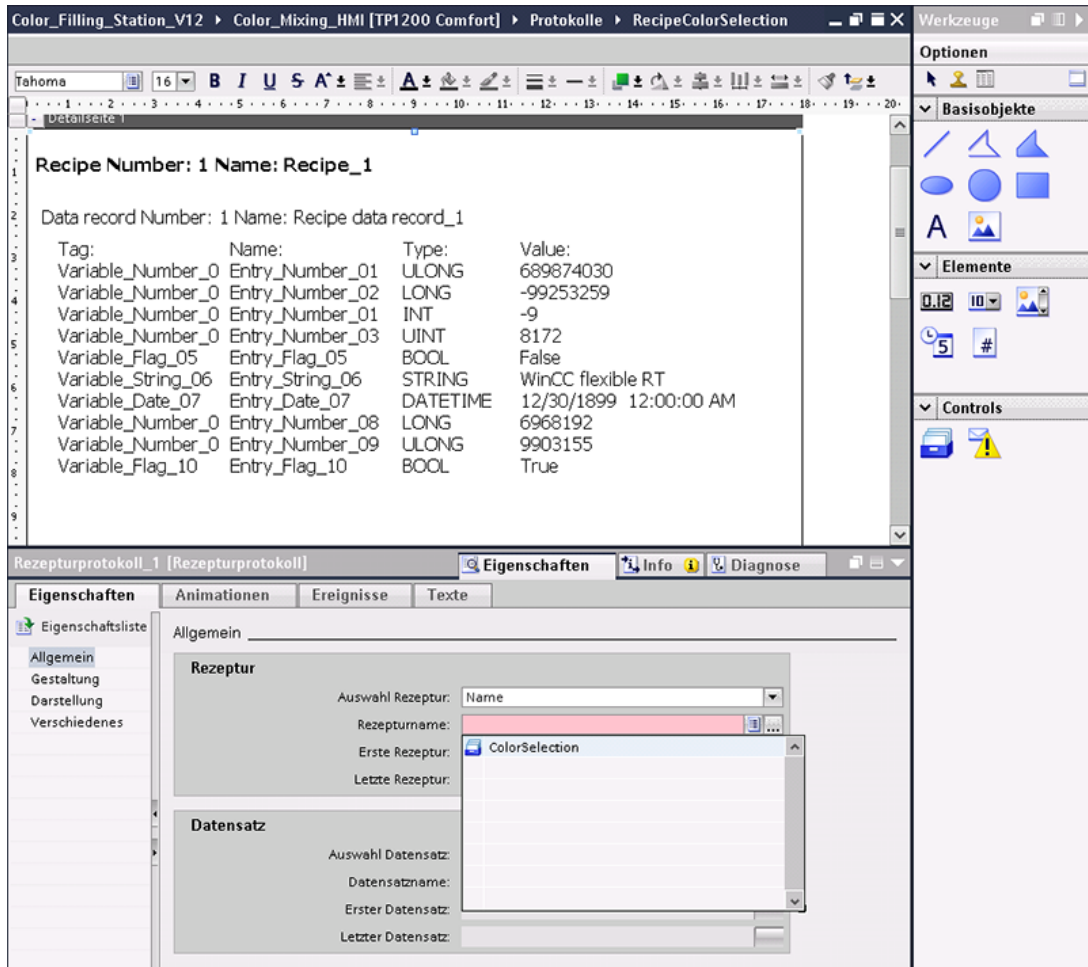
#### Reports

Reports are used to record events in a production process as a basis for product testing and quality control. Alarms and recipe data are output at regular intervals in the form of shift reports for this purpose.

A report has been created in WinCC for the "Tank\_Level" log with alarms for the fill level.



A report for recipes has also been created in this project.



The reports should be output on a daily basis to a printer which is connected to the HMI device.

The cyclical output was created with the help of the Scheduler.



### 3.3.3 Insert HMI device from libraries

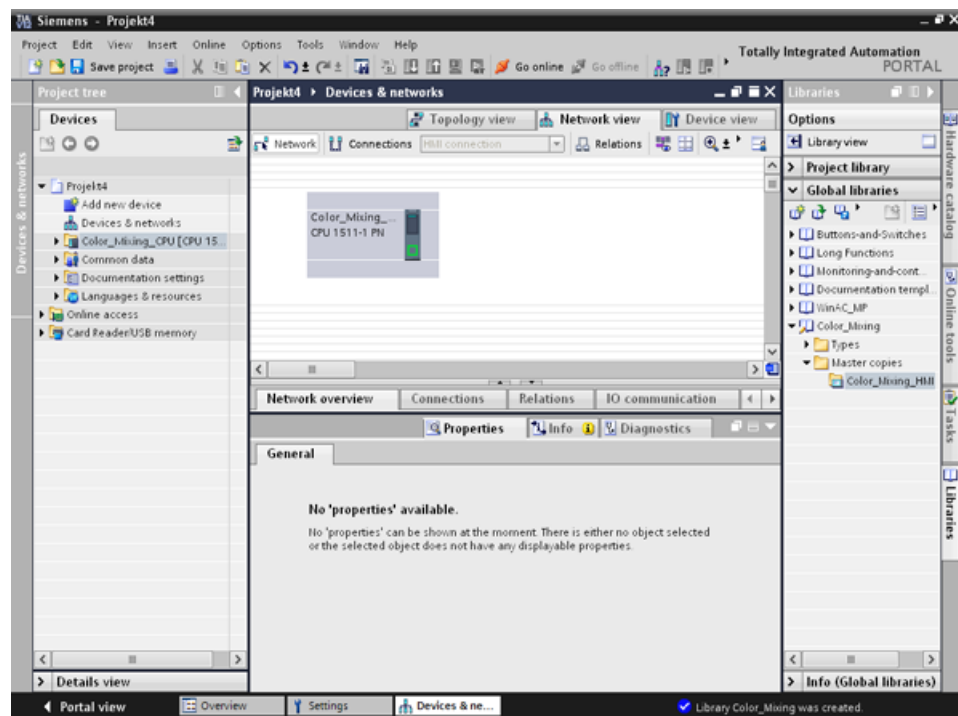
#### 3.3.3.1 Storing an object in a library

##### Introduction

The global library includes a preconfigured HMI device.

##### Procedure

1. Open the global library.
2. Drag-and-drop the HMI device "Color\_Mixing\_HMI" into the "Devices & Networks" editor.



3. The mouse pointer changes into a crosshair with an object symbol attached.

##### Result

The preconfigured HMI device is created and can be connected to the CPU.

### 3.3.4 Configuring HMI connection

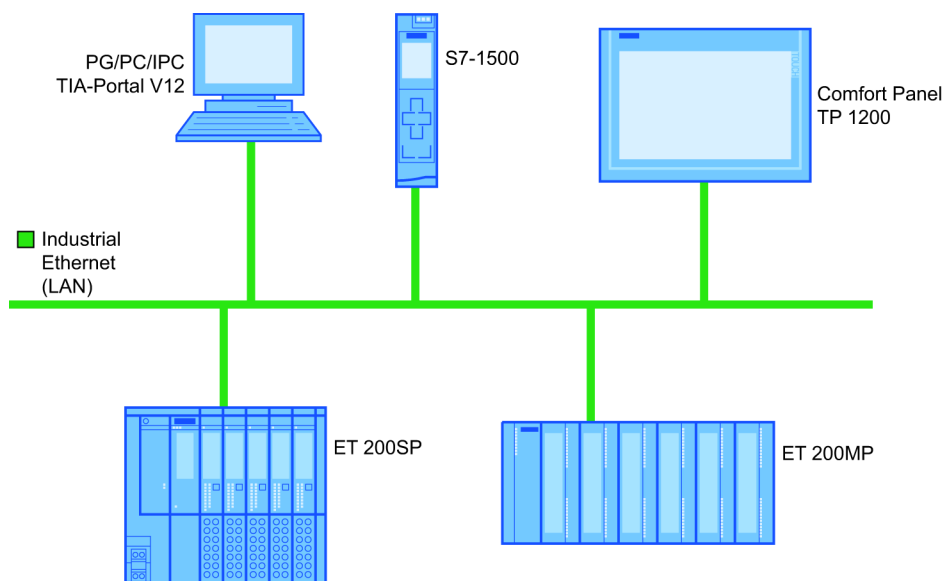
#### 3.3.4.1 Communication between devices

##### Communication

The data exchange between devices is referred to as communication.

The devices can be interconnected directly or via a network.

The interconnected devices in communication are referred to as communication partners.



Data transferred between the communication partners may serve different purposes:

- Display processes
- Operate processes
- Output alarms
- Archive process values and alarms
- Document process values and alarms
- Administer process parameters and machine parameters

##### Basic information for all communication

The basis for all types of communication is a network configuration. In a network configuration, you specify the connection that exists between the configured devices.

With the network configuration, you also ensure the necessary prerequisites for communication, in other words:

- Every device in a network is assigned a unique address.
- The devices carry out communication with consistent transmission characteristics.

### 3.3.4.2 Configuring HMI connection

#### Introduction

You configure an HMI connection between the Comfort Panel TP1200 and the CPU via PROFINET in the "Devices & Networks" editor.

The available communication partners in the project are displayed graphically in the network view.

 <b>CAUTION</b>
<b>Communication via Ethernet</b>
In Ethernet-based communication, the end user is responsible for the security of his data network.
Targeted attacks can overload the device and interfere with proper functioning.

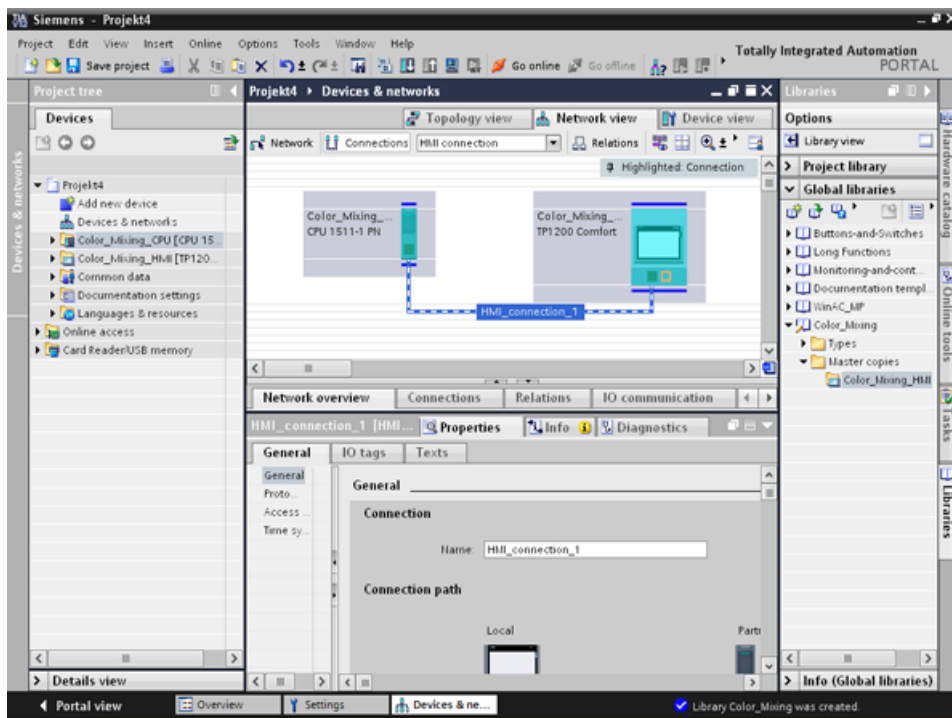
#### Requirements

The following communication partners are created in the "Devices & Networks" editor:

- HMI device: SIMATIC Comfort Panel
- CPU: SIMATIC S7-1500

### Procedure

1. Click the "Connections" button and select "HMI connection" for the connection type. The devices available for connection are highlighted in color.
2. Click the PROFINET interface of the CPU and drag-and-drop a connection to the PROFINET interface of the HMI device.



3. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.

### Note

The created HMI connection is also shown in the tabular area of the editor in the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

### Result

You have created a connection between an HMI device and the CPU.

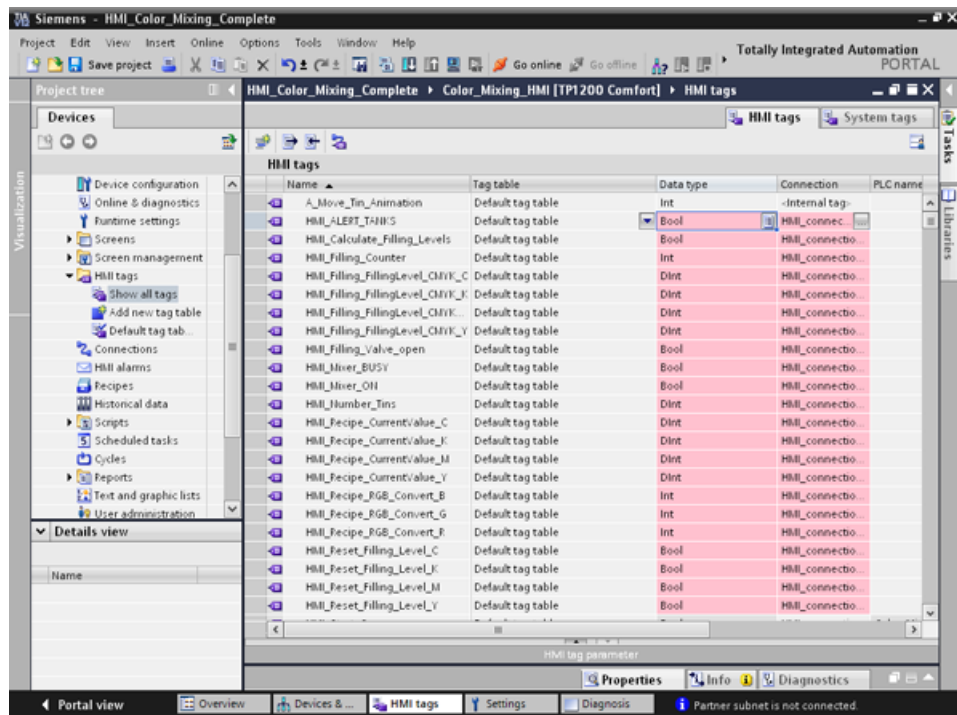
### 3.3.4.3 Connecting HMI tags

#### Introduction

Once you have created the connection of CPU and HMI device, connect the tags of the two devices.

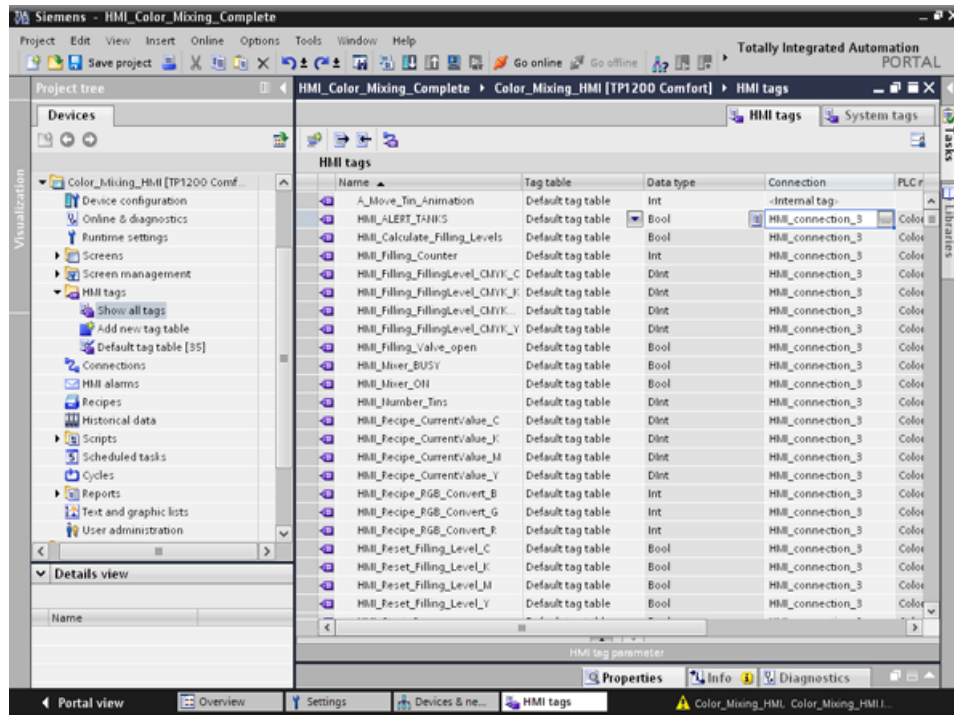
#### Procedure

1. Open the HMI tag editor.



2. Select the HMI connection you have just configured in the "Connections" column.

3. Repeat this procedure for all entries highlighted in red.



## Result

An HMI connection had already been created for tags already configured in the CPU and HMI device.

You have restored this HMI connection.

## **3.3.5 Configuring system diagnostics**

### **3.3.5.1 System diagnostics basics**

#### **Introduction**

You use system diagnostics to detect problems and errors in any part of your plant. WinCC has two display and operating elements for quick error localization.

#### **System diagnostics view**

The alarm view shows the status of a CPU while the system diagnostics view gives you an overview of all devices available in your system: You navigate directly to the cause of the error and to the relevant device. You have access to all devices supporting diagnostics you have configured in the "Devices & networks" editor.

#### **System diagnostics window**

The system diagnostics window is an operating and display element that you can only use in the global screen.

The functions of the system diagnostics window are no different than those of the system diagnostics view. Because the system diagnostics window is configured in the global screen, you can, for example, also specify if the object is closable in Runtime.

### 3.3.5.2 System diagnostics views

#### Introduction

There are four different views available in the system diagnostics display and the system diagnostics window.

- Device view
- Diagnostic buffer view
- Detail view
- Matrix view (for master systems, PROFIBUS, PROFINET only)

#### Device view

The device view shows all the available devices of a layer in a table. Double-clicking on a device opens either the child devices or the detail view. Symbols in the first column provide information about the current status of the device.

Status	Name	Operating state	Slot	Type	Order number	Address	Plant design...
✓	S7-1500-Station_1			S7-1500-Station		32*	
✓	CPU-Proxy_1	■	1	CPU 1511-1 PN	6ES7 511-1AK00-0AB0	49*	



### Diagnostic buffer view

The current data from the diagnostic buffer are shown in the diagnostic buffer view.

N	Date	Time	Event
1	1/11/2012	10:49:59 PM	Communication initiated request: WARM RESTART - CPU changes from STARTUP to RUN ...
2	1/11/2012	10:49:59 PM	Communication initiated request: WARM RESTART - CPU changes from STOP to STARTUP...
3	1/11/2012	10:49:56 PM	Communication initiated request: STOP - CPU changes from RUN to STOP mode
4	1/11/2012	7:32:36 AM	Communication initiated request: WARM RESTART - CPU changes from STARTUP to RUN ...
5	1/11/2012	7:32:36 AM	Communication initiated request: WARM RESTART - CPU changes from STOP to STARTUP...
6	1/11/2012	7:32:34 AM	Communication initiated request: STOP - CPU changes from RUN to STOP mode
7	1/11/2012	7:30:53 AM	Communication initiated request: WARM RESTART - CPU changes from STARTUP to RUN ...
8	1/11/2012	7:30:53 AM	Communication initiated request: WARM RESTART - CPU changes from STOP to STARTUP...
9	1/11/2012	7:21:03 AM	Communication initiated request: STOP - CPU changes from RUN to STOP mode
...	1/11/2012	4:33:55 AM	Communication initiated request: WARM RESTART - CPU changes from STARTUP to RUN ...
...	1/11/2012	4:33:55 AM	Communication initiated request: WARM RESTART - CPU changes from STOP to STARTUP...
...	1/11/2012	4:22:31 AM	Follow-on operating mode change - CPU changes from STOP (initialization) to STOP mode
...	1/11/2012	4:22:31 AM	Power on - CPU changes from NO POWER to STOP (initialization) mode
...	1/11/2012	4:21:59 AM	Power off - CPU changes from STOP to NO POWER mode
...	1/11/2012	4:19:51 AM	System initiated request: STOP - CPU changes from STOP (firmware update) to STOP mode
...	1/11/2012	4:19:51 AM	Device firmware update finished - CPU changes to STOP mode, new startup inhibit set: .
...	1/11/2012	4:19:51 AM	CPU firmware update: Completed successfully - Device firmware update sequence contin...
...	1/11/2012	4:19:03 AM	System initiated request: Firmware update - CPU changes from STOP (initialization) to ST...

### Detail view

The detail view gives detailed information about the selected device and any pending errors. Check whether the data is correct in the detail view. You can not sort error texts in the detail view.

S7-1500 station_1 \ DI32	
> Status	
> Name	DI32
> Betriebszustand	
> Baugruppenträger	0
> Steckplatz	3
> Typ	DI32
> Bestellnummer	6ES7 521-1BL00-0AB0
> Adresse	259*
> Anlagenbezeichnung	
> Positionskennung	
> Hersteller ID	SIEMENS AG
> Hardware Version	97
> Profil ID	
> spezifische Profildaten	0003
> I&M Datenversion	1.1
> Fehlertext	Drahtbruch
	Hilfe: Kontrollieren Sie den Zustand der Verbindungsleitungen.






### Matrix view

The matrix view is only available for master systems. The matrix view shows the status of the subdevices of the master system.

- In PROFIBUS, the numbers assigned by Profibus are used as identification (DP station number).
- The IO devices are numbered consecutively from 1 in PROFINET.



### Navigation buttons

Button	Function
	Opens the child devices or the detail view if there are no child devices.
	Opens the parent device or the device view if there is no parent device.
	Opens the device view.
	Opens the diagnostic buffer view. Only visible in the device view.
	Refreshes the view. Configured softkey, for example F2

### 3.3.5.3 Configuring the system diagnostic view

#### Introduction

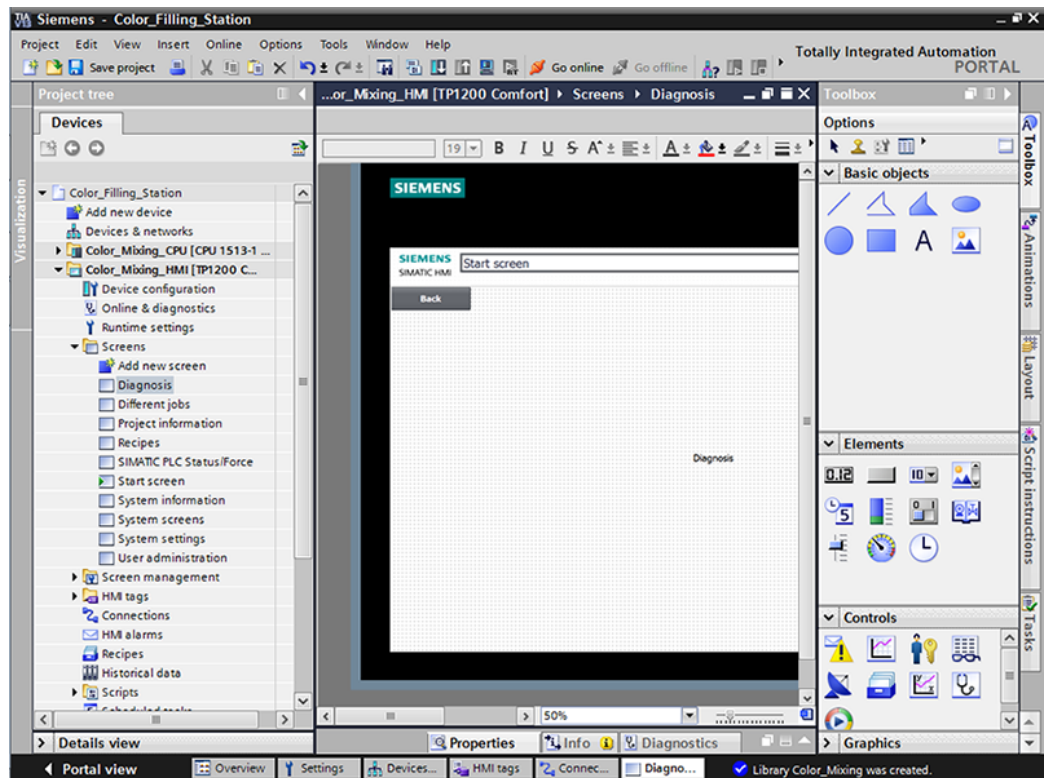
You add a system diagnostics view to your project to get an overview of all devices available in your plant.

#### Requirements

- CPU has been created.
- The Inspector window is open.

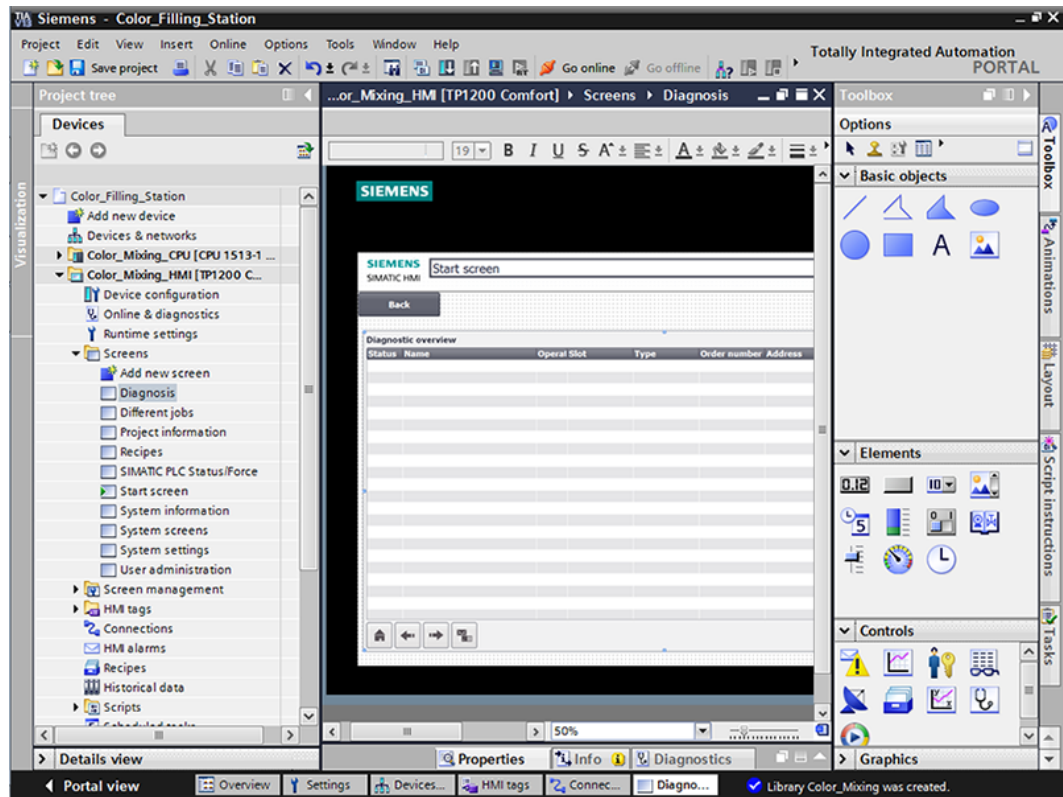
#### Procedure

1. Double-click the "Diagnostics" HMI screen.



3.3 Configure visualization

2. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.



3. Select "Properties > Properties > Columns > Devices/Detail view" in the Inspector window.
4. Enable the columns that you require in the device view for Runtime, for example, State, Name, Slot.
5. Enable the columns that you require in the detail view for Runtime, for example, State, Name, Higher level designation.
6. Enable the columns that you require in the diagnostics buffer view, for example: State, Name, Rack.
7. If necessary, adapt the headers to the columns.
8. Enable "Properties > Properties > Layout > Column settings > Columns moveable" to move the columns in Runtime.
9. You can change the column headers under "Properties > Properties > Column headers", if necessary.

**Result**

The system diagnostics view has been added to the "Diagnostics" screen.

Error messages for the entire plant are now displayed in the system diagnostics view in Runtime.

## **3.3.6 Simulating an HMI device**

### **3.3.6.1 Simulation basics**

#### **Introduction**

You can use the simulator to test the performance of your configuration on the configuration PC. This allows you to quickly locate any logical configuration errors before productive operation.

You can start the simulator as follows:

- In the shortcut menu of the HMI device or in a screen: "Start simulation"
- Menu command "Online > Simulation > [Start|With tag simulator|With script debugger]"
- Under "Visualization > Simulate device" in the portal view.

#### **Requirement**

The simulation/runtime component is installed on the configuration PC.

#### **Field of application**

You can use the simulator to test the following functions of the HMI system, for example:

- Checking limit levels and alarm outputs
- Consistency of interrupts
- Configured interrupt simulation
- Configured warnings
- Configured error messages
- Check of status displays

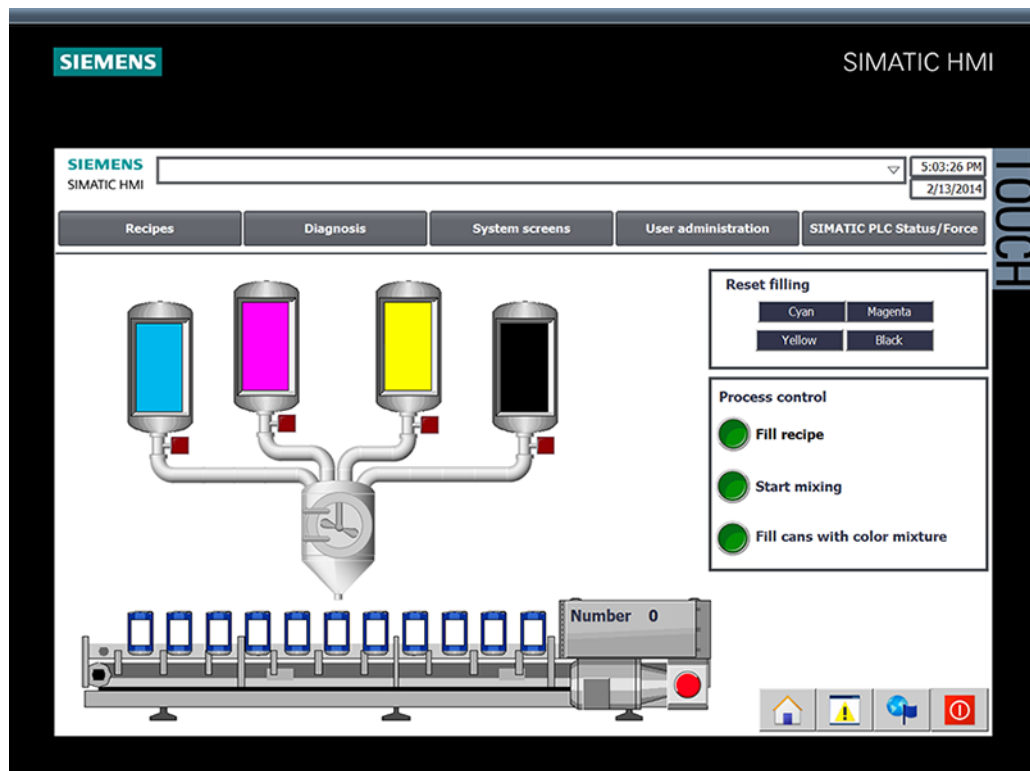
### 3.3.6.2 Operating the panel in simulation

#### Introduction

You simulate the HMI project on your computer.

#### Procedure

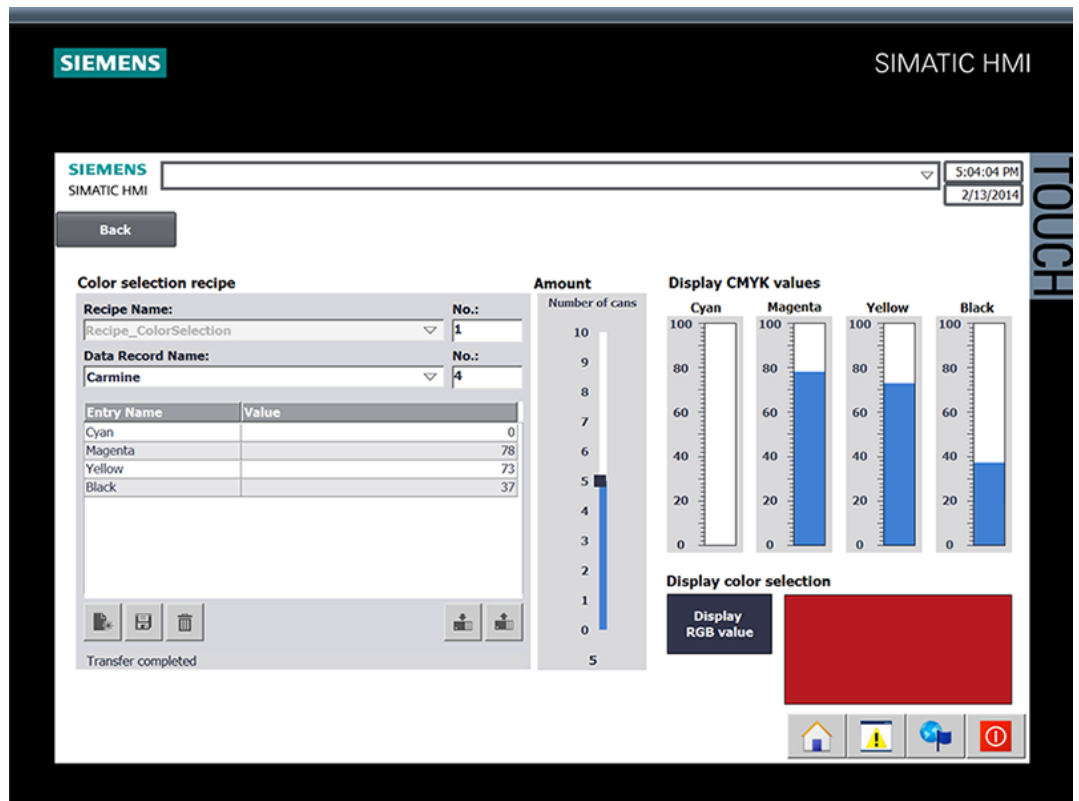
1. Start the simulation of the HMI device.



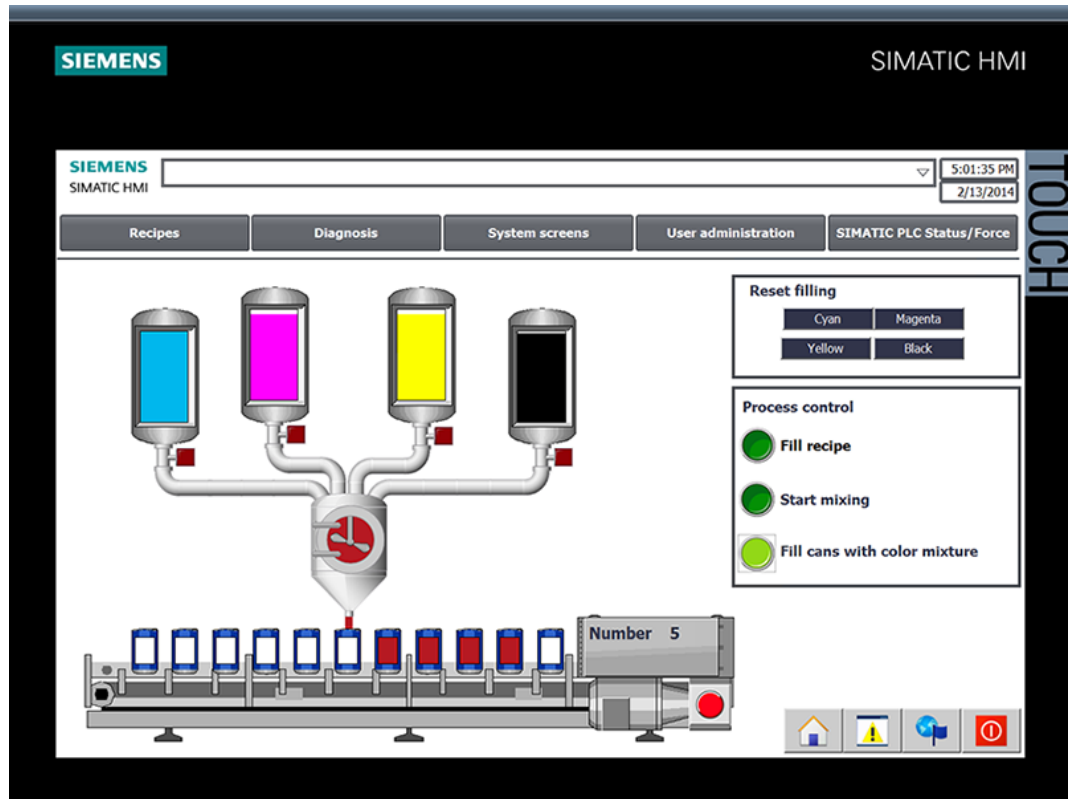
A connection to the CPU is established and the color mixing system is displayed in the simulation.

2. Open the "Recipes" screen and select a color.

3. Specify the number of cans and view the selected color.

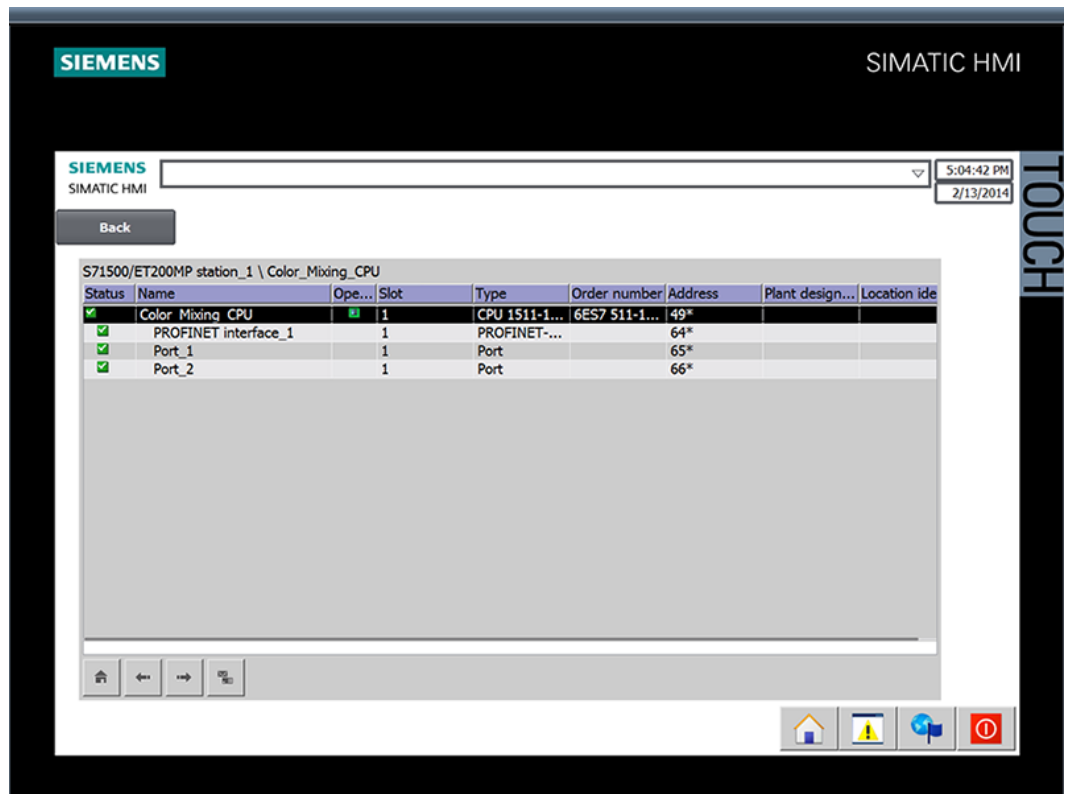


- 4. Go back to the start screen and start production.





5. You can query the current CPU status in the "Diagnostics" screen.



## 3.4 Loading the project into the programming device

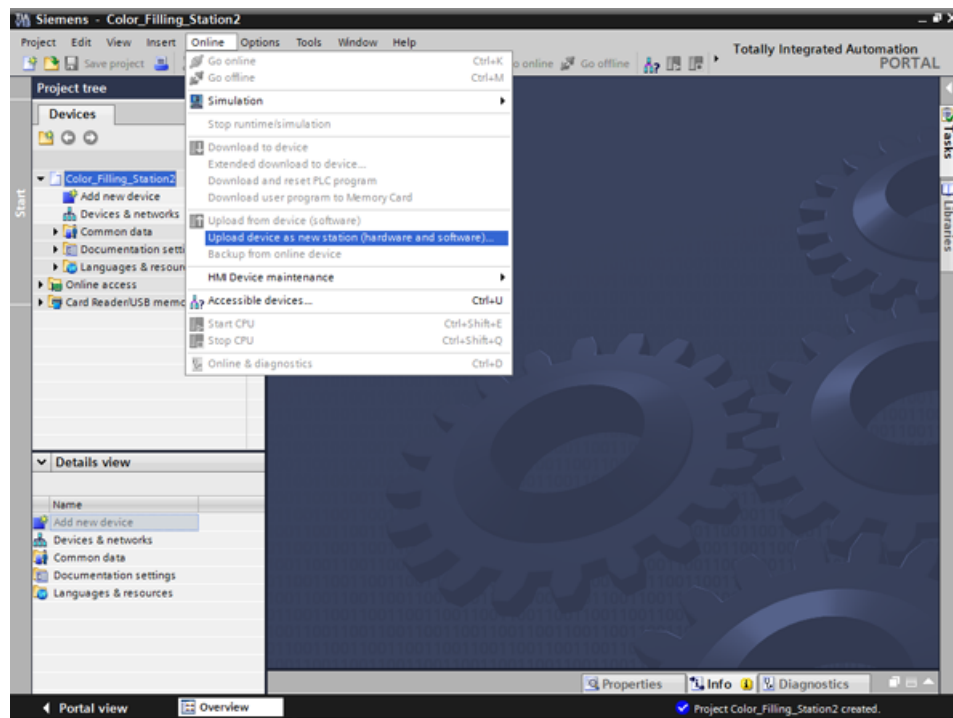
### 3.4.1 Load CPU to project

#### Introduction

You can create a new station including the actual values from the hardware configuration and the user program.

#### Procedure

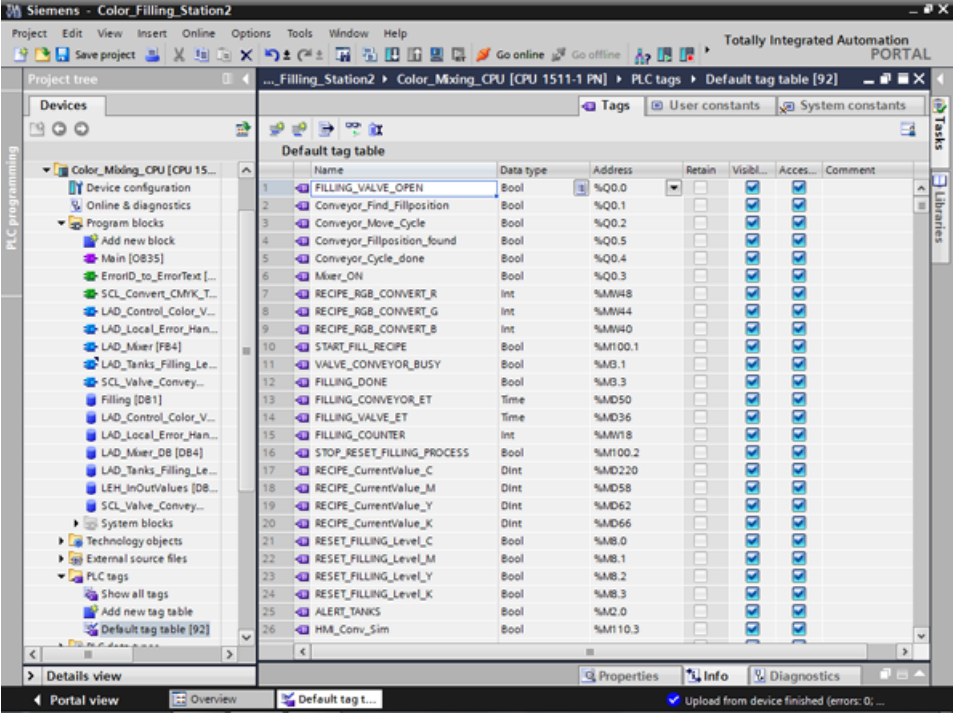
1. Open the dialog for loading from the CPU.



2. Select the interface with which the programming device is connected to the CPU. The search for accessible nodes starts automatically.
3. Load the CPU in the project.

Result

The hardware and software configuration of the CPU are loaded into the project. The project now contains, for example, program blocks and tags.



## 3.5 Team engineering via Inter Project Engineering

### 3.5.1 Basics of "Inter Project Engineering"

#### Introduction

In this section, you will learn about the benefits of team engineering and how to create the required CPU data for an HMI project engineer. As an HMI project engineer, you will learn how to use this CPU data in your project.

#### Distributed configuration

You can use "Inter Project Engineering" to develop the user program and user interface in parallel at different locations. The HMI project engineer requires no CPU user program. There is no need for a STEP 7 installation.

Only tags, blocks, messages and address information of the CPU interfaces are ultimately relevant for the connection of an HMI device to a CPU. The programmer can conveniently export this data to an IPE file, which is imported into the project by the HMI developer. Updates are possible at any time by transferring a new IPE file.

The data are consistent after loading to the CPU and HMI device. The connections to the CPU created in the HMI configuration remain up-to-date.

---

#### Note

You can integrate the new Basic Panels 2nd Generation and Comfort Panels to STEP 7 projects as of V5.4 SP3 with an import into the TIA Portal.

---

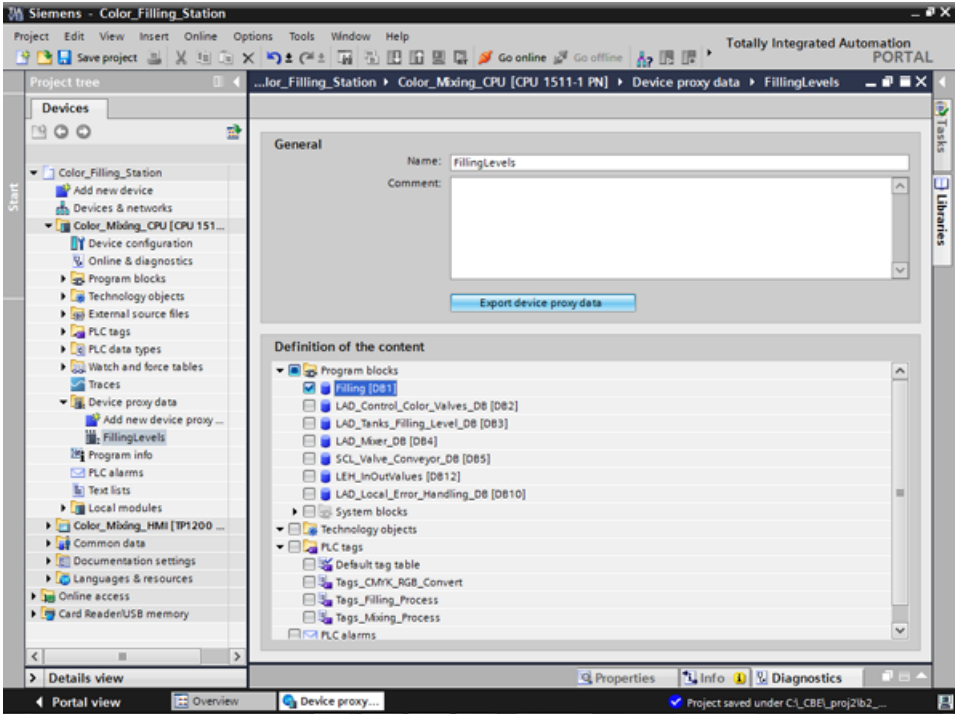
### 3.5.2 Creating an IPE file

#### Introduction

You want to use a compact HMI device to display of fill levels directly at the paint mixing plant. You hire an engineering firm for the visualization and the provide the required CPU data as an IPE file.

#### Procedure

- 1. Add new proxy data for the CPU.
- 2. Enter a name and select the required CPU data.



- 3. Export the proxy data.

#### Result

The IPE file is created. You can send the IPE file, for example, as e-mail to the engineering firm.

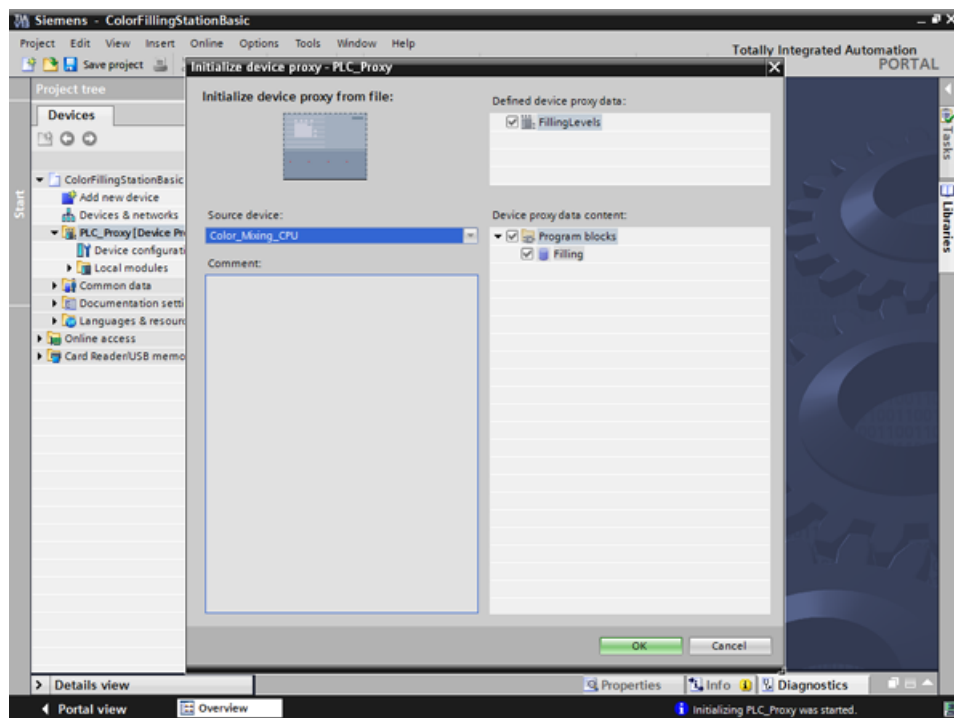
### 3.5.3 Importing an IPE file

#### Introduction

In the engineering office, the project engineer creates a device proxy in a new project and initializes it with the CPU data from the IPE file. The project engineer repeats the initialization for each update of the IPE file.

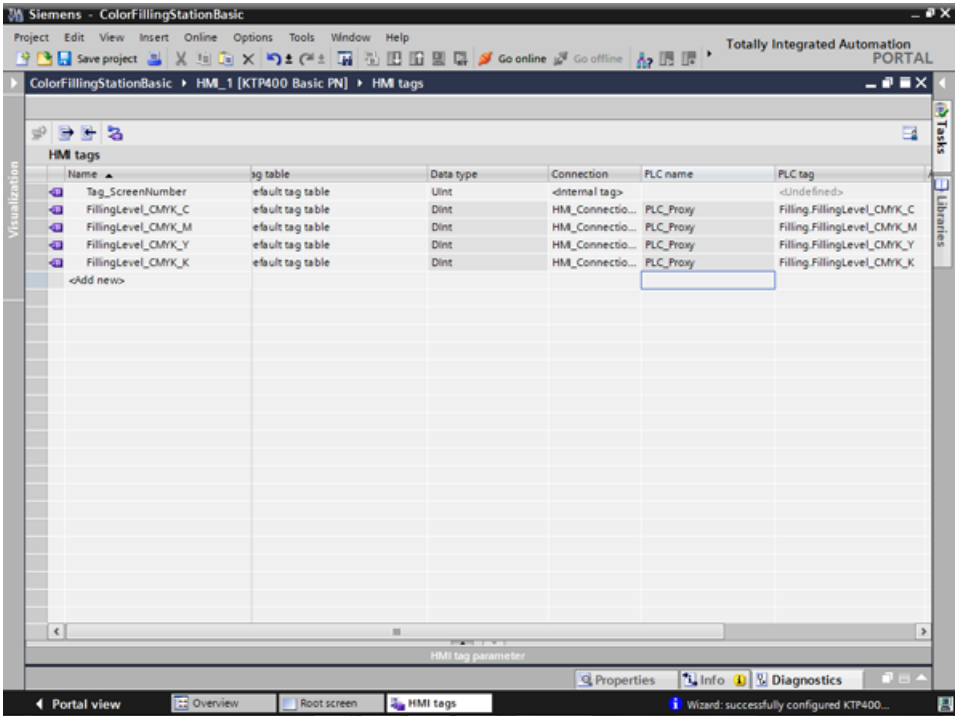
#### Procedure

1. Create the device proxy for the CPU in a new project and initialize it.



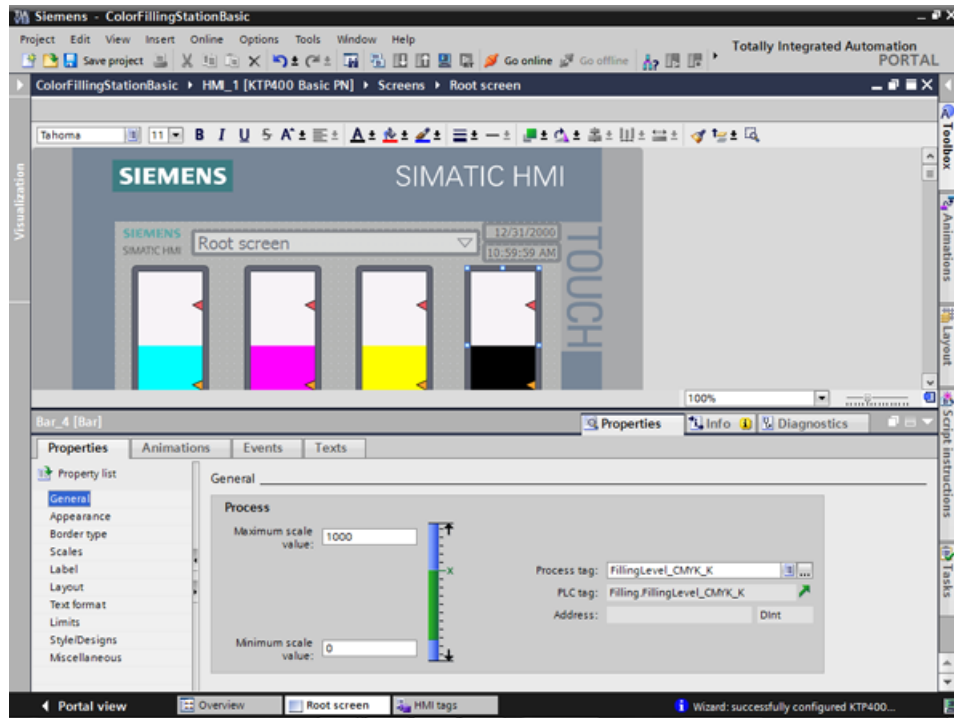
2. Use the Device Wizard to insert a Basic Panel.
3. Create the HMI tag for the fill level of the "Cyan" color and select the PLC tag.

4. Create the other HMI tags in the same way.



5. Configure a bar graph to display the fill level of the "Cyan" color.

- 6. Create a bar graph for the other fill levels in the same way.



- 7. Compile the project.

## Result

The project can now be loaded to the HMI device from the commissioning engineer. The communication with the CPU is up-to-date thanks to the CPU data from the IPE file.



# Security

## 4.1 Overview of the protective functions of the CPU

### Introduction

This chapter describes the following functions for protecting the S7-1500 automation system against unauthorized access:

- Access protection
- Know-how protection
- Copy protection
- Protection by locking the CPU

### Further measures for protecting the CPU

The following measures additionally increase the protection against unauthorized accesses to functions and data of the S7-1500 CPU from external sources and via the network:

- Deactivation of the Web server
- Deactivation of the time synchronization via an NTP Server
- Deactivation of the PUT/GET communication

When the Web server is used, you protect your S7-1500 automation system against unauthorized access by setting password-protected access rights for specific users in the user management.

## 4.2 Using the display to configure additional access protection

### Introduction

On the display of an S7-1500, you can block access to a password-protected CPU (local lock). The access lock is only in effect, when the operating mode switch is in the RUN position.

The access lock applies independently of password protection, i.e. if someone accesses the CPU via a connected programming device and has entered the correct password, access to the CPU is still blocked.

The access block can be set separately for each access level on the display, so that, for example, read access is allowed locally, but write access is not allowed locally.

### Procedure

If an access level with a password is configured in STEP 7, access can be blocked using the display.

Proceed as follows to set the local access protection for an S7-1500 CPU on the display:

1. On the display, select Settings > Protection menu.
2. Confirm the selection using "OK", and configure for each access level, whether access at the RUN mode selector is allowed or not:

Allow: Access to the CPU is possible, provided the corresponding password in STEP 7 is entered.

Deactivated in RUN: When the operating mode switch is in the RUN position, no more users with privileges for this access level can log in to the CPU, even if they know the password. In STOP mode, access is possible with password entry.

### Access protection for the display

A password can be configured for the display in STEP 7 in the properties of the CPU so that the local access protection is protected by a local password.

## 4.3 Know-how protection

You can use know-how protection to protect one or more blocks of the OB, FB, FC type and global data blocks in your program from unauthorized access. You can enter a password in order to restrict access to a block. The password protection prevents the block from being read or changed without authorization.

Without the password only the following data concerning the block can be read:

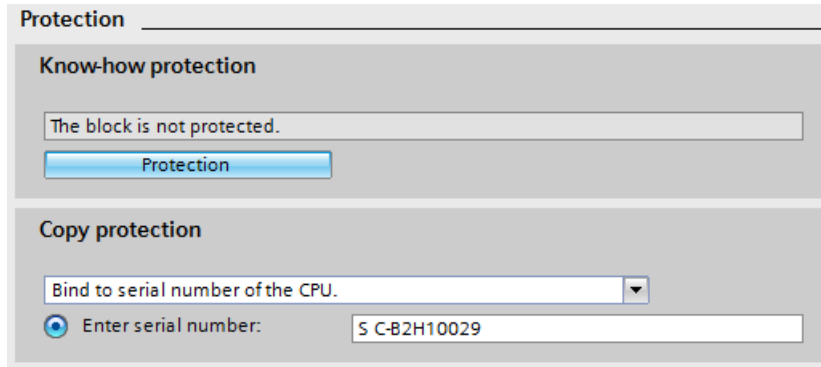
- Block title, comments and block properties
- Block parameters (INPUT, OUTPUT, IN, OUT, RETURN)
- Call structure of the program
- Global tags without information on the point of use

Further actions that can be carried out with a know-how protected block:

- Copying and deleting
- Calling in a program
- Online/offline comparison
- Load

### Setting up block know-how protection

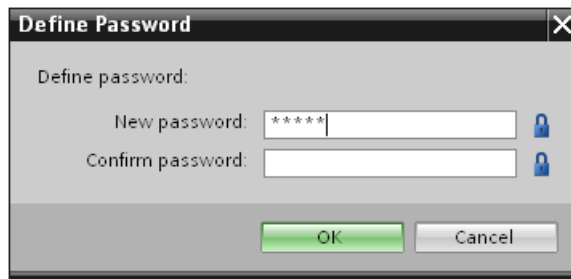
1. Open the properties of the respective block.
2. Select the "Protection" option under "General".



3. Click the "Protection" button to display the "Know-how protection" dialog.



4. Click the "Define" button to open the "Define password" dialog.



5. Enter the new password in the "New password" field. Enter the same password in the "Confirm password" field.
6. Click "OK" to confirm your entry.
7. Close the "Know-how protection" dialog by clicking "OK".

Result: The blocks selected will be know-how-protected. Know-how protected blocks are marked with a lock in the project tree. The password entered applies to all blocks selected.

## Opening know-how protected blocks

1. Double-click the block to open the "Access protection" dialog.
2. Enter the password for the know-how protected block.
3. Click "OK" to confirm your entry.

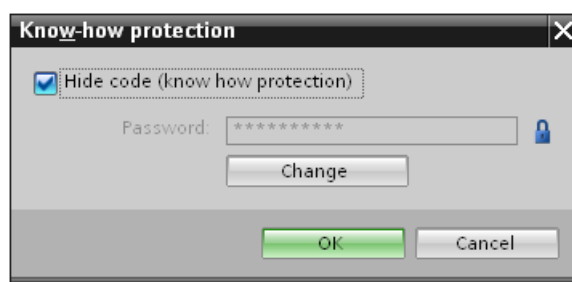
Result: The know-how-protected block will open.

Once you have opened the block, you can edit the program code and the block interface of the block for as long as the block or TIA Portal is open. The password must be entered again the next time the block is opened. If you close the "Access protection" dialog with "Cancel", the block will open but the block code will not be displayed and you will not be able to edit the block.

The know-how protection of the block is not removed if, for example, you copy the block or add it to a library. The copies will also be know-how-protected.

## Removing block know-how protection

1. Select the block from which you want to remove know-how protection. The protected block may not be open in the program editor.
2. In the "Edit" menu, select the "Know-how protection" command to open the "Know-how protection" dialog.
3. Deactivate the "Hide code (Know-how protection)" check box.



4. Enter the password.



5. Click "OK" to confirm your entry.

Result: Know-how protection will be removed from the block selected.

## 4.4 Copy protection

Copy protection allows you to bind the program or the blocks to a specific SIMATIC memory card or CPU. Through the linking of the serial number of a SIMATIC memory card or of a CPU the use of this program or of this block is only possible in combination with a specific SIMATIC memory card or CPU. With this function a program or block can be sent electronically (e.g. by e-mail) or by shipping a memory module.

When you set up such a copy protection for a block, also assign know-how-protection to this block. Without know-how protection, anyone can reset the copy protection. You must, however, set up copy protection first as the copy protection settings are read-only if the block is already know-how-protected.

### Setting up copy protection

1. Open the properties of the respective block.
2. Select the "Protection" option under "General".

3. In the "Copy protection" area, select either the "Bind to serial number of the CPU" entry or the "Bind to serial number of the memory card" entry from the drop-down list.

4. Enter the serial number of the CPU or the SIMATIC memory card.

5. You can now set up the know-how protection for the block in the "Know-how protection" area.

---

#### Note

If you download a copy protected block to a device that does not match the specified serial number, the entire download operation will be rejected. This means that blocks without copy protection will also not be downloaded.

---

## Removing copy protection

1. Remove any existing know-how protection.
2. Open the properties of the respective block.
3. Select the "Protection" option under "General".
4. In the "Copy protection" area, select the "No binding" entry from the drop-down list.

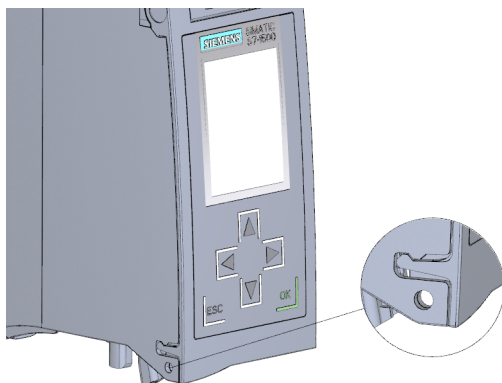


## 4.5 Protection by locking the CPU

Protect your CPU from unauthorized access using a sufficiently secured front cover.

Using the latch on the CPU cover, you have the following options:

- Affix a seal
- Secure the front cover with a lock (shackle diameter: 3 mm)



## 4.6 Configuring access protection for the CPU

### Introduction

The CPU offers four access levels, in order to limit access to specific functions.

By setting up the access levels and the passwords for a CPU, you limit the functions and memory areas that are accessible without entering a password. The individual access levels as well as the entry of their associated passwords are specified in the object properties of the CPU.

**Access levels of the CPU**

Access levels	Access restrictions
Complete access (no protection)	The hardware configuration and the blocks can be read and changed by all users.
Read access	With this access level, read-only access to the hardware configuration and the blocks is possible without entering a password, which means you can download hardware configuration and blocks to the programming device. HMI access and access to diagnostics data is also possible.  Without entering the password, you cannot load any blocks or hardware configuration into the CPU. Additionally, the following are <b>not</b> possible without the password: Test functions which write, changing the operating mode (RUN/STOP), and firmware update (online).
HMI access	With this access level only HMI access and access to diagnostics data is possible without entering the password.  Without entering the password, you can neither load blocks and hardware configuration into the CPU, nor load blocks and hardware configuration from the CPU into the programming device. Additionally, the following are <b>not</b> possible without the password: Test functions which write, changing the operating mode (RUN/STOP), and firmware update (online).
No access (complete protection)	When the CPU is completely protected, no read or write access to the hardware configuration and the blocks is possible. HMI access is also not possible. The server function for PUT/GET communication is disabled in this access level (cannot be changed).  Authentication with the password will again provide you full access to the CPU.

Each access level allows unrestricted access to certain functions without entering a password, e.g. identification using the "Accessible devices" function.

The CPU's default setting is "No restriction" and "No password protection". In order to protect access to a CPU, you must edit the properties of the CPU and set up a password.

Communication between the CPUs (via the communication functions in the blocks) is not restricted by the protection level of the CPU, unless PUT/GET communication is deactivated.

Entry of the right password allows access to all the functions that are allowed in the corresponding level.

**Note**

**Configuring an access level does not replace know-how protection**

Configuring access levels prevents unauthorized changes to the CPU, by restricting download privileges. However, blocks on the SIMATIC memory card are not write- or read-protected. Use know-how protection to protect the code of blocks on the SIMATIC memory card.



## Parameterizing the procedure at access levels

To configure the access levels of an S7-1500 CPU, follow these steps:

1. Open the properties of the S7-1500 CPU in the Inspector window.
2. Open the "Protection" entry in the area navigation.

A table with the possible access levels appears in the Inspector window.

Protection level	Access			Access permission	
	HMI	Read	Write	Password	Confirmation
<input checked="" type="radio"/> Full access (no protection)	✓	✓	✓		
<input type="radio"/> Read access	✓	✓			
<input type="radio"/> HMI access	✓				
<input type="radio"/> No access (complete protection)					

3. Activate the desired protection level in the first column of the table. The green checkmarks in the columns to the right of the respective access level show you which operations are still available without entering the password.
4. In the "Password" column, specify a password for the selected access level. In the "Confirmation" column, enter the selected password again to protect against incorrect entries.

Ensure that the password is sufficiently secure, in other words, that it does not follow a pattern that can be recognized by a machine!

You must enter a password in the first row ("Full access" access level). This enables unrestricted access to the CPU for those who know the password, regardless of the selected protection level.

5. Assign additional passwords as needed to other access levels if the selected access level allows you to do so.
6. Download the hardware configuration to the CPU, so that the access level will take effect.

## Behavior of a password-protected CPU during operation

The CPU protection takes effect after the settings are downloaded in the CPU.

Before an online function is executed, the necessary permission is checked and, if necessary, the user is prompted to enter a password. The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on.

Access authorization to the protected data is in effect for the duration of the online connection, or until the access authorization is manually rescinded with "Online > Delete access rights".

Access to a password-protected CPU in the RUN mode can be limited locally in the display so that access with a password is also not possible.

## 4.7 Configuring protection of the HMI connection

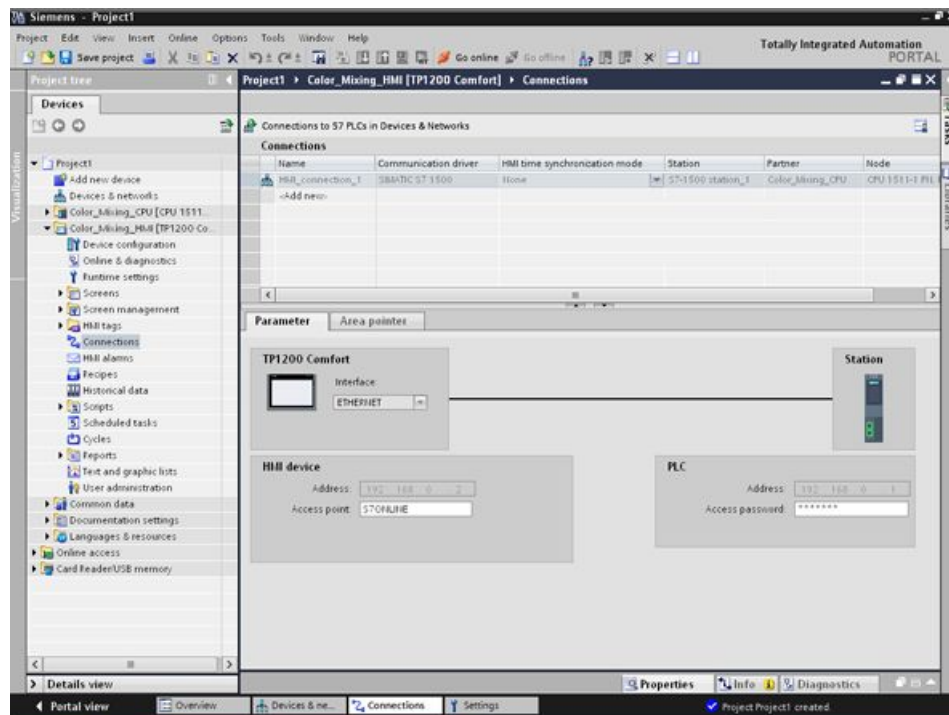
### Introduction

If the protection level "Complete protection" was set for the CPU, the HMI device can only access the CPU with the password stored there.

This function is only available with HMI devices from SIEMENS.

### Procedure

1. Open the "Connections" editor in the project tree.
2. Select the integrated connection.
3. Enter the password for the CPU in the "Password" area.



### Result

The HMI device can now communicate and exchange data with the CPU.