fischertechnik

LEARNING FACTORY 4.0 24V

Accompanying booklet

Status: 22.03.2024

Programming tasks

To solve the programming tasks, you need previous knowledge of the Structured Text programming language (ST or SCL) and how to use the programming tool you are using.

The programming solutions provided have been created with the TIA Portal V16 for a SIMATIC S7-1500 programmable logic controller with CPU1512SP.
Alternatively, other control systems with the appropriate software can also be used here.

You can find an introduction to programming with the TIA Portal here at SIEMENS, for example:

http://www.siemens.de/sce/S7-1500

https://new.siemens.com/global/de/unternehmen/nachhaltigkeit/ausbildung/sce/lern-lehr-unterlagen/erweiterte-programmierung.html

Further information can be found at:
https://github.com/fischertechnik

The global fischertechnik library **Library_LearningFactory_4_0_24V** and the
You can find program solutions under:
https://github.com/fischertechnik/plc_training_factory_24v

Programming task 1:
Configuration and commissioning of the SIMATIC controller with CPU 1512SP

Task definition:
For the LearningFactory_4_0_24V, the complete hardware configuration of the controller with all input/output modules is to be created, loaded and tested.
An unspecified CPU is created in order to determine all components.

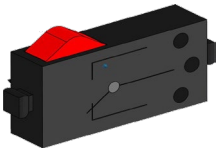
This project with the hardware configuration serves as the basis for all subsequent programming tasks.

Planning
1. wire a START button as a normally open contact (NO) and a STOP button as a normally closed contact (NC) to two free inputs on the control unit. Connect the LearningFactory_4_0_24V to the PLC and switch on the control unit.
2. create the **LearningFactory_4_0_24V_Task01_HW Conf** project in the TIA Portal.
3. add a **SIMATIC S7-1500** controller as a new device in the TIA Portal project **LearningFactory_4_0_24V_Task01_HW Conf.** This is done as described in the following programming instructions as an unspecified CPU.
4. connect the programming device to the CPU and have the configuration of the connected device determined according to the programming instructions.
5. Set the input/output addresses of the detected modules according to the programming instructions and configure the modules.
6. dearchive the global library **Library_LearningFactory_4_0_24V** as shown in the programming instructions and insert the global variable table **Start-Stop** into your project from there.
7. load the hardware configuration with the variable table into the control unit and test it.
8. Create an observation table with the name **Beobachtungstabelle_watch table**, insert the signals from the global variable table and check the function of the buttons by observing them.
9. save and archive the **LearningFactory_4_0_24V_Task01_HW Conf** project.

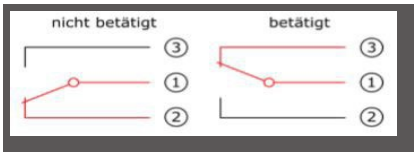
Programming instructions:

Notes on connecting the START and STOP buttons:

Two buttons are required for the following programming tasks. One START push-button as a normally open contact (NormallyOpen) and one STOP push-button as a normally closed contact (NormallyClosed). The Fischertechnik mini pushbuttons can be used for this.

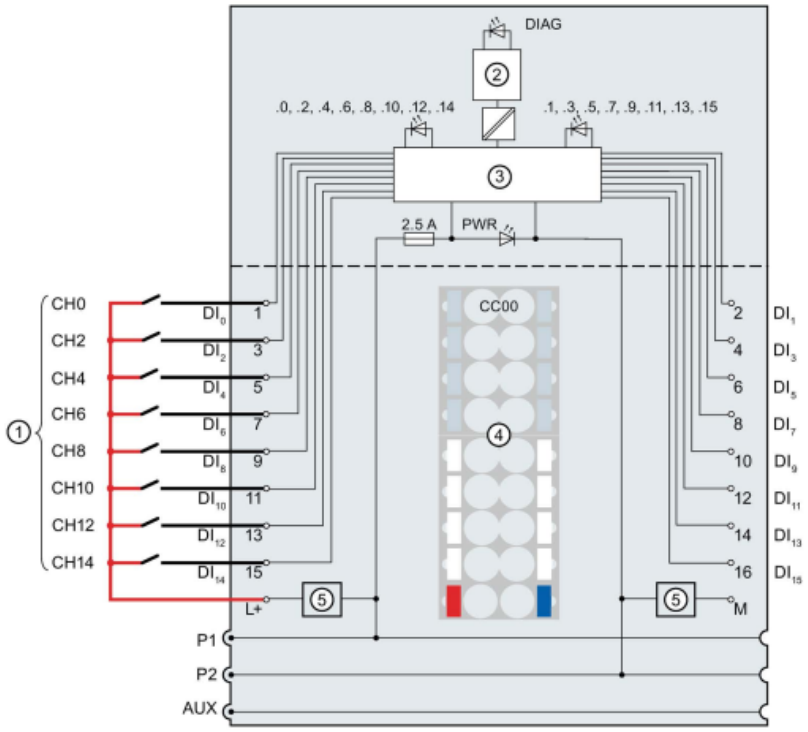


The mini push-buttons are equipped with changeover contacts and can be used both as normally closed (NC) and normally open (NO) contacts. The illustration shows the schematic circuit diagram of the mini push-buttons.



For use as a normally closed contact, the two contacts 1 and 2 are wired to a digital input of the PLC.
For use as a normally open contact (NormallyOpen), the two contacts 1 and 3 are wired to a digital input of the PLC.

Details on the connection to the SIMATIC modules can be found in the manuals.
Here is an example of the pin assignment of the digital input module DI 16x24VDC ST used here on the BaseUnit BU type A0 without AUX terminals (1-wire connection):



Notes on configuring an unspecified CPU 1512SP:

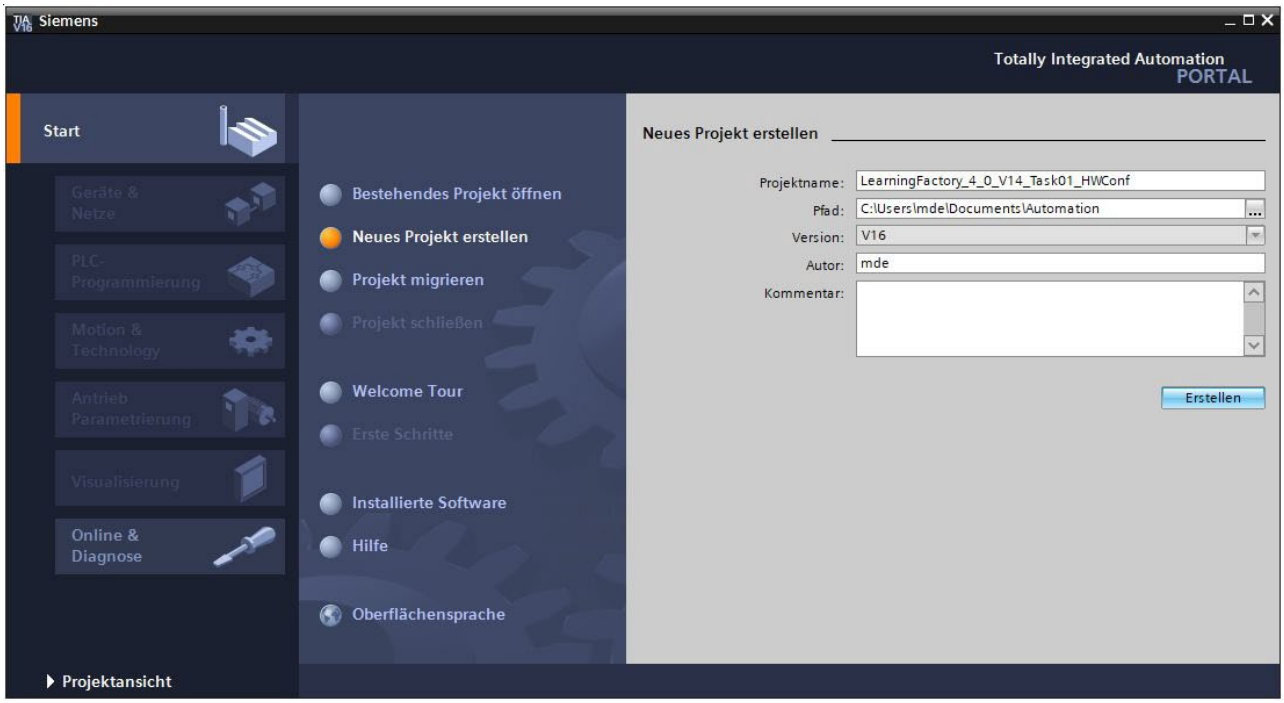
Before a SIMATIC S7-1500 PLC can be programmed, its hardware must be configured.
The quickest way to do this is to create an unspecified CPU and then have all components recognized.
The prerequisite is that the programming device is connected to the control unit via Ethernet and the IP addresses are set so that both participants are in the same subnet.
The setting of the IP address is described in this document in the chapter **Commissioning and adapting the SIMATIC CPU 1512SP controller.**

The configuration for our Learning Factory 4.0 can be carried out in the following steps.

→ First start the Totally Integrated Automation Portal, which is opened here with a double-click.

(→ TIA Portal V18)

→ In the portal view, go to → **Start** → **Create new project**. Adjust the project name, path, author and comment accordingly and click on → **Create**

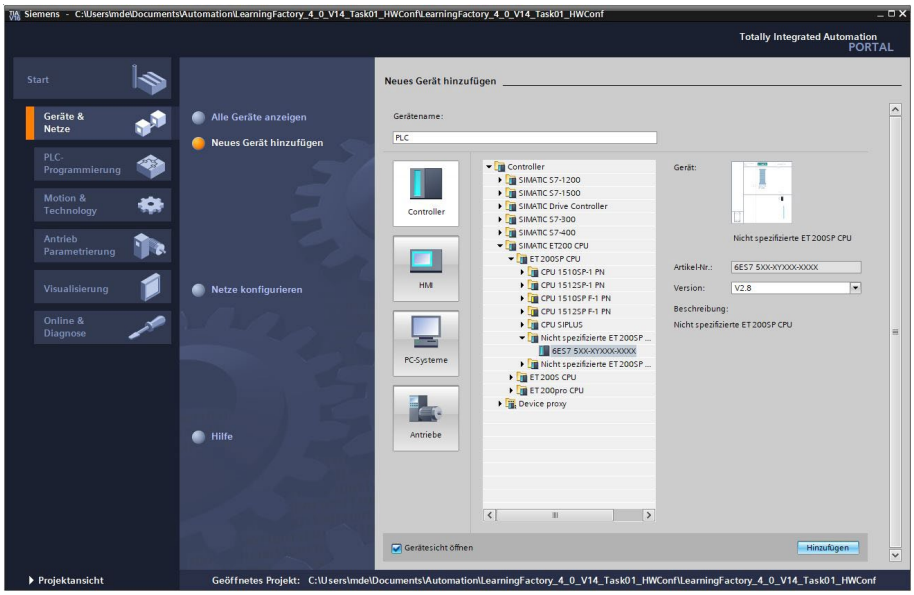


→ In the portal, select → **Start** → **Devices & networks** → **Add new device**.

Create a new CPU. Use an unspecified model of the **SIMATIC ET200 CPU** with the order number **6ES7 5XX-XXXXX-XXXX**.

(Controller → SIMATIC ET200 CPU → ET 200SPCPU → Unspecified ST200SP CPU → 6ES7 5XX-XXXXX-XXXX → V2.8)

Assign a device name (Device name → **PLC**), select **Open device view** and then click on → **Add**.

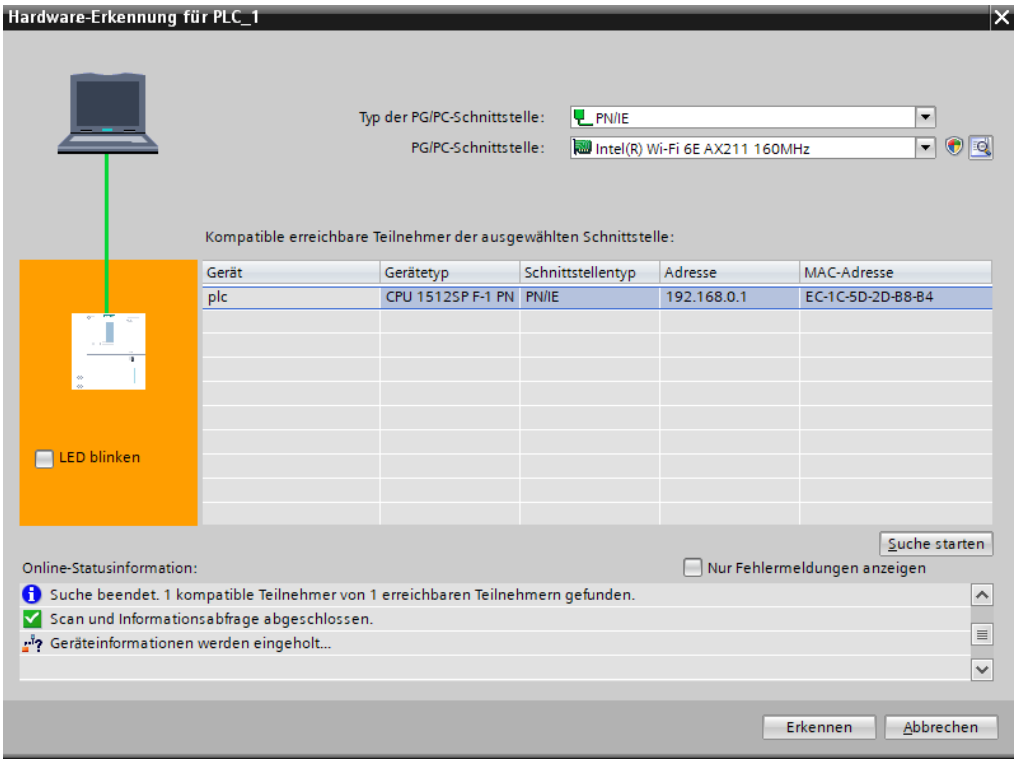


→ The TIA Portal now automatically switches to the project view and displays a note that this device is not specified. To have the hardware configuration detected automatically, start the detection by clicking on **detect** in the yellow info box. (→ detect)

→ First select the type of your PG/PC interface and the network card with which you want to establish a connection to the PLC via Ethernet. (→ **Type of PG/PC interface: PN/IE** → **PG/PC interface: Intel(R) Ethernet ...**)

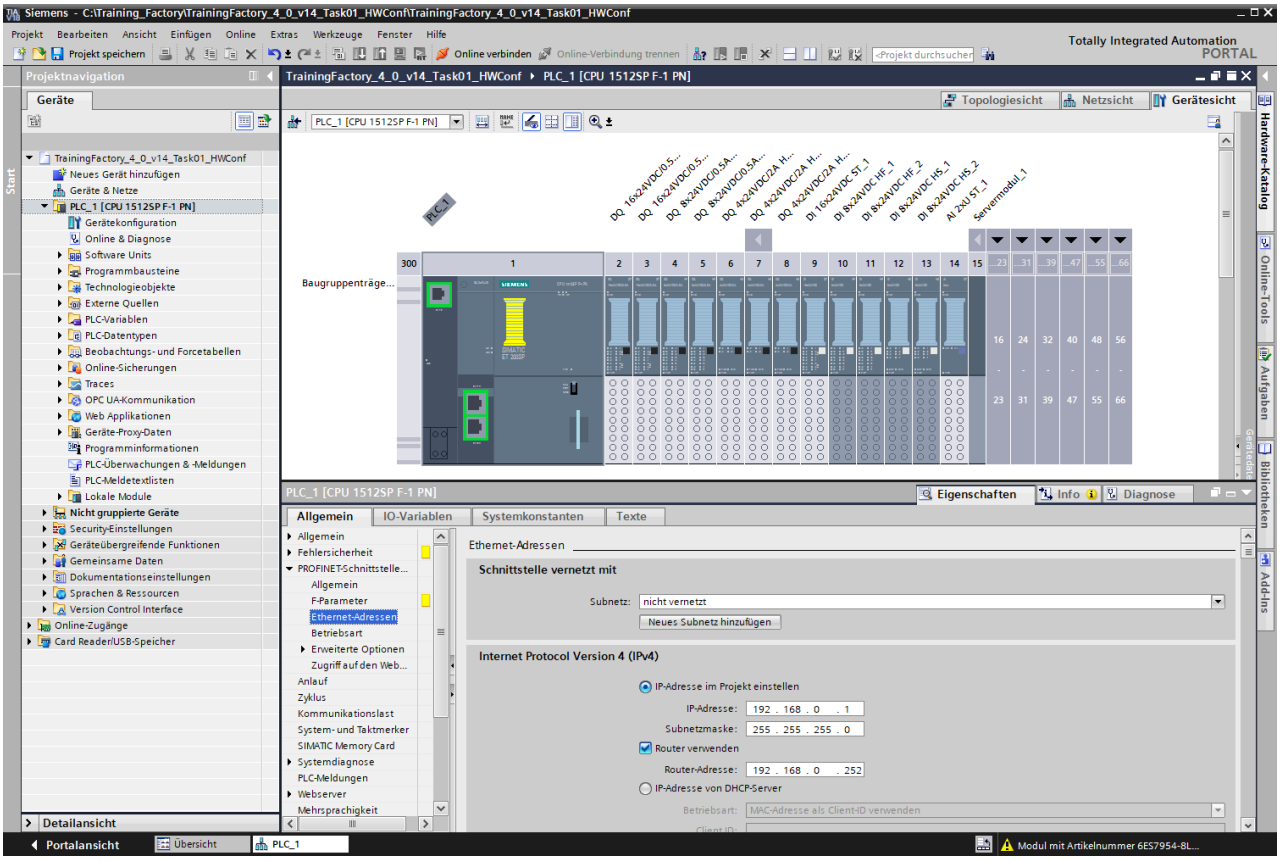
Now the search for participants in the network must be started by clicking on the → **Start search** button.

All accessible devices are then searched for and listed. Once you have selected the correct CPU, clicking on → **Detect** causes the corresponding CPU and connected modules to be recognized.



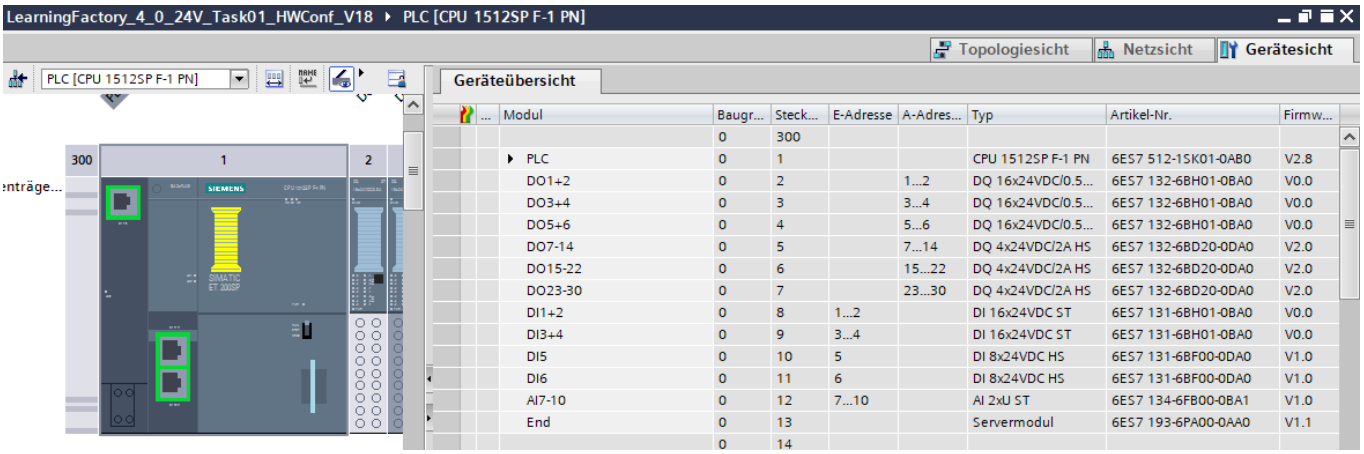
Note: If your CPU is not included in the list, make sure that you have selected the correct network card and that you have established a connection between the laptop and CPU.

→ The TIA Portal now shows the complete device configuration of the selected CPU. Here you can now configure the CPU and the input/output modules according to your specifications.



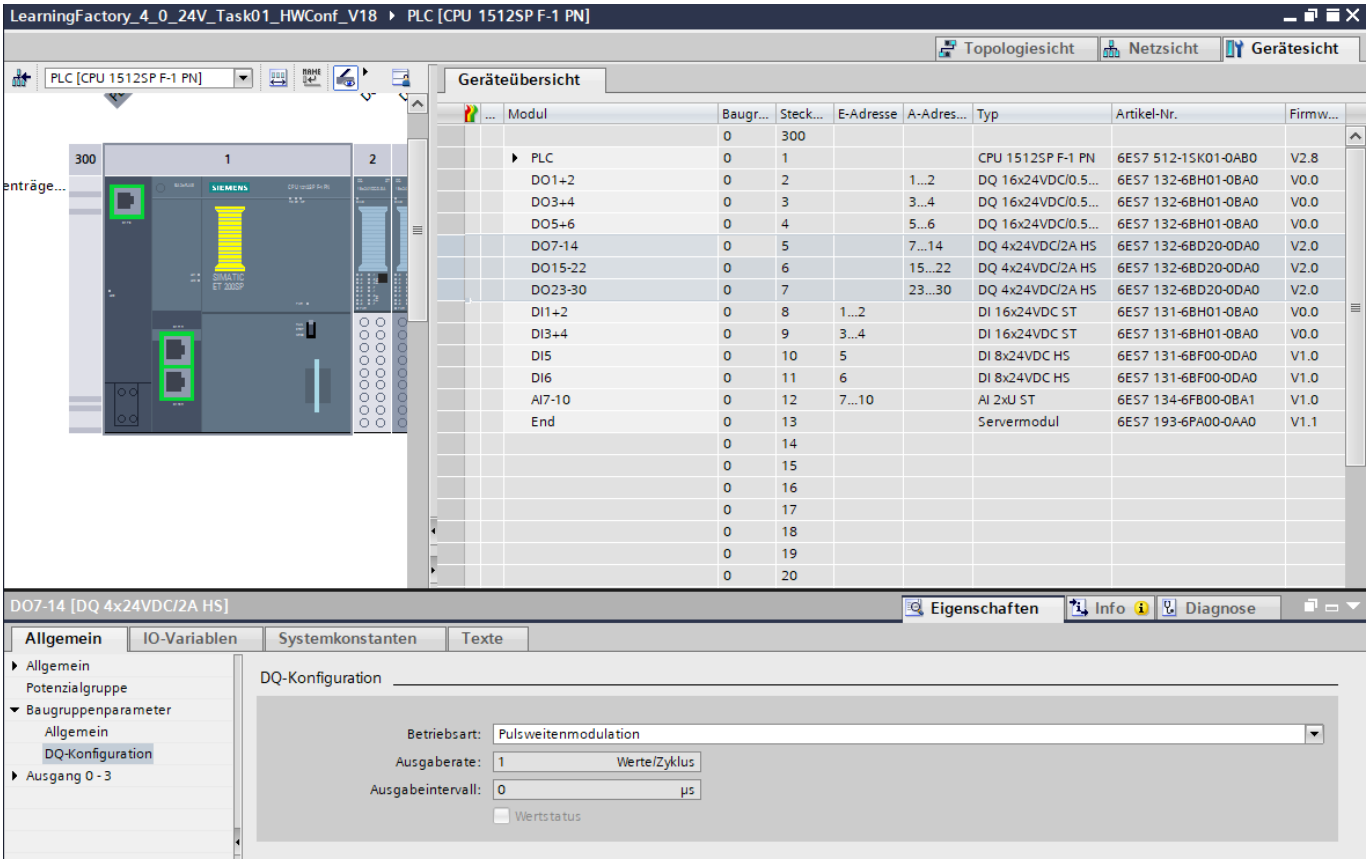
Notes on configuring the input/output modules

The **addresses of the modules** should be set as shown here.



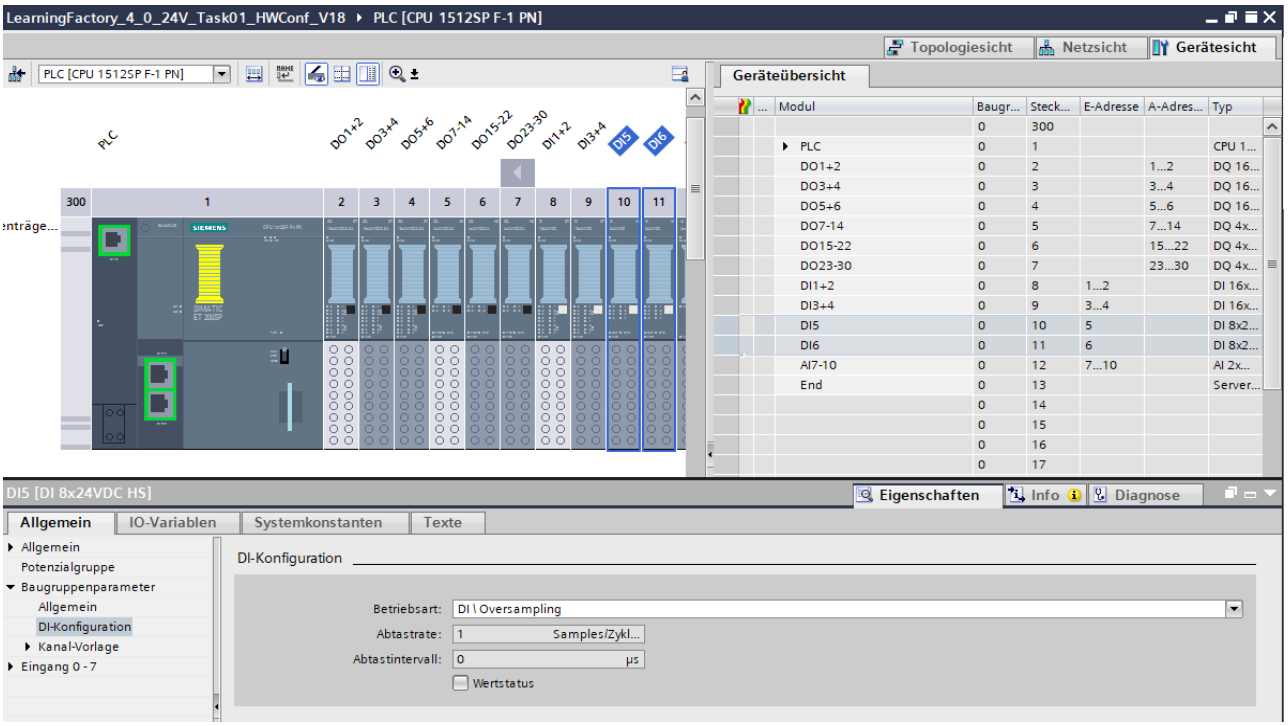
	Modul	Baugr...	Steck...	E-Adresse	A-Adres...	Typ	Artikel-Nr.	Firmw...
		0	300					
	PLC	0	1			CPU 1512SP F-1 PN	6ES7 512-1SK01-0AB0	V2.8
	DO1+2	0	2		1...2	DQ 16x24VDC/0.5...	6ES7 132-6BH01-0BA0	V0.0
	DO3+4	0	3		3...4	DQ 16x24VDC/0.5...	6ES7 132-6BH01-0BA0	V0.0
	DO5+6	0	4		5...6	DQ 16x24VDC/0.5...	6ES7 132-6BH01-0BA0	V0.0
	DO7-14	0	5		7...14	DQ 4x24VDC/2A HS	6ES7 132-6BD20-0DA0	V2.0
	DO15-22	0	6		15...22	DQ 4x24VDC/2A HS	6ES7 132-6BD20-0DA0	V2.0
	DO23-30	0	7		23...30	DQ 4x24VDC/2A HS	6ES7 132-6BD20-0DA0	V2.0
	DI1+2	0	8	1...2		DI 16x24VDC ST	6ES7 131-6BH01-0BA0	V0.0
	DI3+4	0	9	3...4		DI 16x24VDC ST	6ES7 131-6BH01-0BA0	V0.0
	DI5	0	10	5		DI 8x24VDC HS	6ES7 131-6BF00-0DA0	V1.0
	DI6	0	11	6		DI 8x24VDC HS	6ES7 131-6BF00-0DA0	V1.0
	AI7-10	0	12	7...10		AI 2xU ST	6ES7 134-6FB00-0BA1	V1.0
	End	0	13			Servermodul	6ES7 193-6PA00-0AA0	V1.1
		0	14					

For the output modules with addresses DO 7 - DO 30, **pulse width modulation** (PWM) should be set for the **operating mode.**

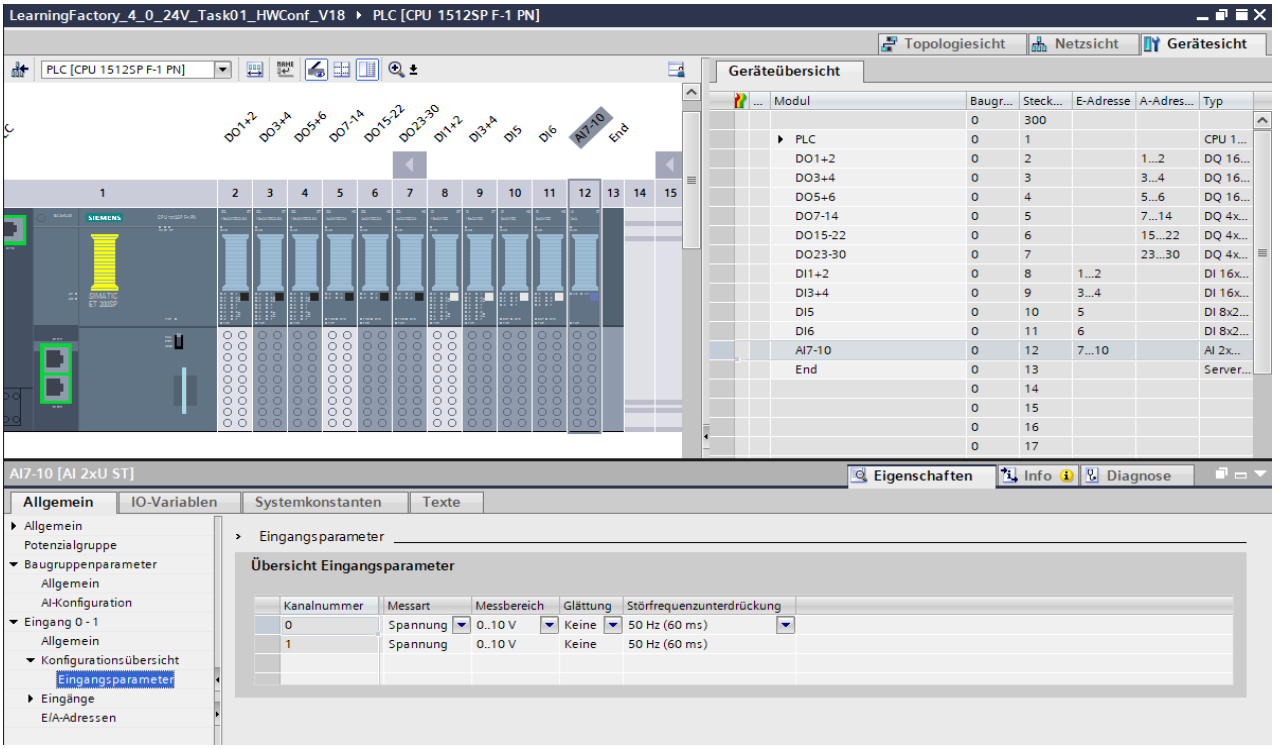


	Modul	Baugr...	Steck...	E-Adresse	A-Adres...	Typ	Artikel-Nr.	Firmw...
		0	300					
	PLC	0	1			CPU 1512SP F-1 PN	6ES7 512-1SK01-0AB0	V2.8
	DO1+2	0	2		1...2	DQ 16x24VDC/0.5...	6ES7 132-6BH01-0BA0	V0.0
	DO3+4	0	3		3...4	DQ 16x24VDC/0.5...	6ES7 132-6BH01-0BA0	V0.0
	DO5+6	0	4		5...6	DQ 16x24VDC/0.5...	6ES7 132-6BH01-0BA0	V0.0
	DO7-14	0	5		7...14	DQ 4x24VDC/2A HS	6ES7 132-6BD20-0DA0	V2.0
	DO15-22	0	6		15...22	DQ 4x24VDC/2A HS	6ES7 132-6BD20-0DA0	V2.0
	DO23-30	0	7		23...30	DQ 4x24VDC/2A HS	6ES7 132-6BD20-0DA0	V2.0
	DI1+2	0	8	1...2		DI 16x24VDC ST	6ES7 131-6BH01-0BA0	V0.0
	DI3+4	0	9	3...4		DI 16x24VDC ST	6ES7 131-6BH01-0BA0	V0.0
	DI5	0	10	5		DI 8x24VDC HS	6ES7 131-6BF00-0DA0	V1.0
	DI6	0	11	6		DI 8x24VDC HS	6ES7 131-6BF00-0DA0	V1.0
	AI7-10	0	12	7...10		AI 2xU ST	6ES7 134-6FB00-0BA1	V1.0
	End	0	13			Servermodul	6ES7 193-6PA00-0AA0	V1.1
		0	14					
		0	15					
		0	16					
		0	17					
		0	18					
		0	19					
		0	20					

For the input modules with the addresses DI 5 to DI 6 for the encoder inputs, **oversampling** should be set in the **DI \ operating mode.**



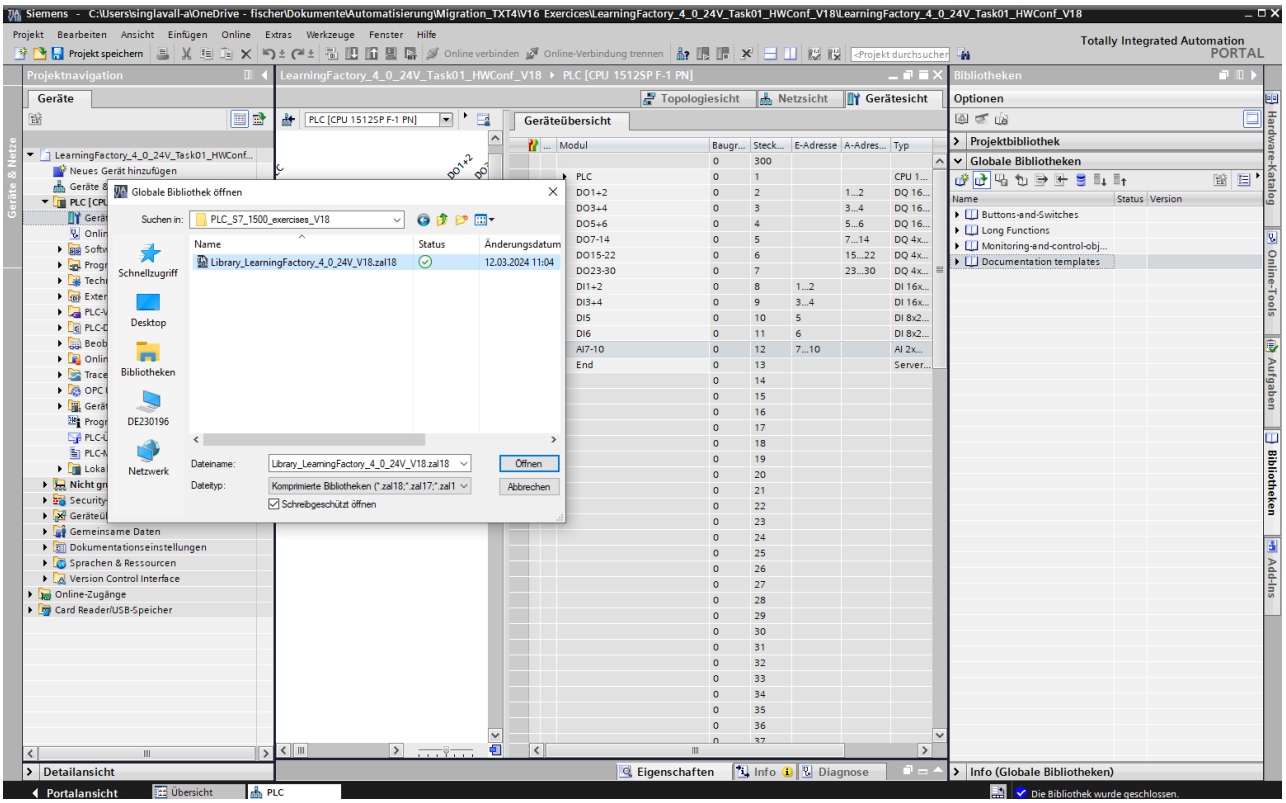
For the channels of the analog input modules, the **voltage measuring type** should be set with the **measuring range 0-10V.**



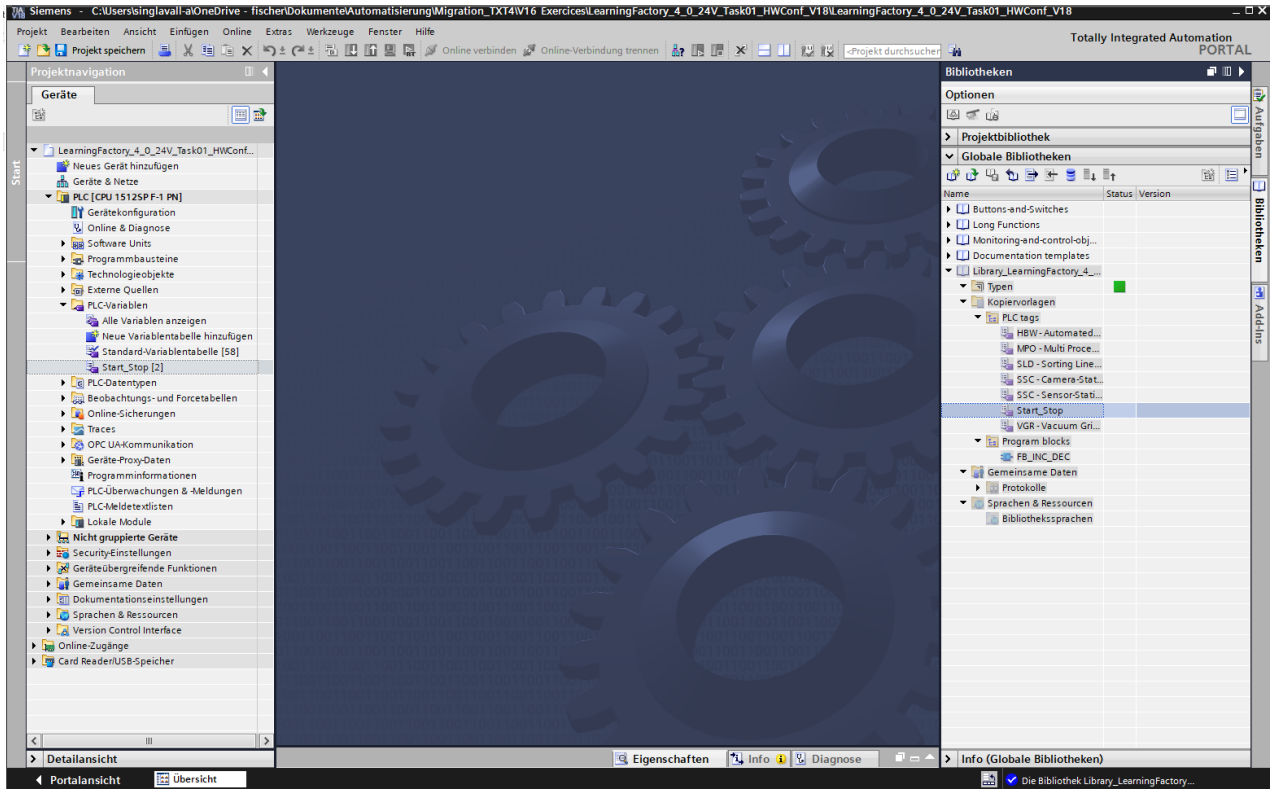
Notes on global libraries:

For the programming tasks, the global variable tables and also a block for encoder evaluation are provided in a compressed library **Library_LearningFactory_4_0_24V.zal18.**

This can be opened by clicking on the icon under **Global libraries** to open a library and selecting the **Library_LearningFactory_4_0_24V.zal18** file.

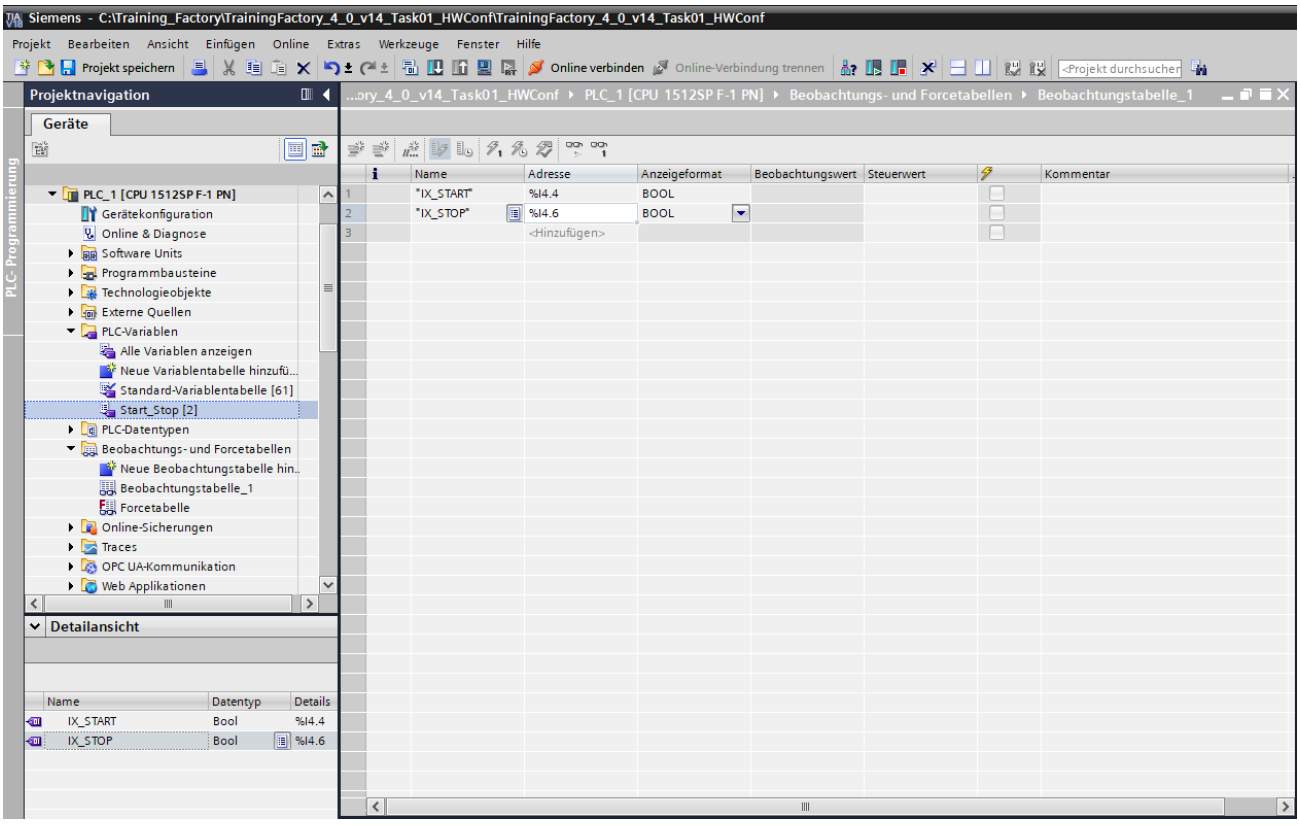


Elements from the library can then simply be copied to the appropriate folders in the project using drag & drop.

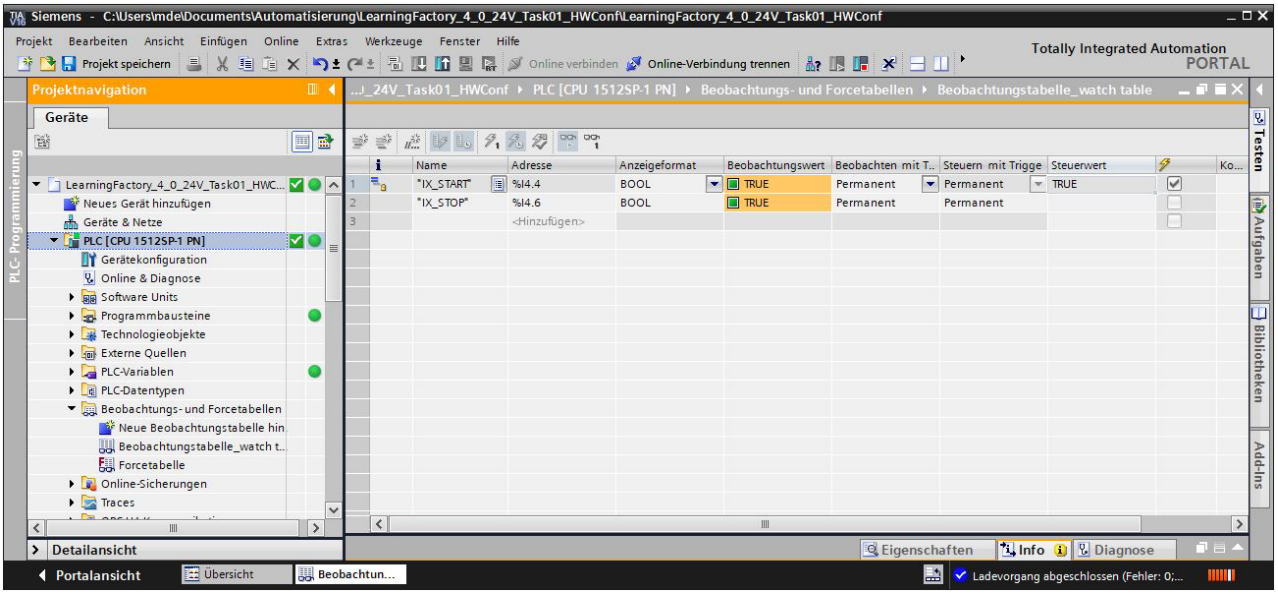


Notes on observation tables:

Observation tables can be created to test the function of the factory modules and to determine the position values later. Variables can then simply be dragged and dropped from the detailed view of the variable tables into the observation **tables.**



In the extended view  , outputs and inputs used in the program can also be permanently controlled  . If there are no buttons for the START and STOP functions, these can be simulated in this way.



Programming task 2:
Sensor station with camera (SSC), color detection with color sensor

Task definition:

In the sensor station with camera (SSC), the color sensor on the input/output station is used to detect the different workpieces blue/white/red.

If a workpiece is on the color sensor, the measured value of the color sensor is evaluated and the result is displayed on an assigned LED. The evaluation only takes place as long as the START button is pressed.

Color assignment:

Blue	LED green (Q5)
White	LED yellow (Q6)
Red	LED red (Q7)
No color detectedLED	red Online status (Q8)

The program should be created in the Structured Text programming language (ST or SCL).

Planning

1. Open the **LearningFactory_4_0_24V_Task01_HW Conf** project in the TIA Portal and save it under the name **LearningFactory_4_0_24V_Task02_SensorsSSC.**
2. Copy the global variable table **SSC-Sensor-Station**, as shown in programming task 1, from the library **Library_LearningFactory_4_0_24V** and insert it into your project.
3. expand the observation table as shown in programming task 1, determine the value ranges for the 3 colors blue/white/red by placing the different workpieces there and check the function of the buttons and sensors.
4. Create library-compatible code block in the Structured Text programming language (ST or SCL) .
5. create a new organization block Main [OB1] in the programming language Structured Text (ST or SCL), call up the library-capable code block there and connect it to the global variables.
6. Load the complete program into the control unit.
7. Test the program on the sensor station with camera (SSC).
8. Monitor the program online and eliminate any errors.
9. save and archive the **LearningFactory_4_0_24V_Task02_SensorsSSC** project.

Instructions for creating library-compatible program modules:

For the structured programming of PLC programs, library-capable code blocks should preferably be created. This means that the input and output parameters of a function or a function block are generally defined in the local variables and are only provided with the current global variables (inputs/outputs) when the block is used.
This has the advantage that the module can be called up several times.

Notes on the color sensor

The fischertechnik color sensor emits white light that is reflected to different degrees by different colored surfaces. The intensity of the reflected light is measured via the phototransistor and read in as a voltage value between 0 V and 9 V at an analog input 0 - 10 V of the PLC. The analogue input module digitizes this measured value (0 - 9 V) into an integer between 0 and 24883.

The measured value depends on the ambient brightness and the distance between the sensor and the colored surface.

Therefore, the values for the 3 different workpieces in the colors blue/white/red should be determined in several measurements.
An average value can be calculated from this for each color. A tolerance range around this mean value should be defined during the evaluation, as the measurement results always have a certain degree of inaccuracy.

Programming task 3:
Sensor station with camera (SSC), camera movement without pulse width modulation (PWM)

Task definition:

In the sensor station with camera (SSC), the camera movement should be controlled directly via the outputs without pulse width modulation (PWM).

Press the START button to start a reference run. The camera moves horizontally clockwise to the position of the reference switch and vertically downwards to the reference position of the vertical axis.
By pressing the STOP button, the camera should move horizontally counterclockwise and vertically upwards for 2 seconds.
If START or STOP is pressed while the camera is already moving, the direction should be reversed.

The program should be created in the Structured Text programming language (ST or SCL).

Planning

1. open the **LearningFactory_4_0_24V_Task01_HWConf** project in the TIA Portal and save it under the name **LearningFactory_4_0_24V_Task03_CameraSSC.**
2. copy the global variable table **SSC-Camera-Station**, as shown in programming task 1, from the library **Library_LearningFactory_4_0_24V** and insert it into your project.
3. Insert the jumpers for the power supply of the bidirectionally controlled motors so that the motors are supplied directly via the +24V of the actuators.
4. expand the observation table as shown in programming task 1 and check the function of the I / O signals.
5. create library-capable code blocks in the Structured Text programming language (ST or SCL).
6. create a new organization block Main [OB1] in the programming language Structured Text (ST or SCL), call up the library-capable code block there and connect it to the global variables.
7. Load the complete program into the control unit.
8. test the program on the sensor station with camera (SSC).
9. monitor the program online and eliminate any errors.
10. save and archive the project **LearningFactory_4_0_24V_Task02_CameraSSC.**
11. Reconnect the jumpers for the power supply of the bidirectionally controlled motors so that they are supplied via the corresponding PWM terminals 2 and 3.

Programming instructions

Notes on pulse width modulation (PWM):

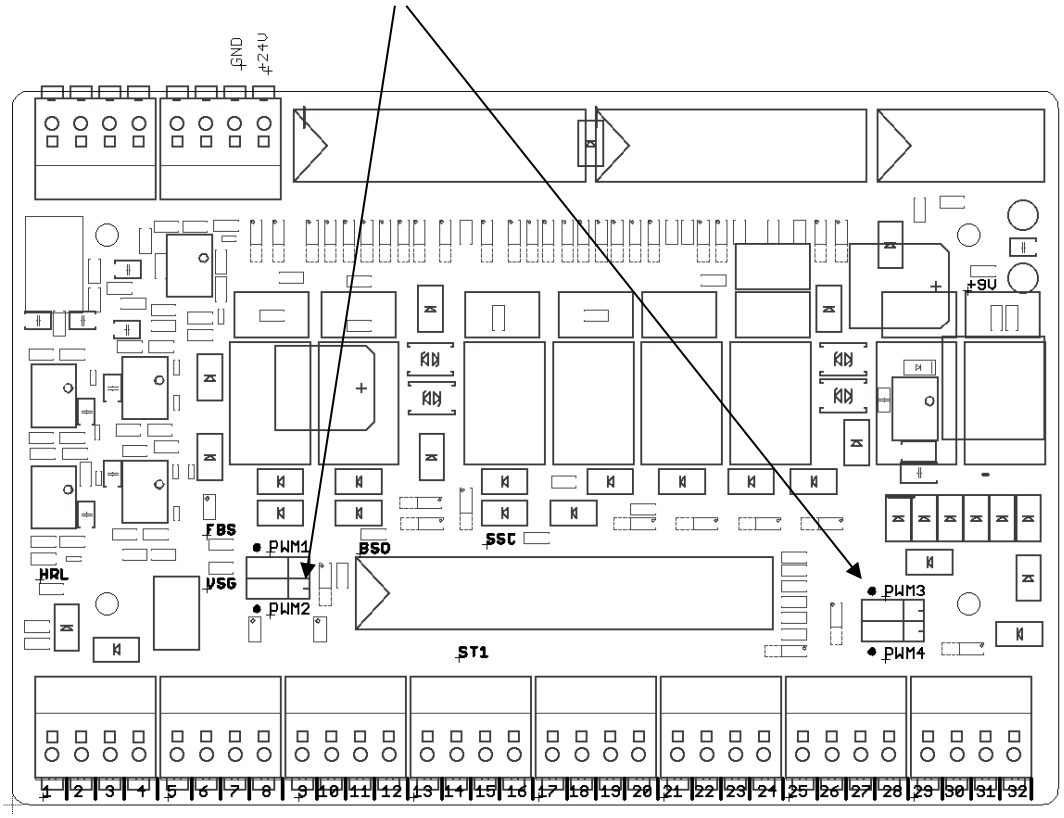
The bidirectionally controlled motors for the horizontal axis (camera rotation) and the vertical axis (camera height) are controlled via relays and can be supplied either directly via the 24V of the actuators or via the associated PWM terminals 2 and 3.

	Sensor station with camera (SSC)
PWM 1	- not used -
PWM 2	Camera height
PWM 3	Rotate camera
PWM 4	- not used -

Assignment of the PWM jumpers:

For this task, the jumpers on the right should be plugged in for the power supply via +24V (actuators).

The jumpers are located here on the adapter board



Notes on programming

As the positive signal edge of the START button is to be evaluated in this task and a timer is used, a function block (FB) must be programmed here as a code block.

Programming task 4:
Sorting line with color detection (SLD); step chain with simple positioning function without color detection

Task definition:

In the sorting line with color detection (SLD), workpieces are to be transported from the start of the belt to the 2nd storage location regardless of their color.

If a workpiece is on the belt in the light barrier at the input and the 2nd storage location is free, the belt is switched on with the START button. From the light barrier after the color sensor, the pulses from the pulse button are counted for position detection. The belt should be stopped as soon as the workpiece is in front of the 2nd ejector cylinder.
Here the workpiece is conveyed from the 2nd extension cylinder to the 2nd storage location.
The workpiece must then be removed by hand.

The process is stopped by pressing the STOP button. Workpieces must be removed from the system before the sorting line can be restarted.

The program should be created in the Structured Text programming language (ST or SCL).

Planning

1. Open the **LearningFactory_4_0_24V_Task01_HW Conf** project in the TIA Portal and save it under the name **LearningFactory_4_0_24V_Task04_SortingSLD.**
2. Copy the global variable table **SLD-Sorting Line with Detection,** as shown in programming task 1, from the library **Library_LearningFactory_4_0_24V** and insert it into your project.
3. expand the observation table as shown in programming task 1 and check the function of the I / O signals.
4. create library-capable code blocks with the step chain in the Structured Text programming language (ST or SCL).
5. create a new organization block Main [OB1] in the programming language Structured Text (ST or SCL), call up the library-capable code block there and connect it to the global variables.
6. Load the complete program into the control unit.
7. test the program on the sorting line with color detection (SLD).
8. Monitor the program online and eliminate any errors.
9. save and archive the **LearningFactory_4_0_24V_Task04_SortingSLD** project.

Programming instructions

Notes on step chain programming in the Structured Text programming language (ST or SCL):

In addition to other possibilities in the Structured Text programming language (ST or SCL) to program a sequence control or a step chain, this can be done, for example, in a case instruction.
The static variable #Step_number always contains the number of the step currently being executed.
Only the commands programmed for the corresponding step number are executed.
The transition to the next step is made by incrementing the #Step_number variable by +1 within the step.
The variable #Step_number is set to 0 for the transition from the last step to step 0 and for resetting the step chain.

```
CASE #Step_number OF
    0:  // Initial step
        ;
    1:  // Step 1
        ;
    2:  // Step 2
        ;

    //....
    ELSE  // Statement section ELSE
        ;
END_CASE;
```

As static variables are also required in the step chain, a function block (FB) must be programmed here as a code block.

Notes on positioning

As the conveyor belt does not move particularly quickly and the position in front of the ejector cylinder does not need to be particularly precise, it is sufficient to record the signals from the pulse probe in the normal cyclical organization block OB1 for this simple positioning function.
A simple counter is sufficient to count the pulses in the program, the value of which is then compared.
You can determine the number of pulses required by activating the conveyor belt motor and observing the counter value.

Programming task 5:
Vacuum gripper (VGR); stepper chain with pulse width modulation (PWM) and simple positioning function

Task definition:
The vacuum suction gripper (VGR) station is used to transport workpieces from the material storage area to the material waste storage area, regardless of their color.

The motors for the movements of the gripper are controlled via relays and supplied with voltage via the associated PWM terminals. The speed of the motors can be set via the pulse width modulation (PWM) output signals. The speed should be set rather low for this task so that positioning is easier.

The sequence is started with the START button. First the suction pad moves back horizontally to the reference switches, then vertically upwards and finally clockwise to the reference position at the material storage.
It then positions the suction cup vertically downwards to the pick-up position at the material storage,
switches the vacuum cleaner on and moves back up to the reference switch.
The suction pad now rotates counterclockwise to the material waste tray and vertically downwards to the deposit position, where it switches the suction pad off again.
The process is stopped by pressing the STOP button.
The program should be created in the Structured Text programming language (ST or SCL).

Planning
1. open the **LearningFactory_4_0_24V_Task01_HWConf** project in the TIA Portal and save it under the name **LearningFactory_4_0_24V_Task05_VacuumGripperVGR.**
2. copy the global variable table **VGR-Vacuum Gripper Robot**, as shown in programming task 1, from the library Library_LearningFactory_4_0_24V and insert it into your project.
3. check whether the jumpers for the power supply of the bidirectionally controlled motors are plugged in so that they are supplied via the associated PWM terminals 1, 2 and 3.
4. expand the observation table as shown in programming task 1 and check the function of the I / O signals.
5. create library-capable code blocks with the step chain in the Structured Text programming language (ST or SCL).
6. create a new organization block Main [OB1] in the programming language Structured Text (ST or SCL), call up the library-capable code block there and connect it to the global variables.
7 Load the complete program into the control unit.
8 Test the program with the vacuum suction pad (VGR).
9. monitor the program online and eliminate any errors.
10. save and archive the **LearningFactory_4_0_24V_Task05_VacuumGripperVGR** project.

Programming instructions

Notes on pulse width modulation (PWM):

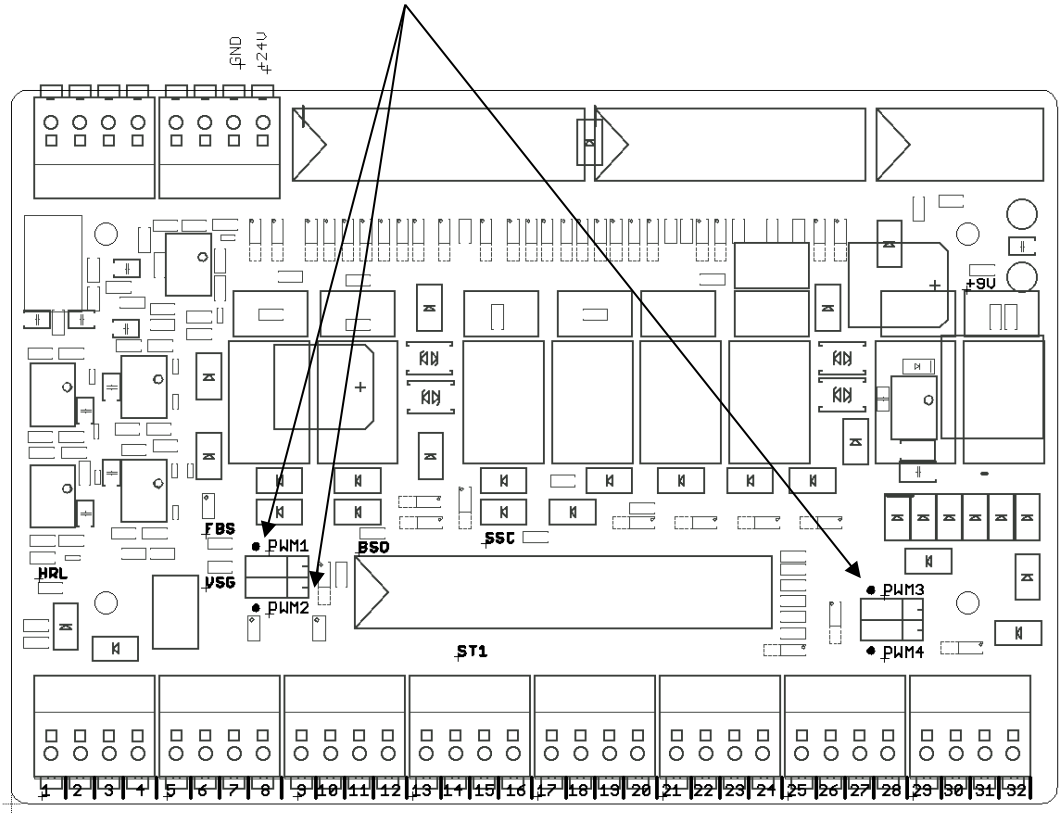
The bidirectionally controlled motors for rotation, the horizontal axis and the vertical axis are controlled via relays and supplied either directly via the 24V of the actuators or via the associated PWM terminals 1, 2 and 3.

	Vacuum suction pads (VGR)
PWM 1	Y(Vertical)
PWM 2	Z(Horizontal)
PWM 3	X(Rotate)
PWM 4	

Assignment of the PWM jumpers:

For this task, the jumpers on the left for PWM should be plugged in. The motors are controlled via relays and supplied with voltage via the corresponding PWM terminals.

The jumpers are located here on the adapter board



In the PLC program, you can specify integers (integer format) at the outputs for the PWM signals and thus control the speed. A low speed should be specified here for all axes with the value 400.

Notes on positioning

Due to the low speed of the 3 axes, it is sufficient to record pulse 1 of the respective encoder in the normal cyclical organization block OB1 for positioning.
A simple counter is sufficient to count the pulses in the program, the value of which is then compared.
You can determine the number of pulses required for the required positions by activating the individual axes and observing the counter value.

Programming task 6:

Multi-processing station with furnace (MPO); step chain with pulse width modulation (PWM) and parallel actions

Task definition:

In the multi-processing station with furnace (MPO), workpieces are to undergo a firing process before they are processed with a saw and then transported to the next station.

The motors for the vacuum cup movements and the turntable are controlled via relays and supplied with voltage via the associated PWM terminals. The speed of the motors can be set via the pulse width modulation (PWM) output signals.

First, a workpiece must be manually placed on the rotary table.
The START button is then used to start the sequence for a workpiece.
At the same time, the oven opens, the turntable rotates to the position near the suction cup and the suction cup moves to the turntable.
The suction cup then moves downwards, switches on and lifts the workpiece off the turntable.
Next, the linear axis moves to the oven, where the extension is extended.
When the workpiece arrives at the furnace, the vacuum gripper places the workpiece on the extension. This retracts, the furnace door closes and the firing process is carried out for 5 seconds.
Now the oven door opens again, the ejector extends and the vacuum gripper picks up the workpiece again to transport it back to the turntable.
The turntable then positions the workpiece under the saw. This processes it for 5 seconds.
The workpiece is then rotated to the position on the belt and pushed onto the belt by the push-out cylinder. The conveyor belt transports the workpiece to the light barrier at the end of the belt, where it can be removed by hand.

The program should be created in the Structured Text programming language (ST or SCL).
To shorten the time for a production cycle, several actions should be carried out simultaneously in one step if possible.
The signals for pulse width modulation (PWM) and for the compressor should not be specified in the steps of the step chain, but in downstream permanent operations.

Planning

1. open the **LearningFactory_4_0_24V_Task01_HWConf** project in the TIA Portal and save it under the name **LearningFactory_4_0_24V_Task06_MultiProcessingMPO.**
2. copy the global variable table **MPO-Multi Processing Station with Oven**, as shown in programming task 1, from the library **Library_LearningFactory_4_0_24V** and insert it into your project.
3. check whether the jumpers for the power supply of the bidirectionally controlled motors are plugged in so that they are supplied via the associated PWM terminals 1 and 3.
4. expand the observation table as shown in programming task 1 and check the function of the I / O signals.
5. create library-capable code blocks with the step chain in the Structured Text programming language (ST or SCL).
6. create a new organization block Main [OB1] in the programming language Structured Text (ST or SCL), call up the library-capable code block there and connect it to the global variables.
7. Load the complete program into the control unit.
8. Test the program with the vacuum suction pad (VGR).
9. monitor the program online and eliminate any errors.
10. save and archive the **LearningFactory_4_0_24V_Task06_MultiProcessingMPO** project.

Programming instructions

Notes on pulse width modulation (PWM):

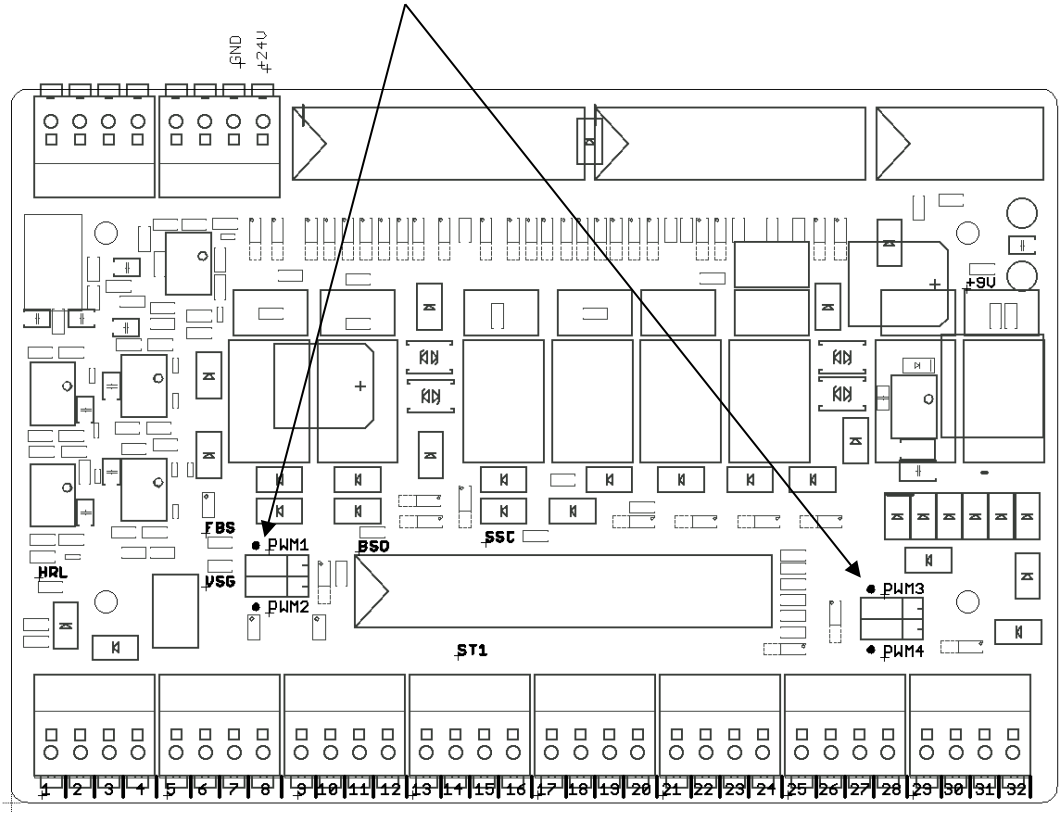
The bidirectionally controlled motors for the slewing ring and the horizontal movement of the suction cup on the linear axis are controlled via relays and supplied either directly via the 24V of the actuators or via the associated PWM terminals 1 and 3.

	Multi Processing Station (MPO)
PWM 1	Slewing ring
PWM 2	- not used -
PWM 3	Suction cup (horizontal)
PWM 4	Oven slide

Assignment of the PWM jumpers:

For this task, the jumpers on the left for PWM should be plugged in. The motors are controlled via relays and supplied with voltage via the corresponding PWM terminals.

The jumpers are located here on the adapter board



In the PLC program, you can specify integers (integer format) at the outputs for the PWM signals and thus control the speed.

Programming task 7:
Automated high-bay warehouse (HBW); step chain with pulse width modulation (PWM) and positioning function with encoder

Task definition:
With the automated high-bay warehouse (HBW) station, a pallet is to be retrieved from the front, upper position (position A1) and placed on the conveyor belt.

The motors for the movements of the storage and retrieval unit are controlled via relays and supplied with voltage via the associated PWM terminals. The speed of the motors can be set via the pulse width modulation (PWM) output signals.
Exact positioning of the axes is to be achieved here with the aid of the encoders.

The sequence is started with the START button. First, the storage and retrieval unit moves the cantilever to the rear and then simultaneously to the reference switches horizontally at the front outside the warehouse and vertically upwards.
In this reference position, the values for positioning are set to your reference value 0.
The storage and retrieval unit then moves to the front, upper position (position A1) for retrieval. There, the cantilever is extended and the pallet is then lifted by briefly moving the vertical axis upwards. The cantilever arm then moves back again.
The storage and retrieval unit then moves to the deposit position on the conveyor belt. There, the cantilever is extended again and the pallet is deposited by briefly moving the vertical axis downwards.
The delivery arm then moves back again while the conveyor belt transports the pallet to the light barrier at the outside end of the belt. The pallet must be removed by hand before the process can be restarted.
The process is stopped by pressing the STOP button.

The program should be created in the Structured Text programming language (ST or SCL).
To shorten the time for a production cycle, several actions should be carried out simultaneously in one step if possible.
The signals for pulse width modulation (PWM) should not be specified in the steps of the step chain, but in downstream permanent operations.

Planning

1. open the **LearningFactory_4_0_24V_Task01_HWConf** project in the TIA Portal and save it under the name **LearningFactory_4_0_24V_Task07_HighBayWarehouseHBW.**
2. copy the global variable table **HBW-Automated High-Bay Warehouse**, as shown in programming task 1, from the library **Library_LearningFactory_4_0_24V** and insert it into your project.
3. Create a **Positioning** group under Program blocks and add a **cyclic interrupt** OB **[OB30]** with a time cycle of 1000 µs according to the programming instructions. Read in the encoder inputs in the sub-process image of this alarm OB.
4. Copy the **FB_INC_DEC** library block for encoder evaluation from the **Library_LearningFactory_4_0_24V** library and insert it into the **Positioning** group in your project.
5. In the **Positioning** group, create a global data block **Positioning [DB90]** with the global variables for position (INT) and direction (BOOL) for the horizontal and vertical axis.
6. call up and connect the **FB_INC_DEC** block for encoder evaluation in the **cyclic interrupt [OB30]** alarm OB for the horizontal and vertical axes.
7. check whether the jumpers for the power supply of the bidirectionally controlled motors are plugged in so that they are supplied via the associated PWM terminals 1, 2, 3 and 4.

8) As shown in programming task 1, add all input/output signals and the values from the global data block **Positioning [DB90]** to the monitoring table. Check the function of the I/O signals and the encoder evaluation using the monitoring table and determine the position values for the positions required in the program.

9. create library-capable code blocks for the step chain in the Structured Text programming language (ST or SCL).
10. create a new organization block Main [OB1] in the programming language Structured Text (ST or SCL), call up the library-capable code block there and connect it to the global variables.

11 Load the complete program into the control unit.

12 Test the program with the automated high-bay warehouse (HBW).

13. monitor the program online and eliminate any errors.

14.save and archive the project **LearningFactory_4_0_24V_Task07_HighBayWarehouseHBW.**

Notes on pulse width modulation (PWM):

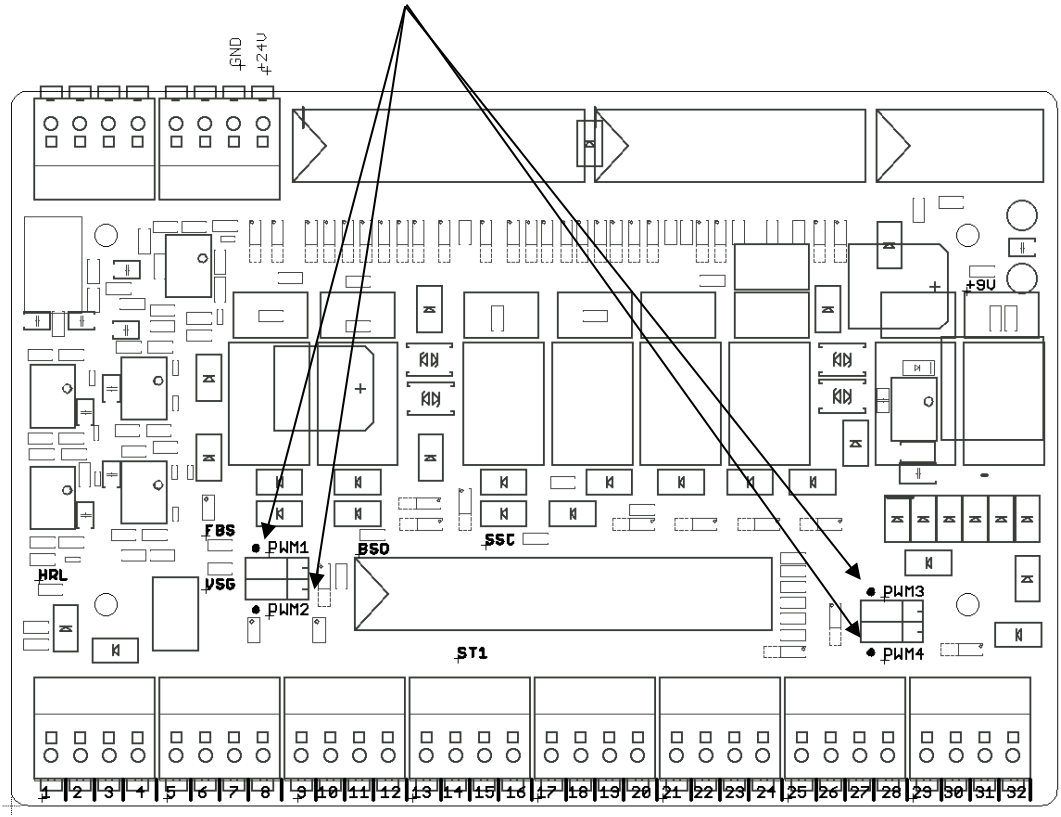
The bidirectionally controlled motors for the boom, the conveyor belt, the horizontal axis and the vertical axis are controlled via relays and can be supplied either directly via the 24V of the actuators or via the associated PWM terminals 1, 2, 3 and 4.

	High-bay warehouse (HBW)
PWM 1	Conveyor belt
PWM 2	X(Horizontal)
PWM 3	Y(Vertical)
PWM 4	Outrigger

Assignment of the PWM jumpers:

For this task, the jumpers on the left for PWM should be plugged in. The motors are controlled via relays and supplied with voltage via the corresponding PWM terminals.

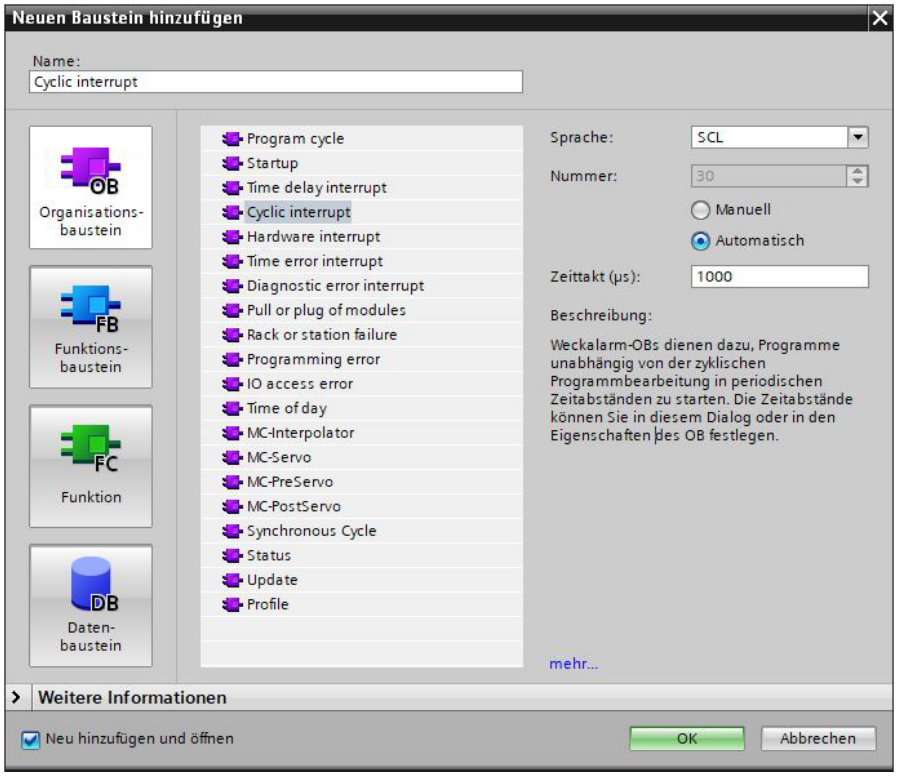
The jumpers are located here on the adapter board



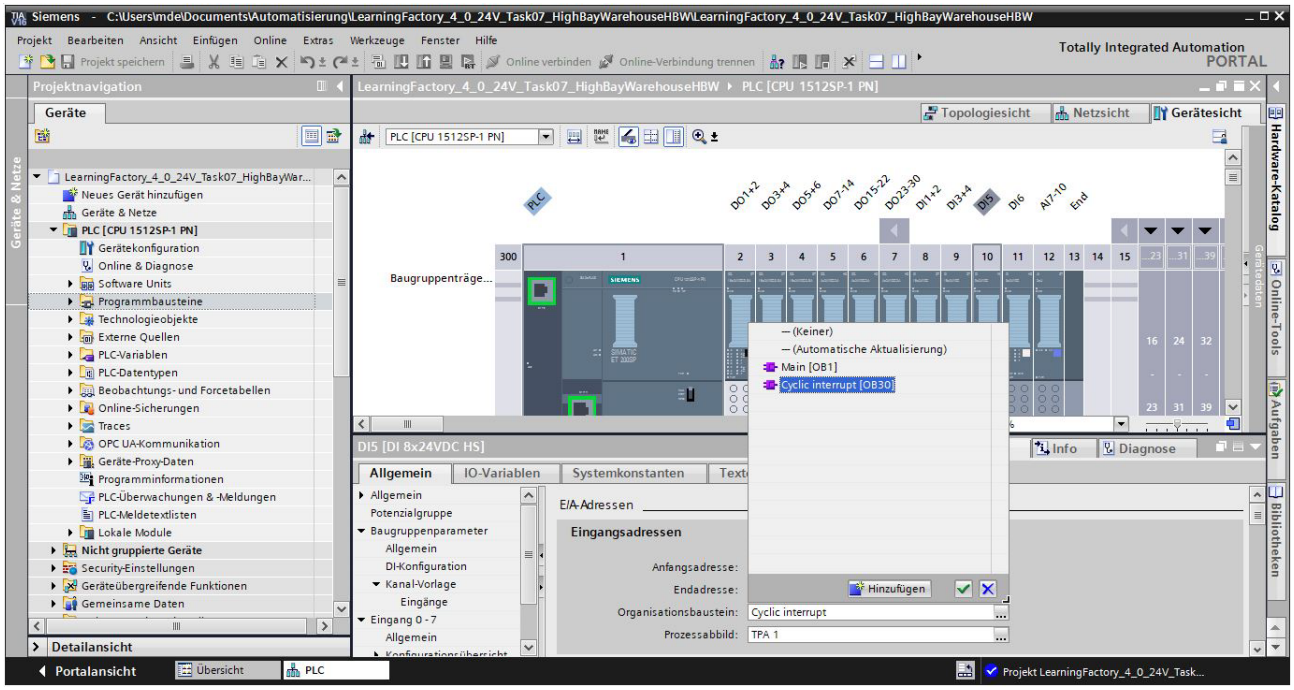
In the PLC program, you can specify integers (integer format) at the outputs for the PWM signals and thus control the speed.

Instructions for creating a wake-up alarm OB and reading in the encoder inputs

With the help of wake-up interrupt OBs, program sections can be started after equidistant time intervals and inputs can also be read in accordingly. Create a wake-up interrupt OB with a cycle time of 1 ms as follows:



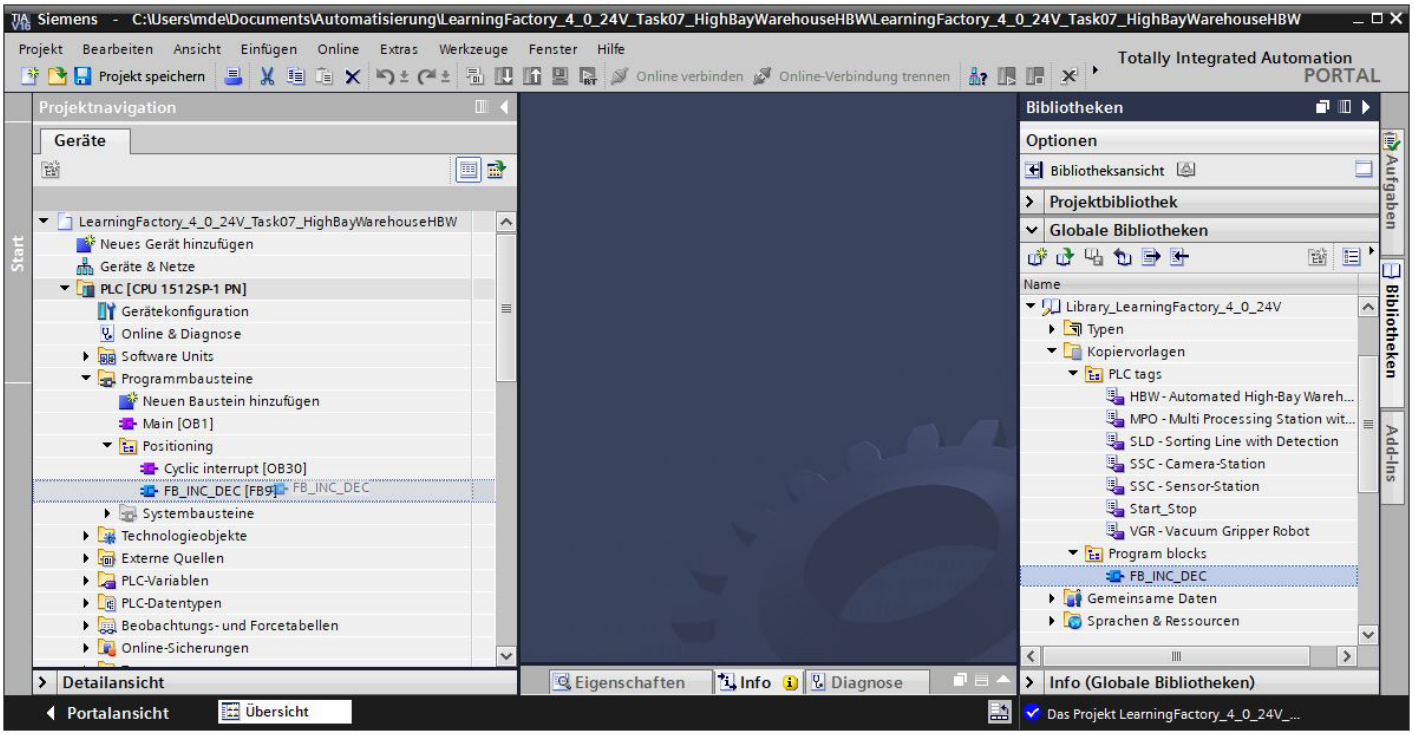
For the modules for the encoder inputs, it must still be set that the inputs are to be read in the sub-process image (TPA 1) of this wake-up alarm OB.



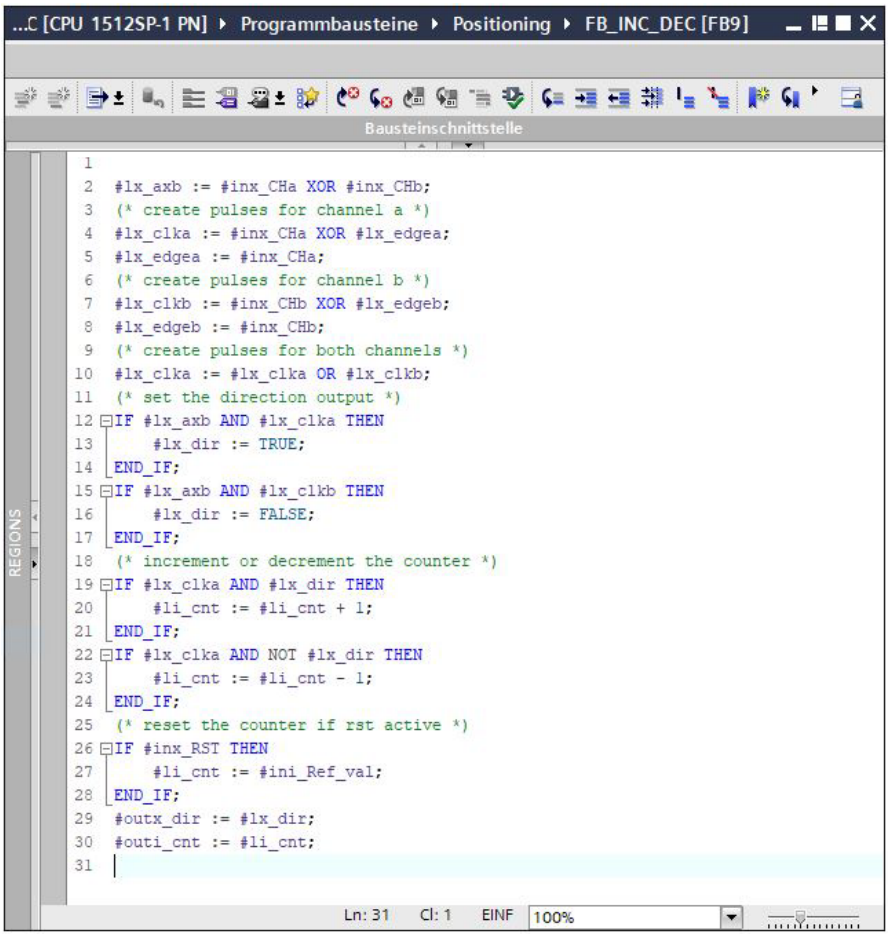
Notes on positioning with encoder and the fischertechnik library block FB_INC_DEC

In this programming task, tracks A+B or pulses 1 and 2 of the respective encoders for the horizontal and vertical axes are to be recorded in a wake-up alarm organization block with a cycle time of 1 ms for exact positioning even at higher movement speeds.

The library **Library_LearningFactory_4_0_24V** already provides a block **FB_INC_DEC** for encoder evaluation. This can simply be copied from there into the project.



The FB_INC_DEC function block for encoder evaluation evaluates the pulses of the two channels at the #inx_CHa and #inx_CHb inputs.
The direction #outx_dir and a position value #outi_cnt are determined from this and made available as outputs.
The position value can be reset to a reference value #ini_REF_val using the #inx_RST input.



The FB_INC_DEC block for encoder evaluation should be called and wired in a wake-up alarm OB for both the horizontal and vertical axis.

```
"FB_INC_DEC_DB"(inx_CHa:=false,
                inx_CHb:=false,
                inx_RST:=False,
                ini_Ref_val:=0,
                outx_dir=>_bool_out_,
                outi_cnt=>_int_out_);
```

The position values for the positions required in the program can be determined by controlling the individual axes and observing the position values in the observation table. The reference position is always 0.