SIEMENS

SCE Training Curriculum

Siemens Automation Cooperates with Education | 05/2017

TIA Portal Module 032-100

Basics of FC Programming
with SIMATIC S7-1500

Cooperates with Education
Automation
SIEMENS

Matching SCE trainer packages for these training curriculums

SIMATIC Controllers

- **SIMATIC ET 200SP Open Controller CPU 1515SP PC F and HMI RT SW**
  Order no.: 6ES7677-2FA41-4AB1
- **SIMATIC ET 200SP Distributed Controller CPU 1512SP F-1 PN Safety**
  Order no.: 6ES7512-1SK00-4AB2
- **SIMATIC CPU 1516F PN/DP Safety**
  Order no.: 6ES7516-3FN00-4AB2
- **SIMATIC S7 CPU 1516-3 PN/DP**
  Order no.: 6ES7516-3AN00-4AB3
- **SIMATIC CPU 1512C PN with Software and PM 1507**
  Order no.: 6ES7512-1CK00-4AB1
- **SIMATIC CPU 1512C PN with Software, PM 1507 and CP 1542-5 (PROFIBUS)**
  Order no.: 6ES7512-1CK00-4AB2
- **SIMATIC CPU 1512C PN with Software**
  Order no.: 6ES7512-1CK00-4AB6
- **SIMATIC CPU 1512C PN with Software and CP 1542-5 (PROFIBUS)**
  Order no.: 6ES7512-1CK00-4AB7

SIMATIC STEP 7 Software for Training

- **SIMATIC STEP 7 Professional V14 SP1 - Single license**
  Order no.: 6ES7822-1AA04-4YA5
- **SIMATIC STEP 7 Professional V14 SP1- Classroom license (up to 6 users)**
  Order no.: 6ES7822-1BA04-4YA5
- **SIMATIC STEP 7 Professional V14 SP1 - Upgrade license (up to 6 users)**
  Order no.: 6ES7822-1AA04-4YE5
- **SIMATIC STEP 7 Professional V14 SP1 - Student license (up to 20 users)**
  Order no.: 6ES7822-1AC04-4YA5

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

Continued training

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

Additional information regarding SCE

siemens.com/sce

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

Table of contents

BASICS OF FC PROGRAMMING

1 Goal

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

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

2 Prerequisite

This chapter builds on the hardware configuration of SIMATIC S7 CPU1516F-3 PN/DP. However, other hardware configurations that have digital input and output cards can be used. You can use the following project for this chapter, for example:

SCE_EN_012_101__Hardware_Configuration_CPU1516F.zap13

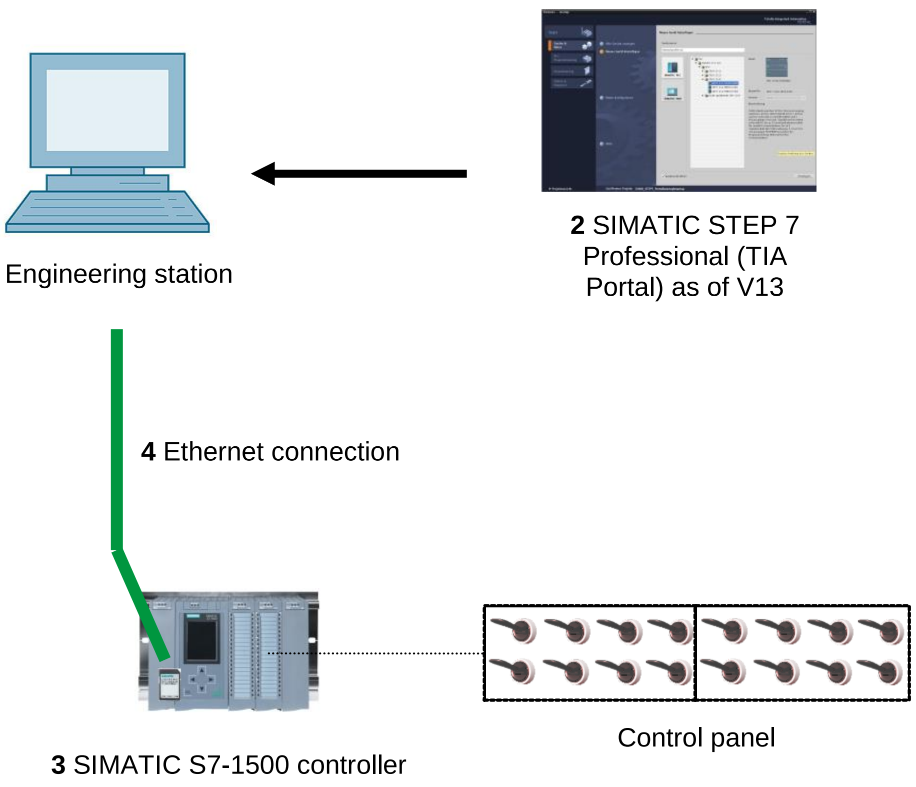
3 Required hardware and software

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

2 SIMATIC STEP 7 Professional software in TIA Portal – as of V13

3 SIMATIC S7-1500/S7-1200/S7-300 controller, e.g. CPU 1516F-3 PN/DP – Firmware as of V1.6 with memory card and 16DI/16DO and 2AI/1AO
Note: The digital inputs should be fed out to a control panel.

4 Ethernet connection between engineering station and controller


1 Engineering station
2 SIMATIC STEP 7 Professional (TIA Portal) as of V13
4 Ethernet connection
3 SIMATIC S7-1500 controller
Control panel

4 Theory

4.1 Operating system and application program

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

– Performing a warm restart

– Updating the process image of the inputs and outputs

– Cyclically calling the user program

– Detecting interrupts and calling interrupt OBs

– Detecting and handling errors

– Managing memory areas

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

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

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

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

– Reaction to interrupts and interrupt inputs

– Error handling during normal program execution

4.2 Organization blocks

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

– Cyclic program processing (e.g. OB1)

– Startup characteristics of the controller

– Interrupt-driven program processing

– Error handling

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

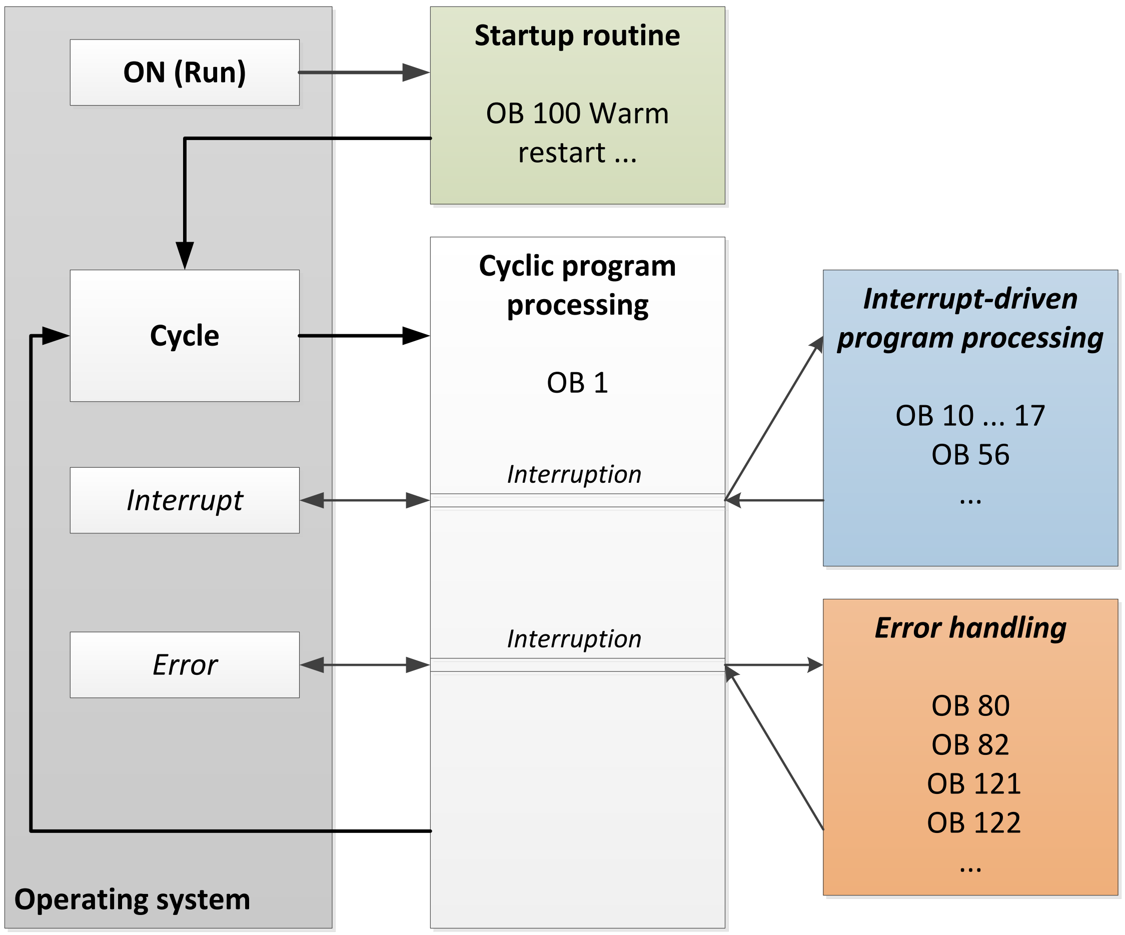


Figure 1: Start events in the operating system and OB call

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

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

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

Table 1 gives a couple of examples of start events for a SIMATIC S7-1500, their possible OB number(s) and the default system reaction in the event the organization block is not present in the controller.

Start event	**Possible OB numbers**	**Default system reaction**
Startup	100, ≥ 123	Ignore
Cyclic program	1, ≥ 123	Ignore
Time-of-day interrupt	10 to 17, ≥ 123	-
Update interrupt	56	Ignore
Scan cycle monitoring time exceeded once	80	STOP
Diagnostic interrupt	82	Ignore
Programming error	121	STOP
IO access error	122	Ignore

Table 1: OB numbers for various start events

4.3 Process image and cyclic program processing

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

The cyclic program processing sequence is as follows:

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

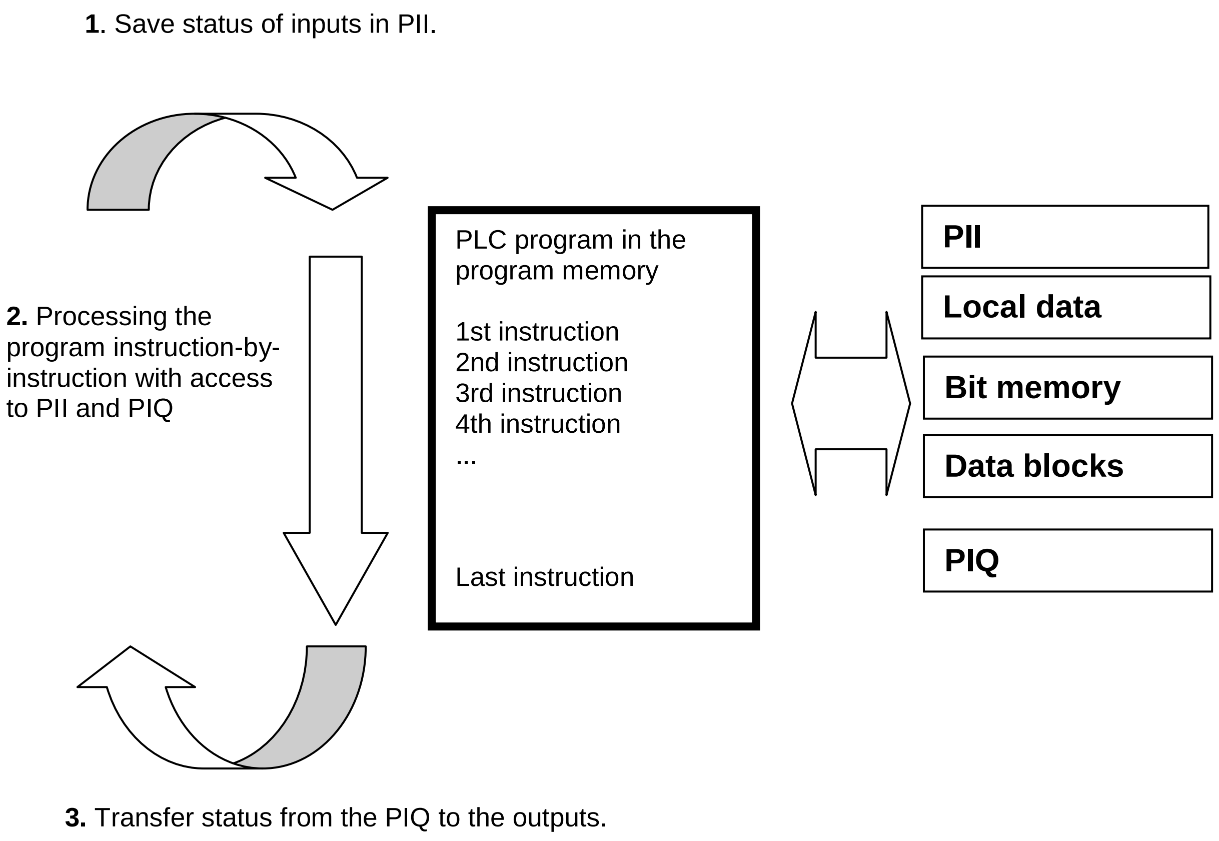

1. Save status of inputs in PII.
2. Processing the program instruction-by-instruction with access to PII and PIQ
PLC program in the program memory
1st instruction
2nd instruction
3rd instruction
4th instruction
...
Last instruction
PII
Local data
Bit memory
Data blocks
PIQ
3. Transfer status from the PIQ to the outputs.

Figure 2: Cyclic program processing

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

4.4 Functions

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

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

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

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

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

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

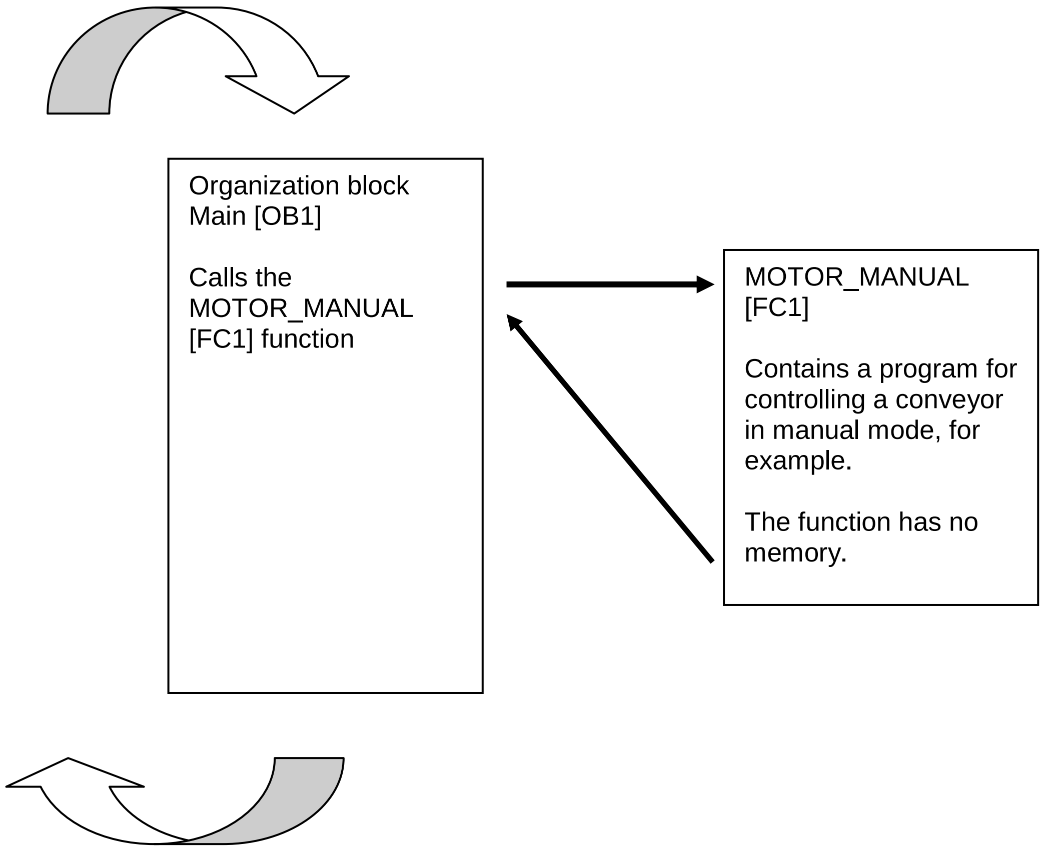

Organization block Main [OB1]
Calls the MOTOR_MANUAL [FC1] function
MOTOR_MANUAL [FC1]
Contains a program for controlling a conveyor in manual mode, for example.
The function has no memory.

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

4.5 Function blocks and instance data blocks

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

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

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

– Whenever timers and counters are required in the blocks.

– Whenever information must be saved in the program, such as pre-selection of the operating mode with a button.

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

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

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

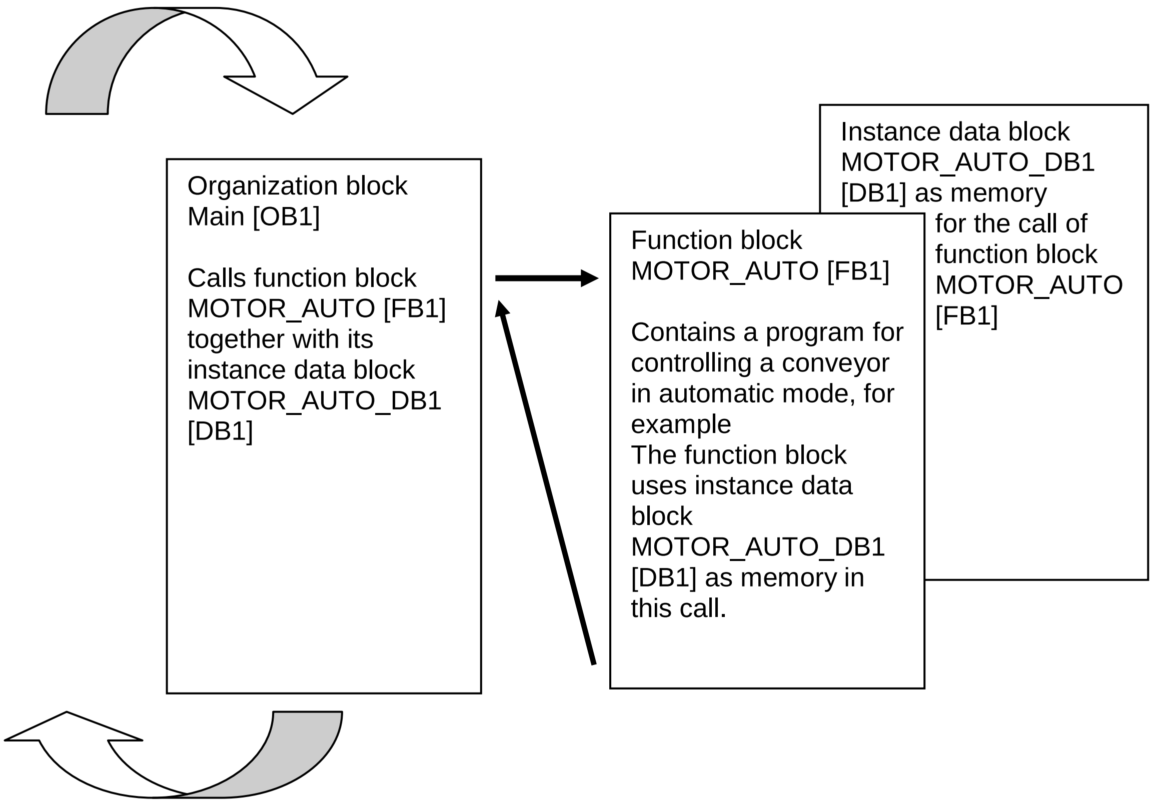

Organization block Main [OB1]
Calls function block MOTOR_AUTO [FB1] together with its instance data block MOTOR_AUTO_DB1 [DB1]
Function block MOTOR_AUTO [FB1]
Contains a program for controlling a conveyor in automatic mode, for example
The function block uses instance data block MOTOR_AUTO_DB1 [DB1] as memory in this call.
Instance data block MOTOR_AUTO_DB1 [DB1] as memory for the call of function block MOTOR_AUTO [FB1]

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

4.6 Global data blocks

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

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

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

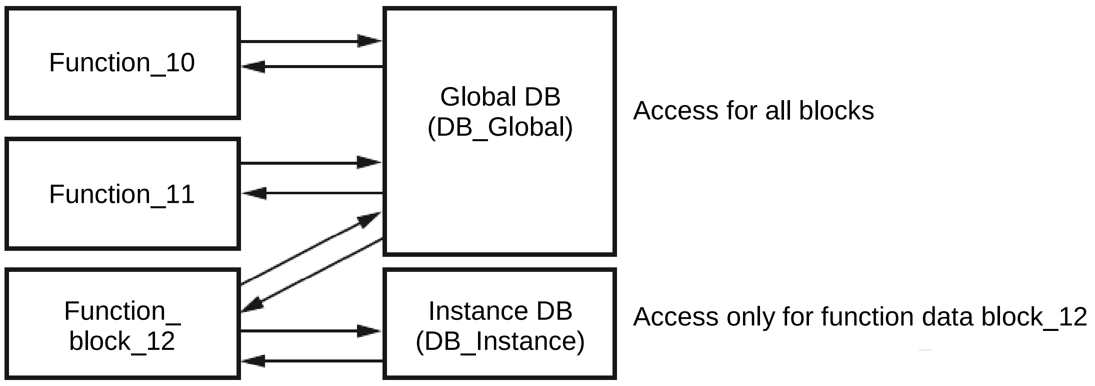


Figure 5: Difference between global DB and instance DB.

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

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

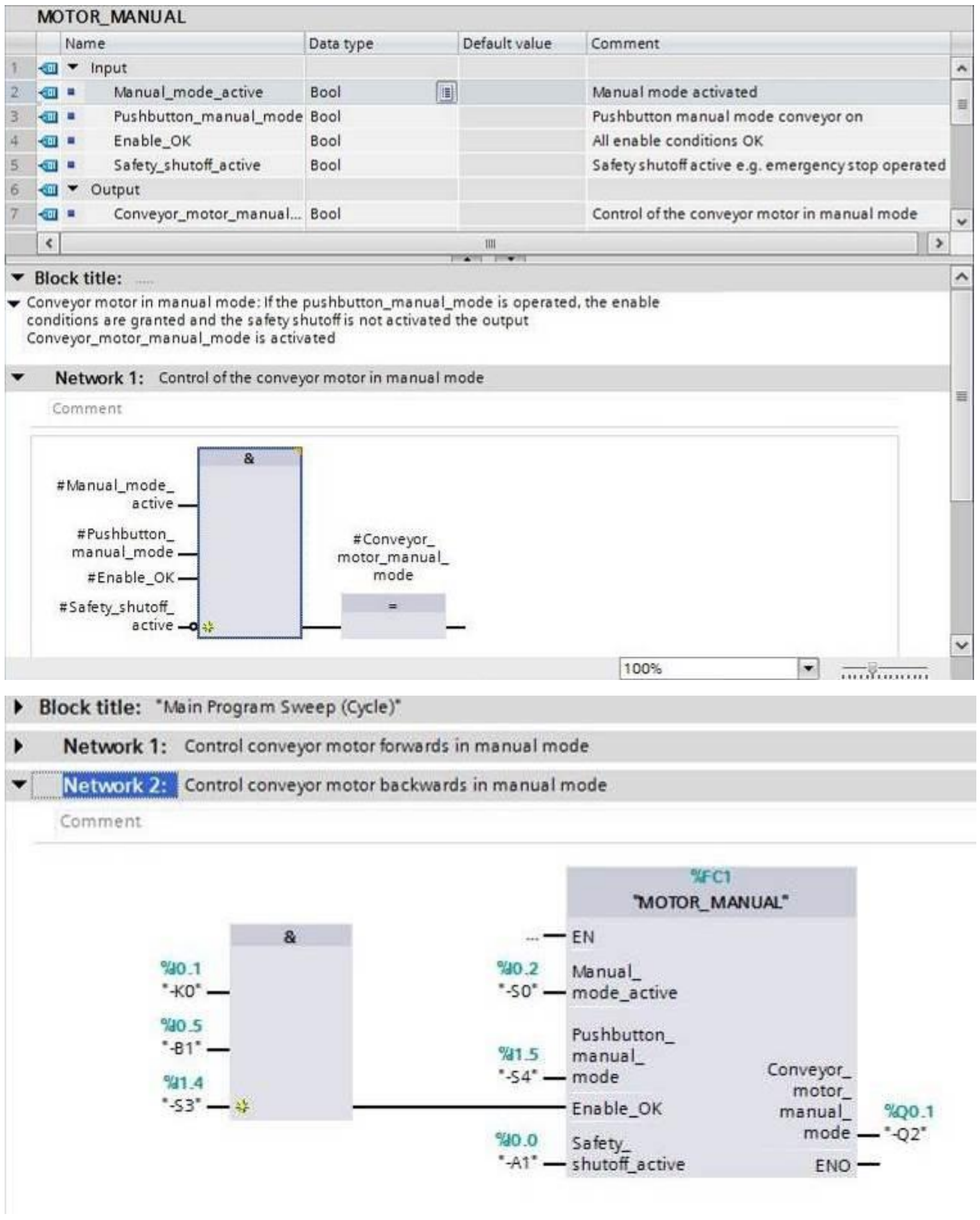
– Saving of recipes for particular products.

4.7 Library-compatible code blocks

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

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

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



MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	Input			
2	Manual_mode_active	Bool		Manual mode activated
3	Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
4	Enable_OK	Bool		All enable conditions OK
5	Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
6	Output			
7	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode

Block title:

Conveyor motor in manual mode: If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

Network 1: Control of the conveyor motor in manual mode

Comment

Block title: "Main Program Sweep (Cycle)"

Network 1: Control conveyor motor forwards in manual mode

Network 2: Control conveyor motor backwards in manual mode

Comment

&
%I0.1 "-K0"
%I0.5 "-B1"
%I1.4 "-S3"
%FC1
"MOTOR_MANUAL"
... EN
%I0.2 "-S0" Manual_mode_active
%I1.5 "-S4" Pushbutton_manual_mode
Enable_OK
%I0.0 "-A1" Safety_shutoff_active
Conveyor_motor_manual_mode %Q0.1 "-Q2"
ENO

Figure 6: Library-compatible function with call in OB1

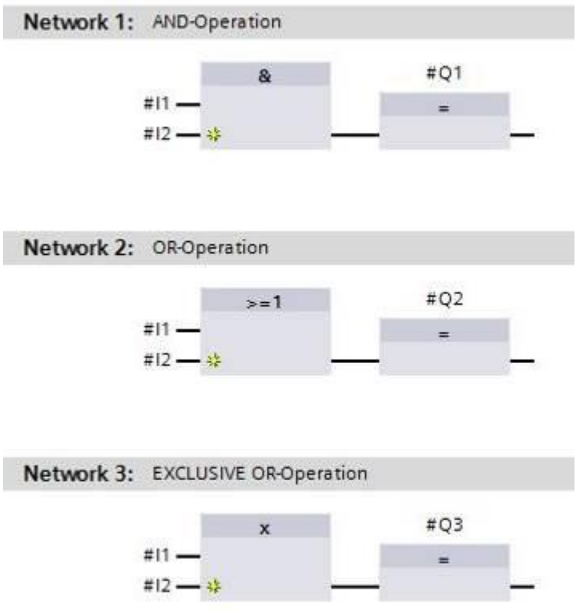
4.8 Programming languages

The available programming languages for programming functions are Function Block Diagram (FBD), Ladder Logic (LAD), Statement List (STL) and Structured Control Language (SCL). For function blocks, the GRAPH programming language is additionally available for programming graphical step sequences.

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

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

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



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

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

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

Figure 7: Binary functions in FBD and associated logic table

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

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

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

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

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

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

5 Task

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

– Manual mode – Conveyor motor in manual mode

6 Planning

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

6.1 EMERGENCY STOP

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

6.2 Manual mode – Conveyor motor in manual mode

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

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

Table 2: Parameters for FC "MOTOR_MANUAL"

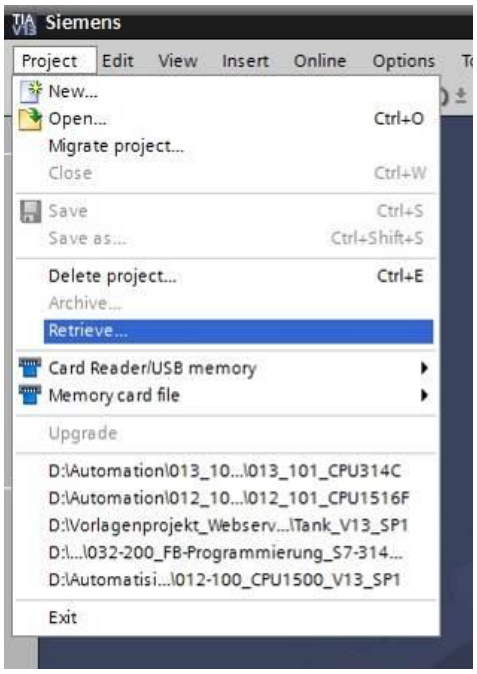
Output Conveyor_motor_manual_mode is ON as long as Pushbutton_manual_mode is pressed, manual mode is activated, the enable conditions are OK and the safety shutoff is not active.

7 Structured step-by-step instructions

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

7.1 Retrieve an existing project

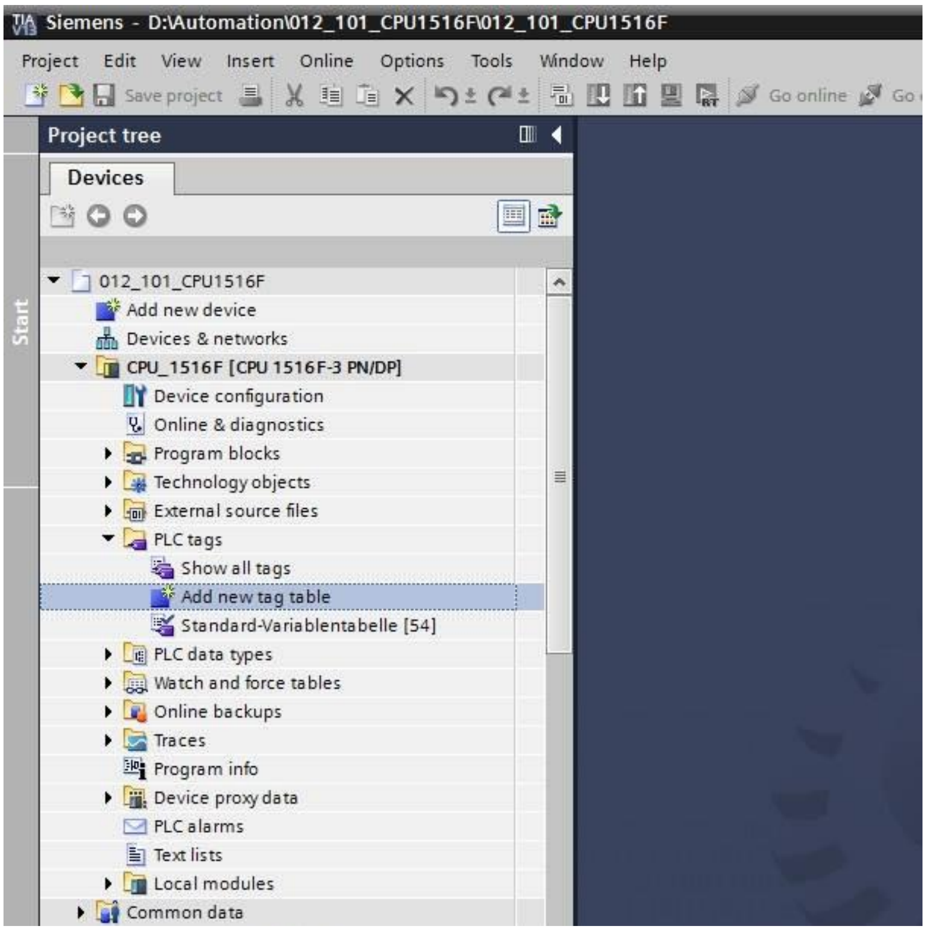
→ Before we can start programming the function (FC) "MOTOR_MANUAL", we need a project with a hardware configuration (e.g. SCE_EN_012_101_Hardware_Configuration_S7-1516F_R1502.zap). To retrieve an existing project that has been archived, you must select the relevant archive with → Project → Retrieve in the project view Confirm your selection with Open. (→ Project → Retrieve → Select a .zap archive → Open)



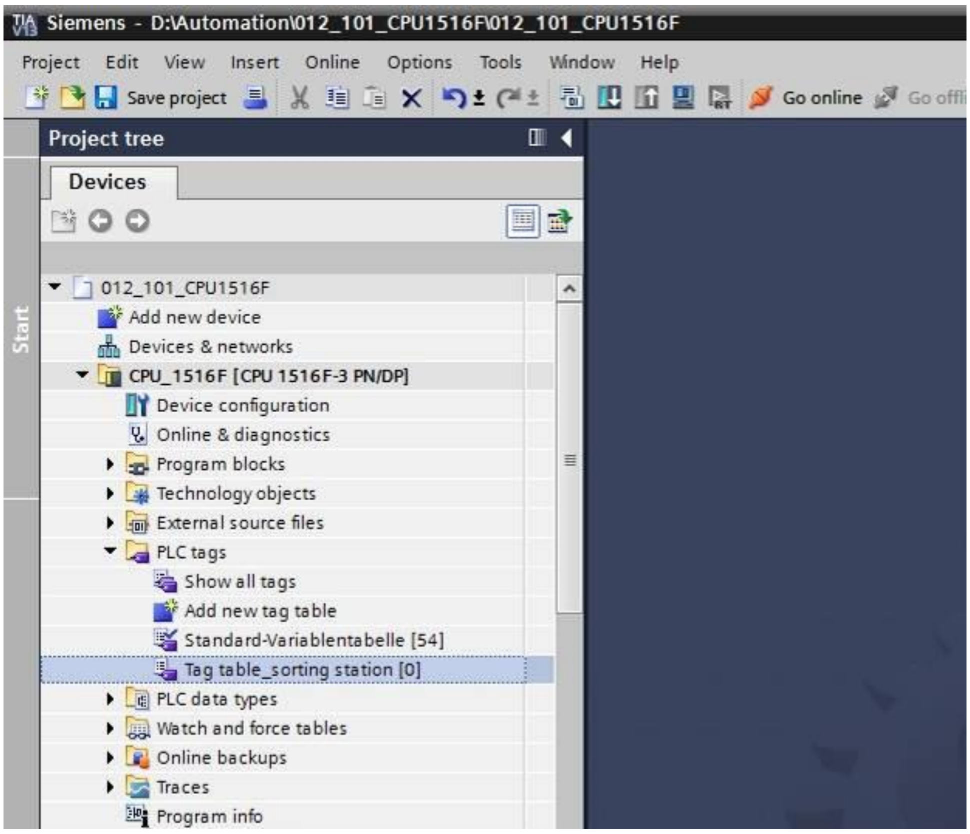
→ The next step is to select the target directory where the retrieved project will be stored. Confirm your selection with "OK". (→ Target directory → OK)

7.2 Create a new tag table

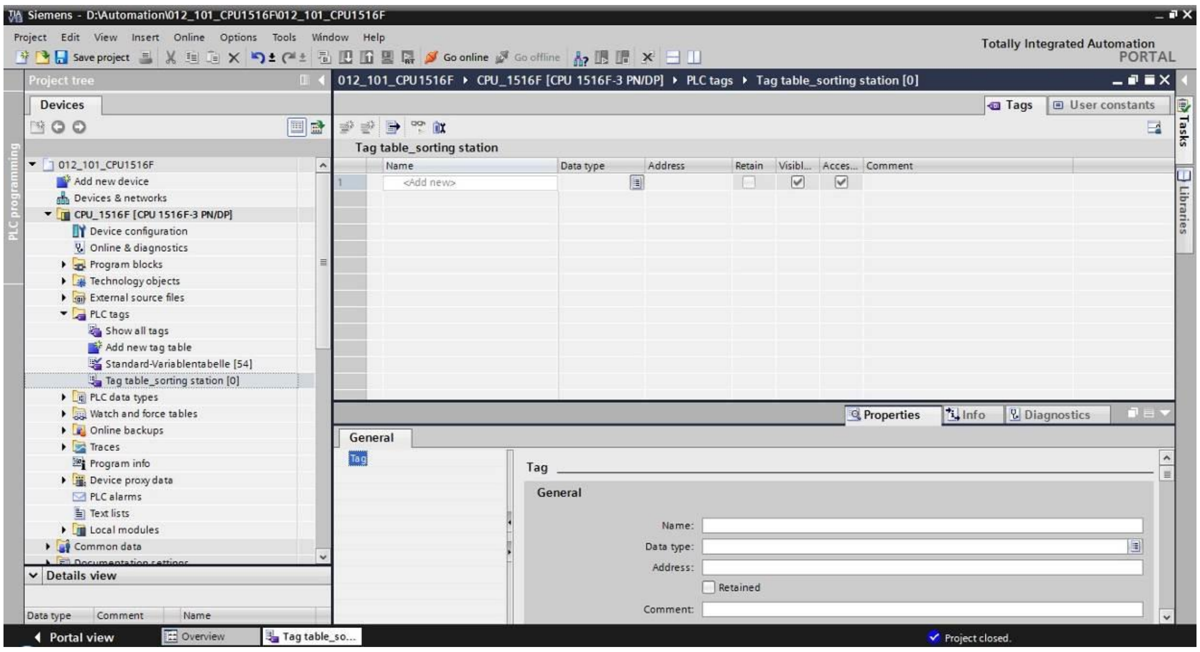
→ In the project view, navigate to the → PLC tags of your controller and create a new tag table by double-clicking → Add new tag table.



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

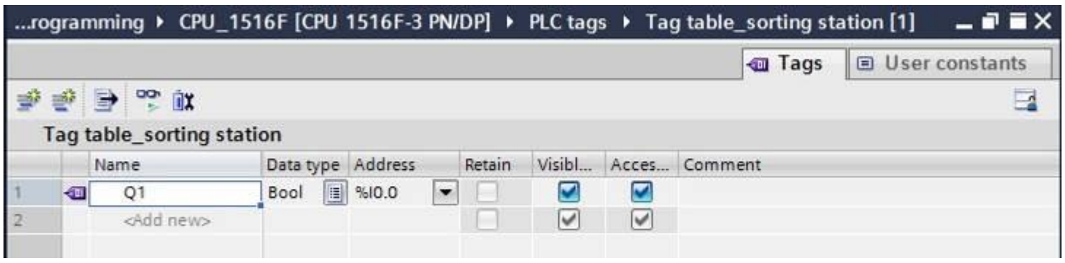


→ Open this tag table with a double-click. (→ Tag_table_sorting_station)



7.3 Create new tags within a tag table

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



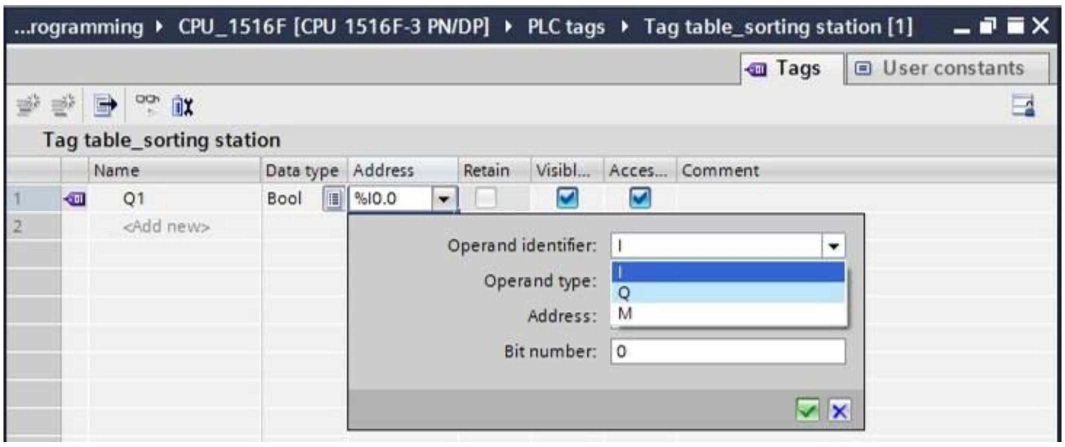
...rogramming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ PLC tags ▸ Tag table_sorting station [1]

Tags | User constants

Tag table_sorting station

	Name	Data type	Address	Retain	Visibl...	Acces...	Comment
1	Q1	Bool	%I0.0	☐	✓	✓	
2	<Add new>			☐	✓	✓	

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



...rogramming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ PLC tags ▸ Tag table_sorting station [1]

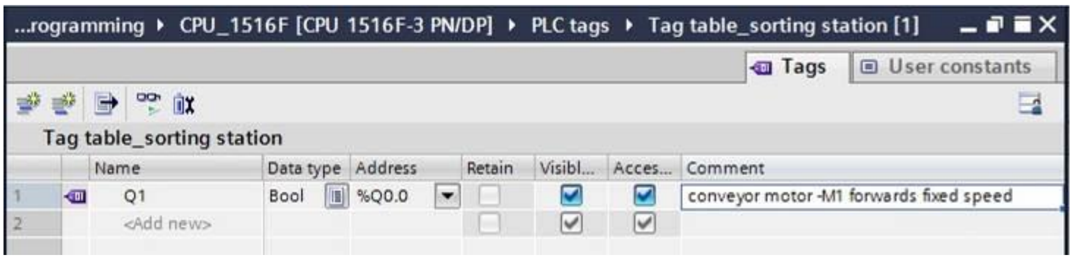
Tags | User constants

Tag table_sorting station

	Name	Data type	Address	Retain	Visibl...	Acces...	Comment
1	Q1	Bool	%I0.0	☐	✓	✓	
2	<Add new>						

Operand identifier: I (I, Q, M)
Operand type:
Address:
Bit number: 0

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



...rogramming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ PLC tags ▸ Tag table_sorting station [1]

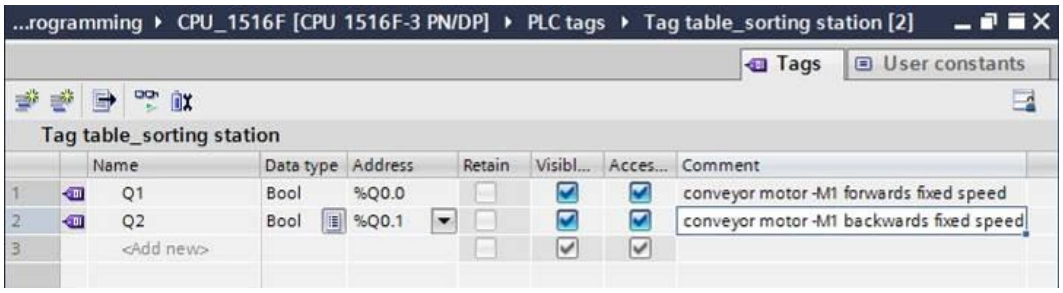
Tags | User constants

Tag table_sorting station

	Name	Data type	Address	Retain	Visibl...	Acces...	Comment
1	Q1	Bool	%Q0.0	☐	✓	✓	conveyor motor -M1 forwards fixed speed
2	<Add new>			☐	✓	✓	

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

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


...rogramming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ PLC tags ▸ Tag table_sorting station [2]
Tags
User constants

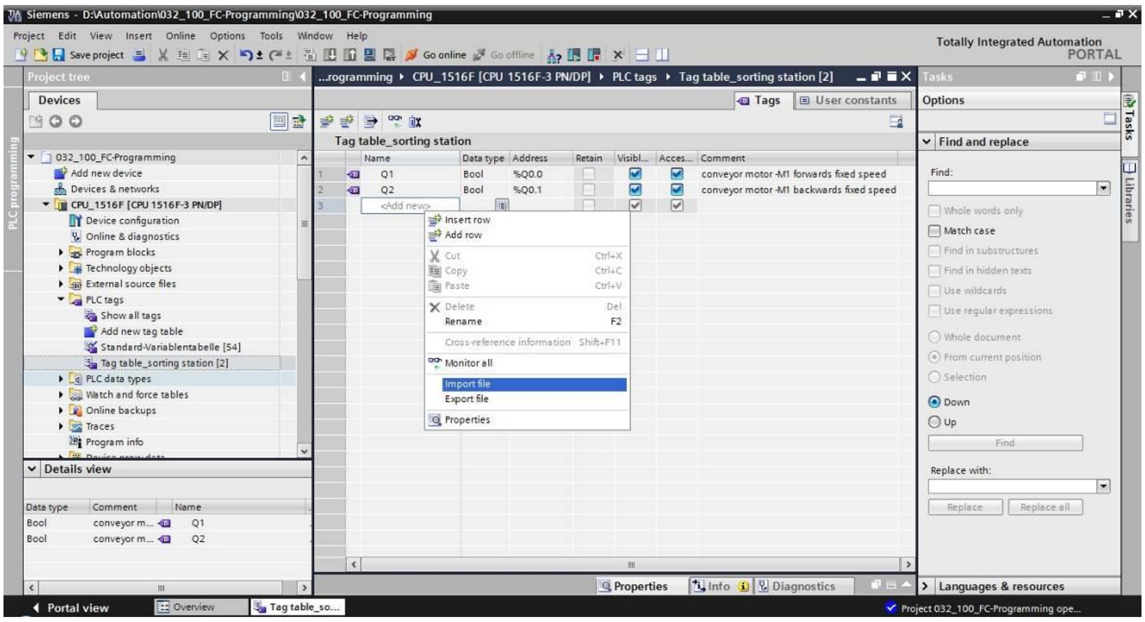
Tag table_sorting station

	Name	Data type	Address	Retain	Visibl...	Acces...	Comment
1	Q1	Bool	%Q0.0	☐	✓	✓	conveyor motor -M1 forwards fixed speed
2	Q2	Bool	%Q0.1	☐	✓	✓	conveyor motor -M1 backwards fixed speed
3	<Add new>			☐	✓	✓	

7.4 Import "Tag_table_sorting_station"

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

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



Tag table_sorting station

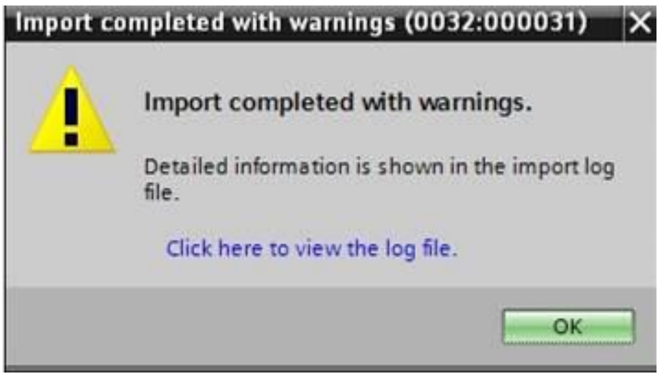
	Name	Data type	Address	Retain	Visibl...	Acces...	Comment
1	Q1	Bool	%Q0.0	☐	✓	✓	conveyor motor -M1 forwards fixed speed
2	Q2	Bool	%Q0.1	☐	✓	✓	conveyor motor -M1 backwards fixed speed
3	<Add new>			☐	✓	✓	

Data type	Comment	Name
Bool	conveyor m...	Q1
Bool	conveyor m...	Q2

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

(→ SCE_EN_020-100_Tag_table_sorting_station… → Open)

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



Import completed with warnings (0032:000031)

Import completed with warnings.

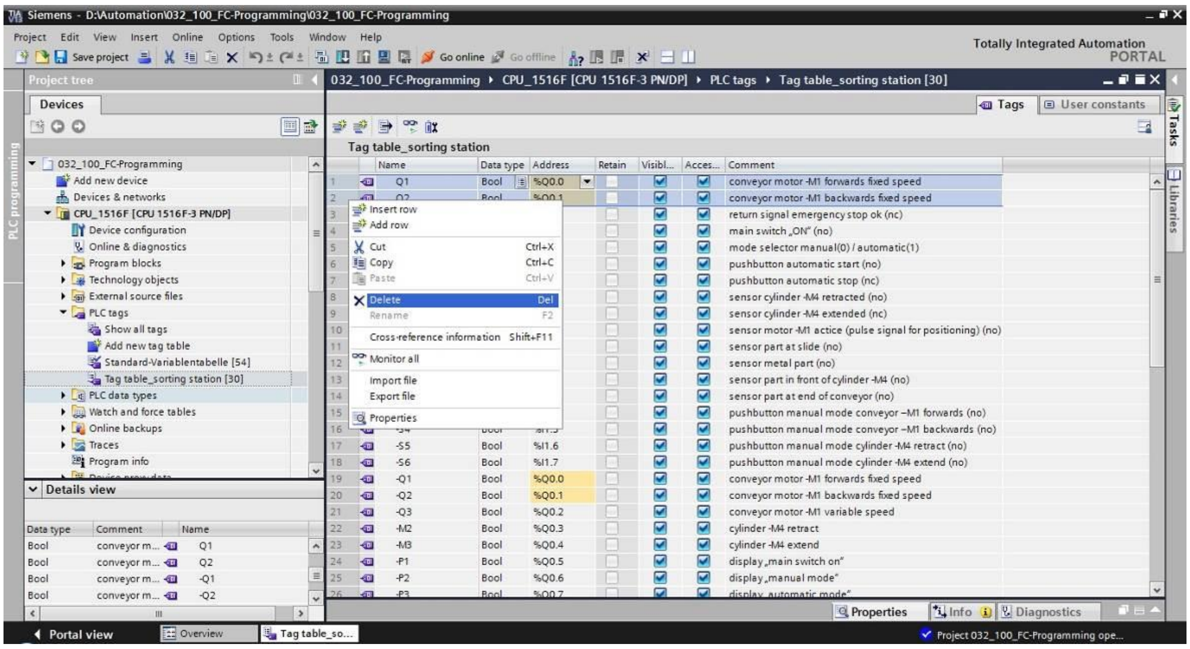
Detailed information is shown in the import log file.

Click here to view the log file.

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

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

(→ Right-click on selected tags → Delete)


Siemens - D:\Automation\032_100_FC-Programming\032_100_FC-Programming
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation PORTAL
Project tree
Devices
032_100_FC-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ PLC tags ▸ Tag table_sorting station [30]
Tags
User constants
Insert row
Add row
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Rename F2
Cross-reference information Shift+F11
Monitor all
Import file
Export file
Properties
Info
Diagnostics
Portal view
Overview
Tag table_so...
Project 032_100_FC-Programming ope...

Tag table_sorting station

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

Project tree:

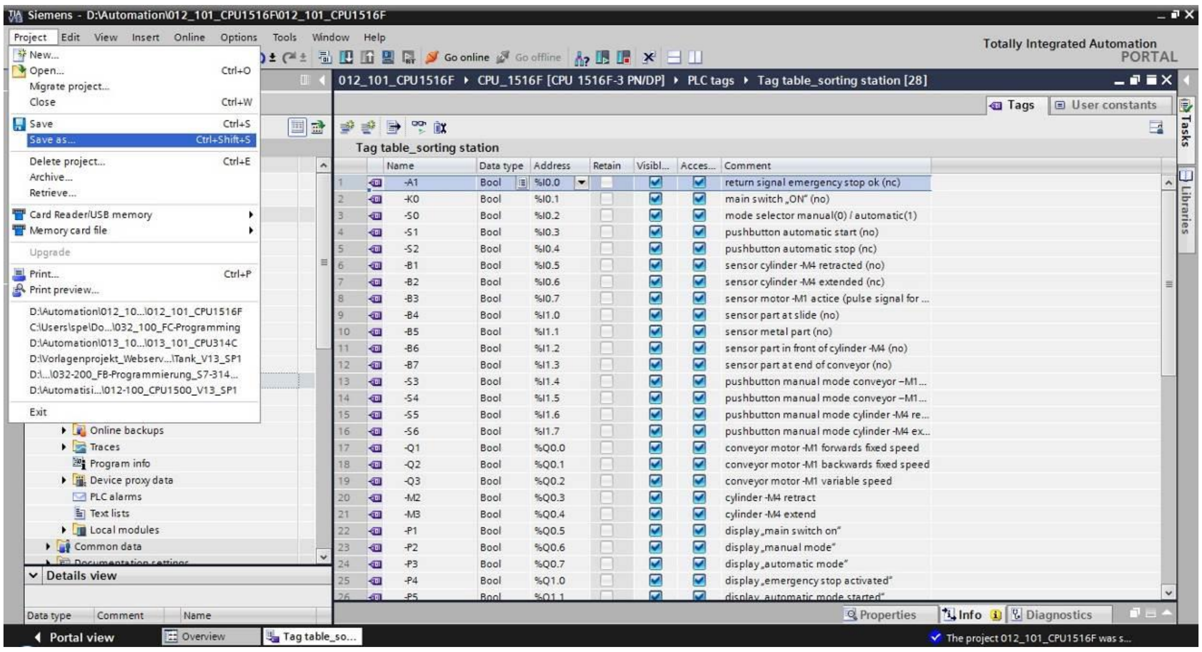
- 032_100_FC-Programming
  - Add new device
  - Devices & networks
  - CPU_1516F [CPU 1516F-3 PN/DP]
    - Device configuration
    - Online & diagnostics
    - Program blocks
    - Technology objects
    - External source files
    - PLC tags
      - Show all tags
      - Add new tag table
      - Standard-Variablentabelle [54]
      - Tag table_sorting station [30]
    - PLC data types
    - Watch and force tables
    - Online backups
    - Traces
    - Program info

Details view

Data type	Comment	Name
Bool	conveyor m...	Q1
Bool	conveyor m...	Q2
Bool	conveyor m...	-Q1
Bool	conveyor m...	-Q2

→ You now have a complete symbol table of the digital inputs and outputs in front of you. Save your project under the name 032-100_FCProgramming.

(→ Project → Save as ...→032-100_FCProgramming → Save)


Siemens - D:\Automation\012_101_CPU1516F\012_101_CPU1516F
Project
Edit
View
Insert
Online
Options
Tools
Window
Help
New...
Open... Ctrl+O
Migrate project...
Close Ctrl+W
Save Ctrl+S
Save as... Ctrl+Shift+S
Delete project... Ctrl+E
Archive...
Retrieve...
Card Reader/USB memory
Memory card file
Upgrade
Print... Ctrl+P
Print preview...
D:\Automation\012_10...\012_101_CPU1516F
C:\Users\spe\Do...\032_100_FC-Programming
D:\Automation\013_10...\013_101_CPU314C
D:\Vorlagenprojekt_Webserv...\Tank_V13_SP1
D:\...\032-200_FB-Programmierung_S7-314...
D:\Automatisi...\012-100_CPU1500_V13_SP1
Exit
Online backups
Traces
Program info
Device proxy data
PLC alarms
Text lists
Local modules
Common data
Details view
Go online
Go offline
Totally Integrated Automation PORTAL
012_101_CPU1516F ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ PLC tags ▸ Tag table_sorting station [28]
Tags
User constants
Properties
Info
Diagnostics
Portal view
Overview
Tag table_so...
The project 012_101_CPU1516F was s...

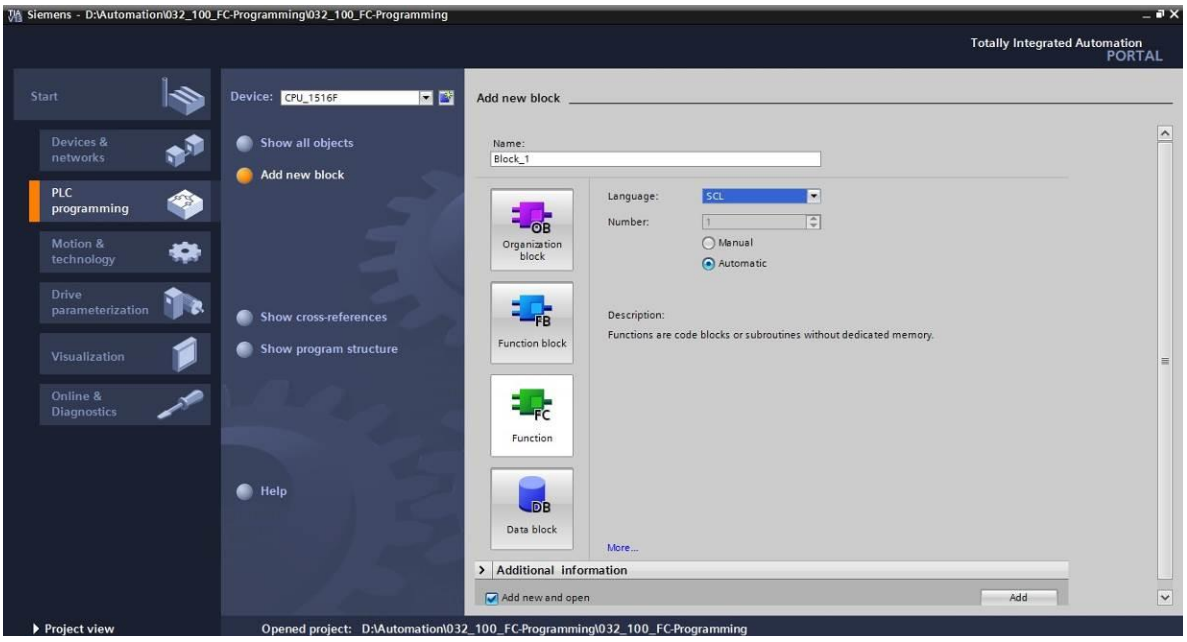
Tag table_sorting station

	Name	Data type	Address	Retain	Visibl...	Acces...	Comment
1	-A1	Bool	%I0.0		✓	✓	return signal emergency stop ok (nc)
2	-K0	Bool	%I0.1		✓	✓	main switch „ON" (no)
3	-S0	Bool	%I0.2		✓	✓	mode selector manual(0) / automatic(1)
4	-S1	Bool	%I0.3		✓	✓	pushbutton automatic start (no)
5	-S2	Bool	%I0.4		✓	✓	pushbutton automatic stop (nc)
6	-B1	Bool	%I0.5		✓	✓	sensor cylinder -M4 retracted (no)
7	-B2	Bool	%I0.6		✓	✓	sensor cylinder -M4 extended (nc)
8	-B3	Bool	%I0.7		✓	✓	sensor motor -M1 actice (pulse signal for ...
9	-B4	Bool	%I1.0		✓	✓	sensor part at slide (no)
10	-B5	Bool	%I1.1		✓	✓	sensor metal part (no)
11	-B6	Bool	%I1.2		✓	✓	sensor part in front of cylinder -M4 (no)
12	-B7	Bool	%I1.3		✓	✓	sensor part at end of conveyor (no)
13	-S3	Bool	%I1.4		✓	✓	pushbutton manual mode conveyor –M1...
14	-S4	Bool	%I1.5		✓	✓	pushbutton manual mode conveyor –M1...
15	-S5	Bool	%I1.6		✓	✓	pushbutton manual mode cylinder -M4 re...
16	-S6	Bool	%I1.7		✓	✓	pushbutton manual mode cylinder -M4 ex...
17	-Q1	Bool	%Q0.0		✓	✓	conveyor motor -M1 forwards fixed speed
18	-Q2	Bool	%Q0.1		✓	✓	conveyor motor -M1 backwards fixed speed
19	-Q3	Bool	%Q0.2		✓	✓	conveyor motor -M1 variable speed
20	-M2	Bool	%Q0.3		✓	✓	cylinder -M4 retract
21	-M3	Bool	%Q0.4		✓	✓	cylinder -M4 extend
22	-P1	Bool	%Q0.5		✓	✓	display „main switch on"
23	-P2	Bool	%Q0.6		✓	✓	display „manual mode"
24	-P3	Bool	%Q0.7		✓	✓	display „automatic mode"
25	-P4	Bool	%Q1.0		✓	✓	display „emergency stop activated"
26	-P5	Bool	%Q1.1		✓	✓	display „automatic mode started"

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

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

(→ PLC programming → Add new block → FC)



Name: Block_1

Language: SCL

Number: 1

Manual [✓ Automatic]

Description:

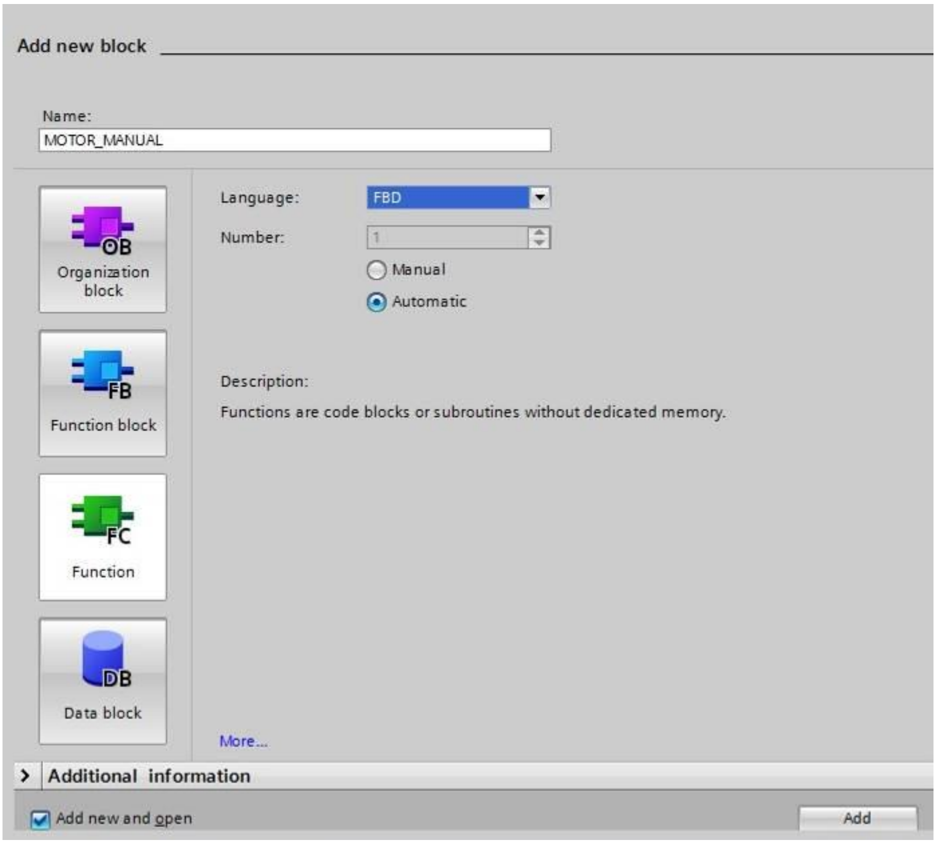
Functions are code blocks or subroutines without dedicated memory.

[✓] Add new and open

Opened project: D:\Automation\032_100_FC-Programming\032_100_FC-Programming

→ Rename your new block to: "MOTOR_MANUAL", set the language to FBD and keep automatic assignment of the number. Select the "Add new and open" check box. You are then taken automatically to your created function block in the project view.Click "Add".

(→ Name: MOTOR_MANUAL→ Language: FBD → Number: Automatic → ✓ Add new and open → Add)



Add new block

Name: MOTOR_MANUAL

Language: FBD

Number: 1

Manual [✓ Automatic]

Description:

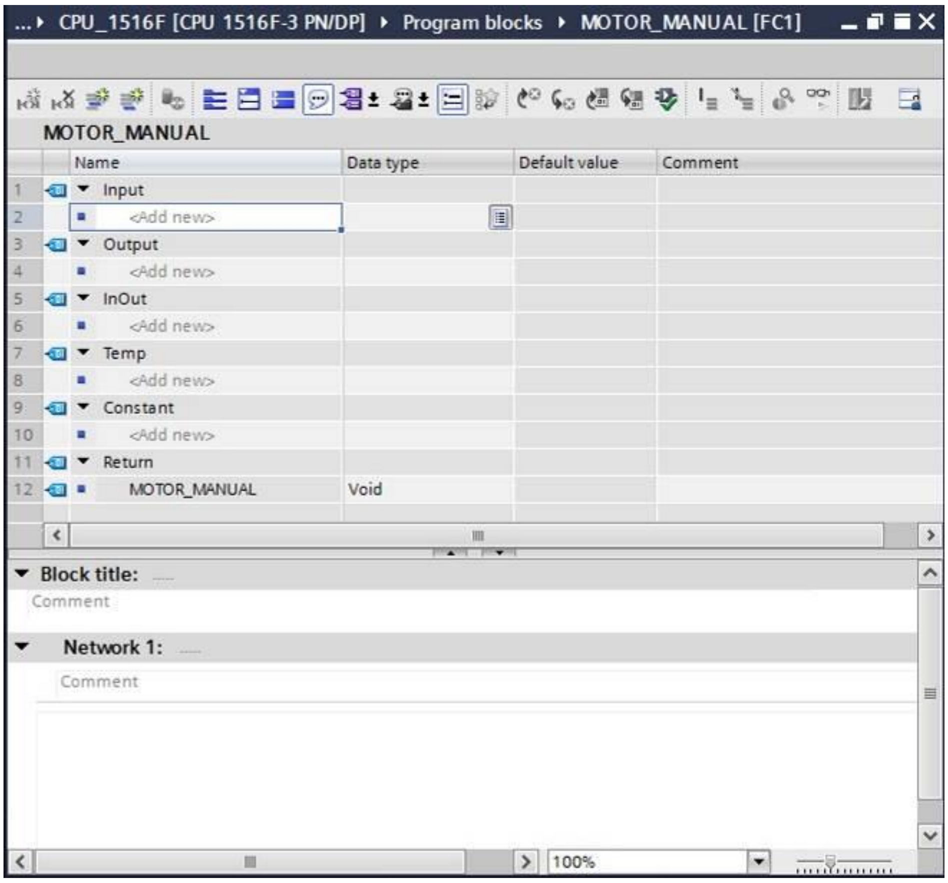
Functions are code blocks or subroutines without dedicated memory.

[✓] Add new and open

7.6 Define the interface of function FC1 "MOTOR_MANUAL"

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

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



...▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]

MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	▼ Input			
2	<Add new>			
3	▼ Output			
4	<Add new>			
5	▼ InOut			
6	<Add new>			
7	▼ Temp			
8	<Add new>			
9	▼ Constant			
10	<Add new>			
11	▼ Return			
12	MOTOR_MANUAL	Void		

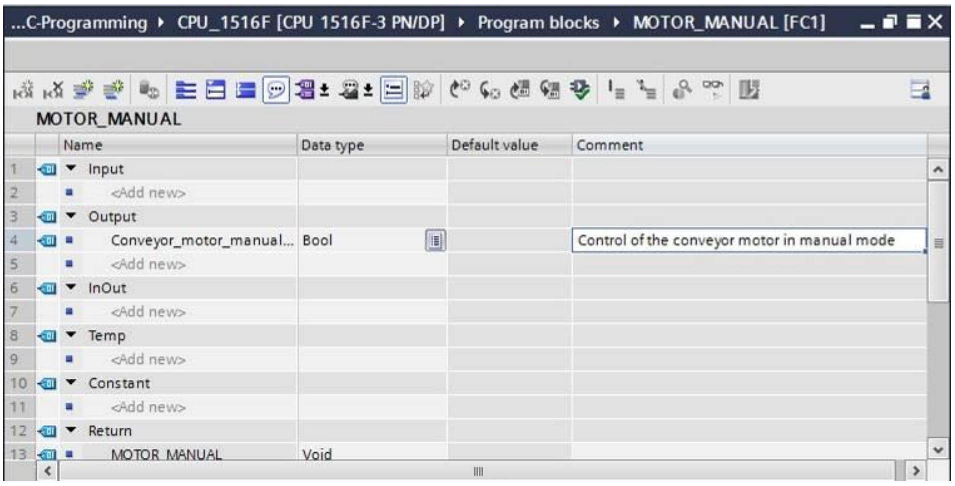
▼ Block title:
Comment

▼ Network 1:
Comment

100%

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

(→ Output: Conveyor_motor_manual_mode → Bool → Control of the conveyor motor in manual mode)


...C-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]

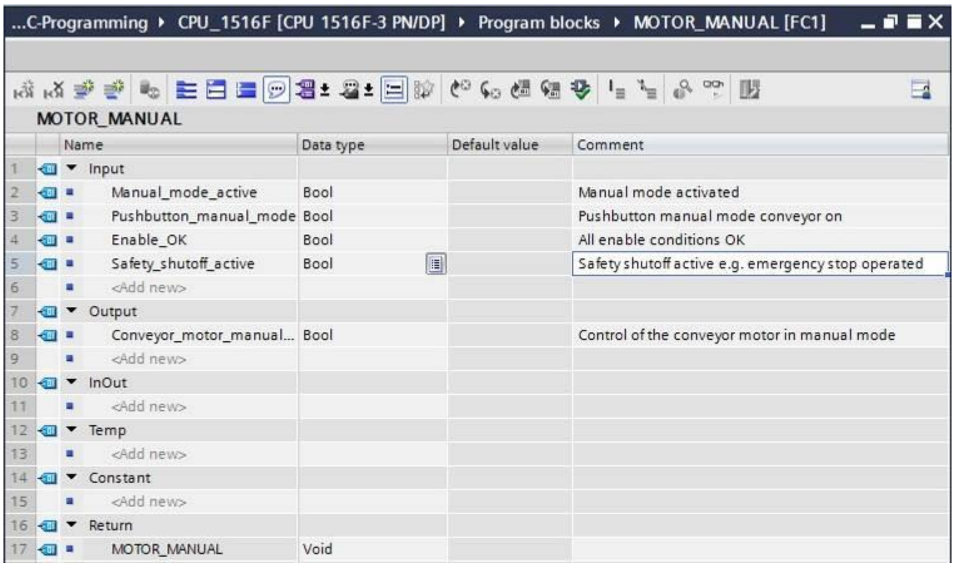
MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	▼ Input			
2	<Add new>			
3	▼ Output			
4	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode
5	<Add new>			
6	▼ InOut			
7	<Add new>			
8	▼ Temp			
9	<Add new>			
10	▼ Constant			
11	<Add new>			
12	▼ Return			
13	MOTOR_MANUAL	Void		

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

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

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

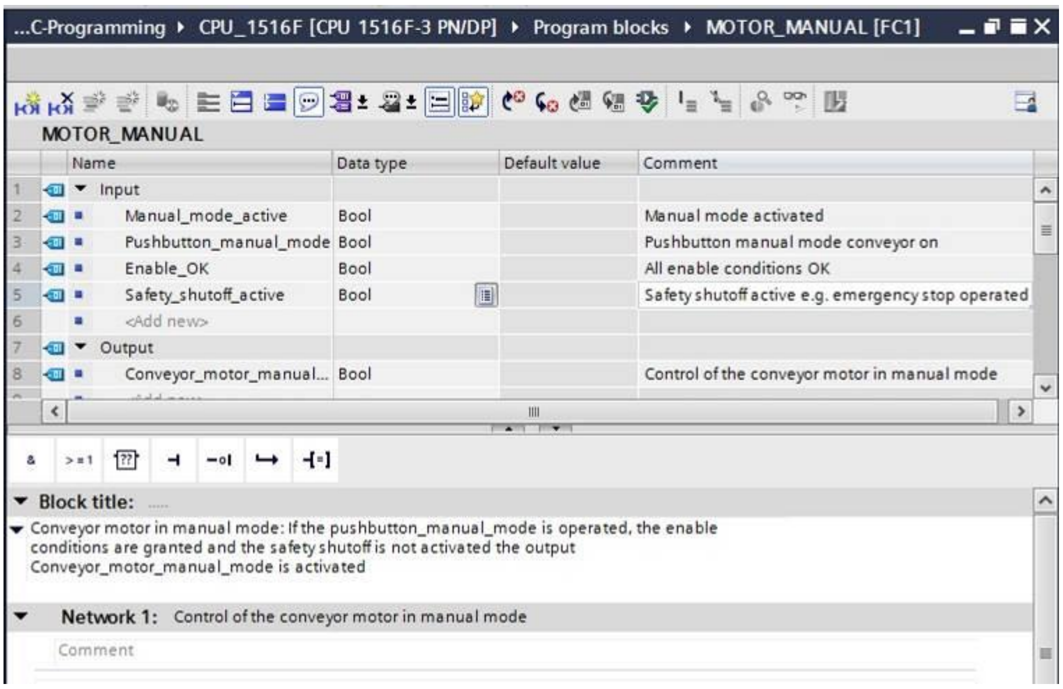

...C-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]

MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	▼ Input			
2	Manual_mode_active	Bool		Manual mode activated
3	Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
4	Enable_OK	Bool		All enable conditions OK
5	Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
6	<Add new>			
7	▼ Output			
8	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode
9	<Add new>			
10	▼ InOut			
11	<Add new>			
12	▼ Temp			
13	<Add new>			
14	▼ Constant			
15	<Add new>			
16	▼ Return			
17	MOTOR_MANUAL	Void		

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

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


...C-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]

MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	▼ Input			
2	Manual_mode_active	Bool		Manual mode activated
3	Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
4	Enable_OK	Bool		All enable conditions OK
5	Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
6	<Add new>			
7	▼ Output			
8	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode

Block title: .....

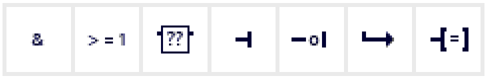
Conveyor motor in manual mode: If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

Network 1: Control of the conveyor motor in manual mode

Comment

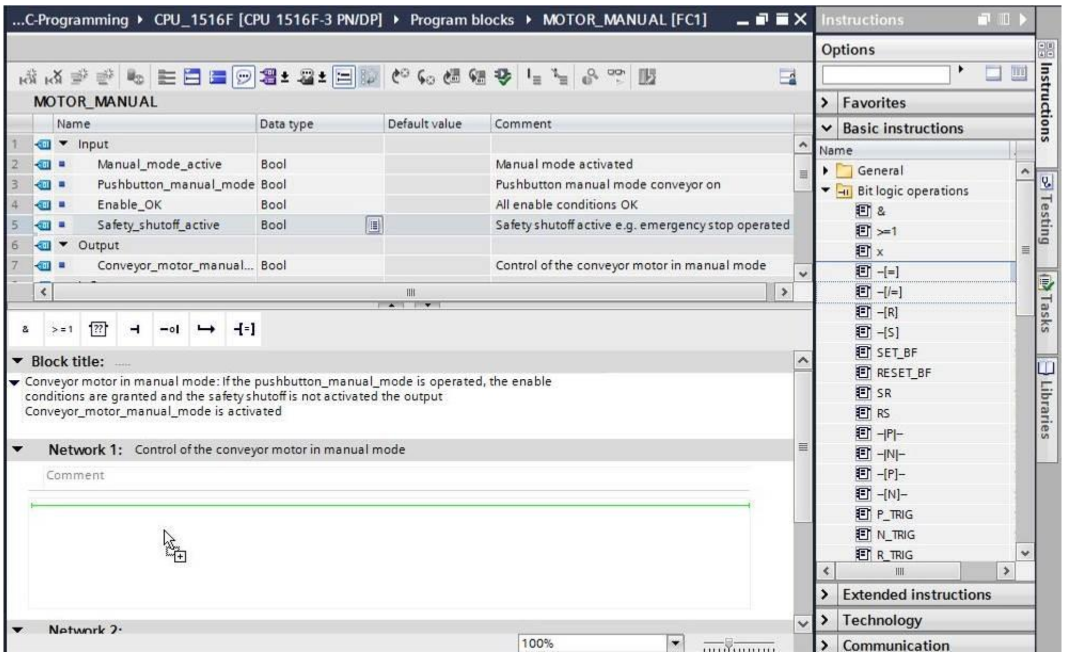
7.7 Program FC1: MOTOR_MANUAL

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



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

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



MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	▼ Input			
2	Manual_mode_active	Bool		Manual mode activated
3	Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
4	Enable_OK	Bool		All enable conditions OK
5	Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
6	▼ Output			
7	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode

Block title:

Conveyor motor in manual mode: If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

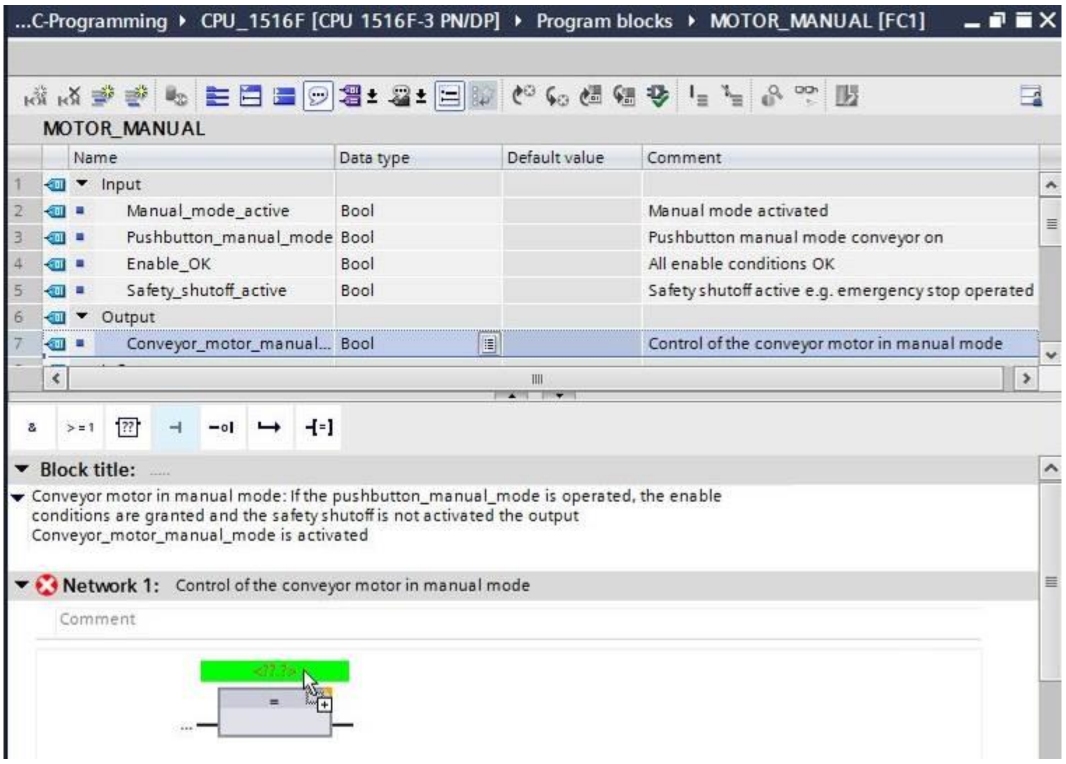
Network 1: Control of the conveyor motor in manual mode

Comment

Network 2:

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

(→ Conveyor_motor_manual_mode)



...C-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]

MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	▼ Input			
2	Manual_mode_active	Bool		Manual mode activated
3	Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
4	Enable_OK	Bool		All enable conditions OK
5	Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
6	▼ Output			
7	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode

Block title:

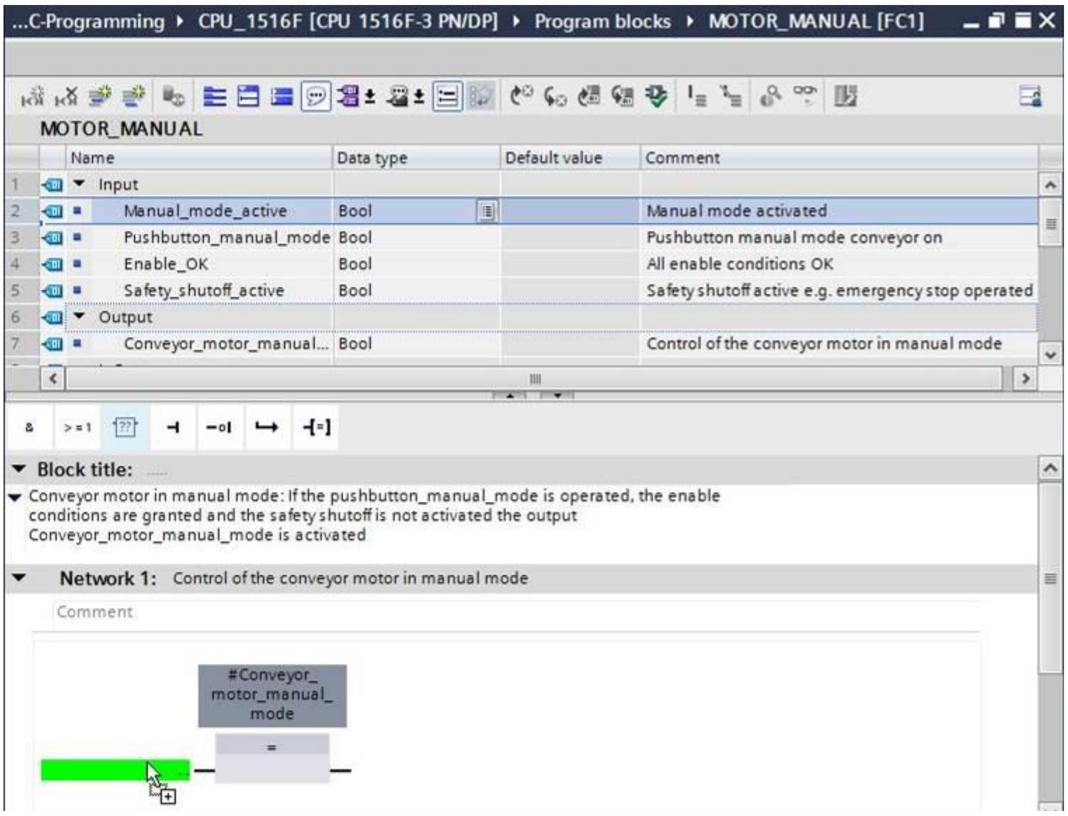
Conveyor motor in manual mode: If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

Network 1: Control of the conveyor motor in manual mode

Comment

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

(→ Manual_mode_active)


...C-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]

MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	▼ Input			
2	Manual_mode_active	Bool		Manual mode activated
3	Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
4	Enable_OK	Bool		All enable conditions OK
5	Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
6	▼ Output			
7	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode

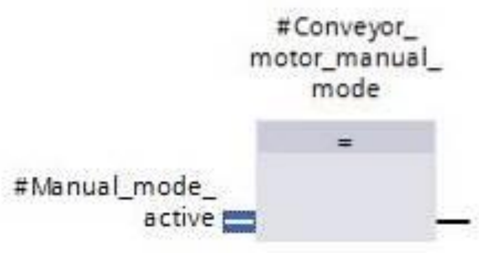
Block title:

Conveyor motor in manual mode: If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

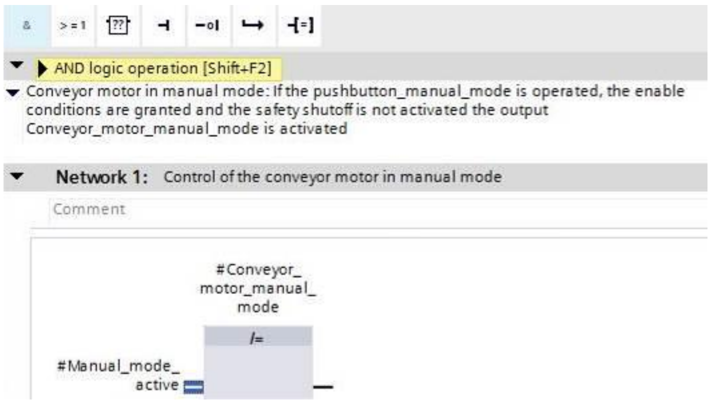
Network 1: Control of the conveyor motor in manual mode

Comment

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



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



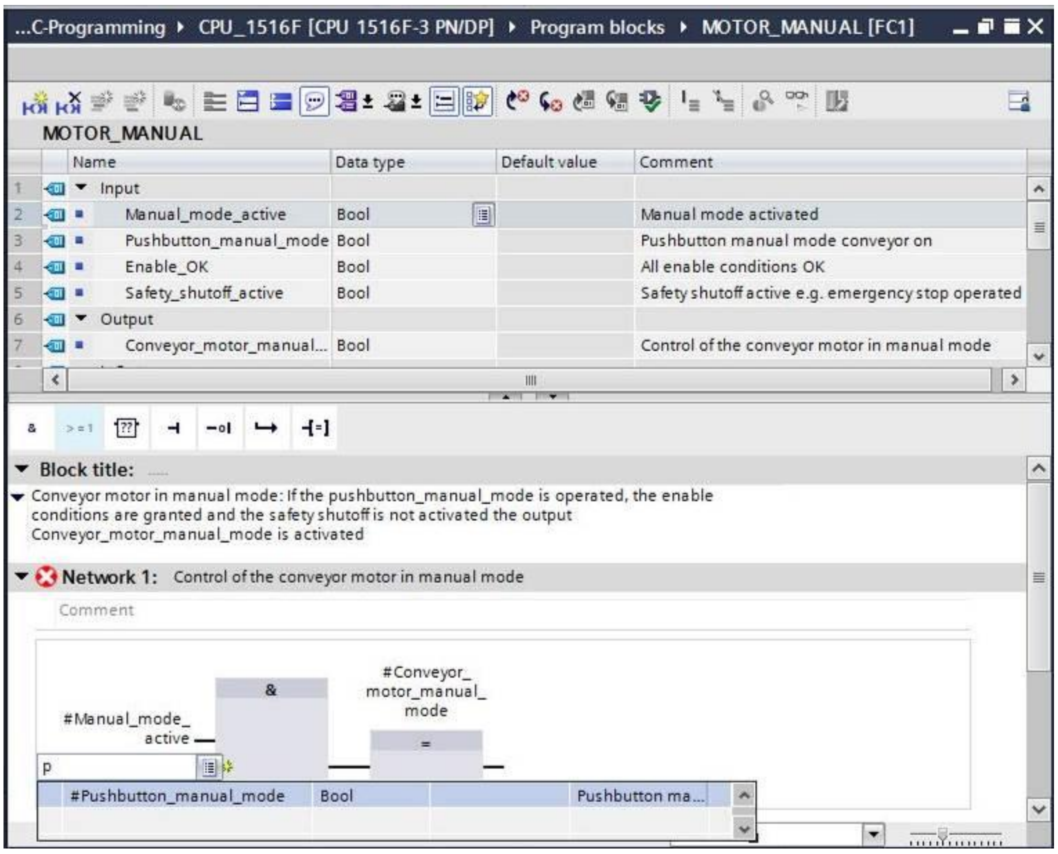
Conveyor motor in manual mode: If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

Network 1: Control of the conveyor motor in manual mode

Comment

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

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


...C-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]
& >=1 ?? -| -o| ↦ -[=]
&
#Manual_mode_ active
P
#Conveyor_ motor_manual_ mode
=

MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	▼ Input			
2	Manual_mode_active	Bool		Manual mode activated
3	Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
4	Enable_OK	Bool		All enable conditions OK
5	Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
6	▼ Output			
7	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode

Block title:

Conveyor motor in manual mode: If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

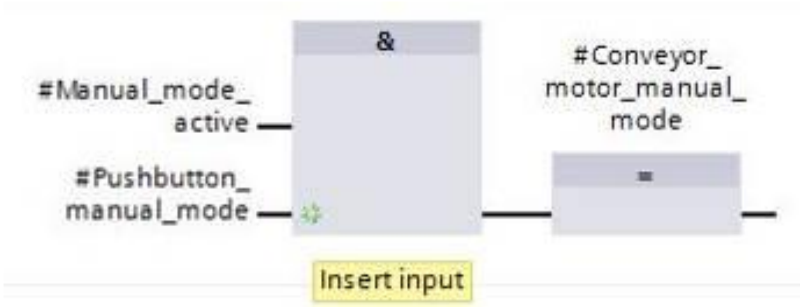
Network 1: Control of the conveyor motor in manual mode

Comment

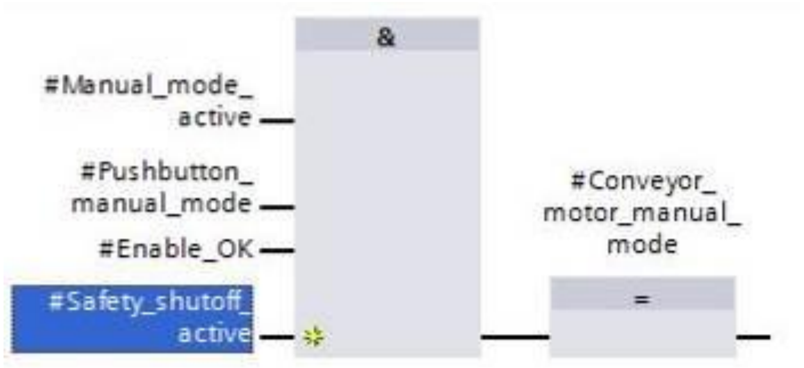
#Pushbutton_manual_mode	Bool		Pushbutton ma...

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

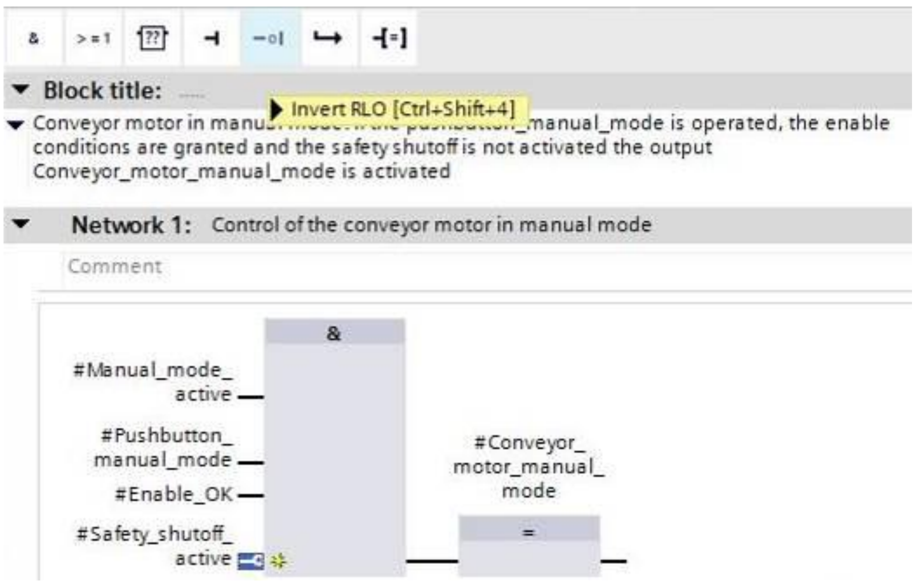
→ To ensure that the output can only be controlled when the enable conditions are met and the safety shutoff is not active, the #Enable_OK and #Safety_shutoff_active input tags are logically combined with the AND logic operation. To do this, click twice on the yellow star of your AND block to add two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



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



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



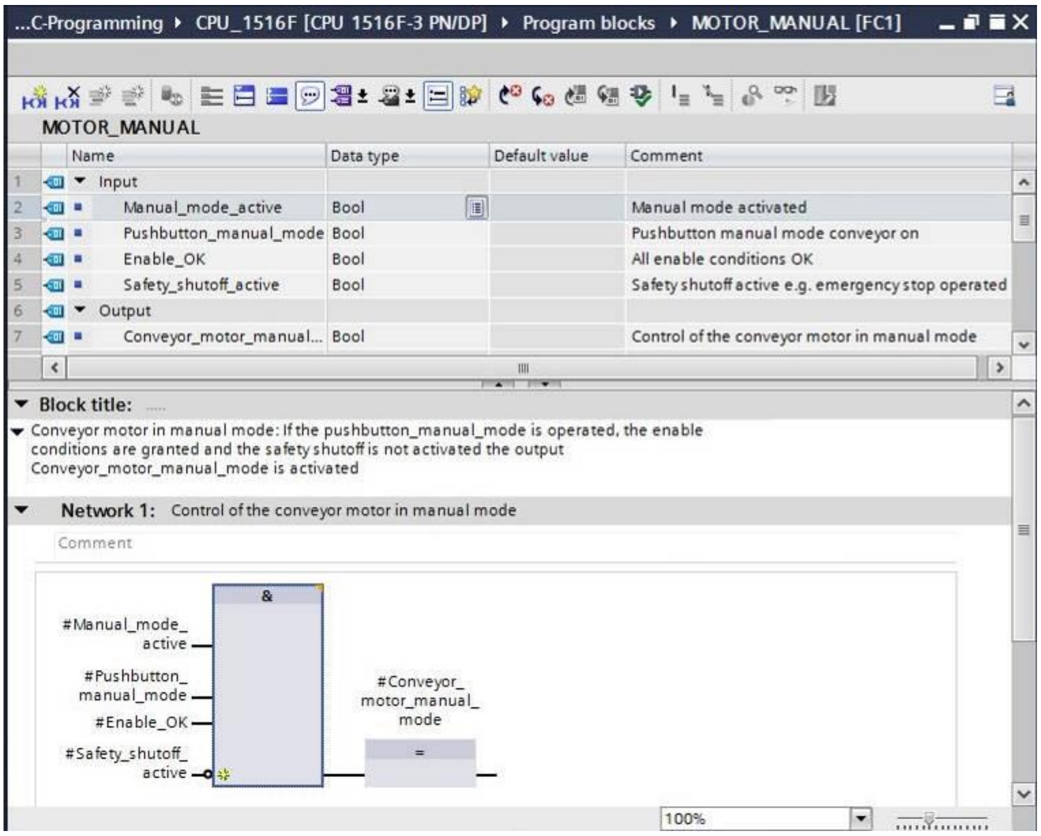
Block title:

Conveyor motor in manual mode. If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

Network 1: Control of the conveyor motor in manual mode

Comment

→ Do not forget to click Save project. The finished function "MOTOR_MANUAL" [FC1] in FBD is shown below.


...C-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]

MOTOR_MANUAL

	Name	Data type	Default value	Comment
1	▼ Input			
2	Manual_mode_active	Bool		Manual mode activated
3	Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
4	Enable_OK	Bool		All enable conditions OK
5	Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
6	▼ Output			
7	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode

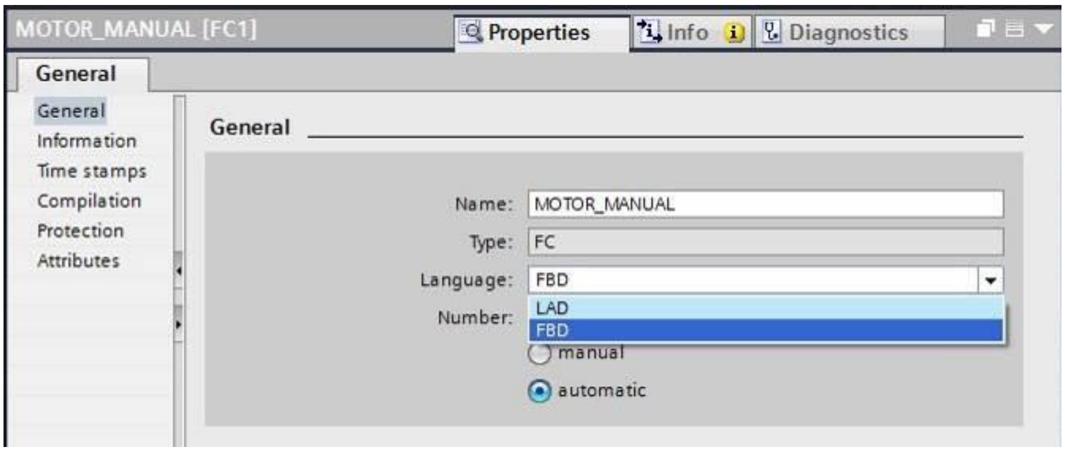
Block title:

Conveyor motor in manual mode: If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

Network 1: Control of the conveyor motor in manual mode

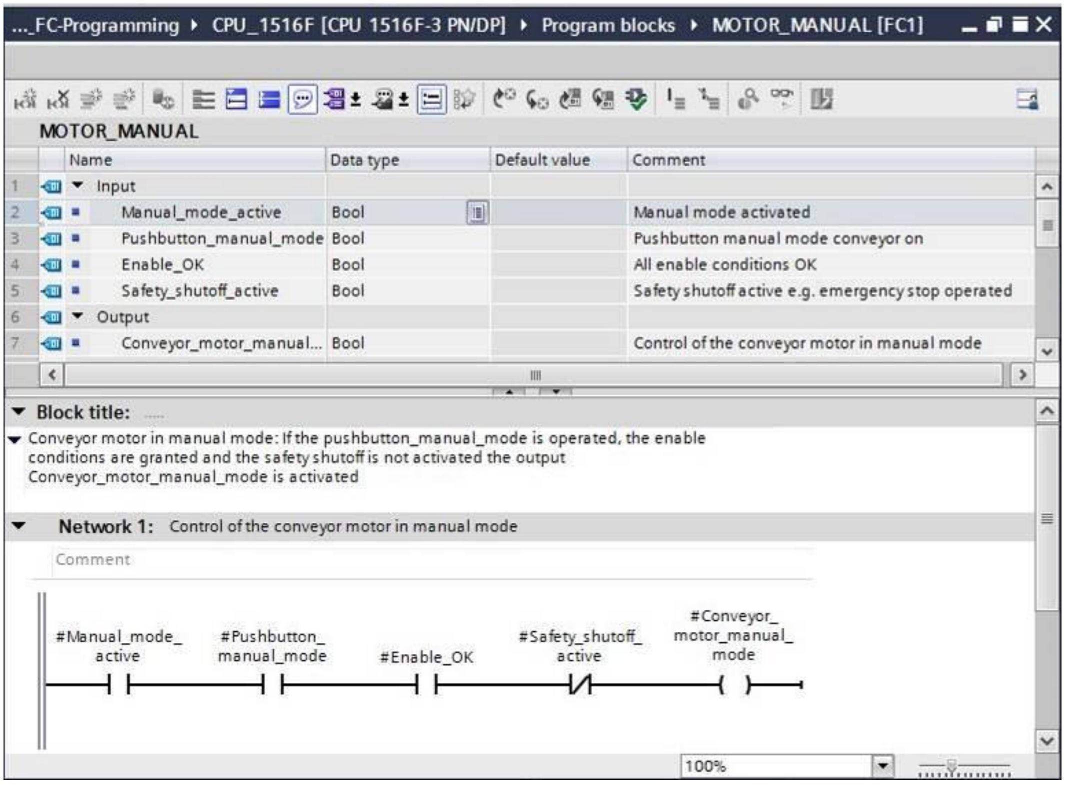
Comment

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



Name: MOTOR_MANUAL
Type: FC
Language: FBD
LAD
FBD
Number:
manual
automatic

→ The program has the following appearance in LAD.


..._FC-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]
MOTOR_MANUAL
100%

	Name	Data type	Default value	Comment
1	▼ Input			
2	Manual_mode_active	Bool		Manual mode activated
3	Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
4	Enable_OK	Bool		All enable conditions OK
5	Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
6	▼ Output			
7	Conveyor_motor_manual...	Bool		Control of the conveyor motor in manual mode

Block title:

Conveyor motor in manual mode: If the pushbutton_manual_mode is operated, the enable conditions are granted and the safety shutoff is not activated the output Conveyor_motor_manual_mode is activated

Network 1: Control of the conveyor motor in manual mode

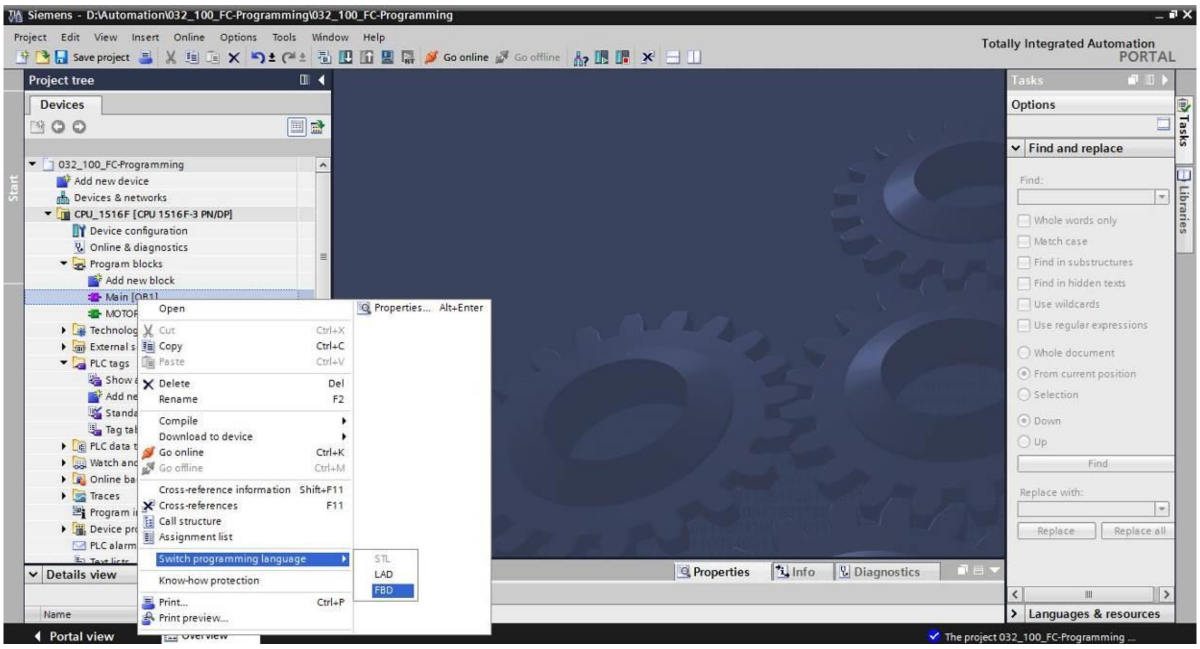
Comment

#Manual_mode_active — #Pushbutton_manual_mode — #Enable_OK — #Safety_shutoff_active — #Conveyor_motor_manual_mode

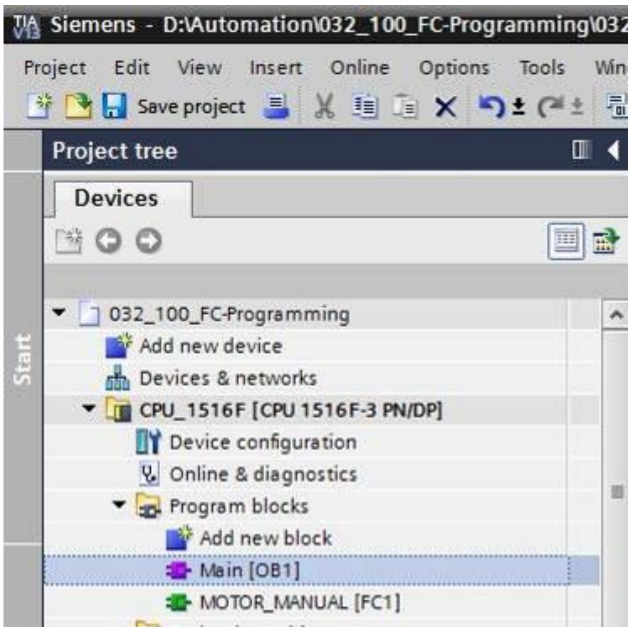
7.8 Program organization block OB1 – Control of the forward belt tracking in manual mode

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

(→ CPU_1516F[CPU 1516F-3 PN/DP → Program blocks → Main [OB1] → Switch programming language → FBD)

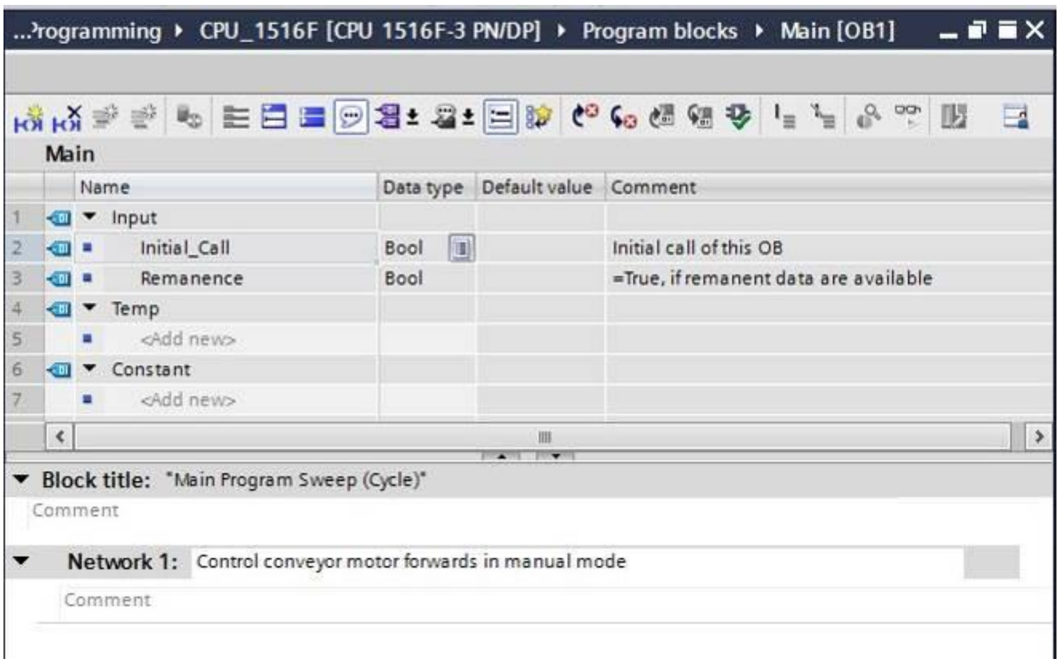


→ Open the "Main [OB1]" organization block with a double-click.



→ Assign Network 1 the name "Control conveyor tracking forward in manual/jog mode"

(→ Network 1:... →Control conveyor motor forwards in manual mode)



...Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ Main [OB1]

Main

	Name	Data type	Default value	Comment
1	▼ Input			
2	Initial_Call	Bool		Initial call of this OB
3	Remanence	Bool		=True, if remanent data are available
4	▼ Temp			
5	<Add new>			
6	▼ Constant			
7	<Add new>			

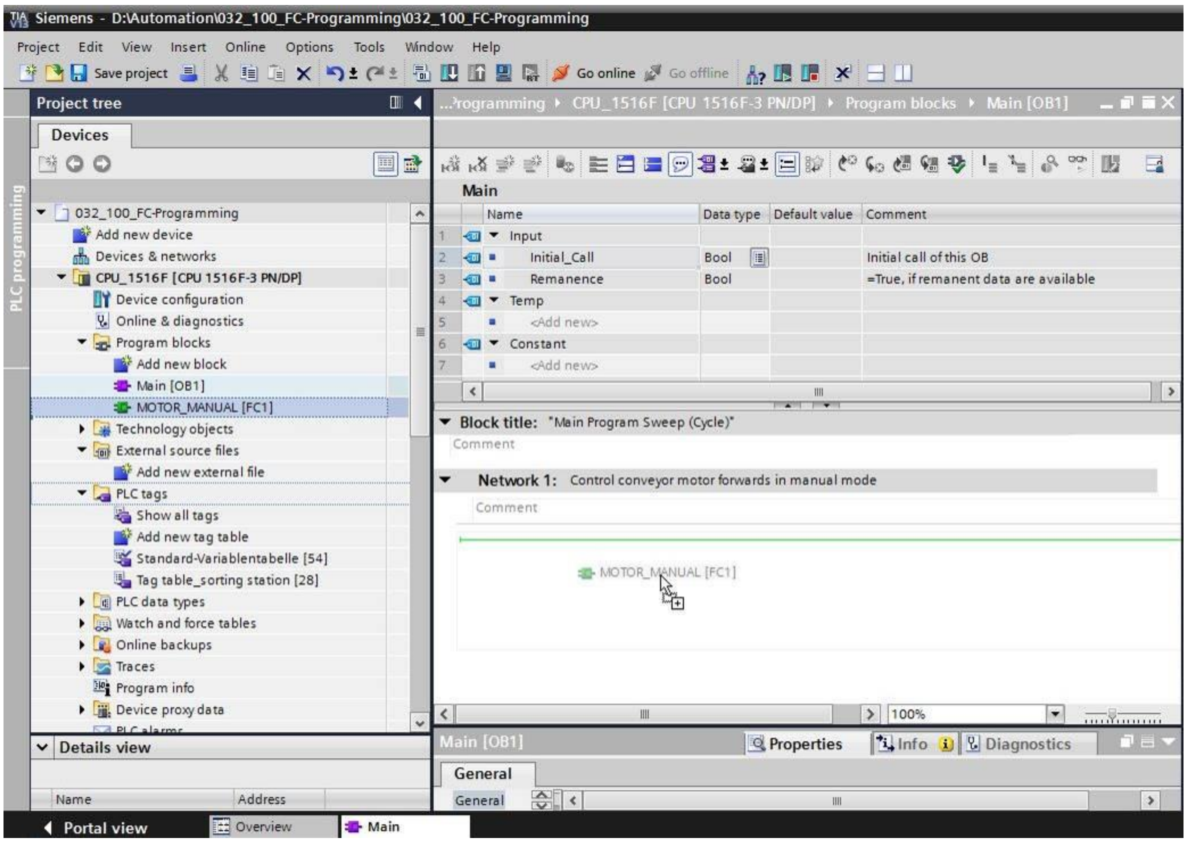
Block title: "Main Program Sweep (Cycle)"

Comment

Network 1: Control conveyor motor forwards in manual mode

Comment

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



Siemens - D:\Automation\032_100_FC-Programming\032_100_FC-Programming

Project tree – Devices

- 032_100_FC-Programming
  - Add new device
  - Devices & networks
  - CPU_1516F [CPU 1516F-3 PN/DP]
    - Device configuration
    - Online & diagnostics
    - Program blocks
      - Add new block
      - Main [OB1]
      - MOTOR_MANUAL [FC1]
    - Technology objects
    - External source files
      - Add new external file
    - PLC tags
      - Show all tags
      - Add new tag table
      - Standard-Variablentabelle [54]
      - Tag table_sorting station [28]
    - PLC data types
    - Watch and force tables
    - Online backups
    - Traces
    - Program info
    - Device proxy data

Main

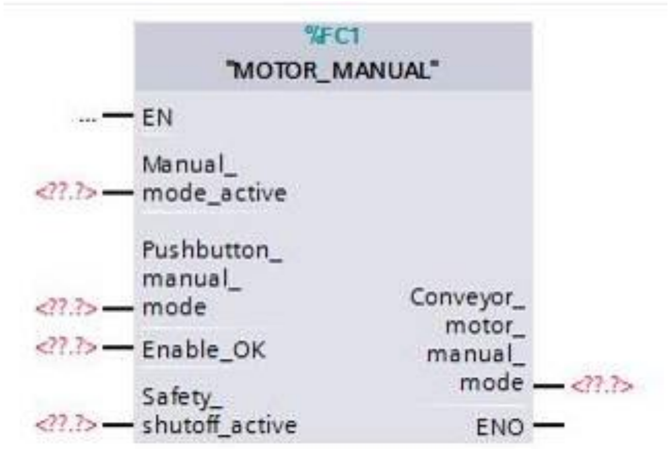
	Name	Data type	Default value	Comment
1	▼ Input			
2	Initial_Call	Bool		Initial call of this OB
3	Remanence	Bool		=True, if remanent data are available
4	▼ Temp			
5	<Add new>			
6	▼ Constant			
7	<Add new>			

Block title: "Main Program Sweep (Cycle)"

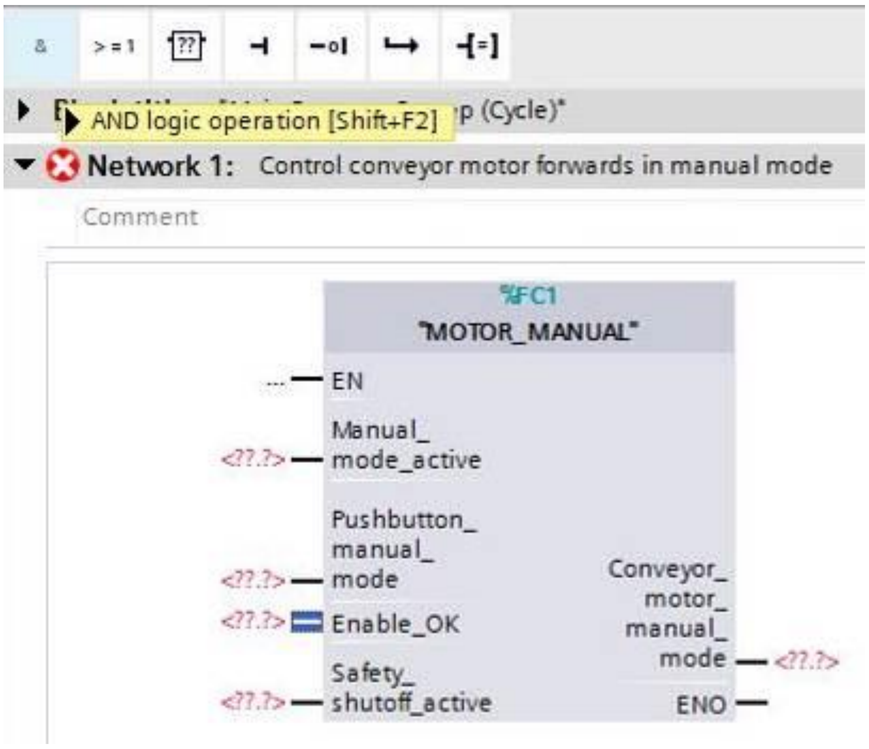
Network 1: Control conveyor motor forwards in manual mode

MOTOR_MANUAL [FC1]

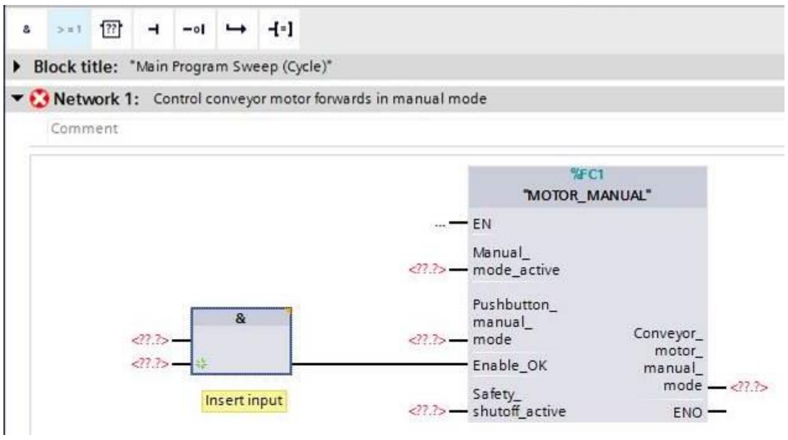
→ A block with the interface you defined and connections EN and ENO are inserted in Network 1.



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



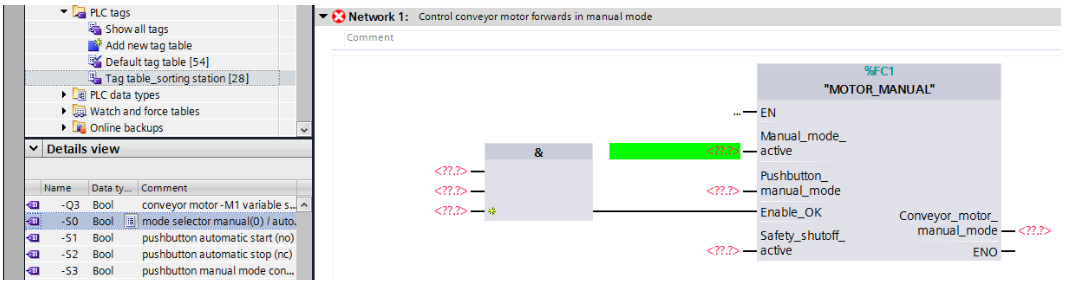
→ Click the yellow star ✳ of the AND block to add another input (→ ✳).



&
>=1
??
Block title: "Main Program Sweep (Cycle)"
Network 1: Control conveyor motor forwards in manual mode
Comment
%FC1
"MOTOR_MANUAL"
EN
Manual_mode_active
Pushbutton_manual_mode
Enable_OK
Safety_shutoff_active
Conveyor_motor_manual_mode
ENO
<??.?>
Insert input

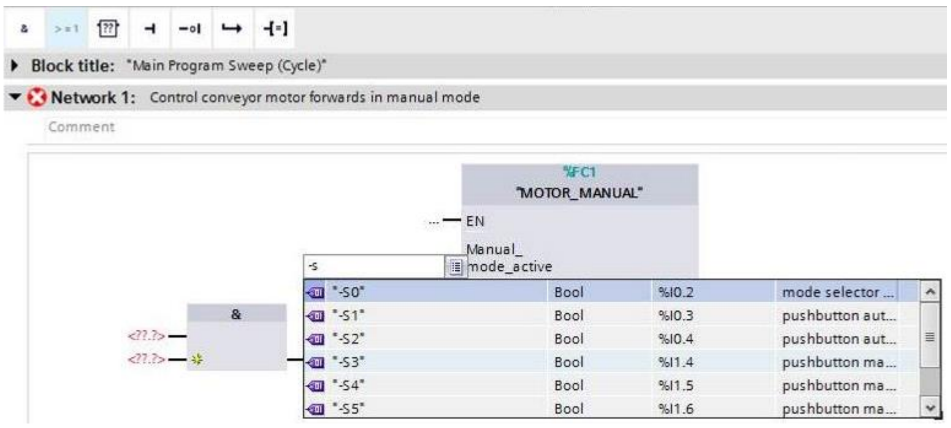
→ To connect the block to the global tags from "Tag_table_sorting_station", we have two options:

→ Either select the "Tag_table_sorting_station" in the project tree and use drag-and-drop to move the desired global tag from the Details view to the interface of FC1 (→ Tag_table_sorting_station → Details view. → -SO2 → Manual_mode_active)



Name	Data ty...	Comment
-Q3	Bool	conveyor motor -M1 variable s...
-S0	Bool	mode selector manual(0) / auto...
-S1	Bool	pushbutton automatic start (no)
-S2	Bool	pushbutton automatic stop (nc)
-S3	Bool	pushbutton manual mode con...

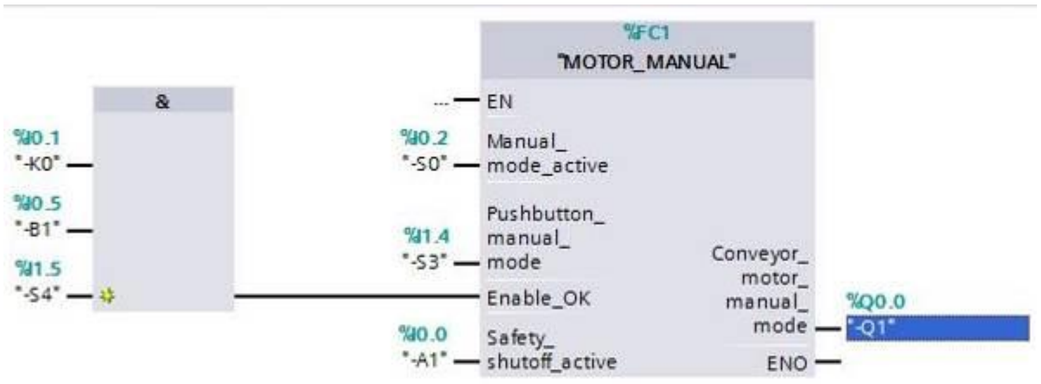
→ Or, enter the starting letters (e.g. "-S") of the desired global tag for <??.?> and select the global input tag "-S0" (%I0.2) from the displayed list (→ Manual_mode_active → -S → -S0).



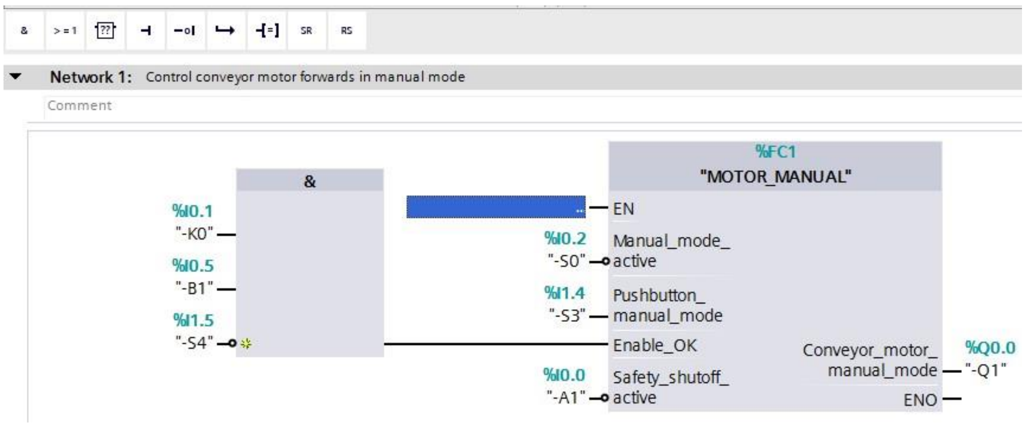
&
>=1
??
Block title: "Main Program Sweep (Cycle)"
Network 1: Control conveyor motor forwards in manual mode
Comment
%FC1
"MOTOR_MANUAL"
EN
Manual_mode_active
-S
<??.?>

"-S0"	Bool	%I0.2	mode selector ...
"-S1"	Bool	%I0.3	pushbutton aut...
"-S2"	Bool	%I0.4	pushbutton aut...
"-S3"	Bool	%I1.4	pushbutton ma...
"-S4"	Bool	%I1.5	pushbutton ma...
"-S5"	Bool	%I1.6	pushbutton ma...

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

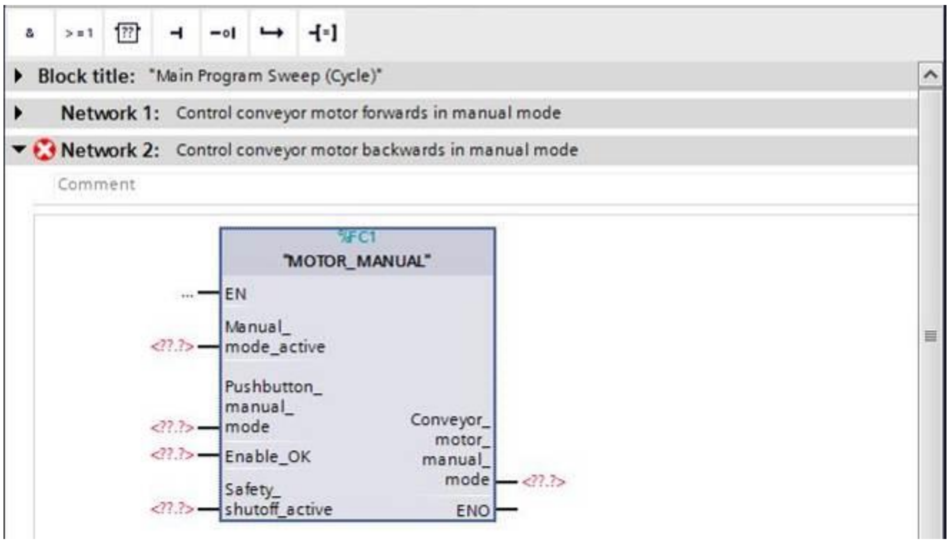


→ Negate the querying of input tags "-S0", "-S4" and "-A1" by selecting them and clicking –o| (→ -S0 → –o| → -S4 → –o| → -A1 → –o|).

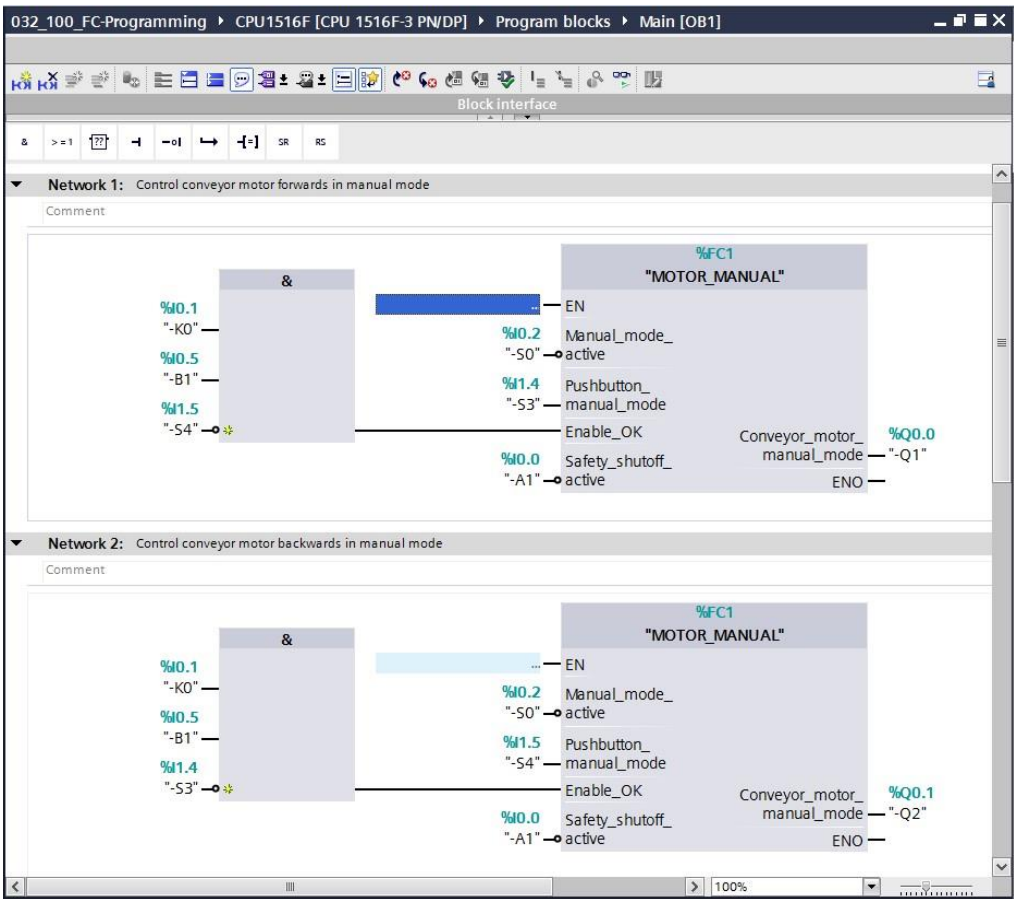


7.9 Program organization block OB1 – Control of the backward belt tracking in manual mode

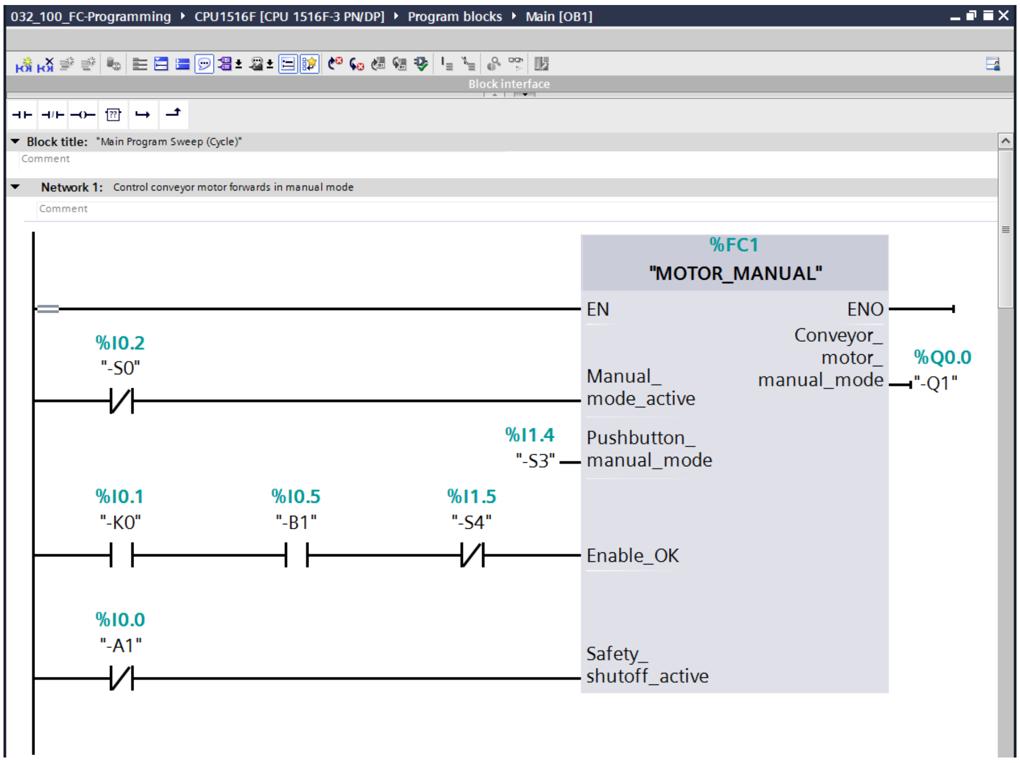
→ Assign Network 2 the name "Control conveyor motor backwards in manual mode" and insert your "MOTOR_MANUAL [FC1]" function using drag-and-drop, as you did previously in Network 1.


Block title: "Main Program Sweep (Cycle)"
Network 1: Control conveyor motor forwards in manual mode
Network 2: Control conveyor motor backwards in manual mode
Comment
%FC1
"MOTOR_MANUAL"
EN
Manual_mode_active
Pushbutton_manual_mode
Enable_OK
Safety_shutoff_active
Conveyor_motor_manual_mode
ENO
<??.?>

→ Connect your function as shown here. You obtain the following result in the FBD (Function Block Diagram) programming language.


032_100_FC-Programming ▸ CPU1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ Main [OB1]
Block interface
Network 1: Control conveyor motor forwards in manual mode
Comment
&
%I0.1 "-K0"
%I0.5 "-B1"
%I1.5 "-S4"
%FC1
"MOTOR_MANUAL"
EN
%I0.2 "-S0" Manual_mode_active
%I1.4 "-S3" Pushbutton_manual_mode
Enable_OK
%I0.0 "-A1" Safety_shutoff_active
Conveyor_motor_manual_mode %Q0.0 "-Q1"
ENO
Network 2: Control conveyor motor backwards in manual mode
Comment
&
%I0.1 "-K0"
%I0.5 "-B1"
%I1.4 "-S3"
%FC1
"MOTOR_MANUAL"
EN
%I0.2 "-S0" Manual_mode_active
%I1.5 "-S4" Pushbutton_manual_mode
Enable_OK
%I0.0 "-A1" Safety_shutoff_active
Conveyor_motor_manual_mode %Q0.1 "-Q2"
ENO
100%

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


032_100_FC-Programming ▸ CPU1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ Main [OB1]

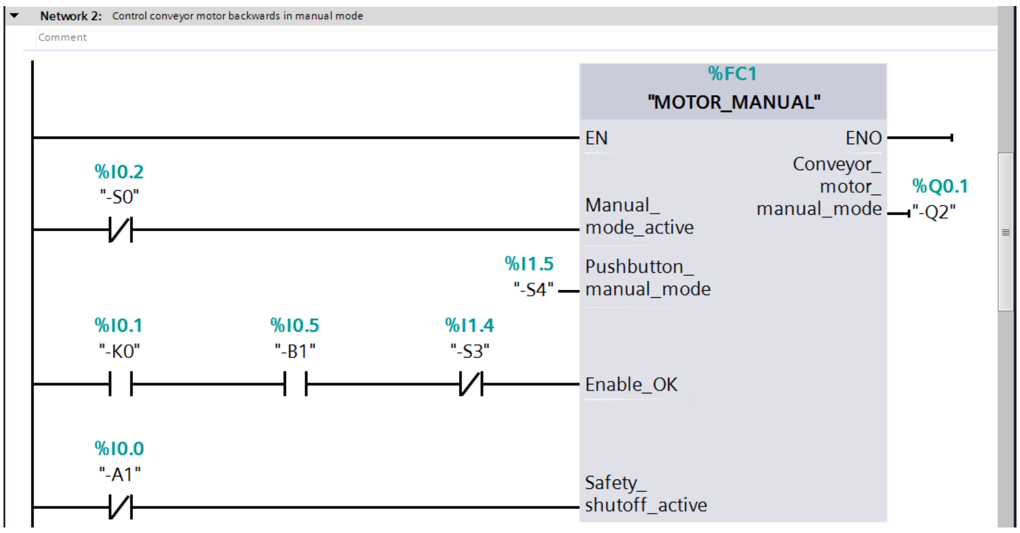
Block interface

Block title: "Main Program Sweep (Cycle)"

Comment

Network 1: Control conveyor motor forwards in manual mode

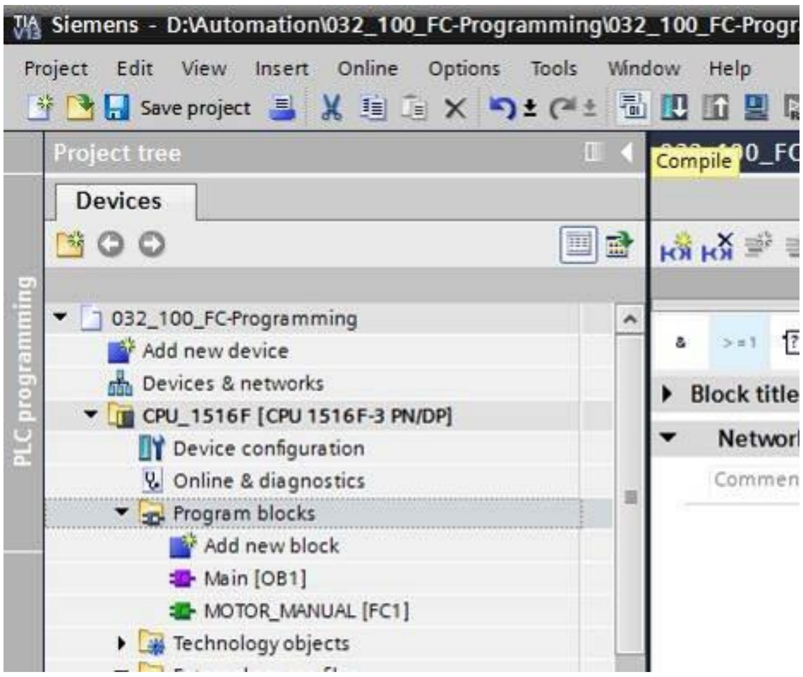
Comment


Network 2: Control conveyor motor backwards in manual mode

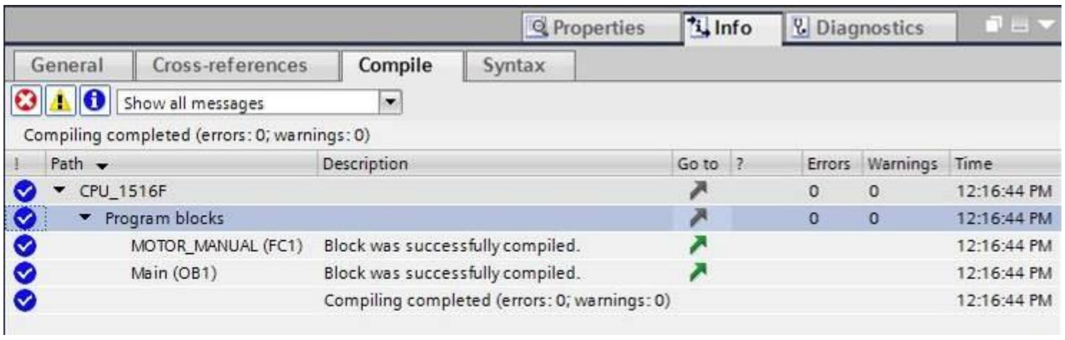
Comment

7.10 Save and compile the program

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



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

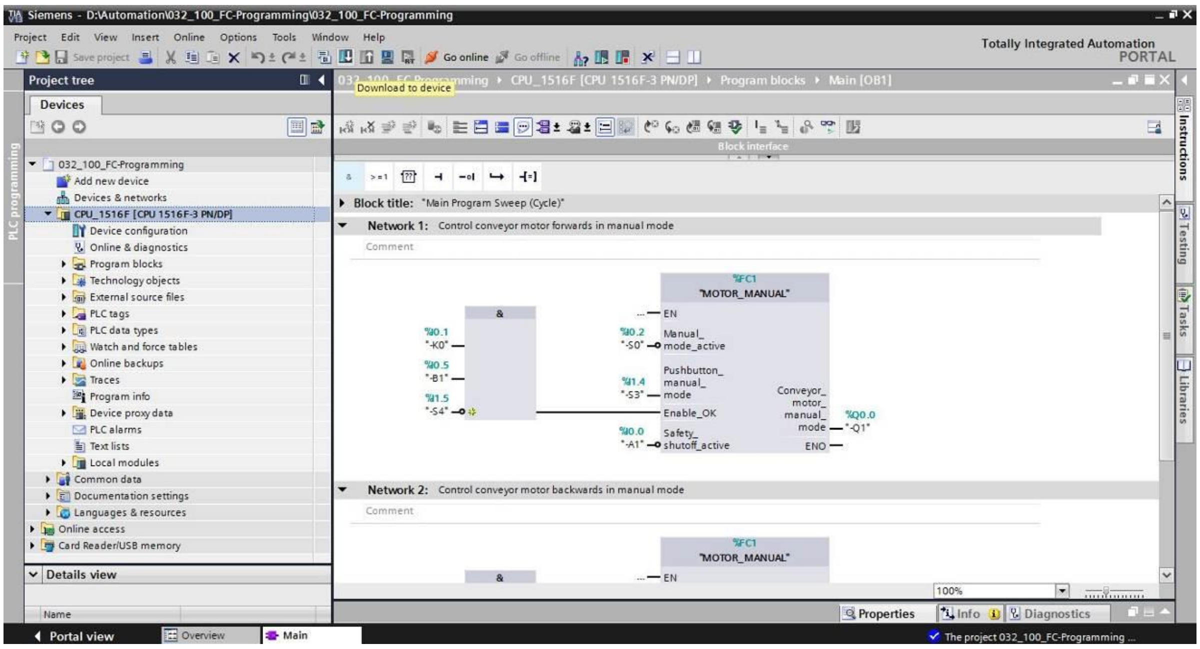


Compiling completed (errors: 0; warnings: 0)

!	Path	Description	Go to	?	Errors	Warnings	Time
✓	CPU_1516F		↗		0	0	12:16:44 PM
✓	Program blocks		↗		0	0	12:16:44 PM
✓	MOTOR_MANUAL (FC1)	Block was successfully compiled.	↗				12:16:44 PM
✓	Main (OB1)	Block was successfully compiled.	↗				12:16:44 PM
✓		Compiling completed (errors: 0; warnings: 0)					12:16:44 PM

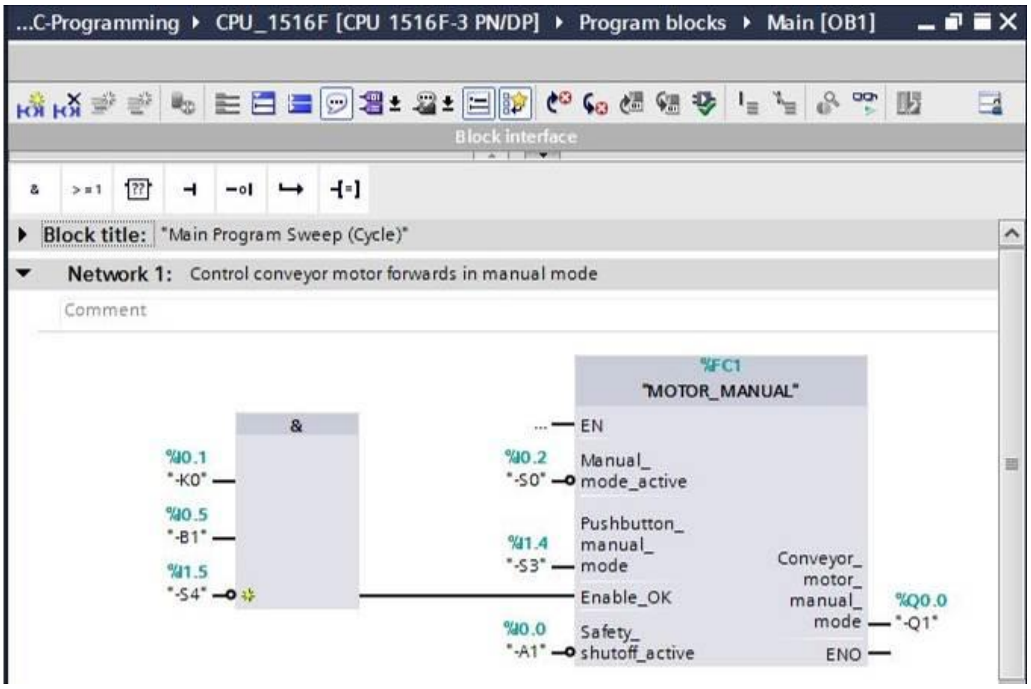
7.11 Download the program

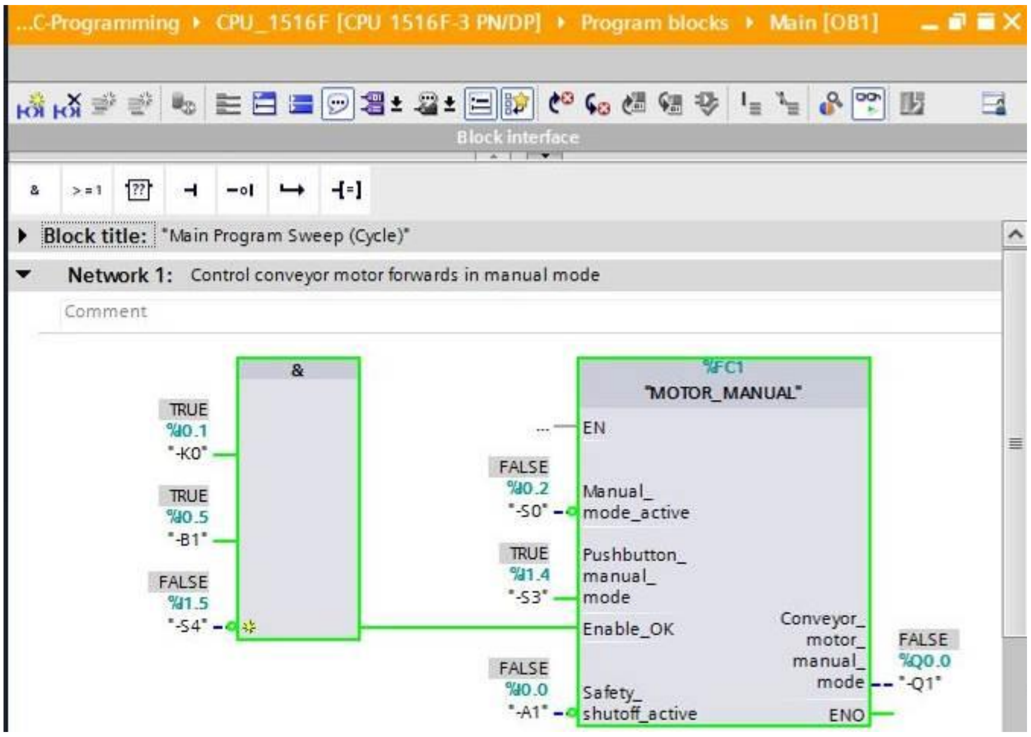
→ After successful compilation, the complete controller with the created program, as previously described in the modules for hardware configuration, can be downloaded (→).



7.12 Monitor program blocks

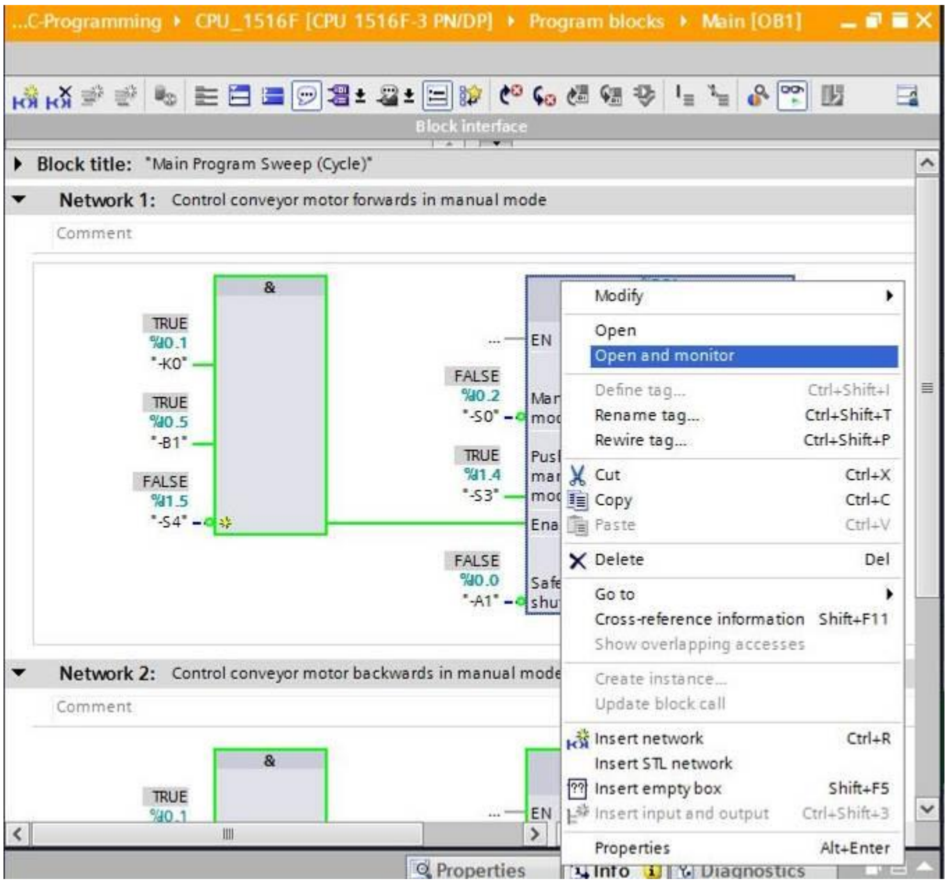
→ The desired block must be open for monitoring the downloaded program. The monitoring can be activated/deactivated by clicking the icon. (→ Main [OB1] →)

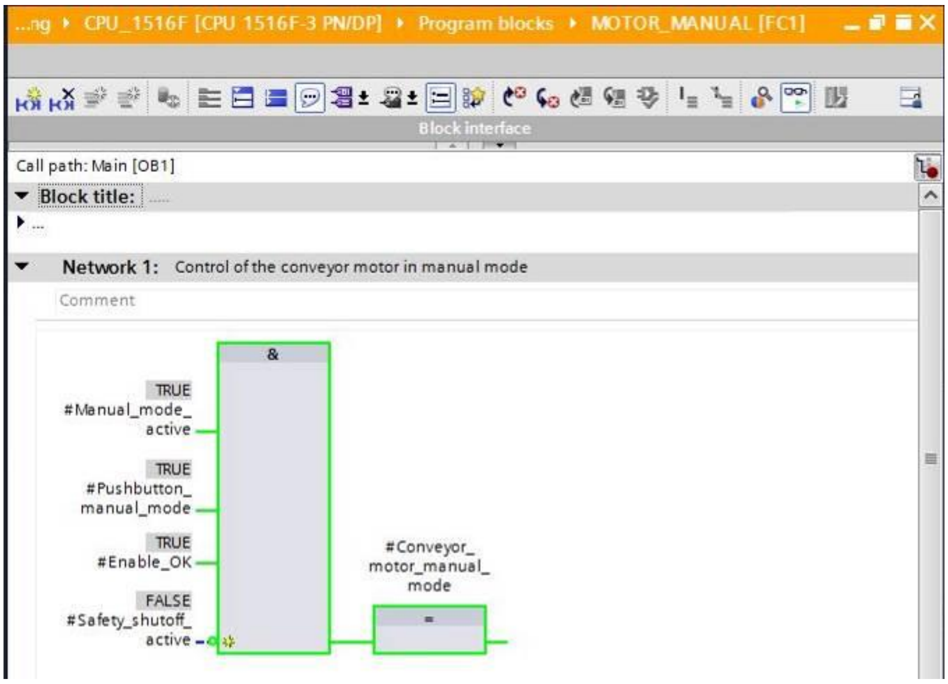

...C-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ Main [OB1]
Block interface
Block title: "Main Program Sweep (Cycle)"
Network 1: Control conveyor motor forwards in manual mode
Comment
%FC1
"MOTOR_MANUAL"
&
%I0.1
"-K0"
%I0.5
"-B1"
%I1.5
"-S4"
EN
%I0.2
"-S0"
Manual_mode_active
%I1.4
"-S3"
Pushbutton_manual_mode
Enable_OK
%I0.0
"-A1"
Safety_shutoff_active
Conveyor_motor_manual_mode
%Q0.0
"-Q1"
ENO



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

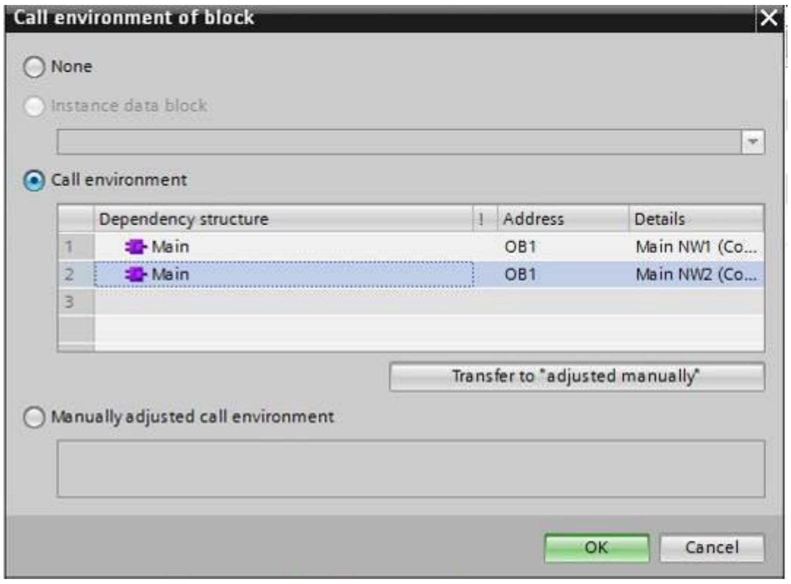
→ The "MOTOR_MANUAL" [FC1] function called in the "Main [OB1]" organization block can be selected directly for "Open and monitor" after right-clicking (→ "MOTOR_MANUAL" [FC1] → Open and monitor).


...C-Programming ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ Main [OB1]
Block interface
Block title: "Main Program Sweep (Cycle)"
Network 1: Control conveyor motor forwards in manual mode
Comment
TRUE %I0.1 "-K0"
TRUE %I0.5 "-B1"
FALSE %I1.5 "-S4"
FALSE %I0.2 "-S0"
TRUE %I1.4 "-S3"
FALSE %I0.0 "-A1"
EN
Modify
Open
Open and monitor
Define tag... Ctrl+Shift+I
Rename tag... Ctrl+Shift+T
Rewire tag... Ctrl+Shift+P
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Go to
Cross-reference information Shift+F11
Show overlapping accesses
Create instance...
Update block call
Insert network Ctrl+R
Insert STL network
Insert empty box Shift+F5
Insert input and output Ctrl+Shift+3
Properties Alt+Enter
Network 2: Control conveyor motor backwards in manual mode
Properties
Info
Diagnostics


...ng ▸ CPU_1516F [CPU 1516F-3 PN/DP] ▸ Program blocks ▸ MOTOR_MANUAL [FC1]
Block interface
Call path: Main [OB1]
Block title:
Network 1: Control of the conveyor motor in manual mode
Comment
&
TRUE #Manual_mode_active
TRUE #Pushbutton_manual_mode
TRUE #Enable_OK
FALSE #Safety_shutoff_active
#Conveyor_motor_manual_mode
=

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

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



Call environment of block

None

Instance data block

[✓ Call environment]

	Dependency structure	!	Address	Details
1	Main		OB1	Main NW1 (Co...
2	Main		OB1	Main NW2 (Co...
3				

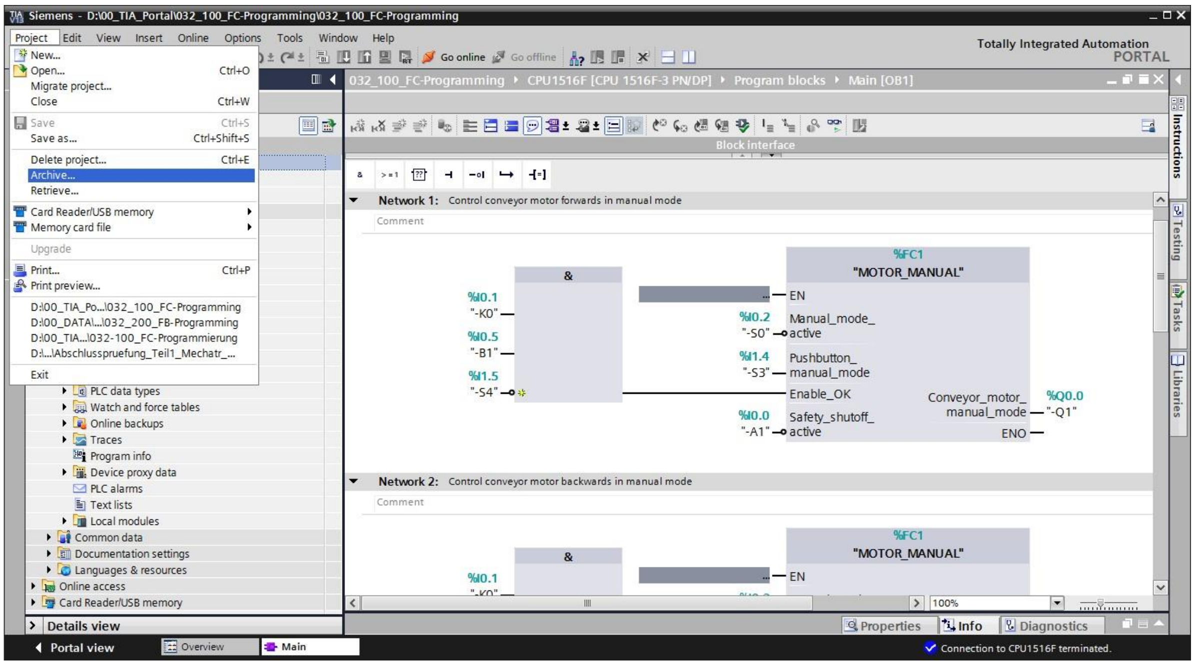
Transfer to "adjusted manually"

Manually adjusted call environment

OK Cancel

7.13 Archive the project

→ As the final step, we want to archive the complete project. Select the → "Archive ..." command in the → "Project" menu. Select a folder where you want to archive your project and save it with the file type "TIA Portal project archive". (→ Project → Archive → TIA Portal project archive → 032-100_FCProgramming.… → Save)


Siemens - D:\00_TIA_Portal\032_100_FC-Programming\032_100_FC-Programming
Project Edit View Insert Online Options Tools Window Help
Totally Integrated Automation PORTAL
New...
Open... Ctrl+O
Migrate project...
Close Ctrl+W
Save Ctrl+S
Save as... Ctrl+Shift+S
Delete project... Ctrl+E
Archive...
Retrieve...
Card Reader/USB memory
Memory card file
Upgrade
Print... Ctrl+P
Print preview...
D:\00_TIA_Po...\032_100_FC-Programming
D:\00_DATA\...\032_200_FB-Programming
D:\00_TIA...\032-100_FC-Programmierung
D:\...\Abschlusspruefung_Teil1_Mechatr_...
Exit
032_100_FC-Programming › CPU1516F [CPU 1516F-3 PN/DP] › Program blocks › Main [OB1]
Block interface
Network 1: Control conveyor motor forwards in manual mode
Network 2: Control conveyor motor backwards in manual mode
%FC1 "MOTOR_MANUAL"
EN ENO
%I0.1 "-K0"
%I0.5 "-B1"
%I1.5 "-S4"
%I0.2 "-S0" Manual_mode_active
%I1.4 "-S3" Pushbutton_manual_mode
Enable_OK
%I0.0 "-A1" Safety_shutoff_active
Conveyor_motor_manual_mode %Q0.0 "-Q1"
PLC data types
Watch and force tables
Online backups
Traces
Program info
Device proxy data
PLC alarms
Text lists
Local modules
Common data
Documentation settings
Languages & resources
Online access
Card Reader/USB memory
Details view
Portal view Overview Main
Properties Info Diagnostics
Connection to CPU1516F terminated.

8 Checklist

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

9 Exercise

9.1 Task – Exercise

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

- Manual mode – extend cylinder
- Manual mode – retract cylinder

Note: *Pay attention to the reusability or encapsulation of the functions.*

9.2 Planning

Plan the implementation of the task on your own.

9.3 Checklist – Exercise

No.	Description	Completed
1	Function FC: CYLINDER_MANUAL created	
2	Interfaces defined	
3	Function programmed	
4	Function FC2 inserted in network 3 of OB1	
5	Input tags connected for Retract cylinder	
6	Output tags connected for Retract cylinder	
7	Compiling successful and without error message	
8	Function FC2 inserted in network 4 of OB1	
9	Input tags connected for Extend cylinder	
10	Output tags connected for Extend cylinder	
11	Compiling successful and without error message	
12	Download successful and without error message	
13	Switch on station (-K0 = 1) Cylinder retracted / Feedback activated (-B1 = 1) EMERGENCY OFF (-A1 = 1) not activated MANUAL mode (-S0 = 0) Do not activate Retract cylinder (-S5 = 0) Activate Extend cylinder (-S6 = 1) then extend cylinder (-M3 = 1) successfully	
14	Switch on station (-K0 = 1) Cylinder extended / Feedback activated (-B2 = 0) EMERGENCY OFF (-A1 = 1) not activated MANUAL mode (-S0 = 0) Do not activate Extend cylinder (-S6 = 0) Activate Retract cylinder (-S5 = 1) then retract cylinder (-M2 =1) successful	
15	Retract cylinder and Extend cylinder cannot be activated simultaneously	
16	Project successfully archived	

10 Additional information

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

www.siemens.com/sce/s7-1500