

# **SCE Training Curriculum**

Siemens Automation Cooperates with Education | 05/2017

## TIA Portal Module 032-100 Basics of FC Programming with SIMATIC S7-1500



#### 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: <u>siemens.com/sce/tp</u>

#### Continued training

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

#### Additional information regarding SCE

siemens.com/sce

#### Information regarding use

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

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

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

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

Use for industrial customer courses is expressly prohibited. We do not consent to commercial use of the training curriculums.

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

1	Goa	al	5
2	Pre	requisite	5
3	Red	quired hardware and software	6
4	The	eory	7
	4.1	Operating system and application program	7
	4.2	Organization blocks	8
	4.3	Process image and cyclic program processing	9
	4.4	Functions	. 11
	4.5	Function blocks and instance data blocks	. 12
	4.6	Global data blocks	. 13
	4.7	Library-compatible code blocks	. 14
	4.8	Programming languages	. 15
5	Tas		. 16
6	Pla	nning	. 16
	6.1	EMERGENCY STOP	. 16
	6.2	Manual mode – Conveyor motor in manual mode	. 16
7	Stru	actured step-by-step instructions	. 17
	7.1	Retrieve an existing project	. 17
	7.2	Create a new tag table	. 18
	7.3	Create new tags within a tag table	. 20
	7.4	Import "Tag_table_sorting_station"	. 21
	7.5	Create function FC1 "MOTOR_MANUAL" for the conveyor motor in manual mode	. 25
	7.6	Define the interface of function FC1 "MOTOR_MANUAL"	. 27
	7.7	Program FC1: MOTOR_MANUAL	. 30
	7.8	Program organization block OB1 – Control of the forward belt tracking in manual mode	. 37
	7.9	Program organization block OB1 – Control of the backward belt tracking in manual mode	. 42
	7.10	Save and compile the program	. 44
	7.11	Download the program	. 45
	7.12	Monitor program blocks	. 46
	7.13	Archive the project	. 48
8	Che	ecklist	. 49
9	Exe	ercise	. 50
	9.1	Task – Exercise	. 50
	9.2	Planning	. 50
	9.3	Checklist – Exercise	. 51
1(	) Add	litional information	. 52

## **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.

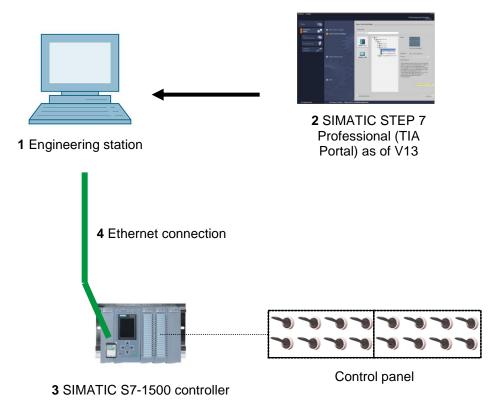
## 2 Prerequisite

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
- 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



## 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.

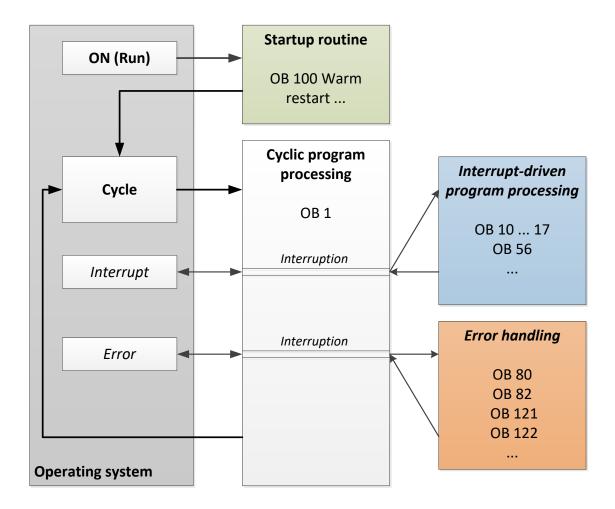


Figure 1: Start events in the operating system and OB call

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**.

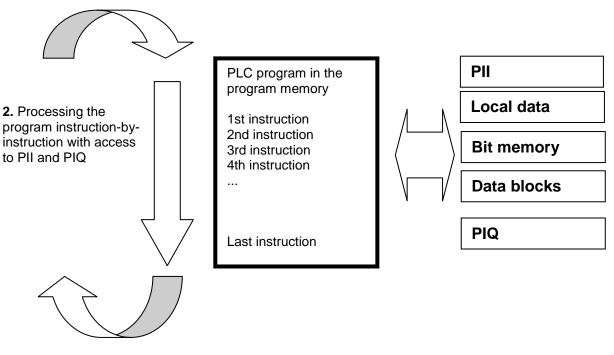
The cyclic program processing sequence is as follows:

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.



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.

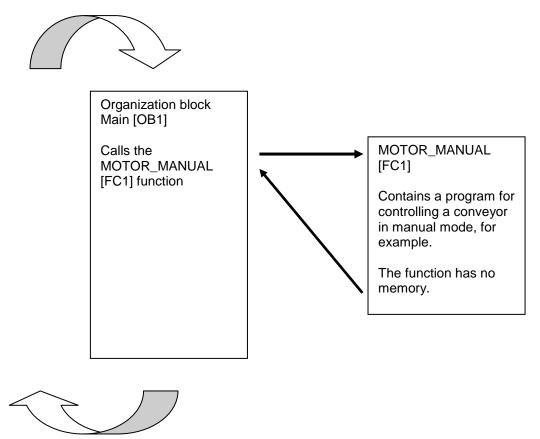


Figure 3: Function with call from organization block Main [OB1]

#### 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.

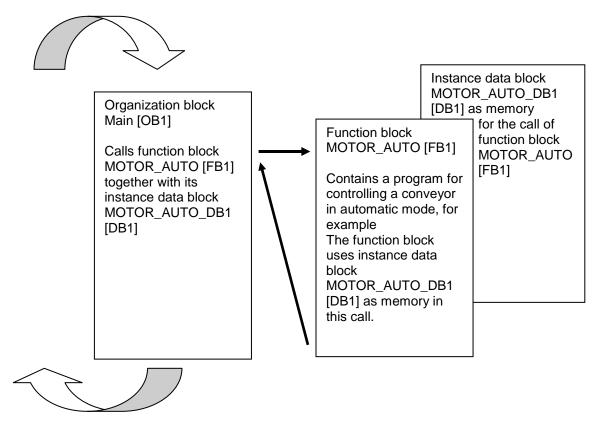


Figure 4: Function block and instance with call from organization block Main [OB1]

#### 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.

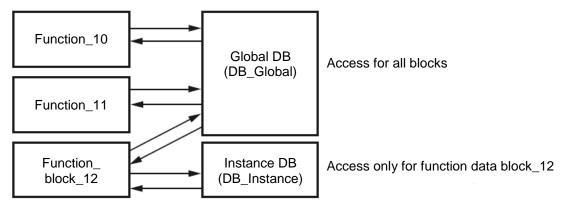


Figure 5: Difference between global DB and instance DB.

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.

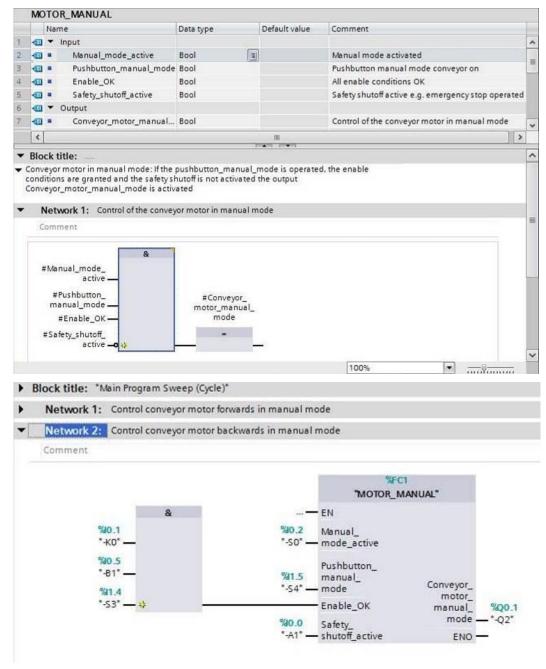


Figure 6: Library-compatible function with call in OB1

#### 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.

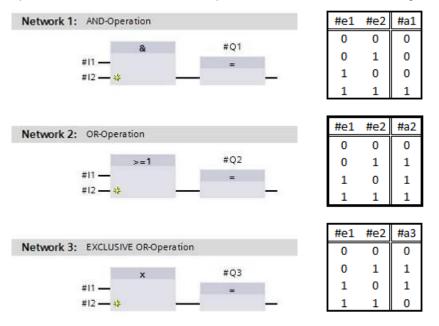


Figure 7: Binary functions in FBD and associated logic table

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

 $\rightarrow$  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  $\rightarrow$ 

 $\label{eq:project} \mbox{Project} \rightarrow \mbox{Retrieve in the project view Confirm your selection with Open.} (\rightarrow \mbox{Project} \rightarrow \mbox{Retrieve in the project} \rightarrow \mbo$ 

Project Edit	View	Insert	Online	Options							
New Open				Ctrl+O							
Migrate pro Close	ject			Ctrl+W							
Save Save as			Ctrl	Ctrl+S +Shift+S							
Delete proj Archive	ect			Ctrl+E							
Retrieve	_										
👕 Card Reade 🎬 Memory ca		nory		;							
Upgrade											
	Upgrade D:lAutomation/013_10\013_101_CPU314C D:lAutomation/012_10\012_101_CPU1516F D:lVorlagenprojekt_Webserv\Tank_V13_SP1 D:l\032-200_FB-frogrammierung_S7-314 D:lAutomatisi\012-100_CPU1500_V13_SP1										
D:\Automat D:\Vorlagen D:\\032-20	ion\012_1 nprojekt_W 00_FB-Prog	01012 ebserv. rammie	_101_CPU \Tank_V1 erung_S7-	1516F 3_SP1 314							

Retrieve  $\rightarrow$  Select a .zap archive  $\rightarrow$  Open)

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

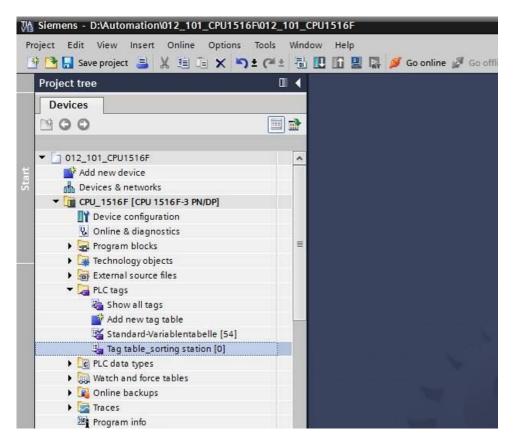
#### 7.2 Create a new tag table

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

2000000	ect Edit View Insert Online Options Tool: 🎦 🗔 Save project 🎩 🐰 🏥 🗎 🗙 🍤 🖢 🥐			Help	1 12	ø	Go online	Go
P	Project tree							
ſ	Devices							
1	300	1	-					
-								
	012_101_CPU1516F							
100	Add new device	-						
Start	Devices & networks							
	CPU_1516F [CPU 1516F-3 PN/DP]							
	Device configuration							
	Q. Online & diagnostics	-						
	Grinne d disgrissites							
20	Technology objects							
	External source files							
	▼ 📮 PLC tags							
	how all tags							
	🗳 Add new tag table							
1000	🗳 Standard-Variablentabelle [54]	1						
	C PLC data types	-						
	Watch and force tables							
	🕨 📴 Online backups							
	🕨 📴 Traces							
	📴 Program info							
	Device proxy data							
	🖂 PLC alarms							
	🔄 Text lists							
	Local modules							
	🕨 🎑 Common data						and the second	

→ Rename the tag table you just created as "Tag\_table\_sorting\_station" (→ right-click "Tag\_table\_1" → "Rename" → Tag\_table\_sorting\_station).

#### SCE Training Curriculum | TIA Portal Module 032-100, Edition 05/2017 | Digital Factory, DF FA



 $\rightarrow$  Open this tag table with a double-click. ( $\rightarrow$  Tag\_table\_sorting\_station)

ject Edit View Insert Online Options 📑 🔒 Saveproject 🝶 💥 🗐 🗐 🗙 🏹 :				🧭 Go offli	ne 🔥 🕅	i x ∃ II					Totally I	ntegrated	d Automation PORTA
Project tree			1_CPU1516F + CF				tags ▶ Ta	ig table	_sorting	station [0]			_ # =>
Devices											🕣 Tag	s 🔳 Ur	ser constants
900		- 	🖻 😷 🗽										
		Tag	table_sorting static	n									
<ul> <li>012_101_CPU1516F</li> </ul>	A		Name		Data type	Address	Retain	Visibl	Acces	Comment			
Add new device		1	<add new=""></add>		cons type								
A Devices & networks								0	C				
CPU_1516F [CPU 1516F-3 PN/DP]	100												
Device configuration													
V. 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 [0]													
Cit PLC data types													
Watch and force tables									0	Properties	Linfo 🖁	Diagnostio	cs =
Online backups		6											-
Firaces		Gene	al										
Program info		Tag		Tag									
Device proxy data				Tay	-								
PLC alarms				Ge	eneral								
Text lists													
Local modules				1		Name:							
Common data						Data type:							(3)
A Cocumentation settings	~			- A		Address							
Details view						nuuress.	Retained						
							Inclained						
Data type Comment Name	1.1					Comment:							

#### 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).

								🕣 Tags	User constants
÷,	1	00° ix							
1		le_sorting sta	ation						
	Na		Data type	Address	Retain	Visibl	Acces	Comment	
	Na		Data type	Address %I0.0	Retain	Visibl	Acces	Comment	

→ Change the address to %Q0.0 (Q 0.0) by entering this directly or by clicking the dropdown 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 → ☑)

ro	grami	ming ► CPU_1	1516F [CI	PU	1516F-3	PN	′DP] ▶	PLC tag	s ▶ Ta	g table_sorting		
										🕣 Tags	User User	constants
- State		🕈 🔅 🕯										<b>=</b>
	Tag ta	ble_sorting st	ation									
	N	lame	Data ty	pe	Address		Retain	Visibl	Acces	Comment		
1	-	Q1	Bool		%10.0	-						
ß		<add new=""></add>						identifier.	C.			
							18	identifier:	100 million		•	
							Ope	Operand type:				
								Address:	M			
							Bi	t number:	0			
											11	
											×	
						_						

 $\rightarrow$  Enter the "Conveyor motor M1 forwards fixed speed" comment for the tag.

								🕣 Tags	User constants
-	<b>1</b>	🖻 端 🕅							
		able sorting sta	ation						
1	lag t	able_solung su	1	1				1	
1	-	Name	Data type	Address	Retain	Visibl	Acces	Comment	
a L	-		Data type	Address %Q0.0	Retain	Visibl	Acces		forwards fixed speed

→ 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".

...rogramming > CPU\_1516F [CPU 1516F-3 PN/DP] > PLC tags > Tag table\_sorting station [2] \_ **- -** X - Tags User constants 🥩 🕐 🖶 😤 🕅 -Tag table\_sorting station Data type Address Retain Visibl... Acces... Comment Name  $\checkmark$ -Q1 conveyor motor -M1 forwards fixed speed conveyor motor -M1 backwards fixed speed 2 -00 02 3 <Add new> 

 $(\rightarrow$  <Add>  $\rightarrow$  Q2  $\rightarrow$  Enter  $\rightarrow$  Comment  $\rightarrow$  Conveyor motor M1 backwards fixed speed)

#### 7.4 Import "Tag\_table\_sorting\_station"

- → To insert an existing symbol table, right-click on an empty field of the created
   "Tag\_table\_sorting\_station". Select "Import file" in the shortcut menu.
  - $(\rightarrow$  Right-click in an empty field of the tag table  $\rightarrow$  Import file)

Project tree		rogrami	ming 🔸 CPU_1516F [CPU 1516F-3 PN/DP] 🔸 PLC tags 🔸 Tag table_sorting station [2] 🛛 🗕 🖬 🔳 🔪	Tasks 🖬 🗊 🕨 🕨
Devices			Tags 🗉 User constants	Options
300		1 2 E	e 🕫 🕅 🖓 👘	Options
2		Tag ta	able_sorting station	✓ Find and replace
<ul> <li>032_100_FC-Programming</li> </ul>	^	1	Name Data type Address Retain Visibl Acces Comment	
Add new device		1 📶	Q1 Bool %Q0.0 🗌 <table-cell> conveyor motor -M1 forwards fixed speed</table-cell>	Find:
🛔 Devices & networks		2 📶	Q2 Bool %Q0.1 🦳 🧭 conveyor motor -M1 backwards fixed speed	
CPU_1516F [CPU 1516F-3 PN/DP]		3	<add news<="" td=""><td>Whole words only</td></add>	Whole words only
Device configuration	=		💕 Insert row	Match case
🖳 Online & diagnostics			Add row	
Program blocks			X cut Ctrl+X	Find in substructures
Technology objects			Copy Ctrl+C	Find in hidden texts
External source files			De Paste Ctrl+V	Use wildcards
✓ Dec tags			X Delete Del	Use regular expressions
a Show all tags			Rename F2	use regular expressions
Add new tag table			Cross-reference information Shift+F11	O Whole document
Standard-Variablentabelle [54]				From current position
Tag table_sorting station [2]		-	Monitor all	O Selection
PLC data types		-	Import file	() selection
Watch and force tables     Watch backups		-	Export file	Down
Traces			C Properties	OUP
Program info				Find
h Davice providete	v			Fing
✓ Details view	_			Replace with:
		-		
Data type Comment Name				Replace Replace all
Bool conveyor m 40 Q1				
Bool conveyor m 💶 Q2				
		<	11	
< III	>		Properties 1 Info 3 V Diagnostics	> Languages & resources

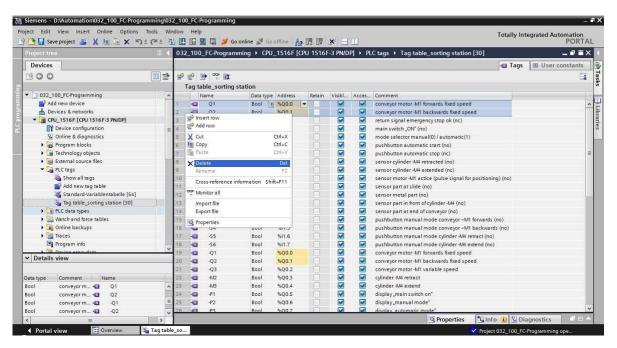
→ Select the desired symbol table (e.g. in .xlsx format) and confirm the selection with "Open".

 $(\rightarrow SCE\_EN\_020-100\_Tag\_table\_sorting\_station... \rightarrow Open)$ 

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

Import co	mpleted with warnings (0032:000031) X
Â	Import completed with warnings.
_	Detailed information is shown in the import log file.
	Click here to view the log file.
	ОК

- → 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.



 $(\rightarrow$  Right-click on selected tags  $\rightarrow$  Delete)

→ 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.

roject Edit View Insert Online Op * New	1				Help						-		Totally Int	tegrated Automation
	rl+0	± (# ±	1070				online 🖉 (							PORTA
Migrate project		u		012_	101_0	CPU1516F	▶ CPU_1:	516F [CPU	1516F-3	PN/DPJ	<ul> <li>PLC tage</li> </ul>	ags  ▶ Tag table_sorting station [28]		_ • • • •
Close Ct	rl+W												🕢 Tags	User constants
Save Ct	trl+S		=>		0 B	00° itt								
Save as Ctrl+Sh	ift+S			-	-	1 2 000								
Delete project Ct	trl+E		1000	16	and and a second	le_sorting			-			1		
Archive			^			me		e Address	Retain	Visibl		Comment		
Retrieve					-0	-A1	10000	11 %10.0				return signal emergency stop ok (nc)		-
Card Reader/USB memory	10				-00	-КО	Bool	%10.1				main switch "ON" (no)		
Card Reader/USB memory Memory card file				3	-	-50	Bool	%10.2				mode selector manual(0) / automatic(1)		
r wemory card file	•			4	-	-51	Bool	%10.3				pushbutton automatic start (no)		
Upgrade			_	5	-00	-52	Bool	%10.4				pushbutton automatic stop (nc)		
Print C	trl+P		-	6	-00	-81	Bool	%10.5				sensor cylinder -M4 retracted (no)		
Print preview				7	-	-B2	Bool	%10.6				sensor cylinder -M4 extended (nc)		
D:Automation\012 10\012 101 CPU151				8	-00	-B3	Bool	%10.7				sensor motor -M1 actice (pulse signal for		
C:lUsers\spe\Do\032_100_FC-Programmir				9	-	-B4	Bool	9611.0				sensor part at slide (no)		
D:Automation/013_10013_101_CPU314				10	-	-85	Bool	%11.1				sensor metal part (no)		
D:/Vorlagenprojekt_Webserv\Tank_V13_S				11	-	-86	Bool	%11.2				sensor part in front of cylinder -M4 (no)		
D:11032-200_FB-Programmierung_S7-314				12	-00	-87	Bool	%11.3				sensor part at end of conveyor (no)		
D:lAutomatisi\012-100_CPU1500_V13_SP				13	-	-53	Bool	%11.4		<b>S</b>		pushbutton manual mode conveyor –M1		
	3			14	-	-54	Bool	%11.5				pushbutton manual mode conveyor -M1		
Exit				15	-	-55	Bool	%11.6				pushbutton manual mode cylinder -M4 re		
Online backups				16	-00	-56	Bool	%11.7				pushbutton manual mode cylinder -M4 ex		
Traces				17	-	-Q1	Bool	%Q0.0		Image: A start and a start		conveyor motor -M1 forwards fixed speed		
Program info				18	-	-Q2	Bool	%Q0.1				conveyor motor -M1 backwards fixed speed		
Device proxy data				19	-00	-Q3	Bool	%Q0.2		<b>S</b>		conveyor motor -M1 variable speed		
PLC alarms				20	-	-M2	Bool	%Q0.3				cylinder -M4 retract		
Text lists				21		-MB	Bool	%Q0.4				cylinder -M4 extend		
Local modules				22	-00	-P1	Bool	%Q0.5				display_main switch on"		
Common data				23	-00	-P2	Bool	%Q0.6				display "manual mode"		
Documentation cettings			~	24	-	-P3	Bool	%Q0.7				display_automatic mode"		
✓ Details view				25	-00	-P4	Bool	%Q1.0				display "emergency stop activated"		
			1	26	-01	-P5	Bool	%01.1				display automatic mode started"		
Data type Comment Name												Q Properties	Info 🗓 🖏 Dia	agnostics

 $(\rightarrow \text{Project} \rightarrow \text{Save as } \dots \rightarrow 032\text{-}100\text{-}\text{FCProgramming} \rightarrow \text{Save})$ 

## 7.5 Create function FC1 "MOTOR\_MANUAL" for the conveyor motor in manual mode

- → In the PLC programming section of the portal view, click "Add new block" to create a new function.
  - $(\rightarrow$  PLC programming  $\rightarrow$  Add new block  $\rightarrow = c$

iemens - D:\Automation\032_1	100_FC-Programming\032_100_FC-Programmin	ing	_ # C
		Totally Integr	ated Automation PORTAL
tart	Device: CPU_1516F	Add new block	
Devices &	Show all objects	Name: Block_1	^
PLC programming	Add new block	Language: SCL  Number: 1	
Motion & 🔅	• 2	Organization block	
Drive parameterization	Show cross-references	Description: Functions are code blocks or subroutines without dedicated memory.	
Visualization	Show program structure	Function block	-
Online & Diagnostics		<b></b>	
		Function	
	e Help	DB Data block	
		Additional information	
		Add new and open Add	~
Project view	Opened project: D:\Automation	on\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".

 $(\rightarrow$  Name: MOTOR\_MANUAL $\rightarrow$  Language: FBD  $\rightarrow$  Number: Automatic  $\rightarrow$   $\blacksquare$  Add new and open  $\rightarrow$  Add)

Name: MOTOR_MANUAL					
Organization block	Language: Number:	FBD 1 Manual O Automatic	•		
Function block	Description: Functions are d	code blocks or subrou	utines without dec	licated memory.	
Function					
Data block	More				

#### 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.

ió.	ы	X 🥩 🕐 🎭 🖿 🗖 🚍 🛛	- <b>2 ± 2 ±</b> = 😥	¢° 6₀	↓ 1 = 1 = 0, 00 ■	
	MO	DTOR_MANUAL				
	-	Name	Data type	Default value	Comment	
1	-	<ul> <li>Input</li> </ul>				
2		Add new>				
3	-	<ul> <li>Output</li> </ul>				
4		Add new>				
5		▼ InOut				
6		<a>Add new&gt;</a>				
7	-01	▼ Temp				
8		Add new>				
9	-	<ul> <li>Constant</li> </ul>				
10		Add new>				
11	-	▼ Return				
12	-	MOTOR_MANUAL	Void			
	۲		III III			>
•	Blo	ock title:	here here			^
C	omi	iment				
•	١	Network 1:				
	C	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.

( $\rightarrow$  Output: Conveyor\_motor\_manual\_mode  $\rightarrow$  Bool  $\rightarrow$  Control of the conveyor motor in manual mode)

	1.10	1 3	🖗 🕐 🥾 🔚 🚍 💬	🖀 ± 🖀 ± 🔚 😥	eo eo eo eo	⊉ <sup>1</sup> ≡ <sup>1</sup> ≡ e <sup>0,</sup> ∞ Ш	
	MC		DR_MANUAL	Data type	Default value	Comment	
1	-		Input	Data type	Delault value	comment	
2	-		<add new=""></add>				÷
3	-00	-	Output				
:	-00			Bool		Control of the conveyor motor in manual mode	
5			<add new=""></add>	(mat)			-17
5	-	•	InOut				
7.5			<add new=""></add>				
3	-	•	Temp				
9			<add new=""></add>				
0	-	•	Constant				
11			<add new=""></add>				
2	-00	•	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".

 $(\rightarrow Manual\_mode\_active \rightarrow Enter \rightarrow Bool \rightarrow Manual mode activated)$ 

→ 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.

0	C-Pro	ogr	ramming + CPU_1516F [Cl	PU 1516F-3 PN/DP]	<ul> <li>Program bl</li> </ul>	ocks ▶ MOTOR_MANUAL [FC1] 📃 🖬 🖬 >
юŝ			🖗 🔮 🎭 🖿 🚍 🂬 : Dr. MANUAL	8:2:50	¢° ⊊₀ ¢≣ 9≣	
	IVIC		me	Data type	Default value	Comment
1	-	2020	Input	outo type	Dendare Police	connent
2			Manual_mode_active	Bool		Manual mode activated
E.	-		Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on
Į.	-		Enable_OK	Bool		All enable conditions OK
5	-		Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated
52			<add new=""></add>			
ti.	-	•	Output			
\$	-		Conveyor_motor_manual	Bool		Control of the conveyor motor in manual mode
Ř.			<add new=""></add>			
0		•	InOut			
1			<add new=""></add>			
2	-	•	Temp			
3			<add new=""></add>			
4	-	•	Constant			
5			<add new=""></add>			
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.

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

	C-Pr	ogr	ammi	ng 🕨	CPU_	_1516	F [C	PU 1516F	-3 PN/DP]	<ul> <li>Program b</li> </ul>	locks 🕨 MOTOR_MANUAL [FC1] 📃 I	7 2	×
13		<u>د</u>		<b>B</b> 22			0	8±21	<b>=</b>	C <sup>0</sup> 60 CE 9	4≱ I <sub>=</sub> 1 <sub>=</sub> 6, ∞ Ⅲ	E	4
P.	-		RMA										
		Na	me					Data type		Default value	Comment		
1	-	•	Input										^
2	-		Ma	nual_	mode_	active		Bool			Manual mode activated		5
3	-00		Pus	shbutt	on_ma	nual_	mode	Bool			Pushbutton manual mode conveyor on		=
4	-		En	able_0	Ж			Bool			All enable conditions OK		
5	-		Sat	fety_sl	nutoff_	active		Bool		ř.	Safety shutoff active e.g. emergency stop ope	rated	
6			<a(< td=""><td>dd nev</td><td>V&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></a(<>	dd nev	V>								
7	-	•	Outpu	t									
8	-		Co	nveyo	_moto	or_man	ual	Bool			Control of the conveyor motor in manual mod	le	
•	<	-	1980							1111			~
	1.4	-							,	• · · · · · · · · · · · · · · · · · · ·			-
8		> = 1	·??]	н	-01	↦	-[=]						
•	Blo	ck	title:										^
	ond	itio		grante	d and	the sa	fetysh	nutoff is not		node is operated the output	d, the enable		
•	1	let	work 1	1: Co	ntrol o	f the c	onvey	or motor in	manual m	ode			
	C	om	ment										=

#### 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.

& >=	??	н	-01	↦	-[=]
------	----	---	-----	---	------

→ 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).

 $(\rightarrow \text{Instructions} \rightarrow \text{Basic instructions} \rightarrow \text{Bit logic operations} \rightarrow -[=])$ 

					Options
a 🗚 🖈 👻 🖦 🖿 🚍 💬	2 ± 2 ± 🖂 🗊	e° 6₀ e≣ 9≣	9 1 1 8 0 m	4	•
MOTOR_MANUAL					> Favorites
Name	Data type	Default value	Comment	-	✓ Basic instructions
🕣 🔻 Input				^	Name
Manual_mode_active	Bool		Manual mode activated		General
🕣 = Pushbutton_manual_mod	e Bool		Pushbutton manual mode conveyor on		▼ Hit logic operations
📹 🔹 Enable_OK	Bool		All enable conditions OK		E &
Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operate	d	E >=1
🕣 🔻 Output					E ×
Conveyor_motor_manual	. Bool		Control of the conveyor motor in manual mode	~	€ -[=]
<		III		-	E -[/=]
	T	*1 1 <b>*</b> 1			E -[R]
>=1 📅 0[=	1				I −[S]
					E SET BF
				1000	
200300000000000000000000000000000000000				^	And a second sec
onveyor motor in manual mode: If the	pushbutton_manual_r	mode is operated,	the enable	^	RESET_BF
onveyor motor in manual mode: If the onditions are granted and the safety s	hutoff is not activated	mode is operated, the output	the enable	^	E RESET_BF
onveyor motor in manual mode: If the onditions are granted and the safety s	hutoff is not activated	mode is operated, the output	the enable	^	E RESET_BF E SR E RS
onveyor motor in manual mode: If the onditions are granted and the safety s	hutoff is not activated vated	the output	the enable	-	된 RESET_BF 된 SR 된 RS 된
onveyor motor in manual mode: If the onditions are granted and the safety s onveyor_motor_manual_mode is acti Network 1: Control of the conve	hutoff is not activated vated	the output	the enable		197 RESET_BF 197 SR 197 RS 197
conveyor motor in manual mode: If the onditions are granted and the safety s conveyor_motor_manual_mode is acti	hutoff is not activated vated	the output	the enable		包 RESET_BF 8) SR 8) RS 8)
conveyor motor in manual mode: If the onditions are granted and the safety s conveyor_motor_manual_mode is acti Network 1: Control of the conve	hutoff is not activated vated	the output	the enable		包 RESET_BF 80 SR 87 RS 1 -IPI- 80 -IVI- 80 -IPI- 80 -IPI- 80 -IVI-
conveyor motor in manual mode: If the onditions are granted and the safety s conveyor_motor_manual_mode is acti Network 1: Control of the conve	hutoff is not activated vated	the output	the enable		記 RESET_BF 記 SR 記 RS 記 -IPI- 記 -IPI- 記 -IPI- 記 -IPI- 記 -INI- 記 FI-IRG
conveyor motor in manual mode: If the onditions are granted and the safety s conveyor_motor_manual_mode is acti Network 1: Control of the conve	hutoff is not activated vated	the output	the enable		된 RESET_BF 원 SR 원 RS 원 -IPI- 원 -INI- 원 -INI- 원 -INI- 원 P_TRIG 원 N_TRIG
onveyor motor in manual mode: If the onditions are granted and the safety s onveyor_motor_manual_mode is acti Network 1: Control of the conve	hutoff is not activated vated	the output	the enable		<ul> <li>配 RESET_BF</li> <li>配 SR</li> <li>配 RS</li> <li>配 -PF-</li> <li>편 -PF-</li> <li>편 -PF-</li> <li>편 -PF-</li> <li>편 -PF-</li> <li>편 -PF-</li> <li>편 -RT-</li> <li>편 P-TRIG</li> <li>편 N_TRIG</li> </ul>
onveyor motor in manual mode: If the onditions are granted and the safety s onveyor_motor_manual_mode is acti Network 1: Control of the conve	hutoff is not activated vated	the output	the enable		원 RESET_BF 원 SR 원 RS 원 - P - 원 - N - 원 -(N)- 원 -[N]- 원 P_TRIG 원 N_TRIG 원 R_TRIG
	hutoff is not activated vated	the output	the enable		E RESET_BF     E     SR     E     SR     E     SR     E     S     S     E     - P -     E     - P -     E     - P -     E     - P -     E     E     - N -     E     E     N_TRIG     E     N_TRIG     E     N_TRIG     E     X     N_TRIG     E     X

→ 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 <.</p>

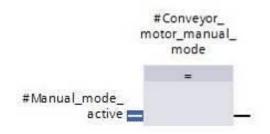
	(→	-00	Conveyor_	_motor_	_manual_	_mode)
--	----	-----	-----------	---------	----------	--------

	C-Pro	ogr	amming + CPU_1516F [Cl	PU 1516F-3 PN/DP]	Program blo	ocks 🕨 MOTOR_MANUAL [FC1] 🛛 🗕 🖬 🖬 🕇	<
	0 L	8	<b>* * % E E E</b> 9	<b>3</b> • 2 • 🗆 🗊	00 Go #8 G8	9 L L & O W H	-
14			DR MANUAL		- +0 (m 10)		
-			me	Data type	Default value	Comment	
1	-	•	Input			//////////////////////////////////////	•
2	-		Manual_mode_active	Bool		Manual mode activated	Ī
3			Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor on	8
4			Enable_OK	Bool		All enable conditions OK	
5	-00		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	
1	<				1111	>	
-					* · · ·		-
8		>=1	[??]• -• -[=]				
-	RIO	ck	title:				~
1000	-		r motor in manual mode: If the	pushbutton manual r	node is operated.		
1	ond	itio	ns are granted and the safety sh	nutoff is not activated t	the output		
3	Conv	eyo	or_motor_manual_mode is activ	ated			
-		1 - 1	www.l.d. Control states and		u du		
	- 10		work 1: Control of the convey	or motor in manual m	ode		
	C	om	ment				
				-			
			<11.10				ł
			= ~+				

→ 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.

N		R MA			[2	🤊 📲 ± 📇 ±		C~ 40 CE	*E 🗸					-	-
		me	NUAL			Data type	1	Default value	Con	nment					
4	-	Input				1 30									ľ
	-	Ma	nual_m	ode_a	ctive	Bool			Mar	nual mod	activat	ed			T
4		Pus	hbutto	n_mar	nual_mo	de Bool			Pus	hbutton r	nanualr	node con	veyor on		1
-	- 10	Ena	ble_O	ĸ		Bool			Alle	enable co	nditions	ОК			
4		Saf	ety_shu	utoff_a	ctive	Bool			Saf	ety shutof	active	e.g. emer	gency stop op	erated	I
4	-	Output	5												
4	=	Cor	weyor_	motor	_manua	I Bool			Cor	trol of the	convey	or motor i	in manual mo	de	1
	<	* *						111						)	i
	lock nveyo	0.5 (2 a. 0 7 (2 )	in mar	nual m	ode: If th	e pushbutton	_manual_n	node is opera	ted, the e	nable					1
					he safety ode is ac	/shutoff is not tivated	activated t	he output							
	Net	work 1	: Con	trol of	the conv	eyor motor in	manual m	ode							ļ
	Com	ment													

 → 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.



 $(\rightarrow \blacksquare$  Manual mode active)

→ Click the <sup>a</sup> icon in your logic toolbar to insert an AND logic operation between the #Manual\_mode\_active tag and your assignment block.

8	> = 1	[??]	٦	-01	↦		
	AND	ogic op	erati	on [Shi	ift+F2]		
col	ndition	s are g	rante	d and	the sat	f the pushbutton_manual_mode is operated, the en ety shutoff is not activated the output activated	abl
	Netv	vork 1	: Co	ntrol o	f the c	onveyor motor in manual mode	
	Comn	nent					
					Convey or_ma mode	nual_	
	#Mar	nual_m a	ode_ ctive		/=	_	

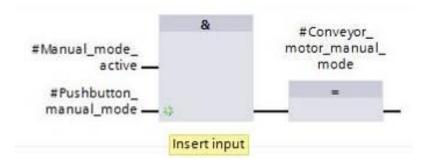
→ Double-click the second input of the & logic operation <??.?> and enter the letter "P" in the field that appears in order to see a list of available tags starting with "P".Click the #Pushbutton\_manual\_mode tag and apply with → Enter.

 $(\rightarrow \&- block \rightarrow <??. \rightarrow P \rightarrow #Pushbutton_manual_mode \rightarrow Enter)$ 

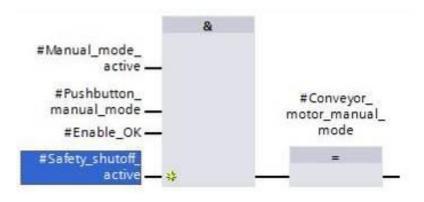
	Nam	R_MANUAL	Data type	Default value	Comment		
		nput					
	-	Manual_mode_active	Bool	1	Manual mode activated		
	-	Pushbutton_manual_mode	Level Level		Pushbutton manual mode conveyor on		
	-	Enable_OK	Bool		All enable conditions OK		
	-	Safety shutoff active	Bool		Safety shutoff active e.g. emergency stop ope	erate	ed
		Dutput					
		Conveyor_motor_manual	Bool		Control of the conveyor motor in manual mot	de	
		-					
	4			101			5
8		1 → -01 → -[=]					>
	>=1 Block ti	tle:			the enable		>
c	>=1 Block tin	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	pushbutton_manual_r	mode is operated,	the enable		>
0	>=1 Block ti Conveyor	tle: motor in manual mode: If the p	pushbutton_manual_r nutoff is not activated t	mode is operated,	the enable		>
000	>=1 Block til Conveyor conditions Conveyor_	tle: motor in manual mode: If the s are granted and the safety sh motor_manual_mode is activ	pushbutton_manual_r nutoff is not activated t ated	mode is operated, the output	the enable		>
	>=1 Block tir Conveyor Conveyor Conveyor	tle: motor in manual mode: If the s are granted and the safety sh motor_manual_mode is activ- rork 1: Control of the convey	pushbutton_manual_r nutoff is not activated t ated	mode is operated, the output	the enable		>
000	>=1 Block til Conveyor conditions Conveyor_	tle: motor in manual mode: If the s are granted and the safety sh motor_manual_mode is activ- rork 1: Control of the convey	pushbutton_manual_r nutoff is not activated t ated	mode is operated, the output	the enable		>
0000	>=1 Block tir Conveyor Conveyor Conveyor	tle: motor in manual mode: If the s are granted and the safety sh motor_manual_mode is activ- rork 1: Control of the convey	pushbutton_manual_r nutoff is not activated t ated	mode is operated, the output	the enable		>
000	>=1 Block tir Conveyor Conveyor Conveyor	tle: motor in manual mode: If the j are granted and the safety sh motor_manual_mode is active ork 1: Control of the convey tent	pushbutton_manual_r nutoff is not activated f ated or motor in manual m #Conveyor_	mode is operated, the output	the enable		>
	>=1 Block ti conveyor conditions conveyor_ S Netw Comm	tle: motor in manual mode: If the j are granted and the safety sh motor_manual_mode is active ork 1: Control of the convey tent	pushbutton_manual_r nutoff is not activated t ated or motor in manual m	mode is operated, the output	the enable		>
000	>=1 Block ti conveyor conditions conveyor_ S Netw Comm	tle: motor in manual mode: If the j are granted and the safety sh motor_manual_mode is active ork 1: Control of the convey tent	pushbutton_manual_r nutoff is not activated i ated or motor in manual m #Conveyor_ motor_manual_	mode is operated, the output	the enable		>

**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 <sup>34</sup>/<sub>2</sub> 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

click	ing -	ol												
8	> = 1	[??]	٦	-01	↦	-[=]								
- Co	lock ti	motor	r in ma	nu	nvert R	LO [Ctrl+S	hift+4]_m	anual	_mo	de is	opera	ated, th	ne enable	9
co	ndition	s are g	grante	d and t	the safe	ety shutof activated	fis not act	tivate	d the	outp	ut			
•	Netv	vork 1	: Co	ntrol of	f the co	nveyor m	otor in ma	nual	mod	e				
	Comn	nent												
					&									
	#Man	nual_m	node_ active											
		77) 2010 - 2010 - 2010	onesen.											
		Pushbu inual_i					#Conve motor ma							
	;	#Enabl	le_OK	_			mod	Contraction of the second	-					
	#Saf	ety_sh		<b></b>			-		L					

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

H	H R	X	\$` \$\ <b>R</b> \$		📲 ± 📲 ± 🖃		Co 🕫 📾	⊉ '≡	x≡ 0, ∞			E	1
	1		R_MANUA	L				1					
		Nar			Data type		Default value	Comment					ļ
	-		Input			-							ŀ
			100 000 000 000 000 000 000 000 000 000	mode_active	Bool				ode activate	-			ł
	-			on_manual_mode						ode conveyor or	1		ł
			Enable_		Bool				conditions C				
				nutoff_active	8001			Safetyshi	itomactive e.	g. emergency st	op opera	ea	
			Output	r motor manual	Baal			Control of		motor in manu	al made		ļ
				motor manual	6001			Control of		motor in manu	aimode		
			contrejo	2 2 2		_			une contrejo.			Contra la	ļ
c	< Bloc	c <b>k t</b> eyor	t <b>itle:</b> r motor in m	anual mode: If the d and the safety sl nual_mode is activ	hutoff is not activ	nual_r		the enable				>	
c	Sonve conve conve conve conve	ck t eyor itior eyor	title: r motor in mi ns are grante r_motor_mai work 1: Co	d and the safety sl	hutoff is not activ ated	nual_r ated t	mode is operated, the output	the enable				>	
c	Sonve conve conve conve conve	ck t eyor itior eyor	t <b>itle:</b> r motor in m rs are grante r_motor_ma	d and the safetysl nual_mode is activ	hutoff is not activ ated	nual_r ated t	mode is operated, the output	the enable				>	
c	Riod Conve Conve N	ck t eyor itior eyor <b>letv</b> omr	title: r motor in m. ns are grante r_motor_mai work 1: Co ment nual_mode_ active	d and the safety si nual_mode is activ ntrol of the convey	hutoff is not activi ated vor motor in mani	ual_r ated 1 ual m	mode is operated, the output	the enable				>	
c	Riod Conve Conve N	ck t eyon itior eyon <b>letv</b> omr	title: r motor in m. ns are grante r_motor_mai work 1: Co ment nual_mode_	d and the safety sl uual_mode is activ ntrol of the convey	hutoff is not activ ated	ual_r ated 1 ual m	mode is operated, the output	the enable				>	

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

MOTOR_MANU/	AL [FC1]	Q Prop	erties	🗓 Info 🔒	Diagnostics	
General	_					
General Information Time stamps	General					
Compilation		Name:	MOTOR_N	MANUAL		
Protection Attributes		Type:	FC			
		1978 - C.	FBD			<b>•</b>
	• 		FBD manu			
			<ul> <li>autom</li> </ul>			

 $\rightarrow$  The program has the following appearance in LAD.

	🥩 👻 🎭 🖿 🚍 💬 : TOR MANUAL		(- 40 (m 45	
CORDONE	Name	Data type	Default value	Comment
-00	▼ Input			
-		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 operated
-	<ul> <li>Output</li> </ul>			
	<ul> <li>Conveyor_motor_manual</li> </ul>	Bool		Control of the conveyor motor in manual mode
<			101	11
and the second se				
Conve	k title: yor motor in manual mode: If the		node is operated	- Hi-
Conve Condi Conve		nutoff is not activated ated	node is operated the output	- Hi-

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

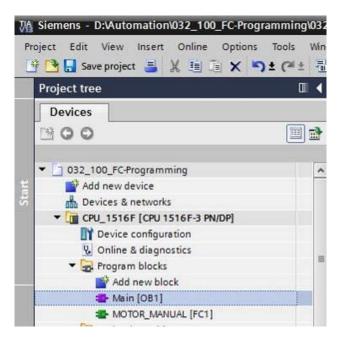
→ Before programming organization block "Main [OB1]", we switch the programming language to FBD (Function Block Diagram). To do so, first click on "Main [OB1]" in the "Program blocks" folder.

 $(\rightarrow$  CPU\_1516F[CPU 1516F-3 PN/DP  $\rightarrow$  Program blocks  $\rightarrow$  Main [OB1]  $\rightarrow$  Switch

programming language  $\rightarrow$  FBD)

M Siemens - D:\Automation\032_100_FC-Programming\032_1	00_FC-Programming	
Project Edit View Insert Online Options Tools Window	v Help 1 🕼 🖳 💋 Goonline 🖉 Gooffline 🛔 17 🖪 🖪 🗶 😑 📋	Totally Integrated Automation PORTAL
Project tree		Tasks III >
Devices		Options
1900 E		Options
		✓ Find and replace
032_100_FC-Programming		
Add new device		Find:
💑 Devices & networks		
▼ ☐ CPU_1516F [CPU 1516F-3 PN/DP]		Whole words only
Device configuration		Match case
Conline & diagnostics		
Program blocks		Find in substructures
Add new block		Find in hidden texts
Anton Open	0 Properties Alt+Enter	Use wildcards
- MOTOR		Use regular expressions
Cut Ctrl+X     External s T Copy     Cut		
External s     Gopy     Ctrl+C     Card+V     Ctrl+C     Ctrl+V		O Whole document
		From current position
Add ne Rename 52		O Selection
Add ne Rename F2		
Compile		Down
Download to device		Oup
Go online Ctri+K		Find
Online ha		L. Thu
Cross-reference information Shift+F11		Replace with:
Not Program in X Cross-references F11		•
Call structure		Replace Replace all
PLC alarm		
En Text liste Switch programming language	STL.	
Details view     Know-how protection	LAD Properties Linfo	S Diagnostics
Frint Ctrl+P	FBD	< III >
Name SPrint preview		> Languages & resources
Portal view		The project 032_100_FC-Programming

 $\rightarrow$  Open the "Main [OB1]" organization block with a double-click.

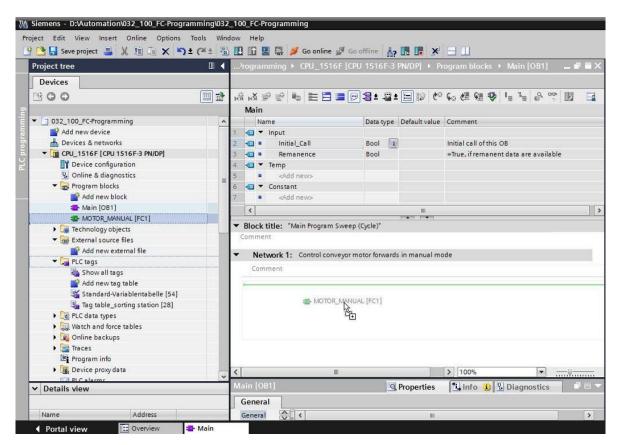


→ Assign Network 1 the name "Control conveyor tracking forward in manual/jog mode"

 $(\rightarrow$  Network 1:...  $\rightarrow$  Control conveyor motor forwards in manual mode)

	Aai				
IV	2000	Name	Data type	Default value	Comment
-	01	▼ Input			
4		Initial_Call	Bool 🔳		Initial call of this OB
-	01	Remanence	Bool		=True, if remanent data are available
4		▼ Temp			
		Add new>			
-	01	▼ Constant			
		Add new>			
	<				
-		c <b>k title:</b> *Main Program Sweep		Install Install	10

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



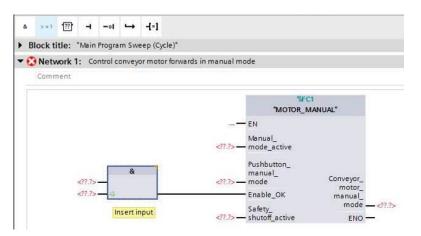
→ A block with the interface you defined and connections EN and ENO are inserted in Network 1.

	%FC "MOTOR_N	Alexandra and a	
	EN		
?.?	Manual_ mode_active		
<22.7>	Pushbutton_ manual_ mode	Conveyor_	
?.?	Enable_OK	motor_ manual_	
?.? —	Safety_ shutoff_active	mode ENO	_ ?.

→ To insert an AND before input parameter "Enable\_OK", select this input and insert the AND by clicking the  $\stackrel{\triangleq}{}$  icon in your logic toolbar (→  $\stackrel{\triangleq}{}$  ).

8	> = 1	[??]	н	-01	↦	-{=]
5	AND	logic o	perati	ion [Sh	ift+F2]	p (Cycle)*
0	Netv	vork 1	: Co	ntrol c	onvey	or motor forwards in manual mode
	Comn	nent				
						%FC1
					7	MOTOR_MANUAL"
				- EN	1	
			<11.7>		nual_ ode_ac	
			?.?		shbutt anual_ ode	
		1	??.?	En En	able_0	
			<11.1>	Sa	fety_ utoff_a	mode — ?.? active ENO —

 $\rightarrow$  Click the yellow star  $\stackrel{3}{\approx}$  of the AND block to add another input ( $\rightarrow$ 



- → 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

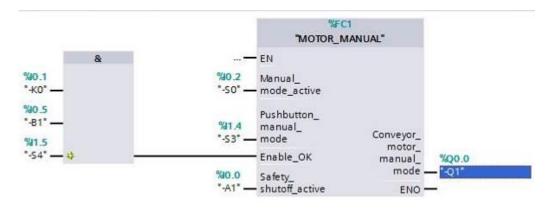
 $(\rightarrow Tag\_table\_sorting\_station \rightarrow Details view. \rightarrow -SO2 \rightarrow Manual\_mode\_active)$ 

PLC tags	SNetwork 1: Control conveyor motor forwards in manual mod	de
lags Show all tags	Comment	
💣 Add new tag table	Comment	
🐒 Default tag table [54]		NECA
ag table_sorting station [28]		%FC1
PLC data types		"MOTOR_MANUAL"
Watch and force tables		— EN
🕨 🙀 Online backups 🛛 🗸		Manual mode
✓ Details view	&	?. — active
	?.? —	Pushbutton
Name Data ty Comment	??.? —	?.? — manual_mode
-Q3 Bool conveyor motor - M1 variable s	?.?	Enable OK Conveyor motor
-SO Bool 🗉 mode selector manual(0) / auto.	Suns 4	
-51 Bool pushbutton automatic start (no)		salety_stitutoi
-S2 Bool pushbutton automatic stop (nc)		?.? — active ENO —

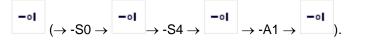
→ Or, enter the starting letters (e.g. "-S") of the desired global tag for <??.?> and select the global input tag "-S0" (%I0.2) from the displayed list (→ Manual\_mode\_active → -S → -S0).

> =	1 177	۲	-01	4	-[=]							
Block	title:	"Main	Progra	m Swe	ep (Cycle	)*						
3 Ne	twork 1	1: Co	ntrol co	onvey	or motor fo	prwards in m	nanualm	node				
Com	nment											
									%FC1			
								MC	TOR_MANU	JAL"		
								тмс — EN	TOR_MANU	JAL"		
					_			EN Manual		JAL"		
					-5			- EN		JAL"		
					-	<b>1</b> *-50*		EN Manual		%I0.2	mode selector	
				&	-	No. of Concession, Name		EN Manual	re		mode selector pushbutton aut	
		<11.7>	_	&	-	• *-S0*		EN Manual	e Bool	%10.2	and the second s	
		<11.75 <11.75		8	4 4 4	"-S0" "-S1"		EN Manual	e Bool Bool	%I0.2 %I0.3	pushbutton aut	
				8	- - -	•-50" •-51" •-52"		EN Manual	re Bool Bool Bool	%I0.2 %I0.3 %I0.4	pushbutton aut pushbutton aut	- 111 - T

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

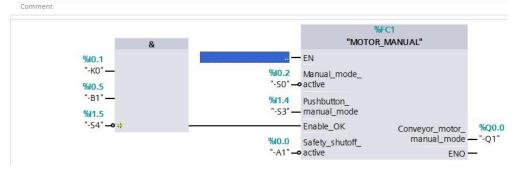


 $\rightarrow$  Negate the querying of input tags "-S0", "-S4" and "-A1" by selecting them and clicking



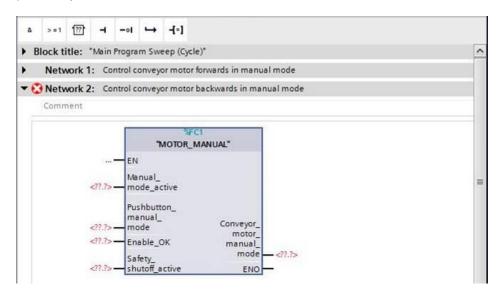
& >=1 122 → -01 → -[=] SR RS

Network 1: Control conveyor motor forwards in manual mode



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

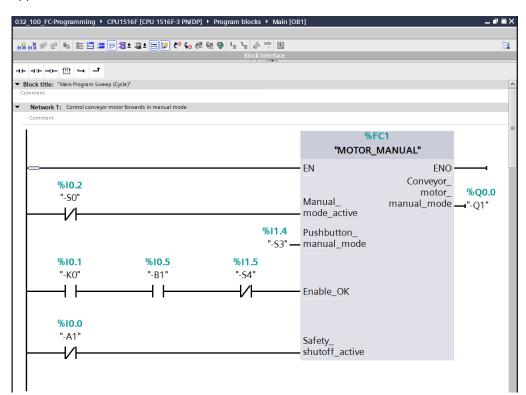
→ Assign Network 2 the name "Control conveyor motor backwards in manual mode" and insert your "MOTOR\_MANUAL [FC1]" function using drag-and-drop, as you did previously in Network 1.

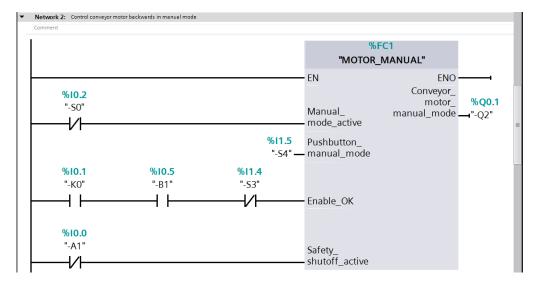


→ Connect your function as shown here. You obtain the following result in the FBD (Function Block Diagram) programming language.

& >=1 [??] → -ol → -[=] SR RS	Block interface	
2 >=1 [??] → -01 → -[=] SR RS		
Network 1: Control conveyor motor forwards in	manual mode	
Comment		
	NECA	
8	%FC1 "MOTOR MANUAL"	
×	— EN	
"-KO" —	%IO.2 Manual mode	
%10.5	"-S0" —• active	
"-B1" —	%11.4 Pushbutton_	
%1.5	"-S3" — manual_mode	
"-54" <b>•</b> *	Enable_OK Conveyor_motor_ %Q0.0	
	W0.0 Safety_shutoff_ manual_mode — "-Q1" "-A1" → active FNO —	
	"-A1" — active ENO —	
Network 2: Control conveyor motor backwards Comment	in manual mode	
Comment		
Comment	%FC1	
&	"MOTOR_MANUAL"	
& %IO.1	"MOTOR_MANUAL"	
& %I0.1 "-K0"—	"MOTOR_MANUAL" EN %I0.2 Manual_mode_	
& %IO.1	"MOTOR_MANUAL" EN %40.2 Manual_mode_ "-S0"	
& %i0.1 "-K0" — %i0.5	"MOTOR_MANUAL" EN %I0.2 Manual_mode_	
& %i0.1 "-K0"	"MOTOR_MANUAL"           %d0.2         Manual_mode_           "-S0" -> active         active           %d1.5         Pushbutton_	

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



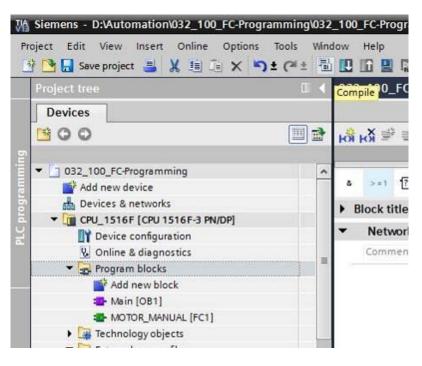


### 7.10 Save and compile the program

 $\rightarrow$  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

 $(\rightarrow \square$  Save project  $\rightarrow$  Program blocks  $\rightarrow \square$ ).

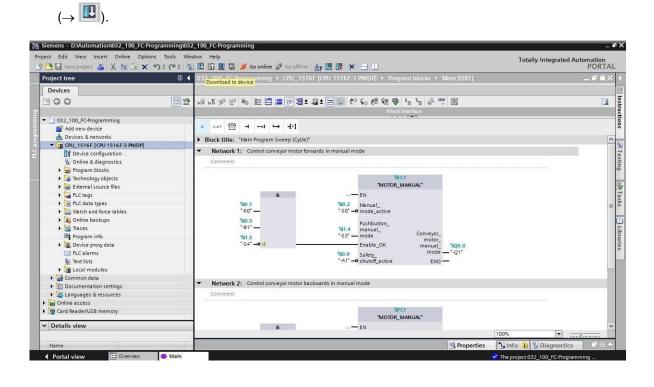


 $\rightarrow$  The "Info", "Compile" area shows which blocks were successfully compiled.

			Q Propert	ies 🚺 Ir	nfo	& Diag	nostics	
Genera	Cross-references	Compile	Syntax					
01	Show all messages							
Compilin	g completed (errors: 0; warn	ings: 0)						
Path	<b>•</b>	Description		Go to	?	Errors	Warnings	Time
💙 🔻 CF	PU_1516F			7		0	0	12:16:44 PM
<b>o</b> -	Program blocks			7		0	0	12:16:44 PM
0	MOTOR_MANUAL (FC1)	Block was succes	sfully compiled.	~				12:16:44 PM
0	Main (OB1)	Block was succes	sfully compiled.	7				12:16:44 PM
0		Compiling compl	eted (errors: 0; warni	nas: 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

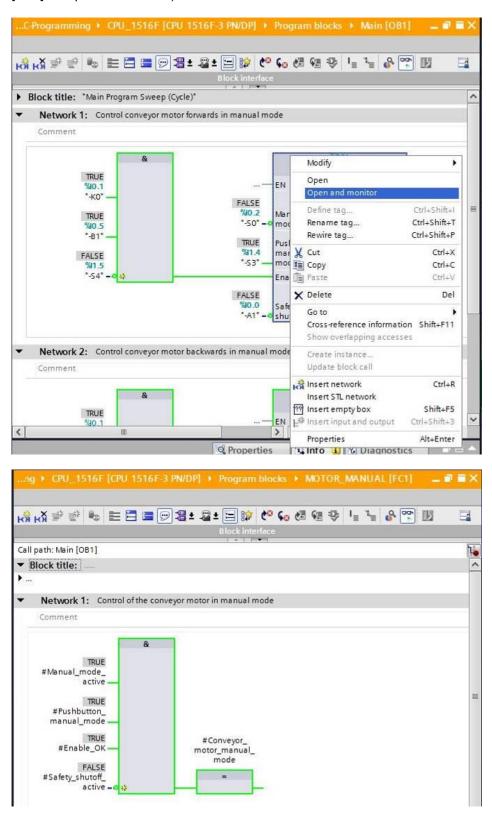
ightarrow The desired block must be open for monitoring the downloaded program. The monitoring

can be activated/deactivated by clicking the  $\square$  icon. ( $\rightarrow$  Main [OB1]  $\rightarrow$ 

...C-Programming + CPU\_1516F [CPU 1516F-3 PN/DP] + Program blocks + Main [OB1] \_ **-** = × kå kä 🕸 🐏 🖶 🚍 🚍 💬 🕮 ± 😫 🖆 ᅇ 🚱 🍓 🍕 🖳 🍹 🖉 🔢 ->=1 ?? 2 --01  $\rightarrow$ -[=] Block title: "Main Program Sweep (Cycle)" ~ Network 1: Control conveyor motor forwards in manual mode Comment "EC1 "MOTOR\_MANUAL" - EN %10.1 %10.2 Manual \*-K0\* -S0" - mode\_active %10.5 Pushbutton\_ "-B1" %11.4 manual\_ Conveyor\_ "-S3" -%11.5 - mode motor\_ manual\_ "-S4" -0 # Enable\_OK %Q0.0 - "-Q1" mode . %10.0 Safety\_ -A1" - shutoff\_active ENO -👸 👸 👻 🐁 📰 🚍 📼 🖉 🗄 😫 🖆 😭 📽 🐨 🔛 🖾 🚳 🌚 🖳 🍡 😵 🔢 & >=1 ?? --ol 1 + -[=] Block title: "Main Program Sweep (Cycle)" ~ Network 1: Control conveyor motor forwards in manual mode Comment %FC1 & "MOTOR\_MANUAL" TRUE EN %10.1 \*-K0\* FALSE %10.2 TRUE Manual "-SO" mode active %10.5 "-B1" TRUE Pushbutton\_ %11 4 manual\_ FALSE \*-53\* mode %11.5 Conveyor -54" -0 Enable\_OK FALSE motor manual %Q0.0 FALSE mode "-Q1" %10.0 Safety -A1" - shutoff\_active 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).



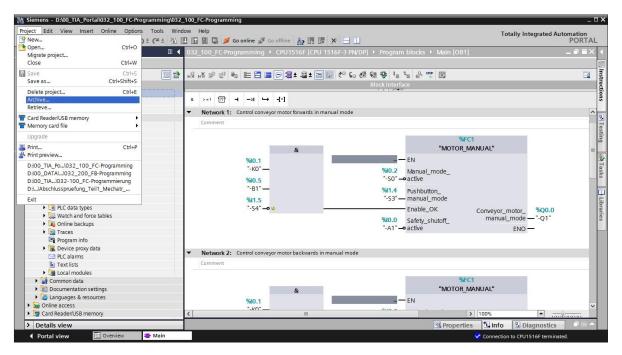
**Note:** The monitoring here is function-related and controller-independent. The actuation of sensors and the station status are shown here with TRUE or FALSE.

→ 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)

	vironment			
0	Dependency structure		1 Address	Details
1	🖀 Main		OB1	Main NW1 (Co
2	🖀 Main		OB1	Main NW2 (Co
		Tra	nsfer to *adjus	ted manually*

#### 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)



### 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) $\rightarrow$ -Q1 = 0	
5	Same as 3 but AUTO mode (-S0 = 1) $\rightarrow$ -Q1 = 0	
6	Same as 3 but switch off station (-K0 = 0) $\rightarrow$ -Q1 = 0	
7	Same as 3 but cylinder not retracted (-B1 = 0) $\rightarrow$ -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) $\rightarrow$ -Q2 = 0	
10	Same as 8 but AUTO mode (-S0 = 1) $\rightarrow$ -Q2 = 0	
11	Same as 8 but switch off station (-K0 = 0) $\rightarrow$ -Q2 = 0	
12	Same as 8 but cylinder not retracted (-B1 = 0) $\rightarrow$ -Q2 = 0	
13	Same as 8 but also activate manual mode conveyor forwards (-S3 = 1) $\rightarrow$ -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.

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	

### 9.3 Checklist – Exercise

## **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