

# Learn-/Training Document

Siemens Automation Cooperates with Education (SCE) | Version V15 and higher

**TIA Portal module 034-100** Basics of FC Programming with SIMATIC IOT2000EDU

siemens.com/sce



#### Matching SCE Trainer Packages for these Learn/Training Document

- SIMATIC IOT2020 with Intel Quark x1000, 512 MB RAM
   Order No.: 124-4037 Can be ordered from RS Components <u>rs-components.com</u>
   SIMATIC IOT2040 with Intel Quark x1020 (+Secure Boot), 1 GB RAM
- SIMATIC 1012040 with Intel Quark x1020 (+Secure Boot), 1 GB RAM Order No.: 6ES7647-0AA00-1YA2
- SIMATIC IOT2000EDU Software Controller executable on IOT2020 and IOT2040 Order No.: 6ES7671-0LE00-0YB0
- SIMATIC IO-Shield: SIMATIC IOT2000 Input/Output Module with 5 DI, 2 DO, 2 AI, ARDUINO Shield for IOT2020/2040
  - Order No.: 6ES7647-0KA01-0AA2
- 3rd Party IO-Shield: IKHDS Power Shield for IOT2020/2040 with 6 DI, 5 DO (RA), 1 DO (PWM), 2 AI, 1 AO

Order No.: 100301 Can be ordered from KAFTAN media UG kaftan-media.com/iot2000

#### SIMATIC STEP 7 Software for Training

- SIMATIC STEP 7 Professional V15 Single license Order no.: 6ES7822-1AA05-4YA5
- SIMATIC STEP 7 Professional V15 Classroom license (up to 6 users) Order no.: 6ES7822-1BA05-4YA5
- SIMATIC STEP 7 Professional V15 Upgrade license (up to 6 users) Order no.: 6ES7822-1AA05-4YE5
- SIMATIC STEP 7 Professional V15 Student license (up to 20 users) Order no.: 6ES7822-1AC05-4YA5

#### **Continued education**

For regional Siemens SCE continued education, get in touch with your regional SCE contact: <u>siemens.com/sce/contact</u>

#### Additional information regarding SCE

siemens.com/sce

#### Information regarding use

The SCE Learn-/Training Document 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 facilities 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, which 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 Learn-/Training Document 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 person: Roland Scheuerer roland.scheuerer@siemens.com.

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

Use for industrial customer courses is explicitly not permitted. We do not consent to commercial use of the Learn-/Training Document.

We would like to thank Michael Dziallas Engineering and all those involved for their support in creating this SCE Learn-/Training Document.

## **Table of contents**

1		al	
2		quirement	
3		quired hardware and software	
4		eory	
	4.1	Operating system and application program	8
	4.2	Organization blocks	9
	4.3	Process image and cyclic program processing	. 10
	4.4	Functions	. 12
	4.5	Function blocks and instance data blocks	. 12
	4.6	Global data blocks	. 14
	4.7	Library-compatible code blocks	. 15
	4.8	Programming languages	. 16
5	Tas	.k	. 17
6	Pla	nning	. 17
	6.1	EMERGENCY STOP	. 17
	6.2	Manual mode – Conveyor motor in manual mode	. 17
	6.3	Technology diagram	. 18
	6.4	Reference list	. 19
7	Stru	uctured step-by-step instructions	. 20
	7.1	Retrieving an existing project	. 20
	7.2	Creating a new tag table	. 21
	7.3	Creating new tags within a tag table	. 23
	7.4	Importing "Tag_table_sorting_station"	. 24
	7.5	Creating function FC1 "MOTOR_MANUAL" for the conveyor motor in manual mode	. 27
	7.6	Defining the interface of function FC1 "MOTOR_MANUAL"	. 29
	7.7	Programming FC1: MOTOR_MANUAL	. 32
	7.8	Programming organization block OB1 – Control conveyor forwards in manual mode	. 39
	7.9	Saving and compiling the program	. 44
	7.10	Downloading the program	. 45

#### Learn-/Training Document | TIA Portal Module 034-100, Edition 2018 | Digital Factory, DF FA

	7.11	Monitoring program blocks	. 46
	7.12	Archiving the project	. 48
	7.13	Checklist	. 49
8		rcise	
	8.1	Task – Exercise	. 50
	8.2	Technology diagram	. 50
	8.3	Reference list	. 51
	8.4	Planning	. 51
	8.5	Checklist – Exercise	. 52
9	Add	itional 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, we introduce **library-compatible** function and 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 Requirement

This chapter builds on the SIMATIC IOT2000 hardware configuration. The task can be realized with any shields that have a corresponding number of digital inputs and outputs. You can use the following project for this chapter, for example:

SCE\_EN\_014-101\_Hardware\_Configuration\_IOT2000.zap14

### 3 Required hardware and software

- 1 Engineering station: requirements include hardware and operating system (for additional information, see Readme on the TIA Portal Installation DVD)
- 2 SIMATIC STEP 7 Professional software in TIA Portal V15 or higher
- 3 SIMATIC IOT2000 controller, e.g. IOT2040 with MicroSD card and IO shield Note: The digital inputs should be fed out to a 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:

- Processing a warm restart
- Updating the process image input and the process image output
- 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 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 organization block 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 you have not assigned an OB to the event, the default system reaction is performed.

Table 1 shows examples for different start events for SIMATIC IOT2000. Possible OB number(s) and the preset system reactions that occur if the respective organization block (OB) is not present in the controller are also illustrated.

Start event	Possible OB number	Default system reaction
Startup	100	Ignore
Cyclic program	1	Ignore
Time-of-day interrupt	10	-
Maximum cycle time exceeded	80	STOP

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 (Q), 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:

- At the beginning of the cyclic program, the query is made as to whether or not the individual inputs carry voltage. This status of the inputs is stored in the process image input (PII). The information 1 or "High" is hereby stored for energized inputs and the information 0 or "Low" for de-energized inputs.
- The CPU then executes the program stored in the cyclic organization block. Then, for the required input information, the CPU accesses the previously read process image input (PII) and the results of logic operation (RLOs) are written to a so-called process image output (PIQ).
- 1. At the end of the cycle, the **process image output** (**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. Store the status of the inputs in the PII.



Figure 2: Cyclic program processing

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

- Mathematical functions that report 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 also 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. They 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 or
- Whenever information must be saved in the program, such as pre-selection of the operating mode with a button.

Function blocks are always executed if 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 and 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 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 to the cycle OB, but is only suitable for very simple programs.

*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 preferably be created. 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 and function blocks for SIMATIC S7-1200 are Function Block Diagram (FBD), Ladder Diagram (LAD) and Structured Control Language (SCL).

The Function Block Diagram (FBD) programming language is presented below.

FBD is a graphical programming language. The representation is based on electronic circuit 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 that enables you to 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" has signal state "1". When the box is processed correctly, enable output "ENO" has signal state "1". If an error occurs during the processing, the "ENO" enable output is reset. If the 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: Control conveyor motor forwards 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 to be moved to FCs and which are to run in OB 1 is planned below.

#### 6.1 EMERGENCY STOP

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 ensures 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
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, the enable is set and the safety shutoff is not active.

### 6.3 Technology diagram

Here, you see the technology diagram for the task.





Schalter der Sortieranlage		Automatikbetrieb		Handbetrieb / Manual mode
Switches of sorting station		Automatic mode		-S3 Tippbetrieb -M1 vorwärts/
-P1 ein/on		-P5 gestartet/started		
-Q0 Hauptschalter/Main switch		-S1 Start/start		-S4 Tippbetrieb -M1 rückwärts/
-P4 aktivient/active				Manual -M1 backwards
-A1 NOTHALT/Emergency stop		-S2 Stopp/stop		-P7 ausgefahren/extended
-P2 Hand/manual -P3 Auto/auto				-S6 Zylinder -M4 ausfahren/ cylinder -M4 extend
-S0 Betriebsart/operating mode				-P6 eingefahren/retracted
				cylinder -M4 retract
	1 1		1	

Figure 9: Control panel

#### 6.4 **Reference list**

The following signals are required as operands for this task.

DI	Туре	Identifier	Function	NC/NO
l 101.4	BOOL	-A1	Return signal emergency stop OK	NC
I 101.3	BOOL	-K0	Station "ON"	NO
l 101.2	BOOL	-B1	Sensor cylinder -M4 retract	NO
I 101.1	BOOL	-S3	Pushbutton manual mode conveyor -M1 forwards	NO
I 100.4	BOOL	-S4	Pushbutton manual mode conveyor -M1 backwards	NO

DQ	Туре	Identifier	Function	
Q 101.0	BOOL	-Q1	Conveyor motor -M1 forwards fixed speed	

#### Legend for reference list

DI	Digital input	DQ	Digital output
AI	Analog input	AQ	Analog output
I	Input	Q	Output

- L Input
- NC Normally Closed
- NO Normally Open

### 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 Retrieving an existing project

→ Before we can start programming the function (FC) "MOTOR\_MANUAL", we need a project with a hardware configuration.

(e.g. SCE\_DE\_014\_101\_Hardware\_Configuration\_IOT2000.zap14). To retrieve an existing project that has been archived, you must select the relevant archive with  $\rightarrow$  Project  $\rightarrow$  Retrieve in the project view. Confirm your selection with "Open". ( $\rightarrow$  Project  $\rightarrow$  Retrieve  $\rightarrow$  Select a .zap archive  $\rightarrow$  Open)



Retrieve archived	project						>
· 🛧 🗄	] → This l	PC > Documents >			~ ē	Search Documents	٩
Organize 👻 Ne	ew folder						?
Desktop	* ^	Name	Date modified	Туре	Size		
🕹 Downloads	*	Automation	6/13/2018 12:35 PM	File folder			
🗄 Documents	1	Automatisierung	6/20/2018 8:09 AM	File folder			
Pictures	*	Benutzerdefinierte Office-Vorlagen	6/7/2018 11:08 AM	File folder			
Nextcloud	*	Siemens	5/2/2018 1:23 PM	File folder			
IoT2000		SCE_EN_014-101_Hardware Configuration_IOT2000_R1806	6/21/2018 9:26 AM	Siemens TIA Porta	216	KB	
IoT2000							
IoT2000_EN							
IoT2000_FC							
OneDrive							
This PC							
Network	~						
	File nam	ne:			~	TIA Portal project archive (*.za	ar v
						0	
						Open Cance	1

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



#### 7.2 Creating a new tag table

→ In the project view, navigate to the  $\rightarrow$  PLC tags of your controller and create a new tag table with a double-click on  $\rightarrow$  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).



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

🖹 🔚 Save project 📕 🐰 🗐 🗎 🗙 🏷 🛨			and the second							PORT
oject tree		014_10	01_IOT2040 → CPU_IO	T2040 [IOT2000ED	U] • PLC tags	🕨 Tag tab	le_sortir	ng_stati	on [0]	_ • •
Devices									🕣 Tags	User constants
	🔲 📑	1	🗎 🕆 🙄 🛍							E
		Tag	table_sorting_station							
014_101_IOT2040	^		Name	Data type	Address	Retain	Acces	Visibl	Comment	
🌁 Add new device		1	<add new=""></add>							
Devices & networks										
CPU_IOT2040 [IOT2000EDU]										
Device configuration										
Q Online & diagnostics	=									
Program blocks										
Technology objects										
External source files										
💌 🛺 PLC tags										
🍇 Show all tags										
📑 Add new tag table										
🌿 Default tag table [0]										
lag table_sorting_station [0]										
PLC data types										
Watch and force tables										
Device proxy data										
📴 Program info										
🖙 PLC supervisions & alarms										
PLC alarm text lists										
Local modules	~									
Details view										
Details view										
Name Data type Details	Co									
itanic bata get betan		<	1							

#### 7.3 Creating 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 (→ <Add> → -Q1 → Enter).

014	014-101_IOT2040 + CPU_IOT2040 [IOT2000EDU] + PLC tags + Tag table_sorting station [1]													
	Tags User constants													
<b>*</b>	🚽 🔮 🖶 😤 🛍													
Т	ag t	table_sortir	ng station											
		Name		Data type	Address	Retain	Acces	Visibl	Comment					
1	-00	-Q1		Bool	%10.0			<b></b>						
2		<add nev<="" th=""><th>V&gt;</th><th></th><th></th><th></th><th><b>~</b></th><th><b>V</b></th><th></th><th></th></add>	V>				<b>~</b>	<b>V</b>						

→ Change the address to %Q101.0 by entering this directly or by clicking the drop-down arrow to open the Address menu. Change the operand identifier to A and the address to 101.

Confirm with Enter or by clicking on the check mark. ( $\rightarrow$  %I0.0  $\rightarrow$  Operand identifier  $\rightarrow$  A  $\rightarrow$  Address $\rightarrow$  101  $\rightarrow$   $\blacksquare$ )

014-	101	_IOT2040 →	CPU_IOT2040 [I	OT2000EDU	•	PLC tags	Tag tak	ole_s	sorting	g statio	n [1]	_ ∎∎×
											🕣 Tags	User constants
1 - A	š,	🖻 🗄 🛸 🕯	Ť									<b>_</b>
Ta	ng t	able_sorting	station									
		Name		Data type		Address	Retain	A	cces	Visibl	Comment	
1	-	-Q1		Bool		%I0.0	-		$\checkmark$			
2		<add new=""></add>				Opt		d typ Idres	e:	11	<ul> <li></li> <li><th></th></li></ul>	

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

014	101	_IOT2040   CP	U_IOT2040 [IOT2000EDU]	<ul> <li>PLC tags</li> </ul>	• Tag table	_sorting	g station	[1] <b>_ Ⅲ■</b> ×							
								Tags 🗉 User constants							
🥩 :	2	fii 😤 🛨 🗲													
Т	Tag table_sorting station														
		Name	Data type	Address	Retain	Acces	Visibl	Comment							
1	-00	-Q1	Bool	🗉 %Q101.0	-	<b></b>	<b></b>	conveyor motor -M1 forwards fixed speed							
2		<add new=""></add>				<b>V</b>	<b>V</b>								

→ 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.... Enter the comment "conveyor motor M1 backwards fixed speed" and change the address to %Q100.7. (→ <Add> → -Q2 → Enter → Comment → Conveyor motor M1 backwards fixed speed)

014	014-101_IOT2040														
								Tags 🗉 User constants							
∌	🥐 🗄	) 🗗 🚏 🛍													
٦	Tag table_sorting station														
	N	lame	Data type	Address	Retain	Acces	Visibl	Comment							
1		-Q1	Bool	%Q101.0			<b></b>	conveyor motor -M1 forwards fixed speed							
2	-	-Q2	Bool	%Q100.7			<b></b>	conveyor motor -M1 backwards fixed speed							
З		<add new=""></add>				<b>V</b>	<b>V</b>								

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

 $\rightarrow$  To insert an existing symbol table, click on the  $\blacksquare$  "Import" icon in the tag table toolbar.

(→ 皆	Import)
------	---------

014	014-101_IOT2040 > CPU_IOT2040 [IOT2000EDU] > PLC tags > Tag table_sorting station [2]														
	Tags 🗉 User constants														
<b>*</b>	<b>.</b>		🕂 🚏 🗊												
1	Tag table_sorting station														
		Nam	2	Data type	Address	Retain	Acces	Visibl	Comment						
1		ı - (	21	Bool	%Q101.0		$\checkmark$	$\checkmark$	conveyor motor -M1 forwards fixed speed						
2	-	- I	22	Bool	%Q100.7		$\checkmark$		conveyor motor -M1 backwards fixed speed						
3		<	Add new>				$\checkmark$	<b>V</b>							

→ Select the desired symbol table (e.g. in .xlsx format) and confirm the selection with "Open". ( $\rightarrow$  SCE\_DE\_020-100\_Tag\_table\_sorting\_station\_IO-Shield ...  $\rightarrow$  Open)

🖳 Open				×	:
$\leftarrow$ $\rightarrow$ $\checkmark$ $\uparrow$ $\square$ $\flat$ This	s PC > Documents > Siemens		✓ Ö Search Sier	mens 🔎	
Organize 🔻 New folder	r			EE 🕶 🔟 📀	
🖈 Quick access	Name	Date modified	Туре	Size	
	Simatic	5/2/2018 1:23 PM	File folder		
OneDrive	SCE_EN_020-100_Tag_table_sorting_statio	6/21/2018 10:05 AM	Microsoft Excel-Ar	10 KB	
💻 This PC					
🔿 Network					
-					
File na	me: SCE_EN_020-100_Tag_table_sorting_station_I0	D-Shield.xlsx	~ All support	ted files (*.xlsx;*.sdf;*. $\vee$	
			Oper		

 $\rightarrow$  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. (0032:000001) 🛛 🗙 🗙
	Import completed successfully.
	Detailed information is shown in the import log file.
	Click here to view the log file.
	OK

→ Tags that already exist in the system are updated and missing tags are added. Tags that exist in the project but not in the import file are retained.

 $\rightarrow$  You now have a complete symbol table of the digital inputs and outputs in front of you. Now save your project under the name 034-100\_FC-Programming.

 $(\rightarrow \text{Project} \rightarrow \text{Save as } \dots \rightarrow \text{034-100}_{\text{FC-Programming}} \rightarrow \text{Save})$ 

₩ Siem	ens - F	=:\00_T	IA_Porta	al\014-1(	01_IOT20	40\014	I-101_	IOT2	040	
Project	Edit	View	Insert	Online	Options	Тоо	ls Wi	indow	/ Help	
New Open Migra		ect			Ctrl+O	) ± (		101_	] [f] [ _IOT20	∎ <b>Г</b> 40
Close					Ctrl+W					
📘 Save					Ctrl+S	1 🗈	🥩 :	*	⇒ ⊶	00
Save	as			Ctr	l+Shift+S				able_so	~
Delet	e proje	ct			Ctrl+E	~			Name	
Archiv							1	-	-Q1	
Retrie	eve						2	-	-Q2	
Multiu	user				•		з	-	-A1	
The Card	Reader/	USB me	emory		•		4	-	-K0	
	ory card		-		•		5		-B1	
Start	basic in	tearity	check				6	-	-S3	
					cul a		7		-54	
Print.	 preview	,			Ctrl+P		8		<add< td=""><td>new</td></add<>	new
								1		
Ma Save current					7		×			
Save in:		A_Portal	^	~						
Quick access	Name	101_IOT2040	)		Date modifi 6/21/2018 1		Type File folder			
Desktop										
Libraries										
This PC										
Network	<						>			
	File name:		4-100_FC-Progr		~		Save			
	Save as typ	pe: TI	A Portal projects	3	~		Cancel			

For unrestricted use in educational and R&D institutions. © Siemens AG 2018. All rights reserved. SCE\_EN\_034-100 FC-Programmierung IOT2000EDU\_R1806.docx

# 7.5 Creating 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 here.



→ Rename your new block to: "MOTOR\_MANUAL" Then set the language to FBD and allow 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)

Add new block			
Nesser			
Name: MOTOR_MANUAL			
	Language:	FBD	
-OB	Number:		
Organization block		O Manual	
DIOCK		<ul> <li>Automatic</li> </ul>	
	Descriptions		
FB	Description:		
Function block	Functions are code	blocks or subroutines without dedicated memory.	
FC			
Function			
DB			
Data block			
	more		
> Additional information	ation		
Add new and open			Add
Mud new and open			Aud

#### 7.6 Defining 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.

034	1-10	00_1	FC-Pro	gram	nminę	g ▶ (	CPU_I	IOT2	040	IOT2	000	EDU] →	Prog	ram l	olocks	s ► N	юто	R_M	ANU	AL [F	C1]				-	. 🖪 I	×
<b>B</b>	ьă	-	` (		EF			<b>8</b> ±	2 ±	별±		😥 🥙	<b>6</b> 0 (	a (1	•	Ç≞ I	= <i>X</i> =	Ģ	¢I	e l		6	 				
						_	_		_													-					_
		Nam	_					Data	type			Offset	Defa	ult valu	Je	Comr	nent										_
1	-	• 1	nput																								^
2		•	<add< td=""><td>new</td><td>&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><th></th><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></add<>	new	>																						
З	-	• (	Output																								≡
4		•	<add< td=""><td>new</td><td>&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><th></th><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></add<>	new	>																						
5			nOut																								
6			<add< td=""><td>new</td><td>&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><th></th><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></add<>	new	>																						
			Temp																								
8	_	•	<add< td=""><td>new</td><td>&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><th></th><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>~</td></add<>	new	>																						~
	-		_			1								<b>A</b>	•								 				
8	>	=1	??	-	-0	<b>→</b>	-[=]																				
-	Bloc	k ti	tle:																								
		nent																									
_																											
•	N	letw	ork 1:																								
	Co	omm	nent																								
																					100%			-		Į	

→ 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)

03	4-1(	00_	_FC-Programming   CPU_IOT2	040 [IOT2000EDU]	🕨 Progi	ram blocks 🕨 I	MOTOR_MANUAL [FC1]	_ ∎∎×
юŝ	ы	Í	🖗 🔮 🐛 🗮 🚍 💬 冶 ±	🏽 ± 🕲 ± 🖃 😥	¢⇔ ⊊⊕ ¢	l 😣 🖷 🍄 (	● 当 目 非 〒 〒 目 き 目 市 目 目 ■	<b>_</b>
	MO	тс	R_MANUAL					
		Na	me	Data type	Offset	Default value	Comment	
1	-	•	Input					^
2		•	<add new=""></add>					
3	-00	•	Output					≡
4	-	•	Conveyor_motor_manual_mode	Bool 🔳			Control of the conveyor motor in manual mode	_
5		•	<add new=""></add>					
6	-	•	InOut					
7		•	<add new=""></add>					
8	-00	•	Temp					
9			<add new=""></add>			▲   ▼		~
						Program		

- → Add parameter #Pushbutton\_manual\_mode as the input interface under Input and confirm the entry with the Enter key or by exiting the text box. Data type "Bool" is assigned automatically. This will be retained. Then enter the corresponding comment "Pushbutton manual mode conveyor on". (→ Pushbutton\_manual\_mode → Enter → Bool → Pushbutton manual mode conveyor on)
- → Now add under Input the parameters #Pushbutton\_manual\_mode, #Enable\_OK and #Safety\_shutoff \_active as additional binary input parameters and check their data types. Then add meaningful comments.

-10	)0_	_FC-Programming      CPU_IOT2	040 [IOT2000EDU	] 🕨 Prog	ram blocks 🕨	MOTOR_MANUAL [FC1]	_ ∎∎×
ьX	1	🖗 🔮 🐛 🖿 🚍 💬 📲 ±	🏖 ± 😫 ± 🖃 😥	¢0 60 6	🗄 🐏 🐂 😍	두 표 표 책 날 날 때 야 한 야 않 않 않 않 않 않 않 않 않 않 않 않 않 않 않 않 않	-
<b>NO</b>	то	R_MANUAL					
	Nai	me	Data type	Offset	Default value	Comment	
1	•	Input					^
1	•	Pushbutton_manual_mode	Bool			Pushbutton manual mode conveyor on	
1	•	Enable_OK	Bool			All enable conditions OK	_
	•	Safety_shutoff_active	Bool 🔳			Safety shutoff active e.g. emergency stop operated	=
	•	<add new=""></add>					
1	•	Output					
	•	Conveyor_motor_manual_mode	Bool			Control of the conveyor motor in manual mode	
	•	<add new=""></add>					
11	•	InOut					
	•	<add new=""></add>					
1	•	Temp					
	•	Add nour			A   V		
			<ul> <li>KX IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII</li></ul>	Image: Second	Image: Second	Image: Second	Name     Data type     Offset     Default value     Comment       Imput     Imput     Imput     Imput     Imput     Imput       Imput     Imput     Imput     Imput     Imput     Imput     Imput       Imput </td

→ For purposes of program documentation, enter 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)

03	034-100_FC-Programming → CPU_IOT2040 [IOT2000EDU] → Program blocks → MOTOR_MANUAL [FC1]														
ю	ы	1	) B	11. 11.	E		= 💬 署	- 🖀 ± 🕲 :	• 🖃 😥	(° 6	🥶 🧐 💝 📢	I <sub>=</sub> <sup>1</sup> <sub>=</sub> <b>\$</b> <sub>1</sub> ¢I ⊕ ∞			3
			R_MA												
		Nam	ne					Data type		Offset	Default value	Comment			
1	_		Input												^
2	-				-	nanua	al_mode	Bool				Pushbutton manual mode	conveyor on		=
3	-			ble_C				Bool	_			All enable conditions OK			
4		•		ety_sh		f_acti	ve	Bool		ļ		Safety shutoff active e.g. e	mergency stop operated		
5		•		ld nev	>										
6			Output												
7					-	tor_m	nanual_mod	e Bool				Control of the conveyor mo	otor in manual mode		~
8		•	<ac< td=""><td>ld nev</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></ac<>	ld nev											
8	>	= 1	??	н	-0	( <b>-</b>	→ -[=]								
-	Blog	ek ti	itle	Motor	cont	trol in	manual mo	le .							
									ual mode	is operate	d, the enable				
c	ondi	ition	s are g	rante	d an	d the	safety shute	ff is not activ			a, are enable				
	onve	eyor	_moto	r_man	nual_	mode	e is activate								
•	N	letv	vork 1	: Co	ntro	l of th	e conveyor r	notor in man	ual mode						
	C	omn	nent												
													100%	<b>•</b>	

#### 7.7 Programming 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. Below, you will learn about the various options for inserting logic blocks.

& >:	1 ??	-	-01	↦	-[=]
------	------	---	-----	---	------

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

		00_TIA_Portal\034-100_FC-Pro			amming								_ 🗆 ×
		/iew Insert Online Options				a	in mu					ated Automation POR	1
_		project 进 🐰 🗓 🗎 🗙 🕨								_	-14		
D 03	34-100_FC	-Programming   CPU_IOT2	040 [IOT2000EDI	J] ▶ Prog	ram blocks 🕨	MOTOR_M	ANUAL [FC	31]	_ # = ×	ln,	structions	<b>-</b> I	
										0	otions		
нđ	i 1≦ Ka 8	🖉 🔍 🗄 🚍 💬 📇 ±	🖀 ± 🔚 ± 🖃 😥	CO 60 (	/通 🤬 🤣 📢	I <sub>=</sub> 1 <sub>=</sub> 6∥	el 🔗 🖻			Г	the tee	🔁 🥸 🗾	Instructions
2	MOTOR_	MANUAL								5	Favorites		Ē
1	Name		Data type	Offset	Default value	Comment				-	Basic instructions		÷
8 1	🕣 🔻 Inp	ut							^	Na		Description	- 5
2 2		Pushbutton_manual_mode	Bool			Pushbutto	n manual mo	de conveyor on	_		General	Description	^ -
ā 3		Enable_OK	Bool			All enable	conditions O	к	_		Bit logic operations		<u> </u>
4		Safety_shutoff_active	Bool	]		Safetyshu	offactive e.g	, emergency stop o	perated	L .		AND logic operati	e e
5		<add new=""></add>								L .	€ >=1	OR logic operatio	
6										L .	E x	EXCLUSIVE OR Io.	- <u>-</u>
- 7			Bool			Control of t	he conveyor	motor in manual m	ode		▲Ē -[=]	Assignment [Shif	
8		<add new=""></add>		-					•		[1] -[R]	Reset output	Tasks
	a >=1 f	?]o -→ -f=1									[] -[S]	Set output	as S
		-									E SR	Set/reset flip-flop	ŝ
											E RS	Reset/set flip-flop	
					d, the enable						- P -	Scan operand for	Libraries
			is not activated the o	utput							- N -	Scan operand for	🛉
11	· -										P_TRIG	Scan RLO for posi	ar
	Networ	k 1: Control of the conveyor mo	otor in manual mode							L .	N_TRIG	Scan RLO for neg.	S
	Commer	at.									Timer operations		
											End of the second se		
								-			Comparator operations		
											主 Math functions		
											🔁 Move operations		
											😽 Conversion operations		$\sim$
										ł	Program control opporti		
										5	Extended instructions		_
										Ľ.			_
		Imput       Pushbutton_manual_mode         ■ Fushbutton_manual_mode       Bool         ■ Enable_OK       Bool         ■ Enable_OK       Bool         ■ Conveyor_motor_manual_mode       Bool         ■ codd newo       Enable_OK         ■ codd newo       Enable_OK         Image:								P	Technology		-
	_			_						<b>`</b>	Communication		-
					<u>Q</u> P	roperties	Info (	追 🖫 Diagnostic	s D E A	>	Optional packages		
	Portal vi	ew Overview	👆 Tag table_so	🔹 мото	DR_MA						Project saved under F:\00_TI	A_Portal\0	

→ Now use drag & 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 clicking on it at the blue symbol 
(→ 
Conveyor\_motor\_manual\_mode)

034-100_FC-Programming      CPU_IOT2	040 [IOT2000EDU	]      Program blocks	MOTOR_MANUAL [FC1]	_ ⊫∎×					
🛨 🖹 🖳 🗮 🚍 💭 🕾 נא א	22 ± 122 ± 🖂 😥	(ං 💪 🖽 🖷 🖶 🤣	🖬 표 部 님 놀 🖩 왜 이 어						
MOTOR_MANUAL									
Name	Data type	Offset Default value	Comment						
1 🕣 🔻 Input				^					
2 🔄 Pushbutton_manual_mode	Bool		Pushbutton manual mode conveyor o	n 📃					
3 🕘 = Enable_OK	Bool		All enable conditions OK	=					
4 🕣 💶 Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency s	top operated					
5 <a>Add new&gt;</a>									
6 🕣 🔻 Output									
7 Conveyor_motor_manual_mode	Bool 🔳		Control of the conveyor motor in man						
8 <add new=""></add>				~					
& >=1 [??] → -0  → -[=]									
<ul> <li>Block title: Motor control in manual mode</li> </ul>				^					
<ul> <li>Conveyor motor in manual mode: If the pusht conditions are canted and the safety shutoff</li> </ul>									
Conveyor_motor_manual_mode is activated	is not activated the ot	nput							
				=					
▼ 🔀 Network 1: Control of the conveyor me	otor in manual mode			=					
Comment									
37.75									
< <u>1.</u> >									
			1007	Y					
			100%	▼Y					

→ 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 & drop to move input parameter #Pushbutton\_manual\_mode to "<??.?>" at the left side of the assignment block. (→ @Pushbutton\_manual\_mode)

034-100_FC-Programming  CPU_IOT2	040 [IOT2000EDU	] 🕨 Program blocks 🕨	MOTOR_MANUAL [FC1]	∎ <b>■</b> ×
* x -* -* = = = = = = = = = = =		40 c All Co	(= 표 표 뷰 날 빌 에 에 이 아 약 🛍	
	-ar - ⊡≋r	(~ %3 (= %= ⇒ ❤	<b>→</b> = <u>→</u> = →= += '= '=   N/ <b>→</b> N C*   O*   >   UE	
Name	Data type	Offset Default value	Comment	
1 🔄 🔻 Input	Data type	Delaurevalue	connient	^
2 Teshbutton_manual_mode	Bool		Pushbutton manual mode conveyor on	
3 - Enable OK	Bool	J	All enable conditions OK	- 1
4 🕣 🛚 Safety_shutoff_active	Bool		Safety shutoff active e.g. emergency stop operated	
5 S Add new>				
6 🕣 🔻 Output				
7 - Conveyor_motor_manual_mode	Bool		Control of the conveyor motor in manual mode	
8 <a>Add new&gt;</a>				×
≥ > 1 1??? 0 [=]				
<ul> <li>Block title: Motor control in manual mode</li> </ul>				^
<ul> <li>Convey r motor in manual mode: If the pusht conditions are granted and the safety shutoff Conveyor_motor_manual_mode is activated</li> </ul>				
Net vork 1: Control of the conveyor me	otor in manual mode			=
Comr ent				
#Conveyor_				
motor_manual_				
mode				
=				
				~
			100%	

→ 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 #Pushbutton\_manual\_mode is already connected, so that the input line has a blue background.



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



→ Double-click on the second input of the & link <??.?>. Then enter the letter "F" in the field that appears in order to see a list of available tags starting with "F". Now click on the tag #Enable\_OK and apply it with → Enter. (→ & block → <??.?> → F → #Enable\_OK → Enter)

034	-100_	FC-Pr	ograr	nmi	ing	•	CPU_	lotz	2040	[IOT2	2000ED	0U] ▶	Progra	ım blo	cks →	MC	TOF	₹_M	ANU	AL	[FC1]	l	_ 19 1	×
ю́я	<mark>⊮X</mark> ≣	è	ii,	E		}	Ð	<b>3</b> ±	- <u>2</u> 1	볞	L 🖃 🛐	? ୯୦	<b>6</b> 0 🖑	<b>6</b>	<b>پ</b>	E I =	$z^{\equiv}$	Ģ	କା	Ŷ	- 00 			
													k interfa	ace										
8	> = 1	??	-	-0	я	↦	-[=	1																
<b>▼</b> B	lock t	itle:	Motor	con	trol	in m	anua	l mod	e															
	Net	work 1	_						otor ir	n man	ual mode	e												
E	m	Pushbi anual_	mode		*	&			motor	nveyo _man node =							1							
•	■ #Er	able_(	ЭК					Bool					All e	nable (	ond	^								
L																~		1	00%				▼	

**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 & drop from the interface description should therefore preferably be used.

→ To ensure that the output is only activated when the safety shutoff is not active, the input tag #Safety\_shutoff\_active should be logically combined with the AND logic operation. To do this, click on the yellow star <sup>3</sup>/<sub>2</sub> of the AND block to add another input.



→ Add the input tag #Safety\_shutoff\_active to your newly created input of the AND element.



→ Negate the input connected to parameter #Safety\_shutoff\_active by selecting it and clicking


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

	MO	10	R_MANUAL					
	1	Nan	ne	Data type	Offset	Default value	Comment	
1		•	Input					
ŀ		•	Pushbutton_manual_mode	Bool 🔳	J		Pushbutton manual mode conveyor on	
ľ	-	•	Enable_OK	Bool			All enable conditions OK	
ľ		•	Safety_shutoff_active	Bool			Safety shutoff active e.g. emergency stop operated	
l		•	<add new=""></add>					
			Output					
1			Conveyor_motor_manual_mode	Bool			Control of the conveyor motor in manual mode	
		•	<add new=""></add>					
12			InOut					
E	> Bloc onve	>=1 ckt eyor	Impose         Impos	outton_manual_mode	is operate	ed, the enable		
E	> Bloc onve ondit onve	>=1 ckt eyor itior eyor	Image: The second s	outton_manual_mode is not activated the ou	is operate			
Co	> Bloc onve ondit onve	>=1 ckt eyor itior eyor	Image: The second s	outton_manual_mode is not activated the ou	is operate			
E	> Bloc onve ondit onve	>=1 ckt eyor itior eyor	Image: The second s	outton_manual_mode is not activated the ou	is operate			
E	> Bloc onve ondit onve	>=1 ckt eyor itior eyor	Image: The second s	outton_manual_mode is not activated the ou	is operate			
E	> Bloc onve ondit onve	>=1 ckt eyor itior eyor	Image: The second s	outton_manual_mode is not activated the ou	is operate Itput			
E	> Bloc onve ondit onve	>=1 ckt eyor itior eyor	Image: The second s	outton_manual_mode is not activated the ou otor in manual mode #Conveyo motor_manu	is operate itput			
E	> Bloc onve ondit onve	>=1 ckt eyor itior eyor	Image: The second s	outton_manual_mode is not activated the ou otor in manual mode #Conveyo	is operate itput			
E	> Bloc onve onve N Cc	>=1 ck t eyor itior eyor	Image: The second s	outton_manual_mode is not activated the ou otor in manual mode #Conveyo motor_manu	is operate itput			

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

Properties			E
MOTOR_MANUAL [FC1]		🔍 Properties	🗓 Info 🔒 🗓 Diagnostics
General Texts			
General	C		
Information	General		
Time stamps			
Compilation	Name:	MOTOR_MANUAL	
Protection	Type:	FC	
Attributes			
	Language	FBD	
Þ	Number	LAD FBD	
		STL	
		<ul> <li>Automatic</li> </ul>	

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

034-100_FC-Programming ► CPU_IOT20	040 [IOT2000EDU	] 🕨 Prog	ıram blocks → I	MOTOR_MANUAL [FC1]	_ ₪■×
kǎ kǎ ở ở ề ♣, 눈 드 드 등 위 봄 MOTOR MANUAL	2 ± 12 ± 🖃 😥	¢0 \$0 (	a a s	i <sub>≡</sub> <sup>1</sup> <sub>≡</sub> <b>6</b> ,	
Name	Data type	Offset	Default value	Comment	
1 💷 🔻 Input					~
2 📶 = Pushbutton_manual_mode	Bool			Pushbutton manual mode conveyor on	
3 📶 = Enable_OK	Bool			All enable conditions OK	=
4 📶 = Safety_shutoff_active	Bool			Safety shutoff active e.g. emergency st	op operated
5 Add new>					
6 📶 🔻 Output					
7 🕣 = Conveyor_motor_manual_mode	Bool			Control of the conveyor motor in manua	al mode
8 <li>Add new&gt;</li>					
9 🕣 🔻 InOut			•		~
H → H → → 1     Block title: Motor control in manual mode     Conveyor motor in manual mode: If the pushb     conditions are granted and the safety shutoff i     Conveyor_motor_manual_mode is activated	utton_manual_mode s not activated the ou		d, the enable		
<ul> <li>Network 1: Control of the conveyor model</li> </ul>	tor in manual mode				
Comment #Pushbutton_ manual_mode #Enable_OK	#Safety_shu active	itoff_		#Conveyor_ motor_manual_ mode ( )	
				120%	▼ <u> </u>

# 7.8 Programming organization block OB1 – Control conveyor forwards in manual mode

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

 $(\rightarrow CPU\_IOT2040[IOT2000EDU] \rightarrow Program blocks \rightarrow Main [OB1] \rightarrow Switch programming language \rightarrow FBD)$ 

恐	Siemens - F:\00_1	TIA_Portal\034-100_FC-Pr	ogramming\034	100_FC-Programming	_ ¤ ×
		Insert Online Option			Totally Integrated Automation
2	📑 🔡 Save proje	ct 📕 🐰 🗎 🗎 🗙 🛛	ີງ± (≃! ± 🖥	🗓 🗓 🖳 🧖 Go online 🖉 Go offline 🕌 🌆 🖪 👫 🧩 🚍 🛄 <search in="" project=""> 👫</search>	PORTAL
	Project tree	□ ◀			Tasks 📑 🔳 🕨
	Devices				Options
	, Bi	🔲 🖻			Options
					✓ Find and replace
		configuration			
tar	😵 Online 🗸 🗸	& diagnostics			Find:
		new block			
	🛄 Ma	Open			
	- MO	14	Ctrl+X		Match case
	<ul> <li>Techno</li> <li>Extern</li> </ul>	The course	Ctrl+C		Find in substructures
	▼ 🕞 PLC ta	📋 Paste	Ctrl+V		Find in hidden texts
	👆 Sho	🗙 Delete	Del		Use wildcards
	📑 Ado		F2		Use regular expressions
	🍯 Sta		•		Down
		Download to device Go online	Ctrl+K		O Up
	🕨 🥘 Watch	🛃 Go offline	Ctrl+M		Find
	Device	🚇 Quick compare	•		
			Ctrl+F		Replace with:
		Generate source from b	olocks 🕨	🔍 Properties 🔛 Info 🔒 💟 Diagnostics 💷	Whole document
ľ	<	X Cross-references	F11		From current position
	✓ Details view	Cross-reference informa	tion Shift+F11		Selection
		Call structure			
				sroperties' available.	Replace Replace all
	Name	Switch programming la	nguage 🕨	STL be shown at the moment. There is either no object selected or the selected object does LAD ayable properties.	
		Know-how protection		FBD	
		Print Print preview	Ctrl+P		
	<	Reperties	Alt+Enter		> Languages & resources
	Portal view	Overview	Ant+Enter	🔝 🗸 Project	t saved under F:\00_TIA_Portal\0

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



→ Assign Network 1 the name "Control conveyor motor forwards in manual mode" (→ Network 1:... → Control conveyor motor forwards in manual mode)



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

Pr	Siemens - F:\00_TIA_Portal oject Edit View Insert F 🍞 📮 Save project 💷 💥	Online Options	Tools Window		
	Project tree	□ ◄		rogramming → CPU_IOT2040 [IOT2000EDU] → Program blocks → Main [OB1]	
	Devices		க்கு≇≇∎	▲ 는 큰 프 ☞ 웹 ± 웹 ± 필 핥 안 % 생 행 장 두 님 님 위 산 & ♡ & _ ⊒	" Instructions
2				Block interface	TT I
ammir	<ul> <li>034-100_FC-Programmin</li> <li>Add new device</li> </ul>	ng <u>^</u>	& >=1 [??]	-1 -0 L> -1=]	tions
log	Devices & networks		▼ Block title: "M	"Main Program Sweep (Cycle)"	
J.	▼ 🛅 CPU_IOT2040 [IOT20		Comment		⇒ Testing
로	Device configurati		Network 1:	1: Control conveyor motor forwards in manual mode	est
	<ul> <li>Program blocks</li> </ul>	ues .	Comment		ing
	Add new block	. –	Comment		
	💶 Main [OB1]		-		👔 Tasks
	MOTOR_MANU/	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA		MOTOR_MANUAL [FC1]	Tas
	Technology object				ks
	External source fil	es			
	▼ 📜 PLC tags 3 Show all tags				
	Add new tag tags	able			Libraries
	Default tag table				arie
	Tag table_sort				S
	PLC data types	-			
	Watch and force to				
	<	>		100%	-
	> Details view		-	🔄 Properties 🚺 Info 🚺 🗓 Diagnostics 👘 🖡 🗕 🗠	
	🔹 Portal view 🛛 🧮	Overview	📲 Main (OB1)	📑 🧊 The programming language of the sele	

 $\rightarrow$  A block with the interface you defined and connections EN and ENO is inserted in Network 1.



 $\rightarrow$  To insert an AND before input parameter "Enable\_OK", select this input and insert the AND

by clicking the $\overset{\clubsuit}{}$ icon in your logic toolbar. ( $ ightarrow \overset{\clubsuit}{}$ )	
034-100_FC-Programming  CPU_IOT2040 [IOT2000EDU]  Program blocks  Main [OB1]	<b>⊫ ■</b> ×
D logic operation [Shift+F2] eep (Cycle)	^
Comment  Comment  Comment  Comment  KFC1  MOTOR_MANUAL"   Pushbutton_ manual_  Conveyor_ mator_  Conveyor_ mator_ Conveyor_ mator_ Conveyor_ mator_ Conveyor_ mator_ Conveyor_ mator_ Conveyor_ mator_ Conveyor_ Con	1
Safety	×
100%	



 $\rightarrow$  Click on the yellow star  $\stackrel{3}{\ast}$  of the AND element 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 drag the desired global tag from the Details view to the interface of FC1 (→ Tag\_table\_sorting\_station → Details view. → -S3 → Pushbutton\_manual\_mode)



Or enter the starting letters (e.g. "S") of the desired global tag for <??.?> and select the global input tag "-S3" (%I101.1) from the displayed list. ( $\rightarrow$  Pushbutton\_manual\_mode  $\rightarrow$  -S  $\rightarrow$  -S3)



→ Insert the other input tags "-S3", "-K0", "-B1", "-S4" and "-A1" and insert output tag "-Q1" (%Q101.0) at output "Conveyor\_motor\_manual\_mode".



#### 7.9 Saving and compiling 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 sicon for compiling in the menu.

 $(\rightarrow \blacksquare \texttt{Save project} \rightarrow \texttt{Program blocks} \rightarrow \blacksquare)$ 

VA	Siemens - C:W	00_TIA_Portal\034-1	00_FC-I	Pro	grar	nmi	ing\	034-10	00_FC	-Prog	ramm	ing	
Pro	Project Edit View Insert Online Options Tools Window Help												
Ż	i 📑 🔒 Save p	oroject 昌 🐰 值 🛙	ĩΧ	١ <b>c</b>	) ±	CI:	± [	a 🗉	lî		ł 💋	Go or	nline
	Project tree			•	03	4-1	00_	FC-Pro	ogran	nming	j) (	CPU_I	от
	Devices							Comp	one				
	- M			•	ιð	ы	1	¢ ≝¢	ii,			93	<b>:</b>
60						Ma	in						
Ē	034-100	FC-Programming		^	-		Nan	ne					Dat
am	📑 Add n	ew device			1		•	Temp					
150	📩 Device	es & networks			2			OB	1_EV_	CLASS			Byt
proj	👻 🔽 📷 CPU_I	OT2040 [IOT2000EDU]	]		3		•	OB	1_SCA	N_1			Byt
PLC	📑 De	vice configuration		≡	4	-	•	OB	1_PRIC	RITY			Byt
	😵 Or	nline & diagnostics			5	-	•	OB	1_OB_	NUMB	R		Byt
	💌 🛃 Pro	ogram blocks				_							
		Add new block			8	-	> = 1	??	-	-0	↦	-[=]	
		Main [OB1]			-	RIo	ck t	itle <sup>, ,</sup>	'Main	Progra	m Swe	een (C	vcle

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

nfo					
		🔍 Prope	rties 🚺	Info i 🗓	Diagnostics
General 追 Cross-reference	s Compile Syntax				
🕄 🛕 🚺 Show all messages	· · · · · · · · · · · · · · · · · · ·				
Compiling finished (errors: 0; warning	JS: 0)				
! Path	Description	Go to ?	Errors	Warnings	Time
🗸 🔻 СРU_ЮТ2040		7	0	0	11:12:28 AM
<ul> <li>Hardware configuration</li> </ul>		~ ~			11:12:29 AM
🗸 🔻 Program blocks		~ ~	0	0	11:12:34 AM
MOTOR_MANUAL (FC1)	Block was successfully compiled.	× 1			11:12:34 AM
	Block was successfully compiled.	×			11:12:37 AM
S Main (OB1)	brock thas successionly complicat				

#### 7.10 Downloading the program

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



(→ 🛄)

## 7.11 Monitoring program blocks

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

X 🖻 👻 🔍 🖿 🚍 🚍 💬 🕾			5 0.
	Block interface		onitoring on/off
>=1 [??] -I -ol			
_			
ock title: "Main Program Sweep (Cycl nment	e)"		
Network 1: Control conveyor motor	forwards in manual mode		
comment			
&	%FC "MOTOR_M		
%101.3 "-ко" —	— EN	ANUAL	
%101.2	Pushbutton_		
"-В1"	%101.1 manual_	Conveyor_	
%4100.4 "-54" — <b>○</b> ↔	"-S3" — mode Enable_OK	motor_ manual_ %Q101.0	
	%101.4 Safety_	mode "-Q1"	
	"-A1" - shutoff_active	ENO —	
Network 2:			
	111	> 100%	▼ <u> </u>
		· · · · ·	
	Block interface		
>=1 [??] ⊣ −0  ↦ -[=]	Block interface		
>=1 ⑦ ← ー이 → ┥=] pck title: "Main Program Sweep (Cycl ament			
ock title: "Main Program Sweep (Cycl	e)*		
ock title: *Main Program Sweep (Cycl Iment Network 1: Control conveyor motor	e)*		
ock title: *Main Program Sweep (Cycl Iment Network 1: Control conveyor motor Comment	e)*		
ock title: *Main Program Sweep (Cycl Iment Network 1: Control conveyor motor Comment	e)*		
Additional Sector Secto	e)"		
Ack title: *Main Program Sweep (Cycl Iment Network 1: Control conveyor motor Comment FALSE %101.3 *K0*	e)*		
Additional Sector Secto	e)* forwards in manual mode		
And the second s	e)* forwards in manual mode "MOTOR_M EN FALSE Pushbutton_		
And title: "Main Program Sweep (Cycl Iment Network 1: Control conveyor motor Comment FALSE %101.3 "K0" FALSE %101.2 "-B1" FALSE	e)" forwards in manual mode "MOTOR_M EN FALSE Pushbutton_ %101.1 manual_		
Ack title: *Main Program Sweep (Cycl Iment Network 1: Control conveyor motor Comment FALSE %101.3 *K0* FALSE %101.2 *-B1*	e)" forwards in manual mode "MOTOR_M EN FALSE Pushbutton_	IANUAL"	
Ack title: "Main Program Sweep (Cycl Iment Network 1: Control conveyor motor Comment FALSE %1101.3 "+K0" FALSE %1101.2 "-B1" FALSE %1100.4	e)" forwards in manual mode MOTOR_M FALSE FALSE Pushbutton_ %101.1 "-53"	Conveyor_ motor_ manual_%Q101.0	
Ack title: "Main Program Sweep (Cycl Iment Network 1: Control conveyor motor Comment FALSE %1101.3 "+K0" FALSE %1101.2 "-B1" FALSE %1100.4	e)" forwards in manual mode FALSE %101.1 "-53" Motor_ Manual_ "-53" Motor_ Manual_ "-53" Motor_ Enable_OK FALSE %101.4 Safety	Conveyor_ motor_ manual_ mode "-Q1"	
Ack title: "Main Program Sweep (Cycl Iment Network 1: Control conveyor motor Comment FALSE %101.3 "-K0" FALSE %1101.2 "-B1" FALSE %1100.4	e)" forwards in manual mode MOTOR_M FALSE FALSE Pushbutton_ %101.1 "-53"	Conveyor_ motor_ manual_%Q101.0	

**Note:** 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:** 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  $\boxed{1}$  icon. ( $\rightarrow$   $\boxed{1}$   $\rightarrow$  Call environment  $\rightarrow$  OK)

Call	envi	ironment of block		×
0	lone	:		
	nsta	nce data block		
				▼
0	Call e	environment		
		Dependency structure	! Addre	
	1	= Main	OB1	@Main ► NW1 (Control conveyor motor forwards in manual mode)
	2			
				Transfer to "adjusted manually"
0	Manu	ually adjusted call environment		
0				
				OK

#### 7.12 Archiving the project

→ As the final step, we want to archive the complete project. Click on the command → "Archive ..." 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". (→ Project → "Archive → TIA Portal project archive → 034-100\_FC Programming. → Save)

100_TIA_Portal\034-100_FC-Pr	ogramming\034-100_FC-Programming	□×
Project Edit View Insert Online Option	s Tools Window Help Totally Integrated Automation	
1 New	) ± (# ± 🖥 🗓 🗓 🖳 💋 Go online 🖉 Go offline 🕌 🌆 🕼 🕞 👫 🚽 📋 🔭 PORTAL	L
Open Ctrl+O Migrate project	100_FC-Programming > CPU_IOT2040 [IOT2000EDU] > Program blocks > MOTOR_MANUAL [FC1] 🛛 🗕 🖬 🖬 🗙	
Close Ctrl+W		
Save Ctrl+S	🚜 🐼 과 관 🔍 돈 🗖 🚍 💬 웹 ± 웹 ± 웹 발 드 🔯 🥙 🚱 웹 🧐 🖓 🗭 🖕 노 노 및 이 신 & 약 🖦 🛛 📮	<b>S</b>
Save as Ctrl+Shift+S	Block interface	Ē
Delete project Ctrl+E		Instructions
Archive Retrieve	2 >=1 [22] → −οι → −[=]	12
	Block title: Motor control in manual mode	
Multiuser	Network 1: Control of the conveyor motor in manual mode	
Card Reader/USB memory     Memory card file	Comment	P Testing
Start basic integrity check		- Bu
Print Ctrl+P	8	
Print preview Ctri+P	#Pushbutton#Conveyor manual_modemotor_manual	谢 Tasks
F:\00_TIA_Portal\\034-100_FC-Programming	#Enable_OK — mode	lsks
F:\00_TIA_Portal\014-101\014-101_IOT2040	#Safety_shutoff	
C:\Us\032_100_FC-Programming_V14_V15	active 🗕 🦛 🔤	🗄 Libraries
C:\Users\schulun\20180618_Osterhofen_1		E I
C:\Users\schulun\20180620_Osterhofen_3 F:\00_TIA_Portal\014_10\014_101_IOT2040		ari
C:\Users\sch\20180619_Osterhofen_Force		es
C:\Users\schul\20180618_Lüftersteuerung		
Exit Alt+F4	100%	
Module	🖳 Properties 🚺 Info 🔛 Diagnostics 💷 🖃 🤜	
	General Cross-references Compile Syntax	
Name	Show all messages	1
Device configuration		
S Online & diagnostics	! Message Go to ? Date Ti	
Program blocks	V Loading completed (errors: 0; warnings: 0).	
Technology objects	Connected to CPU_IOT2040, via address IP=192.168.0.1. 6/21/2018 11	
External source files	Connection to CPU_IOT2040 terminated. 6/21/2018 11	
PLC data types	The project 034-100_FC-Programming was saved successfully.	
Watch and force tables	K	
Portal view     Dverview	■ MOTOR_MA ■ MOTOR_MA ■ Of the project 034-100_FC-Programming	

#### 7.13 Checklist

No.	Description	Checked
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 Activate conveyor manual mode forwards (-S3 = 1) Conveyor motor forwards fixed speed (-Q1 = 1)	
4	Same as 3 but activate EMERGENCY STOP (-A1 = 0) $\rightarrow$ -Q1 = 0	
5	Same as 3 but switch off station (-K0 = 0) $\rightarrow$ -Q1 = 0	
6	Same as 3 but cylinder not retracted (-B1 = 0) $\rightarrow$ -Q1 = 0	
7	Same as 3 but also activate conveyor manual mode backwards (-S4 = 1) $\rightarrow$ -Q1 = 0	
8	Project successfully archived	

## 8 Exercise

#### 8.1 Task – Exercise

In this exercise, the following function of the process description sorting station is also to be planned, programmed and tested:

- Manual mode - Control conveyor motor forwards in manual mode

#### 8.2 Technology diagram

Here, you see the technology diagram for the task.



Figure 10: Technology diagram



Figure 11: Control panel

### 8.3 Reference list

The following signals are required as operands for this task.

DI	Туре	Identifier	Function	NC/NO
I 101.4	BOOL	-A1	Return signal emergency stop OK	NC
I 101.3	BOOL	-K0	Station "ON"	NO
I 101.2	BOOL	-B1	Sensor cylinder -M4 retract	NO
I 101.1	BOOL	-S3	Pushbutton manual mode conveyor -M1 forwards	NO
l 100.4	BOOL	-S4	Pushbutton manual mode conveyor -M1 backwards	NO

DQ	Туре	Identifier	Function	
Q 101.0	BOOL	-Q1	Conveyor motor -M1 forwards fixed speed	
Q 100.7	BOOL	-Q2	Conveyor motor -M1 backwards fixed speed	

Output

#### Legend for reference list

DI	Digital input	DQ	Digital output	

- Al Analog input AQ Analog output
  - Input Q
- NC Normally Closed
- NO Normally Open

#### 8.4 Planning

L

Plan the implementation of the task on your own.

#### 8.5 Checklist – Exercise

No.	Description	Checked
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 Activate conveyor manual mode backwards (-S4 = 1) Conveyor motor backwards fixed speed (-Q2 = 1)	
4	Same as 3, but activate EMERGENCY OFF (-A1 = 0) $\rightarrow$ -Q2 = 0	
5	Same as 3, but switch off station (-K0 = 0) $\rightarrow$ -Q2 = 0	
6	Same as 3, but cylinder not retracted (-B1 = 0) $\rightarrow$ -Q2 = 0	
7	Same as 3, but also activate conveyor manual mode forwards $(-S3 = 1) \rightarrow -Q1 = 0$ and $-Q2 = 0$	
8	Project successfully archived	

## 9 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/iot2000

#### **Further information**

Siemens Automation Cooperates with Education siemens.com/sce

SCE Training Curriculums siemens.com/sce/module

SIMATIC IOT2000 siemens.com/sce/IOT2000

SCE Trainer Packages siemens.com/sce/tp

IOT2000 Forum siemens.com/iot2000-forum

SCE Contact Partners siemens.com/sce/contact

Digital Enterprise siemens.com/digital-enterprise

Industry 4.0 siemens.com/future-of-manufacturing

Totally Integrated Automation (TIA) siemens.com/tia

TIA Portal siemens.com/tia-portal

SIMATIC Controller siemens.com/controller

SIMATIC Technical Documentation siemens.com/simatic-docu

Industry Online Support support.industry.siemens.com

Product catalog and online ordering system Industry Mall **mall.industry.siemens.com** 

Siemens AG Digital Factory P.O. Box 4848 90026 Nuremberg Germany

Subject to change and errors © Siemens AG 2018

siemens.com/sce