

6

Contents

6.	Binary Operations 1	6-2
6.1.	Plant Description: The Conveyor Model as a Distribution Conveyor	6-3
6.2.	Assignment	6-4
6.2.1.	Sensors and Check Symbols	6-5
6.2.2.	Binary Logic Operations: AND, OR and Negation	6-6
6.2.3.	Theory Exercise 1: Sensor and Check Symbols	6-7
6.2.4.	Binary Logic Operations: Exclusive OR (XOR)	6-8
6.3.	Process Images	6-9
6.4.	Cyclic Program Execution	6-10
6.5.	Linear Program	6-11
6.6.	Programming	6-12
6.6.1.	Networks	6-13
6.6.2.	Absolute and Symbolic Addressing	6-14
6.6.3.	Using a PLC Tag as an Operand	6-15
6.6.4.	Renaming / Rewiring PLC Tags	6-16
6.6.5.	Defining (Declaring) Tags while Programming	6-17
6.6.6.	Closing / Saving / Rejecting a Block	6-18
6.6.7.	Compiling a Program	6-19
6.6.8.	Downloading a Program into the CPU	6-20
6.6.9.	Monitoring a Block	6-21
6.7.	Exercise 1: Renaming Main OB	6-22
6.7.1.	Exercise 2: Programming the Jog Mode	6-23
6.7.2.	Exercise 3: Compiling the Program, Downloading it into the CPU	6-24
6.7.3.	Exercise 4: Monitoring the Program	6-25

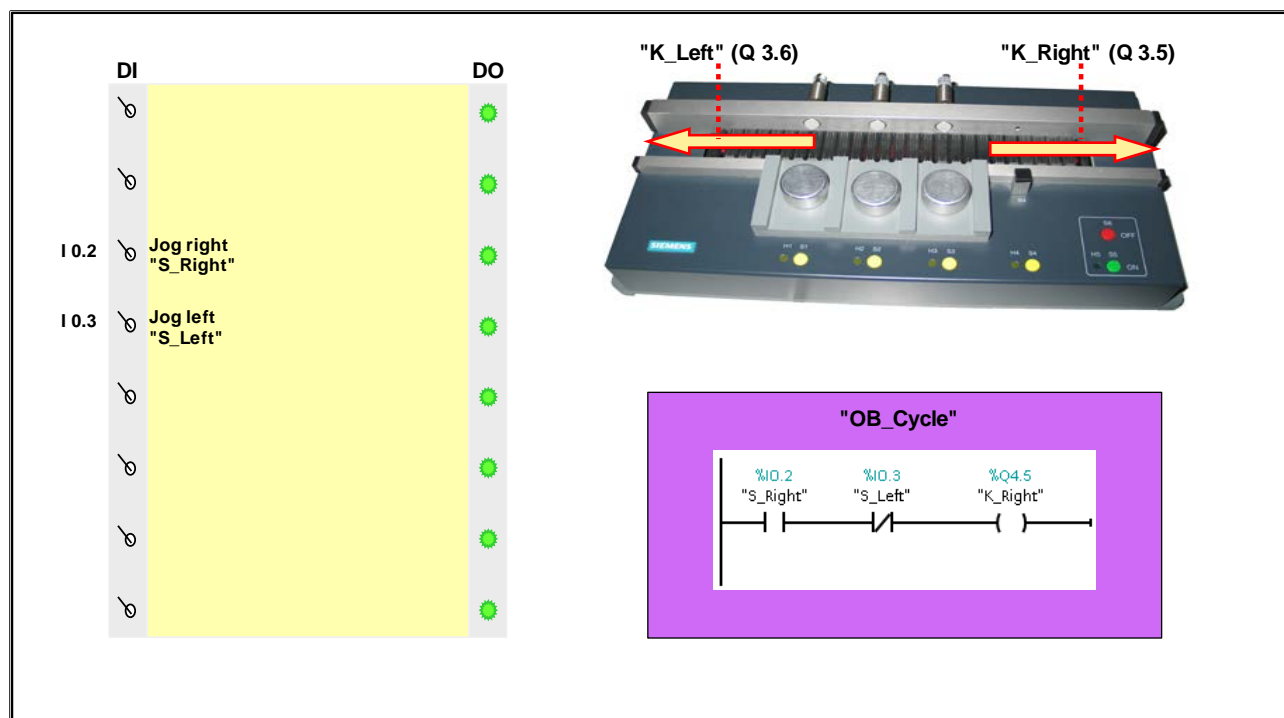
6. Binary Operations 1

At the end of the chapter the participant will ...



- ... know what an assignment is
- ... understand the difference between 'real' NC contacts and NO contacts connected in the hardware, and programmed check symbols
- ... be familiar with the logic operations AND, OR and Exclusive-OR and be able to apply them
- ... be familiar with the process image for inputs and outputs
- ... understand cyclic program execution
- ... know what a linear program is
- ... be familiar with the program editor
- ... be able to rename and rewire PLC tags
- ... be able to carry out a program test with the "Monitor (block)" function

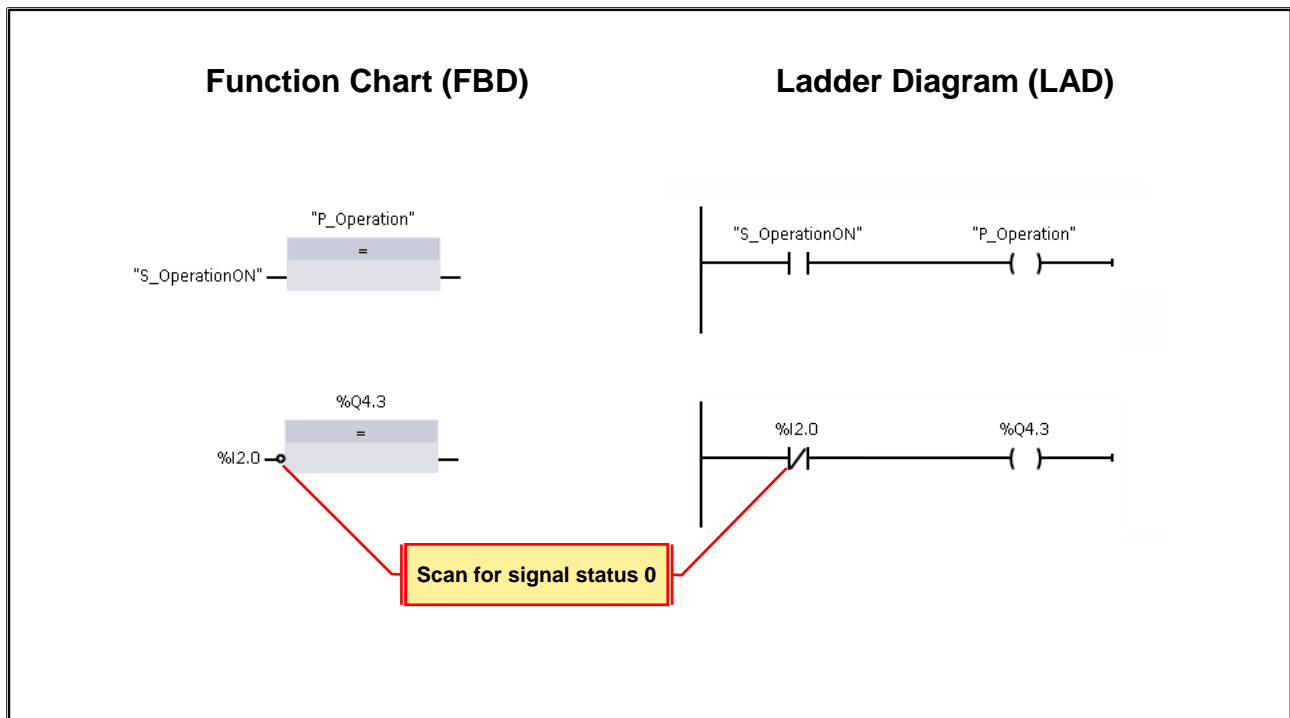
6.1. Plant Description: The Conveyor Model as a Distribution Conveyor



Conveyor Model as a Distribution Conveyor

The distribution conveyor can be jogged to the left and to the right using the switches "S_Right" (I 0.2) and "S_Left" (I 0.3). If both switches are activated simultaneously, then the conveyor must not move (Lock-out!).

6.2. Assignment













Assignment

With an assignment, the specified operand is always assigned the current RLO as status. The assigned RLO remains available after the assignment and can be assigned to a further operand or it can be further logically linked.

Scan for 0 (False)

The scan for 0, that is, the signal status FALSE is fulfilled when the operand has the signal status '0'.

6.2.1. Sensors and Check Symbols

Process			Interpretation in the PLC Program				
The sensor is a ...	The sensor is ...	Voltage present at input?	Signal status at input	Scan for signal status "1"	Scan for signal status "0"	Symbol / Instruction	Result of check
NO contact 	activated 	yes	1	LAD:  "NO contact"	"Yes" 1	LAD:  "NC contact"	"No" 0
	not activated 	no	0		"No" 0		"Yes" 1
NC contact 	activated 	no	0	FBD: 	"No" 0	FBD: 	"Yes" 1
	not activated 	yes	1		"Yes" 1		"No" 0

Sensors of the Process

The use of normally open or normally closed contacts for the sensors in a controlled process depends on the safety regulations for that process. Normally closed contacts are always used for limit switches and safety switches, so that dangerous conditions do not arise if a wire break occurs in the sensor circuit. Normally closed contacts are also used for switching off machinery for the same reason.

All scans can be programmed for signal status, that is, Status '0' and '1' regardless of whether a hardware NO contact or hardware NC contact is connected in the process.

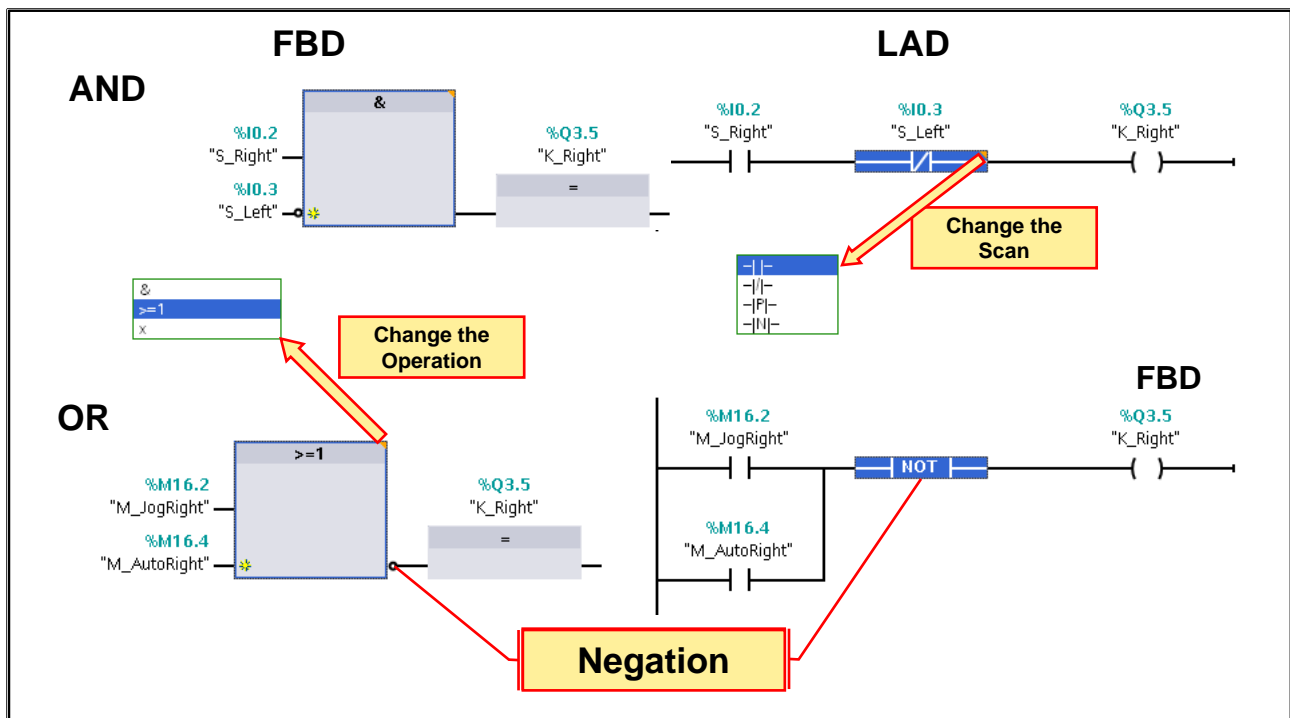
Check Symbols of the Program

In LAD, a symbol with the name "NO contact" is used for checking for signal status "1" and a symbol with the name "NC contact" to check for signal status "0". It makes no difference whether the process signal "1" is supplied by an activated NO contact or a non-activated NC contact.

The "NO contact" symbol delivers the result of check "1" when the scanned operand has Status "1".

The "NC contact" symbol delivers the result of check "1" when the scanned operand has Status "0".

6.2.2. Binary Logic Operations: AND, OR and Negation



AND and OR Logic Operations

With the AND and OR logic operations, basically all binary operands can be scanned, even outputs. Instead of individual operands, the results of other logic operations can also be further logically linked. Also, the logic operations can also be combined.

AND Logic Operation

For an AND logic operation, the result of logic operation (RLO) = '1', when **all** input signals have Status '1'.

OR Logic Operation

For an OR logic operation, the result of logic operation (RLO) = '1', when **at least one** input signal has Status '1'.

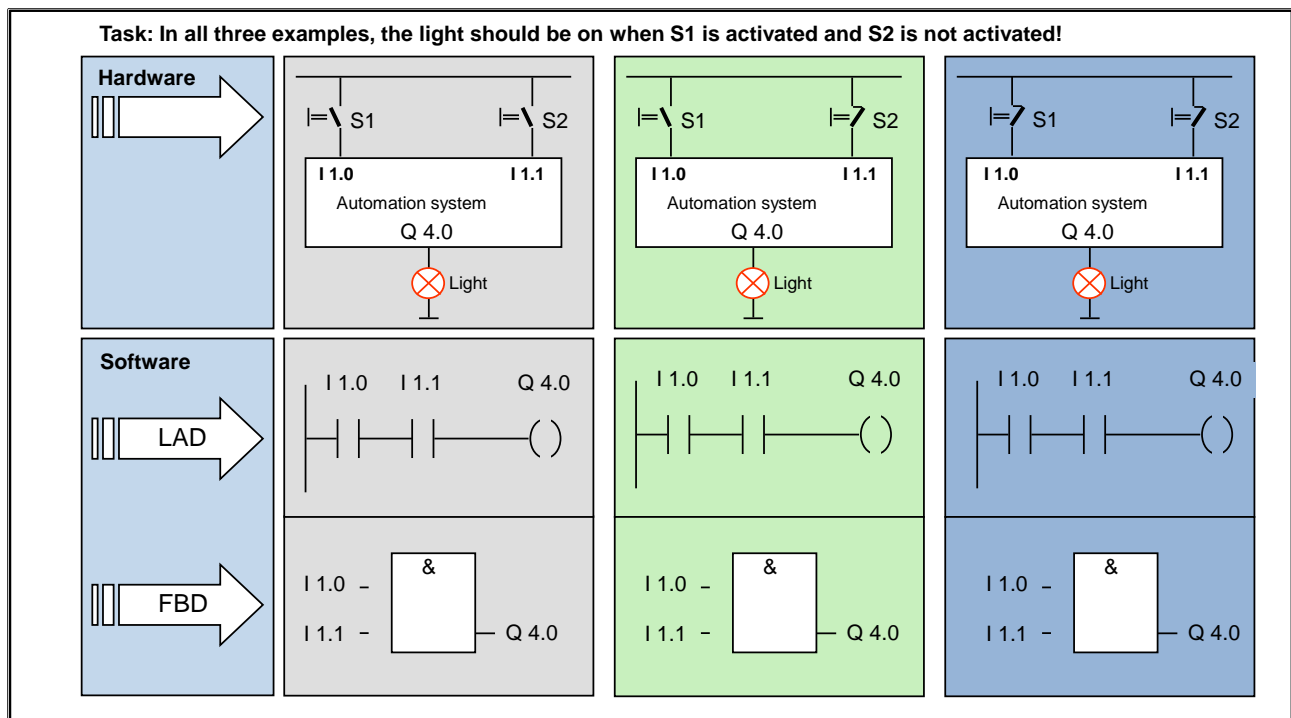
Negation (NOT)

The NOT instruction inverts the result of logic operation (RLO).

If, in the example shown, the RLO of the OR logic operation = '1', the NOT instruction inverts it to RLO '0' and the output is assigned the Status '0'.

If the RLO of the OR logic operation = '0', the NOT instruction inverts it to RLO '1' and the output is assigned the Status '1'.

6.2.3. Theory Exercise 1: Sensor and Check Symbols



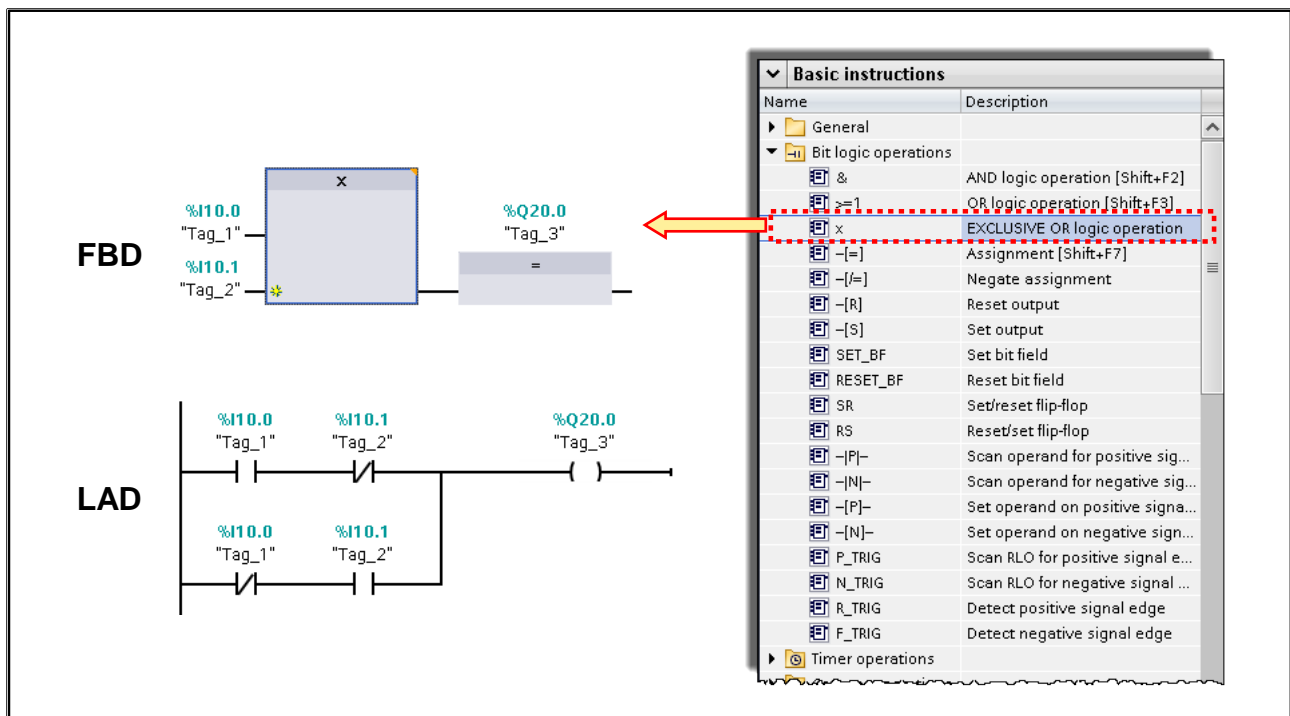
Task

In FBD, complete the logic symbols, and, in LAD correct the check symbols so that the required function is fulfilled.

Note

The terms - "NO contact" and "NC contact" - have different meanings depending on whether they are used in the process-hardware-context or as check symbols in the software.

6.2.4. Binary Logic Operations: Exclusive OR (XOR)



XOR Logic Operation

With the XOR logic operation, basically all binary operands can be scanned, even outputs. Instead of individual operands, the results of other logic operations can also be further logically linked. Also, the logic operations can also be combined.

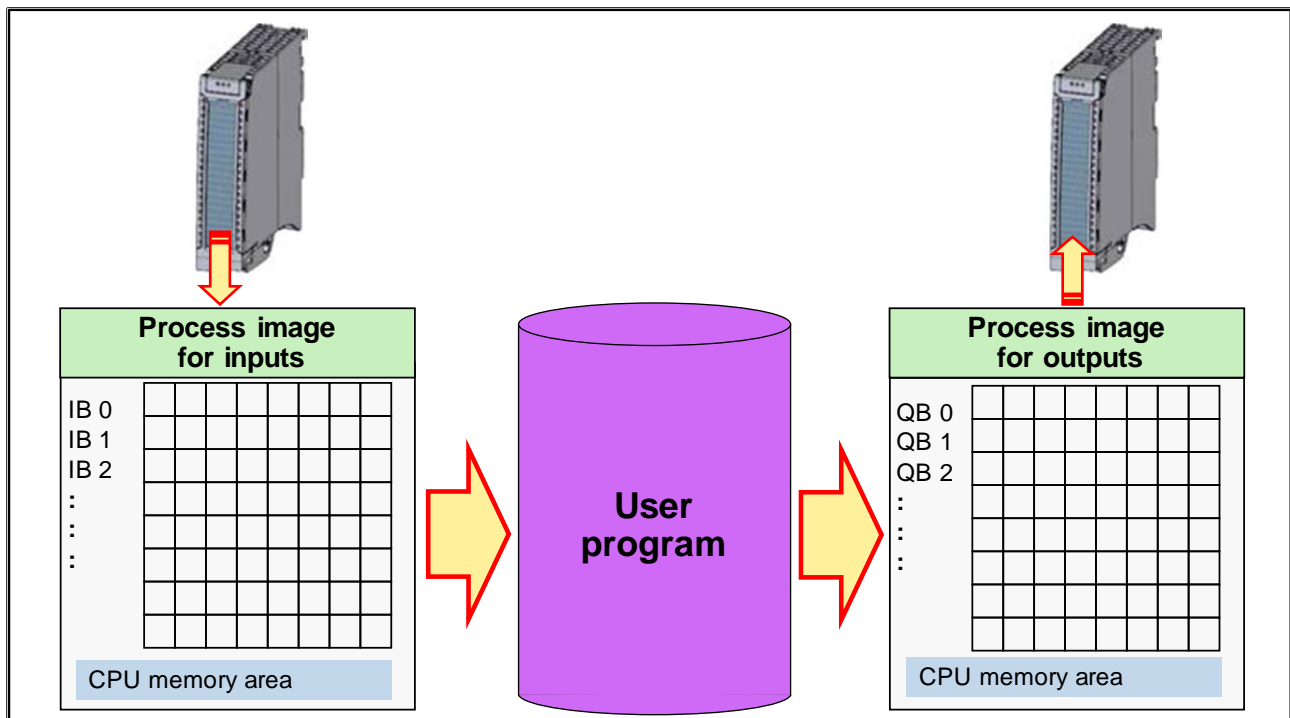
All inputs of the logic operations can be programmed as scan for signal status or Status '0' and '1', regardless of whether a hardware NO contact or NC contact is connected in the process.

- For an XOR logic operation with 2 inputs, the result of logic operation (RLO) = '1', when one and only one of the two input signals has Status '1'.
- For an XOR logic operation with more than 2 operands, the RLO ... = '1', when an uneven number of checked operands has Status '1' = '0', when an even number of checked operands has Status '1'.

XOR in the LAD Programming Language

In the LAD programming language, there is no explicit XOR logic operation. It must be generated by programming the discrete instructions shown in the picture above.

6.3. Process Images



Process Images

For the storage of **all** digital input and output states, the CPU has reserved memory areas: the process image for inputs (PII) and the process image for outputs (PIQ). During program execution, the CPU accesses these memory areas exclusively. It does not access the digital input and output modules directly.

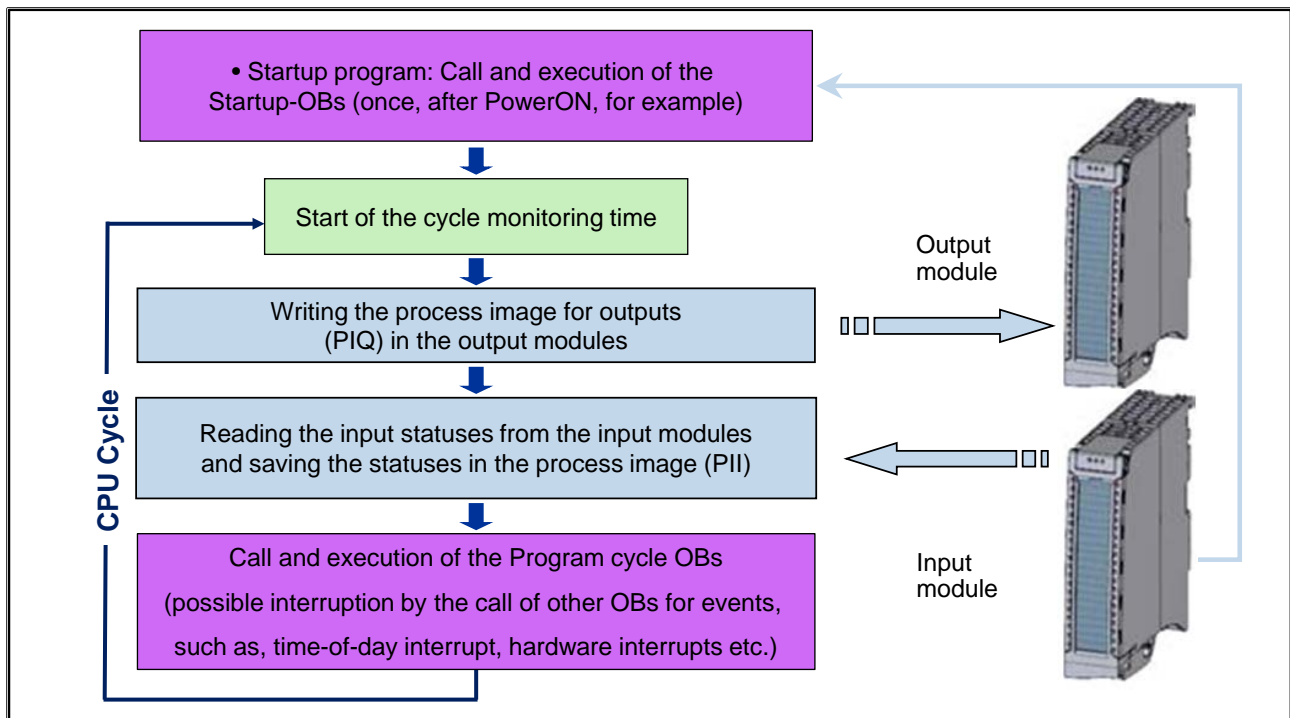
PII

The Process Image for Inputs (PII) is the memory area in which the statuses of all digital inputs are stored. At the beginning of the cycle, the digital input modules read-in to the PII. When an input is linked, the status of this input stored in the PII is linked. This status cannot change within a cycle since the PII is only updated or read-in at the beginning of a cycle. This guarantees that when there are multiple scans of the input in one cycle, the same result is always supplied.

PIQ

The Process Image for Outputs (PIQ) is the memory area in which the statuses of all digital outputs are stored. The PIQ is output to the digital output modules at the beginning of the cycle before the process image for inputs is updated. Outputs can be assigned as well as scanned in the program.

6.4. Cyclic Program Execution



Restart

When you switch on or switch from STOP --> RUN, the CPU carries out a complete restart whereby the programmed Startup-OBs are executed. During restart, the operating system deletes all non-retentive memory bits and, after the Startup-OBs have been executed, it starts the cycle monitoring time. **(The Startup-OBs are dealt with in the chapter "Organization Blocks")**

Cyclic Program Execution

Cyclic program execution occurs in an endless loop. After the execution of a program cycle is completed, the execution of the next cycle occurs automatically. In every program cycle, the CPU carries out the following steps.

- Transfer the output statuses from the process image for outputs to the output modules.
- Scan the statuses of the input signals and update the process image for inputs.
- Execute the instructions of the user program. This happens mainly with the process images, not with the input and output modules directly.

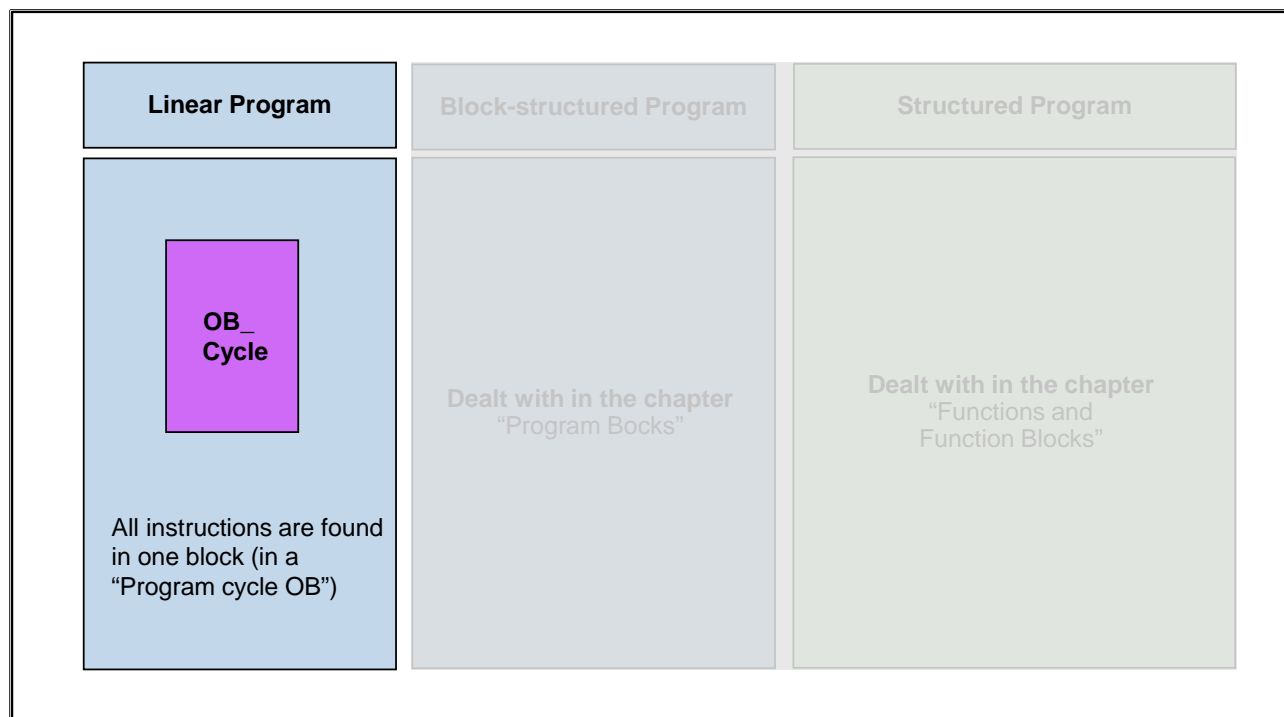
Cycle Time and Cycle Monitoring Time

The time that the CPU requires for the execution of the complete program cycle, is the cycle time which is monitored for time by the CPU operating system. If the cycle time exceeds the cycle monitoring time defined in the CPU properties (**Chapter Devices and Networks**), the "Time-Error-Interrupt-OB" is called. If this is not configured or if the cycle monitoring time is double, the CPU goes into the STOP state.

Using the Process Image or deselecting it for Individual Modules

In the properties of the modules you can select whether the input values are to be automatically adopted in the PII or whether the output values are to be automatically written with values from the PIQ (at the beginning of the cycle), only if a certain OB is executed or never.

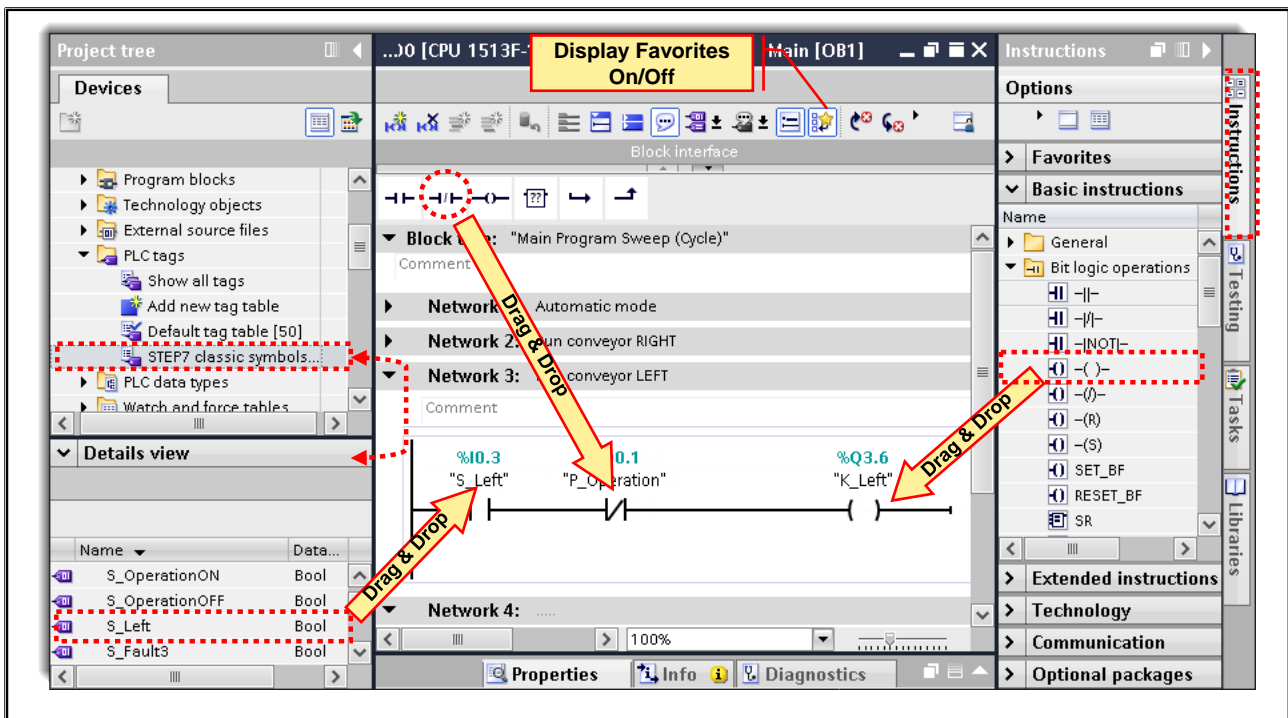
6.5. Linear Program



Linear Program

The entire program is found in one continuous program block (Program cycle OB) which is automatically called by the system. This model resembles a hard-wired relay control that was replaced by an automation system (programmable logic controller). The CPU processes the individual instructions one after the other.

6.6. Programming



Open Block

In the Program blocks folder there is a Program-Cycle-OB which can be opened by double-clicking in the instruction section.

Programming

The instructions within a block can be programmed as follows:

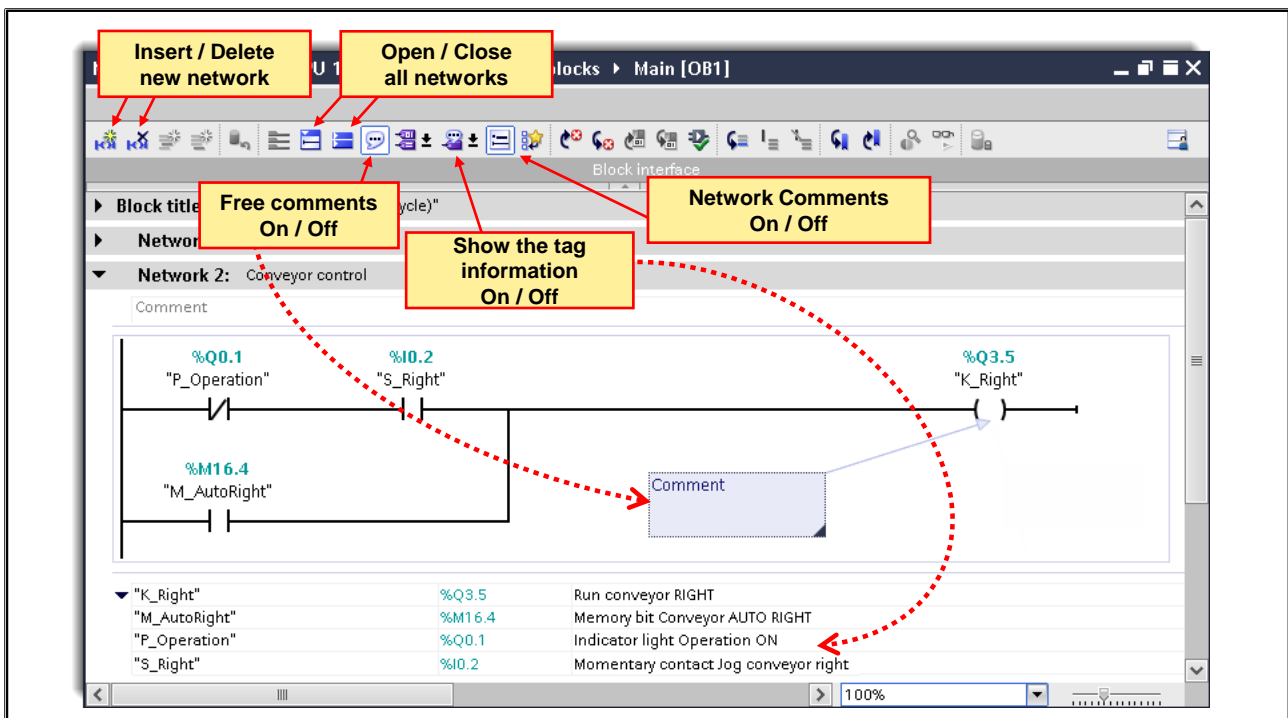
- using drag & drop from the Favorites or the Instructions catalog to anywhere in the program
- by first selecting the location in the program and then double-clicking on the desired instruction in the Favorites or the Instructions catalog
- by selecting (highlighting) the location in the program and the relevant shortcut (for information on the possible shortcuts see Options > Settings > Keyboard shortcuts)

Operands can be entered with an absolute or a symbolic address. If the tag table is highlighted (not opened!) in the Project tree, tags (variables) can also be pulled from the Details view to the appropriate location in the program using drag & drop.

Favorites

Frequently used elements, functions and instructions are available in the Favorites which can be expanded individually from the Instructions catalog using drag & drop.

6.6.1. Networks

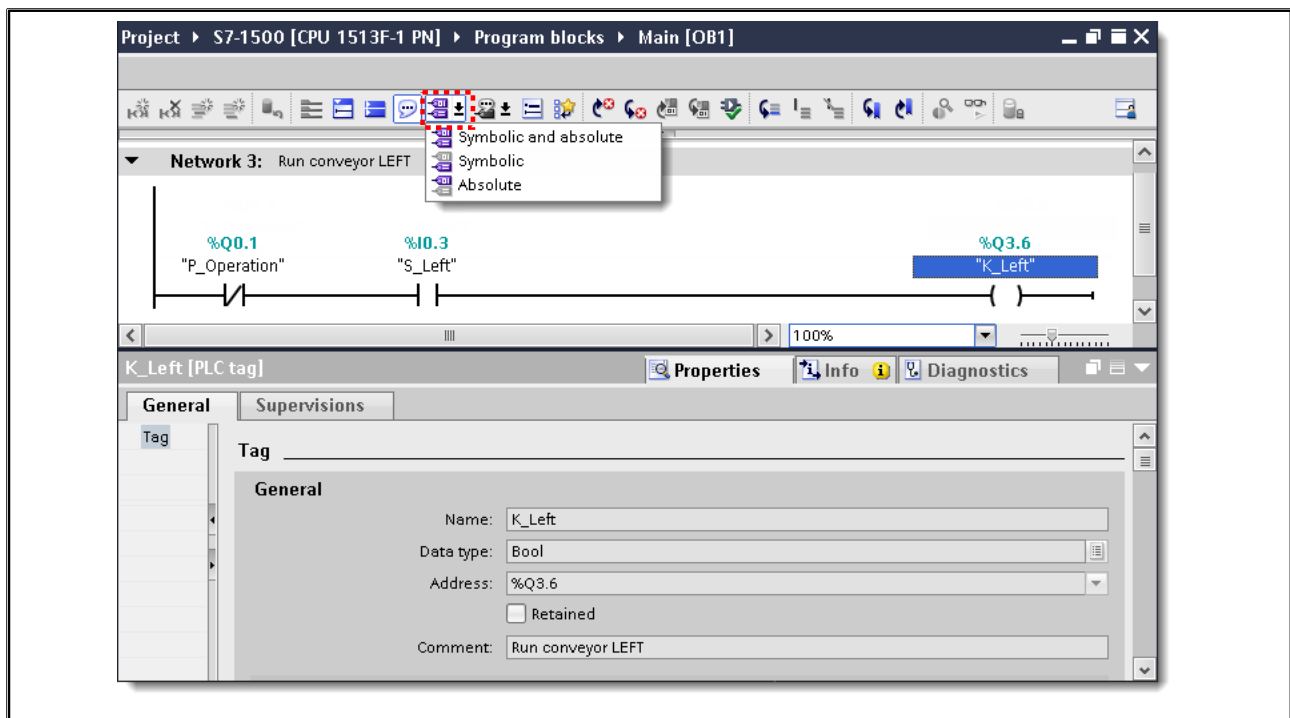


Networks

A block can be divided into individual networks which makes it easier to follow and gives you a better understanding of the program.

Each network can be given a network label and a comment. Within the networks, the individual instructions can be clarified with instruction comments and the comments of the tags/variables used can be displayed via the "Tag information" button.

6.6.2. Absolute and Symbolic Addressing



Absolute and Symbolic Addressing

All global tags/variables (such as, inputs, outputs, memory bits) have both an absolute and a symbolic address. You can define which is to be displayed or with which is to be programmed (see picture).

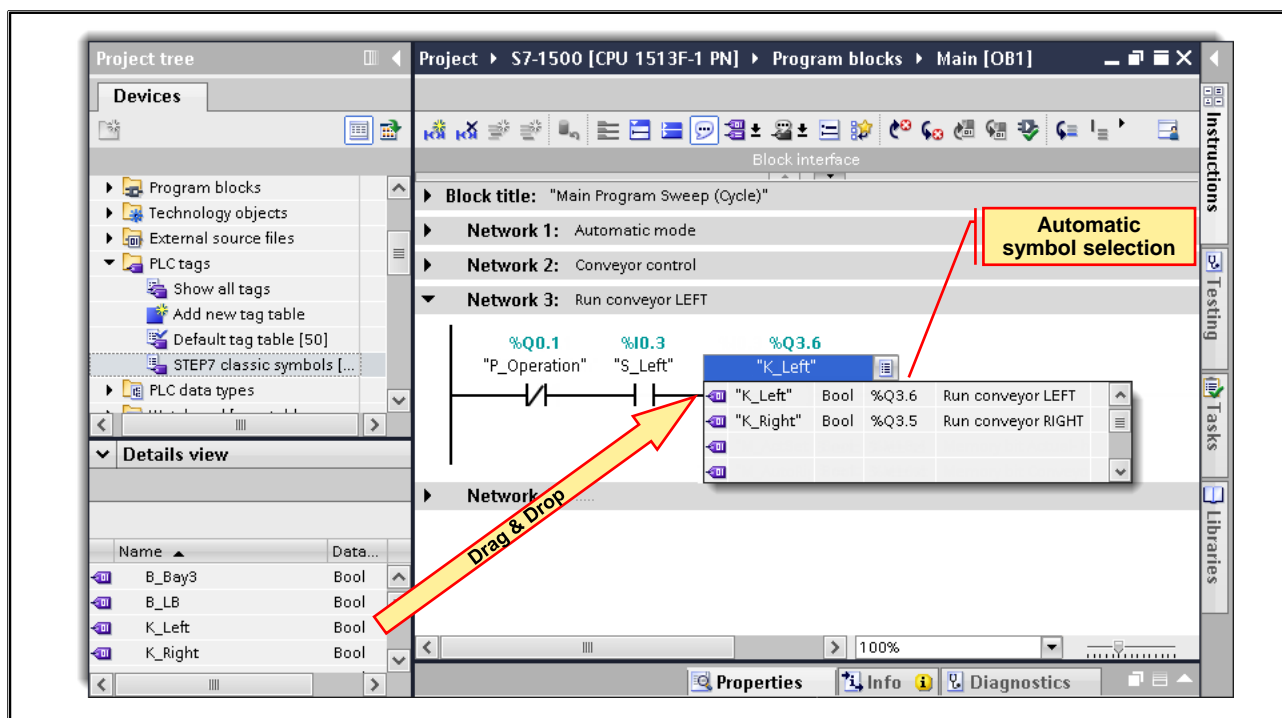
When you use a symbolic address (for example, "K_Left") which has not yet been assigned an absolute address, you can save the block but you cannot compile it and download it into the controller.

When you use an absolute address (for example, M16.2), it is automatically assigned a symbolic default address (for example, "Tag_1") which you can change (later on).

Properties

If a block or the PLC tag table is open in the working area and a tag is selected (highlighted) there, then all details are displayed in the "Properties" tab in the Inspector window.

6.6.3. Using a PLC Tag as an Operand



Using a Tag as Operand

During programming, the name of the tag or the address can be entered. When the symbolic name or the address is input, the Autocompletion automatically appears from which you can select the tag. Furthermore, you can use the Details view to adopt the tag using drag & drop.

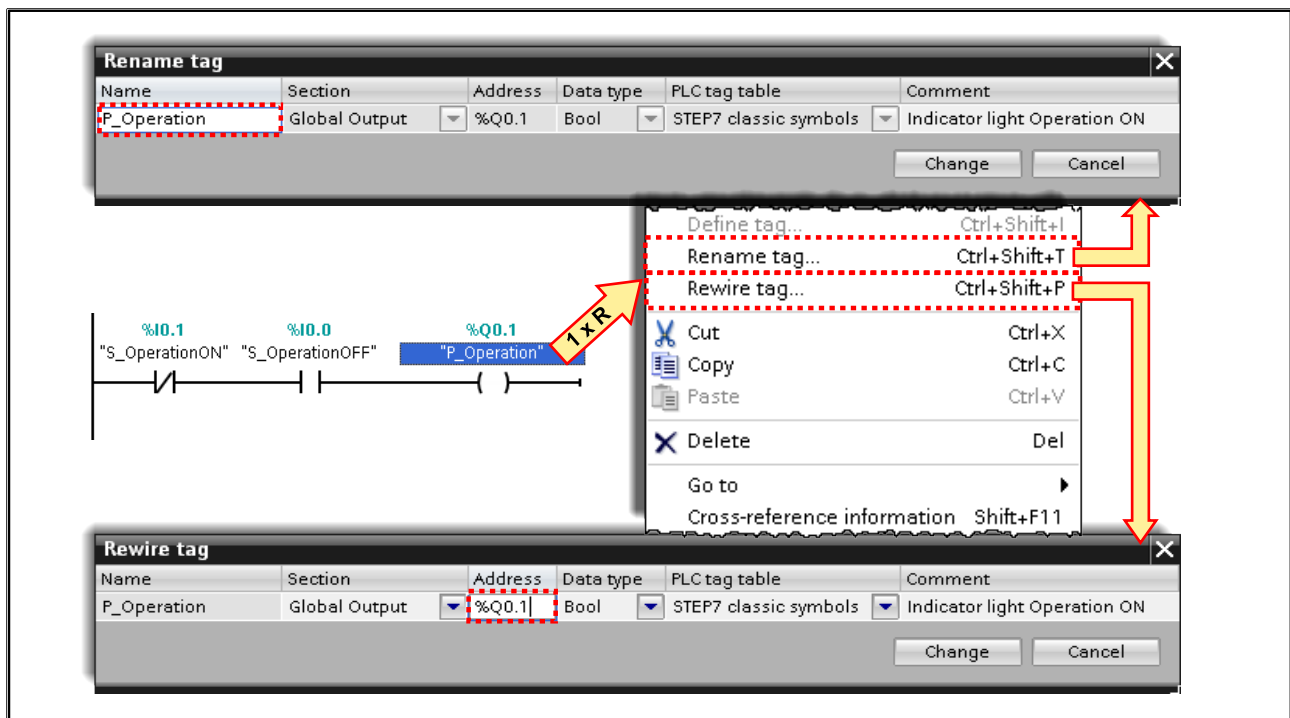
Autocompletion

When operands are selected, after the first letter of the symbolic operand name has been entered, a selection of all the operands whose name start with the entered letter and are of the corresponding data type is displayed. All the operands that are valid for this block are displayed. These are all global tags (also those that are declared in data blocks), local variables (temporary and static) as well as the parameters of the block.

You can also filter directly according to the type of tag, as required:

- Begin with # to only select from local tags of the current block interface,
- Begin with " to only have global tags displayed,
- Begin with % to have all tags filtered according to absolute addresses displayed,

6.6.4. Renaming / Rewiring PLC Tags



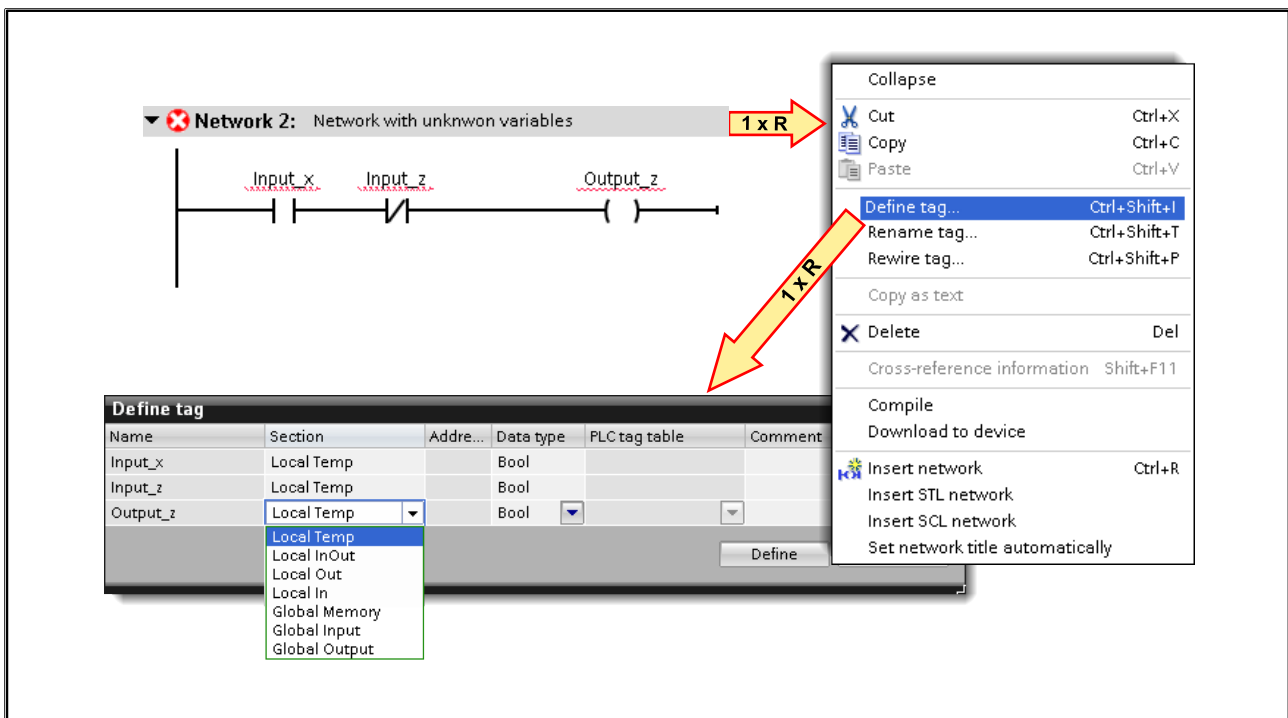
Renaming and Rewiring Tags

Tags can be renamed or rewired as shown in the picture using the Blocks Editor. The changes are immediately adopted in the PLC tag table and affect the entire program.

Tags can also be renamed and rewired directly in the PLC tag table.

- **Rename:**
Change the tag name, while the absolute address remains unchanged.
- **Rewire:**
Change the associated absolute address, while the name remains unchanged.

6.6.5. Defining (Declaring) Tags while Programming



Defining (Declaring) Tags while Programming

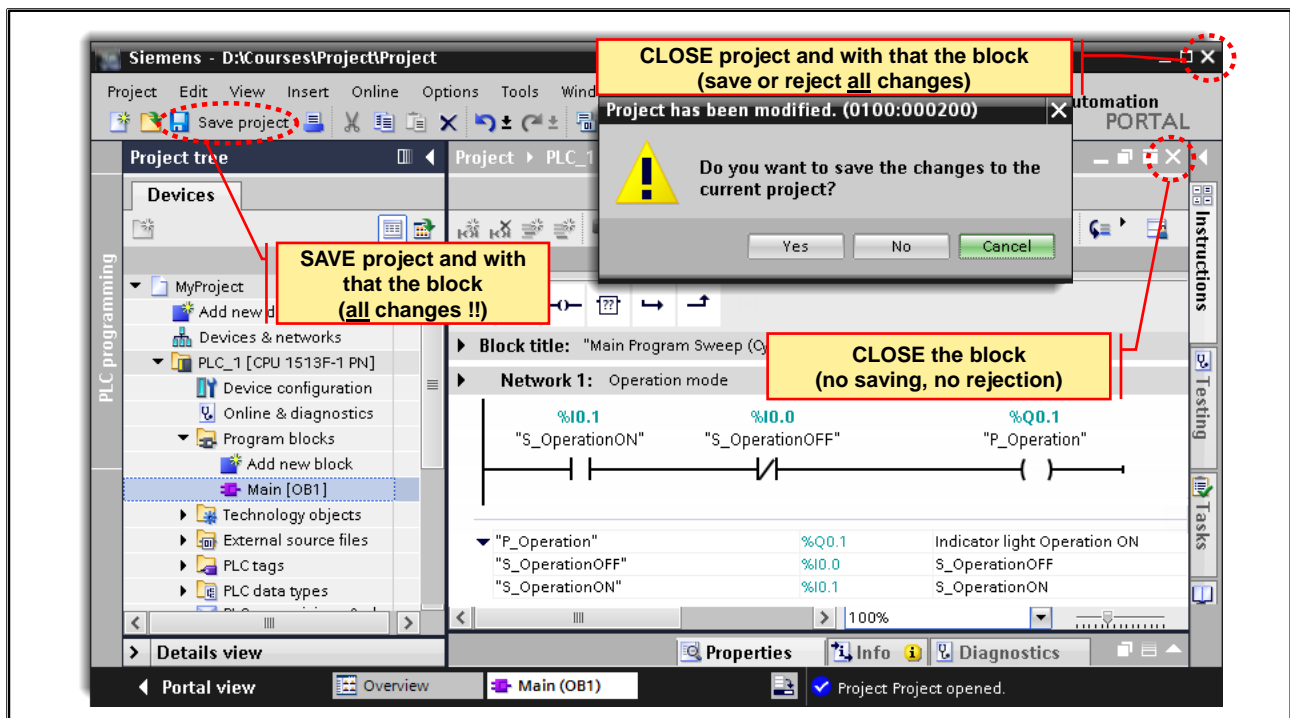
If unknown tags are used during programming, they can be defined later-on via the context menu of the network.

In the "Define tag" dialog, the [Local Temp] temporary memory area (Section) is always suggested. When you change the area, the next free address is suggested.


Advantage:

In the dialog that appears, only addresses which have not been used so far are suggested. In this way errors are avoided, for example, such as the use of bits which belong to an already used word (overlapping accesses).

6.6.6. Closing / Saving / Rejecting a Block



Closing a Block

By clicking on the  symbol in the title bar, the block is merely closed. Changes are neither rejected nor are they saved on the hard drive!

Saving a Block

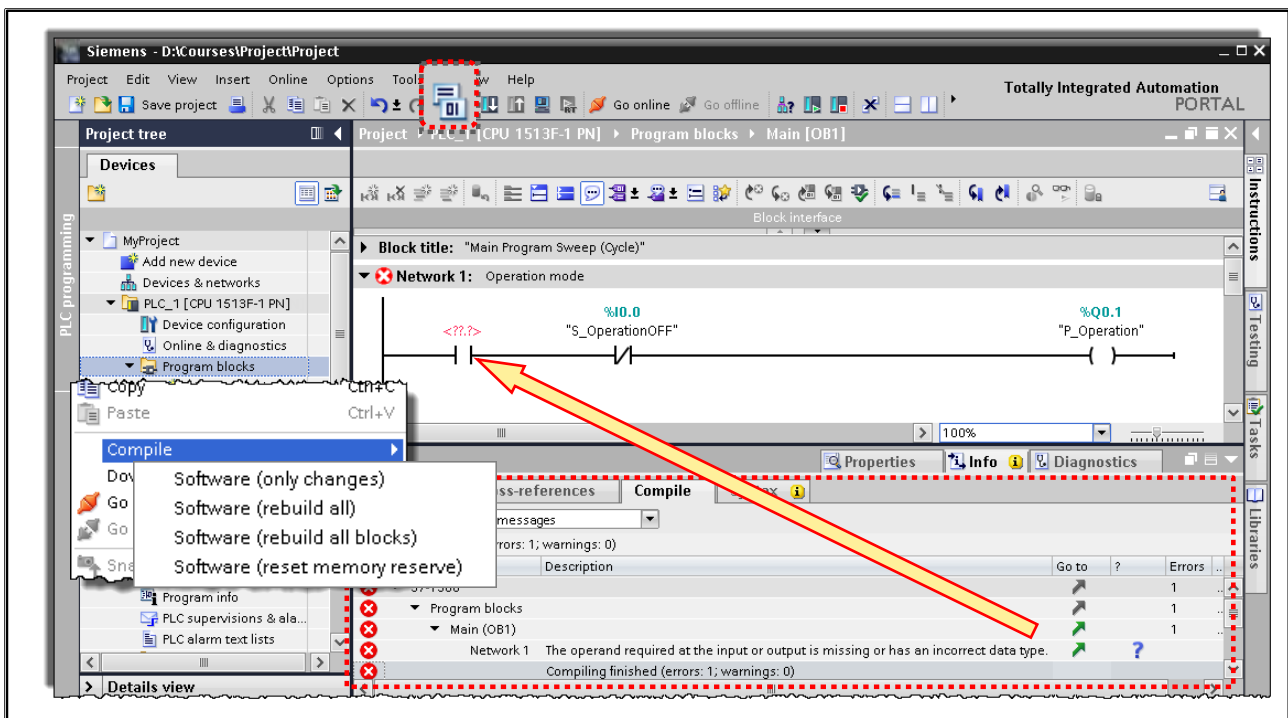


By using "Save project" the entire project, and with that also the block, is saved on the hard drive. All changes made to the project are saved.

Rejecting a Block

It is possible to reject block changes using the function Undo or by closing the entire project without saving. All changes made in the project are rejected.

6.6.7. Compiling a Program



Program Compilation

A delta compilation is always carried out via the "Compile" button, that is, only the changes to the object selected (highlighted) in the Project tree are compiled. By selecting the Program blocks folder, all modified objects (blocks) within the folder are compiled.

Via the context menu in the Project tree, you can select whether only changes or the entire software is to be compiled.

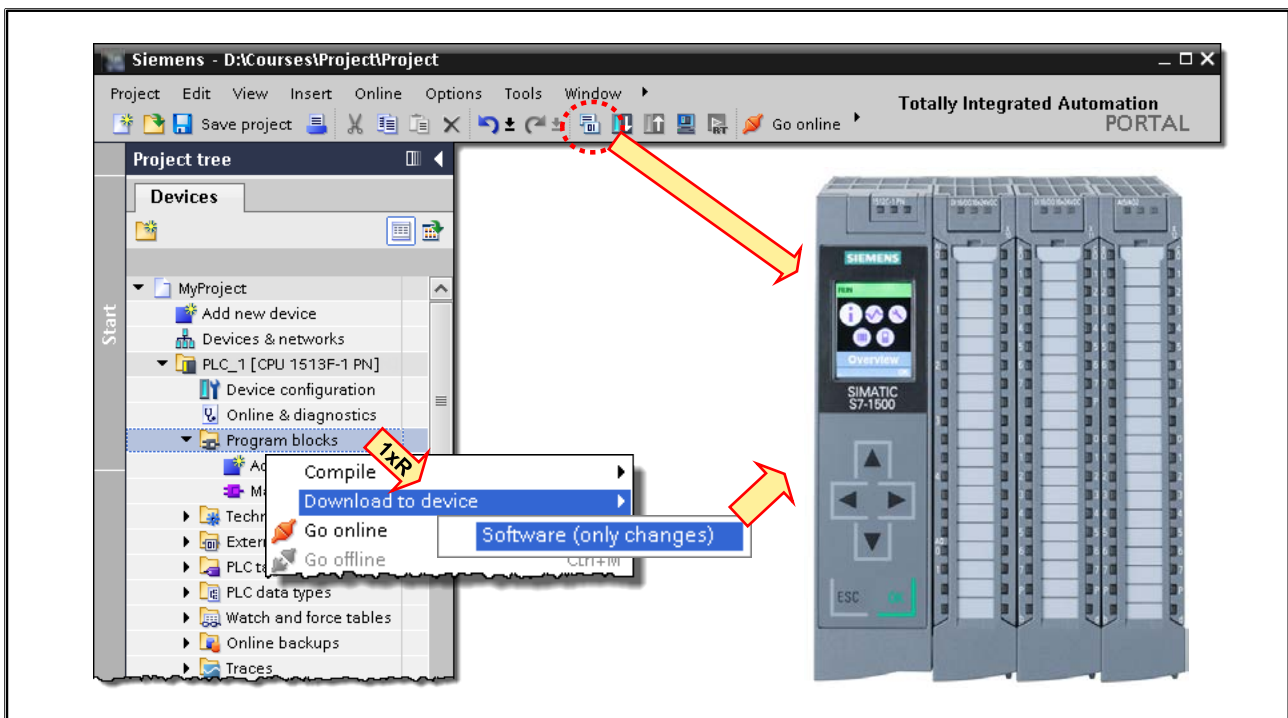
The status of the compilation is displayed in the Inspector window "Info -> Compile". If errors occurred during compilation, you can jump directly from the error entry to the error location by double-clicking on it.

Note:

The menu items Software (rebuild all) and Software (rebuild all blocks) currently have the same meaning.

Software (reset memory reserve) is not yet relevant in this chapter.

6.6.8. Downloading a Program into the CPU



Downloading a Program:

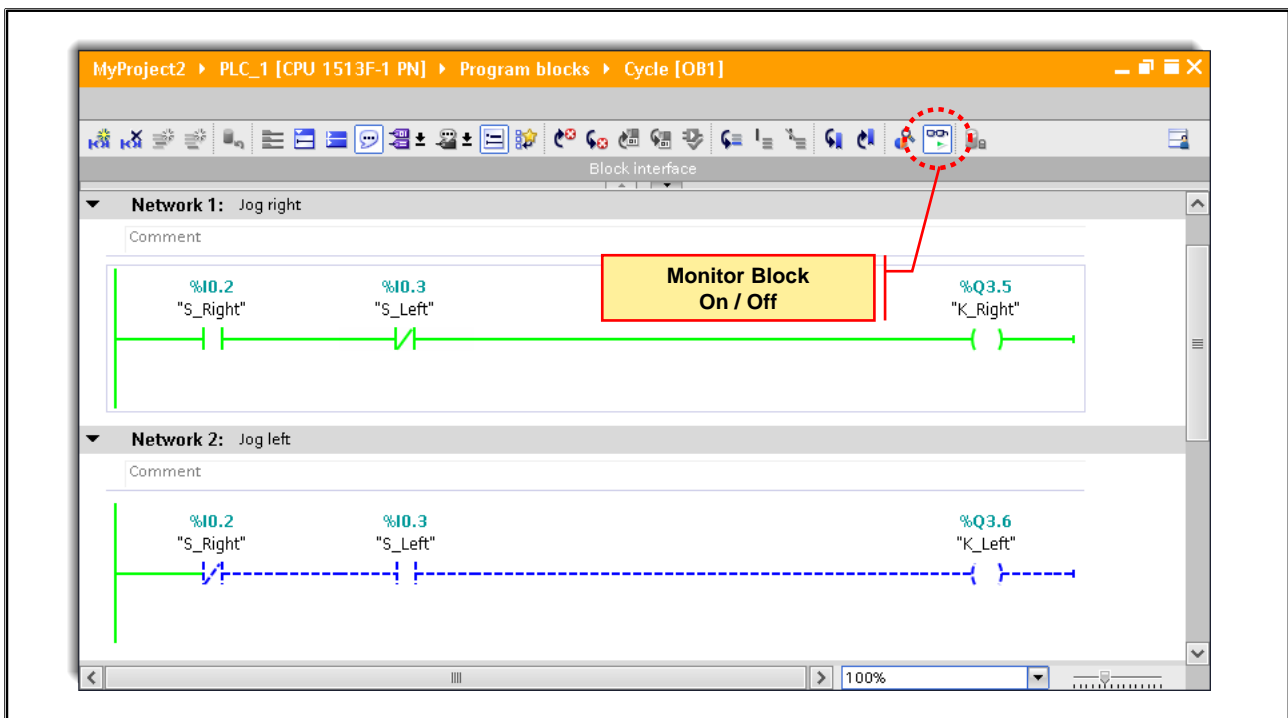
Program (Software) includes the blocks of the user program. The first time you download, they are completely loaded. In subsequent downloads, only the software changes are loaded.

However, all changes are always downloaded, that is, the offline program and the online program are always the same after the download.

Note:

In the context menu of the device (CPU) there is also the menu item "Software (all)" in which all blocks are loaded even if these have already been downloaded to the CPU. You have to make sure, however, that the CPU is stopped in this case.

6.6.9. Monitoring a Block



Monitor

The test function *Monitor* is used to track the program execution within a block. The statuses or contents of the operands used in the block at the time of program execution are displayed on the monitor.

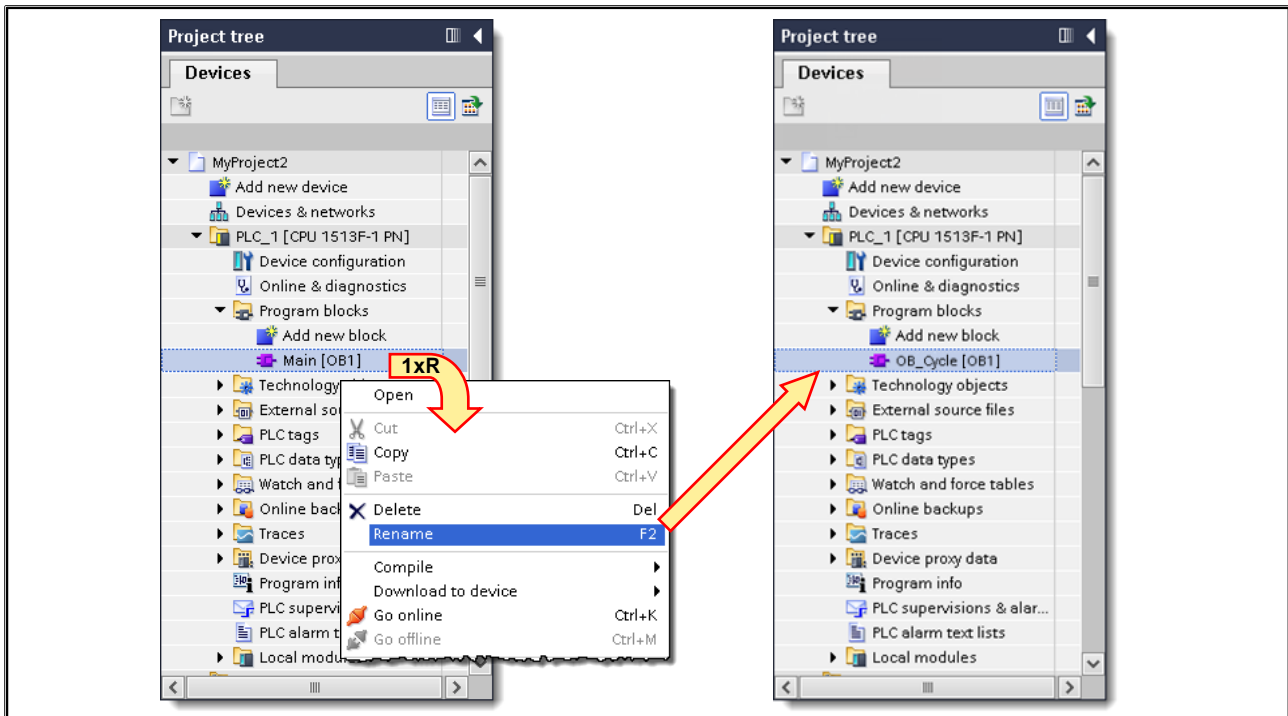
You can only monitor if there is an online connection to the CPU and the offline program is identical to the online program.

In test mode, the statuses of the operands and LAD / FBD elements are represented by different colors.

Examples:

- Status fulfilled → "Element is represented with a green color"
- Status not fulfilled → "Element is represented with a blue color"

6.7. Exercise 1: Renaming Main OB



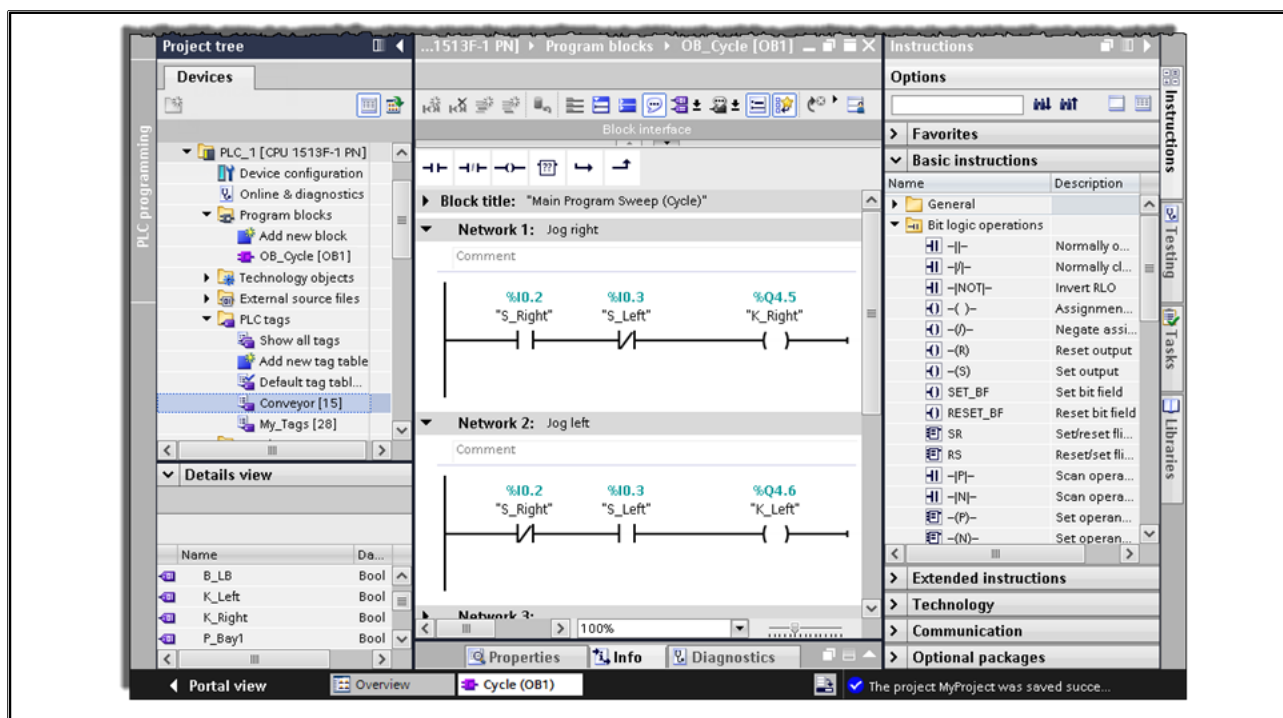
Task

You are to rename the Organization block.

What to Do

1. Open the "Program blocks" folder.
2. Open the context menu of the block "Main" by right-clicking on it.
3. Select the menu item "Rename".
4. Give the block the symbolic name "OB_Cycle".
5. Save your project.

6.7.1. Exercise 2: Programming the Jog Mode



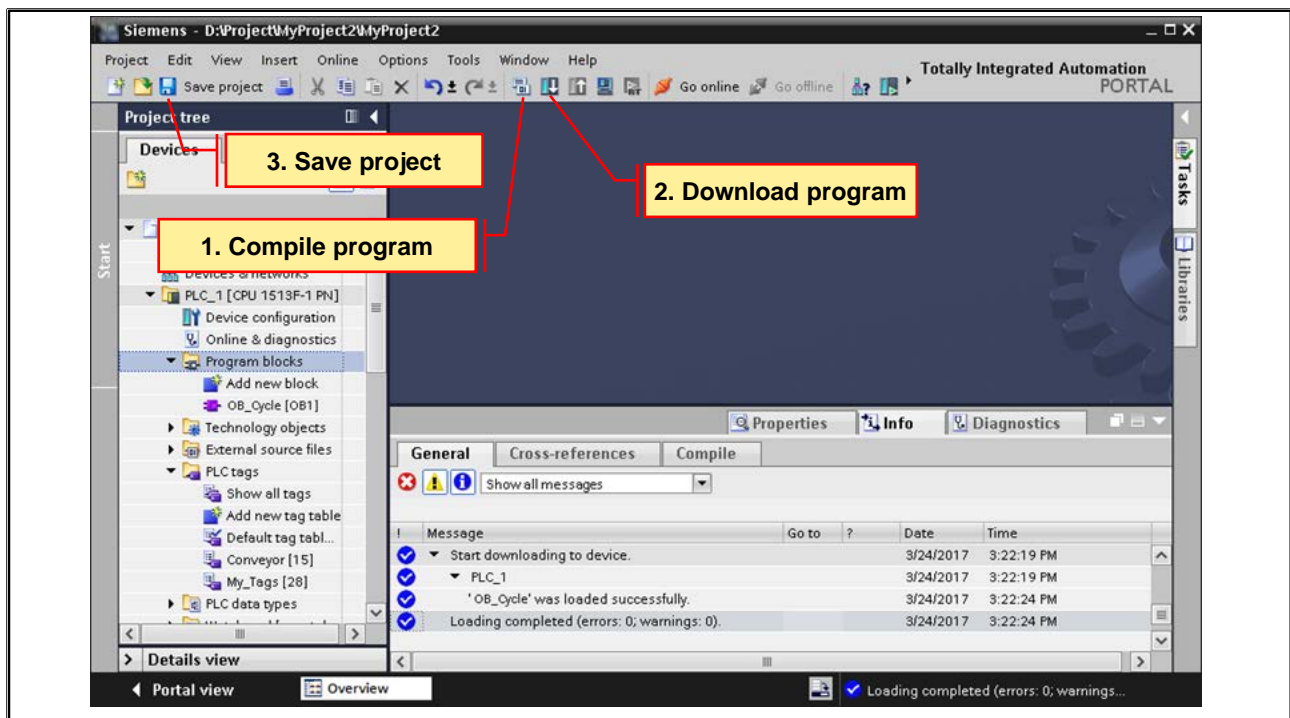
Task

You are to program the Jog mode of the conveyor as shown in the picture.

What to Do

1. Program an AND logic operation by dragging a "Normally open contact" and a "Normally closed contact" from the Favorites into the network using drag & drop.
2. Program an "Assignment" by dragging it from the "Instructions" Task Card to the AND logic operation using drag & drop as shown above.
3. At the first input of the AND logic operation enter the input "S_Right" (I 0.2) as operand (you can enter the symbol as well as the absolute address). Do the same for "S_Left" (I 0.3).
4. In the Project tree, select (do not open!) the tag table "Conveyor" and drag the tag "K_Right" (Q3.5) from the "Details view" as operand above the assignment.
5. Give the network a title.
6. Add a new network and there program an appropriate logic operation for jogging the conveyor to the left.
7. Close the Organization block.
8. Save your project.

6.7.2. Exercise 3: Compiling the Program and Downloading it into the CPU



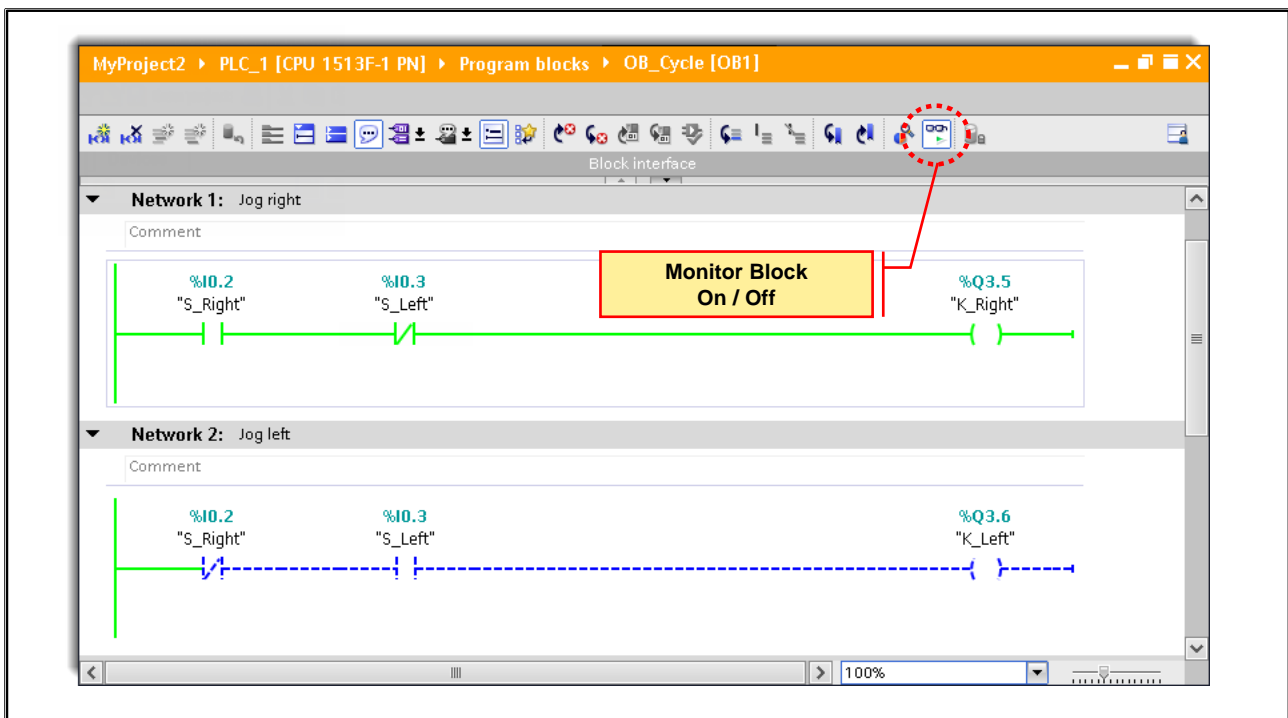
Task

You are to compile and download the program and save the project.

What to Do

1. Select the "Program blocks" folder.
2. Compile your program by means of the "Compile" button in the toolbar.
3. Download your program by means of the "Download" button in the toolbar.
4. Save your project.

6.7.3. Exercise 4: Monitoring the Program



Task

You are to monitor the program online.

What to Do

1. Open "OB_Cycle".
2. Monitor the block (the program) by means of the "Monitor On/Off" button.
3. Activate the switches "S_Left" and "S_Right" on your training case.