Contents

8

8. Binary Operations

At the end of the chapter the participant will …

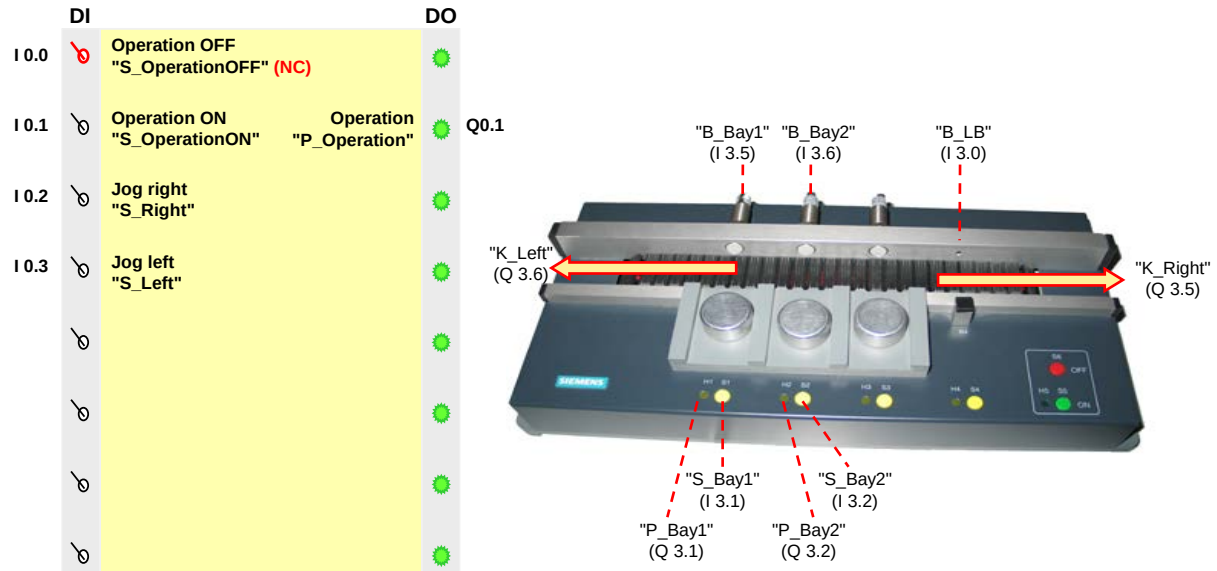
… be familiar with the instructions Set, Reset and will be able to apply them

… know what needs to be considered for a multiple assignment

… know how edges are evaluated

… be familiar with jump instructions as well as absolute and conditional block aborts

8.1. Task Description: The Conveyor Model as Distribution Conveyor with Manual and Automatic Mode



Function Description

The distribution conveyor is used to transport parts from Bay 1 or 2 to the light barrier bay. The Operation (Simulator LED "P_Operation", Q0.1) can be switched on and switched off (Manual / Automatic mode) via the simulator switch "S_OperationON" (I0.1) and "S_OperationOFF" (I 0.0, NC).

- In Manual mode, switched off "P_Operation" (Q0.1)…

…the distribution conveyor can be jogged to the right via the simulator switch "S_Right" (I 0.2) and jogged to the left via the simulator switch "S_Left" (I 0.3).

- In Automatic mode, switched on "P_Operation" (Q0.1)…

... parts are transported from Bay 1 or 2 up to, or, through the light barrier. For this, the part must be placed on the conveyor at exactly one of the two bays and the associated bay pushbutton pressed.

The indicator lights at Bays 1 and 2 show...

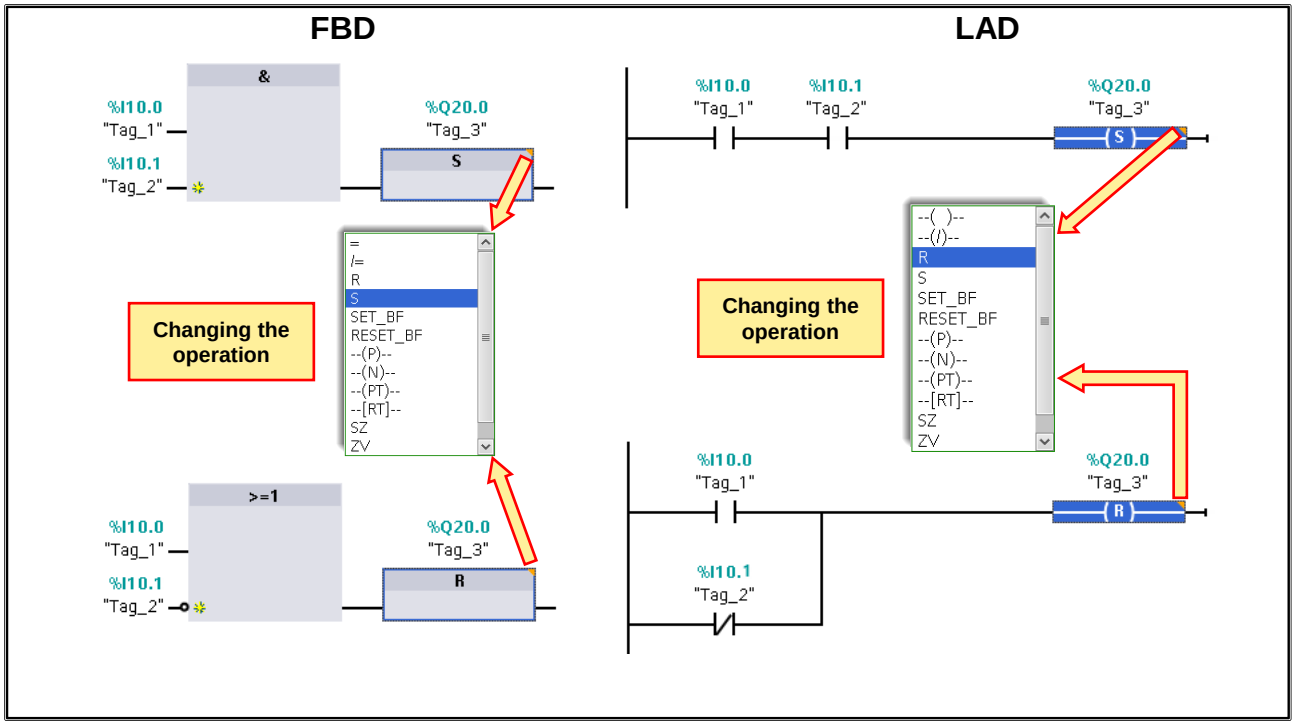
... a constant light when a new part may be placed on the conveyor (distribution conveyor is stopped and both proximity sensors are free)

... a 1Hz flashing light at the bay where a part is detected by the associated proximity sensor, however, only as long as the conveyor has not yet been started (if parts are placed on the conveyor at both proximity sensors, neither indicator light must light up)

... a 2Hz flashing light as long as the distribution conveyor is running.

The indicator light at the light barrier bay shows a 2Hz flashing light as long as the distribution conveyor is running.

8.2. Set, Reset



Set

If the result of a logic operation (RLO), that is, result of scan = "1", the specified operand is assigned Status '1'; for RLO / result of scan = "0", the status of the operand remains unchanged.

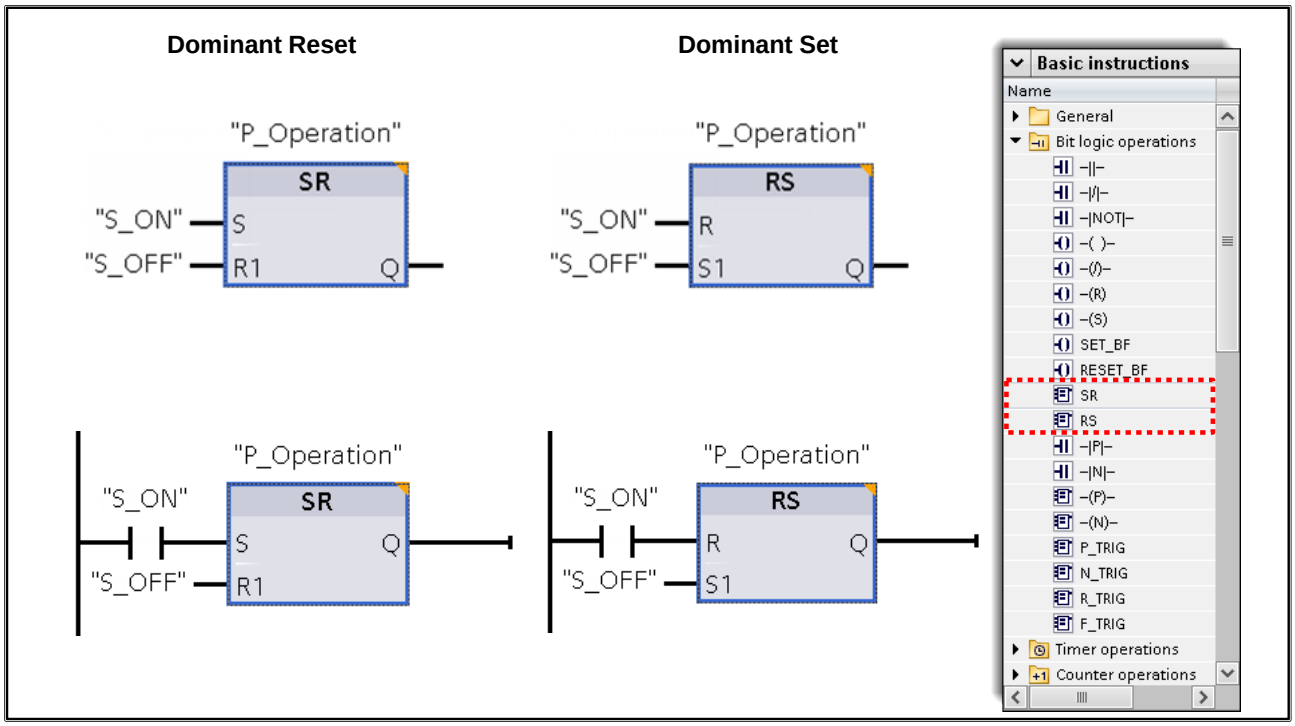
Reset

If the RLO, that is, result of scan = "1" the specified operand is assigned Status ' 0'; otherwise the status of the operand remains unchanged.

Note

In addition to the SET and RESET functions, the graphic above also shows the methodology to change an elements attributes without deleting and inserting a new element type. Simply highlight the element and click the small red "array" in the right corner of the element. Available changes are shown in a drop down menu.

8.2.1. Flip Flops



Memory Function "Flip Flop"

A flip flop has a Set input and a Reset input. The operand is set (TRUE) or reset (FALSE), depending at which input the scan condition is fulfilled.

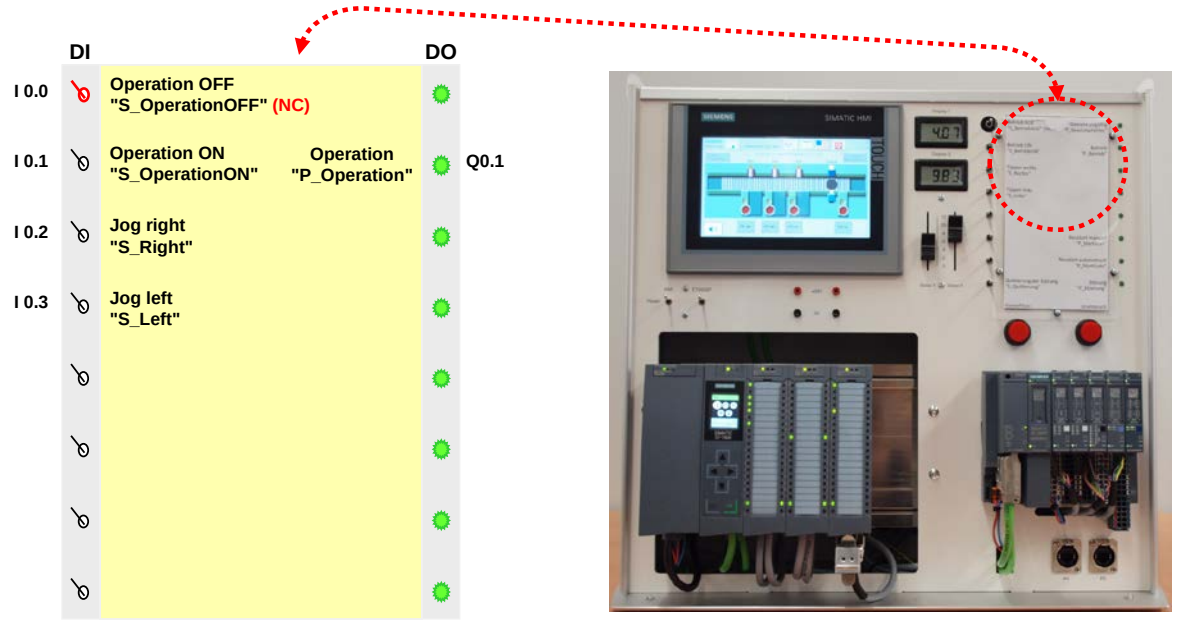
Priority

If the scan condition at both inputs is fulfilled simultaneously, then the priority of the operation decides whether the operand is set or reset. For that reason, there are different symbols for the Dominant Set and Dominant Reset memory functions in LAD and FBD. (The last scan always has priority, Set for "RS" and Reset for "SR").

Note

With a CPU restart, all outputs are reset. That is, they are overwritten with Status '0'.

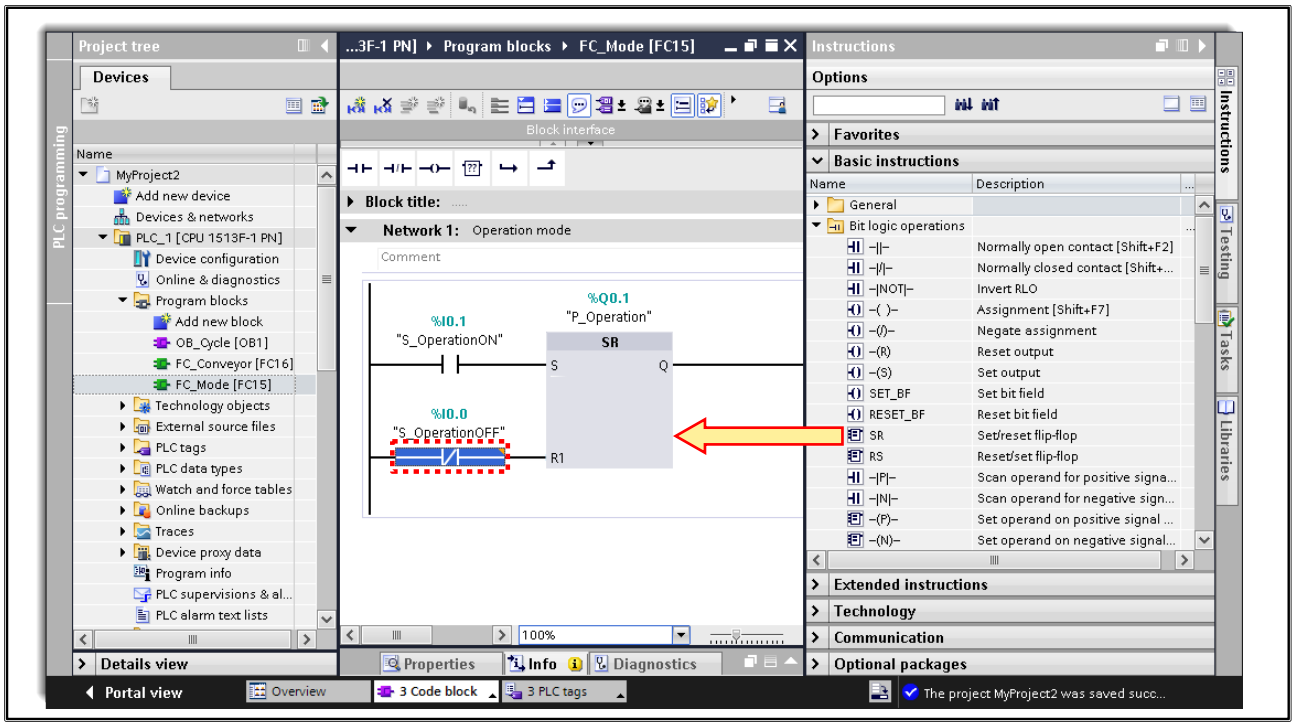
8.3. Task Description: Expanding the "FC_Mode" Function



Task Description

The "FC_Mode" function is to be changed in such a way that the output "P_Operation" is not assigned via the input "S_OperationON" but set by the input (NO) "S_OperationON" and reset via the input (NC) "S_OperationOFF".

8.3.1. Exercise 1: Programming the "FC_Mode" Block Expansion



Task

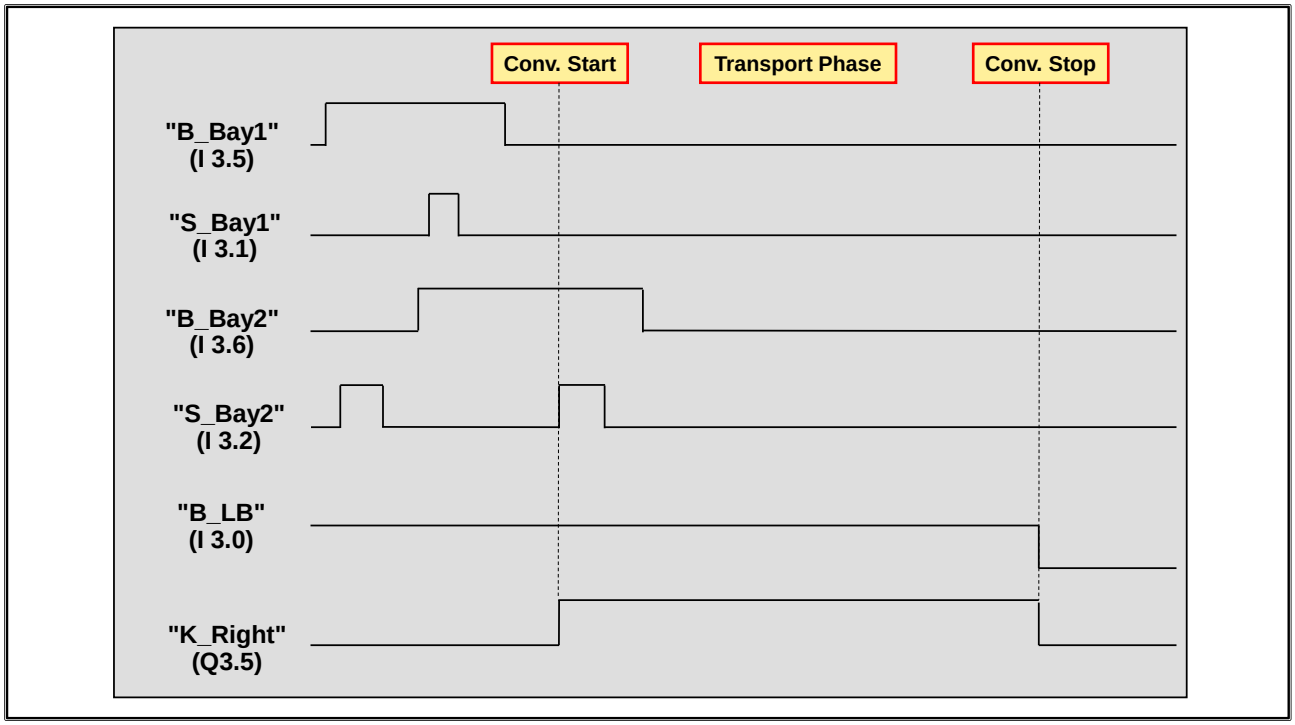
In "FC_Mode" you are to change the code in the operating mode section for the distribution conveyor:

The operation, that is, the indicator light "P_Operation" (simulator LED Q0.1) is switched on via the simulator switch "S_OperationON“ (I 0.1) and is switched off via the simulator switch "S_OperationOFF" (I 0.0, **NC**).

What to Do

1. Open the "FC_Mode" block.
2. Delete the assignment programmed in the last chapter.
3. Open the "Instructions" Task Card and drag the instructions shown in the picture into the block using drag & drop.
4. Link the input for ‘Set’ with the variable "S_OperationON" and the ‘Reset’ input with the variable "S_OperationOFF".
5. Since NC contacts are used to switch off systems for safety reasons (interruption-safe), the input "S_OperationOFF" must be negated.
6. Above the instruction, specify "P_Operation" as the operand.
7. Save, compile, download and test your newly programmed function.

8.4. Task Description: Parts Transportation in Automatic Mode



Task Description:

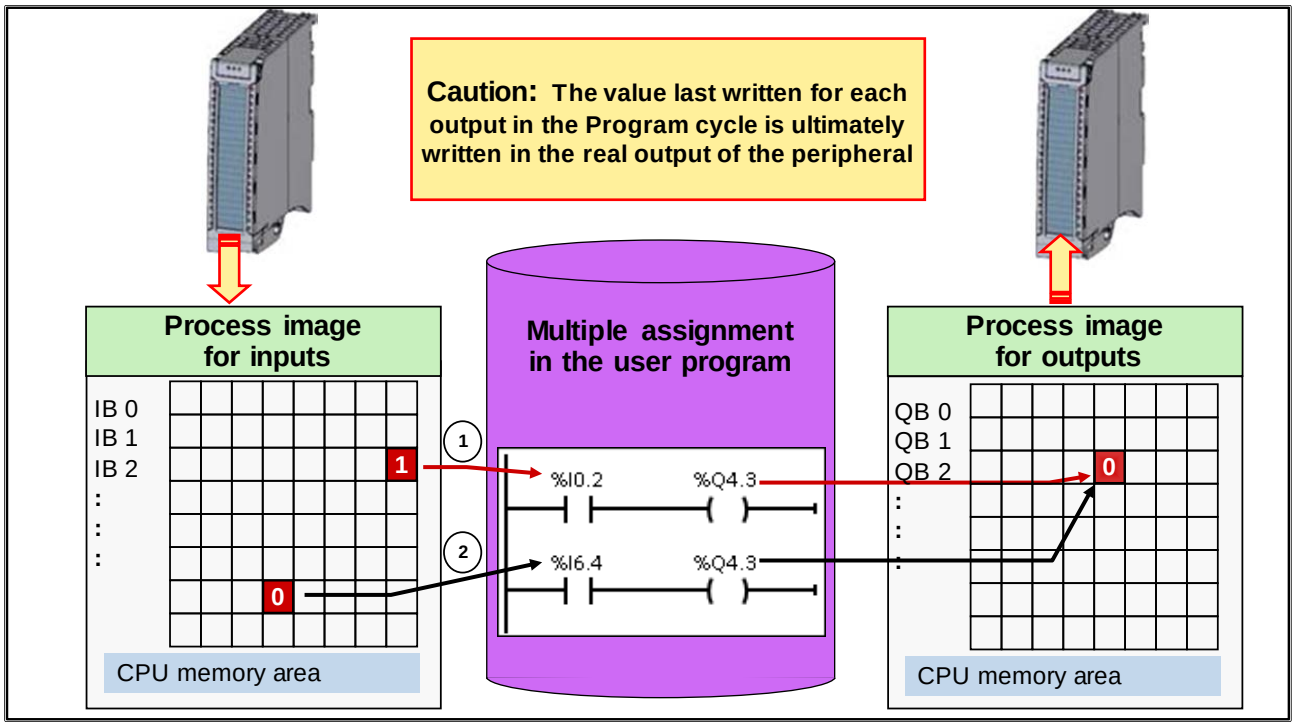
When "P_Operation" (Q0.1) is switched on, parts are to be transported from Bay 1 or Bay 2 into the light barrier "B_LB" (I 3.0). The precondition is that a part is only placed on the conveyor at one of the two bays. If parts are placed on the conveyor at both bays, no transport sequence can be started.

- The conveyor motor "K_Right" (Q3.5) is started when
  - at Bay 1 the proximity sensor "B_Bay1" (I 3.5) is occupied AND at Bay 2 the proximity sensor "B_Bay2" (I 3.6) is NOT occupied AND the pushbutton at Bay 1 "S_Bay1" (I 3.1) is pressed

  OR
  - at Bay 2 the proximity sensor "B_Bay2" (I 3.6) is occupied AND at Bay 1 the proximity sensor "B_Bay1" (I 3.5) is NOT occupied AND the pushbutton at Bay 2 "S_Bay2" (I 3.2) is pressed.
- The conveyor motor "K_Right" (Q3.5) is stopped when
  - the part has reached the light barrier "B_LB" (I 3.0)

  OR
  - "P_Operation" (Q0.1) is switched off.

8.4.1. Multiple Assignment



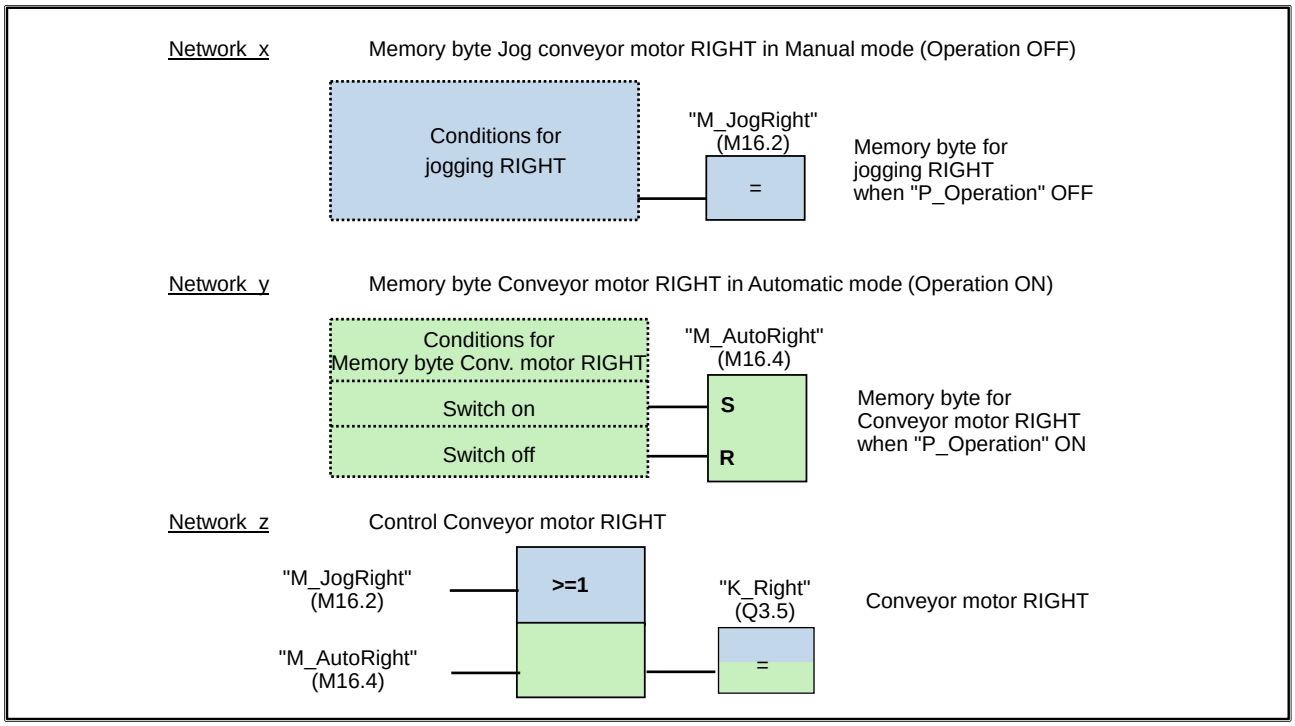
Process Images

For the storage of **all** digital input and output statuses, 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.

"Classic Error": Double Assignment

If an output is assigned a status in several locations in the program, then only the status that was assigned last is transferred to the particular output module. As a rule, these types of double assignments are programming errors.

8.4.2. Exercise 2: Expanding "FC_Conveyor" (Automatic Mode)



Task

You are to expand the "FC_Conveyor" block with the previously described functions.

Solution Notes

The output to move the distribution conveyor to the right "K_Right" (Q3.5) must now be controlled given the following two conditions:

Either when "P_Operation" (Q0.1) is switched off for Jogging RIGHT (in the picture Network x) OR when "P_Operation" (Q0.1) is switched on under the conditions described in the Task Description (in the picture Network y).

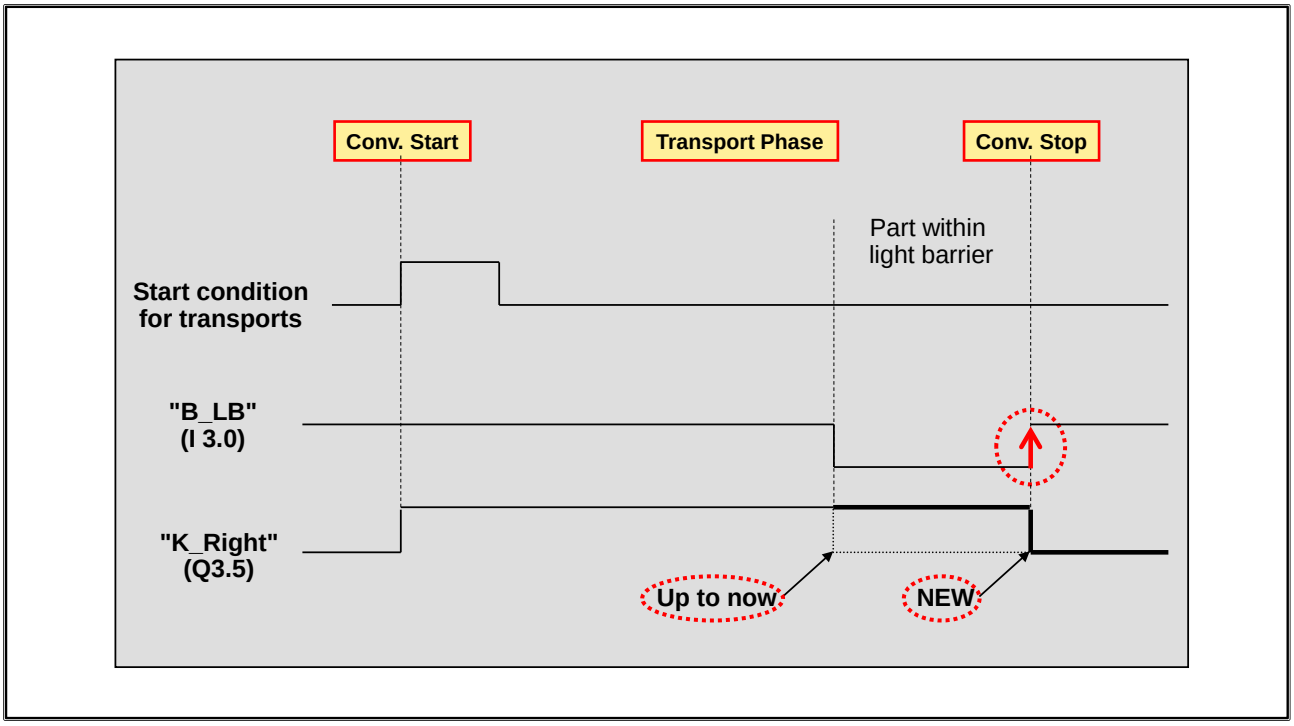
If, in both Network x and y, the result of logic operation of the conditions were each directly assigned to the output "K_Right" (Q 3.5), an error in the form of a double assignment would occur. Jogging the motor RIGHT in manual mode (Network x) would no longer function, since the status assigned to the output here would then be overwritten in Network y.

The problem can be solved by programming a memory bit for each condition or by first assigning the results of the logic operations to a memory bit each in both Network x and y. These are then used in the Network to control the conveyor motor.

What to Do

1. Open the "FC_Conveyor" block.
2. Program the required functions in new networks (see graphic above and the graphic at the previous page). Use the memory bits shown in the picture and provide them with the symbols shown.
3. Modify the already existing network for jogging to the right as described in the solution notes.
4. Download the expanded block into the CPU and check the program function.
5. Save your project.

8.5. Task Description: Parts Transportation THROUGH the Light Barrier



Function Up to Now in FC_Conveyor

When "P_Operation" (Q0.1 = '0') is switched off, you can jog the conveyor motor "K_Right" (Q3.5) using the simulator switches "S_Left" (I 0.3) and "S_Right" (I 0.2).

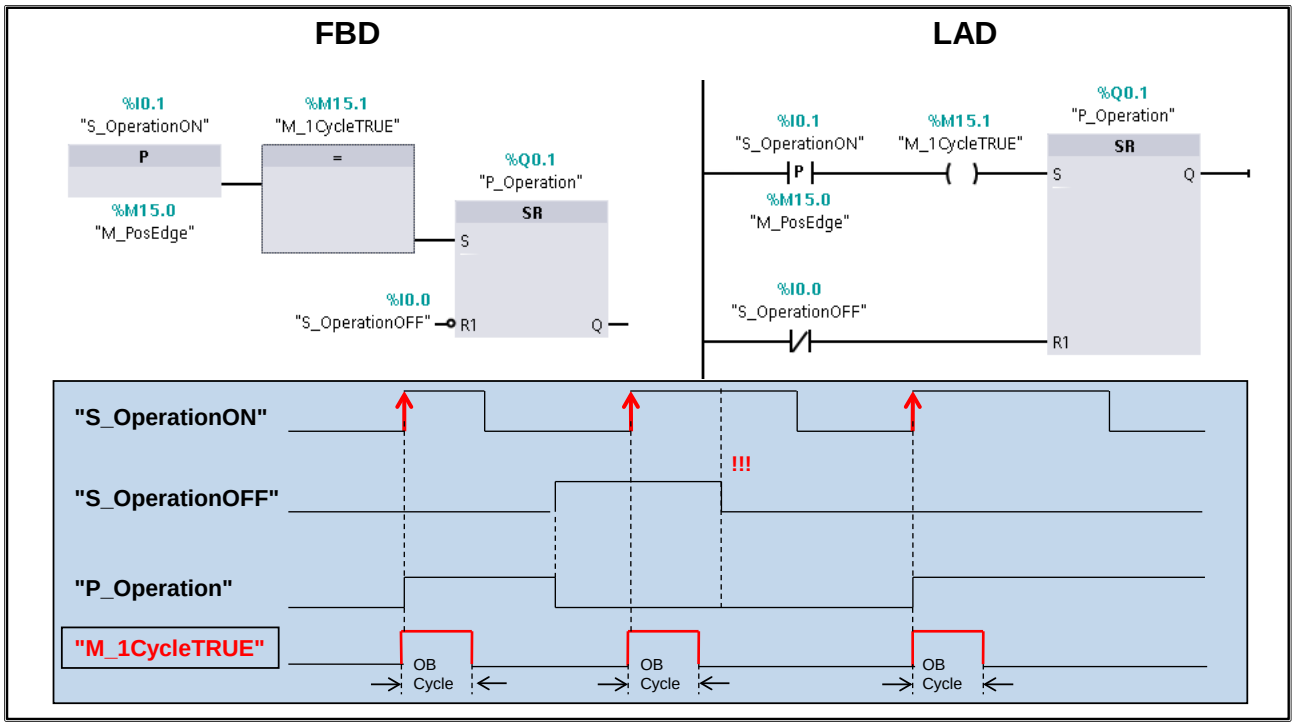
When "P_Operation" (Q0.1) is switched on, the conveyor motor "K_Right" (Q 3.5) is switched on when a part is placed on the conveyor exactly in front of one of the proximity sensors of Bay 1 or 2, and the occupied bay's pushbutton is pressed.

The conveyor motor "K_Right" (Q3.5) is stopped when the part has **reached** the light barrier or "P_Operation" (Q0.1) is switched off.

Task Description:

The function of "FC_Conveyor" to control the conveyor motor when "P_Operation" (Q0.1) is switched on is to remain fundamentally unchanged. However, the conveyor motor is only to stop when the part has **passed through** the light barrier.

8.5.1. Signal Edge Evaluation



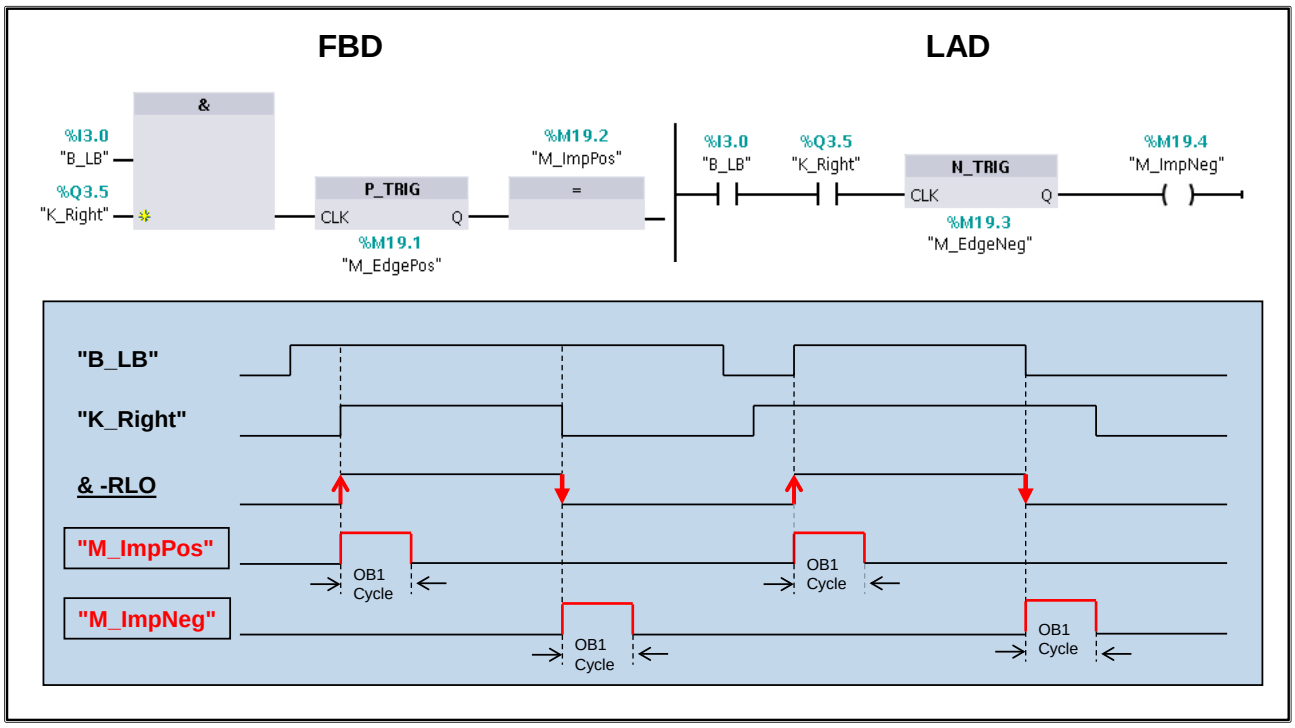
Scanning the Operand for Positive (–|P|–) or Negative (–|N|–) Signal Edge

With a signal edge evaluation, it is possible to detect whether the status of an individual operand (in the example "S_OperationON") has changed from '0' to '1' (rising or positive edge) or from '1' to '0' (falling or negative edge). If this is the case the instruction supplies RLO '1' as the result, which can be further logically linked (in the example as Set condition) or can be assigned to another operand (for example, a memory bit) as status. In the following cycle, the instruction then once again supplies '0' as the result even if "S_OperationON" still is status '1'.

The instruction compares the current status of the operand "S_OperationON" with its status in the previous program cycle. This status is stored in a so-called edge memory bit (in the example "M_PosEdge"). It must be ensured that the status of this edge memory bit is not overwritten elsewhere in the program. For each edge evaluation, a separate edge memory bit must be used accordingly, even then when the same operand (in the example "S_OperationON") is evaluated again, for example, in another block!

As well, the edge is assigned to a pulse memory bit (M_1CycleTRUE) in the example. This has the advantage that the edge of the operand is only scanned once in the program. If, in the further course of the program, the edge of the operand must once again be evaluated, no further edge memory bit is required but rather the status of the pulse memory bit can be scanned.

8.5.1.1. RLO Edge Evaluation



RLO Edge Evaluation (P_TRIG, N_TRIG)

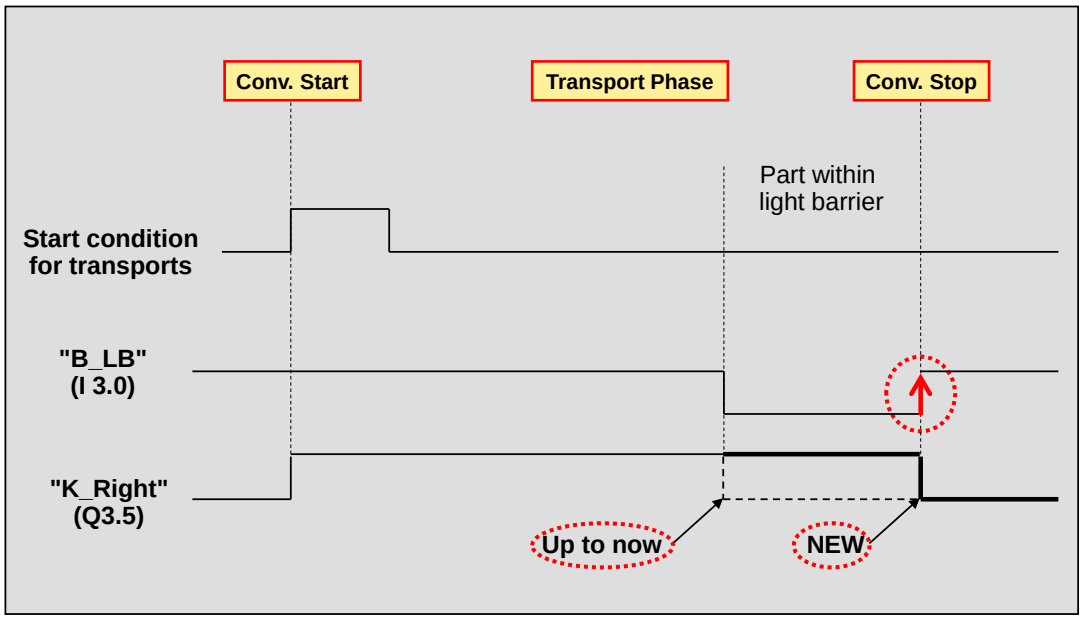
With an RLO edge evaluation, it is possible to detect whether the status of an individual operand or the result of a logic operation has changed from '0' to '1' (rising or positive edge) or from '1' to '0' (falling or negative edge). If this is the case, both edge evaluations supply the result RLO '1' to their output for the duration of one cycle. In the following cycle, the instructions then once again supply RLO '0' as the result even if the status or the RLO of the operand or the logic operation has not changed.

The instructions compare the current status of the operand or the result of a logic operation with its status in the previous program cycle which is stored in a so-called edge memory bit for this (in the example, "M_EdgePos" or "M_EdgeNeg). It must be ensured that the status of this edge memory bit is not overwritten elsewhere in the program. For each edge evaluation, a separate edge memory bit must be used accordingly, even then when the same operand is evaluated again, for example, in another block!

About the examples in the picture:

- For the instructions P_TRIG and N_TRIG, the result, that is, the RLO of the edge evaluation is available on the right at the output. It can be further logically linked or assigned to an operand (in the example, it is assigned to the operand "M_ImpPos" or "M_ImpNeg).

8.5.2. Exercise 3: Integrating an Edge Evaluation in "FC_Conveyor"



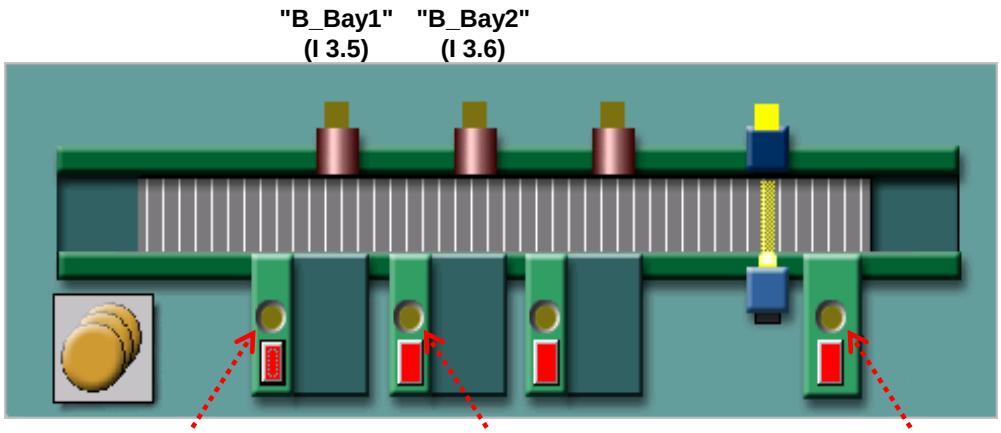
Task:

When "P_Operation" (Q0.1 = '1') is switched on, the parts are to be transported from Bay 1 or 2 THROUGH the light barrier.

What to Do:

1. Program the necessary changes in "FC_Conveyor", by now linking the result of the edge evaluation as the reset condition of the memory byte "M_AutoRight" (memory byte for conveyor motor RIGHT) instead of the light barrier signal "B_LB" (I 3.0) itself. For the necessary edge evaluation of the light barrier signal use the memory byte "M_AuxLB" (M16.0) as an edge memory bit.
2. Download the modified "FC_Conveyor" block into the CPU and check the program function.

8.6. Task Description: Controlling the Indicator Lights, Commissioning "FC_Signal"



Constant light:	**A new part can be placed**	
1Hz flashing light:	**Conveyor can be started**	
2Hz flashing light:	**Caution, conveyor motor is running**	**2Hz flashing light: Caution, conveyor motor is running**

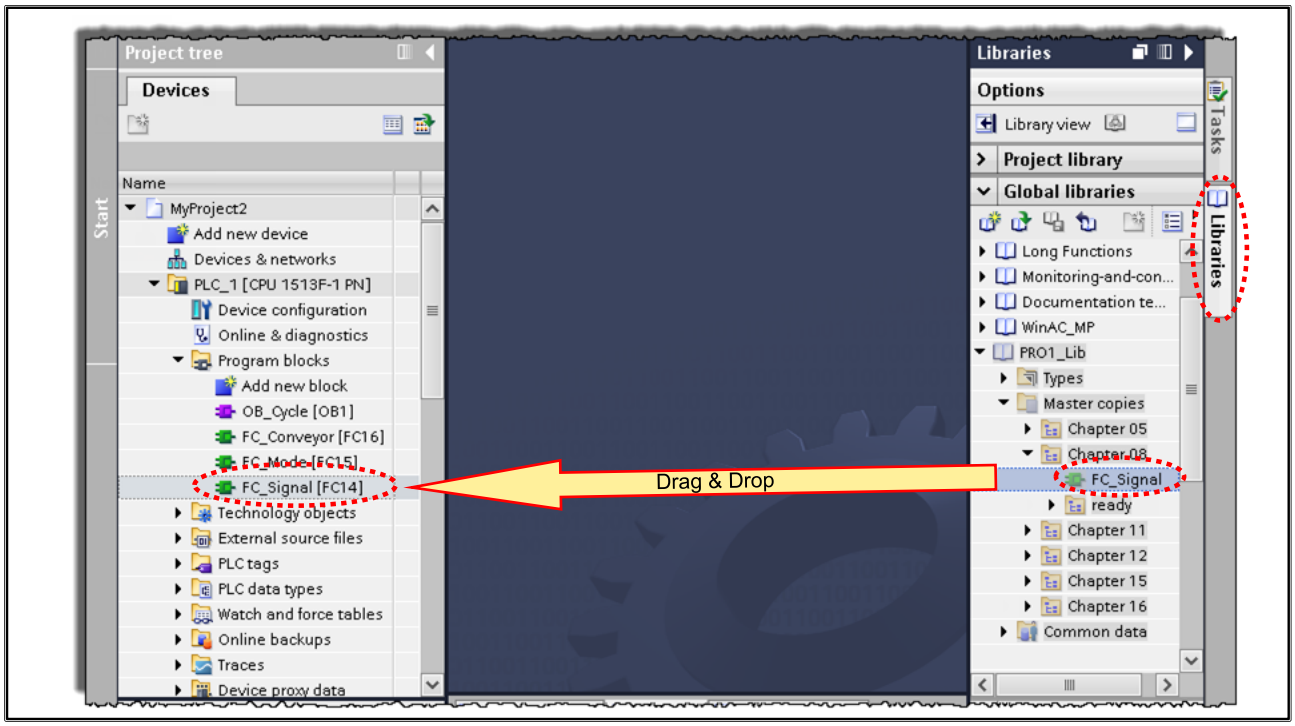
Task Description

When "P_Operation" (Q0.1) is switched on, the indicator lights "P_Bay1" (Q3.1) and "P_Bay2" (Q3.2) are to be controlled as follows:

- They show
  - a constant light when a new part may be placed on the conveyor (distribution conveyor is stopped and both proximity sensors are free)
  - a 1Hz flashing light at the bay where a part is detected by the associated proximity sensor, however, only as long as the conveyor has not yet been started (if parts are placed on the conveyor at both proximity sensors, neither indicator light must light up)
  - a 2Hz flashing light as long as the distribution conveyor is running
- The indicator light at the light barrier bay "P_BayLB" (Q3.4) shows a 2Hz flashing light as long as the conveyor motor is running.

The described functions are already partially programmed in the "FC_Signal" block which is stored in the "PRO1_Lib" global library. The block is to be commissioned and completed by you in the next exercise.

8.6.1. Exercise 4: Writing/Correcting "FC_Signal" and Commissioning It



Task

You are to either program the "FC_Signal" block yourself or you are to copy it from the "PRO1_Lib" global library into the project, and then commission and complete it.

What to Do

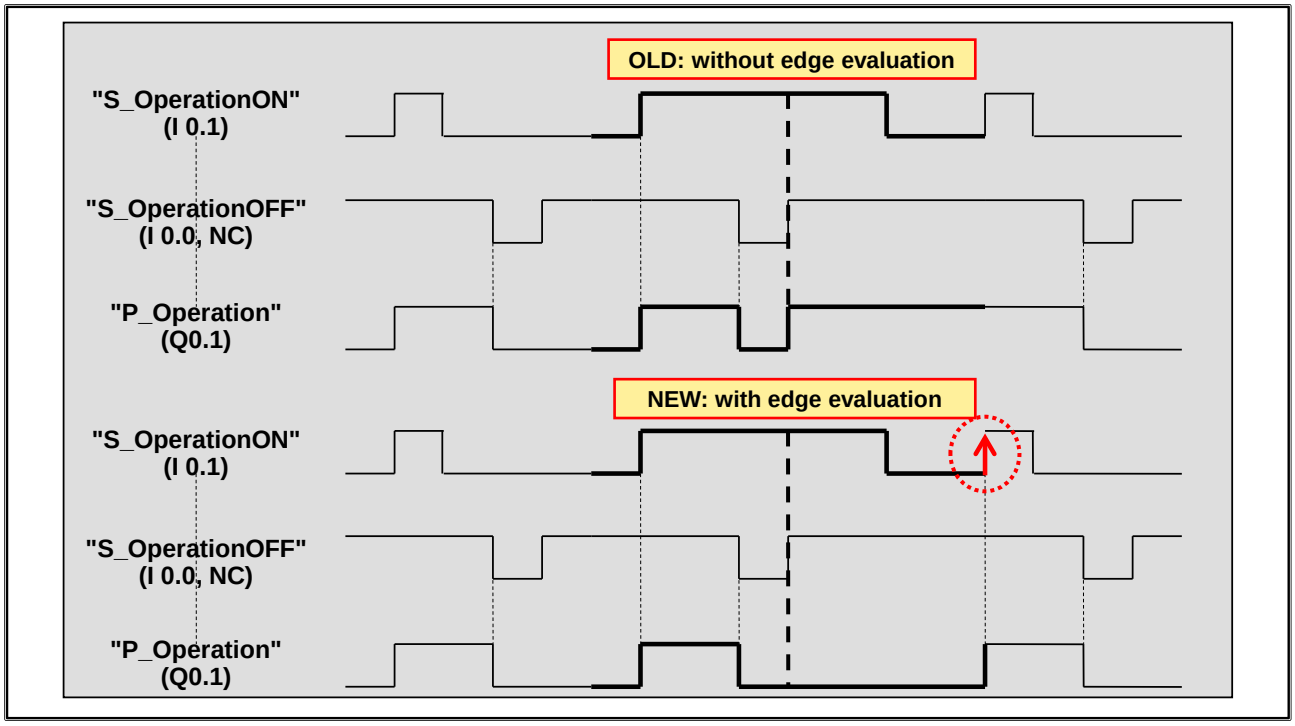
Create the new "FC_Signal" block and program the required functions or:

1. Using drag & drop, copy the "FC_Signal" block into the "Program blocks" folder from the "PRO1_Lib" global library (as shown in the picture).
2. Complete the block so that the indicator lights "P_Bay1" (Q3.1) and "P_Bay2" (Q3.2) show the required signals. (The indicator light "P_BayLB" is already programmed.)
3. Program the call of "FC_Signal" in "OB_Cycle".
4. Download all modified blocks into the CPU and test the program function.
5. Save your project.

Note:

The completely programmed blocks can be found in the library in the folder "Chapter 08 > ready".

8.7. Additional Exercise: Optimizing "FC_Mode"



Function Up to Now of "FC_Mode"

"P_Operation" (Q0.1) is switched on with the simulator switch "S_OperationON" (I 0.1) and switched off with the simulator switch "S_OperationOFF" (I 0.0, **NC**). If you activate both switches simultaneously, the operation remains switched off or is switched off if currently on. If, however, both switches are activated and the OFF switch is let go while the ON switch is activated, the operation switches back on without the ON switch having to be activated again (see picture, upper function diagram "OLD: without edge evaluation").

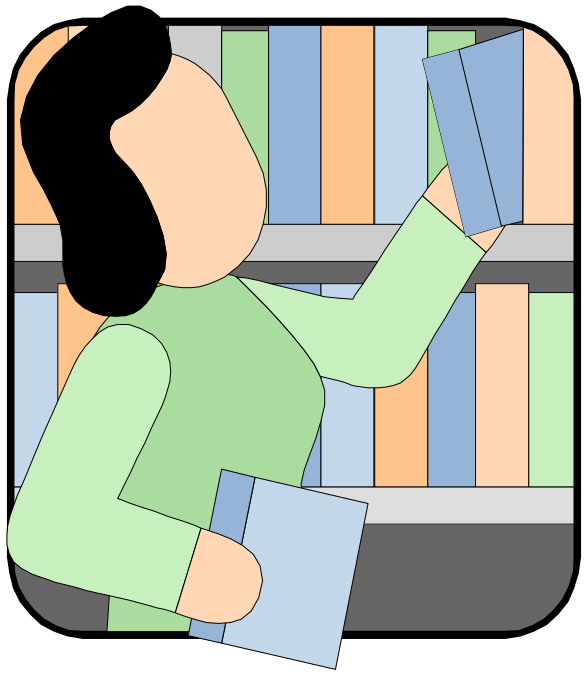
Task:

You are to expand the functionality of "FC_Mode" using edge evaluation so that the ON switch must be activated every time the operation is switched on (see picture, lower function diagram "NEW: with edge evaluation"). The criteria for switching on the system is no longer to be the activated ON switch or its "1" signal, but the function of activating or the "positive edge" of the ON switch signal.

What to Do:

1. In the set condition for "P_Operation" (Q0.1), insert an edge evaluation of the switch "S_OperationON" (I 0.1). For the edge evaluation, use the memory byte "M_AuxOpON" (M15.1) as edge memory bit.
2. Download the modified "FC_Mode" into the CPU and check whether it fulfils the desired functions!

8.8. Additional Information

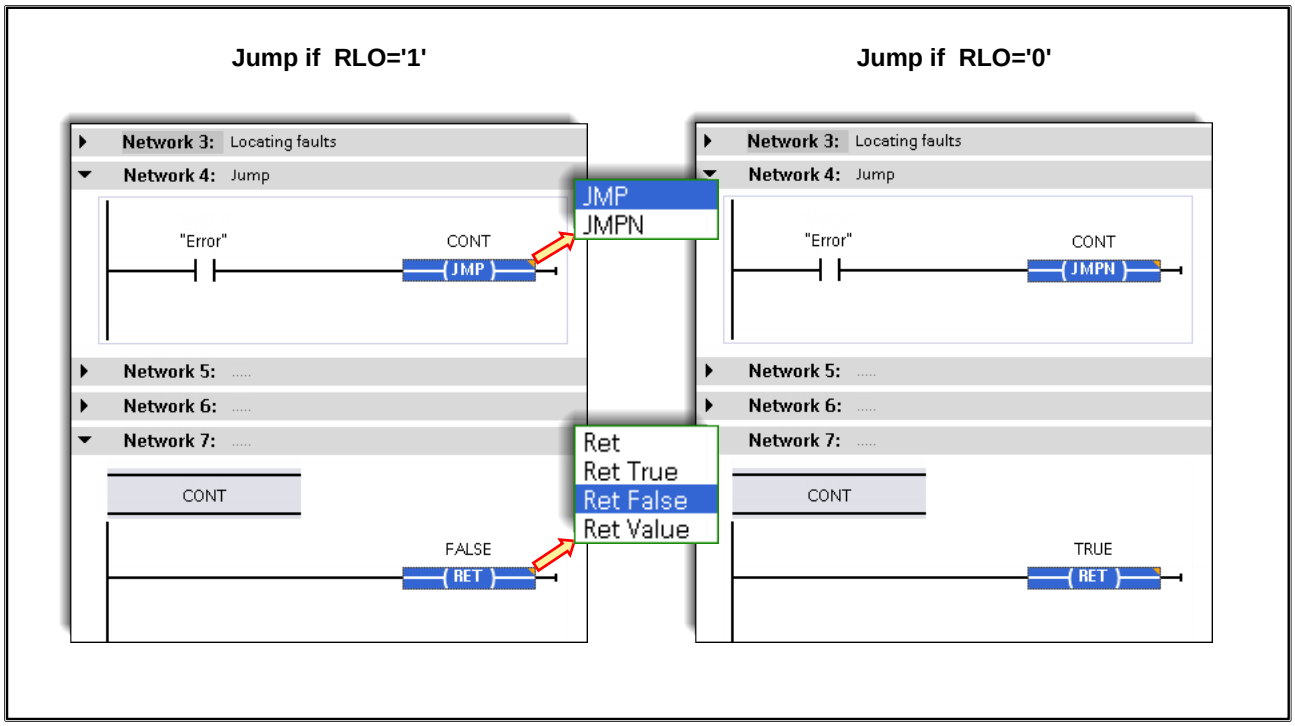


Note

The following pages contain either further information or are for reference to complete a topic.

For more in-depth study we offer advanced courses and self-learning mediums.

8.8.1. Jump Instructions JMP, JMPN, RET



Jump Instructions JMP and JMPN

With the jump instructions JMP and JMPN, the linear execution of the program can be interrupted within a block and continued in another network. With the jump instruction, a Label is specified which also identifies the target network. The specified label must be located in the same block and be unique. Each label can be jumped to from several locations. The jump can take place in networks with higher (forwards) or lower numbers (backwards).

- **JMP:**
  If RLO ='1', the jump into the target network is executed; if RLO ='0', the jump is not executed and the linear program execution continues.
- **JMPN:**
  If RLO ='0', the jump into the target network is executed; if RLO ='1', the jump is not executed and the linear program execution continues.

End Block Execution RET

With the instruction RET the program execution of the entire block is ended. If the input of the instruction is assigned then this is a “Conditional block end” and only leads to a block abort when the scan is fulfilled.
Each block has an "ENO" enable output through which the information as to whether the block was executed without error can be communicated to the calling block. With the help of the RET function, the "ENO" enable output can be written.

- **Ret** (RLO = the result of logic operation. The calling block is supplied the signal status "1" since the RET instruction, as conditional instruction, is only executed when the condition is TRUE.)
- **Ret True** or **Ret False** (The respective value of the constants, TRUE or FALSE, is supplied to the calling program block.)
- **Ret Value** (The value of the Boolean variable <Operand>, which is specified above the instruction, is supplied to the calling program block.)