7.2. Sensors and Symbols

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.

Symbols

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

Example

If an NC contact in the machine is not activated, the signal in the process image table will be "1". You use the NO contact symbol in LAD to check for a signal state of "1". General: The "NC contact" symbol delivers the result of check "1" when the checked address state or status is "1".

<table>
<thead>
<tr>
<th>Process</th>
<th>Interpretation in the PLC Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>The sensor is a ...</td>
<td>Signal state at input</td>
</tr>
<tr>
<td>NO contact</td>
<td>activated</td>
</tr>
<tr>
<td></td>
<td>not activated</td>
</tr>
<tr>
<td>NC contact</td>
<td>activated</td>
</tr>
<tr>
<td></td>
<td>not activated</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.3. Binary logic operations: AND, OR

AND and OR logic operations

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

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

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 is Status '1'.
7.3.1. Binary logic operations: Exclusive - OR (XOR)

XOR logic operation

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

All inputs of the logic operations can be programmed as check for signal state 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 inputs signals is 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 programming languages FBD and LAD

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. ????????????????????
7.4. Assignment, Set, Reset, NOT

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.

Set

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

Reset

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

NOT

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

If, in the example shown, the RLO of the AND logic operation = '1', the NOT instruction inverts it to RLO '0' and the Set instruction is not executed (the status of "Tag_3" (Q20.0) then remains unchanged).

If the RLO of the AND logic operation = '0', the NOT instruction inverts it to RLO '1' and the Set instruction is executed ("Tag_3" (Q20.0) is assigned Status '1').
7.4.1. Flip-Flops

Flip Flop

A flip flop has a Set input and a Reset input. The memory bit is set or reset, depending on which input has an RLO=1.

Priority

If there is an RLO=1 at both inputs at the same time, the priority must be determined. In LAD and FBD there are different symbols for Dominant Set and Dominant Reset memory functions. In STL, the instruction that was programmed last has priority.

Note

With a warm restart of the CPU, all outputs are reset. That is, they are overwritten with the state '0'.
7.6.1. Signal – Edge Detection

Signal edge detection (P, N)

With a signal edge detection it is possible to detect the status change of an individual operand (in the example "T_ON") 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 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 "T_ON" still is status '1'.

The instruction compares the current status of the operand "T_ON" with its status in the previous program cycle. This status is stored in a so-called edge memory bit for this (in the example "M_FL_ON"). It must be ensured that the status of this edge memory bit is not overwritten at another location in the program. For each edge detection, a separate edge memory bit must be used accordingly, even then when the same operand (in the example, "T_ON") is detected once again, for example, in another block!
RLO edge detection (P= P_TRIG)

An RLO edge detection detects whether the status of an individual operand or the RLO 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 detections supply RLO '1' as a result to their output for the duration of one cycle. In the following cycle, the instructions then once again supply RLO '0' as a 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 RLO of the 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_FL_Count_pos" or "M_FL_Count_neg"). It must be ensured that the status of this edge memory bit is not overwritten at another location in the program. For every edge detection, a separate edge memory bit must be used accordingly, even then when the same operand is detected once again, for example, in another block!

On the examples in the picture:

- With the instruction P=, the result or the RLO of the edge detection is immediately assigned to another operand (in the example, the operand "M_Imp_Count_pos") and it is also available for further logic operations.
- With the instruction P_TRIG, the result or RLO must be assigned to another operand (in the example, "M_Imp_Count_neg") or further logically linked.
7.8.2. Bit logic: SET_BF, RESET_BF

Bit logic Set SET_BF / Reset RESET_BF

With the instruction "Reset bit logic", several bits can be reset starting from a specific address. The number of bits which are to be reset can be specified via Parameter N. The bit with the lowest address in the bit logic must be specified via the instruction. The bits are reset in ascending bit and byte addresses.

The instruction is only executed if an RLO ‘1’ is at input EN. For RLO ‘0’, the instruction is not executed.
7.8.3. Program control: 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:**
  For RLO = '1', the jump into the target network is executed; for RLO = '0', the jump is not executed and the linear program execution continues.

- **JMPN:**
  For RLO = '0', the jump into the target network is executed; for 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. The program execution is then continued in the calling block with the instruction that follows the call of this block.