4. Sample Task Function Block for Conveyor Control

When blocks are to be generated that are working in any program like a "Black Box“ as it were, they have to be programmed by using variables. In this case, the following rule applies: that in these blocks, no absolute-addressed inputs/outputs, flags etc. must be used. Within the block, only variables and constants are used.

In the example below, a function block is to be generated with a variable declaration containing a conveyor control that is dependent on the operating mode.

With button 'S1', the operating mode ’Manual’ and with button 'S2' the operating mode ’Automatic’ can be selected.
In the operating mode ’Manual’, the motor is switched on as long as button 'S3’ is operated, whereby button 'S4’ must not be operated.
In the operating mode 'Automatic’, the conveyor motor is switched on with button 'S3' and switched off with button 'S4' (break contact).

Assignment list:

Address	Symbol	Comment
%I 0.0	S1	Button operating mode Manual S1 NO
%I 0.1	S2	Button operating mode Automatic S2 NO
%I 0.2	S3	On button S3 NO
%I 0.3	S4	Off button S4 NC
%Q 0.2	M1	Conveyor motor M1

Note: The Off button S4 is a break contact here in order to ensure wire break safety. That means: if there is a wire break at this button, the system stops automatically. Otherwise, it could not stop if there were a wire break. For that reason, in control engineering all Stop buttons, Off buttons/switches have to be designed as break contacts.

5. Programming the Conveyor Control for the SIMATIC S7-1200

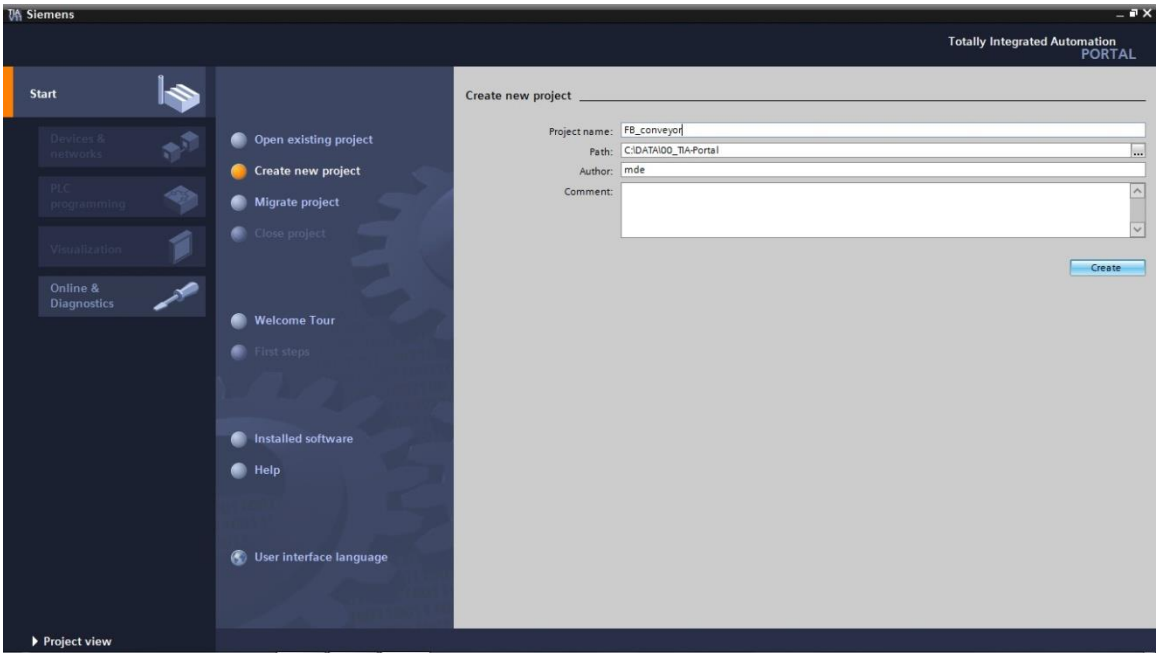
The project is managed and the components are programmed with the software **'Totally Integrated Automation Portal'**.

Here, under a uniform interface, the components such as controller, visual display and networking of the automation solution are set up, parameterized and programmed. Online tools are provided for error diagnosis

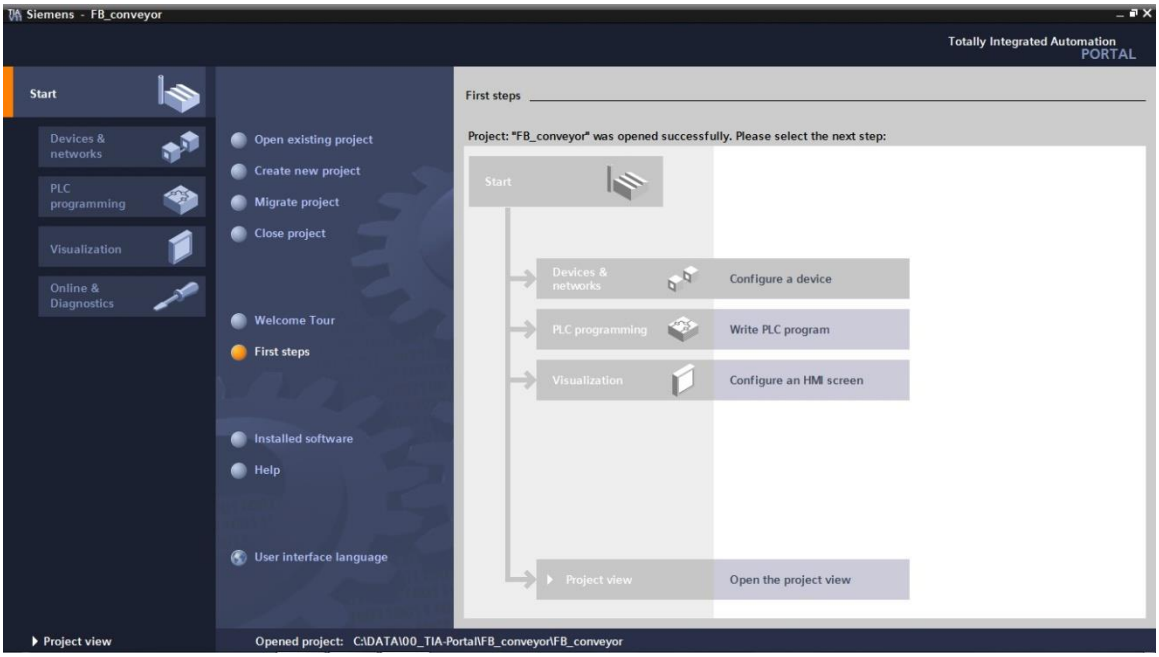
In the steps below, a project can be set up for the SIMATIC S7-1200 and the solution for a task can be programmed:

1. The central tool is the **'Totally Integrated Automation Portal'** that we call here with a double click (→ Totally Integrated Automation Portal V11)

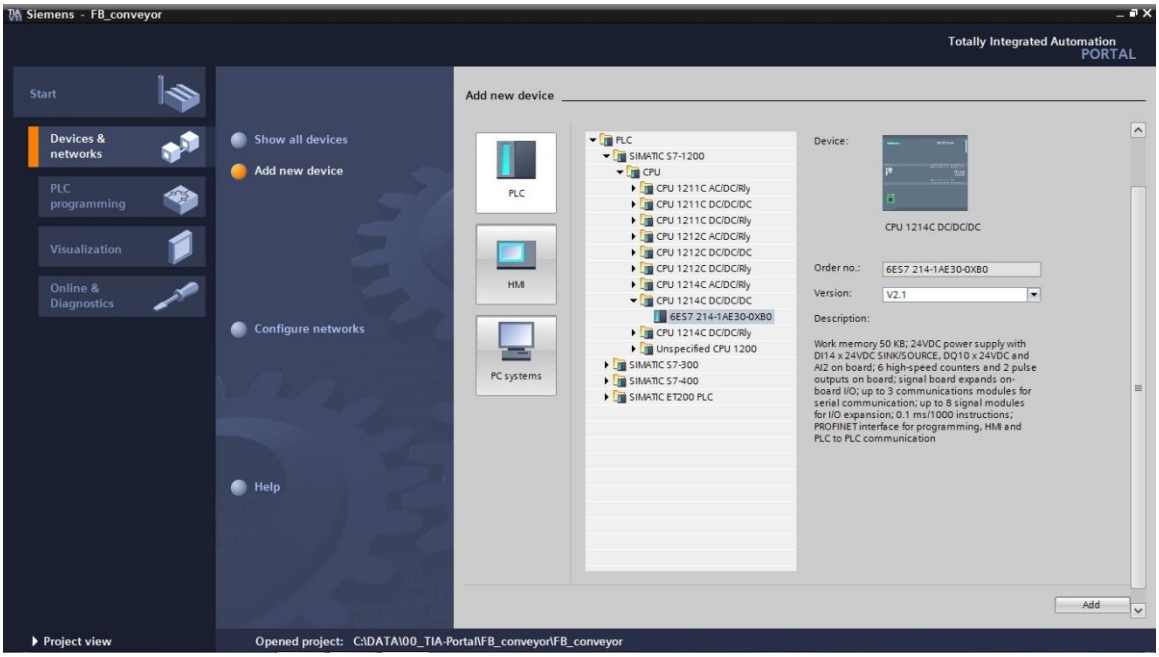
2. Programs for the SIMATIC S7-1200 are managed in projects. Such a project is now set up in the portal view → Create new project → FB_conveyor → Create)



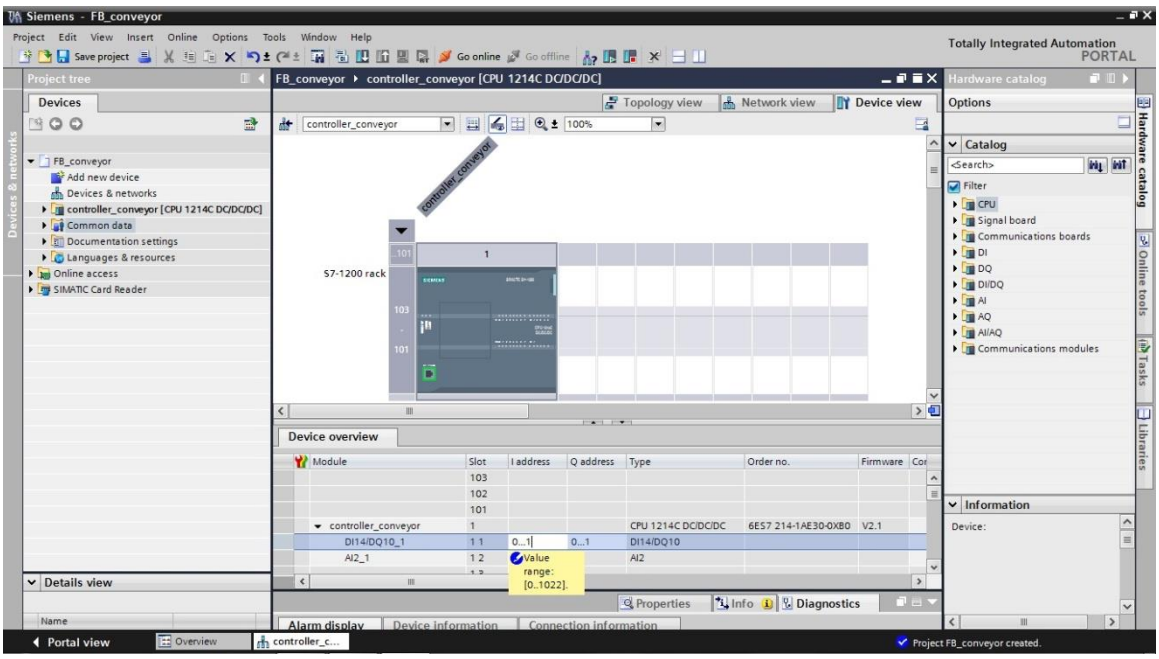
3. **'First Steps'** are suggested regarding the configuration. First, we want to **'Configure a device'.** (→ First Steps → Configure a device)



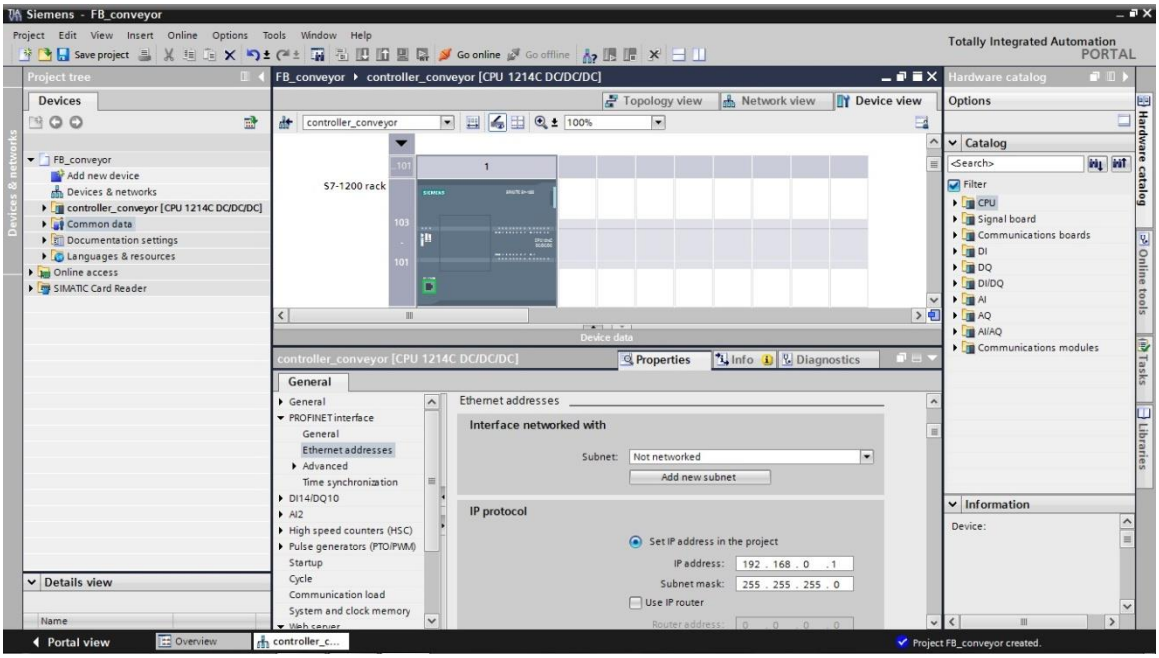
4. Next, we **'Add new device'** with the **'Device name conveyor control'**. To this end, we select from the catalog the **'CPU1214C'** with the matching order number. (→ Add new device → conveyor control → CPU1214C → 6ES7 ……. → Add)



5. Now, the software automatically switches to the project view with the opened hardware configuration. Here, additional modules from the hardware catalog (to the right) can be added and in the **'Device overview'**, the addresses of the inputs and outputs can be set. The integrated inputs of the CPU have the addresses %I0.0 to %I1.5 and the integrated outputs have the addresses %Q0.0 to %Q1.1 (→ Device overview → DI14/DO10 → 0…1)



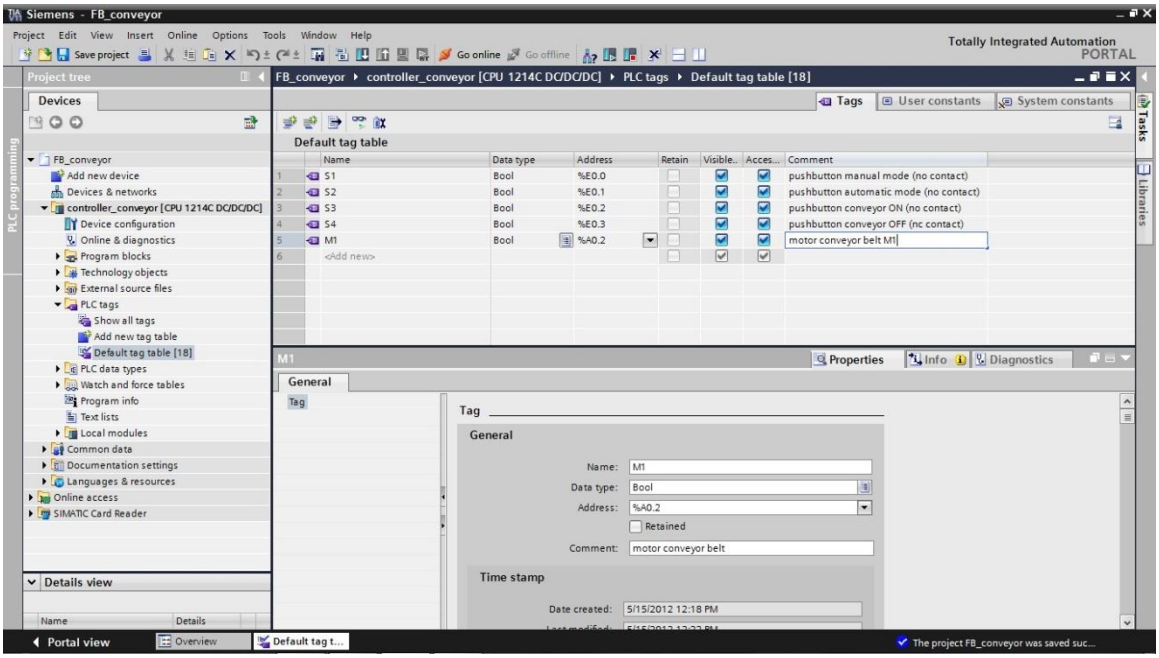
6. So that the software later accesses the correct CPU, its IP address and the subnet mask have to be set.
(→ Properties → General → PROFINET interface → IP address: 192.168.0.1 → Subnet mask: 255.255.255.0)



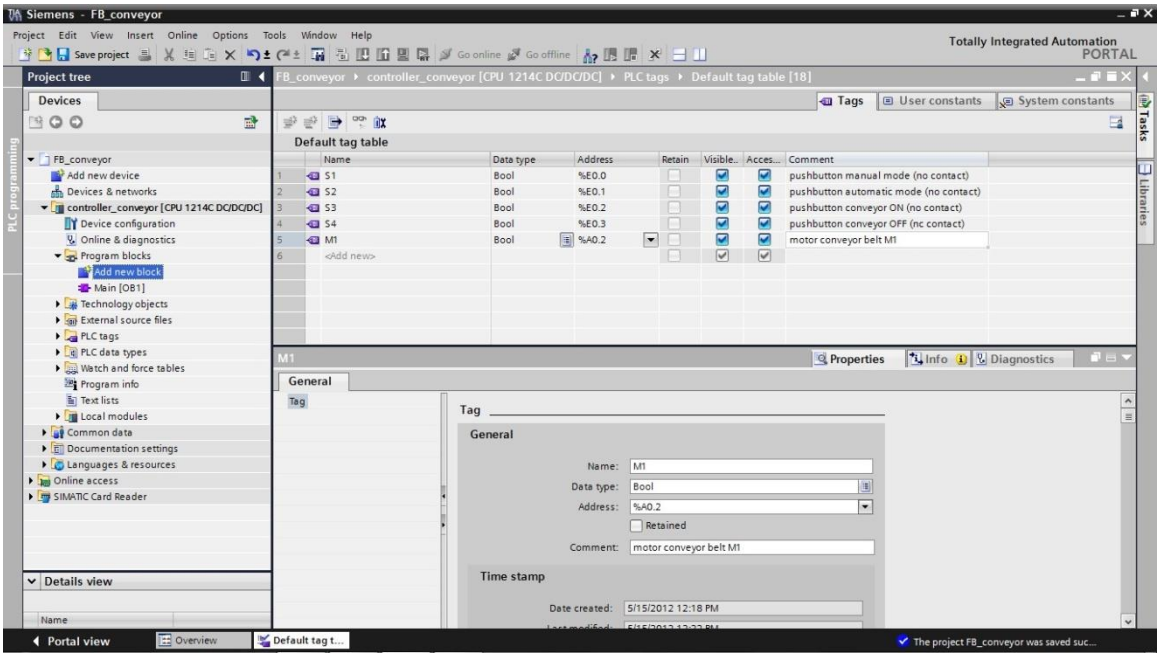
7. Since modern programming is not carried out with absolute addresses but with variables, the **global PLC tags** have to be specified here.

These global PLC variables are descriptive names with comments for those inputs and outputs that are used in the program. Later, during programming, this name is used to access the global PLC tags.
These global tags can be used in the entire program, in all blocks.

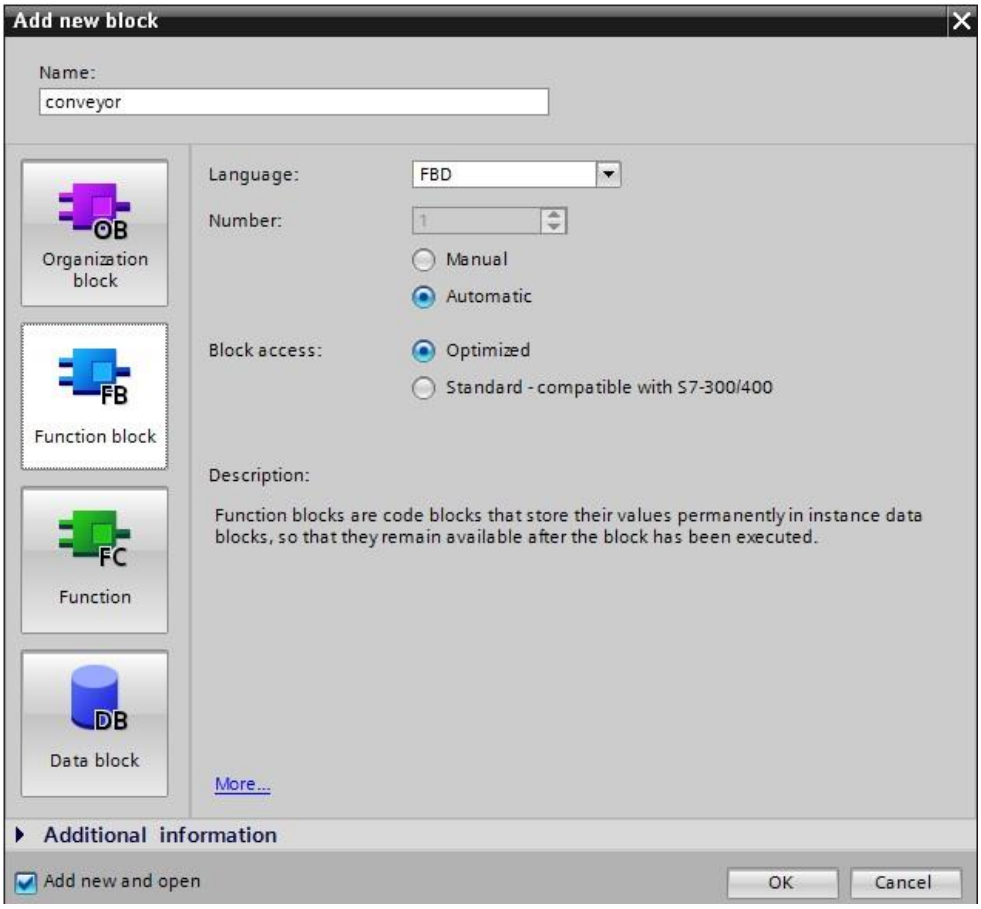
To this end, select in Project Navigation the **'controller_conveyorCPU1214C DC/DC/DC]'** and then **'PLC tags'**. Open the table **'PLC tags'** with a double click and enter the names for the inputs and outputs as shown below (→ controller_conveyor[CPU1214C DC/DC/DC]' → PLC tags→ PLC tags)



8. To generate the function block FB1, first select **'controller_conveyor[CPU1214C DC/DC/DC]'** in project navigation, and then **'Program blocks'**. Now, double click on **'Add new block'** (→ controller_conveyor[CPU1214C DC/DC/DC]' → Program blocks → Add new block)



9. In the selection, select **'Function block (FB)'** and assign the name **'conveyor'**. As programming language, we specify function block diagram **'FBD'**. Enumeration is automatic. Since this FB1 is called later with the symbolic name anyhow, this number is no longer that important. Accept the input with **'OK'.** (→ Function block (FB1) → conveyor → FBD → OK)



10. The block **'conveyor[FB1]'** will be opened automatically. But before we can write the program, we have to declare the block's interface.
When the interface is declared, the local variables -known only in this block- are specified.

The variables are divided into two groups:

- Block parameters that generate the interface of the block for the call in the program.

Type	Name	Function	Available in
Input parameters	Input	Parameters whose values the block reads	Functions, function blocks and some types of organization blocks
Output parameters	Output	Parameters whose values the block writes	Functions and function blocks
In/out parameters	InOut	Parameters whose value the block reads when called, and after processing writes to the same parameter	Functions and function blocks

- Local data that is used for storing intermediate results

Type	Name	Function	Available in
Temporary local data	Temp	Variables that are used for storing temporary intermediate results. Temporary data is retained for one cycle only.	Functions, function blocks and organization blocks
Static local data	Static	Variables that are used for storing static intermediate results in instance data blocks. Static data is retained -also over several cycles-.until it is written anew.	Function blocks

11. To declare local variables, the following variables are needed for our example.

Input:

manual	Here, the signal for selecting the operating mode Manual is entered
automatic	Here, the signal for selecting the operating mode Automatic is entered
on	Here, the start signal is entered
off	Here, the stop signal is entered

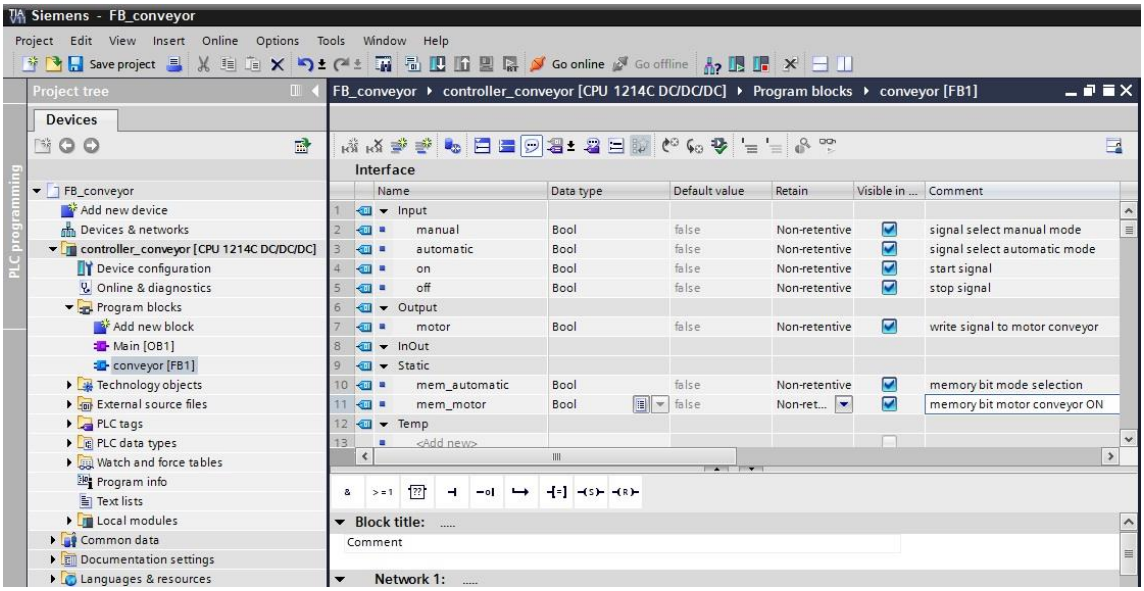
Output:

motor	Here, the output signal for the output conveyor motor is written

Static (exists only in the function blocks FB):

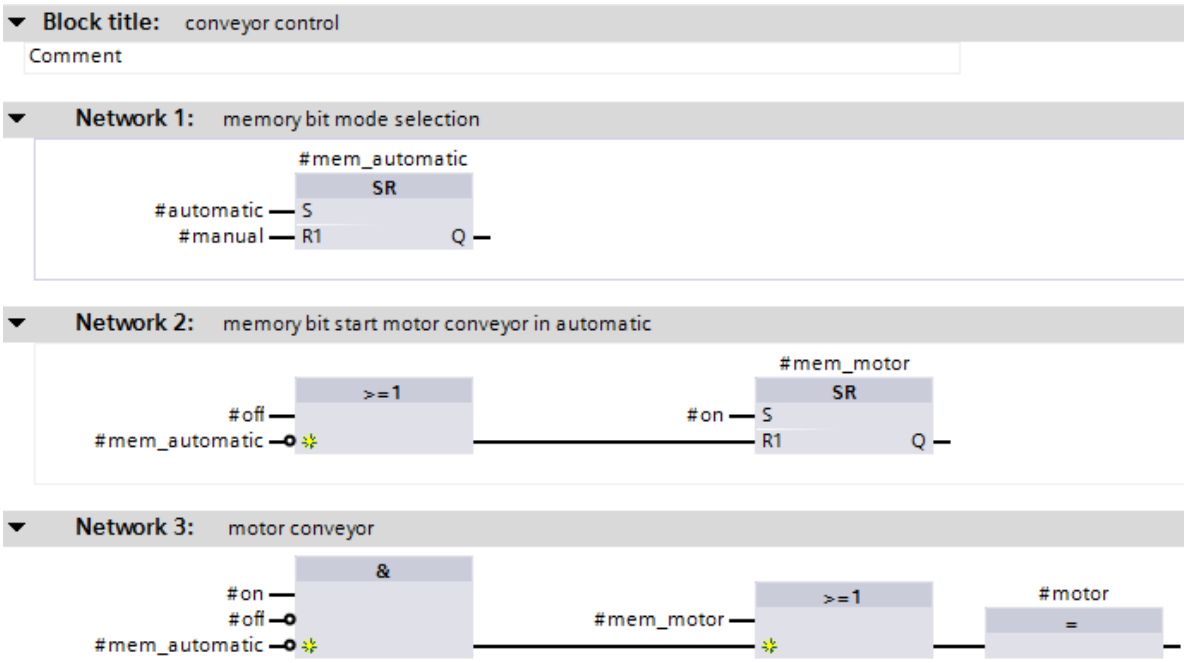
memory_automatic	Here, the preselected operating mode is stored
memory_motor	Here, we store when the motor was started in the Automatic mode

All variables are of the type 'Bool'; that means binary variables that only can have the status '0' (false) or '1' (true).
In this example, it is important to note that the status of the two variables 'memory_automatic' and 'memory_motor' has to be stored over a longer period of time. For that reason, the variable type **'Static'** has to be used here. This variable type in turn exists only in a function block FB.
For the sake of clarity, all local variables should also be provided with a sufficient comment.

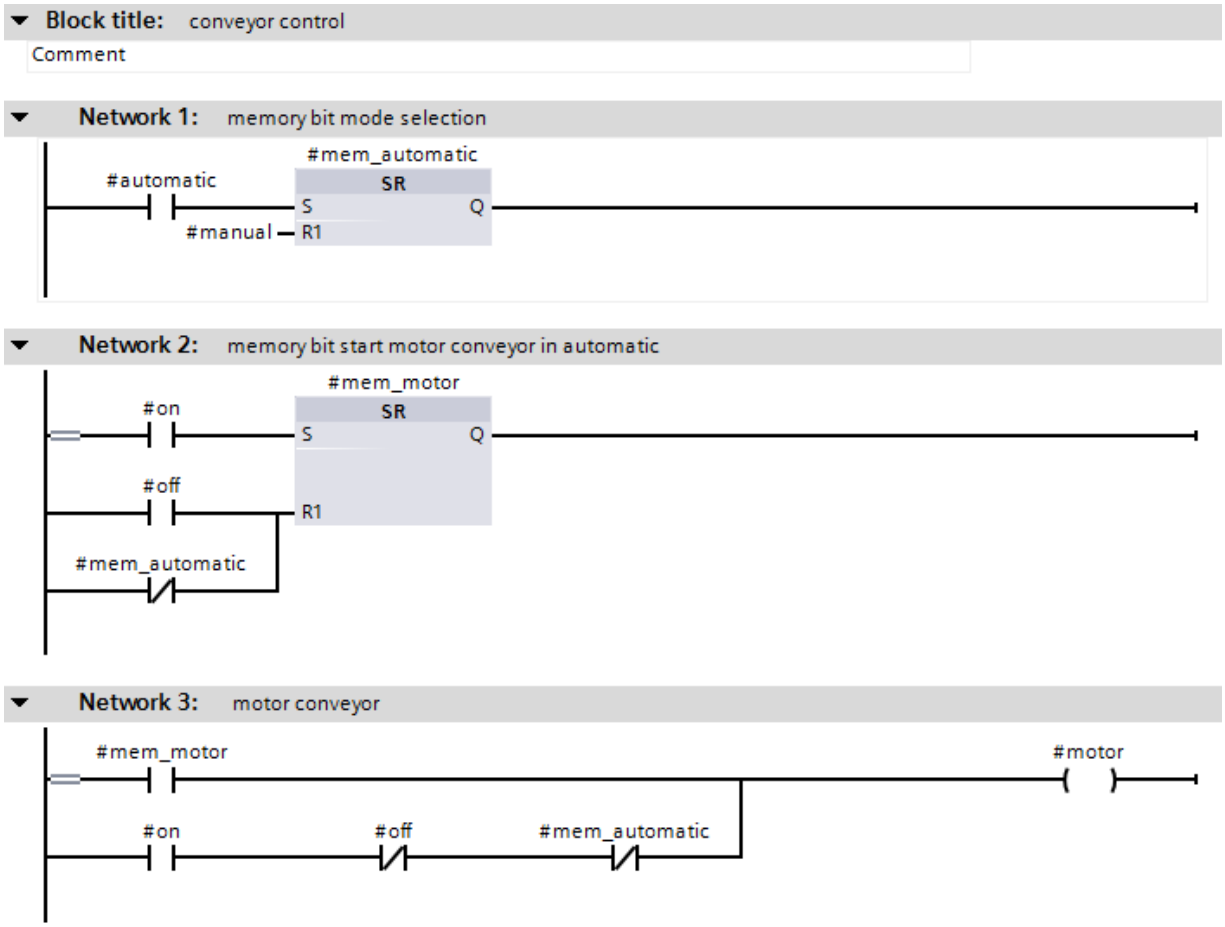


12. After the local variables have been declared, the program can now be entered by using the variable names (variables are identified with the symbol '#'). For the example in FBD, it could look like this:

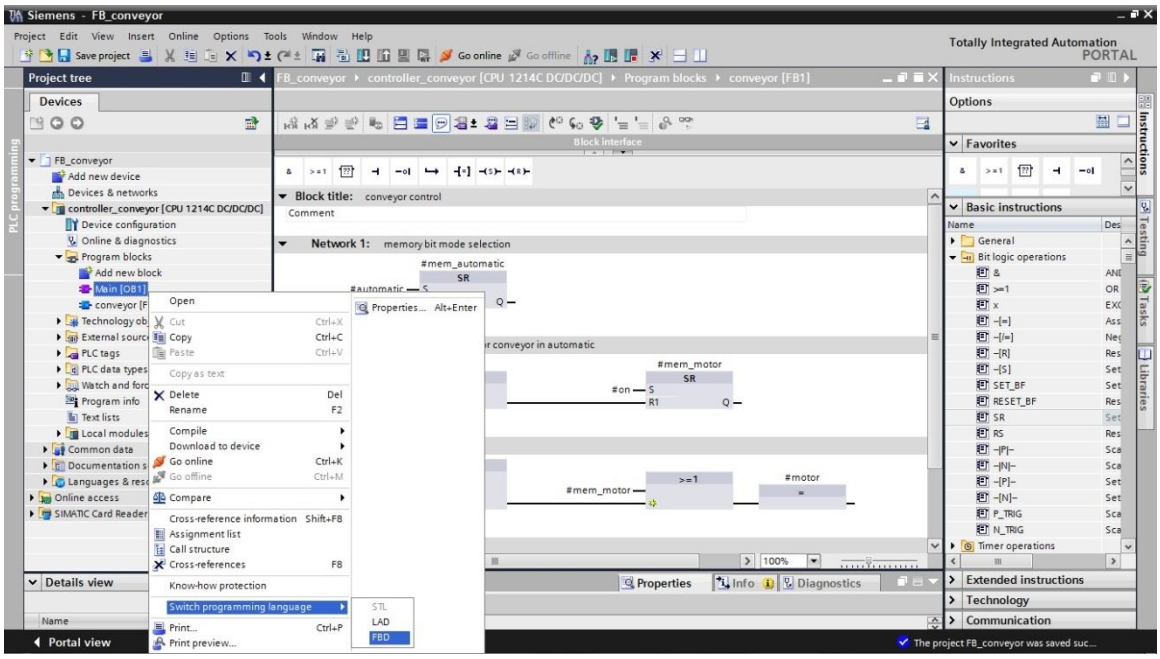
Program in function block diagram (FBD):



Program in ladder diagram (LAD):

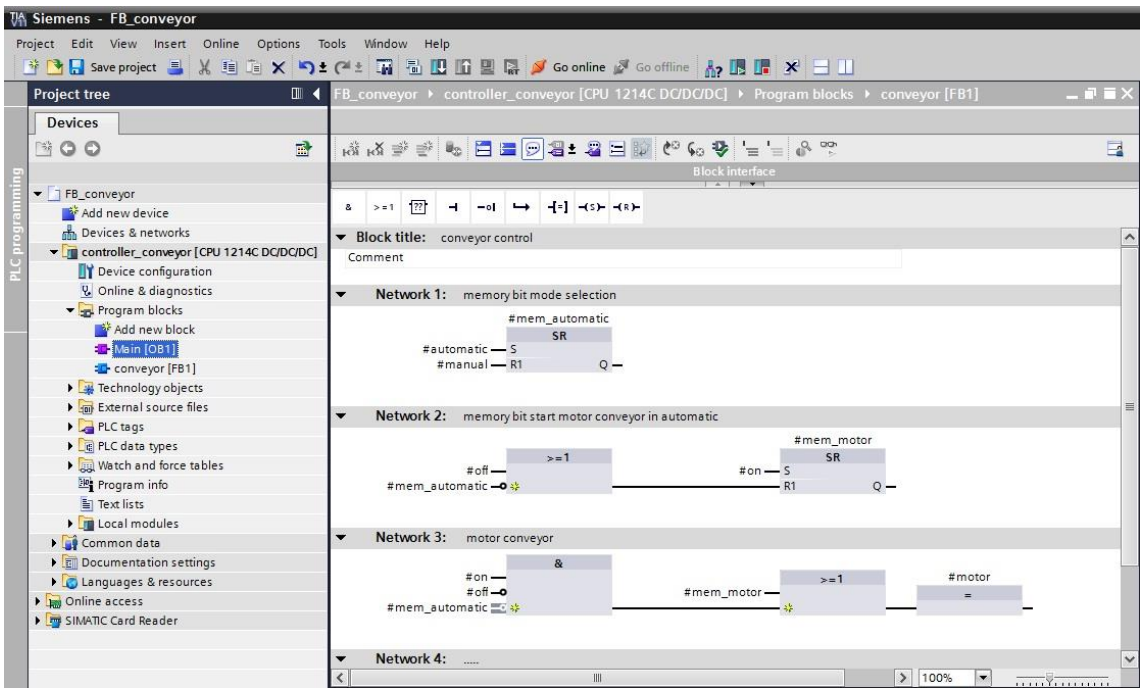


13. Next, right-click on the block **'Main[OB1]'**.
Then, under **'Switch programming language'**, select the function block diagram **'FBD'**.

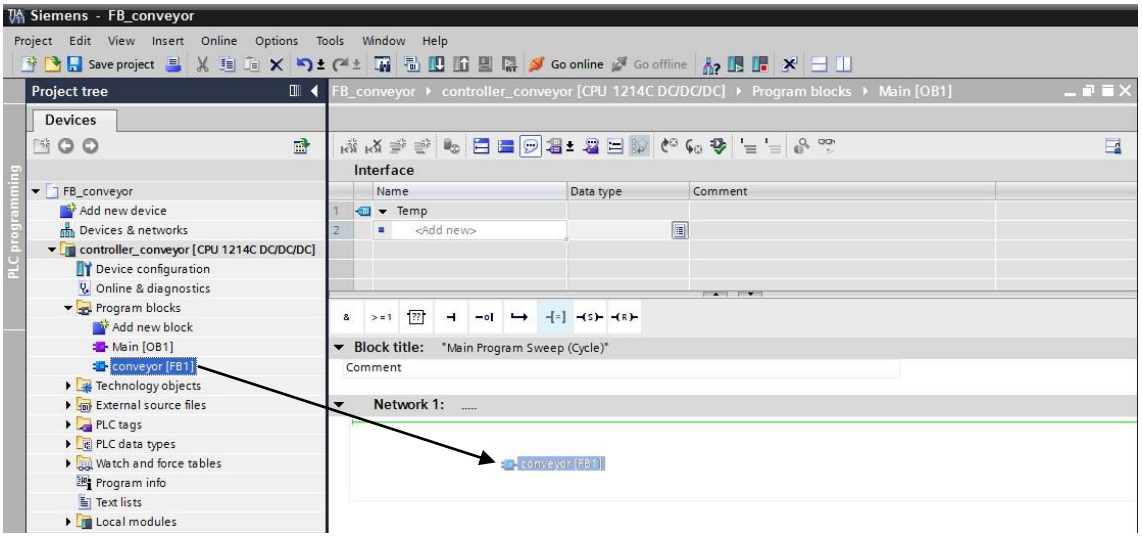


14. Now, the block “conveyor” has to be called from the program block Main[OB1]. Otherwise, the block would not be processed.

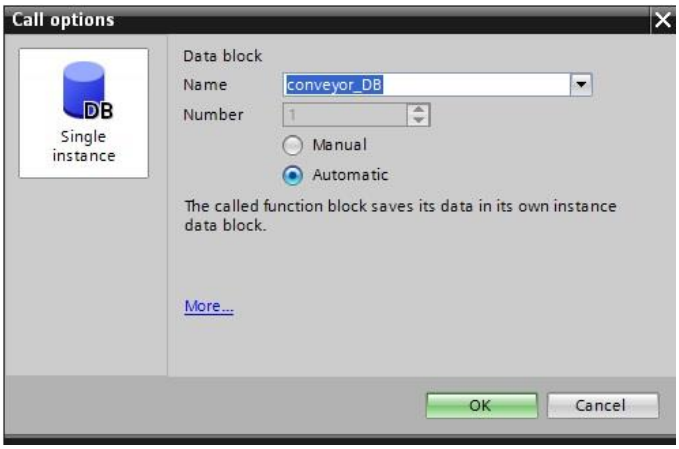
A double click on **‘Main[OB1]’** opens this block. (→ Main[OB1])



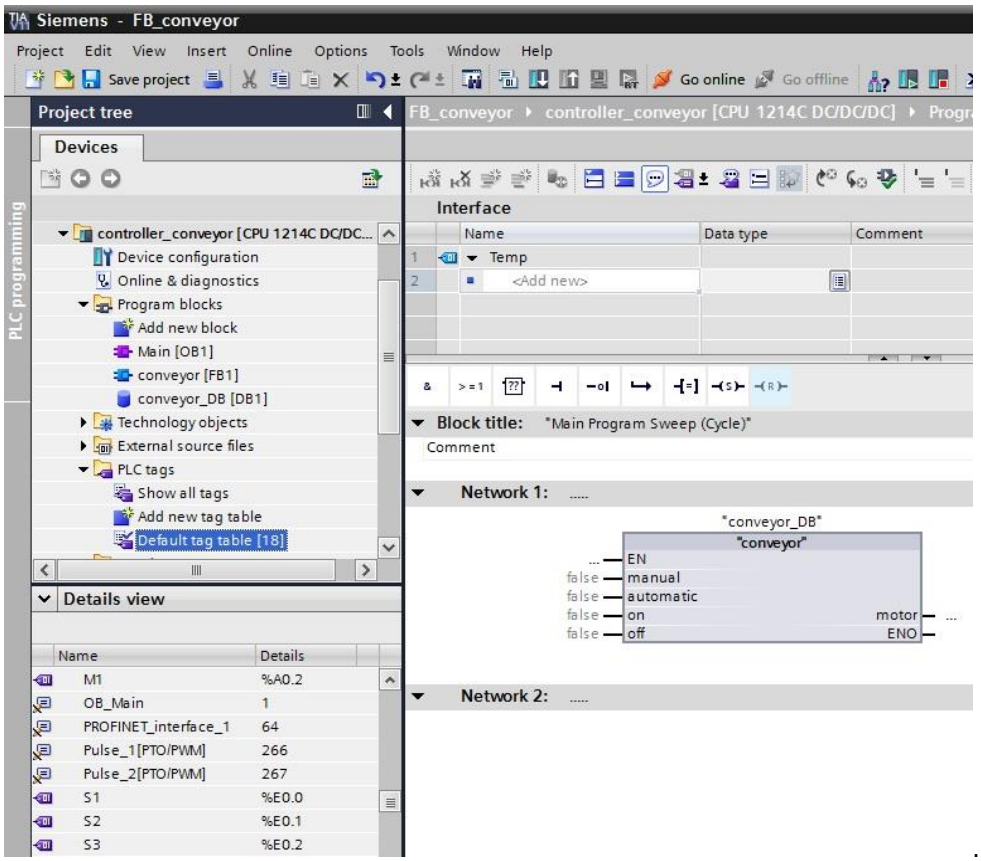
15. Now, you can drag the block **"conveyor[FB1]“** with Drag&Drop to Network 1 of the block Main[OB1]. (→ conveyor[FB1])



16. Since we are dealing with a function block, it has to be provided with memory. In SIMATIC S7-1200, data blocks are provided as memory. Such an assigned data block is called **Instance Data block**.
Here, it is to be specified and generated **'automatically'.** (→ Automatic→ OK)

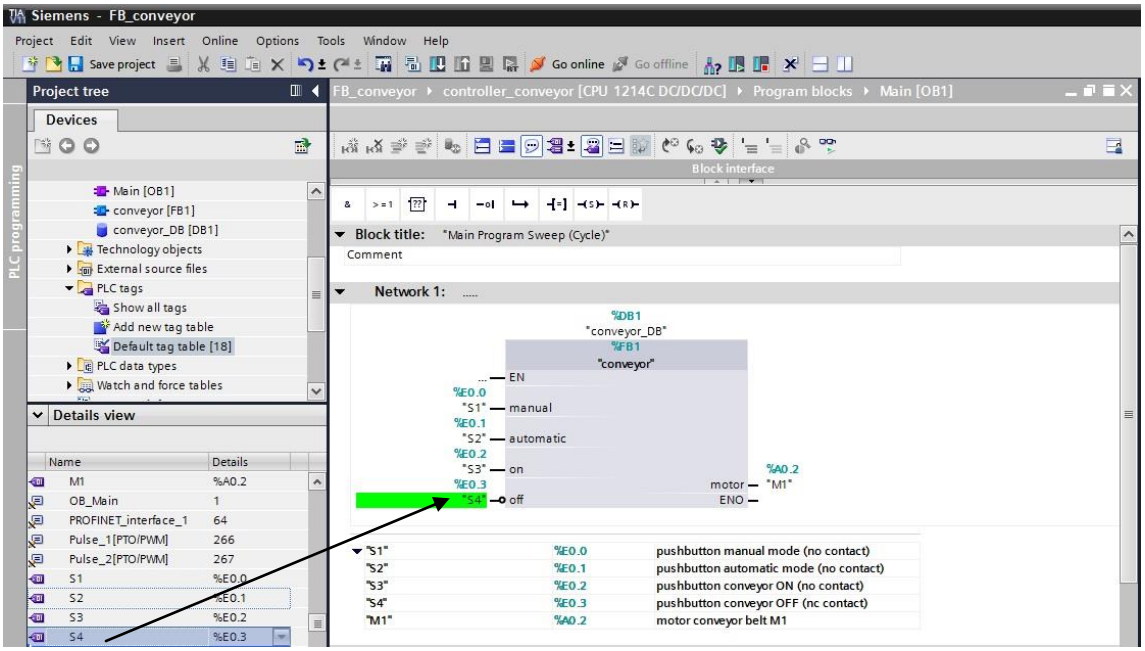


Highlight the default tag table.



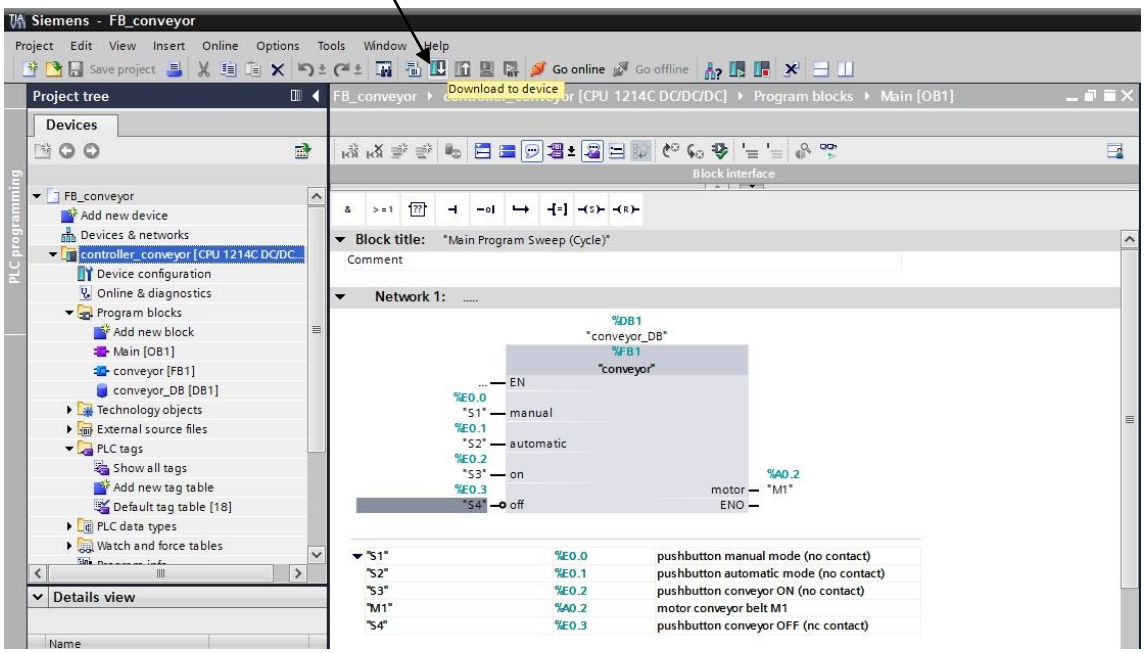
17. In OB1, we now connect the input variables and the output variable with the PLC tags shown here.
To this end, just drag the PLC tags to the block variables.

By clicking on Save project, the project is saved. (→ "S1“ → "S2“ → "S3“ → "S4“ → "M1“ → Save project)

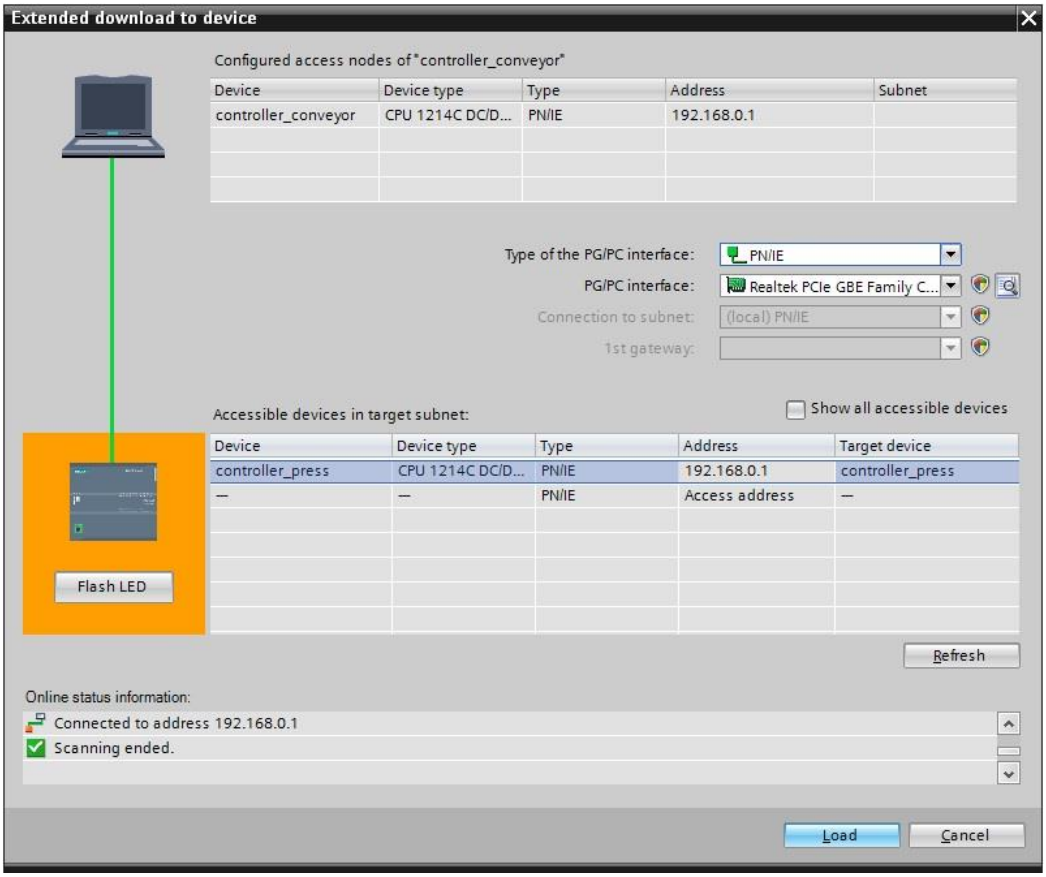


Important!
The Off button S4 is a break contact (NC) and has to be negated at the block during wiring; i.e., the Off function in the block is pending when the Off button S4 is operated and thus no signal is pending at terminal %I0.3.

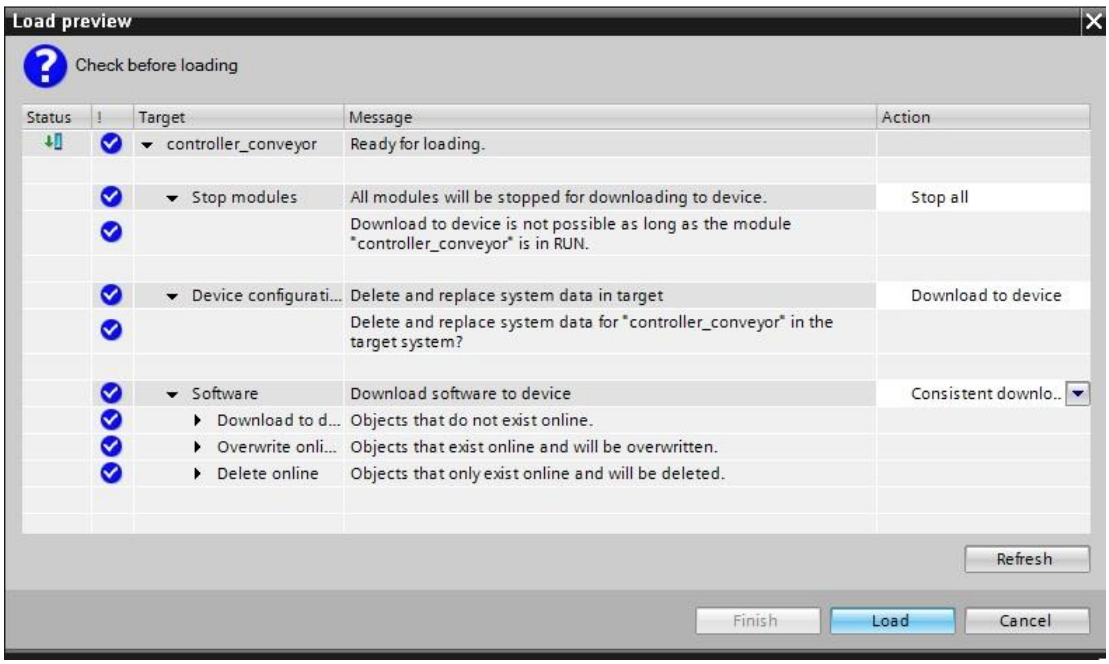
18. To load your entire program to the CPU, first highlight the folder **'controller conveyor'** and then click on the symbol Load to device. (→ controller_conveyor →)



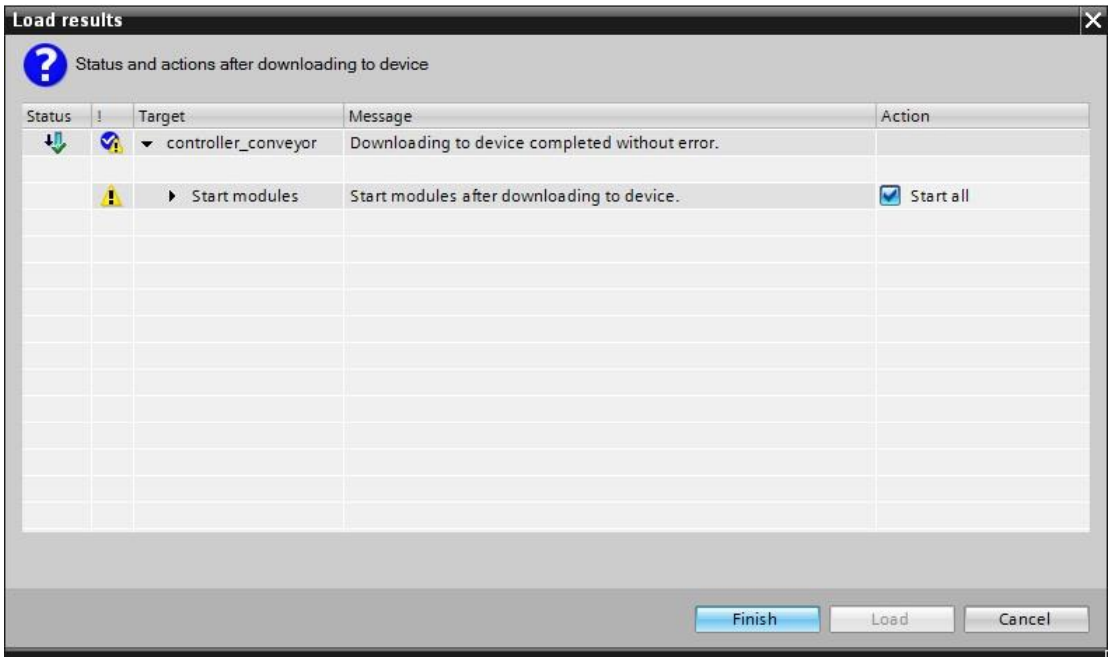
19. If you omitted specifying the PG/PC interface beforehand, a window is displayed where you can still do this.
(→ PG/PC interface for loading → Load)



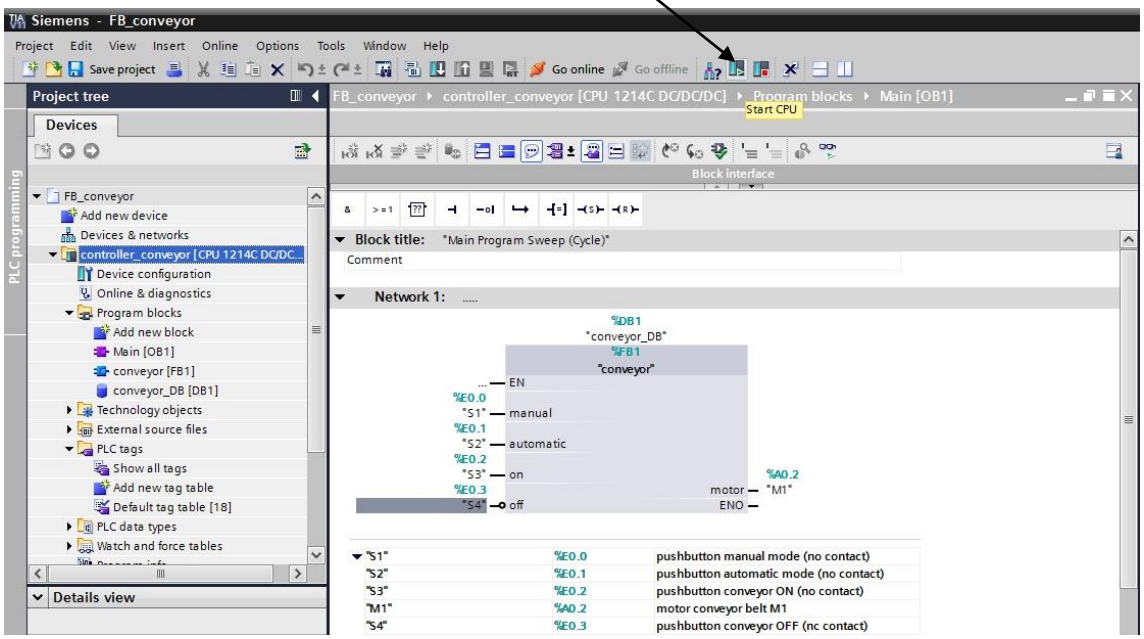
20. Click on 'Load' once more. In a window, the status is indicated during loading. (→ Load)



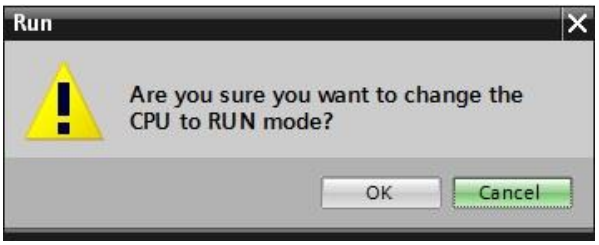
21. If loading was successful, it is shown in a window. Click on **'Finish'.** (→ Finish)



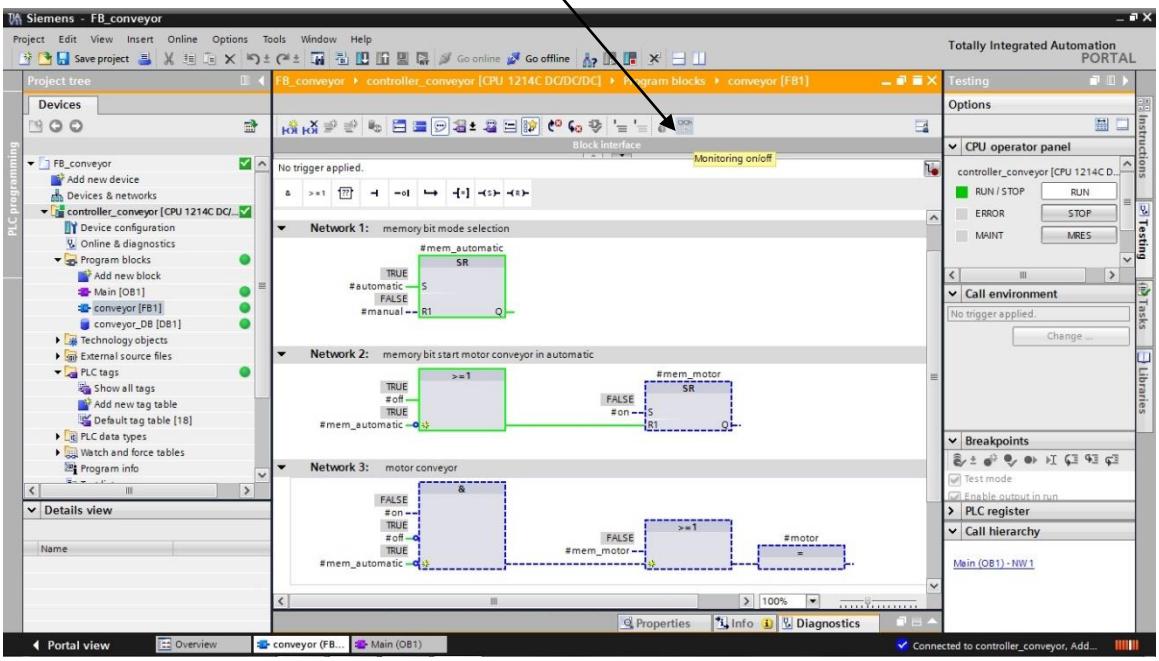
22. Now, start the CPU by clicking on the symbol  .



23. With **'OK'**, confirm the question whether you actually want to start the CPU. (→ OK)

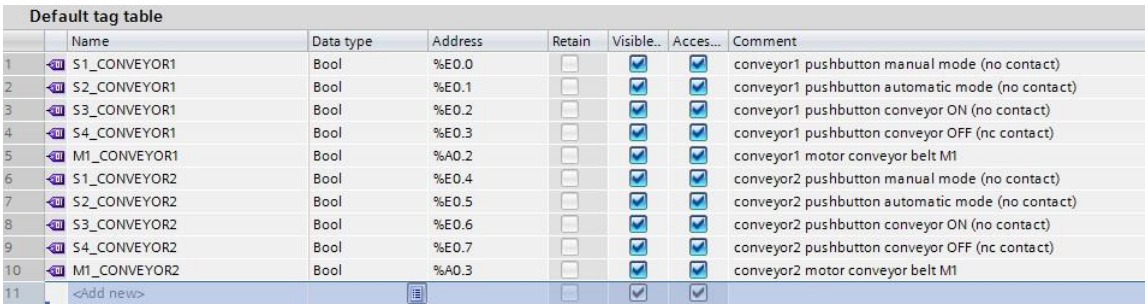


24. By clicking on the symbol  Monitoring On/Off, you can, during the program test, observe the status of the input and output variables at the block “conveyor“, but also the program execution in the block “conveyor“. (→ conveyor[FB1] → )



25. Since our block “conveyor“ was generated according to the rules for standard blocks (no use of global variables within the block!!!!!), it can now be used and called any number of times.

Below, an expanded PLC tag table is shown, with the inputs and outputs for two conveyors.

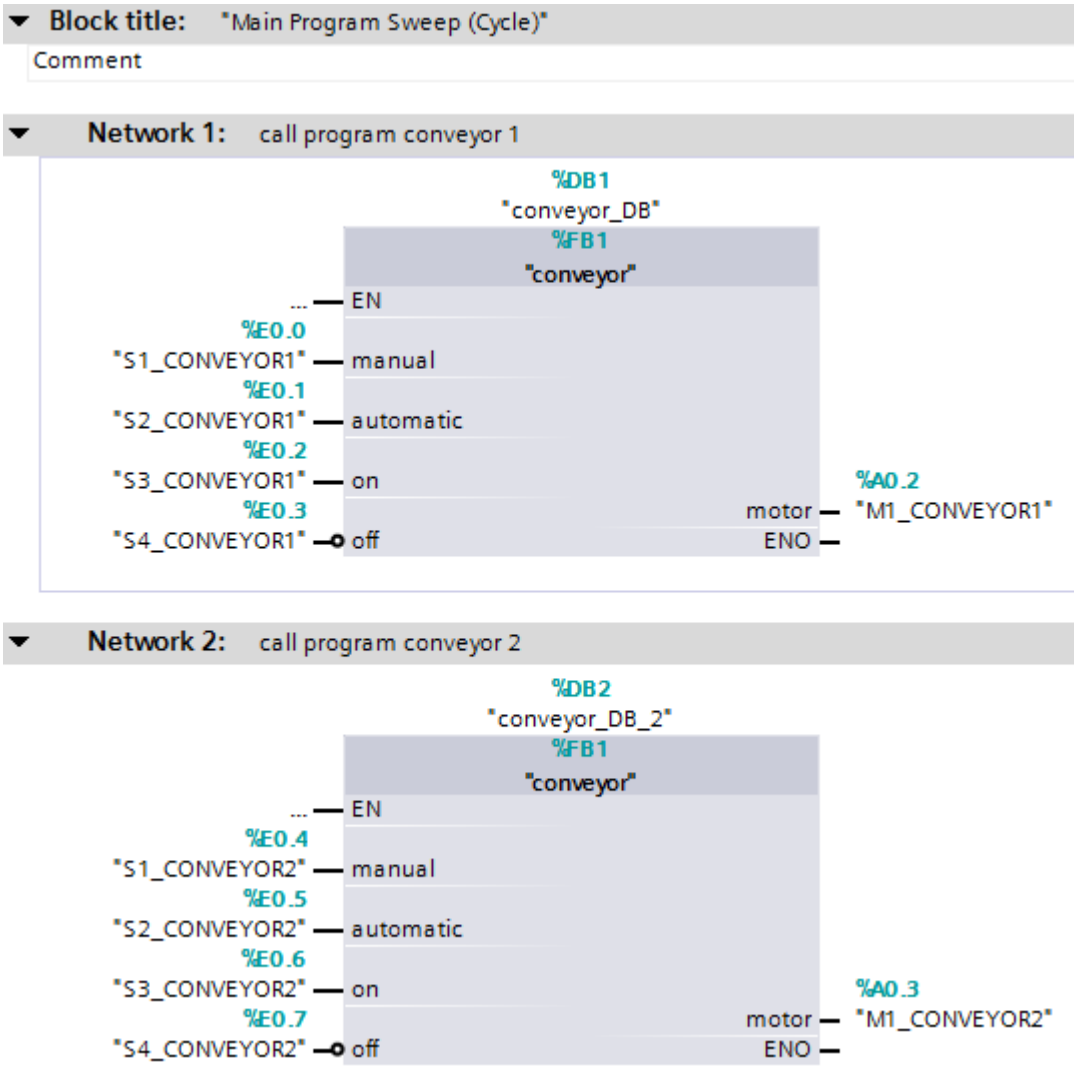


Default tag table

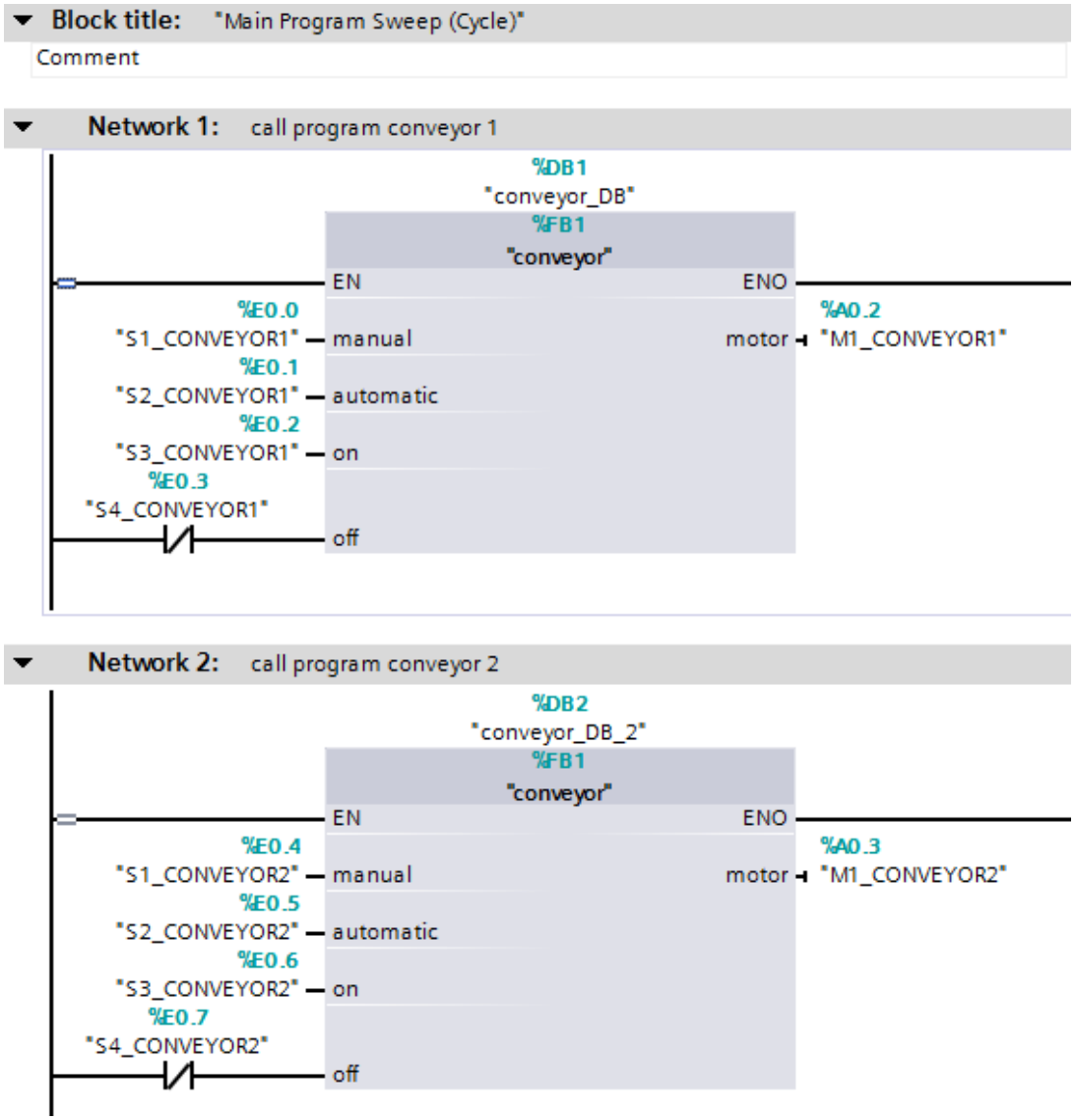
	Name	Data type	Address	Retain	Visible..	Acces...	Comment
1	S1_CONVEYOR1	Bool	%E0.0	☐	☑	☑	conveyor1 pushbutton manual mode (no contact)
2	S2_CONVEYOR1	Bool	%E0.1	☐	☑	☑	conveyor1 pushbutton automatic mode (no contact)
3	S3_CONVEYOR1	Bool	%E0.2	☐	☑	☑	conveyor1 pushbutton conveyor ON (no contact)
4	S4_CONVEYOR1	Bool	%E0.3	☐	☑	☑	conveyor1 pushbutton conveyor OFF (nc contact)
5	M1_CONVEYOR1	Bool	%A0.2	☐	☑	☑	conveyor1 motor conveyor belt M1
6	S1_CONVEYOR2	Bool	%E0.4	☐	☑	☑	conveyor2 pushbutton manual mode (no contact)
7	S2_CONVEYOR2	Bool	%E0.5	☐	☑	☑	conveyor2 pushbutton automatic mode (no contact)
8	S3_CONVEYOR2	Bool	%E0.6	☐	☑	☑	conveyor2 pushbutton conveyor ON (no contact)
9	S4_CONVEYOR2	Bool	%E0.7	☐	☑	☑	conveyor2 pushbutton conveyor OFF (nc contact)
10	M1_CONVEYOR2	Bool	%A0.3	☐	☑	☑	conveyor2 motor conveyor belt M1
11	<Add new>			☐	☑	☑	

26. Now, the block **"conveyor“** can also be called twice in OB1, with different wiring respectively. For each call, another instance data block is specified.

Program in function block diagram (FBD):



Program in ladder diagram (LAD):



Two conveyors separated from each other can now be controlled by means of the same conveyor block. Only another instance data block has to be assigned for each call.