Contents

7

7. Program Blocks and Program Editor

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

... be familiar with the different S7 block types

... be familiar with the principle of "structured programming"

… be able to add/create new blocks (functions)

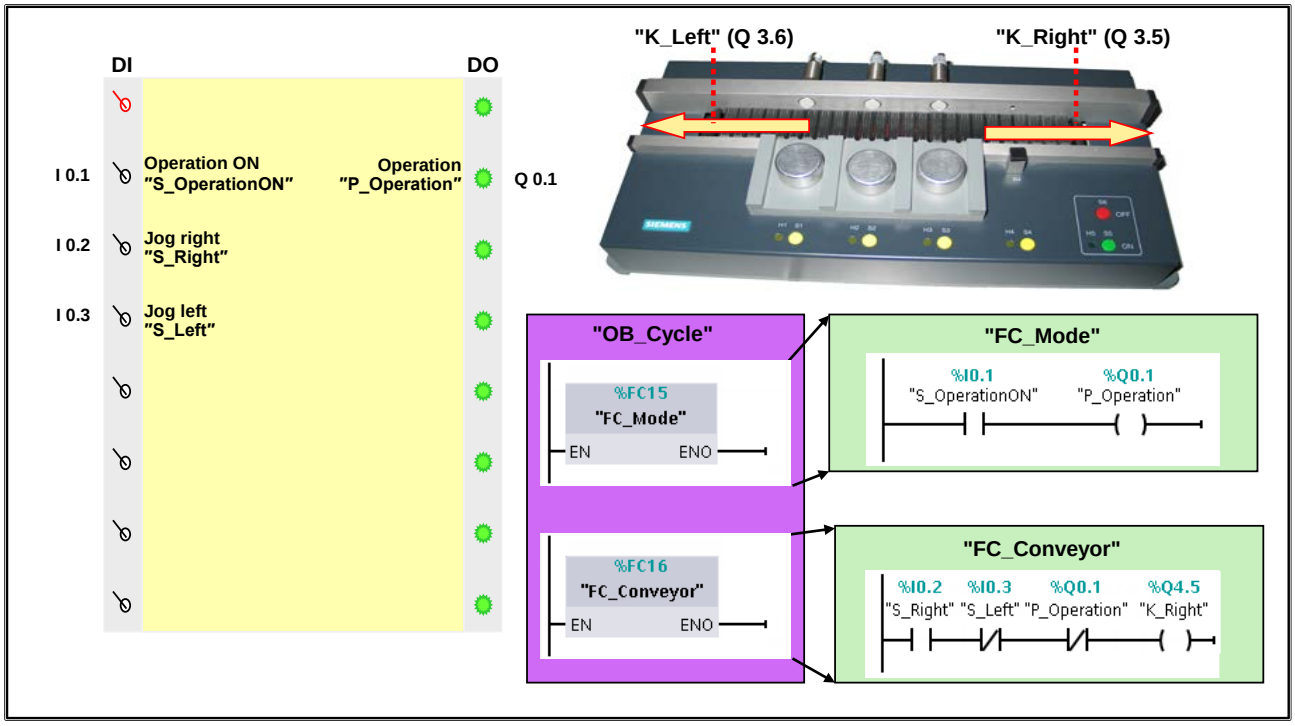
... be able to set block properties

… be able to program a block call

… know what happens when a block is uploaded

… be familiar with the memory concept of the S7-1500

7.1. Plant Description: The Conveyor Model as Distribution Conveyor with Mode Selection



Conveyor Model as a Distribution Conveyor

The Plant is switched ON/OFF and in Automatic or Manual mode using the switch "S_OperationON".

In Manual mode ("P_Operation" = FALSE), the distribution conveyor can be moved to the right and left using the switches "S_Right" and "S_Left". If both switches are activated simultaneously, then the conveyor must not move (Lock-out!).

7.2. Overview of the Blocks in STEP 7

Blocks/Instructions	Properties
Organization block (OB)	- User interface - Graduated priorities (0 to 27) - Specific start information in temporary variables
Function (FC)	- Parameter-assignable (parameters must be assigned with the call) - Without (dedicated) memory (only temporary variables)
Function block (FB)	- Parameter-assignable (parameters can be assigned with the call) - With (dedicated) memory (static variables)
Data block (DB)	- Structured local data storage (Instance DB) (memory of an FB) - Structured global data storage (Global DB) (valid in the entire program)
System instruction without instance	- Instruction without memory stored in the CPU's operating system and callable by the user
System instruction with instance	- Instruction without memory stored in the CPU's operating system and callable by the user (therefore requires an instance)

Code Blocks

The automation system provides various types of blocks in which the user program and the related data can be stored. Depending on the requirements of the process, the program can be structured in different blocks. You can use the entire operation set in all blocks (FB, FC and OB).

Organization Blocks (OBs)

Organization blocks (OBs) form the interface between the operating system and the user program.

Functions (FCs)

A function (FC) contains a partial functionality of the program. It is possible to program functions as parameter-assignable so that when the function is called it can be assigned parameters. As a result, functions are also suited for programming frequently recurring, complex partial functionalities such as calculations.

Function Blocks (FBs)

Basically, function blocks offer the same possibilities as functions. In addition, function blocks have their own memory area in the form of instance data blocks. As a result, function blocks are suited for programming frequently recurring, complex functionalities in which data has to be stored over several cycles (such as closed-loop control tasks).

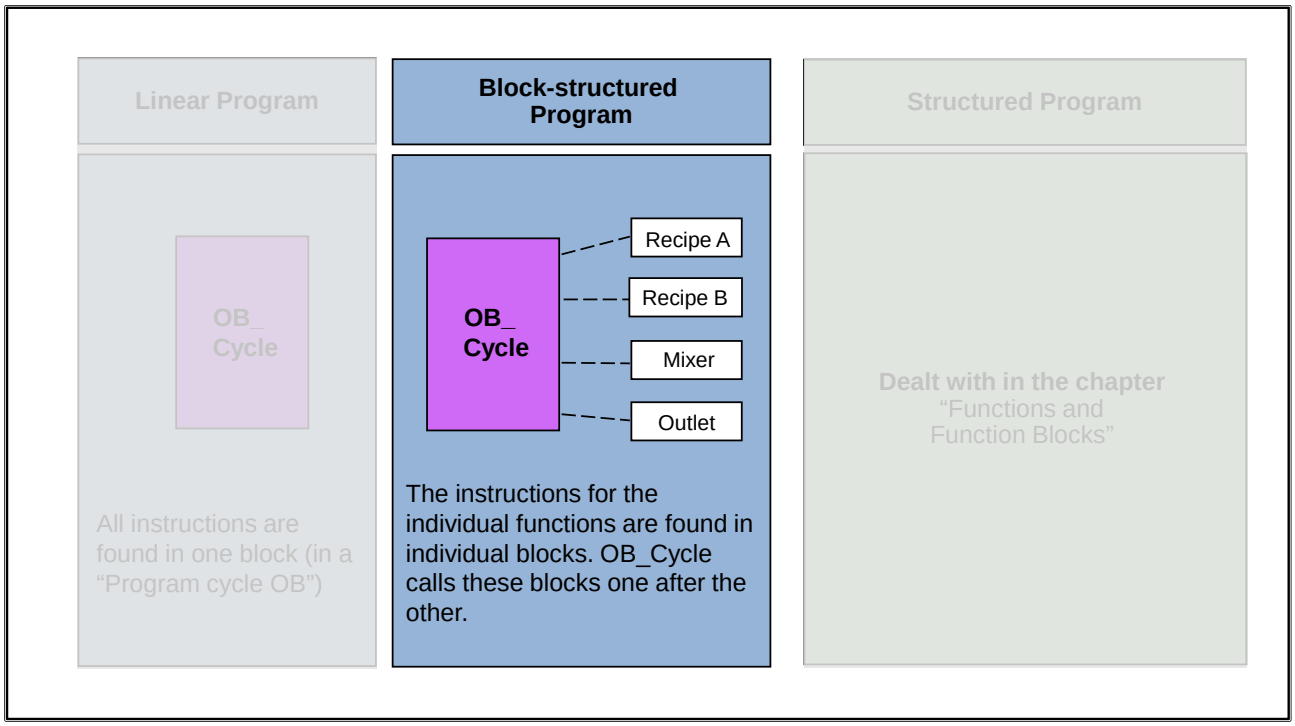
Data Blocks:

Data blocks are used to store data. There are global data blocks and instance data blocks which are used for the data backup of an FB call.

System Instructions:

System instructions are used to solve frequently required standard tasks. They are integrated in the CPU's operating system and therefore do not require any additional memory.

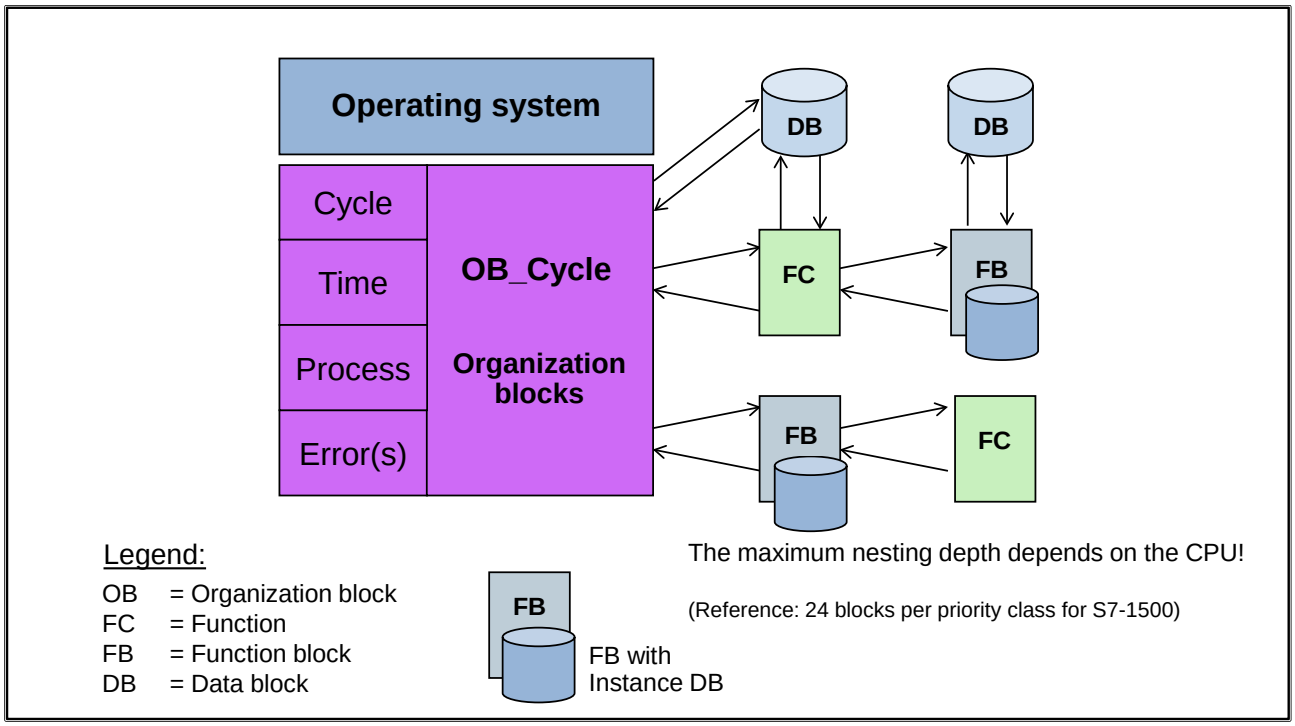
7.2.1. Block-structured Programming



Block-structured Program

The program is divided into blocks, whereby every block only contains the program for solving a partial task. Further structuring through networks is possible within a block. You can generate network templates for networks of the same type. Normally, a cyclically called Organization block contains instructions which call the other blocks in a defined sequence.

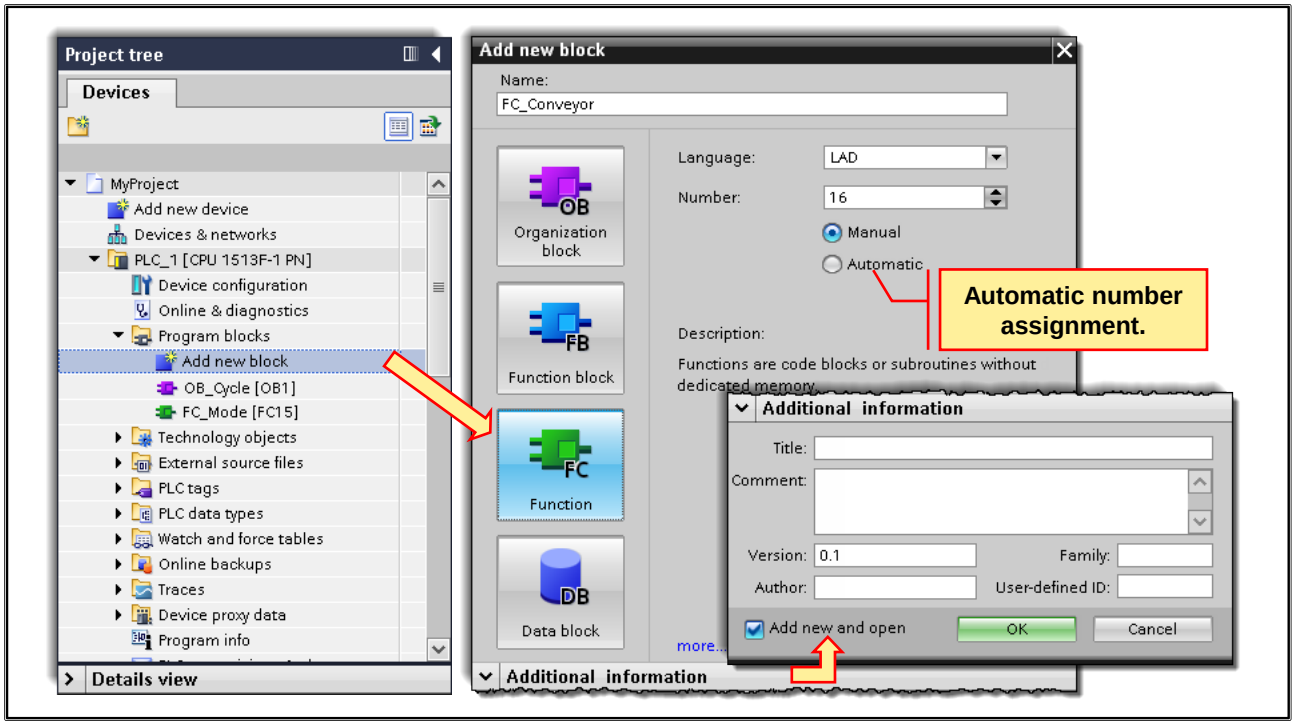
7.2.2. Program Sequence



Organization blocks are automatically called by the CPU's operating system according to certain call events. The Program cycle OBs are responsible for the cyclic program execution. So that blocks, such as FBs and FCs, can be processed, they have to be called by an OB. These blocks, in turn, can call further FBs or FCs.

Data blocks are used for data backup; for this there are two types. Firstly, global data blocks whose structure can be freely defined by the user. And second, the data blocks which are used to store the data of individual function blocks (instance data blocks). The structure of these data blocks depends on the interface of the respective function block. (see chapter "Functions and Function Blocks")

7.3. Adding a New Block

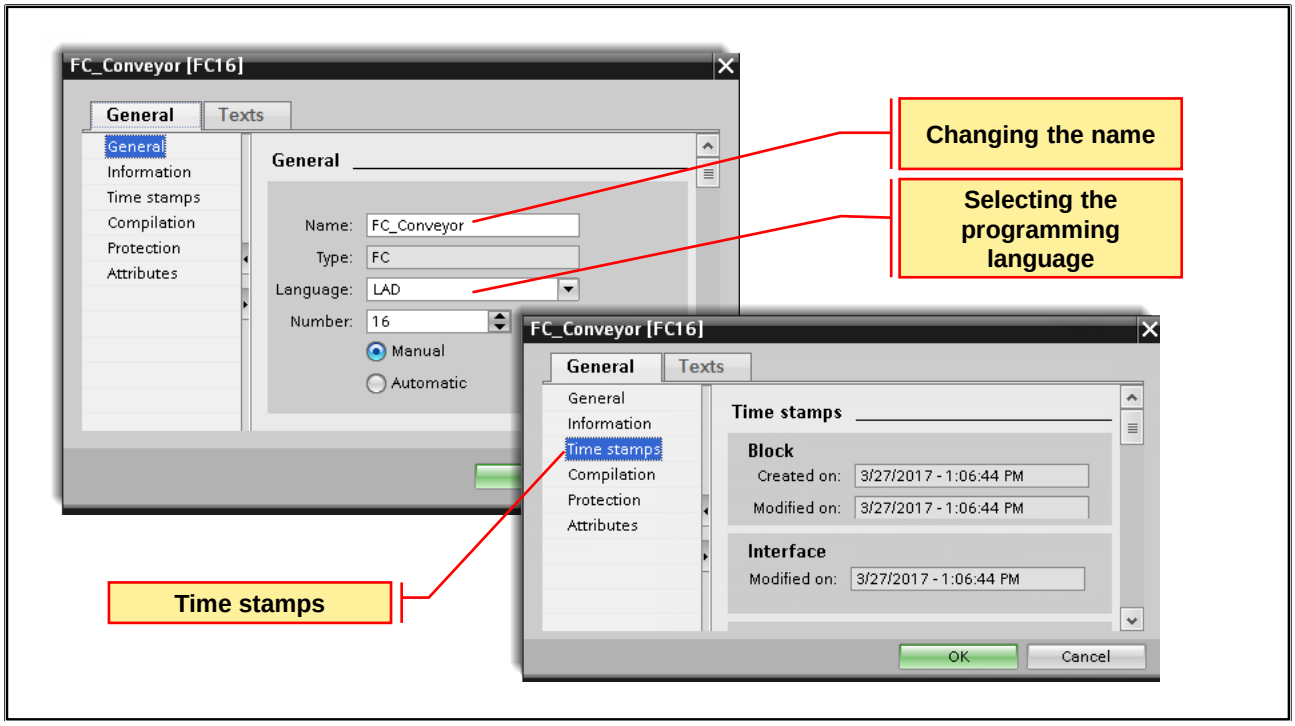


Inserting a Block

A new block is created as shown in the picture. When you create a block, the type of block (OB, FB, FC or DB), the programming language, the symbolic name and number, among other things, must be defined. The block numbers can also be assigned automatically or manually.

Under "Additional information", the block can be documented in more detail, among other things, with a Version number and an Author.

7.4. Block Properties: General, Time Stamps

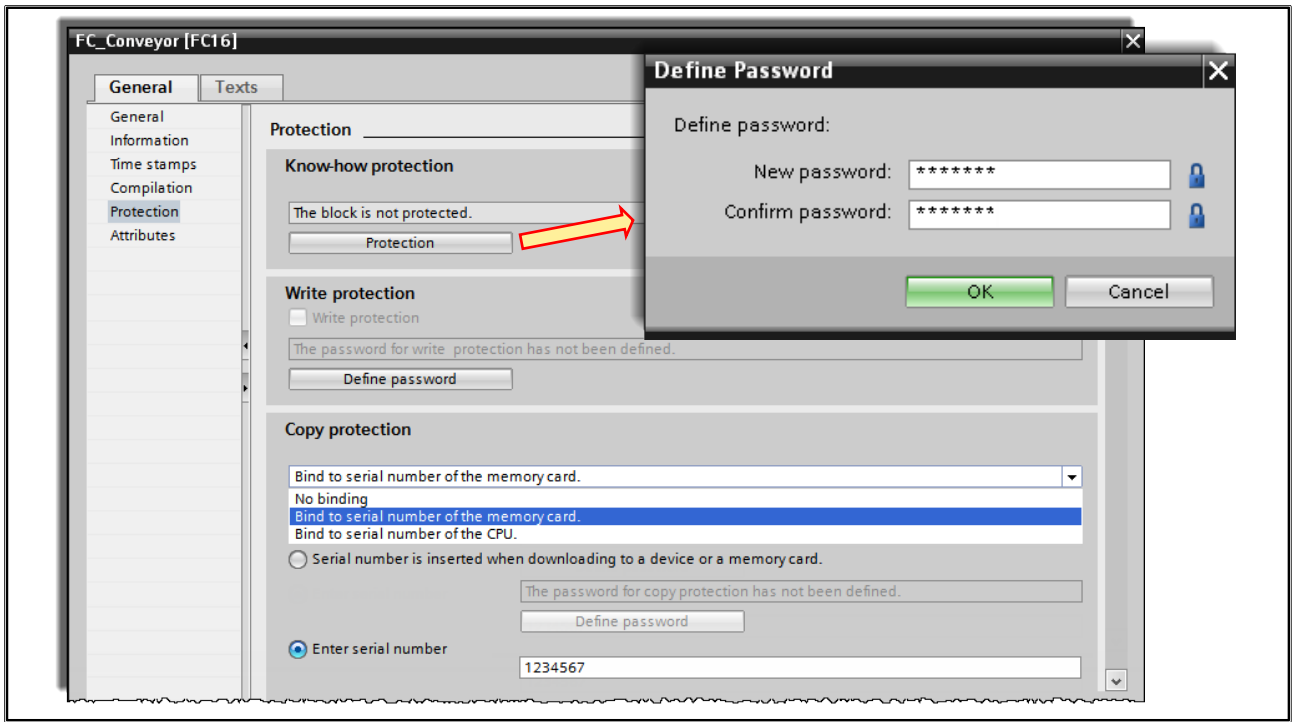


Properties General

Each block has certain properties that you can display and edit. These properties are used to:

- Identify the block
- Display the memory requirements and the compilation status of the block
- Display the time stamp
- Display the reference information
- Specify the access protection
- Display and change the programming language

7.4.1. Block Properties: Know-how Protection and Copy Protection



Know-how Protection

Blocks can be protected from unauthorized access with a password. With a know-how protected block, only the following data can be read:

- Parameters (Input, Output, InOut, Return)
- Block title
- Block comment
- Block properties
- Cross references with the information about which global tags are used

Just like unprotected blocks, know-how protected blocks can also be copied, called, downloaded into the CPU and deleted. The code of the block, however, is protected from unauthorized reading and changing.

Write protection

You can assign write protection for code blocks, which prevent unintentional changes to the block.

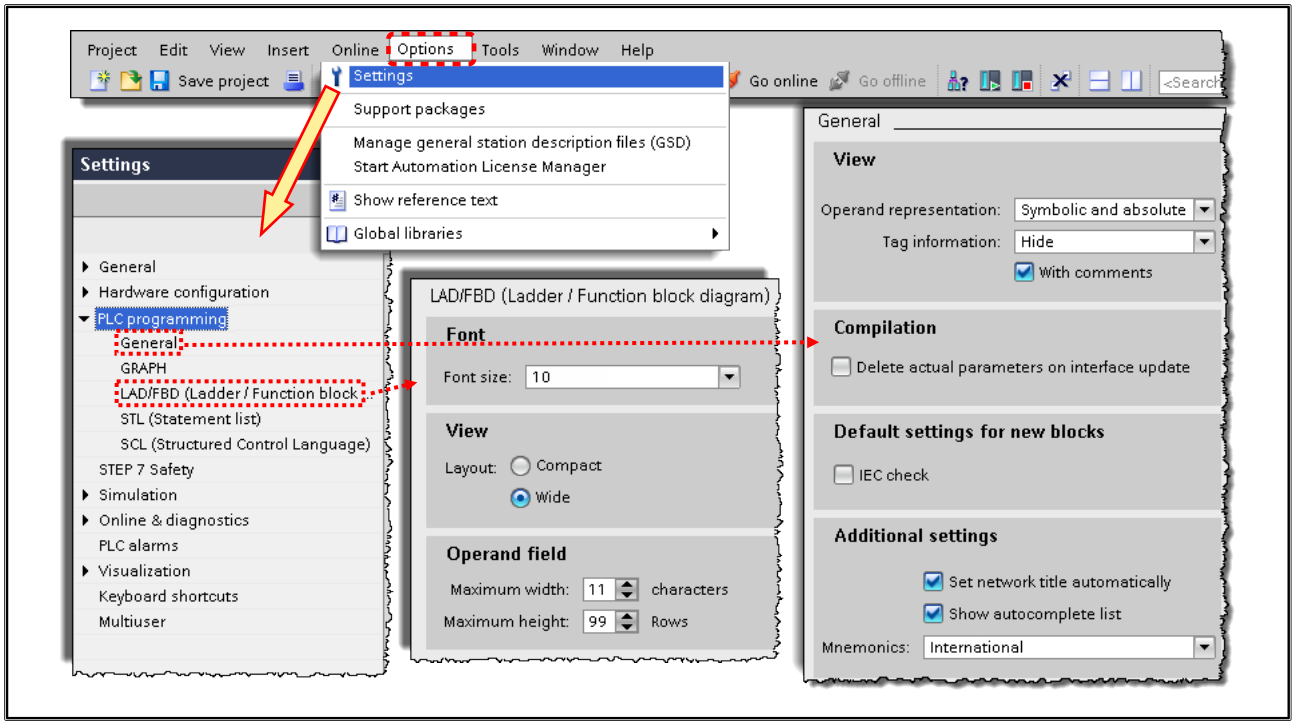
Copy Protection

In addition to know-how protection and write protection, you can also generate a copy protection in which you define with which memory card or on which CPU (each identified through a serial number) the block can be executed.

Caution!

If you forget the password, it is no longer possible to access the block.

7.5. Block Editor Settings

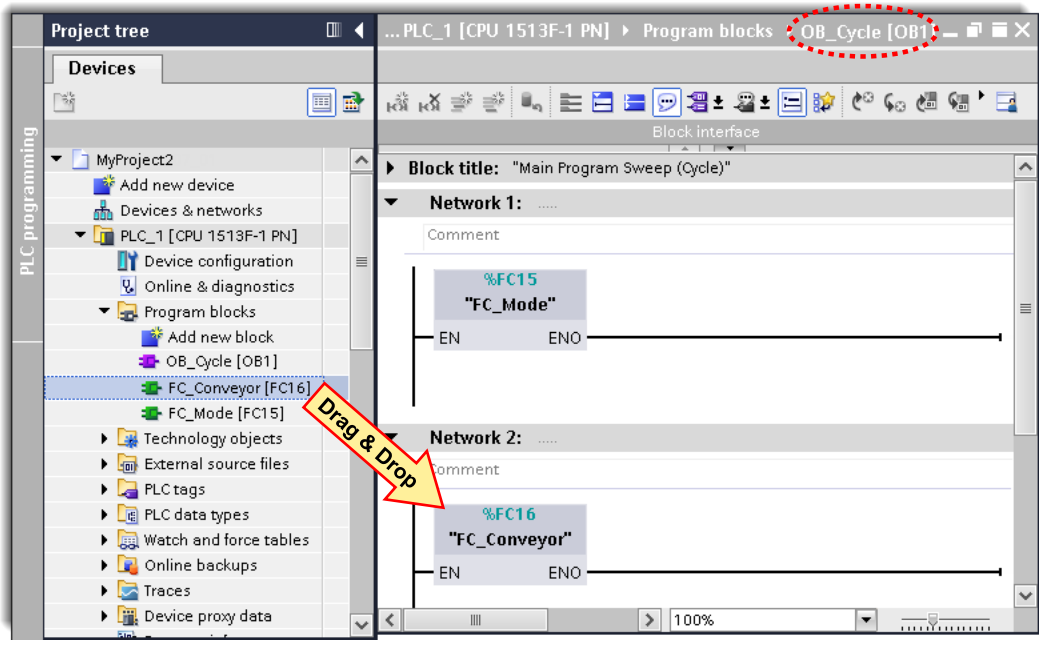


Block Editor Settings

With the settings, you merely define how a block is to be represented when it is opened. In the editor, you can make changes to the representation (such as showing and hiding comments) at any time.

- View
  Setting the Operand representation and Tag information with and without comments when the block is opened
- Compilation
  When "Delete actual parameters on interface update" is activated, the calls of parameterized blocks are also then automatically adjusted if, within the block, parameters which are already supplied with an actual parameter during the call, are deleted after the fact.
- IEC Check
  When IEC check is activated, the compatibility of operands is checked in accordance with IEC 61131.
- Mnemonics
  Setting the syntax for the programming language: German (e.g. E for Eingang (Input)) or International (e.g. I for Input)
- View - Layout
  This changes the vertical spacing between operands and other objects (for example, operands and contact). The change only becomes visible the next time a block is opened.
- Operand Field
  Setting the maximum width and height of function block diagram and ladder diagram symbols

7.6. Programming Block Calls

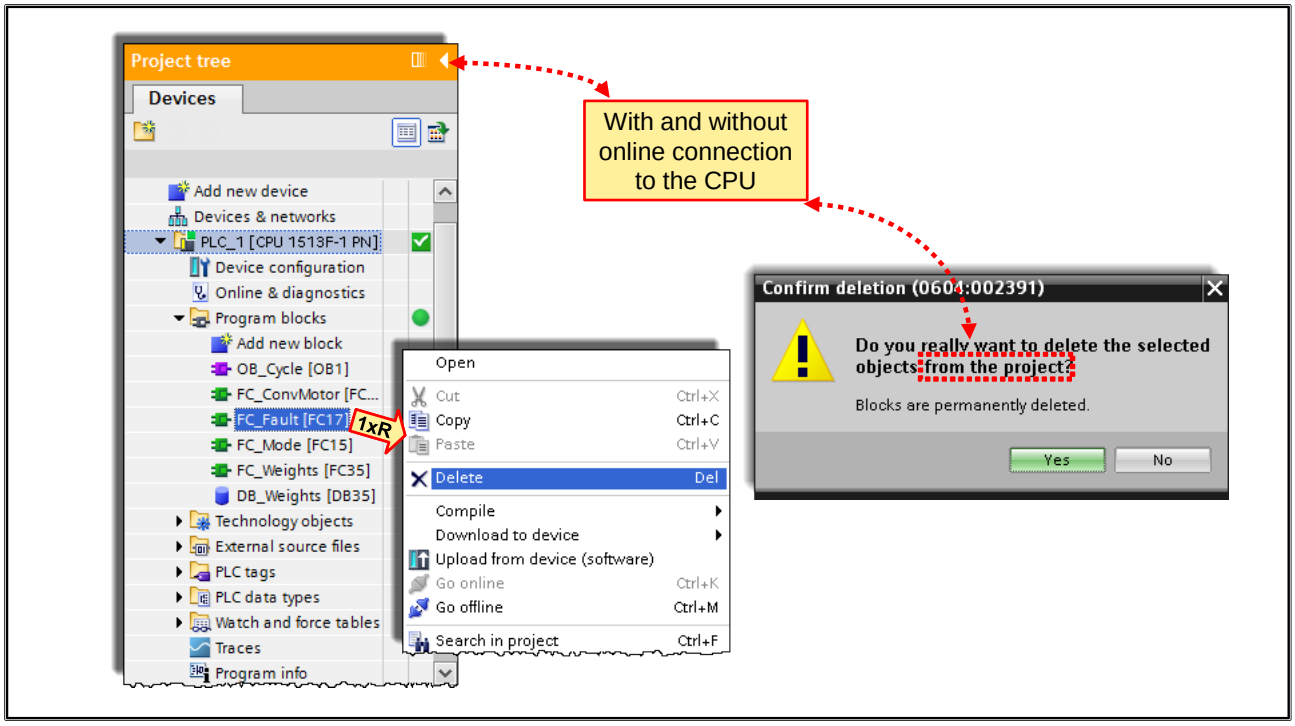


In order to program the call of a block, you simply have to drag the block into the program code of the calling block using drag & drop or copy it using copy & paste.

Block Calls

If a block calls another block, the instructions of the called block are executed. Only when the execution of the called block is completed, is the execution of the calling block taken up again and execution continues with the instruction that follows the block call.

7.7. Deleting Blocks

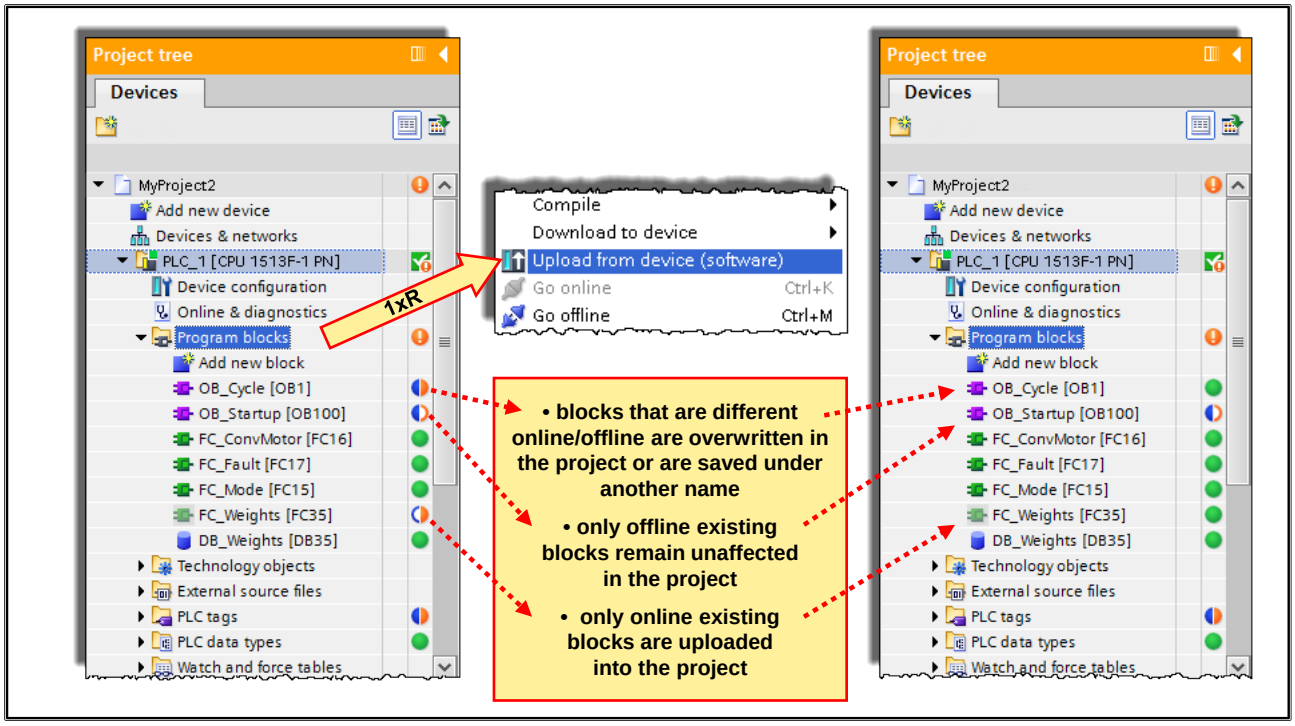


Deleting Blocks

Online, blocks cannot be deleted directly in the CPU. If a block (even with an existing online connection) is selected and "Delete" is activated, the dialog shown in the picture appears with the question of whether the block is to be deleted *offline*.

With subsequent, consistent loading of the entire program, the blocks which only exist in the CPU are deleted there online.

7.8. "Upload" Blocks "from Device" (Upload into Project)



Uploading Blocks into the Project:

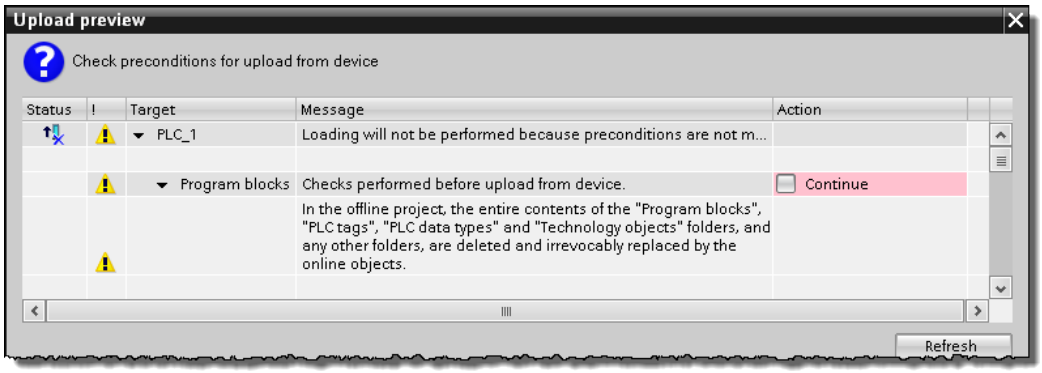
With "Upload from device (software)", individual blocks or the entire CPU program including technology objects, PLC tag tables and PLC data types can be uploaded into the project from the CPU.

If the Blocks folder or individual blocks is/are selected, then...

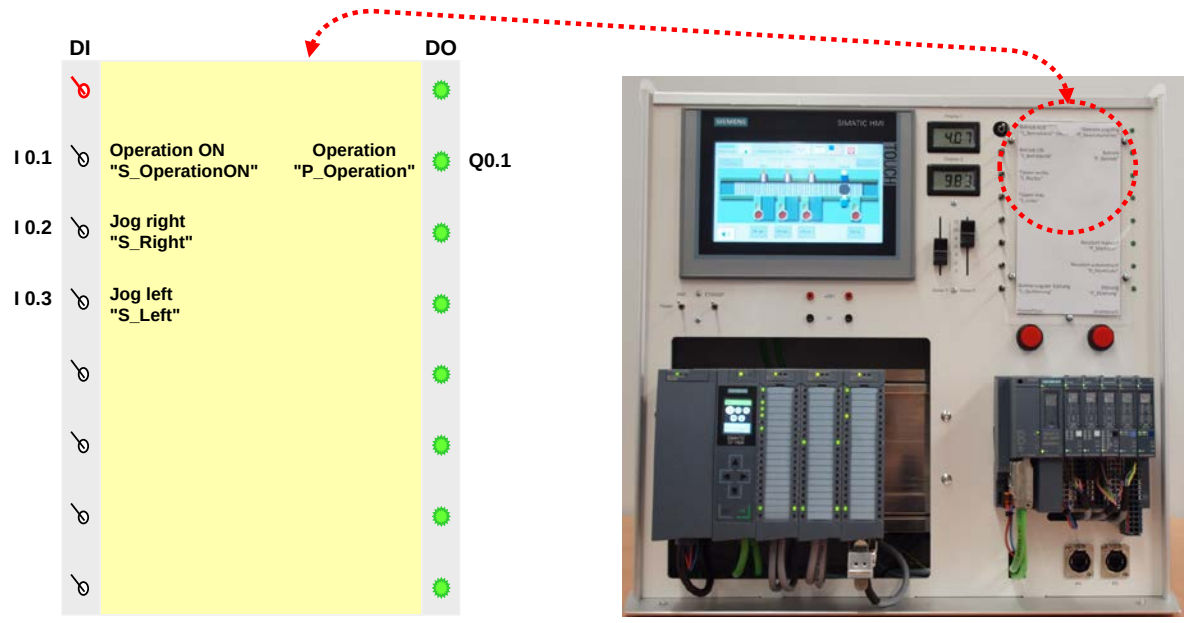
- ... blocks that only exist online in the CPU are uploaded into the offline project.
- ... blocks that are different online / offline are either overwritten in the offline project with the uploaded blocks, or the uploaded blocks are additionally saved under a different name (but with the same number!) in the project. (selectable in the Upload dialog)
- ... blocks that only exists offline are not affected.

If the Station is selected, then...

- ... offline all blocks, PLC data types, technology objects and symbols are **deleted (!)** and the online blocks, PLC data types, technology objects and symbols are uploaded into the project.



7.9. Task Description: Programming the Mode Selection and Manual Mode

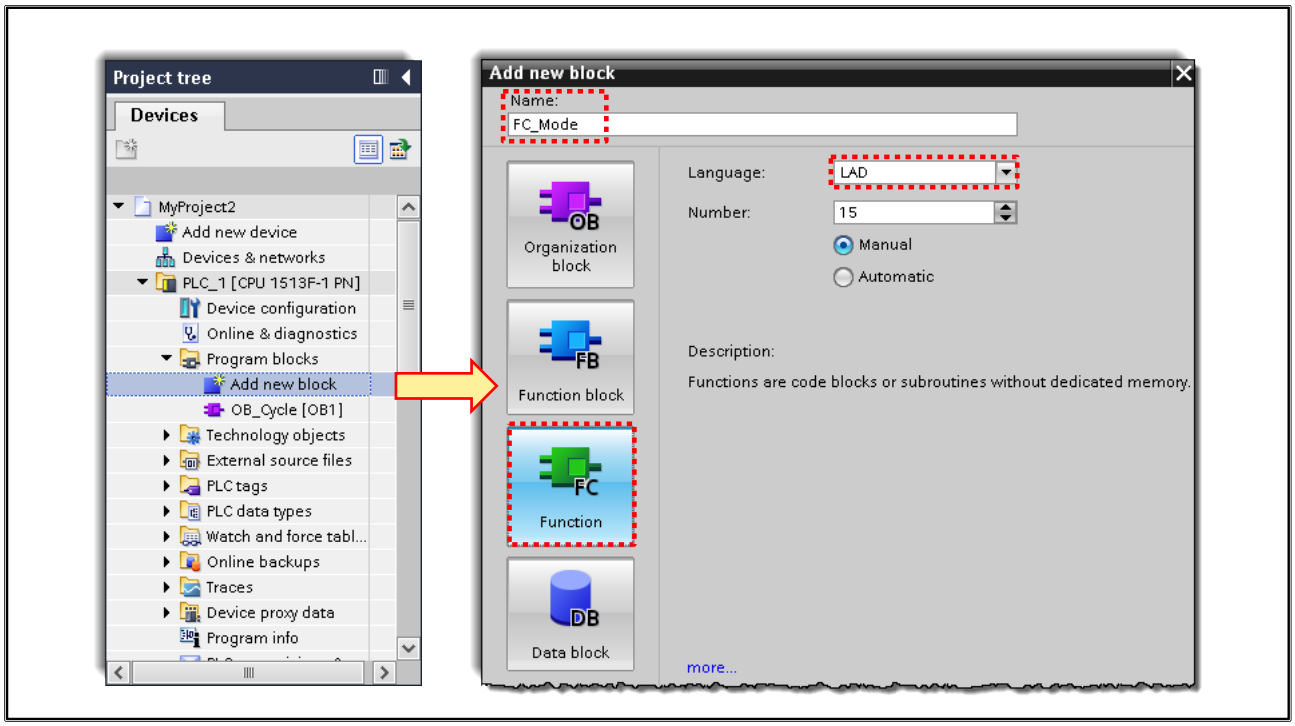


Task Description

The Plant is switched ON/OFF and in Automatic or Manual mode using the switch "S_OperationON".

In Manual mode ("P_Operation" = FALSE), the distribution conveyor can be moved to the right and left using the switches "S_Right" and "S_Left". If both switches are activated simultaneously, then the conveyor must not move (Lock-out!).

7.9.1. Exercise 1: Adding the "FC_Mode" Block



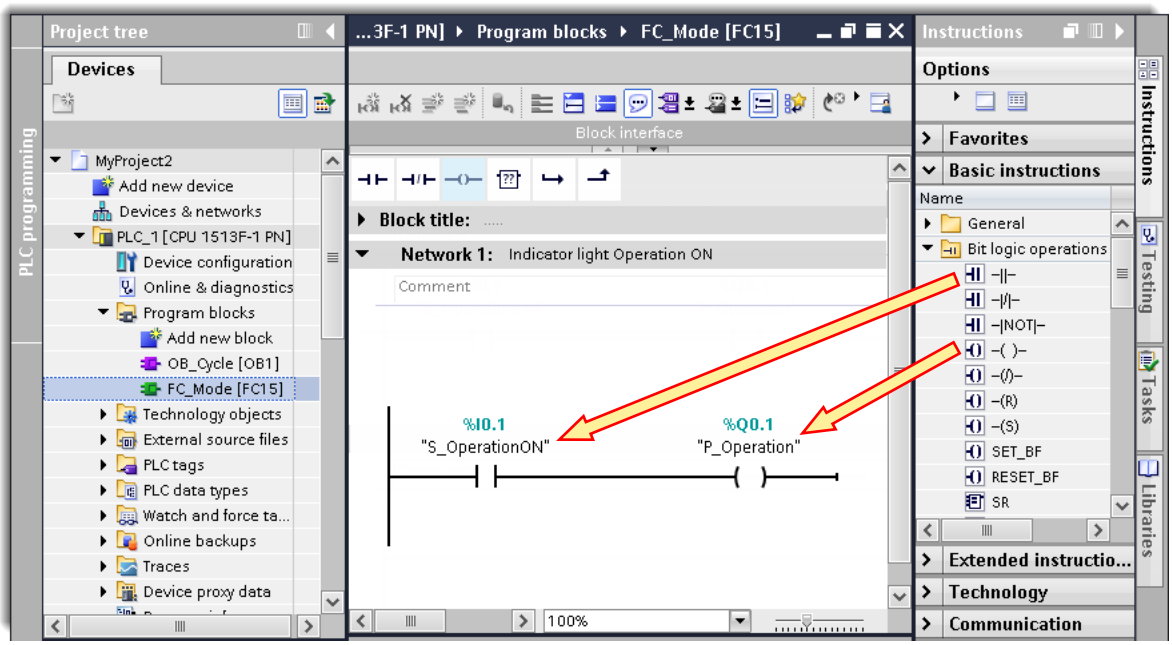
Task

You are to create "FC_Mode" as a new block in which you will program the required mode selection in the next exercise.

What to Do

1. In the "Program blocks" container, double-click on "Add new block".
2. In the dialog that appears, make the entries as shown in the picture above.

7.9.2. Exercise 2: Programming the Mode Selection in "FC_Mode"



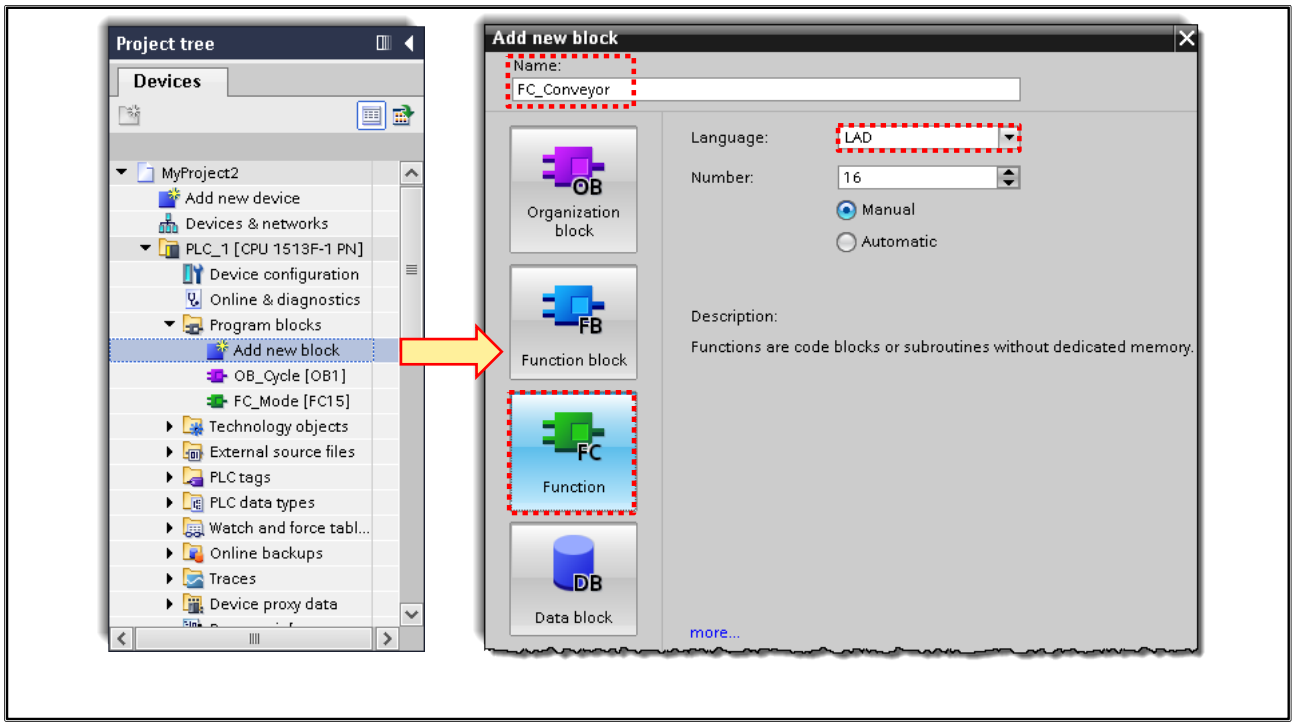
Task

You are to program the mode selection as shown in the picture.

What to Do

1. Program an Instruction by pulling it from the "Instructions" task card using drag & drop.
2. Above the instruction, enter the output "P_Operation" (Q0.1) as the operand (you can enter the symbol as well as the absolute address)
3. Assign the tag "S_OperationON" (I 0.1) as the operand to the input of the Instruction.
4. Label both the block and the network.
5. Close the block.
6. Save your project.

7.9.3. Exercise 3: Adding the "FC_Conveyor" Block



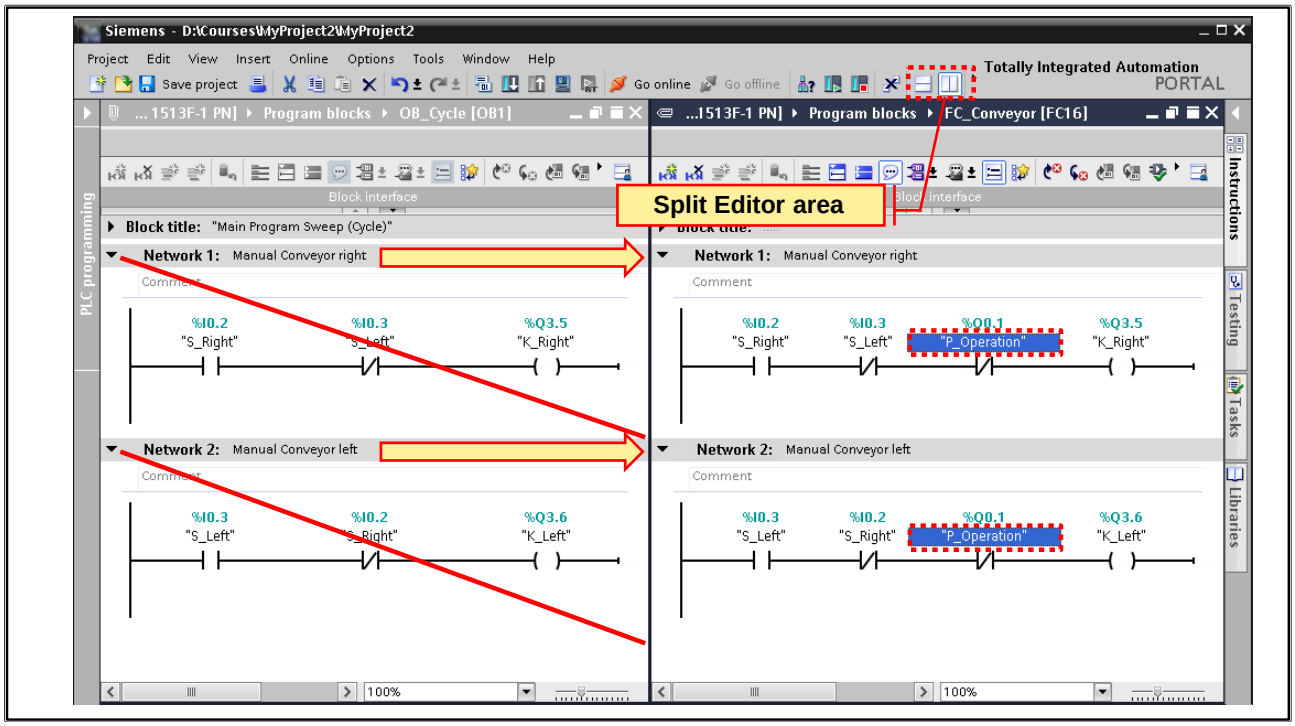
Task

You are to create the function "FC_Conveyor" as a new block in which you will then program the Jog mode that is possible in Manual mode in the next exercise.

What to Do

1. In the "Program blocks" container, double-click on "Add new block".
2. In the dialog that appears, make the entries as shown in the picture above.

7.9.4. Exercise 4: Shifting the Networks from "OB_Cycle" to "FC_Conveyor" and Expanding it



Task

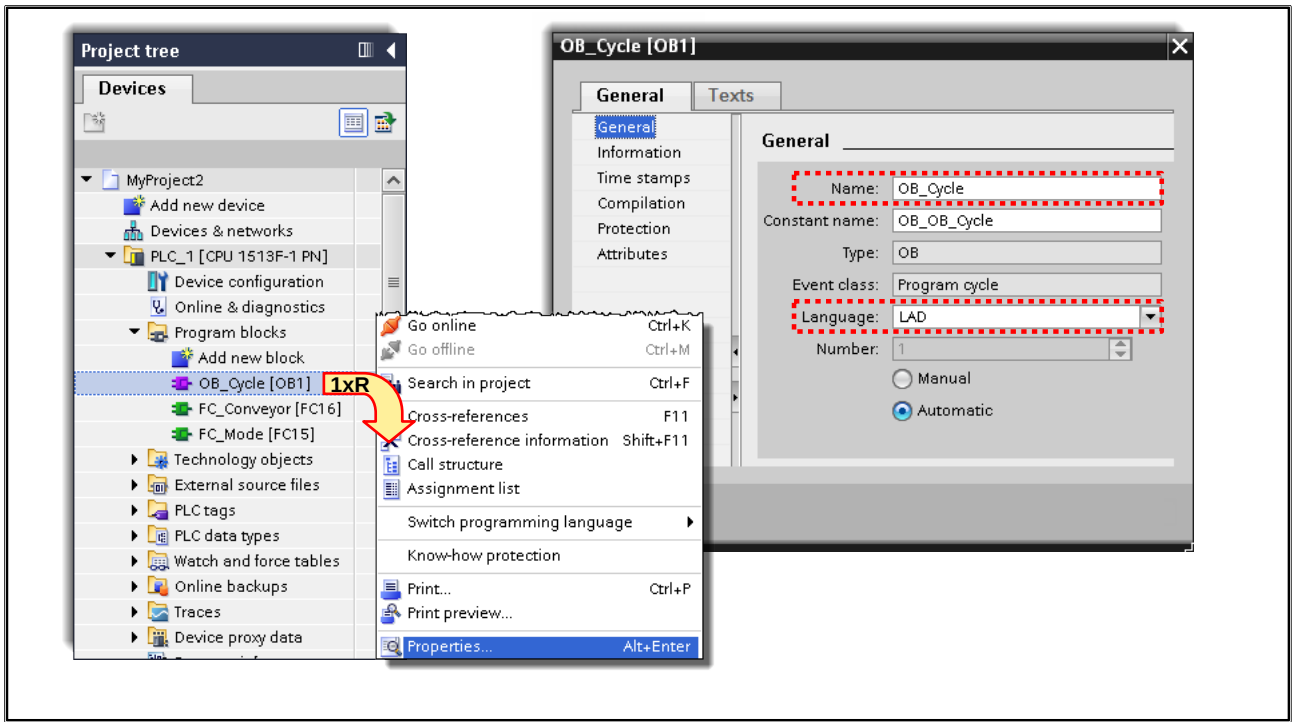
You are to program the Jog mode of the conveyor as shown in the picture.

What to Do

1. Split the Editor (area) horizontally or vertically via one of the appropriate buttons in the toolbar.
2. In each of the areas, open the blocks "OB_Cycle" and "FC_Conveyor".
3. Copy the networks from the "OB_Cycle" block into the "FC_Conveyor" blocks using drag & drop.
4. Since Jogging the conveyor belt is only to be possible when "P_Operation" is FALSE, expand the AND logic in the first network by adding a "Normaly closed contact" You do this by dragging it from the Task Card "Instructions" or from the "Favorites".

   Note:
   If an incorrect type of contact was inserted, click on the contact and use the small red "arrow" in the right corner of the contact to access an expansion menu with the available changes.
5. Assign the "P_Operation" (Q0.1) tag to the new "Normaly closed contact"..
6. Change the logic in Network 2 for Jogging the conveyor to the left in the same way.
7. Delete the networks in "OB_Cycle".
8. Close the blocks.
9. Save your project.

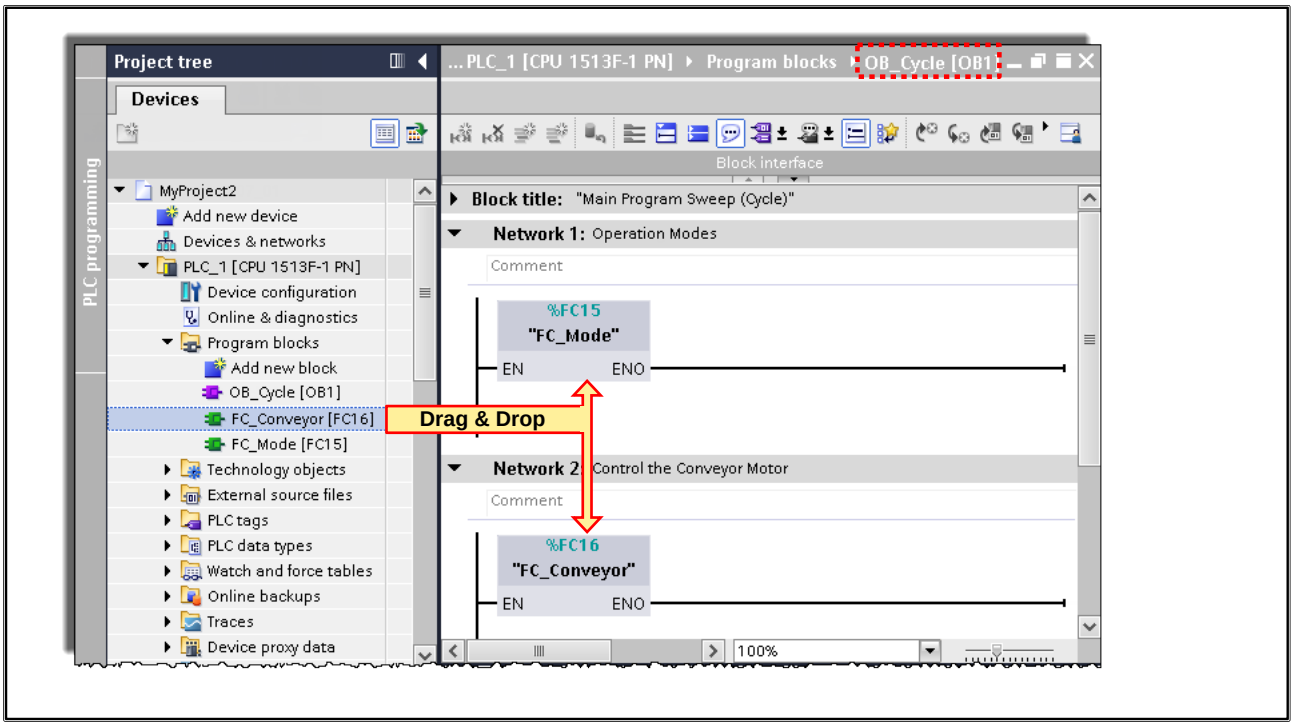
7.9.5. Exercise 5: Checking the "OB_Cycle" Properties



Task

You are to check the Properties of the OB1 block as shown in the picture, and, if necessary, correct them.

7.9.6. Exercise 6: Calling "FC_Mode" and "FC_Conveyor" in "OB_Cycle"



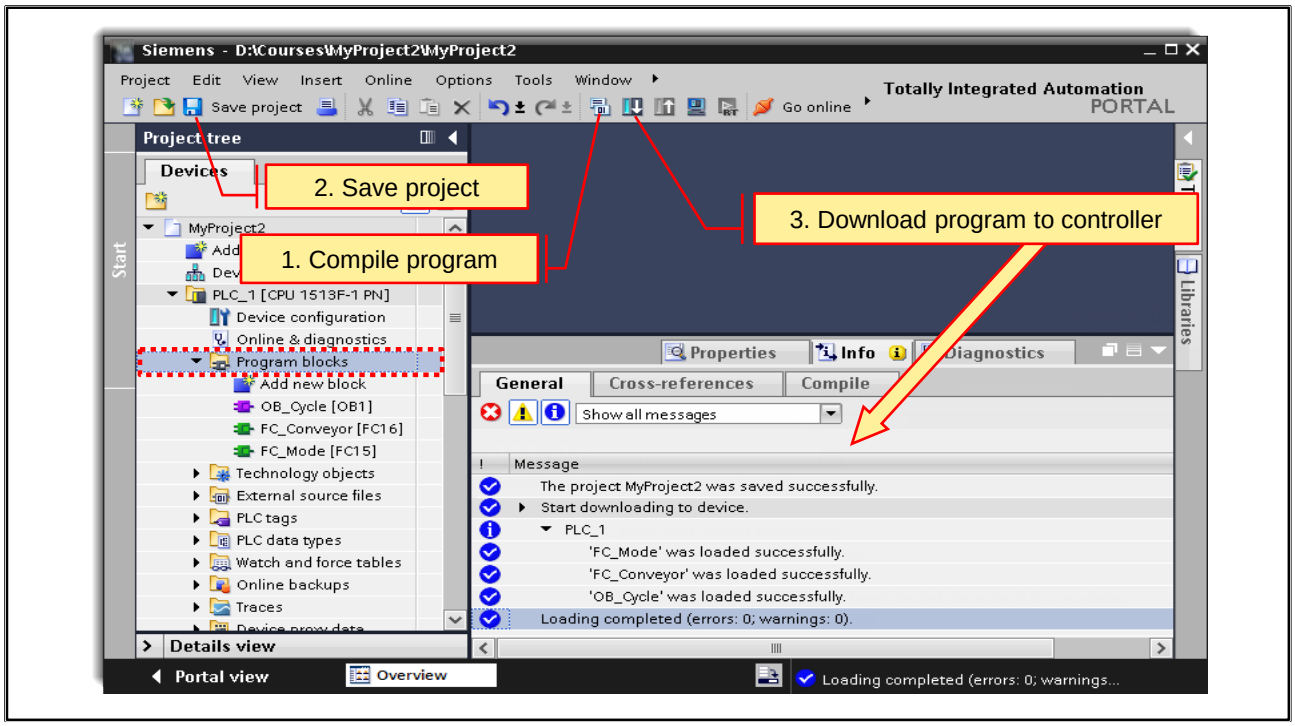
Task

So that the newly created blocks "FC_Mode" and "FC_Conveyor" can be executed cyclically, they must be called in "OB_Cycle".

What to Do

1. Open the "OB_Cycle" block by double-clicking on it.
2. Program the calls of the "FC_Mode" and "FC_Conveyor" blocks as shown in the picture using drag & drop.
3. Edit the labels for the Networks (as in the picture).
4. Close the block.
5. Save your project.

7.9.7. Exercise 7: Compiling, Downloading and Saving the Program



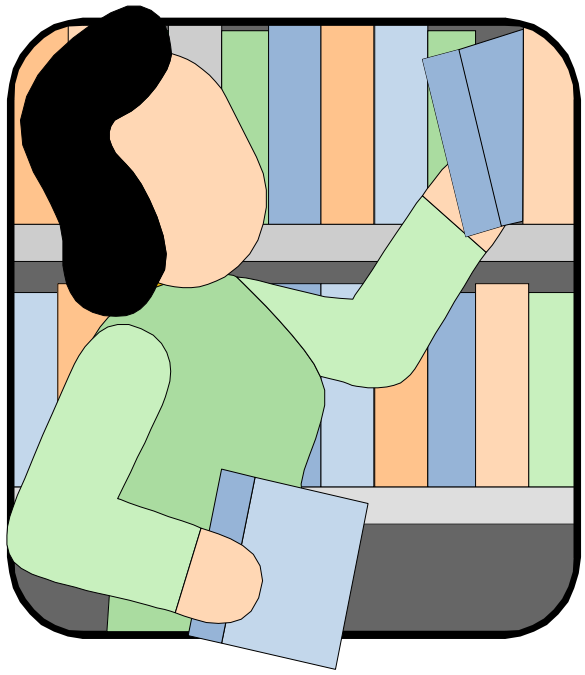
Task

The newly programmed blocks are to be compiled, downloaded into the CPU and saved offline in the project data storage.

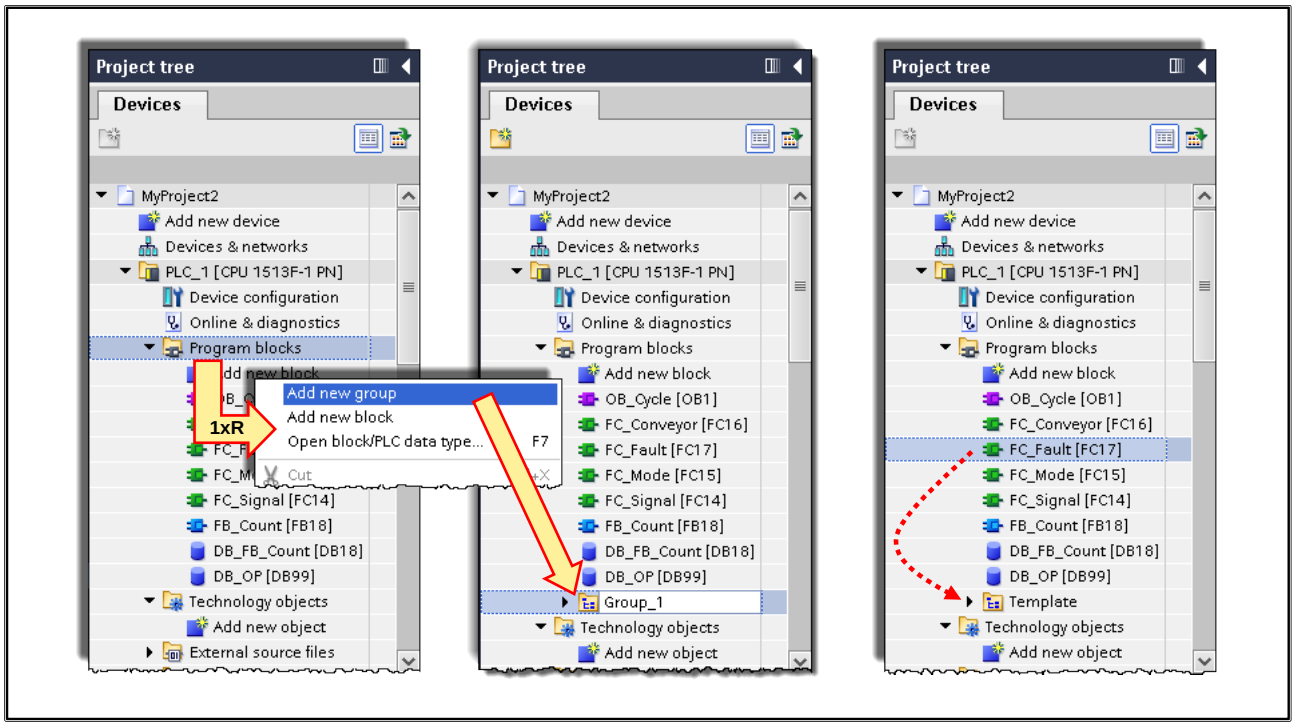
What to Do

1. To compile the entire program (all blocks) and to download it into the CPU, select the "Program blocks" folder in the Project tree.
2. Carry out the steps shown in the picture and check the program functioning
   a. Switching between Manual and Automatic mode by activating the simulator switch "S_OperationON" accordingly as well as checking the LED "P_Operation".
   b. Try jogging with Manual or Automatic mode by activating the simulator switches "S_Right" (I 0.2) and "S_Left" (I 0.3).
3. If necessary, also use the Test function "Monitor in block" or use a watch table.

7.10. Additional Information



7.10.1. Block Groups

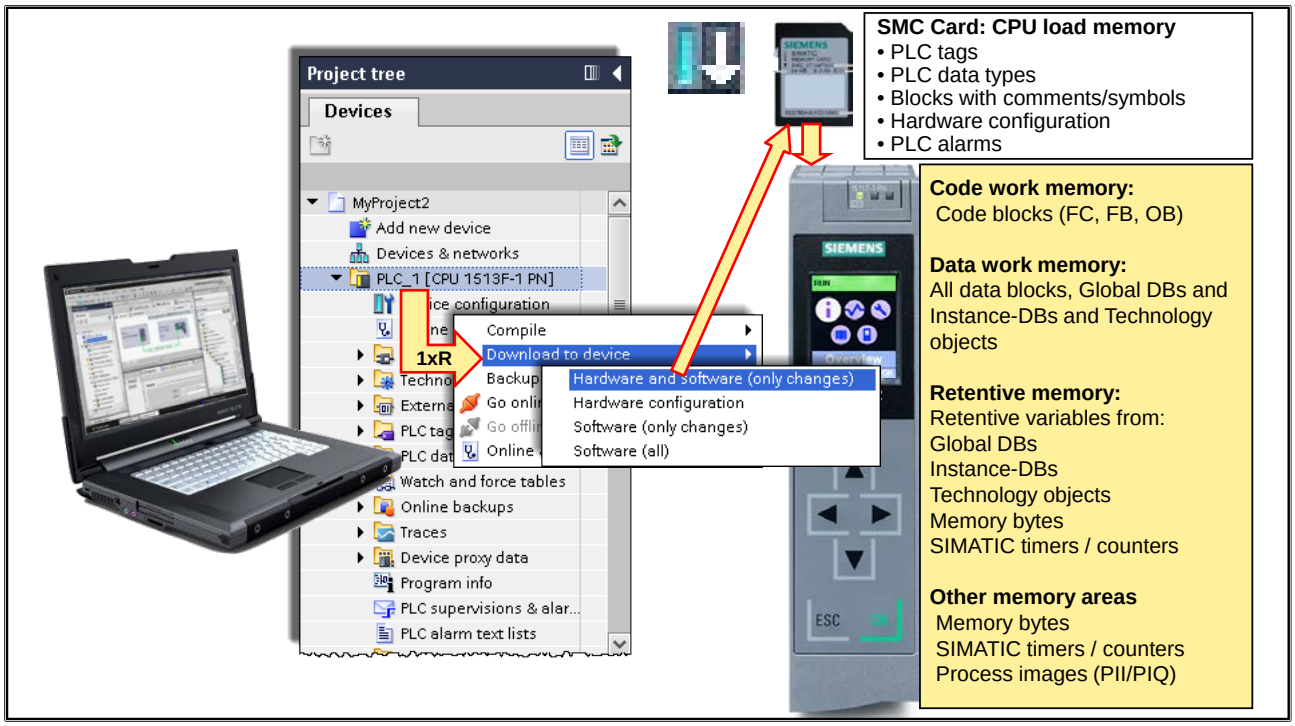


Block Groups

To achieve more clarity, large programs with many blocks can be divided into several block groups. The groupings can, for example, be related to the structure of the system to be controlled. Even if the blocks are managed in different groups, each block must have a unique symbolic name. Regardless of the groupings, the sum of all blocks represents the user program of the controller.

The blocks can easily be shifted between the block groups using drag & drop.

7.10.2. S7-1500 - Memory Concept



Memory Areas of the CPU

The SMC (Simatic Memory Card) is the load memory of the CPU. Accordingly, an inserted SMC is absolutely necessary for operating the CPU.

The data of the entire station is stored on the SMC, that is, the complete S7 program including documentation, PLC tags and data types as well as the complete hardware configuration including distributed I/O and parameter assignments. In addition, other files can also be found on the SMC, such as, recipes, HMI backups.....

When downloading the S7 program into the CPU, all blocks are always first loaded into the load memory from where the CPU automatically copies the parts of the blocks that are relevant for execution into the work memory.

Work Memory

The work memory is a volatile RAM memory which cannot be expanded. It is divided into two areas:

- Code work memory which contains the parts of the code (logic) blocks relevant for execution.
- Data work memory which contains the data of the data blocks and technology objects relevant for execution.

Retentive Memory Depends on the CPU (88-700kByte)

The retentive memory is a non-volatile memory for saving the variables declared as retentive whose value is retained during a power failure.

The contents of the retentive memory is only erased by a

- memory reset
- resetting of the CPU to factory settings