fischertechnik

GROLLMUS

KNOWLEDGE | SOFTWARE | AUTOMATION

24V production line

Structured programming

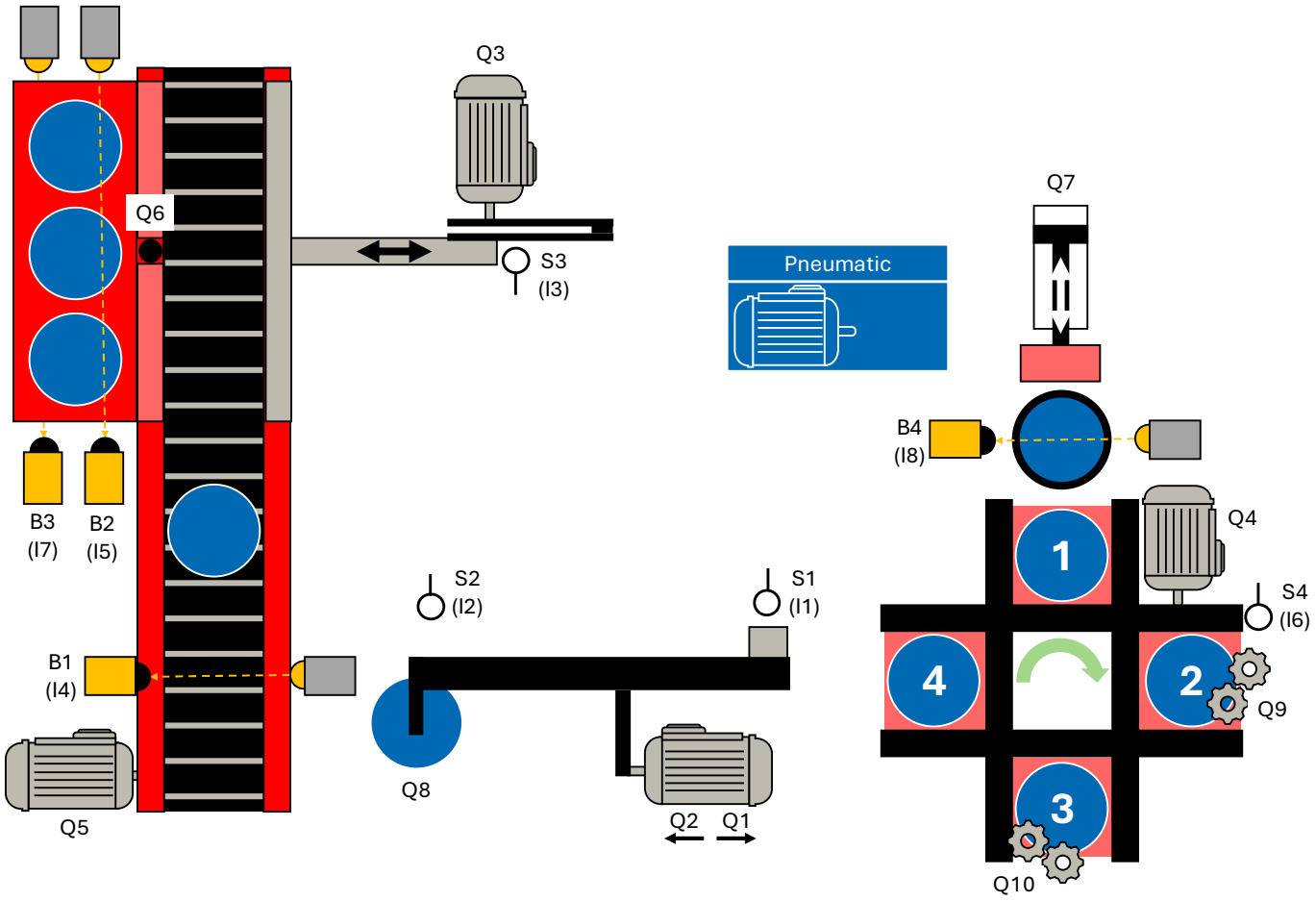


Table of contents

5 Structured programming

5.1 Introduction

Structured programming in PLC systems is used to organize complex programs by dividing them into smaller, clear blocks. This leads to improved readability, maintainability and reusability of the code. The user program can be structured according to technological or functional aspects.

In a PLC program, blocks such as functions (FC) and function blocks (FB) are used to structure program sections.

The blocks should communicate with each other via their block interfaces instead of accessing global variables directly. Parameters are transferred via inputs and outputs as well as InOut parameters.

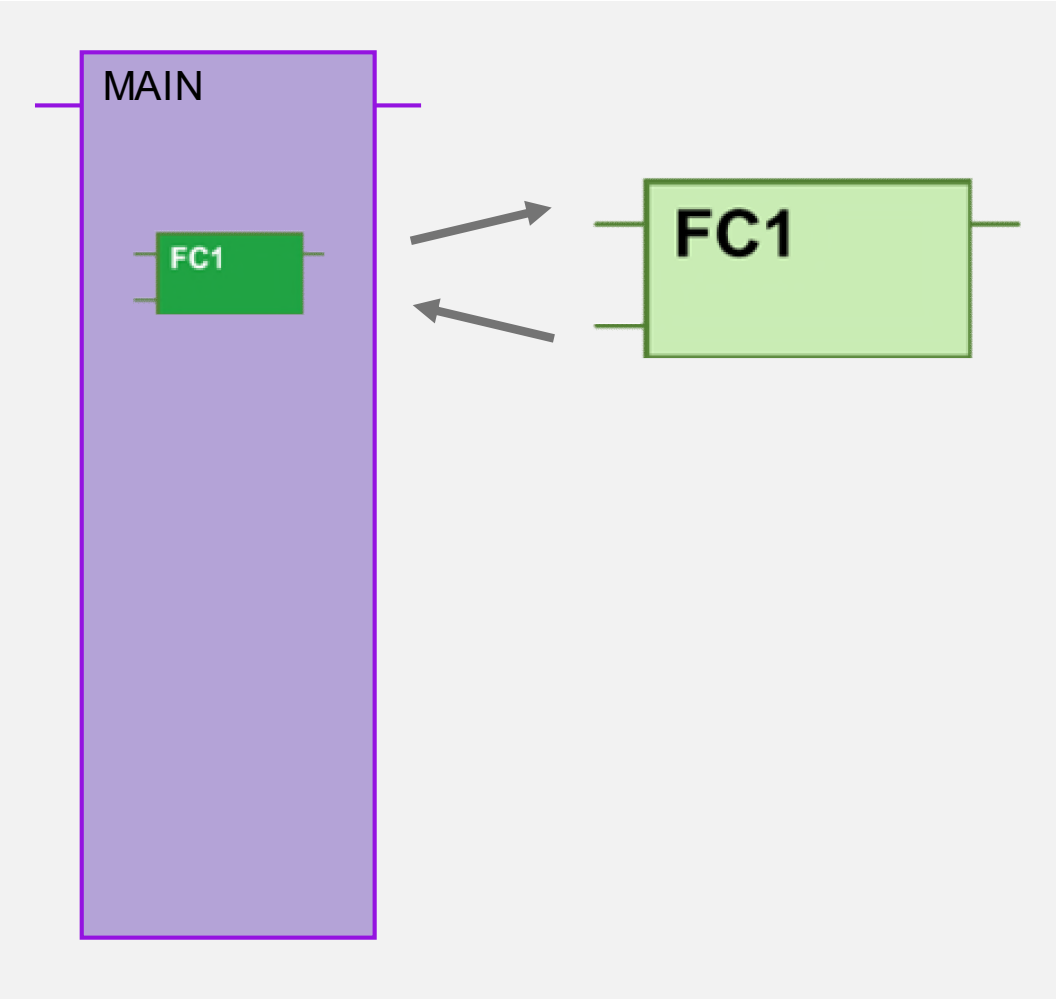
To execute the code blocks in the control program, they must be called.

5.2 Function

Functions (FCs) are code blocks without memory. They have no data memory in which the values of function block parameters could be stored. For this reason, all interface parameters must be connected when a function is called. To save data permanently, global data blocks must first be created.

Functions are ideal for tasks that do not require memory over several cycles, such as mathematical calculations or logical operations.

A function contains a program that is always executed when the function is called by another code block.



Picture 1 Example: Calling a function from MAIN

A function can also be called several times at different points within a program.

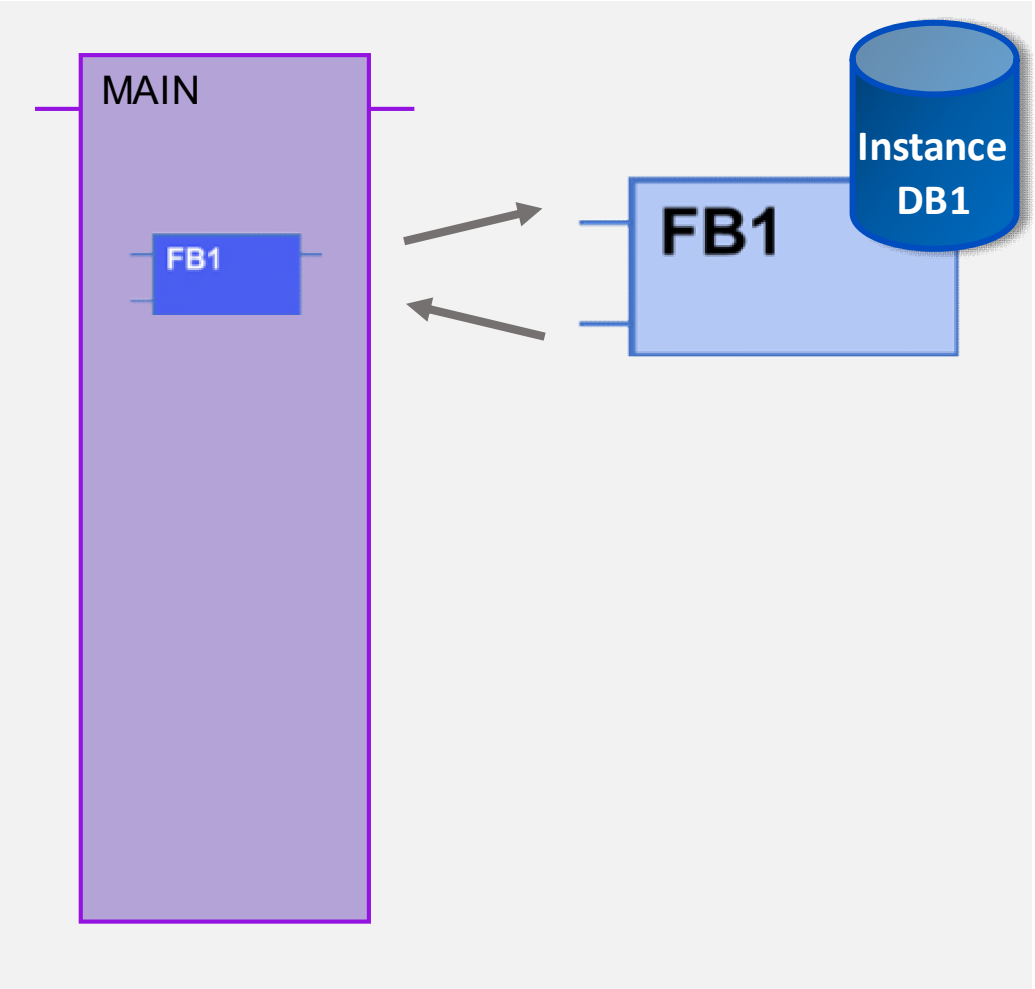
5.3 Function module

Function blocks (FBs) are code blocks that permanently store their input variables, output variables, pass-through variables and also the static variables in instance data blocks so that they are also available after block processing. This is why they are also referred to as blocks with memory.

Function blocks are used for tasks that cannot be realized with functions:

- Whenever times and counters are required in the building blocks or
- if information must be saved in the program (e.g. status of the step chain).

Function blocks are always executed when a function block is called by another code block.



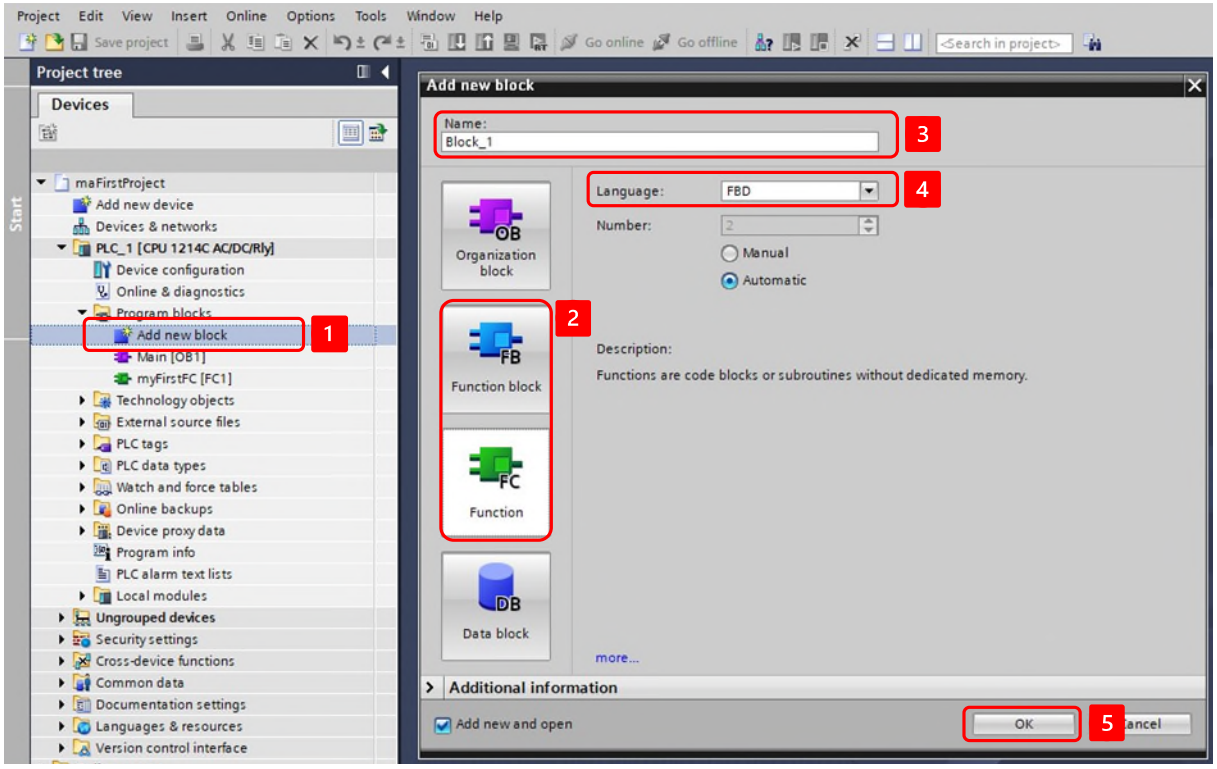
Picture 2 Example: Calling a function module from MAIN

A function module can also be called several times at different points within a program.

A call to a function module is referred to as an instance. Each instance of a function module is assigned a memory area that contains the data with which the function module works.

5.4 Add new module

In the TIA Portal, the blocks are managed in the project navigation below the PLC in the "Program blocks" folder.
Double-clicking on the "Add new block" command within the "Program blocks" folder opens the "Add new block" dialog, which can be used to create a new block.

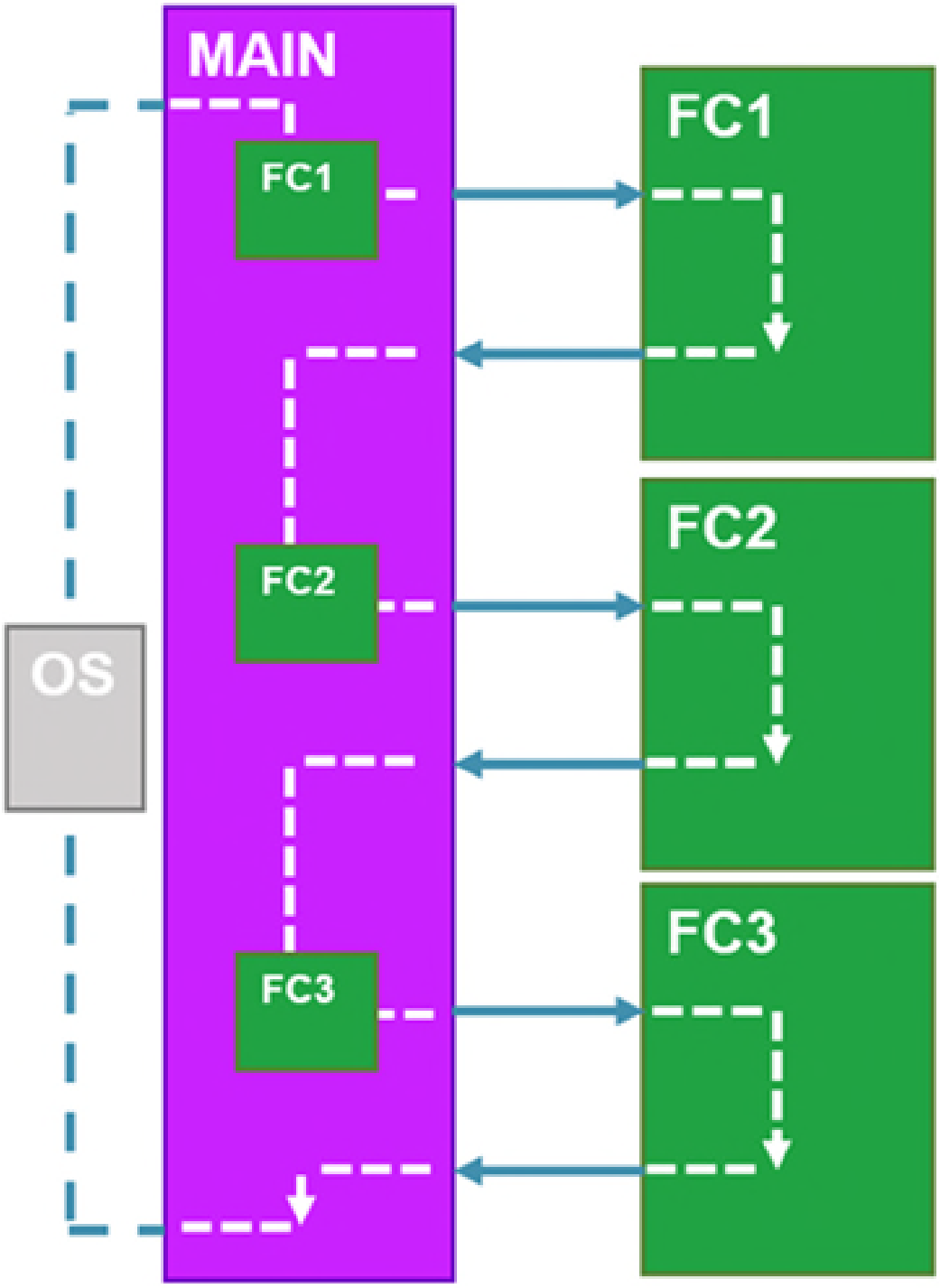


Picture 3 Adding a new module

The block type (2), name (3) and the desired programming language (4) must be selected here.

5.5 Module call

To execute the code blocks in the control program, they must be called. The code module responsible for cyclical program processing is usually referred to as "MAIN". This is started by the operating system and forms the interface to the operating system. The CPU processes the program code located in the "MAIN". The program parts structured in functions and function blocks can be called within the "MAIN".

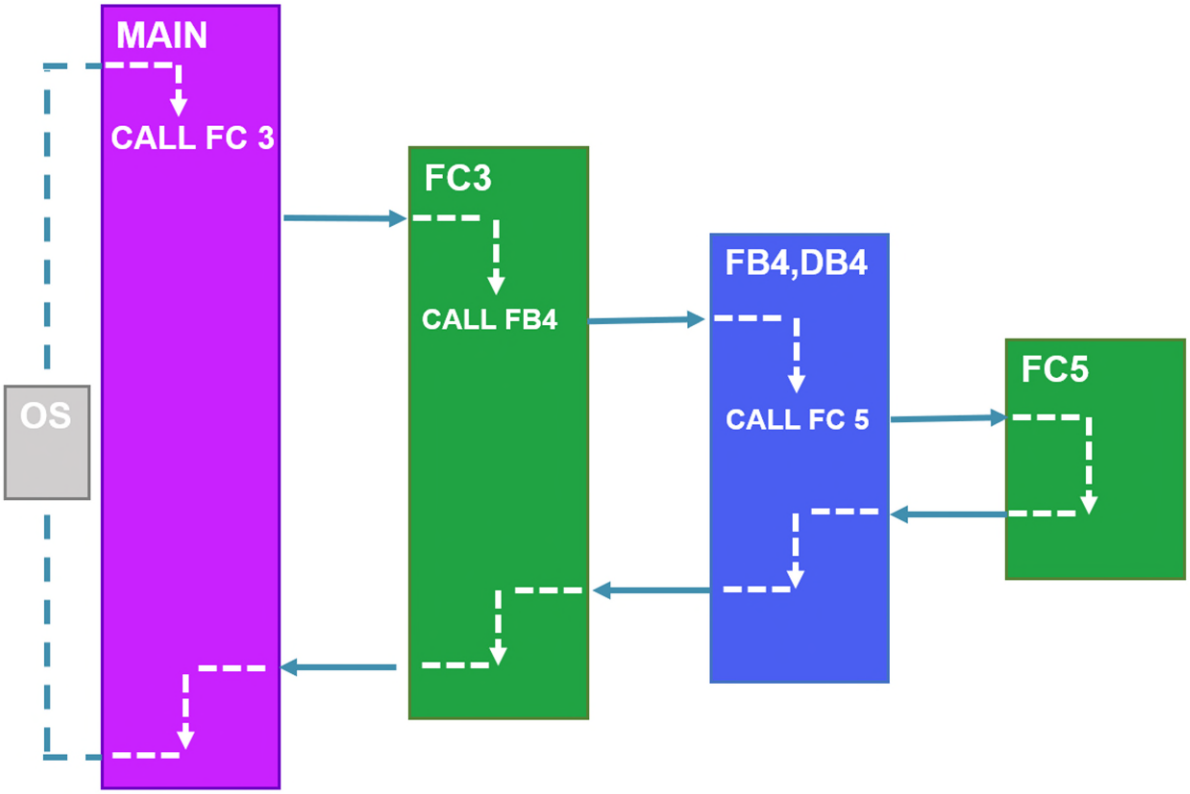


Picture 4 Module call in MAIN

Functions and function blocks structure the program, making it easier to read and maintain.
All called blocks are processed one after the other.

The operating system of the CPU calls the " MAIN " again after the program cycle, whereby all commands programmed in it are executed again.

A block can be processed by calling it from the "MAIN", for example. Alternatively, it can also be called from an FB or FC, which in turn are called in the "MAIN".

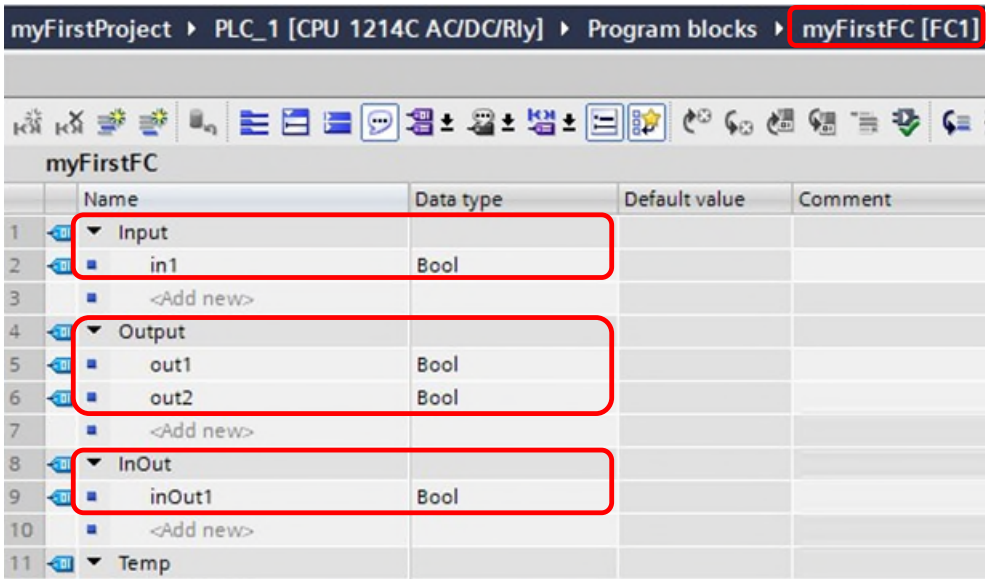


Picture 5 Block call

5.6 Parameter transfer

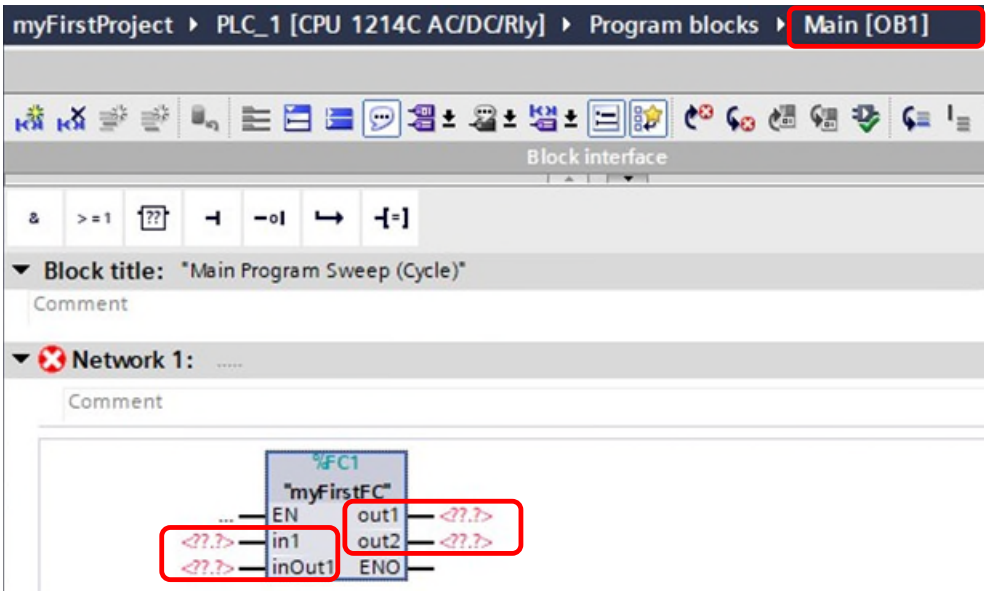
Parameters (variables and variable values) can be transferred when calling functions and function modules. The program code within the code module then works with the transferred values of the formal parameters and can return results via output parameters or the function value.

Data is exchanged via the function block interface. These are declared in the function block interface of the function (FC) or function block (FB) and can be used locally in the function block.



Picture 6 Formal parameters in the function block interface

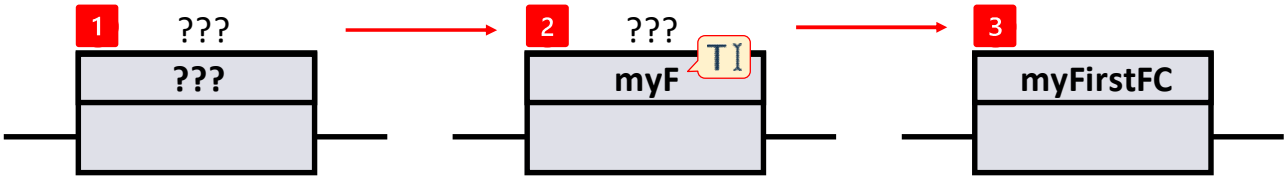
When the block is called, these formal parameters are linked with actual parameters (global variables of the PLC).



Picture 7 Transfer of the current parameters when calling

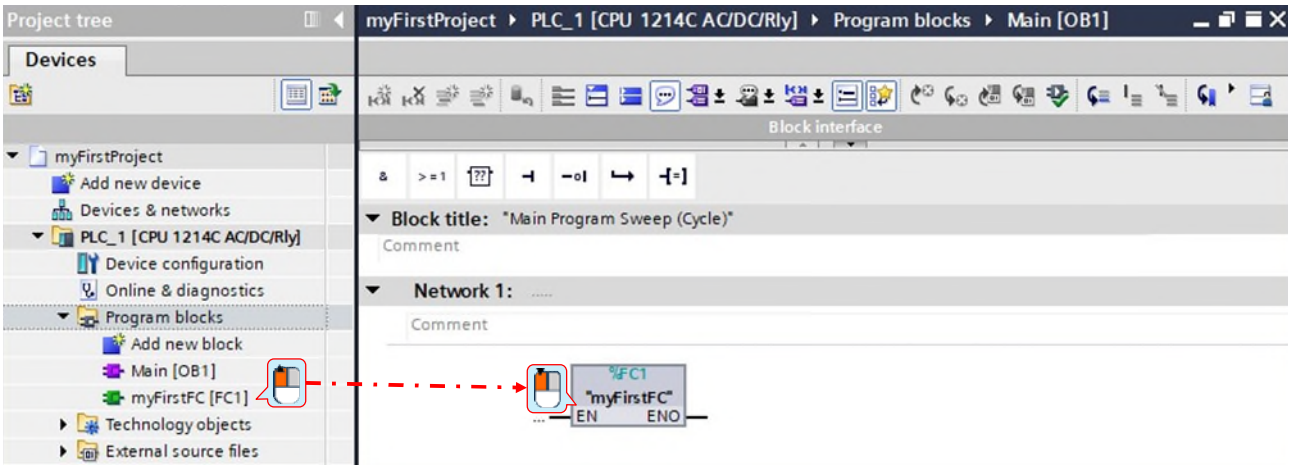
5.7 Calling a function (FC) in FBD

In order for the function to be processed, it must be called by the program. The call can be made by inserting an empty box (TIA shortcut "F8"). After we have inserted the empty box (1), we replace (2) the placeholders ("???") in the empty box with the symbolic function block name, so the empty box is replaced by the corresponding function block call.



Picture 8 Block call from empty box

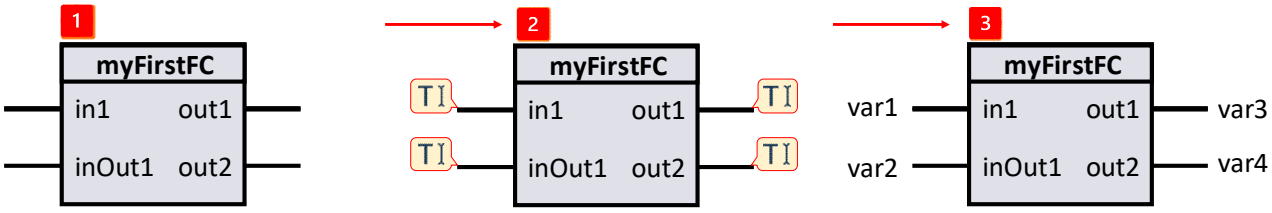
In the TIA Portal, the desired module can also be called up using drag & drop by dragging it from the project navigation to the desired position:



Picture 9 Block call in the TIA Portal

Parameter transfer

If the called block has interface parameters, these are displayed. For functions, the formal parameters must be supplied with actual parameters.

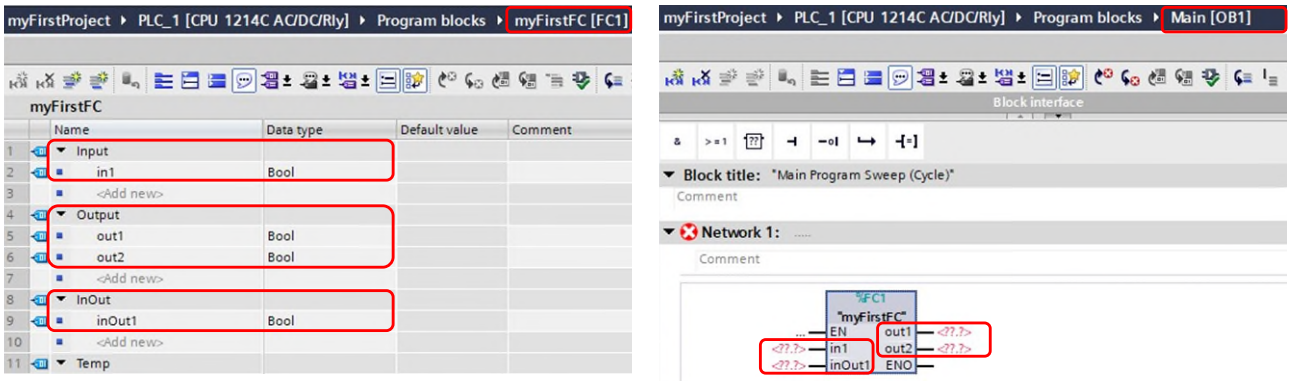


Picture 10 Function with function block interface

Parameters of type "Input" and "InOut" are displayed on the left of the function block using connection legs. Parameters of type "Output" must be connected to the right of the function block.

Example

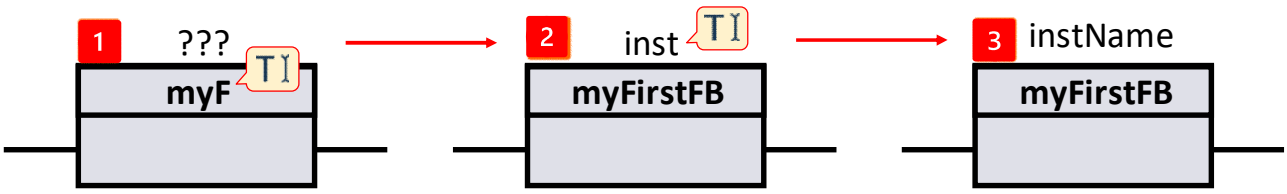
After creating the "myFirstFC" function, it was called in the "Main" (OB1). The parameters have not yet been transferred; the formal parameters to be linked were initially provided with placeholders "<????>".



Picture 11 Function block interface in the TIA Portal

5.8 Calling a function block (FB) in FBD

In order for the function block to be processed, it must be called by the program. The call can be made by inserting an empty box (TIA shortcut "F8"). After we have inserted the empty box (1) and replaced the placeholders in the empty box with the name of the function block, we replace the placeholder above the empty box with the instance name (2).

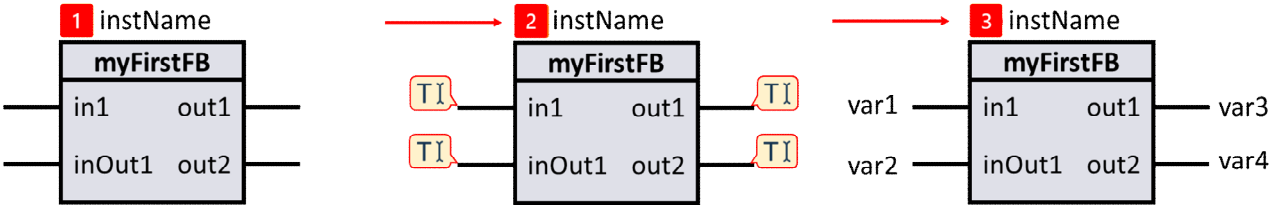


Picture 12 Function block call from empty box

Alternatively, you can call up the function block using drag & drop as shown under Calling up a function.

Parameter transfer

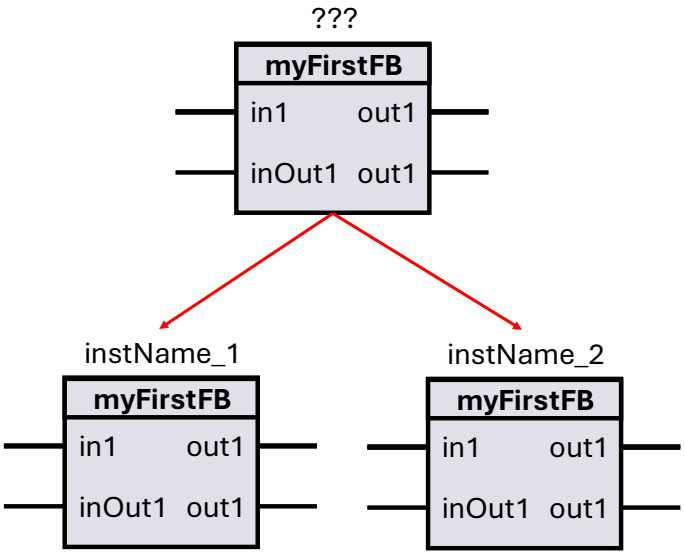
If the called function block has formal parameters in the block interface, these are displayed. It is not always necessary to pass parameters for function blocks, as a private memory area is already assigned to the instance.



Picture 13 Function block with parameter transfer

Instantiate function module multiple times

If a function block (FB) is called twice (instantiated) in the user program, two separate instances are created. In which they can save their data across the cycle.



Picture 14 Instantiation of two FBs

Call options

Depending on the type of calling function block, the instances can be located directly in the function block interface (= multi-instance in the TIA Portal) or stored as global instances (= single instance in the TIA Portal).

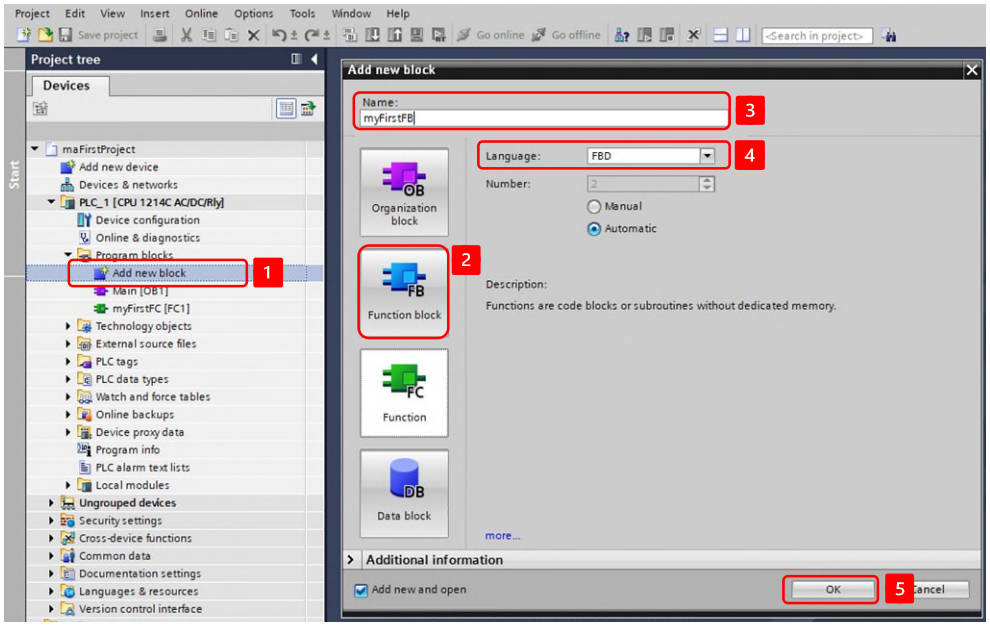
Example

For example, if the FB contains the user program for a calculation of switching operations, each call represents an instance that only uses states at the runtime of this call. A separate instance is required for each call.
This means that all data (information) relating to this calculation is available in this assigned instance.
Before an instance can be created, the assigned FB must already exist.
The variables of the respective instance can be observed in this instance.

5.8.1 Procedure Call with single instance

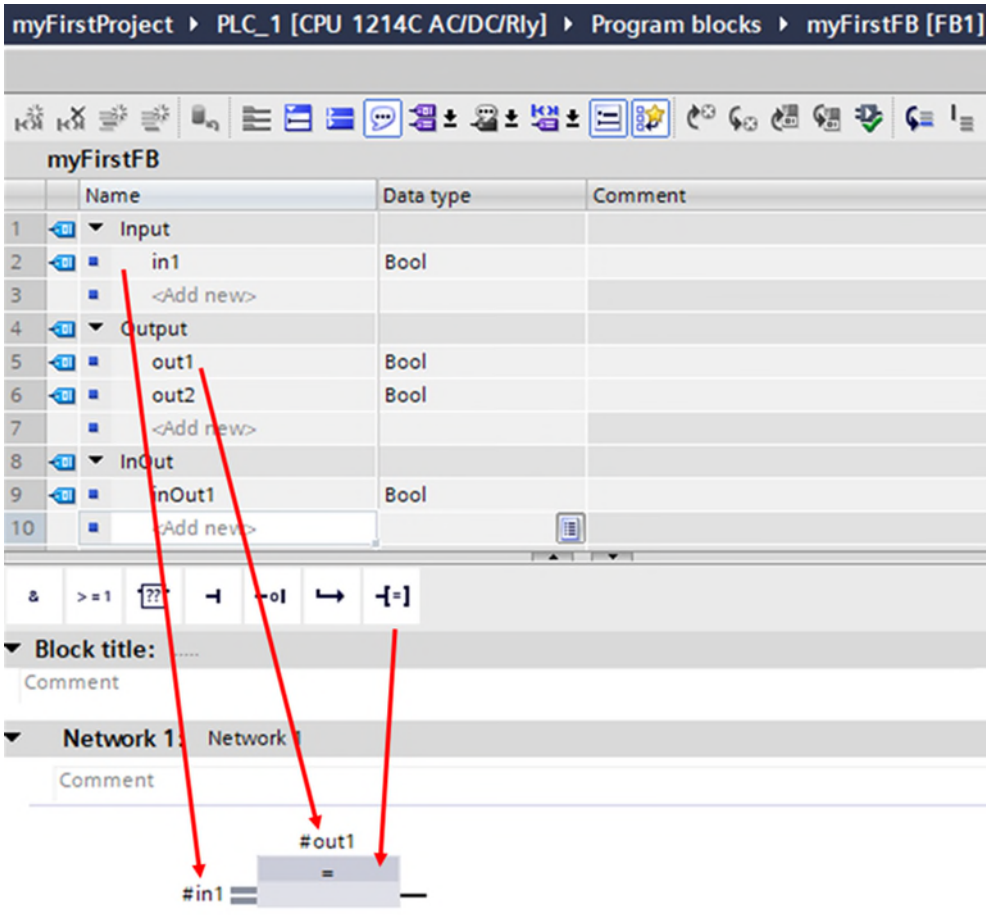
The procedure for calling up the "myFirstFB" function block twice is now shown step by step in the TIA portal:

1. Create the "myFirstFB" function module:



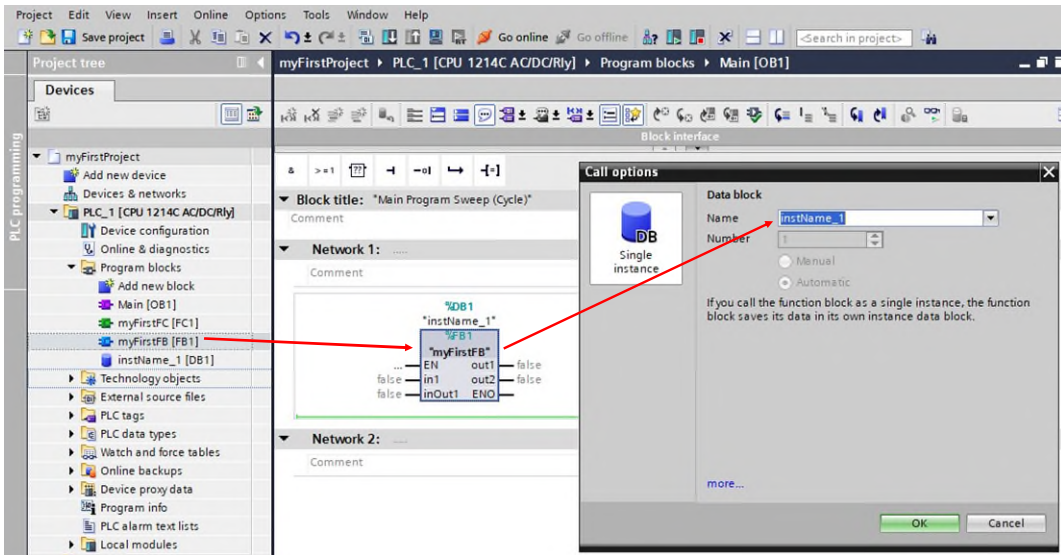
Picture 15 Adding a new block

2. Declaration of the function block interface and connection of the variables in the instruction part of the function block:



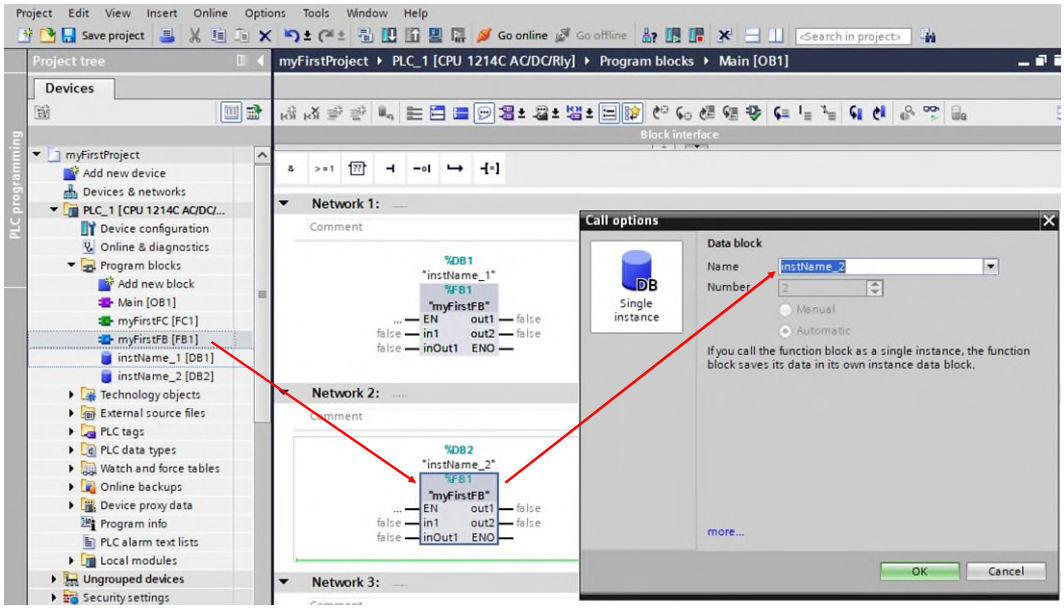
Picture 16 FB with function block interface in the TIA Portal

3. first call of "myFirstFB" in the first network of the OB "MAIN" using drag & drop and declaration of the first instance as a single instance:



Picture 17 Instantiation of first block call

4. second call of "myFirstFB" in the second network of the OB "MAIN" using drag & drop and declaration of the second instance as a single instance:



Picture 18 Instantiation second block call

5. both instance data blocks can be observed:

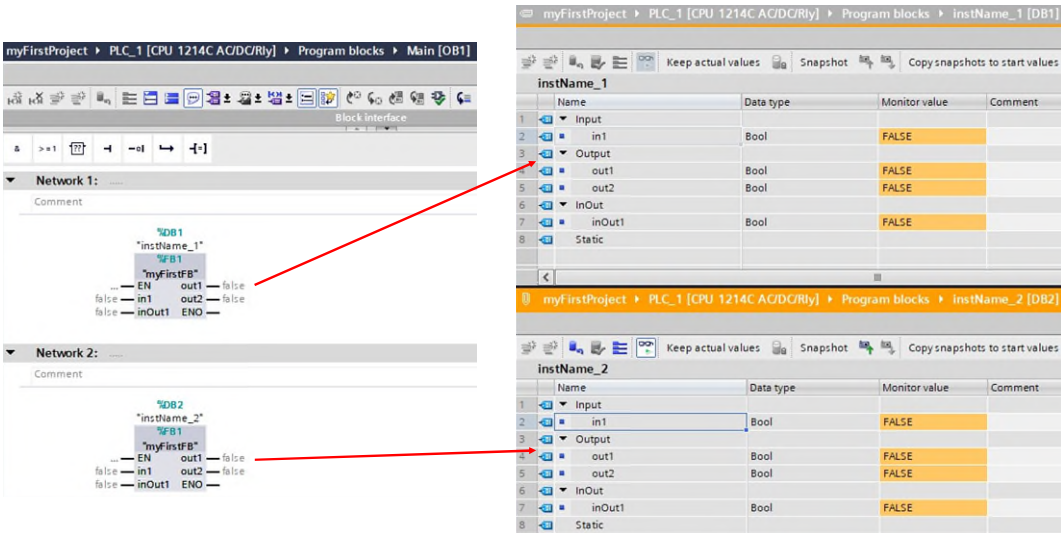
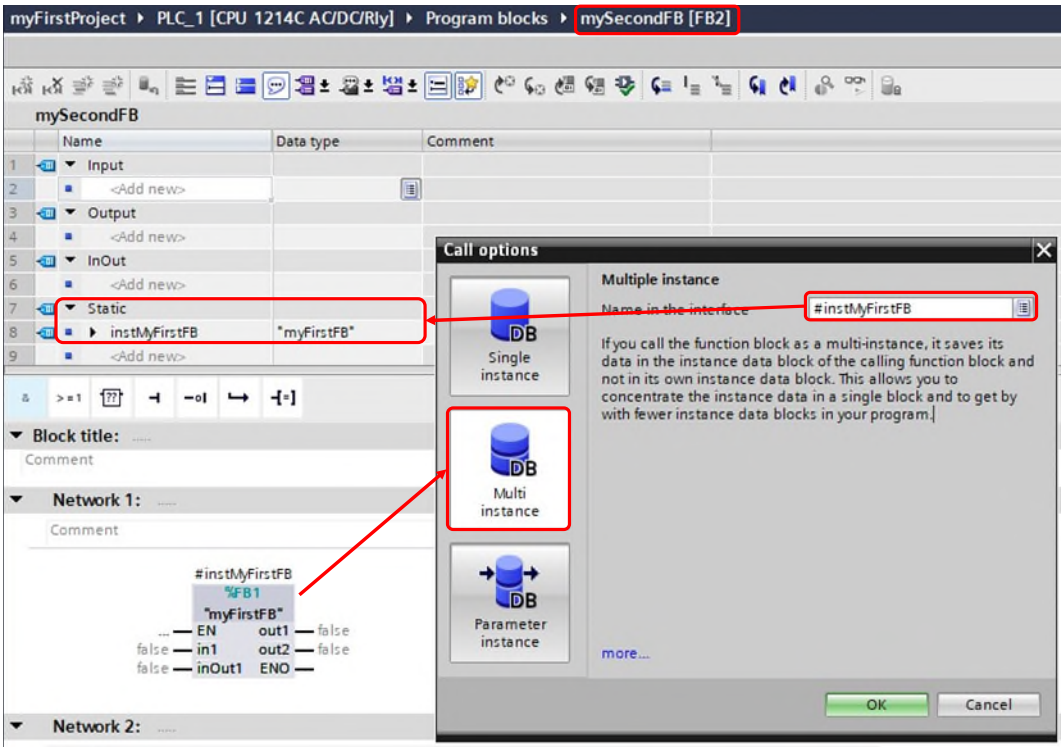


Image 19 Actual values in the instances

Instance data blocks and the possibility of monitoring and controlling them are described in detail in the next chapter (Data blocks).

5.8.2 Call option as multi-instance (TIA portal)

If a function block is called in another function block, the multi-instance can also be selected in the call options.
The number of instance data blocks can be reduced by using multi-instances. When creating subroutines, the use of multi-instances is often absolutely necessary. For example, if you want to implement a runtime counter in a control module for a motor, each motor instance requires its own IEC counter instance. Multi-instances are placed in the block interface of the calling block.



Picture 20 Call options in the TIA Portal

5.8.3 Textual declaration as a multi-instance (CODESYS / Beckhoff)

The textual declaration of the instances is made according to the following scheme.
Syntax:

Instance name (variable name) : Block name (data type);

Example:

```
//Declaration of instances
VAR
  instName_1 : myFirstFB;   //Instance 1
  instName_2 : myFirstFB;   //Instance 2
END_VAR
```

Picture 21 Declaration of the instances

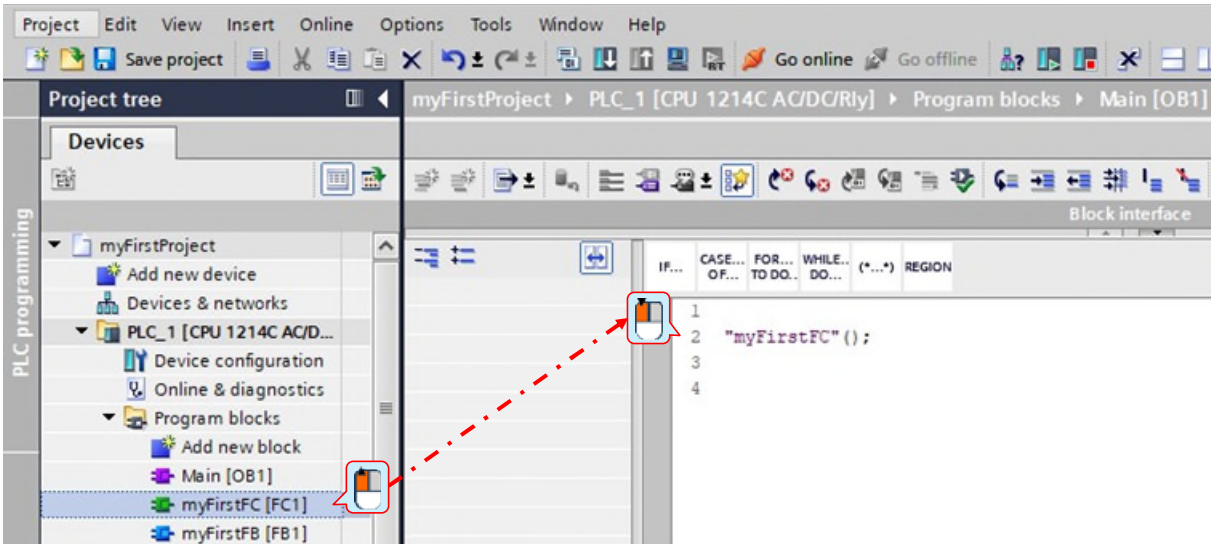
5.9 Calling a function (FC) in ST / SCL

We want to call the function in the program so that it is processed. A function call without a return value in ST / SCL is made using the function name, followed by "()" and a semicolon. The parameters are passed in the round brackets and the semicolon concludes the statement.

```
myFirstFC();
```

Picture 22 Function call ST / SCL

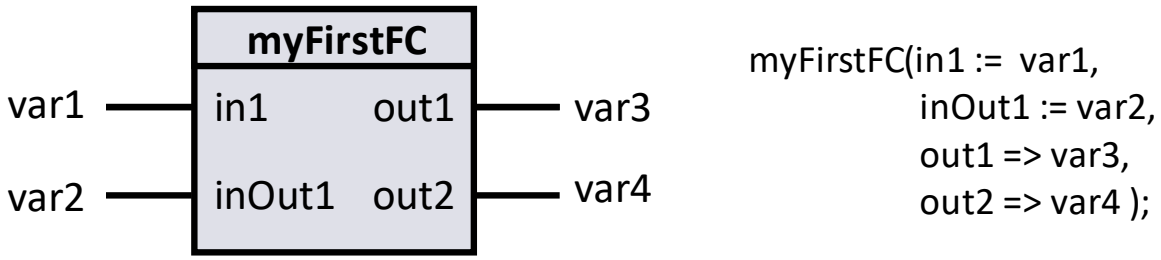
In the TIA Portal, the desired module can also be called up using drag & drop by dragging it from the project navigation to the desired position:



Picture 23 Block call in the TIA Portal

Parameter transfer

If the called block has interface parameters, these are displayed. For functions, the formal parameters must be supplied with actual parameters.
Parameters are passed in round brackets, parameters are separated from each other with ",".

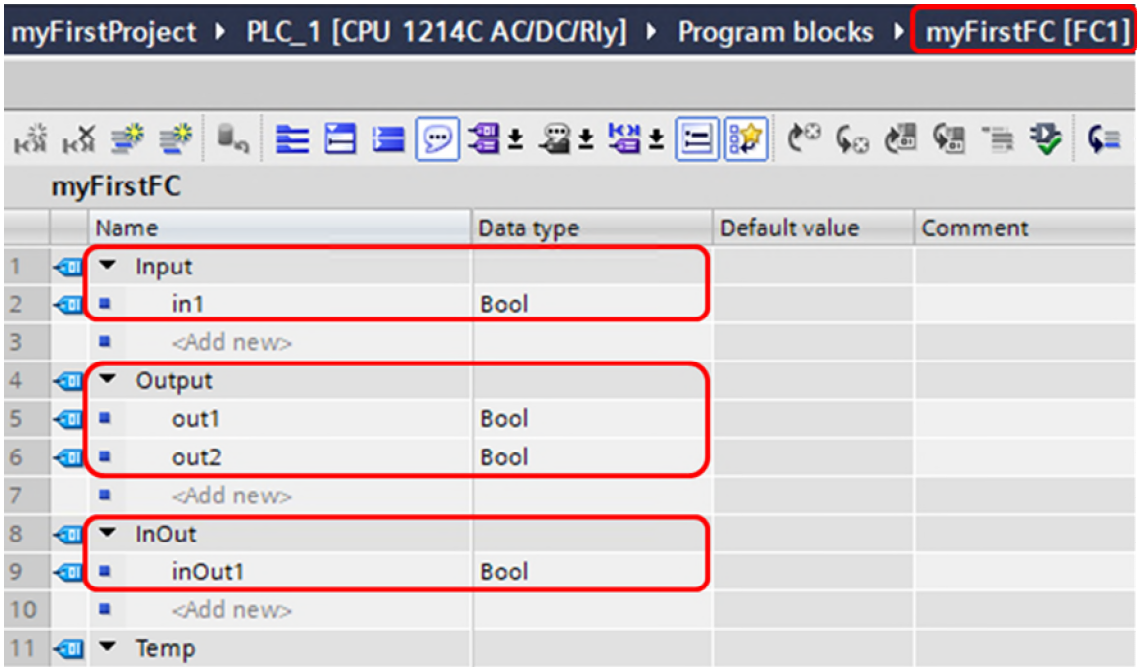


Picture 24 Syntax function call, with parameter transfer in SCL

Parameters of type "Input" and "InOut" are assigned using ":=". Parameters of type "Output" must be connected with "=>".

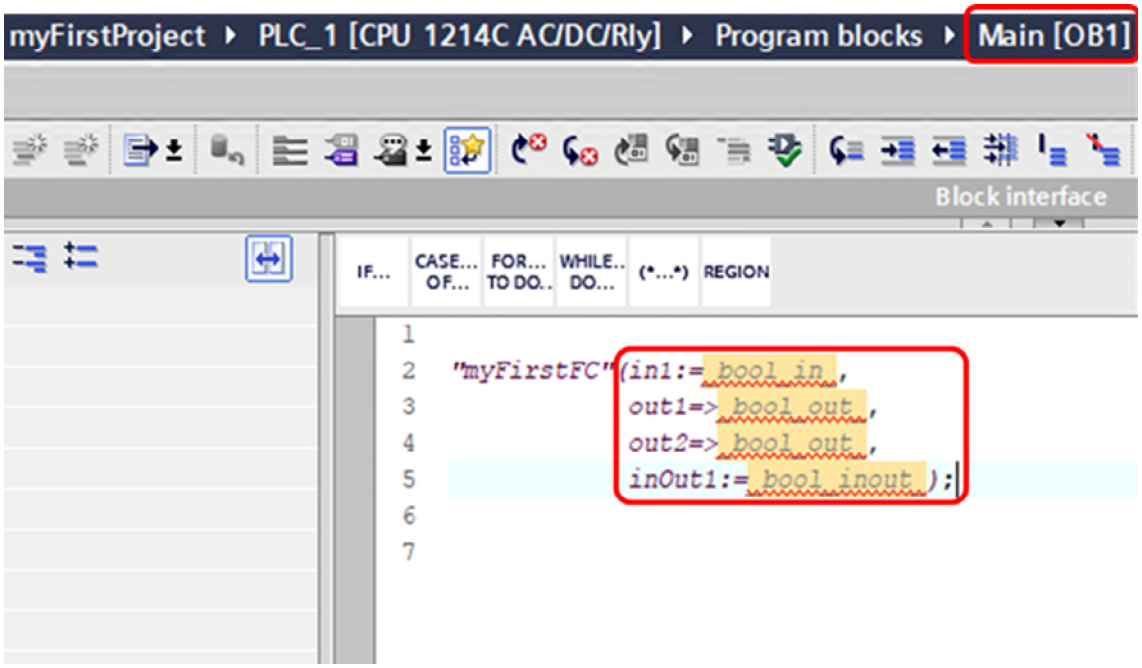
Example

In this screen, the following interface parameters have been declared in the "myFirstFC" function.



Picture 25 Function block interface of a function in the TIA Portal

After the "myFirstFC" function was created, it was called in "MAIN" (OB1). The parameters have not yet been transferred and the formal parameters to be connected are initially assigned placeholders. These placeholders provide information about the data type and parameter type (Input, Output, InOut).



Picture 26 Module interface in the TIA Portal

5.10 Calling a function block (FB) in ST / SCL

To process the function block, we call it in the program. The main difference between calling a function block (FB) is that an instance is assigned to it. The call is similar to a function (FC), but instead of the function block name, the name of the previously declared instance is used, followed by "()" and a semicolon. The parameters are passed in the round brackets and the semicolon concludes the instruction.

```
instMyFirstFB();
```

Picture 27 Syntax instance, without parameter transfer in SCL

Parameter transfer

If the called instance has interface parameters, these are displayed. It is not always necessary to pass parameters for function blocks, as the instance is already assigned a private memory area.

```
instMyFirstFB(in1 :=  var1,
              inOut1 := var2,
              out1 => var3,
              out2 => var4 );
```

Picture 28 Syntax instance, with parameter transfer in SCL

Call options

Depending on the type of calling function block, the instances can be located directly in the function block interface (= multi-instance in the TIA Portal) or stored as global instances (= single instance in the TIA Portal).

Instantiate function module multiple times

If an FB is called twice in the user program, two instances exist. Each instance is assigned its own memory area in which the instance can save its data throughout the cycle.

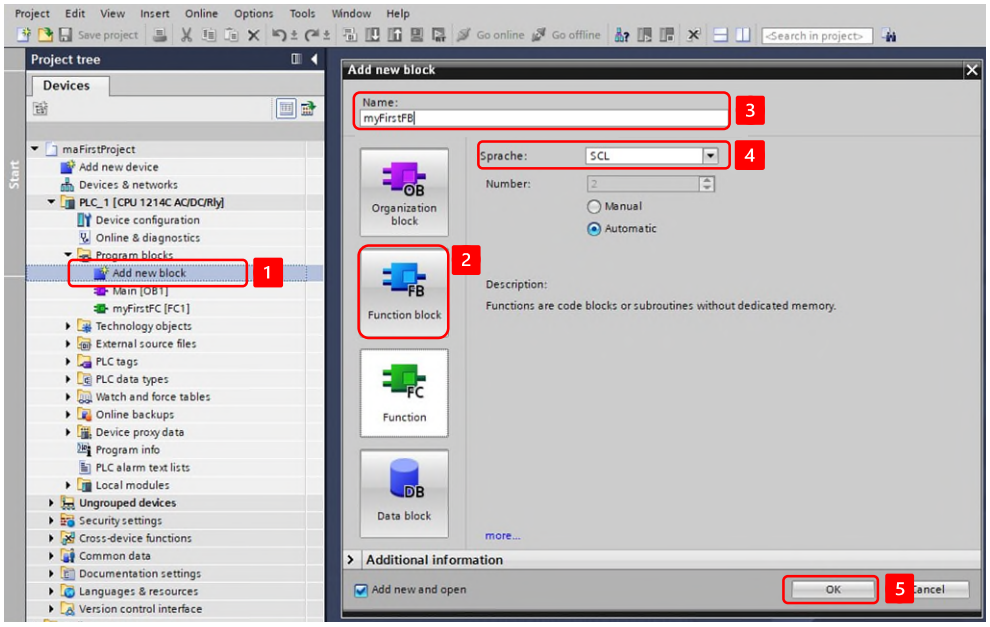
Example

For example, if the FB contains the user program for a calculation of switching operations, each call represents an instance that only uses states at the runtime of this call. A separate instance is required for each call.
This means that all data (information) relating to this calculation is available in this assigned instance.
Before an instance can be created, the assigned FB must already exist.
The variables of the respective instance can be observed in this instance.

5.10.1 Procedure Call with single instance

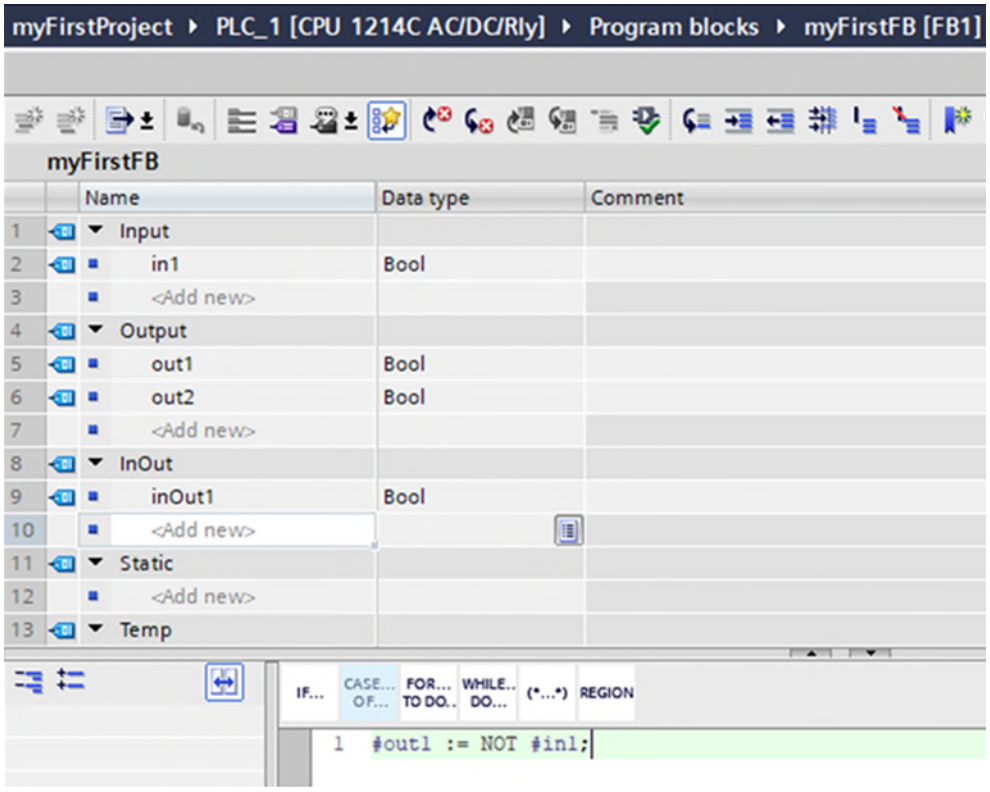
The procedure for calling up the "myFirstFB" function block twice is now explained step by step in the TIA portal.

1. Create the "myFirstFB" function module:



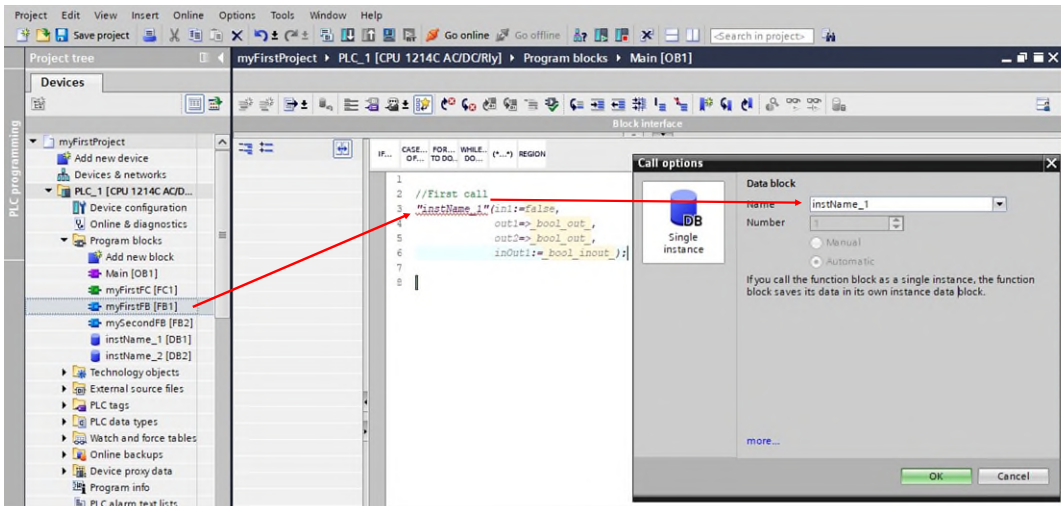
Picture 29 Adding a new block

2. Declaration of the function block interface and connection of the variables in the instruction part of the function block:



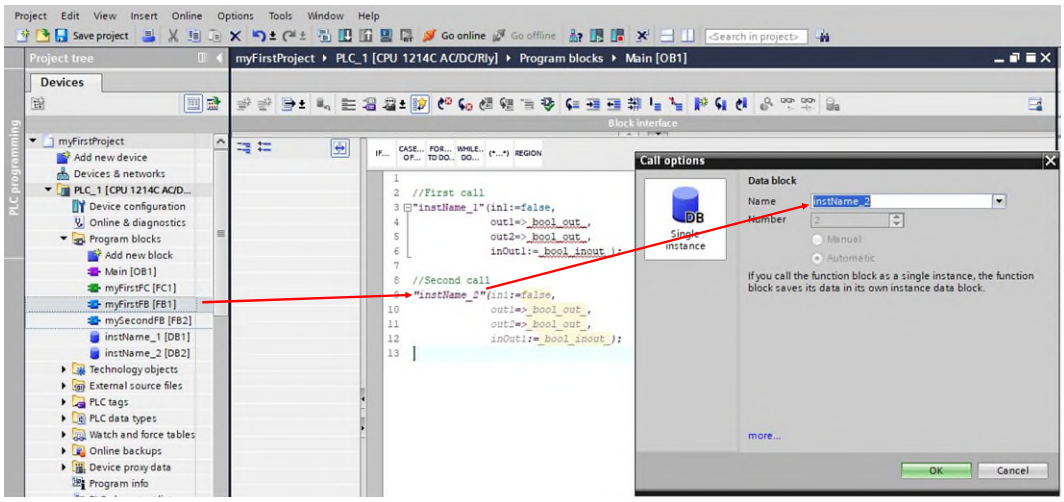
Picture 30 FB with function block interface in the TIA Portal

3. first call of "myFirstFB" in the "MAIN" OB using drag & drop and declaration of the first instance as a single instance:



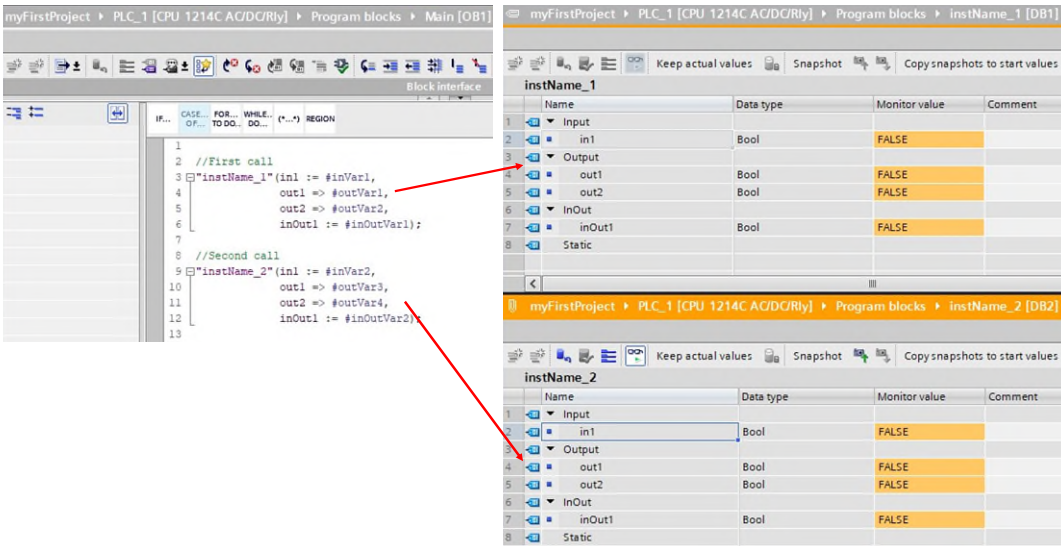
Picture 31 Instantiation of first block call

4. second call of "myFirstFB" in the OB "MAIN" using drag & drop and declaration of the second instance as a single instance:



Picture 32 Instantiation second block call

5. both instance data blocks can be observed:

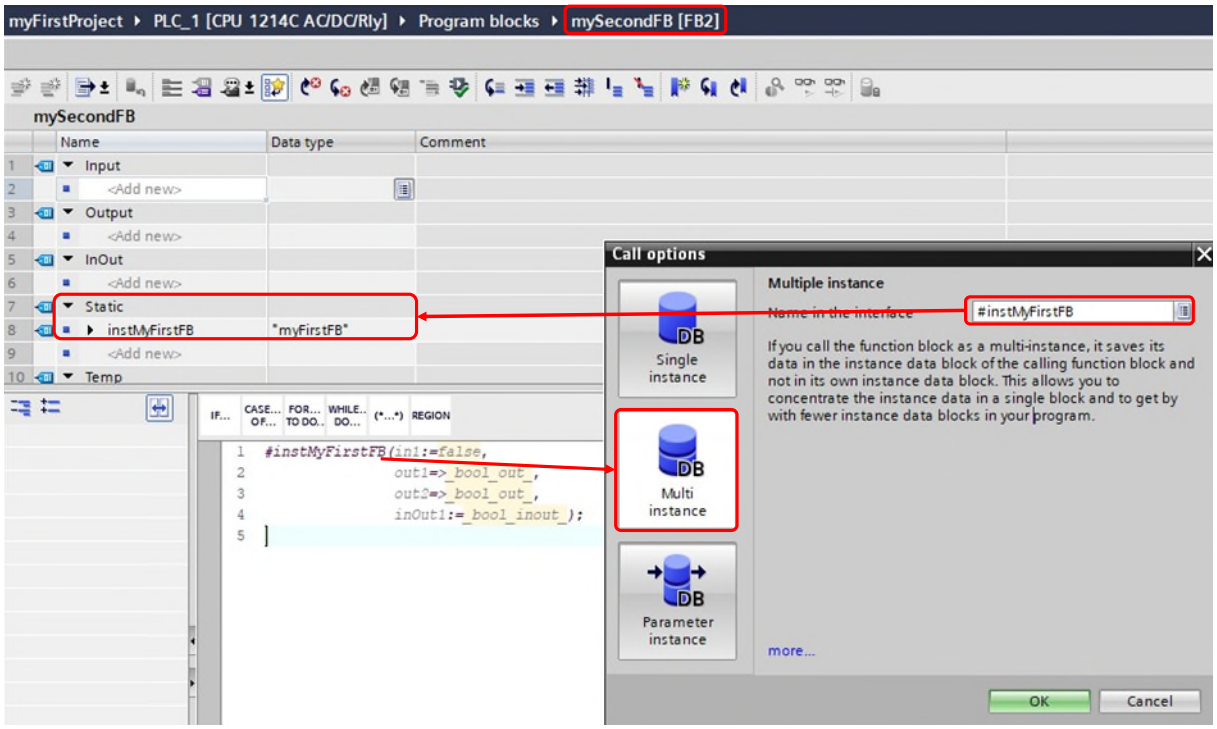


Picture 33 Actual values in the instances

Instance data blocks and the possibility of monitoring and controlling them are described in detail in the next chapter (Data blocks).

5.10.2 Call option as multi-instance (TIA portal)

If a function block is called in another function block, the multi-instance can also be selected in the call options.
The number of instance data blocks can be reduced by using multi-instances. When creating subroutines, the use of multi-instances is often absolutely necessary. For example, if you want to implement a runtime counter in a control module for a motor, each motor instance requires its own IEC counter instance. Multi-instances are placed in the block interface of the calling block.



Picture 34 Call options in the TIA Portal

5.10.3 Textual declaration as a multi-instance (CODESYS / Beckhoff)

The textual declaration of the instances is made according to the following scheme.

Syntax:

Instance name (variable name) : Block name (data type);

Example:

```
//Declaration in the function block interface
VAR
  instMyFirstFB : myFirstFB;   //declaration of the instance
END_VAR


//Implementation in the program
instMyFirstFB(in1:=       ,
              inOut1:=    ,
              out1=>      ,
              out2=>      );
```

Picture 35 Declaration and call of the instance

Instantiate function module multiple times

If an FB is called twice in the user program, two instances exist. Each instance is assigned its own memory area in which the instance can save its data throughout the cycle.

```
//Declaration in the function block interface
VAR
   instName_1 : myFirstFB;   //Instance 1
   instName_2 : myFirstFB;   //Instance 2
END_VAR



//Implementation in the program
//Call 1
instName_1(in1:=          ,
           inOut1:=       ,
           out1=>         ,
           out2=>         );

//Call 2
instName_2(in1:=          ,
           inOut1:=       ,
           out1=>         ,
           out2=>         );
```

Picture 36 Instantiation of two FBs