

Training document for the company-wide automation solution

Totally Integrated Automation (T I A)

MODULE B5

Structured programming with function blocks

This document was provided by Siemens A&D SCE (automation and drive technology, Siemens A&D Cooperates with Education) for training purposes. Siemens does not make any type of guarantee regarding its contents.

The passing on or duplication of this document, including the use and report of its contents, is only permitted within public and training facilities.

Exceptions require written permission by Siemens A&D SCE (Mr. Knust: E-Mail: michael.knust@hvr.siemens.de). Offences are subject to possible payment for damages caused. All rights are reserved for translation and any case of patenting or GM entry.

We thank the company Michael Dziallas Engineering and the instructors of vocational schools as well as further persons for the support with the production of the document.

		PAGE:
1.	Forward	4
2.	Notes for Structured Programming with FCs and FBs	6
3.	Generating Functions Blocks with Variable Declarations	8

The following symbols stand for the specified modules:



Information



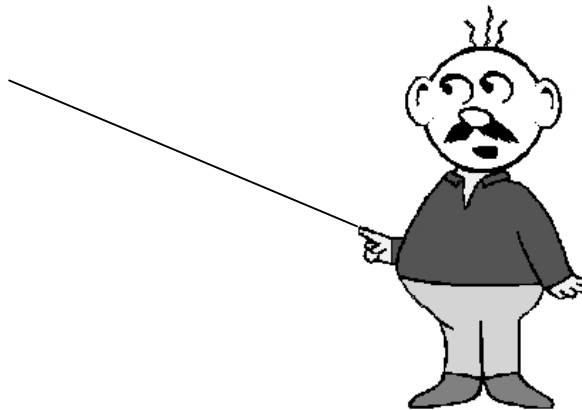
Programming



Example exercise

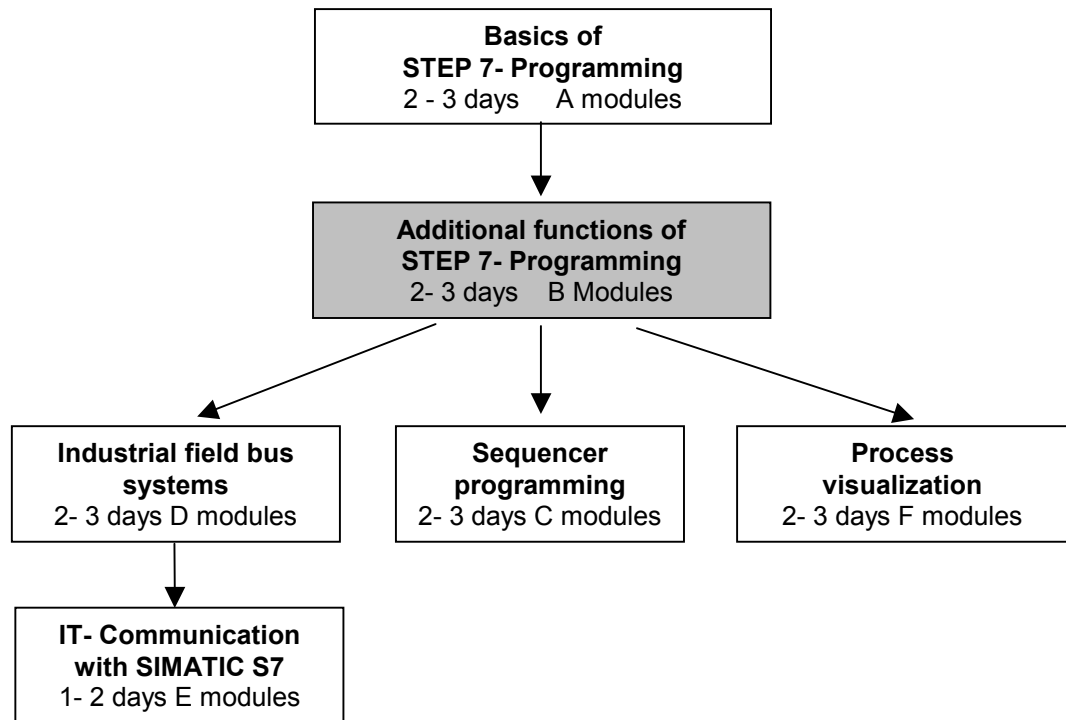


Notes



1. FORWARD

The Module B5 is assigned content wise to **Additional functions of STEP 7- Programming**.



Learning goal:

In this module, the reader should learn how a function block with internal variables is generated for structured programming.

- Generating a function block
- Defining internal variables
- Programming internal variables in a function block
- Calling and parameterizing of a function block in OB1

Requirements:

For the successful use of this module, the following knowledge is assumed:

- Knowledge in the use of Windows 95/98/2000/ME/NT4.0
- Basics of PLC- Programming with STEP 7 (e.g. Module A3 - 'Startup' PLC programming with STEP 7)
- Basics to structured programming (e.g. Appendix I - Basics to PLC –Programming with SIMATIC S7-300)

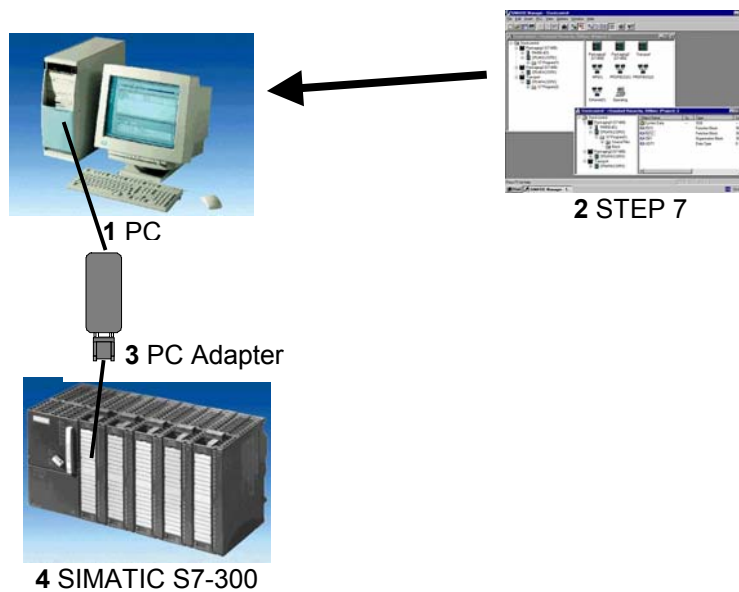
Forward	Notes	Function block with variable declaration
----------------	-------	--

Required hardware and software

- 1 PC, Operating system Windows 95/98/2000/ME/NT4.0 with
 - Minimal: 133MHz and 64MB RAM, approx. 65 MB free hard disk space
 - Optimal: 500MHz and 128MB RAM, approx. 65 MB free hard disk space
- 2 Software STEP 7 V 5.x
- 3 MPI- Interface for the PC (e.g. PC- Adapter)
- 4 PLC SIMATIC S7-300 with at least one digital in- and output module. The inputs must be lead through a functional unit.

Example configuration:

- Power supply: PS 307 2A
- CPU: CPU 314
- Digital input: DI 16x DC24V
- Digital output: DO 16x DC24V / 0.5 A



2. NOTES FOR STRUCTURED PROGRAMMING WITH FCS AND FBS



The program execution is written in blocks in STEP 7. The organization block OB1 is already available.

The program execution describes the interface to the operation system of the CPU and is called automatically from this block and executed cyclically.

By extensive control tasks, one cuts the program into small, manageable and ordered program blocks in functions.

These blocks are then called from the organization block over the block call instructions (Call xx / UC xx / CC xx). If the block end was realized, the program executes further in the previously called block call.

For structured programming, STEP 7 offers the following:

- FB (Function block):

The FB has an assigned storage area. If a FB is called, it can be assigned a data block (DB).

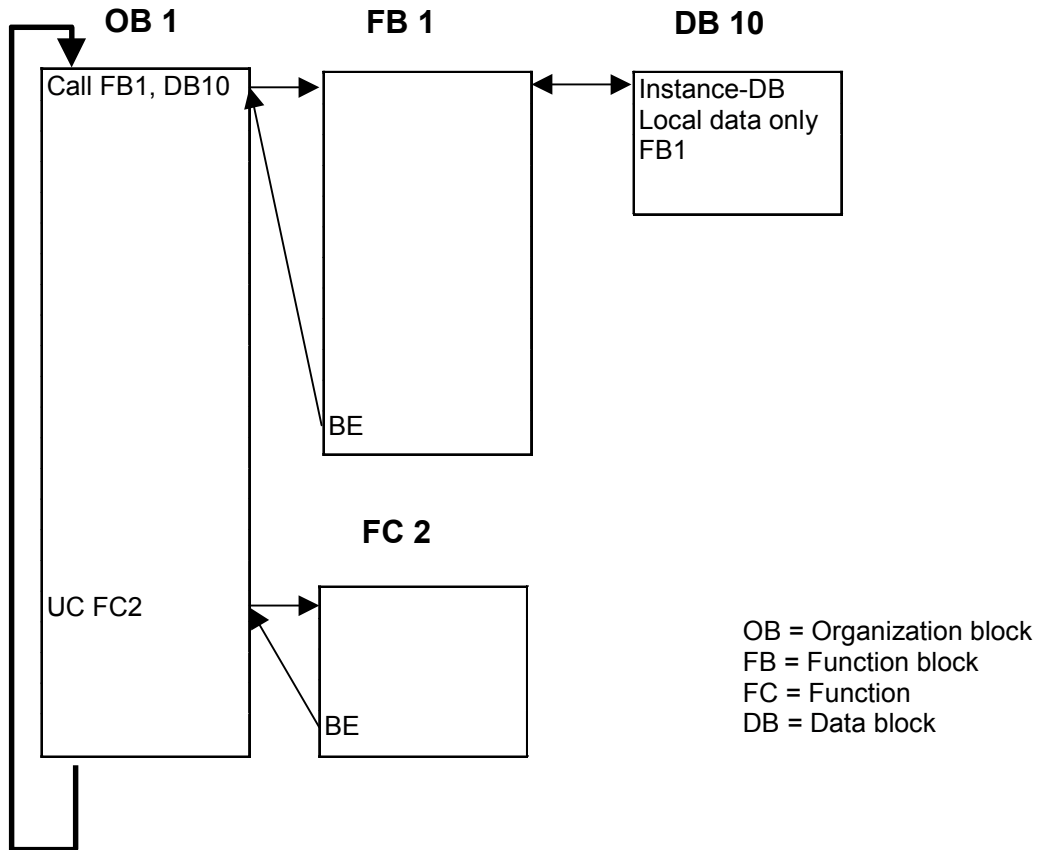
From the data in this instance, the DB can be accessed by a call from the FB. A FB can be assigned different DBs. Further FBs and FCs can also be called over block call instructions in a function.

- FC (Function):

A FC does not possess an assigned storage area. The local data of a function is lost after the editing of the function. Further FBs and FCs can be called over block call instructions in a function.



The structure of a program can look as follows:



Note: In order to use the blocks, they must first be generated. There is also a possibility to program these FCs and FBs in the form of standard blocks under the use of internal variables. Then any function can be called often, whereas another local instance DB must access a FB each time.

3. GENERATING A FUNCTION BLOCK WITH VARIABLE DECLARATION



When blocks are generated with STEP 7, the quasi as a “Black-Box“ in any program functions must be programmed under assignment from variables. Therefore the rules apply, that in these blocks, no absolute addressed In/Outputs, memory bits, timers, counters, etc. are allowed to be used. Single variables and constants come here to be assigned.



In the following example, a function block with variable declaration is to be provided which contains a band control and additionally another cycle counter.

Therefore the band motor is activated with the button ‘S0’ and deactivated with the button ‘S1’.
The traversing program cycles should be counted to a memory bit double word.
The example refers to the displayed addresses:

Inputs:

- In-Button S0 = I 0.0
- Out-Button S1 = I 0.1

Outputs:

- Band motor = Q 4.0

Memory bits:

- Cycle counter = MD20



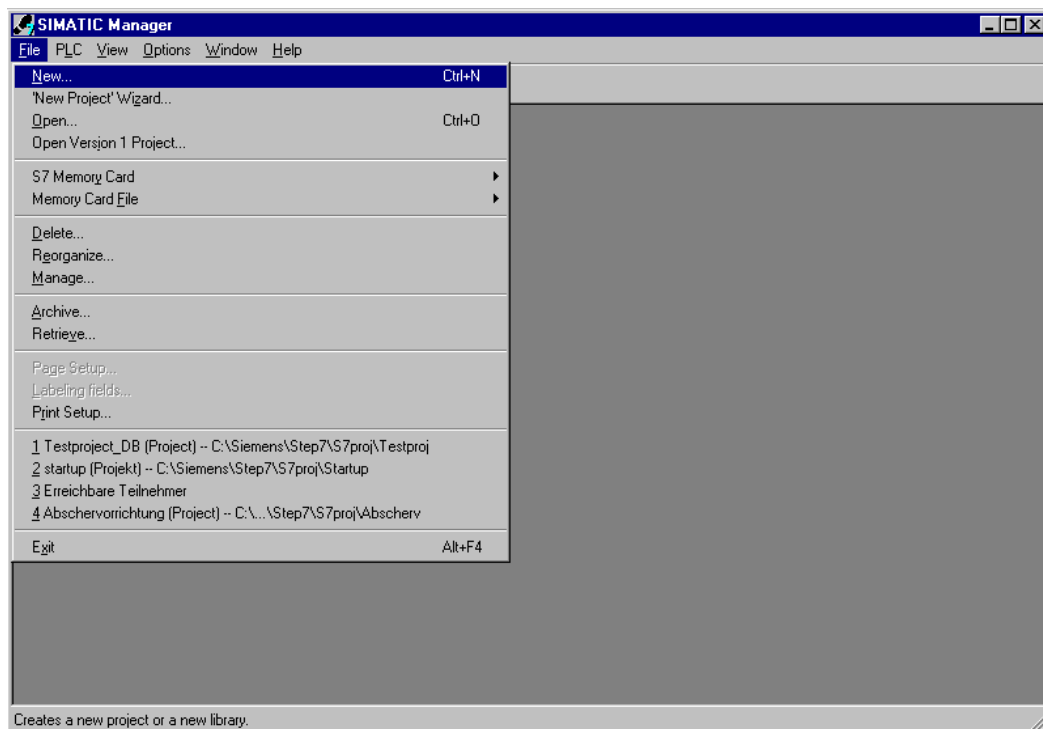
To create this program example, the following steps must be accomplished (with the production of a hardware configuration):

1. Open **SIMATIC Manager** with a double click (→ SIMATIC Manager).



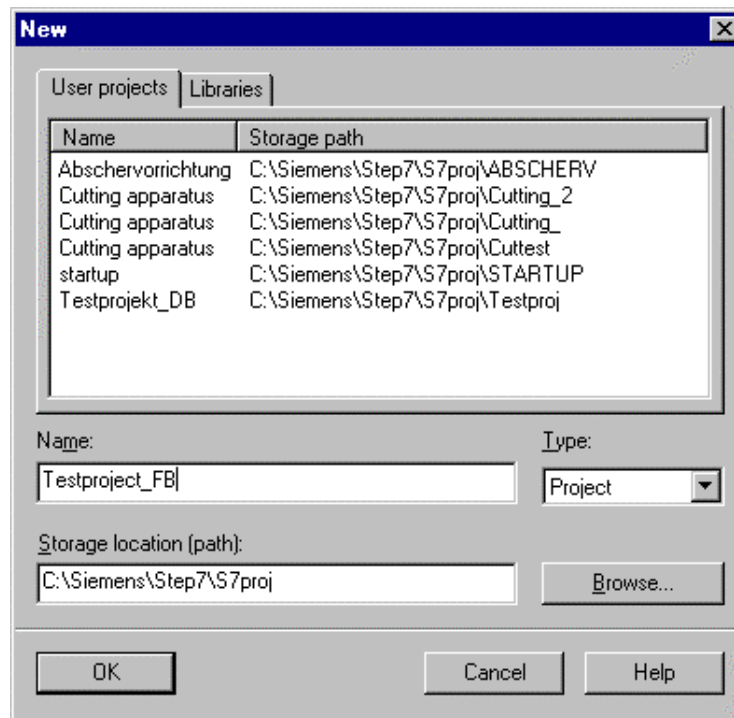
SIMATIC Manager

2. Create a new project (→ File → New)

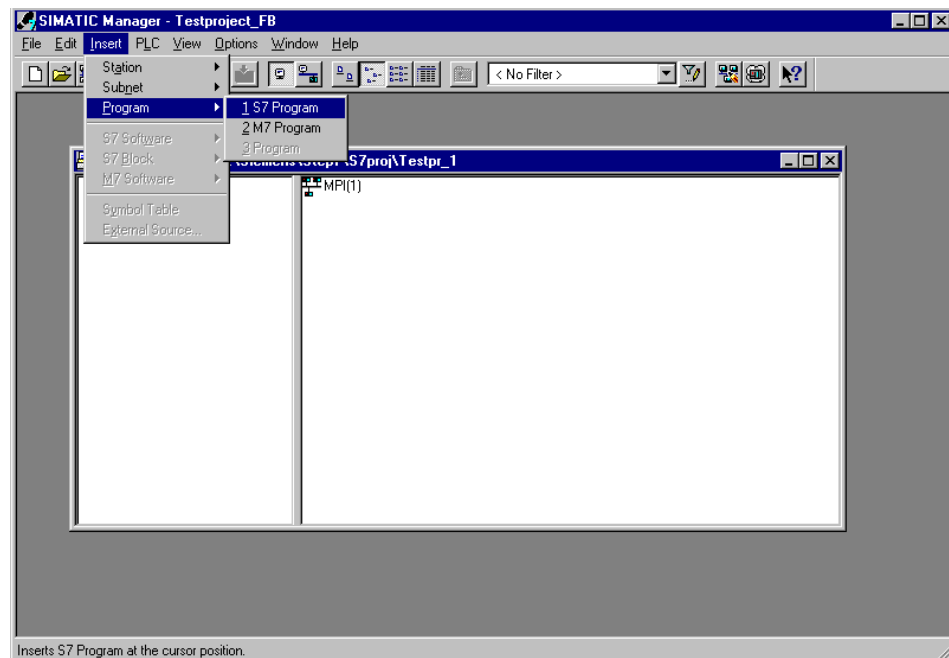




3. Generate a new project, allocate the project with a name **Testproject_FB** (→ Testproject_FB)

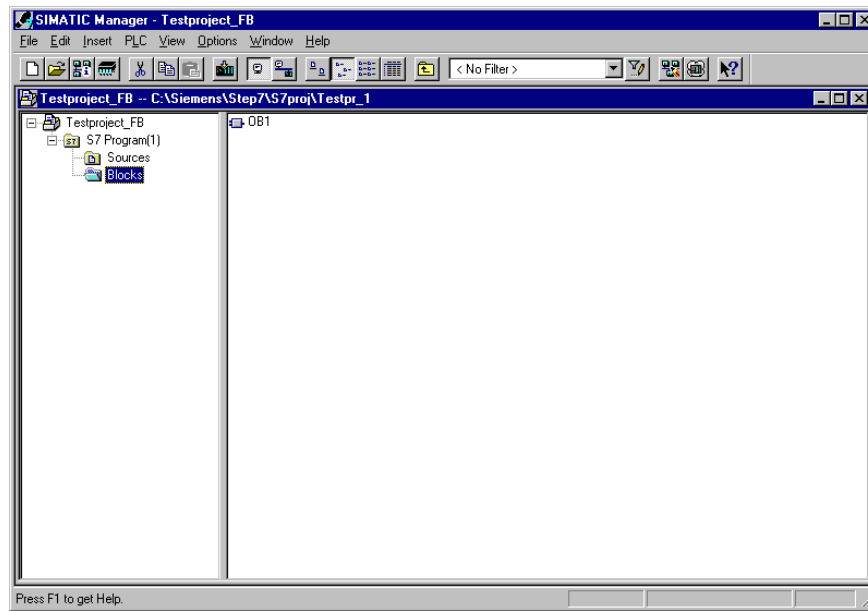


4. Insert a new **S7-Program** (→ Insert → Program → S7-Program).

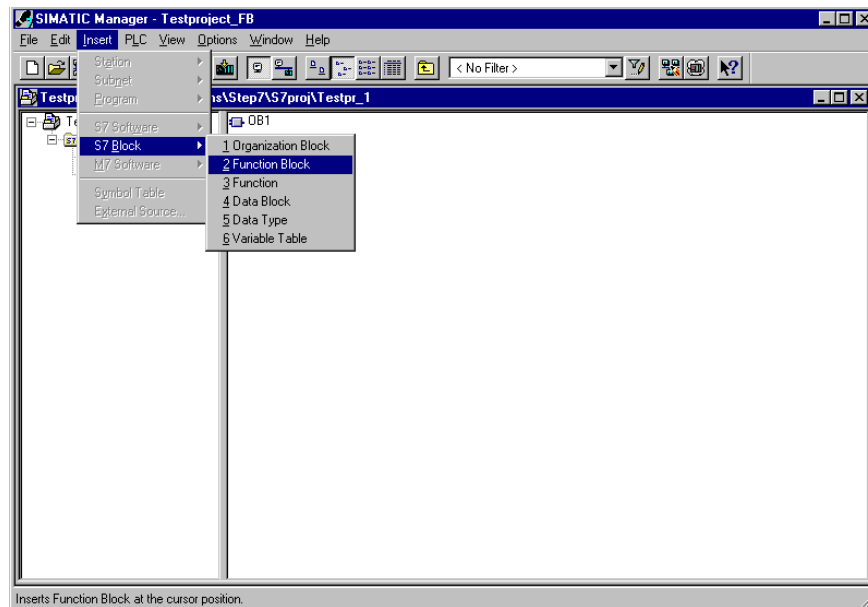




5. Highlight the folder **Blocks** (→ Blocks).



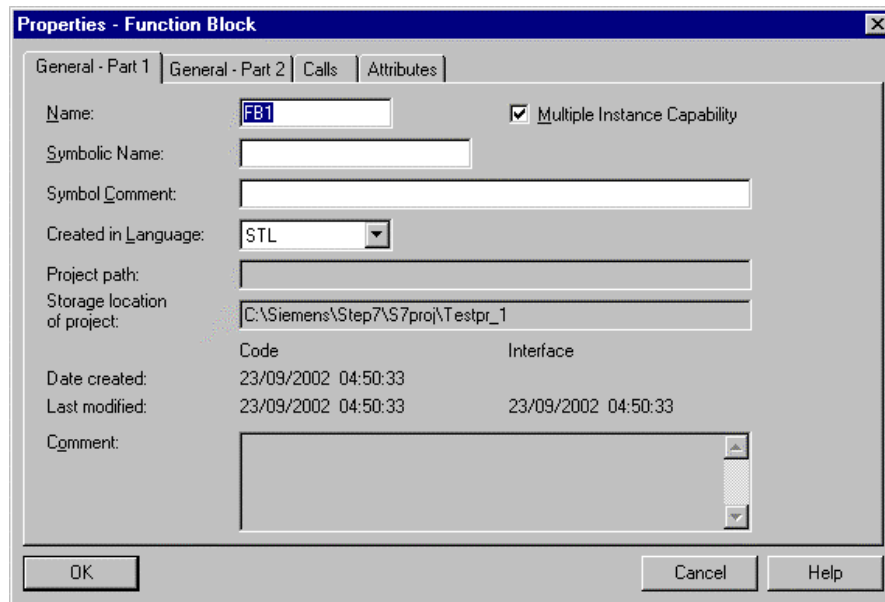
6. Insert a **Function block** (→ Insert → S7 Block → Function block).



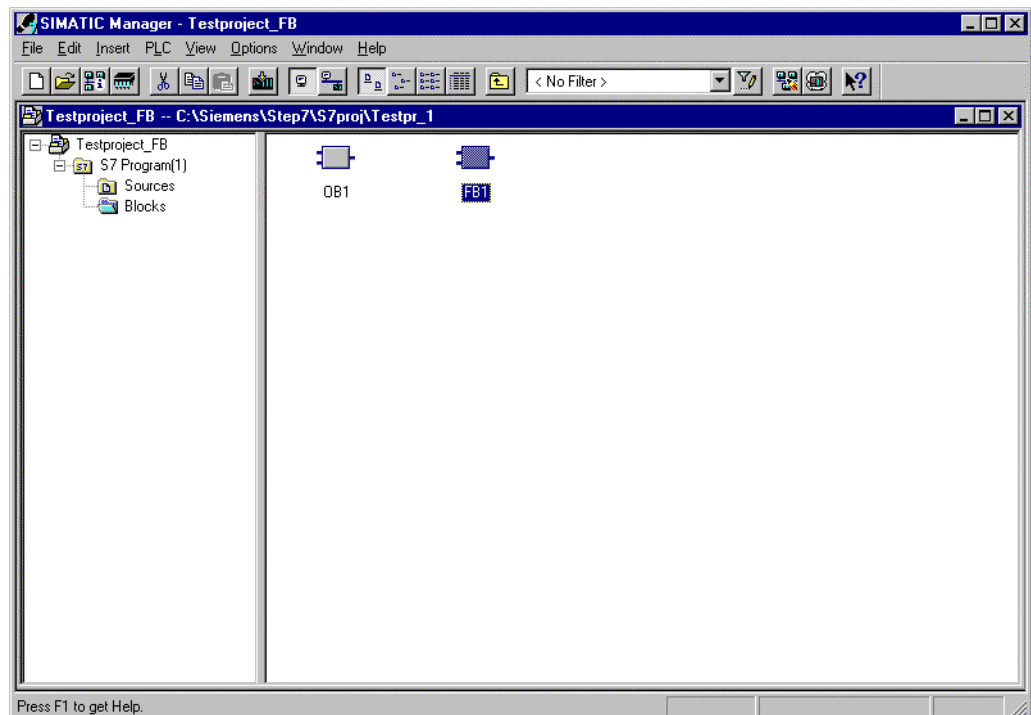
Forward	Notes	Function block with variable declaration
---------	-------	---



7. Enter the name of **FB1** for the FB and click on **OK** (→ FB1 → OK).



8. Open function block **FB1** with a double click. (→ FB1)





9. With **LAD, STL, FBD: Program blocks**, you now have an editor which gives you the possibility to edit your functions.

In addition, the variables should be defined and specified in the variable declarations table, which is displayed in the FB1.

These variables are type 'in', 'out', 'in_out', 'stat' and 'temp'.

Input parameters (IN) only in FBs, FCs, SFBs and SFCs

With help of the input parameters, data is assigned for the processing of the block.

Output parameters (OUT) only in FBs, FCs, SFBs and SFCs

With the output parameters, the results are assigned to the called block.

In/Out parameters (IN_OUT) only in FBs, FCs, SFBs and SFCs

With the in/out parameters, data is assigned to the called block, processed and files the results from the called block into the same variables.

Statistical data (STAT) only in FBs and SFBs

Statistical data is the local data of a function block that is saved in an instance data block and therefore remains preserved until the next processing of the function block.

Temporary data (TEMP) in all blocks

Temporary data is local data of a block that is filed during the processing of a block into the local data stack (L-Stack) and is no longer available after processing.



Note: Here the difference between FB/SFB and FC/SFC is stated. In a FC, there are no statistical variables (**stat**) to regulate because there is no memory for the contents of the variable contents after the processing of the FC. In the FB, these statistical variables are buffer stored in the corresponding local instance DB until the next processing of the FB.

Out of this principle, only the FB is suited for the creation of programs in which data like e.g. step memory bits over more program cycles should remain stored away.



This stipulation of the variables follows by the first name given. The data type is specified and an optional initial value and comment are entered. This example appears as follows:

Address	Declaration	Name	Type	Initial value	Comment
0.0	in	On	BOOL	FALSE	Motor on
0.1	in	Off	BOOL	FALSE	Motor off
2.0	out	Motor	BOOL	FALSE	Band motor
4.0	in_out	Cycle	DINT	L#0	Cycle counter
8.0	stat	Mbl	BOOL	FALSE	Temporary counter

Declaration-
Column specifies the
type of variable.

Initial value to which
the data type must be
compatible
(optional).

Comment to
documentation
(optional).

The absolute address is
created automatically from
STEP 7.
The address format is
BYTE, BIT.

Symbolic name is
referenced with the
absolute address. Over
this address, the variable
can be accessed.

Chosen data
type (see below)
for your data
element.



Note: In the declaration, one of each chosen stationary variable type is displayed. Also displayed by FCs are variables from type 'in', 'out', 'in_out' and 'temp' and by FBs, variables from type 'in', 'out', 'in_out', 'stat' and 'temp'. If a further variable from a particular type is required, then one must click on the variable row in the last column (Column), and then hit **<Enter>**. Then an empty row with this variable type appears.







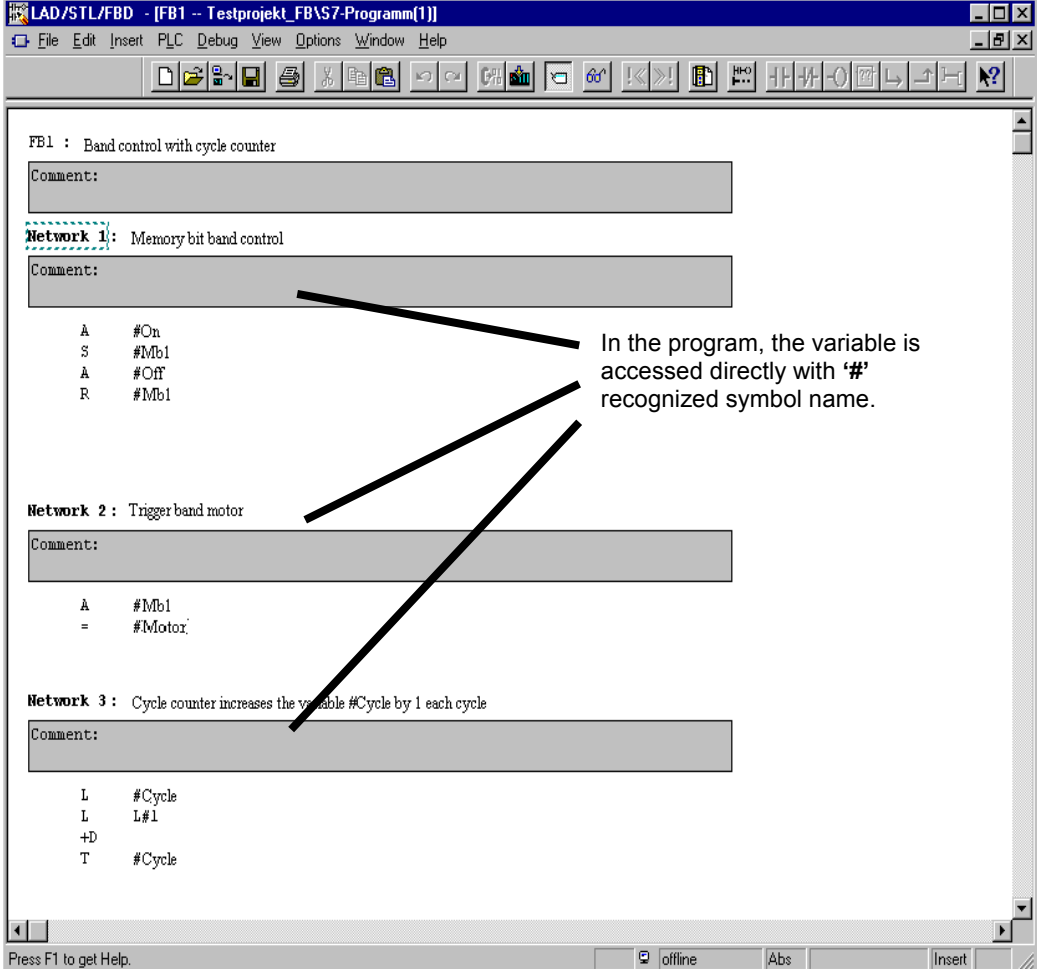
Data in a data block must be determined through data types.
The following standard- data types are defined in the S7 below :

Type and description	Size in Bits	Format-options	Range and number notation (lowest to highest value)	Example
BOOL (Bit)	1	Boolean-Text	TRUE/FALSE	TRUE
BYTE (Byte)	8	Hexadecimal number	B#16#0 to B#16#FF	B#16#10
WORD (Word)	16	Binary number	2#0 to 2#1111_1111_1111_1111	2#0001_0000_0000_0000
		Hexadecimal number	W#16#0 to W#16#FFFF	W#16#1000
		BCD	C#0 to C#999	C#998
		Decimal number unsigned	B#(0,0) to B#(255,255)	B#(10,20)
DWORD (Double word)	32	Binary number	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111	2#1000_0001_0001_1000_1011_1011_0111_1111
		Hexadecimal number	DW#16#0000_0000 to DW#16#FFFF_FFFF	DW#16#00A2_1234
		Decimal number unsigned	B#(0,0,0,0) to B#(255,255,255,255)	B#(1,14,100,120)
INT (Integer)	16	Decimal number signed	-32768 to 32767	1
DINT (Int,32 bit)	32	Decimal number signed	L#-2147483648 to L#2147483647	L#1
REAL (Floating-point number)	32	IEEE floating-point number	Upper limit: +/-3.402823e+38 Lower limit: +/-1.175495e-38	1.234567e+13
S5TIME (Simatic-Time)	16	S7-Time in steps of 10 ms	S5T#0H_0M_0S_10MS to S5T#2H_46M_30S_0MS and S5T#0H_0M_0S_0MS	S5T#0H_1M_0S_0MS S5TIME#1H_1M_0S_0MS
TIME (IEC-Date)	32	IEC-Time in steps from 1ms, integer signed	-T#24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS	T#0D_1H_1M_0S_0MS TIME#0D_1H_1M_0S_0MS
DATE (IEC-Date)	16	IEC-Date in steps of 1 Tag	D#1990-1-1 to D#2168-12-31	DATE#1994-3-15
TIME_OF_DAY (Time)	32	Time in steps of 1ms	TOD#0:0:0.0 to TOD#23:59:59.999	TIME_OF_DAY#1:10:3.3
CHAR (Character)	8	ASCII-Characters	'A', 'B' etc.	'B'

Forward	Notes	Function block with variable declaration
---------	-------	---



10. Now the program can be entered by the use of variable names. (Variables are recognized with the symbol '#'). These variables can be seen in the following example in STL. The function block FB1 should be saved  and downloaded into the CPU . The mode switch of the CPU must be on STOP! (→  → )



FB1 : Band control with cycle counter

Comment:

Network 1: Memory bit band control

Comment:

```

A   #On
S   #Mb1
A   #Off
R   #Mb1
    
```

Network 2: Trigger band motor

Comment:

```

A   #Mb1
=   #Motor
    
```

Network 3: Cycle counter increases the variable #Cycle by 1 each cycle

Comment:

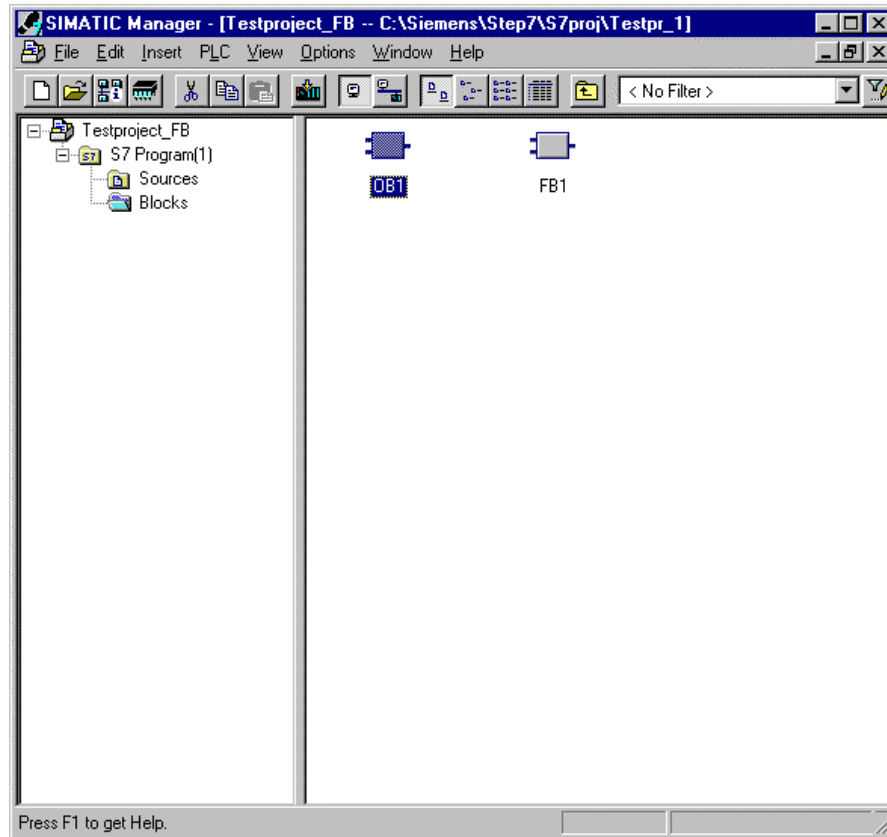
```

L   #Cycle
L   L#1
+D
T   #Cycle
    
```

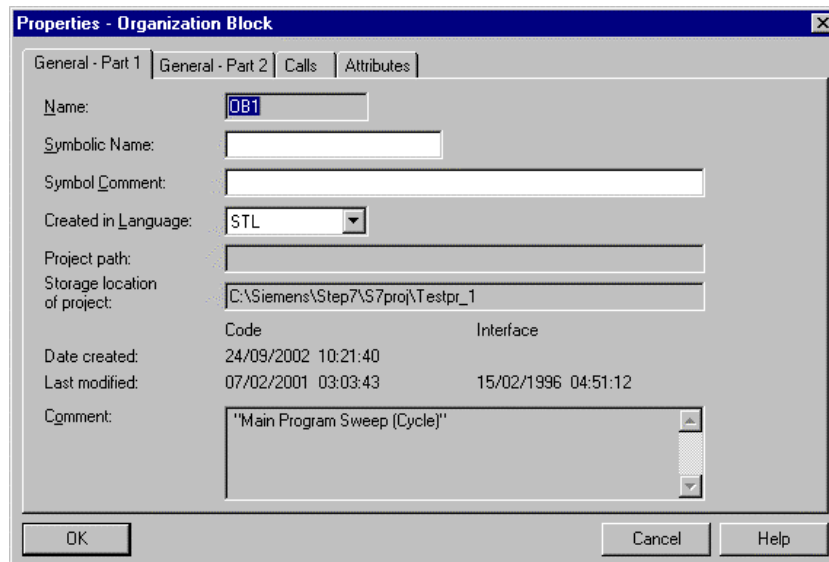
In the program, the variable is accessed directly with '#' recognized symbol name.



- In **SIMATIC Manager**, only the **OB1** is opened in order to program the call of the FB1 (→ OB1).



- Accept the setting with a click on **OK** (→ OK).

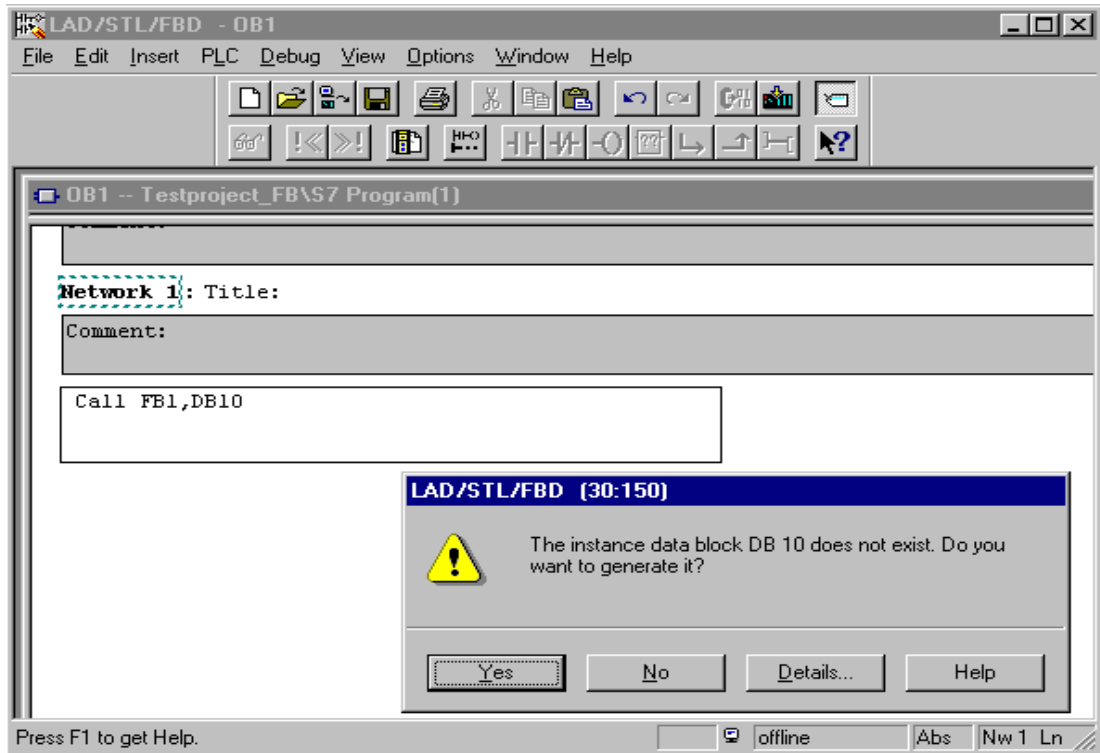




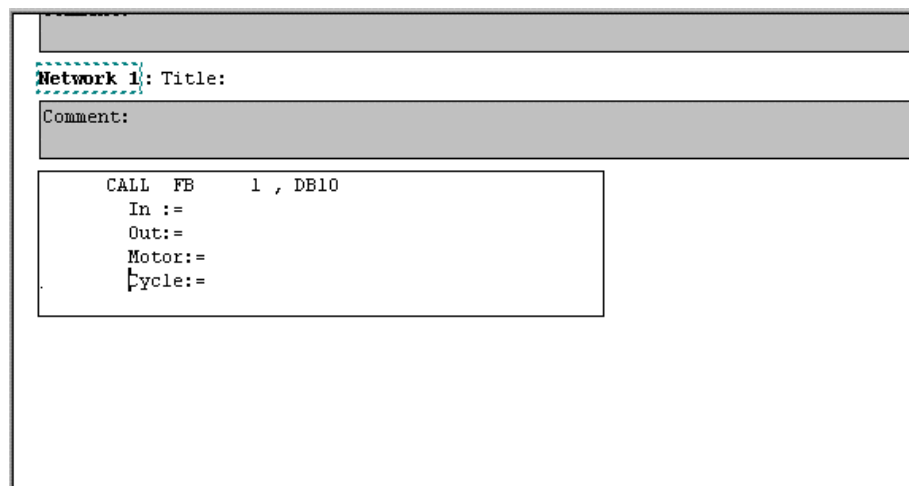
13. With 'LAD, STL, FBD: Program blocks', you now have an editor that gives you the possibility to generate your OB1. The FB1 should be called together with its associated instance DB (also called local DB) with the following instruction line.

CALL FB1,DB10 <Enter>





Therefore, the instance DB (DB10) can automatically be generated when the question is answered with **Yes** (→ Call FB1,DB10 → Yes).

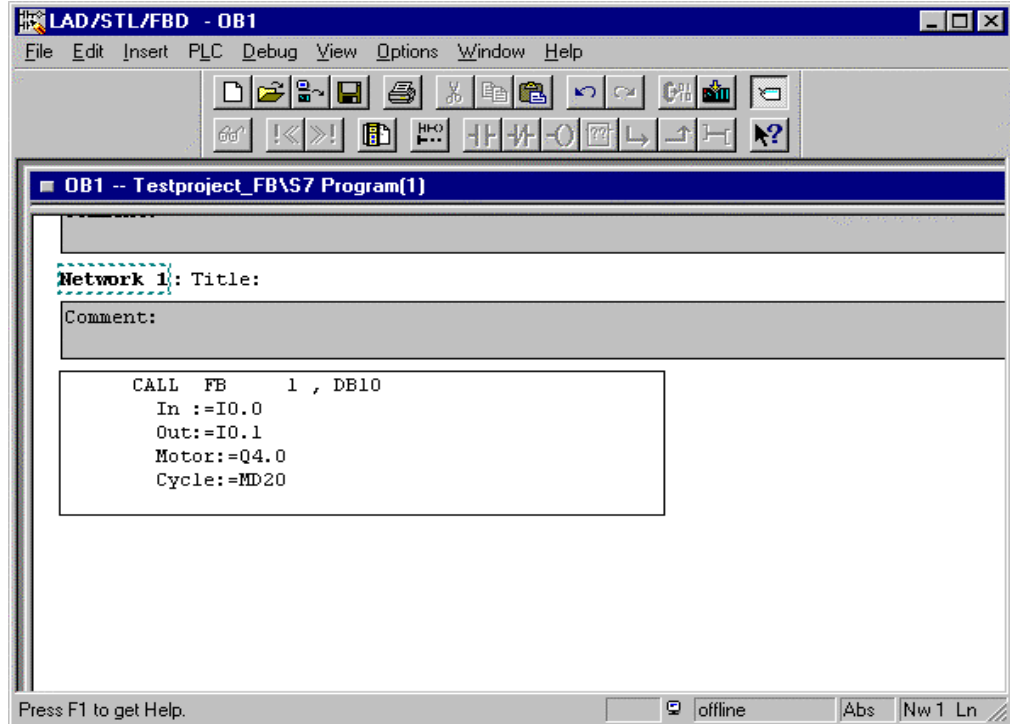


14. Then all variables from type 'in', 'out' and 'in_out' are displayed, so that these variables can be assigned actual parameters (e.g.: I 0.0, MW2 etc ...).







15. In our example, the allocation follows as shown. If the allocation is as follows, the organization block OB1 can be saved  and downloaded . The mode switch of the CPU must be on STOP! (→  → )



Note: On this type, the FB1 can be called several times between the indication of different data blocks and in/output addresses. Thus it represents a standard block for this special setting of tasks.



16. Now in 'SIMATIC Manager', the instance DB (local DB) 'DB10' is chosen and downloaded into the CPU . The mode switch of the CPU must be on STOP!(→ DB10 → )



17. By switching the mode switch to RUN the program is started. The motor switches on when switch I0.0 is activated. It is switched off, as the switch I0.1 is activated. In the memory bit MD20, how often the FB1 from the OB1 is called, is taken into account. The memory bits get a feeling for the cycle time of the OB1. This happens with a high frequency, since the program cycle is very short in the OB1.