Training document for the company-wide automation solution Totally Integrated Automation (T I A)

MODULE A3

'Startup' PLC- Programming with STEP 7

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**	5
2.		**Notes for the Programming of SIMATIC S7-300 with STEP 7**	7
2.1		Automation system SIMATIC S7-300	7
2.2		Program software STEP 7	7
3.		**Installation of the STEP 7 Software**	8
4.		**Program Interface Adjustment (PC- Adapter)**	9
5.		**What is a PLC and what are PLCs used for?**	14
5.1		What is the concept of a PLC?	14
5.2		How does a PLC drive a process?	14
5.3		From where does a PLC get information about the state of a process?	15
5.4		Where does the difference between a Normally Open (NO) and Closed (NC) Contact lie?	15
5.5		How does a PLC communicate with In/Output signals?	16
5.6		How does a program work in a PLC?	17
5.7		How do logic operations in a PLC- Program appear?	18
5.7.1		AND- Logic operation	18
5.7.2		OR- Logic operation	20
5.7.3		Negation	21
5.8		How is a PLC- Program generated ? How does it arrive in the memory of the PLC?	22
6.		**Assembly and Operation of the SIMATIC S7-300**	23
7.		**Example Exercise**	26
8.		**STEP 7- Project Application**	27
9.		**STEP 7- Program Writing in Function Diagram FBD**	35
10.		**STEP 7- Program Debugging in the CPU**	47

The following symbols stand for the specified modules:

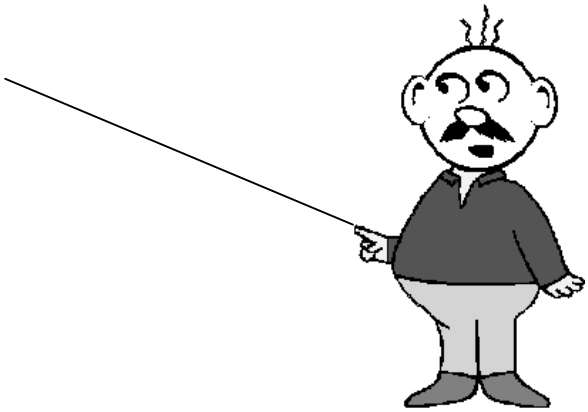
Information

Installation

Programming

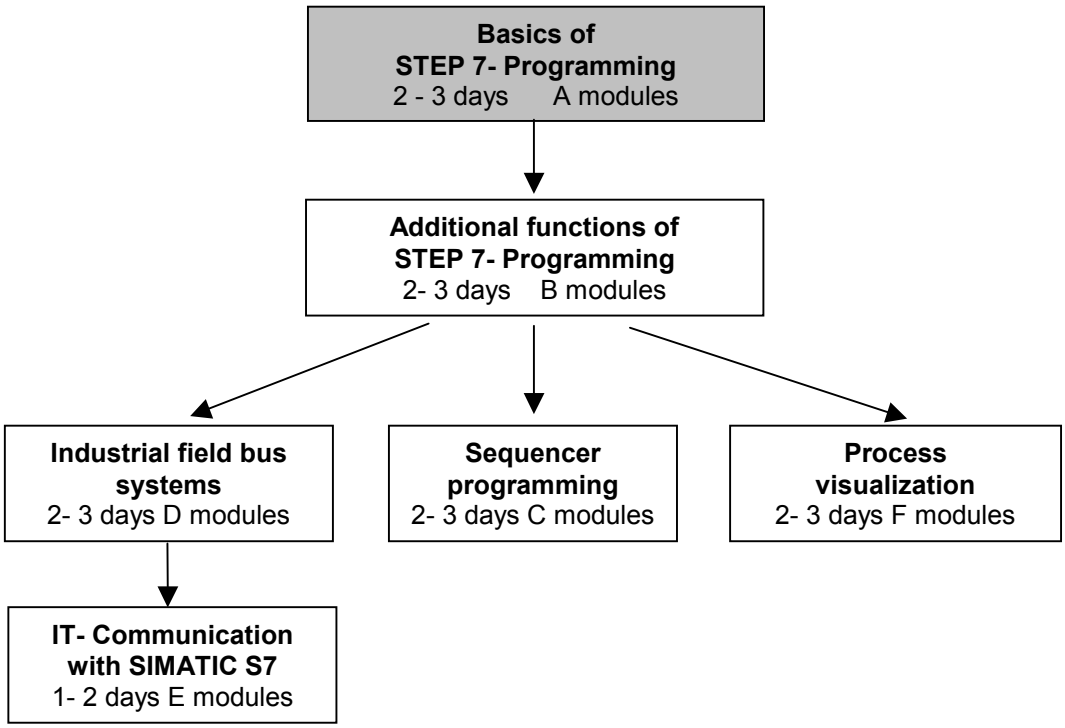
Example exercise

Notes



1. FORWARD

The module A3 is assigned content wise to the **Basics of STEP 7- Programming** and represents a quick start in the STEP-7 Programming.



Learning goal:

In this module, the reader will learn about the programming of a programmable logic controller (PLC) with the programming tool STEP 7. The module arranges the basics and shows the procedure in the following steps by means of a detailed example.

- Installation of software and the modification of a program interface
- Explanation of what a PLC is and how it works
- Structure and operation of a PLC SIMATIC S7-300
- Compilation of an example program
- Loading and debugging of an example program

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

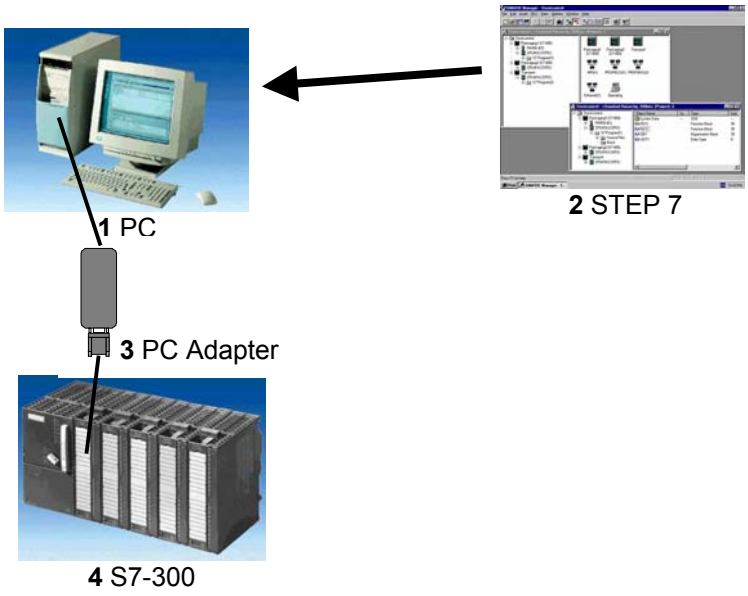
Required hardware and software

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 a minimum of one digital In- and Output device. The inputs must be led out of a switch bay.
Example configuration:
- Power supply: PS 307 2A
- CPU: CPU 314
- Digital inputs: DI 16x DC24V
- Digital outputs: DO 16x DC24V / 0.5 A



2. NOTES FOR THE PROGRAMMING OF SIMATIC S7-300 WITH STEP 7

2.1 AUTOMATION SYSTEM SIMATIC S7-300

The automation system SIMATIC S7-300 is the modular miniature control system for the low and medium power ranges.
There is a comprehensive module spectrum for the optimal adjustment in the automation task.
The S7-Controller consists of a power supply (PS), a central processing unit (CPU) and signal modules for in and/or output devices (I/O devices). If necessary communication processors (CPs) and function modules (FMs) can be used for specific tasks (e.g. stepping motor control).
The programmable logic controller (PLC) supervises and controls a machine or a process in conjunction with an S7 program. The I/O devices are addressed in the S7-Program via the Input (I) and Output addresses (Q).

The system is programmed with the software STEP 7.

2.2 PROGRAM SOFTWARE STEP 7

The software STEP 7 is the program tool for the automation systems

- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7, the following functions can be used for the automation construction:

- Configuring and parameterization of hardware
- Generation of a user program
- Debug, commissioning, and service
- Documentation, archiving
- Operation-/Diagnostic functions

All functions are supported and described through elaborate online help documentation.

3. INSTALLATION OF THE STEP 7 SOFTWARE

STEP 7 has three different versions:

- **STEP 7 Professional version**, which can use all applications provided by STEP 7(S7- GRAPH or S7- PLCSIM). This software package must be authorized.
- **STEP 7 Software for Students** contains the option package S7- PLCSIM. This software package must be authorized and then it can be used for 120 days.
- **STEP 7 Mini** is a restricted version that does not need to be authorized, however, no further option packages can be used (e.g. S7- PLCSIM or S7- GRAPH).

STEP 7 comes on a CD-ROM, which contains the software. With the CD-ROM comes a floppy disk, which contains the authorization of the software. After the data from this disk is transferred to the PC, STEP 7 professional version can be used by the operator.
This authorization disk can also be used on another PC or can be copied in order to authorize the software. For the topic of installation and transmission of authorization, please refer to module A2 (Installation of STEP 7 V5.x/handling of authorization).

To install STEP 7, please proceed to the following steps.

1. Place the STEP 7 CD in the CD- ROM drive.
2. The setup program should start automatically. If not, it can be started by double clicking on the **setup.exe** executable file on the CD. The setup program will guide you through the whole installation process of the STEP 7 software.
3. In order to use the professional or student version of STEP 7, the software must be authorized on your computer. The files from the authorization disk must be transferred onto the PC. This process will execute at the end of the software installation. A dialog window will appear and ask you if you would like to authorize the software. If **Yes** is selected, the authorization disk must be inserted in order to transfer the proper files to the PC.

4. PROGRAM INTERFACE ADJUSTMENT (PC- ADAPTER)

In order to program a SIMATIC S7-300 from the PC or PG, an MPI-Connection is needed. MPI stands for **M**ulti **P**oint **I**nterface and is a communication interface that has connections for up to 32 devices(e.g. PCs,HMI systems, etc.). It is used with HMI (Human Machine Interface) systems to program, serve and observe data exchange between SIMATIC S7 CPUs.

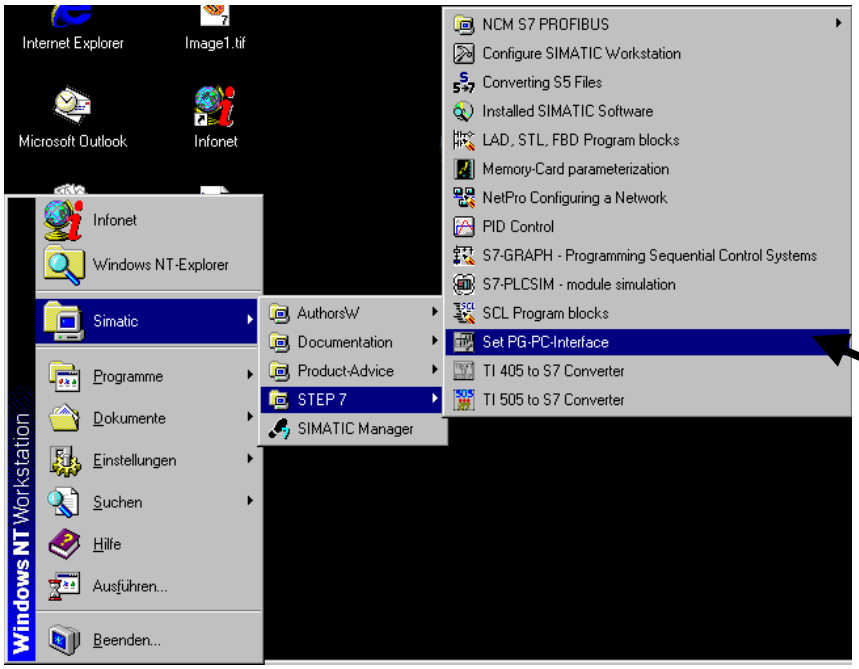
Each SIMATIC S7-300 possesses an integrated interface.

There are many possible ways to attach a PC or laptop to an MPI:

- Integrated ISA- Communication processors for the PG
- ISA- Communication processors for the PC (e.g. MPI-ISA- Card)
- PCI- Communication processors for the PC (e.g. CP5611)
- PCMCIA- Communication processors for the laptop (e.g. CP5511)
- Adapter for the communication over the serial interface of a PC or laptop (e.g. PC-Adapter)

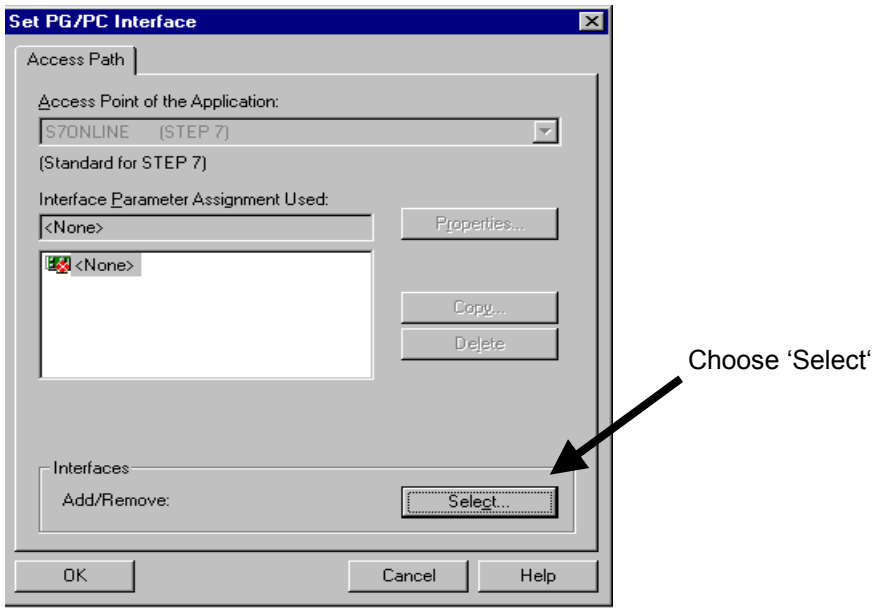
The following steps below describe the calibration and parameters of a PC-Adapter for a PC.

1. Call **Set PG-PC-Interface**. (→ Start → SIMATIC → STEP 7 → Set PG-PC-Interface)

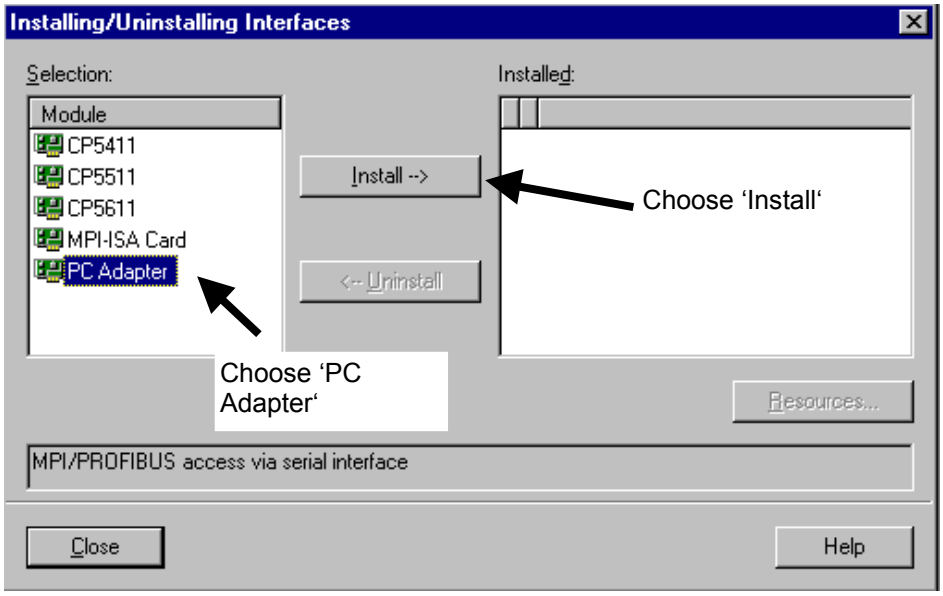


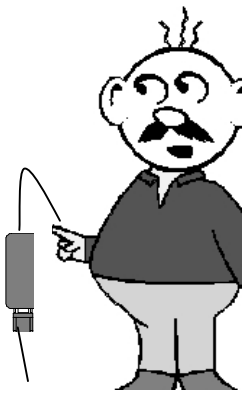
Click 'Set PG-PC-Interface'!

2. The module **Select** is available as the MPI-interface. (→ Select)

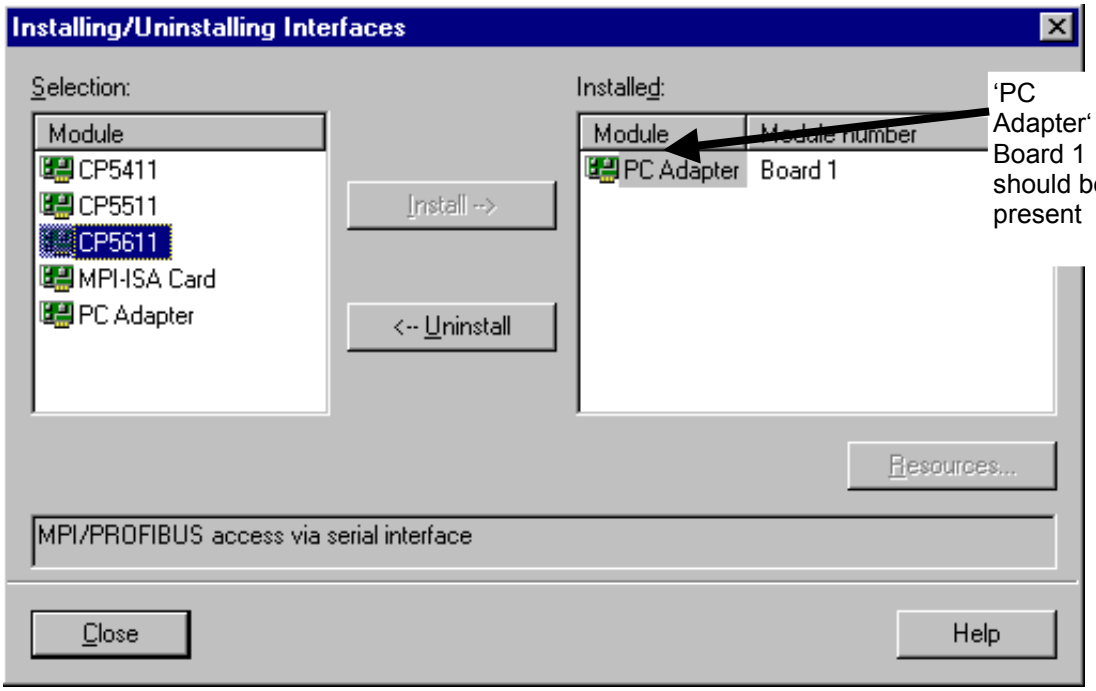


3. Select the desired module e.g. choose **PC-Adapter** and **Install** (→ PC-Adapter →Install).

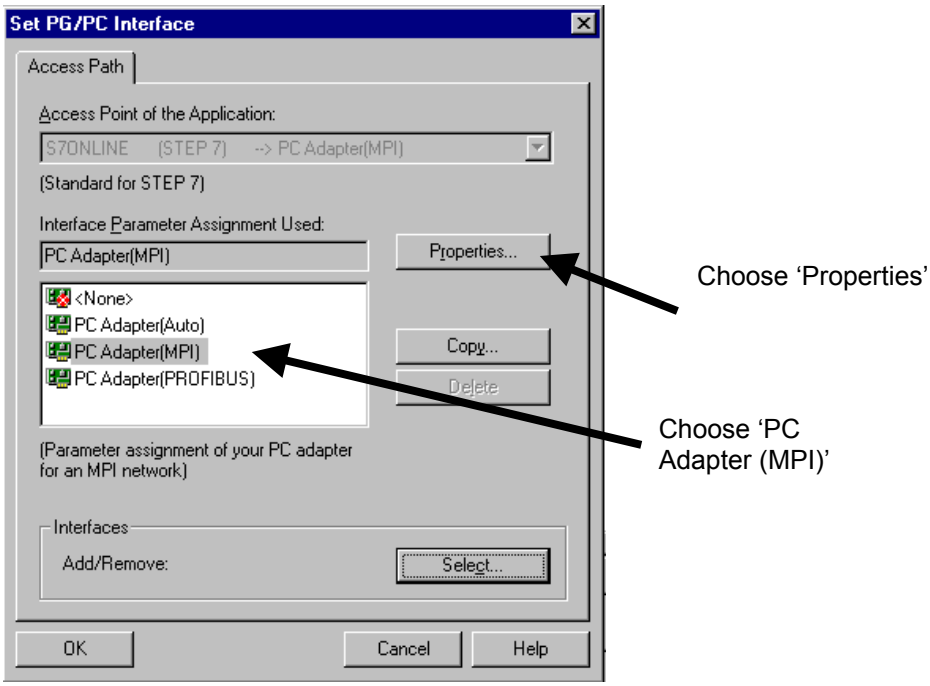




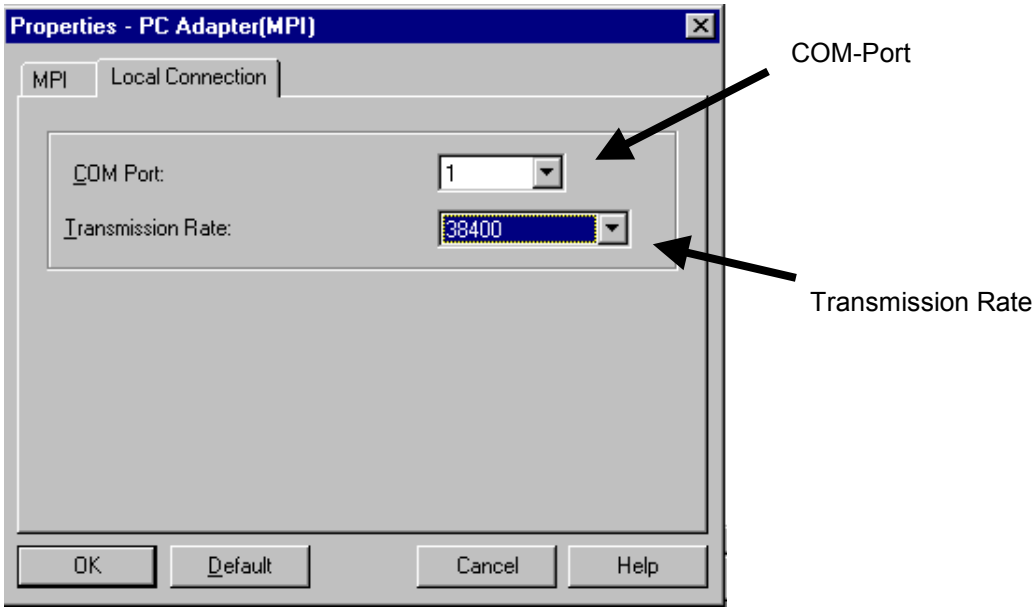
4. Make sure the desired module is present (→ PC Adapter → Close).



5. Choose **Properties** of **PC-Adapter (MPI)** (→ PC Adapter(MPI) → Properties).

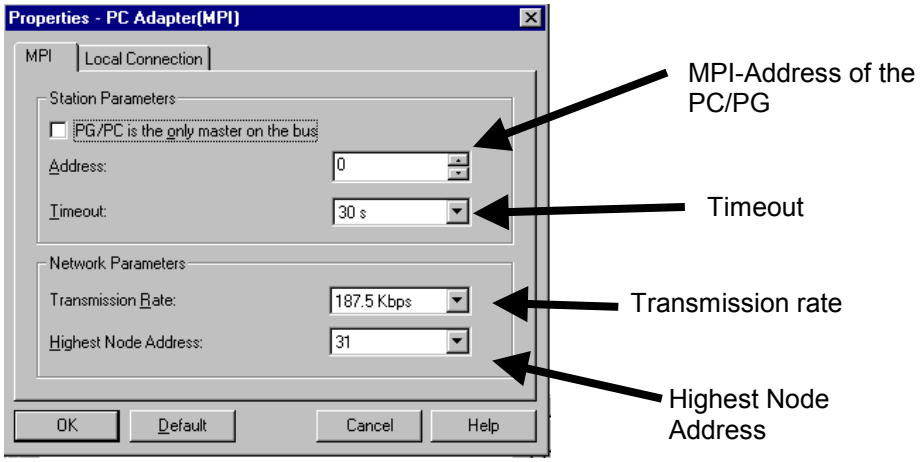


6. Set the **COM-Port** and the **Transmission Rate** of the serial interface.



Note: The transmission rate must be suitably adjusted for the PC adapter! Older PC adapters (PC/MPI cables) should only be processed with a slower transmission rate of 19200 Bit/s .

7. Set the **MPI-Address, Timeout, Transmission Rate** and **Highest Node Address**



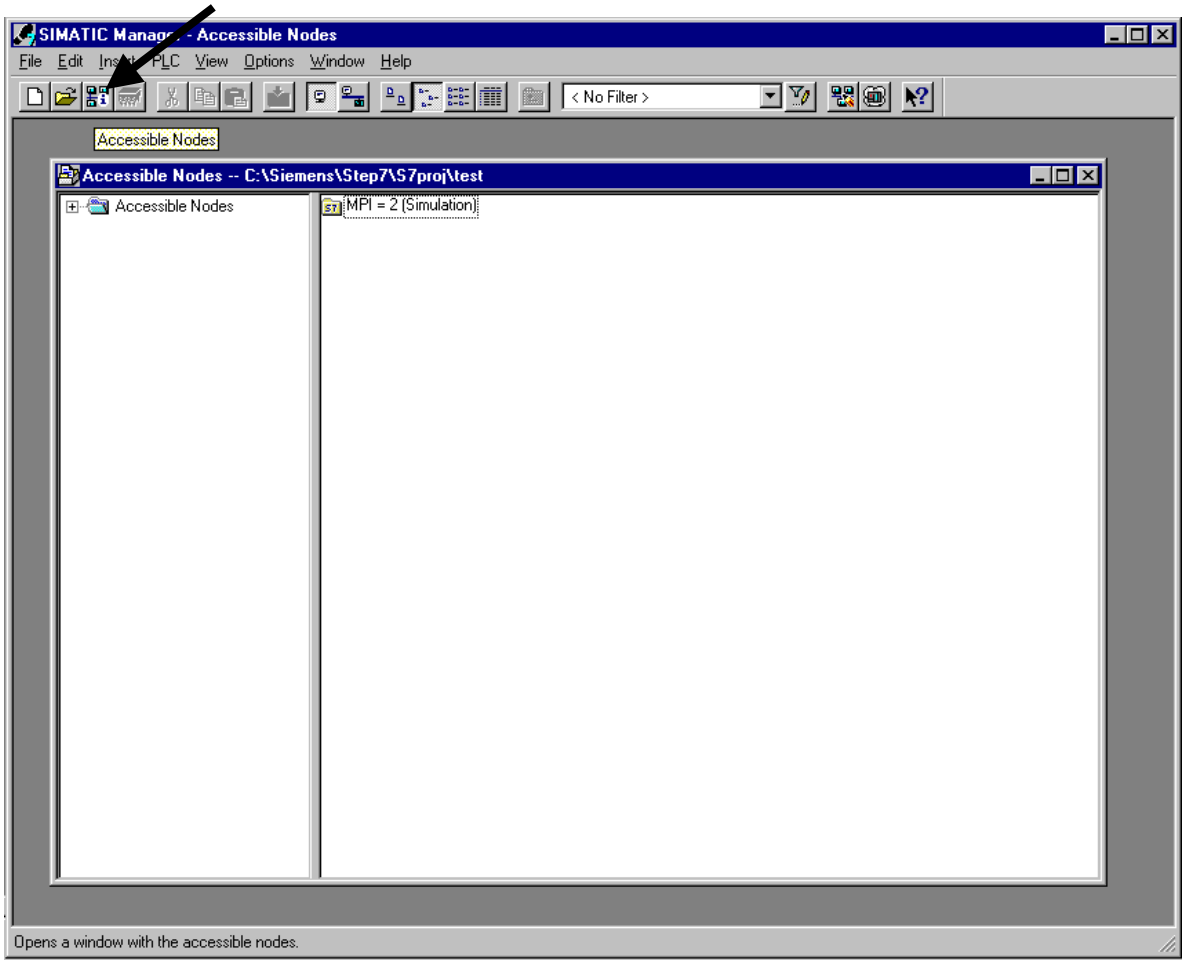
Note: It is recommended to use the preset values!

8. Accept the configuration (→ OK → OK).

9. After the values have been configured, double click on the **SIMATIC Manager** icon (→ SIMATIC Manager).

10. The plug placed from the MPI interface of the PC will appear on the MPI interface of the CPU and switch the voltage supply of the PLC on. The MPI interface is found behind the front flap of the CPU in the form of a 9pin D Sub socket.

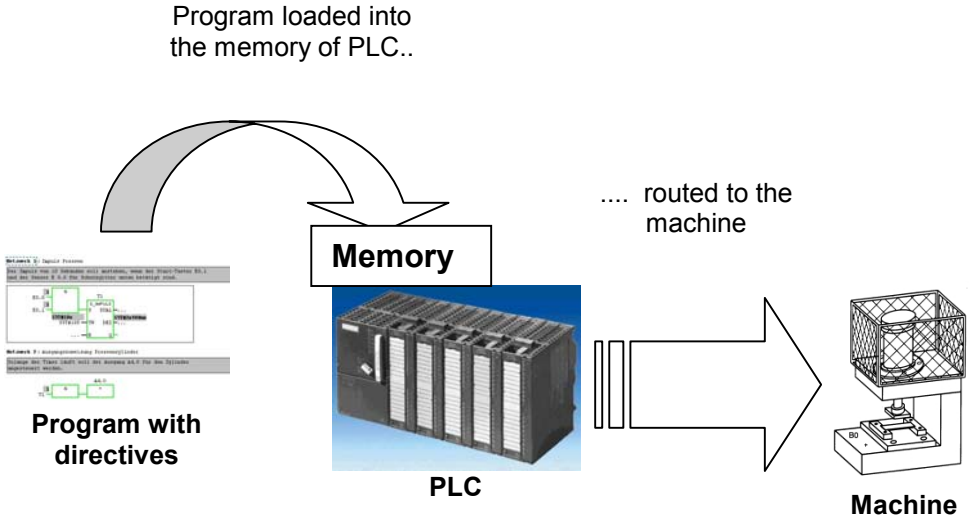
11. When the button – **Accessible Nodes** is clicked and all parameters were correctly selected, the screen will display the following picture with a folder for the reached MPI interface. The MPI-Address of the CPU is also shown, which is calibrated with a 2 (→).



5. WHAT IS PLC AND WHAT ARE PLCS USED FOR?

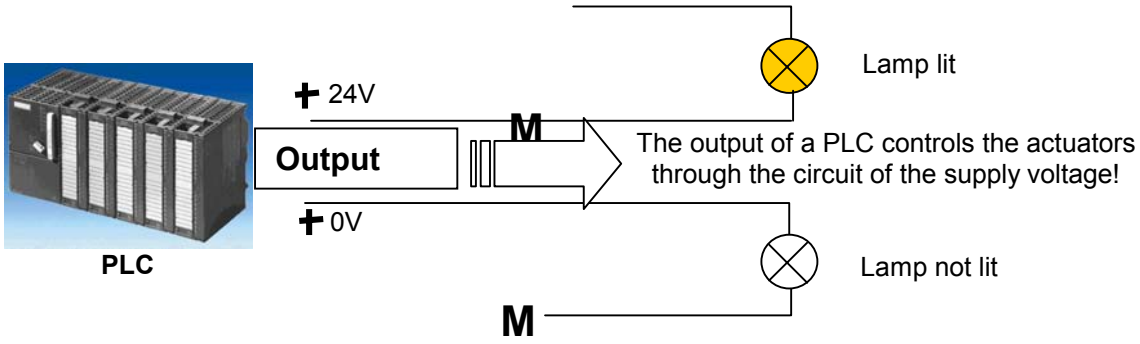
5.1 WHAT IS THE CONCEPT OF A PLC?

PLC is an abbreviation for programmable logic control. This describes equipment that controls a process (e.g. a printing machine for printing newspapers, a bagging plant to bag cement, a press for pressing plastic-shaped parts, etc ...).
This process occurs according to the instructions of a program in the memory of the equipment.



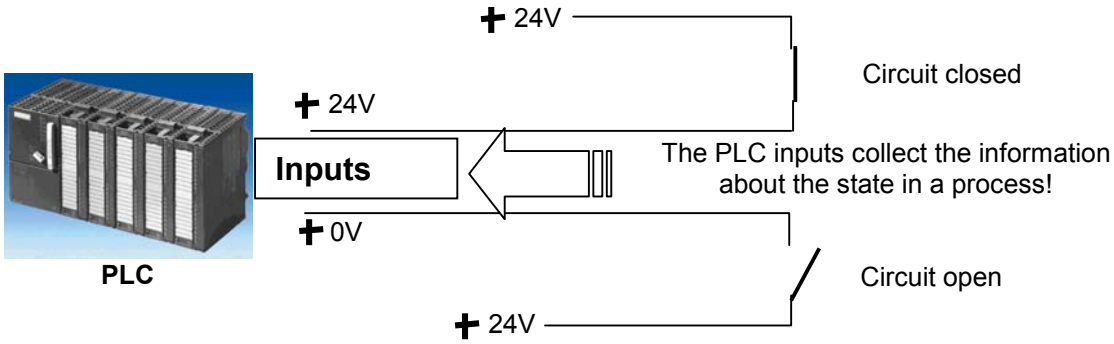
5.2 HOW DOES A PLC DRIVE A PROCESS?

The PLC controls the process, in which **Actuators** are wired as **Outputs** to designated connections of a PLC with a control supply voltage of e.g. 24V. Motors can be switched on and off, valves extended or retracted, or lamps switched on and off through this connection.



5.3 FROM WHERE DOES A PLC GET INFORMATION ABOUT THE STATE OF A PROCESS?

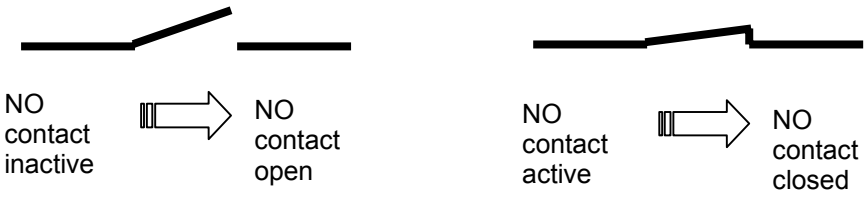
A PLC receives information about the process from **Signal generators** which are wired to the **inputs** of the PLC. These signal generators can be e.g. sensors which recognize whether a working part, switches or buttons lie in a certain position. This position can be closed or opened. Please note the variation between **NC contacts**, which are inactive when closed, and **NO contacts**, which are inactive when open.



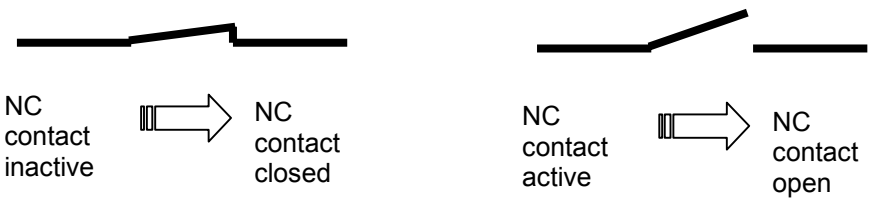
5.4 WHERE DOES THE DIFFERENCE BETWEEN A NORMALLY OPEN (NC) AND CLOSED (NC) CONTACT LIE?

The variation between **NO contacts** and **NC contacts** is within a signal generator

The switch shown here is a NO contact, i.e. it is closed when it is active.



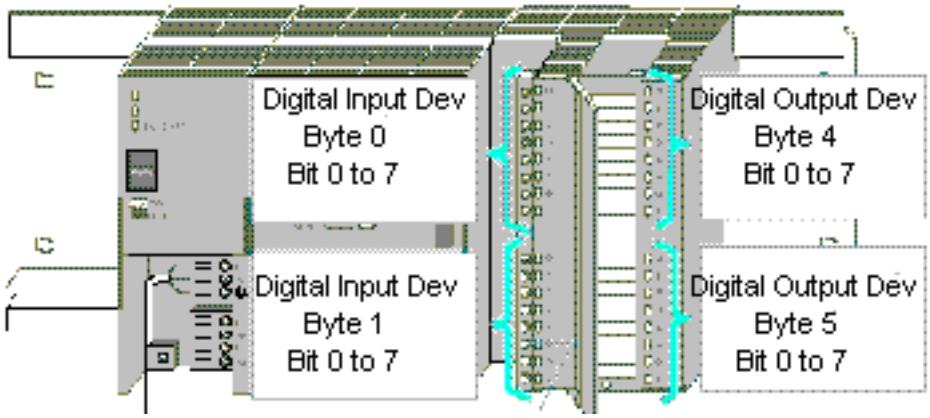
The switch shown here is a NC contact. i.e. it is closed when it is not active.



5.5 HOW DOES A PLC COMMUNICATE WITH IN/OUTPUT SIGNALS?

The designation of a certain input or output within the program is referred to as addressing.
The inputs and outputs of the PLCs are mostly defined in groups of eight on digital input and/or digital output devices. This eight unit is called a **byte**. Every such group receives a number as a **byte address.**
Each in/output byte is divided into 8 individual **bits**, through which it can respond with. These bits are numbered from bit 0 to bit 7. Thus one receives a **bit address**.

The PLC represented here has input bytes 0 and 1 as well as output bytes 4 and 5.



Here e.g. the fifth input from the higher bits responds with the following address:

I 0 . 4

I Here the address type is specified as Input, **0** the byte address and **4** the bit address.
The byte address and bit address are always separated with a point.

Note: For the bit address here, the **4** stands for the fifth input because the count begins at 0.

Here e.g. the lower bits respond with the following address:

Q 5 . 7

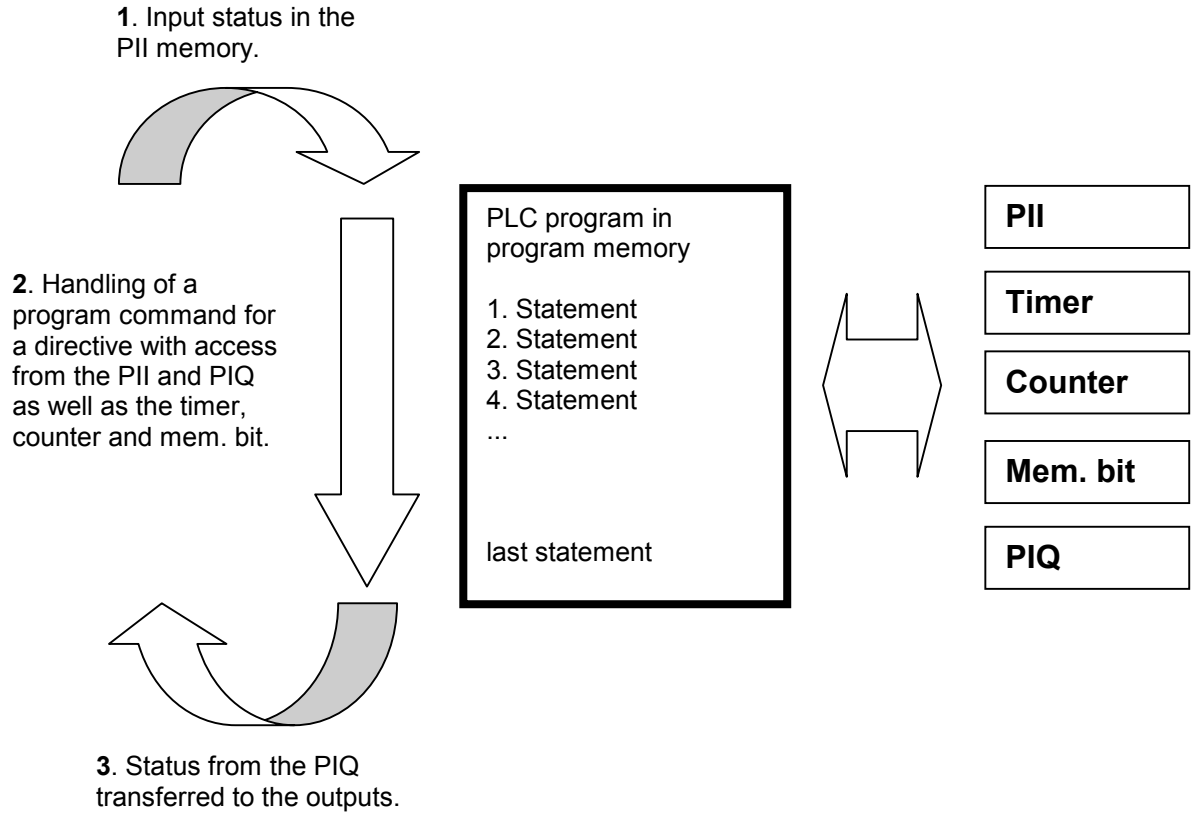
Q Here the address type is specified as Output, **5** the byte address and **7** the bit address.
The byte address and bit address are always separated with a point.

Note: For the bit address here, the **7** stands for the eighth output, because the count begins at 0.

5.6 HOW DOES THE PROGRAM WORK IN A PLC?

Program processing in a PLC happens cyclically with the following execution:

1. After the PLC is switched on, the **processor** (which represents the brain of the PLC) questions if the individual inputs have been transmitted or not. This status of the input is stored in the process- image input table (**PII**). Leading inputs become the information 1 or High when enabled, or the information 0 or Low when not enabled.
2. This processor processes the program deposited into the program memory. This consists of a list of logic functions and instructions, which are successively processed, so that the required input information will already be accessed before the read in PII and the matching results are written into a process-image output table (**PIQ**). Also other storage areas for counters, timers and memory bits will be accessed during program processing by the processor if necessary.
3. In the third step after the processing of the user program, the status from the PIQ will transfer to the outputs and then be switched on and/or off. Afterwards it continues to operate, as seen in point 1.



Note: The time that the processor requires for this execution is called a cycle time. This time is independent from the number and types of commands.

5.7 HOW DO THE LOGIC OPERATIONS IN A PLC PROGRAM APPEAR?

Logic functions can be used in order to specify conditions for the toggling of outputs.
These functions can be provided to the PLC-Program in the programming languages ladder diagram (**LAD**), function block diagram (**FBD**) or statement list (**STL**). For the sake of descriptiveness, we will limit ourselves here to **FBD**. A wide range of different logic operations can be used in PLC programs. **AND** as well as **OR-** operation and **NEGATION** of an input can be frequently used. Basic examples are briefly described below.

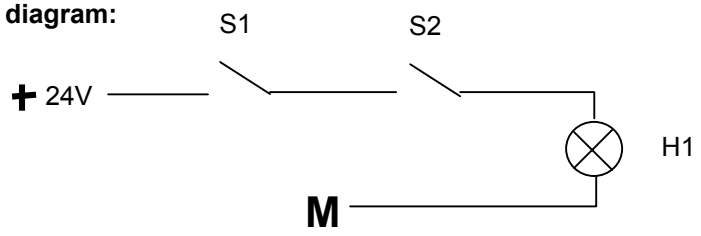
Note: Further information of logic operations can be quickly found in the online help section.

5.7.1 AND- OPERATION

Example of an AND- OPERATION:

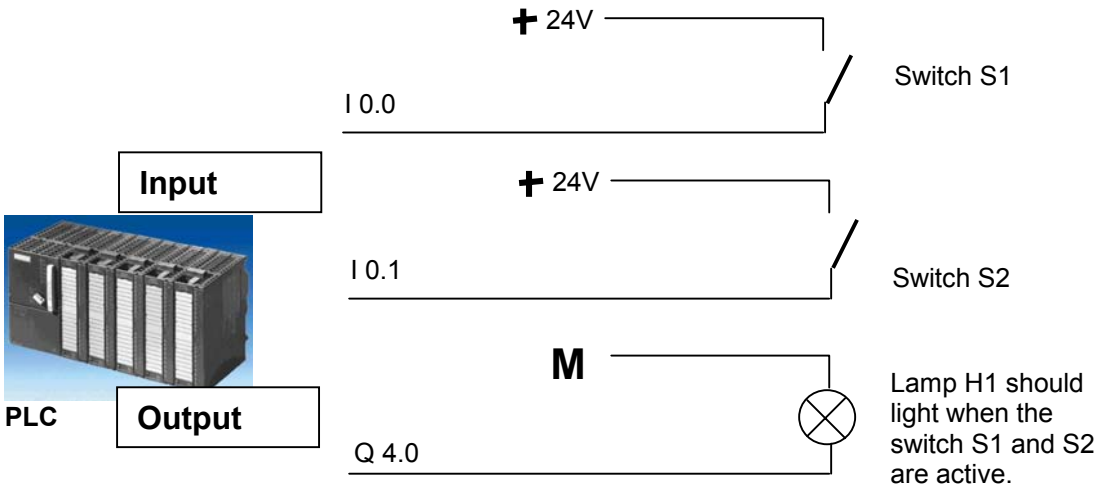
A lamp should ignite when two switches at a closed contact are active at the same time.

Circuit diagram:



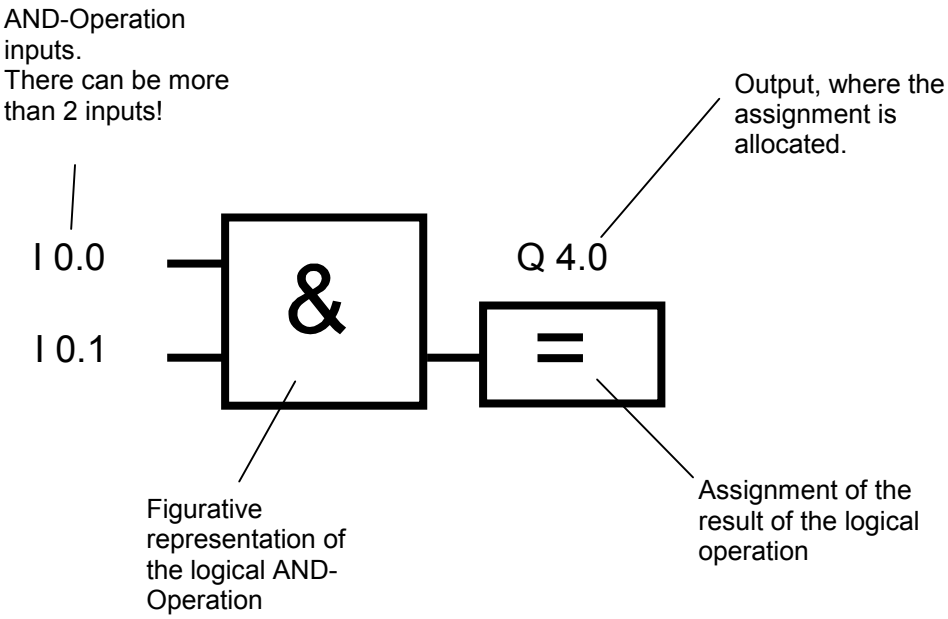
Comment:
The lamp lights when both switches are active.
When the switch S1 **and** S2 are active, the lamp H1 will light up.

A PLC circuit:
In order to implement logic in a PLC program, both switches must be naturally attached at the inputs of the PLC. Here S1 is wired to the input I 0.0 and S2 to the input I 0.1.
In addition, the lamp H1 must be attached to an output e.g. Q 4.0.



AND- Operation in FBD:

In the function diagram FBD, the AND- Operation is programmed and is shown by figurative representation below:

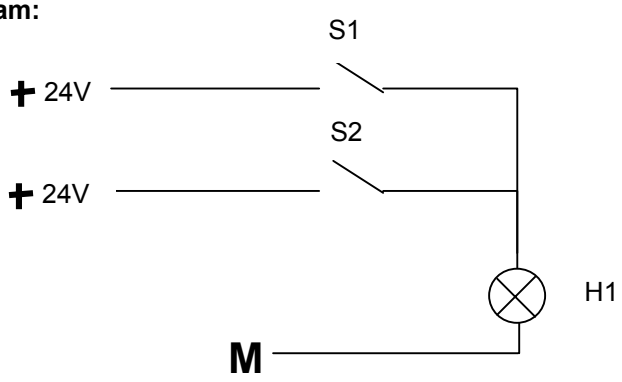


5.7.2 OR- OPERATION

Example of an OR- Operation:

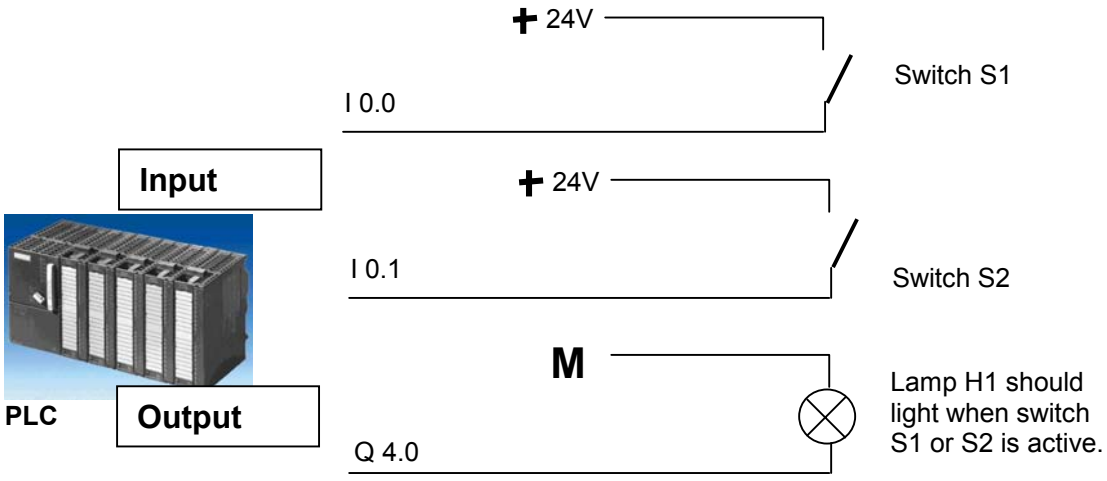
A lamp should ignite when one or both switches at a normally open circuit are active.

Circuit diagram:



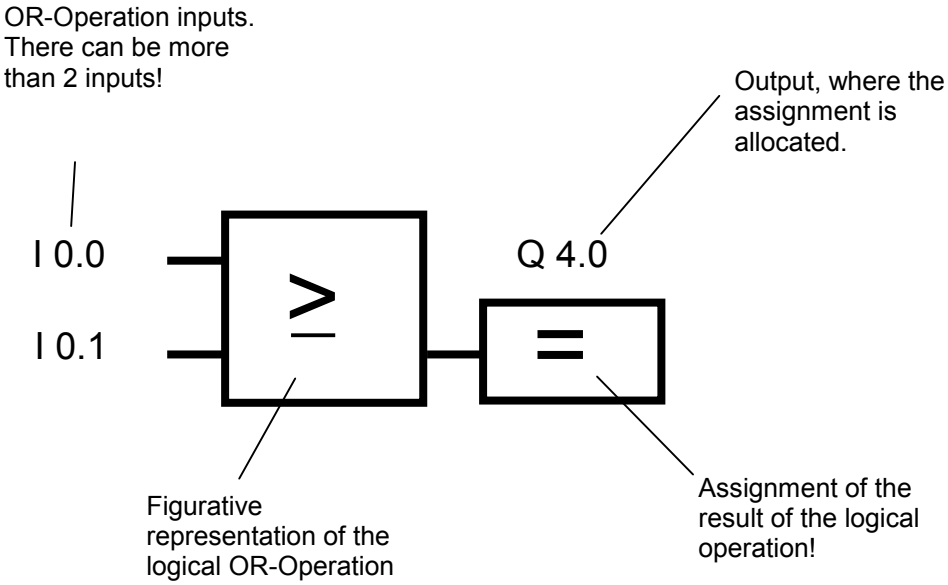
Comment:
The lamp lights when one or both switches are active.
When the switch S1 **or** S2 is active, lamp H1 will light up.

A PLC circuit:
In order to implement logic in a PLC program, both switches must be naturally attached at the inputs of the PLC. Here S1 is wired to the input I 0.0 and S2 to the input I 0.1.
In addition, the lamp H1 must be attached to an output e.g. Q 4.0.



OR- Operation in FBD:

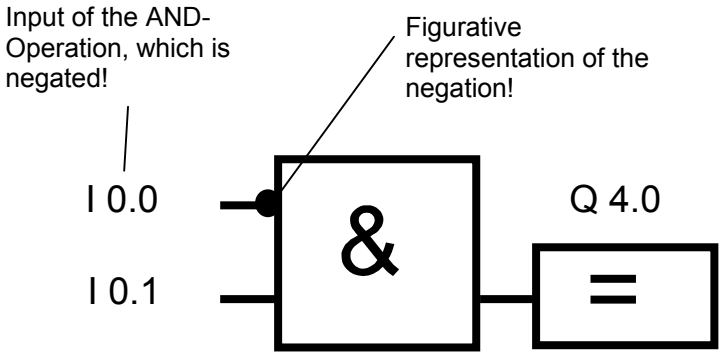
In the function diagram FBD, the OR- Operation is programmed and is shown by figurative representation below:



5.7.3 NEGATION

In logical functions it is often required to know whether a **NO contact** is **not active** or if a **NC contact** is **active** so that there will be no voltage against the appropriate inputs.
This can be achieved with the use of a **Negation** on the input of the AND/OR Operation.

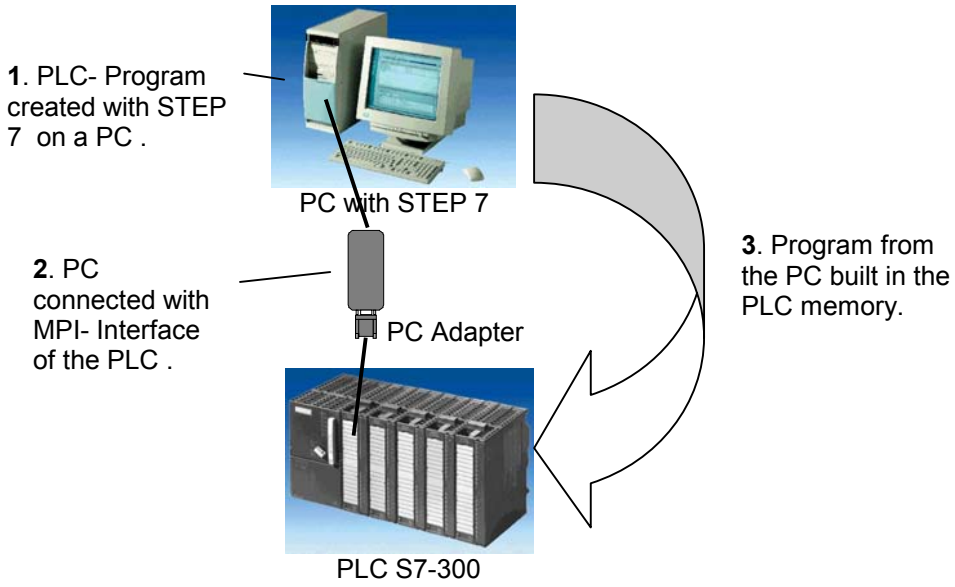
In the function diagram FBD, the negation of the inputs of the AND- Operation is programmed and is shown by figurative representation below:



The output Q 4.0 has the correct value when I 0.0 is not active and I 0.1 is active.

5.8 HOW IS A PLC- PROGRAM GENERATED? HOW DOES IT ARRIVE IN THE MEMORY OF THE PLC?

The PLC program is provided with the software STEP 7 on a PC and buffered there.
After the PC is connected with the MPI interface of the PLC, the program can be loaded with a loading function into the memory of the PLC.



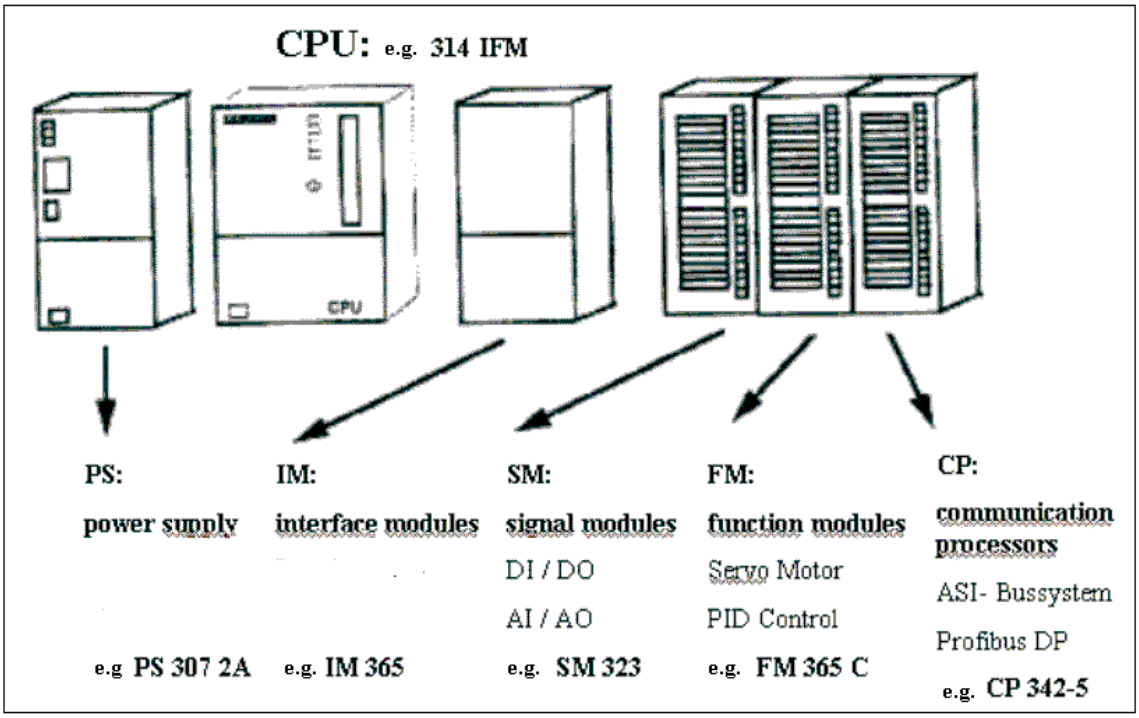
Notice: The exact execution of the program will be described step by step in chapters 8 through 10.

6. ASSEMBLY AND OPERATION OF THE SIMATIC S7-300.

Device spectrum:

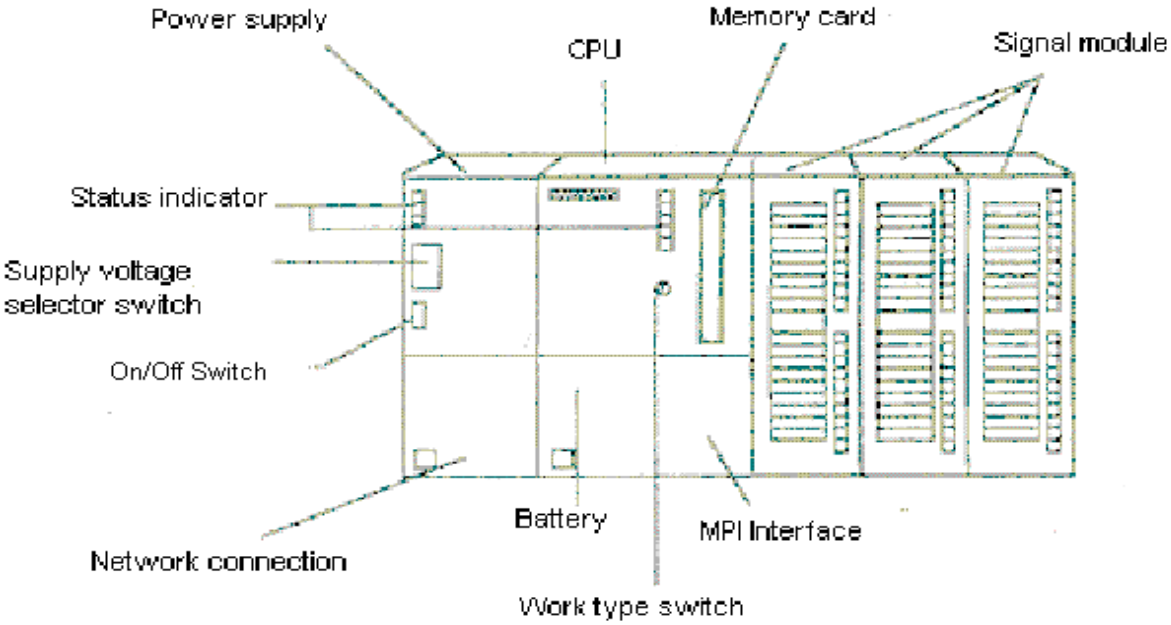
The SIMATIC S7-300 is a modular miniature control system and provides the following device spectrum:

- Central processing units (CPUs) with different power ranges, partly integrated with In-/Outputs (e.g. CPU312IFM/CPU314IFM) or integrated with a PROFIBUS- Interface (e.g. CPU315-2DP)
- Power supply devices (PS) with 2A, 5A or 10A.
- Interface modules (IMs) for a more interconnecting design of the SIMATIC S7-300
- Signal modules (SMs) for digital and analog in- and output.
- Function modules (FMs) for special functions (e.g. stepping motor control)
- Communication processors (CP) for network connection.



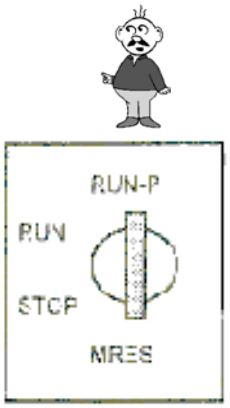
Note: Only a current supply device, any CPU as well as a digital in and output is required for this module.

Important elements of a voltage supply and CPU:



MPI- Interface:
Each CPU possesses an MPI interface for the networking of program devices (e.g. PC adapter). This is found behind a flap at the front of the CPU.

Mode selector:
Each CPU possesses a code switch for the switching of the modes of operation. Certain programmed functions are allowed depending upon the position of the code switch. The following modes of operation are possible:



RUN-P: Program runs; All PG functions are allowed.

RUN: Program runs; Only read PG functions are allowed

STOP: Program does not run; All PG functions are allowed.

MRES; With this position, one can accomplish a reset as described

Memory reset:

Memory reset erases all user data on the CPU each time the program is begun.

This is performed in the following three steps:

Step	Execution	Result
1	Turn the key to the **STOP** position.	STOP indication is shown.
2	Turn the key to the **MRES** position and hold it in this position (approx. 3 seconds) until the STOP- indicator is shown.	The STOP-Indicator expires and after approx. 3 seconds, it will be shown again. With new CPUs, wait until the STOP-Indicator lights up for the second time. **Important:** Between step 2 and step 3 a maximum of 3 seconds should go by.
3	Turn the key back to the **STOP** position and within the following 2 seconds restart in the **MRES** position.	The STOP-Indication blinks for approx. 3 seconds and then lights up again normally: When everything is ok.; The **CPU is reset.**

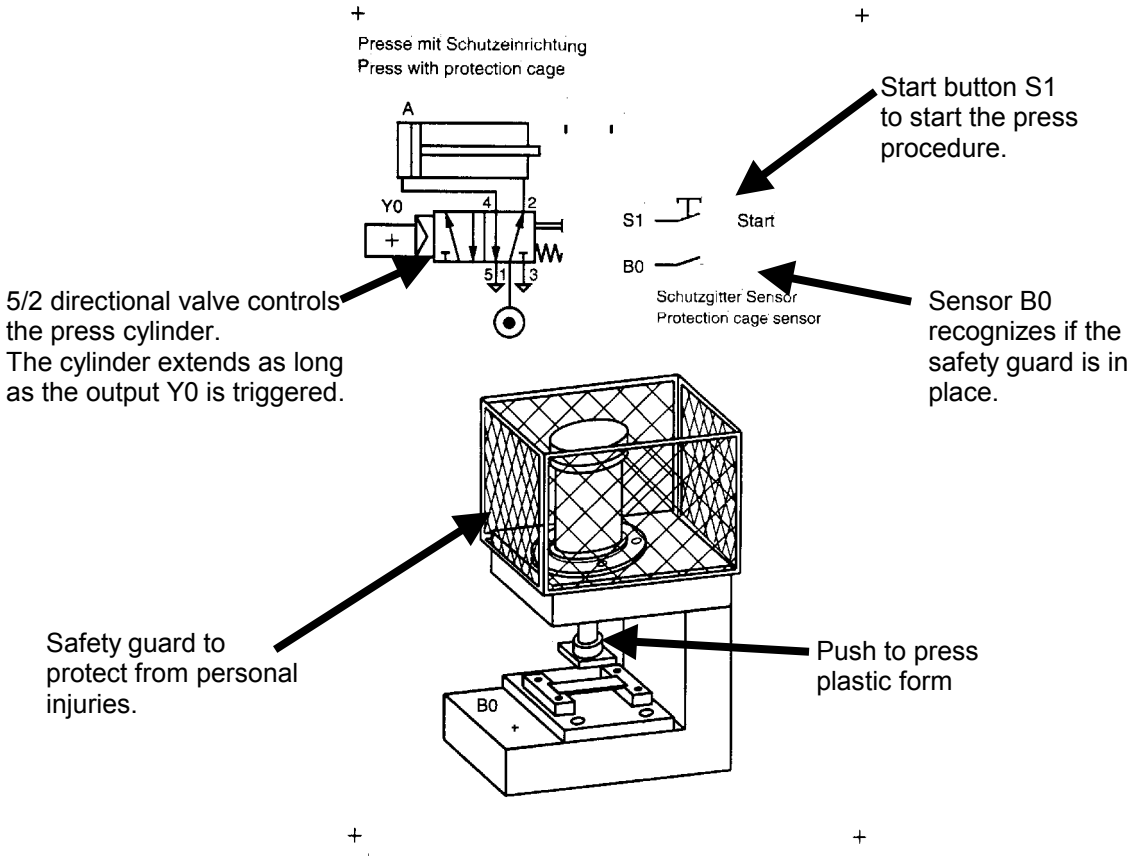
7. EXAMPLE EXERCISE

A simple exercise will be performed for our first STEP 7 program.

A press with a protection device will be released only with a START-Button S1 when the safety guard is closed. This condition is supervised with a sensor safety guard BO.
This case accesses a 5/2 directional valve Y0 for the press cylinder every 10 seconds, so that a plastic form can be pressed.
For safety reasons, the press is raised again when the START-Button S1 is released or when the sensor safety guard B0 is no longer activated.

Allocation map:

Address	Symbol	Comment
I 0.0	B0	Sensor safety guard
I 0.1	S1	Start- Button
Q 4.0	Y0	5/2 directional valve for the press cylinder



8. STEP 7- PROJECT APPLICATION

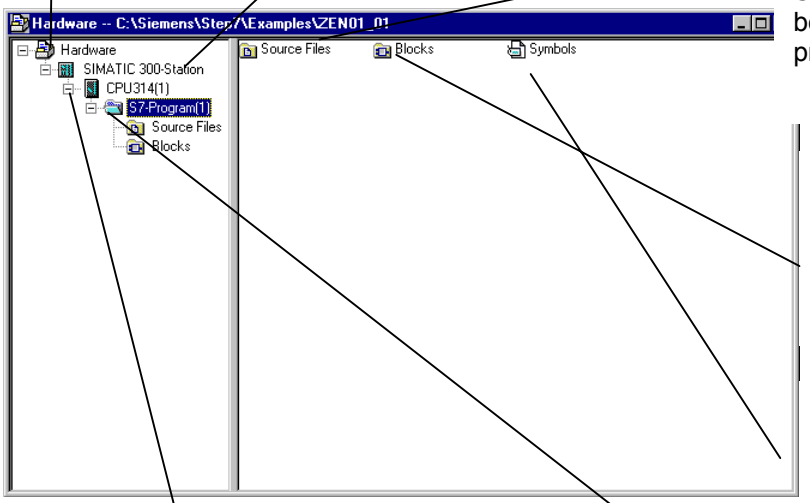
File management takes place in STEP 7 with the **SIMATIC Manager**. Here e.g. program blocks can be copied or be called for further processing with other tools by clicking with the mouse. The operation corresponds to the standards usually seen in WINDOWS 95/98/2000/ME/NT4.0. (e.g. With one right click from the mouse button, one is able to receive the selection menu to each module).
In the folders **SIMATIC 300 station** and **CPU**, the structure of the hardware of a PLC is illustrated. Therefore such a project can always be seen as hardware specific.

In STEP 7, each project is put into a firmly given structure. The programs are stored in the following directories:

Project:
The directory contains the hardware (e.g. SIMATIC 300 Station) and the sub structure (e.g. MPI and PROFIBUS).

SIMATIC 300 Station:
Stored here are the appropriate hardware configuration (Hardware/SC*1) and CPU data.

Source Files/SO*[1]:
Sources are placed here (e.g. SCL- Source Files). They can be converted into executable programs by translation.



Blocks/AP-off*[1]:
Stored here are the program blocks (OB, FB, FC, SFB, SFC, DB etc.).

Symbols/SY*[1]:
Stored here are the symbol lists for symbolic addressing.

CPU:
The S7 program and the interlaced connecting partners (Connection/CO*[1]) are registered here.

S7-Program: The user programs (Blocks/AP-off*[1]), symbol tables (Symbols/SY*[1]), and Source files(Source files/SO*[1]) are administered here.

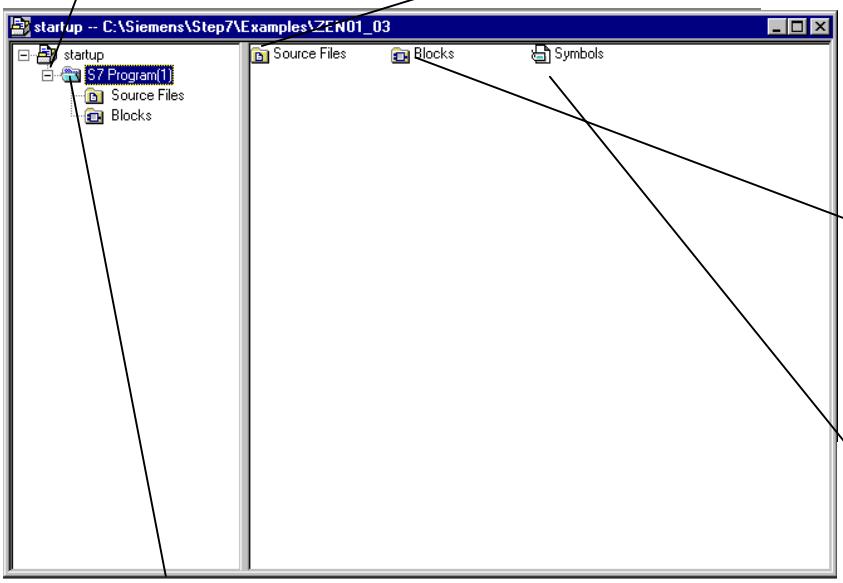
*[1] Terms are from STEP 7 Version 2.x

In order to make a project independent from the hardware, one can create a project that does not contain all possible files.

This project would have the following structure:

Project:
The directory contains the hardware (e.g. SIMATIC 300 Station) and the sub structure (e.g. MPI and PROFIBUS).

Source Files/SO*[1]:
Sources are placed here (e.g. SCL- Source Files). They can be converted into executable programs by translation.



Blocks/AP-off*[1]:
Stored here are the program blocks (OB, FB, FC, SFB, SFC, DB etc.)

Symbols/SY*[1]:
Stored here are the symbol lists for symbolic addressing.

S7-Program: The user programs (Blocks/AP-off*[1]), symbol tables (Symbols/SY*[1]), and Source files(Source files/SO*[1]) are administered here.

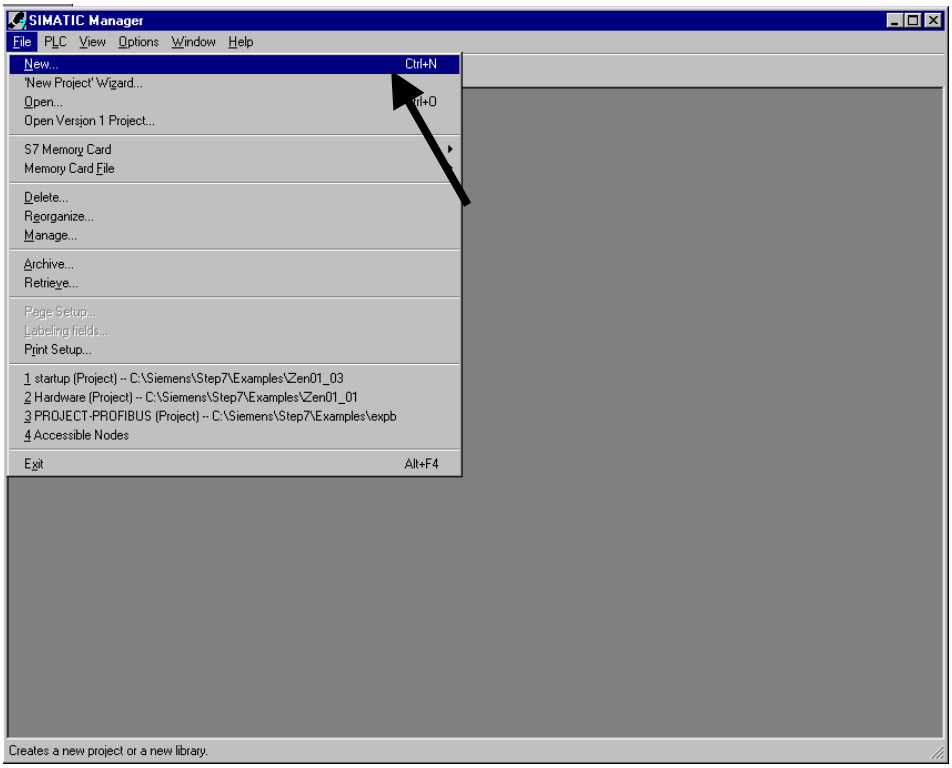
*[1] Terms are from STEP 7 Version 2.x

Note: This example will know the programs provided without the configuration of the hardware. Thus, it will load arbitrary configurations of the SIMATIC S7-300, S7-400 or WinAC. Only the addresses of the inputs and outputs must be adjusted for each individual case.

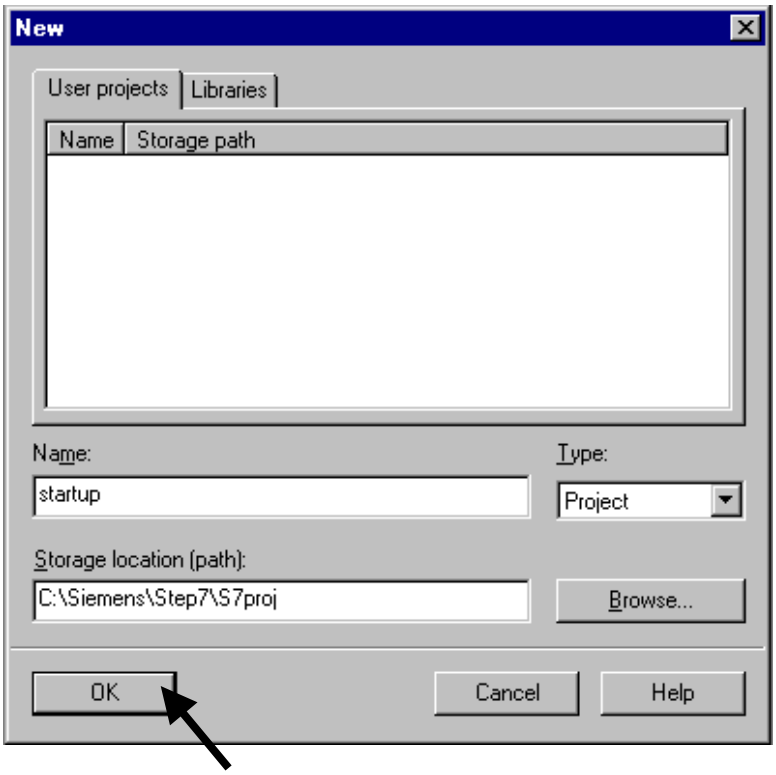
The user must implement the following steps in order to provide a project in which the solution program can be written.

1. The main tool in STEP 7 is the **SIMATIC Manager**, which can be opened with a double click on the icon (→ SIMATIC Manager).

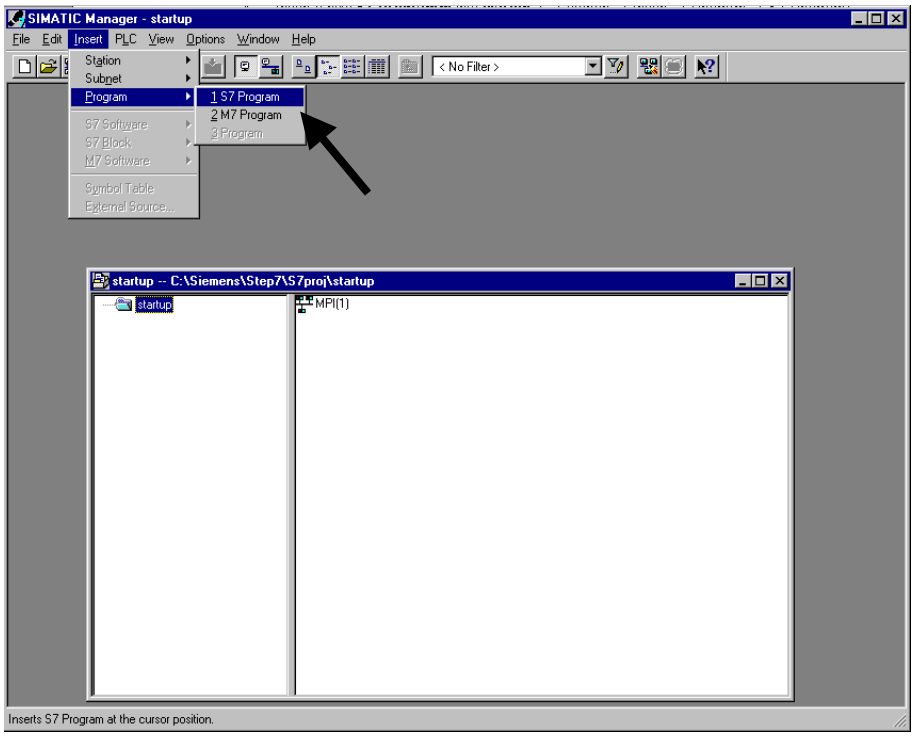
2. STEP 7- Programs are managed in projects. Each project can be newly created (→ File → New).



3. Give the project the **Name startup**. (→ startup → OK)

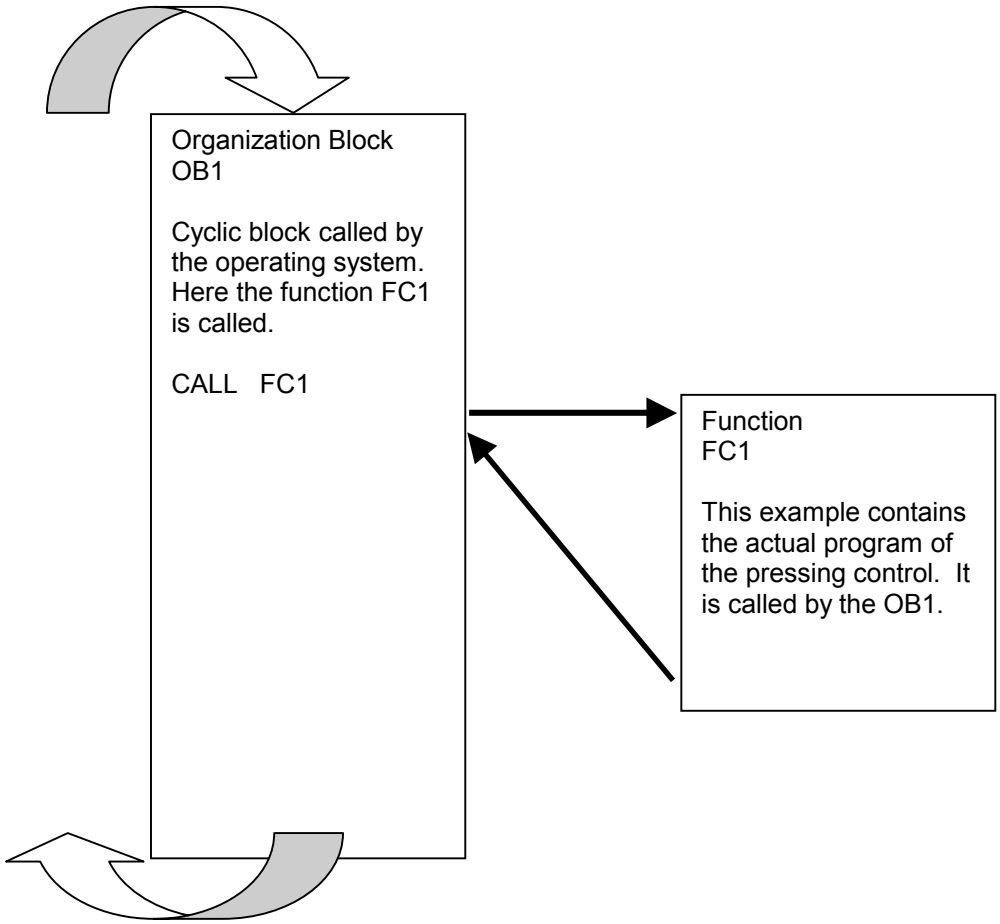


4. Insert a new **S7-Program** into **startup**. (→ startup → Insert → Program → S7-Program)

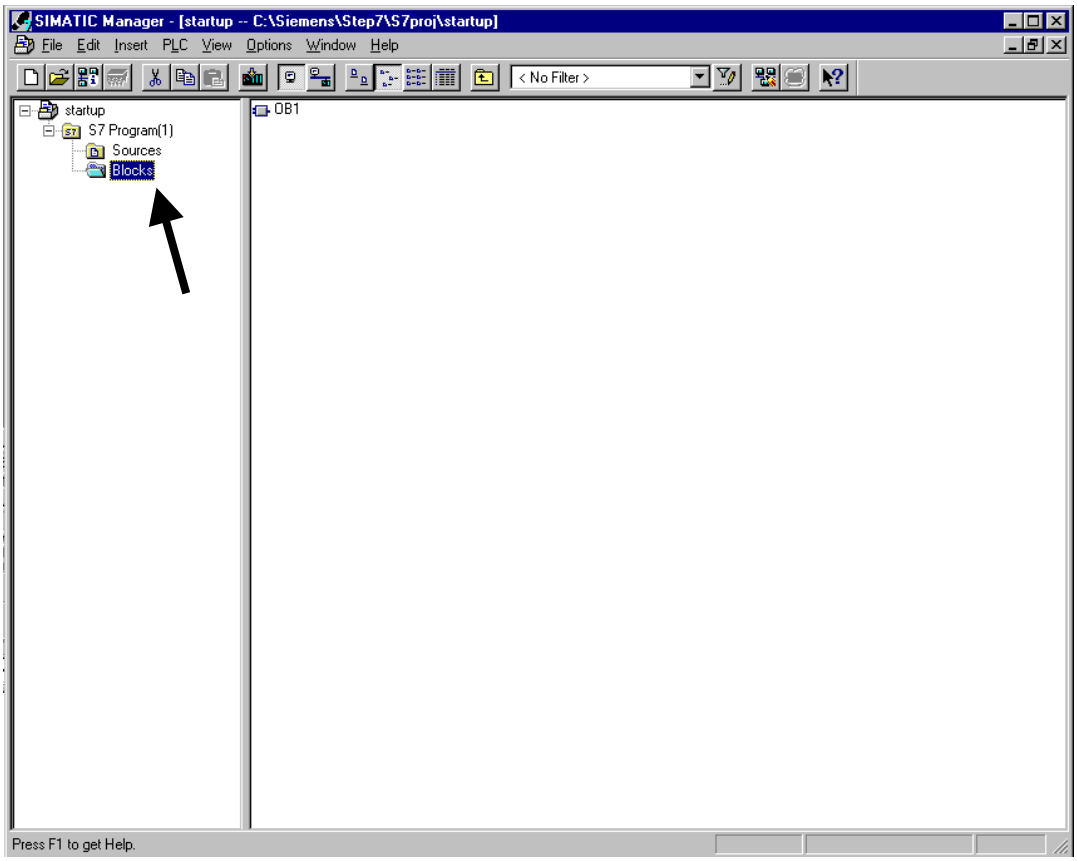


5. The program execution is written into blocks in STEP 7. According to standards, the organization block OB1 is already present. This represents the interface for the operating system of the CPU, which will be automatically called and cyclically worked on.
From this organization block, further blocks e.g. the function FC1 can be called for a program routine. This serves as a process to divide a total task into sub-problems, which are then simpler to solve and simpler to test for functionally.

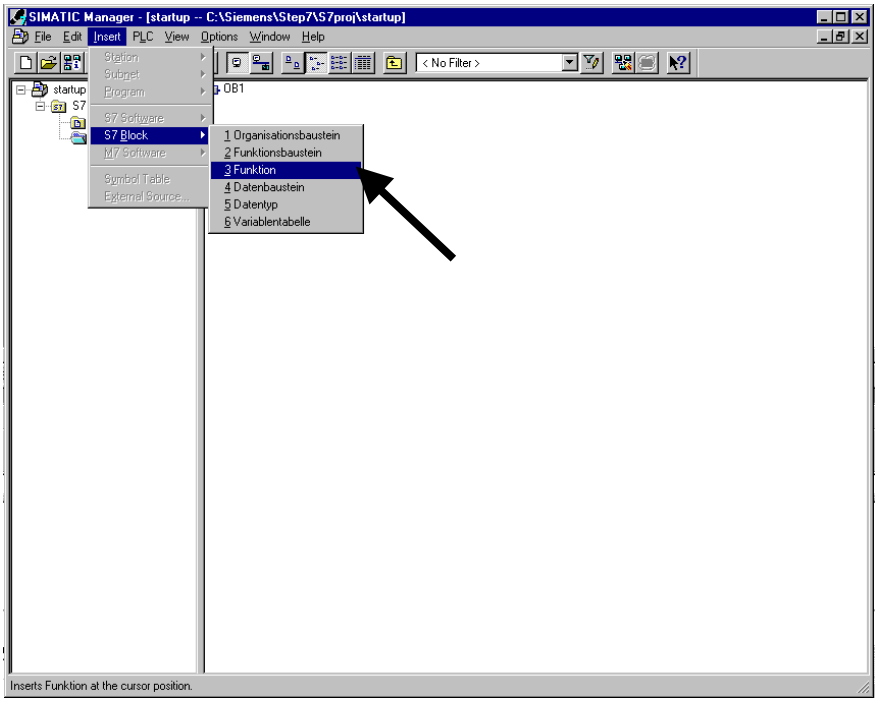
Program structure of the example:



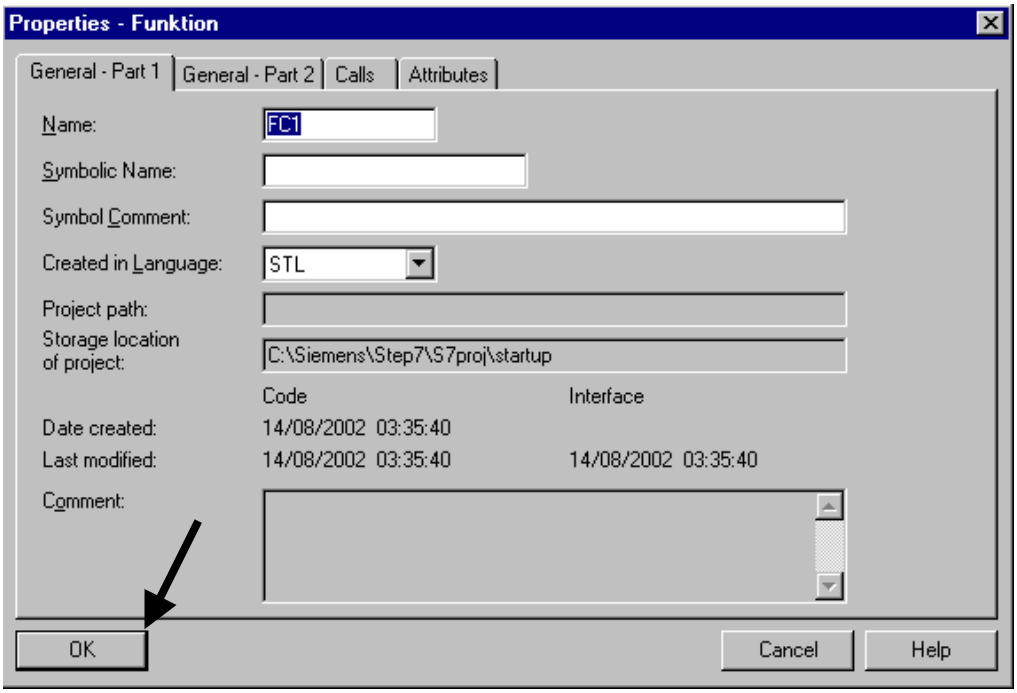
6. In order to further insert the module FC1 into the project, the folder ‘Blocks‘ must be highlighted.(→ Blocks)



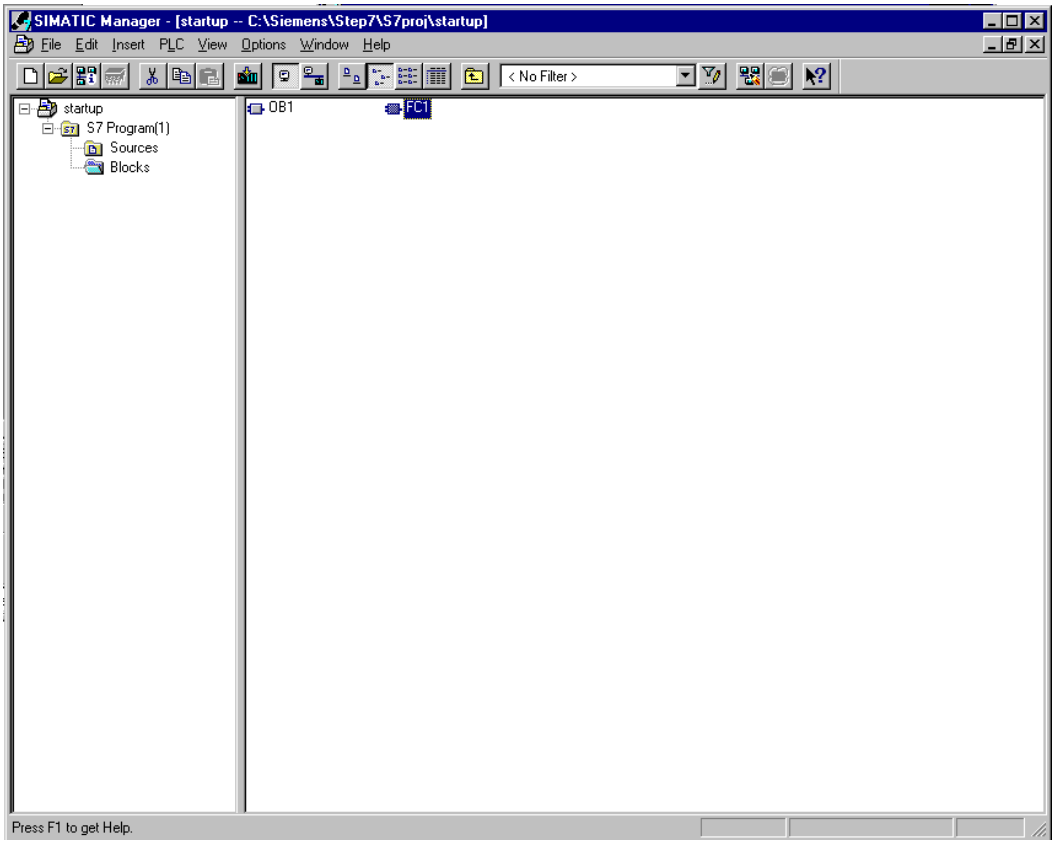
7. The **S7- Block function** is inserted into the folder block. (→ Insert → S7 Block → Function)



8. Now the name of the function can be chosen and further entries for the block document can be made. (→ FC1 → OK)



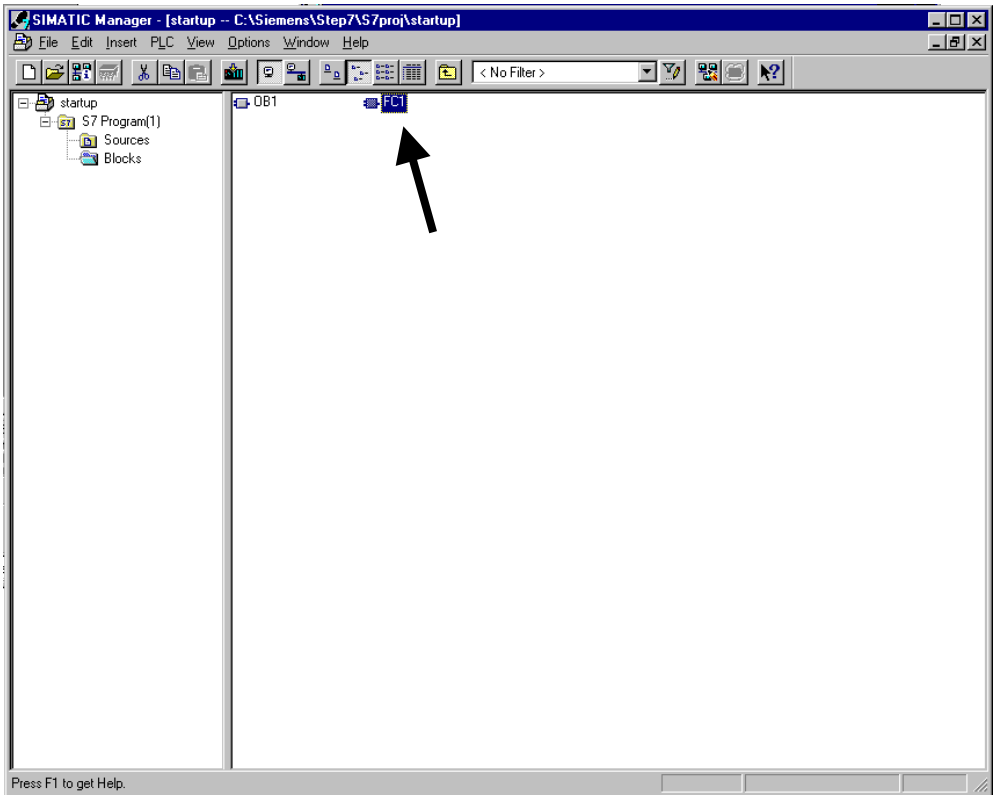
9. The two modules OB1 and FC1 are now available in the SIMATIC Manager and can be further programmed.



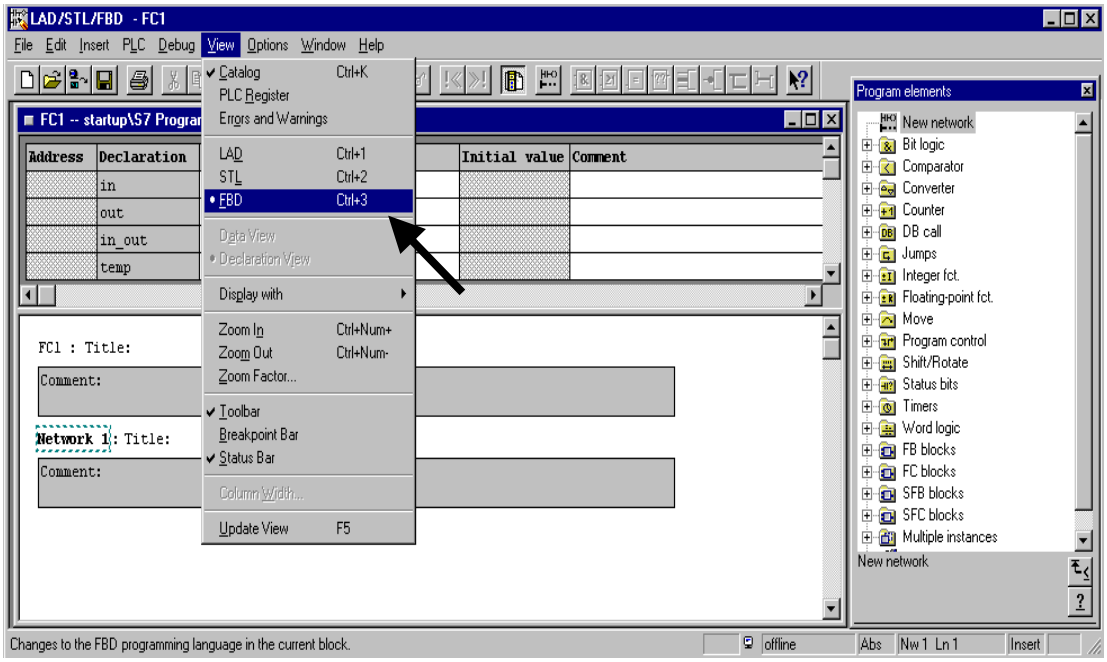
9. STEP 7- PROGRAM WRITING IN FUNCTION DIAGRAM FBD

One possibility to provide a STEP 7 program is the function diagram FBD. A figurative representation of the control problem by means of symbols with function identifiers is shown below. On the left side of the symbol, the inputs are arranged. On the right side, the outputs.

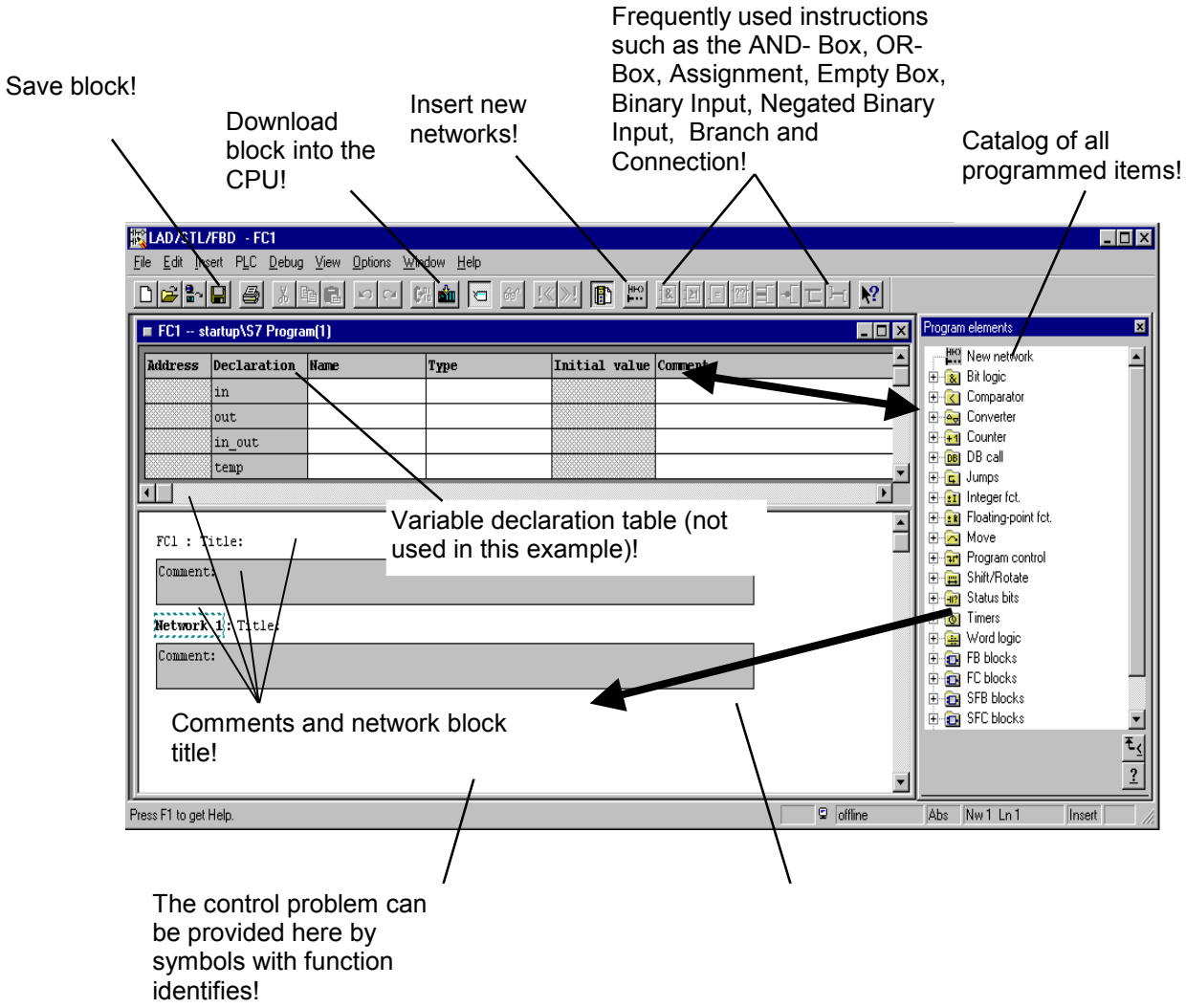
1. The function **FC1** should be worked on here as the first block. The function is opened in the **SIMATIC Manager** with a double click (→ FC1)



2. In the newly opened Editor, the **VIEW** of the programming language function diagram can be changed from **LAD/STL/FBD** to **FBD** (→ View → FBD).

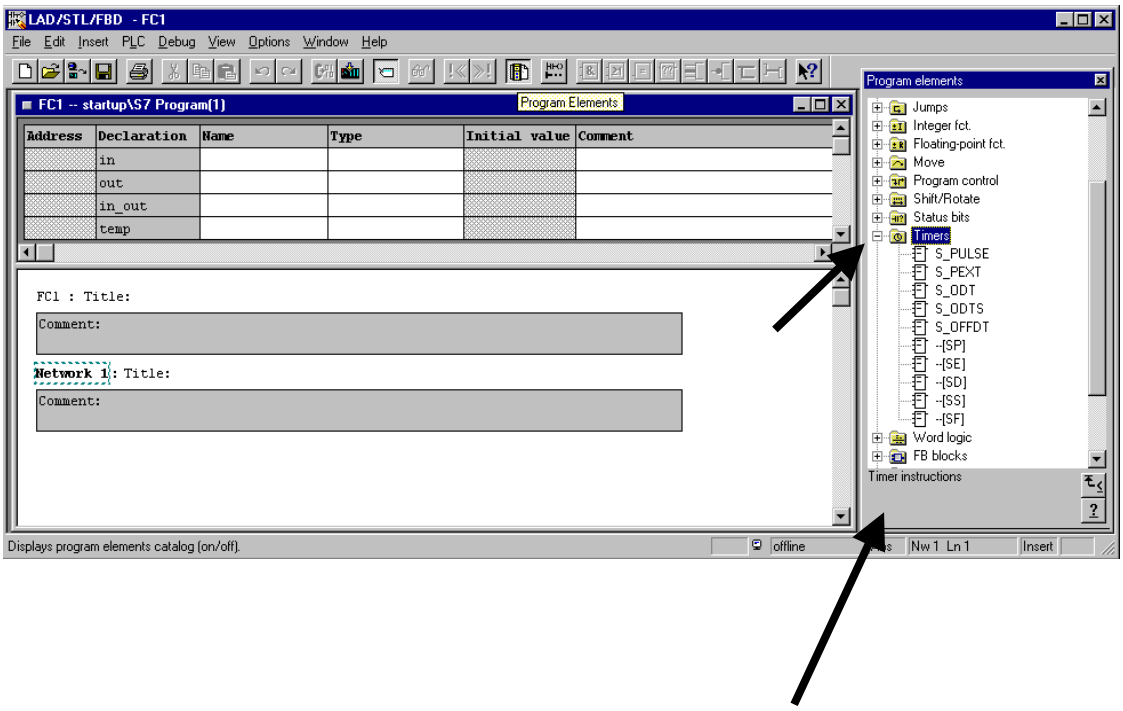


3. The surface program for the programming in the function block diagram (FBD) appears as follows:



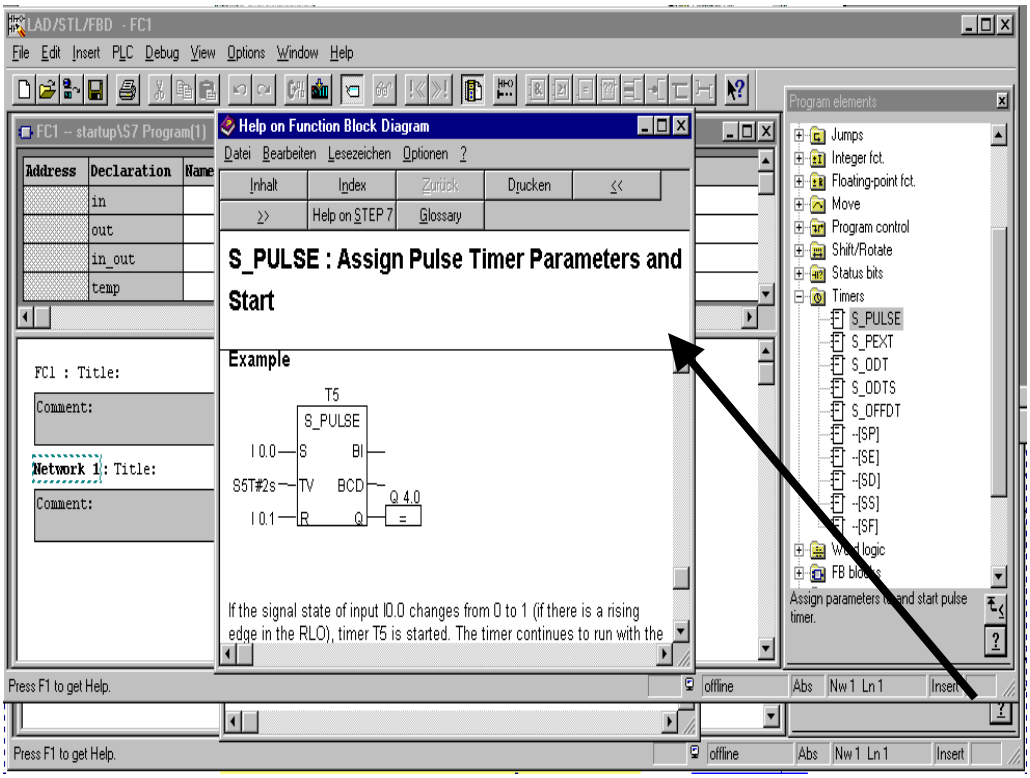
Note: The programs in the STEP 7 blocks are programmed in individual networks. Thus there is a possibly to have a further structuring and improved documentation in the network headings‘ results.

4. For our example, a timer is needed as a pulse. This is called **S_PULSE** and can be found in the catalog under the point **Timers**.(→ Timers → S_PULSE)



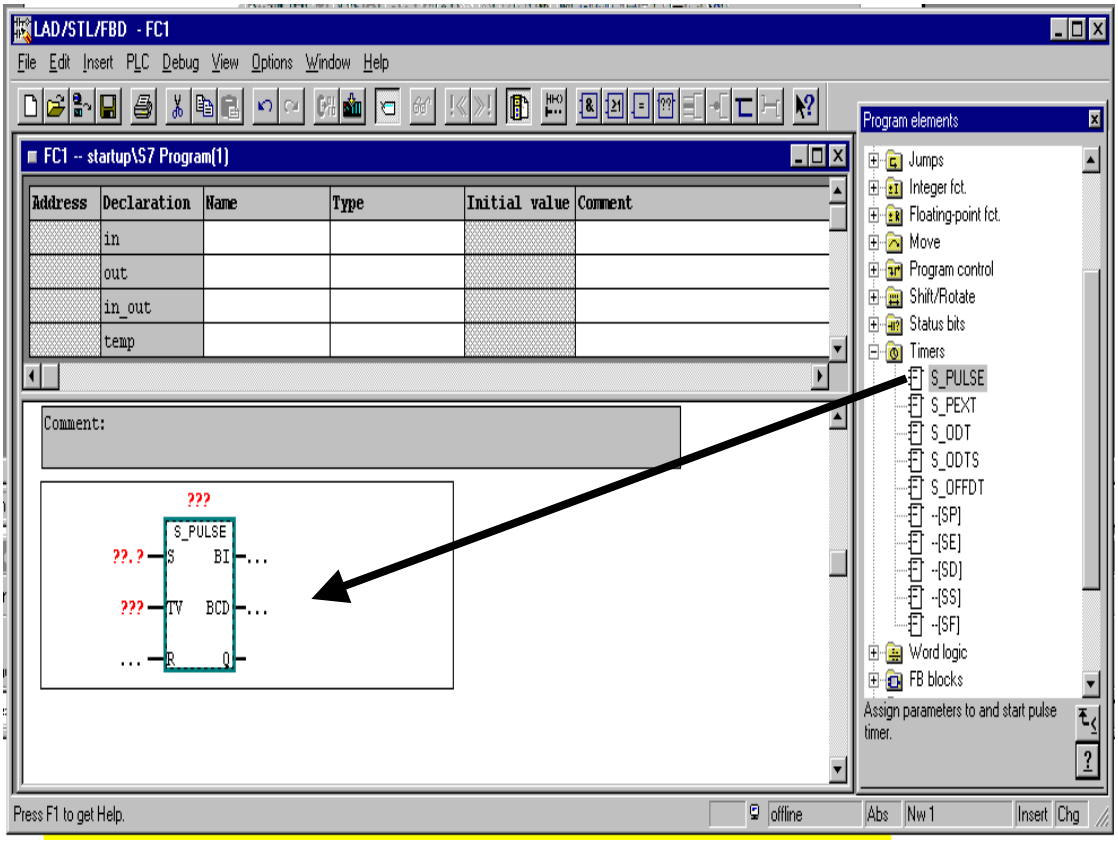
Note: If an operation was selected, it is indicated in the footer of the catalog and is accompanied with a brief description.

5. For exact specifications, information for each operation is contained in the **?** symbol, which is the online assistance manual. It is comprehensive and explains each instruction with a detailed example. (→ **?**)

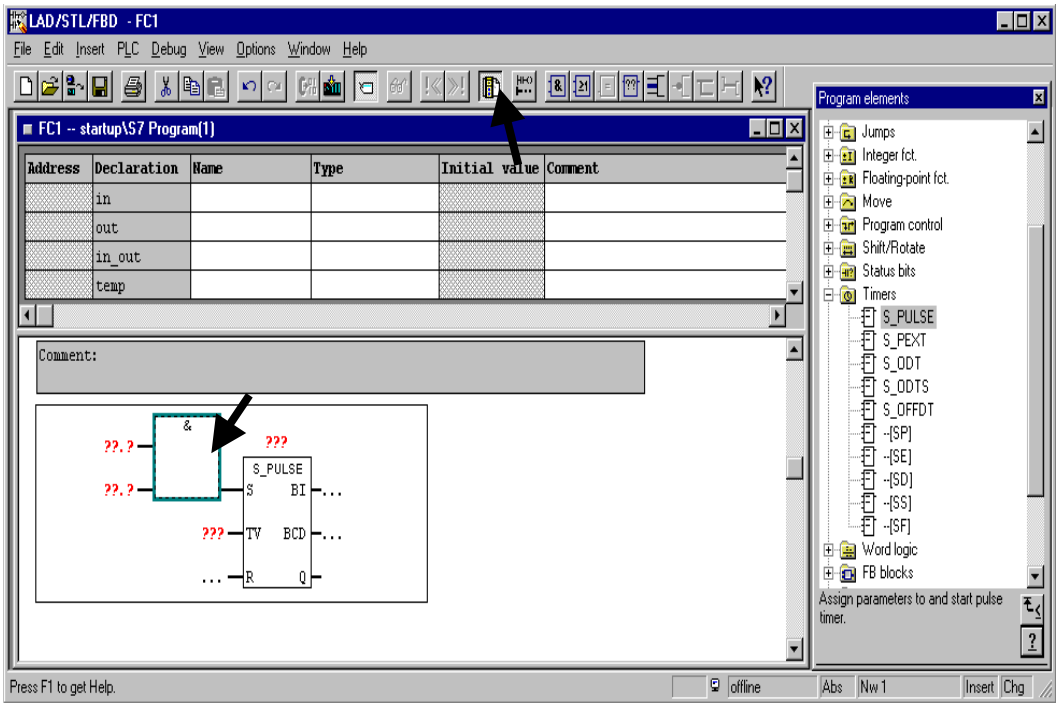


Note: The time as a pulse **S_PULSE**, as used above, holds for as long as the time is given. When the set input **S** is '1', the output **Q** is '1'. If the time is given with a **TV** or signal level **S** of '0', then the output **Q** will be '0'.

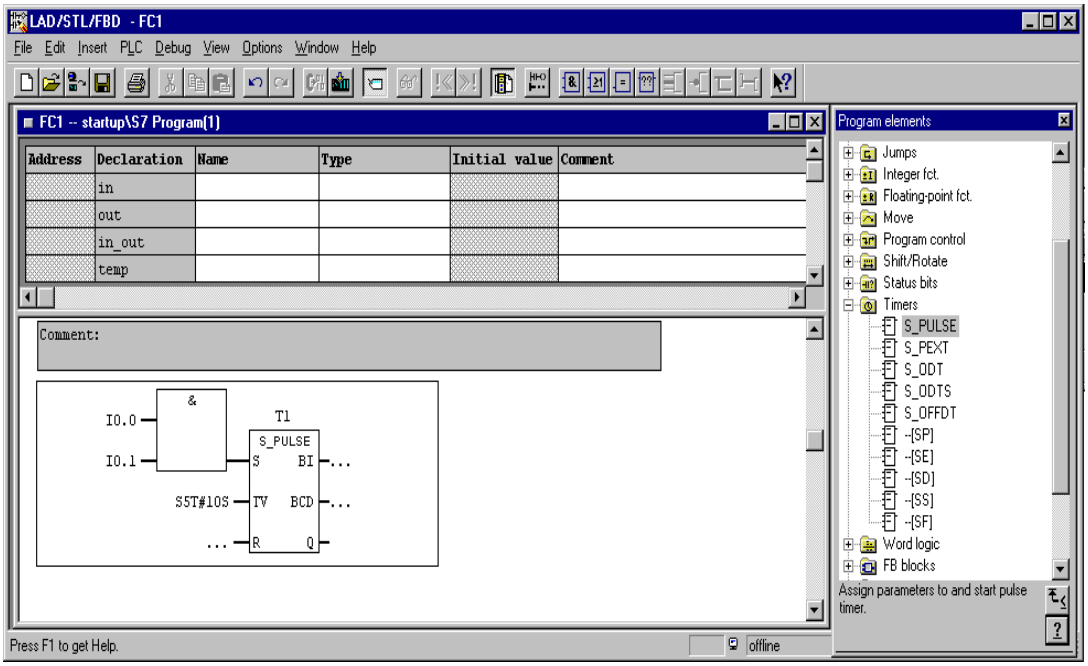
6. The operation S_PULSE is now inserted into the first network, by placing the cursor over S_PULSE, clicking and then holding the mouse button, dragging the S_PULSE to the network field, and then releasing the mouse button (→ S_PULSE).



7. Operations that are frequently used e.g. the AND-Operation can be found in the menu bar. They are inserted by first clicking on the input **S** by the timer, and then on the button (→ S →).



8. The timers need to be designed with **T1** and be registered with a value of 10 seconds in the S5 Time-Format **S5T#10s**. In addition, the inputs should be registered as **I 0.0** and **I 0.1** at the AND- Operation as well as in the network and comment blocks. (→ T1 → S5T#10s → I0.0 → I0.1 → Comment)

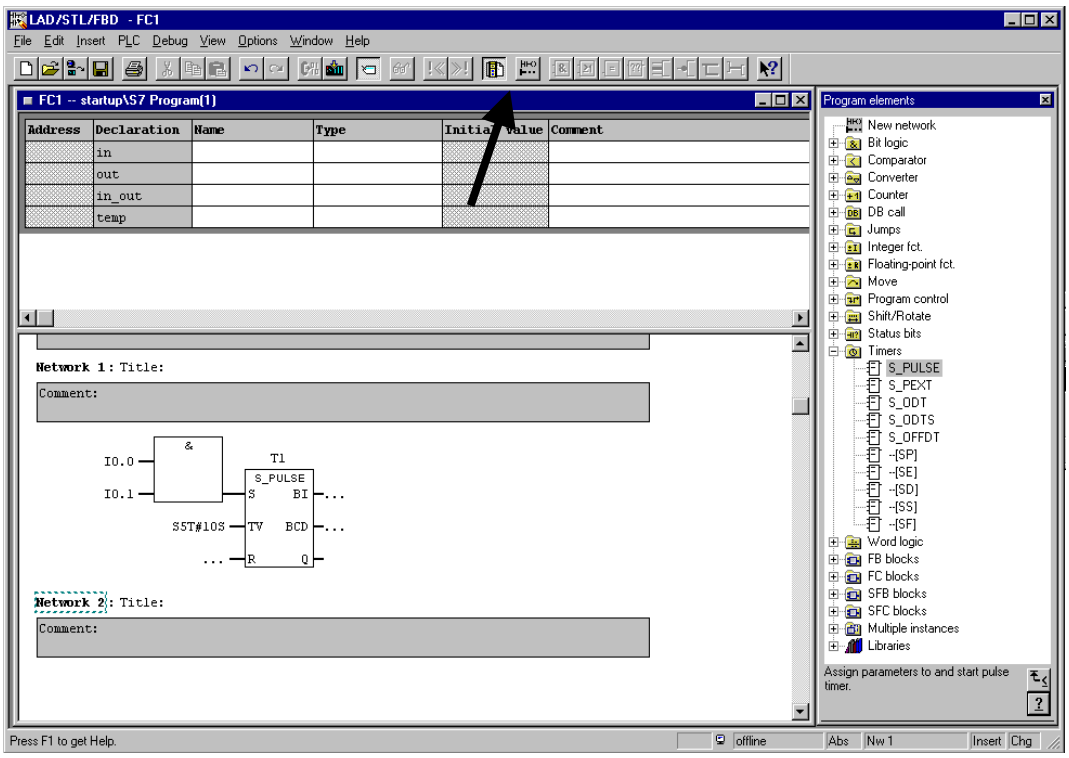


Note: In order to give a time to a timer, the following syntax must be used:

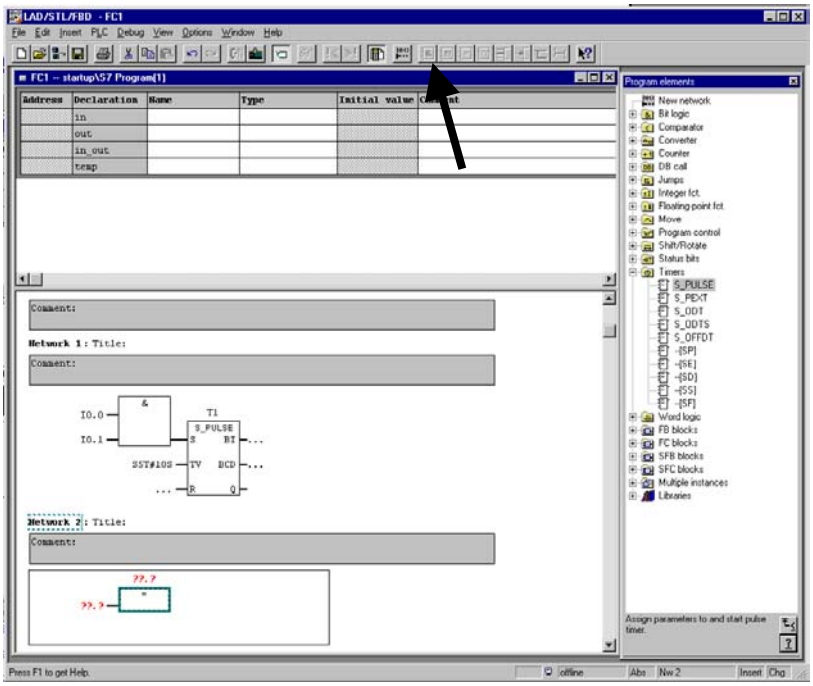
S5T# 10s

S5T# is the first format and directly after it the time (here **10** seconds) is entered. The time can also be given as milliseconds (ms), minutes (m), and hours (H). These units can also be indicated together (e.g. S5T#3M_3S).

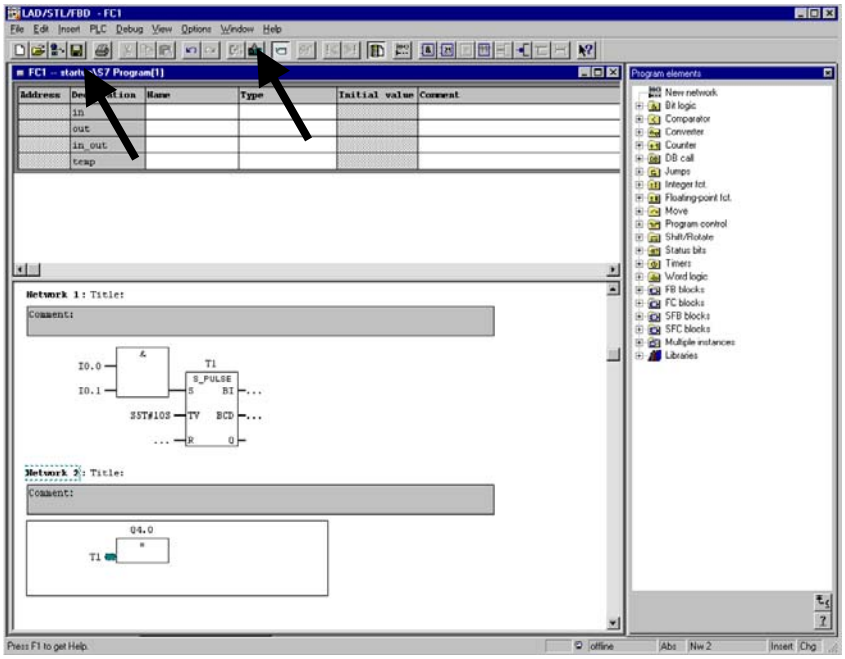
9. A further network is registered by clicking on the symbol in the menu bar. (→)



10. Insert an assignment by clicking once on the symbol (→)

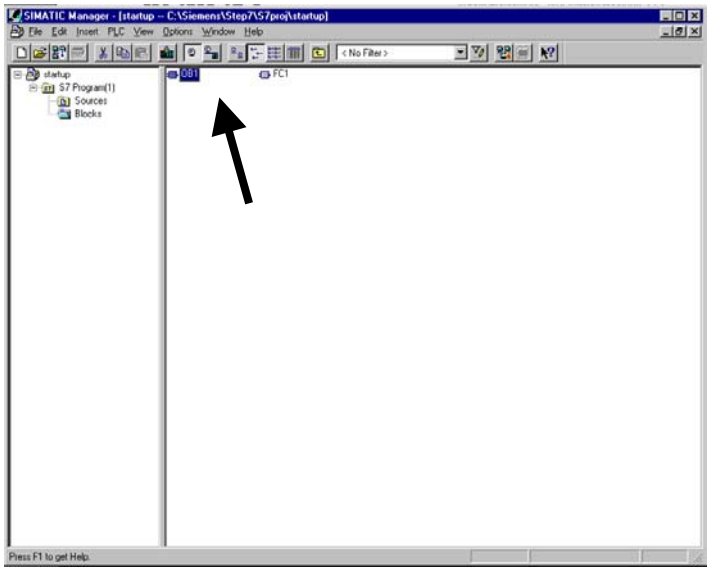


11. The assignment should apply for the output **Q4.0** and will take place as long as the Timer's signal **T1** is 'High'. These two operands must be inserted before the FC1 can be stored and loaded into the PLC. (→ Q 4.0 → T1 → →)

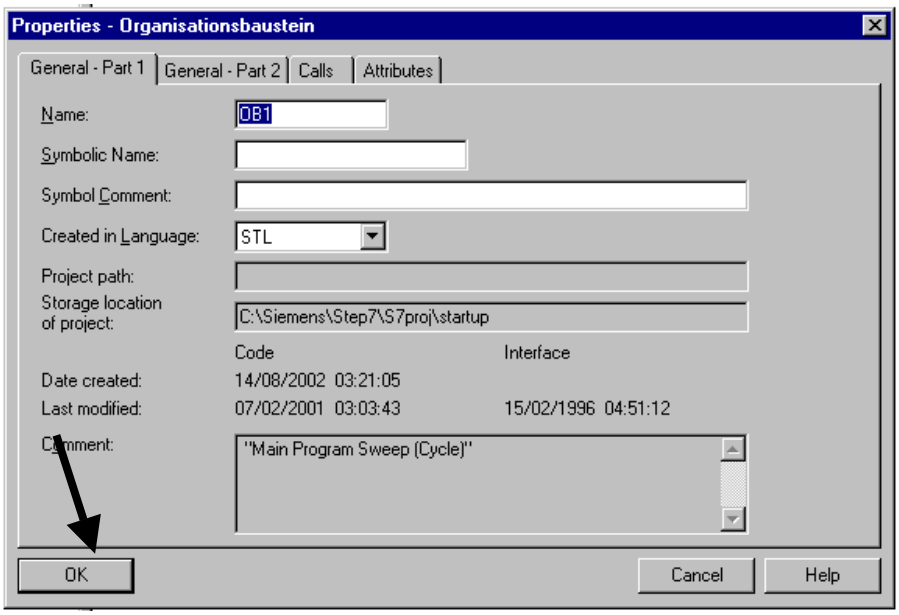


Caution: The editor program „LAD/STL/FBD" was not closed. It can be closed by switching to the SIMATIC Manager in the foot line (Point 12) or by the calling of OB1 with the function "OPEN".

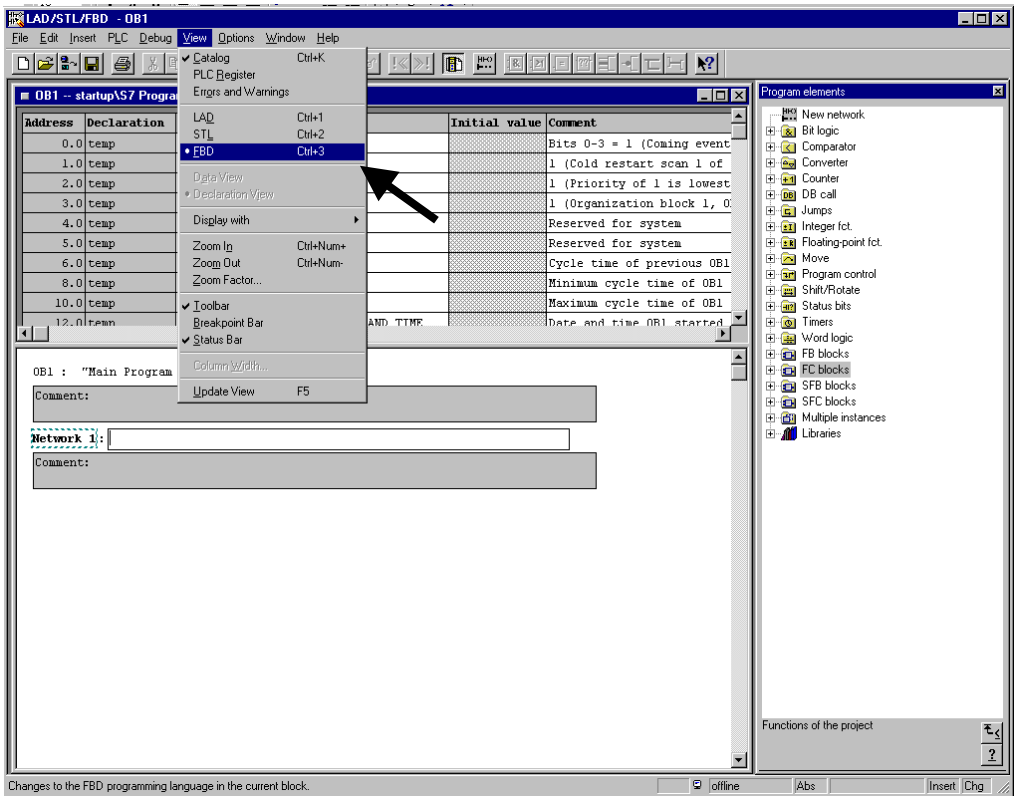
12. To program the **OB1** of the FC-Call, double click on it in the **SIMATIC Manager** (→ SIMATIC Manager → OB1).



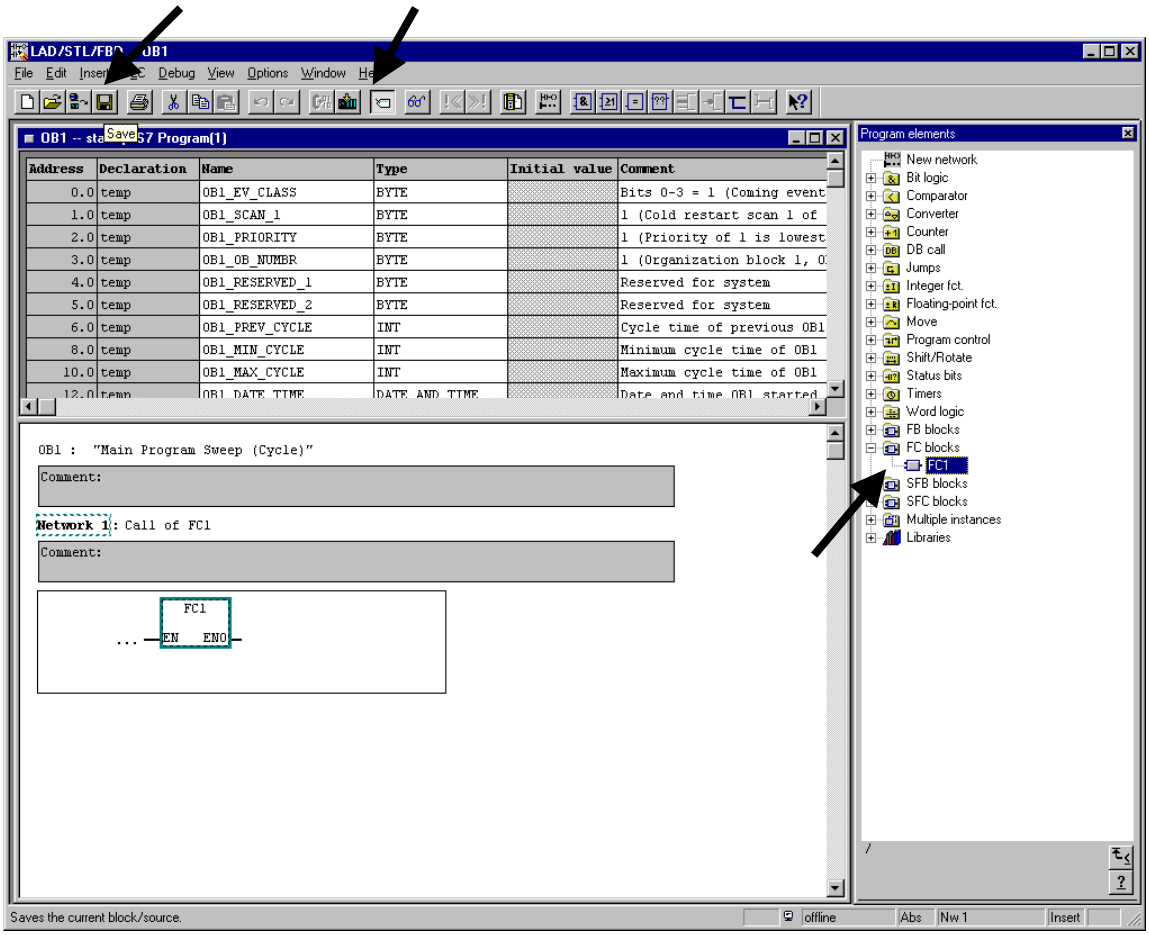
13. The properties of the OB1 are held and accepted with **OK** (→ OK).



14. In the Editor, the view **LAD/FBD/STL** can be changed to **FBD** by clicking on **View**, and then **FBD** for the programming language function diagram.(→ View → FBD)

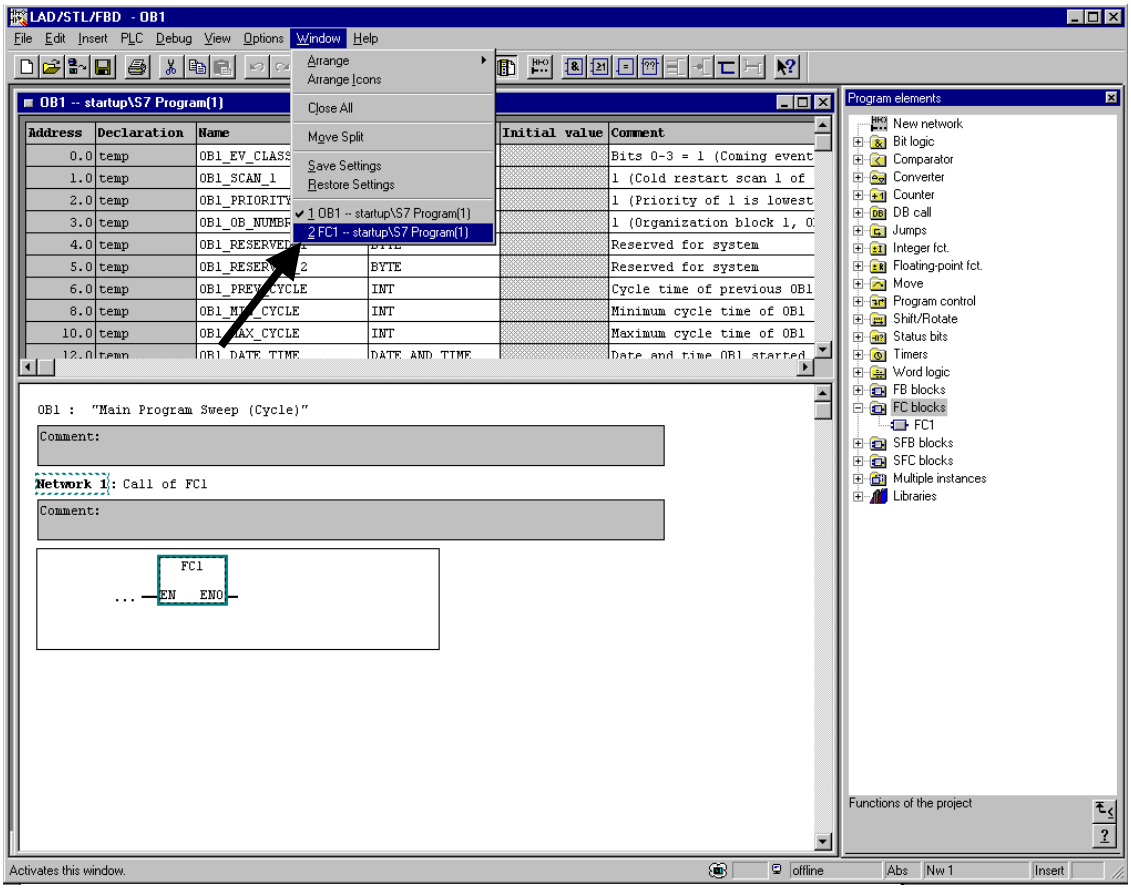


15. The OB1 can be saved by first double clicking on the **FC1** (found under the **FC Block**) in OB1's Network 1 catalog, then clicking the save button and then compiling it with the download button . (→ FC Block → FC1 → →)



10. STEP 7- PROGRAM DEBUGGING IN THE CPU

1. In order to observe the program in FC1, **LAD/FBD/STL**'s block must be changed in the editor under **Window** and then from OB1 to **FC1**. (→ Window → FC1)



2. The program in the FC1 can be observed with a mouse click on the eyeglass symbol . The execution of the timer is demonstrated as the signal state of an input and output. (→)