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

Appendix I Fundamentals of PLC programming with SIMATIC S7-300

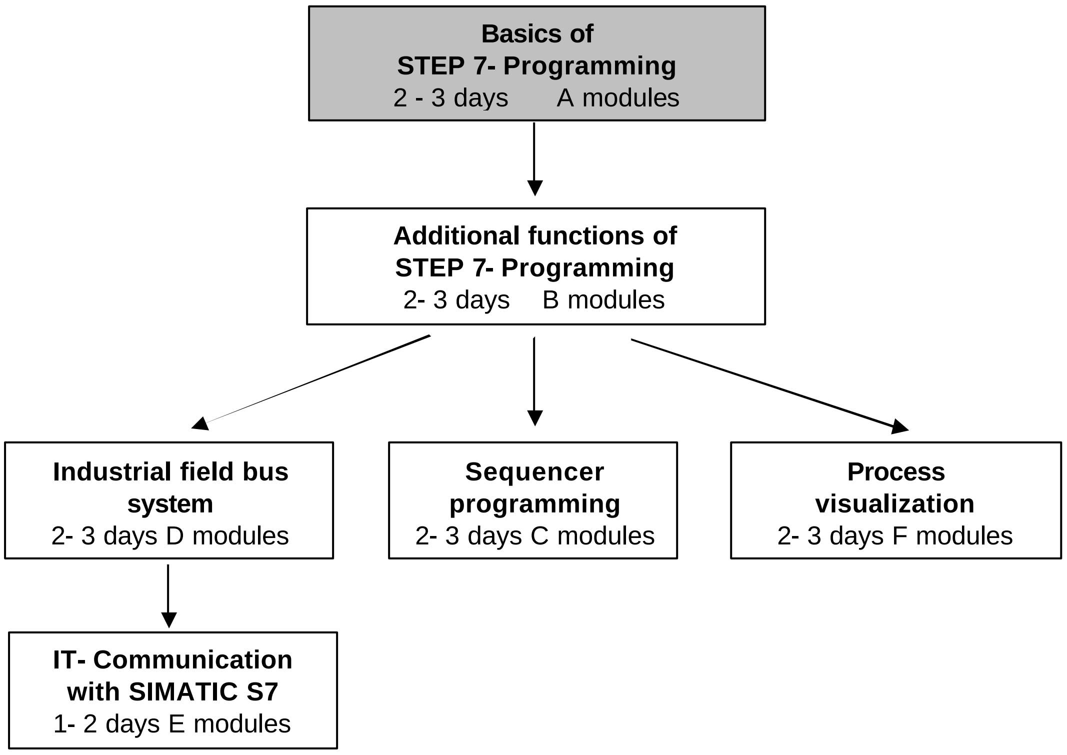
We thank the company Michael Dziallas Engineering and the instructors of vocational schools as well as further persons for the support with the production of the document.

PAGE:

PAGE:

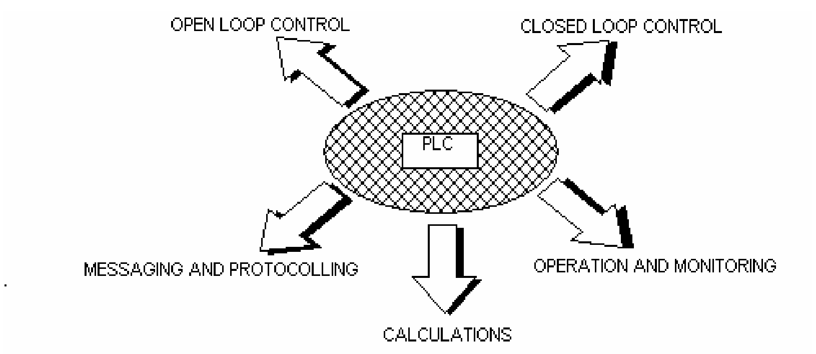
1. FORWARD

Appendix I is the requirement for the processing of the module of the theme **Basics of STEP 7-Programming**.


Basics of STEP 7- Programming 2 - 3 days A modules
Additional functions of STEP 7- Programming 2- 3 days B modules
Industrial field bus system 2- 3 days D modules
Sequencer programming 2- 3 days C modules
Process visualization 2- 3 days F modules
IT- Communication with SIMATIC S7 1- 2 days E modules

Learning goal:

Programmable logic controllers (PLC in brief), are considered today as a principal item of automation. With these controllers, most diverse tasks of automation can be implemented depending on the definition of the problem.



These documents give you an overview of the automation system SIMATIC S7-300 and the appropriate programming software STEP 7.

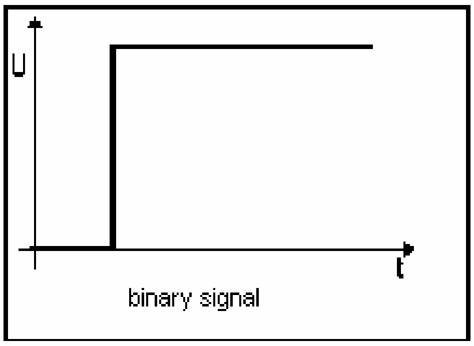
Requirements:
Since the basics are found within this appendix, no special requirements are necessary.

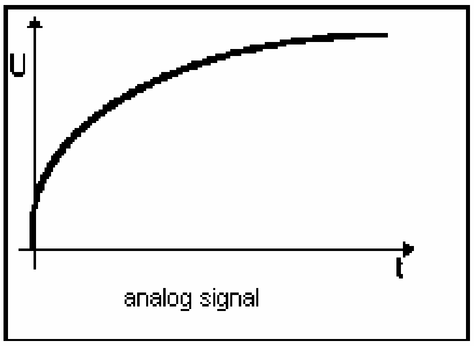
2. FUNCTION AND DESIGN OF A PLC

The controller has the task of leading single operations of a machine or a machine plant that depend on sensor signals after a given function execution.

2.1 TYPES OF SIGNALS IN CONTROL SYSTEM TECHNOLOGY

The electrical signals which are applied at the inputs and outputs can be, in principle, divided into two different groups:





2.1.1 BINARY SIGNAL

Binary signals can take the value of 2 possible states. They are as follows:

Signal state “1“	=	voltage available	=	e.g. Switch on
Signal state “0“	=	voltage not available	=	e.g. Switch off

In control engineering, a frequent DC voltage of 24V is used as a “control supply voltage“ A voltage level of + 24V at an input clamp means that the signal status is “1“ for this input. Accordingly 0V means that the signal status is “0“. In addition to a signal status, another logical assignment of the sensor is important. It’s a matter of whether the transmitter is a “normally closed” contact or a “normally open” contact. When it is operated, a “normally closed” contact supplies a signal status of “0“ in the “active case“. One calls this switching behavior “active 0“ or “active low“. A “normally open” contact is “active 1”/“active high“, and supplies a “1“ signal, when it is operated.
In closed loop control, sensor signals are “active 1“. A typical application for an “active 0“ transmitter is an emergency stop button. An emergency stop button is always on (current flows through it) in the non actuated state (emergency stop button not pressed). It supplies a signal of “1“(i.e. wire break safety device) to the attached input. If operation of an emergency stop button is to implement a certain reaction (e.g. all valves close), then it must be triggered with a signal status of “0“

Equivalent binary digits:
A binary signal can only take the two values (signal statuses) “0“ or “1“. Such a binary signal is also designated as an equivalent binary digit and receives the designation of “Bit“ in the technical language book. Several binary signals result in a digital signal after a certain assignment (code). While a binary signal only provides a grouping of a bivalent size/e.g. for door open/door close), one can form e.g. a number or digit as digital information by the bundling of equivalent binary digits.
The summarization of n-equivalent binary digits allows the representation of 2n different combinations.

One can show 4 different types of information with e.g. 2 equivalent binary digits 2x2:

0	0	Configuration 1 (e.g. Both switches open)
0	1	Configuration 2 (Switch 1 closed / Switch 2 open)
1	0	Configuration 3 (Switch 1 open / Switch 2 closed)
1	1	Configuration 4 (both switches closed)

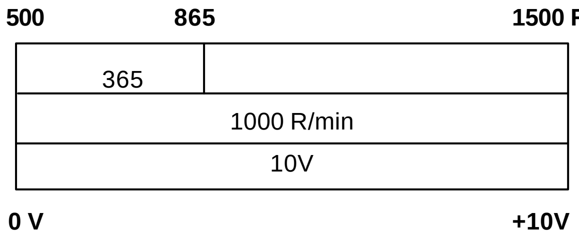
2.1.2 ANALOG SIGNAL

Contrary to a binary signal that can accept only signal statuses („Voltage available +24V“ and “Voltage available 0V“, there are similar signals that can take many values within a certain range when desired. A typical example of an analog encoder is a potentiometer. Depending upon the position of the rotary button, any resistance can be adjusted here up to a maximum value.

Examples of analog measurements in control system technology:

- Temperature -50 ... +150°C
- Current flow 0 ... 200l/min
- Number of revolutions 500 ... 1500 R/min
- Etc.

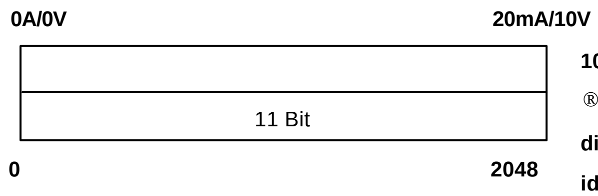
These measurements, with the help of a transducer in electrical voltages, are converted to currents or resistances. E.g. if a number of revolutions is collected, the speed range can be converted over a transducer from 500... 1500 R/min into a voltage range from 0... +10V. At a measured number of revolutions of 865 R/min, the transducer would give out a voltage level of + 3.65V.



10V: 1000 R/min = 0.01 V/R/min

365 R/min x 0.01 V/R/min = 3.65V

If similar measurements are processed with a PLC, then the input must be converted into digital information to a voltage, current or resistance value. One calls this transformation analog to digital conversion (A/D conversion). This means, that e.g. a voltage level of 3.65V is deposited as information into a set of equivalent binary digits. The more equivalent binary digits for the digital representation will be used, in order for the resolution to be finer. If one would have e.g. only 1 bit available for the voltage range 0... +10V, only one statement could be met, if the measured voltage is in the range 0.. +5V or +5V....+10V. With 2 bits, the range can be partitioned into 4 single areas, (0... 2.5/2.5... 5/5... 7.5/7.5... 10V). Usually in control engineering, the A/d converter is changed with the 8th or 11th bit. 256 single areas are normally provided, but with 8 or 11 bits, you can have 2048 single areas.



10V: 2048 = 0,0048828

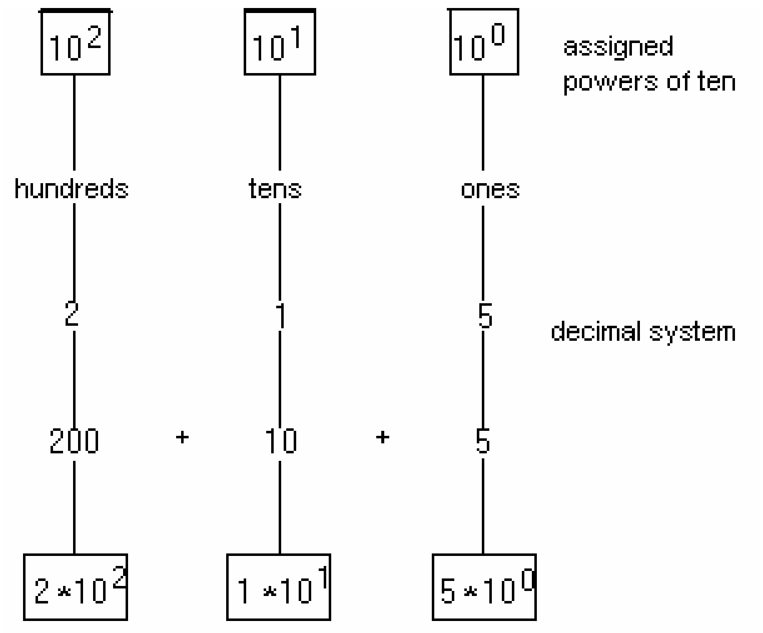
® Voltages with differences <5mV can be identified

2.2 NUMBER SYSTEMS

For the processing of the addresses of memory cells, inputs, outputs, times, bit memories etc. by a programmable controller, the binary system is used instead of the decimal system.

2.2.1 DECIMAL SYSTEM

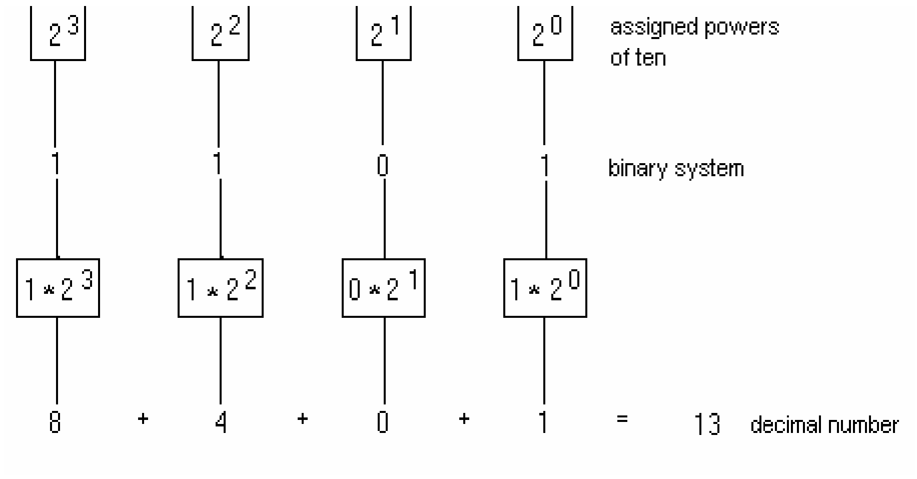
In order to understand the binary number system, it is first necessary to consider the decimal system. Here the number of 215 is to be subdivided. Thereby the hundreds represent the 2, the 1 stand for the tens and the 5 for the ones. Actually, one would have to write 215 in such a way: 200+10+5. If one writes down the expression 200+10+5, with the help of the powers of ten as explained earlier, then one states that each place is assigned a power of ten within the number.



Each number within the decimal system is assigned a power of ten.

2.2.2 BINARY SYSTEM

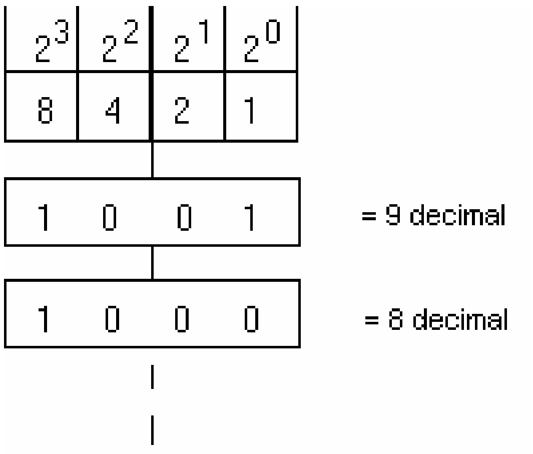
The binary number system uses only the numbers 0 and 1, which are easily represented and evaluated in data processing. Thus it is called a binary number system. The values of a dual number are assigned the power-of-two numbers, as represented below.



Each number assigned within the binary number system is a power-of-two.

2.2.3 BCD - CODE (8-4-2-1-CODE)

In order to represent large numerical values more clearly, the BCD code (binary coded decimal number) is frequently used. The decimal numbers are represented with the help of the binary number system. The decimal digit with the highest value is the 9. One needs to demonstrate the 9 with power-of-two numbers until 2^3, thus using 4 places for the representation of the number.



Because the representation of the largest decimal digit requires 4 binary places, a four-place unit called a tetrad, is used for each decimal digit. The BCD code is thus a 4-Bit-Code

Each decimal number is coded individually. The number of 285 consists e.g. of three decimal digits. Each decimal digit appears in the BCD code as a four-place unit (tetrad).

2	8	5
0010	1000	0101

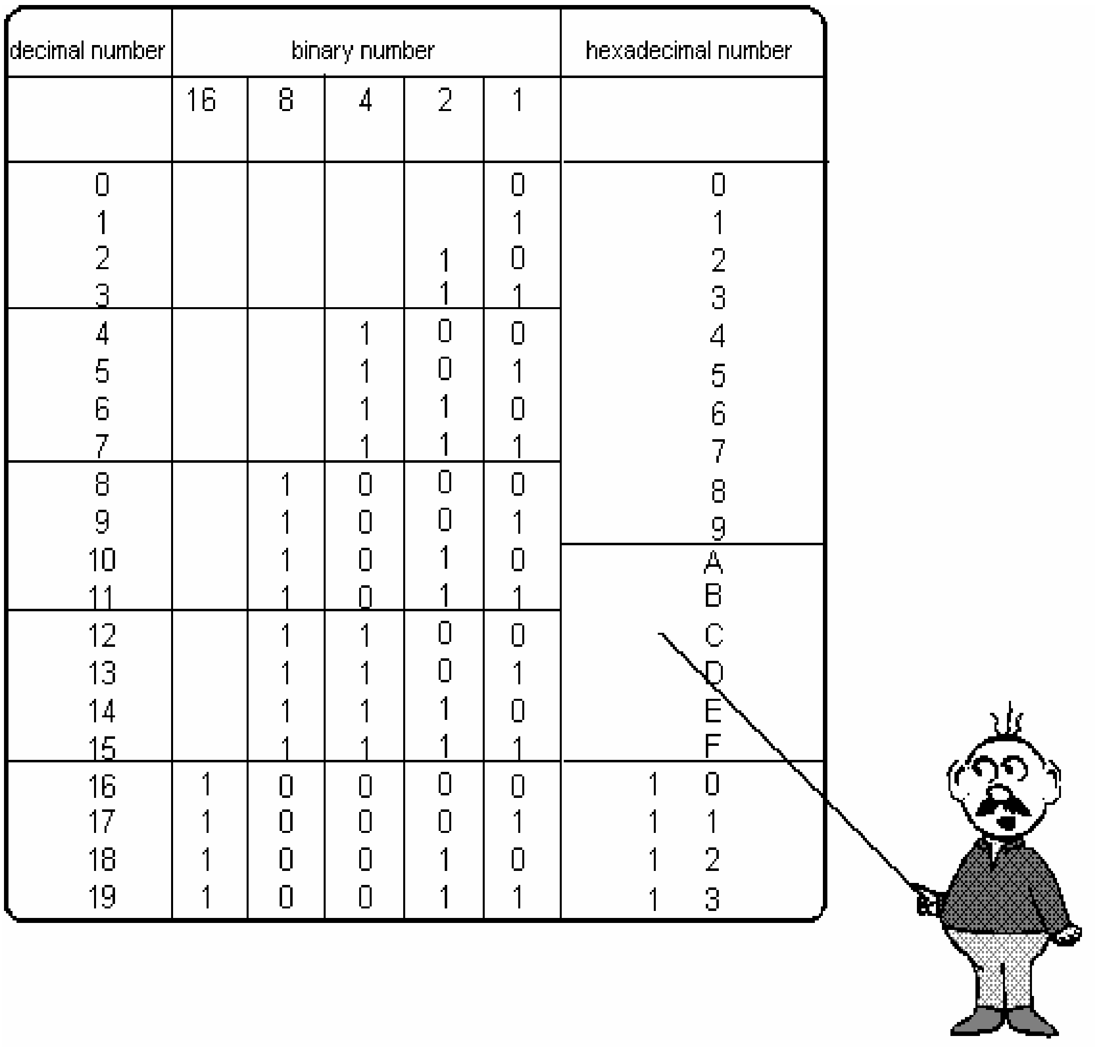
Each decimal digit is represented by an individually coded tetrad.

2.2.4 HEXADECIMAL NUMBER SYSTEM

The hexadecimal number system belongs to the notational systems because value powers of the number 16 are used. The hexadecimal number system is thus a sixteen count system. Each place within a hexadecimal number is assigned a sixteenth power. One needs altogether 16 numbers, including the zero. For the numbers 0 to 9 one uses the decimal system, and for the numbers 10 to 15 the letters A, B, C, D, E and F are used.

Each digit within a hexadecimal number system is assigned a power of the number 16.

2.2.5 DEMONSTRATION OF THE NUMBER SYSTEMS



decimal number	binary number					hexadecimal number
	16	8	4	2	1	
0					0	0
1					1	1
2				1	0	2
3				1	1	3
4			1	0	0	4
5			1	0	1	5
6			1	1	0	6
7			1	1	1	7
8		1	0	0	0	8
9		1	0	0	1	9
10		1	0	1	0	A
11		1	0	1	1	B
12		1	1	0	0	C
13		1	1	0	1	D
14		1	1	1	0	E
15		1	1	1	1	F
16	1	0	0	0	0	1 0
17	1	0	0	0	1	1 1
18	1	0	0	1	0	1 2
19	1	0	0	1	1	1 3

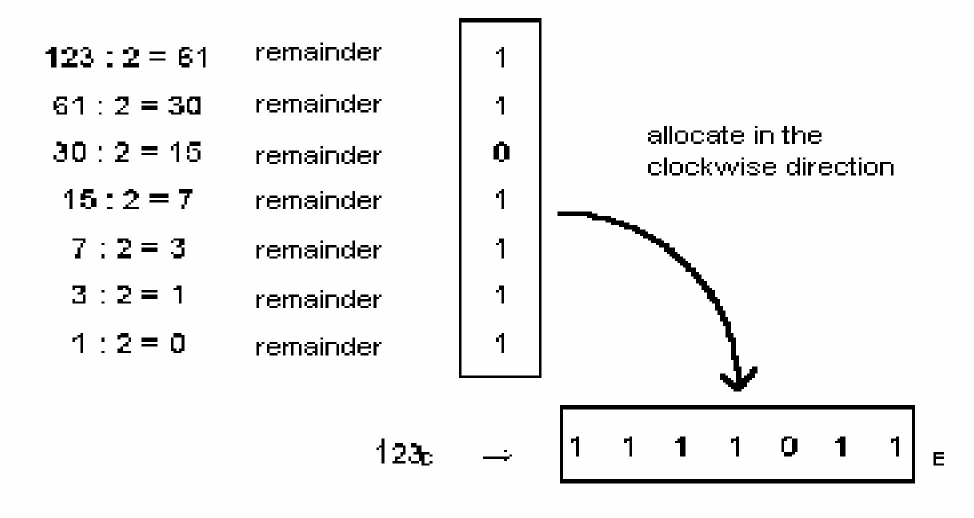
2.2.6 CONVERSION RULES

The transformation of the different number systems are based on simple rules. These rules should be controlled by the PLC users, since they are often used in handling this technology. For the use of a number system on which a given number is based, an index sign is placed at the end of a number. Here “D“ stands for decimal, “B“ for binary, and “H“ for hexadecimal. This marking is often necessary to identify a number system because in each system, different values can be obtained when the same number is used. (e.g.. “111“ in the decimal system has the value 111_D (one hundred eleven). In the binary system it would be 111_B ,which is the decimal value 7 (1x20 + 1x 21 + 1x22). As a hexadecimal number, 111_H would be the decimal value 273 (1x160 + 1x161 + 1x 162).

Converting decimal → binary

Integral decimal numbers are divided by the base 2 until the result of zero is obtained. The remainder obtained with the division (0 or 1) results in a binary number. One needs to also consider the direction that the “remainders“ are written in. The remainder of the first division is the first right bit (low order width unit bit).

e.g.: The decimal number 123 is to be changed into an appropriate dual number.



123 : 2 = 61	remainder	1
61 : 2 = 30	remainder	1
30 : 2 = 15	remainder	0
15 : 2 = 7	remainder	1
7 : 2 = 3	remainder	1
3 : 2 = 1	remainder	1
1 : 2 = 0	remainder	1

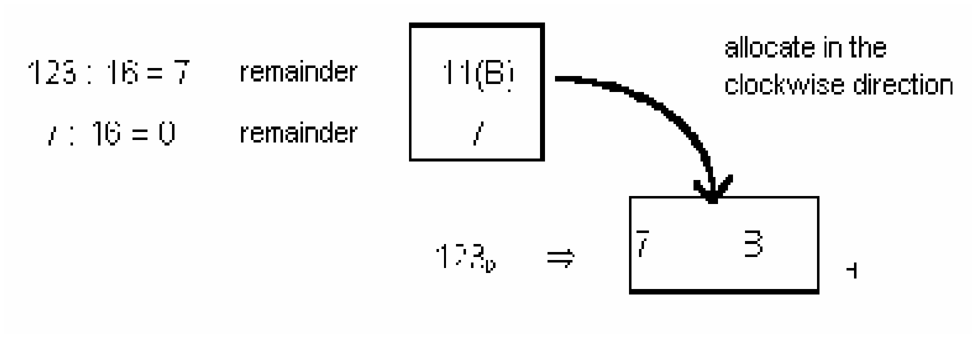
Pattern:

$$\begin{array}{ccccccccccccccc} 1 & & 1 & & 1 & & 1 & & 0 & & 1 & & 1 & & \\ 1x2^6 & + & 1x2^5 & + & 1x2^4 & + & 1x2^3 & + & 0x2^2 & + & 1x2^1 & + & 1x2^0 & & \\ 64 & + & 32 & + & 16 & + & 8 & + & 0 & + & 2 & + & 1 & = & \underline{123} \end{array}$$

Converting decimal → hexadecimal

This transformation is performed exactly like the decimal → binary transformation. The only difference is that instead of using base 2, we use base 16. Thus the number must be divided by 16 rather than by 2.

E.g. The decimal number 123 is to be changed into the appropriate hex number.



Pattern:

7		B		
$7x16^1$	+	$11x16^0$		
112	+	11	=	<u>123</u>

Converting binary → hexadecimal

For the transformation of a dual number into a Hex number, one could first determine the decimal value of the binary number (addition of the priorities). This decimal number could then be changed into a hexadecimal number with the help of the division:16. In addition, there is the possibility of determining the associated hex value directly from the binary number. First of all, the binary number is divided from the right beginning in the quadripartite groups. Everyone of the determined quadripartite groups results in a number of the hexadecimal number system. If necessary, fill the missing bits on the left hand side with zeros
e.g. The binary number 1111011 is to be changed directly into a hex number.

$1\ 1\ 1\ 1\ 0\ 1\ 1_B$

$\boxed{0\quad 1\quad 1\quad 1}\quad \boxed{1\quad 0\quad 1\quad 1}$

$0x2^3 + 1x2^2 + 1x2^1 + 1x2^0 \quad 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0$

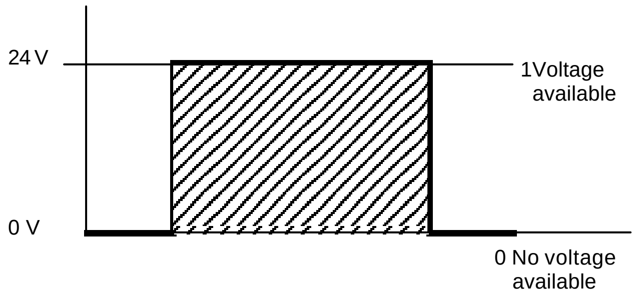
$\boxed{7\qquad\qquad B}_H$

2.3 TERMS FROM COMPUTER SCIENCE

In connection with programmable controllers, terms such as BIT, BYTE and WORD are frequently used in the explanation of data and/or data processing.

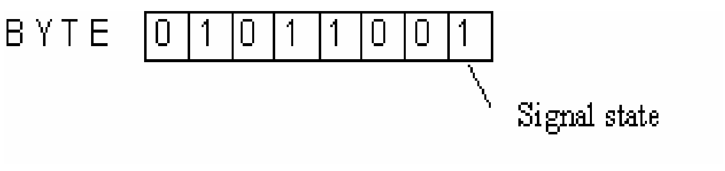
2.3.1 BIT

Bit is the abbreviation for binary digit. The BIT is the smallest binary (bivalent) information unit, which can accept a signal status of “1“ or “0“.



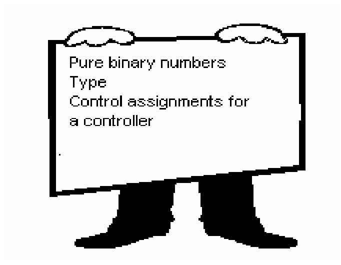
2.3.2 BYTE

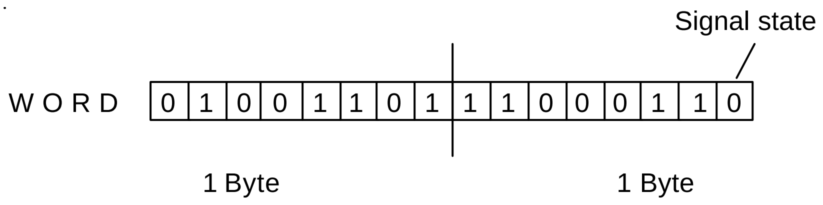
For a unit of 8 binary characters, the term BYTE is used. A byte has the size of 8 bits.



2.3.3 WORD

A word is a sequence of binary characters, which is regarded as a unit in a specific connection. The word length corresponds to the number from 16 binary characters. With words, the following can be represented:





A word also has the size of 2 bytes or 16 bits.

2.3.4 DOUBLE-WORD

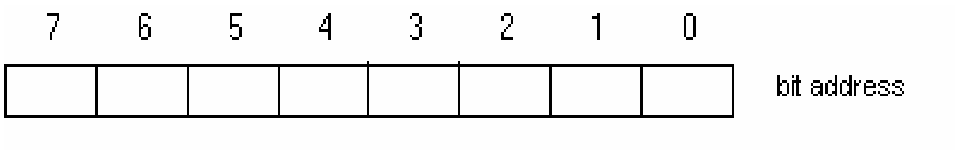
A double-word corresponds to the word length of 32 binary characters.

A double-word also has the size of 2 words, 4 bytes, or 32 bits.

Further units are kilo-bit or kilo-byte, which stand for $2^{10,}$ or 1024 bits, and the mega-bit or mega-byte which stands for 1024 kilo-bits.

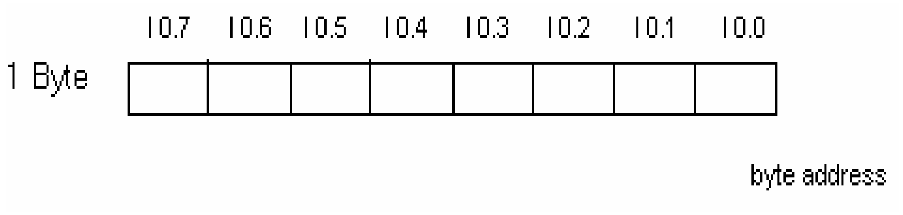
2.3.5 BIT ADDRESS

So that individual bits can be addressed within a byte, each individual bit is assigned a bit location. In each byte the bit gets the bit location 7 on the leftmost side and the bit location 0 on the rightmost side.



2.3.6 BYTE ADDRESS

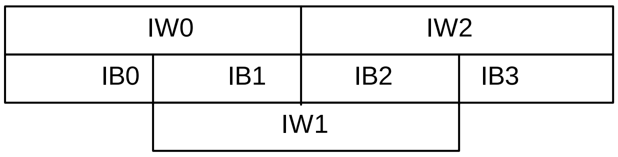
The individual bytes also receive numbers called byte displacements. Additionally, the operand is still marked, so that e.g. IB 2 stands for input byte 2 and QB4 stands for output byte 4. Individual bits are clearly addressed by the combination of bit and byte displacement. The bit location is separated from the byte displacement by one point. The bit location stands to the right of the point, and the byte displacement to the left.



2.3.7 WORD ADDRESS

The numbering of words results in a word address.

Note: The word address is always the smallest address of the two pertinent bytes when using words, e.g. input word(IW),output word(QW), bit memory word(MW), etc. (e.g. With a word that comes from IB2 and IB3, the address is IW2).



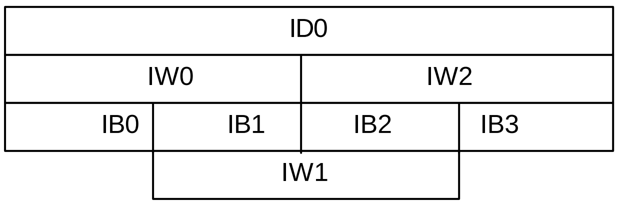
Word address

Note: During word processing it is to be noted that e.g. the input word 0 and the input word 1 are in a byte overlap. In addition, when counting bits, one begins at the rightmost bit. For example the bit0 from IW1 is the bit of I2.0, bit1 is I2.1.... bit7 is I 2.7, bit8 is I1.0.... bit15 ois I1.7. A jump exists between the bits 7 and 8.

2.3.8 DOUBLE-WORD ADDRESS

The numbering of double-words results in a double-word address.

Note: When using double-words e.g. ID,QD,MD etc. the double-word address is the smaller word address of the two pertinent word.



Double word address

Forward **Function and design of a PLC** Programming language STEP 7

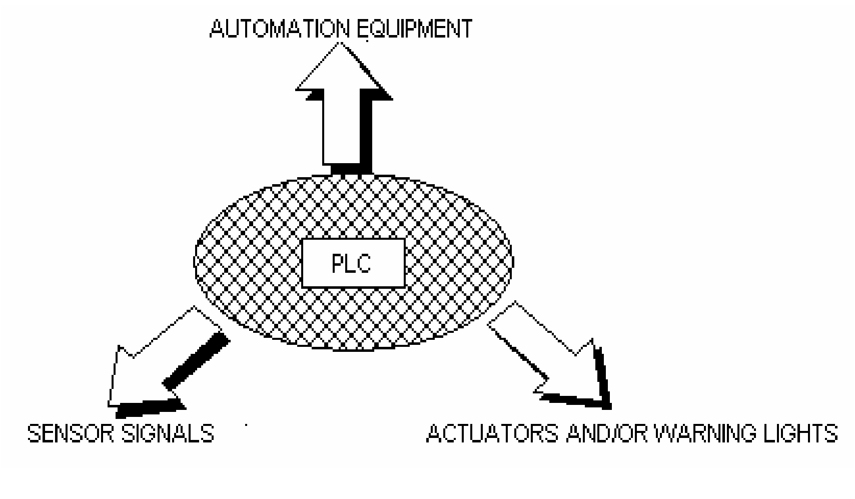
2.4 CONFIGURATION OF A PLC

Programmable logic controllers are manufactured devices in series, which are designed with general problems in mind. All that is required for control engineering are logic elements, memory functions, time elements, counters, etc., which are designated from the manufacturer and are connected to a functional controller by programmed code. The controllers are offered in different functional units. They differ mainly by the number of possible:

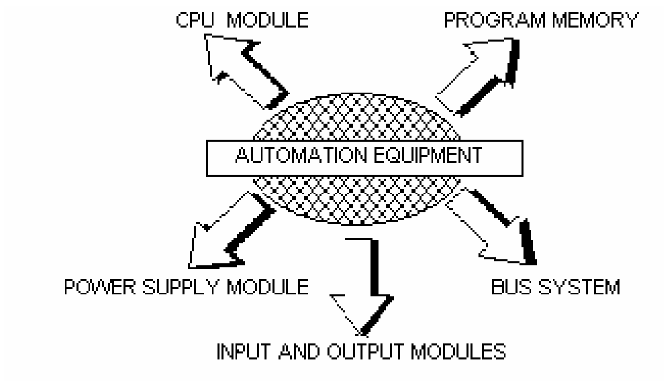
inputs and outputs,
memory space,
counters,
timers,
bit memory functions,
other functions,
operating speed,
as well as the kind of program processing.

Larger controllers are individually built in modular construction from individual modules. With this modular system, controllers run from the basic hardware, that PLC systems arrange, which can be adapted to the application accordingly. For smaller control tasks, compactly developed controllers are offered. They represent self closed units with a pre-defined number of inputs and outputs.

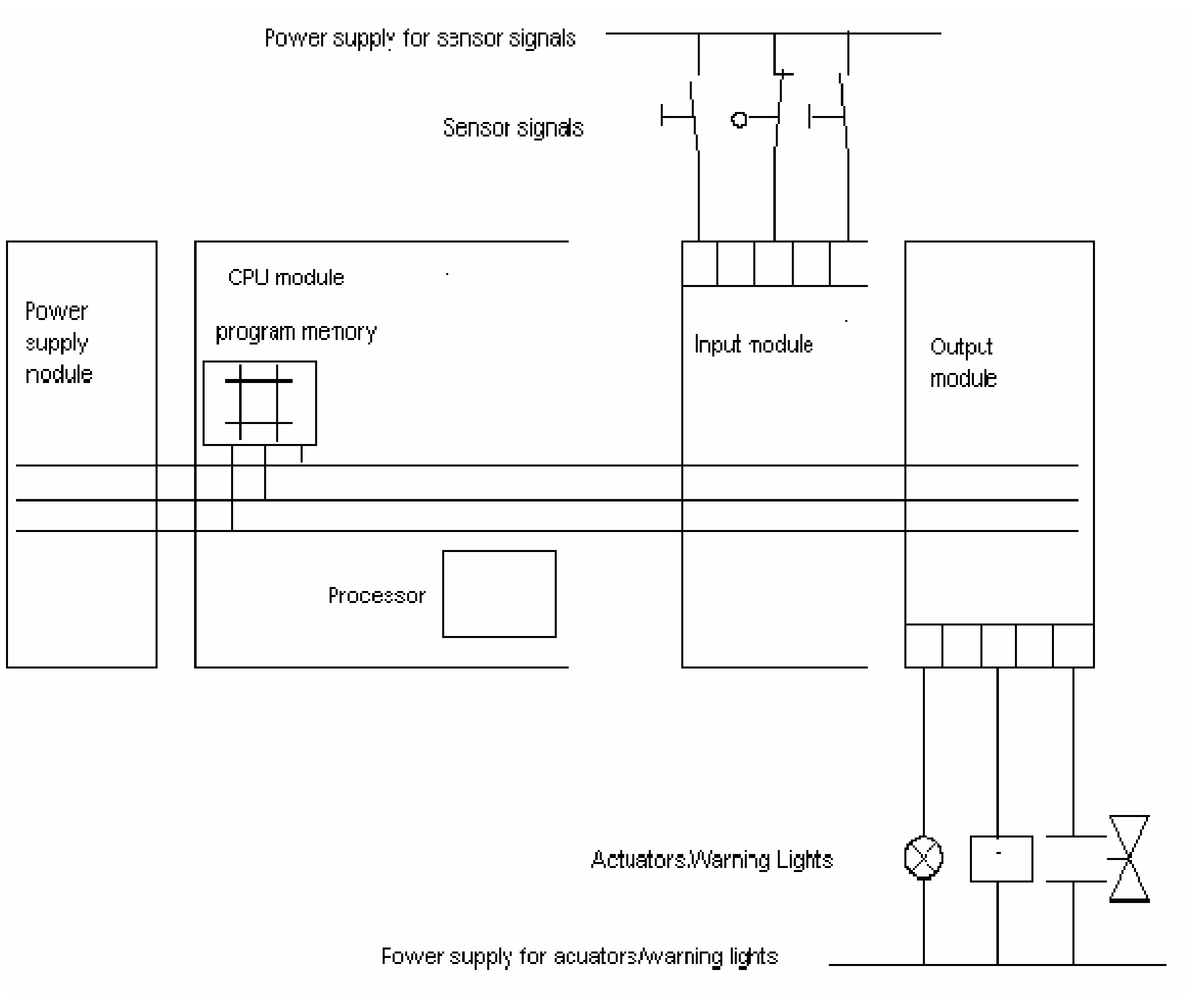
The following, in principle, belong to a programming controller:



Automation equipment essentially contains:



Design of a program controller:

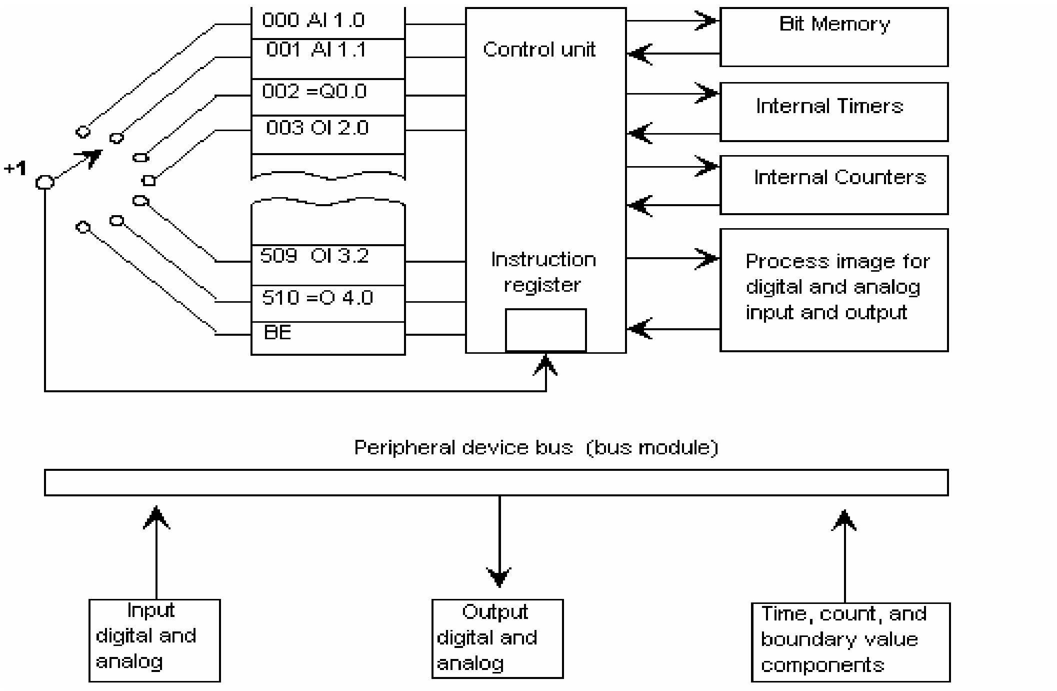


2.5 BASIC FUNCTION MODULES OF A PLC

2.5.1 THE CPU MODULE

The voltage coming from the sensor signals is switched to the pin board of the input device. In the CPU (central processing unit), the processor works on the program in the memory and queries whether the individual inputs of the equipment voltage have voltage or not. Dependent on this condition at the inputs and on the program in the memory, the processor instructs the output device to switch the voltage on the appropriate terminals of the terminal strip. Depending on the tensile state at the terminals of the output modules, the attached actuators and/or warning light are switched on or off.

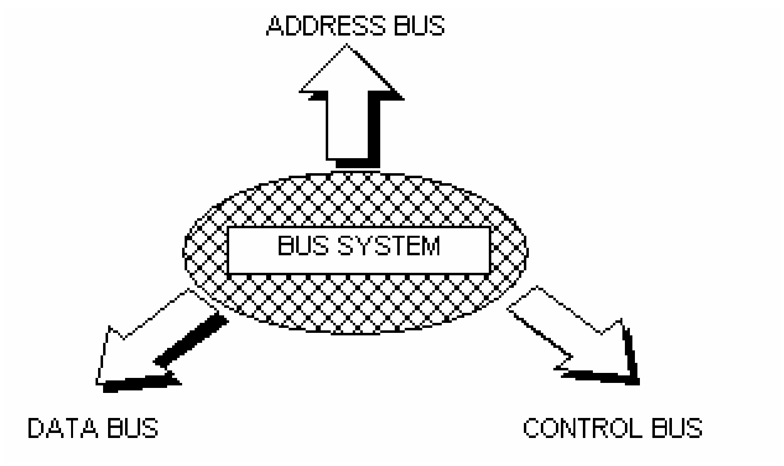
CPU of the PLC:



The address counter successively (serially) queries the program memory instruction for instructions and causes the program-dependent information to transfer from the program memory to the instruction register. All memory of a process is made up of registers. The control mechanism receives its instructions from the instruction register. While the control mechanism works on the current instruction, the address counter pushes the next instruction into the instruction register. After the operations follow the status transfer of the inputs into the processor image input table (PAE), the employment of the timers, counters, accumulators and the transmission of the result of logic operation(RLO) in the processor image output table (PAA). After the processing of the user program's block end (BE), if a module's end is recognized, then the transmission of the respective status follows from the PAA to the outputs.
The peripheral bus completes data exchange between the CPU and the peripherals. The analog and digital input and output devices, as well as a timer, counter, and comparator module belong to the peripheral bus.

2.5.2 THE BUS SYSTEM

The bus system is a collecting line for the transmission of signals. Thus the signal exchange is made in the automation equipment between the processor and the input and output by a process bus system. The bus consists of three parallel signal lines:



- The addresses on the individual modules are addressed with the address bus.
- Data will transfer e.g. from input to output devices with the data bus.
- Signals are conveyed with the control bus for the control and monitoring of the execution of functions within the automation equipment.

2.5.3 THE POWER SUPPLY MODULE

The power supply module produces the voltage for the electronic devices of the automation equipment from network voltage. The height of this voltage amounts to 24 V. Voltages for sensor signals, actuators and warning lights, which lie over 24 V, supply additional voltage for power supply units and/or control transformers.

2.5.4 PROGRAM MEMORY

Memory elements are elements, in which information can be deposited (stored) in the form of binary signals.
Semiconductor memory is used predominantly as program memory. A memory consists of 512, 1024, 2048 etc.. memory cells. It is usually recommended to indicate the capacity of the program memory (i.e. the number of memory cells) in multiples of 1 K (1 K stands for 1024). An instruction for control can be written (programmed) into each memory cell with the help of a programming device. Each binary cell of a memory cell can accept the signal status "1" or "0".

2.5.5 RAM

One designates read/write memory built in semiconductor technology with RAM. The individual storage locations are indicated by addresses, and with the help of the memory cells, can be accessed. The information is often arbitrarily written into the memory cells information. The information is picked out, without the information contents being lost. RAM -memory is however volatile memory i.e. their information contents are lost in case of failure of the supply voltage. RAM memory is electrically deleted. The internal main memory of a SIMATIC S7-300 is this type of RAM. The buffer battery, which can be inserted into the PLC, serves as a the safety device for this memory.

2.5.6 FLASH- EPROM

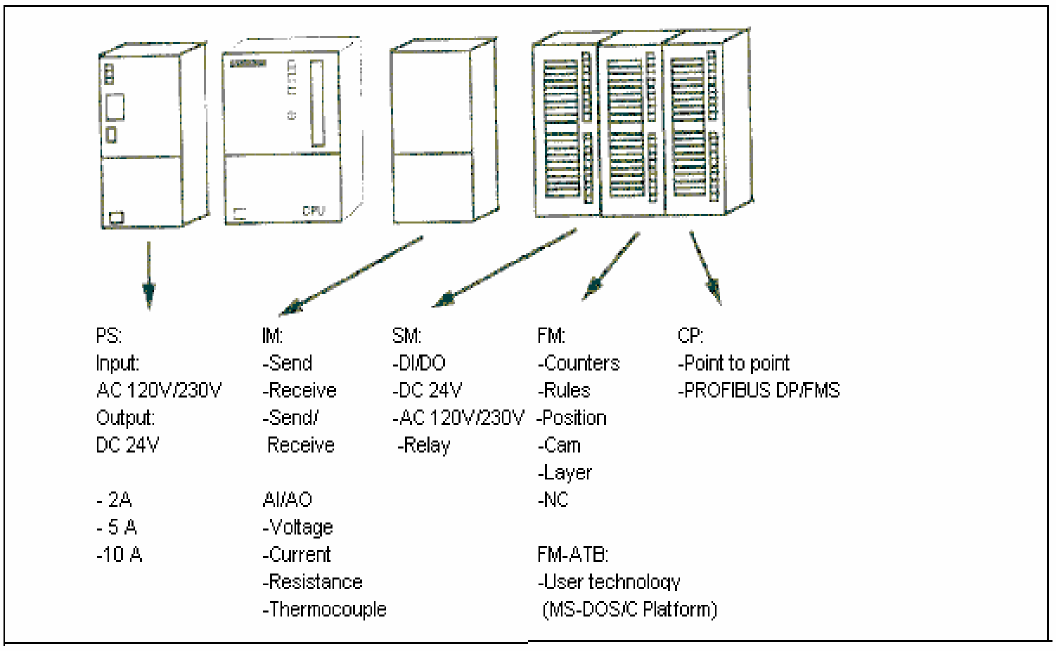
EPROM stands for ERASABLE, PROGRAMMABLE read-only memory. The contents of EPROM is erasable and again programmable by UV light or a voltage. It is well suited to be transported without overflow. In the SIMATIC S7 300 one has the possibility to secure a program on a Memory card (Flash-EPROM) with the PG and to also quickly restore the system after a power failure. This Flash EPROM is recorded and deleted with a voltage of 5V. thus it could also be erased if a power failure occurred when it was connected with the CPU.

2.6 AUTOMATION SYSTEM SIMATIC S7

SIMATIC S7 is a product design of the company SIEMENS and is used for the current series of the programmable controllers.
This SIMATIC S7 - computer family is a module in the automation concept for the manufacturing and process technique: Totally Integrated automation.

2.6.1 SIMATIC S7 - 300

Module Spectrum:
The following kinds of modules, together with the CPU, form the structure of a SIMATIC S7-300:



CPUs – A choice:

Listed below are the CPUs relevant for current training. There are still more efficient CPUs than what is shown and new CPUs can always be added. The list below is characterized by improved arithmetic performance. However, this characterization is not needed with the small programs in training.

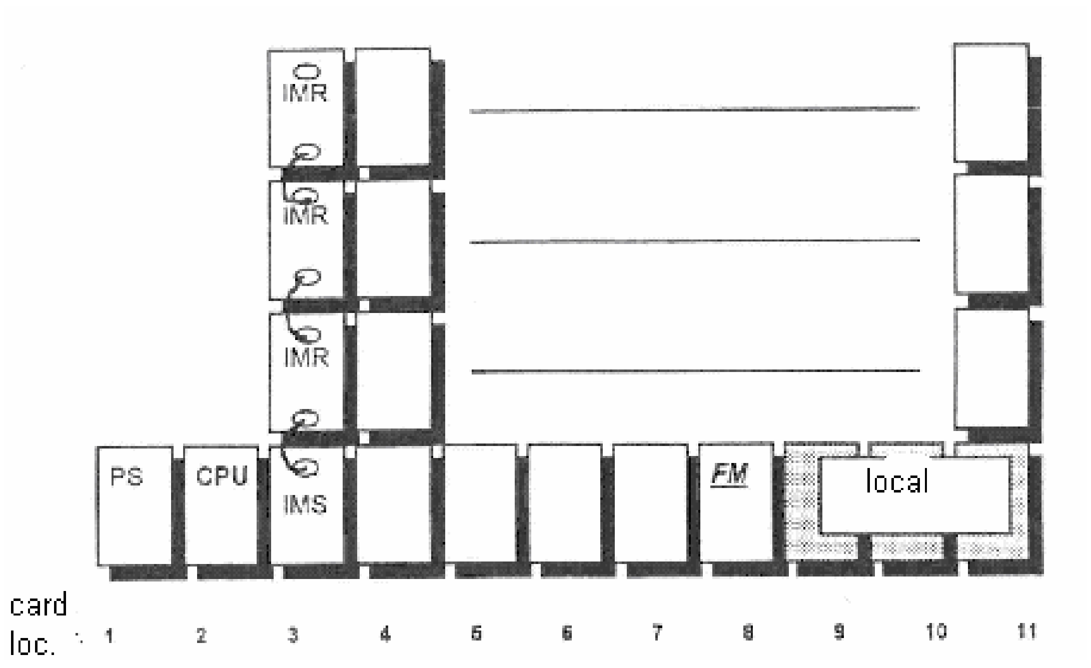
CPU 312 IFM	**CPU 313**	**CPU 314 IFM**	**CPU 314**	**CPU 315**	**CPU315-2DP**
6 Kbyte/2K statements RAM (integrated) 20Kbyte RAM	12Kbyte/4K statements RAM (integrated) 20Kb RAM	24Kbyte/8K statements RAM (integrated) 40Kbyte RAM	24Kbyte/8K statements RAM (integrated) 40Kbyte RAM	48Kbyte/16K statements RAM (integrated) 80Kbyte RAM	48Kbyte/16K statements RAM (integrated) 80Kbyte RAM
128 Byte DI/DO	128 Byte DI/DO	512 Byte DI/DO	512 Byte DI/DO	1024 Byte DI/DO	1024 Byte DI/DO
32 Byte AI/AO	32 Byte AI/AO	64 Byte AI/AO	64 Byte AI/AO	128 Byte AI/AO	128 Byte AI/AO
0,6 ms / 1K Instructions	0,6 ms / 1K Instructions	0,3 ms / 1 K Instructions	0,3 ms / 1K Instructions	0,3 ms / 1K Instructions	0,3 ms / 1K Instructions
1024 bit memories	2048 bit memories	2048 bit memories	2048 bit memories	2048 bit memories	2048 bit memories
32 Counters	64 Counters	64 Counters	64 Counters	64 Counters	64 Counters
64 Timers	128 Timers	128 Timers	128 Timers	128 Timers	128 Timers
10 DI/6DO on board, where 4 DI are for processor alarm or integrated functions faster counters integrated 20KByte EPROM		20 DI/16DO on board, where 4DI are for processor alarm or integrated functions faster counters 4 AI / 1AO on Board Solution: 11Bit + sign integrated 40Kbyte EPROM			Integrated PROFIBUS DP adapter (Master / Slave) Address assignment parameter lock

Compact- CPUs:

In the meantime there is also a number of compact CPUs formed by a more compact building method, whose favored price and integrated communication interfaces are highly suitable for training.
These CPUs are comparable with the standard CPUs in terms of functionality and the handling with the standard CPUs. The following table shows a selection of compact CPUs.

	CPU 312C	**CPU 313C**	**CPU 313C-2DP**	**CPU 314C-2DP**
Instruction runtime (Binary/benchmark/ glide time)	0.2 / 4 / 40µs	0.1 / 2 / 20µs	0.1 / 2 / 20µs	0.1 / 2 / 20µs.
RAM	16 KB	32 KB	32 KB	48 KB
SIMATIC Micro Memory Card	64 KB to 4 MB	64 KB to 4 MB	64 KB to 4 MB	64 KB to 4 MB
On board-Peripherals DI/DO AI/AO *PT100	 10 / 6 -/-	 24 / 16 4+1*/2	 16 / 16 -/-	 24 / 16 4+1* / 2
Technological functions - Counters/Frequency - Pulse output - Closed loop control - Positioning	 2 (10KHz) 2 (2,5 KHz) no no	 3 (30KHz) 4 (2,5 KHz) yes no	 3 (30KHz) 4 (2,5 KHz) yes no	 4 (60KHz) 4 (2,5 KHz) yes 1
Interface - MPI 187,5 kBaud - DP-Functionality	 Yes No	 Yes No	 Yes Master/Slave	 Yes Master/Slave

Expansion possibilities of an SIMATIC S7 - 300



The picture shows the maximum expansion possibilities of a SIMATIC S7-300/CPU314. The total assembly permits up to 32 modules (CPU 31ÌFM and CPU 313 only 8 modules), and for each rack, up to 8 modules.

The following rules apply for the allocation of the card locations:

Card location 1:	power supply PS **(=fixed assignment)**
Card location 2:	central processing unit module CPU **(=fixed assignment)**
Card location 3:	interface module IM **(=fixed assignment)**
Card location 4-11:	signal module SM, function module FM and communications processor CP **(=unfixed assignment)**

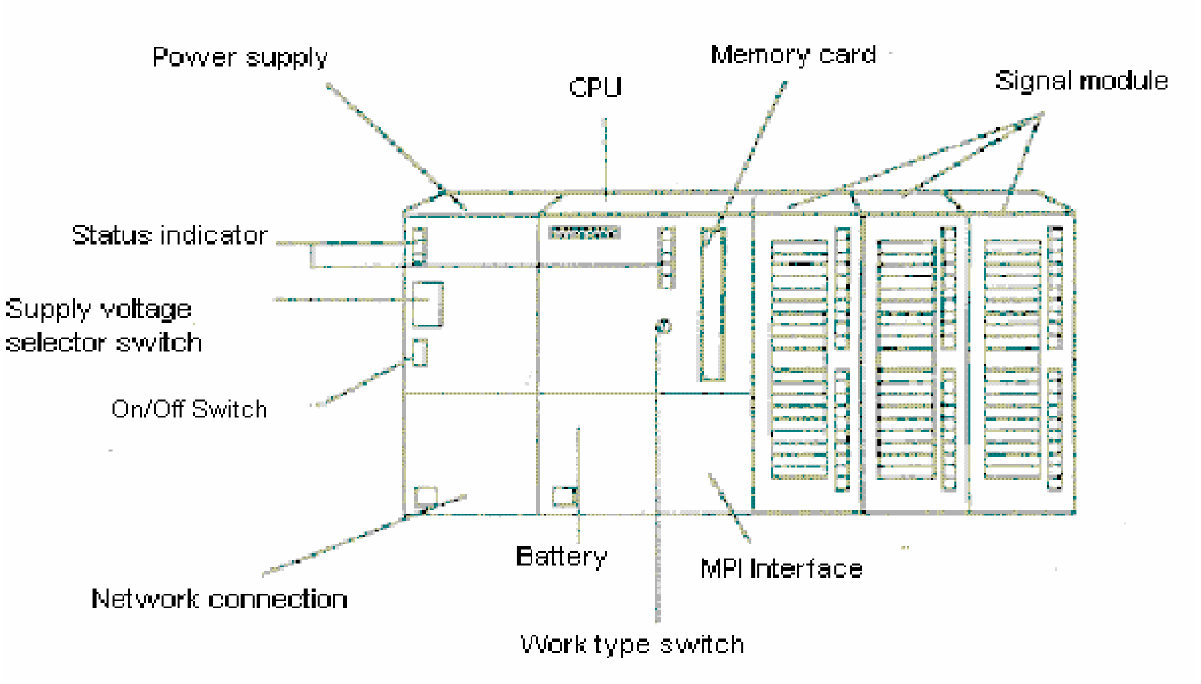
The bus laps further over the interface module(IM 361) between the racks. The connection IMS represents the transmitter and the IMR the receiver. The connection modules must be used at the indicated card locations. In the expansion rack still another power supply needs to be designed. Like the economical variant for the two-line structure, there are hard-wired logic modules such as IM 365 whereby no additional power supply is needed.

The following conduction lengths between the lines are available for:

- structures with 2 lines IM 365 max. 1m
- structures with several lines IM 361 max. 10m

Its own peripherals can be assigned to the functional module, e.g. SINUMERIK FM numerical control. Thus a FM module has its own peripheral range and can access it quickly. This peripheral range is called a local segment. One local segment can be developed per rack. During operation, an access to this peripheral of the CPU is no longer possible.

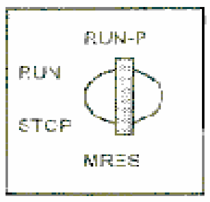
Important elements of the power supply and CPU:



Status- and error indication to the CPU:

Memory bit	Meaning	Clarification
SF (red)	System error	Diagnostic modules indicate a system error.
BAF (red)	Battery low	Announcement if the buffer battery does not contain enough voltage.
DC5V (green)	DC5V-supply for the CPU and rear wall bus	Announcement for the functional internal 5V supply of the CPU.
FRCE (yellow)	Force	Announcement for the condition of the CPU, in which inputs and outputs are force-actuated by a debug function.
RUN (green)	Operation state RUN	Flashes by the running of the CPU- static announcement when the CPU is in the RUN state.
STOP (yellow)	Operation state STOP	Flashes when a memory reset is requested – static announcement, when the CPU is in the STOP state.

Protection concept of the CPU:
Each CPU possesses a code switch for the switching of operation modes. The following operation modes 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

The protection concept for SIMATIC S7-300 makes it possible to protect certain parts of the automation system against unauthorized access. These are:
• The CPU and all programmable modules
• All objects (like e.g. blocks)

These protected parts can be affected by the PG or by B&B devices.

The CPU protection concept is divided into three protection stages. These protection stages determine, what is permitted to a user.
• Stage 1 – Code switch position Run-P or Stop: no protection, all functions are allowed.
• Stage 2 – Code switch position Run: write protection, reading functions is allowed, e.g. observation functions, information functions, compiling from the CPU.
• Stage 3 – Parameter lock over S7-Configuration (Password protection). Only observation and information functions are allowed, if the user does not know the password.
The memory resetting of the CPU can also be accomplished with the code switch as follows:

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- memory bit is shown.	The STOP-memory bit expires and after approx. 3 seconds, it will be shown again. With new CPUs, wait until the STOP-Memory bit lights up for the second time. **Important:** Between step 2 and step 3 should a maximum of 3 seconds 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.**

Forward **Function and design of a PLC** Programming language STEP 7

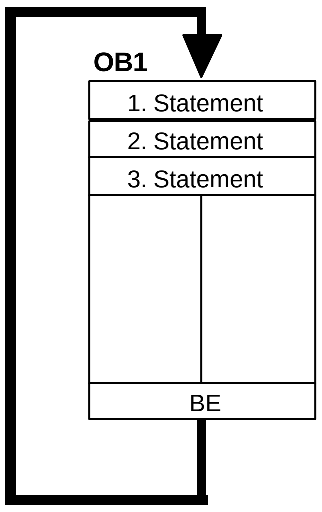
2.7 PROGRAM PROCESSING

2.7.1 PROGRAM MEMORY

During program processing, there are two different possibilities which are dependent on the programming and the used controller. The treatment of an individual instruction requires a certain time (microsecond range). This time for a unique treatment of all instructions is called a cycle time which is the unique runtime of a program (the program scan cycle).

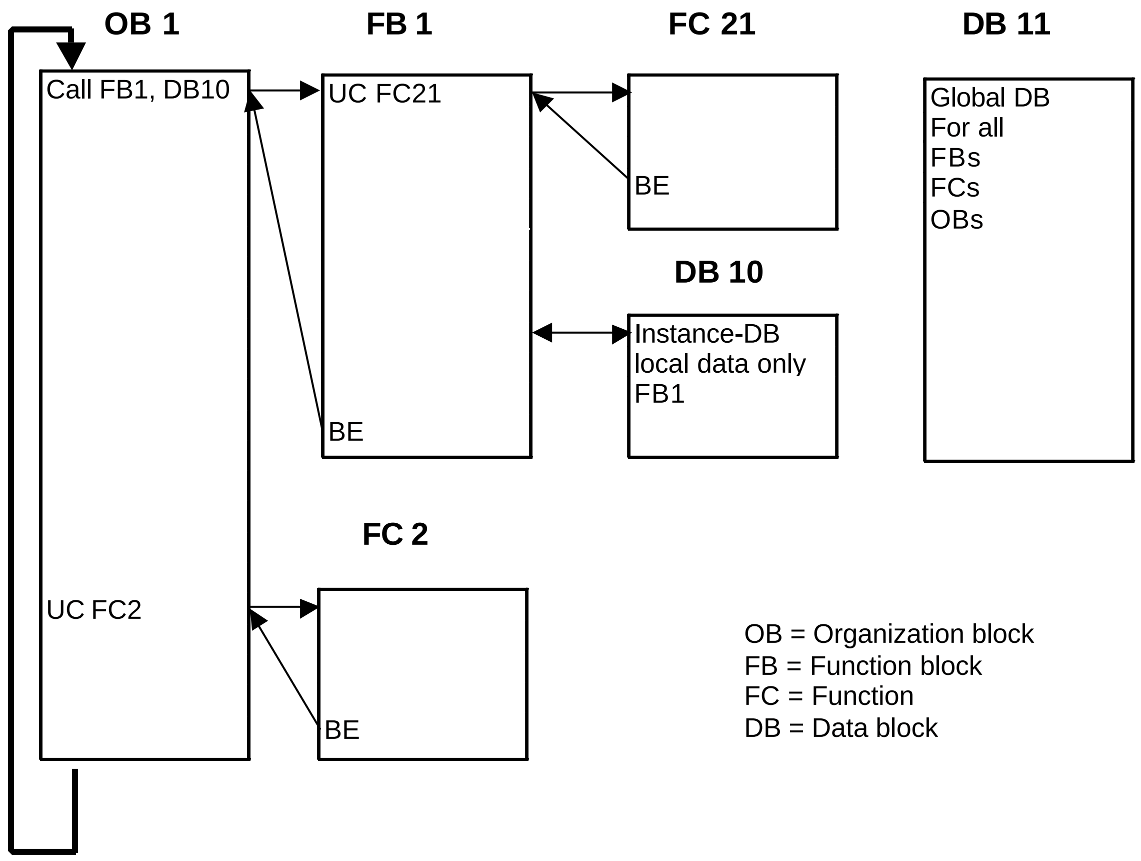
2.7.2 LINEAR PROGRAM PROCESSING

With linear programming the instructions are worked on by the controller in the order in which they are deposited (stored) in the program memory. If the end of the program (BE) is reached, the program processing begins again from the first step.
In terms of cyclic treatment.
The time which equipment needs for a unique treatment of all instructions, is called cycle time. Linear program processing is usually used for simple controllers and can be realized in only one OB.



2.7.3 STRUCTURED PROGRAM PROCESSING

One partitions a program into small, visible and program blocks arranged according to functions by using complex control tasks. This has the advantage of being able to individually debug the program sections of an overall function unit for functionality. The program blocks must be called over block call instructions (call xx/UC xx/CC xx). If a block end was recognized, the previous block is returned to and further processed.



OB = Organization block
FB = Function block
FC = Function
DB = Data block

2.7.4 USER BLOCKS

STEP 7 offers the following user blocks for structured programming:

- OB (Organization block):
  An OB is cyclically called by the operating system and forms the interface between user program and operating system. In this OB, the program that should be edited is transmitted to the control mechanism of the AG over the block call instructions.
- 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):
  An 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.
- DB (Data block):
  DBs are used around storage location for data variables. There are two kinds of data blocks: Global DBs, where all OBs, FBs and FCs can read or write data into the DBs, and instance DBs, which are assigned to a certain FB.

2.7.5 SYSTEM BLOCKS FOR STANDARD- AND SYSTEM FUNCTIONS

System blocks concern finished functions, which are deposited into the CPU.
These blocks can be called by the user and used in the program.

STEP 7 offers the following system blocks:

- SFB (System function block):
  Deposited in the operating system of the CPU and are of the user callable functional block.
- SFC (System function):
  Deposited in the operating system of the CPU and are of the user callable functional block.
- SDB (System data block):
  Storage areas in the program, which are provided by different STEP 7-Tools (e.g.: S7 Configuration, Communication Configuration ...) in order to store data about system settings for the automation system.

3. PROGRAMMING LANGUAGE STEP 7

3.1 THE PROGRAMMING LANGUAGE STEP 7 IN GENERAL

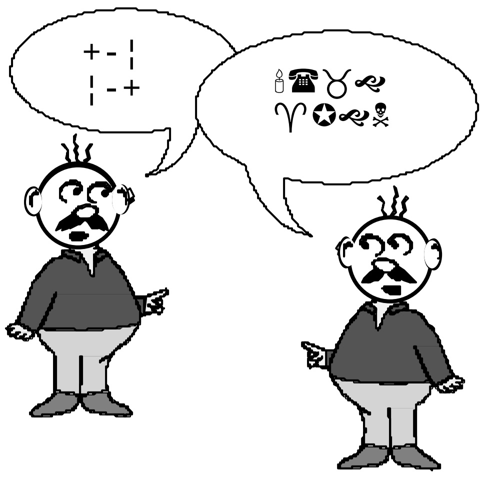
Due to the separation of **SIMATIC S5** through **SIMATIC S7** , a new programming software(**STEP7**) was developed and based on the Standard **IEC 61131**.
STEP 7 runs under **WINDOWS 95**, **98** or **NT** and possesses a graphical user interface.

3.2 CONVERTING STEP 5 Þ STEP 7

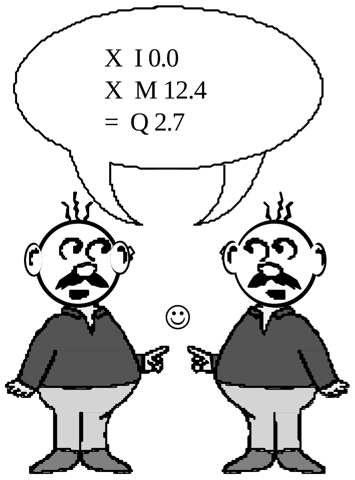
The tool ‘S5 file conversion‘ in STEP 7 allows a STEP 5 file to be converted into a STEP 7 file.

Further information about the STEP 5 conversion process can be found in the STEP 7 reference manuals or conversion guidance in Module B6 (Conversion STEP 5 -> STEP 7).

3.3 THE NORM IEC 61131 FOR A PLC



So far advancements led to a variety of manufacturer-specific languages and dialects in PLC technology. A common linguistic basis became more and more lost, and also with the communication of different PLCs among themselves, had many problems arisen.



With the Norm IEC 61131, the PLC technology was placed on a world-wide uniform basis for the first time. Under the presidency of the USA, the international commission for electro-technology (International Electronical Commission) discharged the PLC standardization in five partial topics:

Part 1:	general definitions and typical operation characteristics
Part 2:	electrical, mechanical, and functional requirements of the devices
Part 3:	5 program languages
Part 4:	user guidelines for all project phases
Part 5:	different manufacturer communication from PLCs

3.4 DIRECTORY STRUCTURE

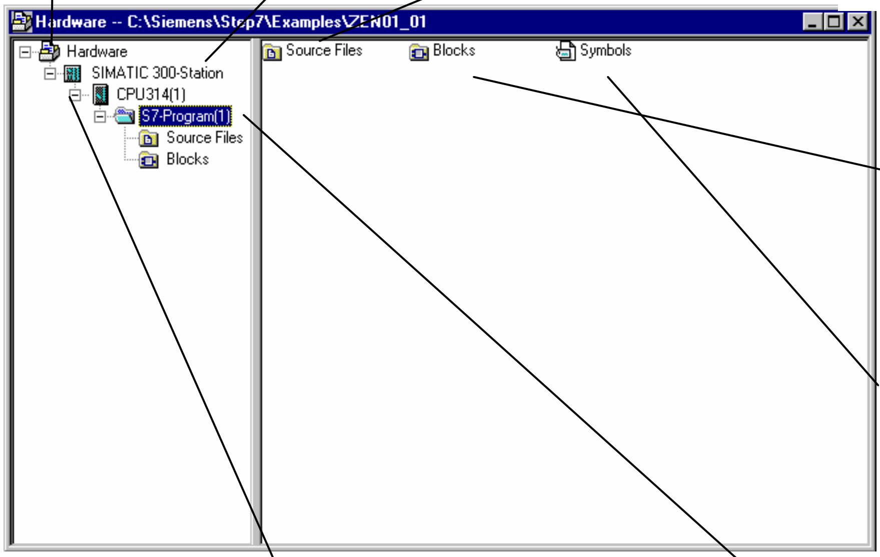
File management takes place in STEP 7 with the **SIMATIC Manager**. Here e.g. program blocks can be copied or be called for further processing with other tools by clicking with the mouse. The operation corresponds to the standards usually seen in WINDOWS 95/98/2000/ME/NT4.0. (in such a way e.g. the possibility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

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

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.

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.

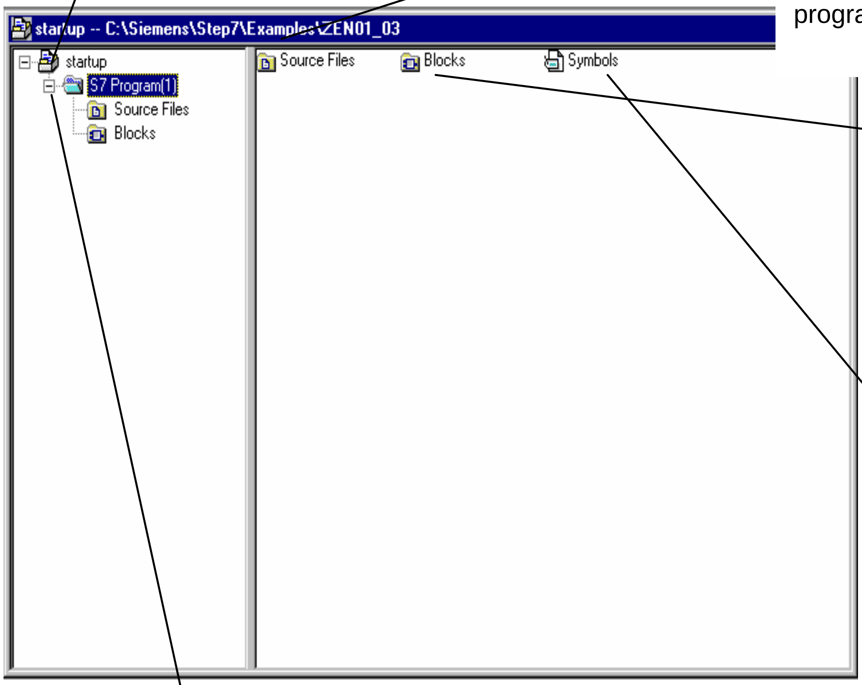
*^{1} Terms are from STEP 7 Version 2.x

In order to make a project independent from the hardware configuration, there is a possibility to create a project that does not contain all 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.)

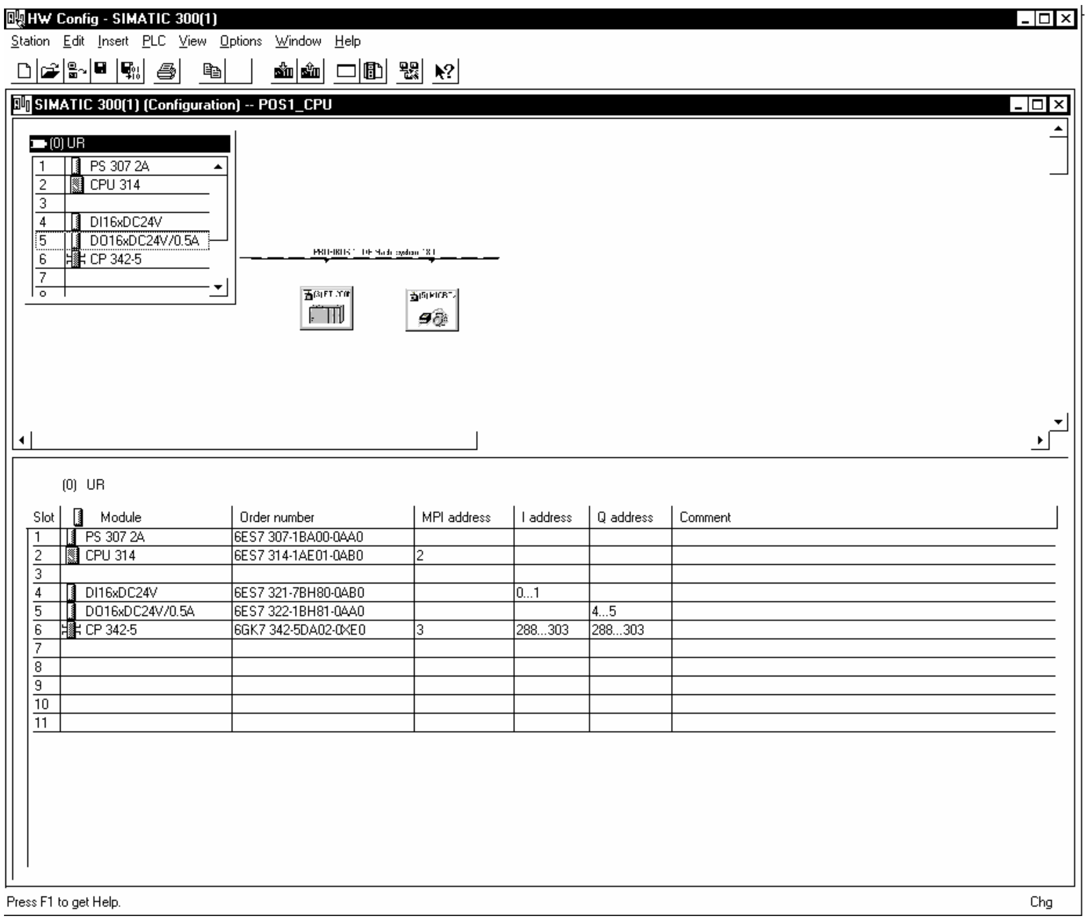
Symbols/SY*^{1}:
Stored here are the symbol lists for symbolic addressing.

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

3.5 CONFIGURATION AND PARAMETERIZATION

During the start of a program, the CPU produces a target configuration and places it into the system data block (SDB). With the Tool ‘Hardware Configure‘ a target configuration deviating from the SDB can be provided and thus the structure of the control be configured. In addition, an existing configuration can be loaded from a CPU. In addition, modules like the CPU can be given parameters (e.g. startup and cycle behavior of a CPU).


HW Config - SIMATIC 300(1)
Station Edit Insert PLC View Options Window Help
SIMATIC 300(1) (Configuration) -- POS1_CPU
(0) UR
1 PS 307 2A
2 CPU 314
3
4 DI16xDC24V
5 DO16xDC24V/0.5A
6 CP 342-5
7
Press F1 to get Help.
Chg

(0) UR

Slot	Module	Order number	MPI address	I address	Q address	Comment
1	PS 307 2A	6ES7 307-1BA00-0AA0				
2	CPU 314	6ES7 314-1AE01-0AB0	2			
3						
4	DI16xDC24V	6ES7 321-7BH80-0AB0		0...1		
5	DO16xDC24V/0.5A	6ES7 322-1BH81-0AA0			4...5	
6	CP 342-5	6GK7 342-5DA02-0XE0	3	288...303	288...303	
7						
8						
9						
10						
11						

3.6 CONTROL STATEMENT

For editing by a programmable logic controller, the control task is divided into individual control instructions. A control statement is the independent unit of a control program. It represents a work regulation for the control mechanism.

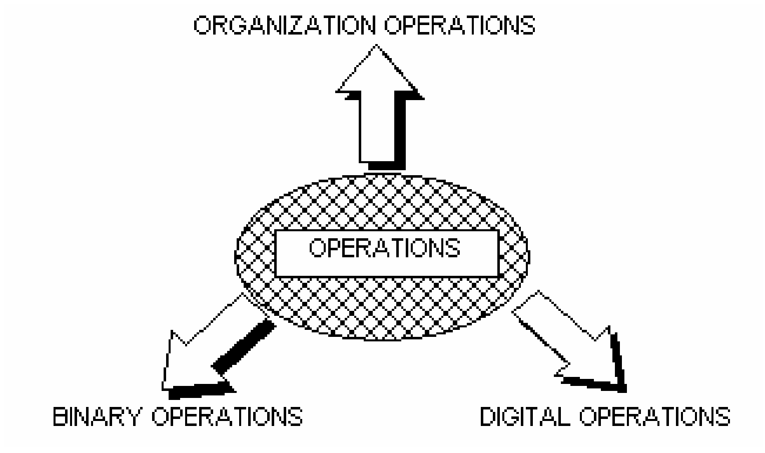
Designations, characteristics and symbols for control statements are defined in a DIN 19 239.

A control statement is build as follows:

Control statement		
Operation part	Operand part	
	Characteristic	Parameter
A	**I**	**0.0**

3.6.1 OPERATION PART

The operation describes the function which can be required. DIN 19 239 differentiates to:



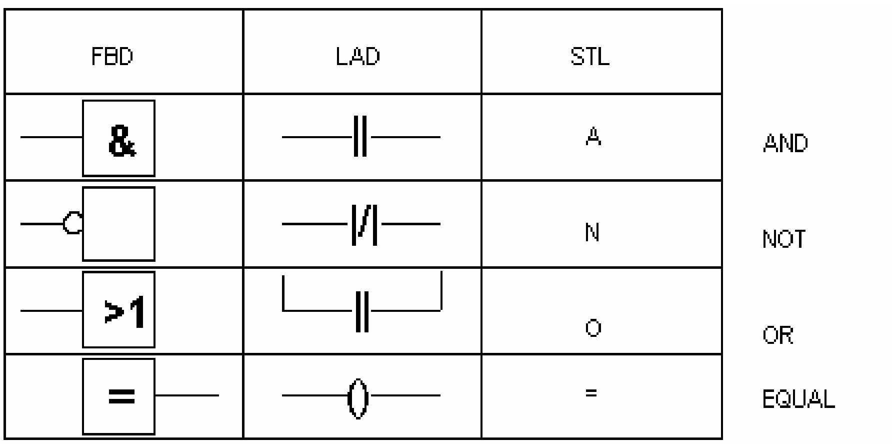
Example of digital operations:

L	. . . . .	Load
T	. . . . .	Transfer
>I	. . . . .	Larger than integer
==R	. . . . .	Equal to counter/timer

etc.

Example of binary operations:

Output of DIN 19 239



FBD	LAD	STL	
&	—| |—	A	AND
—○	—|/|—	N	NOT
>1	—| |—	O	OR
=	—( )—	=	EQUAL

etc.

Example of organization operations:

CC	. . . . .	Condition call
UC	. . . . .	Unconditional call
OPN	. . . . .	Open a data block
JU	. . . . .	Jump unconditional
JC	. . . . .	Jump if RLO=1
BEU	. . . . .	Block end unconditional
BEC	. . . . .	Block end conditional

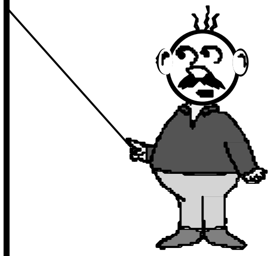
etc.

3.6.2 OPERAND PART

The operand part contains all statements necessary for the execution of the operation.
It indicates with which control mechanism an operation is to be implemented.
The operand characteristic contains the kind of the operand.

For example:

I for Input
Q for Output
M for Memory bit
L for Local data (internal block variables)
T for Timer
C for Counter
OB for Organization block
FB for Function block
FC for Function
DB for Data block
SFB for System functions block
SFC for System function
L#.. for 32-Bit-constant
etc.



The operand parameter indicates the address of the operand.

3.7 ADDRESSING

3.7.1 SYMBOLIC ADDRESSING

Symbolic addressing is often helpful for better understandability of addresses. It makes it possible to assign a symbolic name to a certain absolute address. You can assign e.g. the name END_STOP to the input I 0.0 and BOOL to the data type. Each symbolic name may occur only once. The assignment takes place with the Tool symbol editor, which you can start from the SIMATIC manager.

3.7.2 ABSOLUTE ADDRESSING

The following types of absolute addresses are found in STEP 7:

- immediate addressing
- direct addressing
- memory indirect addressing

Immediate addressing:

During immediate addressing, the operand is coded directly into the operation, i.e. The operation can work or imply the operand with the value that directly follows the address.

Example:

SET	Set the RLO (result of logic operation) to 1.
+D	Add the value from ACCU 1 with the value from ACCU 2 and place the result in ACCU 1.

Direct addressing:

During direct addressing, the operand address is coded into the operation, i.e. the operand indicates the address of the value, which the operation is to process. The operand consists of an operand characteristic and a parameter and points directly to the address of the value.

Example:

A I 0.0	Complete an AND - operation of the input bit I 0.0.
L IB 0	Load input bit IB 0 into ACCU 1.
= Q 4.0	Assign the RLO to the output bit Q 4.0.

Memory indirect addressing:

During memory indirect addressing, the operand address is given indirectly over an operand, which contains the address. I.e. the operand indicates the address of the value, which is to process the operation. The operand consists of an operand characteristic and one of the following pointers:

- a word, that contains the number of a timer (T), counter(C), data block(DB), function(FC) or function block(FB).
- a double-word, that contains the exact address of a value inside of the upper memory, which the operand label specifies.
  The address of the value or the number indicates the operand indirectly over the pointer. The word or double-word can be found in bit memories (M), a data block(DB), an instance data block (IDB) or local data(L).

Example:

A I [MD 3]	Complete the AND - Operation of the input bit. The exact address can be found in the double-word memory bit MD3.
L IB [DID 4]	Load the input bit into the ACCU 1. The exact address Cab be found in the double-word instance DID 4.
OPN DB [MW 2]	Open the data block. The data block number can be found in the memory bit MW 2.

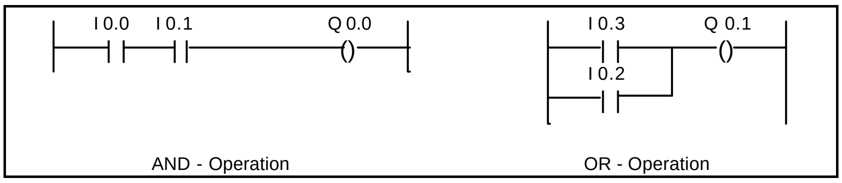
3.8 PROGRAM DESCRIPTION

In STEP 7 a program in the Tool LAD/STL/FBD block programming can be programmed and represented in three different possible languages:

- Ladder diagram LAD
- Function block diagram FBD
- Statement list STL

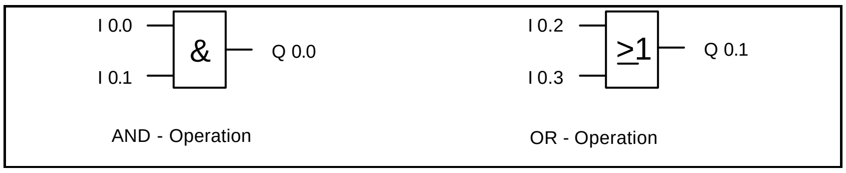
3.8.1 LADDER DIAGRAM LAD

The ladder diagram is the figurative representation of the control task with symbols according to DIN 19 239, which is used in the USA. It has much similarity with the conventional circuit diagram, however the individual current paths are not perpendicular, but rather horizontally arranged due to consideration for the representation of the screen.



3.8.2 FUNCTION BLOCK DIAGRAM FBD (IN STEP 7 VERSION 3.x AND HIGHER)

The operating diagram is the figurative representation of the control task with symbols according to DIN 40 700 and DIN 19 239. The individual functions are represented by a symbol with function identifiers. On the left side of the symbol the inputs are arranged, and on the right side the outputs.



3.8.3 STANDARD LIST STL

In the statement list the control task with individual control statements is described. The control statement(operation and operand) represents the task with mnemonic abbreviations of the function name (according to DIN 19 239).

Operation part:	**Operand part:**		
	Characteristic	**Parameter**	
A	I	0.0	AND- Operation
A	I	0.1	
=	Q	4.0	
O	I	0.2	OR- Operation
O	I	0.3	
=	Q	4.1	

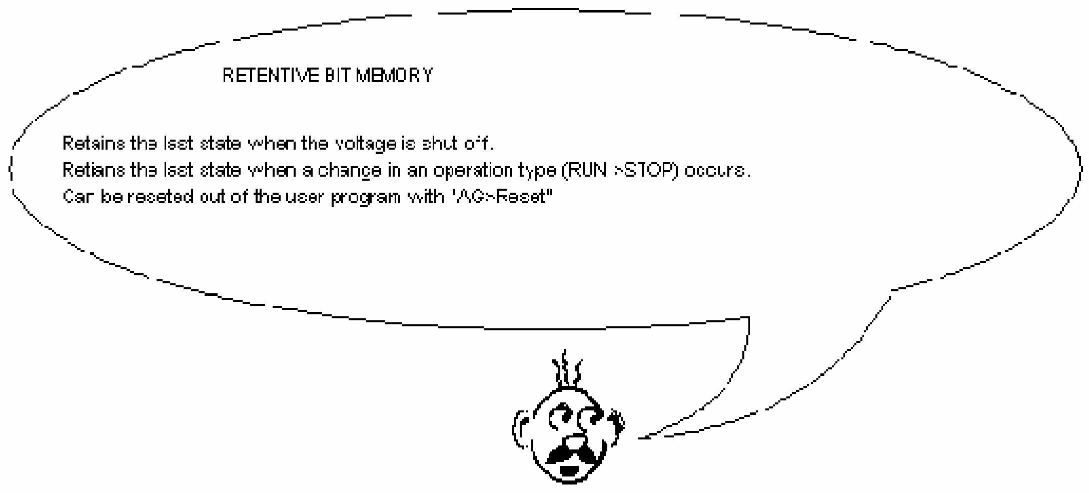
Each type of representation contains special characteristics and determined limits. If determined rules are kept with programming, then it is possible to translate into all three types of representation problem-free. Control programs in LAD or in FBD that were programmed, can be in principle always translated into a statement list STL.
In the program memory of the controllers, the program is always written in STL (however in machine language).

3.9 BIT MEMORY

For operations within the control with which no signal delivery is necessary outside of the control, bit memories are used. Bit memories are electronic memory elements (R-S memory elements) with which two signal statuses ("0" and "1") can be stored. Each PLC has a large number of bit memories available. They are programmed like outputs. In case of failure, the stored content of a bit memory is lost.

3.9.1 RETENTIVE BIT MEMORY

A part of this memory is however retentive (no voltage remains). By a buffer battery in the PLC, power failure is bridged. Therefore the logical conditions remain.


RETENTIVE BIT MEMORY
Retains the last state when the voltage is shut off.
Retians the last state when a change in an operation type (RUN >STOP) occurs.
Can be reseted out of the user program with 'AG>Reset"

By using retentive bit memories, the last system or machine state can be saved before leaving the operating condition. With a restart, the system or machine can continue working in the place from which it was stopped at. The retentivity ranges are specified by the parameters of the CPU in the Tool S7 Configuration.

3.9.2 NON RETENTIVE BIT MEMORY

A non retentive bit memory is reset by the operation type “RUN>STOP“ as well as by "NET 1" .