SCRIPT – PROGRAMMABLE LOGIC CONTROLS (PLC)

VOLUME I

Contents

1. Introduction

The computers have taken huge impact in the last 30 years also in automation technology. Programmable logic controllers replaced the conventional electrical relay controls in modern machines.

✐ **Compare the advantages and disadvantages of PLC compared to the relay control!**

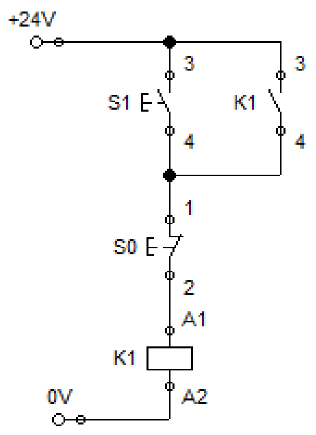


*Fig 1:
Hard-wired control
(relay control)*

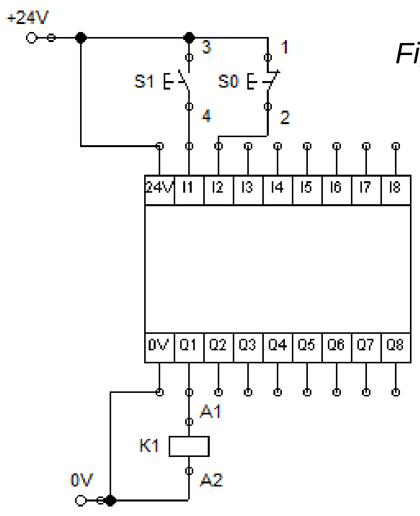


Fig 2: PLC

Advantages of a PLC:

- Less space ;
- less electrical power required;
- Reuse;
- Programmable;
- Greater reliability;
- Easy maintenance;
- More flexibility;
- Allows communication with other CLPs and microcomputers;
- Less wiring
- Reprogramming possible

✐ **Which PLC manufacturers do you know?**

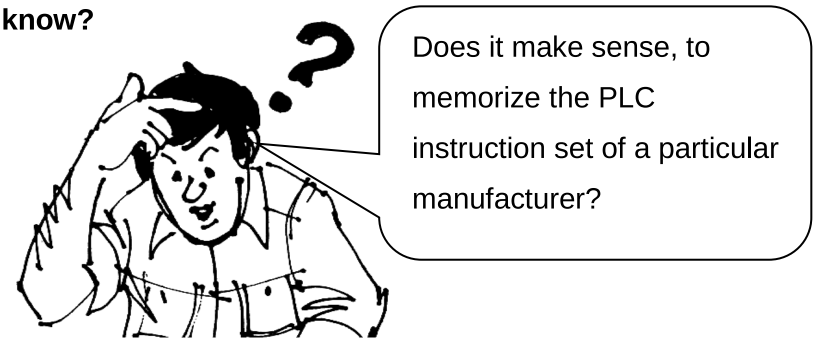
_ _

_ _

_ _

_ _

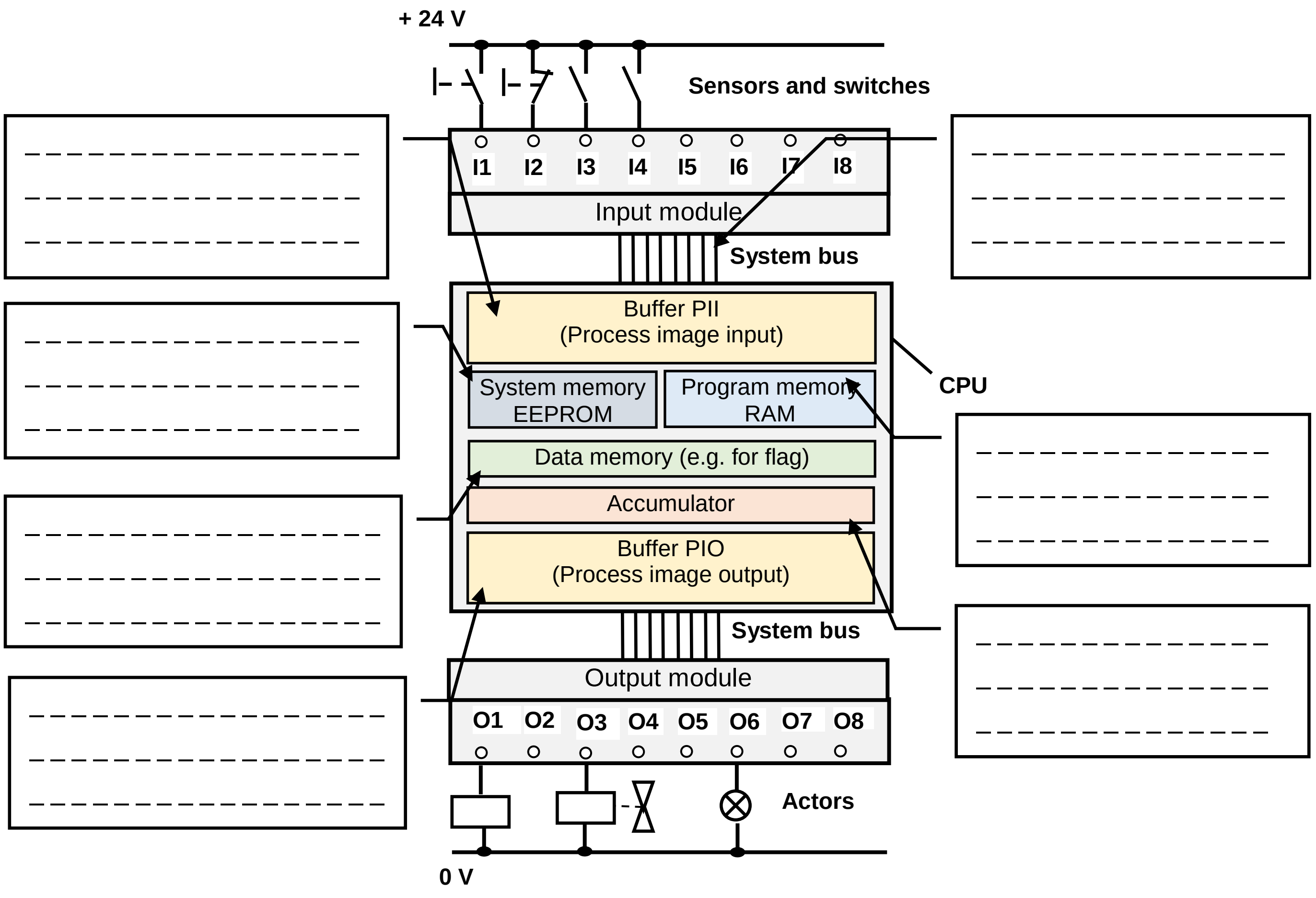
_ _


Does it make sense, to memorize the PLC instruction set of a particular manufacturer?

2. Structure of PLC

2.1 Basic structure

Basically, a PLC is structured like a computer consisting of a CPU and input and output module. They are task-neutral mass-produced.



Note:

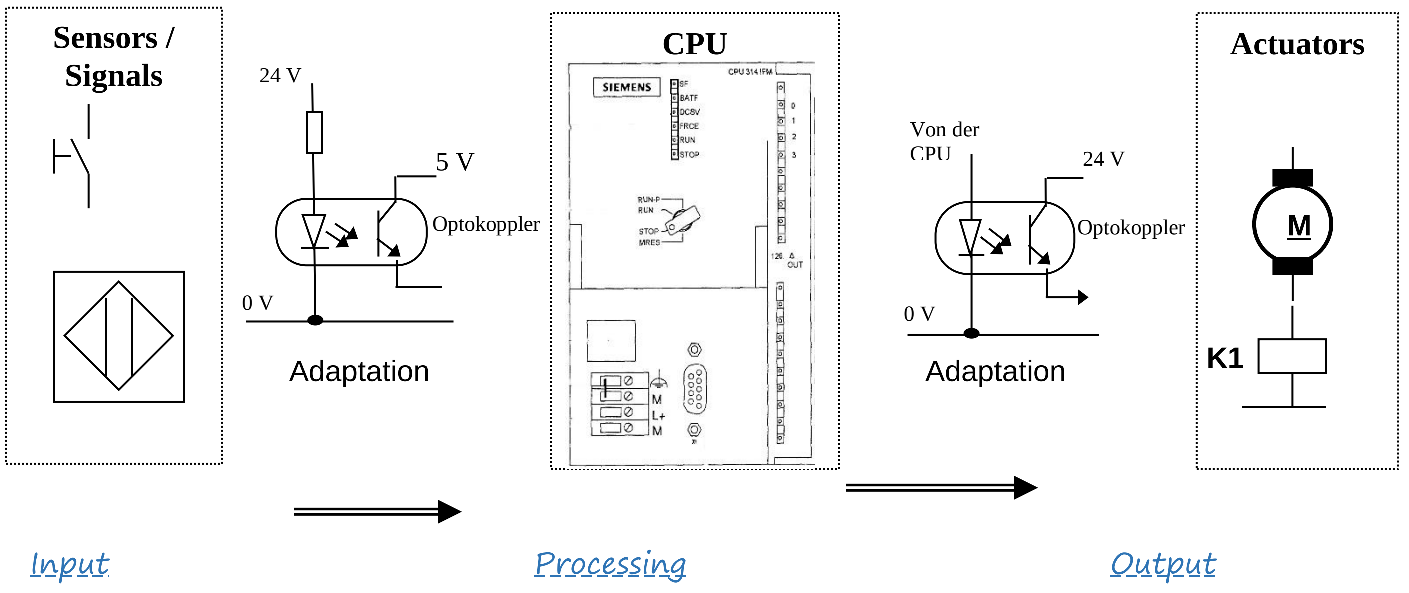
ROM: _

RAM: _

EPROM: _

EEPROM: _

2.2 Functions of the input and output modules

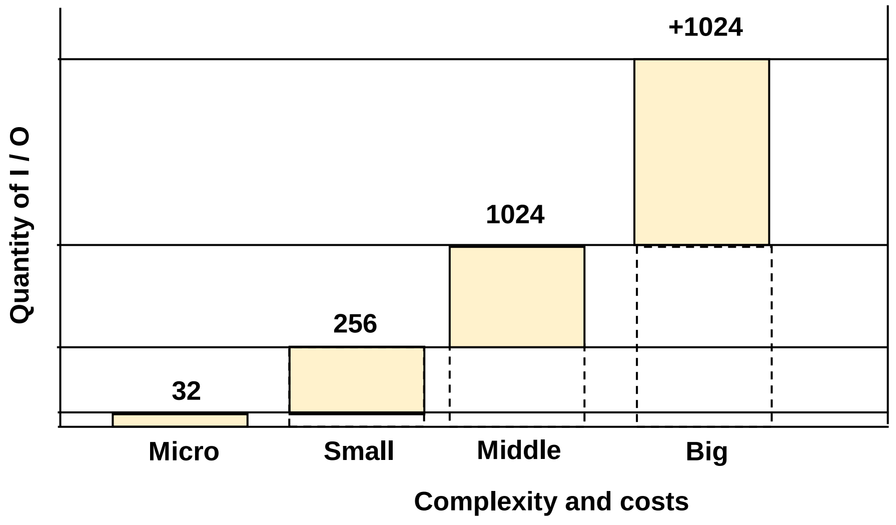


Notes:
The input and output module with its terminals form the interface between field devices and the CPU. The signals received by an input module may come from discrete sensors (such as limit switch, pushbutton, or switch digitizer is 'thumbwheel', etc.) or analog sensors (such as pressure transducers, temperature transducers, etc.).

The CPU consisting of a microprocessor and a memory system is the major component of the PLC. The CPU reads the input-signals, executes the logic according to the user program fault routines (if necessary) and sends signals to the output module.

2.3 Modular and compact structured control devices

Various criteria's have to be considered to choose the right PLC. You can distinguish PLC´s between micro, small, medium or large. The most important criteria's are: functionality, number of inputs and outputs, cost and physical dimensions.



In the context ‘control devices’, the terms '**compact**' and **'modular-structured**' are used:
Larger control devices are divided in individual modules. This **modular** system can (starting from a basic version) assemble PLC systems with different modules together. This allows to adapt the PLC system precisely according to the application (especially the number of inputs and outputs).

For much simpler control tasks, compactly constructed PLC's are offered. These **compact** PLC build a closed unit with a fixed number of inputs and outputs. Example: LOGO S7 von Siemens

3. Function of a PLC

3.1 Introduction

Within a CPU, two different programs are running:
- The operating system
- The user program

Operating system	The operating system is a part of each CPU and organizes all functions and processes of the CPU that are not associated with a specific control task. Its responsibilities include: - Unwinding of cold and warm restart - Updating of the process image of the inputs and the output - The call of the user program - The detection of alarms and alarm calls of Ob´s - How to handle errors - Managing memory - The communication of programming devices and other communication partners
Application program	The user program, you must create your own and load it into the CPU. Your application program is to fulfill the demands of the specific automation task.

3.2 Functioning of the PLC

In which way are the instructions of a user program processed by the CPU? To clarify the operation of a PLC, please write the following program. Watch as the output - which is addressed in the program - behaves when both inputs are simultaneously activated.

```
// Program 1

Network 1:
A     I 124.0
S     Q 124.0  // output is set

Network 2:
A     I 124.1
R     Q 124.0  // output is reset
```

```
// Program 2

Network 1:
A     I 124.0
R     Q 124.0  // output is reset

Network 2:
A     I 124.1
S     Q 124.0  // output is set
```

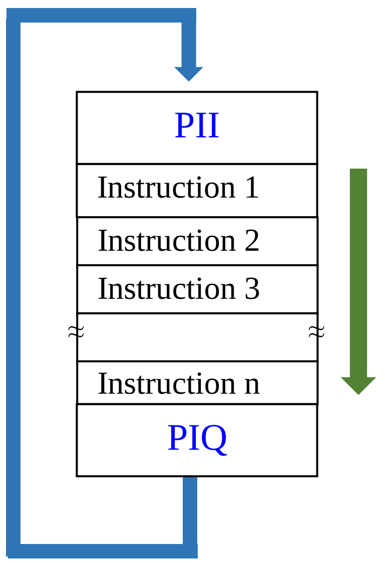
What you see:

The output of the PLC is switched off!

The output of the PLC is switched on!

Describing the function of a PLC:

PII	**p**rocess **i**mage of the **i**nputs
PIQ	**p**rocess **i**mage of the **o**utputs

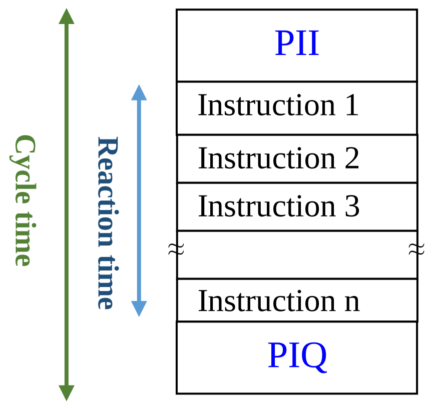


The PLC process the instructions serially
One instruction after another will be executed beginning with instruction 1.

The PLC works periodically
After executing all the instructions and setting the PIQ the whole process begins one more time.

In this context, two terms describe the speed of the PLC:

- Cycle time
  - The cycle time is the time required by the PLC for a single execution of the program. The cycle time is composed of
    - The processing time of an instruction
    - The number of instructions
- reaction time
  - The reaction time of a PLC is generally many times greater than the cycle time. It is composed of
  - Delay time of the input and output modules
  - Read-in and read-out time of the process images
  - Cycle time



A remark to the cycle time: The cycle time is an essential feature of a PLC and usually refers to 1 k = 1024 program instructions. Example: In modern PLC, the average processing time of an instruction is about 200 ns.

4. Introduction in the programming

4.1. PLC-programming languages

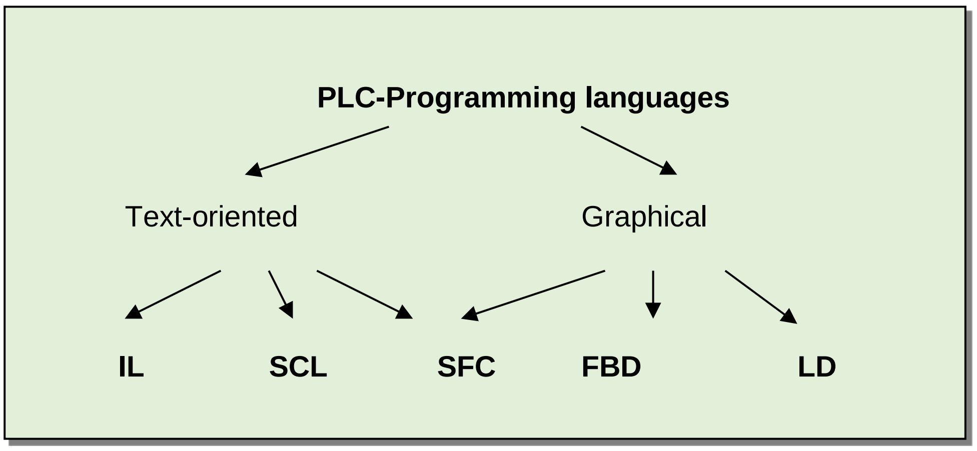
The classical PLC programming languages are:

- Instruction list (IL)
- Ladder diagram (LD)
- Function Block Diagram (FBD)

Added in the last few years are:

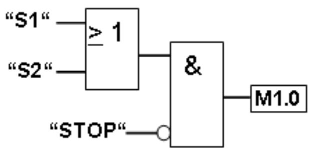
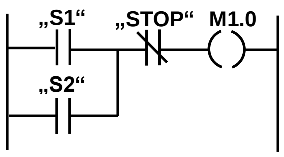
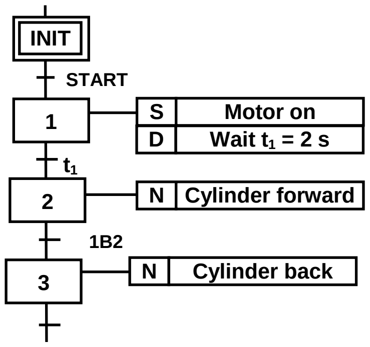
- Sequential Function Chart (SFC)
- Structure control language (SCL)

Fig. 1: Overview PLC-programming languages



The characteristics of these programming languages can be seen at the subsequent page!

Overview Programming languages

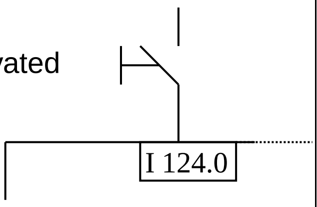
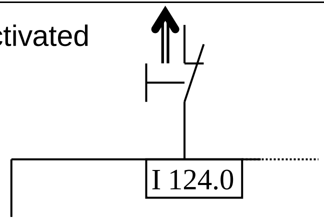
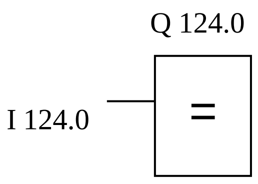
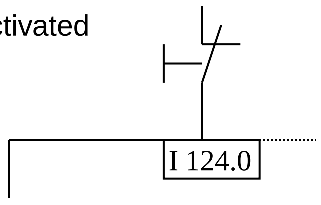
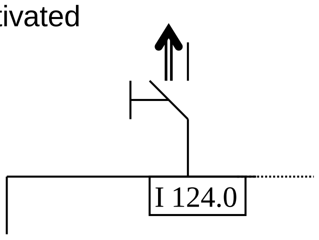
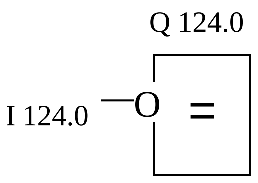
PLC-programming language	**Example**	**characteristics**
Function block diagram (FBD)		Usually does not contain the entire instruction set of the PLC Clear, easy troubleshooting Developed from the digital technology
Ladder diagram (LD)		Usually does not contain the entire instruction set of the PLC Clear, easy troubleshooting Horizontal paths are similar to the circuit diagrams
Instruction list (IL)	**International** / **Step 7** **LD %I X 0** / **O I 0.0** **OR %I X 0** / **O I 0.1** **AND %I X 0** / **AN I 0.3** **= M1.0** / **= M1.0**	includes the entire command set Single instructions in a mnemonic code troubleshooting difficult frequently used
Sequential Function Chart (SCL)	 INIT START 1 S Motor on D Wait t_1 = 2 s t_1 2 N Cylinder forward 1B2 3 N Cylinder back	Command set is very limited Requires a lot of memory Clear, easy troubleshooting Application to sequence controls
Structure controlled language	**FUNCTION** FC2: REAL **VAR_INPUT** X1: REAL X2: REAL **VAR_OUTPUT** X3: REAL **END_VAR** **BEGIN** X3 := X1 + X2; **END** FUNCTION	Clear program structure Contains the entire instruction set Developed on the basis of a high level language (such as Pascal or C)

4.2. Principles of PLC-programming

4.2.1. Programming contacts NO and NC

When creating a program, it should be noted whether the contact (button, sensor ...) is designed as normally open or normally closed. A signal state ‘1' at the input of the PLC can come from a contact NC or from a contact NO which is activated. The programmable controller cannot determine the periphery connected to its input module.

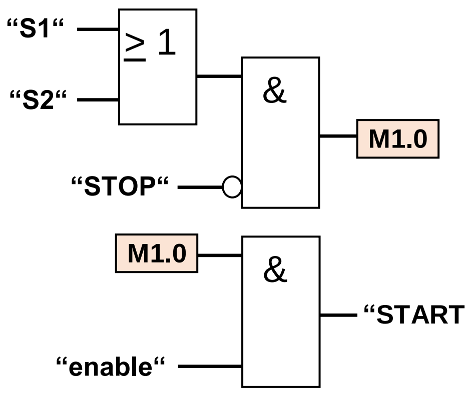
Example: Your task is to query the signal state at the PLC input I124.0 so that the output Q 124.0 is active when the button is pressed. Programming in FBD!

	Compare activated – not activated		presented in FUP
Contact is NO	not activated  Signal at PLC-input: 0	activated  Signal at PLC-input: 1	 Reading the signal change from 0 to 1
Contact is NC	not activated  Signal at PLC-input: 1	activated  Signal at PLC-input: 0	 Reading the signal change from 1 to 0.

4.2.2. Using flags

Do you want to get interim results (logical combination or calculation) of your program and store these? Then, a known way of doing this is to use flags. How flags can be used in Step 7 you can see below:

Fig 1: Extract of a S7-program



```
O    "S1"
O    "S2"
UN   "STOP"
=    M1.0
U    M1.0
U    "enable"
=    "START"
```

A flag is a bit in the memory. Regarding PLC, a distinction between retentive and non-retentive flags can be made:
Non-retentive flags loose their logic state upon failure of the supply voltage.
Retentive flags, however retain their value by buffering the RAM. The use of retentive flags makes sense if important process data such as levels, quantities, positions, etc. have to be retained in case of a breakdown in the operating voltage.

Which memory area is retentive or non-retentive, depends on the CPU. This can refer to the appropriate CPU manual.

4.2.3. Programming modular

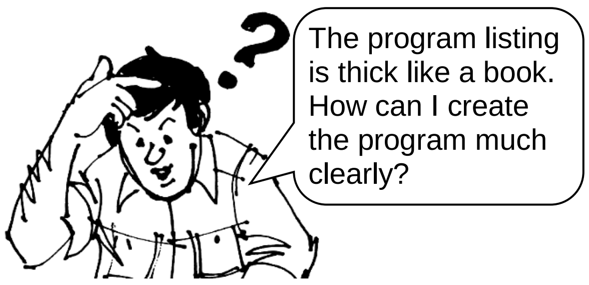

The program listing is thick like a book. How can I create the program much clearly?

Fig.: paperfolding machine from MBO
Source: MBO.de

Divide your program useful in individual modules on. It is useful, for example, if each program module has its own function - e.g. module 1 the logic, module 2 the analog processing, Module 3 the command output, etc.

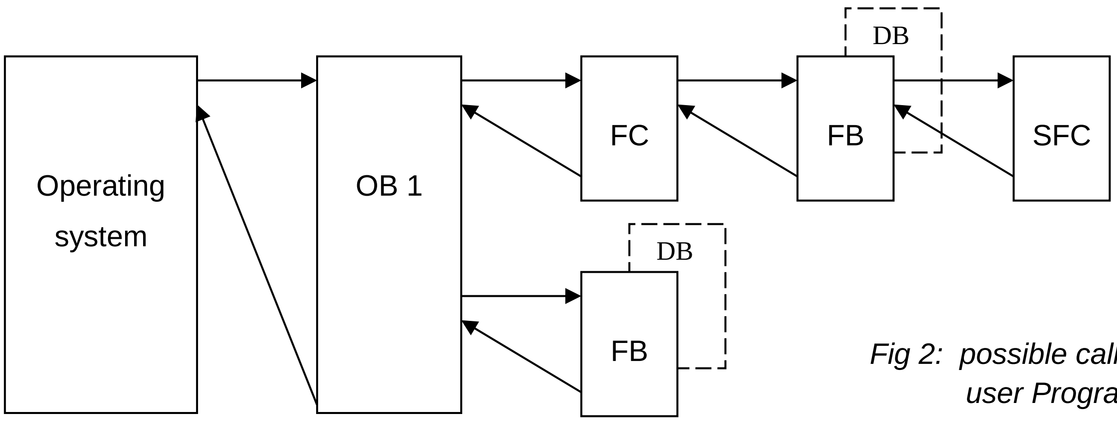


Fig 2: possible call hierarchy in an user Program

The order and nesting of the block calls is known as call hierarchy. The permissible nesting depth depends on the CPU.

Step 7 distinguishes the following program modules:

OB Organization Block	Ob's form the interface between the operating system and the application program. They control the boot up of the PLC, the cyclic call of the user program, the interrupt call, handle errors, etc.
FC Function	FC's you program yourself. These are program modules without memory with the possibility of passing parameters.
FB Function Blocks	FB's are also part of the user program but with advantage to have a "memory".
DB Data blocks	DBs are data areas for storing user data. Step 7 distinguish: - Instance-Data blocks, are assigned to a function block - Global Date blocks containing shared data which can also be defined and used by any blocks
SFB, SFC, internal FB	Predefined modules, provided by STEP 7, which you can parameterize as user.

<u>Work order:</u>

The motor of a machine can be switched on via the latching button S1 and switched off by button S2. Demand: For safety reasons, turn off should be dominant. Which storage element (RS or SR) you have to choose?

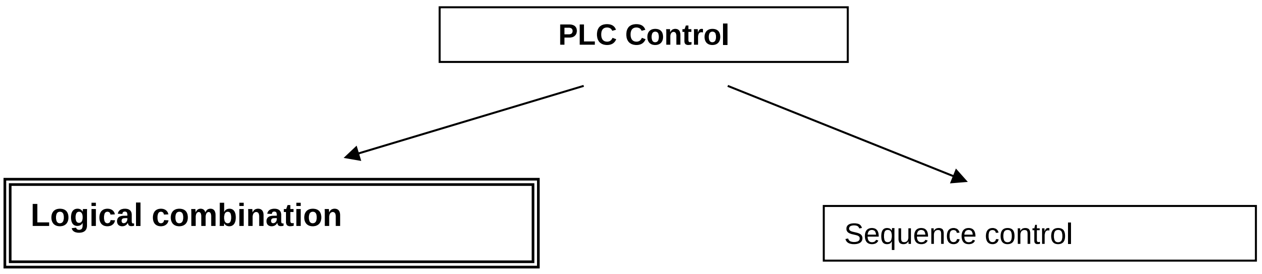
The machine has to be upgraded. The engine should be turned on in a latching mode by pressing the push-button S2 or S3. The motor is switched off by pressing the S1 button or if the overcurrent fuse F1 releases (both NC). Furthermore, the reset signal should be dominant. Program everything in OB 1. The outputs should be activated directly without using of markers.

✐ Now you want to program modular. The logic should be programmed in FC 1, the activation of the outputs in FC 2. Test your program!

5. Combinatorial logical control

5.1. Classification combinatorial control

Before you go on to program simple combinatorial controls, you should classify this kind of control within the following overview:



Here the input signals are combined by logical operations (AND, OR, NOT, timers, memory) to get one or more output signals. The pure combinatorial controls are usually simple structured.

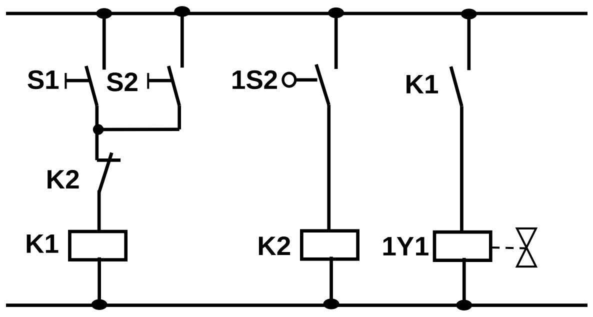
Control with a fixed workflow, which can be divided into individual steps.

5.2.Programming examples

Work order 1: Convert a relay control into a control with PLC

Relay controls are becoming increasingly replaced by popular PLC's for well-known reasons. This should also be your job here. Write the corresponding PLC program to replace the pictured electric control.

Electrical control circuit



symbol table

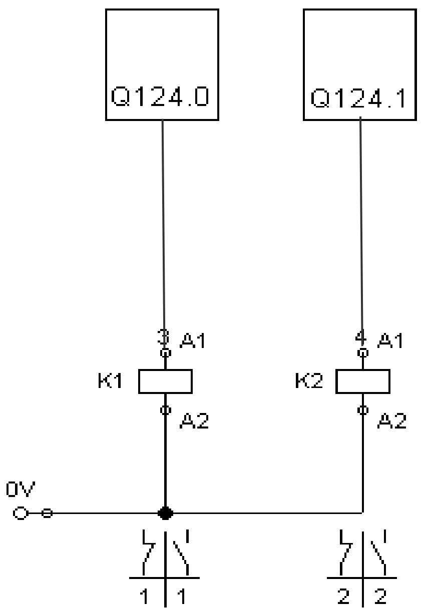
component	E/A	comment
S1	I124.0	Taster “ON” - NO
S2	I124.1	Taster “ON” - NO
S3	I124.2	Taste “OFF” - NC
1Y1	Q124.0	Valve

Work order 2: Change of rotation - indirect

With the following power unit - a so-called H-bridge – you can reverse the rotation of a DC motor.

Functional description:

The motor can started with the S1 button (K1 activated) in counter clockwise rotation (CCR) and with the button S2 (K2) in clockwise rotation (CR). Condition: The STOP button S0 is not actuated.
The switching of the direction of rotation can be made only after de STOP button S0 has been pressed so that it can be ensured that the motor has stopped.



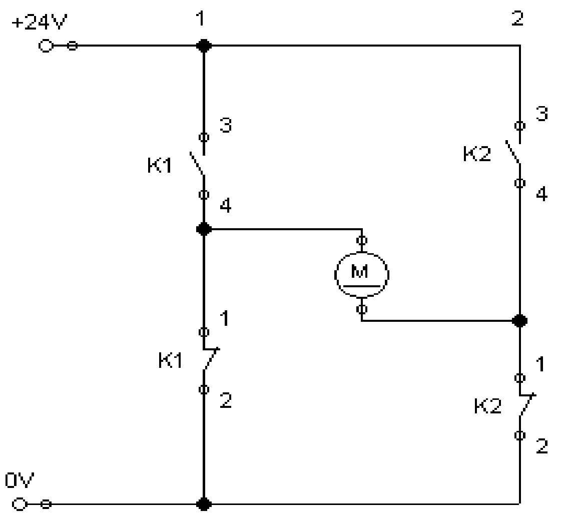


Fig.: Power unit (H-Bridge) for controlling the DC-motor

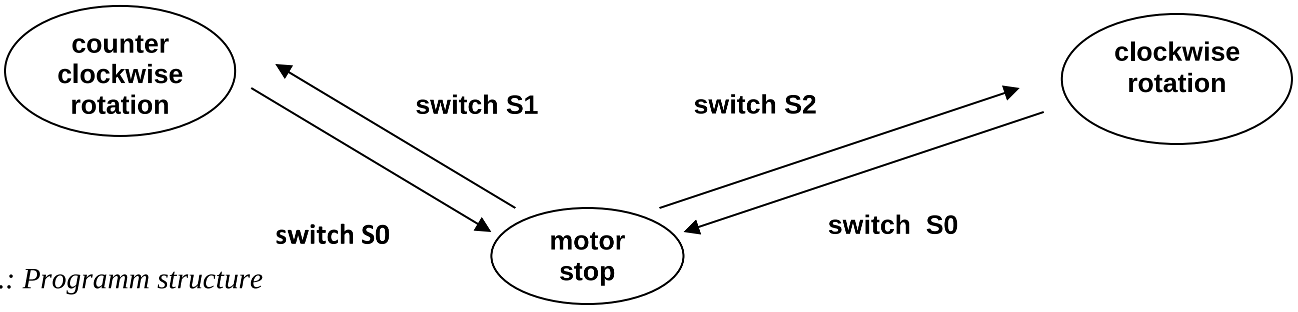
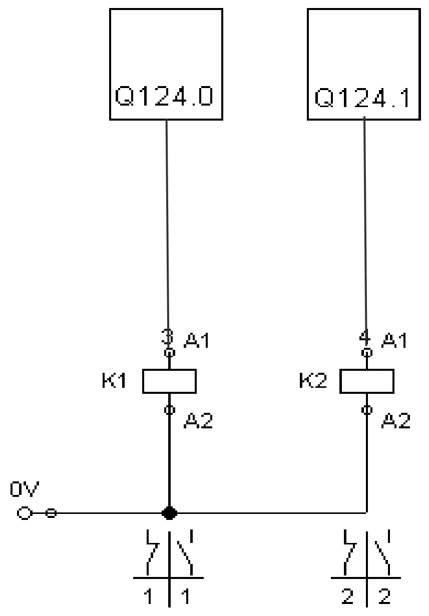


Fig.: Programm structure

Work order 3: change of rotation - direct

Functional description:
The motor can started with the S1 button (K1 activated) in counter clockwise rotation (CCR) and with the button S2 (K2) in clockwise rotation (CR). Condition: The STOP button S0 is not actuated. The switching of the direction of rotation can be done directly from the clockwise rotation into the counter-clockwise rotation and vice versa. Of course this is possible only for small engines.

Power unit - H-bridge



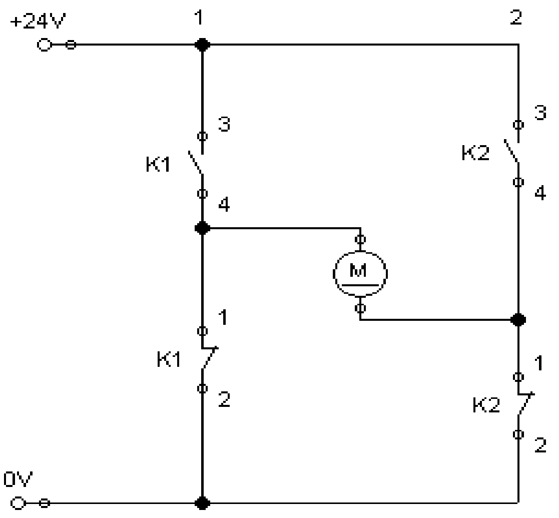
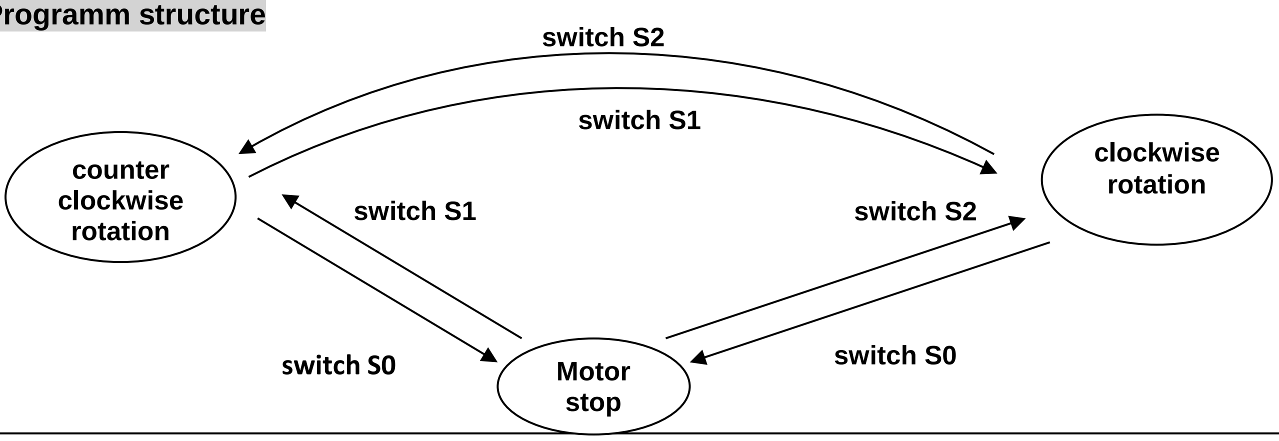


Fig.: Power unit (H-Bridge) for controlling the DC-motor

Programm structure



5.3. Combinatorial control according to disjunctive normal form

Often you have multiple switching-on option for an output. Each of these switching option has its own condition. Here, a solution approach is shown to get a Boolean equation, which is also known as disjunctive ('or') normal form.

Work order: Motor drive with two switches 'ON'

The motor M1 can be switched on by the locking switches S1 and S2. A motor is turned on when either switch S1 or switch S2 is turned on. If S1 and S2 are turned on the motor stops. Which steps are necessary to get the PLC-program?

Step 1: Get a Boolean table

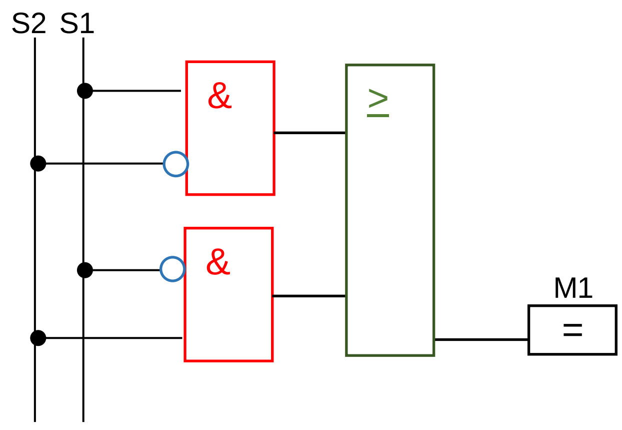
S2	S1	M1
0	0	0
0	1	1
1	0	1
1	1	0

Step 2: Develop function equation from the truth table

Of interest are these cases in which the output is activated (logic `1`)!

$\overline{\text{M}}\text{1} = (\text{S1} \wedge \overline{\text{S2}}) \vee (\overline{\text{S1}} \wedge \text{S2})$

Step 3: Develop logic circuit from the functional equation:



Example: **2 out of 3 circuit**

An injection molding machine may only start or continue the production process when the required operating temperature is reached and maintained. As a production stop of this installation during operation is involved with high costs, the temperature is measured by three sensors. The shut-down should only take place if at least two of the three temperature sensors report a temperature error (signal '1'). In this manner, unnecessary shut down can be prevented, which would cause high costs.

✐ Develop for this system the necessary logic circuit!

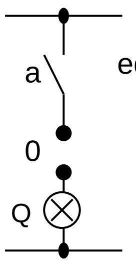
5.4.Simplifying logical circuits

Do you take many circuits too complex? If do so, then you can simplify them.

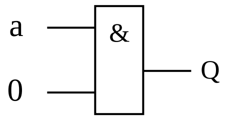
5.4.1. Simplifying by using Boolean algebra

The following rules apply for variables which are used in logical functions. The analogy to the electrical circuit should help you to work out these rules for yourself:

Combined with a constant:

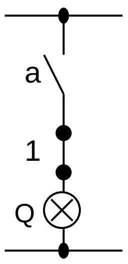


equal to

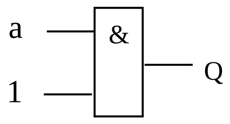


You conclude:

$a \land 0 = 0 \quad (1)$

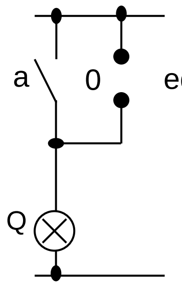


equal to

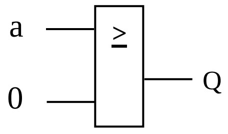


You conclude:

$a \land 1 = a \quad (2)$

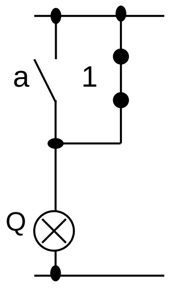


equal to

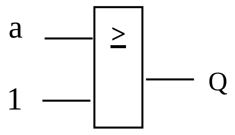


You conclude:

$a \lor 0 = a \quad (3)$



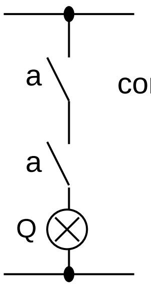
equal to



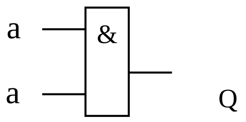
You conclude:

$a \lor 1 = 1 \quad (4)$

Variable combined with itself:

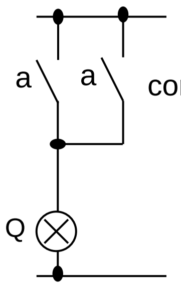


corresponde a

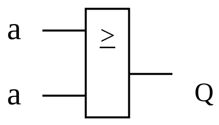


You conclude:

$a \land a = a \quad (5)$

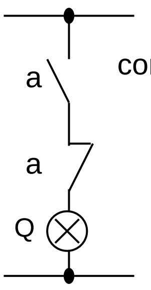


corresponde a

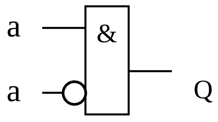


You conclude:

$a \lor a = a \quad (6)$

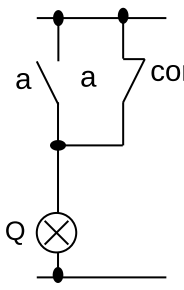


corresponde a

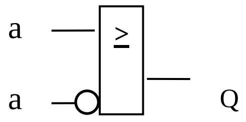


You conclude:

$a \land \overline{a} = 0 \quad (7)$



corresponde a

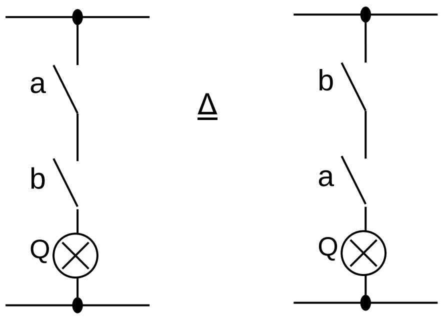


You conclude:

$a \lor \overline{a} = 1 \quad (8)$

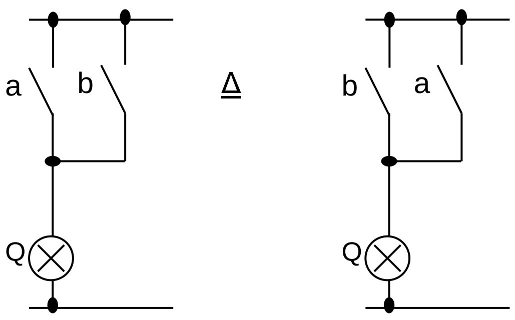
Law of commutation:

The Law of Commutation says that variables which are linked by 'AND' 'or 'OR' may be interchanged with each other.



You conclude:

$a \wedge b = b \wedge a \quad (9)$

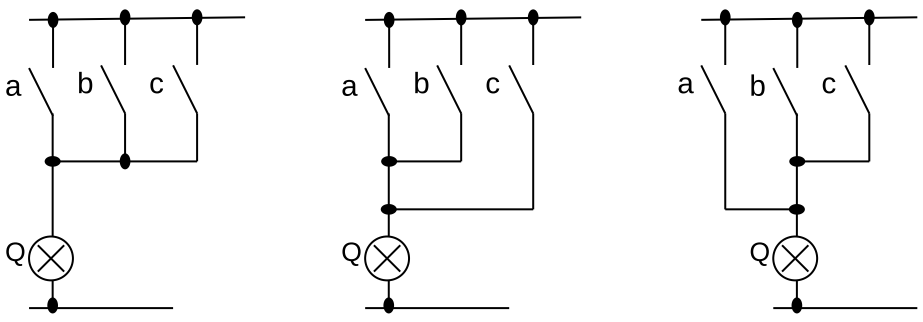


You conclude:

$a \vee b = b \vee a \quad (10)$

Associative rule:

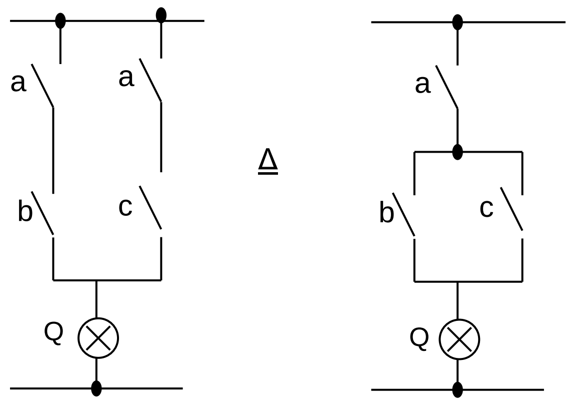
The associative rule tells you how you can summarize single variables, which are linked to each other by, AND 'or' OR '. Summarized variables are indicated by parentheses.



You conclude:

$a \vee b \vee c = (a \vee b) \vee c = a \vee (b \vee c) \quad (11)$

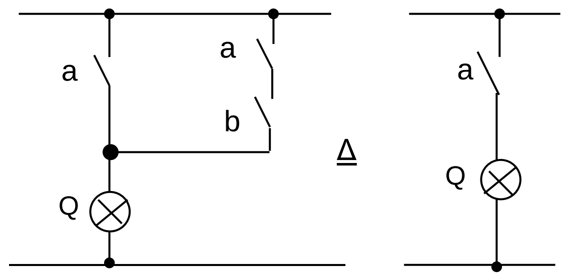
Law of distribution for calculating with parentheses



You conclude:

$(a \wedge b) \vee (a \wedge c) = a \wedge (b \vee c)$ (12)

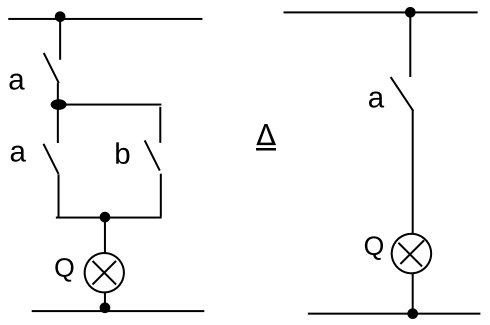
Law of absorption



First case:

You conclude:

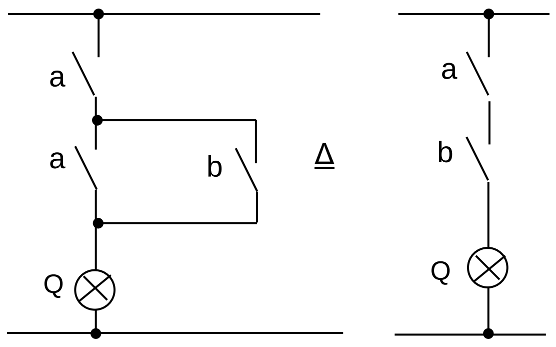
$a \vee (a \wedge b) = a$ (13)



Second case:

You conclude:

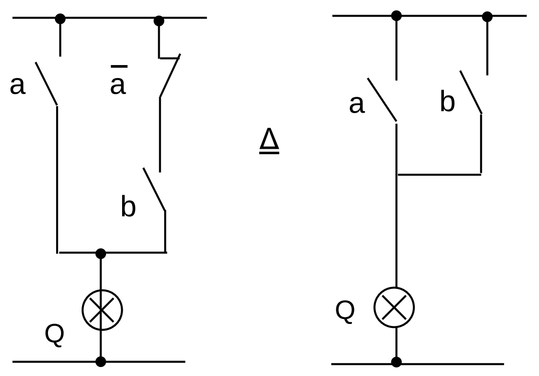
$a \wedge (a \vee b) = a$ (14)

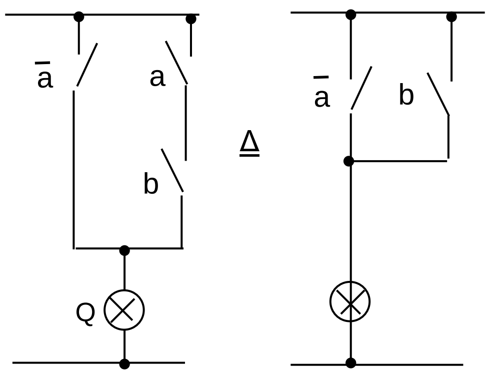


Third case:

You conclude:

$a (a \vee b) = a \wedge b$ (15)





Fourth case:

You conclude:

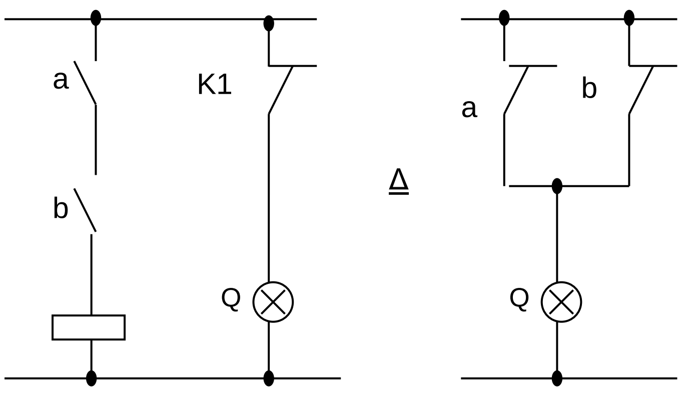
$a \vee (a \wedge b) = a \vee b$ (16)

You conclude:

$a \vee (a \wedge b) = a \vee b$ (17)

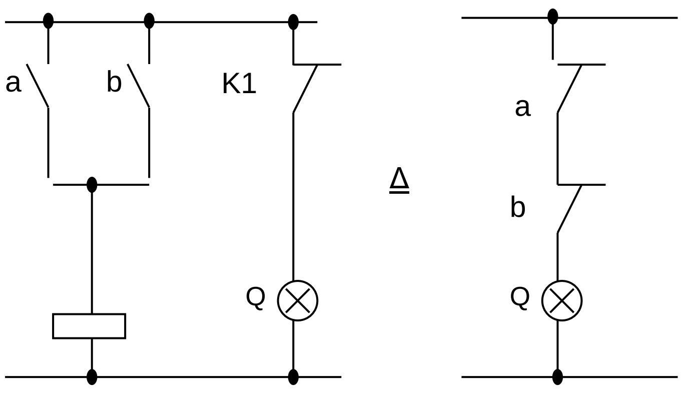
The rule of “de Morgan”

The De Morgan's rules are used to transforming of combinational circuits.



You conclude:

$\overline{a \wedge b} = \overline{a} \vee \overline{b}$ (18)



You conclude:

$\overline{a \vee b} = \overline{a} \wedge \overline{b}$ (19)

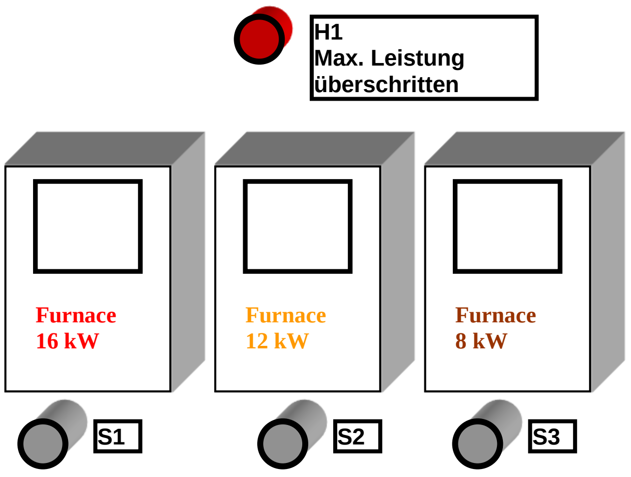
5.4.2. Simplifying via KV-diagrams

KV diagrams are named after their discoverer Karnaugh and Veigh as Karnaugh Veigh diagrams. They descend from set theory. The KV-diagrams can be applied to equations corresponding to the disjunctive normal form!
How the KV-diagrams are structured and how to use them this is described in your Technical Data Manuals.

5.4.3. Exercises

Exercise 1:
Simplify the circuit `2 from 3` of the example above!

Exercise 2:
In order not to exceed the allowable electric power consumption of a foundry, a detector should give a signal when the power exceeds 22 kW. The rated power of the three kilns are: P1 = 16 kW, P2 = 12 kW und P3 = 8 kW.



Steps that you should follow:
1 Create a truth table.
2 Deduce a Boolean equation of this truth table.
3 Simplify, if possible, this equation by using KV-diagrams
4 Create a circuit of logic gates.

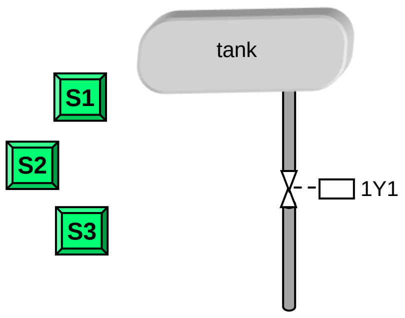
Exercise 3: tank-emptying

A tank should be emptied by three switches which are mounted at various locations. It should be able to switch the valve by a change-over switch with three contacts

1. Determine the relation between the inputs S1, S2, S3, and the output Y by using a function table.

2. Determine the logical equation.

3. Creating a logic circuit.



Exercises – Timers and counter

Exercise 1 - Temperature sensor

If a temperature sensor reports an elevated temperature (1 signal), then the indicator P1light will indicate this after 5s, but only if the temperature after this time is still increased. Another lamp P2 shows the increase in temperature also after 5 seconds, but, in contrast to the lamp P1, even if the temperature after 5s has decreased again. A reset button S1 turns off the lamp P2.

Select the required contacts (NO / NC) and develop your program!

Symbol table:

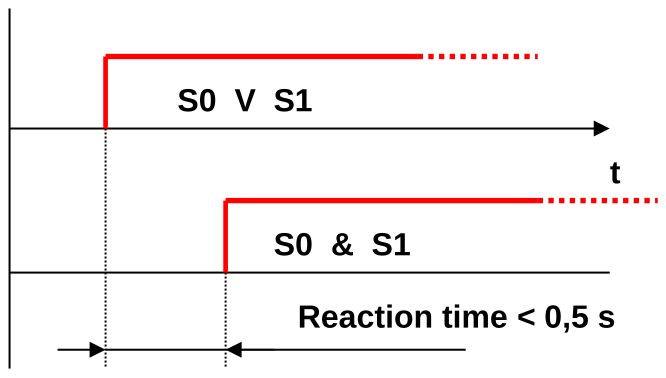
Variable	Address	Data Type	Comment
B1	I124.0	Bool	Tp.-Sensor
S2	I124.1	Bool	reset
P1	Q124.0	Bool	Lamp 1
P2	Q124.1	Bool	Lamp 2

Exercise 2 - Two handed control

A press is to be activated by a time-controlled two-hand control.

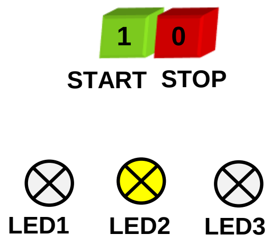
Note:

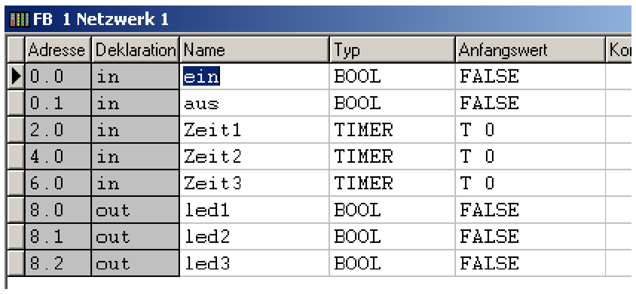
Left-hand button and right-hand button will always be pushed within a certain reaction time. However, this reaction time should not exceed a certain value, otherwise manipulation of this equipment can be suspected.



Variable	Address	Data Type
S1	I124.0	Bool
S2	I124.1	Bool
P1	Q124.1	Bool

Exercise 3: moving lights with 3 LED's




FB 1 Netzwerk 1

Adresse	Deklaration	Name	Typ	Anfangswert	Ko
0.0	in	ein	BOOL	FALSE	
0.1	in	aus	BOOL	FALSE	
2.0	in	Zeit1	TIMER	T 0	
4.0	in	Zeit2	TIMER	T 0	
6.0	in	Zeit3	TIMER	T 0	
8.0	out	led1	BOOL	FALSE	
8.1	out	led2	BOOL	FALSE	
8.2	out	led3	BOOL	FALSE	

Tip: You may take a sequence control (particular knowledge see volume II) program, in which form the individual time elements the transition condition for the next step. Each step is built by a SR-latch.

Exercise 4: Control for conveyor belt

A conveyor system for transporting bulk material is to be controlled. The system consisting of three individual bands.

Start the conveyor system: About the button S1, the system is started. All three belts will commence simultaneously.

Stop the conveyor system: To prevent material build-up, all belts should be emptied. The belts should be switched-off with a time delay of 3 seconds of each other.

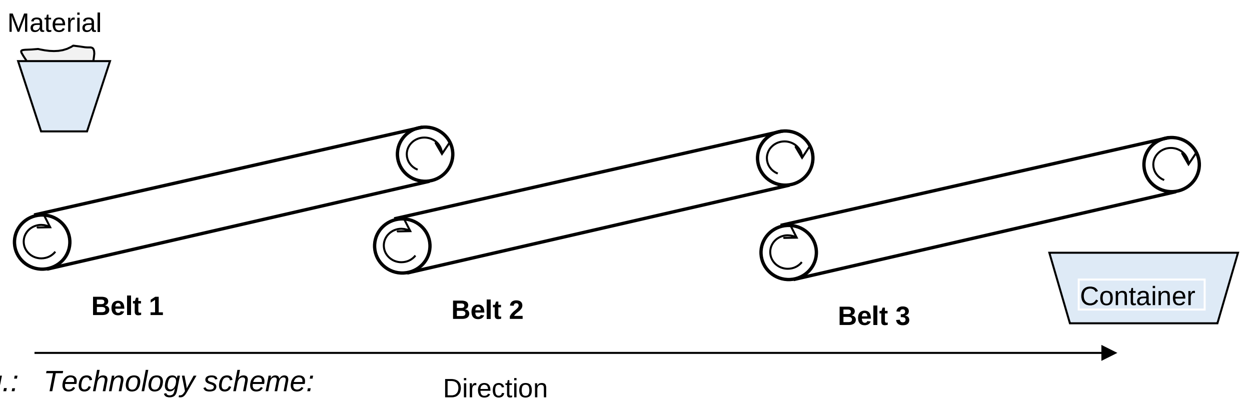
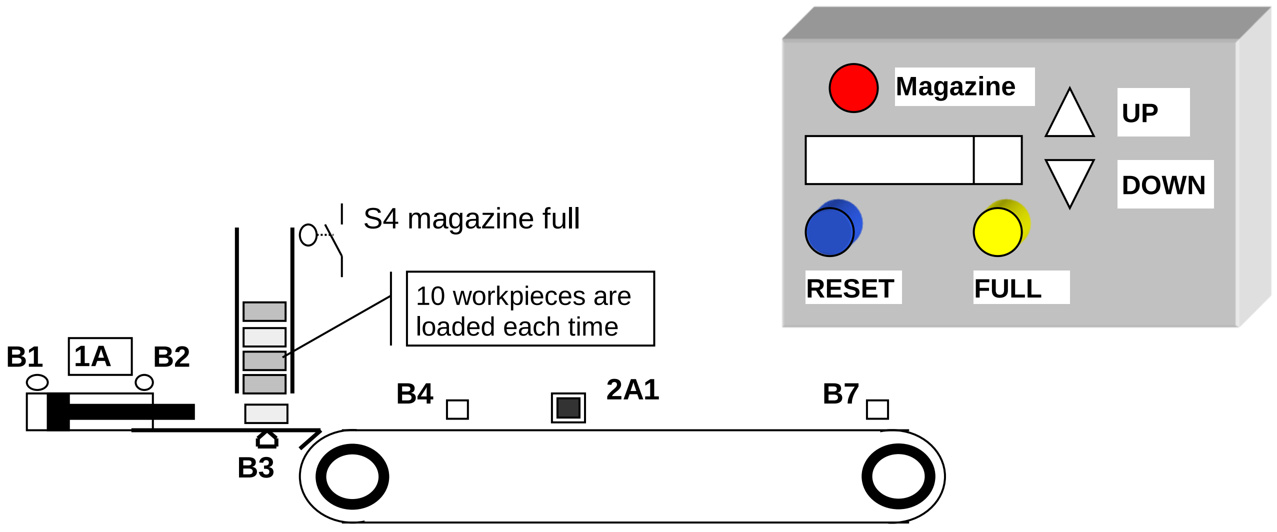


Fig.: Technology scheme:

Required: symbol table, program in FBD

Exercise 5: control for a magazine

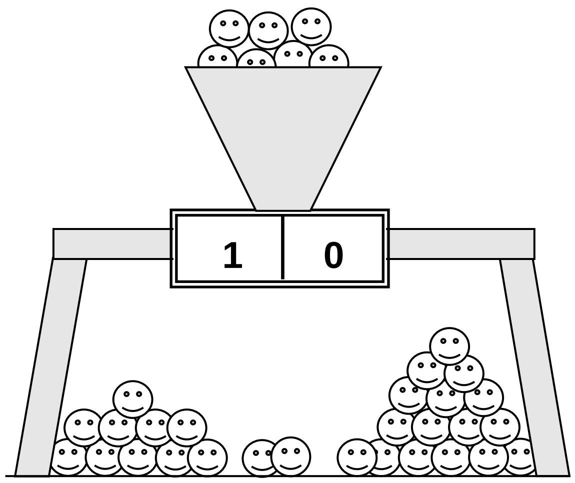


The level of the magazine is monitored by a light barrier. Before each new cycle 10 workpieces are inserted into the magazine. Once the cylinder 1A extends, the count-value is decremented. If a sufficient number of workpieces are in the magazine, then the indicator LED, Magazine' is switched off. If the number of workpieces is between 1 and 3, then should be signaled by a steady light of the indicator lamp H1 that the magazine must be refilled. If the magazine is empty, this should be indicated by a flashing of the indicator light.
After filling the button 'Reset' has to be pressed, so the equipment can continue its work process automatically.

Your program has to fulfill the following items:

- Increase the count by 1 with button UP '
- Reduce the count by 1 with button DOWN '
- Set the counter to 0 with button RESET '
- Set the counter to 10 with button FULL '

6. Introduction in number systems



Example: The device shown on the right counts \`smile's\`. The actual count value is \`1 0\` How many smile´s you have indeed?

a) ☺☺☺☺☺ ☺☺☺☺☺
b) ☺☺☺☺ ☺☺☺☺ ☺☺☺☺ ☺☺☺☺
c) ☺☺

We are used to represent numerical values in the decimal system. But in automation technology the decimal system is - just one of other equal number systems.

Characteristics of a number systems

All the number systems used today in have the following in common:
- The digit (for example 0 ....9, 0...f, etc.)
- The base
- The value within the number that means, which position takes the digit within the number?

In principle how the numerical value is formed, is for all number systems the same. This we want to know for different number systems with different bases.

6.1. Decimal-system

The decimal system has the following characteristics:
- ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Base: 10
- Value: Powers to the base number 10: 1, 10, 100, 1000, etc.

Example: **2 0 5**

$5 * 10^{0}$ = 5
$0 * 10^{1}$ = 0
$2 * 10^{2}$ = 200

<u>Sum = 205</u>

Die Darstellung der Zahl 205 ist in Wirklichkeit eine abgekürzte Schreibweise der Summe 200 + 0 + 5 = 205!

6.2.Dual system

The functioning of the computer is based on the dual system. The dual system is based on the basis 2. The base also determines the number of digits of the character set of the number system.

- two digits: 0, 1
- Base: 2
- Value: Powers to the base number 2: 1, 2, 4, 8, etc.

Example: **1 1 0 0 1 1 0 1**

$1 * 2^0 = 1$

$0 * 2^1 = 0$

$1 * 2^2 = 4$

$1 * 2^3 = 8$

$0 * 2^4 = 0$

$0 * 2^5 = 0$

$1 * 2^6 = 64$

$1 * 2^7 = 128$

Sum = 205

The example shows that a binary number has greater number of digits than the corresponding decimal number. With 8 digits binary numbers can be represented a number of value up to 255, with 16 digits up to 65 535.

6.3.Hexadecimal system

The Hexadecimal system has the number 16 for its base. As mentioned before, the base determines the number of single digit of the character set - here we have the digits 0 through 9 and the letters A to F.

The hexadecimal system is used to represent with as few digits and characters large numbers. In addition, a conversion of a binary number to hexadecimal number and vice versa is very simple.

- Digit: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Base: 16
- Value: Power to the base number 16

Example: **C D**

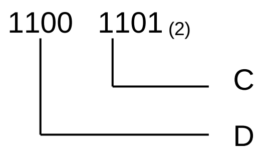
$13 * 16^0 = 13$

$12 * 16^1 = 192$

Sum = 205

Relation between Dual code and Hexadecimal code:

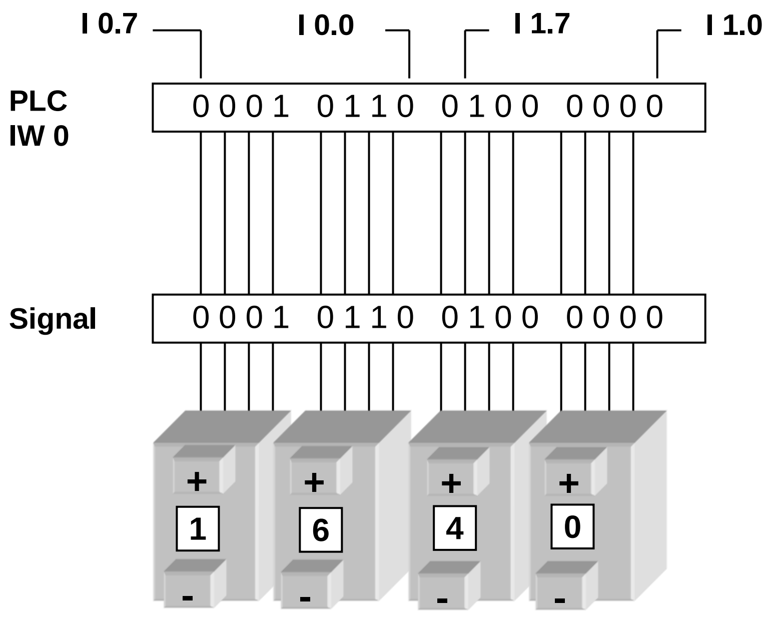
When we compare the binary coded number ‘1100 1101’ with the number in hex ‘CD’, we recognize, that always 4 binary digits build a hex digit. The reason is that with 4 binary numbers exactly 16 numbers can be displayed (from 0 to F).



Example: 1111 1111 1111 1111 $_{\text{Dual}}$ in Hex?

6.4. Binary coded Decimal (BCD)

The BCD-code converts each digit of a decimal numbers into a binary number and therefore is not another number system. The BCD code is required primarily for inputs and outputs.



This presentation is called Binary Coded Decimal or abbreviated BCD code. The individual digits are encrypted with four binary digits (bits). The representation with 4 bits results from the fact that the most significant decimal digit (9_{10}) requires in the binary representation at least 4 digits (1001).

For the presentation of the ten decimal digits 0 through 9 in BCD code the same representation is used as for binary numbers from 0 ... 9. 6 Of the 16 possible combinations with four binary digits the last 6 combination remain unused. We call these combinations "forbidden” and call them pseudo tetrads.

Decimal	**BCD-No**
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	**Pseudo-tetrads**
11	
12	
13	
14	
15	

Decimal	Dual	Hex.	BCD-Code
1	1	1	0001
2	10	2	0010
3	11	3	0011
4	100	4	0100
5	101	5	0101
6	110	6	0110
7	111	7	0111
8	1000	8	1000
9	1001	9	1001
10	1010	A	0001 000
11	1011	B	0001 0001
12	1100	C	0001 0010
13	1101	D	0001 0011
14	1110	E	0001 0100
15	1111	F	0001 0101
16	1 0000	10	0001 0110
17	1 0001	11	0001 0111
18	1 0010	12	0001 1000
19	1 0011	13	0001 1001
20	1 0100	14	0010 0000

6.5.Exercises number systems

1. Convert the number 57_{10} in dual.
2. Convert the number 8_{10} in dual.
3. Convert the number 0111_2 in decimal.
4. Convert the number 10001_2 in decimal.
5. Convert the number 0111_2 in hexadecimal.
6. Convert the number 10001_2 in hexadecimal.
7. Convert the number $57\ _{10}$ in hexadecimal.
8. Convert the number 8_{10} in hexadecimal.
9. Convert the number A_{16} in dual.
10. Convert the number B_{16} in dual.
11. Convert the number A_{16} in decimal.
12. Convert the number B_{16} in decimal.

7. Appendix

7.1. The most important data types

There are predefined elementary data types that can be used for programming in Step 7 in each program block (OB, FC, FB):

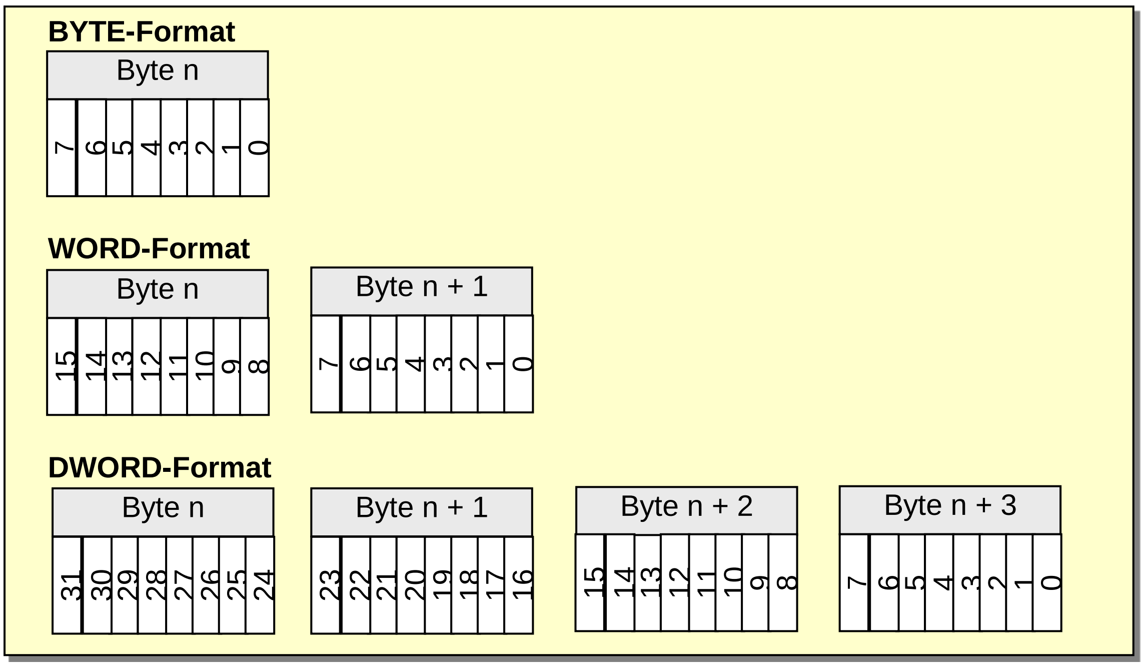
1. Bool

The data type 'BOOL' represents a bit value. The value of this data type is either TRUE or FALSE (0 or 1). The following operations take access of a single bit. Hereby the Bit is either read out or written.
Example:

```
U    I 124.0      //  AND Input I124.0
S    M 124.1      //  Set flag M124.1
```

2. Byte – Word - DWORD

This data bit sequences consists of 8, 16 or 32 bits and thus require 1, 2 or 4 bytes of memory space.



Example:

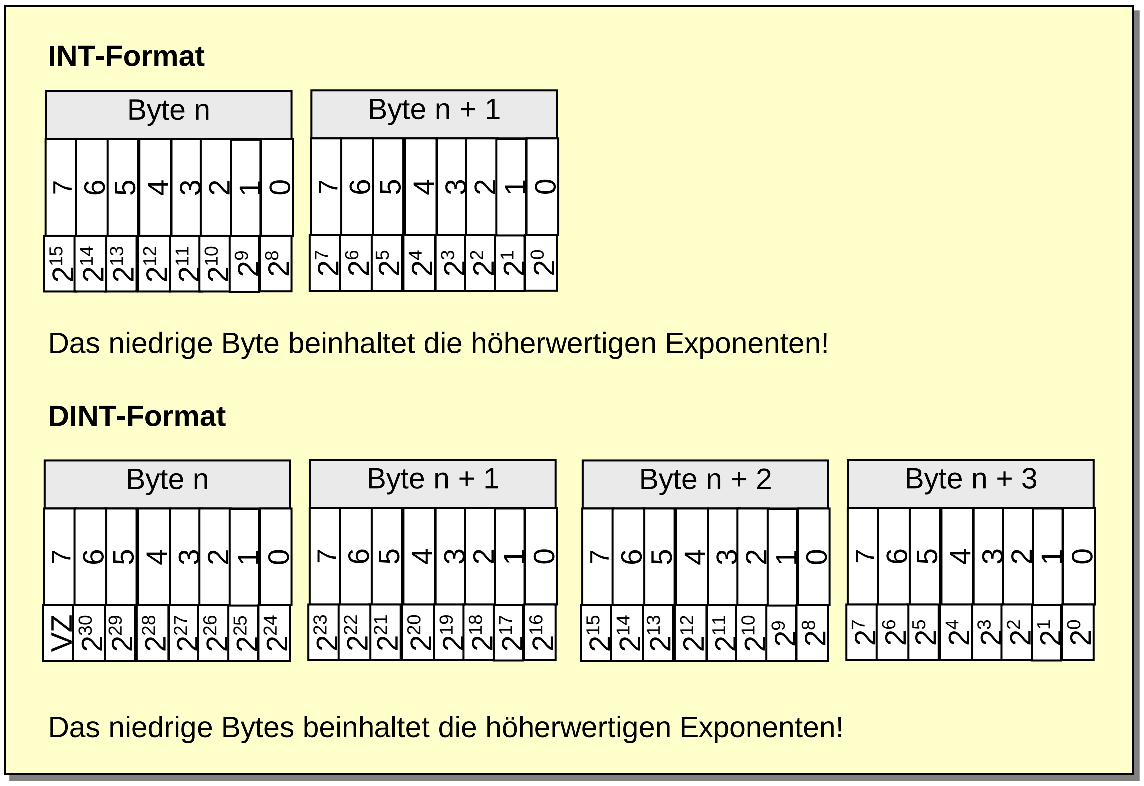
```
L    IB 124          Lade Eingangsbyte 124
T    QB 124          Transferiere in das Ausgangsbyte 124
L    IW 124          Lade Eingangswort 124
T    MW 100          Transferiere in Merkerwort
L    W#16#ABBA       Die Konstante mit dem Wert ABBAhex wird in Akku geladen
T    DB1.DBD 0       Transferiere in DB 1 in Datendoppelword 0
```

3. INT - DINT

Integers (INT) or **double integer (DINT)** represent whole numbers with a sign; they need 2 or 4 bytes of memory space. The range of values is as follows:

INT	from – 32768 / 8000$_{hex}$	to	32 767 (7FFF$_{hex}$)
DINT	from – 2 147 483 648 / 8000 0000 $_{hex}$	to	2 147 483 647 / 7FFF FFFF $_{hex}$

The structure of integer numbers, that means, the position and significance of the individual bits in the memory, is illustrated in the following graphic:



Das niedrige Byte beinhaltet die höherwertigen Exponenten!

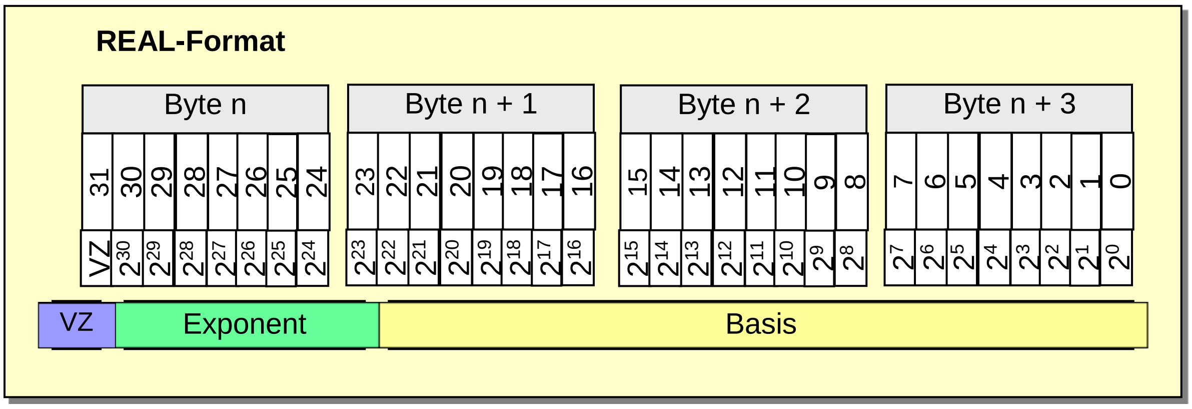
Das niedrige Bytes beinhaltet die höherwertigen Exponenten!

! Memory usage and value of an integer number behave oppositely.

Example:	L	1999	load Integer 1999 in Akku 1
	T	MW 6	transfer integer from Akku 1 in flag word 6
	L	L#123456	load die D-Integer 123456 in Akku 1
	T	MD 100	transfer in Flag word

4. Real-Zahlen

A real number represents a floating point number which requires 32 bits or 4 bytes of memory space. The highest bit (31) is reserved for the sign. This is followed by an exponent and a base.



The **value range** of a **real number** is shown in the following table:

negative	from – 3,402823 * 10^{+38}	to	–1,175494 * 10^{-38}
positive	from 1,175494 * 10^{-38}	to	3,402823 * 10^{+38}

To enable the editor to recognize that he got a REAL number, you must enter a decimal point (even if this should be 0) or alternative use the exponential notation:

Example:	L	1.0	becomes	L 1.000000e+000
	L	0.1	become	L 1.000000e-001
	L	2e2	becomes	L 2.000000e+002

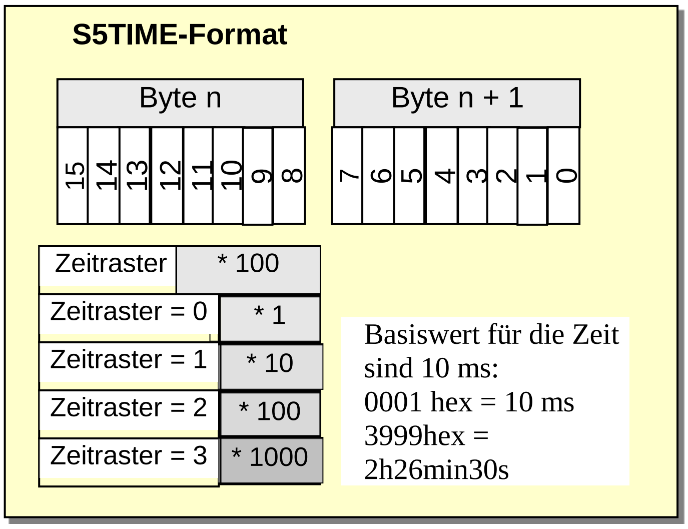
5. CHAR

A variable of the data type '**CHAR**' occupies one byte memory. It contains a character in the ASCII format. Enter a single character as follows:

Example:	L	' A '	loads the character 'A' right-aligned in Akku 1
	L	' AB '	loads the character A and B right-aligned in den Akku 1
	L	' ABCD '	loads the character A, B, C and D in Akku 1.

6. S5TIME

The data type S5TIME is used together with the S5-timers and represents a BCD-coded time value. The first Byte of the determined time frame.



Data type	**description**	**No. of Bits**	**Format, range**
BOOL	single Bit	1	true / false
BYTE	next larger unit of information	8	B#16#00 … B#16#FF
WORD	2 Bytes Δ 1 WORD	16	2#0000_0000_0000_0000 bis 2#1111_1111_1111_1111
DWORD	double Word 2 WORD Δ 1 DWORD	32	DW#16#00000000 … DW#16#FFFFFFFF
INT	Integer	16	-32768 …0 … 32768
DINT	Long integer	32	L#-2147483648 ... L#0 … L#2147483648
REAL	floating-point number	32	
S5TIME	Time value Simatic	16	S5T#1h33m22s, S5 T#11ms
COUNT VALUE	Count value Simatic	16	C#0, C#999
TIME	Time value IEC	32	T#-24d20h31m23s647ms TIME# -24d20h31m23s647ms T#-24.855134d TIME#-24.855134d
DATE	Date	16	D#2013-12-31 DATE#2013-12-31
CHAR	ASCII-character	8	‚U‘ ‚H‘
STRING	ASCII-string	variable	brasil
ARRAY	Data field	variable	Depends on the number and def. of the components
STRUCT	Data structure	variable	Depends on the number and def. of the components
TIME_OF_DAY	Time of day	32	TOD#2013-12-31-22:59:59.999 DATE_AND_TIME#2013-12-31-22:59:59.999
DATE	Date	16	D#2013-11-30 DATE#2013-11-30
DATE_AND_TIME	Date and time	64	DT#2013-12-11-09:33:12.000 DATE_AND_TIME#2013-12-11-09:33:12.000