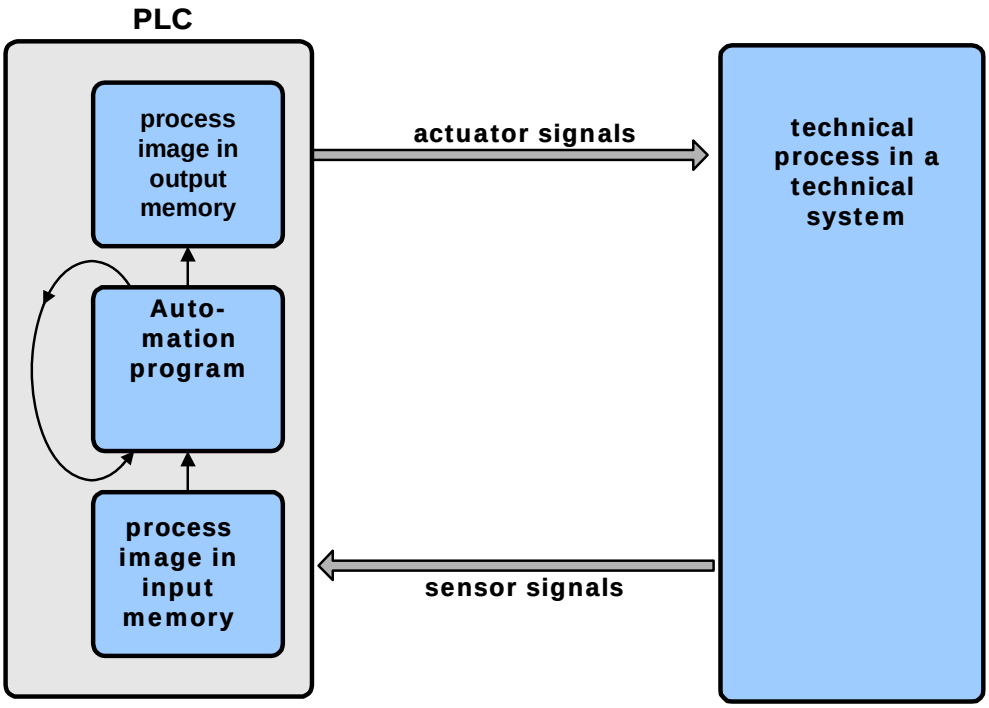
Exercise 8 – PLC Programming

Theory

Functioning of a PLC

A PLC is working in cyclically. At the beginning of each cycle the current input data is read in and stored in the memory. This procedure is denoted as creation of a process image. Following to that the actual processing is done i.e. the calculation of the new basic values based on the stored process image. Changes in the technical process during this calculation phase are not (yet) recognized by the control and so they are not considered in the calculation. After finishing the calculation, the output values in the output memory are released to the technical process and the cycle starts again. Whereby the cycle time isn't constant but depending on the program and execution path.



Identifier

According to the functioning of a PLC, it is differentiated between thee kind of values: inputs, outputs and internal values. Internal values include elements like flip-flops, timing relays and many more. The Identifiers for the particular elements are partly dependent on the manufacturer and the language so there are differences in practice. A general form is:

I<No.>	Input
Q<No.>	Output
...	Internal values

Whereby <No.> is a positive integer (e.g. I1, Q5....)

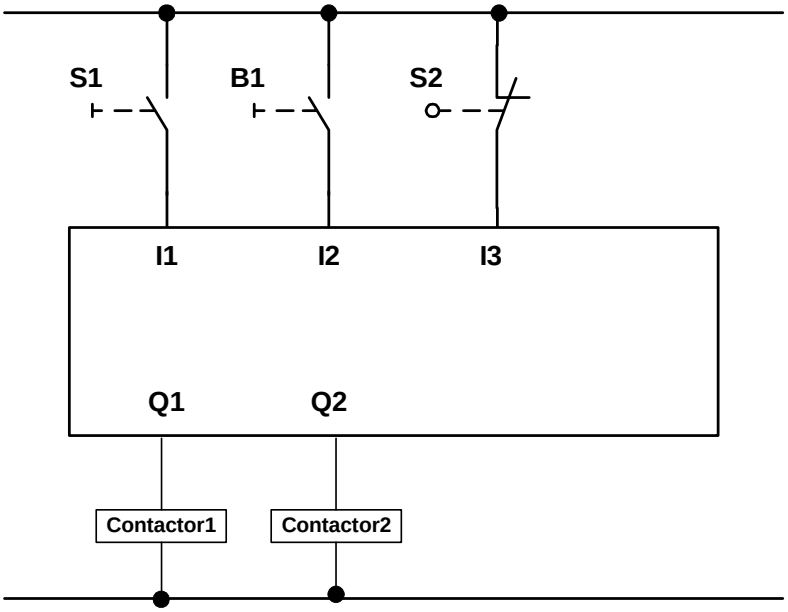
The company Siemens is using a slightly different identification for their widely distributed Simatic control. There the number consists of two digits. This combination results from the applied I/O modules.

E<No.>.<No.>	Input
A<No.>.<No.>	Output
....	Internal values

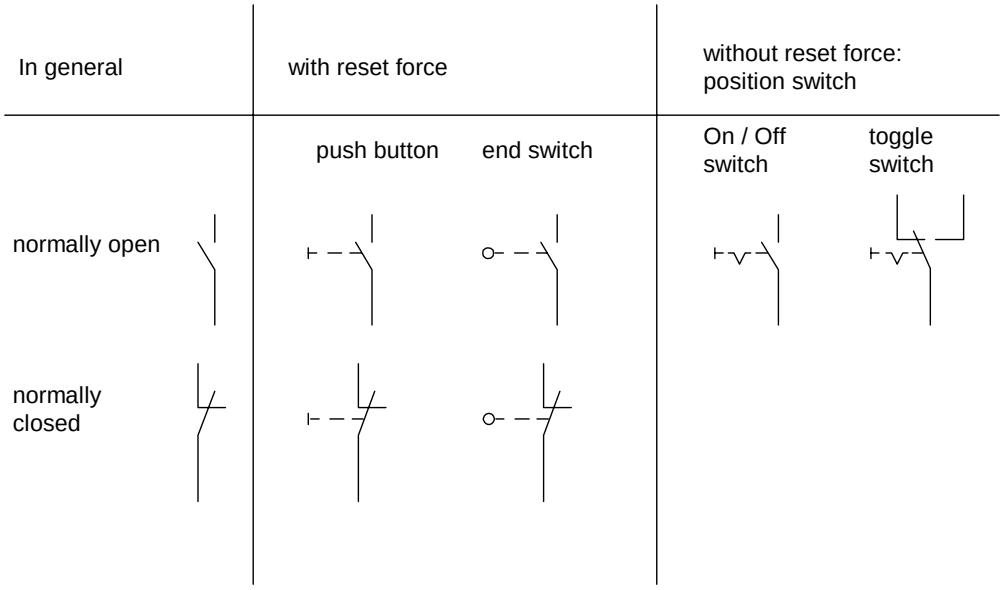
Example: E0.0, A4.1

Circuit of PLC:

In order to realize the internal calculation in a correct way, the circuit plan of the PLC has to be on hand. This gives information about which real process value is switched on which input. The following illustration gives an example for that.



The connected switches can be realized like that:



PLC Programming Languages

There exists large number of languages for programming of programmable logic controller because each manufacturer develops own languages for the controller. However there are four (mostly) standardized languages:

- Structured text
- Instruction list
- Ladder diagram
- Function block language

Besides the first mentioned language “structured text” the others are oriented on the logic of the circuits. This is reflected in the form of programming.

Structured Text

Structured text is a language similar to Pascal which is intended to be for users with programming experience. It offers known structures like loop constructs, conditional branching and so on. It won’t be discussed in further detail.

Instruction List (IL)

The instruction list corresponds to a textual translation of logic connections, like AND, OR, NOT and many more. Thereby every instruction row is built up in the same scheme:

Kind of connection	Identifier
U	E0.0

Example:
One output signal should be switched depending on two input signals (Ouput = Input1 and Input2). The associated instruction list results in:

```
U     E0.1
U     E0.2
=     A0.1
```

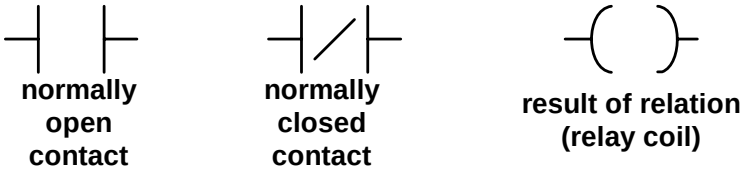
The following table gives an extract of the IL-language elements of the Siemens SPS but is not exhaustive.

Mnemonic	**Description**
=	Assignment
)	Close branching
AUF	Open data block
BEA	Block end absolute
BEB	Block end conditional
CALL	Block call-up
FN	Negative edge
FP	Positive edge
L	Load
LOOP	Program loop
NOT	Negate RLO
O	OR
O(	OR with branching
ON	OR NOT
ON(	OR NOT with branching
R	Reset
S	Set
SA	Timer as OFF delay
SE	Timer as ON delay
SI	Timer as pulse
SET	Set RLO (=1)
SPA	Jump unconditionally
SPB	Jump, if RLO=1
SPBN	Jump, if RLO=0
SPN	Jump, if result <> 0
SS	Timer as a latching ON delay
SV	Start timer as an extended pulse timer
T	Transfer
U	AND
U(	AND with branching
UN	AND NOT
UN(	AND NOT with branching
X	EXCLUSIVE OR
X(	EXCLUSIV OR with branching
XN	EXCLUSIV OR NOT
XN(	EXCLUSIV OR NOT with branching

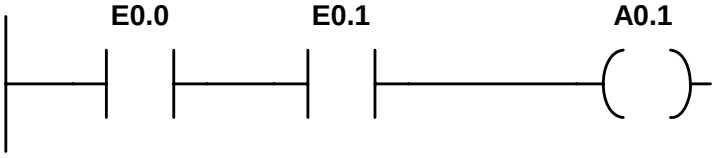
Mnemonic	Description
ZR	Count down
ZV	Count up

Ladder Diagram (LAD)

The nature of the ladder diagram is strongly based on circuit diagrams. AND relations are realized as serial connections and OR relations are realized as parallel connections. Three kinds of elements a distinguished: normally open contact, normally closed contact and the result of the relations. Whereby all the existing values including internal values and outputs, can represent a relation element and take over the function of a switch.



The above mentioned example would have the following form:



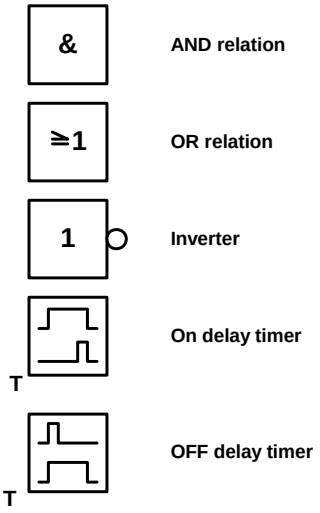
The following tale gives an extract of the LAD language elements of the Siemens Simatic PLC:

Mnemonic	Description
-(ZR)	Count down
-(ZV)	Count up
-(R)	Reset output
-(S)	Set output
-(AUF)	Open data block
-(CALL)	Call up FC / SFC without parameter
-(P)-	Query edge 0→1
-(N)-	Query edge 1→0
-(#)-	Connector
-|/|-	Normally closed contact
-( )	Relay coil / output
-| |-	Normally open contact
-(POS)	Query signal edge 0→1
-(NEG)	Query signal edge 1→0
-(JMPN)	Jump if 0
-(JMP)	Jump if 1
-|NOT|-	Invert result of logic operation (RLO)
-(SZ)	Set counter start value

Mnemonic	Description
-(SA)-	Start timer as OFF delay (SA)
-(SE)-	Start timer as ON delay (SE)
-(SI)	Start timer as pulse
-(SS)	Start timer as latching ON delay
-(SV)	Start timer as extended impulse

Function block language (FBL)

The function block language is a bit stronger based on the logic elements that are combined with each other. The following list gives an not exhaustive overview of the standard modules.



So the example would look like this: