Introduction to the PLC

What's a PLC?

PLC: Programmable Logic Controller

It's a control unit, able to command many actuators, when its connected sensors measure a variation of a specific quantity.

What's a PLC?

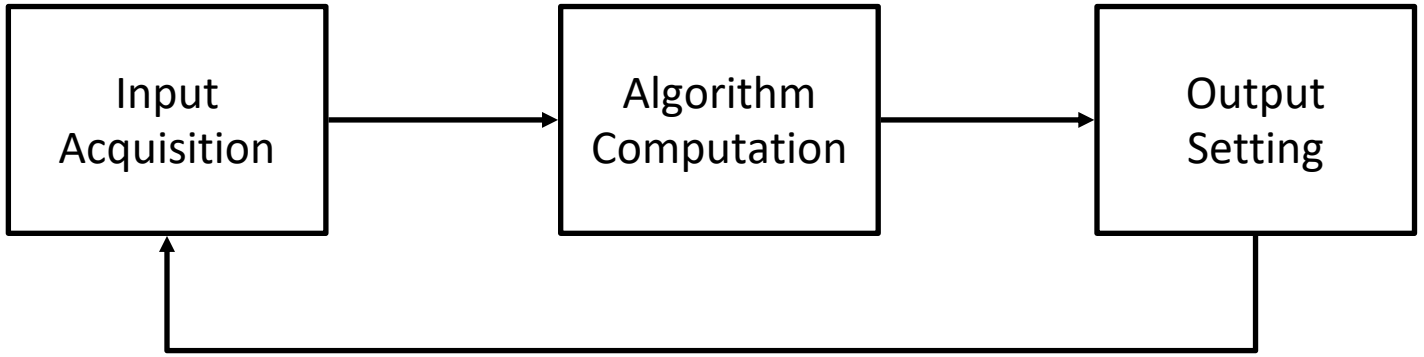
PLC: Programmable Logic Controller

To make the PLC working, it requires a signal with fixed frequency: that's why it has to be real-time.

But, what does "real-time" mean?

"Real-time" means that we can know in advance how much it takes to execute a single piece of code.

PLC Execution Cycle



N.B.: Usually the relaunch time of the execution cycle is fixed. However, sometimes with some PLCs (such as Siemens') if you don't define a function block with a fixed sampling time, the code is executed in an "infinite loop" (in a main function with a "while true" loop). In that case is not possible to know the relaunch time.

Real-time

Each PLC producer customizes one or more OSs to allow the integration with its devices and its environment.

The most common used OSs in industrial automation are:

- VXWorks
- QNX
- Windows embedded (or CE)

Real-time

How does a real time O.S. work?

A real time operative system is a deterministic system, i.e. a system that is able to guarantee in advance the maximum execution time of a piece of code.

Definitions:

- Release time: instant in which the program is available for execution
- Deadline: instant in which the PLC must complete the execution of the code
- Completion time: instant in which the execution is completed

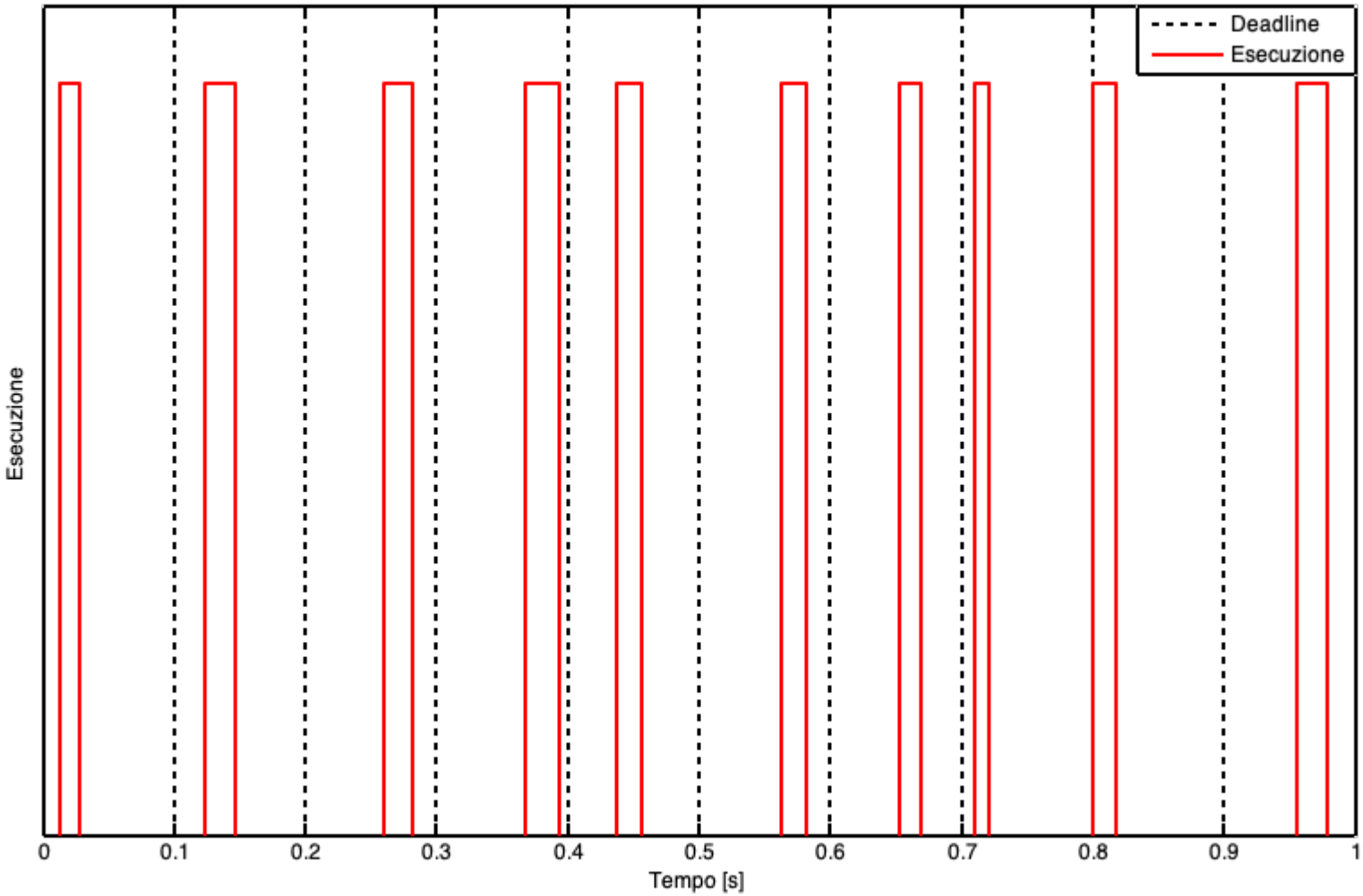
Real-time

When working with a PLC, usually, the previous defined terms are:

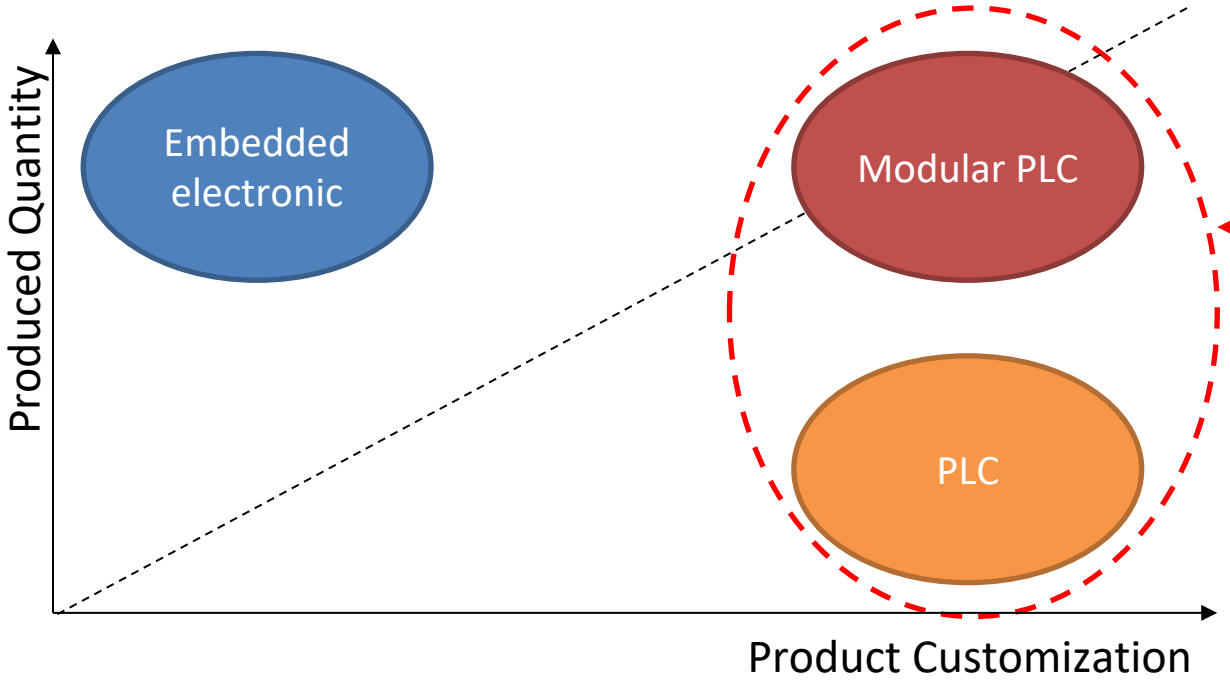
- Release time: the program is ready for the execution at the end of the next deadline.
- Deadline: it's equal to the relaunch time of the program.
- Completion time: it depends on the load of the CPU over which the program is executed

Real-time

Execution example:



When do we need a PLC?



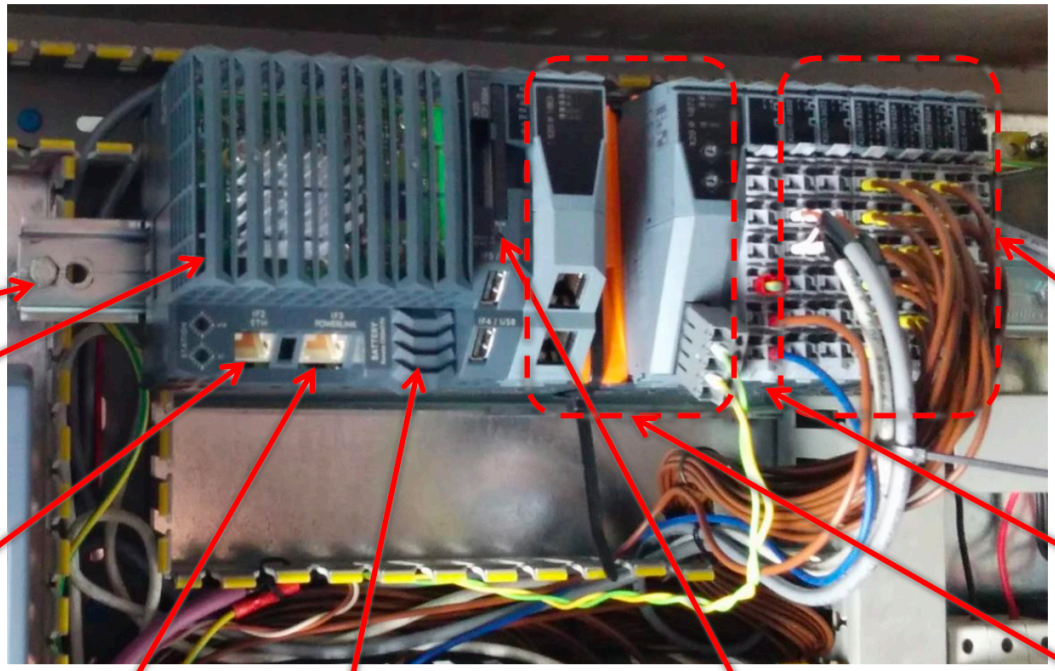
PLC is useful to develop customized systems in reduced number (or for a single-product, such as production lines), when the complexity and the number of I/O interfaces is really high

PLC Types

µPLCs	Middle dimension PLCs	Complex PLCs
		
<= 64 inputs / outputs	< 512 inputs / outputs	>= 512 inputs / outputs

Nowadays, the most of the used PLCs are «modular», so it is possibile to increase the number of inputs and outputs

PLC Structure



Hardware Architecture

The trend of recent years is to bring the PLC architecture closer to that of a normal x86

Usually the memories in the PLCs are:

- EEPROM: in which is stored the S.O. and the control software (on the latest PLCs these memories are removable such as CF, SD, etc…)

- RAM: in which is allocated the software during its execution and the variables. On some PLC there is a backup battery that allows to maintain stored these values in the RAM also in case of power failure.

Hardware Architecture

There are many extension I/O modules. The most common (available for the most of producers) are:

- Analog In/Out ($\pm$10 V, -20÷20 mA, 4÷20 mA)
- Digital In/Out (5 V-DC, 24 V-DC, 240 V-AC)
- Motor control module (encoder, PWM)
- Various input (thermocouple, load cells, etc…)
- Interfaces for field buses

How to program a PLC

All the PLCs are programmable by a PC (usually with the O.S. Windows) by connecting the PLC to a PC using the Ethernet interface.

Almost all the producers use proprietary software for PLCs programming, although some of them are compatible with third-party software (e.g. Codesys)

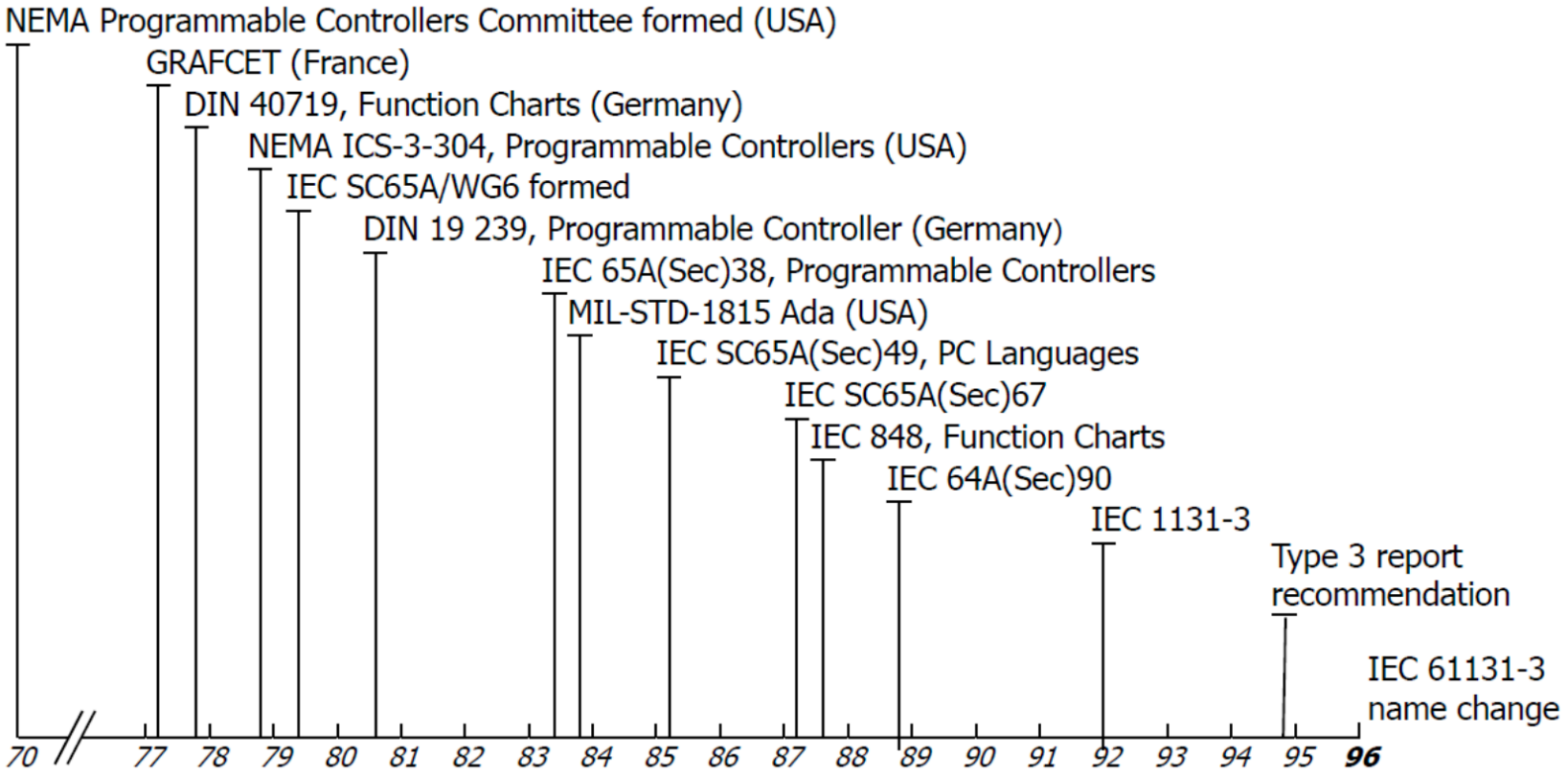
N.B.: We will use the official SW: B&R Automation Studio 4.0

IEC 61131

IEC is a non-profit and non-governative organization born with the aim to define and publish electrical and electrotechnical technology standards.

For what concerning the PLC's standards, the reference standard is the IEC 61131 norm (before known as IEC 1131... Its name changed in 1996).

IEC 61131


NEMA Programmable Controllers Committee formed (USA)
GRAFCET (France)
DIN 40719, Function Charts (Germany)
NEMA ICS-3-304, Programmable Controllers (USA)
IEC SC65A/WG6 formed
DIN 19 239, Programmable Controller (Germany)
IEC 65A(Sec)38, Programmable Controllers
MIL-STD-1815 Ada (USA)
IEC SC65A(Sec)49, PC Languages
IEC SC65A(Sec)67
IEC 848, Function Charts
IEC 64A(Sec)90
IEC 1131-3
Type 3 report recommendation
IEC 61131-3 name change
70
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96

IEC 61131

Why is this norm so important?

- Because it is a standard de-facto in the industry
- Because the 80% of the sold PLCs uses this norm to define their programming languages.
- Because it helps to create a software which is coherent with the ones written by other people.

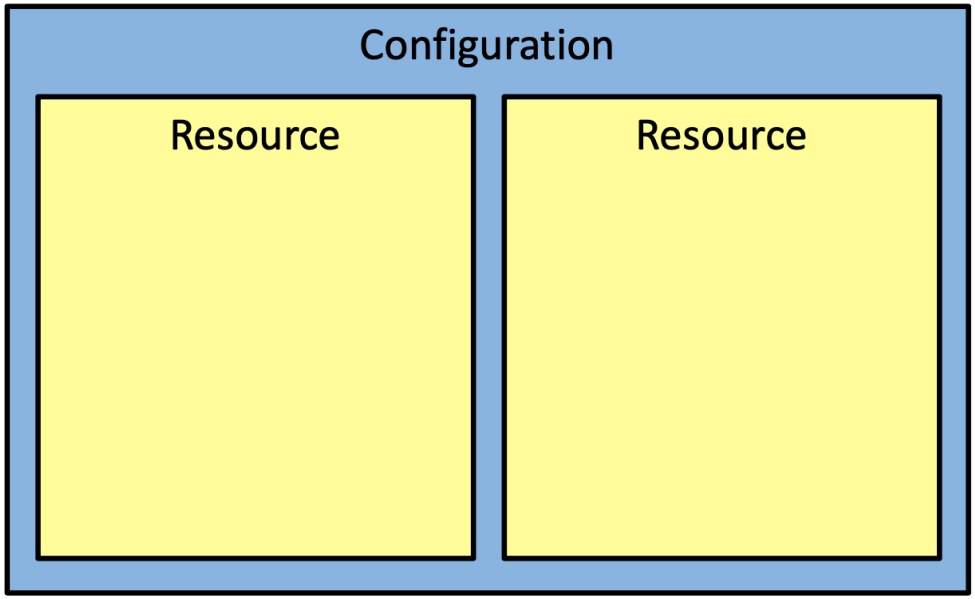
IEC 61131

The norm is divided into 8 parts:

1. General overview, definitions
2. Hardware
3. Programming languages
4. Guidelines for the users
5. Communication
7. Fuzzy-logic programming
8. Guidelines for the implementation

Structure of a PLC application

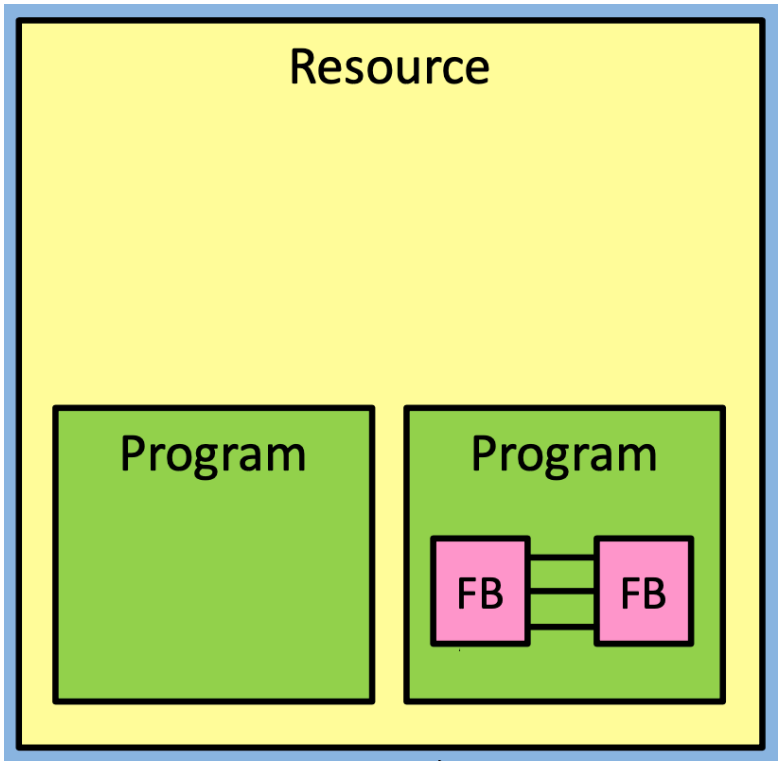
Generally PLCs are multitasking.

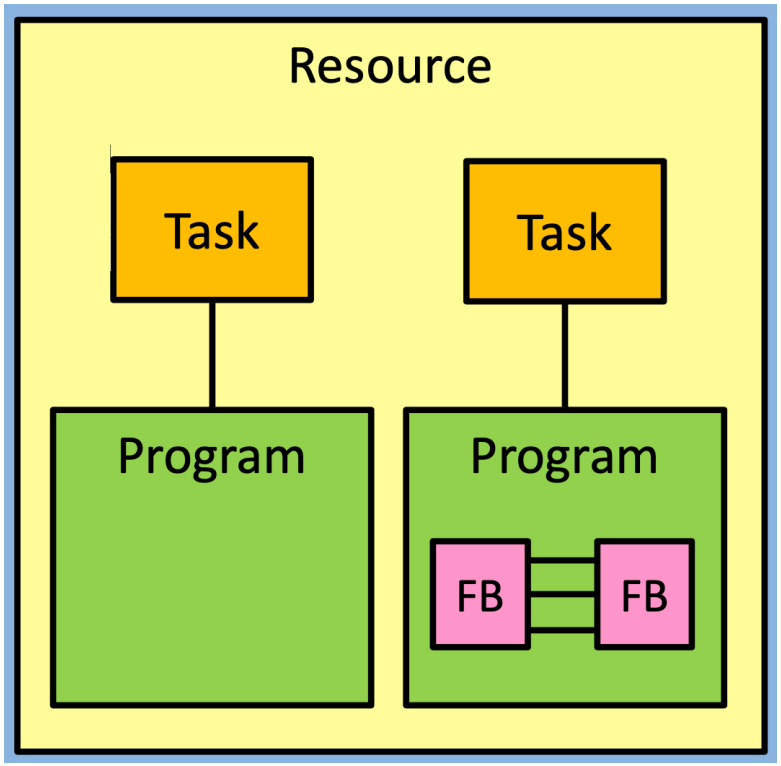


The SW defines the HW configuration of the PLC. Each configuration contains several resources (PLC, CPU, I/O Modules, ...)

Structure of a PLC application

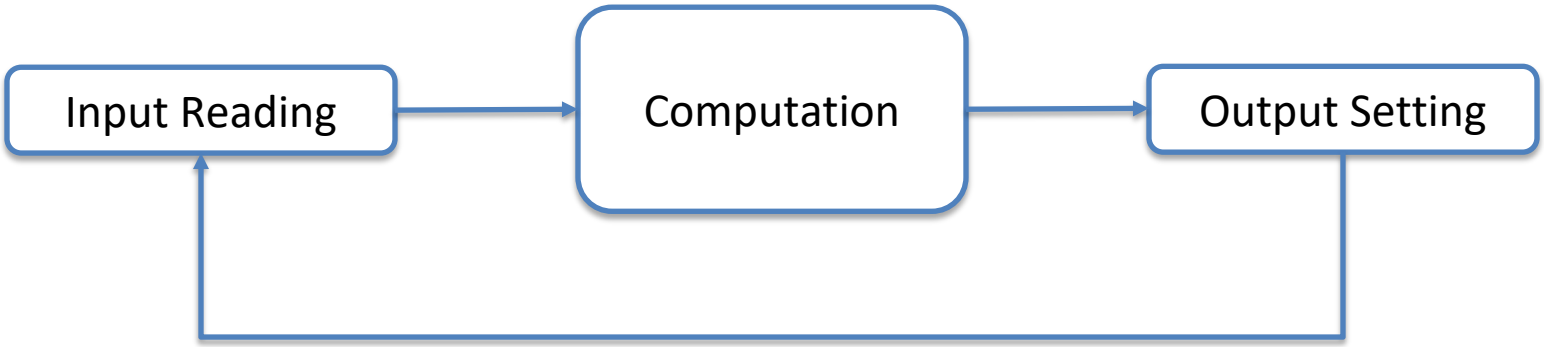
In each resource, we can have many programs, with different tasks.





Structure of a PLC application

Every program follows the same structure:



Structure of a PLC application

The communication between programs can be:

- Inside the same configuration:
  - In a direct way, if the two entities interacting are function blocks (FB) of the same program
  - Using global variables, if the two entities interacting are two different programs

- Between two configurations:
  - In a direct way, if the two entities interacting are communication function blocks (FB)
  - Using a shared path

Programming languages

In the part 3 of the IEC 61131 norm, the 5 available programming languages are described:

- Ladder Diagram
- SFC – Sequential Function Chart
- FBD – Function Block Diagram

} Graphical languages

- Instruction List
- ST – Structured Text

} Textual languages

N.B.: To guarantee the real-time execution of the code, all these languages are translated in low-level instructions

Programming languages

Common elements of the 5 languages of the IEC 61131 norm:

- Variable names
  - The first character can't be a number
  - It's not possible to have two consecutive "_"
  - It's not possible to have spaces
- Keywords
  - PROGRAM, FUNCTION, VAR, END_, etc…
  - BOOL, BYTE, WORD, INT, REAL, TIME, STRING, prefix S, D, L, U, etc…
  - RETAIN, CONSTANT, etc…

Programming languages

Common elements of the 5 languages of the IEC 61131 norm:

- Mathematical and logical functions:
  - ADD, SQRT, SIN, COS, GT, MIN, MAX, AND, OR, etc…

Programming languages

Ladder Diagram

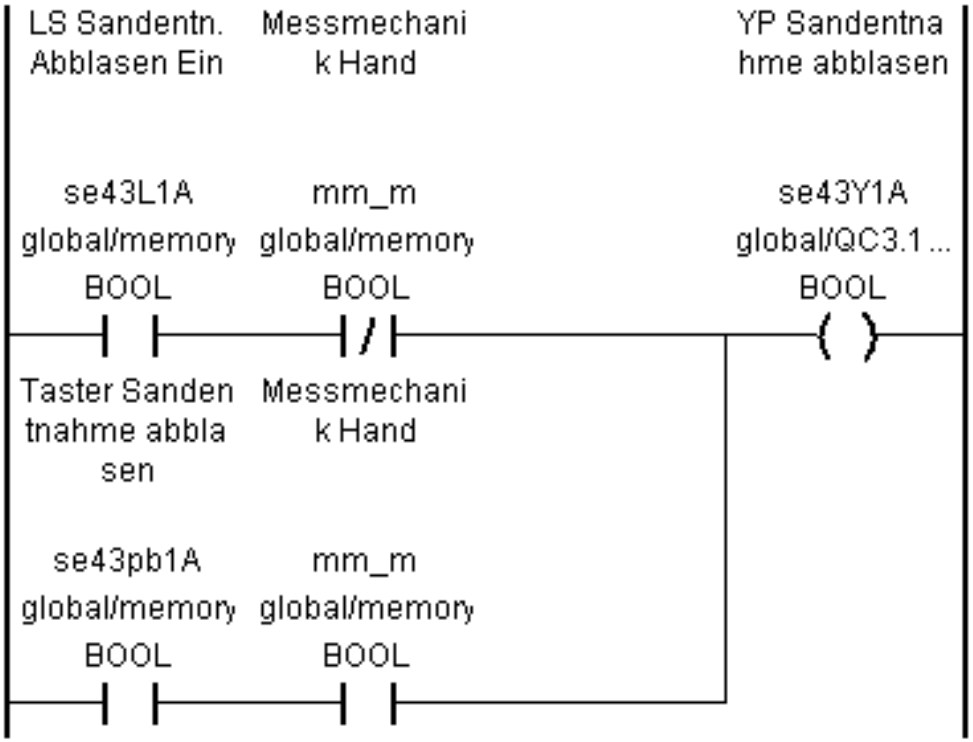
It's the former programming language in the IEC 31131 norm.

Ladder is based on symbols of electrical origin: Rung, Contacts, Coils

It is called "Ladder" because of the aspect that the software realized with this language have.

Programming languages

Ladder Diagram



Programming languages

SFC – Sequential Function Chart

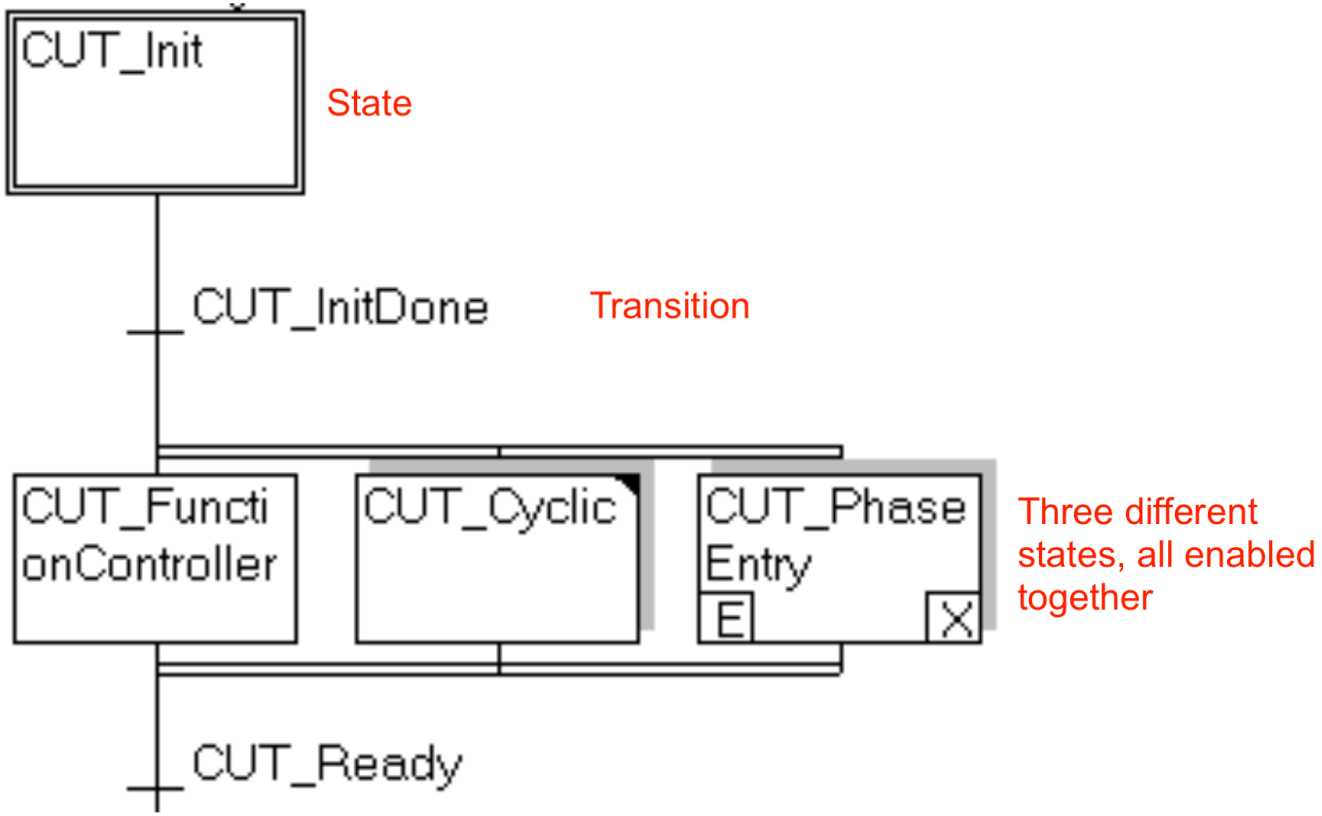
This language derives from the standard IEC 848.

It is a language «sequencing-oriented» and so, it is a language useful to a top-down development.

Ladder is based on: Steps, Transitions, Actions and Oriented Arrows

Programming languages

SFC – Sequential Function Chart



Programming languages

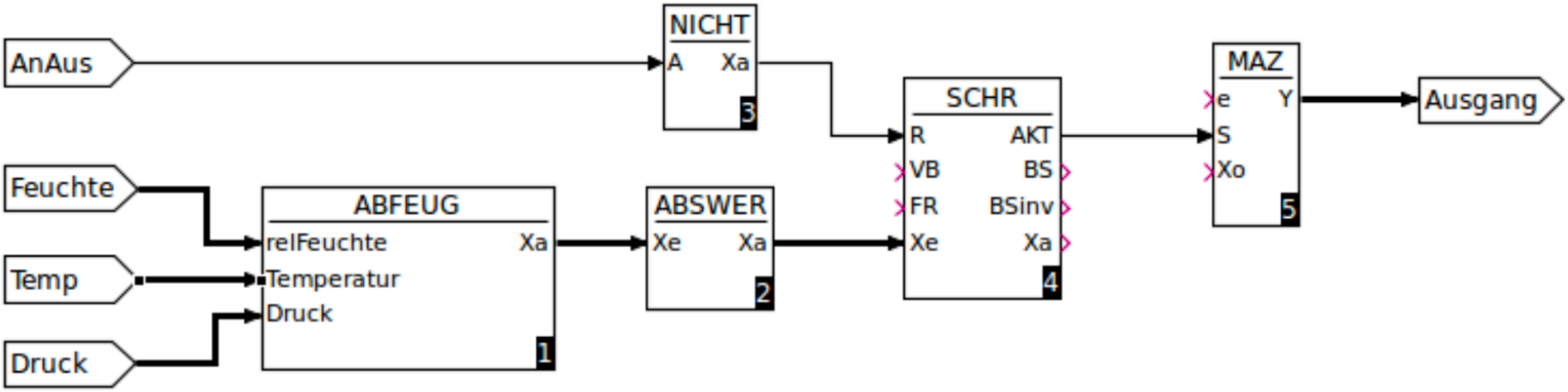
FBD – Function Block Diagram

The FBD language derives from the standard IEC 167. Its graphical representation it similar to that of the block diagram, In which the execution depends on the position of the function blocks.

It’s not a widely used language for the logical control.

Programming languages

FBD – Function Block Diagram



Programming languages

Instruction list

The «Instruction List» is the lowest level language in the IEC 61131 norm.

It is similar to the assembly code, so it is not used in the industry, because it is very complicated to be written.

Programming languages

Instruction list

1.1	LD	$I_{0.0}$	Input Contact
1.2	MOVW	# 980 , VW200	Put 980 in VW200
1.3	LD	SM 0.5	Pulse generator
1.4	EU		Raising edge
1.5	SLW	VW200 # 1	Shift Left Word (1) bit
1.6	LD	SM1.1	Overflow =1 if last bit shifted output =1
1.7	=	$Q_{0.0}$	Output Coil
1.8	MEND		End programming

Programming languages

ST – Structured Text

It's based on some old languages, such as Pascal and Visual Basic.

It is one of the most high-level languages of the IEC 61131 norm.

Some tools allow to generate ST code from other programming languages, block diagrams or models (e.g. the tool "PLC Coder" in Mathworks)

Each PLC producer has its own «dialect» of ST (We will se the B&R «dialect»)

Programming languages

ST – Structured Text

```
FUNCTION_BLOCK FBControl
CASE ssMethodType OF
    0:
        (* Start for Enabled SubSystem: '<S1>/EnabledCase' *)
        (* Start for Enabled SubSystem: '<S3>/Dos' *)
        (* InitializeConditions for Atomic SubSystem: '<S6>/PID - antiwindup' *)

        (* InitializeConditions for UnitDelay: '<S12>/Unit Delay2' *)
        UnitDelay2_DSTATE := 0.0;

        (* InitializeConditions for UnitDelay: '<S10>/Unit Delay' *)
        UnitDelay_DSTATE_a := 0.0;

        (* InitializeConditions for UnitDelay: '<S12>/Unit Delay1' *)
        UnitDelay1_DSTATE := 0.0;

        (* InitializeConditions for UnitDelay: '<S11>/Unit Delay2' *)
        UnitDelay2_DSTATE_f := 0.0;

        (* InitializeConditions for UnitDelay: '<S11>/Unit Delay1' *)
        UnitDelay1_DSTATE_k := 0.0;
        (* End of InitializeConditions for SubSystem: '<S6>/PID - antiwindup' *)
        (* End of Start for SubSystem: '<S3>/Dos' *)
```

Programming languages

During this lessons we will see three different languages:

- Ladder Diagram
- SFC – Sequential Function Chart

- ST – Structured Text