HW and SW technologies for industrial automation

Leonardo Labs
IEC 61131-3 standard - IEC 61131 programming languages - Sequential functional chart

Gianmaria De Tommasi
Email: detommas@unina.it

October 2020

DIETI UNINA UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

Outline

The **Part 3 of the International Standard IEC 61131 (IEC 61131-3** defines:

- the data types
- the software architecture
- the programming languages

to be used for **developing and deploying a control software within an architecture based on Programmable Logic Controllers** (PLC).

- First released in 1993
- Second edition in 2002
- Third edition in 2012 (that support OOP)

Data types - Examples

- Elementary Data Type
  - `BOOL`: 1 bit (1 byte is allocated)
  - `BYTE` : 8 bit (1 byte is allocated)
  - `WORD`: 16 bit (2 byte are allocated)
  - . . .
  - `LWORD`: 64 bit (8 byte are allocated)
  - `INT`: signed integer (2 byte is allocates)
  - `UINT`: unsigned integer
  - `REAL`: floating point
  - `CHAR`: single-byte character
  - `STRING`: variable-length single-byte character string
  - `TIME`: time values in the form of `T#5m90s15ms`
  - `DATE`: calendar date
  - `ANY`: generic data type

Examples - Bit-valued types and integers

Type	Min	Max	Dimension
BYTE	0	255	8 bit
WORD	0	65535	16 bit
DWORD	0	4294967295	32 bit
SINT	-128	127	8 bit
USINT	0	255	8 bit
INT	-32768	32767	16 bit
UINT	0	65535	16 bit
DINT	-2147483648	2147483647	32 bit
UDINT	0	4294967295	32 bit

Data types - Examples

- User-defined Data Types
  - Enumerated data type
  - Subrange data type. Example: `INT(4..20)`
  - Array data type. `ARRAY[1..10] OF ...`
  - Structured data type. `STRUCT ... END_STRUCT`

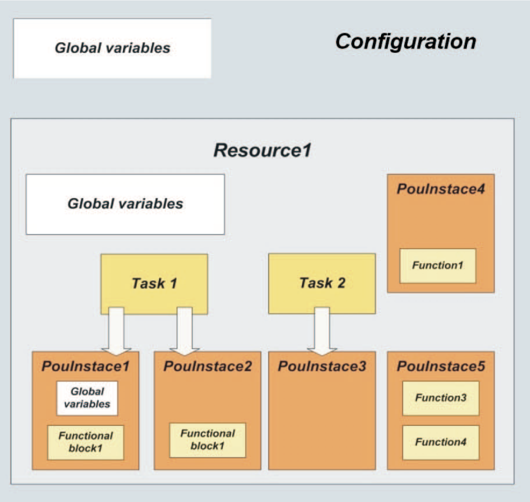
Program organization units (POUs)

- Functions (sometimes referred to as FCs)
- Function blocks (sometimes referred to as FBs)
- Program

Configuration objects

- `RESOURCE` (CPU in a control device)
- `TASK` (control task executed by a resource with a given execution mode)

IEC 61131-3 configuration: an example

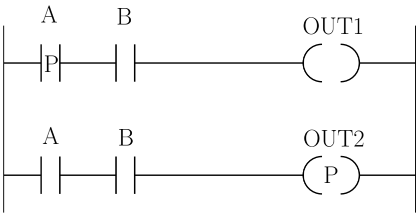


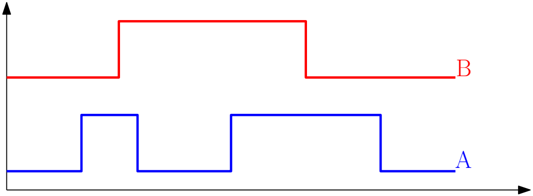
Programming languages

- The IEC 61131-3 standard includes five programming languages:
- ladder diagram (LD)
- functional block diagram (FBD)
- instruction list (IL)
- structured text (ST)
- sequential functional chart (SFC)

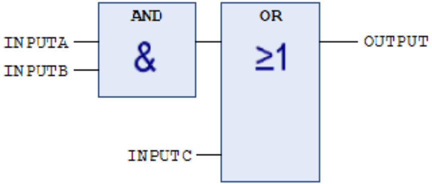
- Two text languages (IL and ST) and three graphic languages (LD, FBD and SFC)
- Three *low-level* languages (LD, FBD and IL) and two high-level ones (ST and SFC)

Ladder diagram





Functional block diagram



Instruction list

```
            LD      Speed
            GT      2000
            JMPCN   VOLTS_OK
            LD      Volts
VOLTS_OK    LD      1
            ST      %Q75
```

Structured text

GAUSS_FORMULA | MAIN

```
FUNCTION GAUSS_FORMULA : INT
// Computes the sum of the first N integer number by using the Gauss formula
VAR_INPUT
// Input integer
    N:INT;
END_VAR
VAR
END_VAR

```

```
GAUSS_FORMULA := N*(N+1)/2;
```

Sequential Functional Chart (SFC) as programming language for PLCs

- SFC is a graphical oriented language, derived from the **Petri nets**, which is a formal tool used to describe the behaviour of discrete event-driven systems (DES)
- With respect to other formal tools, such as *automata* Petri nets, and hence SFC, allow to easily represent the parallelism
- As a programming language, the SFC has been standardized by IEC as evolution of the Grafcet graphical programming language
- The SFC programming language is available on all the major commercial automation platforms

R. Alla
Grafcet: a powerful tool for specification of logic controllers
IEEE Transactions on Control System Technology, 1995

Why using SFCs?

- The use of SFC has a threefold advantage
  - it allows to formally specify control logics without ambiguities (being a formal tool for the description of DES)
  - it represents a possible implementation of the control logic (when used as programming language), that can be directly deployed on PLC-based hardware architectures
  - being a programming language itself, it can be easily translated in text-based program, using general purpose programming languages such as C, C++ or Java

SFC basics

The SFC is a bipartite graph, with two different type of nodes

- STEPS
- and TRANSITIONS

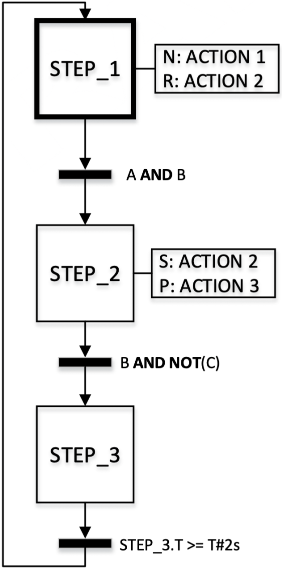
connected by oriented ARCS

STEP		TRANSITION	ARC
STEP_X	Initial step	A AND B OR NOT(C) Transition with the associated logical predicate	Oriented arc
STEP_Y	Step		

SFC - Example

The **basic rules** to build an SFC graph are very simple

- between two steps there must always be one (and only one)
- between two transitions there must always be one (and only one) step



- **A step can be active or non-active**
- The initial step (i.e., the step that is initially active) is represented using a ticker border
- **Given a SFC graph more than one step can be simultaneously active**
  - this is the main difference between finite-state machines (automata) and SFCs
  - this feature allows to easily represent concurrent actions
- **A logical predicate is associated to each transition** (i.e., a logical function that must return `TRUE` or `FALSE`)
- **Two implicit variables are defined for each step in a graph** (and can be used in the logical predicate of the transitions)
  - `NAME_STEP.X`, which is a `BOOL` variable that indicates if a step in active (`==TRUE`) or non-active (`==FALSE`)
  - `NAME_STEP.T`, which is a `TIME` variable that indicates the active time of a step (i.e., how long a step has been active)

SFC actions

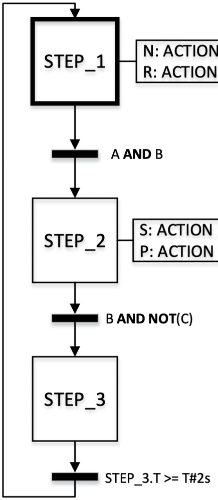
- One or more ACTIONS can be associated to each step
- Different qualifiers can be associated to each action

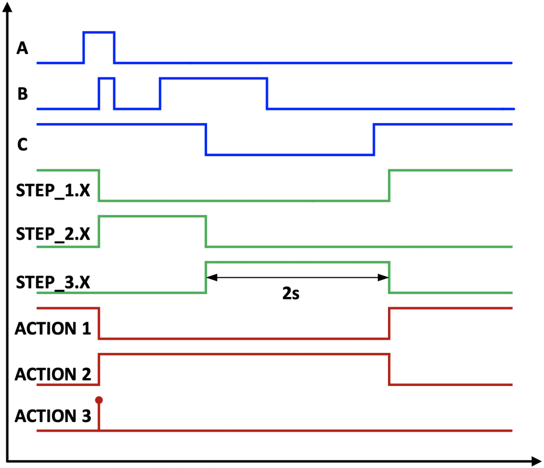
Qualifier	Type of Action	Description
N	Non-stored	The action active as long as the step.
R	overriding Reset	The action is deactivated.
S	Set (Stored)	The action is activated and remains active until a Reset.
L	time Limited	The action is activated for a certain time.
D	time Delayed	The action becomes active after a certain time as long as the step is still active.
P	Pulse	The action is executed just one time if the step is active.
SD	Stored and time Delayed	The action is activated after a certain time and remains active until a Reset.
DS	Delayed and Stored	The action is activated after a certain time as long as the step is still active and remains active up to a Reset.
SL	Stored and time Limited	The action is activated for a certain time.

SFC evolution rules

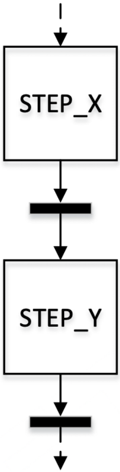
- **The SFC state is the set of active steps**
- The SFC state evolves according to the value of the logical predicates associated to the transitions
- The evolution rules of an SFC graph are the following
  - **A transition is ENABLED if all the upstream steps are active**
  - **An ENABLED transition fires if the associated logical predicate is TRUE**
  - **When e transition fires, it deactivates all the upstream steps and it activate all the downstream steps**

Example of SFC evolution

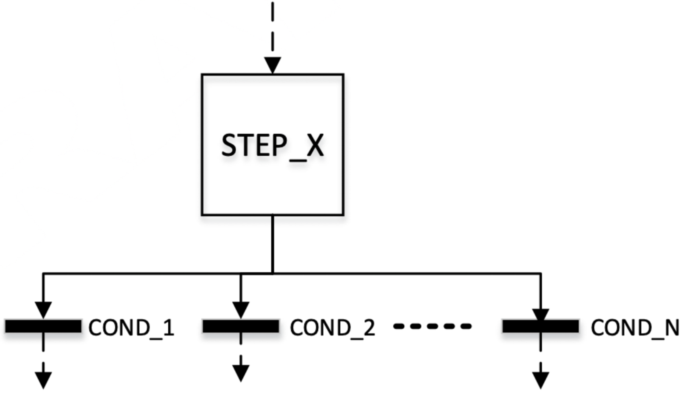




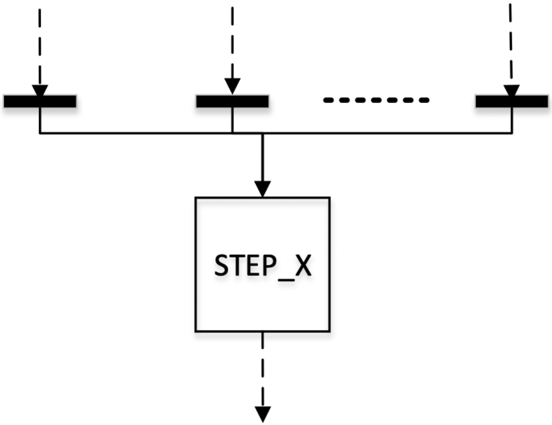
Basic programming structure
Sequence



Choice

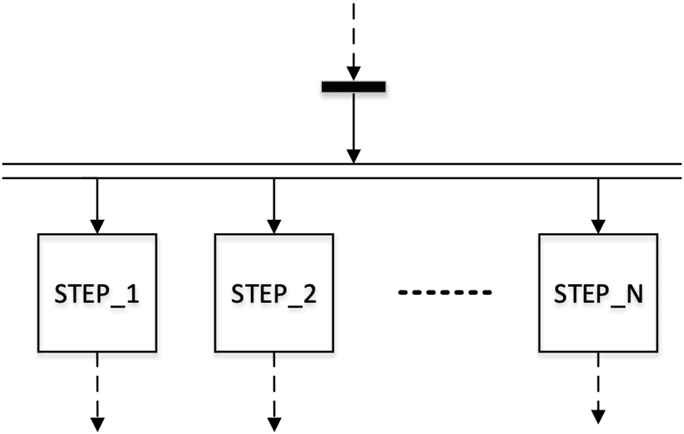


For the choice structure the logical predicates associated to the transitions should be mutually exclusive, in order to avoid ambiguous execution of the SFC

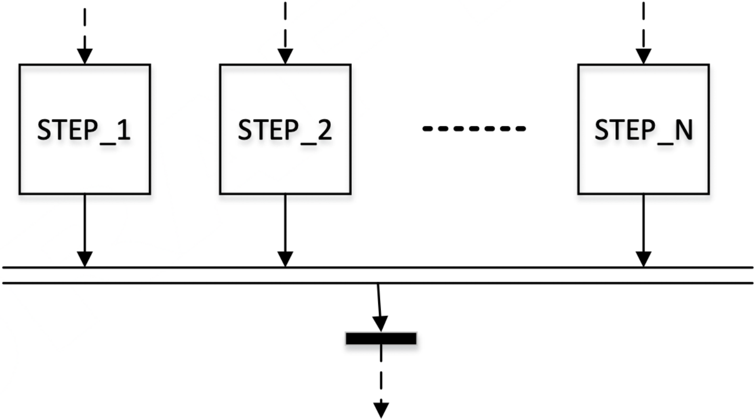


STEP_X

Basic programming structure
Parallelism



Basic programming structure Synchronizartion



Leonardo Labs Aerotech Campus / ACADEMY

PLCOpen Software Construction Guidelines

- PLCopen is an independent worldwide organization providing efficiency in industrial automation based on the needs of users
  - https://plcopen.org/
- Members include suppliers and educational institutes
- Focuses on harmonization of control programming, and application and interfacing engineering.
- Download the PLCopen Software Construction Guidelines

Examples - Aircrafts automation logics

4.3.1 CRANK command

Function ID:	POW#1
Function Name:	CRANK command

Short Description:

This function includes a single automation logic that generates the CRANK command on the basis of the pilot request. The crank is reset if the engine starts (i.e. RPM > RPM_THR) or if a watchdog timer expires (the watchdog timer is implemented using a second SFC graph).

Inputs

Tag	**Name**	**Description**	**Source**	**Type**
IDGND#008	PLT_CRANK	Digital request to trigger the CRANK **ASSUMPTION: the request is impulsive (equivalently the rising edge of the signal will be processed)**	Ground Station	BOOL
IAECU#002	RPM_MEAS	Analog measure of the engine rpm received from the ECU	ECU	REAL

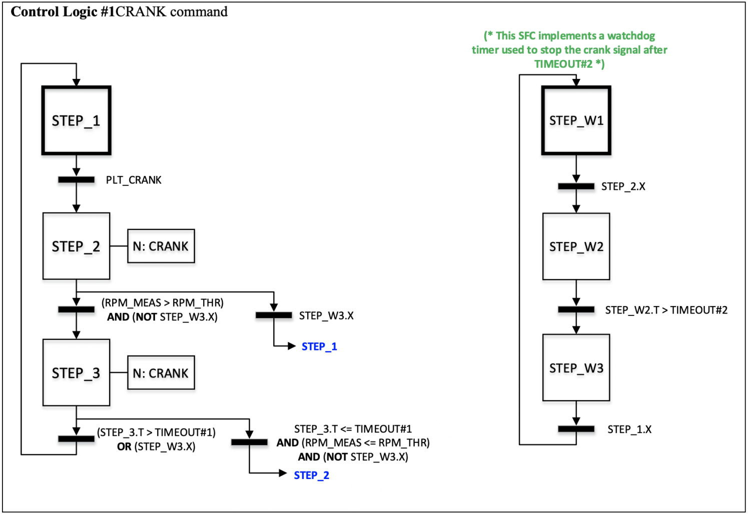
Outputs

Tag	**Name**	**Description**	**Consumer(s)**	**Type**
ODPOW#001	CRANK	Digital output that triggers the CRANK	STARTER	BOOL

Parameters

Name	**Description**	**Type**	**Default Value**	**Valid Range**
TIMEOUT#1	Timeout #1 of Control Logic #1	TIME	TBD	N/A
TIMEOUT#2	Timeout #2 of Control Logic #1 (watchdog timer)	TIME	5 s	N/A
RPM_THR	RPM threshold to set the propeller pitch to the feathering position	REAL	TBD	N/A

Examples - Aircrafts automation logics



4.2.2 Command to the AUX PUMP

Function ID:	FUL#2
Function Name:	Command to the AUX pump

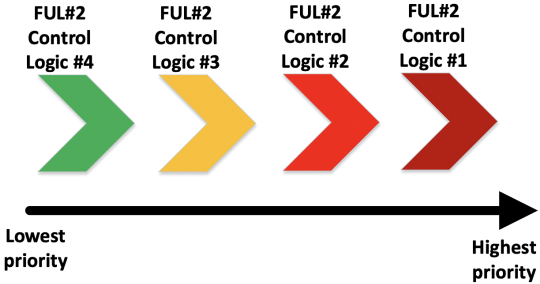
Short Description:

This function includes the following four automation logics:

- **Control Logic #1** processes the shut-off request received by the pilot in order to turn off the AUX pump.
- **Control Logic #2** processes the switch-on and switch-off commands sent by the pilot through the communication link in order to turn on and off the AUX pump.
- **Control Logic #3**processes the measurement from the pressure sensor of the fuel system, in order to turn on the AUX pump if there is a loss of flow in the engine rail
- **Control Logic #4** processes the aircraft altitude and speed (received by the navigation system), in order to turn on the AUX pump during both the takeoff and descent phases

Examples - Aircrafts automation logics

The priority order of the four control logics is reported below:

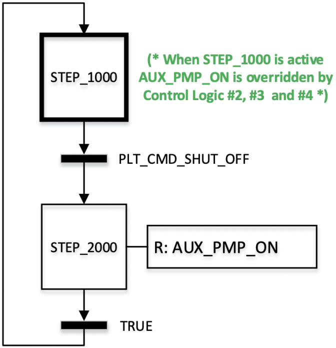


<u>Note that, the execution order of the control logics within each execution cycle must be inverse with respect to their priorities;</u> hence:

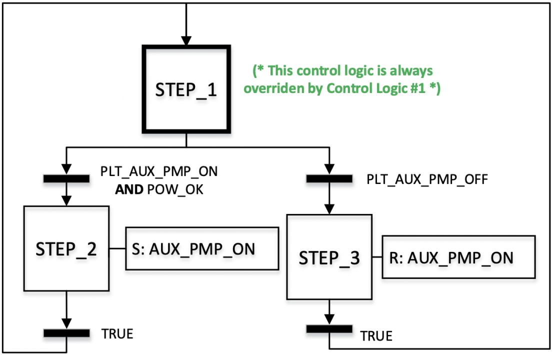
- **Control Logic #1 always overrides the other three control logics**
- **Control Logic #2 overrides Control Logic #3 and #4**
- **Control Logic #3 overrides Control Logic #4**

Leonardo Labs Aerotech Campus / ACADEMY

Control Logic #1 Command to the AUX pump – AUX Pump OFF Pilot input

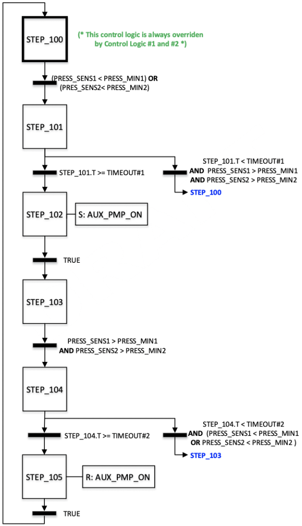


Control Logic #2: Command to the AUX pump – AUX Pump OFF/ON Pilot Input



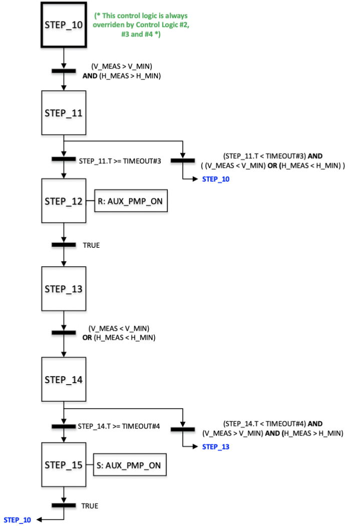
Examples - Aircrafts automation logics

Control Logic #3: Command to the AUX pump – AUX Pump ON Pressure Sensor Input



Examples - Aircrafts automation logics

Control Logic #4: Command to the AUX pump – AUX Pump ON AC Speed and Altitude input



HW and SW technologies for industrial automation

Leonardo Labs
IEC 61131-3 standard - IEC 61131 programming languages - Sequential functional chart

Gianmaria De Tommasi
Email: detommas@unina.it

October 2020

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE