Programming languages for PLC: International Standard IEC61131-3 (part one)

Cesare Fantuzzi (cesare.fantuzzi@unimore.it)

Ingegneria Meccatronica
AA 2012/2013
IEC Standards

- The **International Electrotechnical Commission (IEC)** is a not-for-profit, non-governmental international standards organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

- The IEC held its inaugural meeting on 26 June 1906, following discussions between the British **IEE**, the American **Institute of Electrical and Electronics Engineers (IEEE)** (then called AIEE), and others.

- It currently counts more than 130 countries.

- Originally located in London, the commission moved to its current headquarters in Geneva in 1948.
IEC 61131 is an IEC standard for Programmable logic controllers (PLCs). It was known as IEC 1131 before the change in numbering system by IEC.

- 1 General overview, definitions
- 2 Hardware
- 3 Programming Languages
- 4 User Guidelines
- 5 Messaging Service Specification
- 7 Fuzzy Logic
- 8 Implementation guidelines

C. Fantuzzi       International Standard IEC61131-3       3
Key quality features of IEC 61131-3

- Structured software - through use of Configuration, Resource and Program Organization Units (POUs), that implements software modularization.

- Data Typing – languages restrict operations to only apply to appropriate types of data

- Execution control - use of tasks

- Discrete event system handling - Sequential Function Charts

- Software encapsulation - through use of POUs, structures and complex data types.

C. Fantuzzi International Standard IEC61131-3
History of 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

Source: Dr. J. Christensen
it took 20 years to make that standard...

C. Fantuzzi
International Standard
IEC61131-3
Importance of the IEC61131

- IEC 61131-3 is the most important automation language in industry.
- 80% of all PLCs support it, all new developments base on it.
- Depending on the PLC vendor (and often Country because cultural reasons), some languages are most used.
Application Configuration (Hardware)
Application configuration
(Software)
A Task is a run time (scan) instance of a program.
IEC 61131 – Software Model
(Variable exchange)
Communication model (a)

a) Data flow connection within a program

b) Communication via *GLOBAL* variables

C. Fantuzzi 3. I Linguaggi di Programmazione 11
Communication model (b)

c) Communication function blocks

d) Communication via access paths
IEC Programming Model

- Data types
- Variables
- Program organization units (POU)
  - Functions
  - Function blocks
  - Programs
- Sequential Function Chart (SFC) elements
- Configuration elements
  - Global variables
  - Resources
  - Access paths
  - Tasks

Programming languages

(Programming language)
The five IEC 61131-3 Programming languages

**Graphical languages**

**Function Block Diagram (FBD)**

**Ladder Diagram (LD)**

**Sequential Flow Chart (SFC)**

**Textual languages**

**Structured Text (ST)**

```
VAR CONSTANT X : REAL := 53.8 ;
Z : REAL; END_VAR
VAR aFB, bFB : FB_type; END_VAR

bFB(A:=1, B:=’OK’);
Z := X - INT_TO_REAL (bFB.OUT1);
IF Z>57.0 THEN aFB(A:=0, B:=”ERR”);
ELSE  aFB(A:=1, B:=”Z is OK”);
END_IF
```
Structured Text (ST)

- ST is a textual language similar to “C”, or (for who might remember it) PASCAL.
- ST can be successfully used to develop complex algorithm, data structure handling, etc.
- ST has the syntactical structure of the procedural programming languages:
  - Assignment
  - Choices
  - Iteration
Assignment

- The variable on the left side should be of the same type of the result of the expression of the right side.
- On contrary, the ST compiler will introduce a *variable casting* to set all the variables to the same type.

<variable> := <expression>;
Choices

IF <Boolean_Expr_1> THEN
    <code>;
ELSIF <Boolean_Expr_2> THEN
    <code>;
ELSE
    <code>;
END_IF

CASE <integer_expression> OF
    <integer_value_1> : <code>;
    <integer_value_2> : <code>;
...
ELSE
    <code>;
END_CASE
Iteration

REPEAT
  <code>;
UNTIL <Boolean_Expr>
END_REPEAT;

WHILE <Boolean_Expr> DO
  <code>
END_WHILE;

FOR <integer_Variable>:= <initial_value> TO <final_value> BY <step> DO
  <code>
END_FOR;

C. Fantuzzi  International Standard IEC61131-3
Warning

- The iteration structure may violate the real time concerns of the program.
- An iteration can’t be done to wait for a external variable changes.
<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol</th>
<th>Example</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parenthesization</td>
<td>(expression)</td>
<td>(A+B/C), (A+B)/C, A / (B+C)</td>
<td>10</td>
</tr>
<tr>
<td>Negation</td>
<td>-</td>
<td>-A, - A</td>
<td>8</td>
</tr>
<tr>
<td>Unary Plus</td>
<td>+</td>
<td>+B, + B</td>
<td>8</td>
</tr>
<tr>
<td>Complement</td>
<td>NOT</td>
<td>NOT C</td>
<td>8</td>
</tr>
<tr>
<td>Exponentiation</td>
<td>**</td>
<td>A**B, B ** B</td>
<td>7</td>
</tr>
<tr>
<td>Multiply</td>
<td>*</td>
<td>A*B, A * B</td>
<td>6</td>
</tr>
<tr>
<td>Divide</td>
<td>/</td>
<td>A/B, A / B / D</td>
<td>6</td>
</tr>
<tr>
<td>Modulo</td>
<td>MOD</td>
<td>A MOD B</td>
<td>6</td>
</tr>
</tbody>
</table>
## ST Operators (b)

<table>
<thead>
<tr>
<th>Operation(^a)</th>
<th>Symbol</th>
<th>Example</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>+</td>
<td>A+B, A + B + C</td>
<td>5</td>
</tr>
<tr>
<td>Subtract</td>
<td>-</td>
<td>A-B, A - B - C</td>
<td>5</td>
</tr>
<tr>
<td>Comparison</td>
<td>&lt; , &gt; , &lt;= , &gt;=</td>
<td>A&lt;B</td>
<td>4</td>
</tr>
<tr>
<td>Equality</td>
<td>=</td>
<td>A=B, A=B &amp; B=C</td>
<td>4</td>
</tr>
<tr>
<td>Inequality</td>
<td>&lt;&gt;</td>
<td>A&lt;&gt;B, A &lt;&gt; B</td>
<td>4</td>
</tr>
<tr>
<td>Boolean AND</td>
<td>AND</td>
<td>A AND B</td>
<td>3</td>
</tr>
<tr>
<td>Boolean Exclusive OR</td>
<td>XOR</td>
<td>A XOR B</td>
<td>2</td>
</tr>
<tr>
<td>Boolean OR</td>
<td>OR</td>
<td>A OR B</td>
<td>1 (LOWEST)</td>
</tr>
</tbody>
</table>

\(^a\) Note: ST stands for Structured Text.
Functional Block Language

- Funktionsblocksprache, Langage de blocs de fonction.
- Also called "Function Chart" or "Function Plan" – FuPla
- The function block languages express "combinatorial" programs in a way similar to electronic circuits.
- They draw on a large variety of predefined and custom functions
Function blocks is a graphical programming language, which is akin to the electrical and block diagrams of the analog and digital technique.

It mostly expresses combinatorial logic, but its blocks may have a memory (e.g. flip-flops).
Function Block Elements

The block is defined by its:
- Data flow interface (number and type of input/output signals)
- Black-Box-Behavior (functional semantic, e.g. in textual form).

Signals

Connections that carry a pseudo-continuous data flow.
Connects the function blocks.

Example

PID

(set point)

(set point)

C. Fantuzzi  International Standard IEC61131-3
Function Block Example

C. Fantuzzi  International Standard IEC61131-3  25
Rule of evaluation

- Each signal is connected to exactly one source
  - This source can be the output of a function block or a plant signal.
- The type of the output pin, the type of the input pin and the signal type.
- The signals flow from left to right and from top to bottom.
- Feedback is an exception: the signal direction is from right to the left, and it is identified by an arrow.
Ladder Diagram

- (Kontaktplansprache, langage à contacts)
- The ladder logic is the oldest programming language for PLC
- it bases directly on the relay intuition of the electricians.
- it is widely in use outside Europe.
- It is described here but not recommended for new projects.
Ladder Logic (1)

C. Fantuzzi  International Standard IEC61131-3  28
The contact plan or "ladder logic" language allows an easy transition from the traditional relay logic diagrams to the programming of binary functions.

It is well suited to express combinational logic.

It is not suited for process control programming (there are no analog elements).

The main ladder logic symbols represent the elements:

- **make contact**
- **break contact**
- **relay coil**

C. Fantuzzi  International Standard IEC61131-3
Binary combinations are expressed by series and parallel relay contact:

**Series**

- **Ladder logic representation**: ![Series Ladder Logic](image)
- **“CMOS” equivalent**: ![Series CMOS Equivalent](image)

Coil 50 is active (current flows) when 01 is active and 02 is not.

**Parallel**

- **Ladder logic representation**: ![Parallel Ladder Logic](image)
- **“CMOS” equivalent**: ![Parallel CMOS Equivalent](image)

Coil 40 is active (current flows) when 01 is active or 02 is not.

C. Fantuzzi  
International Standard  
IEC61131-3
The ladder logic is more intuitive for complex binary expressions than literal languages textual expressions.
Ladder logic stems from the time of the relay technology.

As PLCs replaced relays, their new possibilities could not be expressed any more in relay terms.

The contact plan language was extended to express functions:

```
!00 & 01 FUN 02 = 200
```

The intuition of contacts and coil gets lost.

The introduction of «functions» that influence the control flow itself, is problematic.

The contact plan is - mathematically - a functional representation.

The introduction of a more or less hidden control of the flow destroys the freedom of side effects and makes programs difficult to read.

C. Fantuzzi  International Standard IEC61131-3 32