

IEC 61131-3: a standard programming resource

IEC 61131-3 is the first real endeavor to standardize programming languages for industrial automation. With its worldwide support, it is independent of any single company.

IEC 61131-3 is the third part of the IEC 61131 family. This consists of:

- Part 1: General Overview
- Part 2: Hardware
- Part 3: Programming Languages
- Part 4: User Guidelines
- Part 5: Communication

There are many ways to look at part 3 of this standard. Just to name a few:

- the result of the Task Force 3, Programming Languages, within IEC TC65 SC65B
- the result of hard work by 7 international companies adding tens of years of experience in the field of industrial automation
- approx. 200 pages of text, with 60-something tables, including features tables
- the specification of the syntax and semantics of a unified suite of programming languages, including the overall software model and a structuring language.

Another elegant view is by splitting the standard in two parts (see figure 1):

1. Common Elements
2. Programming Languages



Let's look more in detail to these parts:

Common Elements

Data Typing

Within the common elements, the data types are defined. Data typing prevents errors in an early stage. It is used to define the type of any parameter used. This avoids for instance dividing a Date by an Integer. Common datatypes are Boolean, Integer, Real and Byte and Word, but also Date, Time_of_Day and String. Based on these, one can define own personal data types, known as derived data types. In this way one can define an analog input channel as data type, and re-use this over and over again.

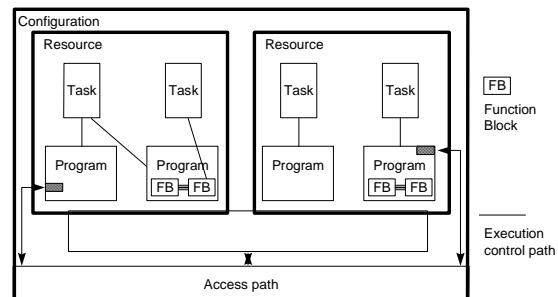
Variables

Variables are only assigned to explicit hardware addresses (e.g. input and outputs) in configurations, resources or programs. In this way a high level of hardware independency is created, supporting the reusability of the software.

The scopes of the variables are normally limited to the organization unit in which they are declared, e.g. local. This means that their names can be reused in other parts without any conflict, eliminating another source of errors, e.g. the scratchpad. If the variables should have global scope, they have to be declared as such (VAR_GLOBAL). Parameters can be assigned an initial value at start up and cold restart, in order to have the right setting.

Configuration, Resources and Tasks

To understand these better, let us look at the software model, as defined in the standard (see below).



At the highest level, the entire software required to solve a particular control problem can be formulated as a *Configuration*. A configuration is specific to a particular type of control system, including the arrangement of the hardware, i.c. processing resources, memory addresses for I/O channels and system capabilities.

Within a configuration one can define one or more *Resources*. One can look at a resource as a processing facility that is able to execute IEC programs. Within a resource, one or more *Tasks* can be defined. Tasks control the execution of a set of programs and/or function blocks. These can either be executed periodically or upon the occurrence of a specified trigger, such as the change of a variable. *Programs* are built from a number of different software elements written in any of the IEC defined languages. Typically, a program consists of a network of *Functions* and *Function Blocks*, which are able to exchange data. Function and Function Blocks are the basic building blocks, containing a datastructure and an algorithm.

Let's compare this to a conventional PLC: this contains one resource, running one task, controlling one program, running in a closed loop. IEC 61131-3 adds much to this, making it open to the future. A future that includes multi-processing and event driven programs. And this future is not so far: just look at distributed systems or real-time control systems. IEC 61131-3 is suitable for a broad range of applications, without having to learn additional programming languages.

Program Organization Units

Within IEC 61131-3, the Programs, Function Blocks and Functions are called Program Organization Units, POU's.

Functions

IEC has defined standard functions and user defined functions. Standard functions are for instance ADD(ition), ABS (absolute), SQRT, SINus and COSinus. User defined functions, once defined, can be used over and over again.

Function Blocks, FBs

Function Blocks are the equivalent to Integrated Circuits, ICs, representing a specialized control function. They contain data as well as the algorithm, so they can keep track of the past (which is one of the differences w.r.t. Functions). They have a well-defined interface and hidden internals, like an IC or black box. In this way they give a clear separation between different levels of programmers, or maintenance people.

A temperature control loop, or PID, is an excellent example of a Function Block. Once defined, it can be used over and over again, in the same program, different programs, or even different projects. This makes them highly re-usable.

Function Blocks can be written in any of the IEC languages, and in most cases even in "C". It this way they can be defined by the user. Derived Function Blocks are based on the standard defined FBs, but also completely new, customized FBs are possible within the standard: it just provides the framework.

The interfaces of functions and function blocks are described in the same way:

```

FUNCTION_BLOCK Example
VAR_INPUT:
  X : BOOL;
  Y : BOOL;
END_VAR

VAR_OUTPUT
  Z : BOOL;
END_VAR

(* statements of functionblock body *)

END_FUNCTION_BLOCK

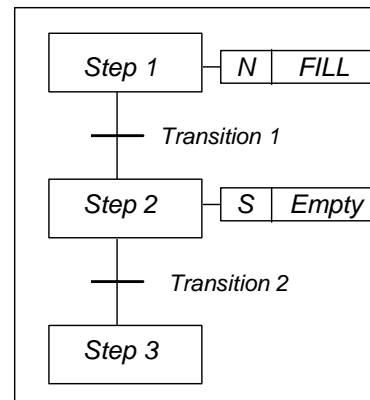
```

The declarations above describe the interface to a function block with two Boolean input parameters and one Boolean output parameter.

Programs

With the above-mentioned basic building blocks, one can say that a program is a network of Functions and Function Blocks. A program can be written in any of the defined programming languages.

Sequential Function Chart, SFC



SFC describes graphically the sequential behavior of a control program. It is derived from Petri Nets and IEC 848 Grafset, with the changes necessary to convert the representation from a documentation standard to a set of execution control elements.

SFC structures the internal organization of a program, and helps to decompose a control problem into manageable parts, while maintaining the overview. SFC consists of Steps, linked with Action Blocks and Transitions. Each step represents a particular state of the systems being controlled. A transition is associated with a condition, which, when true, causes the step before the transition to be deactivated, and the next step to be activated. Steps are linked to action blocks, performing a certain control action. Each element can be programmed in any of the IEC languages, including SFC itself.

One can use alternative sequences and even parallel sequences, such as commonly required in batch

applications. For instance, one sequence is used for the primary process, and the second for monitoring the overall operating constraints.

Because of its general structure, SFC provides also a communication tool, combining people of different backgrounds, departments or countries.

Programming Languages

Within the standard four programming languages are defined. This means that their syntax and semantics have been defined, leaving no room for dialects. Once you have learned them, you can use a wide variety of systems based on this standard.

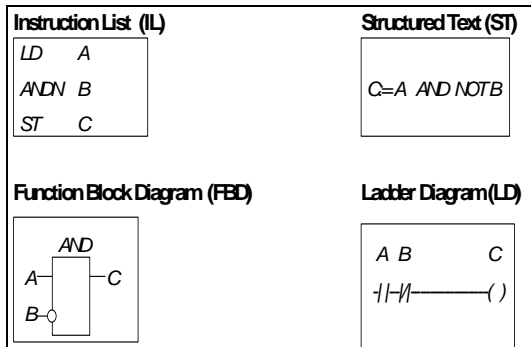
The languages consist of two textual and two graphical versions:

Textual:

- Instruction List, IL
- Structured Text, ST

Graphical:

- Ladder Diagram, LD
- Function Block Diagram, FBD



In the above figure, all four languages describe the same simple program part.

The choice of programming language is dependent on:

- the programmers' background
- the problem at hand
- the level of describing the problem
- the structure of the control system
- the interface to other people / departments

All four languages are interlinked: they provide a common suite, with a link to existing experience. In this way they also provide a communication tool, combining people of different backgrounds.

Ladder Diagram has its roots in the USA. It is based on the graphical presentation of Relay Ladder Logic.

Instruction List is its European counterpart. As textual language, it resembles assembler.

Function Block Diagram is very common to the process industry. It expresses the behavior of functions, function blocks and programs as a set of interconnected graphical blocks, like in electronic circuit diagrams. It looks at a system in terms of the flow of signals between processing elements.

Structured Text is a very powerful high-level language with its roots in Ada, Pascal and "C". It contains all the essential elements of a modern programming language, including selection branches (IF-THEN-ELSE and CASE OF) and iteration loops (FOR, WHILE and REPEAT). These elements can also be nested.

It can be used excellently for the definition of complex function blocks, which can be used within any of the other languages.

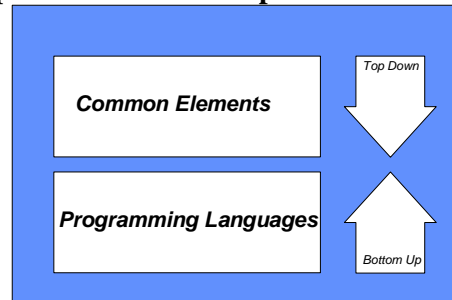
Example in ST:

```
I:=25;
WHILE J<5 DO
  Z:= F(I+J);
END_WHILE

IF B_1 THEN
  %QW100:= INT_TO_BCD(Display)
ENDIF

CASE TW OF
  1,5: TEMP := TEMP_1;
  2:   TEMP := 40;
  4:   TEMP := FTMP(TEMP_2);
ELSE
  TEMP := 0;
  B_ERROR :=1;
END_CASE
```

Top-down vs. bottom-up



Also, the standard allows two ways of developing your program: top down and bottom up. Either you specify your whole application and divide it into sub parts, declare your variables, and so on. Or you start programming your application at the bottom, for instance via derived functions and function blocks. Whichever you choose, the development environment will help you through the whole process.

Implementations

The overall requirements of IEC 61131-3 are not easy to fulfill. For that reason, the standard allows partial implementations in various aspects. This covers the number of supported languages, functions and function blocks. This leaves freedom at the supplier side, but a user should be well aware of it during his selection process. Also, a new release can have a dramatically higher level of implementation.

Many current IEC programming environments offer everything you expect from modern environments: mouse operation, pull down menus, graphical programming screens, support for multiple windows, built in hypertext functions, verification during design.

Please be aware that this is not specified within the standard itself: it is one of the parts where suppliers can differentiate.

Conclusion

The technical implications of the IEC 61131-3 standard are high, leaving enough room for growth and differentiation. This makes this standard suitable to evolve well into the next century.

IEC 61131-3 will have a great impact on the whole industrial control industry. It certainly will not restrict itself to the conventional PLC market. Nowadays, one sees it adopted in the motion control market, distributed systems and softlogic / PC based control systems, including SCADA packages. And the areas are still growing.

Having a standard over such a broad application area, brings numerous benefits for users / programmers. The benefits for adopting this standard are various, depending on the application areas. Just to name a few for the mindsetting:

- reduced waste of human resources, in training, debugging, maintenance and consultancy
- creating a focus to problem solving via a high level of software reusability
- reduced misunderstanding and errors
- programming techniques usable in a broad environment: general industrial control
- combining different components from different programs, projects, locations, companies and/or countries

PLCopen enhancements of the Standard

PLCopen is a vendor- and product-independent worldwide association supporting IEC 61131-3. By implementing this standard on many program development environments, users can move between different brands and types of control with very little training and exchange applications with a minimum of effort.

The organization PLCopen works on the promotion of the usage and/or supply of the standard, as well as to enhance the standard in a technical sense. This latter includes certification and exchange. The committees working within PLCopen and their results are described in short form hereunder.

Technical results

The Technical Committees, TCs, with representatives of PLCopen members, work on specific items.

Within **TC1 - Standards**, PLCopen collects proposals from its members for the IEC 65B WG7 working group, develops a joint position, and distributes related information. This was specifically focused to the second edition of the standard, which was released at the beginning of 2001.

TC2 - Functions defines common libraries of Function (Blocks) for specific application areas. An example is the library definition of Function Blocks for Motion Control. This standardization couples motion control in a logical way to the industrial control. As such, it provides a common look-and-feel towards the users: programmers as well as installation and maintenance people. With multiple implementations of this library, scaling of the control system is much easier, even if there are different architectures and / or controller brands used. The exchange of part of the programs via the PLCopen Reusability Level (see below for more information) plays an essential role here too.

Certification & Conformity testing

TC3 - Certification defines a certification system for PSEs, Program Support Environments (development environments). Each PSE can be tested to show that it complies with a PLCopen specified subset of the IEC 61131-3 standard. This standard contains a large number of so called feature tables and PLCopen has defined which elements of these tables must be supported to fulfill compliancy requirements. In addition, PLCopen extended the standard to support the reusability of user derived Function Blocks between PSEs.

Conformity Level, CL

With the broad range of application areas for IEC 61131-3 not all implementations use exactly the same data types. To accompany this, the certification according to Conformity Level, CL, implies that the supplier of a PSE selects the data types supported by his product matching his compliance statement. All supported IEC features are tested. This means that although the test is a Yes/No test (conformant or non-conformant), there can be differences between two certified products. These differences can influence the reusability of user derived function blocks.

The total number of data types as specified in the Standard amounts to 26 (table 10 and 12 of the Standard). These range from simple digital Yes/No (BOOL) to potential complex structures. Therefore, CL has 26 options: data type supported or not-supported. Only the supported data types are used for testing.

In addition **Reusability Level, RL**, is dedicated to making the programming units functions and function blocks reusable on a different PSE. This is done via exchange in a plain textual format of the language Structured Text, ST. For representation in other languages, a conversion tool to or from ST can be included. This is a major, but natural, contribution of PLCopen to the IEC 61131-3 standard.

Historically there exists a lower class called **Base Level**, defining a core kernel per language of the standard. Although rather restricted, it is feasible to develop applications based on it. Base Level provides an entrance level for the suppliers, showing their commitment to the standard. For the users it provides a normalized interpretation of the standard, especially important if they have to work with systems of different vendors.

Detailed specifications of most of the IEC 61131-3 languages are already finished. Work is in progress for the remaining languages. The compliance test procedure and the accreditation procedure for test laboratories have been established. Independent test laboratories have been accredited and an increasing number of products have been certified. For a complete list please refer to the website www.plcopen.org.

TC4 - Communications works on the relation between communication and programming languages, like the mapping of Profibus and CANopen via IEC 61131-5 onto IEC 61131-3.

TC5 - Safe Software prepares recommendations for applying the IEC 61131-3 standard to and

adapting it for safety related applications, esp. focused to the new safety standards IEC 61408 and 61411.

Promotional Events

An important tasks of PLCopen is to inform users / programmers about standardized industrial control programming, via:

- the PLCopen website: www.plcopen.org;
- publication of a free electronic and printed newsletter, called “PLCopening”;
- publications in the press;
- participation at trade shows and conferences;
- organization of conferences, like the ICP in October, and seminars.

The Promotional Committees **PC1, PC3 and PC4** support these activities.

PC2 - Common training has defined a common basis for training. This includes certification. In this way, certified training centers for training on IEC 61131-3 can be easily identified.

Benefits of Membership

A membership of PLCopen has many benefits for vendors, and institutes. PLCopen strongly supports the user community. For this it created additional membership categories.

By joining PLCopen, one makes a clear statement about your commitment to IEC 61131-3 including the tremendous benefits.

For on-going information, please check the website www.plcopen.org, as well as the electronic magazine to which you can subscribe at this website.