Data Sheet
2101127-AB

Overview of the IEC 61131 Standard

*X*Series Products
IEC 61131 General

INTRODUCTION

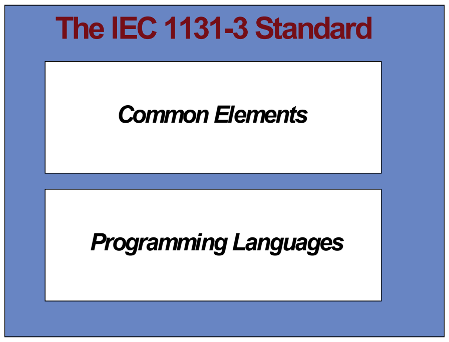
IEC 61131-3 is the first real endeavor to standardize programming languages for industrial automation. With its worldwide support, it is independent of any single company.

IEC 61131-3 standard is the result of a task force (IEC TC65 SC655B) comprised of seven or more international companies, representing several decades of experience in the field of industrial automation. The standard, 200 pages of text, with over 60 features tables, specifies the syntax and semantics of a unified suit of programming languages and a structuring language. It is organized as:

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

One way to view the standard is by splitting it into two parts:

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 data types are Boolean, Integer, Real, Byte and Word, but also Date, Time-of-Day and String. Based on these, one can define their own personal data types, known as derived data types. In this way one can define an analog input channel as a data type, and re-use this over an over again.

Variables

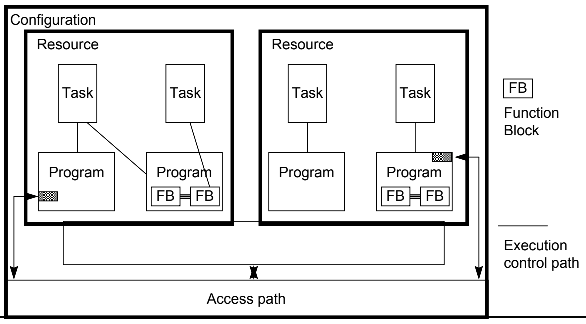
Variables are only assigned to explicit hardware addresses (e.g. input and outputs) in configurations, resources or programs. In this way a high level of hardware independency is created, supporting the reusability of the software.

The scope of the variables is 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 begin with the correct value.

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.e. 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 data structure 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 systems involving multi-processing and event driven programs, which are properties required in more complex distributed systems and 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, POUs.

Functions

IEC includes defined standard functions and supports user defined functions. Standard functions are for instance ADD (addition), ABS (absolute), SQRT, SIN and COS. User defined functions, once defined, can be used over and over again.

Function Blocks, FBs

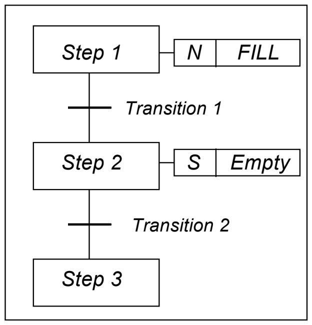
Function Blocks are the equivalent to Integrated Circuits, ICs, representing a specialized control function. They contain data as well as an algorithm. They have a well-defined interface and hidden internals, like an IC or black box. In this way they give a clear separation between different levels of programmers, or maintenance people. With these characteristics, Functions and Function Blocks reflect *best practices* as embraced by object-oriented principles.

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. Completely new, customized FBs are also possible within the standard.

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 graphically describes the sequential behavior of a control program. It is derived from Petri Nets and IEC 848 Grafcet, 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.

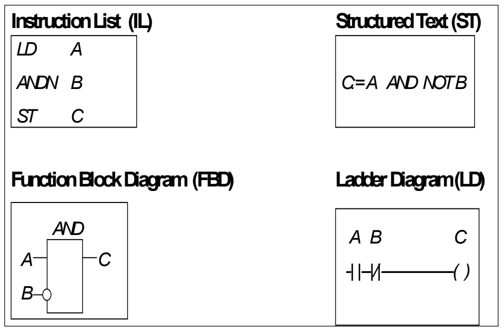
Because of its general structure, SFC also provides a communication and documentation tool, combining people of different backgrounds, departments or countries.

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.

There are four unique languages; two that are textual two that are graphical:

- 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
- How thoroughly the problem is specified
- 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.

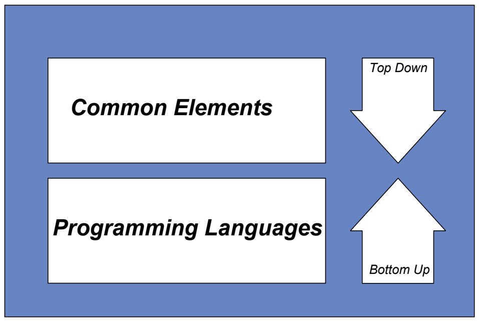
Instruction List is its European counterpart. As a textual language, it resembles assembler.

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 language with its roots in Ada, Pascal and "C". It is well suited for, and can be used to define the nature of complex *Function Blocks*, which can then be used within any of the other languages.

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.



Often complex projects are implemented using a combination of both approaches. Whatever you choose, the development environment will help you through the whole process.

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 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 and programmers:

- 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 form different programs, projects, locations, companies and/or countries.

APPLICATION TO ABB TOTALFLOW'S *X*SERIES TECHNOLOGY

As recognized above, IEC 61131-3 need not be restricted to conventional PLC markets. At ABB Totalflow, we believe the integration of IEC 61131-3 into technology such as ours represents a significant step forward.

With IEC 61131-3, our customers (whether end-users or integrators or OEM companies) are provided a globally recognized software environment that is well suited to many measurement and control applications.

Along with Totalflow's pre-built, industry focused applications, integration of the IEC 61131-3 into our *Renaissance Software Architecture* provides a powerful suite of tools, the combination of which is definitely greater the mere sum of the parts.

Power and Productivity
for a Better World. ™

www.abb.com/totalflow
www.abb.us
www.abb.com

ABB Inc.
Totalflow Products
7051 Industrial Blvd.
Bartlesville, OK 74006
Tel: (918) 338-4888
Fax: (918) 338-4699
(800) 442-3097

ABB Inc.
Totalflow Products
433 Northpark Central Dr., Ste. 100
Houston, TX 77073
Tel: (281) 869-5212
Fax: (281) 869-5203
(800) 442-3097

For more information, please contact your local ABB Totalflow representative or visit our website.