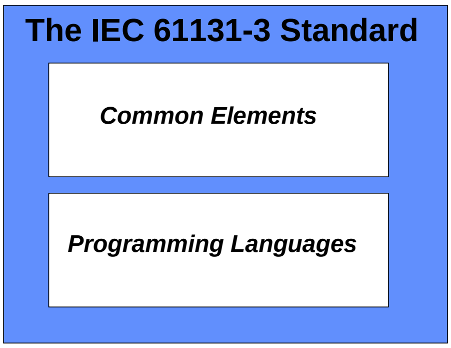
IEC 61131-3: a standard programming resource

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, and is a specification of the syntax and semantics of a unified suite of programming languages, including the overall software model and a structuring language.
It consists of:

- Part 1: General Overview
- Part 2 Equipment Requirements & Tests
- Part 3 Programming Languages
- Part 4 User Guidelines
- Part 5 Communications
- Part 6 Functional Safety
- Part 7 Fuzzy Control Programming
- Part 8 Application Guidelines
- Part 9 Communication Interface

Another elegant view is by splitting the standard in two parts (see figure 1):
1. Common Elements
2. Programming Languages

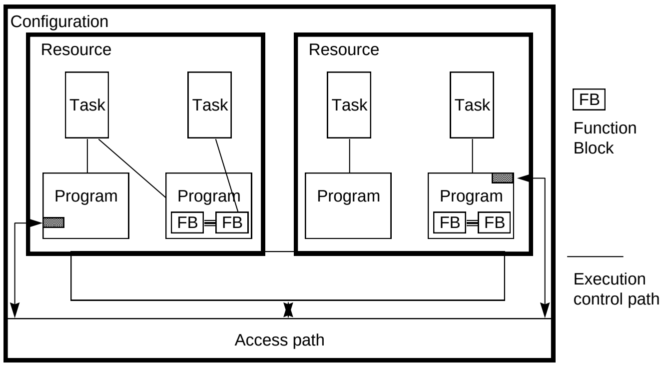


Common Elements

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 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 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. If the variables should have global scope, they have to be declared as such. 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 system required to solve a particular control problem can be formulated as a *Configuration*, including the arrangement of the hardware, 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* (like ADD(ition), SINus and COSinus) and *Function Blocks*, which are able to exchange data. Function and Function Blocks are the basic building blocks, containing a datastructure and an algorithm.
Function Blocks contain data as well as the algorithm, so they can keep track of the past. 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” or C++. In this way they can be defined by the user.

A conventional PLC 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 already includes multi-processing and event driven programs.
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 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.

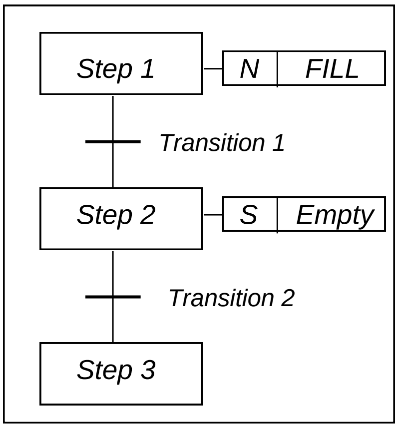
The interfaces of functions and function blocks are described in the same way:

```
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. With this it 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. 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 parallel sequences, such as commonly required in batch applications.
Because of its general structure, SFC provides also a communication tool, combining people of different backgrounds, departments or countries.

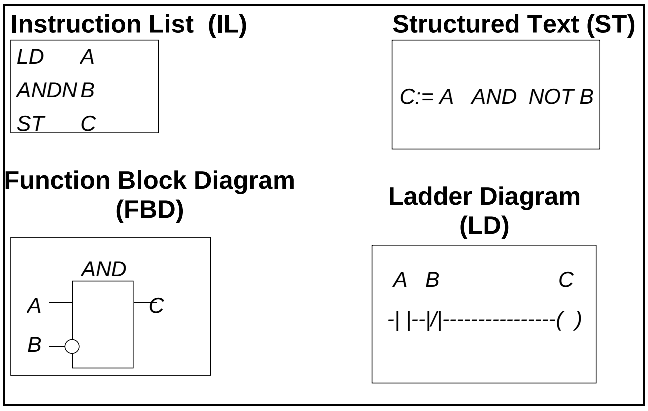
Programming Languages

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:

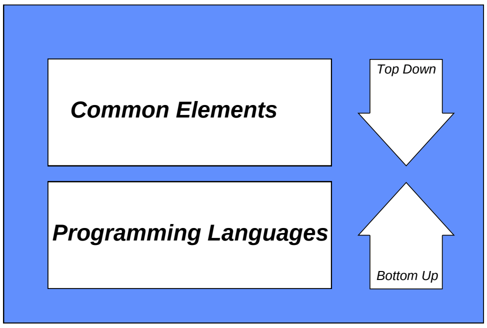
- the programmers' background
- the problem at hand
- the level of describing the problem
- the structure of the control system
- the interface to other people / departments

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. 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.

Program 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, and 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.

The 3rd edition and Object Oriented features

With the release of the 3rd edition in 2013, object oriented features were added to the specification. With this, classes, method, interfaces and namespaces became part of the specification, as well as namespaces, all linking it to a new generation of software programmers, as well as to other software tools.

Conclusion

The technical implications of the IEC 61131-3 standard are high, leaving enough room for growth and differentiation. IEC 61131-3 impacted the whole industrial control industry: not only the conventional PLC market, but also the motion control market, distributed systems and softlogic / PC based control systems, including PACs. And the areas are still growing, including safety, communication and exchange.
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 mindset:

- 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 creates efficiency in industrial automation through the harmonization and integration of logic, motion, and safety, combined with communication and exchange.
The created efficiency is in the development of the application software, the life-cycle cost of the equipment, the training of operating and maintenance personnel as well as of the software engineers.
As vendor- and product-independent worldwide association, PLCopen is depending on its income through its membership fees. For this, PLCopen supports a multi-level membership, ranging from suppliers to educational institutes. PLCopen strongly supports the user community. For this it created additional membership categories.

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, released in 2003, and now to the third edition, which was released in 2013.

TC2 - Functions defines common libraries of Function (Blocks) for specific application areas. Examples are the library definitions of Function Blocks for Motion Control. This standardization integrates motion control with 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, reusability of software and scaling of the control system is much easier, even across different architectures and / or controller brands.

Certification & Conformity testing
TC3 - Certification defines a certification system for IEC 61131-3 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. For this, several levels are defined with **Conformity Level, CL** as the highest level.
In addition **Reusability Level, RL**, is dedicated to making the programming units functions and function blocks reusable on a different PSE in the same programming language.
Historically there exists an entry level called **Base Level**, to show commitment to the standard.
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. A working item is the mapping of the IEC 61131-3 software model to the OPC UA information model, leading to a standard way how OPC UA server-based controllers expose data structures and function blocks to OPC UA clients like HMIs. This work resulted in the specification OPC UA Information Model for IEC 61131-3.
From the beginning it was clear that integrating the OPC UA server in the controller was only the first step. So the working groups initiated the next step by adding the OPC UA client functionality in the controller by defining a set of Function Blocks for IEC61131-3. This specification was released in 2014 with an update in 2016.

TC5 - Safe Software prepares recommendations for applying the IEC 61131-3 standard to safety related applications, esp. focused to machines, and the new safety standards IEC 61508 and 61511 as well as IEC 62061. In addition, it provides guidelines for the user to use safety aspects within their applications, supported by their development environments, as well as library definitions of Function Blocks for Safety applications.

TC6 - XML worked on the specification of XML schemes for all languages, as well as full projects. This specification provides the basis for exchange, as well as coupling to other software tools, including higher-level development tools, simulation tools, documentation tools, and verification tools. The PLCopen XML specification will become an integral part of the IEC 61131 specifications. The international standard IEC 61131-10 is expected to be published in 2018.

Promotional Events
An important task of PLCopen is to inform users / programmers about the benefits of standardized industrial control programming. This is done via:

- the PLCopen website: www.PLCopen.org
- publication of a free electronic newsletter
- publications in the press
- participation at conferences and shows

The Promotional Committees **PC1, PC3 PC4, PC5 and PC6** support these activities.

PC2 - Common training has defined a common basis for training. This includes certification. In this way, certified training centres for training on IEC 61131-3 can be easily identified.
Although there are guidelines for many programming languages, these are nearly non-existent for the important area of industrial control, e.g. IEC 61131-3 and its PLCopen extensions. Therefore PLCopen initiated the development of Software Construction Guidelines.

Benefits of Membership
A membership of PLCopen has many benefits for users, vendors, and institutes. By joining PLCopen, one makes a clear statement about commitment to open architectures. In addition, one can participate in the committees, and as such have upfront information as well as influence on the work done, and use the PLCopen logos.
The annual contribution depends on voting and non-voting rights, the number of relevant employees active, and the nature of the organization. The different types are:

- Companies: commercial organizations active in the IC
- Users: focused to the application and usage of IC
- System Integrators / Re-sellers: added value is not in the IC or its programming environment
- Non-profit organizations
- Educational institutes

For more information, please check www.PLCopen.org.