REAL TIME AUTOMATION

IEC 61131-3

The Fast Guide to Open Control Software

PREPARED BY REAL TIME AUTOMATION
WWW.RTAAUTOMATION.COM

IEC 61131-3

The Fast Guide to Open Control Software

Introduction

IEC 61131-3 is the first vendor-independent standardized programming language for industrial automation. The language was established by the International Electro-technical Commission (IEC), a worldwide standard organization founded in 1906 and recognized worldwide for standards in the controls industry by over 50 countries. The standard is already well established in Europe and is rapidly gaining popularity in North America and Asia as the programming standard for industrial and process control.

The adoption of IEC 61131-3 by the industry is driven by the increasing software complexity of control and automation requirements. The creation time, labor cost, and maintenance of control software has a major impact on control projects that can be improved using the IEC 61131-3 vendor-independent programming language standard. Applying a standard programming language has a positive impact on the software life-cycle that includes requirements analysis, design, construction, testing (validation), installation, operation, and maintenance. The impact on maintenance is important since control software maintenance, including upgrades, is generally two to four times the labor of initial programming.

The IEC 61131-3 standard, combined with powerful new Freescale chip architectures, enables an entire controller to be delivered in an embedded device. Control programs can run distributed and independently rather than concentrated in large controllers. No longer are thousands of lines of control programs required to run in one controller for complex automation applications. This increases performance, improves reliability, and simplifies programs.

IEC 61131-3 provides multiple language support within a control program. The control program developer can select the language that is best suited to a particular task, greatly increasing their productivity. Plus, with a standardized programming interface that is completely independent of the hardware platform, users can greatly reduce the cost of program maintenance and training across company wide automation applications.

The International Electro-technical Commission regulates the IEC 61131 standard.

IEC 61131-3 is hardware-independent. The ability to transport automation solutions to other platforms is vastly improved over PLC applications, offering users and System Integrators a level of reusability never before available. IEC 61131-3 also increases the efficiency and speed of implementing new automation solutions by using readily available control components developed on other projects and by outside developers.

Companies that have chosen to implement IEC 61131-3 find that they reduce human resource costs in training, debugging, and maintenance and improve productivity due to the higher reusability.

Technology Overview

IEC 61131-3 is the international standard for programmable controller programming languages. As such, it specifies the syntax, semantics, and display for the following suite of PLC programming languages:

- **Ladder diagram (LD)**
- **Sequential Function Charts (SFC)**
- **Function Block Diagram (FBD)**
- **Structured Text (ST)**
- **Instruction List (IL)**

IEC 61131-3 is the third component (Part 3) of the IEC 61131 family that consists of:

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

The easiest way to view the standard is to split it into two parts, Common Elements and Programming Languages.

Common Elements

The first portion of IEC 61131-3 is comprised of some common elements.

Data Typing

Data Typing is a common element of the standard that prevents errors early in development. It defines the type of parameters that will be used and attempts to avoid errors like dividing a Date by an Integer. The different type of data supported are Boolean, Integer, Real, Byte, Word, Date, Time-of-Day, and String. The Standard also allows users to define their own variables. These are known as derived data types. Using derived data types, an engineer would be able to define an analog input channel as a data type and re-use it repeatedly.

Variables

Variables are assigned only to explicit hardware addresses or explicit inputs and outputs. These can be assigned in custom configurations and programs. An IEC 61131 system is highly independent and able to function with little to no messaging from an external network.

The Scope of the variable is limited to the organization unit in which they are declared. The great benefit of this feature is that variable names can be reused in other parts without any conflict, eliminating another source of errors. If the variables have Global Scope they must be declared global. Parameters can be assigned for their initial values at start up and restart.

Configuration, Resources, and Tasks

At the highest level, the 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 and includes the arrangement of the hardware (processing resources, memory addresses for I/O channels, and system capabilities).

Within a configuration, one can define resources. A resource can be thought of 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 occurrence of a specified trigger.

For instance, in an IEC 61131 enabled drive, a trigger could be set when RPMs fall below a predefined value. The trigger could start a task to increase speed. These results are instant and come directly from the drive. There is no lag or hand-shaking by an external PLC. This means that there is virtually no risk of losing a message or miscommunication. Feed-back is nearly instantaneous compared to a Programmable Controller with an I/O and Program Scan time.

Programs are built from a number of different software elements written in any of the IEC defined languages; Ladder diagrams, Sequential Function Charts, Function Block Diagrams, Structured Text or Instruction List. It is typical for a pro-gram to consist of a series of high level function blocks written in one or more of these languages.

Program Organization Units

Within IEC 61131-3, the programs, function blocks, and functions are called program organization units, or POUs.

IEC 61131-3 includes defined standard function instances, including ADD, ABS, SQRT, SIN, and COS. The user can also create a custom function block and use that function block multiple times.

Function blocks are software objects that represent a level of more detailed control. They can contain data as well as an algorithm. As software objects, they have a well-defined interface and hidden internals. This creates a clear line between the different levels of the programs. With these characteristics, functions and function blocks reflect best practices as embraced by object-oriented programming principles.

Function blocks can be written in any of the IEC languages and, in most cases, even C using basic building blocks.

Sequential Function Charts, or SFCs, are used to control the sequential behavior of a control program and support syn-chronization and concurrency.

Programming Languages

In IEC 61131-3, five standard programming languages, including syntax and semantics, are defined, leaving no room for dialects. Once you have learned the five languages, you can use a wide variety of systems based on the standard.

The end user is able to choose a programming language based on their knowledge, the problem at hand, external com-ponents, interfaces, or simple preference. All languages are linked and provided a common suite, with a link to existing experience. In this way, they also provide a communication tool, combining people of different backgrounds. Because of the standards structure built on functions and function blocks, users are able to adopt either a top-down or bottom-up strategy to develop their programs.

CoDeSys (www.3s-software.com)

CoDeSys is one of the most powerful IEC 61131-3 programming tools for controllers. CoDeSys supports all five pro-gramming languages of the standard, combining the power of advanced programming languages, such as C or Pascal, with the easy handling and operational functions of PLC programming systems.

Unlike some competitive IEC 1131 offerings, CoDeSys produces native machine code for a large number of common processors. Native machine code is inherently faster and more reliable than interpreted solutions.

The entire programming kit, including a manual and online assistance, is available in German, English or French. Parts of the tool, like the online help, are available in other languages like Russian, Chinese or Spanish.

CoDeSys provides certain competitive advantages, including:

- **Fast Customization**

3S is able to perform a complete test adaptation (including online functionality) on any standard processor hardware within two days. CoDeSys has ready back-ends for all current processors. In order to keep customization time and resulting expenses to a minimum, the run time system, programming system, and code generation are perfectly coordinated, saving your time and ensuring your products reach the marketplace swiftly.

- **A Practical, Easy-to-Use Approach**

Functions like Autodeclare, Autoformat, and a context-sensitive input assistance greatly simplify the use of CoDeSys. All functions are accessible using the keyboard. The exceptionally low number of resources CoDeSys requires ensures fast and efficient work.

- **High Performance**

Native code generators for all common processors guarantee the optimal use of your control system. Due to intelligent algorithms such as 'incremental compile', large projects with thousands of global variables and hundreds of components can be realized in surprisingly short compiling times. CoDeSys supplies users with a broad range of high-performance program development functionalities, including almost all data types specified in the IEC 61131-3, offline simulation, and powerful online functions such as breakpoints, single stepping, power flow, sampling trace, and online change.

Freescale Coldfire Application Story

Real Time Automation, a Freescale DAP (Design Assistance Partner), has recently integrated CoDeSys 1131 open control software with Freescale Coldfire in an OEM controls application.

The application calls for real time control of up to ten control loops and monitoring of many discrete and analog I/O points, and it requires a low cost control solution with embedded I/O that could be installed in thousands of different locations quickly and easily. Installing and programming PLCs with rack or networked I/O was too costly for this application. Freescale Coldifre Technology, embedded I/O, and open IEC 61131 software provided a low cost, supportable, single-unit solution that more than met all the requirements of the application.

RTA deployed Freescale in this application based on the solid track record of 32-bit processors over the past eight years. The Freescale price/performance point, advanced communications support, processing power, integrated Ethernet, and comprehensive development tools were the right combination for this project.

For more information on using Freescale processors with embedded open, IEC 61131 control software visit www.rtaautomation.com.

Figure 1 — Freescale Coldfire Processor integrated with CoDeSys on a din rail mounted chassis with 16 Discrete Input, 16 Discrete Outputs and 16 Analog Inputs