MSD MOTION SYSTEM DESIGN

Save time with reusable code

Standardized programming for automation equipment is here to stay. At the forefront is IEC 61131-3-compatible software, which generates standardized code for use on any number of controls.

Kevin Hull
Senior applications engineer
Yaskawa Electric America Inc.
Waukegan, Ill.

Time is money, and for industrial automation software development, nothing saves more time than the ability to reuse programming code.

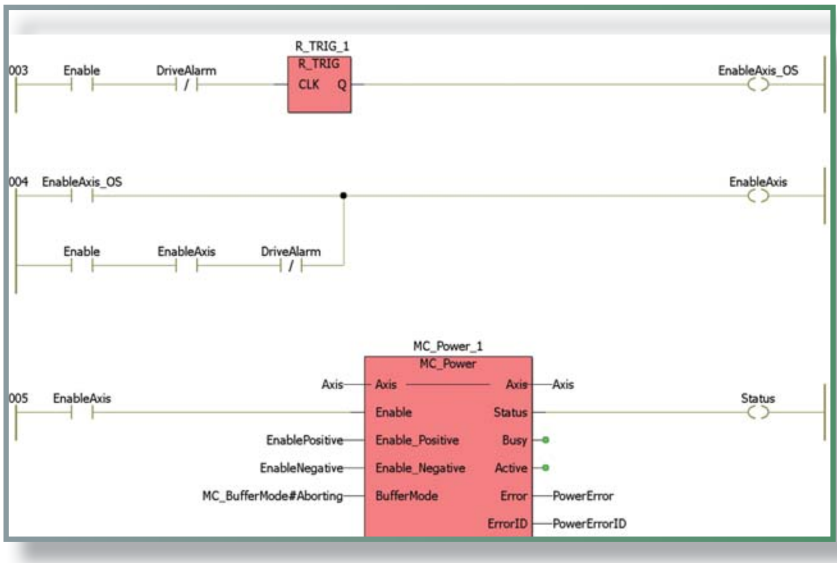
Until now, the norm in automation has been custom and proprietary software for operating vendor-specific hardware — primarily PLCs and motion controllers. But standardized and reusable programming is spreading, thanks to the IEC 61131-3 standard, particularly as refined by the international organization called PLCopen. IEC 61131-3 is a globally recognized standard for industrial automation control programming; PLCopen promotes the use of IEC 61131-3 and other standards via a number of initiatives including training and certification.

The dark days

Before IEC 61131-3 and PLCopen, each industrial automation vendor promoted its own closed and proprietary programming software. Customers invested significant time learning these special programs. Moreover, many programming packages specialized in only one language. For example, ladder logic was popular in North America, but other industries and regions preferred structured text or a special mnemonic language similar to an instruction list.

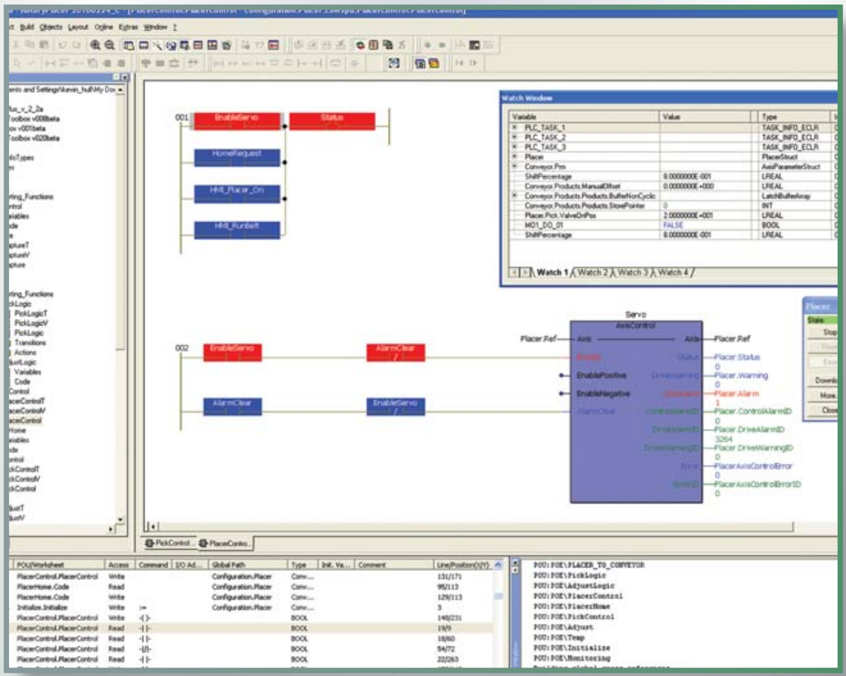
This presented two major problems. First, a new and unique programming software package had to be mastered each time a different controller brand was used. Second, it was very difficult to reuse code, even when programming different controllers from the same vendor.

Even now, proprietary code often drives maintenance costs higher because technical personnel must learn how to use very different software packages for each controller in a plant. What's more, proprietary code deters adoption of newer and more cost-effective platforms because designers shy away from the



IEC 61131-3 allows programming in five languages. The one shown here is ladder diagram — globally one of the most commonly used languages.

The International Electrotechnical Commission (IEC) was founded in 1906 and establishes electrical, electronic, and related codes. Its IEC 61131-3 is the first and only real attempt at creating a global standard for control programming in the industrial automation market.



MotionWorks IEC development software is based on and fully compliant with IEC 61131-3 — to make controls easier to setup and update.

initial costs associated with learning new programming.

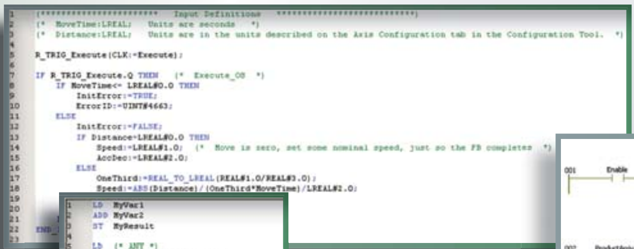
IEC to the rescue

IEC 61131-3-adherent controller-programming software is a standardized alternative. It includes a framework that suppliers can follow when creating automation software packages. The closer a supplier adheres to the standard, the more familiar its products are to engineers versed in IEC 61131-3. This is creating a virtuous circle as the standard becomes more widespread. How?

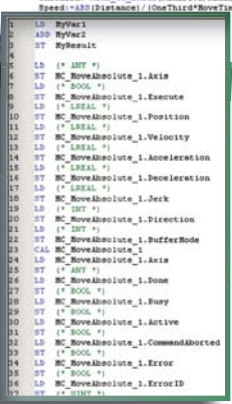
As more users become familiar with IEC 61131-3, additional vendors are compelled to jump on the standards bandwagon. As vendors join in, more users become comfortable with the standard. The result? Widespread adoption by suppliers and users alike.

IEC 61131-3 enables automation programmers to leverage the best of several different programming languages simultaneously within one software-development environment. It also standardizes the programming interface, reducing the learning curve for individuals with different programming backgrounds and skill levels.

For these reasons, programmers can create different elements of a program during software-development specification, design, implementation, testing, installation, and

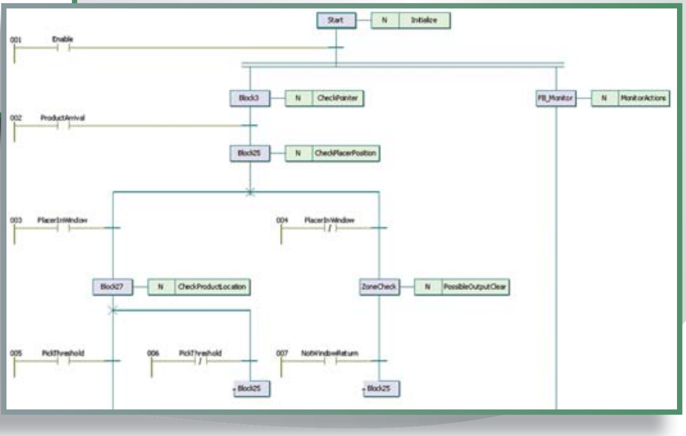


```
(***********************   Input Definitions   ****************************)
(*  MoveTime:LREAL;   Units are seconds   *)
(*  Distance:LREAL;   Units are in the units described on the Axis Configuration tab in the Configuration Tool.  *)

R_TRIG_Execute(CLK:=Execute);

IF R_TRIG_Execute.Q THEN   (*  Execute_OS  *)
    IF MoveTime<= LREAL#0.0 THEN
        InitError:=TRUE;
        ErrorID:=UINT#4663;
    ELSE
        InitError:=FALSE;
        IF Distance=LREAL#0.0 THEN
            Speed:=LREAL#1.0;  (*  Move is zero, set some nominal speed, just so the FB completes  *)
            AccDec:=LREAL#2.0;
        ELSE
            OneThird:=REAL_TO_LREAL(REAL#1.0/REAL#3.0);
            Speed:=ABS(Distance)/(OneThird*MoveTime)/LREAL#2.0;


END_
```



```
LD   MyVar1
ADD  MyVar2
ST   MyResult

LD   (* ANY *)
ST   MC_MoveAbsolute_1.Axis
LD   (* BOOL *)
ST   MC_MoveAbsolute_1.Execute
LD   (* LREAL *)
ST   MC_MoveAbsolute_1.Position
LD   (* LREAL *)
ST   MC_MoveAbsolute_1.Velocity
LD   (* LREAL *)
ST   MC_MoveAbsolute_1.Acceleration
LD   (* LREAL *)
ST   MC_MoveAbsolute_1.Deceleration
LD   (* LREAL *)
ST   MC_MoveAbsolute_1.Jerk
LD   (* INT *)
ST   MC_MoveAbsolute_1.Direction
LD   (* INT *)
ST   MC_MoveAbsolute_1.BufferMode
CAL  MC_MoveAbsolute_1
LD   MC_MoveAbsolute_1.Axis
ST   (* ANY *)
LD   MC_MoveAbsolute_1.Done
ST   (* BOOL *)
LD   MC_MoveAbsolute_1.Busy
ST   (* BOOL *)
LD   MC_MoveAbsolute_1.Active
ST   (* BOOL *)
LD   MC_MoveAbsolute_1.CommandAborted
ST   (* BOOL *)
LD   MC_MoveAbsolute_1.Error
ST   (* BOOL *)
LD   MC_MoveAbsolute_1.ErrorID
ST   (* UINT *)
```

IEC 61131-3 also allows programming in instruction list (lower left), structured text (upper left), and sequential function, below.



maintenance. The software pieces adhere to a common structure, so they work together harmoniously.

Under the IEC hood

The IEC 61131-3 standard has two different parts: **Common Elements** and **Programming Languages**. By using the structuring tools of both, IEC 61131-3 can be used to create software that's maintainable, reusable, understandable, and verifiable.

The Common Elements of IEC 61131-3 include *data typing*, *variables*, *configuration*, and *program organization units*. *Data typing* is the formal definition of parameters, and it's required to standardize data and prevent errors. IEC 61131-3's data types include elementary, generic,

The standards champion: PLCopen

Any standard is worthless unless adopted by the community it serves. Fortunately, PLCopen is making sure that the IEC 61131-3 standard is used widely and correctly.

PLCopen members consist of global automation suppliers and learning institutions that aim to reduce variation in industrial controller programming techniques. PLCopen is completely separate from IEC — allowing it to independently disseminate and promote IEC 61131-3 and other standards throughout the global automation community.

When a standard is created, suppliers often claim adherence or conformity without any independent third party verification. This leaves the user to determine if the product conforms — often a difficult if not impossible task. To address this issue, PLCopen independently tests programming software. Vendors submit products to PLCopen, and the organization verifies compatibility. Once verified, the vendor is permitted to label its software with the PLCopen logo and promote the tested level of verification.

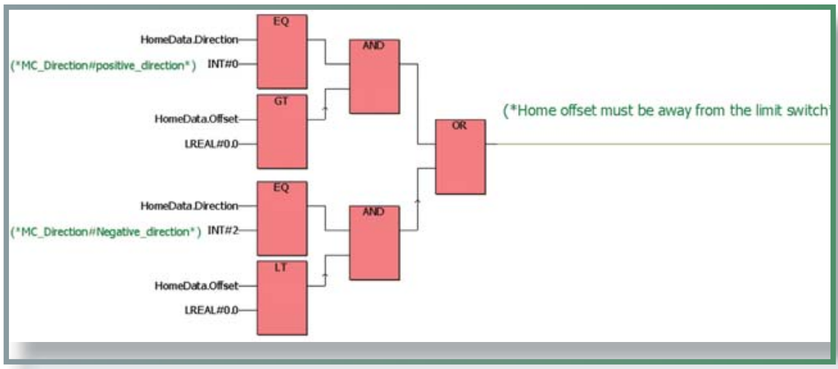
PLCopen also provides a resource for plain-language information about IEC 61131-3 and other automation standards at www.plcopen.org. The site outlines standards from the user's point of view and lists answers to FAQs, to familiarize users with the standard's main concepts.

PLCopen also extends the IEC 61131-3 standard to encompass certain specialty areas of industrial automation: For example, the PLCopen Motion Control Specification provides a standard for motion control behavior and functionality. Motion control function blocks created with conformance to this standard have the same look and feel, and can therefore be reused and integrated into a variety of applications with little or no modification.

Together, IEC 61131-3 and PLCopen increase commonality from vendor to vendor — but allow each vendor to customize performance and capabilities.

and user-defined. Its three kinds of *variables* include symbolic, directly represented, and located. Located refers to a specific hardware address.

Configuration aims to solve the problem of hardware arrangement, memory addressing, and processing resources. Within a configuration or system, various resources can execute IEC programs. Inside the resources are tasks consisting of control programs and function blocks.


EQ
HomeData.Direction
(*MC_Direction#positive_direction*) INT#0
GT
HomeData.Offset
LREAL#0.0
AND
OR
(*Home offset must be away from the limit switch
EQ
HomeData.Direction
(*MC_Direction#Negative_direction*) INT#2
AND
LT
HomeData.Offset
LREAL#0.0

IEC 61131-3 also allows programming in function block diagrams.

IEC 61131-3 common elements

- *Data typing*
- *Variables*
- *Configuration*
- *Program organization units*

Program organization units serve as the container for program code. These units consist of functions, function blocks, and programs. Functions let program elements extend a configuration's instruction set. Function blocks are the basic units from which applications are built. Programs are networks of functions and function blocks.

IEC 61131-3 is multilingual

As befits a global standard, IEC 61131-3 specifies (and speaks) five programming languages, which can all be used within the same program as dictated by the application and programmer's preference. Ladder diagram (LD) and function block (FB) are graphical programming languages, while structured text (ST) and instruction list (IL) are textual. Sequential function chart (SFC) is a graphical tool that organizes programs for sequential and parallel control.

❶ **Function block diagrams** are suitable for motion control and other complex tasks as they visually display high-level interactions. Variables are connected to the blocks by lines that carry information from left to right. The blocks (predefined or user-created) are easily reusable on other projects.

❷ **Structured text** encourages good programming practices by using both structured programming and data structuring. It's an efficient language, and excels in operations with complex mathematical algorithms and assignments. Structured text is composed of predefined statements that dictate program flow and assign values to variables. Variables can be explicitly defined values, internally stored variables, or inputs and outputs.

❸ **Ladder diagram** is perhaps the world's most widely used industrial automation programming language, originally developed to simulate relay logic schematics. Ladder diagrams are best used for control of discrete I/O. Common functions are relay logic, timing, and counting.

In short, ladder rungs flow from left to right and may have a variety of input conditions that eventually lead to a single output instruction. The rungs have assigned addresses that indicate data location. Once programmed conditions are met, output is set accordingly.

❹ **Instruction list** is the simplest and most basic form of the five languages, as it resembles assembly programming of low-end controllers. This language uses a stack structure like an HP calculator and consists of open and closed brackets. Within these stacks are modifiers — which negate inputs or outputs, nest an operation, or put it on a stack to be pulled off, and perform checks for currently evaluated results at the top of the stack. Because IL is the most fundamental level of the five languages, the first three options

Benefits of IEC 61131-3

- *Reduces the learning curve*
- *Makes it easier to employ new controller technologies*
- *Simplifies maintenance*
- *Promotes reuse of software code*
- *Lets designer employ best language for the situation*

The Yaskawa MP2300S controller ➲ can be loaded with software developed in MotionWorks IEC 61131-3-compliant software to execute synchronized sequences and motion control in one integrated platform.

described here (FB, ST, and LD) can all be converted to instruction list. In fact, in the early days of embedded control, IL was the most common programming language, because it easily translated into machine language codes for any processor, ensuring efficiency and fast program execution speed.

❺ **Sequential function chart** organizes programs for sequential and parallel control processing for diagnostic purposes, batch processes, and business applications. It describes the sequential behavior of a control program by breaking the program into manageable parts.

Sequential function chart is a powerful method of structuring sophisticated algorithms. It's not an independent language, but an organizing tool — and it resembles a block diagram composed of steps, actions, and transition conditions.

Steps are units of programming logic that accomplish a particular task, while actions are aspects of that task. Steps represent a particular state of the system, and they perform control functions through the use of actions. Transitions are the mechanisms employed to move from one task to another. A transition is coupled to a condition: When a condition becomes true, it deactivates the previous step and activates the next. Sequential function chart is used for complex sequential applications where multiple operations must be performed simultaneously.

Why IEC?

IEC 61131-3 is spreading because of the many benefits for its users. IEC-compatible software excels at providing an environment for easily coding all the functionality demanded of a modern automation system. IEC 61131-3 also lends itself to the creation of reusable functions and libraries. A function block written in any language can be stored within a library — the easiest way to ensure portability. This allows programmers to import code developed during previous projects instead of starting from scratch.

Function blocks can inherit complex data structures and functionality developed prior to the current project. This quickens the design stage, resulting in lower engineering costs and easier customization. It also allows programmers to employ function blocks that have already been field-tested on prior projects, expediting commissioning and startup.

Although each IEC language has its own purpose and strengths, they all work together seamlessly. In fact, some of the best designs allow interoperation of several languages to command targeted results within one program. Because the different languages can be intermixed in an application, programs are easier to create and understand.

Benefits aside, the IEC standard does have limitations: For example, code written for one vendor's hardware platform often must be significantly modified to run on another vendor's hardware. Even so, IEC 61131-3 has laid the foundation for more efficient engineering.

For more information, call the author at (847) 887-7029 or visit yaskawa.com.

YASKAWA

www.yaskawa.com

(800) YASKAWA (927-5292)

PR.MSD.01