ADVANTECH
Enabling an Intelligent Planet

Technical White Paper

December 2011

Soft Motion Simplifies Motion Control Programming

Author: Advantech
E-mail: eainfo@advantech.com

Motion control was once performed exclusively through tedious and painstaking low-level code-based programming of hardware devices; but, modern machine automation hardware and software greatly simplified this task and allowed some of elements of machine motion software programs to be automatically generated from user-entered move data. At the same time, users began to write machine control software programs by mixing and matching the five IEC 61131-3 programming languages, all of them high-level and some of them graphical.

Once the machine motion software program is generated, the compiled code is downloaded directly to the hardware platform, often, but not necessarily PC-based. This hardware platform then interacts with motion amplifiers and servo motors to control and monitor a complete motion control system.

This method of generating software using high-level and graphical languages will be referred to as softmotion throughout this white paper, a generic term that refers to many vendor-specific software programming packages, including those from Advantech.

This white paper will define machine motion, then compare and contrast traditional and modern programming methods for programming machine motion control systems.

Machine Motion Defined

Motion is a change in location or position of an object. Moving an object to a position in a two-dimensional plane requires knowledge of both the direction and the magnitude of the intended move. Using the Cartesian coordinate system, any two-dimensional location can be identified by an ordered pair of perpendicular lines with scalar relationships to perpendicular reference (*X* and *Y*) axes.

A unique location in three-dimensional space is identified by an ordered triplet of lines, any two of them being perpendicular. These lines, or axes, also have scalar relationships with their reference axes. The added reference axis is called the *Z-axis*, and is perpendicular to both the *X-axis* and the *Y-axis*.

Rotational motion involves the rotation of an object around its center of mass. The location of any point on or within an object other than its center of mass is determined by that point's angular displacement. Rotational motion is simply spinning motion. For example, an electric motor spins on its shaft, or axis. A machining or woodworking lathe spins the work piece. Converting machines, web-type presses, and pulp-and-paper machinery have myriad rollers that spin on their shafts.

Machine motion occurs when a machine causes a change in location or position of an object. A machine that punches holes in steel plates or one that places components on an electronic circuit board are machines that produce two-dimensional movement. Milling machines and robots produce three-dimensional movement. Most machine designs combine two- and three-dimensional movement with rotational movement to accomplish their intended tasks.

Machines used in manufacturing use motion to join or cut materials; form complex shapes; or move materials, parts, assemblies, or finished goods. Joining technologies include welding and brazing. Material movement includes material-handling tasks such as packaging, palletizing, pick-and-place, conveyance, and automatic storage and retrieval.

Cutting technologies include—but are not limited to—metal cutting, turning, milling, drilling, grinding, and sawing. Cutting technologies that involve burning are related to welding and include laser, oxy-fuel burning, and plasma cutting.

These cutting technologies are included in the broad category of machine tools. A machine tool has direct mechanical control of the path of the cutting tool, as opposed to a human directly controlling the tool's path. Machine tools use computerized numerical control to handle machine motion.

Many manufacturers incorporate robotics into their automated processes. ISO Standard 8373 "Robots and robotic devices—Vocabulary" defines an industrial robot as an automatically controlled, reprogrammable, multipurpose manipulator that is programmable in three or more axes.

Robot configurations include articulated, Cartesian (gantry), selectively compliant assembly robot arm (SCARA), delta (spider), polar, and cylindrical. Those most commonly used in industry are articulated, Cartesian, SCARA and delta. Industrial robots are used in assembly, welding, painting, packaging, palletizing, pick-and-place and product inspection/sorting. The automotive manufacturing industry has been a heavy user of robots for decades.

Regardless of the type of automated machine—web-type printing press, CNC machine tool, packaging machine, or robot—a program must control every movement the machine makes. Programs must provide parameters such as magnitude, direction, velocity, acceleration, and deceleration.

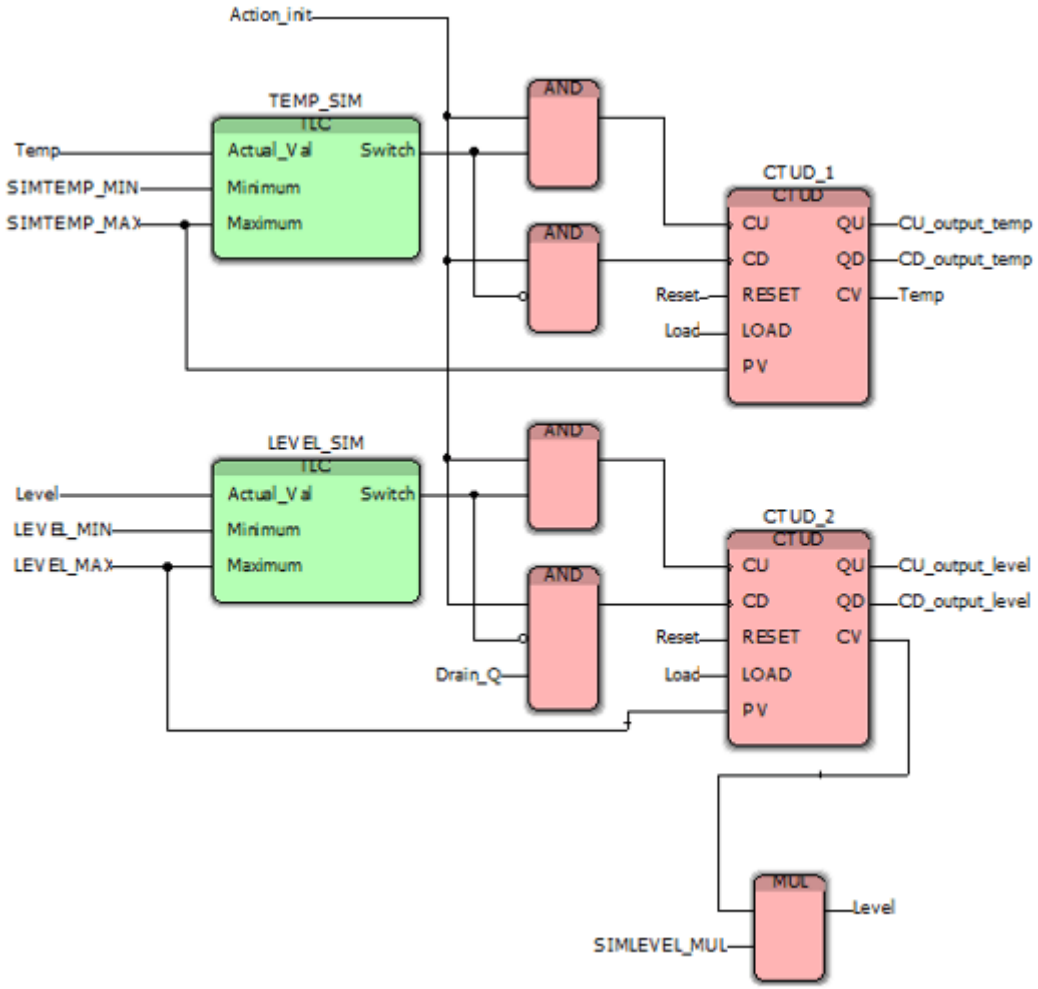
Motion programming details depend on motion complexity. Simple motion sequences such as point-to-point moves, as well as complex sequences that don't stop between sections, must all be considered. In addition, part profiles or recipes for machines that make components must be included in these programs.

PC-based programming software is generally used to generate the machine motion program. This programming software program is typically also capable of generating logic to monitor inputs and control outputs for the balance of the machine.

The PC-based programming software generates compiled code that is downloaded to the machine control hardware platform. This hardware platform can be a PLC, a PC or a motion controller. In most cases, the hardware platform is capable of controlling the entire machine, including motion and other functionality. In terms of motion control, the hardware platform interacts with motion amplifiers and servo motors to control, operate, and monitor the motion control system.

Machine Motion Programming before Soft Motion

Engineers can select motion control hardware from a wide range of choices to satisfy motion-control application requirements and project specifications. Before softmotion, hardware differences required the creation of unique programs for each application—even if the functions were the same.



The fragmented motion control market produces a wide variety of incompatible systems with different architectures and different software tools for development, programming, and maintenance. This incompatibility significantly increases system costs. In addition to the inability to reuse software across platforms, applying different software implementations creates confusion and increases engineering difficulty.

Using custom, proprietary software to program and operate vendor-specific hardware—especially motion controllers and PLCs—has been the norm. Mastering unique programming software for different controllers makes the learning curve steeper. This increases development costs as well maintenance costs, because in each case technical personnel must learn how to use a variety of unique software programs.

Programming motion-control hardware using low-level code is exceedingly tedious. In the past, low-level code was written manually for a handful of processors, usually without the benefit of an operating system. In those days, exchanging data between processors was nearly impossible, or at least not economically feasible.

At that time, although many programming packages specialized in only one language, it was still difficult to represent the logic in ways that other programmers and/or debuggers could understand. Even though ladder logic was typically used for PLC-based systems, there were different user interfaces for different PLC brands. PLC ladder logic was also not suited for many aspects of motion control, so many PLC vendors created custom and proprietary special function blocks to handle motion control tasks.

Even now, when using custom and proprietary software, developers and programmers must start from scratch for each new system or machine. Reinventing the wheel is necessary for every new controller type because code generated from proprietary software isn't reusable, scalable, or portable across platforms. This software incompatibility means that motion control applications are prone to mistakes and difficult to debug and maintain, which increases development costs and time-to-market.

Soft Motion Software Development

Softmotion refers to a method of generating application programs using high-level and graphical programming languages. With some motion control software programming packages, code can be automatically generated from user-entered textual or graphical information.

Most softmotion software programming packages are based on—and conform to—the PLCopen Motion Standard, which fits within the framework defined by the IEC 61131-3 standard. The purpose of the relationship and interaction between the IEC 61131-3 and the PLCopen Motion Standards is to harmonize motion control software development across different hardware platforms.

IEC 61131-3 is a globally-recognized standard for programmable controls. PLCopen promotes the use of IEC 61131-3, as does PLCopen Motion Control, which is one of the technical committees working within PLCopen. The mission of the PLCopen Motion Control Standard is to support, propagate, and promote the IEC 61131-3 standard. Its goal is to ensure that software programs can be transferred among different brands and/or different types of control applications.
The need for standardization emerged as motion control became more integrated with the traditional PLC environment. Different suppliers within the PLCopen organization recognized this need and responded, resulting in the definition of a PLCopen motion control library of reusable components. The characteristics of the library are:

- Programming depends less on specific hardware
- Application software reusability is increased
- Training and support costs are reduced
- Application is scalable across different control levels.

The PLCopen motion control library is based on IEC 61131-3 function blocks, and can be used to create understandable application programs that are reusable across multiple platforms.

The PLCopen motion standard defines common motion-control actions as functions.
Motion-control functions from different vendors that certify to the PLCopen motion standard will

have the same interfaces and behaviors. The modular reusability that IEC 61131-3 provides and that the PLCopen motion standard refines allows motion programmers to focus on their applications instead of coding minutiae.

IEC 61131-3 Details

IEC 61131-3 standardizes the interfaces between PLCs and other controllers and their programming systems, programming languages, different instruction sets, and project structuring and handling. It also standardizes various automation system languages, command sets, and structure models. Using controllers and programming systems that conform to IEC 61131-3 ensures some degree of platform portability and compatibility, and yields the ability to reuse some parts of application programs across platforms.

The two major parts of the IEC 61131-3 standard are the Common Elements and Programming Languages. IEC 61131-3's Common Elements include program organization units (POUs), variables, data typing, and configuration.

POUs: POUs provide structure by defining clearly structured "compartments" for the program code. Each POU has a code part and a variables-declaration part. The different POU types are program, function, and function block. Functions allow program elements to extend the instruction set of a configuration (another Common Element described later in this white paper). A function block is a basic "building block" unit used to build applications. Both function and function block POUs can be used within the same project as well as when using libraries in other projects. Networks of functions and function blocks are POU programs.

Variables: Variables are used instead of directly addressing inputs, outputs, and flags. As with high-level programming languages, variables in the programming project must be declared and can be initialized with a starting value. The three kinds of variables are symbolic, directly represented, and located, which refers to a specific hardware address. It's also possible to declare retentive variables. The values of retentive variables are retained even if the hardware is switched off or loses power.

Data typing: Data typing formally defines parameters. Data types include elementary, generic, and user-defined. IEC 61131-3 requires data typing in order to standardize data and prevent errors. Data types determine the format, size, possible value range, and possible initial value of variables. The data-typing element within the IEC 61131-3 standard makes user-defined data types such as arrays and structures possible.

Configuration: The intention of configuration is to resolve problems with hardware arrangement, memory addressing, and processing resources. A hardware platform contains various resources that can execute IEC 61131-3 programs. These resources contain tasks that consist of control programs and function blocks.

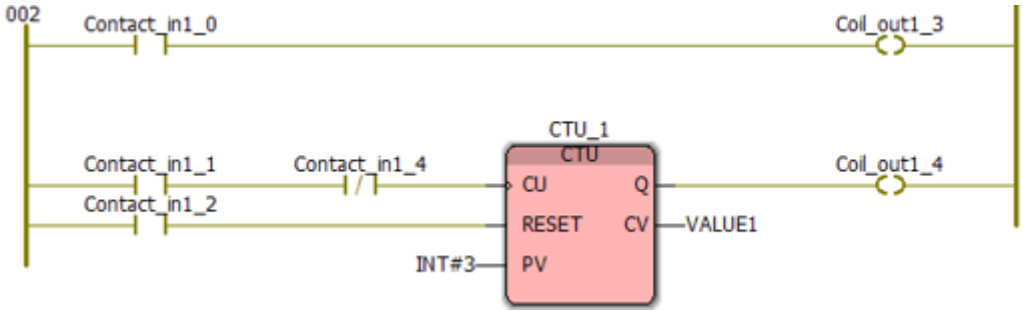
The other major part of the IEC 61131-3 standard consists of Programming Languages. The IEC 61131-3 standard describes three graphical and two textual programming languages. The standard also defines their language elements as well as their syntax.

IEC 61131-3's three graphical languages are Function Block Diagram (FBD), Ladder Diagram (LD), and Sequential Function Chart (SFC). Graphical programming is much easier to understand for non-programmers. The standard's two textual languages are Instruction List (IL) and Structured Text (ST). All of these languages can coexist and exchange data within the same program.

FBD: FBD displays high-level interactions visually, and is well-suited to complex tasks such as motion control. FBD is also used in continuous process industries because of the language's flow-oriented properties. Functions and function blocks are either linked or connected to variables by lines that carry information from left to right. Control logic is created by connecting blocks and elements. The blocks can be predefined or user-created, and can be used on other projects.

Some programming software packages include function blocks dedicated to certain tasks such as particular aspects of motion control. Motion function blocks can be used to execute common motion control functions, and can be easily reused across different applications. Users typically enter motion parameters in the motion function block, and code generation is performed automatically by the programming software.

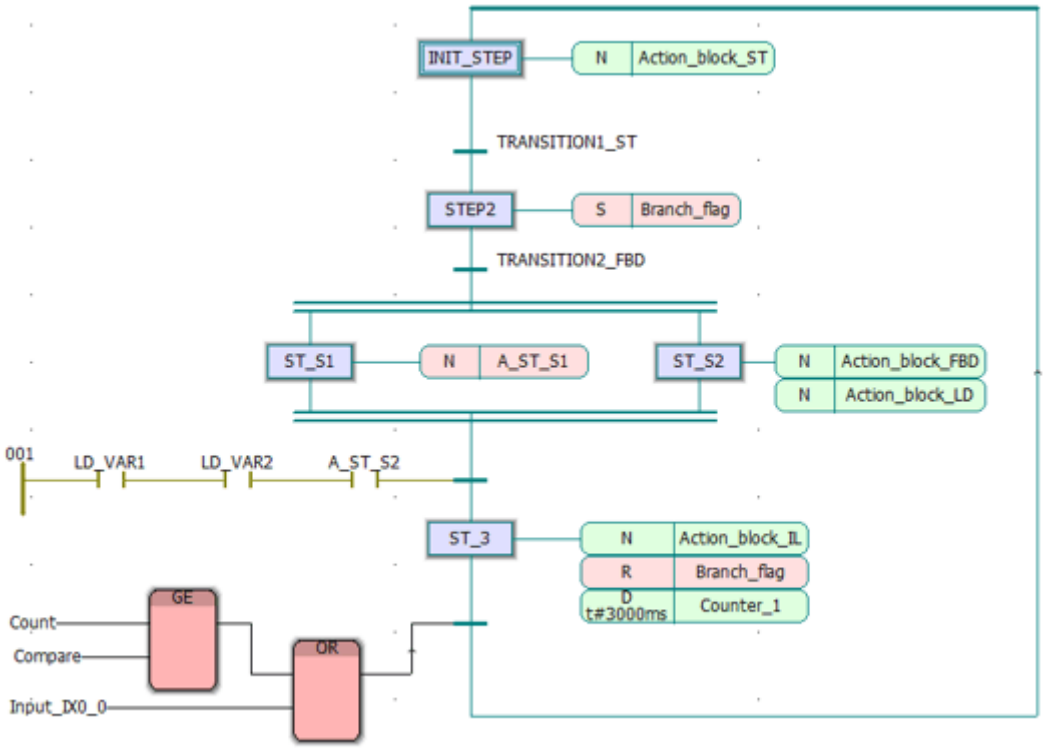
LD: LD, commonly referred to as ladder logic, simulates relay logic schematics and has been used for programming PLCs and other controllers since their invention. It's widely accepted and used worldwide because of its ease of use. LD is ideally suited for programming sequential logic and controlling discrete I/O.



The LD instruction set consists of different types of contacts and coils. These contacts and coils are not hardware; they're software algorithms that symbolically represent the functions contacts and coils would perform in relay logic. Depending on their types, the symbolic contacts lead the power along the ladder rung from left to right, and may have a variety of input conditions that eventually lead to a single output instruction. LD rungs have assigned addresses, which indicate data location. The symbolic coils store incoming values. Both contacts and coils are assigned to Boolean variables. When the programmed conditions are met, the output is set accordingly.

Common LD functions are relay logic, timing and counting. As with FBD, a LD network can be supplemented by jumps, returns, labels, and comments. Some IEC 61131-3 software programming packages and systems allow use of FBD elements in LD networks. If these FBD elements include motion function blocks, then the programming package can be well suited to machine motion applications.

SFC: SFC is a graphical, status-oriented language that's particularly suited to applications that can be structured into clearly identifiable steps. Although it's not an independent language, it is a powerful organizing tool. Code programmed in SFC consists of steps and transitions. A step is a programming logic unit that accomplishes a specific status-related task; actions are aspects of that task. A transition moves the program sequence from one task to another.



Using SFC, a control program can be broken into manageable parts that can reveal the program's sequential behavior. It's ideal for complex sequential applications where multiple operations must be performed simultaneously. SFC is typically used during machine debug and commissioning, as structuring the application into single steps significantly simplifies program diagnosis—especially compared to a typical LD program with a large number of networks.

IL: IL is textual and similar to assembly language. It's the simplest and most basic form of the five IEC 61131-3 languages. IL code consists of a sequence of instructions separated by lines, which consist of one operator, one operand, and one optional modifier. The operator is simply a command; the operand can be a variable, constant, or instance name.

```
1     LD      %IX0.2
2     AND     %IX0.3
3     OR      Action_INIT
4     ST      IL_VAR
5
6     LD      Input_IX0_0
7     JMPC    MANUAL
8
9     (*Timer FB TON*)
10    LD      Timer_start
11    ST      TON_IL.IN
12    LD      PT_TON_IL
13    ST      TON_IL.PT
14    CAL     TON_IL
15    LD      TON_IL.Q
16    ST      Action_INIT
17    STN     Timer_start
18    LD      TON_IL.ET
19    ST      Timer_value
20
```

Because IL is the most fundamental of the IEC 61131-3 languages FBD, ST, and LD can be converted to IL. This makes it easy to translate IL into machine language codes, which ensures fast program execution speed.

ST: ST is a high-level language similar to Pascal or BASIC, making it popular among many programmers. Its syntax and instruction set are well suited for mathematical calculations and data manipulations. ST uses both data structuring and structured programming, both of which encourage good programming practices. It's composed of predefined statements that dictate program flow and assign values to variables, which can be explicitly defined values, internally stored variables, or inputs and outputs.

```
1     CASE MODUS OF
2     1: ROBOT_X:= ROBOT_X + 200;
3        ROBOT_Z := ROBOT_Z + ADD_ARM ;
4        MODUS:=1;
5        IF ROBOT_X >= RANGE_POS_1 THEN
6           MODUS:=2;
7        END_IF;
8     2: ROBOT_X:= ROBOT_X - 200;
9        ROBOT_Z := ROBOT_Z - ADD_ARM ;
10       MODUS:=2;
11       IF ROBOT_X <= RANGE_NEG_1 THEN
12          MODUS:=1;
13       END_IF;
14    END_CASE;
15    ROBOT_Y:= ROBOT_X;
16    COUNTER_1 := COUNTER_1+1;
17    IF COUNTER_1 >1000 THEN
18       COUNTER_1 :=0;
19    END_IF;
20
```

Soft Motion Advantages

According to PLCopen, softmotion programming solutions that conform to the PLCopen Motion Standard enable standard application libraries that are reusable for multiple hardware platforms. Reusable standard application libraries can eliminate confusion while lowering development, maintenance, and support costs. Training costs decrease and engineering becomes easier.

By defining libraries of reusable components, the programming depends less on hardware, and the application becomes scalable across different control solutions. Because of data hiding and encapsulation, the application is also usable on different architectures. For example, application use can range from centralized to distributed, or from integrated to networked control.

Standardized softmotion programming solutions increase application development flexibility. Some softmotion solutions allow mixing of multiple IEC 61131-3 languages in a single worksheet, insertion of new elements into existing networks, and moving of either single objects or networks. Mixing different languages in an application makes programs easier to create and understand.

Another advantage of softmotion is automatically-generated code based on user-entered motion data or profiles. This feature allows users to create programs that can automatically generate code based on desired move profiles that include information such as start point, end point, speed/velocity and acceleration.

Graphical editing tools (for programming with FBD/LD/SFC) enable users to graphically create custom motion and cam profiles. These tools also allow users to combine optimized custom motion profiles with automatic placement, routing, and keyboard operations.

Some softmotion programming solutions also allow users to enter mathematical expressions that provide flexible programming capabilities for advanced calculations and machine control sequences. For example, users can leverage technical computing software such as MATLAB and simulation/model-based design software such as Simulink to create models that can be used as a basis for automatic code generation.

The capability to automatically-generate motion programs based on user-entered motion data reduces manual programming efforts—and in many cases, actually eliminates parts of these efforts because the programming software automatically generates the code.

Together, automatic code generation and reusable standardized modules enable flexible application integration, reducing programming effort and time to market.

Conclusion

Traditional machine motion software development can be frustrating, tedious, prone to error, expensive, and time consuming. Incompatibility among different types of motion control hardware—even from the same vendor—requires different software tools for development, programming, and maintenance.

Learning and using different motion control programming software increases system costs because software can't be reused across platforms. Applying different software implementations creates confusion and increases engineering difficulty.

However, softmotion software programming packages that conform to the IEC 61131-3 and PLCopen motion standards harmonize motion control software development across different hardware platforms. Softmotion solutions enable the development of standard application libraries that are reusable for multiple hardware platforms. Programs produced with softmotion programming software are also easier to understand, particularly for non-programmers.

Softmotion software development is more flexible; lowers training, development, maintenance, and support costs; and shortens time-to-market. These advantages are increasing available conforming software systems, and also increasing compatibility among these systems.

Table 1: Issues with Traditional Machine Motion Software Development

1. Hardware dependent
2. Steep learning curve
3. Not portable
4. Not reusable
5. Difficult to debug
6. Difficult to maintain
7. Tedious to develop
8. Error prone
9. Lengthens time-to-market
10. Incomprehensible to non-programmers

Table 2: Advantages of Soft Motion

1. Hardware independent
2. Flexible
3. Automatic code generation
4. Reusable
5. Portable
6. Lowers costs
7. Eliminates confusion
8. Simplifies engineering
9. Shortens time-to-market
10. Easier to understand for non-programmers

Table 3: IEC 61131-3 Common Elements

1. Program organization units
2. Variables
3. Data typing
4. Configuration

Table 4: Languages Included in the IEC 61131-3 Standard

Language	Type
Function Block Diagram	Graphic
Ladder Diagram	Graphic
Sequential Function Chart	Graphic
Instruction List	Textual
Structured Text	Textual

References

1. Pick-and-Place Applications for Robots, http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Featured-Articles/Pick-and-Place-Applications-for-Robots/content_id/2504
2. Robotics in Electronics, http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Featured-Articles/Robotics-in-Electronics/content_id/2811
3. Motion Control Revealed, http://www.controldesign.com/articles/2004/103.html?page=full
4. IEC 61131-3 and Programming, http://www.controldesign.com/articles/2010/IECProgramming1008.html?page=full

5. Why can't my software behave like my hardware?, http://www.isa.org/PrinterTemplate.cfm?Section=InTech_Home1&template=/ContentManagement/ContentDisplay.cfm&ContentID=83566
6. Save time with reusable code, http://motionsystemdesign.com/system-solutions/save-time-reusable-code-20100501/index.html
7. The IEC 61131 standard: basics and background, http://www.kw-software.com/com/iec-61131-sps/2876.jsp
8. PLCopen, http://www.plcopen.org/

Image 1, Machine. Programming machines like this one to perform motion-related tasks is much simpler with graphical programming software packages that conform to the IEC 61131-3 and PLCopen industry standards.

Image 2, Function Block Diagram. Function Block Diagram displays high-level interactions visually and is ideal for complex tasks such as motion control.

Image 3, Ladder Diagram. Ladder diagram (LD) simulates relay logic. Some software programming packages allow use of motion control and other function blocks within LD networks.

Image 4, Sequential Function Chart. With Sequential Function Chart is ideal for complex sequential applications where multiple operations must be performed simultaneously.

Image 5, Instruction List. Instruction List (IL) is textual and similar to assembly language, and it's the simplest and most basic of the five IEC 61131-3 languages.

Image 6, Structured Text. Structured Text is a high-level language similar to Pascal or BASIC, making it popular among many programmers. Its syntax and instruction set are well suited for the mathematical calculations required for many motion applications.