CONTROL ENGINEERING

PAC or PLC, Controller Selection Considerations

Sponsored by

Sponsor Profile

AutomationDirect is a well known distributor of 12,000+ industrial automation products, such as Programmable Logic Controllers (PLCs), Programmable Automation controllers (PACs), AC drives/motors, operator interfaces/HMI, power supplies, DC motors, sensors, encoders, relays, timers, push buttons, NEMA enclosures, pneumatic components, power products, safety supplies, and much more. In business since 1994, the company headquarters is located just north of Atlanta, Georgia.

Our prices are low.

Our prices are well below the list price of more traditional automation companies because with our business model and focus on efficiency, AutomationDirect has the lowest overhead in the industry.

We make ordering easy and our service is exceptional.

Shop online with our exhaustive product listings or browse our online catalog; fax or phone us – you'll get friendly, efficient service from the most helpful sales team in the business. Independent surveys completed by readers of Control Design magazine placed us at the top of the list for service 11+ years in a row in their Readers' Choice awards. Other surveys by magazines, such as Control Engineering and Control, have echoed the results.

We ship super fast (and FREE on orders over $300).

The majority of our products are stocked for same-day shipping, when you place your order by 6p.m. E.T. (with approved company credit or credit card). Plus, you get free two-day (or better) shipping on orders over $300 U.S. within the U.S. and Puerto Rico; some exclusions apply.

We guarantee it.

We want you to be pleased with every order. That's why we offer a 30-day money-back guarantee on almost every stock product we sell, including our software (see Terms and Conditions for certain exclusions).

For more information, contact us at 800-633-0405 or visit www.automationdirect.com/plcs?control-eng

END

Cover Story: Balancing PLCs, PACs, IPCs

Do You Need A PLC, PAC, Or IPC For Your Next Control Application? Will Programmable Logic Controllers (PLCs) Evolve Into Programmable Automation Controllers (PACs) Or Industrial PCs (IPCs)? Whatever The Name, Get The Best Features And Software For Your Control Applications.

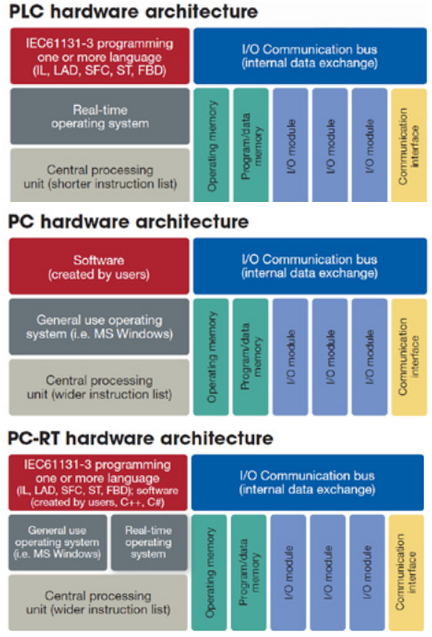
Krzysztof Pietrusewicz, PhD, and Łukasz Urbański, MSc, W Pomeranian University of Technology

01/17/2011

Controller-based applications face a divergence when selecting a controller: Simplicity and ruggedness, or openness and functionality? In math, that's called indeterminacy, but in automation, the engineer should know what controller feature set is best for the application, no matter what you call the logic device.

In 2001, when ARC Advisory Group Analyst Craig Resnick proposed the new term for the new class of controller (PAC, programmable automation controllers), the acronym was coined on the basis of market observations. Functionality of programmable controllers was extended by the main global automation vendors.

Vendors use the PAC acronym to describe a class of products conforming in design and market positioning with similar hardware–software architecture philosophy. Because there are a wide variety of feature sets, it has become difficult to definitively classify each controller as either a PLC or a PAC. Even a small PLC can be programmed with the use of the automatic code generation tool from Matlab/Simulink. The same controller, given 8 GB to 32 GB of data memory, thanks to the compact flash (CF) interface, can be a data acquisition and analyzing tool for predictive maintenance of the machine. The same controller, thanks to virtual interface technology, can be changed into the remote gateway for the whole process.



PAC definition

Programmable automation controllers (PACs) are known for the following main features, differentiating them in functionality from a typical

PLC:

- Multidomain functionality—logic, motion, continuous control systems can be designed with the same hardware platform.
- Single multidiscipline development platform—data tags are stored in a common database.
- One tool for every programming task—control logic, motion control, HMI design for more than one machine in a process.
- Open, modular architecture lets the engineer use only the equipment he need.
- Use of many communication standards (from asynchronical to deterministic ones) and many programming languages (consistent with IEC 61131-3 as well as the higher level programming languages)—the engineer can design a multivendor system simply and efficiently.

But as PLC technology has emerged, some companies look at PLC and PAC differences and choose to use the PAC acronym for their products, even without offering all the features mentioned above.

Many things have changed for programmable controllers since 2002:

- Openness of communication standards is typical with PLC functionality.
- More tools are IEC 61131-3 compliant, extending the normative list of programming languages (IL, LAD, ST, SFC, FBD) with ANSI C or even C++ and C#.

CONTROL ENGINEERING

- Small controllers are equipped with a large amount of data memory (8 GB and more).
- Ethernet TCP/IP has become the most popular programming interface for PLCs.
- Software architecture of PLCs is based on the real-time deterministic multitasking operating systems.

Given the blending of features, it is difficult to define which controllers are PACs and which are PLCs. The simplest definition says that a PAC combines PC elasticity and functionality with the ruggedness of a PLC.

PAC controllers combine PC and PLC hardware/ software architecture, offering flexibility and ruggedness in one efficient system.

Two-system operation

Users in recent years have built applications within two operating systems architectures. Examples include Beckhoff Automation (Twin- CAT PLC under Microsoft Windows XP Professional) and Bernecker & Rainer (AR010 system under Microsoft Windows XP Professional), among others.

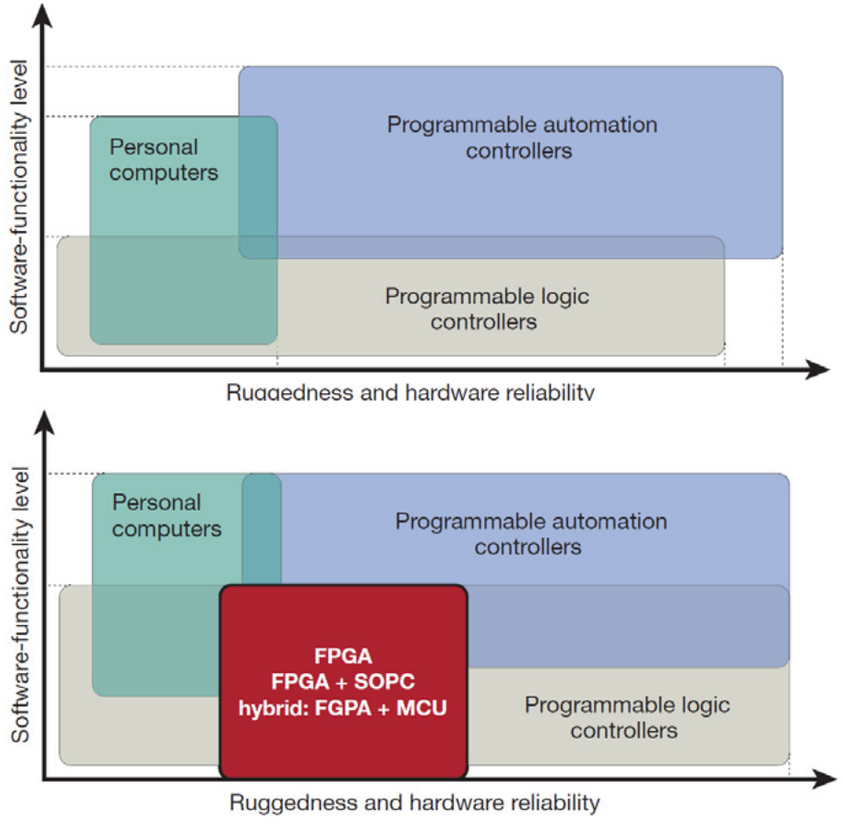
National Instruments PXI series controllers are available in two-boot operating systems versions— real-time and embedded, with Microsoft Windows XP Professional and with Microsoft Windows 7. This design can be booted only in one system mode, different from what the figure shows.

Functionality, classification

As the functionality of PLCs has been extended and newer control technologies have been introduced, the range of controller definitions needs to be reevaluated since 2001.

An increasing number of engineering tools are modernized to comply with the Microsoft Windows 7 32-bit operating system. Fewer 64-bit computers are used, but it hasn't been a problem. The first updates of the programming tools (for development of control applications) for the Microsoft Windows 7 system were released in mid-2010.

Development tools for programmable controllers will continue to receive updates in 2011. Other companies also are working on the 64-bit versions of their tools, with most upgrades expected to be complete by year end.



Name	Supported targets	Description, website
M-Target for Simulink	Bachmann M1 (PC Based)	www.bachmann.info M-Target for Simulink enables the use of Simulink and other toolboxes from the product family for automatic generation of executable real-time programs for the Bachmann M1 automation system.
B&R Automation Studio Target for Simulink	Bernecker & Rainer	www.mathworks.com B&R Automation Studio Target for Simulink provides an easy-to-use but powerful solution for implementing advanced closed-loop controllers on B&R hardware. Providing an additional Simulink blockset for variable and parameter exchange, B&R Automation Studio Target for Simulink supports the whole range of B&R hardware for integrated automation.
M2F Interface	Fix32/iFix environment	www.inteco.com.pl M2F establishes a data exchange link between the Fix32/iFix and Matlab/Simulink environments. M2F includes Fix database blocks, which embed the executable modules inside the Fix database and process data in real time under the control of the Fix scheduler as specified by the source Simulink model. Tasks can be rapidly prototyped and applied to process Fix database signals in real time, such as design and implementation of digital filters, FFT calculation, design and tuning of advanced controllers, finite state machine implementation, and fuzzy and neural data processing. The M2F package is designed for Fix32 and iFix users who wish to apply complex and sophisticated Fix database signal processing. The robust set of functions available in Matlab/Simulink significantly extends the data processing capabilities of the Fix environment. Matlab can be applied to online Fix database processing. Simulink and Real-Time Workshop can be used for automatic generation of stand-alone executable modules.
TwinCAT 3 Integrated	Beckhoff	www.beckhoff.com In TwinCAT 3, Beckhoff introduces the newest software generation for PC-based control technology that will expand the world of automation with many powerful new functions. The broad term for this new technology is eXtended Automation (XA).

Name	Supported targets	Description, website
Mathworks PLC Coder	3S-Smart Software Solutions CoDeSys 2.3, 3.3 B&R Automation Studio 3.0 Beckhoff TwinCAT 2.11 KW-Software Multiprog 5.0 PLCopen XML 2.01 Rockwell Automation RSlogix 5000 17, 18 Siemens Simatic Step 7 IDE 5.4	www.mathworks.com/products/sl-plc-coder/ Simulink PLC Coder generates hardware-independent IEC 61131 structured text from Simulink models, Stateflow charts, and Embedded Matlab functions. The structured text is generated in PLCopen XML and other file formats that support integrated development environments (IDEs).
M2PLC Interface	GE Fanuc Series 90-30 PLC equipped with FPU module or Series 90-70 PLC. (VersaPro ver 2 or Simplicity Control)	www.inteco.com.pl The M2PLC extends PLC functionality with new data processing algorithms by enabling you to use Simulink as a modeling environment and transfer those algorithms to the PLC. Simulink graphical blocks make design of control and data processing algorithms fast and easy. Logic blocks, FSMs, neural network blocks, filters, and different structure controllers can be developed and translated to C code using Real-Time Workshop. M2PLC automatically generates PLC executable files from the Real-Time Workshop generated code, which become blocks in a PLC ladder diagram folder. The M2PLC interface fully integrates a system-level design environment with industrial PLC applications. It is a perfect tool for designers who need to apply complex data processing and control methods to a PLC environment.
PLC Link	Supported targets 3S – CoDeSys Dachmann – M-PLC Beckhoff – TwinCAT KW-Software – Multiprog Phoenix Contact – PCWorX Siemens – Simatic Step7 IecEdit Rockwell Automation – RSLogix5000	www.deifwindpower.com PLC Link is an automatic code generator and tool chain for speeding up development of advanced control systems targeting IEC61131-3 PLCs using The MathWorks' Matlab and Simulink for simulation and environment for Model-Based Design.

Automatic code generation

The number of programming tools consistent with IEC 61131 programming languages continues to increase. Users can program applications with more than just a ladder diagram computation model, typically with three or more languages, including structured text.

New programming software offers new opportunities, such as automatic code generation for programmable controllers. This functionality allows users familiar with the MathWorks Matlab/ Simulink environment to rapidly design and implement control algorithms within the control tasks. Research and development of control strategies is one of the main areas of opportunity for new products. Shortening the 'time to market" of new product development is the reason for employing the mechatronic approach, also known as the model-based design.

Tools for automatic code generation are listed, with a list of controllers for which they are designed.

Future of control systems

How will control systems evolve? Automation vendors are incorporating safety technology in their systems. More are integrating motion control functionality in available programming tools. Rapid prototyping is the most innovative method of evolution, and applications also will accommodate that functionality. In 2020, control architectures will be so open that there will be no problem with interchanging control solutions and hardware modules from different vendors, maybe even on the level of processors and programming tools. This may speed the natural evolution of today's continuing expansion of object-oriented programming tools. Tools will decrease in importance as the team of programmers work more efficiently to solve problems, which is the promise of modern control systems.

- Krzysztof Pietrusewicz, PhD, is assistant professor at the Control Engineering and Robotics chair, Faculty of Electrical Engineering at the West Pomeranian University of Technology, Szczecin, Poland, and contributor to Control Engineering Poland. Łukasz Urbański, MSc, is a PhD student at the Faculty of Electrical Engineering, West Pomeranian University of Technology

END

Linkedin Discussion On PLC Vs. PAC: When And Where?

Users Of Linkedin's Automation And Control Group Discuss The Differences Between PLCs, PACs - When One Option Better Suits A Specific Application.

01/17/2011

Marketing hype or substantial differences? Users offer their observations, opinions. Should you use a PLC (programmable logic controller) or a PAC (programmable automation controller) for your next application? Is there a difference? How should you choose? Following the same topic as the cover story in this issue, users of the Automation and Control Group on LinkedIn offer their thoughts. The general consensus is that a PAC offers a higher level of sophistication.

'PAC is merely a marketing term for the more capable PLCs," says Joseph Stevenson. 'The selection of one platform or another is highly contingent on the size of the process, based on I/O count, required scalability, and what the control system needs to do with that process."

Jeffrey T. Johannessen defines those differences in very specific terms. He observes, 'A PAC is not a PLC. A PLC has a single program path, but PACs can have multiple threads or multitasking, interrupt driven I/O, and event or driven processes."

To manufacturers and users, the key should be to use the right tool for the right job, according to Sloan Zupan. He suggests, 'The application will dictate whether a PLC or a PAC is best suited. A PAC really refers to automation platforms. These platforms typically combine multiple different control disciplines on a single rack. As an example, Mitsubishi Electric offers a product called the iQ Platform, which enables users to select any combination of the following CPU types: sequence, motion, robot, CNC, and C controller. The CPUs can be mixed and matched to fit the requirements of any application perfectly. By leveraging the same power supplies, racks, I/O and communication interfaces startup, normal operation and maintenance becomes much easier. That is the value of a true PAC. On the other hand, a PLC is nice to have for stand-alone applications where sequence and positioning control are all that is needed. The PLCs are flexible in their networking capabilities and can be connected to the PAC systems easily. The same software is used for both the PLC and the PAC because the instruction sets are the same. PLC programs can be easily ported to PAC systems as the control requirements change."

Brad Sibole sees PACs as the future but believes there is still too big a price difference. 'Since the automation field is following the 'from the shop floor to the top floor' mantra, the need of more powerful and robust systems is much needed. PAC is the future of the automation field, but it is hard to justify the cost of a PAC to PLC system at this time." Sibole goes on to offer his own checklist for when a PAC is the better choice. He says, 'Consider a PAC versus a PLC if your application requires:

- Advanced control algorithms
- Extensive database manipulation
- HMI functionality in one platform
- Integrated custom control routines
- Complex process simulation
- Very fast CPU processing, and
- Memory requirements that exceed PLC specifications."

Samit Mathur doesn't want to forget PLCs too quickly. To his thinking, they still have some critical advantages. 'PLC-based systems are much simpler, easier to fix, and offer a simple way to communicate and with high MTBF (on the order of 50 years), low MTTR (< 2 hours). Any Tom, Dick, or Harry can replace the faulty modules and it is by far the best in industrial applications. It's very important to understand the application before selection of program capacity of a PLC in terms of I/O count, communication interfaces, and size of application program."

Paul Sinclair thinks that the term PAC may be a buzzword but sees key differences in functionality over other control platforms. 'The best thing about a PLC or PAC is speed—the logic runs as fast as a racehorse while traditional DCS is slow," he says. 'Using a PAC/HMI combination gives an operator a real-time feeling which cannot be replicated in a DCS/HMI due to the inherently slow scan times. In a perfect world we want accuracy in our logic solvers and engaged operators watching over the process. Call it what you want: PLC, DCS, PAC, or any TLA will do as long as functionality, accuracy, and speed are product characteristics. The line is already pretty grey. My bet is that 10 years from now, DCS and PLC will be legacy products or obsolete like VHS and BETA video formats. The new kids on the block will all be PACs running on unified architecture platforms."

-Compiled by Peter Welander, content manager for Control Engineering. Reach him at pwelander@cfemedia.com.

Online

www.controleng.com/machinecontrol
Join the discussion at Control Engineering's social sites atop www.controleng.com.

CONTROL ENGINEERING

Speaking In Tongues: Understanding The IEC 61131-3 Programming Languages

Long Dismissed As Just A European Phenomenon, The IEC 61131-3 Programmable-Controller-Language Standard Is Gaining Traction In The United States. Many Controls Engineers Are Familiar With One Or A Few Of These Languages, But Not All. That Makes It Difficult For Them To Make The Best Choice For A Given Application Based On Programming-Language Characteristics. See Diagrams.

Ted Thayer, Bosch Rexroth Electric Drives and Controls

01/30/2009

More follows below on each of the languages, with diagrams.

Long dismissed as just a European phenomenon, the IEC61131-3 programmable-controller-language standard is gaining traction in the United States. Many controls engineers are familiar with one or a few of these languages, but not all. That makes it difficult for them to make the best choice for a given application based on programming-language characteristics.

Thanks to the International Electrotechnical Commission (IEC), five standard languages have emerged for programming both process and discrete controllers:

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

When should one be used over another? What are the benefits and disadvantages of each? For an in-depth look at each programming language with code examples, see the online version of this article on the Control Engineering Website at www.control-eng.com via the January 2009 archive.

Choosing An Appropriate Language

With the different programming languages available, it's important to consider a few factors before deciding which to use for your application. Of course, if you're already familiar with a certain language, then the tendency may be to stick with what you know. However, consider the high-level benefits of each language, as detailed below, before making a decision:

- Ease of maintenance by the final user: SFC;
- Universal acceptance of language: LD;
- Acceptance in Europe: IL or ST;
- Speed of execution by the PLC: IL or ST;
- Applications mainly using digital I/O and basic processing: LD or FBD;
- Ease of changing code later: LD;
- Ease of use by newer engineers: ST;
- Ease of implementing complex mathematical operations: ST; and

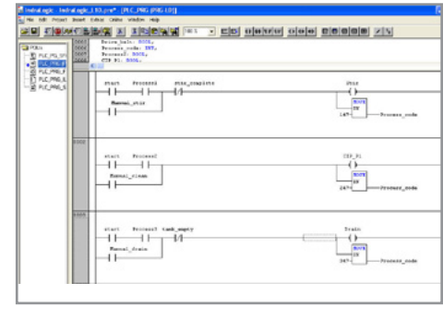
Applications with repeating processes or processes requiring interlocks and concurrent operations: SFC.

Your PLC or PAC platform may also affect the choice of programming languages, as not all automation vendors provide programming software that is fully IEC61131-3 compatible. In fact, most of the non-European vendors do not offer this functionality, or only have a very limited spectrum of options, such as Ladder and SFC.

Another consideration is that not all PLCs are capable of running the various IEC languages due to lack of memory or processor speed. This tends to be the case with many micro PLCs.

While many programmers are locked into a customer specification, if they have the freedom to choose a hardware platform, they should decide which language or languages will work best for the application, then select the hardware and software that support it.

ONLINE Extra



Ladder Diagram

Ladder Diagram (LD or simply Ladder) is probably the most widely used controller programming language. Invented to replace hardwired relay-based control systems, Ladder programming is used in probably 95% of all applications. Visually, this language resembles a series of control circuits, with a series of inputs needing to be "made" or "true" in order to activate one or more outputs.

Ladder has experienced such widespread adoption that almost every programmer in any country or industry can read and write this language. Because it resembles the familiar electric circuit format, even a non-programmer with an electrical background can follow the program for purposes of troubleshooting a problem.

It's also easy to start writing a program in Ladder. With just a basic outline of input and output signals, one can start churning out code. Most other IEC languages require more preparation, such as flowcharting of all potential process flows. Finally, most Ladder implementations allow a program to be organized into folders or subprograms that can be downloaded to the PLC, allowing easy program segmentation.

Ladder programming is ideal for simple material handling applications, for example, where a sensor detects the presence of a box, other sensors check for obstructions, and then an output fires an actuator to push the box to another conveyor. Digital inputs check for various conditions, and the program analyzes the inputs and fires digital outputs in response. There may be timers in the program, or some basic comparisons, or math, but there are no complex functions involved.

As the complexity of PLC functionality has grown, however, Ladder has been challenged to meet these advances and still maintain the paradigm of easy visualization and understanding. Functions, such as PID loops, trigonometry, and data analysis, now required in many control applications can be difficult to implement. Another challenge is that as program size grows, the ladder can become very difficult to read and interpret unless it's extensively documented. Finally, implementing full processes in Ladder can be daunting– picture a ladder rung with an output used in several phases of a process with many input conditions attempting to control exactly when that output needs to turn on.

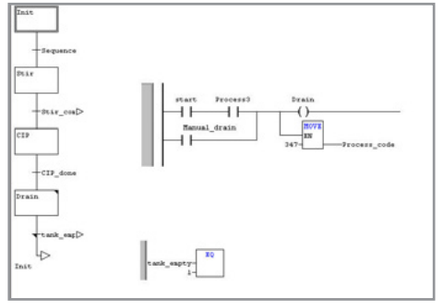
Function Block Diagram

Although Ladder may be the most widespread language, a survey conducted by Control Engineeringmagazine several months ago highlighted growth in the use of programming languages other than Ladder. Function Block Diagram (FBD) programming is an example. Even though the adoption rate for this language has recently slowed relative to other languages such as Structured Text, FBD is probably the second most widely used language.

In many ways, this graphical language resembles a wiring diagram even more so than Ladder code. With FBD, the blocks are "wired" together into a sequence that's easy to follow. It uses the same instructions as Ladder, but visually is more understandable to a viewer who is not versed in relay logic. The major advantage is that programs written in FBD tend to be easy to follow– just follow the path!

FBD is ideal for simpler programs consisting of digital inputs, such as photoelectric sensors, and outputs such as valve manifolds, and could be used in any application where Ladder works well.

However, FBD is not ideal for large programs using special I/O and functions. The large amount of screen space required can quickly make a program unwieldy if it reaches any substantial size. Also, writing a program in FBD requires more upfront preparation to understand the program and how it will flow before any code is written, since it can be difficult to make corrections later.



Sequential Function Chart

Sequential Function Chart (SFC) programming resembles the computer flowcharts that many engineers remember drawing up in their college days. An initial step "action box" (the starting point of a flowchart) is followed by a series of transitions and additional action steps. The SFC concept is simple: an action box, with code inside written in any language of the programmer's choice, is active until the transition step below it activates. The current action box is turned off, and the next one in the sequence is active. The transition step also has code to check that the necessary conditions are met to allow the program to advance to the next step. The language is very friendly to maintenance engineers because its visual nature and natural code segmentation makes it easy to troubleshoot.

On the downside, this style of programming is not suitable for every application, as the structure it forces on a program could add unneeded complexity. A large amount of time must be spent up front preparing and planning before any programming is attempted, or the charts can become unwieldy and difficult to follow. The overhead required for this type of program causes it to execute slower than the other languages.

A final consideration is the inability to convert to other languages. IL, FBD, ST and Ladder programs can easily be interconverted, allowing a piece of code to be displayed in the way most comfortable to the user. SFC, however, cannot be converted.

```
0001     LD       start
0002 AND      Process1
0003 OR       Manual_stir
0004 ANDN         stir_complete
0005 ST       Stir
0006
0007 JMPCN        en_temp0
0008
0009 LD       147
0010 MOVE
0011 ST       Process_code
0012
0013 en_temp0:
0014 LD       start
0015 AND      Process2
0016 OR       Manual_clean
0017 ST       CIP_P1
0018
0019 JMPCN        en_temp1
0020
0021 LD       247
0022 MOVE
0023 ST       Process_code
0024
0025 en_temp1:
0026 LD       start
0027 AND      Process3
0028 OR       Manual_drain
0029 ANDN         tank_empty
0030 ST       Drain
0031
0032 JMPCN        en_temp2
0033
0034 LD       347
0035 MOVE
0036 ST       Process_code
0037
0038 en_temp2:
0039
```

Instruction List

Instruction List programming (IL) consists of many lines of code, with each line representing exactly one operation. Thus, it is very step-by-step in layout and format, which makes the entry of a series of simple mathematical functions easy.

IL is a low-level language and, as such, will execute much faster than a graphical language, such as Ladder. IL is also much more compact and will consume less space in PLC memory. The simple one-line text entry method supported by this language also allows for very fast program entry– no mouse required, no tabs to click! In legacy systems, programs written in IL are easier to display and edit on a handheld programming unit, with no software or laptop required.

Despite IL's advantages, it seems that maintenance and service engineers do not prefer it. This may be because it is less visual than Ladder, which may make it more difficult to interpret what the program is doing and what errors it is experiencing.

IL can make entering complex functions, such as PID loops and complex mathematical computations, a struggle. IL does not lend itself well to any form of structured programming, such as state programming or step ladder, further limiting its usefulness for implementing large programs. It is also arguable that the advantages of speed and compactness are less relevant, given the processing speeds of modern PLCs and the large amounts of memory available.

```
0005    Process1: BOOL;
0006    Drive_belt: BOOL;
0007    Process_code: INT;
0008    Process2: BOOL;
0001
0002
0003    IF (Process1 = TRUE AND start = TRUE OR manual_stir = TRUE)AND stir_complete= FALSE THEN
0004        stir := TRUE;
0005        Process_code :- 147;
0006    END_IF
0007
0008
0009    IF (Process2 = TRUE AND start = TRUE OR Manual_clean=TRUE)THEN
0010        CIP_P1:= TRUE;
0011        Process code:=247;
0012    END_IF
0013
0014    IF (Process3=TRUE AND start= TRUE OR manual_drain=TRUE) AND tank_empty= FALSE THEN
0015        Drain:- TRUE;
0016        Process_code:=047;
0017    END_IF
0018
0019
0020
0021
0022
```

Structured Text

With its IF...THEN loops, CASE selectors, and lines ending in semicolons, Structured Text (ST) closely resembles a high-level computer programming language such as Pascal and C. The aforementionedControl Engineering survey indicated that of all the IEC61131-defined programming languages, ST has seen the greatest increase in adoption.

Among IEC languages, ST perhaps best embraces the growing complexity of PLC programming, such as the process control functions involved in plastics or chemical manufacturing. Trigonometry, calculus, and data analysis can be implemented far more easily than in Ladder or IL. Decision loops and pointers (variables used to do indirect addressing) allow for a more compact program implementation than can be achieved in Ladder. The flexible ST editor that is common in most programming packages makes it easy to insert comments throughout a program, and to use indents and line spacing to emphasize related sections of code. This makes the task of structuring a complex program easier.

ST text-based, non-graphical nature, which is similar to IL, also runs much faster than Ladder. An additional ST benefit is that it comes closer than most of the other languages in achieving the transferability goals of the IEC61131 standard, emancipating a programmer from the hardware platform.

A final benefit is that many students currently graduating from engineering studies have a better background in computer languages than in the basics of electrical wiring, and therefore can more easily become proficient in ST than Ladder programming.

A disadvantage is that, for many previously experienced programmers or maintenance and service personnel, the ST environment is unfamiliar. Writing the code and structure to make it maintenance-friendly can reduce some of its compactness advantages.

As a result, control engineers tend to use ST 'behind the scenes." For example, IEC 61131 allows a programmer to build his or her own functions in one language, then insert them as sub-programs in another language. With this option, programmers often encapsulate an ST program inside an instruction, which is then embedded in a Ladder program.

Why Use Software Verification?

As control-system application programs become ever more complex, maintaining software quality during initial development and subsequent maintenance becomes more difficult. In a white paper available for download from Control Engineering 's online Resource Center, Paul Humphreys, a software engineer with LDRA Ltd. makes the case for extensive testing of control software, and explains the procedures world-class embedded control software developers use to ensure error free applications.

END

What is a Programmable Controller

What are programmable controllers and how do they work?

Programmable controllers are often defined as miniature industrial computers that contain hardware and software used to perform control functions. A controller consists of two basic sections: the central processing unit (CPU) and the input/output interface system. The CPU, which controls all system activity, can further be broken down into the processor and memory system. The input/output system is physically connected to field devices (e.g., switches, sensors, etc.) and provides the interface between the CPU and the information providers (inputs) and controllable devices (outputs). To operate, the CPU "reads" input data from connected field devices through the use of its input interfaces, and then "executes" or performs the control program that has been stored in its memory system. Programs are typically created in ladder logic, a language that closely resembles a relay-based wiring schematic, and are entered into the CPU's memory prior to operation. Finally, based on the program, the PLC "writes" or updates output devices via the output interfaces. This process, also known as scanning, typically continues in the same sequence without interruption, and changes only when a change is made to the control program.

Discrete applications

Programmable controllers are often used to control machines or processes that are sequential in nature, using "discrete" inputs and outputs that have defined states. For example, if a limit switch detects the presence of an object, it provides an "ON" signal to the PLC; if no object is detected, it provides an "OFF" signal. The machine or device typically performs actions based on time or events in a predefined order. The expected sequence is typically interrupted only when an abnormal condition occurs.

Process control applications

Programmable controllers can also control continuous processes that use analog I/O. For example, a temperature sensor may provide a variable signal, such as 0-10 volts, based on the measurement of an actual temperature. The controller program monitors the sensed values continuously and operates devices that may also be analog in nature. This could include setting the position of a valve between 0-100% open, or controlling the speed of a motor. Continuous applications are so called because they typically have no defined start or end once they are initiated; they maintain a process in a "steady" operating state.

Today's controllers

Initially, devices that exhibited the attributes discussed here were known as Programmable Logic Controllers (PLCs). This tended to emphasize that the main functionality of these systems was LOGIC operations. As technology has advanced, so have programming languages and communications capabilities, along with many other important features. These developments seemed to demand the definition of a new class of controller, the Programmable Automation Controller (PAC), which combines features of traditional PLCs with those of personal computers. In the past, size was typically used to categorize controllers, and was often an indication of the features and types of applications it would accommodate. Small, non-modular PLCs (also known as fixed I/O PLCs) generally have less memory and accommodate a small number of inputs and outputs in fixed configurations. Modular PLCs have bases or racks that allow installation of multiple I/O modules, and will accommodate more complex applications. With the emergence of PACs, functionality is the determining factor in categorizing controllers.

Which programmable controller is right for you?

Choosing the most effective controller for your application depends on a number of factors. To begin the selection process, a drawing of the machine or process is a good start. This can help identify field devices and physical requirements for hardware locations. From the drawing, you can determine how many analog and/or discrete devices you will have. Once the field device requirements and hardware locations are defined, you can review controllers that will meet your requirements. Download this Controller Selection Worksheet that will help you work through the considerations for determining the type of controller you will need, regardless of which manufacturers you are evaluating.

Application Brief DL06 PLC puts heaters to the test

Pyromatics Automation Systems of Crystal Lake, Il. was contracted by a customer to develop a Life Cycle Test Station for its electric heating elements.

This test station needed a user-friendly graphical interface to give operators the ability to select multiple ramp/soak parameters, output voltages, temperature sensor ypes, amperage ratings and total cycle

counts on tests for the cast-in electric heater platens. The system also needed to record temperature, volts, and current draw throughout the test for use in quality reports. Also, a failure of the heater required a safe shutdown of the test while alerting the quality department of the alarm condition.

Pyromatics selected the cost-effective DirectLOGIC® DL06 PLC as the heart of the system because of its ability to control up to eight PID loops and the multiple expansion slots available for thermocouple cards and analog input modules. It also controls two heaters, two chillers and an array of panel indicators, buttons, switches and relays.

A C-more 10-inch TFT touch-screen operator interface was used to provide operators with the necessary interface to operate and monitor the tests.

The completed system allows users to quickly connect the heater to be tested, enter test parameters, and run the test. Trend charts on the C-more panel track test parameters and quickly identify potential issues such as sudden drops in current or temperature.

Alarm reporting and history are also automatically recorded, allowing the operator to determine causes of failure. Data from the test can be easily uploaded to a USB thumb drive from the C-more panel. The data can then be imported into the user's choice of word processor or spreadsheet.

Semi cab sheeting production improved

ITS, a design build firm in Columbus, Ohio specializes in industrial automation. The company was contacted by a division of International Harvester responsible for the manufacturing of semi cabs. International Harvester uses automated machines to place aluminum rivets on sheeting that is attached to the frame of the semi cabs. The original CNC machines were becoming antiquated and needed to be upgraded.

ITS chose a DL205 PLC as the new controller for the machines, along with discrete I/O and an H2-CTRIO high-speed counter module that drives a dual axis servo. An H2-ECOM Ethernet Communications card links the machines back to an office for data acquisition. ITS also added a 15-inch touch screen for diagnostics.

In the new system, an operator stamps sheets of aluminum to welded framework with a handful of hand rivets and then places the product onto a dual axis servo table. After the operator selects one of five different parts programs, the machine will navigate the panel under the head assembly, which is responsible for the drilling and riveting, with a tolerance of 1/10 of a millimeter. The panel is drilled and a rivet is installed and squeezed to approximately 1200 PSI, producing a rivet consistency within .003 in. After completion of the panel (between 64 and 138 rivet locations), the machine will return to its home position and await the next product.

The solution increased productivity by approximately 30% and provides an easy way to run and maintain the machines.

Cost-effective I/O simplifies hydroelectric plant controls upgrade

Lockhart Power Company owns and operates a hydroelectric plant located on the Broad River in upstate South Carolina. The plant includes an 8-gate dam feeding a canal that channels the water flow to the powerhouse. The powerhouse contains five turbine generators with a combined power capacity of over 17 MW. The dam and turbine control system receives data from power, flow, and level sensing devices to perform monitoring and control of the dam, generators, and associated equipment.

Lockhart Power contracted North Fork Electric in Crumpler, NC, to lend their expertise to a renovation of the conrtol system.

The system consists of seven DirectLOGIC DL205 micro-modular PLCs with built in PID functionality. Each of the five systems for generator control includes discrete and analog I/O, and an Ethernet communications module. The remaining two PLCs are configured in a master/slave arrangement and control the dam gates, located upriver from the powerhouse, via radio modems. Operator interfaces include two 6-inch color touch screen panels and a Windows NT-based PC running the LookoutDirect SCADA/HMI software package.

In the automatic mode, the PLC can start, stop, and operate the generator, and control startup and synchronization of the turbine. Changing the generator gate position varies the flow of water to the turbine.

The dam control system controls the eight canal gates located at the dam, which regulate the flow of water downstream to the turbines.

For more information please visit: http://www.aboutplcs.com/

END