Alan LeVezu is the Western Regional Systems Manager for IDAC, Incorporated, Milwaukee, WI. LeVezu has been involved in the Installation and design of numerous Opto 22 and PLC installations, including solutions provided for Conner Peripherals, Micron, Toro, and Penn Racquet Sports.

PLCs, Ladder Logic, and Opto 22

In more and more applications, Opto 22 control systems are a better solution than Programmable Logic Controllers (PLCs) and ladder logic. Understanding why this is true requires an explanation. Perhaps the best place to start that explanation is with what PLCs and ladder logic do.

Originally, industrial automation was implemented with discrete relays and timers, which were connected together with copper wire. Using this type of automation structure had a serious disadvantage. To change the function (or logic) of the control system, the system had to be literally re-wired. Re-wiring is time-consuming and expensive. PLCs were invented to replace these banks of relays in semi-automated factories, notably automobile plants in the late 1960s. The programming method that most PLCs use is called ladder logic. Ladder logic is a software representation of the wiring diagrams used in the old days of discrete relays and timers (see Figure 1). Applications that closely resemble the discrete relay model can be controlled with a PLC programmed in ladder logic. When an application deviates from the relay model, programming in ladder logic becomes increasingly more difficult.

Basically, a PLC works by scanning all the input points, solving the user logic program, and then setting its outputs. Fundamentally, this is how PLCs operate.

The functionality of PLC architecture has extremely important implications for what PLCs can do and what they don't do so well. For example, PLCs are scanning devices and do not have event-driven capabilities like Opto 22 control systems. A scanning device runs a loop to check every I/O point over and over again. An event-driven device doesn't have to do anything until it is notified by an I/O point or other source that an action is needed. Scanning is natural in the ladder logic paradigm. Event-driven control isn't.

Understandably, PLC/ladder logic advocates like to use simple applications for comparison purposes. This is because as the complexity and size of the control problem increases and the PLC needs to communicate with the rest of the world, and above all, as the application must be documented, changed and maintained, PLCs become less and less suited to the job.

PLC supporters like to say that PLCs can do anything you can do with a more advanced control system. There is a sense in which this is true. After all, any statement in computer logic can be represented by a combination of on and off switches. But there's a reason why computer programmers don't work in machine language (0s and 1s) unless they absolutely have to. "Possible" and "a good idea" aren't always the same thing. "Possible" and "cost-effective" are often very different.

PLC manufacturers are not unaware of the limitations of PLCs and they have spent years inventing ways to get around them. However, all of them are to a greater or lesser extent makeshift add-ons and most of them require the PLC programmer to work harder and spend more time than on the equivalent solution from Opto 22.

Ladder logic was created to represent an electro-mechanical wiring diagram.

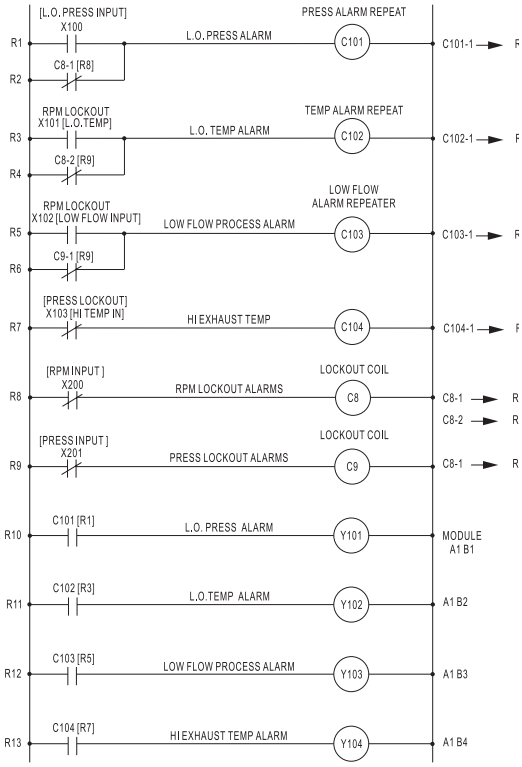

Figure 1

Apples, Oranges, and PLCs

Because PLCs and Opto 22 systems are so different, it is easy to be misled by comparisons. For example, a critical number for a PLC is speed in checking I/O points, often expressed in terms of the number of I/O points scanned per second. Some PLCs are very fast at this, but fast is not necessarily efficient.

With Opto 22's event-driven I/O and distributed intelligence, scan speed isn't an issue. The reaction times for Opto 22's intelligent I/O processors are typically measured in low millisecond time-frames. High-speed counting (20 kHz), pulse width measurement (100 microsecond resolution) and other time-critical operations are standard functions for digital I/O that are monitored and controlled by these I/O processors.

This distributed approach means Opto 22 controllers (the component most directly equivalent to the CPU of a PLC) don't have to waste processing power scanning I/O. They can be much more efficient at performing the other required tasks in the control system, while waiting to service an interrupt signal from the intelligent I/O processors.

Math Functions

Originally, PLCs had little or no math capability. Of course, today, no one would buy a controller that couldn't do even simple mathematics, so ladder logic languages have been retrofitted with at least simple mathematical functions.

Mathematical operations in a PLC are typically done using a very limited set of function blocks, and in many cases they can handle only integer values. If there is a need to do anything more than simple integer math, or a need to support floating point numbers, a PLC programmer is generally required to buy expensive add on cards that require programming in a different language and environment. These add-on cards are a lot of times only available for the high-end, high-priced PLC models. In a lot of PLCs, floating point numbers are not supported, period.

The Opto 22 systems, on the other hand, were designed from the ground up using a 32-bit architecture that easily handles the execution of both integer and floating-point math using an extensive library of math functions available in the OptoControl language.

Speed in processing mathematical functions is important because many modern control functions are inherently mathematical. One of the most important, the PID loop, is actually a fairly complex mathematical function. Yet PID loops are basic to rate controlling processes and control systems have to be able to do a lot of them quickly.

PID

PID handling provides us with a good illustration of the differences between the Opto 22 architecture and PLCs. Many PLCs handle PID loops but almost all perform their PID calculations in the CPU of the PLC. The calculations needed for multiple PID loops can quickly bog down the PLC. In an Opto 22 system, PID loops normally aren't even processed by the controller. They are done by the distributed I/O processors on the I/O units. This provides a system that expands painlessly because adding PID loops usually means adding I/O units. As the number of PID control loops in an Opto 22 system increases, there is no decline in performance.

To illustrate the difference in readability take a look at the ladder logic on the left. This is a small, yet standard approach with ladder logic to lockout alarming during startup of a pump. Compare that with the equivalent OptoControl logic on the right.

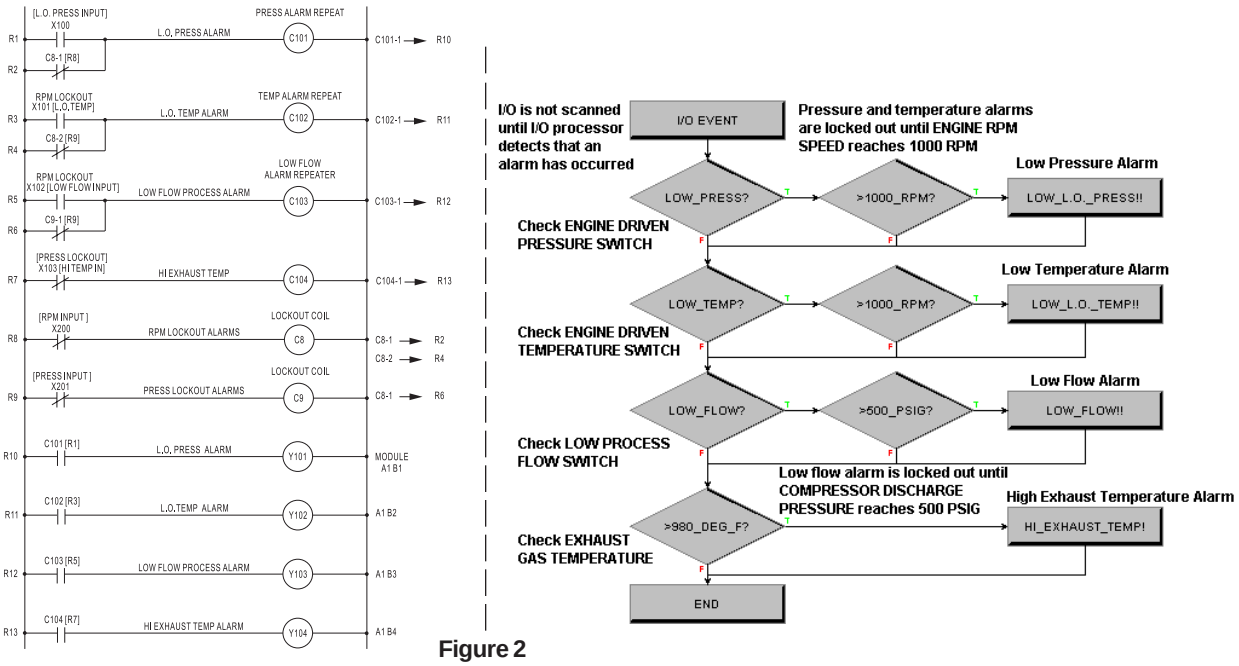

Figure 2

There are some fundamental architectural problems with PLCs. Because a PLC is constantly scanning all of its I/O looking for changed signals, and because almost all of the processing is done in the PLC, scan time is a critical measure of a PLCs ability to perform. Some PLCs are very fast, but even so, as the number of I/O points or PID loops increases, the system's ability to scan the I/O and resolve the logic diminishes. It is much easier to improve and expand a system that uses distributed intelligence, like Opto 22's, than it is to expand a PLC.

Scaleability

When expansion of a PLC system becomes necessary, there are often hard limits. When the system requirements outgrow the existing PLCs capability, expansion is not always an option. Sometimes in order to improve to a platform that provides more capability and processing power, replacement, rather than expansion, may be the only option. In this case, to program the replacement PLC you will probably need a change to a completely different programming environment and require that any existing code be manually translated into this new environment, even if the PLCs are from the same vendor. This could be a tremendous waste of time and money.

Because of the distributed and modular nature of the Opto 22 solution, it is much more difficult to overload an Opto 22 system. But should this happen, the options are much brighter. All the processors in the Opto 22 line are software-compatible. The software developed for one of Opto 22's lower-end controllers can be loaded directly and will run flawlessly in the high-end controllers.

Analog

The differences between PLCs and Opto 22 systems show up very quickly when dealing with analog signals. Many PLCs have to execute logic and instructions in the PLC's CPU to handle the conversion and scaling of analog values, from the raw counts it receives from the physical I/O, into meaningful scaled values (i.e., rate of change, gallons per minute, degrees Fahrenheit, or PSI).

With Opto 22's system, much of the overhead in handling analog signals can be done by the intelligent I/O processor on the I/O units. These processors can be calculating the PID outputs and off-loading time-critical I/O functions from the Opto 22 controllers. They also scale and linearize all the analog signals into meaningful engineering units so that when the values are read, no further conversion processing is required. In addition to the ramping of analog outputs, wave form generation and alarm monitoring are also handled at this atomic level. Because these functions are resident in the I/O unit, there is no degradation in performance as the system size increases.

Communications

Today, more than ever, efficient, well-designed control systems depend on communications to integrate the sharing of information between the various hardware components that make up a complete solution. Intelligent sensors and controllers with built-in serial communication capabilities are now commonplace on the factory floor. Serial communications are a problem for PLCs. PLCs can work together in a peer-to-peer network,

OptoControl was created to be an easy-to-use, easy-to-manage design and development environment for Total Control Solutions.

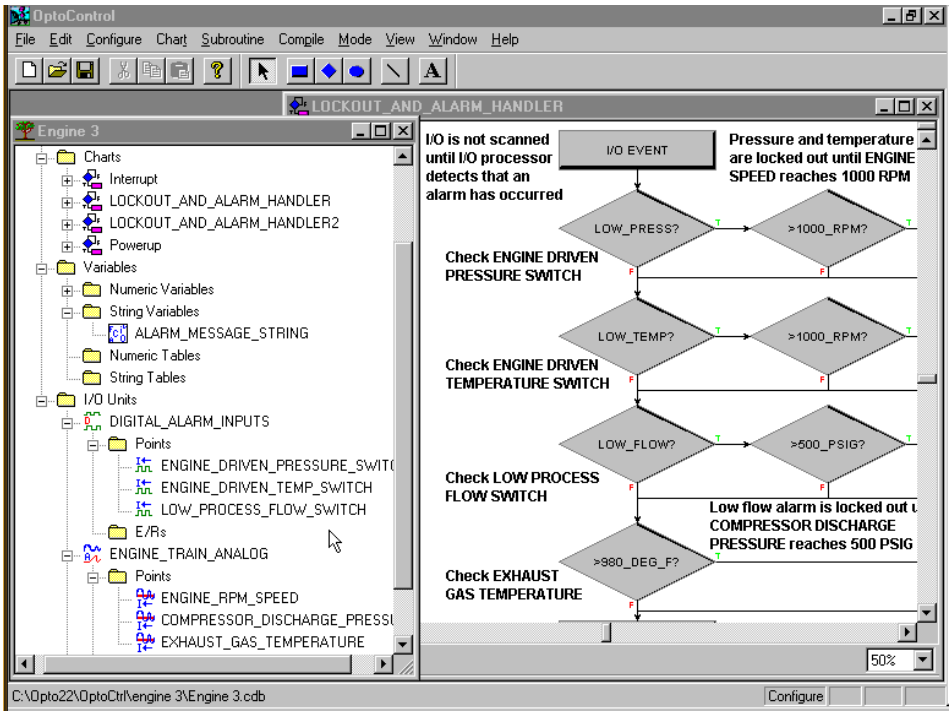

Figure 3

essentially by splitting the ladder logic program up and working as one big PLC. They don't communicate very well with other third-party devices.

How do PLCs handle serial communication? They don't, exactly. To handle serial communications, the PLC user usually has to purchase additional, expensive, auxiliary processors or special-purpose hardware. Basically, an auxiliary processor module has to be added to the PLC rack.

There are a number of problems with add-on approaches from PLC vendors . First, it adds complexity and cost to the PLC. Another problem is that you generally don't program the communication processor in ladder logic. You have to use a second, higher-level, language. That means that to effectively use a PLC in any kind of communicating environment you need to know at least two languages - one Ladder logic languages have evolved somewhat from simple wiring diagrams, but fundamentally they have serious limitations as programming languages.

Programming

PLC makers have tried to improve the situation by offering more "advanced" languages based on concepts like flowcharts. But flowcharts are an inherently uneasy fit with a PLC. These languages generally compile into ladder logic, so even though the program may now be easier to write, there are still functional limitations because the hardware platform was designed to run ladder logic. To add features like ASCII communication capability, or complex math functions to ladder logic, completely different hardware and programming environments have to be implemented and integrated to support functions the ladder logic cannot perform. While this will allow these functions to be performed, it is indirect and inefficient. The link between the two environments is additional overhead that can degrade the overall system performance. In addition, the task of documenting two or more separate programs in an overall system, strategy becomes much more complex and therefore harder to understand. This is important because programming and maintaining programs are among the major costs of control systems today. A system that is hard to program, understand, or document costs extra time and money.

In contrast, Opto 22's programming environment leverages the graphical interface provided by the Microsoft Windows 32-bit platform to present a clear, concise representation of your entire control solution within a single well-documented setting. The development environment streamlines code development, makes the logic easy to understand and leverages the power of Opto 22's distributed hardware platform.

One of the fundamental advantages of Opto 22's OptoControl programming software is readability. It is inherent to the language. Six months after you write an OptoControl program, you can come back to it and understand it. OptoControl is virtually self-documenting. Even if the program has been changed, the documentation will reflect the latest changes.

When programming in the Opto 22 environment, information regarding the project is easy to find and easy to understand. The entire system configuration is graphically presented in the "Strategy Tree." The "Strategy Tree" is a hierarchical treelike representation of the Opto 22 control system: controllers, I/O points, variables and even the logic itself are represented. Icons are used to distinguish the various components that appear in the OptoControl strategy (see Figure 3). The control logic is developed and represented in flowcharts that provide a precise visual illustration of how the process flows.

OptoControl's configurator provides an intuitive user interface that graphically documents the control strategy. The 32-bit Windows environment makes it easy for a programmer to simply point and click to display or edit system configuration or program logic. The flowcharts are created using a very simple set of drawing tools. Instruction insertion and editing is accomplished through the use of standard pull-downs, dialogs, and controls that are now a familiar part of all Windows-based software packages.

OptoControl's integrated real-time debugger makes it easy to follow the flow of control logic and understand what is happening. The debugger presents a tightly-integrated user interface that allows the programmer to enter break-points, do single-step functions and manipulate I/O points or variables in the program. This mode of the OptoControl environment speeds checkout of the entire system. It is also a very powerful diagnostic and troubleshooting tool for maintenance personnel after the project is in production.

Cost

Another popular comparison among PLC supporters is the cost of a particular piece of hardware. This cost comparis on falls apart when you talk about the overall cost of a system, which is the life cycle cost. Life cycle cost is the total cost to purchase, install, program, maintain, upgrade, and operate a control system over its life in the enterprise. In life cycle costs, Opto 22 systems are nearly always less expensive than PLCs and ladder logic, particularly since the costs of programming, expanding, and maintaining the system are so much less. The Opto 22 system performs all the functionality mentioned for math, PID, analog, and serial communication with the standard hardware. The software supports all these functions without the need to add co-processors or special function hardware. In conclusion, the Opto 22 distributed approach generally performs at a higher level and has more capability than an equivalently-priced PLC system.