PLCs, PCs, and PACs:

When the Lines in Motion Control Become Blurred.

ENGINEERING YOUR SUCCESS.

PLCs, PCs, and PACs:

Asking the right questions early simplifies optimizing your next motion control system.

Historically, motion controllers, programmable logic controllers (PLCs), and industrial personal computers (PCs), which have clearly defined functions in a control system, were separate components. With the rise of programmable automation controllers (PACs), motion controllers are increasingly difficult to distinguish from PLCs. Programmers are building custom applications on PCs to create decentralized control schemes that command a wide array of sub-control devices, including motion controllers, drives, vision systems, etc. The trend of merging traditionally separate control components can add confusion and complexity to the task of designing a new machine or expanding the functionality of an existing one. With a bit of knowledge of the different control architectures and knowing the right questions to ask, designers can quickly identify which control scheme will be best for their application. Before starting the controller selection process, it is important to understand what the different options are and why they are used.

Image on front page: A programmable logic controller (PLC) is an industrial solid-state computer that monitors inputs and outputs, and makes logic-based decisions for automated processes or machines. PLCs were designed to replace relays, timers and I/O.

Jim Wiley *is a product manager for servo and stepper drives with Parker Hannifin's Electromechanical and Drives Division, where he began his motion control career in 1996. After graduating from Carnegie Mellon University with a bachelor's degree in mechanical engineering, Jim joined Parker as an application engineer and has served a variety of marketing, product management, and application engineering roles across the entire range of electromechanical technologies.*

Marissa K. Tucker *is the product manager for control and HMI products with Parker Hannifin's Electromechanical and Drives Division, where her duties include creating alignment between key market segments, new product development, and product messaging. Marissa received her bachelor's degree in mechanical engineering from the University of California—Davis and continues to work directly in the market to produce control and HMI products that focus on both capability and ease of use.*

THE DIFFERENCE BETWEEN PLCS, PCS, AND PACS

Essentially, a PLC is a ruggedized control device made up of a microprocessor, memory, and a variety of peripherals. PLCs typically use IEC 61131-3, an industry-standardized set of programming languages, including Ladder Diagram. Ladder logic, a language that reads the same as the electrical diagrams maintenance crews are already familiar with, makes the PLC a popular choice. Most developers and maintenance personnel have experience with programming and debugging Ladder, minimizing the need for training. Standardized programming ensures longevity as the machine can easily be serviced in the future, and reduces the dependency on the original programmer. The major limitation with PLCs is that they were designed to replace relays, timers, and I/O. This left their functionality limited when it came to the realm of motion control and visualization.

Personal computers such as Parker's Industrial PC PowerStation allow users to develop their own application to control multiple sub-devices such as controllers and visualization systems.

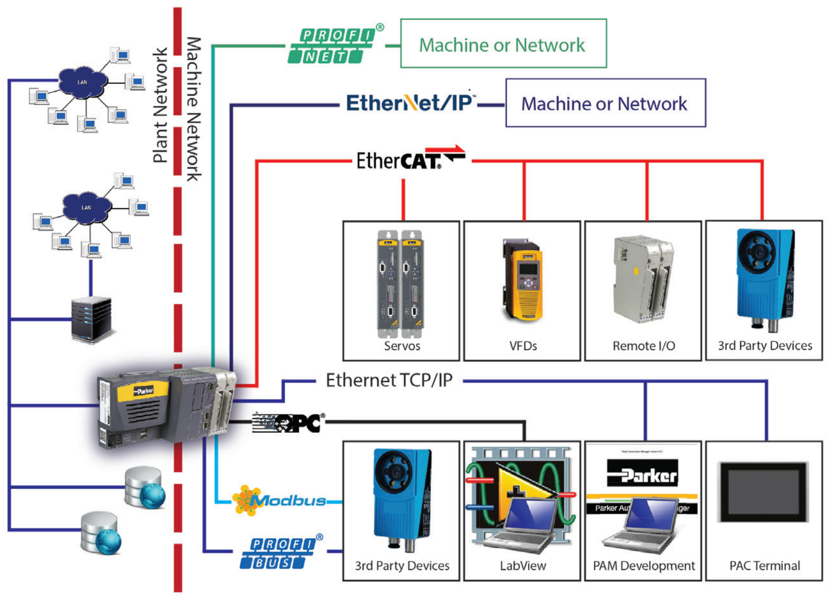
The Parker Automation Controller is an example of a programmable automation controller (PAC). PACs provide the ability for users to develop their own drivers to connect to unique devices using ASCH. Like a PC, it has visualization capabilities built in, allowing the user to develop a complete application that incorporates programming the logic and the human machine interface all in one software suite.

Traditional PLCs typically rely on peripheral devices such as smart drives and stand-alone motion controllers to provide advanced functionality. A potential drawback of this motion-control architecture is the need to maintain separate programs for each device. Smart drives and stand-alone motion controllers often use proprietary languages, canceling out the benefit of using an IEC 61131-3 PLC in the first place.

Future maintenance in this type of control scheme is incredibly difficult because it is not always obvious what is being controlled by the PLC and what is being controlled by the motion controller. In the absence of proper documentation, understanding the machine often involves opening up the cabinet and using a multi-meter to trace connections or directly connecting to the peripheral motion systems via a laptop. If the repair person is not familiar with the proprietary language used by the motion controller, the diagnostics process could lead to excess downtime and increased expenses.

Industrial PCs, first introduced in the mid-1980s, are high-reliability computers with hardware and operating systems engineered to withstand the constant vibrations, temperature extremes, and wet or dusty conditions common in industrial environments. Industrial PCs are most powerful to developers who are comfortable programming their own custom applications using either Visual Basic, C#, C++, etc. Using an industrial PC increases flexibility, giving users the freedom to communicate to any device using either pre-built APIs or by writing their own communication drivers. This freedom leads to the creation of novel applications using smart subsystems that may not have been intended or initially designed to work together. Stand-alone motion controllers, in the form of smart drives or multi-axis, are examples of smart subsystems. Motion-controller manufacturers typically provide an API that allows the developer to send motion commands to the controller—limiting the need to learn a full, separate language. Alternatively, some developers will choose to leverage the real-time capabilities of the motion system and program the device in its native, embedded language, with the PC application calling for these complex routines to run when needed.

Beyond increased flexibility, these applications have several benefits over traditional PLC systems. The HMI (human machine interface) is built right into the control application itself, reducing the need for additional devices for



In response to the demand for more connectivity options, today's motion controllers increasingly offer support for multiple communication protocols. For example, a modern PAC controller provides EtherCAT communication for real-time motion, I/O, snf third-party device connectivity as well as EtherNet/IP, PROFINET, and an OPC Server for machine-to-machine and plant level communications.

visualization. In addition, a single programming language can be used to control all subsystems. This single PLC application can also be a major downfall as the machine ages. As an organization matures or technology changes, preferred programming languages may shift, making it difficult to develop, change, or maintain older applications. In addition, API is not a set standardized by any organization, so migration to a new language or OS may not be possible, even when using the same subsystems.

Motion controllers offer designers highly specialized functionality for controlling and coordinating the movement of motors within a machine. A range of form factors are available as motion-control providers have developed solutions based on smart drives, PCI cards, Ethernet, and just about every field bus ever created. Centralized or distributed solutions offer machine designers nearly endless possibilities for crafting a system that best fits their needs in terms of performance, size, and cost. In general, motion controllers rely on a proprietary language that is tailored to fit motion commands. Most motion controllers have also evolved to incorporate some of the machine-control functions usually associated with PLCs, such as temperature monitoring and discrete I/O control.

Due to the proprietary nature of motion languages, machines designed around a specific controller may include advanced functionality but can sometimes be limited when it comes to expansion. Single-PCB motion controllers cannot easily add additional axes of motion control without ordering a new unit from the factory. Bus-based motion controllers have more flexibility to add axes, but it is important to ensure that the bus is both widely offered by other device makers and increasing in market adoption.

PACs merge PLCs, PCs, and motion controllers into a single device. Rather than requiring a separate stand-alone motion controller, PACs provide multi-axis motion trajectories over a bus—such as EtherCAT—while drives close the local PID loop around the motor. This architecture not only allows the entire system to be programmed with IEC 61131-3, but also within a singular development environment—reaping all the benefits of standardized programming. Maintenance is significantly reduced as the PAC queries the drives to determine the failure mode. Rather than needing to open up a machine to gain access to data, either through manual multi-meter readings or direct connection to sub-devices, all the information can be accessed by connecting to a single PAC. When choosing a PAC, it is important to select a bus system that will allow flexibility when choosing devices as well as withstand the test of time.

Connecting to other machines on the factory floor and integrating internal drives are important considerations of machine design and controller selection.

A PAC controller that supports EtherCAT as its main field bus but also offers support for EtherNet/IP, OPC client/server, Modbus TCP, PROFINET, and PROFIBUS, ensures that the controller is “future-proof” and also compatible with current industrial devices.

UNIVERSAL APPLICATION QUESTIONS

It is critical to weigh a variety of considerations to avoid ending up with a less-than-optimal motion control solution. By asking the right questions before beginning the specification process, the designer can avoid making the wrong control choice:

How is the application likely to evolve over the next 20-25 years? Consider what new functionality and subsystems may be required. Take the time to assess whether the solutions you are considering have the flexibility to readily integrate new devices or subsystems.

Will the system require a centralized, deterministic control scheme? This is common in industrial applications where a single PAC is in control of an entire system. Or does the design require a combination of multiple, decentralized smart devices, such as optical lab instruments, in addition to motion control?

What communication protocols offer the greatest flexibility and longevity? There are so many choices. It is important not only to select a bus that works best for your system (for example, EtherCAT is best for high-speed motion control) but also a bus that is proven, widely used, and growing in installation.

How will space constraints dictate system architecture and component choices? Must the system be compact enough to sit on a benchtop or can it span many meters? For instance, Ethernet-based bus systems can transmit data over extended distances, whereas traditional motion controllers are limited by the quality of digital and analog signals and restricted smaller ranges.

What existing integration and programming resources are available? Many organizations are reluctant to take the time to acquire a new skill set and third-party services may be used in the future to maintain a machine. The choice of programming language is critical in determining how quickly an organization or maintenance crew can diagnose, change, and develop an application.

Designers who do not take the time to determine the best control scheme or choose components too quickly without asking these critical questions have the potential for serious consequences further down the road. It is important to know and avoid these common design mistakes:

Choosing a device without application consideration. When a controller is selected first, with insufficient regard for the application, it constrains the designer's choices, leading to longer and more expensive development cycles. The controller is often selected first when the designer either defaults to a controller he or she is familiar with or "falls in love" with a controller gussied up with all the latest bells and whistles, forcing the designer to use components that may not be ideal for the application because they work with the controller selected. As a result, the designer may need to find work-arounds to get a system to operate correctly, increasing development time. Similar problems can arise when a designer does not take the time to understand all that the application entails.

No place to grow. When new functionality requirements emerge, the wrong controller can make it difficult to expand or extend the system. Without careful consideration of how a motion-control system is likely to evolve over its lifetime, it is far too easy to select a controller (such as a traditional controller based on I/O) with limited ability to accommodate new devices or functionality.

Invest for the future. Selecting an inadequate controller for a given application to save a little money or failing to plan for future necessary expansions of an application all but ensures a less-than-optimal return on investment. Selecting the wrong controller in the early stages of system development will demand additional design time and could force the designer to employ less efficient components to allow the poorly chosen controller to work. As the machine matures and requires maintenance, it may be difficult or impossible to keep it running, and even more so if the programming language used was proprietary or no longer commonly used, and even more so if the original designer has moved on to another organization. If expansion is required to add a feature or device to extend the system's lifespan and usability, but the control bus used is no longer available, the system may have to be redesigned from scratch instead. This can cost significant development time and resources.

All too often in today's quarterly bottom-line-driven business environment, designers are pushed to design the least expensive solution that will serve the application right now. Asking honest questions and answering them fully is the best way to ensure that a new motion control system can continue to evolve along with the application's changing requirements, and continue to provide a return on investment for many years to come.

Parker Hannifin Corporation
Electromechanical & Drives Division
9225 Forsyth Park Dr.
Charlotte, NC 28273
phone **800-358-9070**
fax **707-584-8015**
www.parker.com