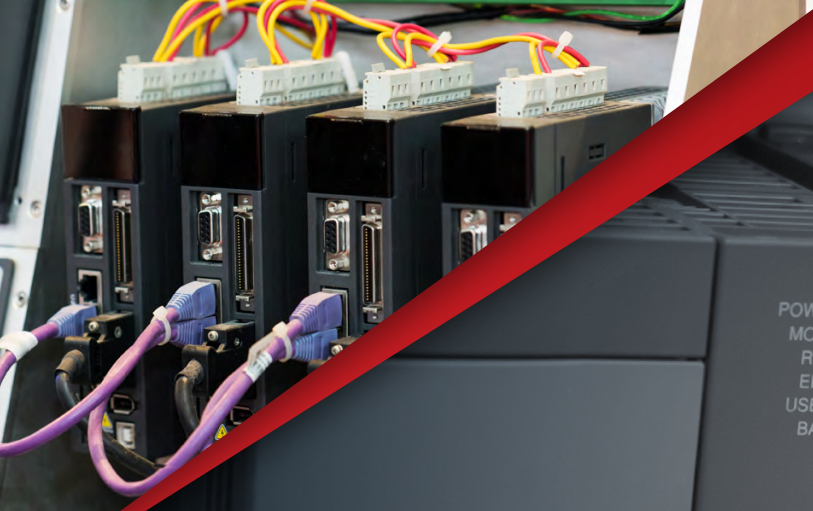
APRIL 2024



TECH TOOLBOX

PROGRAMMABLE LOGIC CONTROLLERS (PLCs)

SPONSORED BY:

MITSUBISHI ELECTRIC
Changes for the Better

Automating the World

WAGO

TECH TOOLBOX

TABLE OF CONTENTS

Brought to you by:

Design World

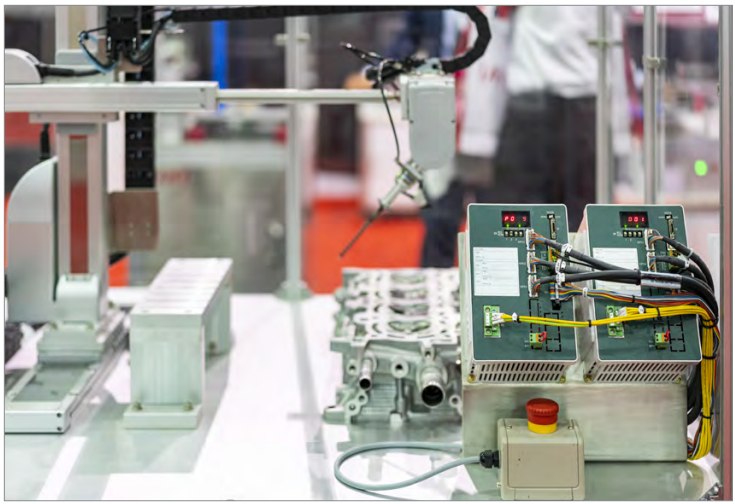
(888) 543-2447 • www.DesignWorld.com • marketing.wtwhmedia.com

Introduction

Programmable logic controllers or PLCs are microprocessor-based components that serve as the programmable smarts for simple or isolated applications. They're also finding increasing use on integrated machinery and automated installations with extensive IIoT connectivity. In this new Tech Toolbox, the editors of Design World detail how PLCs came to replace relay-based controls as well as which PLC logic and memory board arrangements are most common today, as well as backplane interfaces to I/O modules and other circuitry.

Basics of PLCs

Programmable logic controllers or PLCs are highly specialized, programmable microprocessor-based controllers used to control a specific application on a machine or a process. They are used in automation and manufacturing to control assembly lines and machinery on factory floors as well as many other types of mechanical, electrical, and electronic equipment in a plant. They can be used in applications ranging from vending machine controls and packaging machinery to roller coasters and complex camera positioning systems.



Today's PLCs are powerful and flexible products that support interconnected control of automated installations. Image: Dreamstime

The earliest PLCs came along to replace large cabinets of electromechanical relays which were used to control machine processes. Cabinet-based control systems were big and bulky and required an enormous amount of wiring. And when a change in the control system was desired, the only way to do so was to go into the cabinet and rewire connections, which was costly and took up a lot of time.

The introduction of PLCs meant that the same control functions could be accomplished in a fraction of the space of a traditional control cabinet with far less wiring and complexity. Also, if a change in control was desired, the change could be handled via simple ladder-logic programming, eliminating the need to rewire large portions of a control cabinet.

The basic parts of any PLC system include the processor, I/O modules to handle inputs to the controller and outputs to the controlled devices, and some type of user interface ... with the latter as simple as a keypad or a touchscreen interface or a programming link via a PC. The PLC's processor is programmed via the user interface. The I/O modules are used to bring input signals into the PLC's CPU and output control signals to controlled devices such as motors, valves, sensors, and actuators ... among others.

Individual PLC manufacturers may have different programming languages, but most are still based on the fundamental ladder logic programming structure. Although now, they can be programmed in higher-level languages such as C and BASIC. Communication options may include simple RS-232 serial communications to more advanced Ethernet protocols.

One of the most important considerations for any PLC is scan time. Basically, this is the time in which the PLC runs through the program taking in data and updating outputs. This is typically a few milliseconds but can be much longer depending on the program length and the speed of the processor. Higher scan times can accommodate processes with more realtime demands than traditional slower applications where scan speed is not as critical.

Other points to consider include the I/O count and the ability to expand I/O if needed. This can range from a handful of I/O for simple applications to hundreds in more complex machinery. The type of I/O, whether analog or digital, is also important. It's common for manufacturers to supply modules designed for unique input types, such as counting encoder inputs.

Another point to consider is the level of customization needed. For instance, PLCs can be set up with individual web pages where operators can go to view system parameters and machine conditions. They can also be programmed to send alerts and condition status updates to mobile devices such as cell phones to keep operators and engineers abreast of machinery and processes.

Image: Dreamstime

More recently, some manufacturers have introduced so-called PACs (programmable automation controllers). A PAC is similar to a PLC but denotes a controller that accommodates better realtime control needed in some automation applications. Also, special PLCs can include dedicated safety PLCs which monitor machine inputs such as photoelectric sensors, light curtains, magnetically operated sensors, emergency stop buttons, and safety mats.

Native I/O allows PLCs to handle signals from myriad field components.

Image: Dreamstime

PLCs in discrete automation and process automation

PLCs are at the core of myriad automated discrete and process installations. In discrete automation PLCs command the sequencing and coordination of assembly, sorting, and inspection tasks; control the routines associated with filling, sealing, labeling, and conveying discrete (separate) packages; and run robotics through routines to handle separate workpieces or other items with machine tending and pick-and-place motions. PLCs might also control tasks within a given machine – such as cutting, milling, drilling, and shaping as in machine-tool equipment, for example.

Shown here is a typical PLC installation on DIN rail. Image: Dreamstime

In contrast, process automation employs PLCs for the command of operations on continuous flows of materials ... often involving chemical reactions, the management of boiler and tank contents, filtering procedures, and other unstoppable procedures that follow interlinked recipes. Such processes define the operations of refineries and pipelines; chemical production and wastewater treatment; and power generation.

Industries including the pharmaceutical and food and beverage industries use PLCs for both process-related tasks (such as brewing, blending, mixing, separating, reacting, pasteurizing, baking, and solidifying) as well as discrete automation tasks such as dosing, dispensing, bottle filling, packaging, and tracking.

More information on PLCs for automation

PC-based and dedicated PLC motion control
PLCs versus programmable controllers (PACs)
PLC I/O, visual display, and networking options
Update on HMI-PLC combination hardware
Using stepper drives with stepper controls on PLCs

Image: Dreamstime

The evolution of PLCs

Image: Dreamstime

Recent decades have spurred the miniaturization of PLC hardware concurrent with dramatic increases in processing capabilities and connectivity. So, today's PLCs are now compact enough to install in places impossible before to give design engineers more options for system design. Following Moore's law, evermore powerful processors let PLCs execute complex algorithms (and handle ever-larger data sets to support big-data end uses) faster than ever. Support of various industrial protocols allows communication with human-machine interfaces or HMIs, supervisory control and data acquisition (SCADA) systems, and enterprise-level networks.

Side note on

soft PLCs

Software-based soft PLCs are essentially control programs that run on industrial computers (IPCs) or programmable automation controllers instead of dedicated PLC hardware. Soft PLCs allow programming by ladder logic or structured text as well as C/C++ or even Python and other high-level languages even while offering compatibility with common IPC operating systems. The design can simplify integration with other software; enable the simultaneous execution of advanced tasks; and enhance communication via standard Ethernet hardware. By leveraging the computing power of today's industrial computers, PLC programs can also run advanced controls faster and process larger data sets than legacy PLC systems. What's more, soft PLCs are inherently scalable with various modules and design-library elements to accommodate the expansion of system architectures as needed. All this said, traditional PLC installations are in many instances more reliable, cost effective, and accepted by industry than soft PLCs.

PLC advances have also taken the form of easier and more powerful programming. Going beyond traditional ladder logic, these new options are flexible and standardized (especially when adherent to IEC 61131-3) for a common framework that supports the development and maintenance of code. Dazzlingly easy-to-use development software with graphical user interfaces (GUIs) provide an intuitive environment for programming, debugging and simulating tasks.

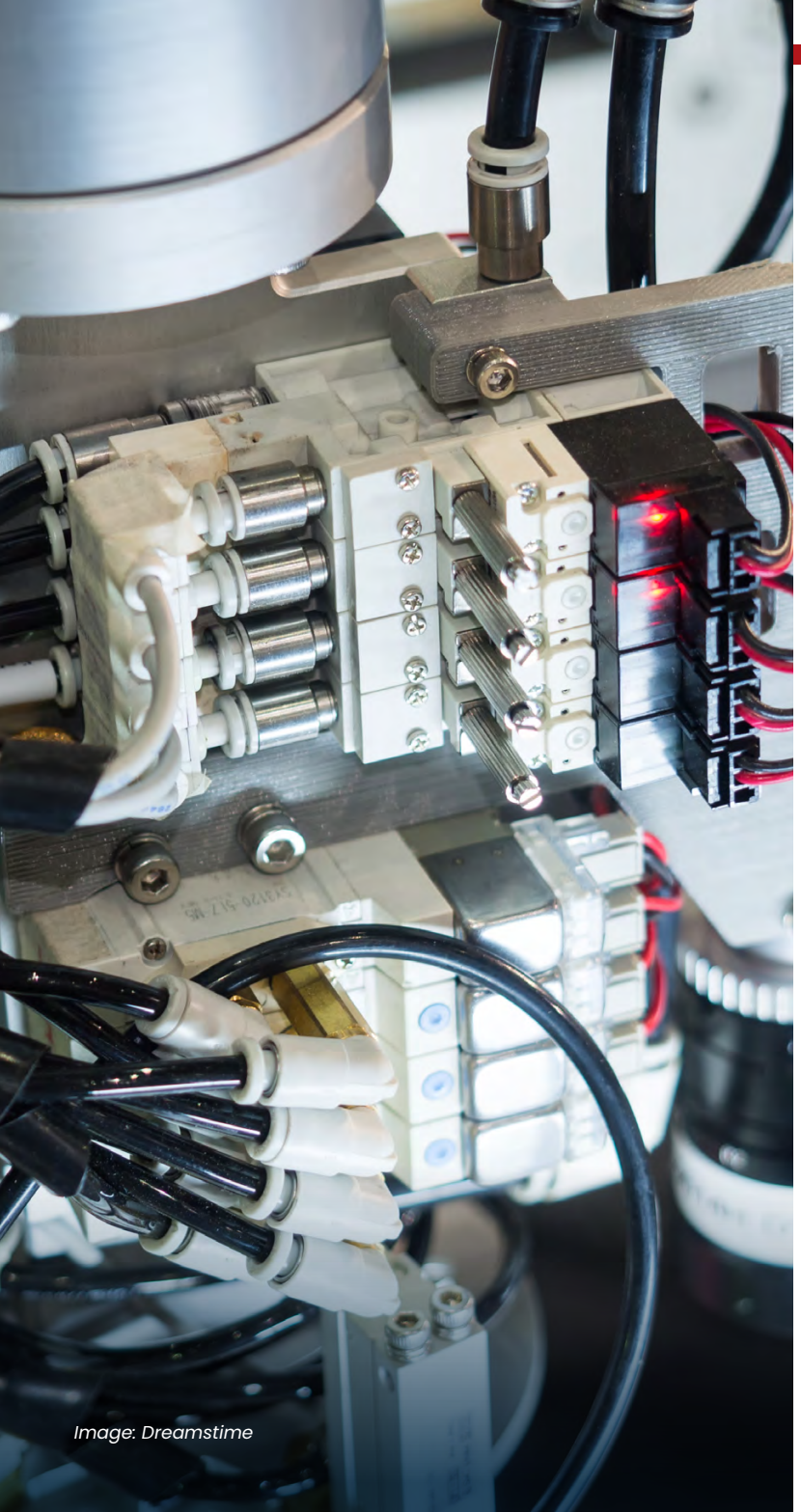


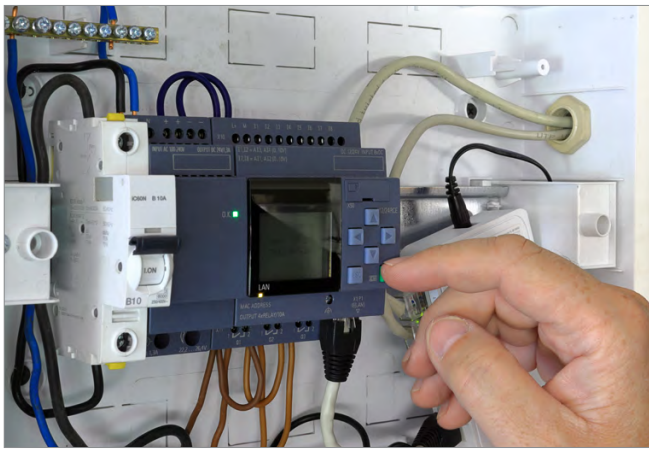
Image: Dreamstime

In fact, PLCs today also impart advanced controls, diagnostics, analytics, and safety to designs impossible in the past. Proportional-integral-derivative, fuzzy-logic, machine-learning, and model-predictive controls are often complemented by system monitoring and fault detection that (with troubleshooting support) maximize productivity with minimal downtime. Where applicable, PLC safety functions complete designs with IEC-standards-adherent implementation of worker safeguards and emergency stops.

I/Os and communications on PLCs

A major hardware cost in control systems is that for inputs and outputs. Devising the most suitable arrangement of I/O takes creation of an I/O map. Then grouping I/Os by location helps engineers identify packaging needed to deploy hardware correctly. One caveat: Unless one is using new breeds of design software, mapping and grouping the design of I/Os doesn't generate a BOM for purchasing, so be careful to make one upfront.

In addition, sometimes the cost of deploying I/O hardware is greater than the cost of the hardware itself. That's because the cost for I/O has continually declined over time. PLC and I/O makers have advanced the technology and created ever-denser packaging. What's more, the ease of servicing for the highest-density modules may make the cost tradeoff an unimportant issue.



Some micro PLCs sport information displays with the ability to process hundreds of function blocks … accepting 12/24 Vdc and offering digital inputs that in some cases also work in analog mode (at 0 to 10 V, for example). Programmable via an Ethernet interface, some such PLCs can communicate with similarly outfitted modules. Integral web-server applications allow the use of user-defined web pages (for system visualization) and IIoT connectivity for monitoring system status on smartphones and other mobile devices. Image: Dreamstime

PLCs are used extensively in process-type automation facilities. Image: Dreamstime

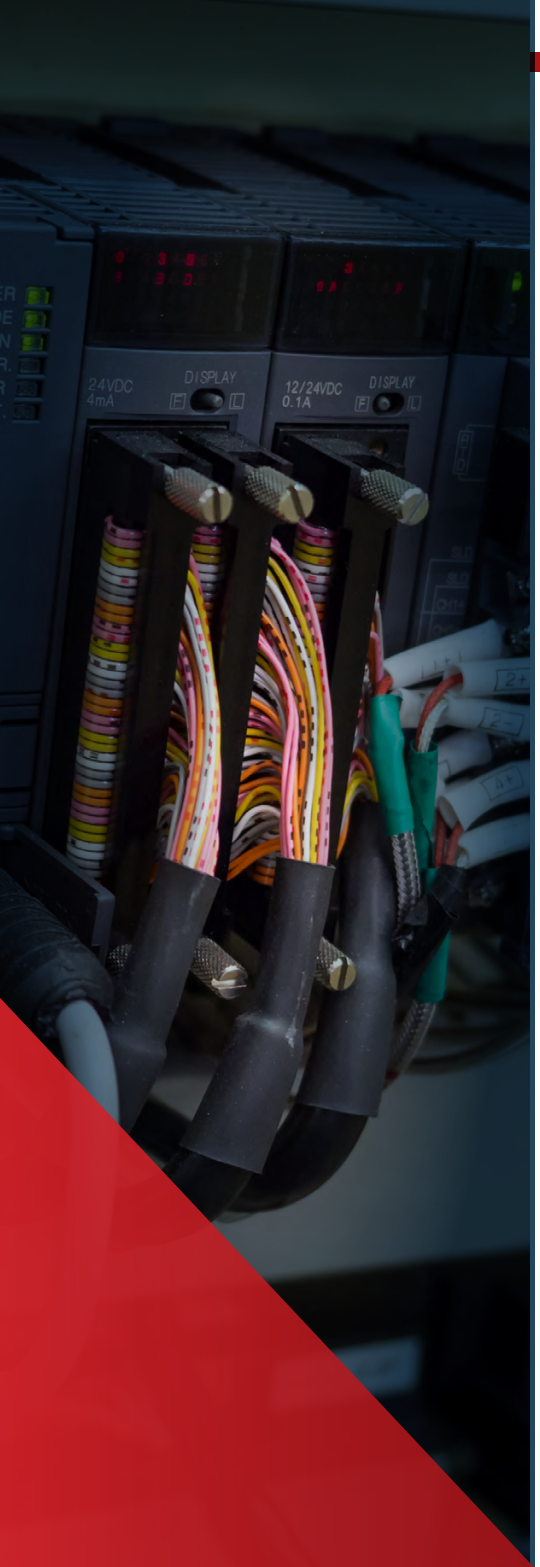
The cost of packaging I/O for metal enclosures has increased, so more vendors than ever now offer package options suited to difficult environments – with some I/O suppliers selling optically isolated modules for explosion-proof designs, for example. This level of packaging and relatively low pricing make fully distributed I/O feasible for many applications.

Today, PLCs even leverage I/O and connectivity to offer industrial internet of things (IIoT) functions such as cloud-system data exchanges; edge computing; and remote monitoring. The shift from proprietary to open Ethernet and fieldbus protocols (including Ethernet/IP, PROFINET, and Modbus TCP) has made device and cross-platform integration easier. Object linking and embedding or OLE for process control (OPC) extends these PLC capabilities to connect with SCADA and manufacturing execution systems or MESs.

Ethernet for I/O leads the way

Control system technologies (including PLCs) are now dominated by Ethernet-based standards for all system-interface types. Small Ethernet gateway modules enable highly distributed control systems with low-cost wiring and connectors and increasingly higher speed. Some vendors even support Ethernet with Precision Time Protocol, also known as IEEE1588. In large material-handling systems such as automated warehouses and airport baggage-handling centers, Ethernet offers solid communication speed, cost, and simplicity.

The other big advantage of Ethernet for I/O networks is the ability to mix and match I/O products from multiple vendors. List all the I/O requirements by group such as digital I/O and analog I/O.



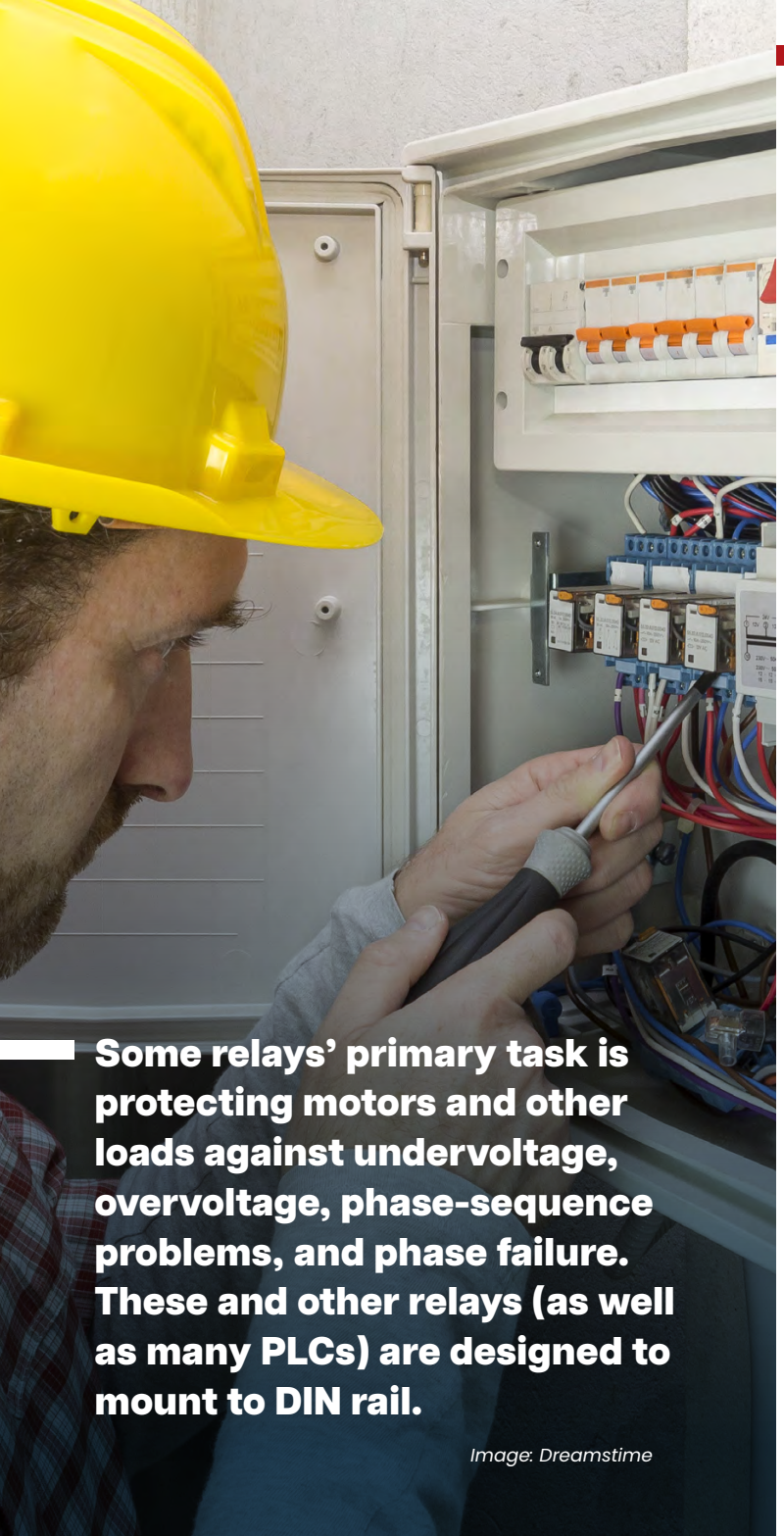
For each category, list required specifications – remembering that high-analog resolution comes at a premium. Here it's best to know what resolution and how many points an application needs. For discrete I/O, be sure to separate inputs and outputs and low power from high power. Ensure power loads and circuit protection are appropriately used. Otherwise, the system will have trouble. Case in point: Solenoids exhibit field-collapse spikes that can (without proper suppression) destroy I/O transistors.

There are myriad I/O options that use epoxy-potted I/O termination blocks, overmolded cordsets, and multi-pin connectors. I/O modules incorporating these forms are well engineered and withstand moisture, washdown, and aggressive chemicals. Many such modules also use LED annunciators to simplify diagnostics. Upfront cost may be higher, but these systems setup quickly.

Connectivity specific to PLC hardware

On PLCs, RS-232 and RS-485 serial ports (complemented by serial protocols such as Modbus RTU and ASCII) assume common communication functions. Some PLCs also have expansion slots and modules to let machine builders add communications specific to a given use. Then Ethernet-based connections assume networking to motors, drives, sensors, and other controllers. PLC installations dominated by serial and Ethernet-based expansion I/Os remotely located (for wider physical reach with less wiring albeit at slightly lower response rates) are classified as distributed controls. In contrast, those with only natively addressable I/O (on the PLC hardware) or local I/O installed adjacent to the PLC and simply employed to enlarge the controller's domain (and with uncompromised speeds refreshed every scan) could be typically classified as traditional centralized control.

PLC discrete I/O allows the connection of limit switches, lights, and other components that produce ac and dc signals of various voltages and currents to facilitate PLC monitoring and control functions. In contrast, PLC analog I/O allows the connection of motor-speed controls, thermocouples, and other components that employ continuously variable voltage or current signals.



Some relays' primary task is protecting motors and other loads against undervoltage, overvoltage, phase-sequence problems, and phase failure. These and other relays (as well as many PLCs) are designed to mount to DIN rail.

Image: Dreamstime

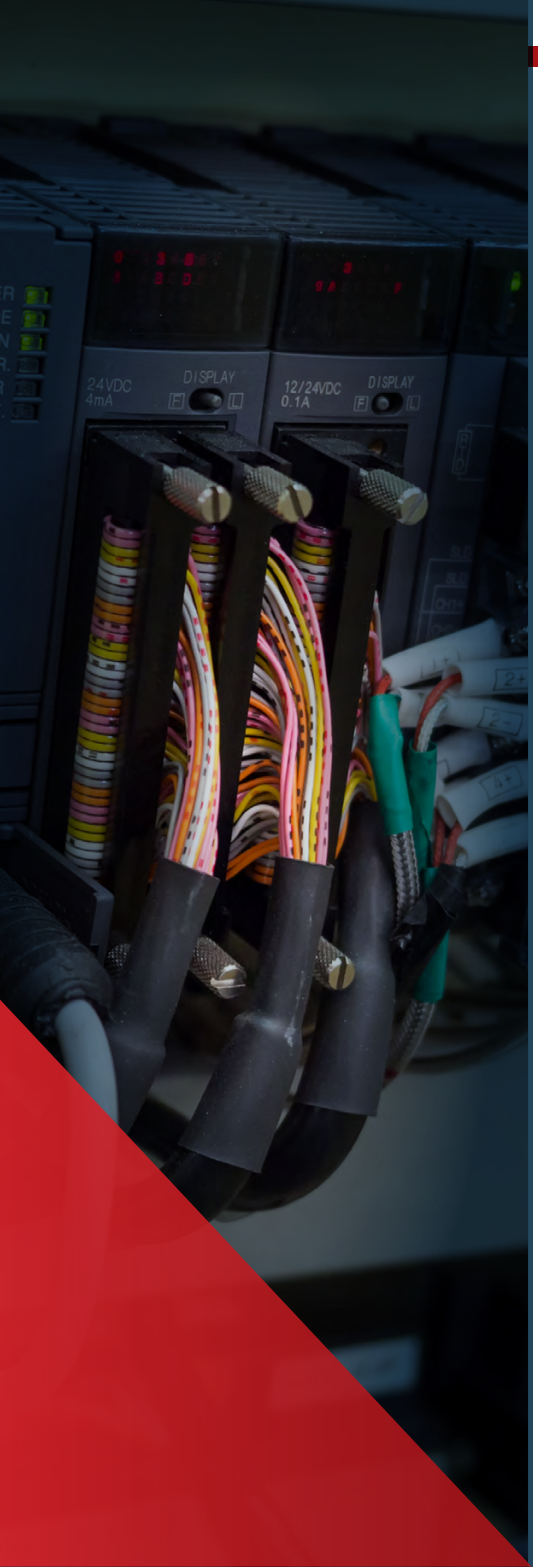
To be clear, the hardware connections outlined here only work when PLC software also supports the protocols required. For example, PLCs must support SMT for email communications and the HTTP application layer protocol for internet connectivity; MQTT for machine to machine (IoT) messaging; FTP for server-client data sharing; and one or more of the industrial communication protocols to make use of the serial and Ethernet-based connections for most automation installations.

Choosing between (or combining) relays and PLCs

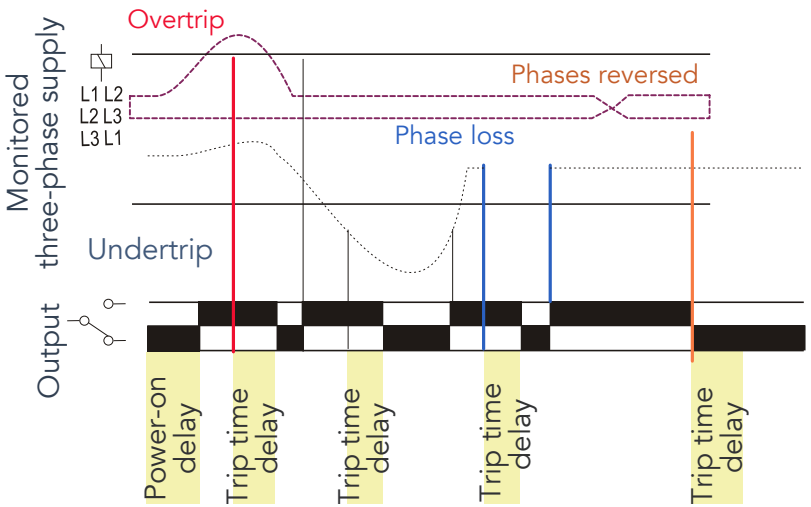
Top design objectives for control installations include capability, reliability, and cost effectiveness. Many new and retrofitted control panels employ components such as PLCs and even industrial PCs (IPCs) and programmable automation controllers (PACs) for advanced connectivity and control. It's often appropriate to use these more complicated (and capable) options and remove or forgo electromechanical relays in the system. In other cases, machines that have simple architectures, fixed functions, or specialty requirements may derive benefit from connectivity and control primarily based on electromechanical relays and SSRs. In still other cases, hybrid approaches are best — to combine various technology types and leverage the benefits of today's controls *and* relays. Let's consider the parameters that can factor into this engineering decision.

Where relays are the suitable choice

Relays are an enduring technology that is simple and efficient — and relay-based control excels at satisfying very specific design requirements. Oftentimes plant personnel and end users are familiar with or prefer their inclusion, and most industrial technicians can install them without issue. That's in contrast with other control options, which necessitate preconfiguration and advanced programming for proper commissioning.



VOLTAGE PROTECTION RELAY TIMING



These are some functions that a relays can execute; adjustments to setpoints are made via rotary DIP switches on the face of the timer relay.

Applications needing little troubleshooting of wired logic benefit from the use of traditional relays as a cost-effective choice. Simple diagnostics are possible with electromechanical relays sporting indicator LEDs (communicating the coil's electrical status) and mechanical flags (communicating contact status) for unambiguous information about the device. But elsewhere, advanced relays include diagnostics as well as communications with microprocessing power and the ability to connect to software via comas they're particularly useful in arrangements where relays interface with motors needing protection against the effects of ground faults, overloads, and other situations that can damage windings.

Further facilitating their application is the increasing convenience of ever-smaller relay footprints for both electromechanical relays and SSRs already discussed. Relay-design advances with optimized circuitry, efficiency, and heatsinks mean today's relays are much smaller than previous generations with the same mA or A switch rating. Even relays having the ice-cube format have in recent years seen advancement with embedded processing.

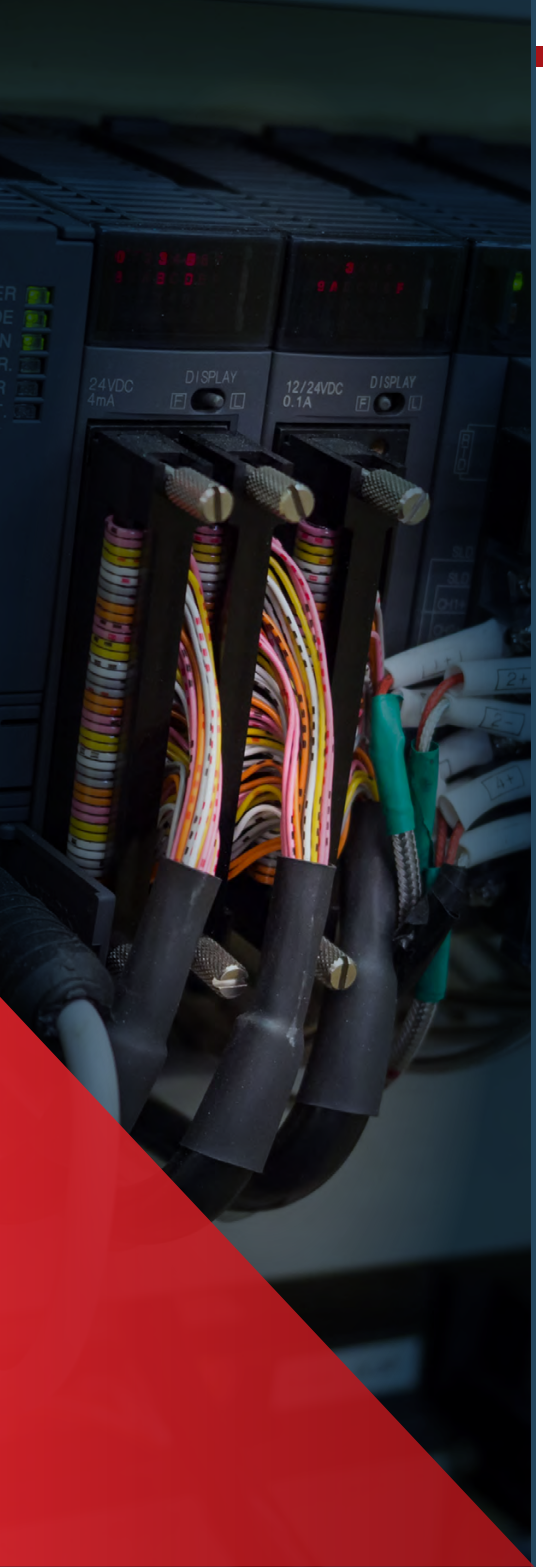
These increasingly compact form factors of smart relays complement their expanded functions. Many such relays are now manufactured in 6-mm slice geometry for DIN-rail mounting, which (besides saving space) also eliminates the need for daisy-chain wiring schemes. In some instances, power bridges (to power multiple relays) further simplify and ruggedize their installation. Another development for simpler relays are increasingly standardized sockets that accept insertion of relays and timers with various pole counts and voltage requirements.

Relays can unburden safety PLCs of alarm and response tasks, which is helpful where programming even modest design changes is a hassle. In fact, smart relays can be more suitable than certain controls on otherwise simple designs needing safety functions — for example, where safety PLCs are prohibitively expensive. The economy of relays (especially on designs needing only a few safety points) can allow safety functionalities that might otherwise be omitted. Here, single-channel relays and smart alarms are top choices for safety sans overcomplicated implementations. In fact, where smart relays are configurable and assume safety functions (going beyond redundant electrical connections with processing capabilities) they've come to closely resemble small safety PLCs. Such configurable relays can have slightly less logic and configurability than PLCs but require less technician knowhow and software for programming.

In fact, several protective relays on the market today include data-processing power as well as advanced communications. These and other smart relays can include EtherNet/IP, CANopen, PROFIBUS, and other protocol communications for the transmission of data about relay-monitored devices.

IO-Link connectivity has had perhaps the most adoption in relays for monitoring functions ... especially of single and three-phase voltage sources. IO-Link-ready relays impart continual access to variables as well as signal scaling — in the past something only possible with PLCs and higher-level controls. In short, signal scaling allows inbound and outbound relay communications for the output of system values and the input of new setpoints. Read more about the spread of IO-Link here and here.

Other configurable relays accepting SIM cards are capable of cellular communications to support M2M functions in remote settings. Such smart relays can transmit instructions and receive alerts even without a wireless network.



Where PLCs are the suitable choice

Controls that take the form of PLCs or incorporate PLC functionalities continue to proliferate in operations needing coordinated system automation. PLCs also excel where logic functions must accommodate reconfigurable equipment – as for machinery involved in producing batch sizes down to one. Such controllers store in memory instantly accessible alternative routines. Case in point: New models of electric vehicles are released every year – requiring annual production-line retooling. Relay-based logic would necessitate physical rewiring and the addition of relay modules to make such changes. That's no issue where onsite personnel is familiar with the design (and may be fastest in some instances) but with no such technicians, PLCs and higher controls accept software-based parametric updates for quick turnaround of new automation routines.

Many PLCs can also handle arrays of I/O nodes and the addition of high-density digital I/O to minimize control-rack size. PLCs impart diagnostic functions to identify failed I/O points requiring replacement – impossible with legacy relays. Plus add-on PLC cards can satisfy the need to supplement with devices exceeding the voltage and current ratings of existing I/O … and options abound to address field-device reactivity.

PLCs have also become increasingly cost-effective – in some instances becoming cost-competitive with relay-socket-connector setups of comparable capabilities. That's in large part because the cost of installation for the latter and the efforts of a technician required to hardwire relay-based systems.

Though we've touched on how some particularly capable smart relays are practically indistinguishable from simple PLCs, some relay logic is limited to simple Boolean control. So where designs go beyond very specific tasks, traditional relays at least necessitate the addition of counters and timing relays– and may not be able to execute all diagnostics required to keep an installation optimized. In contrast, even simple PLCs are capable of counting, timing, and diagnostics – as well as accepting reprogramming for applications that change.

PLCs also facilitate the addition of HMIs for human-readable communication of cycle counts, system status, and faults. With PLCs now allowing technicians more accessibility (with laptops and smartphones) their use has become practical for more applications. In fact, today's most advanced controls can collect and analyze production floor data (as well as distilled data from edge devices with built-in processors) for full IIoT connectivity.

BRIDGING THE IT/OT SECURITY GAP

Cyber attackers are constantly upping their game. Because any network, device, or endpoint can be a potential mark for hackers, when it comes to cybersecurity, there's no such thing as "overprotective."

Dispel is the world's leading provider of industrial control system (ICS) zero trust network access that utilizes both zero trust and moving target defense. Dispel's certified plug-and-play deployment with Mitsubishi Electric hardware gives you easy access to every device in your facility with one install. The result is secure two-way communication that bridges the gap between IT and OT capabilities. Remote users can control and monitor facilities from any approved location via a well-defended pathway. What's more, remote access users don't actually connect to your facility, but a segmented virtual machine that is hardened to the U.S. Defense Information Systems Agency's security standards.

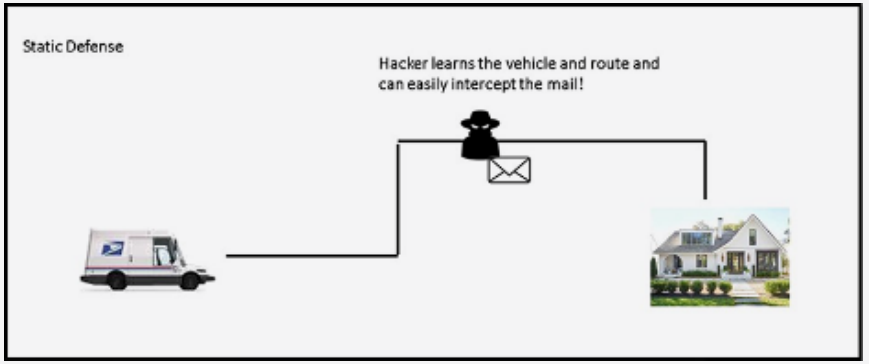
Zero Trust

Dispel's zero trust approach ensures that access to your network is only granted to the right people at the right time. All potential users must put in a request to an administrator or owner and pass a detailed approval process. Users must verify that they match all criteria at the assigned time before they can make the initial connection, which is only provided for specified ports on a single device, with additional devices available to be added at the customer's discretion. Each day, the user is given a fresh, secure environment in which to operate. Once the work concludes, the segmented environment is destroyed.

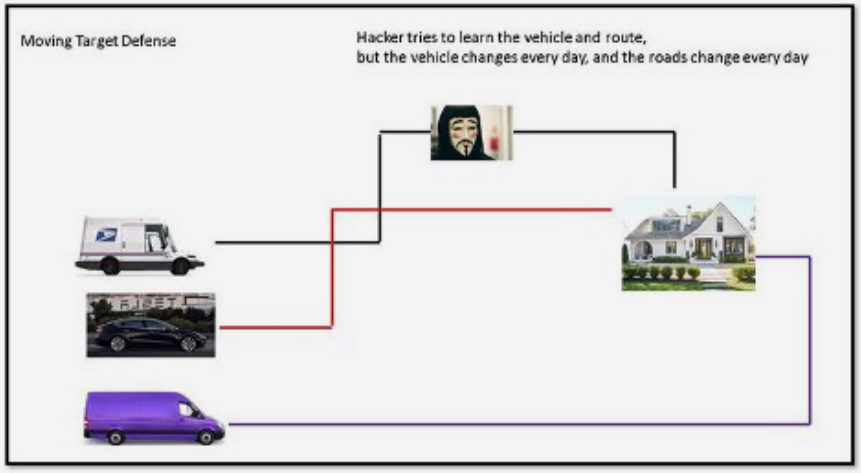
Moving Target

Each time an authorized user connects to the network, the pathway is unique and varied across multiple cloud servers.

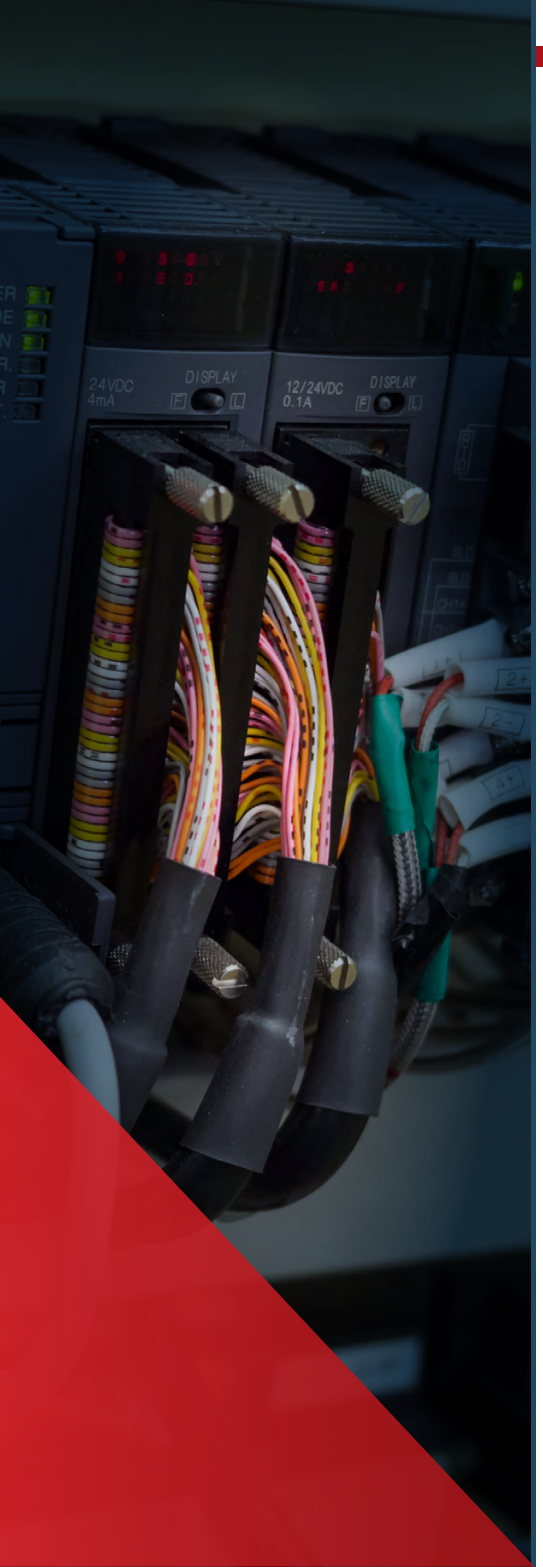
Think of it in terms of a mail truck that takes the same delivery route every day. It would be very easy for someone to monitor that truck's likeness and daily driving path in order to steal their intended target's mail.



Now, imagine that same mail truck is able to change its likeness daily and take a different route each day to deliver the mail. This is moving target defense, which makes it very difficult for someone to learn the pathway from a remote PC to the customer's facility.



Dispel uses multiple general cloud spaces, like Amazon and Azure, to conceal your identity among all the other traffic. If a hacker gains knowledge of the first cloud, the next pathway will be completely different, and the next, and the next...



Where relays complement other controls

We've covered the situations where relays excel and those for which PLCs are most suitable. But a vast array of automated installations benefit from hybrid control architectures that integrate a combination of PLCs and relays —electromechanical relays and SSRs.

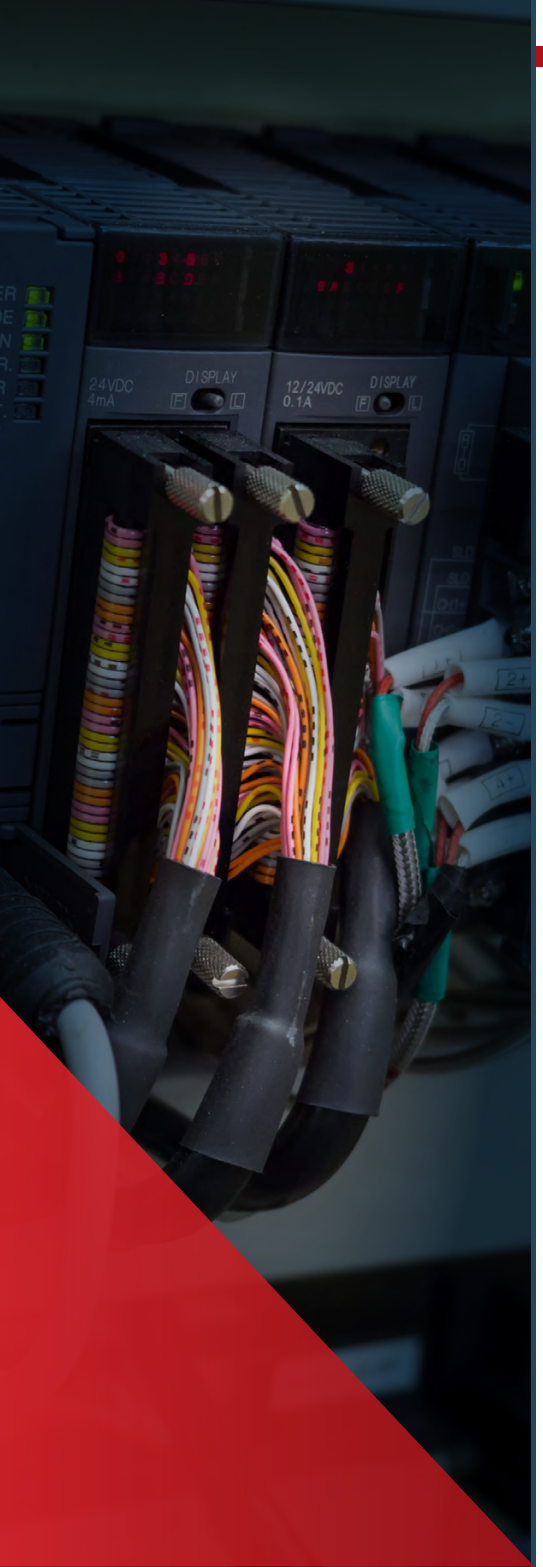
For example, PLCs benefit from the help of relays to switch and control high-current devices. After all, PLCs excel in commanding small electrical loads such as contactors, annunciators, safety indicators, and relay coils of low ampacity. In contrast, advanced relays can switch larger loads of several amps and beyond — associated with electric motors, valves, linear actuators, and other components. Where a particularly inductive field device threatens to damage controls with surge or inrush current (or voltage spikes) interposing relays can (serving as a sacrificial component) complement a PLC. SSRs in particular excel at isolating and protecting controls from EMI — especially in designs that drive particularly inductive loads ... and relays shield PLCs from high-voltage transients when loads are switched off. Of course, relays and PLCs used together must have electromagnetic compatibility.

Still other designs use SSRs or PLC digital output on points needing high-frequency switching because solid-state devices (with their theoretically infinite number of cycles) extend machine life in these situations. SSRs in particular are indispensable for commanding high-speed components in timing, sensing, and machine-vision functions.

During retrofits, portions of a design may accept a PLC upgrade while the rest of the installation continues to run off a relay panel. After all, replacing legacy relay systems with improved models can extend control-panel life. That may mean the swapping out of electromechanical relays with SSRs ... because with automated diagnostics accessible through today's Ethernet-based protocols in particular, SSRs can communicate when failures do occur.

> *PLCs can also benefit from the help of relays to convert field devices' various voltages for signal outputs to one standard (for example, 24 Vdc) if that's all the PLC accepts. Certain SSRs can even execute signal conditioning on these inputs if that's required.*

Elsewhere, relays complement various control and I/O combinations. That's why many high-density I/O components use relays to either convert voltage or amplify current to



accessible values. Relays can complement high-density low-power digital I/O on PACs for some tasks run off traditional high-current equipment. Such hybrid installations are more common for retrofits … but no matter the installation phase, PAC communications and logic often work well with relay-based systems.

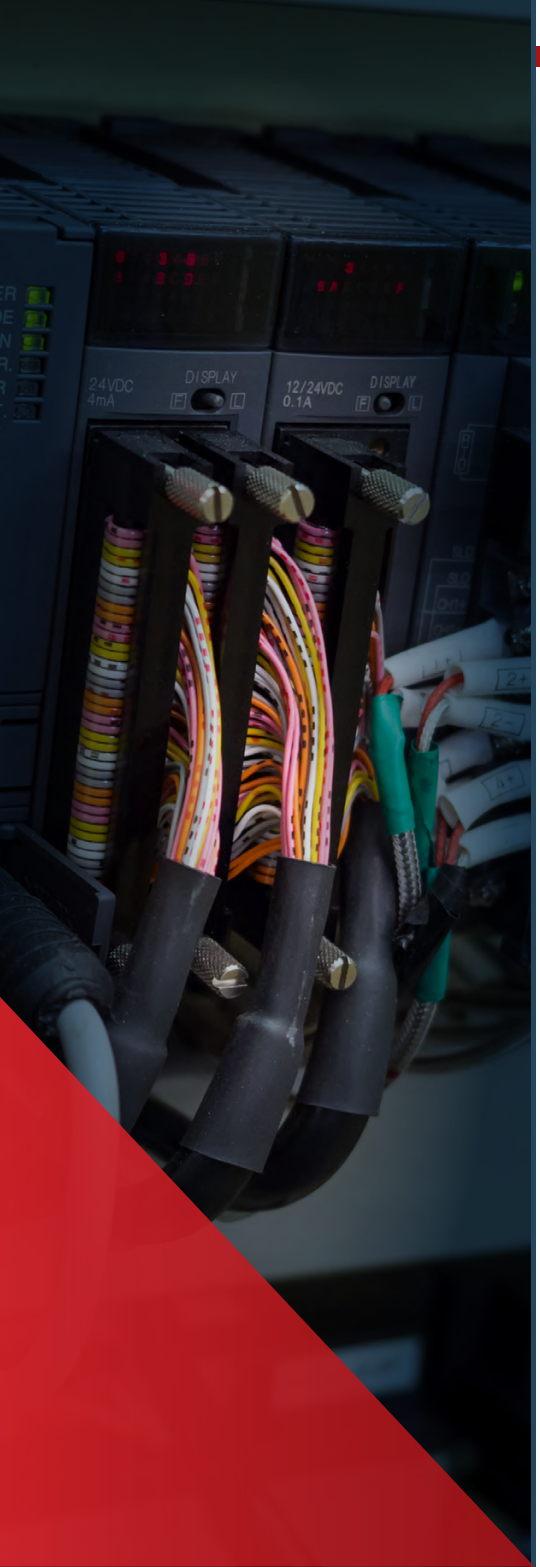
Other engineers pair relays and PLCs when output signals need addressing – typically around the system's PLC where high-ampacity components that the PLC can't handle. Some DIN-rail I/O modules are also designed to integrate with relays for safety – and relay on PLCs for the distributed control of field devices imparting IIoT functionality.

Of course, new designs for both electromechanical relays and SSRs have blurred lines between these subtypes – as well as those between SSRs and other controls. As mentioned, the most advanced programmable relays are often indistinguishable from micro PLCs for delivering configurability via ladder logic and other standard programming … so that relay logic and ladder logic aren't mutually exclusive.

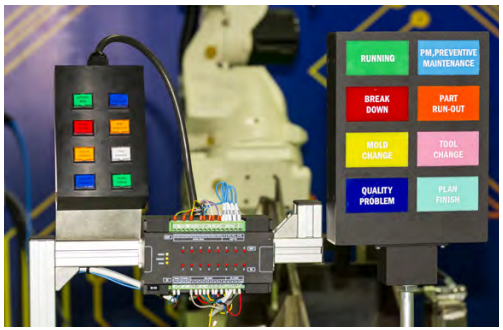
PLCs, PACs, and IPCs for motion applications

As the brain of a servo system, a motion controller is responsible for ensuring the motor is following the specified motion profile and correcting any errors between the commanded value (in terms of position, speed, or torque) and the motor's actual value. The motion controller also handles other supervisory and processing tasks, such as ensuring safety limits are not exceeded and managing I/O (input and output) from devices such as actuators and sensors.

For industrial automation applications (including linear motion systems), three of the most common types of motion controllers are programmable logic controllers (PLCs), programmable automation controllers (PACs), and industrial PCs (IPCs). Although overlapping functionality between the three often means that more than one type of motion controller is suitable for a given application, each controller type has unique capabilities that make it the best-fit choice for certain application requirements.



Capable and simple PLCs



The programmable logic controller (PLC) is probably the most widely-used controller in automation and manufacturing applications. It was originally designed to mimic electrical circuits, replacing the switches and relays that controlled machine motions, for reduced wiring cost and complexity. A basic PLC consists of a processor, memory, I/O to control inputs and outputs, and a user interface, although other peripherals are common.

There are five standard programming languages for PLCs, as defined in IEC 61131-3, allowing programmers to address virtually any application requirement, but the most common is language is arguably ladder diagram — also called ladder logic. PLCs can communicate with other devices via protocols ranging from the relatively simple RS-232 serial communication to Ethernet-based protocols such as EtherCAT or EtherNet/IP.

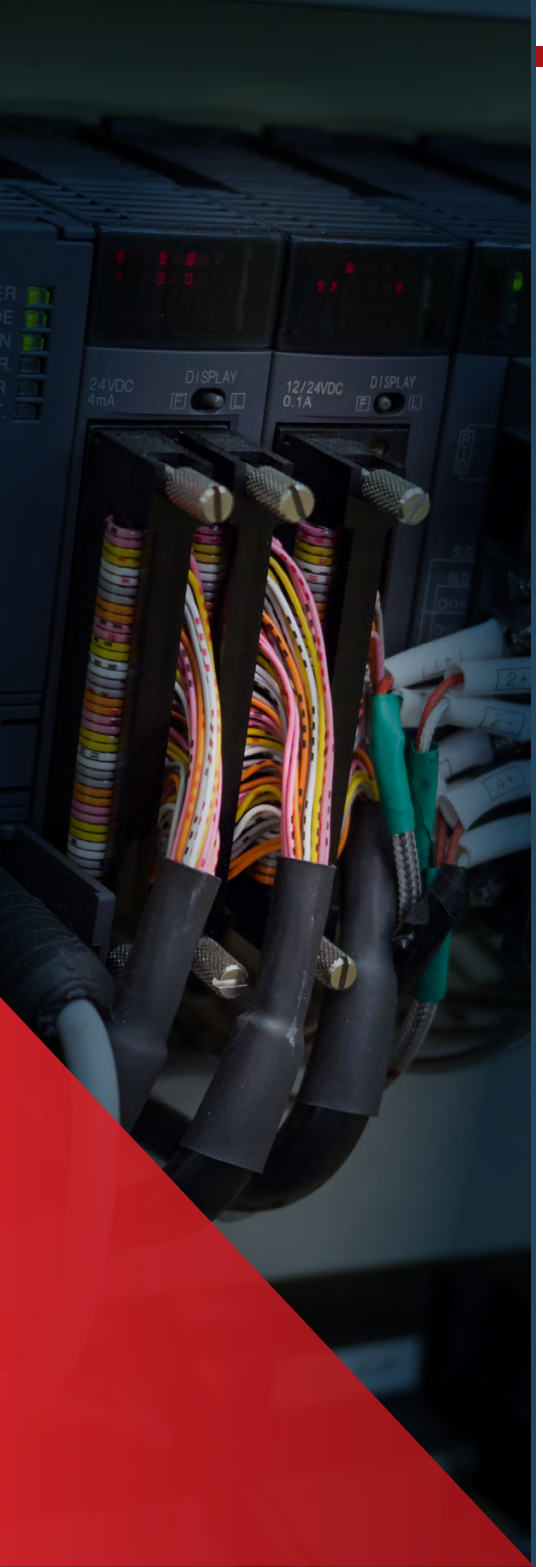
PLCs have the benefit of being familiar (both in terms of hardware and software) to machine programmers and technicians, making them simple to program, diagnose, and service. However, any PLCs with limited processing and storage capabilities should be used for motor-shaft commands or single-axis motion not requiring complex calculations or data handling. Advanced PLCs for more sophisticated motion tasks such as multi-direction or even multi-axis positioning typically necessitate high-speed I/O for components generating signals changing faster than PLC logic-scan times. Here, specialty hardware can capture components' high-frequency signals so the hardware drives switching independent of its scans. Then, a high-frequency PLC output might command a stepper or servomotor to drive an axis through some sequence or stroke.

PACs marry PLC functionality with PC processing power

Programmable automation controllers (PACs) are considered by some to be a hybrid PLC/PC device combining the functionality of advanced PLCs with the processing capability of a PC. There is no formal definition of a PAC, but the ARC Advisory Group (which is widely recognized as having coined the term *programmable automation controller*) says that a PAC should have the following characteristics:

- Multi-domain functionality
- A single, multi-discipline development platform
- Flexible software tools that maximize process flow across machines or processes
- An open, modular architecture
- Compatibility with enterprise networks

While many of these capabilities are found in advanced PLCs, PACs are often distinguished from PLCs by two characteristics: modular design and open architecture.



PACs can also natively handle a larger number of I/O than most PLCs sans expansion modules and can exchange data with other applications and devices ... including with other PACs to form a distributed control system.

Programming can be done in any of the five programming languages defined by IEC 61131-3 for PLCs or in standard PC programming languages such as C/C++. Although many PLCs offer advanced functionality that encroaches on PAC territory, the communication and interoperability of PACs with other devices and systems, coupled with the ability to handle more I/O and larger memory, make PACs a suitable choice for applications that require complex controls, such as coordinated, multi-axis motion or circular interpolation. PACs often excel in applications that require monitoring and control over multiple domains (such as motion and process control) or over an entire plant floor.

IPCs leverage fast processors and large memory

Industrial PCs (IPCs) are in many ways similar to standard PCs, but they're built with more rugged components to withstand the harsh environments often found in industrial applications – including those involving wet and dusty conditions, temperature and humidity extremes, and vibration as well as shock loading. Because they're based on standard PCs, IPCs offer a familiar interface and form factor.

One downside of the standard PC is that it doesn't deliver realtime deterministic performance – a necessity in complex applications such as coordinated motion. To solve this problem, industrial PCs often run both a conventional operation system (such as Windows) and a realtime operating system (RTOS) that allows them to provide deterministic control. That said, some IPCs offer better memory and data processing capabilities than certain legacy PLCs so are sometimes chosen for applications that require intensive calculations and data storage including those employing machine vision or imaging systems. Most all IPCs can interface with higher-level plant operating systems such as ERP (enterprise resource planning) systems, building automation systems or BASs, and SCADA systems as well.

The levers on WAGO's 831 Series MCS MAXI 6 pluggable PCB connectors require no tools for quick, easy, and secure connections. This also helps when field wiring in narrow or difficult to access installation spaces. The pluggable PCB connector functions as an important interface between a storage battery's PCB and the electrical conductors. The clear lever position also indicates whether the clamping point is open or closed. If the lever is closed, then the conductor is connected in a secure and maintenance-free way. Thanks to Push-in CAGE CLAMP® connection technology, solid conductors and fine-stranded conductors with ferrules can be connected by simply pushing them into the unit. The integrated protection against mismating in the 831 Series guarantees reverse-polarity protection. Along with providing high outputs due to a conductor range of 20-8 AWG and a rated current of 37 amps, the 831 Series is also UL and IEC certification for international markets.

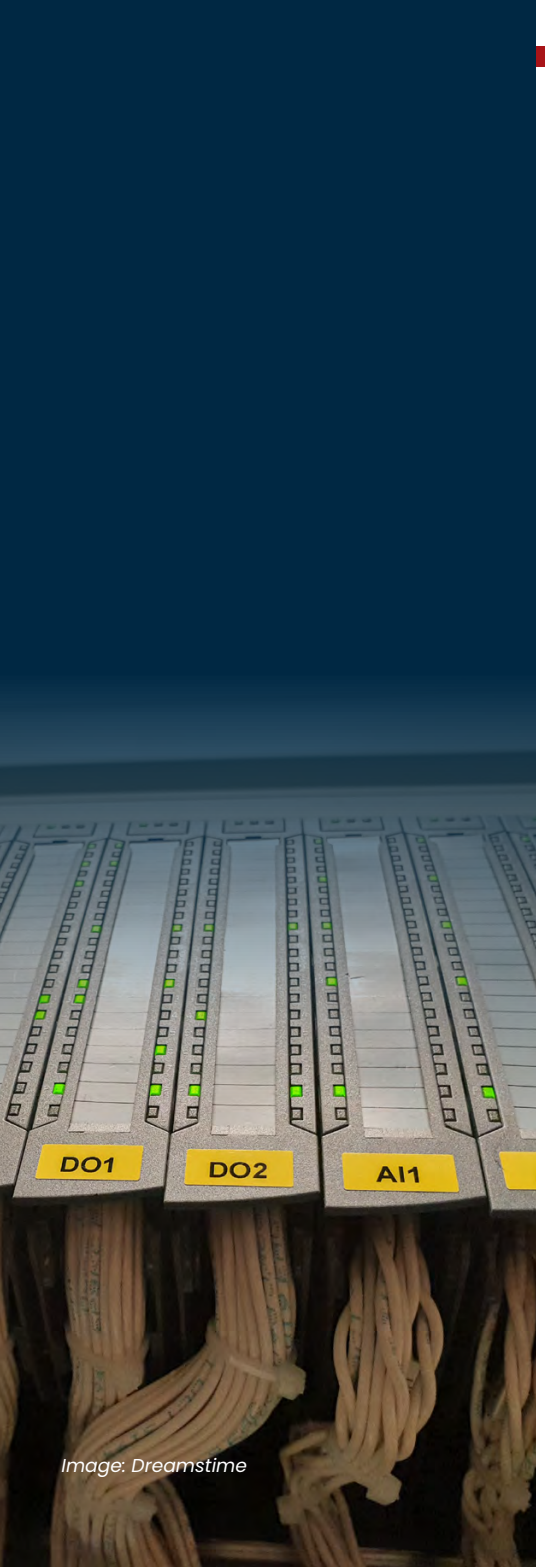


Image: Dreamstime

Blurring lines between programmable controller types

Distributed control systems (DCSs), remote terminal units (RTUs), and PLCs are control systems with hardware and programming designed to meet the requirements of specific applications. Today's programmable automation controller (PAC) hardware runs DCS, RTU, PLC, and PC functions as software to replicate the legacy hardware that's operated in motion systems for decades.

Because controller functionality previously followed application, early-generation controls were engineered with features and functionalities to serve specific individual market segments. So for engineers and users working in segregated market segments served by these legacy controls, the way in which PACs replicate these control schemes provide a level convenience and familiarity during control implementation. In fact, some PACs have controller redundancy (hot standby CPUs) and native Ethernet on their backplanes; these can serve as integrated IIoT automation controllers. Two in-rack application-specific motion options include a two-channel pulse train output (PTO) module for servo drives and a module for interfacing with encoders.

Due to computer processors' increasing capabilities and declining cost, distinctions between various control types have blurred. The PAC itself is the extension of the PLC to incorporate greater data processing and communications capabilities incomprehensible when the PLC was first invented in the late 1960s.

More about DCS, RTU, and PLC setups

Originally, DCSs were a collection of RTUs operating on a network of copper phone wires. When these systems were first developed, communications were limited to simple alarm states from remote pieces of equipment.

RTUs were small standalone controllers that could execute small logical tasks — usually involving some simple information such as elapsed run time, totalizing units of flow or measuring a process value. Early systems had no ability to transmit data, but if design engineers could rent a copper phone line from the phone company, design engineers could create an alarm to indicate that a process value had been exceeded or the RTU needed to be read.

PLCs were invented to replace relay-based control systems and the first hardware standard as electricity became the dominant power source for manufacturing systems. As control requirements became more complex, hardwired relay control became impractical because manufacturing needed more reliable and reconfigurable (programmable) systems. Given the primitive hardware and rigid focus on task execution, early PLCs were a difficult control to network. Today's PLCs are supremely easy to implement.

In fact, PLCs are still more appropriate than PACs in standalone applications such as machine axes that run preset sequences. Rule of thumb: Anywhere PAC functions would otherwise go unused; it still makes sense to use the ever-economical PLC. Pressure from plant personnel and the enduring value of ladder logic make PLCs a first choice in many applications.

Industrial personal computers (PCs) have evolved dramatically with processor advancements. Early industrial PCs were limited to programming terminals and storage devices because their operating systems weren't robust enough for industrial controls. This limitation has disappeared due to improved hardware and widely available realtime operating systems such as Linux.

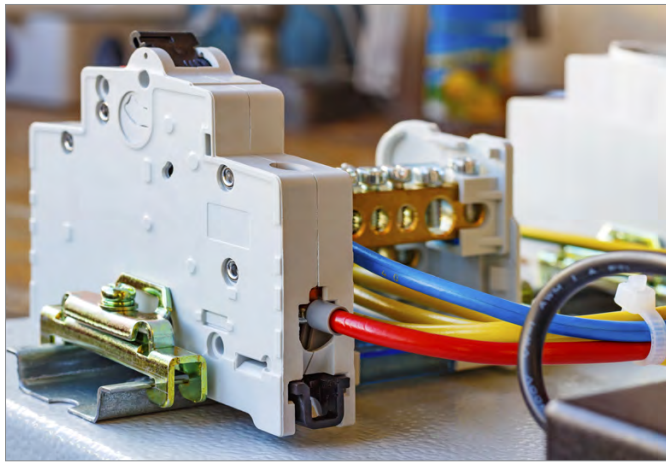
Choosing between a traditional PLC and PAC is complicated by how they share similarities. However, some PACs can multitask with numerous threads — going beyond some PLCs' single program path to manage concurrent operations in complex installations. This is especially useful where multiple machines and systems need fast execution of advanced control algorithms and database manipulation. In fact, PACs run control programs and communicate with HMIs as well as digital and analog I/O.

The way in which some PACs have multi-processor architecture lets them support multiple programming environments — and run any or all DCS, RTU, PLC and PC functions. In addition, programmable automation controllers have high-bandwidth internal architectures that allow multiple processors and multiple tasks to execute simultaneously. In this way, designers can create myriad control-system structures to support any application requirement.

Mounting PLCs on DIN rails

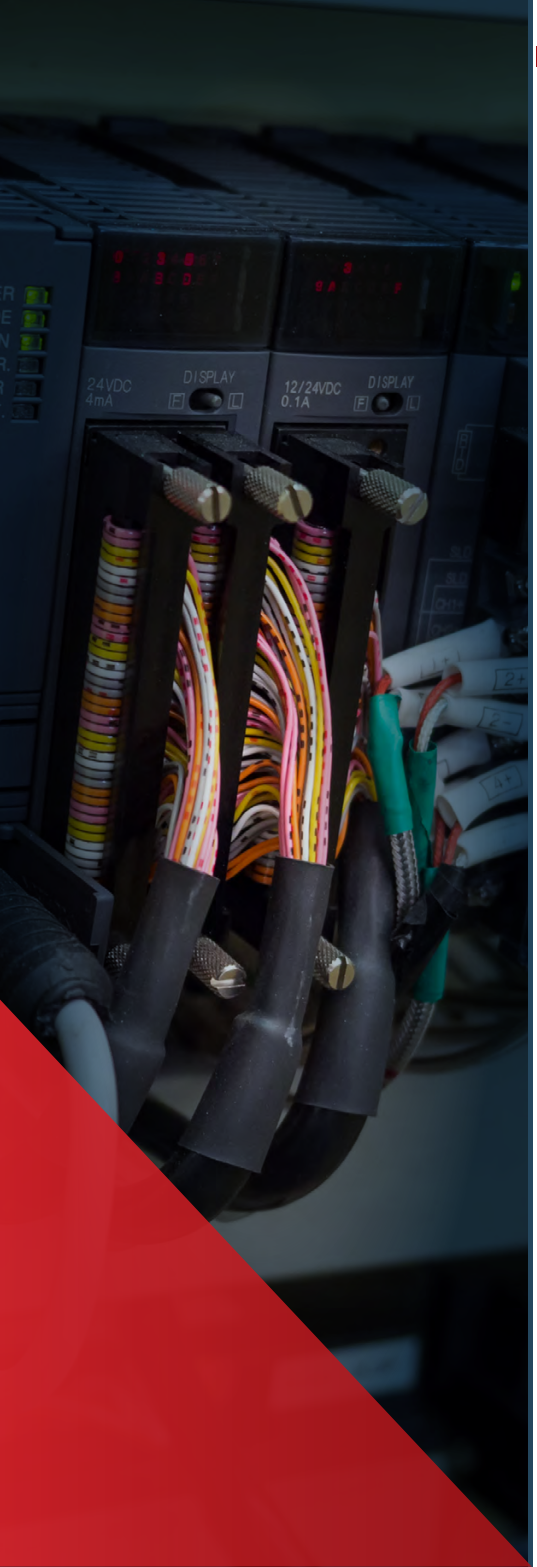
The Deutsches Institut fur Normung (DIN) is a German institute for standardization and a leading provider of technical standards worldwide. It is the organization which has (among other things) established the standard for uniform geometry and electrical connectivity regarding 35-mm DIN rails so ubiquitous in today's control panels and automation equipment.

The ubiquitous nature of components for DIN-rail mounting systems — sometimes simply called racks or bases — makes this standard particularly helpful. That's especially true when panel redesigns are in order. Here, a technician need only to unclamp components, slide them to their new locations, and reclamp them to the DIN rail. In addition, DIN rail prevents upside-down and other incorrect installation.



This is a sample of DIN rail with a module mounted to it — as well as an end clamp (also called an end plate) to prevent components from sliding off the rail end.

Why is DIN rail so ubiquitous? Well, consider terminal blocks — a component commonly found mounted onto such DIN rails. Terminal blocks meant for rail mounting are either DIN compliant or proprietary. The latter come in an array of formats, with geometries that are vendor specific — necessitating strict use of only the vendor's blocks and rail with no possible mixing and matching. In contrast, terminal blocks that comply with DIN requirements are in many instances interchangeable; they also tend to be smaller than proprietary blocks for a given electrical-power rating.



Or consider how some so-called stackable PLCs work as standalone controllers having a CPU sporting a modest array of built-in I/O and accepting the addition of I/O modules. In contrast, rack-based PLCs mount to rails typically cut to a length that accommodates the PLC and all the add-on module the application requires.

> *A typical DIN rail in use might accommodate relay sockets, circuit breakers, terminal blocks, fuses — as well as small drives, industrial-communication devices, PLCs, or other controls*

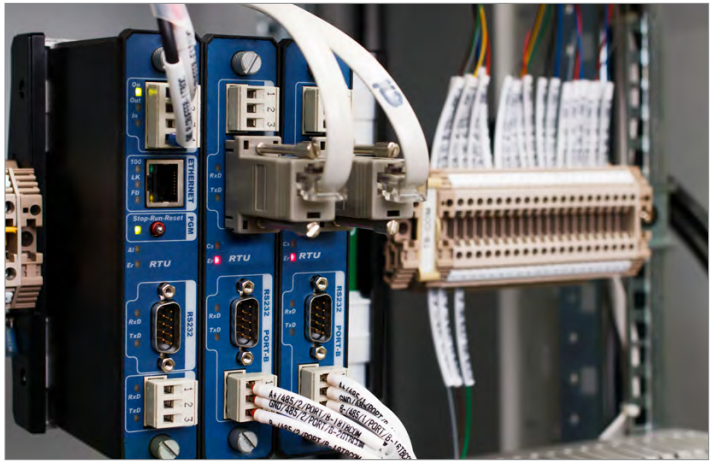
Note that besides their differing rail and mount geometry, DIN and proprietary systems also have differing electrical connections. For example, DIN-style terminal blocks have a dead-front configuration — with recessed termination hardware in their plastic block housing to isolate electrically live parts ... and minimize the risk of shock, even if connections on the block are live. In contrast, some proprietary terminal blocks have open electrical receptacles.

Basic construction and versions of DIN rail

DIN rail is made of straight extruded aluminum or cold-rolled steel tracks that are finished with either chrome or zinc plating. Slots in this rail allow installers to screw or both DIN rail to the inside walls of a control panel or other protected wall. Longitudinal bends in the DIN rail profile serve as shoulders upon which various components can clamp.

Several DIN-rail geometries exist ... and in many cases, components for DIN-rail mounting are capable of clamping to a couple different styles of DIN rail.

DIN 1 or C DIN rail — so called for is C-shaped cross section — is for systems rated to 600 V. With a total face width of 35 mm, this is a particularly common variation. This rail's edges turn inward, so components attach to such rail by clip mechanisms that engage the inside channels of the rail tracks. Note that some technicians nickname component attachments by key dimensions — so a terminal block with hardware to snap onto a 32-mm DIN 1 rail (having a lower lip 22.5-mm deep) may be called a 22.5-mm block for the depth of its lower latching mechanism.



DIN rail provides a standard way of organizing and mounting PLCs, breakers, terminal blocks, relays, drives, and other electric and electronic components. This is an industrial control panel with various power and control components mounted on large-format DIN rail. Image: Dreamstime

G DIN rail – also called asymmetric rail for its extra bit of lip on one longitudinal edge – is also for systems rated to 600 V. In fact, this type of rail closely resembles C DIN rail ... but with its deeper bottom lip, it's often used to hold particularly heavy components that need more mounting engagement. G-shaped rail for the orderly mounting of components is the oldest rack design, dating back to the 1920s ... and predating the first DIN-rail standards by a couple decades.

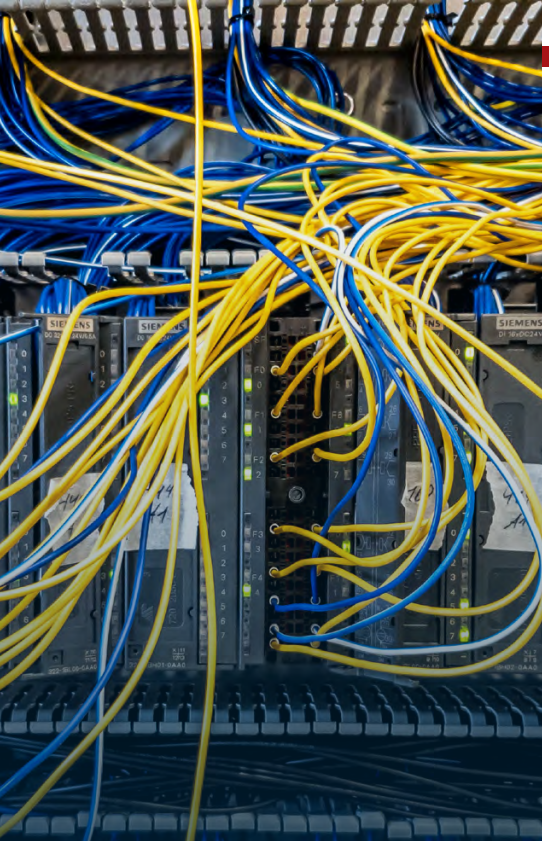
DIN2 or top-hat DIN rail – also called symmetric rail for the matching longitudinal bends of its brimmed-hat cross-sectional profile – is also for systems rated to 600 V. Its edges flare outward; components attach to such rail by wrapping clip mechanisms around these edges.

Mini DIN rail is for systems rated to 300 V. These are far less common than 35-mm DIN rail variations.

75-mm top-hat rail is yet another option for very large electrical and electronic components.

Overview of PLC programming

The first PLCs of the 1970s challenged the mainstay of early electrified automation – that of hardwired electromechanical relay-based control. In contrast with these inflexible switch-based systems, early PLC precursors leveraged the advent of digital computing to run inherently reconfigurable ladder logic replicating those of relay ladder diagrams. Slowly, industrial networks developed to let PLCs connect sensors, actuators, and other edge devices involved in automation. As the circuitry within PLCs became capable of more processing power, ladder-logic capabilities advanced as well ... employing the hardware's boosted memory and processing to run more sophisticated controls.



Good PLC programming and installation aim to avoid complicated and messy wiring.

Image: Dreamstime

The first PLC programmers typically entered code into panel-mounted terminals. Slowly, programming migrated to PCs until (as used today) software with intuitive GUIs proliferated to make PLC programming, troubleshooting, and monitoring easier. So now, PLC programs are written on PCs within vendor software and then loaded into the PLC's memory to run the automated operation at hand.

To promote interoperability and ease of use for industrial-automation controls, the International Electrotechnical Commission IEC 61131-3 standard is:

1. supported and informed by the PLCopen independent vendor and user association
2. Written and employable via the Codesys integrated development environment (IDE) as well as vendor software offerings

... and defines a common PLC programming-language framework as well as PLC languages and software-development conventions. More specifically, IEC 61131-3 defines five accepted PLC programming languages:

- Structured text (ST) serving as a high-level language like Pascal or C
- Function block diagram (FBD) as a graphical language featuring function blocks
- Sequential function chart (SFC) as another graphical language for commanding sequences
- Instruction List (IL) for low-level language functions
- Ladder diagram (LD) sometimes called ladder logic or ladder control — first designed after relay ladder-logic diagrams and the first and still most common PLC programming standard in use.

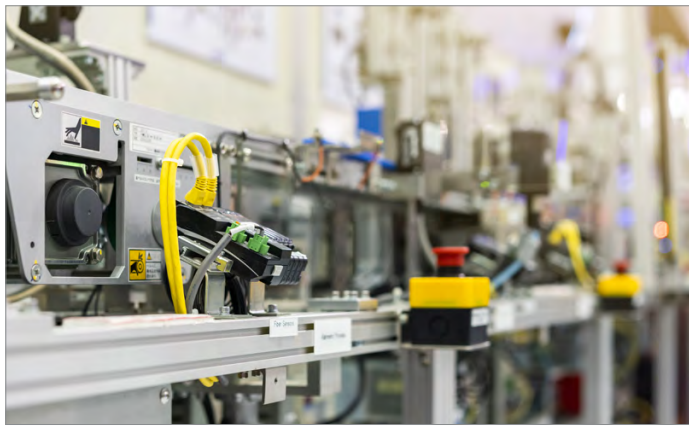
IEC 61131-3 outlines these languages' syntax, semantics, and execution so they're reasonably to fully interoperable, reusable, and cross-platform portable. Then PLCopen defines additional specifications and libraries to complete these specifications. For example, PLCopen motion control standards define programming PLC elements for servomotor and robotic motion systems; safety standards define program requirements to achieve functional safety; and eXtensible Markup Language or XML standards define requirements to exchange PLC information using the XML format. Straight as well as reskinned and vendor-branded Codesys adaptations do leverage the efficiencies of IEC 61131-3 compatibilities.

Image: Dreamstime

All this said, hardware from different PLC manufacturers employs slightly different configuration, programming, and conventions (with memory addressing, instruction sets, symbols, and user interfaces typically unique to each manufacturer). PLCs from a given manufacturer usually require programming via software provided by that same manufacturer … which in turn effectively limits global industry compatibility. However, some PLC suppliers would argue that moderately closed ecosystems are in fact the only way to guarantee top PLC performance and reliability. No matter the PLC maker, all ladder logic (once compiled and downloaded) executes the same way.

More on the original PLC program of ladder logic

As mentioned, ladder-logic programming uses symbols originally based on the drawn representation of relay-based wire-laden circuits. PLC ladder logic contains instructions supported by variables and tags (and detailing contact types, signal values, coil states, timer parameters, and more elaborate functions) as **vertical rails** bridged by numbered **horizontal rungs**. In this way, relay-logic rungs representing wiring bridging power-supply active and 0-V rails are in ladder logic adapted to represent the logic's course through programmed code. But in contrast with the complex relay-system wiring that inspired it, all controlled automation components (acting as inputs and outputs) directly wire to the PLC to be commanded through the ladder logic.



PLCs are simple but flexible and often work best in single-axis motion systems. Image: Dreamstime

In fact, ladder logic has had the momentum to evolve into and endure as a leading programming language because it's relatively easy to understand (especially compared to FORTRAN, C, BASIC, and Pascal text-based languages of yore) and many engineers, programmers, and technicians around the globe are quite familiar with it.

Ladder logic is read like English and other Latin languages – from the top (left to right) and so on downward. PLCs execute the logic via **scans** – with each scan:

- Reading and logging all input states
- Evaluating each rung of ladder logic
- Commanding each rung's last symbol value on down the sequence to prompt outputs to action.

Besides the rails and rungs already detailed, ladder logic also features inputs and outputs, logic expressions, address notations, tag names, and comments. PLC inputs come from field devices (hardwired to the PLC terminals) such as switches or buttons being activated by plant personnel or a machine subsystem. Reflected in the ladder diagram (as contact symbols) are how some inputs' contacts are normally closed (NC) or normally open (NO); states can be true or false, 1 or 0, yes or no, on or off, and high or low. Outputs on the other hand are field components such as switches and electric motors commanded to turn on or off or change their operation in some other way. Outputs in the ladder diagram are represented by the relay coil symbol.

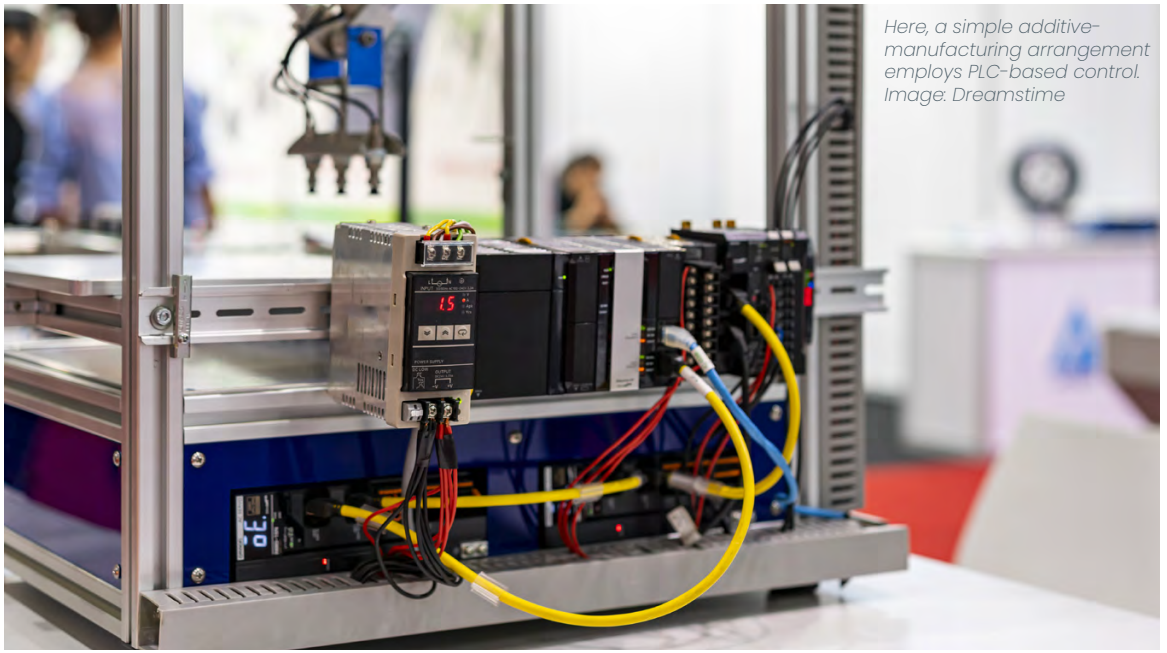
Logic expressions in ladder logic include not, and, if, then, and or. These combine with inputs and outputs to define controls and enable most all automation tasks. Address notations and ***tag names*** (as descriptions allocated to addresses) define the PLC's input, output, and logic expression memory-addressing structure. In contrast, ***comments*** are human-readable descriptions (in some cases appearing as popups in manufacturer programming software) that describe the purpose of each rung's control functions.

Microprocessors in PLCs (like PCs) ultimately run the code as binary strings – 00101011 for example. However, all these values in PLC programming software are defined via human-readable GUI elements. Modern PLC ladder-logic programming software lets users drag and drop control elements in a GUI; then code is automatically generated. In many cases, troubleshooting routines highlight rung sections as they advance through the code from top to bottom to make that easier as well.

Program organization units (POUs) in PLC programming

The software model for PLCs, as defined in the IEC 61131-3 standard, Programmable Controllers – Part 3: Programming Languages, consists of, at the top level, a configuration. The configuration defines the hardware structure and logic of the PLC system and contains resources (the PLC or PAC, for example) which are able to execute programs. These resources are controlled by tasks, which invoke the execution of software blocks that make up the PLC project. These software blocks are called program organization units.

IEC 61131-3 defines three types of program organization units (POUs) – programs, function blocks, and functions.



Here, a simple additive-manufacturing arrangement employs PLC-based control. Image: Dreamstime

Programs in PLC programming

Programs are the highest-level program organization units and can be written in any of the IEC 61131-3 programming languages. Each program is a network of functions and function blocks (see below) that controls the machine or process. Therefore, at least one program is required in any project.

> *Programs can be called by tasks. Function blocks can be called by programs or by other function blocks. Functions can be invoked by any POU type.*

Programs can read and write to other input or output variables and can communicate with other programs. Plus they are the only POUs that can declare global variables, which are made available across the project, and access paths, which allow data to be exchanged between configurations.

Programs can be called (initiated) by tasks. Each time a program is called, if the values of the program are changed, the changes are retained the next time the program is called, even if it is called by another task.

Function blocks in POUs

The most common type of POU, function blocks are segments of reusable code that have internal memory and can return different outputs even when the same inputs are used. In other words, the results of a function block are conditional on the previous output of the function block or the current state of the process or action. An example of a function block is a PID control loop.

Function blocks can be called by programs or by other function blocks ... and in some implementations of IEC 61131-3, they can be called by tasks. The memory that will capture and store the output must be allocated for each unique instance of the function block – a process sometimes called instantiation because it creates an instance of the function block.

Functions in PLC programming

A function can be thought of as a subprogram (often an equation) that has no internal memory, so it returns a value rather than an output. This means that any time a function is invoked (performed), if the same inputs are used, the same value will be returned, regardless of the number of times the function is used. Common examples of functions are ADD and SQRT (square root).

The IEC 61131-3 software model – with programs, function blocks, and functions for creating PLC projects – ensures efficient programming by making it possible to copy and reuse software blocks that have already been implemented and tested elsewhere or to identify and deactivate them for troubleshooting.

The waning of the instruction list (IL) language

The international standard IEC 61131-3 specifies programming languages for programmable controllers (aka programmable logic controllers, or PLCs) including ladder diagram (LD), structured text (ST), function block (FB), instruction list (IL), and sequential function chart (SFC). Of these, instruction list and structured text are considered textual languages, meaning they use text, numbers, and punctuation – like natural language – while the others are considered graphical, using symbols and visual relationships to specify instructions. The IEC introduced this standard set of languages for PLCs to give programmers a toolbox of languages that can address virtually any application, while keeping software vendor-independent.

Components wire back to the PLC.
Image: Dreamstime

Note that sequential function chart (SFC) is not technically a language, but rather a graphical means of partitioning code and visually displaying the machine state or mode.

Instruction list was one of the first PLC programming languages, along with ladder logic. The instruction list language is considered a low-level language, meaning it is very close to machine code – the binary language that a computer's CPU executes directly. With instruction list language, each instruction, or machine command, is placed on a new line. This method of arranging commands in stacked lines is sometimes called a stack-based logic solver similar to that used in RPN logic on some calculators.

Instructions consist of operators, operands, and modifiers, using mnemonics for the operators ... for example, A for and or MOV for move.

A benefit of low-level languages, including instruction list, is that they are very fast and efficient – especially when compared to graphical languages – and use less memory. For this reason, instruction list language is typically used in applications such as control loops that require very fast processing speeds.

However, instruction list programs can be prone to run-time errors, and they can cause infinite loops or illegal arithmetic operations. Most importantly, though, in today's manufacturing environments, personnel other than programmers – including maintenance engineers and electricians – should be able to troubleshoot issues or faults with equipment, including controls and programs. And while instruction list language is very programmer-friendly, without special training in the language, it's very difficult to analyze and troubleshoot the code.

In other words, support personnel need to be trained specifically in instruction list language, which is simply not practical, especially when there are other languages (especially graphical ones) that can address the same applications and problems and are more user-friendly to non-programmers.

To this end, *edition 3.0 of IEC TR 61131-8 (November 2017) industrial-process measurement and control – Programmable controllers – Part 8: Guidelines for the application and implementation of programming languages* notes that in the IEC 61131-1:2013 standard, the instruction list language is marked as deprecated (discouraged from use) for the next release of the standard. The committee's stated reason for this deprecation indicates that an assembler-like language is not up-to-date in modern development environments.

Research suggests that instruction list has not been a popular language for PLC programming in quite some time, with its use being rare in modern motion control applications. This limited use together with its deprecation by the IEC means PLC hardware and software manufacturers may very well stop supporting instruction list programming in future versions of their products.

More information on PLC programming

PLCs versus distributed control systems (DCS)
PLC programming beyond ladder logic
The evolution of CoDeSys software
What are IEC 61131-3 and PLCopen?
PLC function blocks and IEC 61131-3 classifications

XML control-software portability with IEC 61131-3

Say an engineering team wants to switch their design's PLC from one model to another – a relatively infrequent but ordinary specification decision. How well do XML files really work for importing and exporting? How much are component manufacturers really motivated to make that work … and not lock engineers onto one specific hardware?

Engineers are right to be skeptical about this. Ultimately, file portability depends on the system-hardware manufacturers … whether the components are connectivity components, smart drives, or other devices using soft motion and IEC 61131 programming.

PLC and PAC hardware is associated with the most limited file portability; in most cases, controller manufacturers offer online software libraries with pre-written code to simplify new machine setup, differentiate their product line, and (not coincidentally) fix users to their particular brand.

That means even if a programming engineer has his or her own XML files, any instances of code based on the hardware suppliers' libraries will likely not transfer to other devices very easily. Such situations demand that engineers rewrite parts of that code.

That said, limited software portability is ultimately less of a snag than the learning curve (especially in learning a new programming language if that's required) associated with moving from one brand of device to another. Migration will always necessitate adjustments to the code … but programming based on standards means engineers don't need to start from scratch during migrations.

Familiarity with industry-standard programming empowers engineers and spurs component manufacturers to make ever-more competitive technology … as the manufacturers can no longer assume OEMs will never migrate simply because of some familiarity with their software.

When design engineers switch controller brands

Besides migrations made to leverage offerings with open-source programming flexibility, technical support is the next biggest reason design engineers switch controller brands. That's especially true if they consider their current support insufficient or somehow contentious. Excessive lead times can also spur switches – especially if an important end user is left in a lurch during a machine-down or other critical situation. Quick and reliable manufacturer support of an OEM to recommission machinery and get it back up and running is paramount if an end user is losing hundreds of thousands of dollars an hour.

Another area of differentiation is manufacturers' graphical user interface (GUI) for hardware setup. Hardware performance from most manufacturers is fairly competitive across the motion industry, but GUI ease of use and functionality vary widely.

When looking to specify a new controller for new machine builds, design engineers should seek:

- Motion-component suppliers that fully support the IEC 61131-3 standard and not just one language of the standard
- Controllers with ease of integration and programmability – for efficient and sophisticated setup of HMI, PLC, and IoT gateway (Cloud connectivity) functionality … preferably all in a single device
- Hardware with functional safety over a suitable network such as EtherCAT, for example
- Suppliers that offer their software for free or at reasonable cost – so that there's no huge investment needed to try out or license the software.

Structured text versus ladder logic for motion designs

As mentioned, one key strength of programming with IEC 61131-3 is that it allows layering of code in multiple languages. That addresses the varied needs of all personnel types who will ultimately need to operate and access the machine and its code. It also lets engineers get away from using one language for everything. After all, some languages work better for process-oriented tasks than discrete motion control, for example. The IEC 61131-3 environment lets engineers blend even such disparate programming together. There's accommodation of PLCopen motion-control function blocks (with PLCopen being another industry standard to level the playing field for motion-control manufacturers) with motion functions such as MC_Power to power drives and MC_Jog to move motors, for example.

- One common approach is to create motion-control code in structured text (ST). Staunch adherents to structured text feel ST excels in nearly every situation. However, a drawback can be less accessibility for technicians.
- Another common approach is to create motion-control function blocks at the ladder level – especially where maintenance personnel may need to understand and track machine functions) or sequential function charts for higher level perspectives on whole processes.

But aren't function-block diagrams mostly for process control – not motion control? Today's programmable automation controllers (PACs) and the ways motion control has evolved over the last five years means there's now more fluidity in what languages are used where. Controller hardware in the form of PACs is a vast departure from legacy systems with different devices performing different jobs – with a standalone motion controller segregated from the machine's PLC set aside from its HMI.

So it used to be that all this hardware was associated with different processes and software. Now unifying hardware along with IEC 61131-3 has substantially changed the way industrial coders program – and even the ways industry conceptualizes discrete versus process control … as now they're often run off one device with a flexibly selected mix of languages.