

Classe 5 BM - C.L.I.L.

Programmable logic controller for automation systems



Siemens Simatic S7-400 system in a rack, left-to-right: power supply unit (PS), CPU, interface module (IM) and communication processor (CP).

A **programmable logic controller, PLC**, or **programmable controller** is a digital computer used for automation of typically industrial electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or light fixtures. PLCs are used in many machines, in many industries. PLCs are designed for multiple arrangements of digital and analog

inputs and outputs, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed-up or non-volatile memory. A PLC is an example of a "hard" real-time system since output results must be produced in response to input conditions within a limited time, otherwise unintended operation will result.

Before the PLC, control, sequencing, and safety interlock logic for manufacturing automobiles was mainly composed of relays, cam timers, drum sequencers, and dedicated closed-loop controllers. Since these could number in the hundreds or even thousands, the process for updating such facilities for the yearly model change-over was very time consuming and expensive, as electricians needed to individually rewire the relays to change their operational characteristics.

Digital computers, being general-purpose programmable devices, were soon applied to control industrial processes. Early computers required specialist programmers, and stringent operating environmental control for temperature, cleanliness, and power quality. Using a general-purpose computer for process control required protecting the computer from the plant floor conditions. An industrial control computer would have several attributes: it would tolerate the shop-floor environment, it would support discrete (bit-form) input and output in an easily extensible manner, it would not require years of training to use, and it would permit its operation to

be monitored. The response time of any computer system must be fast enough to be useful for control; the required speed varying according to the nature of the process.^[1] Since many industrial processes have timescales easily addressed by millisecond response times, modern (fast, small, reliable) electronics greatly facilitate building reliable controllers, especially because performance can be traded off for reliability.

In 1968 GM Hydra-Matic (the automatic transmission division of General Motors) issued a request for proposals for an electronic replacement for hard-wired relay systems based on a white paper written by engineer Edward R. Clark. The winning proposal came from Bedford Associates of Bedford, Massachusetts. The first PLC, designated the 084 because it was Bedford Associates' eighty-fourth project, was the result. Bedford Associates started a new company dedicated to developing, manufacturing, selling, and servicing this new product: Modicon, which stood for **MO**dular **DIG**ital **CON**troller. One of the people who worked on that project was Dick Morley, who is considered to be the "father" of the PLC. The Modicon brand was sold in 1977 to Gould Electronics, later acquired by German Company AEG, and then by French Schneider Electric, the current owner.

One of the very first 084 models built is now on display at Modicon's headquarters in North Andover, Massachusetts. It was presented to Modicon by GM, when the unit was retired after nearly twenty years

of uninterrupted service. Modicon used the 84 moniker at the end of its product range until the 984 made its appearance.

The automotive industry is still one of the largest users of PLCs.

Early PLCs were designed to replace relay logic systems. These PLCs were programmed in "ladder logic", which strongly resembles a schematic diagram of relay logic. This program notation was chosen to reduce training demands for the existing technicians. Other early PLCs used a form of instruction list programming, based on a stack-based logic solver.

Modern PLCs can be programmed in a variety of ways, from the relay-derived ladder logic to programming languages such as specially adapted dialects of BASIC and C. Another method is state logic, a very high-level programming language designed to program PLCs based on state transition diagrams.

Programming

Early PLCs, up to the mid-1990s, were programmed using proprietary programming panels or special-purpose programming terminals, which often had dedicated function keys representing the various logical elements of PLC programs. Some proprietary programming terminals displayed the elements of PLC programs as graphic symbols, but plain ASCII character representations of contacts, coils, and wires were common.

Programs were stored on cassette tape cartridges. Facilities for printing and documentation were minimal due to lack of memory capacity. The oldest PLCs used non-volatile magnetic core memory.

More recently, PLCs are programmed using application software on personal computers, which now represent the logic in graphic form instead of character symbols. The computer is connected to the PLC through Ethernet, RS-232, RS-485, or RS-422 cabling. The programming software allows entry and editing of the ladder-style logic. Generally the software provides functions for debugging and troubleshooting the PLC software, for example, by highlighting portions of the logic to show current status during operation or via simulation. The software will upload and download the PLC program, for backup and restoration purposes. In some models of programmable controller, the program is transferred from a personal computer to the PLC through a programming board which writes the program into a removable chip such as an EPROM

Functionality

The functionality of the PLC has evolved over the years to include sequential relay control, motion control, process control, distributed control systems, and networking. The data handling,

storage, processing power, and communication capabilities of some modern PLCs are approximately equivalent to desktop computers. PLC-like programming combined with remote I/O hardware, allow a general-purpose desktop computer to overlap some PLCs in certain applications. Desktop computer controllers have not been generally accepted in heavy industry because the desktop computers run on less stable operating systems than do PLCs, and because the desktop computer hardware is typically not designed to the same levels of tolerance to temperature, humidity, vibration, and longevity as the processors used in PLCs. Operating systems such as Windows do not lend themselves to deterministic logic execution, with the result that the controller may not always respond to changes of input status with the consistency in timing expected from PLCs. Desktop logic applications find use in less critical situations, such as laboratory automation and use in small facilities where the application is less demanding and critical, because they are generally much less expensive than PLCs.

Discrete and analog signals

Discrete signals behave as binary switches, yielding simply an On or Off signal (1 or 0, True or False, respectively). Push buttons, limit switches, and photoelectric sensors are examples of devices providing a discrete signal. Discrete signals are sent using either voltage or current, where a specific range is designated

as *On* and another as *Off*. For example, a PLC might use 24 V DC I/O, with values above 22 V DC representing *On*, values below 2VDC representing *Off*, and intermediate values undefined. Initially, PLCs had only discrete I/O.

Analog signals are like volume controls, with a range of values between zero and full-scale. These are typically interpreted as integer values (counts) by the PLC, with various ranges of accuracy depending on the device and the number of bits available to store the data. As PLCs typically use 16-bit signed binary processors, the integer values are limited between -32,768 and +32,767. Pressure, temperature, flow, and weight are often represented by analog signals. Analog signals can use voltage or current with a magnitude proportional to the value of the process signal. For example, an analog 0 to 10 V or 4-20 mA input would be converted into an integer value of 0 to 32767.

Current inputs are less sensitive to electrical noise (e.g. from welders or electric motor starts) than voltage inputs.

BIBLIOGRAPHY

[1] tratto da Wikipedia - l'enciclopedia libera. Anno 2016

[2] Testo adottato: Sistemi ed automazione/3 Autori:Natali-Aguzzi
Editore: Calderini