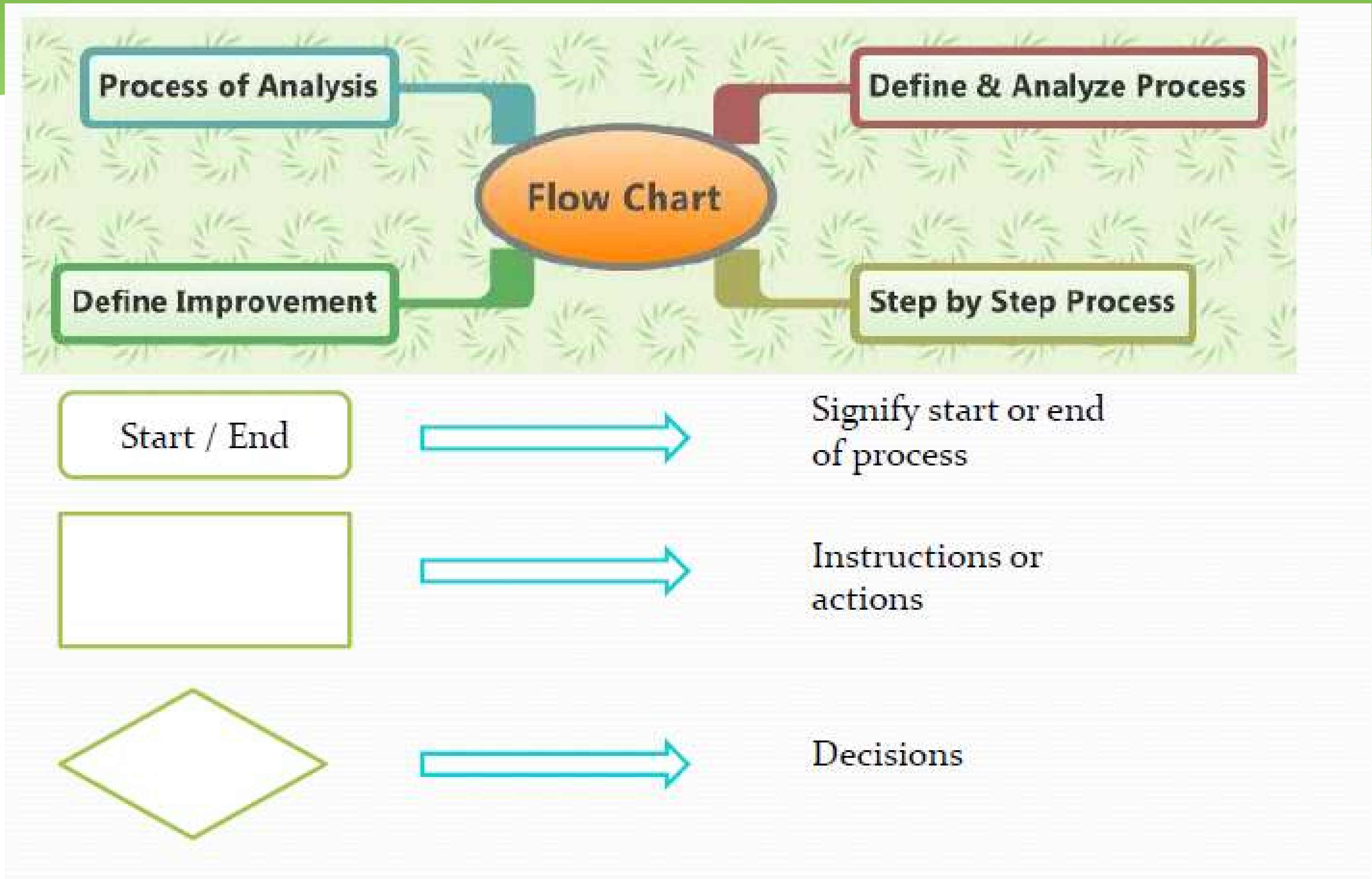
Basic PLC Progrmming

Outline

- Introduction to Programming Software
- Ladder Diagram
- Basic Logic Functions
- Mnuemonic Code
- CX-Programmer

Flowchart

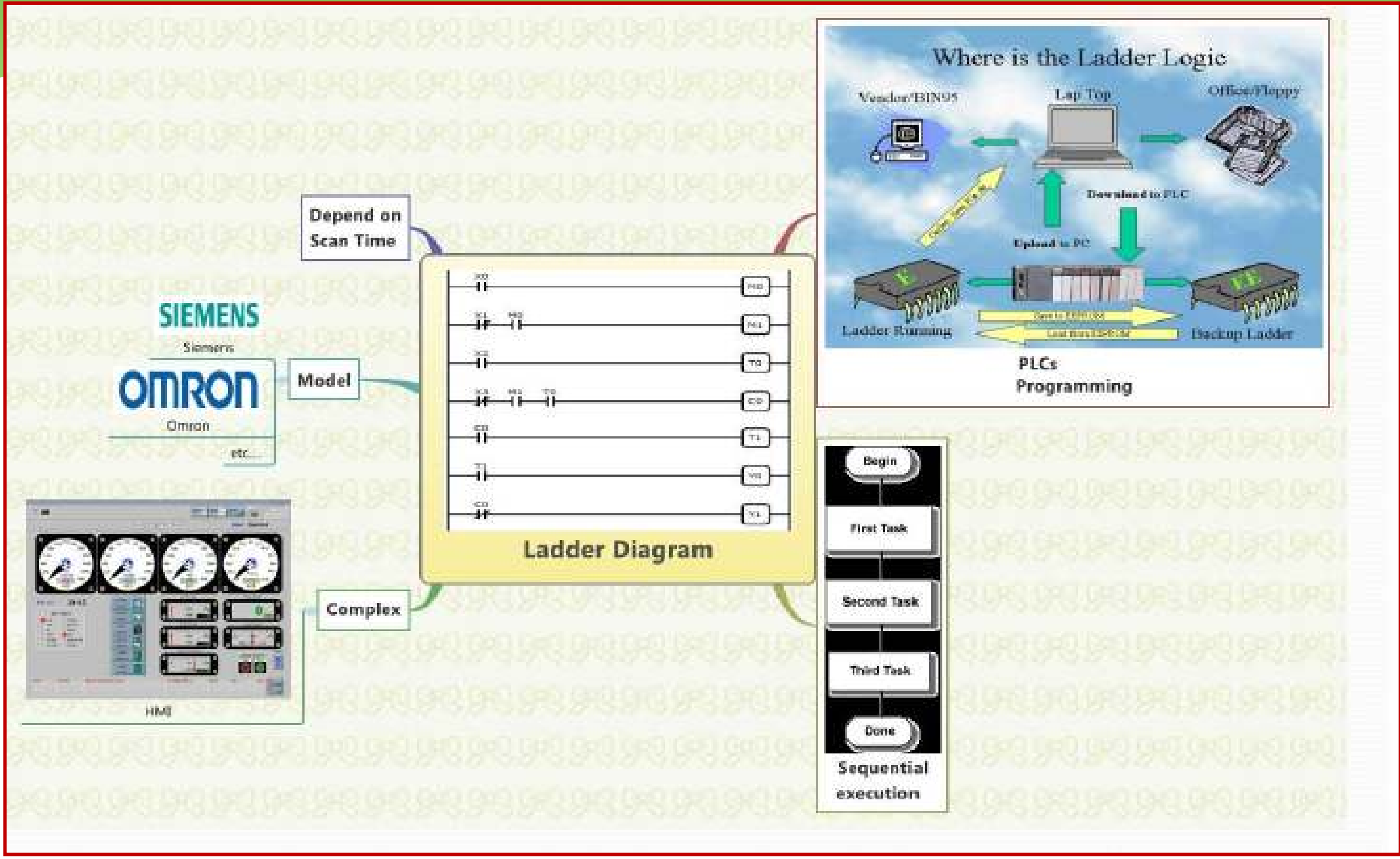


Process of Analysis
Define & Analyze Process
Flow Chart
Define Improvement
Step by Step Process
Start / End
Signify start or end of process
Instructions or actions
Decisions

Ladder Diagram

- Primary programming language for PLCs.
- Other programming methods include:
  - Function block diagrams (FBDs)
  - Structured text (ST)
  - Instruction List (IL)
  - Sequential function charts (SFCs)
- Visual and Graphical language unlike textual high-level, such as C, C++, Java…
- Derived form relay logic diagrams
- Primitive Logic Operations
  - OR
  - AND
  - NOT

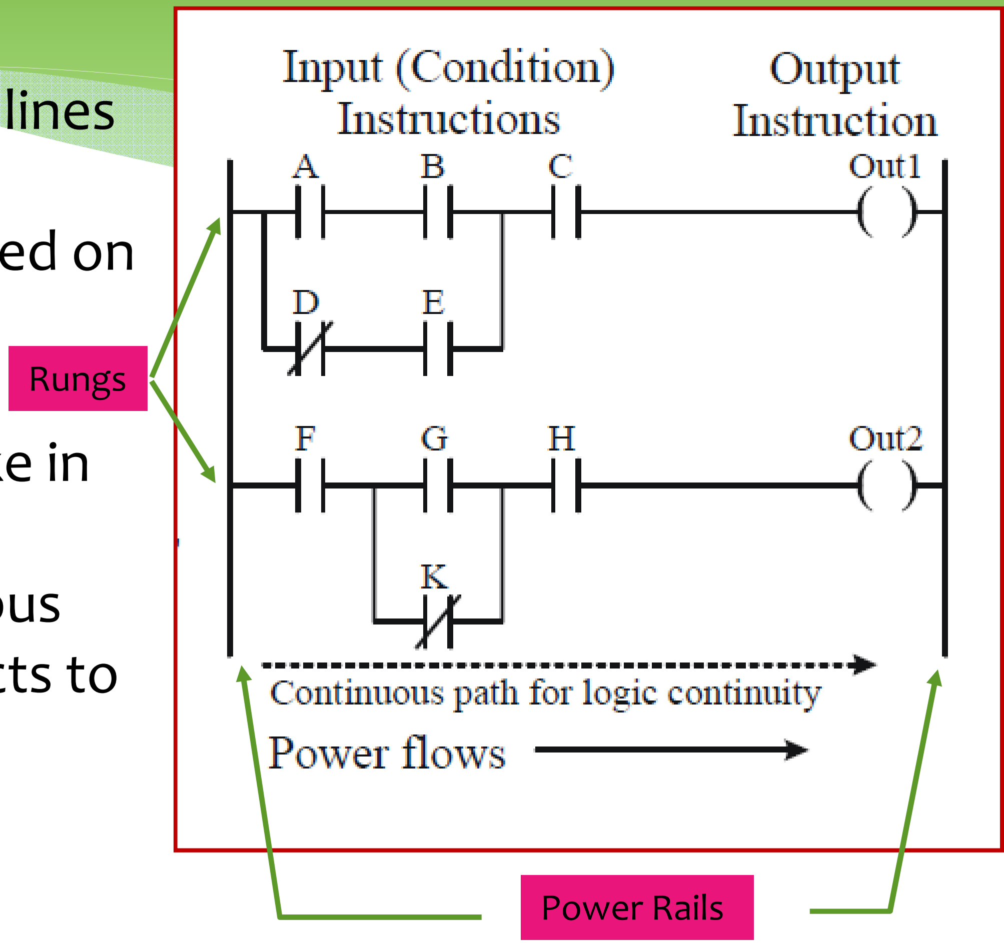
Ladder Diagram



Ladder Diagram

Terminologies

- Power Rails - Pair of vertical lines
- Rungs - Horizontal lines
- Contacts A, B, C, D… arranged on rungs
- Note in PLC Ladder Logic:
  - No Real Power Flow (like in relay ladder)
  - There must be continuous path through the contacts to energize the output



Ladder Diagram

Binary Input Devices

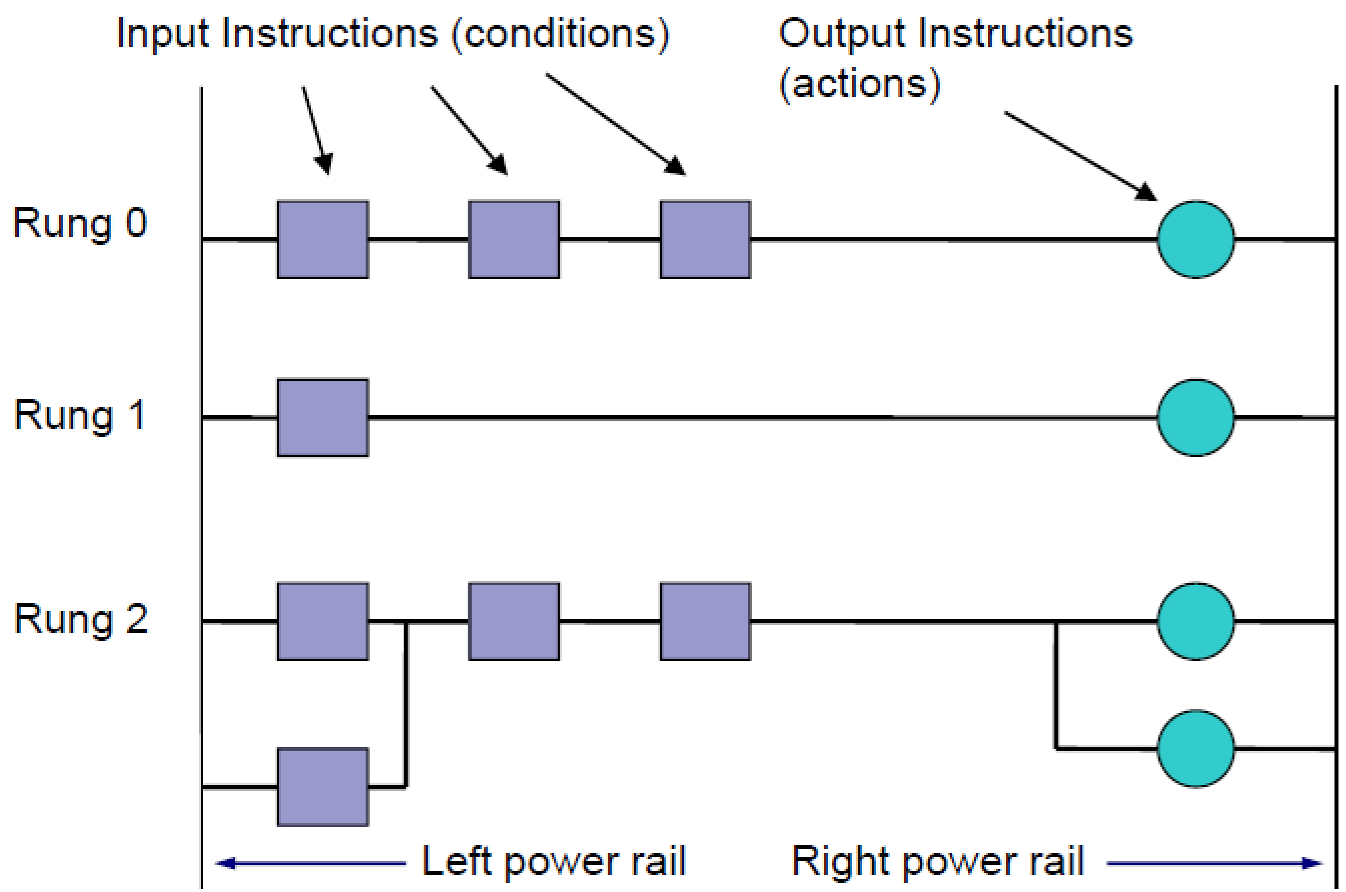
Device	One/Zero Interpretation
Limit switch	Contact/no contact
Photodetector	Contact/no contact
Pushbutton switch	On/off
Timer	On/off
Control relay	Contact/no contact
Circuit breaker	Contact/no contact

Ladder Diagram

Binary Output Devices

Device	One/Zero Interpretation
Motor	On/off
Alarm buzzer	On/off
Lights	On/off
Control relay	Contact/no contact
Valves	Closed/open
Clutch	Engaged/not engaged
Solenoid	Energised/not energised

Anatomy of Ladder Diagram

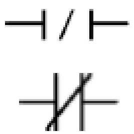


Anatomy of Ladder Diagram

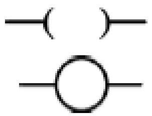
- Input instructions are entered on the left
- Output instructions are entered on the right
- The power rails simulate the power supply lines
  - L1 and L2 for AC circuits and +24V and ground for DC circuits
- Most PLCs allow more than one output per rung
- The processor (or "controller") scans ladder rungs from top-to-bottom and from left-to-right.
  - The basic sequence is altered whenever jump or subroutine instructions are executed.

Basic Ladder Logic Symbols

Normally open contact
Passes power (ON) if coil driving the contact is ON (closed)
Allen-Bradley calls it **XIC** - eXamine If Closed

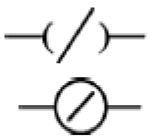


Normally closed contact
Passes power (ON) if coil driving the contact is off (open)
Allen-Bradley calls it **XIO** - e**X**amine **I**f **O**pen



Output or coil

If any left-to-right path of inputs passes power, output is energized
Allen-Bradley calls it OTE - OuTput Energize



Not Output or coil

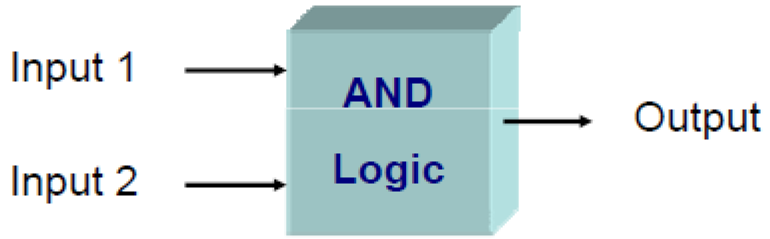
If any left-to-right path of inputs passes power, output is de-energized

Logic Functions

- PLC programming is a logical procedure
- In a PLC program, “things” (inputs and rungs) are either TRUE or FALSE
- If the proper input conditions are **TRUE**:
  - The rung becomes TRUE and an output action occurs (for example, a motor turns on)
- If the proper input conditions are **not TRUE**:
  - The rung becomes FALSE and an output action does not occur

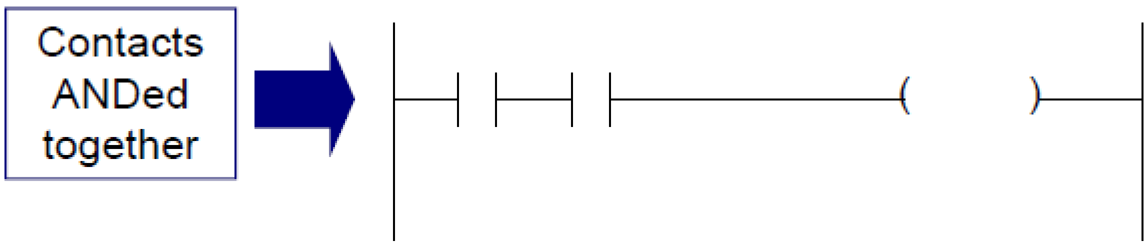
Logic Functions

❑ AND



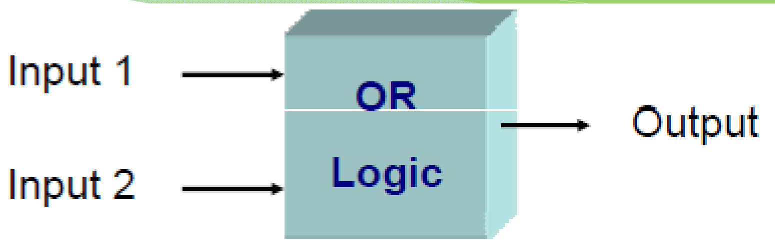
Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

0 → False
1 → True



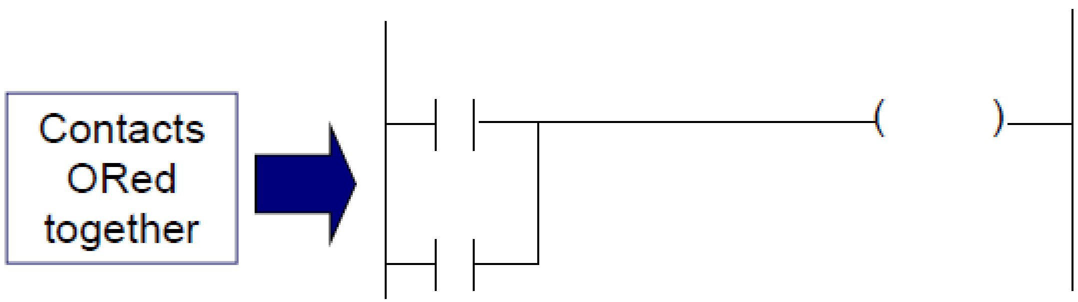
Logic Functions

OR



Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

0 → False
1 → True

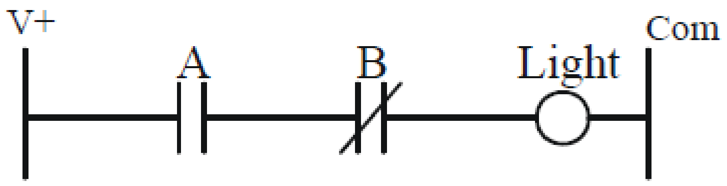


Logic Functions

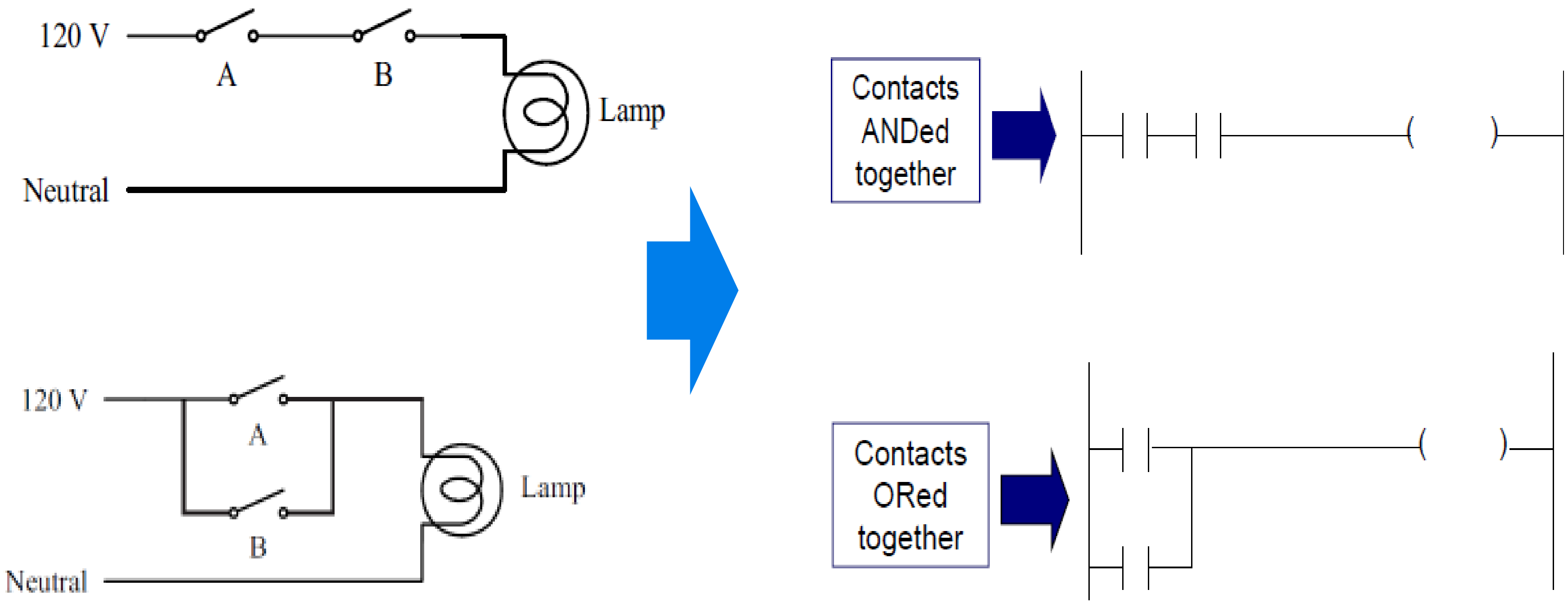
- NOT

NOT Truth Table

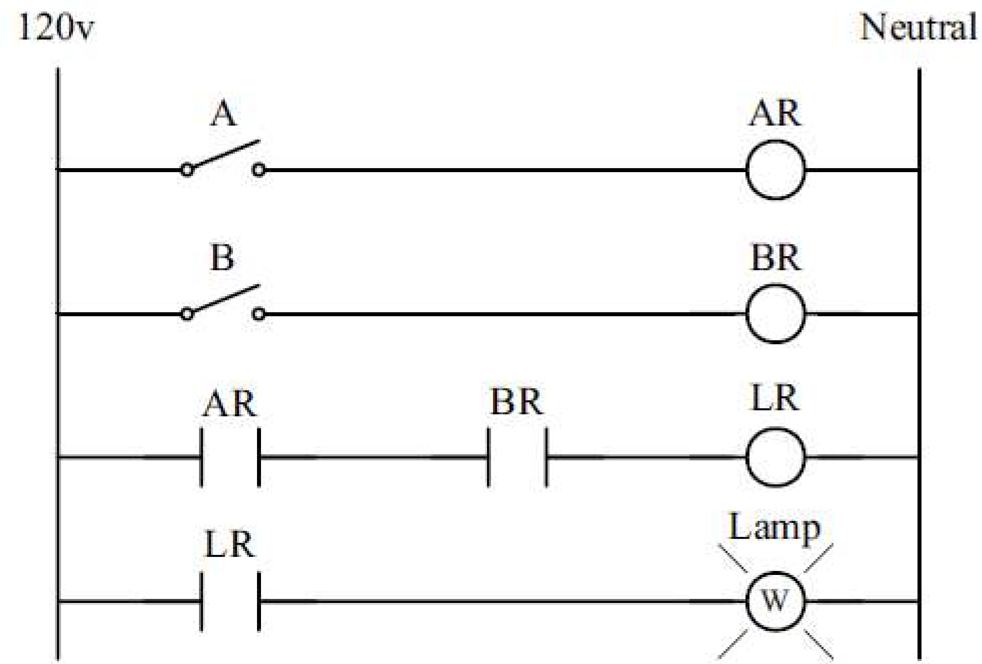
A	B	Light
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	ON
ON	ON	OFF

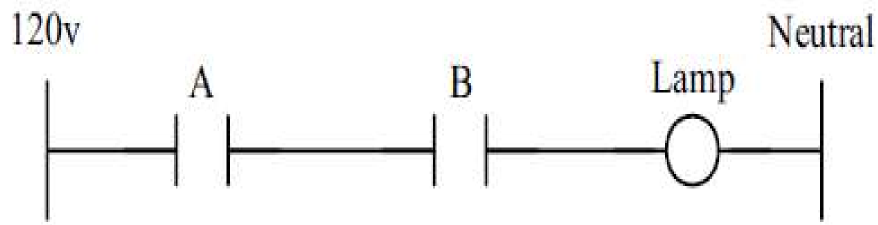


Relay to Ladder Diagram



Relay to Ladder Diagram



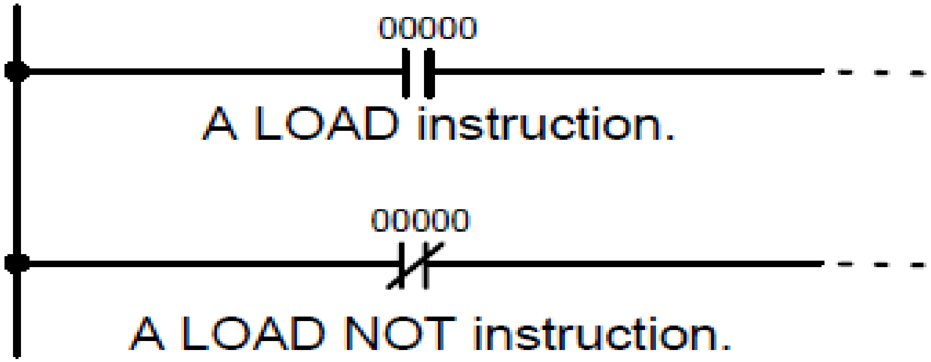


Mneumonic Codes

- These instructions can be derived directly from the ladder logic diagrams and entered into the PLC through a simple programming terminal.
- Ladder logic diagrams can be read by the programming console
- For this reason, ladder diagrams need to be converted into mnuemonic codes that provides same information as ladder diagrams and to be typed directly using programming console.

Examples of Mneumonic Codes

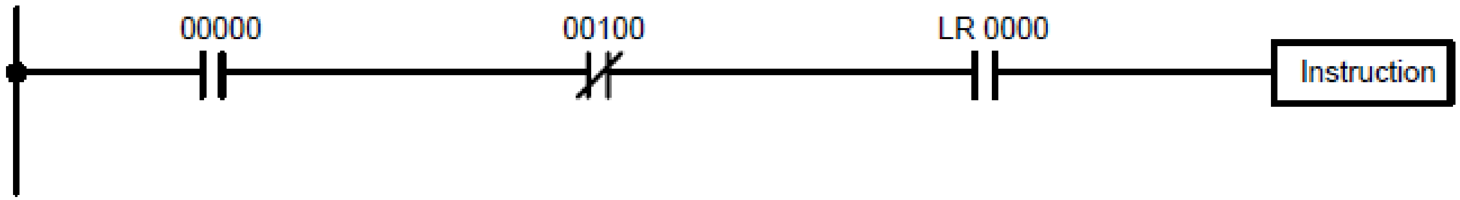
❑ LOAD and LOAD Not



Address	Instruction	Operands
00000	LD	00000
00001	Instruction	
00002	LD NOT	00000
00003	Instruction	

Examples of Mneumonic Codes

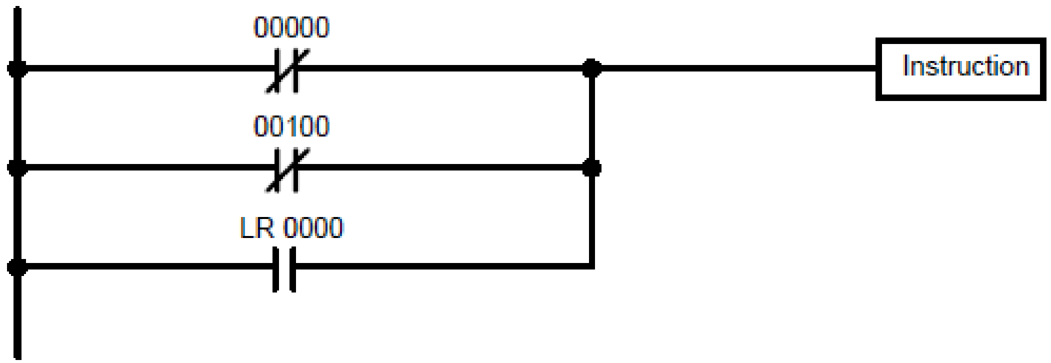
❑ AND and AND Not



Address	Instruction	Operands	
00000	LD		00000
00001	AND NOT		00100
00002	AND	LR	0000
00003	Instruction		

Examples of Mneumonic Codes

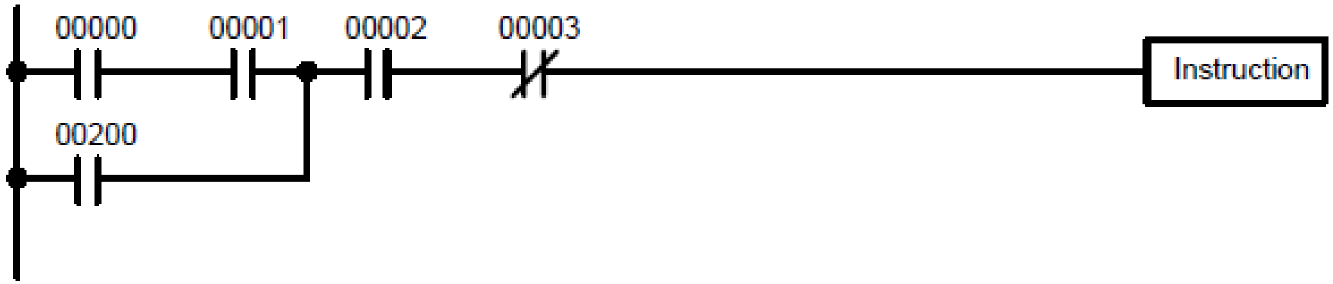
OR and OR Not



Address	Instruction	Operands
00000	LD NOT	00000
00001	OR NOT	00100
00002	OR	LR 0000
00003	Instruction	

Examples of Mneumonic Codes

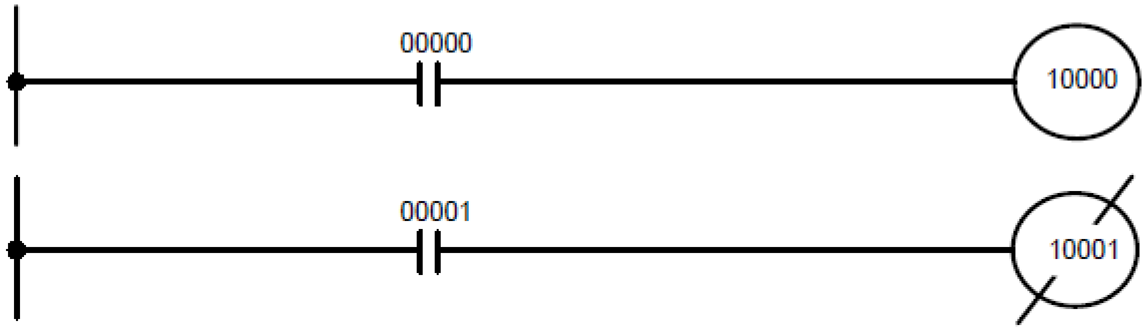
- Combine AND and OR



Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	OR	00200
00003	AND	00002
00004	AND NOT	00003
00005	Instruction	

Examples of Mneumonic Codes

- OUTPUT and OUTPUT Not

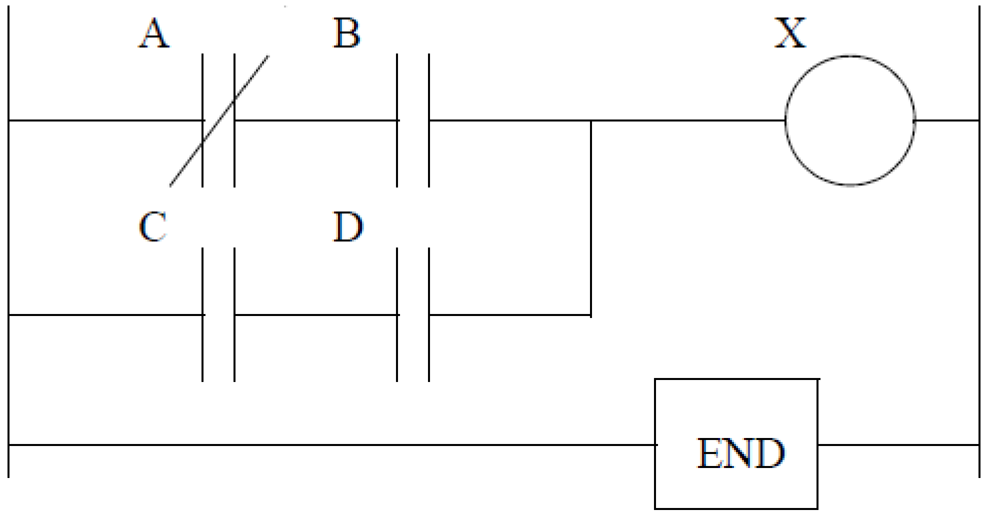


Address	Instruction	Operands
00000	LD	00000
00001	OUT	10000

Address	Instruction	Operands
00000	LD	00001
00001	OUT NOT	10001

Examples of Mneumonic Codes

- Write the mnuemonic code for the following ladder diagram

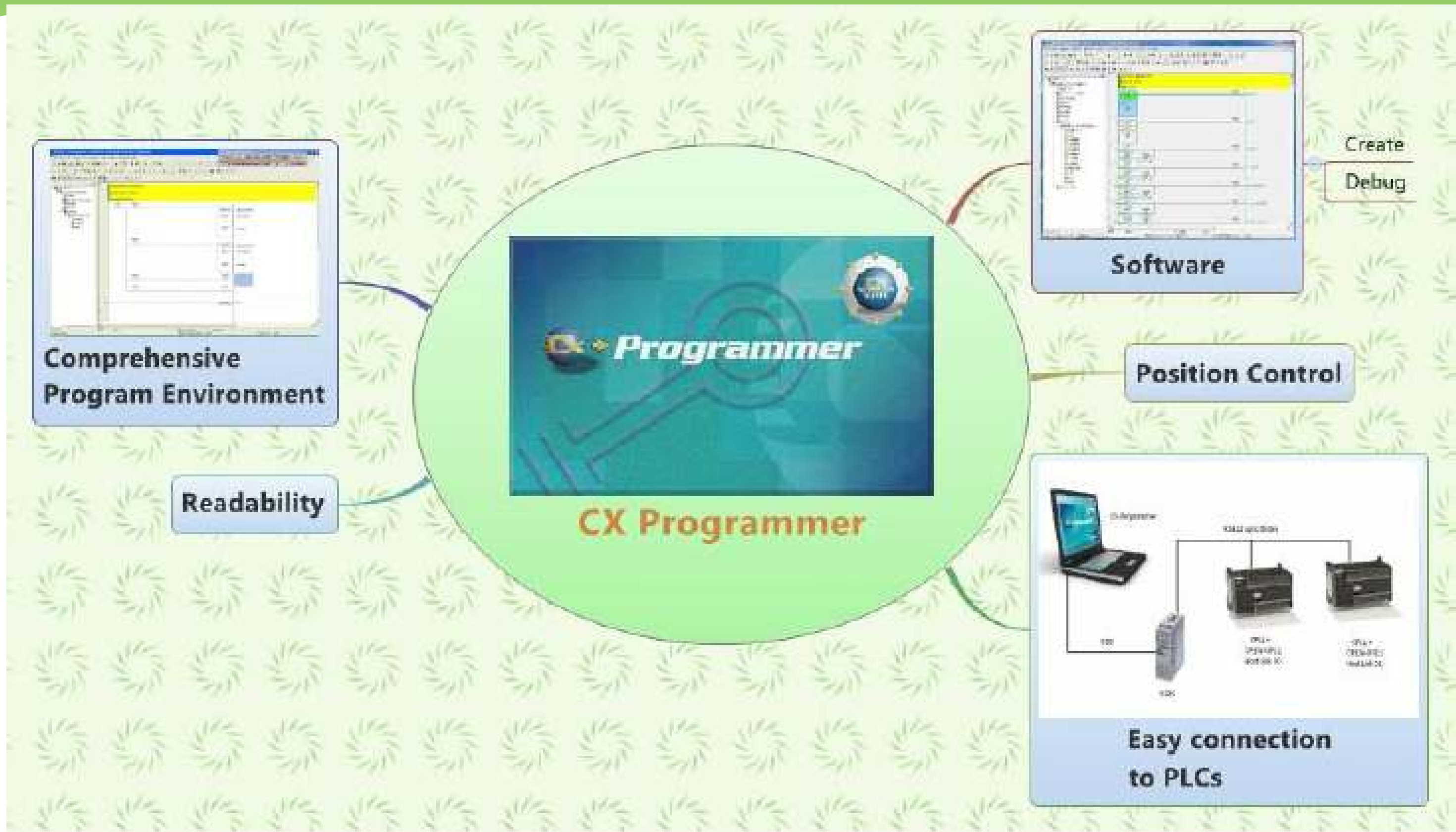


Entering the Ladder Diagram: CX Programmer

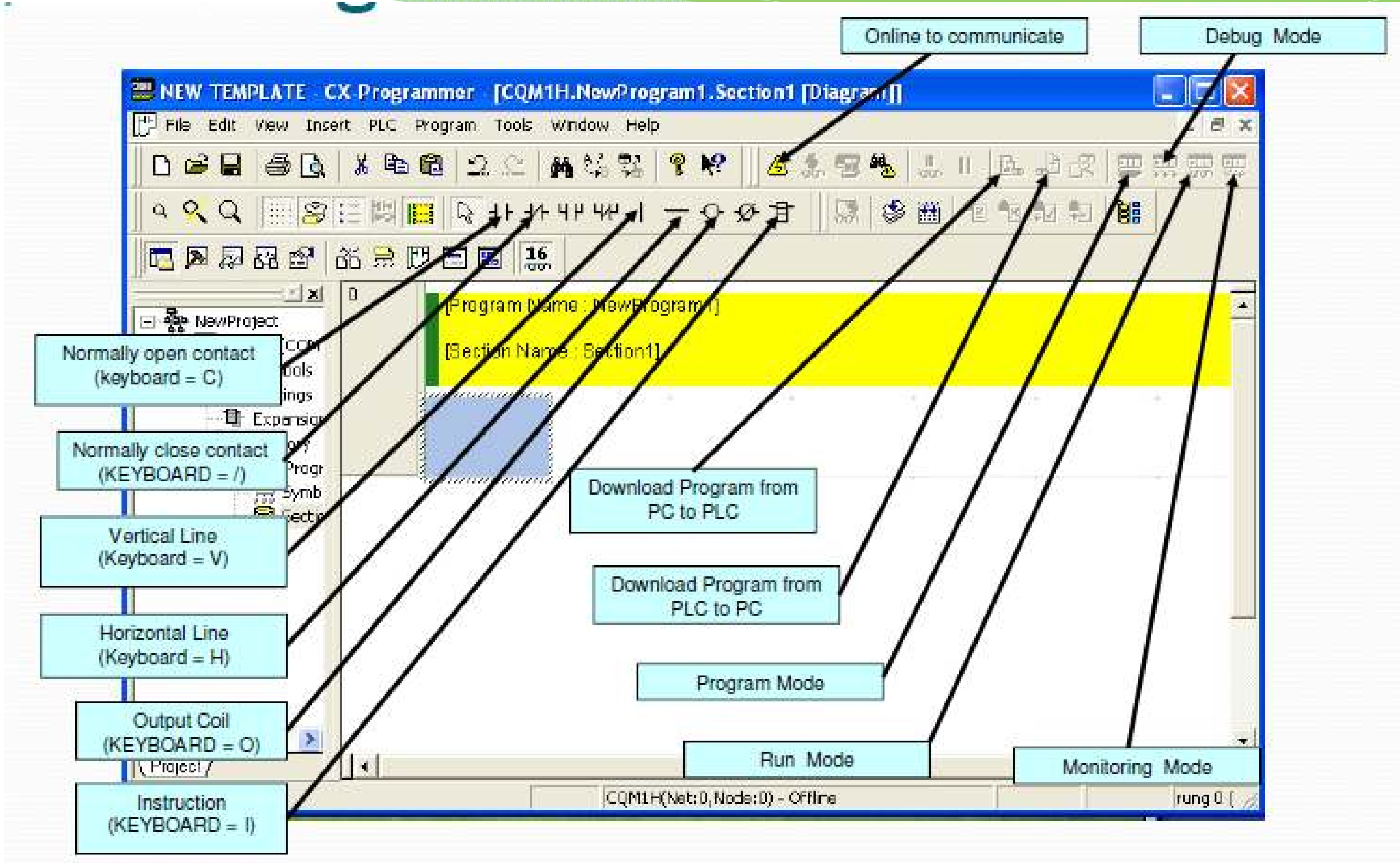
- CX-Programmer, the programming software for all Omron's PLC series, is fully integrated into the CX-One software suite.

- CX-Programmer includes a wide variety of features to speed up the development of your PLC program. New parameter-setting dialogues reduce setup time, and with standard function blocks in IEC 61131-3 structured text or conventional ladder language, CX-Programmer makes development of PLC programs a simple drag & drop configuration.

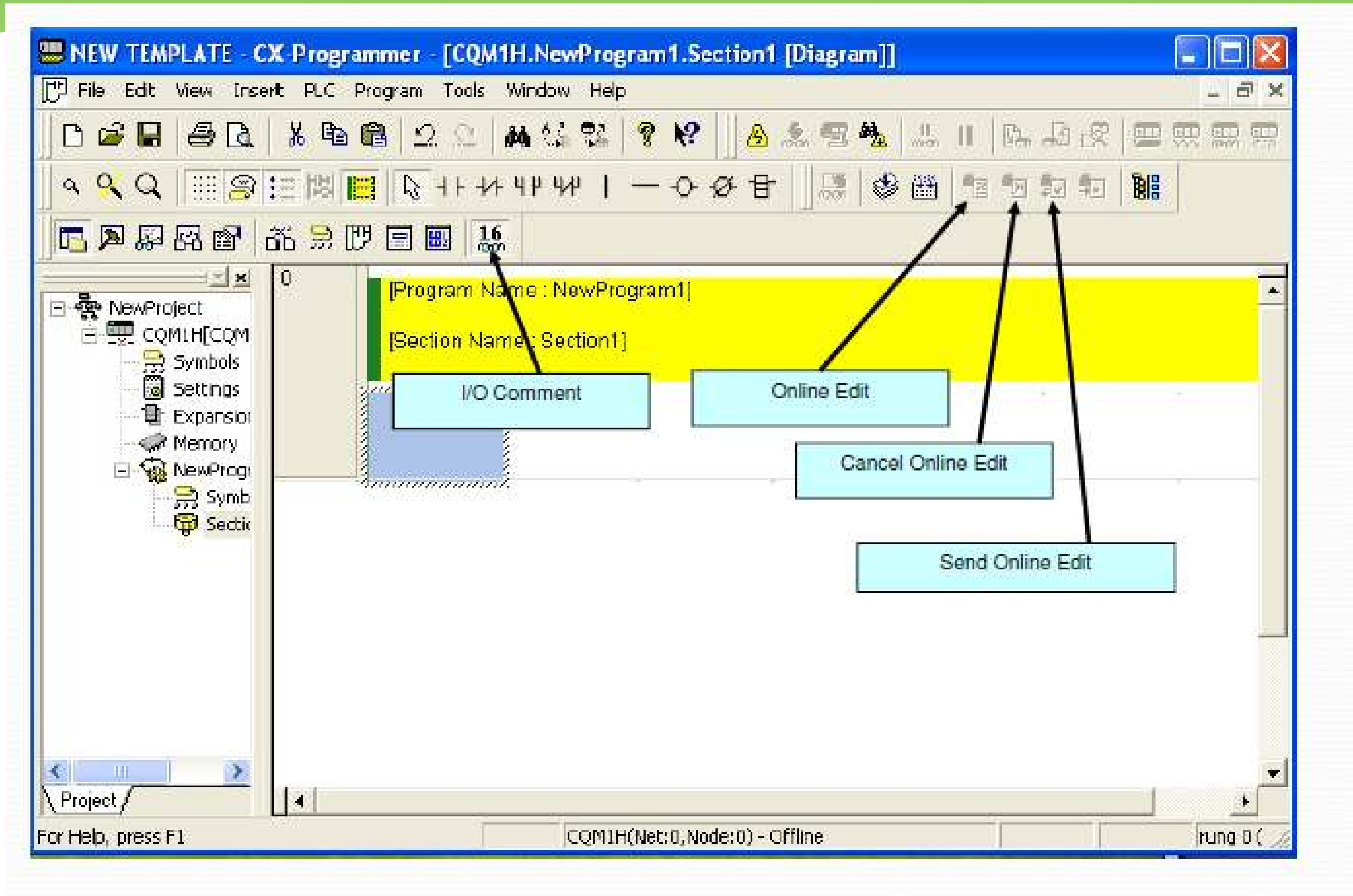
CX Programmer



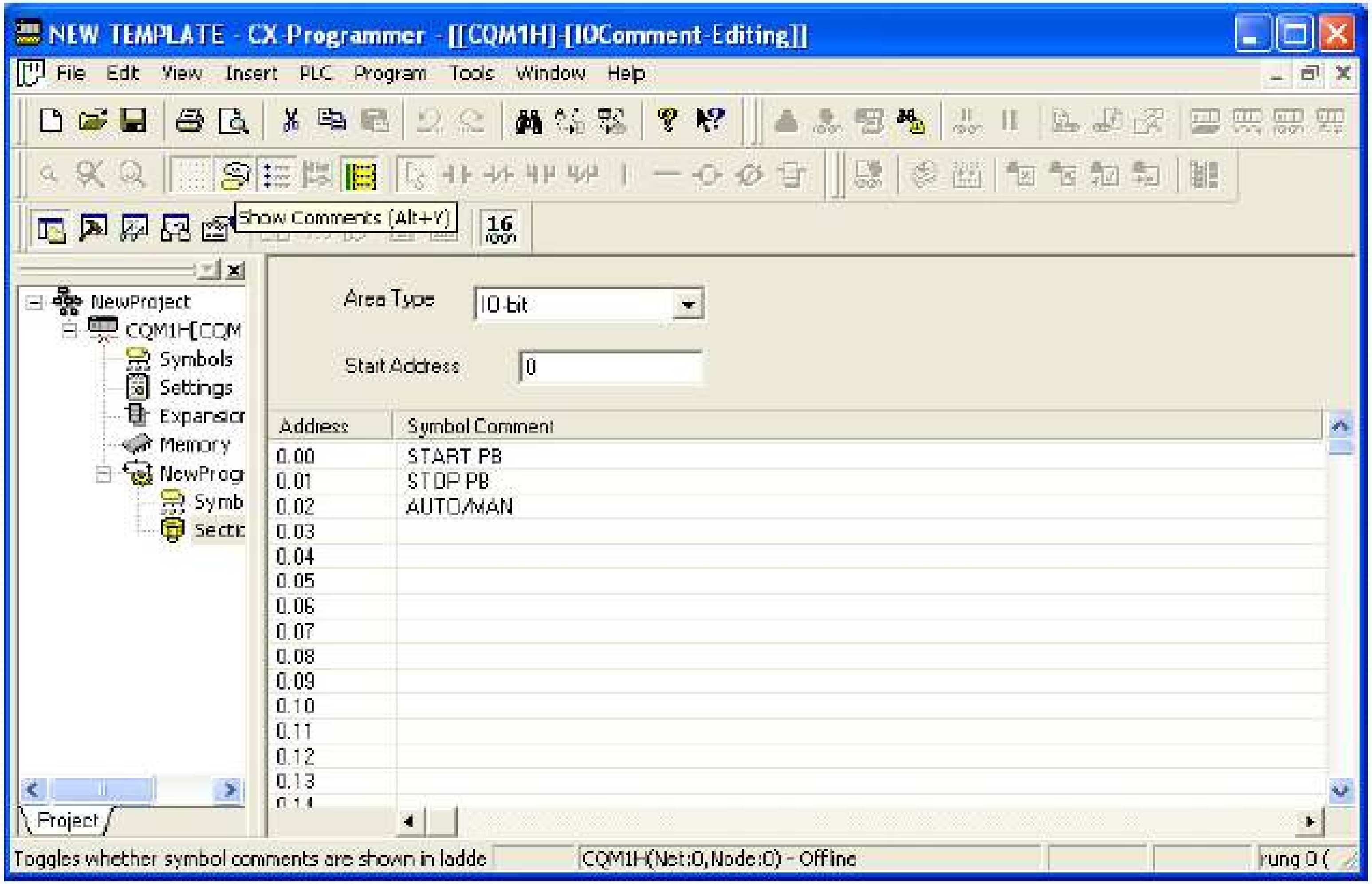
CX Programmer: Overview



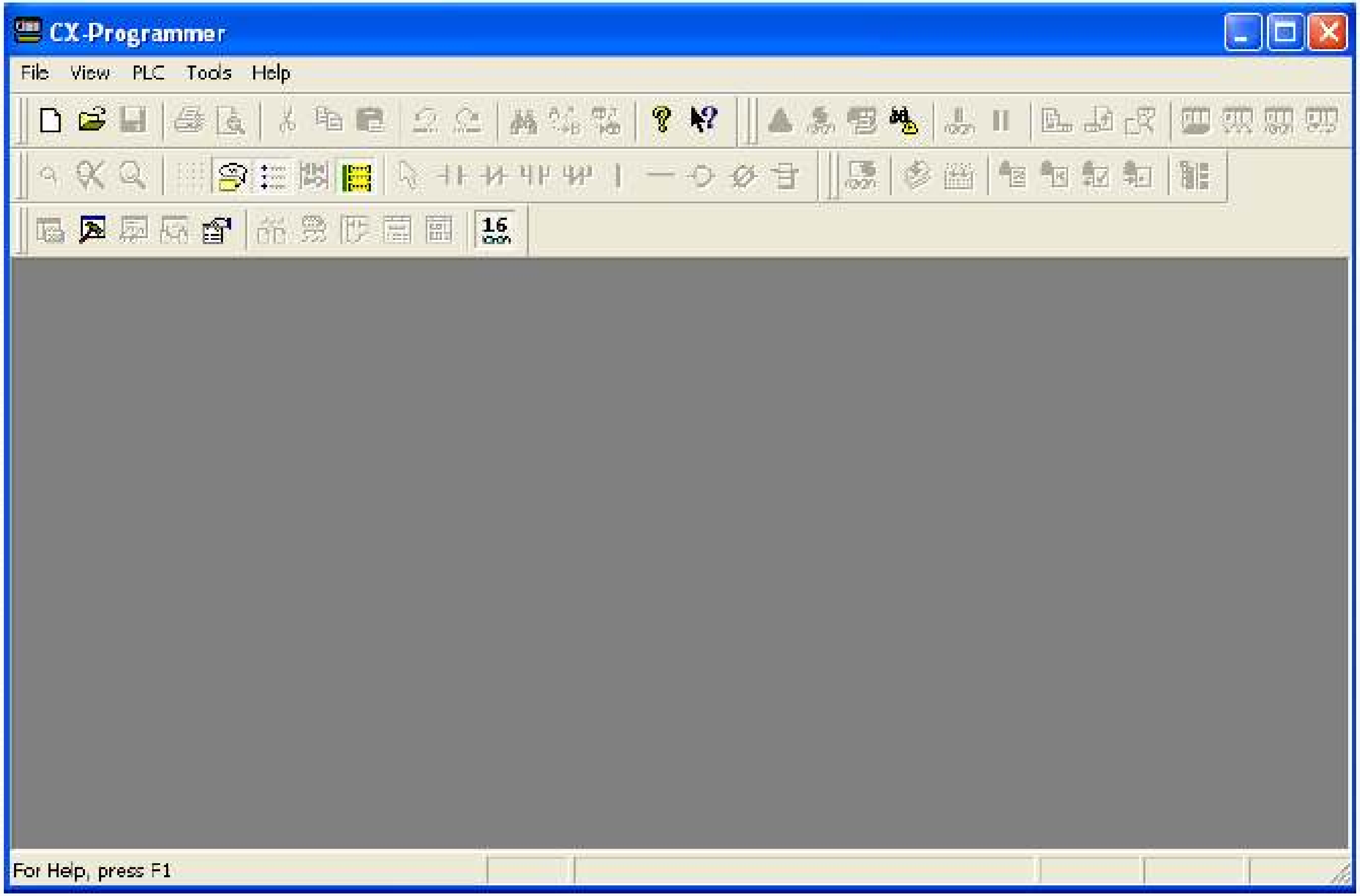
CX Programmer: Overview



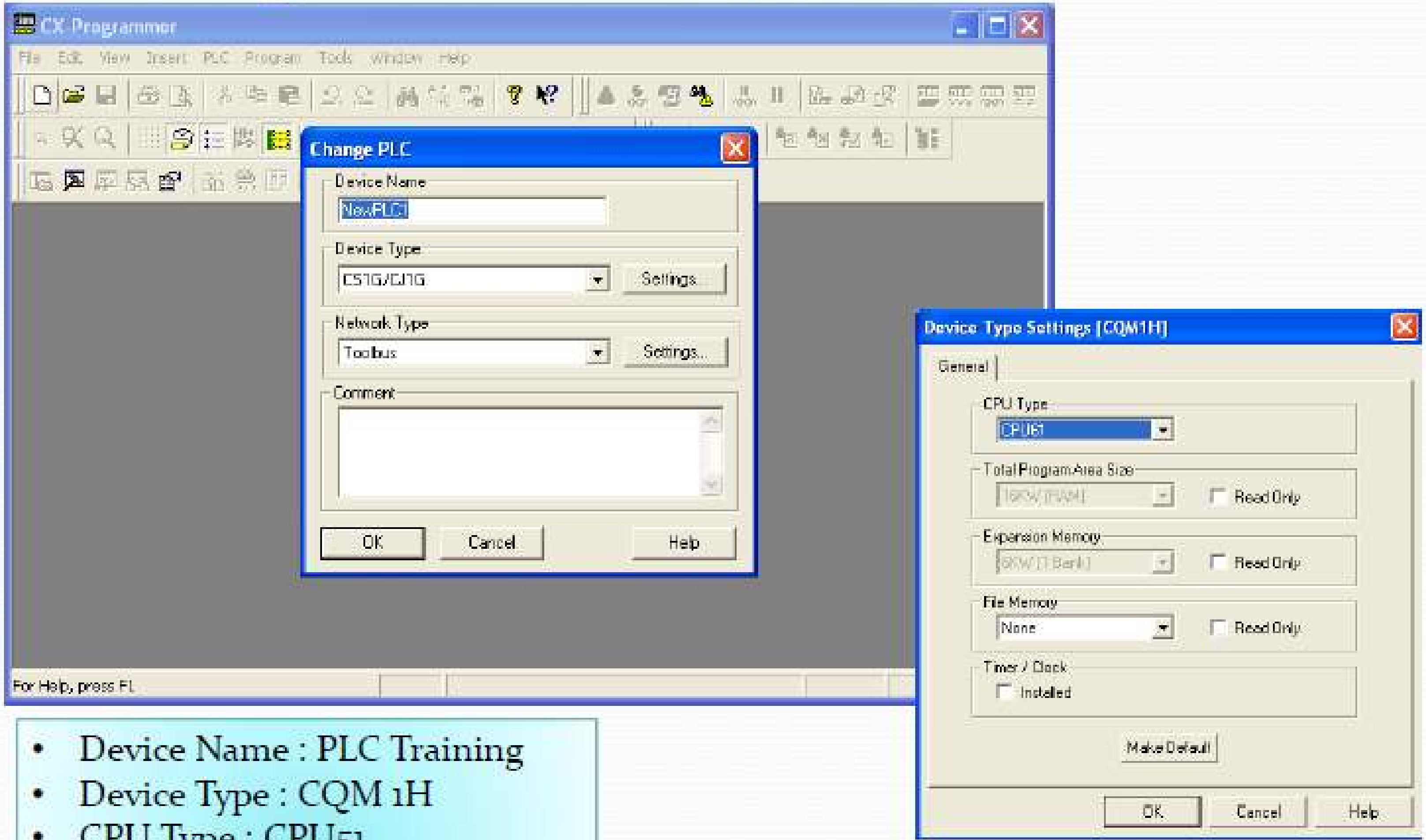
CX Programmer: Input/Output



Opening New File

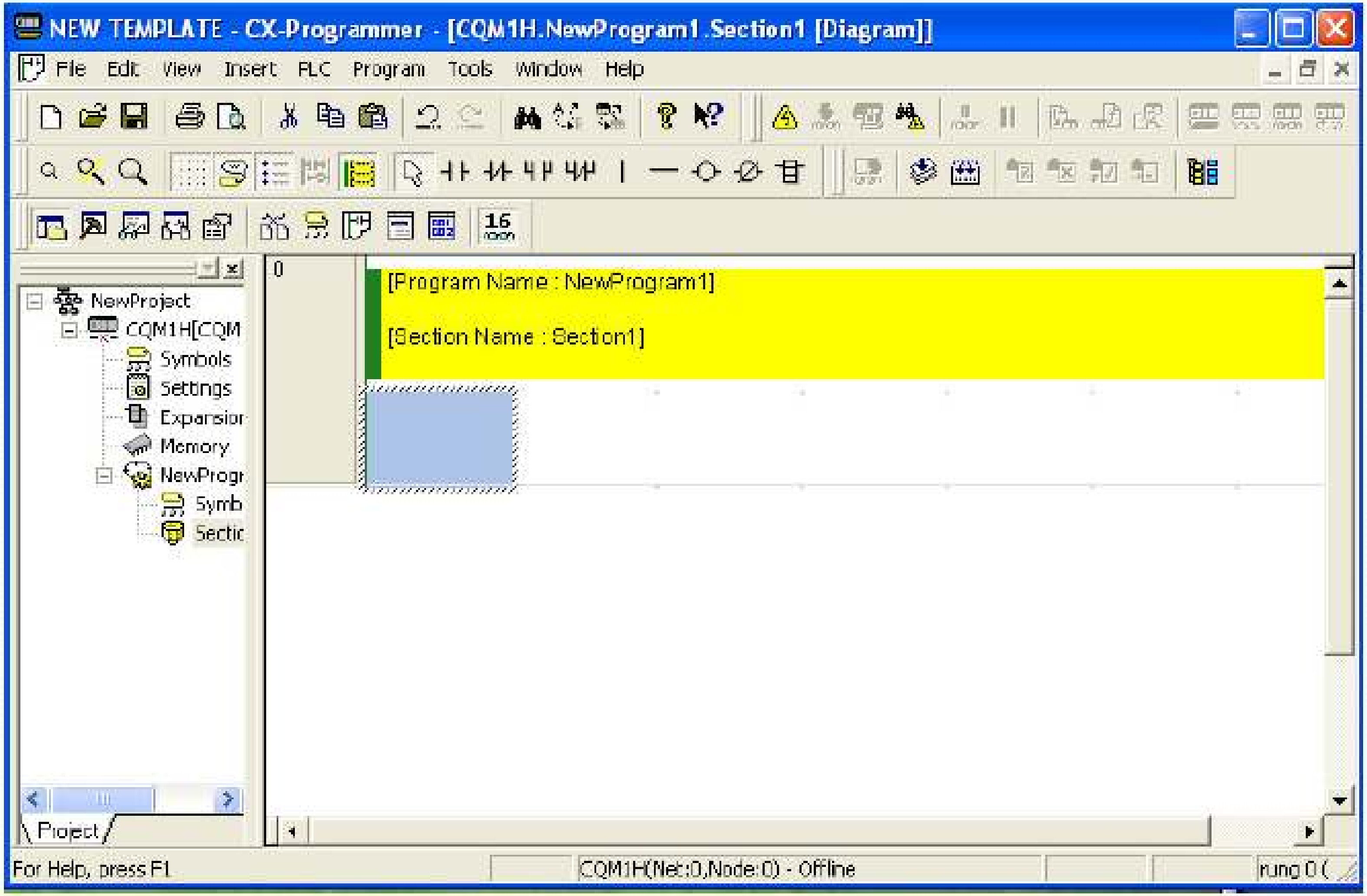


Configure PLC Devices



- Device Name : PLC Training
- Device Type : CQM 1H
- CPU Type : CPU51

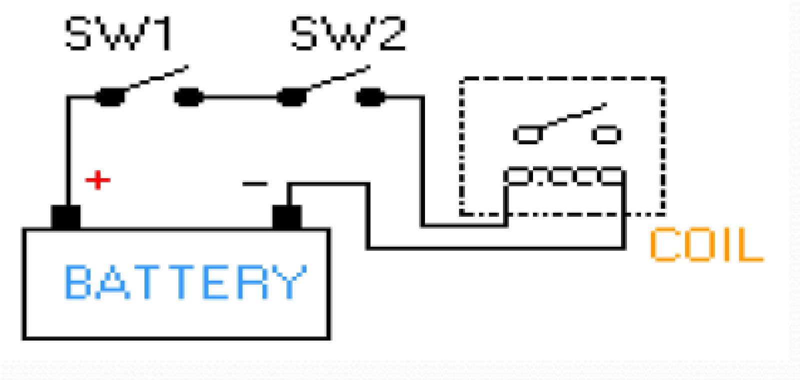
CX Programmer: Programming

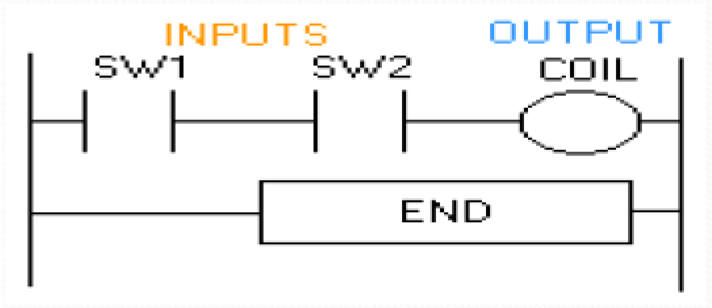


PLC Mode

	Program	Monitor	Run
Delete	✓	✗	✗
Transfer	✓	✗	✗
On line edit	✓	✓	✗
Timer/Counter/DM SV change	✓	✓	✗
Run Indicator	OFF	ON	ON

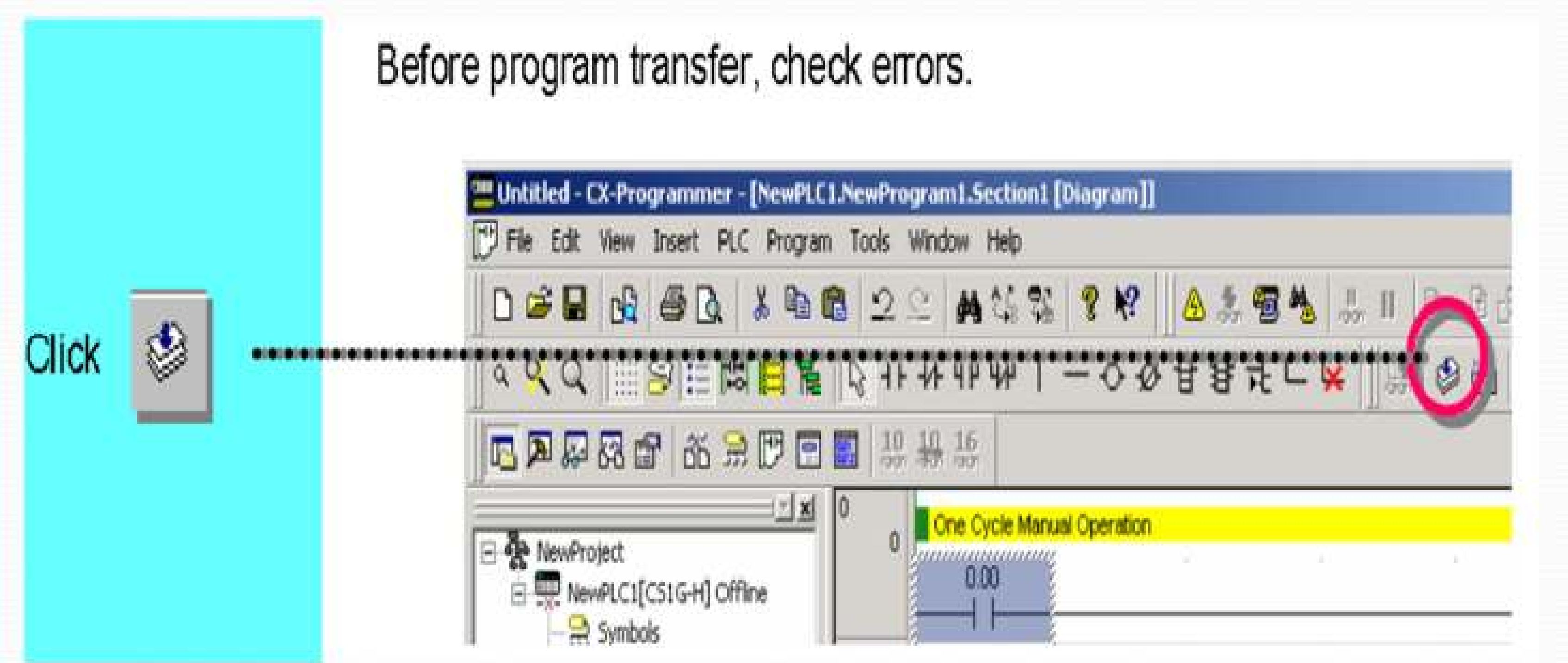
Exercise





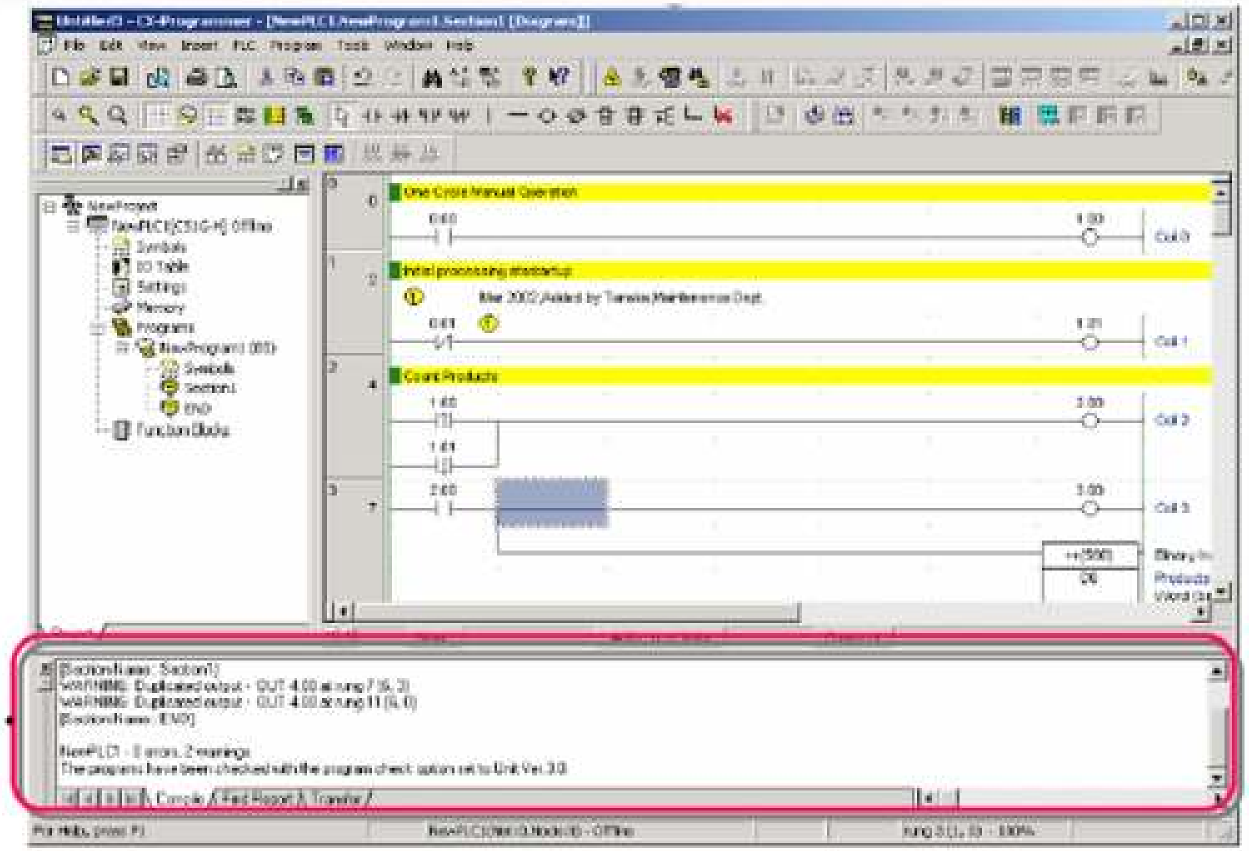
- Draw process flowchart for a given system
- Create the ladder diagran in CX Programmer

Program Error Check (Compile)



Program Error Check (Compile)

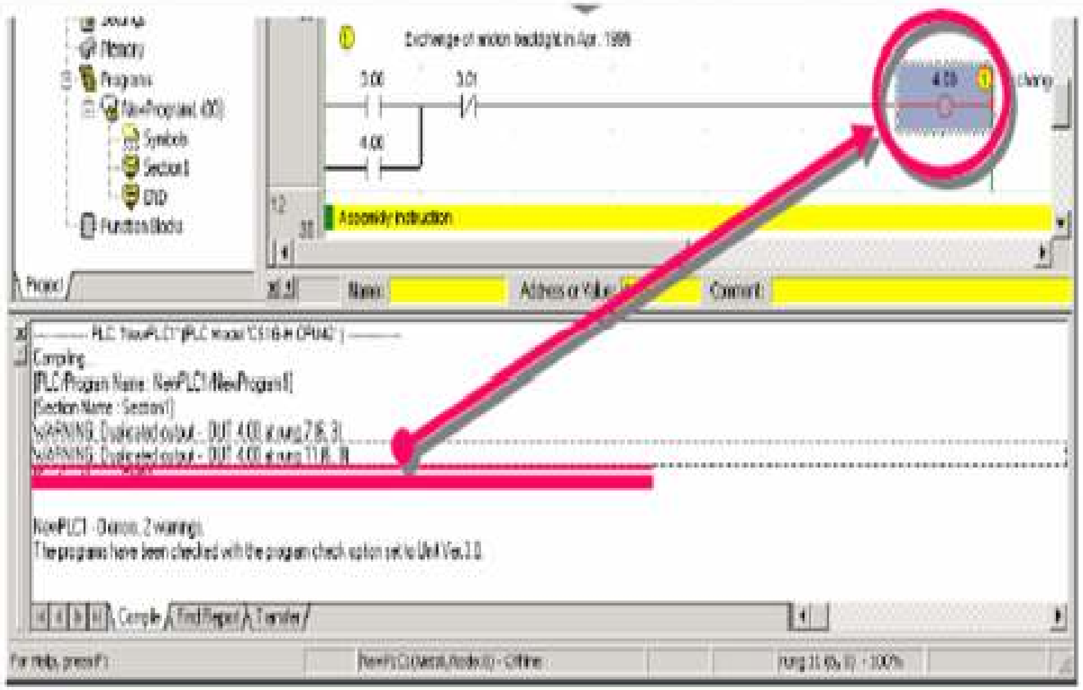
Errors and addresses are displayed on Output Window.



Program Error Check (Compile)

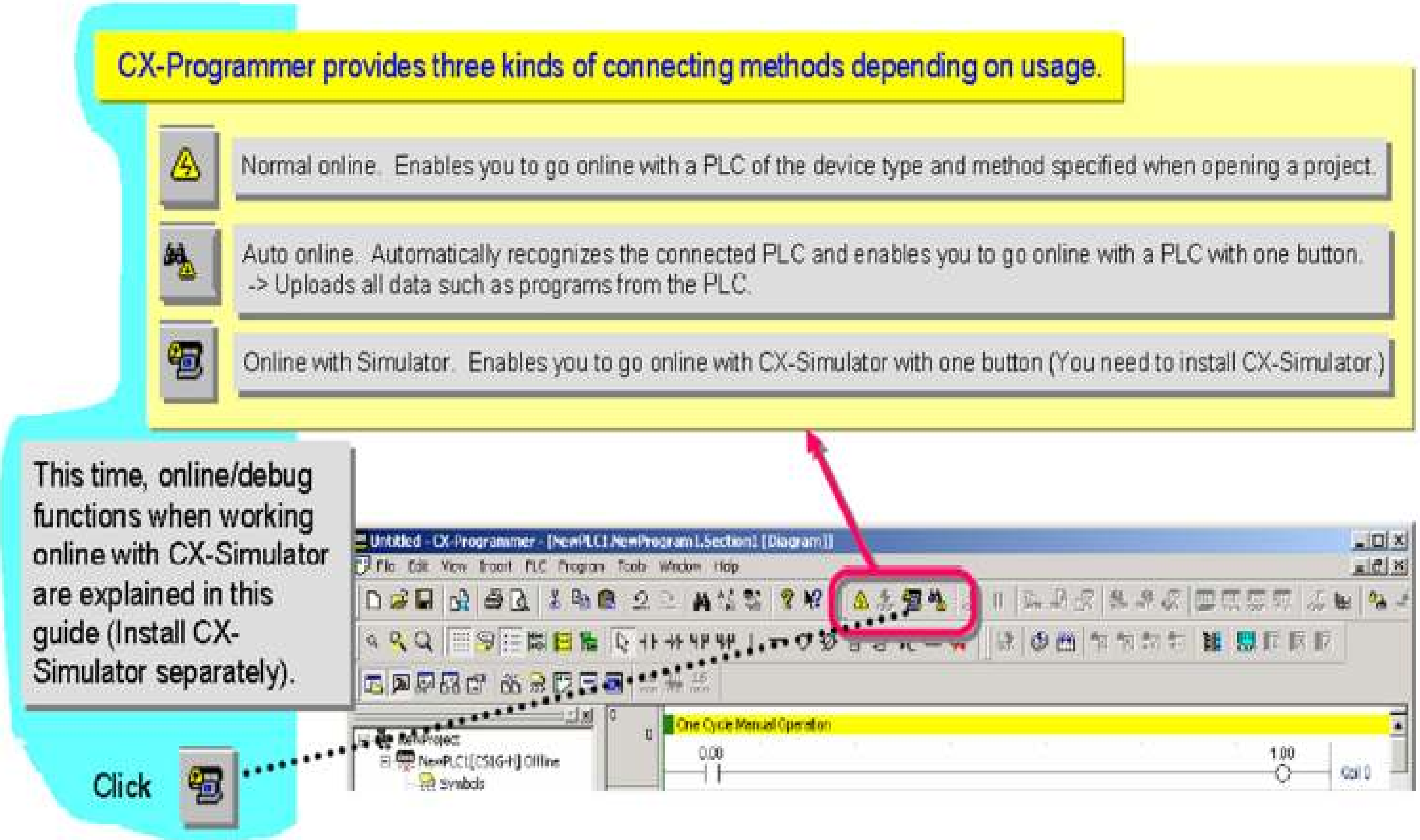
Double-click a displayed error, and the cursor in Ladder Diagram will go to the corresponding error location and the error rung will be shown in red.

Modify the error.

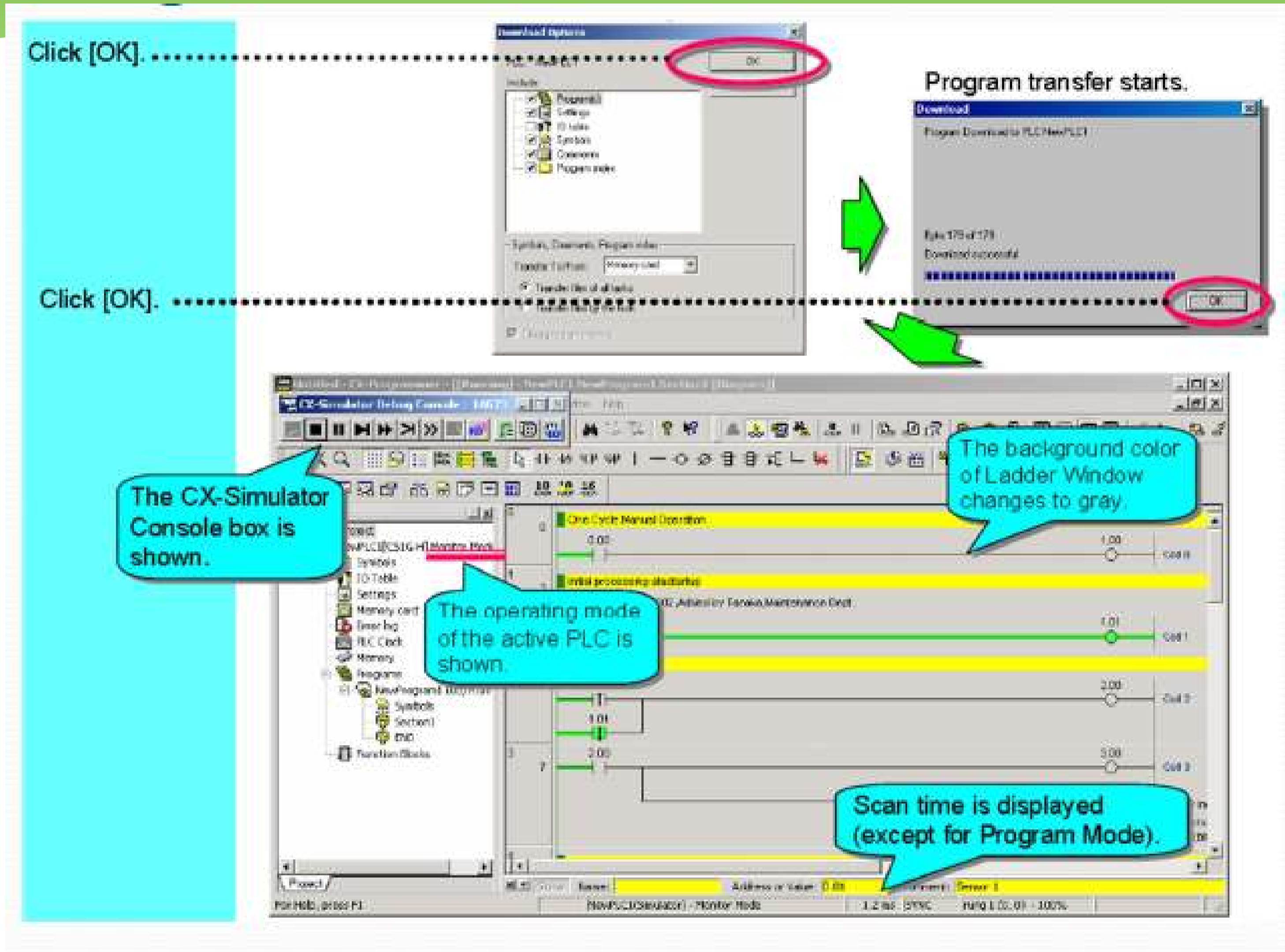


- Output Window automatically opens at program check.
- The cursor moves to an error location by pressing J or F4 key.
- Output Window closes by pressing the ESC key.

Going Online



Going Online



Introduction to Programmable Logic Controllers (PLC's)

Industrial Control Systems

Fall 2006

The Need for PLCs

- Hardwired panels were very time consuming to wire, debug and change.
- GM identified the following requirements for computer controllers to replace hardwired panels.
  - Solid-state not mechanical
  - Easy to modify input and output devices
  - Easily programmed and maintained by plant electricians
  - Be able to function in an industrial environment

The First Programmable Logic Controllers (PLCs)

- Introduced in the late 1960's
- Developed to offer the same functionality as the existing relay logic systems
- Programmable, reusable and reliable
  - Could withstand a harsh industrial environment
  - They had no hard drive, they had battery backup
  - Could start in seconds
  - Used Ladder Logic for programming

Programmable Logic Controller

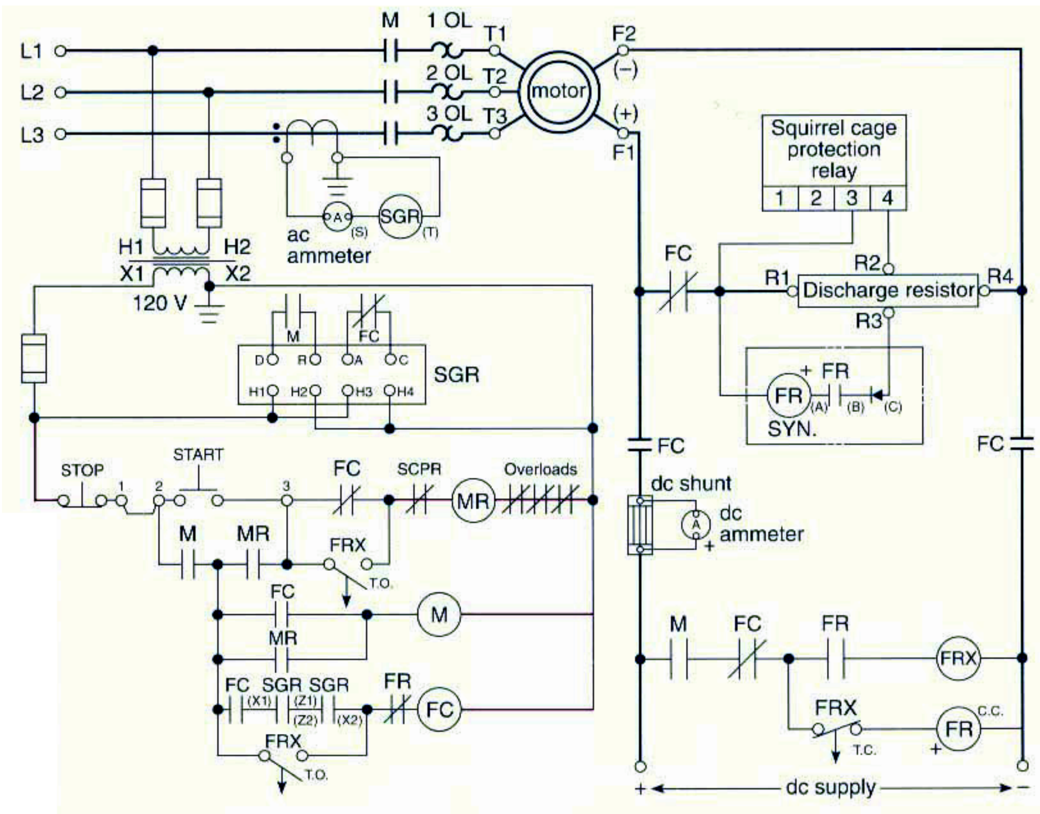
- A programmable logic controller (PLC) is a specialized computer used to control machines and process.
- It uses a programmable memory to store instructions and specific functions that include On/Off control, timing, counting, sequencing, arithmetic, and data handling

Advantages of PLC Control Systems

- Flexible
- Faster response time
- Less and simpler wiring
- Solid-state - no moving parts
- Modular design - easy to repair and expand
- Handles much more complicated systems
- Sophisticated instruction sets available
- Allows for diagnostics “easy to troubleshoot”
- Less expensive

Advantages of a PLC Control System

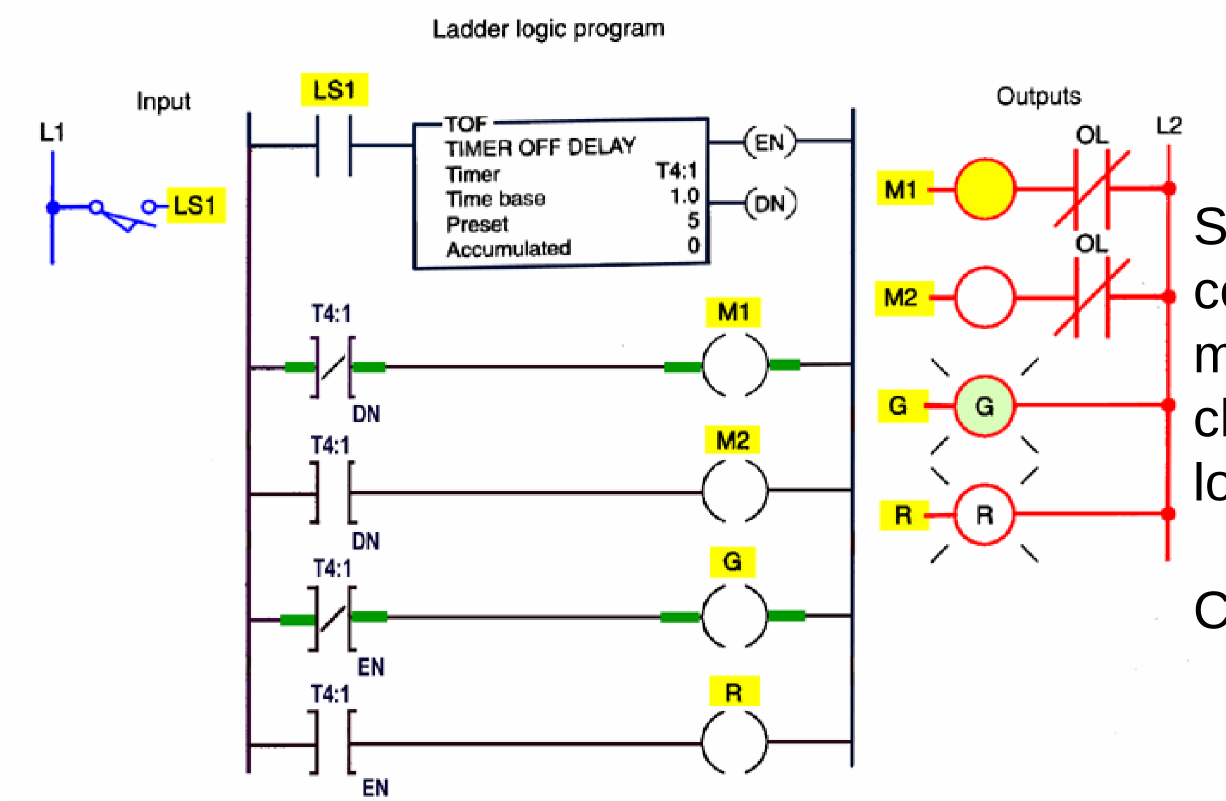
Eliminates much of the hard wiring that was associated with conventional relay control circuits.



The program takes the place of much of the external wiring that would be required for control of a process.

Advantages of a PLC Control System

Increased Reliability:
Once a program has been written and tested it can be downloaded to other PLCs.

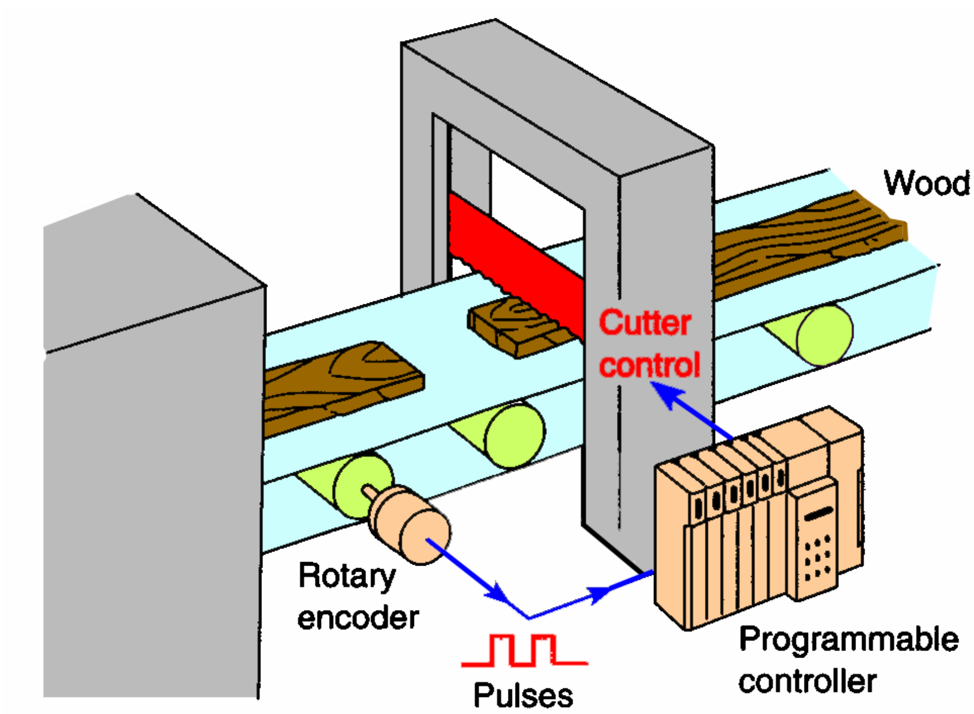


Since all the logic is contained in the PLC's memory, there is no chance of making a logic wiring error.

Conversely ......

Advantages of a PLC Control System

More Flexibility:
Original equipment manufacturers (OEMs) can provide system updates for a process by simply sending out a new program.

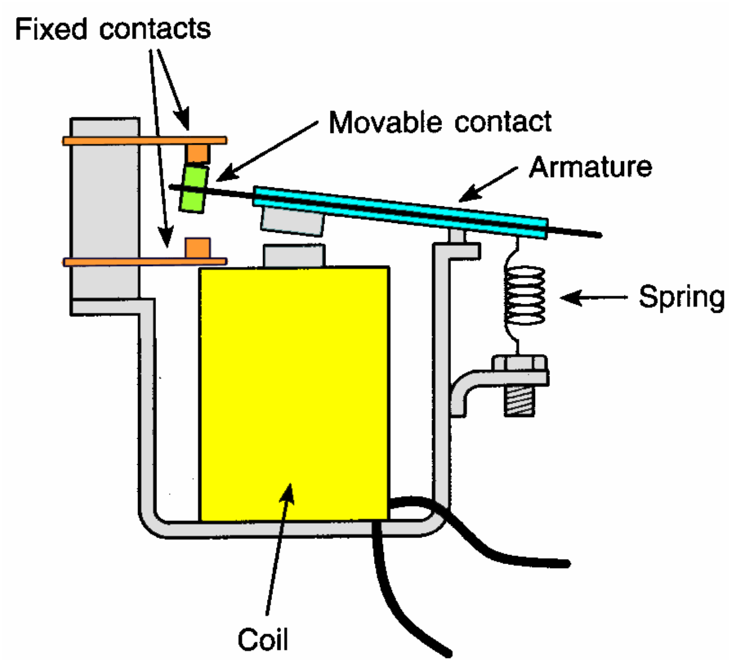


It is easier to create and change a program in a PLC than to wire and rewire a circuit. End-users can modify the program in the field.

Advantages of a PLC Control System

Lower Costs:

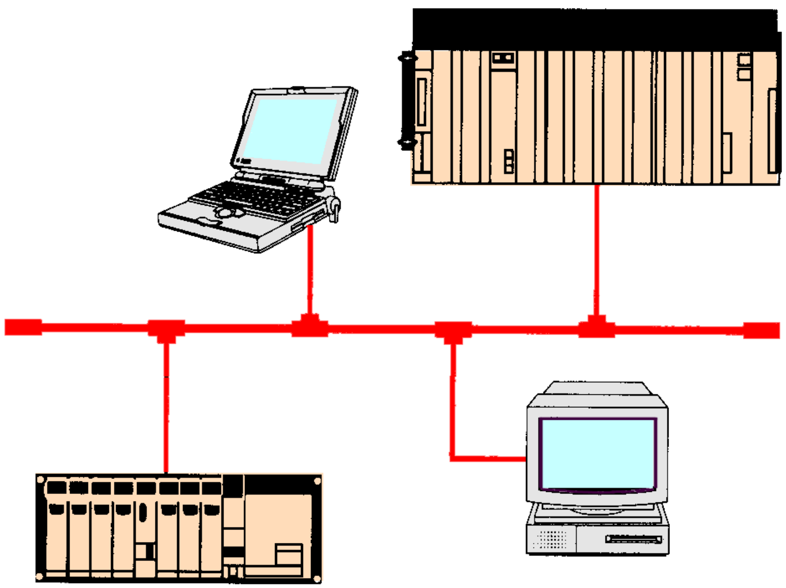
Originally PLCs were designed to replace relay control logic. The cost savings using PLCs have been so significant that relay control is becoming obsolete, except for power applications.



Generally, if an application requires more than about 6 control relays, it will usually be less expensive to install a PLC.

Advantages of a PLC Control System

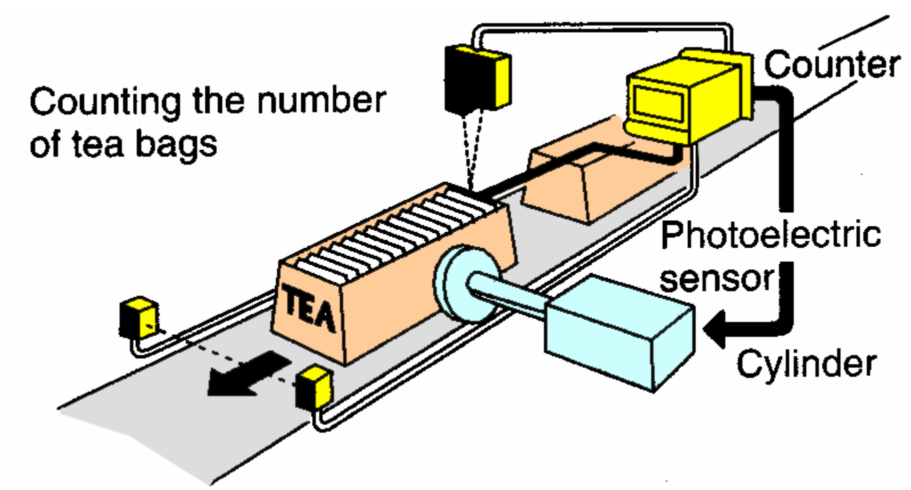
Communications Capability:
A PLC can communicate with other controllers or computer equipment.



They can be networked to perform such functions as: supervisory control, data gathering, monitoring devices and process parameters, and downloading and uploading of programs.

Advantages of a PLC Control System

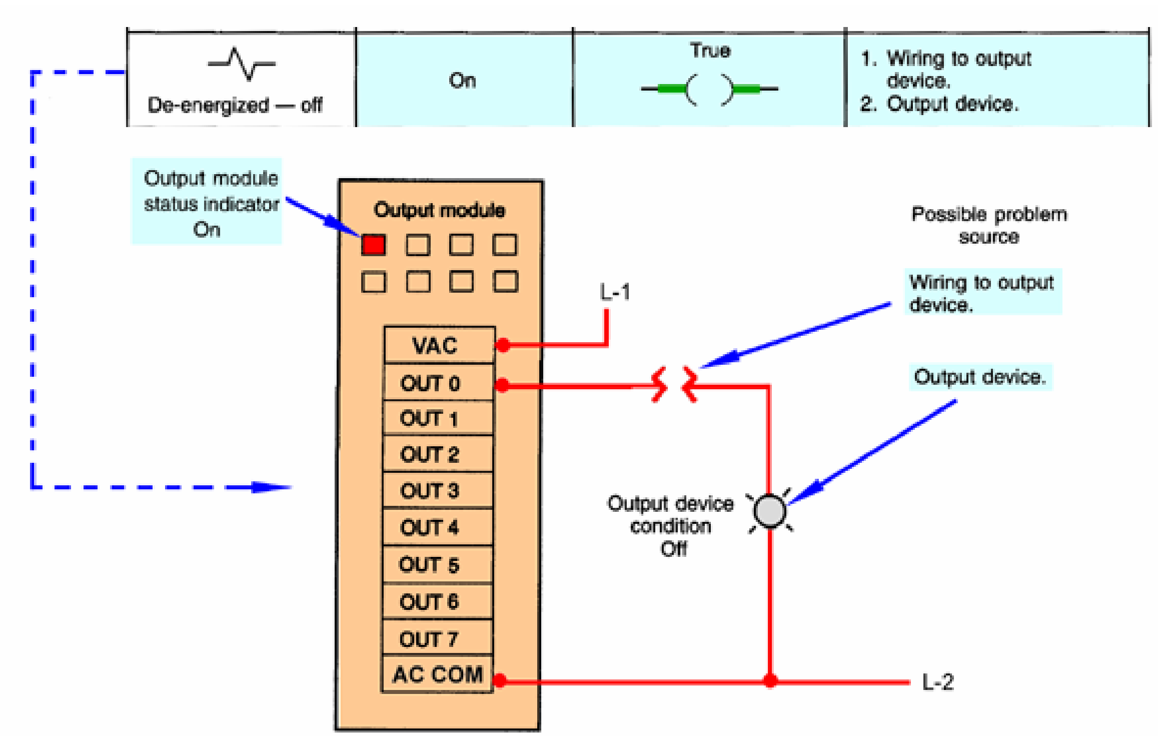
Faster Response Time:
PLCs operate in real-time which means that an event taking place in the field will result in an operation or output taking place.



Machines that process thousands of items per second and objects that spend only a fraction of a second in front of a sensor require the PLC's quick response capability.

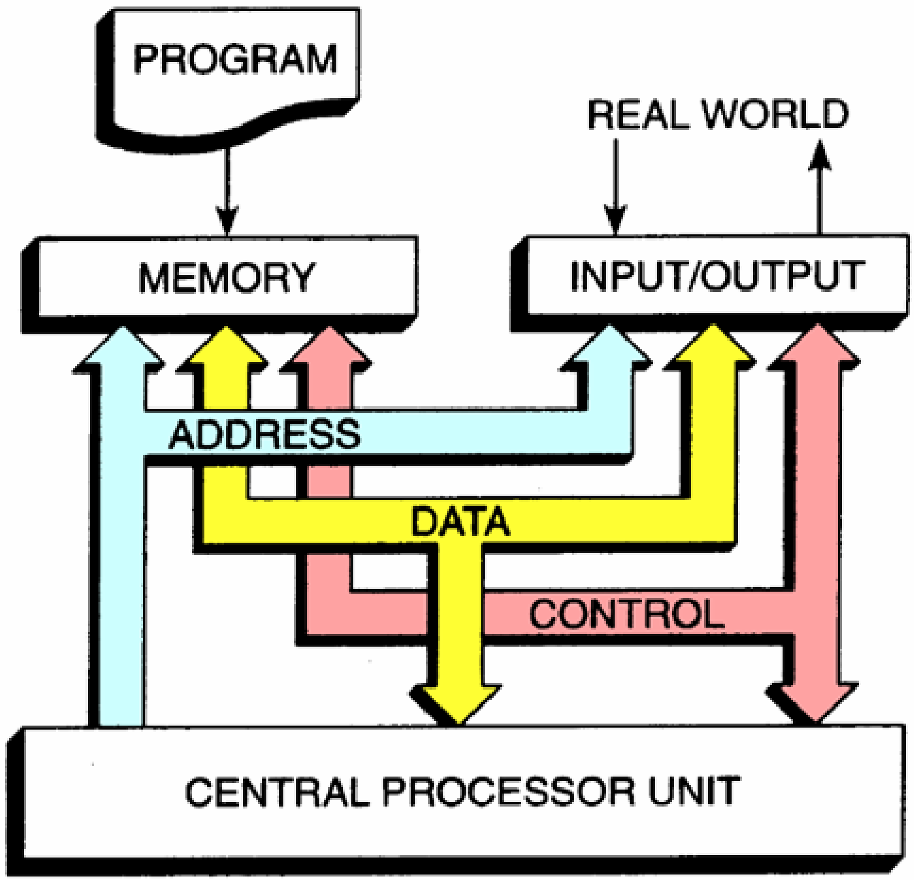
Advantages of a PLC Control System

Easier To Troubleshoot:
PLCs have resident diagnostic and override functions allowing users to easily trace and correct software and hardware problems.



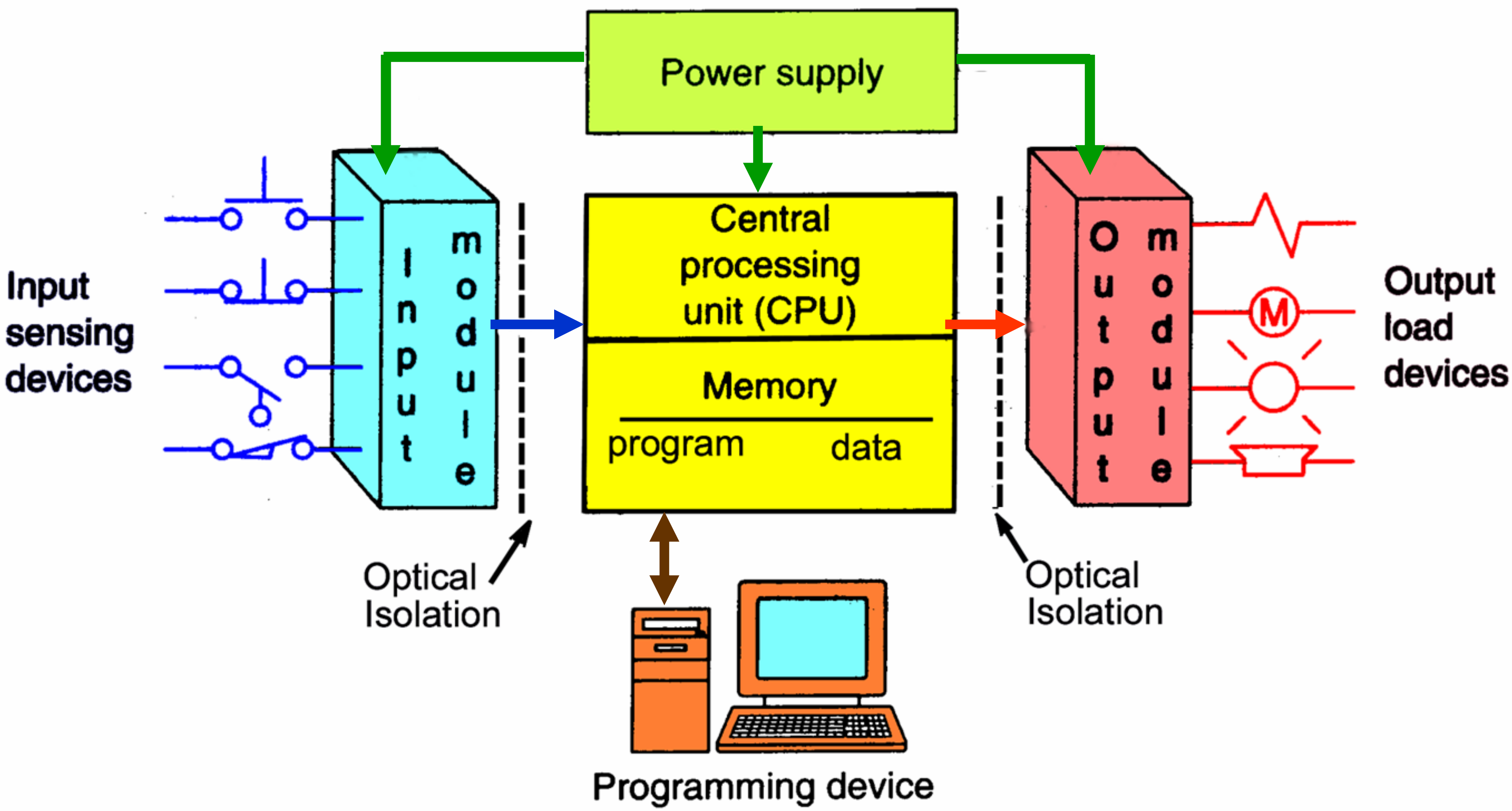
The control program can be watched in real-time as it executes to find and fix problems

PLC Architecture



The structure of a PLC is based on the same principles as those employed in computer architecture.

PLC System



Power supply
Input sensing devices
Input module
Central processing unit (CPU)
Memory
program
data
Output module
Output load devices
M
Optical Isolation
Optical Isolation
Programming device

PLC Architecture

- An *open architecture* design allows the system to be connected easily to devices and programs made by other manufacturers.

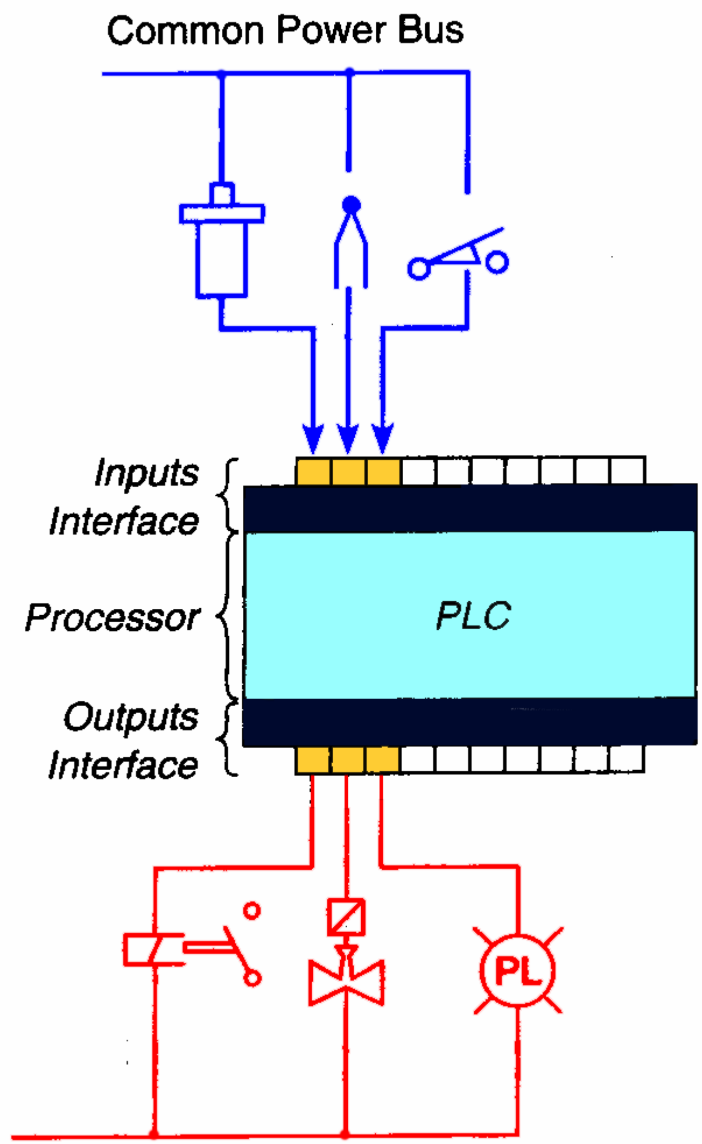
- A *closed architecture* or *proprietary* system, is one whose design makes it more difficult to connect devices and programs made by other manufacturers.

NOTE: When working with PLC systems that are proprietary in nature you must be sure that any generic hardware or software you use is compatible with your particular PLC.

I/O Configurations

Fixed I/O

- Is typical of small PLCs
- Comes in one package, with no separate removable units.
- The processor and I/O are packaged together.
- Lower in cost – but lacks flexibility.

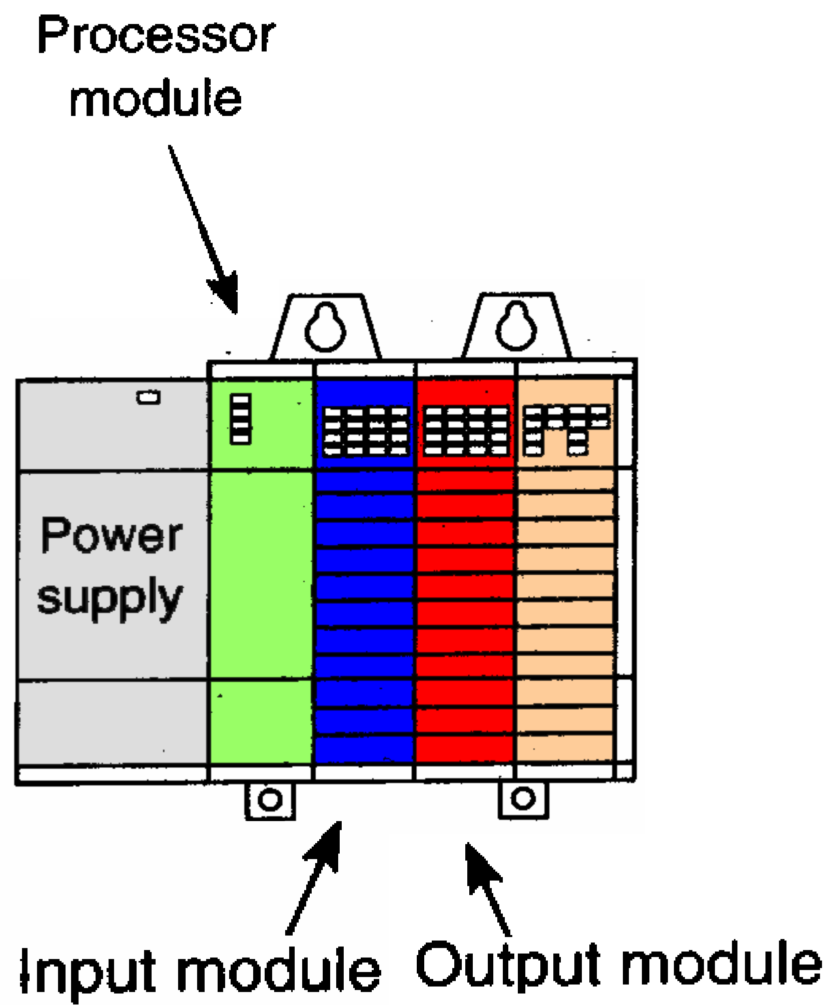


I/O Configurations

Modular I/O

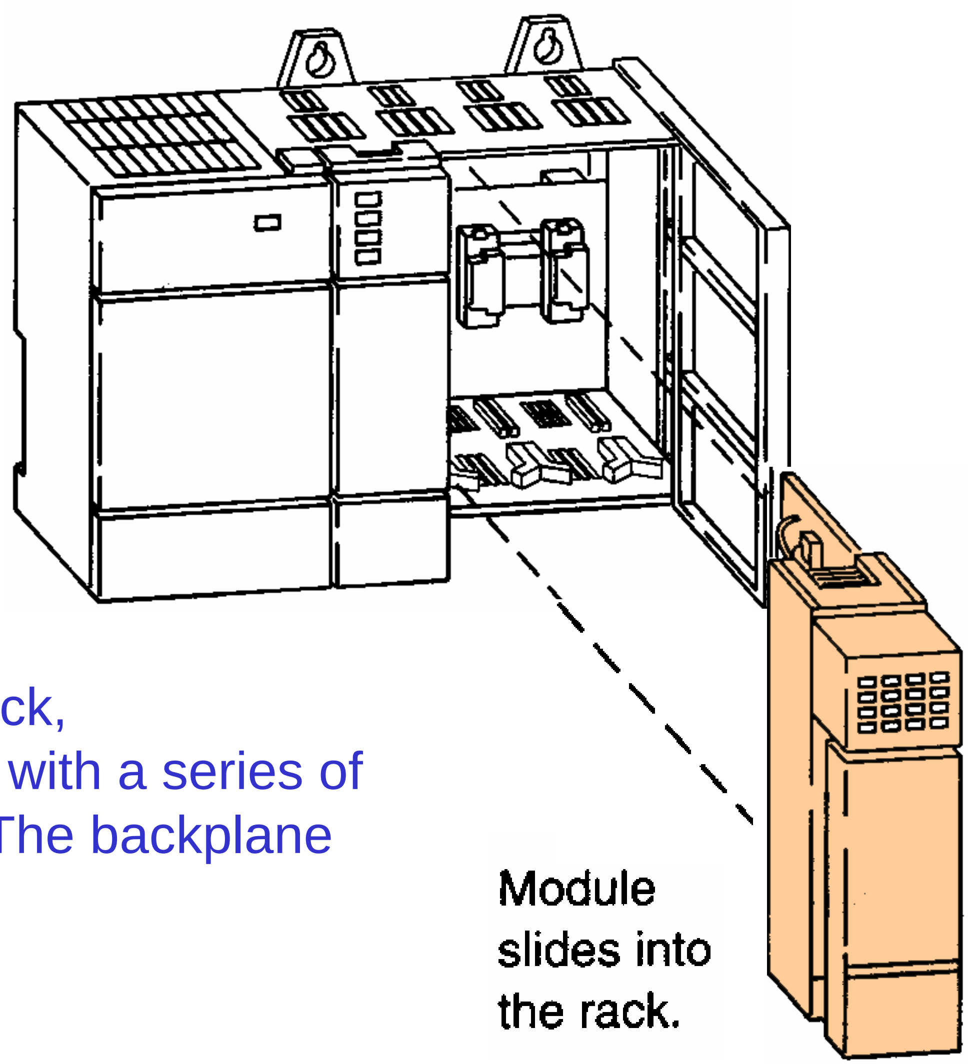
Is divided by compartments into which separate modules can be plugged.

This feature greatly increases your options and the unit's flexibility. You can choose from all the modules available and mix them in any way you desire.



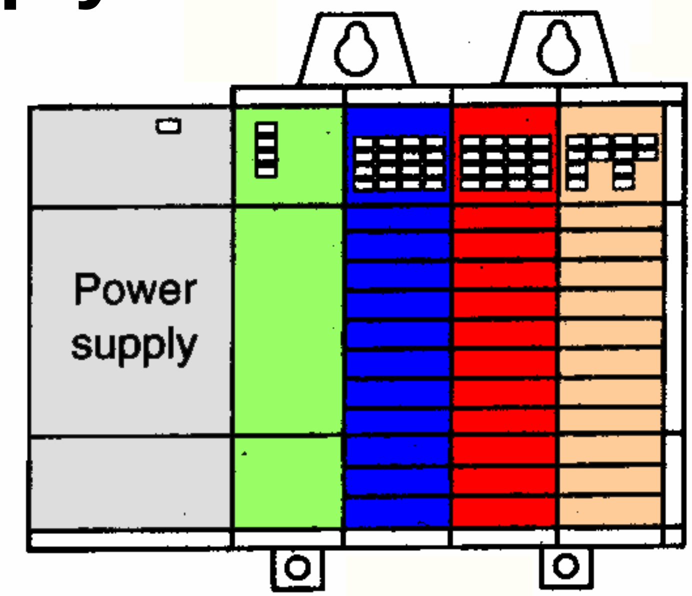
I/O Configurations

Modular I/O



When a module slides into the rack,
it makes an electrical connection with a series of contacts - called the *backplane.* The backplane is located at the rear of the rack.

Power Supply



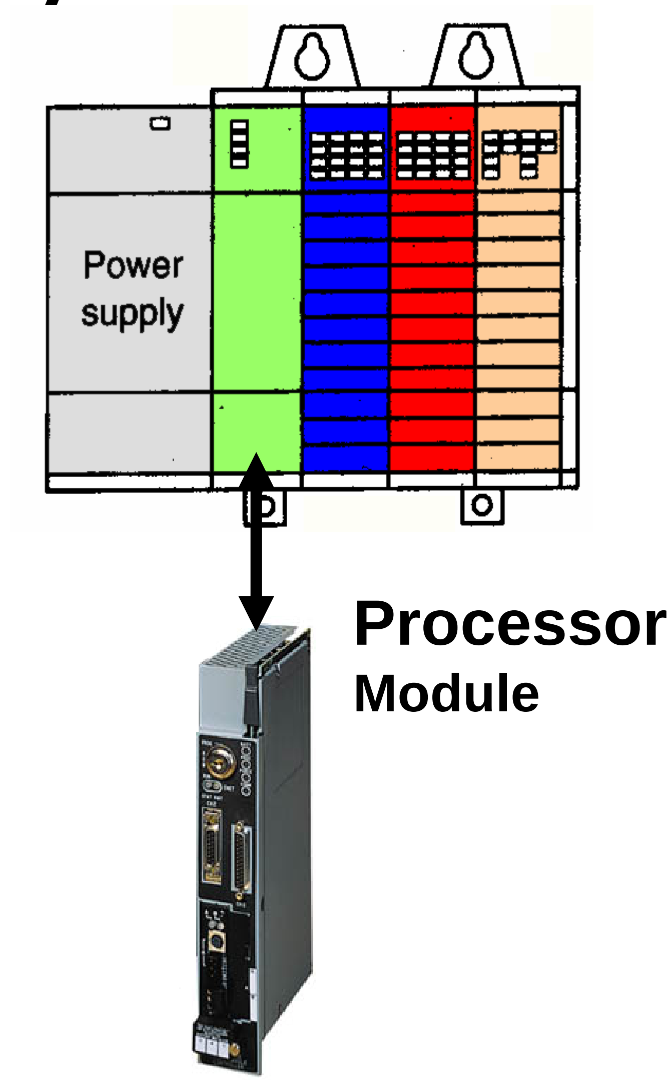
- Supplies DC power to other modules that plug into the rack.

- In large PLC systems, this power supply does not normally supply power to the field devices.

- In small and micro PLC systems, the power supply is also used to power field devices.

Processor (CPU)

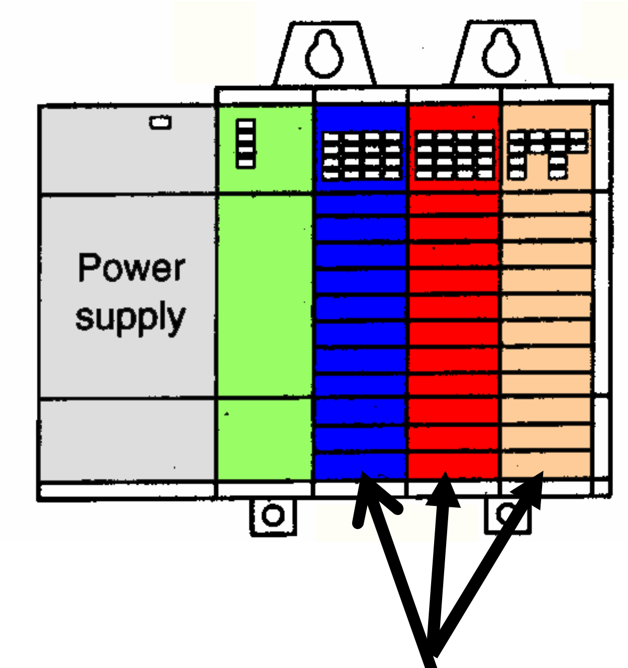
- Is the "brain" of the PLC.
- Consists of a microprocessor for implementing the logic, and controlling the communications among the modules.
- Designed so the desired circuit can be entered in relay ladder logic form.
- The processor accepts input data from various sensing devices, executes the stored user program, and sends appropriate output commands to control devices.

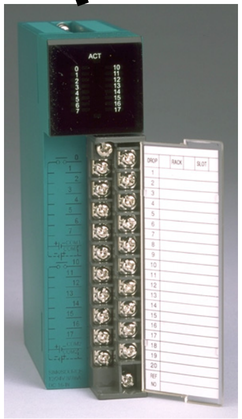


I/O Section

Consists of:

- Input modules
- Output modules.



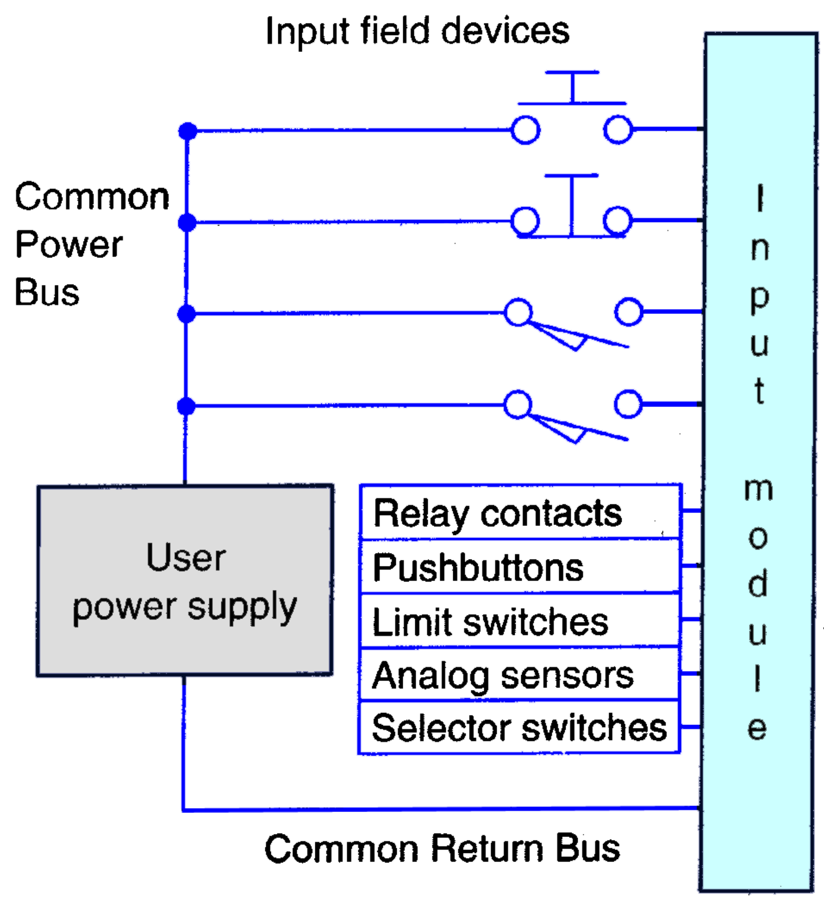


I/O Section

Input Module

- Forms the interface by which input field devices are connected to the controller.

- The terms “field” and “real world”are used to distinguish actual external devices that exist and must be physically wired into the system.

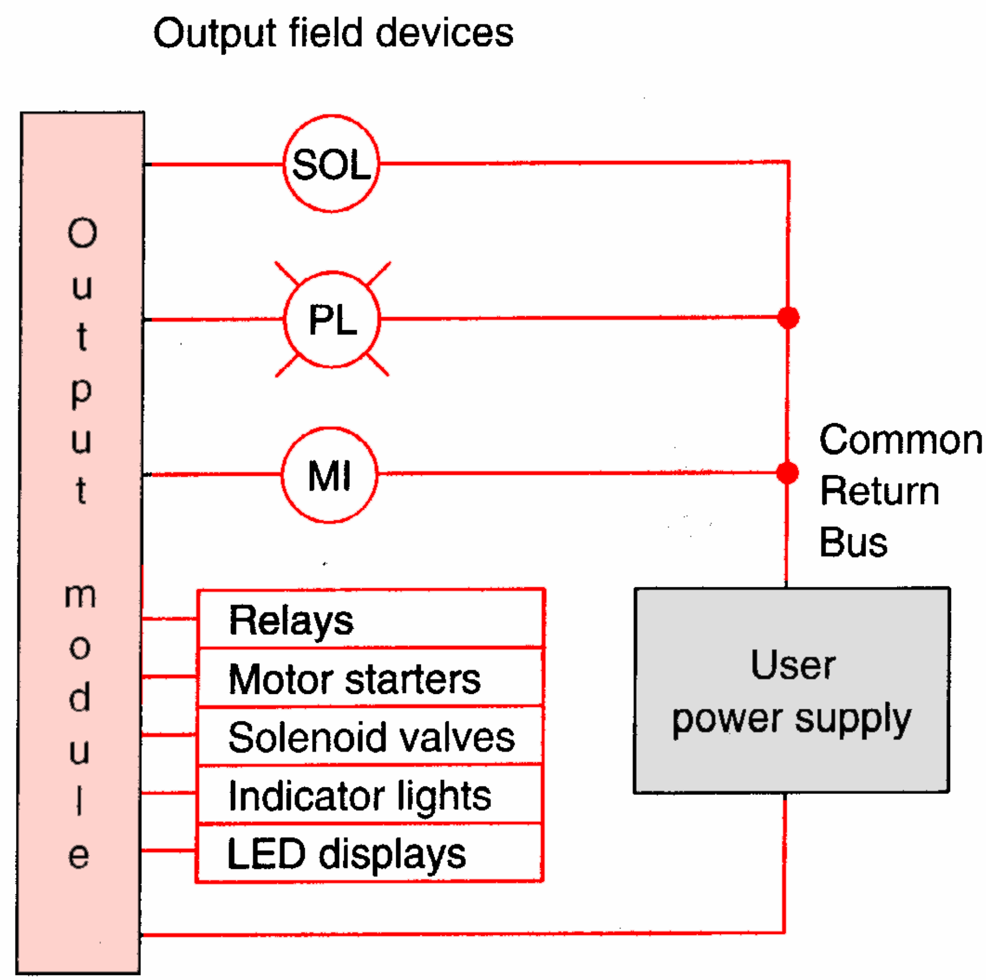


I/O Section

Output Module

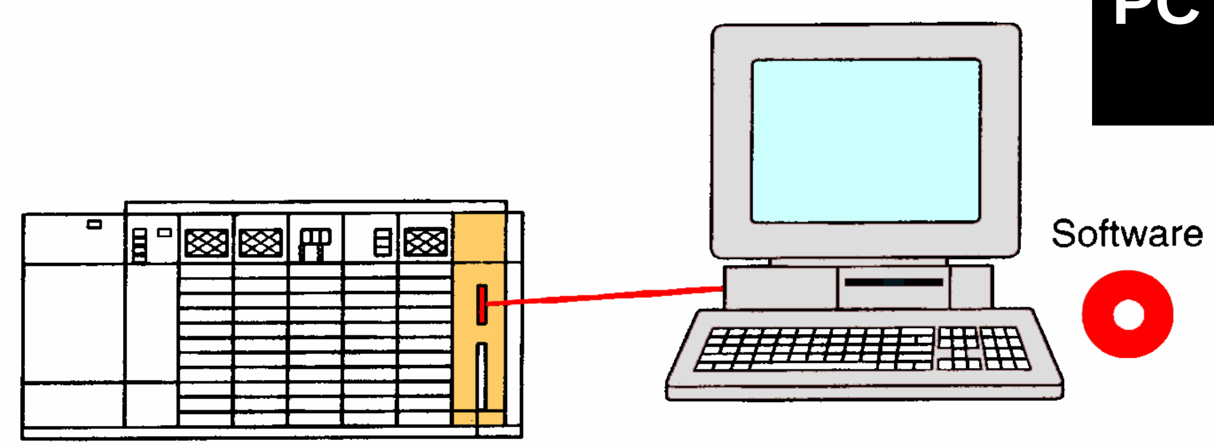
- Forms the interface by which output field devices are connected to the controller.

- PLCs employ an optical isolator which uses light to electrical-ly isolate the internal components from the input and output terminals.



Programming Device

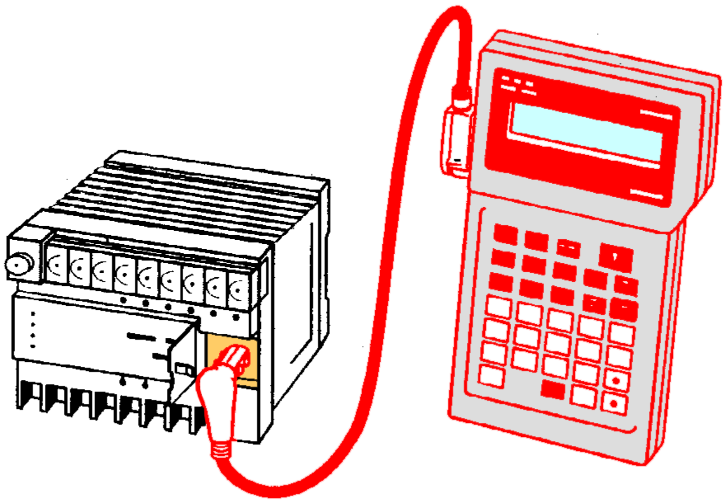
PC with appropriate software



- A personal computer (PC) is the most commonly used programming device
- The software allows users to create, edit, document, store and troubleshoot programs
- The personal computer communicates with the PLC processor via a serial or parallel data communications link

Programming Device

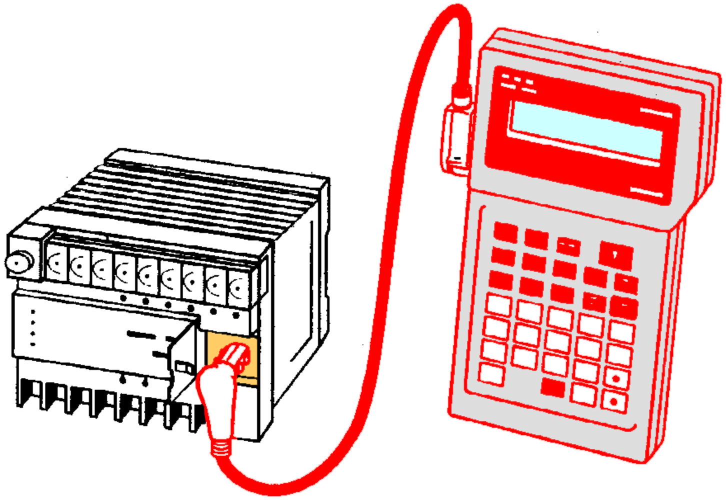
Hand-held unit with display



- Hand-held programming devices are sometimes used to program small PLCs
- They are compact, inexpensive, and easy to use, but are not able to display as much logic on screen as a computer monitor

Programming Device

Hand-held unit with display

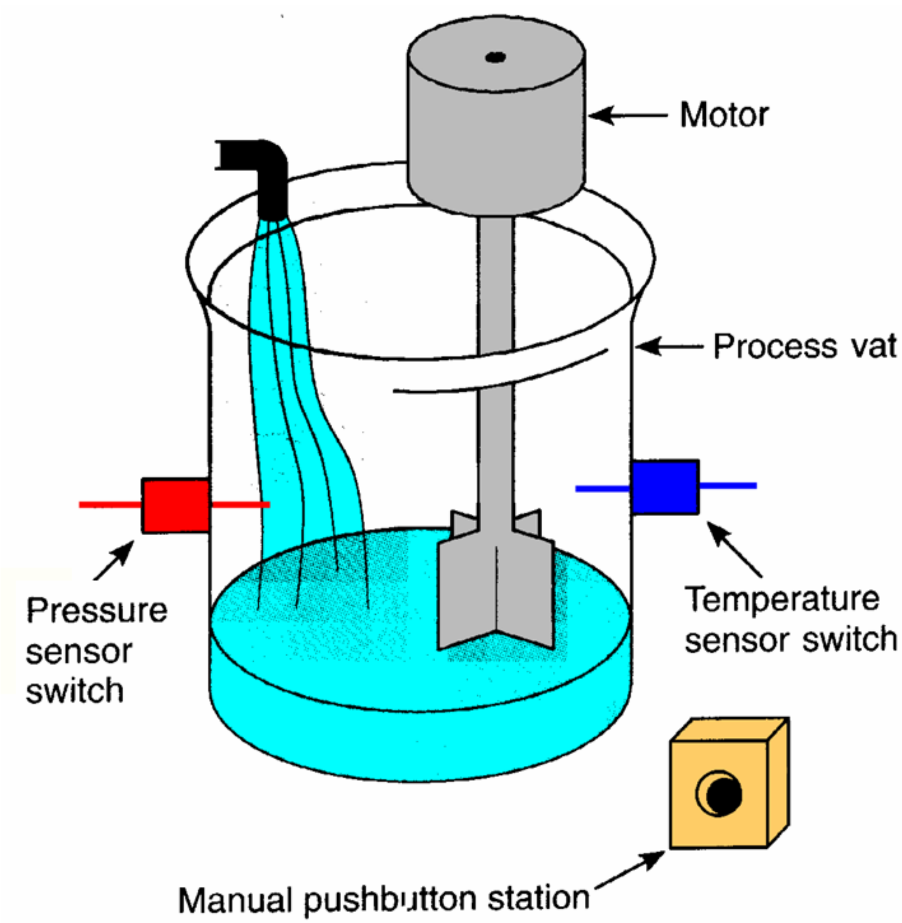


- Hand-held units are often used on the factory floor for troubleshooting, modifying programs, and transferring programs to multiple machines.

PLC Mixer Process Control Problem

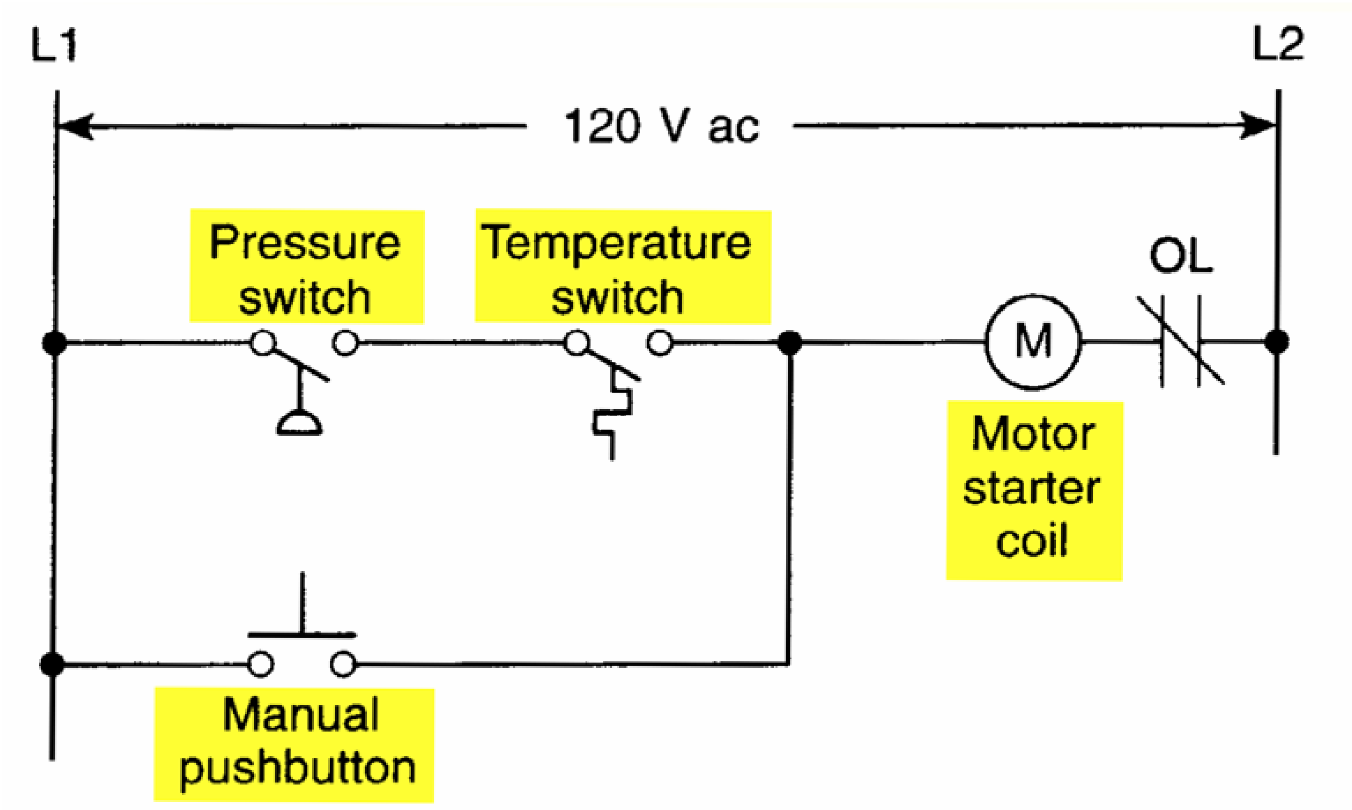
Mixer motor to automatically stir the liquid in the vat when the temperature and pressure reach preset values

Alternate manual pushbutton control of the motor to be provided



The temperature and pressure sensor switches close their respective contacts when conditions reach their preset values

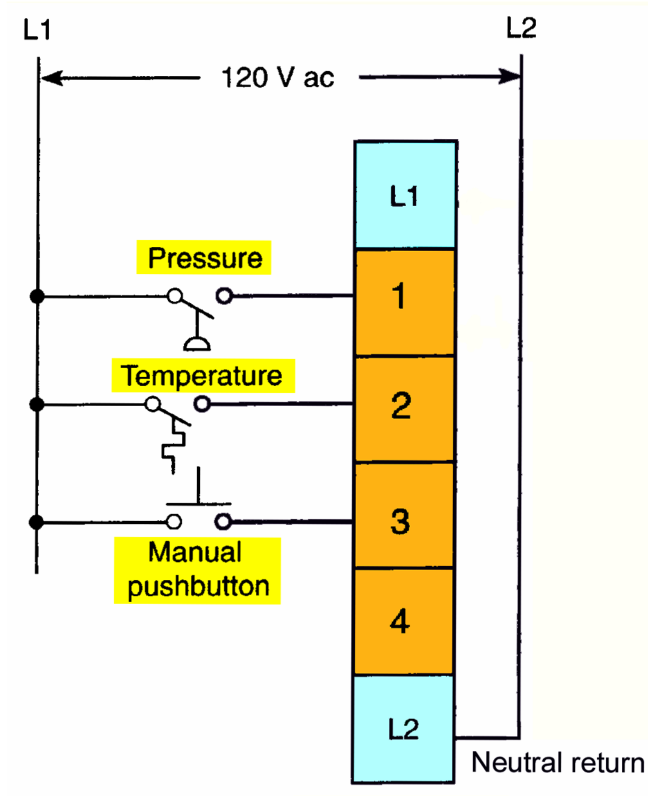
Process Control Relay Ladder Diagram



Motor starter coil is energized when both the pressure and temperature switches are closed or when the manual pushbutton is pressed

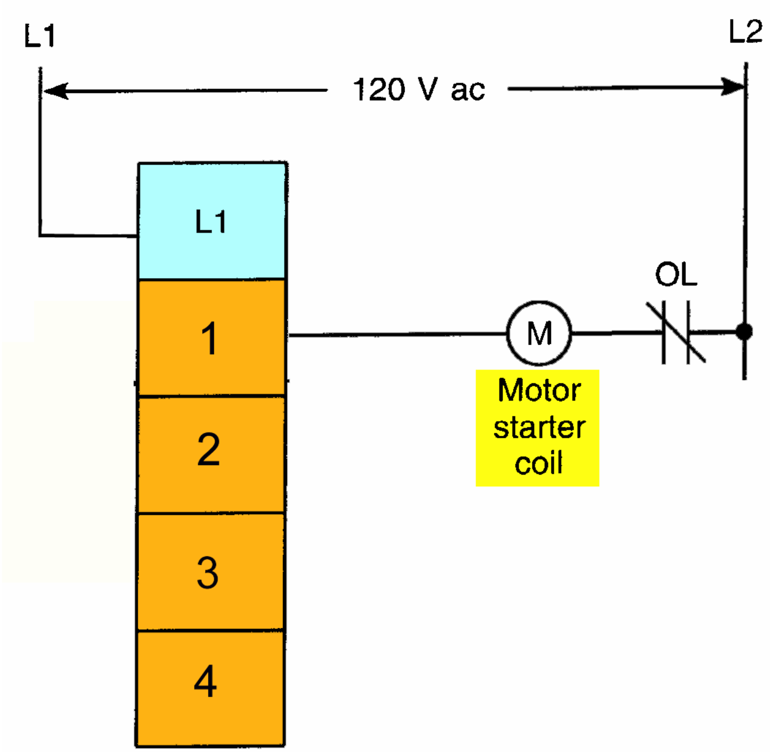
PLC Input Module Connections

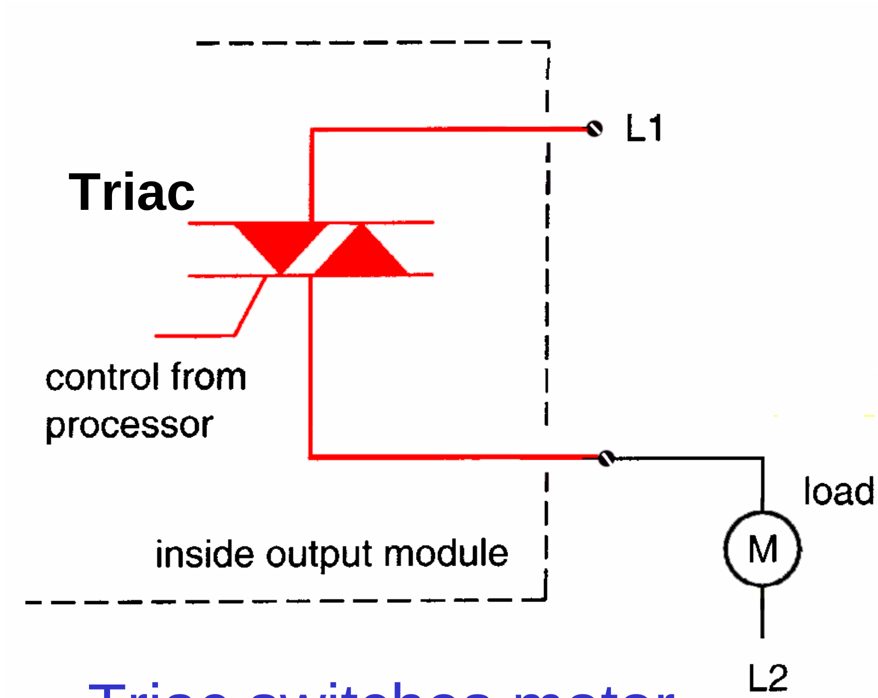
- The same input field devices are used
- These devices are wired to the input module according to the manufacturer's labeling scheme



PLC Output Module Connections

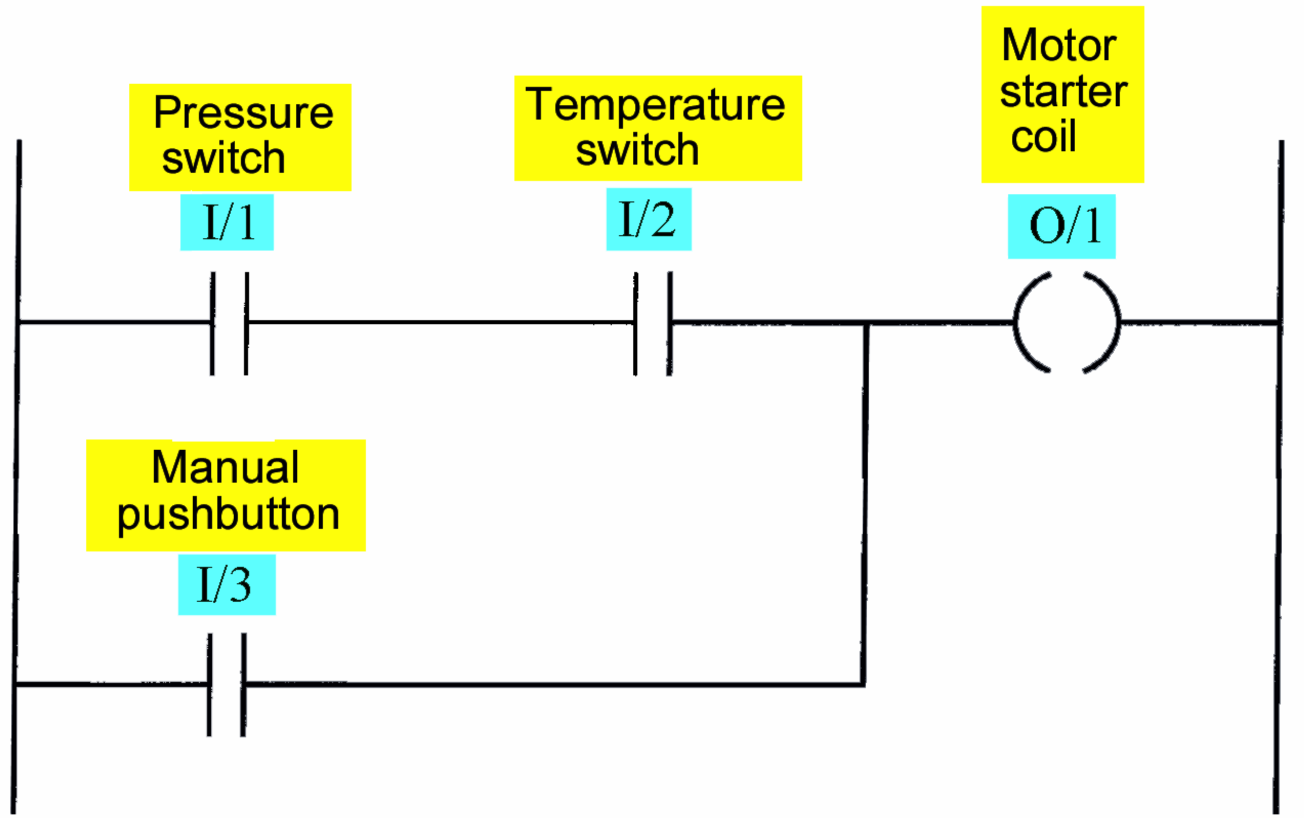
Same output field device is used and wired to the output module





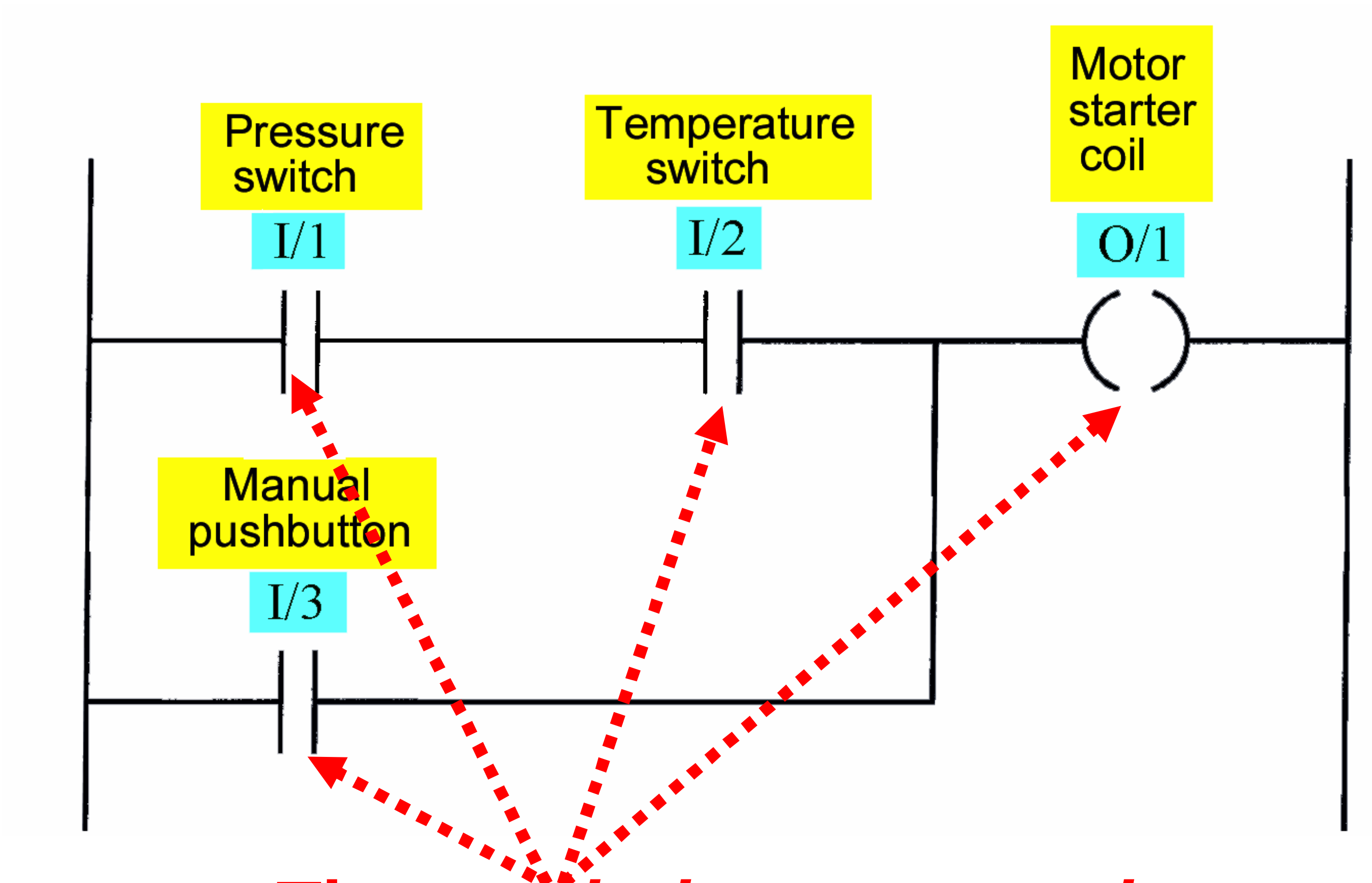
Triac switches motor ON and OFF in accordance with the control signal from the processor

PLC Ladder Logic Program



- The format used is similar to that of the hard-wired relay circuit

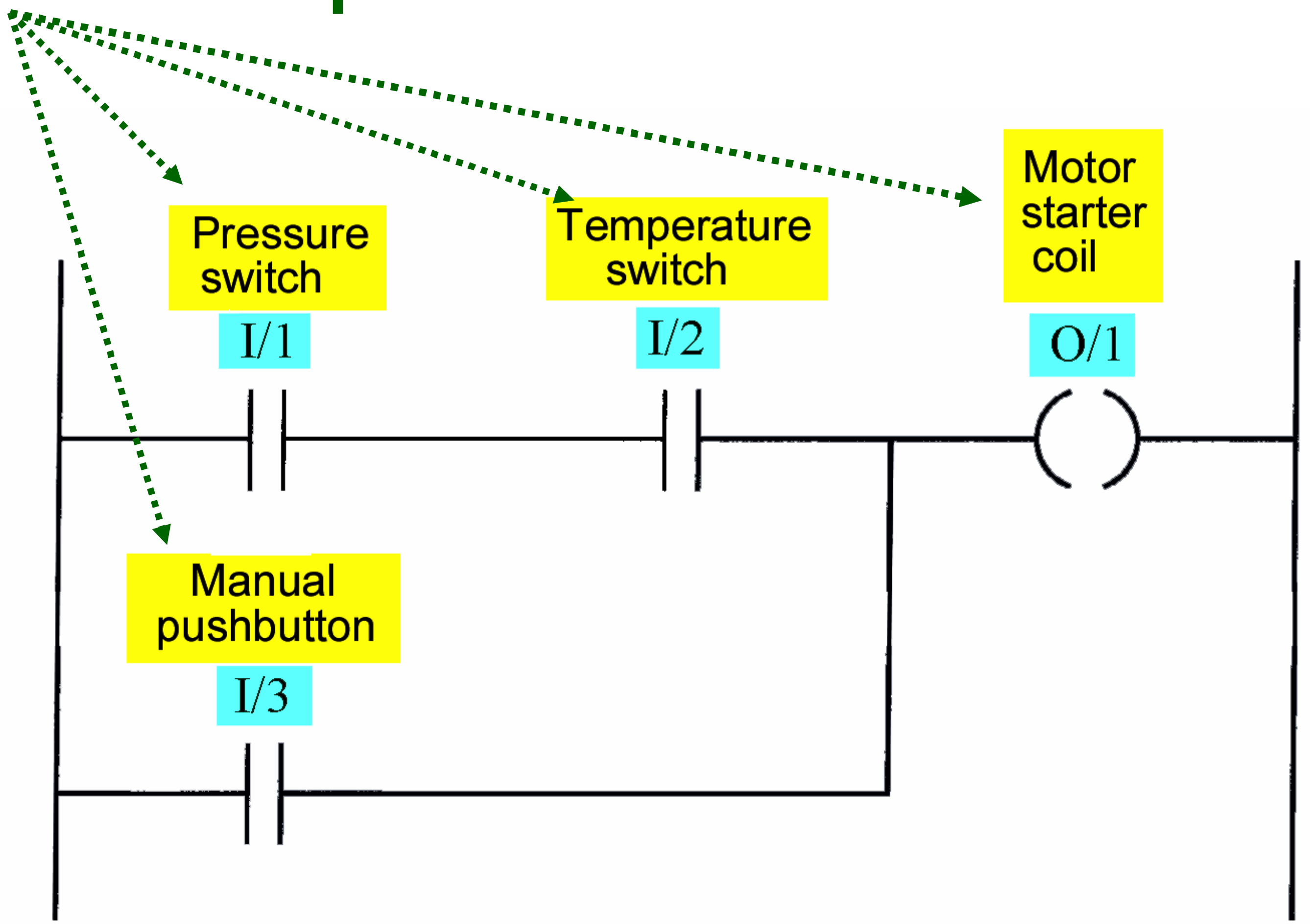
PLC Ladder Logic Program



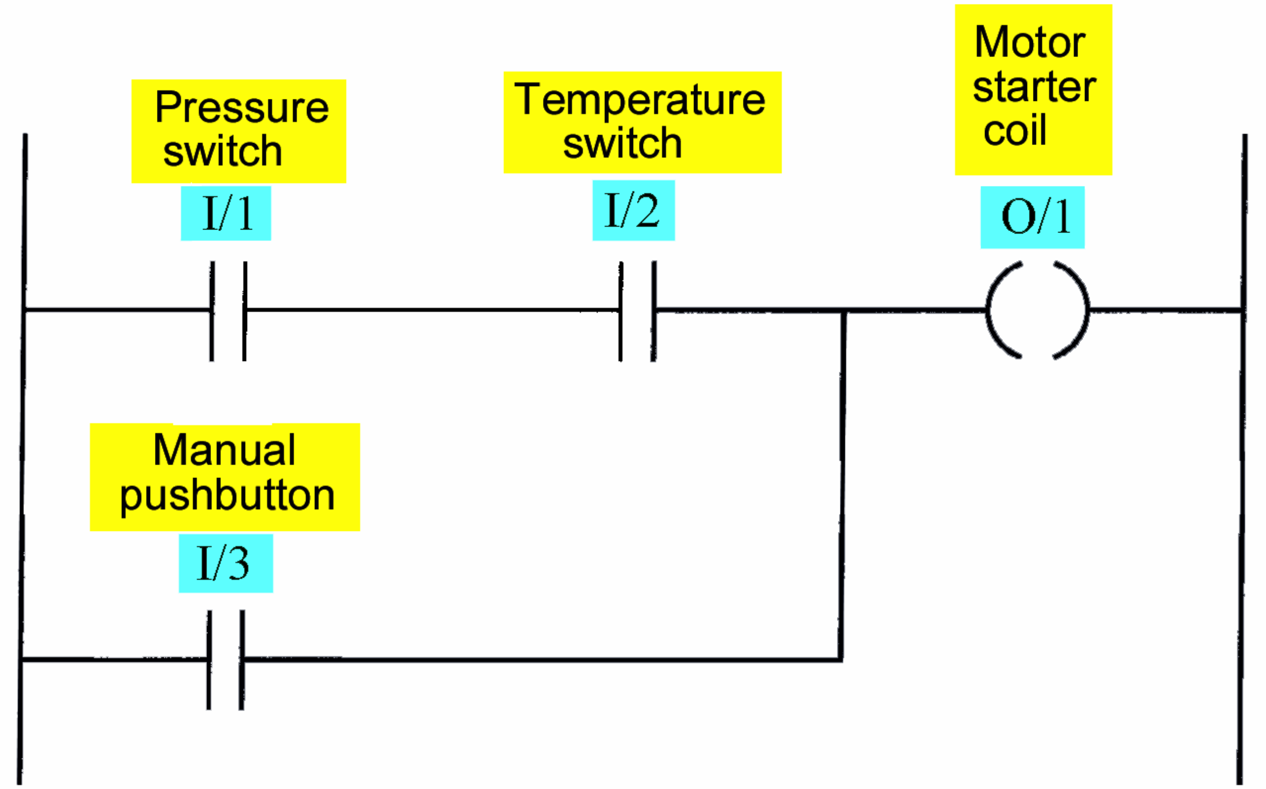
The *symbols* represent *instructions*

PLC Ladder Logic Program

The *numbers* represent *addresses*

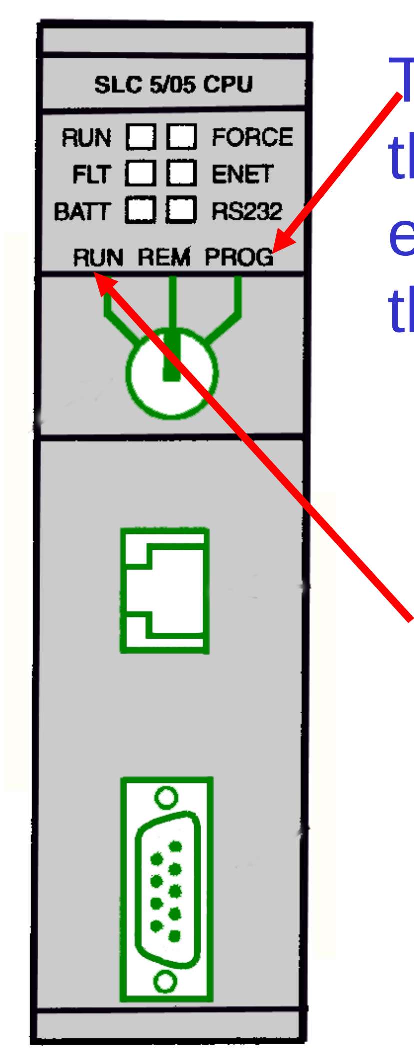


PLC Ladder Logic Program



- I/O address format will differ, depending on the PLC manufacturer. You give each input and output device an address. This lets the PLC know where they are physically connected

Entering And Running The PLC Program

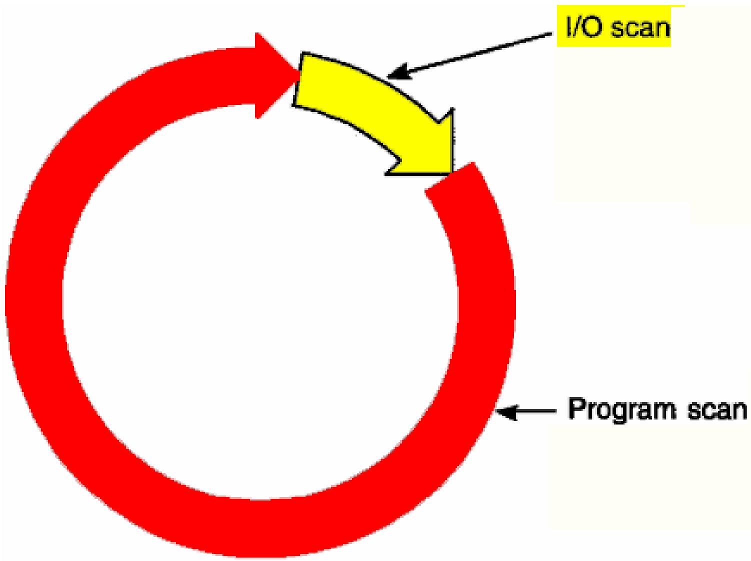


To enter the program into the PLC, place the processor in the PROGRAM mode and enter the instructions one-by-one using the programming device

To operate the program, the controller is placed in the RUN mode, or operating cycle

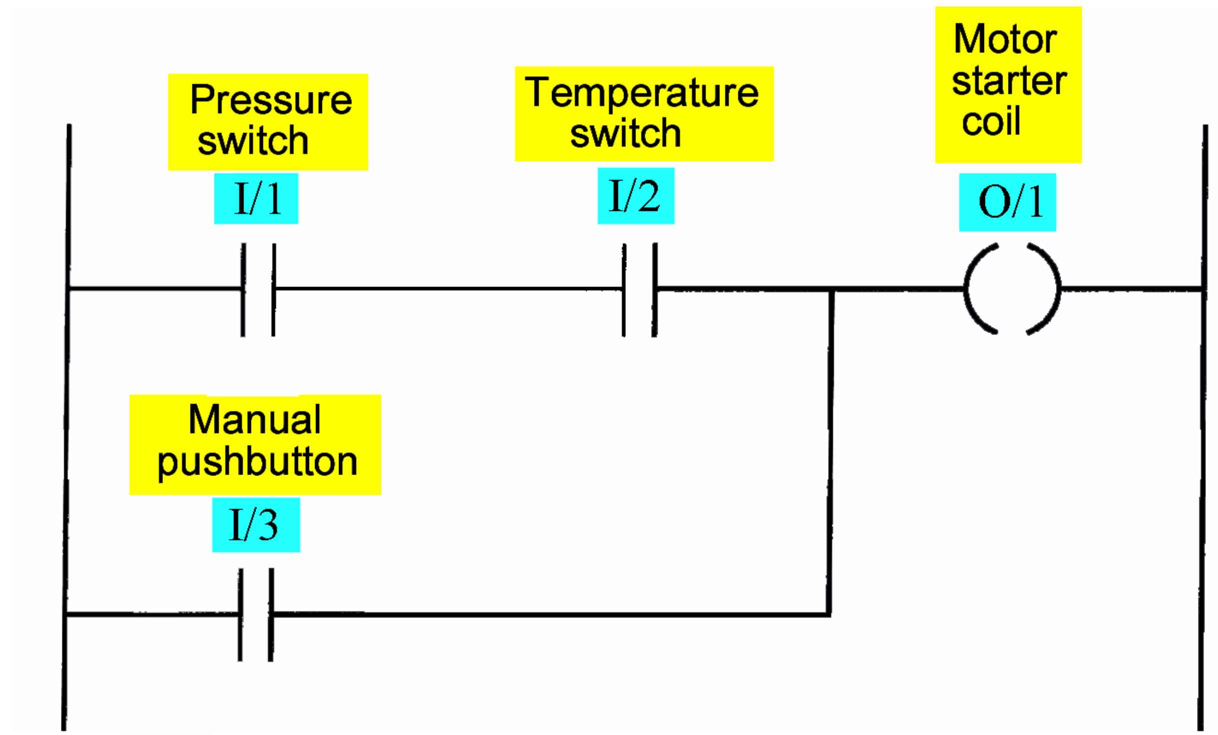
PLC Operating Cycle

During each operating cycle, the controller examines the status of input devices, executes the user program, and changes outputs accordingly



The completion of one cycle of this sequence is called a *scan*. The scan time, the time required for one full cycle, provides a measure of the speed of response of the PLC

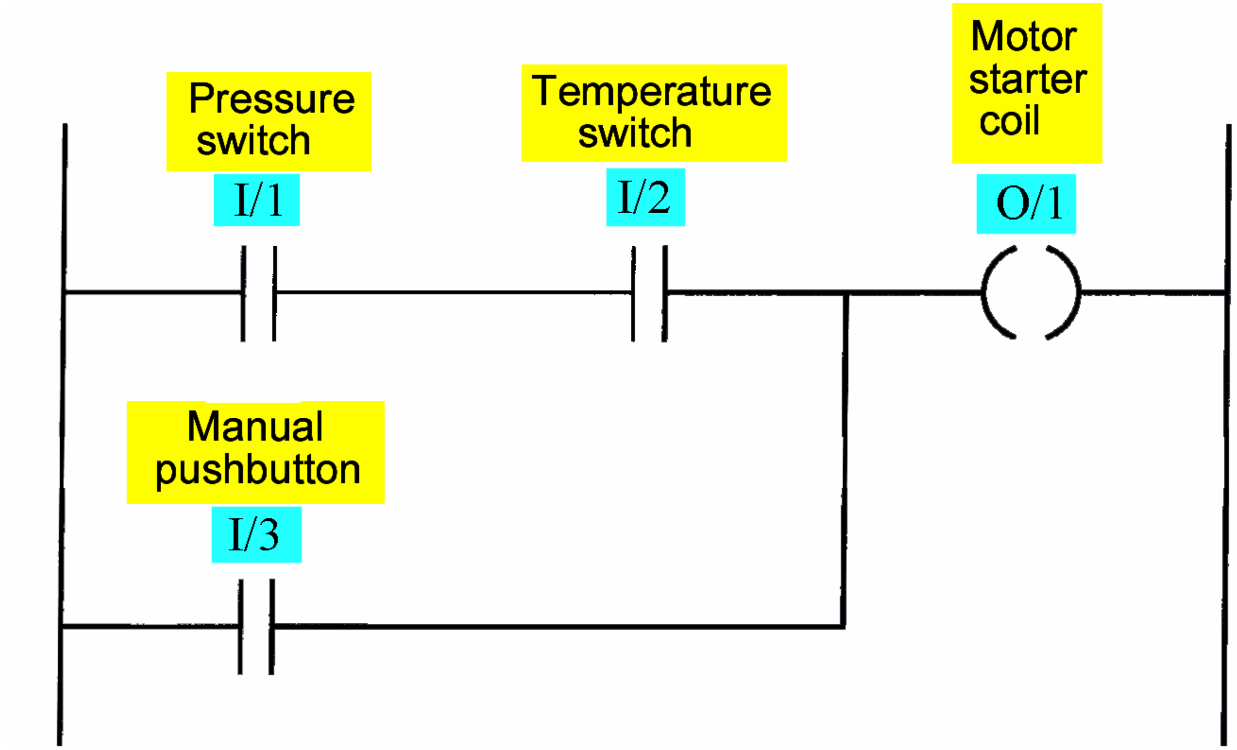
PLC Operating Cycle



Each  can be though of as a set of normally open contacts

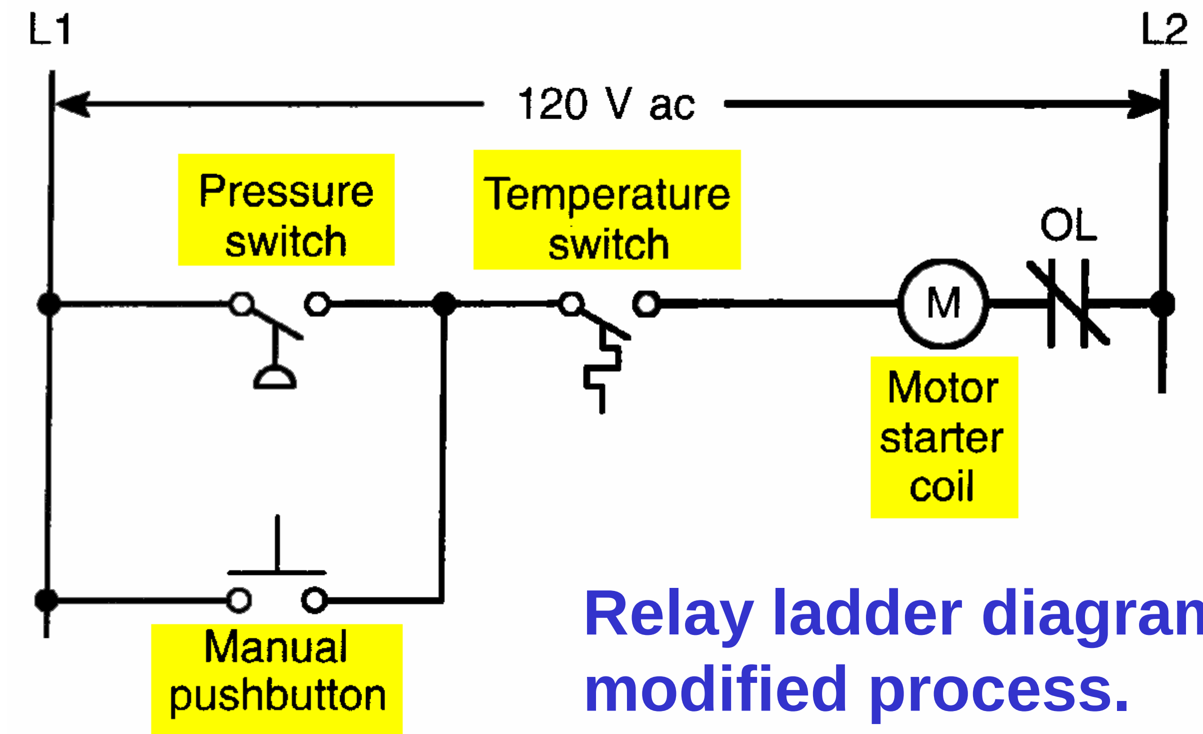
The  can be considered to represent a coil that, when energized, will close a set of contacts.

PLC Operating Cycle



Coil O/1 is energized when contacts I/1 and I/2 are closed or when contact I/3 is closed. Either of these conditions provides a continuous path from left to right across the rung that includes the coil.

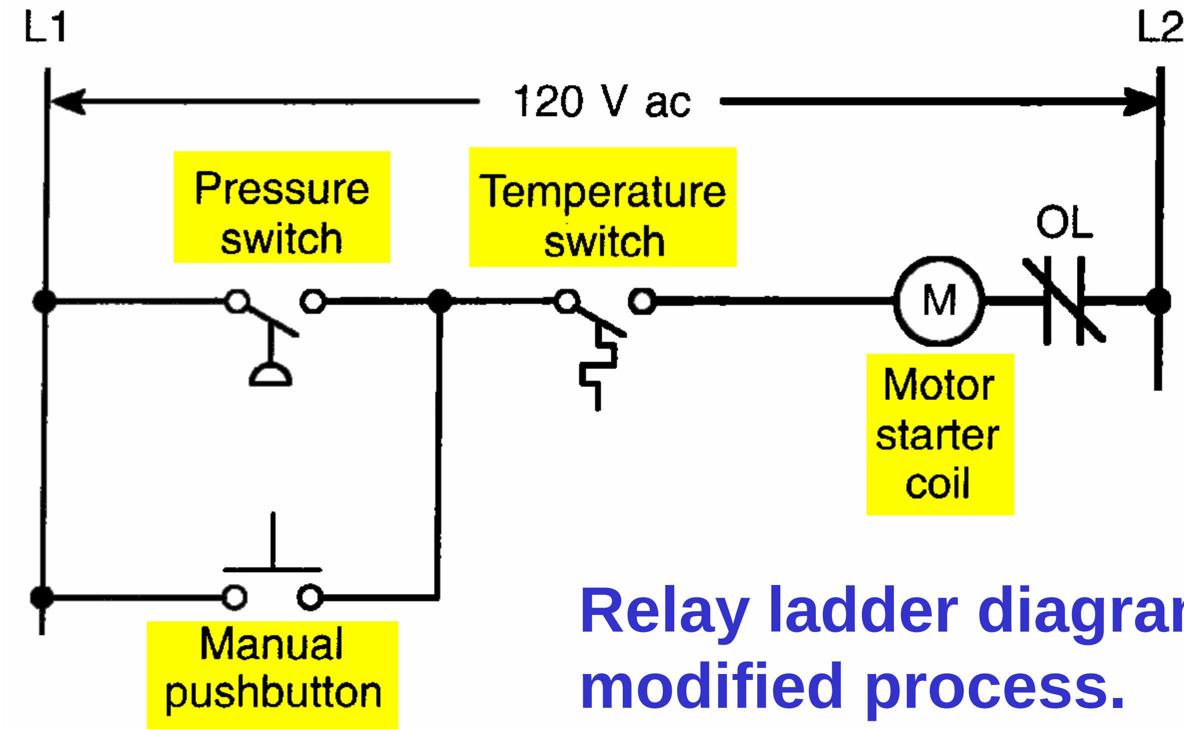
Modifying A PLC Program



Relay ladder diagram for modified process.

The change requires that the manual pushbutton control should be permitted to operate at any pressure but not unless the specified temperature setting has been reached.

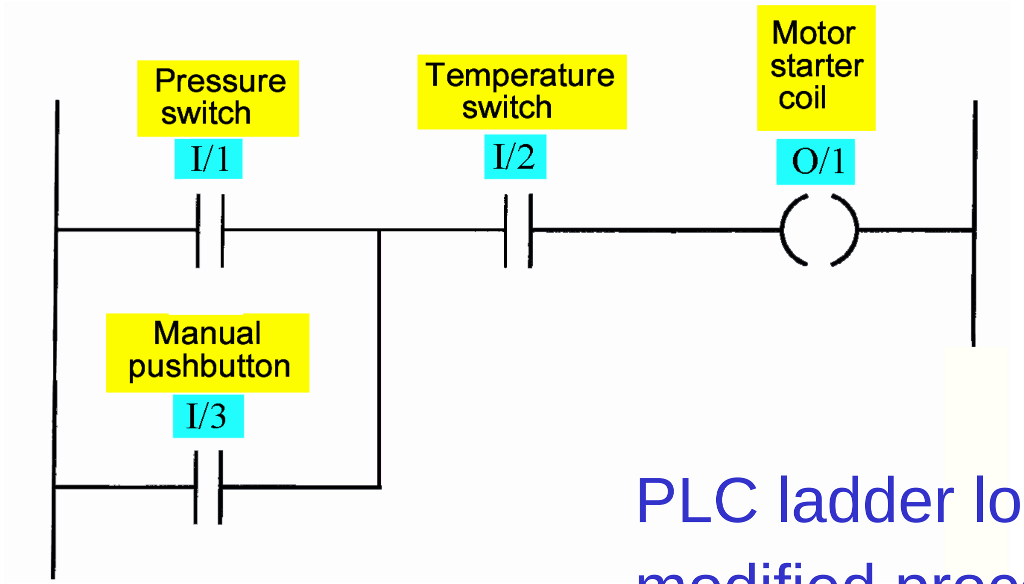
Modifying A PLC Program



Relay ladder diagram for modified process.

If a relay system were used, it would require some rewiring of the system, as shown, to achieve the desired change.

Modifying A PLC Program



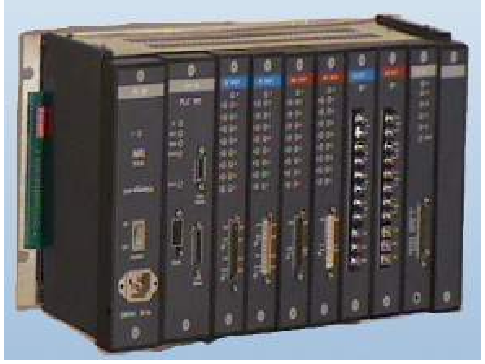
PLC ladder logic diagram for modified process.

If a PLC is used, no rewiring is necessary!

The inputs and outputs are still the same.

All that is required is to change the PLC program

PLCs Versus Personal Computers



Same basic architecture

PLC

- Operates in the industrial environment
- Is programmed in relay ladder logic
- Has no keyboard, CD drive, monitor, or disk drive
- Has communications ports, and terminals for input and output devices

PC

- Capable of executing several programs simultaneously, in any order
- Some manufacturers have software and interface cards available so that a PC can do the work of a PLC

PC Based Control Systems

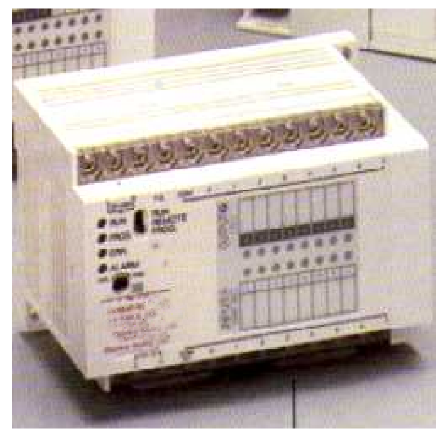
Advantages

- Lower initial cost
- Less proprietary hardware and software required
- Straightforward data exchange with other systems
- Speedy information processing
- Easy customization

PLC Size Classification

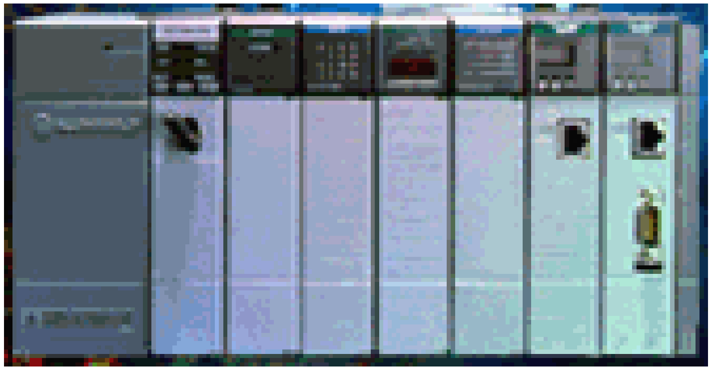
Criteria
- Number of inputs and outputs (I/O count)
- Cost
- Physical size

Nano PLC
- Smallest sized PLC
- Handles up to 16 I/O points



Micro PLC
- Handles up to 32 I/O points

PLC Size Classification

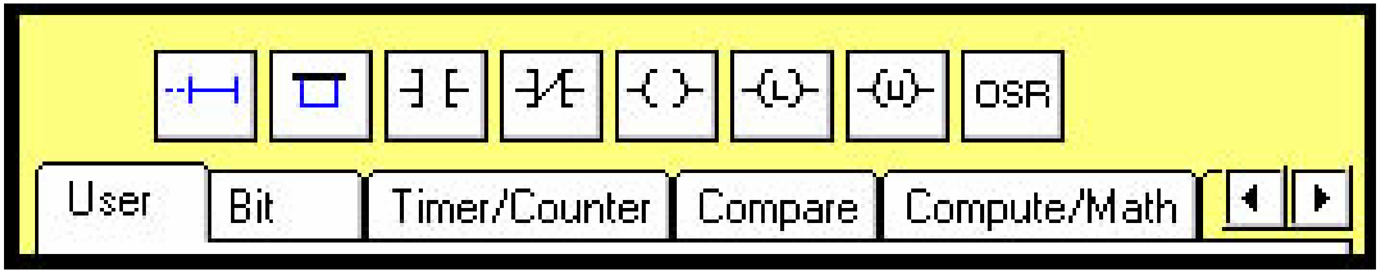


Allen-Bradley SLC-500 Family
- Handles up to 960 I/O points

Allen-Bradley PLC-5 Family
- Handles several thousand I/O points

PLC Instruction Set

The instruction set for a particular PLC type lists the different types of instructions supported.



An instruction is a command that will cause a PLC to perform a certain predetermined operation.

Typical PLC Instructions

XIC (Examine ON) . . . . Examine a bit for an ON condition
XIO (Examine OFF) . . . . . . Examine a bit for an OFF condition
OTE (Output Energize) . . . . Turn ON a bit (non retentive)
OTL (Output Latch) . . . . Latch a bit (retentive)
OTU (Output Unlatch) . . . . Unlatch a bit (retentive)
TOF (Timer Off-Delay) . . . . . Turn an output ON or OFF after its rung has been OFF a preset time interval
TON (Timer On-Delay) . . . . Turn an output ON or OFF after its rung has been ON for a preset time interval
CTD (Count Down) . . . . Use a software counter to count down from a specified value
CTU (Count Up) . . . . Use a software counter to count up to a specified value

Chapter 1 INTRODUCTION

Introduction to PLCs

Programmable logic controllers (PLCs) were introduced to industry between 1968 and 1970 as a way to replace large expensive panels of relays, timers, and counters. Automotive manufacturers were looking for ways to simplify start-up of new car lines after model changeovers each year and save money in the cost of manufacturing installations.

Historically, relays have been used since the late 1800's to control simple processes. They were used in the early days to control railroad crossings. Before simple relay logic was introduced to control railroad crossing arms and alarm lights, accidents at these crossings were contributing to a high toll on human life.

The term “relay” was coined as the name of the device invented by Samuel F. B. Morse who invented the telegraph. The relay was invented as a device to extend the signal or “relay” the signal of the telegraph more than the 20 mile limit of electrical signals at the time of the invention of the telegraph (1836).

Relays, timers, and counters had been the favored choice for electrical and systems engineers to manufacturing facilities, especially in facilities with a large number of machines making discrete parts. Automotive manufacturers top the list of this type of manufacturing. At the same time that costs continued to rise for the engineering and construction of automotive assembly lines, computers were becoming more numerous and less costly. There was, however, a general discomfort among engineers to replace relays with computers. Most were reluctant to place the computer on the plant floor. A compromise was necessary for the engineer that he would be willing to accept. A computer that appeared to be relay-ladder logic to the electrician but able to use the computing capabilities of a computer was the device envisioned. The result of this vision is what is known today as the PLC (Programmable Logic Controller).

Relays as well as timers and counters were the first devices replaced by the PLC. Relays are electromechanical devices that use magnetism caused by power flow through the circuit's coil to energize a core and move a plunger with contacts attached. Contacts change state when the coil is magnetized. Normally open contacts close while normally closed contacts open. Changing contacts combine to complete other circuits. Combinations of relay contacts energizing coils form the basis of Boolean logic. Boolean logic deals with the combination of discrete on-off states to turn on or off other outputs.

The principle of using PLCs as substitutes for relays to reduce the wiring, panel fabrication, and engineering cost looked very appealing to the early PLC user. Like most electronic devices appearing in the early '70's, cost of the early PLC was high and functionality was not well developed. Early PLCs were developed around a mini-computer or special purpose control board. It was not unusual to pay $50,000 or more for a single PLC complete with I/O and still use relays for the most critical circuits. These hybrid designs existed for many years until engineers became convinced that PLCs were as reliable as the relay and could be trusted equally to the relay circuit being replaced. Hybrid circuits using both relays and PLCs existed for many years but have become too expensive for most automation applications in today's market.

In a quote from *Programmable Logic Controllers: An Emphasis on Design and Application* (Erickson, 2005) about the early PLC:

"The machine tool and automotive industries were large users of relay control systems. A simple machine tool would require six months to a year to completely debug (Morley, 2001). Every year, automotive manufacturing facilities would be shut down for two to three months in order to implement the changes due to the new automotive models. The lost production due to these changes was significant."

Richard (Dick) Morley was credited with the invention of the modern PLC. Morley worked for Bedford Associates in Massachusetts in 1968 when the PLC was first conceived. Dick Morley later wrote a book, *History of the PLC* in 2001 published by R. Morley Incorporated. Morley is commonly considered the father of the PLC.

Another quote from Morley in addressing the toughness of the PLC supplied by Bedford Associates, the Modicon controller from Morley's text and found also in Erickson's text:

"Landis [a machine tool company] decided to purchase the MODICON units and not use the PDP-14. When DIGITAL [*sic*] tried it get back into Landis, Landis wrapped a welder cable (operating) around the 084 and poured Coke over the unit. The 084 kept right on trucking. Digital retreated with grace."

Other anecdotal competitive stories abound and make up part of the folk lore of the early PLC. One about a PLC using 15 volt logic shows the competition between early PLC vendors. The vender owning the 15-volt PLC inserted a spark plug in a waveguide and entered a vendor's show with the noise gun in hand (carefully concealed in a briefcase). All of the PLCs were successfully shut down when the waveguide was pointed at their machine except the 15 volt model. This actually happened in the early race to be the best and the toughest PLC in industry. Toughness (in this case, noise immunity) was very important to the manufacturing engineer. What happened to the vendor of the 15-volt PLC? Today this vendor no longer is in the PLC market, displaced by the strong competition in the manufacturing marketplace.

There has been a great deal of interest about the origins of the PLC since those who were responsible for the origination of the PLC have retired and many have passed on. The competition for the selection of the first successful PLC is very interesting. It is repeated in the following article by Randall Brodzik, a writer for Control Engineering and found in the following 8/27/2014 article:

“**Inside the competition for the first PLC**

1971 The race to develop the first programmable logic controllers was underway inside General Motors’ Hydra-Matic Transmission Division in Ypsilanti, Michigan, in 1970. Three finalists had very different architectures.

In 1970, a fierce contest was being waged inside General Motors' Hydra-Matic Transmission Division in Ypsilanti, Michigan. At stake was program ownership of what would become the first programmable logic controllers (PLCs), and the outcome would shape not only the design of the first PLCs, but also the success or failure of their advocates.

PLC incubator

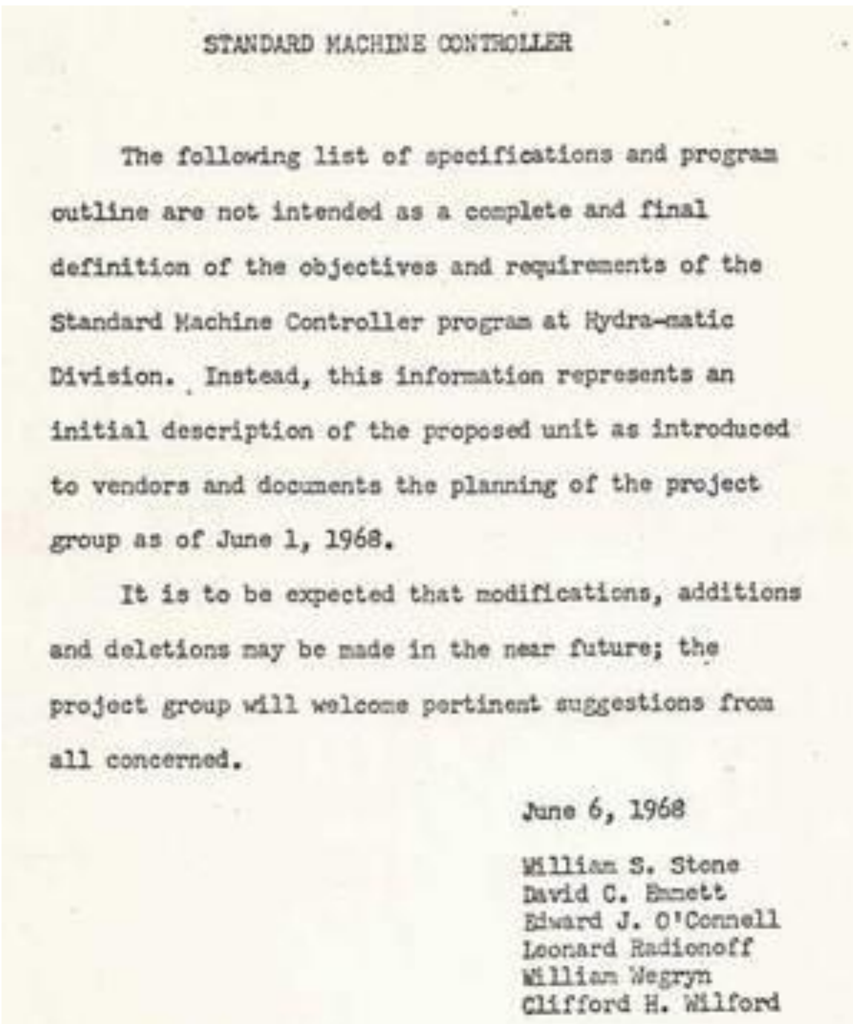
It was natural that GM's premier manufacturing technology group, Hydra-Matic, would be the incubator of the PLC. A nationally recognized technology leader, Hydra-Matic had installed one of the early IBM 1800 computers, the IBM 1801.

Richard Lundy, an owner of one of the original PLC suppliers, tells the story of when Ed O'Connell, a Computer Group engineer, didn't think GM was getting enough support from IBM. ‘An entire test line was running off of the one computer,’ Lundy said, ‘which made it very vulnerable, and Ed didn't like the support he was getting. So he called Watson.’ That's Thomas Watson, Jr., the son of IBM's founder and company CEO. Support for Hydra-Matic's IBM system quickly improved.”

In the following, Dave Emmett is described as the developer of the ‘Standard Machine Controller’.

"In April of 1968, a young Hydra-Matic engineer, Dave Emmett, proposed the development of what he called a 'Standard Machine Controller.' The controller he envisioned would replace the relay systems that controlled machine operations. Emmett was in charge of the Circuitry Group, and he envisioned a machine that would reduce maintenance costs, improve machine diagnostics, and decrease panel space.

Randy Brodzik, who later worked for Emmett, remembers that Emmett had a clear vision of what was needed: 'Dave said the goal was to develop a technology that would significantly reduce the time it took to make changes to a machine control sequence. With relays, first you had to do all the documentation, and then change all the wiring. It would take hours."



STANDARD MACHINE CONTROLLER

The following list of specifications and program outline are not intended as a complete and final definition of the objectives and requirements of the Standard Machine Controller program at Hydra-matic Division. Instead, this information represents an initial description of the proposed unit as introduced to vendors and documents the planning of the project group as of June 1, 1968.

It is to be expected that modifications, additions and deletions may be made in the near future; the project group will welcome pertinent suggestions from all concerned.

June 6, 1968

William S. Stone
David C. Emmett
Edward J. O'Connell
Leonard Radionoff
William Wegryn
Clifford H. Wilford

The letter at left was the original invitation for vendors to supply a system that would become the original PLC for GM.

Another group also was involved at GM. The description of this group follows:

"At about the same time, another group in Hydra-Matic was envisioning a different type of machine control system for GM. The Computer Division had hired Information Instruments, Inc. (3-I) to create a computer controlled assembly machine for the forward clutch line. This new control system had no limits to the number of elements in a ladder diagram, included parallel processing, and could incorporate complex Boolean equations.

Richard Lundy, a program manager for 3-I at the time, said the competition between the two Hydra-Matic Groups was ‘pretty dramatic.’ He described the Circuitry Group as wanting to duplicate existing ladder diagrams, while the Computer Group wanted to use nonsequential programming, similar to that used in current end-of-line test systems, to provide a more robust instruction set and reduce processing time. The advantages and limitations of each approach would become clear as the project progressed.”

Finally, a specification was written and shopped that would deliver a completed system. The RFP was a Request for Proposal. This form is described in the following:

“In April and May of 1968, work was done on a request for proposal that is remarkably simple by today's standards. The RFP, issued in June 1968, included only four pages of design specifications, including a requirement that ‘memory word length shall be at least eight bits.’ Douglas Brant worked on specifications for equipment coming into Hydra-Matic. He said that there was a strong rivalry between the Computer Group and the Circuitry Group in developing the specifications. ‘I guess it boiled down to people's territory,’ Brant said.

Of the many companies who received the RFP, three were selected for evaluation: Digital Equipment Corporation (DEC), 3-I, and Bedford Associates, a consulting firm. Its products were the DEC PDP-14, the 3-I PDQ-II, and Bedford Associates' Modicon 084. (The Modicon name came from MOdular DIgital CONtroller.)” See the following evaluation of the three finalists:

June 3, 1970

Design Considerations	DEC PDP-14	3-I PDQ-II	MODICON 084	Remarks
1. Input/output capability	256 inputs 256 outputs (power and storage)	256 inputs 254 outputs (power and storage)	256 ladder diagram rungs (no more than 4 inputs per rung)	PDQ-II addressing structure can handle up to 512 inputs. Requires rewiring of I/O box. DEC may have similar capability without rewiring.
2. Can outputs be used to store data during power on?	yes	yes	yes	
3. Can outputs be used to store data during power off?	Yes, if equipped with retentive feature	no	no	PDQ-II has no need for retentive memory because data may be stored in its core memory. See item 68 for retentive memory costs.
4. In addition to outputs used to store data, can memory store data?	no	yes	?	PDP-14 cannot because it has a read only memory.
5. Are inputs isolated?	no	yes	yes	Isolation is useful when multiple voltage sources are required or where sneak paths can occur. DEC inputs common in blocks of 8 inputs only.
6. Will basic power supply handle full system of 256 inputs and 256 outputs?	no	yes	–	If most systems require substantially fewer than 256 inputs/256 outputs, this may cause a PDQ-II cost disadvantage.
7. Are programmable timers available?	no	yes	yes	Programmable timers may be used over and over and cannot have their time changed unknowingly. They are awkward if time values for a particular operation must change often. The converse of both statements is true for analog timers. Analog timers also require additional wiring costs and mounting space.
8. Are adjustable analog timers available?	yes	not yet	no	
9. Operating temperature range	0-55°C. at case of NEMA Enclosures	0-60°C. Ambient	0-60°C. Ambient	Actual successful operation in environmental chamber to over 65°C has made it possible to raise the PDQ-II to 60°C.

-1-

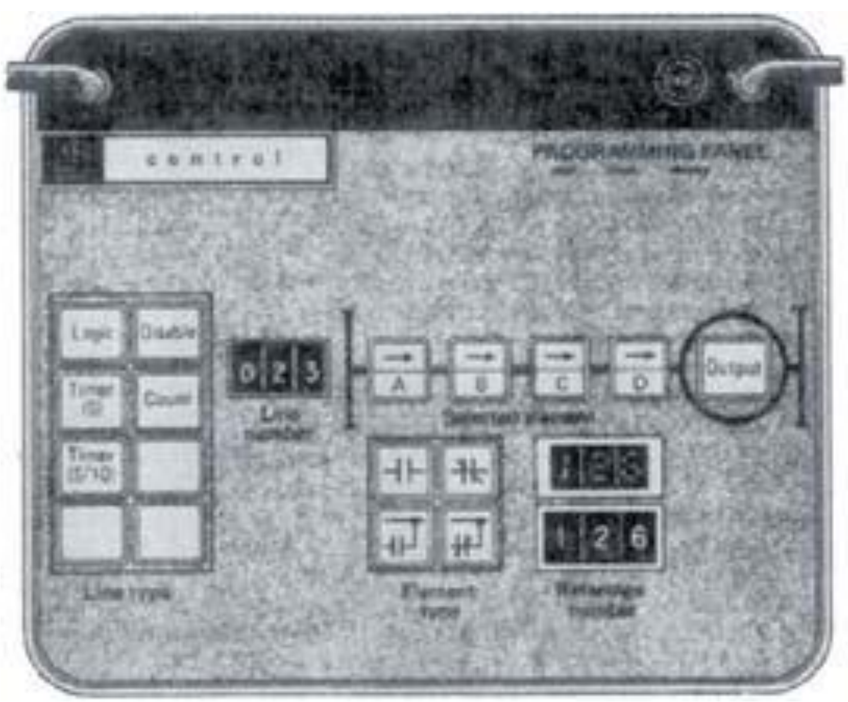
A PLC was installed at GM in June 1969. Its description follows:

“In June of 1969, DEC's PDP-14 was installed to control a gear grinder. Richard Lundy says that in spite of a very strong GM-DEC relationship, DEC's concerns about memory failure significantly limited the PDP-14's competitiveness. To make a change to the DEC program, ‘you had to send the client program in to them, and they would send you back a hardwired memory board,’ Lundy said. Having to remove a controller's memory from the plant to make changes would prove to make the DEC option unsustainable.

Richard Lundy's company, 3-I, was aligned with the Hydra-Matic Computer Group. Lundy said that during the evaluation period, the rivalry between the Computer Group and the Circuitry Group escalated. To the Computer Group, the advantages of higher level logic capabilities were clear and the selection of the PDQ-II was an obvious one. The 3-I solution for the forward clutch line, with its complex programming, had performed beautifully. Lundy said the PDQ-II quickly became known within the Computer Group as ‘pretty damn quick.’

Emmett and the Circuitry Group advocated strongly for the Modicon 084. The programming that was seen as a benefit by the Computer Group was seen as a detriment by Emmett's team. Because Modicon's programming language was similar to the Group's familiar relay ladder diagrams, it was expected to provide the smoothest transition and lowest cost for training and support. Randy Brodzik recalls that Emmett had an implementation view of the project: ‘He said that in order for this new technology to achieve wide acceptance, it needed to emulate what was already in place.’

As an added bonus for Modicon, the 084 was the only controller built into a hardened package, providing plant floor protection that the other two options did not. **And Modicon won**.”



This is the original Programming Panel for the Modicon 084. Yes, the numbers were thumb-wheel-switches which were very hard to turn and left fingers with callouses after entering a program. Note that there is no recording mechanism yet. That will come later.

Definitions of the PLC

NEMA's definition (NEMA or National Electrical Manufacturers' Association) of a programmable logic controller is "A digital electronic apparatus with a programmable memory for storing instructions to implement specific functions such as logic, sequencing, timing, counting, and arithmetic, to control machines and processes." This definition gives a good summation of the functionality of the modern PLC. A PLC must have each of these functional components to be considered a modern PLC. Primarily, the PLC is used to control machines. Programmable logic controllers have varied widely in what is considered a process or a machine. A few PLCs have even been used to control the duration of water for irrigation systems or lawn watering control. The view of what is a process has widened through the years. The modern PLC may even used to control the ingredients sprayed on a car moving through an automatic car wash.

A definition of the PLC from Liptak's *Process Control* states:

"A programmable logic controller (PLC) is an industrially hardened computer-based unit that performs discrete or continuous control functions in a variety of processing plant and factory environments."

These definitions of the PLC, either from NEMA or Liptak, give an overall picture of a computer with a special functionality partial to the automation market and able to control a variety of different machines. The purpose of this text is to more fully understand the use and programming of a PLC and understand applications of PLCs in factory automation.

In the US, the general trend was for the PLC to emulate or copy ladder logic as employed in the standard relay ladder logic or schematic drawing of the day. In other parts of the world, as the PLC was studied, this was not necessarily the case.

In Germany, for instance, STL or Statement List was introduced. From the Siemens book: *Milestones in Automation* by Arnold Zankl is the following quote (p. 54):

"Siemens had at first used STL programming exclusively and had been very successful with it. It seemed reasonable to program something in the way people think of it and describe it verbally. High education standards in Germany and Europe also supported this approach.

In the US, where training for skilled workers was generally less intensive than in other countries, the ladder diagram, derived from the circuit diagram, dominated from the start."

While this statement may not be meaningful to some, the choice of American PLC vendors to pursue Ladder logic while the German PLC vendor Siemens pursued another language, STL, was significant and has had an impact on the marketplace. While the German machine later embraced Ladder as a second language, the reverse has not occurred to any degree with the American market keeping strong ties to Ladder and not embracing other languages. Understanding a variety of languages and being able to program in the language most appropriate for the application is of primary importance and should be a student's goal.

Evolution of the PLC

From that beginning, PLCs have grown in popularity and capability to the extent that they can be found in some part of almost every industry. PLCs are produced by a variety of companies worldwide and several companies that made PLCs in the last 30 years have been forced out of the business by ever increasing competition.

Listed in Table 1-1 below is a time line of the evolution of the PLC from its inception to the modern PLC. See Table 1-1 for a portion of the time line for PLC development. Table 1-2 describes the Siemens PLC and its parallel development in the European market place.

"History of Programmable Logic Controllers (PLCs)

Table 6.5b from *Process Control* by Liptak

1968	Design of PLCs developed for General Motors Corporation to eliminate costly scrapping or assembly-line relays during model changeovers.
1969	First PLCs manufactured for automotive industry
1971	First application of PLCs outside the automotive industry
1973	Introduction of "smart" PLCs for arithmetic operations, printer control, data move, matrix operations, CRT interface, etc.
1975	Introduction of analog PID (proportional, integral, derivative) control, which made possible the accessing of thermocouples, pressure sensors, etc.
1976	First use of PLCs in hierarchical configurations as part of an integrated manufacturing system
1977	Introduction of very small PLCs based on microprocessor technology
1978	PLCs gain wide acceptance, sales approach $80 million
1979	Integration of plant operation through a PLC communication system
1980	Introduction of intelligent input and output modules to provide high-speed, accurate control in positioning applications
1981	Data highways enable users to interconnect many PLCs up to 15,000 feet from each other. More 16-bit PLCs become available. Color graphic CRTs are available from several suppliers
1982	Larger PLCs with up to 8192 I/O become available
1983	"Third party" peripherals, including graphic CRTs, operators' interfaces, "smart" I/P networks, panel displays, and documentation packages, become available from many sources

Table 1-1 Early History of the PLC

The PLC is an American invention but the European manufacture Siemens was quick to augment its automation offering to include its own PLC. The first PLC comparable to the American equivalent is the S3, Siemens' first storage programmable PLC. From *Milestones in Automation* by Arnold Zankl of Siemens is the following history of the Siemens history of automation control and the PLC in Europe.

1957	First prototype controllers from Siemens with germanium transistors
1964	Second generation control and switching systems introduced with board-design using Euro-card
1971	Third generation controllers introduced S1 - using hard-wired logic S2 - using electronic sequence processor

1973	S3 – first storage programmable PLC from Siemens (Simatic S30)
1976	First fault-tolerant and safety-related PLC for burner controls
1979	Simatic S5 introduced
1984	S5-135U multiprocessor PLC with up to 4 central processors for high-speed application
1986	Introduction of distributed I/O for Simatic
1988	Introduction of S5-155U with multiprocessor technology including floating-point arithmetic
1994	Introduction of Simatic S7
2000	Integrated drive systems
2002	TIA Totally Integrated Automation links data to IT domain

Table 1-2 History of the Siemens PLC

The advancement of PLC technology continues to the present day with newer PLCs introduced to meet the customers' demands. While early milestones of Table 1-1 show a general direction of the PLC toward a modular device capable of communicating to I/O, a programming terminal and to other PLCs, the PLC of today is also capable of communicating over a highway to a Human Machine Interface (HMI). PLCs are also capable of reading I/O from a variety of sources including I/O through other PLCs. I/O may be distributed using a number of different configurations. Some I/O may use a proprietary network while other I/O may use a public network or highway such as Ethernet. Connectivity to intelligent devices has increased through the years as well with PLCs interfacing to most types of manufacturing equipment such as bar code readers, weigh scales, servo motors and other intelligent devices.

At one time PLCs were simply known as the PC. With IBM and the introduction of the personal computer, naming standards had to change. PC was used for the personal computer so PLC manufacturers changed the name to Programmable Logic Controller. At some time during this early development of the PLC, Allen-Bradley trademarked the PLC name. The name PLC has been used by most engineers to describe the product known today as the programmable logic controller.

Liptak considers the reasons in his book for the popularity of PLCs. They are:

- "Ease of programming and reprogramming in the plant
- A programming language that is based on relay wiring symbols familiar to most plant electrical personnel
- High reliability and minimal maintenance
- Small physical size
- Ability to communicate with computer systems in the plant
- Moderate to low initial investment cost
- Rugged construction
- Modular design" (from *Process Control* by Liptak)

The programmable controller of today has grown in capability and shrunk in size from the first and second generation PLCs of the 1970's and 1980's. Where relays, timers, and counters were the concern of early design, inclusion of numbers and numeric manipulation quickly became part of the PLC. Instructions for comparing, adding, subtracting, multiplying and dividing of numbers were added to the language. As engineers demanded more sophistication from their PLC, designers turned to the microprocessor and, in part, emulated the instruction sets of the popular microprocessors of the day. Shift, Rotate, AND, OR, XOR and other instructions from the microprocessor were added. Finally added were floating point numbers and more sophisticated instructions to handle complex algorithms such as the FIFO and LIFO stack operation and the PID block.

The German manufacturer Siemens first introduced Statement List as its primary language for the PLC. Statement List or STL was introduced to incorporate the power of the microprocessor. This was accomplished by implementing a version of assembler language, a more difficult language to master but one that was more flexible than Ladder. In general, the German Siemens is considered more powerful but difficult to learn. Implementation, on the other hand, is usually simpler with the Siemens approach. As the text is used, hopefully these opinions will become more evident.

Step 7 is the programming language used to program the Siemens PLC. Newly introduced is Step 7 Basic for the new Siemens 1200. Later, the same language introduced for the 1200 will be introduced in the 300 and 400 lines of processors. STL will give way to SCL, a Pascal-type language and the assembler language will be dropped. While different languages will be used for different applications, Ladder or LAD will suffice for many early applications. Becoming familiar with LAD and its use will be a primary early concern. Later, use of Function Block and SCL will provide methods to better describe some applications. Also, the use of grouping of logic into function blocks will provide a method of encapsulation of logic similar to the programming language C's function, structure and class environment.

A look at hardware is important but will not be stressed. The use of language for development of control of a process will be stressed. Hardware changes with cost and function. Use of hardware for writing programs is necessary. Study of certain types of hardware instead of the programming will not be emphasized.

S7 refers to the newer lines of processors from Siemens. There is a S7-200 which is being phased out with the 1200's introduction and the S7-300/400 lines. The S7-200 has little in common with the 300/400 since it was developed by the TI engineers at the time that Siemens purchased the TI PLC division in the early 1990's. The earlier S5 processor also is being phased out. It is an older model, a contemporary to the Allen-Bradley PLC-2 and PLC-5 processor.

From an article on the Omron website:

"S5 is an old model, who was a contemporary of AB-2-PLC and PLC-5. S5 is the Intel 8031 series of microprocessors, which Siemens has licensed the production is based. They also own a coprocessor, which is used in some models S5 gear single Boolean logic. Siemens pressed overwhelming number of features from simple 8-bit processor. Punishment for this, however, that a statement S5 was very primitive and only slightly higher than the 8031 assembler. Look at the 8031 assembly and S5 Instruction List (IL), and you see the similarities. In contrast, AB has a different approach and use the mini-computer technology in their designs early PLC. Early

PLC2S used 4 processors AMD 2900 bit slice to a separate 16-bit processor. It was expensive in terms of hardware, but they had fewer restrictions in the software caused."

The use of PLCs gives one the reason why a language such as LAD is used instead of C for writing programs for manufacturing applications. It is simply quicker and cheaper to write. It is interesting to note that the PLC is perhaps changing its name again (from PC to PLC first) to PAC (Programmable Automation Controller. This new name may never catch on with some but for the newer embedded control processor of today, it may.

The Allen-Bradley PLC has endured from the beginning. After overtaking Modicon in the early 1980's, A-B never looked back at competition in the US (except for today's challenge from Siemens). The need for a fundamental knowledge of Allen-Bradley programming is required as well as Siemens. Today, both are required.

The PLC's Microprocessor Background

PLCs are closely related to computers because of their computational abilities and logic solving speed. PLCs receive inputs, solve logic, and set the appropriate outputs based on the inputs and programmed logic. Unlike computers, a program in one part of the PLCs memory cannot lock out a program in another part of memory. An errant program statement cannot cause all programs to stop execution, a very important concept if electricians and people other than the original programmer are to be allowed to change program statements. However, with advances of instruction sets to include program interrupt instructions, it is not always guaranteed that one part of a program will not interfere with another and even stop execution of the system.

The PLC can be broken down into these four basic components:

1) Central Processing Unit (CPU)
2) Memory
3) Inputs and Outputs (I/O)
4) Power Supply

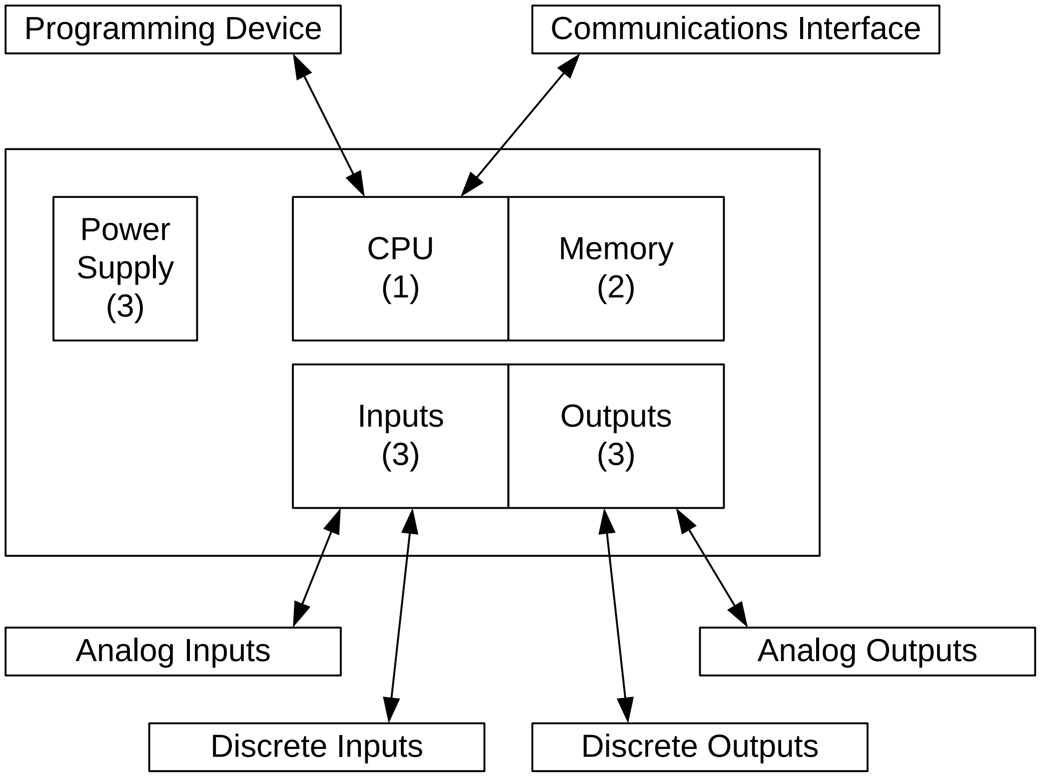


Fig. 1-1 Block Diagram of the PLC

Storage memory provides for variable storage and retrieval. Recipes and other table information reside here as well as the variables used in logic. Programming more sophisticated applications such as recipes hint at the using the PLC for more sophisticated applications. A basic rule for LAD logic has been that it must be maintainable by people other than professional computer programmers. This includes maintenance personnel who will be used to maintain the system once installed in a plant. As programs grow in complexity and length, ways have been sought to again make the logic simpler to implement and trouble-shoot. Although memory is used for a number of functions, the use of LAD only as a language in the PLC falls short in its ability to handle complex data storage and math arrays. Also, because of the complexity of larger programs, a change in one area can affect other areas, causing hardship when changes need to be made. Language selection is the most important aspect of application development.

Many “religious wars” were waged in the 1990’s involving the choice of the PC or PLC for automation. Those using the argument that the PLC would be out and that PCs would be found in all applications, while not discounted, were not considering the unique capabilities that PLCs had that the PC could not be counted on to deliver. The PLC could be counted on for their ruggedness and ability to restart even in the most difficult conditions. Distributed I/O was counted on to read on a consistent basis all the I/O in the system and report to the cpu. What PCs were capable to deliver was the openness and flexibility and very high performance levels. PLCs have always lagged their PC cousins in these areas but have not given ground based on their overall advantages that cannot be discounted such as ruggedness. From the Siemens text *Milestones in Automation* (pg 154) by Arnold Zankl, Zankl describes the choice of PC or PLC:

“Experts estimate that in 2005 over 90% of all automation solutions were implemented with programmable controllers. So, contrary to many forecasts, the PC has not replaced the standard PLC.”

The PC will not displace the PLC but which PLC emerges as the dominant platform or language is still to be determined.

PLC Hardware and Safety

Inputs are devices such as limit switches, push buttons, photo-eyes, proximity switches, and relays from other systems. Outputs are devices such as lights, relays, motor starters, solenoids. Motors and other outputs capable of producing motion must be controlled in such a way that the machine is run safely. The PLC must control the process safely from the moment the machine is turned on until the machine is turned off for the day or year. PLC programs must be written in such a way that all events of the machine are monitored in order to guarantee a safe and orderly control of the machine.

PLCs were not viewed as being as safe as relays when first developed. Engineers did not trust the computer as much as the simpler relay. This view changed as PLCs became more reliable. Interestingly, PLCs were made to resemble rugged equipment so engineers would accept them in the rugged industrial environment. Today, PLCs are manufactured that easily could fit in the palm of one's hand. These PLCs have the same error checking and hardware ruggedness as the big PLC of the 1970's. Ironically, these little PLCs have much more functionally with far superior instruction sets and faster scan times. They remind one of the evolutions of the hand-held calculator or the personal computer as the old box has given way to the more powerful yet less expensive new box.

The design of the functional aspects of the system must include all the electrical and programming aspects of the system as well as the mechanical, fluid power, air power and guarding systems.

The safety system runs in tandem with the production system. It must not impede the production system but run in parallel with it and oversee any hazards. The focus of the two systems is somewhat different. The production’s system focus is on throughput. The safety system is focused on protection.

A safe system is not entirely possible and a risk is always present. The purpose of the safety system is to reduce the risk to acceptable levels. **An absolutely safe system is not possible.**

How PLCs Solve Logic

Programming a PLC requires a program that solves the same basic logic again and again guarding against the unexpected and keeping the machine running in an acceptable manner. There must be an orderly flow to the logic to control the machine. PLCs all follow the same general format utilizing the following four steps:

1) initialize from a safe state - usually off
2) sense inputs
3) solve logic in the program
4) outputs turned on or off to mechanical devices

Steps 2, 3, and 4 are repeated again and again very rapidly to provide the orderly solving of logic and simulation of relays, timers, and counters. The process, while involving a number of complex actions, breaks down into the following:

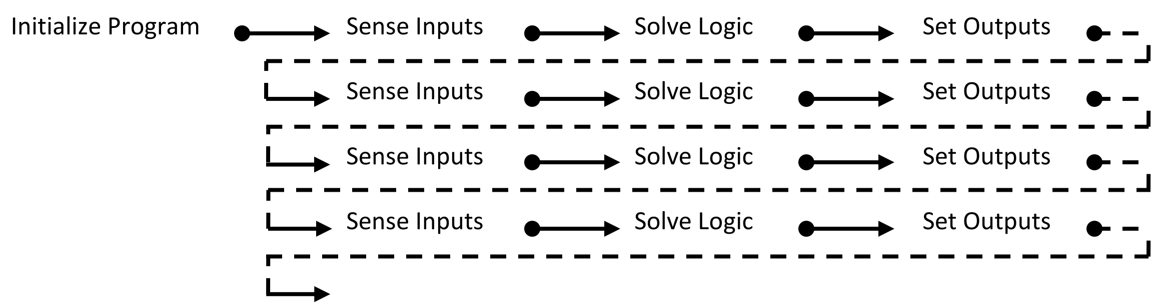


Fig. 1-2 Solving the PLC Program Scan by Scan

This process repeats each one to 5 millisecond and is referred to as the scan of the PLC. Each repeated sensing of inputs, solving of logic and setting of outputs is a scan of the PLC.

Siemens describes the cycle time of a processor as the entire time needed to read the inputs, execute the program one time and process the outputs. Their blocks are divided into networks and each network is divided into a number of statements. The time line is described as follows:

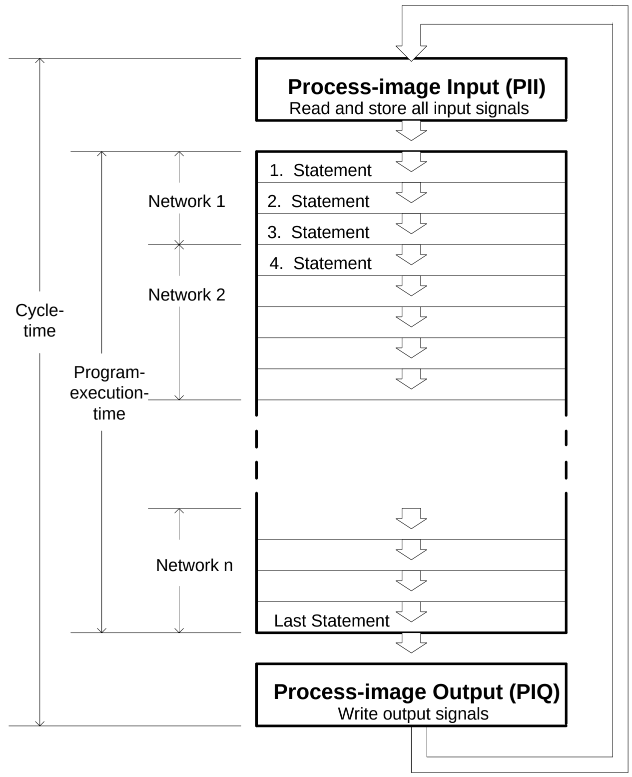


Fig. 1-3 Cycle Time of the Siemens PLC

There must also be an easy way to start and stop the PLC and its program from executing. This is referred to as the modes of *stop* and *run* (or *program* and *run*). A running PLC program is analogous to turning on the power to a relay panel. When power is turned on, the machine responds in an orderly way per the design of the engineer and the wiring of the electrician. Likewise, when the PLC is turned from *program* to *run*, the CPU begins executing the application based on the rules of the program. These rules form the basis for the operation of the system. First, outputs are defaulted to an off or pre-defined state. Then, as the inputs are read, the program is solved sequentially one logic statement at a time. Outputs are turned on or off based on the program and the input conditions. If a program is written and configured correctly, the outputs should fail to a safe state. Some outputs may be required to turn on if the processor fails. Most devices are designed so that if the controller fails, the output turns off and the device being controlled returns to a safe state. In other words, most but not all outputs turn off when the program returns to the *program* or non-running state. For safety's sake, design a system to return all outputs to the safe state, either on or off, when the program is not actively running a machine. And when the program is running a machine, devices are programmed to run in only a safe manner. This is a main concern of the programmer, to design a safe machine as well as a working machine.

Richard Morley describes in his own way the invention of the PLC through a number of youtube videos. While they are not the most watched videos on youtube, you may try them sometime. It is interesting to note that the philosophy of creating the PLC was for simplicity and the use of if-then statements. Really, all the statements are of this type – if-then.

PLCs in World Economy

Manufacturers of PLCs have been many and varied in the past with a stiffening competition over the last twenty years. The effect has been a thinning of the ranks of PLC vendors. It costs much more to bring products to market than it did a few years ago. Foreign competition has caught up and in many ways surpassed domestic PLC manufacturers' technology. A number of buy-outs, consolidations and joint operating agreements have thinned the number of PLC manufacturers to a few. Allen-Bradley is the mainstay American company producing PLCs. Also in the US are General Electric and the combined Modicon-Sq D PLC organization. Siemens in recent years has made significant inroads in the US market and overtaken all but Allen-Bradley. In Europe, the dominant PLC manufacturer is Siemens and in the Far East, Japan's Omron and Mitsubishi. The emerging manufacturing base in China favors Siemens although competition is very strong between many PLC manufacturers in China as well as in Mexico and Central and South America.

PLCs vary in size and type in a way similar to other manufactured products. Common to most manufacturers are the full size, compact, mini, micro and nano versions. Not surprisingly, the Japanese tend to dominate the mini, micro and nano end of the product while the German and Americans tend to dominate the larger models. This is changing, however, with a number of American models getting smaller and smaller as well as the Japanese pushing upward into the larger models.

Global PLC Markets Share – Historical Trends

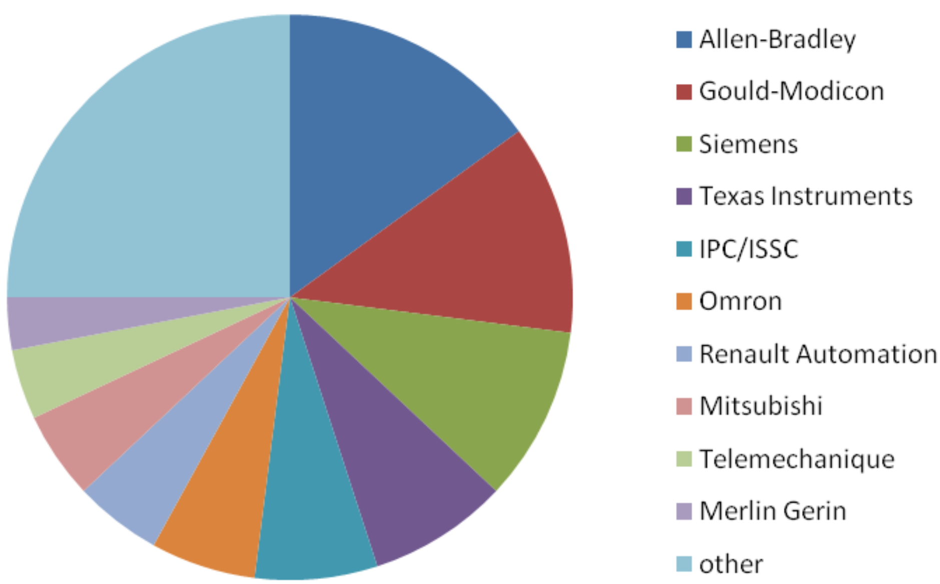


Fig. 1-4 Global PLC Market share in 1982/83

A view back to the early 1980's gives the major PLC vendor as "other" with the largest brand being Allen-Bradley. They were just ahead of Gould-Modicon who had dominated much of the earlier decade of the PLC market. Third was Siemens followed by Texas Instrument and IPC/ISSC. Four of the top 5 PLC vendors were from the US and the American PLC vendors tended to dominate.

By 1993, Siemens had overtaken Allen-Bradley with about 25% of the world market. Allen-Bradley was still first in the US but the market had globalized. Mitsubishi was third and Omron fourth. Modicon had been purchased by AEG and was still in fifth place. Texas Instruments was now part of Siemens as purchases of controls companies by other controls companies intensified. See Fig. 1-5 below:

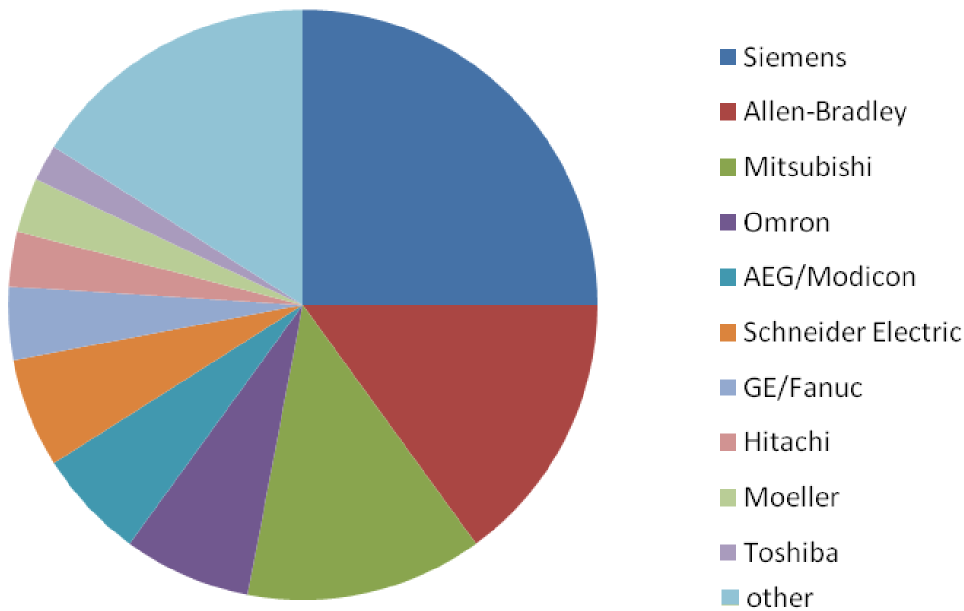


Fig. 1-5 Global PLC Market share in 1992/93

Since 1992, Siemens has continued its place as number one and widened its lead somewhat. Allen-Bradley continues in a strong second place. One company not known for its PLC entry appeared in the 2007/2008 market survey – ABB. ABB is a very strong controls company in the world and had been absent from earlier surveys but appears on this analysis although in a small position overall.

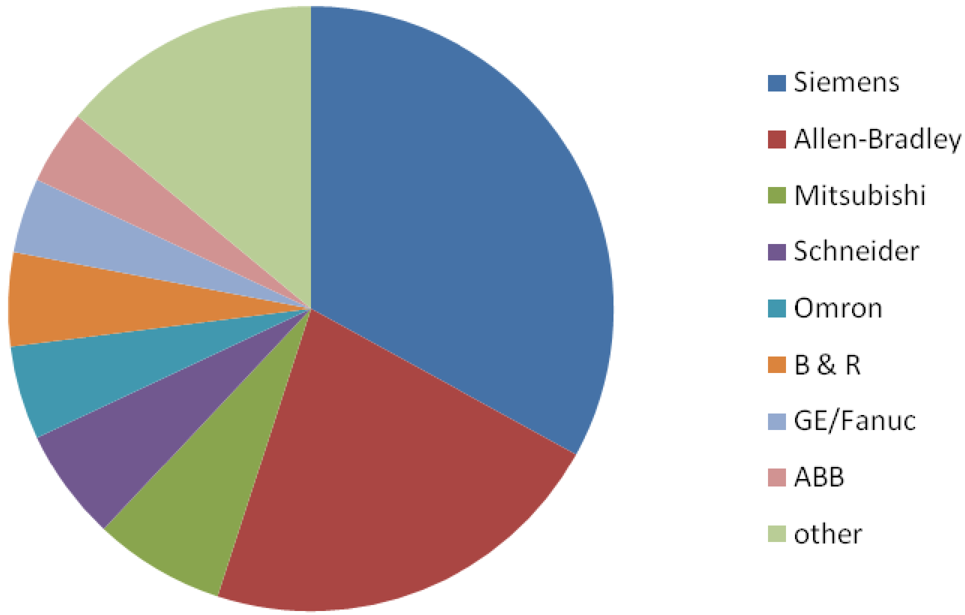


Fig. 1-6 Global PLC Market share in 2007/2008

Choice of a PLC must include Siemens as well as Allen-Bradley for the US market. Siemens' strength can be seen in its attention to detail and its global strategies. One should thoroughly study the languages found in the Siemens PLCs in order to write logic for the world market. STL (Statement List) should be learned as well as LAD (Ladder) and FBD (Function Block Diagram). It is not enough to insist only on LAD with an occasional subroutine written in FBD or other language.

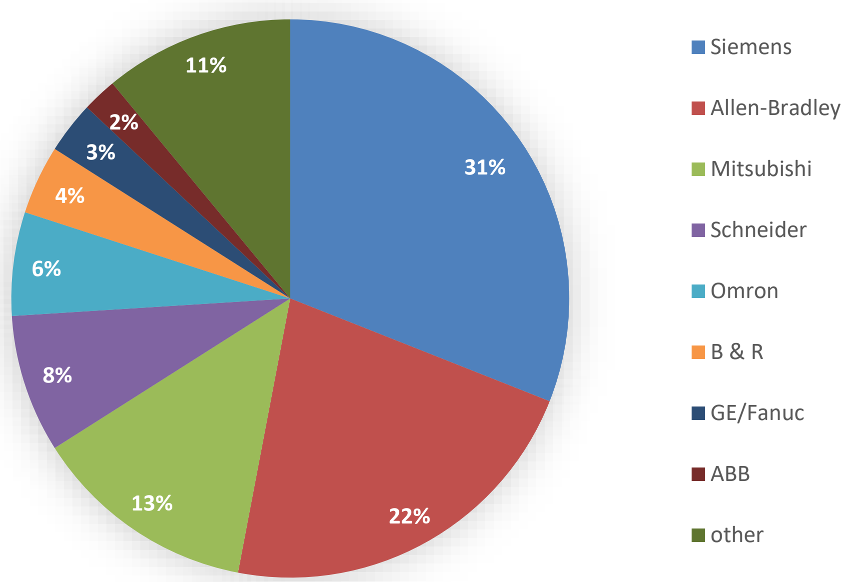


Fig. 1-7 Global PLC Market June 2018

A Long Quote from Siemens Milestones Text

From Choosing a PLC from *Milestones in Automation* by Arnold Zankl – Siemens: (pg 53)

“**Programming of the first programmable controllers**

To those involved in Europe and the US, it was clear from the start that, to be widely accepted, programmable controllers would also have to be easily programmable by skilled workers, electricians, installation engineers, and manual workers – not just by expert programmers. It was important to meet the target groups on familiar territory with regard to their existing level of technical know-how.

The plant electrician could, of course, read circuit diagrams. Engineers were somewhat familiar with Boolean algebra or could easily learn it. And young people who already know how to use progamming languages could quickly learn how to use menmonic instructions.

These basic programming languages had asserted themselves relatively quickly, and these are now defined in the global standard IEC 51131.1: Ladder diagram, function block diagram and statement list resp. instruction list.”

The Siemens text continues on the next page (54) as follows:

“Siemens had at first used STL programming exclusively and been very successful with it. It seemed reasonable to program something in the way people think of it and describe it verbally. High education standards in Germany and Europe also supported this approach.

In the USA, where training for skilled workers was generally less intensive than in other countries, the ladder diagram, derived from the circuit diagram, dominated from the start.”

To study the Siemens PLC in the US, one must recognize a change in attitude that the European worker has accepted for a much longer time – that programming must be flexible and written in the language best suited for the application. The American student must follow the rigors of the STL or SCL and FBD langauges as well as LAD in order to successfully compete in the marketplace today.

IEC 61131-3

IEC 61131-3 was intended to achieve the long-term aim of creating user software largely vendor-independent and being able to port it to devices of difference to system integrators who want to use different target systems. The chart below compares Allen-Bradley, Siemens and the IEC 61131-3 international PLC language:

Allen-Bradley RSLogix	Siemens Simatic	IEC 61131-3	
Ladder Relay Ladder	LAD Ladder Diagram	LD Ladder Diagram	Based on circuit diagram
FBD Function Block Diagram	FBD Function Block Diagram	FBD Function Block Diagram	Based on switching circuit systems
SFC Sequential Function Chart	S7-Graph for sequencers	SFC Sequential Function Chart	For sequential control
	S7-HiGraph State-transition diagrams		For asynchronous processes
	CFC Continuous Function Chart		In the form of technology oriented diagrams
	STL Statement List	IL Instruction List	Similar to assembler
ST Structured Text	S7-SCL Structured Control Language	ST Structured Text	Pascal-like high-level language

Table 1-3   PLC Languages

(The following article by Jeremy Pollard is re-printed with permission by the author. The article will explain what IEC 611131-3 is and what the benefits are. While this topic is beyond the scope of an introductory chapter, this seemed to be the only place to include it.)

IEC 61131-3 by the Numbers

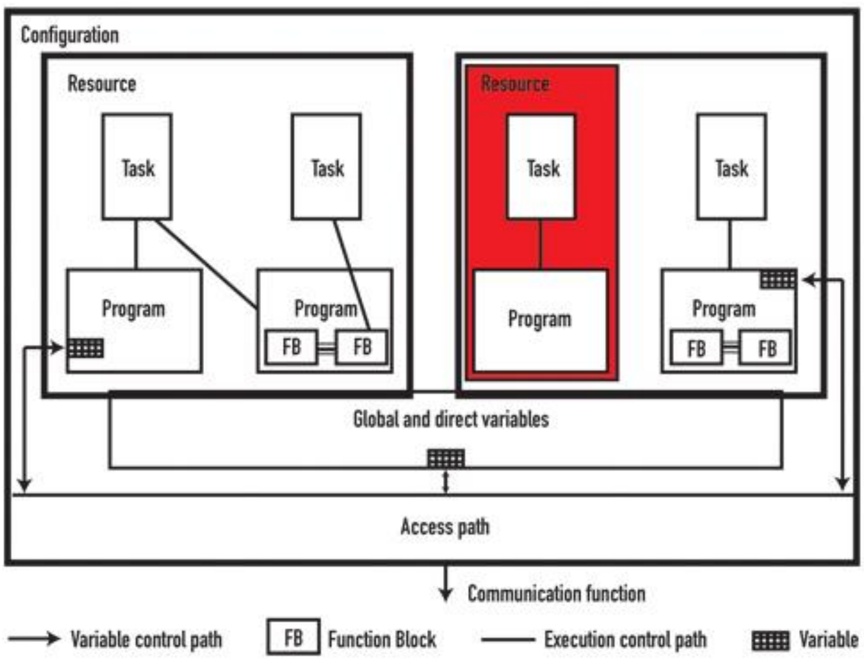
The Programming Standard for Controllers: What It Is, What It Is Not and Its Benefits

By Jeremy Pollard

The International Electrotechnical Commission's (www.IEC.ch) 61131-3 international standard "Programmable Controllers—Part 3: Programming Languages" was published in 1993 after 15 years in development and first used for PC-based control programming in the 1990s. Typically programmable logic controllers (PLCs) had their own vendor-developed programming platforms, but in the past decade, major PLC vendors from Europe and North America initiated and supported the new standard with their new programming platforms.

The intent of IEC 61131-3 is to normalize PLC and control systems' programming by standardizing functionality such as program entry, instruction visualization, data types and syntax. The general requirements section includes models for software, communication—external as well as internal instruction and variable parameter passing—and programming. The model functions are as follows:

- The software model introduces the following configuration concepts: resources such as CPUs; tasks such as executable application software; named variables used for storage; and communication paths (Figure 1). An IEC-based hardware "client" could run multiple tasks in one configuration or have multiple configurations.
- The communication model specifies how data is passed to different tasks or configurations or within the same task. Global variables are introduced.
- The programming model is tied closely with the concept of common elements, which enable the use of common data types, variable declaration and data formats such as dates and times. Program organizational units (POUs) also are common elements.



IEC 61131-3 Can Go Beyond

Figure 1: A conventional PLC's software (on the right, in red) consists of one resource, running one task and controlling one program. However, a controller using IEC 61131-3 (on the left) can go beyond PLCs by providing a multi-tasking, real-time environment that can be used in a broader range of applications without requiring users to learn added programming languages. If vendors allow it, this multi-tasking means a function block-controlled task could run along with a ladder logic task with data sharing and global variables. At the highest level, the entire software required to solve a particular control problem is called a Configuration. Within it, one or more resources can be defined. A resource is like a CPU in the system, and it includes one or more tasks. These control the execution of different parts of programs, called functions or function

blocks, which are the basic building blocks of data structures and algorithms. All of these can exchange their information via the global or direct variables, even to the outside world via communication functions such as OPC.

These POUs were developed to reduce the often-implied meanings of instructions and blocks. There are three types of POUs defined in the standard. The program POU is the program; no news there. The function-block POU is a program block with inputs and outputs used for tasks such as timers and counters. The function POU lets various program elements extend the instruction set of the configuration. The intent of these POUs is to reduce the programmatic differences between suppliers, so a timer is a timer, regardless of who provides it.

In addition, the standard defines data types, such as Boolean and integer. It also defines the use and format of derived data types and functions, which must conform to the previously mentioned standard data types.

Meanwhile, the programming model extends a PLC-type programming environment with three languages and two graphical representation languages. Ladder logic is most used in discrete applications and is the most widely coded language in automation today. It is mainly used and supported by the North American PLC vendors.

Instruction List, which is a rudimentary, assembler-type language has been a European programming staple for years. The third language is Statement List, which is similar to Pascal.

As a result, the graphical languages are Sequential Function Chart and Function Block. These are graphical representations of processes and have underlying code written in one of the three languages or in an alternate language, such as C++.

Finally, the standard defines the abstraction between the IEC model and the PLC/control hardware by using device tags and not addresses. Similar to programming in a high-level language such as Visual Basic, it allows the programmer to define a memory point, not by address, but by using a descriptive name. The standard defines the characters you can use in the description. The outside world is tied into the model based on the hardware used.

The desired effect of the standard is to reduce the user's learning curve between vendors, and with work beginning as early as 1978, it was developed to meet that need.

What IEC 61131-3 Is Not

The published IEC 61131 standard encourages extensibility, meaning any company that writes an IEC 61131-based product can add things to its "standard" product as long as it tells the user what changes and additions have been made relative to the standard. However, IEC 61131 isn't a standard. It is a specification for vendors that want to develop a control software programming environment according to a set of guidelines.

PLCopen (www.plcopen.org), the global association supporting IEC 61131, says you can't have a standard without certification. However, no one and no company can claim that its ladder logic editor creates compliant code. There's no way to certify it. And yet, one of Rockwell Automation's web pages for RSLogix 5 makes claims about the "RSLogix family of IEC-1131-compliant ladder logic programming packages."

In addition, even if it did create compliant code, Rockwell Automation would just need to say where the extensions are and what effect they have on that compliancy, and then they can claim compliance.

To make sure I wasn't being overly biased, I enlisted colleagues on the Automation List at www.Control.com. "The conformance criteria are so general, it is virtually meaningless," says independent consultant Michael Griffin about compliancy.

Also, a published PLCopen document states, "The overall requirements of IEC 61131-3 are not easy to fulfill. For that reason, the standard allows partial implementations in various aspects. This covers the number of supported languages, functions and function blocks. This leaves freedom at the supplier side, but a user should be well aware of it during his selection process." This doesn't help us very much.

The original intent of the specification or standard was to create a common platform for control software development. When the programmable controls industry emerged in the 1970s, all software was written using dedicated hardware and firmware. With the advent of the personal computer, vendors developed their own development environments. It is no different today. The software programs that have been developed support only the vendor's hardware. There is no common platform here. We are no further along than we were 20 years ago.

For instance, using RSLogix 5000 or Schneider Electric's Unity is no different than when we used fixed programming terminals. You can't share programs, although PLCopen is trying to develop an XML transfer specification.

I anticipate that vendors will have import routines, but few, if any, will have export routines.

Likewise, the specification does not cover off-the-scan issues and program-execution methods. Process behavior most likely will change on different hardware platforms, since the software environment is different.

Some of the visualizations of the code are supposed to be close. If you look at ControlLogix5000, which Rockwell Automation states is IEC 61131-compliant, and compare the visualization with that of KW Software, for example, there won't be much similarity.

Tag-based programming is a plus, but even in the 1980s, many products had the ability to use symbols. With IEC 61131, you have to use symbols, but the length of the tag names is variable. You can't use a 40-character symbol with a product that supports only 20.

To me, IEC 61131 is similar to SQL, UNIX or C. They're standard products to some degree, but they're not called standards. The varying flavors don't interoperate, because, before the world went Microsoft, some compilers had very different syntax and ways of doing things.

In fact, Julien Chouinard, managing director of ICS Triplex–ISaGRAF, calls IEC 61131 a compromise. However, while I think IEC 61131 has fallen short of expectations, this doesn't mean there are no benefits to the specification.

Benefits of IEC 61131-3

Now, make no mistake about IEC 61131-3. We're still in the same environment we were in before its creation. Rockwell Automation, Schneider Electric, Siemens Industry, GEIP and Wago all have their own programming environments. The business of automation won't allow for real interoperability of these competing products.

The main difference from 20 years ago is that there is some form of commonality. There are some good reasons to use an IEC 61131-based product. Tag names permit abstraction of the hardware I/O, so databases from product to product could be maintained externally. All will use tag names-based variable allocation and typically the common elements, such as Boolean, will behave the same. That doesn't mean, however, that all vendors support all the common elements.

Most products will have all five languages. A large benefit of an IEC 61131-integrated development environment (IDE) is access to the function block part of the software. The power in function block programming is tremendous. This is the type of programming the older DCS products had 20 years ago. You can program a very complex part of the application and have it hidden behind a block. Some argue that this improves troubleshooting capabilities.

In return, it requires a better programming skill, as well as a good handle on the project scope and project management. This is foreign territory to many ladder programmers.

A selling feature of the standard is the ability to develop control strategies in the right language. While I agree with this, the plant floor maintenance staff usually doesn't include a structured text guru. This is scary for the controls engineer who develops the application, so oftentimes he'll just use ladder logic. Therefore, one of the major benefits of function-block programming is left untapped as a result.

Proponents of IEC 61131 swear by its training and investment benefits. They say if you use Company A's IEC 61131, and later you have to use Company B's IEC 61131, then you already know how to program the hardware. This is not necessarily so, but there is some truth to it.

Many companies use a development package from 3S, some use a product from Softing, and others create their own. If an OEM used IEC software from four different hardware vendors, and they all licensed their software from one company, the OEM stands a better chance of being able to migrate the learning curve to the new hardware and of being able to reuse the software.

Will the training cycle be shorter? Probably. But from vendor to vendor, there is some level of commonality that can be borrowed, but the impact may be limited.

Will IEC 61131 help the maintenance person trying to solve a problem at 3:00 a.m.? For most user companies that standardize on a single hardware platform, the answer is "no." They'll get used to whatever the software tool is. IEC 61131 isn't important.

It might be important to you if you leave your company and have IEC 61131 on your resume. It surely can't hurt.

If you're a machine builder or a manufacturer with multiple vendors that all use IEC 61131, maybe it helps. There are some inherent similarities that could help getting refamiliarized with a hardware platform or a software troubleshooting tool, but I think this is a personal function rather than a software tool function.

No doubt IEC 61131 is here to stay. Early misunderstandings of the benefits, and resulting misconceptions of what an IEC 61131-based product actually was turned off many potential users.

Open is as open does. IEC 61131 is the same. It is simply a platform for a software product design tool for hardware. It provides some benefits or the OEM and the user, but mainly the vendors benefit, in my perception.

It is no panacea. It can help in our quest for a better automation platform, but we will leave that for IEC 61499, a true standard. As long as we don't screw it up.

Jeremy Pollard has been writing about technology and software issues for many years. He is publisher of "The Software User Online," has been involved in control system programming and training for more than 25 years and was previously North American managing director of PLCopen (www.plcopen.org), which drafted IEC 61131.

Some Key Features of IEC 61131-3

Structured Software – through use of configuration, resource and function blocks
Strong Data Typing – by using languages that restrict operations so they only apply to a appropriate data types
Execution Control – through use of tasks
Complex Sequential Behavior – through sequential function charts
Software Encapsulation – through use of function blocks, structures and complex data types

The standard allows for the creation of programs that are highly structured, yet flexible. It provides tools for creating programs with complex sequences and promotes the creation of reusable modules or function blocks.

(end of article by Jeremy Pollard)

Other Topics Close to PLC's

Topics of interest in a PLC text in addition to programming languages should include:

1	Distributed I/O
2	Fabrication of a system including the cabinet, lay-out, drawings
3	Fault tolerant/safe systems
4	RS485 networks including Profibus and DeviceNet, simple protocols and simple ASCII
5	Use of HMI (Human Machine interface)
6	Ethernet networks, wireless
7	Relation between PLC and CNC and PCS (DCS), robotics
8	Sensors and actuators
9	Batch Systems
10	MES (Manufacturing Execution System) to include a. Business Modeling Factory b. Enterprise Asset Management c. Plant Maintenance Management d. Dispatch e. Trace and Tracking
11	ERP(Enterprise Resource Planning) (or how to make it all work together)

Table 1-4 Subjects of Interest to the PLC Programmer

While not each topic will be discussed in detail, the purpose of the text is to include as much about each of these topics as possible.

Some topics that are not included but should be of interest to the control engineer include:

1	Chemistry
2	Fluid Flow (sizing of valves, pumps, pipes)
3	Statics and Dynamics (sizing of servo systems, motors, etc)
4	Local area networks, wide area networks
5	Database development
6	Automatic Control – modeling using statistical methods and dif eq
7	Machine Design

Table 1-5 Subjects Not Included but of Interest to the PLC Programmer

Each of these areas may be of interest to the student but will not be discussed in detail in this text.

Summary

The emphasis is on programming with LAD and other languages in the most efficient manner to compete in the world marketplace. While hardware will not be emphasized, its use and inclusion in any text is necessary.

The following picture is from a Siemens school in Germany in 2008. These are instructors from around the world for PLCs. I was there but not in the picture.

Exercises

1. *A specification by this company was written that created the first PLC. Name the company and town (or area) in which this occurred:

2. *Name a PLC manufacturer that was absorbed by another PLC manufacturer.

3. *Name a PLC manufacturer that would be found predominantly in a Japanese auto company's manufacturing facility.

4. *Name a PLC manufacturer that would be found predominantly in a European factory.

5. *Name a PLC manufacturer that would be found predominantly in a US factory:

6. What is the significance of the picture below to the study of PLCs?

ENGINEERING COMMUNITY REMEMBERS PLC INVENTOR DICK MORLEY

> Morley, who died in October 2017, was revered as much for his ingenuity as for his modesty and the encouragement and mentorship he provided other engineers. REBECCA ZUMOFF — DECEMBER 19, 2017

Dick Morley, who passed away Oct. 17 at 84, was made of engineering legends. He said he created the first programmable logic controller (PLC) because he had a hangover on New Year's Day and he needed to save time. He dropped out of MIT because he didn't want to learn German. He also quit an engineering job because they wouldn't let him ski on weekdays when the lines were short. He rode Harleys, fostered dozens of children, and had a hand in creating the floppy disk, anti-lock brakes, and automated vehicle control. Morley, who helped found Modicon and Andover Controls Corp, which are both now part of Schneider Automation Inc., was also the author of several books and a regular contributor to industry magazines.

Morley was revered as much for his ingenuity as for his modesty and the encouragement and mentorship he provided other engineers.

"Dick Morley wasn't just the inventor of modern-day automation-he was also a mentor and a friend to many of our members and leaders around the world," said International Society of Automation President Steve Pflantz in a written statement. "He was a one-of-a-kind person, someone you could never forget. His humor and wit, along with his incredibly creative way of looking at life, made him a force for good in our industry, our society, and the world. He will be missed."

Morley was an angel investor for many young engineers and held annual "Geek Pride Day" festivals at his barn in New Hampshire, which he had converted to a workshop for engineering innovation. He was never comfortable being called the inventor of the PLC because he said others played a role in its creation.

"When we first met at a trade show many years ago, I praised Dick, the inventor of the PLC, for being the reason any integrators have a job at all. 'Oh, I didn't invent it,' I remember him responding. 'That was just an idea whose time had come, and I just happened to be the guy working on it,"' Rick Caldwell, president and founder of SCADAware wrote in an article after Morley's passing. Caldwell considered Morley a mentor, and said he came up with the name SCADAware.

"He never wanted fame, he only wanted to invent new things," wrote Walt Boyes, editor and publisher of the Industrial Automation and Process Control INSIDER. "You'd think that the man who invented the floppy disk, the handheld terminal, zone building HVAC, was the father of the PLC, and created the people mover for Detroit and Disney World, among the more than 100 patents he held, would be a household name, but Dick was a surprisingly private individual who didn't really want or enjoy credit for all that, and the limelight. So names like Bill Gates and Steve Jobs became famous, while Dick Morley just went on inventing."

Despite his accomplishments, Morley didn't die a wealthy man. He spent much of his savings on medical treatments for his son, who died of aplastic anemia at the age of 32, and for his wife of 56 years, who died in 2012. Friends and fellow engineers started a GoFundMe campaign to help him pay his medical bills a few years after his wife died. The campaign was also used to purchase him a modest headstone.

In Morley's honor, the ISA started the Richard E. "Dick" Morley Innovation Scholarship with a $50,000 endowment. The organization also pledged to match the next $50,000 in donations.

"We know that one of the most important parts of Morley's life was his work with young people," Pflantz said in the scholarship announcement. "He was passionate about giving kids a chance to innovate, to redefine their lives, and to make a difference in the world. We intend to make sure that this scholarship fund continues that legacy in some small way."

Chapter 1 INTRODUCTION

Introduction to PLCs

Programmable logic controllers (PLCs) were introduced to industry between 1968 and 1970 as a way to replace large expensive panels of relays, timers, and counters. Automotive manufacturers were looking for ways to simplify start-up of new car lines after model changeovers each year and save money in the cost of manufacturing installations.

Historically, relays have been used since the late 1800's to control simple processes. They were used in the early days to control railroad crossings. Before simple relay logic was introduced to control railroad crossing arms and alarm lights, accidents at these crossings were contributing to a high toll on human life.

The term “relay” was coined as the name of the device invented by Samuel F. B. Morse who invented the telegraph. The relay was invented as a device to extend the signal or “relay” the signal of the telegraph more than the 20 mile limit of electrical signals at the time of the invention of the telegraph (1836).

Relays, timers, and counters had been the favored choice for electrical and systems engineers to manufacturing facilities, especially in facilities with a large number of machines making discrete parts. Automotive manufacturers top the list of this type of manufacturing. At the same time that costs continued to rise for the engineering and construction of automotive assembly lines, computers were becoming more numerous and less costly. There was, however, a general discomfort among engineers to replace relays with computers. Most were reluctant to place the computer on the plant floor. A compromise was necessary for the engineer that he would be willing to accept. A computer that appeared to be relay-ladder logic to the electrician but able to use the computing capabilities of a computer was the device envisioned. The result of this vision is what is known today as the PLC (Programmable Logic Controller).

Relays as well as timers and counters were the first devices replaced by the PLC. Relays are electromechanical devices that use magnetism caused by power flow through the circuit's coil to energize a core and move a plunger with contacts attached. Contacts change state when the coil is magnetized. Normally open contacts close while normally closed contacts open. Changing contacts combine to complete other circuits. Combinations of relay contacts energizing coils form the basis of Boolean logic. Boolean logic deals with the combination of discrete on-off states to turn on or off other outputs.

The principle of using PLCs as substitutes for relays to reduce the wiring, panel fabrication, and engineering cost looked very appealing to the early PLC user. Like most electronic devices appearing in the early '70's, cost of the early PLC was high and functionality was not well developed. Early PLCs were developed around a mini-computer or special purpose control board. It was not unusual to pay $50,000 or more for a single PLC complete with I/O and still use relays for the most critical circuits. These hybrid designs existed for many years until engineers became convinced that PLCs were as reliable as the relay and could be trusted equally to the relay circuit being replaced. Hybrid circuits using both relays and PLCs existed for many years but have become too expensive for most automation applications in today's market.

In a quote from *Programmable Logic Controllers: An Emphasis on Design and Application* (Erickson, 2005) about the early PLC:

"The machine tool and automotive industries were large users of relay control systems. A simple machine tool would require six months to a year to completely debug (Morley, 2001). Every year, automotive manufacturing facilities would be shut down for two to three months in order to implement the changes due to the new automotive models. The lost production due to these changes was significant."

Richard (Dick) Morley was credited with the invention of the modern PLC. Morley worked for Bedford Associates in Massachusetts in 1968 when the PLC was first conceived. Dick Morley later wrote a book, *History of the PLC* in 2001 published by R. Morley Incorporated. Morley is commonly considered the father of the PLC.

Another quote from Morley in addressing the toughness of the PLC supplied by Bedford Associates, the Modicon controller from Morley's text and found also in Erickson's text:

"Landis [a machine tool company] decided to purchase the MODICON units and not use the PDP-14. When DIGITAL [*sic*] tried it get back into Landis, Landis wrapped a welder cable (operating) around the 084 and poured Coke over the unit. The 084 kept right on trucking. Digital retreated with grace."

Other anecdotal competitive stories abound and make up part of the folk lore of the early PLC. One about a PLC using 15 volt logic shows the competition between early PLC vendors. The vender owning the 15-volt PLC inserted a spark plug in a waveguide and entered a vendor's show with the noise gun in hand (carefully concealed in a briefcase). All of the PLCs were successfully shut down when the waveguide was pointed at their machine except the 15 volt model. This actually happened in the early race to be the best and the toughest PLC in industry. Toughness (in this case, noise immunity) was very important to the manufacturing engineer. What happened to the vendor of the 15-volt PLC? Today this vendor no longer is in the PLC market, displaced by the strong competition in the manufacturing marketplace.

There has been a great deal of interest about the origins of the PLC since those who were responsible for the origination of the PLC have retired and many have passed on. The competition for the selection of the first successful PLC is very interesting. It is repeated in the following article by Randall Brodzik, a writer for Control Engineering and found in the following 8/27/2014 article:

“**Inside the competition for the first PLC**

1971 The race to develop the first programmable logic controllers was underway inside General Motors’ Hydra-Matic Transmission Division in Ypsilanti, Michigan, in 1970. Three finalists had very different architectures.

In 1970, a fierce contest was being waged inside General Motors' Hydra-Matic Transmission Division in Ypsilanti, Michigan. At stake was program ownership of what would become the first programmable logic controllers (PLCs), and the outcome would shape not only the design of the first PLCs, but also the success or failure of their advocates.

PLC incubator

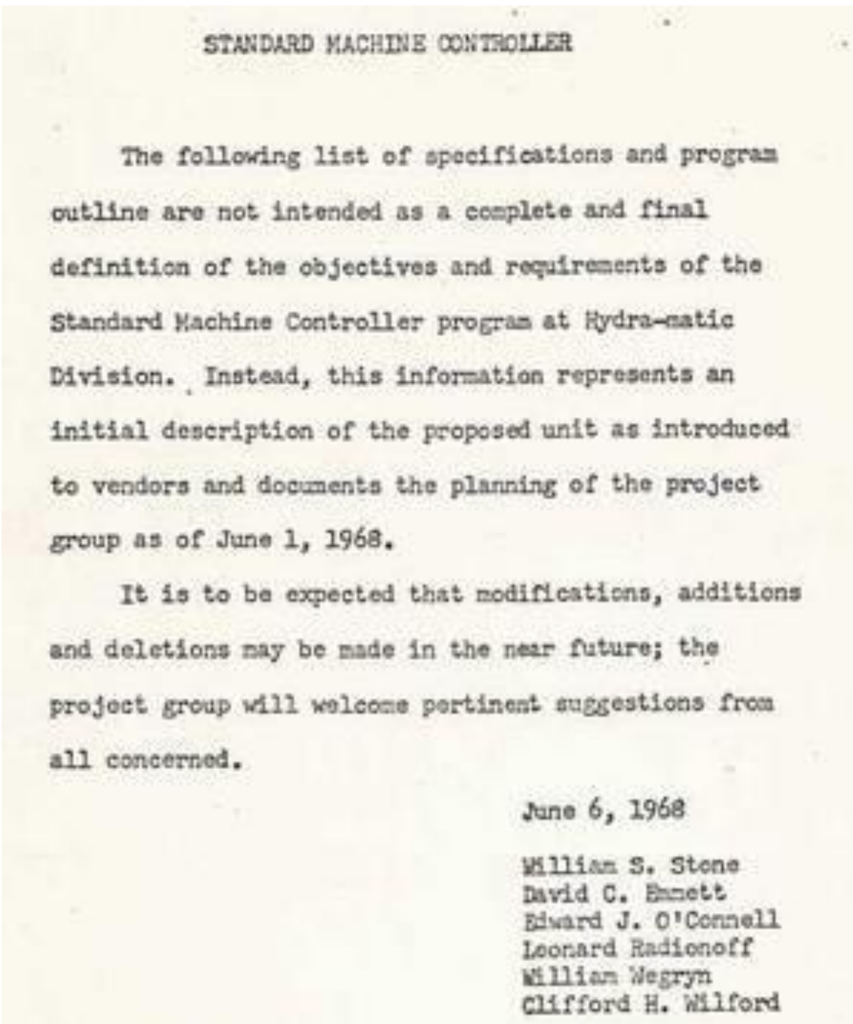
It was natural that GM's premier manufacturing technology group, Hydra-Matic, would be the incubator of the PLC. A nationally recognized technology leader, Hydra-Matic had installed one of the early IBM 1800 computers, the IBM 1801.

Richard Lundy, an owner of one of the original PLC suppliers, tells the story of when Ed O'Connell, a Computer Group engineer, didn't think GM was getting enough support from IBM. ‘An entire test line was running off of the one computer,’ Lundy said, ‘which made it very vulnerable, and Ed didn't like the support he was getting. So he called Watson.’ That's Thomas Watson, Jr., the son of IBM's founder and company CEO. Support for Hydra-Matic's IBM system quickly improved.”

In the following, Dave Emmett is described as the developer of the ‘Standard Machine Controller’.

"In April of 1968, a young Hydra-Matic engineer, Dave Emmett, proposed the development of what he called a 'Standard Machine Controller.' The controller he envisioned would replace the relay systems that controlled machine operations. Emmett was in charge of the Circuitry Group, and he envisioned a machine that would reduce maintenance costs, improve machine diagnostics, and decrease panel space.

Randy Brodzik, who later worked for Emmett, remembers that Emmett had a clear vision of what was needed: 'Dave said the goal was to develop a technology that would significantly reduce the time it took to make changes to a machine control sequence. With relays, first you had to do all the documentation, and then change all the wiring. It would take hours."



STANDARD MACHINE CONTROLLER

The following list of specifications and program outline are not intended as a complete and final definition of the objectives and requirements of the Standard Machine Controller program at Hydra-matic Division. Instead, this information represents an initial description of the proposed unit as introduced to vendors and documents the planning of the project group as of June 1, 1968.

It is to be expected that modifications, additions and deletions may be made in the near future; the project group will welcome pertinent suggestions from all concerned.

June 6, 1968

William S. Stone
David C. Emmett
Edward J. O'Connell
Leonard Radionoff
William Wegryn
Clifford H. Wilford

The letter at left was the original invitation for vendors to supply a system that would become the original PLC for GM.

Another group also was involved at GM. The description of this group follows:

"At about the same time, another group in Hydra-Matic was envisioning a different type of machine control system for GM. The Computer Division had hired Information Instruments, Inc. (3-I) to create a computer controlled assembly machine for the forward clutch line. This new control system had no limits to the number of elements in a ladder diagram, included parallel processing, and could incorporate complex Boolean equations.

Richard Lundy, a program manager for 3-I at the time, said the competition between the two Hydra-Matic Groups was ‘pretty dramatic.’ He described the Circuitry Group as wanting to duplicate existing ladder diagrams, while the Computer Group wanted to use nonsequential programming, similar to that used in current end-of-line test systems, to provide a more robust instruction set and reduce processing time. The advantages and limitations of each approach would become clear as the project progressed.”

Finally, a specification was written and shopped that would deliver a completed system. The RFP was a Request for Proposal. This form is described in the following:

“In April and May of 1968, work was done on a request for proposal that is remarkably simple by today's standards. The RFP, issued in June 1968, included only four pages of design specifications, including a requirement that ‘memory word length shall be at least eight bits.’ Douglas Brant worked on specifications for equipment coming into Hydra-Matic. He said that there was a strong rivalry between the Computer Group and the Circuitry Group in developing the specifications. ‘I guess it boiled down to people's territory,’ Brant said.

Of the many companies who received the RFP, three were selected for evaluation: Digital Equipment Corporation (DEC), 3-I, and Bedford Associates, a consulting firm. Its products were the DEC PDP-14, the 3-I PDQ-II, and Bedford Associates' Modicon 084. (The Modicon name came from MOdular DIgital CONtroller.)” See the following evaluation of the three finalists:

June 3, 1970

Design Considerations	DEC PDP-14	3-I PDQ-II	MODICON 084	Remarks
1. Input/output capability	256 inputs 256 outputs (power and storage)	256 inputs 254 outputs (power and storage)	256 ladder diagram rungs (no more than 4 inputs per rung)	PDQ-II addressing structure can handle up to 512 inputs. Requires rewiring of I/O box. DEC may have similar capability without rewiring.
2. Can outputs be used to store data during power on?	yes	yes	yes	
3. Can outputs be used to store data during power off?	Yes, if equipped with retentive feature	no	no	PDQ-II has no need for retentive memory because data may be stored in its core memory. See item 68 for retentive memory costs.
4. In addition to outputs used to store data, can memory store data?	no	yes	?	PDP-14 cannot because it has a read only memory.
5. Are inputs isolated?	no	yes	yes	Isolation is useful when multiple voltage sources are required or where sneak paths can occur. DEC inputs common in blocks of 8 inputs only.
6. Will basic power supply handle full system of 256 inputs and 256 outputs?	no	yes	-	If most systems require substantially fewer than 256 inputs/256 outputs, this may cause a PDQ-II cost disadvantage.
7. Are programmable timers available?	no	yes	yes	Programmable timers may be used over and over and cannot have their time changed unknowingly. They are awkward if time values for a particular operation must change often. The converse of both statements is true for analog timers. Analog timers also require additional wiring costs and mounting space.
8. Are adjustable analog timers available?	yes	not yet	no	
9. Operating temperature range	$0-55^{\circ}C$. at case of NEMA Enclosures	$0-60^{\circ}C$. Ambient	$0-60^{\circ}C$. Ambient	Actual successful operation in environmental chamber to over $65^{\circ}C$ has made it possible to raise the PDQ-II to $60^{\circ}C$.

-1-

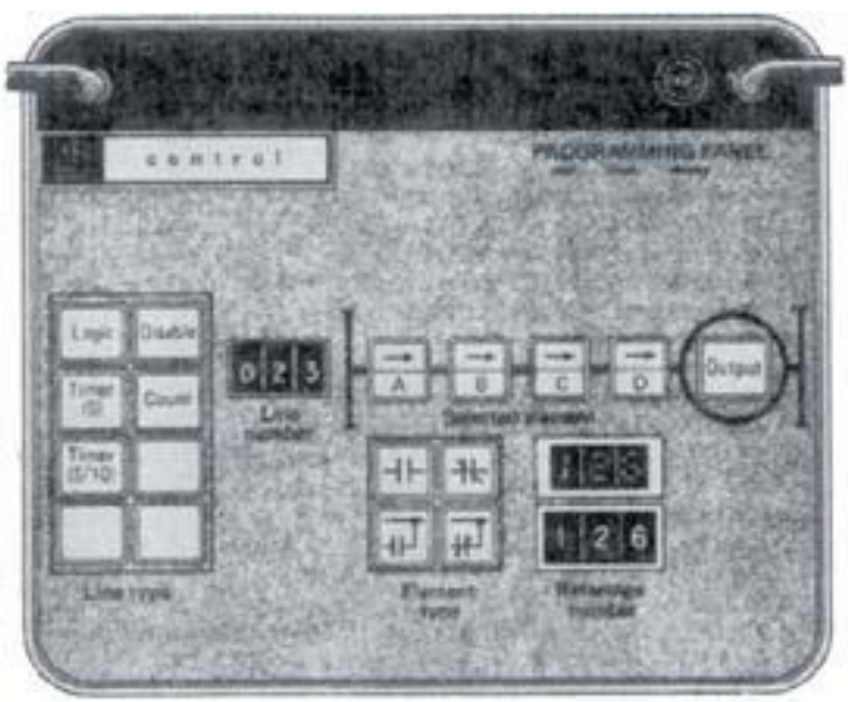
A PLC was installed at GM in June 1969. Its description follows:

“In June of 1969, DEC's PDP-14 was installed to control a gear grinder. Richard Lundy says that in spite of a very strong GM-DEC relationship, DEC's concerns about memory failure significantly limited the PDP-14's competitiveness. To make a change to the DEC program, ‘you had to send the client program in to them, and they would send you back a hardwired memory board,’ Lundy said. Having to remove a controller's memory from the plant to make changes would prove to make the DEC option unsustainable.

Richard Lundy's company, 3-I, was aligned with the Hydra-Matic Computer Group. Lundy said that during the evaluation period, the rivalry between the Computer Group and the Circuitry Group escalated. To the Computer Group, the advantages of higher level logic capabilities were clear and the selection of the PDQ-II was an obvious one. The 3-I solution for the forward clutch line, with its complex programming, had performed beautifully. Lundy said the PDQ-II quickly became known within the Computer Group as ‘pretty damn quick.’

Emmett and the Circuitry Group advocated strongly for the Modicon 084. The programming that was seen as a benefit by the Computer Group was seen as a detriment by Emmett's team. Because Modicon's programming language was similar to the Group's familiar relay ladder diagrams, it was expected to provide the smoothest transition and lowest cost for training and support. Randy Brodzik recalls that Emmett had an implementation view of the project: ‘He said that in order for this new technology to achieve wide acceptance, it needed to emulate what was already in place.’

As an added bonus for Modicon, the 084 was the only controller built into a hardened package, providing plant floor protection that the other two options did not. **And Modicon won**.”



This is the original Programming Panel for the Modicon 084. Yes, the numbers were thumb-wheel-switches which were very hard to turn and left fingers with callouses after entering a program. Note that there is no recording mechanism yet. That will come later.

Definitions of the PLC

NEMA's definition (NEMA or National Electrical Manufacturers' Association) of a programmable logic controller is "A digital electronic apparatus with a programmable memory for storing instructions to implement specific functions such as logic, sequencing, timing, counting, and arithmetic, to control machines and processes." This definition gives a good summation of the functionality of the modern PLC. A PLC must have each of these functional components to be considered a modern PLC. Primarily, the PLC is used to control machines. Programmable logic controllers have varied widely in what is considered a process or a machine. A few PLCs have even been used to control the duration of water for irrigation systems or lawn watering control. The view of what is a process has widened through the years. The modern PLC may even used to control the ingredients sprayed on a car moving through an automatic car wash.

A definition of the PLC from Liptak's *Process Control* states:

"A programmable logic controller (PLC) is an industrially hardened computer-based unit that performs discrete or continuous control functions in a variety of processing plant and factory environments."

These definitions of the PLC, either from NEMA or Liptak, give an overall picture of a computer with a special functionality partial to the automation market and able to control a variety of different machines. The purpose of this text is to more fully understand the use and programming of a PLC and understand applications of PLCs in factory automation.

In the US, the general trend was for the PLC to emulate or copy ladder logic as employed in the standard relay ladder logic or schematic drawing of the day. In other parts of the world, as the PLC was studied, this was not necessarily the case.

In Germany, for instance, STL or Statement List was introduced. From the Siemens book: *Milestones in Automation* by Arnold Zankl is the following quote (p. 54):

"Siemens had at first used STL programming exclusively and had been very successful with it. It seemed reasonable to program something in the way people think of it and describe it verbally. High education standards in Germany and Europe also supported this approach.

In the US, where training for skilled workers was generally less intensive than in other countries, the ladder diagram, derived from the circuit diagram, dominated from the start."

While this statement may not be meaningful to some, the choice of American PLC vendors to pursue Ladder logic while the German PLC vendor Siemens pursued another language, STL, was significant and has had an impact on the marketplace. While the German machine later embraced Ladder as a second language, the reverse has not occurred to any degree with the American market keeping strong ties to Ladder and not embracing other languages. Understanding a variety of languages and being able to program in the language most appropriate for the application is of primary importance and should be a student's goal.

Evolution of the PLC

From that beginning, PLCs have grown in popularity and capability to the extent that they can be found in some part of almost every industry. PLCs are produced by a variety of companies worldwide and several companies that made PLCs in the last 30 years have been forced out of the business by ever increasing competition.

Listed in Table 1-1 below is a time line of the evolution of the PLC from its inception to the modern PLC. See Table 1-1 for a portion of the time line for PLC development. Table 1-2 describes the Siemens PLC and its parallel development in the European market place.

"History of Programmable Logic Controllers (PLCs)

Table 6.5b from *Process Control* by Liptak

1968	Design of PLCs developed for General Motors Corporation to eliminate costly scrapping or assembly-line relays during model changeovers.
1969	First PLCs manufactured for automotive industry
1971	First application of PLCs outside the automotive industry
1973	Introduction of "smart" PLCs for arithmetic operations, printer control, data move, matrix operations, CRT interface, etc.
1975	Introduction of analog PID (proportional, integral, derivative) control, which made possible the accessing of thermocouples, pressure sensors, etc.
1976	First use of PLCs in hierarchical configurations as part of an integrated manufacturing system
1977	Introduction of very small PLCs based on microprocessor technology
1978	PLCs gain wide acceptance, sales approach $80 million
1979	Integration of plant operation through a PLC communication system
1980	Introduction of intelligent input and output modules to provide high-speed, accurate control in positioning applications
1981	Data highways enable users to interconnect many PLCs up to 15,000 feet from each other. More 16-bit PLCs become available. Color graphic CRTs are available from several suppliers
1982	Larger PLCs with up to 8192 I/O become available
1983	"Third party" peripherals, including graphic CRTs, operators' interfaces, "smart" I/P networks, panel displays, and documentation packages, become available from many sources

Table 1-1 Early History of the PLC

The PLC is an American invention but the European manufacture Siemens was quick to augment its automation offering to include its own PLC. The first PLC comparable to the American equivalent is the S3, Siemens' first storage programmable PLC. From *Milestones in Automation* by Arnold Zankl of Siemens is the following history of the Siemens history of automation control and the PLC in Europe.

1957	First prototype controllers from Siemens with germanium transistors
1964	Second generation control and switching systems introduced with board-design using Euro-card
1971	Third generation controllers introduced S1 - using hard-wired logic S2 - using electronic sequence processor

1973	S3 – first storage programmable PLC from Siemens (Simatic S30)
1976	First fault-tolerant and safety-related PLC for burner controls
1979	Simatic S5 introduced
1984	S5-135U multiprocessor PLC with up to 4 central processors for high-speed application
1986	Introduction of distributed I/O for Simatic
1988	Introduction of S5-155U with multiprocessor technology including floating-point arithmetic
1994	Introduction of Simatic S7
2000	Integrated drive systems
2002	TIA Totally Integrated Automation links data to IT domain

Table 1-2 History of the Siemens PLC

The advancement of PLC technology continues to the present day with newer PLCs introduced to meet the customers' demands. While early milestones of Table 1-1 show a general direction of the PLC toward a modular device capable of communicating to I/O, a programming terminal and to other PLCs, the PLC of today is also capable of communicating over a highway to a Human Machine Interface (HMI). PLCs are also capable of reading I/O from a variety of sources including I/O through other PLCs. I/O may be distributed using a number of different configurations. Some I/O may use a proprietary network while other I/O may use a public network or highway such as Ethernet. Connectivity to intelligent devices has increased through the years as well with PLCs interfacing to most types of manufacturing equipment such as bar code readers, weigh scales, servo motors and other intelligent devices.

At one time PLCs were simply known as the PC. With IBM and the introduction of the personal computer, naming standards had to change. PC was used for the personal computer so PLC manufacturers changed the name to Programmable Logic Controller. At some time during this early development of the PLC, Allen-Bradley trademarked the PLC name. The name PLC has been used by most engineers to describe the product known today as the programmable logic controller.

Liptak considers the reasons in his book for the popularity of PLCs. They are:

- "Ease of programming and reprogramming in the plant
- A programming language that is based on relay wiring symbols familiar to most plant electrical personnel
- High reliability and minimal maintenance
- Small physical size
- Ability to communicate with computer systems in the plant
- Moderate to low initial investment cost
- Rugged construction
- Modular design" (from *Process Control* by Liptak)

The programmable controller of today has grown in capability and shrunk in size from the first and second generation PLCs of the 1970's and 1980's. Where relays, timers, and counters were the concern of early design, inclusion of numbers and numeric manipulation quickly became part of the PLC. Instructions for comparing, adding, subtracting, multiplying and dividing of numbers were added to the language. As engineers demanded more sophistication from their PLC, designers turned to the microprocessor and, in part, emulated the instruction sets of the popular microprocessors of the day. Shift, Rotate, AND, OR, XOR and other instructions from the microprocessor were added. Finally added were floating point numbers and more sophisticated instructions to handle complex algorithms such as the FIFO and LIFO stack operation and the PID block.

The German manufacturer Siemens first introduced Statement List as its primary language for the PLC. Statement List or STL was introduced to incorporate the power of the microprocessor. This was accomplished by implementing a version of assembler language, a more difficult language to master but one that was more flexible than Ladder. In general, the German Siemens is considered more powerful but difficult to learn. Implementation, on the other hand, is usually simpler with the Siemens approach. As the text is used, hopefully these opinions will become more evident.

Step 7 is the programming language used to program the Siemens PLC. Newly introduced is Step 7 Basic for the new Siemens 1200. Later, the same language introduced for the 1200 will be introduced in the 300 and 400 lines of processors. STL will give way to SCL, a Pascal-type language and the assembler language will be dropped. While different languages will be used for different applications, Ladder or LAD will suffice for many early applications. Becoming familiar with LAD and its use will be a primary early concern. Later, use of Function Block and SCL will provide methods to better describe some applications. Also, the use of grouping of logic into function blocks will provide a method of encapsulation of logic similar to the programming language C's function, structure and class environment.

A look at hardware is important but will not be stressed. The use of language for development of control of a process will be stressed. Hardware changes with cost and function. Use of hardware for writing programs is necessary. Study of certain types of hardware instead of the programming will not be emphasized.

S7 refers to the newer lines of processors from Siemens. There is a S7-200 which is being phased out with the 1200's introduction and the S7-300/400 lines. The S7-200 has little in common with the 300/400 since it was developed by the TI engineers at the time that Siemens purchased the TI PLC division in the early 1990's. The earlier S5 processor also is being phased out. It is an older model, a contemporary to the Allen-Bradley PLC-2 and PLC-5 processor.

From an article on the Omron website:

"S5 is an old model, who was a contemporary of AB-2-PLC and PLC-5. S5 is the Intel 8031 series of microprocessors, which Siemens has licensed the production is based. They also own a coprocessor, which is used in some models S5 gear single Boolean logic. Siemens pressed overwhelming number of features from simple 8-bit processor. Punishment for this, however, that a statement S5 was very primitive and only slightly higher than the 8031 assembler. Look at the 8031 assembly and S5 Instruction List (IL), and you see the similarities. In contrast, AB has a different approach and use the mini-computer technology in their designs early PLC. Early

PLC2S used 4 processors AMD 2900 bit slice to a separate 16-bit processor. It was expensive in terms of hardware, but they had fewer restrictions in the software caused."

The use of PLCs gives one the reason why a language such as LAD is used instead of C for writing programs for manufacturing applications. It is simply quicker and cheaper to write. It is interesting to note that the PLC is perhaps changing its name again (from PC to PLC first) to PAC (Programmable Automation Controller. This new name may never catch on with some but for the newer embedded control processor of today, it may.

The Allen-Bradley PLC has endured from the beginning. After overtaking Modicon in the early 1980's, A-B never looked back at competition in the US (except for today's challenge from Siemens). The need for a fundamental knowledge of Allen-Bradley programming is required as well as Siemens. Today, both are required.

The PLC's Microprocessor Background

PLCs are closely related to computers because of their computational abilities and logic solving speed. PLCs receive inputs, solve logic, and set the appropriate outputs based on the inputs and programmed logic. Unlike computers, a program in one part of the PLCs memory cannot lock out a program in another part of memory. An errant program statement cannot cause all programs to stop execution, a very important concept if electricians and people other than the original programmer are to be allowed to change program statements. However, with advances of instruction sets to include program interrupt instructions, it is not always guaranteed that one part of a program will not interfere with another and even stop execution of the system.

The PLC can be broken down into these four basic components:

1) Central Processing Unit (CPU)
2) Memory
3) Inputs and Outputs (I/O)
4) Power Supply

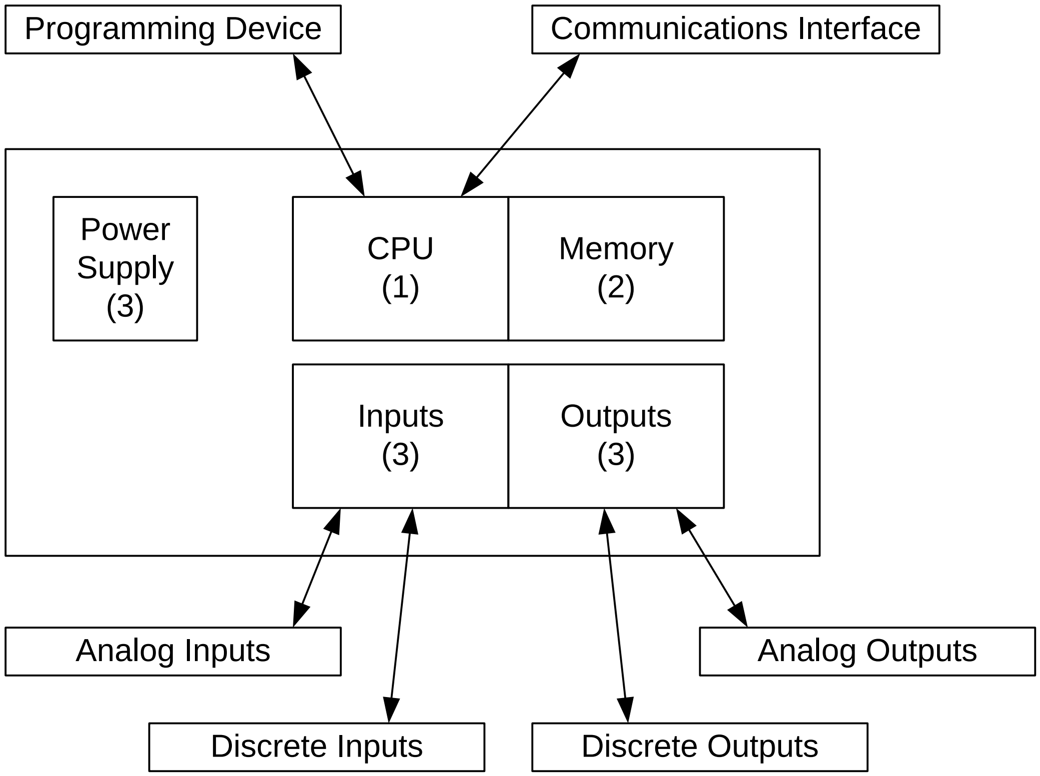


Fig. 1-1 Block Diagram of the PLC

Storage memory provides for variable storage and retrieval. Recipes and other table information reside here as well as the variables used in logic. Programming more sophisticated applications such as recipes hint at the using the PLC for more sophisticated applications. A basic rule for LAD logic has been that it must be maintainable by people other than professional computer programmers. This includes maintenance personnel who will be used to maintain the system once installed in a plant. As programs grow in complexity and length, ways have been sought to again make the logic simpler to implement and trouble-shoot. Although memory is used for a number of functions, the use of LAD only as a language in the PLC falls short in its ability to handle complex data storage and math arrays. Also, because of the complexity of larger programs, a change in one area can affect other areas, causing hardship when changes need to be made. Language selection is the most important aspect of application development.

Many “religious wars” were waged in the 1990’s involving the choice of the PC or PLC for automation. Those using the argument that the PLC would be out and that PCs would be found in all applications, while not discounted, were not considering the unique capabilities that PLCs had that the PC could not be counted on to deliver. The PLC could be counted on for their ruggedness and ability to restart even in the most difficult conditions. Distributed I/O was counted on to read on a consistent basis all the I/O in the system and report to the cpu. What PCs were capable to deliver was the openness and flexibility and very high performance levels. PLCs have always lagged their PC cousins in these areas but have not given ground based on their overall advantages that cannot be discounted such as ruggedness. From the Siemens text *Milestones in Automation* (pg 154) by Arnold Zankl, Zankl describes the choice of PC or PLC:

“Experts estimate that in 2005 over 90% of all automation solutions were implemented with programmable controllers. So, contrary to many forecasts, the PC has not replaced the standard PLC.”

The PC will not displace the PLC but which PLC emerges as the dominant platform or language is still to be determined.

PLC Hardware and Safety

Inputs are devices such as limit switches, push buttons, photo-eyes, proximity switches, and relays from other systems. Outputs are devices such as lights, relays, motor starters, solenoids. Motors and other outputs capable of producing motion must be controlled in such a way that the machine is run safely. The PLC must control the process safely from the moment the machine is turned on until the machine is turned off for the day or year. PLC programs must be written in such a way that all events of the machine are monitored in order to guarantee a safe and orderly control of the machine.

PLCs were not viewed as being as safe as relays when first developed. Engineers did not trust the computer as much as the simpler relay. This view changed as PLCs became more reliable. Interestingly, PLCs were made to resemble rugged equipment so engineers would accept them in the rugged industrial environment. Today, PLCs are manufactured that easily could fit in the palm of one's hand. These PLCs have the same error checking and hardware ruggedness as the big PLC of the 1970's. Ironically, these little PLCs have much more functionally with far superior instruction sets and faster scan times. They remind one of the evolutions of the hand-held calculator or the personal computer as the old box has given way to the more powerful yet less expensive new box.

The design of the functional aspects of the system must include all the electrical and programming aspects of the system as well as the mechanical, fluid power, air power and guarding systems.

The safety system runs in tandem with the production system. It must not impede the production system but run in parallel with it and oversee any hazards. The focus of the two systems is somewhat different. The production’s system focus is on throughput. The safety system is focused on protection.

A safe system is not entirely possible and a risk is always present. The purpose of the safety system is to reduce the risk to acceptable levels. **An absolutely safe system is not possible.**

How PLCs Solve Logic

Programming a PLC requires a program that solves the same basic logic again and again guarding against the unexpected and keeping the machine running in an acceptable manner. There must be an orderly flow to the logic to control the machine. PLCs all follow the same general format utilizing the following four steps:

1) initialize from a safe state - usually off
2) sense inputs
3) solve logic in the program
4) outputs turned on or off to mechanical devices

Steps 2, 3, and 4 are repeated again and again very rapidly to provide the orderly solving of logic and simulation of relays, timers, and counters. The process, while involving a number of complex actions, breaks down into the following:

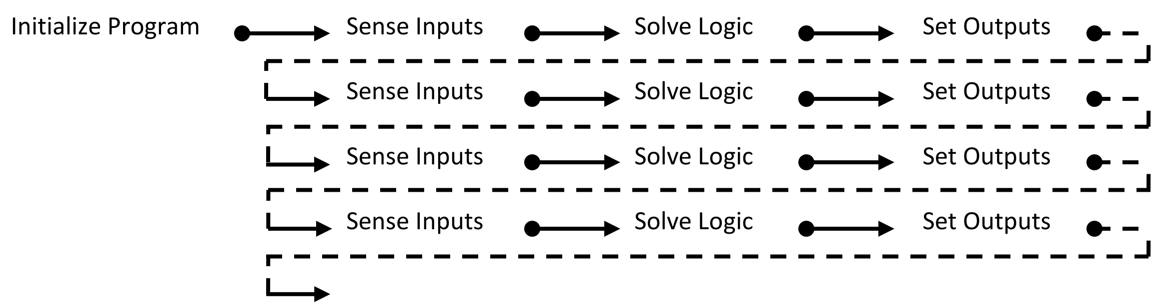


Fig. 1-2 Solving the PLC Program Scan by Scan

This process repeats each one to 5 millisecond and is referred to as the scan of the PLC. Each repeated sensing of inputs, solving of logic and setting of outputs is a scan of the PLC.

Siemens describes the cycle time of a processor as the entire time needed to read the inputs, execute the program one time and process the outputs. Their blocks are divided into networks and each network is divided into a number of statements. The time line is described as follows:

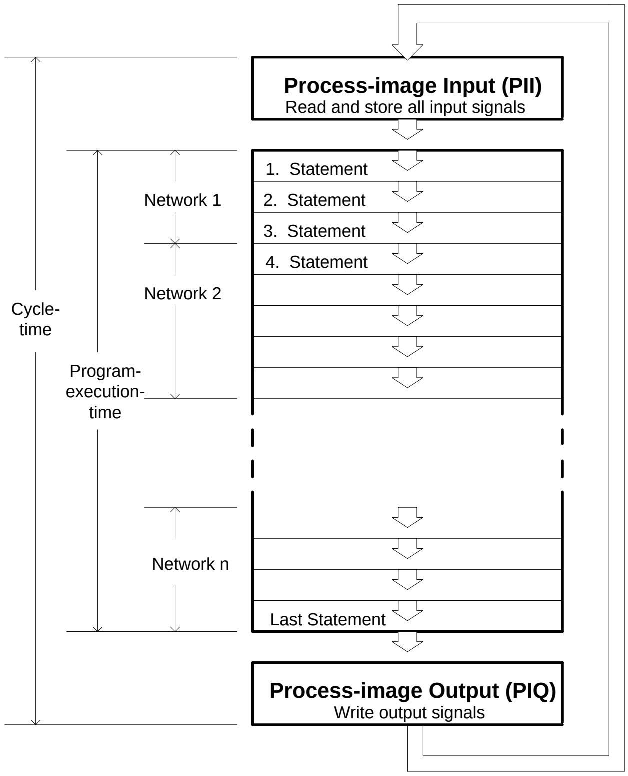


Fig. 1-3 Cycle Time of the Siemens PLC

There must also be an easy way to start and stop the PLC and its program from executing. This is referred to as the modes of *stop* and *run* (or *program* and *run*). A running PLC program is analogous to turning on the power to a relay panel. When power is turned on, the machine responds in an orderly way per the design of the engineer and the wiring of the electrician. Likewise, when the PLC is turned from *program* to *run*, the CPU begins executing the application based on the rules of the program. These rules form the basis for the operation of the system. First, outputs are defaulted to an off or pre-defined state. Then, as the inputs are read, the program is solved sequentially one logic statement at a time. Outputs are turned on or off based on the program and the input conditions. If a program is written and configured correctly, the outputs should fail to a safe state. Some outputs may be required to turn on if the processor fails. Most devices are designed so that if the controller fails, the output turns off and the device being controlled returns to a safe state. In other words, most but not all outputs turn off when the program returns to the *program* or non-running state. For safety's sake, design a system to return all outputs to the safe state, either on or off, when the program is not actively running a machine. And when the program is running a machine, devices are programmed to run in only a safe manner. This is a main concern of the programmer, to design a safe machine as well as a working machine.

Richard Morley describes in his own way the invention of the PLC through a number of youtube videos. While they are not the most watched videos on youtube, you may try them sometime. It is interesting to note that the philosophy of creating the PLC was for simplicity and the use of if-then statements. Really, all the statements are of this type – if-then.

PLCs in World Economy

Manufacturers of PLCs have been many and varied in the past with a stiffening competition over the last twenty years. The effect has been a thinning of the ranks of PLC vendors. It costs much more to bring products to market than it did a few years ago. Foreign competition has caught up and in many ways surpassed domestic PLC manufacturers' technology. A number of buy-outs, consolidations and joint operating agreements have thinned the number of PLC manufacturers to a few. Allen-Bradley is the mainstay American company producing PLCs. Also in the US are General Electric and the combined Modicon-Sq D PLC organization. Siemens in recent years has made significant inroads in the US market and overtaken all but Allen-Bradley. In Europe, the dominant PLC manufacturer is Siemens and in the Far East, Japan's Omron and Mitsubishi. The emerging manufacturing base in China favors Siemens although competition is very strong between many PLC manufacturers in China as well as in Mexico and Central and South America.

PLCs vary in size and type in a way similar to other manufactured products. Common to most manufacturers are the full size, compact, mini, micro and nano versions. Not surprisingly, the Japanese tend to dominate the mini, micro and nano end of the product while the German and Americans tend to dominate the larger models. This is changing, however, with a number of American models getting smaller and smaller as well as the Japanese pushing upward into the larger models.

Global PLC Markets Share – Historical Trends

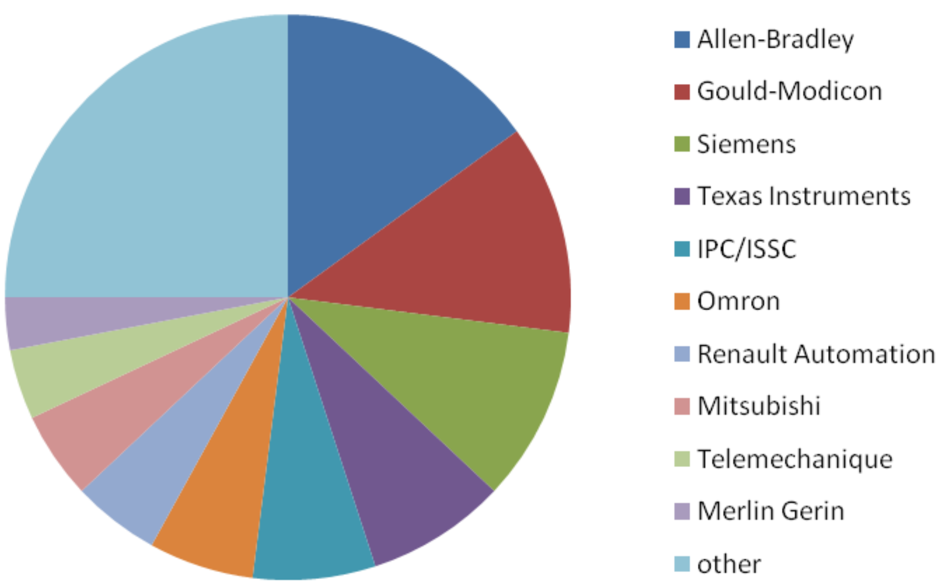


Fig. 1-4 Global PLC Market share in 1982/83

A view back to the early 1980's gives the major PLC vendor as "other" with the largest brand being Allen-Bradley. They were just ahead of Gould-Modicon who had dominated much of the earlier decade of the PLC market. Third was Siemens followed by Texas Instrument and IPC/ISSC. Four of the top 5 PLC vendors were from the US and the American PLC vendors tended to dominate.

By 1993, Siemens had overtaken Allen-Bradley with about 25% of the world market. Allen-Bradley was still first in the US but the market had globalized. Mitsubishi was third and Omron fourth. Modicon had been purchased by AEG and was still in fifth place. Texas Instruments was now part of Siemens as purchases of controls companies by other controls companies intensified. See Fig. 1-5 below:

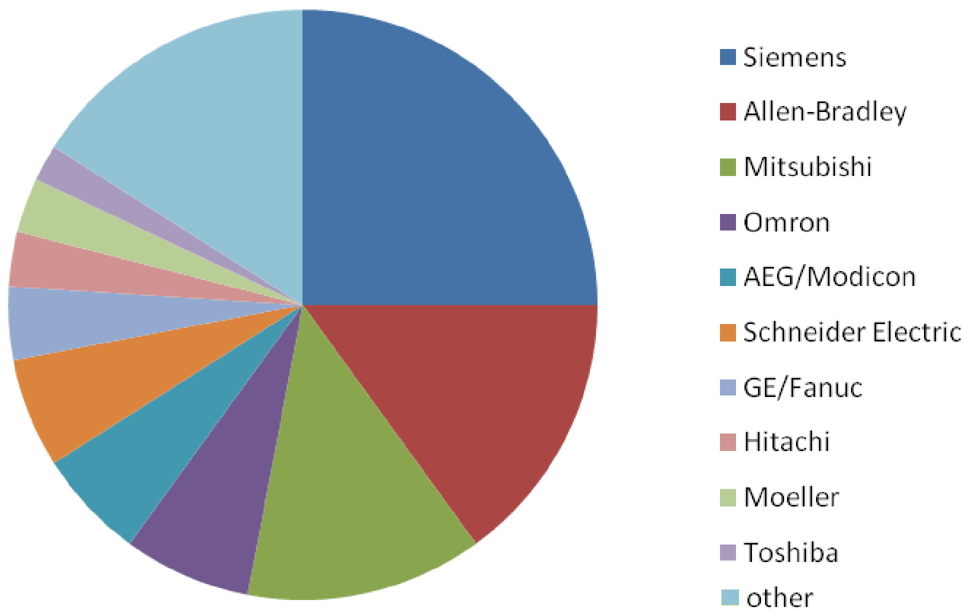


Fig. 1-5 Global PLC Market share in 1992/93

Since 1992, Siemens has continued its place as number one and widened its lead somewhat. Allen-Bradley continues in a strong second place. One company not known for its PLC entry appeared in the 2007/2008 market survey – ABB. ABB is a very strong controls company in the world and had been absent from earlier surveys but appears on this analysis although in a small position overall.

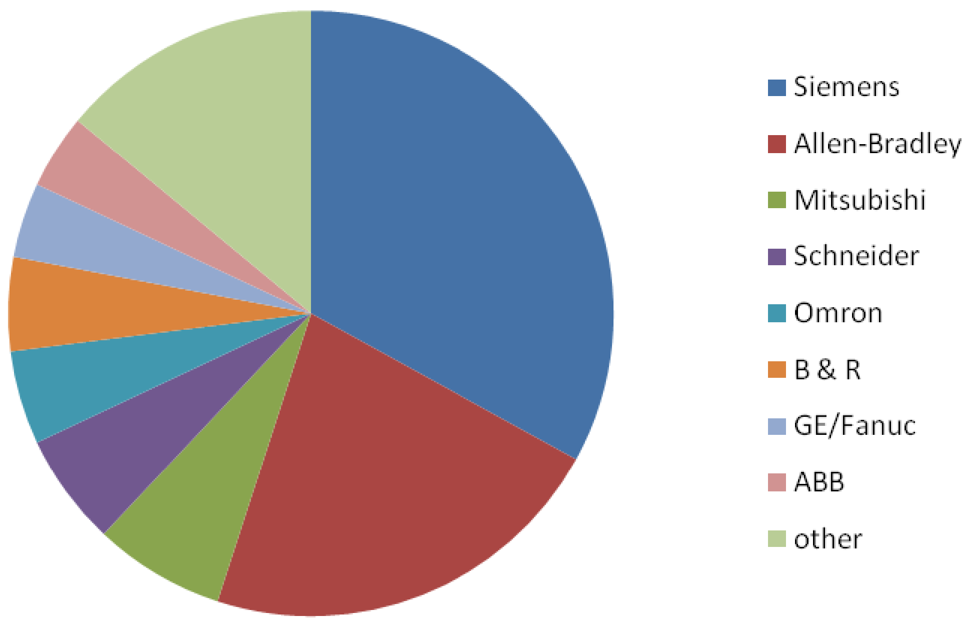


Fig. 1-6   Global PLC Market share in 2007/2008

Choice of a PLC must include Siemens as well as Allen-Bradley for the US market. Siemens' strength can be seen in its attention to detail and its global strategies. One should thoroughly study the languages found in the Siemens PLCs in order to write logic for the world market. STL (Statement List) should be learned as well as LAD (Ladder) and FBD (Function Block Diagram). It is not enough to insist only on LAD with an occasional subroutine written in FBD or other language.

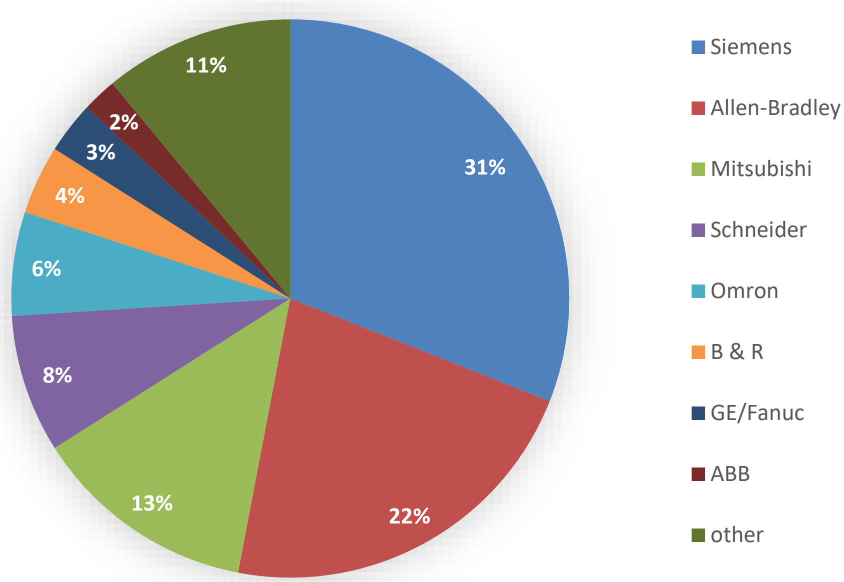


Fig. 1-7 Global PLC Market June 2018

A Long Quote from Siemens Milestones Text

From Choosing a PLC from *Milestones in Automation* by Arnold Zankl – Siemens: (pg 53)

"**Programming of the first programmable controllers**

To those involved in Europe and the US, it was clear from the start that, to be widely accepted, programmable controllers would also have to be easily programmable by skilled workers, electricians, installation engineers, and manual workers – not just by expert programmers. It was important to meet the target groups on familiar territory with regard to their existing level of technical know-how.

The plant electrician could, of course, read circuit diagrams. Engineers were somewhat familiar with Boolean algebra or could easily learn it. And young people who already know how to use progamming languages could quickly learn how to use menmonic instructions.

These basic programming languages had asserted themselves relatively quickly, and these are now defined in the global standard IEC 51131.1: Ladder diagram, function block diagram and statement list resp. instruction list."

The Siemens text continues on the next page (54) as follows:

"Siemens had at first used STL programming exclusively and been very successful with it. It seemed reasonable to program something in the way people think of it and describe it verbally. High education standards in Germany and Europe also supported this approach.

In the USA, where training for skilled workers was generally less intensive than in other countries, the ladder diagram, derived from the circuit diagram, dominated from the start."

To study the Siemens PLC in the US, one must recognize a change in attitude that the European worker has accepted for a much longer time – that programming must be flexible and written in the language best suited for the application. The American student must follow the rigors of the STL or SCL and FBD langauges as well as LAD in order to successfully compete in the marketplace today.

IEC 61131-3

IEC 61131-3 was intended to achieve the long-term aim of creating user software largely vendor-independent and being able to port it to devices of difference to system integrators who want to use different target systems. The chart below compares Allen-Bradley, Siemens and the IEC 61131-3 international PLC language:

Allen-Bradley RSLogix	Siemens Simatic	IEC 61131-3	
Ladder Relay Ladder	LAD Ladder Diagram	LD Ladder Diagram	Based on circuit diagram
FBD Function Block Diagram	FBD Function Block Diagram	FBD Function Block Diagram	Based on switching circuit systems
SFC Sequential Function Chart	S7-Graph for sequencers	SFC Sequential Function Chart	For sequential control
	S7-HiGraph State-transition diagrams		For asynchronous processes
	CFC Continuous Function Chart		In the form of technology oriented diagrams
	STL Statement List	IL Instruction List	Similar to assembler
ST Structured Text	S7-SCL Structured Control Language	ST Structured Text	Pascal-like high-level language

Table 1-3 PLC Languages

(The following article by Jeremy Pollard is re-printed with permission by the author. The article will explain what IEC 611131-3 is and what the benefits are. While this topic is beyond the scope of an introductory chapter, this seemed to be the only place to include it.)

IEC 61131-3 by the Numbers

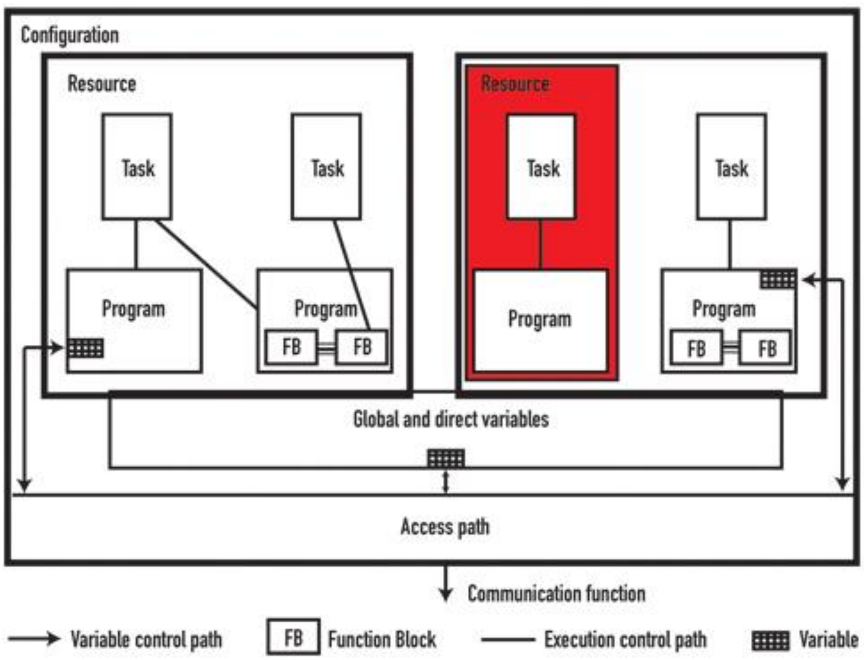
The Programming Standard for Controllers: What It Is, What It Is Not and Its Benefits

By Jeremy Pollard

The International Electrotechnical Commission's (www.IEC.ch) 61131-3 international standard "Programmable Controllers—Part 3: Programming Languages" was published in 1993 after 15 years in development and first used for PC-based control programming in the 1990s. Typically programmable logic controllers (PLCs) had their own vendor-developed programming platforms, but in the past decade, major PLC vendors from Europe and North America initiated and supported the new standard with their new programming platforms.

The intent of IEC 61131-3 is to normalize PLC and control systems' programming by standardizing functionality such as program entry, instruction visualization, data types and syntax. The general requirements section includes models for software, communication—external as well as internal instruction and variable parameter passing—and programming. The model functions are as follows:

- The software model introduces the following configuration concepts: resources such as CPUs; tasks such as executable application software; named variables used for storage; and communication paths (Figure 1). An IEC-based hardware "client" could run multiple tasks in one configuration or have multiple configurations.
- The communication model specifies how data is passed to different tasks or configurations or within the same task. Global variables are introduced.
- The programming model is tied closely with the concept of common elements, which enable the use of common data types, variable declaration and data formats such as dates and times. Program organizational units (POUs) also are common elements.



IEC 61131-3 Can Go Beyond

Figure 1: A conventional PLC's software (on the right, in red) consists of one resource, running one task and controlling one program. However, a controller using IEC 61131-3 (on the left) can go beyond PLCs by providing a multi-tasking, real-time environment that can be used in a broader range of applications without requiring users to learn added programming languages. If vendors allow it, this multi-tasking means a function block-controlled task could run along with a ladder logic task with data sharing and global variables. At the highest level, the entire software required to solve a particular control problem is called a Configuration. Within it, one or more resources can be defined. A resource is like a CPU in the system, and it includes one or more tasks. These control the execution of different parts of programs, called functions or function

blocks, which are the basic building blocks of data structures and algorithms. All of these can exchange their information via the global or direct variables, even to the outside world via communication functions such as OPC.

These POUs were developed to reduce the often-implied meanings of instructions and blocks. There are three types of POUs defined in the standard. The program POU is the program; no news there. The function-block POU is a program block with inputs and outputs used for tasks such as timers and counters. The function POU lets various program elements extend the instruction set of the configuration. The intent of these POUs is to reduce the programmatic differences between suppliers, so a timer is a timer, regardless of who provides it.

In addition, the standard defines data types, such as Boolean and integer. It also defines the use and format of derived data types and functions, which must conform to the previously mentioned standard data types.

Meanwhile, the programming model extends a PLC-type programming environment with three languages and two graphical representation languages. Ladder logic is most used in discrete applications and is the most widely coded language in automation today. It is mainly used and supported by the North American PLC vendors.

Instruction List, which is a rudimentary, assembler-type language has been a European programming staple for years. The third language is Statement List, which is similar to Pascal.

As a result, the graphical languages are Sequential Function Chart and Function Block. These are graphical representations of processes and have underlying code written in one of the three languages or in an alternate language, such as C++.

Finally, the standard defines the abstraction between the IEC model and the PLC/control hardware by using device tags and not addresses. Similar to programming in a high-level language such as Visual Basic, it allows the programmer to define a memory point, not by address, but by using a descriptive name. The standard defines the characters you can use in the description. The outside world is tied into the model based on the hardware used.

The desired effect of the standard is to reduce the user's learning curve between vendors, and with work beginning as early as 1978, it was developed to meet that need.

What IEC 61131-3 Is Not

The published IEC 61131 standard encourages extensibility, meaning any company that writes an IEC 61131-based product can add things to its "standard" product as long as it tells the user what changes and additions have been made relative to the standard. However, IEC 61131 isn't a standard. It is a specification for vendors that want to develop a control software programming environment according to a set of guidelines.

PLCopen (www.plcopen.org), the global association supporting IEC 61131, says you can't have a standard without certification. However, no one and no company can claim that its ladder logic editor creates compliant code. There's no way to certify it. And yet, one of Rockwell Automation's web pages for RSLogix 5 makes claims about the "RSLogix family of IEC-1131-compliant ladder logic programming packages."

In addition, even if it did create compliant code, Rockwell Automation would just need to say where the extensions are and what effect they have on that compliancy, and then they can claim compliance.

To make sure I wasn't being overly biased, I enlisted colleagues on the Automation List at www.Control.com. "The conformance criteria are so general, it is virtually meaningless," says independent consultant Michael Griffin about compliancy.

Also, a published PLCopen document states, "The overall requirements of IEC 61131-3 are not easy to fulfill. For that reason, the standard allows partial implementations in various aspects. This covers the number of supported languages, functions and function blocks. This leaves freedom at the supplier side, but a user should be well aware of it during his selection process." This doesn't help us very much.

The original intent of the specification or standard was to create a common platform for control software development. When the programmable controls industry emerged in the 1970s, all software was written using dedicated hardware and firmware. With the advent of the personal computer, vendors developed their own development environments. It is no different today. The software programs that have been developed support only the vendor's hardware. There is no common platform here. We are no further along than we were 20 years ago.

For instance, using RSLogix 5000 or Schneider Electric's Unity is no different than when we used fixed programming terminals. You can't share programs, although PLCopen is trying to develop an XML transfer specification.

I anticipate that vendors will have import routines, but few, if any, will have export routines.

Likewise, the specification does not cover off-the-scan issues and program-execution methods. Process behavior most likely will change on different hardware platforms, since the software environment is different.

Some of the visualizations of the code are supposed to be close. If you look at ControlLogix5000, which Rockwell Automation states is IEC 61131-compliant, and compare the visualization with that of KW Software, for example, there won't be much similarity.

Tag-based programming is a plus, but even in the 1980s, many products had the ability to use symbols. With IEC 61131, you have to use symbols, but the length of the tag names is variable. You can't use a 40-character symbol with a product that supports only 20.

To me, IEC 61131 is similar to SQL, UNIX or C. They're standard products to some degree, but they're not called standards. The varying flavors don't interoperate, because, before the world went Microsoft, some compilers had very different syntax and ways of doing things.

In fact, Julien Chouinard, managing director of ICS Triplex–ISaGRAF, calls IEC 61131 a compromise. However, while I think IEC 61131 has fallen short of expectations, this doesn't mean there are no benefits to the specification.

Benefits of IEC 61131-3

Now, make no mistake about IEC 61131-3. We're still in the same environment we were in before its creation. Rockwell Automation, Schneider Electric, Siemens Industry, GEIP and Wago all have their own programming environments. The business of automation won't allow for real interoperability of these competing products.

The main difference from 20 years ago is that there is some form of commonality. There are some good reasons to use an IEC 61131-based product. Tag names permit abstraction of the hardware I/O, so databases from product to product could be maintained externally. All will use tag names-based variable allocation and typically the common elements, such as Boolean, will behave the same. That doesn't mean, however, that all vendors support all the common elements.

Most products will have all five languages. A large benefit of an IEC 61131-integrated development environment (IDE) is access to the function block part of the software. The power in function block programming is tremendous. This is the type of programming the older DCS products had 20 years ago. You can program a very complex part of the application and have it hidden behind a block. Some argue that this improves troubleshooting capabilities.

In return, it requires a better programming skill, as well as a good handle on the project scope and project management. This is foreign territory to many ladder programmers.

A selling feature of the standard is the ability to develop control strategies in the right language. While I agree with this, the plant floor maintenance staff usually doesn't include a structured text guru. This is scary for the controls engineer who develops the application, so oftentimes he'll just use ladder logic. Therefore, one of the major benefits of function-block programming is left untapped as a result.

Proponents of IEC 61131 swear by its training and investment benefits. They say if you use Company A's IEC 61131, and later you have to use Company B's IEC 61131, then you already know how to program the hardware. This is not necessarily so, but there is some truth to it.

Many companies use a development package from 3S, some use a product from Softing, and others create their own. If an OEM used IEC software from four different hardware vendors, and they all licensed their software from one company, the OEM stands a better chance of being able to migrate the learning curve to the new hardware and of being able to reuse the software.

Will the training cycle be shorter? Probably. But from vendor to vendor, there is some level of commonality that can be borrowed, but the impact may be limited.

Will IEC 61131 help the maintenance person trying to solve a problem at 3:00 a.m.? For most user companies that standardize on a single hardware platform, the answer is "no." They'll get used to whatever the software tool is. IEC 61131 isn't important.

It might be important to you if you leave your company and have IEC 61131 on your resume. It surely can't hurt.

If you're a machine builder or a manufacturer with multiple vendors that all use IEC 61131, maybe it helps. There are some inherent similarities that could help getting refamiliarized with a hardware platform or a software troubleshooting tool, but I think this is a personal function rather than a software tool function.

No doubt IEC 61131 is here to stay. Early misunderstandings of the benefits, and resulting misconceptions of what an IEC 61131-based product actually was turned off many potential users.

Open is as open does. IEC 61131 is the same. It is simply a platform for a software product design tool for hardware. It provides some benefits or the OEM and the user, but mainly the vendors benefit, in my perception.

It is no panacea. It can help in our quest for a better automation platform, but we will leave that for IEC 61499, a true standard. As long as we don't screw it up.

Jeremy Pollard has been writing about technology and software issues for many years. He is publisher of "The Software User Online," has been involved in control system programming and training for more than 25 years and was previously North American managing director of PLCopen (www.plcopen.org), which drafted IEC 61131.

Some Key Features of IEC 61131-3

Structured Software – through use of configuration, resource and function blocks
Strong Data Typing – by using languages that restrict operations so they only apply to a appropriate data types
Execution Control – through use of tasks
Complex Sequential Behavior – through sequential function charts
Software Encapsulation – through use of function blocks, structures and complex data types

The standard allows for the creation of programs that are highly structured, yet flexible. It provides tools for creating programs with complex sequences and promotes the creation of reusable modules or function blocks.

(end of article by Jeremy Pollard)

Other Topics Close to PLC's

Topics of interest in a PLC text in addition to programming languages should include:

1	Distributed I/O
2	Fabrication of a system including the cabinet, lay-out, drawings
3	Fault tolerant/safe systems
4	RS485 networks including Profibus and DeviceNet, simple protocols and simple ASCII
5	Use of HMI (Human Machine interface)
6	Ethernet networks, wireless
7	Relation between PLC and CNC and PCS (DCS), robotics
8	Sensors and actuators
9	Batch Systems
10	MES (Manufacturing Execution System) to include a. Business Modeling Factory b. Enterprise Asset Management c. Plant Maintenance Management d. Dispatch e. Trace and Tracking
11	ERP(Enterprise Resource Planning) (or how to make it all work together)

Table 1-4   Subjects of Interest to the PLC Programmer

While not each topic will be discussed in detail, the purpose of the text is to include as much about each of these topics as possible.

Some topics that are not included but should be of interest to the control engineer include:

1	Chemistry
2	Fluid Flow (sizing of valves, pumps, pipes)
3	Statics and Dynamics (sizing of servo systems, motors, etc)
4	Local area networks, wide area networks
5	Database development
6	Automatic Control – modeling using statistical methods and dif eq
7	Machine Design

Table 1-5   Subjects Not Included but of Interest to the PLC Programmer

Each of these areas may be of interest to the student but will not be discussed in detail in this text.

Summary

The emphasis is on programming with LAD and other languages in the most efficient manner to compete in the world marketplace. While hardware will not be emphasized, its use and inclusion in any text is necessary.

The following picture is from a Siemens school in Germany in 2008. These are instructors from around the world for PLCs. I was there but not in the picture.

Exercises

1. *A specification by this company was written that created the first PLC. Name the company and town (or area) in which this occurred:

2. *Name a PLC manufacturer that was absorbed by another PLC manufacturer.

3. *Name a PLC manufacturer that would be found predominantly in a Japanese auto company's manufacturing facility.

4. *Name a PLC manufacturer that would be found predominantly in a European factory.

5. *Name a PLC manufacturer that would be found predominantly in a US factory:

6. What is the significance of the picture below to the study of PLCs?

ENGINEERING COMMUNITY REMEMBERS PLC INVENTOR DICK MORLEY

> Morley, who died in October 2017, was revered as much for his ingenuity as for his modesty and the encouragement and mentorship he provided other engineers. REBECCA ZUMOFF — DECEMBER 19, 2017

Dick Morley, who passed away Oct. 17 at 84, was made of engineering legends. He said he created the first programmable logic controller (PLC) because he had a hangover on New Year's Day and he needed to save time. He dropped out of MIT because he didn't want to learn German. He also quit an engineering job because they wouldn't let him ski on weekdays when the lines were short. He rode Harleys, fostered dozens of children, and had a hand in creating the floppy disk, anti-lock brakes, and automated vehicle control. Morley, who helped found Modicon and Andover Controls Corp, which are both now part of Schneider Automation Inc., was also the author of several books and a regular contributor to industry magazines.

Morley was revered as much for his ingenuity as for his modesty and the encouragement and mentorship he provided other engineers.

"Dick Morley wasn't just the inventor of modern-day automation-he was also a mentor and a friend to many of our members and leaders around the world," said International Society of Automation President Steve Pflantz in a written statement. "He was a one-of-a-kind person, someone you could never forget. His humor and wit, along with his incredibly creative way of looking at life, made him a force for good in our industry, our society, and the world. He will be missed."

Morley was an angel investor for many young engineers and held annual "Geek Pride Day" festivals at his barn in New Hampshire, which he had converted to a workshop for engineering innovation. He was never comfortable being called the inventor of the PLC because he said others played a role in its creation.

"When we first met at a trade show many years ago, I praised Dick, the inventor of the PLC, for being the reason any integrators have a job at all. 'Oh, I didn't invent it,' I remember him responding. 'That was just an idea whose time had come, and I just happened to be the guy working on it,"' Rick Caldwell, president and founder of SCADAware wrote in an article after Morley's passing. Caldwell considered Morley a mentor, and said he came up with the name SCADAware.

"He never wanted fame, he only wanted to invent new things," wrote Walt Boyes, editor and publisher of the Industrial Automation and Process Control INSIDER. "You'd think that the man who invented the floppy disk, the handheld terminal, zone building HVAC, was the father of the PLC, and created the people mover for Detroit and Disney World, among the more than 100 patents he held, would be a household name, but Dick was a surprisingly private individual who didn't really want or enjoy credit for all that, and the limelight. So names like Bill Gates and Steve Jobs became famous, while Dick Morley just went on inventing."

Despite his accomplishments, Morley didn't die a wealthy man. He spent much of his savings on medical treatments for his son, who died of aplastic anemia at the age of 32, and for his wife of 56 years, who died in 2012. Friends and fellow engineers started a GoFundMe campaign to help him pay his medical bills a few years after his wife died. The campaign was also used to purchase him a modest headstone.

In Morley's honor, the ISA started the Richard E. "Dick" Morley Innovation Scholarship with a $50,000 endowment. The organization also pledged to match the next $50,000 in donations.

"We know that one of the most important parts of Morley's life was his work with young people," Pflantz said in the scholarship announcement. "He was passionate about giving kids a chance to innovate, to redefine their lives, and to make a difference in the world. We intend to make sure that this scholarship fund continues that legacy in some small way."