Chapter Two
Programmable Logic Controller (PLCs)

2.1 Introduction

In everyday operations or industrial processes, we come across situations where there is a need to control some device or a physical quantity such as time, temperature, sound, light and so on, to get the required result or output. For example, do you think an airplane would be useful to a pilot, if he cannot make it go where he wants it to go or would an air-conditioner be useful, if the temperature in a room cannot be controlled in both the examples; there is a need to control a process. A control system is a system that can sense, switch and/or control an operation. It operates on an input signal and controls the process in order to provide an output signal. This is shown in the block diagram in Figure 2.1 [2].

Figure 2.1 Elements of a control system

- **Types of controllers**

Different types of controllers could be used based on the requirements of the application. Some examples are included below:

1. Relays and Contactors

Simple electromechanical devices like relays and contactors are most widely used for controlling a discrete manufacturing process.

2. Microcontrollers

Microcontroller is a special purpose computer that can do one job, for example, the one that is used in an automatic washing machine and in a microwave.

3. Programmable Logic Controllers

A Programmable Logic Controller (or PLC) is a specialized digital controller that can control machines and processes. It monitors inputs, makes decisions, and controls outputs in order to automate machines and processes. It uses programmable memory to store instructions and specific functions that include on/off control, timing , counting, sequencing, arithmetic, and data handling [3]. The first PLC systems evolved from conventional computers in the late 1960s and early 1970s, these first PLCs were installed primarily in automotive plants.

Traditionally, the auto plants had to be shut down for up to a month at model changeover time. The early PLCs were used along with other new automation techniques to shorten the changeover time. The hardwired planes were very time consuming to wire, debug and change. But the GM identifier the following requirements for computer controllers to replace hardwired planes:

- 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 PLC keyboard reprogramming procedure replaced the rewiring of a panel full of wires, relays, timers, and other components. The new PLCs helped reduce changeover time to a matter of a few days.

2.1.1 PLCs versus relay control

For years, the question many engineers, plant managers, and Original Equipment Manufacturers (OEMs) asked was, Should i be using a programmable controller. Even today, many control system designers still think that they are faced with this decision. One thing, however, is certain

today's demand for high quality and productivity can hardly be fulfilled economically without electronic control equipment. With rapid technology developments and increasing competition, the cost of programmable controls has been driven down to the point where a PLC-versus-relay cost study is no longer necessary or valid. Programmable controller applications can now be evaluated on their own merits. If system requirements call for flexibility or future growth, a programmable controller brings returns that outweigh any initial cost advantage of a relay control system. Even in a case where no flexibility or future expansion is required, a large system can benefit tremendously from the troubleshooting and maintenance aids provided by a PLC. The extremely short cycle (scan) time of a PLC allows the productivity of machines that were previously under electromechanical control to increase considerably. Also, although relay control may cost less initially, this advantage is lost if production downtime due to failures is high.

2.1.2 PLC versus Microcontroller

Usually PLCs are used in an industrial environment, where as the microcontrollers are smaller and well suited for embedded situations. PLCs are programmed with readymade blocks or programming elements, whereas in Microcontrollers a programming language must be used to write a programming code.

2.1.3 PLCS versus computer control

The architecture of a PLC's CPU is basically the same as that of a general purpose computer; however, some important characteristics set them apart. First, unlike computers, PLCs are specifically designed to survive the harsh conditions of the industrial environment. A well-designed PLC can be placed in an area with substantial amounts of electrical noise, electromagnetic interference, mechanical vibration, and no condensing humidity. A second distinction of PLCs is that their hardware and software are designed for easy use by plant electricians and technicians. The hardware interfaces for connecting field devices are actually part of the PLC itself and are easily

connected. The modular and self-diagnosing interface circuits are able to pinpoint malfunctions and, moreover, are easily removed and replaced.

Also, the software programming uses conventional relay ladder symbols, or other easily learned languages, which are familiar to plant personnel [1].

2.1.4 Advantage of PLC Control Systems

- Fixable.

In the past electronic controlled production machine require it, own controller, that is to say fifteen machines may require fifteen controller. Now it is possible to use just one model of PLC to run any one of the fifteen machines.

- Faster response time.
- Less and simpler wiring.

When a PLC program circuit or sequence design change is made, the PLC program can be changed from a keyboard sequence in a matter of minutes. No rewiring is required for a PLC-controlled system.

- Solid- state- no moving parts.
- Modular design - easy to repair expand.
- Handles much more complicated systems.
- Communication capability.

A PLC can be communicated with other controllers or computer equipment.

- Sophisticated instruction sets available.

The PLC programming can be accomplished in the ladder mode by an engineer, electrician or possibly a technician.

Alternatively, a PLC programmer who works in digital or Boolean control systems can also easily perform PLC programming.

- Less expensive.

Increased technology makes it possible to conclusion more function into smaller and less expensive packages. Now you can purchase a PLC with

numerous relays, timers, counters and other function for a few hundred dollars.

- Speed of operation.

Relays can take an unacceptable amount at time to actuate; the operation of speed of PLC is very fast.

- Reprogramability and correction error.

It can be programmed, controlled and operated by a person Unskilled in operating (programming) computers.

A PLC's operator draws the lines and devices of ladder diagrams with a keyboard/mouse onto a display screen. The resulting ladder diagram is converted into computer machine language and run a program. Also, if a programming error has to be corrected in a PLC control ladder diagram, a change can be typed in quickly, that means when a PLC program circuit a sequence design change is mode, the PLC program can be change from keyboard.

- Pilot Running.

A PLC programmed circuit can be pre run and evaluate in a Lab. The program can be typed in tested, observed and modified if needed, saving valuable factory time [3].

2.1.5 Disadvantages of PLC Control System

- Cost of PLC is high.
- PLCs are designed by semiconductors, which are depends on thermal characteristics.
- You need a programmer (know-how) skill.
- Debugging the PLC sometimes consumes a lot of time.

2.1.6 PLC application

The PLC has been successfully applied in virtually every segment of industry, including steel mills, paper plants, food-processing plants, chemical plants, and power plants. PLCs perform a great variety of control tasks, from

repetitive ON/OFF control of simple machines to sophisticated manufacturing and process control.

2.2 PLC Architecture

Figure 2.2 and Figure2.3 show the PLC architecture and major component from it. It is consisting of central processor, memory, input/output and all buses.

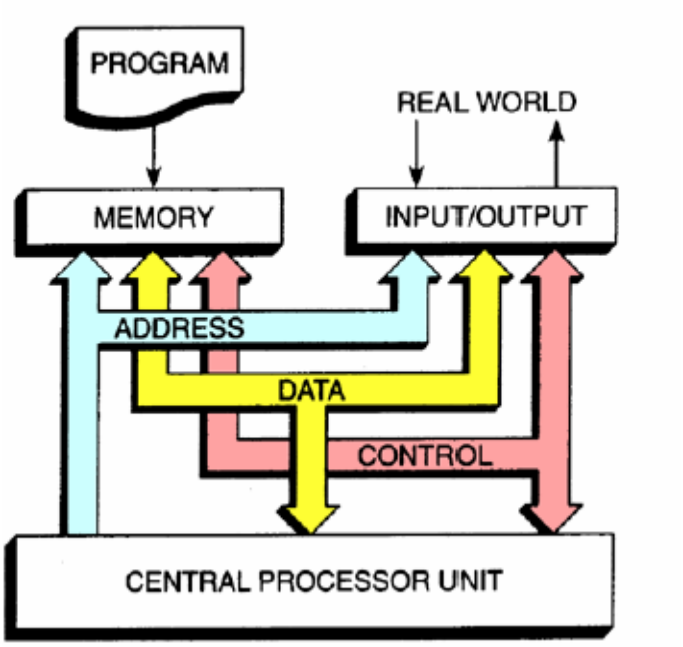


Figure 2.2: PLC architecture

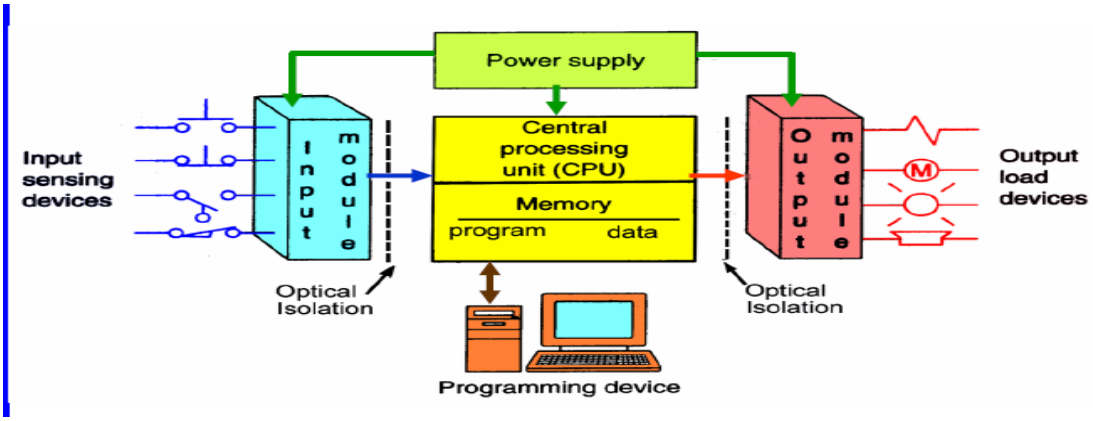


Figure 2.3: Major components of a common PLC

An open architecture design allows the system to be connected easily to devices and programs made by other manufactures. A PLC is a computer, but different type from one you need. In small PLCs, the processor, solid state memory, I/O module and power supply are board in a single compact unit.

The programming device with keyboard and liquid crystal display (LCD) is separate. In larger PLC, the processor and memory are in one unit, the power supply in a second unit and I/O interface in additional unit. In Figure 2.3.b we see the fixed memory contains the program set by the manufacture. This system program which has the same function as a DOS program in PC is set into special IC called ROM. PLC may have also other type of solid-state memory such as PROM, EPROM and EEPROM.

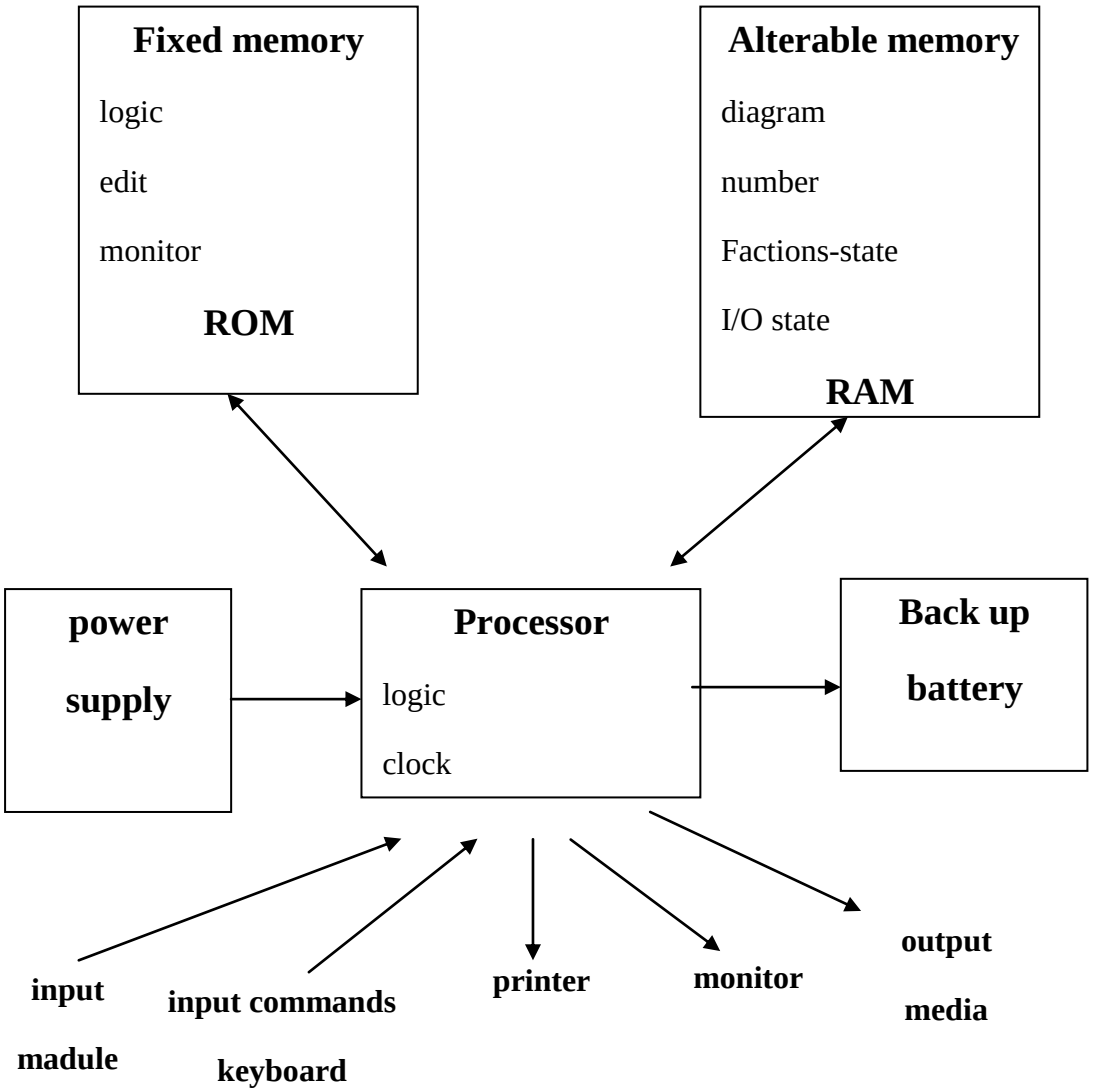


Figure 2.4: Inside the PLC

2.2.1 Power supply

Provides the voltage needed to run the primary PLC components.

2.2.2 Inputs and outputs modules

Provides signal conversion and isolation between the internal logic-level signals inside the PLC and the field's high level signal. The I/O interface section of a PLC connects it to external field devices. The main purpose of the I/O interface is to condition the various signals received from or sent to the external input and output devices. Input modules converts signals from discrete or analog input devices to logic levels acceptable to PLC's processor. Output modules converts signal from the processor to levels capable of driving the connected discrete or analog output devices [3].

- Output devices: are switched on by the PLC. This can be anything electrical such as the following:

- Direct Current (D.C) motor (e.g. to start a conveyor belt)
- Alternate Current (A.C) motor (e.g. to start a pump)
- linear electric actuator
- solenoid valve in hydraulic system
- solenoid valve in plant system (e.g. to open a pipe line valve or allow steam into a heat)
- lights (e.g. traffic lights)
- alarms (e.g. fire alarm or oil level alarm)
- heating elements (e.g. heater in a hydraulic tank)

2.2.3 Processor

Provides intelligence to command and govern the activities of the entire PLC systems. The size of Central Processor Unit (CPU) depends on the size of process to be controlled. It is important to size the system CPU accords to internal memory needed to run the process. Some CPUs can have additional memory easily added at a late date; others cannot be added to an expanded.

Many CPUs contain backup battery that keeps the operating program in storage in the event of power failure. The basic operating program is permanently stored in CPU and is not lost when input power is lost. The

process ladder program is not permanently stored. Back up battery enables the CPU to return the operating ladder program if power is lost.

2.2.4 Programming Device

Used to enter the desired program that will determine the sequence of operation and control of process equipment or driven machine. The types of programming devices are:

- Hand held unit with Light Emitter Diode (LED) / LCD display.
- Desktop type with a Cathode Ray Time (CRT) display.
- Compatible computer terminal.

2.2.5 Memory system

The memory system in the processor module has two parts: a system memory and an application memory shown in Figure 2.5 [3].

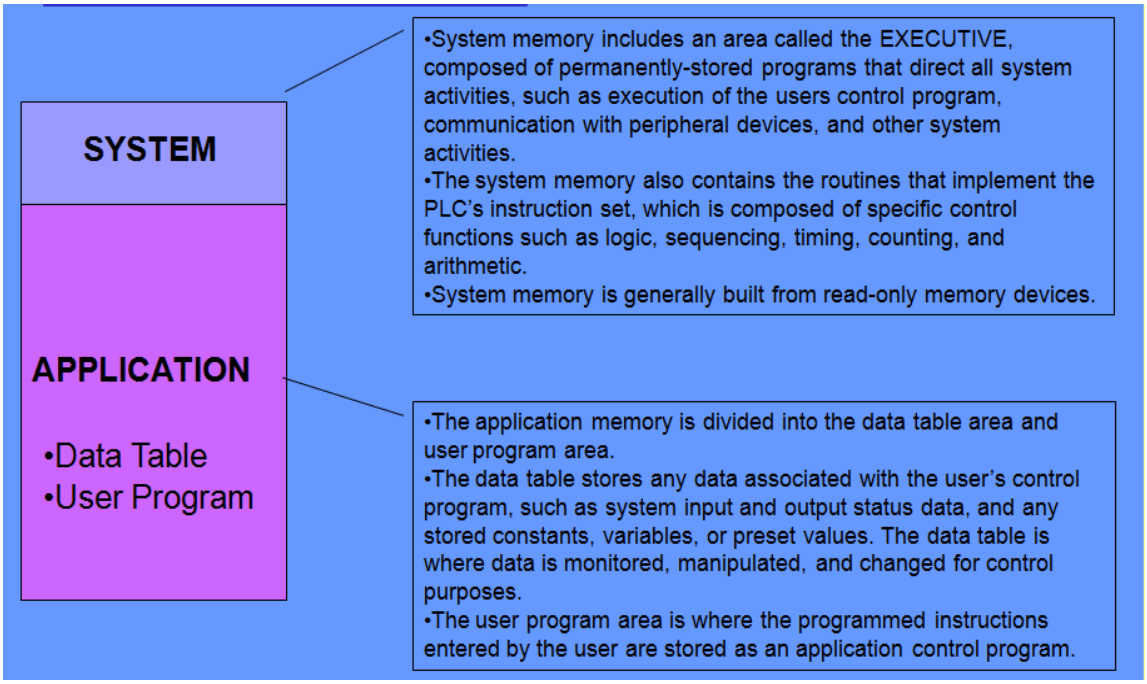


Figure 2.5: Memory map organizations

2.3 Operating systems and application programs

A PLC contains a basic operating system that allows for:

- Downloading and executing user (ladder logic) programs.
- Communicating with devices: I/O modules and Other PLCs on a network.
- Holding configuration data such as:

Number and type of I/O modules present in the PLC system and Status information [4].

2.3.1 User program execution

A PLC executes an initialization step when placed in run mode, and then repeatedly executes a scan cycle sequence.

The basic PLC scan cycle consists of three steps as follows:

- Input scan

During the input scan, data is taken from all input modules in the system and placed into an area of PLC memory referred to as the input image area shown in Figure 2.6.

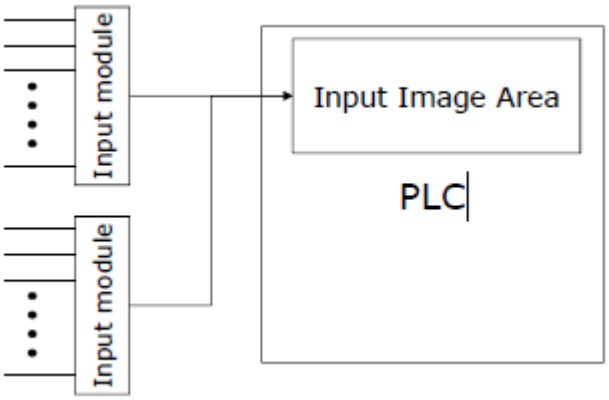


Figure 2.6: Input scan

During the input scan, input terminals are read and the input image area is updated accordingly.

- User program scan

  During the program scan, data in the input image area is applied to the user program, the user program is executed and the output image area is updated shown in Figure 2.7.

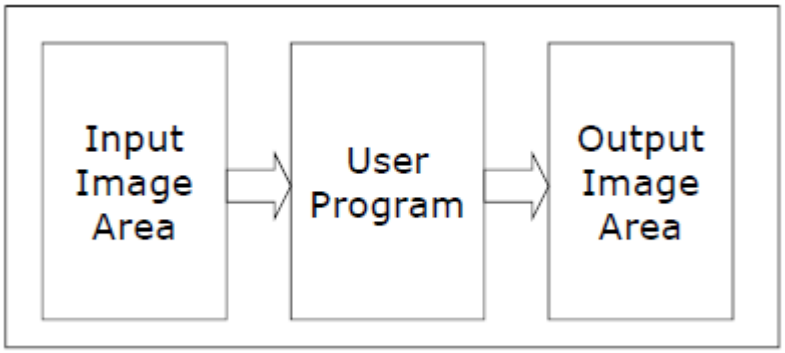


Figure 2.7: User program scan

- **Output scan**

During the output scan, data is taken from the output image area and sent to all output modules in the system shown in Figure 2.8 [4].

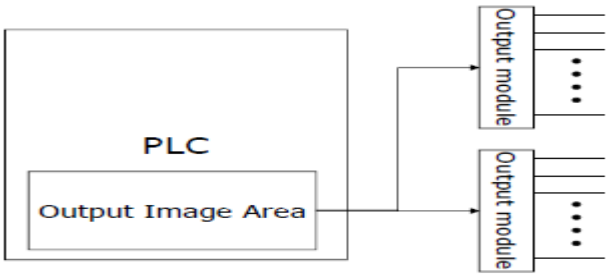


Figure 2.8: Output scan

2.4 PLC programming

As PLCs have developed and expanded, programming languages have developed with them. Programming languages allow the user to enter a control program into a PLC using an established syntax. Today's advanced languages have new, more versatile instructions, which initiate control program actions. These new instructions provide more computing power for

single operations performed by the instruction itself. For instance, PLCs can now transfer blocks of data from one memory location to another while, at the same time, performing a logic or arithmetic operation on another block. As a result of these new, expanded instructions, control programs can now handle data more easily. This may be done through a programming panel or by connection to a computer. There are several types of key pad to a full blown hand held computer with graphics screen. Computers are able to run programming software with graphics, simulation, diagnostics and monitoring. This could be a laptop carried to the site or a main computer some distance away. Often the programme is developed and tested on the computer and the programme is transferred to the PLC. This could be by a communication link, by a magnetic tape, compact disc or more likely with an EEPROM.

PLC programming equipment exists to allow you write, edit and monitor a program. In most cases the programming device, the program/monitor (PM) must be connected to CPU while the program is writer. Other PM allow you to program at line and the down load the program in PLC CPU, the program is written in ladder logic. In addition to new programming instructions, the development of powerful I/O modules has also changed existing instructions. These changes include the ability to send data to and obtain data from modules by addressing the modules locations. For example, PLCs can now read and write data to and from analog modules. All of these advances, in conjunction with projected industry needs, have created a demand for more powerful instructions that allow easier, more compact, function-oriented PLC programs[1]. The three types of programming languages used in PLCs are:

• Ladder

• Boolean

• Grafcet

The ladder and Boolean languages essentially implement operations in the same way, but they differ in the way their instructions are represented and how they are entered into the PLC. The Grafcet language implements control

instructions in a different manner, based on steps and actions in a graphic oriented program.

2.4.1 Ladder diagram

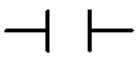
Programmable controllers are generally programmed in ladder diagram (or relay diagram) which is nothing but a symbolic representation of electric circuits. Symbols were selected that actually looked similar to schematic symbols of electric devices, and this has made it much easier for electricians to switch to programming PLC controllers. The line that defines the grouping of PLC ladder instructions, however, is usually drawn between functional instruction categories. These instruction categories include:

- Ladder relay.
- Timing.
- Counting.
- Program/flow control.
- Arithmetic.
- Data manipulation.
- Data transfer.
- Special function (sequencers).
- Network communication.

2.4.2 PLC ladder instructions

1) Contact symbol

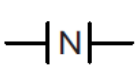
The basic ladder logic contact symbols are listed below.



Normally open (NO) contact. Passes power (**on**) if coil driving the contact is **on** (closed).

Normally closed (NC) contact. Passes power (**on**) if coil driving the contact is **off** (open).

Positive transition sensing contact. If conditions before this instruction change from **off** to **on**, this instruction passes power for only one scan (until rung is scanned again).

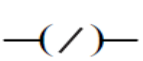


Negative transition sensing contact. If conditions before this instruction change from **on** to **off**, this instruction passes power for only one scan (until rung is scanned again).

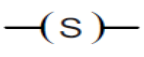
2) Coil symbols

Coils represent relays that are energized when power flows to them. When a coil is energized it causes a corresponding output to turn on by changing the state of the status bit controlling the output to 1. That same output status bit maybe used to control normally open or normally closed contact anywhere in the program. The basic ladder logic coil (output) symbols are listed below.

Output or *coil*. If any left-to-right path of instructions passes power, the output is energized. If there is no continuous left-to-right path of instructions passing power, the output is de-energized.

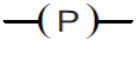


Negated coil. If any left-to-right path of inputs passes power, the output is de-energized. If there is no continuous left-to-right path of instructions passing power, the output is energized.

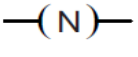


Set coil. If any rung path passes power, output is energized and remains energized, even when no rung path passes power.

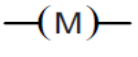
Reset coil. If any rung path passes power, output is de-energized and remains de-energized, even when no rung path passes power.



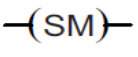
Positive transition sensing coil. If conditions before this instruction change from **off** to **on**, coil is turned **on** for one scan.



Negative transition sensing coil. If conditions before this instruction change from **on** to **off**, coil is turned **on** for one scan.



Retentive memory coil. Like the ordinary coil, except the value of the output is retained even when the PLC is stopped or power fails.

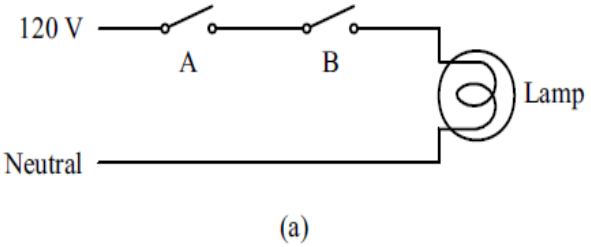


Set retentive memory coil. Like the set coil, except the value of the output is retained even when the PLC is stopped or power fails.

Reset retentive memory coil. Like the reset coil, except the value of the output is retained even when the PLC is stopped or power fails.

3) AND operation

Figure2.9 shows the series switch circuit and truth table.



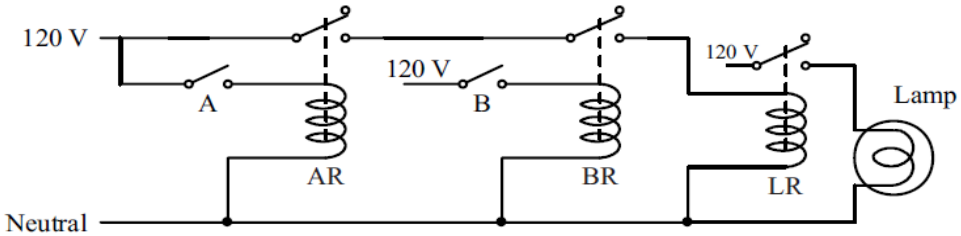
(a)

A	B	Lamp
off	off	off
off	on	off
on	off	off
on	on	on

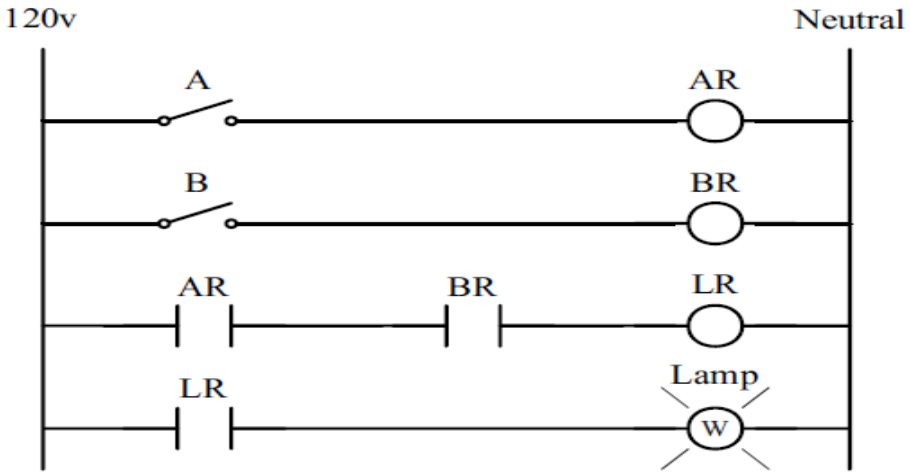
(b)

Figure 2.9: Series switch circuit and truth table

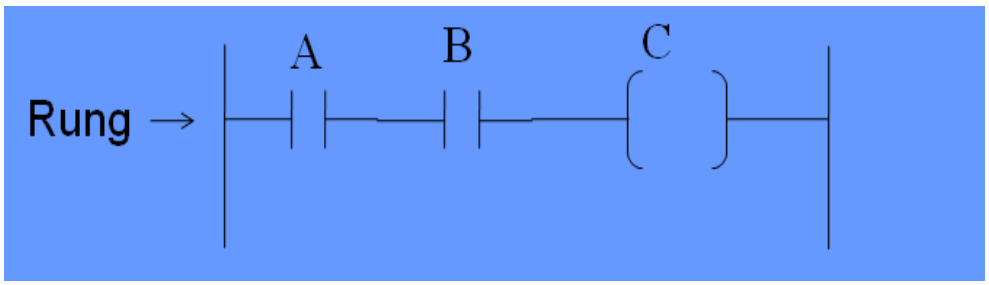
Figure 2.10 shows the series switch relay circuit, logic circuit and PLC ladder logic, respectively.



(a)



(b)



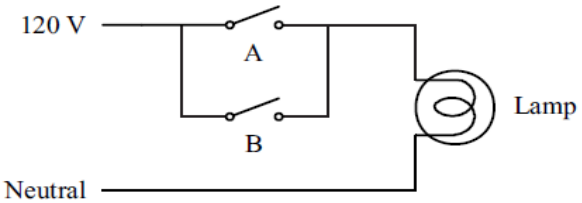
(c)

Figure 2.10: Series switch relay and ladder logic circuits: (a) relay circuit; (b) relay ladder logic circuits; (c) PLC ladder logic.

Each rung or network on a ladder program represents a logic operation. In the rung above, both inputs A and B must be true (1) in order for the output C to be true (1).

4) OR operation

Figure2.11 shows the series switch circuit and truth table.

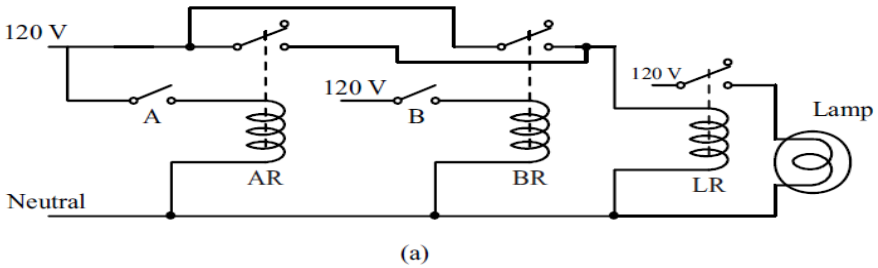


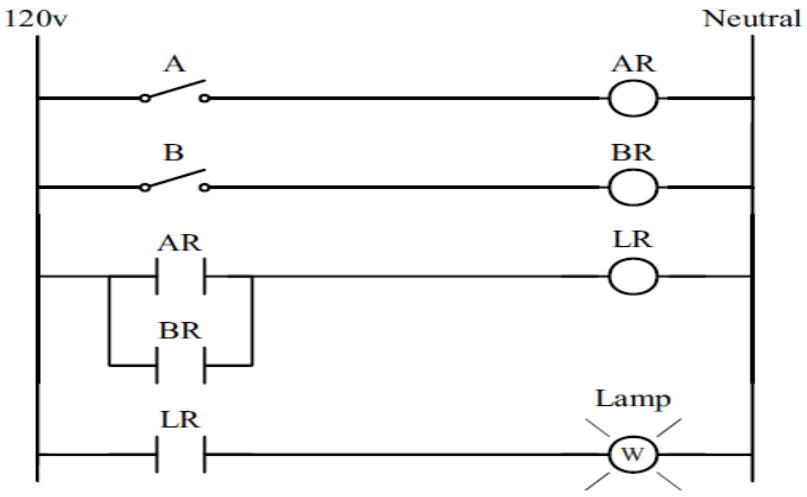
A	B	Lamp
off	off	off
off	on	on
on	off	on
on	on	on

(b)

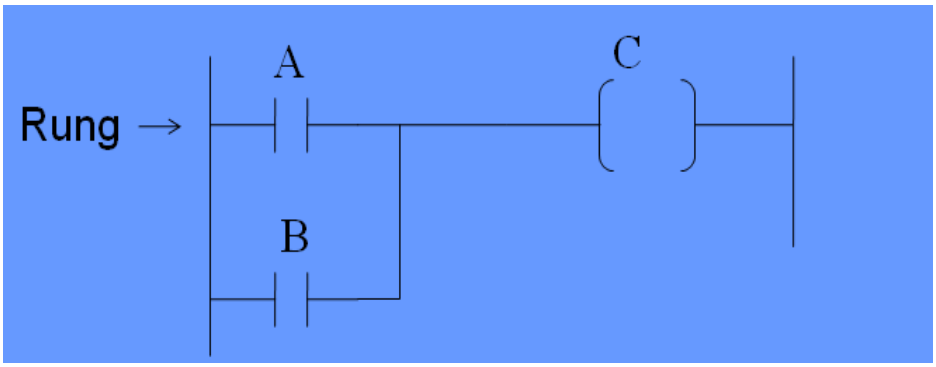
Figure 2.11: Parallel switch circuit and truth table

Figure 2.12 shows the parallel switch relay circuit, ladder logic citcuit and PLC ladder logic, respectively.





(b)



(c)

Figure 2.12: Parallel switch relay and ladder logic circuits: (a) relay circuit; (b) relay ladder logic circuits; (c) PLC ladder logic.

In the rung above, it can be seen that either input A or B is be true (1), or both are true, then the output C is true (1).

5) NOT operation

Figure 2-13 NOT operation

In the rung above, it can be seen that if input A is be true (1), then the output C is true (0) or when A is (0), output C is 1. A machine switches on if either of two start switches is closed and all of three stop switches are closed.

6) PLC timers

The timers of a PLC are realised in the form of software modules and are based on the generation of digital timing. Memory space is allocated in system memory to store the values of the delay time. The representation of the timer address varies from manufacturer to manufacturer. For sake of understanding consider the time T1, T2 for timer addresses. The typical numbers of timers available in commercial PLC are 64, 128, 256, 512 or even more. To explicitly reset timer, and Result of logic operation (RLO) of 1 has to be applied at the reset port. There are two types of PLC timer:

- PLC on delay timer: The timer will be ON state when it receives a start input signal and .The signal state of output changes to 0 from 1, when preset timing is reached. The signal state of the output changes from 0 to 1 when preset time has been reached with reference to change of RLO from 0 to 1(ON) at the start input .Functional diagram is shown in Figure 2.14:

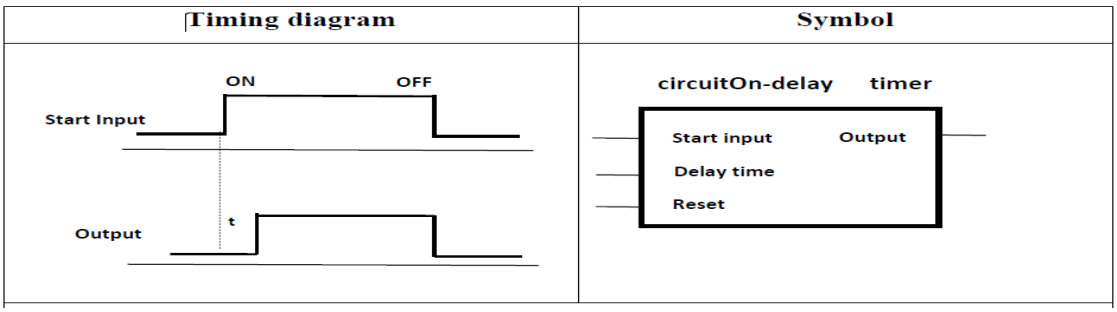


Figure 2.14: PLC on delay timer

- PLC off delay timer: The timer will be ON state when it receives a start input signal and the signal state of output changes to 1from 0, when preset timing is reached. The signal state of the output changes from 1 to 0 when preset time has been reached with reference to change of RLO from 1 to 0(OFF) at the start input. Functional diagram is shown in Figure 2.15:

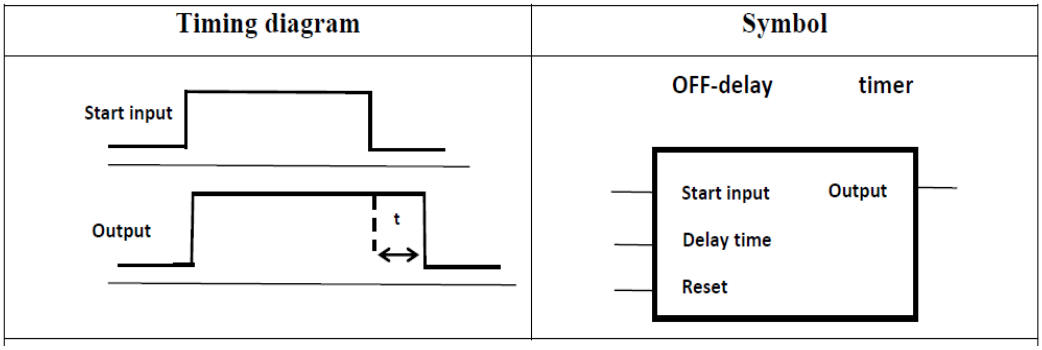


Figure 2.15: PLC OFF delay timer

7) PLC counters:

Counters are used to detect pieces numbers and events. Controllers frequently need to operate with counters in practice. For example: a counter in circuit is required if exactly 20 identical components are to be conveyed to a conveyor belt via a sorting device. There are two basic counter types a) Count Up b) Count down When the input to count up counter goes true the accumulator value will increase by 1 (no matter how long the input is true). If the accumulator value reaches the present value the counter bit will be set. A countdown counter will decrease the accumulator value until the preset value is reached. Symbols are shown in Figure 2.16 [5].

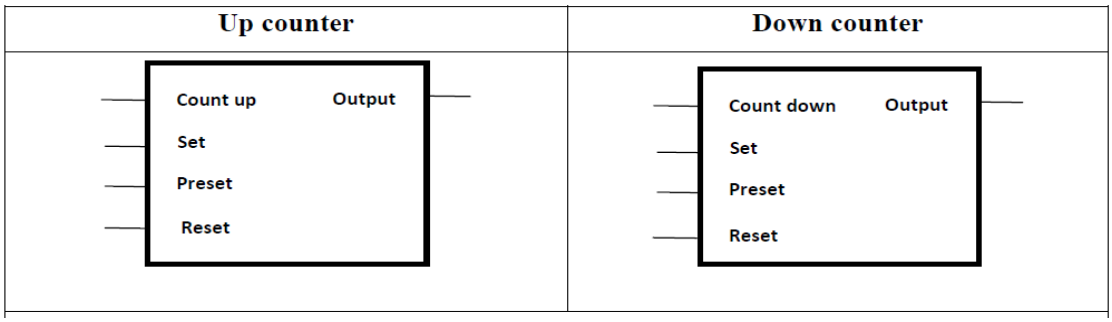


Figure 2.16: Up and down counter

2.4.3 Boolean

Some PLC manufacturers use Boolean language, also called Boolean mnemonics, to program a controller. The Boolean language uses Boolean algebra syntax to enter and explain the control logic. That is, it uses the AND, OR, and NOT logic functions to implement the control circuits in the control

program. Figure 2.17 shows a hardwired logic circuit and its Boolean representation.

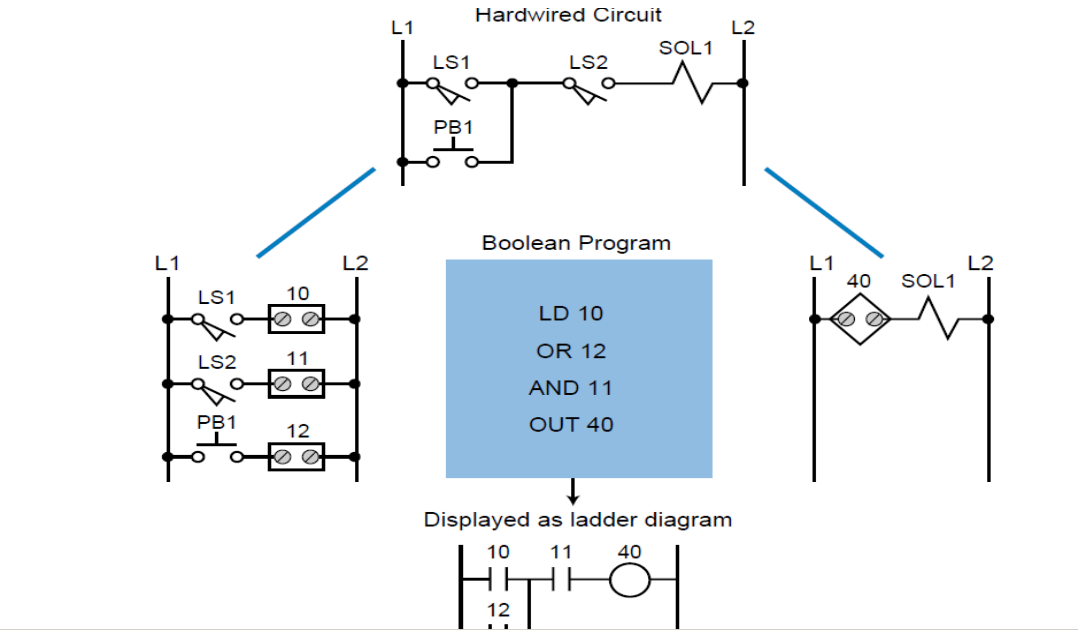


Figure 2.17: Hardwired logic circuit and its Boolean representation

The Boolean language is primarily just a way of entering the control program into a PLC, rather than an actual instruction-oriented language. When displayed on the programming monitor, the Boolean language is usually viewed as a ladder circuit instead of as the Boolean commands that define the instruction.

2.4.4 Grafcet

Grafcet (Graphe Fonctionnel de Commande Étape Transition) is a symbolic, graphic language, which originated in France that represents the control program as steps or stages in the machine or process. In fact, the English translation of Grafcet means step transition function charts. Few programmable controllers may be directly programmed using Grafcet. However, several Grafcet software manufacturers provide off-line Grafcet programming using a personal computer. Once programmed in the PC, the Grafcet instructions can be transferred to a PLC via a translator or driver that translates the Grafcet program into a ladder diagram or Boolean language

program. Using this method, a Grafcet software manufacturer can provide different PLCs that use the same language.

2.5 Introduction of Proportional Integral Derivation (PID)

As the general application of process control, the open loop methodology may be good enough for most situations, because the key control elements or components are more sophisticated, and the performances of which are getting better, there is no doubt, the stability and reliability may meet the desired requirement. It is the way to get not bad value with great economic consideration. But the characteristics of the elements or components may change following the time eclipse and the controlling process may be affected by the change of loading or external disturbances, the performance of open loop becomes looser; it is the weakness of such solution. Thus, closed loop (with the sensors to feedback the real conditions of controlling process for loop calculation) PID control is one of the best choices for manufacturing process to make perfect quantity and best products.

PLC provides digitized PID mathematical algorithm for general purpose application, it is enough for most of applications, but the response time of loop calculation will have the limitation by the scan time of PLC, thus it must be taken into consideration while in very fast closed loop control. Depends on the requirement, the users may apply the suitable controller for different applications; it is much better of the thinking that the control algorithm is so simple and easy to operate and the final result will be good enough, that's all. Types of controller could be activated from the PID mathematical expression as follows:

2.5.1 Proportional Controller (P)

P controller is mostly used in first order processes with single energy storage to stabilize the unstable process. The main usage of the P controller is to

decrease the steady state error of the system. The closed-loop transfer function with a proportional controller is:

$$\frac{X(s)}{F(s)} = \frac{K_p}{s^2 + 10s + (20 + K_p)} \quad (2.1)$$

The proportional gain factor Kp increases the steady state error of the system decreases, reduces the rise time and increases the overshoot.

2.5.2 Proportional Integral Controller (PI)

PI controller is mainly used to eliminate the steady state error resulting from P controller. However, in terms of the speed of the response and overall stability of the system, it has a negative impact. This controller is mostly used in areas where speed of the system is not an issue. Since PI controller has no ability to predict the future errors of the system it cannot decrease the rise time and eliminate the oscillations. If applied, any amount of I guarantees set point overshoot. The closed-loop transfer function with a PI control is:

$$\frac{X(s)}{F(s)} = \frac{K_p s + K_i}{s^3 + 10s^2 + (20 + K_p s + K_i)} \quad (2.2)$$

The integral controller (Ki) decreases the rise time, increases both the overshoot and the settling time, and eliminates the steady-state error.

2.5.3 Proportional Derivative Controller (PD)

The aim of using P-D controller is to increase the stability of the system by improving control since it has an ability to predict the future error of the system response. In order to avoid effects of the sudden change in the value of the error signal, the derivative is taken from the output response of the system variable instead of the error signal. Therefore, D mode is designed to be proportional to the change of the output variable to prevent the sudden changes occurring in the control output resulting from sudden changes in the error signal. The closed-loop transfer function of the given system with a PD controller is:

$$\frac{X(s)}{F(s)} = \frac{K_d s + K_p}{s^2 + (10 + K_d)s + (20 + K_p)} \quad (2.3)$$

The derivative controller (Kd) reduces both the overshoot and the settling time.

2.5.4 Proportional Integral Derivation Controller (PID)

P-I-D controller has the optimum control dynamics including zero steady state error, fast response (short rise time), no oscillations and higher stability.

The necessity of using a derivative gain component in addition to the PI controller is to eliminate the overshoot and the oscillations occurring in the output response of the system. One of the main advantages of the P-I-D controller is that it can be used with higher order processes including more than single energy storage [6]. The closed-loop transfer function of the given system with a PID controller is:

$$\frac{X(s)}{F(s)} = \frac{K_d s^2 + K_p s + K_i}{s^3 + (10 + K_d)s^2 + (20 + K_p)s + K_i} \tag{2.4}$$

The effects of each of controller parameters, Kp, Ki and Kd on a closed-loop system are summarized in the Table 2.1 [7].

Table 2.1 Response Constants

CL response	Rise time	Overshoot	Settling time
Kp	Decrease	Increase	Small Change
Ki	Decrease	Increase	Increase
Kd	Small Change	Decrease	Decrease

When you are designing a PID controller for a given system, follow the steps shown below to obtain a desired response.

1. Obtain an open-loop response and determine what needs to be improved.
2. Add a proportional control to improve the rise time
3. Add a derivative control to improve the overshoot

4. Add an integral control to eliminate the steady-state error

5. Adjust each of Kp, Ki, and Kd until you obtain a desired overall response [3].