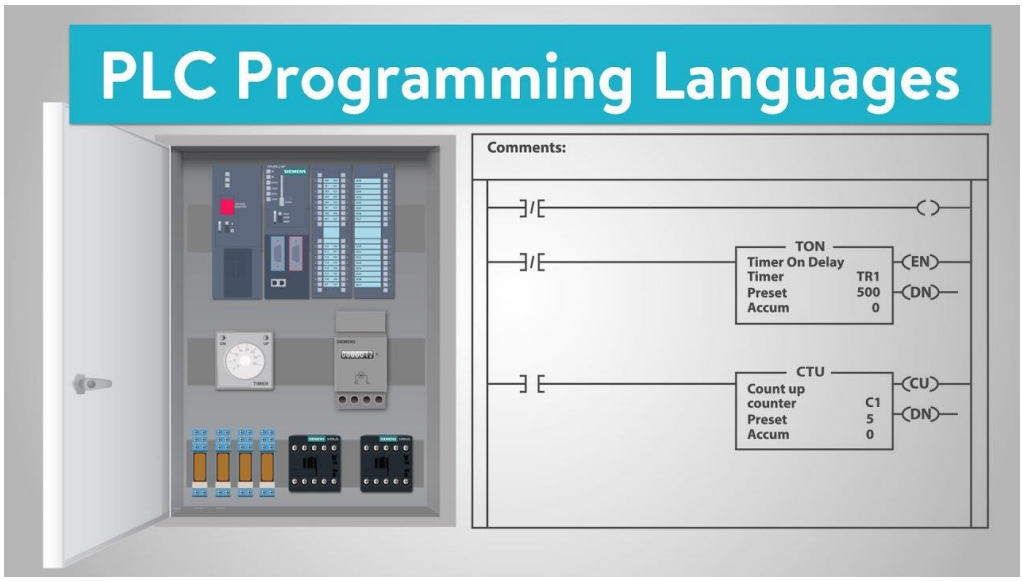
DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute Affiliated to VTU, Belagavi)

ShavigeMalleshwara Hills, Kumarswamy Layout, Bangalore-560111

Department of Robotics and Artificial Intelligence

DRIVE SYSTEMS FOR ROBOTICS (22RI34)



UNIT 5: NOTES

ELECTRICAL DRIVES CHARACTERISTICS AND PROGRAMMABLE LOGIC CONTROLLERS (PLC)

III SEMESTER, B.E.

DEPARTMENT OF ROBOTICS AND ARTIFICIAL INTELLIGENCE

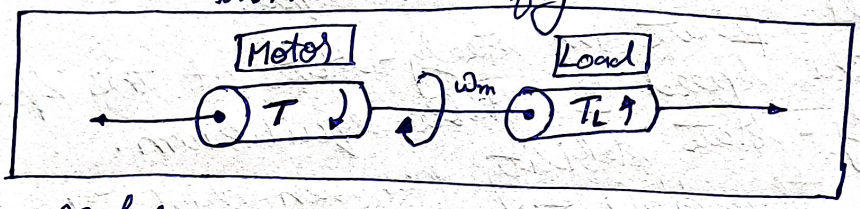
Faculty in charge:
Dr.Rakesh.M
Assistant Professor
D.S.C.E

Unit-5

Equations governing motor load dynamics :-

Fundamental Torque equations :-

An motor generally drives a load (machine) through some transmission system. While motor always rotates, the load may rotate or may undergo translational motion. The motor load system by an equivalent rotational system is as shown in the figure.



Various notations are used, as

→ J = Polar moment of inertia of motor-load system referred to motor shaft, $Kg\text{-}m^2$.

→ ω_m = Instantaneous angular velocity of motor shaft, rad/sec.

→ T = Instantaneous value of developed motor torque, N-m.

→ T_L = Instantaneous value of load (resisting) torque, referred to motor shaft, N-m.

Load torque includes friction & windage torque of motor. Motor-load system of above figure can be described by the following fundamental torque equations.

$$T - T_L = \frac{d}{dt}(J\omega_m) = J\frac{d\omega_m}{dt} + \omega_m\frac{dJ}{dt} \quad \text{---(1)}$$

motor

The above equation is applicable to variable inertia drives such as mine winders, industrial robots etc.

For drives with constant inertia $(dJ/dt) = 0$. Therefore

$$T = T_L + J\frac{d\omega_m}{dt} \quad \text{---(2)}$$

The above equation (2) shows that torque developed by motor is counter balanced by a load torque T_L and a dynamic torque $J(d\omega_m/dt)$. The torque component $J(d\omega_m/dt)$ is called the "dynamic torque" because it is present only during the transient operations.

(*) Steady State Stability :-

→ Equilibrium speed of a motor-load system is obtained when motor torque equals the load torque.

→ Drive will operate in steady-state at this speed, provided it is the speed of stable equilibrium.

→ The steady-state stability has been developed to readily evaluate the stability of equilibrium point from the steady state speed torque curves of the motor & the load, thus avoiding solution of differential equations valid for transient operations of the drive.

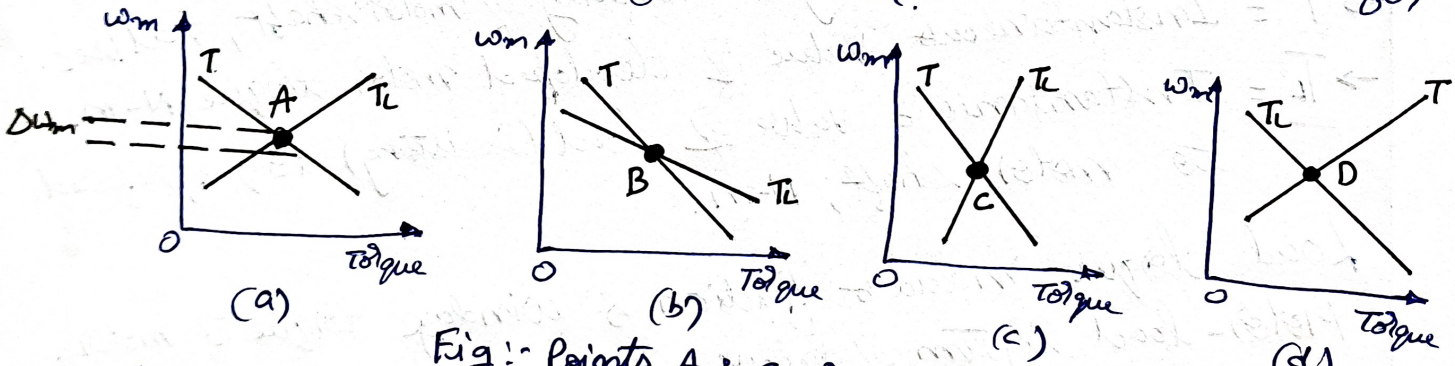


Fig :- Points A & C are stable and D are unstable

→ Steady state stability of equilibrium point A in fig 2.9(a),

→ The equilibrium point will be termed as stable when the operation will be restored to it after a small departure from it due to a disturbance in the motor or load.

→ The disturbance cause reduction in the motor or load; ie $\Delta\omega_m$. in the speed.

→ At new speed motor torque is greater than the load torque, motor will be accelerated & restored of operation to point 'A' hence the drive is steady-stable at point A.

→ If the same motor drives another load, let us examine the equilibrium at point B

Unit-5

Programmable Logic controllers

Discrete process control:-

→ Discrete process control systems deals with parameters and variables that are discrete and that are discrete and that change values at discrete moments in time.

→ The parameters and variables are typically binary, they can achieve two possible values 1, 0. the values mean ON or OFF, true or false, object present or not present, high voltage or low voltage value & so on.

→ The binary variables in discrete process control are associated with input signals to the controller & output signals from the controller.

→ Discrete process control can be divided into categories

① Logic control:- Event-driven changes in the system

② Sequence control:- time-driven changes in the system

* Both control types are referred to a switching systems; because they switch their output values on and off in response to changes in events or time.

Logic control:- Logic control system is a switching system whose output at any moment is determined exclusively by the values of the current inputs.

Programmable logic controllers :-

→ A Programmable logic controllers (PLC) can be defined as a microcomputer-based controller that uses stored instructions in a programmable memory to implement logic, sequencing timing, counting and arithmetic functions through digital & analog input/output (I/O) modules for both controlling and process the machines & any activity.

→ PLC applications are found both in the process industries & discrete manufacturing industries.

→ A PLC is an industrial computer control system that continuously monitors the state of input devices & makes decision based upon a custom program to control the state of output devices.

→ Examples of applications in process industries include chemical processing, paper mill operations and food production.

→ PLCs are primarily associated with discrete manufacturing industries to control individual machines, machine cell, transfer lines, material handling equipment, robotics, and automated storage devices.

⊛ Elements of Logic control :-

→ The basic elements of logic control are the logic gates AND, OR and NOT.

→ In each case, the logic gates is designed to provide a specified output value based on the values of the input(s).

→ For both inputs & outputs, the values can be either binary values: 0 or 1.

→ For purposes of industrial control, 0 (zero) is defined as OFF and 1 (one) as ON.

→

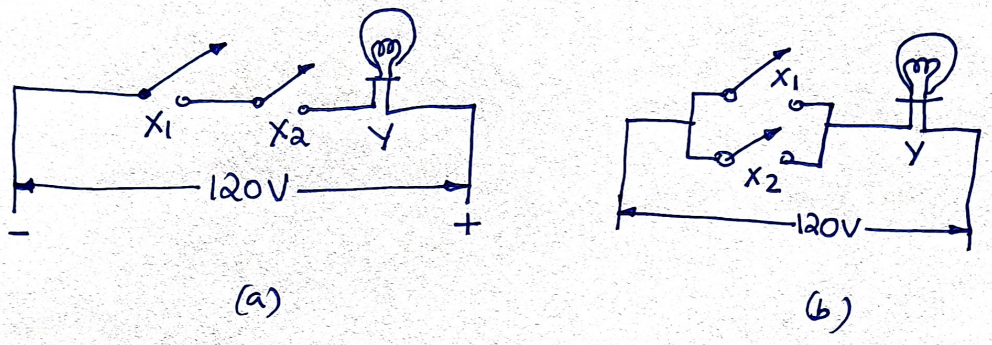


fig-①: Electrical circuit illustrating the operations of logic gates @AND & ⑥OR

→ The AND gate outputs a value of 1, if all of the inputs are 1 and otherwise, fig-① @ represents the logic gate operation of AND.

→ If switch X_1 & X_2 representing inputs in the circuit are both closed, the lamp (Y) (representing the output) is on.

→ The AND gate might be used in automated manufacturing operations to indicate that two or more operations/actions is successfully completed. eg:- The robot works in forging industries only if all three conditions must be satisfied before loading & unloading forge press as it is incorporated with interlock systems.

→ The OR gate outputs a value of 1 if either of the inputs has value of 1 and 0 otherwise. Figure-1(b) shows how the OR gate operates. In this case, two input signals X_1 & X_2 are arranged in parallel circuit, so that if either switch is closed the lamp (Y) will turn on.

→ The possible use of OR gate in manufacturing system is for safety monitoring, suppose that two sensors are utilized to monitor two different safety hazards. When either hazards occurs, the respective sensor emits a positive signal that sounds alarm.

Both the AND and OR gates can be used with two or more inputs. The NOT gate has a single input. The NOT gate reverses the input signal, if the input is 1, then the output is 0; if the input is 0, then the output is 0. The NOT gate can be used to open a circuit upon receipt of a control signal.

TABLE 9.1 Binary Sensors and Actuators Used in Discrete Process Control

Sensor	*One/Zero Interpretation*	*Actuator*	*One/Zero Interpretation*
Limit switch	*Contact/no contact*	Motor	On/off
Photodetector	*On/off*	Control relay	Contact/no contact
Push-button switch	*On/off*	Light	On/off
Timer	*On/off*	Valve	Closed/open
Control relay	*Contact/no contact*	Clutch	Engaged/not engaged
Circuit breaker	*Contact/no contact*	Solenoid	Energized/not energized

SCAN AnyScanner

Boolean Algebra and Truth Tables. The logic elements form the foundation for a special algebra that was developed around 1847 by George Boole and that bears his name. Its original purpose was to provide a symbolic means of testing whether complex statements of logic were TRUE or FALSE. In fact, Boole called it *logical algebra*. It was not until about a century later that Boolean algebra was shown to be useful in digital logic systems. We briefly describe some of its fundamentals here.

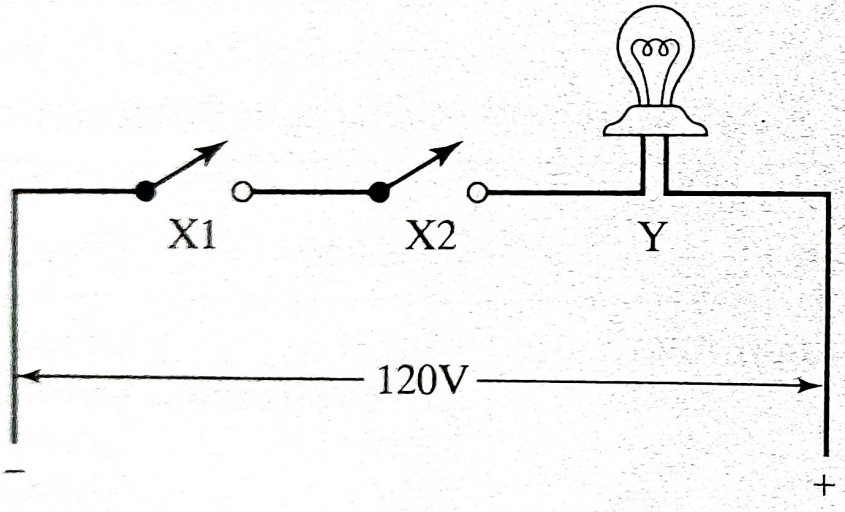


Figure 9.1 Electrical circuit illustrating the operation of the logical AND gate.

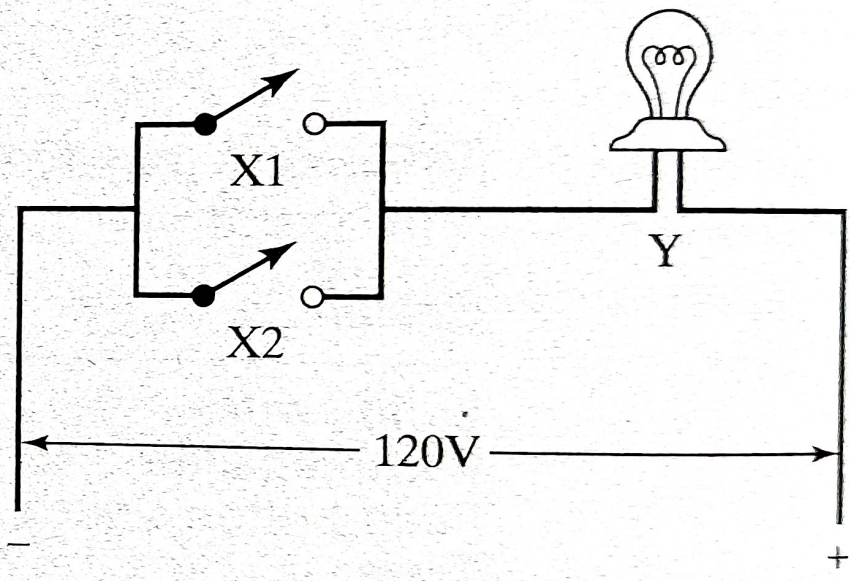


Figure 9.2 Electrical circuit illustrating the operation of the logical OR gate.

SCAN AnyScanner

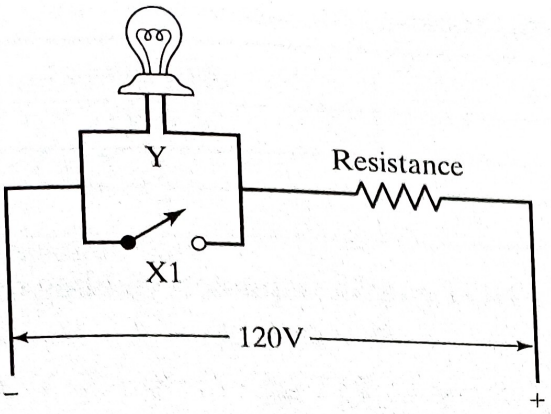


Figure 9.3 Electrical circuit illustrating the operation of the logical NOT gate.

In Boolean algebra, the AND function is expressed as

$$Y = X1 \cdot X2 \tag{9.1}$$

This is called the logical product of X1 and X2. As a logic statement, it means: Y is true if both X1 and X2 are true; otherwise, Y is false. The truth table is often used to present the operation of logic systems. A *truth table* is a tabulation of all of the combinations of input values to the corresponding logical output values. The truth table for the AND gate for four possible combinations of two input binary variables is presented in Table 9.2.

The OR function in Boolean algebra notation is given by

$$Y = X1 + X2 \tag{9.2}$$

This is called the logical sum of X1 and X2. In logic, the statement says: Y is true if either X1 or X2 is true; otherwise, Y is false. The outputs of the OR function for four possible combinations of two input binary variables are listed in the truth table of Table 9.3.

The NOT function is referred to as the negation or inversion of the variable. It is indicated by placing a bar above the variable (e.g., $\overline{X1}$). The truth table for the NOT function is listed in Table 9.4, and the corresponding Boolean equation is as follows:

$$Y = \overline{X1} \tag{9.3}$$

In addition to the three basic elements, there are two more elements that can be used in switching circuits: the NAND and NOR gates. The logical NAND gate is formed

TABLE 9.2 Truth Table for the Logical AND Gate

Inputs		*Output*
X1	X2	$Y = X1 \cdot X2$
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 9.3 Truth Table for the Logical OR Gate

Inputs		*Output*
X1	X2	$Y = X1 + X2$
0	0	0
0	1	1
1	0	1
1	1	1

SCAN AnyScanner

TABLE 9.4 Truth Table for the Logical NOT Gate

Inputs	*Output*
X1	$Y = \overline{X1}$
0	1
1	0

by combining an AND gate and a NOT gate in sequence, yielding the truth table shown in Table 9.5(a). In equation form,

$$Y = \overline{X1 \cdot X2} \tag{9.4}$$

The logical NOR gate is formed by combining an OR gate followed by a NOT gate, providing the truth table in Table 9.5(b). The Boolean algebra equation for the NOR gate is written as follows:

$$Y = \overline{X1 + X2} \tag{9.5}$$

Various diagramming techniques have been developed to represent the logic elements and their relationships in a given logic control system. The logic network diagram is one of the most common methods. Symbols used in the logic network diagram are illustrated in Figure 9.4. We demonstrate the use of the logic network diagram in several examples later in this section.

TABLE 9.5 Truth Tables for (a) the Logical NAND Gate and (b) Logical NOR Gate

(a) NAND			(b) NOR		
Inputs		*Output*	*Inputs*		*Output*
X1	X2	$Y = \overline{X1 \cdot X2}$	X1	X2	$Y = \overline{X1 + X2}$
0	0	1	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0

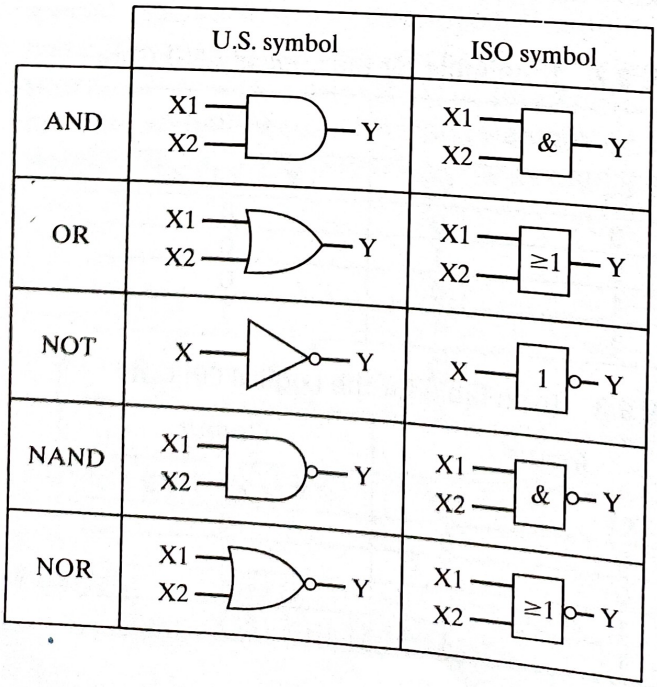


Figure 9.4 Symbols used for logical gates: U.S. and ISO.

TABLE 9.6 Laws and Theorems of Boolean Algebra

Commutative Law:

$$X + Y = Y + X$$
$$X \cdot Y = Y \cdot X$$

Associative Law:

$$X + Y + Z = X + (Y + Z)$$
$$X + Y + Z = (X + Y) + Z$$
$$X \cdot Y \cdot Z = X \cdot (Y \cdot Z)$$
$$X \cdot Y \cdot Z = (X \cdot Y) \cdot Z$$

Distributive Law:

$$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$$
$$(X + Y) \cdot (Z + W) = X \cdot Z + X \cdot W + Y \cdot Z + Y \cdot W$$

Law of Absorption:

$$X \cdot (X + Y) = X + X \cdot Y = X$$

De Morgan's Laws:

$$\overline{(X + Y)} = \overline{X} \cdot \overline{Y}$$
$$\overline{(X \cdot Y)} = \overline{X} + \overline{Y}$$

Consistency Theorem:

$$X \cdot Y + X \cdot \overline{Y} = X$$
$$(X + Y) \cdot (X + \overline{Y}) = X$$

Inclusion Theorem:

$$X \cdot \overline{X} = 0$$
$$X + \overline{X} = 1$$

SCAN AnyScanner

LADDER LOGIC DIAGRAMS

The logic network diagrams of the type shown in Figures 9.5 and 9.6(b) are useful for displaying the relationships between logic elements. Another diagramming technique that exhibits the logic and, to some extent, the timing and sequencing of the system is the ladder logic diagram. This graphical method has an important virtue in that it is analogous to the electrical circuits used to accomplish the logic and sequence control. In addition, ladder logic diagrams are familiar to shop personnel who must construct, test, maintain, and repair the discrete control system.

In a ladder logic diagram, the various logic elements and other components are displayed along horizontal lines or rungs connected on either end to two vertical rails, as illustrated in Figure 9.7. The diagram has the general configuration of a ladder, hence its name. The elements and components are *contacts* (representing logical inputs) and loads, also known as *coils* (representing outputs). Inputs include switches and relay contacts, and

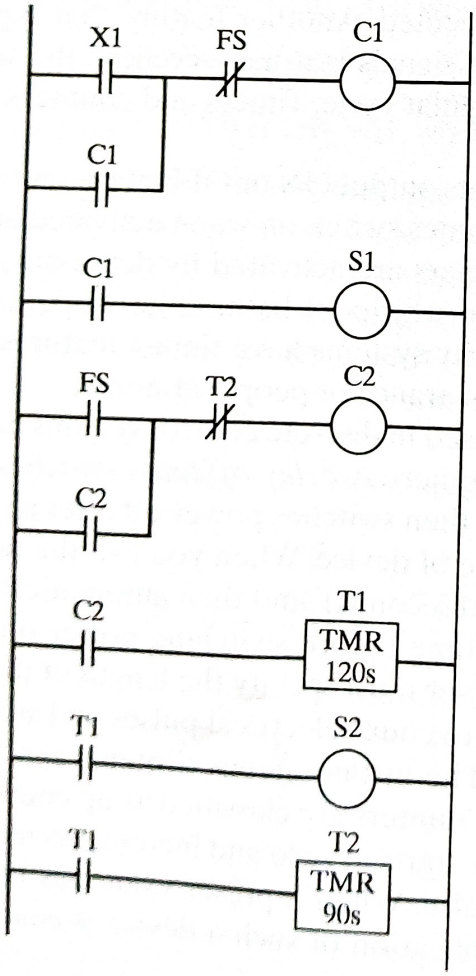


Figure 9.7 A ladder logic diagram.

loads include motors, lamps, and alarms. The power (e.g., 115 V alternating current) to the components is provided by the two vertical rails. It is customary in ladder diagrams to locate the inputs to the left of each rung and the outputs to the right.

Symbols used in ladder diagrams for the common logic and sequencing components are presented in Figure 9.8. There are two types of contacts: normally open and normally closed. A *normally open contact* remains open until activated. When activated, it closes to allow current to flow. A *normally closed contact* remains closed, allowing current to flow until activated. When activated, it opens, thereby turning off the flow of current. Normally open contacts of a switch or similar device are symbolized by two short vertical lines along a horizontal rung of the ladder, as in Figure 9.8(a). Normally closed contacts are shown as the same vertical lines only with a diagonal line across them as in Figure 9.8(b). Both types of contacts are used to represent ON/OFF inputs to the logic circuit. In addition to switches, inputs include relays, on/off sensors (e.g., limit switches and photodetectors), timers, and other binary contact devices.

Output loads such as motors, lights, alarms, solenoids, and other electrical components that are turned on and off by the logic control system are shown as nodes (circles) as in Figure 9.8(c). Timers and counters are symbolized by squares (or rectangles) with appropriate inputs and outputs to properly drive the device as shown in Figure 9.8(d) and (e). The simple timer requires the specification of the time delay and the identification of the input contact that activates the delay. When the input signal is received, the timer waits the specified delay time before switching on or off the output signal. The timer is reset (output is set back to its initial value) by turning off the input signal.

Counters require two inputs. The first is the pulse train (series of on/off signals) that is counted by the counter. The second is a signal to reset the counter and restart the counting procedure. Resetting the counter means zeroing the count for an up counter and setting the starting value for a down counter. The accumulated count is retained in memory for use if required for the application.

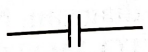
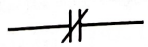
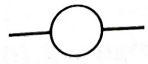
Ladder symbol	Hardware component
(a) 	Normally open contacts (switch, relay, other ON/OFF devices)
(b) 	Normally closed contacts (switch, relay, etc.)
(c) 	Output loads (motor, lamp, solenoid, alarm, etc.)
(d) 	Timer
(e) 	Counter

Figure 9.8 Symbols for common logic and sequence elements used in ladder logic diagrams.

SCAN AnyScanner

EXAMPLE 9.3 Three Simple Lamp Circuits

The three basic logic gates (AND, OR, and NOT) can be symbolized in ladder logic diagrams. Create diagrams for the three lamp circuits illustrated in Figures 9.1, 9.2, and 9.3.

Solution: The three ladder diagrams corresponding to these circuits are presented in Figure 9.9(a)-(c). Note the similarity between the original circuit diagrams and the ladder diagrams shown here. Notice that the NOT symbol is the same as a normally closed contact, which is the logical inverse of a normally open contact.

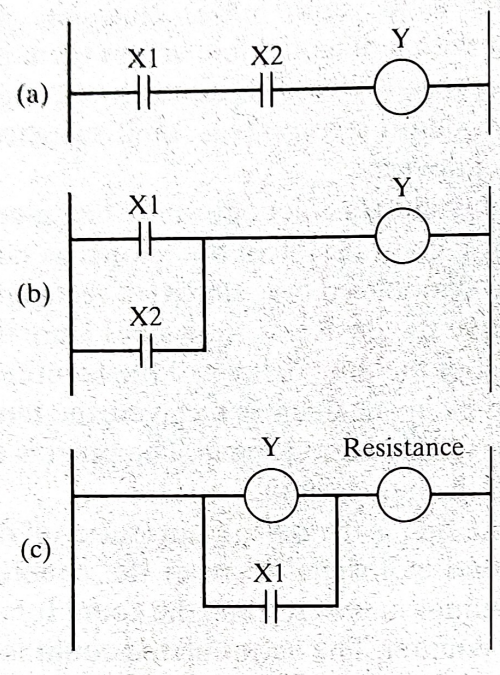


Figure 9.9 Three ladder logic diagrams for lamp circuits in (a) Figure 9.1, (b) Figure 9.2, and (c) Figure 9.3.

EXAMPLE 9.4 Push-Button Switch

The operation of the push-button switch of Example 9.2 can be depicted in a ladder logic diagram. From Figure 9.6, let START be represented by X1, STOP by X2, and MOTOR by Y, and create the diagram.

Solution: The ladder diagram is presented in Figure 9.10. X1 and X2 are input contacts, and Y is a load in the diagram. Note how Y also serves as an input contact to provide the POWER-TO-MOTOR connection.

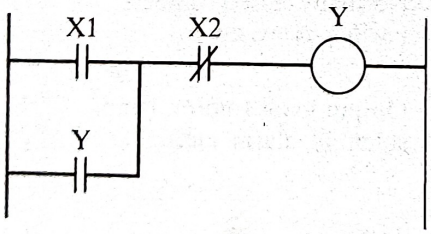


Figure 9.10 Ladder logic diagram for the push-button switch in Example 9.4.

EXAMPLE 9.5 Control Relay

The operation of a control relay can be demonstrated by means of the ladder logic diagram presented in Figure 9.11. A relay can be used to control on/off actuation of a powered device at some remote location. It can also be used to

SCAN AnyScanner

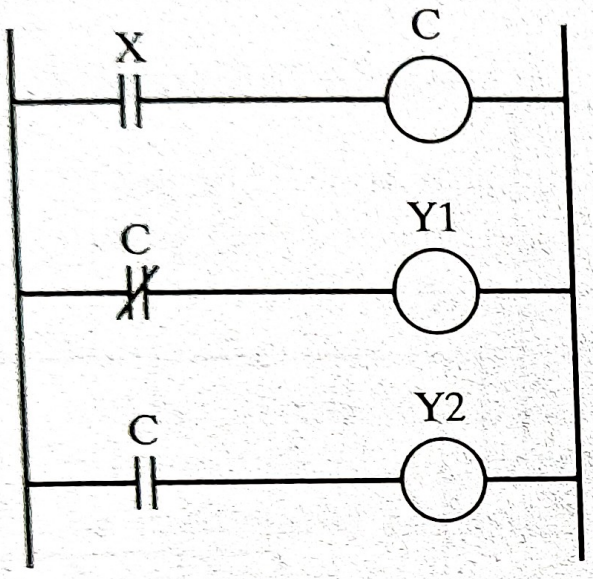


Figure 9.11 Ladder logic diagram for the control relay in Example 9.5.

define alternative decisions in logic control. Our diagram illustrates both uses. The relay is indicated by the load C (for control relay), which controls the on/off operation of two motors (or other types of output loads) Y1 and Y2. When the control switch X is open, the relay is deenergized, thereby connecting the load Y1 to the power lines. In effect, the open switch X turns on motor Y1 by means of the relay. When the control switch is closed, the relay becomes energized. This opens the normally closed contact of the second rung of the ladder and closes the normally open contact of the third rung. In effect, power is shut off to load Y1 and turned on to load Y2.

AnyScanner

PROGRAMMABLE LOGIC CONTROLLERS

A programmable logic controller (PLC) can be defined as a microcomputer-based controller that uses stored instructions in programmable memory to implement logic, sequencing, timing, counting, and arithmetic functions through digital or analog input/output (I/O) modules, for controlling machines and processes. PLC applications are found in both the process industries and discrete manufacturing. Examples of applications in process industries include chemical processing, paper mill operations, and food production. PLCs are primarily associated with discrete manufacturing industries to control individual machines, machine cells, transfer lines, material handling equipment, and automated storage systems. Before the PLC was introduced around 1970, hard-wired controllers composed of relays, coils, counters, timers, and similar components were used to implement this type of industrial control (Historical Note 9.1). Today, many older pieces of equipment are being retrofitted with PLCs to replace the original hard-wired controllers, often making the equipment more productive and reliable than when it was new.

SCAN AnyScanner

There are significant advantages to using a PLC rather than conventional relays, timers, counters, and other hard-wired control components. These advantages include (1) programming the PLC is easier than wiring the relay control panel; (2) the PLC can be reprogrammed, whereas conventional controls must be rewired and are often scrapped instead; (3) PLCs take less floor space than relay control panels; (4) reliability is greater, and maintenance is easier; (5) the PLC can be connected to computer systems more easily than relays; and (6) PLCs can perform a greater variety of control functions than can relay controls.

SCAN AnyScanner

9.3.1 Components of the PLC

A schematic diagram of a PLC is presented in Figure 9.13. The basic components of the PLC are the following: (1) processor, (2) memory unit, (3) power supply, (4) I/O module, and (5) programming device. These components are housed in a suitable cabinet designed for the industrial environment.

The *processor* is the central processing unit (CPU) of the programmable controller. It executes the various logic and sequencing functions by operating on the PLC inputs to determine the appropriate output signals. The typical CPU operating cycle is described in Section 9.3.2. The CPU consists of one or more microprocessors similar to those used in PCs and other data processing equipment. The difference is that they have a real-time operating system and are programmed to facilitate I/O transactions and execute ladder logic functions. In addition, PLCs are hardened so that the CPU and other electronic components will operate in the electrically noisy environment of the factory.

Connected to the CPU is the PLC *memory unit*, which contains the programs of logic, sequencing, and I/O operations. It also holds data files associated with these programs, including I/O status bits, counter and timer constants, and other variable and parameter values. This memory unit is referred to as the user or application memory because its contents are entered by the user. In addition, the processor also contains the operating system memory, which directs the execution of the control program and coordinates I/O operations. The operating system is entered by the PLC manufacturer and cannot be accessed or altered by the user.

A *power supply* of 115 V ac is typically used to drive the PLC (some units operate on 230 V ac). The power supply converts the 115 V ac into direct current (dc) voltages of ±5 V. These low voltages are used to operate equipment that may have much higher voltage and power ratings than the PLC itself. The power supply often includes a battery backup that switches on automatically in the event of an external power source failure.

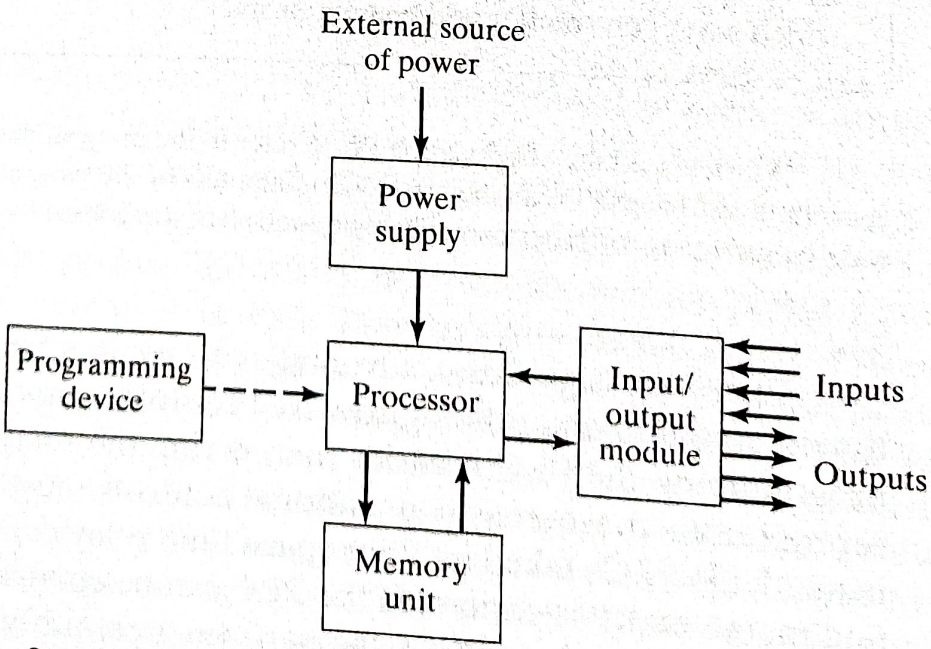


Figure 9.13 Components of a PLC.

SCAN AnyScanner

The *input/output module* provides the connections to the industrial equipment or process that is to be controlled. Inputs to the controller are signals from limit switches, push-buttons, sensors, and other on/off devices. Outputs from the controller are on/off signals to operate motors, valves, and other devices required to actuate the process. In addition, many PLCs are capable of accepting continuous signals from analog sensors and generating signals suitable for analog actuators. The size of a PLC is usually rated in terms of the number of its I/O terminals, as indicated in Table 9.8.

The PLC is programmed by means of a programming device. The programming device is usually detachable from the PLC cabinet so that it can be shared among different controllers. Different PLC manufacturers provide different devices, ranging from simple teach-pendant type devices, similar to those used in robotics, to special PLC programming keyboards and displays. Personal computers can also be used to program PLCs. A PC used for this purpose sometimes remains connected to the PLC to serve a process monitoring or supervisory function and for conventional data processing applications related to the process.

SCAN AnyScanner

9.3.4 Programming the PLC

Programming is the means by which the user enters the control instructions to the PLC through the programming device. The most basic control instructions consist of switching, logic, sequencing, counting, and timing. Virtually all PLC programming methods provide instruction sets that include these functions. Many control applications require additional instructions to accomplish analog control of continuous processes, complex control logic, data processing and reporting, and other advanced functions not readily performed by the basic instruction set. Owing to these differences in requirements, various PLC programming languages have been developed. A standard for PLC programming was published by the International Electrotechnical Commission in 1992, entitled *International Standard for Programmable Controllers* (IEC 1131–3). This standard specifies three graphical languages and two text-based languages for programming PLCs, respectively: (1) ladder logic diagrams, (2) function block diagrams, (3) sequential functions charts, (4) instruction list, and (5) structured text. Table 9.9 lists the five languages along with the most suitable application of each. IEC 1131–3 also states that the five languages must be able to interact with each other to allow for all possible levels of control sophistication in any given application.

SCAN AnyScanner

TABLE 9.9 Features of the Five PLC Languages Specified in the IEC 1131–3 Standard

Language	*Abbreviation*	*Type*	*Applications Best Suited for*
Ladder logic diagram	(LD)	Graphical	Discrete control
Function block diagram	(FBD)	Graphical	Continuous control
Sequential function chart	(SFC)	Graphical	Sequencing
Instruction list	(IL)	Textual	Discrete control
Structured text	(ST)	Textual	Complex logic, computations, etc.

Ladder Logic Diagram. The most widely used PLC programming language today involves ladder diagrams (LDs), examples of which are shown in several previous figures. As indicated in Section 9.2, ladder diagrams are very convenient for shop personnel who are familiar with ladder and circuit diagrams but may not be familiar with computers and computer programming. To use ladder logic diagrams, they do not need to learn an entirely new programming language.

Direct entry of the ladder logic diagram into the PLC memory requires the use of a keyboard and monitor with graphics capability to display symbols representing the components and their interrelationships in the ladder logic diagram. The symbols are similar to those presented in Figure 9.8. The PLC keyboard is often designed with keys for each of the individual symbols. Programming is accomplished by inserting the appropriate components into the rungs of the ladder diagram. The components are of two basic types: contacts and coils, as described in Section 9.2. Contacts represent input switches, relay contacts, and similar elements. Coils represent loads such as motors, solenoids, relays, timers, and counters. In effect, the programmer inputs the ladder logic circuit diagram rung by rung into the PLC memory with the monitor displaying the results for verification.

Function Block Diagrams. A function block diagram (FBD) provides a means of inputting high-level instructions. Instructions are composed of operational blocks. Each block has one or more inputs and one or more outputs. Within a block, certain operations take place on the inputs to transform the signals into the desired outputs. The function blocks include operations such as timers and counters, control computations using equations (e.g., proportional-integral-derivative control), data manipulation, and data transfer to other computer-based systems. We leave further description of these function blocks to other references, such as Hughes [3] and the operating manuals for commercially available PLC products.

Sequential Function Charts. The sequential function chart (SFC, also called the *Grafcet* method) graphically displays the sequential functions of an automated system as a series of steps and transitions from one state of the system to the next. The sequential function chart is described in Boucher [1]. It has become a standard method for documenting logic control and sequencing in much of Europe. However, its use in the United States is more limited, and we refer the reader to the cited reference for more details on the method.

Instruction List. Instruction list (IL) programming also provides a way of entering the ladder logic diagram into PLC memory. In this method, the programmer uses a low-level computer language to construct the ladder logic diagram by entering statements

SCAN AnyScanner

TABLE 9.10 Typical Low-Level Language Instruction Set for a PLC

STR	Store a new input and start a new rung of the ladder.
AND	Logical AND referenced with the previously entered element. This is interpreted as a series circuit relative to the previously entered element.
OR	Logical OR referenced with the previously entered element. This is interpreted as a parallel circuit relative to the previously entered element.
NOT	Logical NOT or inverse of entered element.
OUT	Output element for the rung of the ladder diagram.
TMR	Timer element. Requires one input signal to initiate timing sequence. Output is delayed relative to input by a duration specified by the programmer in seconds. Resetting the timer is accomplished by interrupting (stopping) the input signal.
CTR	Counter element. Requires two inputs: One is the incoming pulse train that is counted by the CTR element, the other is the reset signal indicating a restart of the counting procedure.

that specify the various components and their relationships for each rung of the ladder diagram. Let us explain this approach by introducing a hypothetical PLC instruction set. Our PLC "language" is a composite of various manufacturers' languages. It contains fewer features than most commercially available PLCs. We assume that the programming device consists of a suitable keyboard for entering the individual components on each rung of the ladder logic diagram. A monitor capable of displaying each ladder rung (and perhaps several rungs that precede it) is useful to verify the program. The instruction set for our PLC is presented in Table 9.10 with a concise explanation of each instruction. Let us examine the use of these commands with several examples.

EXAMPLE 9.7 Language Commands for AND, OR, and NOT Circuits

Using the command set in Table 9.10, write the PLC programs for the three ladder diagrams from Figure 9.10, depicting the AND, OR, and NOT circuits from Figures 9.1, 9.2, and 9.3.

Solution: Commands for the three circuits are listed below, with explanatory comments.

Command	Comment
(a) STR X1	Store input X1
AND X2	Input X2 in series with X1
OUT Y	Output Y
(b) STR X1	Store input X1
OR X2	Input X2 parallel with X1
OUT Y	Output Y
(c) STR NOT X1	Store inverse of X1
OUT Y	Output Y

SCAN AnyScanner

In the early 1990s, PCs began to encroach into applications formerly dominated by PLCs. Previously, PLCs were favored for use in factories because they were designed to operate in harsh environments, while PCs were designed for office environments. In addition, with their built-in I/O interfaces and real-time operating systems, PLCs could be readily connected to external equipment for process control, whereas PCs required special I/O cards and programs to enable such functions. Finally, personal computers sometimes lock up for no apparent cause, and usually lockups cannot be tolerated in industrial control applications. PLCs are not prone to such malfunctions.

These PLC advantages notwithstanding, the technological evolution of programmable logic controllers has not kept pace with the development of personal computers, new generations of which are introduced with much greater frequency than PLCs. There is much more proprietary software and architecture in PLCs than in PCs, making it difficult to mix and match components from different vendors. Over time, these factors have resulted in a performance disadvantage for PLCs. Performance lags its PC counterpart by as much as two years, and the gap is increasing. PC speeds are typically doubling every 18 months or so, while improvements in PLC technology occur much more slowly and require that individual companies redesign their proprietary software and architectures for each new generation of microprocessors.

PCs are now available in more sturdy enclosures for the dirty and noisy plant environment. They can be equipped with membrane-type keyboards for protection against

AnyScanner

factory moisture, oil, and dirt. They can be ordered with I/O cards and related hardware to provide the necessary devices to connect to the plants' equipment and processes. Operating systems designed to implement real-time control applications can be installed in addition to traditional office software. PLC makers are responding to the PC challenge by including PC components and features in their controller products to distinguish them from conventional PLCs. Nevertheless, the future is likely to see increasing numbers of PCs used in factory control applications where PLCs would have formerly been used.

There are two basic approaches used in PC-based control systems [10]: soft logic and hard real-time control. In the *soft logic* configuration, the PC's operating system is Windows, and control algorithms are installed as high-priority programs under the operating system. However, it is possible to interrupt the control tasks in order to service certain system functions in Windows, such as network communications and disk access. When this happens, the control function is delayed, with possible negative consequences to the process. Thus, a soft logic control system cannot be considered a real-time controller in the sense of a PLC. In high-speed control applications or volatile processes, lack of real-time control is a potential hazard. In less critical processes, soft logic works well.

In a *hard real-time control* system, the PC's operating system is the real-time operating system, and the control software takes priority over all other software. Windows tasks are executed at a lower priority under the real-time operating system. Windows cannot interrupt the execution of the real-time controller. If Windows locks up, it does not affect the controller operation. Also, the real-time operating system resides in the PC's active memory, so a failure of the hard disk has no effect in a hard real-time control system.

SCAN AnyScanner

SCADA :

SCADA (**supervisory Control and Data Acquisition**) generally refers to industrial control system (ICS): Supervisory Control and Data Acquisition (SCADA) systems are essential for monitoring and controlling industrial processes and infrastructure in various sectors such as power generation, water treatment, oil and gas, manufacturing, and transportation. They provide real-time data collection, control, and analysis of equipment and processes, enabling operators to ensure that everything runs efficiently and safely. SCADA (Supervisory Control and Data Acquisition) is a system that collects data from various sensors at a factory, plant, or in other remote locations and sends this data to a central computer that then manages and controls the data.

That is computer systems that monitor and control industrial, infrastructure, or facility-based processes, as described below:

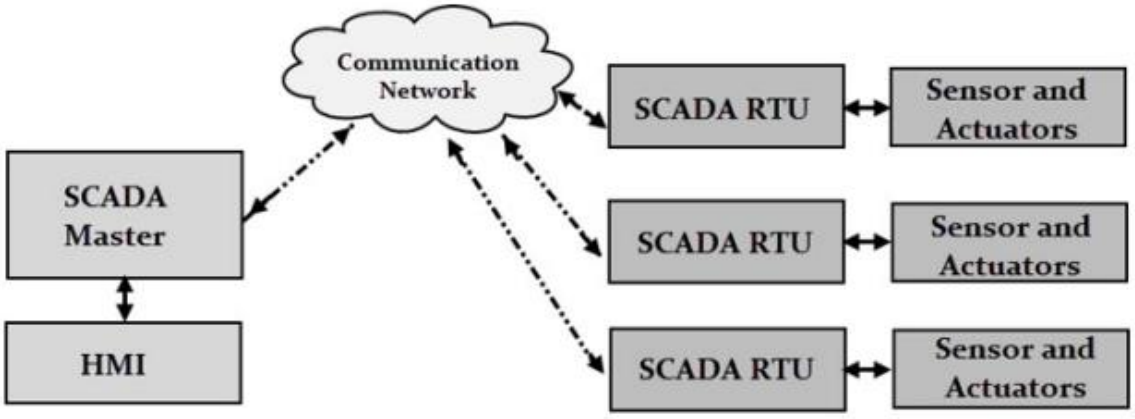
- Industrial process includes those of manufacturing, production, power generation, fabrication and refining, and may run in continuous, batch, repetitive, or discrete modes.
- Infrastructure processes may be public or private, and include water treatment and distribution, wastewater collection and treatment, oil and gas pipelines, electrical power transmission and distribution, wind farms, Civil defense sirens systems, and large communication systems.
- Facility processes occur both in public facilities and private ones, including buildings, airports, ships, and space stations. They monitor and control HVAC access, and energy consumption.
- For machine operation, production data monitoring, generating production reports, Trend monitoring of certain process parameters like temperature, viscosity & Diagnostic system in Factory automation machines.

Components of SCADA system

A SCADA system usually consists of the following subsystems:

- **A Human machine interface or HMI** is the apparatus which presents process data to a human operator and through this, the human operator monitors and controls the process.
- **A supervisory (computer) system**, gathering (acquiring) data on the process and sending commands (control) to the process.
- **Remote terminal Units (RTUs)** connecting to sensors in the process, converting sensor signals to digital data and sending digital data to the supervisory system.
- (**PLCs) used as field devices** because they are more economical, versatile, flexible, and configurable than special-purpose RTUs.
- **Communication infrastructure** connecting the Supervisory system to the Remote terminal units either on Ethernet or RS485 bus or some form of Network communication.

Components of SCADA (Block diagram of SCADA)



Master Station:

It is the heart of SCADA system. It has a dedicated computer in a central location. It monitors and controls the RTUs. The master station consists of Engineering work stations, HMI (Human Machine Interface) stations and large databases (for storing data). The master station performs the following functions:

- Collects and processes information from the RTUs
- Stores collected data on a database
- Provides interface to the operators through HMIs.

Communication equipment:

- Communication is carried out between Master Station and Remote Terminal Units.
- The communication is bidirectional (both to and from).
- It can be wired or wireless. Wired communication can be through twisted pair cables or fiber optic cables or telephone lines. Wireless communication can be using radio signals or satellites.

Remote Terminal Unit (RTUs) :

- These are special units like PLCs which are placed at geographically distributed field sites.
- They are connected with sensors for getting various information like voltage, current, temperature or pressure.
- They are also connected with actuators like pumps, relays or valves
- RTUs collect information from the field and controls the field devices.
- Sometimes they store data in a local data base and waits for instruction from Master Station to send data.

Features of SCADA:

- Data acquisition is done by the Master Station with the help of RTUs
- Display of information in the form of pictures or text is provided on several HMIs
- The SCADA executes supervisory form of control. Control of equipment which are at remote locations is done from the master station
- Alarm Processing – There is facility to alert the operator by informing the place and time of an event.
- Information storage and reports – Data is stored in a temporary data base for 40 days or 12 months. Later it is shifted to a permanent storage device.

It has applications in the following areas:

- Mining industries
- Water distribution system
- Waste water treatment
- Oil and gas pipelines
- Electrical Power distribution In SCADA, data from various remote locations are collected using sensors using RTUs (Remote Terminal Units) and displays the data on HMIs (Human Machine Interfaces) at the Master Station. The data is also recorded on the SCADA database.