SIC1405	LOGIC AND DISTRIBUTED CONTROL SYSTEMS (For EIE and E&C)	L	T	P	Credits	Total Marks
		3	0	0	3	100

COURSE OBJECTIVES

To provide the Fundamentals of PLC, PLC programming concepts, PLC applications

To provide in-depth understanding of DCS, SCADA, and Computer Controlled Systems which are used in automation of various machines, processes and systems in industries

UNIT 1 BASICS OF PROGRAMMABLE LOGIC CONTROLLER (PLC) 9 Hrs.

Definition- overview of PLC systems - Input/ Output modules - Power supplies - I/O slots, General PLC programming procedures - programming on-off outputs, Auxiliary commands and functions - creating ladder diagrams from process control descriptions - Ladder logic for traffic light control system. PLC basic programming - digital logic Gates - Boolean algebra. Basic PLC functions-register basics - timer functions - counter functions.

UNIT 2 PLC INTERMEDIATE FUNCTIONS 11 Hrs.

Arithmetic functions - number comparison functions - skip and MCR functions - data move systems. PLC Advanced intermediate functions- utilizing digital bits - sequencer functions - PLC Advanced functions: alternate-programming languages - operation. PLC-PID functions -PLC installation - trouble shooting and maintenance-controlling a robot - processes with PLC - design of inter locks and alarms using PLC. Use of PC as PLC - Application of PLC - Case Study of Bottle Filling System, Field Bus and HART Protocol

UNIT 3 DISTRIBUTED CONTROL SYSTEMS (DCS) 9 Hrs.

Evolution of DCS - building blocks - different architectures-comparison of architectures- detailed descriptions and functions of local control units - basic elements & functions-operator stations - data highways - redundancy concepts.

UNIT 4 DISTRIBUTED CONTROL SYSTEMS: COMMUNICATION FACILITIES AND APPLICATIONS 9 Hrs.

DCS communication Facilities - communication system requirements - architectural issues - protocol issues - communication system standards - operator interfaces - low level and high level operator interfaces - Operator Displays - Engineering interfaces - low level and high level engineering interfaces - Applications of DCS - oil and gas (extraction, production and storage)

UNIT 5 COMPUTER CONTROLLED SYSTEMS 7 Hrs.

Basic building blocks of Computer controlled systems - SCADA - data acquisition System - supervisory Control - Direct digital Control - software - Velocity algorithm & Position algorithm

Max. 45 Hours

TEXT / REFERENCE BOOKS

1. John Webb, W, Ronald Reis, A.,: “Programmable Logic Controllers Principles and Applications”, 3rd Edition, Prentice hall Inc., New Jersey, 1995.
2. Krishna Kant, “Computer based Industrial Control”, Prentice Hall India. 1997.
3. Lucas, M.P.,: “Distributed Control Systems”, Van Nostrand Reinhold Co., New York, 2nd Edition, 1986
4. Petruzella., “Programmable Logic Controllers” 3rd Edition, Tata McGraw Hill, Newyork.
5. Hughes, T. “Programmable Logic Controllers”, ISA Press 1994. 4th Edition.
6. Mckloni, D.T. “Real Time Control Networks”, ISA Press 1994.

END SEMESTER EXAM QUESTION PAPER PATTERN

Max. Marks : 100 **Exam Duration : 3 Hrs.**

PART A : 10 Questions of 2 marks each-No choice **20 Marks**

PART B : 2 Questions from each unit of internal choice, each carrying 16 marks **80 Marks**

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS
DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT 1 – BASIC OF PROGRAMMABLE LOGIC CONTROLLER – SIC1405

UNIT 1 BASICS OF PROGRAMMABLE LOGIC CONTROLLER (PLC)

Definition- overview of PLC systems - Input/ Output modules - Power supplies - I/O slots, General PLC programming procedures - programming on-off outputs, Auxiliary commands and functions - creating ladder diagrams from process control descriptions - Ladder logic for traffic light control system. PLC basic programming - digital logic Gates - Boolean algebra. Basic PLC functions-register basics - timer functions - counter functions.

UNIT I : PROGRAMMABLE LOGIC CONTROLLER (PLC) BASICS 1.1 1.1 1.1**Definition**

A programmable logic controller (PLC) is a special form of micro-processor-based controller that uses a programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting and arithmetic in order to control machines and processes and are designed to be operated by engineers with perhaps a limited knowledge of computers and computing languages.

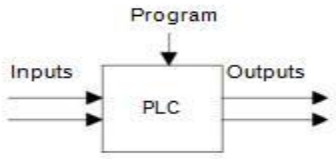


Figure 1.1 A programmable logic controller

PLCs have the great advantage that the same basic controller can be used with a wide range of control systems. To modify a control system and the rules that are to be used, all that is necessary is for an operator to key in a different set of instructions. There is no need to rewire. The result is a flexible, cost effective, system which can be used with control systems which vary quite widely in their nature and complexity.
PLCs are similar to computers but whereas computers are optimised for calculation and display tasks,
PLCs are optimised for control tasks and the industrial environment. Thus PLCs are:

1 Rugged and designed to withstand vibrations, temperature, humidity and noise.
2 Have interfacing for inputs and outputs already inside the controller.
3 Are easily programmed and have an easily understood programming language which is primarily concerned with logic and switching operations.

The first PLC was developed in 1969. They are now widely used and extend from small self-contained units for use with perhaps 20 digital inputs/outputs to modular systems which can be used for large numbers of inputs/outputs, handle digital or analogue inputs/outputs, and also carry out proportional-integral-derivative control modes.

1.2 The PLC System
Typically a PLC system has the basic functional components of processor unit, memory, power supply unit, input/output interface section, communications interface and the programming device. Figure 1.2 shows the basic arrangement.

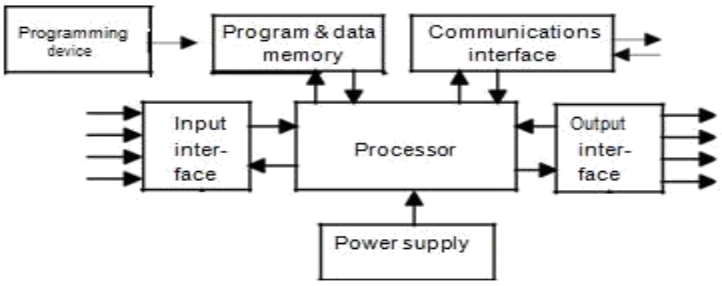


Figure 1.2 The PLC system

1.The processor unit or central processing unit (CPU) is the unit containing the microprocessor and this interprets the input signals and carries out the control actions, according to the program stored in its memory, communicating the decisions as action signals to the outputs.
2. The power supply unit is needed to convert the mains a.c. voltage to the low d.c. voltage (5 V) necessary for the processor and the circuits in the input and output interface modules.
3.The programming device is used to enter the required program into the memory of the processor. The program is developed in the device and then transferred to the memory unit of the PLC.
4. The memory unit is where the program is stored that is to be used for the control actions to be exercised by the microprocessor and data stored from the input for processing and for the output for outputting.
5. The input and output sections are where the processor receives information from external devices and communicates information to external devices. The inputs might thus be from switches or other sensors such as photo-electric cells, temperature sensors, or flow sensors, etc. The outputs might be to motor starter coils, solenoid valves, etc. Input and output devices can be classified as giving signals which are discrete, digital or analogue. Devices giving discrete or digital signals are ones where the signals are either off or on. Thus a switch is a device giving a discrete signal, either no voltage or a voltage. Digital devices can be considered to be essentially discrete devices which give a sequence of on− off signals. Analogue devices give signals whose size is proportional to the size of the variable being monitored. For example, a temperature sensor may give a voltage proportional to the temperature.
6. The communications interface is used to receive and transmit data on communication networks from or to other remote PLCs (Figure 1.3). It is concerned with such actions as device verification, data acquisition, synchronization between user applications and connection management.

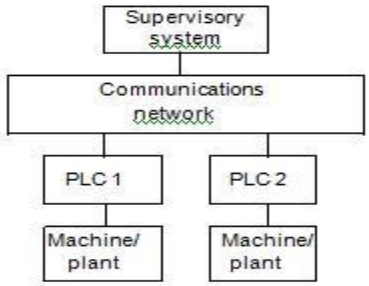


Figure 1.3 Basic communications model

1.3 Internal architecture

Figure 1.4 shows the basic internal architecture of a PLC. It consists of a central processing unit (CPU) containing the system microprocessor, memory, and input/output circuitry. The CPU controls and processes all the operations within the PLC. It is supplied with a clock with a frequency of typically between 1 and 8 MHz. This frequency determines the operating speed of the PLC and provides the timing and synchronisation for all elements in the system. The information within the PLC is carried by means of digital signals. The internal paths along which digital signals flow are called *buses*. In the physical sense, a bus is just a number of conductors along which electrical signals can flow. It might be tracks on a printed circuit board or wires in a ribbon cable. The CPU uses the *data bus* for sending data between the constituent elements, the *address bus* to send the addresses of locations for accessing stored data and the *control bus* for signals relating to internal control actions. The *system bus* is used for communications between the input/output ports and the input/output unit.

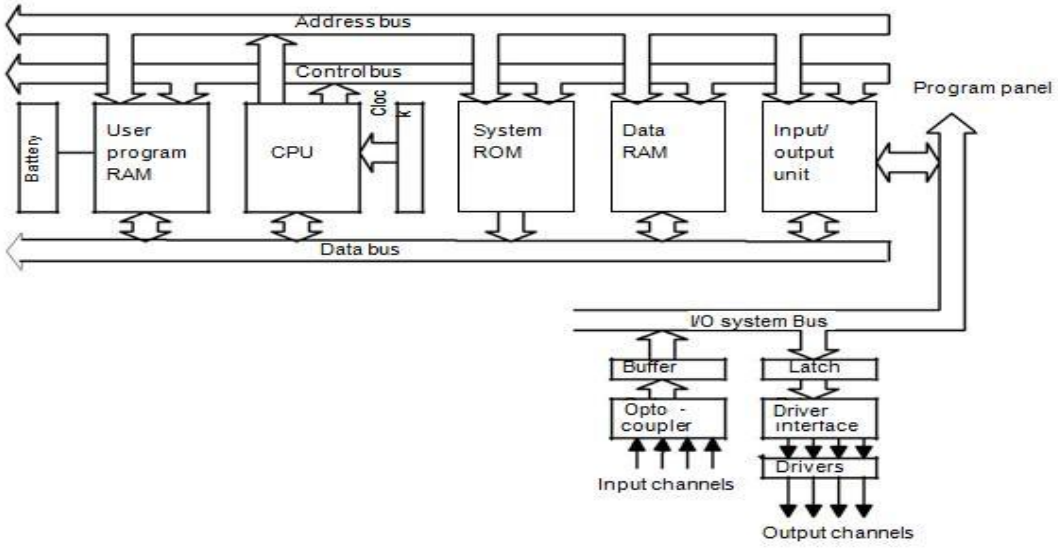


Figure 1.4 *Architecture of a PLC*

The CPU

The internal structure of the CPU depends on the microprocessor concerned. In general they have:

1. An arithmetic and logic unit (ALU) which is responsible for data manipulation and carrying out arithmetic operations of addition and subtraction and logic operations of AND, OR, NOT and EXCLUSIVE-OR.
2. Memory, termed registers, located within the microprocessor and used to store information involved in program execution.
3. A control unit which is used to control the timing of operations.

The buses

The buses are the paths used for communication within the PLC. The information is transmitted in binary form, i.e. as a group of *bits* with a bit being a binary digit of 1 or 0, i.e. on/off states. The term *word* is used for the group of bits constituting some information. Thus an 8 -bit word might be the binary number 00100110. Each of the bits is communicated simultaneously along its own parallel wire. The system has four buses:

1. The *data bus* carries the data used in the processing carried out by the CPU. A microprocessor termed as being 8-bit has an internal data bus which can handle 8-bit numbers. It can thus perform operations between 8-bit numbers and deliver results as 8-bit values.
2. The *address bus* is used to carry the addresses of memory locations. So that each word can be located in the memory, every memory location is given a unique *address*. Just like houses in a town are each given a distinct address so that they can be located, so each word location is given an address so that data stored at a particular location can be accessed by the CPU either to read data located there or put, i.e. write, data there. It is the address bus which carries the information indicating which address is to be accessed. If the address bus consists of 8 lines, the number of 8-bit words, and hence number of distinct addresses, is 28 = 256. With 16 address lines, 65 536 addresses are possible.
3. The *control bus* carries the signals used by the CPU for control, e.g. to inform memory devices whether they are to receive data from an input or output data and to carry timing signals used to synchronise actions.
4. The *system bus* is used for communications between the input/output ports and the input/output unit.

Memory

There are several memory elements in a PLC system:

1 System read-only-memory (ROM) to give permanent storage for the operating system and fixed data used by the CPU.
2 Random-access memory (RAM) for the user's program.
3 Random-access memory (RAM) for data. This is where information is stored on the status of input and output devices and the values of timers and counters and other internal devices. The data RAM is sometimes referred to as a data table or register table. Part of this memory, i.e. a block of addresses, will be set aside for input and output addresses and the states of those inputs and outputs. Part will be set aside for preset data and part for storing counter values, timer values, etc.
4 Possibly, as a bolt-on extra module, erasable and programmable read-only-memory (EPROM) for ROMs that can be programmed and then the program made permanent.

The programs and data in RAM can be changed by the user. All PLCs will have some amount of RAM to store programs that have been developed by the user and program data. However, to prevent the loss of programs when the power supply is switched off, a battery is used in the PLC

to maintain the RAM contents for a period of time. After a program has been developed in RAM it may be loaded into an EPROM memory chip, often a bolt-on module to the PLC, and so made permanent. In addition there are temporary buffer stores for the input/output channels. The storage capacity of a memory unit is determined by the number of binary words that it can store. Thus, if a memory size is 256 words then it can store 256* 8 = 2048 bits if 8-bit words are used and 256 □ 16 = 4096 bits if 16-bit words are used. Memory sizes are often specified in terms of the number of storage locations available with 1K representing the number 210, i.e. 1024. Manufacturers supply memory chips with the storage locations grouped in groups of 1, 4 and 8 bits. A 4K % 1 memory has 4 % 1 % 1024 bit locations. A 4K % 8 memory has 4 % 8 % 1024 bit locations. The term byte is used for a word of length 8 bits. Thus the 4K % 8 memory can store 4096 bytes. With a 16-bit address bus we can have 2 16 different addresses and so, with 8-bit words stored at each address, we can have 2 16 % 8 storage locations and so use a memory of size 216 % 8/210 = 64K % 8 which we might be as four 16K % 8 bit memory chips.

Input/output unit

The input/output unit provides the interface between the system and the outside world, allowing for connections to be made through input/output channels to input devices such as sensors and output devices such as motors and solenoids. It is also through the input/output unit that programs are entered from a program panel. Every input/output point has a unique address which can be used by the CPU. It is like a row of houses along a road, number 10 might be the 'house' to be used for an input from a particular sensor while number '45' might be the 'house' to be used for the output to a particular motor.The input/output channels provide isolation and signal conditioning functions so that sensors and actuators can often be directly connected to them without the need for other circuitry. Electrical isolation from the external world is usually by means of optoisolators (the term optocoupler is also often used). Figure 1.5 shows the principle of an optoisolator. When a digital pulse passes through the light-emitting diode, a pulse of infrared radiation is produced. This pulse is detected by the phototransistor and gives rise to a voltage in that circuit. The gap between the light-emitting diode and the phototransistor gives electrical isolation but the arrangement still allows for a digital pulse in one circuit to give rise to a digital pulse in another circuit.

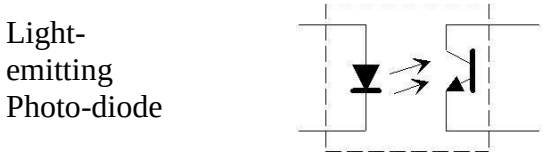


Figure 1.5 *Optoisolator*

The digital signal that is generally compatible with the microprocessor in the PLC is 5 V d.c. However, signal conditioning in the input channel, with isolation, enables a wide range of input signals to be supplied to it. A range of inputs might be available with a larger PLC, e.g. 5 V, 24 V, 110 V and 240 V digital/discrete, i.e. on− off, signals (Figure 1.6). A small PLC is likely to have just one form of input, e.g. 24 V.

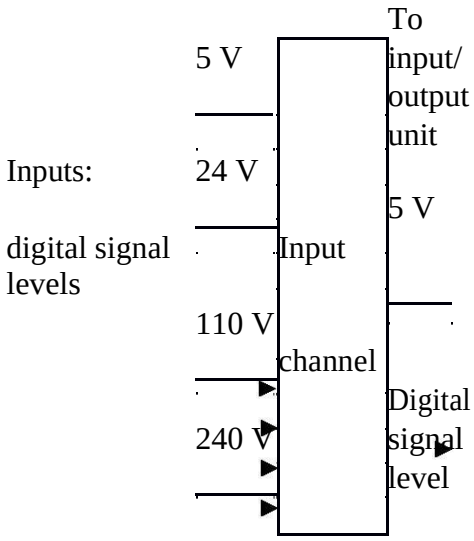


Figure 1.6 *Input levels*

The output from the input/output unit will be digital with a level of 5 V. However, after signal conditioning with relays, transistors or triacs, the output from the output channel might be a 24 V, 100 mA switching signal, a d.c. voltage of 110 V, 1 A or perhaps 240 V, 1 A a.c., or 240 V, 2 A a.c., from a triac output channel (Figure 1.7). With a small PLC, all the outputs might be of one type, e.g. 240 V a.c., 1 A. With modular PLCs, however, a range of outputs can be accommodated by selection of the modules to be used.

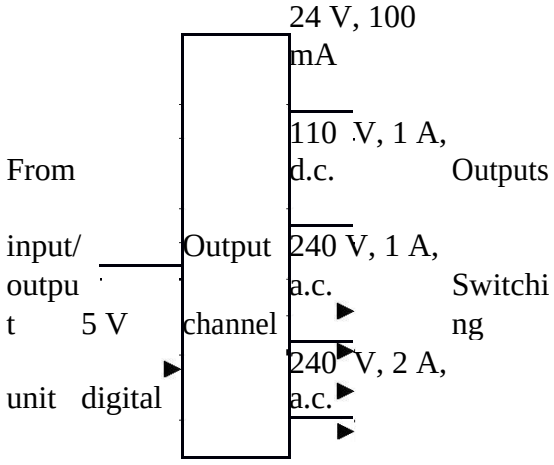


Figure 1.7 *Output levels*

Outputs are specified as being of relay type, transistor type or triac type

1. With the *relay type*, the signal from the PLC output is used to operate a relay and is able to switch currents of the order of a few amperes in an external circuit. The relay not only allows small currents to switch much larger currents but also isolates the PLC from the external circuit.

Relays are, however, relatively slow to operate. Relay outputs are suitable for a.c. and d.c. switching. They can withstand high surge currents and voltage transients.

2 The *transistor type* of output uses a transistor to switch current through the external circuit. This gives
a considerably faster switching action. It is, however, strictly for d.c. switching and is destroyed by overcurrent and high reverse voltage. As a protection, either a fuse or built-in electronic protection are used. Optoisolators are used to provide isolation.

3 *Triac* outputs, with optoisolators for isolation, can be used to control external loads which are connected to the a.c. power supply. It is strictly for a.c. operation and is very easily destroyed by overcurrent. Fuses are virtually always included to protect such outputs.

Sourcing and sinking

The terms *sourcing* and *sinking* are used to describe the way in which d.c. devices are connected to a PLC. With sourcing, using the conventional current flow direction as from positive to negative, an input device receives current from the input module, i.e. the input module is the source of the current (Figure 1.8(a)). If the current flows from the output module to an output load then the output module is referred to as sourcing (Figure 1.8(b)). With sinking, using the conventional current flow direction as from positive to negative, an input device supplies current to the input module, i.e. the input module is the sink for the current (Figure 1.9(a)) . If the current flows to the output module from an output load then the output module is referred to as sinking (Figure 1.9(b)).

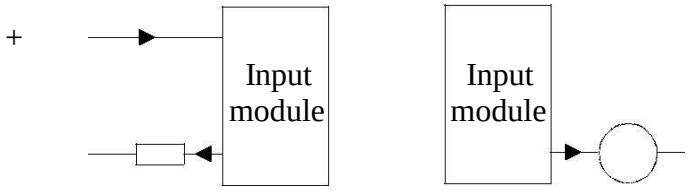
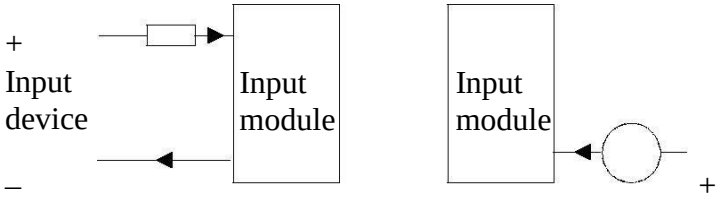


Figure 1.8 *Sourcing*



(a) (b) Output load

Figure 1.9 *Sinking*

1.4 Programming Procedure

As an introduction to ladder diagrams, consider the simple wiring diagram for an electrical circuit in Figure 1.10(a) . The diagram shows the circuit for switching on or off an electric motor. We can redraw this diagram in a different way, using two vertical lines to represent the input power rails and stringing the rest of the circuit between them. Figure 1.10(b) shows the result. Both circuits have the switch in series with the motor and supplied with electrical power when the switch is closed. The circuit shown in Figure 1.10(b) is termed a *ladder diagram*.

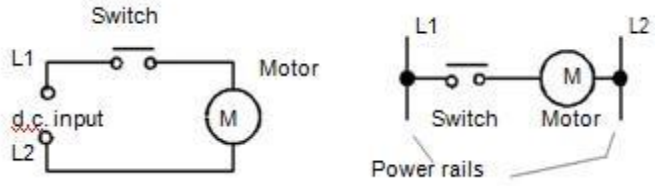


Figure 1.10 *Ways of drawing the same electrical circuit*

With such a diagram the power supply for the circuits is always shown as two vertical lines with the rest of the circuit as horizontal lines. The power lines, or rails as they are often termed, are like the vertical sides of a ladder with the horizontal circuit lines like the rungs of the ladder. The horizontal rungs show only the control portion of the circuit, in the case of Figure 1.10 it is just the switch in series with the motor. Circuit diagrams often show the relative physical location of the circuit components and how they are actually wired. With ladder diagrams no attempt is made to show the actual physical locations and the emphasis is on clearly showing how the control is exercised.

Figure 1.11 shows an example of a ladder diagram for a circuit that is used to start and stop a motor using push buttons. In the normal state, push button 1 is open and push button 2 closed. When button 1 is pressed, the motor circuit is completed and the motor starts. Also, the holding contacts wired in parallel with the motor close and remain closed as long as the motor is running. Thus when the push button 1 is

released, the holding contacts maintain the circuit and hence the power to the motor. To stop the motor, button 2 is pressed. This disconnects the power to the motor and the holding contacts open. Thus when push button 2 is released, there is still no power to the motor. Thus we have a motor which is started by pressing button 1 and stopped by pressing button 2.

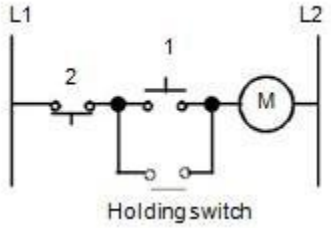


Figure 1.11 *Stop-start switch*

1.5 PLC ladder programming

A very commonly used method of programming PLCs is based on the use of *ladder diagrams*. Writing a program is then equivalent to drawing a switching circuit. The ladder diagram consists of two vertical lines representing the power rails. Circuits are connected as horizontal lines, i.e. the rungs of the ladder, between these two verticals.

In drawing a ladder diagram, certain conventions are adopted:

1 The vertical lines of the diagram represent the power rails between which circuits are connected.

2 Each rung on the ladder defines one operation in the control process.

3 A ladder diagram is read from left to right and from top to bottom, Figure 1.12 showing the scanning motion employed by the PLC. The top rung is read from left to right. Then the second rung down is read from left to right and so on. When the PLC is in its run mode, it goes through the entire ladder program to the end, the end rung of the program being clearly denoted, and then promptly resumes at the start This procedure of going through all the rungs of the program is termed a cycle. The end rung might be indicated by a block with the word END or RET for return, since the program promptly returns to its beginning.

4 Each rung must start with an input or inputs and must end with at least one output. The term input is used for a control action, such as closing the contacts of a switch, used as an input to the PLC. The term output is used for a device connected to the output of a PLC, e.g. a motor.

5 Electrical devices are shown in their normal condition. Thus a switch which is normally open until some object closes it, is shown as open on the ladder diagram. A switch that is normally closed is shown closed.

6 A particular device can appear in more than one rung of a ladder. For example, we might have arelay which switches on one or more devices. The same letters and/or numbers are used to label the device in each situation.

7 The inputs and outputs are all identified by their addresses, the notation used depending on the PLC manufacturer. This is the address of the input or output in the memory of the PLC

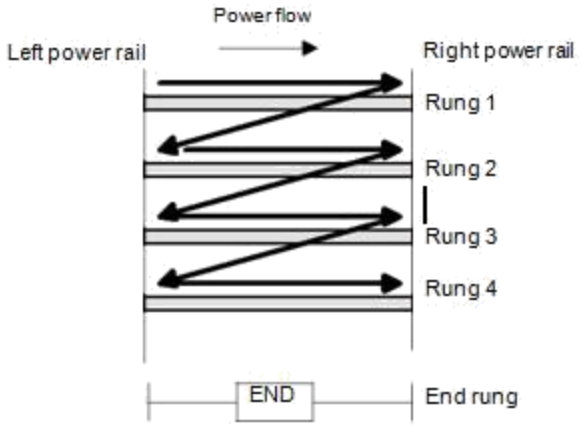


Figure 1.12 *Scanning the ladder program*

Figure 1.13 shows standard IEC 1131-3 symbols that are used for input and output devices. Some slight variations occur between the symbols when used in semi-graphic form and when in full graphic. Note that inputs are represented by different symbols representing normally open or normally closed contacts. The action of the input is equivalent to opening or closing a switch. Output coils are represented by just one form of symbol. Further symbols will be introduced in later chapters.

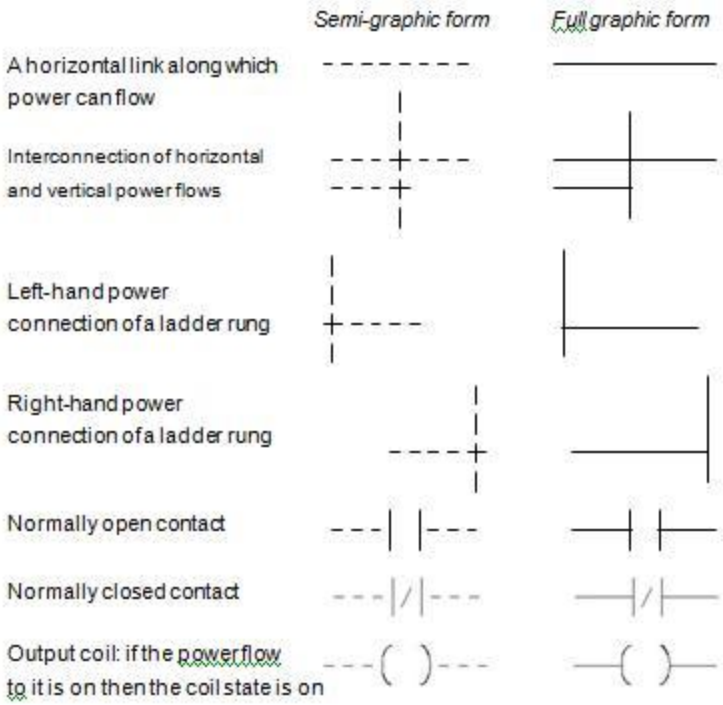


Figure 1.13 *Basic symbols*

To illustrate the drawing of the rung of a ladder diagram, consider a situation where the energising of an output device, e.g. a motor, depends on a normally open start switch being activated by being closed. The input is thus the switch and the output the motor. Figure 1.14(a) shows the ladder diagram.

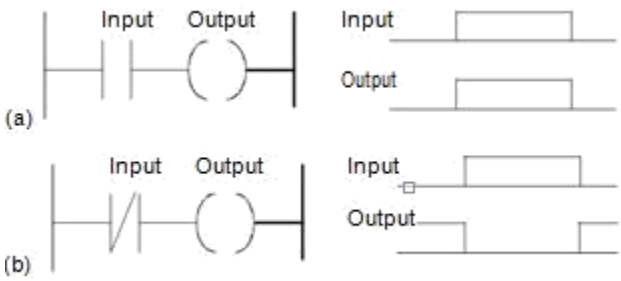


Figure 1.14 *A ladder rung*

Starting with the input, we have the normally open symbol | | for the input contacts. There are no other input devices and the line terminates with the output, denoted by the symbol ( ). When the switch is closed, i.e. there is an input, the output of the motor is activated. Only while there is an input to the contacts is there an output. If there had been a normally closed switch |/| with the output (Figure 1.14(b)), then there would have been an output until that switch was opened. Only while there is no input to the contacts is there an output.

In drawing ladder diagrams the names of the associated variable or addresses of each element are appended to its symbol. Thus Figure 1.15 shows how the ladder diagram of Figure 1.14(a) would appear using (a) Mitsubishi, (b) Siemens, (c) Allen-Bradley, (d) Telemecanique notations for the addresses. Thus Figure 1.15(a) indicates that this rung of the ladder program has an input from address X400 and an output to address Y430. When wiring up the inputs and outputs to the PLC, the relevant ones must be connected to the input and output terminals with these addresses.

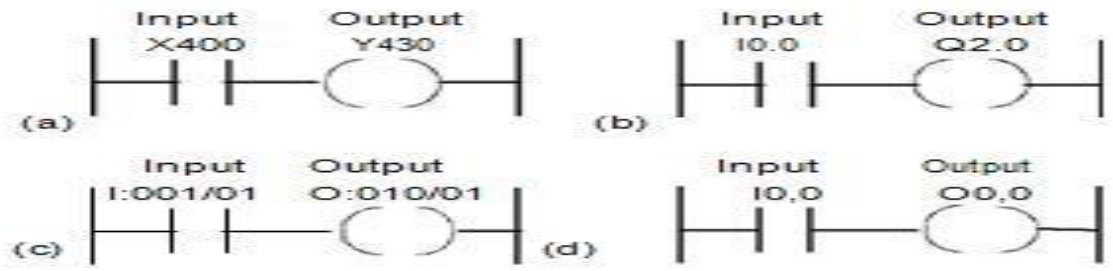


Figure 1.15 *Notation: (a) Mitsubishi, (b) Siemens, (c) Allen-Bradley, (d) Telemecanique*

1.6 Logic functions

There are many control situations requiring actions to be initiated when a certain combination of conditions is realised. Thus, for an automatic drilling machine , there might be the condition that the drill motor is to be activated when the limit switches are activated that indicate the presence of the workpiece and the drill position as being at the surface of the workpiece. Such a situation involves the AND logic function, condition A and condition B having both to be realised for an output to occur. This section is a consideration of such logic functions.

1.6.1 AND

Figure 1.16(a) shows a situation where an output is not energised unless two, normally open, switches are both closed. Switch A and switch B have both to be closed, which thus gives an AND logic situation. We can think of this as representing a control system with two inputs A and B (Figure 1.16(b)). Only when A and B are both on is there an output. Thus if we use 1 to indicate an on signal and 0 to represent an off signal, then for there to be a 1 output we must have A and B both 1. Such an operation is said to be controlled by a logic gate and the relationship between the inputs to a logic gate and the outputs is tabulated in a form known as a truth table. Thus for the AND gate we have:

Inputs		Output
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

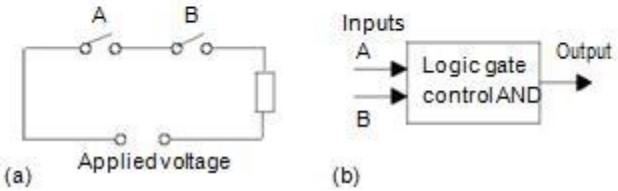


Figure 1.16 *(a) AND circuit, (b) AND logic gate*

An example of an AND gate is an interlock control system for a machine tool so that it can only be operated when the safety guard is in position and the power switched on.
Figure 1.17(a) shows an AND gate system on a ladder diagram. The ladder diagram starts with | |, a normally open set of contacts labelled input A, to represent switch A and in series with it | |, another normally open set of contacts labelled input B, to represent switch B. The line then terminates with O to represent the output. For there to be an output, both input A and input B have to occur, i.e. input A and input B contacts have to be closed (Figure 1.17(b)). In general:
On a ladder diagram contacts in a horizontal rung, i.e. contacts in series, represent the logical

AND operations.

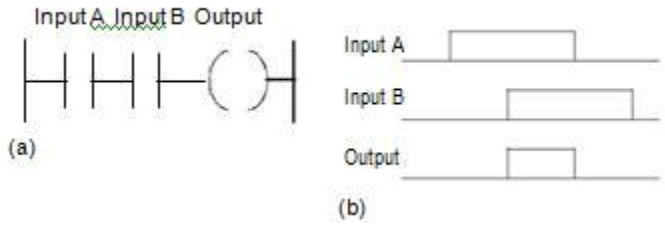


Figure 1.17 *AND gate with a ladder diagram rung*

1.6.2 OR

Figure 1.18(a) shows an electrical circuit where an output is energised when switch A or B, both normally open, are closed. This describes an OR logic gate (Figure 1.18(b)) in that input A or input B. must be on for there to be an output. The truth table is:

Inputs		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	1

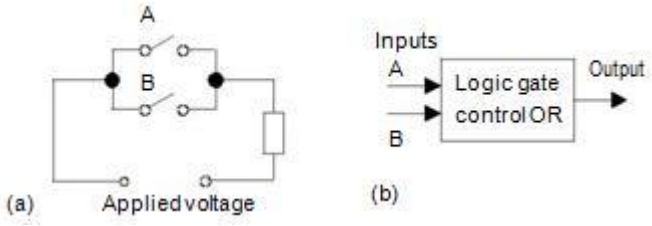


Figure 1.18 *(a) OR electrical circuit, (b) OR logic gate*

Figure 1.19(a) shows an OR logic gate system on a ladder diagram. The ladder diagram starts with | |, normally open contacts labelled input A, to represent switch A and in parallel with it | |, normally open contacts labelled input B, to represent switch B. Either input A or input B have to be closed for the output to be energised (Figure 1.19(b)). The line then terminates with O to represent the output. In general:

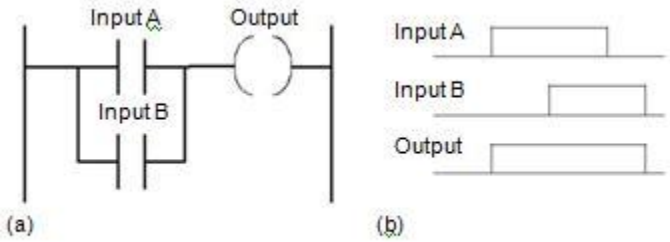


Figure 1.19 *OR gate*

1.6.3 NOT

Figure 1.20(a) shows an electrical circuit controlled by a switch that is normally closed. When there is an input to the switch, it opens and there is then no current in the circuit. This illustrates a NOT gate in that there is an output when there is no input and no output when there is an input (Figure 1.20(c)). The gate is sometimes referred to as an *inverter*.

The truth table is:

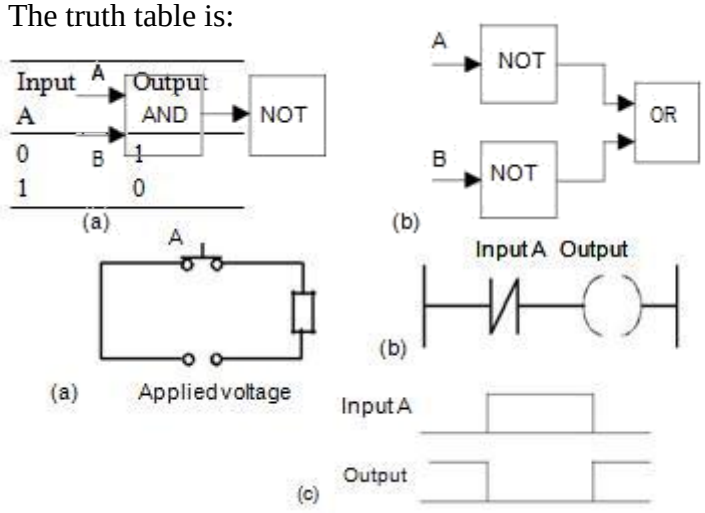


Figure 1.20 *(a) NOT circuit, (b) NOT logic with a ladder rung, (c) high output when no input to A*

Figure 1.20(b) shows a NOT gate system on a ladder diagram. The input A contacts are shown as being normally closed. This is in series with the output ( ). With no input to input A, the contacts are closed and so there is an output. When there is an input to input A, it opens and there is then no output.

An example of a NOT gate control system is a light that comes on when it becomes dark, i.e. when there is no light input to the light sensor there is an output.

1.6.4 NAND

Suppose we follow an AND gate with a NOT gate (Figure 1.21(a)). The consequence of having the NOT gate is to invert all the outputs from the AND gate. An alternative, which gives exactly the same results, is to put a NOT gate on each input and then follow that with OR (Figure 1.21(b)). The same truth table occurs, namely:

Inputs		Output
A	B	
0	0	1
0	1	1
1	0	1
1	1	0

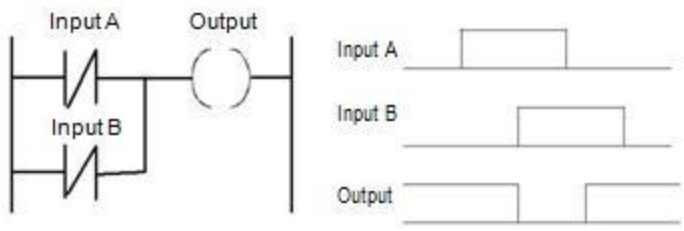


Figure 1.21 *NAND gate*

Both the inputs A and B have to be 0 for there to be a 1 output. There is an output when input A and input B are not 1. The combination of these gates is termed a NAND gate.

Figure 1.21 shows a ladder diagram which gives a NAND gate. When the inputs to input A and input B are both 0 then the output is 1. When the inputs to input A and input B are both 1, or one is 0 and the other 1, then the output is 0. An example of a NAND gate control system is a warning light that comes on if, with a machine tool, the safety guard switch has not been activated and the limit switch signalling the presence of the workpiece has not been activated. Suppose we follow an OR gate by a NOT gate (Figure 1.22(a)). The consequence of having the NOT gate is to invert the outputs of the OR gate. An alternative, which gives exactly the same results, is to put a NOT gate on each input and then an AND gate for the resulting inverted inputs (Figure 1.22(b)). The following is the resulting truth table:

Inputs		Output
A	B	
0	0	1
0	1	0
1	0	0
1	1	0

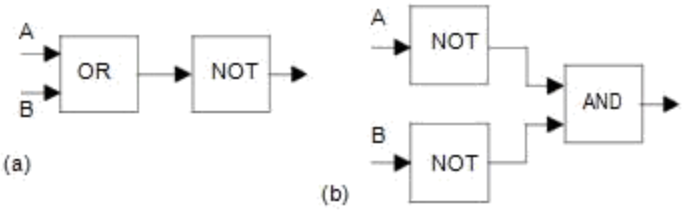


Figure 1.22 *NOR gate*

The combination of OR and NOT gates is termed a NOR gate. There is an output when neither input A or input B is 1.

Figure 1.23 shows a ladder diagram of a NOR system. When input A and input B are both not activated, there is a 1 output. When either X400 or X401 are 1 there is a 0 output.

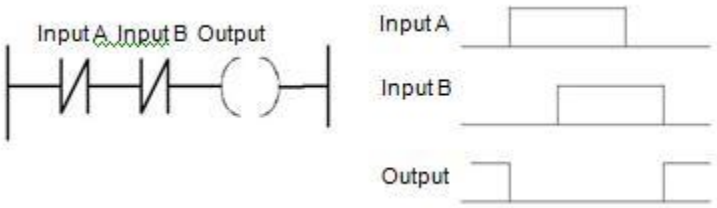


Figure 1.23 *NOR gate*

1.6.5 Exclusive OR (XOR)

The OR gate gives an output when either or both of the inputs are 1. Sometimes there is, however, a need for a gate that gives an output when either of the inputs is 1 but not when both are 1, i.e. has the truth table:

Such a gate is called an *Exclusive* OR or XOR gate. One way of obtaining such a gate is by using NOT, AND and OR gates as shown in Figure 1.24

Inputs		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

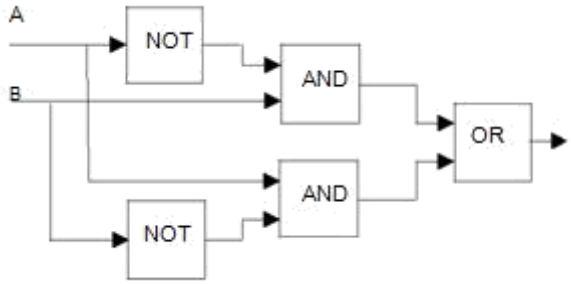


Figure 1.24 *XOR gate*

Figure 1.25 shows a ladder diagram for an XOR gate system. When input A and input B are not activated then there is 0 output. When just input A is activated, then the upper branch results in the output being 1. When just input B is activated, then the lower branch results in the output being 1. When both input A and input B are activated, there is no output. In this example of a logic gate, input A and input B have two sets of contacts in the circuits, one set being normally open and the other normally closed. With PLC programming, each input may have as many sets of contacts as necessary.

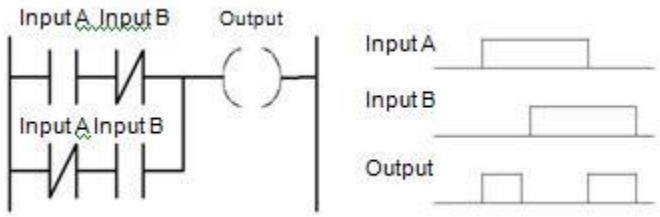


Figure 1.25 *XOR gate*

1.7 Latching

There are often situations where it is necessary to hold an output energised, even when the input ceases. A simple example of such a situation is a motor which is started by pressing a push button switch. Though the switch contacts do not remain closed, the motor is required to continue running until a stop push button switch is pressed. The term latch circuit is used for the circuit used to carry out such an operation. It is a self-maintaining circuit in that, after being energised, it maintains that state until another input is received.

An example of a latch circuit is shown in Figure1.26 When the input A contacts close, there is an output. However, when there is an output, another set of contacts associated with the output closes. These contacts form an OR logic gate system with the input contacts. Thus, even if the

input A opens, the circuit will still maintain the output energised. The only way to release the output is by operating the normally closed contact B.

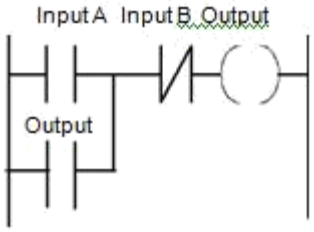


Figure 1.26 *Latched circuit*

1.8 **Timers**

In many control tasks there is a need to control time. For example, a motor or a pump might need to be controlled to operate for a particular interval of time, or perhaps be switched on after some time interval. PLCs thus have timers as built-in devices. Timers count fractions of seconds or seconds using the internal CPU clock.

PLC manufacturers differ on how timers should be programmed and hence how they can be considered. A common approach is to consider timers to behave like relays with coils which when energised result in the closure or opening of contacts after some preset time. The timer is thus treated as an output for a rung with control being exercised over pairs of contacts elsewhere (Figure 1.27(a)). This is the predominant approach used in this book. Some treat a timer as a delay block which when inserted in a rung delays signals in that rung reaching the output (Figure 1.27(b)).

There are a number of different forms of timers that can be found with PLCs. With small PLCs there is likely to be just one form, the on -delay timers. These are timers which come on after a particular time delay (Figure 1.28 (a)). Off-delay timers are on for a fixed period of time before turning off (Figure 1.28(b)). Another type of timer that occurs is the pulse timer. This timer switches on or off for a fixed period of time (Figure 1.28(c)). Figure 1.29 shows the IEC 1131-3 standard symbols for timers. TON is used to denote on-delay, TOF off-delay. TP pulse timers. On-delay is also represented by T− 0 and off-delay by 0− T. The time duration for which a timer has been set is termed the preset and is set in multiples of the time base used. Some time bases are typically 10 ms, 100 ms, 1 s, 10 s and 100 s. Thus a preset value of 5 with a time base of 100 ms is a time of 500 ms. For convenience, where timers are involved in this text, a time base of 1 s has been used.

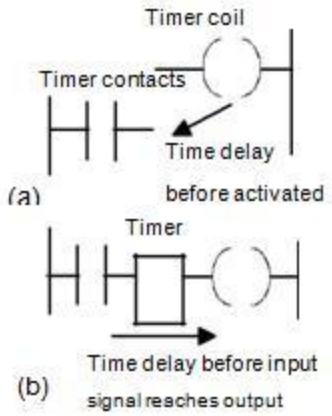


Figure 1.27 *Treatment of timers*

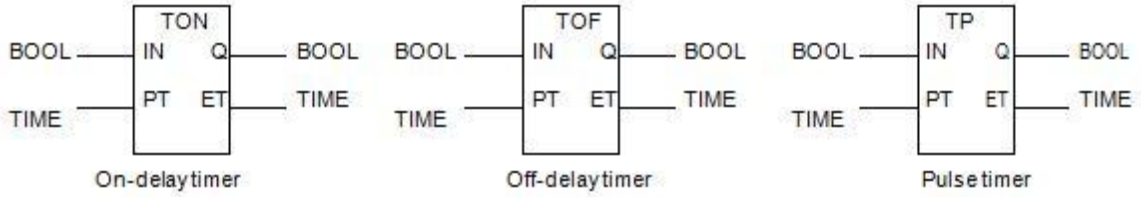


Fig1.28 *Timers: (a) on-delay, (b) off-delay, (c) pulse*

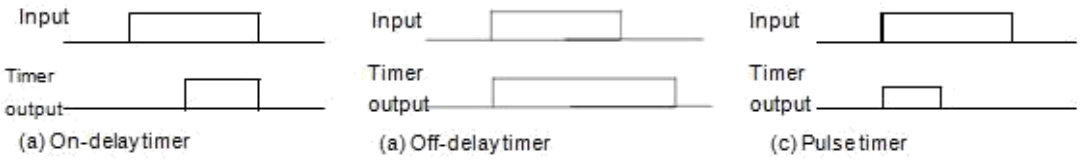


Figure 1.29 *IEC 1131-1 standards. BOOL indicates a Boolean input/output, i.e. on/off. IN is the input. Q is the output. ET is the elapsed time output. PT is the input used to specify the time.*

1.8 Programming timers

All PLCs generally have delay-on timers, small PLCs possibly having only this type of timer. Figure 1.30(a) shows a ladder rung diagram involving a delay-on timer. Figure 1.30(a) is typical of Mitsubishi. The timer is like a relay with a coil which is energised when the input In 1 occurs (rung 1). It then closes, after some preset time delay, its contacts on rung 2. Thus the output occurs some preset time after the input In 1 occurs. Figure 1.30(b) shows the timer to be a delay item in a rung, rather than as a relay, the example being for Siemens. When the signal at the timer's start input changes from 0 to 1, the timer starts and runs for the programmed duration, giving its output then to the output coil. The time value (TV) output can be used to ascertain the amount of time remaining at any instant. A signal input of 1 at the reset input resets the timer whether it is running or not. Techniques for the entry of preset time values vary. Often it requires the entry of a constant K command followed by the time interval in multiples of the time base used.

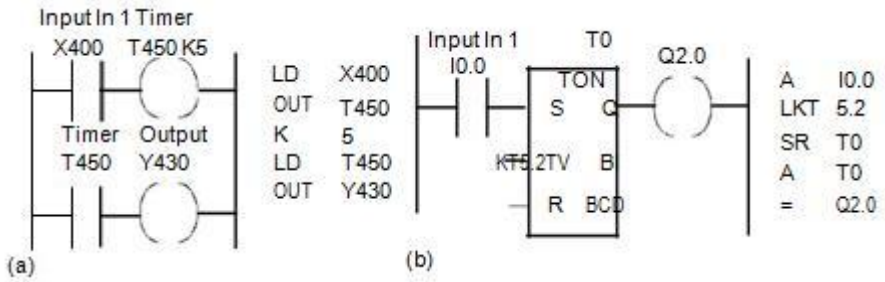


Figure 1.30 *Timers: (a) Mitsubishi, (b) Siemens*

Sequencing

As an illustration of the use of a timer, consider the ladder diagram shown in Figure 1.31(a). When the input In 1 is on, the output Out 1 is switched on. The contacts associated with this output then start the timer. The contacts of the timer will close after the preset time delay, in this case 5.5 s. When this happens, output Out 2 is switched on. Thus, following the input In 1, Out 1 is switched on and followed 5.5 s later by Out 2. This illustrates how timed sequence of outputs can be achieved. Figure 1.31(b) shows the same operation where the format used by the PLC manufacturer is for the timer to institute a signal delay.

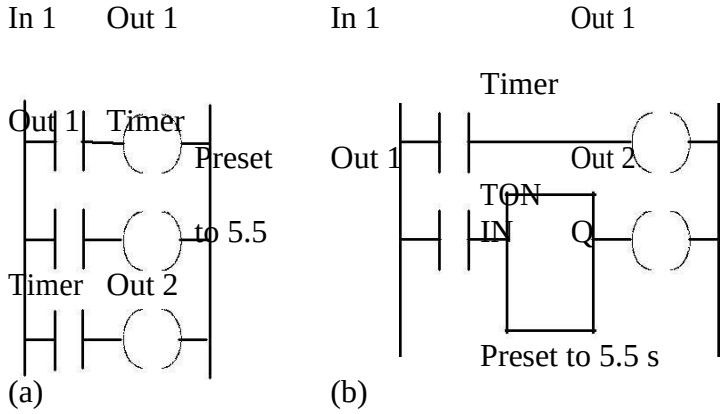


Figure 1.31 *Sequenced outputs*

Figure 1.32 shows two versions of how timers can be used to start three outputs, e.g. three motors, in sequence following a single start button being pressed. In (a) the timers are programmed as coils, whereas in (b) they are programmed as delays. When the start push button is pressed there is an output from internal relay IR1. This latches the start input. It also starts both the timers, T1 and T2, and motor 1. When the preset time for timer 1 has elapsed then its contacts close and motor 2 starts. When the preset time for timer 2 has elapsed then its contacts close and motor 3 starts. The three motors are all stopped by pressing the stop push button. Since this is seen as a complete program, the end instruction has been used.

Cascaded timers

Timers can be linked together, the term *cascaded* is used, to give longer delay times than are possible with just one timer. Figure 1.33(a) shows the ladder diagram for such an arrangement. Thus we might have timer 1 with a delay time of 999 s. This timer is started when there is an input to In 1. When the 999 s time is up, the contacts for timer 1 close. This then starts timer 2. This has a delay of 100 s. When this time is up, the timer 2 contacts close and there is an output from Out 1. Thus the output occurs 1099 s after the input to In 1. Figure 1.33(b) shows the Mitsubishi version of this ladder diagram and the program instructions for that ladder.

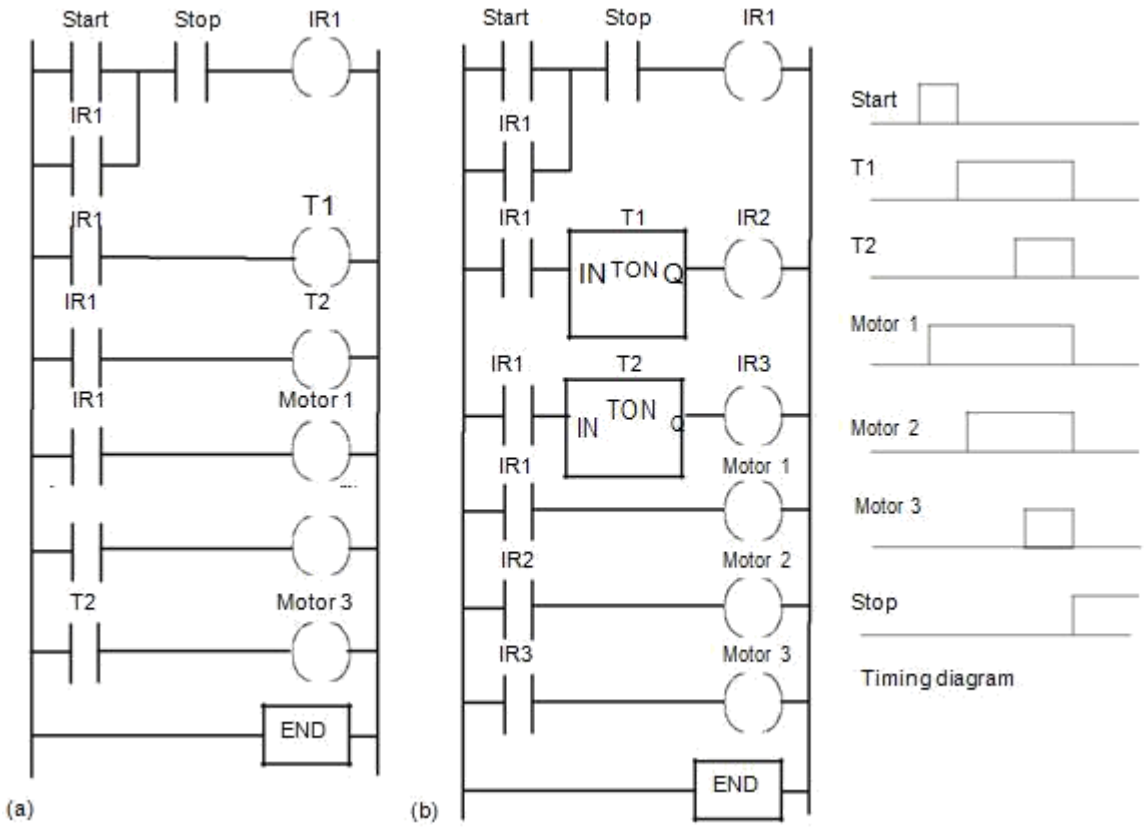


Figure 1.32 *Motor sequence*

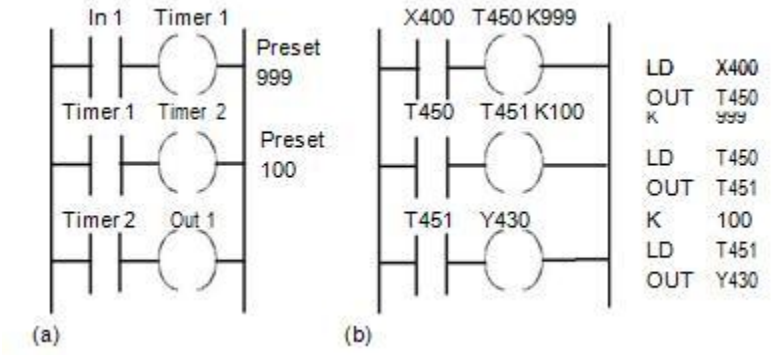


Figure 1.33 *Cascaded timers*

1.8.1 On-off cycle timer

Figure 1.34shows how on-delay timers can be used to produce an *on -off cycle timer*. The timer is designed to switch on an output for 5 s, then off for 5 s, then on for 5 s, then off for 5 s, and so on. When there is an input to In 1 and its contacts close, timer 1 starts. Timer 1 is set for a delay of 5 s. After 5 s, it switches on timer 2 and the output Out 1. Timer 2 has a delay of 5 s. After 5 s, the contacts for timer 2, which are normally closed, open. This results in timer 1, in the first rung, being switched off. This then causes its contacts in the second rung to open and switch off timer

2. This results in the timer 2 contacts resuming their normally closed state and so the input to In 1 causes the cycle to start all over again.

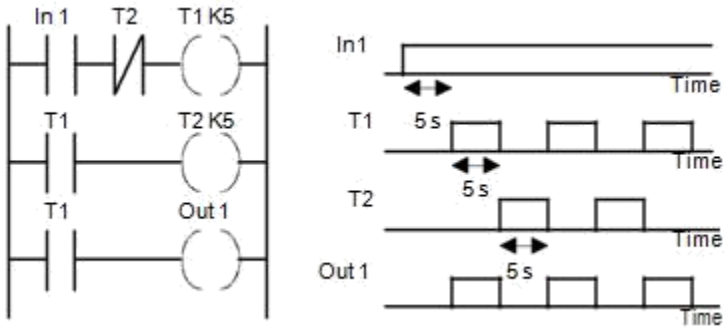


Figure 1.34 *On-off cycle timer*

Figure 1.35 shows how the above ladder program would appear in the format used with a timer considered as a delay, rather than as a coil. This might, for example, be with Siemens or Toshiba. When input In 1 closes, the timer T1 starts. After its preset time, there is an output to Out 1 and timer T2 starts. After its preset time there is an output to the internal relay IR1. This opens its contacts and stops the output from Out 1. This then switches off timer T2. The entire cycle can then repeat itself.

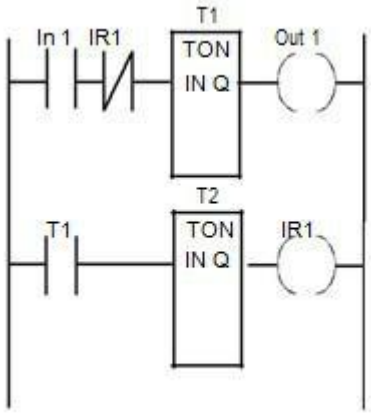


Figure 1.35 *On–off cycle timer*

Off-delay timers

Figure 1.36 shows how a on-delay timer can be used to produce an *off-delay timer*. With such an arrangement, when there is a momentary input to In 1, both the output Out 1 and the timer are switched on. Because the input is latched by the Out 1 contacts, the output remains on. After the preset timer time delay, the timer contacts, which are normally closed, open and switch off the output. Thus the output starts as on and remains on until the time delay has elapsed. Some PLCs have, as well as on-delay timers, built-in off-delay timers and thus there is no need to use an on-delay timer to produce an off-delay timer. Figure 1.37 illustrates this for a Siemens PLC, giving the ladder diagram and the instruction list. Note that with this manufacturer, the timer is considered to be a delay item in a rung, rather than as a relay. In the rectangle symbol used for the timer, the 0 precedes the T and indicates that it is an on-delay timer. As an illustration of the

use of an off-delay timer, consider the Allen-Bradley program shown in Figure 1.38. TOF is used to indicate that it is an off-delay, rather than on-delay (TON) timer. The time base is set to 1:0 which is 1 s. The preset is 10 so the timer is preset to 10 s. In the first rung, the output of the timer is taken from the EN (for enable) contacts. This means that there is no time delay between an input to I:012/01 and the EN output. As a result the EN contacts in rung 2 close immediately there is an I:012/01 input. Thus there is an output from O:013/01 immediately the input I:012/01 occurs. The TT (for timer timing) contacts in rung 3 are energised just while the timer is running. Because the timer is an off-delay timer, the timer is turned on for 10 s before turning off. Thus the TT contacts will close when the set time of 10 s is running. Hence output O:012/02 is switched on for this time of 10 s. The DN (for done) contacts which are normally closed, open after the 10 s and so output O:013/03 comes on after 10 s. The DN contacts which are normally open, close after 10 s and so output O:013/04 goes off after 10 s.

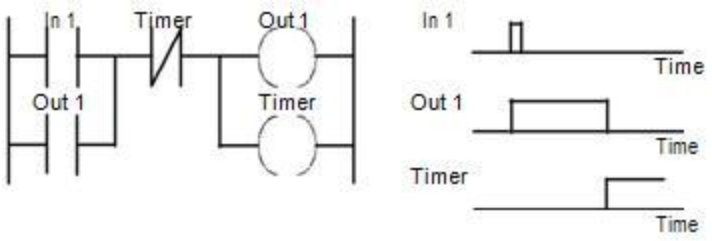


Figure 1.36 *Off-delay timer*

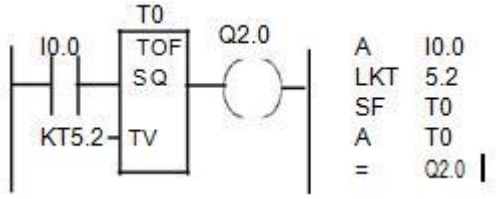


Figure 1.37 *Off-delay timer*

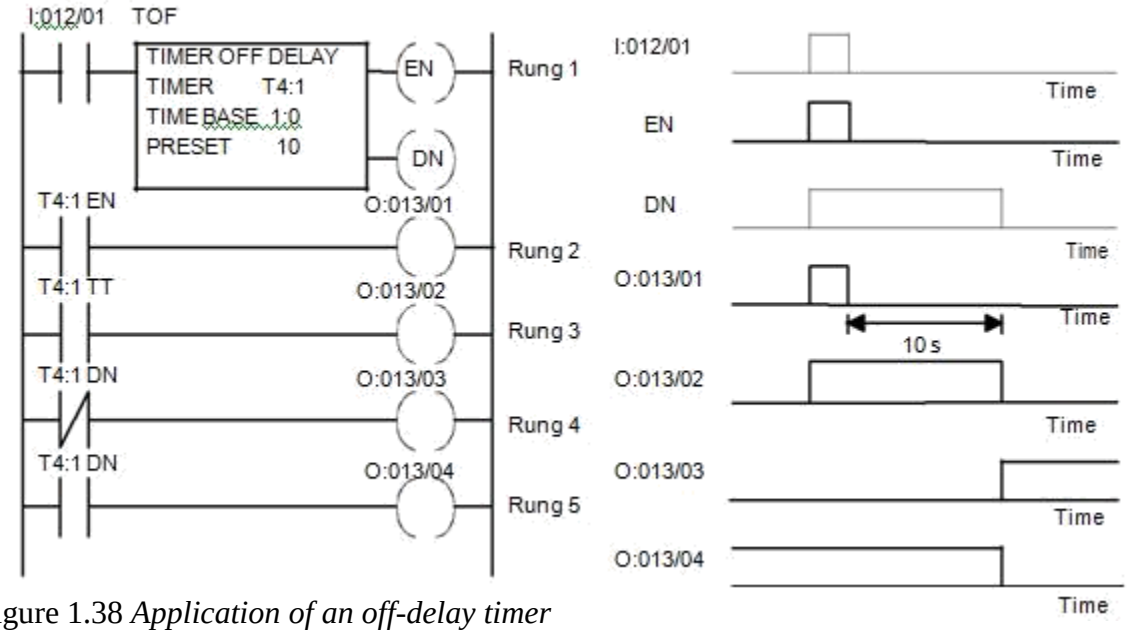


Figure 1.38 *Application of an off-delay timer*

Pulse timers

Pulse timers are used to produce a fixed duration output from some initiating input. Figure 1.39(a) shows a ladder diagram for a system that will give an output from Out 1 for a predetermined fixed length of time when there is an input to In 1, the timer being one involving a coil. There are two outputs for the input In 1. When there is an input to In 1, there is an output from Out 1 and the timer starts. When the predetermined time has elapsed, the timer contacts open. This switches off the output. Thus the output remains on for just the time specified by the timer.

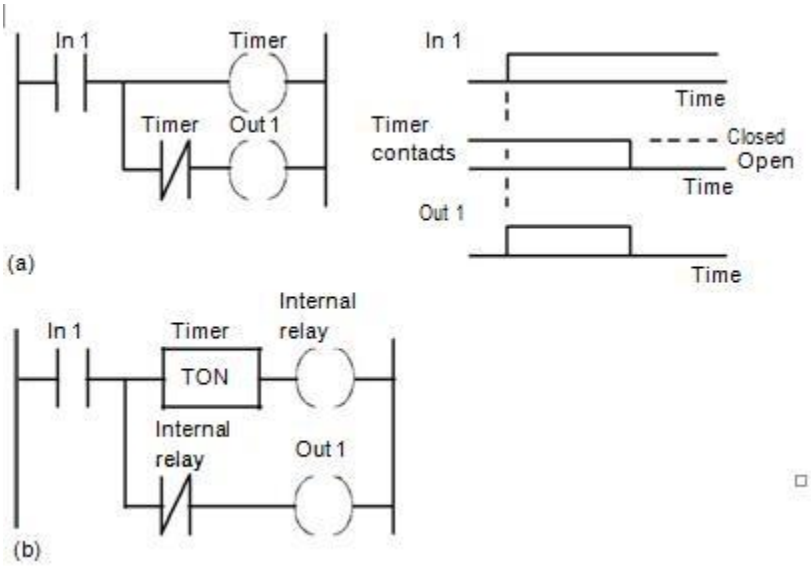


Figure 1.39 *Pulse-on timer*

Figure 1.39(b) shows an equivalent ladder diagram to Figure 9.13(a) but employing a timer which produces a delay in the time taken for a signal to reach the output.
In Figure 1.39, the pulse timer has an output switched on by an input for a predetermined time, then switching off. Figure 1.40 shows another pulse timer that switches an output on for a predetermined time after the input ceases. This uses a timer and two internal relays. When there is an input to In 1, the internal relay IR 1 is energised. The timer does not start at this point because the normally closed In 1 contacts are open. The closing of the IR 1 contacts means that the internal relay IR 2 is energised. There is, however, no output from Out 1 at this stage because, for the bottom rung, we have In 1 contacts open. When the input to In 1 ceases, both the internal relays remain energised and the timer is started. After the set time, the timer contacts, which are normally closed, open and switch off IR 2. This in turn switches off IR 1. It also, in the bottom rung, switches off the output Out 1. Thus the output is off for the duration of the input, then being switched on for a predetermined length of time.

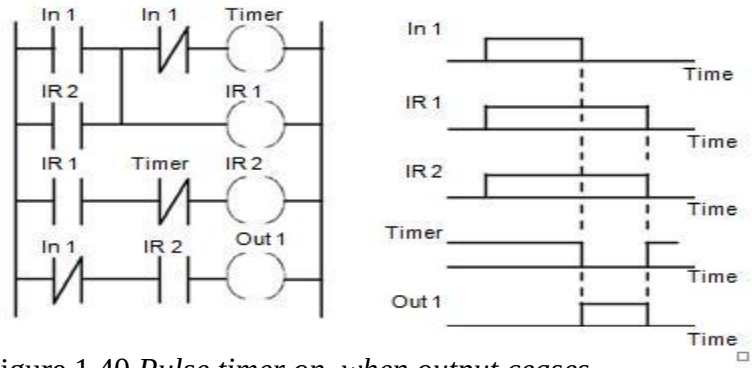


Figure 1.40 *Pulse timer on, when output ceases*

Programming examples

Consider a program (Figure 1.41) that could be used to flash a light on and off as long as there is some output occurring. Thus we might have both timer 0 and timer 1 set to 1 s. When the output occurs, then timer 0 starts and switches on after 1 s. This closes the timer 0 contacts and starts timer 1. This switches on after 1 s and, in doing so, switches off timer 0. In so doing, it switches off itself. The lamp is only on when timer 0 is on and so we have a program to flash the lamp on and off as long as there is an output.

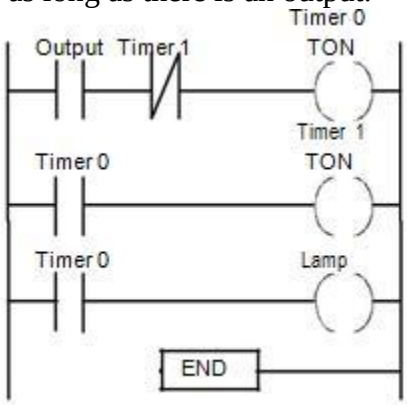


Figure 1.41 *Flashing light*

1.9 Counters

Counters are provided as built-in elements in PLCs and allow the number of occurrences of input signals to be counted. This might be where items have to be counted as they pass along a conveyor belt, or the the number of revolutions of a shaft, or perhaps the number of people passing through a door.

Forms of counter

A counter is set to some preset number value and, when this value of input pulses has been received, it will operate its contacts. Thus normally open contacts would be closed, normally closed contacts opened.

There are two types of counter, though PLCs may not include both types. These are down-counters and up-counters. *Down-counters* count down from the preset value to zero, i.e. events are subtracted from the set value. When the counter reaches the zero value, its contacts change state. Most PLCs offer down counting. *Up-counters* count from zero up to the preset value, i.e. events are added until the number reaches the preset value. When the counter reaches the set value, its contacts change state.

Different PLC manufacturers deal with counters in slightly different ways. Some count down (CTD), or up (CTU), and reset and treat the counter as though it is a relay coil and so a rung output. In this way, counters can be considered to consist of two basic elements: one relay coil to count input pulses and one to reset the counter, the associated contacts of the counter being used in other rungs. Figure 1.42(a) illustrates this. Mitsubishi is an example of this type of manufacturer. Others treat the counter as an intermediate block in a rung from which signals emanate when the count is attained. Figure 1.42(b) illustrates this. Siemens is an example of this type of manufacturer.

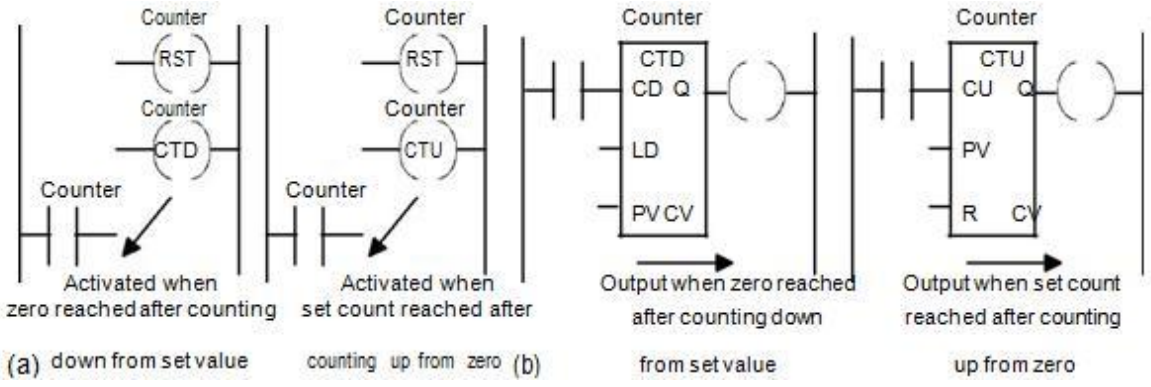


Figure 1.42 *Forms of representation of counters. In (a) RST is reset. In (b), the IEC 1131-3 representation, CD is count down input, LD is for loading the input, PV is for the preset value, CV the current count value, CU is count up input, and R is for the reset input.*

Programming

Figure 1.43 shows a basic counting circuit. When there is a pulse input to In 1, the counter is reset. When there is an input to In 2, the counter starts counting. If the counter is set for, say, 10 pulses, then when 10 pulse inputs have been received at In 2, the counter's contacts will close and there will be an output from Out 1. If at any time during the counting there is an input to In 1, the counter will be reset and start all over again and count for 10 pulses.

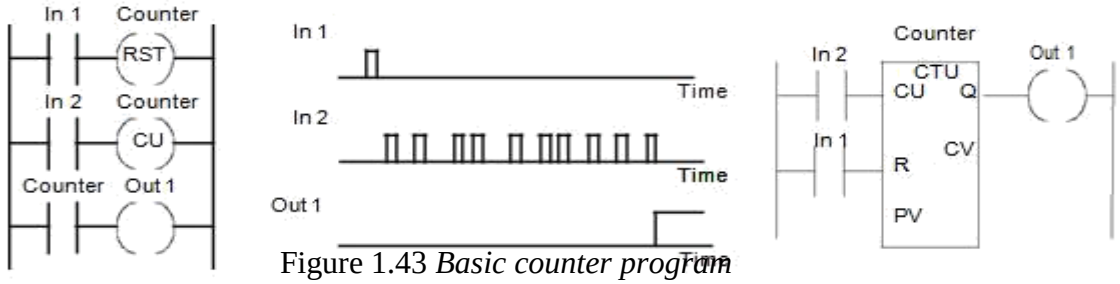


Figure 1.43 *Basic counter program*

Up and down counting

It is possible to program up - and down-counters together. Consider the task of counting products as they enter a conveyor line and as they leave it, or perhaps cars as they enter a multi-storage parking lot and as they leave it. An output is to be triggered if the number of items/cars entering is some number greater than the number leaving, i.e. the number in the parking lot has reached a 'saturation' value. The output might be to illuminate a 'No empty spaces' sign. Suppose we use the up-counter for items entering and the count down for items leaving. Figure 1.44(a) shows the basic form a ladder program for such an application can take. When an item enters it gives a pulse on input In 1. This increases the count by one. Thus each item entering increases the accumulated count by 1. When an item leaves it gives an input to In 2. This reduces the number by 1. Thus each item leaving reduces the accumulated count by 1. When the accumulated value reaches the preset value, the output Out 1 is switched on. Figure 1.44(b) shows the implementation of this program with an Allen-Bradley program.

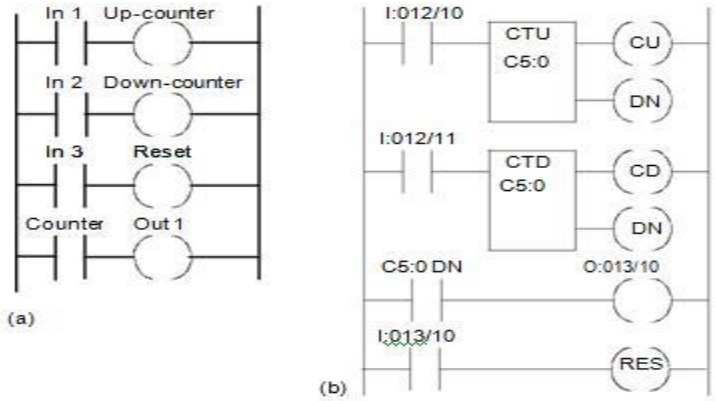


Figure 1.44 *(a) Using up- and down-counters, (b) Allen-Bradley program*

Up-down counters are available as single entities. Figure 1.45 shows the IEC 1131-3 standard symbol. The counter has two inputs CU and CD and counts up the number of pulses detected at the input CU and counts down the number of pulses detected at input CD. If the counter input

reaches zero, the QD output is set on and the counting down stops. If the count reaches the maximum value PV, the QU output is set on and the counting up stops. CV is the count value. LD can be used to preset the counter output CV with the value PV. The reset R clears the counter input to zero. Figure 1.46 shows how the above system might appear for a Siemens PLC and the associated program instruction list. CU is the count up input and CD the count down. R is the reset. The set accumulator value is loaded via F0.0, this being an internal relay.

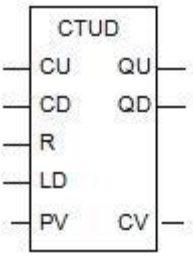


Figure 1.45 *IEC 1131-3 standard symbol for up-down counter*

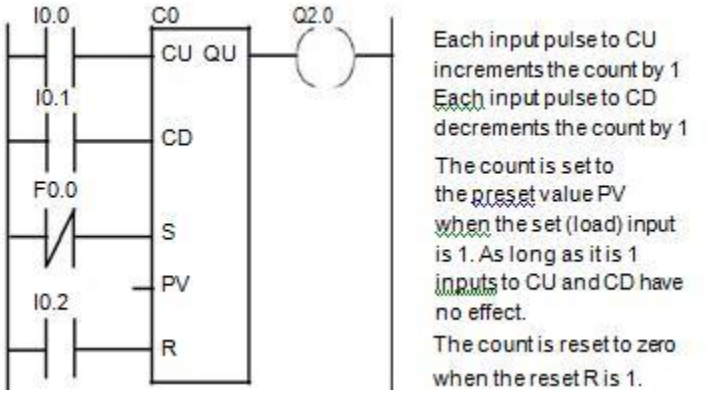


Figure 1.46 *Up and down counting with a Siemens PLC*

Timers with counters

A typical timer can count up to 16 binary bits of data, this corresponding to 32 767 base time units. Thus, if we have a time base of 1 s then the maximum time that can be dealt with by a timer is just over 546 minutes or 9.1 hours. If the time base is to be 0.1 s then the maximum time is 54.6 minutes or just short of an hour. By combining a timer with a counter, longer times can be counted. Figure 1.47 illustrates this with an Allen-Bradley program. If the timer has a time base of 1 s and a preset value of 3600, then it can count for up to 1 hour. When input I:012/01 is activated, the timer starts to time in one second increments. When the time reaches the preset value of 1 hour, the DN bit is set to 1 and the counter increments by 1. The DN bit setting to 1 also reset the timer and the timer starts to time again. When it next reaches its preset time of 1 hour, the DN bit is again set to 1 and the counter increments by 1. With the counter set to a preset value of 24, the counter DN bit is set to 1 when the count reaches 24 and the output O:013/01 is turned on. We thus have a timer which is able to count the seconds for the duration of a day and would be able to switch on some device after 24 hours.

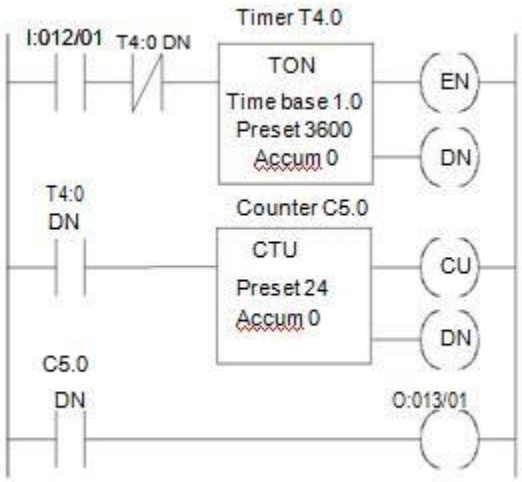


Figure 1.47 *Using a counter to extend the range of a timer*

Questions

UNIT- I Part- A

Sl.no	Part A - Questions	CO	Level
1	Describe the name PLC	CO2	L1
2	How do you select the PLC for a particular application?	CO2	L1
3	List out the difference between PLC and Computer	CO2	L1
4	Identify the programming method in PLC	CO2	L2
5	Compare the timers and counter functions of PLC	CO3	L5
6	Design the Ladder diagram for AND and NOR gates	CO1	L6
7	Create the program for the given boolean Y = AB'+ C'	CO1	L6
8	Classify the timer and counter functions in PLC	CO3	L4
9	Name any two input and output devices of PLC	CO2	L1
10	Sketch the ladder diagram of EX-OR and NAND gates	CO1	L3

UNIT- I Part – B

Sl.no	Part B - Questions	CO	Level
1	Sketch the architecture of PLC and mention the working of the individual blocks	CO2	L3
2	Discuss about the PLC programming procedures	CO2	L2
3	Design a PLC Ladder for Drill Press operation with the detailed operation	CO6	L6
4	Explain in detail about the timers and counters in PLC with design examples	CO3	L2, L6
5	Develop the PLC program for the Traffic Light controller	CO3	L6
6	(i). Explain about the PLC I/O units. (ii). Sketch a ladder diagram for a three-motor system having the following conditions: Motor 1 (M1) starts as soon as the start switch is on; after 10 seconds, M1 goes off and motor 2 (M2) starts. After 5 seconds, M2 goes off and M3 starts. After 10 seconds, M3 goes off. M1 starts and the cycle is repeated.	CO2, CO3	L2, L6
7	Sketch the ladder diagram of all the digital gates	CO1	L3

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS
DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT 2 – PLC INTERMEDIATE FUNCTIONS – SIC1405

UNIT 2 PLC INTERMEDIATE FUNCTIONS

Arithmetic functions - number comparison functions - skip and MCR functions - data move systems. PLC Advanced intermediate functions- utilizing digital bits - sequencer functions - PLC Advanced functions: alternate-programming languages - operation. PLC-PID functions -PLC installation - trouble shooting and maintenance-controlling a robot - processes with PLC - design of inter locks and alarms using PLC. Use of PC as PLC - Application of PLC - Case Study of Bottle Filling System, Field Bus and HART Protocol

2.1 ARITHMETIC FUNCTIONS

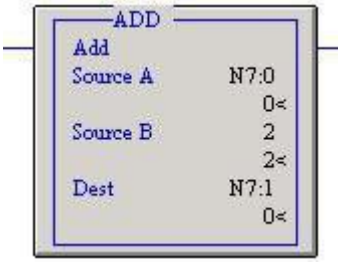
Almost all PLCs have simple ladder diagram instructions to add, subtract, multiply, and divide two numbers. A ladder rung for an instruction used to perform an arithmetic operation typically has three parts. First are the input conditions that must be true in order for the computation to take place; this can be any combination of examine instructions. Second are the locations of the two numbers to be operated upon; these locations are often entered into the ladder diagram as get instructions, which resemble examine instructions and which tell the program where to find the numbers in memory. The third and final part of an arithmetic ladder rung is the output location; it's usually entered as an address assigned to the actual arithmetic instruction (+, -, x, or /), which resembles a relay coil instruction.

The following is a list of the comparison instructions

- ADD - Adding
- SUB - Subtract
- MUL - Multiply
- DIV - Division
- DDV - Double Divide

2.1.1 ADD - Adding

Symbol



Definition

When rung conditions are true, this output instruction adds Source A to Source B and stores the result at the destination address. Source A and Source B can either be values or addresses that contain values, however Source A and Source B cannot both be constants.

- Carry (C), Sets if carry is generated; otherwise resets. Cleared For floating value
- Overflow (V), Sets if underflow; otherwise resets
- Zero (Z), Sets if the result is Zero; otherwise resets;

- Sign (S), Sets if result is negative; otherwise resets;

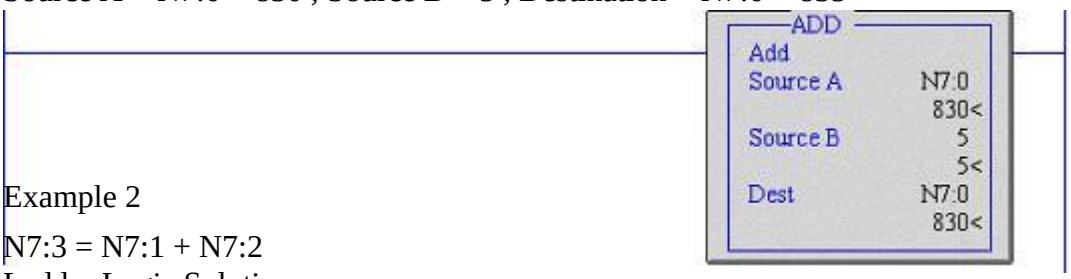
Syntex: ADD

Example 1

N7:0 = N7:0 + 5

Ladder Logic Solution

Source A = N7:0 = 830 , Source B = 5 , Destination = N7:0 = 835



Example 2

N7:3 = N7:1 + N7:2

Ladder Logic Solution

Source A = N7:1 = 32000 , Source B = N7:2 = 32000, Destination = -1536 (why).

This is because the result of the addition could not fit in 16 bit register, and therefore we have the overflow bit on.

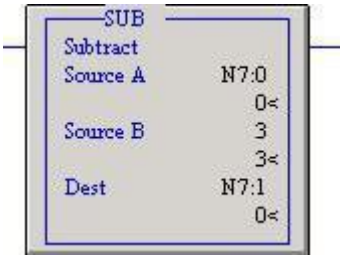
Clearing minor error that might be generated if we have an overflow.

Note:

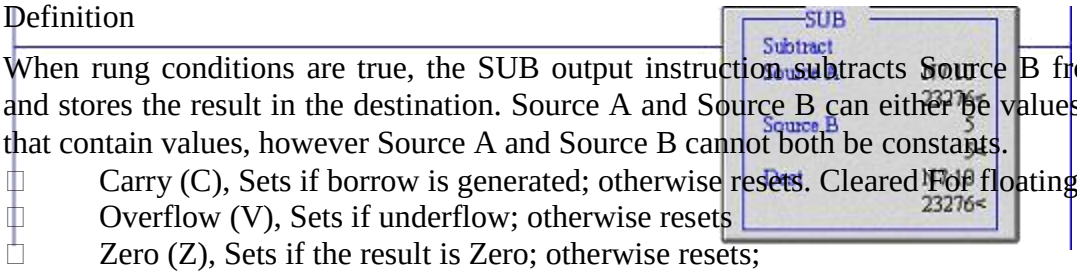
- If the destination value is > 32767 then the overflow bit S:0/1 bit will be set hence setting the minor bit error S:5/0. If this bit is not cleared then it will case major error.
- Cant program Source A and Source B as constant, at least one of them should be a variable.

2.1.2 SUB - Subtract

Symbol



Definition



When rung conditions are true, the SUB output instruction subtracts Source B from Source A and stores the result in the destination. Source A and Source B can either be values or addresses that contain values, however Source A and Source B cannot both be constants.

- Carry (C), Sets if borrow is generated; otherwise resets. Cleared For floating value
- Overflow (V), Sets if underflow; otherwise resets
- Zero (Z), Sets if the result is Zero; otherwise resets;
- Sign (S), Sets if result is negative; otherwise resets;

SUB – Subtract

When rung conditions are true, the SUB output instruction subtracts Source B from Source A and stores the result in the destination. Source A and Source B can either be values or addresses that contain values, however Source A and Source B cannot both be constants.

Example 1

Equation: N7:10 = N7:10 - 5

Ladder Logic Solution

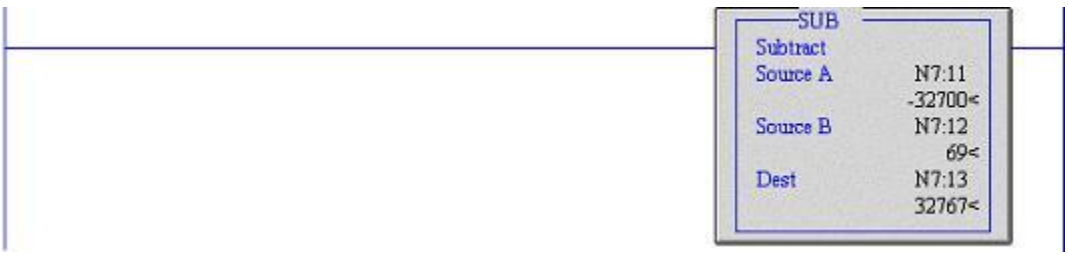
Source A = n7:10 = 23276, Source B = 5, Destination = N7:10 = 23271.

Example 2

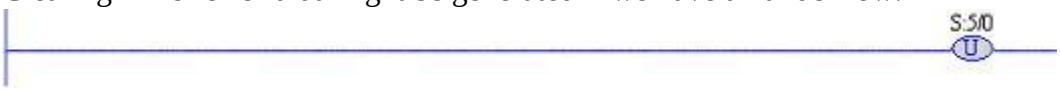
Equation: N7:13 = N7:11 - N7:12

Ladder Logic Solution

Source A = N7:11 = 32700, Source B = N7:12 = 69 , Destination = 32767 (why..). The result is invalid since the maximum negative value in a register is -32768, therfore the result is invalid and the overflow bit is on.



Clearing minor error that might be generated if we have an underflow.

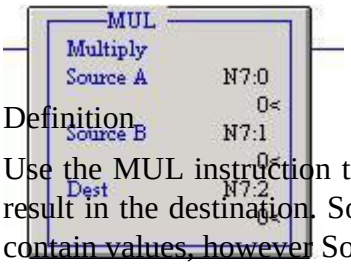


Note

- If the destination value is < -32768 then the underflow bit S:0/1 bit will be set hence setting the minor bit error S:5/0. If this error is not cleared then it will case major error.
- Can not program Source A and Source B as constant, at least one of them should be a variable.

2.1.3 MUL - Multiply

Symbol



Definition

Use the MUL instruction to multiply one value (source A) by another (source B) and place the result in the destination. Source A and Source B can either be constant values or addresses that contain values, however Source A and Source B cannot both be constants.

The math register contains the 32-bit signed integer result of the multiply operation. This result is valid at overflow.

- Carry (C), Always reset
- Overflow (V), Sets if overflow; otherwise resets
- Zero (Z), Sets if the result is Zero; otherwise resets;
- Sign (S), Sets if result is negative; otherwise resets;

MUL - Multiply

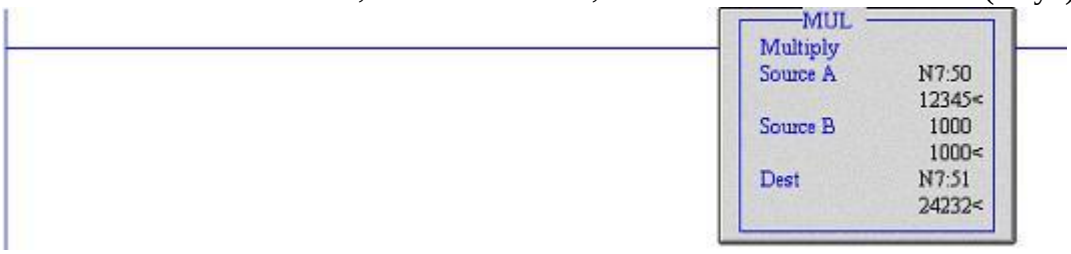
Use the MUL instruction to multiply one value (source A) by another (source B) and place the result in the destination. Source A and Source B can either be constant values or addresses that contain values, however Source A and Source B cannot both be constants.

Example 1

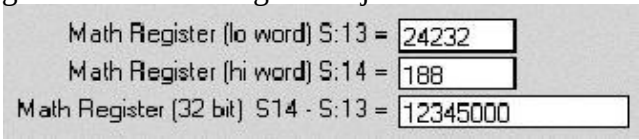
Equation: N7:51 = N7:50 * 1000

Ladder Logic Solution

Source A = N7:50 = 12345 , Source A = 1000, Destination = N7:51 = 24232(why..)



Math Register: When N7:50 = 12345 this will generate a value > 32767 this will generate an overflow status bit. But we will still get a valid result in the math register. We have to make sure to clear the minor error generate before we get a major error.



Example 2

Equation: F8:52 = N7:52 * F8:50 (F is for floating points).

Ladder Logic Solution

Source A = N7:52 = 10 , Source B = F8:50 = 1.234, Destination = F8:52 = 12.34

Notice how we mixed Integer (N) numbers with floating (F) numbers.

Example 3

Equation: N7:52 = F8:52 * F8:54 (12 = 1.23 * 10.0)

Ladder Logic Solution

Source A = F8:53 = 1.23, Source B = F8:54 = 10.0, Destination = N7:52=12

Notice how the value of two floats got truncated with the use of integer (N).

2.1.4 DIV - Divide

Symbol

Definition

When rung condition is true, this output instruction divides Source A by Source B and stores the result in the destination and the math register. The value stored in the destination is rounded. The value stored in the math register consists of the unrounded quotient (placed in the most significant word) and the remainder (placed in the least significant word).

Source A and Source B can either be constant values or addresses that contain values, however Source and Source B cannot both be constants.

- Carry (C), Sets if carry is generated; otherwise resets. Cleared For floating value
- Overflow (V), Sets if division by zero or overflow; otherwise resets
- Zero (Z), Sets if the result is Zero; otherwise resets, undefined if overflow
- Sign (S), Sets if result is negative; otherwise resets; undefined if overflow.

DIV - Divide

When rung condition is true, this output instruction divides Source A by Source B and stores the result in the destination and the math register. The value stored in the destination is rounded. The value stored in the math register consists of the unrounded quotient (placed in the most significant word) and the remainder (placed in the least significant word)

Example

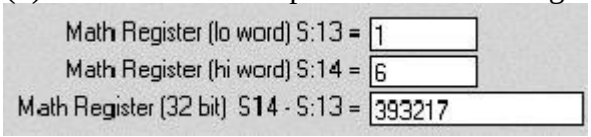
Equation: N:64 = N7:60 / 2 (13 / 2 = 6.5)

Notice the result in N7:64 is rounded up to 7).

Ladder Logic Solution

Source A = N7:60 = 13, Source B = 2, Destination = N7:64 = 7, Notice the result in N7:64 is rounded up to 7).

Math Register: The value stored in the math register consists of the unrounded quotient placed in the most significant word (6) and the remainder placed in the least significant (1)



Example

Equation: F8:62 = f8:50 / F8:61 (100.0 / 0.26 = 384.6154). The result is rounded to the closest value. Ladder Logic Solution

Source A = F8:60 = 100.00 , Source B = F8:61 = 0.26, Destination = F8:62 = 3846154

Example

Equation: N7:63 = N7:62 / F8:63 (100 / 0.9 = 111.111)

Ladder Logic Solution

Source A = N7:62 = 100 , Source B = F8:63 = 0.9 , Destination = N7:63 = 111 (Rounded)

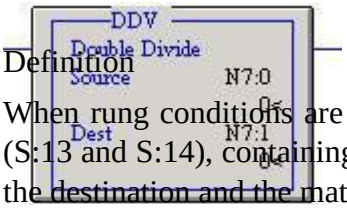
Notice that since N7:63 are an integer it will only hold the quotient part.

Note

- Division by zero (0) will generate a minor fault. S:5/0 must be cleared to prevent a major fault.

2.1.5 DDV - Double Divide

Symbol



Definition

When rung conditions are true, this output instruction divides the contents of the math register (S:13 and S:14), containing 32 bits of data, by the source (16 bits of data) and stores the result in the destination and the math register.

The math register initially contains the dividend of the DDV operation. Upon execution the unrounded quotient is placed in the most significant word of the math register. The remainder is placed in the least significant word of the math register.

- Carry (C), Sets if carry is generated; otherwise resets. Cleared For floating value

- Overflow (V), Sets if division by zero or result > 32 767 or < 32768; otherwise resets
- Zero (Z), Sets if the result is Zero; otherwise resets;
- Sign (S), Sets if result is negative; otherwise resets; undefined if overflow.

- When rung conditions are true, this output instruction divides the contents of the math register (S:13 and S:14), containing 32 bits of data, by the source (16 bits of data) and stores the result in the destination and the math register.
- Example
- Equation: N7:72 = N7:70 * N7:71 (30000 * 1000 = 60000)
- Notice that the result of this multiplication will not fit in N7:72(16 bit), but the math register (32) will be able to hold on to this value. So if we look at the value of N7:72 = -5536.

Ladder Logic Solution

- Now lets have a look at the value held in the math register. The value is 60000, which is a valid result. Now we can use this value with the DDV instruction.
- Next we want to divide the value held in the math register in 6. The DDV instruction will take the value from the math register directly and divided it by 6. Now the result of this division could be held in N7:74.

- Upon execution the unrounded quotient is placed in the most significant word of the math register. The remainder is placed in the least significant word of the math register.
- If the result of the DDV is > 32767 or < 32768 then a minor fault will be generated.

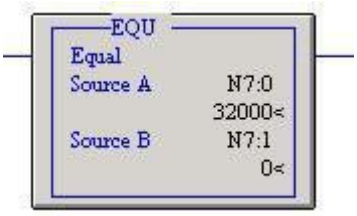
2.2 NUMBER COMPARISON FUNCTIONS

The following is a list of the comparison instructions in SLC 500:

- **EQU - Equal**
- **NEQ - Not Equal**
- **LES - Less Than**
- **LEQ - Less Than or Equal**
- **GRT - Greater Than**
- **GEQ - Greater or Equal**
- **MEQ - Masked Comparison for Equal**
- **LIM - Limit Test**

2.2.1 EQU - Equal

Symbol



Definition

- Test whether two values are equal or not.
- If source A and Source B are equal, the instruction is logically true. Source A must be an address. Source B can either be a program constant or an address. Negative integers are stored in two's complement.

Test whether two values are equal or not.

Example

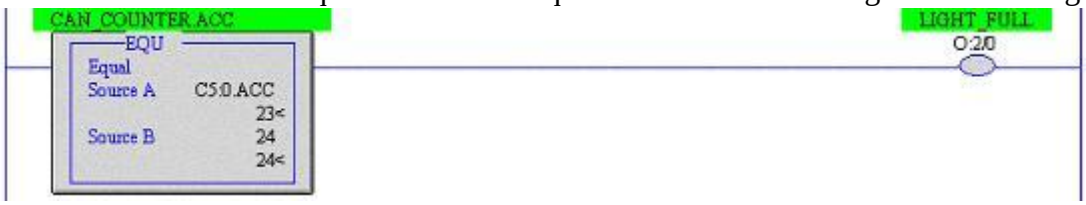
Let's say we have a counter that counts the number of cans that passes in front of a photocell. We want to check to counter to see if the counter reaches 24 (full box), then we want to turn the light on to inform the operator that we have a full box ready.

Ladder Logic Solution

- Photocell I:1.0/0
- Light O:2.0/0

When the photocell will go from false to true the counter c5:0 will increment by one.

Check if the counter is equal to 24. If the equation is true then the light will be energized.

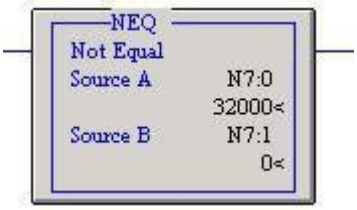


Note

- We could have used the C5:0.DN to verify if the counter has reached 24, but we used the EQU for illustration purpose.

2.2.2 NEQ - Not Equal

Symbol



Definition

- Test whether one value is not equal to a second value.
- If Source A and Source B are not equal, the instruction is logically true. If the two values are equal, the instruction is logically false.
- Source A must be an address. Source B can be either a program constant or an address. Negative integers are stored in two's complement.

Test whether one value is not equal to a second value.

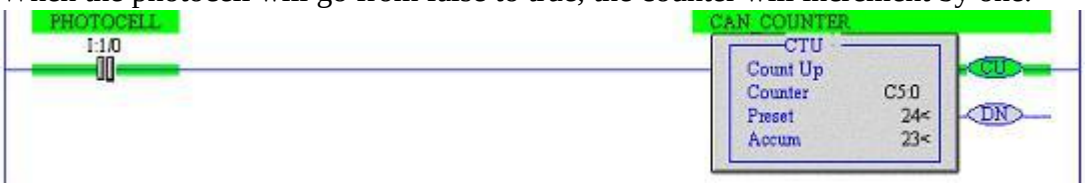
Example

Let's say we have a counter that counts the number of cans that passes in front of a photocell. We want to check to counter to see if the counter reaches 24 (full box), then we want to turn the green light on, but if the count is not equal to 24 then we want turn the read light on.

Ladder Logic Solution

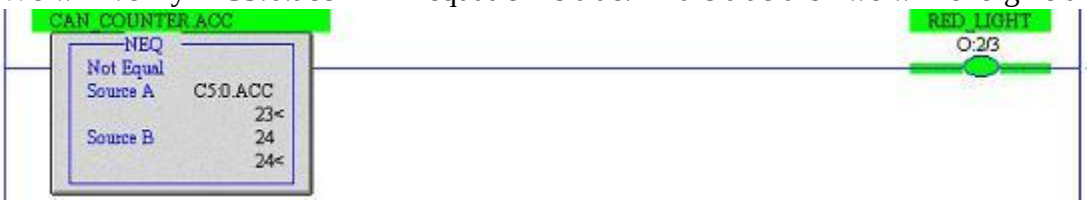
- Photocell I:1.0/0
- Light Green O:2.0/2
- Light Red O:2.0/3

When the photocell will go from false to true, the counter will increment by one.



We will verify if C5:0.Acc = 24 equation is true. If it is then we will energize the light full.

We will verify if C5:0.acc <> 24 equation is true. If it is true then we will energize the red light.

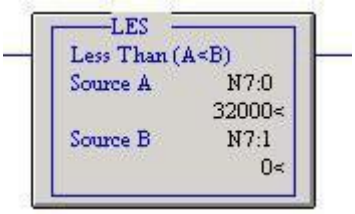


Note

- The red light will basically be on except when the count is equal to 24.

2.2.3 LES - Less Than

Symbol



Definition

- Test whether one value is less than a second value.
- If Source A is less than the value at source B the instruction is logically true. If the value at source A is greater than or equal to the value at source B, the instruction is logically false.
- Source A must be an address. Source B can either be a program constant or an address. Negative integers are stored in two's complement.

2.2.4 NEQ - Not Equal

Test whether one value is not equal to a second value.

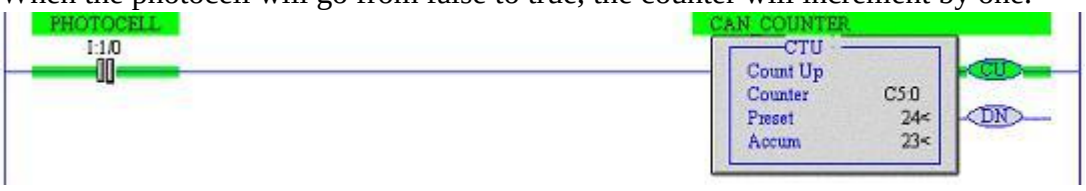
Example

Let's say we have a counter that counts the number of cans that passes in front of a photocell. We want to check to counter to see if the counter reaches 24 (full box), then we want to turn the green light on, but if the count is not equal to 24 then we want turn the read light on.

Ladder Logic Solution

- Photocell I:1.0/0
- Light Green O:2.0/2
- Light Red O:2.0/3

When the photocell will go from false to true, the counter will increment by one.



We will verify if C5:0.Acc = 24 equation is true. If it is then we will energize the light full.

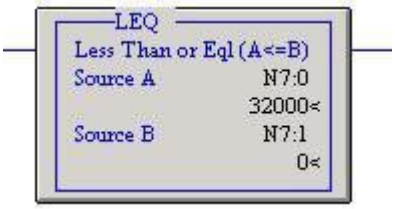
We will verify if C5:0.acc <> 24 equation is true. If it is true then we will energize the red light.

Note

- The red light will basically be on except when the count is equal to 24.

2.2.5 LEQ - Less Than or Equal

Symbol



Definition

- Test whether one value is less than or equal to a second value.
- If value at source A is less than or equal to the value at source B, the instruction is logically true.
- If the value at source A is greater than or equal to the value at source B, the instruction is logically false.
- Source A must be an address. Source B can either be a program constant or an address. Negative integers are stored in two's complement.

Test whether one value is less than or equal to a second value.

Example

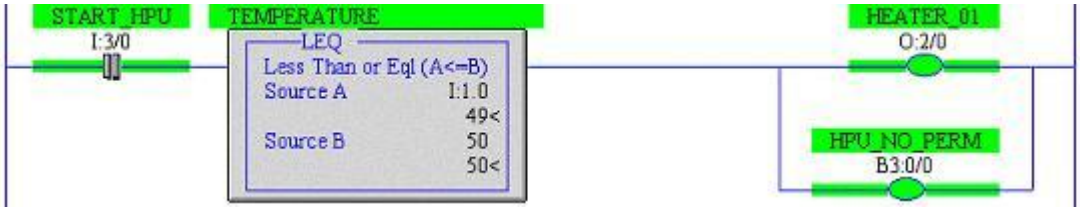
Let say we have a HPU system that we want to monitor the temperature of the oil. When we start the HPU system, we want to check If the temperature of the oil is

less than or equal to 50C(Celsius) then we will run Heater 1 on. If the temperature is less than 70C then we want to run Heater 2. We should also disallow the HPU to run if the temperature is less or equal to 50C.

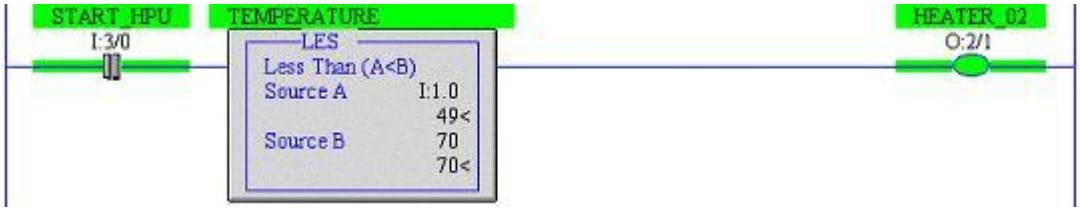
Ladder Logic Solution

- Temperature element I:1.0/0
- Heater 01 O:2.0/0
- Heater 02 O:2.0/1
- HPU no permission B3:0/0
- HPU start O:2.0/2

The start button is pushed then we check to see if the temp <= 50c is true. If it is true then we energize Hearter1 and we disallow to run the HPU.



While the start is still on we check if the temperature is below 70c if it is then we run the second heater to accelerate the heating process.



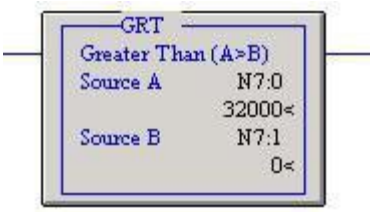
As long the as the temperature is below 50 then we will not allow the HPU to start.

Note

- For simplicity reasons, we are assuming that the push buttons are maintained signals.
- Notice that the temperature value is passed via an analog input (16 bit word).

2.2.6 GRT - Greater Than

Symbol



Definition

- Test whether one value is greater than the second value.
- If the value at source A is greater than the value at source B, the instruction is logically true.
- If the value at source A is less than or equal to the value at source B, the instruction is logically false.
- Source A must be an address. Source B can either be a program constant or an address. Negative integers are stored in two's complement.

Test whether one value is greater than the second value.

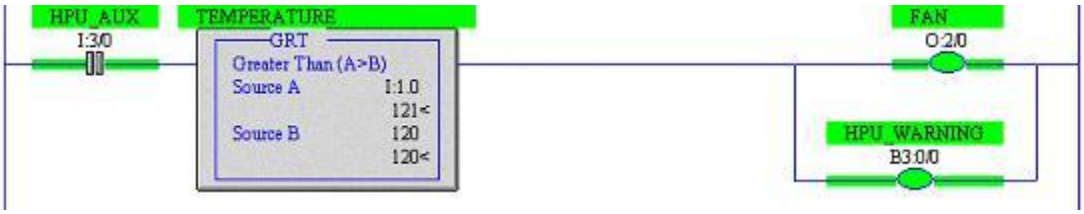
Example

Let's say we have a HPU system that we are monitoring its oil temperature. When the temperature of the oil is greater than 120 degree Celsius, we want to send a warning and start the fan to cool down the oil.

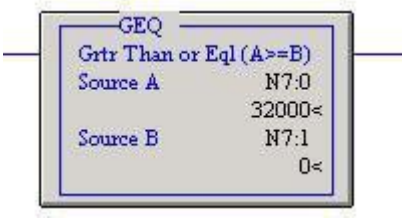
Ladder Logic Solution

- Temperature I:1.0
- Fan O:2.0/0
- HPU auxiliary I:3.0/0
- HPU Warning B3:0/0

While the HPU is running if the temperature gets higher than 120 degree then the fan will start and a warning will be set to on.



Symbol



Definition

- Test whether one value is greater or equal to a second value.
- If the value at source A is greater than or equal the value at source B, the instruction is logically true.
- If the value at source A is less than to the value at source B, the instruction is logically false.
- Source A must be an address. Source B can either be a program constant or an address. Negative integers are stored in two's complement.

Test whether one value is greater or equal to a second value.

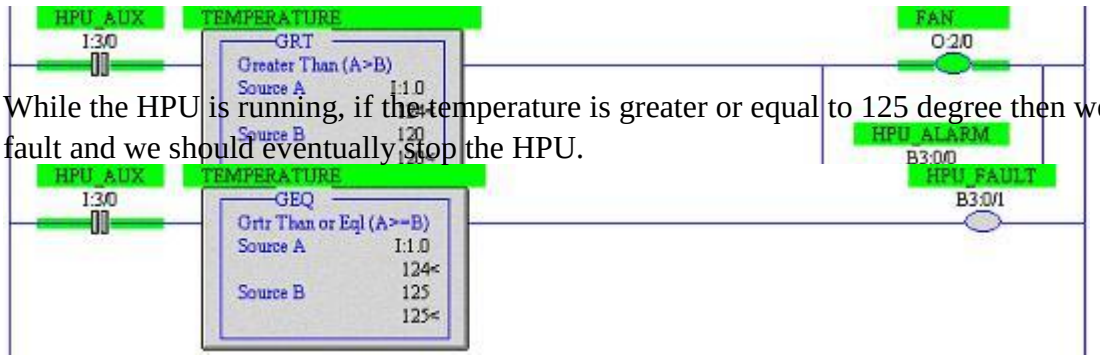
Example

Let's say we have a HPU system that we are monitoring its oil temperature. When the temperature of the oil is greater than 120 degree Celsius, we want to send a warning and start the fan to cool down the oil. Now if the temperature is greater or equal to 125 then we want to stop the HPU cause it is getting very hot. We also want generate a fault to the operator to let him know why the HPU have stopped.

Ladder Logic Solution

- Temperature I:1.0
- Fan O:2.0/0
- HPU auxiliary I:3.0/0
- HPU Alarm B3:0/0
- HPU Fault B3:0/1

If the HPU is running and the temperature is above 124 degree then we start the fan to cool it down.



While the HPU is running, if the temperature is greater or equal to 125 degree then we energize a fault and we should eventually stop the HPU.

Note

- HPU fault should de-energize the HPU start signal.
- HPU fault could be shown with a flashing light to inform the operator as to why the HPU have stopped.

2.2.7 MEQ - Masked Comparison for Equal

Symbol

Definition

- Test portion of two values to see whether they are equal. Compares 16 bit data of a source address to 16 bit data at a reference address through a mask
- Use the MEQ instruction to compare data at a source address with data at a compare address. Use of this instruction allows portions of the data to be masked by a separate word.
- Source is the address of the value you want to compare.
- Mask is the address of the mask through which the instruction moves data. The mask can be a hexadecimal value.
- Compare is an integer value or the address of the reference.
- If the 16 bits of data at the source address are equal to the 16 bits of data at the compare address (less masked bits), the instruction is true. The instruction becomes false as soon as it detects a mismatch. Bits in the mask word mask data when reset; they pass data when set.

Test portion of two values to see whether they are equal. Compares 16 bit data of a source address to 16 bit data at a reference address through a mask.

Example

Lets say we have four counters. Each counter goes from 0 to 9. Since 0 to 9 can be represented in 4 bits then we can combine the four counters in one word of 16 bits. This is to reduce the amount of memory counters will need.

- Counter 1 = N7:0 bit 0,1,2,3
- Counter 2 = N7:0 bit 4,5,6,7
- Counter 3 = N7:0 bit 8,19,10,11
- Counter 4 = N7:0 bit 12,13,14,15

Now that we have all four counters in one word we want to compare our result to a certain value. MEQ is the solution for such a problem. MEQ will make a comparison of a value based on the bits specified in the mask section.

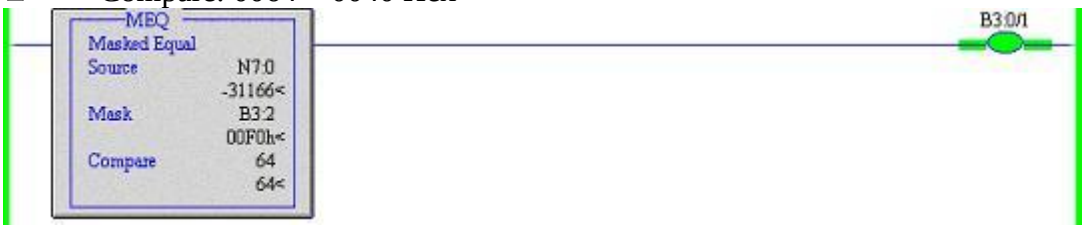
Ladder Logic Solution

Comparing N7:0 = -31166 with 2 is energizing B3:0/0 why? Well cause:

- Source: -31166 = 8642 Hex
- Compare: 0002 = 0002 hex

Comparing N7:0 = -31166 with 64 is energizing B3:0/1 why? Well cause:

- Source: -31166 = 8642 Hex
- Compare: 0064 = 0040 Hex

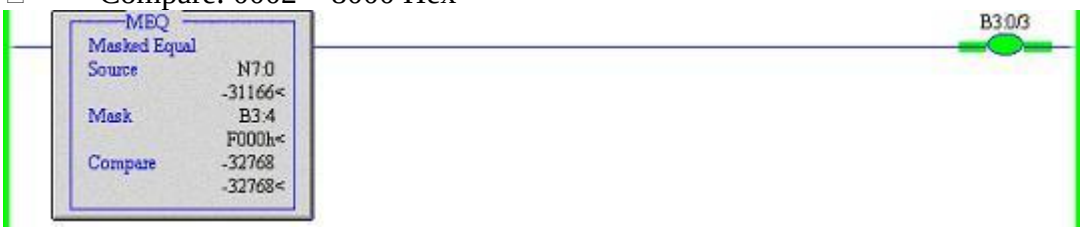


Comparing N7:0 = -31166 with 1536 is energizing B3:0/2 why? Well cause:

- Source: -31166 = 8642 Hex
- Compare: 0600 = 0002 Hex

Comparing N7:0 = -31166 with -32768 is energizing B3:0/3 why? Well cause:

- Source: -31166 = 8642 Hex
- Compare: 0002 = 8000 Hex



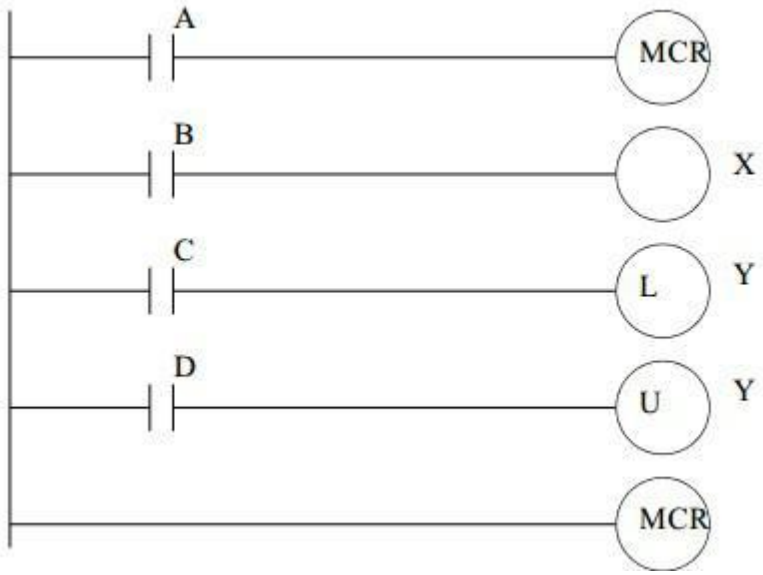
Note

- When the mask is equal to F = 1111 in Binary. So if you want to mask only the first 3 bits out of 4 then your mask should be 0111, which is equal to 7 in Hex. What this mean is that your binary bit is on then the comparison would be done on that specific bit or bits.

2.3 SKIP AND MCR FUNCTIONS

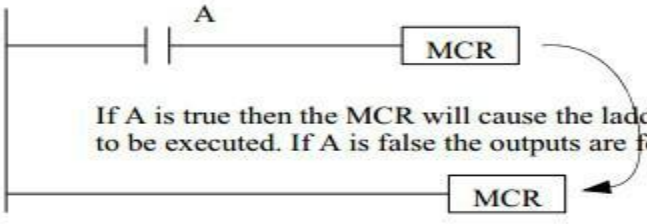
2.3.1 MASTER CONTROL RELAYS (MCRs)

In an electrical control system a Master Control Relay (MCR) is used to shut down a section of an electrical system, as shown earlier in the electrical wiring chapter. This concept has been implemented in ladder logic also. A section of ladder logic can be put between two lines containing MCR's. When the first MCR coil is active, all of the intermediate ladder logic is executed up to the second line with an MCR coil. When the first MCR coil in inactive, the ladder logic is still examined, but all of the outputs are forced off. Consider the example in Figure If A is true, then the ladder logic after will be CTU Counter C5:0 Preset 6 CTU Counter C5:1 Preset 11 part present C5:0/DN part present pneumatic cylinder C5:1/DN C5:0 C5:1 RES RESplc timers - 9.18 executed as normal. If A is false the following ladder logic will be examined, but all of the outputs will be forced off. The second MCR function appears on a line by itself and marks the end of the MCR block. After the second MCR the program execution returns to normal. While A is true, X will equal B, and Y can be turned on by C, and off by D. But, if A becomes false X will be forced off, and Y will be left in its last state. Using MCR blocks to remove sections of programs will not increase the speed of program execution significantly because the logic is still examined.



If the MCR block contained another function, such as a TON timer, turning off the MCR block would force the timer off. As a general rule normal outputs should be outside MCR blocks, unless they must be forced off when the MCR block is off. A MCR MCR

Note: We will use MCR instructions to implement some of the state based programs. This allows us to switch off part of the ladder logic. The one significant note to remember is that any normal outputs (not latches and timers) will be FORCED OFF. Unless this is what you want, put the normal outputs outside MCR blocks.



Note: If a normal input is used inside an MCR block it will be forced off. If the output is also used in other MCR blocks the last one will be forced off. The MCR is designed to fully stop an entire section of ladder logic, and is best used this way in ladder logic designs.

2.3.2 DATA MOVE SYSTEMS

Data transfer instructions move, or transfer, numerical data within a PLC,either in single register units or in blocks (a group of registers). These instructions can move data to or from any location in the memory data table, with the exception of user-restricted areas. Typical uses of data transfer instructions are the movement of constant and/or preset values to counters and timers, the reading of analog inputs and multibit input modules, and the transferring of data to output modules.The functional block group of data transfer instructions forms perhaps the most useful set of functions available in enhanced PLCs, after the basic relay instructions. The names of the data transfer instructions may differ depending on the controller, yet they implement the same transfer functions. Table 9-9 shows the different instructions available with data transfer operations.

MOVE

A *move* (MOV) functional block instruction transfers information from one location to another, with the destination location being a single bit or register. Figure 9-99 shows *move bit* (MOVB) and *move register* (MOVR) functional blocks. Some PLCs also offer *move byte* instructions.

Data Transfer Instructions

	Instruction	**Symbol**	**Function**
		First In–First Out	FIFO
		Transfer	
Move	MOV/MOVB/		
	MOVR/MOVM	Sort	SORT
	MOVBK		
	REG-TABLE/		
Table Move	TABLE-REG		
Block Transfer—In/Out	BKXFER		

ASCII Transfer ASCII XFER

Transfers information from one location to another

Moves data from a group of register locations to another location

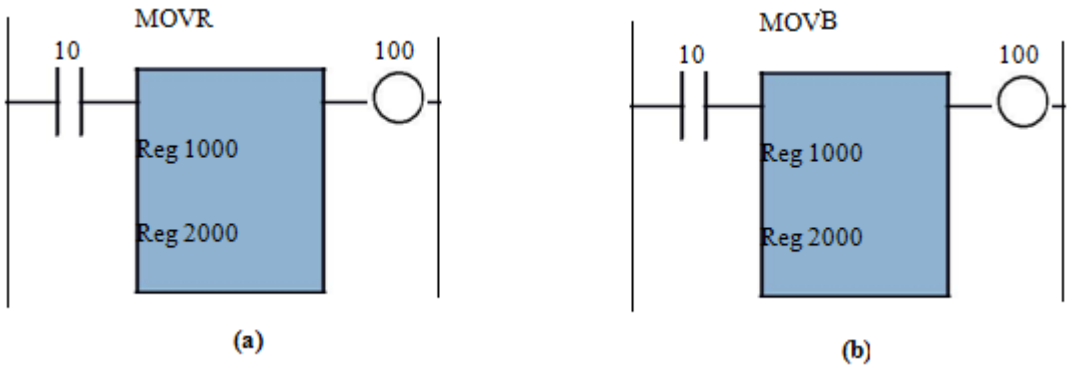
Transfers data from a block or table to a register
Stores a block of data in specified memory or register locations

Transmits ASCII data between a peripheral device and a PLC

Constructs a table or queue for storing data

Sorts the data in a block of registers in ascending/descending order

Data transfer instructions.



Contents of register 1000 are moved to register 2000 (destination register can be an I/O register)

Figure. (a) Move bit and **(b)** move register functional blocks.

Some PLCs perform a move function to special word table locations. In this case, the PLC automatically coverts the copied data to the proper numerical format for the destination location. For example, a register or word might contain a BCD value that, when transferred to another register or word, is stored as a binary value, thus executing a BCD-to-binary conversion within the move instruction.

Another type of move instruction, a *move mask* (MOVM) instruction, masks certain bits within the register. Figure 9-100 illustrates this type of move block. The move mask block transfers the data in register 1000 to register 1100, with the exception of the bits specified by a 0 in the mask register 2000.

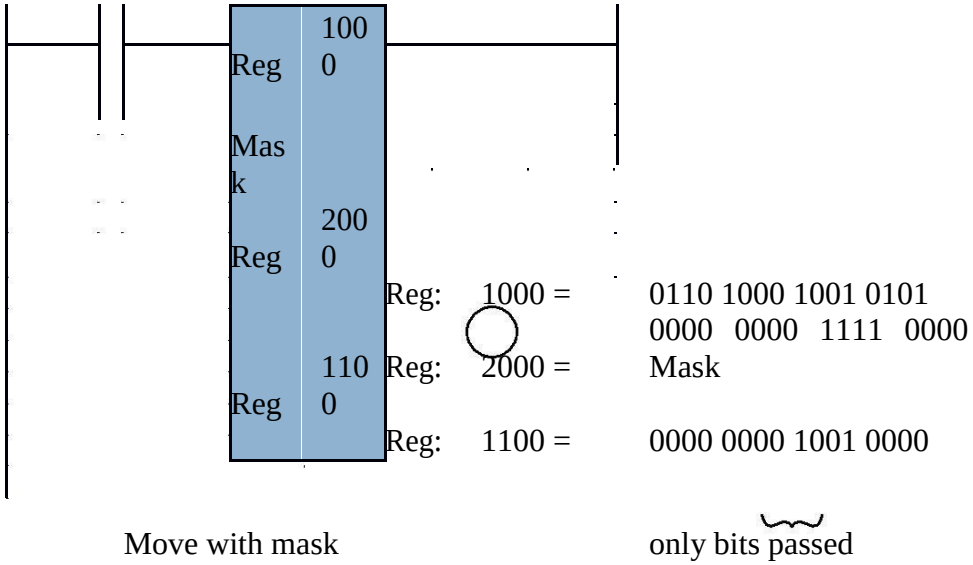


Figure Move mask functional block.

Yet another move instruction found in some controllers is the *move status* instruction. This block function transfers system or I/O module status data to a storage/result register. This information can then be masked, compared, or examined to determine the status of major or minor faults in the system or an I/O module. With this information, the controller can take corrective action through the control program, if necessary.

MOVE BLOCK

A *move block* (MOVBK) instruction copies a group of register or word locations from one place to another. The length of the block is generally user-specified. Figure 9-101 illustrates a move block instruction. When energized, the control input triggers the execution of this block. The block function then transfers data from locations 1000 through 1023 (length = 20) to locations 2000 through 2023, respectively. The data in registers 1000 through 1023 is left unchanged. In some PLCs, the user can specify how many locations can be transferred during one scan (rate per scan).

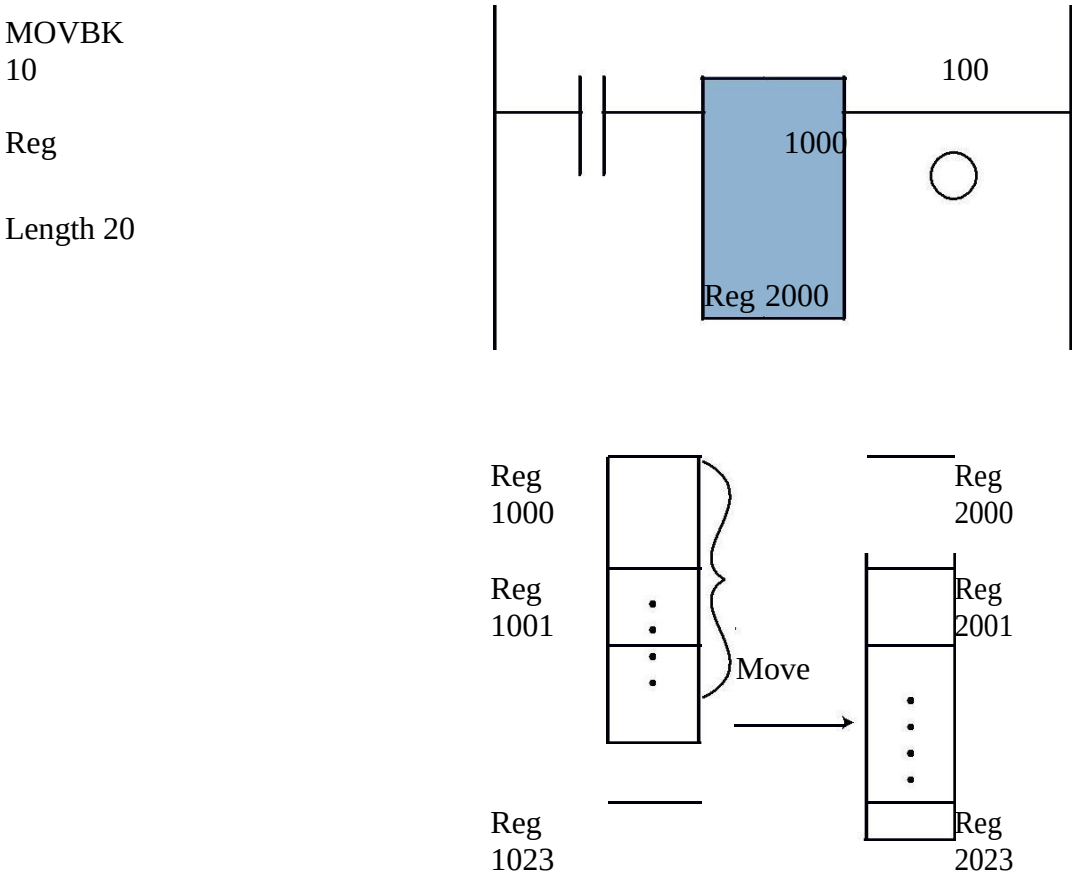


Figure Move block functional block.

TABLE MOVE

A *table move* instruction transfers data from a block or table to a register or word in memory. There are two types of table move instructions: *table-to-register* (TABLE-REG) and *register-to-table* (REG-TABLE). The maincharacteristic of a table move block is the manipulation of a pointer register, which specifies the particular table location in which the register or word value will be stored. Figure 9-102 shows a table move block.

The transition of the control input from OFF to ON enables a table move instruction, which then increments the contents of the pointer register every time the middle input, the increment (INCR) pointer, transitions from OFF to ON. The bottom input of the table move block resets the pointer to zero (initialize to top of table). If data must be stored to or retrieved from a specific table location, the pointer register can be loaded with the appropriate value, which points to the specified location. A set parameter or move register instruction loads this information prior to the table move.

Referencing Figure, the length specifies the number of word locations in the table to be moved (8 in this example), beginning at the starting location (register 2000). After the table move block transfers the data from these eight locations, it energizes the top output. It energizes the middle output when the pointer register has reached the end of the table.

Applications of the table move instruction include the loading of new data into a table, the storage of input information (e.g., analog) from special modules, and the input of error information from a controlled process. It is also useful when changing preset parameters in timers and counters and when simultaneously driving a group of 16 outputs through I/O registers. A table move instruction is also used when looking up values in a table for comparison, linear interpolation, etc.

BLOCK TRANSFER—IN/OUT

Some PLCs provide *block transfer* (BXFER) instructions, which are prima-rily used with special I/O modules to transfer blocks of data. The two basic types of block transfers are *block transfer in* and *block transfer out*. Figure 9-105 shows a block transfer in/out instruction. The module address location of the transfer data may be explicitly marked as the rack and slot location of the interface. For example, the block transfer input in Figure 9-106 shows that the contents to be read from the intelligent module (rack 01, slot 03, 8 channels) will be stored in registers 1000 through 1007.

The rack and slot indicate the location of the input or output module. The rack and slot entries in the block may be combined into one address in some PLCs.

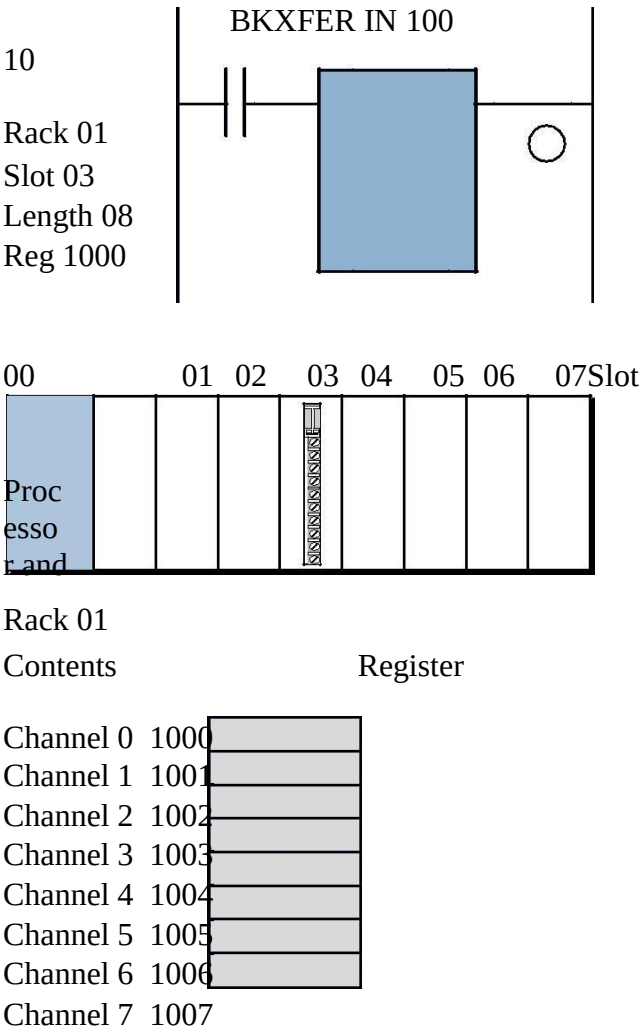


Figure. Block transfer in instruction.

The control input, when enabled, executes a block transfer instruction. During a block transfer in function, the instruction stores data about the I/O module in memory locations or registers starting at the specified register location. The block length specifies how many locations are needed to store the I/O module data. For example, the data from an analog input module with four input channels can be read all at once, if the length is specified as four. A block transfer out instruction operates in a similar manner, with the address of the output module determining the destination of the data transfer. The top output of the block transfer instruction, when energized, signals the completion of the transfer operation.

2.4 UTILIZING DIGITAL BITS

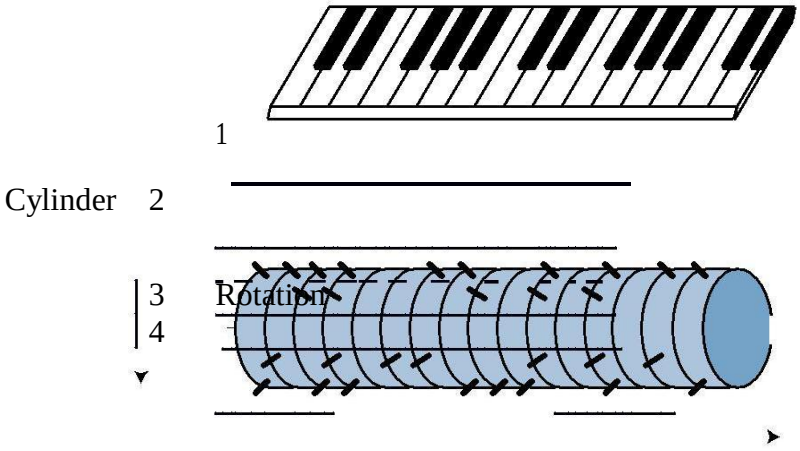
- **Bit Set**
- **Bit Clear**
- **Bit Follow**

According to the input enable signal the bit will be converted to 1 if bit set is used and to zero if bit clear is used and follows the enable signal if bit bit follow signal is used.

SEQUENCER

A *sequencer* (SEQ) block is a powerful instruction that simulates a drum timer. A sequencer is analogous to a music box mechanism, in which each peg produces a tone as the cylinder rotates and strikes the resonators. In a sequencer, each peg (bit) can be interpreted as a logic 1 and no peg as a logic 0.

A sequencer table, which is similar to a spread-out music box cylinder, provides sequencer information. Figure 9-111 illustrates a cylinder and sequencer table comparison. The number of bits in a sequencer can vary from 8 to 64 or more. The width of the table may also vary, as may the size of a cylinder. Through I/O registers, which map real output points, each step in a sequencer table can become an output representing one of the pegs.



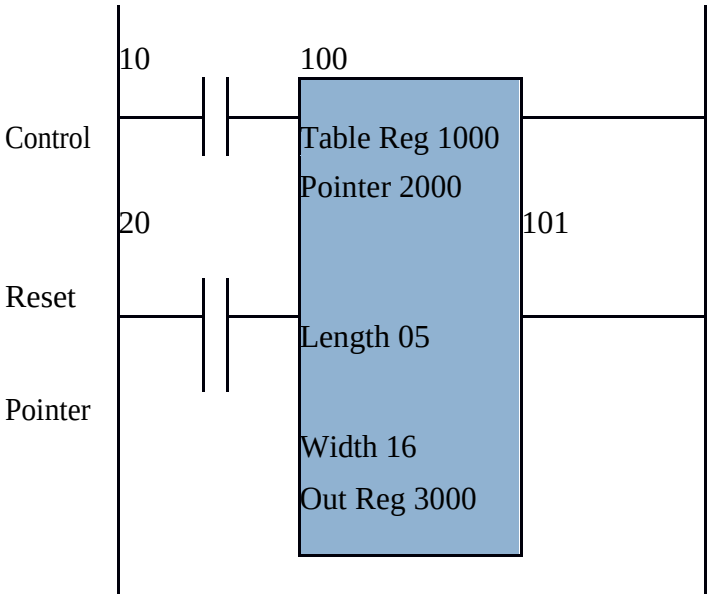
Bit locations in table

1	1	1 1	0 0	1	1	0 0	1 0	1	0	1
2	0	1 1	0 0	0	0	1 0	1 0	1	0	0
3	1	0 0	1 1	1	1	0 1	0 1	0	1	1
4	1	0 1	0 1	1	0	0 0	1 0	1	0	1
5	0	1 1	0 0	0	1	1 1	0 0	1	0	0

1 = Energize
0 = De-energize

Figure. Comparison of a music box cylinder and a sequencer table.

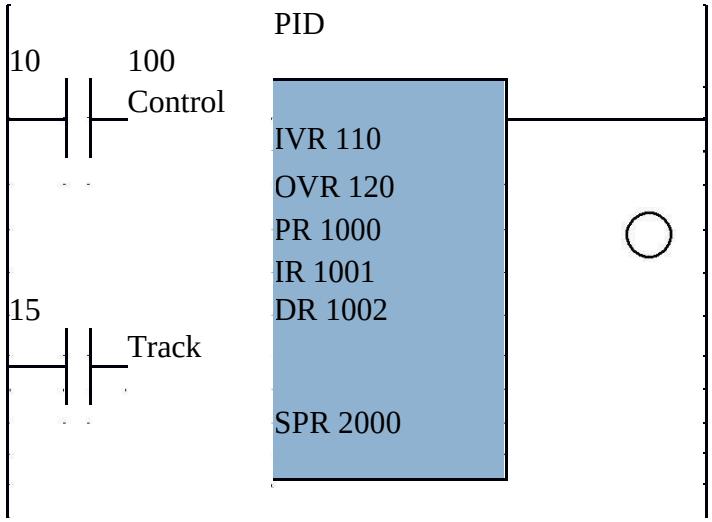
Figure shows a typical sequencer functional block. An OFF-to-ON transition of the control input initiates this block, causing the contents of the sequencer table to be output in a sequential manner. The pointer register points to each step being output (i.e., the table register location). Every time the control input is energized, the pointer register is automatically incremented, thus pointing to the next table location. Depending on the PLC, either an event or time may drive the control input line; therefore, sequencers may be either event driven or time driven. A reset pointer input can reset the pointer register to zero (point to step 1), if needed. The sequence length and width specify how many steps and bits are in the table, respectively. When-ever the sequencer instruction is enabled, it energizes the block's first output. The second output indicates the end of the sequencer table.


10
100
Control
Table Reg 1000
Pointer 2000
20
101
Reset
Length 05
Pointer
Width 16
Out Reg 3000

block.

2.7 PID

PLCs that are capable of performing analog control using the PID algorithm use *proportional - integral-derivative* (PID) functional blocks. The user speci-fies certain parameters associated with the algorithm to control the process correctly. Figure 9-116 illustrates a typical PID block.



Register 110 maps analog input module
Register 120 maps analog output module

Figure. PID functional block.

An energized control input enables a PID block's automatic operation. The bottom input track, when energized, determines whether the PID variables are being tracked but not output. If the block is not enabled (i.e., in manual mode), the controller can still track the variables when the track line is enabled. The user specifies the input variable register (IVR) and the output variable register (OVR), which are associated with the locations of the analog modules (input and output). The proportional register (PR), integral register (IR), and derivative register (DR) hold the gain values that must be specified for the three parts of the control process. The set point register (SPR) holds the target value for the process set point. Depending on the controller, the user can specify other block variables, such as dead times, high and low limits, and rate of update. The top output of the PID block indicates an active loop
control, while the middle and bottom outputs indicate low- and high-limit alarms, respectively. Some PLC manufacturers provide a fill-in-the- blanks screen (see Figure 9-117) during the programming of a PID instruction, so that the user can input the different parameters.

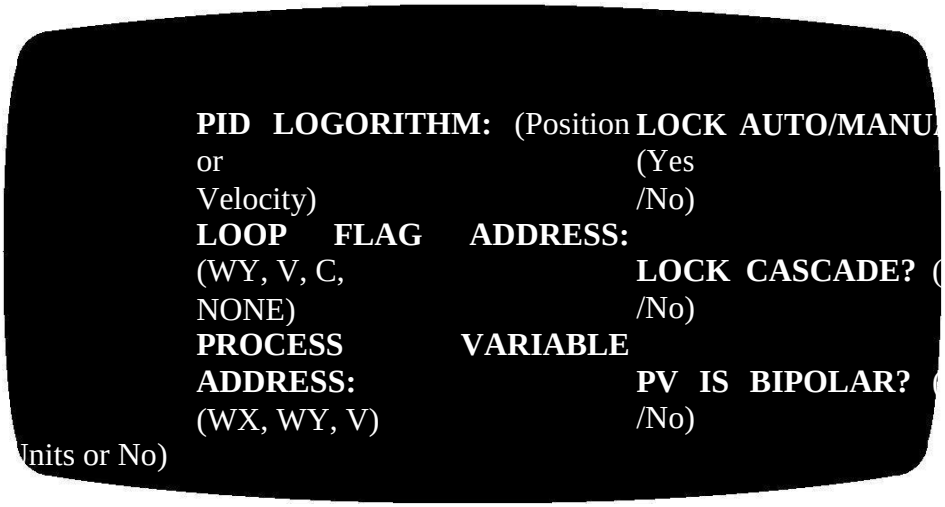


Figure. Fill-in-the-blanks screen.

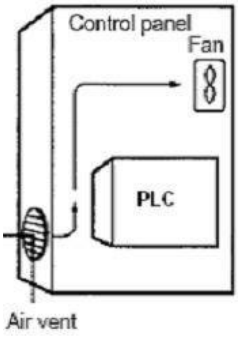
Some controllers provide PID capabilities without the PID block instruction. In this case, the controller uses a special PID module that contains all of the input/output parameters. An output instruction, such as block transfer out or move data, transfers the set point and gain parameters to the module during initialization of the program. The control program can alter this module data if any parameter changes are required.

2.8 PLC INSTALLATION,TROUBLESHOOTING and MAINTANENCE INSTALLATION

1. **Panel/Cabinet Installation**

Consider PLC operation, maintenance, and surrounding conditions when installing the PLC in a panel or cabinet. The operating temperature range for the PLC is 0°C to 55°C. Be sure that there is adequate ventilation for cooling;

- Allow enough space for air circulation.
- Do not install the PLC above equipment that generates a large amount of heat, such as heaters, transformers, or large resistors.
- Install a cooling fan or system when the ambient temperature exceeds 55°C



Power lines & high-voltage equipment can cause electrical noise in the PLC ;

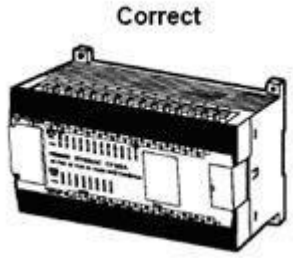
- Do not install the PLC in a panel or cabinet with high-voltage equipment
- Allow at least 200 mm between the PLC and nearby power lines

Ensure that the PLC can be accessed for normal operation and maintenance;

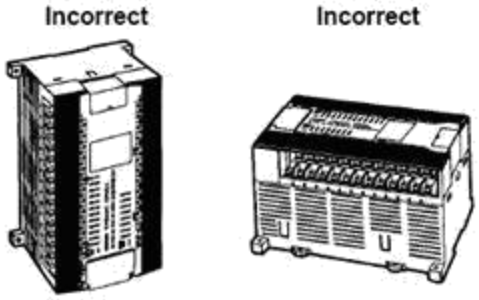
- Provide a clear path to the PLC for operation and maintenance. High-voltage equipment or power lines could be dangerous if they are in the way during routine operations.
- The PLC will be easiest to access if the panel or cabinet is installed about 3 to 5 feet above the floor

2. Installing the CPU Unit & I/O Unit

The small PLC must be installed in the position shown below to ensure adequate cooling.



Do not install the small PLC in either of the following positions

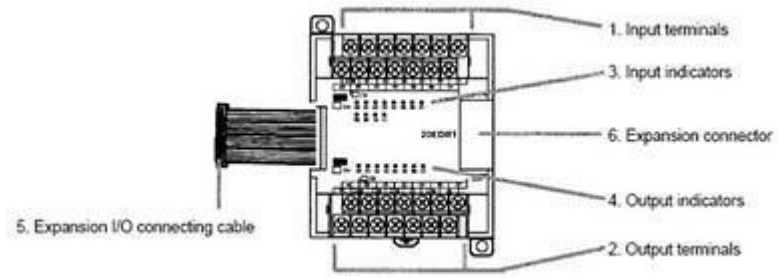


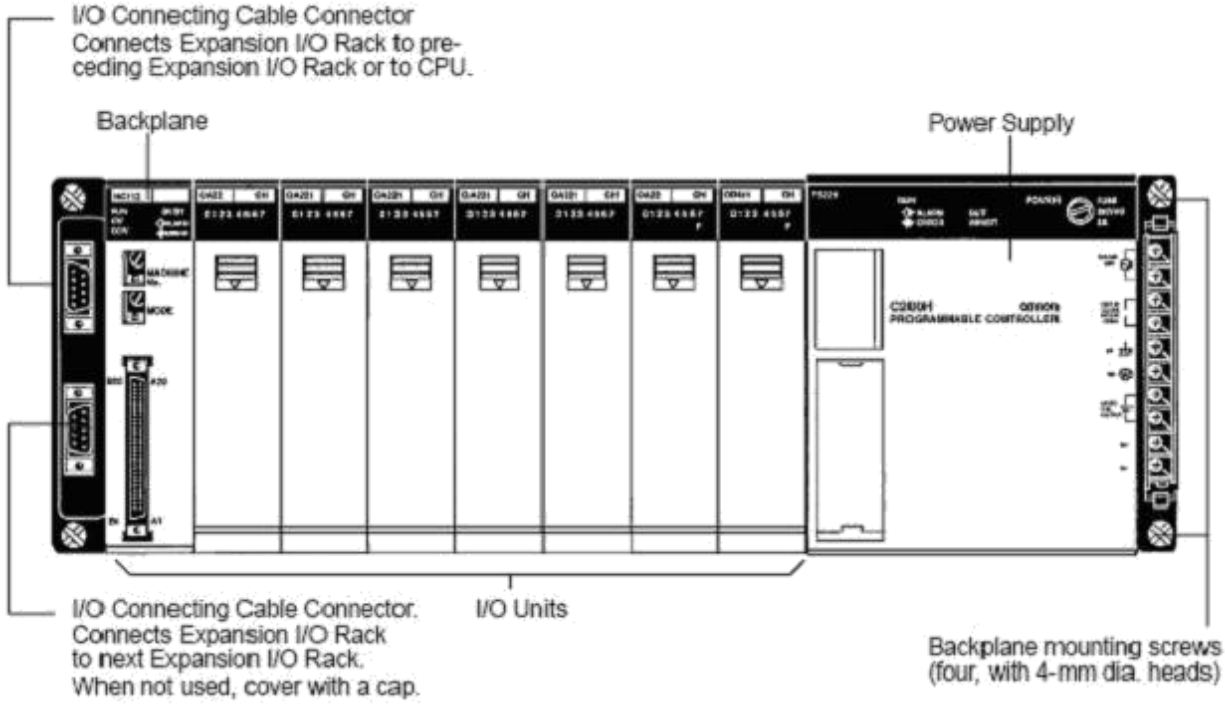
The small PLC can be installed on a horizontal surface or on a DIN track.

Lower the small PLC so that the notch on the back of the PLC catches the top of the DIN Track. Push the PC forward until the lock snaps into place. For the big PLC before installing, the Units have to compiled one by one. There is no single Unit that can be said to constitute a Rack PLC. To build a Rack PLC, we start with a Backplane The Backplane is a simple device having two functions. The first is to provide physical support for the Units to be mounted to it. The second is to provide the connectors and electrical pathways necessary for connecting the Units mounted to it. The core of the PLC is the CPU. The CPU contains the program consisting of the series of steps necessary for the control task. The CPU has a built-in power supply, and fits into the rightmost position of the Backplane.The CPU of the big PLC has no I/O points built in. So, in order to complete the PLC we need to mount one or more I/O Units to the Backplane. Mount the I/O Unit to the Backplane by locking the top of the I/O Unit into the slot on the Backplane and rotating the I/O Unit downwards as shown in the following diagram. Press down on the yellow tab at the bottom of the slot, press the I/O Unit firmly into position, and then release the yellow tab.

3. Installing the Expansion Unit or Expansion I/O Unit

The Expansion Unit or Expansion I/O Unit are usually attached when amount of I/O devices to be controlled increase its amount over than capacities of the existing I/O Unit or attached when needed to a special need like temperature sensor. The following shown the example of ExpansionUnits.





For the small PLC use the following procedure when connecting an Expansion Unit or Expansion I/O Unit;Remove the cover from the CPU Unit's or the Expansion I/O Unit's expansion connector. Use a flat-blade screwdriver to remove the cover from the Expansion I/O Connector.Insert the Expansion I/O Unit's connecting cable into the CPU Unit's or the Expansion I/O Unit's expansion connector. Replace the cover on the CPU Unit's or the Expansion I/O Unit's expansion connector.

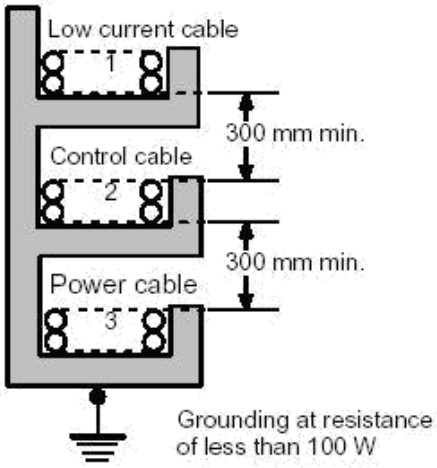
4. Installing I/O devices

I/O devices are attached at the place have been determined in the work plan and wiring diagram. For switches are usually attached at the panel while the sensor, selenoid and motor is usually placed at the machine to be controlled.

5. Wiring and connections

Duct Work

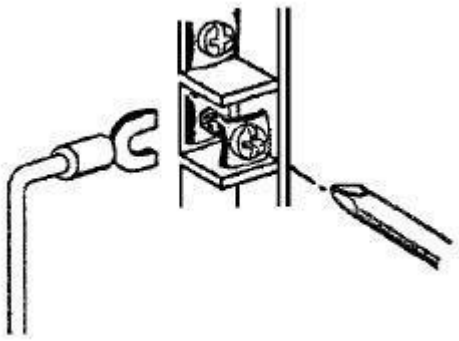
Hanging Ducts If power cables carrying more than 10 A 400 V, or 20 A 220 V must be run alongside the I/O wiring (that is, in parallel with it), at least 300 mm must be left between the power cables and the I/O wiring as shown below.



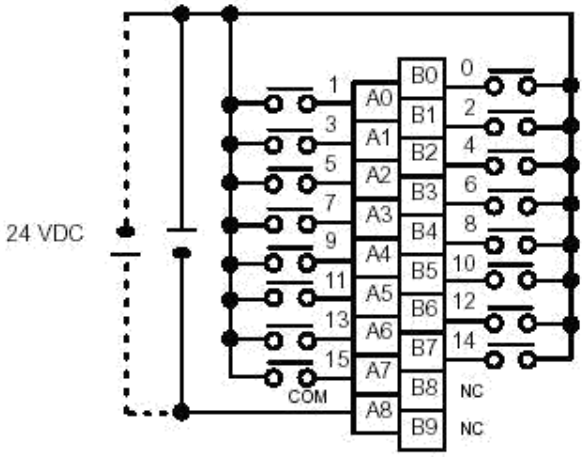
Where: 1 = I/O wiring
2 = General control wiring
3 = Power cables

I/O connections

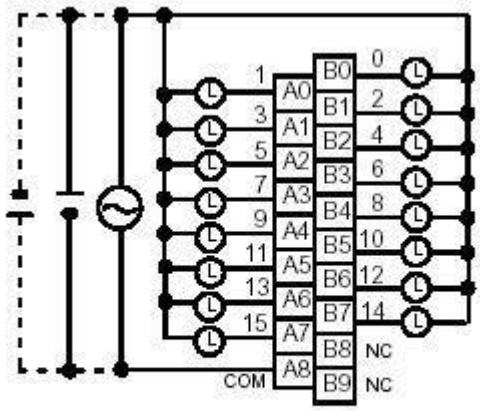
Connect the I/O Devices to the I/O Units. Use 1.25-mm2 cables or larger The terminals have screws with 3.5-mm diameter heads and self -raising pressure plates. Connect the lead wires to the terminals as shown below. Tighten the screws with a torque of 0.8 N _ m.



The following diagrams show the input configurations. This input configuration depend on specification of the Input Unit will be used.



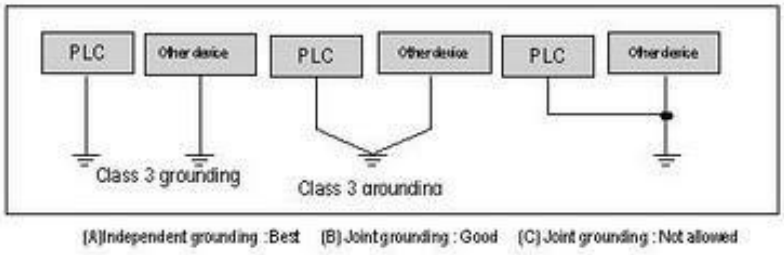
The following diagrams show the output configurations. This output configuration depend on specification of the Output Unit will be used. See the specification before install.



250 VAC 24 VDC max.
(inductive load: 2 A resistive load: 2 A) (8 A/Unit)

Grounding

This PLC has sufficient protection against noise, so it can be used without grounding except for special much noise. However, when grounding it should be done conforming to below items. Ground the PLC as independently as possible. Class 3 grounding should be used (grounding resistance 100Ω or less). When independent grounding is impossible, use the joint grounding method as shown in the figure below (B). Use thicker grounding wire. Grounding point should be as near as possible to the PLC to minimize the distance of grounding cable.



(A)Independent grounding :Best (B)Joint grounding : Good (C)Joint grounding : Not allowed

MAINTANENCE

1. Applying the safety procedure

During execution of the work, the safety procedure must be executed truly so that the risk of the work accident can be avoided.

Example of applying the safety procedure;

- Use the safety equipment
- Follow the instruction of safety procedure
- Comprehending fringe of writing on the wall or emergency

2. Checking the installation and power supply

- To do the maintenance and reparation of the PLC system, one of the important matter that must be done is perform the inspection to the PLC installation as according to the manual instruction

Preventive maintenance

The main system components of a PLC system are semiconductors, and it contains few components with limited lifetimes. Poor environmental conditions, however, can lead to deterioration of the electrical components, making regular maintenance necessary. The standard period for maintenance checks is 6 months to 1 year, but more frequent checks are required if the PLC is operated in more demanding conditions

Preventive maintenance consisted of several activity below;

Pre maintenance

Pre maintenance is a preparation activity, matters which require to be prepared for example;

- Prepare the maintenance equipment
- Prepare the maintenance material especially weared routinely, for example; cleanser material, Iubricant material, corrosion preventative material, etc.
- Prepare the maintenance documentation
- Prepare the power supply and air compressor

Daily maintenance

The following table shows the inspection and items which are to be checked daily.

Periodic Maintenance

- Turn OFF the power to the PLC.
- Detach the terminal block by unlocking the lock levers at the top and bottom of the terminal block.
- While pushing down the lock lever on the Backplane with a screwdriver as shown below, remove the Output Unit.
- Remove the screw from the top of the Unit (Phillips screwdriver).
- Detach the case from the Unit (flat-blade screwdriver).
- Pull out the printed circuit board.
- Insert a new fuse. A spare fuse is provided inside the rear of the case when the Unit is delivered.
- Reassemble in reverse order of assembly.

Output Unit Relays

To replace a Relay, follow the steps below:

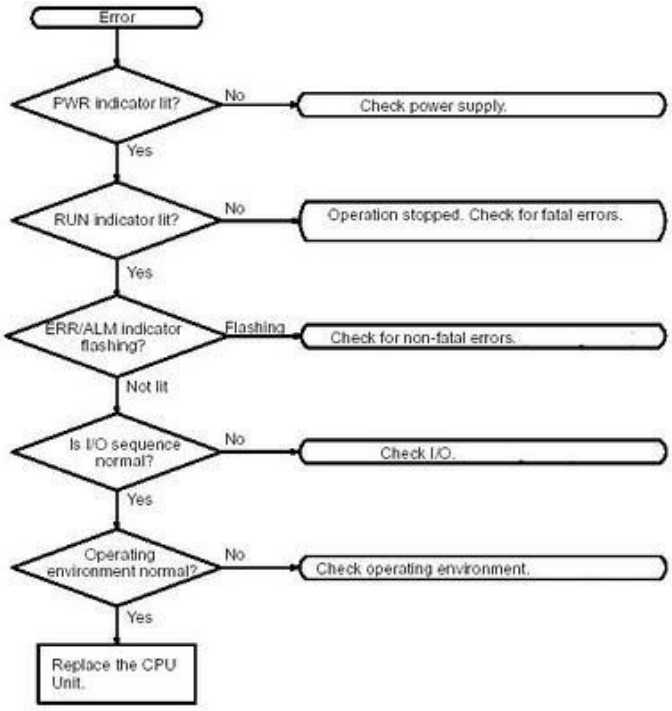
- Turn OFF the power to the PLC.
- Detach the terminal block by unlocking the lock levers at the top and bottom of the terminal block.
- While pushing down the lock lever on the Backplane with a screwdriver, remove the Output Unit.

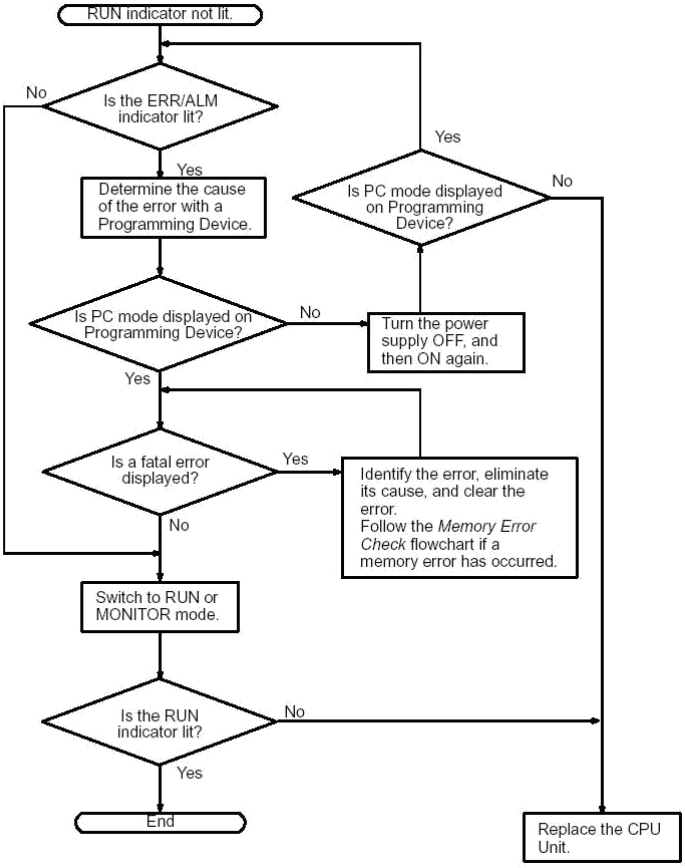
Batteries

- Some RAM Packs use a battery. When the battery is nearly discharged, the ALARM indicator blinks and the message "BATT FAIL" appears on the Programming Console. When this occurs, replace the battery within one week to avoid loss of data. The battery comes together with its connector as a set. To replace the Battery Set, follow the steps below. The entire replacement must be completed within five minutes to ensure that the data will not be lost.

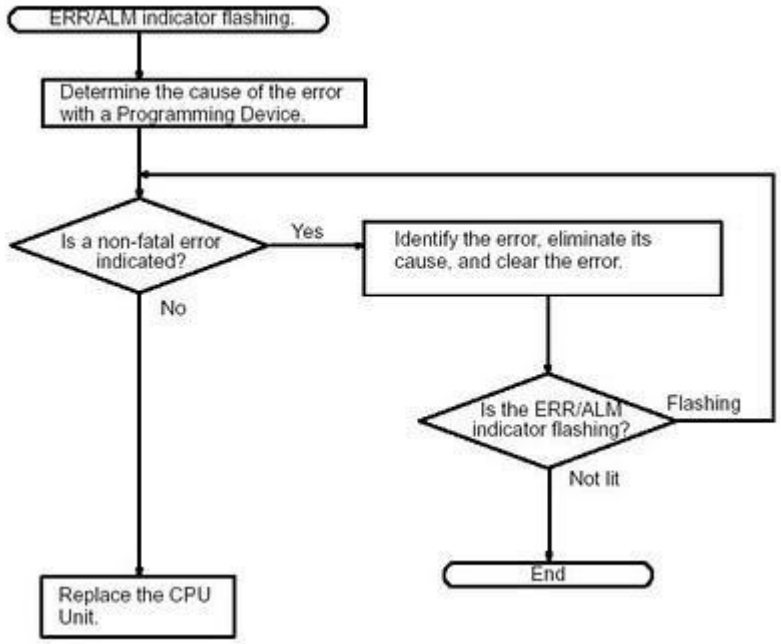
PLC Troubleshooting

The following example explains the procedure for determining the cause of troubles as well as the errors and corrective actions to the Omron PLC. Use the following flowcharts to troubleshoot errors that occur during operation.



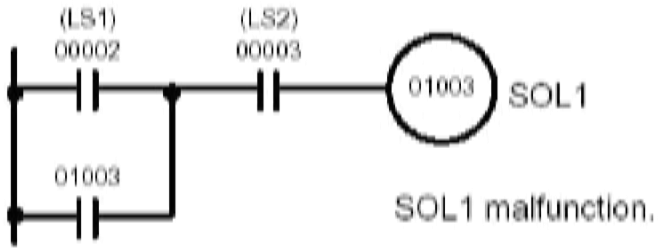


3. Non- Fatal Error Check

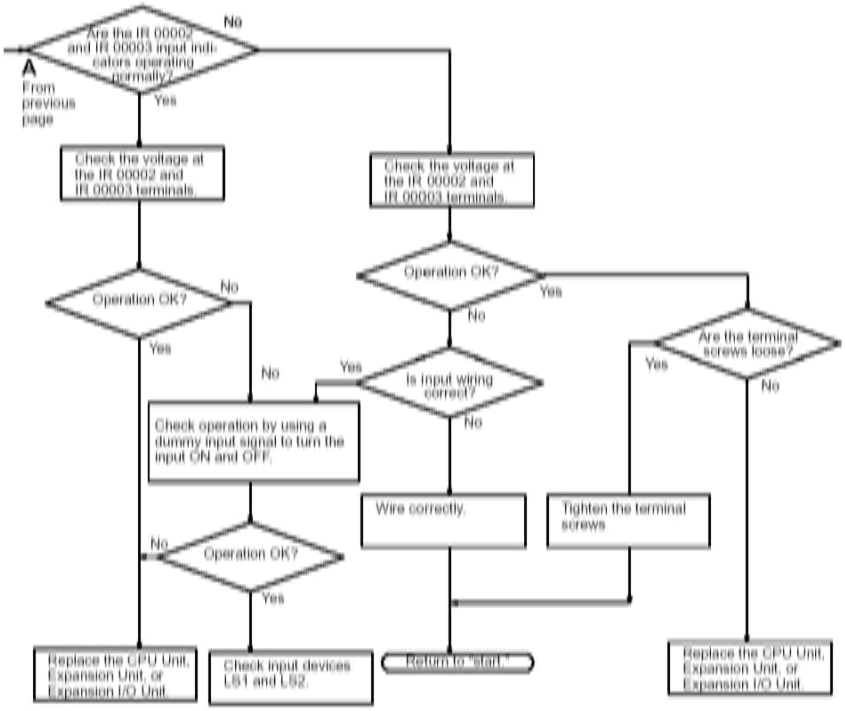


4. I/O Check

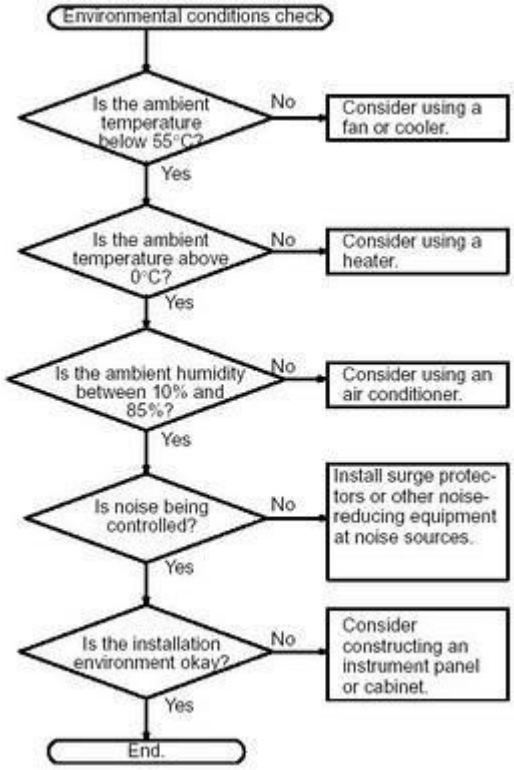
The I/O check flowchart is based on the following ladder diagram section.


(LS1)
00002
(LS2)
00003
01003
SOL1
01003
SOL1 malfunction.

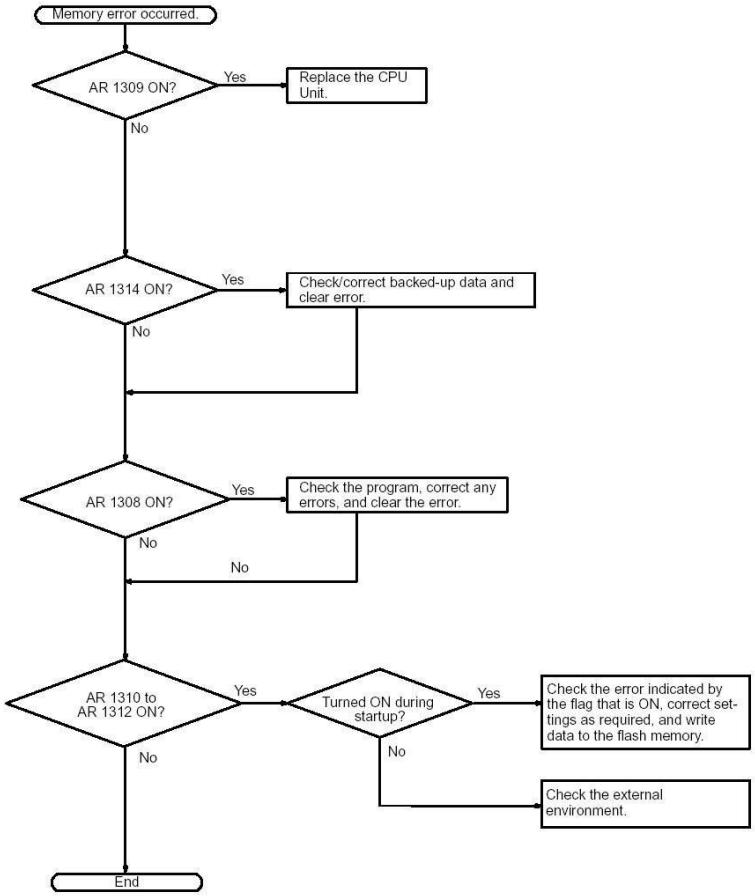

Malfunction of SOL1
Is the IR 01003 output indicator operating normally?
No
Yes
Check the voltage at the IR 01003 terminals.
Wire correctly.
Correct the short circuit or limit the connected load to the specified range.
Monitor the ON/OFF status of IR 01003 with a Programming Device.
Operation OK?
No
Is output wiring correct?
No
Yes
Operation OK?
No
A
To next page
Yes
Yes
Is the external power supply shorted or overloaded?
Yes
No
Disconnect the external wires and check the conductivity of each wire.
Yes
Operation OK?
No
Check output device SOL1.
Replace the CPU Unit, Expansion Unit, or Expansion I/O Unit with the problem.


A
From previous page
Are the IR 00002 and IR 00003 input indicators operating normally?
No
Yes
Check the voltage at the IR 00002 and IR 00003 terminals.
Check the voltage at the IR 00002 and IR 00003 terminals.
Operation OK?
No
Yes
Operation OK?
Yes
No
Are the terminal screws loose?
Yes
No
Yes
Is input wiring correct?
No
No
Check operation by using a dummy input signal to turn the input ON and OFF.
Wire correctly.
Tighten the terminal screws
No
Operation OK?
Yes
Replace the CPU Unit, Expansion Unit, or Expansion I/O Unit.
Check input devices LS1 and LS2.
Return to "Start."
Replace the CPU Unit, Expansion Unit, or Expansion I/O Unit.

5. Environmental Conditions Check



6. Memory Error Check



ALTERNATE PROGRAMMING LANGUAGES

The IEC 1131-3 standard defines two graphical languages and two text-based languages for use in PLC programming. The graphical languages use symbols to program control instructions, while the text-based languages use character strings to program instructions.

Graphical languages

- ladder diagrams (LD)
- function block diagram (FBD) *Text-based languages*
- instruction list (IL)
- structured text (ST)

Additionally, the IEC 1131-3 standard includes an object-oriented program-ming framework called **sequential function charts (SFCs)**. SFC is sometimes categorized as an IEC 1131-3 language, but it is actually an organizational structure that coordinates the standard's four true program-ming languages (i.e., LD, FBD, IL, and ST). The SFC structure is much like a flowchart-type of programming framework, utilizing different languages for different control tasks and also routing control program actions. The SFC structure has its roots in the early French standard of Grafcet (IEC 848).

The IEC 1131-3 standard is a graphic/object-oriented block programming method, which increases the programming and troubleshooting flexibility of its programmable controllers. It allows sections of a program to be individu-ally grouped as tasks, which can then be easily interlocked with the rest of the program. Thus, a complete IEC 1131-3 program may be formed by many small task programs represented inside SFC graphic blocks. The combination of languages available in the IEC 1131-3 standard also enhances PLC programming and troubleshooting by providing not only a better program-ming language, but also a better method for implementing control solutions.

The IEC 1131-3 uses a wide variety of standard data functions and function blocks, which operate on a large number of data variable types. Table 10-1 shows some examples of these data types and functions, as well as some typical function blocks. *Data variable type* refers to the kind of data received by the controller (e.g., binary, real numbers, time data, etc.), while *datafunctions* are the operations performed on the data (e.g., comparison, invert,addition, etc.). *Function blocks* are sets of data function instructions that work on blocks of data. Moreover, *variable scope* refers to the extent that a variable can be used in an application. For example, global variables can be used by any program in an application, while local variables can only be used by one particular program. Note that, in addition to the standard types of variables, functions, and blocks, the IEC 1131-3 allows for other types of vendor- and user-defined PLC programming elements. Thus, the IEC 1131-3 does not specify a set number of programming features, but rather establishes the groundwork for standard and additional functions.

2.9 Field Bus and HART Protocol

Field bus: Introduction ,Concept of Field bus technology and layers, International field bus standards,
Field bus for use in industrial control, HART protocol: method of operation, Structure of HART protocol, Operating conditions , Industrial Applications.

2.9.1 HART Communication Protocol

HART (Highway Addressable Remote Transducer) is a smart (intelligent) instrumentation protocol provides digital communication to analog process control instruments. It is designed for applications where actual data is collected from instruments, sensors, and actuators by digital communication techniques. It was the first bi-directional digital communication scheme for process transmitters that didn't disturb the analog signal. The process could be left running during communication.
The range of HART applications is broadened due to the recent HART development, the Device Description Language (DDL), provides a universal software interface to new and existing devices.

HART in a Process Control system

In a process control system the function of process transmitter is to transmit the changes in the process variable in terms of 4-20mA signal to the controller. The controller detects this current variation by measuring the voltage across the current sense resistor.

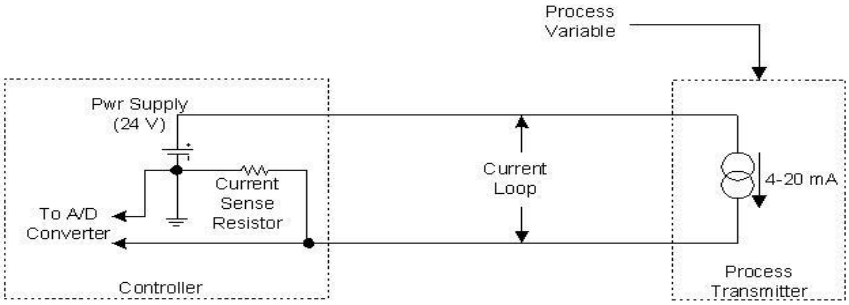


Figure 2.1 Simple Process Control Loop

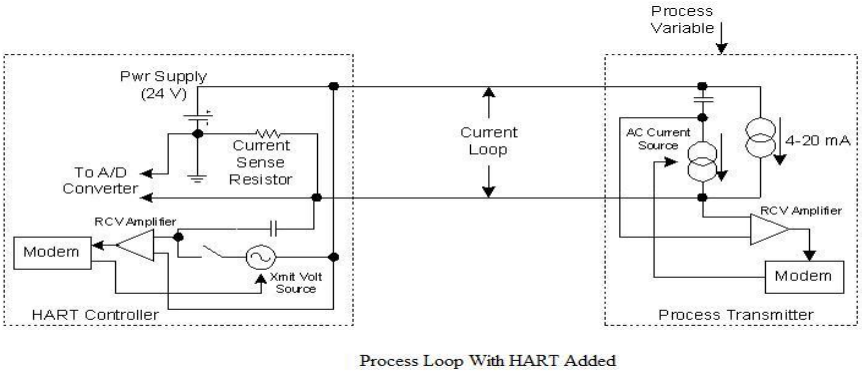


Figure 2.2 Process Control Loop with HART added

Figure above shows a process control system with HART added. Both the transmitter and the receiver has now include a “modem” and a "receive amplifier". The process transmitter also has an AC-coupled current source, and the controller an AC-coupled voltage source. The switch in series with the voltage source in the HART controller is normally open.
To send a HART message, the process transmitter turns ON its AC-coupled current source. When it is turned ON it superimposes a high-frequency carrier current of about 1 mA onto the normal transmitter output current. The current sense resistor at the controller converts this variation into a voltage. The voltage is sensed by the controller's receive amplifier and fed to the controller's modem which demodulates the signal. To send a HART message in the other direction ie., from the controller to the process transmitter, the HART Controller closes its transmit switch. This can be achieved by superimposing a voltage of about 500 mV to the controller signal.

- The most important performance features of the HART protocol include:
- proven in practice, simple design, easy to maintain and operate
- compatible with conventional analog instrumentation
- simultaneous analog and digital communication
- option of point-to-point or multi-drop operation
- flexible data access via up to two master devices
- supports multivariable field devices
- sufficient response time of approx. 500 ms
- open de-facto standard freely available to any manufacturer or user Devices which support the HART protocol is grouped into
- master (host) and
- slave (field) devices.

Master devices include handheld terminals as well as PC-based work places, e.g. in the control room. HART slave devices, on the other hand, include sensors, transmitters and various actuators.
The HART data is superimposed on the 4 to 20 mA signal via a FSK modem. This enables the devices to communicate digitally using the HART protocol, while analog signal transmission takes place at the same time HART communication is often used for such simple point-to-point connections.

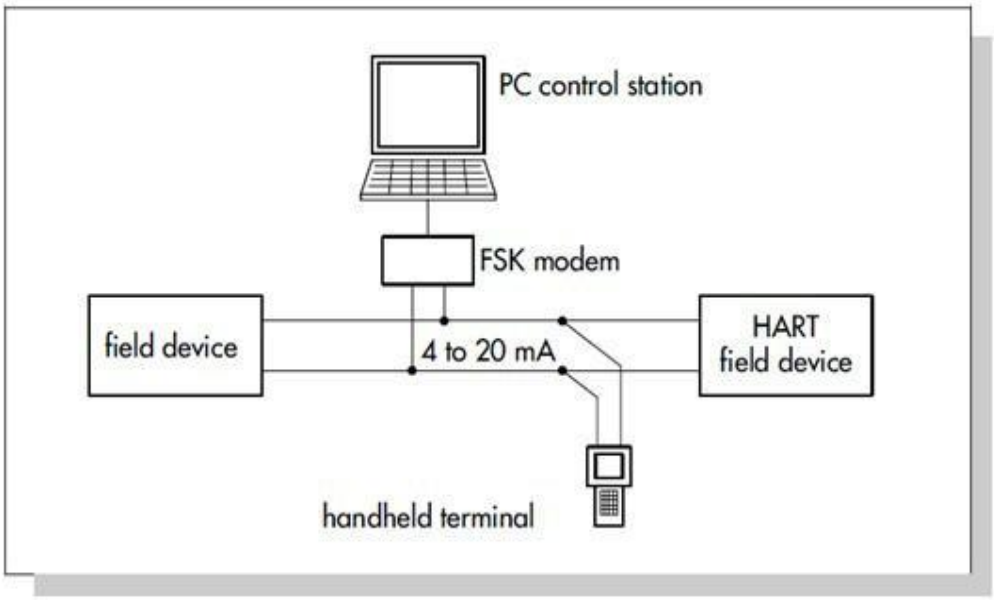


Figure 2.3 Connection of HART Devices

HART Communication Protocol

OSI layers	HART layers
application	HART commands
presentation	
session	
transport	
network	
data link	HART protocol rules
physical layer	Bell 202

Figure 2.4 HART Communication Protocol

The HART protocol utilizes the OSI reference model. The HART protocol implements only the layers 1, 2 and 7 of the OSI model. The layers 3 to 6 remain empty since their services are either not required or provided by the application layer 7.

Functions of the HART Layers
Physical layer
Coding

Data transmission between the masters and the field devices is physically done by superimposing an encoded digital signal on the 4 to 20 mA current loop. Since the coding has no mean values, an analog signal transmission taking place at the same time is not affected. This enables the HART protocol to include the existing simplex channel transmitting the current signal and an additional half-duplex channel for communication in both directions.
To encode the bits, the FSK method (Frequency Shift Keying). The two digital values 0 and 1 are assigned to the following frequencies
Logical 0 2200Hz
Logical 1 1200Hz

Data Rate

Transmit data at the rate of 1200 bits/s.
The HART specification defines that master devices send voltage signals while the field devices (slaves) convey their messages using load-independent currents. The current signals are converted to voltage signals at the internal resistance of the receiver (at its load). To ensure a reliable signal reception, the HART protocol specifies the total load of the current loop including the cable resistance to be between minimum 230 ohms and maximum 1100 ohms.
Connection type

The HART masters are simply connected in parallel to the field devices so the devices can be connected and disconnected during operation because the current loop need not be interrupted.

Wiring

HART wiring in the field usually consists of twisted pair cables. If very thin and/or long cables are used, the cable resistance increases and, hence, the total load. As a result, the signal attenuation and distortion increases while the critical frequency of the transmission network decreases.

Services of layer 2

Access control

The HART protocol operates according to the master-slave method. Any communication activity is initiated by the master, which is either a control station or an operating device. HART accepts two masters, the

- primary master usually the control system
- secondary master a PC laptop or handheld terminal used in the field.

HART field devices the slaves never send without being requested from the master. They respond only when they have received a command message fromthe master . Once a transaction, i.e. a data exchange between thecontrol station and the field device, is complete, the master will pause for a fixed time period before sending another command, allowing the other masterto break in.

.

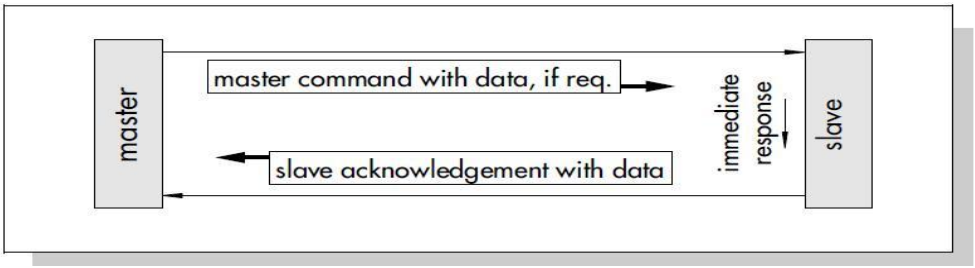


Figure 2.6 HART transaction: data exchange between master and slave HART

Communication modes

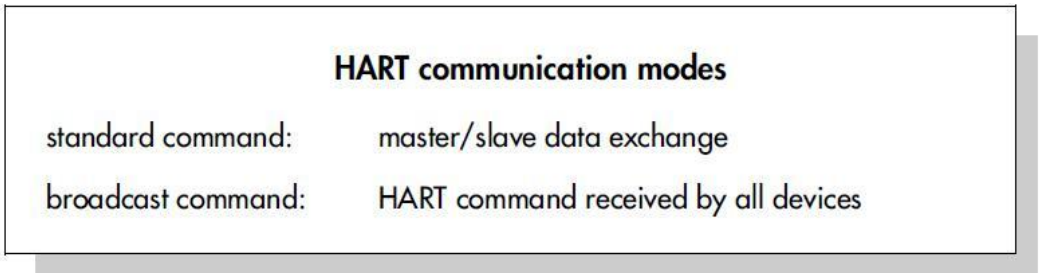


Figure 2.7 HART Communication modes Master /slave data exchange

The simplest form of a transaction is a master telegram which is directly followed by a response or acknowledgement telegram from the slave. This communication mode is used for the normal data exchange.

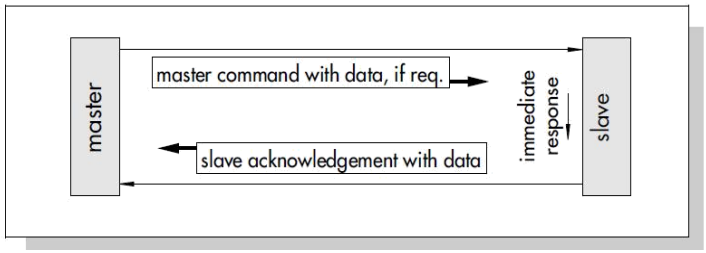


Figure 2.8 Master /slave data exchange

When connection is established, the HART command 11 can be used to send a broadcast message to all devices to check the system configuration.

Burst Communication mode

Some HART devices support the optional burst communication mode. A single **field device cyclically sends message telegrams with short 75-ms breaks,** alternately to primary as well as the secondary master.

Point-to-point connection

The HART master device is connected to exactly one HART field device.

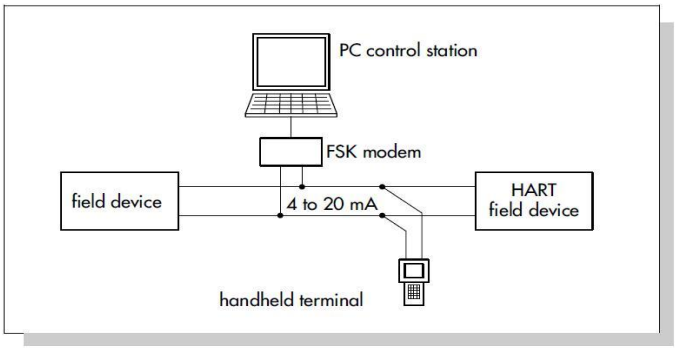


Figure 2.9 Point-to-point connection

HART Networks

The number of accessible devices can be increased by using a multiplexer. Network variants include

- multi drop
- FSK bus and
- networks for split-range operation.

Multi-drop

This enables a large number of HART devices to be connected in a network using a multiplexer. The user selects a particular current loop for communication. As long as the communication takes place, the multiplexer connects the current loop to the host. Due to the cascaded multiplexer structure, the host can communicate with many (> 1000) devices, all with the address zero.

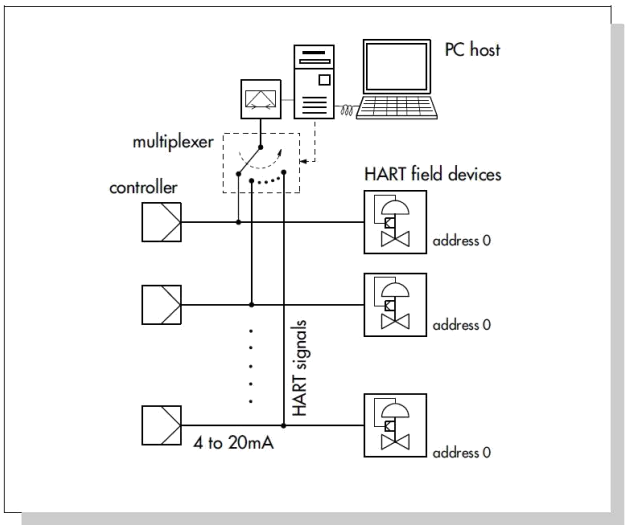


Figure 2.10 Field device connected through Multiplexer

Multidrop mode

In multidrop mode, up to 15 field devices are connected in parallel to a single wire pair. The host distinguishes the field devices by their preset addresses which range from 1 to 15. In multidrop operation, the devices exchange their data and measured values only via the HART protocol.

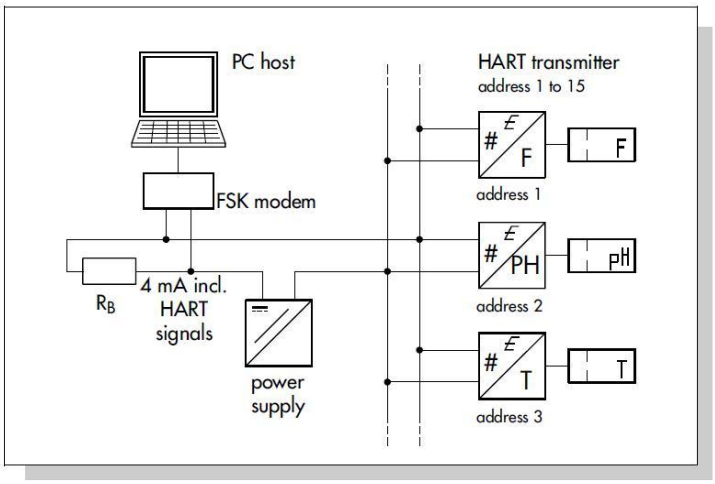


Figure 2.11 Multidrop mode

FSK bus

The HART protocol can be extended by company-specific functions. It can connect approximately 100 HART devices and address them. This requires special assembly-type isolating amplifiers.

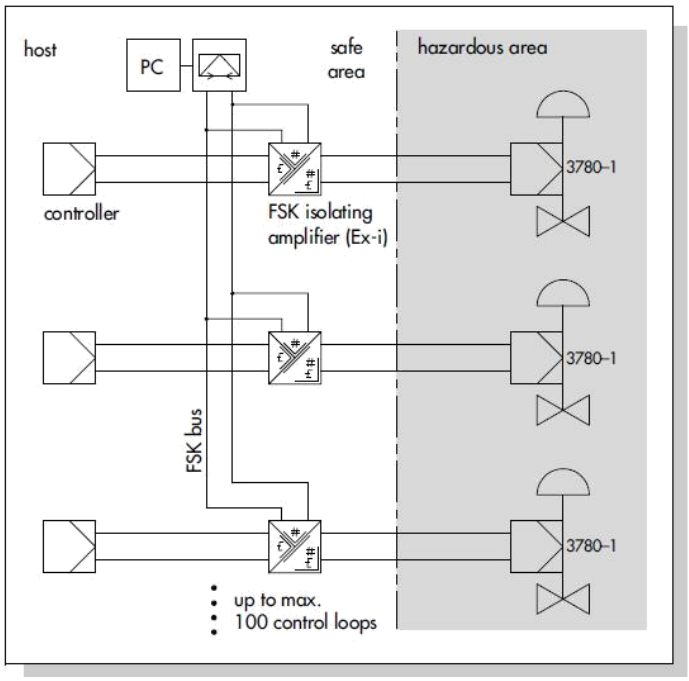


Figure 2.12 FSK bus

The HART devices are connected to their analog current signal and the common FSK bus line via the isolating amplifier. The isolating amplifiers act as impedance converters. This enables also devices with high load to be integrated in the communication network. To address the devices, a special, long form of addressing is used. During the configuration phase, the bus address and the tag number of each device are set via the point-to-point line. During operation, the devices operate with the long addresses.

Bus for split-range operation

There are special applications which require that several usually two actuators receive the same control signal. A typical example is the split-range operation of control valves.

- One valve operates in the nominal current range from 4 to 12 mA,
- the other one uses the range from 12 to 20 mA.

In split -range operation, the control valves are connected in series in the current loop. When both valves have a HART interface, the HART host device must be able to distinguish with which valve it must communicate.To be able to use HART communications also for such applications as the split-range operation, the HART positioner always takes the analog current signal as a reference variable, independent of the device address.

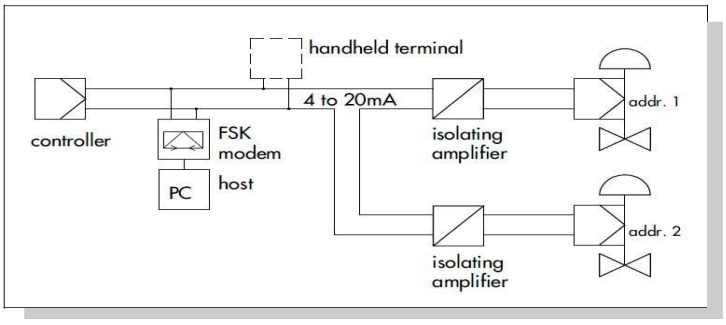


Figure 2.13 Bus for split-range operation

HART Message format

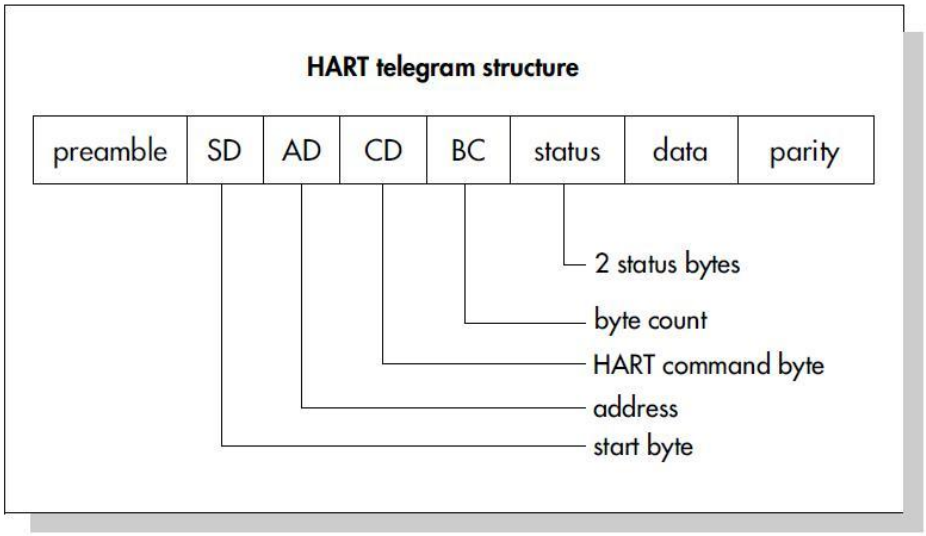


Figure 2.14 HART Message format

The elements of the HART telegram perform the following tasks:

- The preamble consisting of three or more hexadecimal characters synchronizes the signals of the participants.
- The start byte indicates which participant is sending (master, slave, slave in burst mode) and whether the short frame or the long frame format is used.
- The address field of the short frame format contains one byte with one bit serving to distinguish the two masters and one bit to indicate burst-mode telegrams. For the addressing of the field devices, 4 bits are used (addresses 0 to 15).
- The address field of the long frame format contains five bytes, hence, the field device is identified using 38 bits.
- The command byte encodes the master commands of the three categories, Universal, Common-practice and Device-specific commands. The significance of these commands depends on the definitions in the application layer 7.
- The byte count character indicates the message length, which is necessary since the number of data bytes per telegram can vary from 0 to 25. This is the only way to enable the recipient to clearly identify the telegram and the checksum. The number of bytes depends on the sum of the status and the data bytes.
- The two status bytes are included only in reply messages from slaves and contain bit-coded information. They indicate whether the received message was correct and the operational state of the field device. when the field device operates properly, both status bytes are set to logical zero.
- The data can be transmitted as unsigned integers, floating-point numbers or ASCII-coded character strings. The data format to be used is determined by the command byte, however, not all commands or responses contain data.
- The checksum byte contains the longitudinal parity of all the bytes of a telegram.

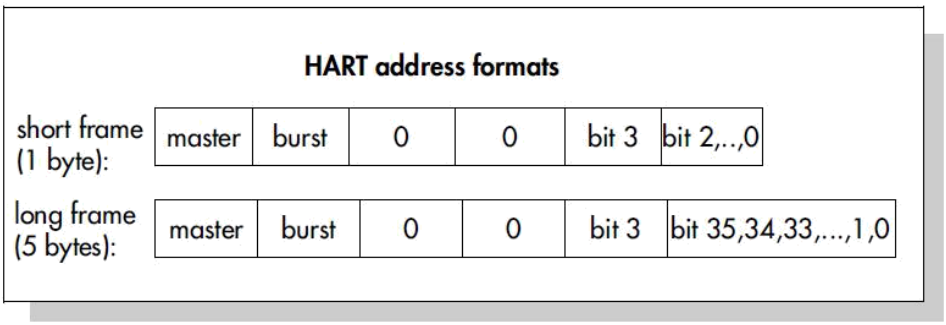


Figure 2.15 HART address format Application layer: HART commands

The communication routines of HART master devices and operating programs are based on HART commands which are defined in the application layer of the HART protocol.
Pre-defined commands enable the master device to give instructions to a field device or send messages/data. So set points, actual values and parameters can be transmitted and various services for start-up and diagnostics performed.
The field devices immediately respond by sending an acknowledgement telegram which can contain requested status reports and/or the data of the field device.

HART COMMANDS

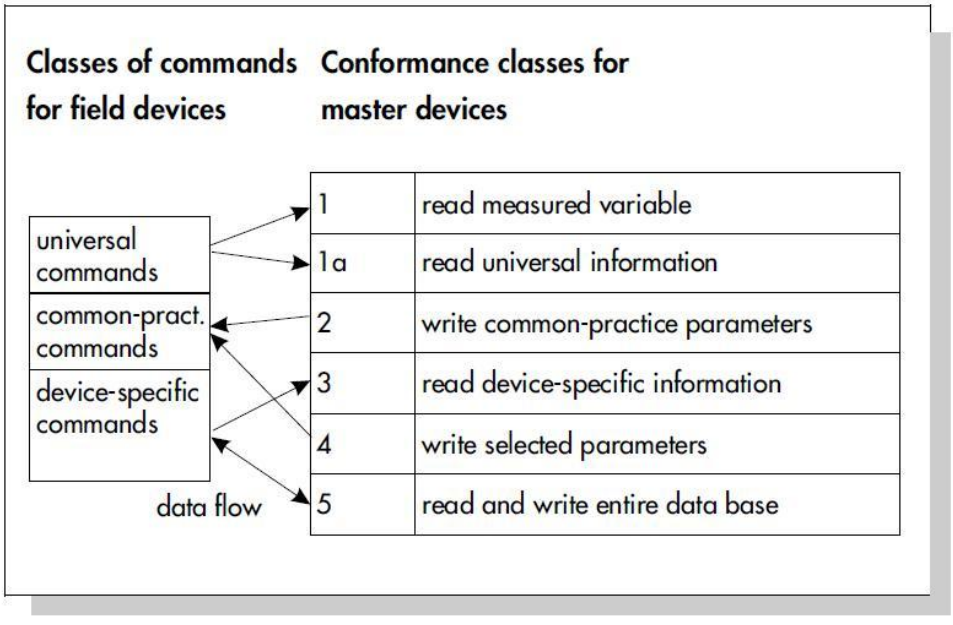


Figure 2.16 HART COMMANDS

- Depending on the tasks to be executed, the HART master device uses a command that can be assigned to one of the six different conformance classes. Each conformance class contains a subset of HART commands which cover a special administrative or control-related range of tasks.
- Field devices interpret and process only those HART commands that are directed to them or to all participants. Each command belongs to one of three classes of commands.
- **Universal commands** are understood and used by all field devices operating with the HART protocol (device designation, firmware no., etc.).

- **Common-practice commands** are usually supported by many, but not necessarily all, HART field devices. (Read variable, set parameter, etc.). Most of the HART field devices are able to interpret and respond to common-practice commands.
- **Device-specific commands** support functions that are unique to each device.These commands provide access to data about the type and construction of a device as well as information on the maintenance state and start-up
- Most of the field devices support commands of all three classes: they understand all universal commands, the common-practice commands tailored to them and special, device- and manufacturer-specific commands.

Universal Commands
HART-Command 0 : Read Transmitter Unique Identifier
HART-Command 1 : Read Primary Variable
HART-Command 2 : Read Current and Percent of Range
HART-Command 3 : Read all dynamic Variables and Current
HART-Command 6 : Write Polling Address
HART-Command 11 : Read Unique Identifier Associated With Tag
HART-Command 12 : Read Message
HART-Command 13 : Read Tag, Descriptor, Date
HART-Command 14 : Read Primary Variable Sensor Information
HART-Command 15 : Read Primary Variable Output Information
HART-Command 16 : Read Final Assembly Number
HART-Command 17 : Write Message
HART-Command 18 : Write Tag, Descriptor, Date
HART-Command 19 : Write Final Assembly Number

Commom Practice Commands
HART-Command 34 : Write Primary Variable Damping Value
HART-Command 35 : Write Primary Variable Range Values
HART-Command 38 : Reset Configuration Changed Flag
HART-Command 40 : Enter/Exit Primary Variable Current Mode
HART-Command 45 : Trim Primary Variable Current DAC Zero
HART-Command 46 : Trim Primary Variable Current DAC Gain
HART-Command 48 : Read Additional Transmitter Status
Slot - Commands
Unsigned-char-Variable
HART-Command 128 : Read unsigned-char-variable
HART-Command 129 : Write unsigned-char-variable
Unsigned-int-variables
HART-Command 130 : Read unsigned int-variable
HART-Command 131 : Write unsigned-int-variable
Float-variable
HART-Command 132 : Read float-variable
HART-Command 133 : Write float-variable
String-variables
HART-Command 134 : Read string-variable
HART-Command 135 : Write string-variable
Other Commands
HART-Command 140 : Clear totalizer and overflow
HART-Command 150 : Initialization of internal and external database

HART-Command 151 : Download of internal into external database
HART-Command 165 : Read settings "Progr. Output"
HART-Command 166 : Write settings "Progr. Output"

2.9.2 INDUSTRIAL APPLICATIONS OF HART PROTOCOL

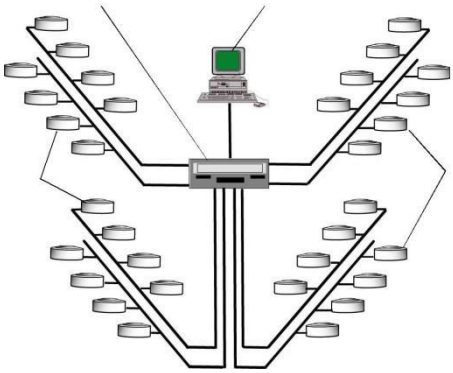
Hart multidrop network for tank level and inventory management

Accurate measurements for inventory management are essential in all industries. The HART communication protocol enables companies to make sure inventory management is as efficient, accurate, and low cost as possible. Tank level and inventory management is an ideal application for a HART multidrop network . The HART network digital update rate of two PVs per second is sufficient for many tank-level applications. A multidrop network provides significant installation savings by reducing the amount of wiring from the field to the control room as well as the number of I/O channels required. In addition, many inexpensive process-monitoring applications are commercially available to further cut costs. A HART multiplexer is used to digitally scan field devices for level-measurement and status information. The information is forwarded to the host application using the Mod bus communication standard. Multivariable instruments further reduce costs by providing multiple process measurements, such as level and temperature, which reduces the wiring and number of process penetrations required.

MULTIDROP FOR TANK FARM MONITORING

In one tank farm application, 84 settlement tanks and filter beds on a very large site (over 300,000 m2) are monitored using HART multidrop networks. The HART architecture required just eight cable runs for 84 tanks, with 10–11 devices per run . Over 70 individual runs of over 500 m each were eliminated. Cable savings were estimated at over $40,000 when compared to a conventional installation. RTU I/O was also reduced, which resulted in additional hardware and installation savings. The total installed cost was approximately 50% of a traditional 4–20 mA installation.

Control Room



Tank Farm Monitoring with Multidrop

Figure 2.17 Multi drop for tank farm monitoring

WASTEWATER TREATMENT PLANT UPGRADE

A Texas wastewater treatment plant replaced stand-alone flow meters and chart recorder outstations that required daily visits for totalization with a HART system. HART- based

magnetic flow meters were multi dropped into HART RTUs to create a cost -effective SCADA network. The use of HART technology reduced system and cable costs, enhanced measurement accuracy, and eliminated time-consuming analog calibration procedures. A system of 11 HART multi drop networks was used to connect 45 magnetic flow meters from different plant areas. Each flow meter communicated flow rate and a totalized value over the HART network. Multi drop networks eliminated the need for additional hardware and PLC programming while providing a more accurate totalized value. Complex and costly system integration issues were also avoided—for example, there was no need for synchronization of totals between the host and field PLCs. Multi drop networking further reduced the installation cost by reducing the required number of input cards from the traditional 45 (for point -to-point installations) to 11. Maintenance was simplified because of access to instrument diagnostic and status data.

REMOTE REZEROING IN A BREWERY

The benefits of remote monitoring and rezeroing of smart transmitters using the HART protocol are dramatically illustrated in this example of two smart transmitters that control the fluid level in lauter tubs in a brewhouse application. Similar benefits would be realized in any application involving a closed vessel. Two smart transmitters are installed on each lauter tub—one on the bottom of the tank and the other about nine inches from the bottom. The bottom transmitter is ranged ±40 inwater; the upper transmitter is ranged 0–30 inH2O. As the lauter tub is filled, the bottom transmitter senses level based on pressure. When the level reaches the upper transmitter, that point is marked as the new zero-level point, and the upper transmitter becomes the primary sensing instrument for the lauter-tub level. The nine-inch zero-level offset from the bottom of the tank is necessary to accommodate loose grain that settles in the bottom of the tank.

Transmitters that are coordinated and working together control fluid level in each lauter tub to within a few barrels. However, the upper transmitter requires periodic maintenance or replacement and rezeroing. An undetected false upper-transmitter level reading can cause a tank level error of up to 40 gallons.The usual procedure for transmitter rezeroing takes about 95 minutes and has been required as frequently as twice a day. Rezeroing a transmitter using configuration software and PLC interface modules eliminates the need to locate and identify the problem at the site as well as the need for verification by control-room personnel and greatly reduces the chance for inadvertent errors. Estimated total time to rezero each transmitter is reduced to 15 minutes.Through the configuration software’s instrument-status and diagnostic capabilities, a false level indication can be automatically detected while a lauter tub fill is in progress. The affected transmitter can then be automatically rezeroed by programming logic in the programmable controller to issue the appropriate command to the instrument.

2.10 FIELD BUS

Introduction to Fieldbus Systems

Fieldbus is consisting of two terms, Field and Bus [Fieldbus Introduction]. To start, the meaning of Field, as defined in industrial world, is a geographical or contextual limited area. From the industry point of view the Field is an abstraction of the plant levels. As for the term Bus is a well-known word in computer science as a set of common line that electrically (or even optically) connects various units (circuits) in order to transfer the data among them.

The origin of the fieldbus was to replace any point-to-point links between the field devices (Field Devices are simply the Sensors and Actuators of the plant) and their controllers (PLC's) by a digital single link on which all the information is transmitted serially and multiplexed in time. We will see that the fieldbus transfers, in most cases, this information in small-sized packets in serial manner. Choosing the serial transmission has many merits in comparison with other kinds of transmission like parallel transmission. For instance, the sequential or serial transmission reduces the total required number of the connecting lines over greater distances than that of the point-to-point or even parallel transmissions.
A set of rules must be defined in order to accomplish data transfer between the units along the bus. This set of rules is called Communication Protocol or just the Protocol. This is unlike the case of the ordinary point-to-point transmission where any two connected entities send and receive data from each other whenever the data is available. The protocol is responsible for two important rules on the bus, the mechanism that any unit can acquire or size the bus, and the synchronization between those multi-units on the bus.

Concept of Field bus technology and layers

The seven layers of the OSI model are
1- **Application layer** which provides the services that are required by specific applications.
2- **Presentation layer** is responsible for the data interpretation, which allows interoperability among different equipments.
3- **Session layer** is concerned with execution of any remote actions.
4- **Transport layer** is responsible for the end-to-end communication control.
5- **Network layer** is concerned with logical addressing process of nodes and routing schemes.
6- **Datalink layer** is responsible for the access to the communication medium, and for the logical transfer of the data.
7- **Physical layer** is concerned with the way that the communication is done.

Any communication system that is based on the OSI seven layers will provide higher flexibility and compatibility with products from different vendors. Modification to the **MAP** project was necessary as the node implementation become more complex in order to support all the services of the OSI reference model. The modification allowed the short length control data packets, which occurs at high rates, to be directly transmitted through the application layer to the data link layer. The resulting field bus is referred to as a 3-layered Architecture. These layers are: the Application layer, the Datalinklayer, the Physical layer.
One may assume that the other four layers of the OSI model that are not available in the fieldbus hierarchy have disappeared along with their own functions and services. The main function of the presentation layer is to support the interoperability between different equipments which is currently done by the application layer in the field bus. The assembling and disassembling of data packets which was the function of the transport layer is done now by the datalink layer in the fieldbus network. If routers to be used in some fieldbus networks, then the routing service, which was assigned to the network layer, is done by the application layer in most cases in the fieldbus.

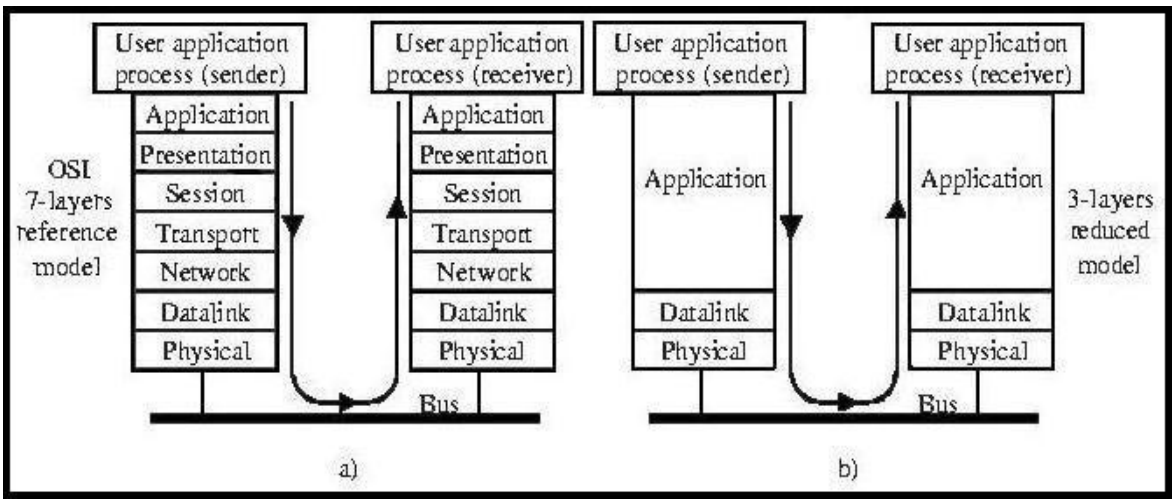


Figure 2.18 The OSI 7-layers reference model (a), and the reduced fieldbus 3-layer structure (b)

There are several protocols and services that are laid in the 3-layerd hierarchy of the fieldbus network. This will lead to a great difficulty in evaluating one and unique international fieldbus standard. There are several fieldbus protocols in the world. The designer of the DCCS communications system has multi -option solutions to fulfill his system requirements. These requirements are varied from one situation to another. In many situations the quality of services and the system throughput are the common requirements in all the DCCS systems. Also a fast response time is usually required by the real-time computer controlled networks designers.

Different Types of Fieldbuses

The fieldbus technology started at the early beginnings of the 80's in many countries. For example, the Factory Instrumentation Protocol (FIP), the Controller Area Network (CAN), and the Process Field Bus (PROFIBUS), the Process Network (P-NET) protocol, are all appeared in the same time period at the beginning of the 1980's.

CERN Recommendations on Fieldbuses

CERN is the European Organization that was established in 1954 to mainly serve the research of physics. But with the time passed the CERN became more interested in other aspects that may have some relationship with the physics. For example, the CERN involved in computer networking as the World Wide Web (**WWW**) was first developed at CERN in the beginnings of the 1990's.Now-a-days CERN is engaged in a large scale project of LHC experiments where fieldbuses is used to control and monitor the equipments of these LHC experiments. The working group finally found out that recommending one fieldbus may not satisfy all the users, so instead they had recommended three Fieldbuses. These are:

1- Controller Area Network (**CAN**).
2- Process Field Bus (**PROFIBUS**).
3- World Factory Instrumentation Protocol (**WorldFIP**).

2.10.1 Controller Area Network (CAN)

The **CAN** protocol is a priority-based bus network using a Carrier Sense Multiple Access with Collision Avoidance (**CSMA/CA**) medium access scheme. In this protocol any station can access the bus whenever it becomes idle. The collision resolution mechanism is very simple and is supported by the frame structure, or more specifically by its twelve leading bits, denoted as start bit and identifier fields. This identifier field serves for two different purposes. On one hand it is used to identify any message stream in a CAN network. Take for example a sequence of messages concerning the remote reading of a specific process variable. On the other hand, it is a priority field, which enables the collision resolution mechanism to schedule the contending messages.

This collision resolution mechanism in CAN works as follows: when the bus becomes idle, every station with pending messages will start to transmit. Due to its open-collector nature, the CAN bus acts as a big AND-gate, where each station is able to read the bus status. During the transmission of the identifier field, if a station is transmitting a "**1**" and reads a "**0**", it means that there was a collision with at least one higher-priority message, and this station aborts the message transmission at once. Thus the highest-priority message being transmitted will proceed without encountering any collision, and thus will be successfully transmitted to its destination. Obviously, each message stream must be uniquely identified.The collision resolution mechanism that is used by the CAN protocol imposes certain limitations to the bus length and its transmission data rate. For example, considering a bus length of 40m, the maximum data rate is 1Mbps.

2.10.2 Process Field Bus (PROFIBUS)

The ProfiBus communication model used to combine the distributed application processes into a common process, using communications relationships. All objects of a real device that can be communicated, such as variables, programs, data ranges are called communication objects. The process in a field device that is reachable for communication is called a virtual field device (VFD). The VFD contains the communication objects that may be manipulated by the services of the application layer.
The PROFIBUS protocol is based on a token passing procedure used by master stations to grant the bus access to each other, and a master-slave procedure used by master stations to communicate with slave stations. The PROFIBUS token passing procedure uses a simplified version of the Timed-token protocol. The medium access functions which are implemented at the layer-2 of the OSI reference model, is called Fieldbus Data Link (FDL). In addition to controlling the bus access and the token cycle time, the FDL is also responsible for the provision of data transmission services for the application layer. PROFIBUS supports four data transmission services which are: Send Data with No-Acknowledge (SDN); Send Data with Acknowledge (SDA); Request Data with Reply (RDR) and Send and Request Data (SRD). The SDN is an unacknowledged service used for broadcasts from a master station to all other stations on the bus. An important characteristic of other services is that they are immediately answered, with a response or an acknowledgement. This feature, also called "immediate-response", is particularly important for the real-time bus operation.
In addition to these services, industrial applications sometimes require the use of periodical transmission methods. A FDL-controlled polling method may be used to scan field devices, such as sensors or actuators.

2.10.3 World Factory Instrumentation Protocol (WorldFIP)

A WorldFIP network interconnects stations with two types; either a bus arbitration station or production/consumption stations. At any given time instant, only one station can be the bus arbitration station. Hence, in WorldFIP, the medium access control is centralized, and performed by the active bus arbitrator which is known as the BA. WorldFIP supports two basic types of transmission services: exchange of identified variables and exchange of messages. In WorldFIP, the exchange of identified variables is based on a Producer/Consumer model, which relates producers and consumers within a distributed system. In order to manage any transactions associated with a single variable, a unique identifier is associated with each variable. The WorldFIP Data Link Layer (DLL) is made up of a set of produced and consumed buffers, which can be locally accessed or remotely accessed through network services. In WorldFIP networks, the bus arbitrator table (BAT) regulates the scheduling of all buffer transfers. There are two types of buffer transfers that can be considered in WorldFIP. These are: periodic and aperiodic . The BAT imposes the schedule of the periodic buffer transfers, and also regulates the

aperiodic buffer transfers in the times that there are no periodic buffer transactions.

Industrial application of field bus

In compliance with hazardous area requirements, such as low power and intrinsic safety, the use of the RS485 interface with its high transmission rates is also possible in and Ex zone. This intrinsically safe bus is obtained by using an RS485 IS Coupler. The SIEMENS Fieldbus isolating transformer provides this functionality, requires no programming, and has a repeater to amplify the signal data on the bus line. Figure 1 shows the various topologies possible with RS485 IS network.

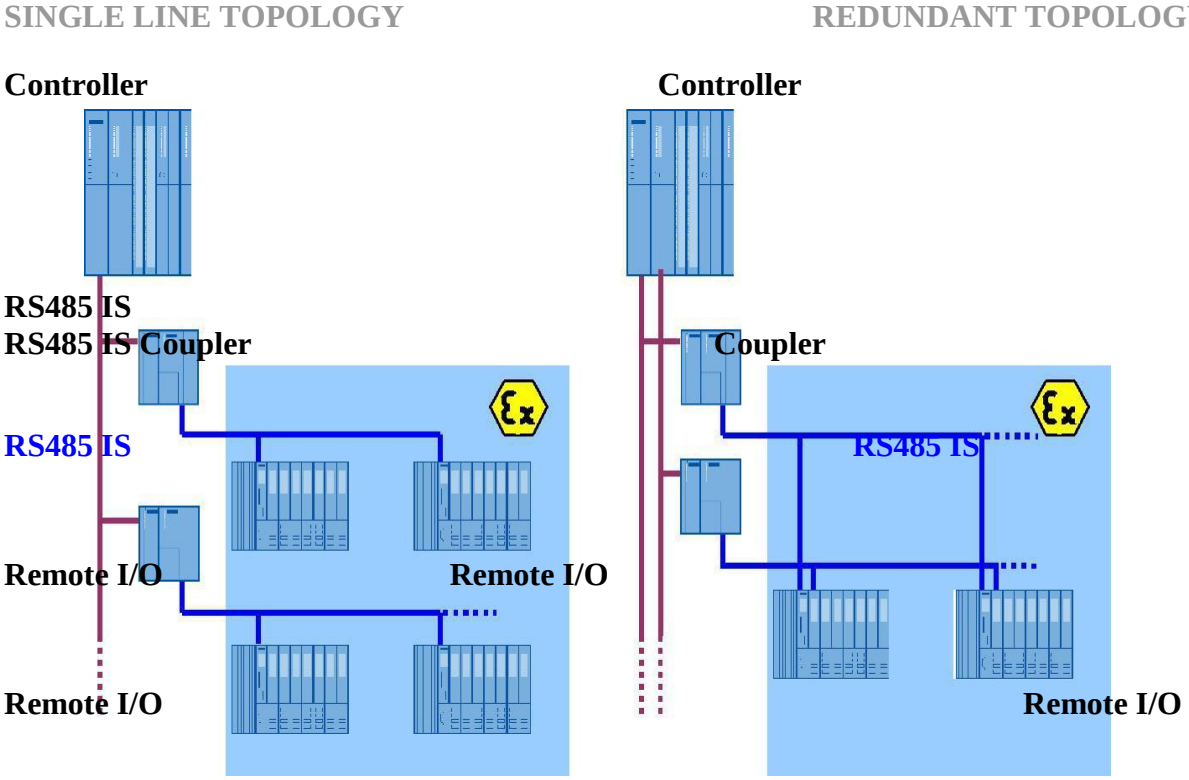


Figure 2.19 RS485 IS topologies

In the MBP-IS version, this transmission technology is especially suitable for use in hazardous areas in Process Automation, and is therefore widely used in applications of the chemical, oil and gas industries. Explosion protection is implemented via limiting power in the incoming bus supply or more frequently in the installation components in the field. Working on field devices during active operation is made possible, for example, by means of intrinsically safe ignition protection. Linear and tree structures and combinations of both are thus possible. In practice, the "trunk & spur topology" (Fig 2) has established itself as the de-facto standard, as it is especially clear and well-laid-out.

SINGLE LINE TOPOLOGY

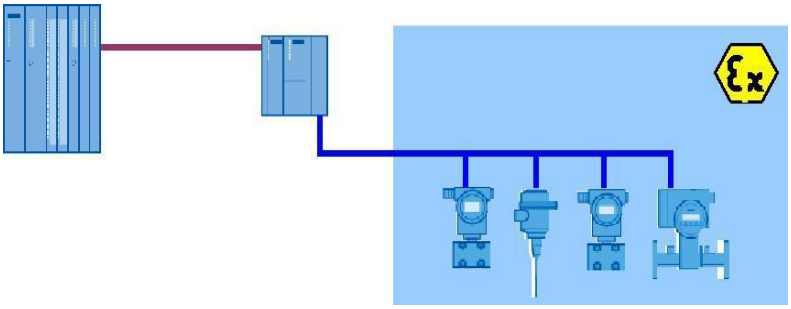
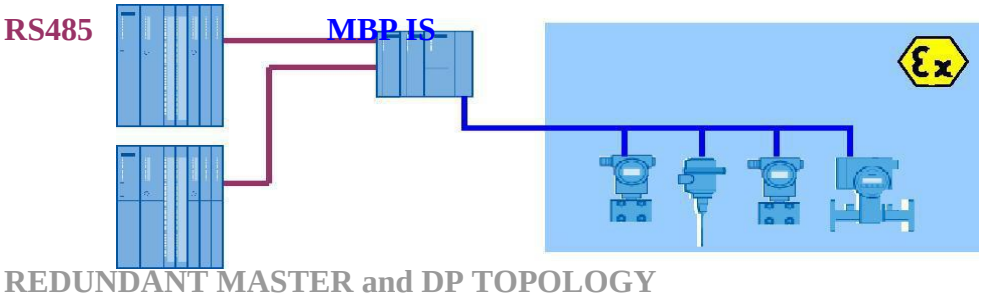


Figure 2.20 SINGLE LINE TOPOLOGY



REDUNDANT MASTER and DP TOPOLOGY

DP/PA Link **MBP IS**

Figure 2.21 Integration of MBP-IS with PROFIBUS

PROFIBUS PA also offers ring topology to provide redundancy mechanism on the MBP bus. The combination of redundant couplers and field devices with active field distributors implements ring redundancy and creates expanded media redundancy. Subsegments which have become effective due to a short circuit or wire break are automatically and seamlessly operated further via a coupler each in a line structure.

RING TOPOLOGY
on PA
DP/PA Link - Redundant

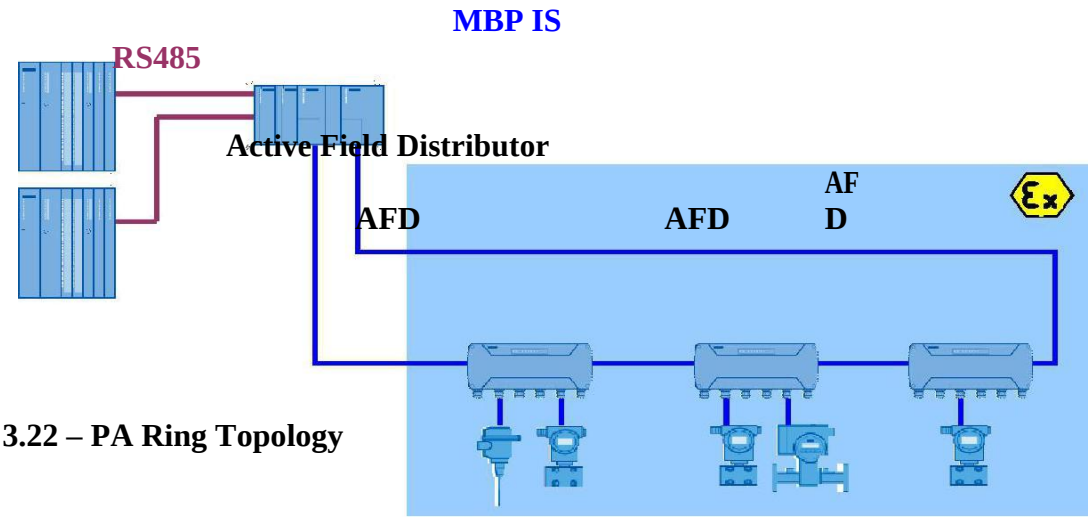


Figure 3.22 – PA Ring Topology

QUESTIONS

UNIT- II Part- A

Sl.no	Part A - Questions	CO	Level
1	List out the various arithmetic and number comparison functions of PLC	CO3	L1
2	State about Skip and MCR functions	CO3	L1
3	Point out the PLC shift register functions	CO3	L4
4	Classify the PLC functions with bits	CO3	L4
5	Show the format of PLC block data transfer function	CO2	L3
6	List the Seven layers of OSI model	CO4	L1
7	Define HART protocol	CO4	L1
8	Show the message format of HART protocol	CO4	L3
9	List out the various HART networks.	CO4	L1
10	Name the various standards of field bus	CO4	L1

UNIT- II Part – B

Sl.no	Part B – Questions	CO	Level
1	Explain about PLC arithmetic and number comparison functions	CO3	L2
2	Discuss the various data handling functions of PLC	CO3	L2
3	Design the ladder diagram of pick and place robot with sequencer function.	CO6	L6
4	Develop a ladder diagram for dishwasher operation	CO6	L6
5	Explain about the different HART networks	CO4	L4
6	Discuss in detail about field bus with OSI model	CO4	L2
7	Give case studies on bottle filling system using PLC	CO3	L5

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS
DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT 3 – DISTRIBUTED CONTROL SYSTEMS – SIC1405

UNIT 3 DISTRIBUTED CONTROL SYSTEMS (DCS)

Evolution of DCS - building blocks - different architectures-comparison of architectures-detailed descriptions and functions of local control units - basic elements & functions-operator stations - data highways - redundancy concepts.

3.1 Distributed control system (DCS)

DCS is a control system for a plant or process, where the control elements are distributed throughout the system. It is in contrast to non-distributed systems, that uses a single controller at a central location. In a DCS, a hierarchy of controllers are connected by communication networks for command and for monitoring. To command and to monitor a hierarchy of controllers is connected by communications networks. Example scenarios where DCS might be used are given below:

- Chemical plants
- Petrochemical plants (oil) and refineries
- Pulp and Paper Mills
- Boiler controls and power plant systems
- Nuclear power plants
- Environmental control systems
- Water management systems
- Metallurgical process plants
- Pharmaceutical manufacturing plants
- Sugar refining plants
- Dry cargo and bulk oil carrier ships
- Formation control of multi-agent systems

The processor receives information from input modules and sends information to output modules. The input modules receive information from input instruments in the process and transmit instructions to the output instruments in the field. Computer buses or electrical buses connect the processor and modules through multiplexer or demultiplexers. Also it connect the distributed controllers with the central controller and finally to the Human Machine Interface (HMI) or control consoles. Early minicomputers were used in the control of industrial processes ever since the beginning of the 1960s. For example IBM 1800, was an early computer that had input/output hardware to gather process signals in a plant for conversion from field contact levels (for digital points) and analog signals to the digital domain.

The first industrial control computer system was built in the year 1959 at the Texaco Port Arthur, Texas, refinery with an RW-300 of the Ramo-Wooldridge Company. DCS was introduced in 1975, where both Honeywell and Japanese electrical engineering firm Yokogawa introduced their own independently produced DCSs at roughly the same time, with the TDC 2000 and CENTUM systems. Also US -based Bristol introduced their UCS 3000 universal controller in 1975. In 1980, Bailey (now part of ABB) introduced the NETWORK 90 system. In 1980, Fischer & Porter Company (now also part of ABB) introduced DCI-4000 (DCI stands for Distributed Control Instrumentation). DCS largely came due to the increased availability of microcomputers and the proliferation of

microprocessors in the world of process control. Computers are applied to process automation for some time in the form of both Direct Digital Control (DDC) and Set Point Control. In the early 1970s Taylor Instrument Company, (now part of ABB) developed the 1010 system, Foxboro the FOX1 system and Bailey Controls the 1055 systems. All of them were DDC applications implemented within minicomputers (DEC PDP-11, Varian Data Machines, MODCOMP etc.) and connected to proprietary Input / Output hardware. Central to the DCS model was the addition of control function blocks. One of the first embodiments of object-oriented software, function blocks were self contained "blocks" of code that emulated analog hardware control components and performed tasks that are essential to process control, such as execution of PID algorithms. Function blocks continue to endure as the predominant method of control for DCS suppliers, and are supported by key technologies such as Foundation Field bus today.

Digital communication between distributed controllers, workstations and other computing elements (peer to peer access) are one of the primary advantages of DCS. Attention was duly focused on the networks, that provides all important lines of communication that is , for process applications, specific functions such as determinism and redundancy. Hence many suppliers embraced the IEEE 802.4 networking standard. This decision set the stage for the wave of migrations essential when information technology moved into process automation and IEEE 802.3 rather than IEEE 802.4 prevailed as the control LAN.

DCS conveyed distributed intelligence to the plant and built up the presence of computers and microprocessors in process control, it didn't manage the cost of the scope and openness important to bind together plant asset necessities. In various cases, the DCS was just a digital replacement of the same functionality provided by analog controllers and a panel board display. In 1980s, users began to look at DCSs as a development of the basic process control.

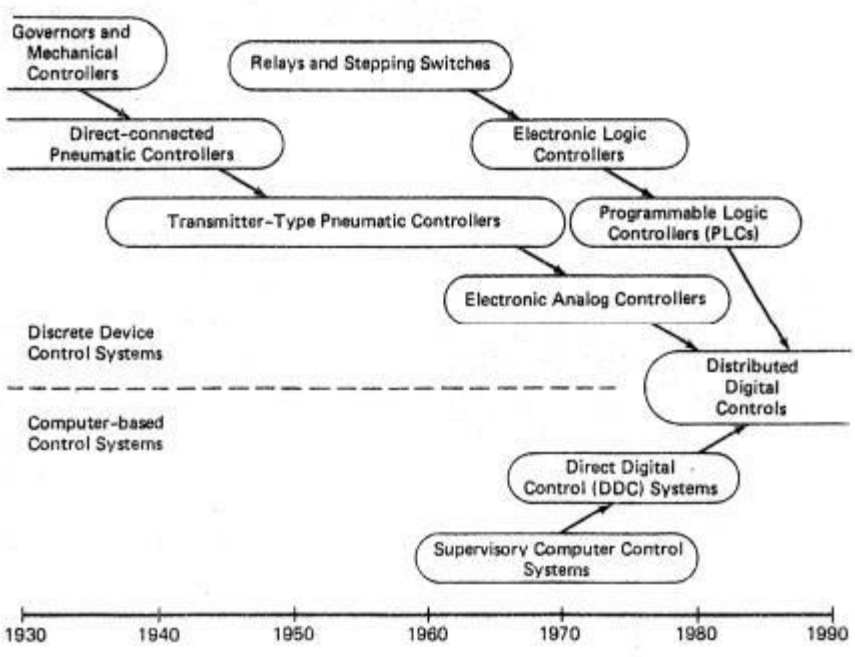


Fig. 3.1 Evolution of DCS

3.2 Traditional Control System Developments

The idea of distributed control systems is not a new one. The early discrete device control systems listed in the figure above were distributed around the plant. Individual control devices such as governors and mechanical controllers were situated at the process equipment to be control local readouts of set points. Control outputs were also available, and a means to change the control mode from manual to automatic are also usually provided. It is up to the operator to coordinate the control of the numerous gadgets that made up the total process. These controllers provides a more flexible way to select and adjust the control algorithms, yet all of the elements of the control loop such as sensor, controller, operator interface, and output actuator were still located in the field. There was no mechanism for communication between controllers.
Later in 1930's a new architecture was developed in which measurements made at the process were converted to pneumatic signals at standard levels, which were then transmitted to the central station. The required control signals were computed at this location, then transmitted back to the actuating devices. The great advantage of this architecture is that all the necessary information will be available to the operator at the central location. Thus, the operator was able to make better control decisions to operate the plant with a greater degree of safety and economic return.

Later in 1950s and early 1960s, the technology started to shift from pneumatics to electronics . This change reduced the installation cost and also eliminated the time lag in pneumatic systems . These advantages became more significant as the size of the plants increased. Another consequence of the centralized control architecture was the development of the split controller structure . In this type of controller, the operator display section of the controller is panel mounted in the control room and the computing section will be located in a separate room.

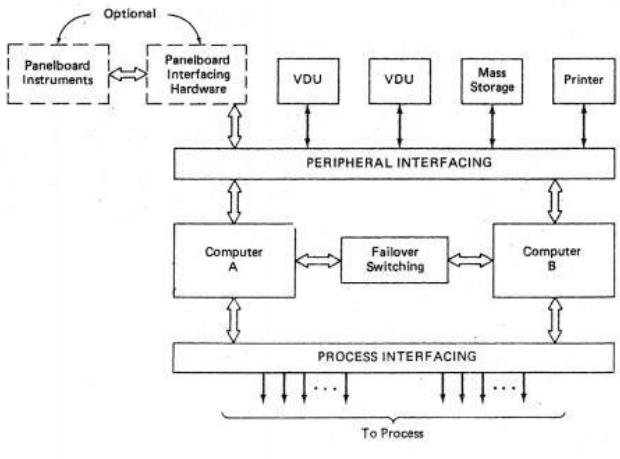
3.3 Resulting System Architectures

As a result of the developments , two industrial control system architectures came into existence by the end of the 1970s . The first architecture is a hybrid one, which makes use of a combination of discrete control hardware and computer hardware in a central location to implement the necessary control functions. In hybrid architecture, the local control of the plant operations is implemented by using discrete analog and sequential logic controllers . Operator interface is usually provided by panel board instruments. The plant management operations are performed by supervisory computer and its associated data acquisition system. The supervisory computer is used to perform operations such as operating point optimization, alarming, data logging, and historical data storage and retrieval. It is also used to operating point optimization, alarming, data logging, and historical data storage and retrieval.

In the second architecture, all the system functions are implemented in high performance computer hardware in a central location. To accomplish this task, redundant computers are required so that the failure of a single computer does not shut the whole process .Operator interface for the plant management functions is provided using

computer-driven VDUs. Operator interface for continuous and sequential closed-loop control are also implemented using VDUs.

The computers can be interfaced to standard panel board instrumentation so that the operator in charge can use a more familiar set of operator interface. The major difference between the two architecture is the location and the implementation of the first- level continuous and sequential logic control functions . By the late 1970s, the hybrid system architecture became a more popular approach in industrial practice . The chemical and petroleum process industries heavily favored this approach.



Central Computer System Architecture.

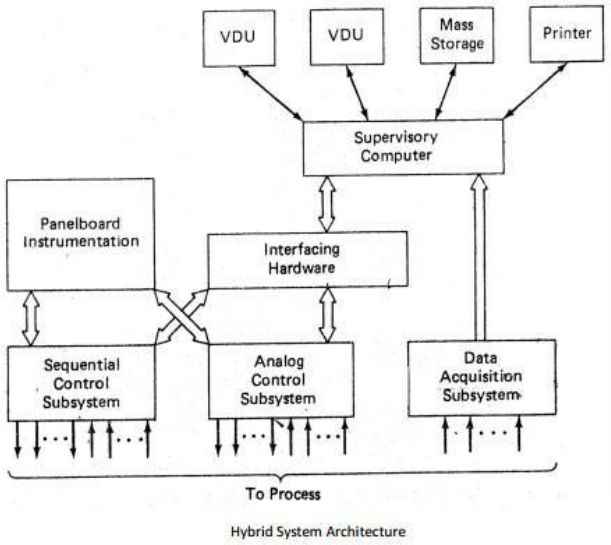


Fig.3.2 Hybrid system Architecture

Fig.3.3 Central computer system Architecture

Though centralized computer and hybrid system architectures provides significant advantages over existing architectures, they suffered from a number of disadvantages . They are- The CPU represents a single point failure that can shutdown the entire process if it is lost. To overcome the above drawback, another computer is used as a ‘‘hot standby’‘ to take over if the primary control computer fails. This approach lead to a new architecture which is more expensive than an analog control system that performs a comparable set of functions. Another problem with these computer-based systems has been that the software required to implement all these functions is extremely complex.

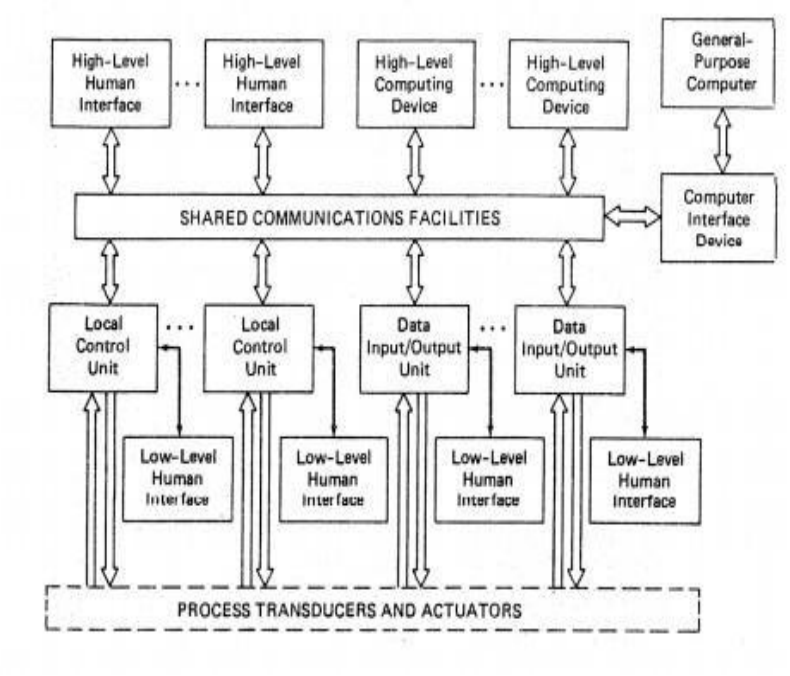


Fig.3.4 Generalized Distributed Control System Architecture

3.4 COMPARISON WITH PREVIOUS ARCHITECTURES

Scalability and expandability—The hybrid system architecture is quite modular than distributed system architecture. The central computer architecture is designed for only a small range of applications which is not cost-effective for applications smaller than its design size and it cannot be expanded once its memory and performance limits are reached.

Control capability—Hybrid architecture has only limited functions available in the hardware modules .To add a function involves adding hardware and rewiring the control system. While in central computer and distributed architectures advantages of using digital control is available

Operator interfacing capability—The operator interface in the hybrid system consists of conventional panel board instrumentation for control and monitoring functions and a separate video display unit. In the central computer and distributed architectures, VDUs

are used as the primary operator interface for supervisory control functions. The VDUs in the distributed system are driven by microprocessor which can be applied in a cost effective way to small systems as well as large ones.

Integration of system functions—A high degree of integration minimizes user problems in procuring, interfacing, starting up, and maintaining the system. The hybrid system is poorly integrated. The central computer architecture is well integrated because the functions are performed only by the same hardware. The distributed system lies somewhere in between, depending on how well the products are designed to work together.

Significance of single point fai1ure—In the central computer architecture, the failure of the supervisory computer will cause the entire plant to shut down unless a backup computer is used. Hence centralized architecture is very sensitive to single-point failures while the hybrid and distributed architectures are insensitive to single-point failures due to the modularity of their structure.

Installation costs—The installation costs of the hybrid system is high since custom wiring is needed for internal system interconnections and long wiring runs are needed from sensors to control cabinets and the large volume of control modules are required to implement the system.

The central computer architecture requires less cost the module inter- connection wiring are eliminated and VDUs are used to replace much of the panel-board instrumentation. The distributed system reduces costs further by using a communication system to replace the sensor wiring runs .

Maintainability—The hybrid system is particularly poor in this area because large number of spare modules are required. The central computer architecture is better than hybrid architecture since the range of module types is reduced . The maintainability of the distributed system architecture is excellent since there are only a few general-purpose control modules in the system. The spare parts and personnel training requirements are also minimal in distributed architecture.

3.5 A comparison of architectures

While the figure describes the basic elements of all microprocessor based local control units, the current offerings of controllers in the market place exhibit endless variations on this structure. The controllers differ in size, I/O capability, range of functions provided, and other architectural parameters depending on the application and the vendor who designed the equipment.

3.6 Architectural Parameters

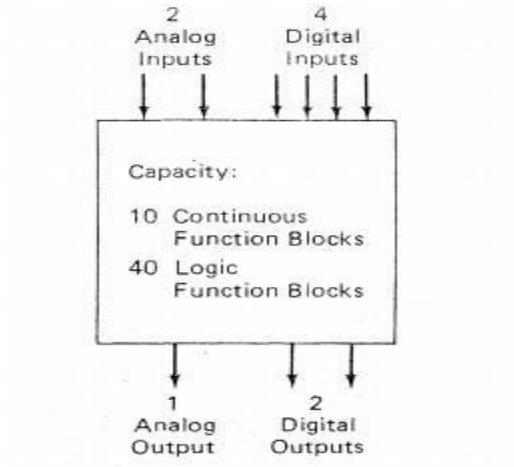
When evaluating the controllers in the market or when specifying a new one, the control system designer is facing with the problem of choosing a controller architecture that best meets the needs of the range of applications in which the controller is to be used. Few of the major architectural parameters that must be selected include the following:

1. Size of controller—This refers to the number of function blocks and/ or language statements that can be executed by the controller, and also the number of process I/O channels provided by the controller.
2. Functionality of controller—This refers to the mix of function blocks or language statements provided by the controller (e.g., continuous control, logic control,

arithmetic functions, or combinations of the above) .Also it refers to the mix of process input and output types (e.g, analog or digital) provided by the controller.

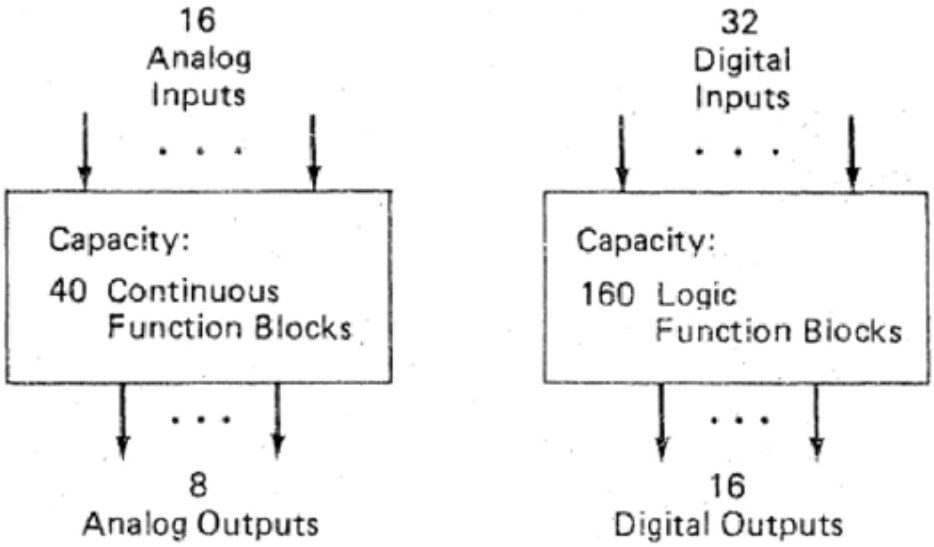
3. Performance of controller—It refers to the rate at which the controller scans inputs, processes function blocks or language statements, generates outputs, also includes the accuracy with which the controller performs these operations.
4. Communication channels out of controller— ln addition to process inputs and output channels, the controller must provide other communication channels to operator interface devices and to other controllers and devices in the system. The number, type and speed of these channels are key controller design parameters.
5. Controller output security—In a real-time process control system, a mechanism must be provided (usually manual backup or redundancy) to ensure that the control output is maintained despite a controller failure so that a process shutdown can be avoided.

Unfortunately, it is not generally possible to select any one of these architectural parameters independently from all of the others, since there is a great degree of interaction among them. Hence, selecting the best combination for the range of applications to be considered is more a matter of engineering judgment than a science. Each vendor of microprocessor based systems has a different view of the range of applications intended for the controller, and as a result designs an LCU architecture that quite often differs from that of its competitors. To illustrate some of the differences in LCU architectures, three representative LCU configurations are shown in Figures. They are not intended to represent particular commercially available products but, different classes of controllers on the market today.

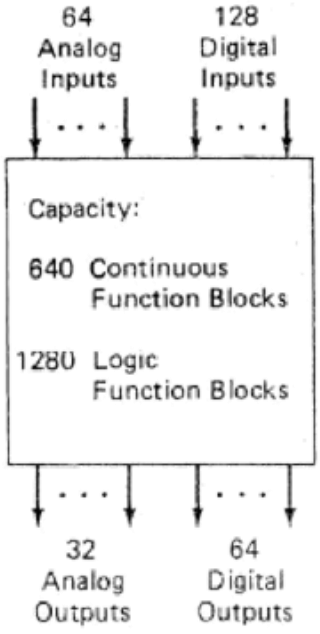


LCU Architecture—Configuration A

Fig.3.5 LCU Architecture – Configuration A



LCU Architecture—Configuration B



LCU Architecture—Configuration C

Fig.3.6 LCU Architecture – Configuration B and C

Configuration A represents a class of single-loop LCU that provides both analog and digital inputs and outputs and executes both continuous and logic function blocks. Configuration B represents an architecture in which two different types of LCUs

implement the full range of required continuous and logic functions. Configuration C gives a multiloop controller architecture in which both continuous and logic functions are performed.

Comparison of LCU Architectures			
ARCHITECTURE PARAMETERS	**CONFIGURATION A (SINGLE- LOOP)**	**CONFIGURATION B (2 LCU TYPES)**	**CONFIGURATION C (MULTI-LOOP)**
Controller size	Number of functions needed for single PID loop or motor controller	Includes functions and I/O needed for eight control loops and a small logic controller	System size is equivalent to small DDC system
Controller Functionality	Uses both continuous and logic function blocks	Continuous and logic function blocks split between controllers	Uses both continuous and logic function blocks, can support high level languages
Controller Scalability	High degree of scalability from small to large systems	Requires both controller types even in small systems	Not scalable to very small systems
Controller Performance	Requirements can be met with inexpensive hardware	Because of functional split, performance requirements are not excessive	Hardware must be high performance to execute large number of functions
Communication Channels	Need inter module communications for control; only minimum needed for human interface	Functional separation requires close interface between controller types	Large communication requirement to human interface; minimal between controllers.
Controller output security	Controller has single loop integrity; usually only manual backup Is needed	Lack of single loop integrity requires redundancy in critical applications.	Size of controller requires redundancy in all applications

LCU Configuration A - In configuration A, the controller size is the minimum that required to perform a single loop of control or a single motor control function or other simple sequencing function. Two digital outputs are provided to allow the controller to drive a pulsed (raise/lower) positioner or actuator. Twice as many inputs as outputs are provided to allow implementation of algorithms such as cascade control, temperature compensation of flows, and interlocking of logic inputs and continuous control loops.

A general purpose controller requires both continuous and sequential (logic) function blocks to be included in its library. In most industrial control applications, the performance of the LCU is adequate to sample all inputs, compute all of the function blocks, and generate all outputs in the range of 0.1 to 0.5 seconds maximum. Since configuration has such a small number of inputs, function blocks, and outputs, the performance requirement can be met easily by a simple and inexpensive set of microprocessor-based hardware (e g an eight-bit microprocessor and matching memory components). The communication requirements on configuration A for purposes of human interfacing are minimal since only one loop is controlled and a few input points monitored. However it is important that a secure inter controller communication channel be provided so that the single-loop controller can participate in complex control system structures that contains other LCUs.

An important architectural feature to be considered when evaluating industrial controllers is the provision for control output security in commercially available controllers, one or both of the following methods are used to permit continued operation of the process in the event of a controller failure.

(1) A backup feature is provided to allow the operator to adjust the control output manually if the automatic controller fails, or

(2) A redundant controller is provided to allow continuation of automatic control if the primary controller fails. The choice of method depends on the number of control outputs that would be lost if a controller fails.

An operator can handle a small number of loops (usually in the range of one to four) manually, that is so for small controllers usually only a manual backup is provided. In case of controller architecture A, the single-loop integrity of the controller configuration allows the simple and inexpensive option of manual backup to be used in most applications. For controllers that implement large numbers of control loops, some form of control redundancy is usually provided.

LCU Configuration B - In the first place, two different types of LCUs (continuous control and logic control are used to provide the full range of required controller functionality). In general increasing the number of types of controllers in the system has both positive and negative effects on the positive side, that is the specialized design of each controller allows it to match the functional needs of the corresponding application more closely than would a single general purpose controller. As a result, a particular control application would require a smaller number of controller hardware modules on the other hand, an increase in the number of controller types from one to two also results in increased interfacing requirements between controllers, additional documentation and training needs, and a decrease in production volume for each controller type (resulting in higher unit costs and longer lead times). In addition, this increase will reduce the scalability of the resulting system, since in the general case both of the controller types

will be needed even in the smallest system. The optimum tradeoff is dependent on the range of applications foreseen for the system being designed.
With respect to controller size, configuration B is medium in scale. The continuous control portion has the form of an eight-loop controller and the logic control portion can be viewed as a small programmable logic controller (PLC) or equivalent. Because of the split in functions and the relatively small size of each controller, the performance requirements on the controller hardware is to meet the 0.1 to 0.5 seconds cycle time are not excessive. This controller is usually implemented using a high performance eight bit or an average performance 16-bit microprocessor and matching memory components.

In the area of communication channels required, configuration B calls for a well designed interface between the two controller types, since in many systems both controllers must operate in close coordination to integrate continuous anti logic control functions. The ability to communicate with other controllers in the distributed system must be provided, but the communication performance level need not be as high as in configuration A since a greater percentage of communications takes place within the LCU. Of course, as with Configuration A, a communication channel to human interface devices has to be provided, but the channel must have a larger bandwidth because of the larger volume of traffic required per controller.
Finally, regarding output security, the lack of single-loop integrity of configuration B implies that a redundant controller must be used in all critical applications (i e , those in which a controller failure would cause a significant upset to the process). Of course if the application requires both continuous and logic control, redundancy must be provided for both of the controller types in the system. It has to be noted that full one-on-one redundancy often is not used in commercially available controllers. Instead, one redundant controller may be used to back up several primary controllers. This can reduce the cost of redundancy to some extent, but increases the complexity of the hardware used for interfacing the primary and backup controllers.

LCU Configuration C is closer to the structure to direct digital control (DDC) systems than the other two configurations. It is designed as multiloop controller in which all functions are performed by one CPU in conjunction with its associated memory and I/O boards. This places stringent requirements on the performance of the hardware since all of the control algorithms in the LCU must be executed within 0.5 seconds of less. This LCU configuration is implemented with one or more 16-bit microprocessors or a 32-bit microprocessor in conjunction with support hardware such as arithmetic coprocessors to attain the required speed.

In this configuration, it also becomes feasible that it include a high-level Language (usually BASIC or FORTRAN) in addition to or instead of function blocks. This architecture has a number of advantages over configurations A and B. For example, requirements for communication among controllers are minimal because of the high density of functions implemented per controller. Also this LCU's high-level language capability allows it to implement complex user-defined control and computational algorithms. However, it is not as scalable to small systems as the other two architectures, and the communication port or ports to the operator interface hardware must handle a large volume of traffic. In addition, since a failure of this type of LCU could affect a large number of control loops, are redundant CPU (and perhaps redundant I/O hardware as

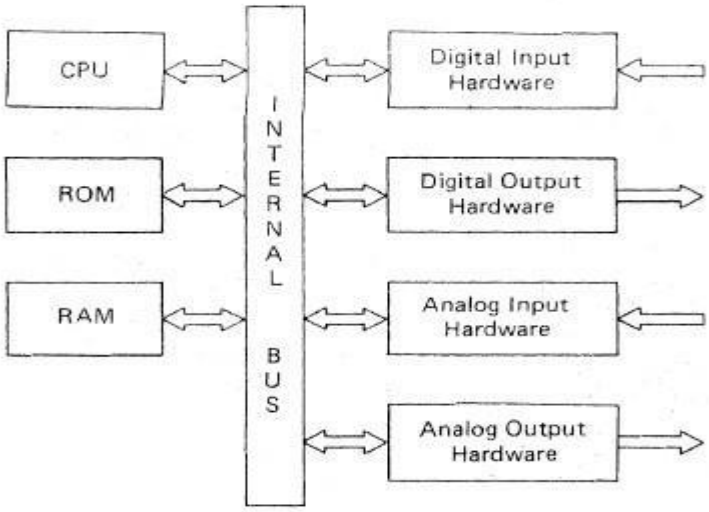
well). Redundant hardware may not be required if only a few loops are affected by a failure or if the controller is performing only high-level control functions and is not manipulating control outputs directly.

3.5 LOCAL CONTROL UNIT (LCU)

* Smallest collection of hardware that performs closed loop control and interfaces directly with the process.
* It takes input from process measuring devices and commands from operator.
* Computes the control outputs needed to make process follow the command.
* Sends control output to actuators, drives valves and other mechanical devices.

Requirements of LCU

* Flexibility of changing the control configuration.
* Ability to use the controller without being a computer expert.
* Ability of the LCU to communicate with other elements in the system.
* Ability to bypass the failed controller manually.



Basic Elements of a Local Control Unit.

Fig.3.7 Basic Elements of a Local Control Unit

The LCU also have input/output circuitry so that it can communicate with the external world by reading in or receiving, analog and digital data as well as sending similar signals out. Generally, the CPU communicates with the other elements in the LC U over an internal shared bus that transmits addressing, data control, and status information in addition to the data .The controller structure shown in figure is the minimum that required to perform basic control functions. In a noncritical application in which the control function never changes, this structure might be adequate. The control algorithms could be coded in assembly language and loaded into ROM. After the controller was turned on, it would read inputs, execute the control algorithms, and generate control outputs in a fixed cycle in definitely. However, because the situation is

not this simple in industrial control applications, the controller structure shown in Figure must be enhanced to include the following:

1. Flexibility of changing the control configuration -In industrial applications the same controller product usually is used to implement a great variety of different control strategies. Even for a particular strategy, the user usually wants the flexibility of changing the control system tuning parameters without changing the controller hardware. Therefore the control configuration cannot be burned into ROM but must be stored in a memory medium whose contents can be changed, such as RAM. Unfortunately, RAM is usually implemented using semiconductor technology that is volatile that is, it loses its contents if the power is turned off (whether due to power failure, routine maintenance or removal of the controller from its cabinet). Therefore, some provision must be made for restoring the control configuration, either from an external source or from a nonvolatile memory within the controller itself.
2. Ability to use the controller without being a computer expert-The typical user of an industrial control system is generally familiar with the process to be controlled, knows the basics of control system design, and has worked with electric analog or pneumatic control systems before. However, the user is usually not capable of or interested in programming a microprocessor in assembly language. He or she simply wants to be able to implement the selected control algorithms. Therefore a mechanism for allowing the user to ‘ configure” the LCU’s control algorithms in a relatively simple way must be provided.
3. Ability to bypass the controller in case it fails so that the process still can be controlled manually- Shutting down the process is very expensive and undesirable for the control system user. Since all control equipment has the potential of failing no matter how carefully it has been designed, the system architecture must allow an operator to “take over” the control loop and run it by hand until the control hardware is repaired or replaced.
4. Ability of the LCU to communicate with other LCUs and other elements in the system-Controllers in an industrial control system do not operate in isolation but must work in conjunction with other controllers, devices, and human interface devices.

3.6 Redundant Controller Designs

The manual backup approach to control system security is viable if the LCU in question handles only a few loops and if the process being controlled is relatively slow. On the other hand situations, however, require some form of controller redundancy to ensure that automatic control of the process is maintained in spite of an LCU failure.

By their nature, redundant control system structures are very complex than those that rely on manual backup. The addition of redundant elements to the basic control system will always result in an increase in system cost and also in additional maintenance to service the extra hardware. The redundant structure must be designed carefully to ensure that system reliability actually increases enough to offset these drawbacks. Some of the guidelines to follow in evaluating or designing a redundant control system are given as follows.

The redundant architecture should be kept as simple as possible as there is a law of diminishing returns in redundancy design. At some point, adding more hardware will reduce system reliability.

1. As much as possible, the architecture must minimize single points of failure. The redundant hardware elements must be in dependent as possible so that the failure of any one does not

bring the rest down as well (in each design, however, there is always at least one single point of failure; this element must be designed to be as simple and reliable as possible.)

2. Also, the redundant nature of the controller configuration should be transparent to the user; that is, the user should be able to deal with the redundant system in the same way as a non redundant one. This includes both operational and engineering functions (e.g., control system configuration and tuning). If one of the redundant elements fails, the steps to follow in repairing and restarting the system is to be clear to the user.
3. The process should not be bumped or disturbed either when one of the redundant elements fails or when the user puts the repaired element back on line.
4. After a control element has failed, the system should not have to rely on it to perform any positive action or to provide any necessary information to other elements in the system until after repair or replacement .
5. The redundant LCU architecture must have the capability for "hot" spare replacement; that is, allow for the replacement of failed redundant elements without shutting down the total LCU. The following discussion will describe, in order to increase complexity, several approaches for designing a redundant LCU architecture:
   - CPU redundancy
   - One-on-one redundancy
   - One-on-many redundancy
   - Multiple active redundancy

In each case, the key advantages and disadvantages of the approach will be listed. No attempt will be made to recommend a best approach; as always, this will depend on the particular control system application.

3.6.1 CPU Redundancy

Only one redundant element will be active at a time. The backup element takes over if the primary fails. In the first configuration, only the CPU portion of the LCU is redundant, but the I/O circuitry will not be redundant. This configuration is more popular in areas where large number of control loops have to be implemented. Only one of the CPUs is active which performs operations such as reading inputs, performing control computations, and generating control outputs at any one time. The user designates the primary CPU, through the priority arbitrator circuit shown in Figure, using a switch setting mechanism. After startup, the arbitrator starts monitoring the operation of the primary CPU and if the arbitrator detects a failure in the primary , it will transfer the priority to the backup. During this operation, the backup CPU will periodically update its internal memory by reading the state of the primary CPU through the arbitrator .Though, both CPUs are connected to the plant communication system, only the primary will be active in transmitting and receiving messages over this link. The operator and engineering interface used in this system is the high-level human interface which is usually a CRT-

based video display unit that interfaces with the LCU over the shared communication facilities. Only the primary CPU will accept control commands and tuning changes transmitted by the VDU. The primary CPU, in turn, updates the backup CPU with this updated information. By following the same methodology, all monitoring and status information in the LCU is transmitted to the VDU by the primary CPU.

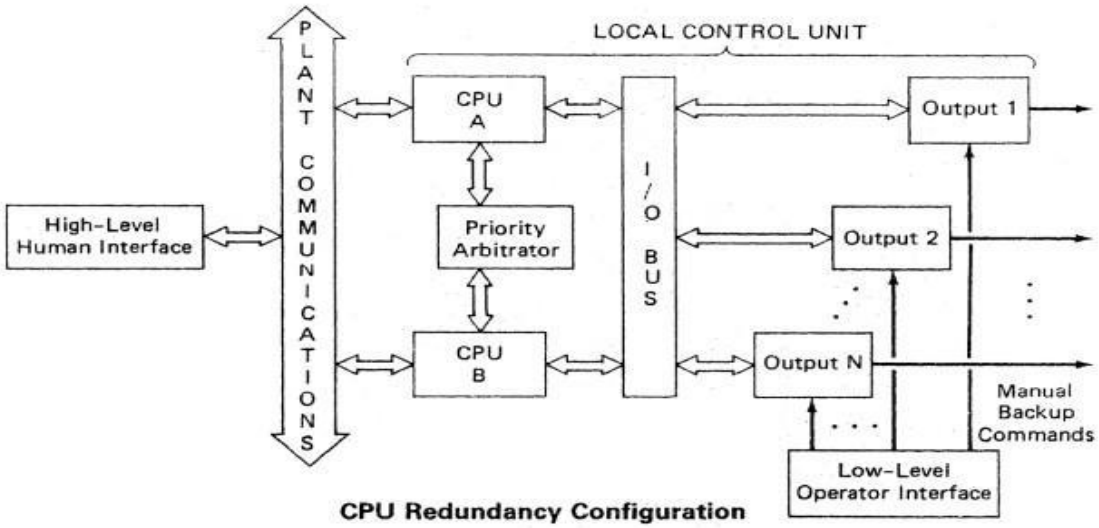


Fig.3.8 CPU Redundancy Configuration

The key advantages of this approach- relatively easy to implement and cost-effective. The redundant elements can interface easily to the plant communication facilities and the 110 bus, which are shared communication channels. The cost of the redundant hardware will not be excessive since only the CPU hardware is duplicated. Problems may occur with this architecture if it is not designed properly. For example, the I/O bus and the priority arbitrator will represent potential single points of failure in the configuration. They must be designed in such a way that their failure does not affect both CPUs . Another potential problem is that the low-level operator interface is physically located near the LCU.

3.6.2 One-on-One Redundancy

The remaining three redundancy approaches provides redundancy in the control output circuit as well as in the CPU hardware. Hence, most of these architectures do not support a low-level operator interface for manual backup purposes. This approach shown in Figure provides a total backup LCU to the primary LCU . An output switching block must be included to transfer the outputs when the controller fails, since the control output circuit is duplicated . Similar to first redundant configuration a priority arbitrator designates the primary and backup LCUs and activates the backup if a failure in the primary is detected. It also serves as the means to update the internal states of the backup

LCU . In this configuration, the arbitrator has the additional responsibility of sending a command to the output switching circuitry if the primary LCU fails, causing the backup LCU to generate the control outputs Communications with the high-level human interface which are handled in the same way as in the CPU-redundant configuration.

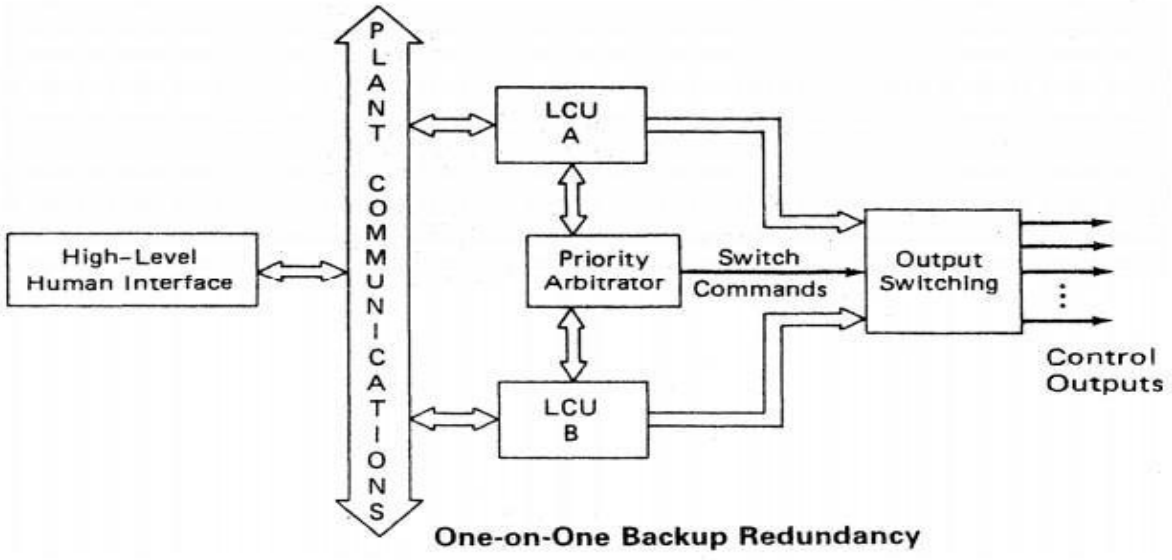


Fig.3.9 One –on – One Backup Redundancy

The main advantage of the one-on-one configuration are no manual backup is needed. It eliminates any questions that may arise with a partial redundancy approach , particularly regarding the decision on which elements are to be redundant. However, it also suffers from a number of disadvantages . First, it is an expensive approach to redundancy, but additional equipments must be included to manage the redundant ones. It also suffers from potential single-point failure problems with the arbitrator and the output switching circuitry.

3.6.3 One-on -Many Redundancy

In this configuration, a single LCU is used as a standby to back up any one of several primary LCUs. An arbitrator is required to monitor the status of the primary controller and switch to the backup when a failure occurs. Unfortunately, there is no way to know beforehand, for which primary controller the backup has to replace. Hence, a very general switching matrix is necessary to transfer the I/O from the failed controller to the backup. It is also not possible to preload the backup controller with the control system configuration for any particular primary LCU. Rather, the configuration is loaded into the backup LCU from the primary LCU only after the primary has failed. The cost of this approach is lower than the other three redundancy configurations because only a small portion of the control hardware is duplicated. The switching matrix is an element whose operation is essential to control the loops concerned, but it also represents a potential single point failure. Because of this complexity, it must be designed very carefully so that its failure does not affect the other loops. Second, the approach relies on the failed controller to provide a copy of the control system configuration to the backup LCU. A better approach would be provided if we store a copy of each primary LCU’s control

configuration in the arbitrator. When an LCU failure occurs, the arbitrator could then load the proper configuration into the backup LCU.

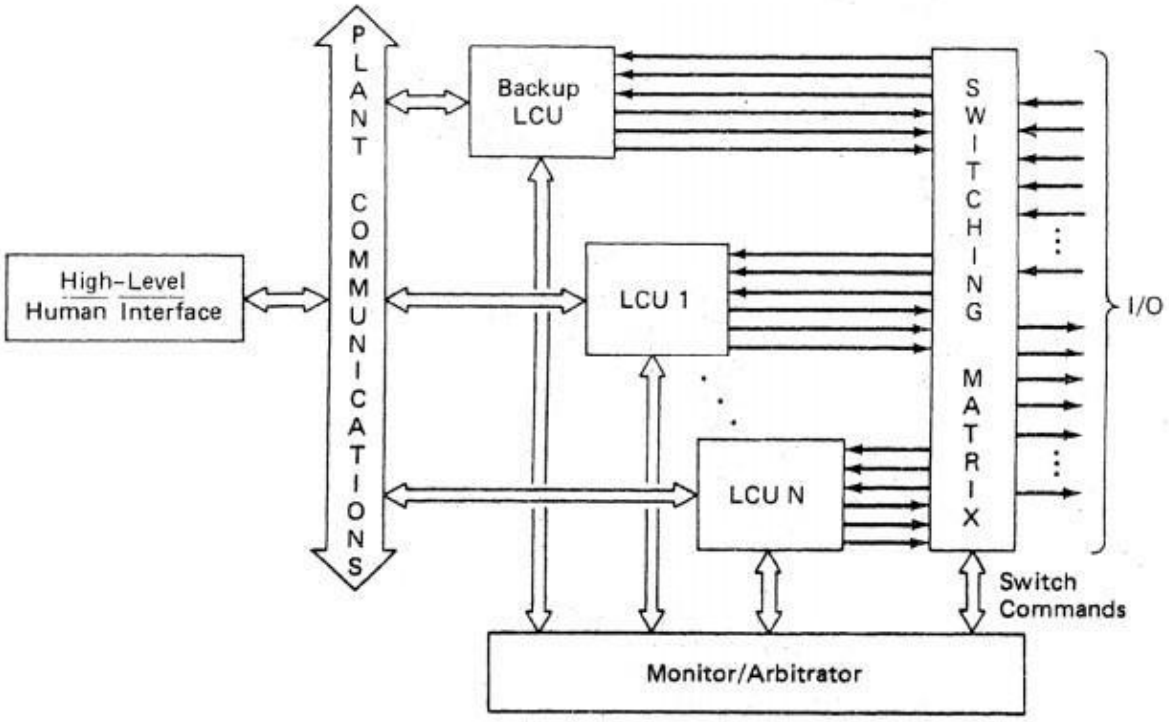


Fig.3.10 Multiple Active Redundancy

In this approach, three or more redundant LCUs are used to perform the same control functions. In this one of the redundant controllers will be active at the same time in reading process inputs, computing the control calculations, and generating control outputs to the process. Each LCU has access to all of the process inputs needed to implement the control configuration. An output voting device selects one of the valid control outputs from the controllers and transmits it to the control process. When a controller fails, it is designed to generate an output outside the normal range. The output voting device will then discard this output as an invalid one. In the case of analog control outputs, the output voting device, is often designed to select the median signal; in the case of digital control outputs, the voting device is designed to select the signal generated by at least two out of the three controllers. Each controller has access to the output of the voting device to check its own operation and shut down if its output disagrees significantly with that of the other controllers.

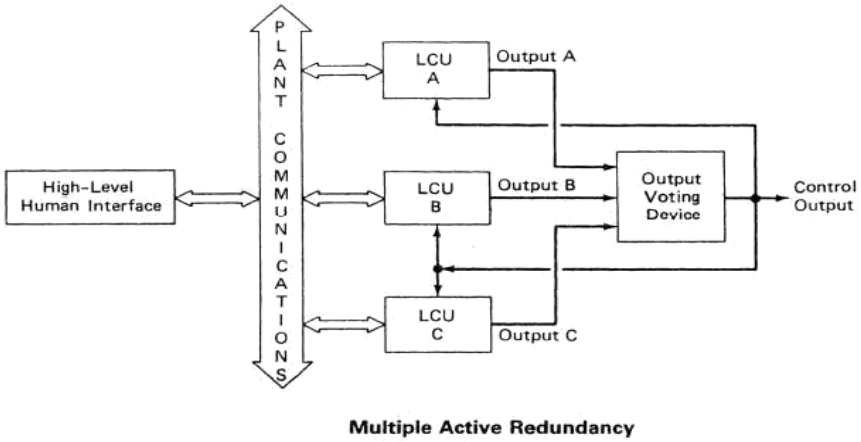


Fig.3.11 Multiple Active Redundancy

The multiple active redundancy configuration is a very sophisticated approach which is very complex and obviously costs three times more than a non redundant system. In its modified form it has been used in safety systems for nuclear power plants. More often, this configuration has been used in high-reliability computer control applications such as aerospace industry . This approach may find its way into selected process control applications as hardware costs continue to decline and configurations become standardized. The main advantage of this approach is that, as long as the output voting device is designed for high reliability, it significantly increases the reliability of the control system. However, this architecture suffers various disadvantages in addition to cost. In most of the applications where this configuration has been used so far, the approach has implemented a fixed configuration. The process of ensuring that control system configuration and tuning changes have been implemented properly is not trivial. Other drawbacks of this approach are that the added hardware requires increased maintenance and also the system is very complex.

UNIT- III Part- A

Sl.no	Part A - Questions	CO	Level
1	Define DCS	CO2	L1
2	Write few applications of DCS	CO2	L6
3	Point out the various elements in DCS	CO2	L4
4	Discuss the significance of LLHI	CO2	L2
5	List the various architecture of DCS	CO2	L1
6	Classify the various redundant controllers in DCS	CO2	L1
7	Define redundant controller	CO2	L1
8	Tell the functions of LCU	CO2	L1
9	Compare LLHI and HLHI	CO2	L5
10	Report the architecture parameter to be considered while selecting LCU	CO2	L3

UNIT- III Part – B

Sl.no	Part B - Questions	CO	Level
1	Explain the generalized architecture of DCS with neat sketch	CO2	L2
2	Discuss and compare the different architecture of DCS	CO2	L2
3	Sketch and discuss about local control unit in DCS	CO2	L3
4	Explain and compare the various redundancy controllers	CO2	L2
5	Compare Hybrid and central computer system architecture with construction and working	CO2	L2
6	List different configurations of LCU, explain the various blocks in LCU	CO2	L1, L2

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS
DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

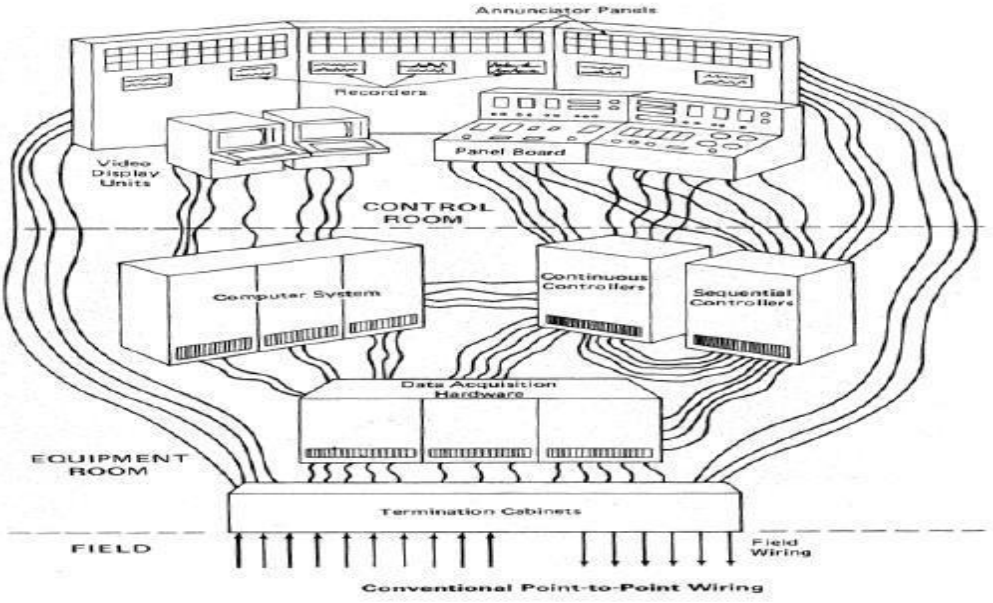
UNIT 4 – DISTRIBUTED CONTROL SYSTEMS : COMMUNICATION FACILITIES AND APPLICATION – SIC1405

UNIT 4 DISTRIBUTED CONTROL SYSTEMS: COMMUNICATION FACILITIES AND APPLICATIONS

DCS communication Facilities - communication system requirements - architectural issues - protocol issues - communication system standards - operator interfaces - low level and high level operator interfaces - Operator Displays - Engineering interfaces - low level and high level engineering interfaces - Applications of DCS - oil and gas (extraction, production and storage)

4.1 DCS Communication Facilities

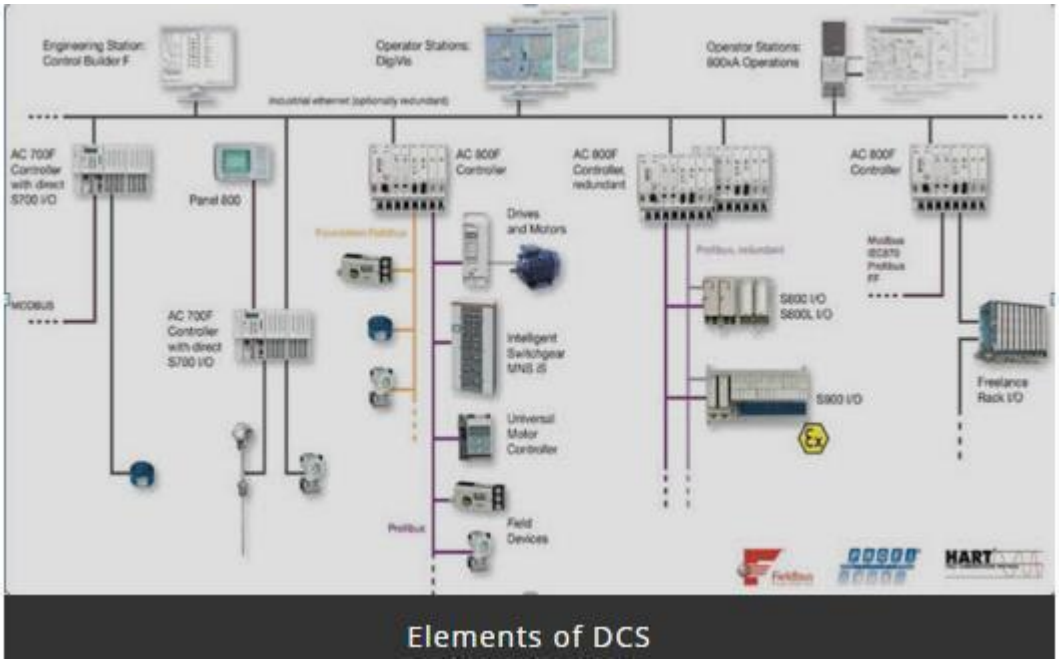
In conventional non distributed control systems, the connections that allow communication between the various system elements are configured on a per-job basis. This system consists of a combination of continuous controllers, sequential controllers, data acquisition hardware, panel board instrumentation, and a computer system. The controllers communicate with each other by means of point-to-point wiring, usually within the control cabinets. This custom wiring reflects the particular control system configuration selected. The controllers are connected to the corresponding panel board instrumentation and to the computer system by means of pre fabricated cables. The computer obtains information from the data acquisition modules using similar hard wiring or cabling that is specific to the particular module configuration implemented. This approach to interconnecting system elements has proven to be expensive to design and check out, difficult to change, burdensome to document, and subject to errors. It becomes even more cumbersome if the system elements are distributed geographically around the plant. The first step taken to improve this situation was to introduce the concept of distributed multiplexing in the early 1970s. This concept was first used in the process control industry to implement large-scale data acquisition systems, which at that time had grown to several thousand inputs in size. To reduce the cost of wiring, remote multiplexers located near the sensors in the field were used to convert the inputs to digital form and transmit them back to the data acquisition computer over a shared communication system.



4.1 Fig. Conventional Point to Point Wiring

When distributed control systems were introduced in the late 1970s, the use of digital communications was extended to control-oriented systems as well. The communication system began to be viewed as a facility that the various elements and devices in the distributed network share, as the "black box" representation in Figure shows. Replacing dedicated point-to-point wiring and cabling with this communications facility provides a considerable number of benefits to the user: 1. The cost of plant wiring is reduced significantly (see References, since thousands of wires are replaced by the few cables or buses used to implement the shared communication system. 2. The flexibility of making changes increases, since it is the software or firmware configurations in the system elements that define the data interconnection paths and not hard wiring. 3. It takes less time to implement a large system, since the wiring labor is nearly eliminated, configuration errors are reduced, and less time is required to check out the interconnections 4 The control s tem is more reliable due to the significant reduction In physical connections in the system (a major source of failures) However, replacing hard wiring with the shared communications network of a distributed control system also raises a number of questions for the user In a conventional system, communications between system elements travel at the speed of light, essentially with zero delay. Also, since the hard-wired communication channels between elements are dedicated, there is no danger of overloading a channel. In the case of a shared communication system, the user must be able to judge whether the response time and capacity of the shared system (in addition to many other performance factors) are adequate for the application. This can be a difficult problem for the user, since there are significant differences and few standards among the various communication systems available on the market today

4.2 DCS ARCHITECTURE ISSUES



Elements of DCS

Fig.4.2 Elements of DCS

Distributed Control System continuously interacts with the processes in process control applications ones it gets instruction from the operator. It also facilitates to variable set points and

opening and closing of valves for manual control by the operator. Its human machine interface (HMI), face plates and trend display gives the effective monitoring of industrial processes.

4.2.1 Engineering PC or controller

This controller is the supervisory controller over all the distributed processing controllers. Control algorithms and configuration of various devices are executed in this controller. Network communication between processing and engineering PC can be implemented by simplex or redundant configurations.

4.2.2 Distributed controller or Local control unit

It can be placed near to field devices (sensors and actuators) or certain location where these field devices are connected via communication link. It receives the instructions from the engineering station like set point and other parameters and directly controls field devices.

It can sense and control both analog and digital inputs / outputs by analog and digital I/O modules. These modules are extendable according to the number of inputs and outputs. It collects the information from discrete field devices and sends this information to operating and engineering stations.

In above figure AC 700F and AC 800Fcontrollers acts as communication interface between field devices and engineering station. Most of the cases these act as local control for field instruments.

4.2.3 Operating station or HMI

It is used to monitor entire plant parameters graphically and to log the data in plant database systems. Trend display of various process parameters provides the effective display and easy monitoring.

These operating stations are of different types such as some operating stations (PC's) used to monitor only parameters, some for only trend display, some for data logging and alarming requirements. These can also be configured to have control capabilities.

4.2.4 Communication media and protocol

Communication media consists of transmission cables to transmit the data such as coaxial cables, copper wires, fiber optic cables and sometimes it might be wireless. Communication protocols selected depends on the number of devices to be connected to this network.

For example, RS232 supports only for 2 devices and Profibus for 126 devices or nodes. Some of these protocols include Ethernet, DeviceNet, foundation filed bus, modbus, CAN, etc.

In DCS, two or more communication protocols are used in between two or more areas such as between field control devices and distributed controllers and other one between distributed controllers and supervisory control stations such as operating and engineering stations.

Important features of DCS

• To handle complex processes:

In factory automation structure, PLC-Programming Logic Controller is used to control and monitor the process parameters at high speed requirements. However due to limitation of number of I/O devices, PLC's cannot handle complex structure.

Hence DCS is preferred for complex control applications with more number of I/O's with dedicated controllers. These are used in manufacturing processes where designing of multiple products are in multiple procedures such as batch process control.

4.2.5 System redundancy:

DCS facilitates system availability when needed by redundant feature at every level. Resuming of the steady state operation after any outages, whether planned or unplanned is somewhat better compared to other automation control devices. Redundancy raises the system reliability by maintaining system operation continuously even in some abnormalities while system is in operation.

4.2.6 Lot of Predefined function blocks:

DCS offers many algorithms, more standard application libraries, pre-tested and pre-defined functions to deal with large complex systems. This makes programming to control various applications being easy and consuming less time to program and control.

Powerful programming languages:

It provides more number of programming languages like ladder, function block, sequential, etc for creating the custom programming based on user interest.

4.2.7 More sophisticated HMI:

Similar to the SCADA system, DCS can also monitor and control through HMI's (Human Machine Interface) which provides sufficient data to the operator to charge over various processes and it acts as heart of the system. But this type of industrial control system covers large geographical areas whereas DCS covers confined area.

DCS completely takes the entire process plant to control room as a PC window. Trending, logging and graphical representation of the HMI's give effective user interface. Powerful alarming system of DCS helps operators to respond more quickly to the plant conditions

4.2.8 Scalable platform:

Structure of DCS can be scalable based on the number I/O's from small to large server system by adding more number of clients and servers in communication system and also by adding more I/O modules in distributed controllers.

4.2.9 System security:

Access to control various processes leads to plant safety. DCS design offers perfect secured system to handle system functions for better factory automation control. Security is also provided at different levels such as engineer level, entrepreneur level, operator level, etc.

4.3 OPERATOR INTERFACES

The control and communication equipment performs the bulk of the automated functions that are required for operating an industrial process For this automated equipment to be used in a safe and effective manner, however, it is absolutely necessary to have a well-engineered human interface system to permit error-free interactions between the humans and the automated system Two distinct groups of plant personnel interact with the control system on a regular basis Instrumentation and control system engineers—These people are responsible for setting up the control system initially and adjusting and maintaining it from time to time afterwards Plant operators—These people are responsible for monitoring, supervising, and running the process through the control system during startup, operation, and shutdown conditions As the generalized distributed control system architecture, a human interface capability can be provided at one or both of two levels Through a low-level human interface (LLHI) connected directly to the local control unit or data input/output unit (DI/OU) via dedicated cabling, Through a high-level human interface (HLHI) connected to an LCU .or DI/OU only through the shared communications facility. The low-level human interface equipment used in distributed control systems usually resembles the panelboard instrumentation (stations, indicators, and recorders) and tuning devices used in conventional electric analog control systems. The HLHI equipment makes maximum use of the latest display technology (e.g., CRTs or flat-panel displays) and peripheral devices (e.g., printers and magnetic storage) that are available on the market; it is configured in a console arrangement that allows operator and engineer to be seated during use. When it is included in the system configuration, the LLHI generally is located geographically close to (within 100—200 feet of) the LCU or DI/OU to which it is connected On the other hand, the HLHI can be located anywhere in the plant, including the central control room. The needs of the application will determine whether the particular installation has one or both levels of interface Figures shows some examples of typical installations and their corresponding equipment configurations. Figure 5.1 illustrates a relatively small and simple installation A single LCU located in the plant

equipment room (sometimes called the relay room) performs all of the required control functions Low-level human interface units located in the equipment room and the plant control room provide the complete operator and instrument engineer interface for the control system This type of equipment configuration is typical of d stand- alone control system for a small process or of a small digital control system installed in a plant controlled primarily with conventional electrical analog or pneumatic equipment

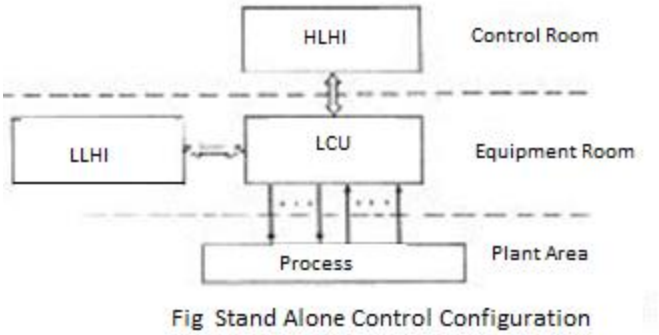


Fig.4.3 Stand Alone Control Configuration

Figure shows a typical structure of a complete plantwide control system Several LCUs are used to implement the functions required in controlling the process, therefore, the control is functionally distributed However, the LCUs are all located in a central equipment room area, and so it is not a geographically distributed control system Both high-level and low-level human interface devices are located in the control room area for operational purposes Most of the operator control functions are performed using the high-level interface, the low-level interface is included in the configuration primarily to serve as a backup in case the high-level interface fails A high evel human interface is located in the instrument engineer's area so that control system monitoring and analysis can be done without disturbing plant operations This type of installation is typical of early distributed control system configurations in which equipment location and operator interface design followed conventional practices

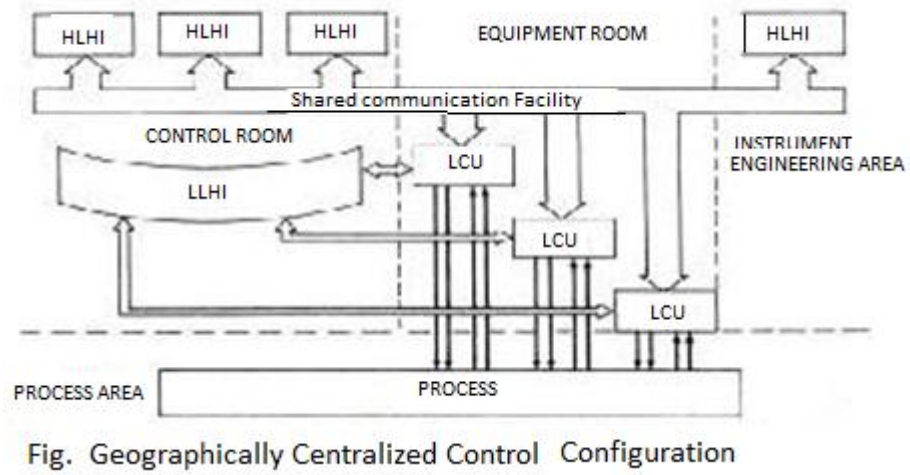


Fig.4.4 Geographically Centralized Control Configuration

Figure shows a fully distributed control system configuration In this case, each LCU is located in the plant area closest to the portion of the process that it controls Associated low-level human interface equipment (if provided) is also located in this area The control room and instrument engineering areas contain high-level human interface units, which are used to perform all of the primary operational and engineering functions The low-level units are used only as manual backup controls in case the high-level equipment fails or needs maintenance This configuration takes advantage of two areas of equipment savings that result from a totally distributed system

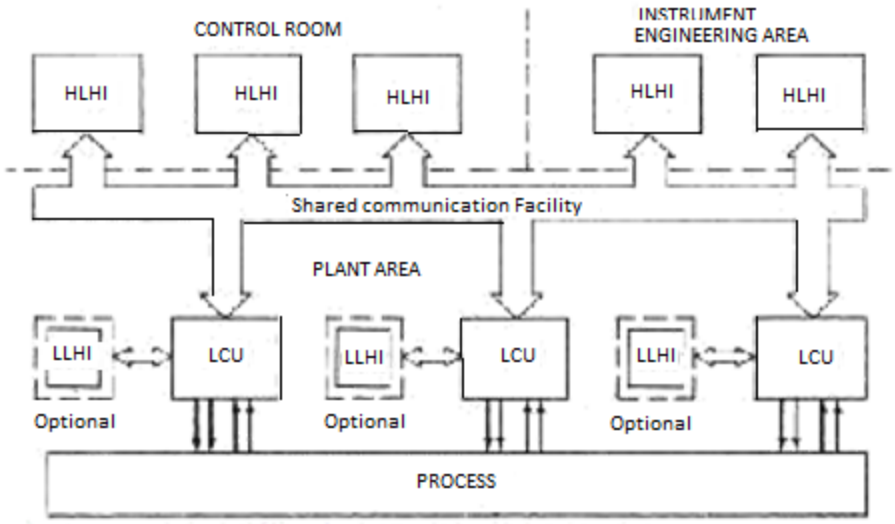


Fig. Geographically Distributed Control Configuration

Fig.4.5. Geographically Distributed Control Configuration

architecture, reduction in control room size (by eliminating panelboard equipment), and reduction in field wiring costs (by placing LCUs near the process).

4.4 OPERATOR INTERFACE REQUIREMENTS

Despite the continuing trend toward increased automation in process control and less reliance on the operator, the basic responsibilities of the operator have remained largely the same in the last fifty years. Most of the changes have come in the relative emphasis on the various operator functions and the means provided to accomplish them. As a result, the operator interface in a distributed control system must allow the operator to perform tasks in the following traditional areas of responsibility process monitoring, process control, process diagnostics, and process record keeping. In addition, it is important to design the operator interface system using human factors design principles (also called ergonomics) to ensure that the operator can perform these tasks in an effective manner with minimum risk of confusion or error The following paragraphs provide a discussion of the key functional requirements in each of these areas

4.4.1 Process Monitoring

A basic function of the operator interface system is to allow operator (whether one or more) to observe and monitor the current state of the process This function includes the following specific requirements.

- The current values of all process variables of interest in the system must be available for the operator to view at any time This includes both continuous process variables (e g , flows, temperatures, and pressures) and logical process variables (e g , pump on/off status and switch positions) The operator must have rapid access to any variable, and the values displayed must be accurate and current If the information provided is not valid for some reason (e g , a sensor has failed or has been taken out of service for maintenance), this condition should be readily visible to the operator.
- Each process variable, rather than being identified by a hardware address only, must be identifiable by a "tag" or name assigned by the instrument engineer, a descriptor that expands on and describes the tagged variable must be associated with the tag The tag and descriptor give the variable a meaning relative to the process, an example might be to label a certain temperature with a tag of TT075B and a corresponding descriptor "COLUMN TEMPERATURE 75 IN AREA B."
- The value of the process variable must be in engineering units that are meaningful to the operator, and those units must be displayed along with the variable values. In the temperature example just given, the engineering units might be in degrees Fahrenheit or Celsius.
- In many cases, the operator is interested in variables that are functions of or combinations of the basic process variables being measured (e.g. an average of several temperatures, a maximum of several flows, or a computed enthalpy). The operator must have these computed variables available at all times in the same formats as the basic variables (i e , tags, descriptors, and engineering units).

Another monitoring function of the operator interface is to detect abnormalities in the state of the process and to report them to the operator. In its simplest form, this is the familiar function of alarming. Some of the specific requirements of this function are the following :

- The control and computing hardware in the distributed system identifies the alarm statuses of individual variables in the process. The operator interface system must report these statuses to the operator in a clear manner Types of alarms for each variable—such as high, low, and deviation (from a nominal value)—must be differentiated clearly True alarms must be differentiated from indications of process equipment status that do not denote an abnormal condition requiring operator action.
- The operator interface must also report similar alarm statuses for computed variables 3 The operator interface must either display the alarm limits along with the process variable or make them easily accessible to the operator.
- When the system has detected an alarm condition, the interface must alert the operator to this condition in unambiguous terms and require the operator to acknowledge the existence of the alarm.
- If the system detects multiple alarm conditions within a short time period, the operator interface must inform the operator that multiple alarms have occurred, preferably with some indication of the priority of the various alarm conditions.

- In some processes, "abnormal operation" can be detected only by looking at a combination of several process variables and noting if this combination is within an allowable region of operation In this case, the operator interface system must provide an appropriate mechanism to allow the operator to view this multivariable alarm status condition and interpret it properly.

When monitoring the process, an operator is interested in not only the current value of a process variable but also its trend in time. This gives the operator an idea of the direction in which the process is moving and whether or not there is trouble ahead. For this reason, the operator interface system must provide the operator with fast access to the recent history of selected process variables in the plant; these variables are called trended variables. Some specific requirements in trending are that:

- It must be possible to group the trended variables by related process function as well as by similarities in time scale of interest. For example, it might make sense to group all temperatures that are associated with a particular portion of the process.
- The trend graph must clearly label the engineering units, time increments, and absolute time of day of the trended variables.
- The operator must be able to obtain a precise reading (in engineering units) of both the current value as well as past values of the trended variable.
- If at all possible, the same graph displaying the trend should also show auxiliary information that would help the operator evaluate the status of the trended variable. This information might include the nominal value of the variable, the set point of the associated control loop, the allowed range of the variable, or the allowed rate of change.

4.4.2 Process Control

The process monitoring capabilities just described provide the necessary information for the operator's primary function— process control. The following specific operator interface requirements come under the category of process control

- The operator interface must allow the operator to have rapid access to all of the continuous control loops and logic sequences in the process control system.
- For each continuous control loop, the interface must allow the operator to perform all of the normal control functions changing control modes (e g , automatic, manual, or cascade), changing control outputs in manual mode, changing set points in automatic mode, and monitoring the results of these actions.
- The interface must allow the operator to perform such logic control operations as starting and stopping pumps or opening and closing valves. If interlocking logic is included in these operations, the interface
  - must allow the operator to observe the status of the most recently requested command, the current logic state of the process, and the status of any permissives (interlocking signals) that may be preventing execution of the requested command.
- In the case of a batch control sequence, the operator interface must allow the operator to observe the current status of the sequence and to interact with it to initiate new steps or halt the sequence, as required.
- In both the continuous and sequential control cases, the interface system must allow the operator to have access to and be able to manipulate the control outputs despite any singlepoint failure in the equipment between the operator interface and the control outputs.

4.4.3 Process Diagnostics

Monitoring and controlling the process under normal operating conditions are relatively simple functions compared to operation under abnormal or hazardous conditions caused by failures in plant equipment or in the instrumentation and control system. The operator interface system must provide enough information during these unusual conditions to allow the operator to identify the equipment causing the problem, take measures to correct it, and move the process back to its normal operating state. The first step in this sequence is to determine whether it is the instrumentation and control equipment that is causing the problem To this end, the distributed control system should provide the following diagnostic features and make the results of the diagnostic tests available to the operator.

- Ongoing tests and reasonableness checks on the sensors and analyzers that measure the process variables of interest.
- Ongoing self-tests on the components and modules within the distributed control system itself controllers,

  communication elements, computing devices, and the human interface equipment itself.

Historically, diagnosing problems within the process itself has been a manual function left to the operator. Operator interface systems have been designed to display all of the available process information (both relevant and irrelevant), and the operator has had to sort it all out and come up with the right diagnosis. This was not a bad approach when the operator had to contend with small processes, those characterized by only a few hundred process variables. More recently, however, processes have grown to such a size that describing them takes 5,000—10,000 process variables (many of which may be strongly interacting). It has become extremely difficult for an operator to identify the source and nature of a fault in an item of process equipment in this environment. A conventional alarming system, for example, may indicate the most immediate failure symptom but provide few clues as to the original source of the alarm condition. As a result, diagnostic functions that automatically detect process faults are now often required in distributed control systems. These functions may include:

- First-out alarming functions, which tell the operator which alarm in a sequence occurred first,
- Priority alarming functions, which rank the current alarms by their importance to process operation, allowing the operator to safely ignore the less important ones, at least temporarily,
- More advanced diagnostic functions that use a combination of alarming information and data on process variables to identify the item of failed process equipment and (in some cases) the mode of failure.

Many of these advanced alarming and diagnostic functions are application- oriented ones that the designer must configure for the specific process of interest, however, the distributed control system must support the implementation of these functions.

4.4.4 Process Record Keeping

One of the more tedious duties that operating people in a process plant must perform has been to walk the board, that is, to take a pencil and clipboard and periodically note and record the current values of all process variables in the plant Depending on the pro- cess, the frequency for doing this has ranged from once an hour to once every several hours This logged information, along with the trend recordings obtained automatically, serves as a useful record of plant

operating status during each shift The record-keeping burden has increased significantly in recent years due (in part, at least) to governmental reporting requirements related to pollution monitoring, product liability, and worker safety regulations. Record-keeping was one of the first functions to be automated using conventional computer systems In state-of-the-art distributed control systems, this function often can be implemented in
the operator interface system without the use of a separate computer Specific record-keeping requirements include the following

- Recording of short-term trending information—To record in real time mode.
- Manual input of process data—The operator must be able to enter manually collected process information into the system for record- keeping purposes. This information includes both numeric data and operator notes or journal entries.
- Recording of alarms—These are logged on a printer, a data storage device, or both, as they occur. Often, the return-to-normal status and operator acknowledgments must also be logged. The information recorded includes the tag name of the process variable, the time of alarm, and the type of alarm (high, low, or deviation). There must also be a mechanism that allows convenient review of the alarm in- formation.
- Periodic records of process variable information—The values of selected variables are logged on a printer, data storage device, or both, on a periodic basis every few minutes or every hour, depending on the dynamics of the variable The operator or instrument engineer may decide to store an averaged value Over the sampling period in- stead of the instantaneous value.
- Long-term storage and retrieval of information—The alarms and periodic logs as described above are accessible for short periods of time, commonly, for a single eight-hour shift or a single day. In addition, the same information, or a smoothed or filtered version of it, must be stored on a long-term basis (months or years). The system must include a mechanism for easy retrieval or ‘‘instant replay’‘ of such information
- Recording of operator control actions—Some process plants require the actions of the operator affecting control of the process to be recorded automatically These include changes in control mode, set point, manual output, or logic command Clearly this recording function must be implemented in such a way that the operator cannot deactivate it.

4.5 Guidelines for Human Factors Design

in the past, equipment used for operator interfacing has often been designed more for the convenience of the equipment vendor or architect-engineer than for ease of use by the operator In recent years, it has become clear that a small investment made in the proper design of human interfacing equipment pays handsome dividends fever operator errors (which can cause plant downtime or damage to equipment), less operator fatigue (which can cause a loss in productivity), and more efficient use of operating personnel. Some general design guidelines for designing operator interface systems for industrial control include the following:

- Consider the full range of expected operator population (e.g. , male and female, large and small, right-handers and left-handers).
- Take into account common minor disabilities in operators (e.g., color blindness and nearsightedness).
- Design the system for operators, not for computer programmers or engineers .

- Allow rapid access to all necessary controls and displays
- Arrange equipment and displays to make sense from an operational point of view, cluster with respect to process unit, functional operation, or both
- Make consistent use of colors, symbols, labels, and positions to minimize operator confusion
- Do not flood the operator with a lot of parallel information that is not structured in any way, the information should be prioritized, organized in a meaningful manner, and reported only when it changes significantly
- Ensure that the operator's short-term memory is not overtaxed when performing a complex sequence of operations provide aids such as operator guides, menus, prompts, or interactive sequences for assistance in these operations These aids are particularly important in stressful situations, during which short-term memory is not a reliable source of operating information
- As much as possible, design the system to detect and filter out erroneous operator inputs, when an error occurs, the system must tell the operator what the input error was and what to do next
- Make sure the control room environment (e g , light, sound levels, and layout) is consistent with the selection and design of the control room equipment In some respects, these guidelines may only seem to state obvious, common-sense design principles, however, a glance at existing operator interface designs shows that these principles are very often violated, either for design expediency or through ignorance of these ergonomic issues.

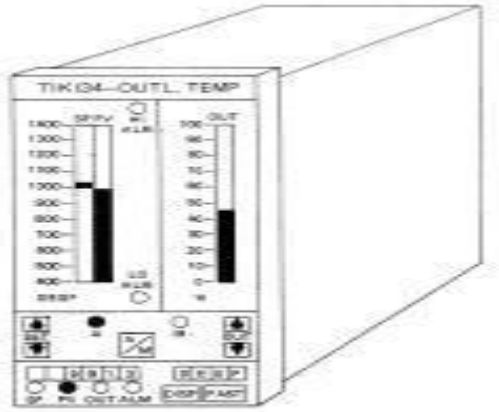
4.6 LOW LEVEL OPERATOR INTERFACES

The low level operator interface (LLOI) in a distributed control system is connected directly to the LCU and is dedicated to controlling and monitoring that LCU. This contrasts with the high Ievel operator interface (HLOI), which can be associated with multiple LCUs and is connected to them through the shared communication facility. LLOIs are used in a variety of applications, in some cases in conjunction with high-level operator interfaces (HLOIs) and in others in place of them. In some applications, all operator functions are performed through the HLOI and no low-level interface is required except during emergency or failure conditions. There are a number of motivations for using an LLOI: It provides an interface that is familiar to operators trained to use panelboard instrumentation, since it is usually designed to resemble that type of instrumentation. It is usually less expensive than an HLOI in small applications (say, less than 50 control loops). It can provide manual backup in case the automatic control equipment or the HLOI fails. LLOI instrumentati6usually includes the following devices: control stations, indicator stations, alarm annunciators, and trend recorder. In some distributed control systems, the vendor offers exactly the same type of instrumentation as used in his conventional analog and logic control systems. More often, however, the vendor supplies smart (microprocessor- based) instrumentation, which offers the user functionality beyond that available in conventional panelboard instrumentation.

4.6.1 Continuous Control Station

One type of panelboard instrumentation used in process control systems is the manual/automatic station associated with a continuous control loop. The stations discussed here are split stations; that is, they are separate from the LCU. Figure illustrates a typical version of a smart continuous control station in a distributed control system. As in the case of most conventional control stations, this one has bar graph indicators that display the process variable ("PV"), associated set

point (“SP”), and the control output as a percent of scale (“OUT”). In addition, however, the smart station includes a shared digital display to provide a precise reading of each of these variables in engineering units. The units used are indicated in an accompanying digital display of are printed on a removable label (in this example, “DEGF” or “%“). The shared digital display also can be used to indicate the high and low alarm limits (“HI ALM” and “LO ALM”) on the process variable when selected by the operator.



Pushbuttons allow the operator to change the mode of control (e.g. , manual, automatic, or cascade) and to ramp the set point (“SET”) or control output (“OUT”), depending on the mode Usually, both fast and slow ramping speeds are provided for the convenience of the operator. The indications the control station often provides include any alarms associated with the process variable being controlled and an indication of the operational status (whether “healthy” or not) of the associated. To minimize requirements for spare parts, one basic control station should be used for all types of associated control loops: standard PID, cascade, ratio, or bias. The station can be customized through the configuration of options in the electronics (using jumpers or switches) and on the front plate indicators and switches (using different overlays or faceplates as appropriate). To be effective as a manual backup station in addition to its role as a single-loop operator interface, the control station must be connected directly to the control output section of the LCU or to the associated field termination panel. In this arrangement, a ‘‘hard” control output (e.g. , a 4—20 ma signal) generated by the station can pass as a backup control signal in case the LCU fails or is under maintenance. The direct connection a so allows the process variable. Input to come into the station for

indication to the operator during manual control. To keep the control output from going through a step change in value when the backup mode is initiated or concluded (automatic bumpless transfer), the station and the LCU must be aware of each other's nominal control output signal These signal values and other useful information (such as alarm and diagnostic status signals) are often sent over a direct serial communication link between the LCU and the station.

4.6.2 Manual Loader Station

Some applications use the HLOI as the primary L6Mi:l station and don't require a full-blown continuous control station for each loop. However, a device is still needed to hold the 4—20 ma control output signal. if the LCU fails or is taken off-line for maintenance or other reasons In this situation, a simple manual loader station is a low-cost alternative to the continuous control station for the purposes of backup The manual loader station is plugged in at the same point as the continuous control station but only allows the operator to run the loop in manual mode Sometimes the process variable is displayed, more often it is not Any balancing of the control output to accomplish bumpless transfer to or from backup is accomplished manually Both the continuous control station (if used for backup) and the manual loader station should be powered from a different supply than that used for the LCU, to ensure continuous backup in case of an LCU power failure.

4.6.3 Indicator Station

If the operator must be able to monitor process variables not associated with control loops, a panelboard indicator station can be provided as a part of the LLOI family of products. The indicator station is similar to the control station in that it provides both bar graph and digital numeric readouts of the process variables in engineering units. Of course, an indicator station requires no control push buttons However, it often provides alarming and LCU diagnostic indications, as does the continuous control station.

Logic Station

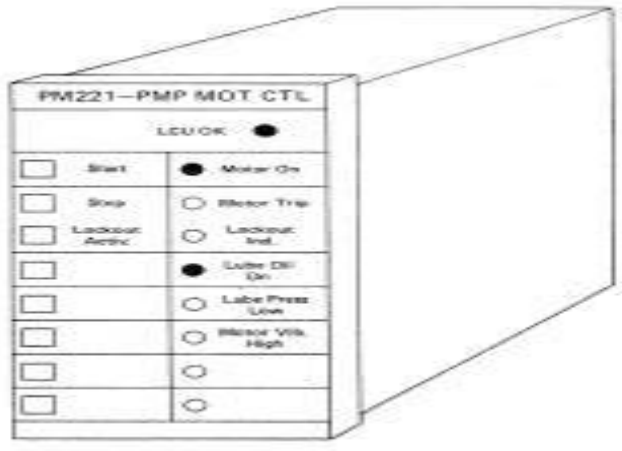


Figure illustrates a control station for a logic control or sequential control system. It consists simply of a set of pushbuttons and indicating lights that are assigned different meanings (through labels) depending on the logic functions being implemented. This type of station is used to turn

pumps on and off, start automatic sequences, or provide permissives or other operator inputs to the logic system. In some systems, the logic control station performs a manual backup function similar to that performed by the continuous control station in case of a failure of an LCU. More often, however, the logic station acts simply as a low-cost operator interface; if the LCU fails, the logic outputs revert to their default or safe states. Smart Annunciators Alarm annunciators in distributed control systems are often microprocessor-based, providing a level of functionality beyond the capability of conventional hard-wired annunciator systems. These smart annunciators can provide such functions as: Alarm prioritization—The annunciator differentiates between status annunciation and true alarms (and how critical the alarms are); Annunciation and acknowledgment mode options—The operator receives a variety of audible and visible alarm annunciation signals (e g , horns, buzzers, flashing lights, and voice messages) A range of alarm acknowledgment and silencing modes also can be provided. First-out annunciation—The annunciator displays the first alarm that appears within a selected group. Alarm "cutout"—The annunäiator suppresses an alarm condition if other specified status conditions are fulfilled. The last function is valuable in minimizing meaningless ‘‘nuisance’‘ alarms. For example, if a pump fails and triggers an alarm, the pump-failed status signal can be used to lock out other related alarms such as ‘‘low flow’‘ or ‘‘pump speed low," since they are not meaningful given the failed operating status of the pump of course, the four alarm logic functions previously listed also can be accomplished within the LCUs in the distributed system itself; however, in some applications it may be convenient to incorporate the functions externally in the annunciators.

4.6.4 Chart Recorders

Although conventional round chart or strip chart recorders are often used to record process variables in a distributed control system, digital recorders which use microprocessors are becoming more cost-effective and popular The digital recorder gathers trend data in its memory and displays the data to the operator using a liquid crystal panel or other flat display device For hard-copy output, the recorder uses an impact- or heat-type of printing mechanism instead of pen-and-ink to record the information. In some models, the recorder draws the chart scales as it is recording, so that plain paper instead of chart paper can be used. The recorder often provides such functions as automatically labeling time and range of variable directly on the chart, using alphabetic or numeric characters. Also, each process variable can he recorded using a different symbol or color to allow the operator to distinguish between the variables easily. Because of the flexibility of the printing mechanism and the memory capabilities of the recorder, intermittent printing of the process variables can supplement the display output without losing any of the stored information.

4.7 HIGH LEVEL OPERATOR INTERFACES

The high-level operator interface in a distributed control system is a shared interface that is not dedicated to any particular LCU Rather, the HLOI is used to monitor and control the operation of the process through any or all of the LCUs in the distributed system information passes between the HLOI and the LCUs by means of the shared communications facility. While the LLOI hardware resembles conventional panelboard instrumentation, HLOI hardware uses CRT or similar advanced display technology in console configurations often called video display units (VDUs). The HLOI accepts operator inputs through keyboards instead of the switches, push-buttons, and potentiometers characteristic of conventional operator interface panels Other digital hardware prints, stores, and manipulates information required in the operator interface system. In

general, the use of microprocessor-based digital technology in the design of the HLOI system allows the development of a hardware configuration that departs radically from the design of panelboardbased operator interfaces This configuration provides the user with several significant advantages over previous approaches: Control room space is reduced significantly; one or a few VDUs can replace panelboards several feet to 200 feet in length, saving floor space and equipment expense. One can design the operator interface for a specific process plant in a much more flexible manner The hardware operations of panelboard design and implementation (e g , selecting control stations and cutting holes in panels) are eliminated. Instead, CRT display formats define the key operator interface mechanism. One can change these formats during startup and duplicate selected information displays wherever necessary. Using microprocessors permits cost-effective implementation of functions that previously could be accomplished only with expensive computers. These include color graphic displays that mimic the organization of the process, information presented in engineering units, and advanced computing and data storage and retrieval functions.

However, one must take great care in designing the HLOI to achieve these benefits while minimizing any negative effects or concerns on the part of the operating personnel. It became evident during the early introduction of this technology to the marketplace that operators accept properly designed HLOIs very quickly. This is especially true of younger operators who have been preconditioned and pretrained by video games and personal computers.

4.7.1 Architectural Alternatives

All high-level operator interface units in distributed control systems are composed of similar elements operator display, keyboard or other input device, main processor and memory, disk memory storage, interface to the shar4d communication facility; and hard-copy devices and other peripherals. However, the architectures of the various HLOIs on the market vary significantly depending on the way in which these common elements are structured.

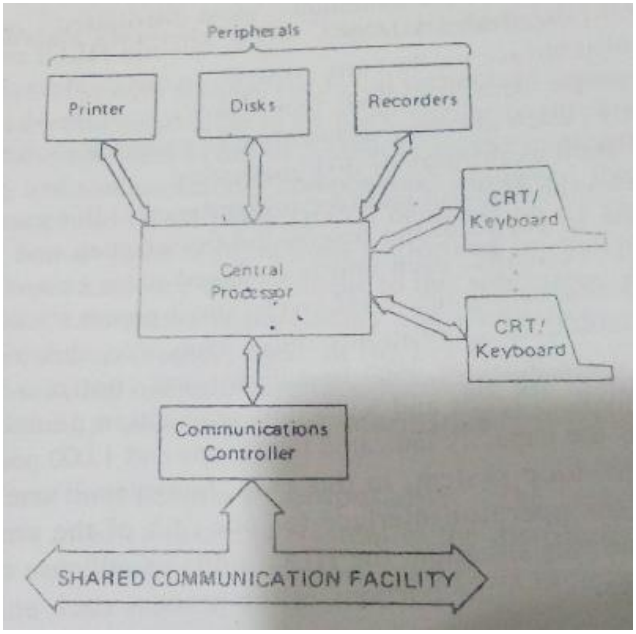


Fig.4.7 Centralized HLOI configuration

Figure shows architecture used in computer-based control systems and early distributed systems. In this architecture, there is a single central processing unit (and associated random-access memory) that performs all of the calculations, database management and transfer operations, and CRT-and-keyboard interfacing functions for the entire HLOI system. A separate communications controller interfaces the central processor with the shared communications facility. There are several advantages to this configuration. First, there is a single database of plant information that is updated from the communication system As a result, each of the CRTs has access to any of the control loops or data points in the system This is desirable, since it means that the CRTs are all redundant and can be used to back each other up in case of a failure. Other advantage is that the peripherals can be shared and need not be dedicated to any particular CRT/keyboard combination. This can reduce the number of peripherals required in some situations. The disadvantages of this configuration are similar to those of all centralized computer system. It is an "all eggs in one basket" configuration, and so is vulnerable to single-point failures. In some cases, redundant elements can be provided; but this approach can lead to complex peripheral-switching and memory-sharing implementations. Any single-processor, single-memory configuration has limitations on the number of loops and data points it can handle before its throughput or memory capacity runs out. In many operator interface systems of this type, the display response times are long and the size of system that can be handled is severely limited. The centralized architecture is not easily scalable for cost-effectiveness: if it is designed properly to handle large systems, it may be too expensive for small ones.

Because of these limitations, most distributed control systems use a decentralized HLOI design. That is, several HLOI units provide the operator interface for the entire system. In this context, an HLOI unit refers to a single element or node that makes use of the shared communication facility. Each unit may include one or more CRTs, keyboards, or penpherals such as printers or disk memories. When the operator interface is distributed in this manner, the issue arises of how to partition the responsibilities of each unit to cover the entire process Usually, each unit is designed to be cost-effective when monitoring and controlling a relatively small process (say, 400 control loops and 1 ,000 data acquisition points) However, this means that several of these units must be used to monitor and control a larger process (say, 2,000 control loops and 5,000 data acquisition points) For example, five units of the capacity indicated (400 loops and I ,000 points) could just cover the 2,000—loop system In this case, however, if one of the control units failed, the operator interface for one—fifth of the process would be lost To avoid this situation, the HLOI units usually are configured for significant overlap in the portions of the process each unit covers.

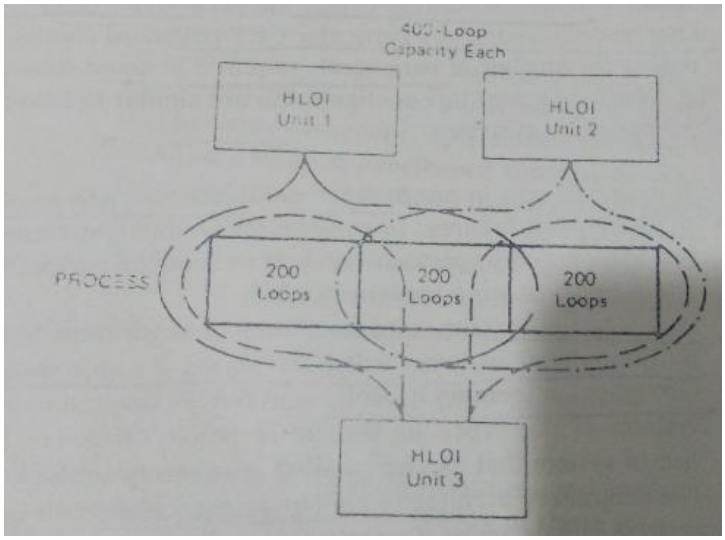


Fig.4.8 Overlap in HLOI Scope of control

Figure illustrates a two—to—one overlap configuration, in which three HLOI units control and monitor a 600—loop process With this approach, the loss of any single HLOI unit does not affect the capability of the operator interface system to control the process Overlap obviously is not an issue if each HLOI is designed to be large enough to accommodate all of the points in an installation (say, 5,000 to 10,000 points) This design approach results in a more expensive version of an HLOI than one designed to handle a smaller number of points. How ever, in this case each HLOI unit is capable of backing up any other unit in the system. The first versions of distributed HLOI systems introduced to the marketplace had a relatively fixed configuration of elements, such as that shown below.

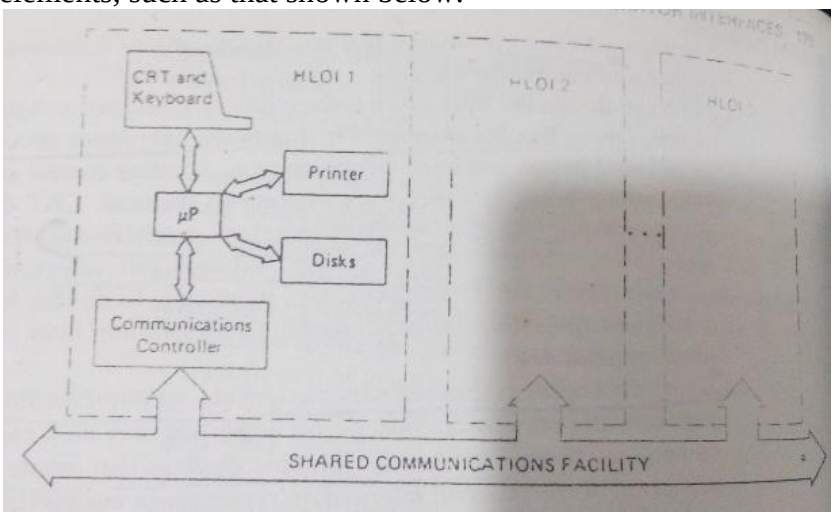


Fig.4.9 Fixed HLOI Configuration

That is, a single HLOI unit consisted of a communications controller, main processor, CRT and keyboard, and associated mass storage The only option for the user was whether to include a printer or other hard-copy device Because of this fixed configuration of elements, the scope of control and data acquisition of the HLOI unit also was fixed. Later versions of HLOI units have

been designed to be modular the user can buy the base configuration at minimum cost or expand it to handle a larger number of control loops and data points. Figure shows one example of a modular HLOI configuration.

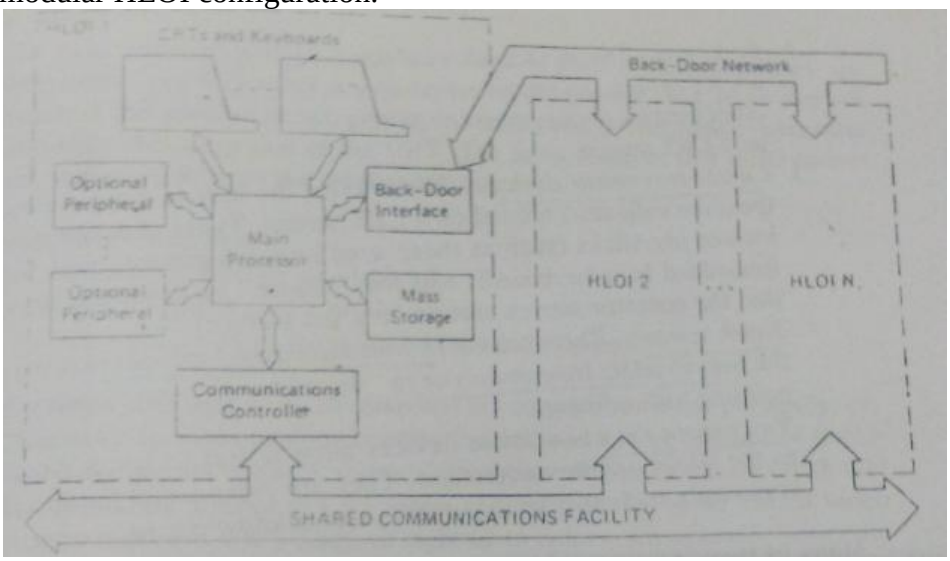


Fig. 4.10 Modular HLOI configuration

The base set of hardware in this case is a communications controller, main processor, single CRT and keyboard, and mass storage unit. However, the system is designed to accommodate optional hardware such as: One or more additional CRTs to allow monitoring of a portion of the process (for example) while the primary CRT is being used for control purposes. Additional keyboards for configuration or backup purposes. Hard-copy devices such as printers or CRT screen copiers. Additional mass storage devices for long—term data storage and retrieval Interfaces to trend recorders, voice alarm systems, or other external hardware Interface ports to any special communication systems such as back- door networks to other HLOI units or diagnostic equipment Backups to critical HLOI elements such as the main processor, communications controller, or shared memory The modular approach to HLOI unit design significantly improves configuration flexibility The user can select the base configuration for small applications and add the optional hardware as the user sees fit Of course, the performance of the main processor and other hardware must be ad- equate to provide the display update and response capability required, even with the maximum size hardware configuration. 5.3.1 Hardware Elements in the Operator Interface Since the HLOI system is based on digital technology, many of the hard- ware elements that go into the system are similar to those used in other

portions of the distributed control system (e g , the microprocessors and the memory and communications components). However, the performance requirements of the HLOI place special demands on its elements; also, the on-line human interface functions performed by the HLOI require display and input hardware that is unique to this subsystem. Microprocessor and Memory Components.

The high-level operator interface is the most complex subsystem in the distributed control hierarchy As a result, the microprocessors used in its implementation must be faster and more powerful than microprocessors used elsewhere in the distributed system. The microprocessor hardware selected for use in the HLOI in commercially available systems tends to have the following characteristics : It uses one or more standard 16- or 32-bit microprocessors available from multiple vendors; these are not single-sourced special components Its processors are members of a family of standard processors and related support chips It is designed to operate in a multiprocessor configuration, using a standard bus for communication between processors and an efficient real-time operating s stem as an environment for application software. The last characteristic is important, since to accomplish the desired computing and data control functions at the speeds required by the real-time operating environment, most HLOI configurations must use multiple processors. The HLOI requires the same types of semiconductor memory devices as used in the LCUs and the shared communications facility RAM for temporary storage of changing data (e g , current values of process variables), ROM for storage of predetermined, unchanging information (e g standard display formats and computational algorithms), and nonvolatile, alterable memory for storage of data that changes only infrequently (e g custom graphic display formats).

The main differences between the memory requirements for the HLOI and those for the LCU are that the HLOI requires shorter access times and greater amounts of memory to meet its high performance and large data storage requirements.

4.7.2 Operator Input and Output Devices

The primary function of the HLOI subsystem is to allow communications between the operator and the automatic portions of the distributed control system. Therefore, the particular data input and output devices selected are vital to the usability of the system and its acceptance by operating personnel. A great variety of operator input devices have been considered for use in industrial control systems: Keyboard—There are two kinds of keyboard in use: the conventional electric typewriter type and the flat-panel type; Light pen—This device allows a person to ‘‘draw’‘ electronically on a CRT screen. In industrial control systems, the light pen is more often used by the operator to select among various options displayed on a CRT screen Cursor-movement devices—These allow the user to move a cursor (position indicator) around on a CRT screen. Types of devices used include joy sticks (such as those used in video games), track balls (imbedded in a keyboard), and the “mouse” (a hand-held device that the operator moves around on a flat surface) Touch screens—There are CRTs with associated hardware that allows the user to select from menus or to “draw” on the screen by directly pointing with a finger. Voice-input devices—These devices allow a user to speak directly to the HLOI and be understood. They are most useful for specific commands such as selecting displays or acknowledging alarms. Many of these devices were originally developed for use in other applications, such as military or aerospace systems or in terminals for computers or computer-aided design systems In evaluating these devices, the designer of an industrial control system must remember that the needs,

background and training of an operator in a process control environment differ substantially from those of an astronaut, military aviator, or engineer. For example, the use of hand-held devices such as light pens or mice in industrial applications has been criticized on the grounds that operators prefer not to use any devices that require a separate operation— removal from or return to storage—for their use. As a result, the most popular input device technologies in industrial control systems are keyboards and touch screens. CRTs—This still is the dominant technology in the area of operator displays , CRTs come in either color or monochrome versions Flat-panel displays—These include gas plasma, vacuum fluorescent, liquid-crystal, and electroluminescent displays, most are mono chrome only. Voice-output devices—Usually these are used to generate alarm messages that the operator will hear under selected plant conditions. The messages can be prerecorded on tape or computer generated by voice synthesizer.

In industrial distributed control systems, the CRT has been and continues to be the predominant visual display device used in HLOI systems. Some flat-pane1 display devices have been used in aerospace and military applications; however, in general they have not yet reached the point of development at which their performance— cost ratio seriously threatens that of the CRT The main features that differentiate the various CRTs on the market include: 1. Size of screen; 2 Number of colors available, 3 Resolution of pixels on the screen, 4 Character oriented versus bit-mapped displays

4.7.3 Peripherals.

In addition to the processing, memory, input, and display devices required to perform the basic operator interface functions, the HLOI configuration must include the following additional peripheral devices to implement the full range of functions: Fixed disk drives—This type of mass memory uses non removable memory media to store large amounts of information that must be accessed rapidly (such as standard and graphical display formats) or to provide a location for temporary storage of historical data. One example of such a memory device is the Winchester disk, a magnetic memory disk with a capacity in the 5—100 Mbyte range Another is the read/write optical disk, which is used for even higher density storage (500—1 ,000 Mbytes) Removable mass memory—This type of mass memory, typified by the floppy disk, uses removable memory disks for storage of information that is to be downloaded to other elements in the distributed system (such as control configurations, tuning parameters, and custom programs) It also can be used for long-term storage of small quantities of historical data, magnetic tape can be used for large quantities Read-only optical disks, such as those in commercial video applications, are useful for storage of large amounts of unchanging information, such as text used in operator guides. Printers and plotters—.At least one printer is required to implement the logging and alarm recording functions The printer can also pro- vide a hard copy of the CRT displays , or a separate printer can be dedicated to that function A black-and-white dot matrix printer is the most prevalent type used, since it can implement either function Pen plotters, ink-jet printers, or thermal transfer printers can provide full-color hard copies of CRT displays . Since these devices tend to be slow, it as important that the HLOI be designed so that the keyboard and CRT are active and available to the operator while the printing or plotting process is going on .

4.8 ENGINEERING INTERFACES

The functions that the plant’s instrument and control system engineer performs are usually quite different and separate from those that the operator performs. This separation arises due to

standard plant practices or even union contracts, which spell out the responsibilities of each group of workers. Because of this, many vendors of distributed control systems provide an engineering interface that is totally independent of the operator interface. Other vendors recognize that there are many common features and functions that both the operator and engineering interfaces require. As a result, these vendors combine the functions in a single set of hardware, but allow the hardware to take on different ‘‘personalities” depending on its current user. As in the case of the operator interface, vendors normally provide the user a choice between two levels of engineering interface hardware : Low-level, minimum-function devices that are inexpensive and justifiable for small systems High-level, full-function devices that are more powerful (and expensive), but which are needed and justifiable for medium and large sized distributed control systems.

4.9 ENGINEERING INTERFACE REQUIREMENTS

In the past, defining and setting up a monitoring and control system consisting of conventional discrete analog and sequential modules has involved a tremendous amount of engineering labor The instrument engineers had to select and procure the control and data acquisition modules, mount them in cabinets, and do a significant amount of custom wiring between modules. Then the engineers had to test and check out the entire system manually prior to field installation. Similarly, they had to select operator interface instrumentation, mount it in panel boards, wire it up, and test it. They had to prepare documentation for the entire configuration of control and operator interface hardware and the corresponding control logic diagrams, usually from manually generated drawings When errors in the control system were found, new modules had to be obtained, wiring redone, and documentation laboriously updated. A small number of general-purpose, microprocessor-based modules have replaced dozens of dedicated function hardware modules. The use of shared communication links has reduced or eliminated custom inter module wiring. Control system logic has been expressed in function block logic diagrams instead of hardware schematics CRT-based operator consoles have replaced huge panel boards filled with arrays of stations, indicators, annunciators, and recorders As a result of these simplifications in the architecture of industrial control systems, the process of engineering such a system has become simpler in a corresponding manner.

However, a significant amount of engineering work is still necessary to put a distributed control system out into the field Special hardware has been developed as a part of the distributed control system architecture to make this process as painless as possible. This engineering interface hardware must perform functions in the following categories

System configuration—Define hardware configuration and interconnections, as well as control logic and computational algorithms;

Operator interface configuration—Define equipment that the operator needs to perform his or her functions in the system, and define the relationship of this equipment with the control system itself; System documentation—Provide a mechanism for documenting the system and the operator interface configuration and for making changes quickly and easily;

System failure diagnosis—Provide a mechanism to allow the instrument engineer to determine the existence and location of failures in the system in order to make repairs quickly and efficiently.

4.9.1 General Requirements

The engineering interfaces in a distributed control system must be designed for the appropriate class of users of the equipment. These ergonomic requirements and several other requirements dealing with the specific functions of the engineering interface are as follows:

Access security—The engineering interface defines and modifies the control logic and tuning parameters (among other items) of the process control system. Therefore, the system must include a security mechanism to keep unauthorized users from getting access to the system configuration through the engineering interface.

Ergonomic design—The instrumentation and control engineer using the engineering interface equipment is likely to have more technical training than the process operators; however, the engineer often will not have a computer hardware or software background. Therefore, the engineering interface should be designed to accept inputs through an interactive prompt-andresponse sequence, rather than through entering of data tables or program statements. Data input formats should be in a form understandable to a process instrumentation engineer, not to a computer expert (no octal or hexadecimal. data formats, please). If the distributed control system has more than one type of engineering interface, there should be consistency of user interactions from one device to the other. Data reasonableness and consistency—As much as possible, the engineering interface system should be able to check an engineer's inputs for reasonableness and for consistency with previously entered information. If there is a problem, the system should advise the engineer of the situation and provide prompts that allow the engineer to correct the problem. In addition, it should not be necessary to enter the same information into the system more than once (e.g., an engineer should not have to enter an alarm limit once for alarming and again for trending). Also, the engineering interface system should keep track of the information entered to ensure consistency. If more than one engineering interface device is in use in the system at the same time, there must be a mechanism in the system to make sure that changes made by one such device are picked up by the others. Finally, if the engineering inputs for several related pieces of hard- ware and control logic must be consistent (e g , in defining parameters for a controller, an operator station, a termination unit, and a PID control algorithm), the system itself should provide a check of consistency and notify the user of any errors.

User convenience—The engineering interface devices must be designed for convenient use. For example, it should not be necessary to shut the control system down in order to connect or disconnect the engineering interface to the system. Similarly, the devices should be designed for easy storage when not in use

Cost-effectiveness—The cost to the user of the engineering interface must be small when compared to the total installed cost of the control system.

4.9.2 System Configuration Requirements

Setting up any process control system requires a certain amount of manual work determining control system requirements, selecting and procuring hardware, physically assembling the various modules and other elements into cabinets, and connecting external wiring between transmitters and field termination points in the control cabinets However, the introduction of distributed control systems has made it possible to automate quite a number of system configuration functions The degree of automation depends on the level of sophistication of the

engineering interface equipment used. Whatever the level, the engineer must perform the following system configuration functions (as a minimum) in implementing a distributed process control system: The engineer must define the addresses of input and output points in the system with respect to their location in the shared communications facility (e.g. , by identifying the point number within the module, the module number within the cabinet, and the cabinet number within the communication hierarchy) This is equivalent to assigning telephone numbers (by area code,. exchange, and number) to people in a country so that you know where to reach them.
The engineer must define the tag names and associated descriptors that humans will use to identify certain (not necessarily all) points in the system, and relate these tags to the hardware addresses. (It is much more convenient for a human to describe a thermocouple input point as ‘‘ TT 106—Reflux Temperature ‘‘ than as point number 27—5— 18.”) The engineer must define any signal conditioning that needs to be done with the input points in the system: e.g,
linearization, zero and span shifting, or conversion to engineering units Similar conditioning functions may need to be defined for control outputs to compensate for non linear control drive characteristics, for example. The engineer must select, configure, and tune the control and computational algorithms in the system. Data to enter and verify include PID and other control algorithms, tuning constants, alarm limits, logic sequences, batch control recipes, and
high-level language statements (e g , in BASIC or FORTRAN) The engineer also must select any auxiliary functions that the system is to perform on certain points, such as trending, logging, and long-term data storage and Retrieval . When the implementation of the control and computational algorithms in item (4) requires transmitting data through the shared communications facility, the engineer must define the linkages from one element to another.

4.9.3 Requirements for Operator Interface Configuration

Configuring the control and computational functions of the distributed control system is only part of engineering a process control system. An equally important aspect of the job is selecting and configuring the operator interfaces to the system. The degree of support for this that the engineering interface can provide depends on how the operator interface system is implemented through panelboard instrumentation or through VDUs. Ideally, the engineering interface should support at least the following functions: It should allow the engineer to select and define the devices and mechanism the operator will use in running the process. These include hardware such as control stations, indicators, recorders, mimic panels, alarm indicators and related devices (or their equivalent in CRT displays). It should relate the operator interface devices or displays to the control and instrumentation hardware in the field. This includes labeling the devices or displays with the tags, descriptors, and engineering units appropriate to the process points being controlled or monitored.

4.9.4 Documentation Requirements.

Documenting the hardware configuration and functional logic diagrams of an industrial process control system is one of the most painful and costly aspects of engineering the system. In conventional control systems, almost all of this documentation is accomplished manually. Distributed control systems that include engineering interfaces can change this situation significantly In these systems the instrumentation engineer enters the bulk of the configuration information through the engineering interface and stores it in a mass data storage facility For this reason a great potential clearly exists for automating the documentation function This potential

can be realized if the engineering interface meets the following requirements The engineering interface system must include a hard-copy device such as a printer to support the documentation function. The documentation system must support both tabular and graphical data formats The formats must be adjustable so that the user can adhere to the company's in-house documentation conventions.

One of the most difficult tasks of the instrumentation engineer is to keep track of field changes to the control system. The documentation function in the engineering interface must handle these changes as well as handle the original engineering design As much as possible, the documentation process should be automatic, requiring no special actions on the part of the instrumentation engineer. Experience has shown that documentation will not be done if it is too much trouble.

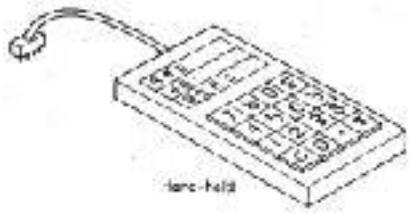
4.9.5 Diagnosis of System

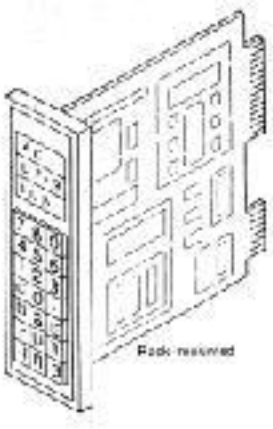
The increased intelligence of the individual devices in a distributed control system allows them to implement self-diagnostic algorithms and report any failures directly to the maintenance personnel in the plant. The engineering interface unit is one of the key mechanisms by which this failure information is reported to the instrumentation and control system engineer. The engineering interface unit can also be a very useful tool in debugging and troubleshooting control logic. To support the diagnosis of system problems, the engineering interface system must meet the following requirements:

- There must be mechanisms that allow the instrumentation engineer to identify any failed devices in the system down to the level of repairable elements These include modules, sensors, power supplies, and communication devices
- If a partial failure of an element occurs, there must be a mechanism that allows the instrumentation engineer to determine the severity and nature of the partial failure in order to identify the best course of action to solve the problem
- There must be mechanisms to expedite troubleshooting of control s stems during initial checkout as well as during on-line operation These mechanisms must be applicable to continuous, sequential, and batch control systems
- Diagnosis of process problems is not within the scope of the engineering interface system. This diagnosis is important, but it is an application-oriented function implemented within the distributed control and computing system itself

4.10 LOW LEVEL ENGINEERING INTERFACES

The low-level engineering interface is usually a microprocessor-based device designed either as an electronic module that mounts in a rack or as a hand-held portable device. To minimize cost, the device usually is designed with a minimal keyboard and alphanumeric display so that the instrument engineer can read data from and enter data into the device Some LLEIs use small CRTs or flat-panel displays to implement the human interface. Some versions of the LLEI must connect directly to and communicate with only one local control unit or data input / output unit at a time





More sophisticated versions can connect to a local branch of the shared communications facility and are able to communicate with any one of several LCUs or DI / DOs in adjacent cabinets or areas of the plant. In either case, the LLEI can be connected or disconnected while the LCU or DI/OU is powered and in operation, it is not necessary to shut the process down Some devices include interfaces to low-cost peripherals, such as a printer (for documentation purposes), or a mass memory device, such as a tape cassette or microfloppy diskette (for storage of configuration data)

In general, the LLEI is a dedicated device that is not used for operational purposes Therefore, when it is not needed for engineering use, it can be disconnected from the distributed control system and locked up for safe- keeping This keeps the control configuration and tuning constants secure from alteration by unauthorized personnel. Because of the minimal display and user input capabilities built into the LLEI, the interaction between user and device is often not as user-friendly as it is between user and high-level device. There is usually heavy reliance on using code words, symbols, or dedicated function keys to implement man-machine communications in the low-level device The engineer generally has to document the input data on configuration forms first, then enter the data through the interface However, the entry of data can be designed to take place in an interactive request-and-response manner. The interface device can also make some basic reasonableness and consistency checks on data entries and input sequences and give the engineer error messages when it detects a problem The displays and input keyboard can be made flexible enough so that instrument engineering personnel can enter and read out data in engineering units that they understand.

4.10.1 System Configuration

When the system provides only an LLEI, the hardware in the system is selected and configured manually To simplify this task, most vendors of distributed systems provide a systems engineering guide or similar document that leads the user in a step-by-step manner through the hardware configuration procedure In this situation, the primary purpose of the LLEI Is to provide a tool for configuring the algorithms in the system controllers (i e , entering control strategies, setting tuning parameters, selecting alarm limits, and so forth). Some low-level interfaces have a removable mass memory device (such as a small cassette, diskette drive, or magnetic strip) incorporated in their designs for storage of control configurations When connected to a power supply, the engineer can then use the interface to develop and edit control strategies even if the controller itself is not connected All that is required is that the configuration limitations of the target controller be stored in the engineering interface device. The availability of mass memory also makes it possible to download control configurations into the target module from the

engineering interface, or conversely to "upload" the configurations from the controller. This function is very convenient when it is necessary to install a control configuration in a spare controller used to replace a failed controller. In general, the LLEI is not designed to support generation and editing of high-level language programs such as BASIC and FORTRAN. A separate terminal or HLEI is required to provide this capability.

4.10.2 Operator Interface Configuration

The distributed control systems that require only a low—level engineering interface generally are small ones involving a limited number of control loops In this type of system, the operator interface also is simple, usually consisting of a small number of dedicated panelboard instrumentation devices (with no shared CRT-based console). Therefore, all configuration of the operator interface is done manually.

As in the case of all panelboard instrumentation, the connections between the stations and indicators and the controllers in these small systems are established through the manual process of hard wiring or cabling. Assigning tags to the control loops and system inputs and labeling the stations and indicators with these tags and the corresponding engineering units also are manual operations. These tasks can he very time consuming and prone to error. Also, changes are difficult to make, since there is a significant lead time involved in procuring the new panelboard hardware, cabling, and labels needed to implement the changes.

4.10.3 Documentation

In a small system that requires only a low-level engineering interface, the vendor usually provides the user with very little or no help in automating the process of documenting the hardware and control configurations All documentation is a manual process, sometimes with the help of standard forms in the system engineering guide When changes are made to a system of this type in the field after initial start-up, documenting these changes is a manual process that can be hazard. It requires great discipline for the user organization to ensure that these field changes are tracked and identified accurately.

4.10.4 Diagnosis of System Problems

The LLEI, since it is not always connected to the distributed control system and often is not even located in the control room, provides little help in the way of notifying the operator of hardware failures in the control system. The operator must rely on the self-diagnostic capabilities of the distributed equipment itself to detect and make known the failure to the operator. Failures of this equipment generally are indicated at the control cabinets through alarm lights on the individual pieces of equipment. The failures are alarmed in the control room either through a conventional annunciator panel or through indications on the panelboard instrumentation. However, the LLEI can be very useful in helping the instrumentation and control system engineer identify the specific location and nature of the failure. For example, the LLEI can interrogate a controller or other element that has failed; the element, if it still

can communicate, can report on the specific problem that caused the failure through the engineering interface Also, the instrument engineer can use the LLEI as a digital voltmeter to trace through the control logic and identify problems. Some LLEIs are equipped with a verify function that allows an automatic comparison of the actual configuration in the controller with the correct one stored in the interface's mass memory. This function can be very helpful in checking out a controller during initial start-up and in making sure the configuration hasn't changed after it has been in use in the field for some time.

4.11 HIGH LEVEL ENGINEERING INTERFACES

The high-level engineering interface allows a user to reap the full benefits of the flexibility and control capabilities of a distributed control system while minimizing the system engineering costs associated with such a system. The HLEI is implemented in the form of a CRT-based console, or VDU, similar to the high-level operator interface unit. The internal architecture of the VDU is modular, using multiple microprocessors. This approach provides a significant amount of flexibility in accommodating hardware options for engineering purposes (e.g., special keyboards or printers).

4.11.1 Dual-Console Functions: In a few distributed systems, the engineering console is a specialized device that is dedicated to the engineering function. In most offerings, however, the engineering console can also be used as an operator's console; a key lock on the console implements the switch between the two console "personalities The first position permits only operator functions (e.g. , display selection, control operations such as mode selection and set-point modification., and trend-graph selection). The second position allows engineering functions (such as control logic configuration, modification and tuning, and system documentation) as well as operator functions.

Some systems provide a third key-lock position that allows the user to perform tuning operations but does not allow the user to modify the control logic structure Implementing dual console ' personalities" in a single piece of hardware is very cost-effective for the user, since the engineering functions are needed only during initial system start up and occasionally thereafter when system modifications are made. During the other periods of time, the hardware is put to good use as an operator s console. However, the console is always connected to the system and is conveniently available when engineering functions need to be performed. The incorporation of a key lock ensures that the system configuration is secure while allowing authorized plant personnel access to the system .

Special Hardware Required : The engineering functions are much more oriented towards data entry and documentation than are operator functions Because of this, the hardware requirements for an engineering console are somewhat different than those for a console used strictly for

operations . One major difference is in the keyboard provided for the user The operator's console uses a flat-panel, dedicated-function keyboard for ruggedness and simplicity of operation The engineer's console, however, requires a general purpose keyboard to promote speed of data entry and to support wider range of human interface functions.

This keyboard usually is a push-button type whose keys are laid out either in the traditional QWERTY configuration found on most typewriters or in the more recent Dvorak configuration (a human-engineered version that repositions the letters to promote faster touch typing). The vendor may supply additional special-purpose keys to allow the user to select special characters, symbols, and colors employed in generating control configurations and displays. To further simplify the process of building custom color-graphic CRT displays, some vendors also provide a digitizer tablet and stylus, a light pen, or other device as auxiliary hardware for the engineering console. (A digitizer
tablet is a touch-sensitive pad on which the engineer enters information using the stylus) This hardware supplements the cursor keys, touch screens, and other human interfacing hardware supplied with the operator's console. When the plant operator is using the console, the special keyboard or auxiliary hardware usually is stowed away in a storage location.

Only when the console's personality is changed to become an engineer's console is this hardware brought out In addition to this auxiliary hardware, some vendors also supply special color-graphic printing or plotting devices, which allow more sophisticated system documentation to be generated than is possible with the standard printer that comes with an operator's console.

4.11.2 Portable Engineering Interface : Some distributed control system vendors offer a CRT-based engineering interface device that is a compromise between the full-function engineering console and the minimum-function low-level device. Usually, this is a portable unit that includes a bulk memory device such as a floppy disk or cassette tape drive for storage of system configuration data This unit generally is designed to plug into and interface with a single LCU or cab- met It can be very useful and cost-effective device for certain system configurations. However, its design will not be discussed here since its functions are a subset of those implemented in the HLEI console.

4.11.3 System Configuration

Because of its computational, display and data storage capabilities, the HLEI can play a major role in automating the process of configuring a distributed control system. The entire database that defines the configuration of the hardware, control structures, and computational algorithms of the distributed system can be stored in the HLEI.
The following information dealing with the system hardware configuration can be entered by means of an interactive dialog between the engineering interface and the instrument engineer :

- Number, type, and location (relative to the shared communications facility) of each hardware module in the LCUs and DI/OUs in the distributed system;
- Definition of any hardware options (such as expansion hardware, jumper positions, or switch settings) selected on each module;
- Definition of each input point (by type and hardware address) in each of the hardware modules;
- Number, type, and location of all operator and engineering consoles in the system;

Number, type, and location of any other devices in the system that communicate using the shared communications facility (e.g. , computer interfaces, and special logging or computing devices). While the instrumentation engineer enters most of this information manually, the HLEI can acquire some of the data (such as the addresses and types of devices using the shared communications facility) automatically. It can do this by means of a sequence of broadcast messages through the communications facility asking any devices that are listening to identify themselves and transmit any configuration information that they have When implemented, this approach can save a substantial amount of the engineer' s time and effort and cut down dramatically on the number of configuration errors introduced.

In either case, once this information is entered in the system, the engineer uses the HLEI to make reasonableness and consistency checks on the information, save it on a mass memory medium, and document it in a printed format. During normal operation, this information is not used anywhere else in the distributed system. It is saved to document the hardware configuration installed in the plant for the instrument engineers' use and for downloading in case of hardware failure and replacement.

In addition to this hardware-related information, the engineering interface is also used to store information dealing with the control and computational functions of the system. At a minimum this information includes the following Point tags and associated tag no, engineering unit definitions , and related point addresses, Logic state descriptors (e g , on/off, and run/stop) that correspond to digital points in the system

Control algorithms implemented in each of the LCUs, Signal conditioning and linearization algorithms implemented in the DI/OUs, Communications linkages needed to transfer information among the control and computational algorithms in different LCU s and DI/OUs in the system using the shared communications facility; High-level language computational algorithms implemented in the LCUs.

4.11.4 Control and Computational Logic Configuration

One of the major benefits of using an HLEI in a distributed control system is that it simplifies the process of configuring the control and computational logic in the system. The high-level console provides one or both of the following control logic configuration capabilities :

- A fill-in-the-blanks function in which the instrument engineer interacts with the console through a sequence of prompts and responses to configure the control logic in the system
- A graphics capability in which the instrument engineer draws the control configuration on the screen with a light pen or on a digitizer tablet with a stylus This capability is similar to that provided in computer-aided-design (CAD) systems, and allows the user to define the control logic using a familiar function block format.

After laying out the basic control structure graphically, the instrument engineer can define the parameters associated with the function blocks through an interactive sequence.

Storage of Configurations As described previously, one of the basic functions of the HLEI is to provide the engineer with a tool for configuring the control logic and computational algorithms in the target hardware that will execute the algorithms. However, the HLEI can be used to perform additional useful functions if it includes a mass memory facility that allows duplicate storage of this configuration information. The mass memory facility is generally implemented using one or more types of storage media: floppy disk, hard disk, magnetic tape, bubble memory, or optical disk The additional functions that configuration storage allows include the following:

- The engineer can use the interface to configure the control logics and computational algorithms without the presence of the target LC1 In this situation, the configurations are stored directly into the mass memory of the interface. Of course, the interface must be aware of the limitations of the target LCU (in terms of memory size and co4utational speed, for example) and ensure that the configurations entered do not exceed these limitations. After storage, the engineer can then download the configurations at a later time to the LCUs once this hardware becomes available This feature can dramatically cut the time necessary to deliver a control system, since the job engineering can begin and the control logics can be configured and stored long before the actual target hardware is available.
- The HLEI can be used to verify that the engineer has correctly entered the control logic and algorithms into the LCU and that they have not been altered; the HLEI does this by performing a bulk comparison of the logic in the LCU with that stored in the mass memory Any mismatches are displayed for the engineer, who can correct them immediately If an LCU or other element of the system fails, it can be replaced by a new element. The appropriate control and computational algorithms can then be downloaded from the mass memory to the new or repaired element This results in a very short mean time to repair for that element and a minimum downtime of the function performed by that element

- The engineering interface also should support the uploading of database information from a hardware device to the interface This capability helps in documenting functions performed by modules or elements that are removed from service or moved from one control system to another.

4.11.5 Operator Interface Configuration

When a high-level operator interface is used in the DCS, there must be a mechanism that allows the instrument engineer to configure or change its display structures and parameters in a convenient manner. In most DCS, this mechanism is the high level engineering interface. This device allows the engineer to change the layout of the series of displays that make up the ‘‘panel board’‘ used to control plant operations.

For maximum flexibility, the engineering interface must be designed in such a way that it can configure any HLOI in the DCS. That is, the system should be able to download displays and display hierarchies configured on one engineer’s console to any other operator’s console in the system over the shared communications facility. This capability implies that displays configured for one operator’s console can be duplicated on other consoles without the need for people physically transporting magnetic storage media (e.g., floppy disks) from one console to another.

The first step in configuring an HLOI is to structure a hierarchy of displays. This includes specifying the following information as a minimum:

- Number of areas in the plant and their identifying tag names and descriptors,
- Number of groups in each area and the tag names and descriptors for each area,
- Assignment of control loops and input points to a group or to multiple groups,
- Types of displays to be used at each level, whether preformatted displays or custom graphics;
- Linkages between displays in the hierarchy (i e , the paths that the operator must follow in moving from one display to. another; these linkages are pre specified in some systems and configurable in others,
- Assignment of points in the system to the special functions of the operator console trending, alarming, logging, and long-term data storage and retrieval, specification of the parameters involved in each special function (e g , trend periods, logging formats, and data storage frequency)
- Defining and laying out graphics displays is a more challenging task, since they are usually completely user-configurable and do not follow any standard format. Of course, the vendor’s display hardware imposes limitations on the structure of these displays, such as resolution, number of colors available, pixel control method (character oriented or bit-mapped), special characters and symbols, and so forth.

Usually, the vendor provides a graphics display engineering kit that spells out these limitations and helps the user to sketch each display on a special grid form before entering it into the engineer’s console In laying out each display, the engineer must enter the following information as a minimum:

4.11.6 Graphic symbols—The engineer must define the new graphic symbols to be used in the operator interface displays Usually, the system vendor supplies a base set of symbols that

represent standard items of process equipment These include symbols for piping, valves, pumps, and so forth The ISA standard provides a typical list of these symbols However, each user generally has a need to configure additional symbols that are appropriate to the particular application The symbol library is expanded to include these new symbols along with the standard ones, they are then also available for incorporation in the user's displays.

4.11.7 Static background elements—These are elements in the display (both graphic and alphanumeric) that do not change with time These usually include the major items of process equipment, the interconnecting piping, the equipment labels, and engineering unit identifiers The information required for each element includes its shape, size, and color

4.11.8 Dynamic graphics elements—These are the graphics elements in the display that change to reflect corresponding changes in the process. For example, pumps may change color when they are turned on and off, piping may change color when it contains fluid, the displayed levels of fluid in tanks may move up and down to reflect actual fluid levels , bar graphs may change to depict temperature profiles. Or the actual symbols may change For example, one symbol may represent a closed valve and another symbol an open valve In each case, the dynamic element must be linked to the corresponding tag that defines the process state represented by the graphics element For instance, the engineer can link a tag representing a digital process input to the graphics element illustrating an open or closed valve. In addition to the tag definition, the engineer must specify how the dynamic element will change (e g ,in shape or color) in response to the change in tag state and enter this information into the definition of the display.

4.11.9 Dynamic alphanumeric elements—The engineer must also define and link these to the corresponding tags. These elements include alphabetic labels that change color, wording, character size, or brightness as a function of process conditions. They also include numeric values or logic states that the VDU updates to reflect changes in the process variables represented by the corresponding tags. Alarm conditions can be indicated by color changes or by having the VDU place the element into a blinking condition. Usually, there are limits to the number of dynamic elements that a single graphics display can in— dude, 200 to 400 elements is a typical range

4.11.10 Control stations—If the graphics display structure supports continuous or sequential control operations (or both), the engineer must define the controllable elements in the graphic display and label them with tag numbers, reference numbers, or both. This allows the operator to call up the appropriate control stations on the graphics display and connect them to these controllable elements

4.11.11 Poke points —If the operator's console provides a touch screen input capability, the display specification will include the definition of poke points or fields that will be sensitive to operator inputs. As in the case of the dynamic display elements, the engineer must link each poke point to the corresponding tag that represents a push- button input from the operator.

4.11.12 Documentation

The high-level engineering interface can provide the user significant benefits in the area of documentation. After the configuration process has been completed, all information defining the hardware, control logic, computational algorithms, and displays for the system is stored in mass

memory in the engineering interface. This makes it possible to completely automate the process of documenting this information in a hard-copy format. Ideally, the intelligence and printing and plotting hardware in the HLEI should be designed to generate the following documentation automatically:

- Lists of hardware modules in the distributed system, including their location by number of LCU or DI/OU.
- Documentation of the control configuration and associated tuning parameters for each LCU, in both listing and graphic formats showing the control system structure, listing includes any high-level language programs or batch control recipes executed in the LCUs.
- Listing of the tags in the distributed system, along with the associated tag descriptor information and the hardware addresses of the physical inputs or module outputs that correspond to these tags.
- Listing of the special operator interface functions that are associated with each tag, such as trending, logging, and long-term storage and retrieval functions.

Definition of the operator displays in the system, including a drawing of the display hierarchy, a listing of the displays in the hierarchy, and a printout of the formats of each display.

4.12 Diagnosis of System Problems

Most of the hardware is microprocessor-based and has the intelligence to perform on-line selfdiagnostics to evaluate its own "health." When a failure or other problem occurs, this hardware is aware of it and reports the problem to the higher-level elements in the distributed system.

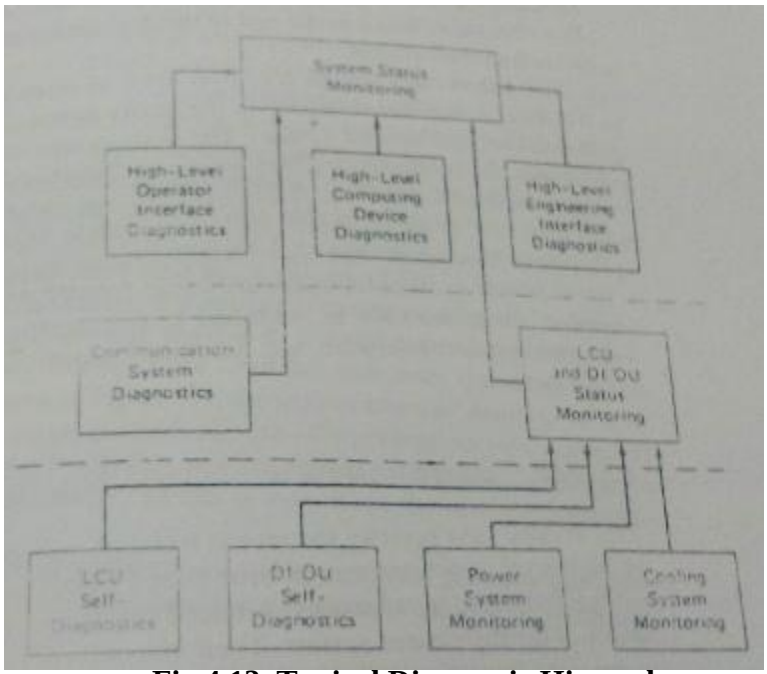


Fig.4.13 Typical Diagnostic Hierarchy

Figure shows a typical diagnostic hierarchy. At the lowest level, the LCUs and DI/OUs perform diagnostics, and report the results up through the shared communications facility The communication system elements also check their own ‘‘health’ and provide status reports to the system-level elements in the distributed system the high level operator interfaces, engineering interfaces, and computing devices. Finally, these system-level elements also execute their own online diagnostics and report the results to the other system-level elements in the system. The overall diagnostic status of the equipment in the system is indicated on the top line of each display in the HLOI system. If a problem develops, an equipment status alarm goes off and the operator or instrument engineer can call up a hierarchy of diagnostic displays to pinpoint the problem.

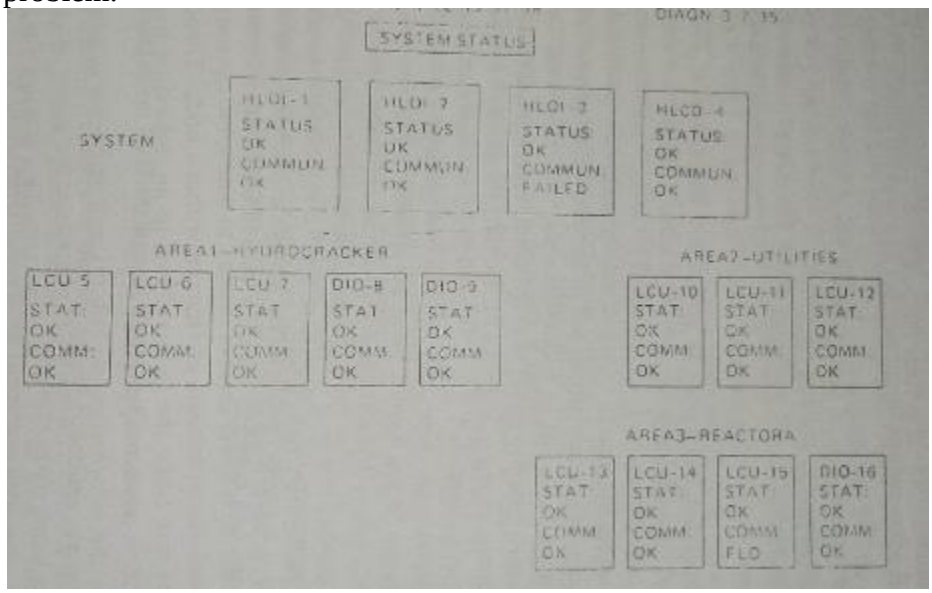


Fig. 4.14 Example of a Diagnostic Display

Figure shows an example of the highest level of display in this hierarchy This display shows a map of the nodes or drops on the shared communications facility, these contain the major hardware elements that exist in the distributed system Any failure in the elements within a node are reported on this display If a failure or problem is identified at this level the operator or instrument engineer can call up the next lower level of display to determine the cause of the problem in the individual module or element within a node Through this method, the user can trace the failure down to an individual hardware element, which then can be replaced or repaired. Since the information on failures and problems is available within the distributed system through this diagnostic hierarchy, it can also be logged on a printer and saved in a long-term data storage file. This allows the user to accumulate statistics on the rates and modes of failure of the various pieces of hardware in the distributed system. This information is essential to planning a cost effective strategy for maintaining a spare-parts inventory and in providing inputs to an overall program for managing plant maintenance.

UNIT- IV Part- A

Sl.no	Part A - Questions	CO	Level
1	State the geographically centralized and geographically distributed control system	CO2	L1
2	List the communication facilities in DCS	CO4	L1
3	Express the different types of operator display in DCS	CO2	L2
4	Point out some of the devices used for LLOI interface.	CO2	L4
5	Compare LLOI and HLOI.	CO2	L4
6	Define Plant level display	CO2	L1
7	Discuss the meaning of area level display	CO2	L2
8	List the requirements of engineering interface	CO2	L1
9	Predict some of applications of DCS	CO2	L2
10	Name some of the DCS communication system standards	CO4	L1
11	Describe about Shared communication facility	CO4	L1
12	List the different network topologies of DCS	CO4	L1

UNIT- IV Part – B

Sl.no	Part B - Questions	CO	Level
1	Explain in detail about low and High level operator interfaces in DCS	CO2	L2
2	Discuss in detail about high level and low engineering interfaces	CO2	L2
3	List the different operator displays in DCS, brief all the operator display	CO2	L1
4	Express the application of DCS in oil industry	CO5	L2
5	Explain in detail about the application of DCS in gas industry	CO5	L2
6	Point out the different architectural issues in DCS	CO2	L4
7	Explain the Protocol issues of DCS	CO4	L2
8	Categorize the Different communication system standards of DCS	CO4	L4

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS
DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT 5 – COMPUTER CONTROLLED SYSTEMS – SIC1405

UNIT 5 COMPUTER CONTROLLED SYSTEMS

Basic building blocks of Computer controlled systems - SCADA - data acquisition System - supervisory Control - Direct digital Control - software - Velocity algorithm & Position algorithm

5.1 Computer Control System

Need for computer in a control system- functional block diagram of a control system- direct digital control supervisory control-scheduling control- digital control interfacing: process inputs, outputs interface-data loggers-data acquisition system- type of displays, computer - various input/outputs- computer control action- treatment of inputs, outputs, and control strategies.

5.2 Need for computer in a control system

Computers are also used for controlling and monitoring tasks. Such computers are called as controlsystems. Like any other computer, control systems are also made up of three parts:

1. Input devices called sensors feed data into the computer.
2. The computer then processes the input data (according to the program)
3. As a result of the processing, the computer can turn on or off output devices called actuators.

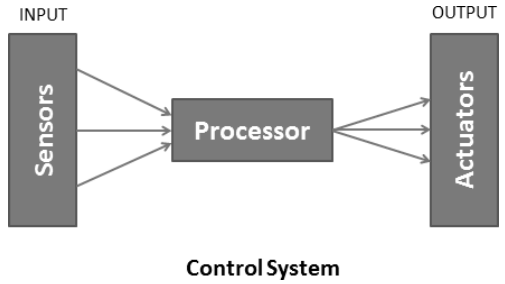


Fig. 5.1 : Computer Control System

Computer has no senses, but it is possible to connect sensors to it. A sensor is a device that converts a real-world property (e.g. temperature, light, pressure etc.) into data that a computer can process.

Example for sensors is:

- Temperature sensor
- Pressure sensor
- Humidity sensor
- Light sensor
- Sound sensor

Actuators

An actuator is a device, controlled by a computer, which can affect the real-world. Some of the examples are – motor, pump, buzzer , heater etc.

Processing (decision making)

The steps followed by the computer in a control system are just about the same for all systems:
Check the data from the sensors.
If necessary, turn on/off one or more of the actuators. Go back to step 1.

ADC (Analogue to Digital Converter)

Computers are digital devices, hence only understand digital data. Whereas sensors are analogue devices and produce only analogue data. So to make the computer able to understand the data sent by the sensors there is a need of converter called Analogue to Digital Converter.

DAC (Digital to Analogue Converter)

Sometimes computer will have to send instructions/commands to actuators. computers are digital devices and actuators are analogue devices. To make this communication possible,it is must to use a converter between the computer and actuators, called Digital to Analogue Converter (DAC)

Advantages of using computer control

1. Computers don't need breaks - they can work 24 hours 7 days without stopping.
2. Computers don‘t need to be paid. To buy and install a computerized control system can be very expensive, but, in the long-term, money is saved by not having to employee staff to do the work.
3. Computers can work in extreme conditions that human beings cannot. e.g. nuclear power stations, chemical factories, paint-spraying areas.
4. Computers are more accurate than human beings.
5. They respond more quickly than a human could.

Functional block diagram of a control system

In continuous time control systems, all the system variables are continuous signals. Whether the system is linear or nonlinear, all variables are continuously present and therefore known (available) at all times.

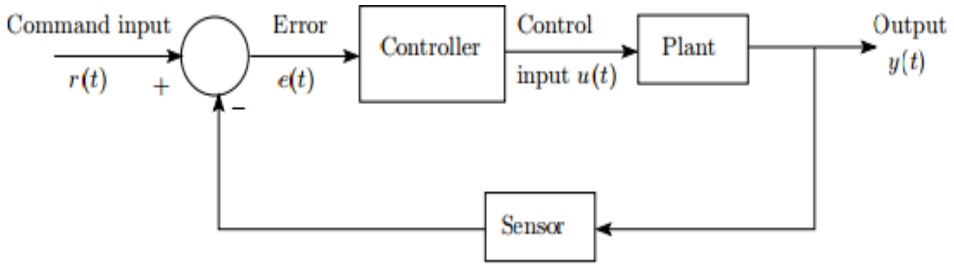


Fig. 5.2: A typical closed loop continuous time control system

In a digital control system, the control algorithm is implemented in a digital computer. The error signal is discretized and fed to the computer by using an A/D (analog to digital) converter. The controller output is again a discrete signal which is applied to the plant after using a D/A (digital to analog) converter.

5.3 Basic Elements of Control System

Plant: A physical object to be controlled.
Feedback Control System (Closed-loop Control System): A system which compares output to some reference input and keeps output as close as possible to this reference.
Open-loop Control System: Output of the system is not feedback to the system.
Control Element: These are the components required to generate the appropriate control signal.
Feedback Element: These are the component required to establish the functional relationship between the primary feedback signal and the controlled output.
Reference Input: This is an external signal applied to a feedback control system in order to command a specified action of the plant. It often represents ideal plant output behavior.
Controlled Output: controlled output is that quantity or condition of the plant which is controlled

5.4 Discrete time Control System

Discrete time control systems are control systems in which one or more variables can change only at discrete instants of time. These instants, which may be denoted by kT(k=0,1,2,…) specify the times at which some physical measurement is performed or the times at which the memory of a digital computer is read out.

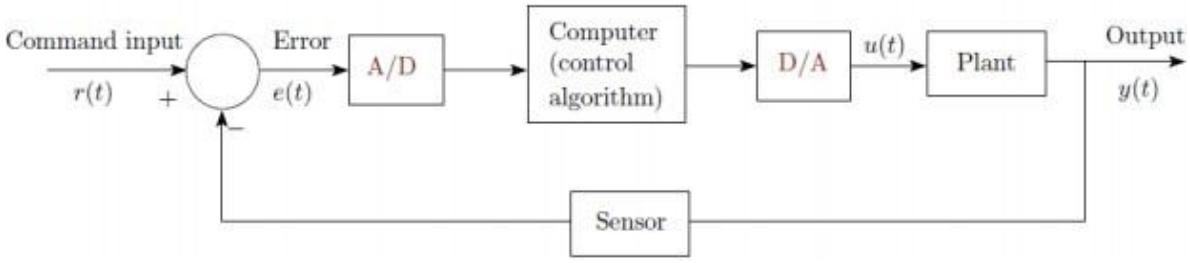


Fig. 5.3: Block diagram of discrete time control system

Continuous time control systems whose signals are continuous in time are described by differential equation, whereas discrete control systems that involve sampled data signals or digital signals and possibly continuous time signals as well are described by difference equation.

5.4 Direct digital control supervisory control

- Direct digital control uses a programmable digital computer to process information for the purpose of determining the correct control action.
- The input information for the computer comes from analog sensors (such as temperature sensors) and digital sensors (such as switches).
- This information is used as variable data in a predetermined set of instructions called the control program
- The word "direct" in direct digital control implies that the controller has immediate control over the final control element.
- Feedback control system where the controller action is attained numerically by a programmable digital device.

Construction and Working

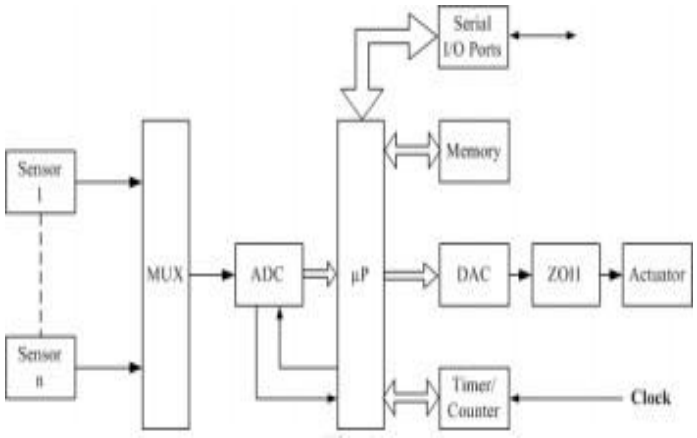


Fig .5.4: Block Diagram of Direct Digital Control

The DDC directly interfaces to the process for data acquisition and control purpose. Because of the nature of digital devices, signals from the plant have to be converted into a suitable form before they can be transferred for processing by computer. Similarly, signal generated by the computer must be presented in a form compatible with the plant.
The multiplexer acts like a switch under microprocessor control. It switches and presents at its output the analog signal from a sensor / transmitter. ADC transforms analog values obtained from sensors into digital signal so that a digital processing unit can work on them. The output of digital control logic is linked with final control elements through DAC. If pneumatic or hydraulic action is required, it is accomplished with electronic-to-pneumatic or electronic-to hydraulic converters.

The microprocessor performs the following tasks 1. It reads the various process variables from different transmitters through multiplexer and ADC. 2. It determines the error for each control loop and executes control strategy for each loop. 3. It outputs the correction value to control valve through DAC.
Other important pieces of hardware that form part of a digital control loop are Sampler: The sampler is essentially a switch, operating usually at fixed intervals of time. When the switch closes, it grabs or samples the signal at that instant of time. Thus, if the source signal is continuous, the output of the sampler is a series of pulses, and the magnitude of each pulse is equal to the magnitude of the continuous signal at the instant of sampling. Sample hold devices: The computer output of DAC is a train of pulses. If this is a control signal, the process will be driven by pulses. This is not acceptable driving the valve at certain opening when pulse is present and closing the valve in between the pulses. To overcome this problem Sample hold device is used which holds the signals from DAC. The most common type is the Zero Order hold (ZOH), where each pulse is held until the next pulse comes along.

5.5 Benefits of DDC:

1. Improved Effectiveness: As the control loop logic is embedded with the software, this logic can be readily tweaked according to the requirements of the process. Based upon continuous monitoring of the process, more complex control schemes, energy and optimization strategies can be implemented.
2. Improved operational efficiency: DDC has the capabilities like offering visualization and storage of data in various formats, trend analysis of data for fault diagnosis and preventive maintenance schedules. They increase the operational efficiency of the plant. Communication capabilities permit remote monitoring and control and help designers to visualize, diagnose and troubleshoot a problem.
3. Increased Energy Efficiency: Energy efficiency routines can be programmed easily. Further, monitoring of energy consumption patterns by each unit permits change of various set points, resulting in efficient utilization of energy.
4. Economy: When a process gets more complex and elaborate, one can easily check that a direct digital control may even be economically more viable considering initial equipment cost, operational cost and savings from performance improvement and modifications.

Applications

- Automation strategy (In steel industry)
- Kiln automation (In cement industry)
- Water treatment plant
- Water distribution control

5.6 Supervisory control
SCADA is an acronym for Supervisory Control and Data Acquisition. SCADA systems are used to monitor and control a plant or equipment in industries such as telecommunications, water and waste control, energy, oil and gas refining and transportation. These systems encompass the transfer of data between a SCADA central host computer and a number of Remote Terminal Units (RTUs) and/or Programmable Logic Controllers (PLCs), and the central host and the operator terminals.

Components of SCADA

1. Human Machine Interface (HMI)

It is an interface which presents **process data to a human operator**, and through this, the human operator monitors and controls the process.

2. Supervisory (computer) system

It gathers data on the process and sending commands *(*or control*)* to the process.

3. Remote Terminal Units (RTUs)

It connects to sensors in the process, converting sensor signals to digital data and sending digital data to the supervisory system.

4. Programmable Logic Controller (PLCs)

It is used as field devices because they are more economical, versatile, flexible, and configurable than special-purpose RTUs.

5. Communication infrastructure

It provides connectivity to the supervisory system to the Remote Terminal Units.

Architecture of SCADA

The block diagram of SCADA system shown in the figure represents the basic SCADA architecture. The SCADA (supervisory control and data acquisition) systems are different from distributed control systems that are commonly found in plant sites. When distributed control systems cover the plant site, SCADA system cover much larger geographic areas.

The below figure depicts an integrated SCADA architecture which supports TCP/IP, UDP and other IP based communication protocols as well as industrial protocols like Modbus TCP, Modbus over TCP or Modbus over UDP. These all work over cellular, private radio or satellite networks.

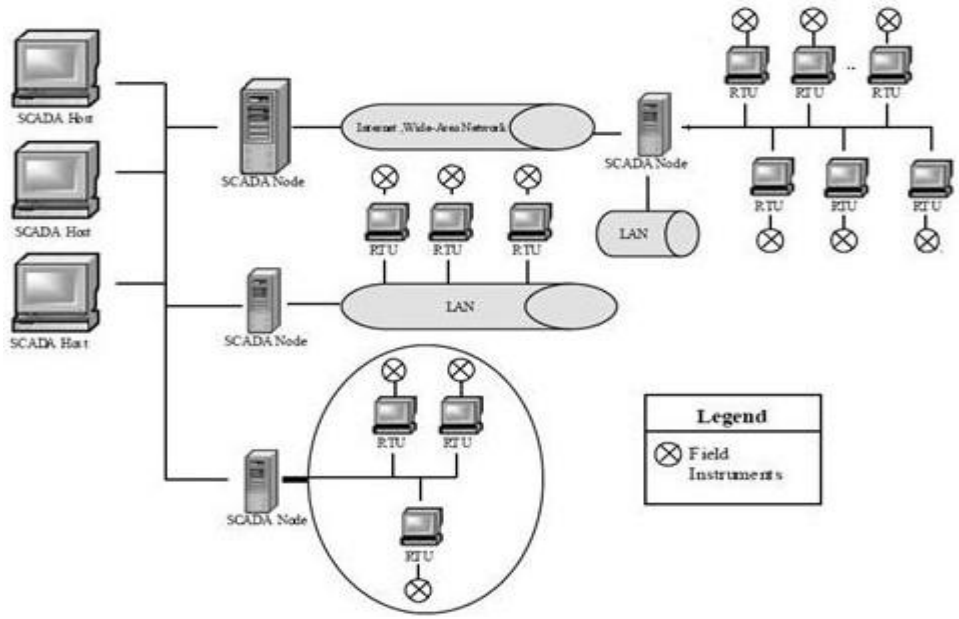


Fig. 5.5 : Architecture of SCADA

In complex SCADA architectures, there are a variety of wired and wireless media & protocols involved in getting data back to the monitoring site. This allows implementation of powerful IP based SCADA networks over landline, mixed cellular and satellite systems. SCADA communications can utilize a diverse range of wired and wireless media.

The choice of the existing communication depends on the characterization of a number of factors. The factors are remoteness, available communications at the remote sites, existing communications infrastructure, polling frequency and data rates. These factors impact the final decision for SCADA architecture. Therefore, a review of SCADA systems evolution allows us to better understand many security concerns.

5.7 Types of SCADA systems

1. First Generation: Monolithic or Early SCADA systems
2. Second Generation: Distributed SCADA systems
3. Third Generation: Networked SCADA systems
4. Fourth Generation: Internet of things technology, SCADA systems

Monolithic or Early SCADA Systems

Minicomputers are used earlier for computing the SCADA systems. In earlier times, during the time of first generation, monolithic SCADA systems were developed wherein the common network services were not available. Hence, these are independent systems without having any connectivity to other systems.

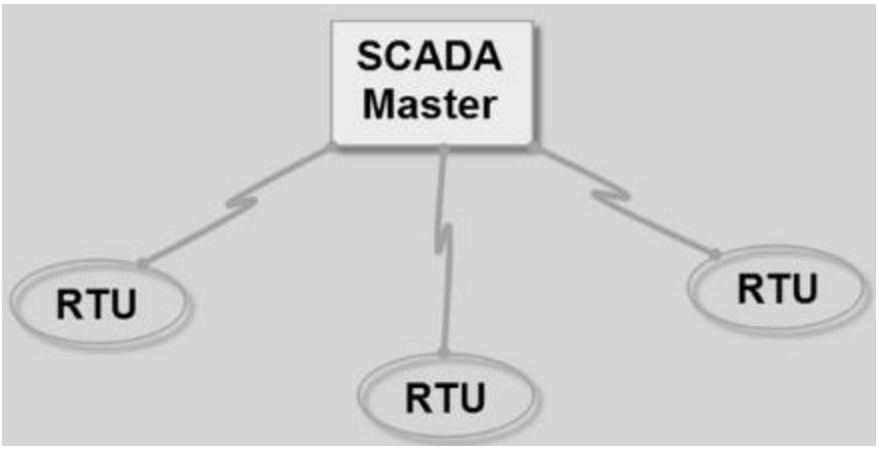


Fig .5.6: Monolithic or Early SCADA Systems

Distributed SCADA Systems

In the second generation, the sharing of control functions is distributed across the multiple systems connected to each other using Local Area Network (LAN). Hence, these were termed as distributed SCADA systems. These individual stations were used to share real-time information and command processing for performing control tasks to trip the alarm levels of possible problems.

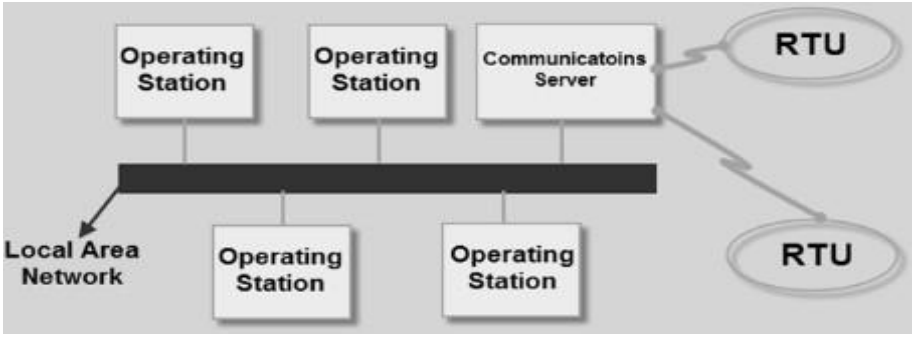


Fig .5.7: Distributed SCADA Systems

The cost and size of the station were reduced compared to the first generation system, as each system of the second generation was responsible for performing a particular task with reduced size and cost.

Networked SCADA Systems

The current SCADA systems are generally networked and communicate using Wide Area Network (WAN) Systems over data lines or phone. These systems use Ethernet or Fiber Optic Connections for transmitting data between the nodes frequently. These third generation SCADA systems use Programmable Logic Controllers (PLC) for monitoring and adjusting the routine flagging operators only in case of major decisions requirement.

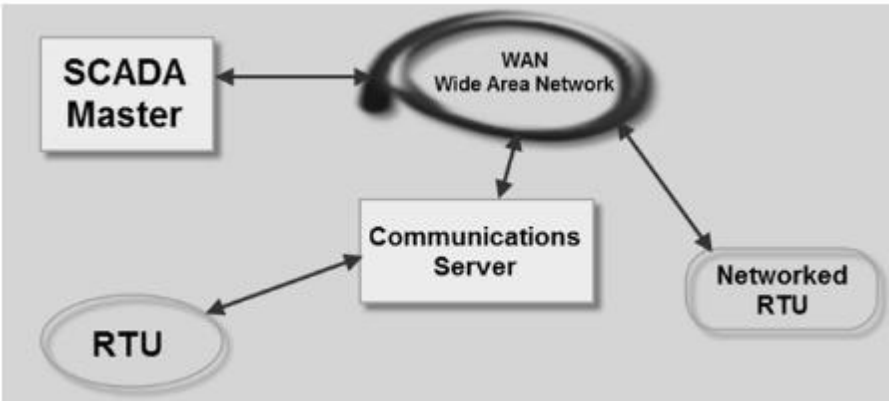


Fig .5.8: Networked SCADA Systems

Internet of Things

In fourth generation, the infrastructure cost of the SCADA systems is reduced by adopting the internet of things technology with the commercially available cloud computing. The maintenance and integration is also very easy for the fourth generation compared to the earlier SCADA systems. These SCADA systems are able to report state in real time by using the horizontal scale from the cloud computing facility; thus, more complex control algorithms can be implemented which are practically sufficient to implement on traditional PLC.

5.8 Applications of SCADA

- SCADA systems are used for monitoring a variety of data like flows, currents, voltages, pressures, temperatures, water levels, and etc.,
- Manufacturing Industries
- Waste Water Treatment and Distribution Plants
- SCADA in Power System

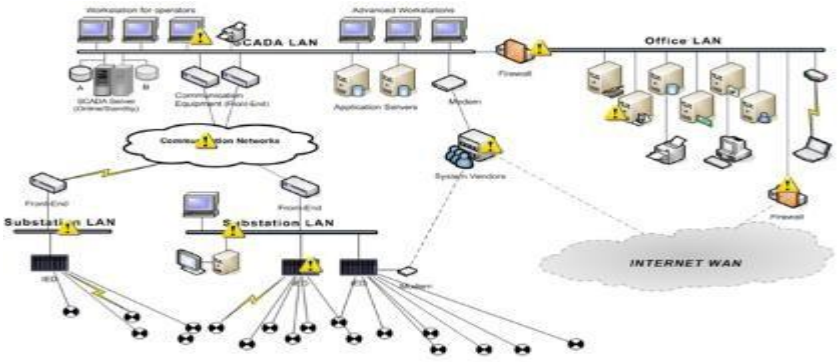


Fig .5.9: Internet of Things

scheduling control

- Scheduling refers to a set of policies and mechanisms to control the order of work to be performed by a computer system
- Of all the resources in a computer system that are scheduled before use, the CPU is by far the most important
- Multiprogramming is the (efficient) scheduling of the CPU
- The basic idea is to keep the CPU busy as much as possible by executing a (user) process until it must wait for an event, and then switch to another process

5.9 Scheduler

In multiprogramming systems, when there is more than one run able process (i.e., ready), the operating system must decide which one to activate. The decision is made by the part of the operating system called the scheduler, using a scheduling algorithm.

In the beginning—there was no need for scheduling, since the users of computers lined up in front of the computer room or gave their job to an operator.

Batch processing—the jobs were executed in first come first served manner.
Multiprogramming—life became complicated! The scheduler is concerned with deciding policy, not providing a mechanism.

Types of scheduling

- Short-term scheduling
- Medium-term scheduling
- Long-term scheduling

Short-term scheduling

Short-term scheduler, also known as the process or CPU scheduler, controls the CPU sharing among the __ready‘‘ processes. The selection of a process to execute next is done by the short-

term scheduler. Usually, a new process is selected under the following circumstances:

- When a process must wait for an event.
- When an event occurs (e.g., I/O completed, quantum expired).
- When a process terminates.

Medium-term scheduling
Medium-term scheduling involves suspending or resuming processes by swapping (rolling) them out of or into memory

Long-term scheduling
Long term scheduling is done when a new process is created. It initiates processes and so controls the degree of multi-programming (number of processes in memory).
Simple policies for long term scheduling are

- Simple *First Come First Served* (FCFS): it's essentially a FIFO scheme. All job requests (e.g. a submission of a batch program, or an user trying to log in in a time shared system) are honoured up to a fixed system load limit, further requests being refused tout court, or enqueued for later processing.
- Priority schemes. Note that in the context of long term scheduling ``priority'' has a different meaning than in dispatching: here it affects the choice of a program to be entered the system as a process, there the choice of which ready process process should be executed.

Digital control interfacing - process inputs, outputs interface

An interface is the common point at which two or more systems communicate with each other.

Digital I/O interfaces are commonly used in PC based DAQ systems to provide monitoring and control for industrial processes, generate patterns for testing in the laboratory and communicate with peripheral equipment such as data loggers and printers which have parallel digital I/O capabilities. It is clear that these types of digital, or discrete, inputs and outputs (I/O) are much easier for microprocessor-based data acquisition systems to deal with than analog signals as the computer has a fully binary environment. Similar to analog-to-digital converters used for analog I/O, digital I/O is designed to deal directly with Transistor-to-Transistor Logic (TTL) level voltage changes. TTL typically sets the low-voltage level between 0 and 0.8 V and the high-voltage level between 2.0 and 5.0 V. Voltage levels between 0.8 and 2.0 V are not allowed. A voltage change, then, from the high range to the low range (or vice versa) represents a digital change of state from high to low, on to off, etc. and because acquiring an analog signal is more complex than acquiring a digital one, analog I/O channels also are more expensive.

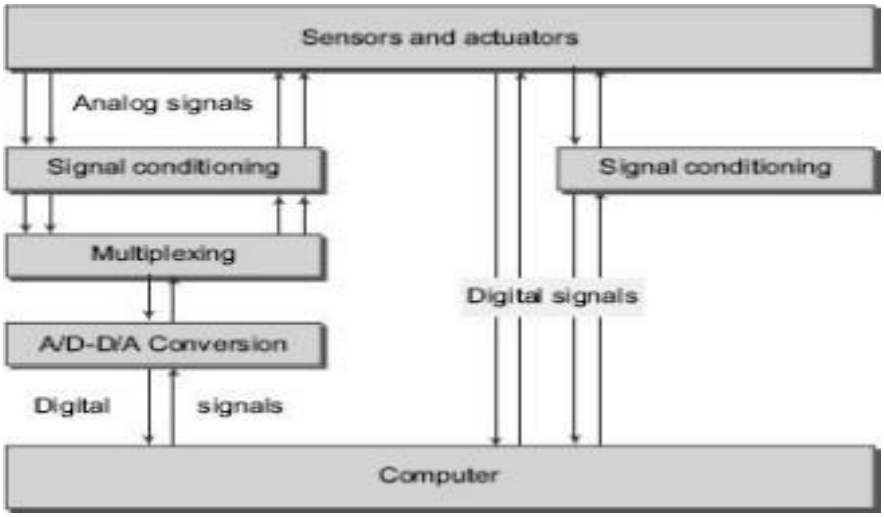


Fig .5.10: Digital I/O cards

Digital Inputs

Many types of digital input signals from switch closures relay contacts, or TTL compatible interfaces can be read directly by digital I/O cards (Figure 1.5). Other types of inputs may require some signal-conditioning, most likely to reduce higher level voltage changes to TTL levels. A variety of signal-conditioning modules are available to provide isolation and other digital-conditioning functions. The most common type of digital input is the contact closure (Figure 1.5.1). Essentially, a sensor or switch of some type closes or opens a set of contacts in accordance with some process change. An applied electrical signal then determines whether the circuit is open or closed. Current flows if the circuit is closed, registering a ‖1‖ in a transistor at

the computer interface. Conversely, an open circuit retains a high voltage (and no current), registering a ‖0‖ at the transistor.

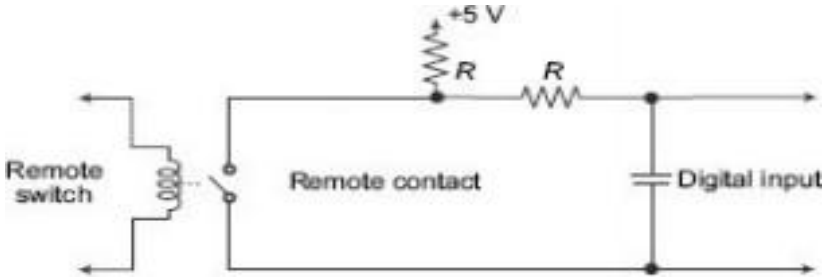


Fig .5.11: Contact type digital input

Digital Outputs

At its simplest, a digital output provides a means of turning something on or off. Applications range from driving a relay to turning on an indicator lamp to transmitting data to another computer. For latching outputs, a –1‖ typically causes the associated switch or relay to latch, while a –0‖ causes the switch to unlatch. Devices can be turned on or off, depending on whether the external contacts are normally open or normally closed. Standard TTL level signals can be used to drive 5 V relay coils; a protective diode is used to protect the digital output circuitry (Figure 1.4.3). Because data acquisition boards can typically supply only 24 mA of driving current, they are intended primarily to drive other logic circuits, not final control elements. Scaling may be needed so that logical voltage levels are sufficient to cause switching in larger relays. Outputs intended to drive larger solenoids, contactors, motors, or alarms also may require a boost.

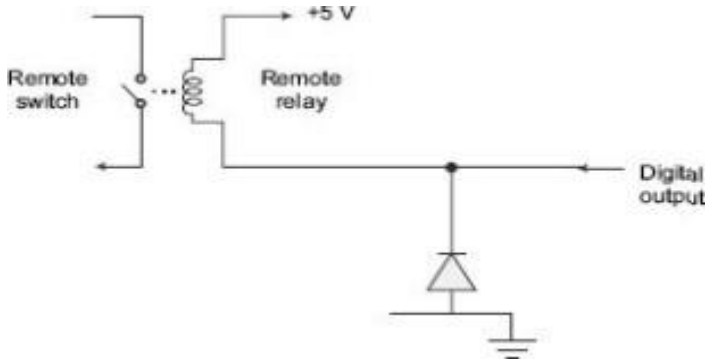


Fig .5.12: Contact type digital input

Counter/Timer and Pulse I/O

A somewhat separate class of digital I/O is pulse inputs and outputs, which typically is associated with frequency, counting or totalisation applications. Pulse inputs might be used to count the rotations of a turbine flow meter; pulse outputs might be used to drive a stepping motor. Pulse inputs are handled in much the same way as digital logic inputs, but the output of the sensing circuit is normally connected to a counter rather than a specific bit position in the input register. Successive pulses increment or decrement the counter. Add an elapsed time measure and a frequency or pulse rate can readily be determined. Similar to an analog-to-digital converter, a counter is characterized by its number of bits—an N-bit counter can accumulate up to 2N discrete events. Thus, a 16-bit counter can count to 2 16 = 65,536.

5.11 Data Logger

Data loggers are is stand-alone devices that can record information electronically from internal or external sensors or other equipment that provide digital or serial outputs.

Features

(a) **Stand-alone Operation**: Most data loggers are normally configured with a PC, some models can be configured from the front panel provided by the manufacturer. Once the data loggers are configured, they don‘t need the PC to operate.

(b) **Support for Multiple Sensor Types**: Data loggers often have universal input type which can accept input from common sensors like thermocouple, RTD, humidity, voltage, etc.

(c) **Local Data Storage**: All data loggers have local data storage or internal memory unit, so all the measured data is stored within the logger for later transfer to a PC.

(d) **Automatic Data Collection**: Data loggers are designed to collect data at regular intervals, 24 hours a day and 365 days a year if necessary, and the collection mode is often configurable.

Data logging and recording are both analog terms in the field of measurement. Data logging is basically measuring and recording of any physical phenomena or electrical parameter over a period of time. The physical phenomena can be temperature, strain, displacement, flow, pressure, voltage, current, resistance, power, and many other parameters

The data logger collects information about the state of any physical system from the sensors. Then the data logger converts this signal into a digital form with the help of an A/D converter. This digital signal is then stored in some electronic storage unit, which can be easily transferred to the computer for further the analysis.

5.11.1 Components of data logger

1. Hardware components like sensors signal conditioning, and analog-to-digital converter, etc.
2. Long-term data storage, typically onboard memory or a PC
3. Software for collecting data, analyzing and viewing

5.11.2 Industrial Data logging and display

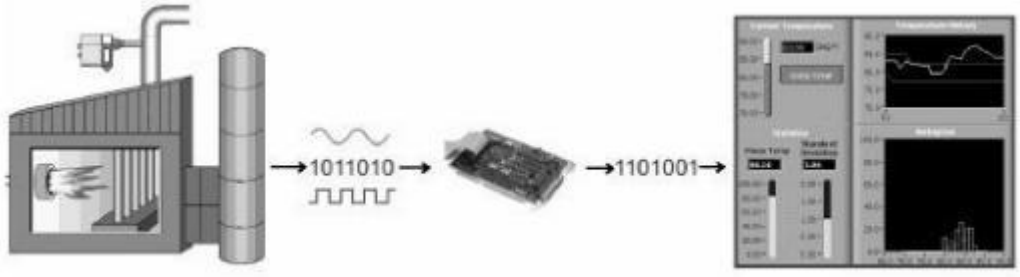


Fig.5.13: Industrial Data logging and display

5.12 Data acquisition system

A data acquisition system is a device or an integrated system used to collect information about the state or condition of various parameters of any process. For example, collecting day-to-day temperature of a particular location can be termed data acquisition. In simple words DAQ is defined as "Data acquisition is the process by which physical phenomena from the real world are transformed into electrical signals that are measured and converted into a digital format for processing, analyzing, and storage by a computer".

5.12.1 Basic components of data acquisition systems

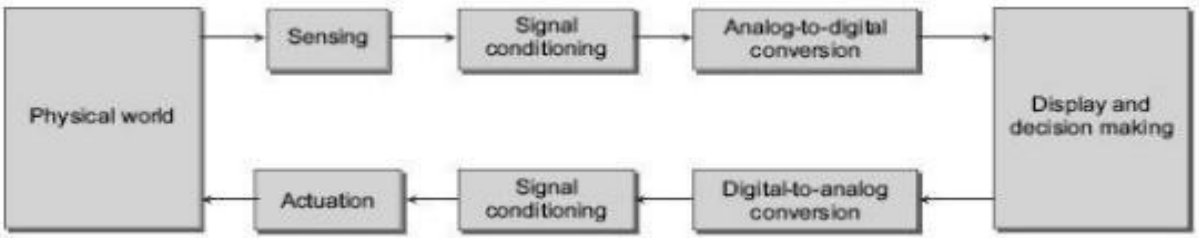


Fig. 5.14: Basic components of data acquisition systems

- Sensors and transducers
- Field wiring
- Signal conditioning
- Data acquisition hardware
- PC (operating system)
- Data acquisition software

Sensors and transducers

Sensors and Transducers Transducer/Sensor converts input energy from one form to another form. According to the type, output sensors are classified in two types: digital sensors and analog sensors. The sensors which can produce a digital output signal, that is a digital representation of the input signal, having discrete values of magnitude measured at discrete times, are called digital sensors. A digital sensor must output logic levels that are compatible with the digital receiver. Examples of digital sensors include switches and position encoders. Analog sensors produce an output signal that is directly proportional to the input signal, and is continuous in both magnitude and in time. Most physical variables such as temperature, pressure and acceleration are continuous in nature and are readily measured with an analog sensor.

Signal Conditioning

Most sensors and transducers generate signals that must be conditioned before a measurement or DAQ device can reliably and accurately acquire the signal. This front-end processing is referred to as signal conditioning. A signal conditioner may create excitation for certain transducers such

as strain gauges and resistance temperature detectors, which require external excitation voltages or currents. The main tasks performed by signal conditioning are as follows:
Filtering Amplification Linearisation Isolation Excitation

PC

The PC used in a data acquisition system can greatly affect the speeds at which data can be continuously and accurately acquired, processed, and stored for a particular application. Where high-speed data acquisition is performed with a plug-in expansion board, the throughput provided by bus architectures, such as the PCI expansion bus, is higher than that delivered by the standard ISA or EISA expansion bus of the PC.
The particular application, the microprocessor speed, hard-disk access time, disk capacity and the types of data transfer available, can all have an impact on the speed at which the computer is able to continuously acquire data. All PCs, for example, are capable of programmed I/O and interrupt-driven data transfers. The use of Direct Memory Access (DMA), in which dedicated hardware is used to transfer data directly into the computer‘s memory, greatly increases the system throughput and leaves the computer‘s microprocessor free for other tasks. In normal operation, the data acquired, from a plug-in data acquisition board or other DAQ hardware (e.g. data logger), is stored directly to system memory. Where the available system memory exceeds the amount of data to be acquired, data can be transferred to permanent storage, such as a hard disk, at any time. The speed at which the data is transferred to permanent storage does not affect the overall throughput of the data acquisition system.

A/D Conversion

A process of converting an analog signal into a digital signal comprises measuring the amplitude of the analog signal at consistent time intervals and producing a set of signals representing the measured digital value. The information in the digital signals and the known time interval enables one to convert the digital signal back to the analog signal. Analog to digital conversion of a continuous input signal normally occurs in two steps: sampling and quantisation. The sampler takes a time-varying analog input signal and converts it to a fixed voltage, current, electrical charge, or other output level. The quantiser takes the constant sampled level and compares it to the closest level from a discrete range of values called quantisation levels.

D/A Conversion

A digital-to-analog converter, or simply DAC, is a semiconductor device that is used to convert a digital code into an analog signal. Digital-to-analog conversion is the primary means by which digital equipment such as computer-based systems are able to translate digital data into real-world signals that are more understandable to or useable by humans, such as music, speech, pictures, video, and the like.

Display devices

After collecting information about the state of some process, the next consideration is how to present it in a form where it can be readily used and analysed. standards of good practice for

presenting data in either graphical or tabular form are covered, using either paper or a computer monitor screen as the display medium.
Example : Recorders, CRT

Advantages

- Reduced data redundancy
- Reduced updating errors and increased consistency
- Greater data integrity and independence from applications programs
- Improved data access to users through use of host and query languages
- Improved data security
- Reduced data entry, storage, and retrieval costs
- Facilitated development of new applications program

Disadvantages

- Database systems are complex, difficult, and time-consuming to design
- Substantial hardware and software start-up costs
- Damage to database affects virtually all applications programs
- Extensive conversion costs in moving form a file-based system to a database system
- Initial training required for all programmers and users

5.13 Computer control action- treatment of inputs, outputs, and control strategies

One of the primary requirements for a computer is that it is able to take data in, and produce output that contains data to be interpreted. I/O (Input/output) refers to the processes by which the CPU communicates with devices that handle the input and output.

Interaction of Computer and I/O Devices

The basic problem in the management of I/O devices is most easily seen by comparion to the memory system. With the memory system all timings are known. All memory transcations are controlled by a clock signal on bus. One can design a control system based on assumptions that a memory operation will complete within a fixed time.

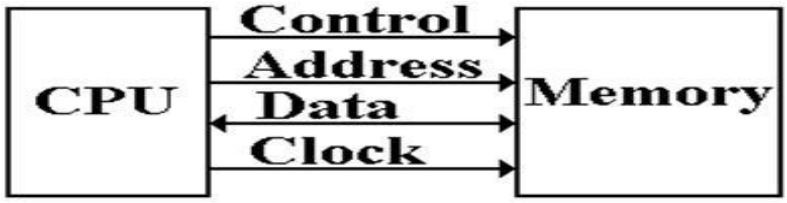


Fig .5.15 : Interaction of Computer and I/O Devices

Timing of I/O operations presents some challenges to the system designer. For many I/O devices, the time to complete the data transfer either cannot be estimated or varies quite a bit. We need protocols for interfacing these I/O devices to the CPU.

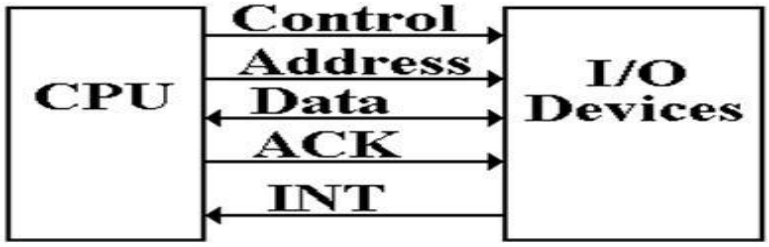


Fig .5.16 : Interaction of Computer and I/O Devices with acknowledgement

5.14 The Four Strategies

Here are the simple definitions of the four I/O strategies.

Program Controlled I/O
This is the simplest to implement. The executing program manages every aspect of I/O processing. I/O occurs only when the program calls for it. If the I/O device is not ready to perform its function, the CPU waits for it to be ready; this is **"busy waiting"**.

The next two strategies are built upon program controlled I/O.

Interrupt Driven I/O
In this variant, the I/O device can raise a signal called an **"interrupt"** when it is ready to perform input or output. The CPU performs the I/O only when the device is ready for it.

In some cases, this interrupt can be viewed as an alarm, indicating an undesirable event.

Direct Memory Access
This variant elaborates on the two above. The I/O device interrupts and is sent a –word count‖ and starting address by the CPU. The transfer takes place as a block.

I/O Channel
This assigns I/O to a separate processor, which uses one of the above three strategies.

UNIT- V Part- A

Sl.no	Part A - Questions	CO	Level
1	Differentiate conventional and computer controllers.	CO2	L4
2	List out the basic components of Data Acquisition System?	CO2	L1
3	Define the term DDC	CO2	L1
4	Express the term SCADA	CO2	L2
5	Define data logger	CO2	L1
6	Point out some advantages and disadvantages of SCADA	CO2	L4
7	List out the algorithms used in computer controlled system	CO4	L1
8	Sketch the block diagram of computer control system	CO2	L3
9	State about scheduling Control?	CO2	L1
10	List out the benefits of direct digital control	CO2	L1
11	Discuss the meaning of remote terminal units	CO2	L2
12	Name the different components in SCADA	CO2	L1
13	Predict the types of SCADA system	CO2	L2
14	List out the features of data logger	CO2	L1

UNIT- V Part – B

Sl.no	Part B - Questions	CO	Level
1	Explain in detail about direct digital control with neat block diagram	CO2	L2
2	Discuss in detail about SCADA with neat block diagram	CO2	L2
3	Discuss in detail about the basic components of data acquisition systems	CO2	L2
4	Express the type of SCADA system and explain in detail	CO2	L2
5	Explain about Data loggers with an example	CO2	L2
6	Explain and derive the Velocity and position algorithm	CO4	L2