PROGRAMING LOGIC CONTROL SYSTEMS

Near East University
Faculty of Engineering
Electrical & Electronics Dept.

Günay GÜNAY

CONTENTS

CHAPTER 3

CHAPTER 4

INTRODUCTION

In the world of automation, the programmable logic controller (PLC) has become a standart for control It now not only replaces the earlier relay controls but has taken over many additional control functions. All basic instructions are discussed, and conversion from relay ladder diagrams to logic ladder diagrams is emphasized.

The need for low-cost, versatile and easily commissioned controllers has resulted in the development of programmable-control systems - standard units based on a hardware CPU and memory for the control of machines or processes. Originally designed as a replacement for the hard-wired relay and timer logic to be found in traditional control panels, PLCs provide ease and flexibility of control based en programming and executing simple logic instructions (often in ladder diagram form). PLCs have internal functions such as timers, counters and shift registers, making sophisticated control possible using even the smallest PLC.

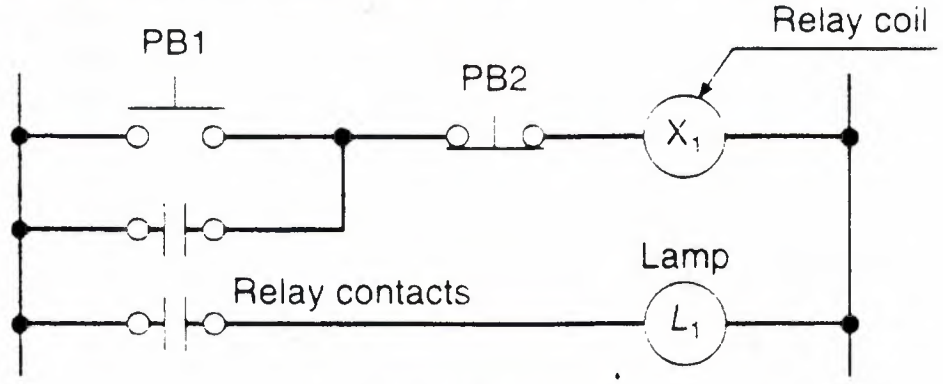
A programmable controller operates by examining the imput signals from a process and carrying out logic instructions (which have been programmed into its memory) on these input signals, producing ouput signals to drive process equipment or machinery. Standart interfaces built in to PLCs allow them to be directly connected to process actuators and transducers (e.g. pumps and valves) without the need for intermediate circuitry or relays.

1) OVERVIEW

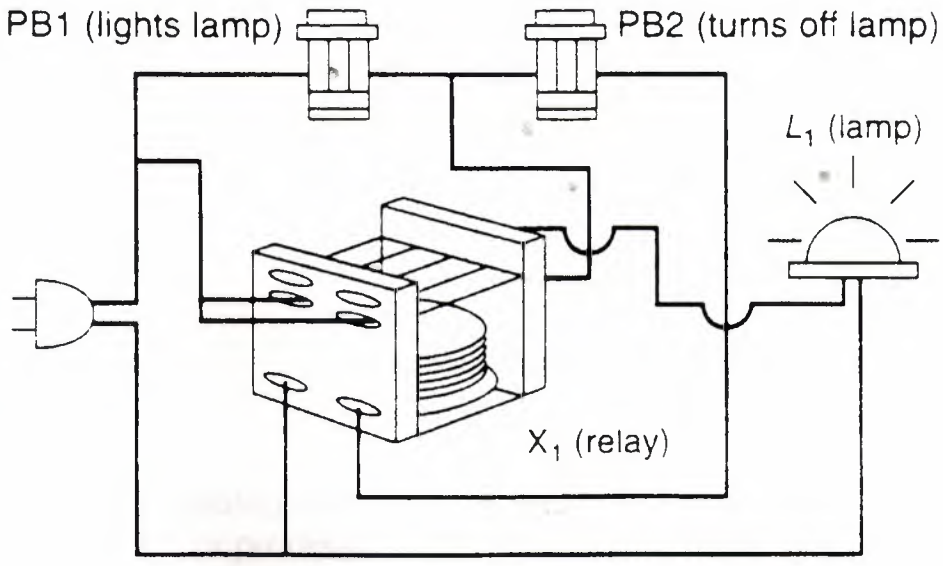
A programmable controller is a computer designed for use in machines. Unlike a computer, it has been designed to operate in the industrial environment and is equipped with special inputs/outputs and a control programming language. The common abbreviation used in industry for these devices, PC, can be confusing because it is also the abbreviation for personal computer. Therefore, some manufacturers refer to their programmable controller as a PLC, which is an abbreviation for programmable logic controller.

Initially the PLC was used to replace relay logic, but its ever-increasing range of functions means that it is found in many and more complex applications. As the structure of a PLC is based on the same principles as those employed in computer architecture, it is capable of performing not only relay switching tasks, but also other applications such as counting, calculating, comparing, and the processing of analog signals

Programmable controllers offer several advantages over a conventional relay type of control. Relays have to be hard-wired to perform a specific function (figure 1)



(*a*) Schematic diagram



(*b*) Wiring diagram

Fig.1
Hard-wired relay type of control

When the system requirements change, the relay wiring has to be changed or modified, which requires time. In extreme cases, such as in the auto industry, complete control panels had to be replaced since it was not ecoomically feasible to rewire the old panels with each model changeover. The programmable controller has eliminated much of the hand wiring associated with conventional relays control circuits. It is small and inexpensive compared to equivalent relay-based process control systems. Programmable controlers also offer solid-state reliability, lower consumption, and ease of expandability. If an application has more than a half-dozen relays, it probably will be less expensive to install a PLC. Simulating a hundred relays, timers, and counters is not a problem even on small PLCs.

A personal computer can be made into a programmable controller if you provide some way for the computer to receive information form devices such a pushbuttons or switches. You also need a program to process the inputs and decide the means of turning OFF and ON load devices. A typical PLC can be divided into three parts, as illustrated in the block diagram of Fig. 2 These three components are the central processing unit (CPU), the input/output (1/0) section, and the programming device. The programmable controller is an event-driven device, which means that an event taking place in the field will result in an operation or output taking plce.

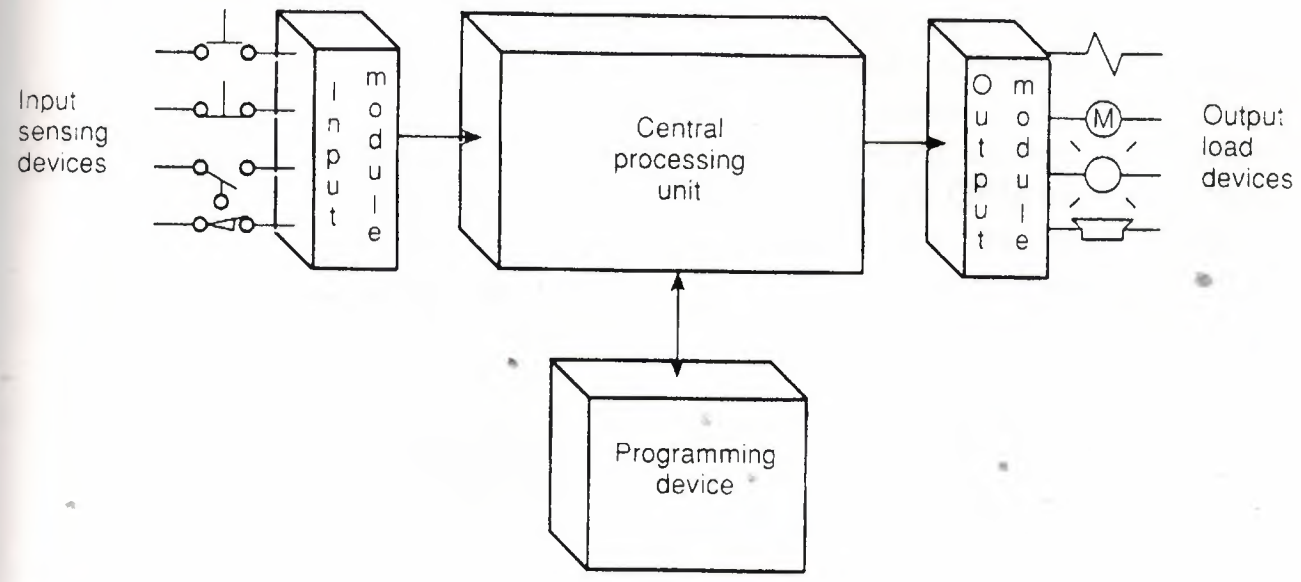


Fig. 2
PLC block diagram

The central processing unit (CPU) is the heart of the PLC system. A typical central processing unit or processor is shown in Fig. 3. The CPU is a microprocessor-based system that replaces control relays, counters, timers, and sequencers. A processor appears only once in a PLC, and it can be either a one-bit or a word

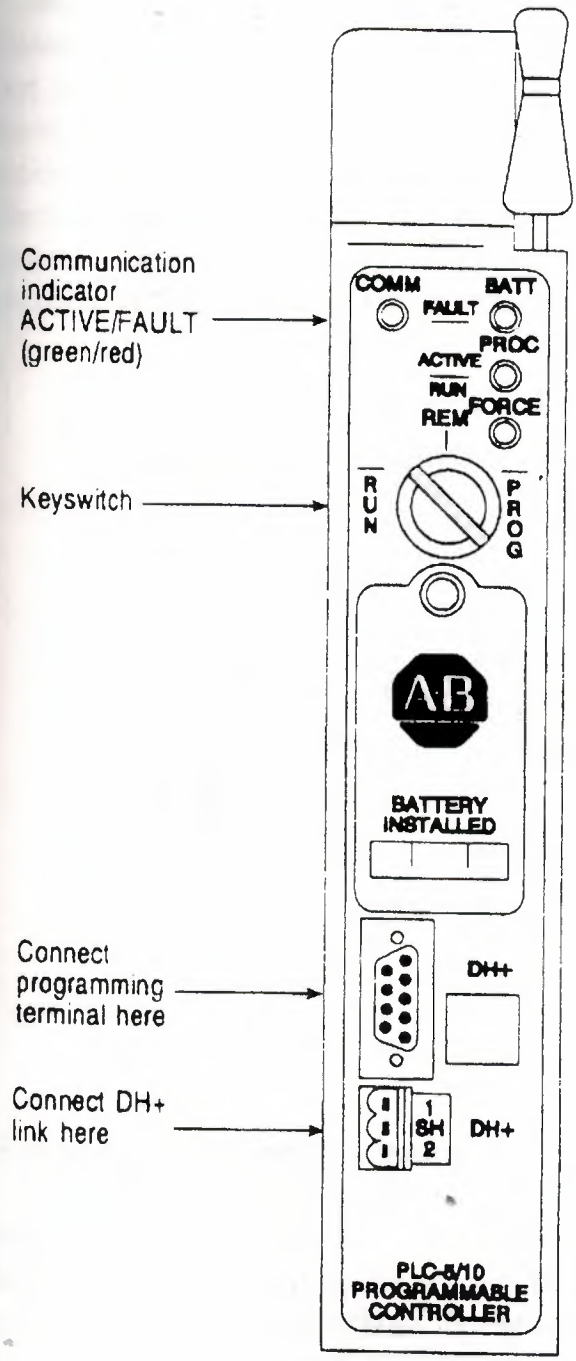


Fig. 3
Typical processor unit.

processors are used when processing text and umerical data, calculations, gauging, controlling, and recording, as well as the simple processing of signals in binary code, are required. The principle of operation of a CPU can be briefly described as follows:

The CPU accepts (reads) input data from various sensing devices, executes the stored user program from memory, and sends appropriate output commands to control devices.

A direct current (de) power source is required to produce the low-level voltage used by the processor and the I/: modules. This power supply can be housed in the CPU unitd or may be a separately mounted unit, depending on the PLC system manufacturer.

Most CPUs contain backup batteries that keep the operating program in storage in the event of a plant power failure. Typical retenivebackup time is one month to one year.

The CPU contains various elecktrical parts and receptacles for connecting the cables that go to the other units as well as to operational key switches.

Typical operation key switch positions are:

OFF: System cannot be run or programmed.

Run: Allows the system to run, but no program alterations can be made.

Program: Disables outputs ad allows ereating, modifying, and deleting of programs.

The processor memory module is a major part of the CPU housing (Fig. 4)

Memory is where the control plan or program is held or stored in the controller. The information stored in the memory relates to the way the input and output data should be processed. The complexity of the program determines the amount of memory required. Memory elements store individual pieces of information called bits (for binary digits). he actual control program is held within electronic memory storage components, such as the RAMs and EEPROMs. The processing unit scans data from the input and output modules and stores their conditions in the memory. The processor unit then scans the user program stored in

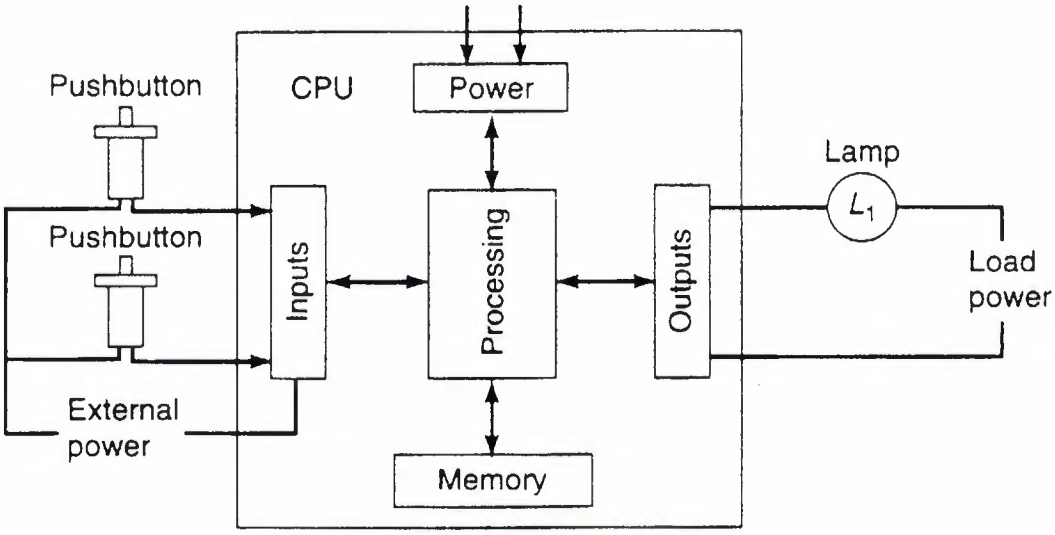


Fig. 4
Processor memory

the memory and makes decisions that cause outputs to change.

Memory can be placed into two categories: volatile and nonvolatile. Volatile memory will lose its stored information if all operating power is lost or removed. Volatile memory is easily altered and quite suitable for most applications when supported by battery backup. Nonvolatile memory can retain stored information when power is removed accidentally or intentionally. PLCs make use of many different types of volatile and nonvolatile memory devices. Following is a generalized description of a few of the more common types:

RAM. Random access memory (RAM) is designed so that information can be jritten into or read from the memory. Today's controllers, for the most parts, use the CMOS-RAM jith battery support for user program memory. RAM provides an excellent means for easily creating and altering a program.

ROM. Read-only memory (ROM) is designed so that information stored in memory can only be read and, under ordinary circumstances, cannot be changed. Infrmation foud in the ROM is placed there by the manufacturer for the internal use

EEPROM. Elecktrically erasable programmable read-only memory (EEPROM) is a nonvolatile memory that offers the same programming flexibility as does RAM. It provides permament storage of the program but can be easily changed using standard programming devices.

The I/O section of a PLC sqconsists of input modules and output modules. THE I/O system forms the interface by which field devices are connected to the controller. The purpose of this interface is to condition the various signals received from or sent to external field devices. Input devices such as pushbuttons, limit switches, sensors, selecktor switches, and thumbwhell switches are hard-wired to terminals on the input modules. Output devices such as small motors, motor starters, solenoid valves, and indicator lights are hard-wired to the terminals on the output modules. These devices are also referred to as "field" or "real-word" inputs and outputs. The terms fields or real-word" are used to distinguish actual external devices that exist and must be physically wired from the internal user program that duplicates the function of relays, timers, and counters. Some programmable controllers have separate modules for inputs and outputs; others have the inputs and outputs connected as an integral part of the controller (Fig.5a) When the module is slid into the rack, it makes an electrical connection with a series of contacts called the backplane, located at the rear of the rack. The PLC processor is also connected to the backplane and can communicate with all the modules in the rack [see Fig. 5 b)

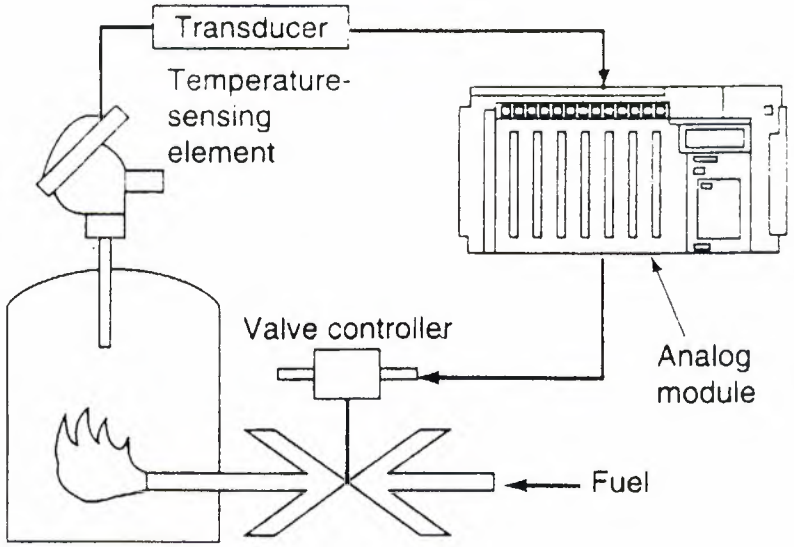
Input interface modules accepts signals from the machine or process devices (e.g., 120 V ac) and convert them into signals (e.g., 5 V de) that can be used by the controller. Output interface modules convert controller signals (e.g., 5 V de) into external signals (e.g., 120 V ac) used to control the machine or process. There are many types of inputs and outputs that can be connected to a programmable controller, and they can all be divided into two groups: digital (alsa known as discrete) and analog.

That digital inputs and outputs are those that operate because of changes in a discrete state or level. Analog inputs and outputs change continuously over a variable range.

The most common type of I/O interface module is the discrete type. This type of interface connects field input devices of the ON/OFF nature such as selector switches, pushbuttons, and limit switches. Liikewise, output control is limited to devices such as lights, small motors, solenoids, and motor starters that require simple ON/OFF switching. Analog inputs and outputs are used in more complex control applications such as furnace temperature control (Fig. 6

Each I/O module is powered by some fieldsupplied voltage source. (Fig. 7)

Since these voltages can be of different magnitudes or types, I/O modules are available at various ac and de voltage and current ratings. Both voltage and current must match the electrical requirements of the system to which it is connected. There are typically 4, 8, 12, 16, or 32 terminals per module. PLC manufacturers have a wide variety of input and out put modules available. The



Furnace temperature control

Fig. 6
Analog I/O module

analog I/O modules provide an interface to a variety of analog signals, including both voltage (e.g., 1- to 5-V) and current (e.g., 4- to 20-mA) ranges.

Signals are connected to the PLC though input modules. Input modules perform four tasks in the PLC control system:

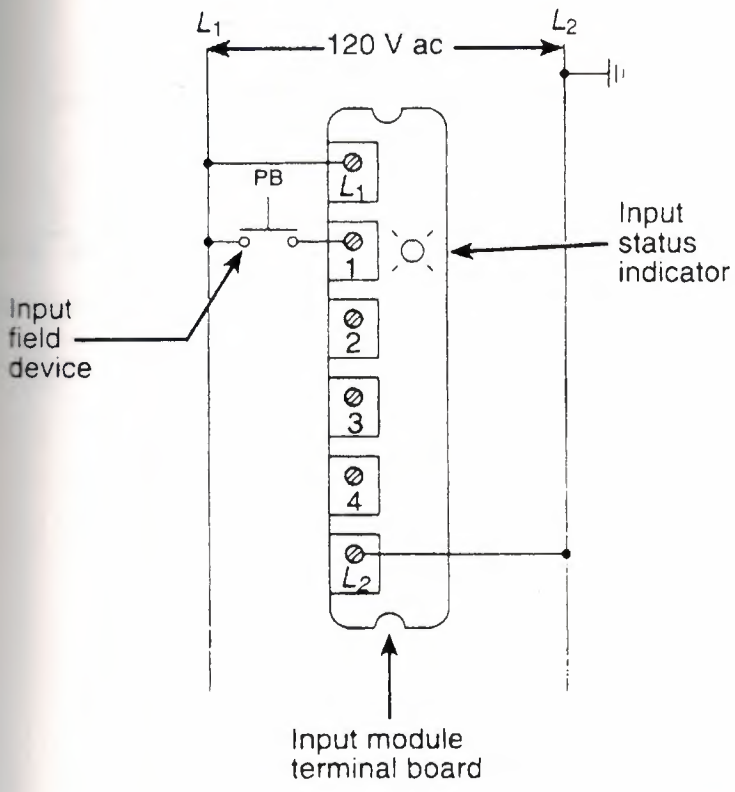
They sense when a signal is received from a sensor on the machine.

They convert the input signal to the correct voltage level for the particular PLC.

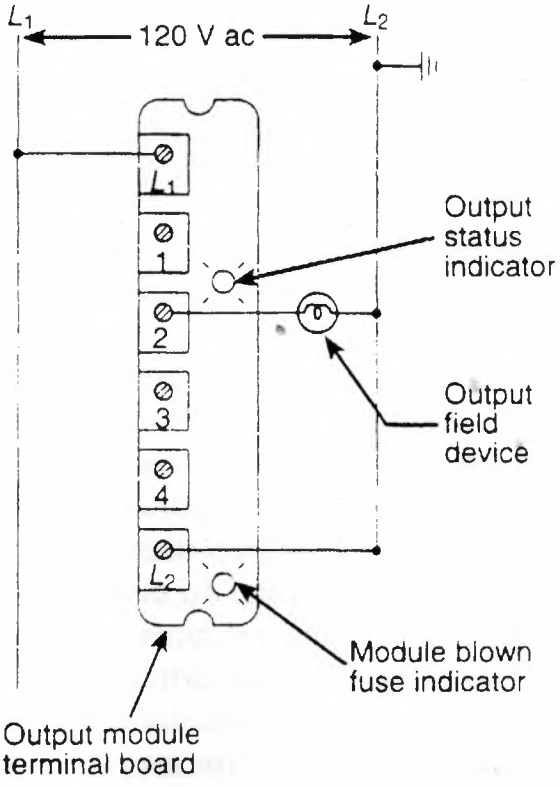
They isolate the PLC from fiuctuations in the input signal's voltage or current.

They send a signal to the PLC indicating which sensor originated the signal.

(fig. 8) shows a simplified block and wiring diagram for one input of a typical ac interface input module. The input circuit is composed of two basic section: the power section and the logic section. The power and logic sections are normally coupled together with a circuit, which elecktrically separates the two. When the pushbutton is closed, 120 V ac is applied to the bridge rectifier through resistors R1 and R2. This produces a low-level de voltage, which is applied across the LED of the optical isolator. the zener diode (ZD) voltage rating sets the minimum level of voltage that can be detected. When light from the LED strikes the phototransistor, it switches into conduction, and the status of the pushbutton is communicated in logis or low-leved de voltage to the processor. The optical isolator not only sperates the higher ac input voltage from the logic circuits but also prevents damage to the processor due to line voltage transients. Optical isolation also helps reduce the effects of elecktrical noise, common in the industrial enviroment, which can cause erratic operation of the processor. Coupling and isolation can also be accomplished by use of a pulse transformer or reed relay.



(*a*) Typical discrete four-point, 120-V ac, input module



(*b*) Typical discrete four-point, 120-V ac, 4-A output module

Fig. 7
I/O module powered by a field-supplied voltage source

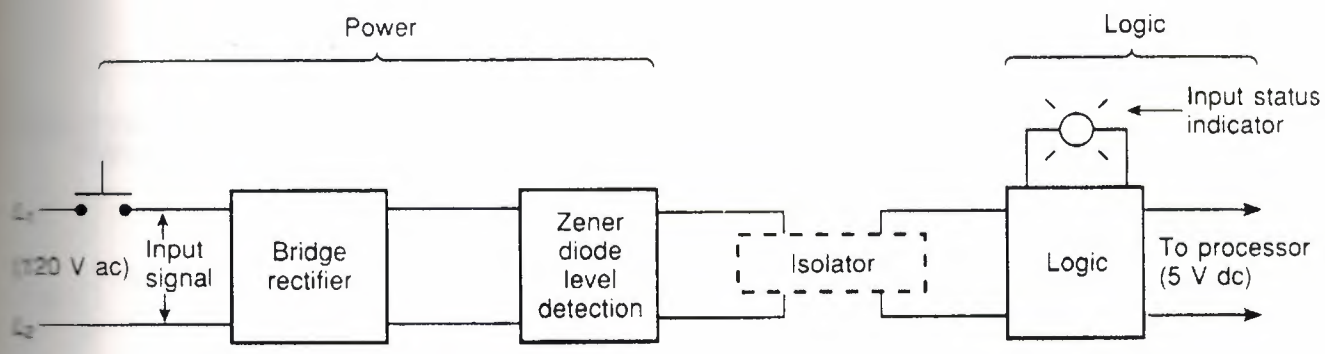
The output interface module of a programmable controller acts as a switch to supply power from the user power supply to operate the output. The output under the control of the program is fed from the processor to a logic circuit that will receive and store the processor command that is required to make an output become active. The output switching devices most often used to switch power to the load in programmable controllers are:

Relay for ac or dc loads

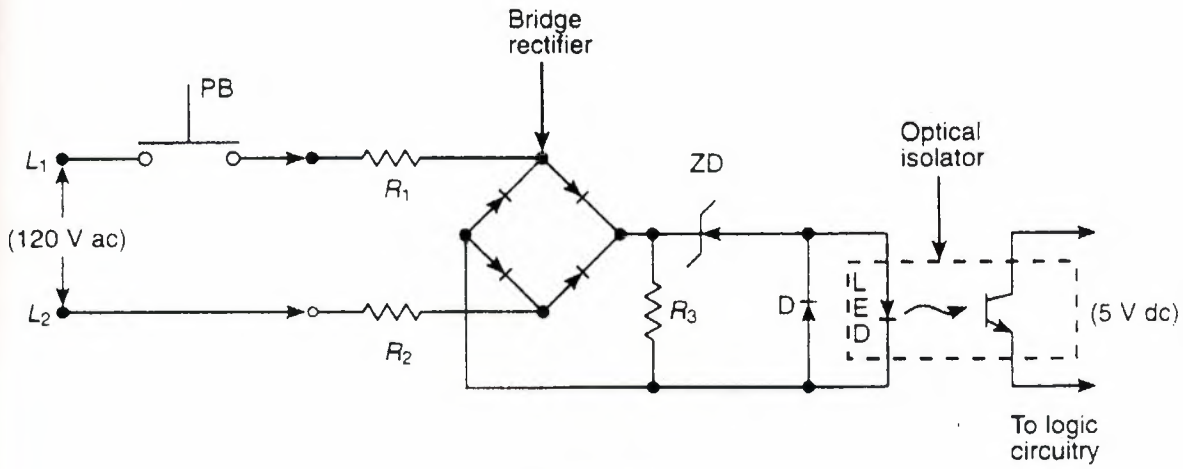
Triac for ac loads only

Transistors for dc loads only

The output module has a function similar to that of the input module except in reverse order. (fig. 9) shows a simplified block and wiring diagram of a typical ac interface output module. As part of its normal operations, the processor sets the output status according to the logic program. When the processor calls for an output, a voltage is applied across the LED of the isolator. The LED then emits light, which switches the phototranssistor into



(*a*) Block diagram



(*b*) Wiring diagram

Fig. 8
Alternating current interface input module

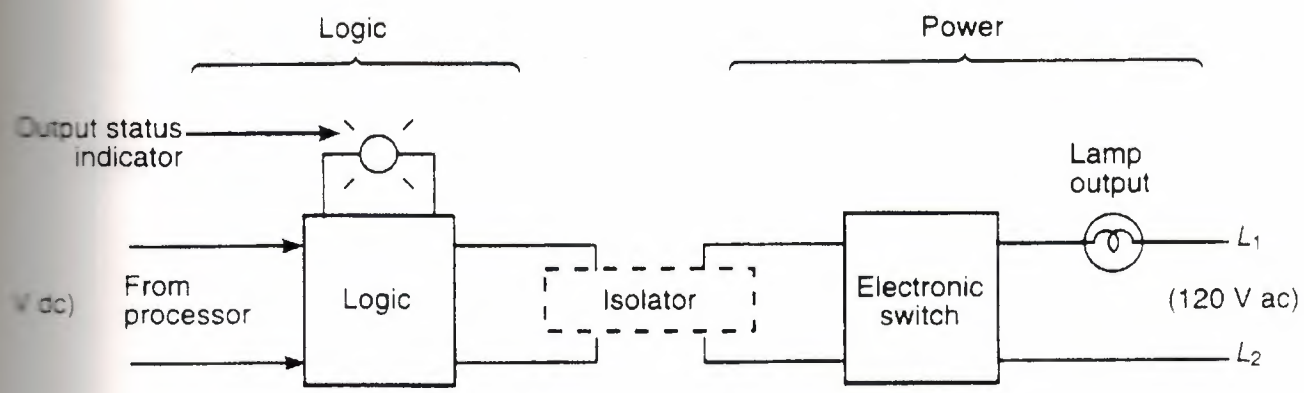
conduction. This in turn. Switches the triac intoconductions, which, in turn, turns on the lamp. Since the triac conducts in either direction, the output to the lamp is alternating current.

That the triac, rather than having ON and OFF status, actually has LOW and HIGH resistance levels, respectively. In its OFF state (HIGH resistance), a small currents leakage of a few milliamperes still flows trough the triac.

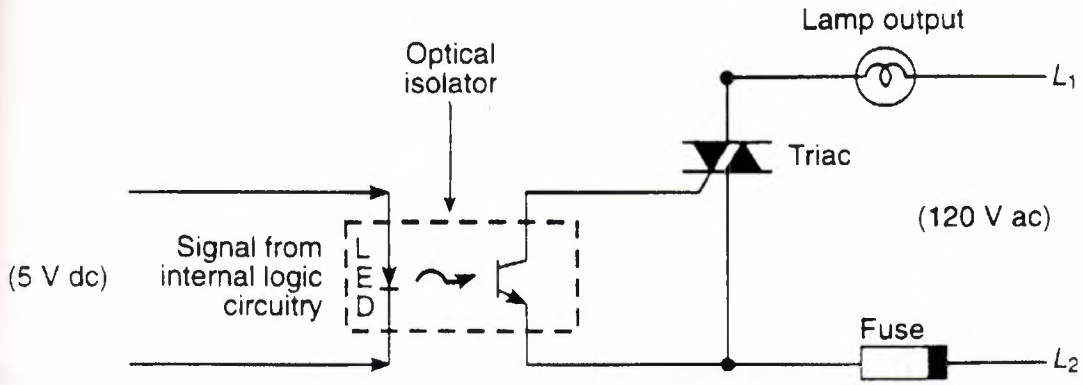
As with input circuits, the output intrface is usually provided with LEDs that indicate the status of each output. In addition, if th module contains a fuse, a fuse status indicator may also be used.

Output interface modules are normally designed to handle currents in the 2- to 3-A range. To protect the output module circuits, specified current ratings should not be exceeded. For controlling larger loads, such as large motor starter as shown in Fig. 10 When a control relay is used in this manner, it is called an interposing relay.

Analog input interface modules contain the circuitry necessary to accept analog voltage or current signals from analog field devices. These inputs are



(*a*) Block diagram



(*b*) Wiring diagram

Fig. 9
Alternating current interface output module

converted form and analog to a digital value by an analog-to-digital (A/D) converter circuit. The conversion value, which is proportional to the analog signal, is expressed as a 12-bit binary or as a three-digit binary-coded decimal (BCD) for use by the processor. Analog input sensing devices include temperature, light, speed, pressure, and position transduers.

The analog output interface module receives digital data from the processor that is converted into a proportional voltage or current to control an analog field device. The digital data is passed through a digital-to-analog (D/A) converter circuit to produce the necessary analog form. Analog output devices include small motors, valves, analog meters, and seven-segment displays.

Each port or terminal on input and output modules is asigned a unique address number This address, is used by the processor to identify the location of the device in order to monitor or control it. The type of the module and the actual physical location of the terminal determines the programming address. The addressing format of inputs and outputs depends on the particular PLC used and

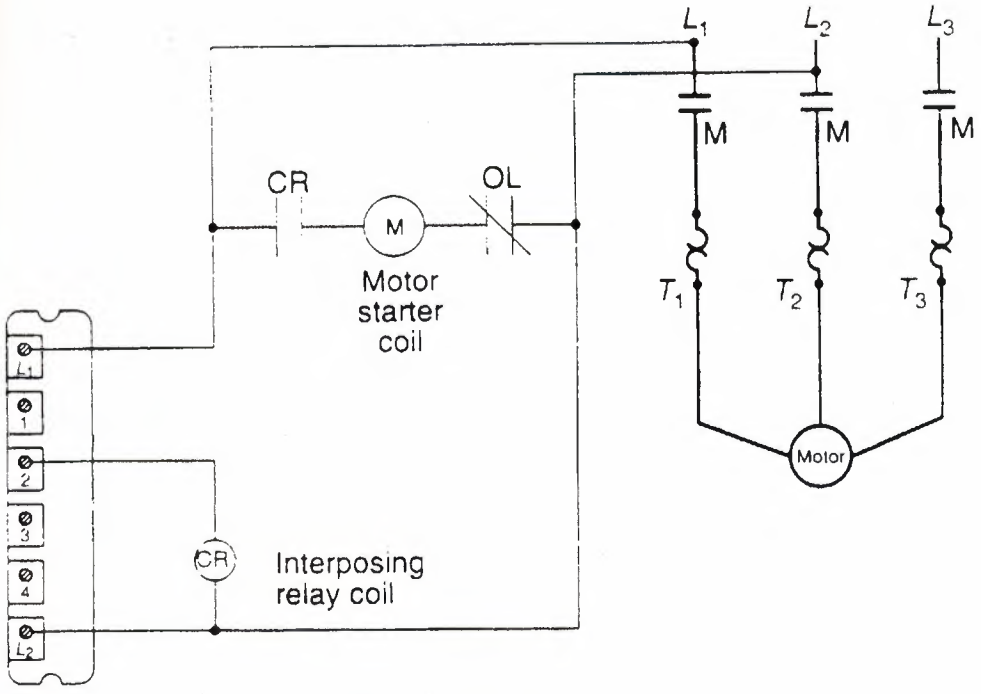


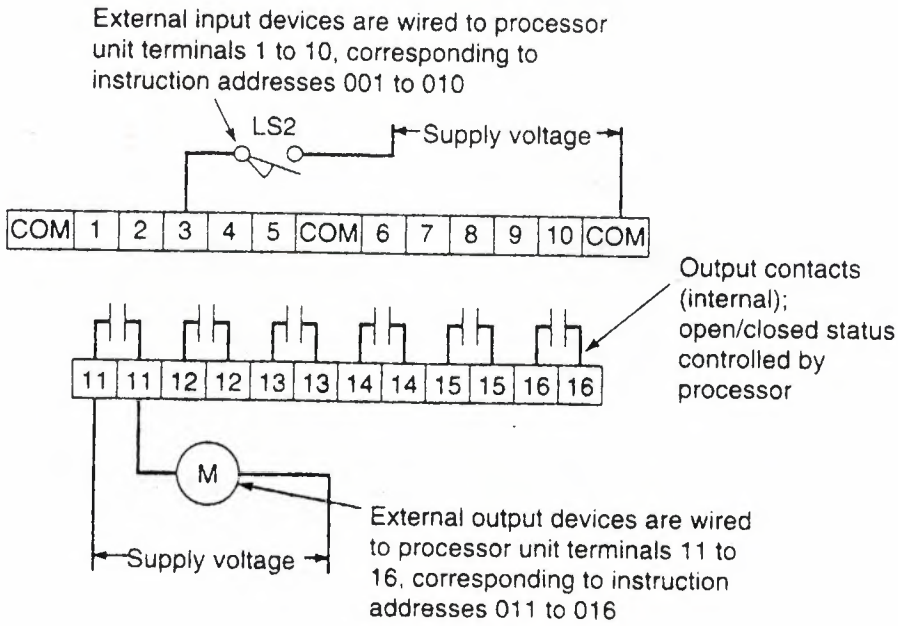
Fig. 10
Interposing relay connection

can normally be found in the particular PLC's user's manual. These addresses can be represented in decimal, octal, or hexadecimal terms depending upon the number system used by the PLC.

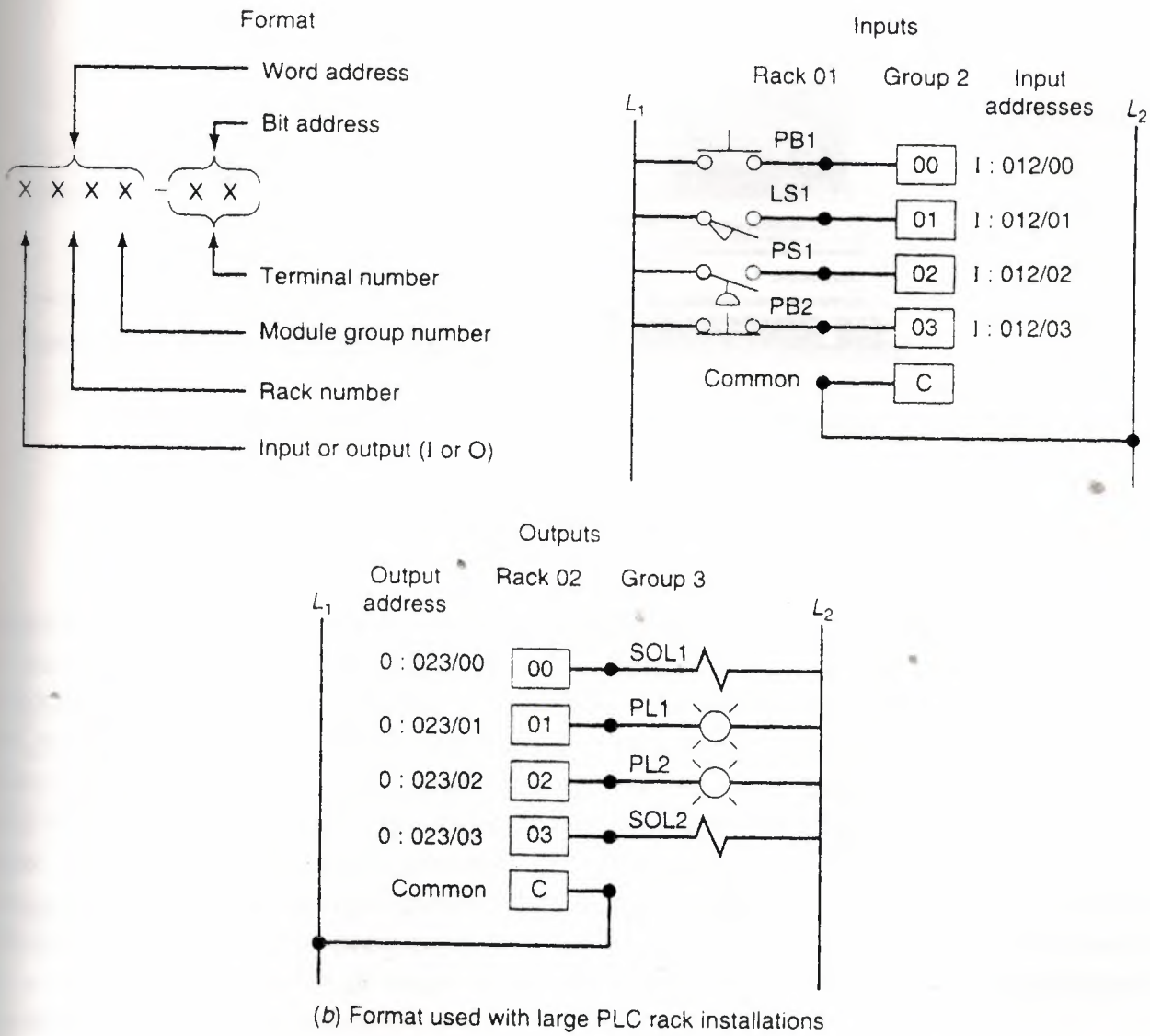
The micro PLC format illutrated in (Fig. 11a) uses a limited number of points of control. Each input and output device must have a specific address. With the large PLC rack installation shown in Fig. (11 b) the loction of a module within a rack and the terminal number of a module to which an input or output device is connected will determine the address of the device.

The programmable controller system requires two power supplies. One provides the power necessary for field devices and output loads to operate and is provided by the programmable controller user. The second power supply is provided internally or as a module that is part of the programmable controller system. This power supply provides internal direct current to operate the processor logic circuitry and I/O assemblies. the voltage that it provides will depend on the type of integrated circuits (ICs) within the system. If the system is made up of transistor-transistor logic (TTL) ICs, the internal power supply will be 5 V, but if the integrated circuit is a complementary metal oxide semiconductor (CMOS) type, the power supply voltage will be in the range of 3 V to 18 V.

The programming device provides the primary means by which the user can communicate with the circuits of the programmable controller It allows the user to enter, edit, and monitor programs by connecting into the processor unit and allowing access to the user memory. the programming unit can be a liquid crystal

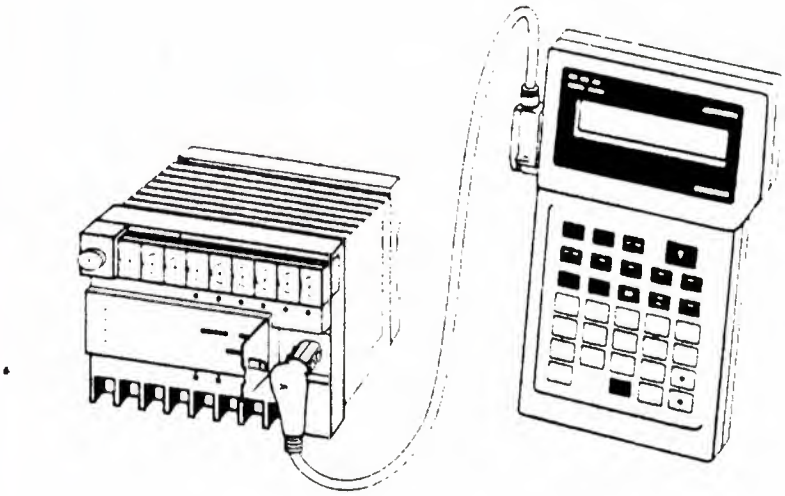


(*a*) Micro PLC format

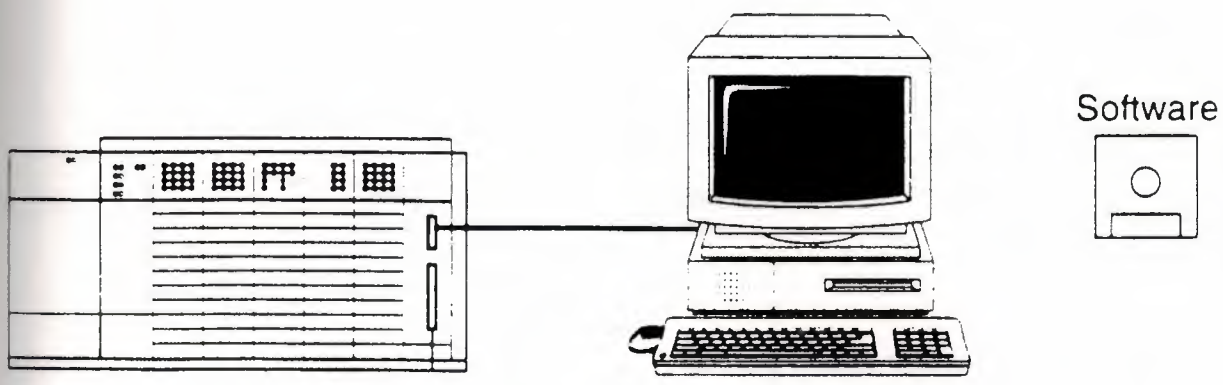


(*b*) Format used with large PLC rack installations

Fig. 11
Typical addressing formats



(*a*) Handheld programming device



(*b*) PLC has one program in memory at a time

Fig. 12
Programming devices

display (LCD) handheld terminal, a single-line LED display unit, or a keyboard and a video display unit. The video display offers the advantage of displaying large amounts of logic on the screen, simplifying the interpretatin of the program. the programming unit communicates with the processor via a serial or parallel data communications link. If the programming unit is not in use, it may be unplugged and removed. Removing the programming unit will not affect the operation of the user program. A personal computer jith appropriate software can also act as a program terminal, making it possible to carry out the programming away from the physical location of the programmable controller. When the program is complete, it is saved to some form of mass storage and downloaded to the programmable controller when required.

A handlend device is shown in (Fig. 12) The display is usually an LED or liquid-crystal display (LCD), and the keyboard consists of numeric keys,

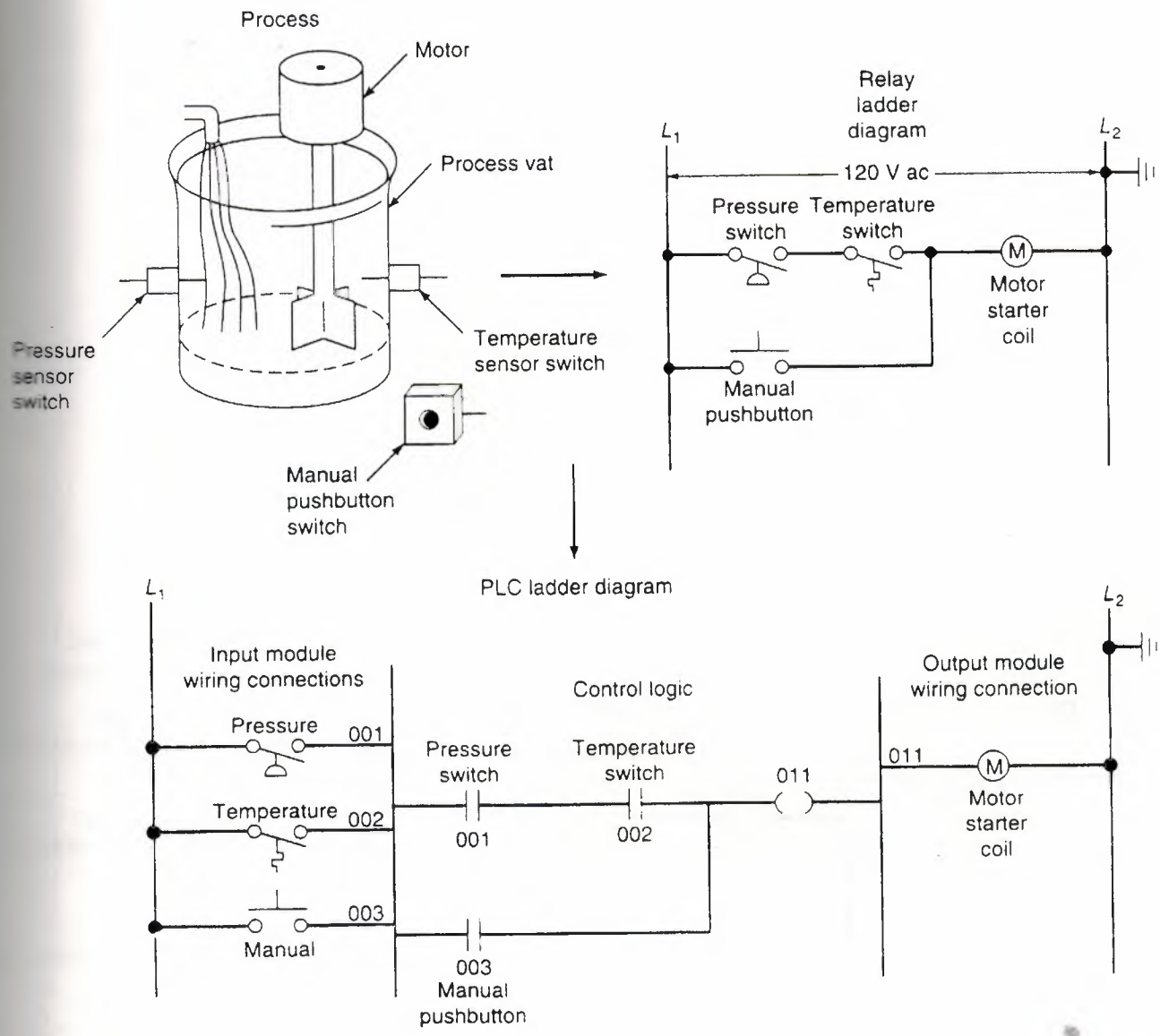
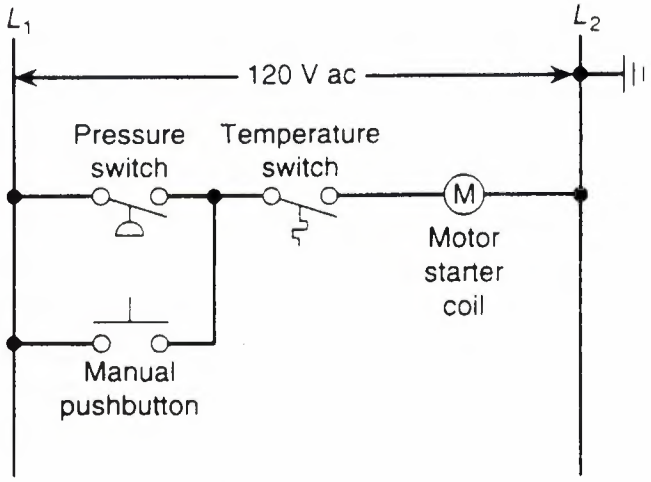


Fig. 13
Process control application

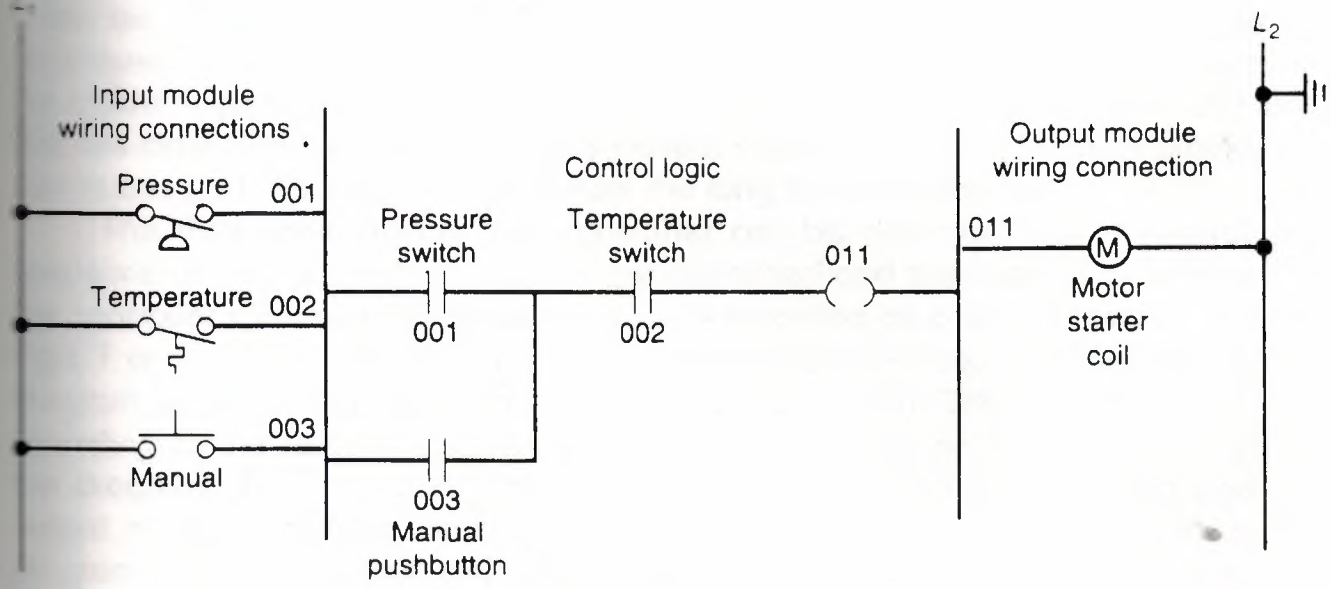
programming instruction keys, and special function keys. To change the program in the PLC, it is necessary either to enter a nej program directly from the keyboard or to download one from the hard disk while online.

When programmable controllers were first introduced, the ladder diagram was selected to be the fundamental programming format because it was well known in the elecktrical and elecktronics industry.

To get an idea of how a PLC operates, consider the simple process control application illustrated in (fig 13) Here a mixer motor is to be used to automatically stir the liquid in a vat when the temperature and pressure reach preset values. In addition, direct manual operation of the motor is provided by means of a separate pushbutton station. the process is monitored with temperature and pressure sensor switches that close their respective contacts when conditions reach their



(*a*) Hard-wired relay circuit requires rewiring for changed application



(*b*) PLC system requires only that control logic be changed; all input and output module connections remain the same

Fig. 14
Modified process control application.

preset values.

This control problem can be solved using the relay method for motor control shown in the relay ladder diagram. The motor starter coil (M) is energized when both the pressure and temperature switches are closed or when the manual pushbutton is pressed.

Now let's look at the way a PLC might be used for this application. The same input field devices (pressure switch, temperature swiitch, and pushbutton) are

used. these devices are hard-wired to an appropriate input module ad dress according to the manufacturer's addressing format. the same output field device (motor starter coil) is also used. this device is hard-wired to an appropriate output module address according to the manufacturer's addressing format.

Next, the PLC ladder control logic diagram is constructed and programmed into the memory of the CPU. A typical ladder logic diagram for this process is shown. The format used is similar to the layout of the hard-wired relay ladder circuit. the individual symbols represent instructions and the numbers represent the instruction addresses. When programming the controller, these instructions are entered one by one into the processor memory from the operator terminal keyboard. Instructions are stored in the user program portion of the processor memory.

To operate the program, the controller is placed in the RUn mode, or operating cycle. During each operating cycle, the controller examines the status of input devices, executes the user program, and changes outputs accordingly. Each -I I- can be thought of as a set of normally open (NO) contacts. The -( )- can be considered to represent a coil that, when energized, will close a set of contacts. In the ladder logic diagram shown, the coil 011 is energized when contacts 001 and 002 are closed or jhen contact oo3 is closed. Either of these conditions provides a continuous path from left to right across the rung that includes the coil.

The RUN operation of the controller can be described by the following sequence of events. First, the inputs are examined and their status is recorded in the controller's memory (a closed contact is recorded as a signal that is called a logic 1 and an open contact by a signal that is called a logic 0). Then tha ladder diagram is evaluated, with each internal contact given OPEN or CLOSED status according to the record. If these contacts provide current path from left to right in the diagram, the out put coil memory location is given a logic 1 value and the output module interface contacts will close. If there is no conducting path on the program rung, the output coil memory location is set to logic 0 and the output module interface contacts will be open. the completion of one cycle of this sequence by the controller is called a scan. the can time, the time required for one full cycle, provides a measure of the speed of response of the PLC.

As mentioned, one of the important features of a PLC is the ease with which the program can be changed. For example, assume that our original process control circuit for the mixing operation must be modified as shown in the relay ladder diagram. the change requires that the manual pushbutton control be permitted to operate at any pressure but not unless the specified temperature setting has been reached.

If a relay system were used, it would require some rewiring of the system to achieve the desired change (Fig. 14 a) However, if a PLC system were used, no rewiring would be necessary. The inputs an outputs are still the same. All that is required is to change the PLC ladder logic diagram as shown (Fig. 14 b)

2) NUMBER SYSTEMS AND CODES

Knowledge of number systems other than the decimal numbering system is quite useful when working with PLCs or with almost any type of digital equipment. This is true because a basic requirement of these devices is to represent, store, and operate on numbers. In general, PLCs work on binary numbers in one form or another; these are used to represent various codes or quantities. Often the programmer needs to be able to perform conversion between the systems, and to perform math functions within each system.

Decimal	Octal	Hexadecimal	Binary
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111
16	20	10	10000
17	21	11	10001
18	22	12	10010
19	23	13	10011
20	24	14	10100

The decimal system, which is most common to us, has a base of 10. The base of a number system determines the total number of different symbols or digits used by that system. For instance, in the decimal system, 10 uniquie numbers or digits - the digits 0 through 9-are used. (Figure 15) shows a comparison between four common numbering systems; decimal (base 10), octal (base 8), hexadecimal (base 16), and binary (base 2). Note that all numbering systems start at zero.

Fig. 15 Number system comparisons.

The value of a decimal number depends on the digits that make up the number and the place value of each digit. A place (weight) value is assigned to each position that a digit would hold from right to left. In the decimal system the first position, starting from the furthest right position, is 0; the second is 1; and so on up to the last position. the weighted value of each position can be expressed as the base (10 in this case) raised to the power of the position. For the decimal system, then, the position weights are 1, 10, 100, 1000, and so on. (Figure 16) on page 340 illustrates the way the value of a decimal number can be calculated by multiplying each digit by the weight of its position and summing the results.

The binary numbering system (base 2) is the basis of all digital systens. Two states exist in digital equipment, an ON states which is representative of one

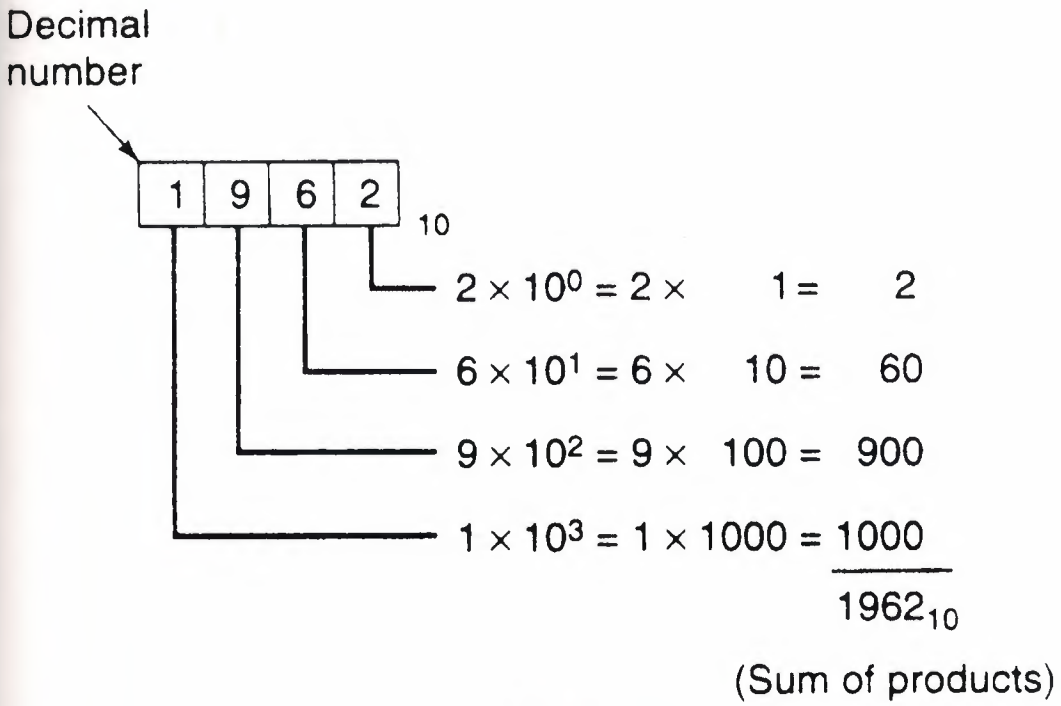


Fig. 16
Weighted value in the decimal system

(1), and an OFF condition, which is representative of zero (0). the ON condition in a circuit is approximately equal to supply voltage, and the OFF to zero volts or ground. A third state may exist in some logic circuits to produce tri-state logic. This condition is a high-impedance or no-voltage state and is not considered in the binary system.

The only allowable digits in a binary numbering system are zero (0) and one (1). Since the binary system uses only two digits, each position of a binary number can go through only two changes, and then a 1 is carried to the immediate left position.

The decimal equitvalent of a binary number is calculated in a manner similar to that used for a decimal number. this time the weighted values of the positions are 1, 2, 3, 8, 16, 32, 64, and so on. Instead of being 10 raised to the power of the position, the weighted value is 2 raised to the power of the position. Figure 17) shows how the binary number 10101101 is converted to its decimal equivalent: 173.

Each digit of a binary number is known as a bit. In a PLC the processor-memory element consists of hundreds or thousands of locations. These locations, or registers, are referred to as words. Each word is capable of storing data in the from of binary digits, or bits. The number of bits that a word can store depends on the type of PLC system used. Eight-bit and sixteen-bit words are hte most common. As the technology continues to develop, 32-bit or larger words will be possible. Bits can also be grouped within a word into bytes. Usually a group of 8 bits is a

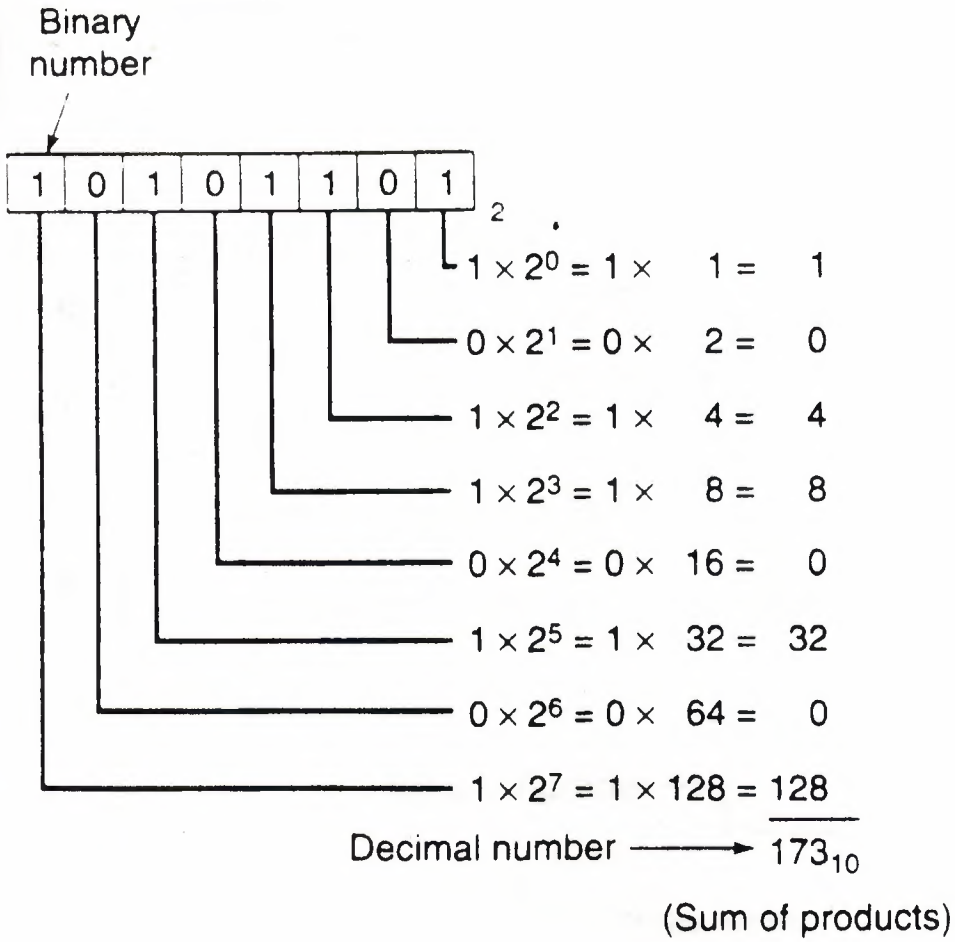


Fig. 17
converting a binary number to a decimal number.

byte, and a group of 1 or more bytes is a word. (Figure 18) illustrates a 16-bit word made up of 2 bytes. The least significant bit (LSB) is the digit that represents the smallest value and the most significant bit (MSB) is the digit that represents the largest value. A bit within the word can exist only in two states: a logical 1 or ON condition or a logical 0 or OFF condition.

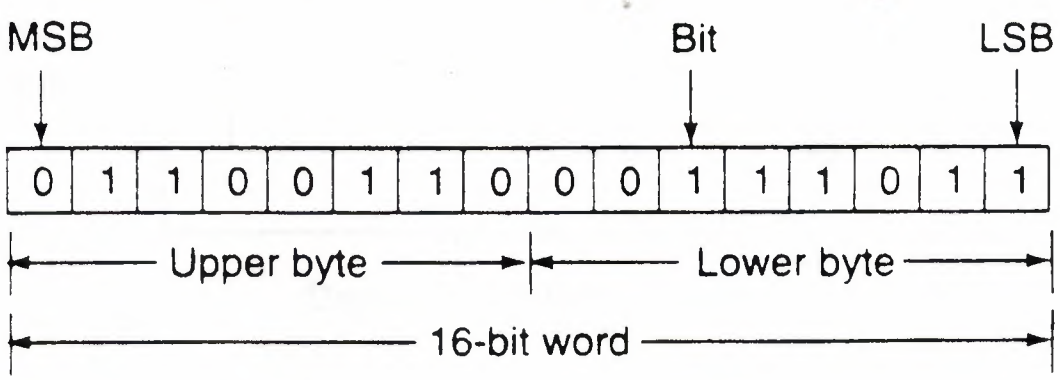


Fig. 18
A 16-bit word

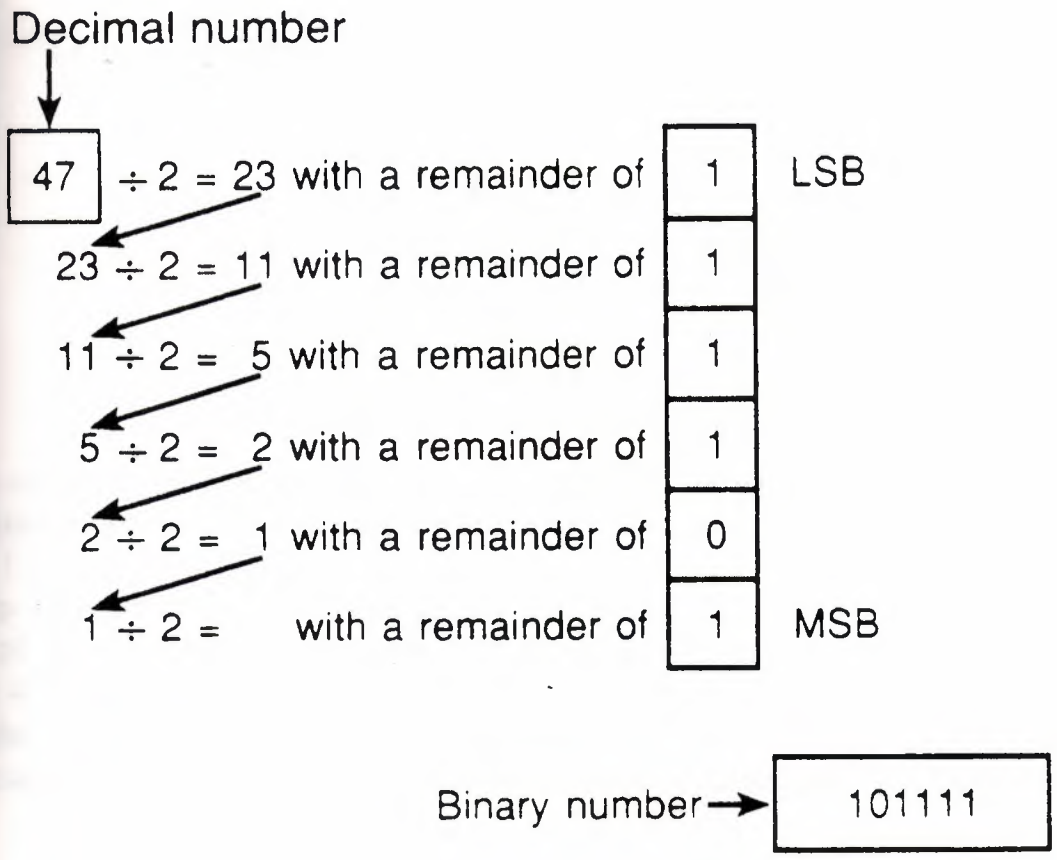


Fig. 19
Converting a decimal number to a binary number.

The size of the programmable controller memory relates to the amount of user program that can be stored. If a memory size is 884 words, then it can actually store 14,144 (884x16) bits of information using 16-bit words or 7072 (884 x 8) using 8-bit words. Therefore, when comparing different PLC systems, you must know the number of bits per word of memory in order tho determine the

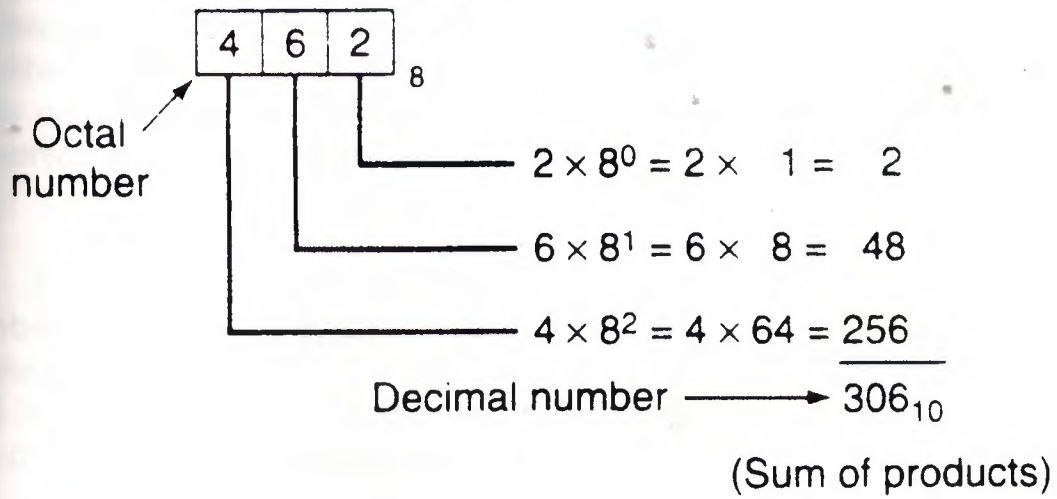


Fig. 20
Converting an octal number to a decimal number

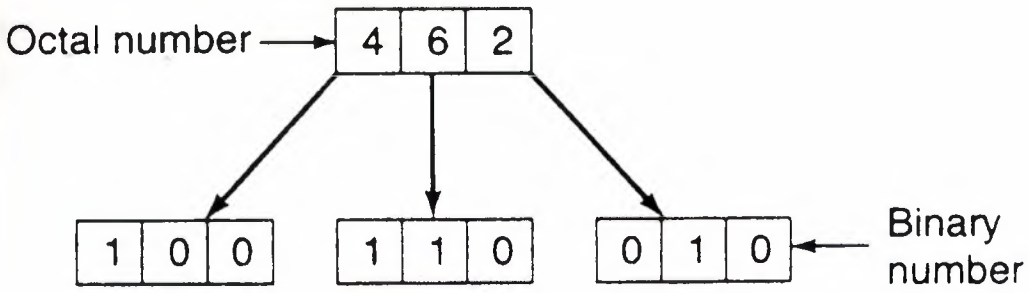


Fig. 21
Converting an octal number to binary number

relative capacity of the systems' memories. Normally programmable controllers do not require storage space above 128 K and in many instances need a memory size of only 1 K to 2 k.

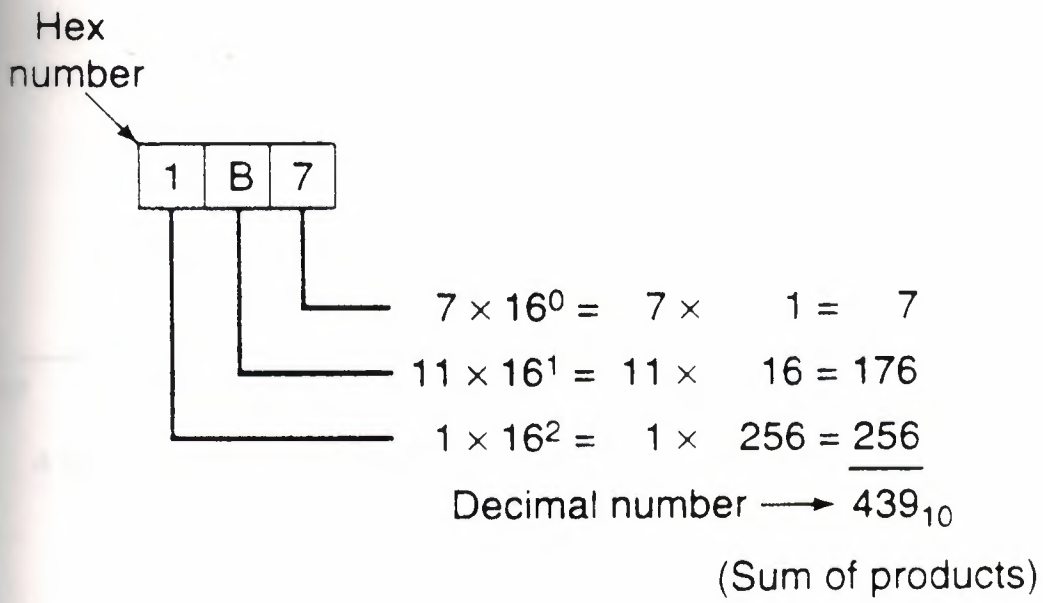
To convert a decimal number to its binary equivalent, we must perform a series of divisions by 2. (Figure 19) illustrates the conversion of the decimal number 47 to binary. We start by dividing the decimal number by 2. If there is no remainder, a 0 is placed in the LSB. The result of the division is brought down, and the process repeated until the result of successive divisions has been reduced to 0.

To express a number in the binary system requires many more digits than in the decimal system. Too many binary digits can become cumbersome to read or write. To solve this problem, other related numbering systems are brought into use.

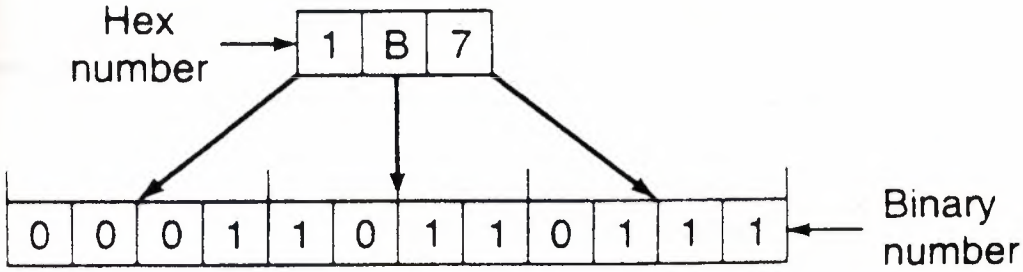
The octal numbering system, a base 8 system, is often used in microprocessor, computer, and programmable controller systems because 8 data bits make up a byte of information which can be addressed by the PLC user or programmer. In some instances, programmable controller manufacturers use the octal system to number wiring terminals, programmable controller racks, and other PLC hardware. The octal number system makes use of 8 digits: 0 through 7. As in all other number systems, each digit in an octal number has a weighted decimal value according to its position. (Figure 20) illustrates how the octal number 462 is converted to its decimal equivalent: 306.

As mentioned, octal is used as a convenient means of handling large binary numbers. For example, the octal number 462 can be converted to its binary equivalent by assembling the 3-bit groups as illustrated in (Fig. 21) Thus, octal 462 is binary 100110010 and decimal 306. Notice the simplicity of the notation. The octal 462 is much easier to read and write than its binary equivalent.

The hexadecimal (hex) number system provides even shorter notation than



(*a*) Converting a hexadecimal number to a decimal number



(*b*) Converting a hexadecimal number to a binary number

Fig. 22
Hexadecimal numbering system

octal. Hexadecimal uses a base of 16. It employs 16 digits: numbers 0 through 9, and letters A through F, with A through F being substituted for numbers 10 through 15, respectively. The techniques for converting hexadecimal to decimal and decimal to hexadecimal are the same as those used for binary and octal (Fig. 22)

The binary coded decimal (BCD) system provides a convenient means to handle large numbers that need to be input to or output from a PLC. The BCD system represents decimal numbers a patterns of Is and Os. This system provides

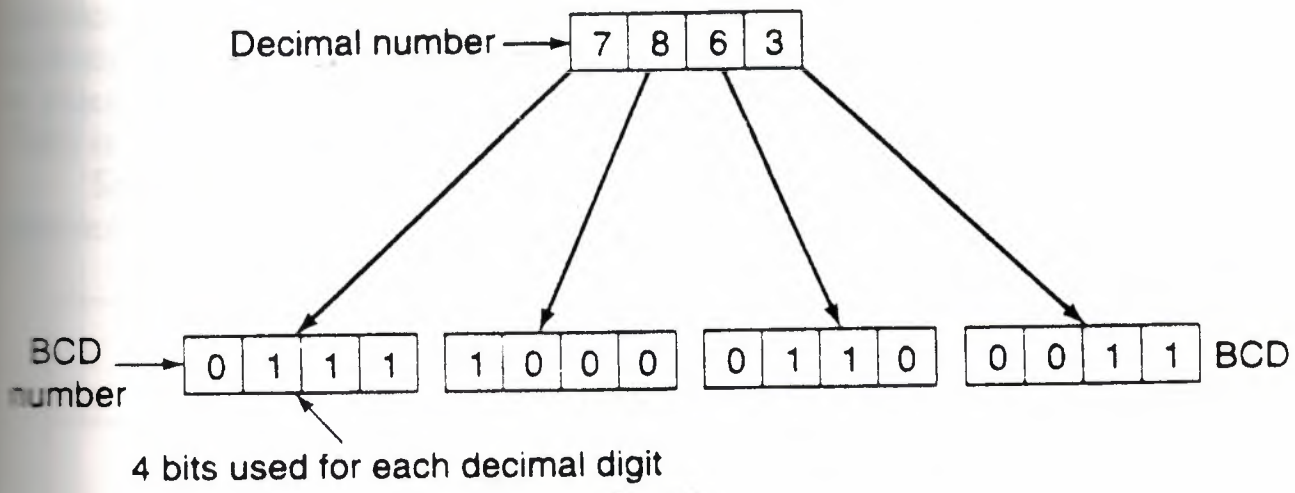
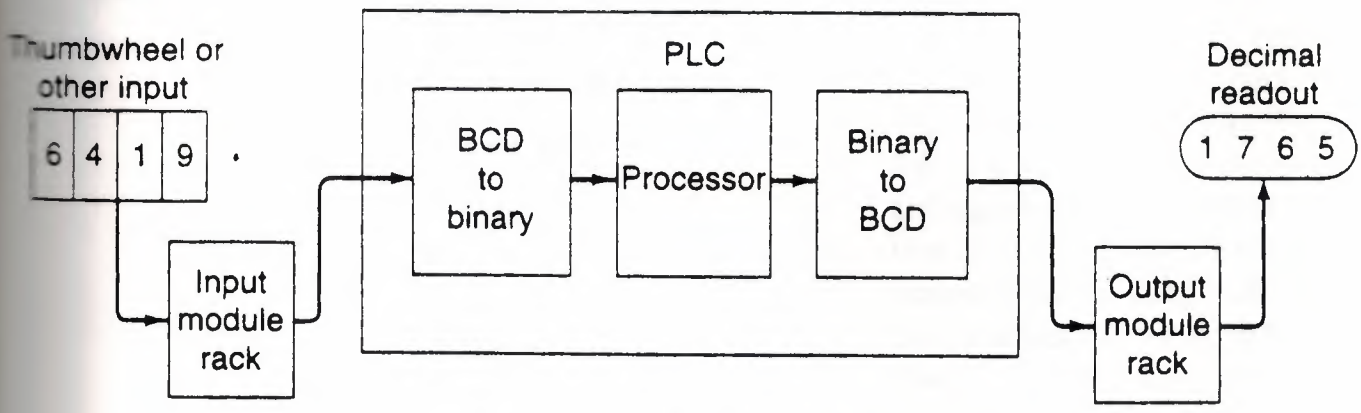
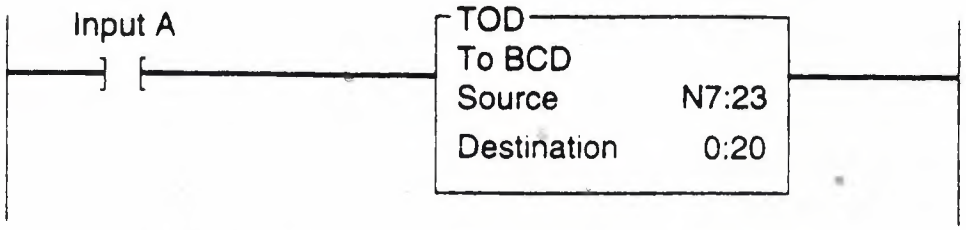


Fig. 23
The BCD representation of a decimal number



(*a*) PLC processors function in binary, not BCD or decimal



(*b*) Example of convert-to-decimal instruction

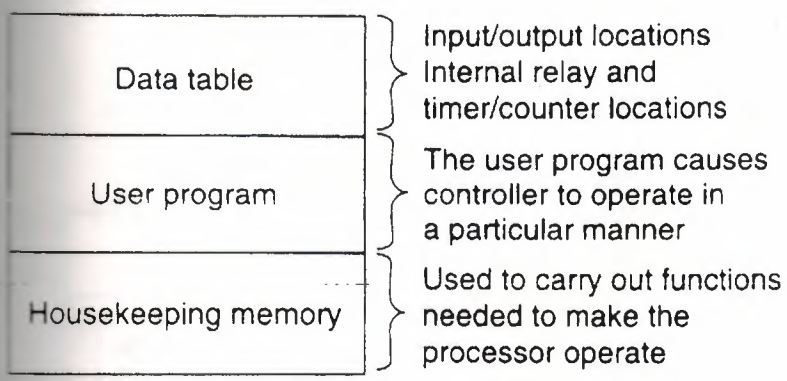
Fig. 24
PLC number conversion

a means of converting a code readily handled by humans (decimal) to a code readily handled by the equipment (benary).

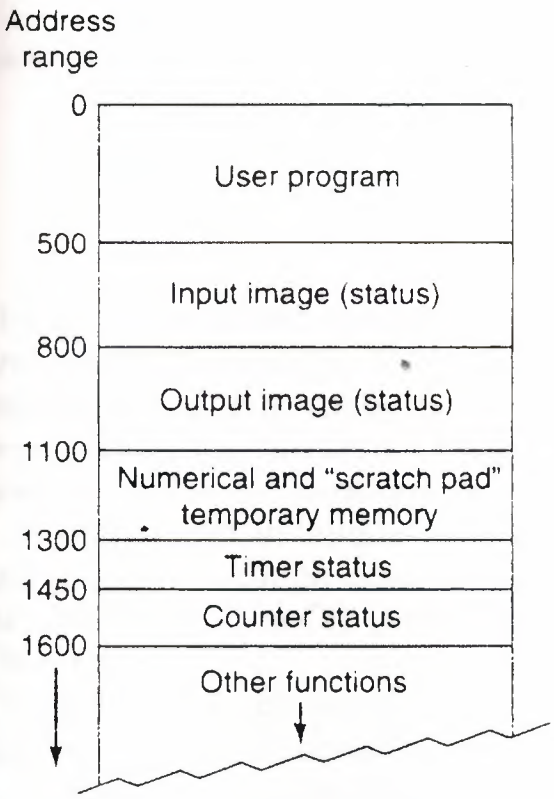
The BCD code employs 4 binary bits, with the weights of 1, 2, 4, and 8, to represent each numeral in the decimal system. This is referred to as the 8421 code, since 8421 is the natural binary progression. the BCD representation of a

decimal number is obtained by replacing each decimal digit by its BCD equivalent. To distinguish the BCD numbering system from a binary system, a BCD designation is placed to the right of the units digit. the BCD representation of the decimal 7863 is illustrated in Fig.

Scientific calculators are available to convert numbers back an forth between decimal, binary, octal, and hexadecimal. They are inexpensive and easy to use, for example, in converting a number displayed in decimal to one in binary. This simply involves one key stroke to change the display mode fram decimal to binary. In addition, many PLCs contain number conversion functions for converting numbers back and forth as illustrated in Fig. 12-25. As shojn in (Fig. 24 a) BCD-to-binary conversion is required for the input. Binary-to-BCD conversion is required for the output. In (Fig. 24 b) the convert-to-decimal instruction will convert the binary bit pattern at the source address, N7: 23, into a BCD bit pattern of the same decimal value as the destination address, 0:20. The instruction executes every time it is scanned and the instruction is true.



(*a*) General organization



(*b*) Memory map shows how memory is organized

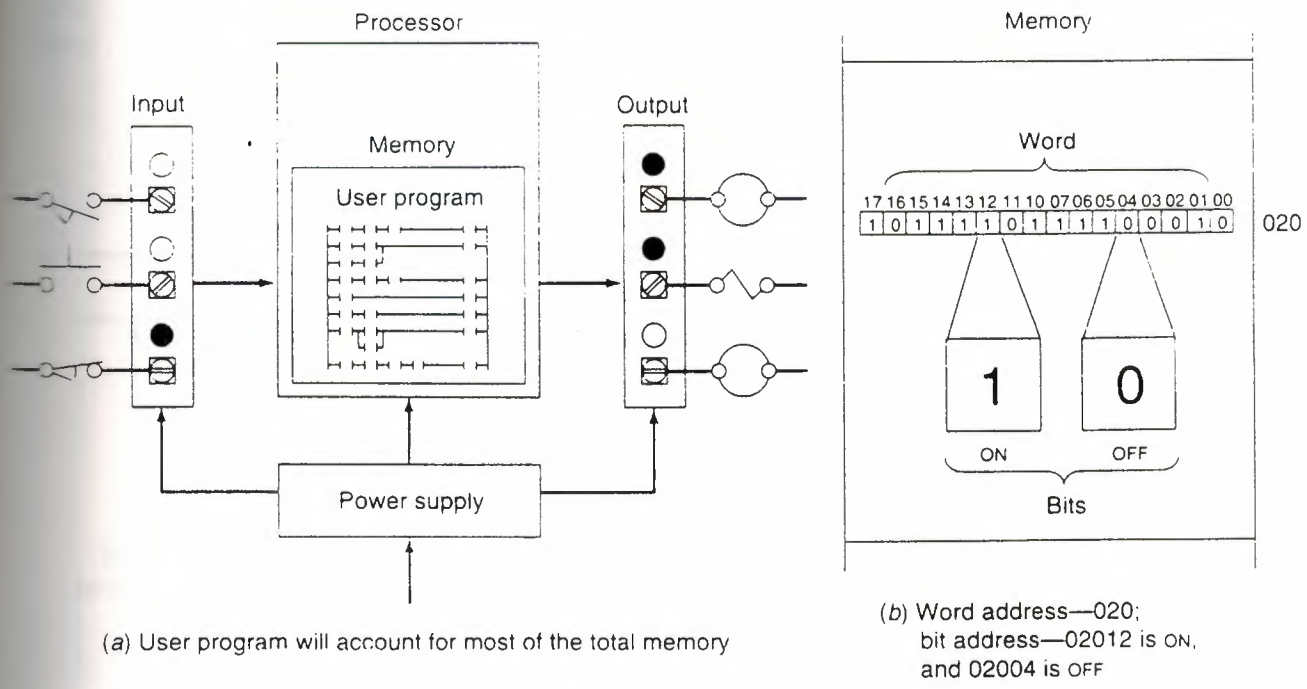
Fig. 25
Memory organization

BASICS OF PLC PROGRAMING

To program a programmable controller, it is necessary to have some knowjedge of how its memoruy is organized. Figure 25 bpows a typical PLC memory organization

known as a memory map. The individual sections, their order, and the sections enght will vary and may be fixed or variable depending on the manufacturer and model.

The user program is where the programmed logic ladder diagram is entered and stored (Fig. 26) The user program will accdunt for most of the total memoruy of a given PLC system. İt contains teh logic that controls the machine operationl. This instructions that are programmed in a ladder of memory.



(*a*) User program will account for most of the total memory

(*b*) Word address—020; bit address—02012 is ON, and 02004 is OFF

Fig. 26
User program

The data tale stores program. This includes such information as teh status of input and output devices (Fig. 27) , timer and counter values, data storage, and so on, Contents of the data table can be divided into two categories: status data and numbers or codes. Status is ON/OF type of information represented by Is and os, stored in unigue bit locations. Number or code information is represented by groups of bits stored in unigue register or word locations. The address nummer assigned to an instrucution associates it with o particular status bit. Tihs bit will be either ON (Logic 1) or OFF (Logic 0), indicating jhether the instruction is

TRUE OR FALSE

The data table can be divided into the following theree sections according to tih teylpe of information to be remembered; input image table, output image table,and timer and counter storağe, The input image table stores the satatus of digital inputs, which are connected tu input interface circuits. (Figure shows typical connections to the input image table throuğh thi input module. When the siwitch is closed, the processor detects a voltage at the input terminal and recordks that

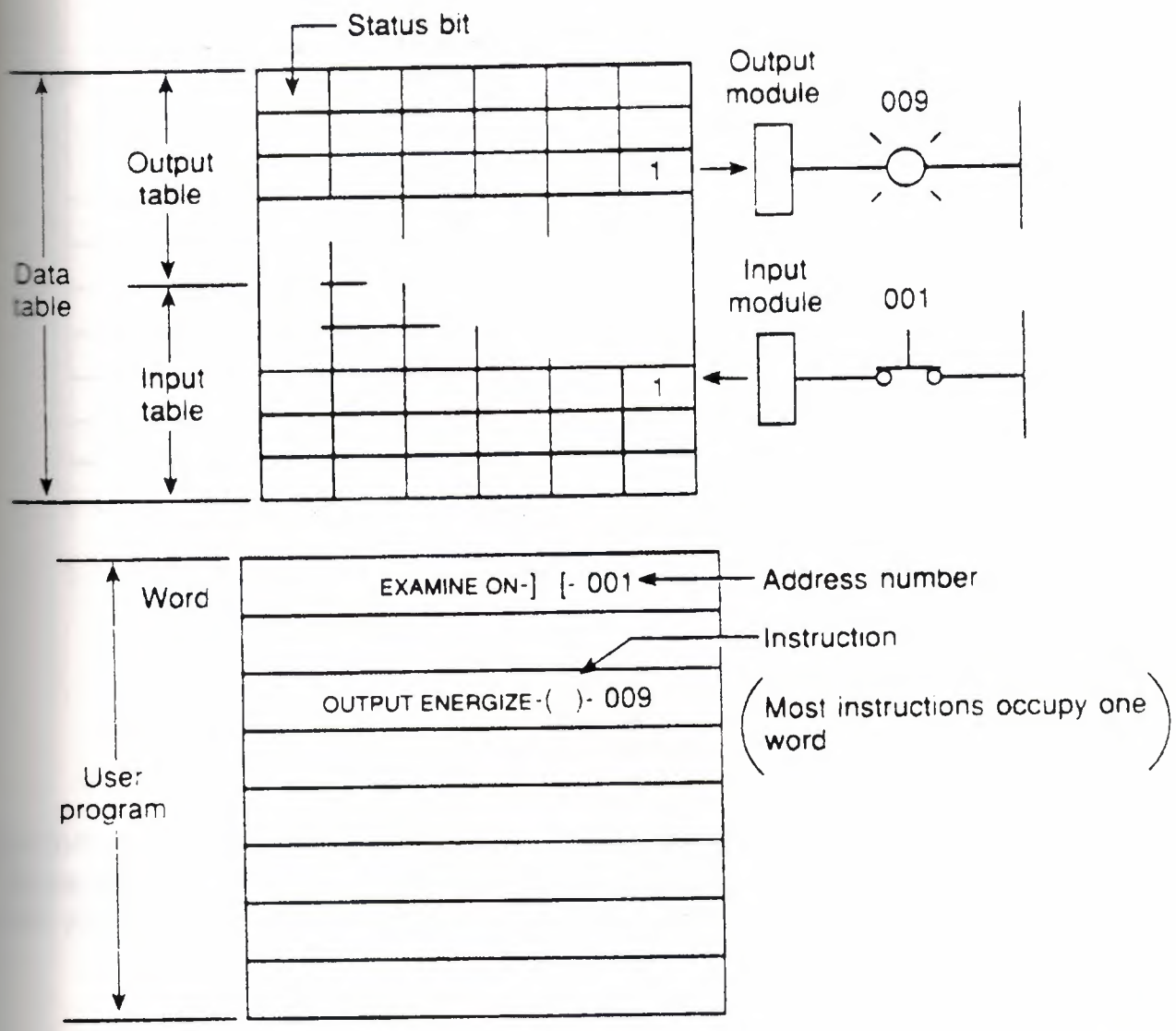


Fig. 27
Data table

information bay storing a binariy 1 in the proper bitlocation. Each connected input has a bit in the input image table that corresponds exactly to the terminal to wich the input is connected.

The input image table is constantly being changed to reflect the current status of the switch. If the input is ON (Switch closed), its corresponding bit in the table is set to 1. If the input is OFF (Switch open) the corresponding bit is "cleared," or reset to 0.

The output image table is an array of bits that controls the status of digital output devices, which are connected to output interface circuits. (Figure 29) shows a typical connection of lights to the output image table through the output module. The status of the lights (ON/OFF) is controlled by the user program and indicated by the presence of ıs (ON) and üs (OFF). Each connected output has a bit in the output image table that corresponds exactly to the terminal to wich the

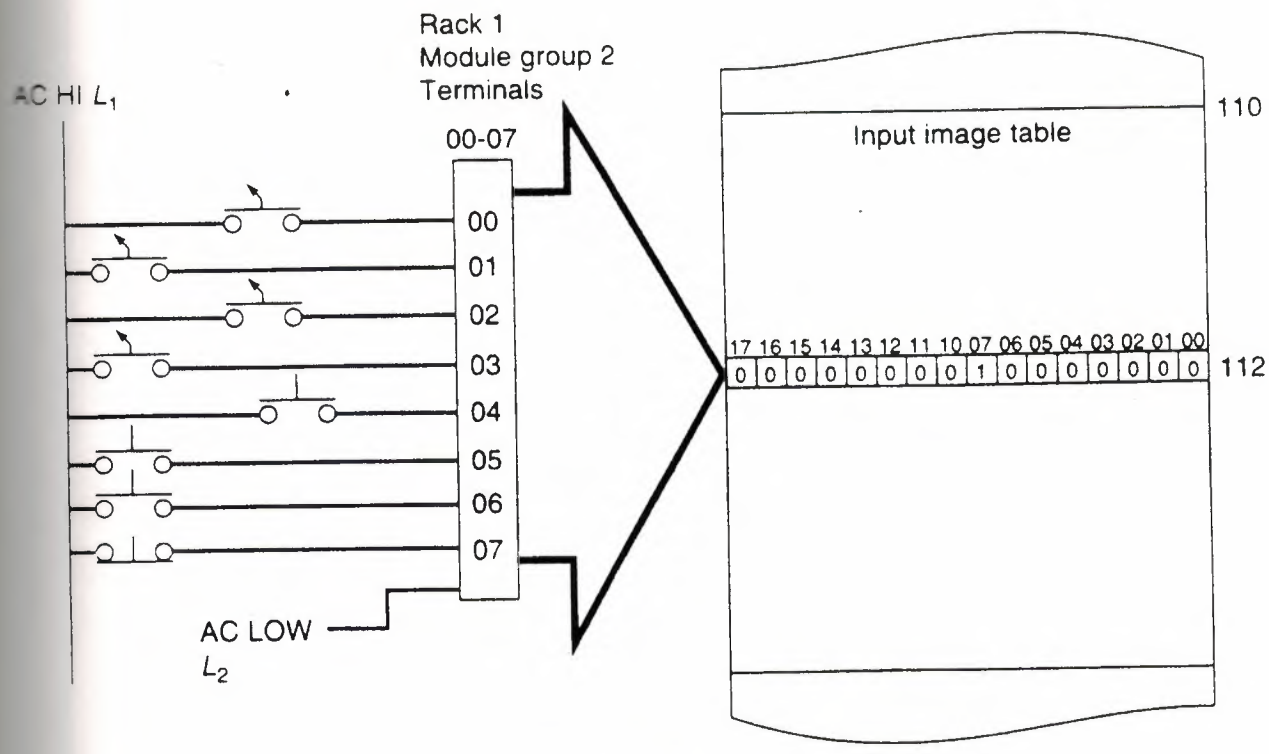


Fig. 28
Input image table

output is connected. If the program calls for a specific output to be ON, its corresponding bit in the table is set to 1. If the program calls forr the output to be OFF, its corresponding bit in the table is set to 0.

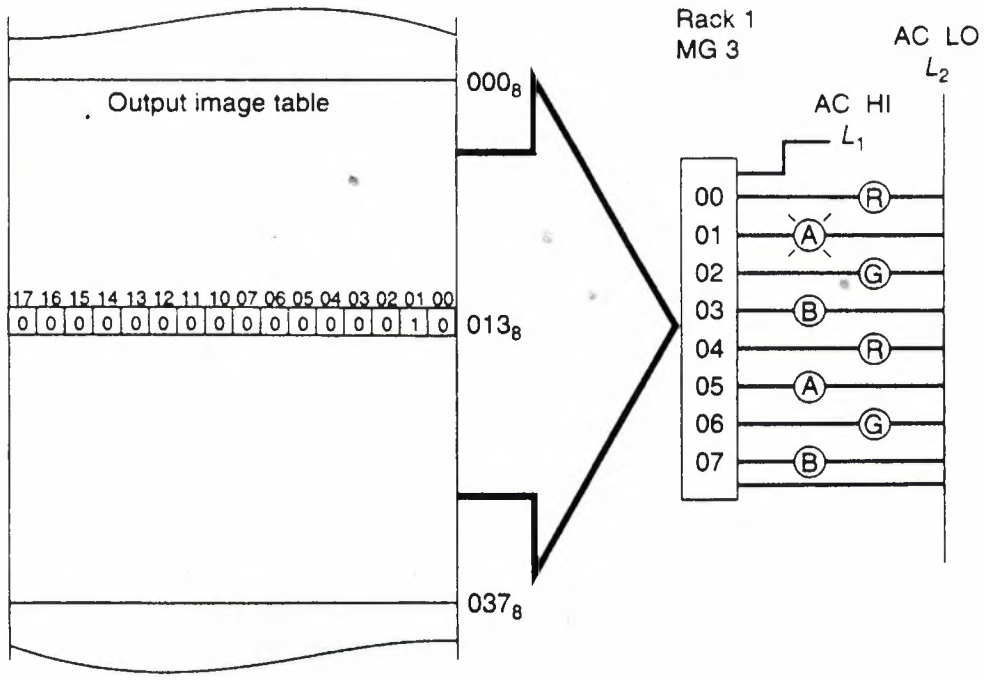


Fig. 29
Output image table

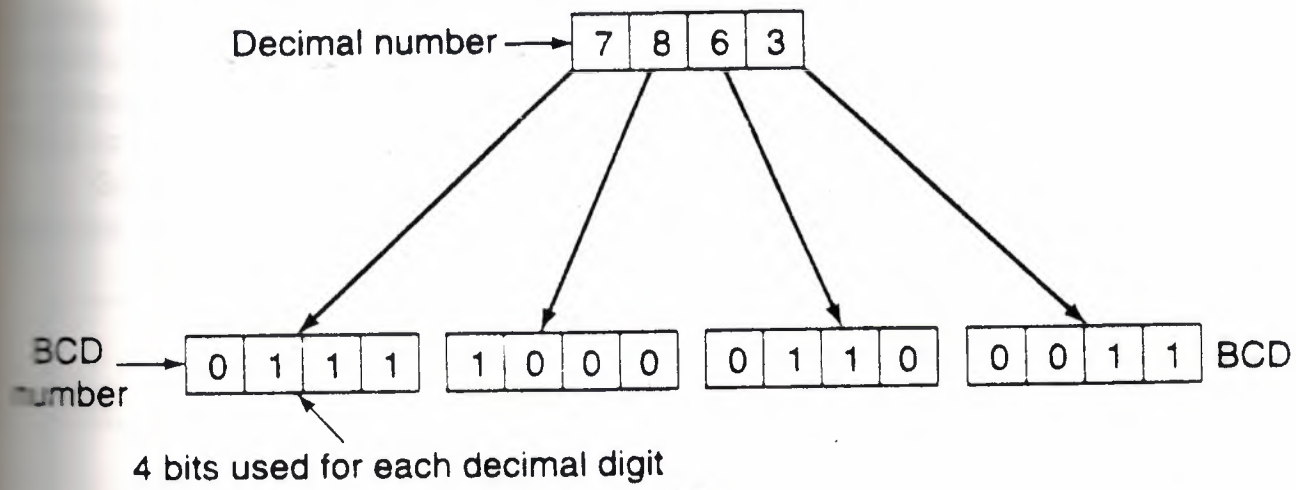
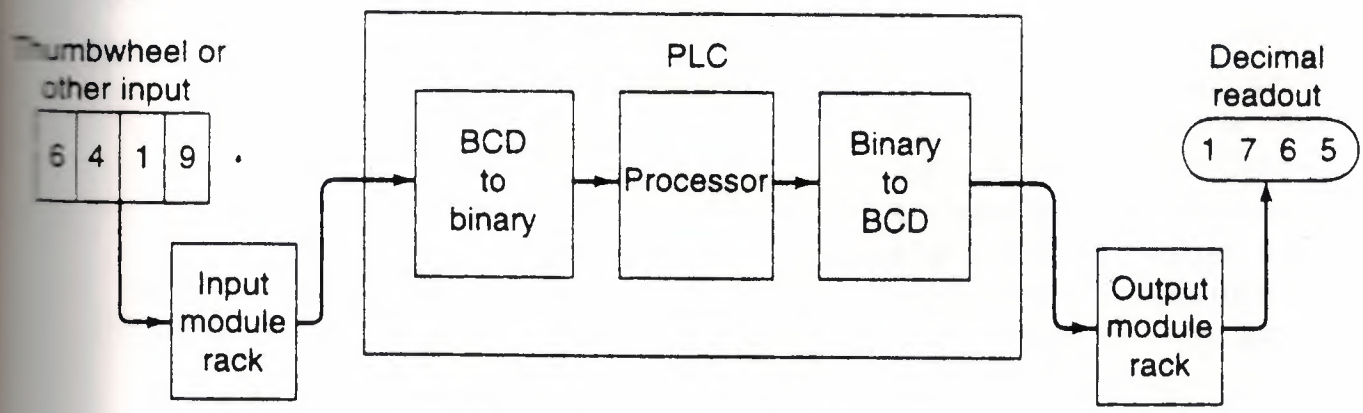
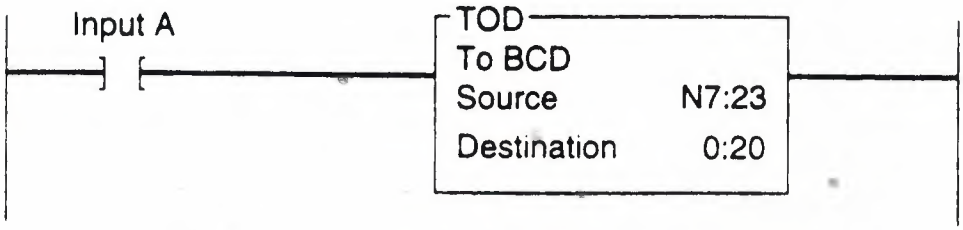


Fig. 23
The BCD representation of a decimal number



(*a*) PLC processors function in binary, not BCD or decimal



(*b*) Example of convert-to-decimal instruction

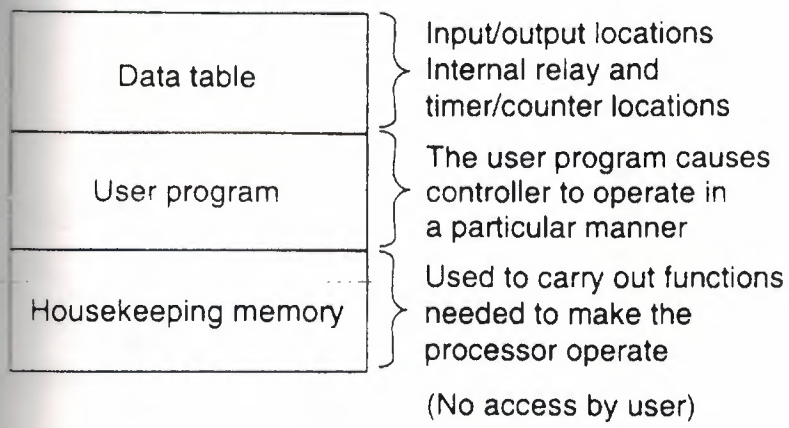
Fig. 24
PLC number conversion

a means of converting a code readily handled by humans (decimal) to a code readily handled by the equipment (benary).

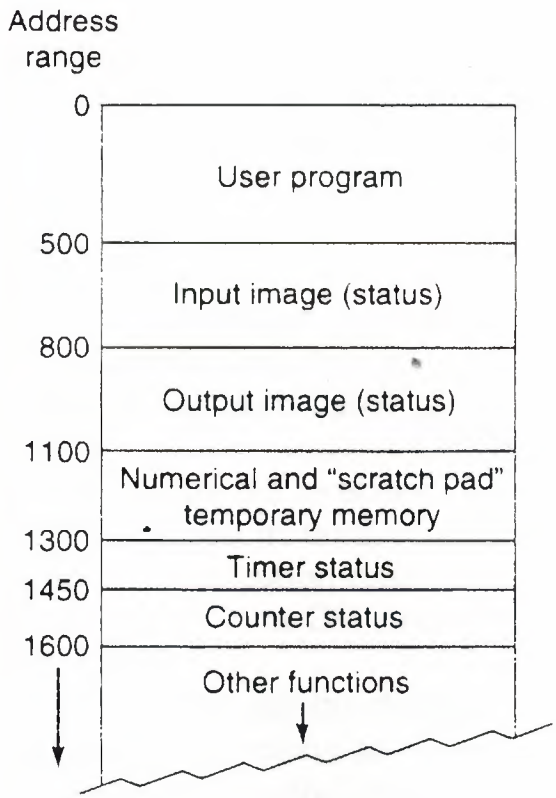
The BCD code employs 4 binary bits, with the weights of 1, 2, 4, and 8, to represent each numeral in the decimal system. This is referred to as the 8421 code, since 8421 is the natural binary progression. the BCD representation of a

decimal number is obtained by replacing each decimal digit by its BCD equivalent. To distinguish the BCD numbering system from a binary system, a BCD designation is placed to the right of the units digit. the BCD representation of the decimal 7863 is illustrated in Fig.

Scientific calculators are available to convert numbers back an forth between decimal, binary, octal, and hexadecimal. They are inexpensive and easy to use, for example, in converting a number displayed in decimal to one in binary. This simply involves one key stroke to change the display mode fram decimal to binary. In addition, many PLCs contain number conversion functions for converting numbers back and forth as illustrated in Fig. 12-25. As shojn in (Fig. 24 a) BCD-to-binary conversion is required for the input. Binary-to-BCD conversion is required for the output. In (Fig. 24 b) the convert-to-decimal instruction will convert the binary bit pattern at the source address, N7: 23, into a BCD bit pattern of the same decimal value as the destination address, 0:20. The instruction executes every time it is scanned and the instruction is true.



(*a*) General organization



(*b*) Memory map shows how memory is organized

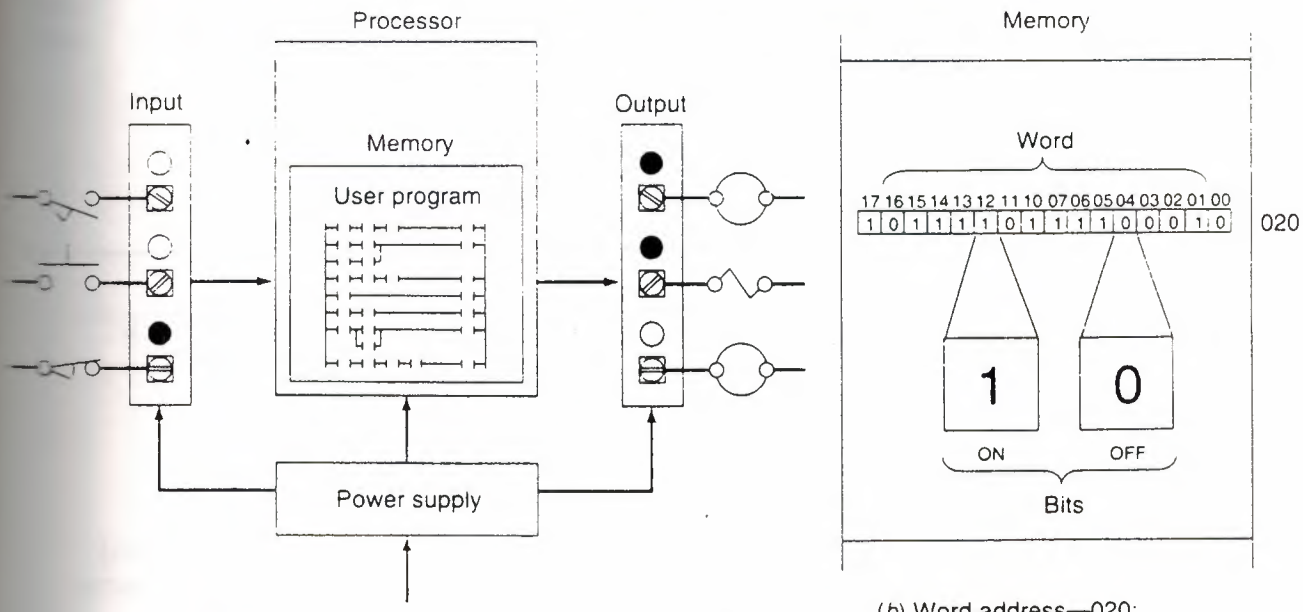
Fig. 25
Memory organization

BASICS OF PLC PROGRAMING

To program a programmable controller, it is necessary to have some knowjedge of how its memoruy is organized. Figure 25 bpows a typical PLC memory organization

known as a memory map. The individual sections, their order, and the sections enght will vary and may be fixed or variable depending on the manufacturer and model.

The user program is where the programmed logic ladder diagram is entered and stored (Fig. 26) The user program will accdunt for most of the total memoruy of a given PLC system. İt contains teh logic that controls the machine operationl. This instructions that are programmed in a ladder of memory.



(*a*) User program will account for most of the total memory

(*b*) Word address—020;
bit address—02012 is ON,
and 02004 is OFF

Fig. 26
User program

The data tale stores program. This includes such information as teh status of input and output devices (Fig. 27) , timer and counter values, data storage, and so on, Contents of the data table can be divided into two categories: status data and numbers or codes. Status is ON/OF type of information represented by Is and os, stored in unigue bit locations. Number or code information is represented by groups of bits stored in unigue register or word locations. The address nummer assigned to an instrucution associates it with o particular status bit. Tihs bit will be either ON (Logic 1) or OFF (Logic 0), indicating jhether the instruction is

TRUE OR FALSE

The data table can be divided into the following theree sections according to tih teylpe of information to be remembered; input image table, output image table,and timer and counter storağe, The input image table stores the satatus of digital inputs, which are connected tu input interface circuits. (Figure shows typical connections to the input image table throuğh thi input module. When the siwitch is closed, the processor detects a voltage at the input terminal and recordks that

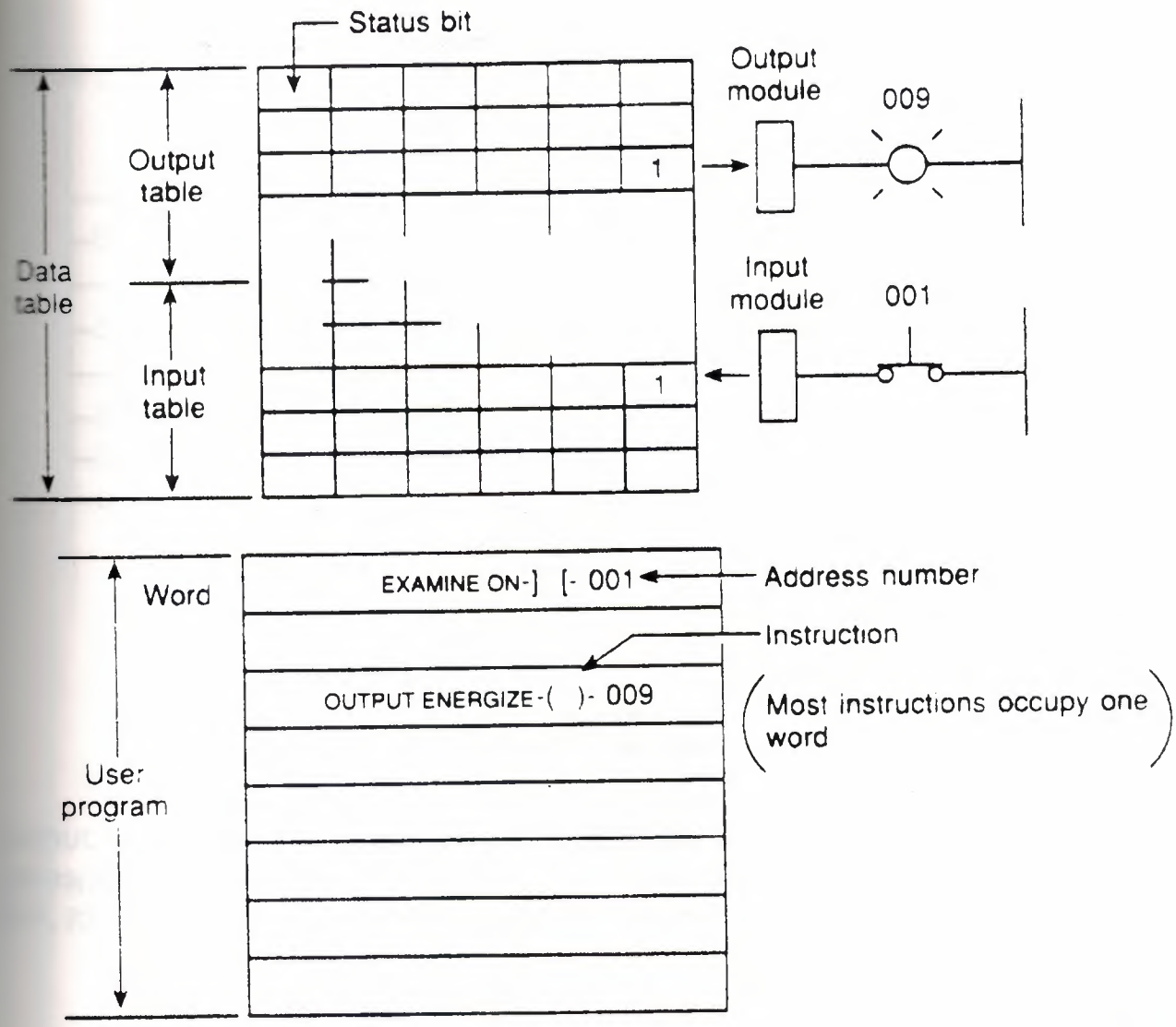


Fig. 27
Data table

information bay storing a binariy 1 in the proper bitlocation. Each connected input has a bit in the input image table that corresponds exactly to the terminal to wich the input is connected.

The input image table is constantly being changed to reflect the current status of the switch. If the input is ON (Switch closed), its corresponding bit in the table is set to 1. If the input is Off (Switch open) the corresponding bit is "cleared," or reset to 0.

The output image table is an array of bits that controls the status of digital output devices, which are connected to output interface circuits. (Figure 29) shows a typical connection of lights to the output image table through the output module. The status of the lights (ON/OFF) is controlled by the user program and indicated by the presence of ıs (ON) and üs (Off). Each connected output has a bit in the output image table that corresponds exactly to the terminal to wich the

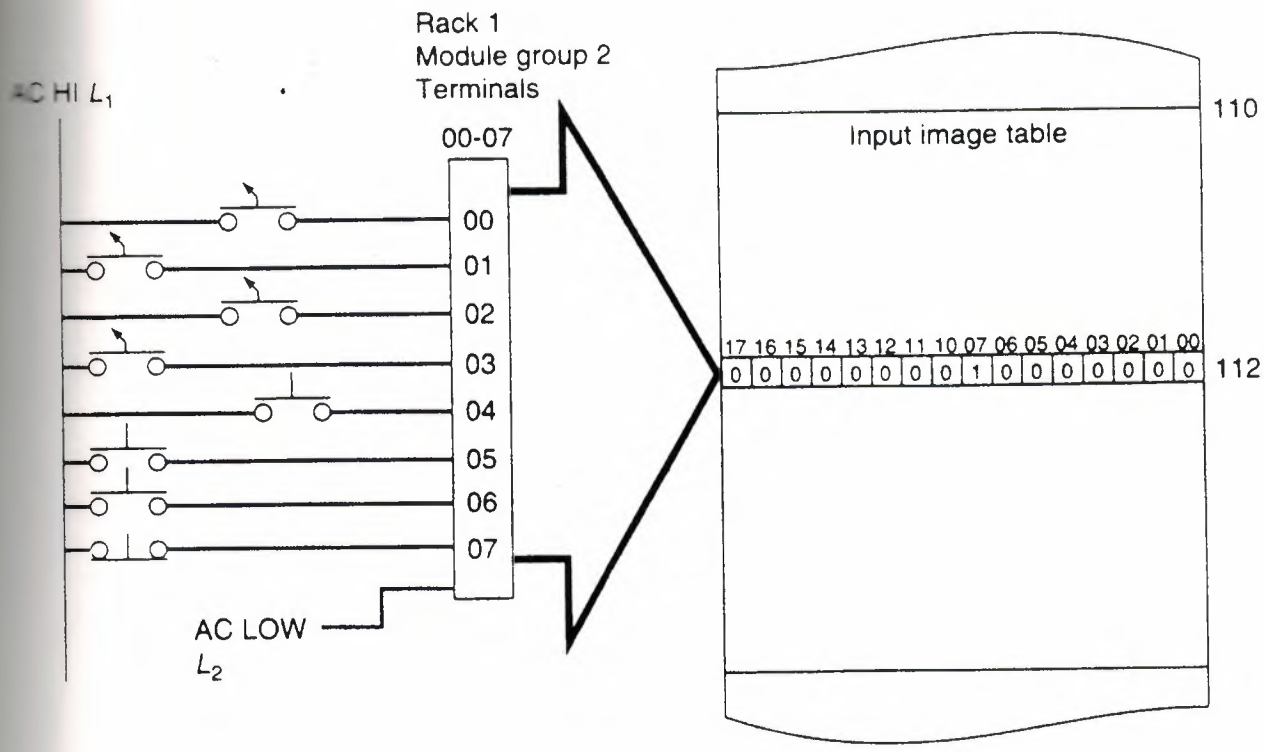


Fig. 28
Input image table

output is connected. If the program calls for a specific output to be ON, its corresponding bit in the table is set to 1. If the program calls forr the output to be OFF, its corresponding bit in the table is set to 0.

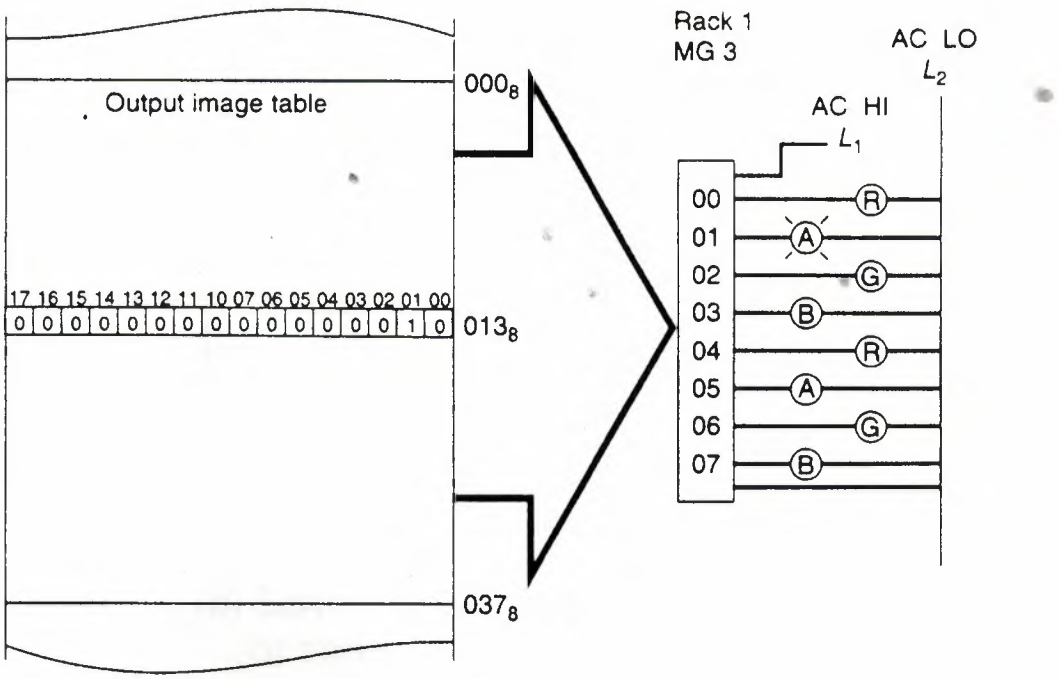
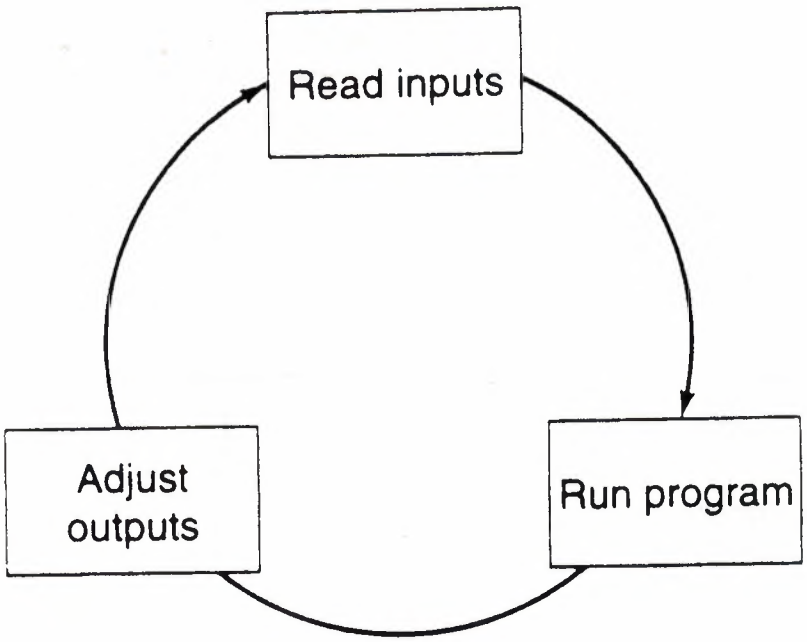
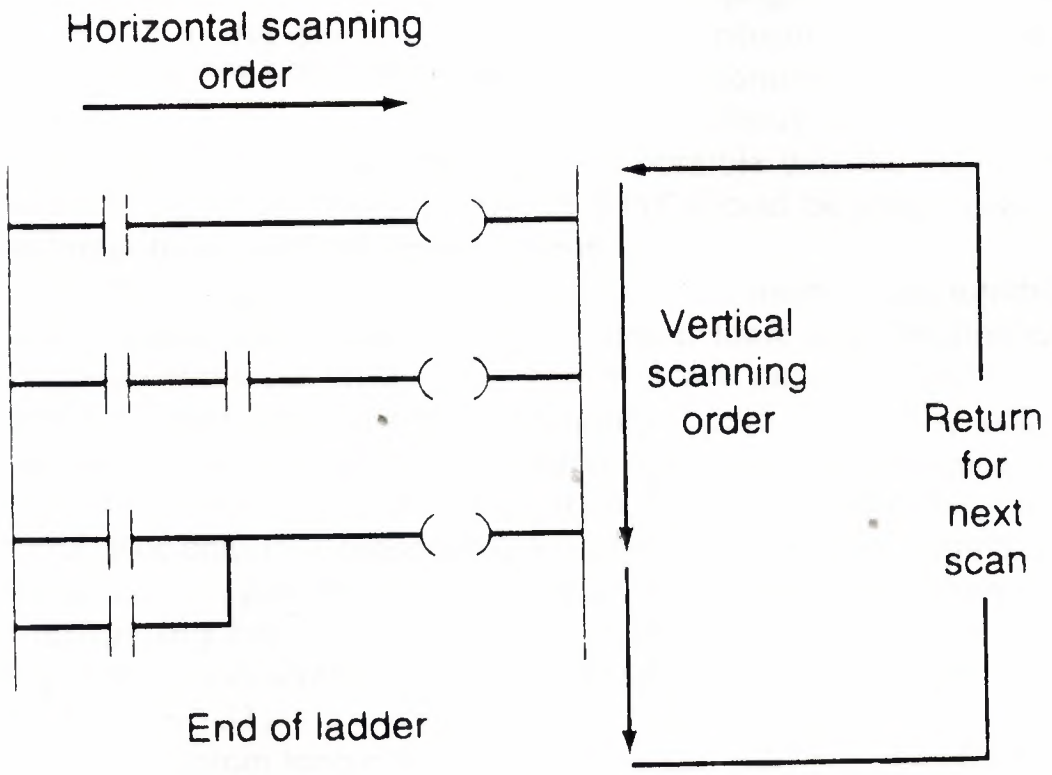


Fig. 29
Output image table



(*a*) Typical scan cycle



(*b*) Scanning can be vertical or horizontal

Fig. 30
Scan sequence

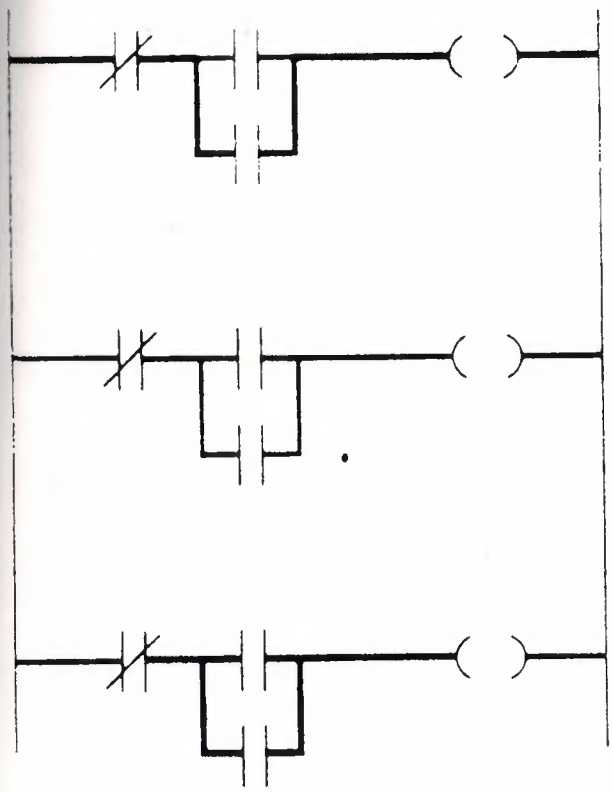


Fig. 31
Monitoring a relay ladder logic diyagram

During each operating cycle, the processor reads all the inputs,takes these values, and according to the user program energizes or deenergizes the outputs.This process is knojn asa scan.

The scan is normally a continuous and seguential process of readiing the status of inputs, evaluating the control logic,and updating the outputs.(Figure 30 İllus trates this process.A Scan its inputs and generate appropriate control responses at its outputs.Scan time varies with program content and length. A scan can take from about 1 to 20 ms. If a controler has to react to an ihput signal that changes states tji states tjice during the scan time, it is possible that the PLC willnever be able to detect this change. The scan time of a PLC should be knojn to ensure that scanning is faster than any field device operation.

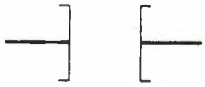
The term PLC programming language refers tu the method bay which the user communicates information to the PLC. Relay ladder logic was the first and most popular language available on the PLC, and it is still popular. Most PLCs on the market today can be programed either exclusively or partially in relay ladder logic.

Relay ladder logic is a graphical programming language designed to closely represent teh apperance of a wrid relay system. İt offers considerable advantages for PLC control. Not only is it reasonably intuitive, especially for techniscians with relay experience, it is particularly effective in an on-line mode when the PLC is actually performing control, Operation of the logic is apparent from the highlighting of the various relay contacts and coils on screen, identifying the logic state in real time (Fig 31)

The ladder diagram language is basically a symbolic set of instructions used to create the controller program. These ladder instruction symbols are arranged to obtian the desired contrl logic that be entered into the memory of teh PLC . Because the instruction set is composed of contact symbols lader diagram language is also referred to as contact symbology. Representations of contacts and coils are the basici saymbols of the logic ladder diagram instruction set (Fig.32)

A. EXAMINE ON instruction.

Symbol



Typically represents any input to control logic.

The input can be a connected switch or pushbutton, a contact from a connected output, or a contact from an internal output.

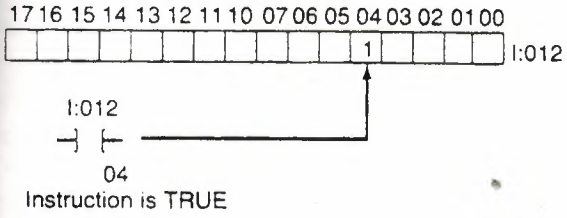
Has a bit-level address.

The status bit will be either 1 (ON) or 0 (OFF).

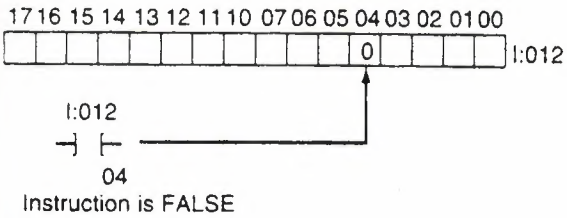
The status bit is examined for an ON condition.

If the status bit is 1 (ON), then the instruction is TRUE.

If the status bit is 0 (OFF), then the instructon is FALSE.



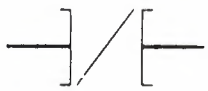
Instruction is TRUE



Instruction is FALSE

B. EXAMINE OFF instruction.

Symbol



Typically represents any input to the control logic.

The input can be a connected switch or pushbutton, a contact from a connected output, or a contact from an internal output.

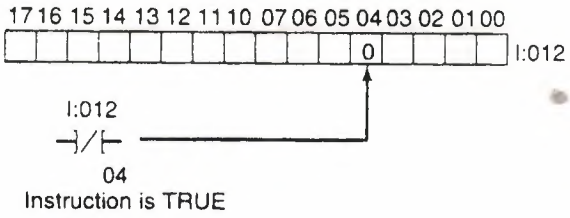
Has a bit-level address.

The status bit will be either 1 (ON) or 0 (OFF).

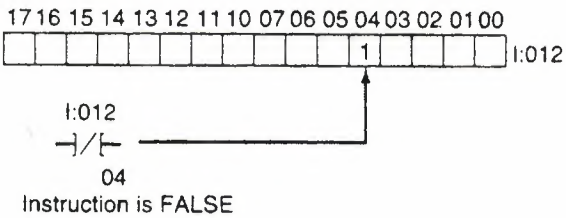
The status bit is examined for an OFF condition.

If the status bit is 0 (OFF), then the instruction is TRUE.

If the status bit is 1 (ON), then the instruction is FALSE.



Instruction is TRUE

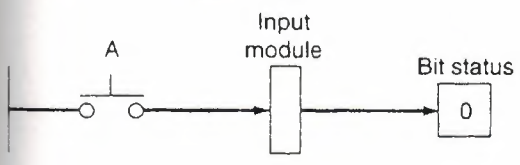


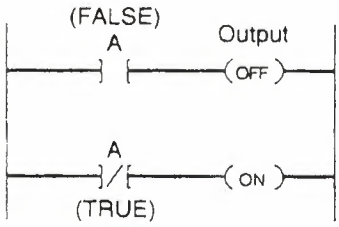
Instruction is FALSE

(*Continued on next page.*)

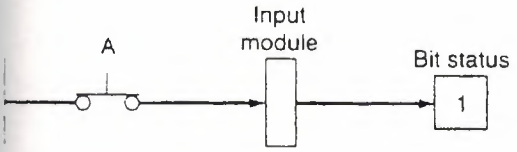
Fig. 32
(Continued)

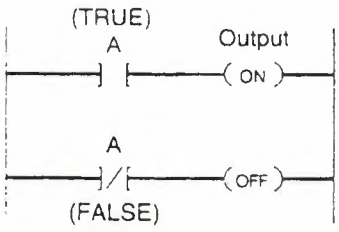
C. Status bit examples





Button not actuated





Button actuated

D. OUTPUT ENERGIZE

Symbol

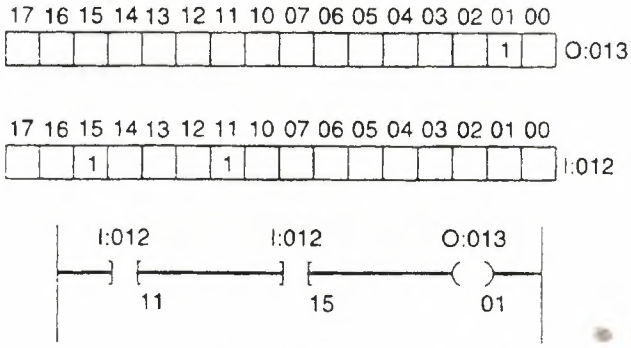
Typically represents any output that is controlled by some combination of input logic.

An output can be a connected device or an internal output (internal relay).

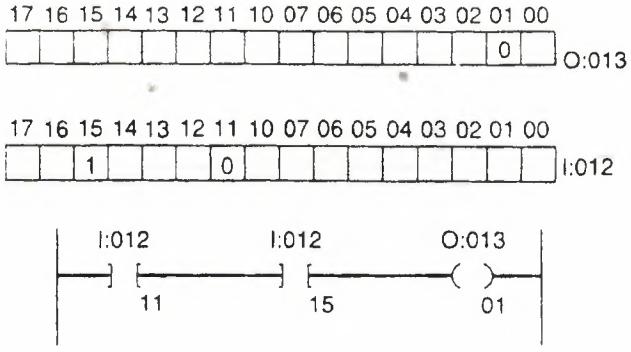
If any left-to-right path of input conditions is TRUE, the output is energized (turned ON).

The status bit of the addressed OUTPUT ENERGIZE instruction is set to 1 (ON) when the rung is TRUE.

The status bit of the addressed OUTPUT ENERGIZE instruction is reset to 0 (OFF) when the rung is FALSE.

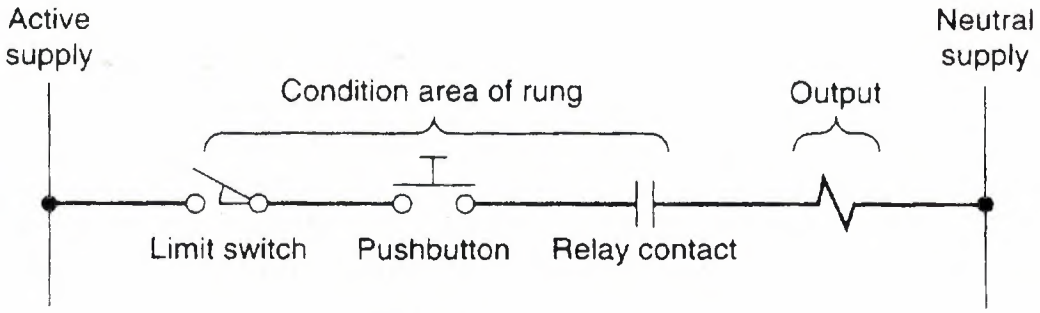


OUTPUT ENERGIZE instruction—TRUE

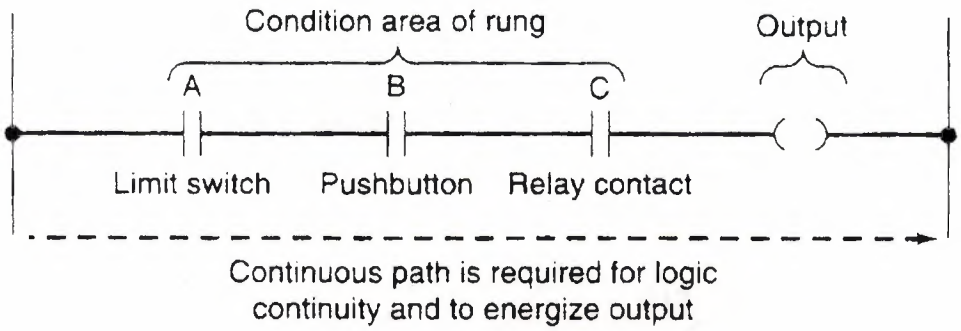


OUTPUT ENERGIZE instruction—FALSE

Fig. 32a
Basic set of instructions that perform functions similar to relay functions



(*a*) Hard-wired format

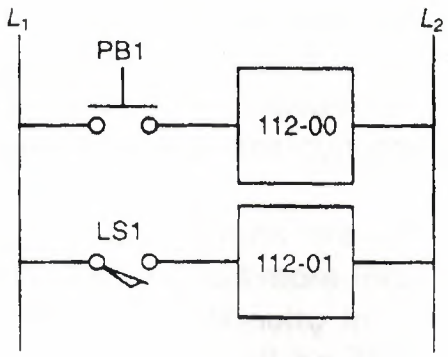


(*b*) PLC ladder diagram format

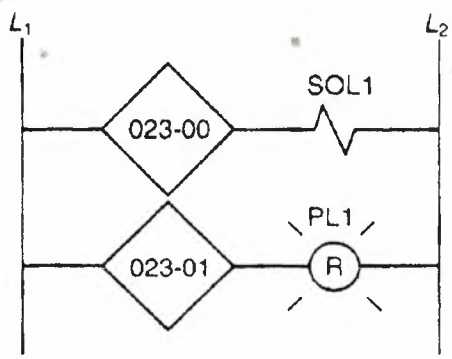
Fig. 33
Ladder rung

The following three are the basic symbols used to translate relay control logic to contact symbolic logic:

Instruction	symbol	Mnemonic
EXAMINE ON		XIC
EXAMINE OFF		XIO
OUTPUT ENERGIZE		OTE



(*a*) Input connections



(*b*) Output connections

Fig. 34
I/O connection diagram

The main function of the logic ladder diagram program is to control outputs based on input conditions. This control is accomplished through the use of what is referred to as a ladder rung. In general, a rung consists of a set of input conditions,represented by contact instructions, and an output instruction at the end of the rung represented bay the coil symbol (Fig 33) Each contact or coil symbol is referenced with an address number that identifies what is bein evaluated and what is being controlled. The same contact instructioin can be used throughout the program whenever that condition needs to be evaluated. For an output to be activated or energized, at least one left-to-right path of contacts must be closed. O complete closed path is referred to as having logic continuity. When logic continuity exists in at least one path the rung condition is said to be TRUE, The rung condition is FALSE İf no path has continuity.

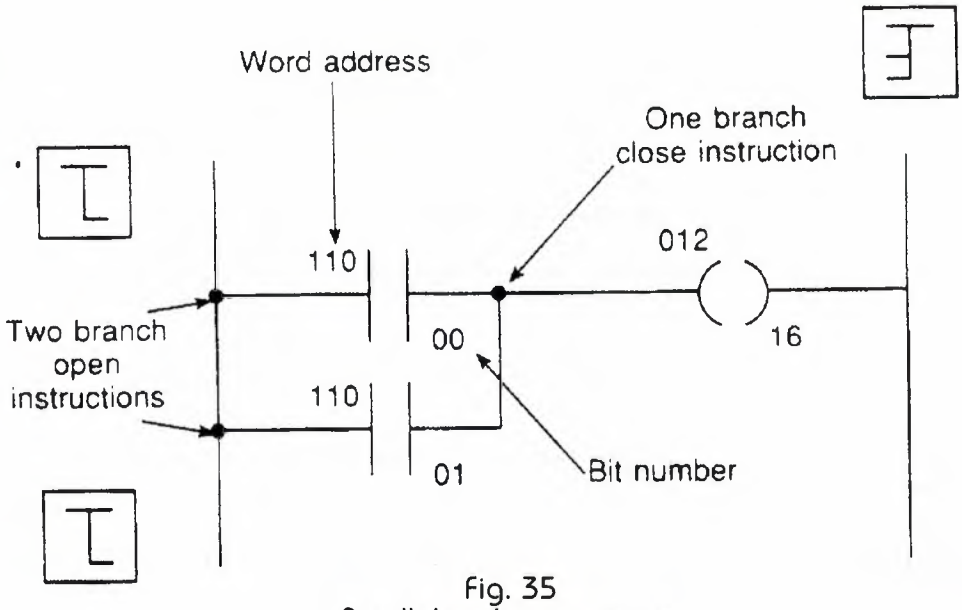
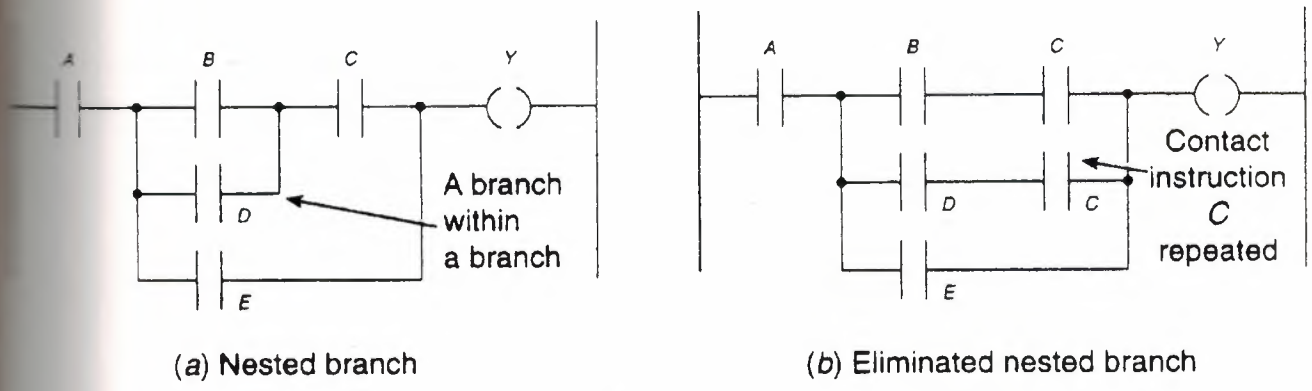


Fig. 35
Parallel path instructions.

To complete the entry of a relay-type instruction,yu must assign an addresse number to it.This munber will indicate what PLC inputis connected to what input device and what PLC output will drive what output device. The assignment of 1/O address is some times included in the 1/0 connection diagram as shown in (Fig 34) inputs and outputs are typically represented buy sguares and diamonds, respectively.

Branch instructions are used to create paralel paths of input condition instructions.This allows more than one combination of input conditions (OR Logic) tu establish logic continuity in a rung. (Figure 35) illustrates o simple branching condition. TKhe rung will be TRUE A branch START instruction is used to beğin each paralel logic brancl. A single branch CLOSE instruction is used to close the parallel branch.

In some PLC models the programming of a branch circuit within a brnanch



(*a*) Nested branch (*b*) Eliminated nested branch

Fig. 36
Nested branch

circuit o a nested barnch cüannot be done directly. İt is possible, however, to program a logically eguivalent Programming morğe than the allowable series elements or parallel branches will resultin an error massağe. Also there is a further limitation wihth some PLCs:there can be onlay one output per rung. and the output must belocated on the right-hand end of a rung.

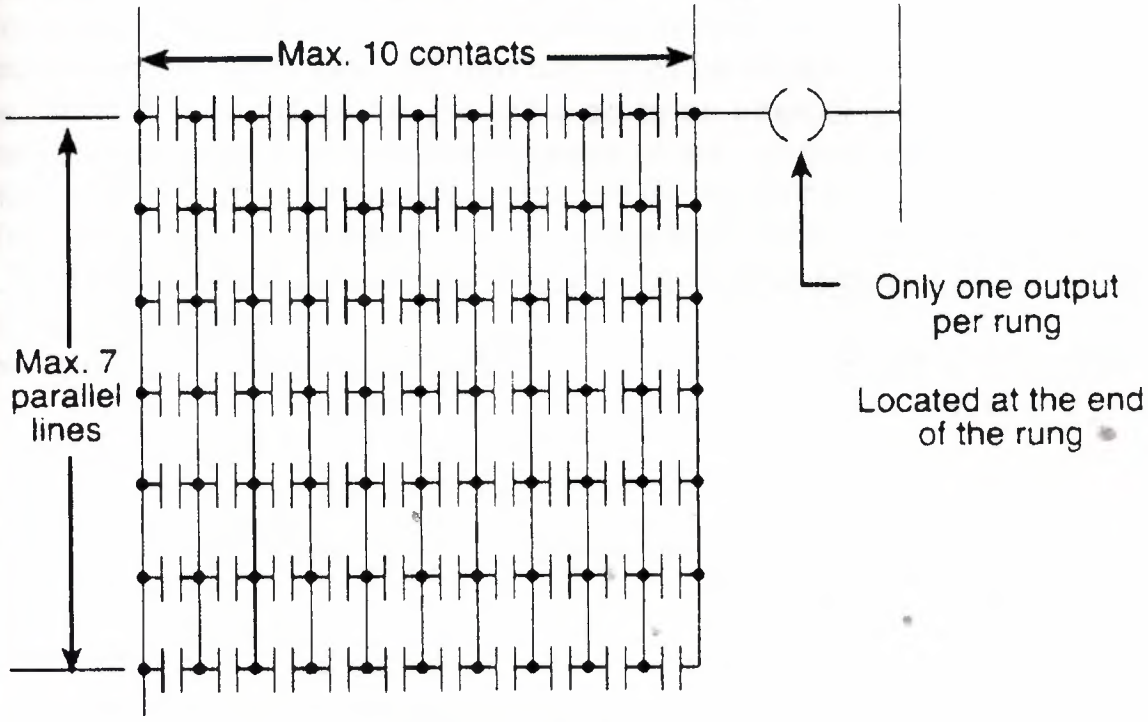


Fig. 37
Typical PLC matrix limitation diagram

Most PLCs have an areo of the memoruy allocated for what are known as internal storage bits.These storage bits are also called internal outpus, internal coils, intenal The internal output operates just as any output that is controlled by programmed logic; however, the output is used strictly fofinternal output doesnot directly control an output device.

An internal control relay can be used when a circuit reguires more series

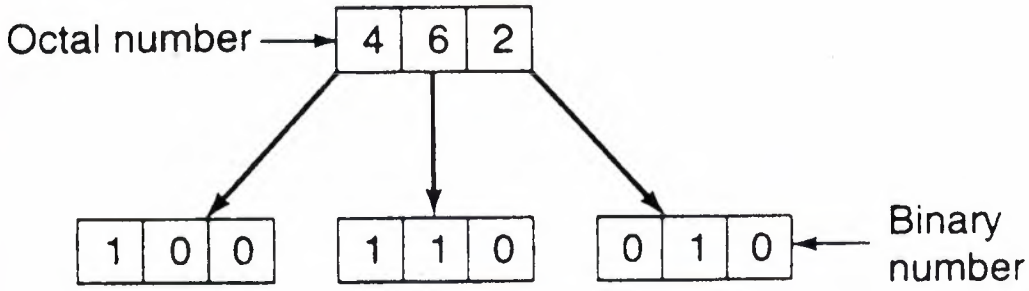


Fig. 21
Converting an octal number to binary number

relative capacity of the systems' memories. Normally programmable controllers do not require storage space above 128 K and in many instances need a memory size of only 1 K to 2 k.

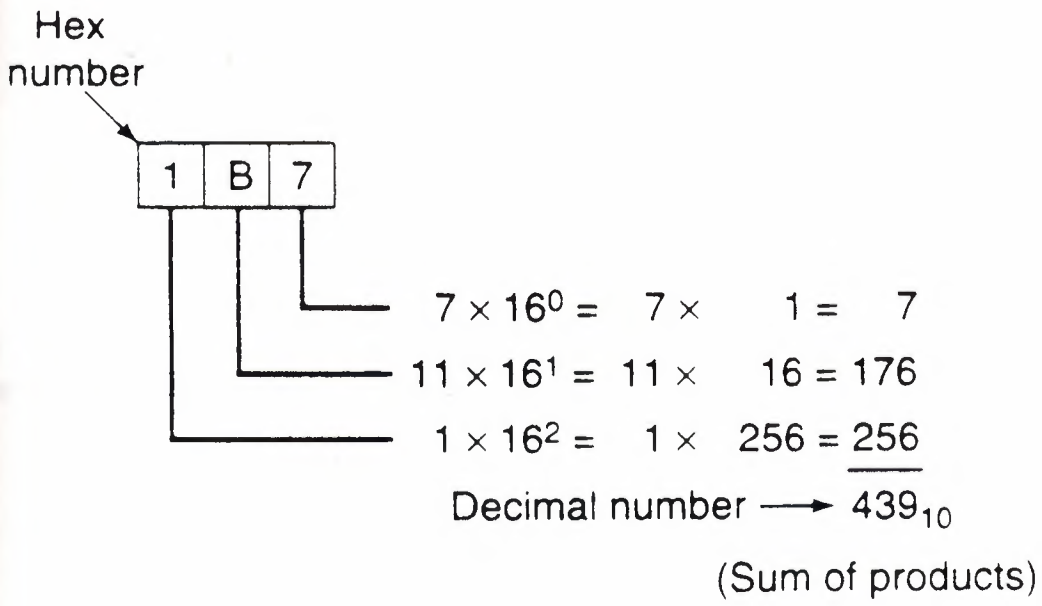
To convert a decimal number to its binary equivalent, we must perform a series of divisions by 2. (Figure 19) illustrates the conversion of the decimal number 47 to binary. We start by dividing the decimal number by 2. If there is no remainder, a 0 is placed in the LSB. The result of the division is brought down, and the process repeated until the result of successive divisions has been reduced to 0.

To express a number in the binary system requires many more digits than in the decimal system. Too many binary digits can become cumbersome to read or write. To solve this problem, other related numbering systems are brought into use.

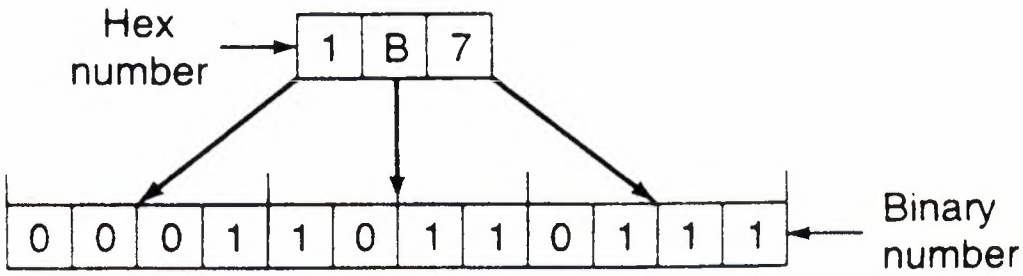
The octal numbering system, a base 8 system, is often used in microprocessor, computer, and programmable controller systems because 8 data bits make up a byte of information which can be addressed by the PLC user or programmer. In some instances, programmable controller manufacturers use the octal system to number wiring terminals, programmable controller racks, and other PLC hardware. The octal number system makes use of 8 digits: 0 through 7. As in all other number systems, each digit in an octal number has a weighted decimal value according to its position. (Figure 20) illustrates how the octal number 462 is converted to its decimal equivalent: 306.

As mentioned, octal is used as a convenient means of handling large binary numbers. For example, the octal number 462 can be converted to its binary equivalent by assembling the 3-bit groups as illustrated in (Fig. 21) Thus, octal 462 is binary 100110010 and decimal 306. Notice the simplicity of the notation. The octal 462 is much easier to read and write than its binary equivalent.

The hexadecimal (hex) number system provides even shorter notation than



(*a*) Converting a hexadecimal number to a decimal number



(*b*) Converting a hexadecimal number to a binary number

Fig. 22
Hexadecimal numbering system

octal. Hexadecimal uses a base of 16. It employs 16 digits: numbers 0 through 9, and letters A through F, with A through F being substituted for numbers 10 through 15, respectively. The techniques for converting hexadecimal to decimal and decimal to hexadecimal are the same as those used for binary and octal (Fig. 22)

The binary coded decimal (BCD) system provides a convenient means to handle large numbers that need to be input to or output from a PLC. The BCD system represents decimal numbers a patterns of Is and Os. This system provides

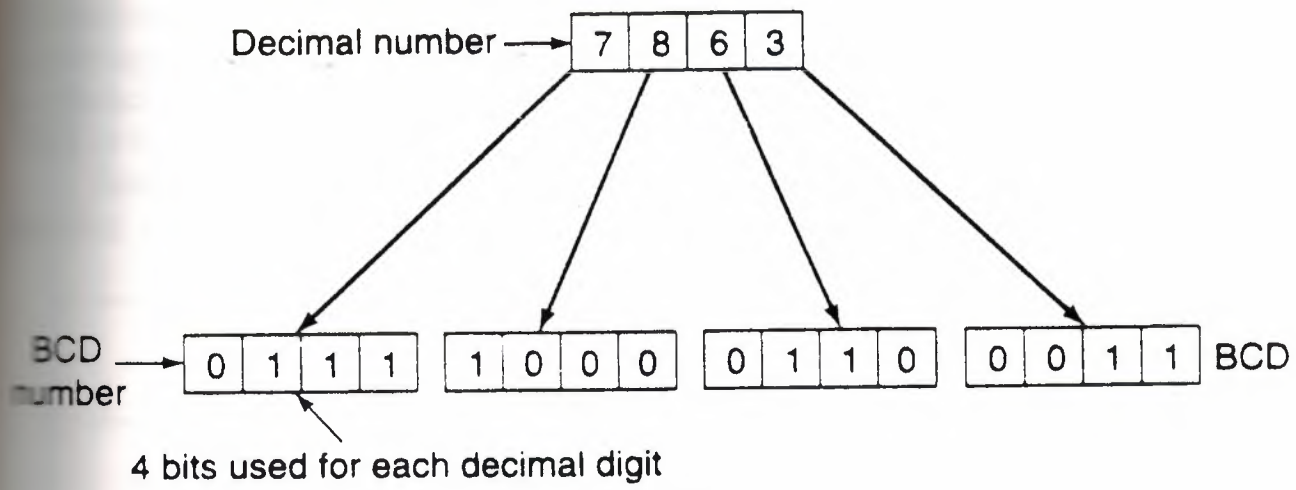
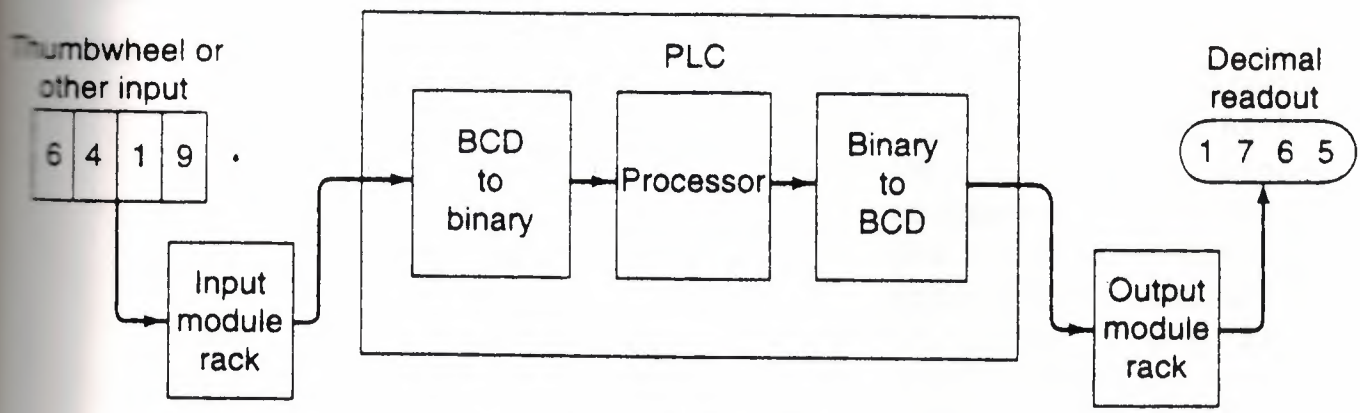
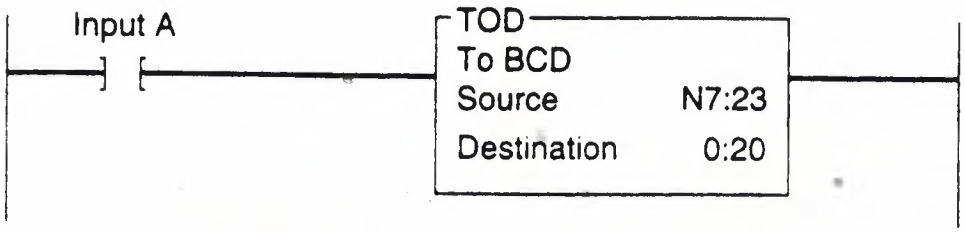


Fig. 23
The BCD representation of a decimal number



(*a*) PLC processors function in binary, not BCD or decimal



(*b*) Example of convert-to-decimal instruction

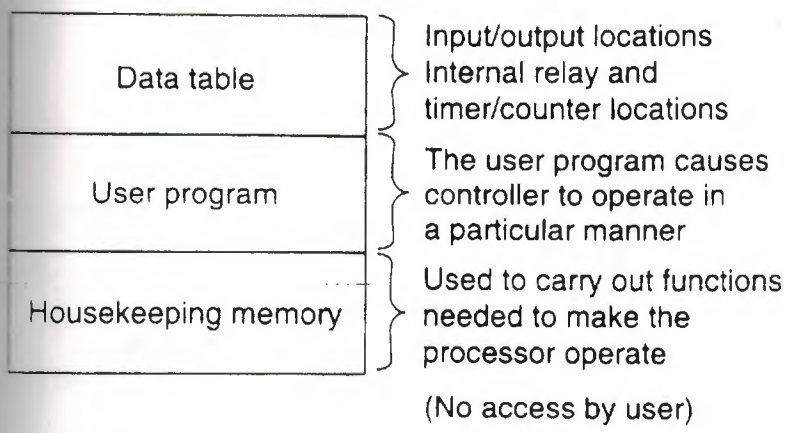
Fig. 24
PLC number conversion

a means of converting a code readily handled by humans (decimal) to a code readily handled by the equipment (benary).

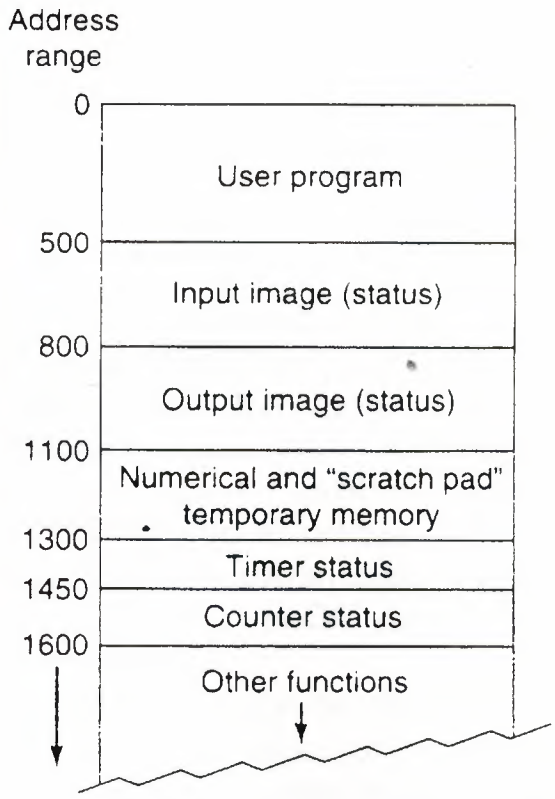
The BCD code employs 4 binary bits, with the weights of 1, 2, 4, and 8, to represent each numeral in the decimal system. This is referred to as the 8421 code, since 8421 is the natural binary progression. the BCD representation of a

decimal number is obtained by replacing each decimal digit by its BCD equivalent. To distinguish the BCD numbering system from a binary system, a BCD designation is placed to the right of the units digit. the BCD representation of the decimal 7863 is illustrated in Fig.

Scientific calculators are available to convert numbers back an forth between decimal, binary, octal, and hexadecimal. They are inexpensive and easy to use, for example, in converting a number displayed in decimal to one in binary. This simply involves one key stroke to change the display mode fram decimal to binary. In addition, many PLCs contain number conversion functions for converting numbers back and forth as illustrated in Fig. 12-25. As shojn in (Fig. 24 a) BCD-to-binary conversion is required for the input. Binary-to-BCD conversion is required for the output. In (Fig. 24 b) the convert-to-decimal instruction will convert the binary bit pattern at the source address, N7: 23, into a BCD bit pattern of the same decimal value as the destination address, 0:20. The instruction executes every time it is scanned and the instruction is true.



(*a*) General organization



(*b*) Memory map shows how memory is organized

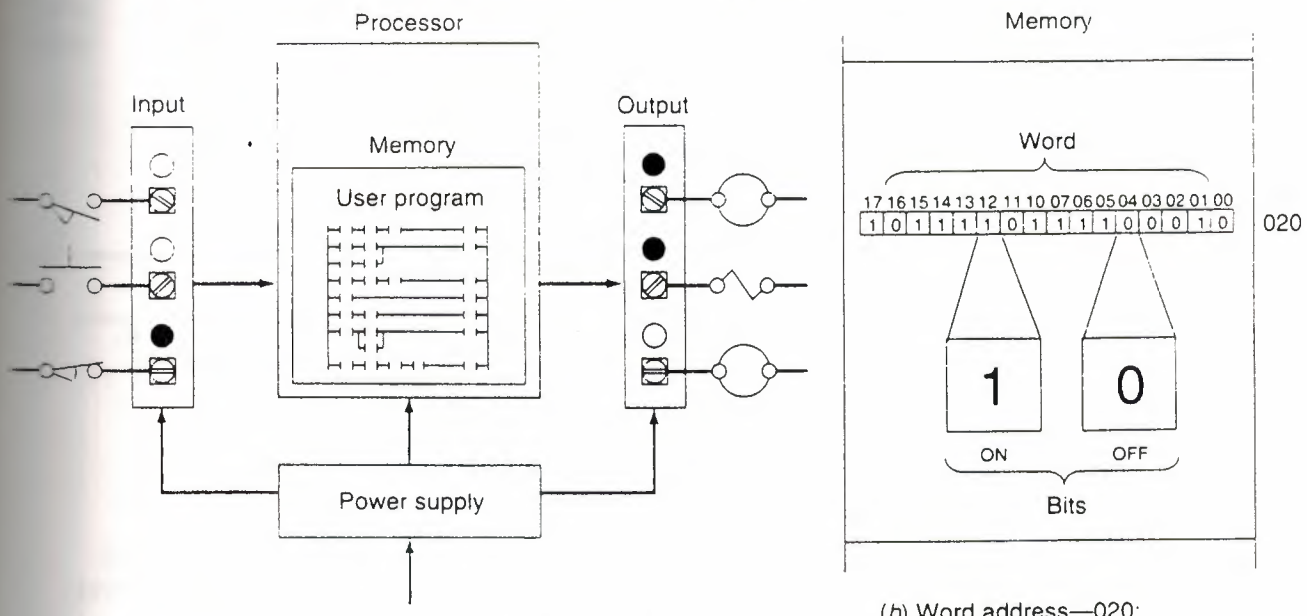
Fig. 25
Memory organization

BASICS OF PLC PROGRAMING

To program a programmable controller, it is necessary to have some knowjedge of how its memoruy is organized. Figure 25 bpows a typical PLC memory organization

known as a memory map. The individual sections, their order, and the sections enght will vary and may be fixed or variable depending on the manufacturer and model.

The user program is where the programmed logic ladder diagram is entered and stored (Fig. 26) The user program will accdunt for most of the total memoruy of a given PLC system. It contains teh logic that controls the machine operationl. This instructions that are programmed in a ladder of memory.



(*a*) User program will account for most of the total memory

(*b*) Word address—020;
bit address—02012 is ON,
and 02004 is OFF

Fig. 26
User program

The data tale stores program. This includes such information as teh status of input and output devices (Fig. 27) , timer and counter values, data storage, and so on, Contents of the data table can be divided into two categories: status data and numbers or codes. Status is ON/OF type of information represented by Is and os, stored in unigue bit locations. Number or code information is represented by groups of bits stored in unigue register or word locations. The address nummer assigned to an instrucution associates it with o particular status bit. Tihs bit will be either ON (Logic 1) or OFF (Logic 0), indicating jhether the instruction is

TRUE OR FALSE

The data table can be divided into the following theree sections according to tih teylpe of information to be remembered; input image table, output image table,and timer and counter storağe, The input image table stores the satatus of digital inputs, which are connected tu input interface circuits. (Figure shows typical connections to the input image table throuğh thi input module. When the siwitch is closed, the processor detects a voltage at the input terminal and recordks that

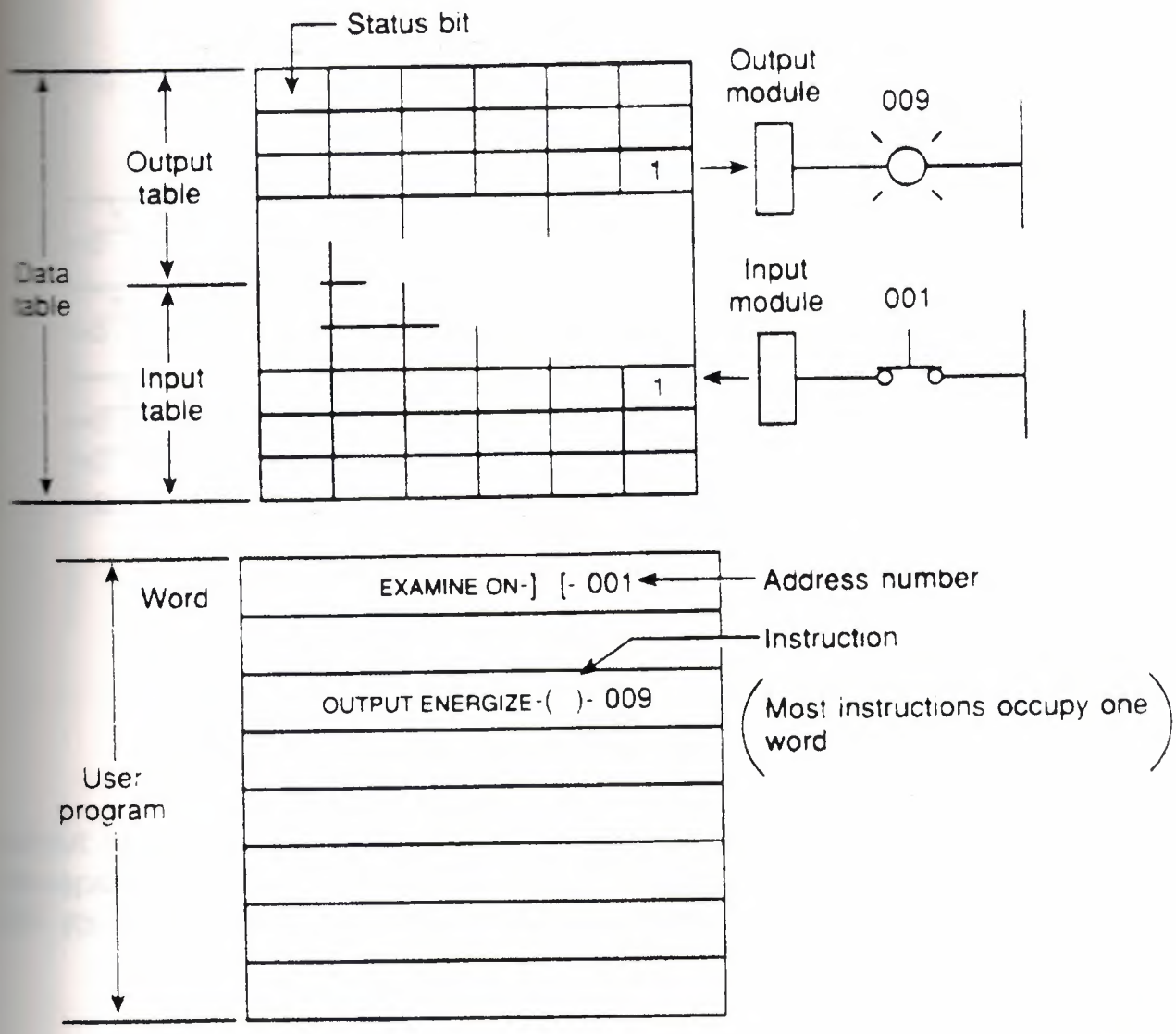


Fig. 27
Data table

information bay storing a binariy 1 in the proper bitlocation. Each connected input has a bit in the input image table that corresponds exactly to the terminal to wich the input is connected.

The input image table is constantly being changed to reflect the current status of the switch. If the input is ON (Switch closed), its corresponding bit in the table is set to 1. If the input is OFF (Switch open) the corresponding bit is "cleared," or reset to 0.

The output image table is an array of bits that controls the status of digital output devices, which are connected to output interface circuits. (Figure 29) shows a typical connection of lights to the output image table through the output module. The status of the lights (ON/OFF) is controlled by the user program and indicated by the presence of ıs (ON) and üs (OFF). Each connected output has a bit in the output image table that corresponds exactly to the terminal to wich the

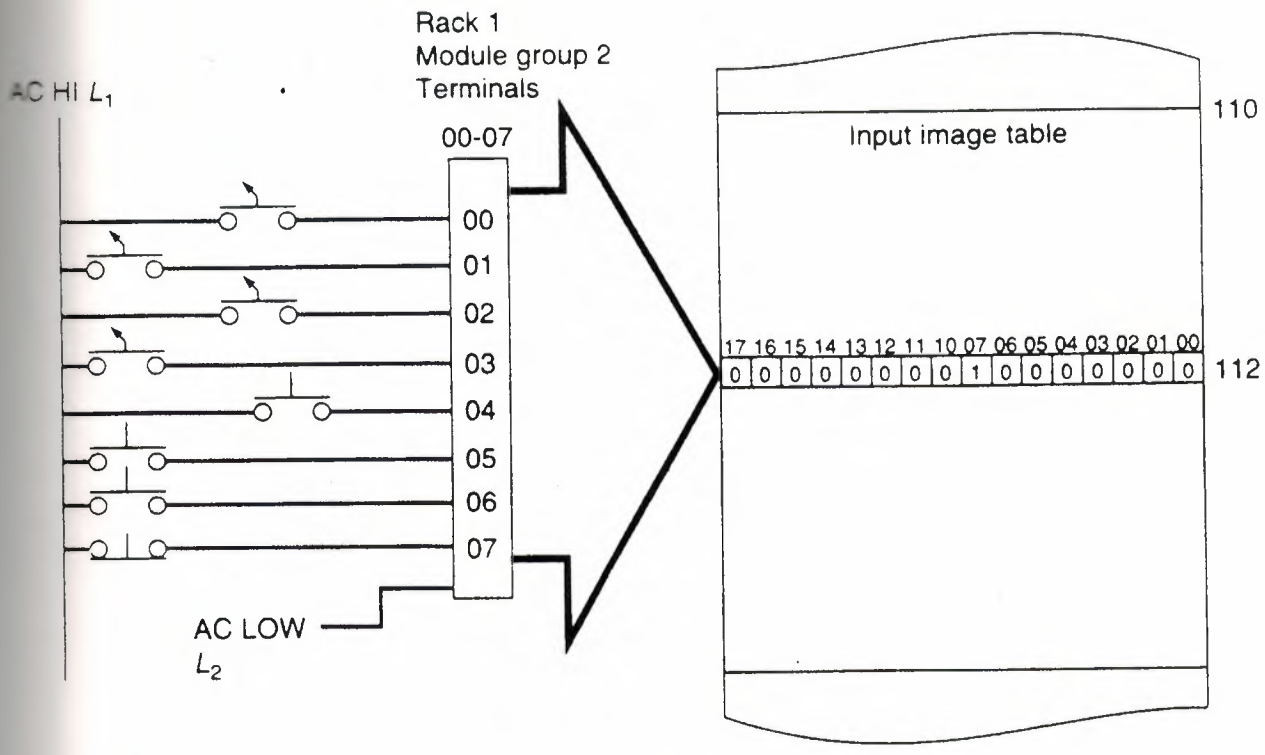


Fig. 28
Input image table

output is connected. If the program calls for a specific output to be ON, its corresponding bit in the table is set to 1. If the program calls forr the output to be Off, its corresponding bit in the table is set to 0.

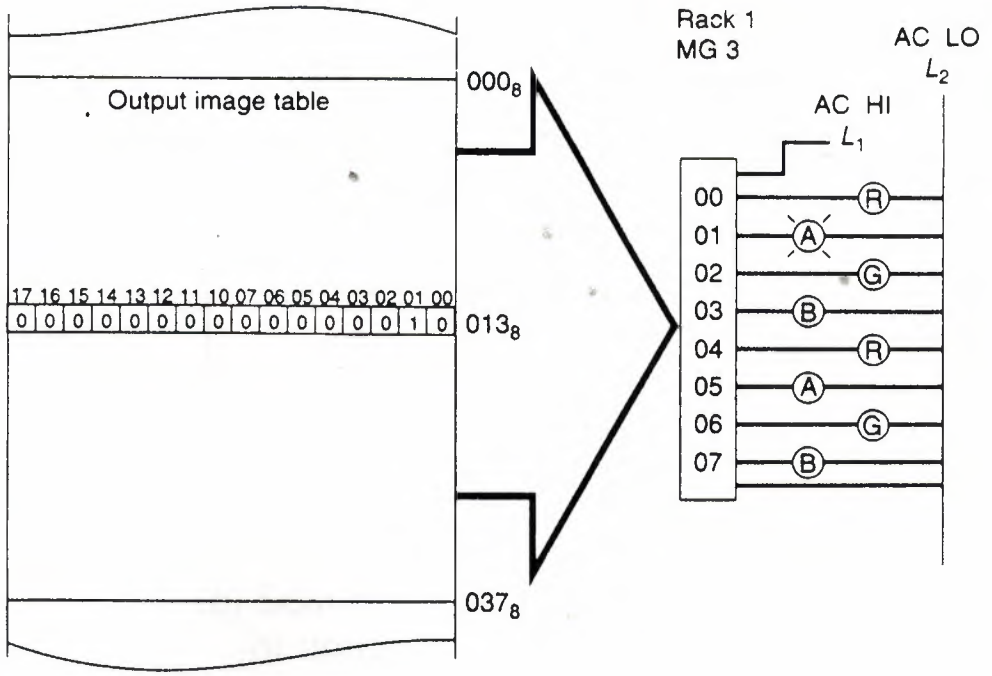
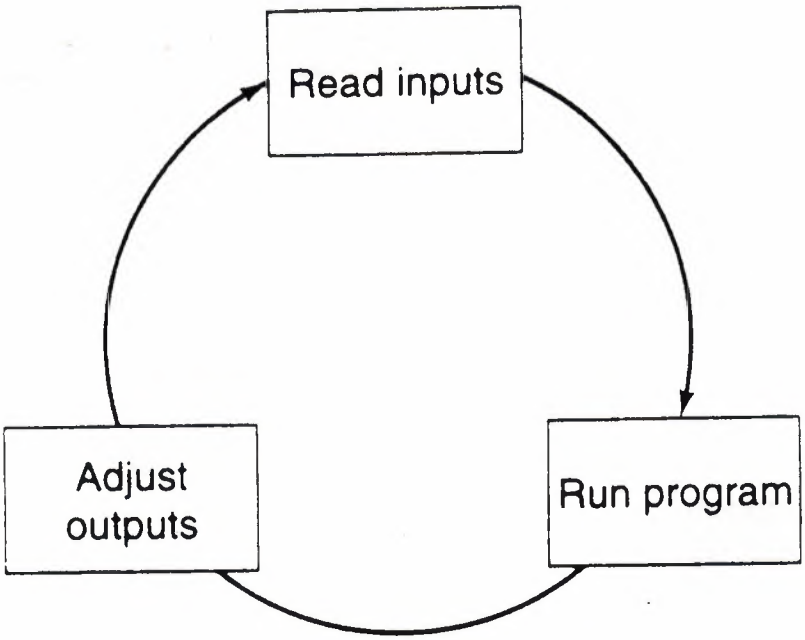
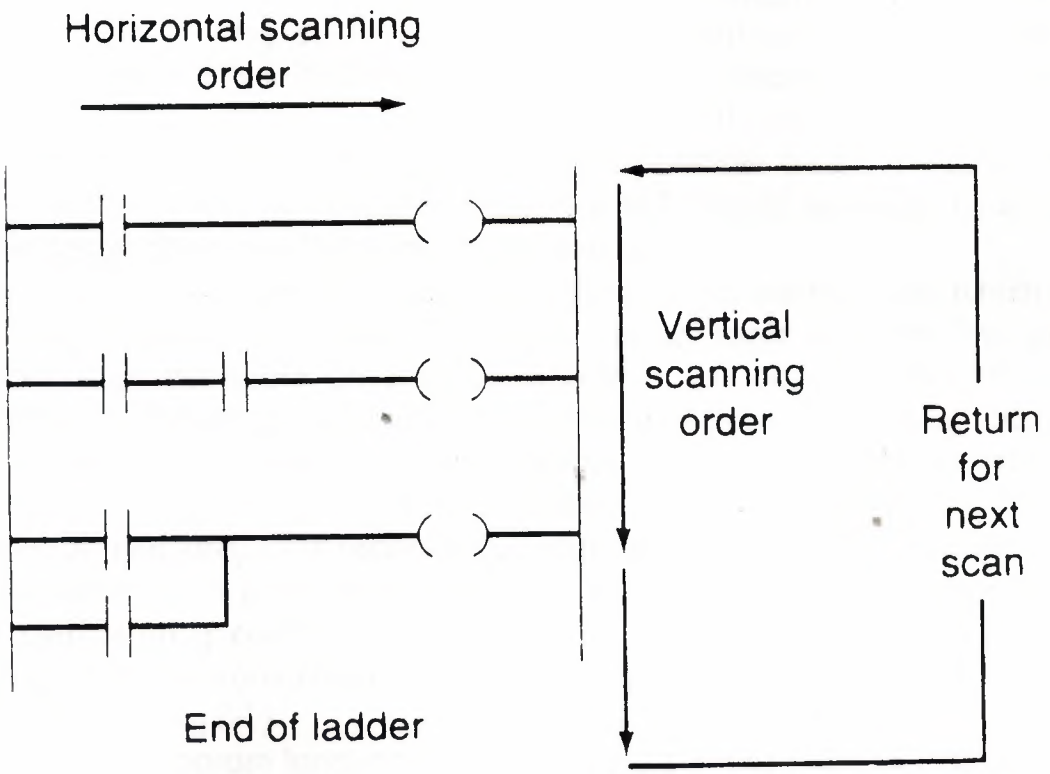


Fig. 29
Output image table



(*a*) Typical scan cycle



(*b*) Scanning can be vertical or horizontal

Fig. 30
Scan sequence

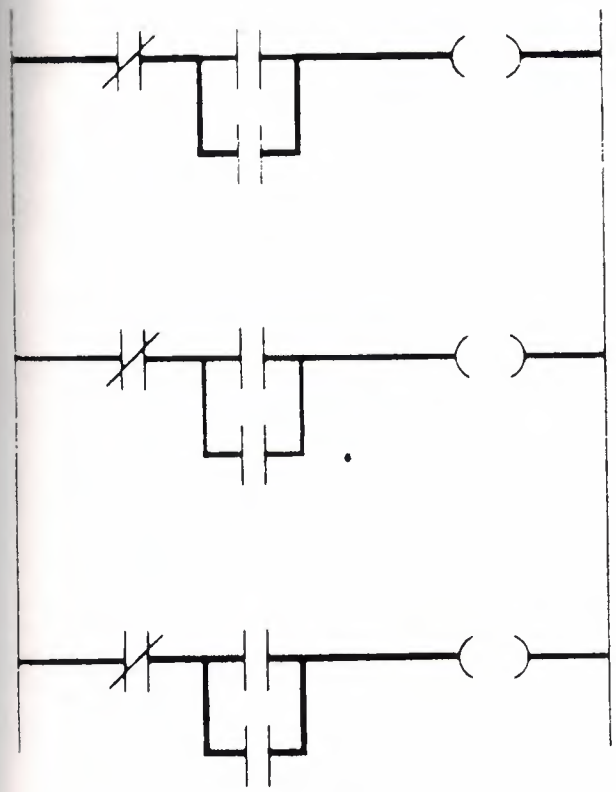


Fig. 31
Monitoring a relay ladder logic diyagram

During each operating cycle, the processor reads all the inputs,takes these values, and according to the user program energizes or deenergizes the outputs.This process is knojn asa scan.

The scan is normally a continuous and seguential process of readiing the status of inputs, evaluating the control logic,and updating the outputs.(Figure 30 İllus trates this process.A Scan its inputs and generate appropriate control responses at its outputs.Scan time varies with program content and length. A scan can take from about 1 to 20 ms. If a controler has to react to an ihput signal that changes states tji states tjice during the scan time, it is possible that the PLC willnever be able to detect this change. The scan time of a PLC should be knojn to ensure that scanning is faster than any field device operation.

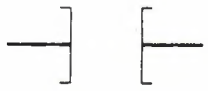
The term PLC programming language refers tu the method bay which the user communicates information to the PLC. Relay ladder logic was the first and most popular language available on the PLC, and it is still popular. Most PLCs on the market today can be programed either exclusively or partially in relay ladder logic.

Relay ladder logic is a graphical programming language designed to closely represent teh apperance of a wrid relay system. İt offers considerable advantages for PLC control. Not only is it reasonably intuitive, especially for techniscians with relay experience, it is particularly effective in an on-line mode when the PLC is actually performing control, Operation of the logic is apparent from the highlighting of the various relay contacts and coils on screen, identifying the logic state in real time (Fig 31)

The ladder diagram language is basically a symbolic set of instructions used to create the controller program. These ladder instruction symbols are arranged to obtian the desired contrl logic that be entered into the memory of teh PLC . Because the instruction set is composed of contact symbols lader diagram language is also referred to as contact symbology. Representations of contacts and coils are the basici saymbols of the logic ladder diagram instruction set (Fig.32)

A. EXAMINE ON instruction.

Symbol



Typically represents any input to control logic.

The input can be a connected switch or pushbutton, a contact from a connected output, or a contact from an internal output.

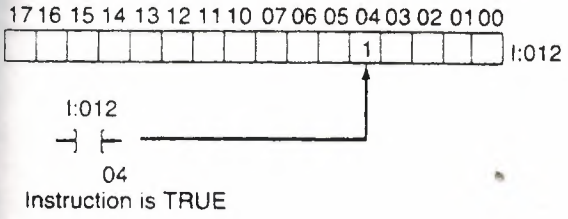
Has a bit-level address.

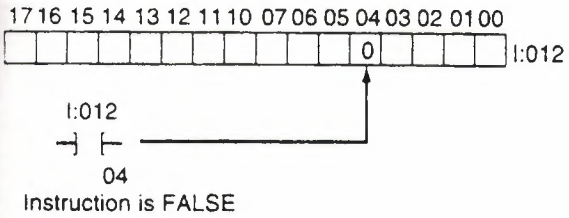
The status bit will be either 1 (ON) or 0 (OFF).

The status bit is examined for an ON condition.

If the status bit is 1 (ON), then the instruction is TRUE.

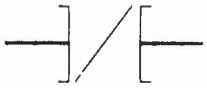
If the status bit is 0 (OFF), then the instructon is FALSE.





B. EXAMINE OFF instruction.

Symbol



Typically represents any input to the control logic.

The input can be a connected switch or pushbutton, a contact from a connected output, or a contact from an internal output.

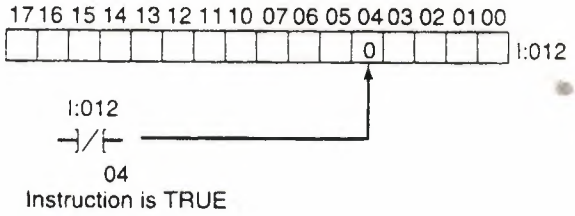
Has a bit-level address.

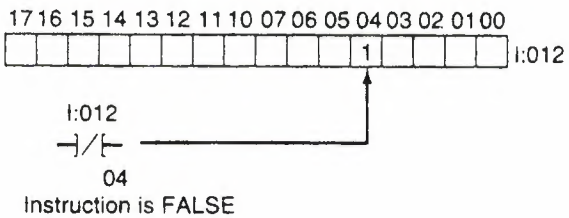
The status bit will be either 1 (ON) or 0 (OFF).

The status bit is examined for an OFF condition.

If the status bit is 0 (OFF), then the instruction is TRUE.

If the status bit is 1 (ON), then the instruction is FALSE.

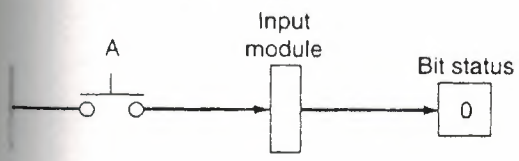


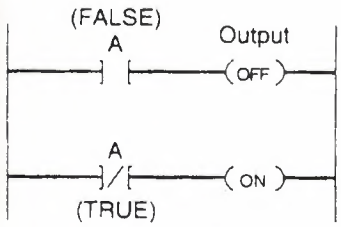


(*Continued on next page.*)

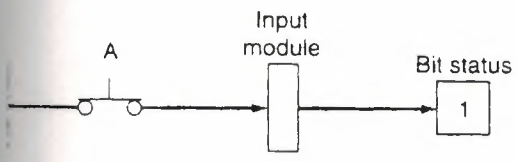
Fig. 32
(Continued)

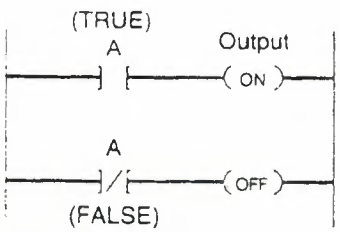
C. Status bit examples





Button not actuated





Button actuated

D. OUTPUT ENERGIZE

Symbol

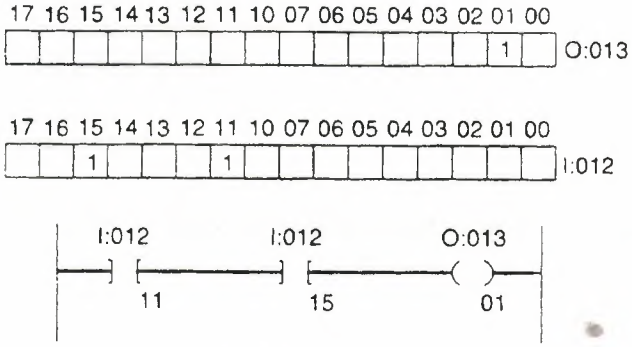
Typically represents any output that is controlled by some combination of input logic.

An output can be a connected device or an internal output (internal relay).

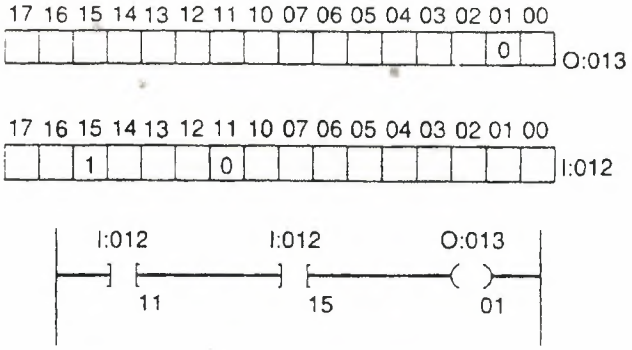
If any left-to-right path of input conditions is TRUE, the output is energized (turned ON).

The status bit of the addressed OUTPUT ENERGIZE instruction is set to 1 (ON) when the rung is TRUE.

The status bit of the addressed OUTPUT ENERGIZE instruction is reset to 0 (OFF) when the rung is FALSE.

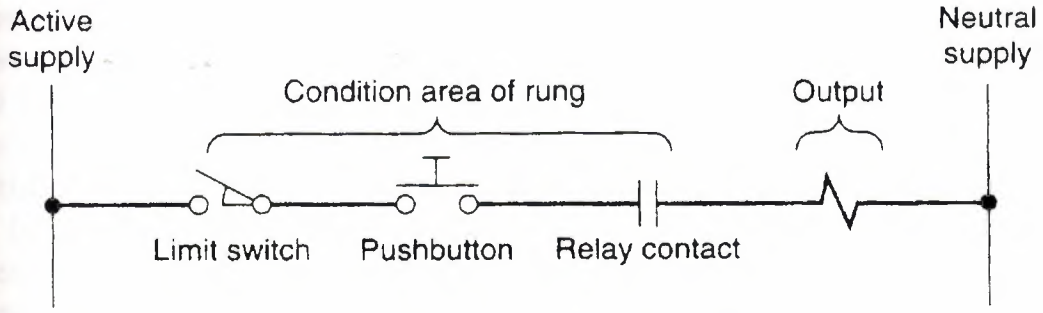


OUTPUT ENERGIZE instruction—TRUE

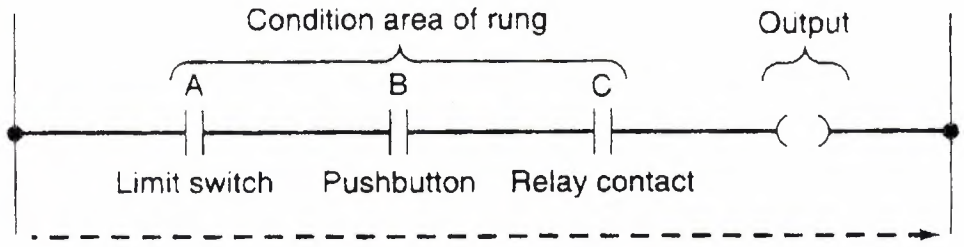


OUTPUT ENERGIZE instruction—FALSE

Fig. 32a
Basic set of instructions that perform functions similar to relay functions



(*a*) Hard-wired format



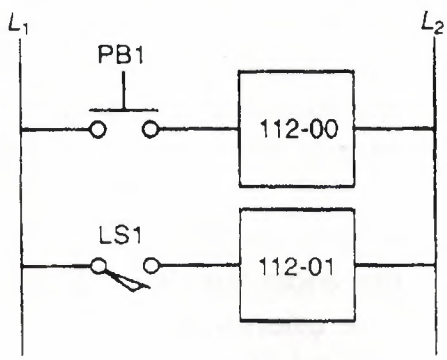
Continuous path is required for logic continuity and to energize output

(*b*) PLC ladder diagram format

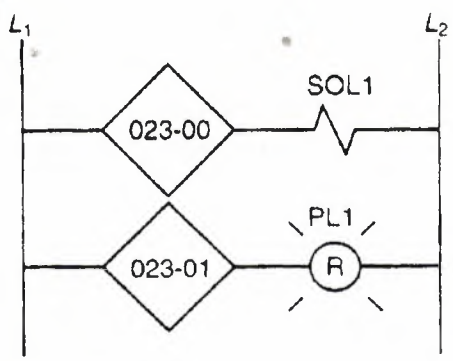
Fig. 33
Ladder rung

The following three are the basic symbols used to translate relay control logic to contact symbolic logic:

Instruction	symbol	Mnemonic
EXAMINE ON		XIC
EXAMINE OFF		XIO
OUTPUT ENERGIZE		OTE



(*a*) Input connections



(*b*) Output connections

Fig. 34
I/O connection diagram

The main function of the logic ladder diagram program is to control outputs based on input conditions. This control is accomplished through the use of what is referred to as a ladder rung. In general, a rung consists of a set of input conditions,represented by contact instructions, and an output instruction at the end of the rung represented bay the coil symbol (Fig 33) Each contact or coil symbol is referenced with an address number that identifies what is bein evaluated and what is being controlled. The same contact instructioin can be used throughout the program whenever that condition needs to be evaluated. For an output to be activated or energized, at least one left-to-right path of contacts must be closed. O complete closed path is referred to as having logic continuity. When logic continuity exists in at least one path the rung condition is said to be TRUE, The rung condition is FALSE If no path has continuity.

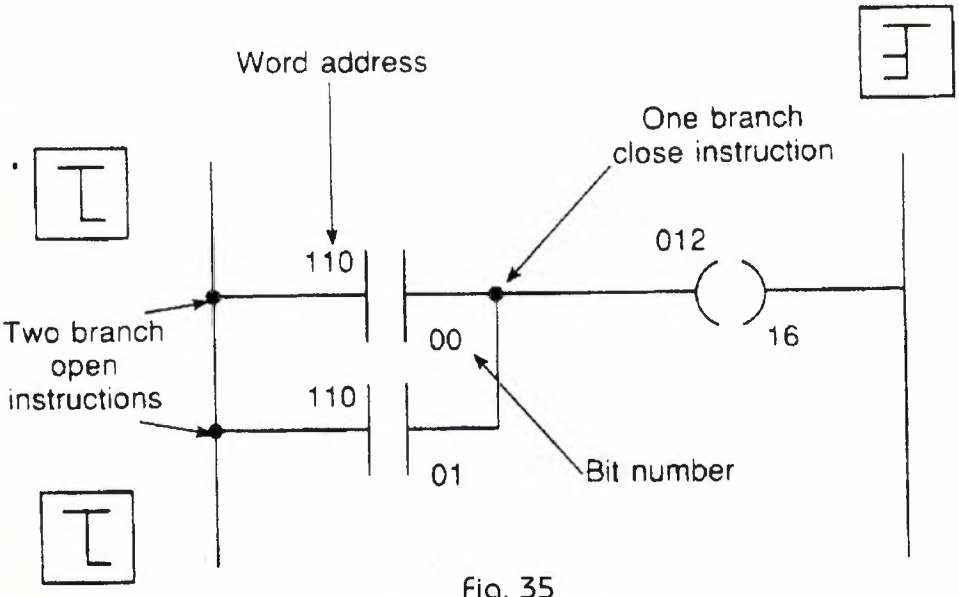
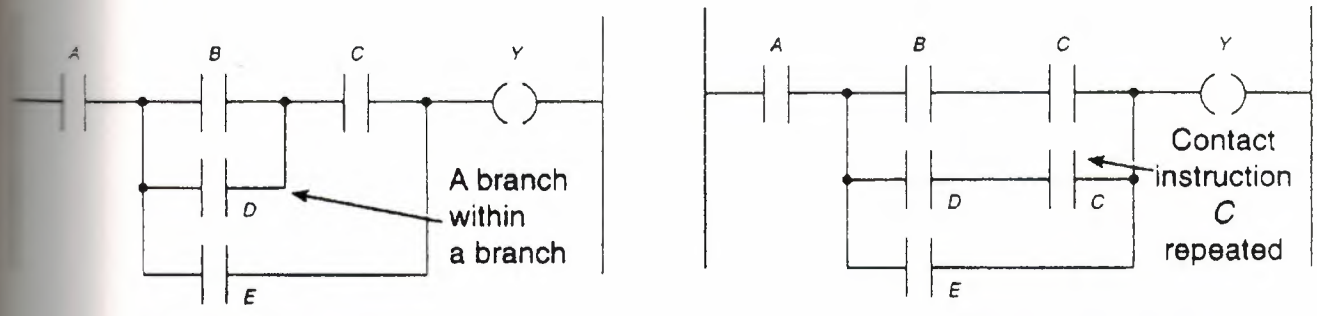


Fig. 35
Parallel path instructions.

To complete the entry of a relay-type instruction,yu must assign an addresse number to it.This munber will indicate what PLC inputis connected to what input device and what PLC output will drive what output device. The assignment of 1/O address is some times included in the 1/0 connection diagram as shown in (Fig 34) inputs and outputs are typically represented buy sguares and diamonds, respectivelу.

Branch instructions are used to create paralel paths of input condition instructions.This allows more than one combination of input conditions (OR Logic) tu establish logic continuity in a rung. (Figure 35) illustrates o simple branching condition. TKhe rung will be TRUE A branch START instruction is used to beğin each paralel logic brancl. A single branch CLOSE instruction is used to close the parallel branch.

In some PLC models the programming of a branch circuit within a brnanch



(*a*) Nested branch (*b*) Eliminated nested branch

Fig. 36
Nested branch

circuit o a nested barnch cüannot be done directly. İt is possible, however, to program a logically eguivalent Programming morğe than the allowable series elements or parallel branches will resultin an error massağe. Also there is a further limitation wihth some PLCs:there can be onlay one output per rung. and the output must belocated on the right-hand end of a rung.

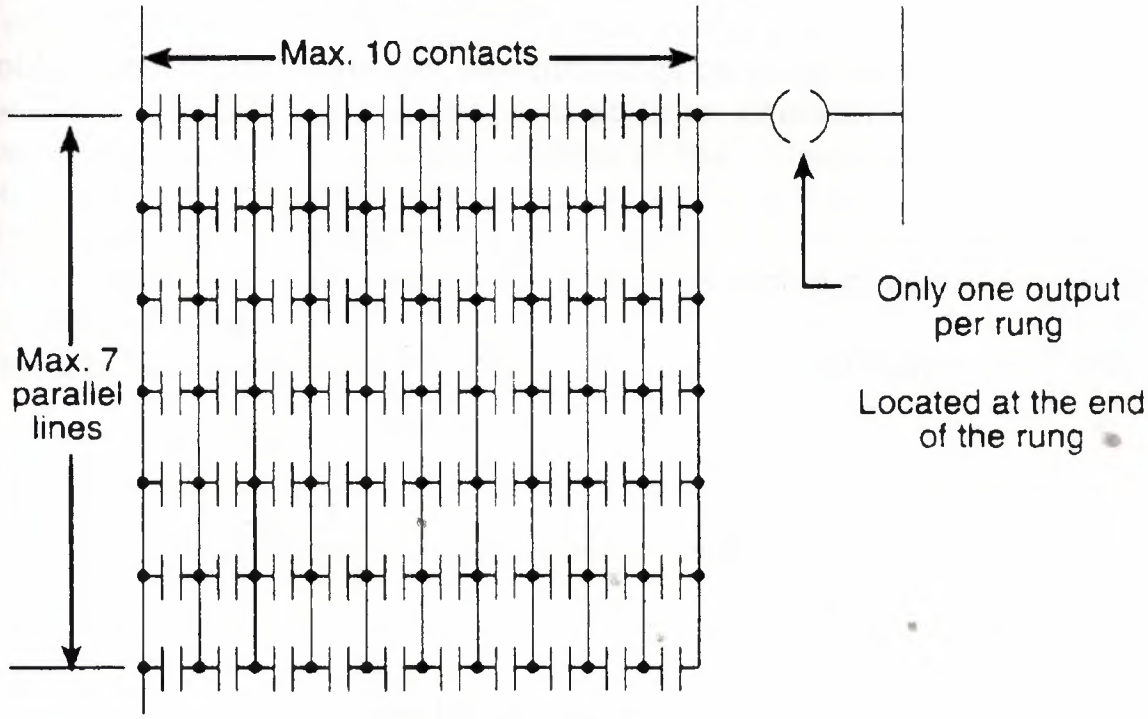


Fig. 37
Typical PLC matrix limitation diagram

Most PLCs have an areo of the memoruy allocated for what are known as internal storage bits.These storage bits are also called internal outpus, internal coils, intenal The internal output operates just as any output that is controlled by programmed logic; however, the output is used strictly fofinternal output doesnot directly control an output device.

An internal control relay can be used when a circuit reguires more series

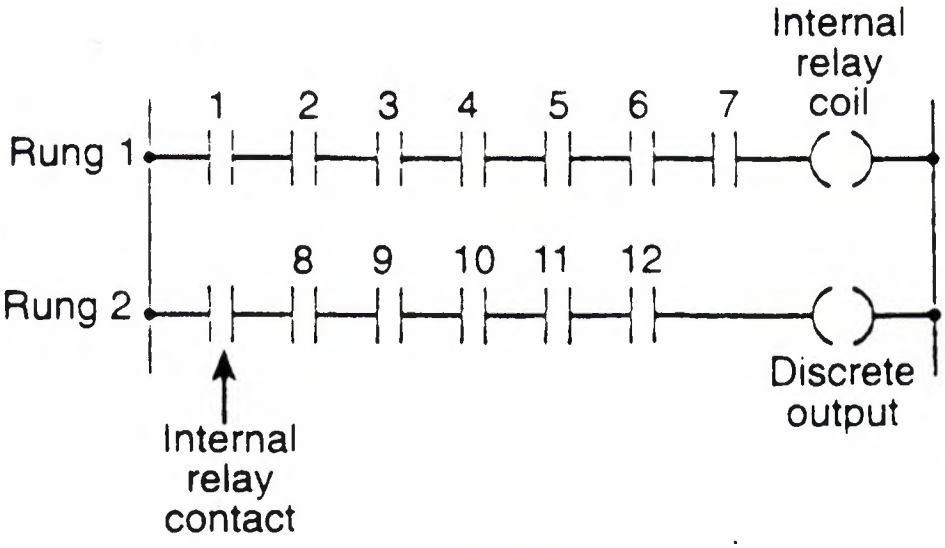
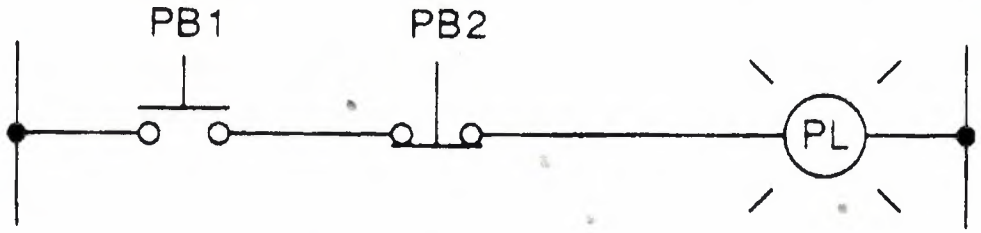


Fig. 38
Internal control relay

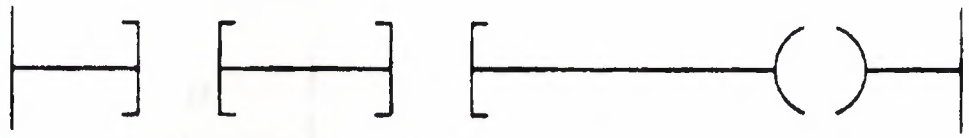
contacts than the rung allows (Figure 38) shows a circuit that allowjs for only 7 series contacts when 12 are actually reguired for the programmed logic. To solve this problem, the contacts are split into two rungs as shown. The first rung contains 7 of the reguired contacts and is programmed to an internal relay. The address of the internal relay would also be the address of the internal raley would also be the address of thi first EXAMINE ON contact on the second rung.

The remaingn 5 contacts are programmed, followed by the discrete output.The advantage of an internal storage ibt is that it does not unnecessarily ocupy a physical output.

A simple program using the EXAMINE ON instruction is shown in (Fig. 39) This

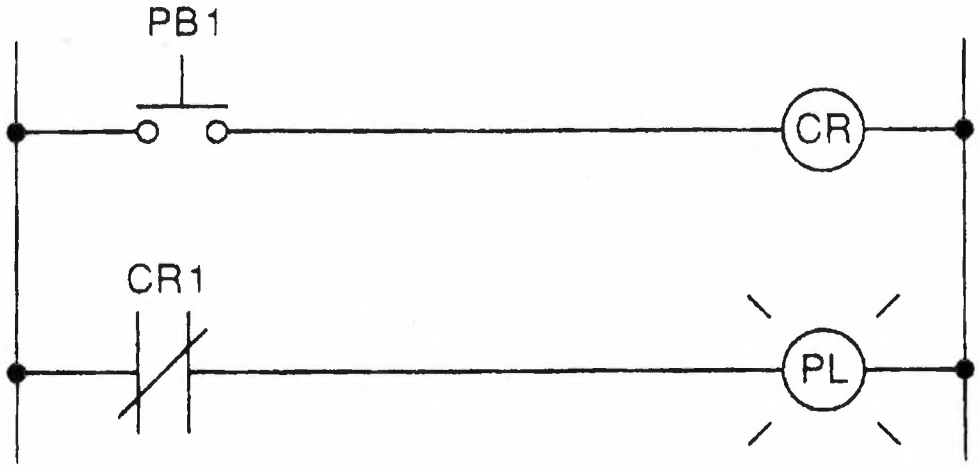


(*a*) Hard-wired circuit



(*b*) User program providing the same results

Fig. 39
Simple program using the examine on instruction



(*a*) Hard-wired circuit

(*b*) User program providing the same results

Fig. 40
Simple program using the examine off instruction

Figure shows a hard-wired circuit and a user program that provides the same results.You will note that both the NO and the NC pushbuttons are represented bay the EXAMINE ON symbol.

This is because the normal state of an input (NO or NC) doesnot matter to the controler.

What does matter is that output, then the EXAMINE ON instruction is used. Since both PB1 and PB2 mustbe closed to energize the pilot light, the EXAMINE ON

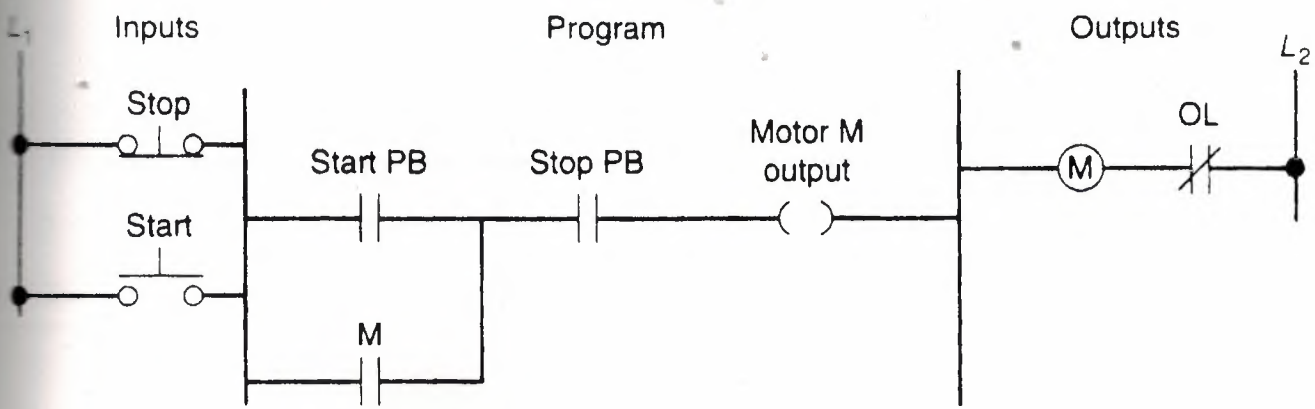
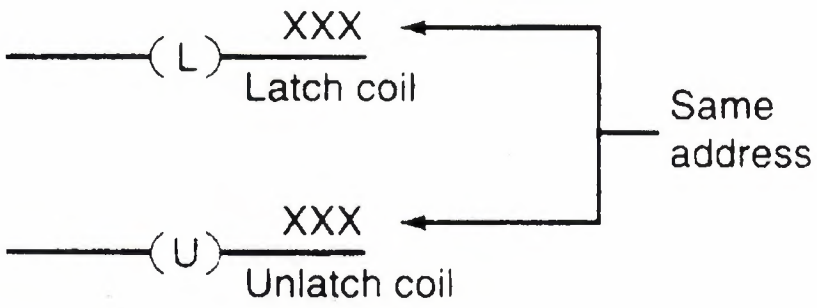
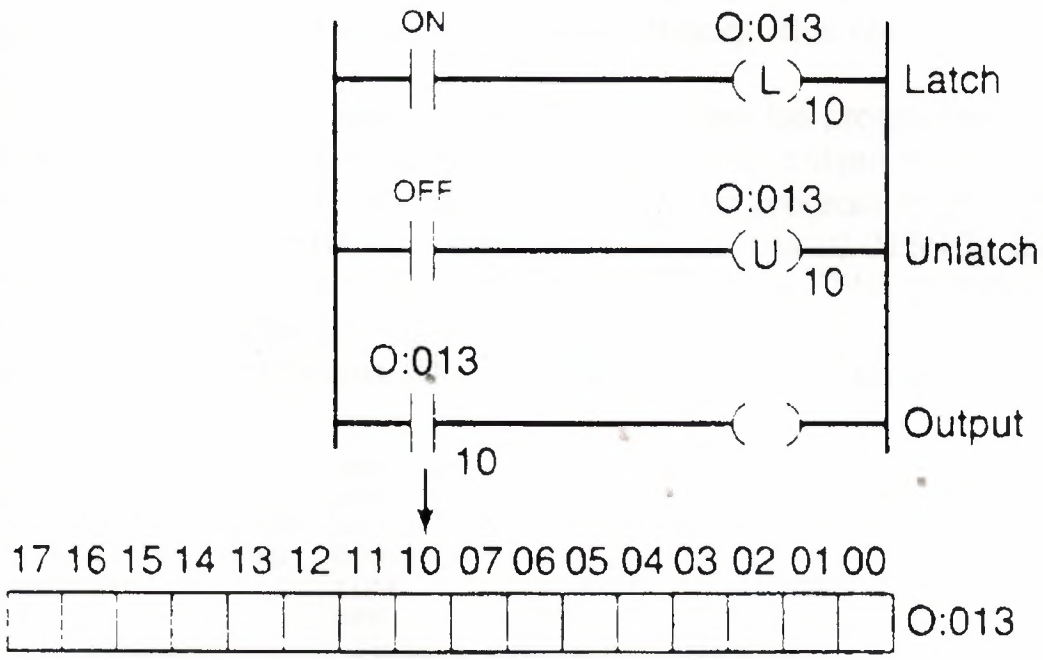


Fig. 41
Start/stop program

Instruction	Symbol	Mnemonic
Output latch	-(L)-	OTL
Output unlatch	-(U)-	OTU



(a) Latch and unlatch coils have same address



(*b*) Control logic

Fig. 42
Output latch and output unlatch instructions

instruction is used for both.

A simple program using the EXAMINE OFF instruction is shown in (Fig 40) Again both the hard-wired circuit and user program are shown,in the hard-wired circuit, when the pushbutton is open, Relay, coil CR is deenergized an d contacts CR 1 close to siwitch the pilot light on. When the poshbuton is closed, relay coil CR is energized, and contacts CR 1 open to switch the pilot light OFF. The pushbutton is represented in the user program by an EXAMINE OFF instructionl. This is because the rung must be TRUE when the external pushbutton is open, and false jwhen the pushbutton is closed. Using an EXAMINE OFF instruction to represent the pushbutton satisfies these reguirements. The No or NC mechanical action of the pushbuton is not a considerationl. It is important to remember that the user program is not an electirical circuit buta logic circuit. In effect we are interested in logic continuity when establishing an output.

The START/STOP program shown in (Fig.41) can be used to start or stop a motor, or process. Note thatthe stop pushbuttoninput is programmed as an EXAMINE ON instruction. This is because the stop pushbotton is wired normally closed. Since the start pushbutton (Normally open) is a momentary device (allows continuity only when closed) a contact from the output coil is used to seal in the circuit. Often,the seal-in contact is an input from the motor starter auxiliary contacts.

That electromagnetic latching relays are used where it is necessary for contacts to stay open and/or closed even though the coil is energized only momentarily.

An electromagnetic latching relay function can be programmed on a PLC to work lise its real-world counterpart.The use of the output LATCH and output UNLATCH coil instructions is illustrated in the ladder program of (Fig 42) Both the latch (L) and the unlatch (U) coil have the same address (0:013/10), When the On button is momentarily actuated, the latch rung becomes TRUE and the latch status bit (10) isset to 1. andso the output is switched ON. This status bit will remain set to 1 when logical cointinuity of the latch rung is lost. When the unlatch rung

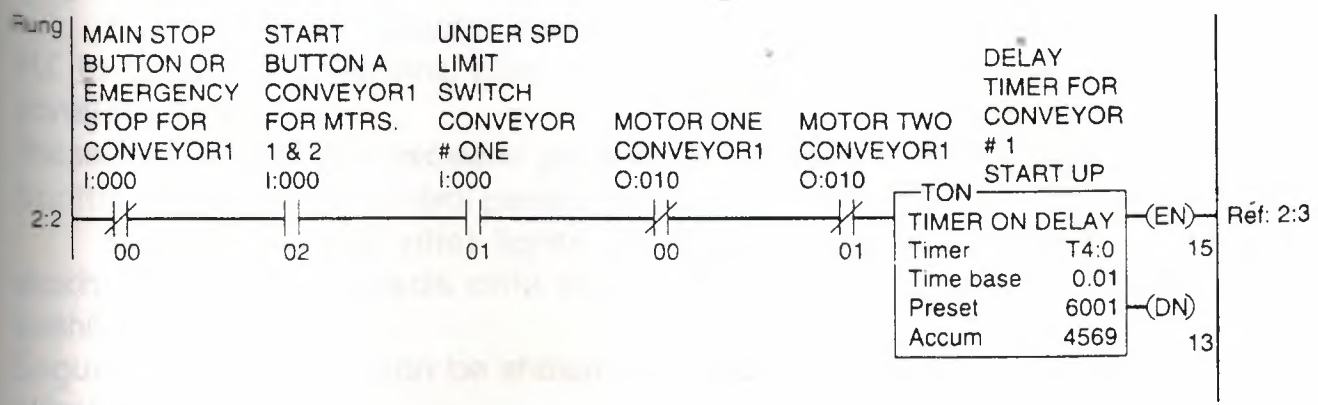


Fig. 43
PLC documentation.

becomes TRUE (OFF button Actuated), the status bit (10) is reset to 0 ond so the output is siwitched OFF. The LATCH/UNLATCH instruction is retentive on power lost. Thet is to say. İf the relay is latched, it will remain latched if power is lost and then restored.

The method of entergn a program is throuğgh the operator terminal keyborad or handheld programming device. The ladder diagram is entered by pushing keys on the key board in a prescibed seguence. The results are displayed on either the CRT of a desktop programer or wihth an LED or LCD for a hadhed programmerl.Because hardware and programming technigues varay uwiht ecah manufacturer, it is necessary to refer to the programming manual for a specific PLC to determine how the instructions are entered.

The programming devica can also be used to select the varioufs processor modes of operation. Again, the number of different operating modes and the method of accessing them varies wiht the manufacturer. Regardless of PLC model,some comman operatign modes are CLEAR MEMORY, PROGRAM,TEST, and RUN.

Mode Description

CLEAR MEMORY Used to erase tha contents of teh on-board RAM memory.

PROGRAM Used tu enter a new program or update an existing one in the internal RAM memory

TEST Used to operate or monitor the user program withhout energizing any outputs

RUN Used to execute the user program

Input devices are monitored and output devices are energized accordingly.

After you enter the rungs in your ladder program, you may want to document what the instrucion or rung does for someone else who may be maintainin the program. The use of personal computers and programable controler manufacturer software allows guality documentation to be produced (Fig 43).

Documentation may include rung, instruction, and address comments as well as rung cross references where repeatedlyused addresses are cross-referenced to rung numbers.

A programmable graphic interface can be connected to communicate With a PLC to replace pushbuttons, pilot lights, thumbwheels, and other operator control panel devices

These luminescent touchscreens provide an operator interface that operates like traditional hard-wired control panels. Features include:

Pushbuttons and pilot lights are replaced by realistic-looking icoıns.The machine operator needs only to tocuh the display panel to activate the pushbuttons.

Seguential operations can be shown in graphich format for easier wiewing of the operation.

The operator can change timer and counter presets buy touching the numeric keypad graphic on the touchscreen.

The screen can show alarms, complete with time of occurrence and location. Variables can be displayed as they change over time.

TIMER AND COUNTER INSTRUCTION

The advantage of PLC timers and counters is that their settings can be easily altered, or the number of them used in a circuit can bi increased or decreased, by programming changes without wiring changes. Counter and timer addresses are usually specified by the programmable controller manUfacturer and are located in a specific area of the data organization table.The number of timers and countres that can be programmed depends upon the model of PLC you are using. However, the availability usually far exceeds the reguirement.

Timers and counters are output instructions.

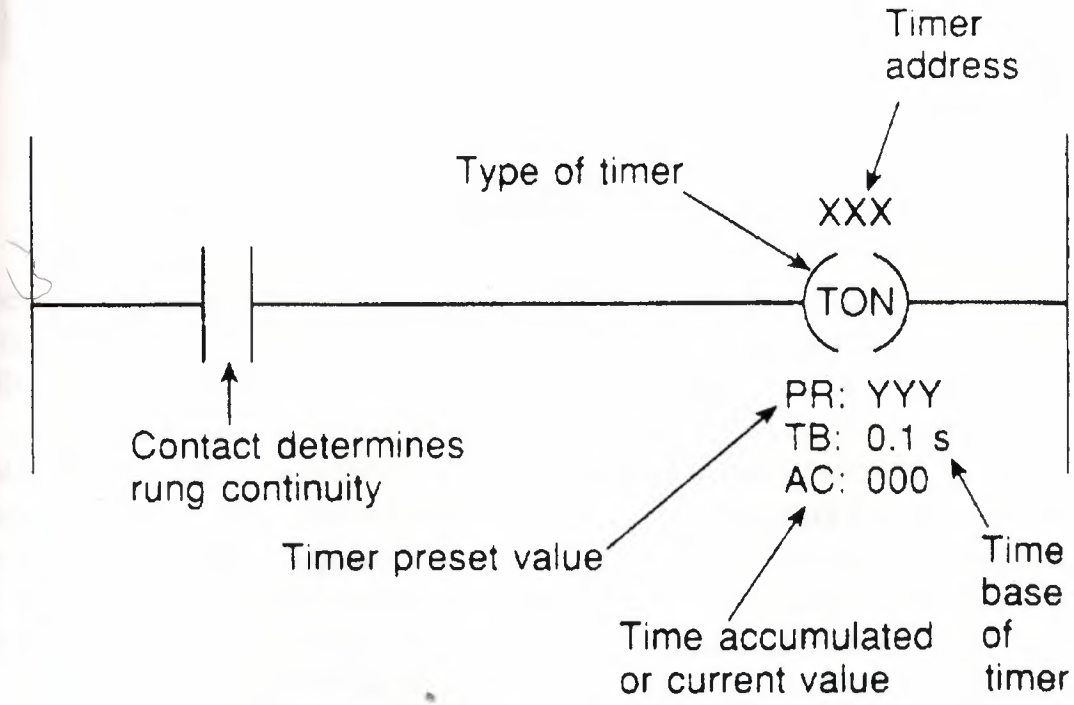


Fig. 44
Coil-formatted timer instructions

In general,there are three different timers: the on-delay timer (TON),the off-delay timer (TOF), and the retentive timer on(RTO).

There are two tifferent counter instructionsthe count-up counter (CTU) and the countdown counter (CTD).A reset instruction is reguired to reset both retentive timer instructions and the counter instructions.

PLC timers provide the same functions as mechanical and electronic timing relays. that timers are used to activate or de-activate a device after o preset interval of time.

There are two methods used to represnet o timer instruction withn a PLC's

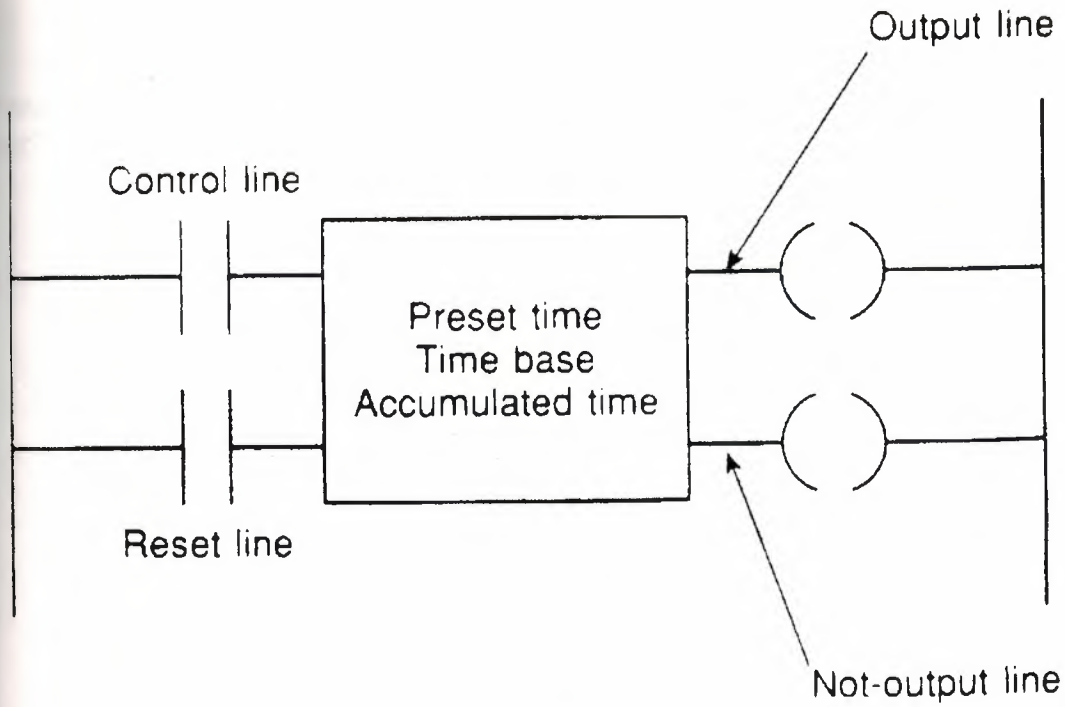
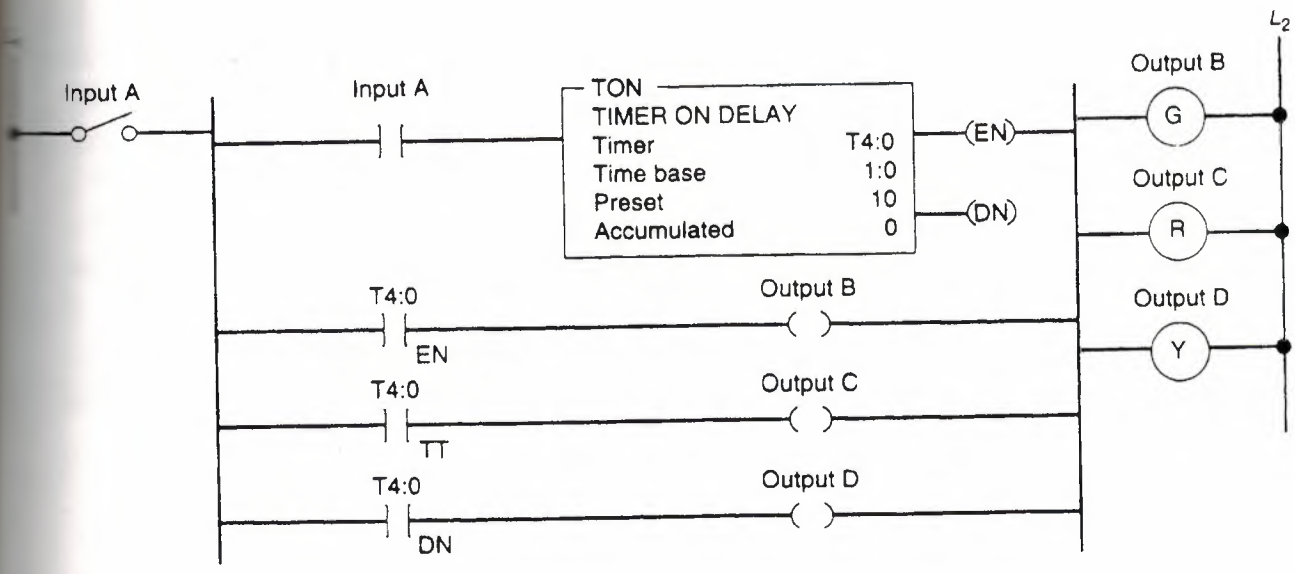


Fig. 45
Block-formatted timer instructions

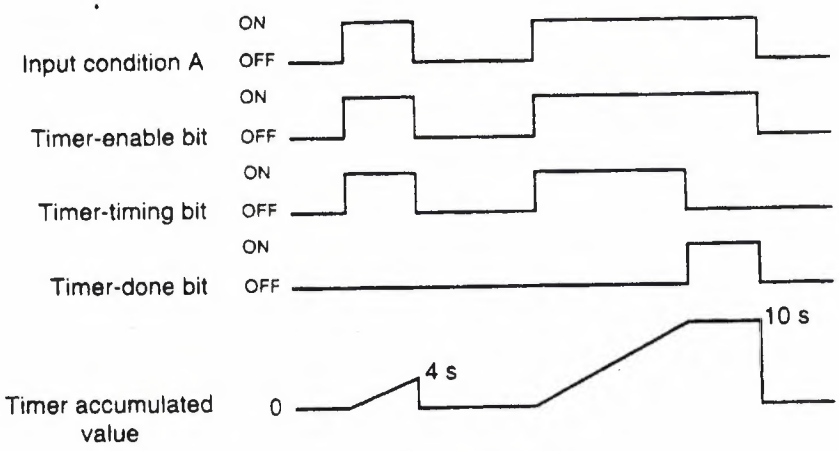
ogic ladder program.The first depicts the timer instruction as a relay coil similar to hat illustradet in (Fig 44) The timer is assigned an address as well as being dentified as a timer. Also included as part of the timer instruction is the time base of the timer the timer's preset value or timedelay period, and the accumulated value or current time-delay period for the timer. When thi timer rung has logic continuity, the timer begins counting time-based intervals and times until the accumulated time eguals the preset time, the output is energized and the timed output contact associadet with the output is closed. The timed contact can be used throughut the program as a NO or NC contact as many times as you wish.

The second timer format is referred to as a block format. (Fig 45) İllustrates a generic block format. The timer black has two input conditions associated wihth it:the control/and reset. The control line controls the actual timing operation of the timer.The reset line resets the timer's accumulated value to zero. Ali block-formatted timers provide at least one output signal from the timer. When a singleoutput is provided. İt is used to signal the completion of the timing cycle. For dual-output timer instructions, the second output signal operates in the reverse mode. Whenever the timer has not reached its timed-out state, the second output is on And the first output remains OFF. As soon as the timer reaches its timed-out state, the second output is turned OFF and the first output is turned ON.

The on-delay timer(TON) is the most commonly used timer. The on-delay timer operatesso that when the rung containing the timer is true, the timer time-out period, an output is made active. The timed output becomes active sometime after the timer becomesacvtive;bence teh timer is said to have an ON delay, (Fig 46) shows aladder diagram containing o typical on-delay timer.The timer



(*a*) Ladder diagram



(*b*) Timing diagram

Fig. 46
On-delay timer

instruction conssits of there date tablewords:
the control word, the preset word,and the accumuladet word.

Control wordl.

The control word uses three bits: the enable bit (EN),the time-timing bit (TT) and the done bit (DN).

The enable bit is true (has status of 1) whenever the timer instruction is false, the enable bit is false (has status of 0)

The timer-timing bit is true whenever the accumulated value of the timer is changing, which means the timer is timing.

When the timer is not timing, the accumulated value is not changing,so the timertiming bit is false.

The done bit changes state whenever the accumulated value reaches the

reset value.

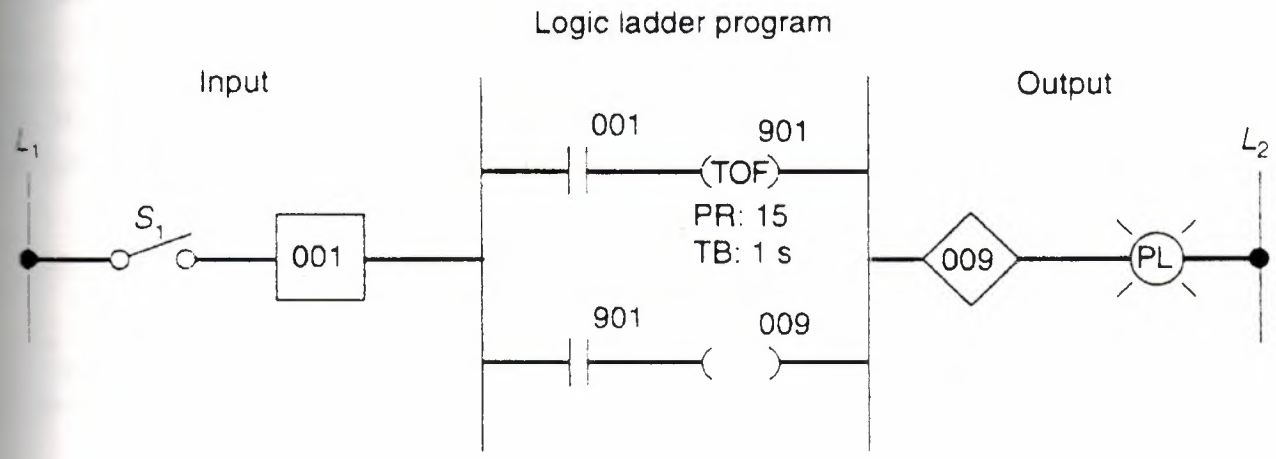
Preset World

The preset valueis the set pointof the timer, that is, the value up to which the timer will time.

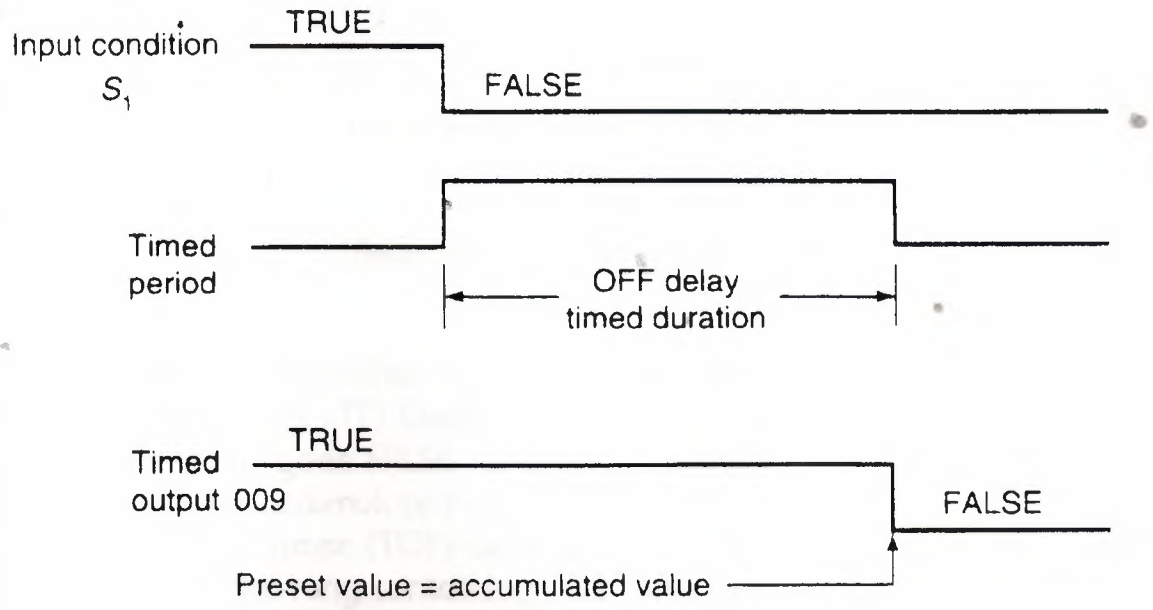
Accumulated world.

The accumulated value is the value that increments as thi timer is timing. The accumulated value will stop incrementign when its value reaches the preset value.

This timer instruction reguires you to enter a time base, which is either 1. 0 or 0.01 for long or short time delays. The actual preset time interval is the time



(*a*) Programmed circuit



(*b*) Timing diagram

Fig. 47
Off-delay programmed timer

base multiplied by the value stored in the timer's preset woird. The actual accumulated time interval is the time base multiplied bay the value stored in the timer's accumulated word. The timeris activated by closing the switch.

The preset time for this timer is 10 s,at which time output D will be energized. When the swich is closed, the timer begins countign, and counts untilthe accumulated time eguals the prest value; the output is then energized. If the switch is opened before the timer is timed out, the accumulated time is automaically reset to zerdo. This timer configuration is termed nonretentive since loss of power flow toi the timer causses the timer instruction tu reset. This timing operation is that of on ON-DELAY timer, since output D is switched on 10 s after the switch has been actuadet from the OFF to the ON position.

In (Fig 47) the timing diagram first shows the timer timing to 4 s and then going FALSE, The timer resets,and both the timertiming bit and the enable bit go FALSE. The accumulated value also resets to 0. Input A then goes TRUE Again and remains TRUE in execes of 10 s. When the accumulated value reaches 10 s. When

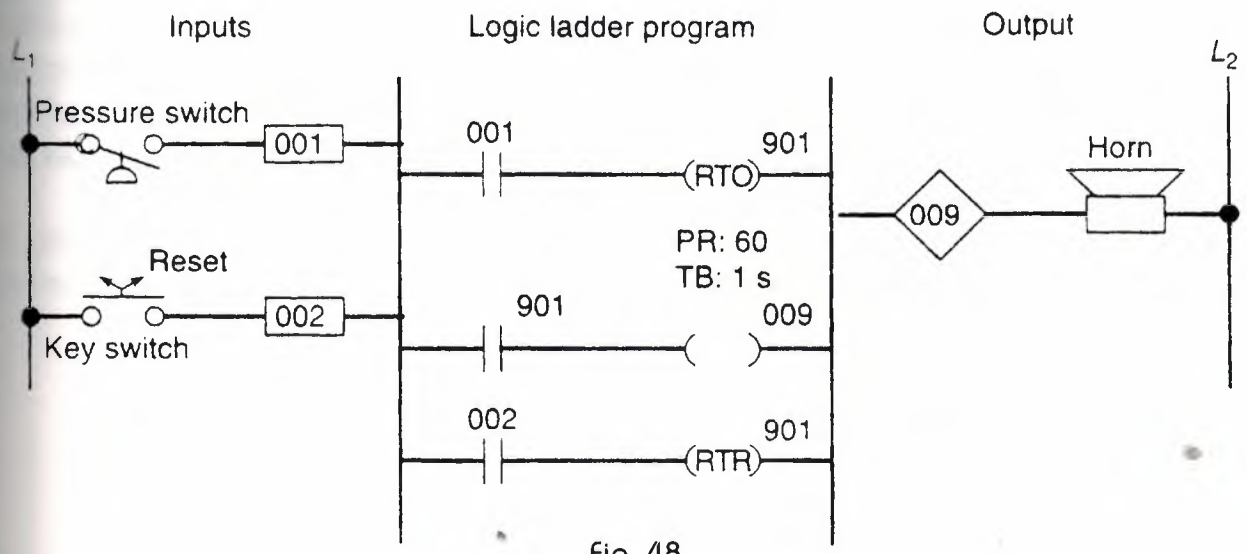


Fig. 48
Retentive on-delay alarm program

the accumulated value reaches 10 s. the done bit (DN) goes form FALSE toTRUE and the timer-timing bit (TT) Goes from TRUE to FALSE, When input A goes FALSE, the timer instruction goes FALSE and also resets, at which time the control bits are all reset and the accumulated value resets to 0.

The off-delay timer (TOF) operation will keep the output energized for a timed period after the rung containing the timer has gone false. The programming of an off-delay timer using the coil-formatted timer instruction. If logic continuity is lost,the timer begins counting time-based intervals until the accumaleted time eguals the programmed preset value. When the switch connected to input 001 is first closed, timed output 009 is set to 1 immediately and the lamp is switched

ON. If this switch is now opened, logic continuity is lost and the timer begins counting. After 15 s. when teh accumulated time eğuals kthe preset time, the output is reset tu 0 and the lamp switches OFF. If logic continuity is gained before the timer is timed out,the accumulated time is reset to zero. That is why this timer is classified as nonretentive.The PLC programmed tenetive on-delay timer (RTO) operates in the same ways as the nonretentive on-delay timer (TON) with one major exceptioin.This is a retentive timer reset (RTR) instruction.Unlike the TON, the RTO will hold its accumulated value when thi timer rung goes FALSE and willi continue timing here it left off whenthe timer rung goes TRUE again. This timer must be accompanied by a timer RESET İnstruction to resete the accumulated value of the timer to zero. The RTR instruction isthe only automatic menas of resetting the accumulated value of a retentive timer.The RTR instruction must be addressed to the same word as the RTO instruction. Normally, then the accumulated value of the referenced timer is reset to zero.

The program in ('Fig 48)shows a practical for on.RTO. The RTO timer detects whenever a piping system has sustained a cumulative overpressure condition of 60 s . At that point, a horn is automatically sounded to call attention to the malfunction. When alerterd, maintenance personnel can silence the alarm by switching the key switch s1 to the RESET (contact closed) position. After the problem has been corrected, the alarm system can be reactivated buy switching the key swiutch the key switch to the ON (contact open) position.

The counter instructions allow for the countingof events and the controlling of other events based on the accumulated count. Counters operate on transition rathen than length of time. Programmed counters can count up, count dojn, or be

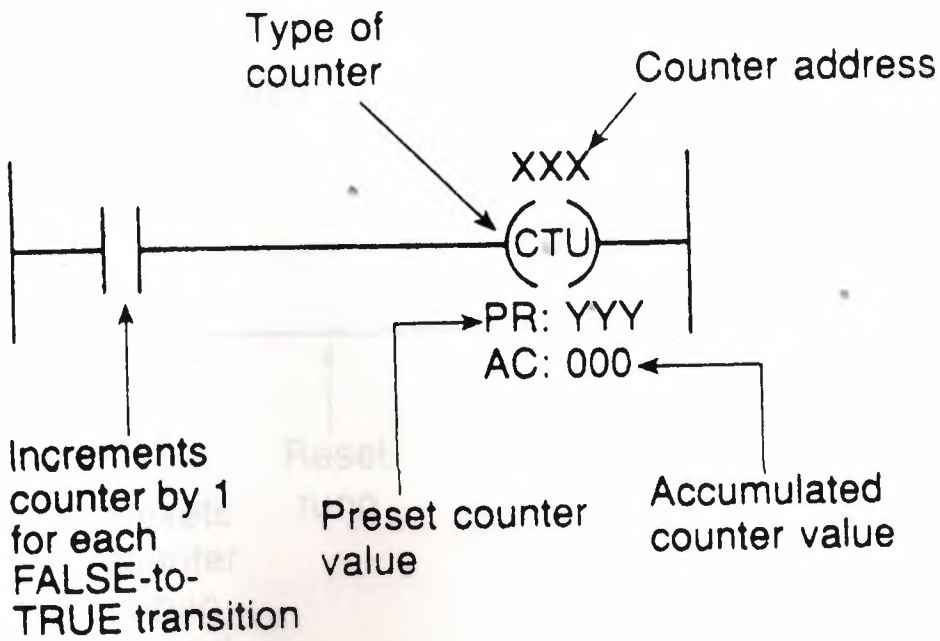


Fig. 49
Coil-formatted counter instruction.

combined tu count up and dojn and reguire a reset instruction.

PLC countershave programing formats similar to timer formats.A coil programming format, similar to that shown in (Fig.49)

İs used by some manufacturers. The counter is assigned an address as well asbeing identified as o counter.Alsoincluded as part of the counter instruction are the counter's preset value and the current accumaleted count for the counter.The up-counter increments its accumulated value by 1 each timethe counter run makes a FALSE to TRUE transition.

When the accumulated count eguals the preset conut,the output is energized, and the counter output is closed.The counter contact can be used throughout the programas an No or NC contact asmany you wish.

A counter reset instruction, which permits the counter to be reset, is also used in conjunction with the counter instruction. Up-Counters are always reset to zero. Downcounters may be reset to zero or to some preset value. Some manufacturers include the reset function as a part of the general counter instruction, while others dedicate a separete instruction for resetting of the counter. (Figure 50)

Shows a generic coil-formatted counter instruction with a separete instruction forresetting the counter. When programmed, the counter reset coil (TR) is given the same reference address as the counter (CTU) that it is to reset. The reset instruction is activated whenever the CTR rung condition is TRUE.

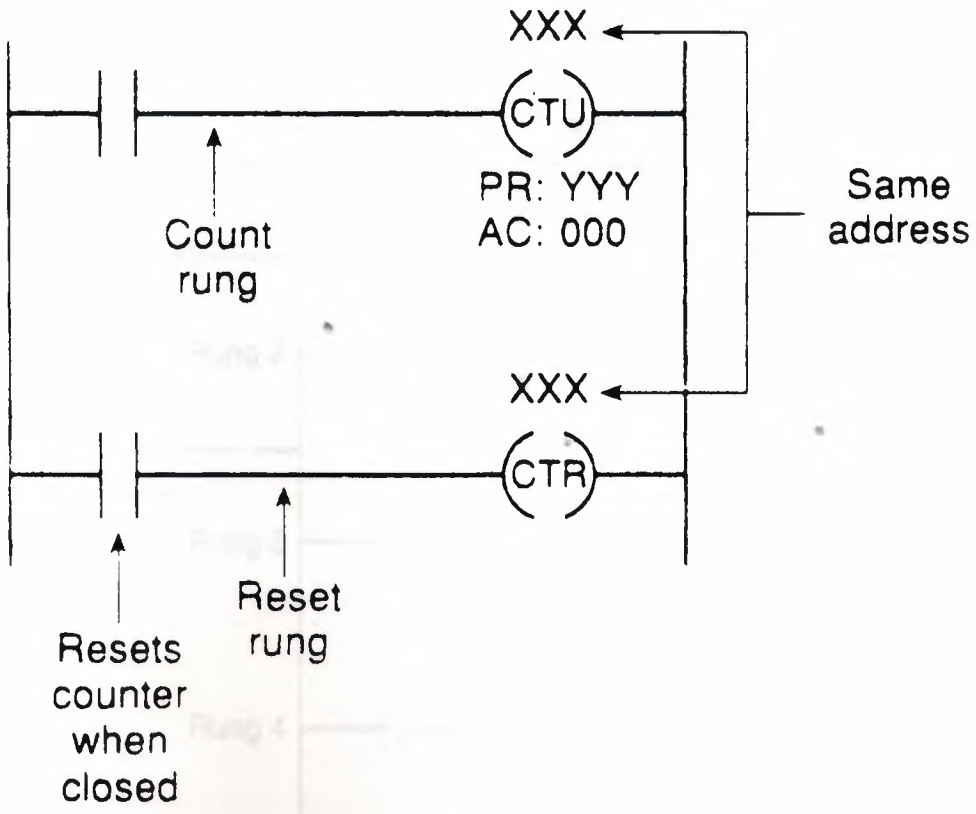


Fig. 50
Coil-formatted counter and reset instruction

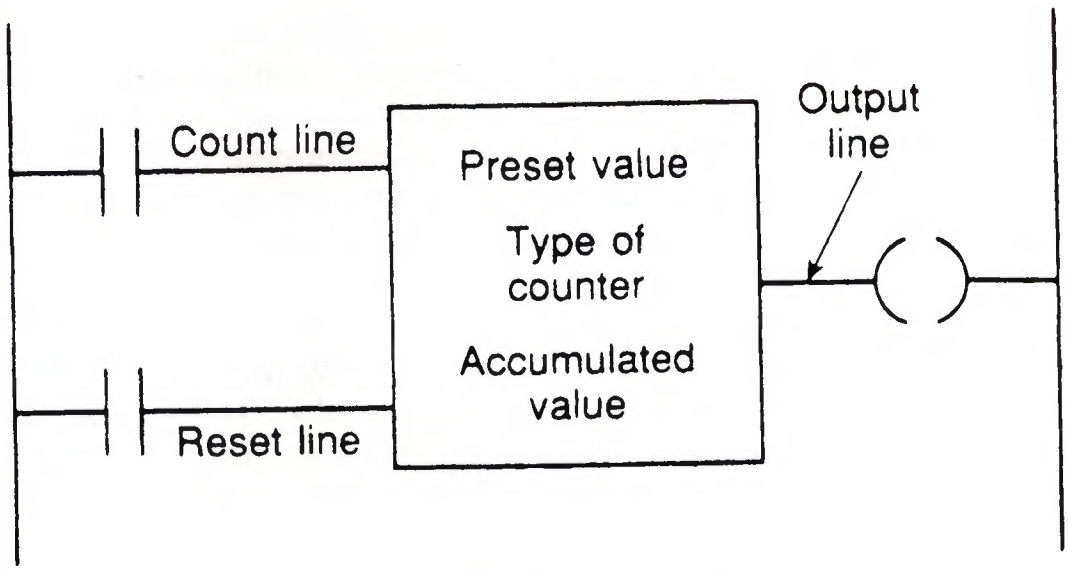


Fig.51
Block-formatted counter instruction.

Figure 51 Shows a generic block-formatted counter the type of counter (up or down) along with the counters preset value and accumulated or current value. The counter has two input conditions associted with it :COUNT and Reset. All PLC counters operate, or count, on the leading edge of the input signal.The counter

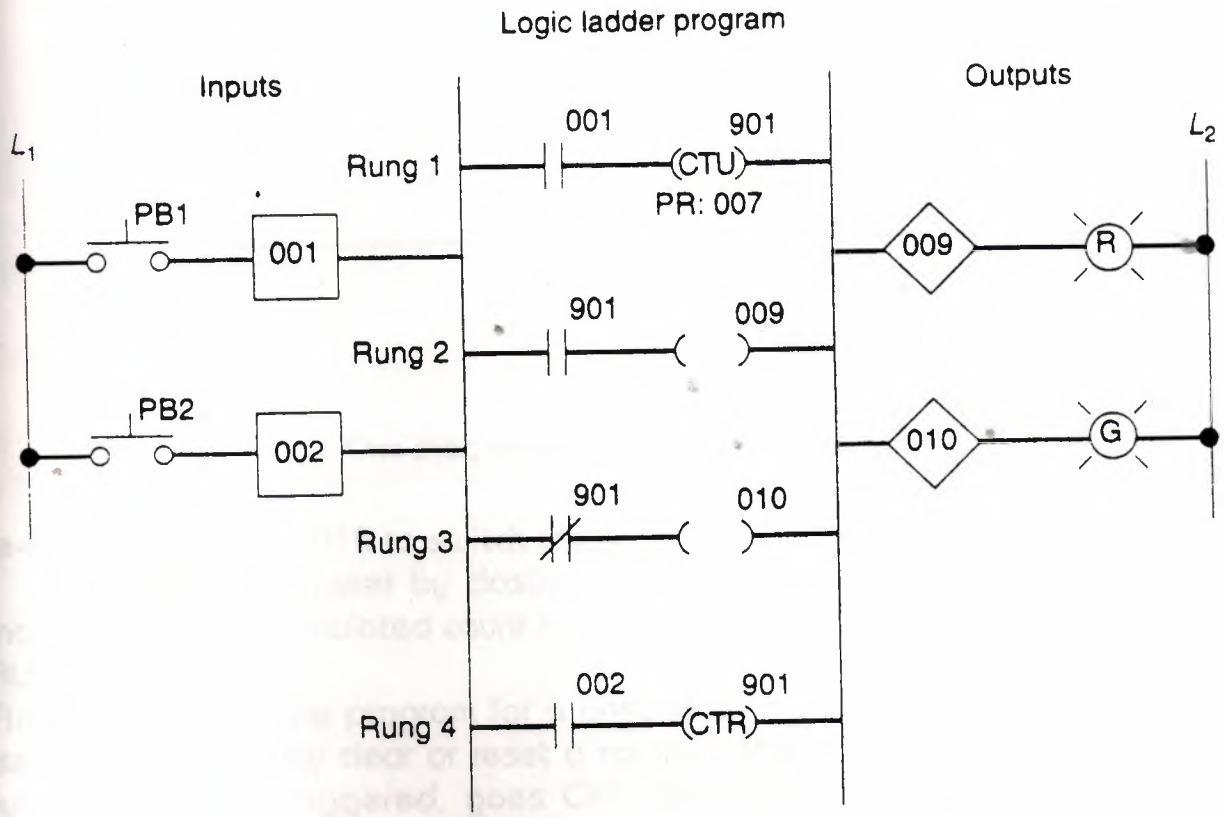


Fig. 52
Simple up-counter program

will either increment or decrement whenever the count input transfer from an OFF state to an ON state. The counter will not operate on the trailing edge, or ON-to-OFF transition, of the input condition. Most PLC counters are normally retentive.

That is to say, whater count was contained in the counter at the time of a processor shutdown will be restored to the counter on powerup.The counter may be reset, however , if the reset condition is activated at the time of power restoration.

KThe-up counter output instruction will increment by 1 each time the counted event occurs. (Figure 52) shows the program and timing diagram for a simple up-counter. This control application is designed to turn the green pilot light ON and the red pilot light OFF after an accumulated count of 7. Operating pushbutton PBI provides the OFF-to-ON transition pulses that are counted by the conuter. The preset value of the counter is set for 007. Each FALSE-to-TRUE transition of rung 1 increasses the counter's accumulated value by 1. After 7 pulses, or count when the preset counter value eguals the accumaleted counter value,output 901 is energized. As a result, rung 2 becomes TRUE, energizing output 009 to swit the red pilot light ON. At the same time,rung 3 becomes FALSE,

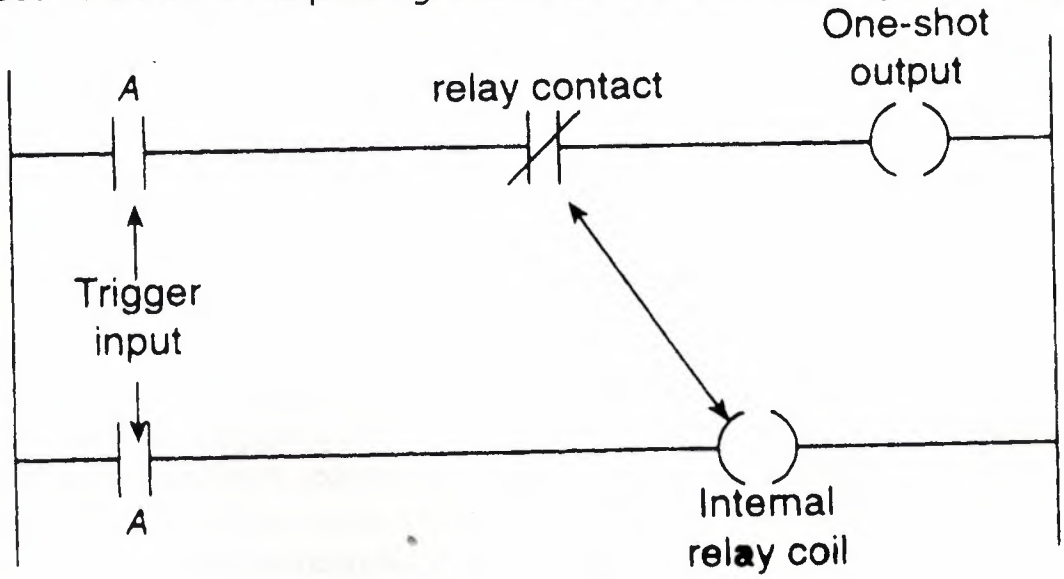


Fig. 53
One-shot, or transitional, contact program

de-Energizing output 010 to switch the green pilot light OFF.

The counter is reset by closing pushbuton PB 2, which makes rung 4 TRUE and resets the acciumulated count to zero. Counting can resume when rung 4 goes FALSE again.

(Figure 53) shows the program for a oneshot or transitonal, contact circuit ogften used to automatically clear or reset a counter. The program generates an output pulse that, when triggered, goes OFF. The one-shot can be triggered form a momentary signal or from one that comes ON and stays ON for some time.

Whichever is used, the one-shot is triggered by the leading edge (OFF to

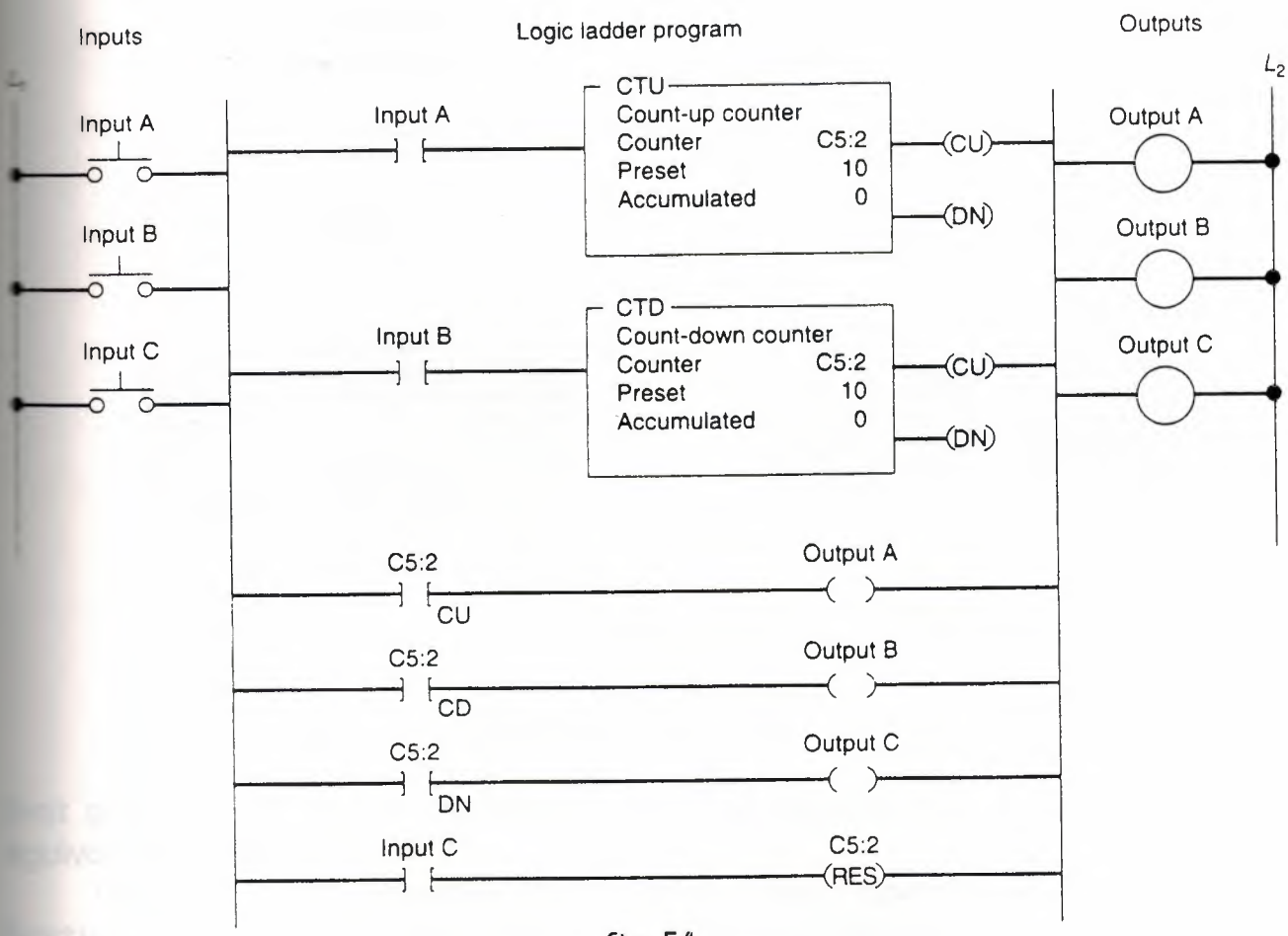


Fig. 54
Up-down counter program

ON) transition o2f the input signal. It stays ON for one scan and then goes OFF. It stays OFF Untilthe trigger goes OFF, and then comes ON again.The one-shot is perfect for resetting both counters and timers since it stays ON for one scan onluy.

The down-counter output instruction will count dowen or decrement by 1 each time ecach time the coundet event occurs.

Each time the down count event occurs, the accumulated value is decremented. Ronmally a down- counter is used in conjunction with an up-counter to form an up/down-counter. (Figure 54)

Show the program for a black-for-matted up/down-counter program. Note that the same addresse is given the up-counter instruction,the down-counter instruction, and the reset instruction.Allthere instructions will be looking at the same address in the counter file. When input A goes from FALSE to TRUE , one count is addet tothe unts is added to the accumulated value. Wheninput B goes from FALSE to TRUE , one count is subtracted from the accumulated value.

5) DATA MANIPULATION INSTRUCTIONS

Data manipulation instructions enable the programmable controler to take on some of the gualities of a computer system. Most PLCs now have this ability to manipulate data that is stored in memory. It is this extra computer characteristic

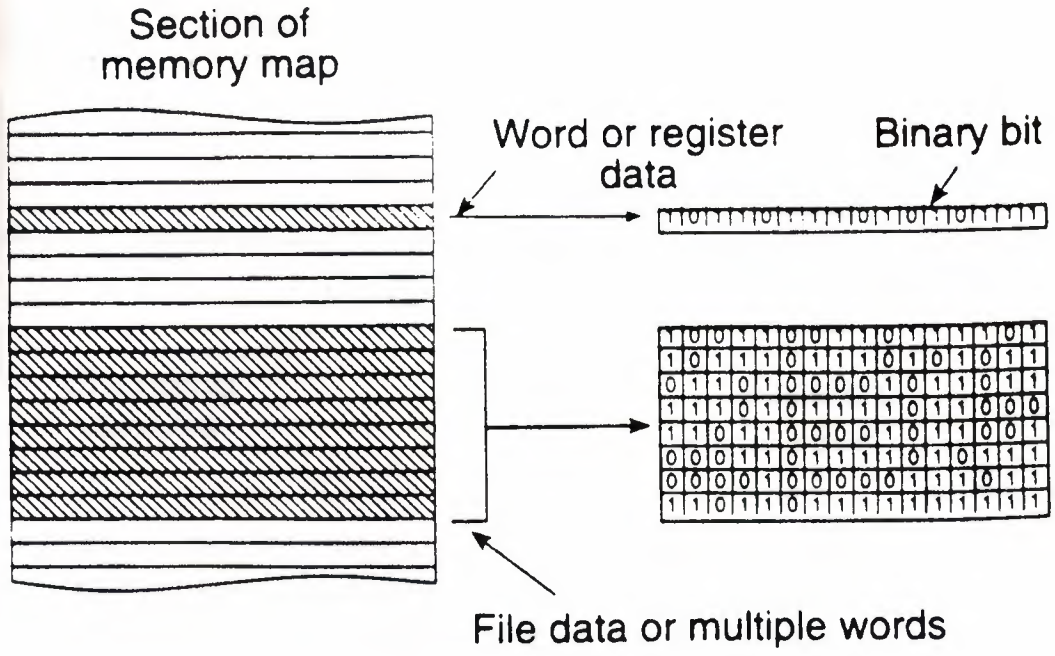


Fig. 55
Word data and file data.

that gives the PLC capabilities that go far beyond the convetnional relay eguivalent instructions.

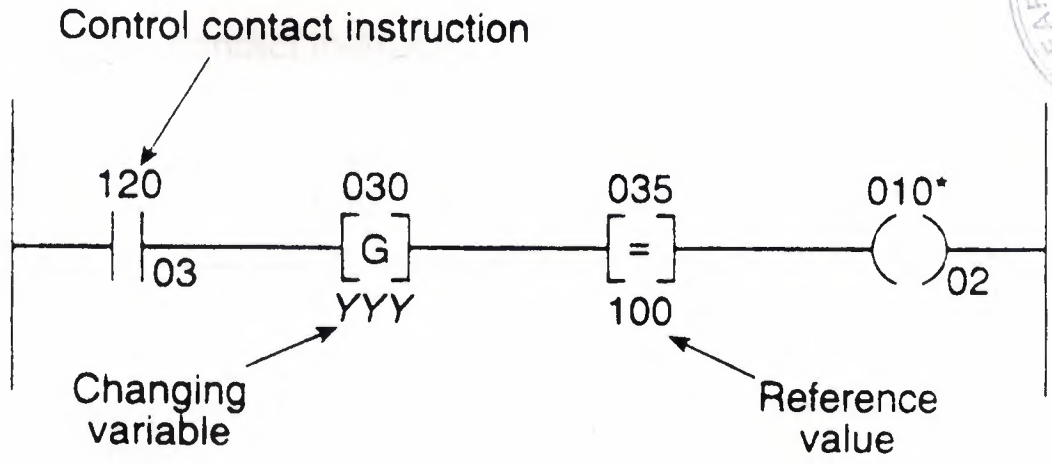
Data manipulation involves the transfer of data, operatin on data with math functioins, data conversions, data comparison and logical operations on data. There are two basic classes of instructionsto accamplish this: instructions that opeğrate on word data, and those that operate on file data, which is multiple words. (Figure 55) illustrates the difference between word data and file data.

Data transfer instructions simply involve the transfer of the contents from one word or register to onather. Figüre 56 illustrates wordlevel move instructions.Data transfer instructions can address almost any location in thememory . Prestored values can be automatically retrived and placed in any new location.

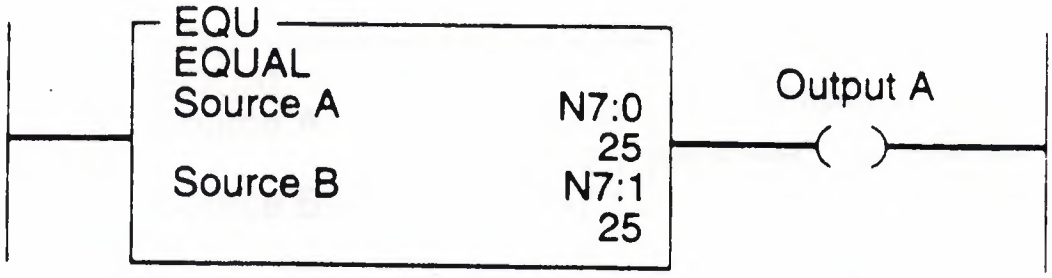
That location maybe the preset register for a timer or counter or even an output register that controls a 7-segment display.

Allen-Bradley PLC -2 protocol, the PUT instruction is used with the GET instruction to form a data transfer rung.

When input 110/10 is TRUE, the GET/PUT instructions tell the processor to get the numeric value 005 stored in word 020 and put it into word 130, in every case the PUT instruction must be precedded by a GET instruction. In Fig 12-58 (B) The Allen-Bradley PLC-5 protocol,the block-formatted moive (MOV) instruction is used to copy data form a source in the source, N 7:56 , is being copied into the addreess indicated in the destination, N 7: 60. This value Will be copied every time the instruction is scanned and the instruction is TRUE, When the rung goes FALSE, the destination address will retain the value, unlessit is changed elsewhere in the program. The instruction may be programmed with input conditions



(*a*) Allen-Bradley PLC-2 protocol: Output 010-02 will be energized when input 120-03 is TRUE and the value in word 030 is equal to the reference value 100 in word 035.



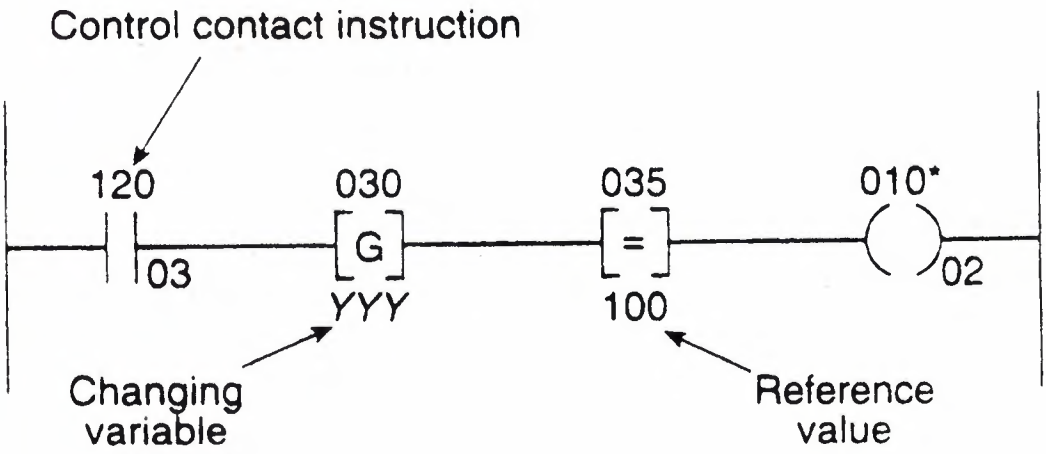
(*b*) Allen-Bradley PLC-5 protocol: When the value in source A equals the value in source B, the instruction is TRUE, causing output A to go TRUE. Source A or source B could also be a constant stored in the instruction.

Fig. 56
Word-level move instruction.

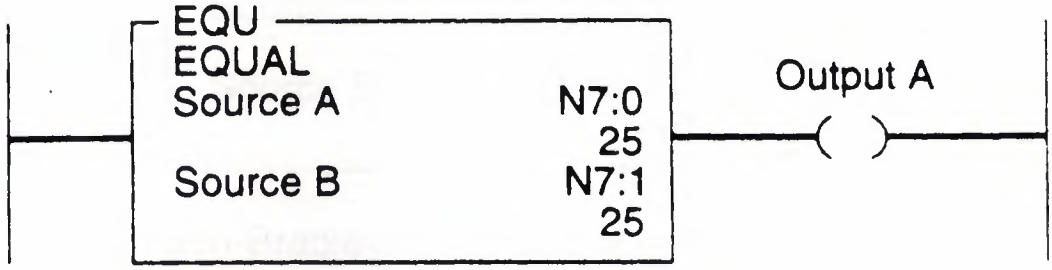
preceding it, or it may be programmed unconditionally.

Data-Compare instructions compare thi data stored in two or more words (or reğisters) and make decisions based on the program instructions (Figs 57 to 61). Nmeric values in two words of memoruy can be compared for each of the following conditions depending on the PLC:

Name	Mnemonic	Symbol
LESS THAN	LES	
EQUAL TO	EQU	
GREATER THAN	GRT	



(*a*) Allen-Bradley PLC-2 protocol: Output 010-02 will be energized when input 120-03 is TRUE and the value in word 030 is equal to the reference value 100 in word 035.



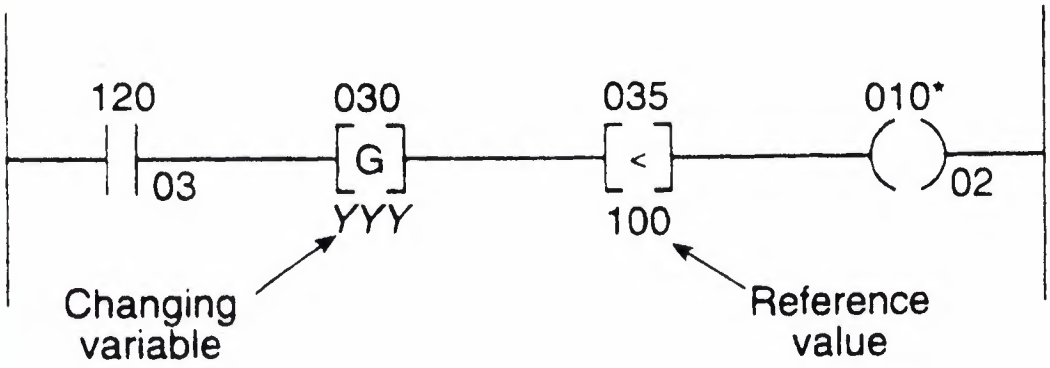
(*b*) Allen-Bradley PLC-5 protocol: When the value in source A equals the value in source B, the instruction is TRUE, causing output A to go TRUE. Source A or source B could also be a constant stored in the instruction.

Fig. 57
Equal instruction

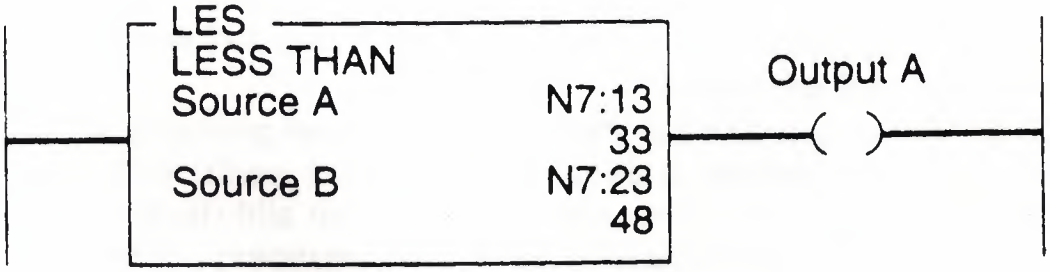
LESS THAN OR EQUEL TO LEQ

GREATER THAN OR EQUAL TO GEQ

Date comparison concepts have already been used whit teh itemer and counter insturcutions. In both of these instructions an output was trned ON or OFF when the accumulated value of the timer or counter egualed its preset value (ACPR), What actually occurred was thatthe accumulated numeric data in one memoruy



(*a*) Allen-Bradley PLC-2 protocol: Output 010-02 will be energized when input 120-03 is TRUE and the value in word 030 is less than the reference value 100 in word 035.



(*b*) Allen-Bradley PLC-5 protocol: When the value in source A is less than the value stored in source B, the instruction is TRUE, and output A will be TRUE; otherwise, output A will be FALSE.

Fig. 58
Less than instruction.

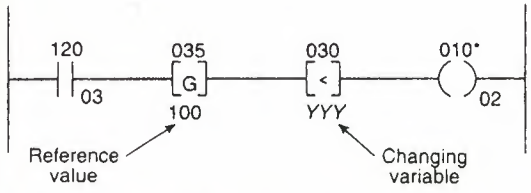
word was compared to the reset value of another memoruy word on each scan of the procesor. When the processor saw that the accumulated value was egual to(=)the preset value it switched the output ON or OFF.

The limit test instruction compares a test value to values in the low limit and in the high limit. It can function in either of two ways:

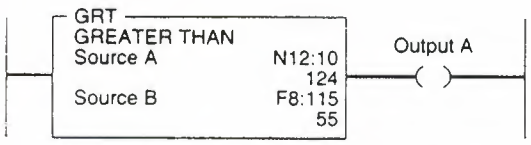
If the high limit has a greater value than the low limit, hten the instruction is true if the value of the test is between or egUal to the values of the hgih limit and the low limit.

If the value of the low limit is greater than the value of the high limit, the instruction is true if the value of the test is egual to orless than the low limit or egual to or greater than the high limit.

In (Fig 62) the high limit has a value of 50. and the low limit a value of 25.



(*a*) Allen-Bradley PLC-2 protocol: Output 010-02 will be energized when input 120-03 is TRUE and the value in word 030 is greater than the reference value 100 in word 035.

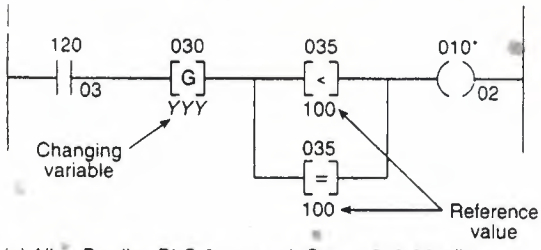


(*b*) Allen-Bradley PLC-5 protocol: When the value at source A is greater than the value at source B, the instruction is TRUE and output A will be TRUE; otherwise, it is FALSE.

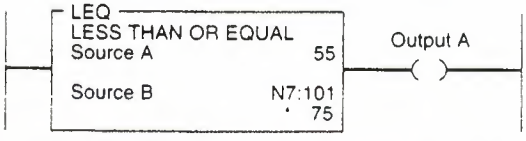
Fig. 59
Greater than instruction.

The instruction is TRUE, then for values of the test form 25 through 50.b The instruction as shown is TRUE Because the value of the test is 48.In (Fig 620) the high limit has a value of 50. and the low limit a value of 100. The instruction is TRUE, then for test values of 50 and lesss than 50 .and 100 and greater than 100. The instruction as shown is TRUE because the test value is 125.The use of comparison instructions is generallay straightforward. However, one common programming error involves thi use ofg these instruction in a PLC program to control the flow of a raw material into vessel. The receiving vessel has its wieght continuously monitored by the PLC Program as it fills.When the weight reaches a preset value, the flowis cut off.Wnille the vessel, fills the PLC preforms o comparison between the vessels current weight and a predetermined constant programmed in the processor. If the programmer uses only the EQUAL TO Instruction, the instruction becomes TRUE and the flow is stopped. However should the supply system leak additional inaterial into the vessel. the total weigh of the material could rise above the preset value, causing the instruction to go FALSE and the vessel to overgill. The simplest solution toı this problem is to program the comparison instruction as GREATER THN OR EQUALTO. This way any execess materila entering the vessel will not affect the filling operation. ıt may be necessary, in some cases, to include additional programming to indicate a serious overfill condition.

Set point control in its simplest form compares an input value, such as an analog or thumbwheel inputs, to a set point value. A discrete output signal is provided if the input value is less than, egual to, or

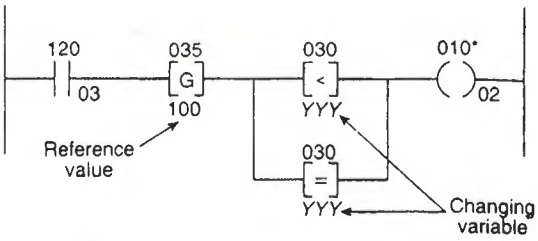


(*a*) Allen-Bradley PLC-2 protocol: Output 010-02 will be energized when input 120-03 is TRUE and the value in word 030 is less than or equal to the reference value 100 in word 035.

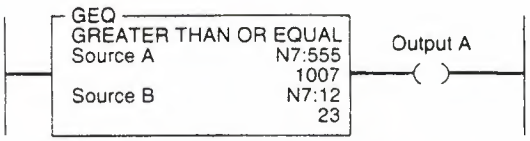


(*b*) Allen-Bradley PLC-5 protocol: When the value at source A is less than or equal to the value at source B, the instruction is TRUE and output A will be TRUE; otherwise, it is FALSE.

Fig. 60
Less than or equal to intruction.



(*a*) Allen-Bradley PLC-2 protocol: Output 010-02 will be energized when input 120-03 is TRUE and the value in word 30 is greater than or equal to the reference value 100 in word 035.



(*b*) Allen-Bradley PLC-5 protocol: When the value at source A is greater than or equal to the value at source B, the instruction is TRUE and output A will be TRUE; otherwise, output A will be FALSE.

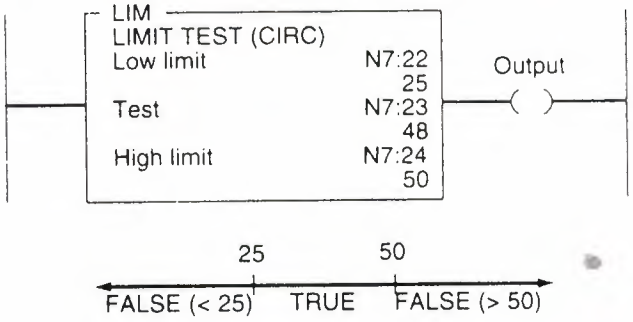
Fig. 61
Greater than or equal to instruction.

greater than the set point value.

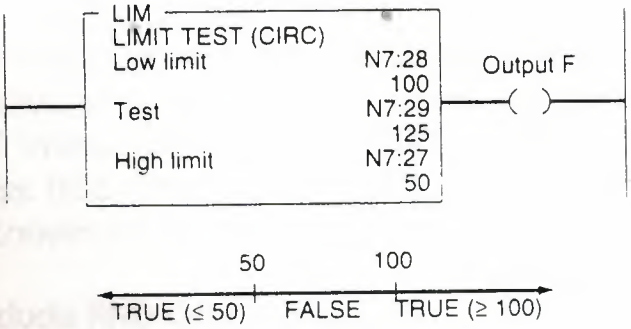
The temperatuer control program of (Fig 63) is one example of set point simple OFF/ON control of the electric beating elements of an oven. The oven is tu maintain an averağeset point temperature of 600 F, with a variation of about 1 percent between teh Off nad On cycles. Therefore, the electric heaters will be turned On when the temperatuere of the oven is 597 f or less and stay ON until the temperature risse to 603 F or more. The electric heaters stay Off until the tempretature drorps down to 597 F, at which time the cycle repeats itself. Rung 2 contains a GET/LESS OR EQUAL TO logic instruction and rung 3 contains the eguivalent of a GET/GREATER THAN OR EQUAL TO logic instruction.

Rung 4 contains the logic for switching the heaters ON and OFF according to the high and low set points. Rung 1 contains the logic that allows the thermocouple temperature to be monitored by the LED display board.

In complek industrial machine or process control, it is necessary to carry out mathematical functions ranging from basic arithmetic, such as add, subtract, divide, and multply, to complex mathematics used in flow measurement and PID control. Math instructionsallow the PLC to perform arithmetic functions on values stored in memory words (Figs 64 to 67)



(*a*) High limit is 50, and low limit is 25



(*b*) High limit is 50, and low limit is 100

Fig. 62
Limit test instruction.

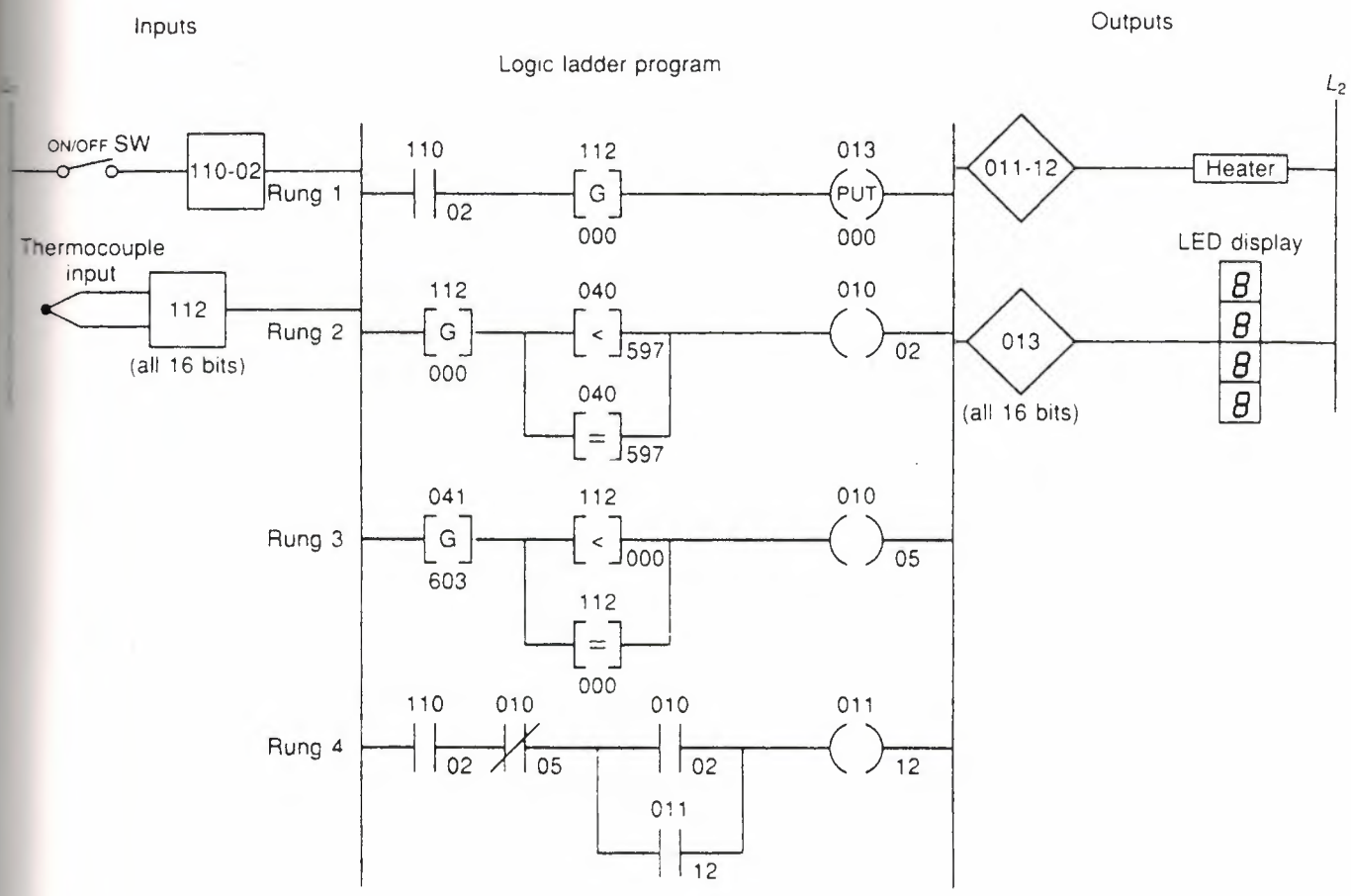
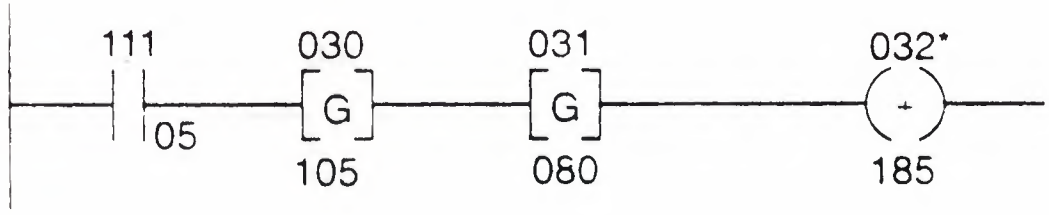


Fig. 63
Set point temperature control program

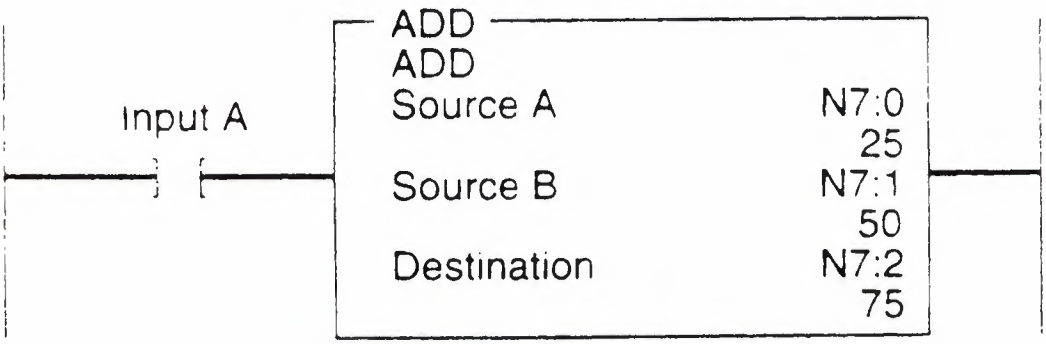
Figure 68 shows program used to convert celsius temperature to Fahrenheit. İn this application the thumbwheel swiutch connected to the input module indicates Celsius temperature. The program con verts the celsius üikhideüdi ur üni oeüiqe üesa in the date table to Fahrenheit values for display.

The formula forms the basis for the program. ın this example a temperature reading of 60 c is assumed ın rung 1.the GET instruction at address 030 multiplies the temperature (60C) by 9 and stores the product (540) in address 032. ın rung 2.the GET instruction at address 033 divides 5 into 540 and stores the guotient 108 in address 034. ın rung 3 the GET instruction at address 035 adds 32 to the value of 108 and sum 140 un address 036. Thus 60 C = 140 F. ın rung 4 the GET/PUT instruction pair transfers the converted temperature reading 140 F to the LED display.

Word-level logical instructions.include AND OR Exclusive OR (XOR), and NOT instructions. Allen-Bradley PLC-5 protocol for these instructions is illustrated in Figs. 69 to A file is a group of consecutive words in the data tamble taht have o defined start and end are used tu store information. For example, a batch process program may contain several separete recipes in different files that could be selected by an



(*a*) Allen-Bradley PLC-2 protocol: When input device 111/05 is TRUE, the value of word 030 (105) is added to the value of word 031 (080) and the sum (185) is stored in word 032.



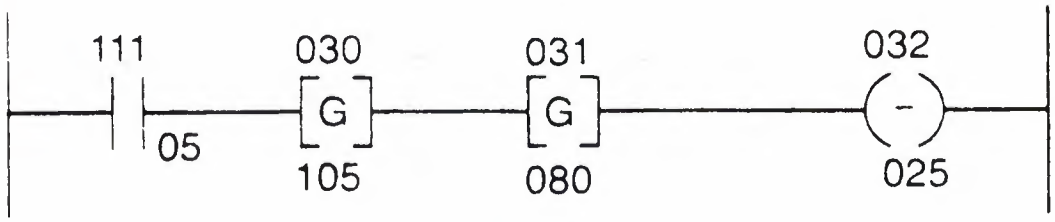
(*b*) Allen-Bradley PLC-5 protocol: When input device A is TRUE, the value stored at address N7:0 (25) is added to the value stored at address N7:1 (50) and the sum is stored at address location N7:2 (75).

Fig. 64
ADD instruction.

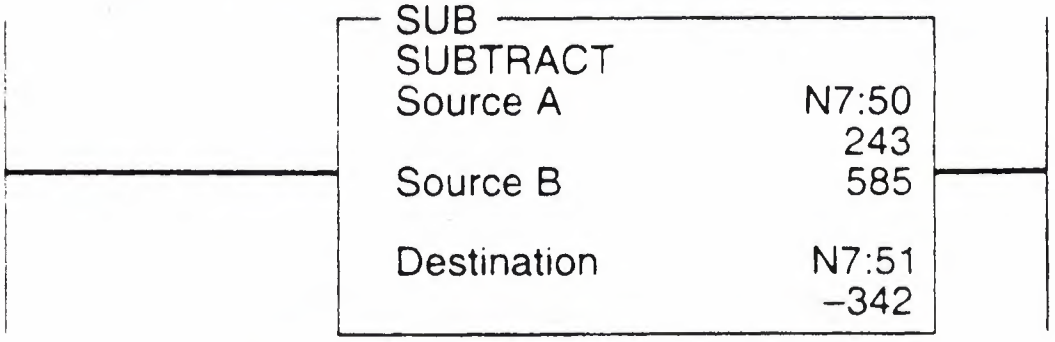
operator.

In some instances it may be necessary to shift complete files from one location to another within the programmable controler memory. Such data shift are termed file-tofile shiftsfile-to file shifts are used when the data in one file represents a set of conditions that must interact with the programmable controller program a nummber of times and, therefore, must remain intact after each operation. Because the data within this file must also bechanged by the program action. A second file is used to handle the data changes, and the information within that file is allowed to be altered by the program. The tada in the first file, however, remains constant and can, therefore, becalled upon tu be used a number of times.

Figure 73 illustrates the file-to-file copy instruction protocol used with the Allen Bradley PLC-5 family of controller, Data from the expressionfile N 7:20 will be



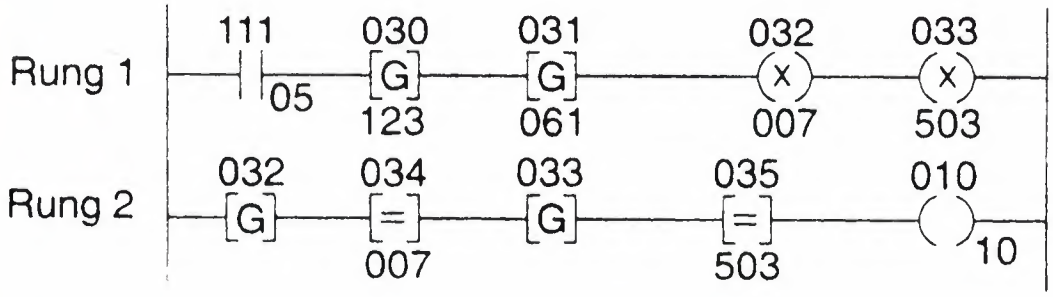
(*a*) Allen -Bradley PLC-2 protocol: SUBTRACT instruction rung. When input device 111/05 is TRUE, the value of word 031 (080) is subtracted from the value of word 030 (105), and the difference (025) is stored in word 032. Only positive values can be used with some PLCs.



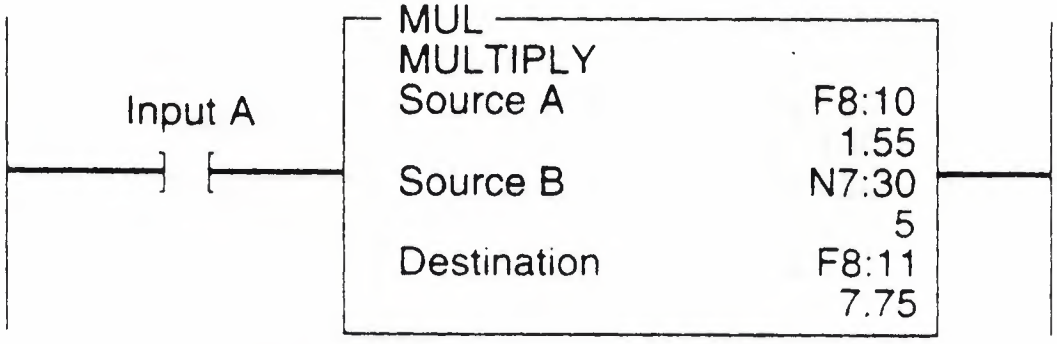
(*b*) Allen-Bradley PLC-5 protocol: Allows negative values to be used. Here the instruction is programmed unconditionally, i.e., the subtract operation will take place every time the instruction is scanned.The value stored at the address indicated in source B is subtracted from the value stored at the address indicated in source A, and the answer is stored at the address location indicated in the destination. Also, this example shows the instruction with a constant entered in source B. A constant may be entered in either source A or source B, and it is stored in the instruction.

Fig. 65
Subtract instruction.

copied into the testination file N7:50 .The length of the two files is set by the value entered in the control element word. R 6:1LEN in this instruction we have also used the ALL mode, which menas all of the data will be transferred in the first scan in which the FAL instruction sees a FALSE-toTRUE transition. The DN bit will



(*a*) Allen-Bradley PLC-2 protocol: When input device 111/05 is TRUE, the value of word 030 (123) is multiplied by the value of word 031 (061), and the product (7 503) is stored in word 032 (007) and word 033 (503). As a result, rung 2 will become TRUE, turning output 010/10 ON.



(*b*) Allen-Bradley PLC-5 protocol: Value stored at the address indicated in source A, F8:10, being multiplied by the value stored at the address in source B, N7:30, the result being stored at the address in the destination, F8:11.

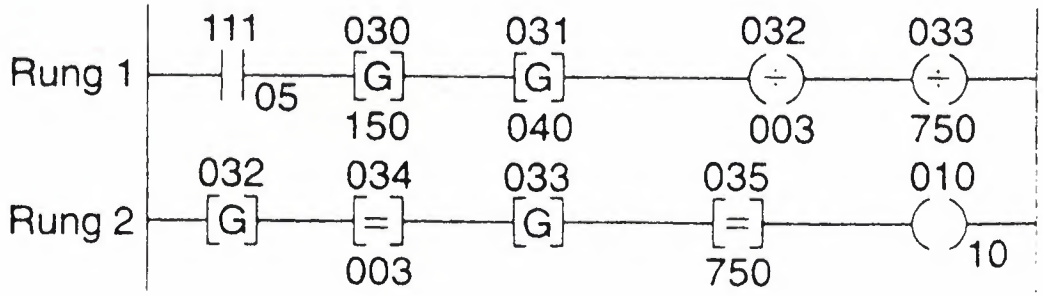
Fig. 66
MULTIPLY instruction.

also come ON in that scan.,unless an error occurs in the transfer of data, in which case the ER bit will be set, the instruction will stop operation at that position, and then the scan will continue at teh next instruction.

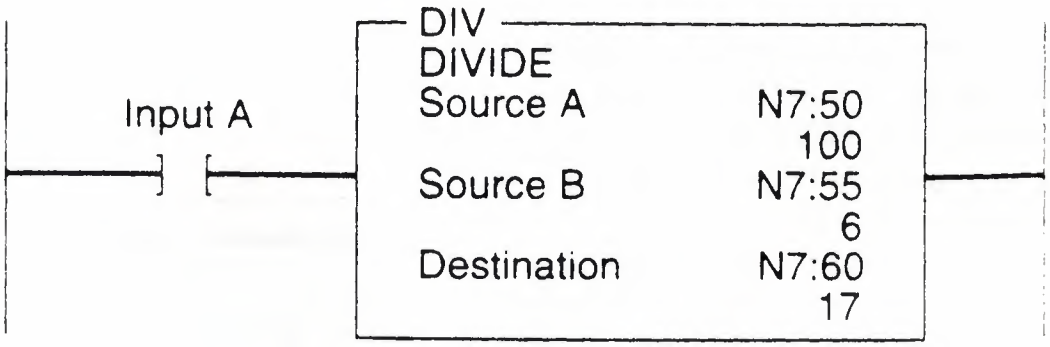
SHIFT REGISTER AND SEQUENCER INSTRUCTIONS

The shift register instruction is often used with conveyors to monitor and control the flow of the individual parts. In general, it can be used in the control of machines and processes in which parts are continually shifted from one position to the next.

You can program a shift register to shift status data either right or left. This is accomplished by shifting either status or values through data files. When



(*a*) Allen-Bradley PLC-2 protocol: When input device 111/05 is TRUE, the value of word 030 (150) is divided by the value of word 031 (040), and the quotient is stored in words 032 and 033 (003.750 or 3.75). As a result, rung 2 will become TRUE, turning output 010/10 ON.



(*b*) Allen-Bradley PLC-5 protocol: The example divides the value stored at source A's address (N7:50) by the value stored at source B's address (N7:55) and stores the result at the destination's address (N7: 60). Note that the result stored at the destination address is rounded off. If the remainder is 0.5 or above, the result is rounded up. If the remainder is less than 0.5, the answer is rounded down.

Fig. 67
DIVIDE istruction.

tracking parts on a status basic, bit shift registers are used. Bit shift instructions will shift bit status from a source bit address, through a data file, and out to an unload bit, one bit at a times. there are two bit shift instructions: bit shift left (BSL), which shifts bit status from a lower address number to a higher address number through a data file; and the bit shift right (BSR), which shifts data from a higher address number to a lower address number through a data file. (Figure 74

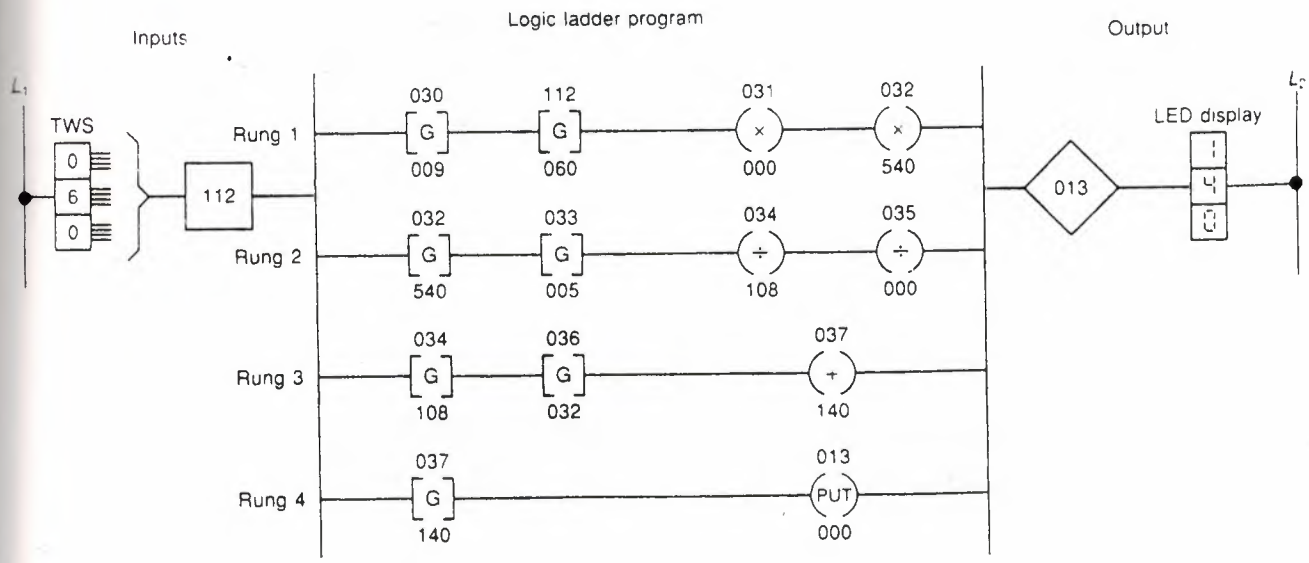


Fig. 68
Converting Celsius temperature to Fahrenheit.

Source A	Source B	Destination
0	0	0
1	0	0
0	1	0
1	1	1

(*a*) Truth table (1 appears in destination only when both A and B have a 1)

AND
BITWISE AND
Source A B3:5
1100110011001100
Source B B3:7
1111111100000000
Destination B3:10
1100110000000000

(*b*) AND instruction (destination bits are result of logical AND operation)

Fig. 69
ADD instruction.

a) the basic concept of a shift register; a common shift pulse or clock causes each bit in the shift register to move one position to the right. At some point, the number of data bits fed into the shift register will exceed the register's storage capacity. When this happens, the first data bits fed into the shift register by the shift pulse are lost out the end of the shift register. (Figure 74 b) shows a shift-right register. The status data (1 or 0) enters the register and is automatically shifted through the register from one bit address to the next on a time or event-driven basis. In the shift-left register shown in (Fig. 74 c) normally, the shift register instruction is retentive. (Figure 74 d) shows an 8-bit circulating shift register. (Figure 74 e) is an example of the bit shift right instruction (BSR) used with the Allen-Bradley PLC-5 protocol. When the instruction goes TRUE, the status of the bit address, I:002/05, is shifted into B3: 101/07, which is the twenty-fourth bit in the file. The status

Source A	Source B	Destination
0	0	0
1	0	1
0	1	1
1	1	0

(*a*) Truth table (1 appears at destination when source A or source B have a 1 but not when both have a 1)

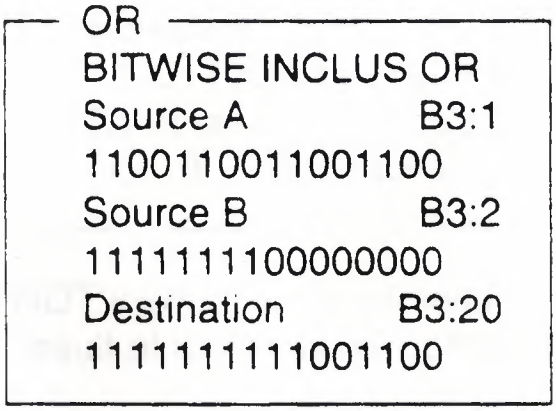
XOR
BITWISE EXCLUS OR
Source A B3:15
1100110011001100
Source B B3:17
1111111100000000
Destination B3:25
0011001111001100

(*b*) XOR instruction (destination bits are result of logical XOR operation)

Fig. 70
EXclusive OR (XOR) instruction.

Source A	Source B	Destination
0	0	0
1	0	1
0	1	1
1	1	1

(*a*) Truth table (1 does not appear at destination only when both source A and source B are 0)



(*b*) OR instruction (destination bits are result of logical OR operation)

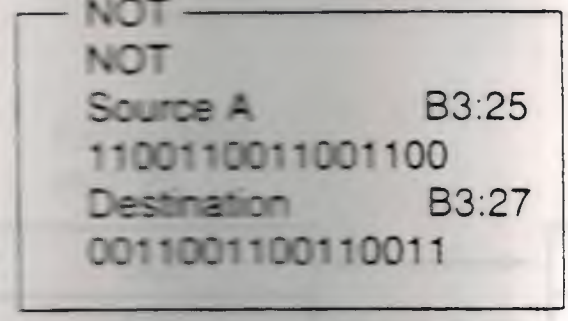
Fig. 71
OR instruction.

of all of the bits in the file are shifted one position to the right, through the length of 24 bts. The status of B3: 100/00 is shifted to the unload bit, R6:1/UL. The status previously in the unload bit is lost.

(Figure 75) illustrates a spray painting operation controlled by a shift register. The shift register's function is used to keep track of the items to be sprayed. As the parts pass along the production line, the shift register bit patterns represent the items on thi conveyor hangers to be painted. the logic of this operation is such that when a part to be painted and a part hanger occur together, indicated by the simultaneous operation of LS1 and LS2, a logical 1 is input into the shift register. The logiccal 1 will cause the undercoat spray gun to operate, and, five steps later, when a 1 occurs in the shift register, the top coat spray gun is operated. Limit swtch 3 counts the parts as they exit the oven. The count obtained by limit switch 1 and limit switch 3 should be equal at the end of the spray painting run and is an indication that the parts commencing the sray

Source A	Destination
0	1
1	0

(*a*) Truth table (bits are inverted between source A and the destination, with the 0s changed to 1s and the 1s changed to 0s)



(*b*) NOT instruction (destination bits are result of logical NOT operation)

Fig. 72
NOT instruction.

painting run equal the parts that have completed it. A logical : in the shift register indicates that the conveyor has no parts on it to be sprayed and therefore inhibits the operation of the spray guns.'

Sequencer instructions are used to control machines that operate in a sequencer instructions is to converse program memory. The sequencer output instruction functions in a manner similar to a mechanical drum switch, which is used to control output devices sequentially.

The programming of sequences will vary between programmable controller manufacturers, but the operational concepts are the same. The sequence of events controlled by the sequencer is determined by the bit panttern of each consecutive word and the number of words in the sequence.

(Figure 76) shojs the way a typical sequencer output instruction works. In this example, 16 lights are used for outputs. Each light represents one bit address (1 through 16) of output word 050. The lights are programmed in a four-step sequence to simulate the operation of two-way traffic lights.

Data is entered into a work file for each sequencer step as illustrated.

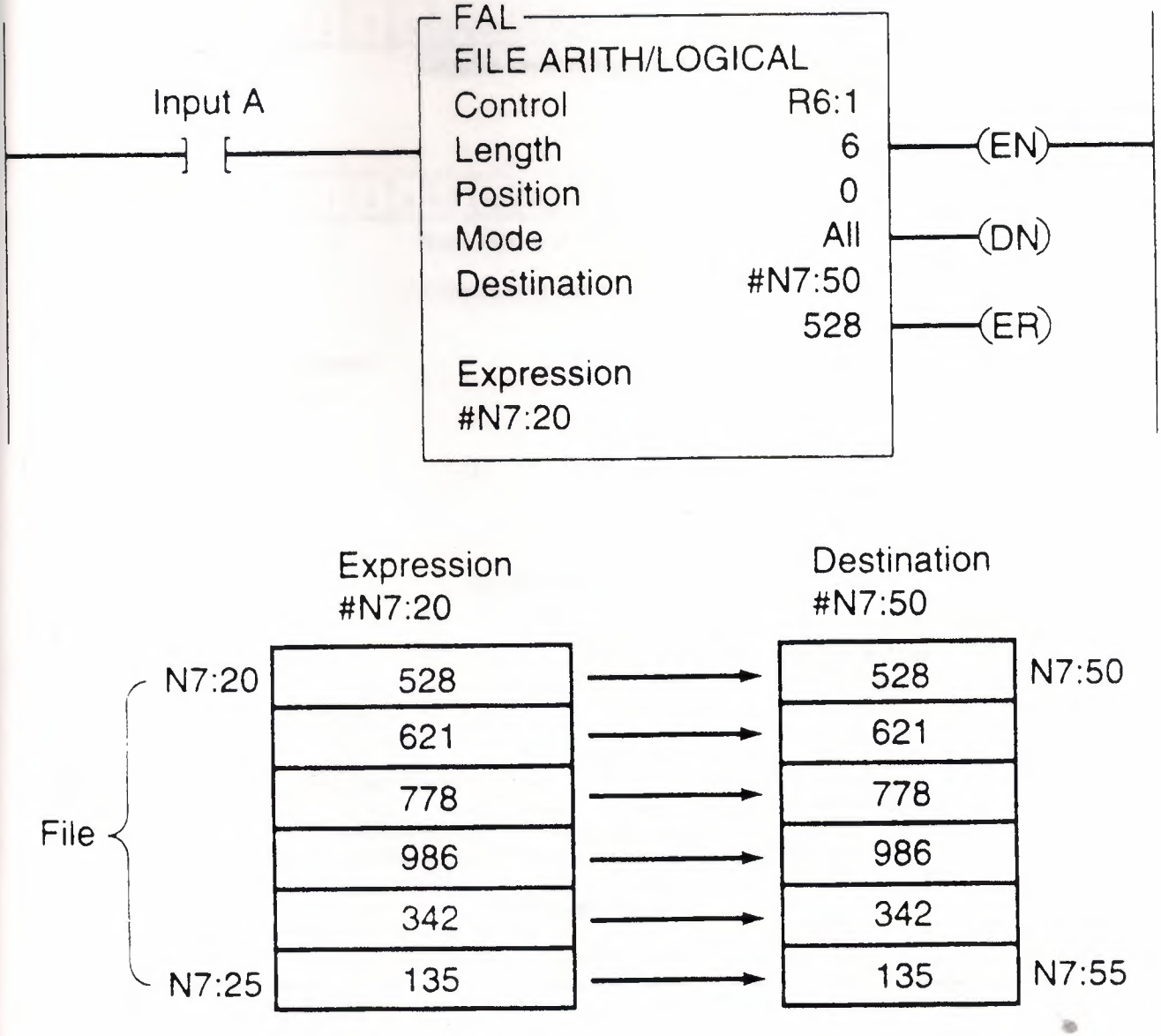
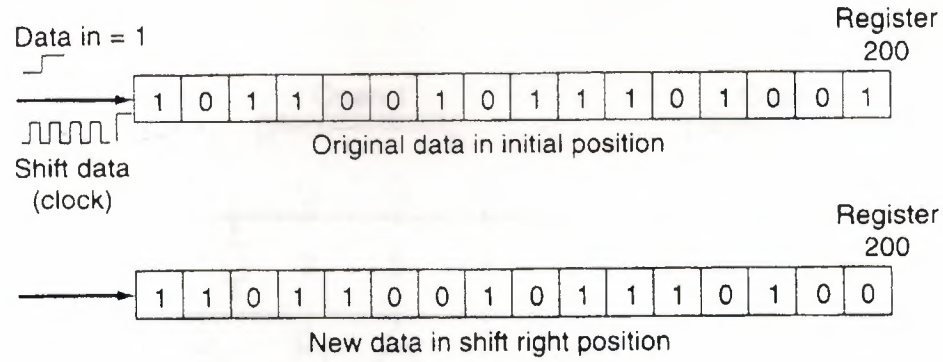


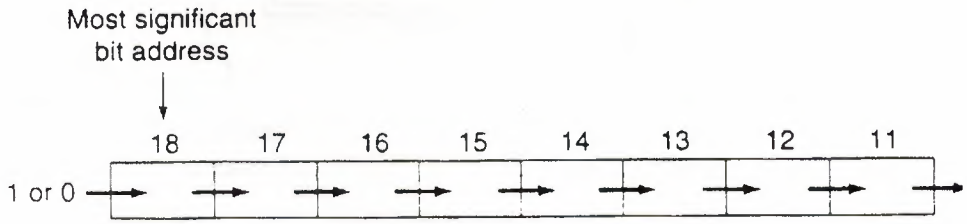
Fig. 73
File-to-file copy instruction.

Using the programmer, binary information (1s and 0s) that reflects the desired light sequence is entered into each word of the file. For ease of programming, some PLCs allow the word file data to be entered using the octal, hexadecimal, BCD, or similar number system. When this is the case, the required binary information for each sequencer step must first be converted to whatever number system is employed by the PLC. This information is the entered with the programmer into the word file.

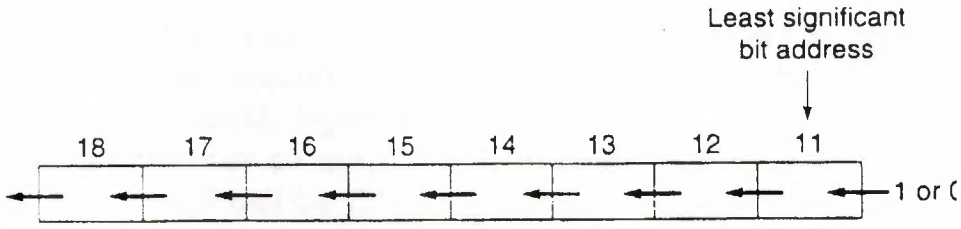
Once the data has been entered into the word file of the sequencer, the PLC is ready to control the lights. When the sequencer is activated and advanced to step 1, the binary information in word 060 of the file is transferred into word 050 of the output. As a result, lights 1 and 12 will be switched ON and all the rest will remain OFF. Advancing the sequencer to step 2 will transfer the data from word



(*a*) Basic concept of shift register

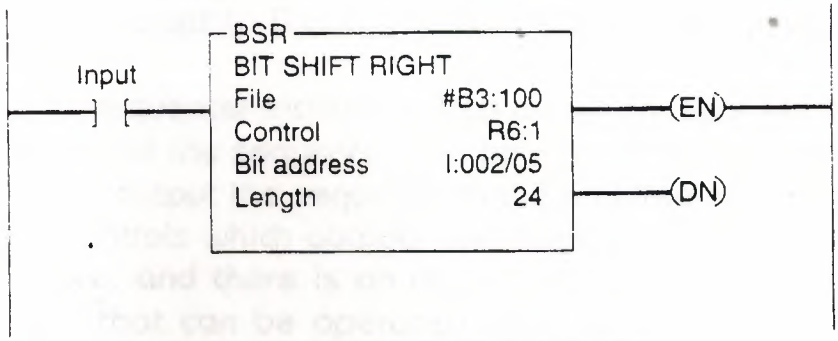


(*b*) Shift right register



(*c*) Shift left register

(*d*) 8-bit circulating shift register



(*e*) Example of bit shift right instruction (BSR) used with Allen-Bradley PLC-5 protocol

Fig. 74
Shift registers.

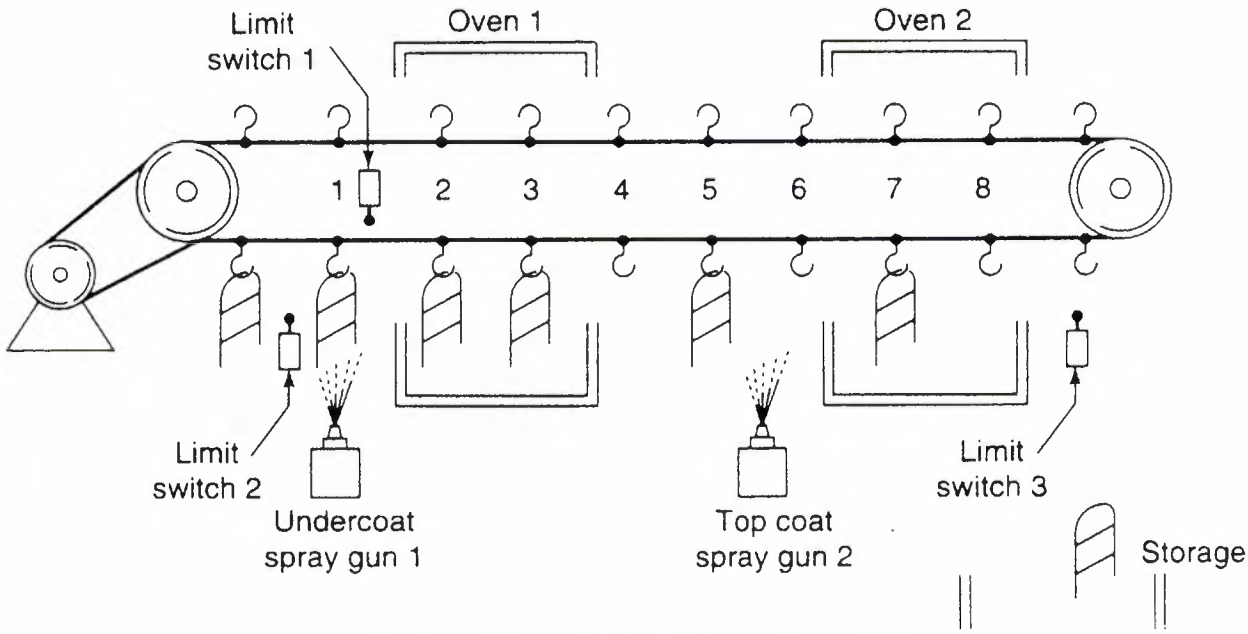


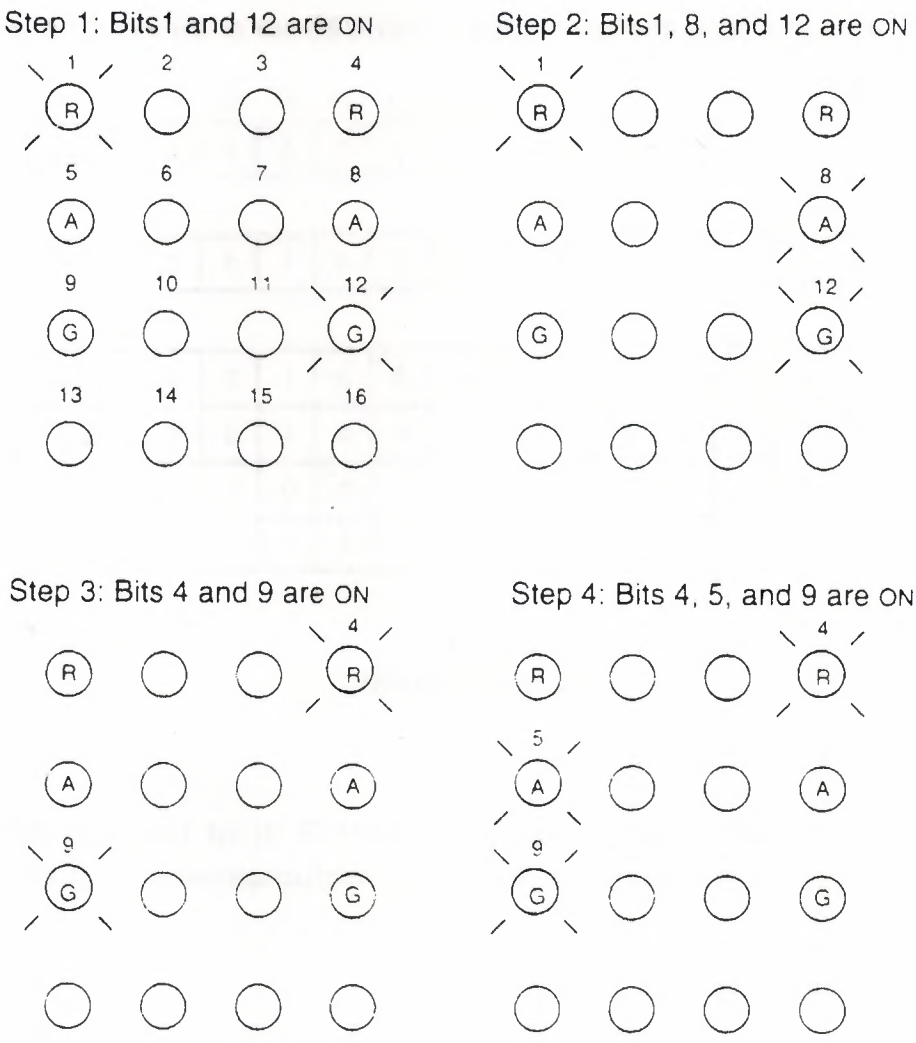
Fig. 75
Shift register spray painting application.

061 into word 050. As a result, lights 1, 8, and 12 will be On and all the rest will be OFF. Advancing the sequencer to step 3 will transfer the data form word 062 into word 050. As a result, lights 4 and 9 will be On and all the rest will be OFF, Advancing the sequencer to step 4 will transfer the data from word 050. As a result, lights 4, 5, and 9 will be ON and all the rest will be OFF. When the last step is reached, the sequencer is either automatically or manually reset to step 1.

When a sequencer operates on an entire output word, there may be outputs associated with the word that do not need to be controlled by the sequencer. In our example, bits 2, 3, 5, 7, 10, 11, 13, 14, 15, and 16 of output word 050 are not used by the sequencer but could be usd elsewhere in the program.

To prevent the sequencer from controlling these bits of the output word, a mask word (040) is used. The use of a mask word is illustrated in (Fig. 77) mask word file to the output word. For each bit of output word 050 that the sequencer is to control, the corresponding bit of mask word 040 must be set to 1. All other bits of output word 050 are set to 0 and therefore can be used independently of the sequencer.

To program a sequencer instruction into a PLC, an input file is created which holds the bit pattrent of the sequence read from a number of input words, and an output file is used to output the sequence to the required devices. The bit pattern of the mask word controls which outputs are made active. Sequencer instructions are usually retentive, and there is an upper limit as to tha number of external outputs and steps that can be operated upon by a single instruction. Many sequencer instructions reset the sequencer automatically to step 1 upon completion of the last sequence step. Other instructions provide an individual reset control line or a combination of both. (Figure 78) illustrates a typical block formatted sequencer instruction.



(*a*) Sequencer steps

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Word 050	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Output
Word 060	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	Step 1
Word 061	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	Step 2
Word 062	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	Step 3
Word 063	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	Step 4

Four-word file

(*b*) Binary information for each sequencer step

Fig. 76
Four-word sequence

A sequencer program can be event-driven or time-driven. An event-driven sequencer operates like a mechanical stepper switch that increments by one step

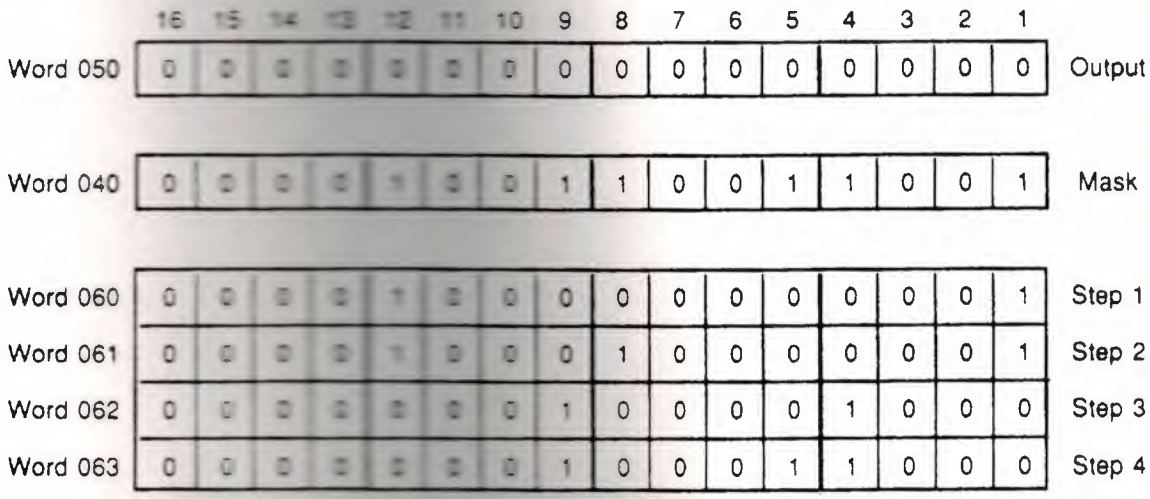


Fig. 77
Using a mask word.

for each pulse applied to it. A time-driven sequencer operates like a mechanical drum switch that increments automatically after a preset time period.

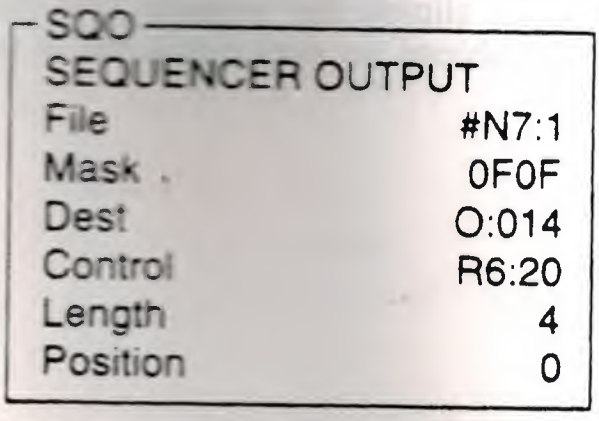


Fig. 78
Sequencer instruction.

CONCLUSION

The Programing logic control system ensures that kegs are distrubuted to washing and fiiling lanes a manner which keeps lane starvation at the minumum level. The system automatically resets the flowmeter set points and carries out standart deviation calculations to ensure that fiiling is accurate. Automatic recalibration facilities are provided to take account of the need to wash and fill different sizes of kegs during the operating periods. The gas pressure in every keg is checked againist master transducers to ensure that the required pressures are achieved.

Comprehensive on-line monitoring facilities are provided, which include the numer of kegs washed, filled, and rejected. Full details of the reasons why kegs are rejected are provided in tabulated from on enalde production problems to be quickly identified.