1. Human Machine Interfaces (HMIs)

A *human machine interface (HMI)* can be connected to communicate with a PLC and to replace pushbuttons, selector switches, pilot lights, thumbwheels, and other operator control panel devices Luminescent touch-screen keypads provide an operator interface that operates like traditional hardwired control panels.

Human machine interfaces give the ability to the operator and to management to view the operation in real time. Through personal computer–based setup software, can configure display screens to:

• Replace hardwired pushbuttons and pilot lights with realistic-looking icons. The machine operator need only touch the display panel to activate the pushbuttons.

• Show operations in graphic format for easier viewing.

• Allow the operator to change timer and counter presets by touching the numeric keypad graphic on the touch screen.

• Show alarms, complete with time of occurrence and location.

• Display variables as they change over time.

2. Memory organization takes into account the way a PLC divides the available memory into different sections.

The memory space can be divided into two broad categories:

program files

data files.

Program files are the part of the processor memory that stores the user ladder logic program. The program accounts for most of the total memory of a given PLC system. It contains the ladder logic that controls the machine operation. This logic consists of instructions that are programmed in a ladder logic format. Most instructions require one word of memory.

The data files store the information needed to carry out the user program. This includes information such as the status of input and output devices, 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/OFF type of information represented by 1s and 0s, stored in unique bit locations. Number or code information is represented by groups of bits that are stored in unique byte or word locations.

Program Files

Program files are the areas of processor memory where ladder logic programming is stored. They may include:

• **System functions (fi le 0)** —This fi le is always included and contains various system-related information and user-programmed information such as processor type, I/O confi guration, processor fi le name, and password.

• **Reserved (fi le 1)** —This fi le is reserved by the processor and is not accessible to the user.

• **Main ladder program (fi le 2)** —This fi le is always included and contains user-programmed instructions that defi ne how the controller is to operate.

• **Subroutine ladder program (fi les 3–255)** —These fi les are user-created and are activated according to subroutine instructions residing in the main ladder program fi le.

Data Files

The data fi le portion of the processor's memory stores input and output status, processor status, the status of various bits, and numerical data. All this information is accessed via the ladder logic program. These fi les are organized by the type of data they contain and may include:

• **Output (fi le 0)** —This fi le stores the state of the output terminals for the controller.

• **Input (fi le 1)** —This fi le stores the status of the input terminals for the controller.

• **Status (fi le 2)** —This fi le stores controller operation information and is useful for troubleshooting controller and program operation.

• **Bit (fi le 3)** —This fi le is used for internal relay logic storage.

• **Timer (fi le 4)** —This fi le stores the timer accumulated and preset values and status bits.
• **Counter (fi le 5)** —This fi le stores the counter accumulated and preset values and status bits.
• **Control (fi le 6)** —This fi le stores the length, pointer position, and status bit for specifi c instructions such as shift registers and sequencers.
• **Integer (fi le 7)** —This fi le is used to store numerical values or bit information.
• **Reserved (fi le 8)** —This fi le is not accessible to the user.
• **Network communications (fi le 9)** —This fi le is used for network communications if installed or used like fi les 10–255.
• **User-defi ned (fi les 10–255)** —These fi les are userdefined as bit, timer, counter, control, and/or integer data storage.

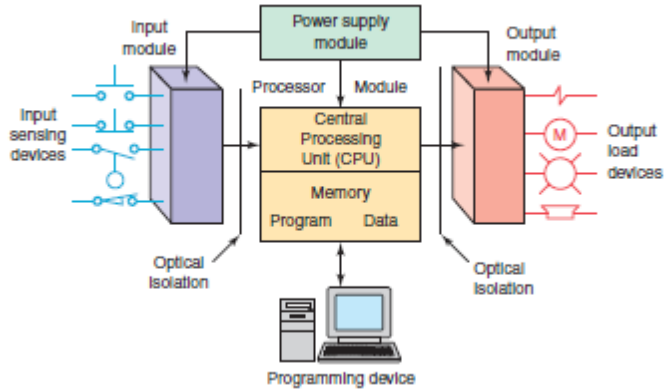
1. Parts of PLC typical

PLC can be divided into parts, as illustrated in Figure 1-8 . These are the *central processing unit (CPU)* ,the *input/output (I/O)* section, the *power supply,* and the *programming device.*

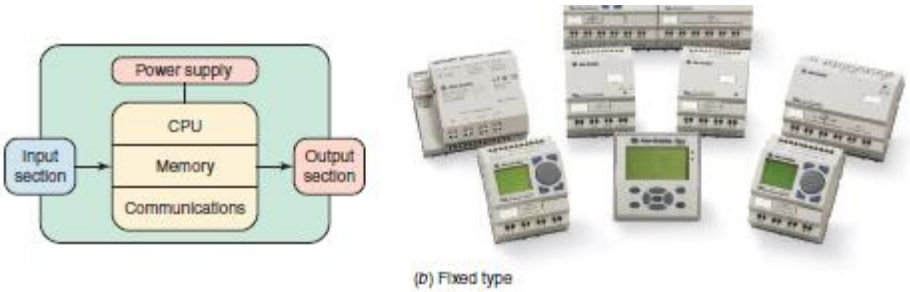
There are two ways in which I/Os (Inputs/Outputs) are incorporated into the PLC: fi xed and modular. *Fixed I/O* (Figure 1) is typical of small PLCs that come in one package with no separate, removable units. The processor and I/O are packaged together, and the I/O terminals will have a fi xed number of connections built in for inputs and outputs. The main advantage of this type of packaging is lower cost. The number of available I/O points varies and usually can be expanded by buying additional units of fi xed I/O. One disadvantage of fi xed I/O is its lack of fl exibility; you are limited in what you can get in the quantities and types dictated by the packaging. Also, for some models, if any part in the unit fails, the whole unit has to be replaced.

Modular I/O (Figure 2) is divided by compartments into which separate modules can be plugged. This feature greatly increases your options and the unit's flexibility.

The basic modular controller consists of a rack, power supply, processor module (CPU), input/output (I/O modules), and an operator interface for programming and monitoring. The modules plug into a rack. When a 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



(a) Modular type



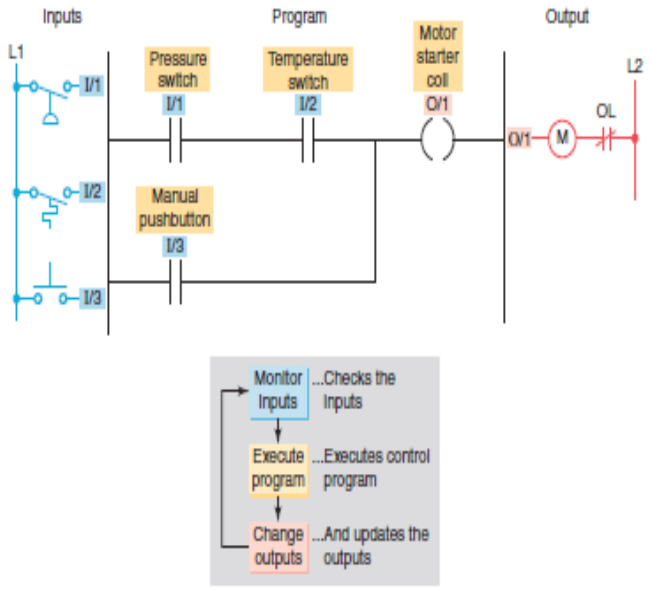
(b) Fixed type

The *processor* (CPU) is the "brain" of the PLC. A typical processor usually consists of a microprocessor for implementing the logic and controlling the communications among the modules. The processor requires memory for storing the results of the logical operations performed by the microprocessor. Memory is also required for the program EPROM or EEPROM plus RAM.

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, and sensors are hardwired to the input terminals. Output devices such as small motors, motor starters, solenoid valves, and indicator lights are hardwired to the output terminals. To electrically isolate the internal components from the input and output terminals, PLCs commonly employ an optical isolator, which uses light to couple the circuits together.
The external devices are also referred to as "fi eld" or "real-world" inputs and outputs. The terms *fi eld* or *real world* 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.

2. A programmable logic controller operates in real time in that an event taking place in the fi eld will result in an operation or output taking place. The RUN operation for the process control scheme can be described by the following sequence of events:

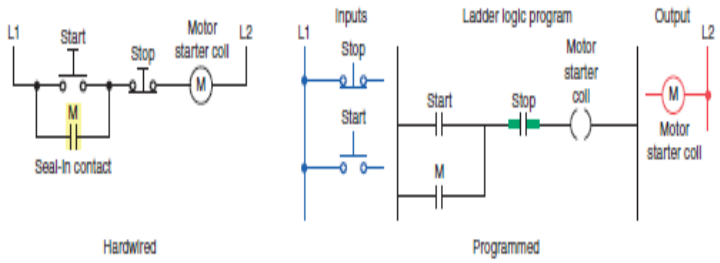
• First, the pressure switch, temperature switch, and pushbutton inputs are examined and their status is recorded in the controller's memory.
• A closed contact is recorded in memory as logic 1 and an open contact as logic 0.
• Next the ladder diagram is evaluated, with each internal contact given an OPEN or CLOSED status according to its recorded 1 or 0 state.
• When the states of the input contacts provide logic continuity from left to right across the rung, the output coil memory location is given a logic 1 value and the output module interface contacts will close.
• When there is no logic continuity of 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 scan time, the time required for one full cycle, provides a measure of the speed of response of the PLC.
• Generally, the output memory location is updated during the scan but the actual output is not updated until the end of the program scan during the I/O scan.



Module -2

3. Seal-In Circuits

Seal-in, or *holding,* circuits are very common in both relay logic and PLC logic. Essentially, a seal-in circuit is a method of maintaining current flow after a momentary switch has been pressed and released. In these types of circuits, the seal-in contact is usually in parallel with the momentary device
The motor stop/start circuit shown in Figure is a typical example of a seal-in circuit. The hardwired circuit consists of a normally closed stop button in series with a normally open start button. The seal-in auxiliary contact of the starter is connected in parallel with the start button to keep the starter coil energized when the start button is released. When this circuit is programmed into a PLC, both the start and stop buttons are examined for a closed condition because both buttons must be closed to cause the motor starter to operate.



3 b. Electromagnetic latching relays are designed to hold the relay closed after power has been removed from the coil.
Latching relays are used where it is necessary for contacts to stay open and/or closed even though the coil is energized only momentarily. Figure shows a latching relay that uses two coils. The *latch* coil is momentarily energized to set the latch and hold the relay in the latched position. The *unlatch* or release coil is momentarily energized to disengage the mechanical latch and return the relay to the unlatched position.

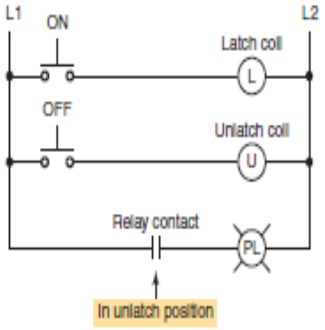


Figure shows a hardwired control circuit for an electromagnetic latching relay. The operation of the circuit can be summarized as follows:

• The contact is shown with the relay in the *unlatched* position.
• In this state the circuit to the pilot light is open and so the light is off.
• When the ON button is *momentarily* actuated, the latch coil is energized to set the relay to its latched position.
• The contacts close, completing the circuit to the pilot light, and so the light is switched on.
• The relay coil does *not* have to be continuously energized to hold the contacts closed and keep the light on.
• The only way to switch the lamp off is to actuate the OFF button, which will energize the unlatch coil and return the contacts to their open, unlatched state.
• In cases of power loss, the relay will remain in its original latched or unlatched state when power is restored.

Module -3
4 a

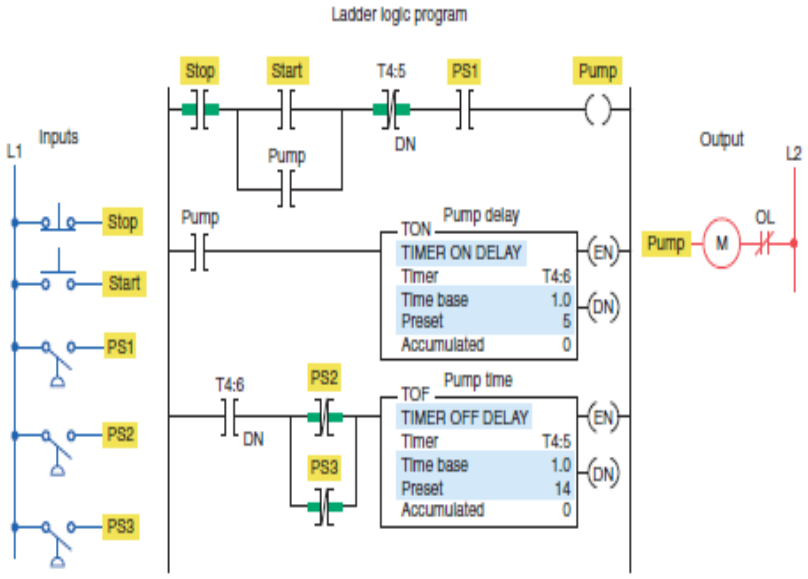


Figure 7-24 Fluid pumping process.

b. retentive timers

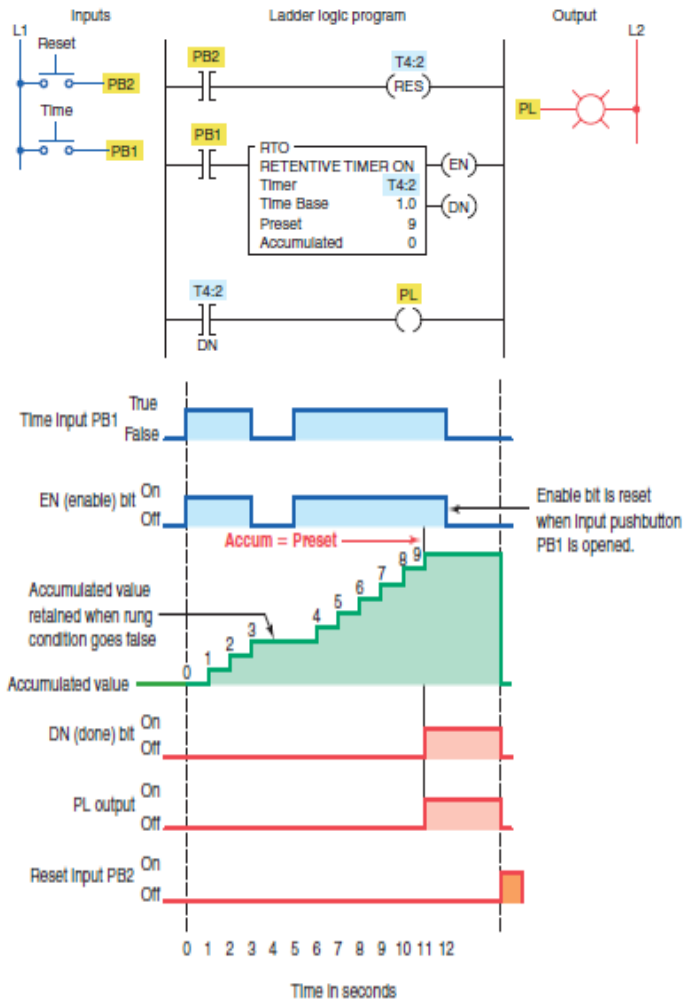


Figure shows a PLC program for a retentive on-delay timer. The operation of the program can be summarized as follows:

• The timer will start to time when time pushbutton PB1 is closed.

• If the pushbutton is closed for 3 seconds and then opened for 3 seconds, the timer accumulated value will remain at 3 seconds.

• When the time pushbutton is closed again, the timer picks up the time at 3 seconds and continues timing.

- When the accumulated value (9) equals the preset value (9), the timer done bit T4:2/DN is set to 1 and the pilot light output PL is switched on.

• Whenever the momentary reset pushbutton is closed the timer accumulated value is reset to 0.

The timing operation can be summarized as follows:

• When the timing rung is true (PB1 closed) the timer will commence timing.

• If the timing rung goes false the timer will stop timing but will recommence timing for the stored accumulated value each time the rung goes true.

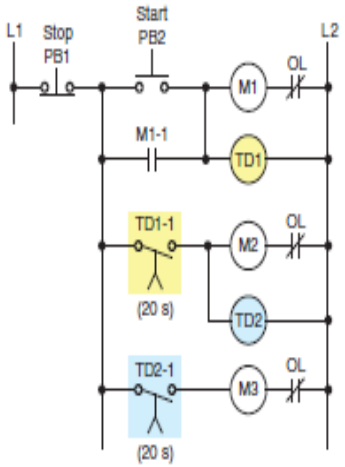
• When the reset PB2 is closed, the T4:2/DN bit is reset to 0 and turns the pilot light output off. The accumulated value is also reset and held at zero until the reset pushbutton is opened.

Cascading Timers

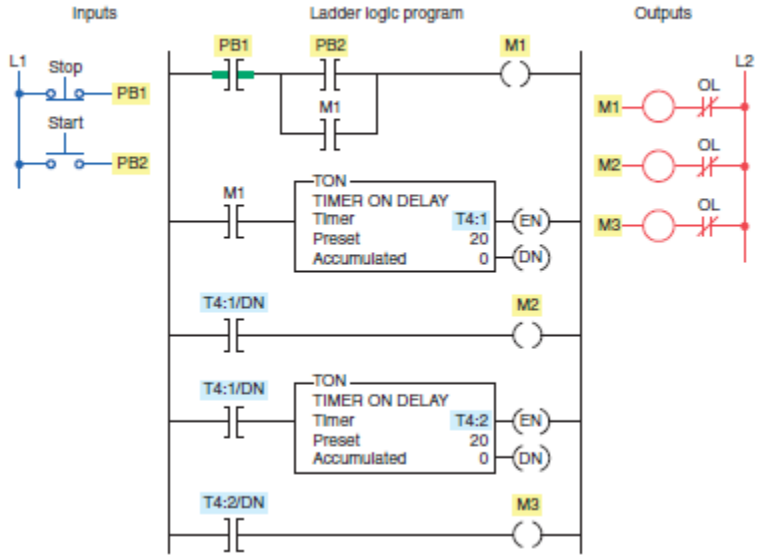
The programming of two or more timers together is called *cascading.* Timers can be interconnected, or cascaded, to satisfy a number of logic control functions.

Figure 7-30 shows how three motors can be started automatically in sequence with a 20 s time delay between each using two hardwired on-delay timers. The operation of the circuit can be summarized as follows:

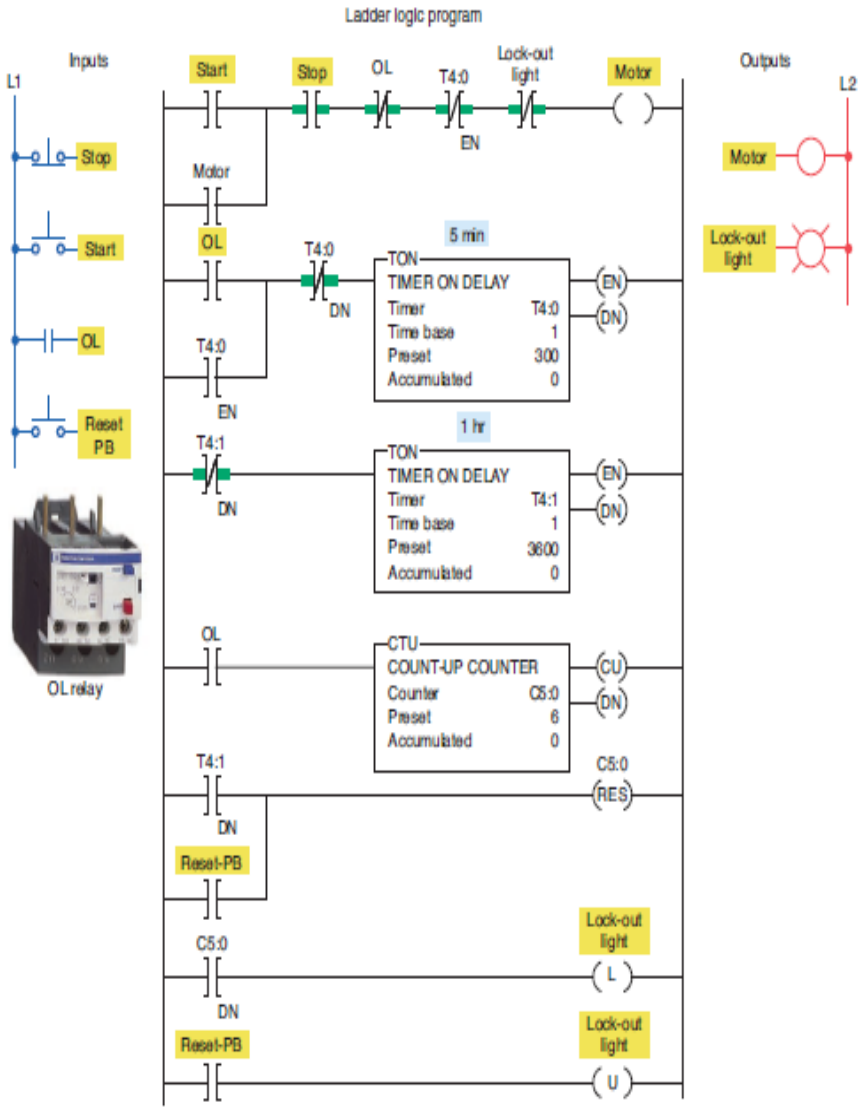
• Motor starter coil M1 is energized when the momentary start pushbutton PB2 is actuated.
• As a result, motor 1 starts, contact M1-1 closes to seal in M1, and timer coil TD1 is energized to begin the fi rst time-delay period.
• After the preset time period of 20 s, TD1-1 contact closes to energize motor starter coil M2.
• As a result, motor 2 starts and timer coil TD2 is energized to begin the second time-delay period.



5 a 24 Hour Program



5 b



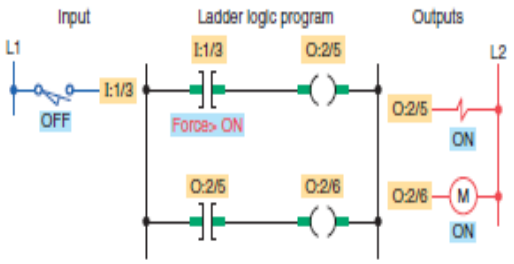
6a The operation of the program can be summarized as follows:

• The processor ignores the actual state of input limit switch I:1/3.

• Although limit switch I:1/3 is off (0 or false) the processor considers it as being in the on (1 or true) state.

• The program scan records this, and the program is executed with this forced status.

• In other words, the program is executed as if the limit switch were actually closed



The operation of the program can be summarized as follows:
• The processor ignores the actual state of solenoid output O:2/5.
• The programming device sets the force state in the output force data fi le and the PLC implements the force to turn solenoid output O:2/5 on even though the output image table fi le indicates that the user logic is setting the point to off.
• M output O:2/6 remains off because the status bit of output O:2/5 is not affected by the force instruction.
• Not all brands of PLCs operate this way. For example, forcing an output with a GE Fanuc controller will cause the contacts that have the same address as the output to also change to the appropriate state.

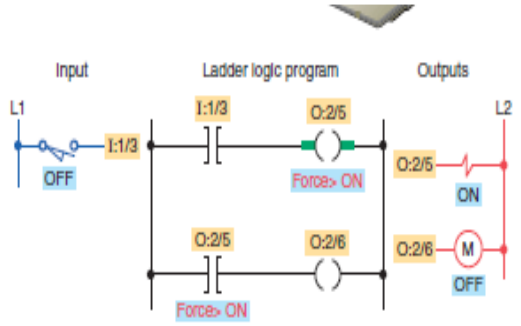


Figure 9-20 Forcing an output on.

b. The *temporary end (TND)* instruction is an output instruction used to progressively debug a program or conditionally omit the balance of your current program fi le or subroutines. When rung conditions are true, this instruction stops the program scan, updates the I/O, and resumes scanning at rung 0 of the main program fi le.

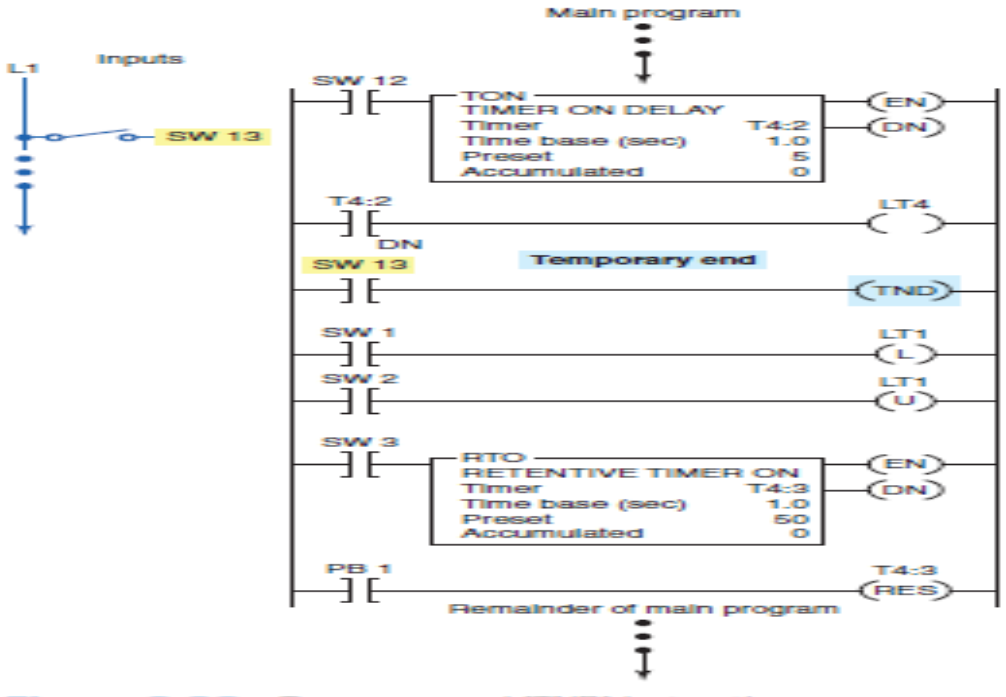


Figure 9-28 Temporary end (TND) instruction.

7.a The *fi le arithmetic and logic (FAL)* instruction is used to copy data from one fi le to another and to do fi le math and fi le logic. This instruction is available only on Allen- Bradley PLC-5 and ControlLogix platforms. An example of the FAL instruction is shown in Figure 10-11 .
The basic operation of the FAL instruction is similar in all functions and requires the following parameters and PLC-5 addresses to be entered in the instruction:

Control

• Is the fi rst entry and the address of the control structure in the control area (R) of processor memory.
• The processor uses this information to run the instruction.
• The default fi le for the control fi le is data fi le 6.
• The control element for the FAL instruction must be unique for that instruction and may not be used to control any other instruction.
• The control element is made up of three words.
• The control word uses four control bits: bit 15 (enable bit), bit 13 (done bit), bit 11 (error bit), and bit 10 (unload bit).

Length

• Is the second entry and represents the fi le length. • This entry will be in words, except for the floating-point file, for which the length is in elements. (A floating-point element consists of two words.)
• The maximum length possible is 1000 elements. Enter any decimal number from 1 to 1000.

Position

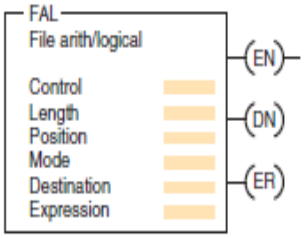
• Is the third entry and represents the current location in the data block that the processor is accessing.
• It points to the word being operated on.
• The position starts with 0 and indexes to 1 less than the fi le length.
• You generally enter a 0 to start at the beginning of a fi le. You may also enter another position at which you want the FAL to start its operation.
• When the instruction resets, however, it will reset the position to 0.
• You can manipulate the position from the program.

Mode

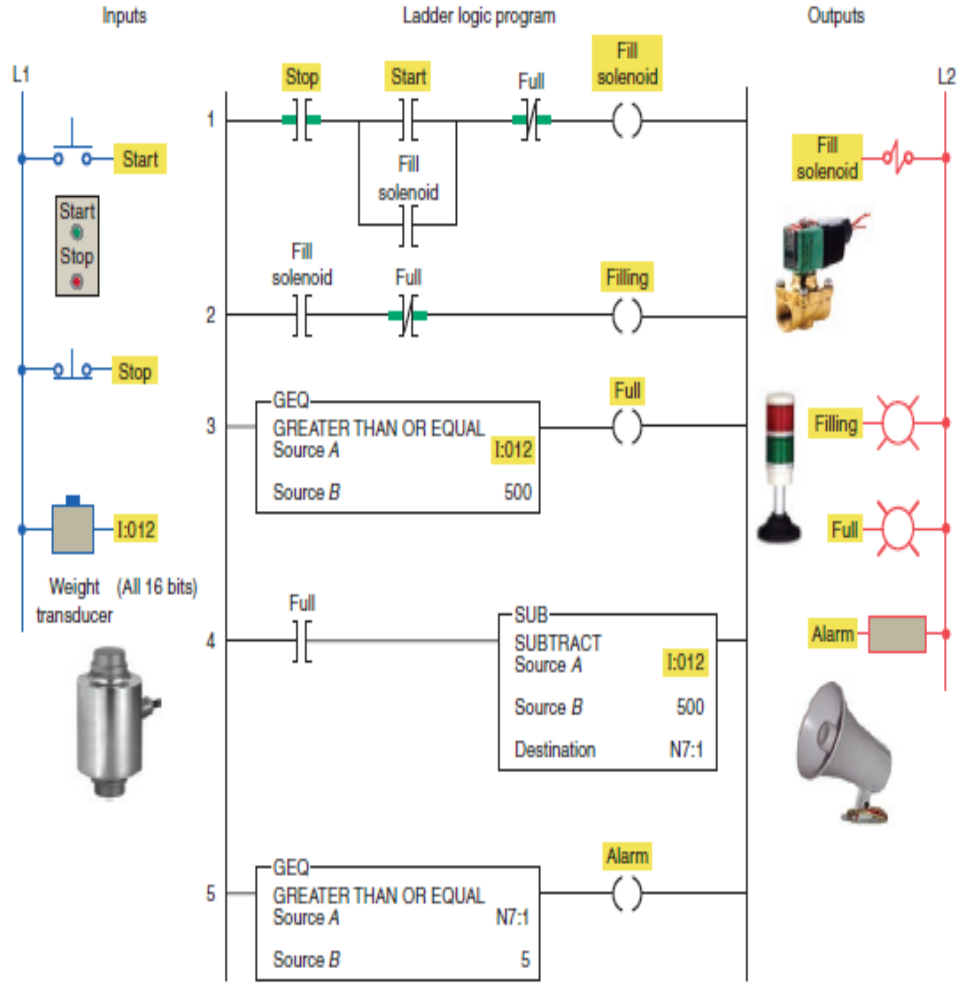
• Is the fourth entry and represents the number of file elements operated on per program scan. There are three choices: all mode, numeric mode, and incremental mode.

All Mode

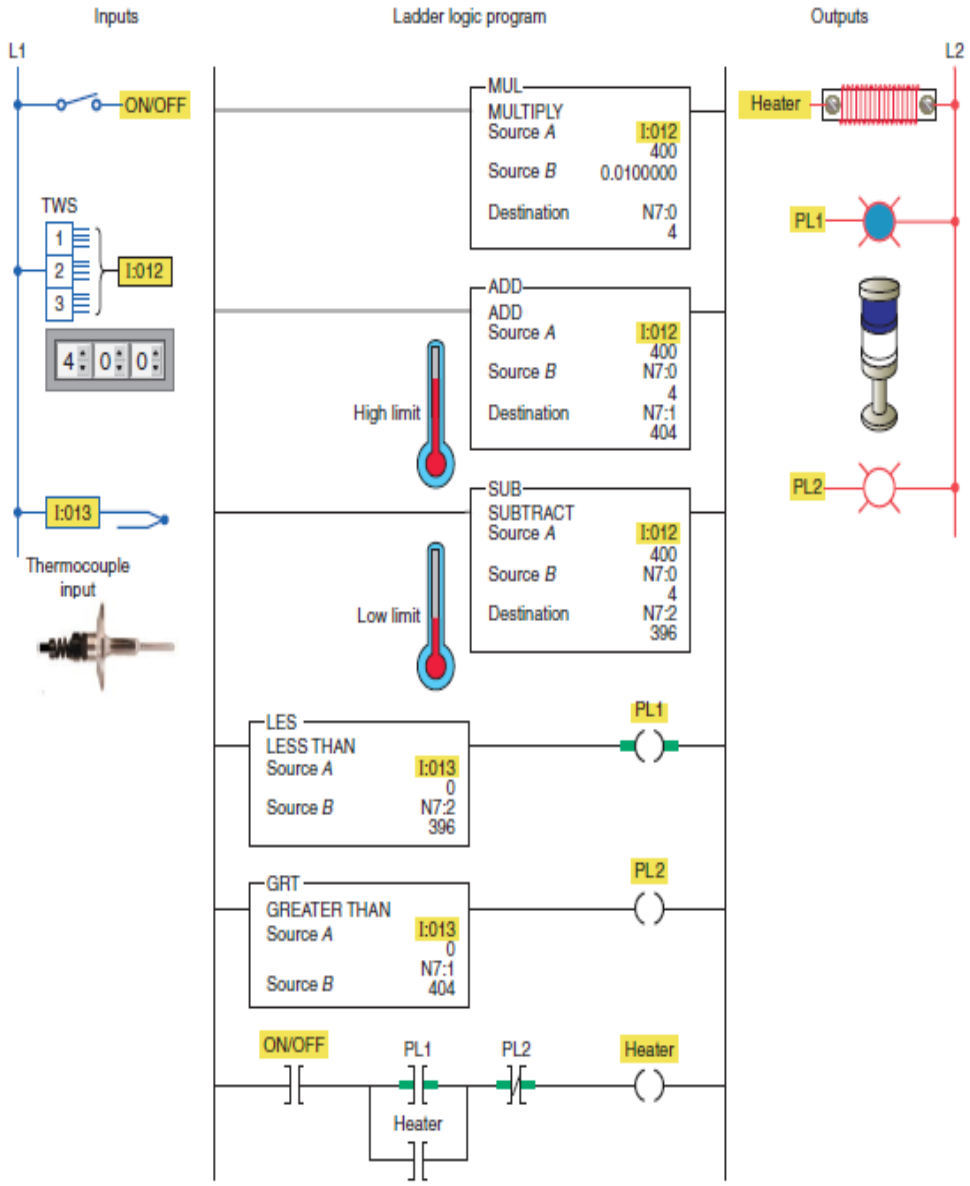
• For this mode you enter the letter *A.*
• In the all mode, the instruction will transfer the complete fi le of data in *one* scan.
• The enable (EN) bit will go true when the instruction goes true and will follow the rung condition.
• When all of the data have been transferred, the done (DN) bit will go true. This change will occur on the same scan during which the instruction goes true.
• If the instruction does not go to completion due to an error in the transfer of data (such as trying to store too large or too small a number for the data-table



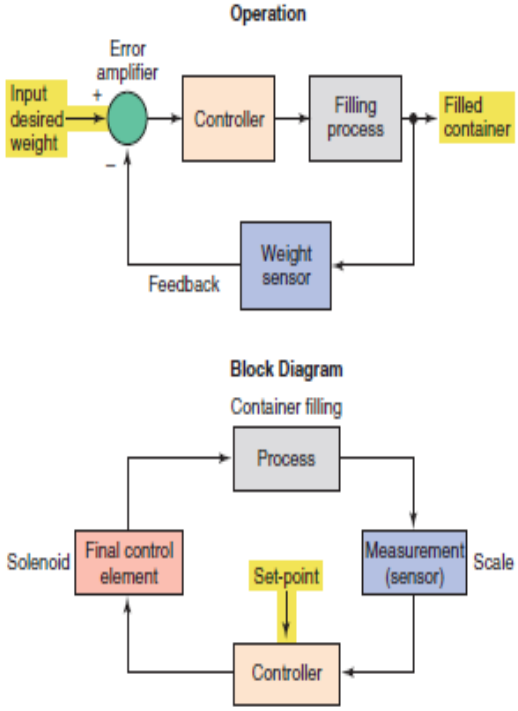
b. vessel overfill program



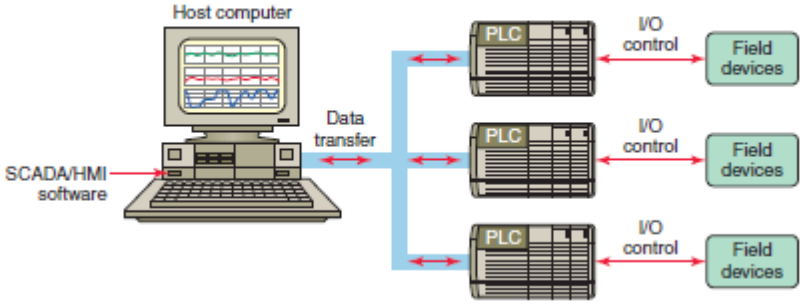
8a. temperature control



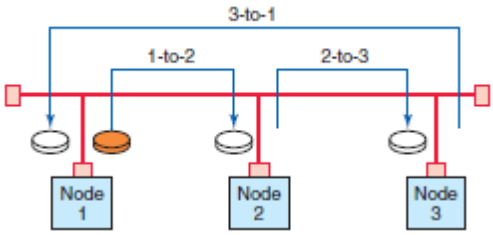
b.



10 a.Data collection is simplified by using a SCADA (supervisory control and data acquisition) system, shown in fig. Exchanging data from the plant floor to a supervisory computer allows data logging, data display,
trending, downloading of recipes, setting of selected parameters, and availability of general production data. The additional supervisory control output capabilities allow you to tweak your processes accurately for maximum efficiency. In general, unlike distributive control systems, a SCADA system usually refers to a system that coordinates but does not control processes in real time .
In a typical SCADA system, independent PLCs perform I/O control functions on field devices while being supervised by a SCADA/HMI software package running on a host computer, as illustrated in Figure 14-48 . Process control operators monitor PLC operation on the host computer and send control commands to the PLCs if required. The great advantage of a SCADA system is that data are stored automatically in a form that can be retrieved for later analysis without error or additional
work. Measurements are made under processor control and then displayed onscreen and stored to a hardcopy. Accurate measurements are easy to obtain, and there are no mechanical limitations to measurement speed.



B 1. In a *token passing* network, a node can transmit data on the network only when it has possession of a token. A token is simply a small packet that is passed from node to node as illustrated in Figure 14-32 . When a node finishes transmitting messages, it sends a special message to the next node in the sequence, granting it the token. The token passes sequentially from node to node, allowing each an opportunity to transmit without interference. Tokens usually have a time limit to prevent a single node from tying up the token for a long period of time



The access method most often used in master/slave protocols is *polling.* The master/slave network is one in which a master controller controls all communications originating from other controllers. This confi guration is illustrated in Figure and consists of several slave controllers and one master controller. Its operation can be summarized as follows:

• The master controller sends data to the slave controllers.
• When the master needs data from a slave, it will *poll* (address) the slave and wait for a response.
• No communication takes place without the master initiating it.
• Direct communication among slave devices is not possible.
• Information to be transferred between slaves must be sent first to the network master unit, which will, in turn, retransmit the message to the designated slave device.
• Master/slave networks use two pairs of conductors. One pair of wires is used for the master to transmit data and the slave to receive them. On the other pair, the slaves transmit and the master receives.