➢Chapter 10
Data Manipulation Instructions

❑Data Manipulation

- Data manipulation instructions allow numerical data stored in the controller's memory to be operated on within the control program.
- The use of data manipulation extends a controller's capability from that of simple on/off control based on binary logic, to quantitative decision making involving data comparisons, arithmetic, and conversions—which in turn can be applied to analoge and positioning control.
- There are two basic classes of instructions to accomplish data manipulation: instructions that operate on word data and those that operate on file, or block, data, which involve multiple words.
- The data manipulation instructions allow the ovement,manipulation, or storage of data in either single- or multiple-word groups from one data memory area of the PLC to another.
- Data manipulation can be placed in two broad categories: *data transfer and data comparison.*

❑Data Manipulation

➢Figure shows the Move/Logical menu tab for the SLC 500 PLC and its associated RSLogix software.

➢The commands can be summarized as follows:

MOV (Move) —Moves the source value to the destination.

MVM (Masked Move) —Moves data from a source location to a selected portion of the destination.

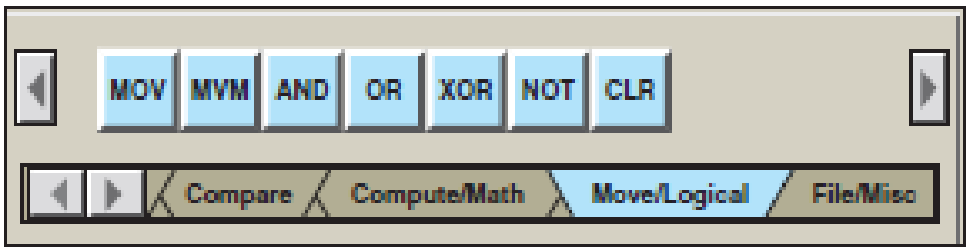
AND (And) —Performs a bitwise AND operation.

OR (Or) —Performs a bitwise OR operation.

XOR (Exclusive Or) —Performs a bitwise XOR operation.

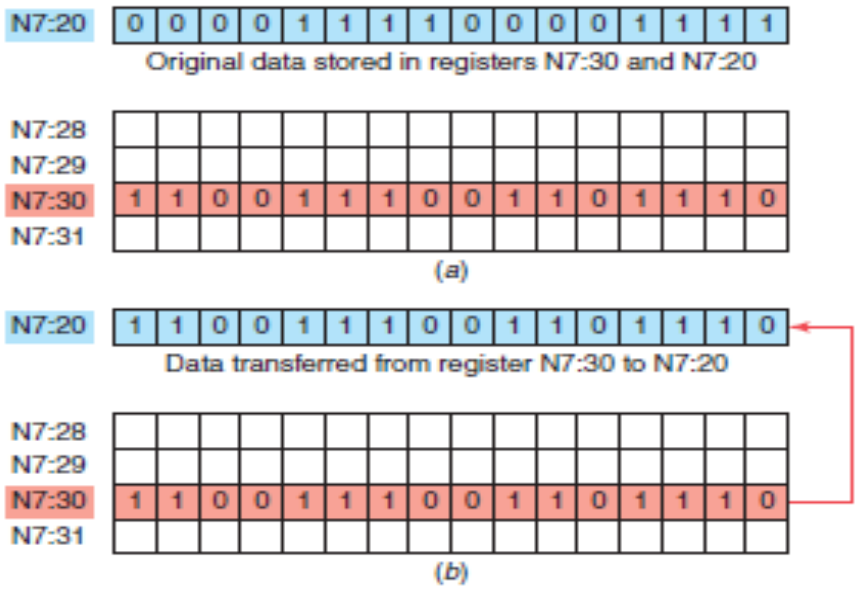
NOT (Not) —Performs a bitwise NOT operation.

CLR (Clear) —Sets all bits of a word to zero.



❑Data Transfer Operations

- Data transfer instructions simply involve the transfer of the contents from one word or register to another.
- Figure *a and b illustrate the concept of moving numerical* binary data from one memory location to another.
- Figure *a shows the original data are in register N7:30* and N7:20.
- Figure *b shows that after the data transfer* has occurred register N7:20 now holds a duplicate of the information that is in register N7:30.
- The previously existing data stored in register N7:20 have been replaced with those of N7:30.
- This process is referred to as *writing over the existing data.*



❑Data Transfer Operations

- Data transfer instructions can address almost any location in the memory.
- Prestored values can be automatically retrieved and placed in any new location.
- That location may be the preset register for a timer or counter or even an output register that controls a seven-segment display.
- SLC 500 controllers use a block-formatted *move (MOV)* instruction to accomplish data moves.
- The MOV instruction is used to copy the value in one register or word to another.
- This instruction copies data from a *source register* to a *destination register.*

❑Data Transfer Operations

➢*Figure shows an example of* the MOV instruction.

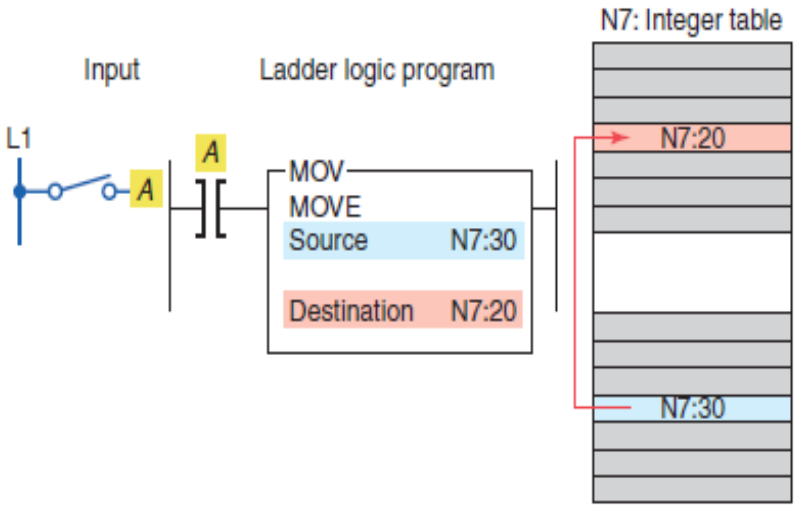
➢The operation of the program can be summarized as follows:

• When the rung is true, input switch *A closed, the* value stored at the source address, N7:30, is copied into the destination address, N7:20.

• When the rung goes false, input switch *A opened,* the destination address will retain the value unless it is changed elsewhere in the program.

• The source value remains unchanged and no data conversion occurs.

T • he instruction may be programmed with input conditions preceding it, or it may be programmed unconditionally



❑Data Transfer Operations

➢ The *move with mask (MVM) instruction differs slightly* from the MOV instruction because a *mask word is involved* in the move.

➢The operation of a mask word can be summarized as follows:

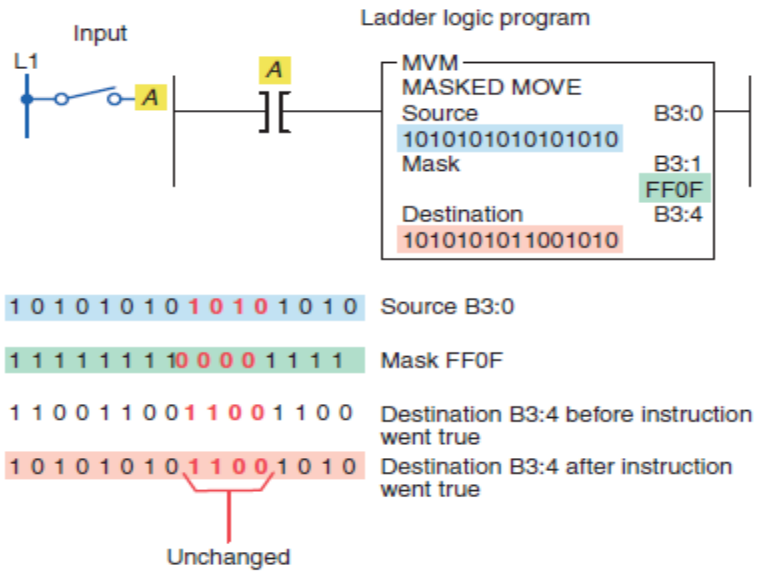
• The pattern of characters in the mask determines which source bits will be passed through to the destination address.

• The bits in the mask that are set to zero (0) do not pass data.

• Only the bits in the mask that are set to one (1) will pass the source data through to the destination.

• Bits in the destination are not affected when the corresponding bits in the mask are zero.

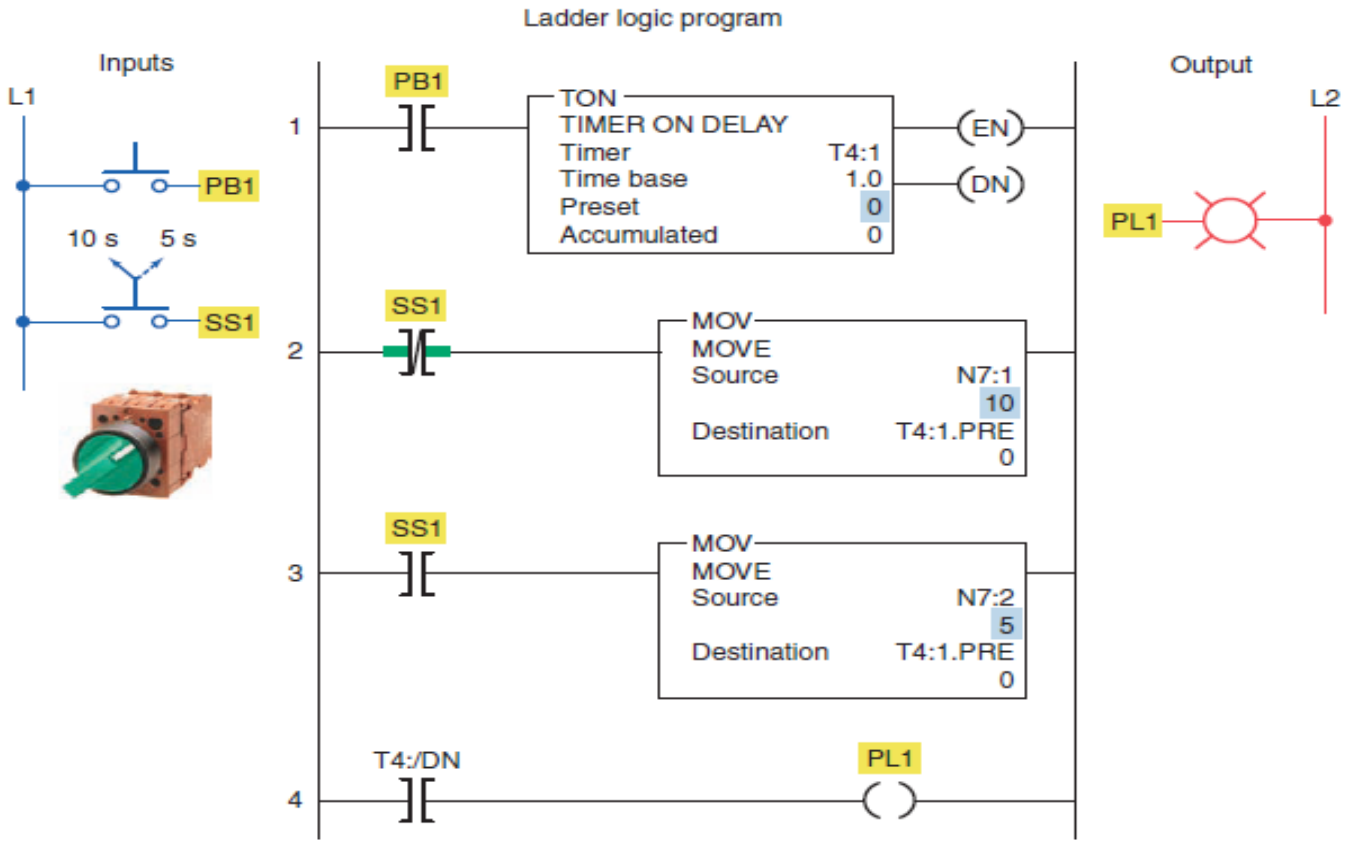
• The MVM instruction is used to copy the Desired part of a 16-bit word by masking the rest of the value.



❑Data Transfer Operations

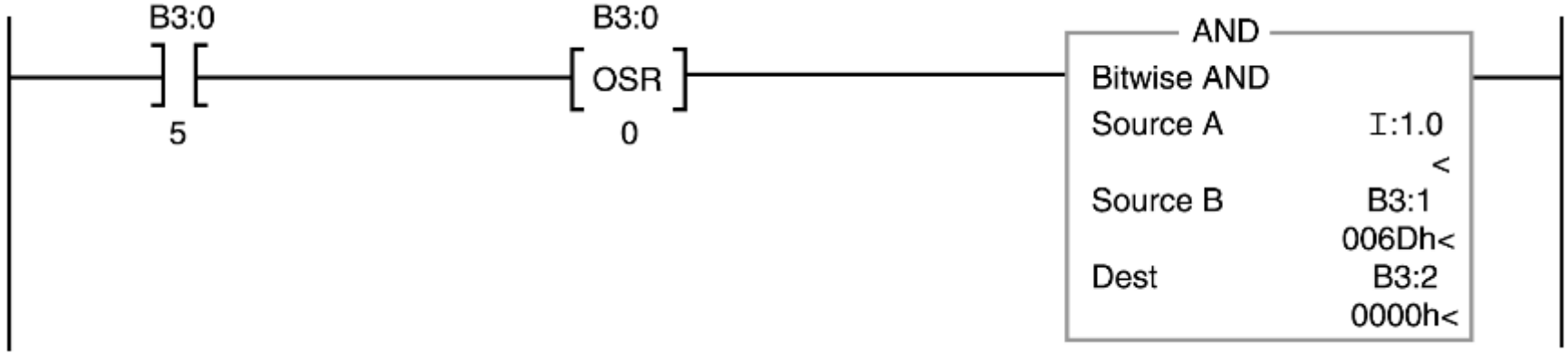
➢ The program of Figure illustrates how the move (MOV) instruction can be used to create variable preset timer values.

➢ A two-position selector switch is operated to select one of two preset timer values.



❑Data Transfer Operations

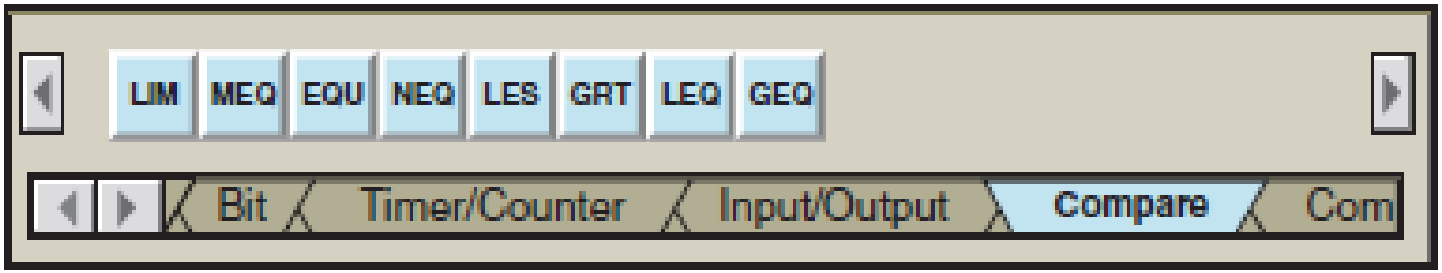
The AND,OR, XOR,NOT,CLR instruction performs a logical operation on two 16-bit words.



❑Data Compare Instructions

➢ Comparison instructions are used to test pairs of values to determine if a rung is true.

➢Figure shows the Compare menu tab for the Allen-Bradley SLC 500 PLC and its associated RSLogix software.



❑Data Compare Instructions

➢The compare instructions can be summarized as follows:

LIM (Limit test) —Tests whether one value is within the limit range of two other values.

MEQ (Masked Comparison for Equal) —Tests portions of two values to see whether they are equal.

Compares 16-bit data of a source address to 16-bit data at a reference address through a mask.

EQU (Equal) —Tests whether two values are equal.

NEQ (Not Equal) —Tests whether one value is not equal to a second value.

LES (Less Than) —Tests whether one value is less than a second value.

GRT (Greater Than) —Tests whether one value is greater than a second value.

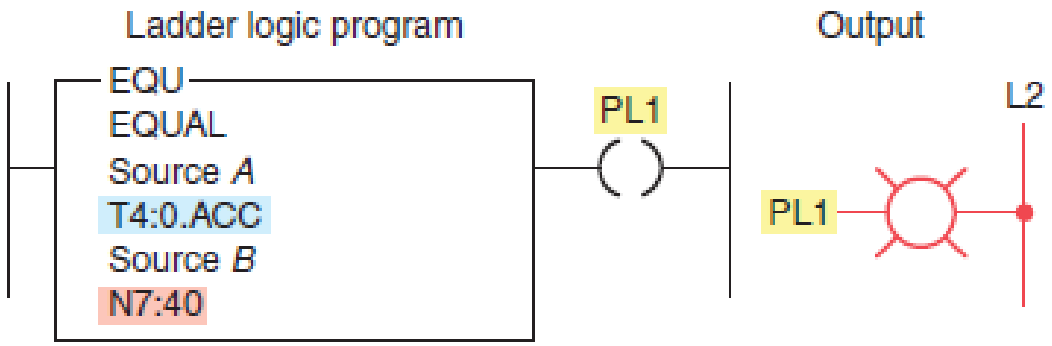
LEQ (Less Than or Equal) —Tests whether one value is less than or equal to a second value.

GEQ (Greater Than or Equal) —Tests whether one value is greater than or equal to a second value.

❑Data Compare Instructions

➢The *equal (EQU) instruction is an input instruction that* compares source *A to source B: when source A is equal* to source *B, the instruction is logically true; otherwise it* is logically false.

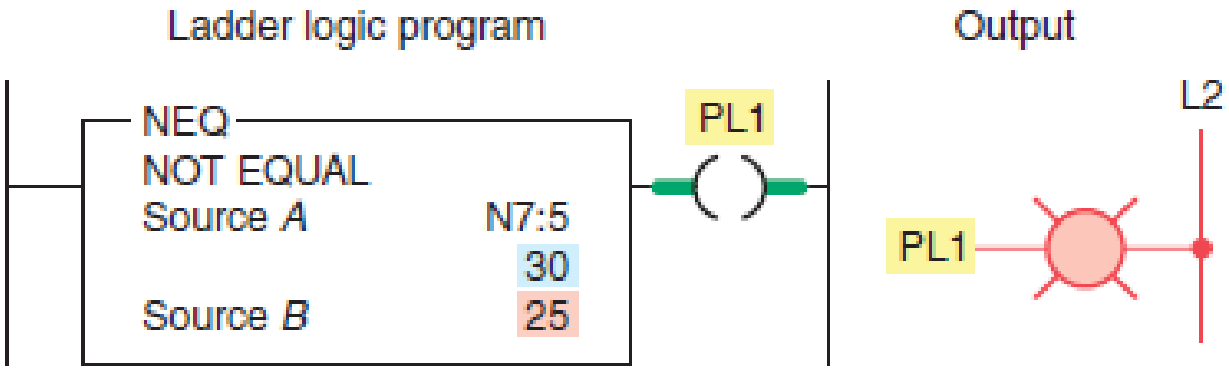
➢Figure shows an example of an EQU logic rung



❑Data Compare Instructions

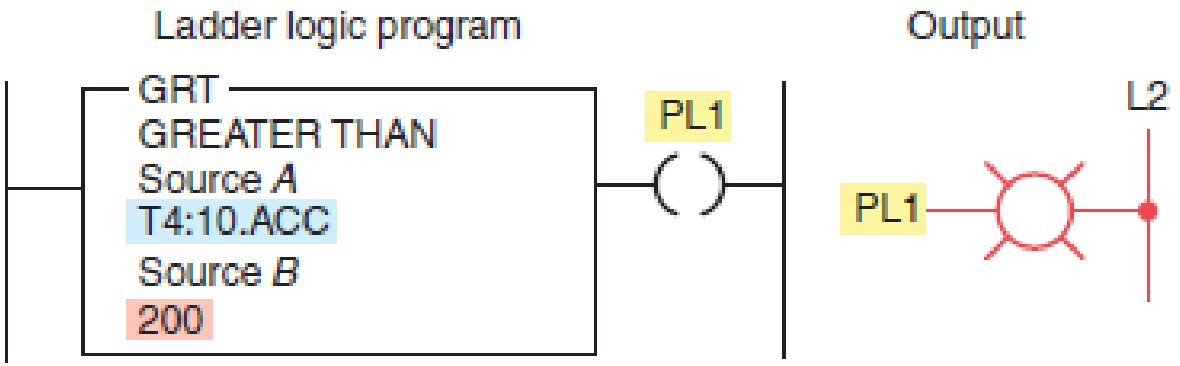
➢The *not equal (NEQ) instruction is an input instruction* that compares source *A to source B: when source A is not* equal to source *B, the instruction is logically true; otherwise* it is logically false.

➢Figure shows an example of an NEQ logic rung.



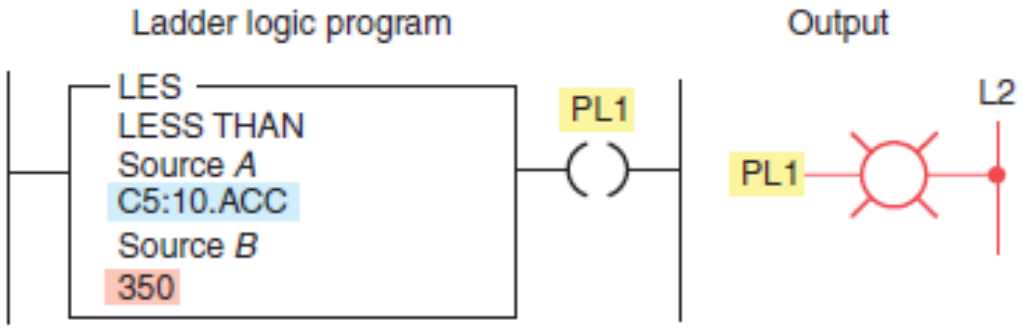
❑Data Compare Instructions

➢The *greater than (GRT) instruction is an input instruction* that compares source *A to source B: when source A* is greater than source *B, the instruction is logically true;* otherwise it is logically false



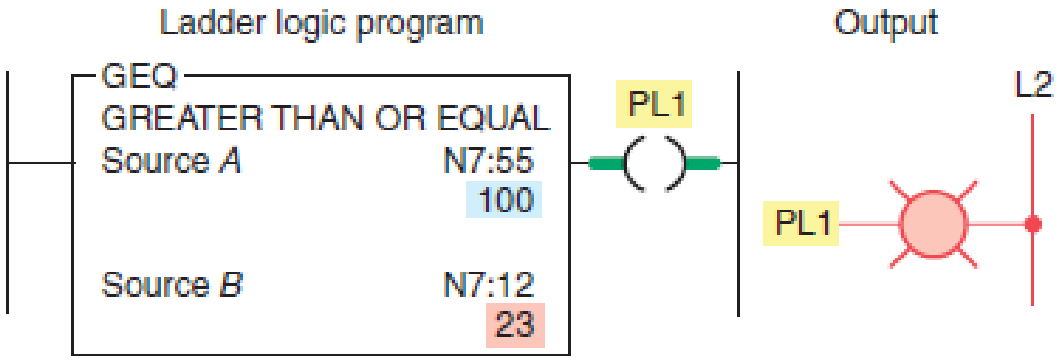
❑Data Compare Instructions

➢The *less than (LES) instruction is an input instruction* that compares source *A to source B: when source A is less* than source *B, the instruction is logically true; otherwise* it is logically false



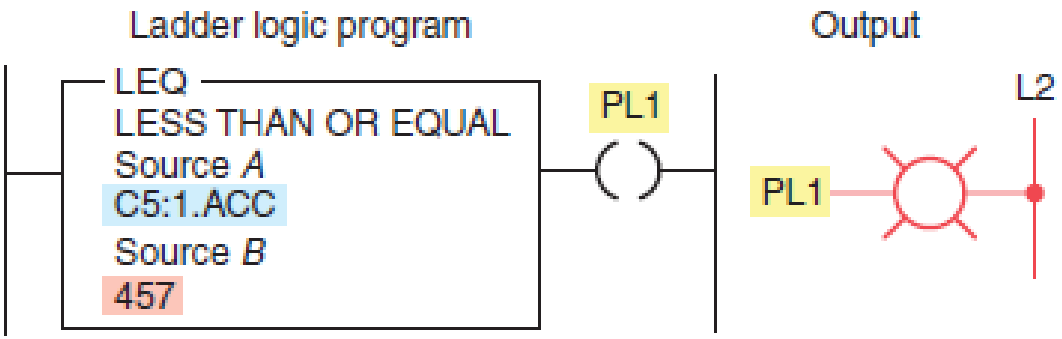
❑Data Compare Instructions

➢The *greater than or equal (GEQ) instruction is an input* instruction that compares source *A to source B: when* source *A is greater than or equal to source B, the instruction* is logically true; otherwise it is logically false



❑Data Compare Instructions

The *less than or equal (LEQ) instruction is an input* instruction that compares source *A to source B: when* source *A is less than or equal to source B, the instruction* is logically true; otherwise it is logically false



❑Data Compare Instructions

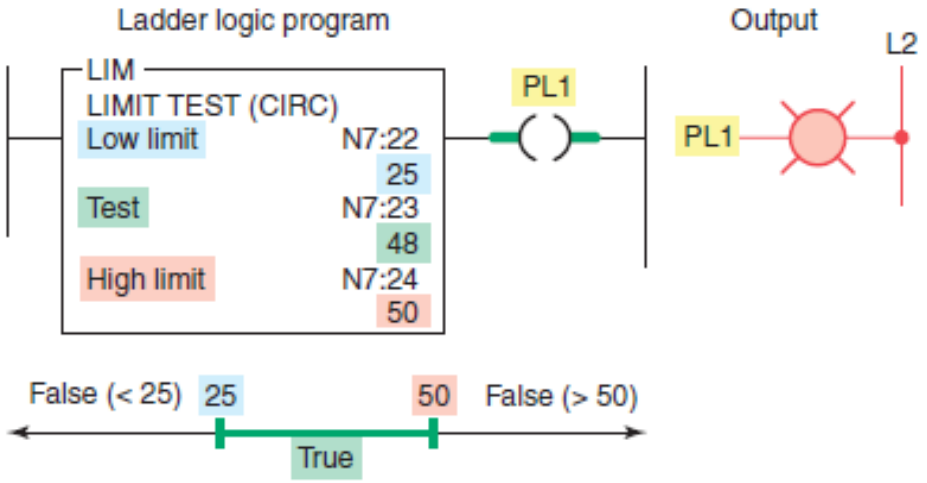
➢The *limit test (LIM) instruction is used to test whether* values are within or outside the specified range.

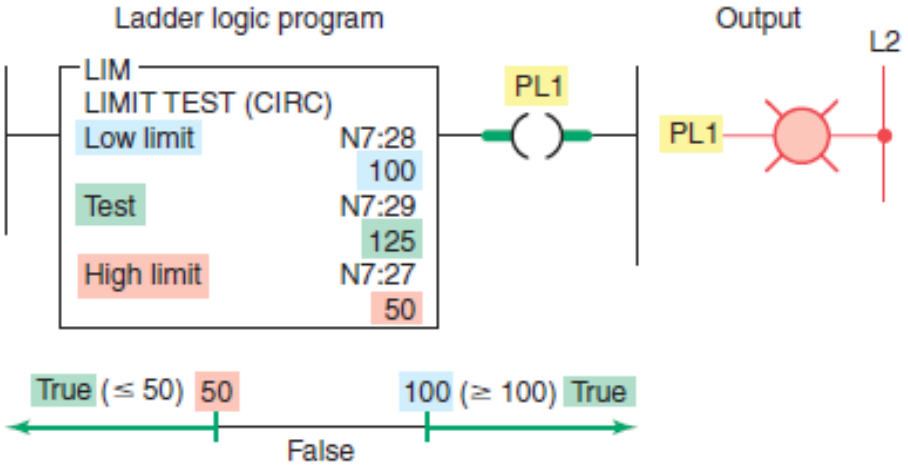
➢Applications in which the limit test instruction is used include allowing a process to operate as long as the temperature is within or outside a specified range.

➢Programming the LIM instruction consists of entering three parameters: low limit, test, and high limit. The limit test instruction functions in the following two ways:

- ***The instruction is true if —***The lower limit is equal to or less than the higher limit, and the test parameter value is equal to or inside the limits. Otherwise the instruction is false.
- ***The instruction is true if —***The lower limit has a value greater than the higher limit, and the instruction is equal to or outside the limits. Otherwise the instruction is false.

❑Data Compare Instructions



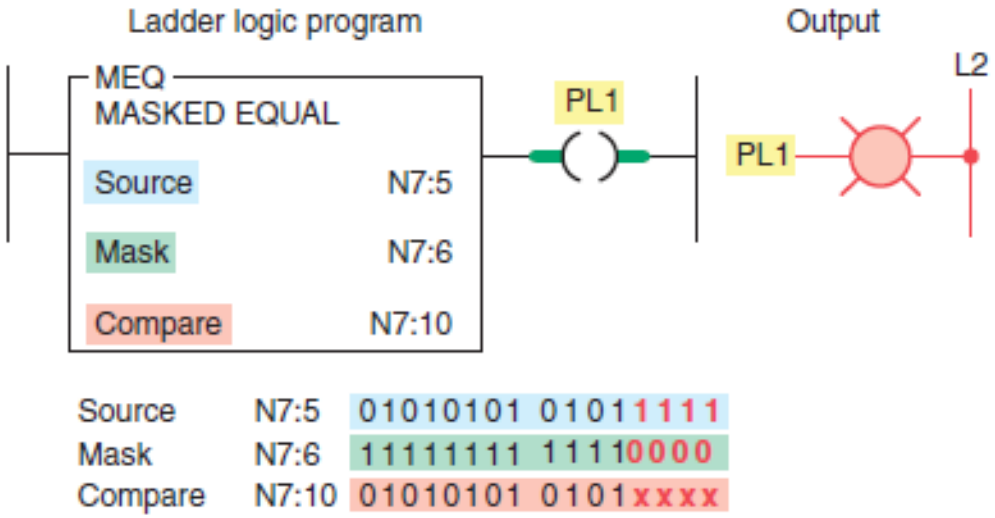


❑Data Compare Instructions

➢ The *masked comparison for equal (MEQ) instruction* compares a value from a source address with data at a compare address and allows portions of the data to be masked.

➢ One application for the MEQ instruction is to compare the correct position of up to 16 limit switches when the source contains the limit switch address and the compare stores their desired states.

➢The mask can block out the switches you don't want to compare (Figure).



❑10.4 Data Manipulation Programs

➢ Data manipulation instructions give new dimension and flexibility to the programming of control circuits.

➢For example, consider the hardwired relay-operated, time-delay circuit in Figure .

➢This circuit uses three electromechanical time-delay relays to control four solenoid valves.

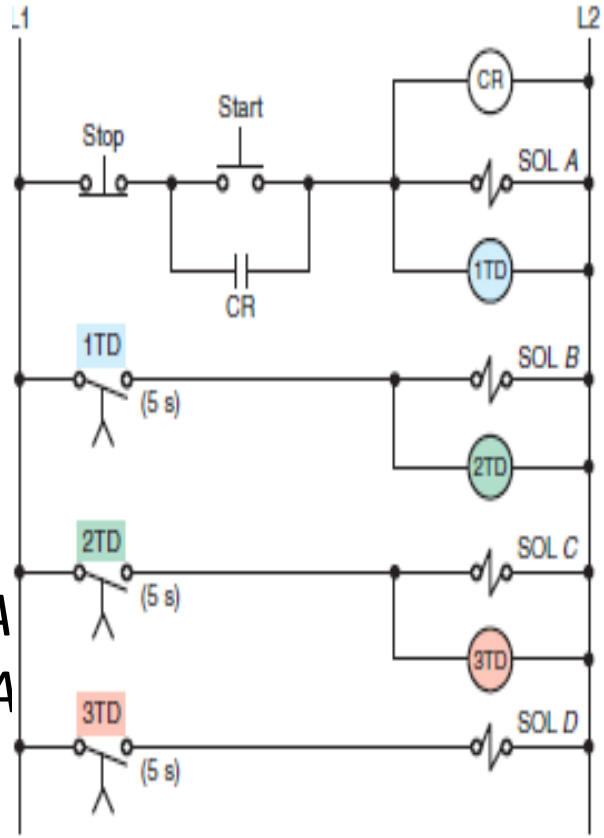
➢The operation of the hardwired circuit can be summarized as follows:

• When the momentary start pushbutton is pressed solenoid *A is energized immediately.*

• Solenoid *B is energized 5 s later than solenoid A.*

• Solenoid *C is energized 10 s later than solenoid A*

• Solenoid *D is energized 15 s later than solenoid A*

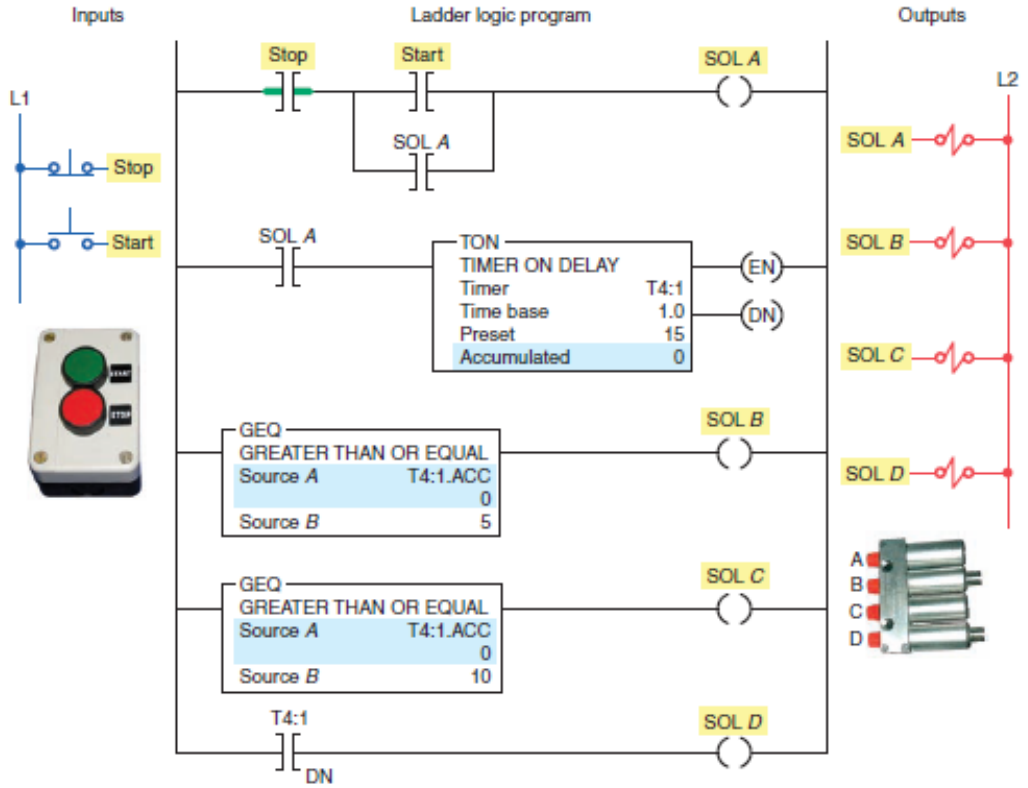


❑10.4 Data Manipulation Programs

➢The hardwired time-delay circuit could be implemented using a conventional PLC program and three internal timers.

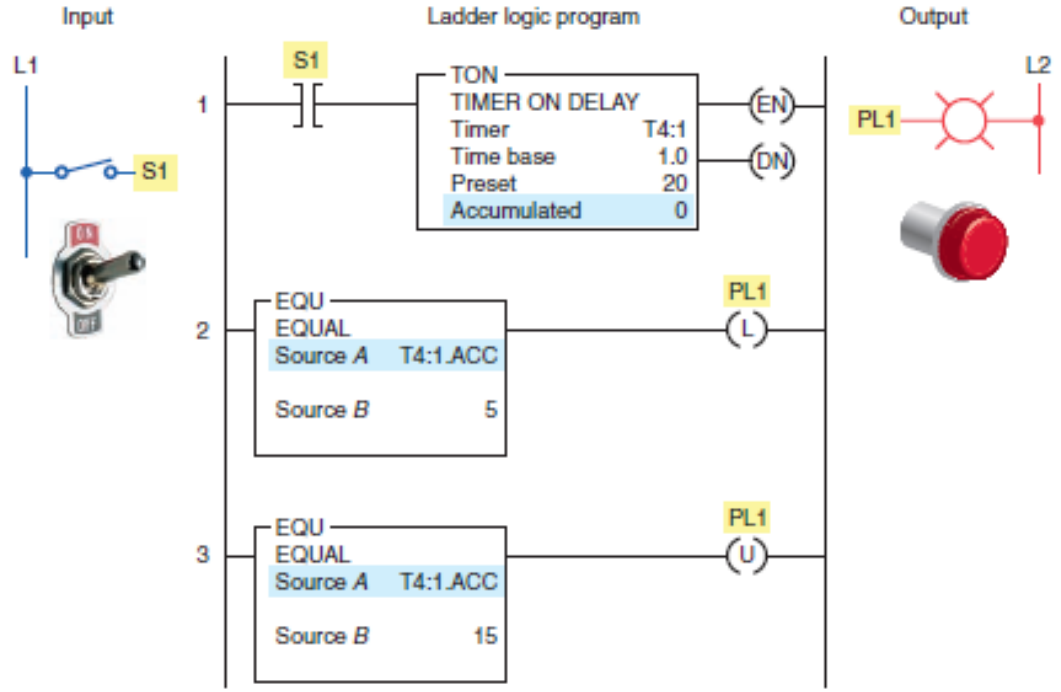
➢However, the same circuit can be programmed using only *one internal timer along with data compare instructions.*

➢Figure shows the program required to implement the circuit using only one internal timer.



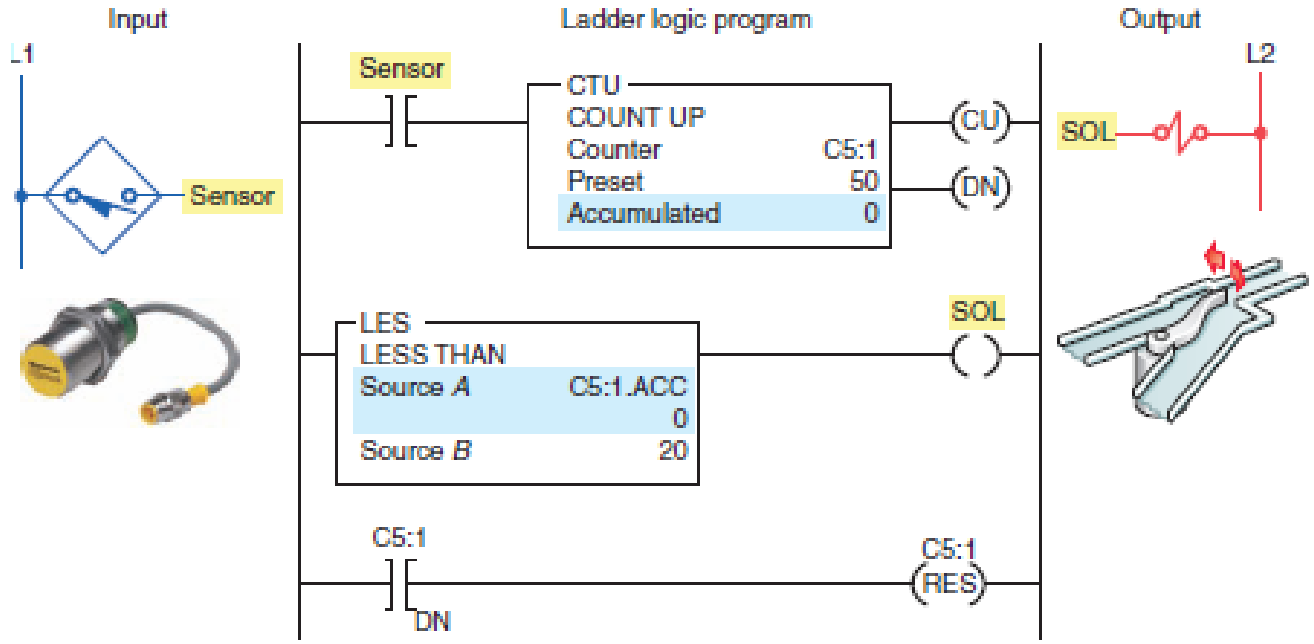
❑Data Manipulation Programs

➢Figure shows an application of an on-delay timer program implemented using the EQU instruction.



❑Data Manipulation Programs

➢Figure shows an application of an up-counter program implemented using the LES instruction.



❑Data Manipulation Programs

➢The use of comparison instructions is generally straightforward.

➢However, one precaution involves the use of these instructions in PLC programs used to control the flow in vessel filling operations (Figure).

➢This control scenario can be summarized as follows:

• The receiving vessel has its weight monitored continuously by the PLC program as it fi lls.

• When the weight reaches a preset value, the flow is cut off.

• While the vessel fills, the PLC performs a comparison between the vessel's current weight and a predetermined constant programmed in the processor.

❑Data Manipulation Programs

- If the programmer uses only the equal instruction, problems may result.
- As the vessel fills, the comparison for equality will be false.
- At the instant the vessel weight reaches the desired preset value of the equal instruction, the instruction becomes true and the flow is stopped.
- However, should the supply system leak additional material into the vessel, the total weight of the material could rise *above the preset value, causing the* instruction to go false and the vessel to overfill.
- The simplest solution to this problem is to program the comparison instruction as a greater than or equal to instruction. This way, any excess material entering the vessel will not affect the filling operation.
- It may be necessary, however, to include additional programming to indicate a serious overfill condition.

❑Numerical Data I/O Interfaces

➢The expanding data manipulation processing capabilities of PLCs led to the development of I/O interfaces known as numerical data I/O interfaces.

➢In general, numerical data I/O interfaces can be divided into two groups:

• those that provide interface to *multibit digital devices*

•*and those* that provide interface to *analoge devices.*

➢The multibit digital devices are like the discrete I/O because processed signals are discrete (on/off).

➢The difference is that, with the discrete I/O, only a *single bit is* required to read an input or control an output.

➢Multibit interfaces allow a group of bits to be input or output *as a unit.*

➢*They can be used to accommodate devices that require* BCD inputs or outputs.

❑Numerical Data I/O Interfaces

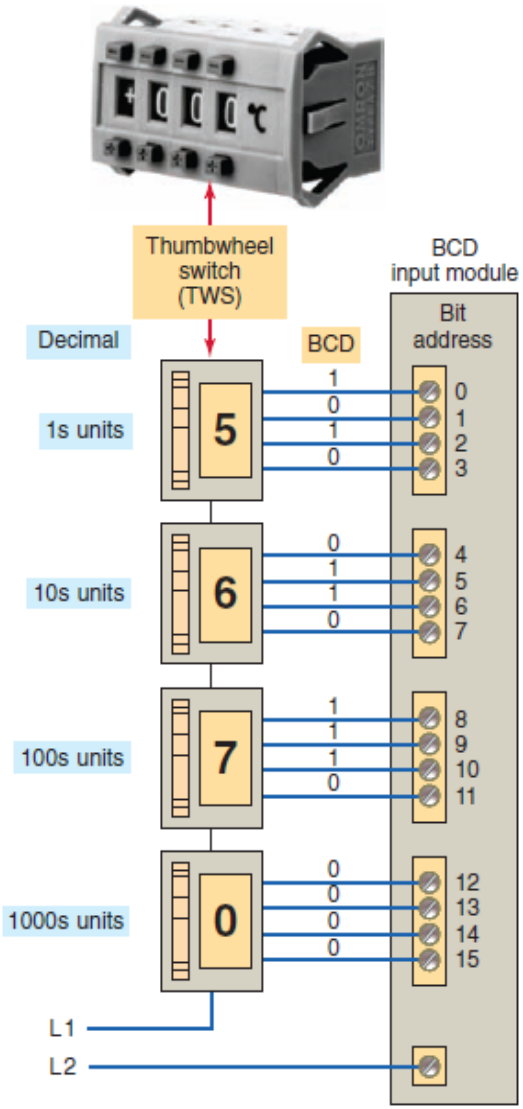
➢The *thumbwheel switches (TWS), shown in Figure* , are typical BCD input devices.

➢Each one of the four switches provides four binary that correspond to the decimal number selected on the switch.

➢The conversion from a single decimal digit to four Binary digits is performed by the TWS device.

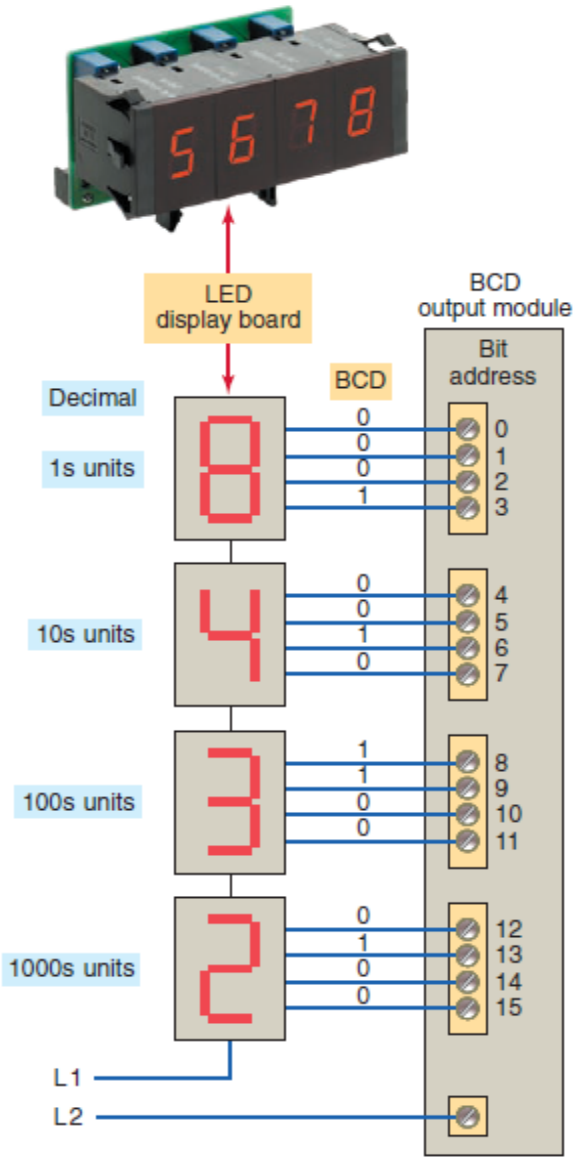
➢The BCD input module allows the processor to accept the 4-bit digital codes and input their data into specific register or word locations in memory to be used by the control program.

➢Data manipulation instructions can be used to access the data from the input module allowing a person to change set points, timer, or counter presets *externally without* modifying the control program.



❑Numerical Data I/O Interfaces

- The *seven-segment LED display board, shown in Figure* , is a typical Binary Coded Decimal (BCD) output device.
- It displays a decimal number that corresponds to the BCD value it receives at its input.
- Conversion of the four binary bits to a single decimal digit on the display is performed by the LED display device.
- The BCD output module is used to output data from a specific register or word location in memory.
- This type of output module enables a PLC to operate devices that require BCD coded signals.

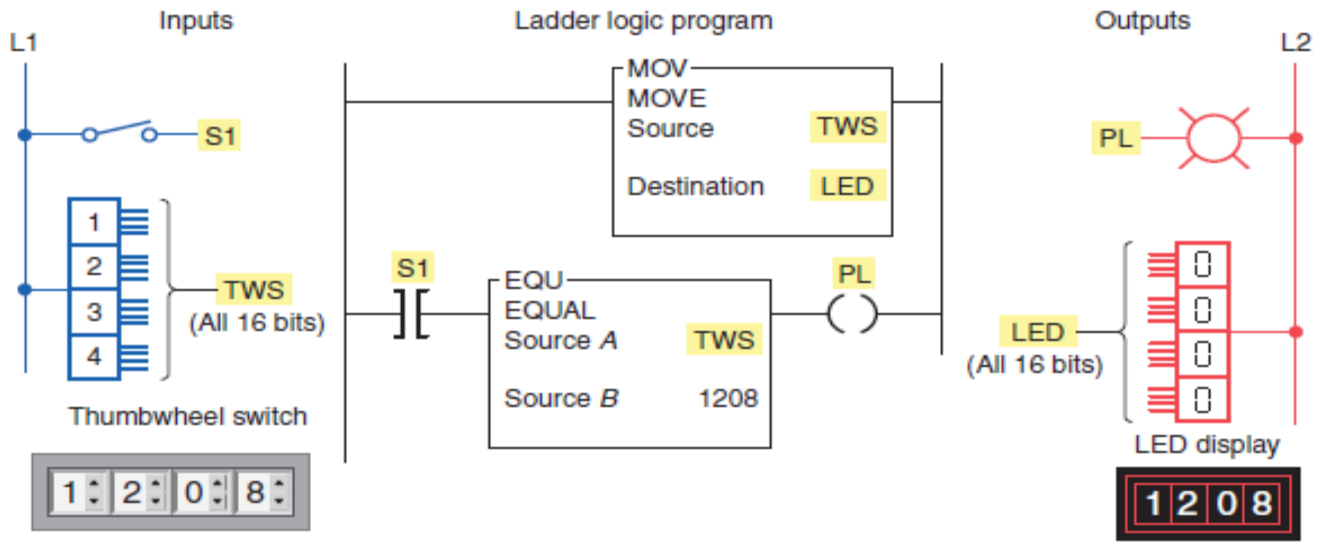


❑Numerical Data I/O Interfaces

➢Figure shows a PLC program that uses a BCD input interface module connected to a thumbwheel switch and a BCD output interface module connected to an LED display board.

➢The program is designed so that the LEDs display the setting of the thumbwheel switch.

➢Both the MOV and EQU instructions form part of the program.



❑Numerical Data I/O Interfaces

➢Analoge modules convert analoge signals to 16-bit digital signals (input) or 16-bit digital signals to analoge values (output).

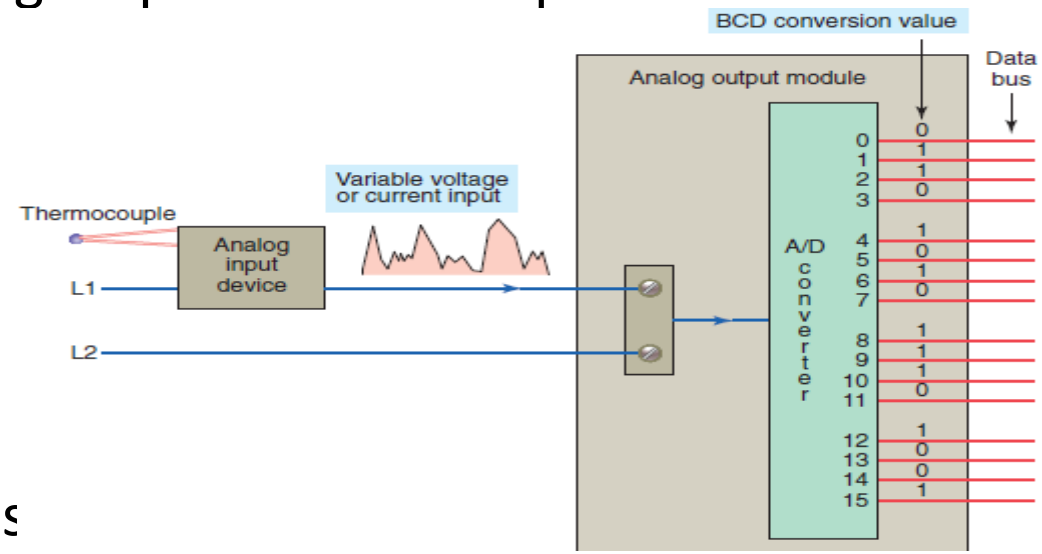
➢An analoge I/O will allow monitoring and control of analoge voltages and currents.

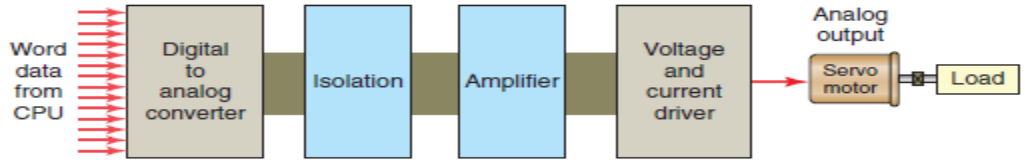
➢Figure illustrates how an analoge input interface operates.

➢The operation of this input module can be summarized as follows:

• The analoge input module contains the circuitry necessary to accept analoge voltage or current signals from field devices

•





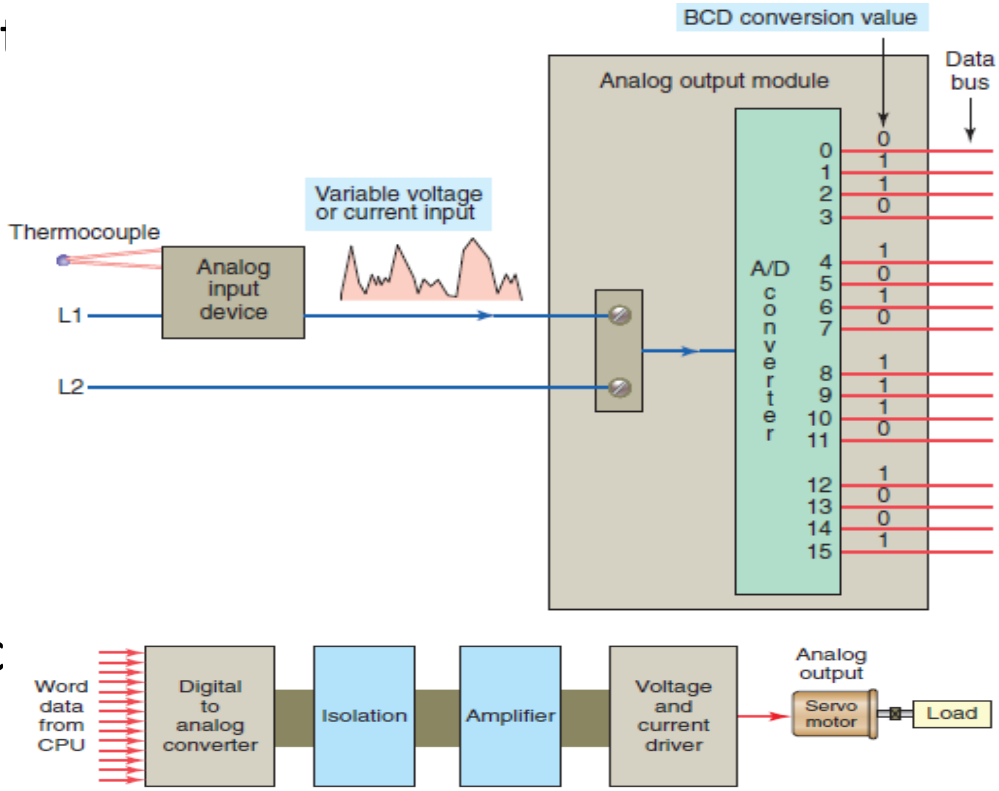
❑Numerical Data I/O Interfaces

➢The input signal is converted from an analoge to a digital value by an analoge-to-digital (A/D) converter circuit.

• The conversion value, which is proportional to the analog signal, is passed through the controller's data bus and stored in a specific register or word location in memory for la program.

• The function of the analoge output module is to accept a range of numeric values output from the PLC program and to produce a varying current or Voltage signal required to control a connected analog output devic

•



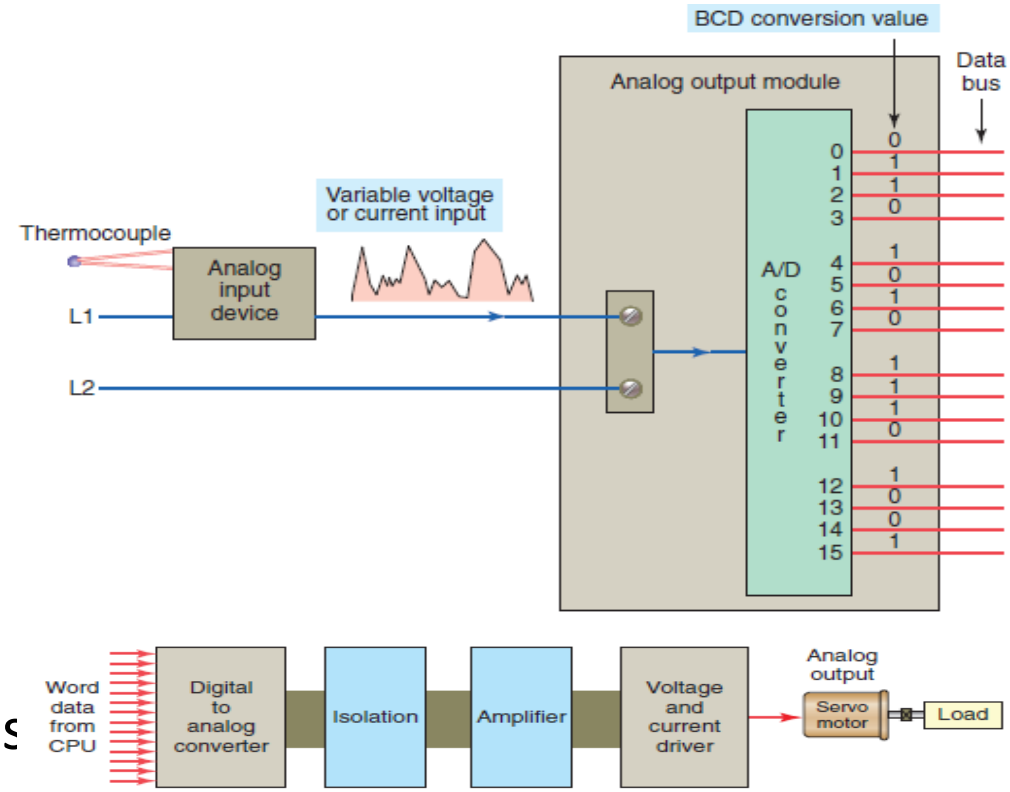
❑Numerical Data I/O Interfaces

• Data from a specific register or word location in the CPU memory are passed through the controller's data bus to the digital-to-analog (D/A) converter.

• The analog output from the D/A converter is then used to control the analog output device.

• The level of the analog signal output is based on the digital value of the data word supplied by the CPU and manipulated by the control program.

• These output interfaces normally require an external power supply that meets certain current and voltage requirements



❑Closed-Loop Control

- In open-loop control, no feedback loop is employed and system variations which cause the output to deviate from the desired value are not detected or corrected.
- A closed-loop system utilizes feedback to measure the actual system operating parameter being controlled such as temperature, pressure, flow, level, or speed.
- This feedback signal is sent back to the PLC where it is compared with the desired system set-point.
- The controller develops an error signal that initiates corrective action and drives the final output device to the desired value..

❑Closed-Loop Control

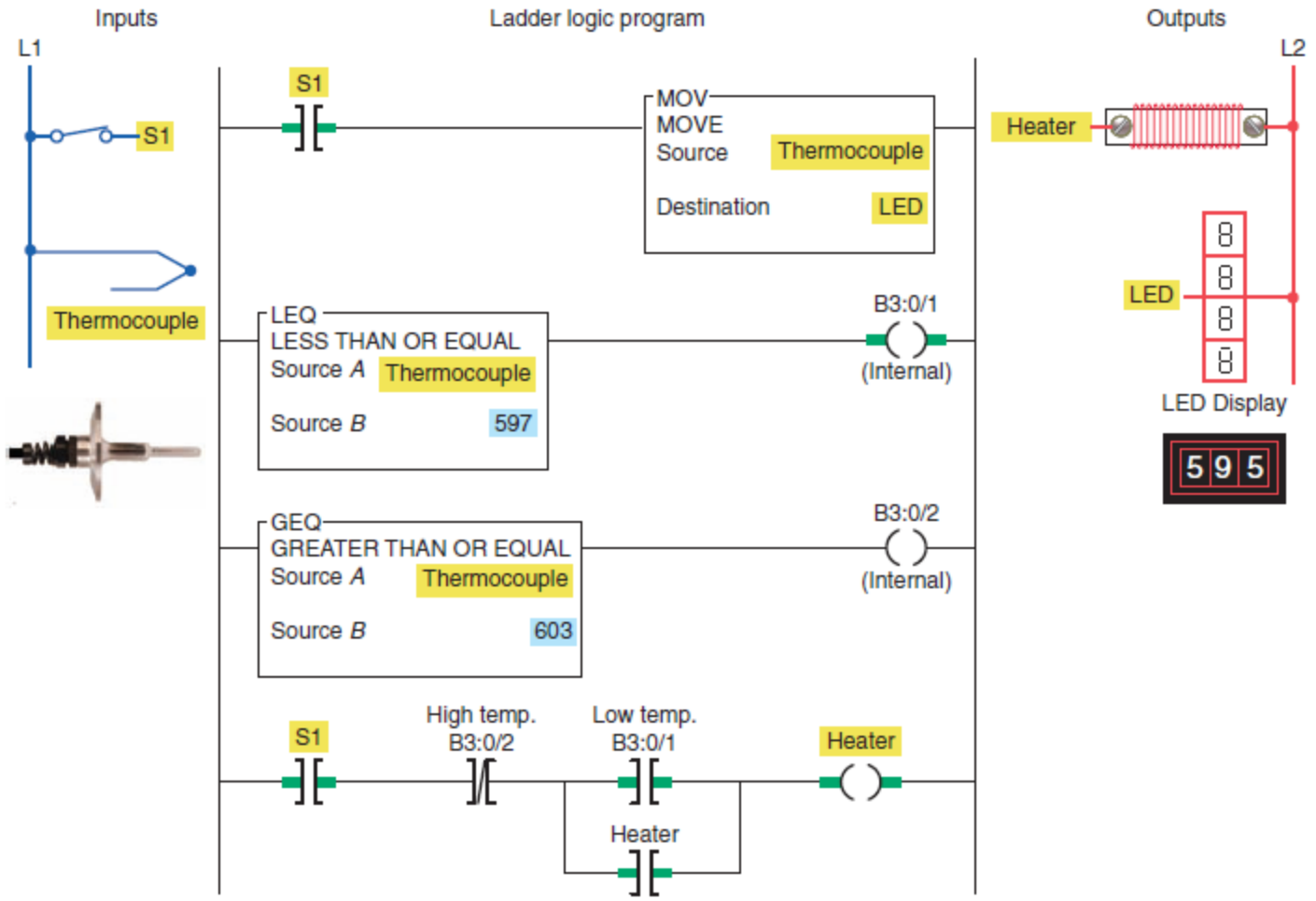
➢PLC *set-point control in its simplest form compares* an input value, such as analoge or thumbwheel inputs,to a set-point value.

➢A discrete output signal is provided if the input value is less than, equal to, or greater than the set-point value.

➢ The temperature control program of Figure is one example of set-point control.

➢In this application, a PLC is to provide for simple off/on control of the electric heating elements of an oven.

❑Closed-Loop Control

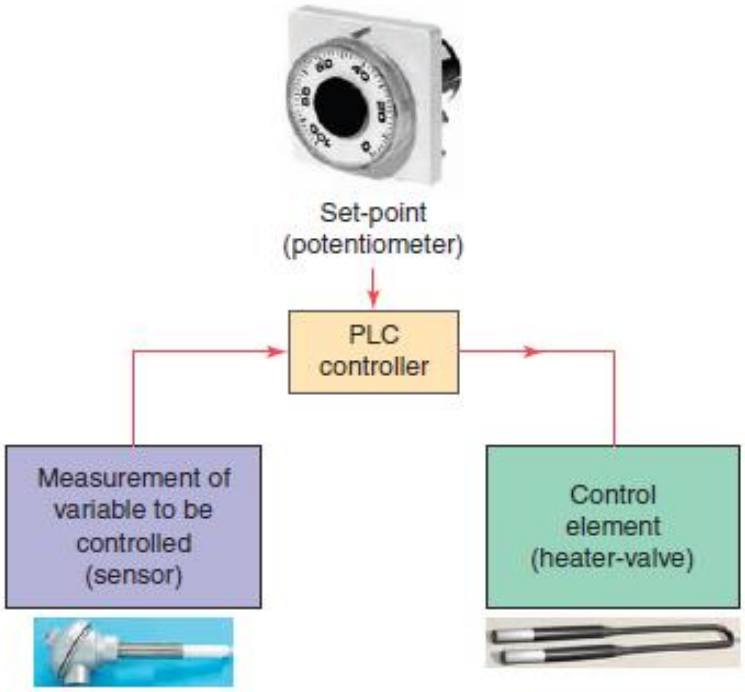


❑Closed-Loop Control

- Several set-point control schemes can be performed by different PLC models.
- These include on/off control, proportional (P) control, proportional-integral (PI) control, and proportional-integral-derivative (PID) control.
- Each involves the use of some form of closed-loop control to maintain a process characteristic such as a temperature, pressure, flow, or level at a desired value.
- When a control system is designed such that it receives operating information from the machine and makes adjustments to the machine based on this operating information, the system is said to be a closed-loop system.

❑Closed-Loop Control

- ➢The block diagram of a closed-loop control system is shown in Figure .
- ➢A measurement is made of the variable to be controlled.
- ➢This measurement is then compared to a reference point, or set-point.
- ➢If a difference (error) exists between the actual and desired levels, the
- ➢PLC control program will take the necessary corrective action.
- ➢Adjustments are made continuously by the PLC until the difference between the desired and actual output is as small as is practical.



❑Closed-Loop Control

➢With on/off PLC control (also known as *two-position* and *bang-bang control), the output or final control element* is either on or off—one for the occasion when the value of the measured variable is above the set-point and the other for the occasion when the value is below the set-point.

➢The controller will never keep the final control element in an intermediate position.

➢Most residential thermostats are on/off type controllers.

➢On/off control is inexpensive but not accurate enough for most process and machine control applications.

➢On/off control almost always means overshoot and resultant system cycling.

➢For this reason a *deadband usually exists* around the set-point.

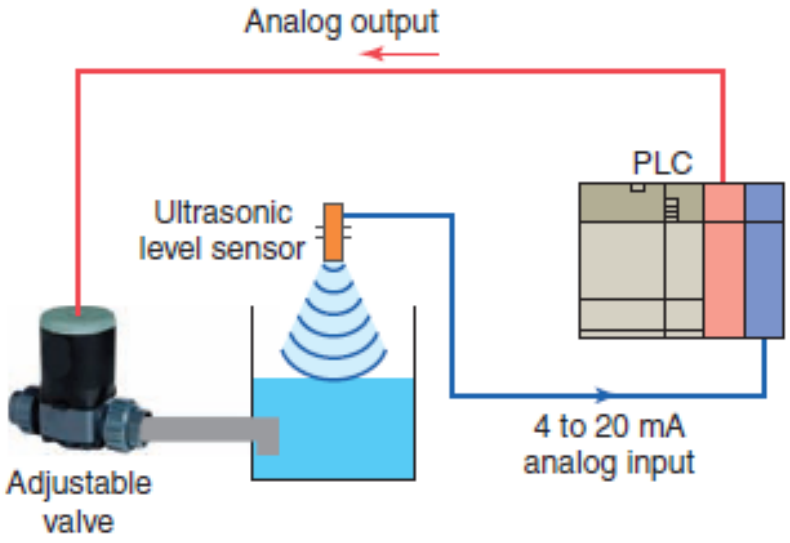
➢ The deadband or hysteresis of the control loop is the difference between the on and off operating points.

❑Closed-Loop Control

➢*Proportional controls are designed to eliminate the* hunting or cycling associated with on/off control.

➢ They allow the final control element to take intermediate positions between on and off.

➢This permits *analog control* of the final control element to vary the amount of energy to the process, depending on how much the value of the measured variable has shifted from the desired value.



❑Closed-Loop Control

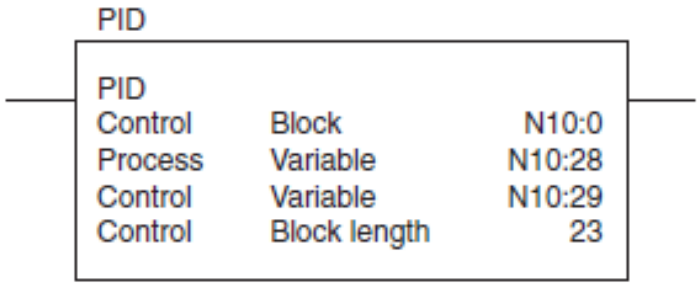
➢*Proportional-integral-derivative (PID) control is the* most sophisticated and widely used type of process control.

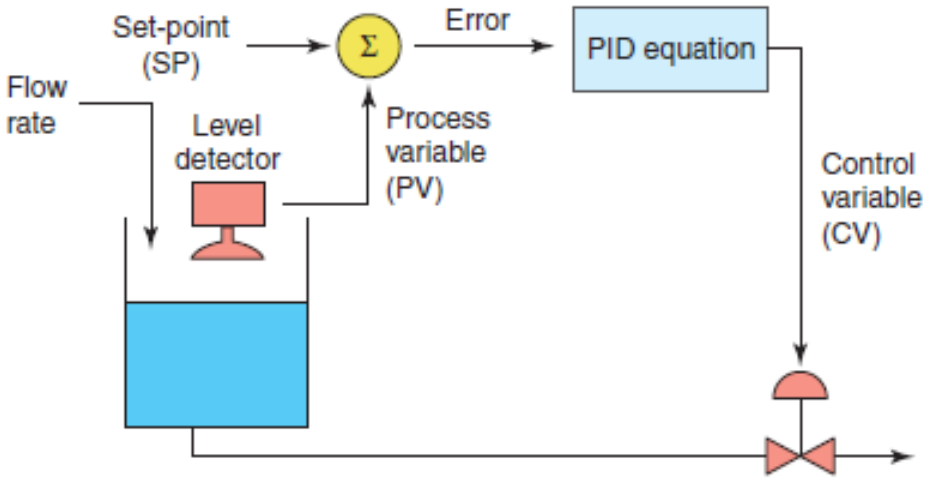
➢PID operations are more complex and are mathematically based.

➢ PID controllers produce outputs that depend on the *magnitude, duration, and rate of change of the system error* signal.

➢Sudden system disturbances are met with an aggressive attempt to correct the condition.

➢ A PID controller can reduce the system error to 0 faster than any other controller.





❑Closed-Loop Control

Control Block is the file that stores the data required to operate the instruction.

- Process Variable (PV) is an element address that stores the process input value.
- Control Variable (CV) is an element address that stores the output of the PID instruction.