Chapter 10

Data Manipulation Instructions

10.1

Data Manipulation

***Data manipulation* instructions allow numerical data stored in the controller's memory to be operated on within the control program.**

There are two classes of data manipulation instructions: instructions that operate on word data and those that operate on file data, which involve multiple words.

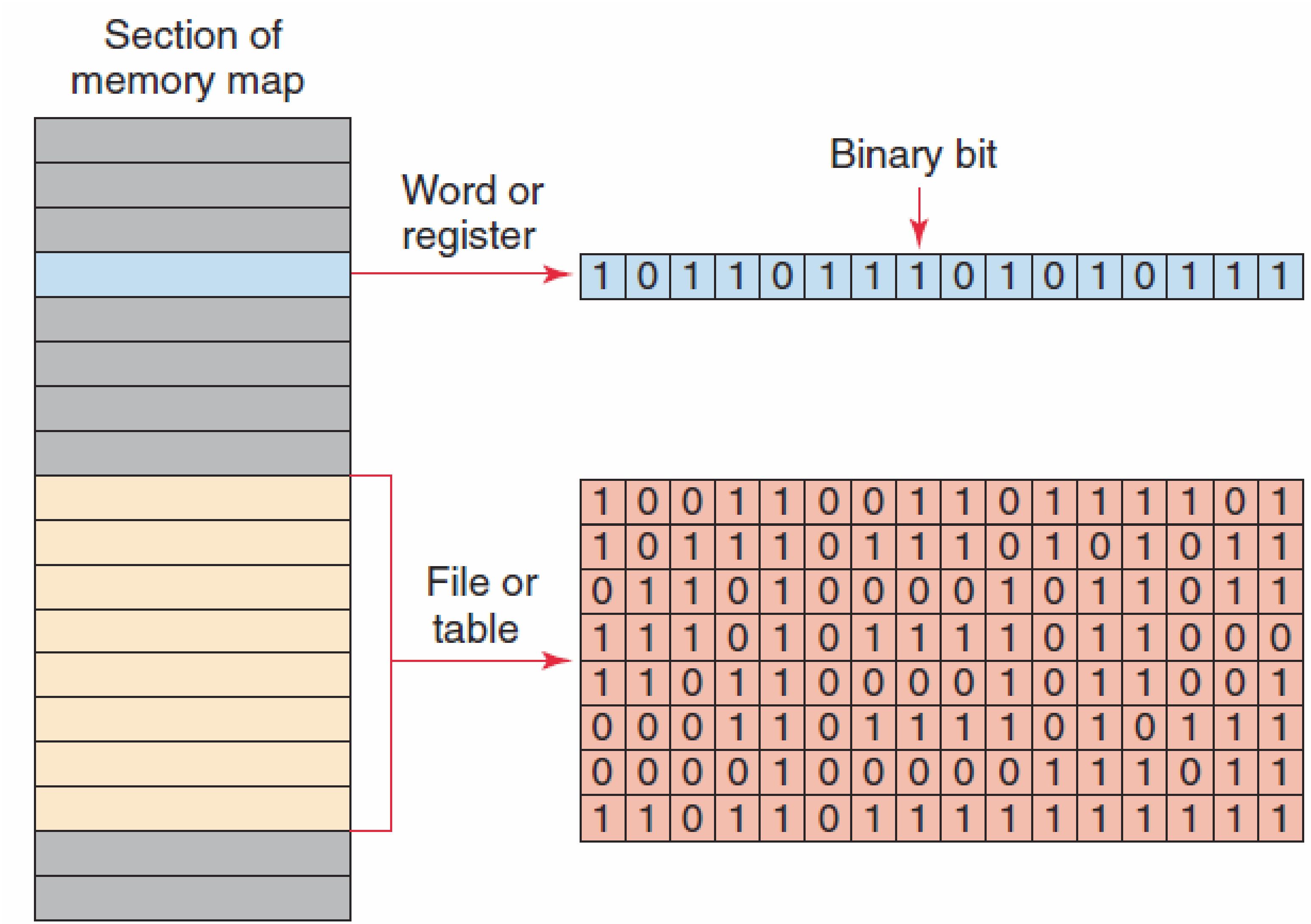
Word Data

1	0	1	1	0	1	1	1	0	1	0	1	0	1	1	1

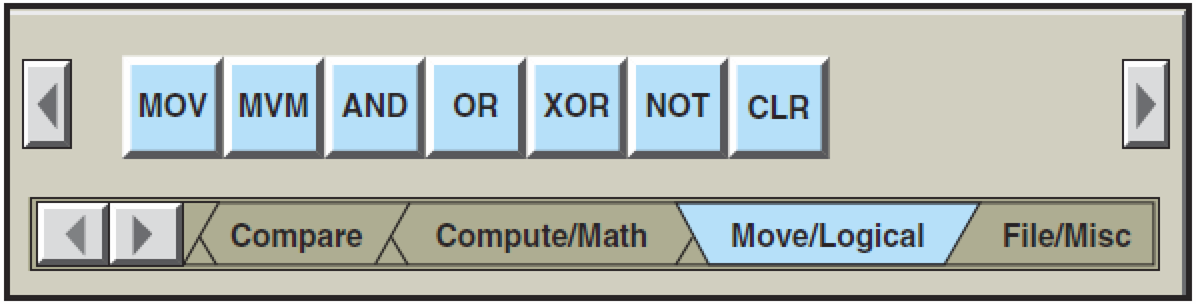
File Data

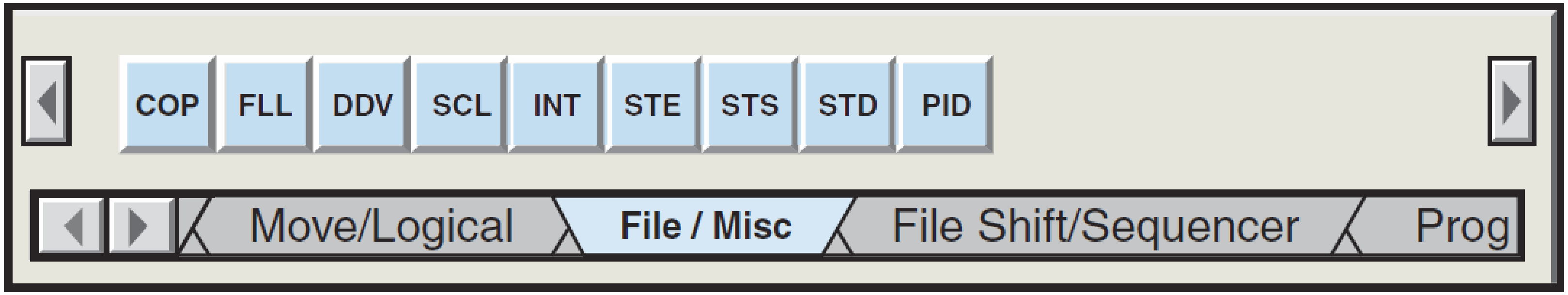
1	0	0	1	1	0	0	1	1	0	1	1	1	1	0	1
1	0	1	1	1	0	1	1	1	0	1	0	1	0	1	1
0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	1
1	1	1	0	1	0	1	1	1	1	0	1	1	0	0	0
1	1	0	1	1	0	0	0	0	1	0	1	1	0	0	1
0	0	0	1	1	0	1	1	1	1	0	1	0	1	1	1
0	0	0	0	1	0	0	0	0	0	1	1	1	0	1	1
1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1

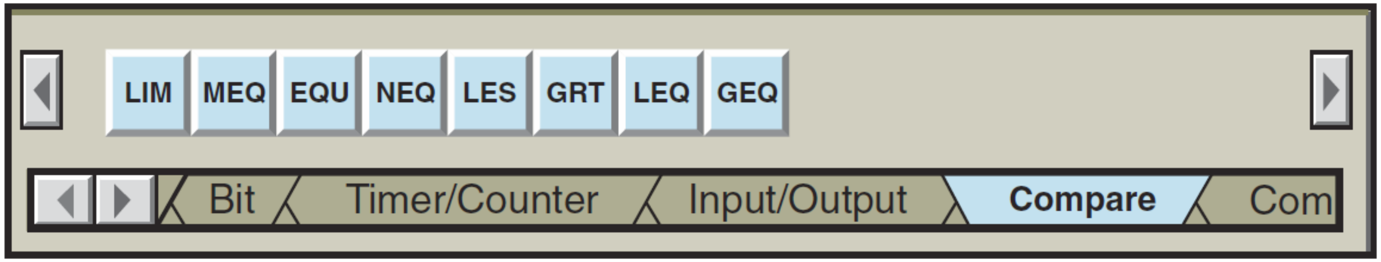
Data *files*, *words*, and *bits*.



The data manipulation instructions are *output* instructions which allow the *movement*, *manipulation*, or *storage* of data.



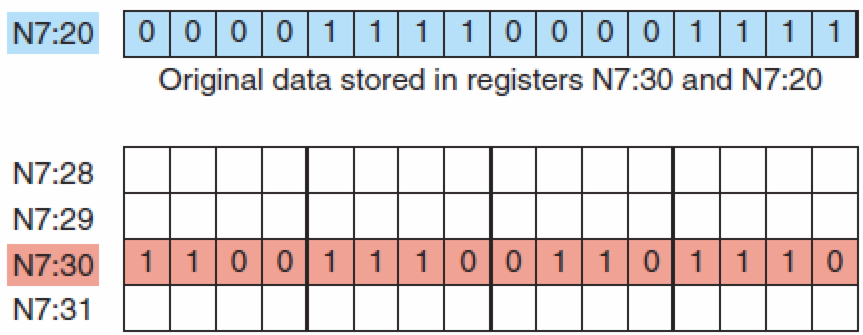




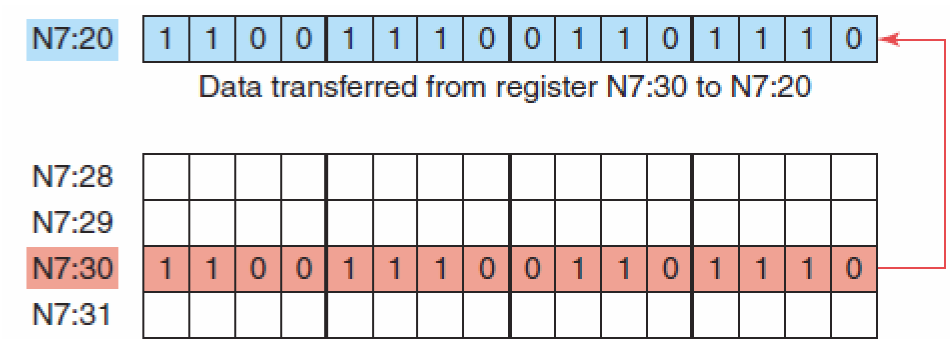
10.2

Data Transfer Operations

Data transfer **instructions involve the transfer of the contents from one word or register to another.**

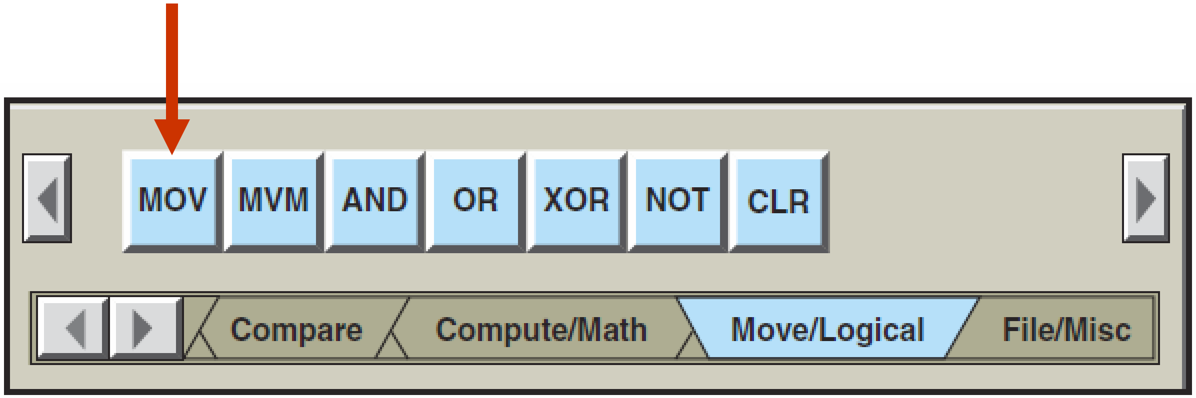


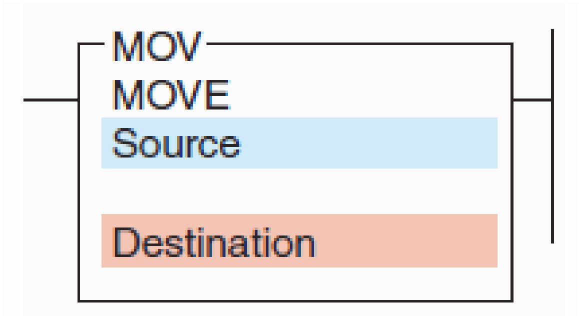
Original data are in register N7:30 and N7:20



After the data transfer has occurred register N7:20 now holds a duplicate of the information that is in register N7:30.

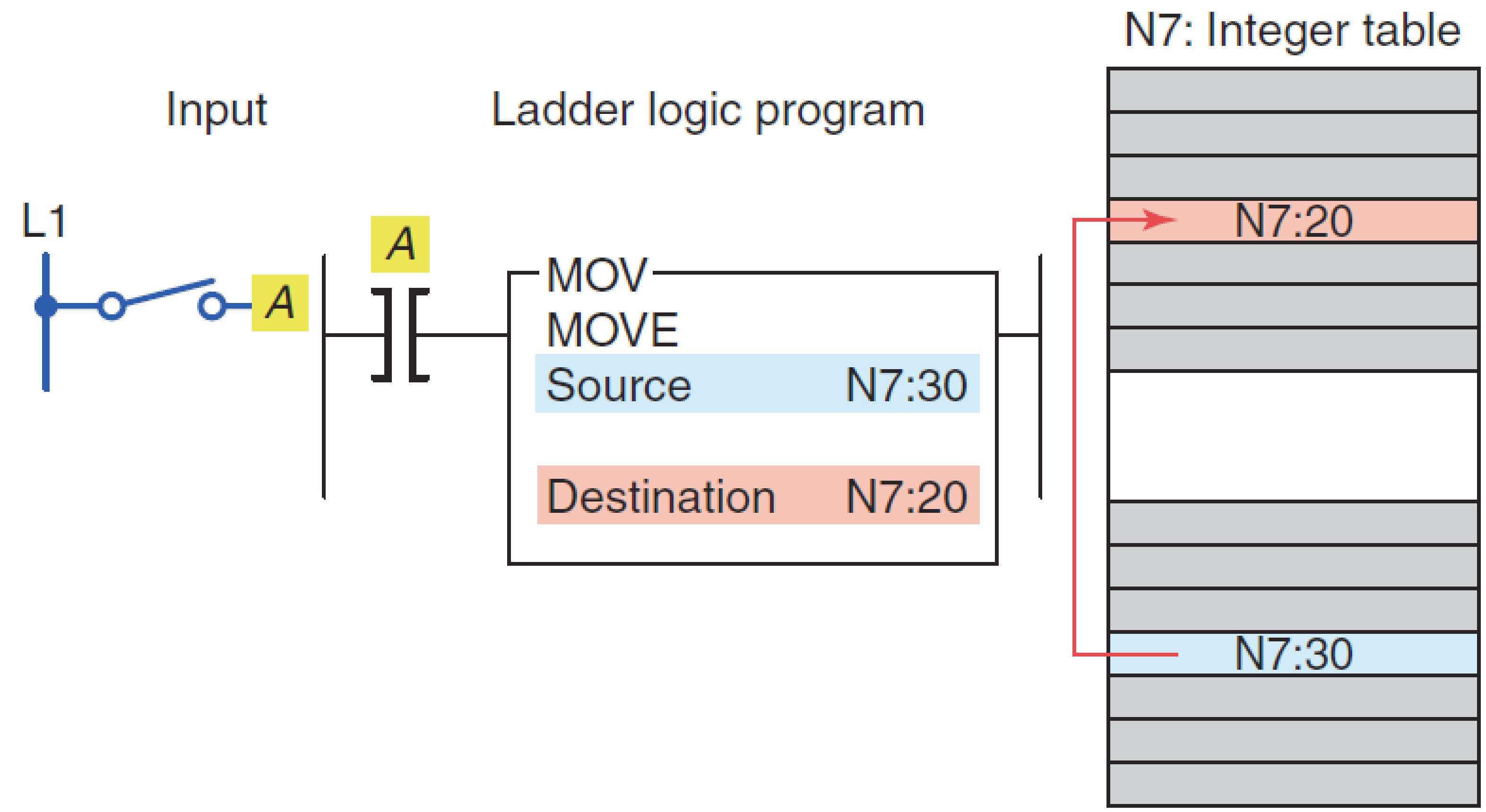
The *move MOV instruction* is used to copy the value in one register or word to another.



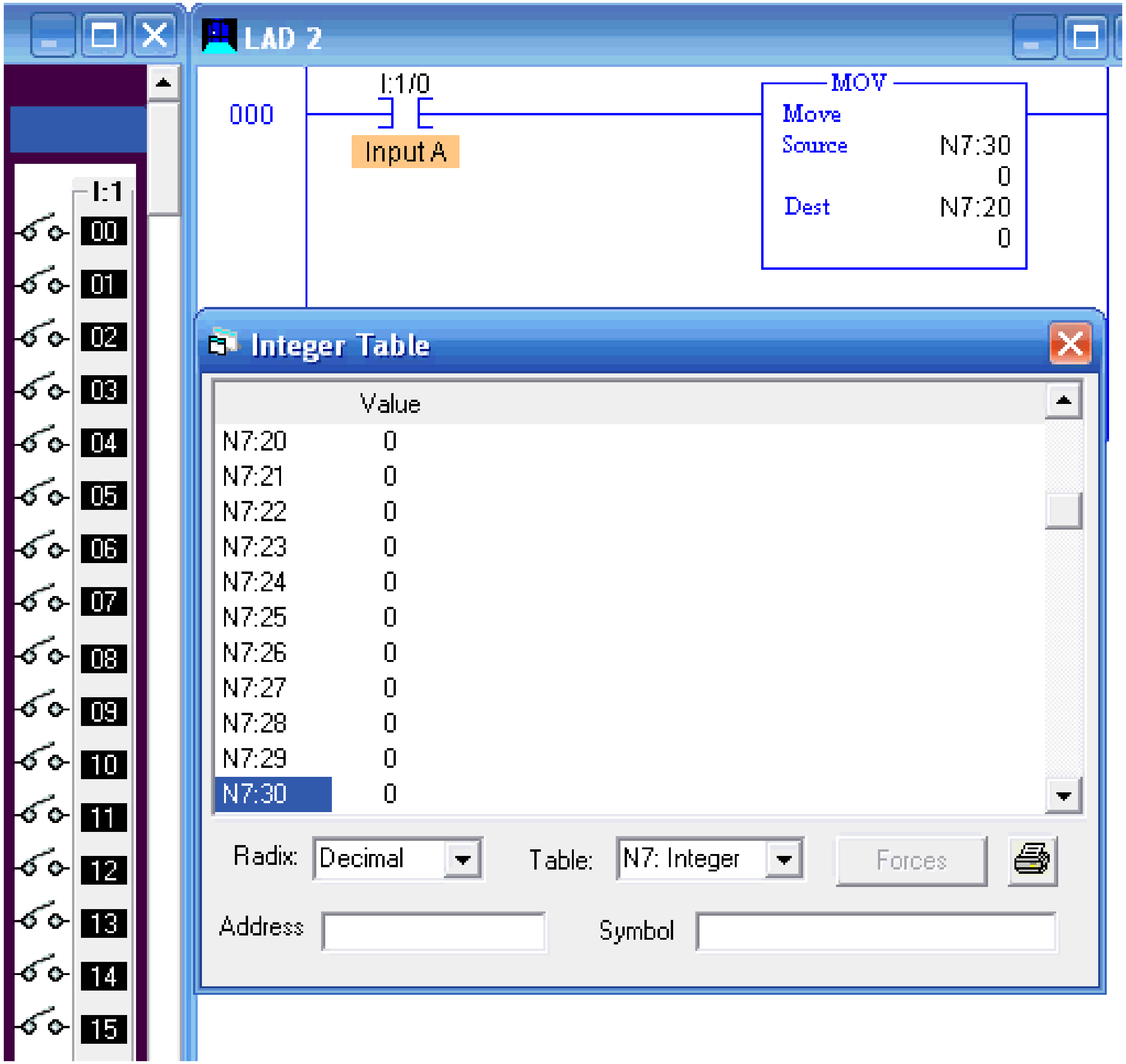


When the rung is true, this instruction copies data from a source register to a destination register.

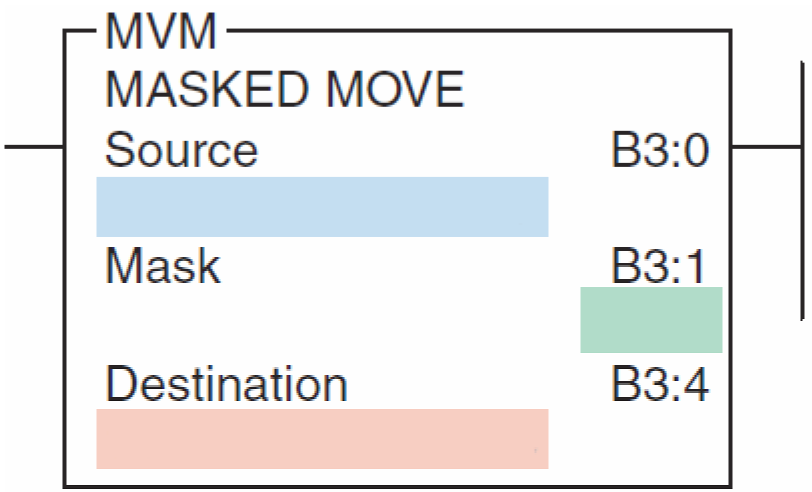
SLC 500 move instruction program.



Simulated move instruction program.

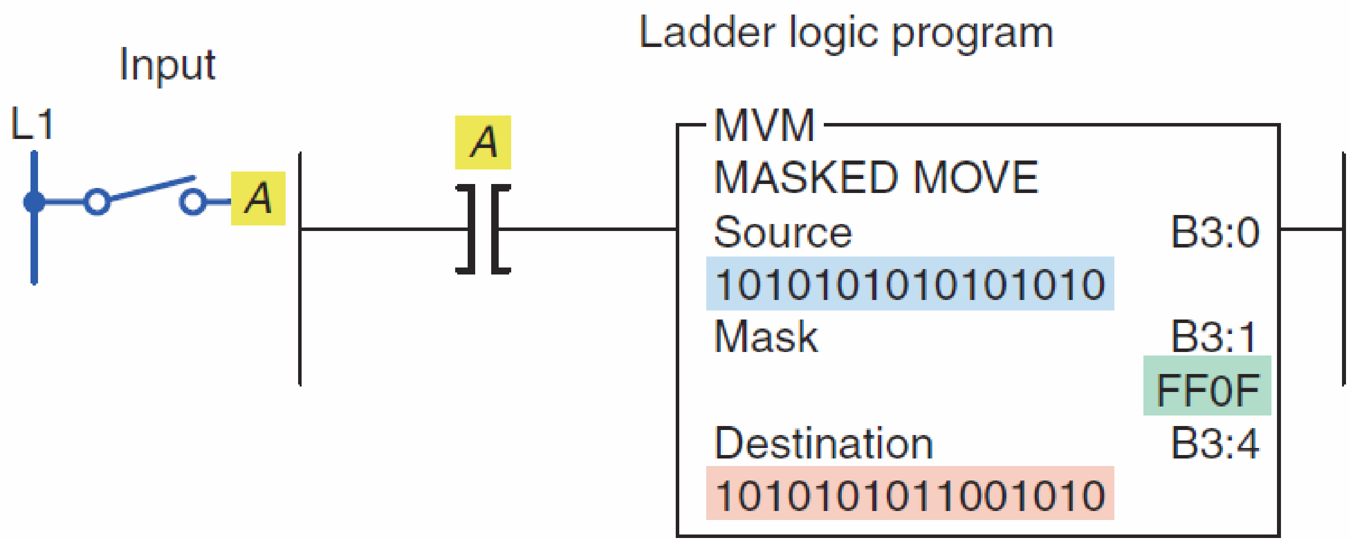


With the *move with mask (MVM)* instruction the data being moved must pass through a mask word to get to their destination address.



Masking refers to the action of hiding a portion of a binary word before transferring it to the destination address.

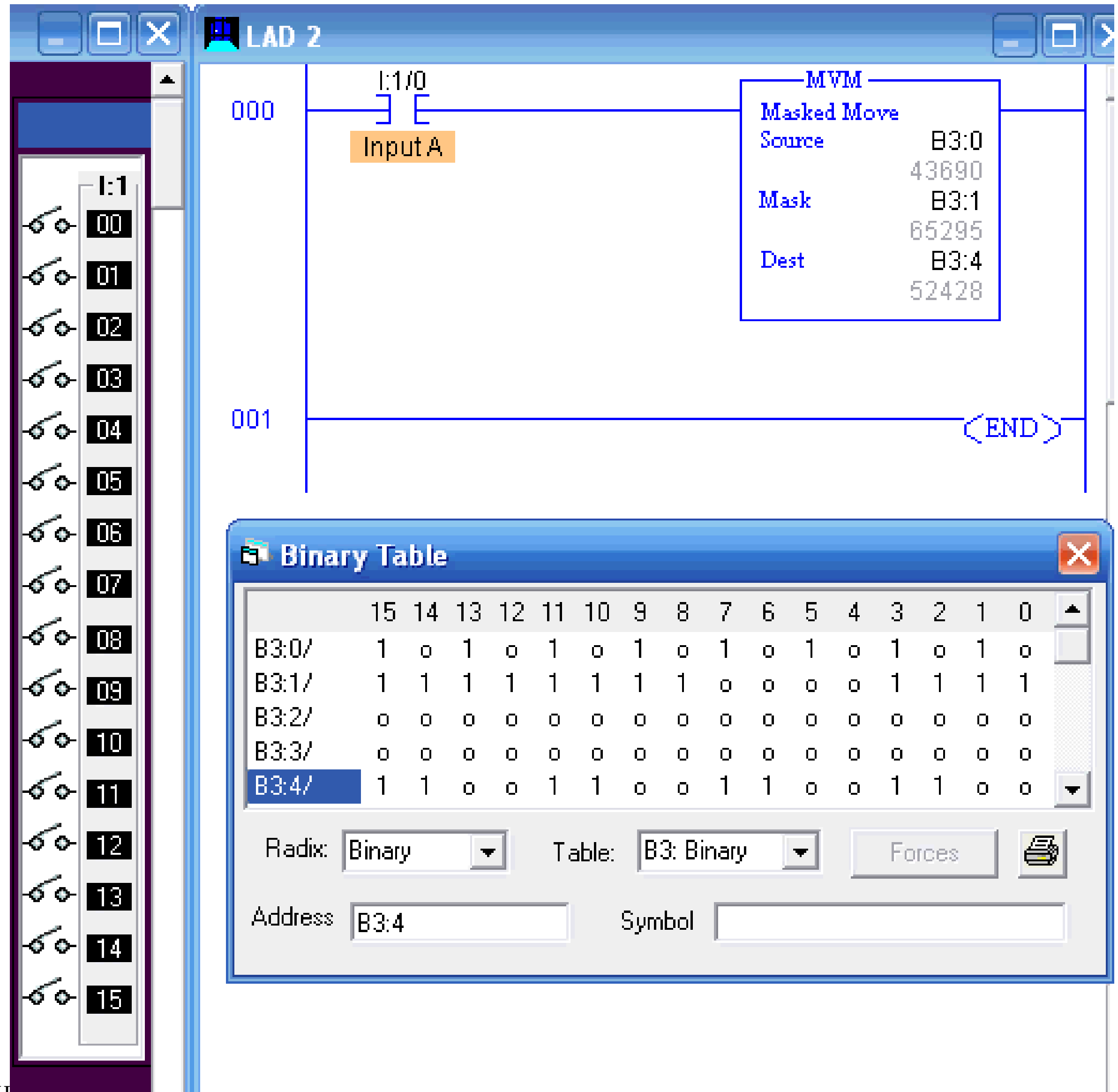
Move with mask (MVM) instruction program.



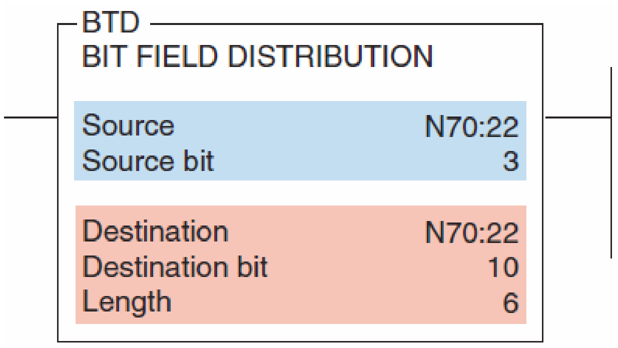
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	Source B3:0
1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1	Mask FF0F
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0	Destination B3:4 before instruction went true
1 0 1 0 1 0 1 0 1 1 0 0 1 0 1 0	Destination B3:4 after instruction went true

Unchanged

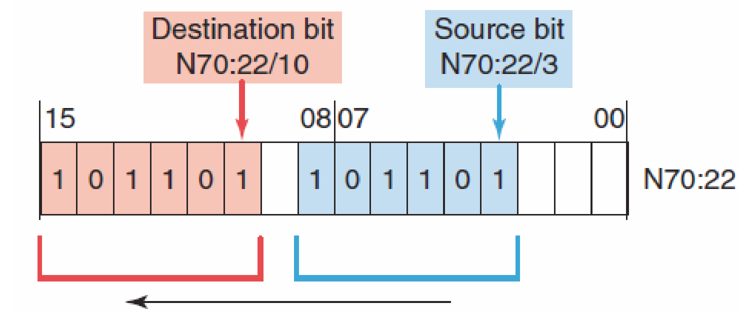
Simulated move with mask (MVM) program.

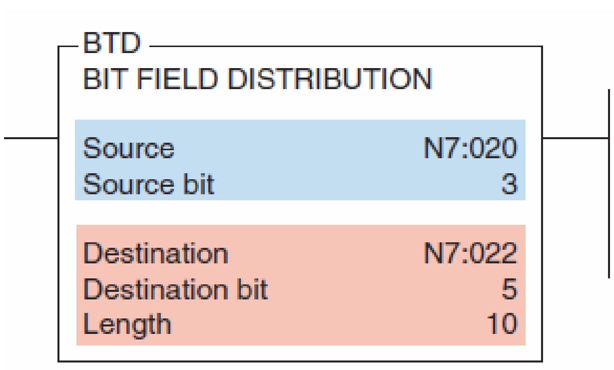


The *bit distribute (BTD)* instruction is used to move bits within a word or between words.

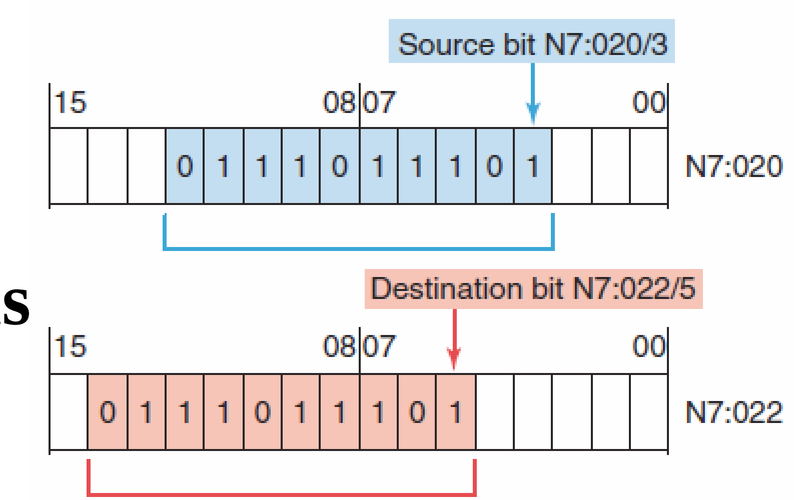


Moving bits within words

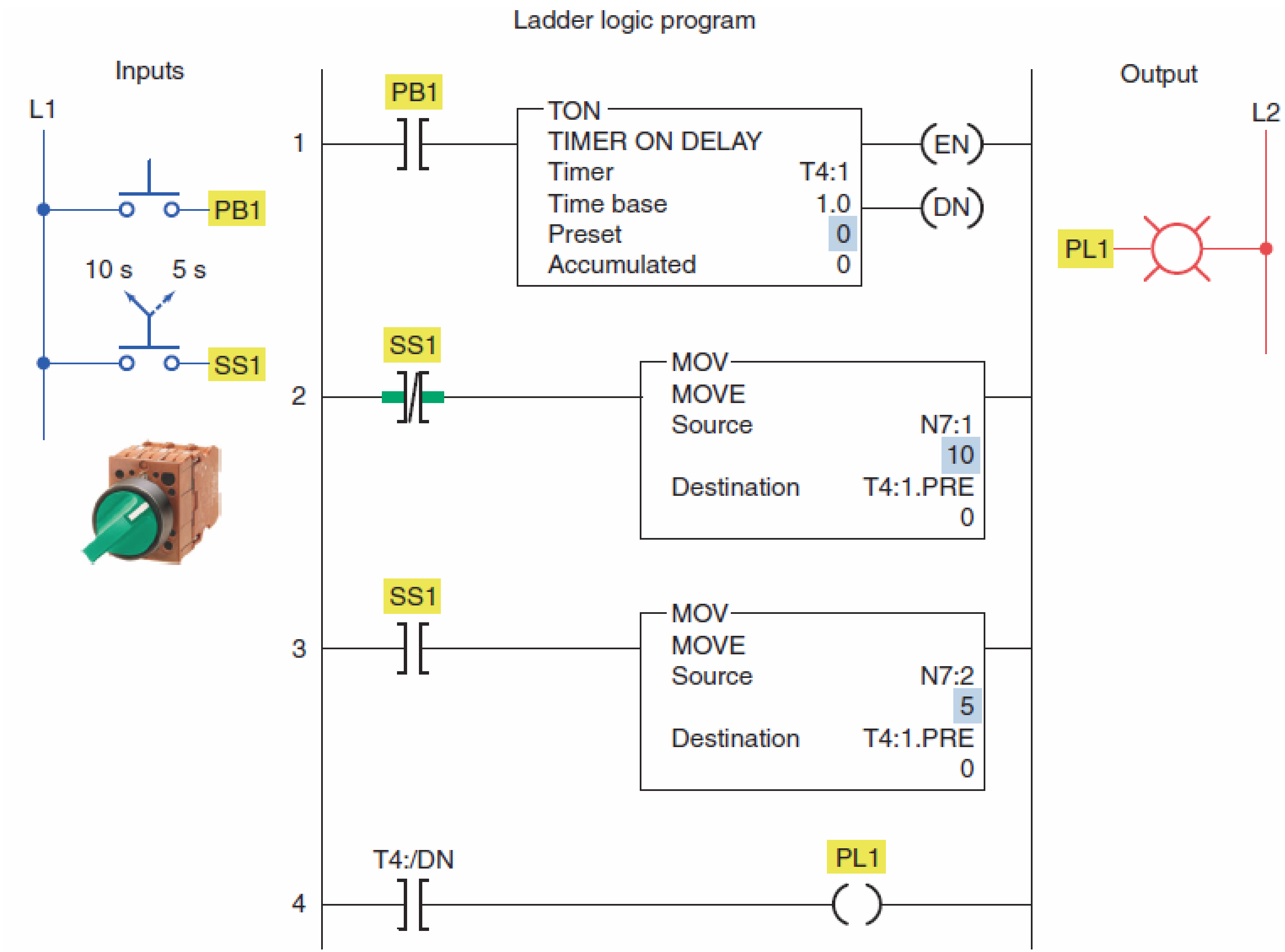




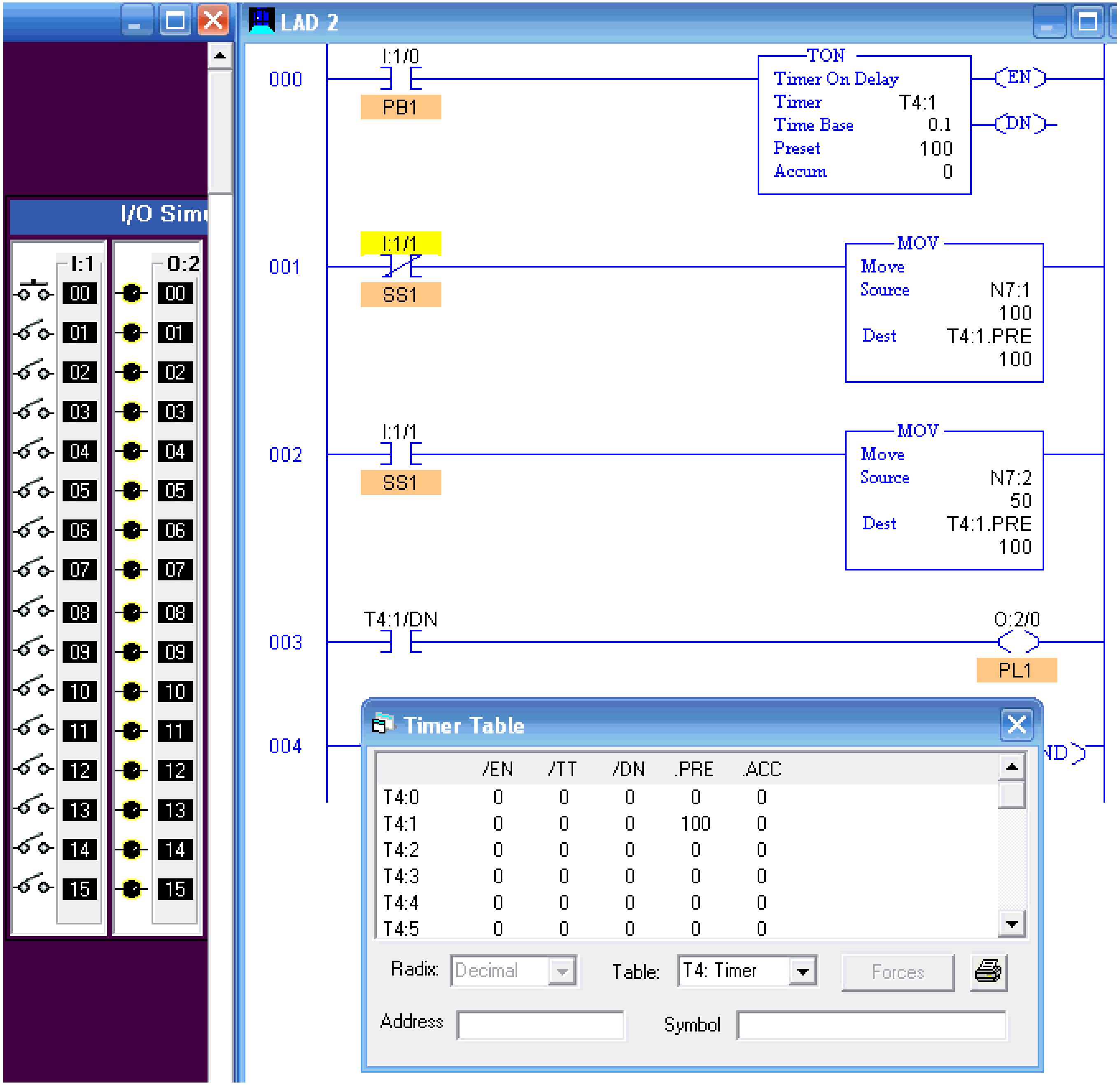
Moving bits between words



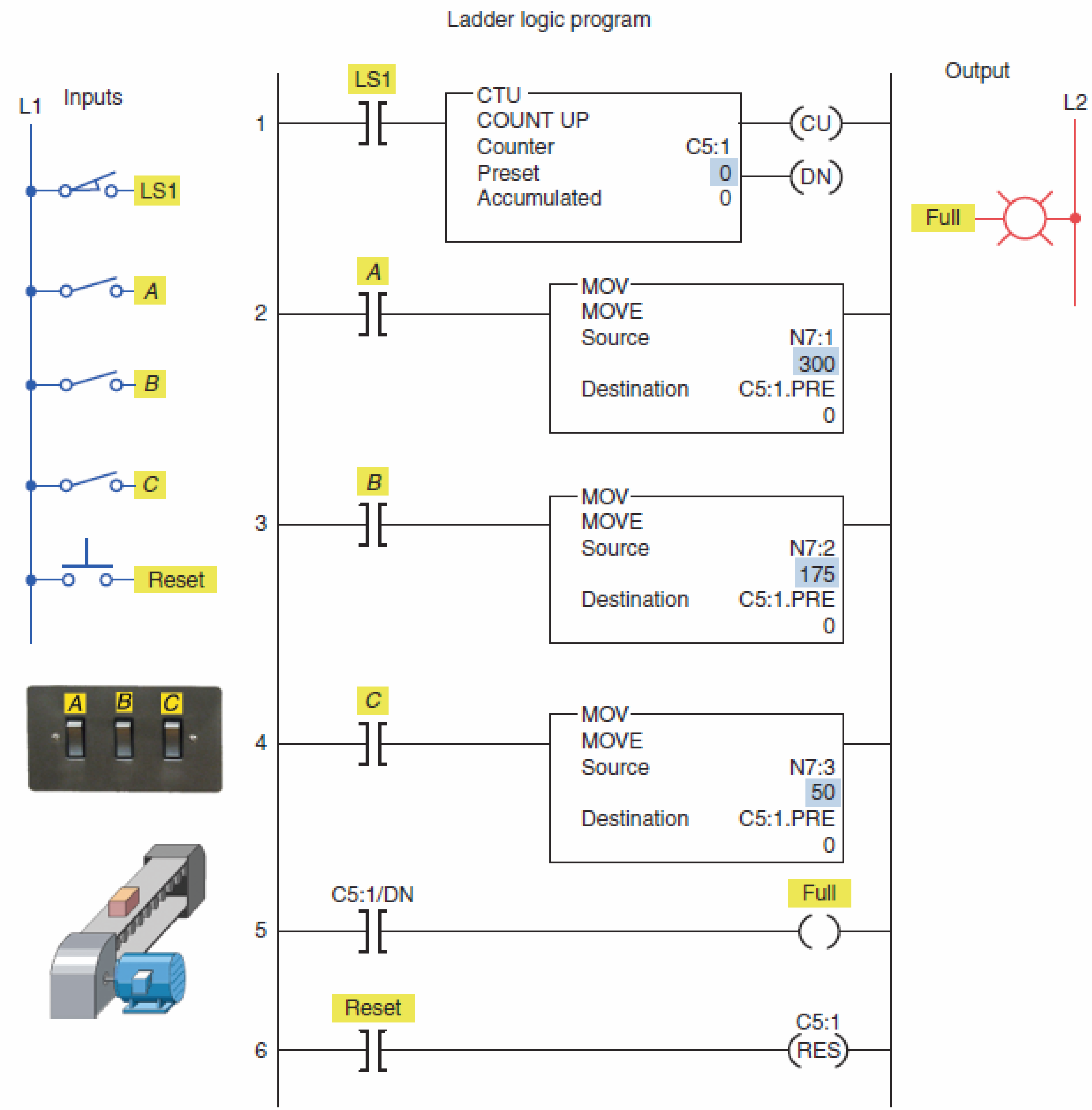
Move instruction used to change the preset time of a timer program.



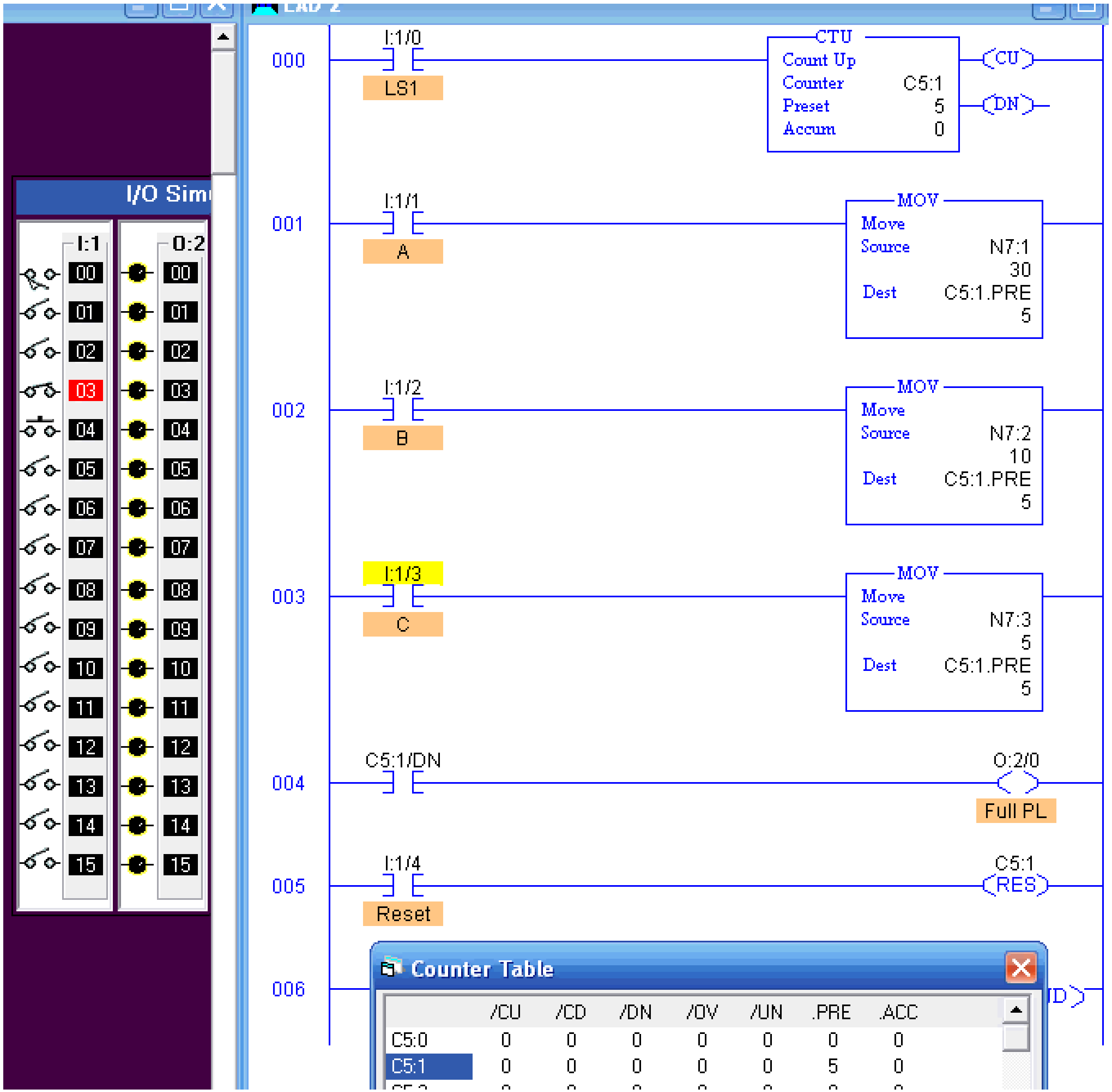
Simulated move instruction used to change the preset time of a timer program.



Move instruction used to change the preset count of a counter program.

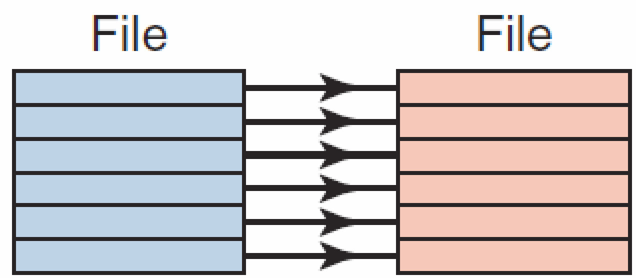


Simulated move instruction used to change the preset count of a counter program.

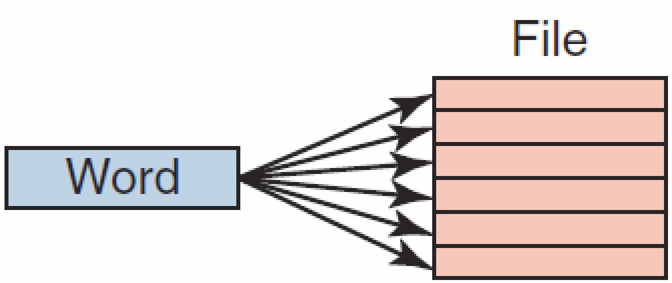


A *file* is a group of related consecutive words in the data table that have a defined *start* and *end* and are used to store information.

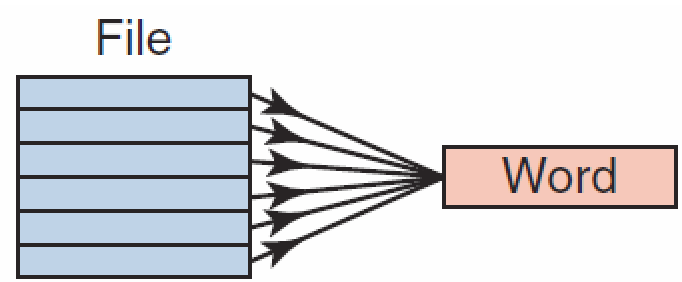
Moving data using file instructions



File-to-file move



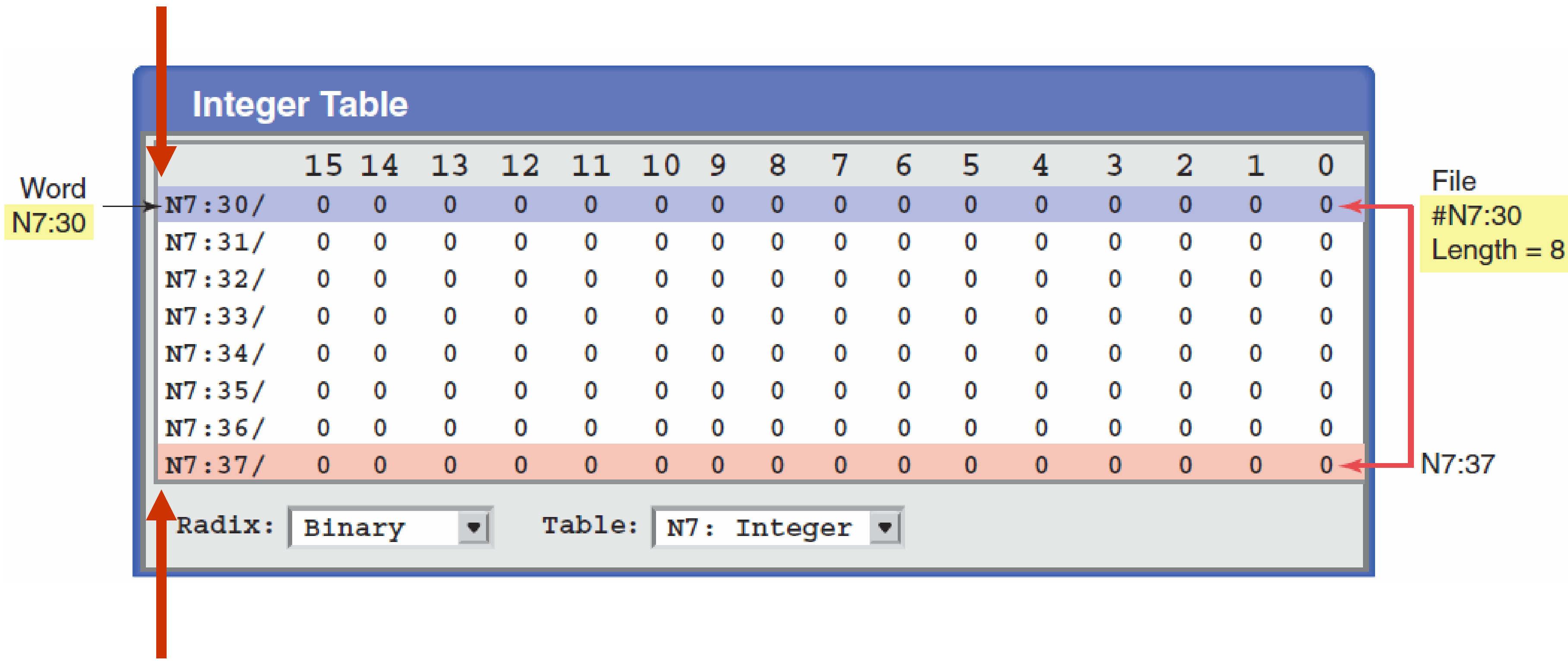
Word-to-file move



File-to-word move

SLC 500 word and file *addressing*.

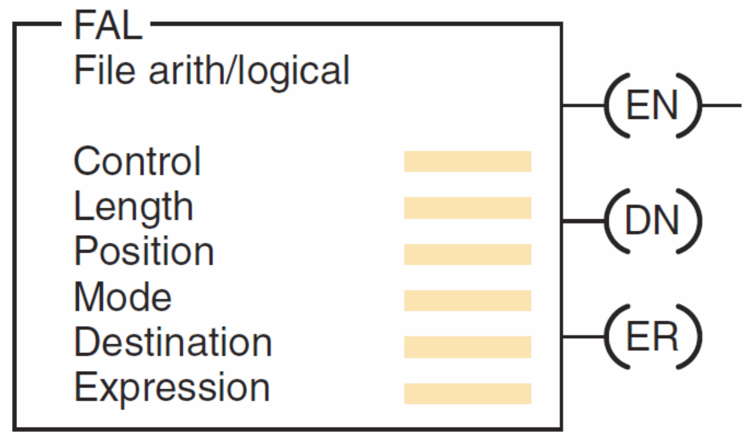
Address #N7:30 represents the starting address of a group of consecutive words in integer file 7.



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N7:30/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N7:31/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N7:32/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N7:33/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N7:34/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N7:35/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N7:36/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N7:37/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The length is eight words, which is determined by the instruction where the file address is used.

The *file arithmetic and logic (FAL)* instruction is used to copy data from one file to another and to do file math and file logic.

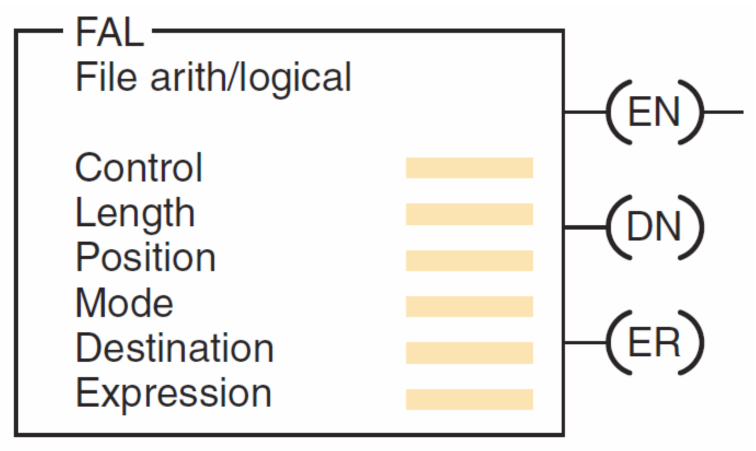


Control word uses four control bits: enable bit, done bit, error bit, and unload bit.

Length represents the file length in words.

Position points to the word being operated on.

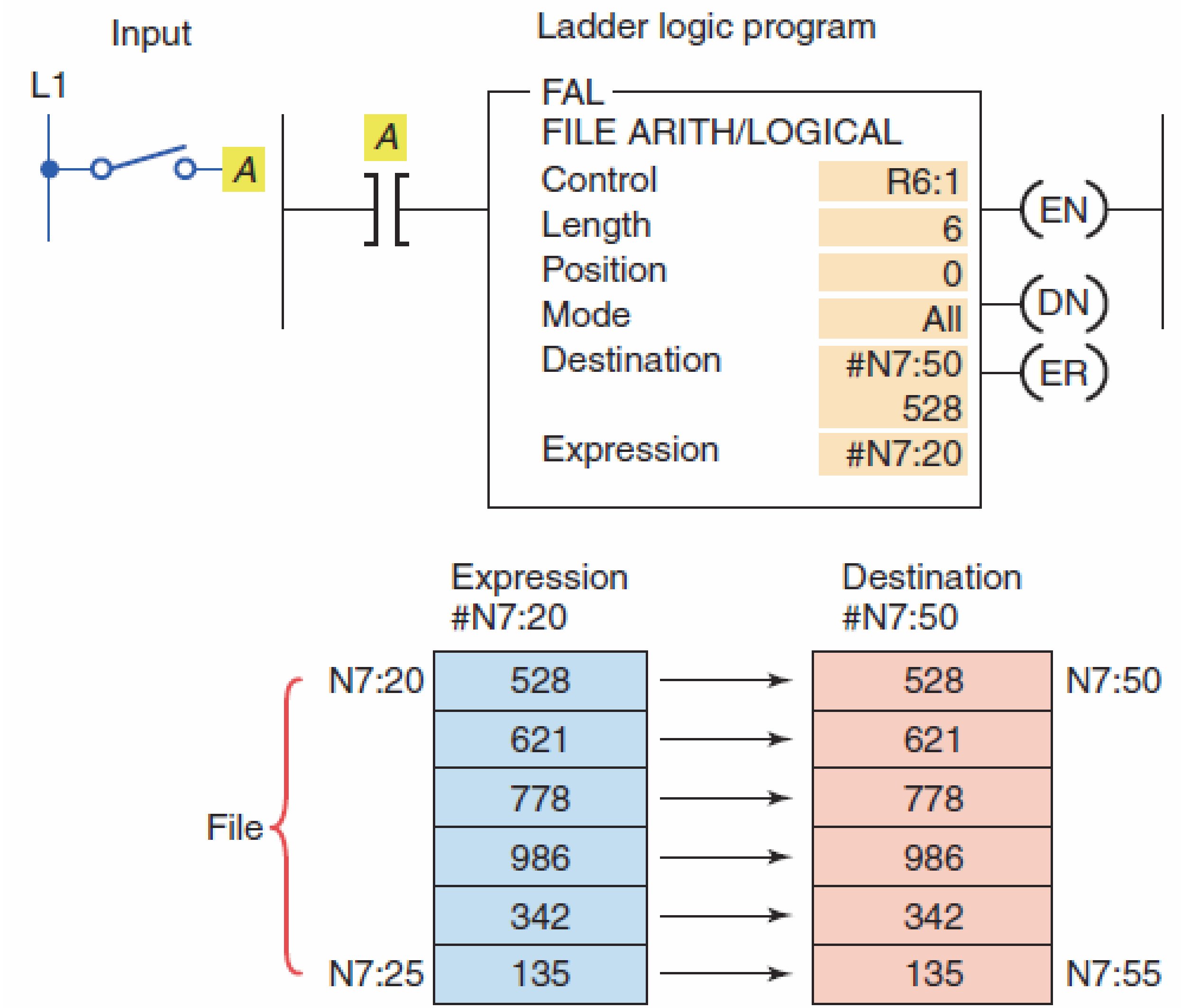
Mode (All, Numeric or Incremental) represents the number of file elements operated on per program scan.



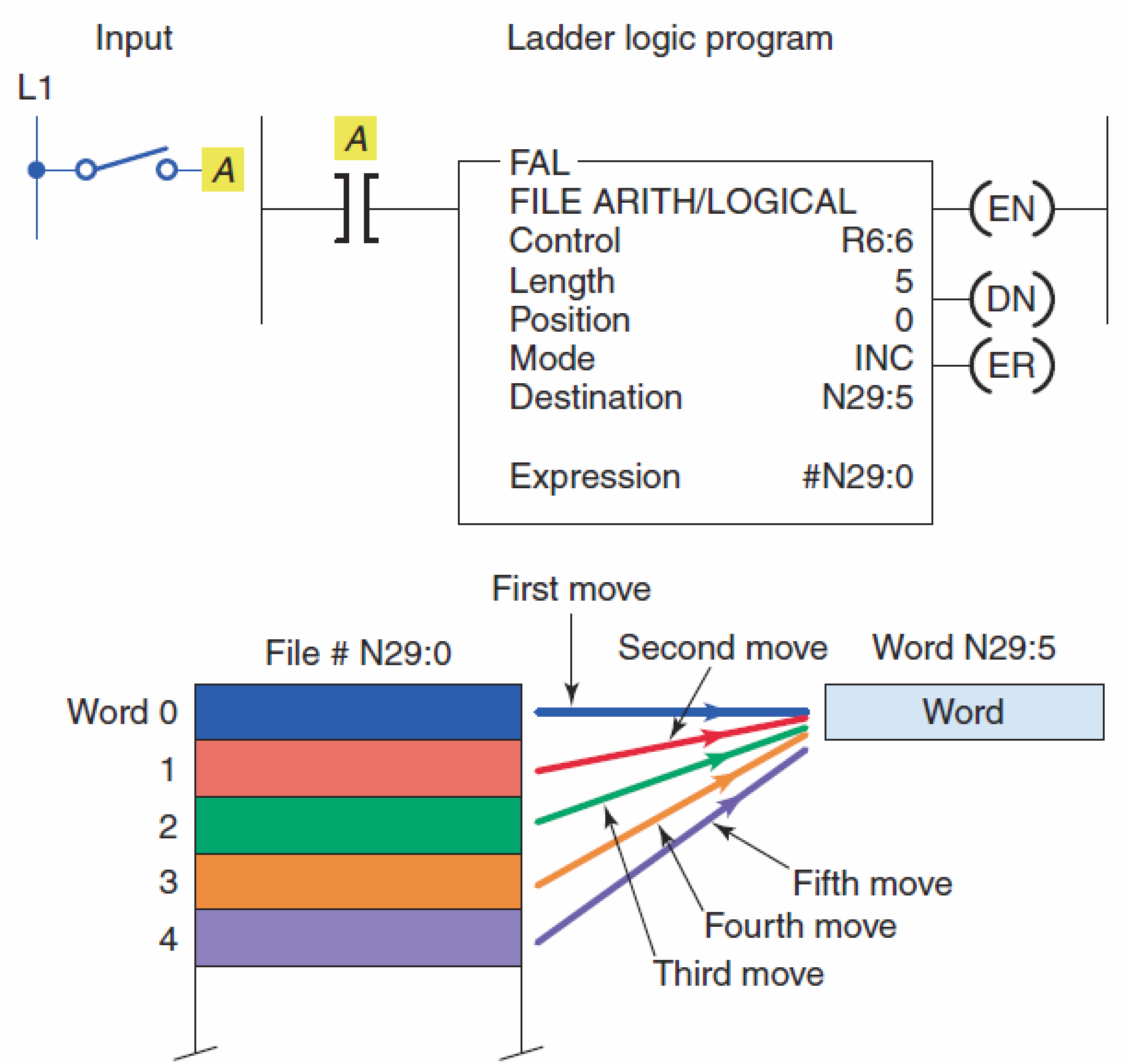
Destination is the address at which the processor stores the result of the operation.

Expression contains addresses, program constants, and operators that specify the source of data and the operations to be performed. The expression entered determines the function of the FAL instruction.

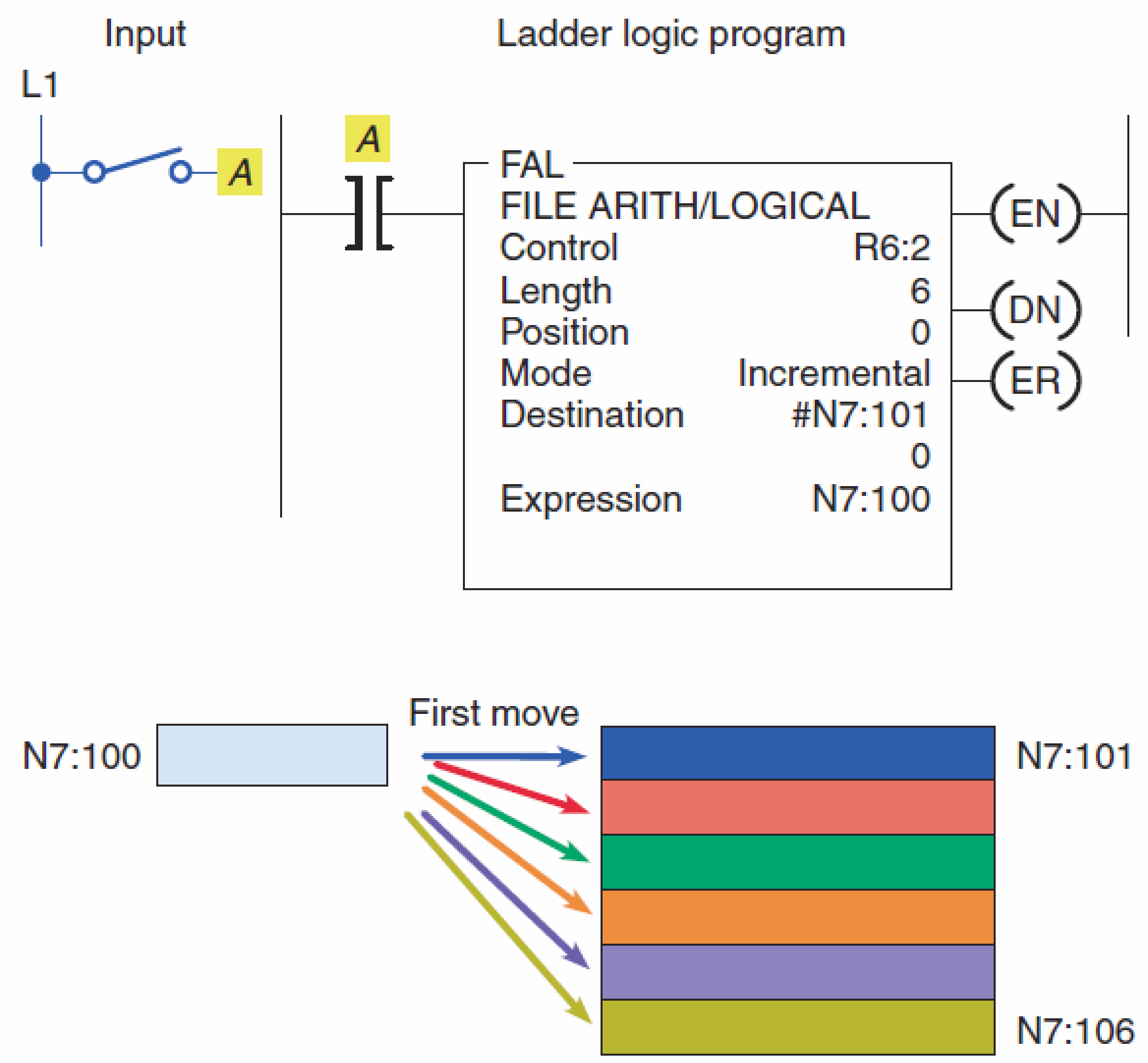
File-to-file copy function program using the FAL instruction.



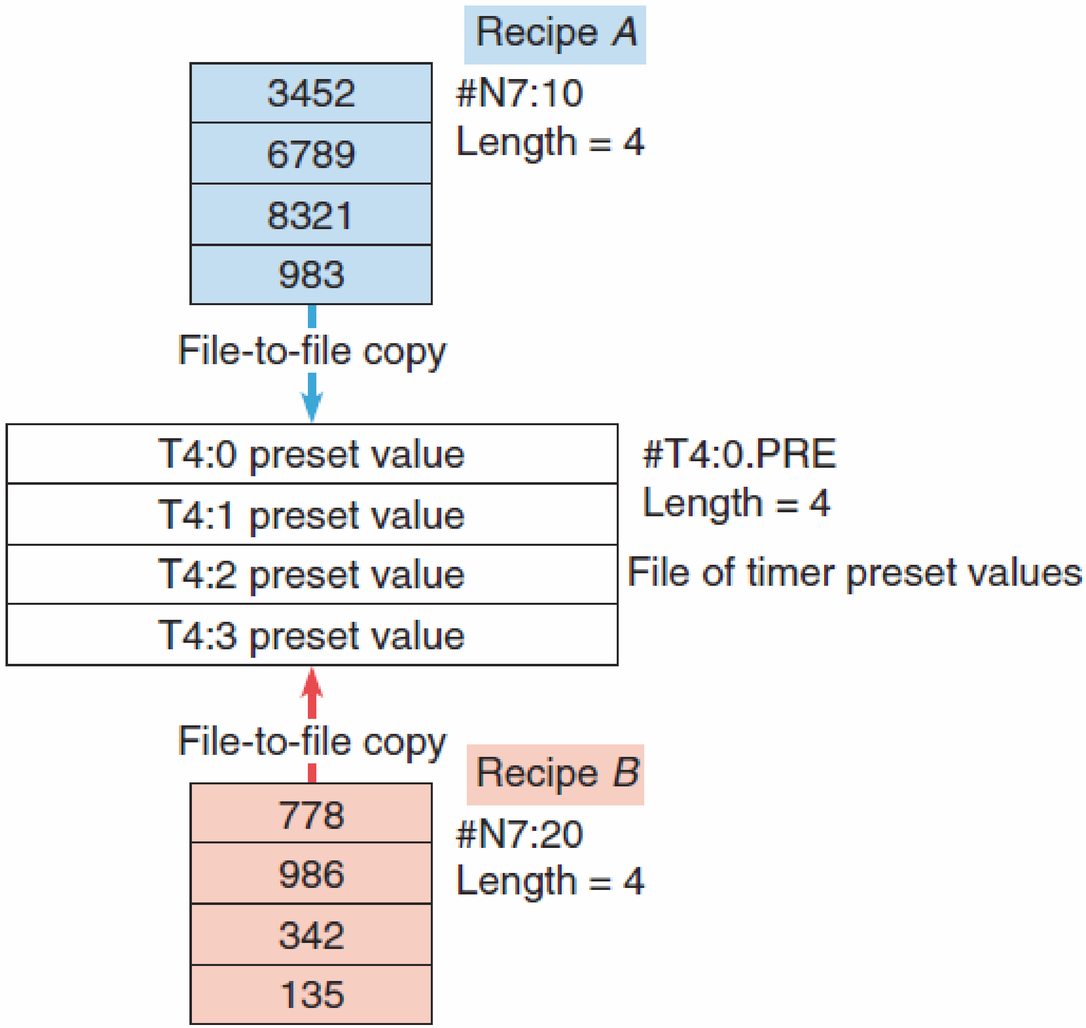
File-to-word copy function using the FAL instruction.



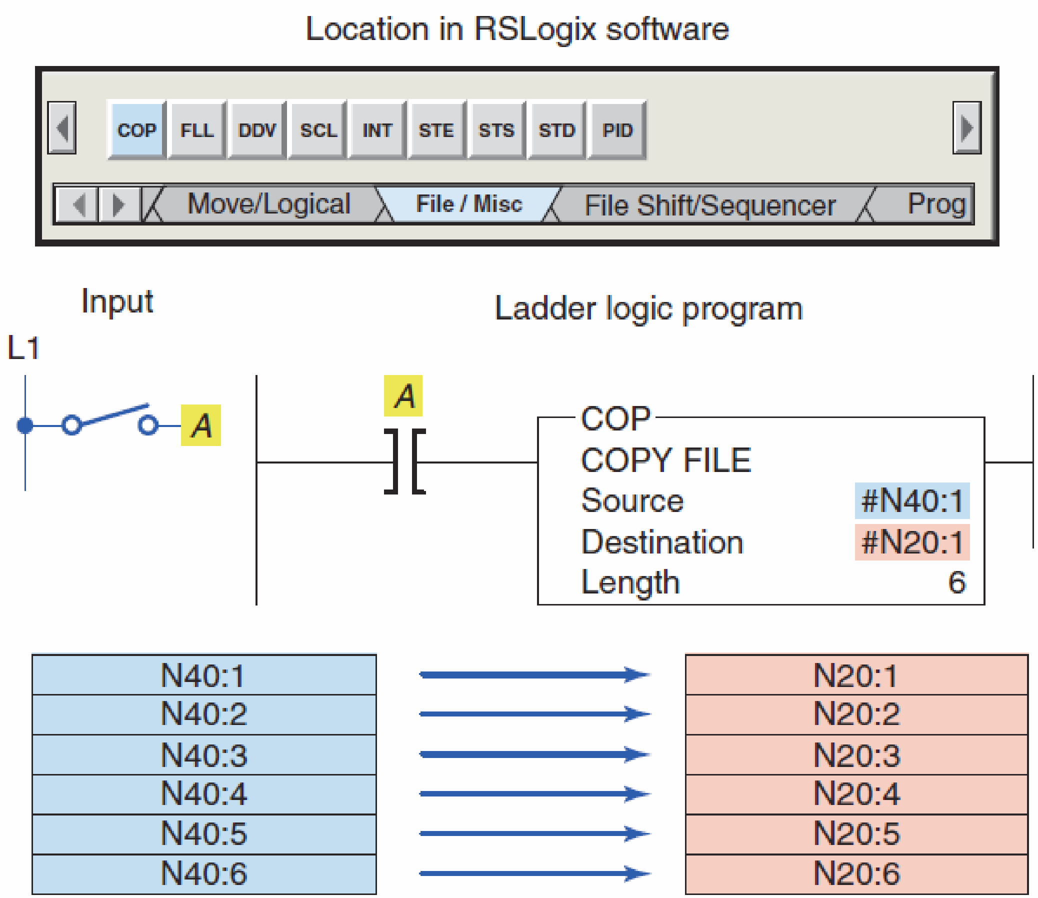
Word-to-file copy function using the FAL instruction.



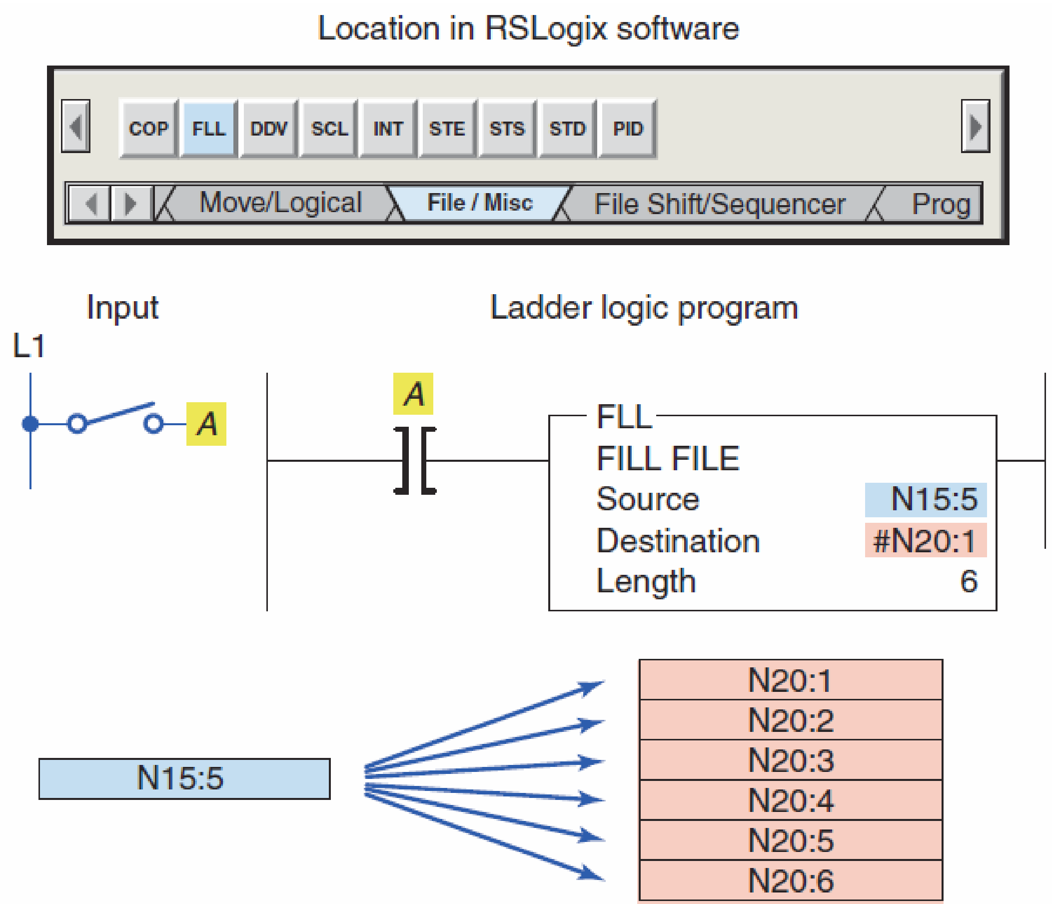
Copying recipes and storing values for timer presets.



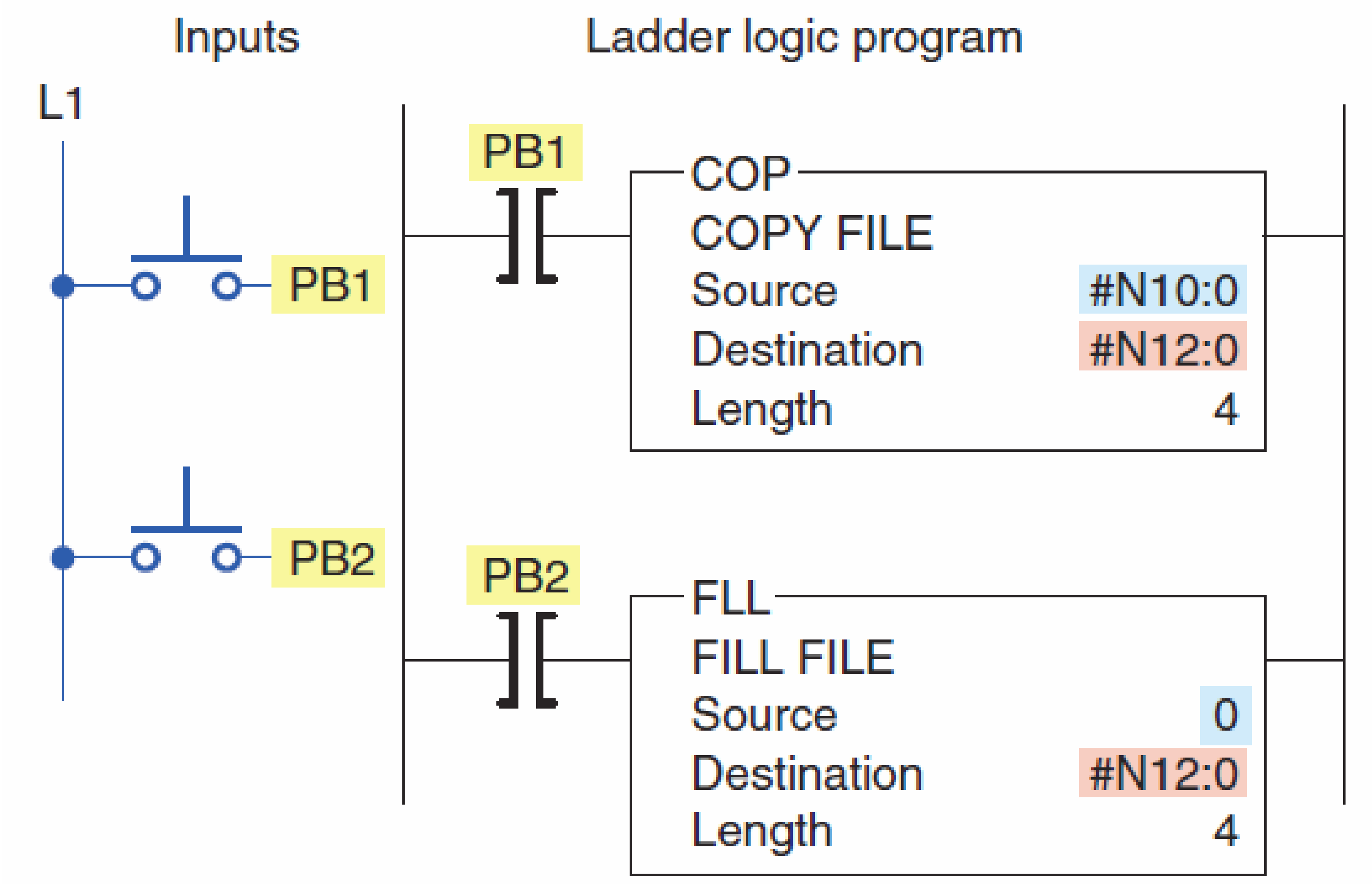
The *file copy (COP)* instruction and the *fill file (FLL)* instruction are high-speed instructions that operate more quickly than the same operation with the FAL instruction.



There is *no control element* to monitor and data conversion does not take place, so the source and destination should be the same file types.



Using the FLL instruction to change all the data in a file to *zero*.



10.3

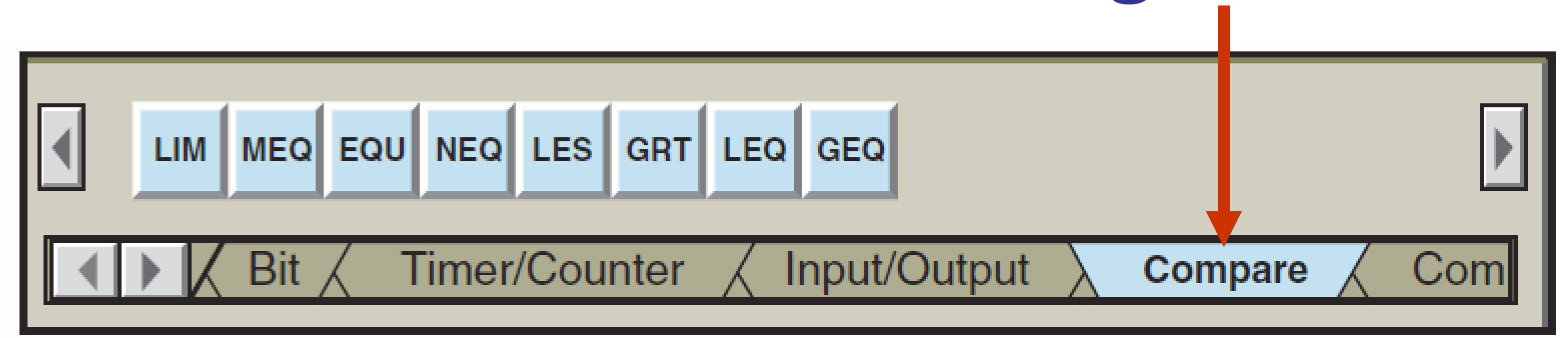
Data Compare Instructions

Data compare instructions are *input* instructions used to compare numerical values.

Name	Symbol
Equal to	(=)
Not equal to	(≠)
Less than	(<)
Greater than	(>)
Less than or equal to	(≤)
Greater than or equal to	(≥)

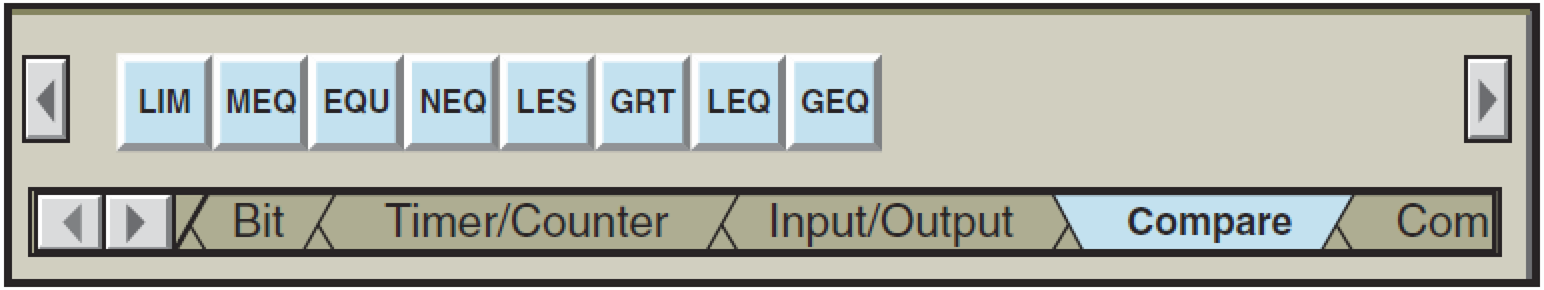
These instructions compare the data stored in two or more words (or registers) and make decisions based on the program instructions.

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



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.

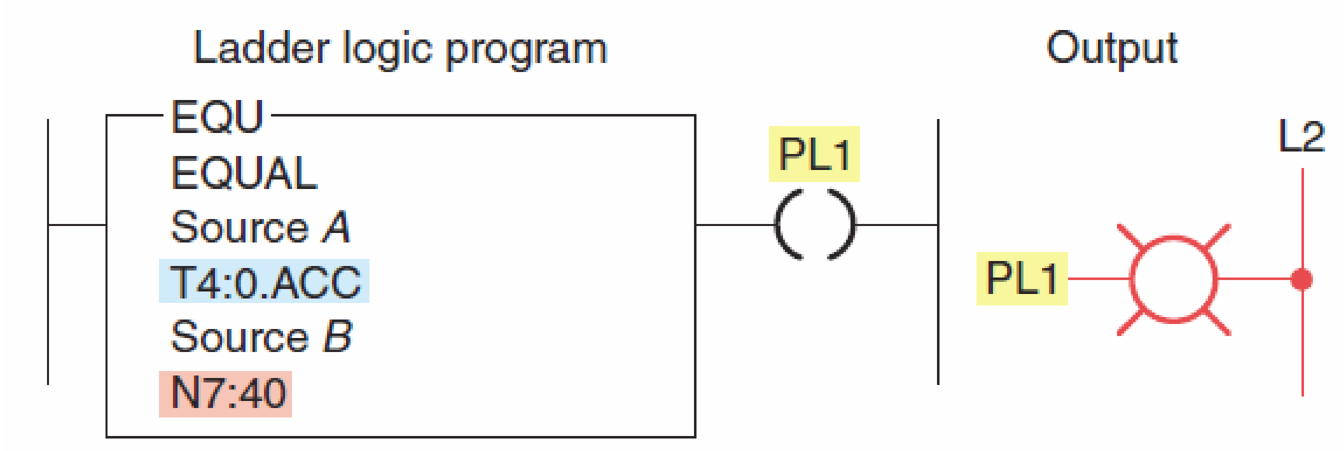
LES (Less Than) —Tests whether one value is less than a second value.

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.

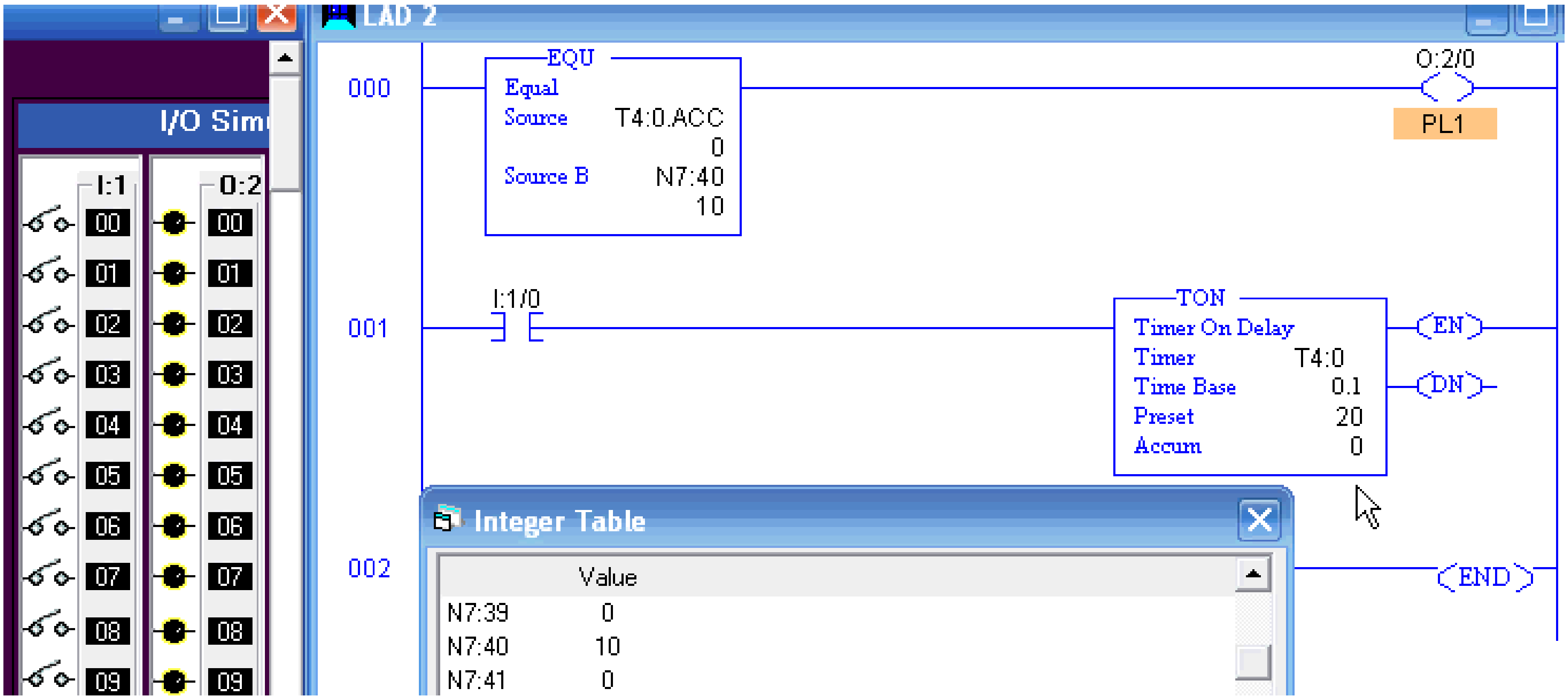
The *equal (EQU)* instruction rung compares the value of *source A* to that of *source B*.



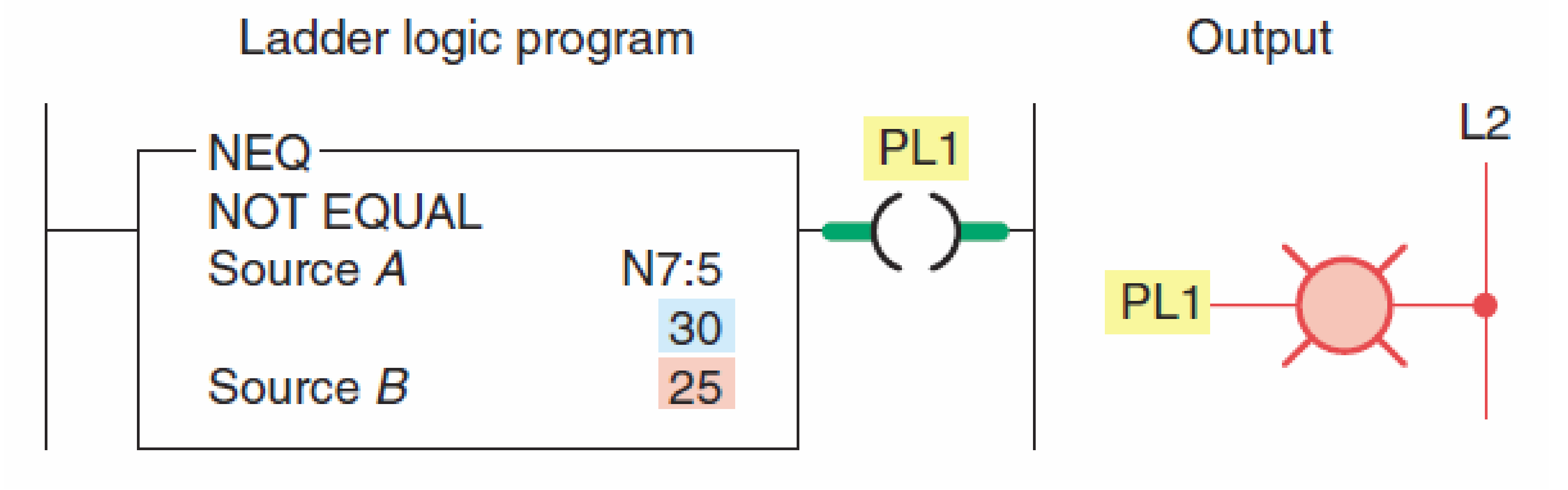
When source *A* is equal to source *B*, the instruction is logically true and PL1 is switched ON.

When source *A* is not equal to source *B*, the instruction is logically false and PL1 is switched OFF.

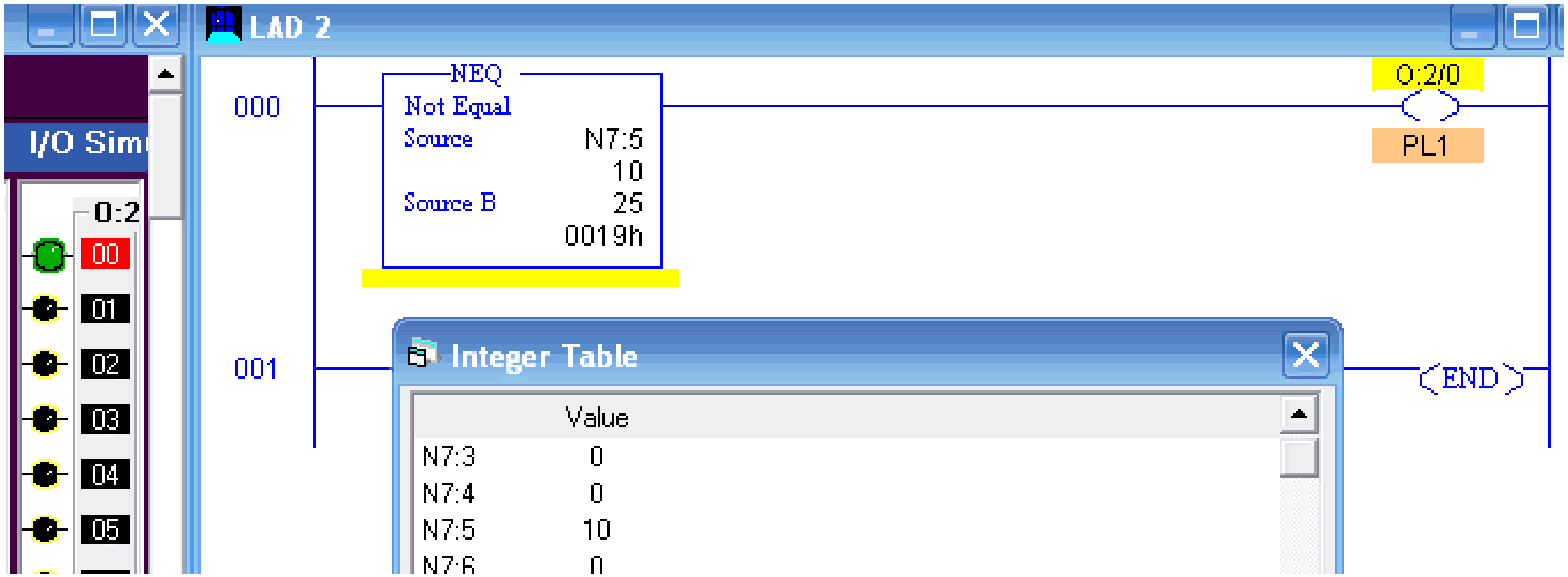
Simulated equal (EQU) instruction program rung.



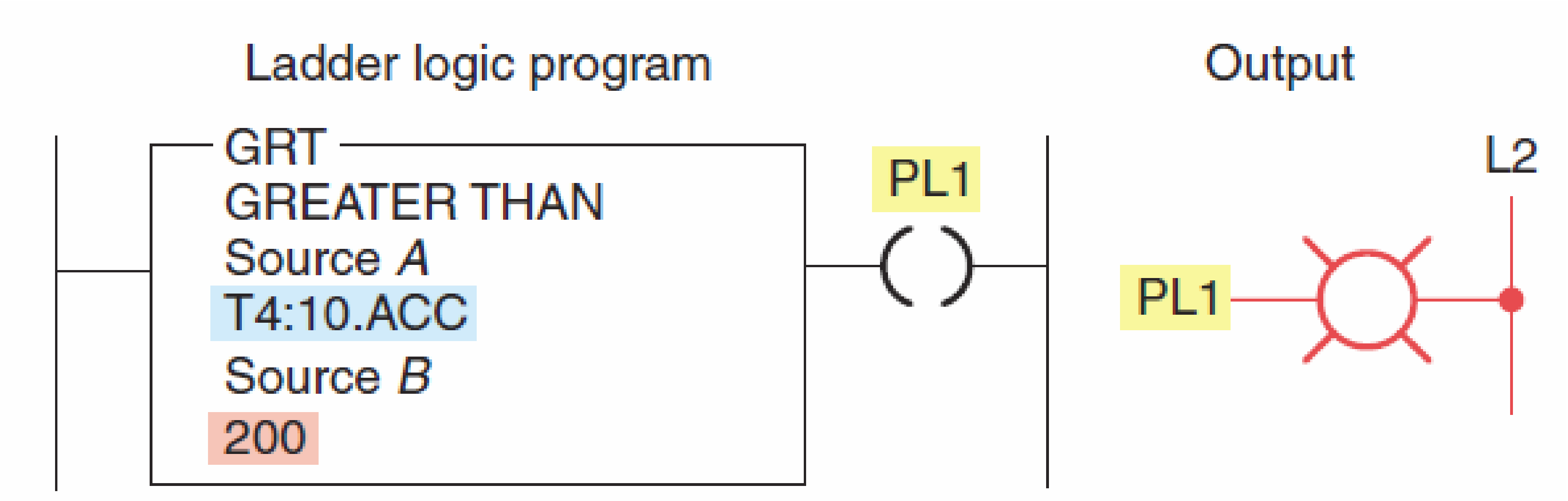
The *not equal (NEQ)* instruction is logically true when source *A is not equal* to source *B*, otherwise it is logically false.



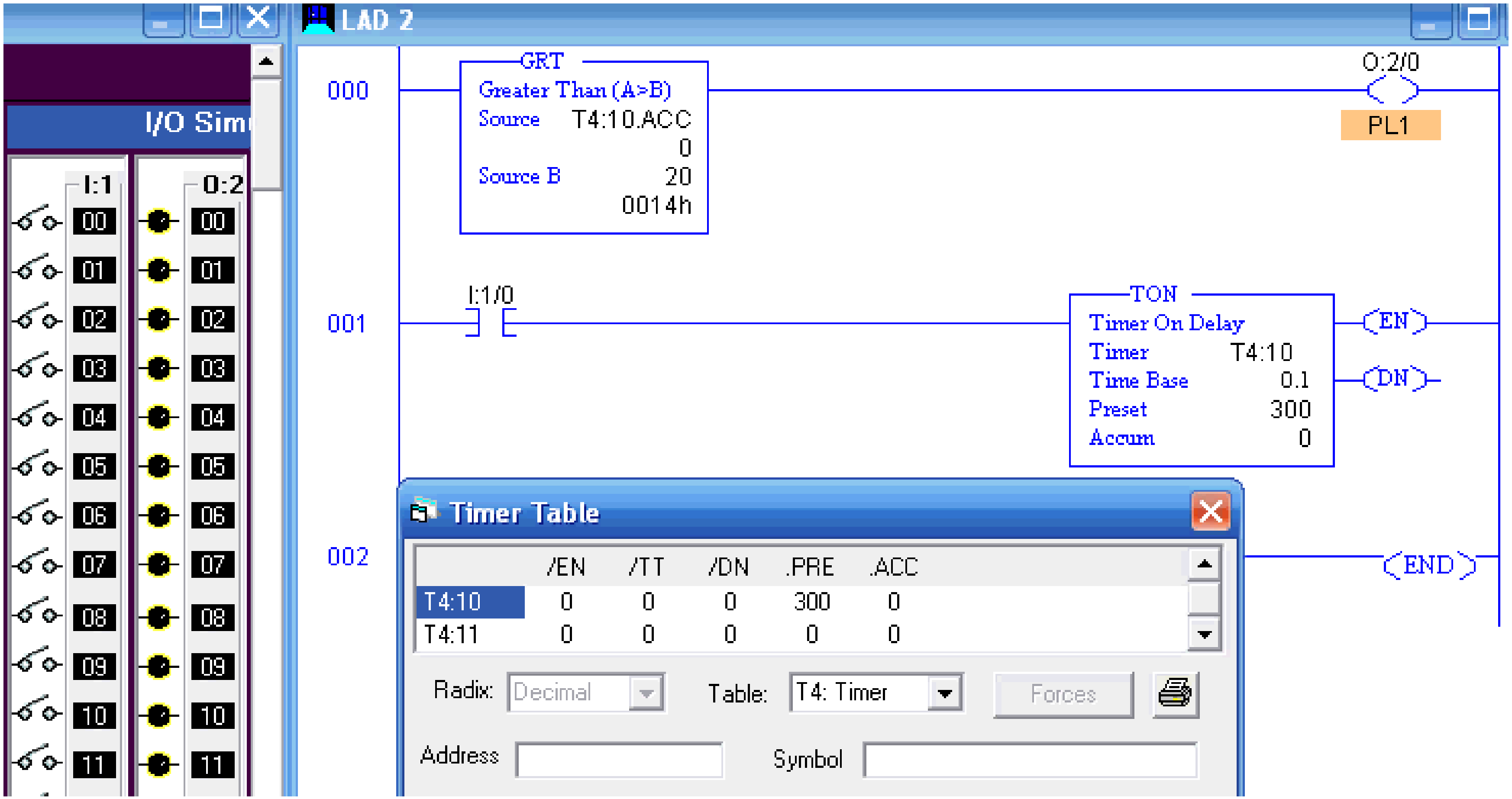
Simulated not equal (NEQ) instruction program rung.



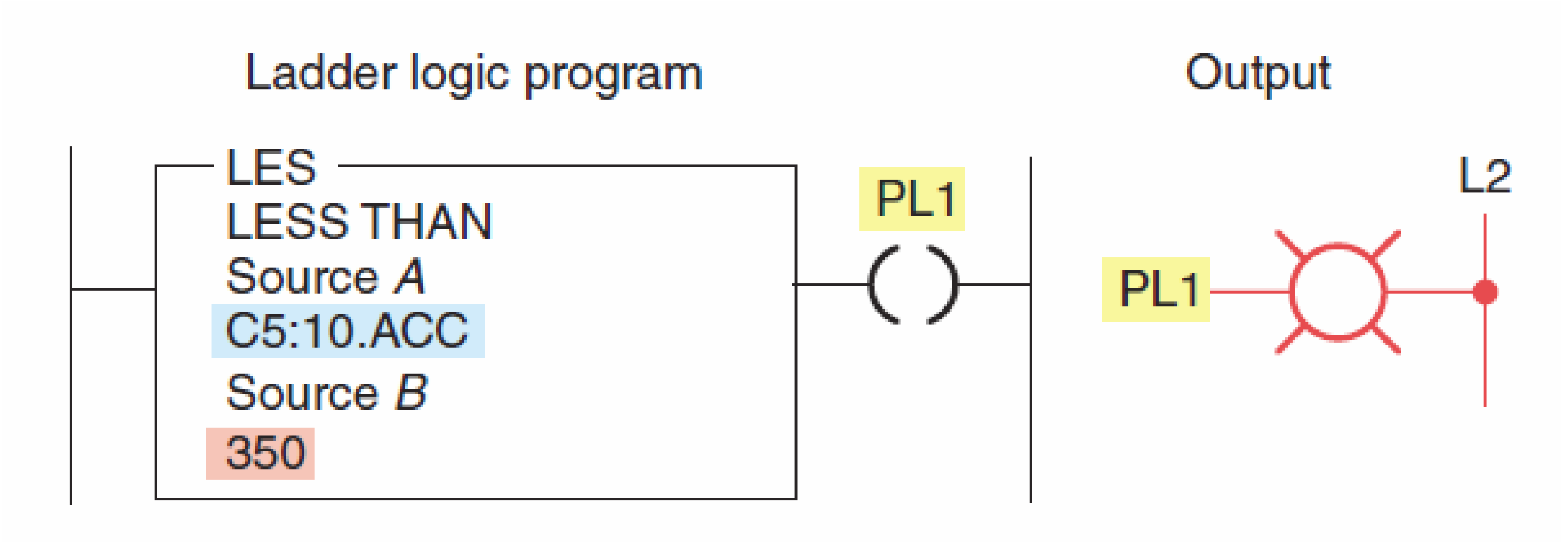
The *greater than (GRT)* instruction is logically true when source *A* is greater than source *B*, otherwise it is logically false.



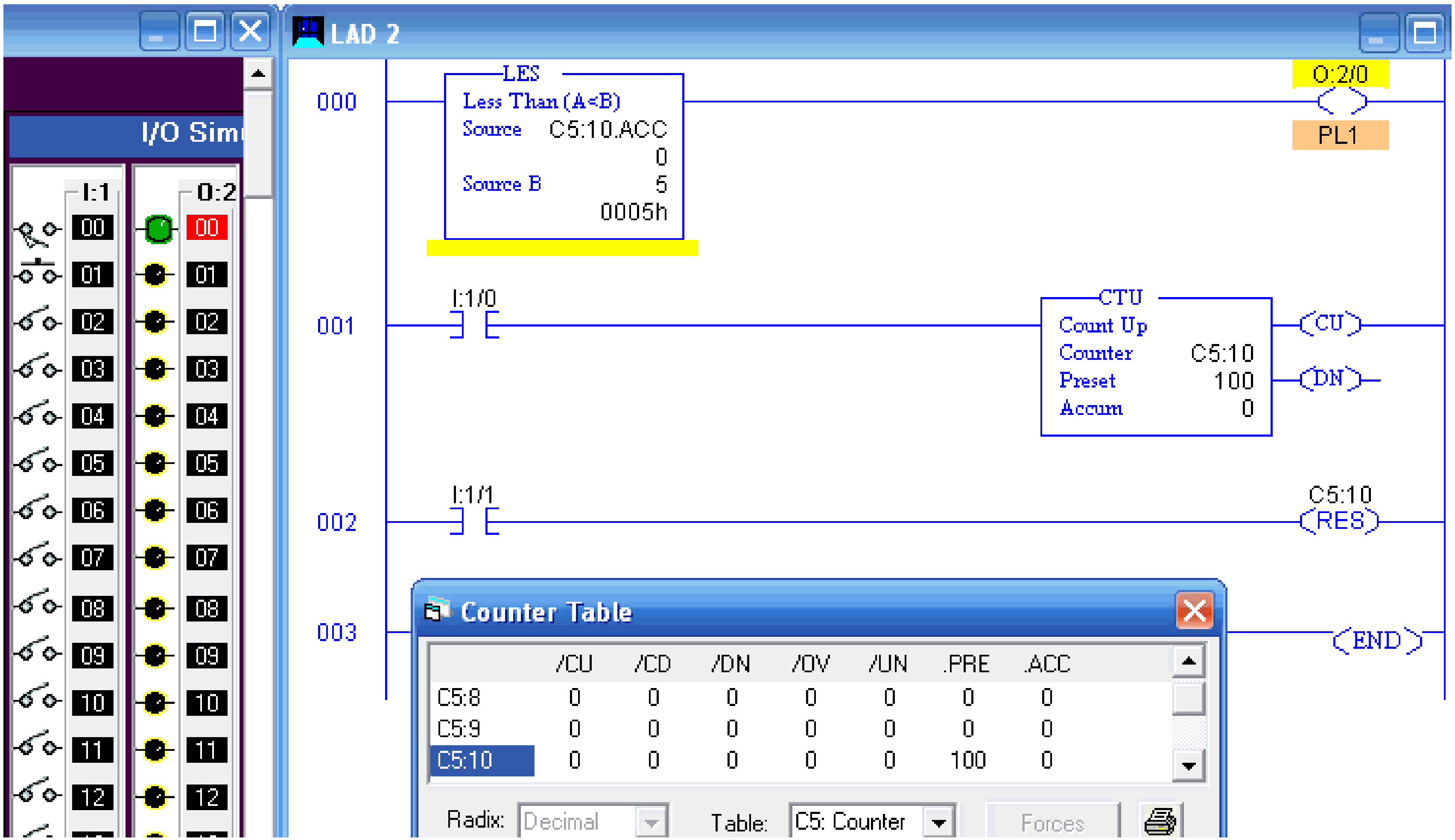
Simulated greater than (GRT) instruction program rung.



The *less than (LES)* instruction is logically true when source *A* is less than source *B*, otherwise it is logically false.

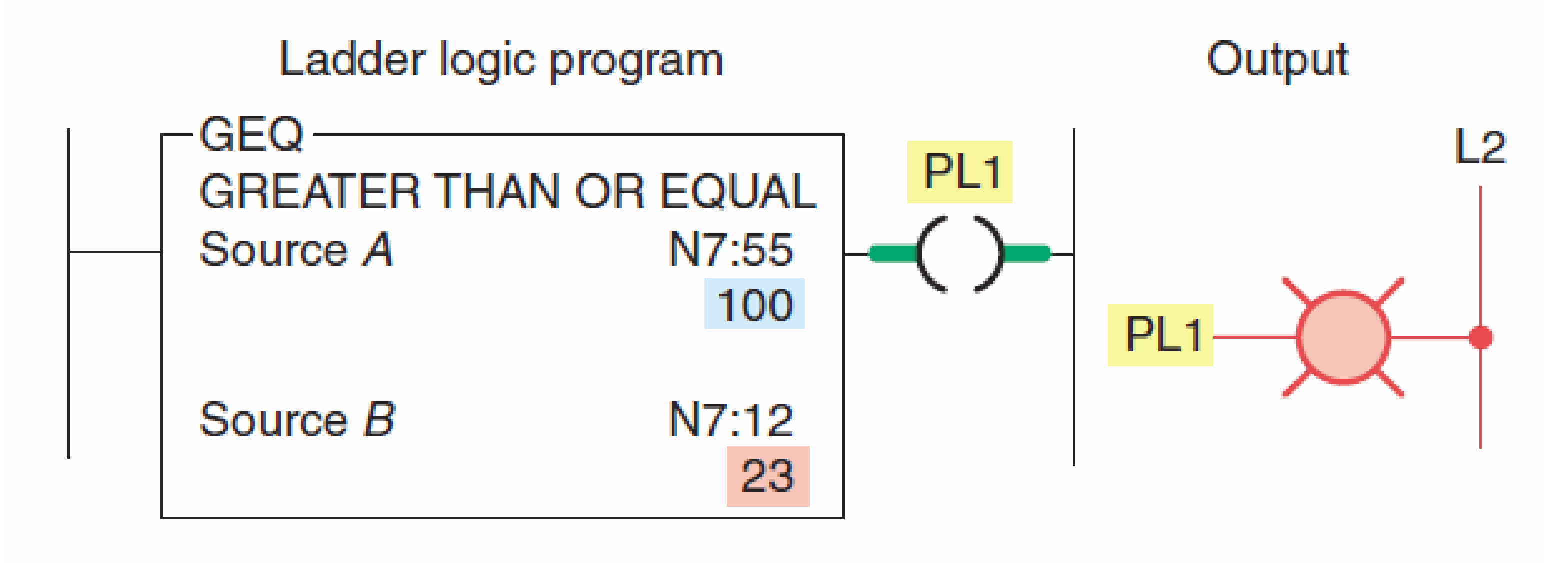


Simulated less than (LES) instruction program rung.

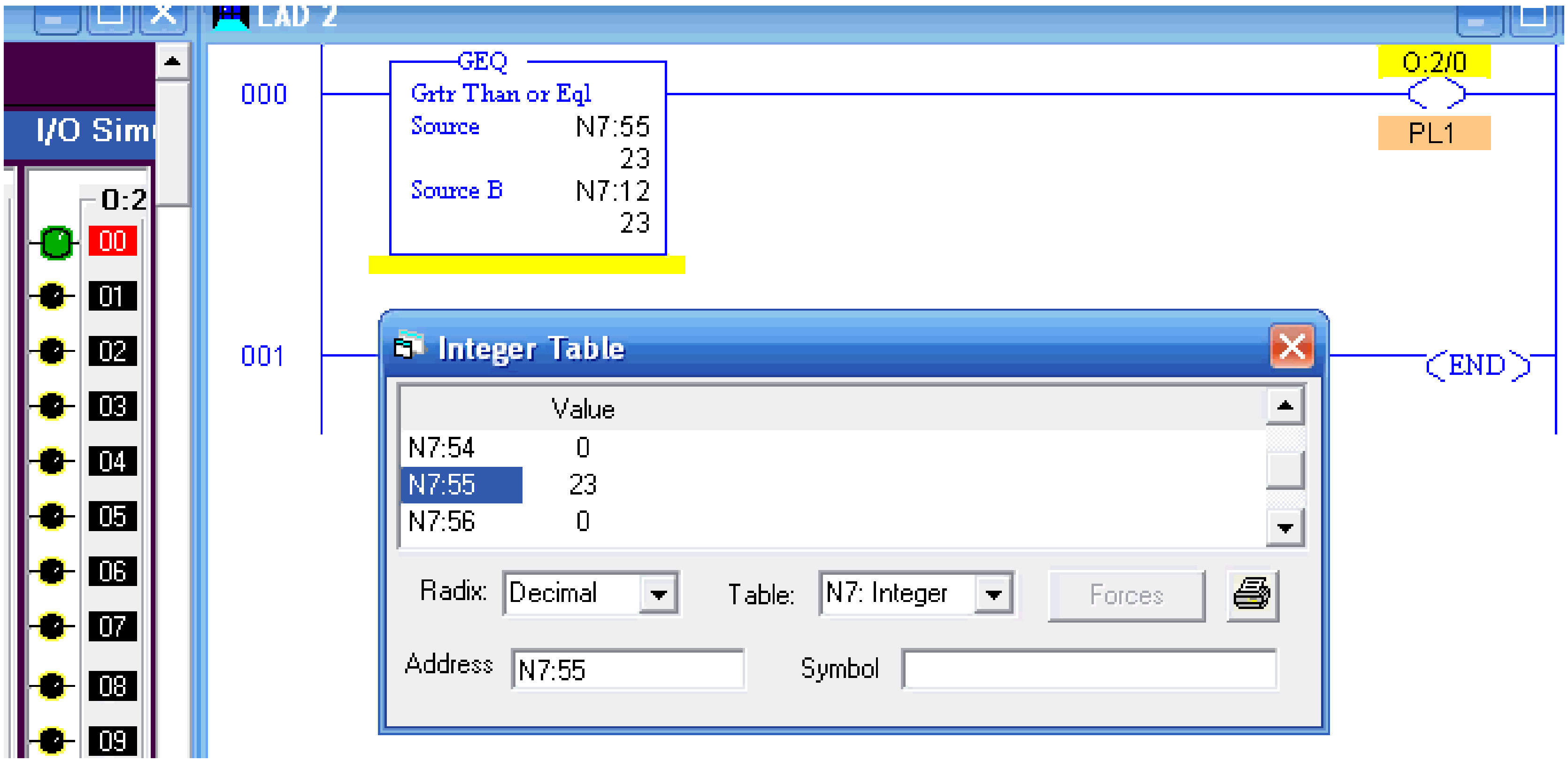


	/CU	/CD	/DN	/OV	/UN	.PRE	.ACC
C5:8	0	0	0	0	0	0	0
C5:9	0	0	0	0	0	0	0
C5:10	0	0	0	0	0	100	0

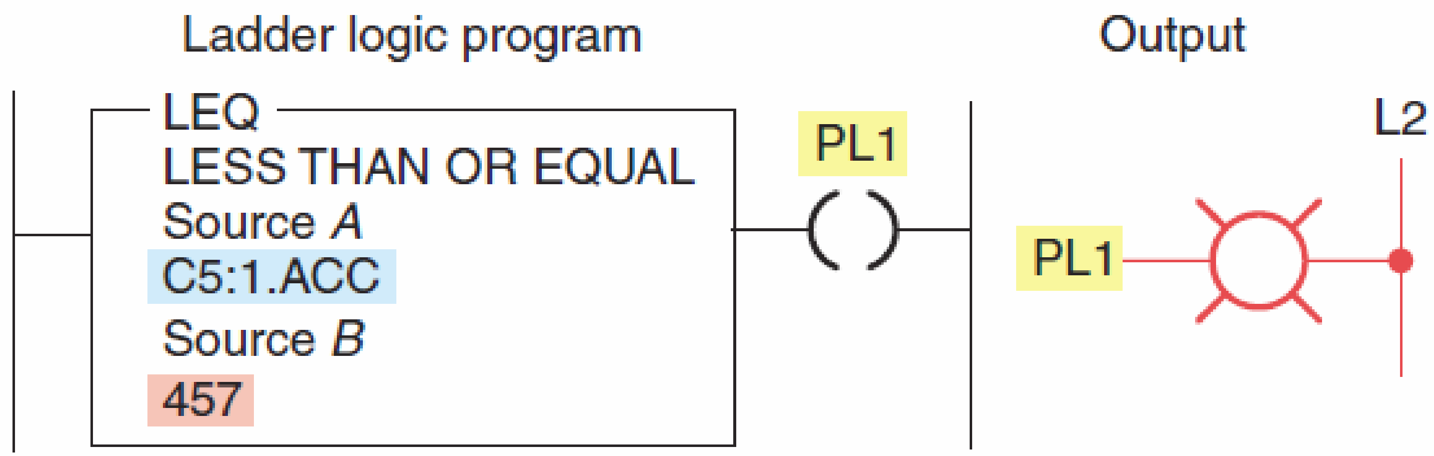
The *greater than or equal (GEQ)* instruction is logically true when source *A* is greater than or equal to source *B*, otherwise it is logically false.



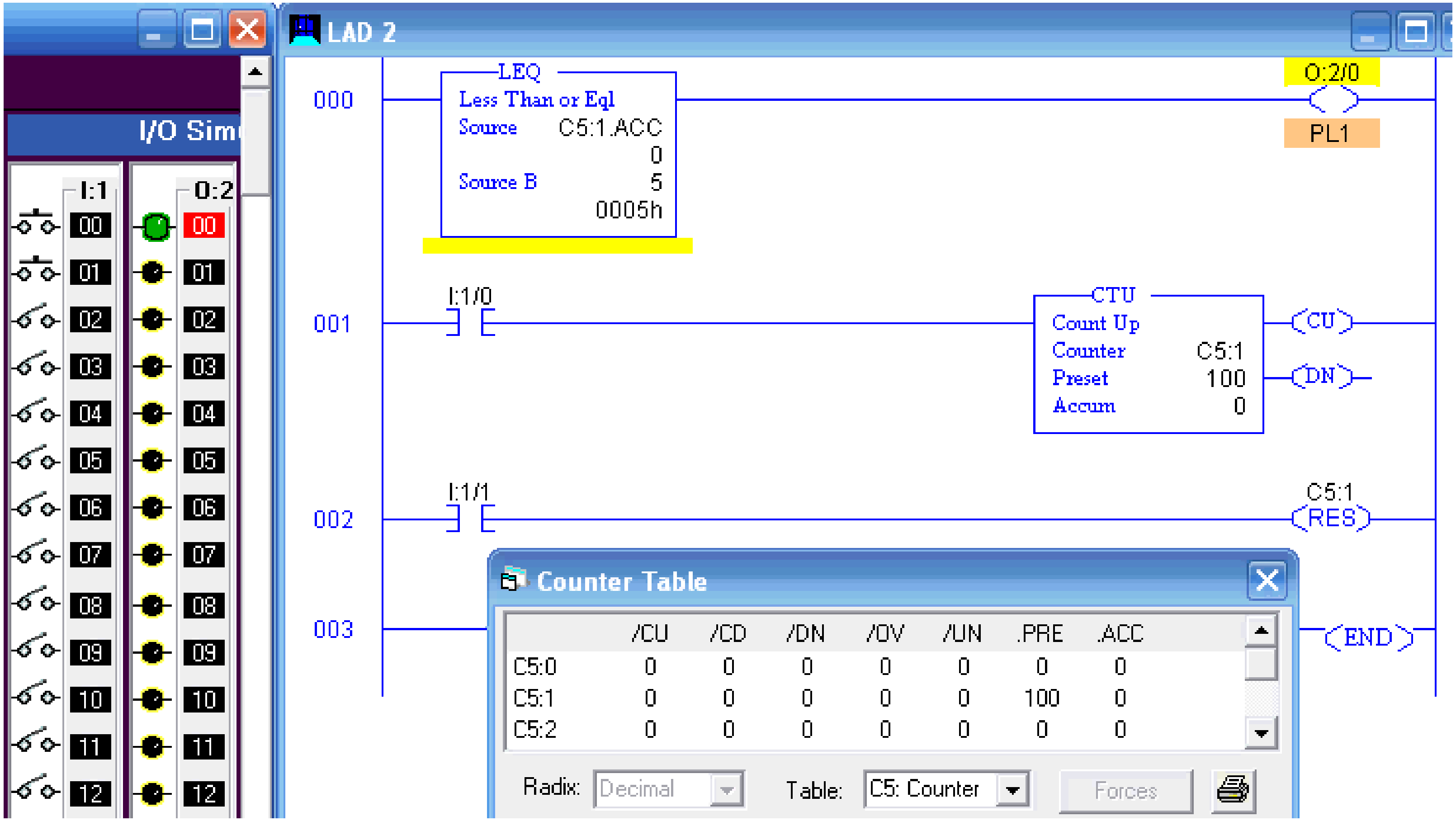
Simulated greater than or equal (GEQ) instruction program rung.



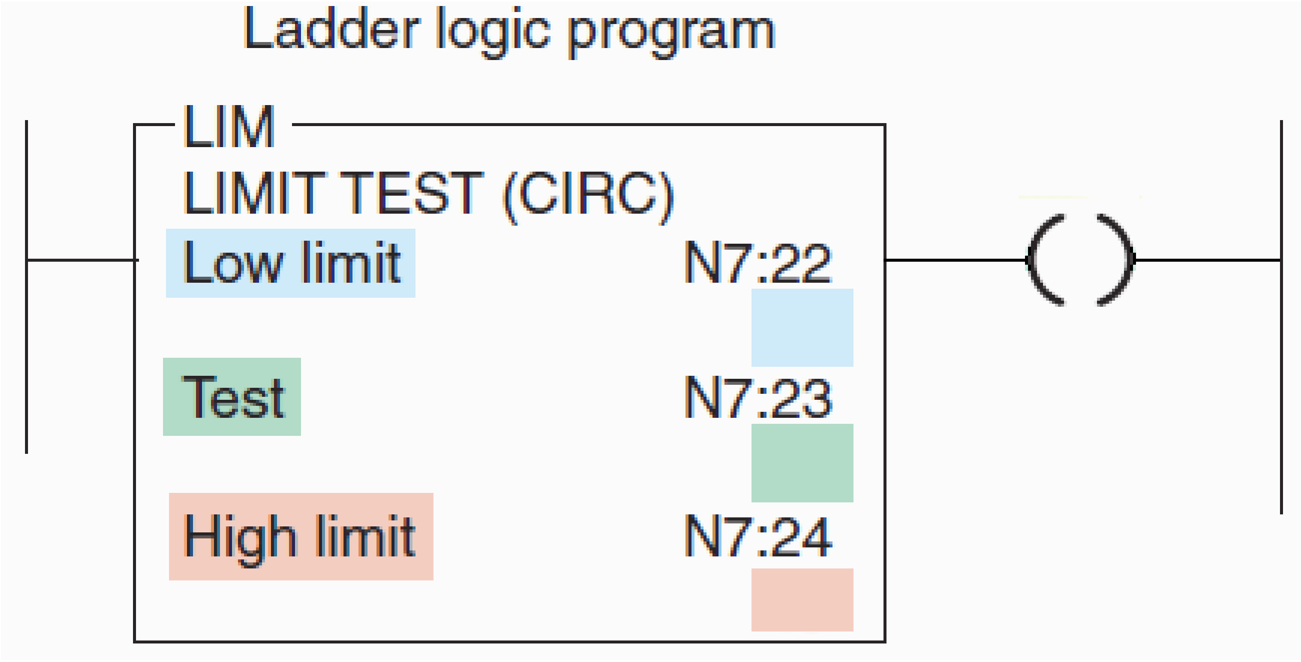
The *less than or equal (LEQ)* instruction is logically true when source *A* is less than or equal to source *B*, otherwise it is logically false.



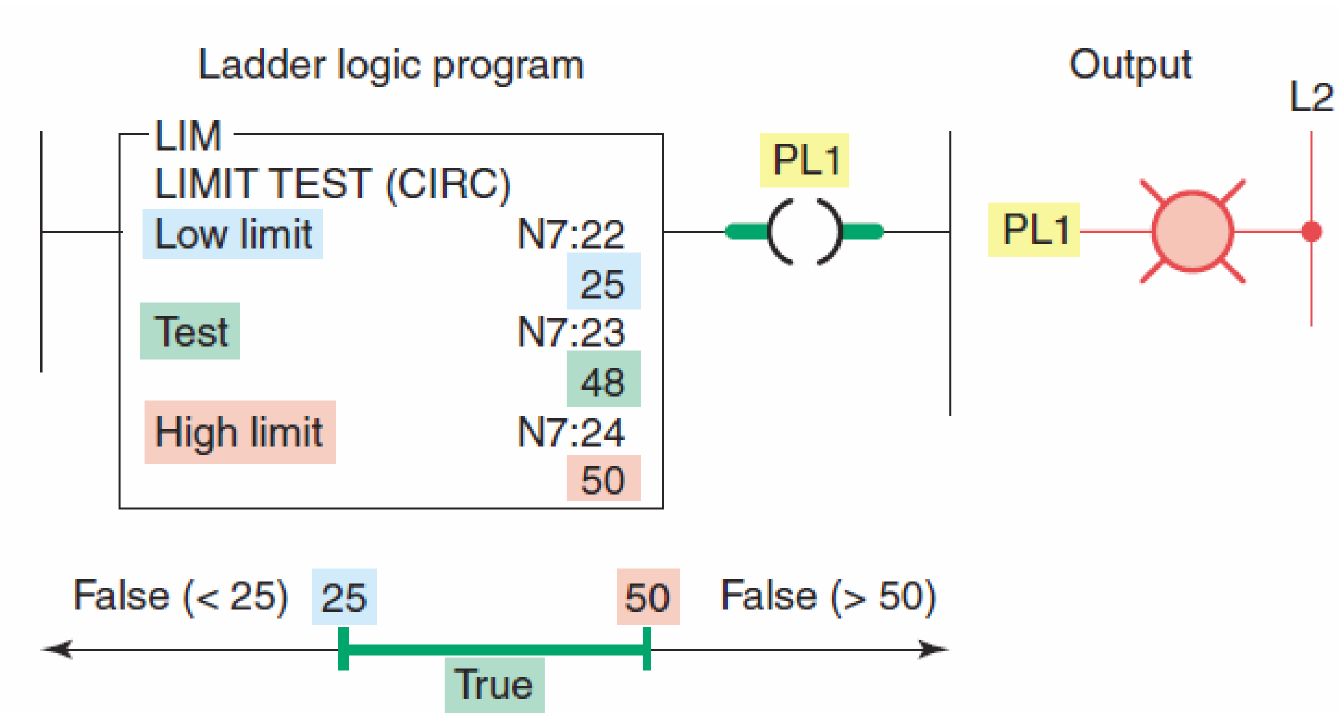
Simulated less than or equal (LEQ) instruction program rung.



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



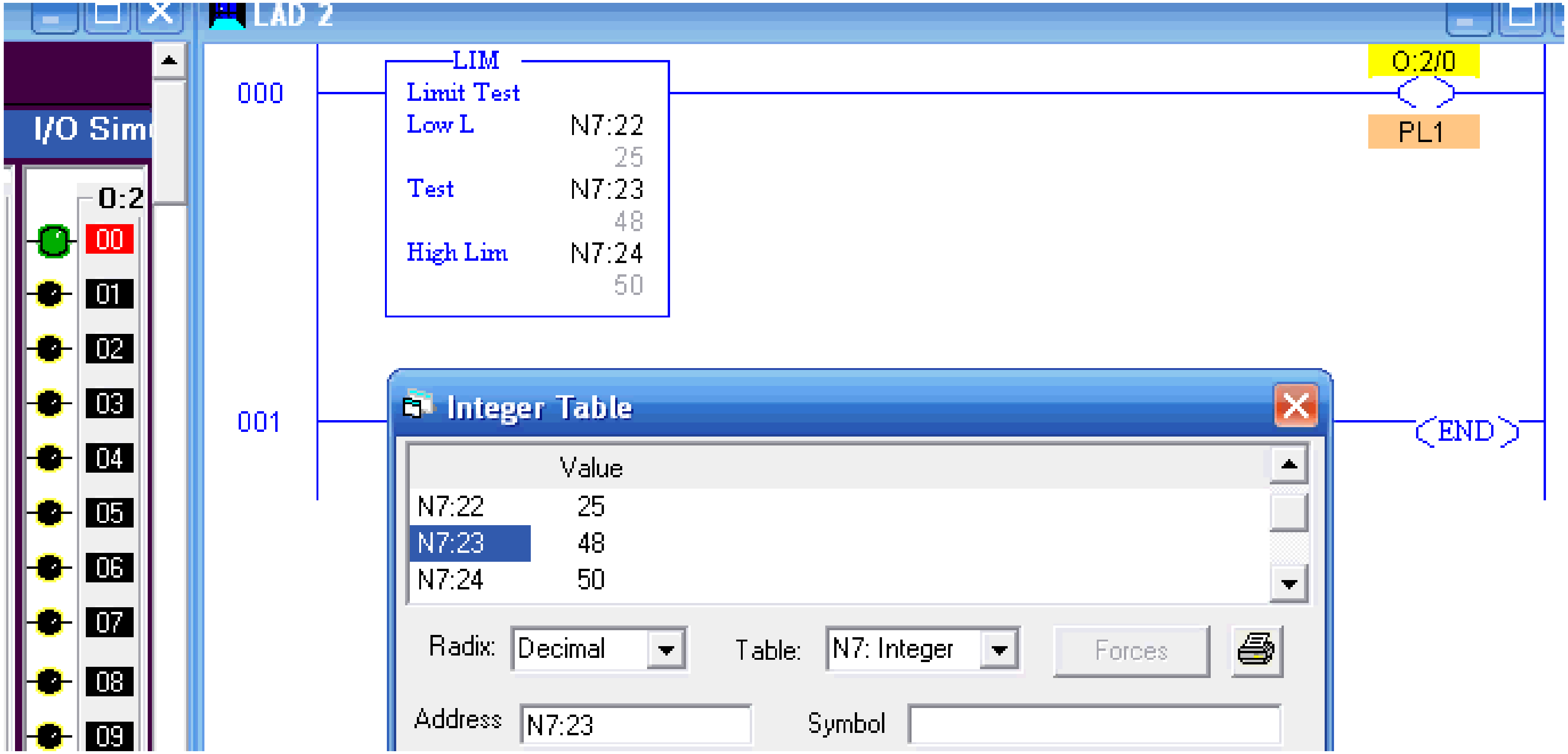
The low limit is less than the high limit.



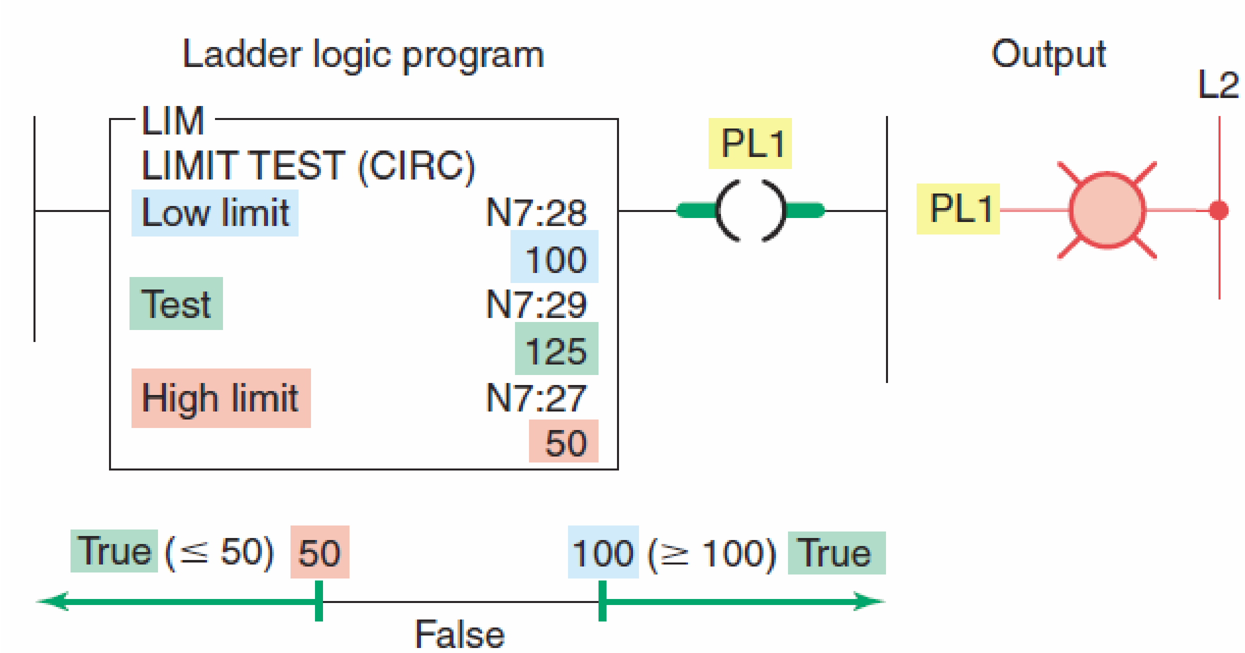
Instruction is true for test values 25 through 50.

Instruction is false for test values less than 25 or greater than 50.

Simulated limit test (LIM) instruction rung.



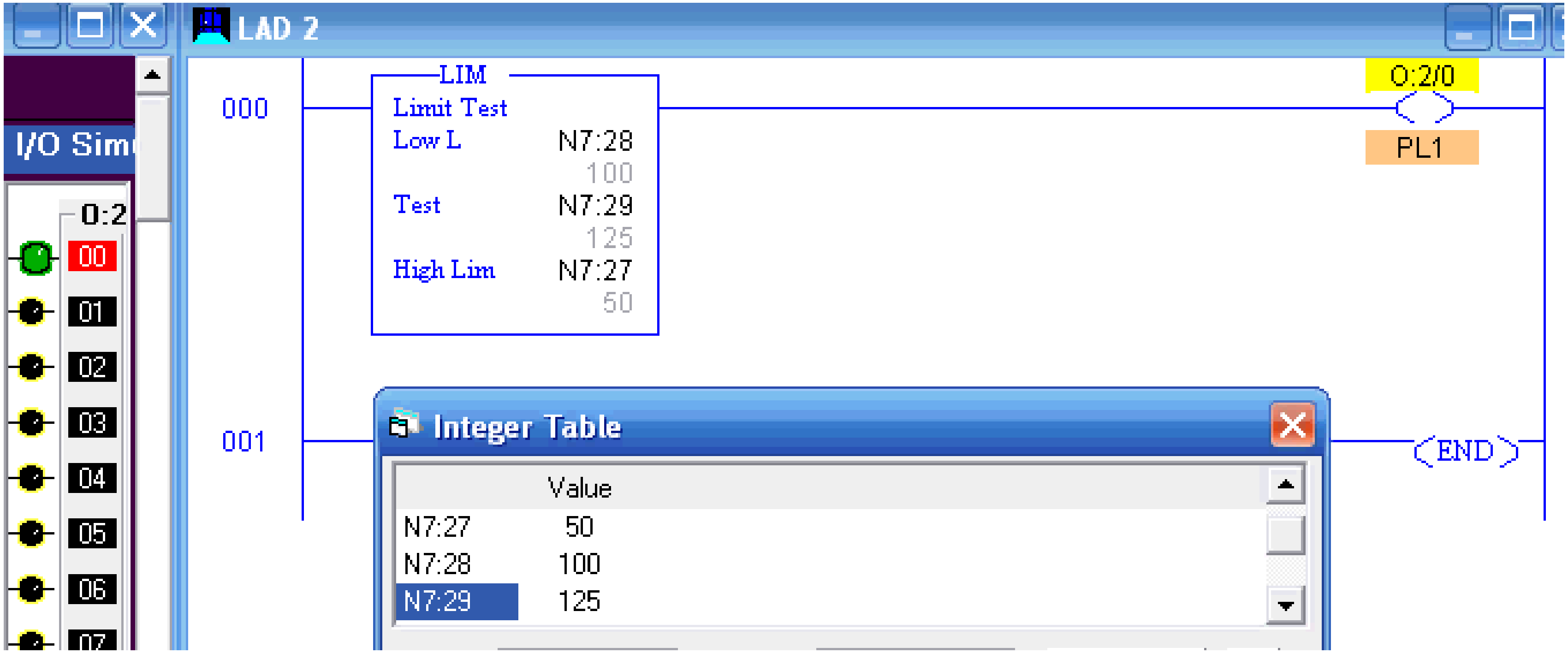
The low limit is greater than the high limit.



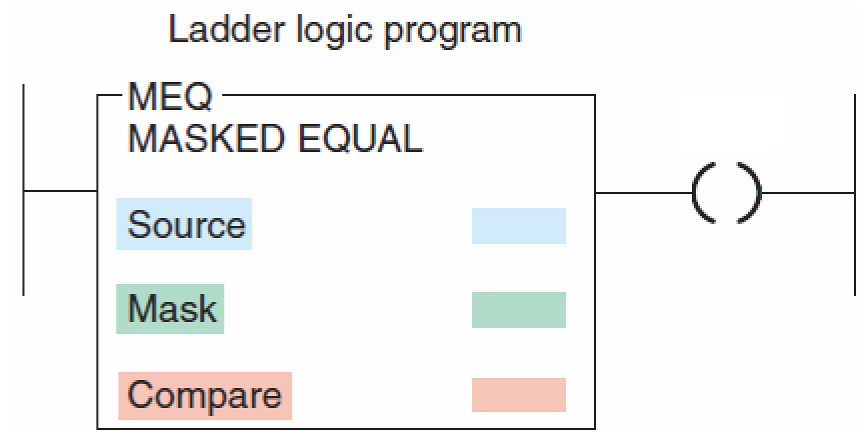
Instruction is true for test values of 50 and less than 50 and for test values of 100 and greater than 100.

Instruction is false for test values greater than 50 and less than 100.

Simulated limit test (LIM) instruction rung.

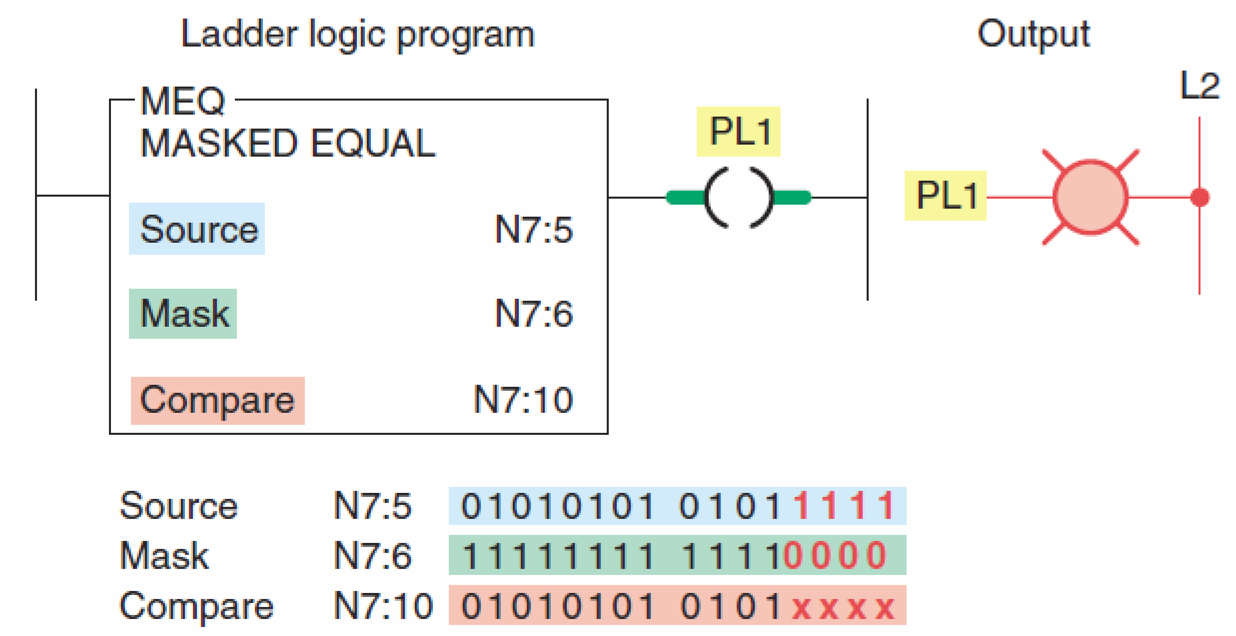


The *masked comparison for equal (MEQ)* instruction compares a value from a source address with data at a compare address and allows portions of the data to be masked.



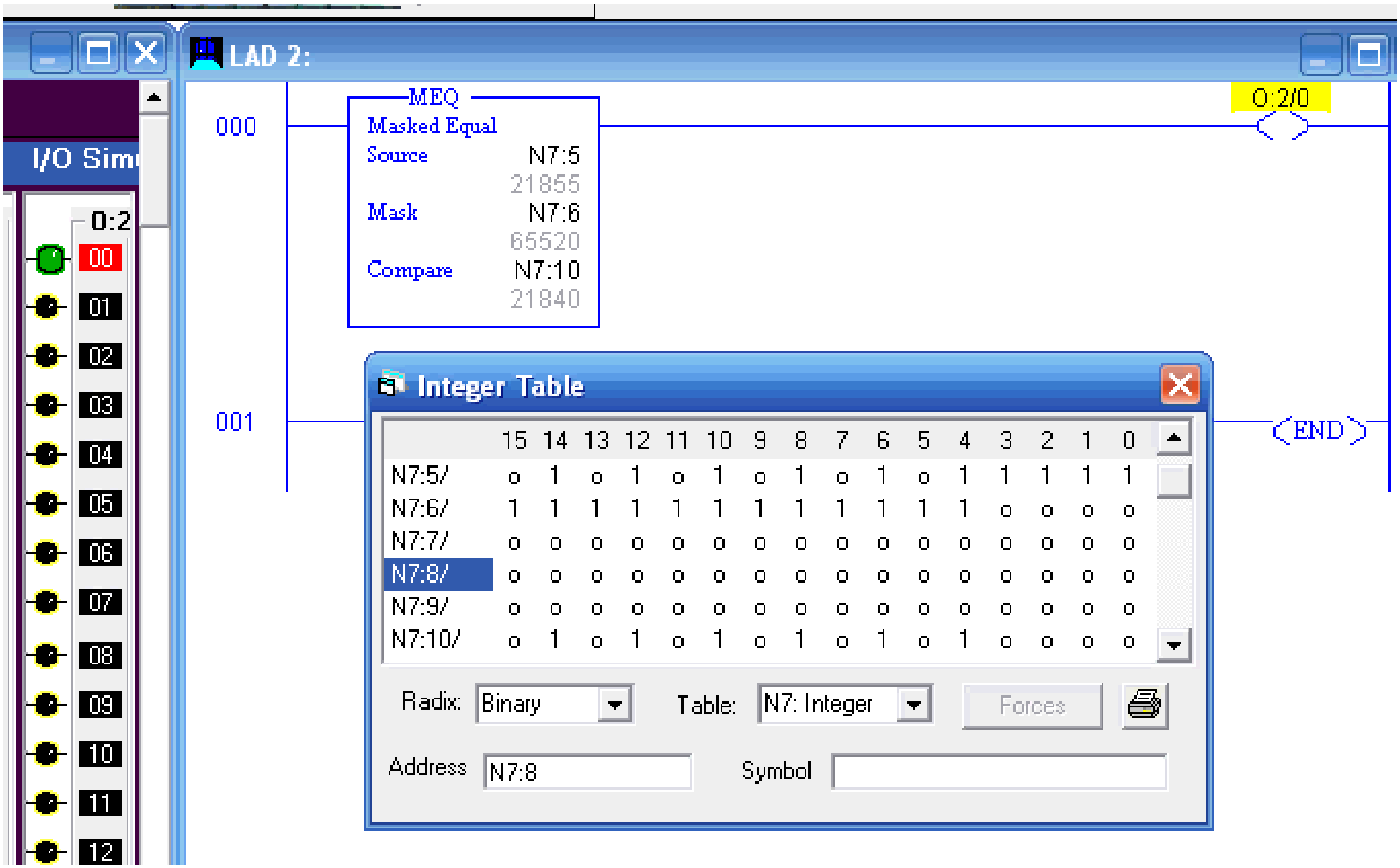
The MEQ instruction can be used to compare the correct position of up to 16 limit switches when the source contains the limit switch address and the compare stores their desired states.

MEQ instruction rung.



When the data at the source address match the data at the compare address bit-by-bit (less masked bits), the instruction is true.

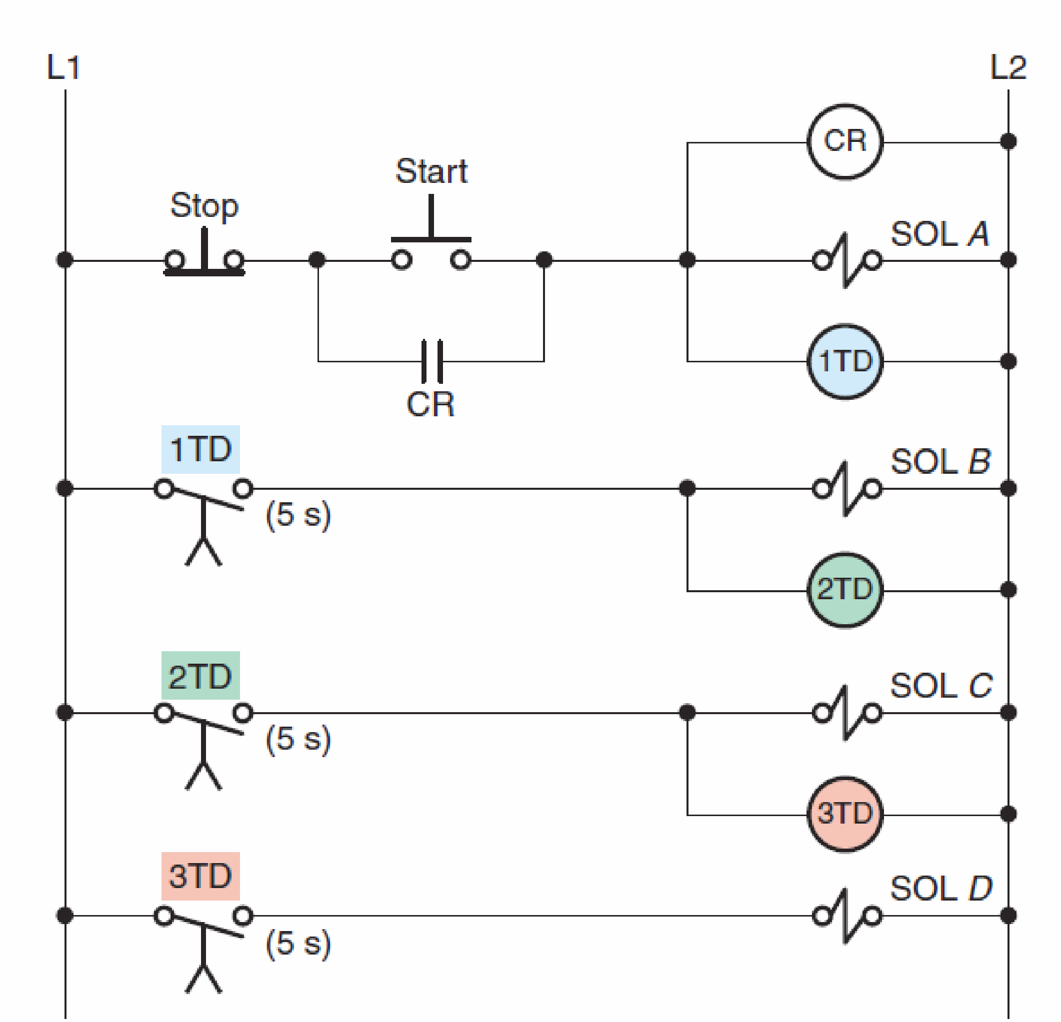
Simulated MEQ instruction rung.



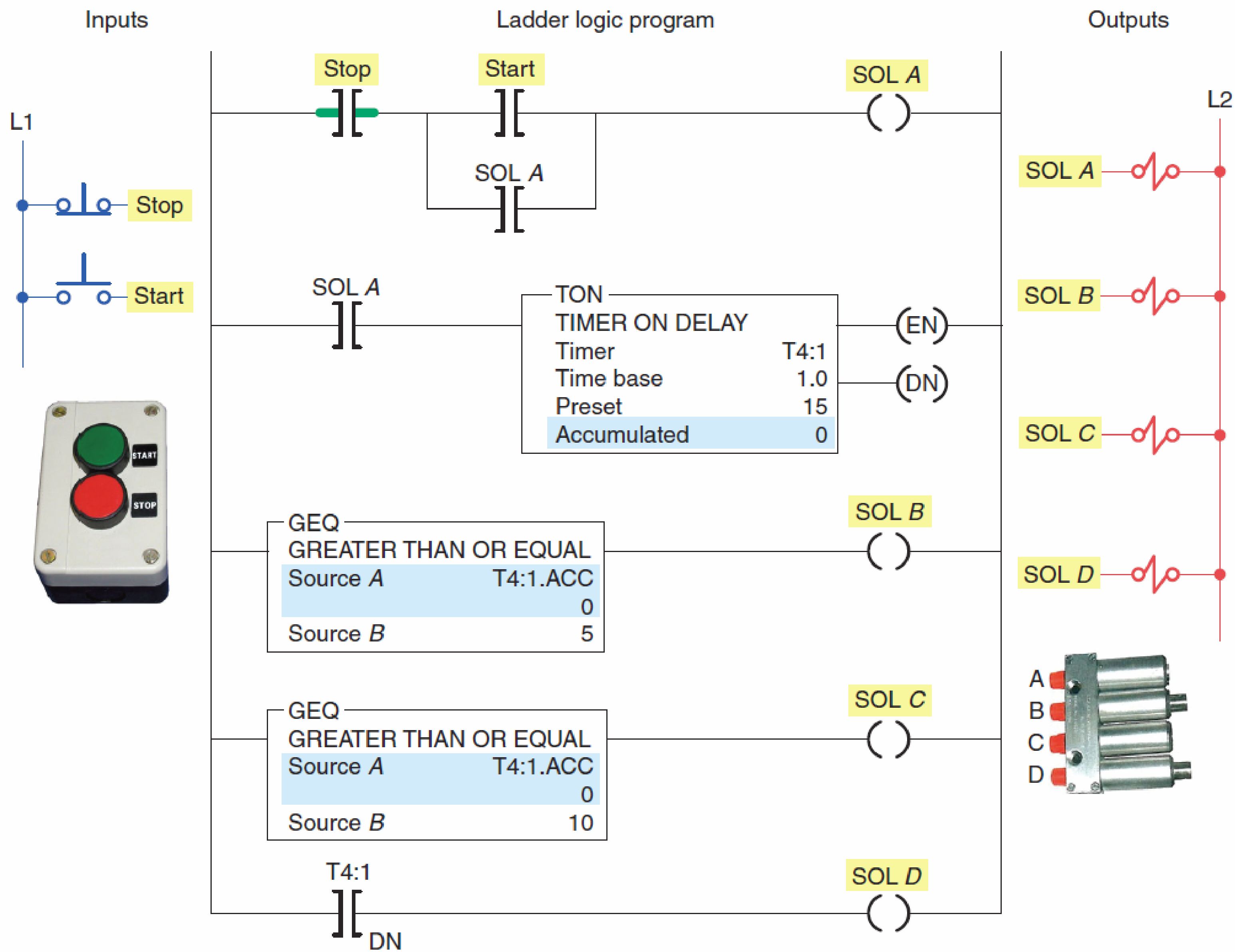
10.4

Data Manipulation Programs

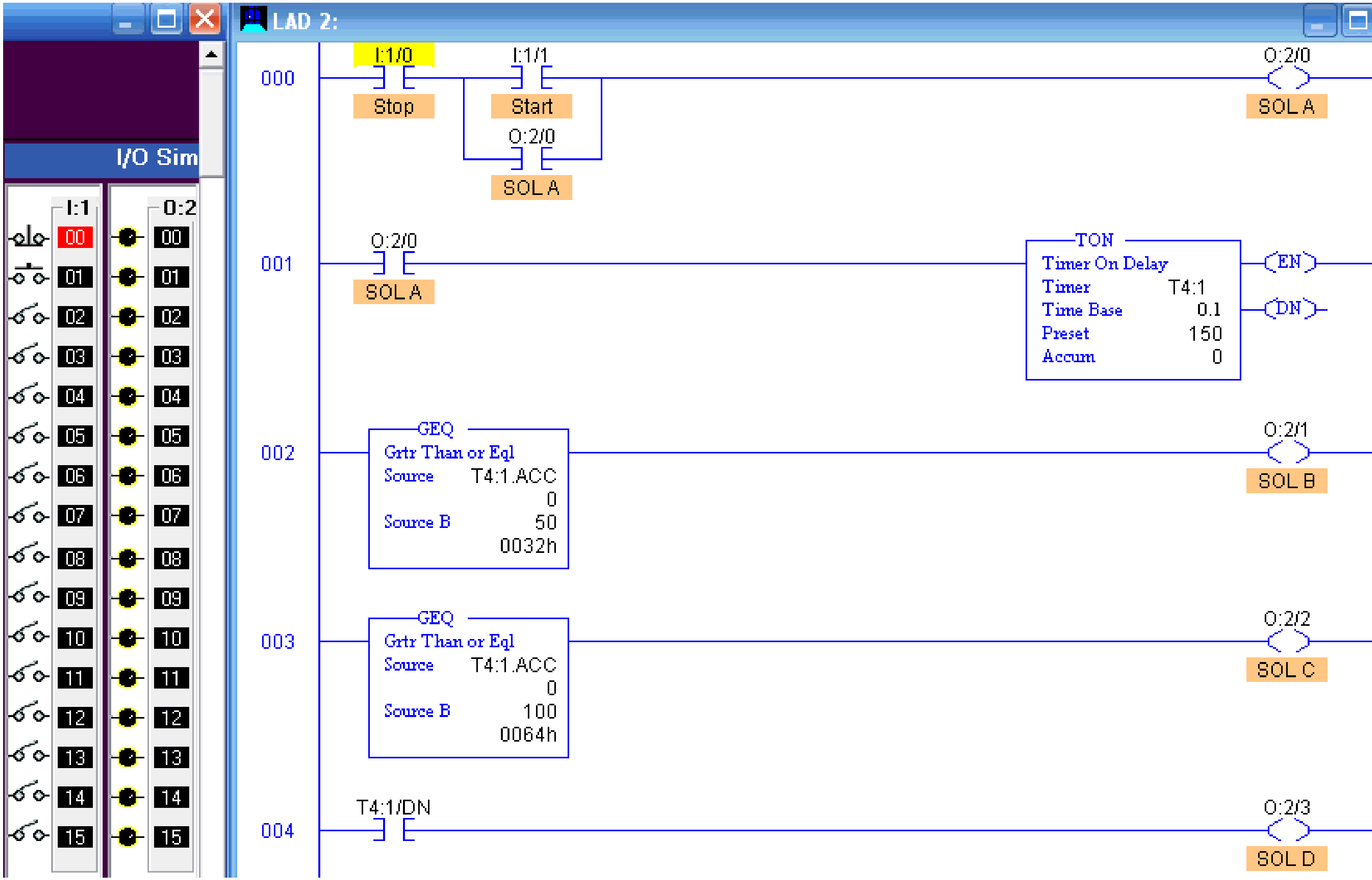
Hardwired relay-operated, time-delay circuit.



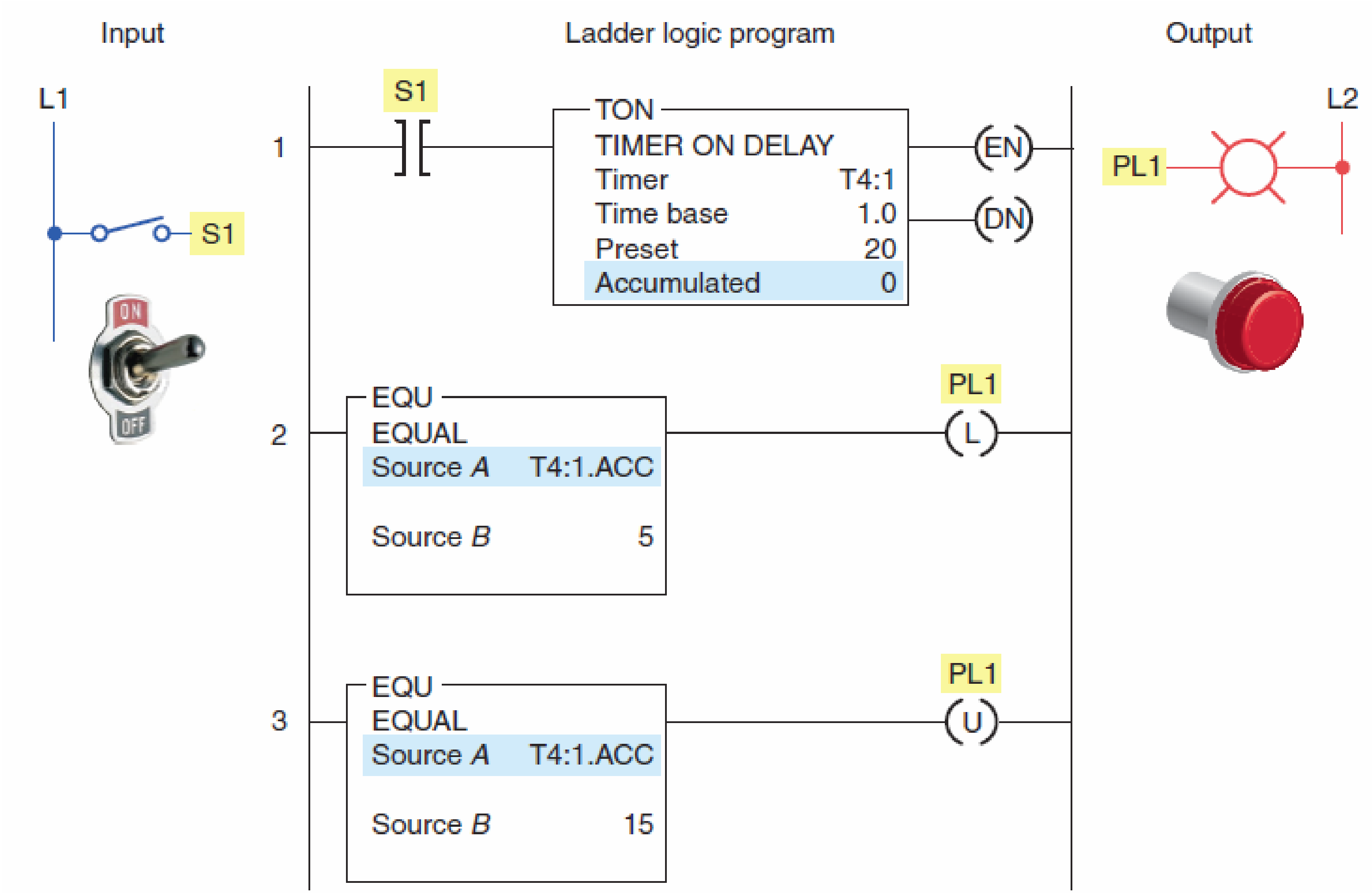
Equivalent programmed time-delay circuit.



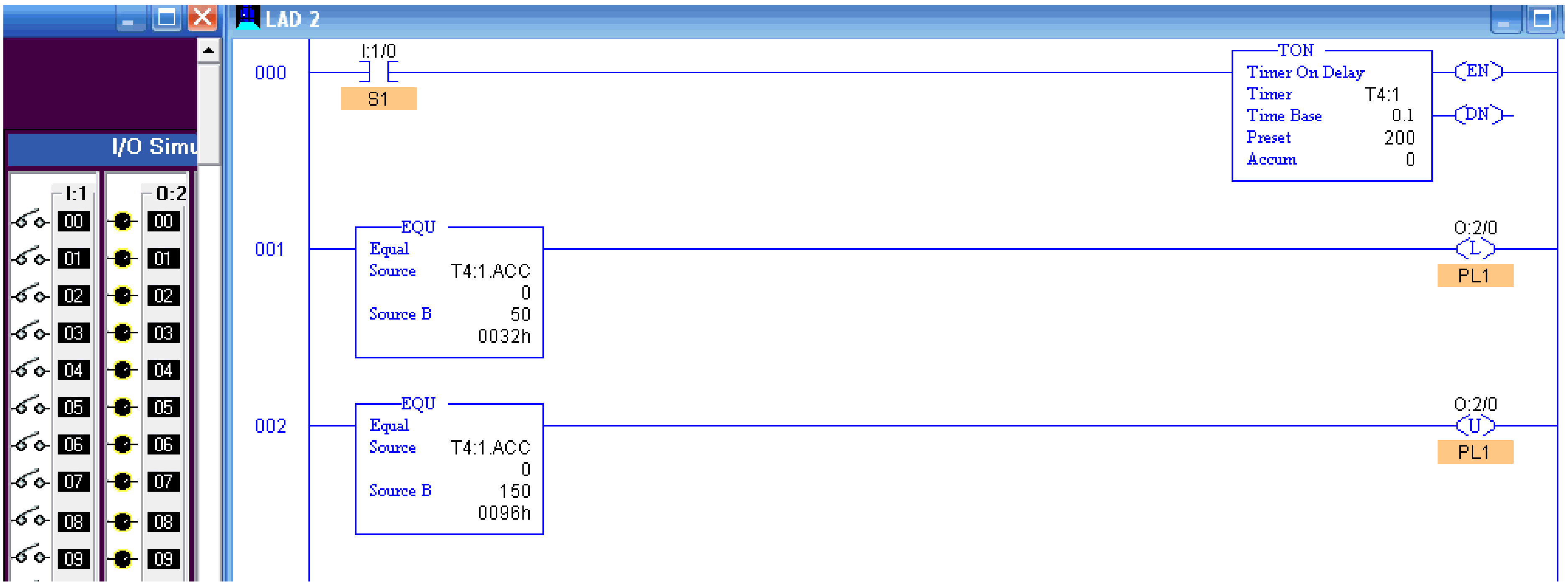
Simulated programmed time-delay circuit.



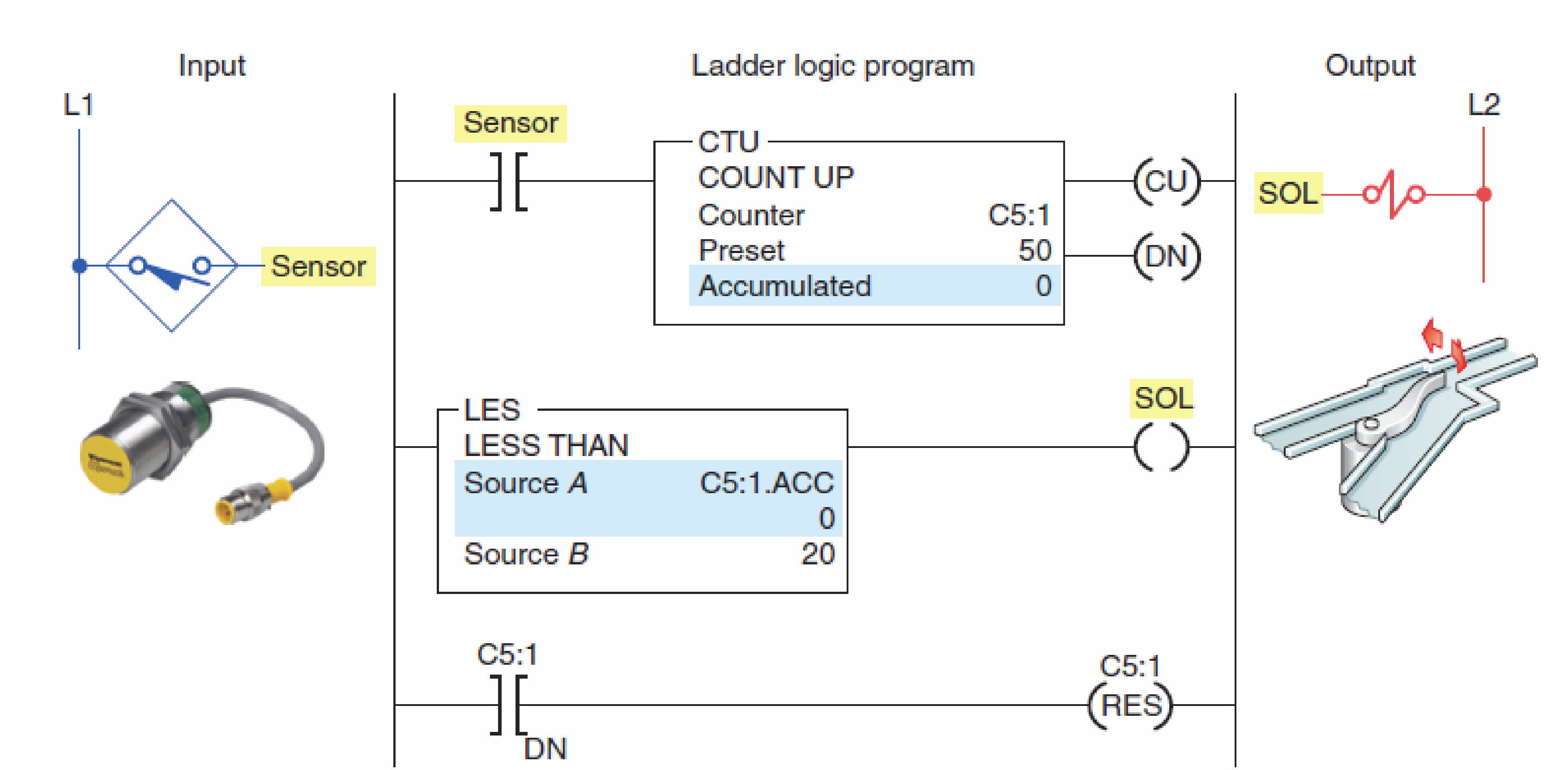
Timer program implemented using the EQU instruction.



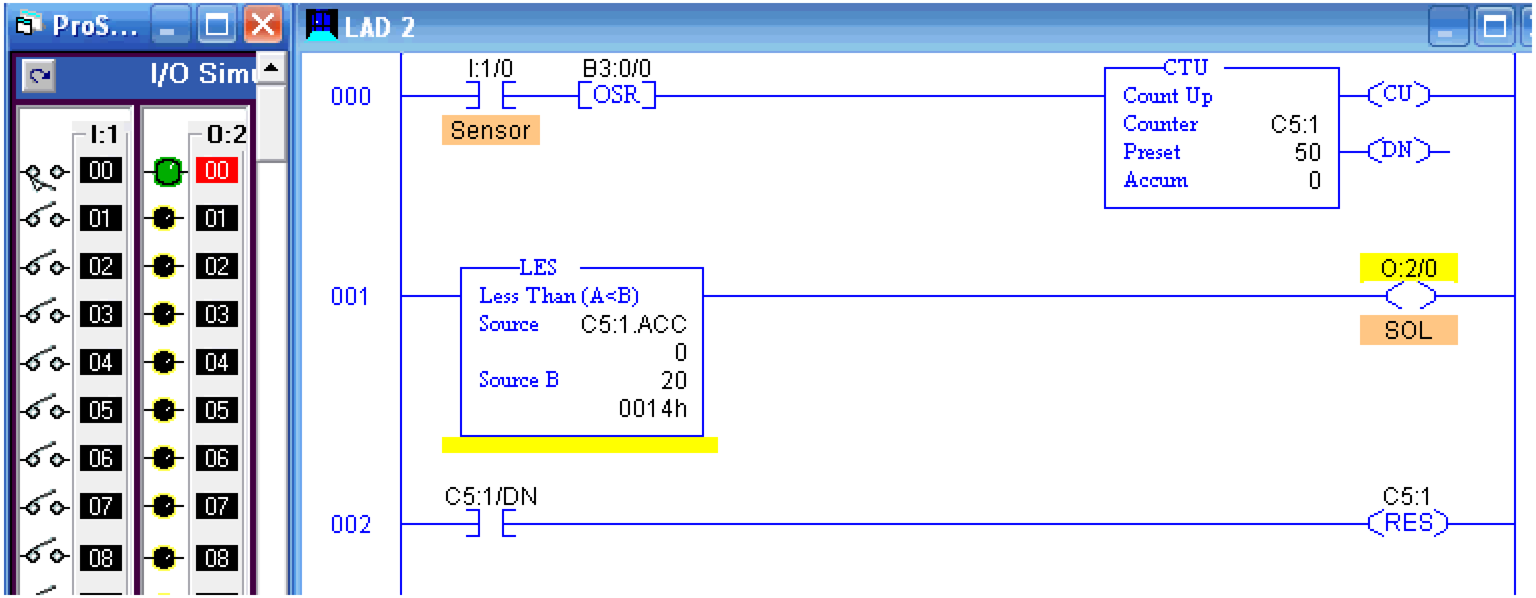
Simulated timer program implemented using the EQU instruction.



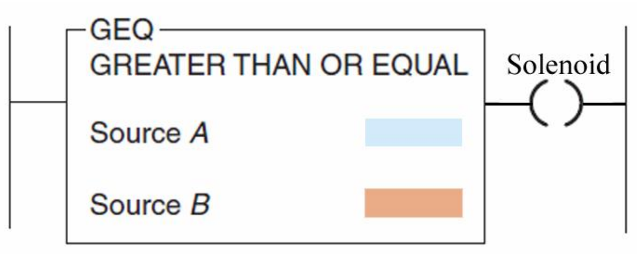
Counter program implemented using the LES instruction.



Simulated counter program implemented using the LES instruction.



Vessel filling operation.



- The receiving vessel has its weight monitored continuously by the PLC program as it fills.
- When the weight reaches a preset value, the flow is cut off.
- Should the system leak additional material into the vessel, the total weight of the material could rise above the preset value causing an equal instruction, if used, to go false and the vessel to overfill.

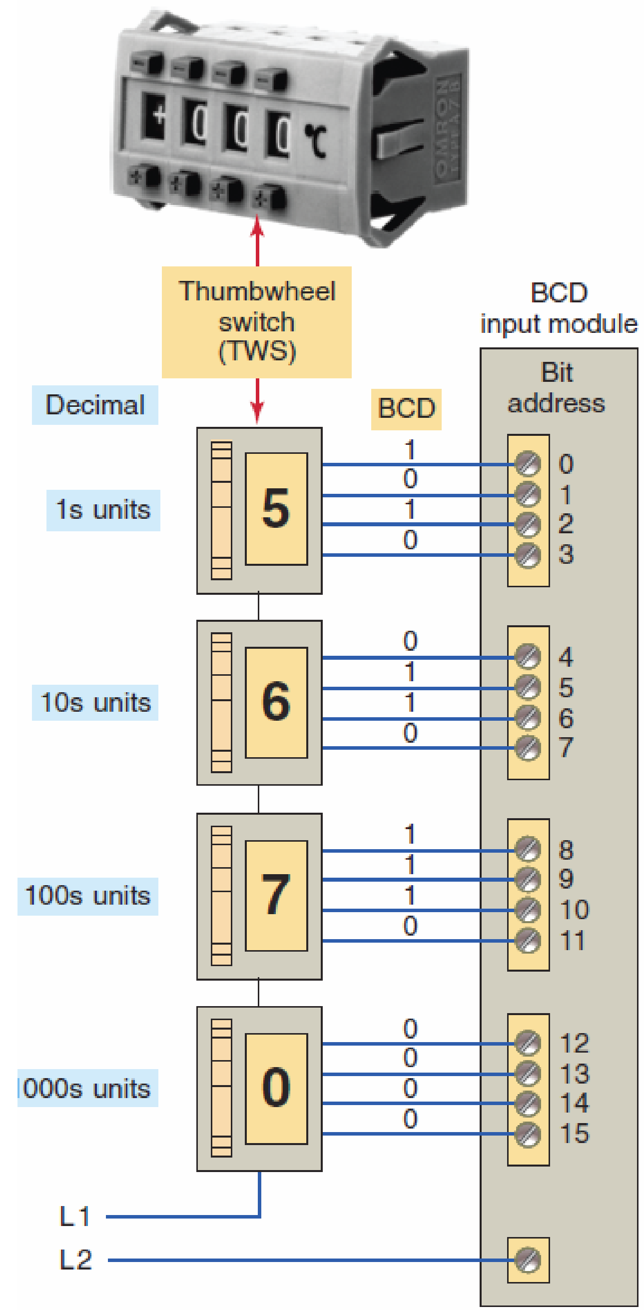
10.5

Numerical Data I/O Interfaces

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

Each one of the switches provides four binary digits at its output that correspond to the decimal number selected on the switch.

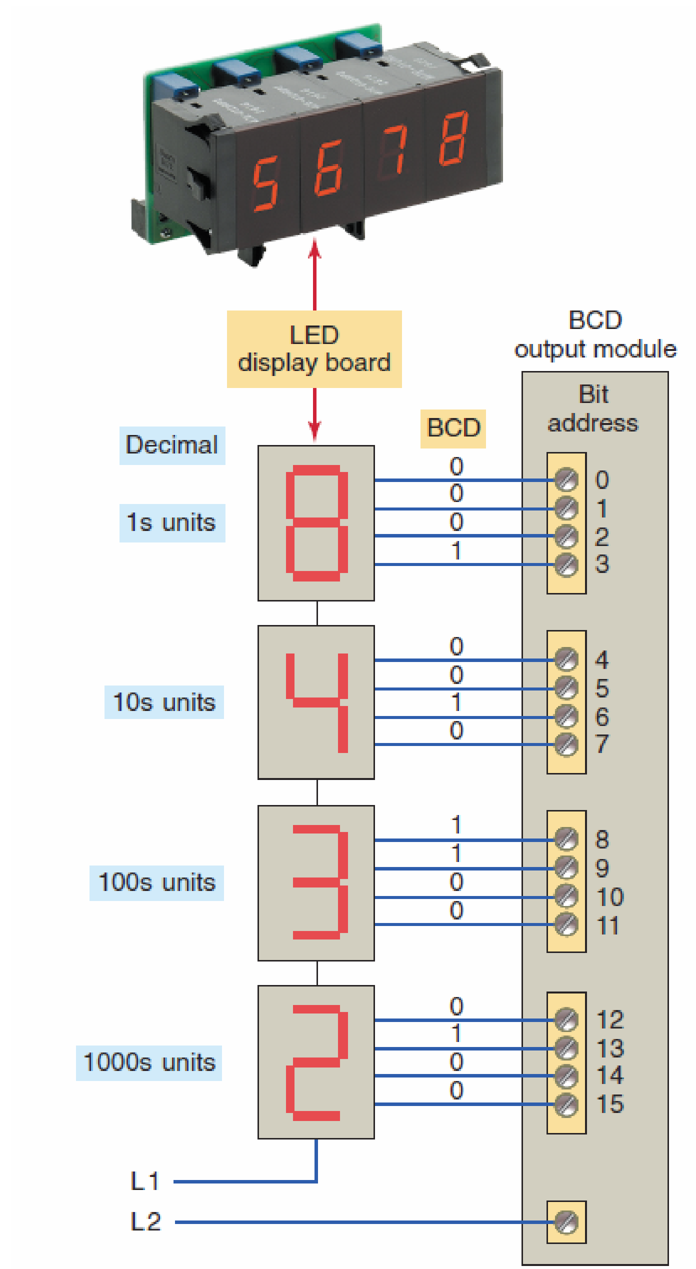
The BCD input module allows the processor to accept the 4-bit digital codes and input their data into specific register or word locations.



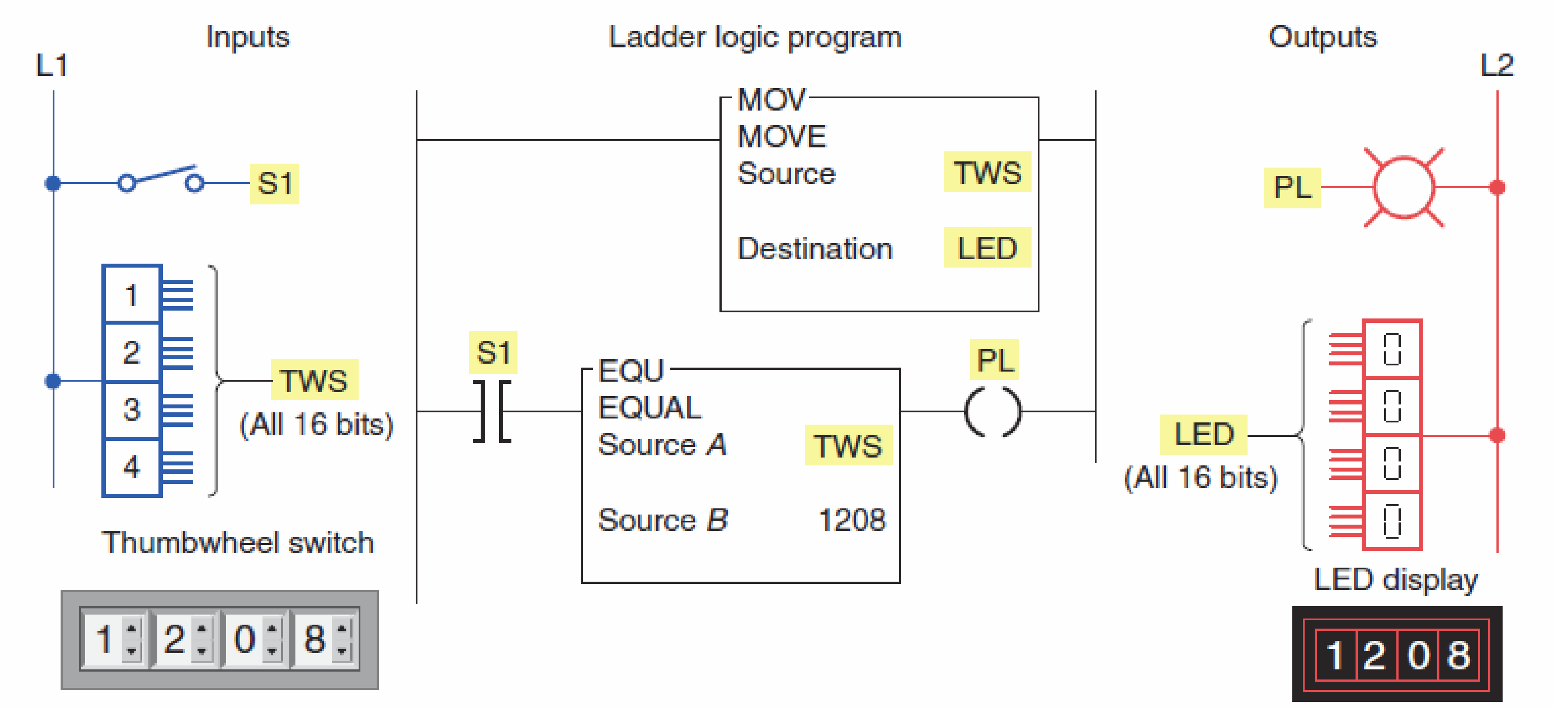
The *seven-segment LED* display is a typical Binary Coded Decimal output device.

It displays a **decimal number** that corresponds to the **BCD value** it receives at its input.

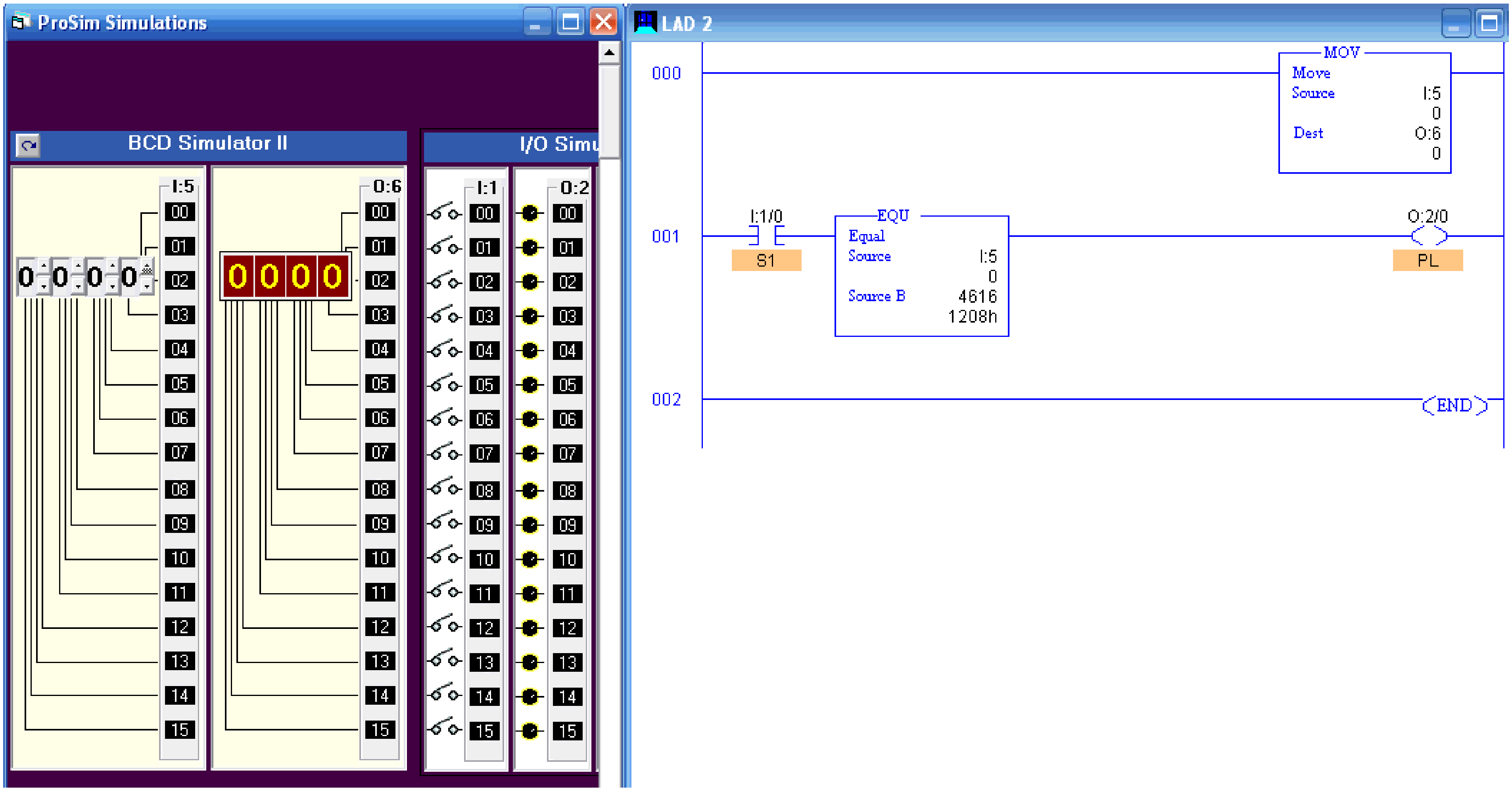
The **BCD output module** is used to output data from a specific register or word location.



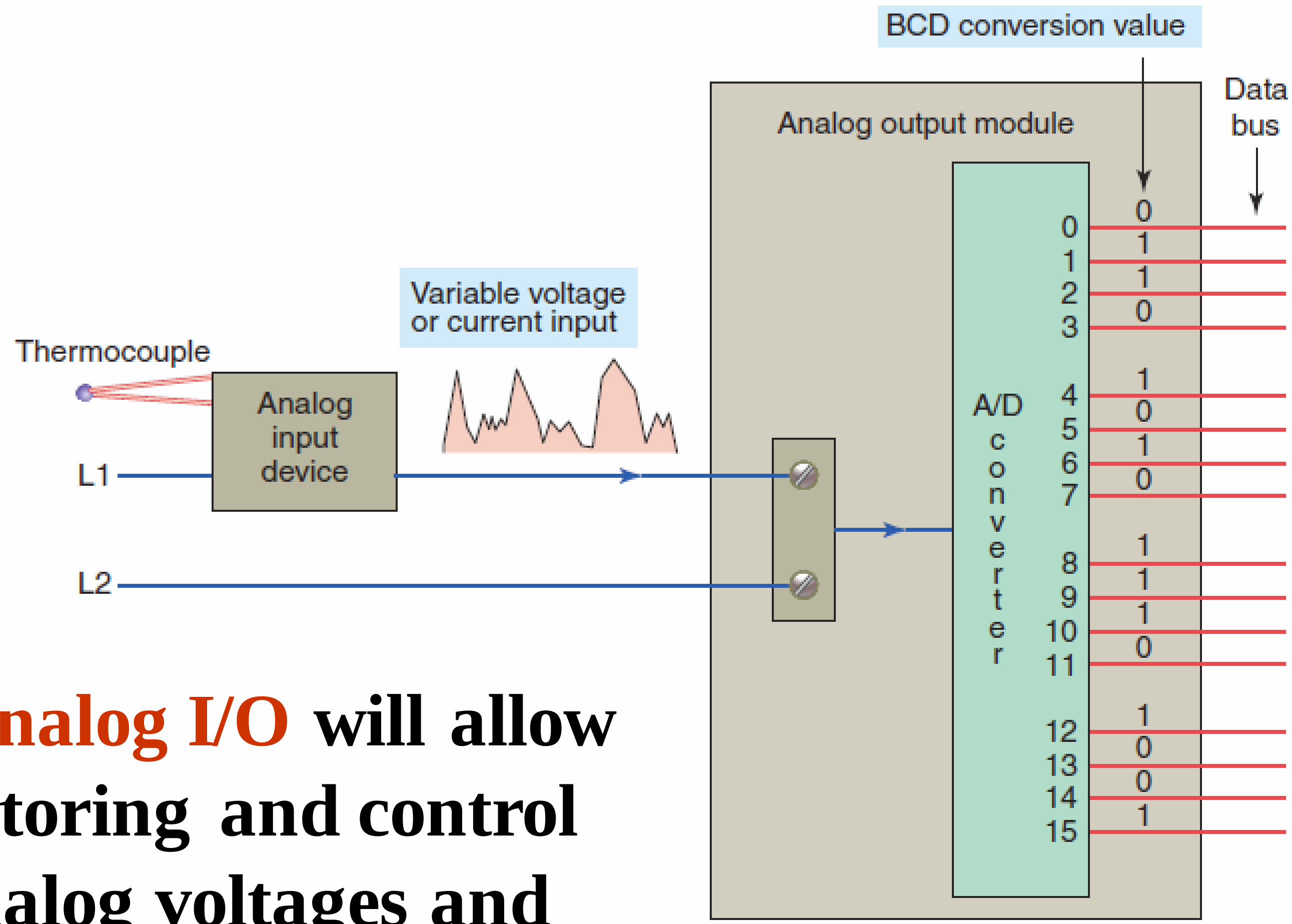
Program for monitoring the setting of a thumbwheel switch.



Simulated program for monitoring the setting of a thumbwheel switch.

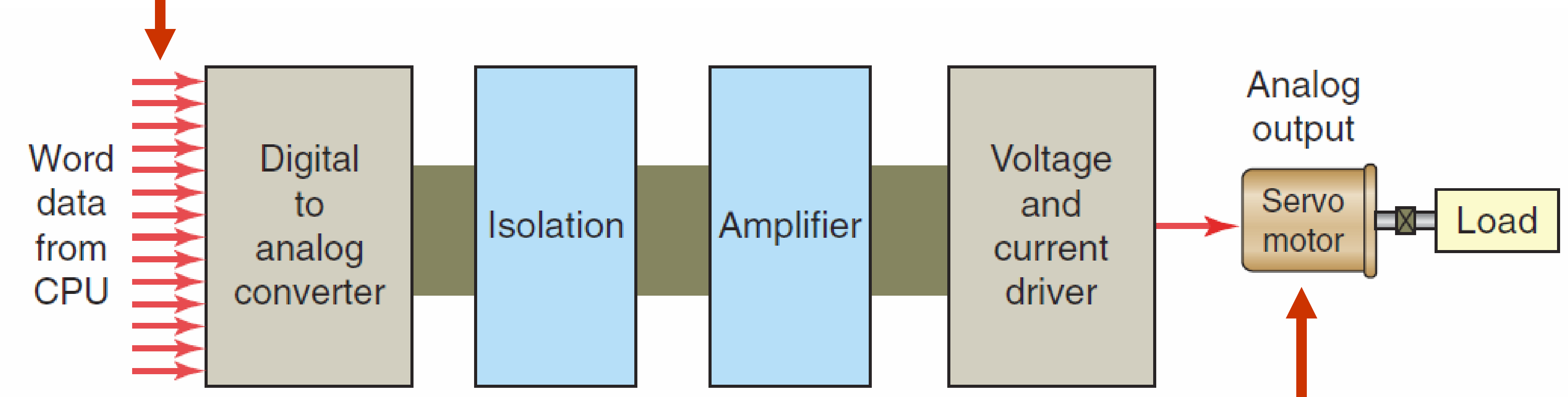


Analog modules **convert analog signals to 16-bit digital signals (input) or 16-bit digital signals to analog values (output).**



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

The *analog output interface module* receives numerical data from the processor.

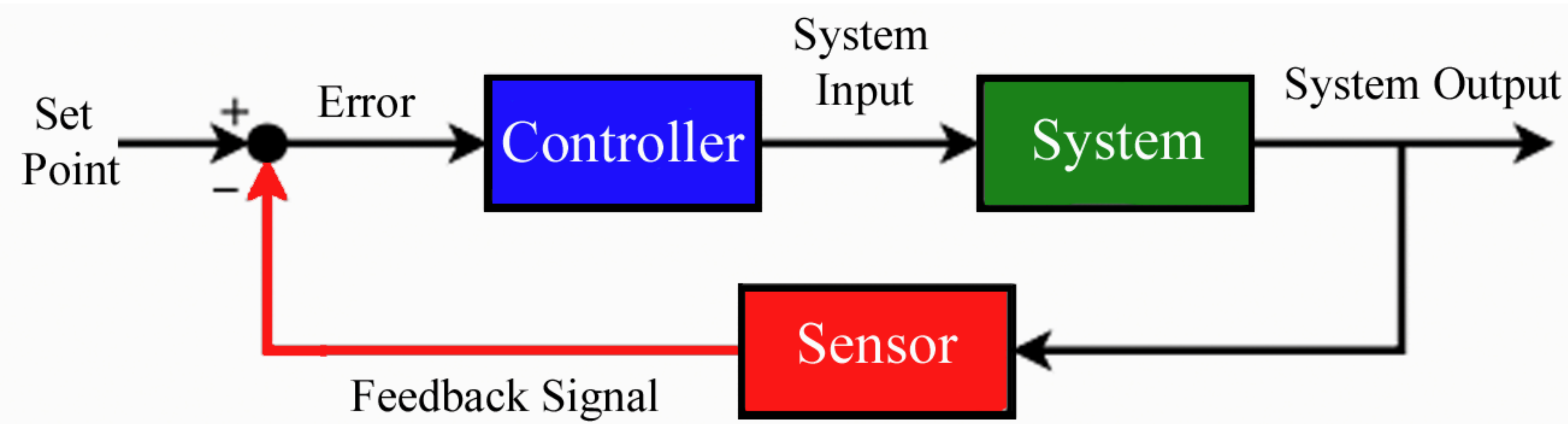


These data are then translated into a proportional voltage or current to control an analog field device.

10.6

Closed loop Control

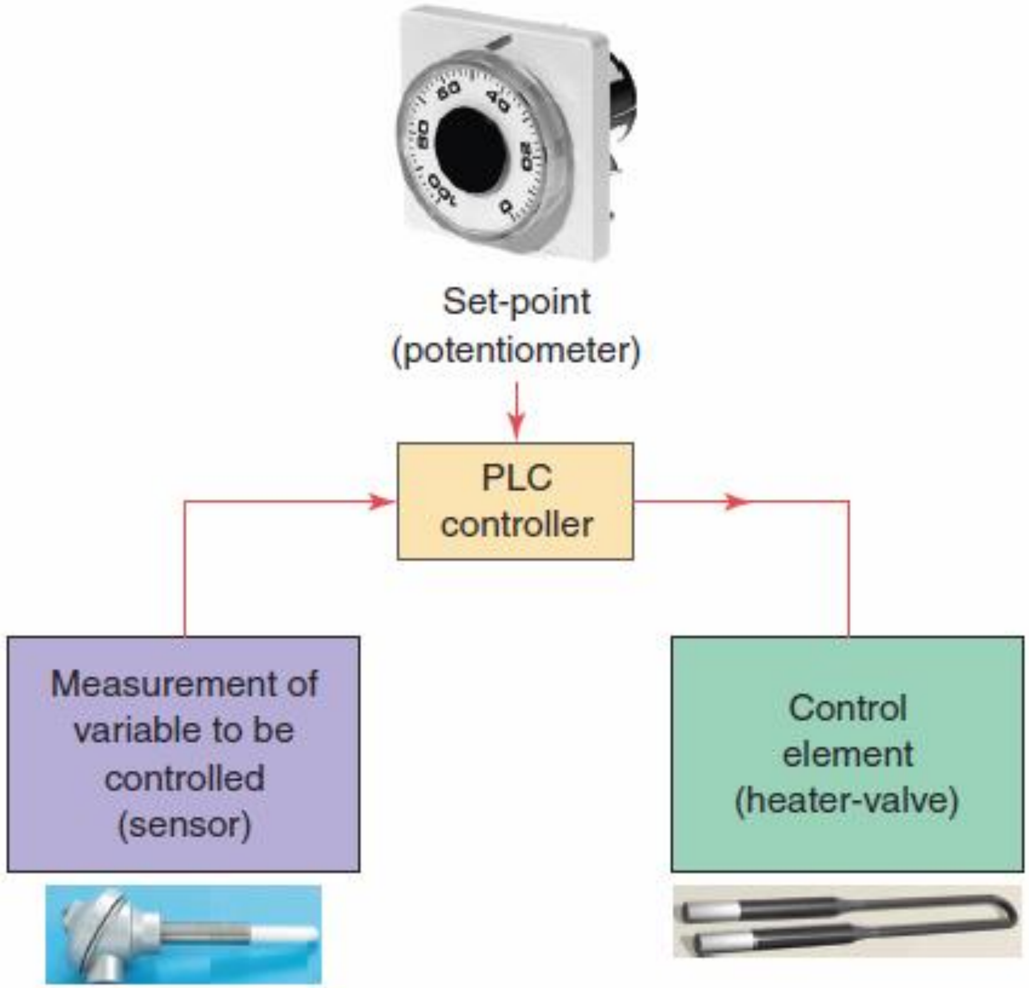
In *open-loop control*, no feedback loop is employed and system variations which cause the output to deviate from the desired value are not detected or corrected.



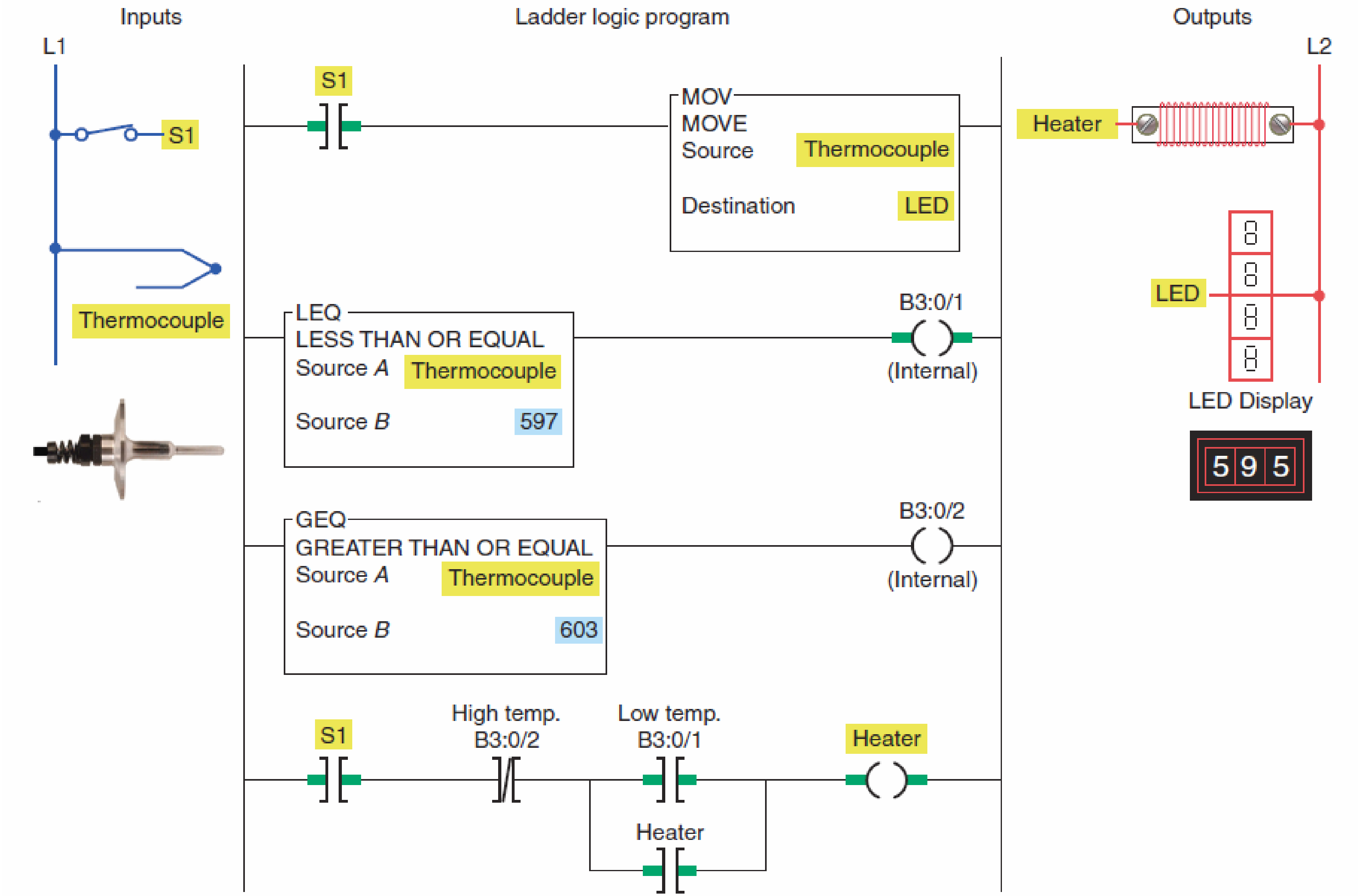
A closed-loop system utilizes feedback to measure the actual system operating parameter being controlled.

PLC closed-loop control system.

Adjustments are made continuously by the PLC until the difference between the desired and actual output is as small as is practical.

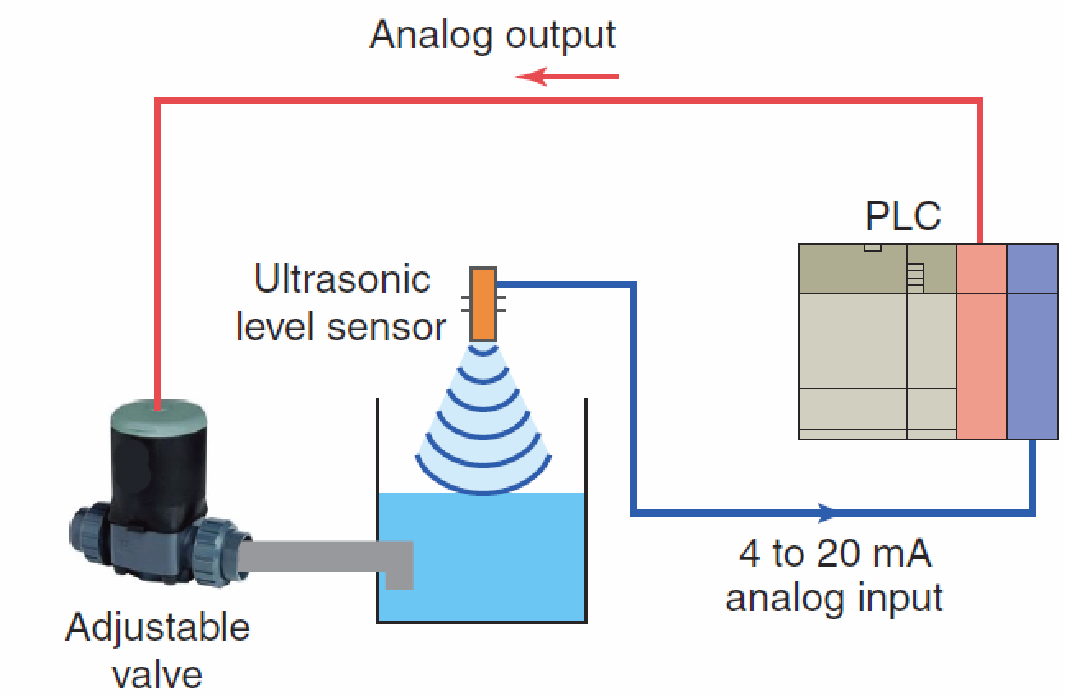


With on/off PLC control the output is either on or off.



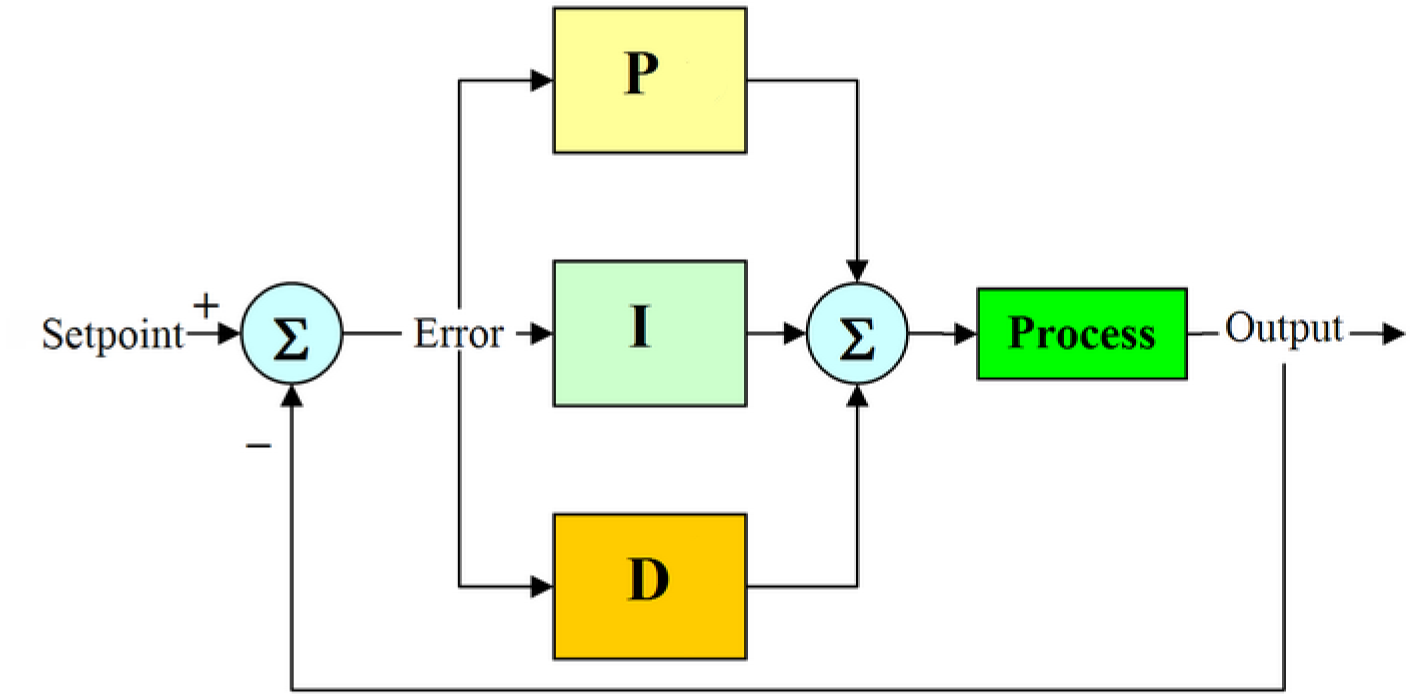
Proportional controls **are designed to eliminate the hunting or cycling associated with on/off control.**

Proportional control allows the control element to take intermediate positions between on and off.



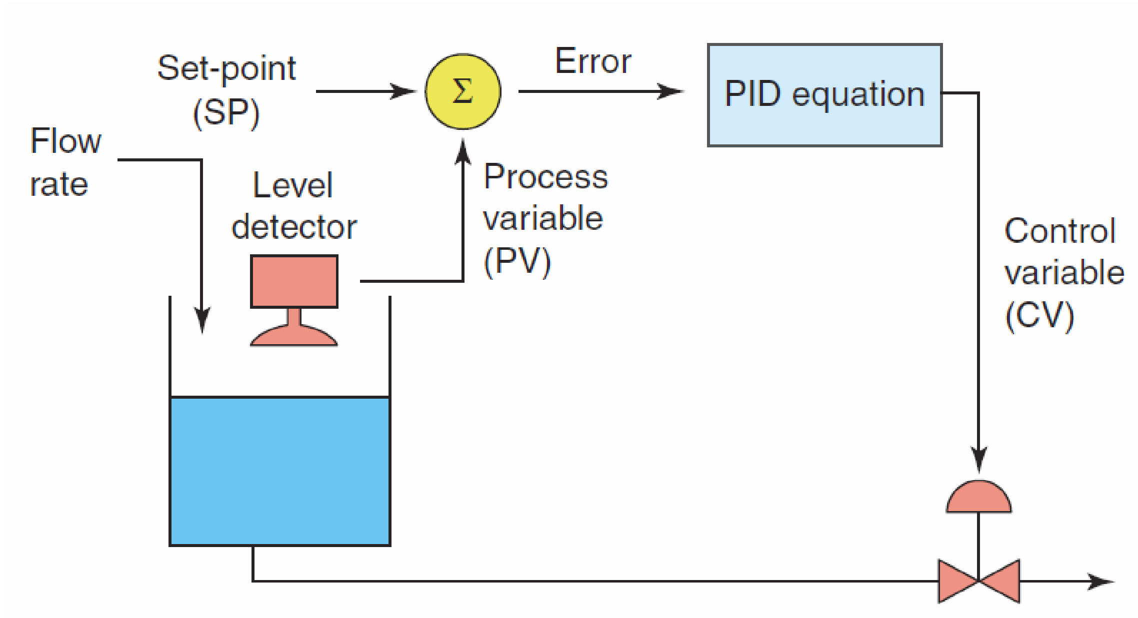
The PLC analog output module controls the amount of fluid flow by adjusting the percentage of valve opening.

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



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

PID control loop



➢Operating information from the machine is called the process variable or feedback.

➢Input from the operator that tells the controller the desired operating point is called the set-point .

➢The difference between the set-point and the process variable is called the error.

Programmable controllers are either equipped with *PID I/O modules* that produce PID control or have *sufficient mathematical functions* of their own to allow PID control to be carried out.