

PLC Programming



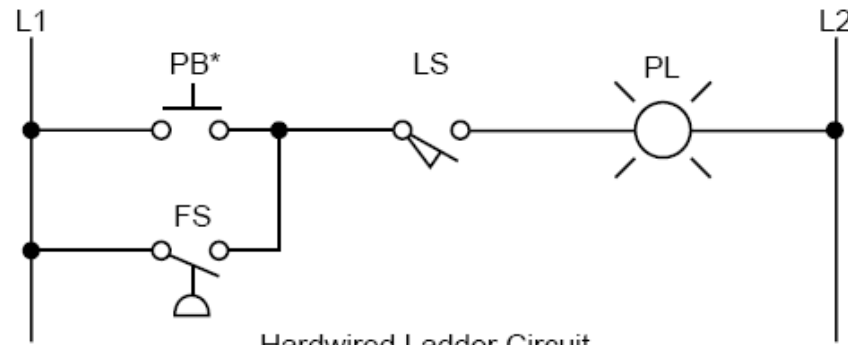
DR. TAREK A. TUTUNJI

PLC Programming

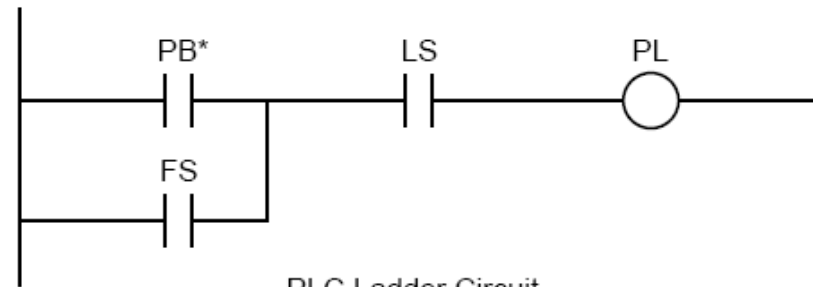


- As PLCs developed and expanded, programming languages have developed with them.
- The three types of programming languages used in PLCs are:
 - ladder
 - Boolean
 - Grafcet
- In this chapter, we will concentrate on ladder programming

Ladder Language



Hardwired Ladder Circuit



PLC Ladder Circuit

*Note: The PLC will know the elements PB, LS, FS, and PL by their addresses once the address assignment has been performed.

Figure 9-1. Hardwired logic circuit and its PLC ladder language implementation.

Boolean Language

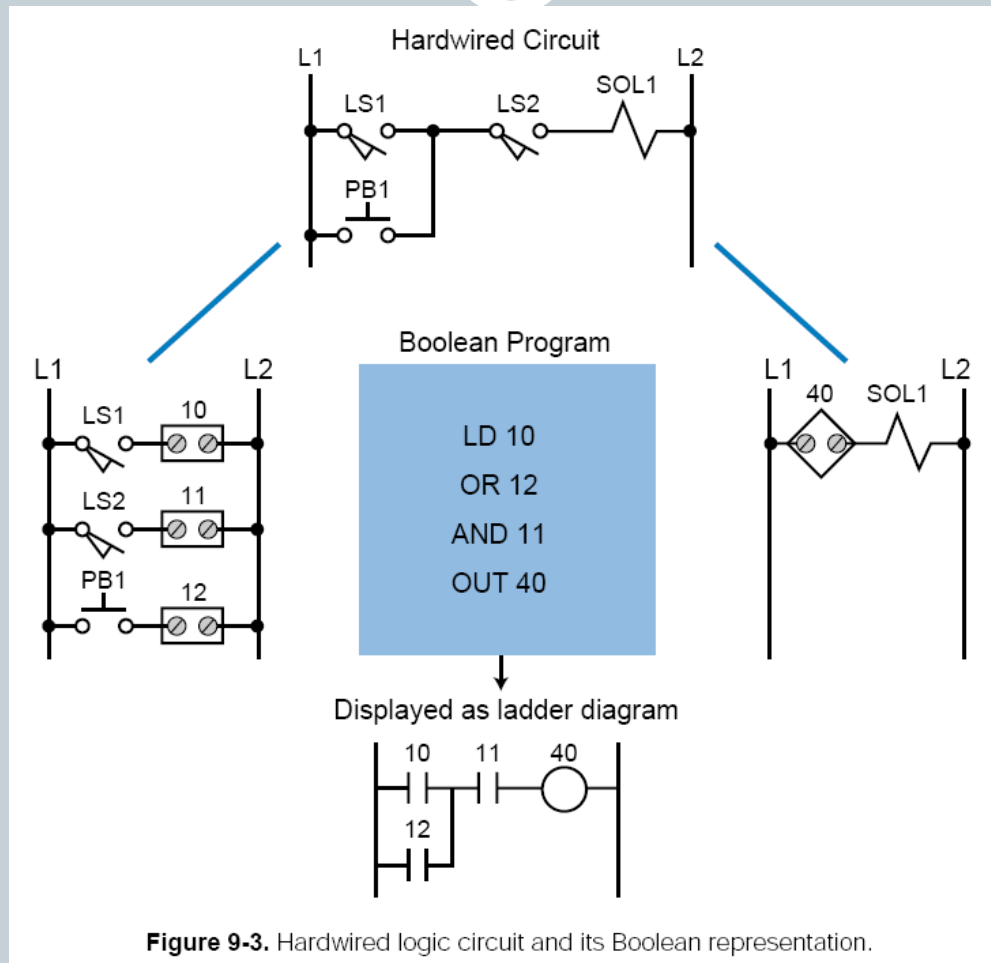


Figure 9-3. Hardwired logic circuit and its Boolean representation.

Grafcet

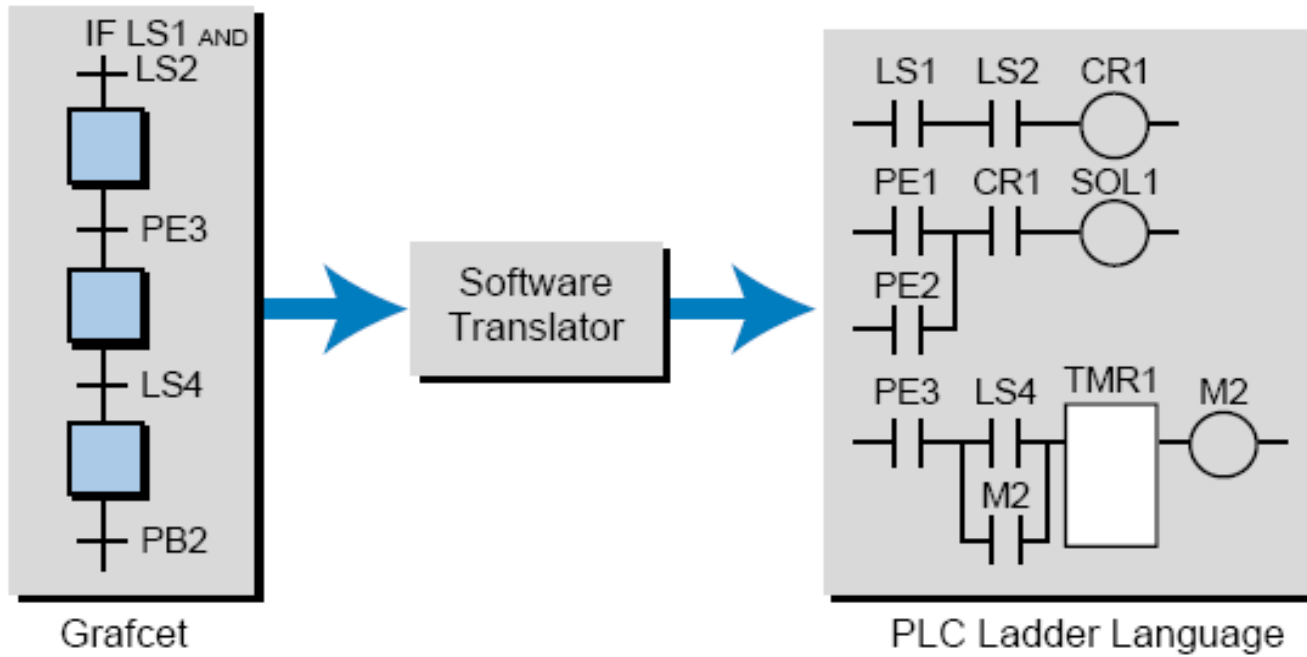
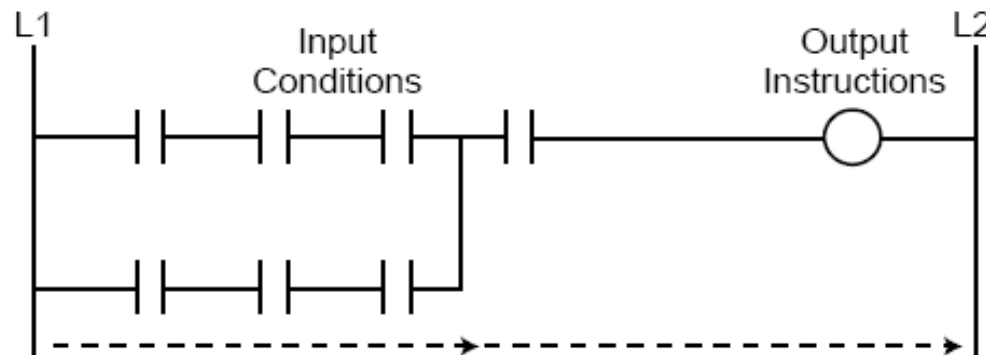


Figure 9-5. Grafcet translation.

Ladder Diagram Format



- The main functions of a ladder diagram program are to control outputs and perform functional operations based on input conditions.
- Ladder diagrams use rungs to accomplish this control.
- A rung consists of a set of input conditions (represented by contact instructions) and an output instruction at the end of the rung (represented by a coil symbol).

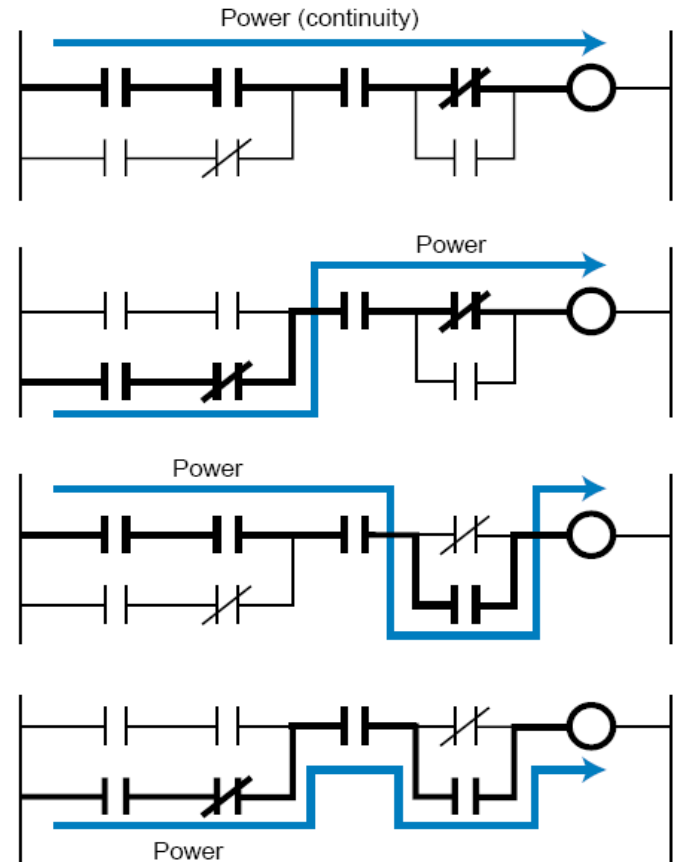


A continuous path is required for logic continuity

Ladder Diagram Format



- A ladder rung is TRUE (i.e., energizing an output or functional instruction block) when it has logic continuity.
- Logic continuity exists when power flows through the rung from left to right.



Ladder Diagram Format



- When a ladder diagram contains a functional block, contact instructions are used to represent the input conditions that drive (or *enable*) the block's logic.

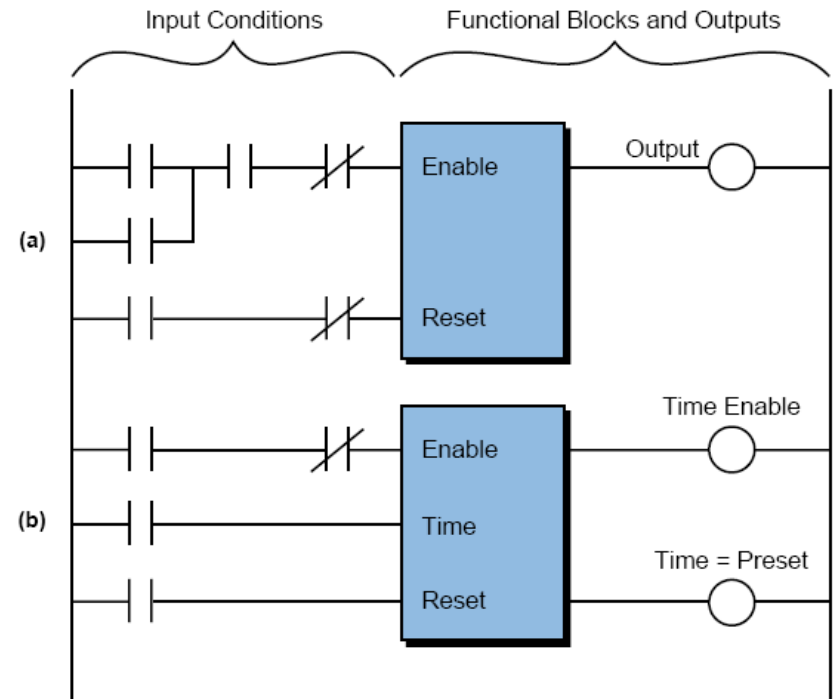


Figure 9-9. Functional block instructions with (a) one enable line and one output and (b) one enable line, a start timing command, and two outputs.

Ladder Instructions



1. Ladder relay
2. Timing
3. Counting
4. Program/flow control
5. Arithmetic
6. Data manipulation
7. Data transfer
8. Special function (sequencers)
9. Network communication

Relay Ladder Instructions



- **Ladder relay instructions** are the most basic instructions in the ladder diagram instruction set.
- These instructions represent the ON/OFF status of connected inputs and outputs.
- Ladder relay instructions use two types of symbols:
 - **Contacts** represent the input conditions that must be evaluated in a given rung to determine the control of the output.
 - **Coils** represent a rung's outputs

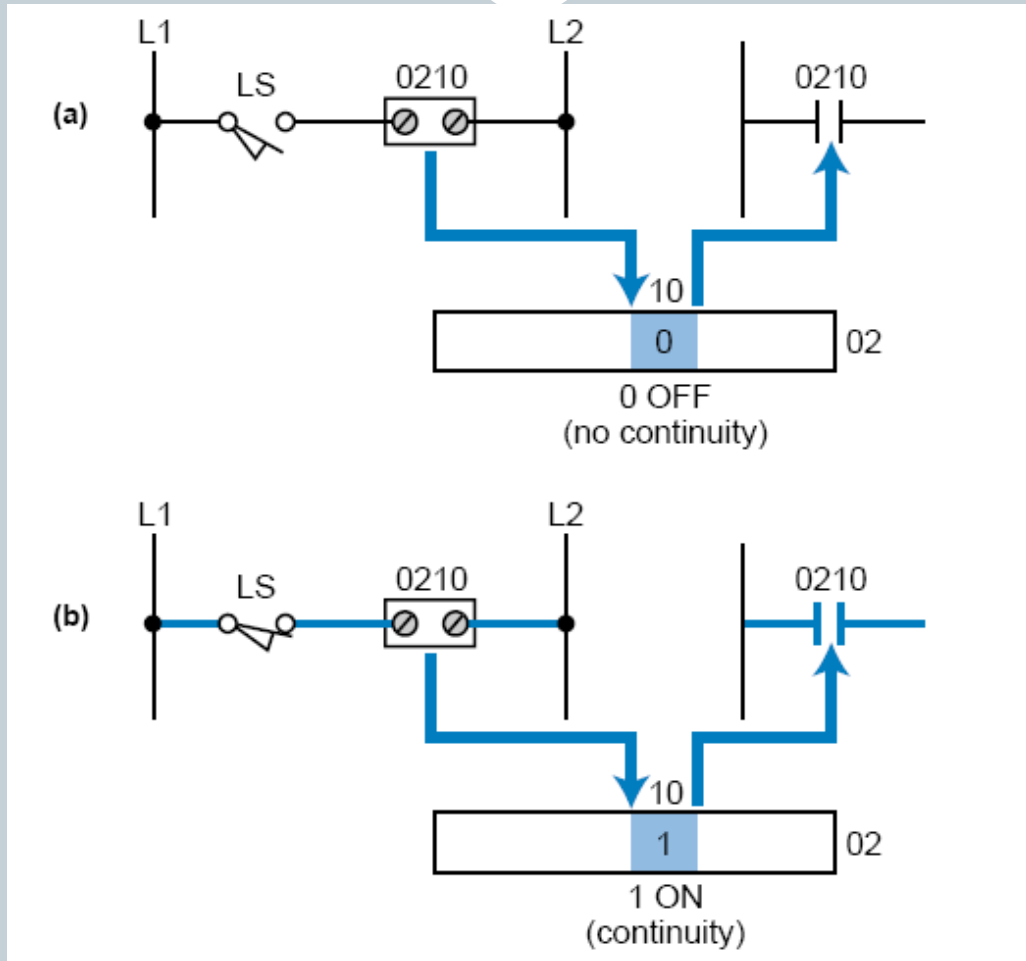
Ladder Relay Instructions



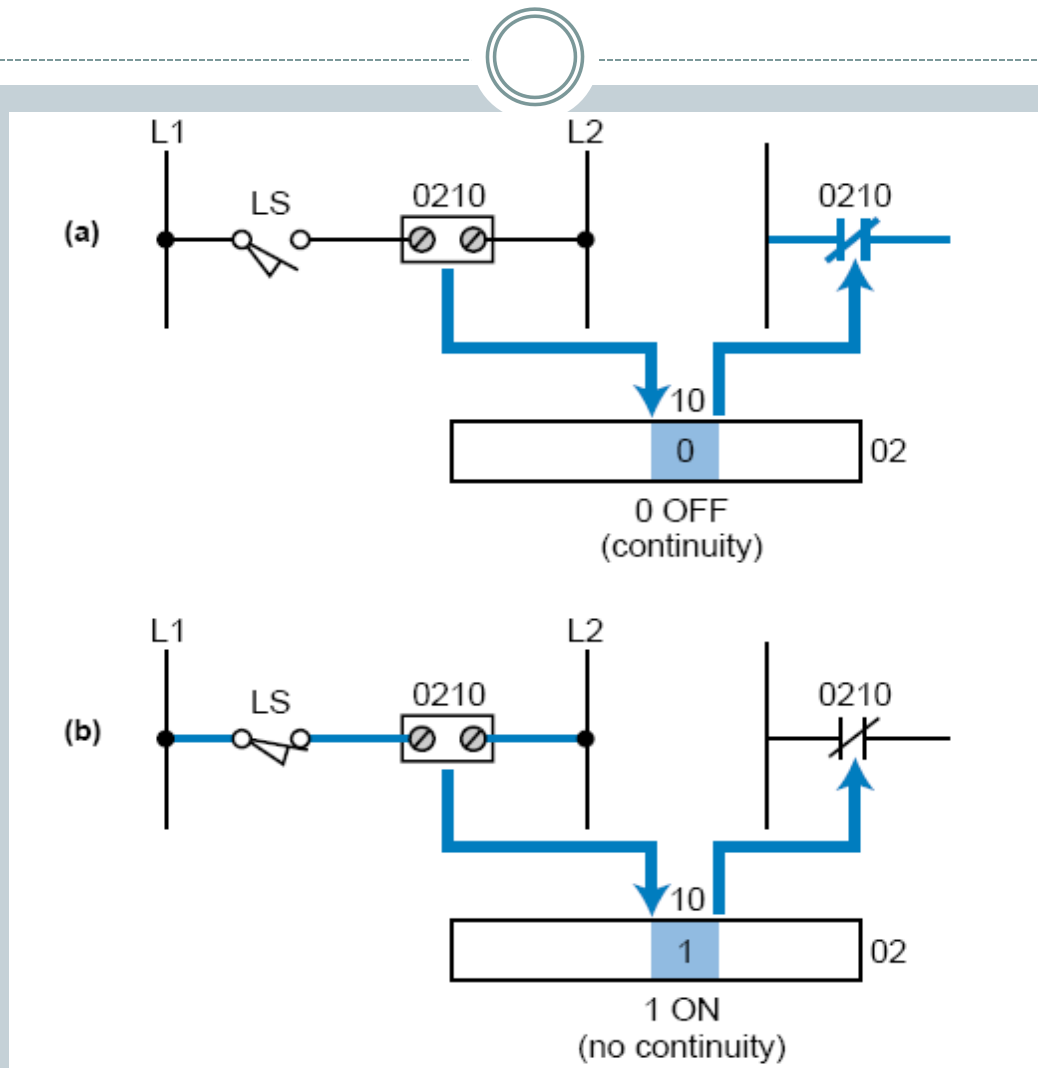
Ladder Relay Instructions <i>(Purpose: To provide hardwired relay capabilities in a PLC)</i>		
Instruction	Symbol	Function
Examine-ON/Normally Open		Tests for an ON condition in a reference address
Examine-OFF/Normally Closed		Tests for an OFF condition in a reference address
Output Coil		Turns real or internal outputs ON when logic is 1
NOT Output Coil		Turns real or internal outputs OFF when logic is 1
Latch Output Coil		Keeps an output ON once it is energized
Unlatch Output Coil		Resets a latched output
One-Shot Output		Energizes an output for one scan or less
Transitional Contact		Closes for one scan when its trigger contact makes a positive transition

Table 9-2. Ladder relay instructions.

Examine ON / Normally Open



Examine OFF / Normally Closed



Output Coil

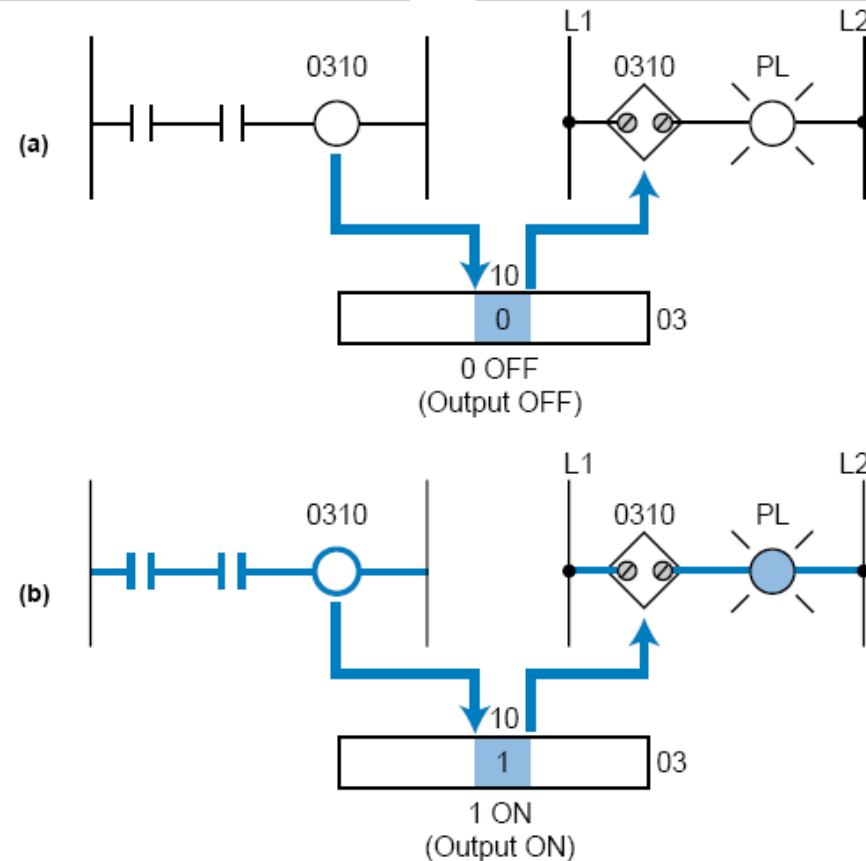


Figure 9-19. (a) An output coil instruction with a logic 0 reference address and (b) an output coil instruction with a logic 1 reference address.

Output Coil

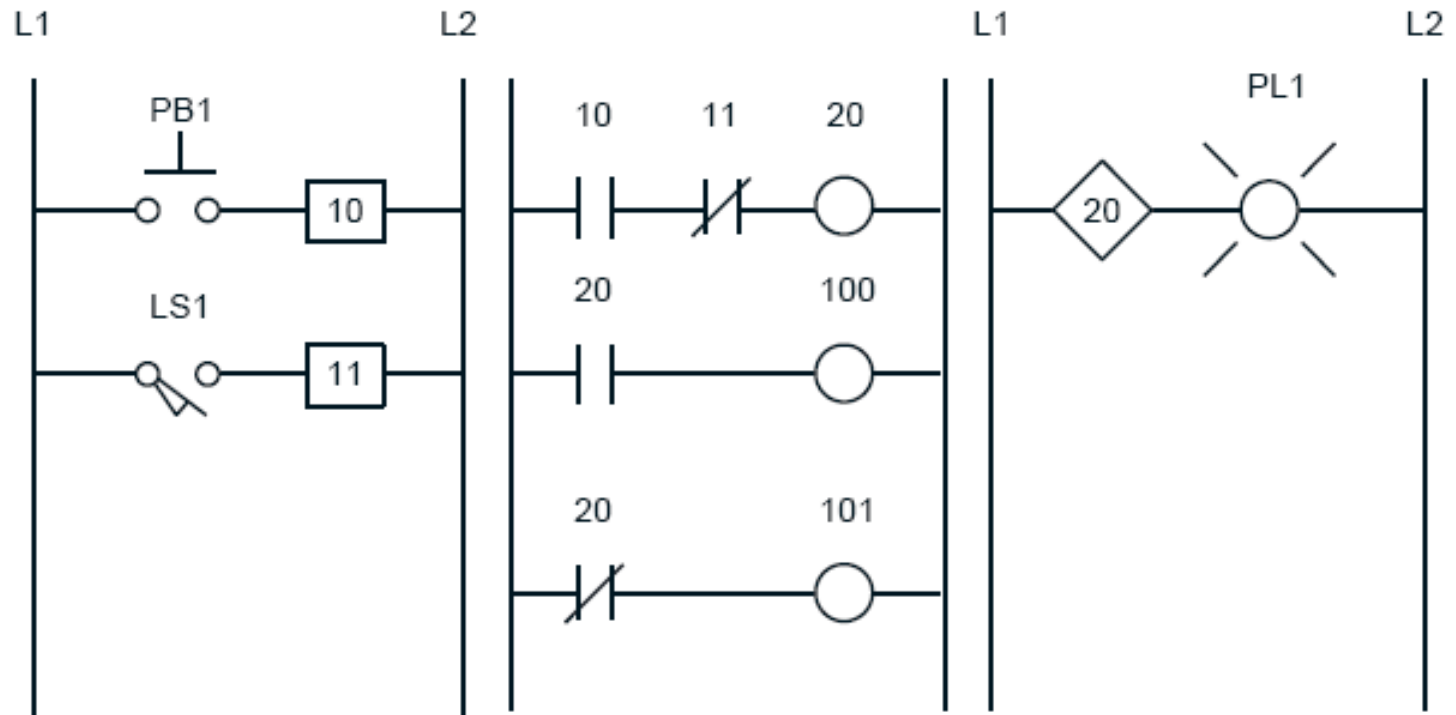


Figure 9-20. Normally open and normally closed contacts driving real and internal output coils.

Not Output Coil

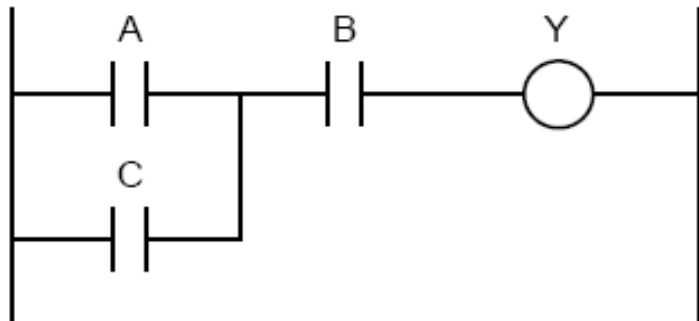


Figure 9-21. Ladder rung for Example 9-3.

$$Y = (A + C) \cdot B$$

$$\begin{aligned} \bar{Y} &= \overline{(A + C) \cdot B} \\ &= \overline{(A + C)} + \bar{B} \\ &= (\bar{A} \cdot \bar{C}) + \bar{B} \end{aligned}$$

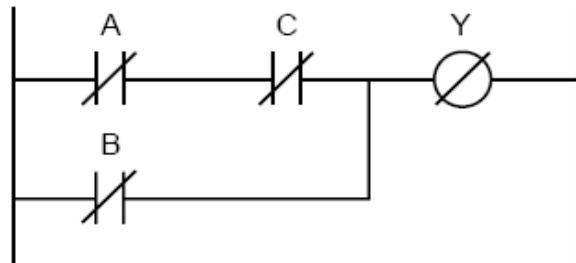


Figure 9-22. Implementation of Figure 9-21 using a NOT coil.

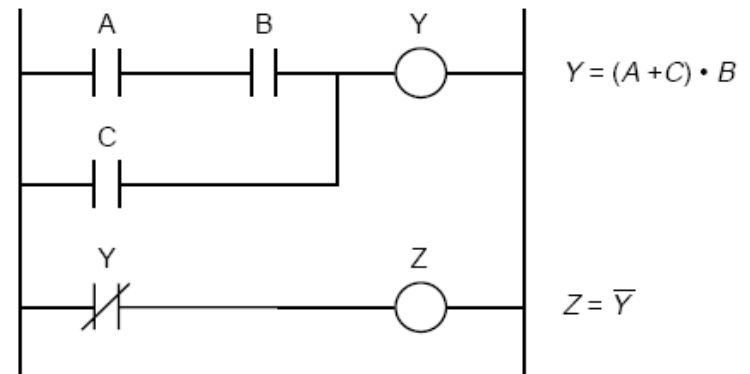


Figure 9-23. Implementation of the NOT Y logic without a NOT coil.

Latch and Unlatch Output Coil



- A *latch coil* instruction causes an output to remain energized even if the status of the contacts that caused the output to energize changes.
- The latched output will remain ON until it is unlatched by an unlatch output instruction.
- An *unlatch coil* instruction resets a latched output with the same reference address.

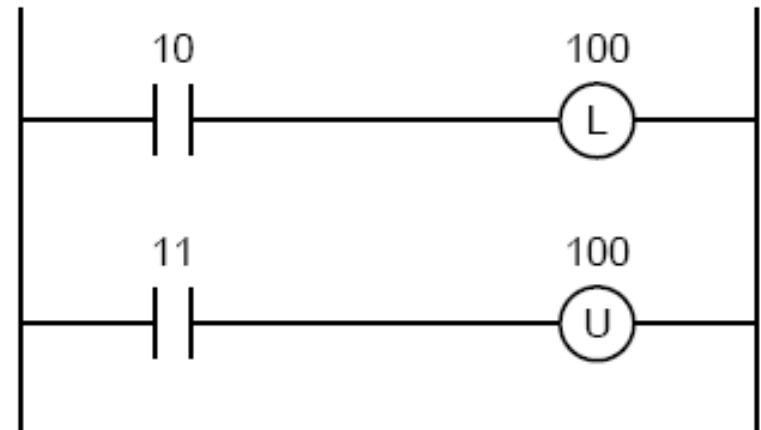
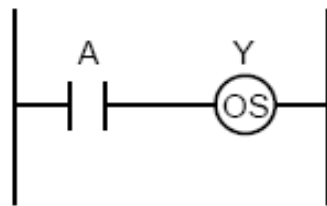


Figure 9-24. Latch and unlatch coil instructions.

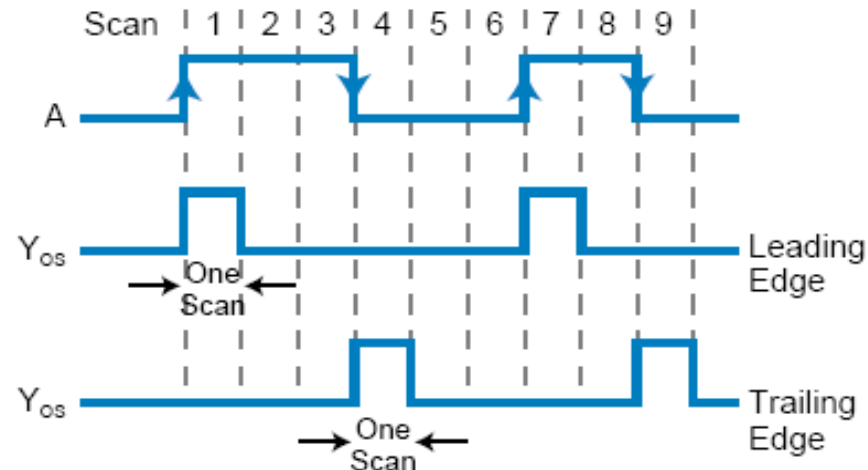
One-Shot Output



- A *one-shot output* instruction operates in a manner similar to an output coil instruction—if the ladder rung has continuity, the one-shot output will be energized (ON). However, the length of time that a one-shot output is ON is one scan



(a)



(b)

Figure 9-26. (a) A one-shot output instruction and (b) its timing diagram.

Transitional Contact

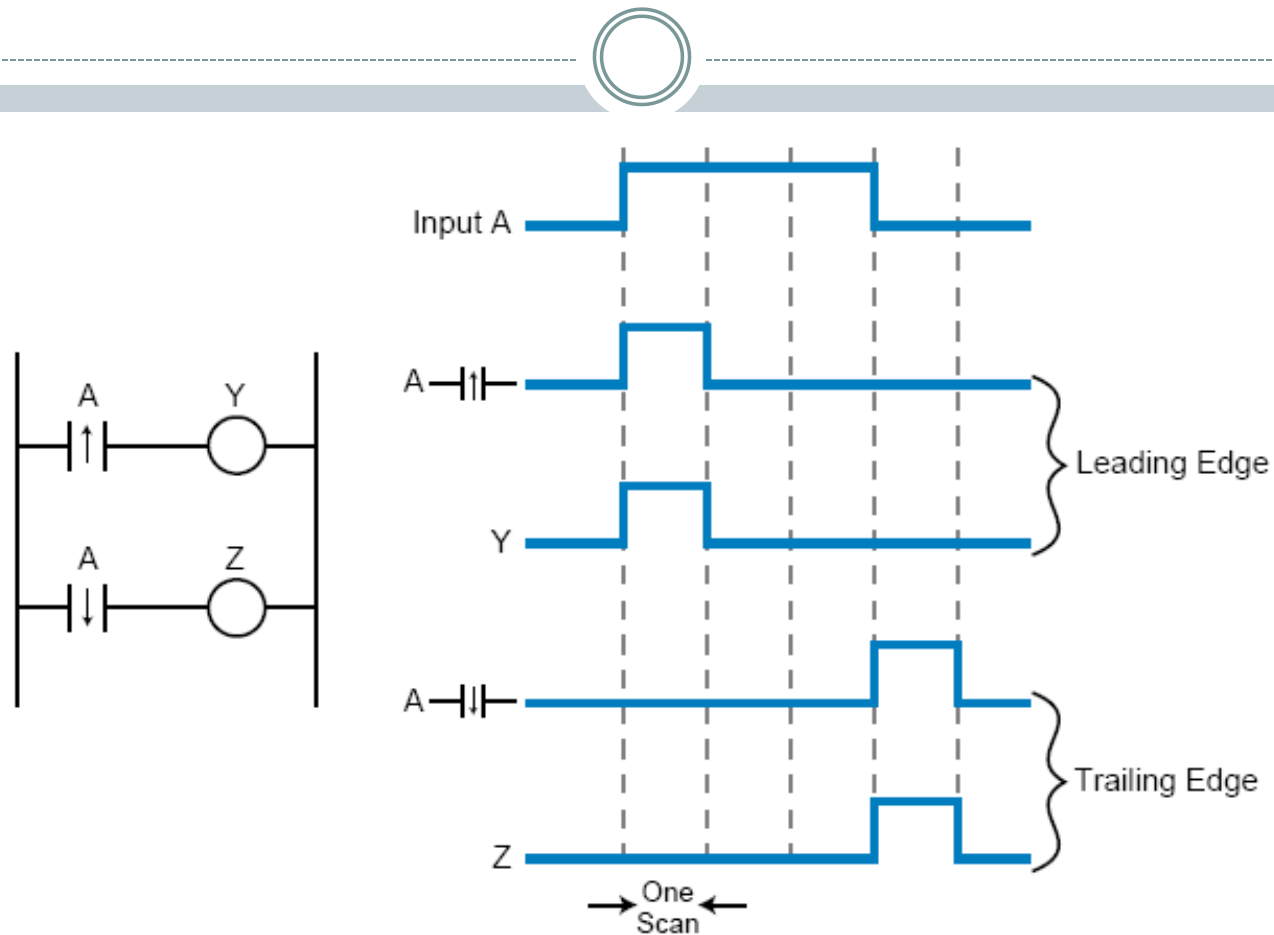


Figure 9-27. Leading- and trailing-edge transitional contact instructions and their timing diagrams.

Ladder Scan Evaluation



- The processor starts solving a ladder program after it has read the status of all inputs and stored this information in the input table.
- The solution starts at the top of the ladder program, beginning with the first rung and proceeding one rung at a time.
- As the processor solves the control program, it examines the reference address of each programmed instruction, so that it can assess logic continuity for the rung being solved.
- Even if the output conditions in the rung being solved affect previous rungs, the processor will not return to the previous rung to resolve it.

Scan Evaluation

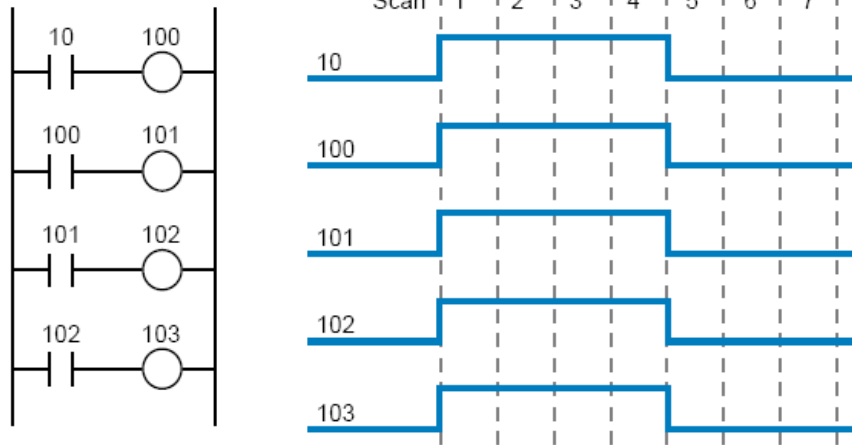


Figure 9-28. Ladder rung where all outputs turn ON in the same scan.

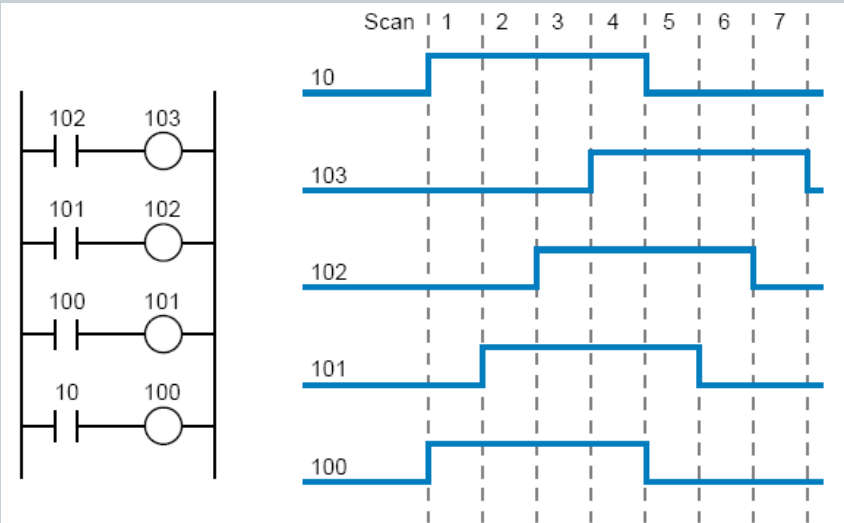


Figure 9-29. Ladder rung where the outputs turn ON in different scans.

Normally Open / Normally Closed

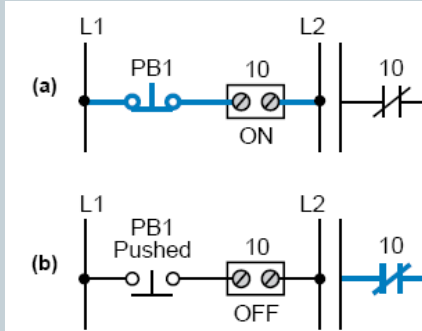


Figure 9-34. Power flow through the circuit shown in Figure 9-32 with (a) PB1 not pushed and (b) PB1 pushed.

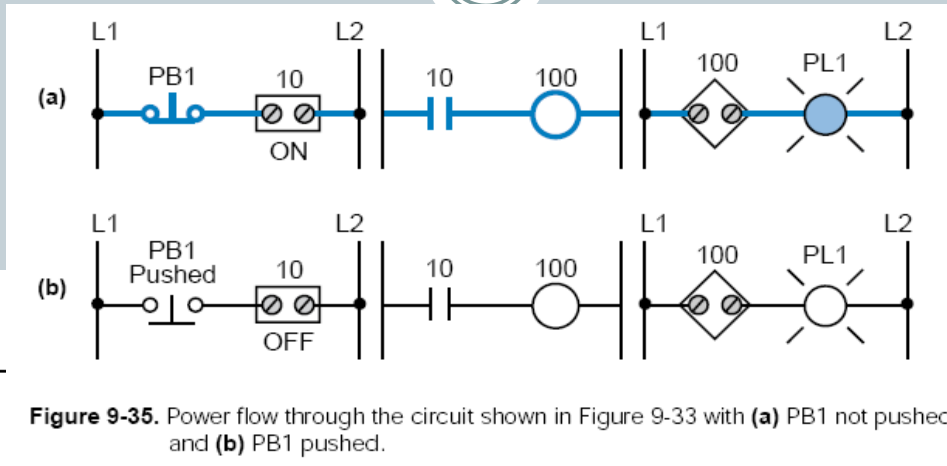
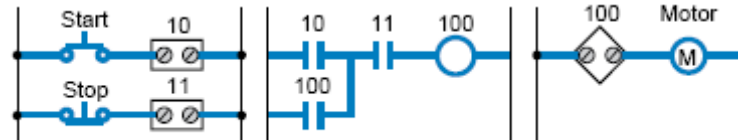
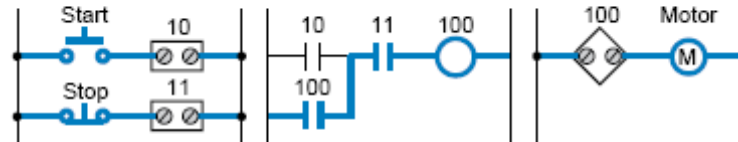


Figure 9-35. Power flow through the circuit shown in Figure 9-33 with (a) PB1 not pushed and (b) PB1 pushed.

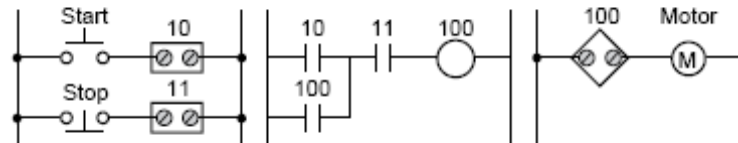
Normally closed stop push button programmed as normally open



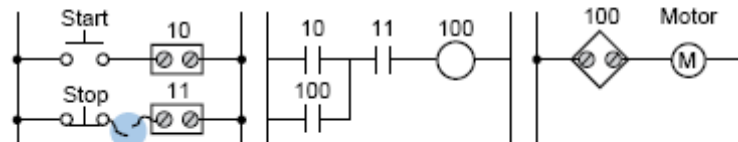
- (a) The normally closed stop push button is programmed as normally open. Contact 100 is used as an interlock with the start push button after the start is pushed. When the start push button is pressed, the motor turns ON.



- (b) After the start push button is pressed and released, the motor remains ON.

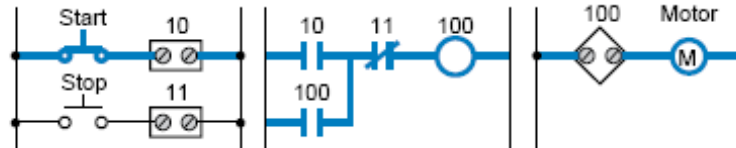


- (c) If the stop push button is pressed when the motor is ON, the motor will turn OFF.

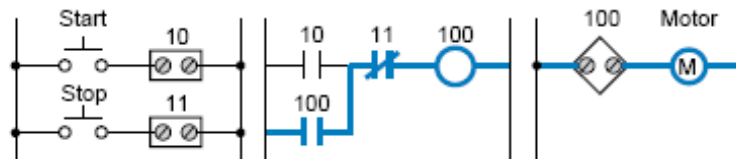


- (d) If the stop push button connection breaks when the motor is ON, the motor will turn OFF.

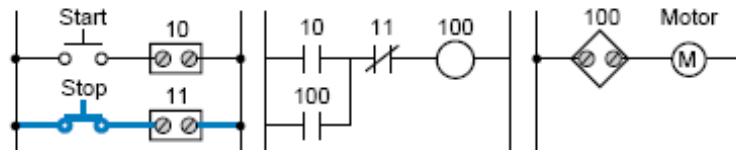
Normally open stop push button programmed as normally closed



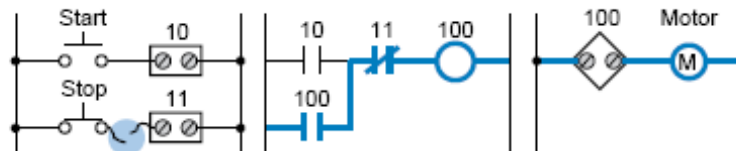
- (a) The normally open stop push button is programmed as normally closed. When the start push button is pressed, the motor turns ON.



- (b) After the start push button is pressed and released, the motor remains ON.



- (c) If the stop push button is pressed when the motor is ON, the motor will turn OFF.



- (d) If the stop push button connection breaks when the motor is ON, pressing the stop push button will not turn the motor OFF. This is a dangerous situation.

Timers and Counters



- PLC timers and counters are internal instructions that provide the same functions as hardware timers and counters.
- They activate or deactivate a device after a time interval has expired or a count has reached a preset value.
- Timer and counter instructions are generally considered internal outputs.

Timers and Counters



- **Timer instructions** may have one or more *time bases* (TB) which they use to time an event.
- The time base is the resolution, or accuracy, of the timer.

Required Time	Number of Ticks	Time Base (secs)
10 sec	10	1.00
10 sec	100	0.10
10 sec	1000	0.01

Note: Required time = (# of ticks)(Time base)

Table 9-3. Time bases.

Timers and Counters



- Timers are used in applications to add a specific amount of delay to an output in the program.
- **Counter instructions** are used to count events, such as parts passing on a conveyor belt.
- Counters, along with timers, must have two values:
 - Preset value is the target number of ticks or counting numbers that must be achieved before the timer or counter turns its output ON.
 - Accumulated value is the current number of ticks (timer) or counts (counter) that have elapsed during the timer or counter operation.

Timers and Counters



- Suppose a three AC cycle (60 Hz) is needed
- The estimated delay of the three cycles is
 - $3/60 = 50$ msec
- The PLC program can use a time base of 0.01 second and count 5 ticks

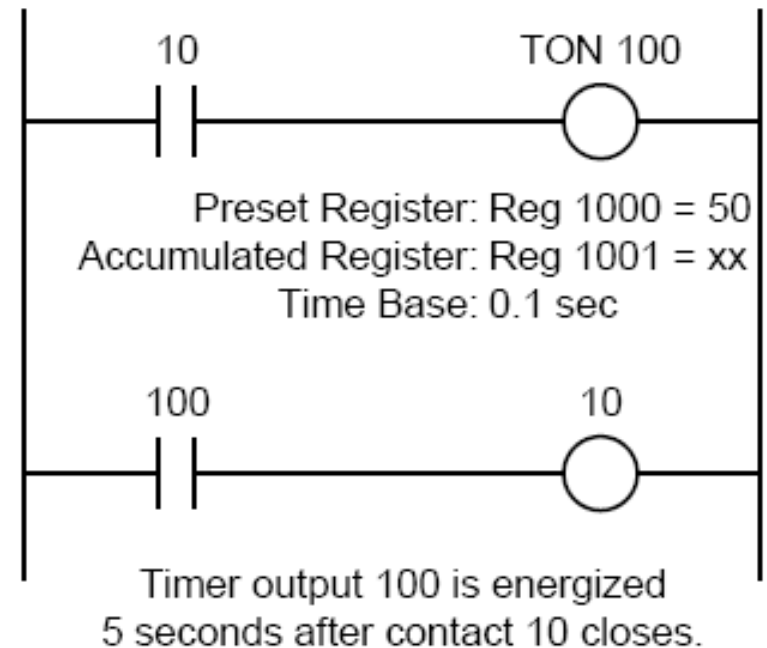


Figure 9-41. Example relay ladder circuit.

Timers Instructions



Timer Instructions <i>(Purpose: To provide hardware timer capabilities in a PLC)</i>		
Instruction	Symbol	Function
ON-Delay Energize Timer	TON 	Energizes an output after a set time period when logic 1 exists
ON-Delay De-energize Timer	TON 	De-energizes an output after a set time period when logic 1 exists
OFF-Delay Energize Timer	TOF 	Energizes an output after a set time period when logic 0 exists
OFF-Delay De-energize Timer	TOF 	De-energizes an output after a set time period when logic 0 exists
Retentive ON-Delay Timer	RTO 	Energizes an output after a set time period when logic 1 exists and then retains the accumulated value
Retentive Timer Reset	RTR 	Resets the accumulated value of a retentive timer

Table 9-4. Timer instructions.

Timer Instructions

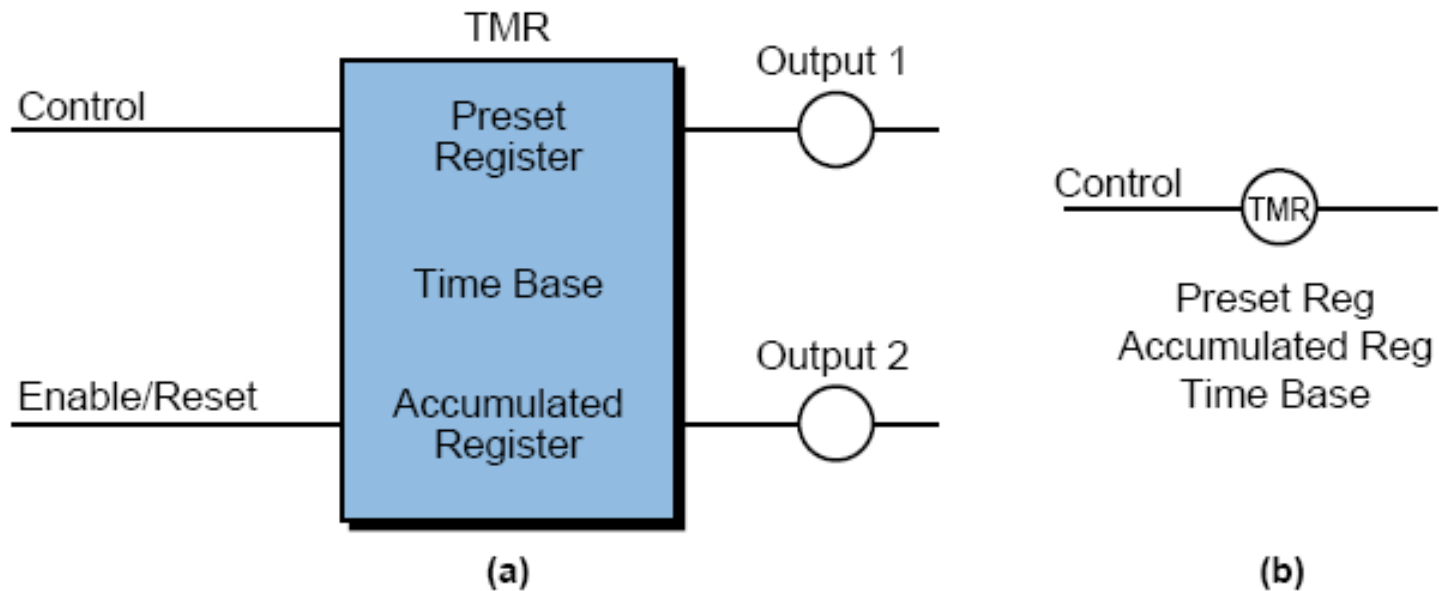


Figure 9-42. (a) Block format and (b) ladder format timer instructions.

On-Delay Energize Timer



- An *ON-delay energize timer* (TON) output instruction either provides time-delayed action or measures the duration for which some event occurs.
- Once the rung has continuity, the timer begins counting time-based intervals (ticks) and counts down until the accumulated time equals the preset time.
- When these two values are equal, the timer energizes the output and closes the timed-out contact associated with the output.
- The timed contact can be used throughout the program as either a normally open or normally closed contact.
- If logic continuity is lost before the timer times out, the timer resets the accumulated register to zero.

On-Delay Timers

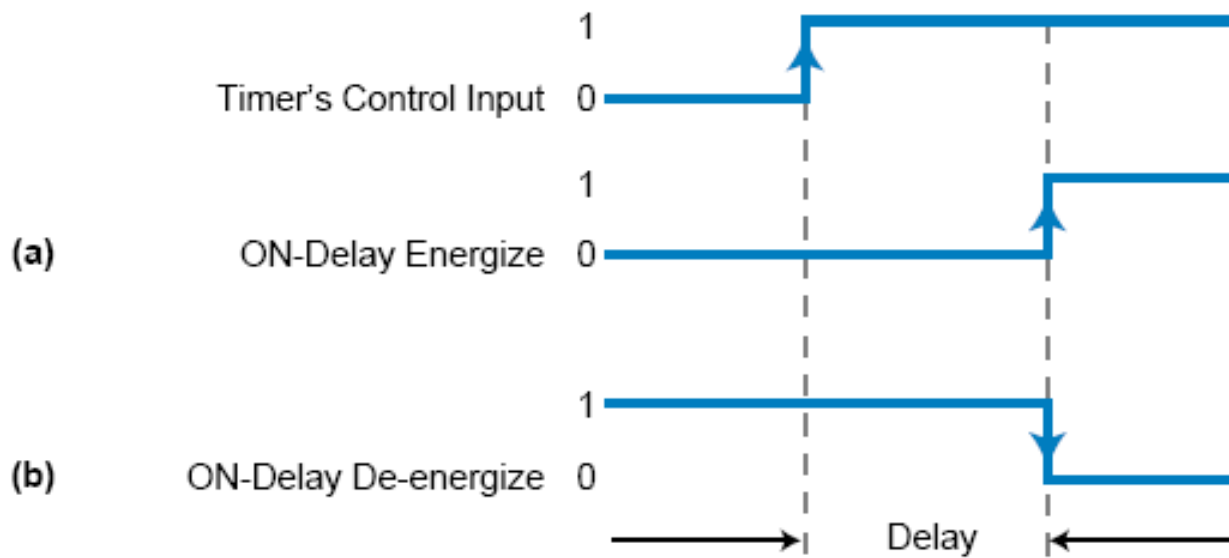


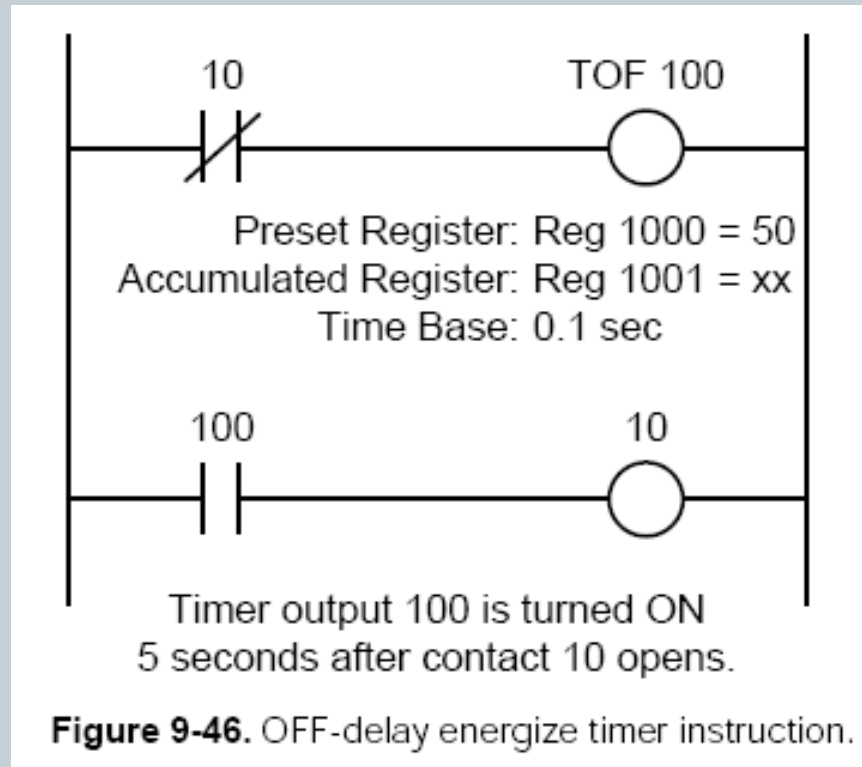
Figure 9-45. Timing diagram for (a) an ON-delay energize timer and (b) an ON-delay de-energize timer.

Off-Delay Energize Timer



- An *OFF-delay energize timer* (TOF) output instruction provides time-delayed action.
- If the control line rung does not have continuity, the timer begins counting time-based intervals until the accumulated time value equals the programmed preset value.
- When these values are equal, the timer energizes the output and closes the timed-out contact associated with the output.
- If logic continuity occurs before the timer times out, the accumulated value resets to zero.

Off-Delay Energize Timer



Off-Delay Timers

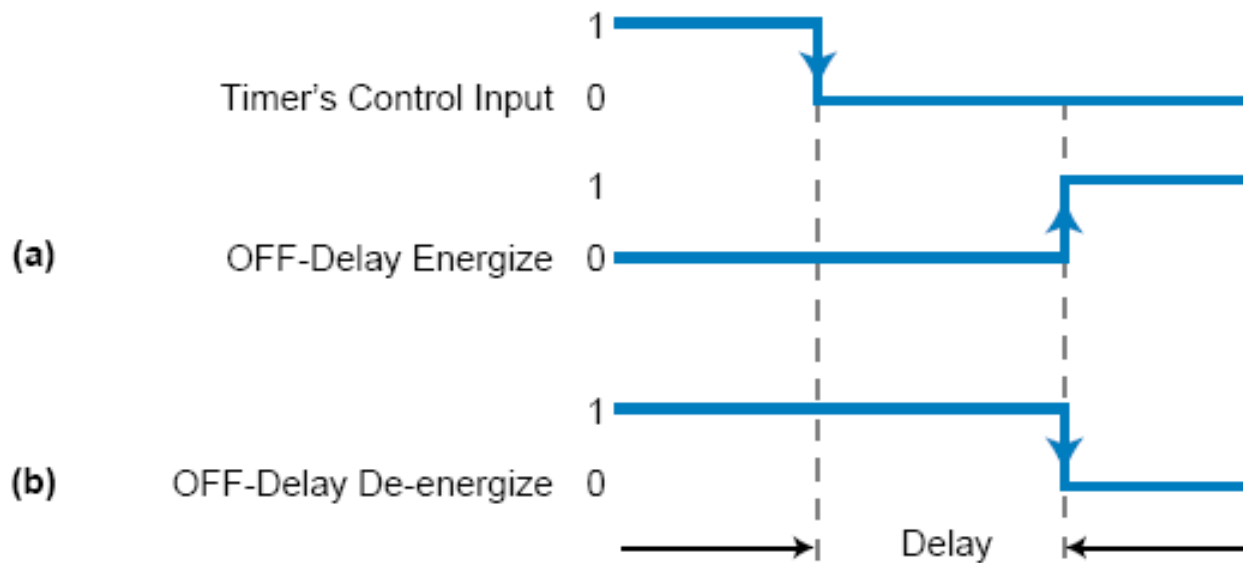


Figure 9-47. Timing diagram for **(a)** an OFF-delay energize timer and **(b)** an OFF-delay de-energize timer.

Retentive On-Delay Timer



- A *retentive ON-delay timer* (RTO) output instruction is used if the timer's accumulated value must be retained even if logic continuity or system power is lost.
- If any rung path has logic continuity, the timer begins counting time-based intervals until the accumulated time equals the preset value.
- The accumulated register retains this accumulated value, even if power or logic continuity is lost before the timer has timed out.

Retentive Timer Reset



- A *retentive timer reset* (RTR) output instruction is the only way to automatically reset the accumulated value of a retentive timer.
- If any rung path has logic continuity, then this instruction resets the accumulated value of its referenced retentive timer to zero.
- Note that the retentive timer reset address will be the same as the retentive timer output instruction it is resetting.

Counter Instructions

- There are two basic types of counters: those that can count up and those that can count down.

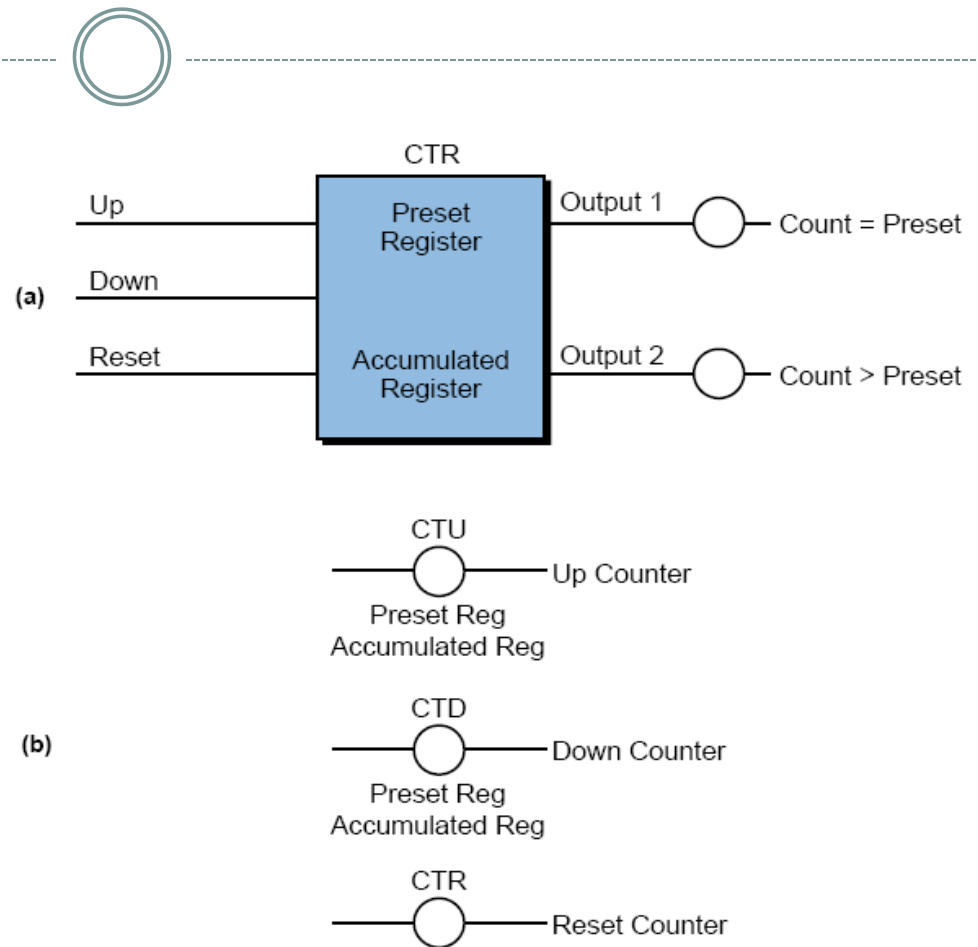


Figure 9-48. (a) Block format and (b) ladder format counter instructions.

Counter Instructions






Counter Instructions <i>(Purpose: To provide hardware counter capabilities in a PLC)</i>		
Instruction	Symbol	Function
Up Counter	CTU 	Increases the accumulated register value every time a referenced event occurs
Down Counter	CTD 	Decreases the accumulated register value every time a referenced event occurs
Counter Reset	CTR 	Resets the accumulated value of an up or down counter

Table 9-5. Counter instructions.

Up Counter



- An *up counter* (CTU) output instruction adds a count, in increments of one, every time its referenced event occurs.
- An up counter increases its accumulated value (the count value in its accumulated register) each time the up-count event makes an OFF-to-ON transition.
- When the accumulated value reaches the preset value, the counter turns ON the output, finishes the count, and closes the contact associated with the referenced output.

Down Counter



- A *down counter* (CTD) output instruction decreases the count value in its accumulated register by one every time a certain event occurs.
- Sometimes, a down counter is used in conjunction with an up counter to form an *up/down counter*, given that both counters have the same reference registers.
- For example, while an up counter counts the number of filled bottles that pass a certain point, a down counter with the same reference address can subtract one from the accumulated count value every time it senses an empty or improperly filled bottle

Counter Reset



- A *counter reset* (CTR) output instruction resets up counter and down counter accumulated values to zero.
- When programmed, a counter reset coil has the same reference address as the corresponding up/down counter coils.
- If the counter reset rung condition is TRUE, the reset instruction will clear the referenced address.

Counters

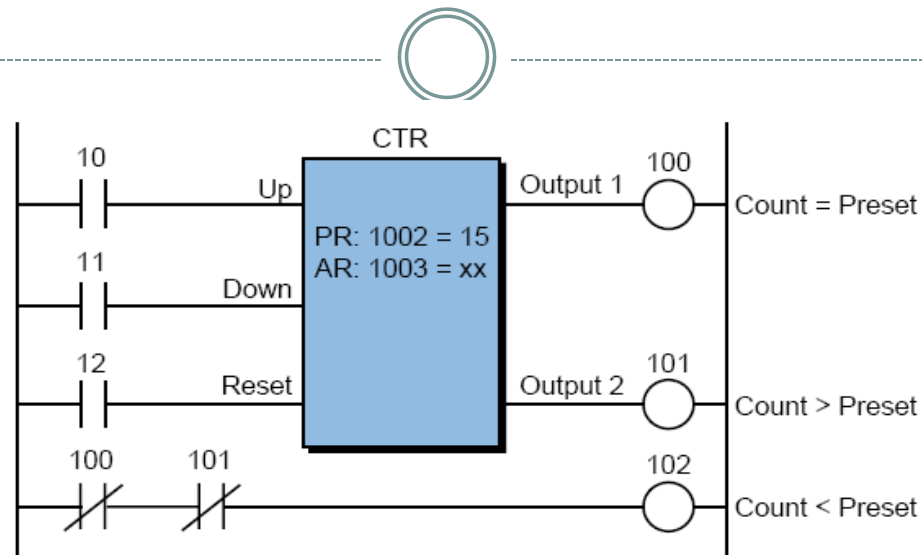


Figure 9-49. Counter function block with up, down, and reset counter instructions.

- The counter will count up when contact 10 closes, count down when contact 11 closes, and reset register 1003 to 0 when contact 12 closes.
- If the count is equal to 15 as a result of either an up or down count, output 100 will be ON.
- If contents of register 1003 are greater than 15, output 101 will be ON.
- Output 102 will be ON if the accumulated count value is less than 15

Counter Example



- A block counter instruction being used to count parts as detected by a photoelectric eye (PE) input.
- The preset value of counts is 500.
- Modify this circuit so that it will automatically reset every time the counter reaches 500.
- Also, add the instructions necessary to implement an output coil that indicates that the count has reached 500.

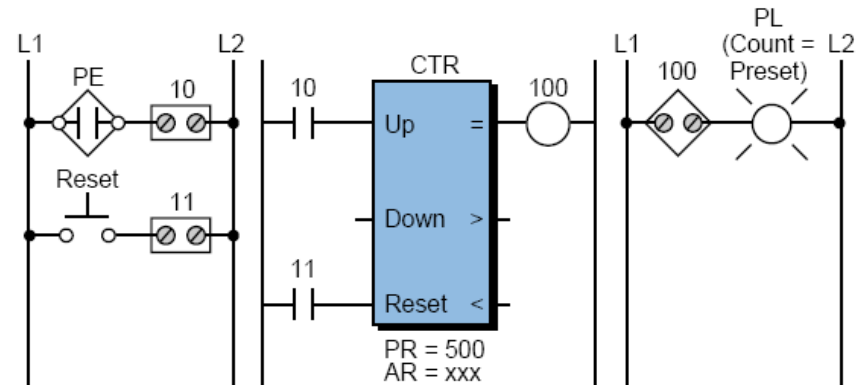


Figure 9-50. Functional block counter instruction.

Counter Example

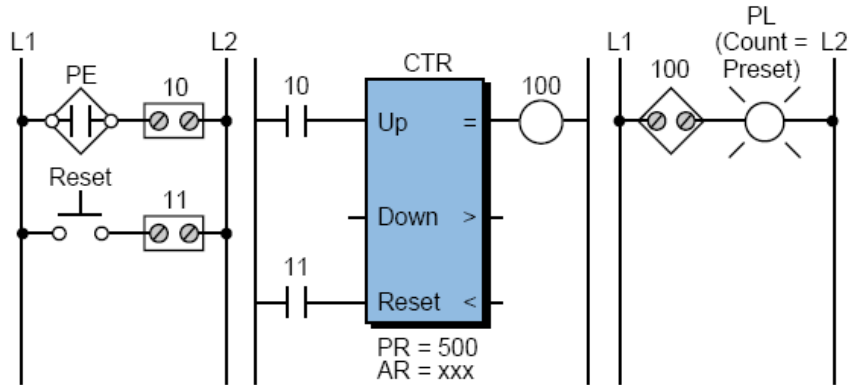


Figure 9-50. Functional block counter instruction.

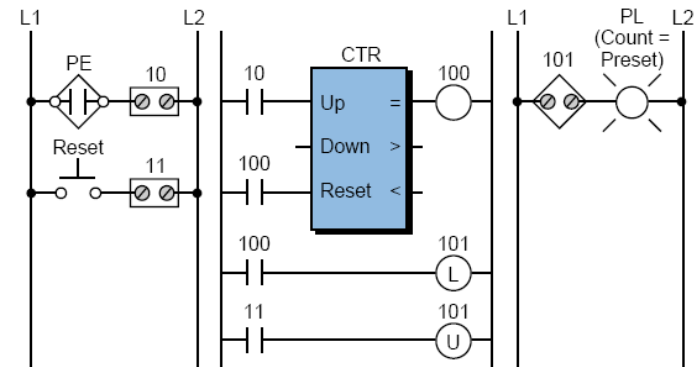


Figure 9-51. Automatically resetting counter.

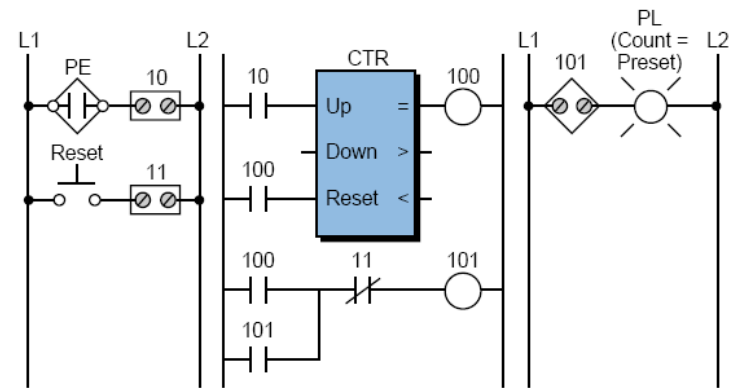


Figure 9-52. Solution to Example 9-7.

Program / Flow Control Instructions



- **Program/flow control instructions** direct the flow of operations, as well as the execution of instructions, within a ladder program.
- They perform these functions using branching and return instructions, which are executed when certain already programmed control logic conditions occur.

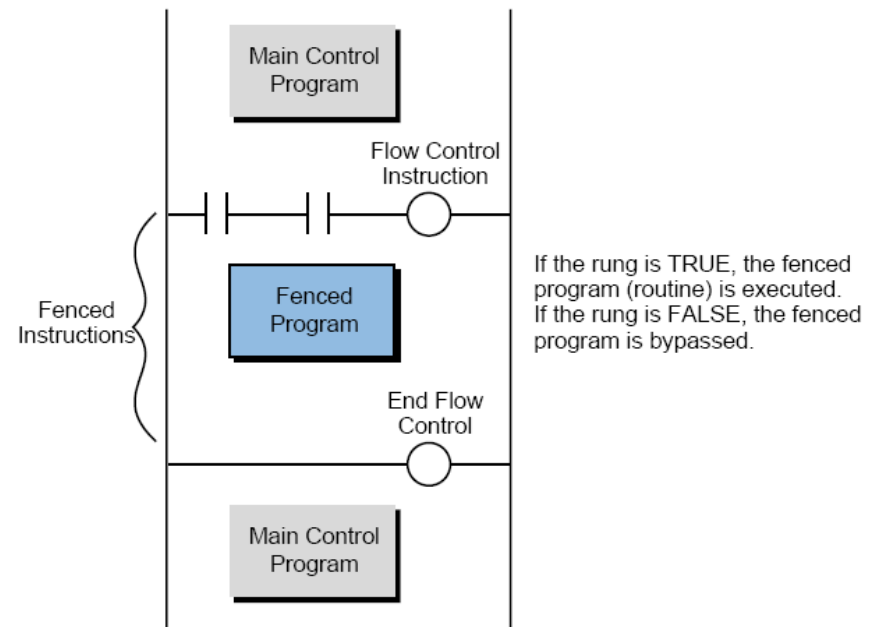


Figure 9-53. A fence created using a program/flow control instruction.

Program / Flow Control Instructions



Program/Flow Control Instructions <i>(Purpose: To direct the evaluation/execution of instructions in a ladder program)</i>		
Instruction	Symbol	Function
Master Control Relay		Activates/deactivates the execution of a group of ladder rungs
Zone Control Last State		Determines whether or not a group of ladder rungs will be evaluated
End		Identifies the last rung of an MCR or ZCL instruction
Jump To		Jumps to a specified rung in the program if certain conditions exist
Go To Subroutine		Goes to a specified subroutine in the program if certain conditions exist
Label		Identifies the target rung of a JMP or GOSUB instruction
Return		Terminates a ladder subroutine

Table 9-6. Program/flow control instructions.

- These Instructions are usually used in pairs

Master Control Relay



- Activates or deactivates the execution of a group or zone of ladder rungs.
- An MCR rung is used in conjunction with an END rung to fence a group of rungs

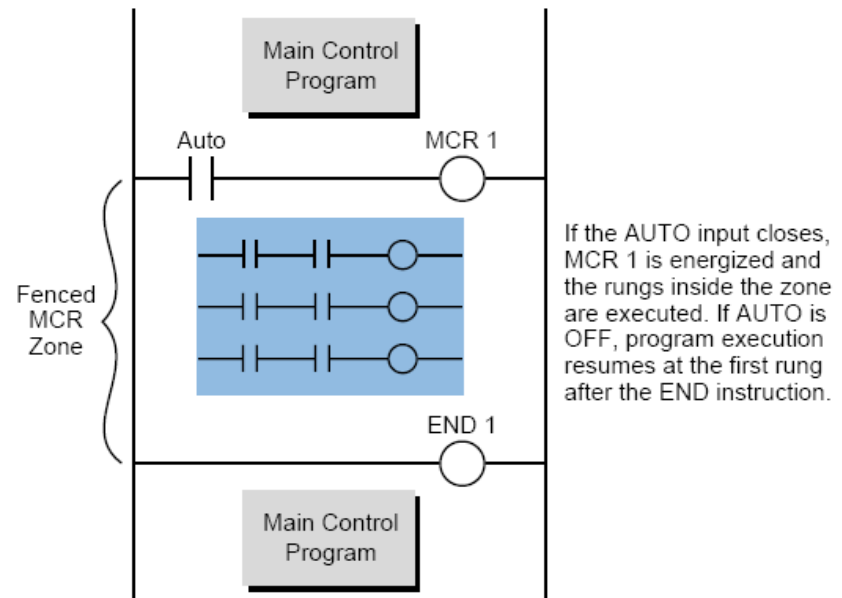


Figure 9-54. Example of an MCR instruction.

Jump To

- A *jump to* (JMP) instruction allows the control program sequence to be altered if certain conditions exist.
- If the rung condition is TRUE, the jump to coil reference address tells the processor to jump forward and execute the target rung.

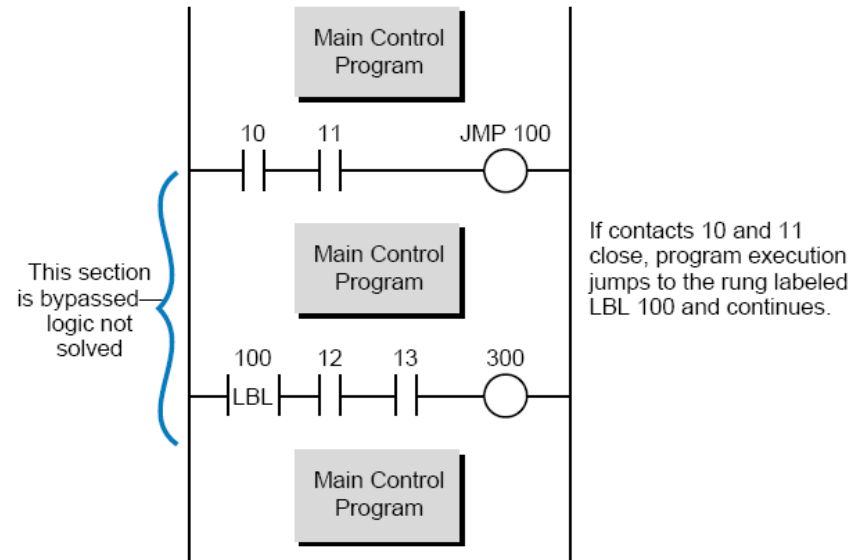


Figure 9-55. Example of a jump to instruction.

Go To Subroutine



- A *go to subroutine* (GOSUB) output instruction also allows normal program execution to be altered if certain conditions exist.
- A *label* (LBL) instruction identifies the ladder rung that is the target destination of a jump to or GOSUB instruction.
- A *return* (RET) instruction terminates a ladder subroutine and is programmed with no conditional inputs. When the control program encounters this instruction, it returns to the main program.

Go To Subroutine

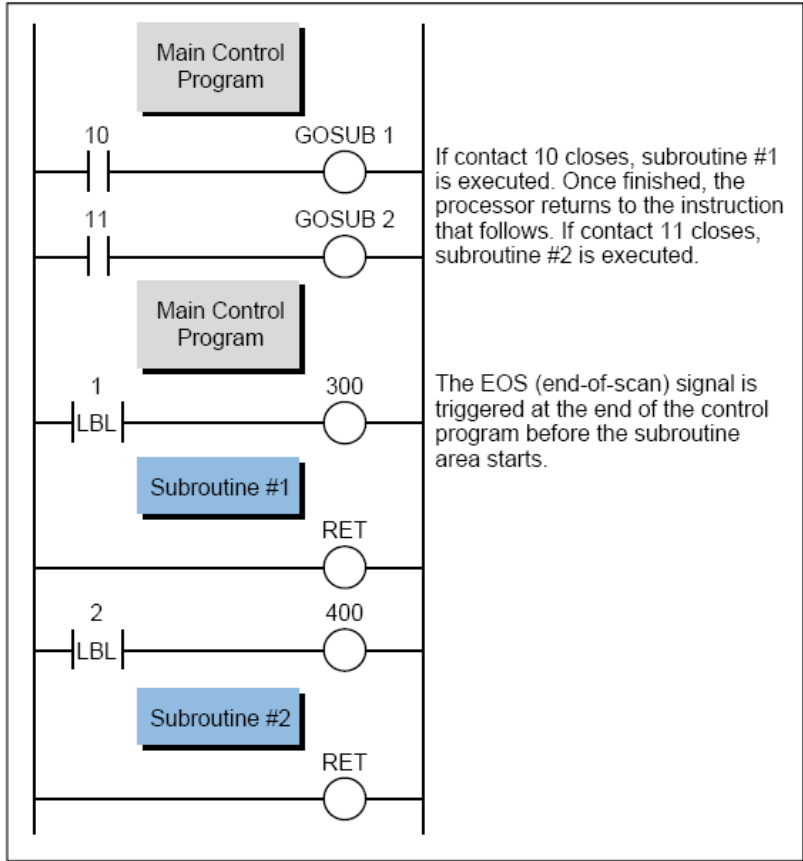


Figure 9-56. PLC with assigned subroutines at the end of the program.

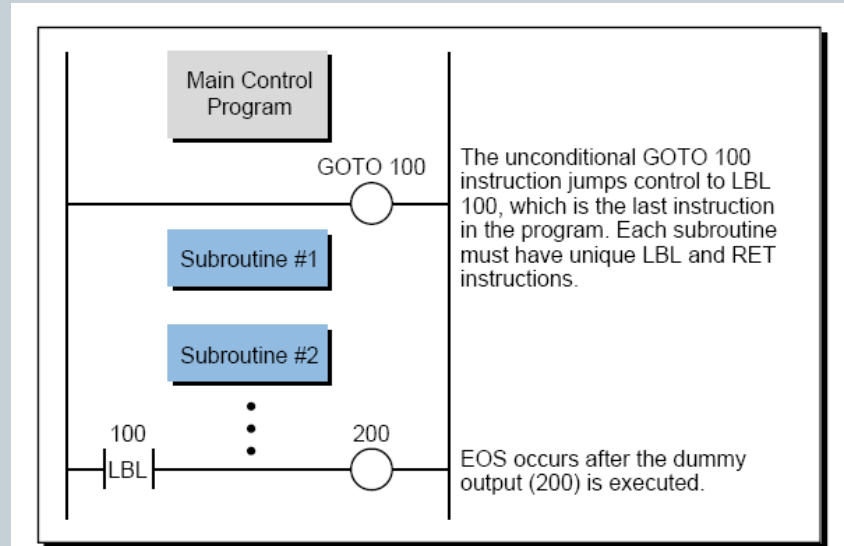


Figure 9-57. User-created subroutine area.

Arithmetic Instructions




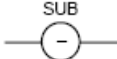

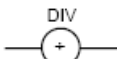
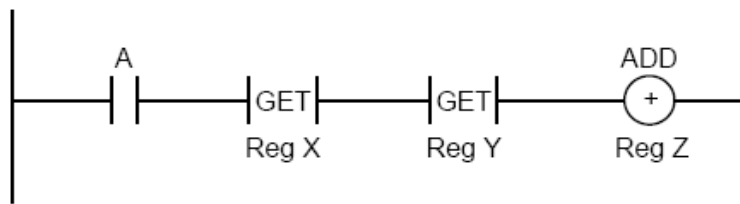
Arithmetic Instructions		
<i>(Purpose: To allow PLCs to perform mathematical functions with register data)</i>		
Instruction	Symbol	Function
Addition—Ladder		Adds the values stored in two registers
Addition—Block	ADD	Adds the values stored in two registers
Subtraction—Ladder		Subtracts the values stored in two registers
Subtraction—Block	SUB	Subtracts the values stored in two registers
Multiplication—Ladder		Multiplies the values stored in two registers
Multiplication—Block	MUL	Multiplies the values stored in two registers
Division—Ladder		Finds the quotient of the values in two registers
Division—Block	DIV	Finds the quotient of the values in two registers
Square Root—Block	SQR	Calculates the square root of a register value

Table 9-7. Arithmetic instructions.

Addition



If A closes, the contents of register X and register Y are added and stored in register Z. If A does not close, no addition is performed. If contact A was omitted, the addition would be performed in every scan.

Figure 9-61. Ladder format addition.

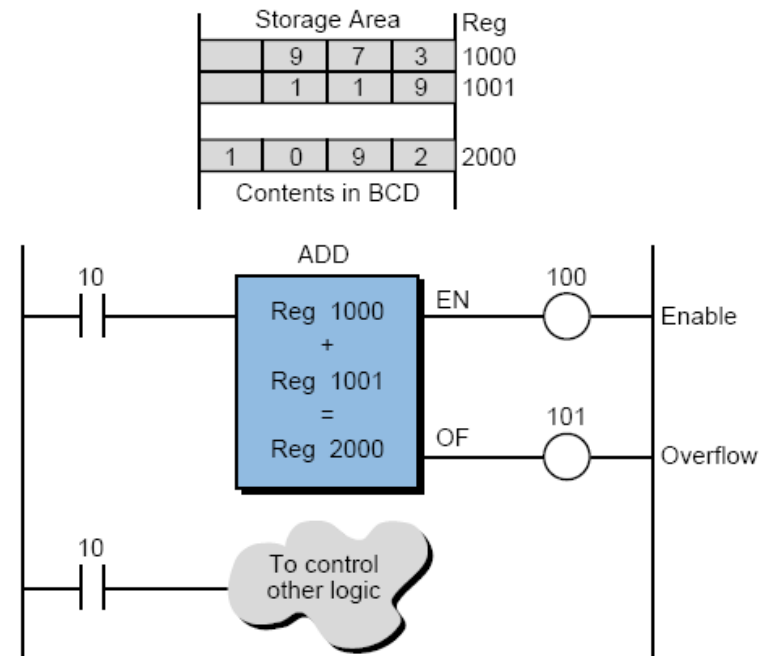


Figure 9-62. Addition functional block.

Addition Example



- Two ingredients are added to a reactor tank for mixing.
- Analog input modules, which provide 12-bit information in BCD, send data about the two ingredients' flows to the PLC.
- The values are stored in registers 1000 and 1001.
- Implement instructions to keep track of the total amount of the combined ingredients, so that this information can be displayed on a monitor for the operator.

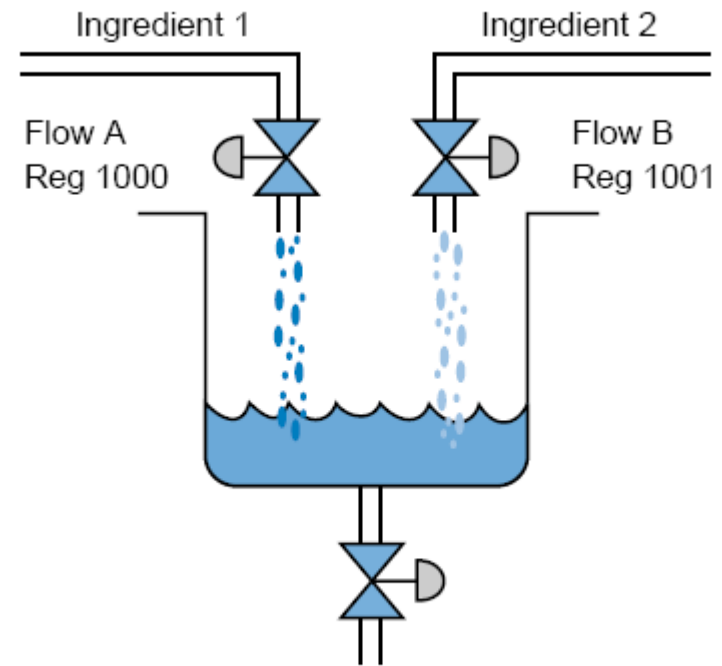
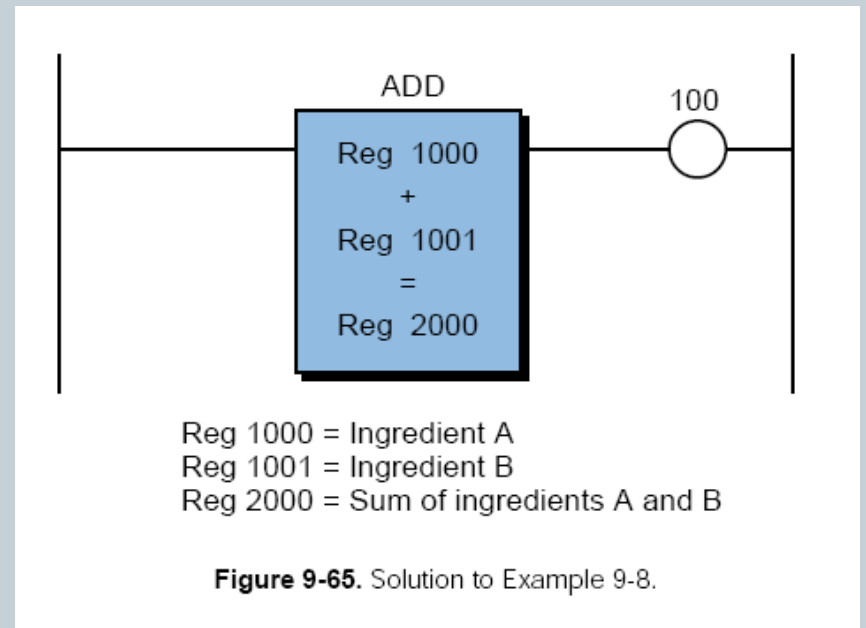
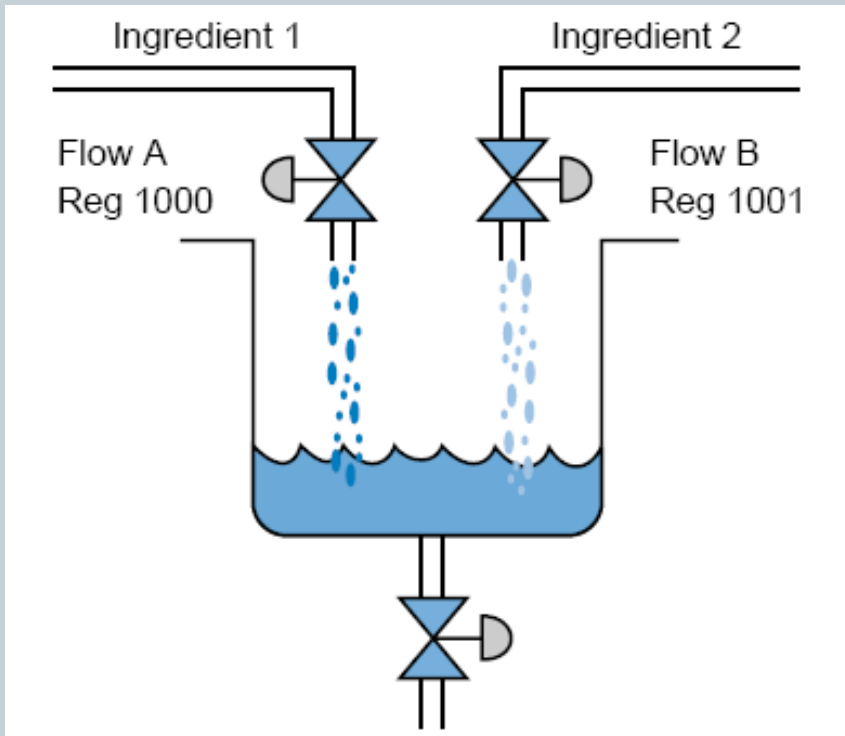
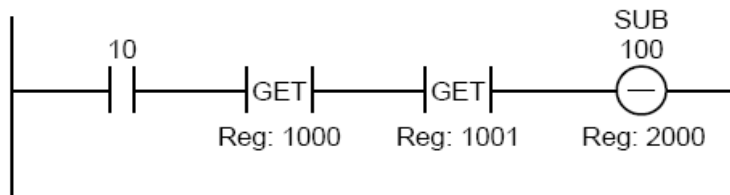


Figure 9-64. Flow of two ingredients into a reactor tank.

Addition Example



Subtraction



Storage Area	Reg
2 4 8 7 5	1000
1 2 6 6	1001
2 3 6 0 9	2000

Contents in
Decimal (Binary)

If contact 10 closes, the value in register 1001 is subtracted from the value in register 1000 (Reg 1000 – Reg 1001) and the result is stored in register 2000. If contact 10 does not close, no subtraction is performed.

Figure 9-66. Ladder format subtraction instruction.

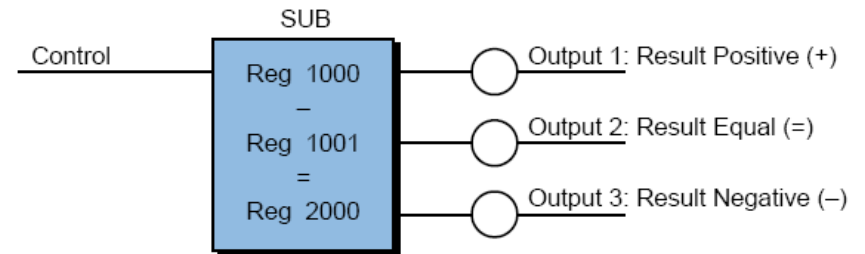


Figure 9-68. Subtraction block with sign outputs.

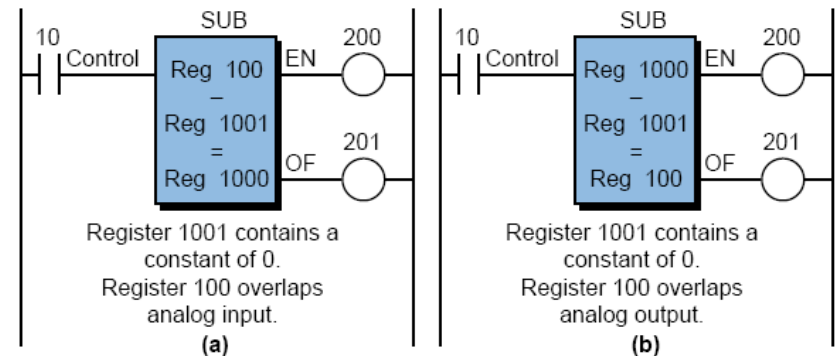


Figure 9-69. Subtraction block used to (a) read an analog input and (b) write an analog output.

Division

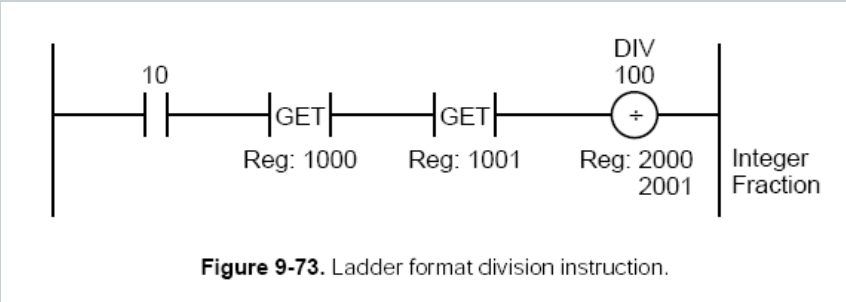


Figure 9-73. Ladder format division instruction.

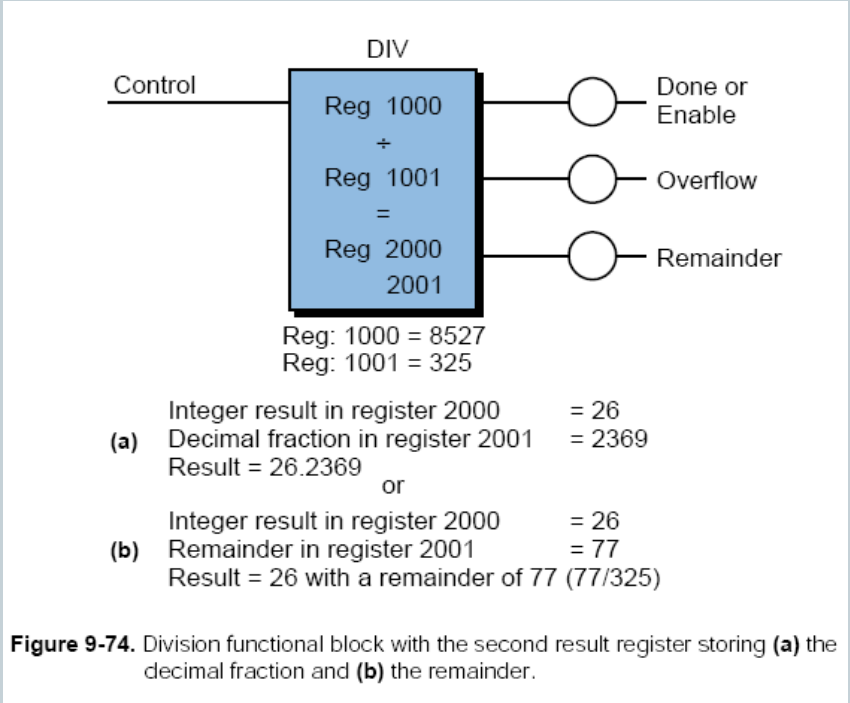


Figure 9-74. Division functional block with the second result register storing (a) the decimal fraction and (b) the remainder.

Square Root

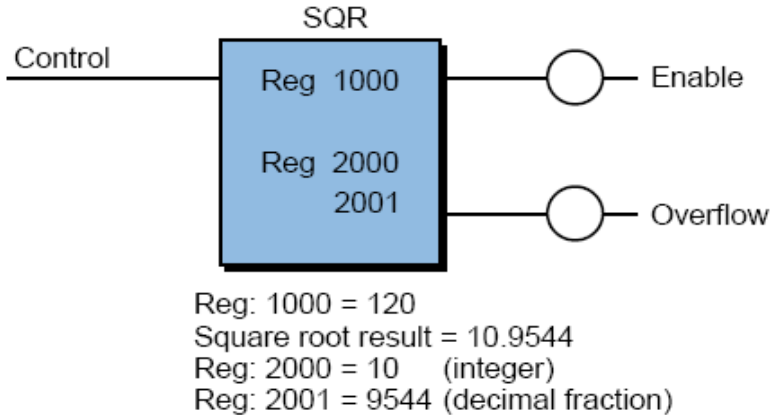


Figure 9-75. Square root functional block.

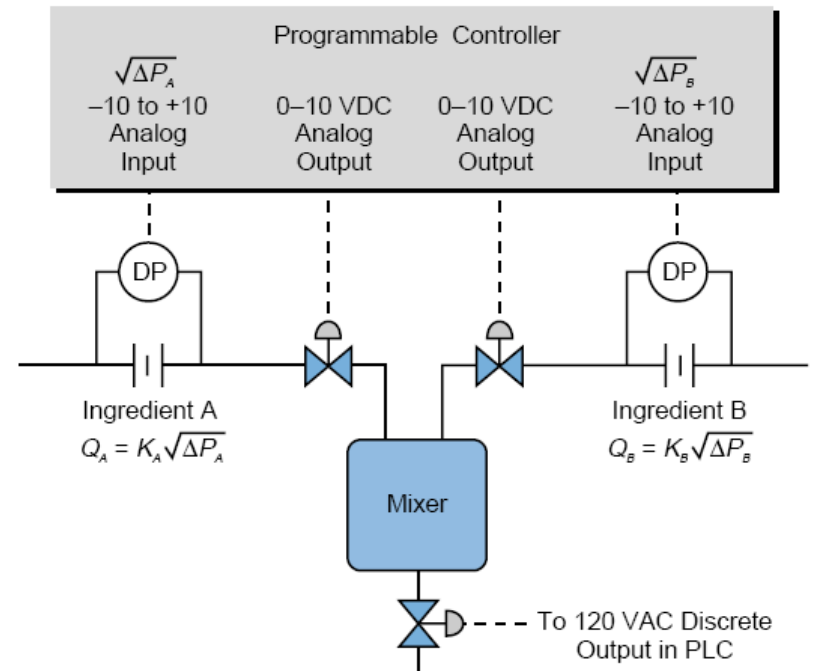


Figure 9-76. Square root instruction application in a DP flow meter.

Data Manipulation Instructions



- Relay-type instructions are limited to the control of internal and external outputs based on the status of specific bit addresses, data manipulation instructions allow multi-bit operations.
- Data manipulation instructions handle operations that take place within one, two, or more registers.

Data Manipulation Instructions



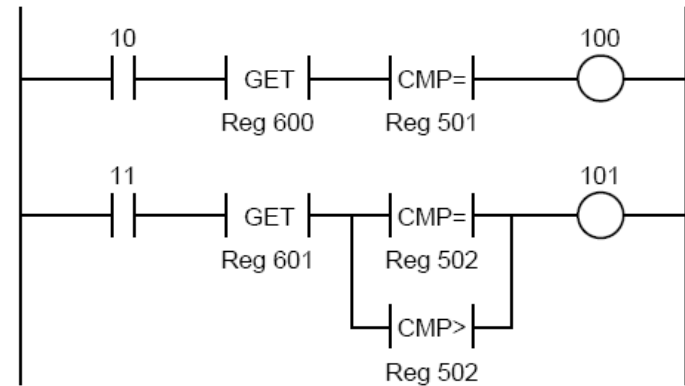
Data Manipulation Instructions <i>(Purpose: To provide multibit, multiregister operations in a PLC)</i>		
Instruction	Symbol	Function
Data Comparison	CMP/LIM	Compares the values stored in two registers
Logic Matrix	AND/OR/NAND NOR/NOT/XOR	Performs logic operations on two or more registers
Data Conversion	ABS/COMPL INV/BIN-BCD	Changes the value stored in a register to another format
Set Constant Parameters	SET	Loads a register with a fixed value
Increment	INCR	Increases the contents of a register by one
Shift	SHIFT	Moves the bits in a register to the right or left
Rotate	ROT	Shifts register bits right/left and moves the shifted-out bit to the other end of the register
Examine Bit	XBON/XBOFF	Examines the status of a single bit in a memory location

Table 9-8. Data manipulation instructions.

Data Comparison



- *Data comparison (CMP)* instructions compare the values stored in two registers.
- These instructions are useful when checking for values in the application program.
- There are three basic data comparisons: *compare equal to*, *compare greater than*, and *compare less than*.
- Based on the results of these comparisons, the processor can turn outputs ON or OFF and perform other operations.



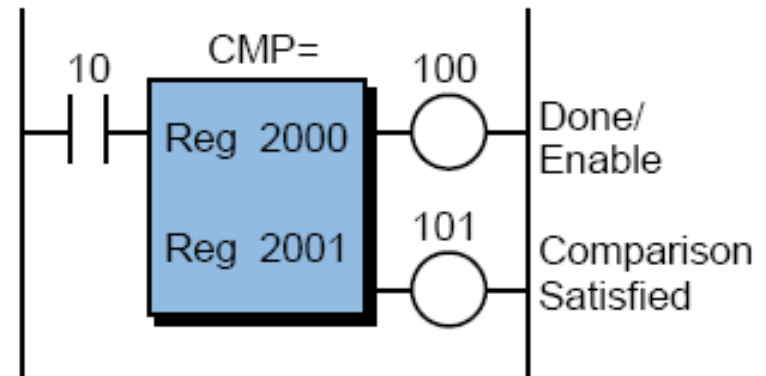
If contact 10 closes, the contents of register 600 are compared to the contents of register 501; if they are equal, coil 100 is turned ON. If contact 11 closes, the contents of register 601 are compared to the contents of register 502; if they are greater than or equal to register 502, output 101 is turned ON.

Figure 9-77. Ladder format comparisons.

Data Comparison



- The compare functional block compares the contents of two registers, register 2000 and register 2001, for a specific comparison, in this case, equal to.
- The block instruction energizes output coil 100 when the comparison occurs, and it energizes output coil 101 if the comparison has been satisfied.

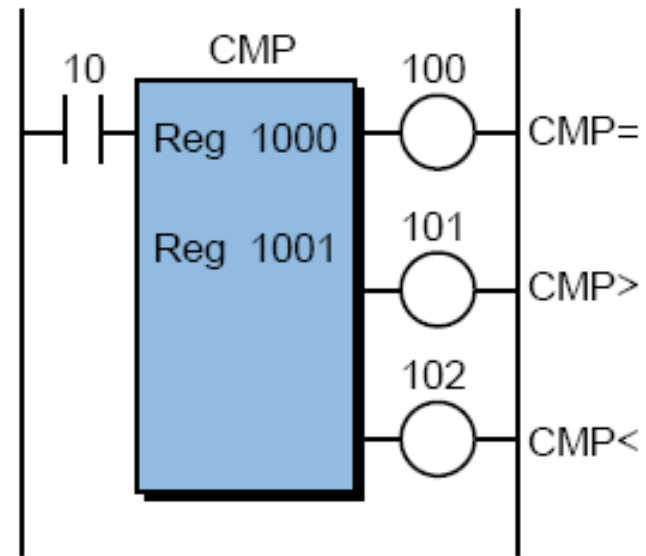


Note: Other comparison functional block instructions include CMP>, CMP≥, CMP<, CMP≤, and CMP≠.

Data Comparison



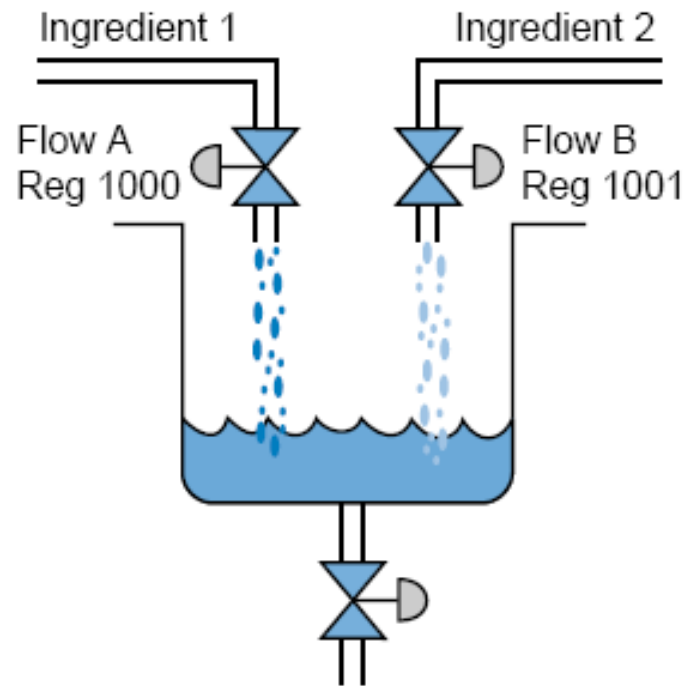
- Some PLCs may also have one comparison block, which has several outputs, that performs multiple compare functions at the same time.
- This type of comparison block compares the data in the registers and then turns ON the output corresponding to the outcome of the comparison (i.e., less than, greater than, equal to).



Example



- Two ingredients are being poured into a reactor tank.
- The first two ladder rungs open the valves for ingredients A and B, allowing them to be poured into the tank.
- Implement an instruction block that ensures that the valves close when ingredient A reaches 500 gallons and ingredient B reaches 750 gallons.



Example

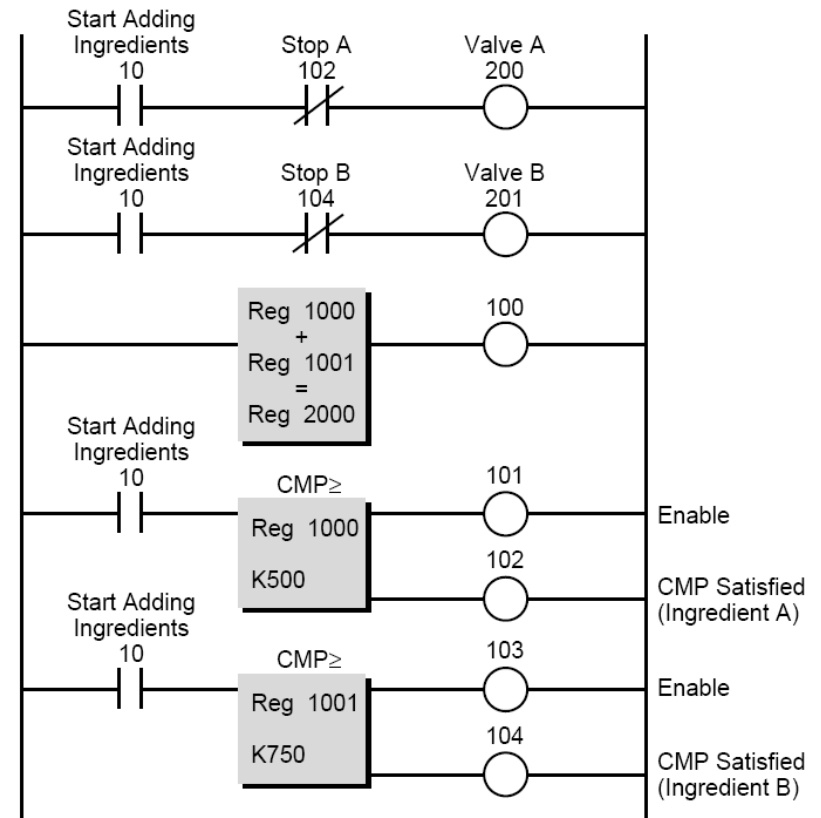
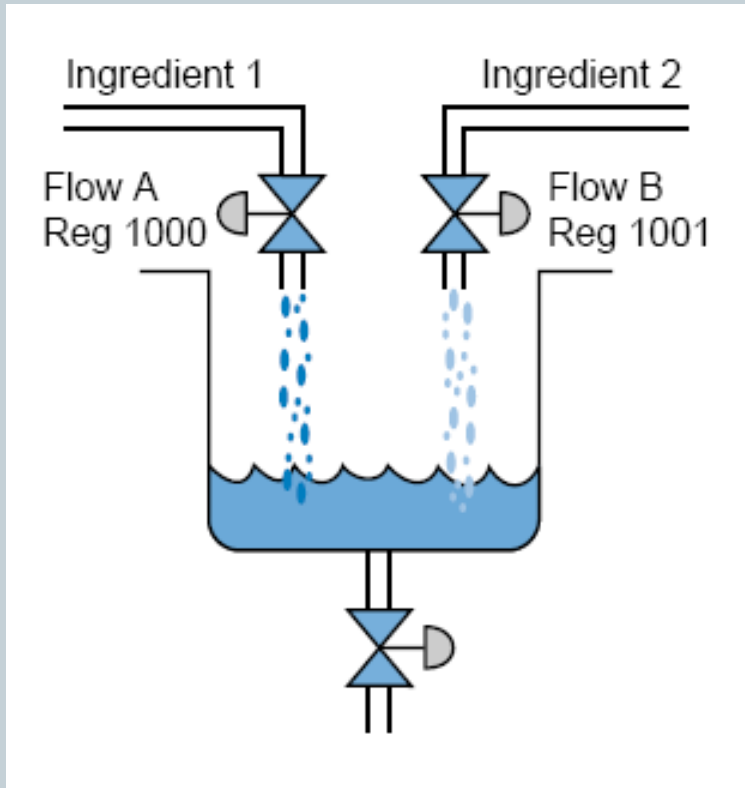


Figure 9-81. Solution to Example 9-9.

Logic Matrix



- A *logic matrix* functional block performs AND, OR, exclusive-OR, NAND, NOR, and NOT logic operations on two or more registers.
- The block specifies the type of logic function to be performed, while the user specifies the registers inside the block.
- In this example, registers 1000 and 1100 hold the operand values, while register 2000 holds the result of the operation.

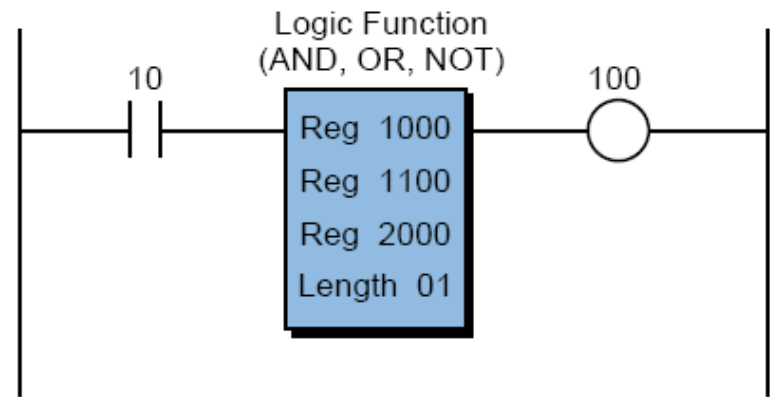


Figure 9-82. Logic matrix functional block.

Logic Matrix

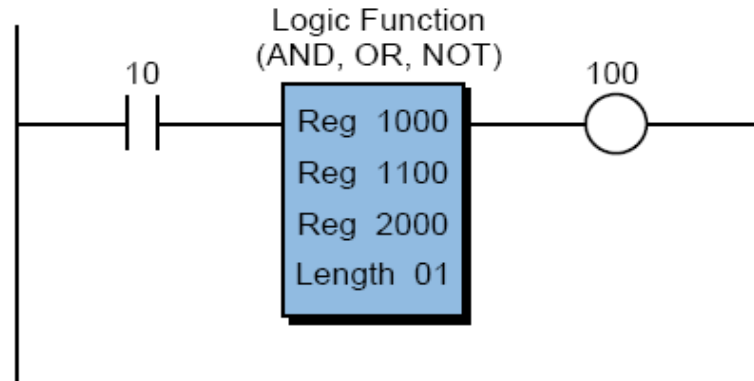


Figure 9-82. Logic matrix functional block.

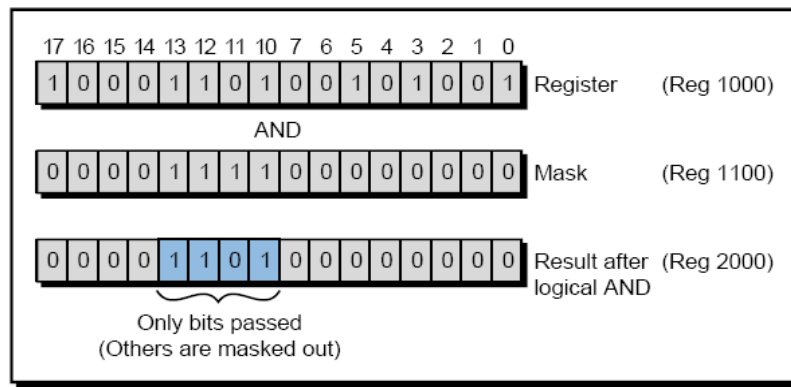
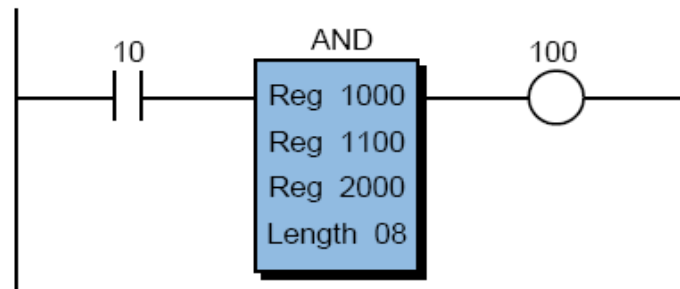


Figure 9-83. Logic matrix function block used to mask out bits.

Logic Matrix



Reg 1000 Holds data to be masked
Reg 1100 Holds mask
Reg 2000 Holds results

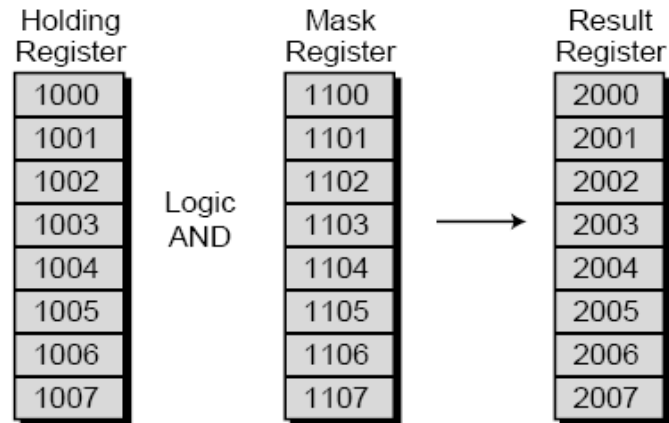


Figure 9-84. Logic matrix function block example.

Data Conversion



- *Data conversion* instructions change the contents of a given register from one format to another.
- Typical data conversion instructions include BCD-to-binary, binary-to-BCD, absolute, complement, and inversion.

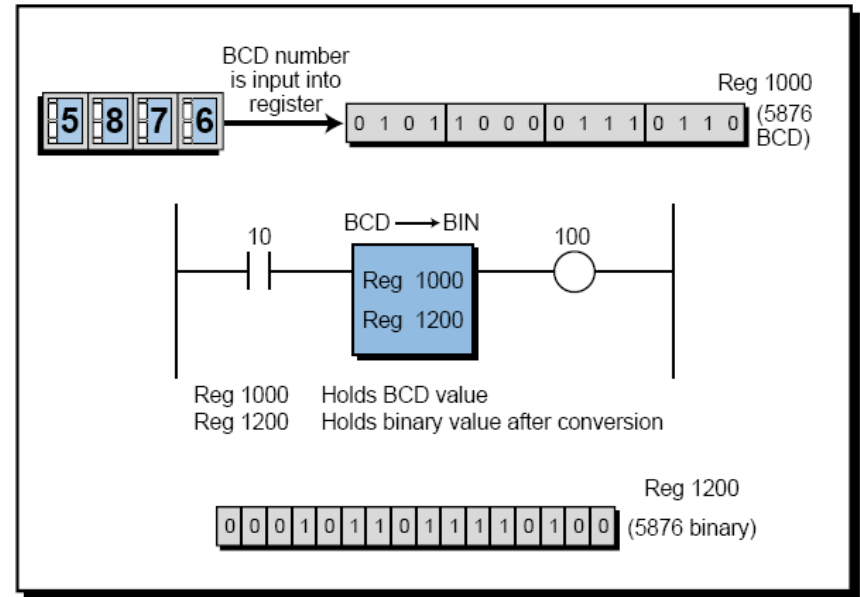
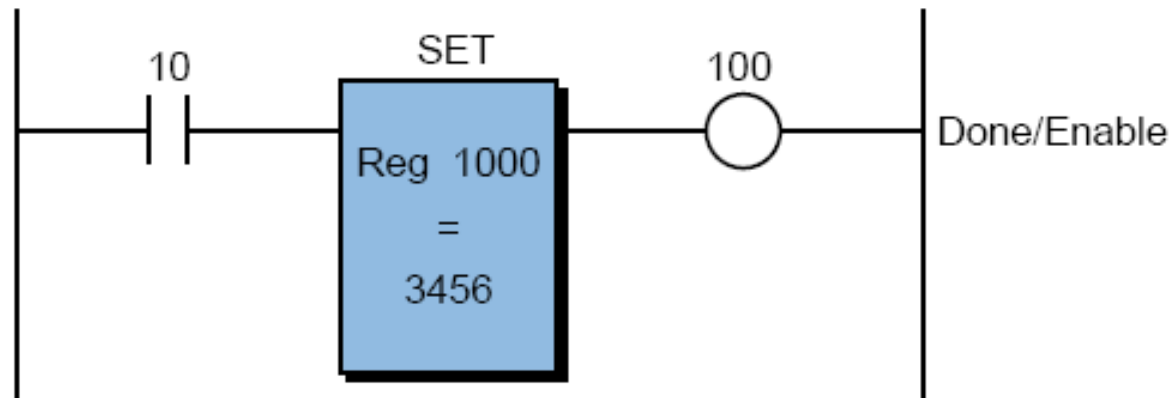


Figure 9-85. BCD-to-binary data conversion.

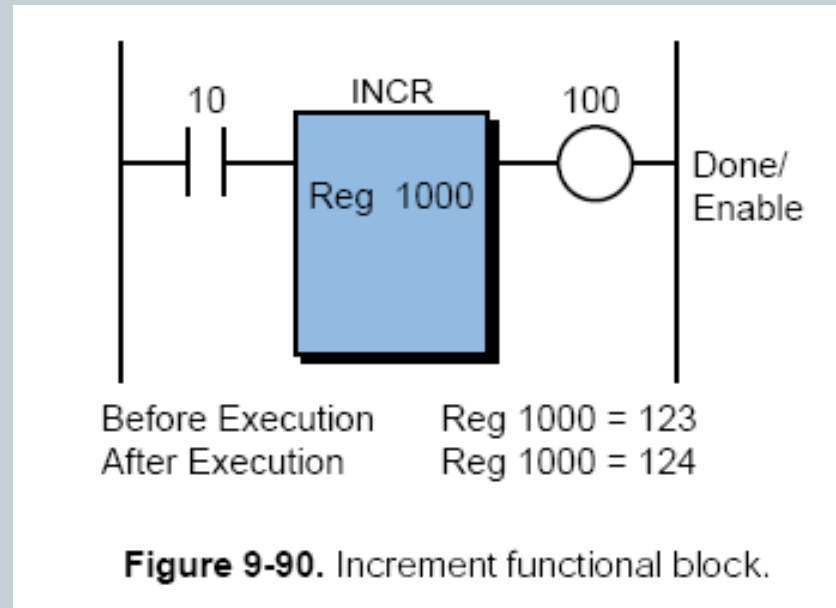
Set Constant Parameter



After Execution Reg 1000 = 3,456

Figure 9-89. Set constant parameters functional block.

Increment



Shift and Rotate

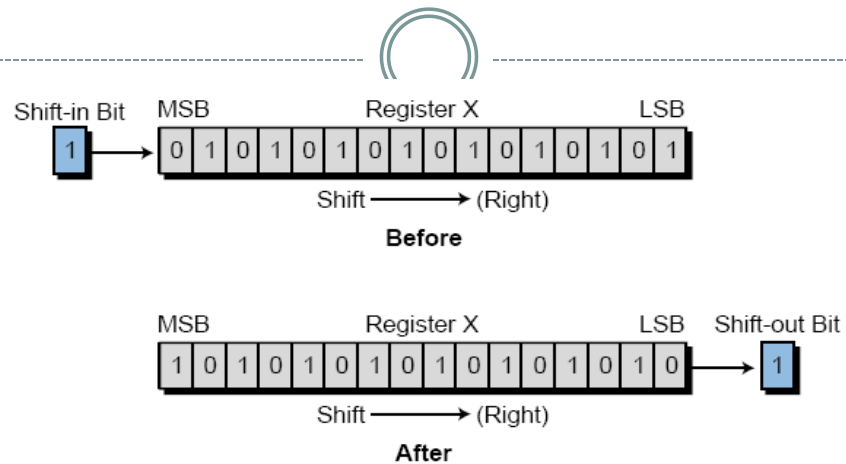


Figure 9-91. Right-shift execution.

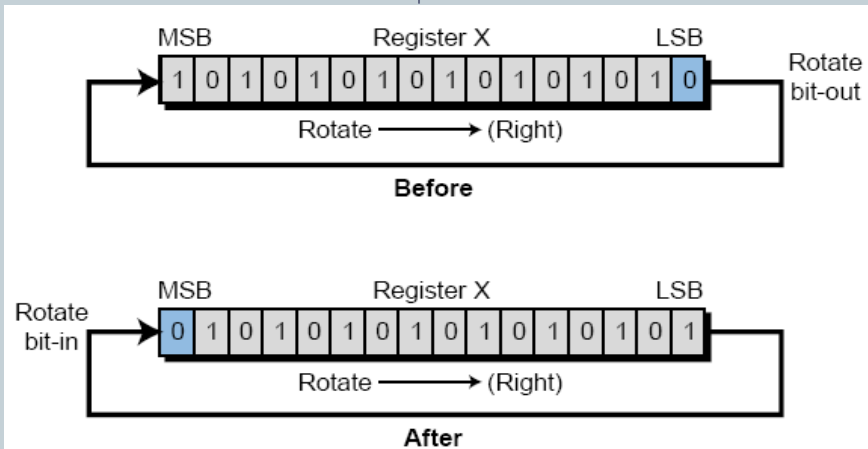
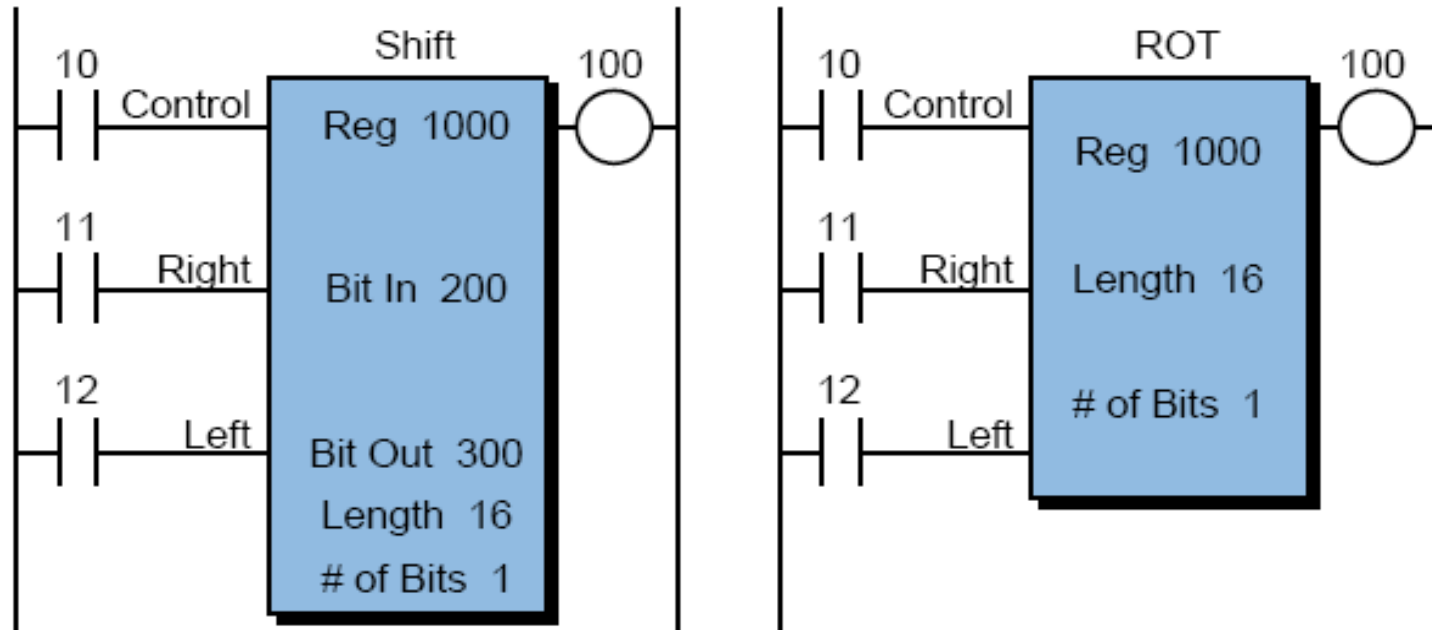


Figure 9-92. Right-rotate execution.

Shift and Rotate



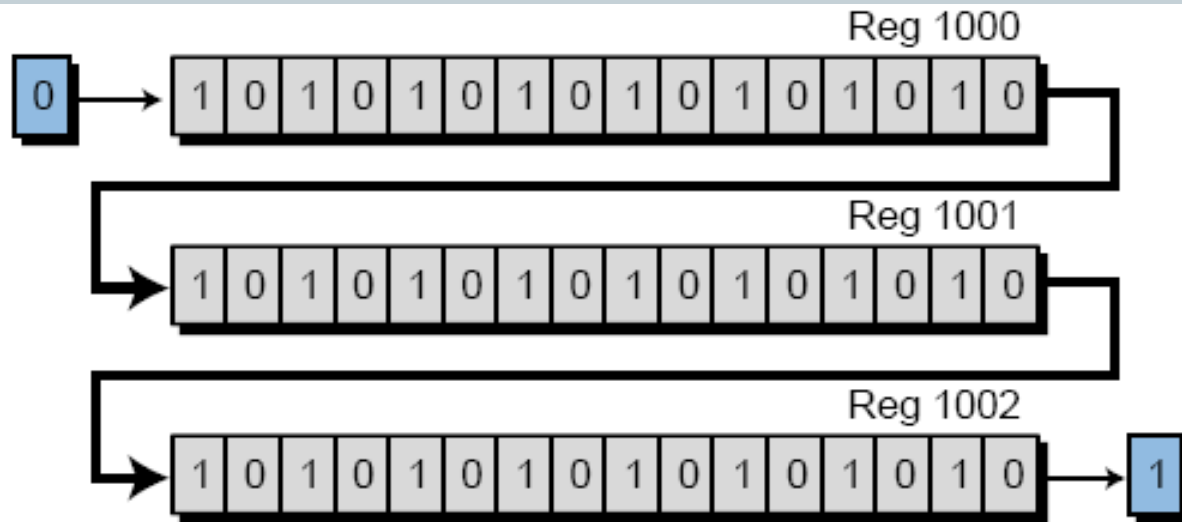
Bit in or out can be a real I/O address or a bit in a register

(a)

(b)

Figure 9-93. (a) Shift and (b) rotate functional blocks.

Shift and Rotate



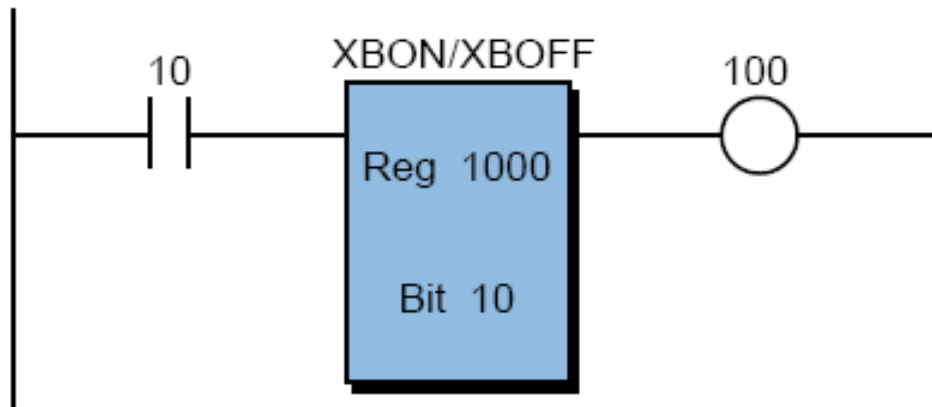
Shift Length = 3 (48 bits)

of Bits = 1

Shift in Bit = 0

Figure 9-94. Example of a right-shift instruction.

Examine Bit



Examines bit 10 of register 1000 for an ON (XBON) or an OFF (XBOFF) status.

Figure 9-95. Examine bit functional block.

Example



- A PLC application controls a batching process where the reading of a temperature input (Batch Temp) is critical to the process.
- The process's temperature transducer is connected to a four-channel, 0–10 VDC analog input module with a 12-bit resolution.
- The remaining four bits of each channel are used as status indicators for the module.
- Illustrate how to test for a fault in this analog input interface's critical temperature measurement.

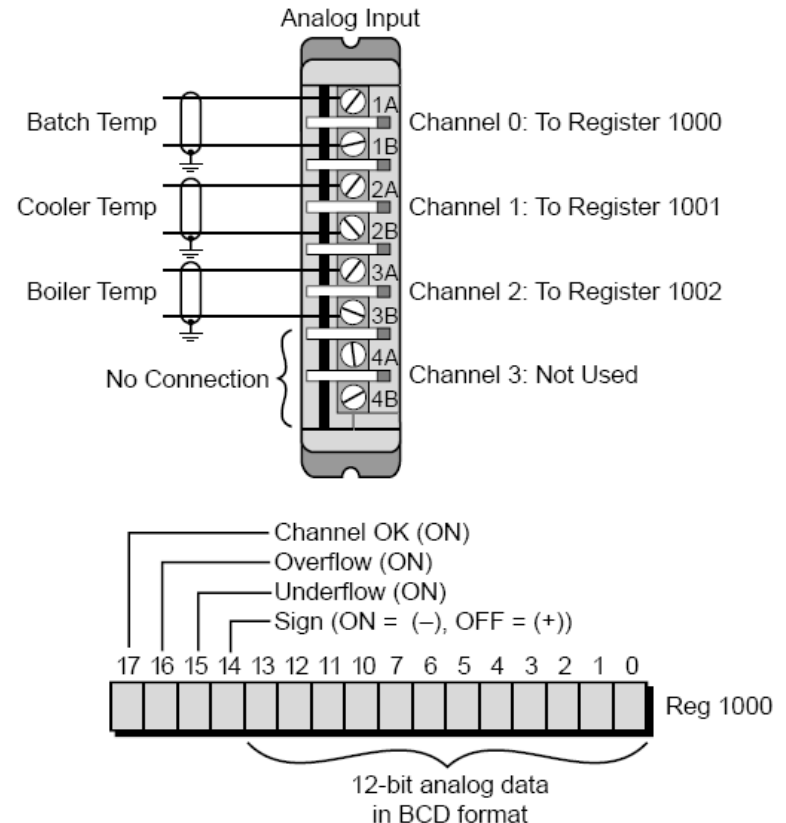


Figure 9-96. Analog input interface for a batching application.

Example



- By testing bit 17 of register 1000 (which is the destination of the critical temperature reading channel) for an OFF condition
- If bit 17 is OFF, a fault has occurred; if it is ON, the channel is OK.

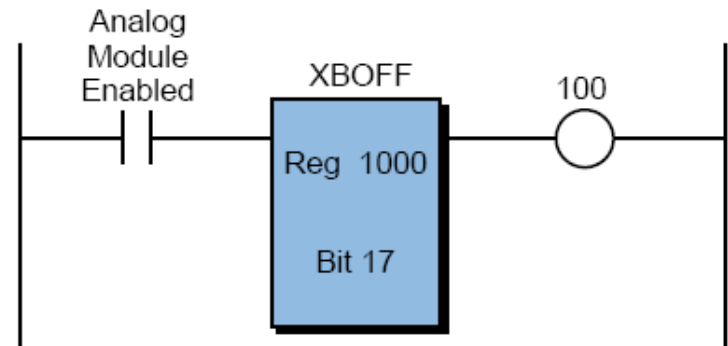
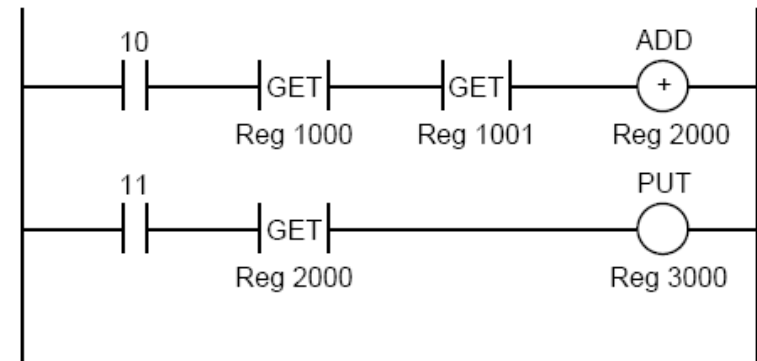


Figure 9-97. A fault has occurred if register 1000 bit 17 is 0 (OFF).

Data Transfer Instructions



- **Data transfer instructions** move, or transfer, numerical data within a PLC, either in single register units or in blocks (a group of registers).
- A GET data transfer instruction accesses data from a certain register, whereas a PUT instruction stores data in a specified register.



If contact 10 closes, the contents of registers 1000 and 1001 are added and stored in register 2000. If contact 11 closes, the contents of register 2000 are transferred (stored) into register 3000. The contents of register 2000 are not altered.

Figure 9-98. GET and PUT instructions used in the ladder format.

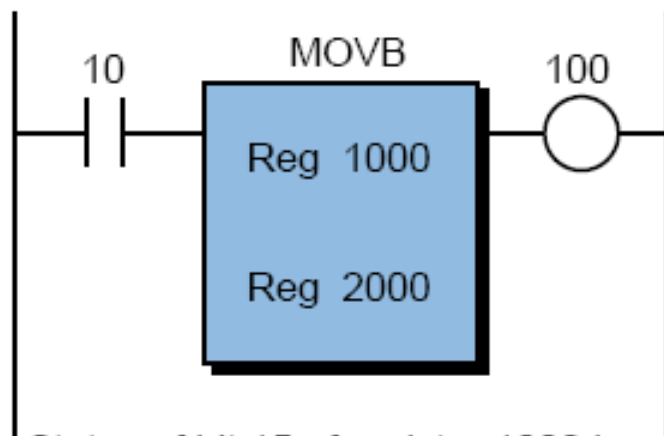
Data Transfer Instructions



Data Transfer Instructions <i>(Purpose: To move numerical data within a PLC)</i>		
Instruction	Symbol	Function
Move	MOV/MOVB/ MOVR/MOVM	Transfers information from one location to another
Move Block	MOVBK	Moves data from a group of register locations to another location
Table Move	REG-TABLE/ TABLE-REG	Transfers data from a block or table to a register
Block Transfer— In/Out	BKXFER	Stores a block of data in specified memory or register locations
ASCII Transfer	ASCII XFER	Transmits ASCII data between a peripheral device and a PLC
First In–First Out Transfer	FIFO	Constructs a table or queue for storing data
Sort	SORT	Sorts the data in a block of registers in ascending/descending order

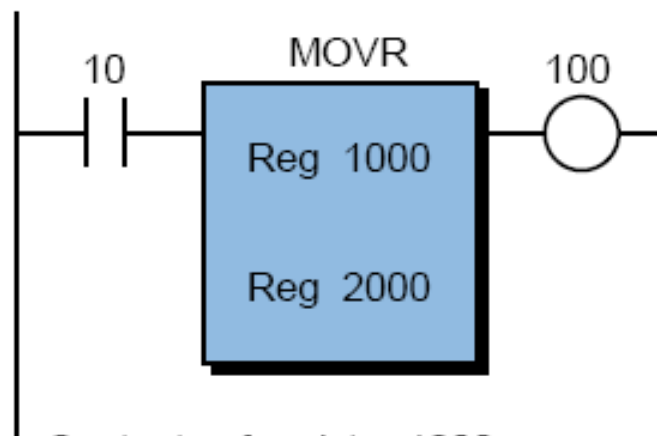
Table 9-9. Data transfer instructions.

MOVE



Status of bit 15 of register 1000 is moved to bit 07 of register 2000

(a)

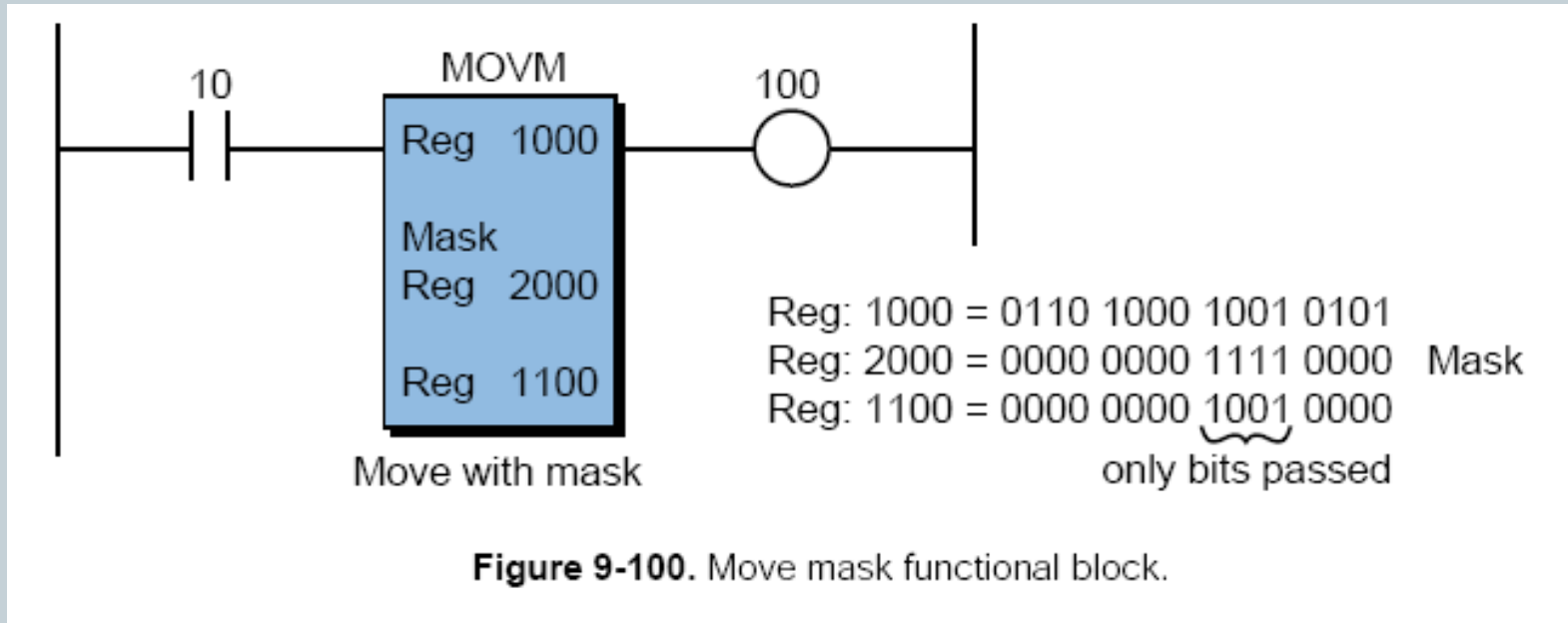


Contents of register 1000 are moved to register 2000 (destination register can be an I/O register)

(b)

Figure 9-99. (a) Move bit and (b) move register functional blocks.

MOVE Mask



MOVE Block

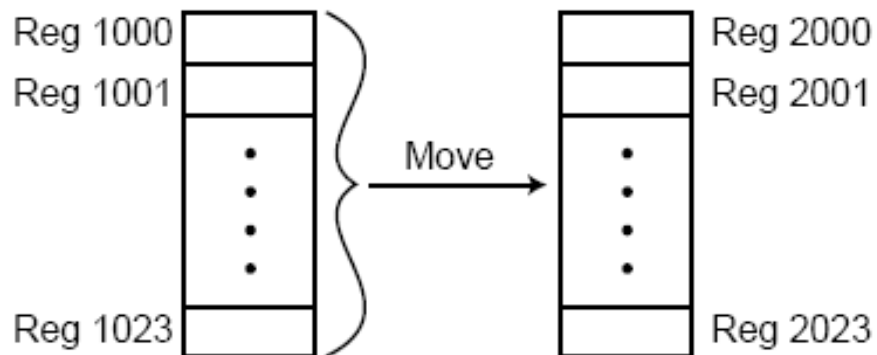
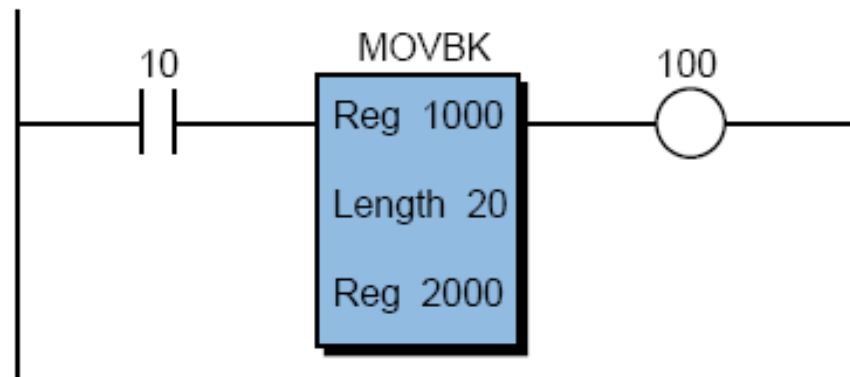


Figure 9-101. Move block functional block.

Special Functions Instructions



Special Function Instructions <i>(Purpose: To allow specialized operations in a PLC)</i>		
Instruction	Symbol	Function
Sequencer	SEQ	Outputs data in a time-driven or event-driven manner
Diagnostic	DIAG	Compares actual input data with reference data
Proportional-Integral-Derivative	PID	Provides closed-loop control of a process

Table 9-10. Special function instructions.

Network Communication Instructions




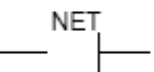
Network Communication Instructions <i>(Purpose: To allow communication through a local area network)</i>		
Instruction	Symbol	Function
Network Output		Passes one-bit status information from a PLC to a network
Network Contact		Captures status information from a network output
Network Send	NET SEND	Sends register information to a network
Network Receive	NET RCV	Captures available register data in a network
Send Node	SEND NODE	Sends register data to a specific node in a network
Get Node	GET NODE	Retrieves register data from a specific node in a network

Table 9-11. Network communication instructions.