



Chapter 5

Basics of PLC Programming

5.1



Processor Memory Organization

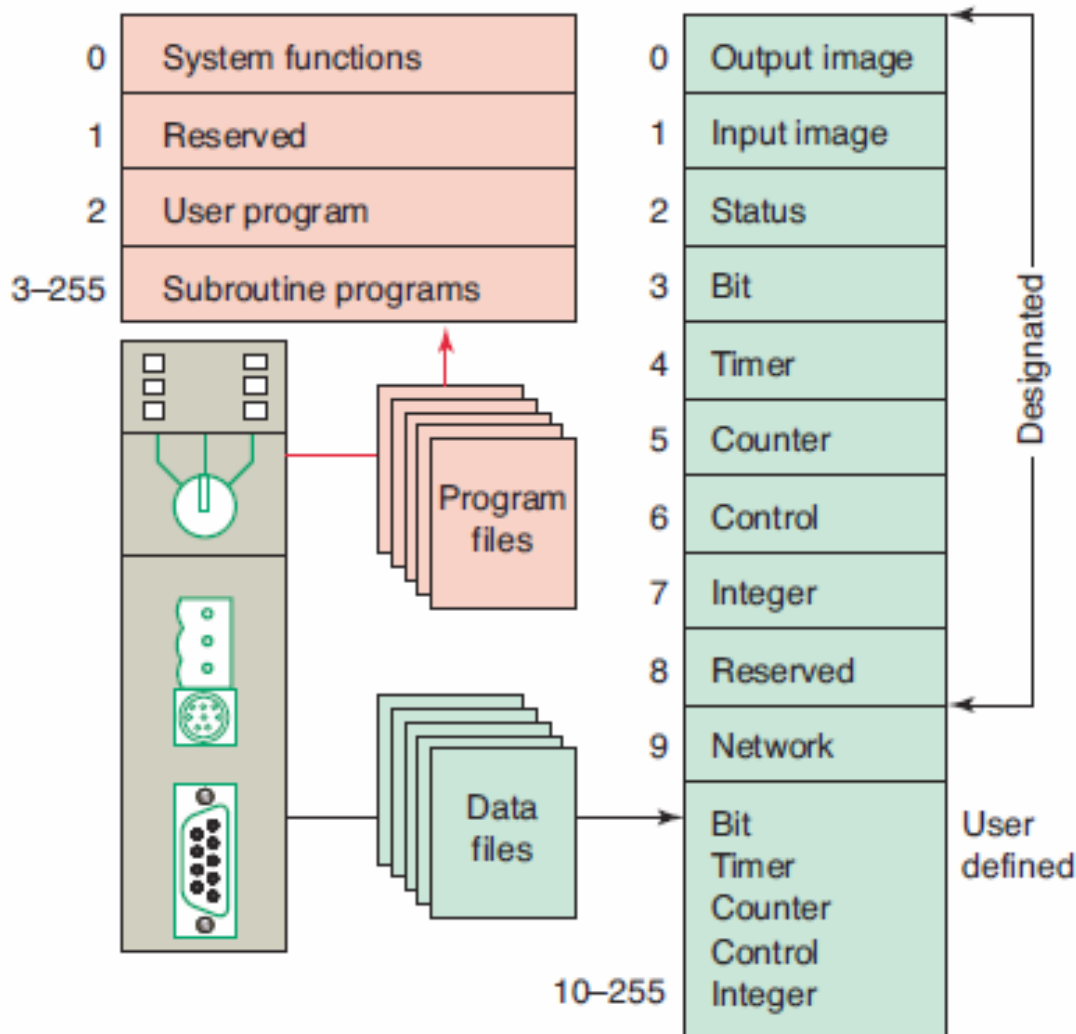
The *memory structure* for a PLC processor consists of several areas, some of these having specific roles.

With **rack-based** memory structures addresses are derived using the rack number, the I/O module slot number and the screw terminal number where the I/O device is wired.



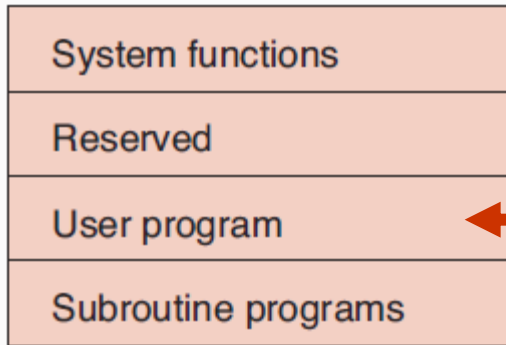
With **tag-based** memory structures all data are assigned a variable name called a tag. A program can be developed using only tag names but you must assign input and output terminals to input and output tags before the program can be executed

The memory space can be divided into two broad categories: *program files* and *Data files*.

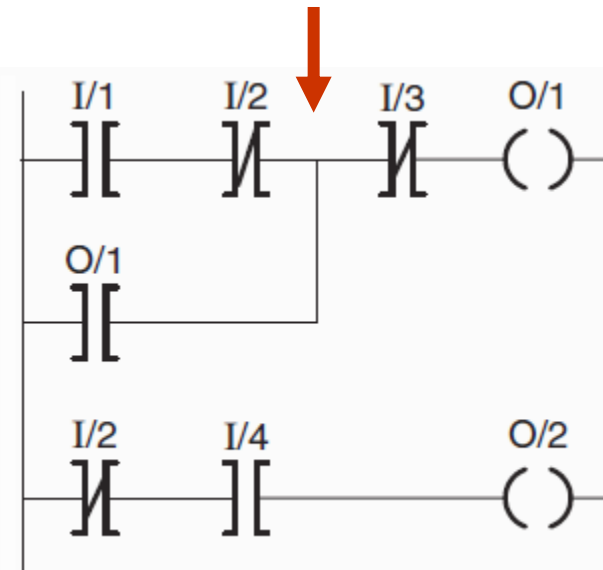
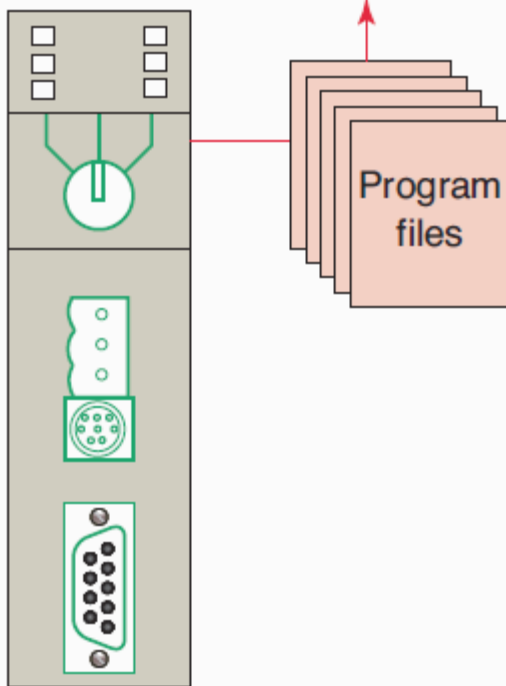


Program and Data file organization for the SLC 500 controller.

Program files are the areas of processor memory where ladder logic programming is stored.

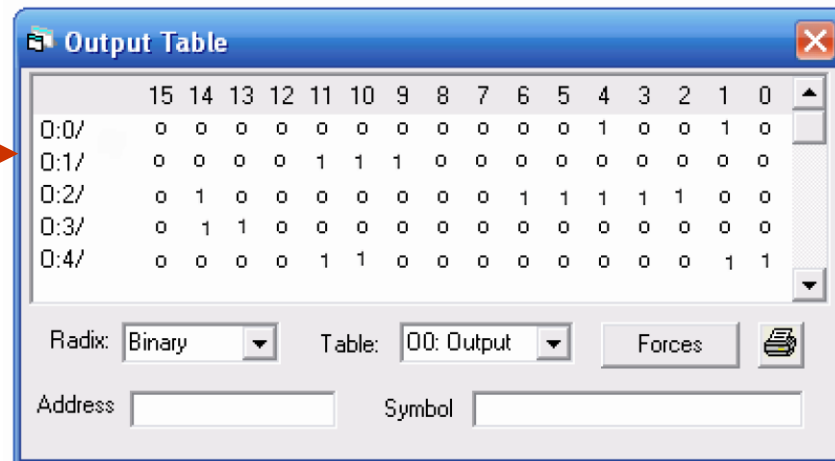
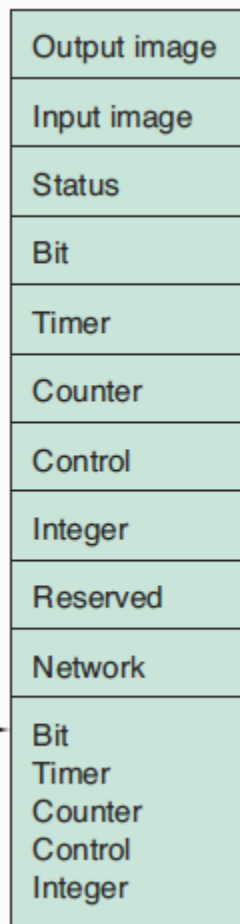


Program files are the part of the processor memory that stores the user **ladder logic program**. The program accounts for most of the total memory of a given PLC system.

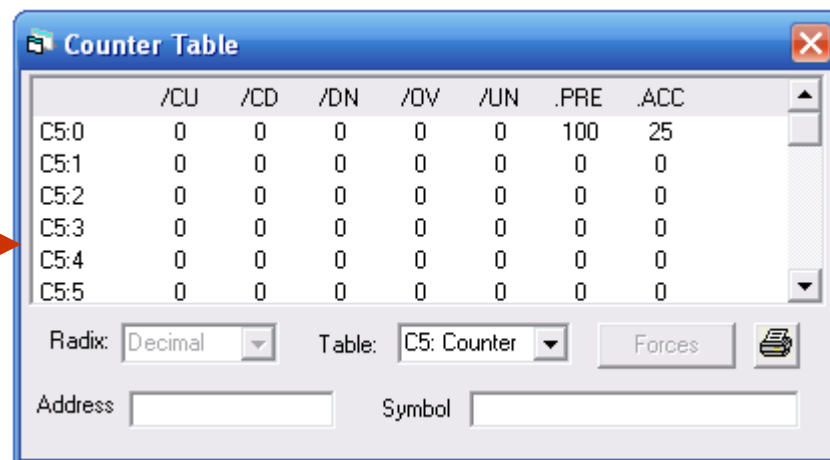


The *Data file* portion of the processor's memory stores input and output status, processor status, the status of various bits, and numerical data.

These files are organized by the **type of data** they contain.

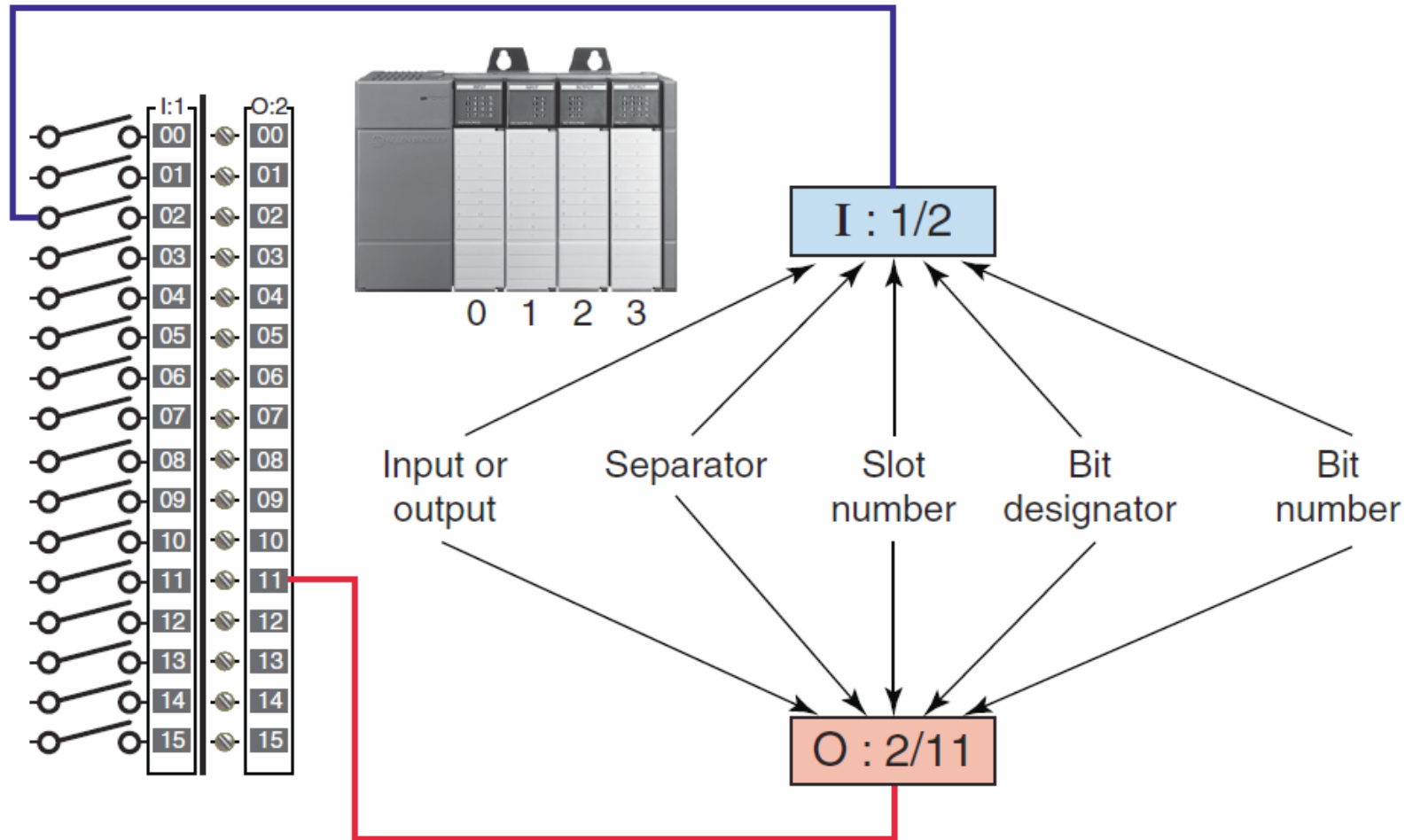


	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0:0/	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0:1/	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
0:2/	0	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0
0:3/	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0:4/	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1

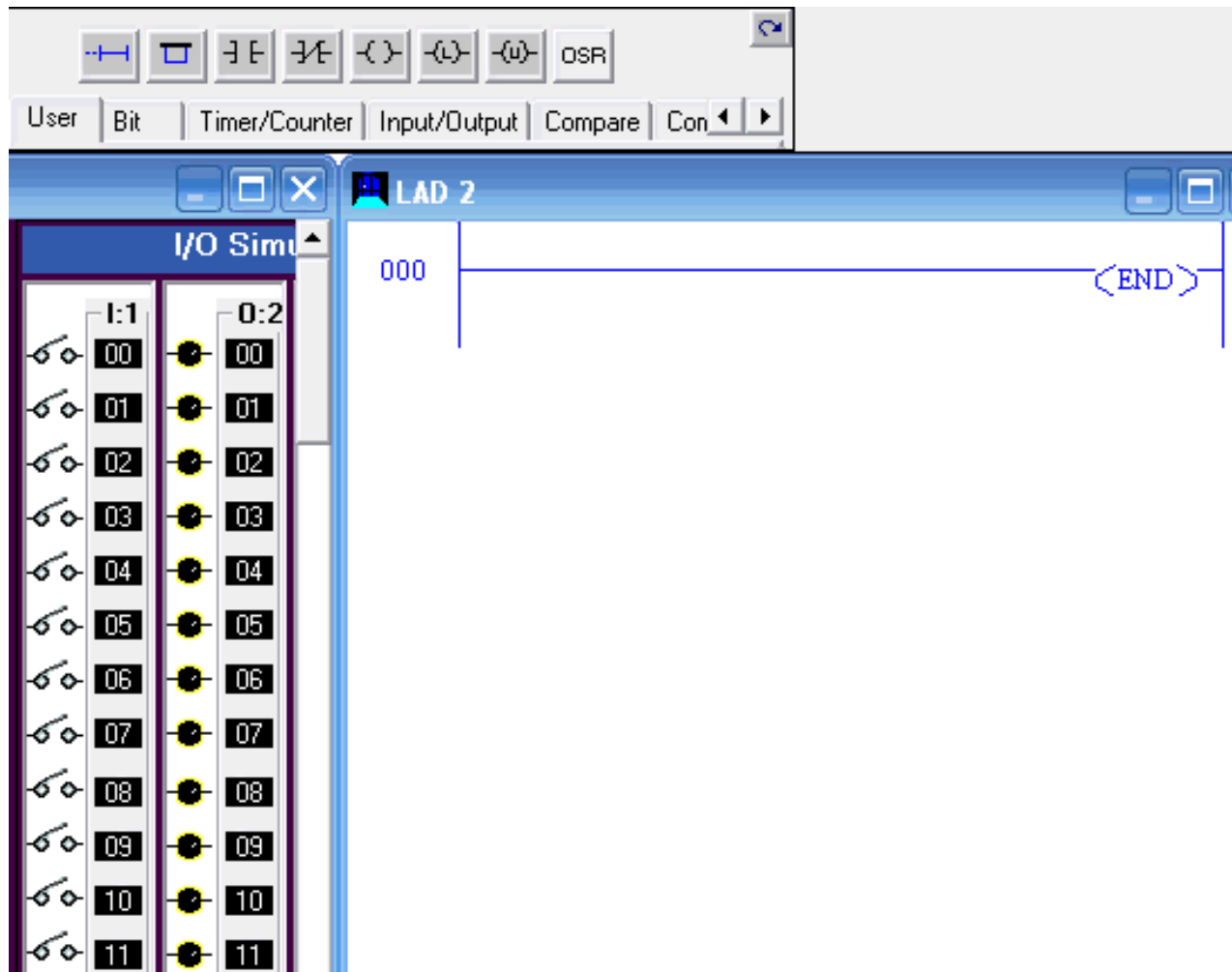


	/CU	/CD	/DN	/OV	/UN	.PRE	.ACC
C5:0	0	0	0	0	0	100	25
C5:1	0	0	0	0	0	0	0
C5:2	0	0	0	0	0	0	0
C5:3	0	0	0	0	0	0	0
C5:4	0	0	0	0	0	0	0
C5:5	0	0	0	0	0	0	0

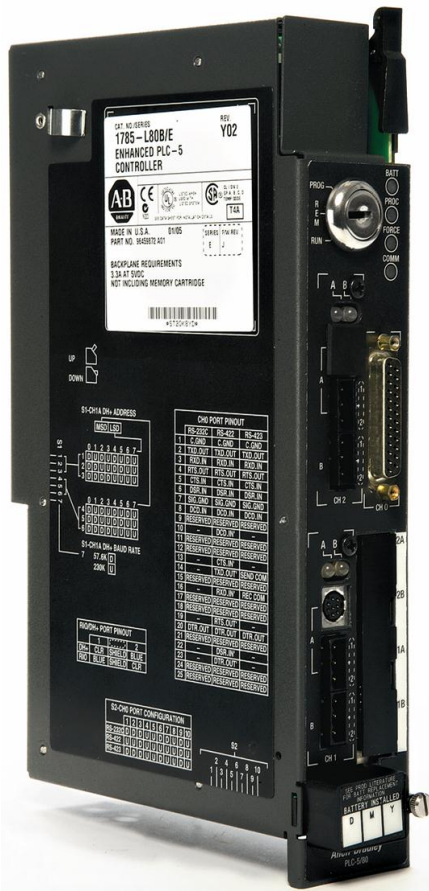
I/O address format for the *SLC* family of PLCs.



Simulated I/O addressing format for the *SLC* family of PLCs.



Memory organization for a *PLC-5* controller.



Address range

O:000

O:037

I:000

I:037

S:000

S:031

B3:000

B3:999

T4:000

T4:999

C5:000

C5:999

R6:000

R6:999

N7:000

N7:999

F8:000

F8:999

Size, in elements

Output image file

32

Input image file

32

Processor status

32

Bit file

1–1000

Timer file

1–1000

Counter file

1–1000

Control file

1–1000

Integer file

1–1000

Floating-point file

1–1000

Files to be assigned file nos. 9–999

1–1000 per file

The PLC-5 and SLC 500 store all data in global data tables and are based on *16-bit* operations.

You access these data by specifying the **address** of the data you want.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I:0/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I:1/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I:2/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I:3/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I:4/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Radix: Binary Table: I1: Input Forces

Address Symbol

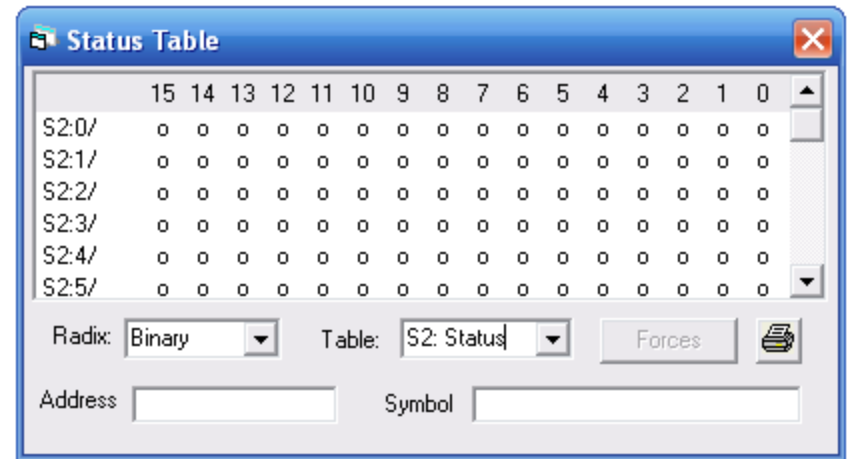
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
O:0/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O:1/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O:2/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O:3/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O:4/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Radix: Binary Table: O0: Output Forces

Address Symbol

The addresses in the output data file and the input data file are potential **locations** for I/O modules mounted in the chassis.

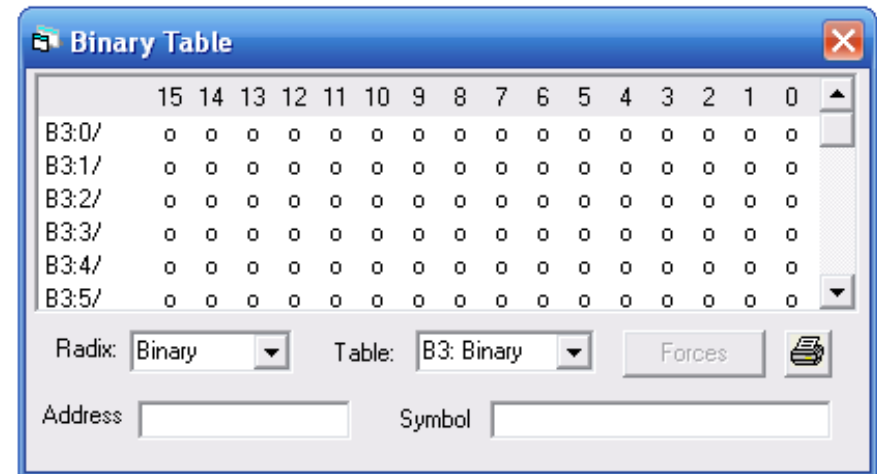
The **status data file** contains information about the processor status



The Status Table window displays a grid of bits for six status registers (S2:0/ through S2:5/). Each register has 16 bits, numbered 15 down to 0. All bits in the grid are currently set to 0. Below the grid, there are controls for Radix (set to Binary), Table (set to S2: Status), a Forces button, a printer icon, and input fields for Address and Symbol.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S2:0/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S2:1/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S2:2/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S2:3/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S2:4/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S2:5/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

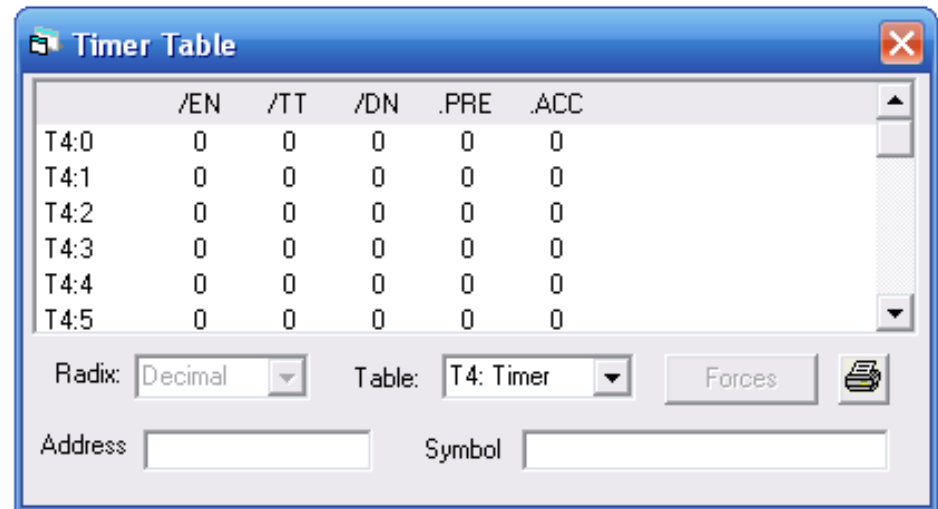
The **bit data file** stores bit status and frequently serves for storage when using internal outputs.



The Binary Table window displays a grid of bits for six bit registers (B3:0/ through B3:5/). Each register has 16 bits, numbered 15 down to 0. All bits in the grid are currently set to 0. Below the grid, there are controls for Radix (set to Binary), Table (set to B3: Binary), a Forces button, a printer icon, and input fields for Address and Symbol.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3:0/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B3:1/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B3:2/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B3:3/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B3:4/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B3:5/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

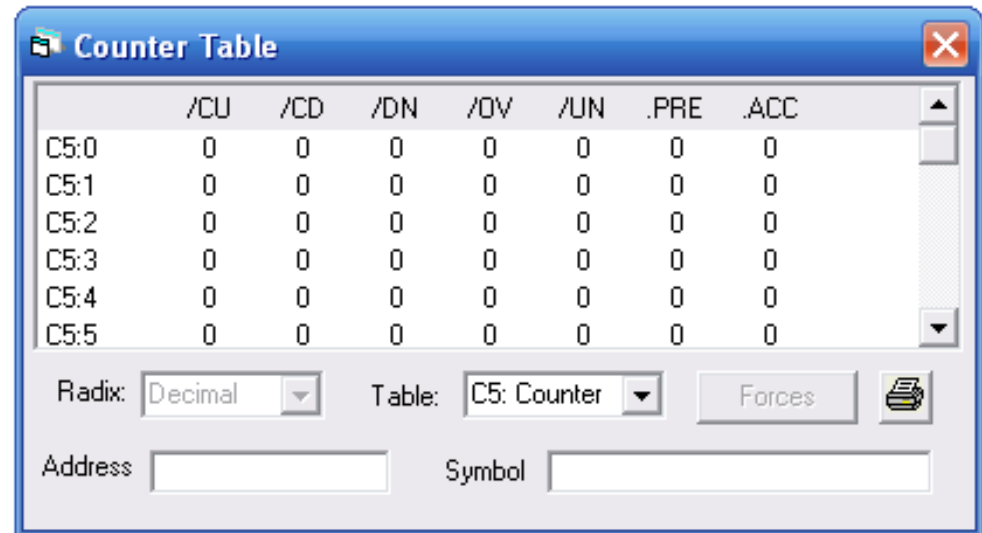
The **timer file** stores the timer status and timer data.



The screenshot shows a dialog box titled "Timer Table" with a table of timer data. The table has columns for /EN, /TT, /DN, .PRE, and .ACC. The data rows are T4:0 through T4:5, all with values of 0. Below the table are controls for Radix (set to Decimal), Table (set to T4: Timer), a Forces button, a print icon, and input fields for Address and Symbol.

	/EN	/TT	/DN	.PRE	.ACC
T4:0	0	0	0	0	0
T4:1	0	0	0	0	0
T4:2	0	0	0	0	0
T4:3	0	0	0	0	0
T4:4	0	0	0	0	0
T4:5	0	0	0	0	0

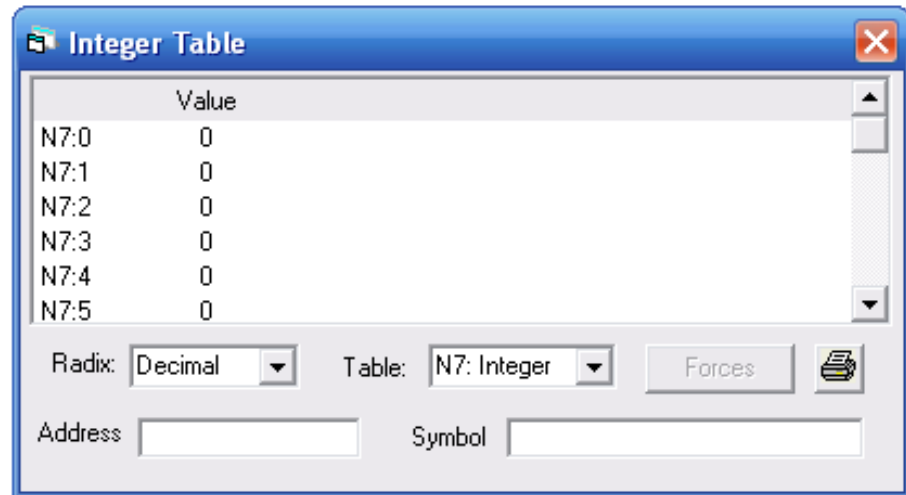
The **counter file** stores the counter status and counter data.



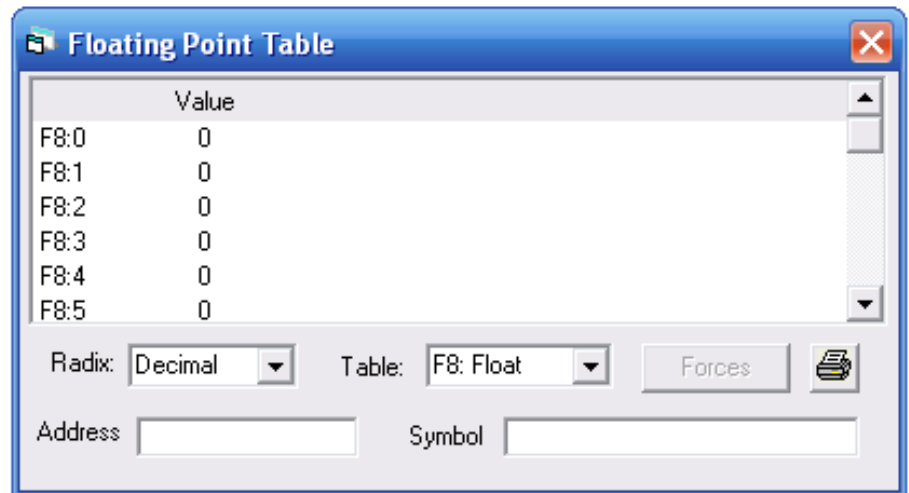
The screenshot shows a dialog box titled "Counter Table" with a table of counter data. The table has columns for /CU, /CD, /DN, /OV, /UN, .PRE, and .ACC. The data rows are C5:0 through C5:5, all with values of 0. Below the table are controls for Radix (set to Decimal), Table (set to C5: Counter), a Forces button, a print icon, and input fields for Address and Symbol.

	/CU	/CD	/DN	/OV	/UN	.PRE	.ACC
C5:0	0	0	0	0	0	0	0
C5:1	0	0	0	0	0	0	0
C5:2	0	0	0	0	0	0	0
C5:3	0	0	0	0	0	0	0
C5:4	0	0	0	0	0	0	0
C5:5	0	0	0	0	0	0	0

The **integer file** stores integer data values.



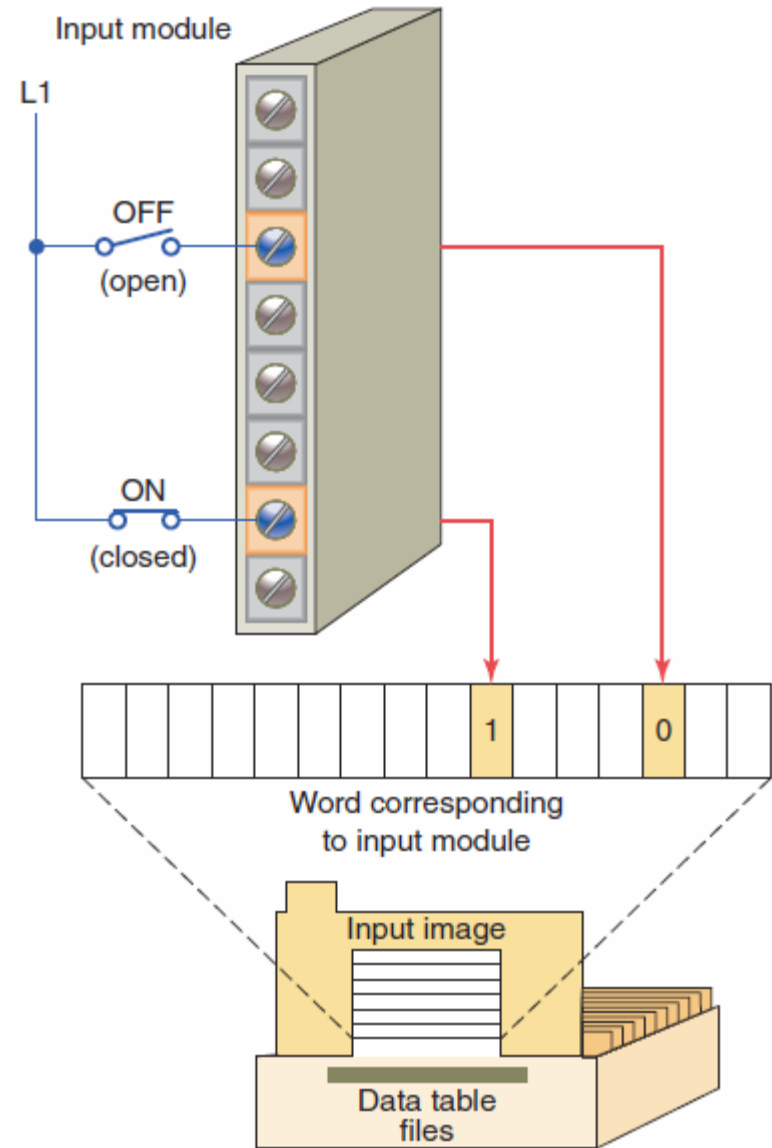
The **floating point file** can store data that requires a decimal point.



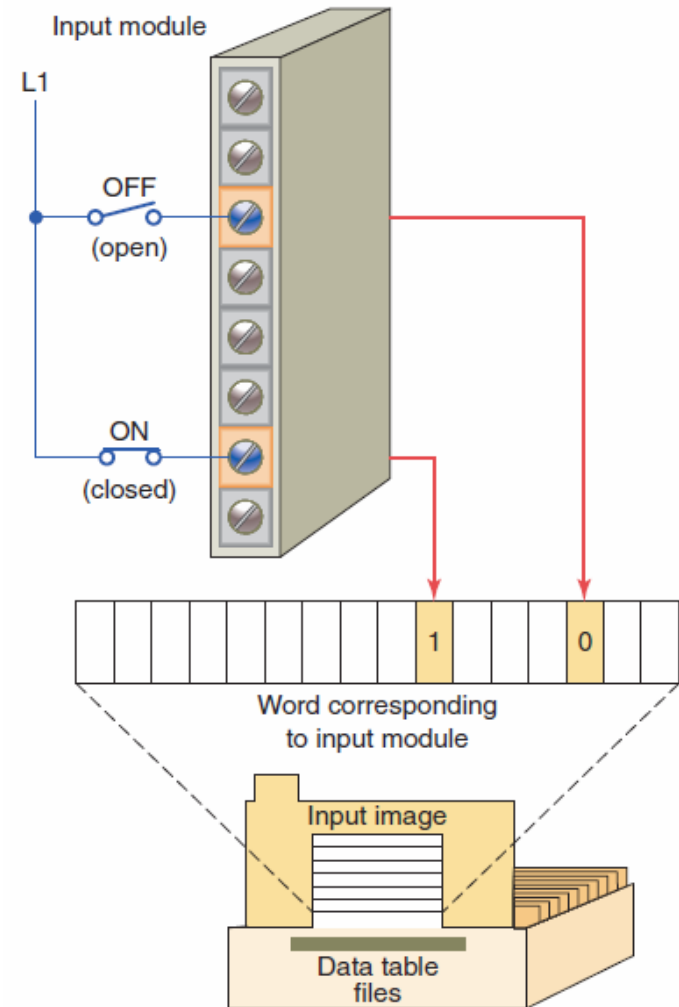
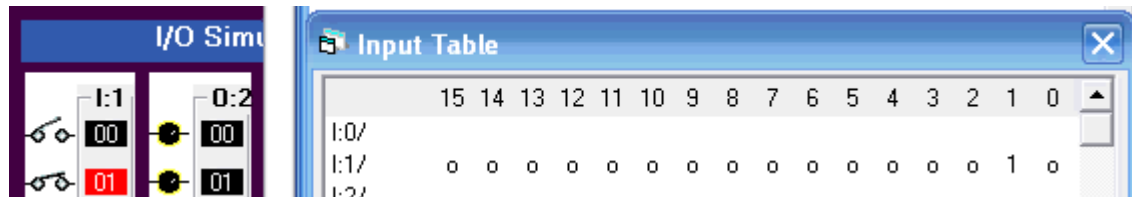
The *input image* table is allocated to storing the *on/off status* of connected discrete inputs.

If the input is **on** (switch closed), its corresponding bit in the table is set to **1**.

If the input is **off** (switch open), the corresponding bit is cleared, or reset to **0**.



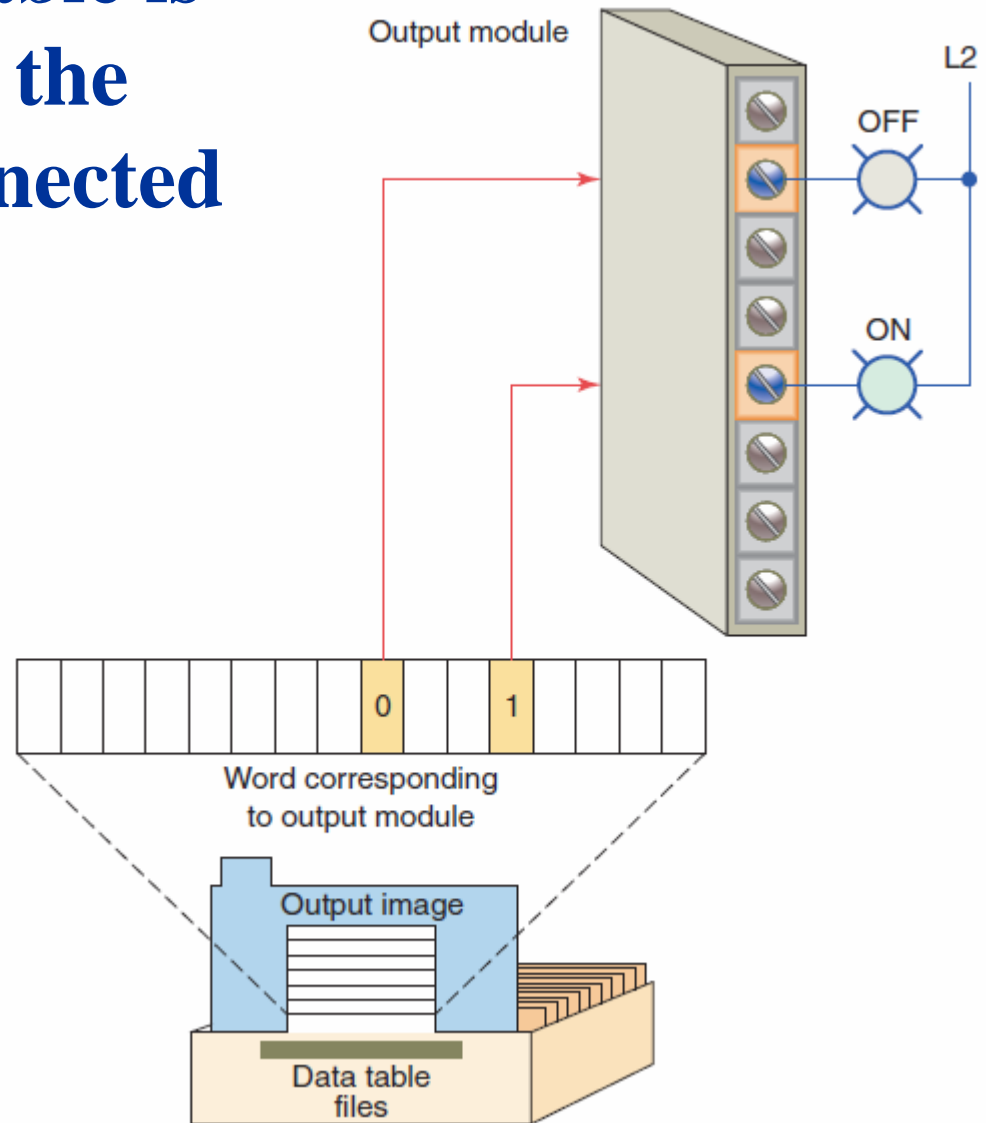
Simulated operation of the *input image* table *on/off* status of connected discrete inputs.



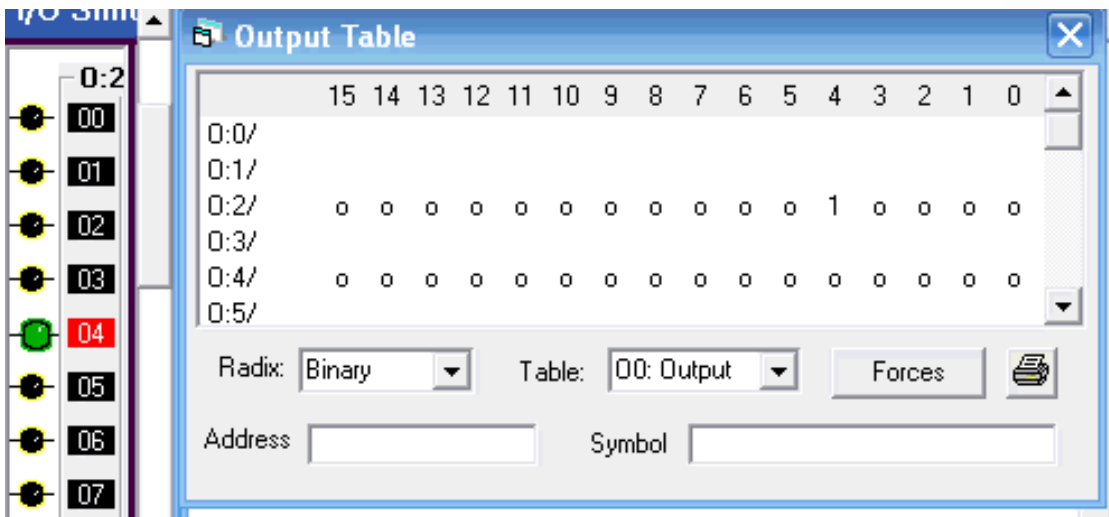
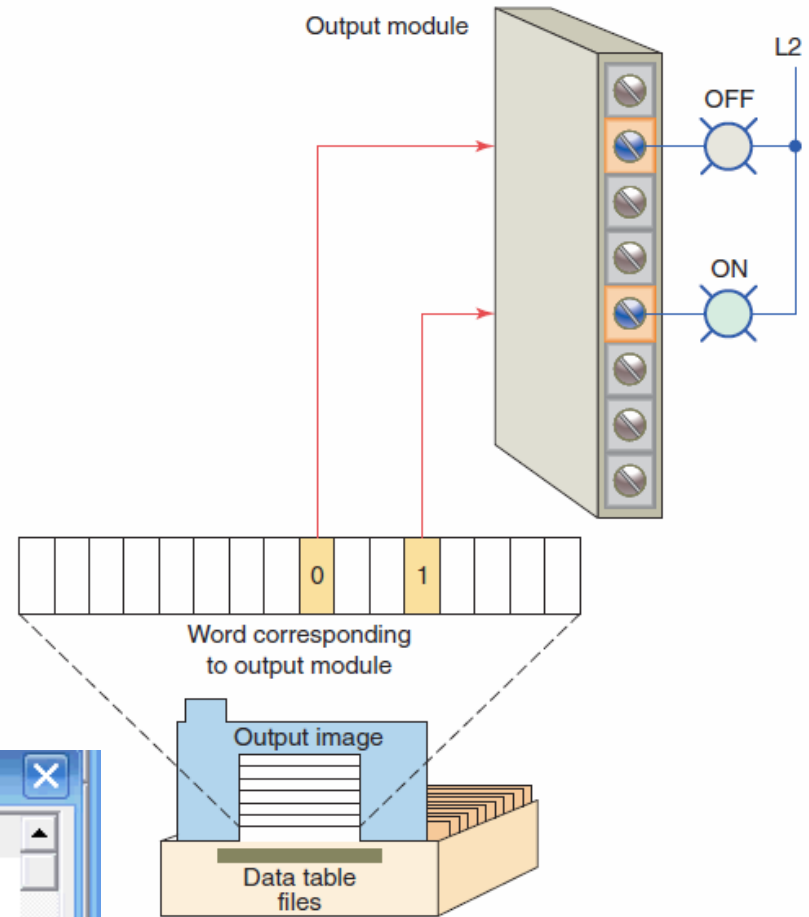
The *output image* table is allocated to storing the on/off status of connected discrete outputs.

If the program calls for an output to be **ON**, its corresponding bit in the table is set to **1**.

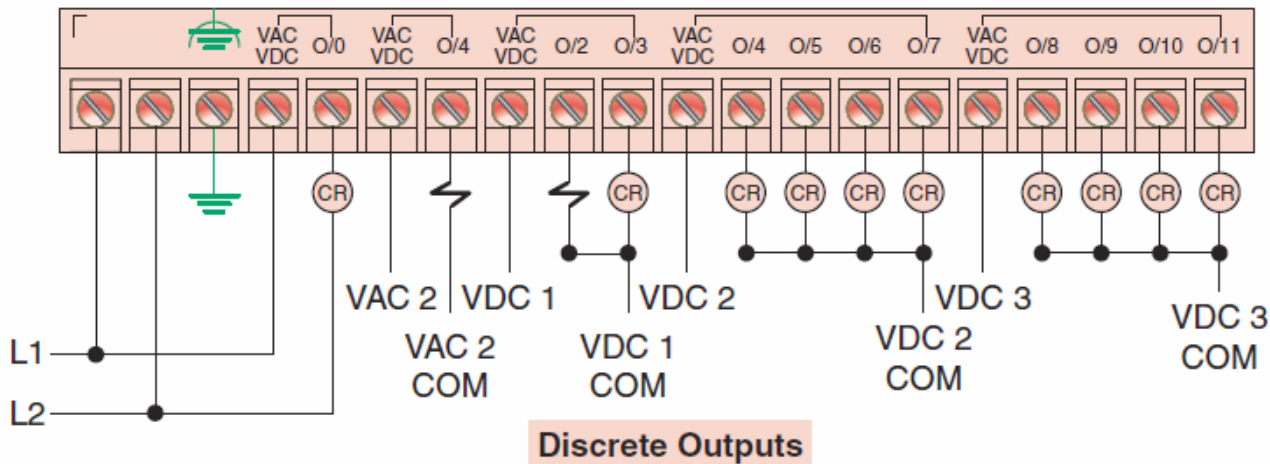
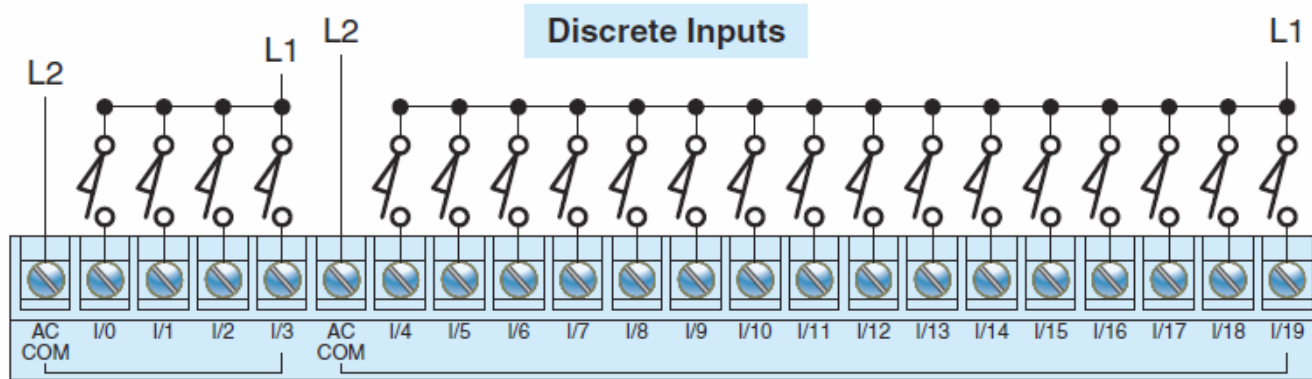
If the program calls for an output to be **OFF**, its corresponding bit in the table is set to **0**.



Simulated operation of the *output image* table on/off status of connected discrete outputs.



Typically, *micro PLCs* have a *fixed* number of inputs and outputs.



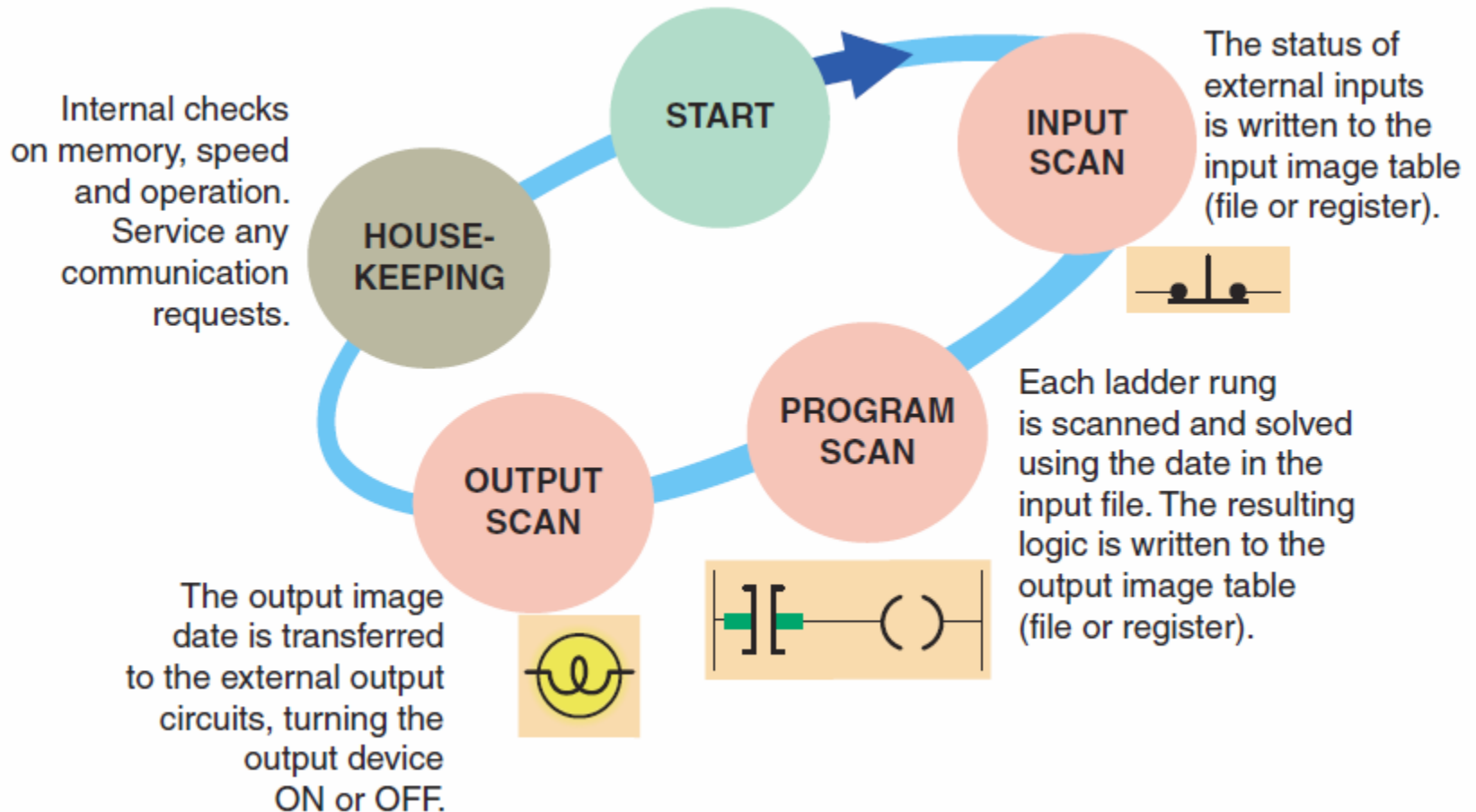
This controller has **20** discrete inputs with addresses **I/0** through **I/19** and **12** discrete outputs with addresses **O/1** through **O/11**.

5.2



Program Scan

During each *program scan cycle*, the processor reads all the inputs, takes these values, and energizes or de-energizes the outputs according to the user program.



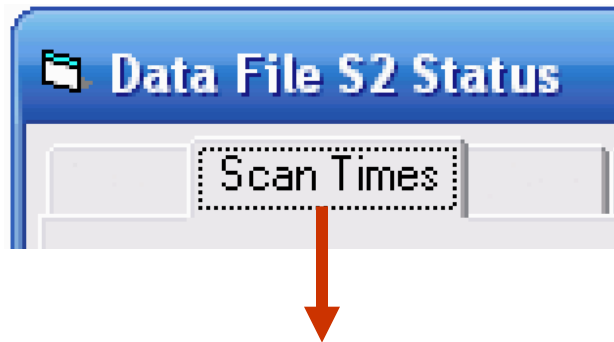
The *time* it takes to complete a scan cycle is a measure of how fast the controller can *react* to changes in inputs.



If a controller has to react to an input signal that changes states twice during the scan time, it is possible that the PLC will never be able to **detect** this change.

The scan time is a function of:

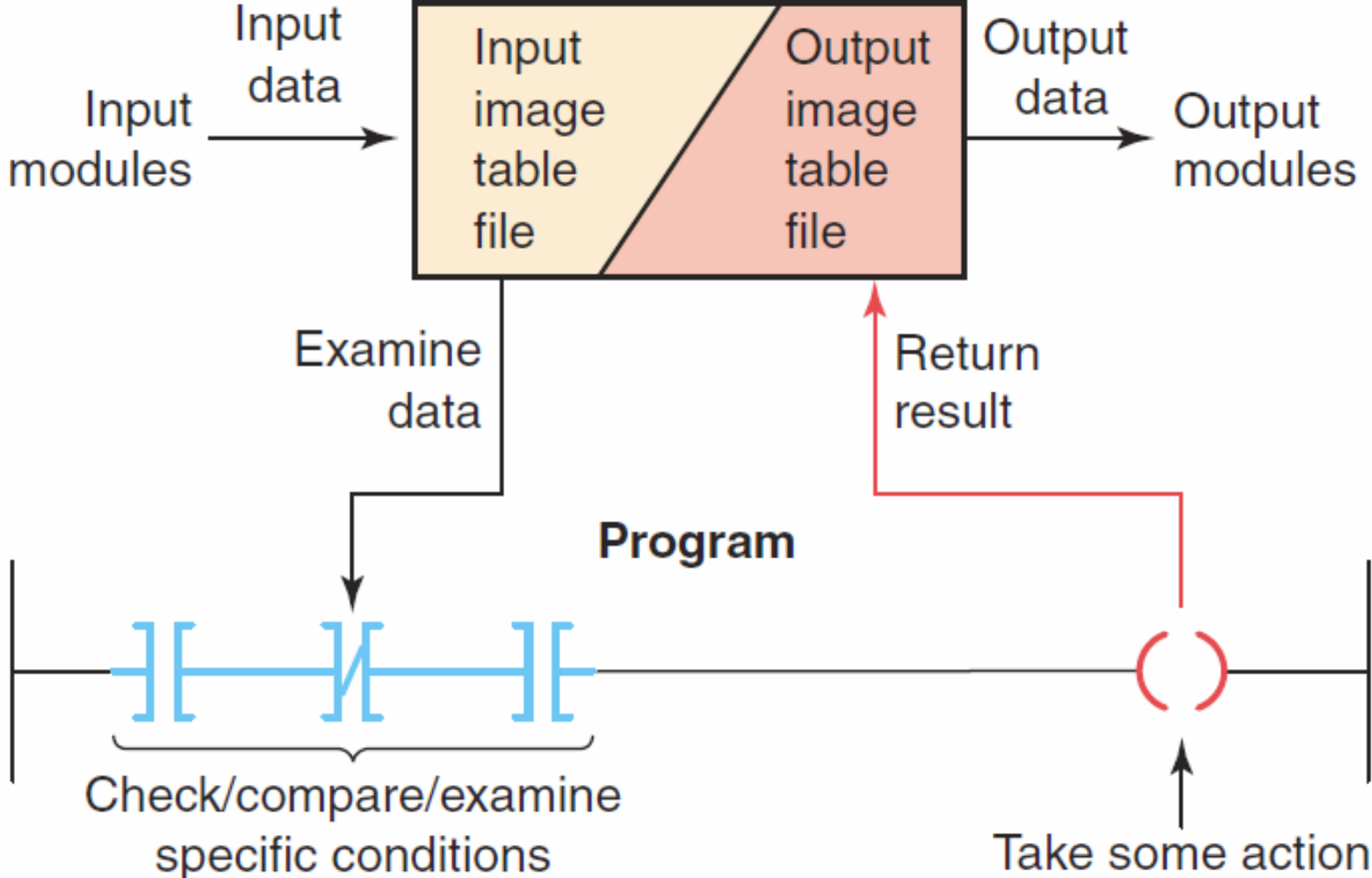
- The **speed** of the processor module
- The **length** of the ladder program
- The **type** of instructions executed
- The actual ladder **true/false conditions**



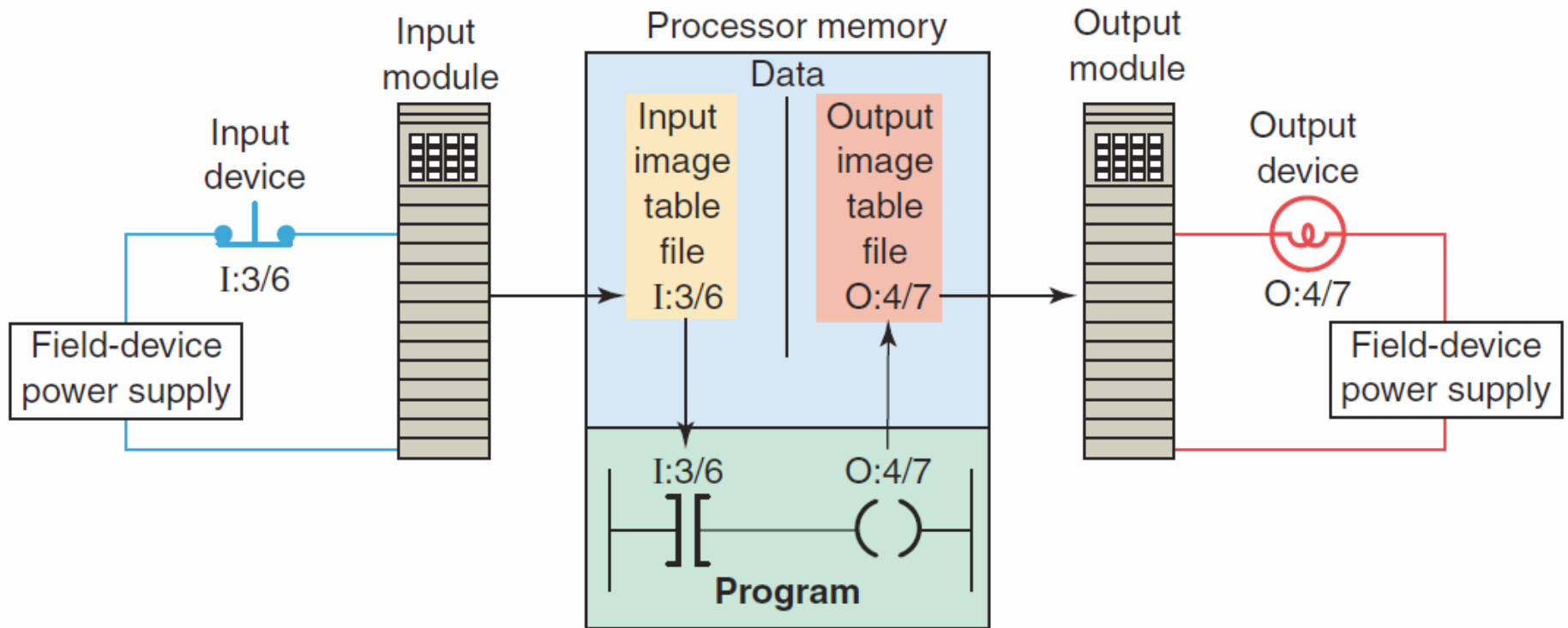
The PLC computes the scan time each time the **END** instruction is executed.

Typical scan time data include the **maximum** scan time and the **last** scan time.

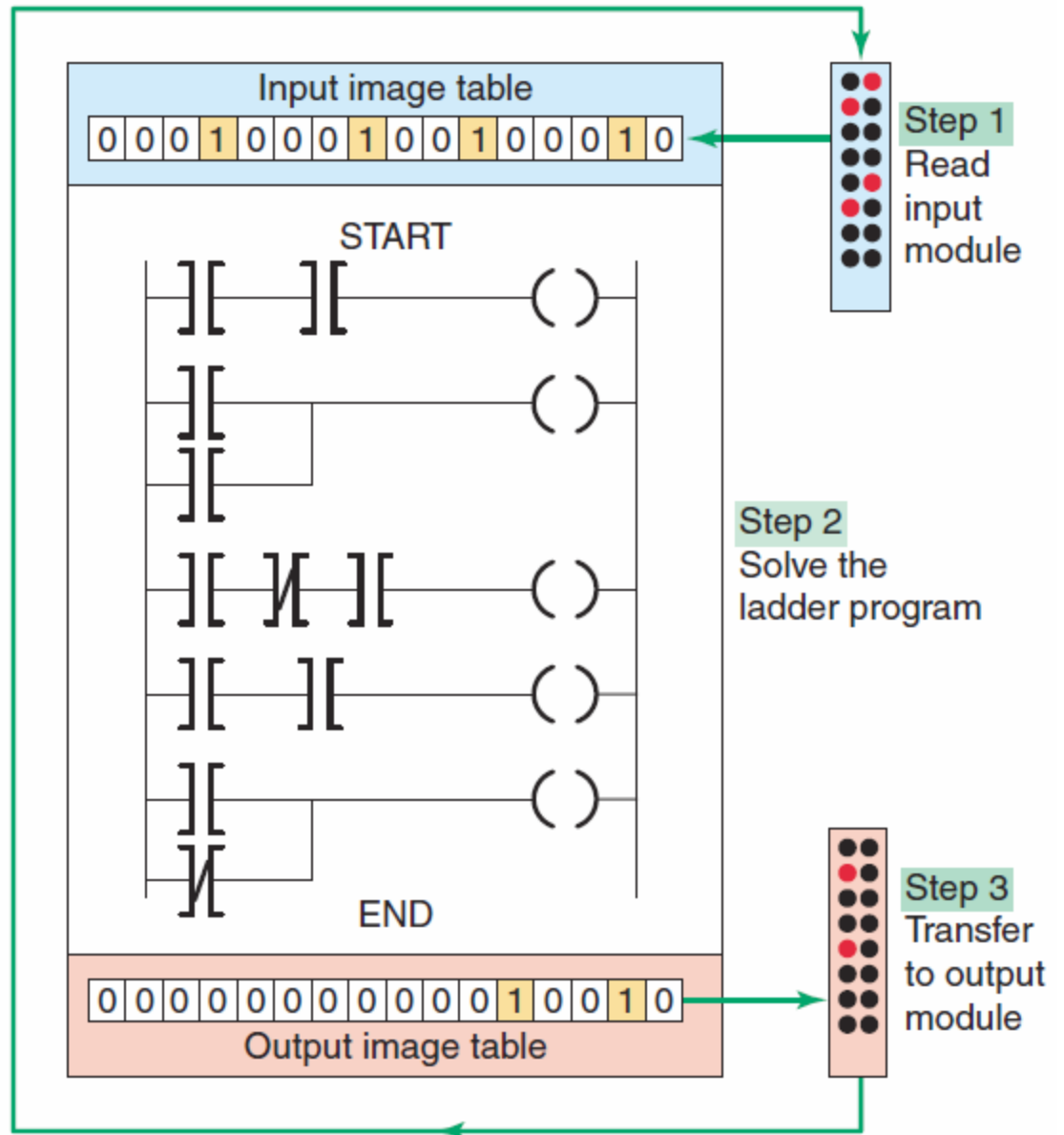
Overview of the data flow during the scan process.



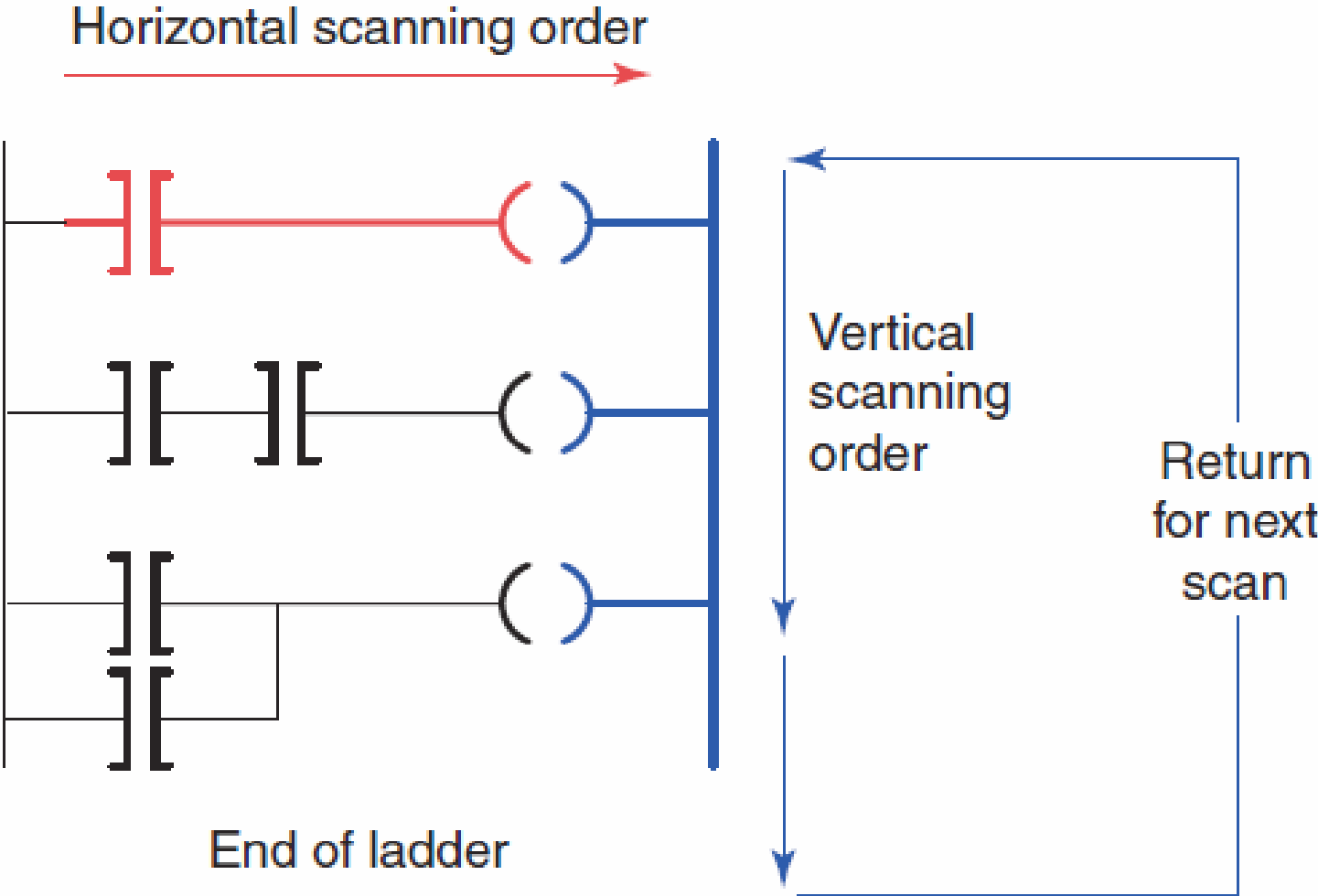
Scan process applied to a *single rung* program.



Scan process applied to a *multiple rung* program.



Vertical versus horizontal scan patterns.

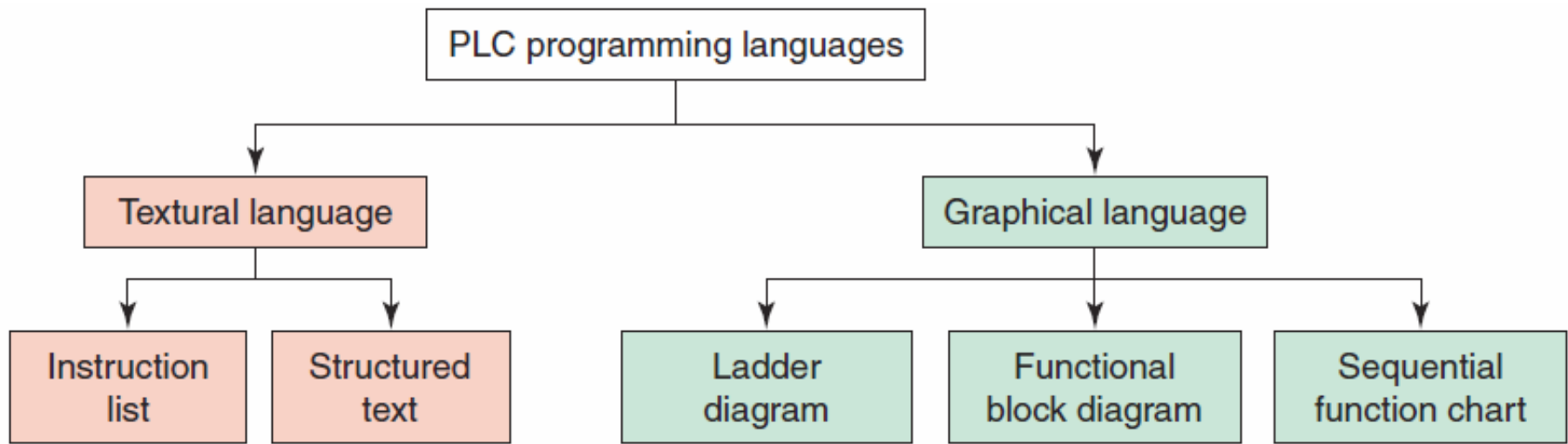


5.3



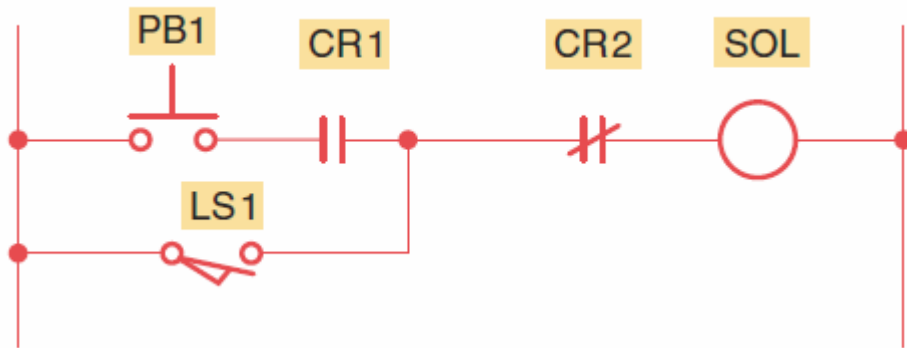
PLC Programming Languages

PLC programming language refers to the method by which the user communicates information to the PLC.

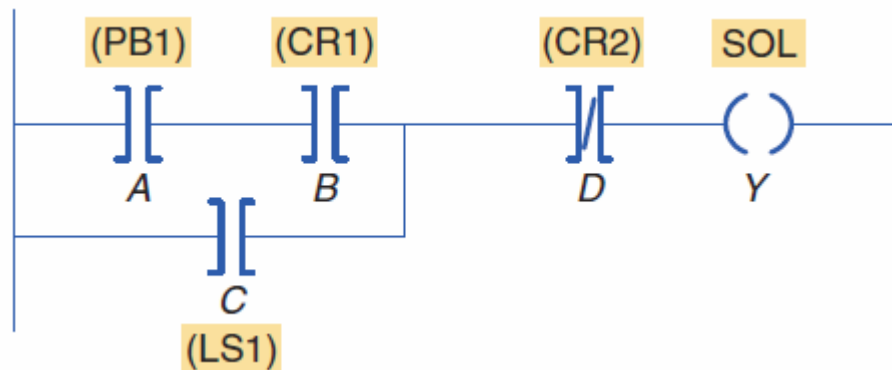


Standard PLC programming languages

Ladder diagram language is the most commonly used PLC language and is designed to mimic hardwired relay logic.

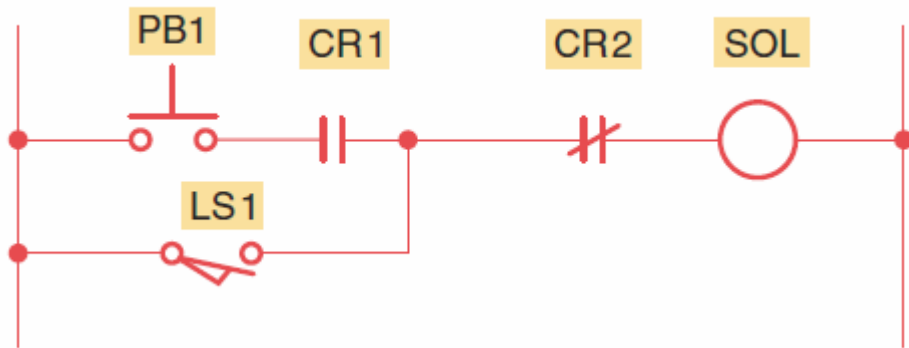


Hardwired **relay**
control circuit



Equivalent **ladder**
diagram program

Instruction list programming language consists of a series of instructions that refer to the basic *AND*, *OR*, and *NOT* logic gate functions.



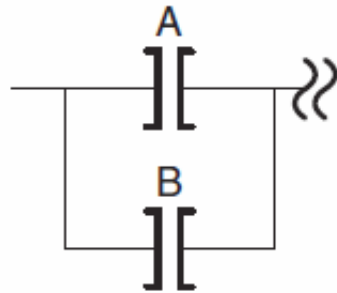
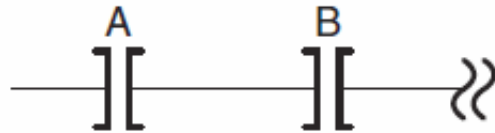
Hardwired **relay control circuit**

START	PB 1
AND	CR1
OR	LS1
AND NOT	CR2
OUT	SOL

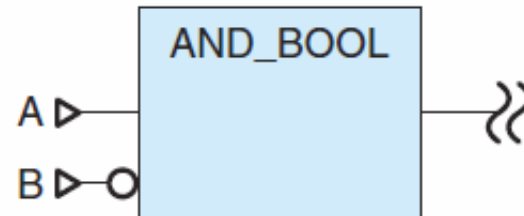
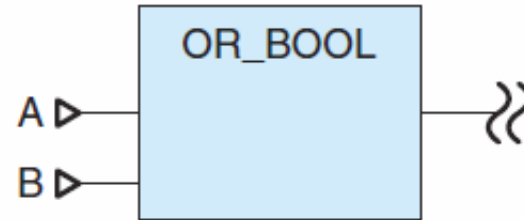
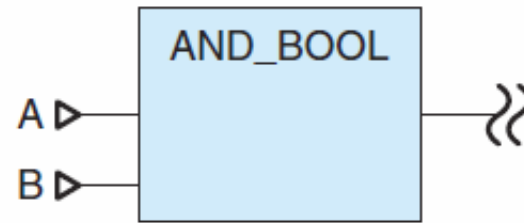
Equivalent **instruction list program**

Functional block diagram programming uses instructions that are programmed as blocks wired together to accomplish certain functions.

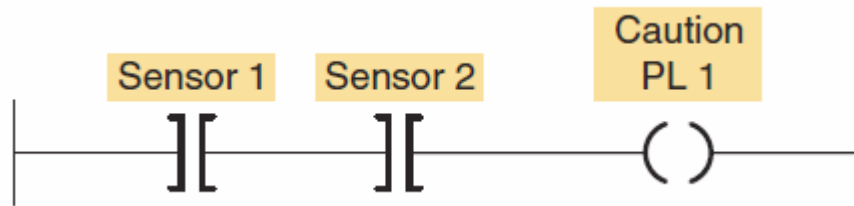
Ladder logic



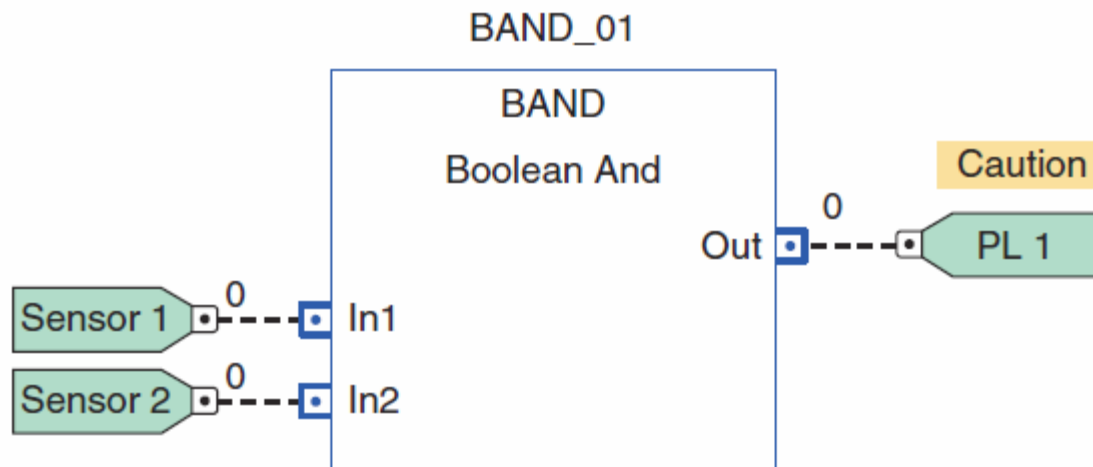
Functional block diagram equivalent



Ladder diagram and functional block diagram programming used to produce the same logical output.



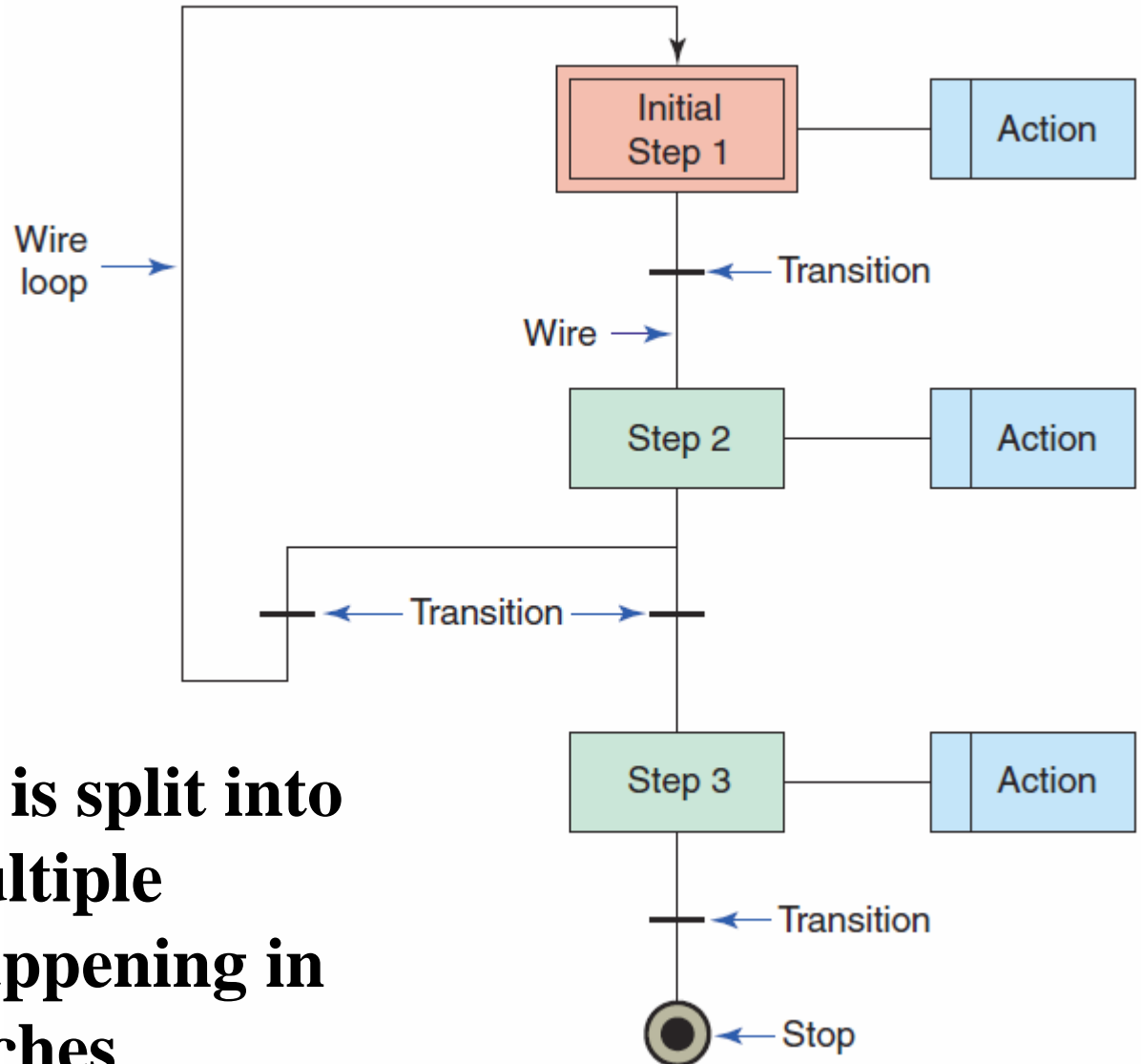
Ladder diagram



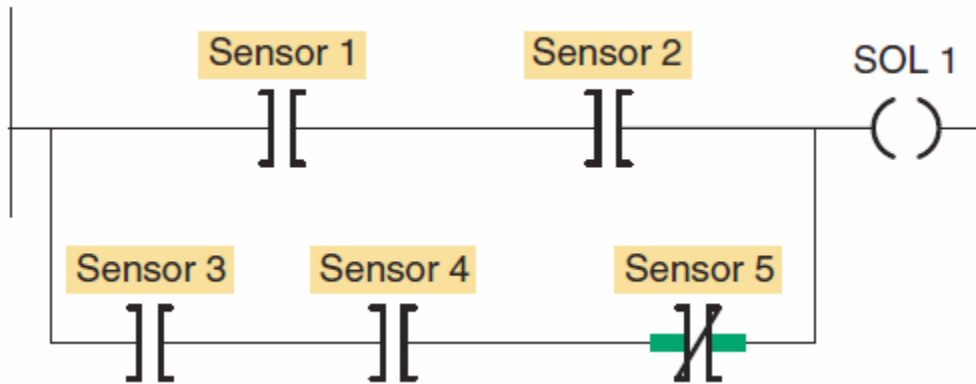
Equivalent function block diagram.

*Sequential
function chart
programming
language is
similar to a
flowchart of
your process.*

The program is split into **steps** with multiple operations happening in parallel branches.



Structured text is a high level language primarily used to implement more *complex* procedures.



Ladder diagram

```
IF Sensor_1 AND Sensor_2 THEN
    SOL_1 := 1;
ELSEIF Sensor_3 AND Sensor_4 AND NOT Sensor_5 THEN
    SOL_1 := 1;
END_IF;
```

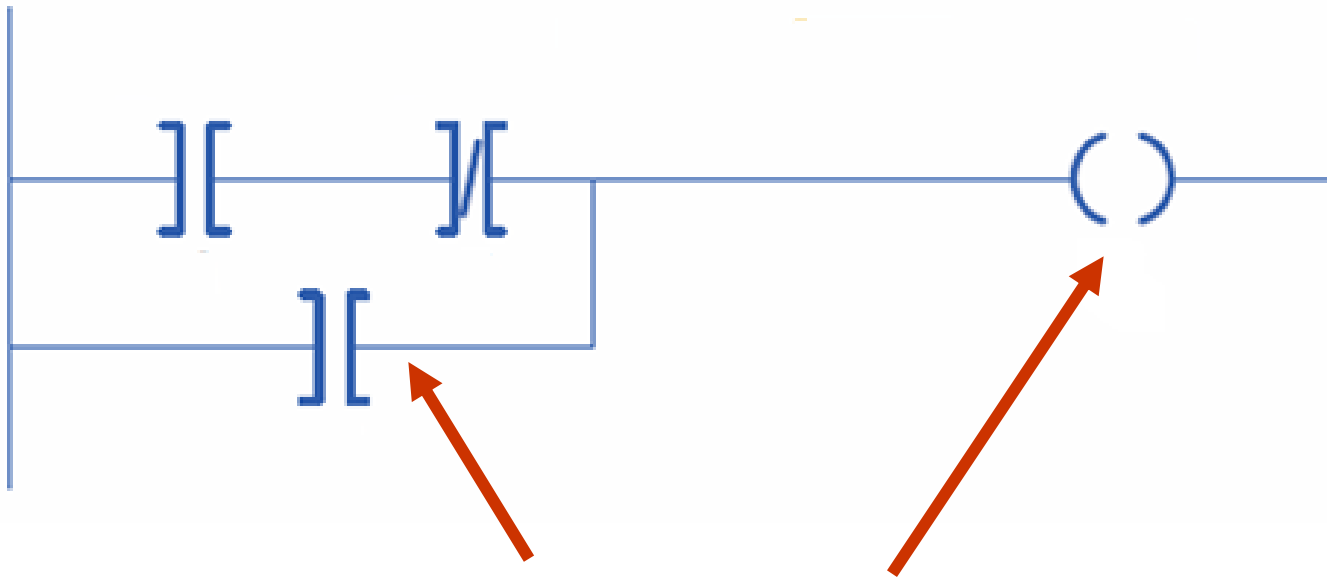
**Equivalent
structured text
program.**

5.4



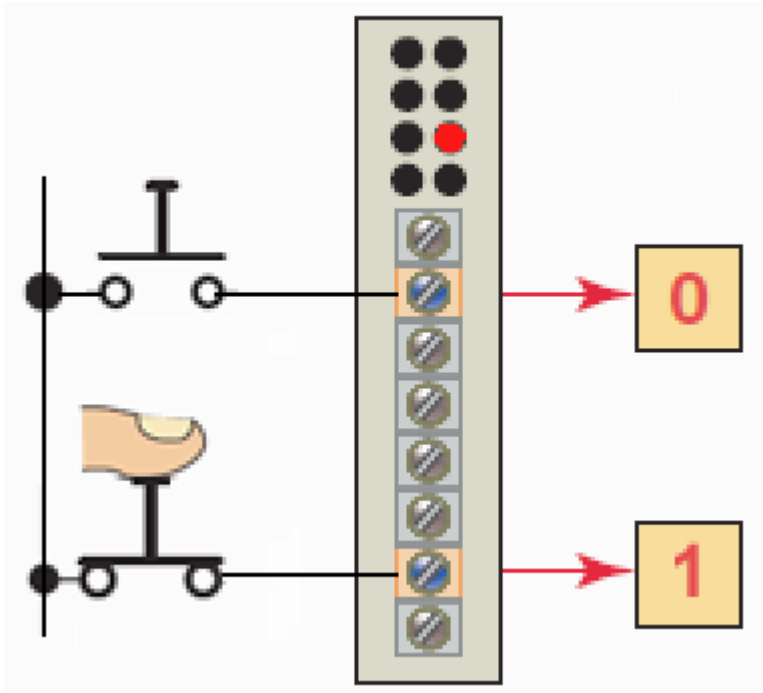
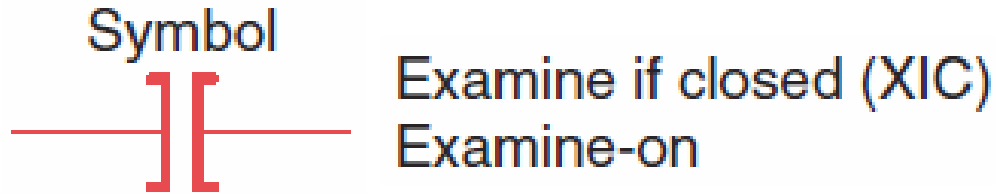
Relay Type Instructions

The ladder diagram language is a *symbolic* set of instructions used to create the controller program.



Representations of **contacts** and **coils** are the basic symbols of the logic ladder diagram instruction set.

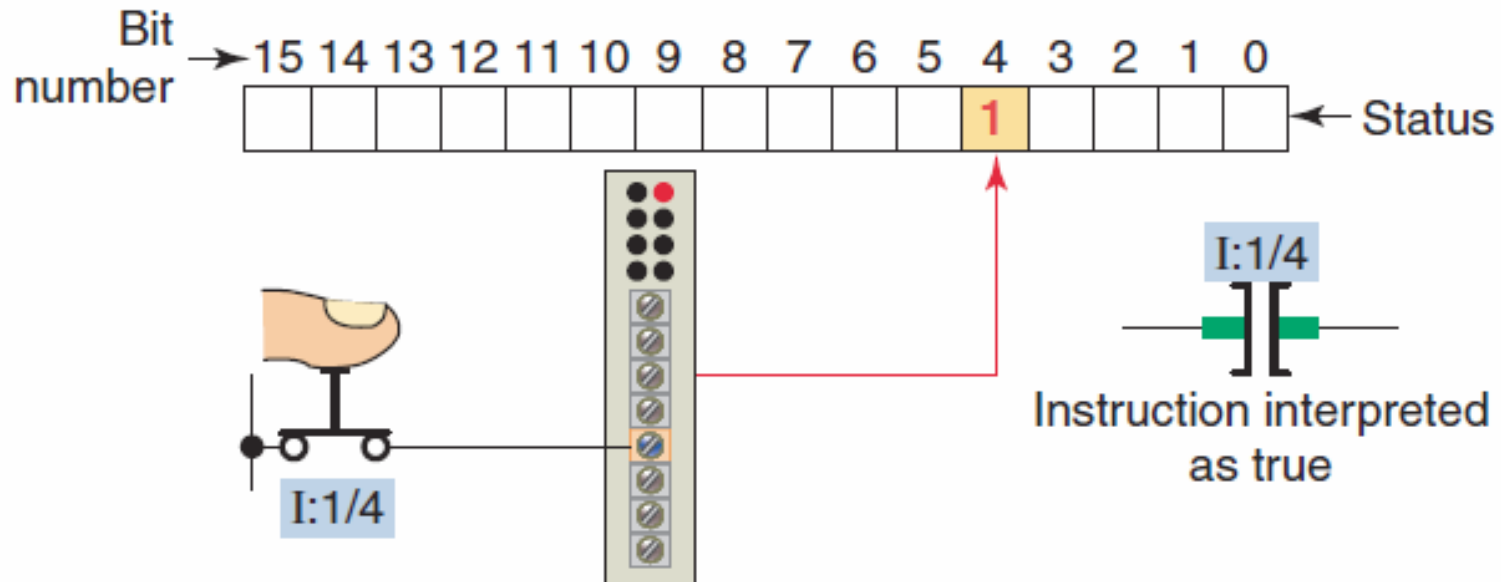
The *Examine If Closed (XIC)* instruction looks and operates like a *normally open* relay contact.



Associated with each XIC instruction is a **memory bit** linked to the status of an input device or an internal logical condition in a rung.

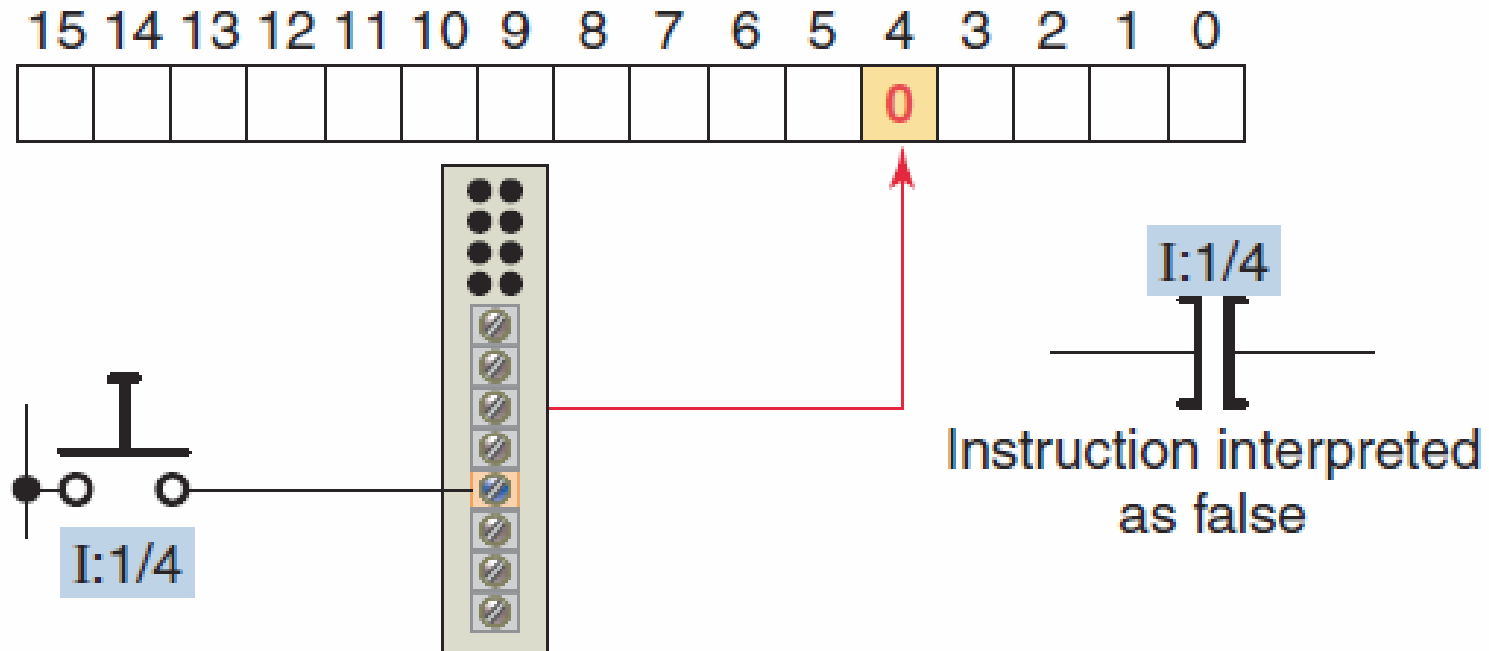
The memory bit is set to *1 or 0* depending on the status of the input.

A **1** corresponds to a **true** status or **on** condition.



If the instruction memory bit is a **1 (true)** this instruction will **allow rung continuity** through itself, like a closed relay contact.

A **0** corresponds to a **false** status or **off** condition.



If the instruction memory bit is a **0 (false)** this instruction will **not allow rung continuity** through itself and will assume a normally open state just like an open relay contact.

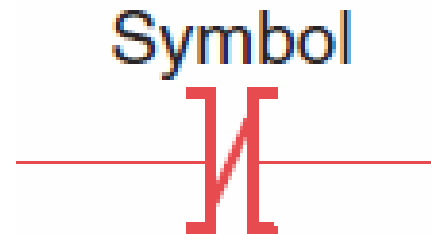
Simulated *Examine If Closed (XIC)* instruction operation.

The screenshot displays a PLC simulation environment. On the left is the I/O Simulator, showing a rack of 10 digital input modules (I:1 to I:9) and one digital output module (O:2). Each module has a switch and a corresponding indicator light. The LAD 2 window shows a logic network with a normally open contact labeled '000' and a normally closed contact labeled 'O:2/0'. An 'Input Table' dialog box is open, showing a table of input states for modules I:0/ through I:5/ across 16 bits (15 to 0). The table shows that all inputs are currently 0. The 'Radix' is set to 'Binary', the 'Table' is 'I1: Input', and there is a 'Forces' button.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I:0/																
I:1/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I:2/																
I:3/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I:4/																
I:5/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The *Examine If Open (XIO)* instruction looks and operates like a *normally closed* relay contact.

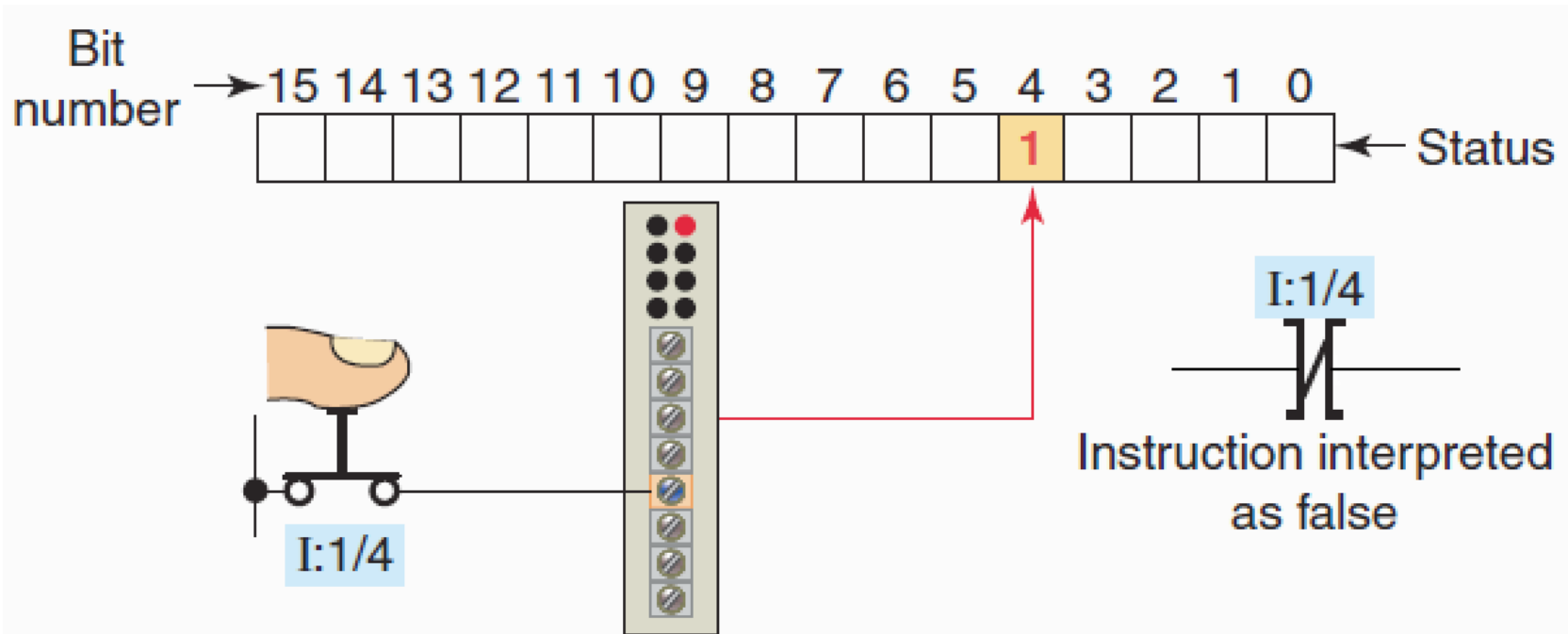
Examine if open (XIO)
Examine-off



This instruction asks the PLC's processor to examine if the contact is **open**.

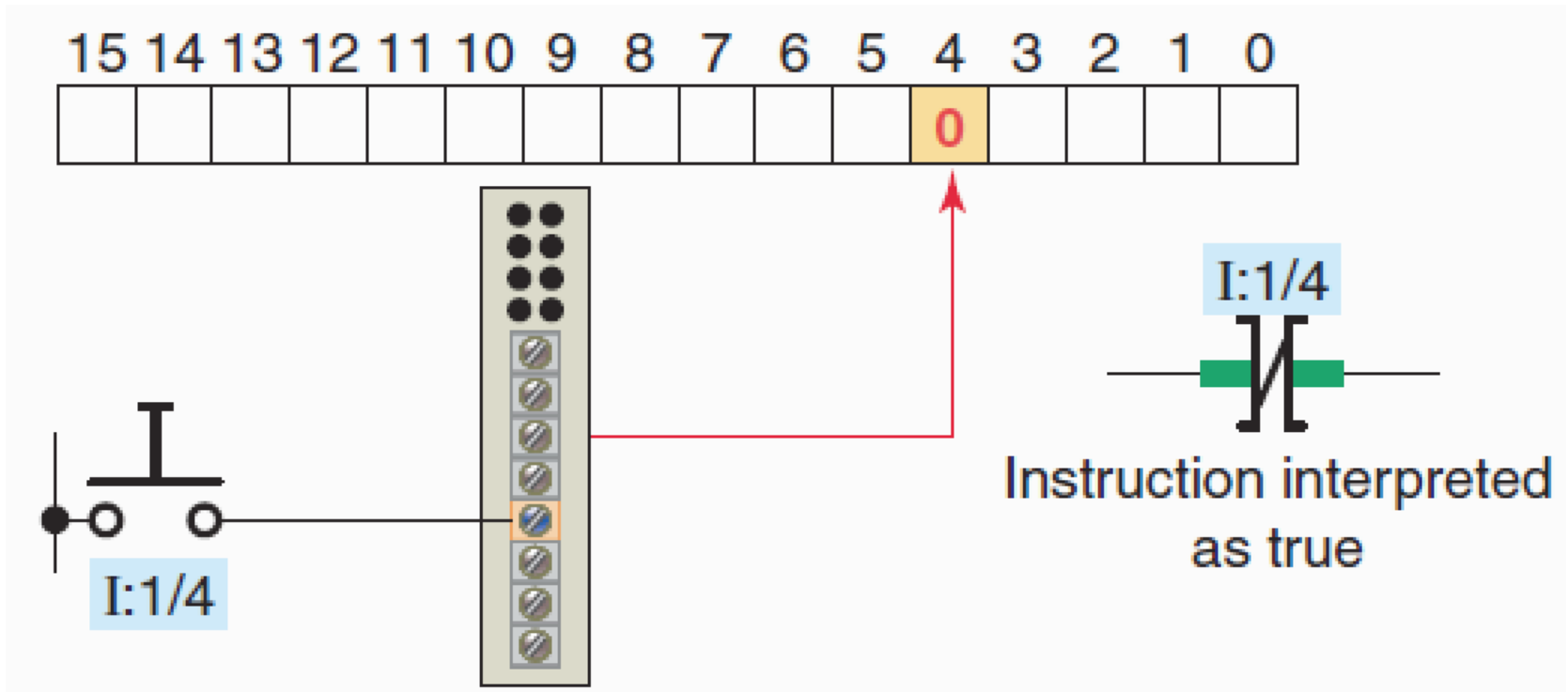
It does this by examining the **bit** at the memory location specified by the address for a **0 or 1**.

As with any other input the memory bit is set to 1 or 0 depending on the status of the input. A **1** corresponds to a true status or **on condition**.



The instruction is interpreted as **false** when the bit is **1** and **will not** allow **rung continuity** through itself.

A **0** corresponds to a **off** condition.



The instruction is interpreted as **true** when the bit is **0** and **will not allow rung continuity** through itself.

Simulated *Examine If Open (XIO)* instruction operation.

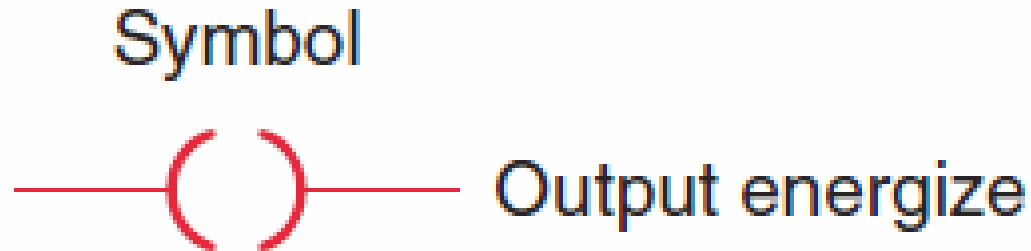
The screenshot displays a simulated PLC environment. On the left, the 'I/O Simulator' shows a vertical stack of input and output modules. Input I:1/4 is shown as a closed switch, while output O:2/0 is shown as an energized lamp. The main window shows a Ladder Logic (LAD) diagram for 'LAD 2' with a normally closed contact labeled 'I:1/4' connected in series to a coil labeled 'O:2/0'. An 'Input Table' window is overlaid on the diagram, showing a grid of input states for addresses I:0/ through I:5/ across bit positions 0 to 15. The table shows that only bit 4 of input I:1/ is set to 1, while all other bits are 0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I:0/																
I:1/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I:2/																
I:3/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I:4/																
I:5/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Radix: Binary Table: I1: Input Forces

Address Symbol

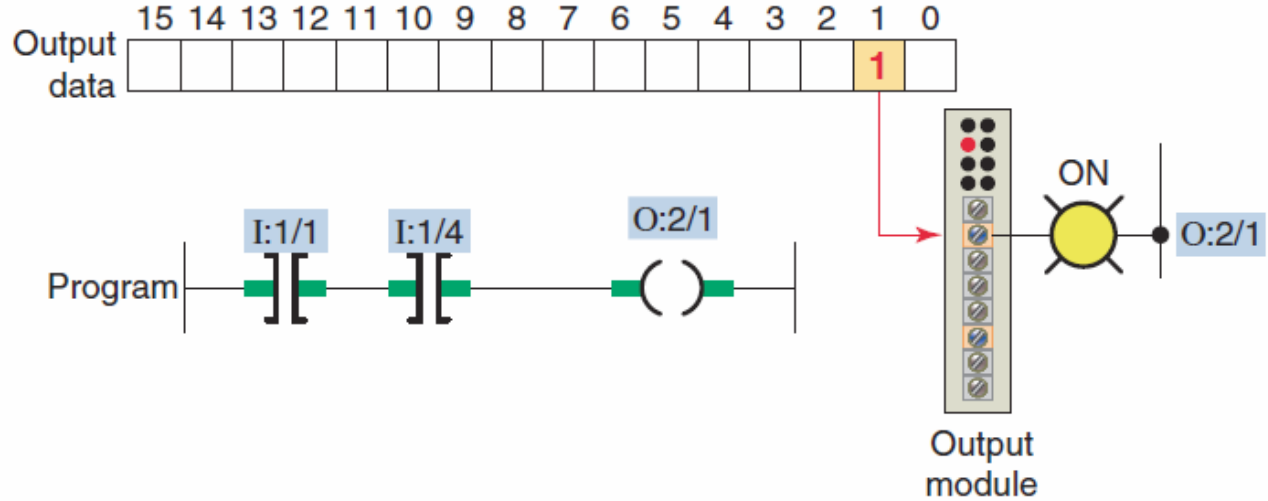
The *Output Energize (OTE)* instruction looks and operates like a *relay coil*.



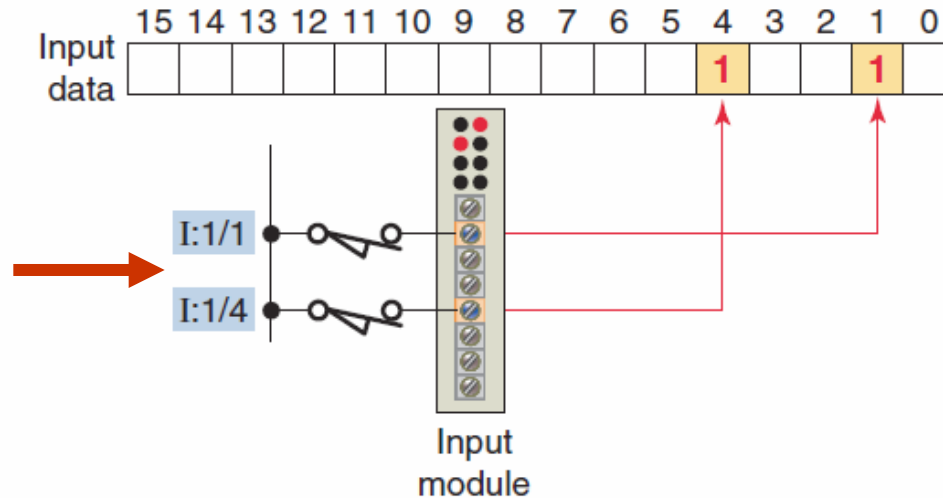
This instruction signals the PLC to **energize** (switch on) or **de-energize** (switch off) the output.

The instruction is associated with a memory bit that **energizes** the output when set to **1** and **de-energizes** the output when reset to **0**.

OTE instruction
is set to **1** to
energize the
output.



A true logic
path is
established by
the input
instructions in
the rung.

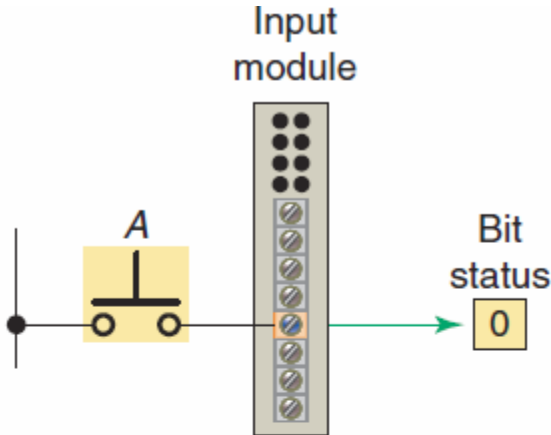


Simulated *Output Energize (OTE)* instruction operation.

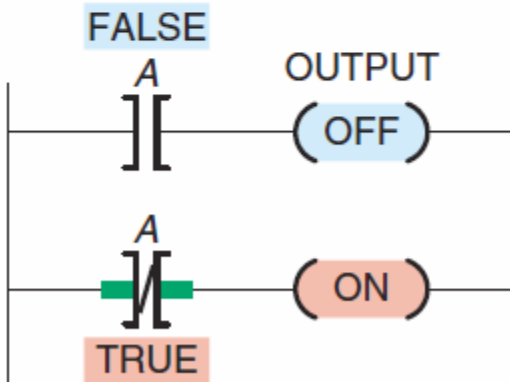
The screenshot displays a PLC simulation environment. On the left, the I/O Simulator shows a vertical array of 10 input and 10 output modules. Input 01 is highlighted in red and is in the 'ON' position. Output 01 is highlighted in green and is in the 'ON' position. The main window shows a Ladder Logic (LAD) diagram for network 000. It consists of two normally open contacts labeled I:1/1 and I:1/4 in series, connected to an Output Energize (OTE) instruction labeled O:2/1. An 'Output Table' window is overlaid on the LAD diagram, showing the current state of outputs 0:0/ through 0:5/. The table has 16 columns representing bits 15 down to 0. The row for output 0:2/ shows a '1' in the bit 1 position and '0's elsewhere. The 'Forces' button is visible in the bottom right of the Output Table window.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0:0/																
0:1/																
0:2/															1	
0:3/																
0:4/																
0:5/																

Action of the *field device* and PLC *bit*.

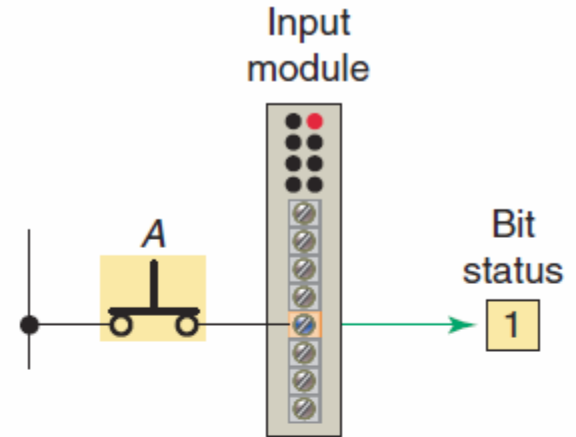


Ladder logic program

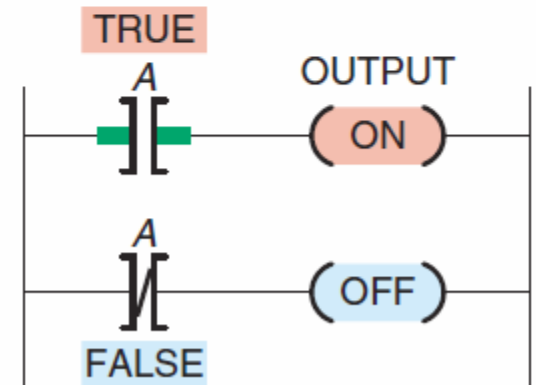


Button not actuated

A signal present makes the NO bit (1) true; a signal absent makes the NO bit (0) false. The reverse is true for an NC bit.



Ladder logic program



Button actuated

Simulated operation of the field input device and the PLC bit.

The screenshot displays a PLC simulation environment with three main components:

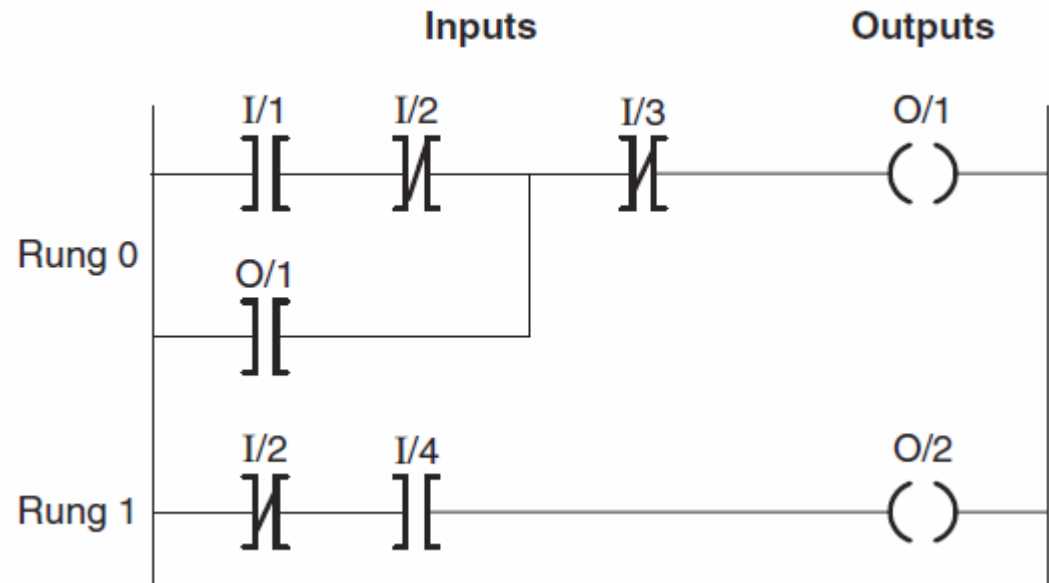
- I/O Simulation:** A vertical panel on the left showing 12 input points (I:1 to I:11) and 12 output points (O:1 to O:11). Each point has a switch icon and a status indicator. Input 01 is shown as closed (green light), and output 01 is shown as energized (red light).
- Ladder Logic (LAD 2):** A logic diagram with two rungs. Rung 000 contains a normally open contact labeled 'I:1/0' (with a yellow highlight) connected to a coil labeled 'O:2/0'. Rung 001 contains a normally closed contact labeled 'I:1/0' (with a yellow highlight) connected to a coil labeled 'O:2/1' (with a yellow highlight). Both contacts are labeled 'A' in an orange box.
- Input Table:** A table window showing the current state of inputs. The table has 16 columns (0-15) and 6 rows (I:0/ to I:5/). The 'I:1/' row shows a '0' in column 1, indicating that input 1 is active.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I:0/																
I:1/															0	
I:2/																
I:3/																
I:4/																
I:5/																

The main function of the ladder logic diagram program is to *control outputs* based on *input conditions*.

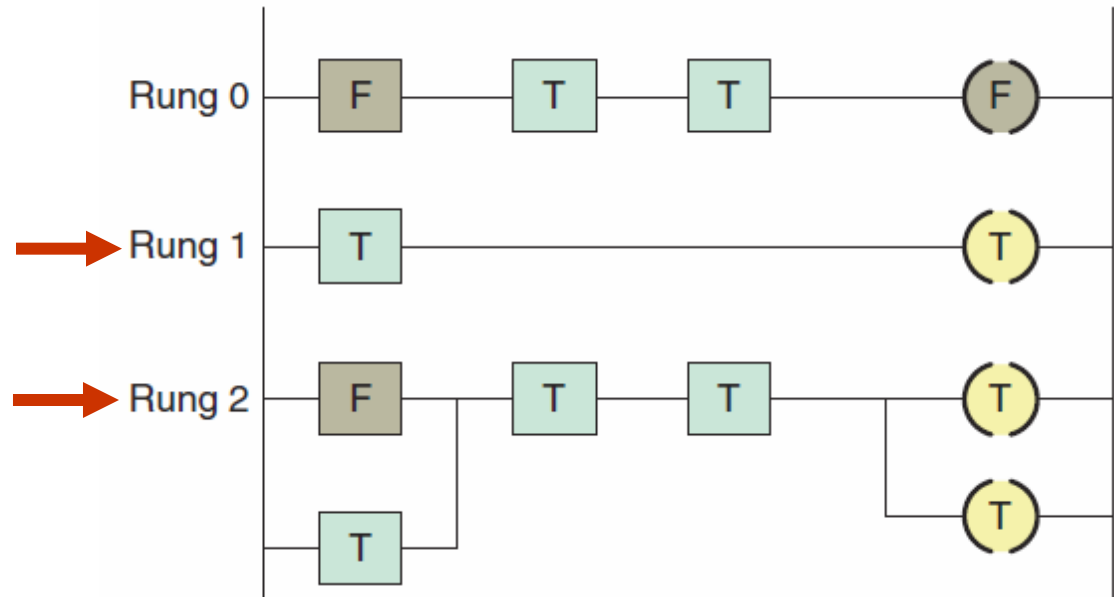
Each contact or coil symbol is referenced with an **address** that identifies what is being **evaluated** and what is being **controlled**.

The **same contact** instruction can be used **throughout** the program whenever that condition needs to be evaluated.



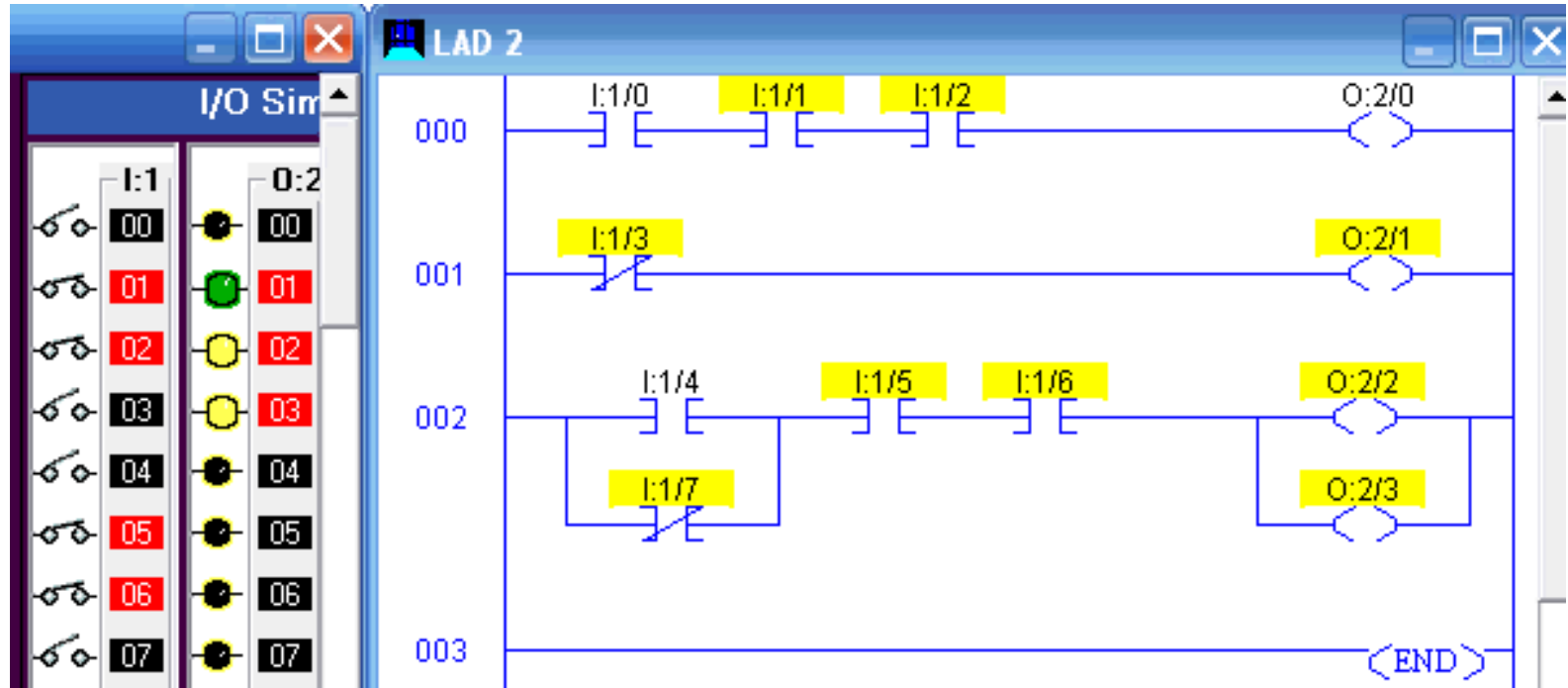
For an output to be activated or energized, at least one left-to-right *true logical path* must exist.

A complete closed path is referred to as having **logical continuity**.



When logical continuity exists in at least one path, the rung condition and Output Energize instruction are said to be **true**.

Simulated operation of logic continuity.



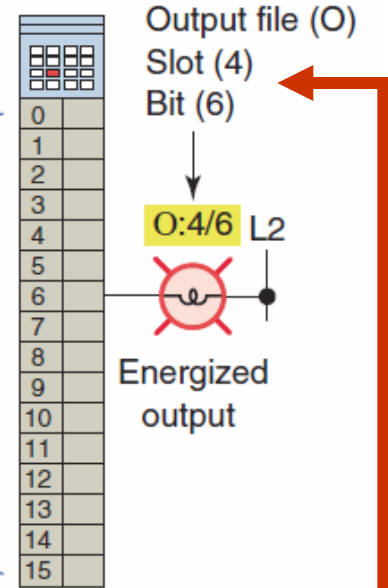
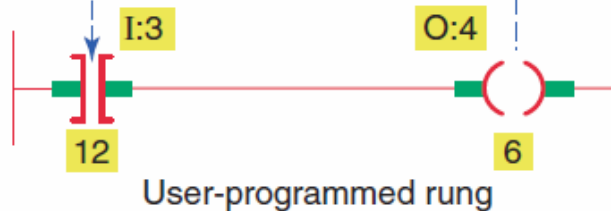
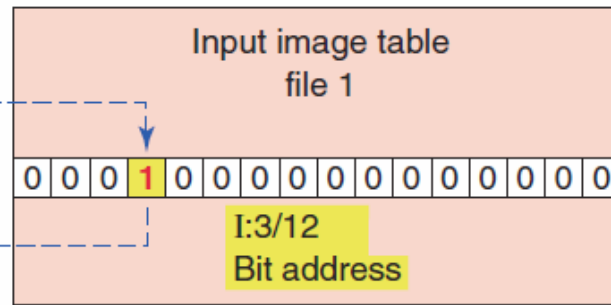
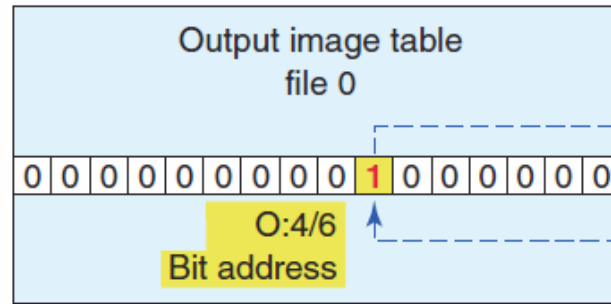
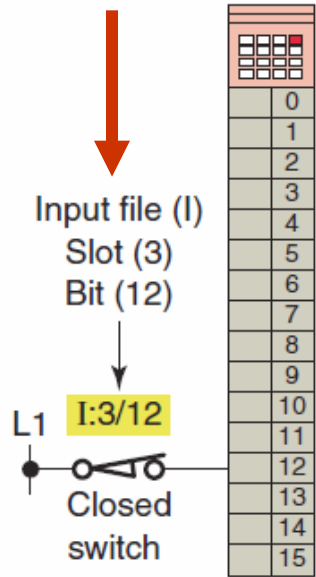
5.5



Instruction Addressing

To complete the entry of a relay-type instruction, you must assign an *address* to each instruction.

Address indicates what input is connected to what input device



Address indicates what output is connected to what output device

Simulated operation of instruction addressing.

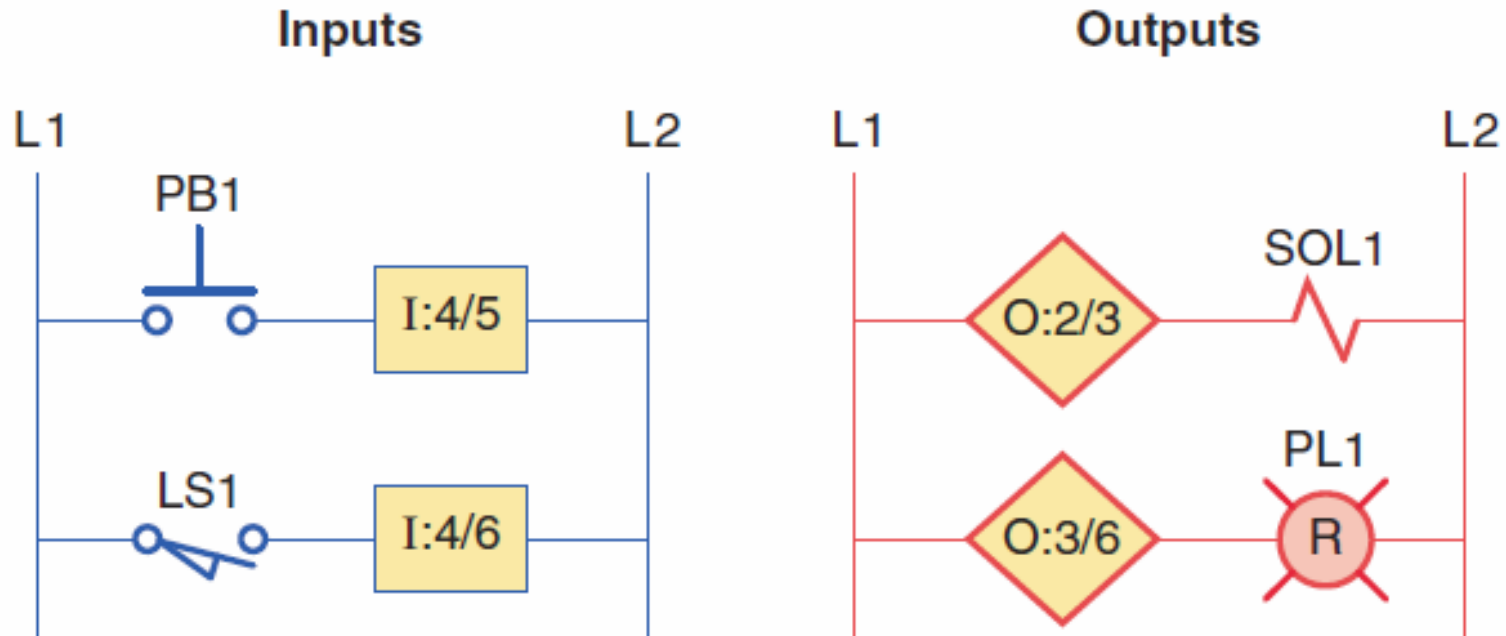
The screenshot displays a PLC simulation environment. At the top, the PLC status is 'OnLine' with a 'Download' button. The mode is set to 'RUN', and the scan counter shows '0 Scans'. Below this is a rack of PLC modules. The main window is titled 'LAD 2' and shows a ladder logic program with two rungs:

- Run 000: A normally open contact with a question mark is connected to a coil with a question mark.
- Run 001: An 'END' instruction.

Below the LAD editor is the 'I/O Simulator II' window, which contains four columns of digital I/O indicators:

- Column 1: Inputs I:1 (00-13), all switches are open.
- Column 2: Outputs O:2 (00-13), all LEDs are lit.
- Column 3: Inputs I:3 (00-13), all switches are open.
- Column 4: Outputs O:4 (00-13), all LEDs are lit.

The assignment of an I/O address can be included in the *I/O connection* diagram.



Inputs and outputs are typically represented by squares and diamonds, respectively.

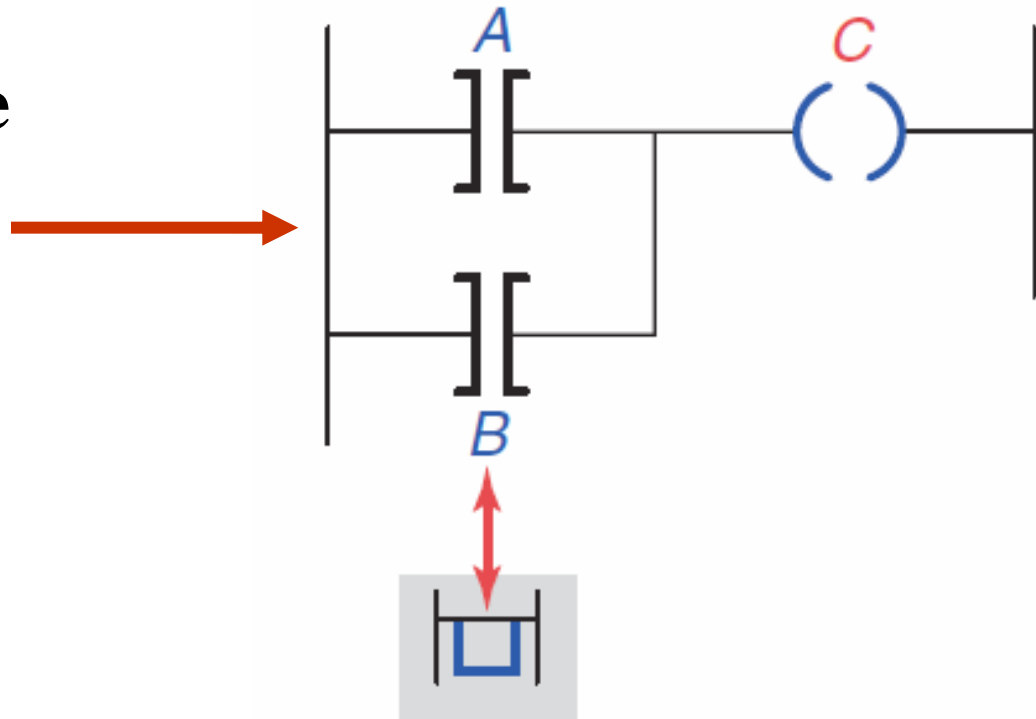
5.6



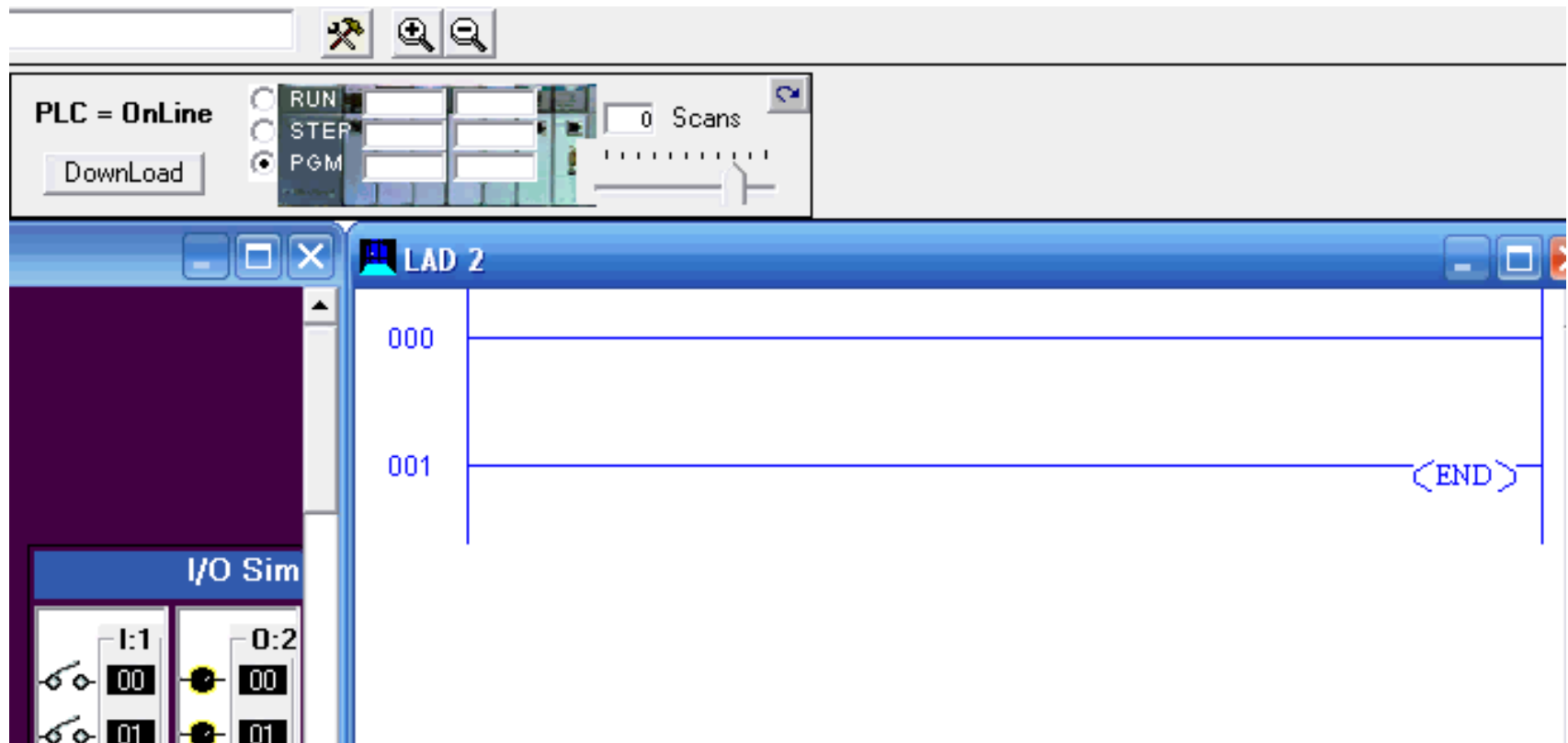
Branch Instructions

Branch instructions are used to create *parallel* paths of input condition instructions.

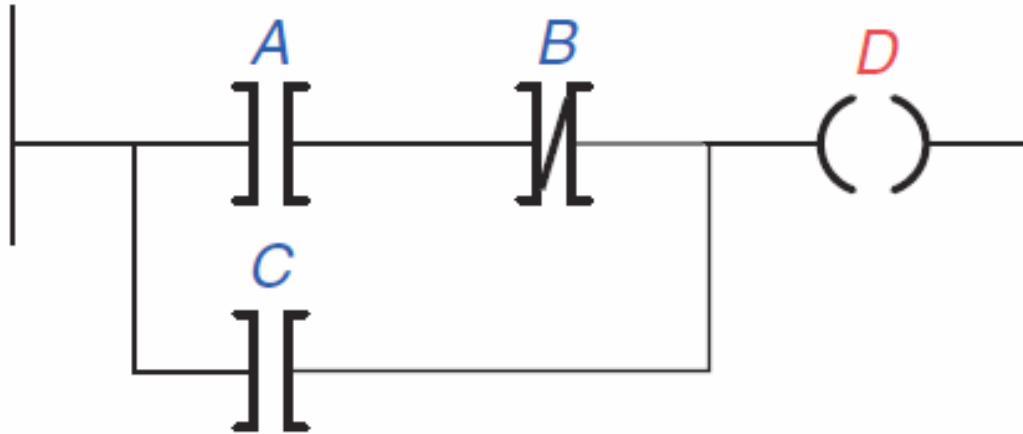
The rung will be true if either instruction **A** or **B** is true.



Simulated branch instructions.

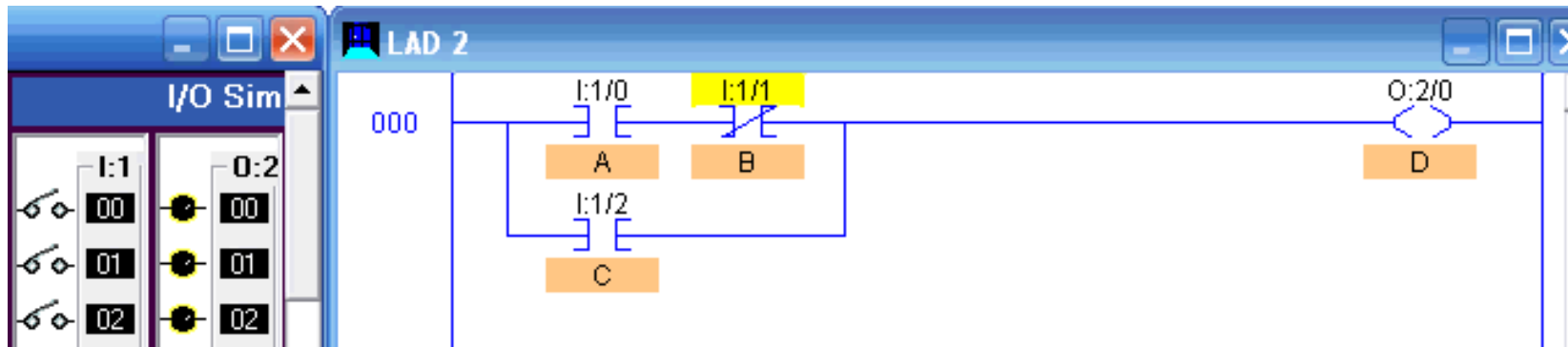


Parallel branches can be used to allow more than one *combination* of input conditions.



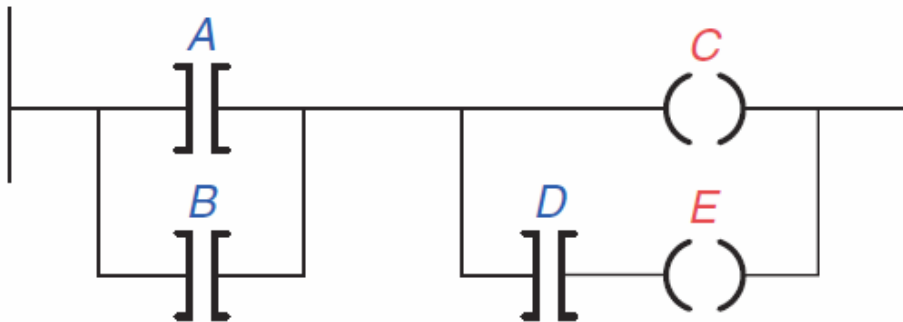
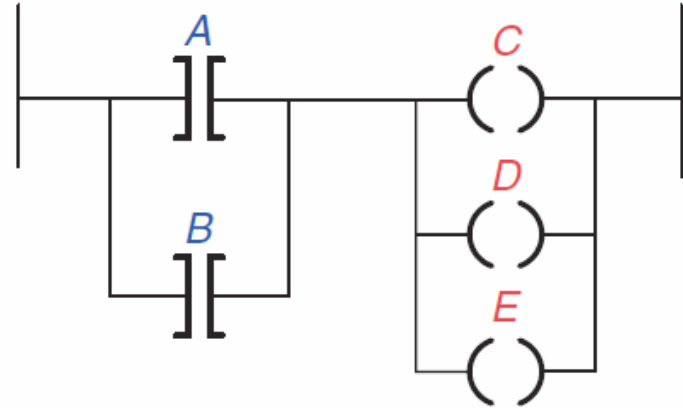
Either *A* **and not *B*, **or** *C* provides logical continuity and energizes output *D*.**

Simulated program, either A **and not B, **or** C provides logical continuity and energizes output D.**



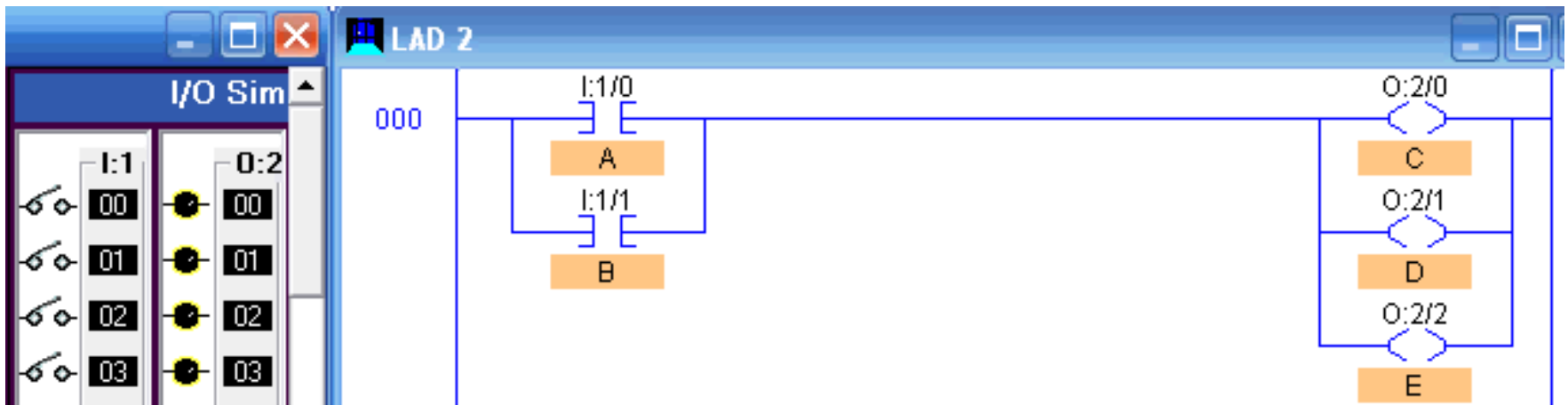
Output branching allows a true logic path to control *multiple* outputs

Either *A* or *B* provides a true logical path to **all three** output instructions: *C*, *D*, and *E*.

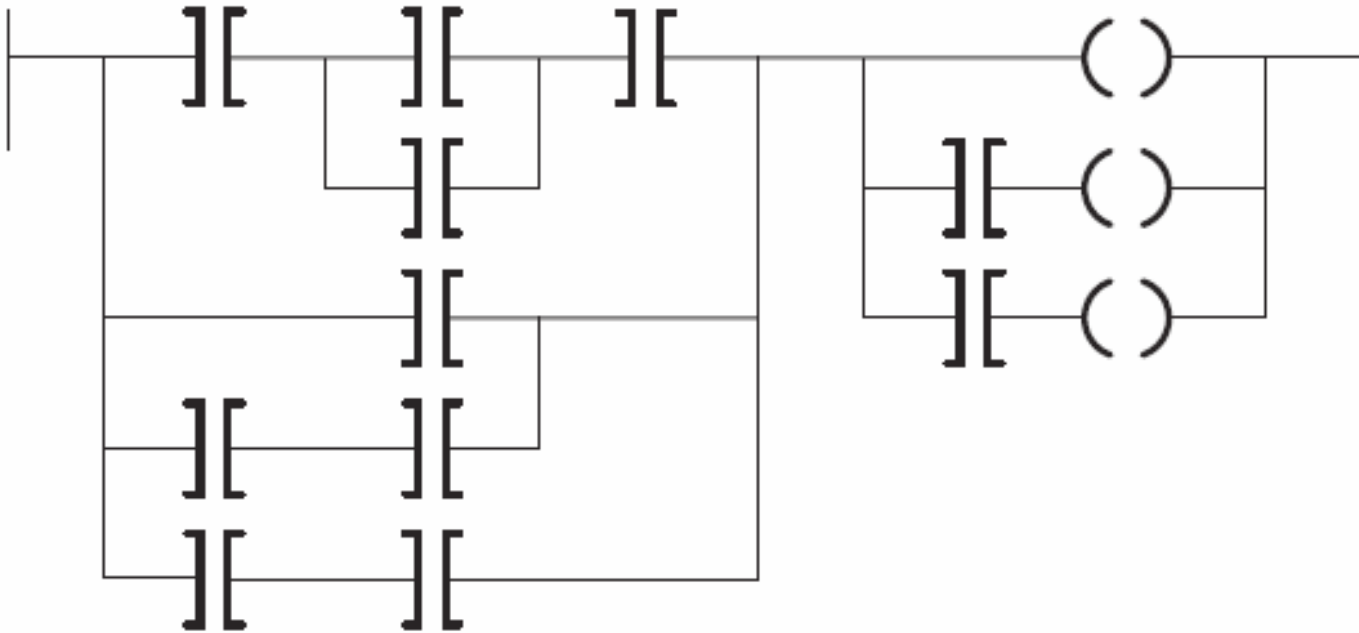


Additional input instructions can be programmed in the output branches.

Simulated program, either *A* or *B* provides a true logical path to **all three** output instructions: *C*, *D*, and *E*.

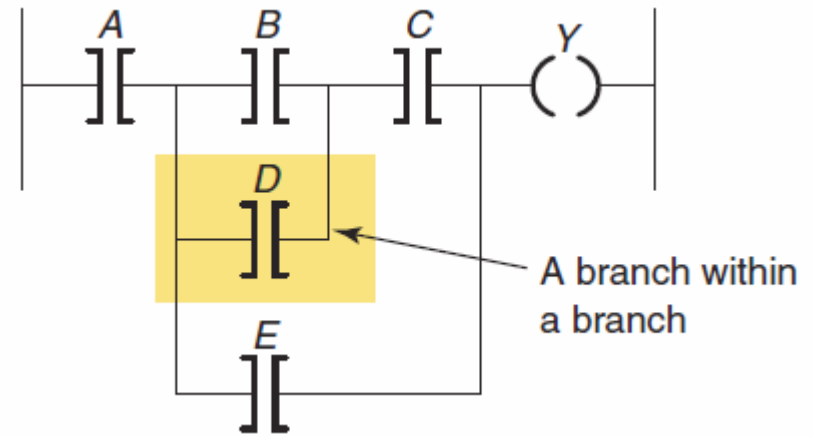


Input and output branches can be *nested* to avoid redundant instructions and to speed up processor scan time.

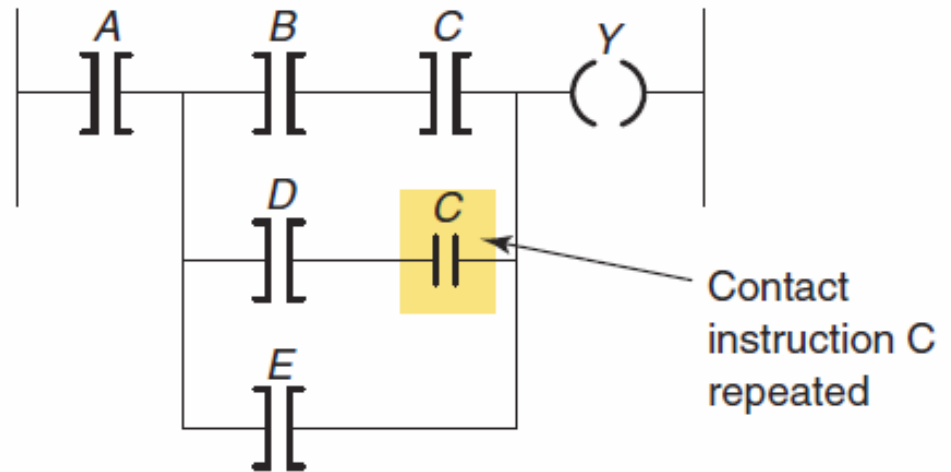


A **nested branch starts or ends within another branch.**

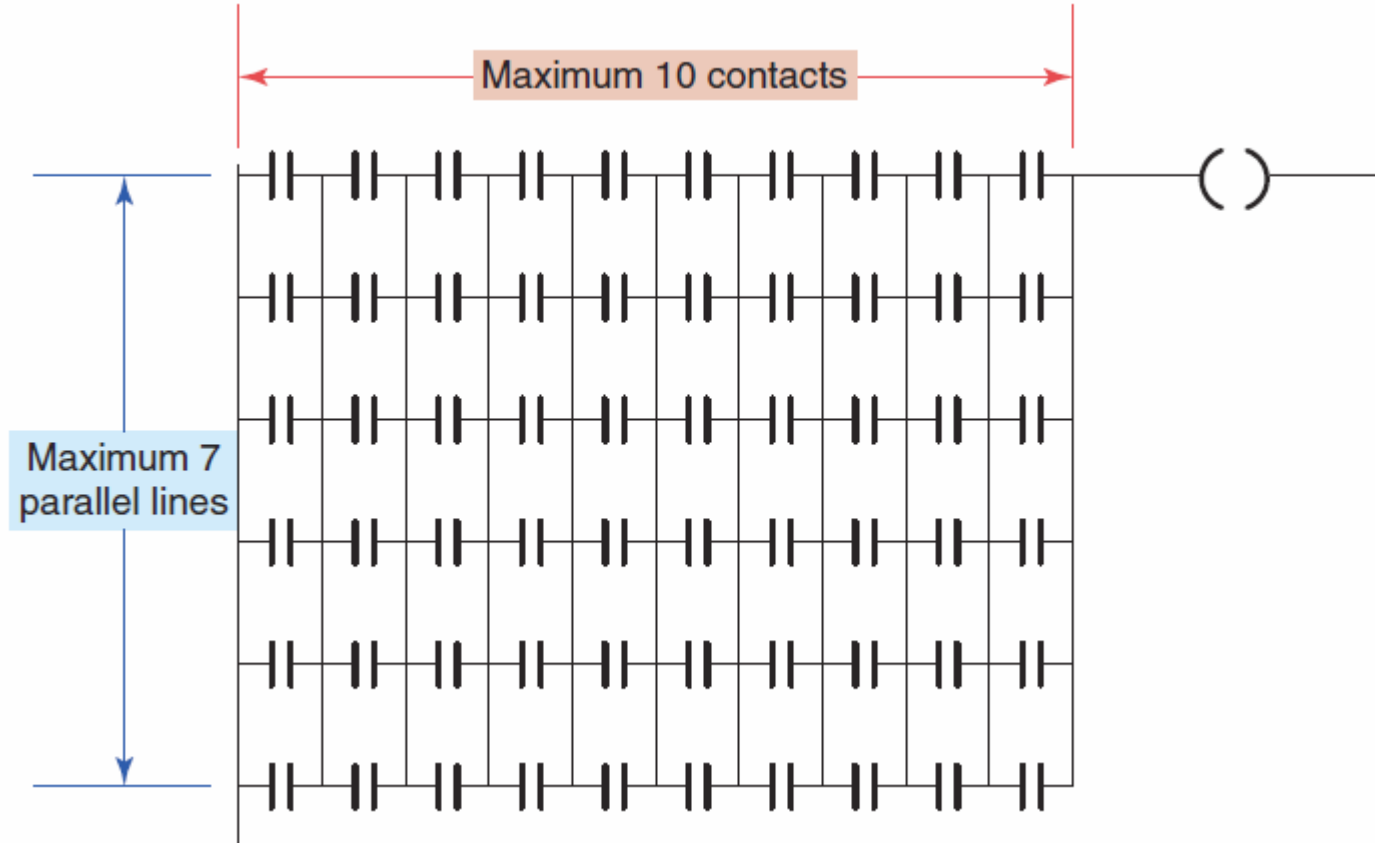
In some PLC models, the programming of a nested branch **cannot** be done directly.



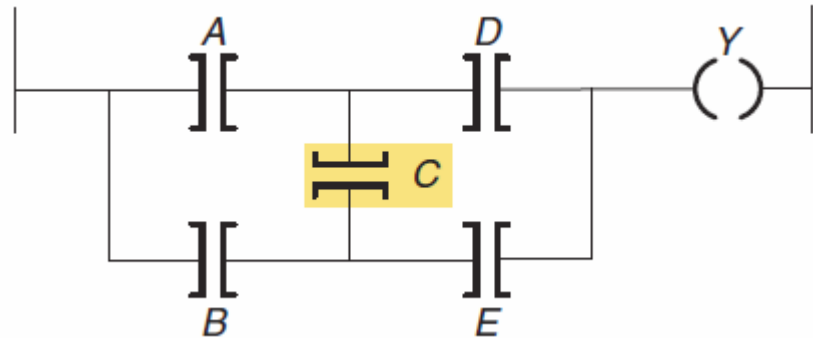
It is possible, however, to program a **logically equivalent** branching condition.



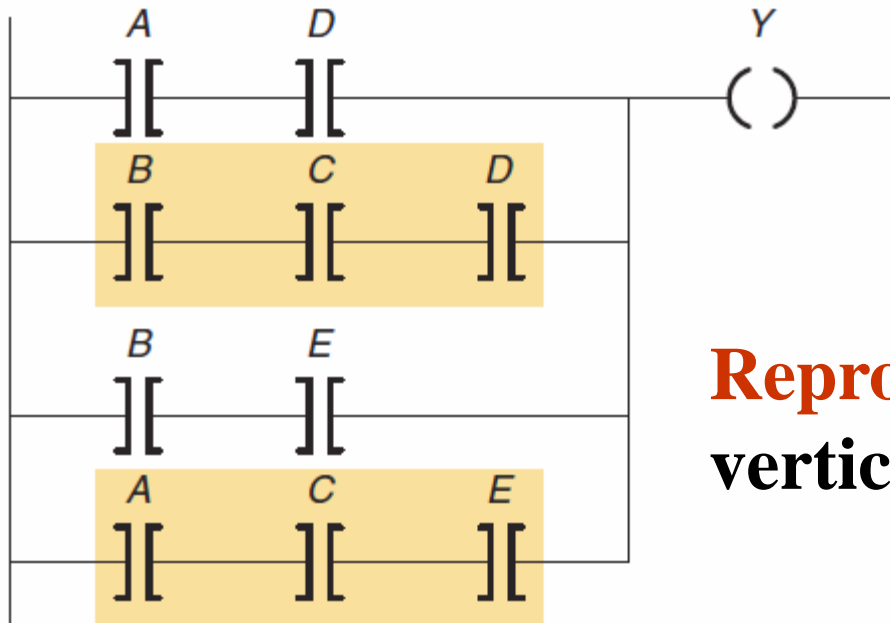
There may be *limitations* to the number of *series contact* instructions that can be included in one rung of a ladder diagram as well as limitations to the number of *parallel branches*.



The PLC will not allow for programming of **vertical contacts**.

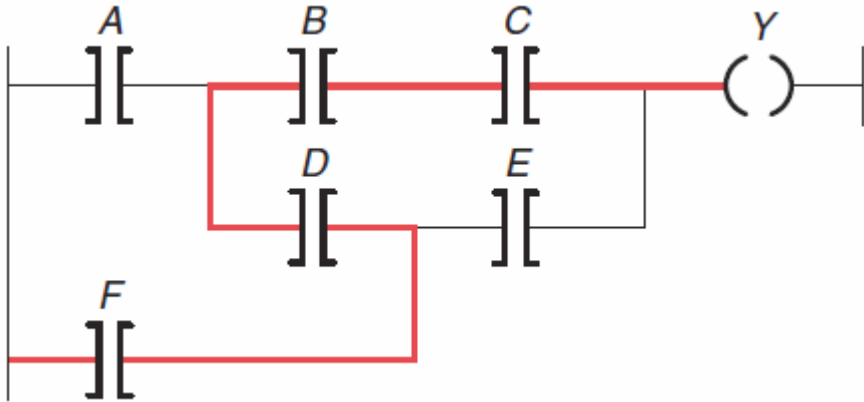


Boolean equation: $Y = (AD) + (BCD) + (BE) + (ACE)$



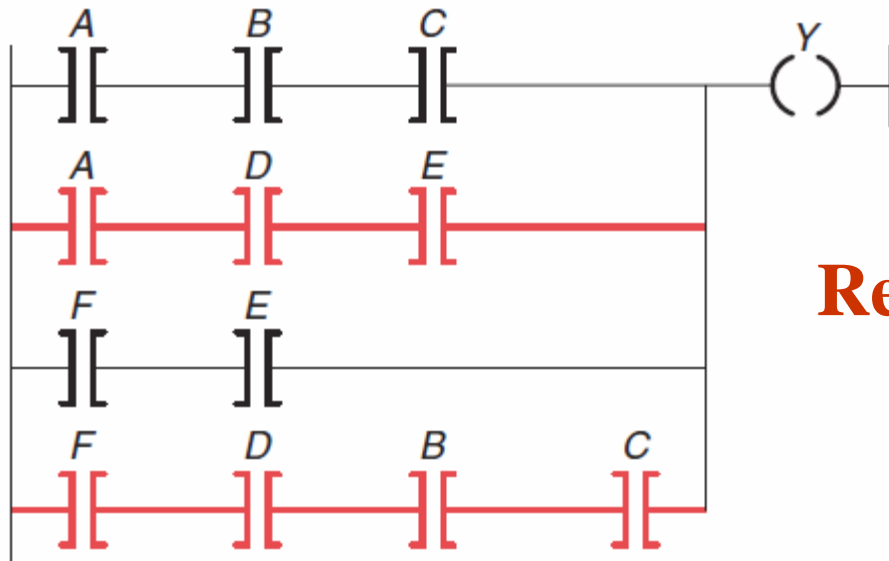
Reprogrammed to eliminate vertical contact.

The processor examines the ladder logic rung for logic continuity from **left to right**.



If programmed as shown, contact combination **FDBC** would be **ignored**.

Boolean equation: $Y = (ABC) + (ADE) + (FE) + (FDBC)$



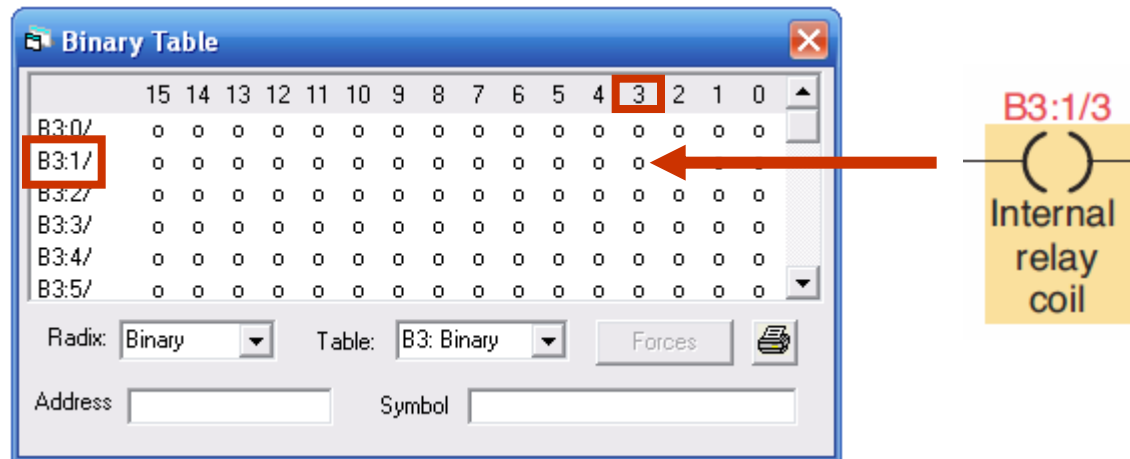
Reprogrammed circuit.

5.7



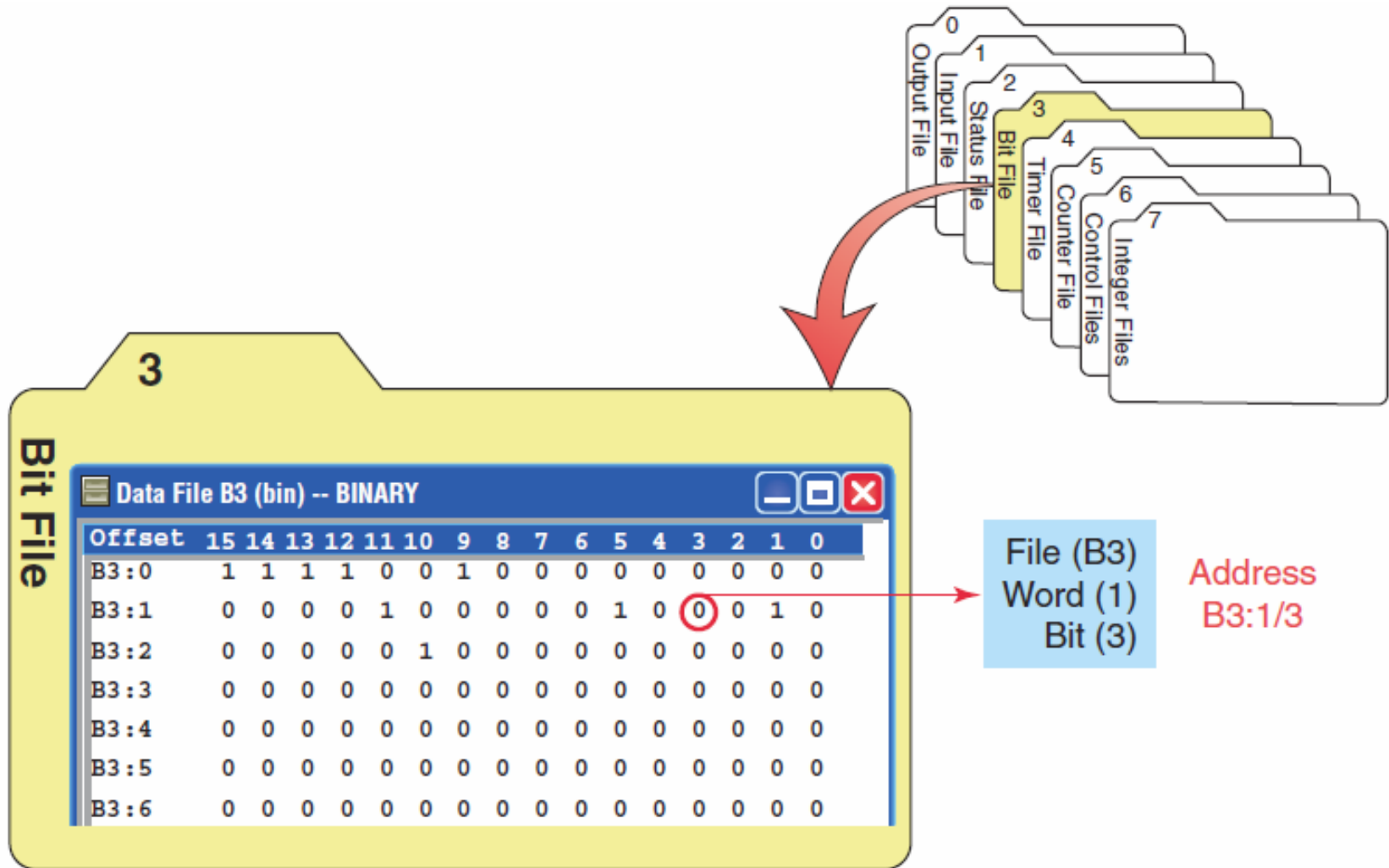
Internal Relay Instructions

An internal output does not directly control an output field device.

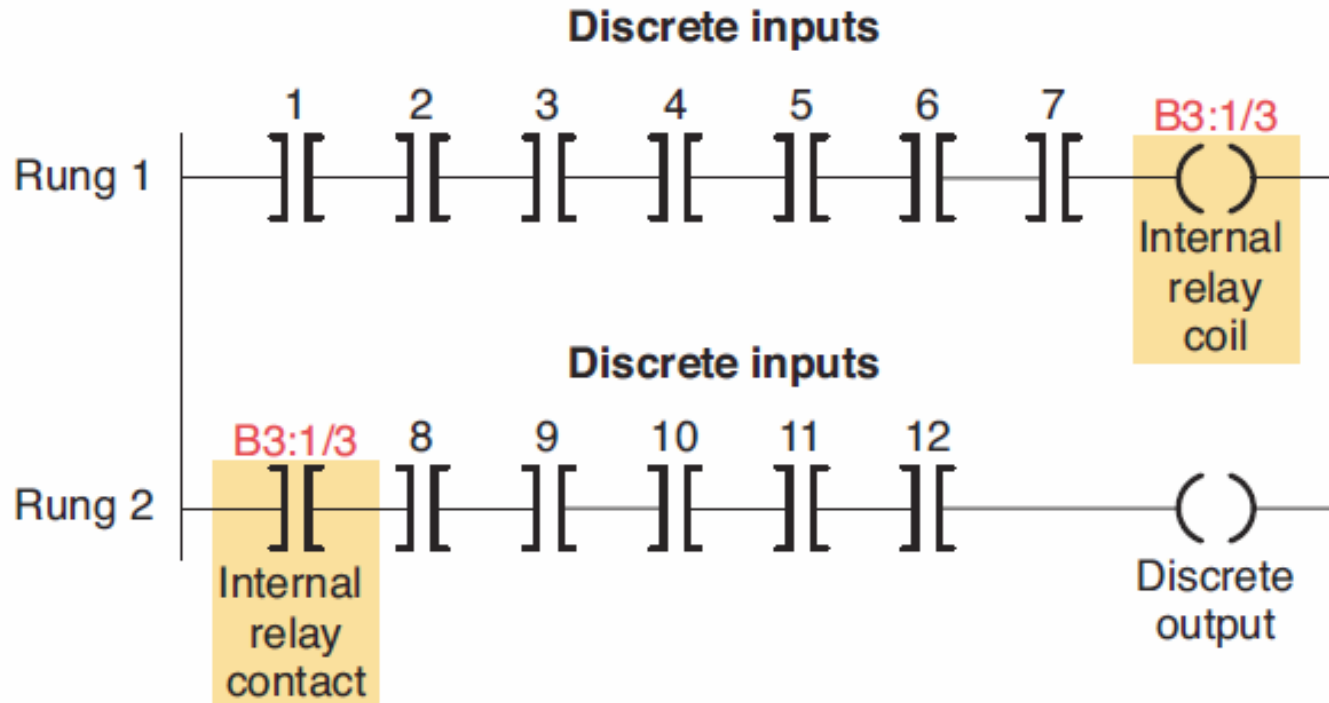


The advantage of using internal outputs is that there are many situations in which an **output** instruction is **required** in a program but **no** physical connection to a **field device** is needed.

SLC 500 controllers use **bit file B3** for internal bit addressing.

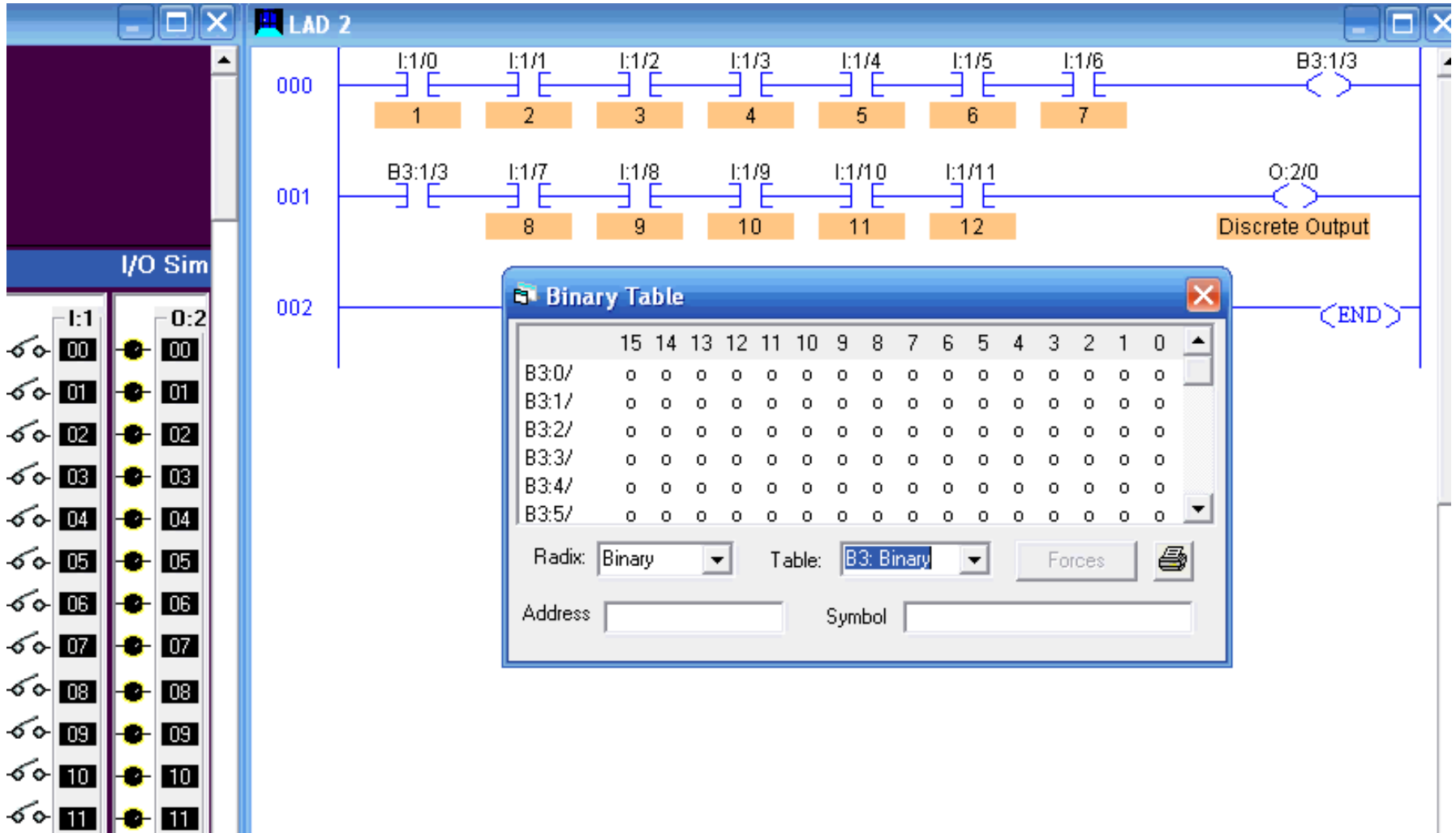


Internal relay used for a program that requires *more series contacts* than the rung allows.



This PLC allows for only **7** series contacts when **12** are actually required for the programmed logic.

Simulated internal relay program.

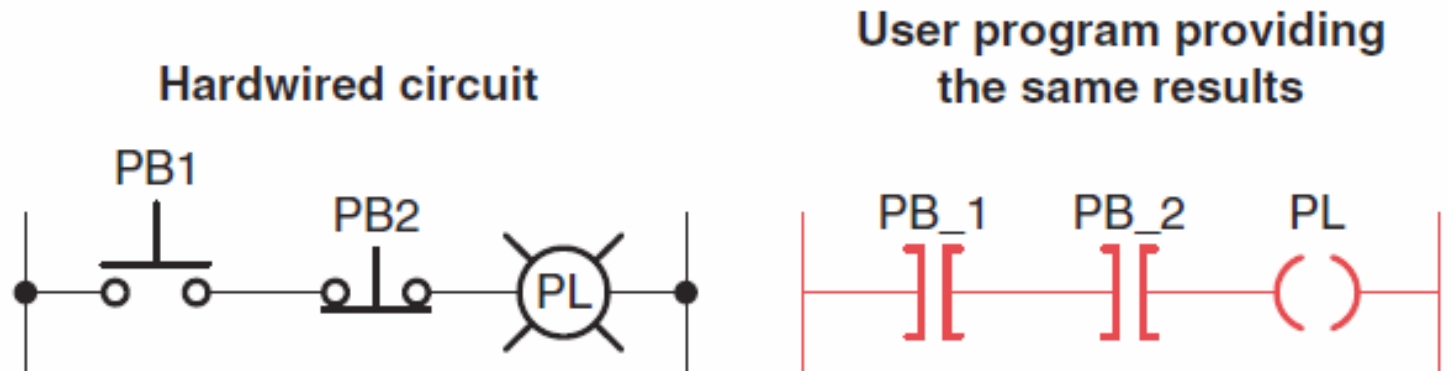


5.8



Programming Examine If Closed and Examine If Open Instructions

Examine If Closed (*XIC*) instruction



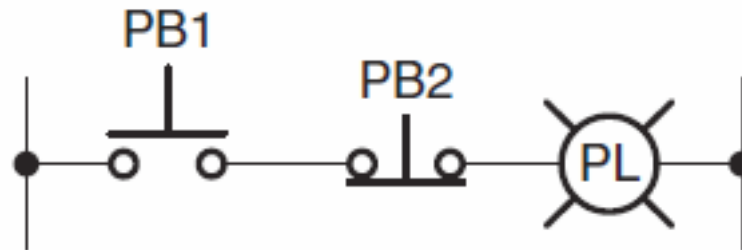
Both the **NO** and the **NC** pushbuttons are represented in the program by the **Examine If Closed** instruction.

The normal state of the field input device (**NO** or **NC**) **does not matter** to the controller.

What matters is that **contacts** need to **closed** to energize the output.

Simulated Examine If Closed (XIC) instruction

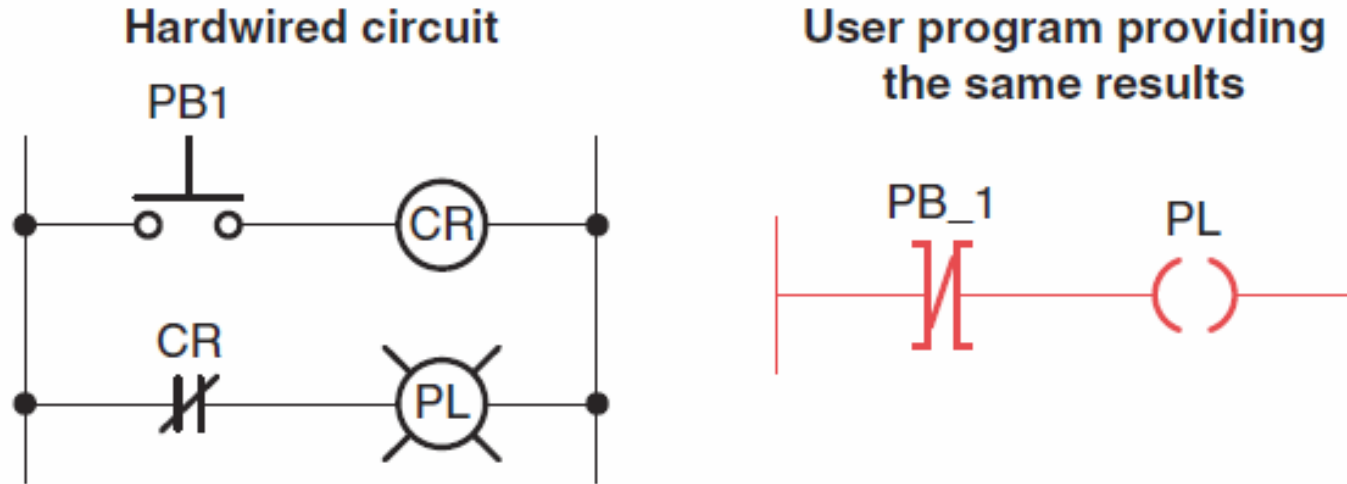
Hardwired circuit



The screenshot shows a software interface for simulating a PLC. On the left is an I/O simulation panel with 16 digital inputs (I:1/00 to I:1/10) and 16 digital outputs (O:2/00 to O:2/10). The input I:1/01 is currently active (red). The main window displays a Ladder Logic (LAD) diagram with two rungs. The first rung contains two normally open contacts labeled I:1/0 (PB1) and I:1/1 (PB2), followed by a coil labeled O:2/0 (PL). The second rung contains an END instruction. An 'Input Table' dialog box is open, showing the current state of the 16 inputs. The table has columns for bit positions 15 through 0 and rows for inputs I:0/ through I:5/. Input I:1/1 is shown as '1', while all other inputs are '0'. The dialog also includes a 'Radix' dropdown set to 'Binary', a 'Table' dropdown set to 'I1: Input', and fields for 'Address' and 'Symbol'.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I:0/																
I:1/		o	o	o	o	o	o	o	o	o	o	o	o	o	1	o
I:2/																
I:3/																
I:4/																
I:5/																

Examine If Open (*XIO*) instruction

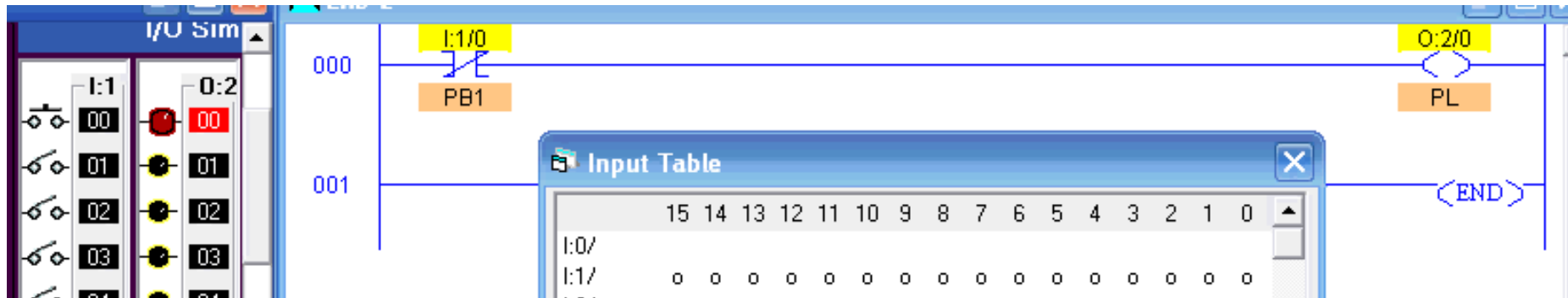
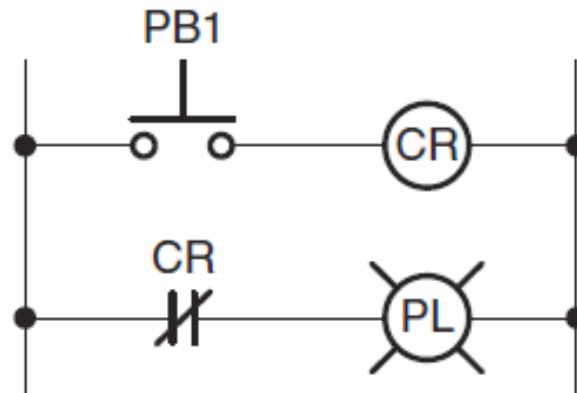


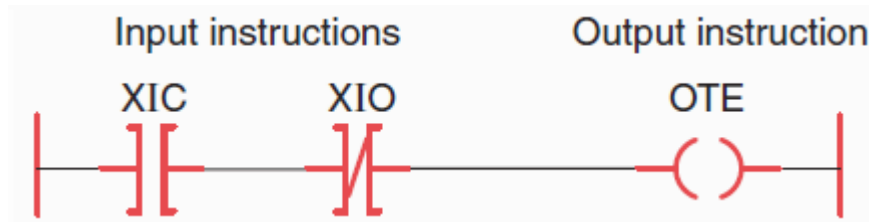
The **pushbutton** is represented in the user program by an **Examine If Open** instruction.

This is because the rung must be **true** when the external pushbutton is **open** and **false** when the pushbutton is **closed**.




Simulated Examine If Open (XIO) instruction

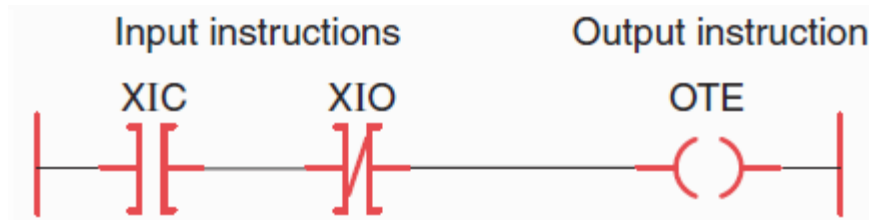
Hardwired circuit





The **logic states (0 or 1)** indicate whether an instruction is true or false and is the basis of controller operation.

If the data table bit is	The status of the instruction is		
	XIC EXAMINE IF CLOSED 	XIO EXAMINE IF OPEN 	OTE OUTPUT ENERGIZE 
Logic 0	False	True	False
Logic 1	True	False	True



The **time aspect** relates to the repeated scans of the program, wherein the input table is updated with the most current status bits.

Time	Instruction outcome		
	XIC	XIO	OTE
t_1 (initial)	False	True	False
t_2	True	True	Goes true
t_3	True	False	Goes false
t_4	False	False	Remains false

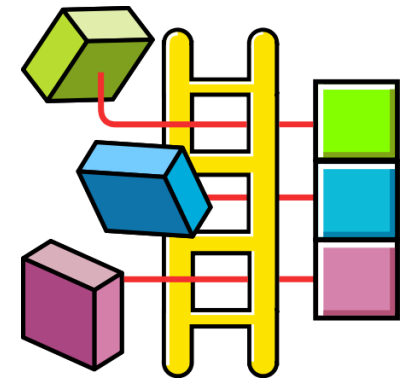
Input bit status		
XIC	XIO	OTE
0	0	0
1	0	1
1	1	0
0	1	0

5.9



Entering The Ladder Diagram

Allen-Bradley's *RSLogix* software packages are windows programming packages used to develop ladder logic programs.



Software, in **various versions**, can be used to program the PLC-5, SLC 500, ControlLogix, and MicroLogic family of processors.

A personal computer is most often used and is adapted to the particular PLC model through the use of the **relevant programmable controller software**.

RSLogix SLC 500 main window.

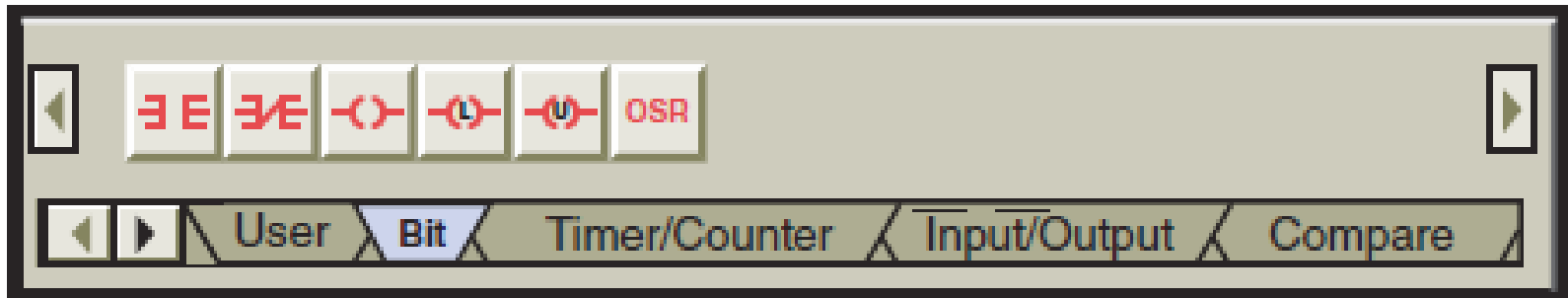
The screenshot displays the RSLogix 500 Pro software interface. The main window title is "RSLogix 500 Pro - IC500DM0.R55". The menu bar includes File, Edit, View, Search, Comms, Tools, Window, and Help. The toolbar contains various icons for file operations and editing. The status bar at the top shows "OFFLINE", "No Forces", "No Edits", and "Forces Disabled". The driver is identified as "AB_DF1-1" and the node as "1d".

The main workspace shows a project tree on the left with folders for "Project", "Help", "Controller", "IO Configuration", "Channel Configuration", "Multipoint Monitor", and "Program Files". Under "Program Files", there are sub-routines: "SYS 0 -", "SYS 1 -", "LAD 2 - MAIN_LADDR", "LAD 3 - PID_SUB_RT", "LAD 4 - SUB_RT_2", and "LAD 5 - SUB_RT_3".

The main workspace displays a Ladder Logic (LAD) diagram for "LAD 2 -- MAIN_LADDR". The diagram shows a red vertical bar representing the power rail, labeled "0000". A green box at the top of the diagram is titled "Start Up Diagnostic Check" and contains the text "The rungs in this section perform a diagnostic check at start up." Below this, there are two normally open contacts in series: "CTRL_ENABLED" (address B3:0/0) and "TOGGLE_SW_1" (address I:1/0). The output of this series combination is connected to a coil labeled "1746-IA4". This coil is connected to another normally open contact labeled "START_UP_CHECK" (address O:3/0).

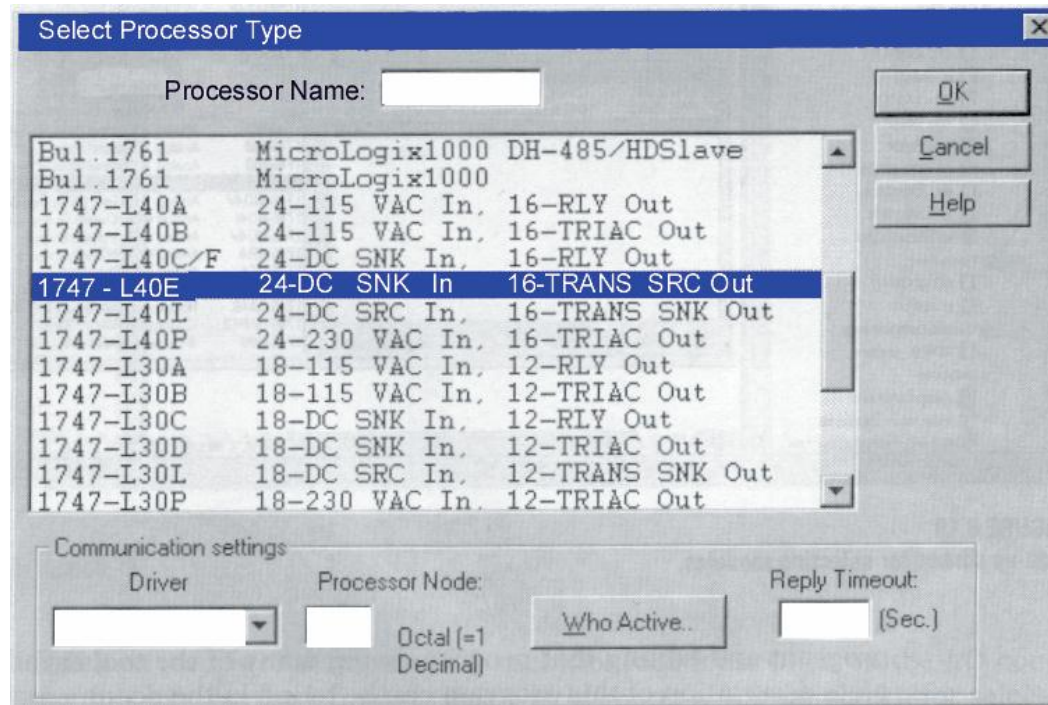
The bottom status bar shows "For Help, press F1", a counter value of "2:0000", and buttons for "APP", "READ", and "Disabled".

Instruction toolbar with *bit instructions* selected.



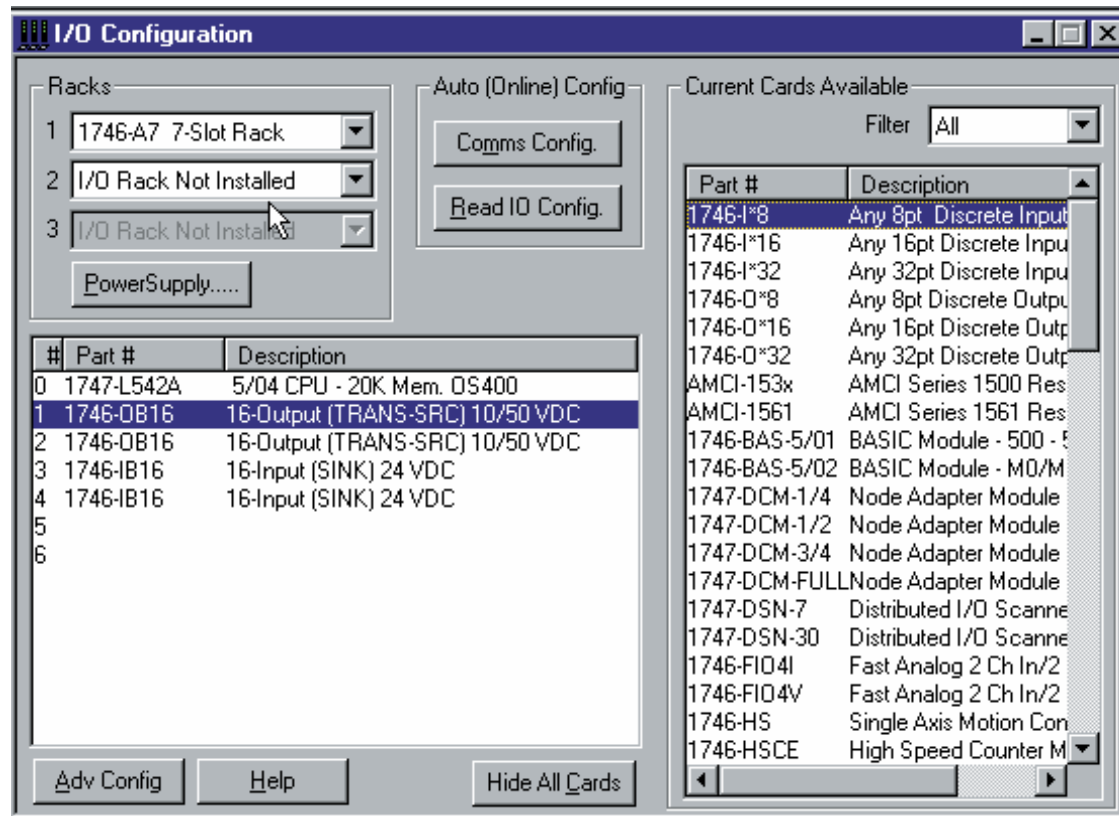
To place an instruction on a rung, **click** its icon on the toolbar and simply **drag** the instruction straight off the toolbar onto the rung of the ladder.

The programming software needs to know what *processor* is being used in conjunction with the user program.



The **Select Processor Type** screen contains a list of the different processors that the RSLogix software can program.

The *I/O Configuration* screen lets you double click or drag-and-drop a module from an all-inclusive list to assign it to a slot in your configuration.



Data File screens contain data that are used in conjunction with ladder program instructions and include input and output files as well as timer, counter, integer, and bit files.

Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3:0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
B3:1	1	1	0	1	0	0	1	0	1	0	0	0	0	0	0	1
B3:2	1	1	0	1	1	0	1	0	1	0	1	0	1	1	0	1
B3:3	1	1	0	0	1	0	1	0	1	0	1	0	1	1	0	1
B3:4	1	1	0	0	0	0	0	0	1	0	1	0	0	1	0	1

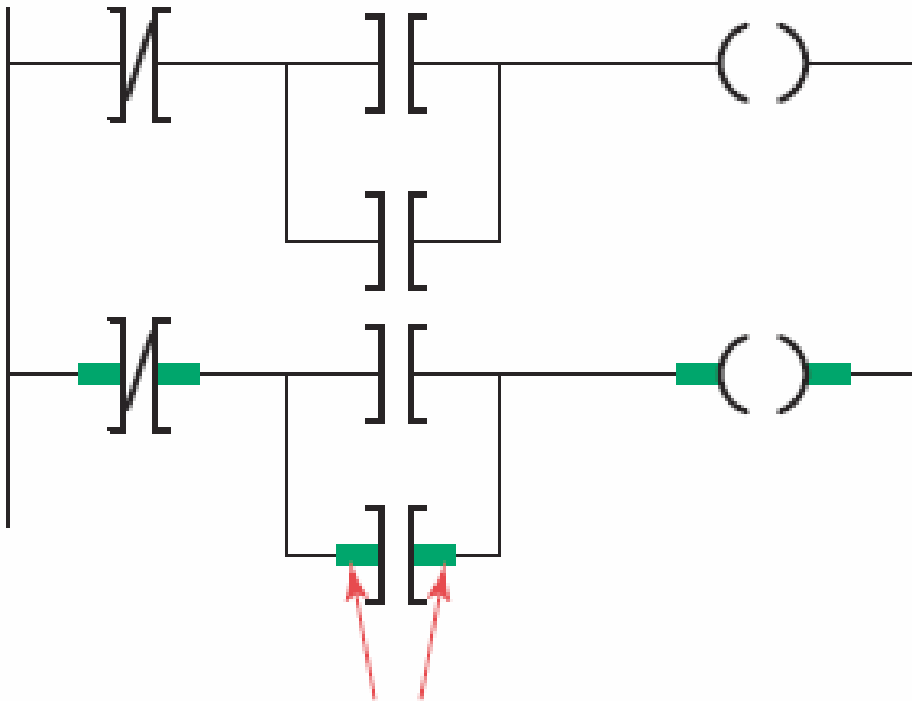
Symbol: Radix:

Desc:

Columns:

The bit file **B3** is used for internal relays.

Relay ladder logic is a *graphical* programming language designed to closely represent the appearance of a wired relay system.



Highlighted rungs indicate the instruction is true.

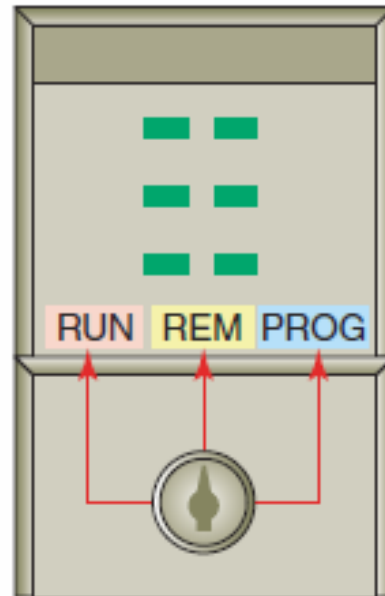
The logic is apparent from the **highlighting** which identifies the logic state of contacts in real time and which rungs have logic continuity.

5.10



Modes Of Operation

A processor has basically two modes of operation: the *program mode* and some variation of the *run mode*.



A **three-position keyswitch** may be used to select different processor modes of operation.

The **program mode** is used to enter a new program, edit or update an existing program, upload files and download files.

The **run mode** is used to execute the user program.

The **test mode** is used to operate or monitor the user program without energizing any outputs.

The **remote position** allows the PLC to be remotely changed between program and run mode by a personal computer connected to the PLC processor.

