



Chapter 9

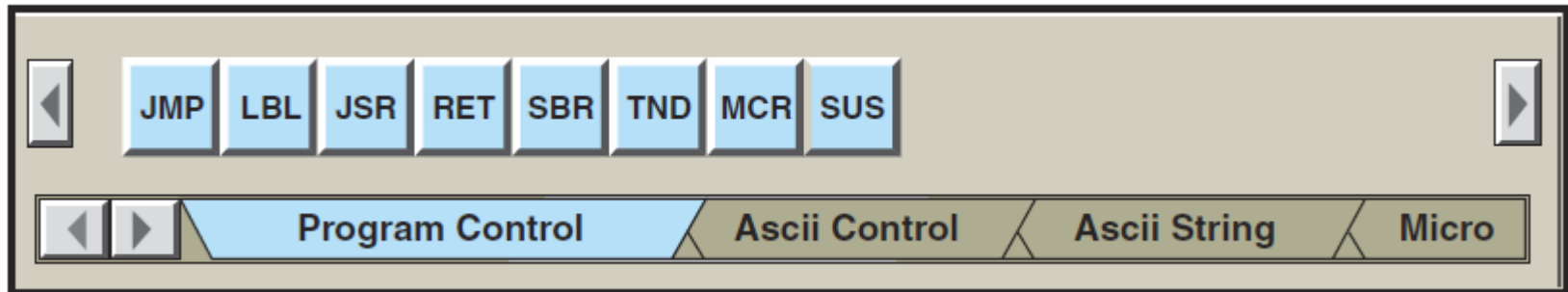
Program Control Instructions

9.1



Master Control Reset Instruction

Program control instructions are used to enable or disable a block of logic program or to move execution of a program from one place to another place.

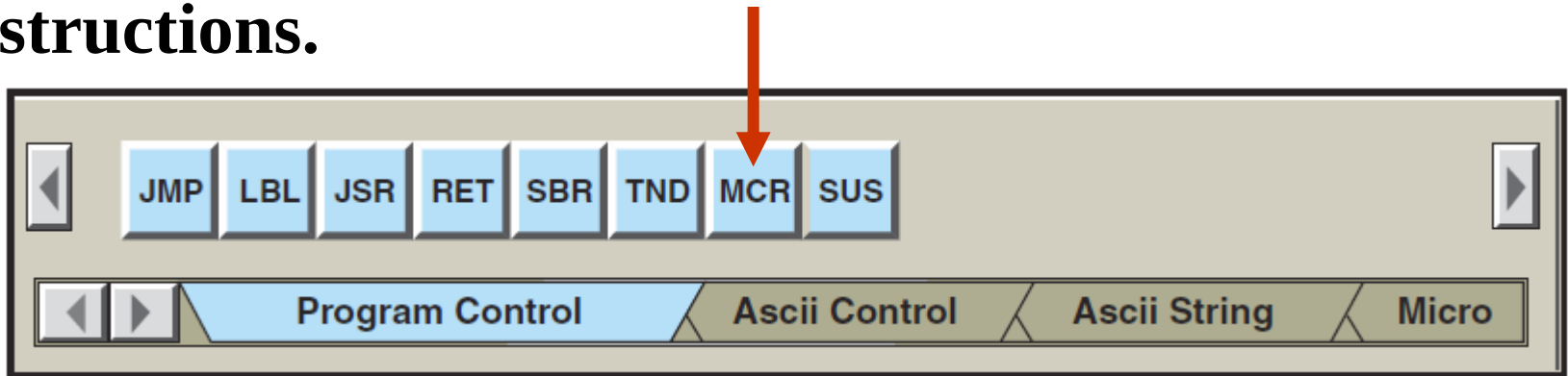


Program Control menu tab for the Allen-Bradley
SLC 500 PLC

The diagram shows a three-rung ladder logic circuit. The power supply is L1 on the left and L2 on the right. The first rung is the master stop/start control: it starts with a normally open contact labeled 'Master stop', followed by a parallel combination of a normally open contact labeled 'Master start' and a normally closed contact labeled 'MCR' (which is highlighted in blue). This is followed by a coil labeled 'MCR' (also highlighted in blue). The second rung controls the conveyor roller motor (CR): it starts with a normally open contact, followed by a parallel combination of a normally open contact and a normally closed contact labeled 'CR4'. This is followed by a coil labeled 'CR'. The third rung controls motor M1: it starts with a normally open contact, followed by a parallel combination of a normally open contact and a normally closed contact labeled 'CR1'. This is followed by a coil labeled 'M1'. The fourth rung controls motor M2: it starts with a normally open contact labeled 'CR2', followed by a normally closed contact labeled 'M1', and then a coil labeled 'M2'. Both the M1 and M2 coils are in parallel with a common output line that goes to two output devices, both labeled 'OL' and represented by a switch symbol.

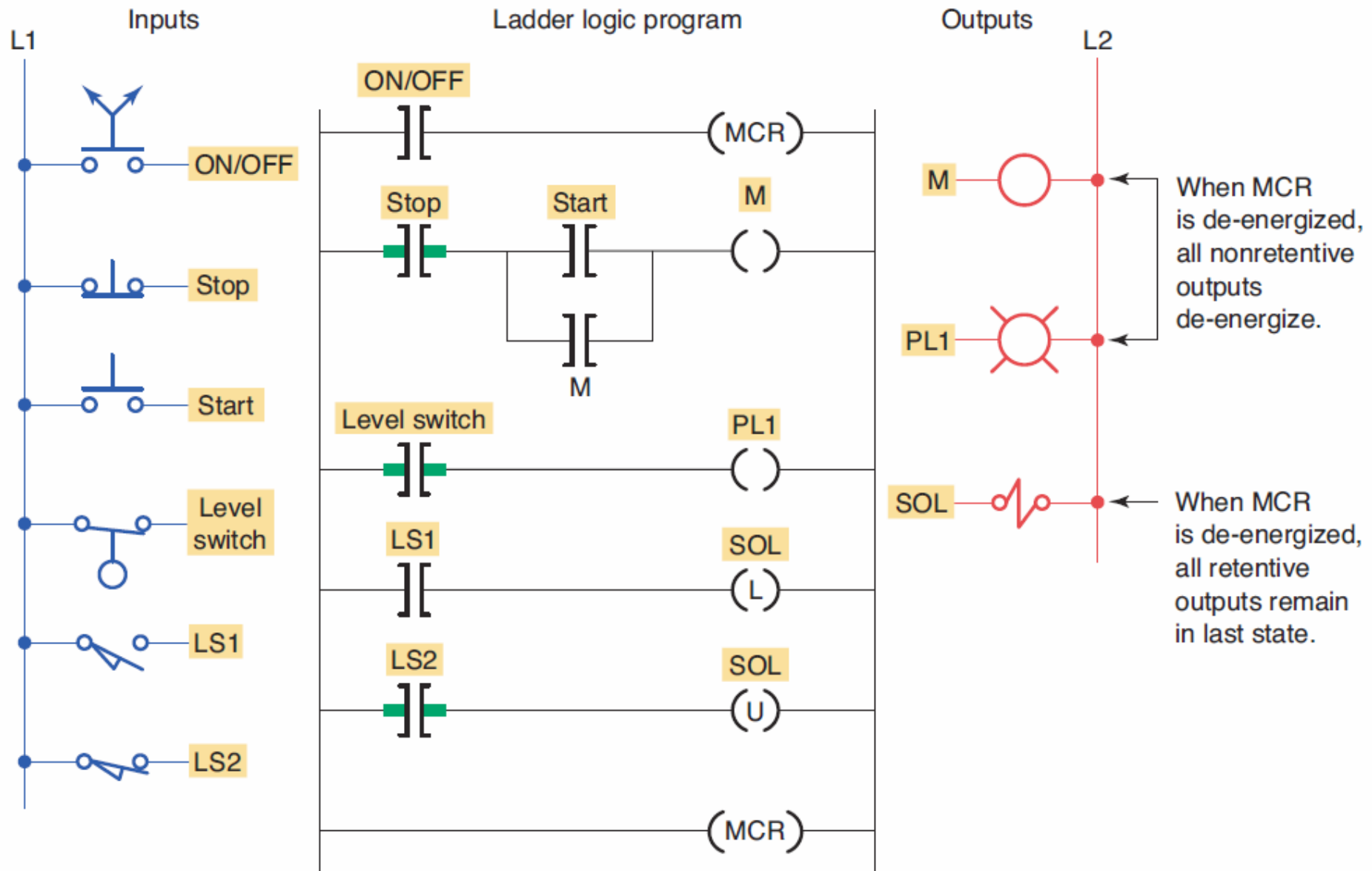
PLC manufacturers offer a form of a master control relay as part of their instruction set.

MCR (Master Control Reset) - Clears all set nonretentive output rungs between the paired MCR instructions.

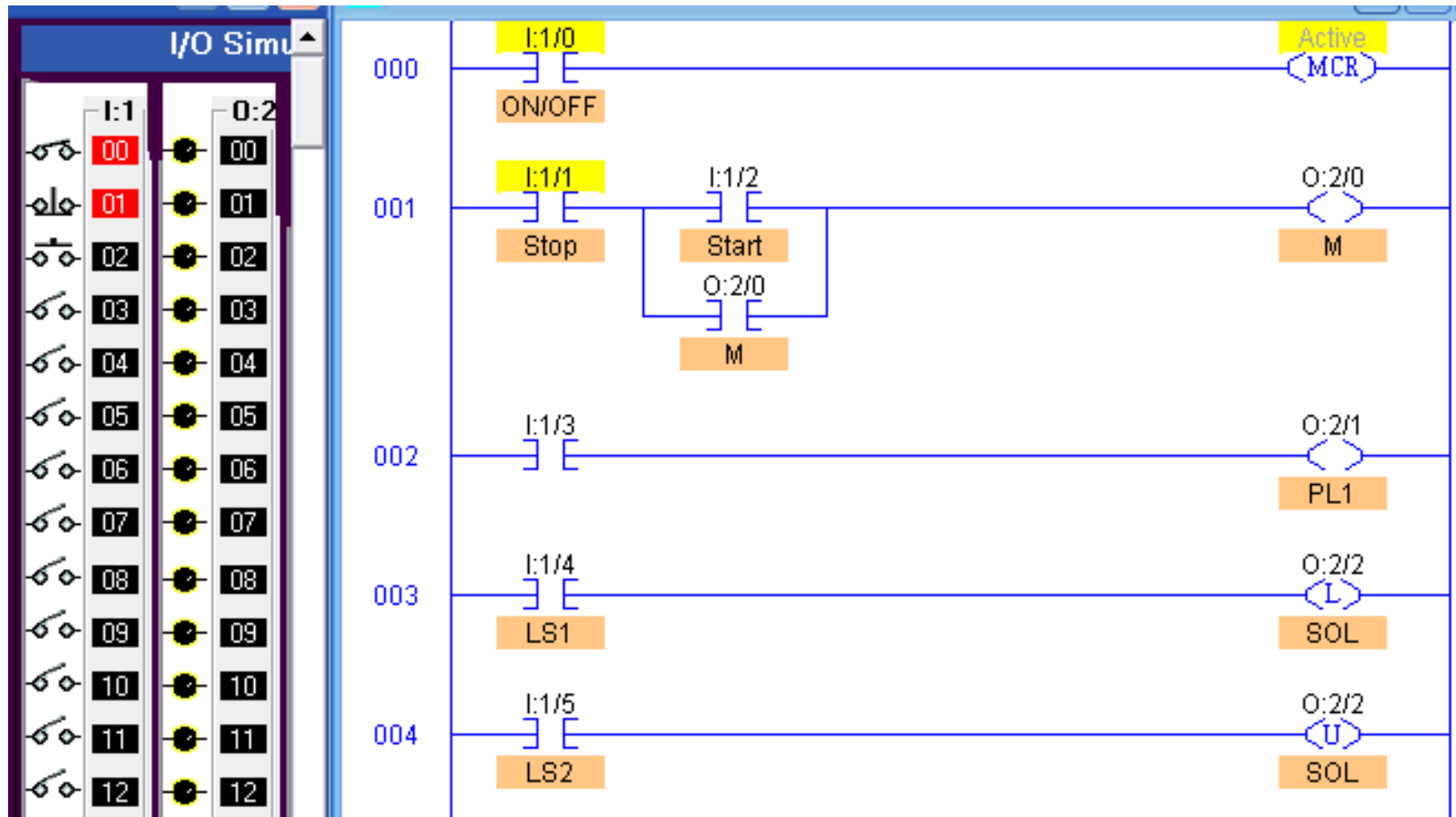


Because the MCR instruction is programmed, for safety reasons, it **should not** be used as a **substitute** for a hardwired master control relay, which provides **emergency** I/O power shutdown.

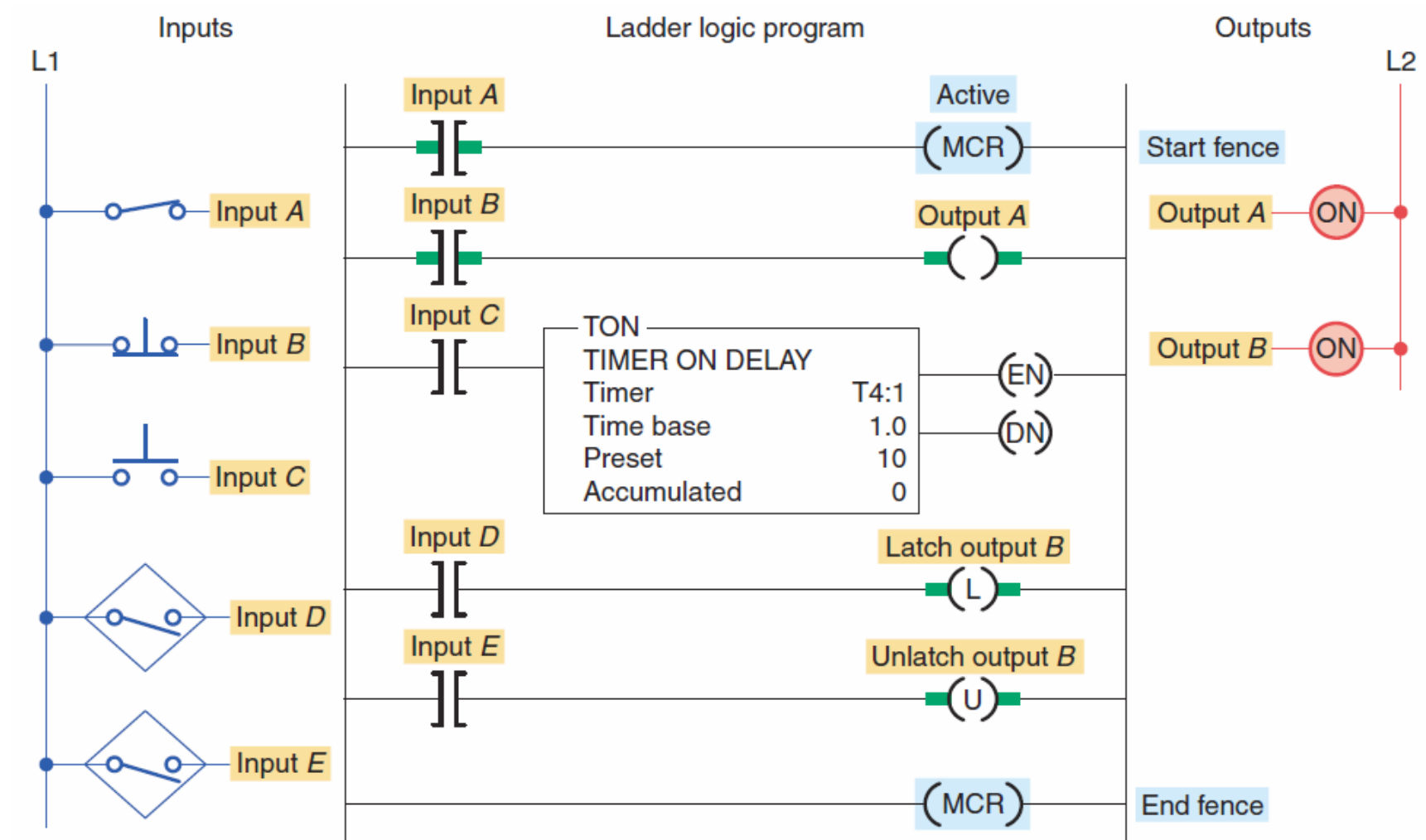
Programmed Master Control Reset instruction.



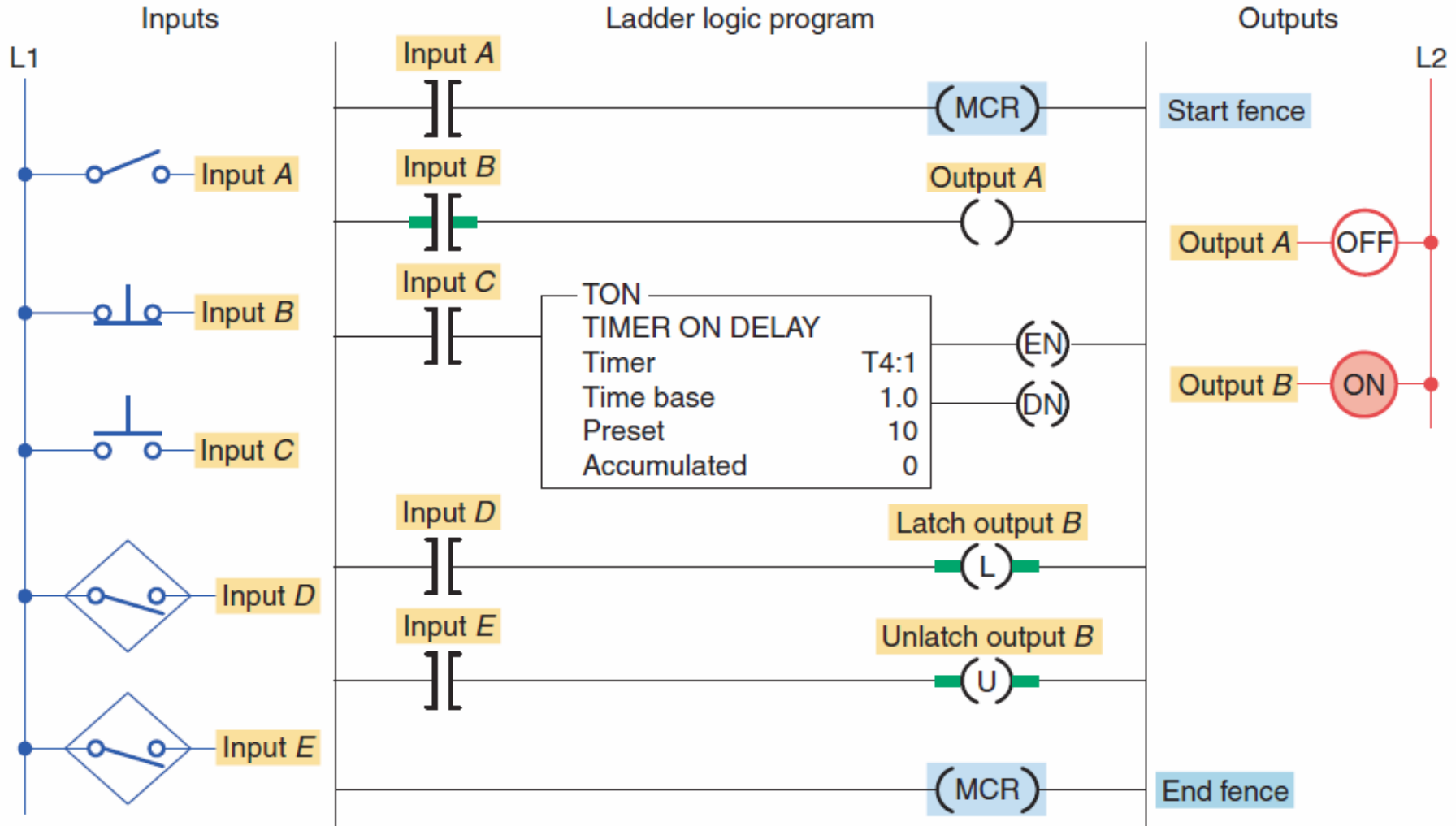
Simulated Master Control Reset instruction.



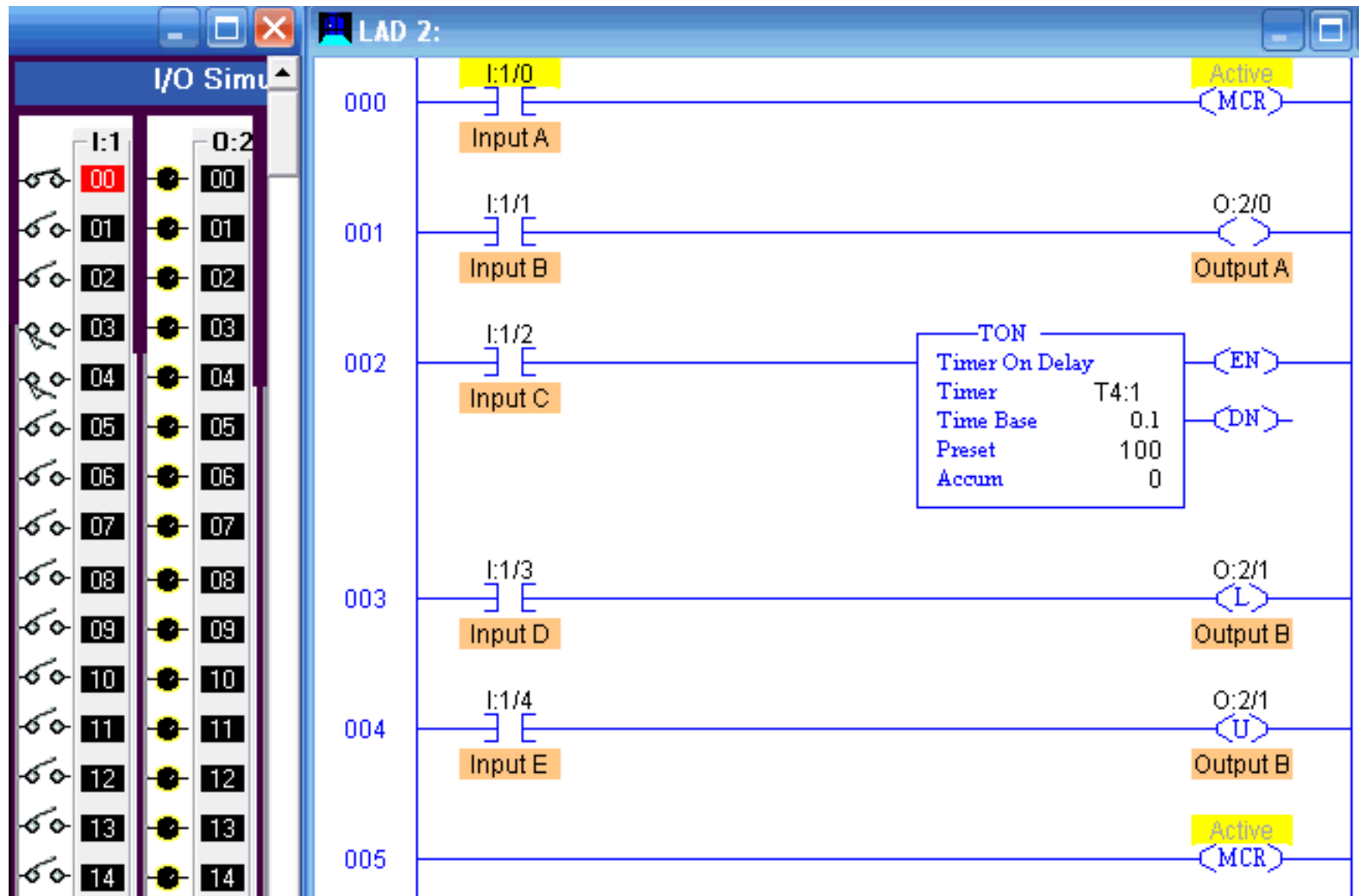
MCR fenced zone with the zone true.



MCR fenced zone with the zone false.



Simulated MCR fenced zone.



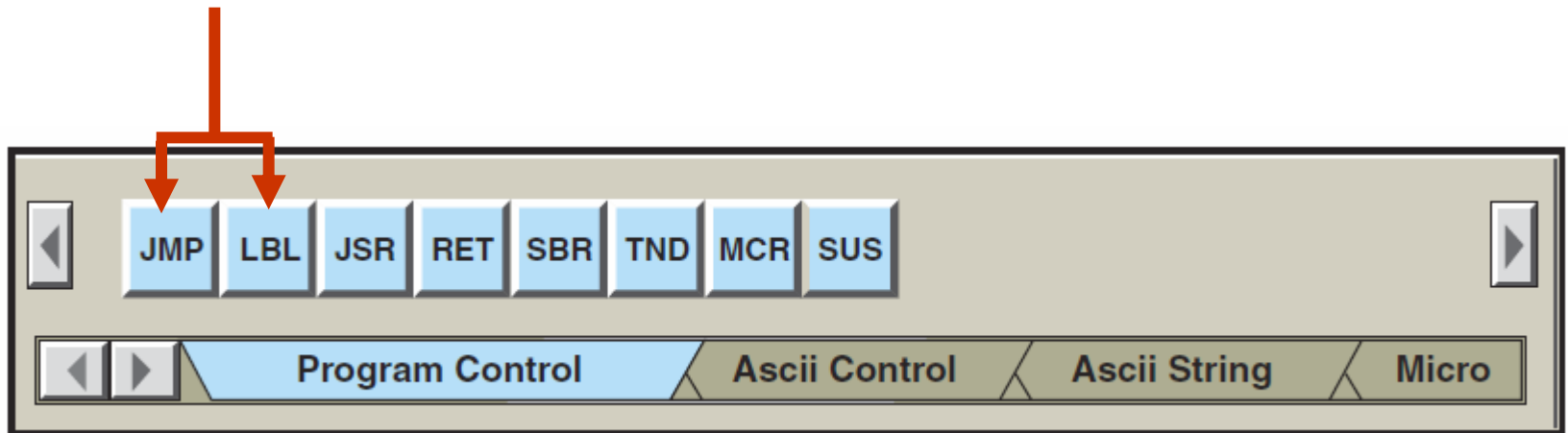
9.2



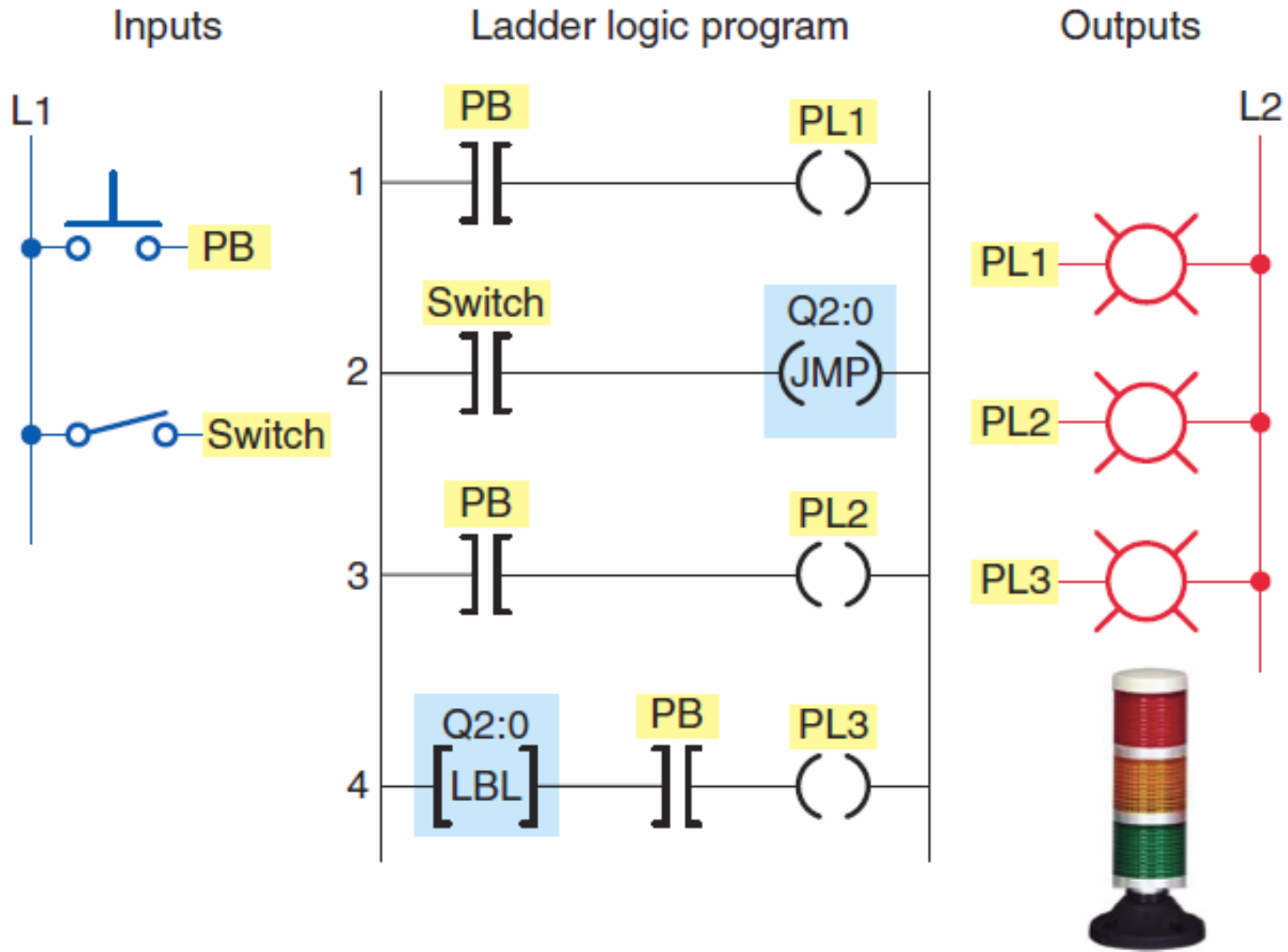
Jump Instruction

The *jump (JMP)* instruction is used to jump over certain program instructions when certain conditions exist.

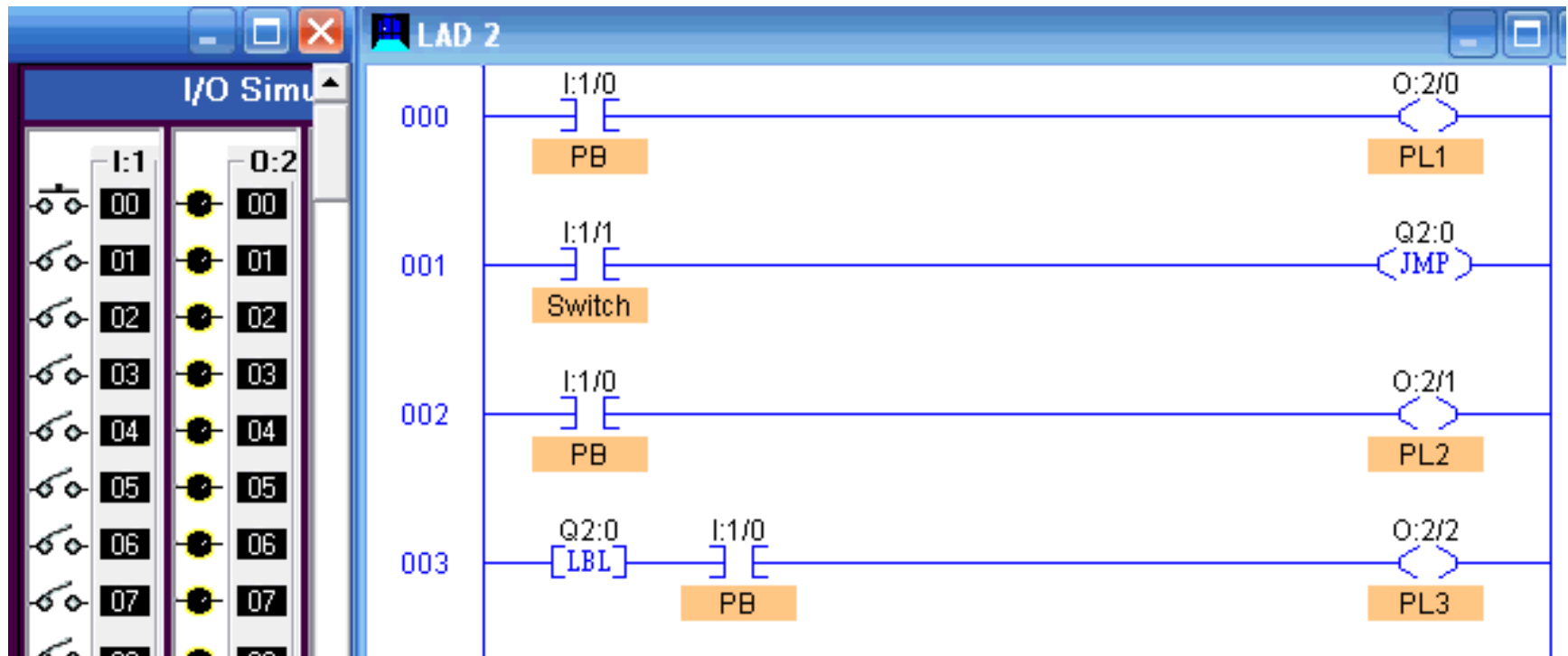
The **label instruction** is used to identify the ladder rung that is the **target destination** but does not contribute to logic continuity.



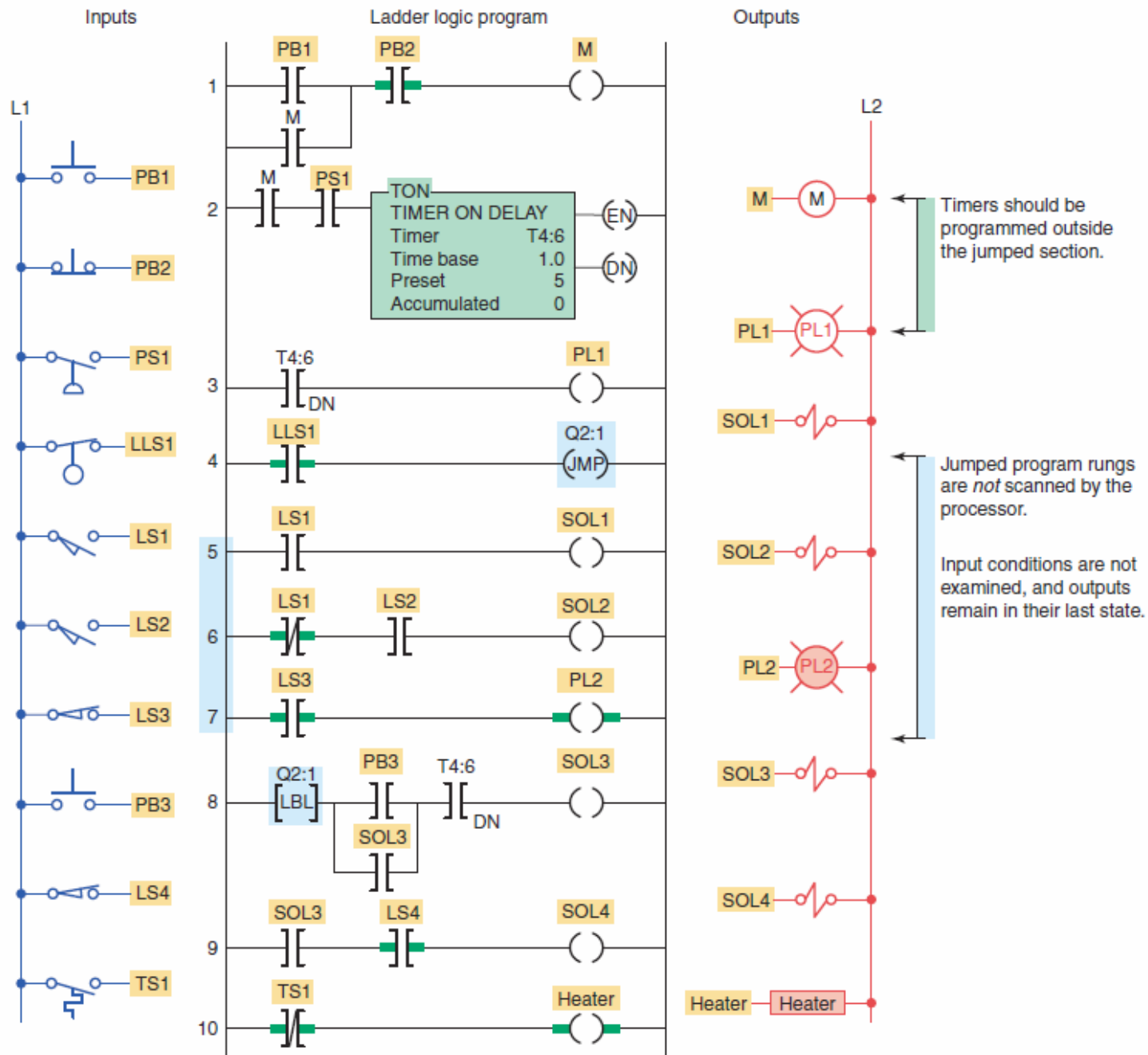
Jump (JMP) operation program.



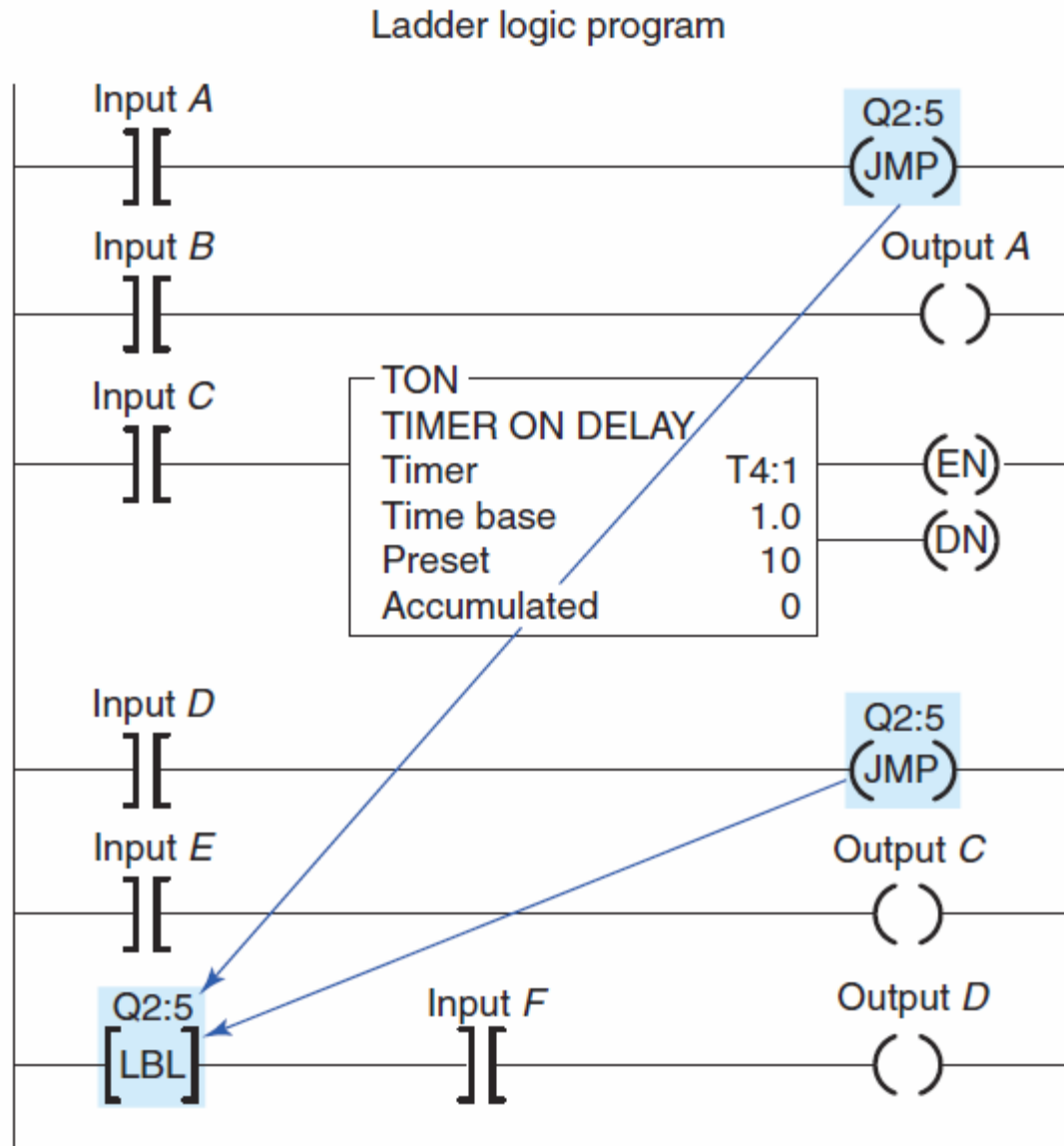
Simulated jump (JMP) operation program.



Effect on instructions of jumped rungs.



Jump-to-label from two locations.



9.3



Subroutine Functions

A subroutine is a short program that is used by the main program to perform a specific function.

JSR (Jump to Subroutine)

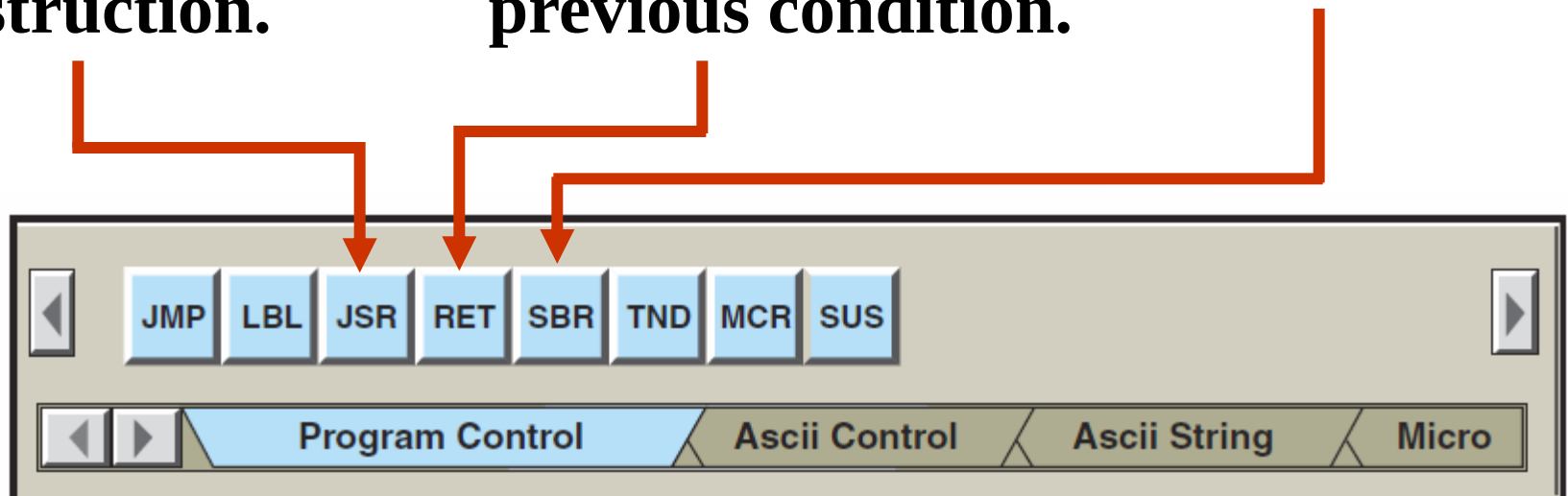
Jump to a designated subroutine instruction.

RET (Return from Subroutine)

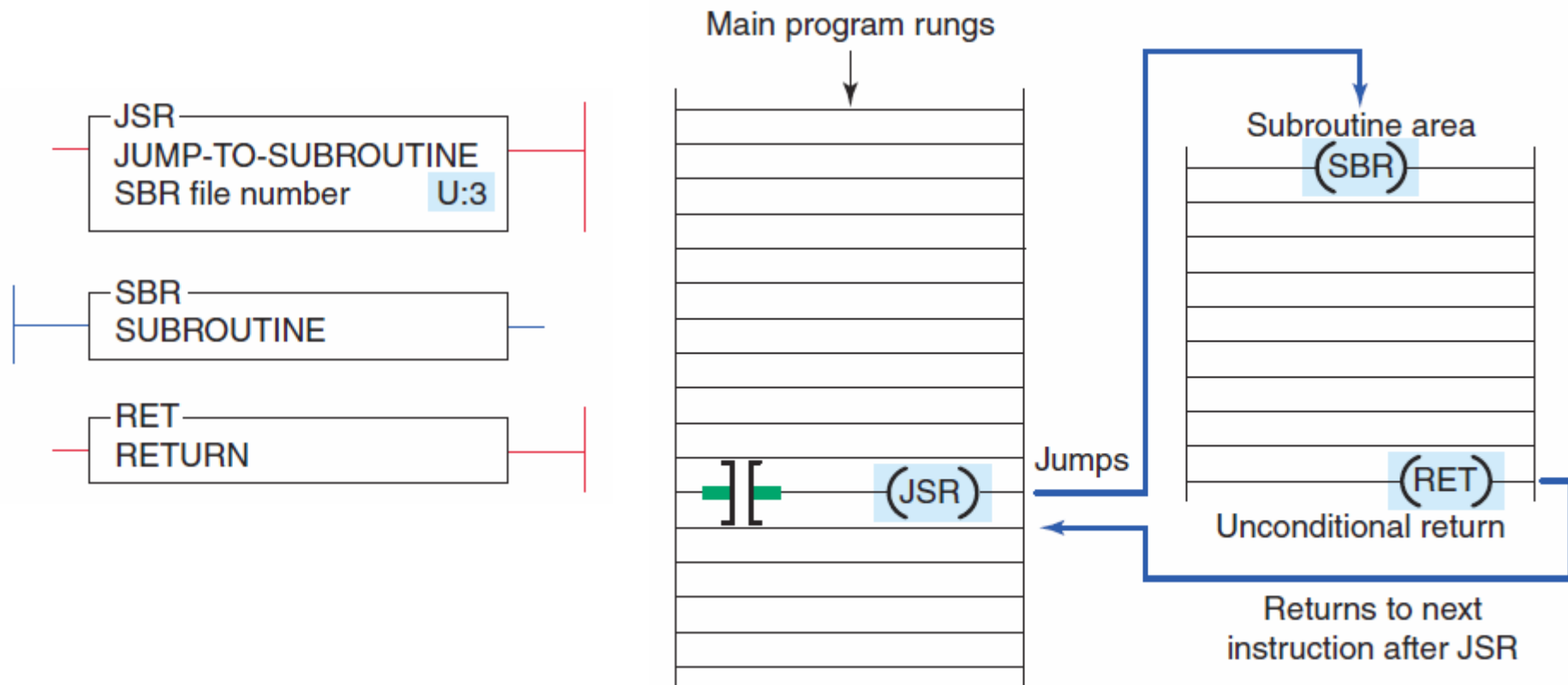
Exits current subroutine and returns to previous condition.

SBR (Subroutine)

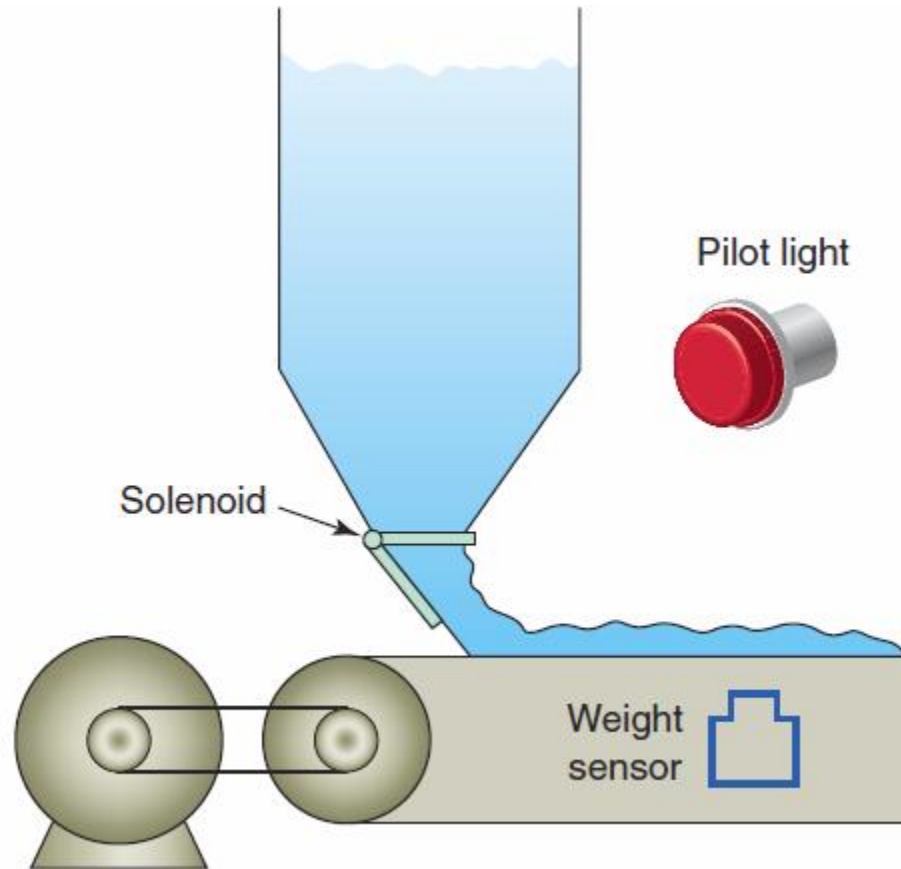
Identifies the subroutine program.



When a subroutine is **called** from the main program, the program is able to escape from the main program and **go to** a program **subroutine** to perform certain functions and **then return** to the main program.

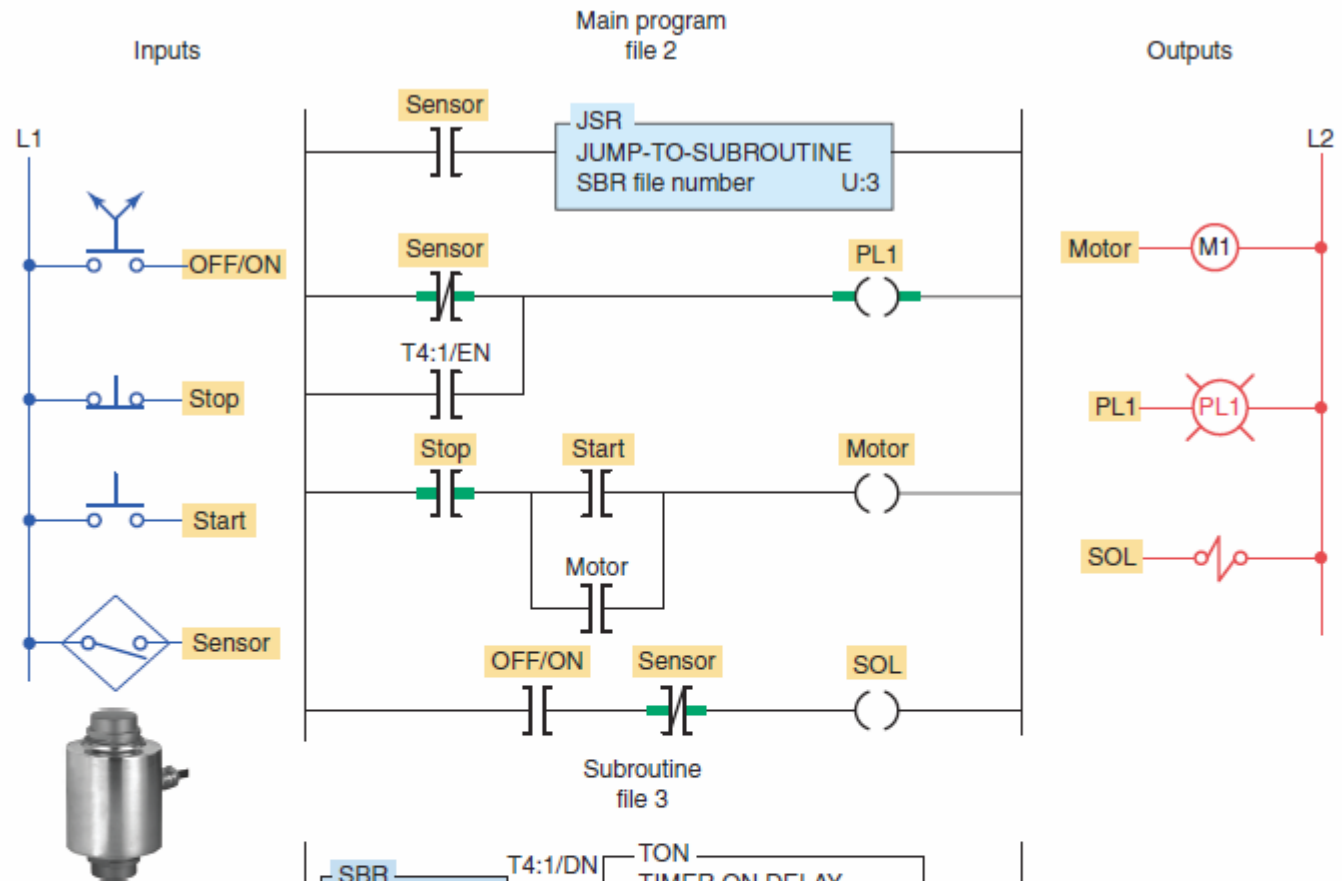


Flashing pilot light subroutine program.

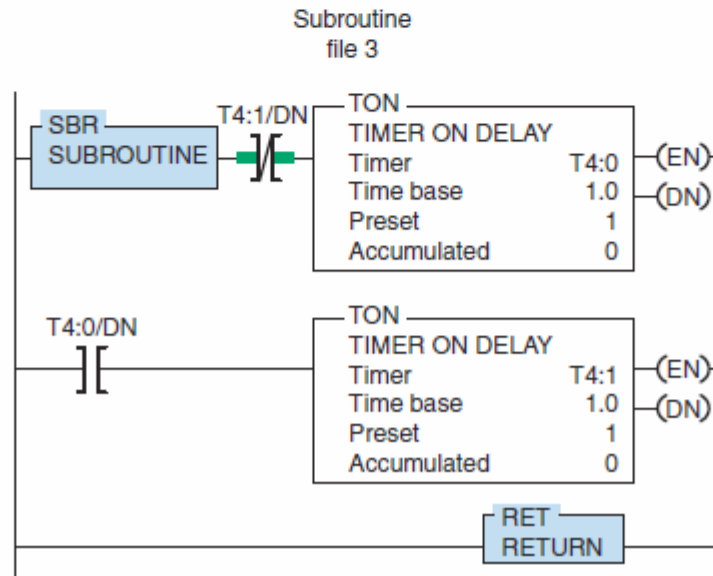


Anytime the weight on the conveyor exceeds a preset value, the subroutine program is scanned and pilot light changes from on to flashing.

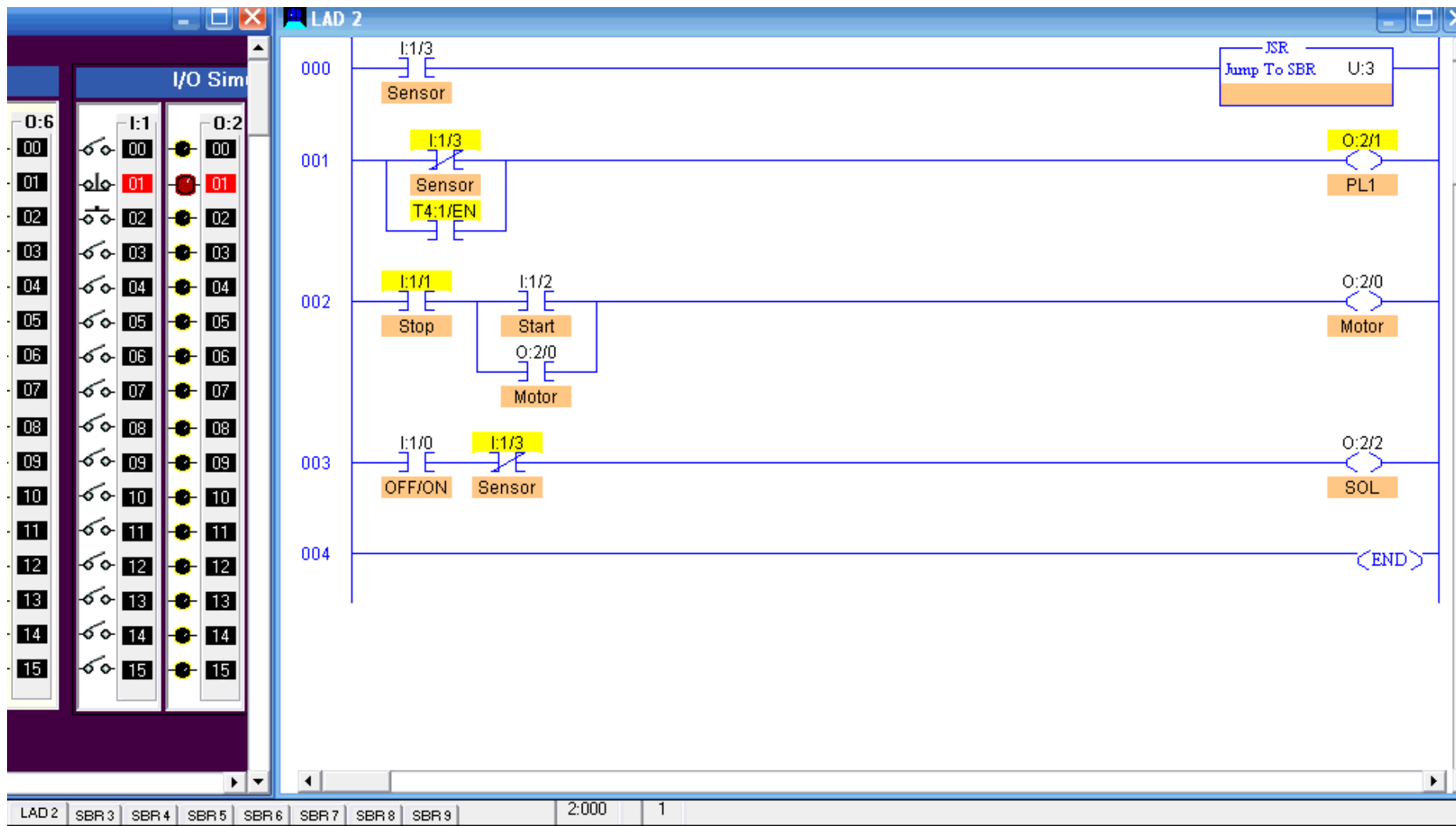
Main Program

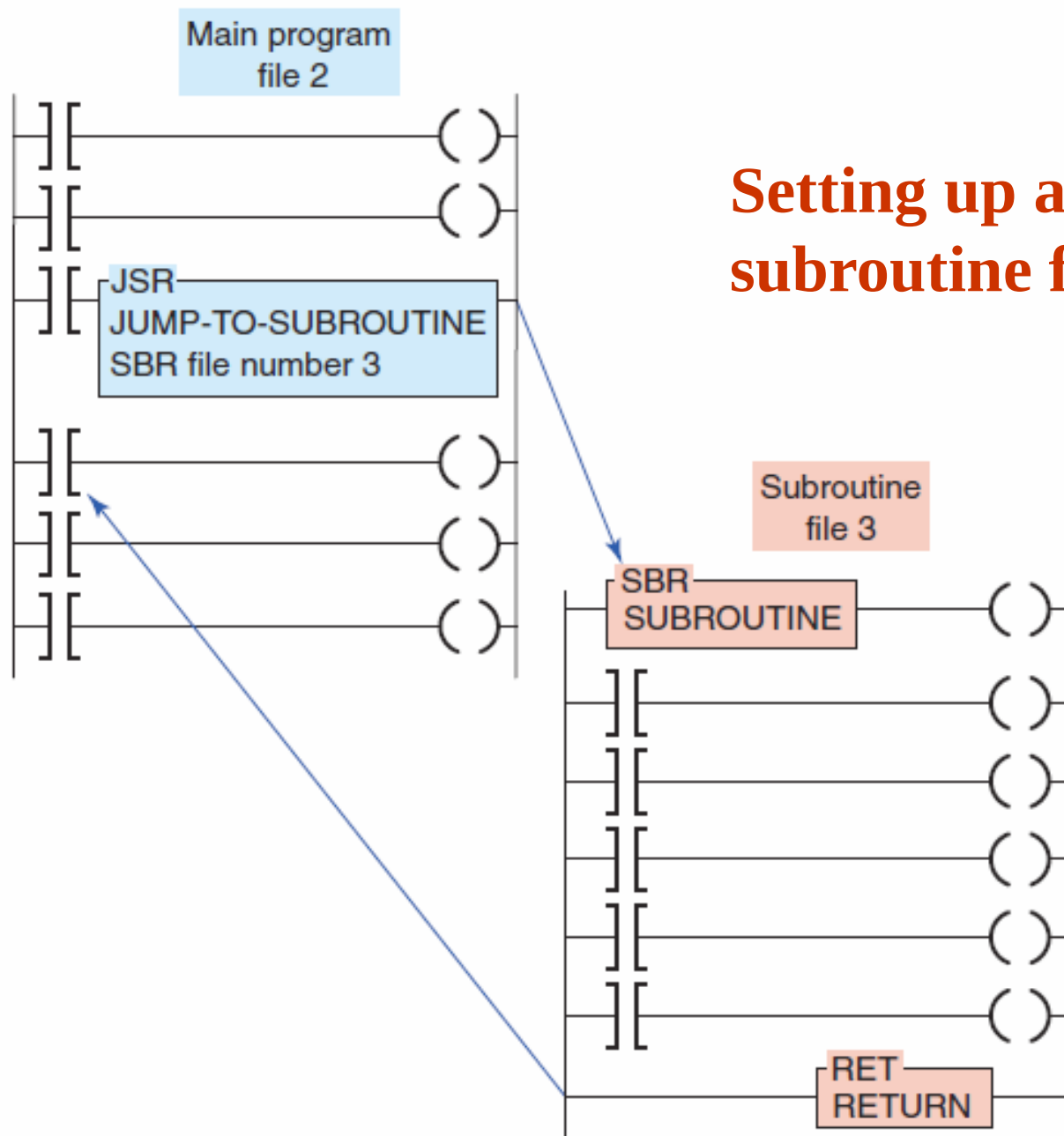


Subroutine Program

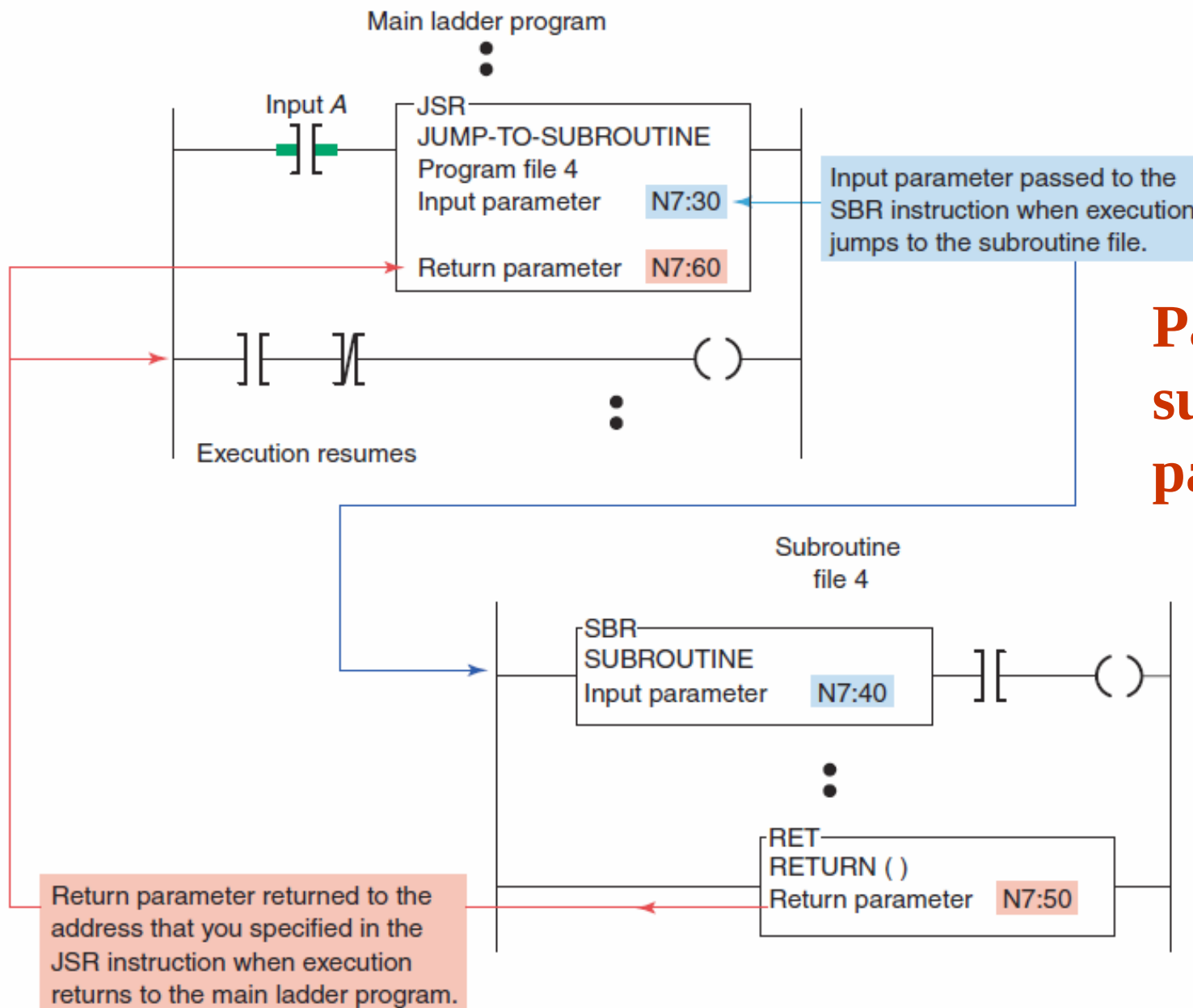


Simulated subroutine program.



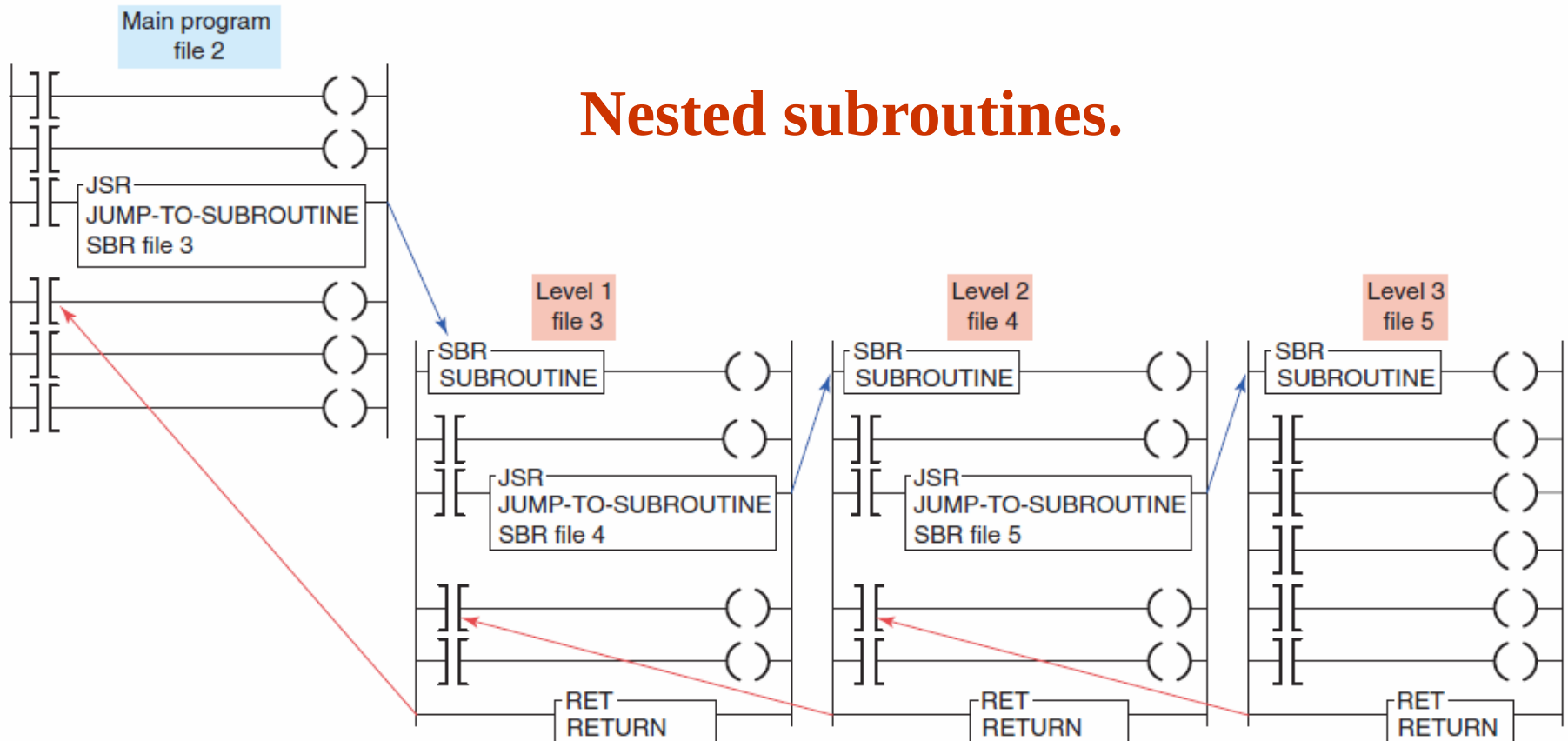


**Setting up a
subroutine file.**



Passing subroutine parameters

Nested subroutines.



9.4



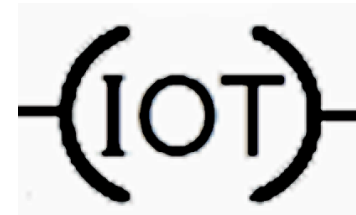
Immediate Input and Immediate Output Instructions

The *immediate input* and *immediate output* instructions interrupt the normal program scan to update the input and output image table files with current data.



Immediate input (IIN)

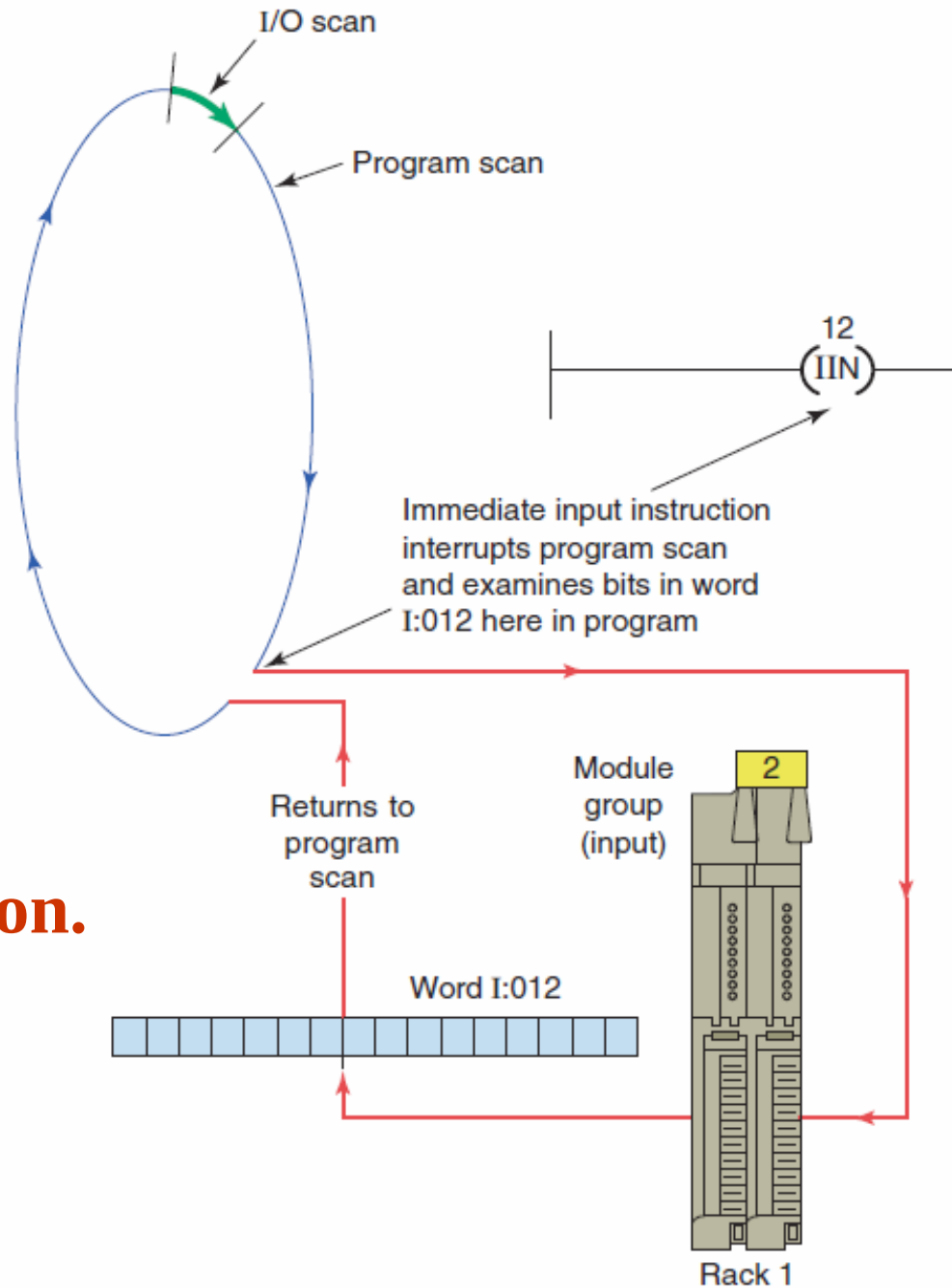
PLC-5 instruction
is used to read an
input condition
before the I/O
update is performed.



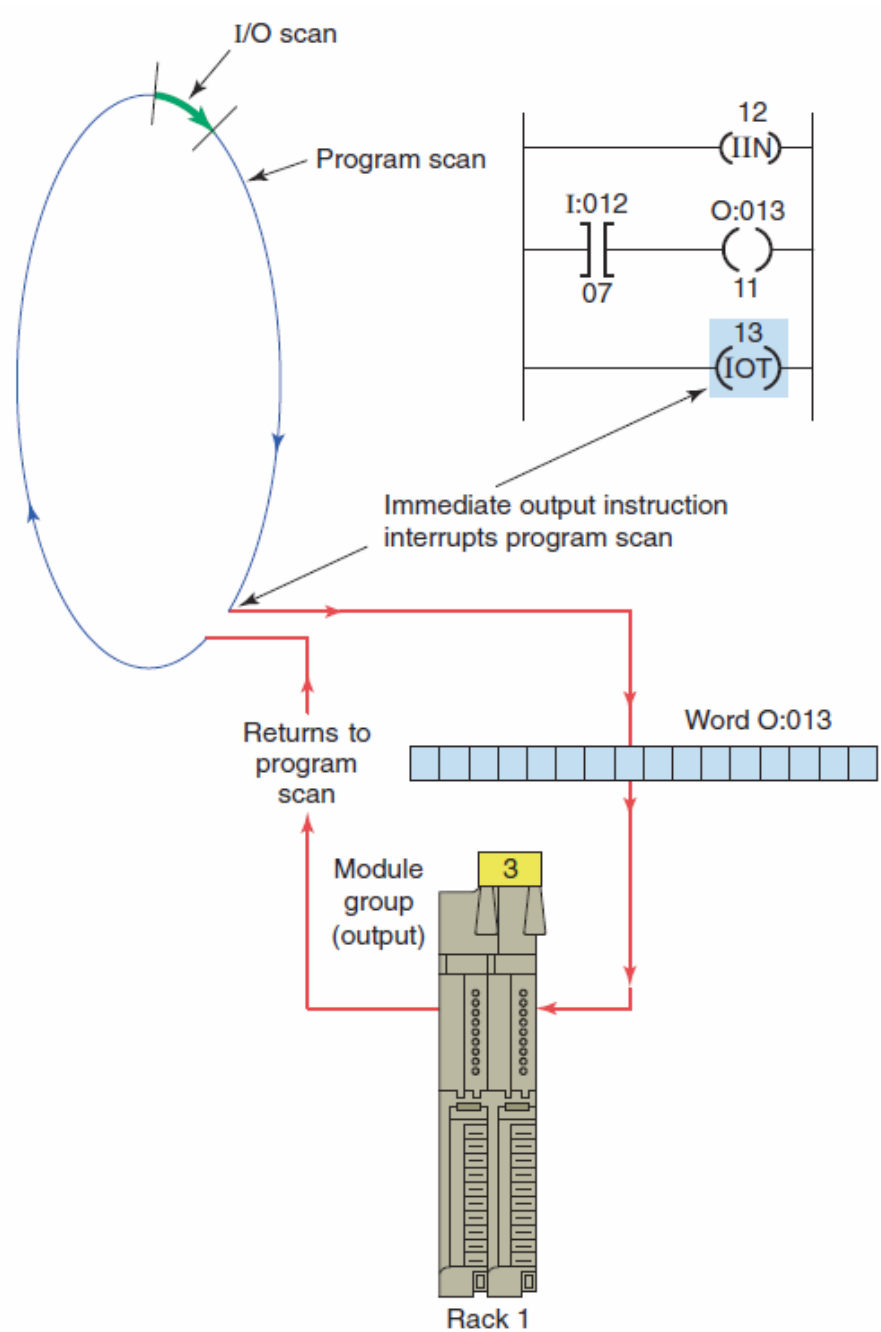
Immediate output (IOT)

PLC-5 instruction
is used to update the
status of an output
device **before** the I/O
update is performed.

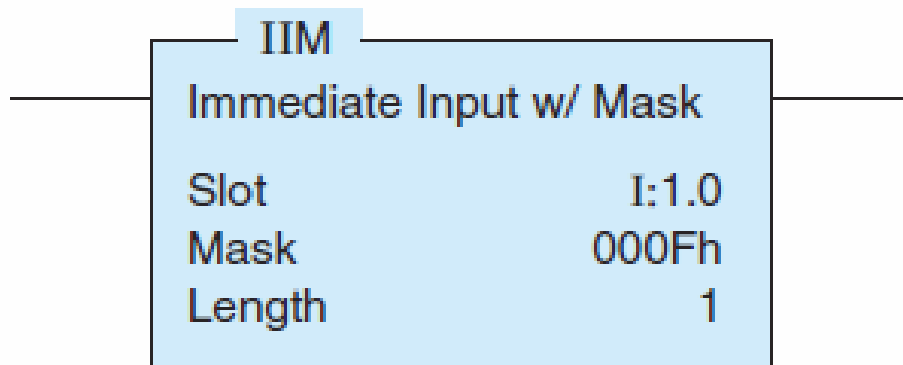
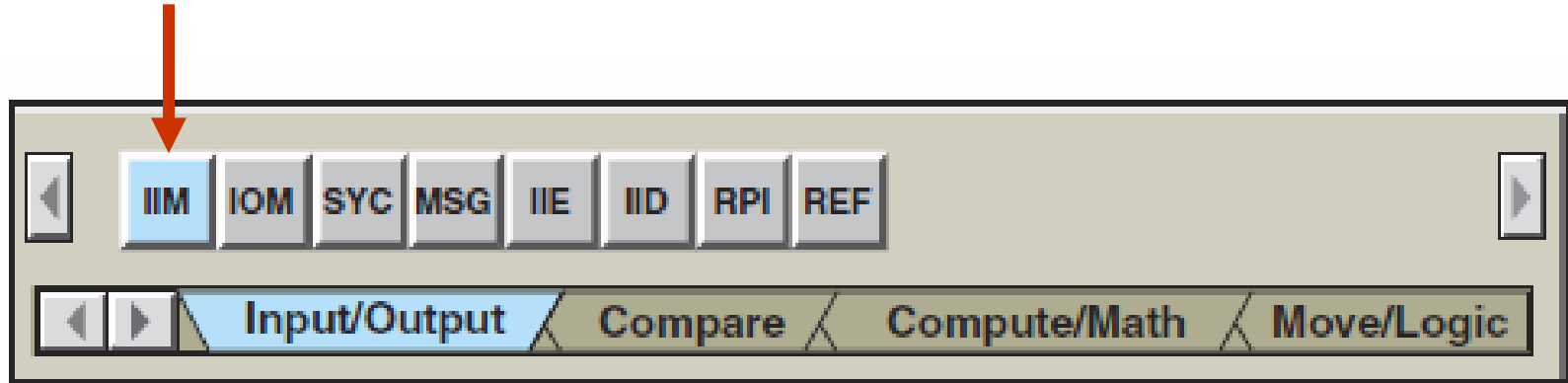
Immediate input instruction operation.



Immediate output instruction **operation.**

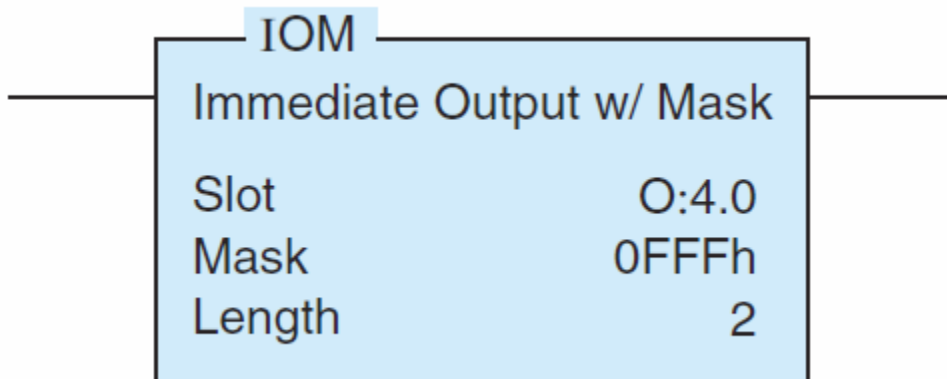
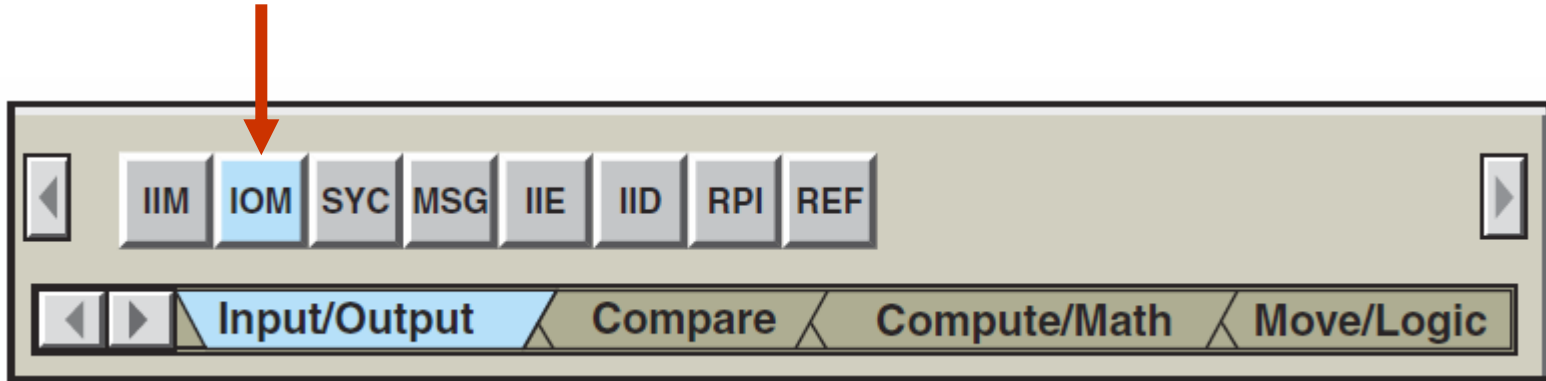


SLC 500 *Immediate Input with Mask (IIM)* instruction.



When the rung is true, the program scan is interrupted, and data from a specific input slot are transferred through the **mask to the input data file.**

SLC 500 *Immediate Output with Mask (IOM)* instruction.



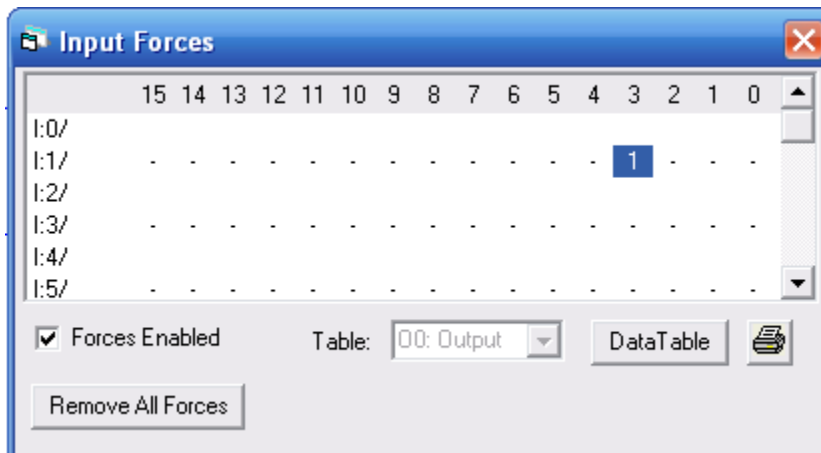
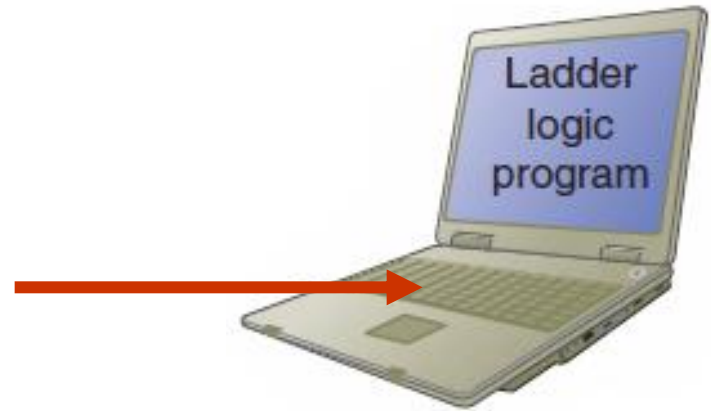
When the rung is true, the program scan is interrupted to update output data to the **module located in the slot specified in the instruction.**

9.5

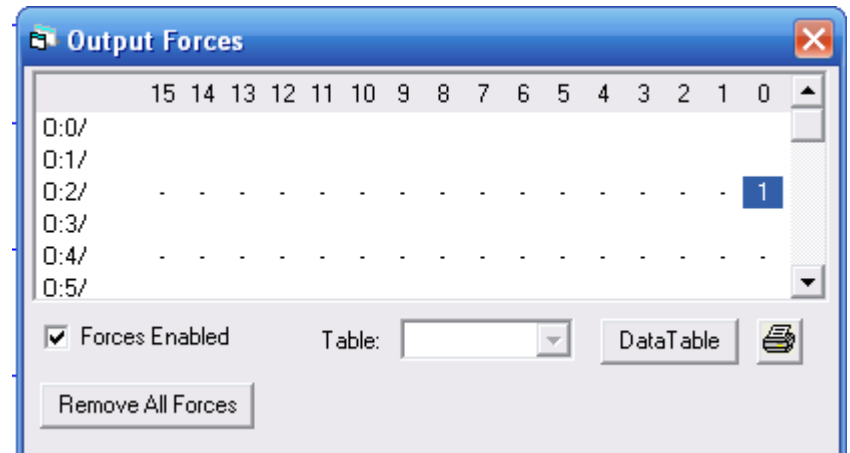


Forcing External I/O Addresses

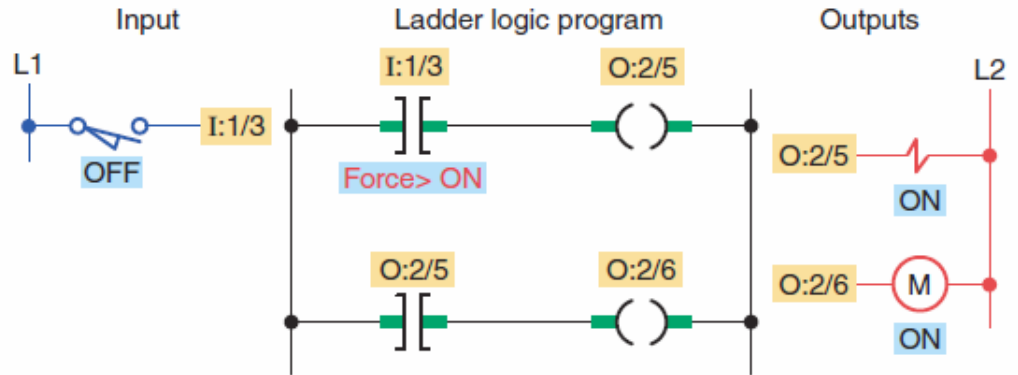
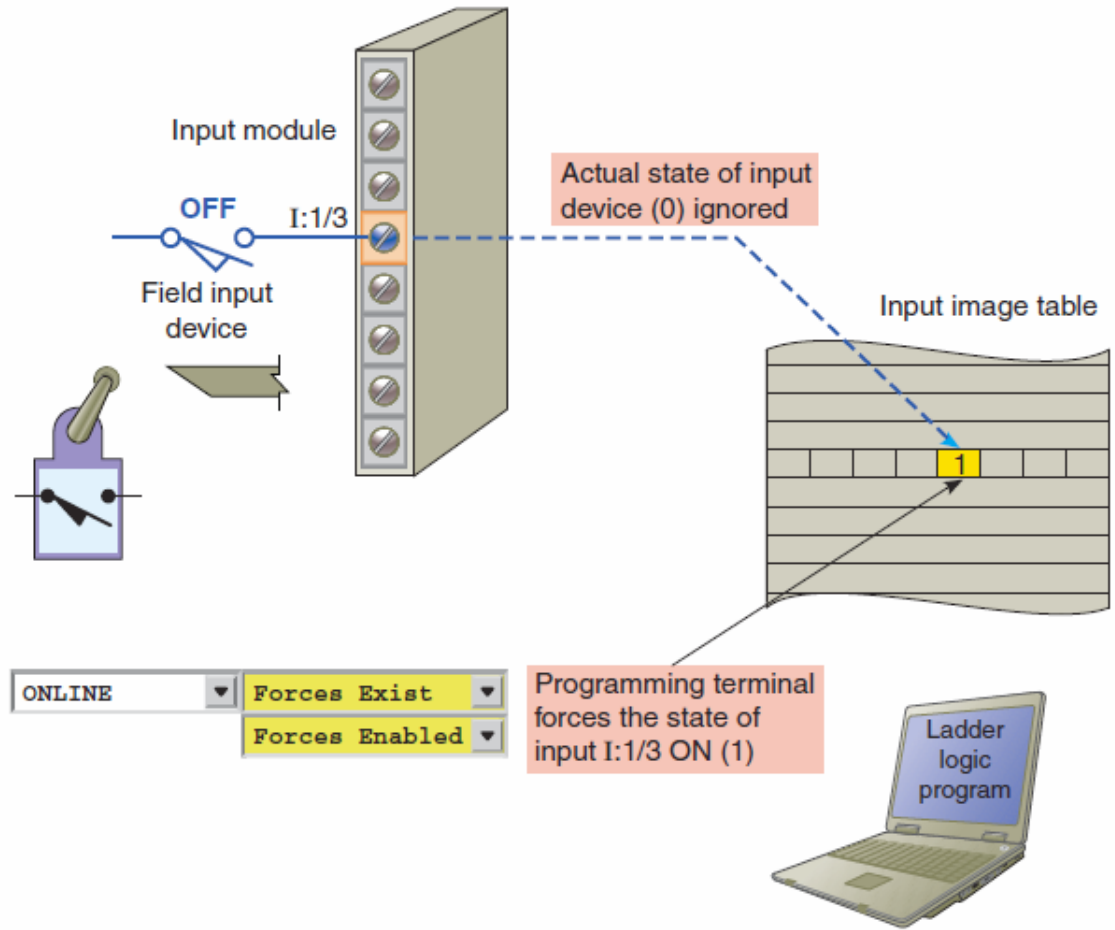
Forcing allows the PLC user to turn an external input or output *on or off* from the keyboard of the programming device.



Forcing is accomplished regardless of the actual state of the field device.



Forcing an input on.

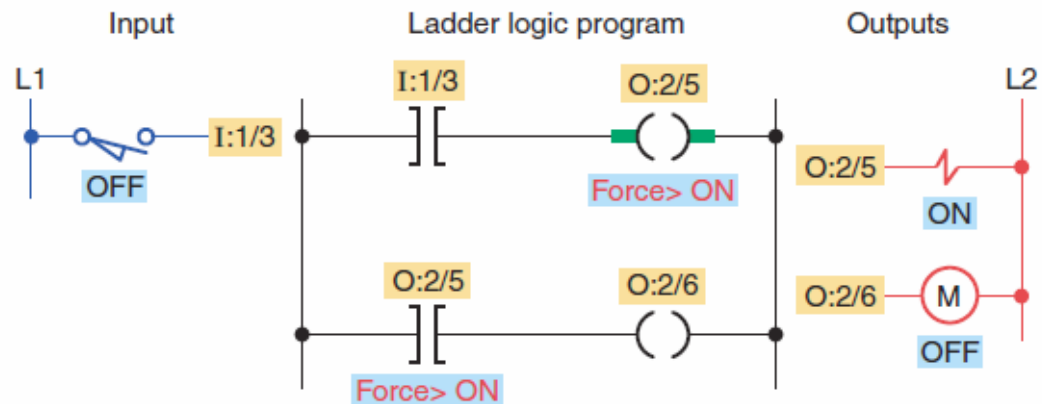
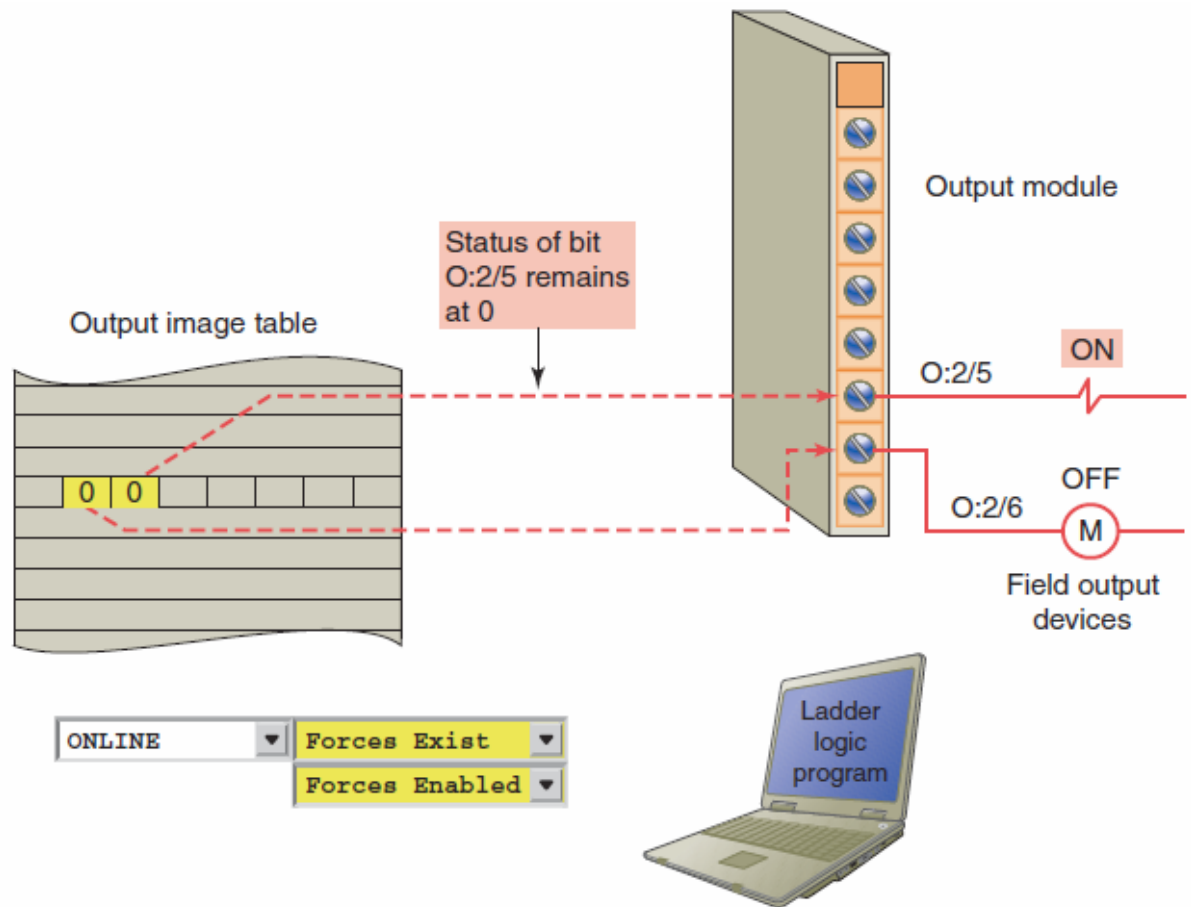


Simulating forcing an input on.

The screenshot displays the ProSim software interface for simulating a PLC. The top status bar indicates the PLC is **OnLine** and shows a **Download** button. The main window shows a ladder logic diagram (LAD 2) with three rungs. The first rung has a normally open contact labeled **I:1/3** connected to a coil labeled **O:2/5**. The second rung has a normally open contact labeled **O:2/5** connected to a coil labeled **O:2/6**. The third rung is labeled **002** and ends with a **(END)** symbol. On the left, the **I/O Simulation** panel shows a list of inputs (I:1 to I:16) and outputs (O:2 to O:16). The **Input Forces** dialog box is open, showing a table for forcing inputs. The table has columns for bit positions 15 down to 0. The rows are labeled **I:0/**, **I:1/**, **I:2/**, **I:3/**, **I:4/**, and **I:5/**. The **Forces Enabled** checkbox is unchecked. The **Table** dropdown is set to **00: Output**. A **Remove All Forces** button is at the bottom of the dialog.

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

Forcing an output on.



Simulating forcing an output on.

ONLINE No Forces
No Edits Forces Disabled
Driver: CSS DDE-Link Node: 1o
PLC = OnLine
Download
RUN 0000H 0000H 471 Scans
STEP 0000H 0000H
PGM 0000H 0000H

ProSi... LAD 2

I/O Simulation

I:1 O:2

00 00
01 01
02 02
03 03
04 04
05 05
06 06
07 07
08 08
09 09

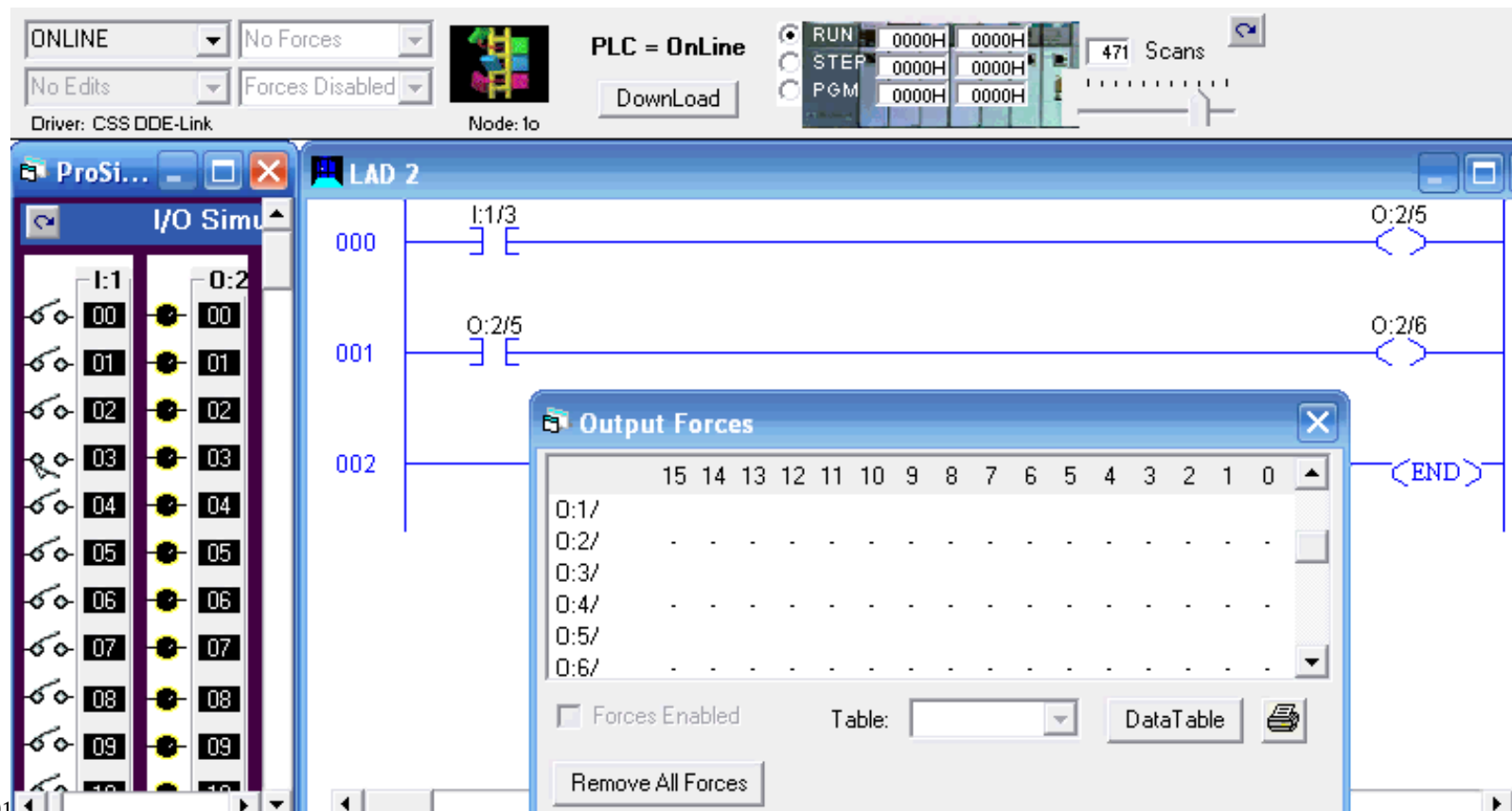
000 I:1/3 O:2/5
001 O:2/5 O:2/6
002 O:2/6 <END>

Output Forces

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

☐ Forces Enabled Table: DataTable
Remove All Forces

By forcing *outputs off*, you can prevent the controller from energizing those outputs even though the ladder logic, which normally controls them, may be true.



You can enter and enable or disable forces while you are monitoring your file offline, or in any processor mode while monitoring your file online.

The screenshot shows a software window titled "Data File I1 (bin) . . INPUT Forces". It contains a table with 16 columns labeled "Offset" from 15 down to 0. Bit 3 (offset 4) is circled in blue. The row for "I:1.0" has a red "1" in the column for offset 4. The row for "I:2.0" has dots in the last four columns. Below the table are navigation arrows, a text field containing "I:1.0/3", a "Radix:" dropdown, a "Symbol:" text field, a "Columns:" dropdown, and a "Desc:" text field. At the bottom are four buttons: "Enable", "Remove All", "Data File", and "Help".

Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I:1.0	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.
I:2.0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

**Forces version of the data table with bit
I:1/3 forced on.**

If used *incorrectly*, force functions can cause injuries to persons working around a system, and/or equipment damage.



You must understand the *potential effect* that forcing given inputs or outputs will have on machine operation in order to avoid possible *personal injury* and *equipment damage*.

9.6



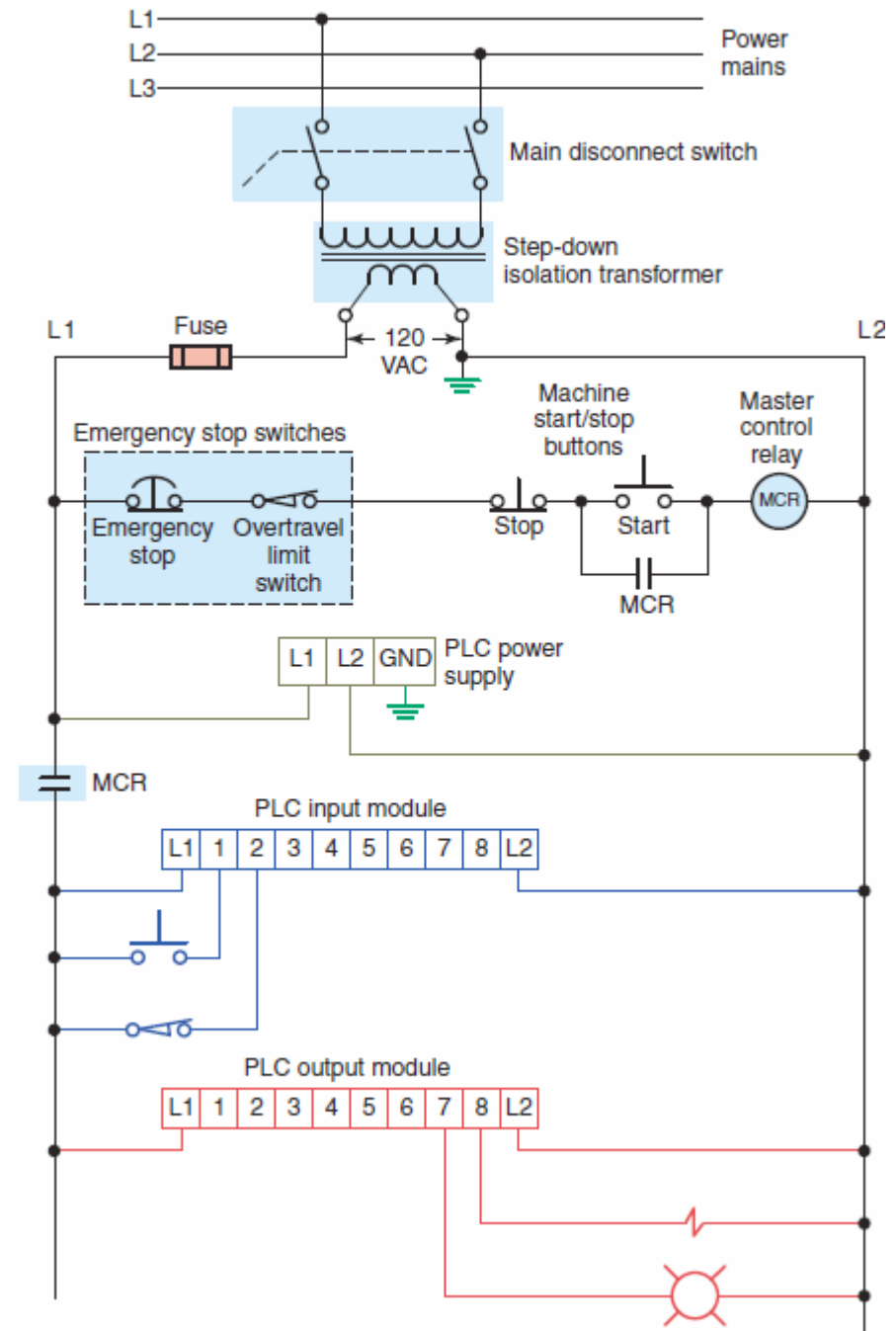
Safety Circuitry

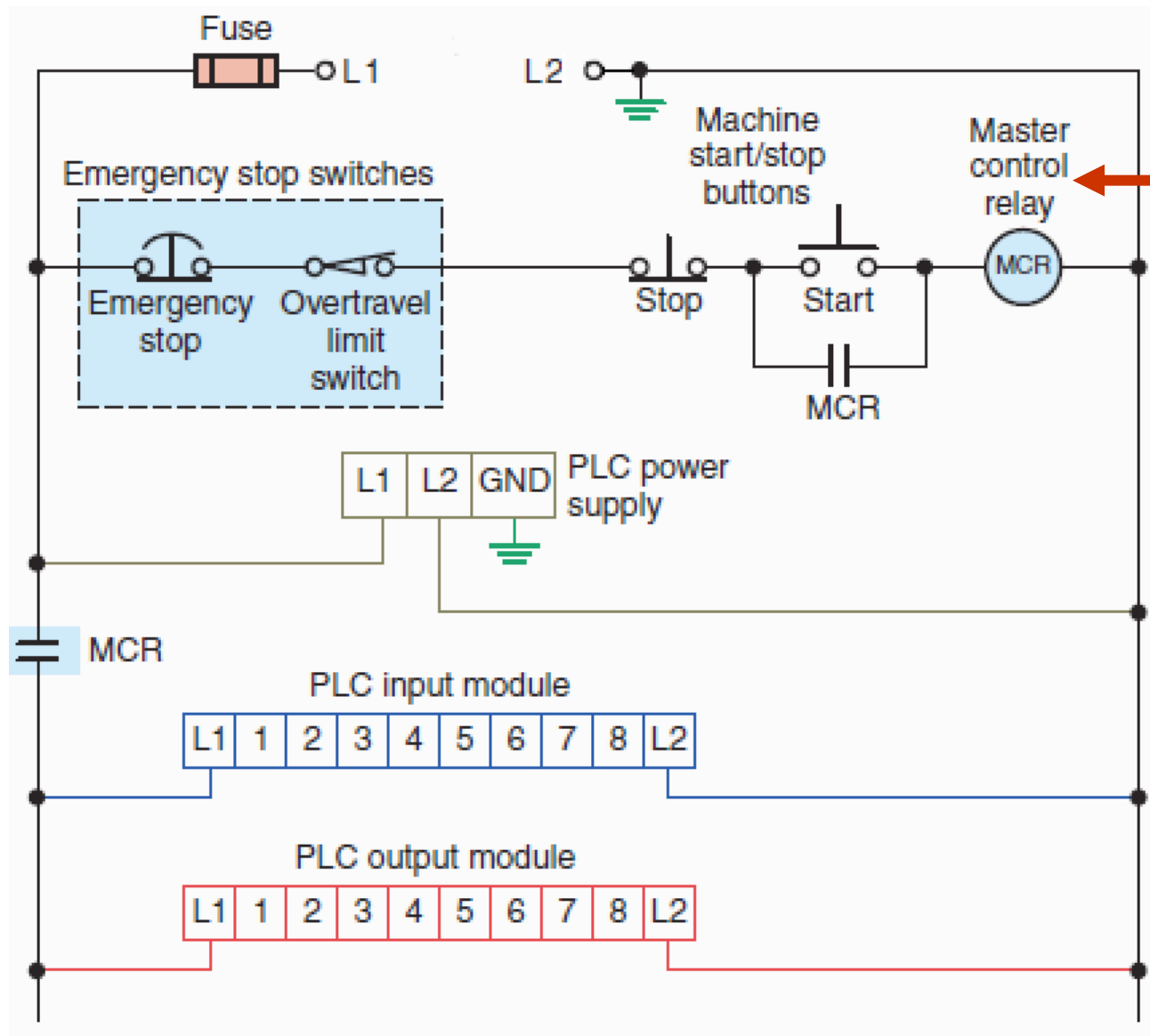
Sufficient emergency circuits must be provided to stop either *partially or totally* the operation of the controller or the controlled machine or process.

These circuits should be **hardwired** outside the controller.



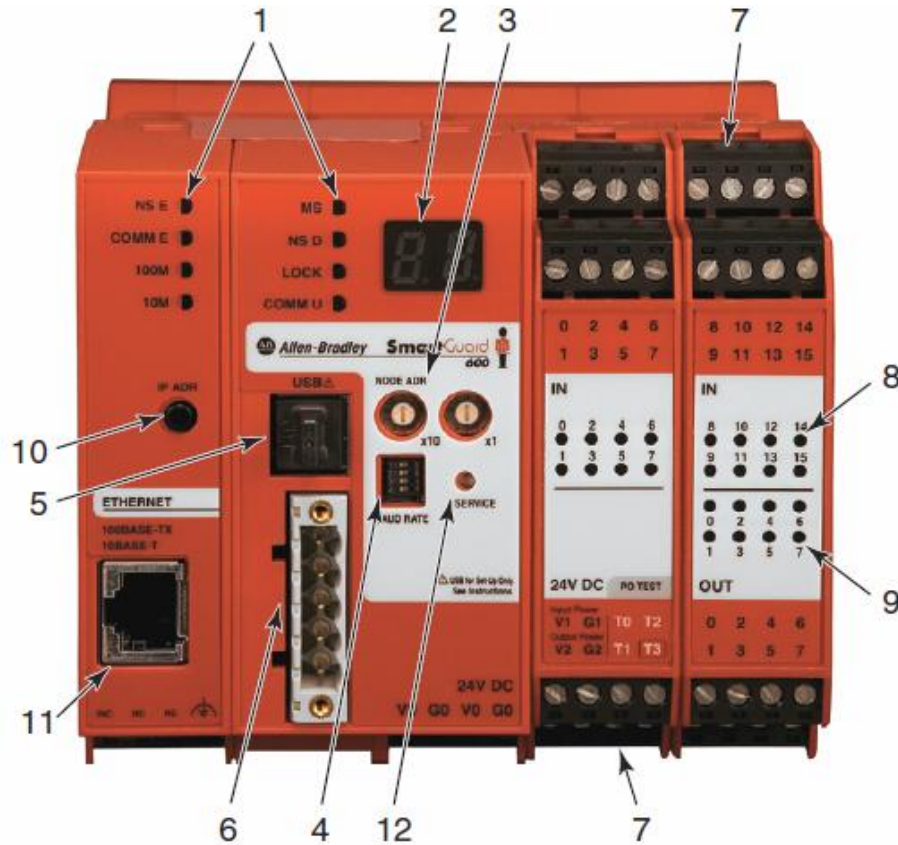
Safety **wiring** requirements for a PLC installation.





The **master control relay** must be able to inhibit all machine motion by removing power to the machine **I/O devices** when the relay is deenergized.

Safety PLCs are now available for applications that require more advanced safety functionality.

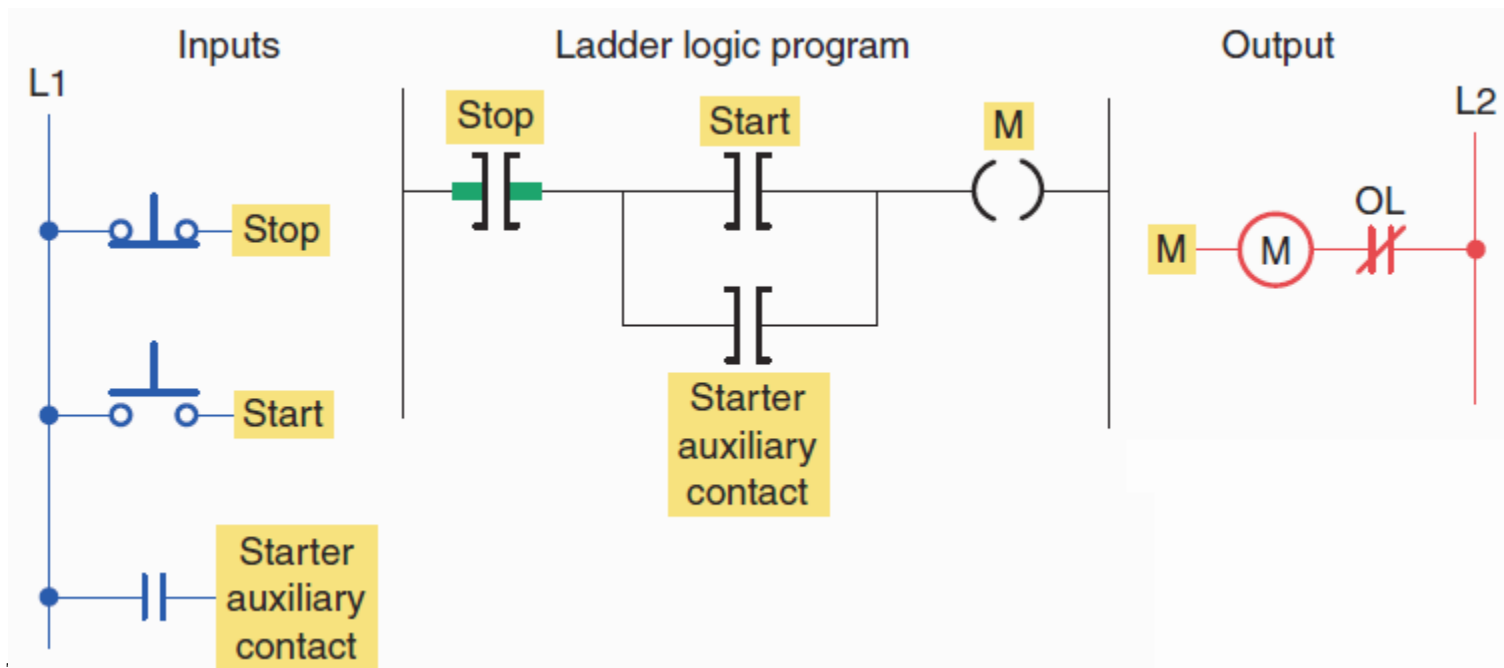
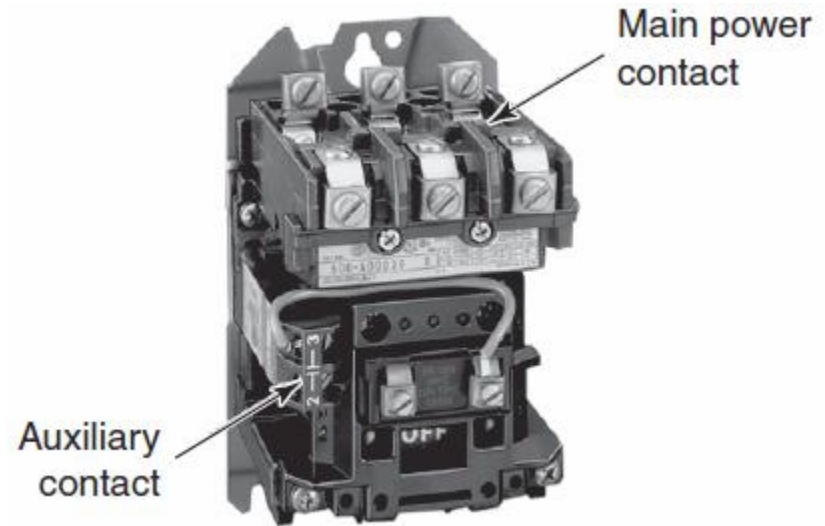


Number	Feature
1	Module status indicators
2	Alphanumeric display
3	Node address switches
4	Baud rate switches
5	USB port
6	DeviceNet communication connector
7	Terminal connectors
8	Input status indicators
9	Output status indicators
10	IP address display switch
11	Ethernet connector
12	Service switch

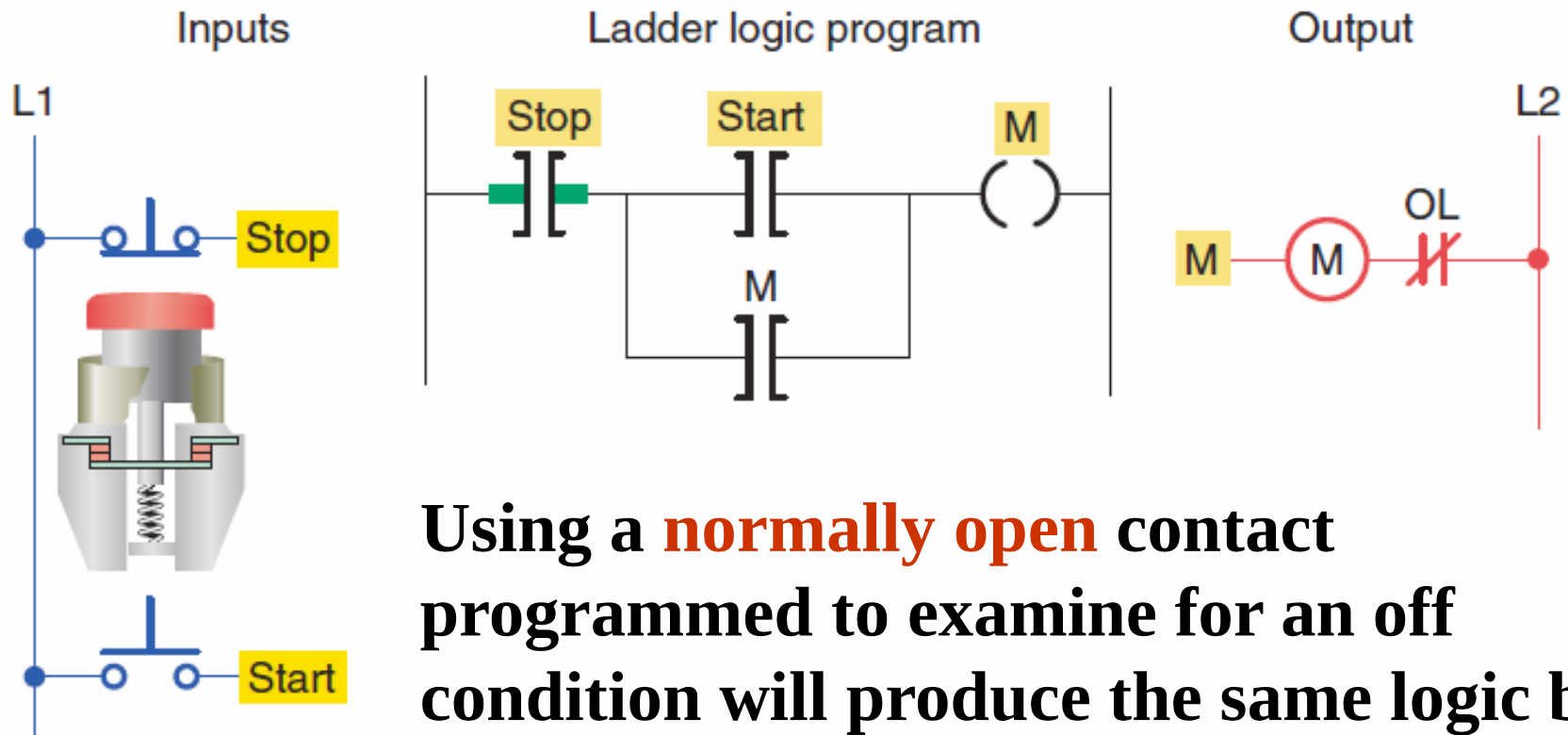
Both standard and safety PLCs have the ability to perform control functions but a standard PLC was not initially designed to be *fault tolerant and fail-safe*.

- **Safety PLCs have redundant (dual) microprocessors.**
- **Safety PLCs have an internal output circuit associated with each input for the purpose of testing the input circuitry.**
- **Safety PLCs use power supplies designed specifically for use in safety control systems and redundant backplane circuitry between the controller and I/O modules.**

The use of the field-generated starter auxiliary contact is *safer* because it provides positive feedback to the processor about the exact status of the motor.



All stop buttons should be wired using a *normally closed* contact programmed to examine for an *on* condition.



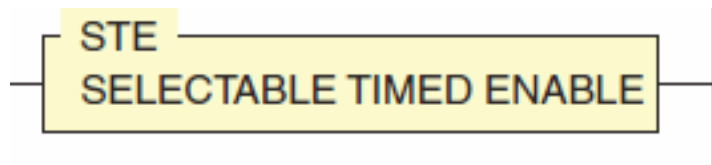
Using a **normally open** contact programmed to examine for an off condition will produce the same logic but is **not considered to be as safe**.

9.7



Selectable Timed Interrupt

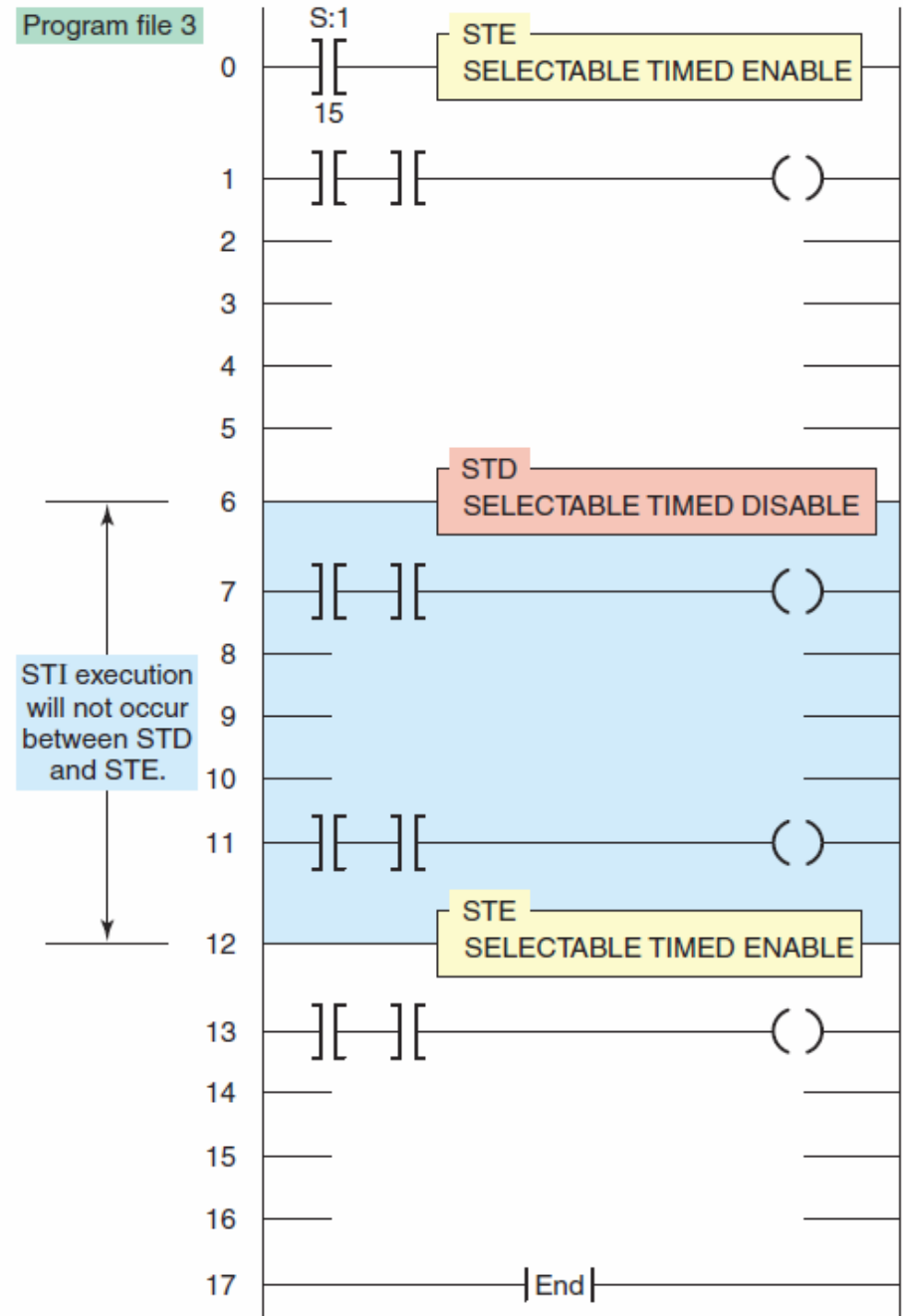
The *selectable timed interrupt (STI)* instruction is used to *interrupt* the scan of the main program file automatically, on a *time basis*, to scan a *specified subroutine file*.



The *selectable timed disable (STD)* instruction is generally paired with the *selectable timed enable (STE)* instruction to create zones in which STI interrupts cannot occur.



The *selectable timed disable (STD)* instruction is generally paired with the *selectable timed enable (STE)* instruction to create zones in which STI interrupts cannot occur.



9.8



Fault Routine

A fault subroutine file, if used, determines how the processor responds to a programming error.

➤ When the processor detects a **major fault**, it looks for a fault routine. If a fault routine exists, it is executed; if one does not exist, the processor shuts down.

➤ When there is a fault routine, and the fault is **recoverable**, the fault routine is **executed**.

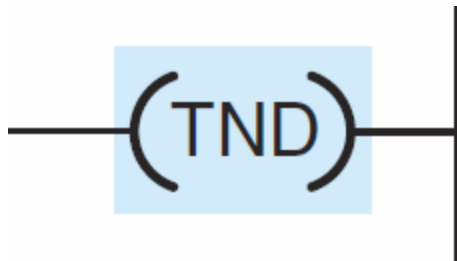
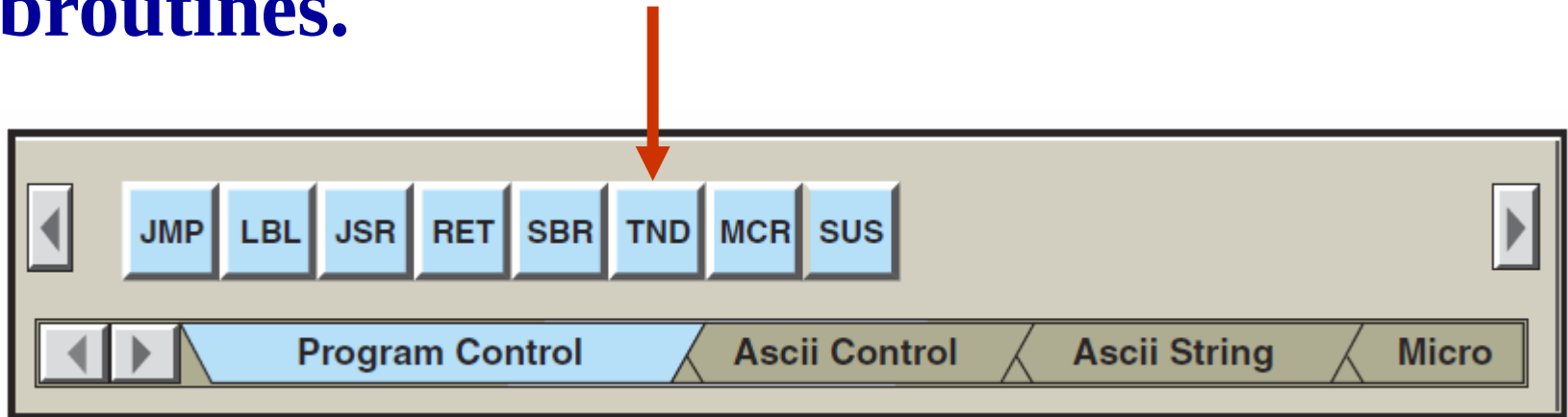
➤ If the fault is **non-recoverable**, the fault routine is scanned once and shuts down.

9.9



Temporary End Instruction

The *temporary end (TND)* instruction is used to progressively debug a program or conditionally omit the balance of your current program file or subroutines.

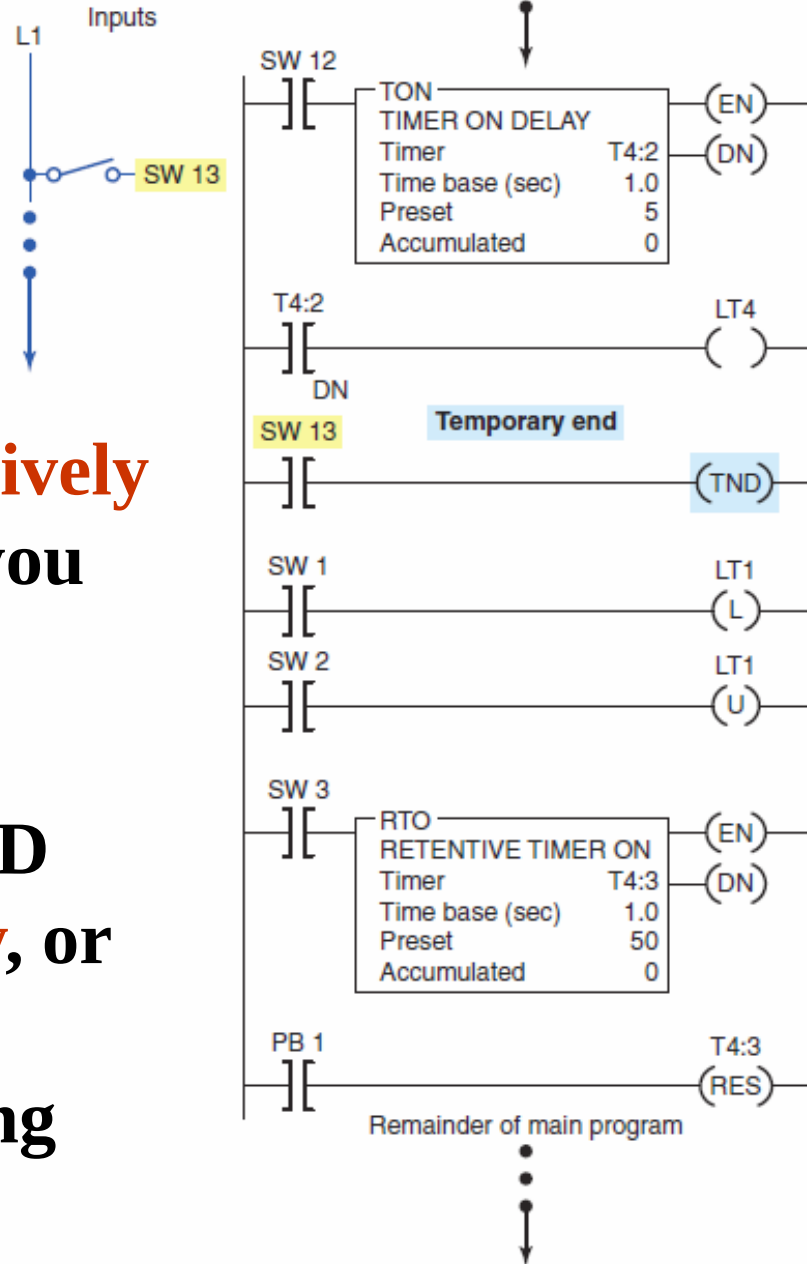


When rung conditions are **true**, this instruction **stops** the program scan, **updates** the I/O, and **resumes** scanning at rung **0** of the main program file.

➤ The TND instruction lets your program run **only up to this instruction.**

➤ You can **move it progressively** through your program as you debug each new section.

➤ You can program the TND instruction **unconditionally**, or you can **condition** its rung according to your debugging needs.



Simulated temporary end instruction program.

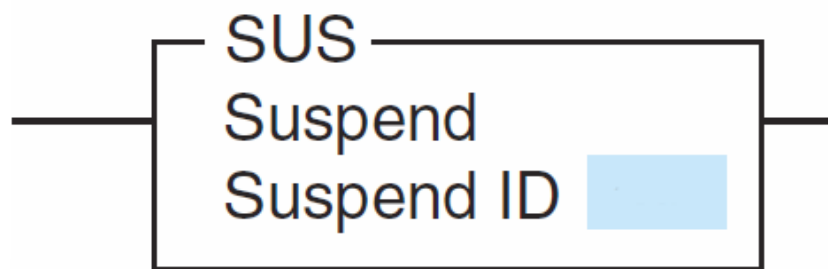
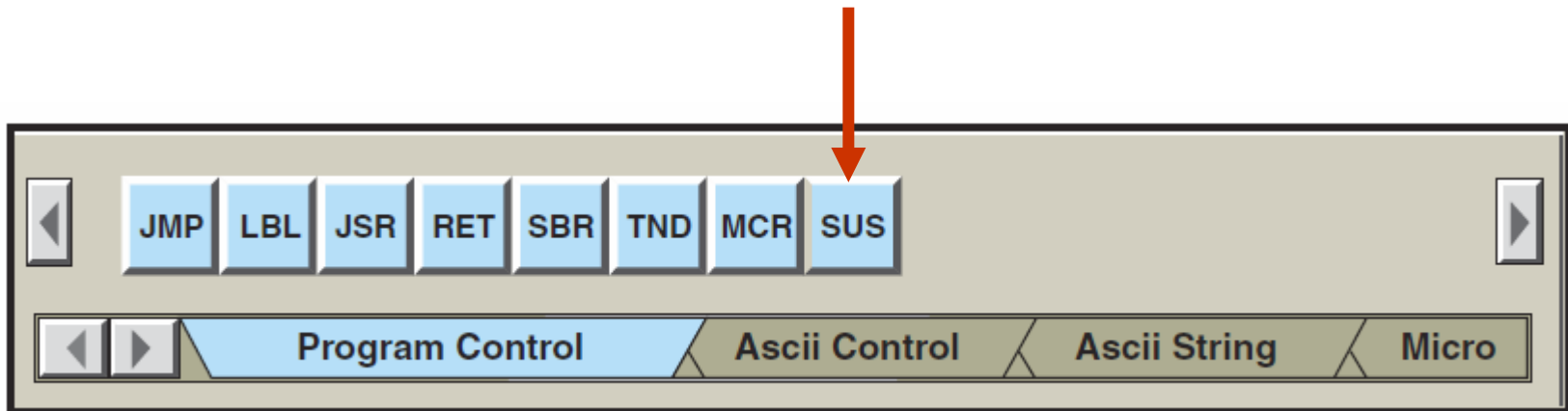


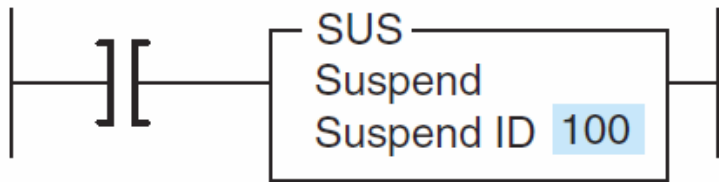
9.10



Suspend Instruction

The *suspend* (*SUS*) instruction is used to trap and identify specific conditions during system troubleshooting and program debugging.





➤ When the rung is **true**, the SUS output instruction places the controller in the suspend mode and the PLC immediately **terminates scan** cycling.

➤ All ladder logic **outputs** are **de-energized**, but other status files have the data present when the suspend instruction is executed.

➤ The SUS instruction writes the suspend ID number (**100**) to S:7 as it executes.

➤ You can include several SUS instructions in a program, each with a **different suspend ID**.