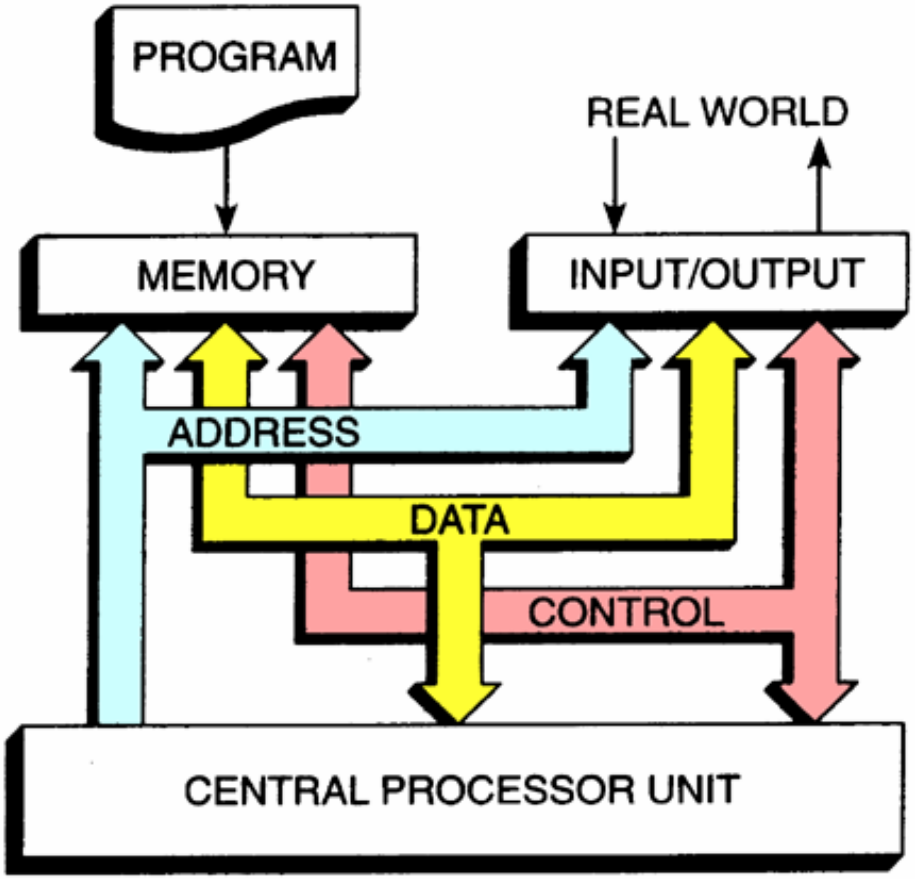
Basics of PLC Programming

Industrial Control Systems

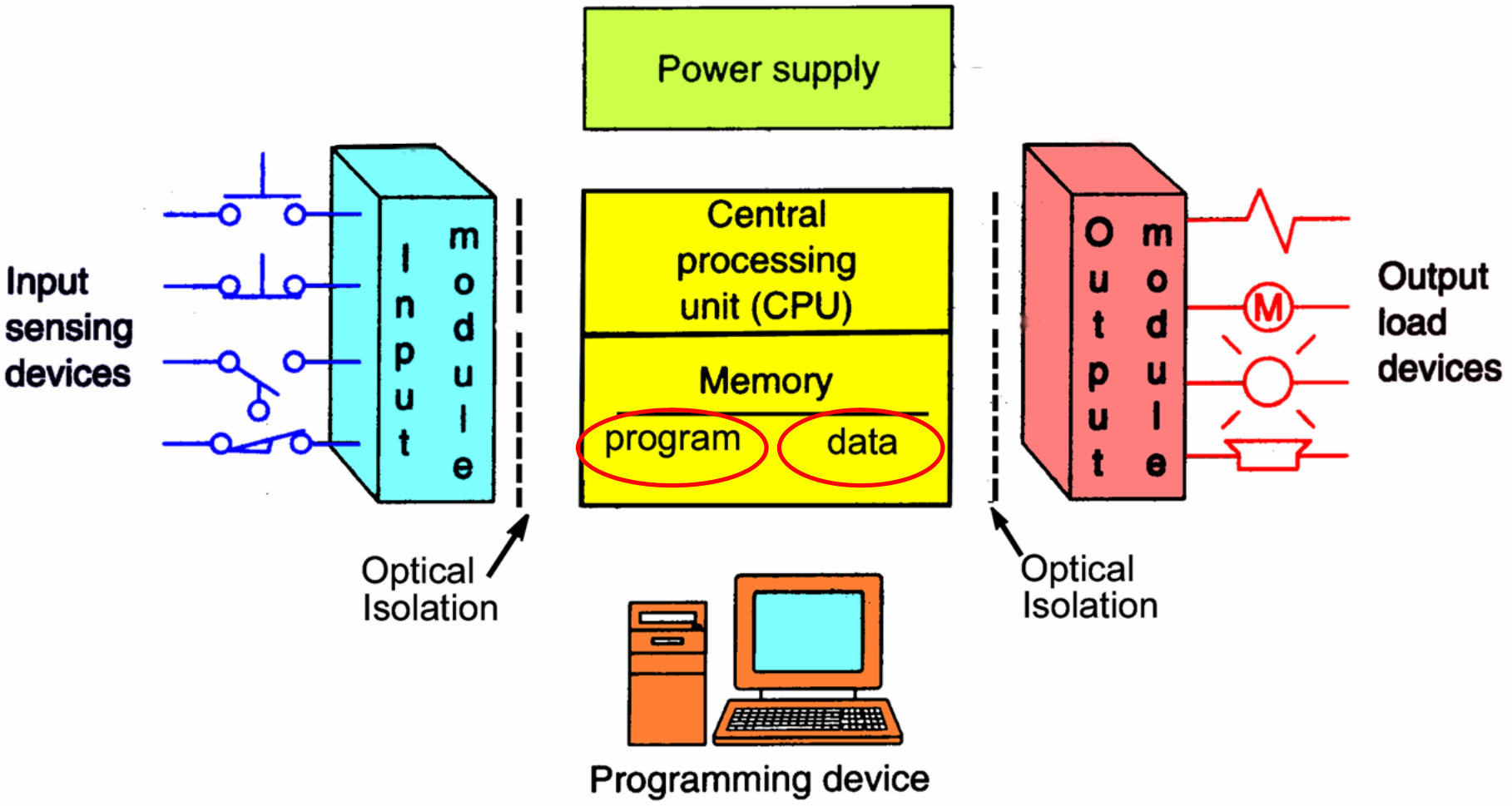
Fall 2006

PLC Architecture



The structure of a PLC is based on the same principles as those employed in computer architecture.

PLC System


Power supply
Central processing unit (CPU)
Memory
program
data
Input module
Output module
Input sensing devices
Output load devices
Optical Isolation
Optical Isolation
M
Programming device

Processor Memory Organization

The memory of a PLC is organized by types.

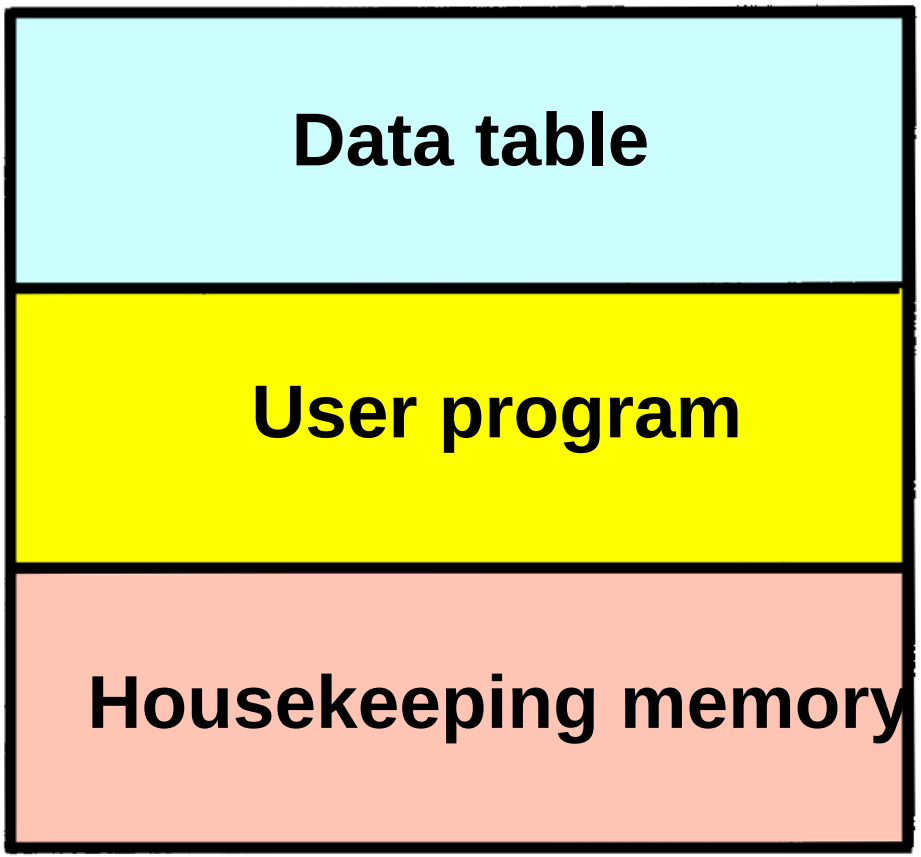
The memory space can be divided into two broad categories:

***program* and *data* memory.**

Advanced ladder logic *functions* allow controllers to perform calculations, make decisions and do other complex tasks. Timers and counters are examples of ladder logic functions. They are more complex than basic inputs contacts and output coils and rely heavily upon data stored in the memory of the PLC.

Memory Map

A memory map can be used to show how memory is organized in a PLC.

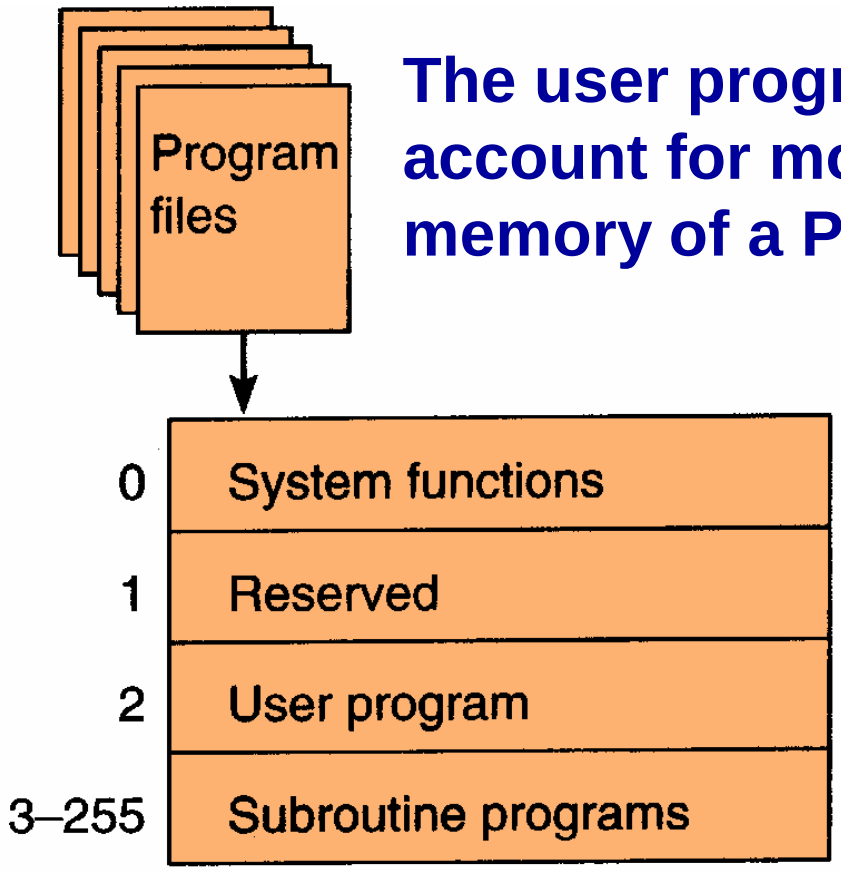


- **Input/output locations**
- **Internal relay and timer/counter locations**

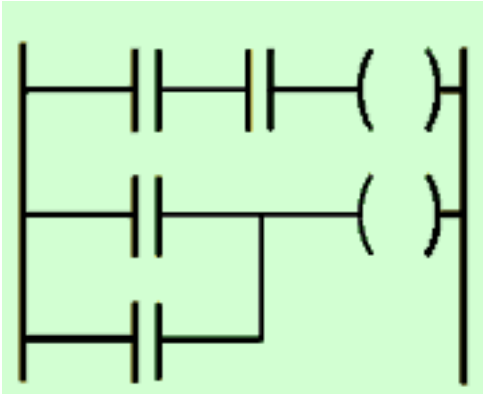
The user program causes the controller to operate in a particular manner

Used to carry out functions needed to make the processor operate (no access by user)

Program Files



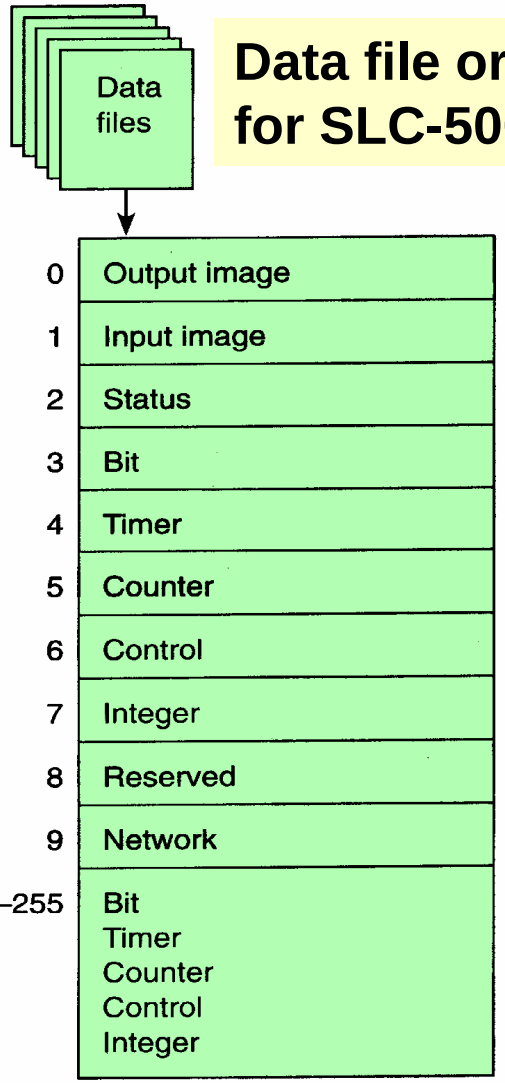
The user program will account for most of the memory of a PLC system.



Program file organization for SLC-500 controller.

- **Program files contain the logic controlling machine operation.**
- **This logic consists of instructions that are programmed in a ladder logic format.**

Data Files

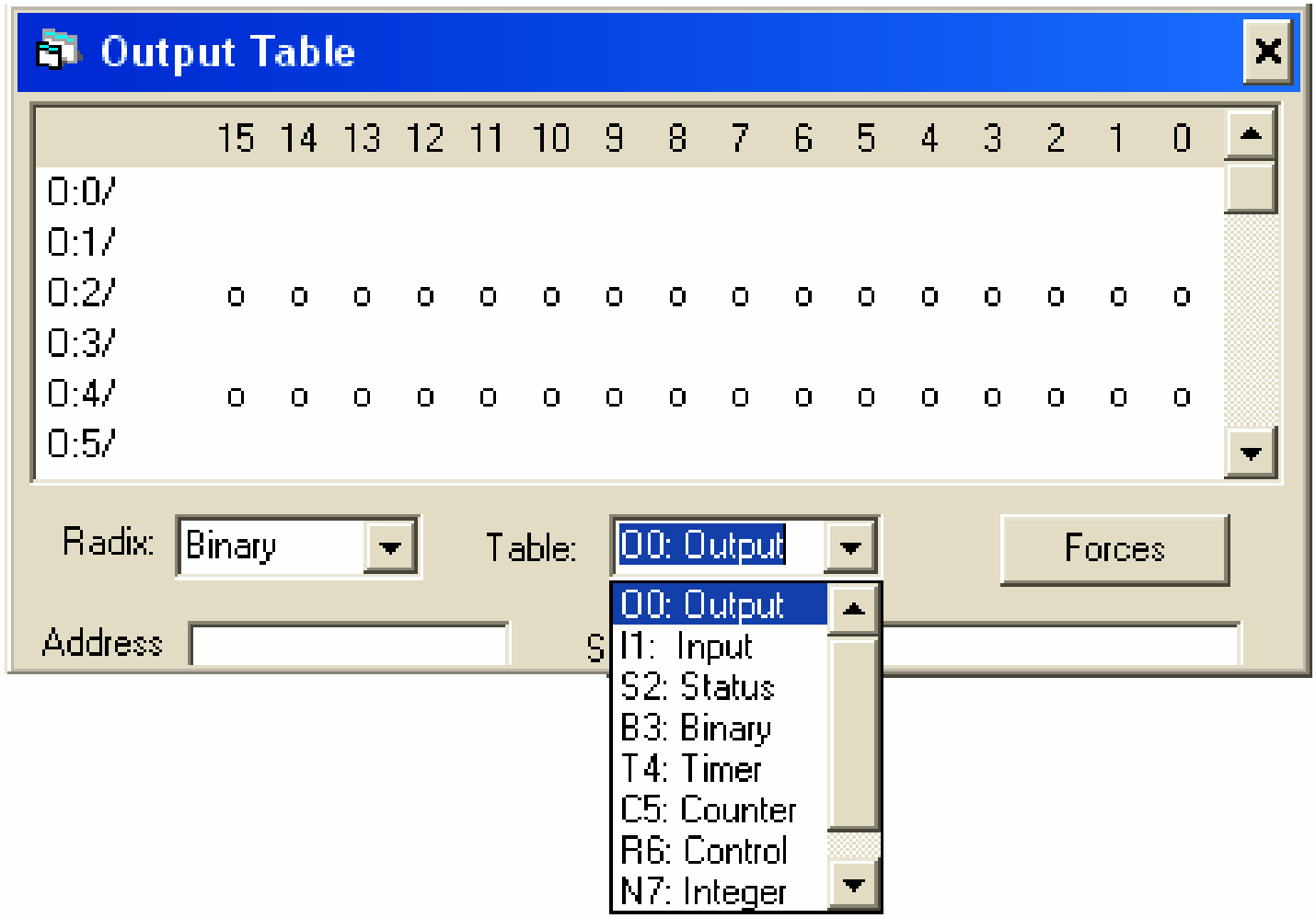


Data file organization for SLC-500 controller.

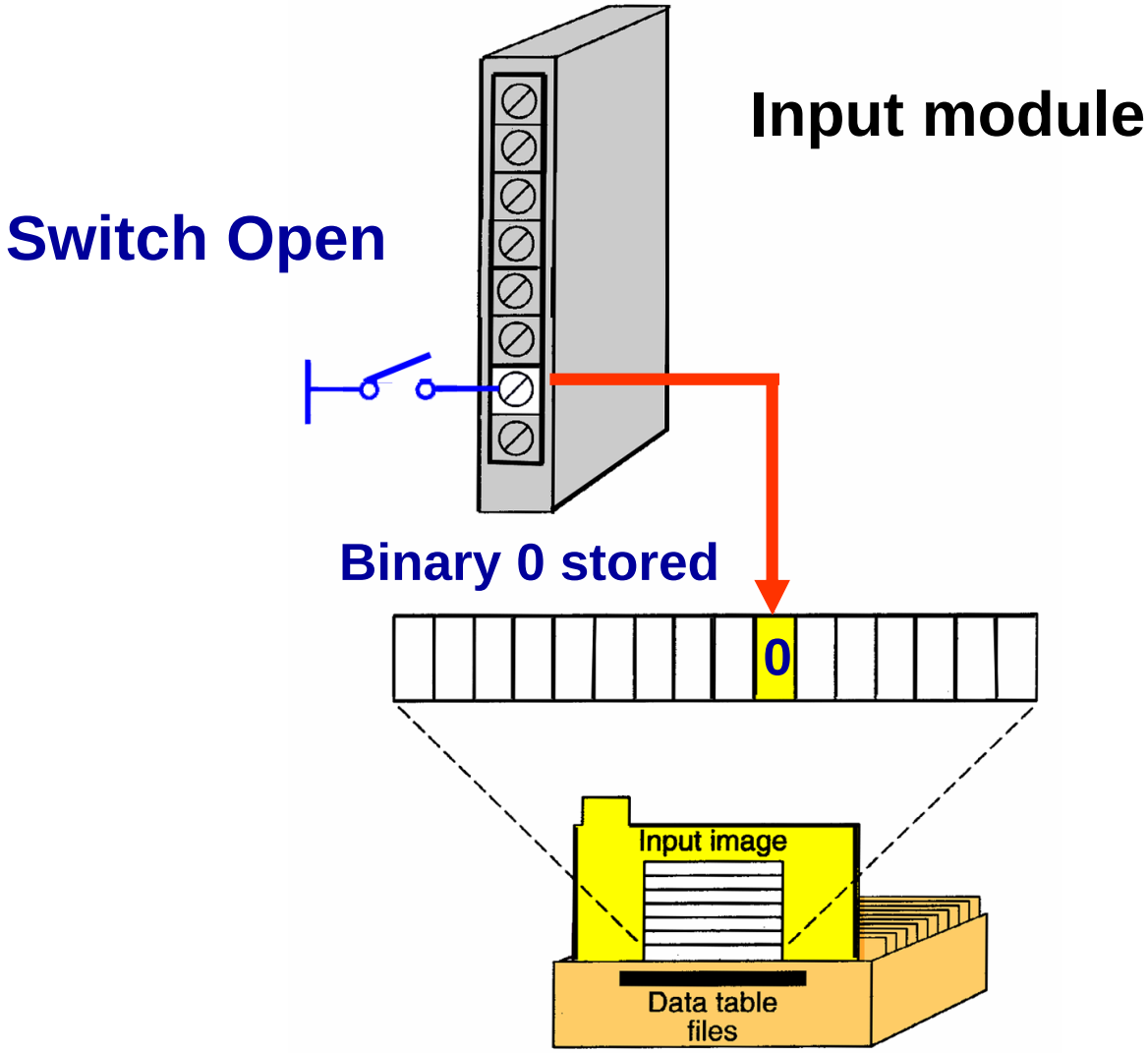
The data file portion of memory stores input and output status, processor status, the status of various bits and numerical data.

Data Files

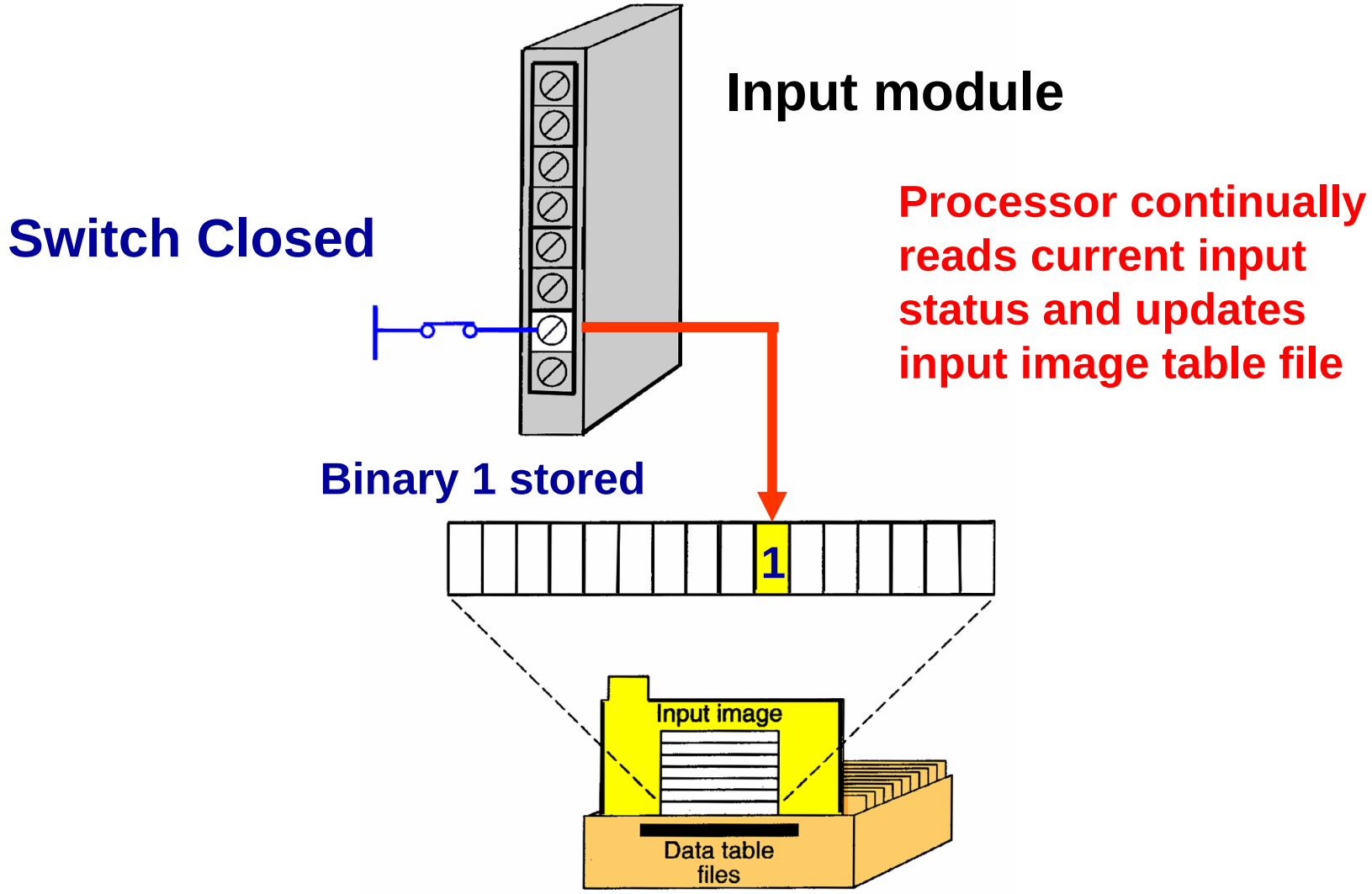
Data files are organized by the type of data they contain



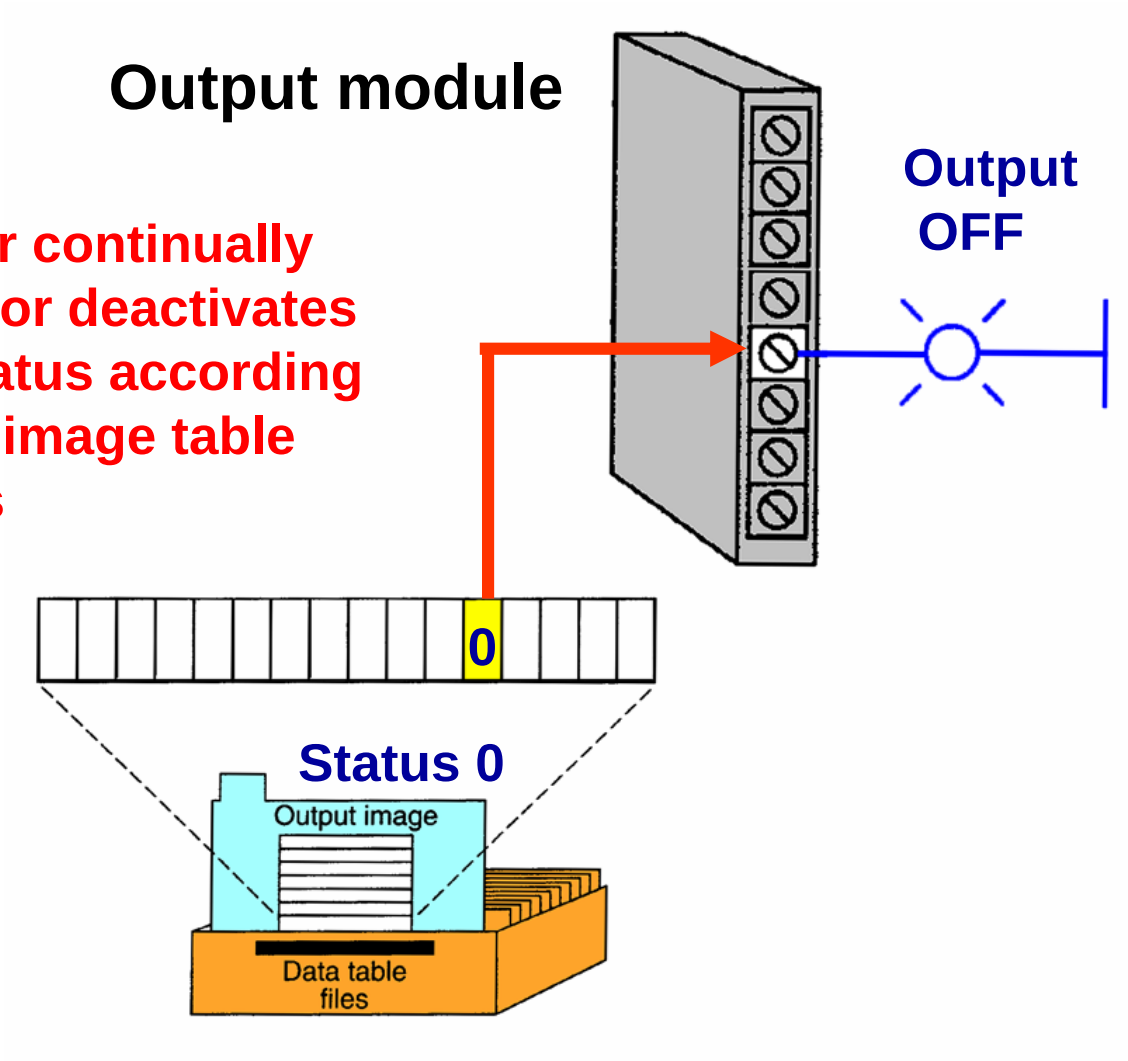
Input Table File Operation



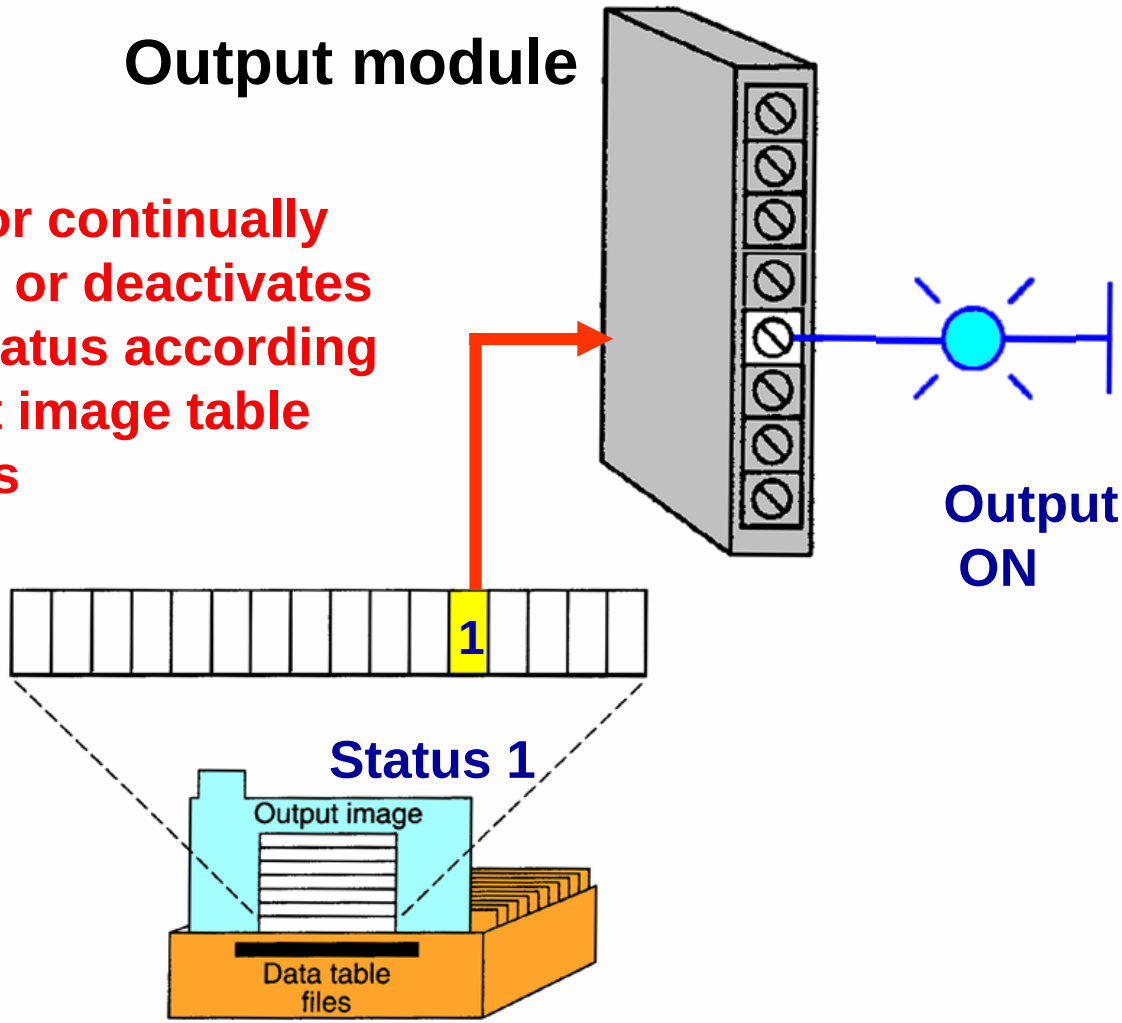
Input Table File Operation



Output Table File Operation

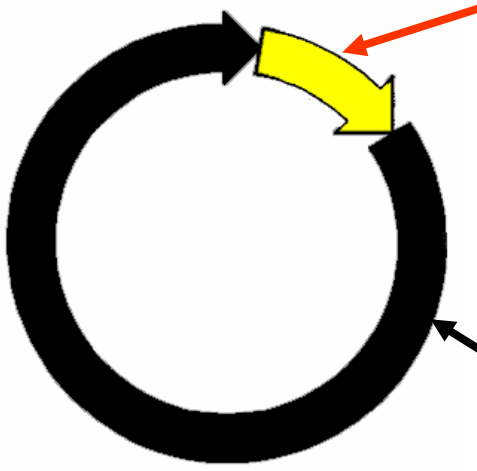


Output Table File Operation



Program Scan

During each operating cycle, the processor reads all inputs, takes these values, and energizes or de-energizes the outputs according to the user program. This process is known as a *scan*.



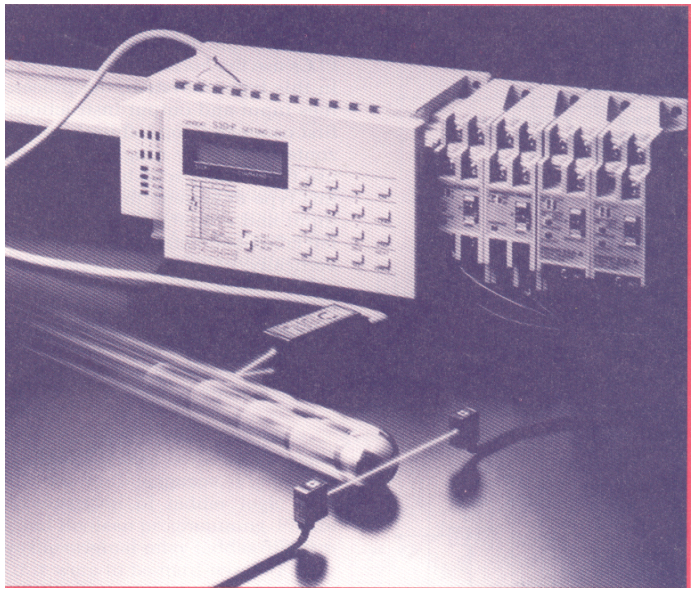
I/O scan – records status data of input devices. Energizes output devices that have their associated status bits set to ON (1)

Program scan – instructions are executed sequentially

Because the inputs can change at any time, the PLC must carry on this process continuously.

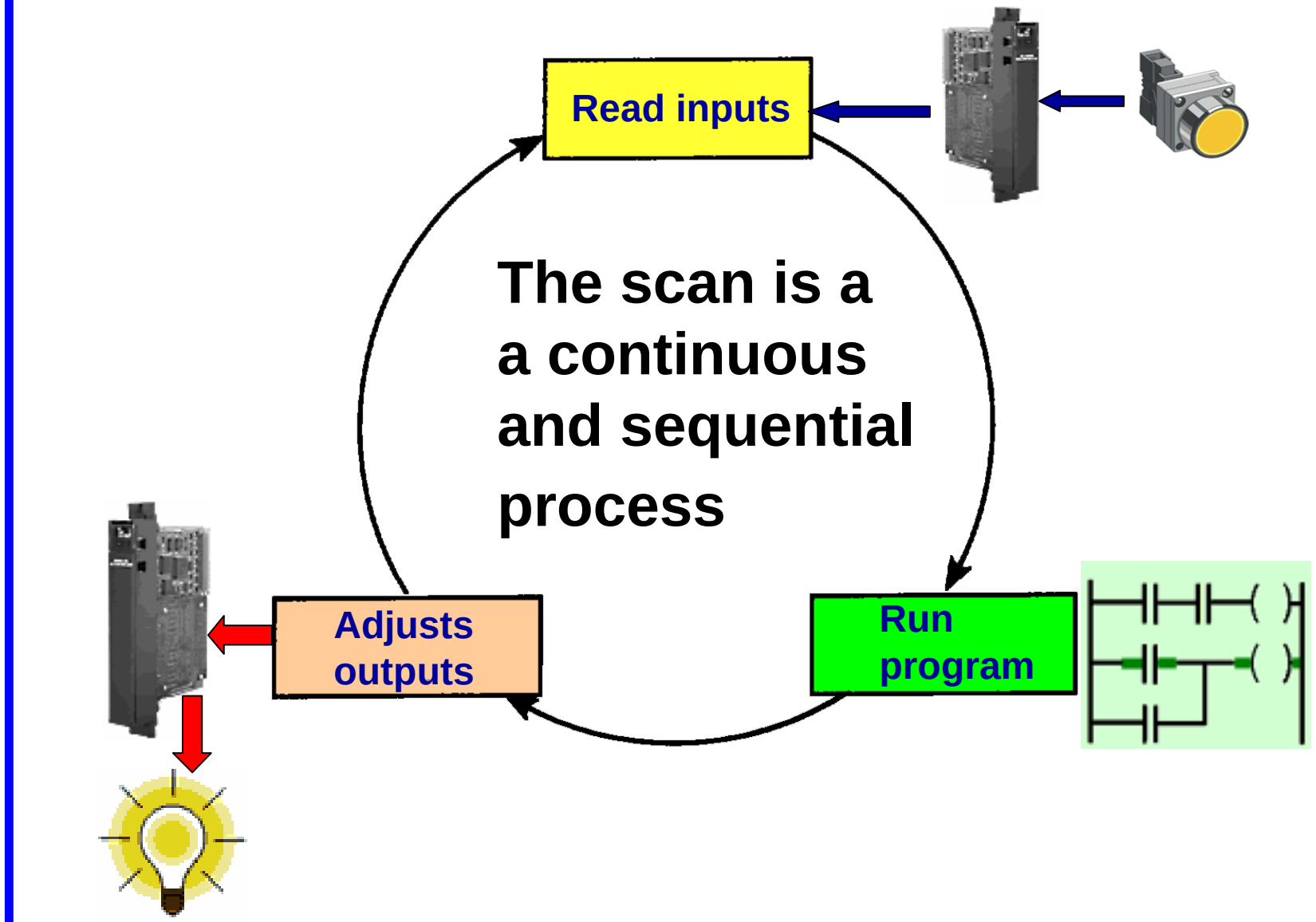
Scan Process

The scan *time* indicates how fast the controller can react to changes in inputs. Scan times vary with computer model and program content, and length. If a controller has to react to an input signal that changes states twice during the scan time, it is is possible that the PLC will never be able to detect this change.

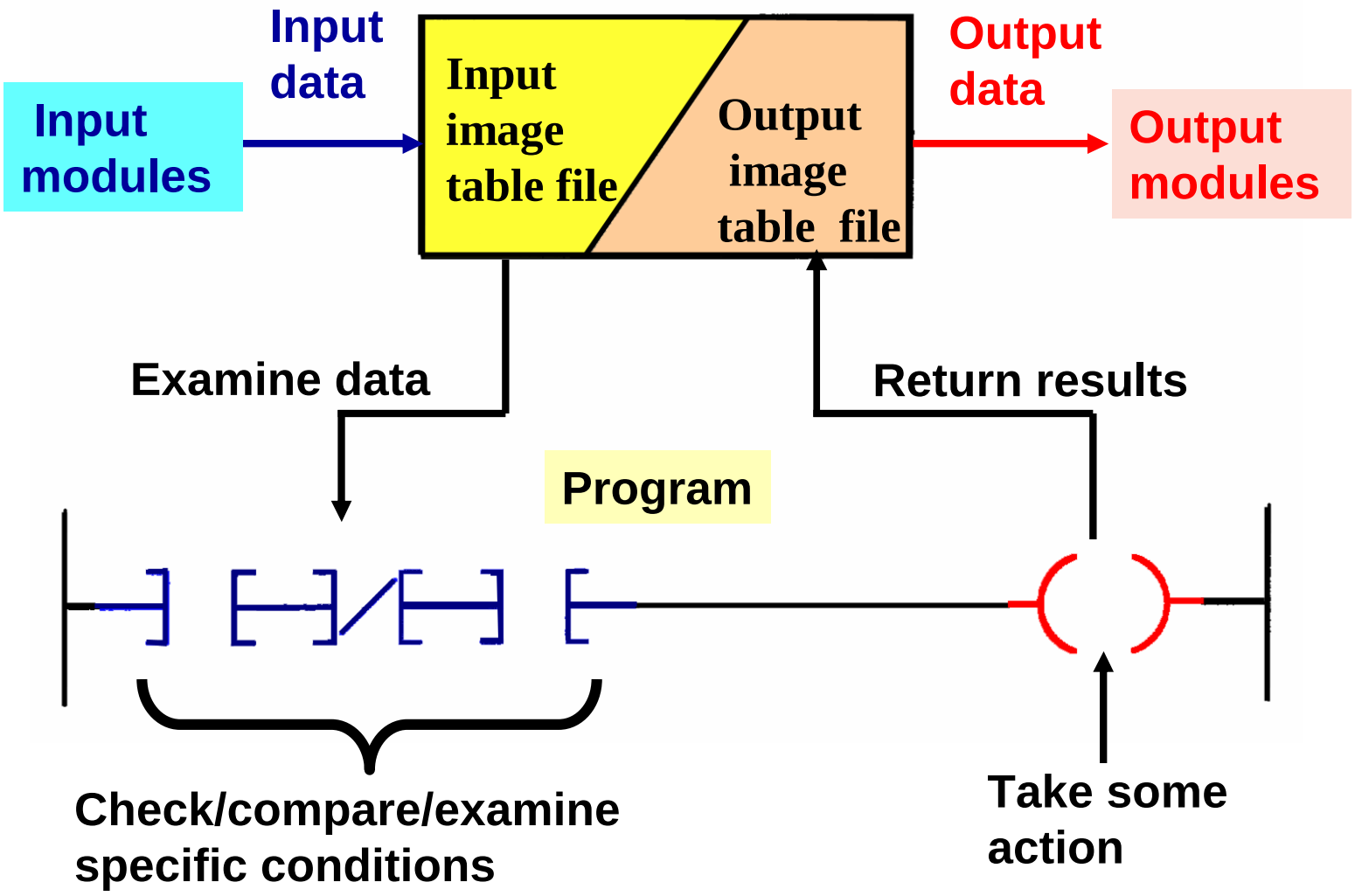


Scan time may be a concern in high speed operations

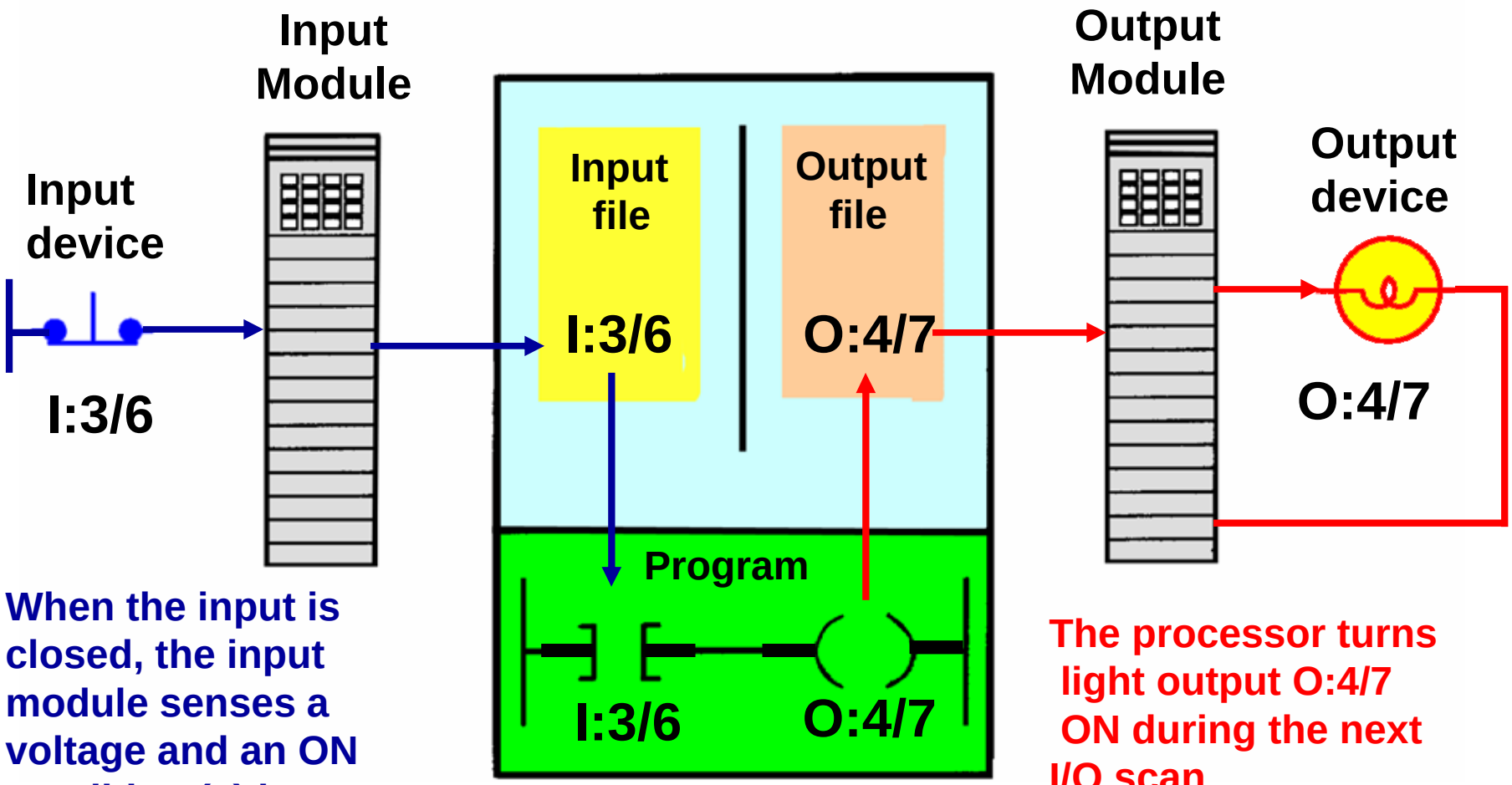
Scan Process



Data Flow Overview



Scan Process

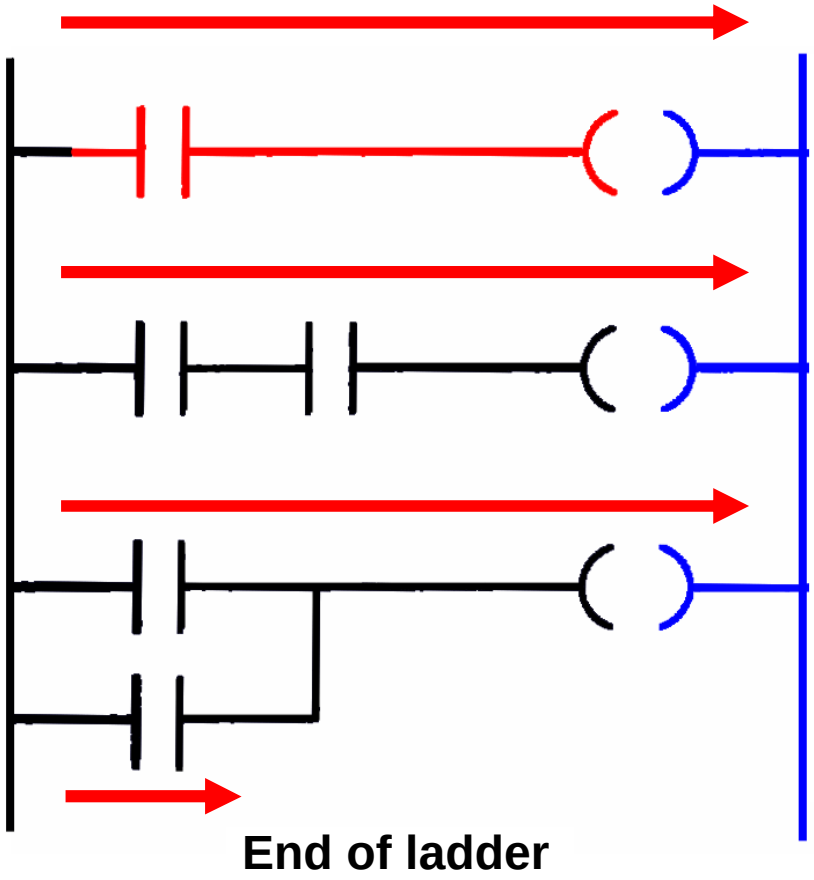


When the input is closed, the input module senses a voltage and an ON condition (1) is entered into the input table bit I:3/6

During the program scan the processor sets instructions I:3/6 and O:4/7 to ON (1)

The processor turns light output O:4/7 ON during the next I/O scan

Scan Patterns



Horizontal Scanning Order

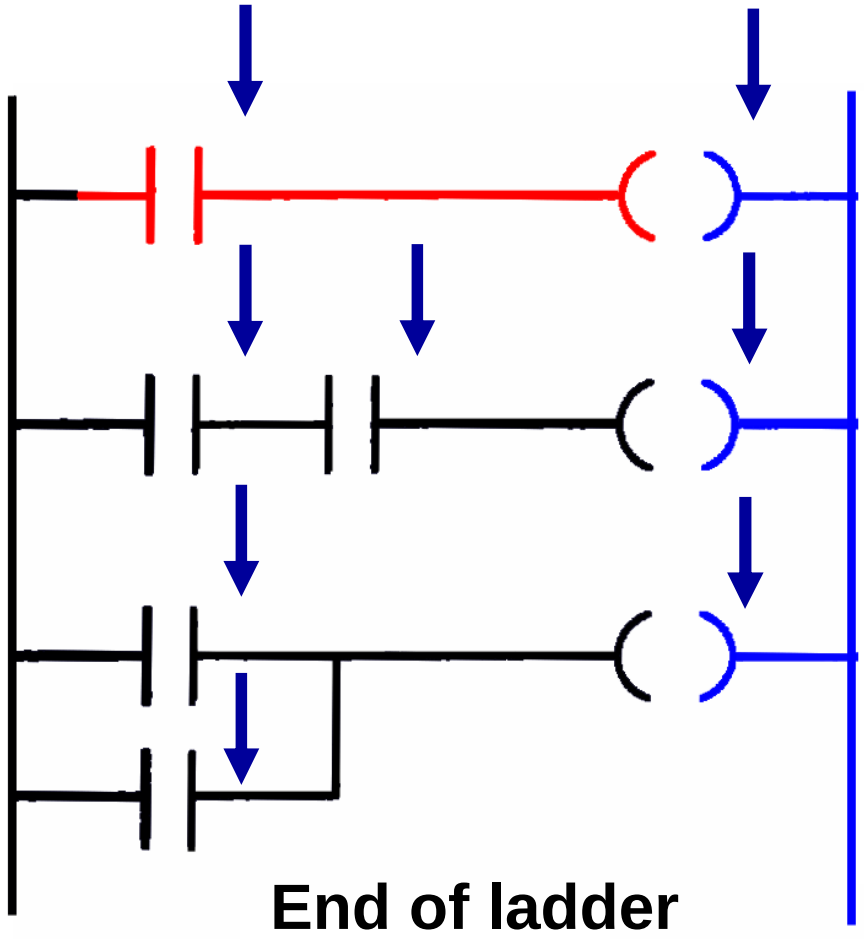
The processor examines input and output instructions from the first command, top left in the program, horizontally, rung by rung.

In addition to the program itself, the scan time is also dependent on the clock frequency of the processor!

Scan Patterns

Vertical Scanning Order

The processor examines input and output instructions from the first command, vertically, column by column and page by page. Pages are executed in sequence.

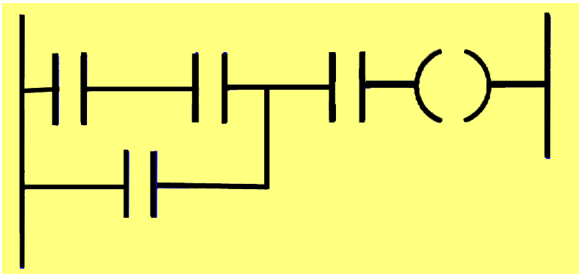


Misunderstanding the way the PLC scans can cause programming bugs!

PLC Programming Languages

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

The three most common language structures are: ladder diagram language, Boolean language, and functional chart.

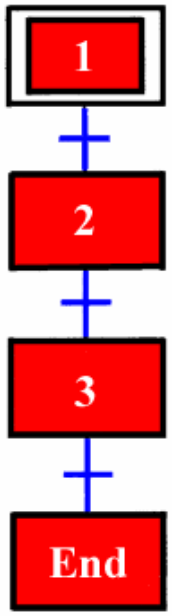


Ladder diagram language

START	PB1
AND	CR1
OR	LS1
AND NOT	CR2
OUT	SOL

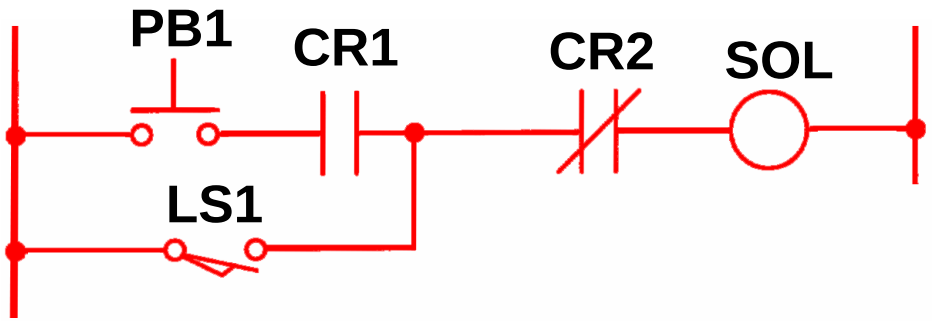
Boolean equation: $Y = [(AB) + C]\,\overline{D}$

Boolean language

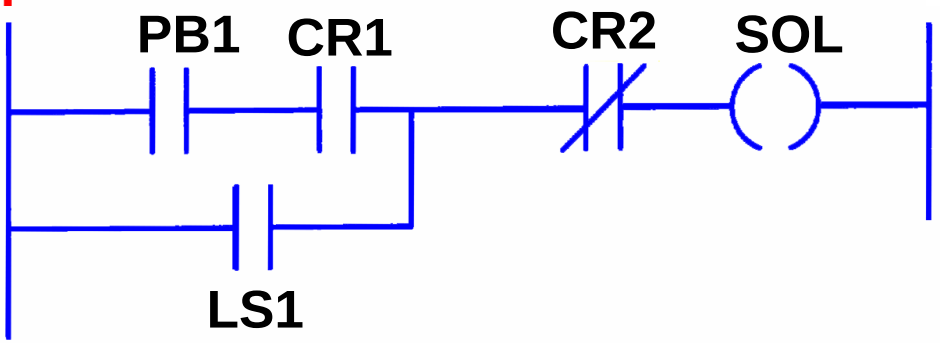


Functional chart

Comparing Programming Language



Relay Schematic



Equivalent ladder diagram language

START	PB1
AND	CR1
OR	LS1
AND NOT	CR2
OUT	SOL

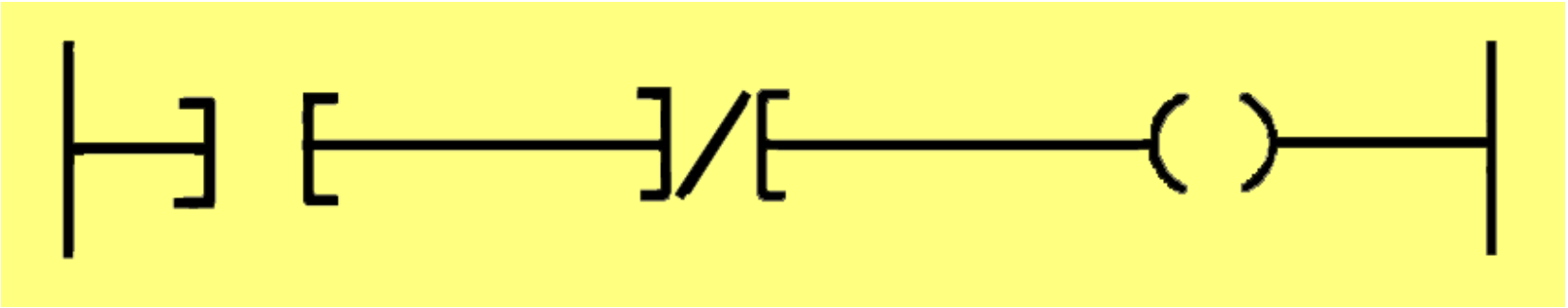
Boolean equation: $Y = [(AB) + C]\,\overline{D}$

Equivalent Boolean language

Relay-Type Instructions

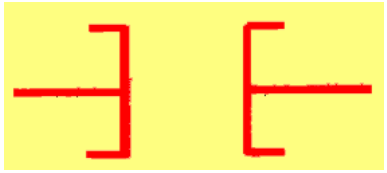
The ladder diagram language is basically a *symbolic* set of instructions used to create the controller program.

These ladder instructions symbols are arranged to obtain the desired control logic.



Examine If Closed (XIC) Instruction

Symbol



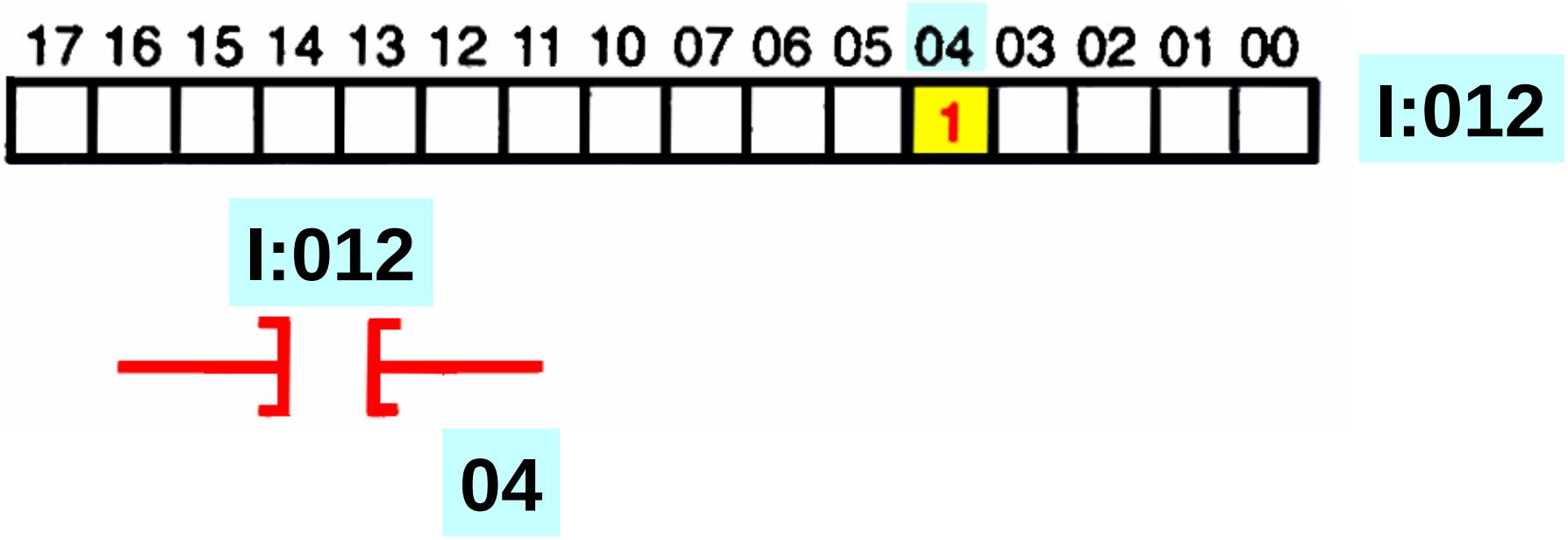
Analogous to the normally open relay contact. For this instruction we ask the processor to EXAMINE IF (the contact is) CLOSED (XIC)

Typically represents any input. Can be a switch or pushbutton, a contact from a connected output, or a contact from an internal output.

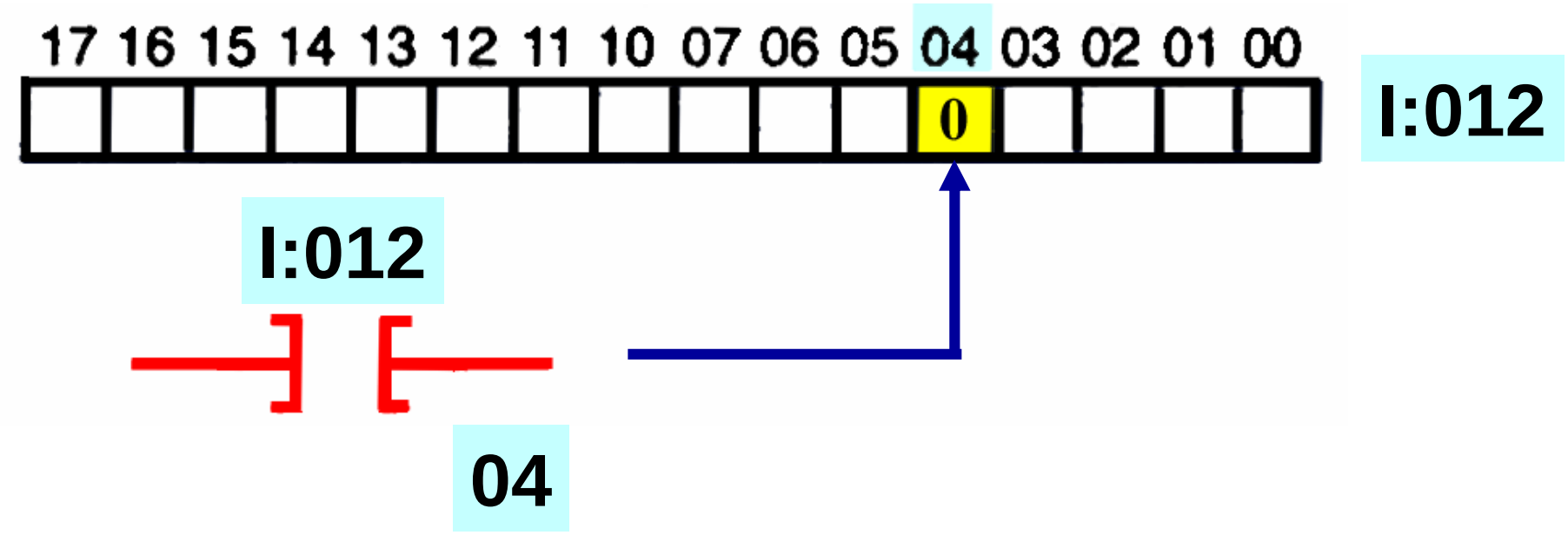
Has a bit-level address which is examined for an ON condition.

The status bit will be either 1 (ON) or 0 (OFF).

Examine If Closed (XIC) Instruction

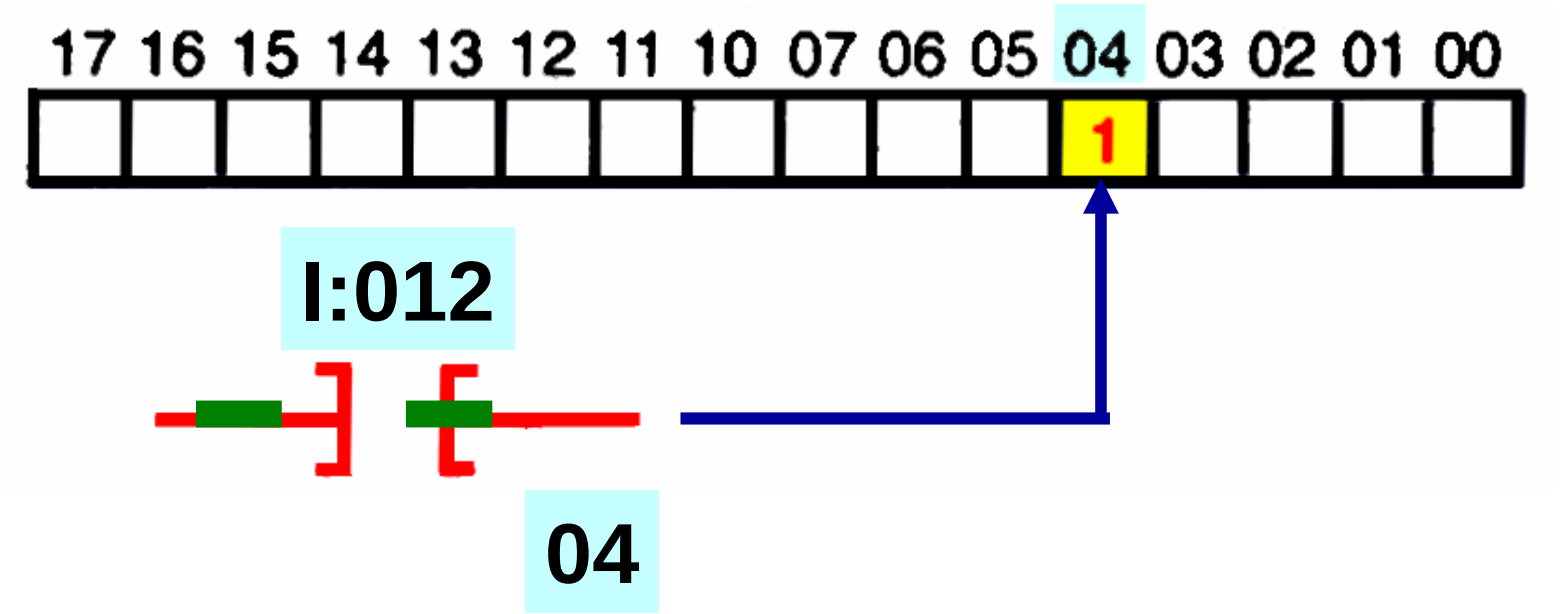


Examine If Closed (XIC) Instruction



If the status bit is 0 (OFF), then the instruction is FALSE.

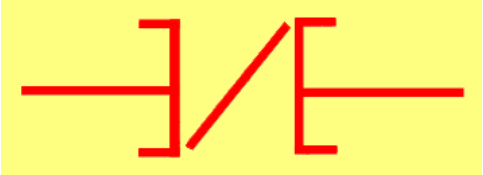
Examine If Closed (XIC) Instruction



If the status bit is 1 (ON), then the instruction is TRUE.

Examine If Open (XIO) Instruction

Symbol



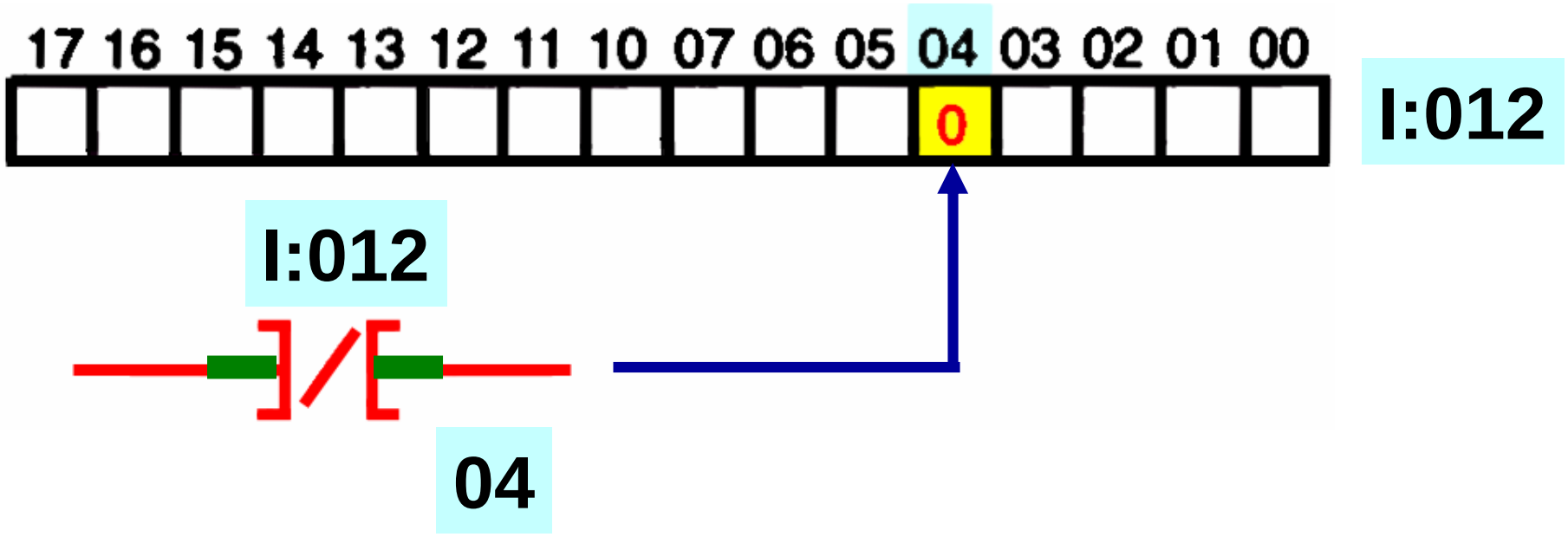
Analogous to the normally closed relay contact. For this instruction we ask the processor to EXAMINE IF (the contact is) OPEN (XIO).

Typically represents any input. Can be a switch or pushbutton, a contact from a connected output, or a contact from an internal output.

Has a bit-level address which is examined for an OFF condition.

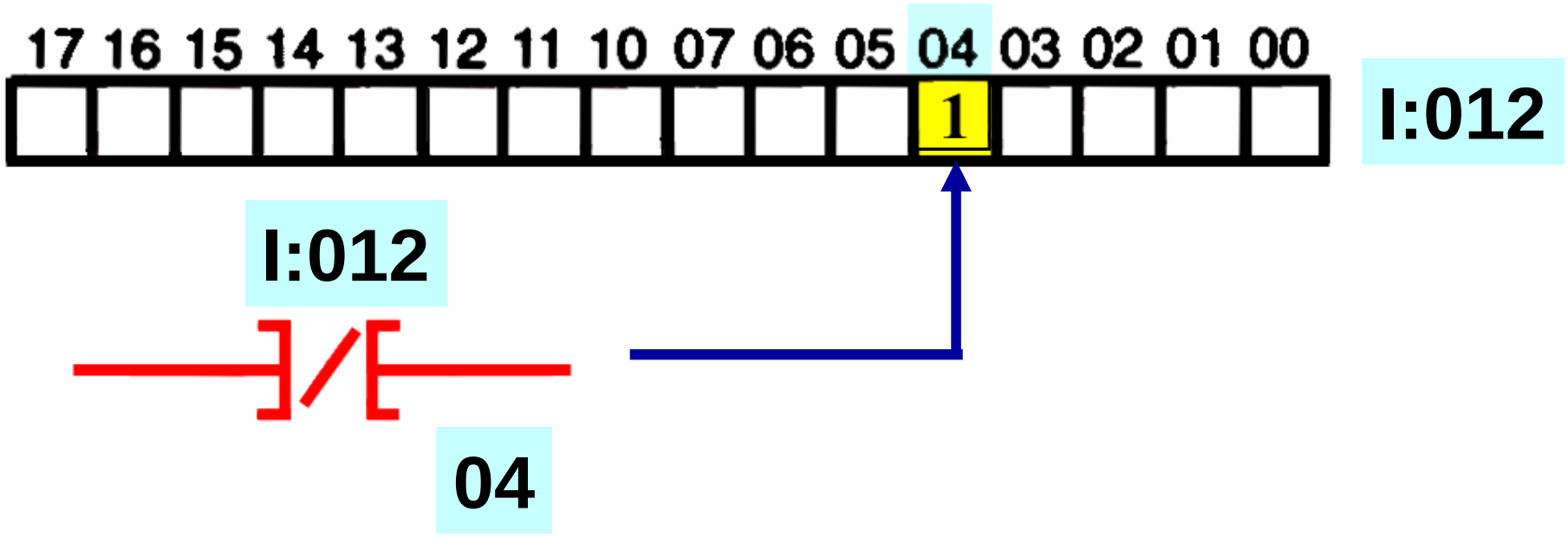
The status bit will be either 1 (ON) or 0 (OFF).

Examine If Open (XIO) Instruction



If the status bit is 0 (OFF), then the instruction is TRUE.

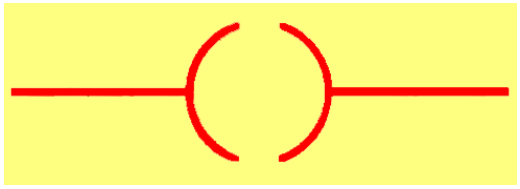
Examine If Open (XIO) Instruction



If the status bit is 1 (ON), then the instruction is FALSE.

Output Energize (OTE) Instruction

Symbol

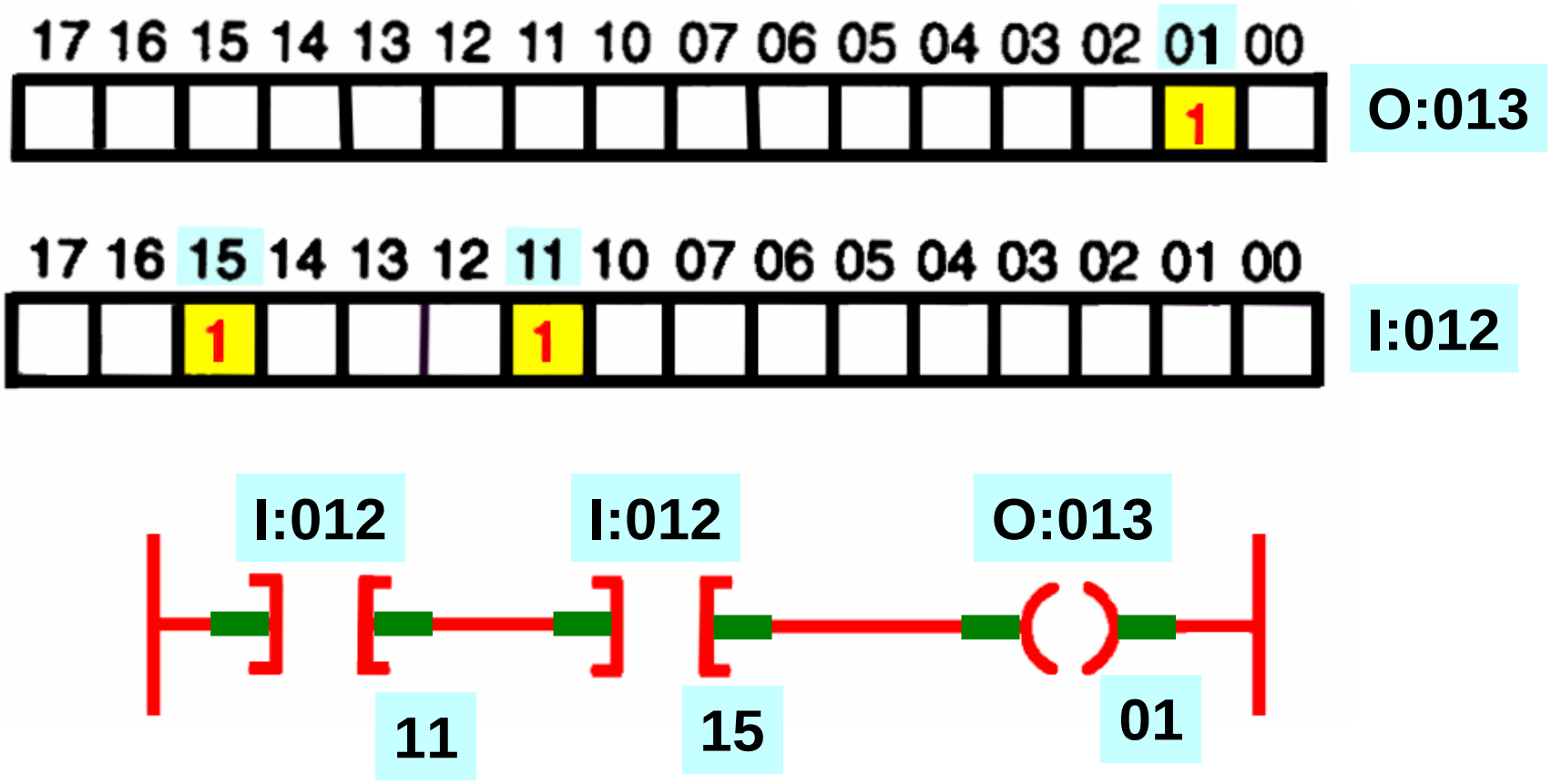


Analogous to the relay coil. The processor makes this instruction true (analogous to energizing a coil) when there is path of true XIC and XIO instructions in the rung.

Typically represents any output that is controlled by some combination of input logic. Can be a connected device or an internal output (internal relay).

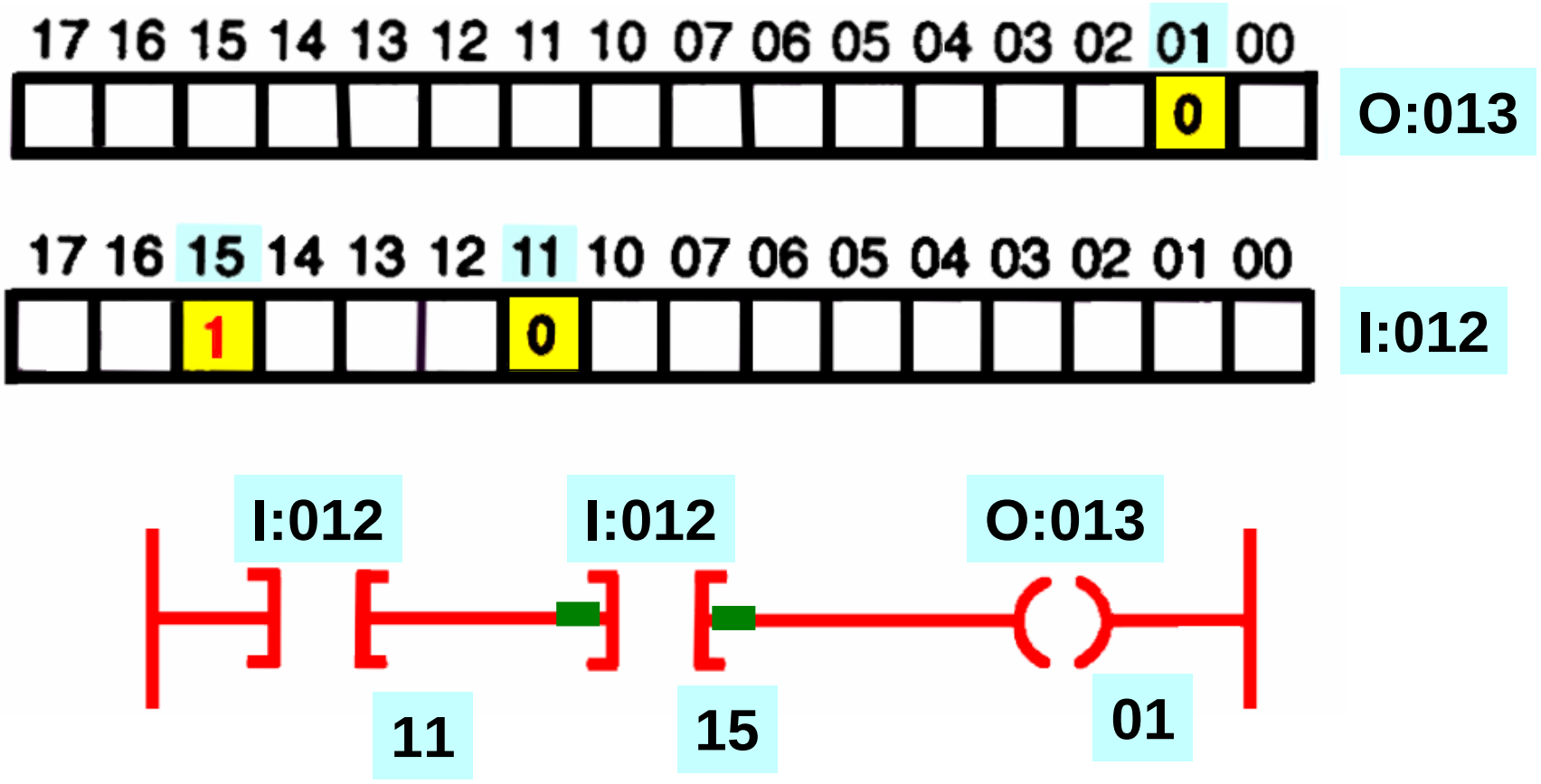
If any left-to-right path of input conditions is TRUE, the output is energized (turned ON).

Output Energize (OTE) Instruction



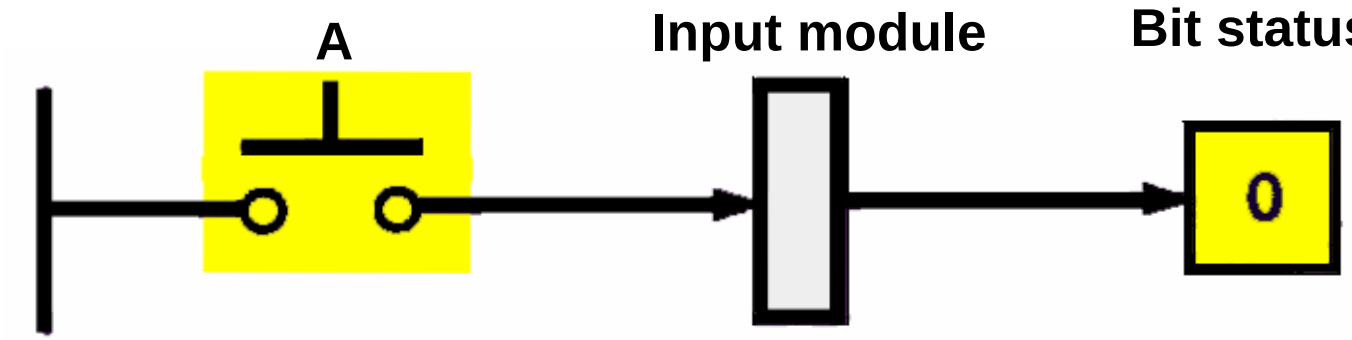
OUTPUT ENERGIZE instruction - TRUE

Output Energize (OTE) Instruction

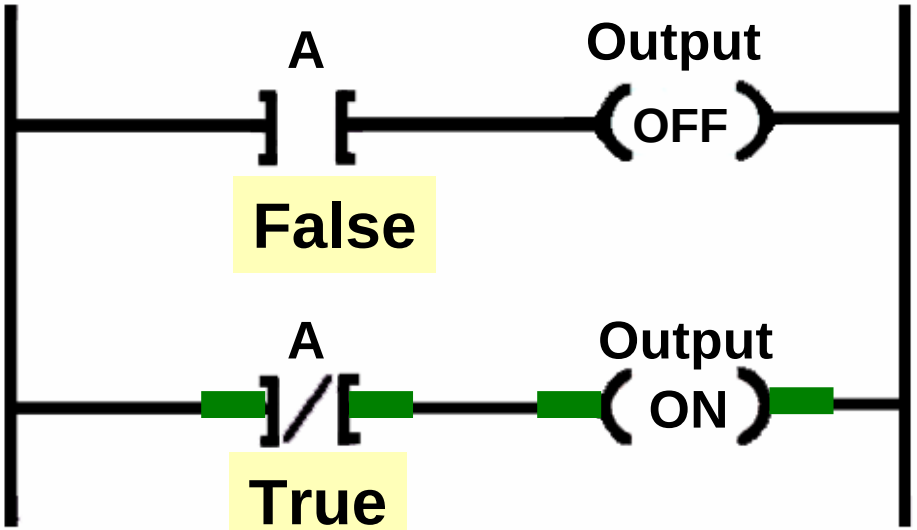


OUTPUT ENERGIZE instruction - FALSE

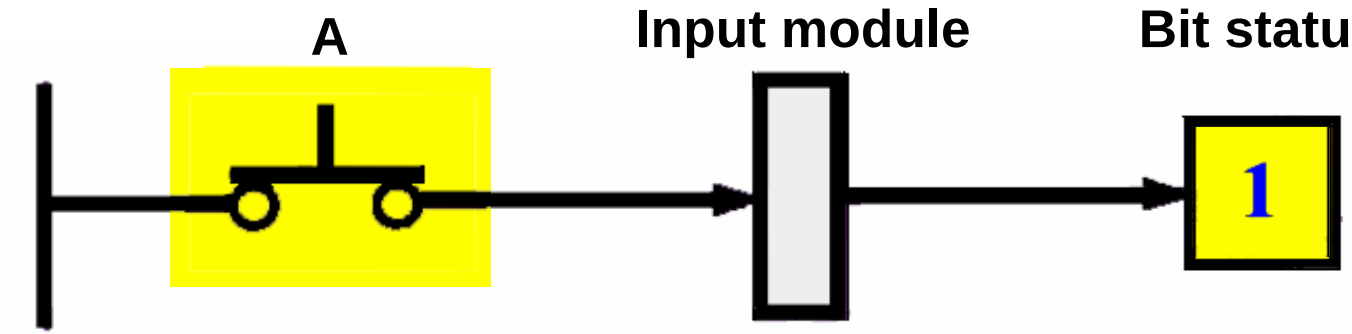
Status Bit Examples



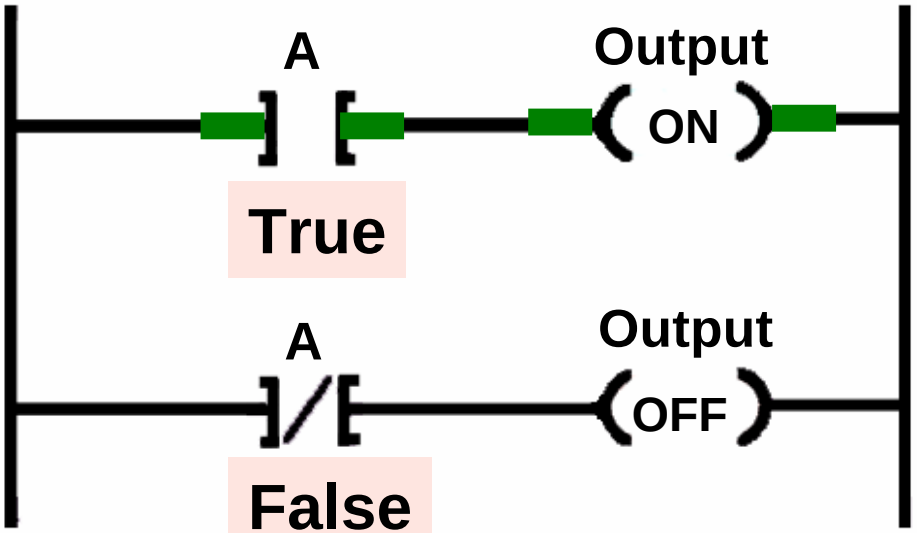
Button not actuated



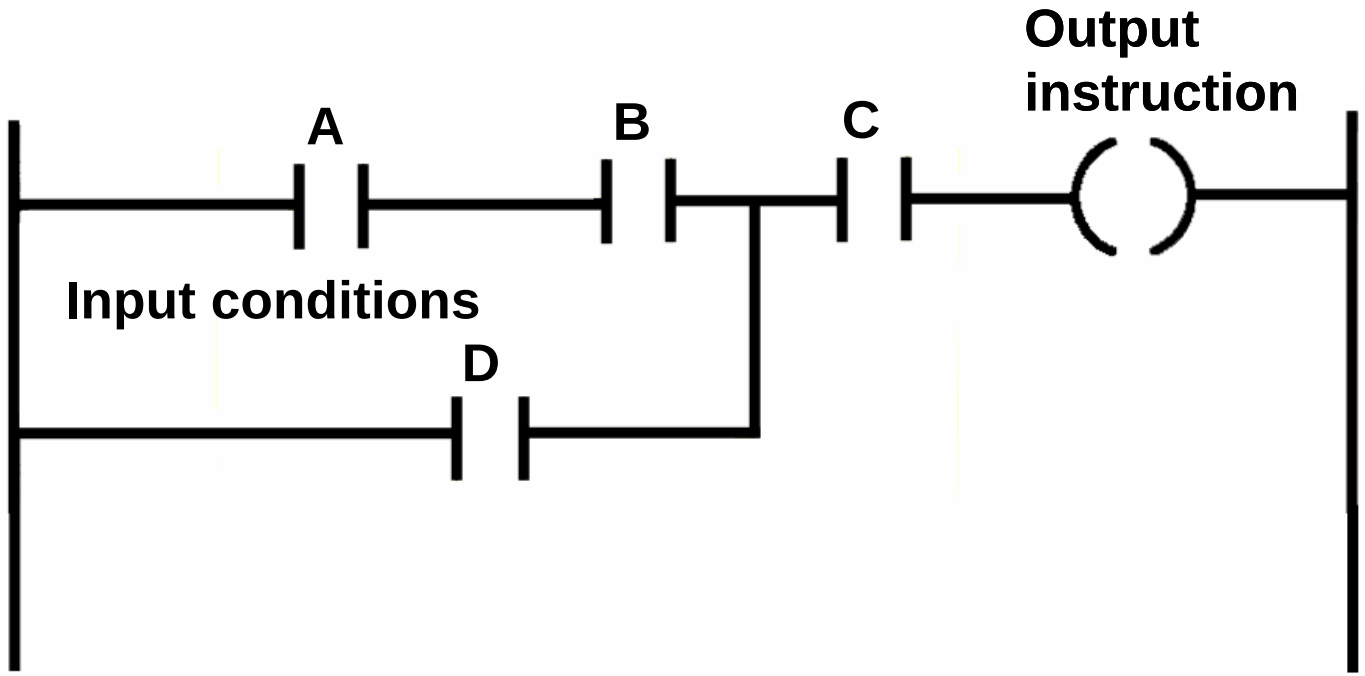
Status Bit Examples



Button actuated

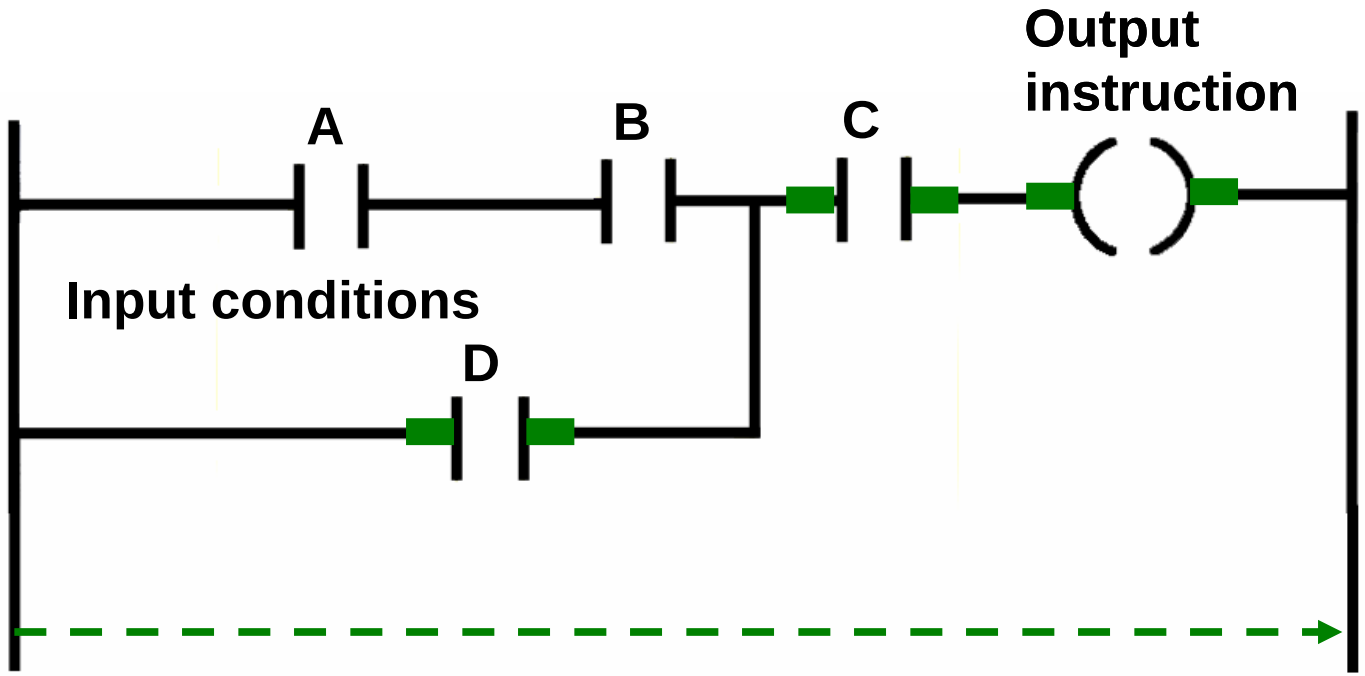


Ladder Rung



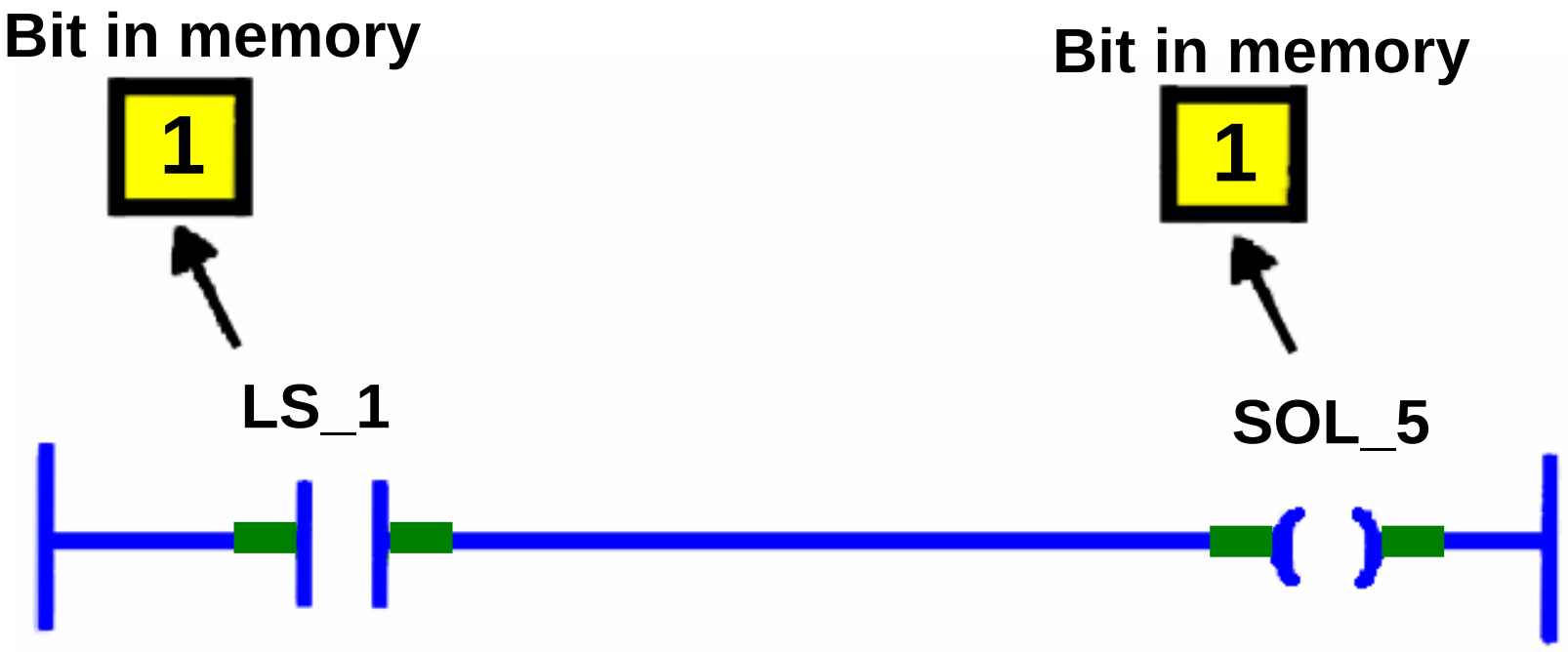
A ladder rung consists of a set of input conditions, represented by contact instructions, and an output instruction at the end of the rung, represented by the coil symbol.

Ladder Rung



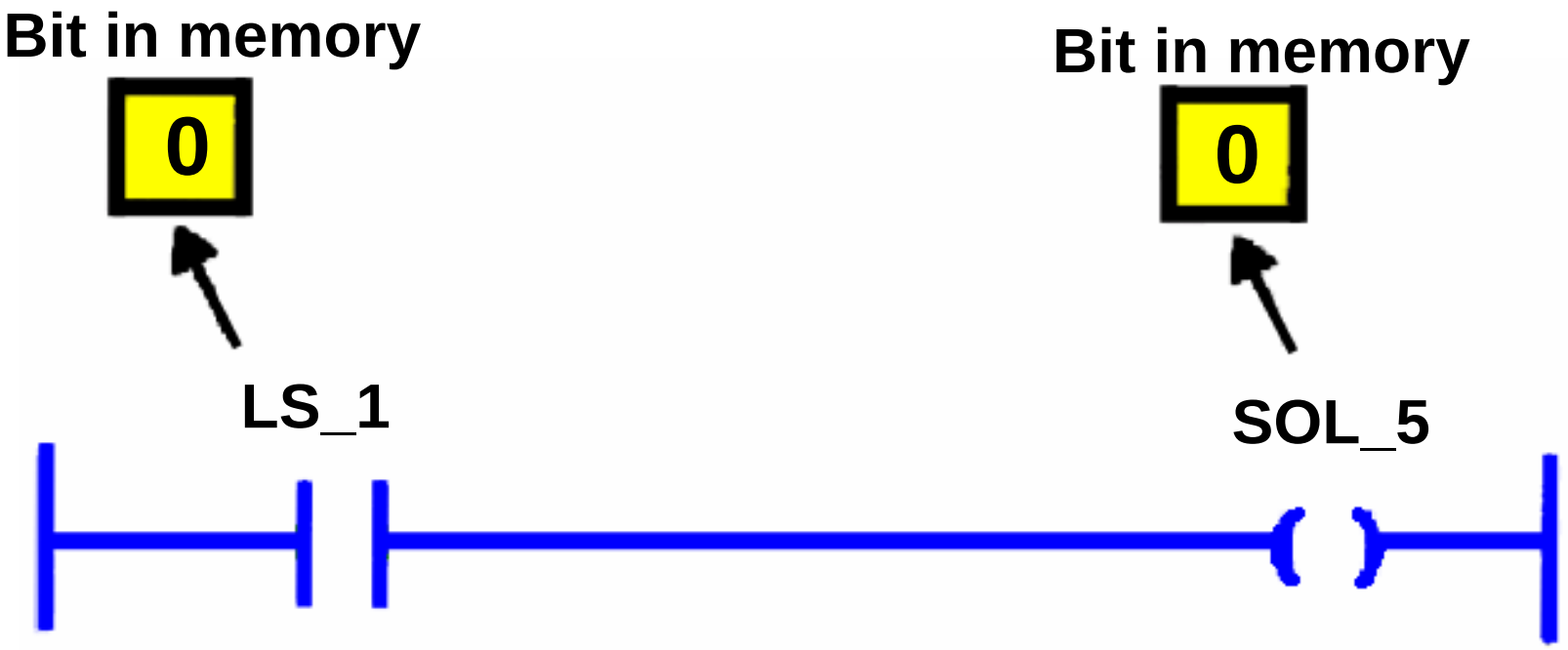
For an output to be activated or energized, at least *one* left-to-right path of contacts most be closed. A complete path is referred to as having *logic continuity*. When logic exists the rung condition is said to be TRUE.

Rung Continuity



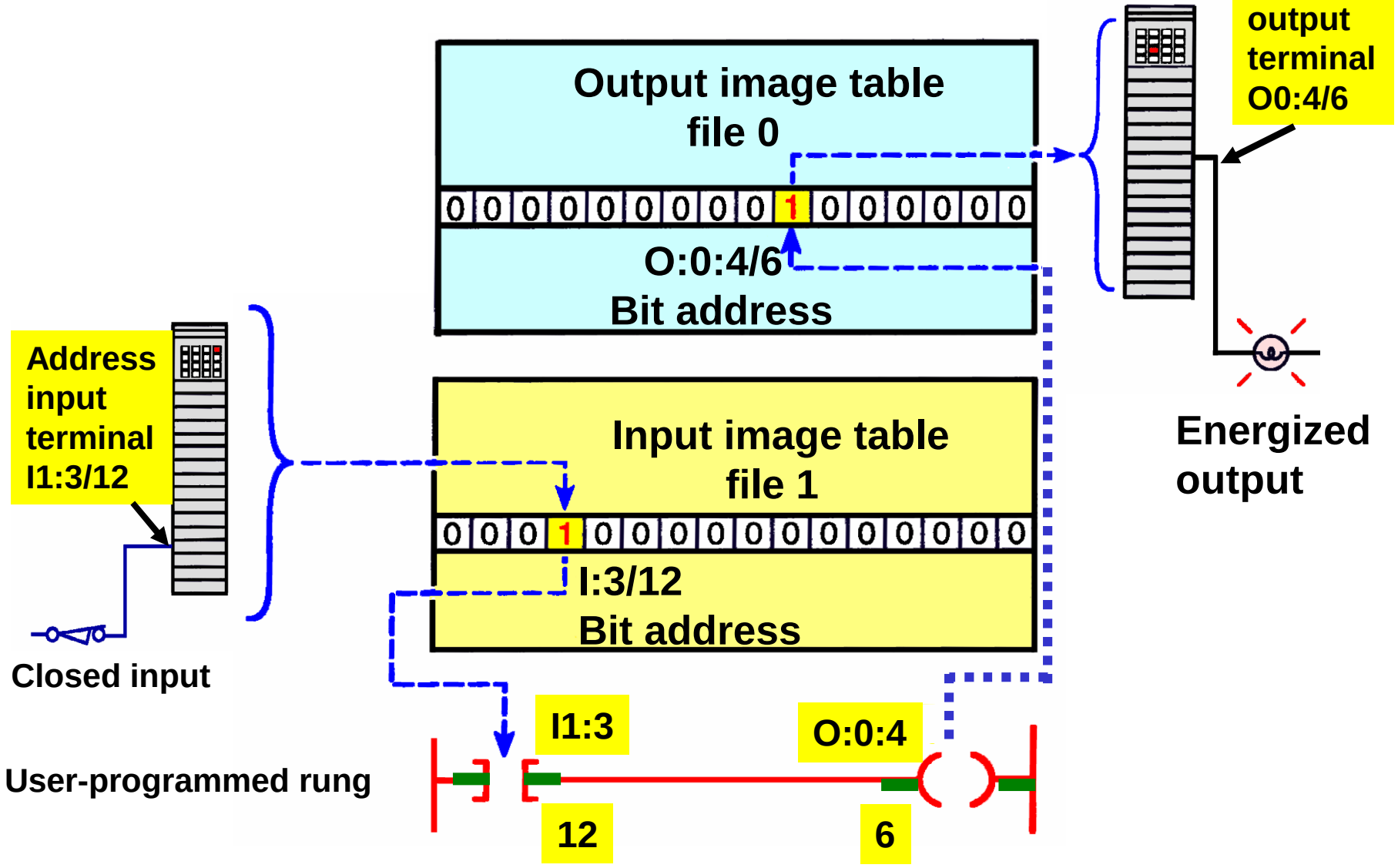
The Examine If Closed instruction is TRUE making the rung TRUE

Rung Continuity

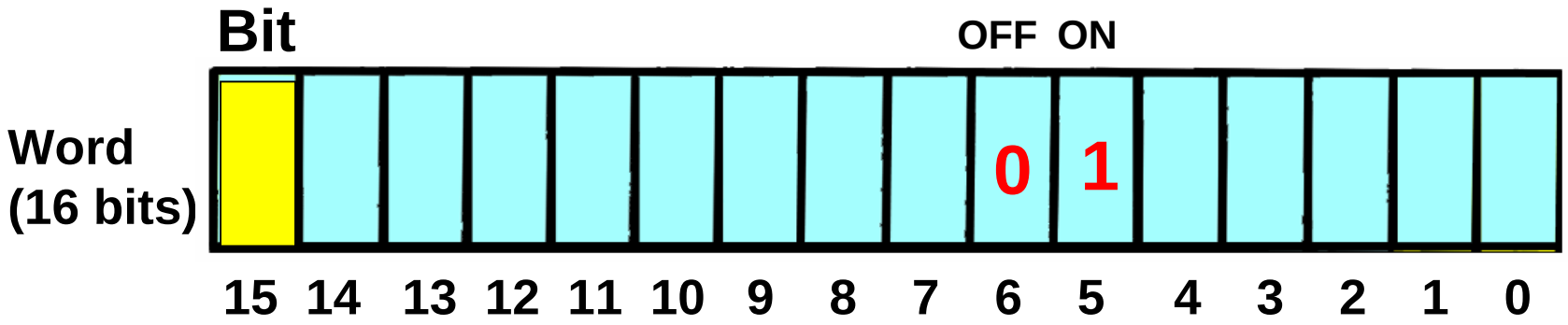


The Examine If Closed instruction is FALSE making the rung False

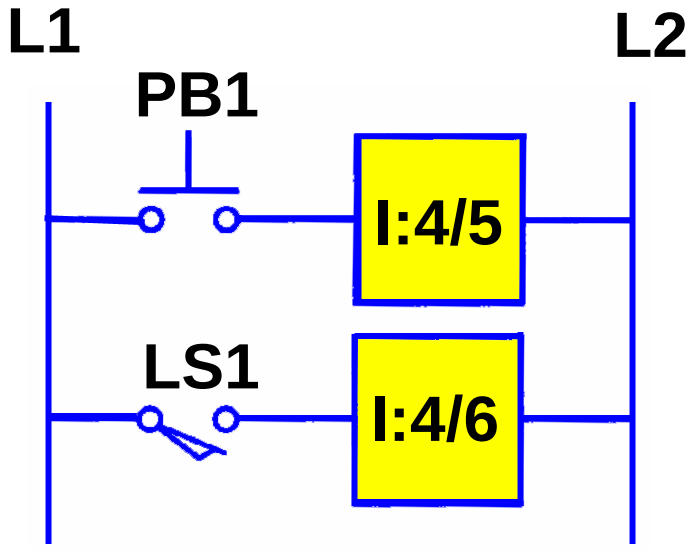
Allen-Bradley SLC-500 Controller Addressing

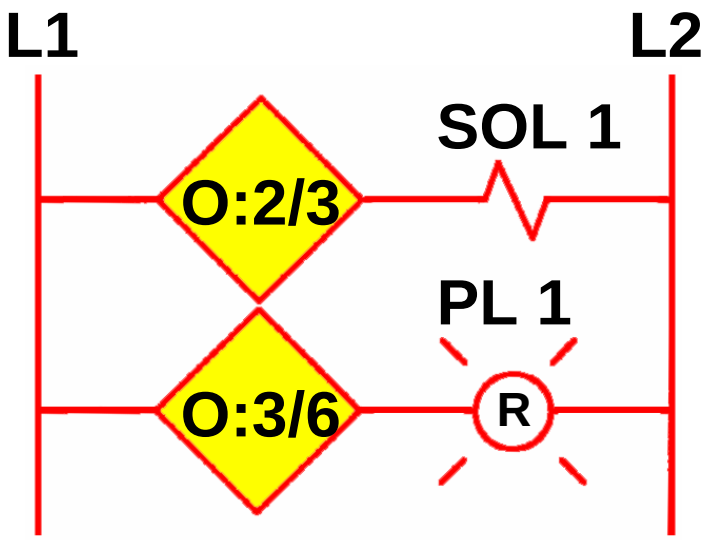


Structure of A 16-Bit Word

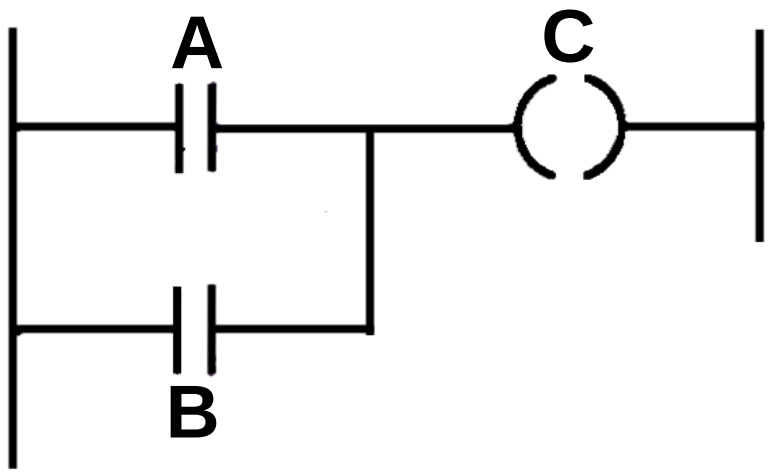


I/O Connection Diagram



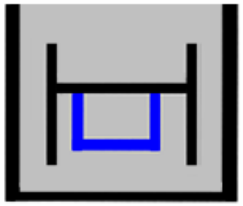


Parallel Input Branch Instructions



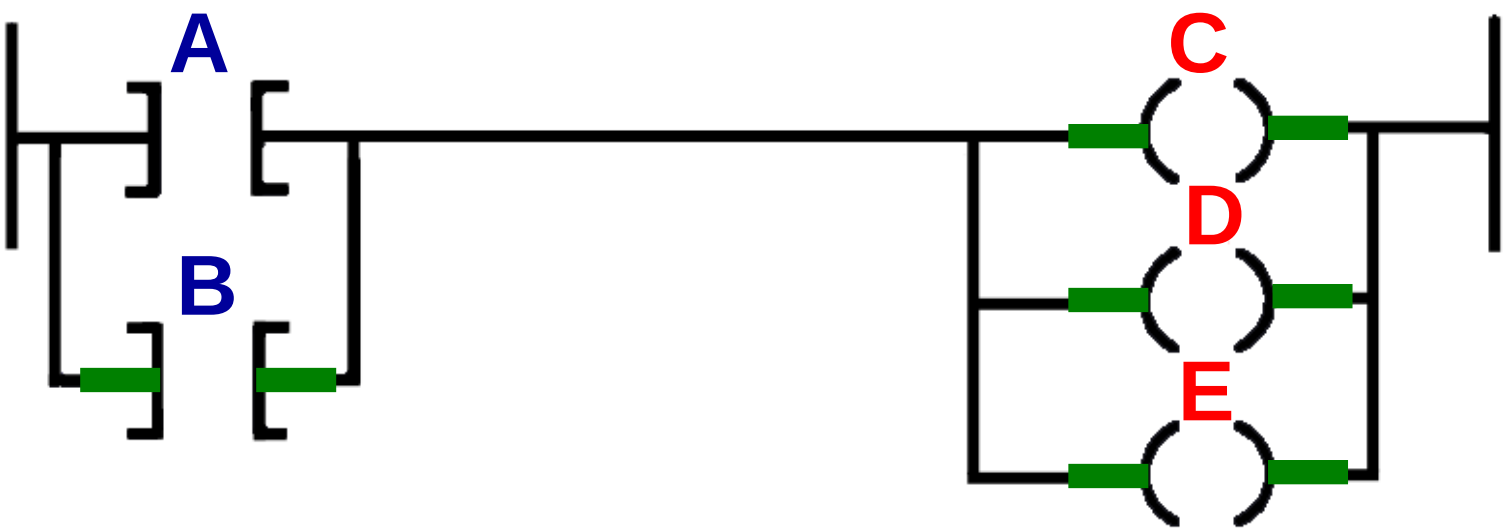
Branch instructions are used to create parallel paths of input condition instructions. If at least one of these parallel branches forms a true logic path, the logic is enabled.

Add A Branch



Click this icon on the instruction toolbar to place a branch in your ladder logic. If your cursor is on an instruction, the branch is placed immediately to the right of the instruction. If your cursor is on the rung number, the branch is placed on the rung.

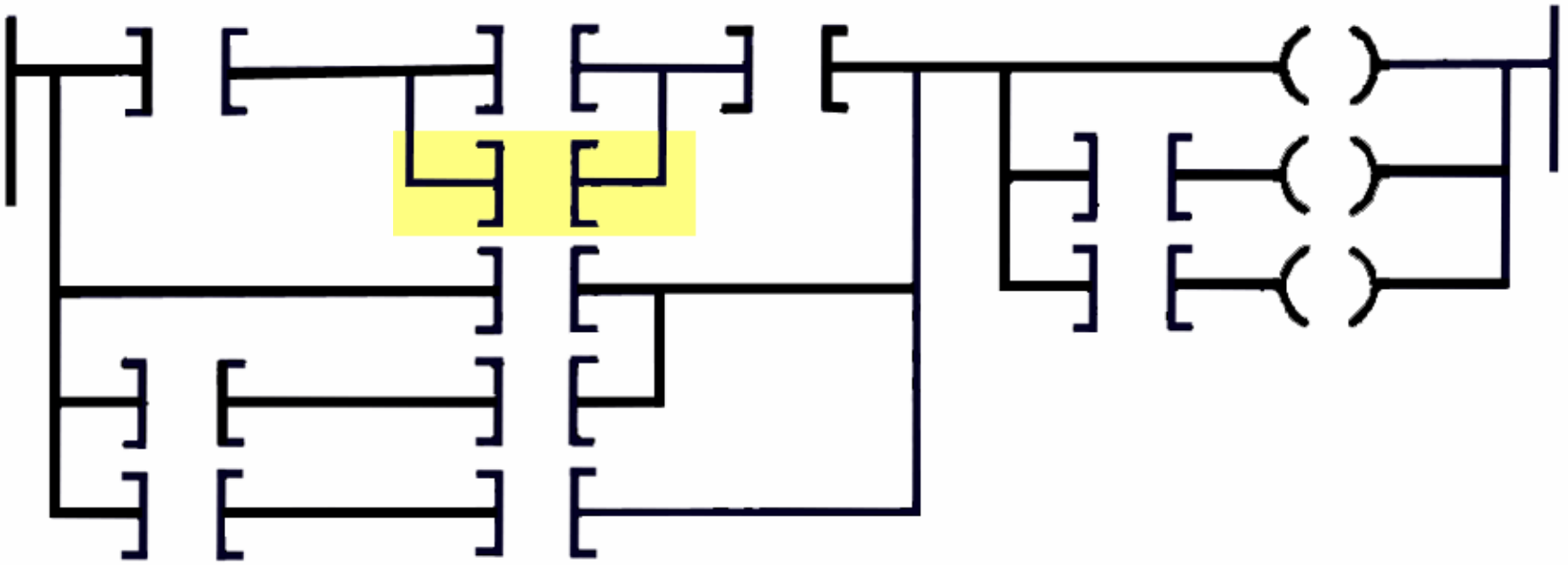
Parallel Output Branching



On most PLC models, branches can be established at both the input and output portion of the rung.

With output branching, you can program parallel outputs on a rung to allow a true logic path to control multiple outputs.

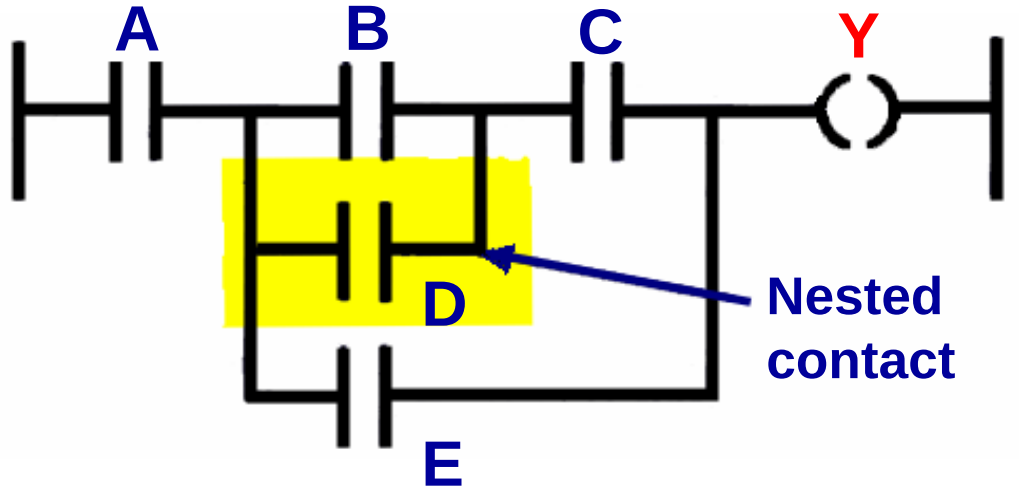
Nested Input and Output Branches



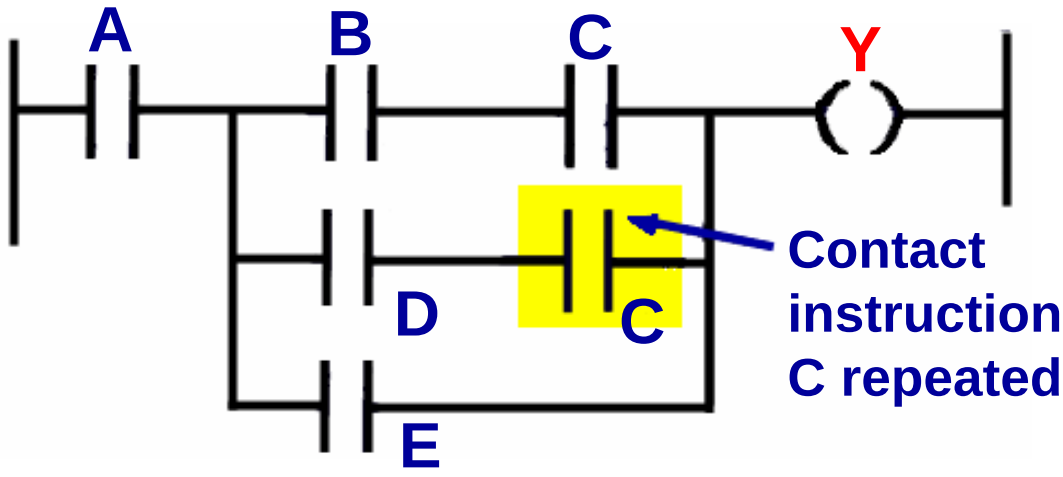
Input and output branches can be *nested* to avoid redundant instructions and to speed up the processor scan time.

A nested branch starts or ends within another branch.

Nested Contact Program

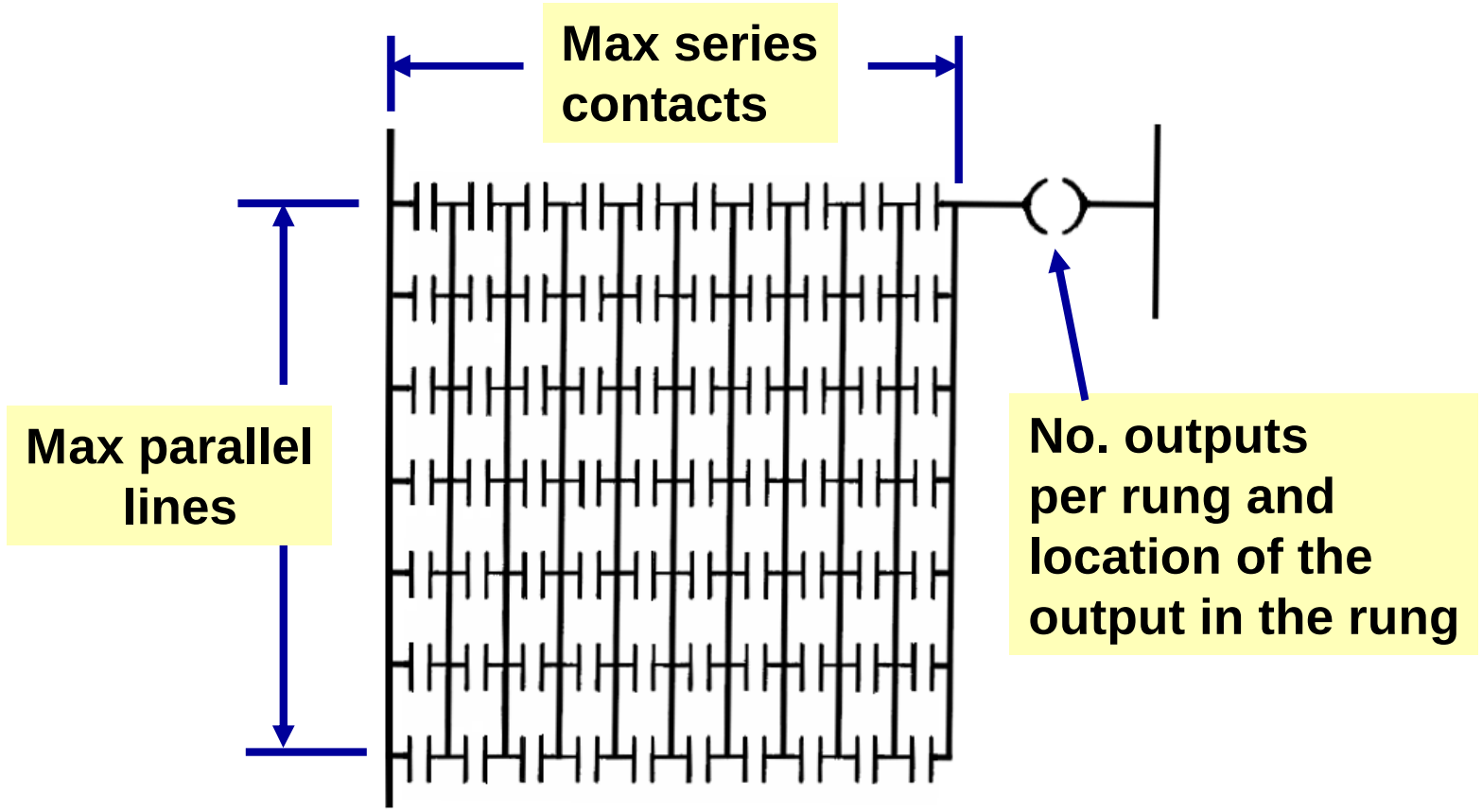


On some PLC models, the programming of a nested branch circuit cannot be done directly.



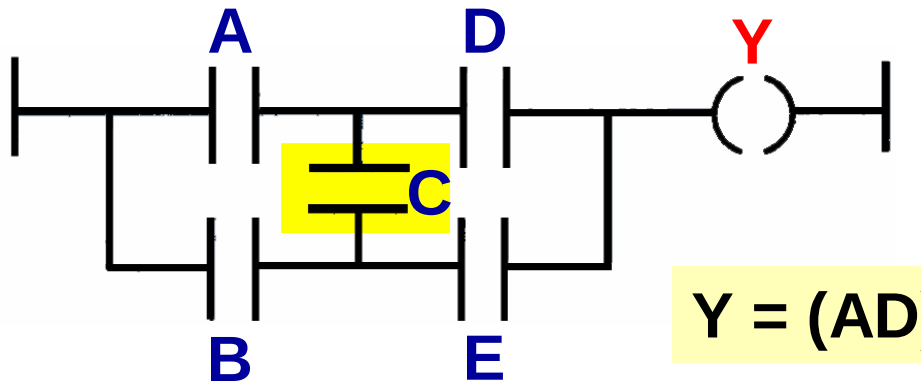
Reprogrammed to obtain the required logic.

PLC Matrix Limitation Diagram



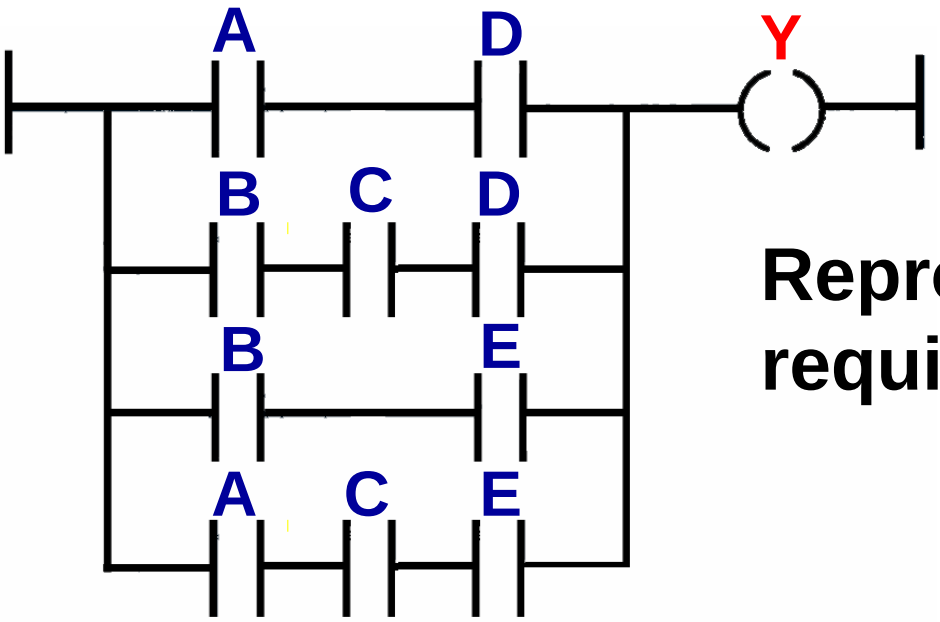
There may be limitations to the number of series contacts instructions, number of parallel lines, and the number of outputs and their location on the rung.

Programming of Vertical Contacts



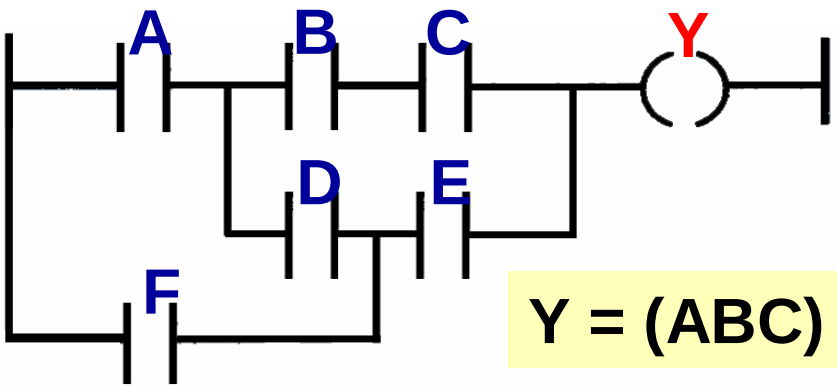
Original program

Y = (AD) + (BCD) + (BE) + (ACE)



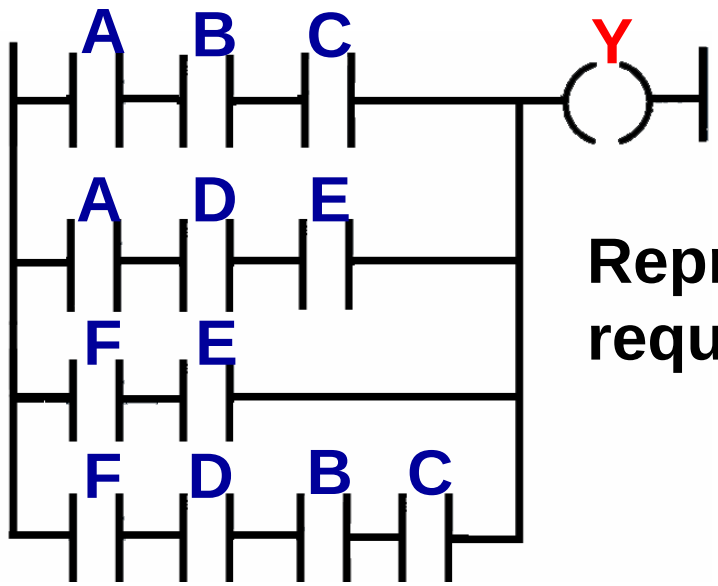
Reprogrammed to obtain the required logic

Programming for Different Scan Patterns



Original program

Y = (ABC) + (ADE) + (FE) + (FDBC)



Reprogrammed to obtain the required logic

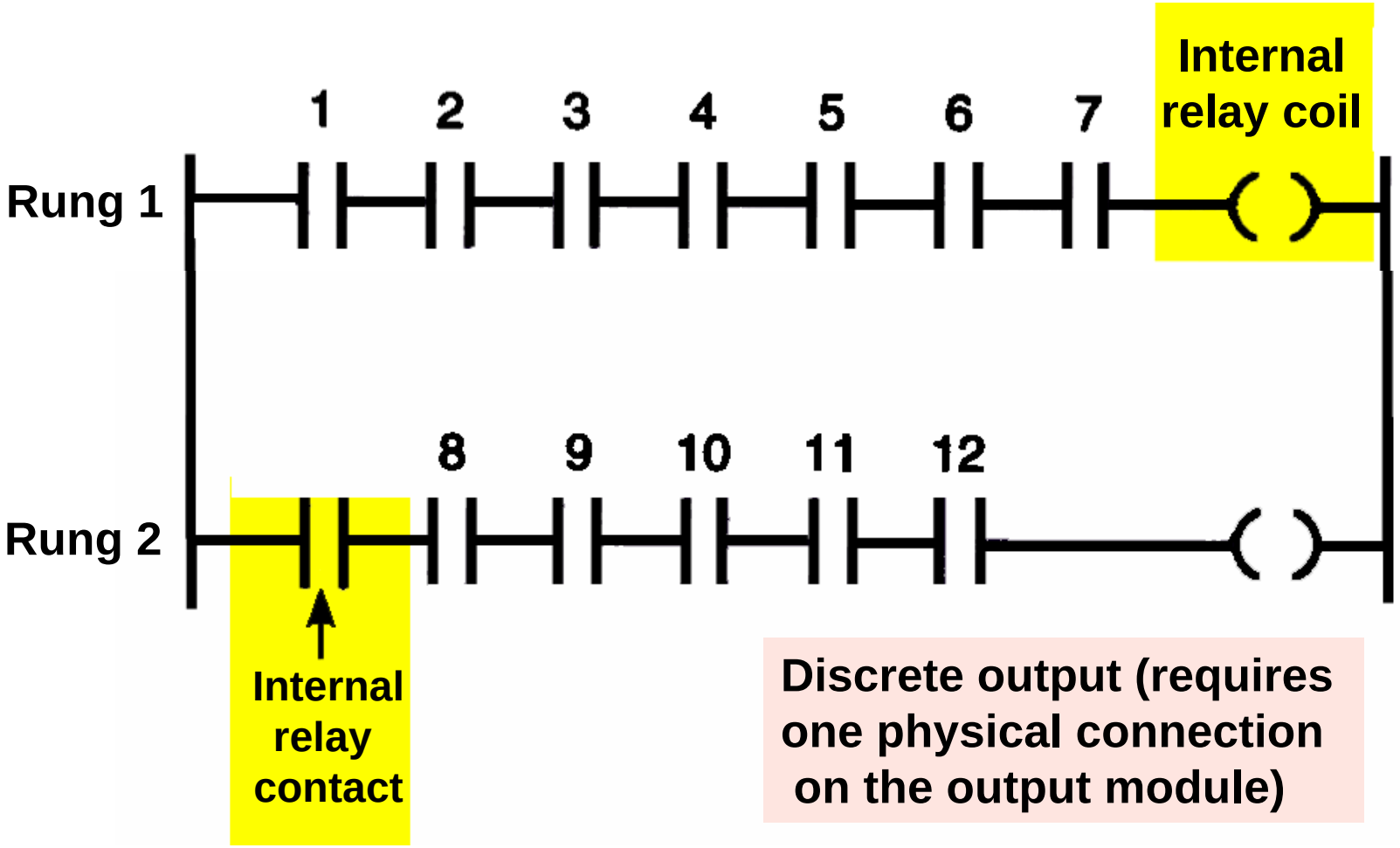
Internal Control Relay

The internal output operates just as any other output that is controlled by programmed logic; however, the output is used strictly for internal purposes.

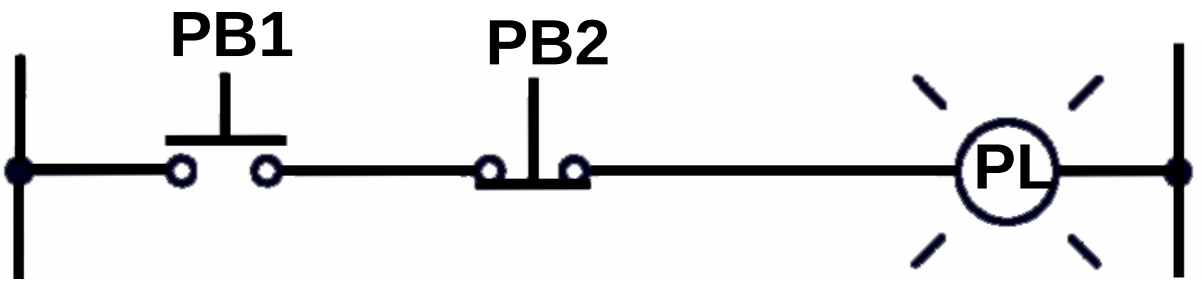
The internal output does not directly control an output device.

The advantage of using internal outputs is that there are many situations where an output instruction is required in a program, but no physical connection to a field device is needed. Their use in this type of instance can minimize output card requirements.

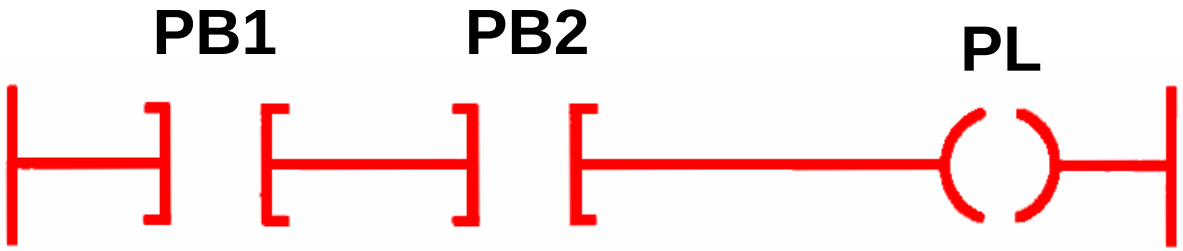
Extending the Number of Series Contacts Using an Internal Control Relay



Programming The XIC Instruction



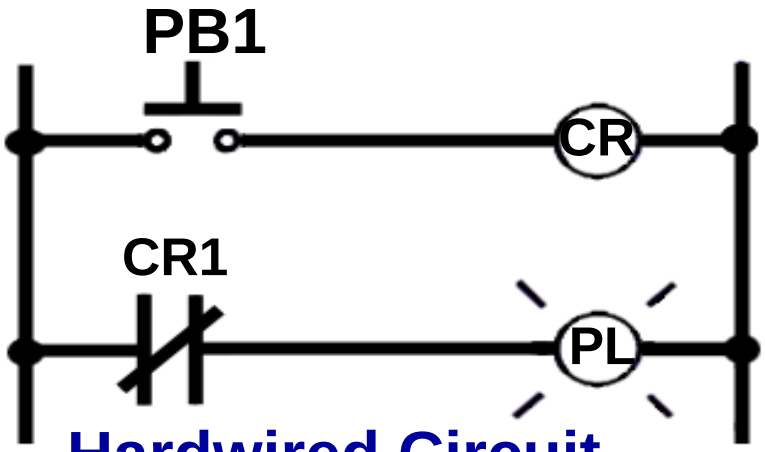
Hardwired Circuit



User program providing the same results

Note that both pushbuttons are represented by the XIC symbol. This is because the normal state of an input (NO or NC) does not matter! What does matter is that if contacts need to close to energize the output, then the XIC instruction is used. Since both PB1 and PB2 must close to energize the PL, the XIC instruction is used for both.

Programming The XIO Instruction



Hardwired Circuit

User program providing the same results

When the pushbutton is *open* in the hardwired circuit, relay coil CR is de-energized and contacts CR1 close to switch the PL on. When the pushbutton is *closed*, relay coil CR is energized and contacts CR1 open to switch the PL off. The pushbutton is represented in the user program by an XIO instruction. This is because the rung must be true when the external pushbutton is *open*, and false when the pushbutton is *closed*.

Operation of The XIC and XIO Instructions

Summary of status conditions

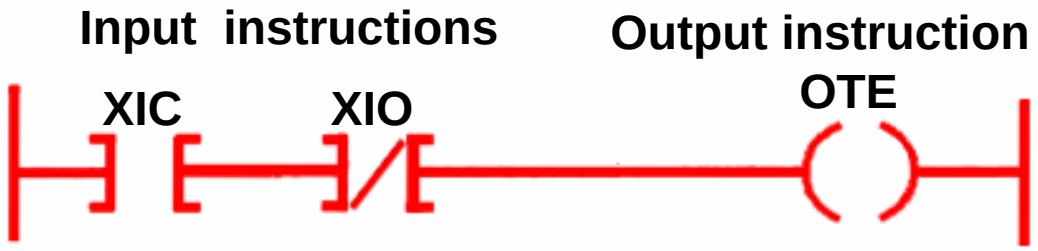
If the data table bit is:	The status of the instruction is:		
	XIC Examine If Closed	XIO Examine If Open	OTE Output Energize
Logic 0	False	True	False
Logic 1	True	False	True

Operation of The XIC and XIO Instructions

State of the output as determined by the changing state of the inputs in the rung

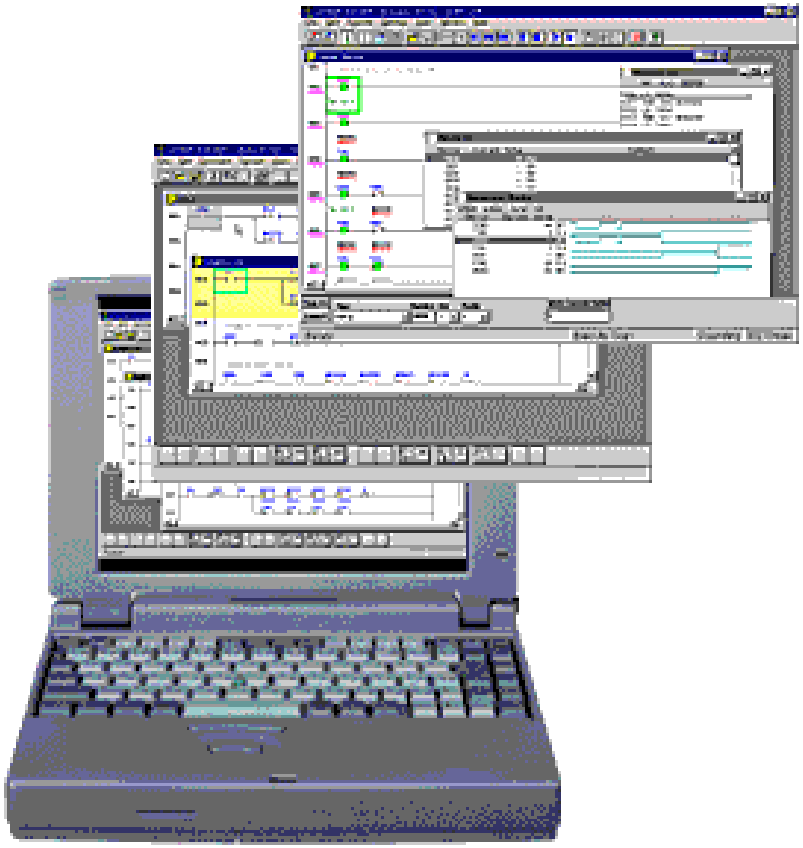
Time	Inputs		Output
	XIC	XIO	OTE
t_1 (initial)	False	True	False
t_2	True	True	Goes true
t_3	True	False	Goes false
t_4	False	False	Remains false

Bit status		
XIC	XIO	OTE
0	0	0
1	0	1
1	1	0
0	1	0



Entering the Ladder Diagram

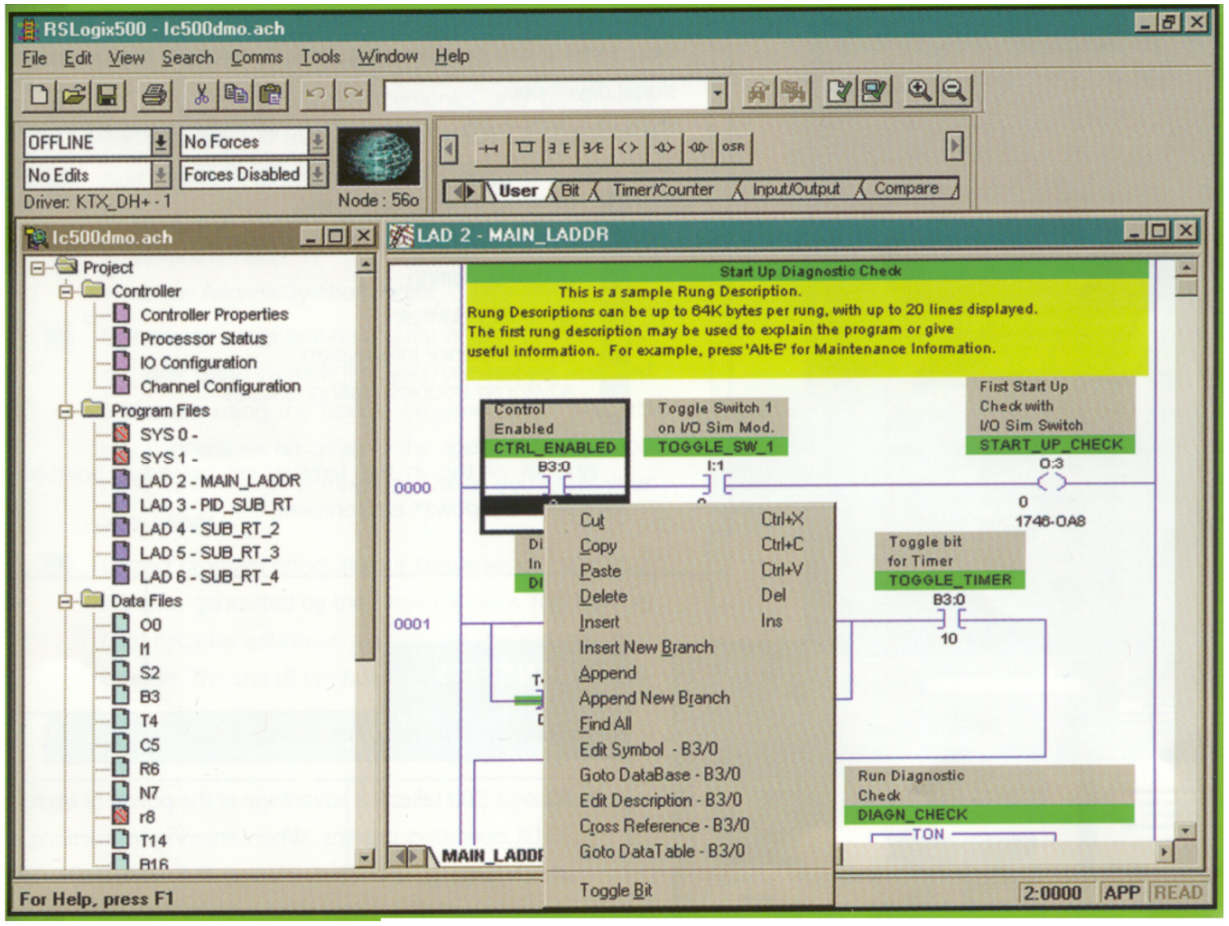
A personal computer is most often used to enter the ladder diagram.



The computer is adapted to the particular PLC model using the relevant programmable controller software.

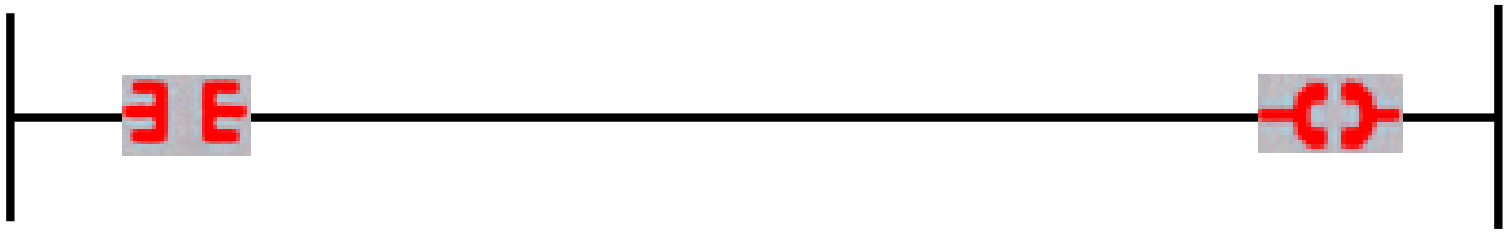
RSLogix Main Screen

Different screens, toolbars and windows dialog boxes are used to navigate through the Windows environment



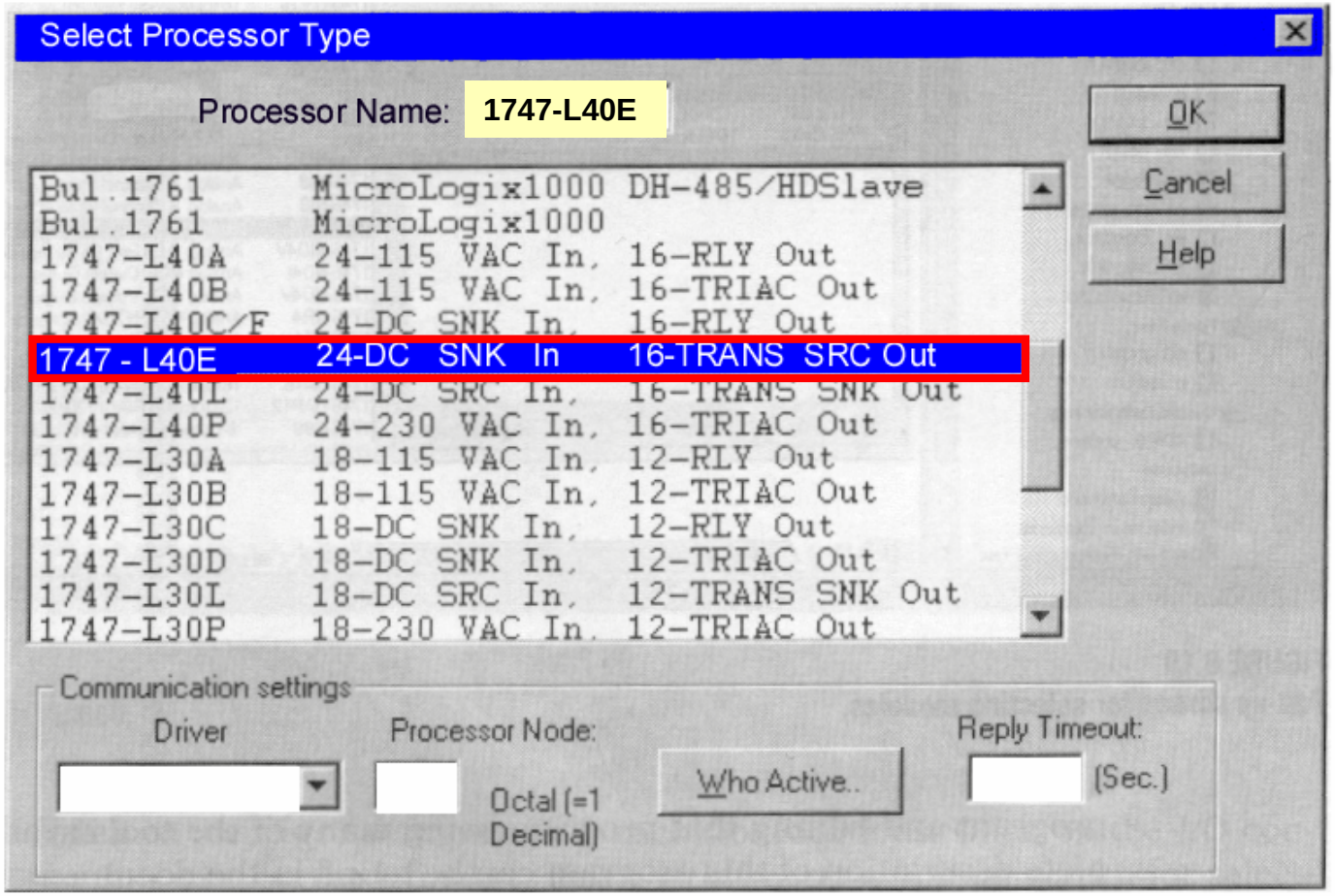
Bit Instructions Tool Bar

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.



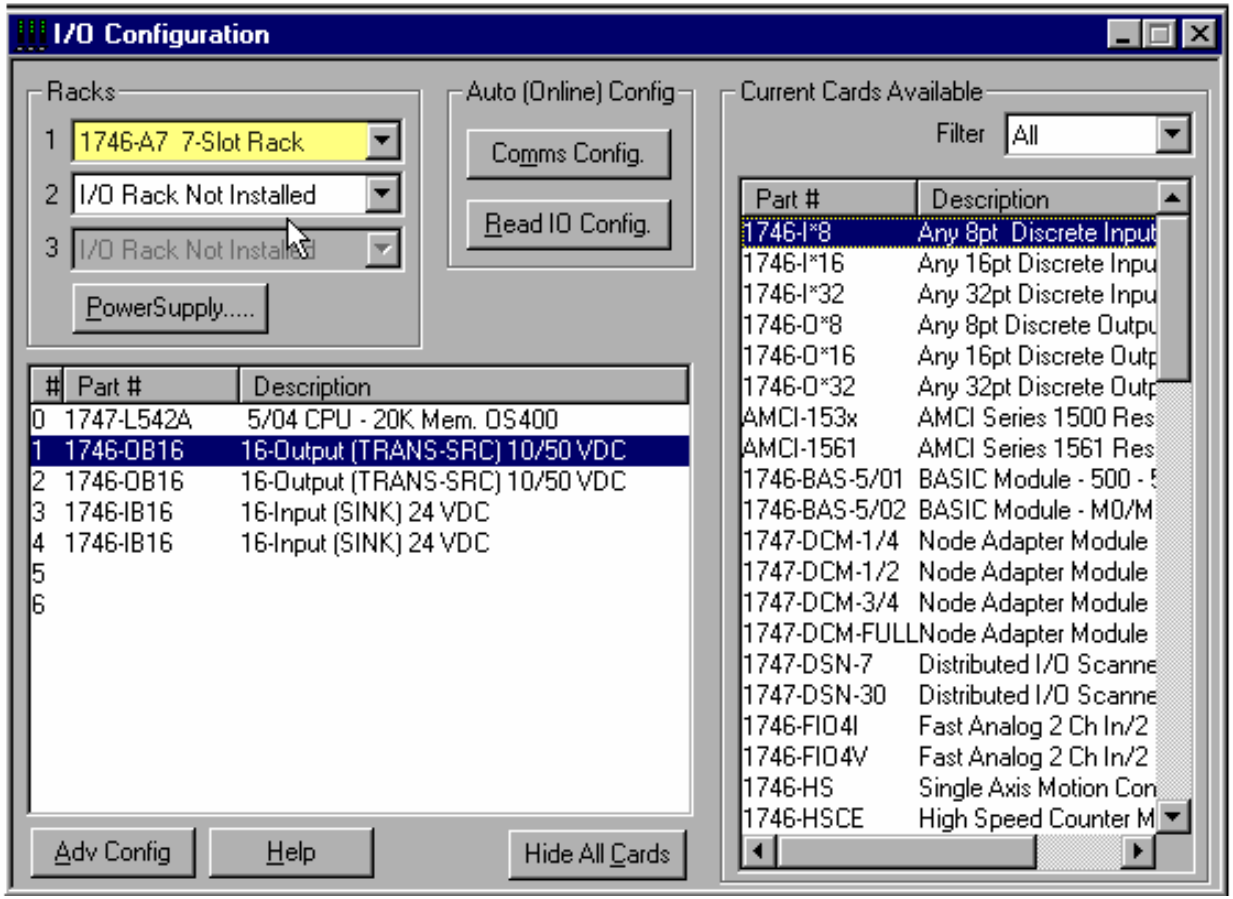
Select Processor Type Screen

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



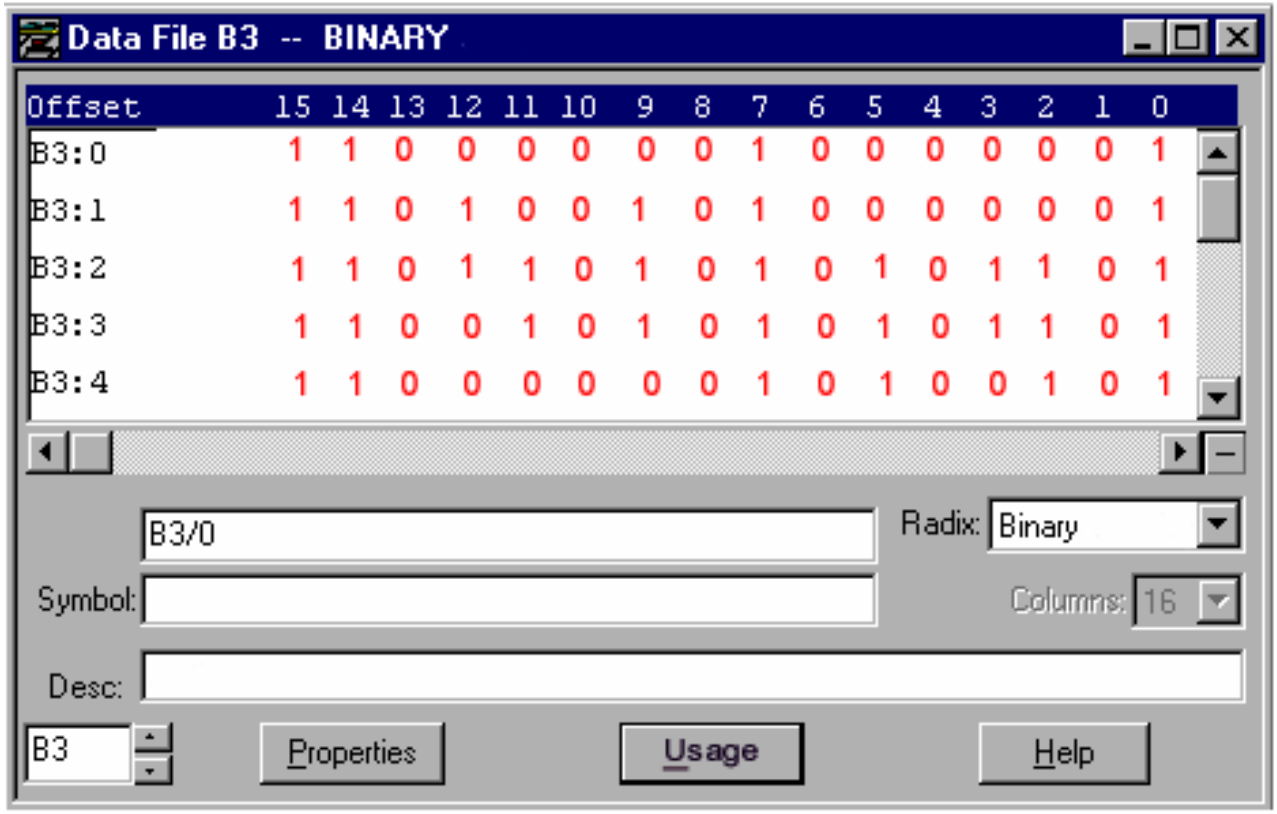
You simply scroll down the list until you find the processor you are using and select it.

I/O Configuration Screen



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

Data File Screen

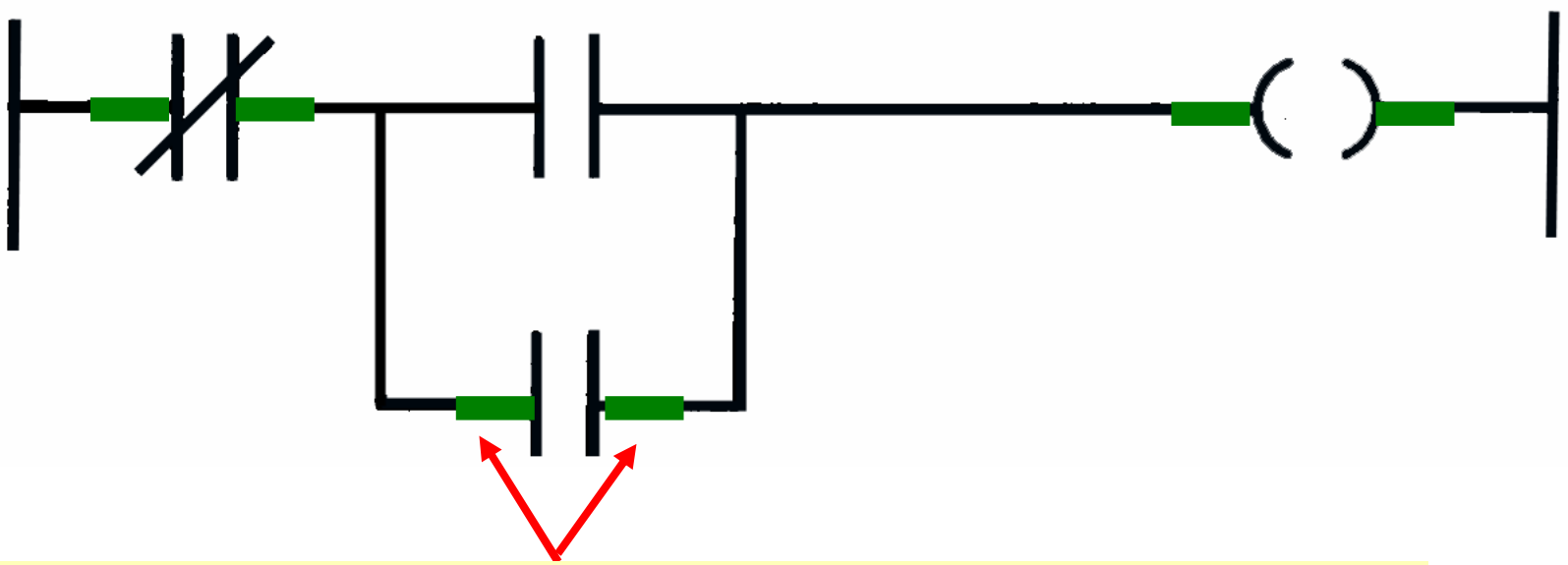


Data file screens contain data that is used in conjunction with ladder program instructions. These include:

Input
Output
Timer
Counter
Integer
Bit

Monitoring a Ladder Logic Program

Operation of the logic is apparent from the highlighting of rungs of the various instructions on screen, which identifies the logic state in real time and has logic continuity.



Highlighted rungs indicate the instruction is true

Modes of Operation

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

Program Mode – may be used to

- Ø **enter a new program**
- Ø **edit or update an existing program**
- Ø **upload files**
- Ø **download files**
- Ø **document programs**
- Ø **change software configurations**

When the PLC is switched into the program mode, all outputs from the PLC are forced off regardless of their rung logic status, and the ladder I/O scan sequence is halted.

Variations of the Run Mode

Run Mode – is used to execute the user program. Input devices are monitored and output devices are energized accordingly.

Test Mode – is used to operate, or monitor, the user program without energizing any outputs.

Remote Mode – allows the PLC to be remotely changed between *program* and *run* mode by a personnel computer connected to the PLC processor.