Almamoun University collage

Power electrical Engineering

المسيطرات الرقمية والمعالج الدقيق

Third year Class

Dr Hussam Asper

2stSemester
2023 / 2024

Lecture 6

PLC Programming Languages

The term *PLC programming language* refers to the method by which the user communicates information to the PLC. The standard IEC 61131 (Figure 5-12) was established to standardize the multiple languages associated with PLC programming by defining the following five standard languages:

- **Ladder Diagram (LD)** —a graphical depiction of a process with rungs of logic, similar to the relay ladder logic schemes that were replaced by PLCs.
- **Function Block Diagram (FBD)** —a graphical depiction of process flow using simple and complex interconnecting blocks.
- **Sequential Function Chart (SFC)** —a graphical depiction of interconnecting steps, actions, and transitions.
- **Instruction List (IL)** —a low-level, text-based language that uses mnemonic instructions.
- **Structured Text (ST)** —a high-level, text-based language such as BASIC, C, or PASCAL specifically developed for industrial control applications.

Ladder diagram language is the most commonly used PLC language and is designed to mimic relay logic. The ladder diagram is popular for those who prefer to defi ne control actions in terms of relay contacts and coils, and other functions as block instructions. Figure 5-13 shows a comparison of ladder diagram programming and instruction list programming. Figure 5-13a shows

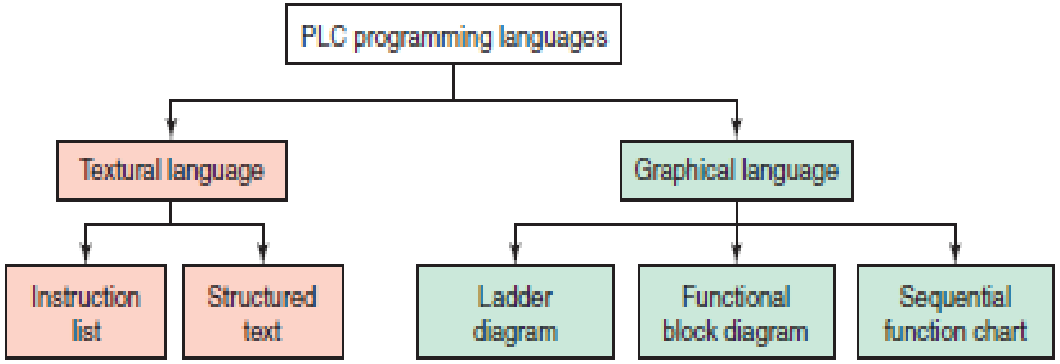
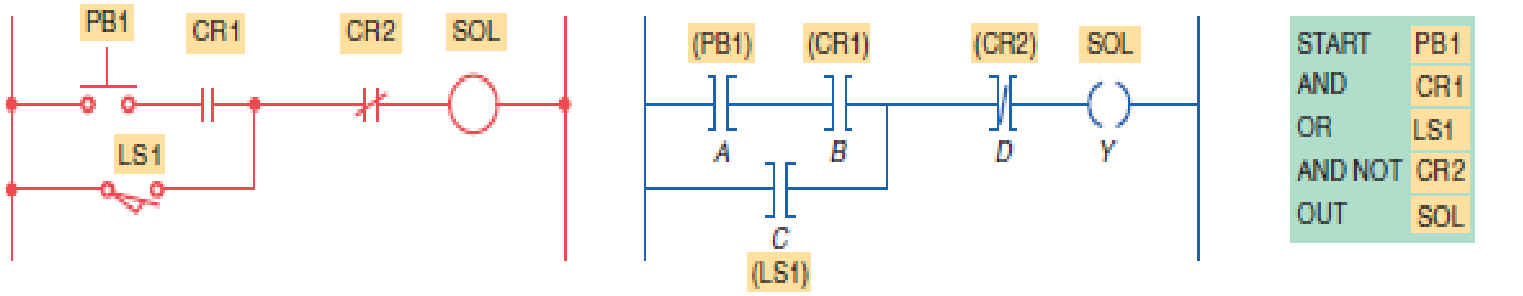


Figure 5-12 Standard IEC 61131 languages associated with PLC programming.



(a) Hardwired relay control circuit

(b) Equivalent ladder diagram (LD) program

(c) Equivalent instruction list (IL) program

Figure 5-13 Comparison of ladder diagram and instruction list programming.

the original relay hardwired control circuit. Figure 5-13b shows the equivalent logic ladder diagram programmed into a controller. Note how closely the ladder diagram program closely resembles the hardwired relay circuit. The input/output addressing is generally different for each PLC manufacturer. Figure 5-13c show how the original hardwired circuit could be programmed using the instruction list programming language. Note that the instructional list consists of a series of instructions that refer to the basic AND, OR, and NOT logic gate functions.

Functional block diagram programming uses instructions that are programmed as blocks wired together on screen to accomplish certain functions. Typical types of function blocks include logic, timers, and counters. Functional block diagrams are similar in layout to electrical/electronic block diagrams used to simplify complex systems by showing blocks of functionality. The primary concept behind a functional block diagram is data flow. Function blocks are linked together to complete a circuit that satisfies a control requirement. Data flow on a path from inputs, through function blocks or instructions, and then to outputs.

The use of function blocks for programming of programmable logic controllers (PLCs) is gaining wider acceptance. Rather than the classic contact and coil representation of ladder diagram or relay ladder logic programming, function blocks present a graphical image to the programmer with underlying algorithms already defined. The programmer simply completes needed information within the block to complete that phase of the program. Figure 5-14 shows function block diagram equivalents to ladder logic contacts.

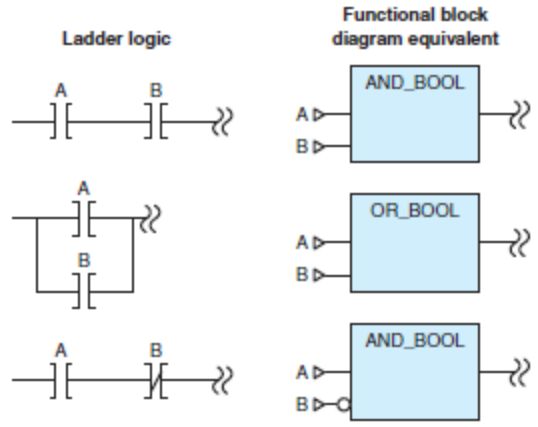


Figure 5-14 Function block diagram equivalents to ladder logic contacts.

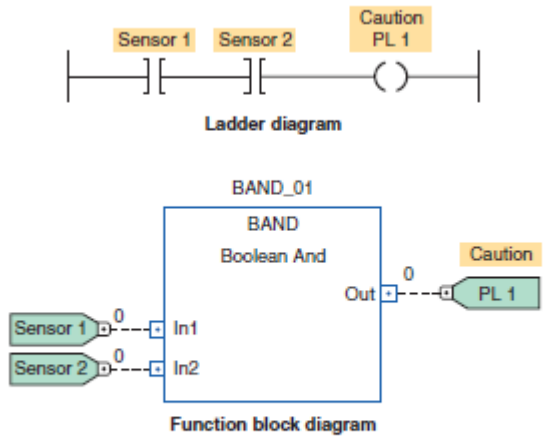


Figure 5-15 PLC ladder and equivalent function block diagram.

Figure 5-15 illustrates how ladder diagram and functional block diagram programming could be used to produce the same logical output. For this application, the objective is to turn on caution pilot light PL 1 whenever both sensor switch 1 and sensor switch 2 are closed. The ladder logic consists of a single rung across the power rails. This rung contains the two input sensor instructions programmed in series with the pilot light output instruction. The function block solution consists of a logic *Boolean And* function block with two input references tags for the sensors and a single output reference tag for the pilot light. Note there are no power rails in the function block diagram.

Sequential function chart programming language is similar to a flowchart of your process. SFC programming is designed to accommodate the programming of more advanced processes. This type of program can be split into steps with multiple operations happening in parallel branches. The basic elements of a sequential function chart program are shown in Figure 5-16 .

Structured text is a high level text language primarily used to implement complex procedures that cannot be easily expressed with graphical languages. Structured text uses statements to defi ne what to execute. Figure 5-17 illustrates how structured text and ladder diagram programming could be used to produce the same logical output.

For this application, the objective is to energize SOL 1 whenever either one of the two following circuit conditions exists:

- Sensor 1 and Sensor 2 switches are both closed.
- Sensor 3 and Sensor 4 switches are both closed and Sensor 5 switch is open.

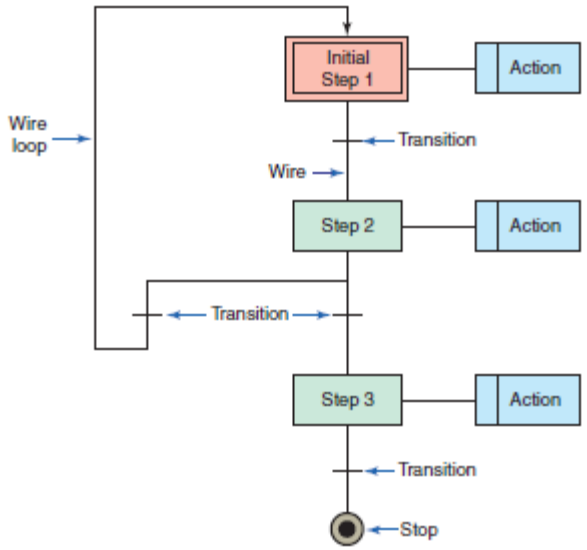
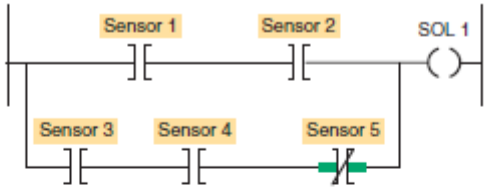


Figure 5-16 Major elements of a sequential function chart program.



Ladder diagram (LD) program

```
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;
```

Structured text (ST) program

Figure 5-17 PLC ladder and equivalent structured text program.

Relay-Type Instructions

The ladder diagram language is basically a *symbolic* set of instructions used to create the controller program. These ladder instruction symbols are arranged to obtain the desired control logic that is to be entered into the memory of the PLC. Because the instruction set is composed of contact symbols, ladder diagram language is also referred to as *contact symbology.*

Representations of contacts and coils are the basic symbols of the logic ladder diagram instruction set. The three fundamental symbols that are used to translate relay control logic to contact symbolic logic are Examine If Closed (XIC), Examine If Open (XIO), and Output Energize (OTE). Each of these instructions relates to a single bit of PLC memory that is specified by the instruction's address.

The symbol for the *Examine If Closed (XIC)* instruction is shown in Figure 5-18 . The XIC instruction, which is also called the Examine-on 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. This instruction asks the PLC's processor to examine if the contact is *closed.* It does this by examining the bit at the memory location specified by the address in the following manner:

- The memory bit is set to 1 or 0 depending on the status of the input (physical) device or internal (logical) relay address associated with that bit.
- A 1 corresponds to a true status or on condition.
- A 0 corresponds to a false status or off condition.
- When the Examine-on instruction is associated with a physical input, the instruction will be set to 1 when a physical input is present (voltage is applied to the input terminal), and 0 when there is no physical input present (no voltage applied to the inputterminal).
- When the Examine-on instruction is associated by address with an internal relay, then the status of the bit is dependent on the logical status of the internal bit with the same address as the instruction.
- If the instruction memory bit is a 1 (true) this instruction will allow rung continuity through itself, like a closed relay contact.
- 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.

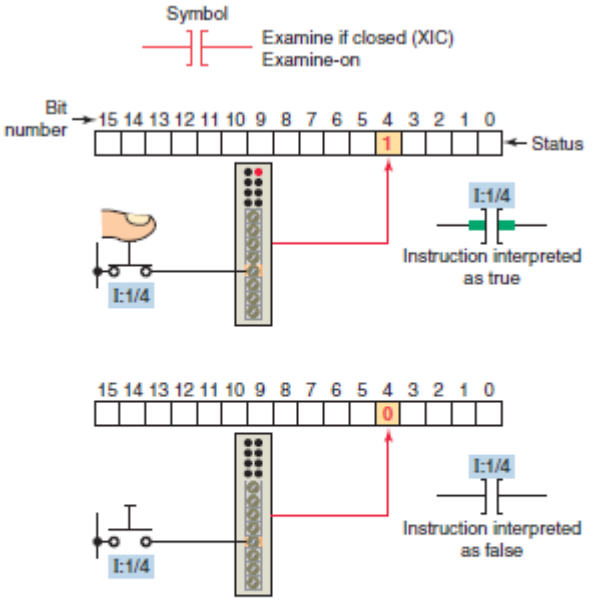


Figure 5-18 Examine If Closed (XIC) instruction.

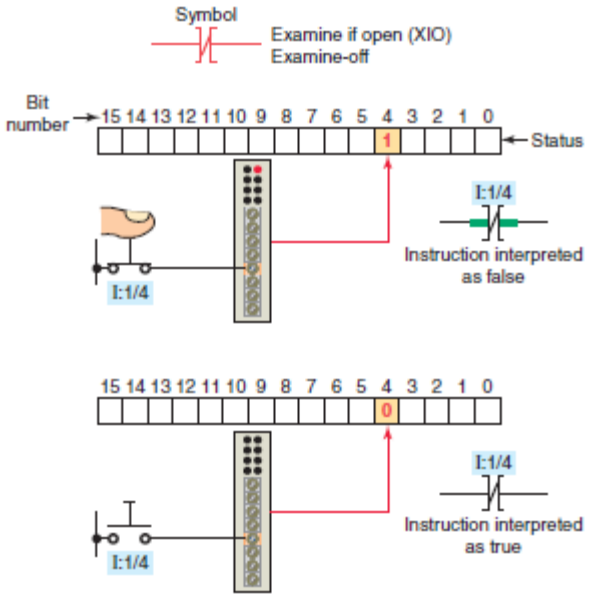


Figure 5-19 Examine If Open (XIO) instruction.

The symbol for the *Examine If Open (XIO)* instruction is shown in Figure 5-19 . The XIO instruction, which is also called the Examine-off instruction, looks and operates like a normally closed relay contact. Associated with each XIO instruction is a memory bit linked to the status of an input device or an internal logical condition in a rung. 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 in the following manner:

- As with any other input the memory bit is set to 1 or 0 depending on the status of the input (physical) device or internal (logical) relay address associated with that bit.
- A 1 corresponds to a true status or on condition.
- A 0 corresponds to a false status or off condition.
- When the Examine-off instruction is used to examine a physical input, then the instruction will be interpreted as false when there is a physical input (voltage) present (the bit is 1) and will be interpreted as true when there is no physical input present (the bit is 0).
- If the Examine-off instruction were associated by address with an internal relay, then the status of the bit would be dependent on the logical status of the internal bit with the same address as the instruction.
- Like the Examine-on instruction, the status of the instruction (true or false) determines if the instruction will allow rung continuity through itself, like a closed relay contact.
- The memory bit always follows the status (true = 1 or false = 0) of the input address or internal address assigned to it. The interpretation of that bit, however, is determined by which instruction is used to examine it.
- Examine-on instructions always interpret a 1 status as true and a 0 status as false, while Examine-off instructions interpret a 1 status as false and a 0 status as true.

The symbol for the *Output Energize (OTE)* instruction is shown in Figure 5-20 . The OTE instruction looks and operates like a relay coil and is associated with a memory bit. This instruction signals the PLC to energize (switch on) or de-energize (switch off) the output. The processor makes this instruction true (analogous to energizing a coil) when there is a logical path of true XIC and XIO instructions in the rung. The operation of the Output Energize instruction can be summarized as follows:

- The status bit of the addressed Output Energize instruction is set to 1 to energize the output and to 0 to de-energize the output.
- If a true logic path is established with the input instructions in the rung, the OTE instruction is energized and the output device wired to its terminal is energized.
- If a true logic path cannot be established or rung conditions go false, the OTE instruction is de-energized and the output device wired to it is switched off.

Sometimes beginner programmers used to thinking in terms of hardwired relay control circuits tend to use the same type of contact (NO or NC) in the ladder logic program that corresponds to the type of fi eld switch wired to the discrete input. While this is true in many instances, it is not the best way to think of the concept. A better approach is to separate the action of the fi eld device from

the action of the PLC bits as illustrated in Figure 5-21 . 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. A signal present makes the NC bit (1) false; a signal absent makes the NO bit (0) true.

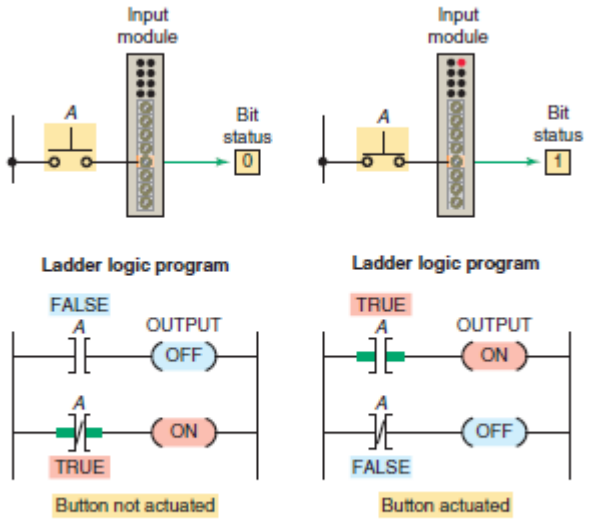


Figure 5-21 Separating the action of the field device and PLC bit.

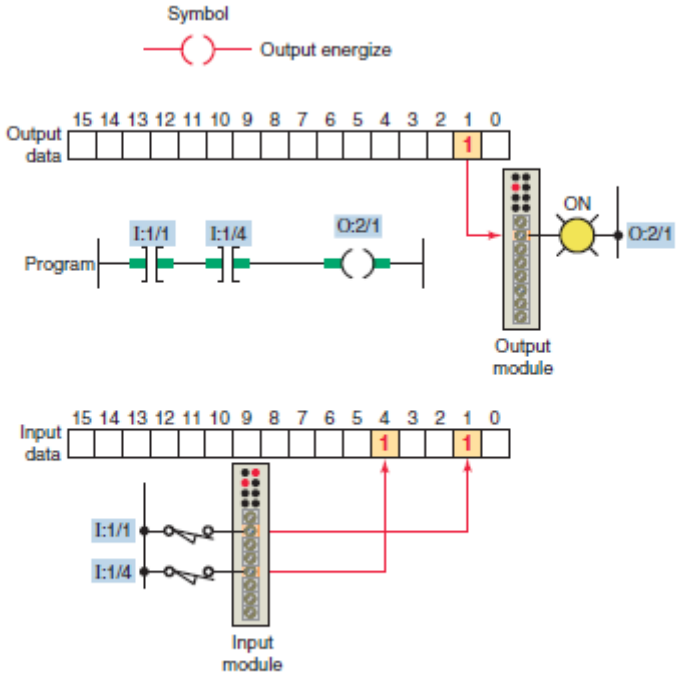


Figure 5-20 Output Energize (OTE) instruction.

The main function of the ladder logic diagram program is to control outputs based on input conditions, as illustrated in Figure 5-22 . This control is accomplished through the use of what is referred to as a ladder rung. In general, a rung consists of a set of input conditions, represented by contact instructions, and an output instruction at the end of the rung, represented by the coil symbol. 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. The number of ladder logic relays and input and output instructions is limited only by memory size. Most PLCs allow more than one output per rung.

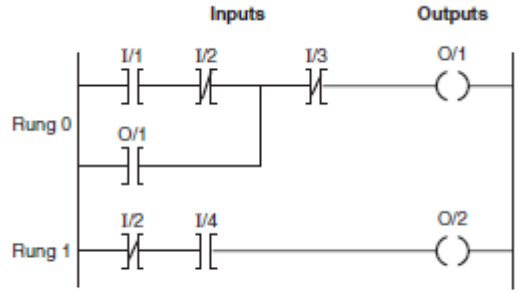


Figure 5-22 Ladder logic diagram rungs.

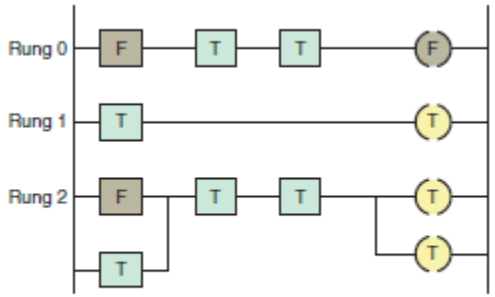


Figure 5-23 Logical continuity.

For an output to be activated or energized, at least one left-to-right true logical path must exist, as illustrated in Figure 5-23 . 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. The rung condition and OTE instruction are false if no logical continuity path has been established. During controller operation, the processor evaluates the rung logic and changes the state of the outputs according to the logical continuity of rungs.

Instruction Addressing

To complete the entry of a relay-type instruction, you must assign an address to each instruction. This address indicates what PLC input is connected to what input device and what PLC output will drive what output device.

The addressing of real inputs and outputs, as well as internals, depends on the PLC model used. Addressing formats can vary from one PLC family to another as well as for different manufacturers. These addresses can be represented in decimal, octal, or hexadecimal depending on the number system used by the PLC. The address identifies the function of an instruction and links it to a particular bit in the data table portion of the memory. Figure 5-24 shows the addressing format for an Allen-Bradley SLC 500 controller. Addresses contain the slot number of the module where input or output devices are connected. Addresses are formatted as fi le type, slot number, and bit.

The assignment of an I/O address can be included in the I/O connection diagram, as shown in Figure 5-25 . Inputs and outputs are typically represented by squares and diamonds, respectively.

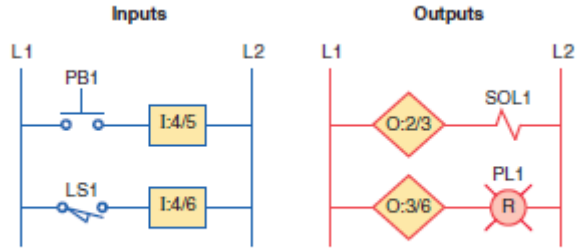


Figure 5-25 I/O connection diagram.

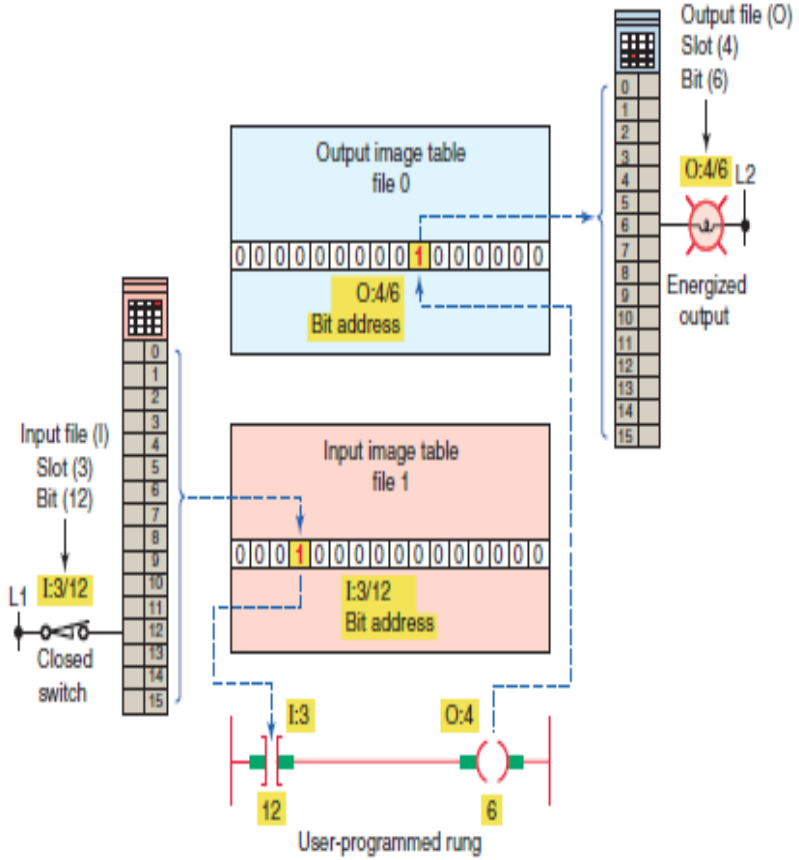


Figure 5-24 Addressing format for an Allen-Bradley SLC 500 controller.