LADDER DIAGRAMS

The ladder diagram, also referred to as a schematic or elementary diagram, is used by the electrician or technician to speed their understanding of how a circuit works. Figure 7–2 shows the same circuit as Figure 7–1, but in ladder diagram form.

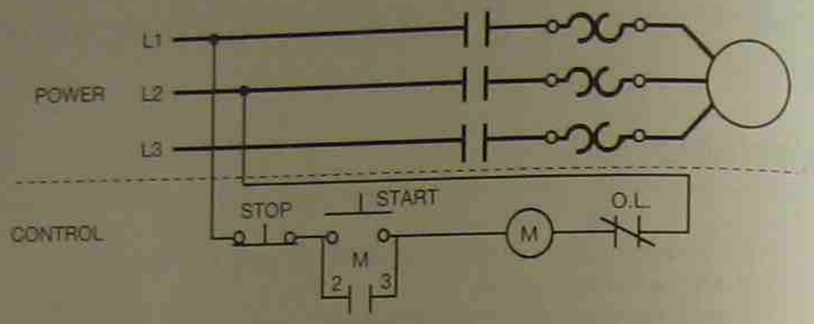


Figure 7–2 Ladder Diagram

To simplify the circuit and help to understand its configuration, the power portion of the circuit is shown separate from the control portion. No attempt is made to show the actual physical location of the components. Since the motor connections (power portion) are the same for any three-phase motor, it is common practice not to show the motor starter or the motor. By not showing the power portion of the circuit, a ladder diagram is created, showing only the control portion of the diagram (Figure 7–3).

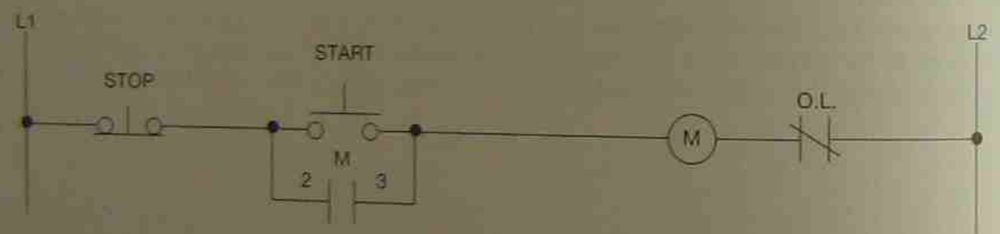


Figure 7–3 Simplified Ladder Diagram

The power required for the control circuit is always shown as two vertical lines, while the actual line(s) of logic are drawn as horizontal lines. The power lines, or rails as they are often called, are like vertical sides of a ladder, whereas the horizontal logic lines are like the rungs of a ladder.

When referring back to Figure 7–1, it is easy to see the physical relationship between the *STOP/START* station, the motor starter coil (M), the overload contacts (O.L.), and the holding contacts (2 and 3), but it is difficult to determine the electrical relationship. The ladder diagram in Figure 7–3, however, clearly shows the electrical relationship between all of the control circuit components.

LADDER DIAGRAM RULES

Some basic rules for ladder diagrams are as follows:

1. A ladder diagram is read like a book; from left to right and from top to bottom.
2. The vertical power lines (rails) of the ladder diagram represent the *voltage potential* of the circuit. The potential could be AC or DC, and varies in voltage from 6 V to 480 V. Standard labeling for the rails is L1 and L2. L1 is AC high or hot for AC circuits, and positive or plus (+) for DC circuits. L2 is AC low or neutral for grounded AC circuits, and negative or minus (–) for DC circuits. The rails may also be marked X1 and X2 when the voltage potential is derived from a transformer.
3. Devices or components are shown in order of importance whenever possible. In Figure 7–3 the *STOP* button is shown ahead of the *START* button. For safety reasons, the *STOP* button has a higher order of importance than the *START* button.
4. Electrical devices or components are shown in their normal condition. The normal condition of electrical diagrams is the circuit deenergized (*OFF*) and with no external forces such as pressure, or flow, etc., acting on the device. The *STOP* button is shown closed because that is the normal position for the *STOP* button. The holding contacts (2 and 3) of coil M are shown open. This is the normal position for these contacts when coil M is deenergized. The normally open (N.O.) M holding contacts 2 and 3 do not close until there is a complete path for current flow to coil M. When coil M energizes, M contacts 2 and 3 close, providing a parallel path for current flow with the *START* button.
5. Contacts associated with relays, timers, motor starters, and the like, always have the same number or letter designation as the device that controls them. This labeling method holds true no matter where the contacts(s) appear in the circuit. For example, in Figure 7–3 the N.O. holding contacts 2 and 3 are controlled (activated) by motor starter coil M. Therefore, the contacts are identified with the letter M.
6. All contacts associated with a device change position when the device is energized.

   Figure 7–4 shows a control relay (CR) controlled by a switch (S-1) on Rung 1 of the ladder diagram. Rung 2 shows a normally closed (N.C.) control relay contact in series with a green indicator lamp. Rung 3 shows a normally open (N.O.) control relay contact in series with a red indicator light.

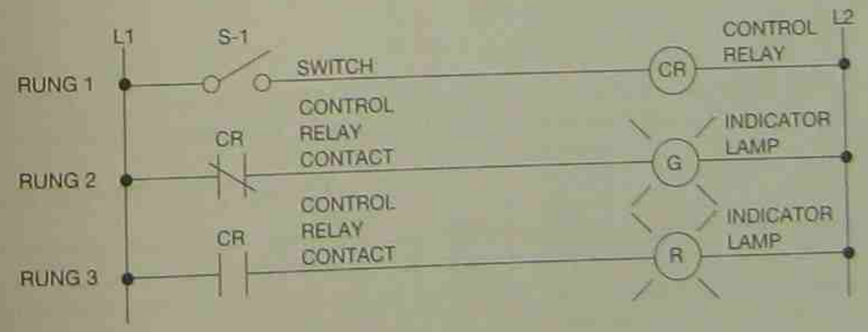


Figure 7–4 Three Rung Ladder Diagram

When power is applied to the rails of the ladder diagram, the only device in the circuit that operates is the green indicator lamp. The green indicator lamp lights due to a complete path for current flow through the normally closed (N.C.) control relay contacts. These contacts are normally closed and only change position and open when the control relay in Rung 1 is energized. When switch S-1 is closed, completing the path for current flow and energizing the CR in Rung 1, the N.C. CR contacts in Rung 2 open, while the N.O. CR contacts in Rung 3 close. The action of the contacts will turn *OFF* the green lamp in Rung 2 and turn *ON* the red lamp in Rung 3. As long as the control relay remains energized through S-1, the normally closed contact in Rung 2 remains open, and the normally open contact in Rung 3 remains closed. When S-1 is opened and CR deenergizes, the contacts controlled by CR will return to their normal state (N.C. in Rung 2 and N.O. in Rung 3).

7. In a ladder diagram, devices that perform a *STOP* function are normally wired in series. Figure 7–5 shows two switches wired normally closed (N.C.) that control a green indicator lamp.

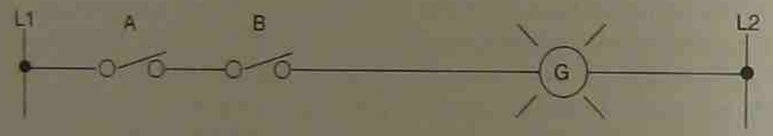


Figure 7–5 Two Switches Wired in Series

With the two switches wired in series, both A and B must remain closed for the lamp to remain lit. If either switch is opened, the green lamp will go out. When switches and/or contacts are wired in series, they are said to have an AND relationship. The AND relationship requires that both A *and* B must be closed for the lamp to light. A truth table for this concept is shown in Figure 7–6.

SWITCH	SWITCH	INDICATOR LAMP
A	B	G
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

Figure 7–6 Truth Table for Series Devices

8. Devices that perform a *START* function are normally wired in parallel. Figure 7–7 shows two switches (A and B) wired in parallel to control a red indicator lamp. In this configuration, if either switch A *or* B is closed, the red lamp will light.

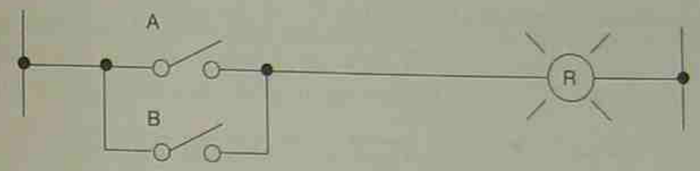


Figure 7–7 Two Switches Wired in Parallel

When switches or contacts are wired in parallel, they are said to have an OR relationship. The OR relationship requires that either A *or* B be closed for the red indicator lamp to light. A truth table for this concept is shown in Figure 7–8.

SWITCH	SWITCH	INDICATOR LAMP
A	B	R
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON

Figure 7–8 Truth Table for Parallel Devices

With this understanding of what a ladder diagram is, and the rules that apply to it, a discussion of a basic motor *STOP/START* circuit (shown in Figure 7–2) can begin.

BASIC *STOP/START* CIRCUIT

As stated earlier in this chapter, the wiring diagram in Figure 7–1 is great for showing actual physical location of the circuit wiring and the components. It does not, however, show the electrical relationship of the devices as simply as the ladder diagram. The wiring diagram is used for original installation and some troubleshooting, whereas the ladder diagram is used to show the electrical relationship of the components, and to speed understanding of how the circuit works.

From viewing the ladder diagram in Figure 7–9, it can be seen that when power is applied to the circuit, the motor starter coil M cannot energize because there is an incomplete path for current flow due to the open *START* button and the normally open M contacts (2 and 3). The *START* button and the N.O. M contacts are wired in parallel and have an OR relationship. When the *START* button is pushed, a path for current exists from L1 potential through the normally closed *STOP* button, through the now closed *START* button through the coil of the motor starter (M), and on through the N.C. overload contacts to L2 potential.

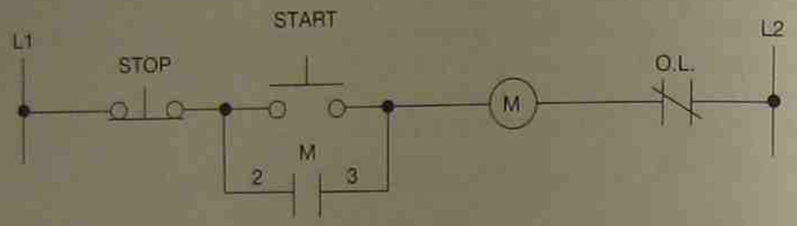


Figure 7–9 Ladder Diagram for Basic *STOP/START* Circuit

When the starter coil M energizes, the M contacts (2 and 3) close, providing an alternate path for current flow. At this point, the *START* button could be released, and the circuit would remain energized, or held in, by the holding contacts (2 and 3) of the motor starter. When contacts from a motor starter or other device are wired in this fashion, they are often referred to as holding, maintaining, or sealing contacts as the circuit is held, maintained, or sealed-in after the *START* button is released.

When the holding contacts (2 and 3) are closed, the main motor contacts of the motor starter are also closed and the motor is started. The operation of the motor is normally taken for granted and is not shown on the ladder diagram. By keeping the ladder diagram as simple and uncluttered as possible, the relationship between components and how the control portion of the circuit works is greatly enhanced.

Figure 7–10 again shows the wiring diagram of a motor *STOP/START* circuit. While this diagram looks entirely different from the ladder diagram, both are electrically the same. This comparison shows the electrician or technician why the ladder diagram is preferred.

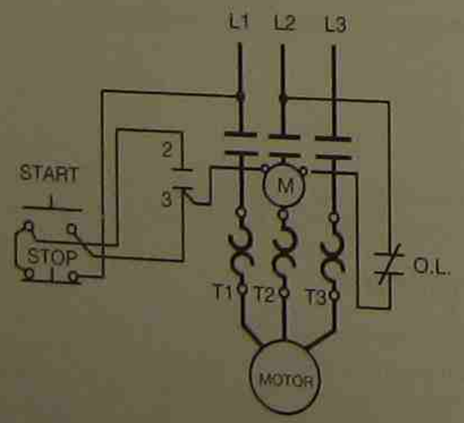


Figure 7–10 Wiring Diagram for Basic *STOP/START* Circuit

The ladder diagram has been the "working language" of electricians and electrical engineers for many years, and helps explain why most programmable controllers are programmed using ladder logic. While this method of programming is welcomed by some, it has frustrated other PLC users who have not been exposed to, or trained in, RELAY LADDER LOGIC.

SEQUENCED MOTOR STARTING

Relay ladder diagrams can become large and complex. It is not the purpose of this text to cover them in great detail, but instead to discuss the basic rules and present some concepts to enhance understanding of circuits that are discussed in later chapters.

Figure 7–11 shows a ladder diagram for a circuit that starts three motors.

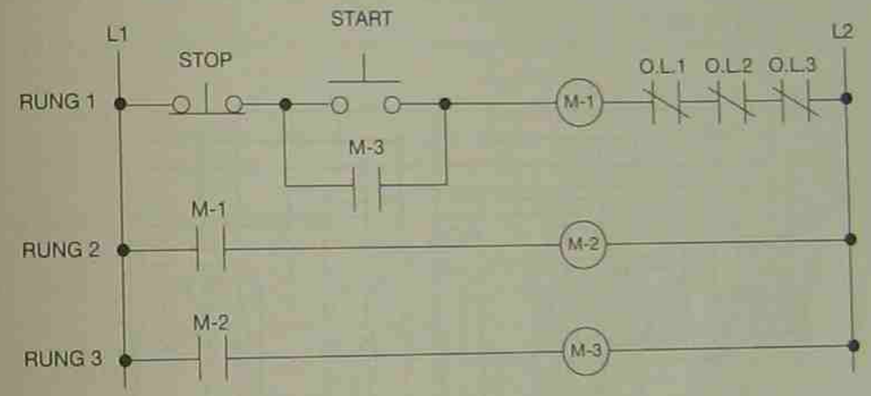


Figure 7–11 Three Motor Start Circuit

Rung 1 contains the *STOP/START* buttons and the motor starter coil M-1 for Motor 1. Notice that the holding contacts wired in parallel with the *START* button are not M-1 contacts, but instead are M-3 contacts. With this arrangement, Rung 1 cannot be sealed in, or maintained, unless Motor Starter 3 energizes and closes its contacts. Additionally, the M-1 contacts in Rung 2 must close to energize Motor Starter 2 (M-2) and M-2 contacts in turn must close in Rung 3 to energize Motor Starter 3 (M-3). When the *START* button of this circuit is pushed, it operates as follows:

1. M-1 energizes, closing the N.O. M-1 contacts in Rung 2 and energizes M-2.
2. M-2 N.O. contacts close in Rung 3 and energize M-3.
3. M-3 N.O. contacts in Rung 1 close and act as holding contacts to keep the circuit energized after the *START* button is released.

Note: *This sequence happens almost instantaneously.*

4. Pushing the *STOP* button deenergizes M-1 which deenergizes M-2 in Rung 2 when the normally open M-1 contacts go open. M-2's deenergizing opens the M-2 contacts in Rung 3 and deenergizes M-3. With M-3 deenergized, the N.O. M-3 contacts in Rung 1 open.

5. By wiring all three overload contacts in series with M-1 in Rung 1, an overload on any motor would shut down all motors. An open overload contact would have the same effect as pushing the *STOP* button.

It could be said that this circuit consists of basically three elements: inputs, outputs, and logic.

The inputs consist of the *STOP* button, the *START* button, and the overload contacts. The outputs are motor starters M-1, M-2, and M-3. The logic that caused the sequential starting were normally open contacts M-1, M-2, and M-3.

These three elements—inputs, outputs, and logic—also work well with programmable controllers. The inputs are wired to input modules, the outputs are wired to output modules, and the processor performs the logic functions.

Figure 7–12 shows the wiring diagram for the three-motor circuit just discussed. This diagram further illustrates the point, that while wiring diagrams are great for giving the physical location of components, they do not show the control function of the circuit as clearly as a ladder diagram does.

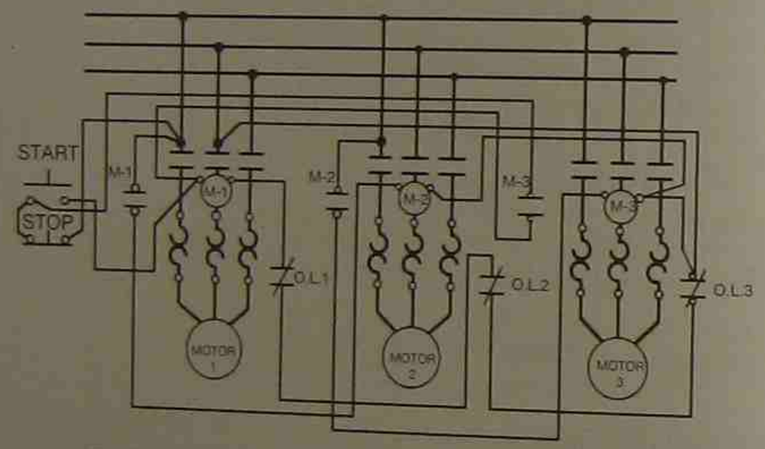


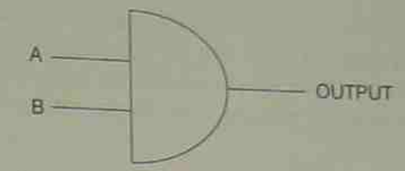
Figure 7–12 Wiring Diagram for Three Motor Circuit

DIGITAL LOGIC GATES

While the typical PLC is programmed using ladder logic symbols, some are programmed using digital logic notations such as AND, OR, NOT, etc. To better understand these digital logic notations and to see how they compare to relay ladder logic, we will cover six basic digital logic gates.

Figure 7–13 shows a two input AND gate.

From the truth table, we see that both inputs A *and* B must be TRUE, or set to 1, before the output is turned *ON*, or set to 1. The AND gate functions like the two switches that were wired in series to a lamp in Figure 7–5. Both Switch A and Switch B had to be closed for the lamp to light.



INPUT A	INPUT B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

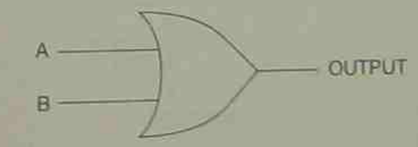
Figure 7–13 Two Input AND Gate with Truth Table

Figure 7–14 shows AND logic for two programmed input devices wired in series to an output device.

A	B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

Figure 7–14 Two Input Devices Wired in Series with Truth Table

Figure 7–15 shows a two input OR gate.



A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

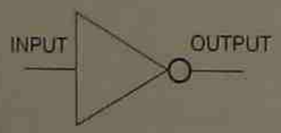
Figure 7–15 Two Input OR Gate with Truth Table

From the truth table, we see that if either input A *or* B are TRUE, or set to 1, the output will be turned *ON*, or set to 1. The OR gate functions like the two switches that were wired in parallel to a lamp in Figure 7–7. If either switch A or B was closed, the lamp would light. Figure 7–16 shows OR logic for two input devices wired in parallel to an output device.

A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

Figure 7–16 Two Input Devices Wired in Parallel with Truth Table

The next gate is called a NOT gate and is often referred to as an inverter. The inverter, or NOT gate, will have only one input lead and one output lead. If the input is *OFF*, or set to 0, then the output will be *ON*, or set to 1. If the input is *ON*, or set to 1, then the output will be *OFF*, or set to 0. Figure 7–17 shows a NOT gate with a truth table. The circle in the output line is used to indicate an inverted function.

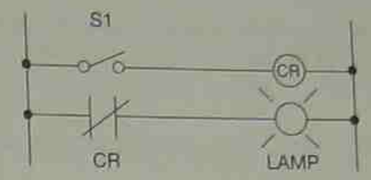


INPUT	OUTPUT
0	1
1	0

NOT (INVERTER)

Figure 7–17 NOT Gate with Truth Table

The NOT gate functions like the normally closed contacts in Rung 2 for the three-rung ladder diagram in Figure 7–4. As long as the control relay (CR) in Rung 1 remains deenergized, or *OFF*, the normally closed CR contact in Rung 2, that controls the green indicator lamp, will be TRUE and the the indicator lamp will be *ON*. Figure 7–18 shows a normally closed CR contact controlling a lamp and the truth table for the circuit.



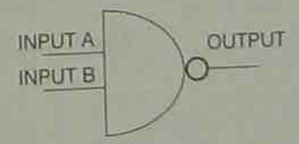
CR	LAMP
0	1
1	0

Figure 7–18 Normally Closed CR Contact Controlling a Lamp with NOT logic.

As long as the single pole switch (S_1) that is wired in series with the CR coil is open, CR is *OFF*, or set to 0. With CR *OFF*, the logic for the normally closed contacts will be TRUE, and the lamp will be *ON*. When S_1 is closed, CR will energize, turn *ON*, the normally closed CR contacts will open, and the light will be turned *OFF*. The truth table reflects the action of the CR coil that controls the action of the CR contacts. It may help to understand the truth table if we think that the normally closed CR contacts are controlled by the CR coil. If the CR coil of *OFF*, or set to 0, we can think of the CR normally closed contacts also being closed, or set to 1. As the input will be inverted when the CR coil is energized, or set to 1, then the output device controlled by the NC contacts will be FALSE, or set to 0.

The programmable controller will use NOT logic in the same way as described above. If the output device is set to 0, or *OFF*, any normally closed contacts associated with the output device with the same address will also be set to 1. The NOT logic will be used in the next chapter for the Examine Off instruction. When a normally closed contact is addressed with the same address as an output coil, the normally closed contact will be true as long as the output coil is *OFF*, or FALSE.

By combining the NOT gate with the AND gate, we get what is called a NAND gate. Figure 7–19 shows a two-input NAND gate with a truth table.

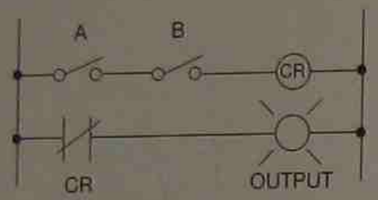


A	B	OUTPUT
0	0	1
0	1	1
1	0	1
1	1	0

Figure 7–19 NAND Gate with Truth Table

As discussed with the NOT logic, the circle is used to indicate an invert function. By placing the invert, or NOT, symbol at the output of the AND gate, the output can only be TRUE when one or both of the inputs are FALSE or set to 0. Figure 7–20 shows the equivalent relay circuit.

To help understand the logic of the NAND gate, consider the ladder diagram in Figure 7–20.

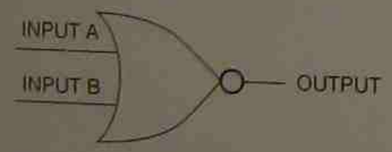


A	B	OUTPUT
0	0	1
0	1	1
1	0	1
1	1	0

Figure 7–20 Ladder Diagram with NAND Logic

With both inputs A and B *OFF*, or set to 0, the inverted output will be set to 1, or be turned *ON*. If only input A is set to 1, the inverted output will remain set to 1 because input B is still open, or set to 0. If input A is opened and input B is closed or is set to 1, the inverted output will again remain set to 1. However, If both A and B are closed, or set to 1, then the inverted output will be set to 0, or *OFF*. From the ladder diagram, we can see that with both input devices A and B open, CR would not be energized and the output would be set to 1, or *ON*. For the output device to go FALSE, or to 0, both input switches A *and* B will be closed, or set to 1, as illustrated in the truth table.

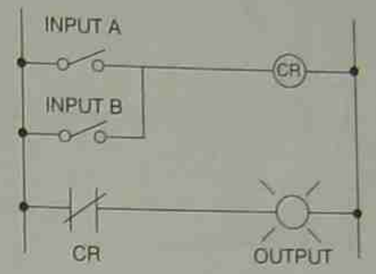
When we combine the NOT gate with an OR gate, we get what is called a NOR gate. Figure 7–21 shows the NOR gate with a truth table.



A	B	OUTPUT
0	0	1
0	1	0
1	0	0
1	1	0

Figure 7–21 NOR Gate with Truth Table

To help understand the logic of the NOR gate, consider the ladder diagram in Figure 7–22.

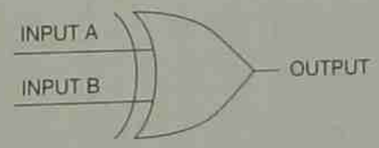


A	B	OUTPUT
0	0	1
0	1	0
1	0	0
1	1	0

Figure 7–22 Ladder Diagram with NOR Logic

From this figure we can see that if either input A *or* B is closed the control relay (CR) coil will energize and the normally closed contacts in Rung 2 will open and turn *OFF* the output. With this configuration, the only time the output lamp will light is if both switch A and B are open, or set to 0.

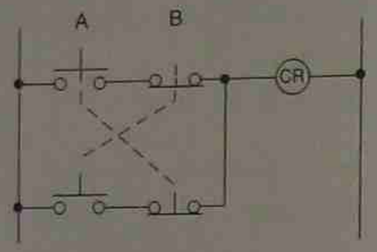
The final logic gate that will be covered is the Exclusive OR gate (XOR). The exclusive OR gate with truth table is shown in Figure 7–23.



A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

Figure 7–23 XOR Logic Gate with Truth Table

The XOR logic gate will only turn the output *ON* when *either* input A *or* B is *ON*, but *not both ON*. This logic gate can be compared to the two double-circuit pushbuttons shown in Figure 7–24a.



A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

Figure 7–24a Ladder Logic Circuit Equivalent to the XOR Gate

This ladder diagram shows us that as long as both pushbuttons are not pushed, the output device will not be turned *ON*. If button A is pushed, a complete path for current now exists, as shown in Figure 724b. Similarly, if button B is pushed, a complete path for current now exists, as shown in Figure 7–24c.

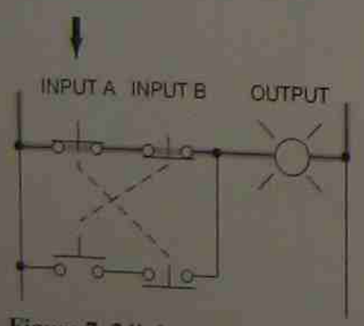


Figure 7–24b Input A Closed and Output is *ON*

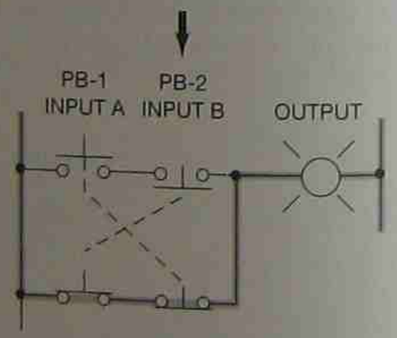
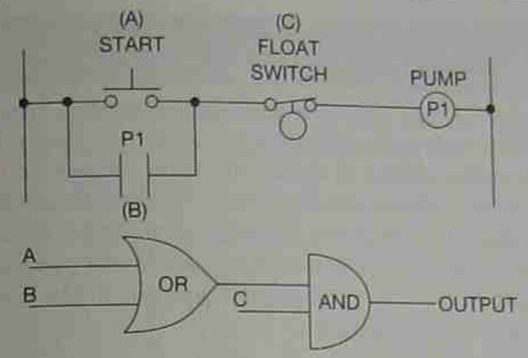


Figure 7–24c Input B Closed and Output is *ON*

Logic gates can be combined to created very complicated control logic. Figure 7–25 shows an OR gate combined with an AN gate to duplicate the logic of a ladder diagram that contains a start button, holding contacts, float switch, and pump starter. Compare the logic of the ladder diagram with the logic gate equivalent circuit and the truth table.



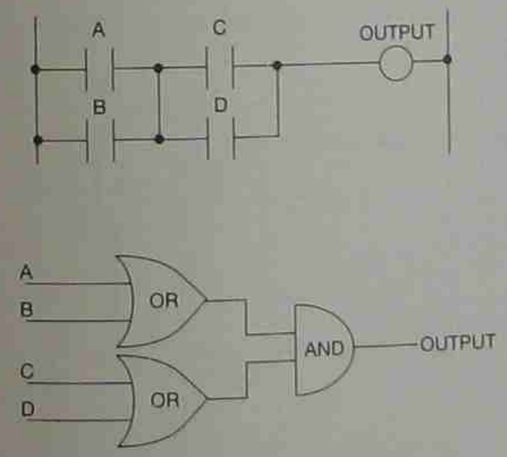
A	B	C	OUTPUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Figure 7–25 Combining an OR gate and an AND gate.

Figure 7–26 shows a ladder diagram with four contacts that control an output device. We can see from the diagram that to turn on the output, the following combinations of contacts are required.

A and C
A and D
B and D
B and C

By combining two OR gates and an AND gate, we can duplicate the logic of the four contacts, as shown in Figure 7–26 and verified by the truth table.



A	B	C	D	E
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Figure 7–26 Combining OR and AND Gates

Chapter Summary

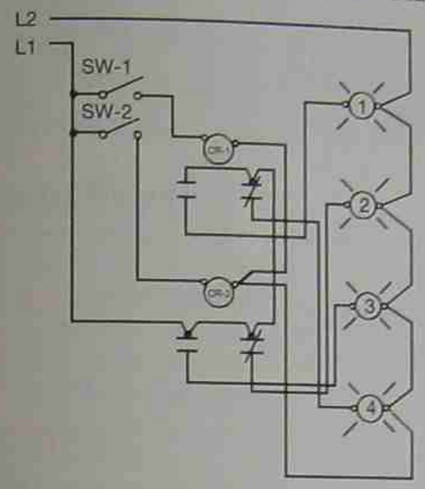
There are basically two types of electrical diagrams: wiring diagrams and ladder diagrams. Wiring diagrams show actual physical location and wiring, whereas ladder diagrams show electrical relationships. The simplified ladder diagram speeds understanding of circuit operation and is used for circuit design and troubleshooting. The vertical sides of the ladder diagram are referred to as *rails*, while the horizontal lines or logic are called **rungs**. On electrical diagrams, devices are always shown in their normal or deenergized condition. When two or more devices are wired in series, they perform an AND function, while two or more devices wired in parallel perform an OR function. The elements of the ladder diagram are inputs, outputs, and logic.

Logic gates can be used that duplicate the logic of the pushbuttons, contacts, and control device typically used in motor control circuits. Common logic gates are AND, OR, NOT, NOR, NAND, and XOR (Exclusive OR). Logic gates can be combined to create complex control circuits. The elements of the ladder diagram are inputs, outputs, and logic. Ladder diagrams are favored over wiring diagrams when a basic understanding of the control circuit is needed. The wiring diagram is used to show actual physical location and relationships between components, whereas the ladder diagram shows electrical relationships without regard to actual location.

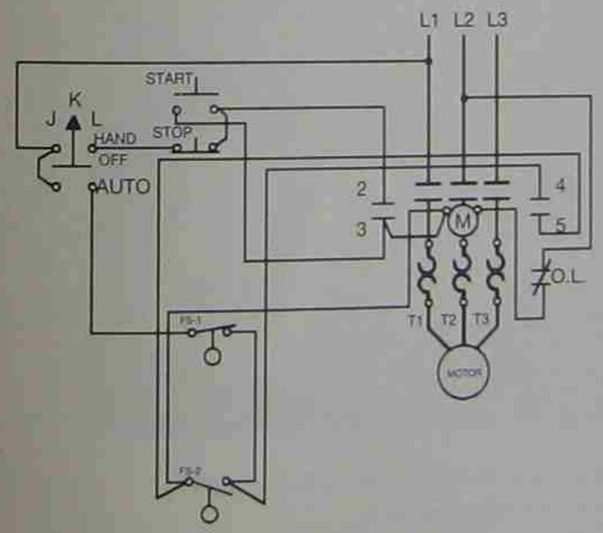
Review Questions

1. Define the terms *normally open* and *normally closed*.
2. Describe the difference between a wiring diagram and a ladder (schematic) diagram.
3. Explain the operation of the circuit in Figure 7–9 if M contacts 2 and 3 do not close.
4. Contacts wired in parallel have what relationship?
   a. AND
   b. OR
5. Contacts wired in series have what relationship?
   a. AND
   b. OR
6. The two main vertical lines of a ladder diagram are often referred to as:
   a. rungs
   b. power ports
   c. rails
   d. tracks
   e. none of the above
7. The horizontal lines of a ladder diagram are referred to as:
   a. rungs
   b. power ports
   c. rails
   d. tracks
   e. none of the above
8. Devices that are intended to perform a *STOP* function are normally wired in _____ with each other.

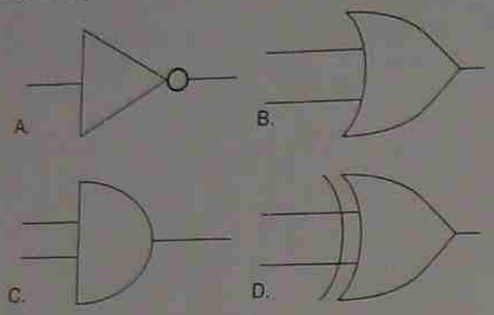
9. Devices that are intended to perform a *START* function are normally wired in _____ with each other.
10. How are contacts that are associated with relays, motor starters, timers, and the like, identified?
11. Convert wiring diagram 7–A into a ladder diagram.



12. Convert wiring diagram 7–B into a ladder diagram.



13. Identify the following logic gates.



CHAPTER

8 Relay Type Instructions

Objectives

After completing this chapter, you should have the knowledge to:
- Understand the *EXAMINE ON* instruction.
- Understand the *EXAMINE OFF* instruction.
- Write and understand the logic for a standard *STOP/START* motor circuit.

The next step in understanding how the programmable logic controller works is to learn how ladder logic is changed into processor logic. The actual programming is accomplished using either a desktop, hand-held programming device, or computer. Figures 8–1a shows a Modicon Micro PLC and the Modicon Hand–Held Programmer for programming and monitoring. Figure 8–1b shows the keyboard of the hand-held programmer.

Figure 8–1a Modicon Micro and Hand-Held Programmer
(Courtesy of Modicon Inc.)

Regardless of the type of programmer used, some common relay symbols are standard. These symbols include normally open contacts, normally closed contacts, and coil or output. Figure 8–1b shows that the keyboard has no symbols for input devices such as *STOP* buttons, limit switches, and pressure switches. The contacts from all input devices are programmed using either the N.O. or N.C. relay contact symbols (above the letters A and B on the keyboard). The actual programming devices used by the different PLC manufacturers are covered in Chapter 9, but first the relay logic used for PLCs must be discussed and *understood*.

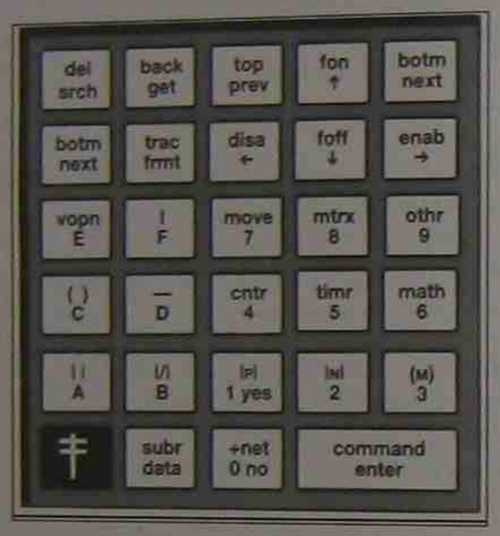


Figure 8–1b Modicon Hand-Held Programmer Keyboard

PROGRAMMING CONTACTS

A PLC is normally programmed using a ladder logic-type language. Ladder logic is a good choice for a programming language because it closely resembles the way circuits are hardwired. Electricians and technicians feel comfortable with, and understand, ladder logic, so programming with a ladder logic-type language makes good sense. While there are many similarities between standard RELAY LADDER LOGIC and the ladder logic used for programming a PLC, there are some distinct differences.

Hardwired contacts in a motor control circuit control the path for current flow to the output (coil, light, solenoid, etc.). The contact symbols that are used when programming a PLC are actually logic instructions that the processor uses to make decisions.

The PLC normally open (N.O.) contact symbol is actually an instruction that tells the processor to look for an *ON* condition at the address that corresponds to the symbol. If an *ON* condition exists, the instruction is said to be logically *true*, and logic continuity exists. This is much like saying that if a contact is closed, current can flow.

Note: *As stated earlier in the text, the terms current flow and power flow are often used by the various PLC manufacturers to indicate that a circuit is complete, or logically true.*

Figure 8–2 shows a simple circuit containing a single pole switch and a lamp for an output. Figure 8–3 shows the equivalent circuit when programmed with a PLC. Addresses shown are Allen-Bradley PLC-5 format. An "I" preceding a word and bit number indicates an input, whereas an "O" preceding a word and bit address indicates an output.

Figure 8–2 Simple Circuit

Figure 8–3 Equivalent Circuit Programmed with a PLC

In Figure 8–2, if the switch is closed, current flows through the switch contacts and the lamp lights. In Figure 8–3, the single-pole switch is shown as a normally open contact symbol with the address I:010/00. Referring back to the memory structure in Chapter 5, the address I:010/00 is bit 00 of I/O image word 010 using PLC-5 format. The lamp, or output, is shown as a circle, and is given the address O:012/01. The output is actually bit 01 of output image word 012.

Because of the way this program is written, the normally open contact symbol tells the processor to look at address location I:010/00 (the single-pole switch); if a closed (*ON*) condition is found, then the logic of the circuit is true. When the logic of the circuit is true, the processor is instructed to turn *ON* output O:012/01. There is, in reality, no actual electrical connection between the switch (I:010/00) and the lamp (O:012/01). Instead, it is the processor that turns the lamp *ON* or *OFF* depending on the logic of the program that is written and the status of the input device.

Figure 8–4 shows the switch and lamp as they are wired to their respective I/O modules. It is the N.O. contact symbol in the program that tells the processor to examine the single-pole switch for an *ON* condition. If the switch is open (*OFF*), the program logic is not true and the processor will not turn on the lamp. On the next processor scan, however, if the switch has been closed, it will be *ON*, the logic of the circuit will be true, and the processor will turn *ON* the lamp.

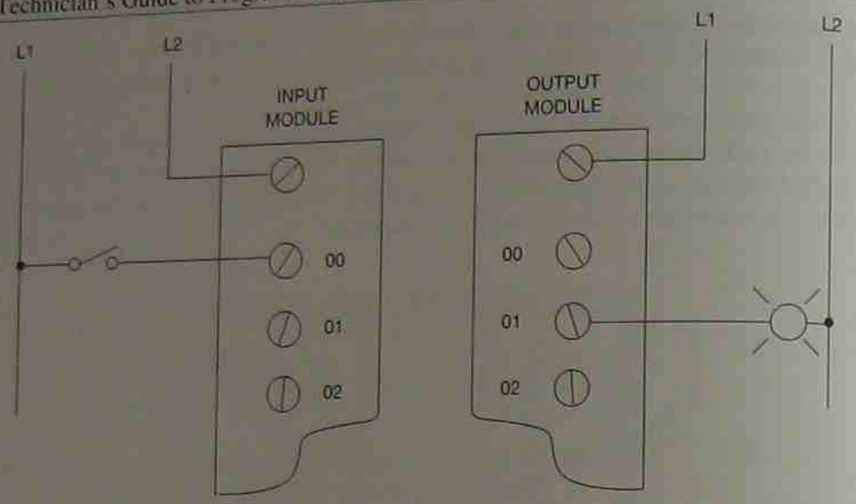


Figure 8–4 Input and Output Devices Wired to I/O Modules

Figure 8–5a shows the bit status of input image word 010 when the switch is open. With the switch open (*OFF*), the logic of the circuit cannot be true so the lamp (output image word 012 bit 01) will not be *ON*. The *OFF* condition of the switch and the lamp is indicated by a 0 in bit location 00 in input word 010 and output image word 012 bit 01. When the switch is closed, the bit that represents the switch (00) will change to a 1. This makes the circuit logically TRUE, the processor will turn the lamp to *ON* (bit 01), and a 1 will be shown in that location (Figure 8–5b).

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OUTPUT IMAGE WORD 012

Figure 8–5a Bit Status for I/O Word 1 With Switch Open

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

OUTPUT IMAGE WORD 012

Figure 8–5b Bit Status for I/O Word 1 After Switch is Closed

As the processor views the normally open contact instruction as a request to examine a given address for an *ON* condition, it is referred to an **EXAMINE ON** instruction. The opposite instruction, the normally closed contact, is referred to as an **EXAMINE OFF** instruction.

These instructions are also referred to as examine if closed (XIC) and examine if open (XIO) as shown in Table 8–1.

Table 8–1 Examine If closed (XIC) and Examine If Open (XIO)

Examine On	Examine Off
–] [–	–]/[–
Normally Open	Normally Closed
Examine if Closed	Examine if Open
XIC	XIO

The EXAMINE OFF instruction is only logically true when the device referenced is *OFF*, or open. Figure 8–6 shows the EXAMINE OFF instruction now used for address I:010/00. The processor is asked to examine the address location for an *OFF* (open) condition. If an *OFF* condition is found, then the instruction is logically true and the output O:012/01 would be turned *ON*. If, on the other hand, the switch was found to be *ON* (closed), the logic would be false and the lamp would not be turned *ON*.

Figure 8–6 EXAMINE OFF Instruction

At first these two instructions may seem to be contrary to the logic of hardwired contacts, so it is important to remember that these are instructions to the processor, and are not hardwired contacts. A review of both instructions seems appropriate.

EXAMINE ON

Whenever the processor sees an N.O. contact in the user program, it views the contact symbol as a request to examine the address of the contact for an *ON* condition. If the N.O. contact has an input address, and if the real-world input device is closed, or *ON*, the processor sets the appropriate bit in the input register to 1, or *ON*. As the EXAMINE ON instruction is looking for an *ON* condition, a bit set to 1, or *ON*, is a true condition, and a logic path exists through the contacts. If the real-world input had been open, or *OFF*, the processor would have cleared the appropriate bit to 0, or *OFF*, and the contact would be false as far as the logic of the ladder diagram was concerned and would not allow a logic path.

EXAMINE OFF

When the *N.C.* symbol is programmed in a ladder diagram, the processor views it as a request to examine the address of the contact for an *OFF* condition. Any address that is actually *OFF* becomes logically true and power can flow. If a N.C. contact has an input address and the real-world input device is open, or *OFF*, the processor sets the bit to 0, or *OFF*. As the EXAMINE OFF instruction is looking for an *OFF* condition, a bit set to 0, or *OFF*, is a true condition and there would be logic continuity, so power can flow through the contact. If the input device had been closed, or *ON*, the bit would be set to 1, or *ON*. The EXAMINE OFF instruction can only be logically true when an *OFF* condition exists. Any bit set to 1 is viewed as an *ON* condition which makes an EXAMINE OFF (*N.C.*) contact false and no power can flow.

To further reinforce the EXAMINE ON and EXAMINE OFF instruction concepts, a look at a standard Stop/Start station may be helpful. Figure 8–7a shows a standard Stop/Start circuit with overload contacts using a standard ladder diagram. Figure 8–7b shows the equivalent circuit program for a PLC.

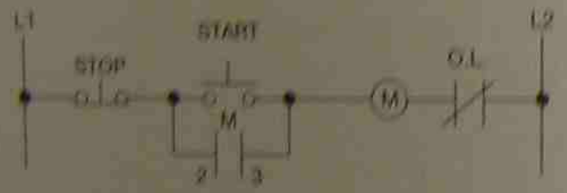


Figure 8–7a Standard STOP/START Ladder

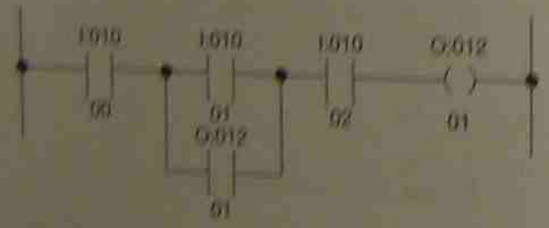


Figure 8–7b Equivalent STOP/START Circuit Programmed with a PLC

Note: *Addresses shown are PLC-5 format. An "I" preceding a word and bit number indicates an input whereas an "O" preceding a word and bit address indicates an output. Since the output must be the last item programmed on a Rung, the overload contacts (O.L./I:010/02) are programmed ahead of the motor.*

Once the input devices are wired to the input module(s) as shown in Figure 8–7c, and the PLC system is "powered up", or turned *ON*, the processor scans the inputs and sets the corresponding bits to 1 or 0 depending on the status of the real-world input devices. If an input is open, the corresponding bit is set to 0, or *OFF*, whereas any bit that represents a closed device will be set to 1, or *ON*.

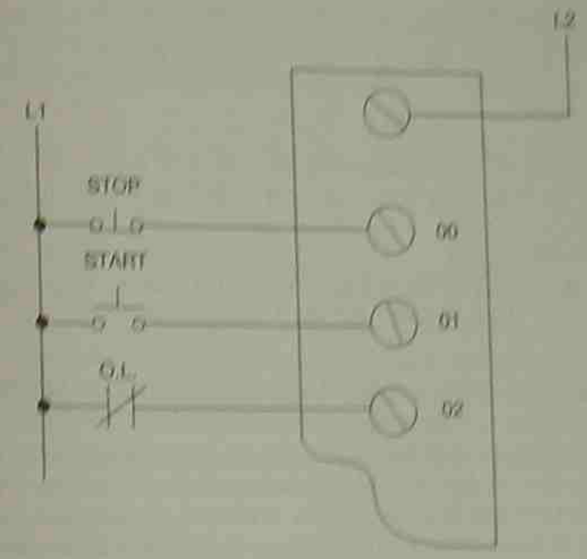


Figure 8–7c Input Devices Connected to an Input Module

In Figure 8–7c, the STOP button (I:010/00) and the O.L. contact (I:010/02) are programmed as normally opened contacts (EXAMINE ON). Because the STOP button and overload contacts are actually closed, bits 00 and 02 of word 010 are set to 1, or *ON*. Figure 8–8a shows the bit status of input image word 010 with the processor in the *RUN* mode.

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OUTPUT IMAGE WORD 012

Figure 8–8a Bit Status for Input Image Word 010 and Output Image Word 012 With the Processor in RUN Mode

With the STOP button and the O.L. contacts (bits 00 and 02), set to 1, or *ON*, we need only press the START button to complete the circuit. When the START button (I:010/01) is depressed, bit 01 is set to 1 (*ON*) during the next processor scan (shown in Figure 8–8b), and the circuit logic to the output is complete.

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OUTPUT IMAGE WORD 012

Figure 8–8b Bit Status for Input Image Word 010 and Output Image Word 012 While START Button is Being Depressed

With the circuit logic now complete (true), the processor sets bit 01 of output word 012 to 1, or *ON* (Figure 8–8c), and motor O:012/01 is *energized* and held energized by holding contacts O:012/01. The holding contacts O:012/01 have the same address as the motor (O:012/01) because they both reference the same bit (01) of word 012 in the output register. The holding contacts do not actually exist as hardwired contacts, but are used to maintain the logic path for the circuit. The EXAMINE ON instruction with the address O:012/01 is the equivalent of holding contacts, and when the motor is energized (turned *ON*), bit 01 of word 012 set to 1, or *ON*, and an alternate logic path is now complete.

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

OUTPUT IMAGE WORD 012

Figure 8–8c Bit Status for Input Image Word 010 and Output Image Word 012 After Output O:010/01 is Energized and the START Button is Released

When the STOP button (input I:010/00) is depressed, the processor clears bit 00 to 0, and the circuit logic is broken, or goes FALSE. Bit 01 is cleared to 0, and the real-world output device connected to terminal 01 of the output module drops out (turns *OFF*). Words 010 and 012 in the I/O register now appear as shown in Figure 8–8d, with only bit 02, the overload contact, set to 1.

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OUTPUT IMAGE WORD 012

Figure 8–8d Bit Status for Input Image Word 010 and Output Image Word 012 With the STOP Button Depressed

When the STOP button is released or closed again, bit 00 is again set to 1, or *ON* (Figure 8–8e). The output (O:012/01) is not energized, however, as the START button (bit 01) is 0 (*OFF*), and the holding contacts (bit 01 of word 012) are cleared to 0 (*OFF*).

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OUTPUT IMAGE WORD 012

Figure 8–8e Bit Status for Input Image Word 010 and Output Image Word 012 With STOP Button Released

Another way to look at the normally open and normally closed symbols used for programming the PLC is called the relay analogy.

For the sake of discussion, imagine that each input device is connected to an invisible control relay inside the input module, and that each control relay has one normally open and one normally closed contact as shown in Figure 8–9.

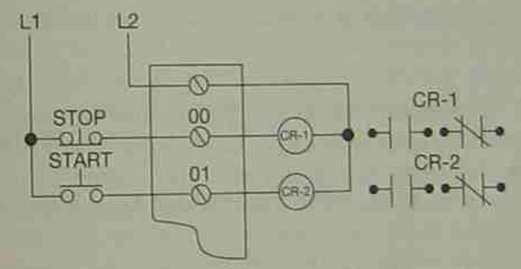


Figure 8–9 Imaginary Control Relays Wired to an Input Module

Connect one lamp to the normally open contacts, and another lamp to the normally closed contacts of CR-1, as shown in Figure 8–10.

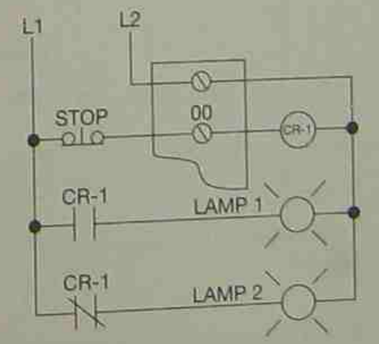


Figure 8–10 Lamps Wired to N.O. and N.C. CR-1 Contacts

When power is applied to L1 and L2, CR-1 energizes through the normally closed contacts of the STOP button. With CR-1 energized, the normally open contacts of CR-1 close, and lamp 1 lights as indicated in Figure 8–11. The normally closed contacts of CR-1 are now open, so lamp 2 *cannot* light.

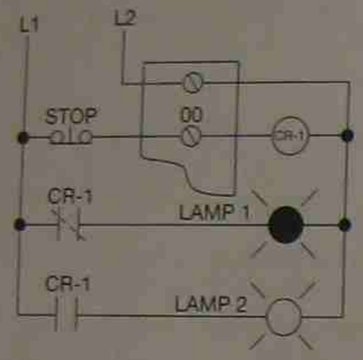


Figure 8–11 Lamp 1 Lights with Power Applied to L1 and L2

Wired in this manner, normally open CR-1 contacts controlled by a normally closed pushbutton will close or *conduct* when power is applied to the circuit. Likewise, normally open contacts *programmed* to represent a normally closed pushbutton will conduct when power is applied to the PLC.

A normally open START button connected to an input terminal and an invisible or imaginary control relay (shown in Figure 8–12) would not light Lamp 1 until the START button was depressed, and would only stay lit as long as the button was held down. Lamp 2 would light as soon as power was applied, but would go out when the START button was depressed and CR-2 energized (Figure 8–13).

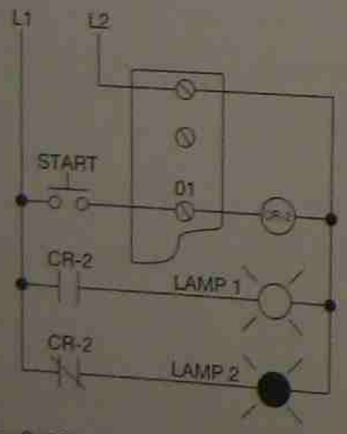


Figure 8–12 Power Applied—START Button Not Depressed

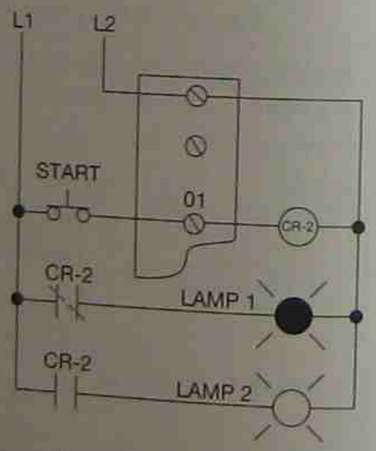


Figure 8–13 Power Applied—START Button Depressed

The rules for contacts that represent real-world input devices are shown in Figures 8–14a and 8–14b.

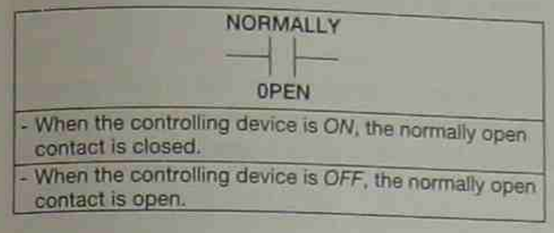


Figure 8–14a Normally Open Contact of an External Input

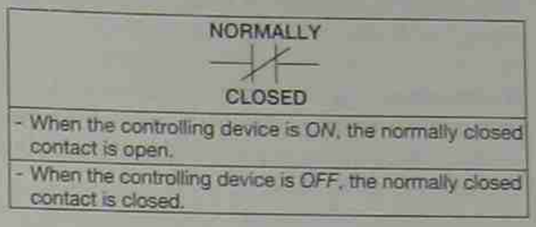


Figure 8–14b Normally Closed Contact of an External Input

There are no invisible control relays in the input modules, and there are also no symbols on the programming device for STOP buttons, START buttons, limit switches, and the like. As long as relay contact symbols must be used in place of regular input symbols, the relay analogy is an easy way to explain why normally open contacts are programmed to represent normally closed input devices.

It does not matter which approach you use to understand the logic behind the way that PLCs are programmed, as long as you do understand it. The author believes that the EXAMINE ON and EXAMINE OFF approach is the easiest and clearest way to look at programming. The conversion of ladder diagram to PLC program will be quite simple once you clearly understand the concept of EXAMINE ON and EXAMINE OFF.

Clarifying EXAMINE ON and EXAMINE OFF

From the previous discussion and examples, it appears that all input devices, whether they are normally open or normally closed, are programmed as N.O. contacts to achieve the desired results in the ladder diagram. For many circuit applications this is true, and a big advantage of this programming technique is the ability to turn single-pole input devices into double-, three-, or four-pole devices in the circuit. Figure 8–15 shows a double-pole pressure switch (PS-1) controlling two outputs: motor 1 and motor 2.

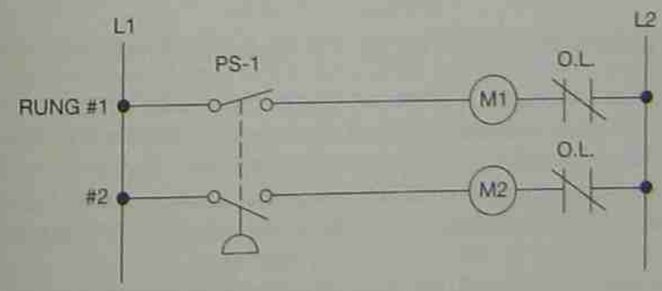


Figure 8–15 Double Circuit Pressure Switch

When power is applied to the circuit, motor 1 will start through the N.C. contacts of PS-1 in Rung 1. Motor 2 cannot start, however, due to the N.O. contacts of PS-1 in Rung 2. When PS-1 is actuated, the N.C. contacts in Rung 1 will open and motor 1 will go *OFF*, while the N.O. contacts in Rung 2 will close, turning motor 2 *ON*.

By programming the same circuit on a PLC, the necessity of buying a double circuit pressure switch is eliminated. One N.O. and 1 N.C. contact having the same address is used (see Figure 8–16a). The address is actually the address of a discrete N.C. single circuit pressure switch (illustrated in Figure 8–16b).

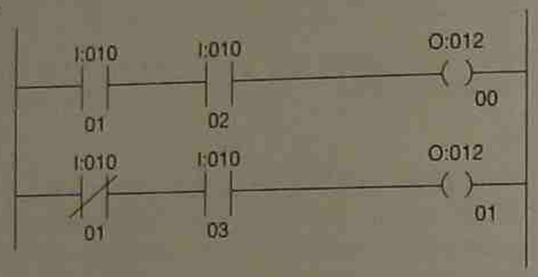


Figure 8–16a Double Circuit Pressure Switch Circuit Programmed for a Typical PLC

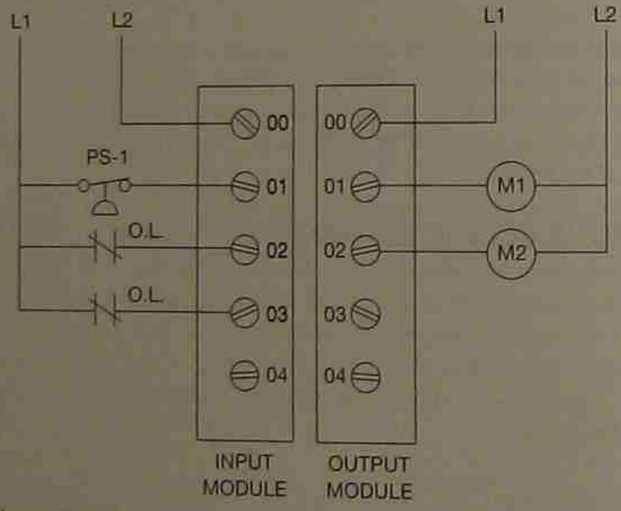


Figure 8–16b Actual Wiring of a Single-Pole Pressure Switch

When the processor is placed in the RUN mode, it examines all N.O. contacts for an *ON* condition, or EXAMINE ON. As PS-1 and both overload contacts are closed (*ON*), bits 01, 02, and 03 of input word 010 are set to 1, making those portions of the ladder diagram true. When the processor EXAMINES OFF the N.C. contact (I:010/01), it sees that bit 01 is set to 1 so that this part of the ladder diagram is false. Motor 1, address O:012/00, is *ON* and motor 2, address O: 012/01, is *OFF*.

When the pressure switch is actuated, the processor continues scanning the user program and examines all N.O. contacts for *ON*, and all programmed N.C. contacts for *OFF*. Since the pressure

switch (PS-1) is now actuated, the N.C. contacts are open, and the N.O. contacts (I:010/01) go false, turning *OFF* motor 1 (O:012/00), whereas the N.C. contacts (I:010/01) go true, and turn motor 2 (O:012/01) *ON*.

Even though only double, three, and four poles were mentioned, there is no limit (except user memory size) to the number of times an input device can be addressed and used in a programmed circuit. This programming technique allows for six-pole, seven-pole, eight-pole, and so on, devices to be programmed using only a single-pole discrete device.

There are many more applications and circuits that normally require two- or three-pole devices that now only require single-pole devices when programmed for a PLC. Examples are double-pole limit switches for forward and reversing circuits, and double-pole pressure switches for duplex controllers.

When programming contacts (N.O. or N.C.) which are controlled by outputs, the familiar standard relay logic is used. Figure 8–17a shows a standard STOP/START station with pilot lights. Lamp 1 (green) indicates power is available, and lamp 2 (red) indicates the circuit is activated. Figure 8–17b shows how the circuit is programmed.

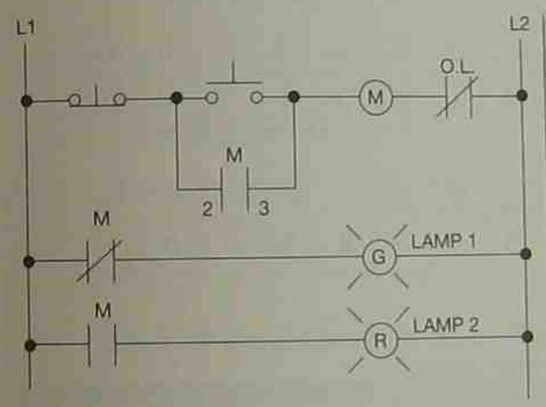


Figure 8–17a Ladder Diagram for STOP/START Station with Indicator Lamps

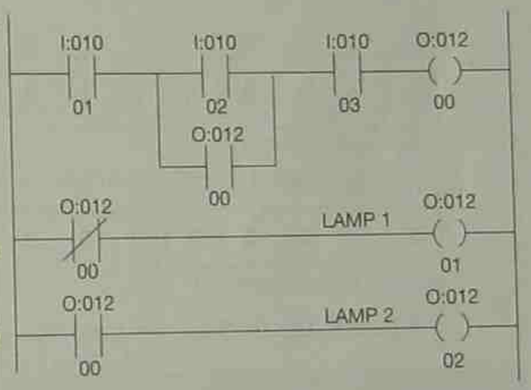


Figure 8–17b Programmed Circuit for STOP/START Station with Indicator Lamps

When the processor is placed in the RUN mode, lamp 1 lights due to the N.C. contacts O:012/00. When the START button is depressed, output O:012/00 energizes, N.C. contacts O:012/00 open (go false), and lamp 1 (O:012/01) goes out. N.O. contacts O:012/00 close (go true), lamp 2 (O:012/02) turns *ON*, and the holding contacts (O:012/00) go true to complete the circuit logic.

Notice that the output, holding contacts, N.C. contact, and N.O. contact for the lamps all have the same address (O:012/00). The address refers to bit 00 of word 012, and this same bit (00) is referenced four times in the ladder diagram. The EXAMINE OFF, or N.C. contact, is logically true when bit 00 is cleared to 0 (*OFF*), whereas the EXAMINE ON, or N.O. contacts, are not logically true until bit 00 is set to 1 (*ON*).

Figure 8–18a shows the bit status for the circuit with power applied; Figure 8–18b, with the START button depressed; Figure 8-18c, with the START button released; Figure 8-18d, with the STOP button depressed; and Figure 8-18e, after STOP button is released.

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

OUTPUT IMAGE WORD 012

Figure 8–18a Bit Status for Input Image Word 010 and Output Image Word 012 With Power Applied

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

OUTPUT IMAGE WORD 012

Figure 8–18b Bit Status for Input Image Word 010 and Output Image Word 012 With START Button Depressed

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

OUTPUT IMAGE WORD 012

Figure 8–18c Bit Status for Input Image Word 010 and Output Image Word 012 With START Button Released

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

OUTPUT IMAGE WORD 012

Figure 8–18d Bit Status for Input Image Word 010 and Output Image Word 012 With STOP Button Depressed

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

INPUT IMAGE WORD 010

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

OUTPUT IMAGE WORD 012

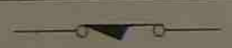
Figure 8–18e Bit Status for Input Image Word 010 and Output Image Word 012 With STOP Button Released

Chapter Summary

The relay logic used for programming the PLC at first seems to be in conflict with standard ladder logic. But once the concepts of EXAMINE ON and EXAMINE OFF are understood, the logic process is easy to understand. An EXAMINE ON instruction looks for an *ON* condition, and will be logically true when an *ON* condition (a 1) is found. The EXAMINE OFF instruction will be logically true when an *OFF* condition (a 0) is found. Looking at the actual status of the bits within individual I/O words is another way to help understand how the processor logic works. Using the XIC or examine if closed (EXAMINE ON) and the XIO or examine if open (EXAMINE OFF) approach may be helpful. Others find that the relay analogy approach to understanding the normally open (EXAMINE ON) and normally closed (EXAMINE OFF) symbols used for programming is better. Another approach is to accept the fact that an N.C. STOP button, or a similar closed input device must be programmed using an N.O. contact symbol, and just have the philosophy that logic is relative to application and don't worry about it.

Review Questions

1. Briefly describe the action of the EXAMINE ON instruction.
2. When a normally open (N.O.) limit switch is wired to an input module, and programmed using a N.O. contact symbol (EXAMINE ON), the instruction will be true when (check all correct answers):
   a. power is applied and the key switch is in the RUN position.
   b. the limit switch is closed.
   c. as long as the limit switch is open
   d. never
3. If the normally open limit switch in Question 2 is programmed using a N.C. contact symbol (EXAMINE OFF), the instruction will be true when (check all correct answers):
   a. power is applied and the key switch is in the RUN position.
   b. the limit switch is closed.
   c. as long as the limit switch is open
   d. never
4. Briefly describe the action of the EXAMINE OFF instruction.
5. Indicate the logic, (T [True] or F [False]) for the following contacts:

CONDITION OF INPUT DEVICE	PROGRAM INSTRUCTION	LOGIC TRUE-FALSE
		T F
		T F
		T F
		T F

6. The XIO instruction is the same as:
   a. EXAMINE ON
   b. EXAMINE OFF
7. The XIC instruction is the same as:
   a. EXAMINE ON
   b. EXAMINE OFF

CHAPTER 9 Programming a PLC

Objectives

After completing this chapter you should have the knowledge to:

- Explain the term *On Line Programming*.
- Describe basic programming techniques.
- Define the term *mnemonic* and give examples of mnemonic names.
- Describe the *Force On* and *Force Off* features, and the hazards that could be associated with both.

The electrical symbol keys on dedicated programmers, or programming software, that represent N.O. contacts, N.C. contacts, branch circuit start and end, coils/outputs, timers, counters, and so forth, use OP codes (operating codes) to tell the processor what to do.

Note: *The terms "branch circuit start" and "branch end" refer to the start and end of circuit junctions where contacts are connected in parallel.*

With a 16-bit word of memory, four bits are usually used for an OP code to tell the processor *what* to do, and the remaining 12 bits are the address and tell the processor *where* to do it. When the key for a normally open contact symbol (Examine ON) is pushed and then followed by an address, one word of user memory is used. Four bits tell the processor to treat this as an N.O. contact, or a request to Examine ON. The next twelve bits hold the address or location of the input, output, or internal location with which the contact is associated.

For most PLC systems, each N.O. and N.C. contact, branch start and end, and coil (output) instruction requires one word of user memory, whereas timers and counters require from two to five, depending on the PLC. In reality, when a contact is entered and addressed, one full word of user memory is used. Most programmers or software programs display the total number of memory words used at the completion of developing the program. The total also includes any words of memory, if any, that the processor used for internal purposes.

Since the user program requires one full word for each contact or coil, and two or more words for each timer and/or counter programmed, it is not unusual for the user memory to use more of the total memory words than the storage memory, since the storage memory only uses 1 bit of a 16-bit word to store the status of the various input and output devices.

Programmers and programming software can create, modify, monitor, and load programs into user memory, and can also make changes to the program while the processor and driven equipment is running. This feature is often referred to as On Line Programming. Changing the program while the processor is running must *only* be done by persons with a complete understanding of the circuit operation and the process or driven equipment. To prevent unauthorized On Line programming, a key switch is provided, either on the programming device or on the processor, to restrict the access to a monitor only mode, or to Off Line and monitor only mode. When programming using a personal computer and specialized software, a **password** may be used to limit the access to the processor program. Passwords act like a key switch. If the wrong password is entered when requested, the user is denied further access to the program.

Off Line programming, which means that the program is being developed off line (without being connected to the process or driven equipment) is the most common method of programming, and, of course, the safest. Since few programs are ever created without mistakes, it is always best to create the program Off Line. Once the program is complete, it should be tested while still in the Off Line mode. After testing and verifying the program (circuit) in Off Line, the PLC can be put in the On Line mode for final verification, testing, and operation.

The function or functions of the processor that can be locked out with the key switch or by using passwords are fairly standard, but vary from PLC to PLC.

Most PLCs are designed so that any input or output contact (N.O. or N.C.) or output coil can be **FORCED ON** or **OFF**. This feature (and troubleshooting aid) allows the operator to FORCE ON, or make a contact or coil go *ON* regardless of actual status or circuit logic of the input or output device. Similarly, contacts and coils can be FORCED OFF, or turned *OFF*, regardless of the actual device status or circuit logic.

Caution: The FORCE ON and FORCE OFF feature should *never* be used except by personnel who completely understand the circuit *and* the process machinery or driven equipment. An understanding of the potential effect that forcing a given contact or coil has on machine operation is essential if hazardous and/or destructive operation is to be avoided.

Earlier PLCs were programmed using dedicated programming terminals like the T3 terminal shown in Figure 9–1a, 9–1b, and 9–1c. Although these terminals were rugged and made for the industrial environment, they were very expensive and could only by used for programming and nothing else. Today, most programming is done using cheaper personal computers and software designed specifically for programming the various PLCs on the market.

Figure 9–1a Sealed Touchpad Keyboard *(Courtesy of Allen-Bradley)*

Figure 9–1b Keyboard Overlays
(Courtesy of Allen-Bradley)

Figure 9–1c Installing an Overlay on the Keyboard
(Courtesy of Allen-Bradley)

PROGRAMMING WITH A COMPUTER

Many companies have developed software for programming PLCs. For purposes of illustrating how the software works, and the relative ease of programming, the author has selected the RSLogix software created by Rockwell Software for programming the Allen-Bradley family of programmable controllers. This software, in various versions, can be used to program the PLC-5, SLC 500, or the MicroLogix family of processors. The software is Windows® based and very user friendly. For this programming example, we will use RSLogix 500 to program an SLC 5/03 processor mounted in a 4-slot chassis as shown in Figure 9–2a.

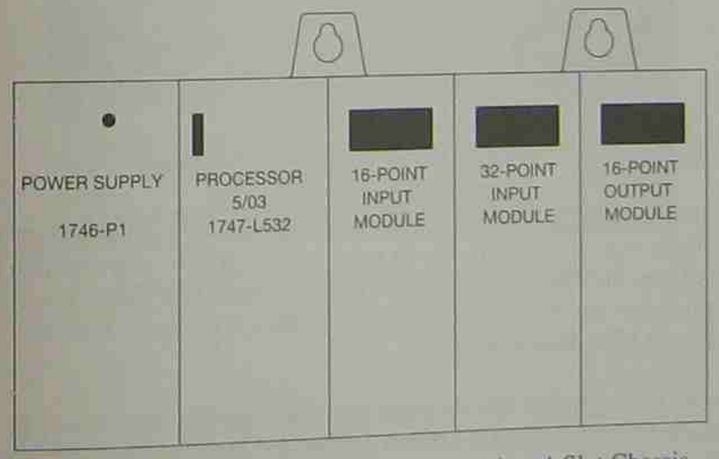


Figure 9–2a SLC 5/03 Processor Mounted in a 4-Slot Chassis

The circuit to be programmed is a simple STOP/START circuit that uses a START button, a STOP button, and holding contacts to control an exhaust fan. The ladder diagram of the circuit is shown in Figure 9–2b.

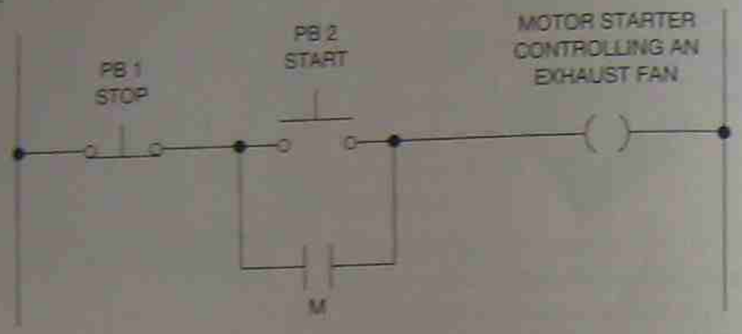


Figure 9–2b Basic STOP/START Circuit

To start the RSLogix software, double click on the RSLogix icon for the PLC family you are using. The opening screen for the RSLogix 500 software is shown in Figure 9–3.

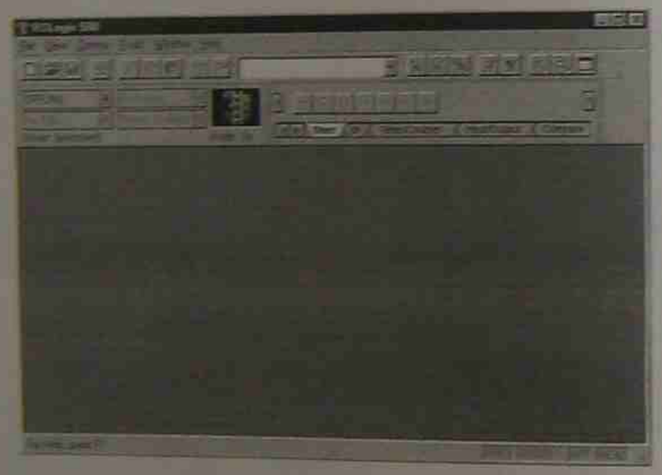


Figure 9–3 RSLogix Opening Screen

Using the computer mouse, left click on *File* and then click on *New*. The window that appears, titled *Select Processor Type*, lists the processors that the RSLogix 500 software can program. Scroll down the list until you find the processor you are using. Figure 9–4 shows the window. Note that an SLC 5\03 processor has been highlighted. Note also that below the 5\03 are listings for MicroLogix 1000 and 1500 processors.

Once the processor has been selected, the processor name needs to be entered. In Figure 9–4, the box where the name of the processor is entered, Processor Name:, shows the name as *UNTITLED*. The name given the processor is typically the name of the process that the processor controls.

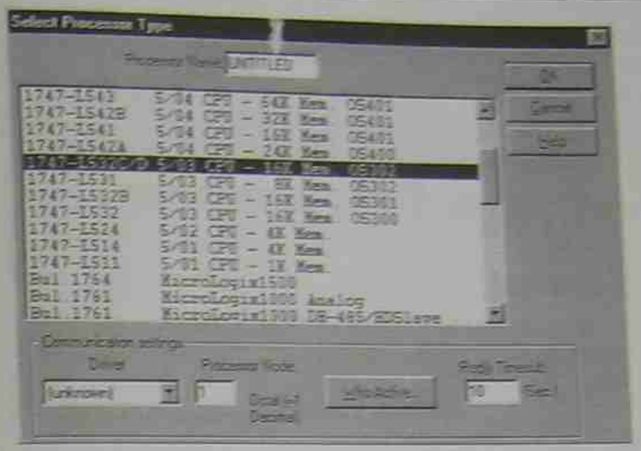


Figure 9–4 Select Processor Type Screen

Examples are Lumber Stacker 1, Log Sorter, Palletizing Unit, etc. For this example, the processor name will be TEST. Figure 9–5 shows the window after the processor name has been entered.

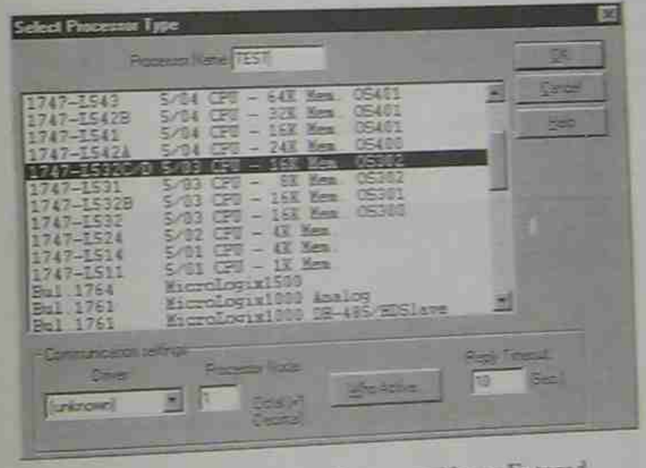


Figure 9–5 Screen with Processor Name Entered

The next step is to single click on the *OK* button which causes the software to enter the information for the processor that has been selected (SLC 5/03). Figure 9–6 shows the new screen.

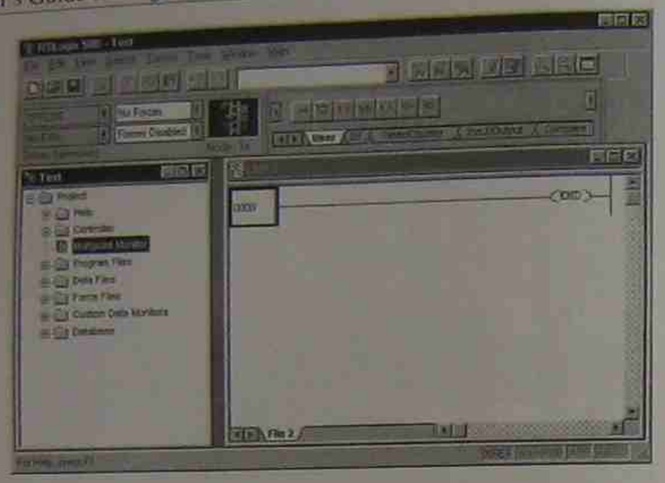


Figure 9–6 Screen After Processor Information has been Stored

The left side of the screen shows the *Project Tree* while the right side of the screen is the *Programming Area*. Either area can be increased in size, minimized, or closed by left clicking the mouse on the appropriate symbol.

Under the project tree, the main folders are Help, Controller, Multipoint Monitor, Program Files, Data Files, Force Files, Custom Data Monitors, and Data Base. Not all of these folders will be discussed since the intent of this section is merely to illustrate the relative ease of programming using the RSLogix software, not a definitive programming guide.

Double clicking on the *Controller* folder opens the folder and produces the screen shown in Figure 9–7.

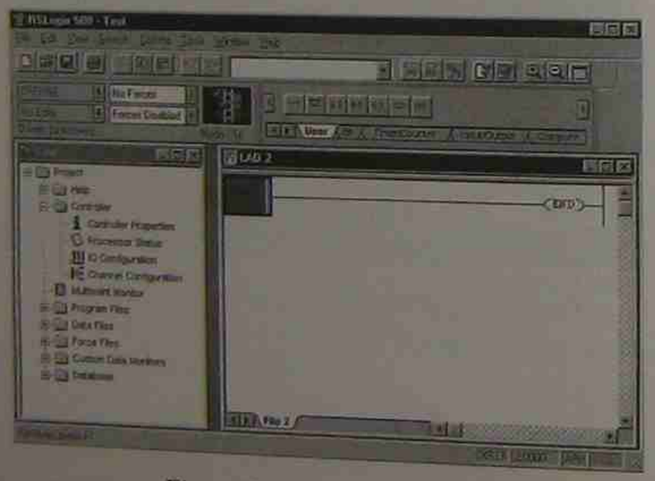


Figure 9–7 Controller Screen

Double clicking on *Controller Properties* produces the screen shown in Figure 9–8. From this screen, we can see the number of program files and data files, and also determine the memory used and the memory left. In the screen shown there is an (*) in both the Memory Used and Memory Left areas, because no program has been developed yet and no memory has been used.

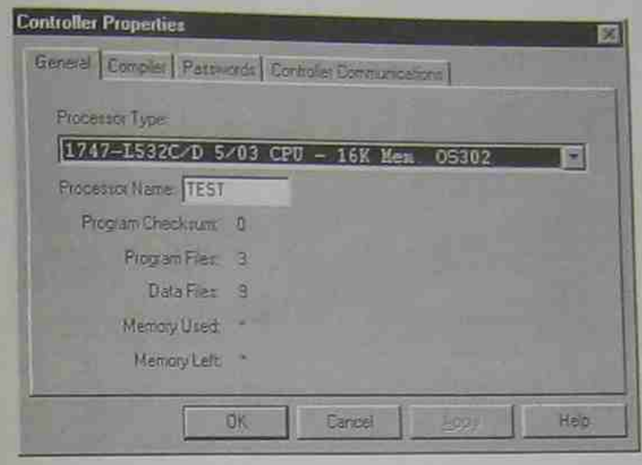


Figure 9–8 Controller Properties Screen

Closing out *Controller Properties* and double clicking on the *IO Configuration* symbol produces the screen shown in Figure 9–9. From this screen, we can configure the I/O. As stated earlier, the SLC 5/03 processor is installed in a 4-slot chassis. From this window select either a 4-slot, 7-slot, 10-slot, or 13-slot chassis by clicking the button on Rack 1.

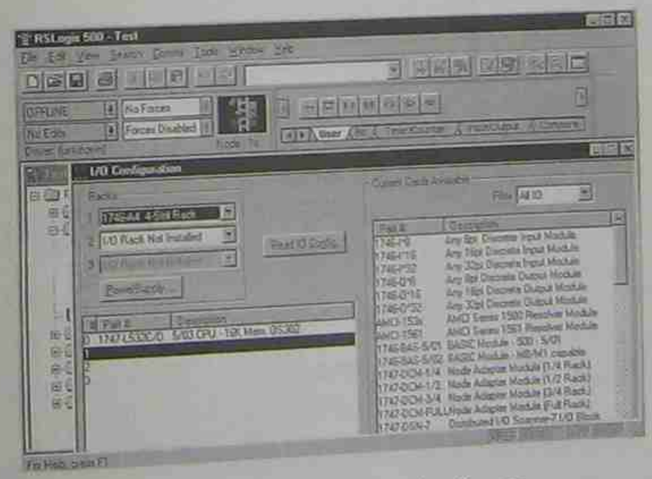


Figure 9–9 IO Configuration Screen

Notice that in Figure 9–9 the 5/03 processor is shown installed in slot 0. The processor will always be installed in slot 0 when using modular I/O. To enter the correct I/O module that is installed in slot 1 of our 4-slot chassis, find the module on the list to the right and double click. Figure 9–10 shows the window after double clicking on 1746-I*16, which is a 16-point discrete input module installed in slot 1 of the 4-slot chassis (Figure 9–2a).

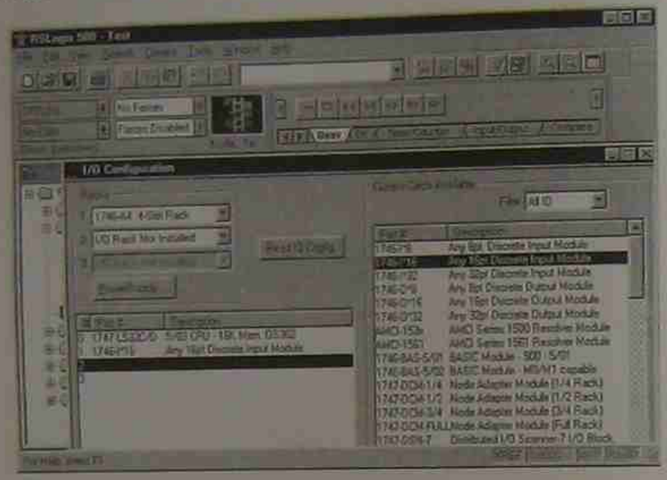


Figure 9–10 I/O Configuration Window With 16-Point Input Module Selected for Slot 1

To complete the I/O Configuration, select a 32-point input module (1746-I*32) for slot 2 and a 16-point output module (1746-O*16) for slot 3. The completed I/O Configuration is shown in Figure 9–11. If the processor was in the On Line mode, the software would look at the I/O modules installed in the chassis and automatically enter the appropriate I/O module type and number into this window.

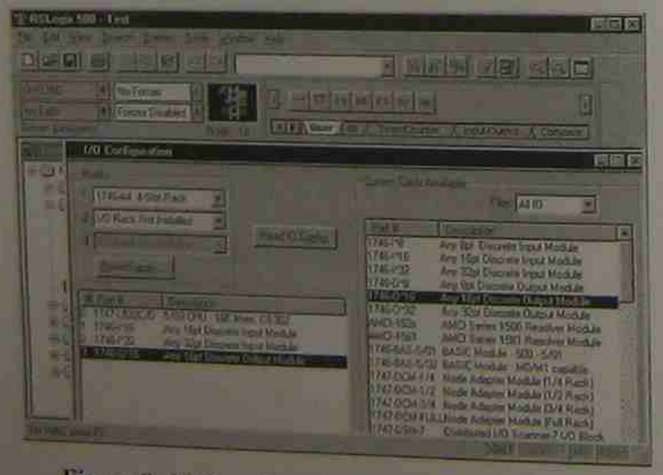


Figure 9–11 Completed IO Configuration Window

The software has the ability to indicate the minimum size of power supply that should be used based on the number and type of I/O that have been selected. To determine the correct power supply to use, single click on the *Power Supply* button. The *Power Supply Loading* window is shown in Figure 9–12. This window shows that our SLC 5/03 system consists of 1 Rack, and the minimum power supply recommended is a 1746-P1.

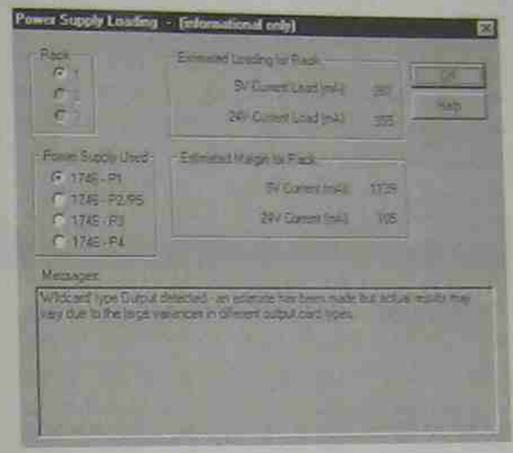


Figure 9–12 Power Supply Loading Window

This window estimates the loading of the rack. In this illustration, the estimated 5-volt current load in mA (milliamps) is 861 and the 24-volt current load is estimated at 355 mA. This window also estimates the margin of mA load that is provided by a 1746-P1 power supply beyond the mA load for the mix of I/O currently installed. The estimate for the 5-volt current is 1139 mA. We could add an additional 1139 mA of load before the P1 power supply would be at full capacity. The 24-volt current load margin is given a 105 mA. Note the message at the bottom of the window that says:

> "Wildcard" type output detected - and estimate has been made but actual results may vary due to the large variances in different output cards types.

This message informs the programmer that the current values given are only estimates and the type of discrete output cards used will determine the actual mA load. Common sense dictates that if the estimated mA margin for the rack is low, the next size power supply should be used.

The RSLogix software gives an ***OVERLOAD—Use a Larger Power Supply*** warning when the mix of I/O requires a larger power supply than has been selected. To determine the correct power supply, once an overload warning has been displayed, click on the next larger power supply and see if the warning disappears. If not, click on the next larger power supply listed. Continue to click on the next larger power supply until the OVERLOAD display disappears.

After closing the Controller Properties screen, open the *Data Files* folder by double clicking on the folder. The contents of this folder are shown in Figure 9–13.

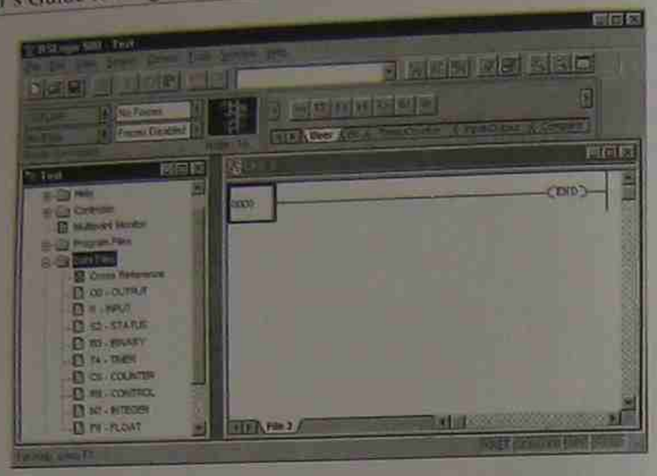


Figure 9–13 Data File Folder Contents

The Data File folder allows the user to determine the status of I/O files, as well as the status file (S2), binary file (B3), timer file (T4), counter file (C5), control file (R6), integer file (N7), and the floating point file (F8).

When the processor is in the *On Line Mode*, the actual status of the input and output devices, 1 or 0, is determined by looking at the bit status in either the Input or Output file. Double clicking on I1-*INPUT* produces the *Data File I1—(bin) Input* window shown in Figure 9–14.

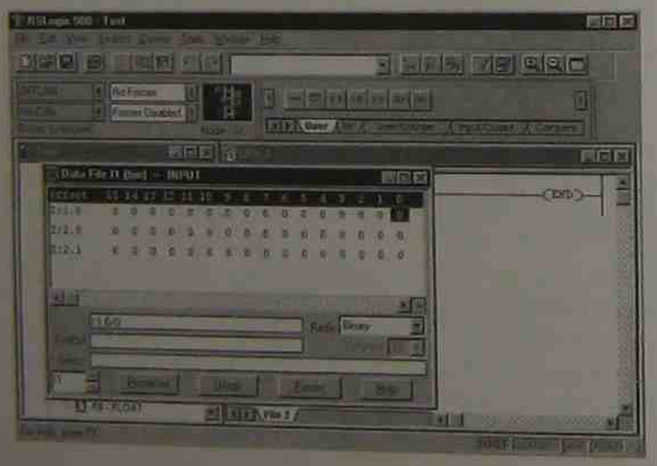


Figure 9–14 Data File Window

This window shows the status of the first three words of the input file. The first address I:1.0 indicates this is an input module installed in slot 1. If we refer back to Figure 9–2a, the 16-point input module was indeed installed in slot 1. Also, in Figure 9–11, an input module 1746-I*16 was shown installed in slot 1 of the 4-slot chassis. The individual output devices are represented by bits 0 through 15. Again note that the bit addresses are in decimal format, not in octal, like the Allen-Bradley PLC-2 and PLC-5 families of PLCs.

The next two addresses are I:2.0 and I:2.1. These addresses indicate that an input module is installed in slot 2 but requires two 16-bit words of memory. If we refer back to Figure 9–2a again, a 32-point input module was installed in slot 2. Figure 9–11 showed that input module 1746-I*32 was installed in slot 2 of the 4-slot chassis. Because this input device has 32 points, or terminals, it requires two words of memory as shown in Figure 9–14.

From this window, each bit (input device) can be assigned a *Symbol* and a *Description*. The symbol tells what the device is and the description explains what it does. The symbol and the description are entered by typing the information into the appropriate box. For the example in Figure 9–15, for address I:1.0/0, the symbol was identified as a PB 1 (pushbutton 1), and the description as *Stop*. This is the address that will be used for the STOP button in the STOP/START circuit to be programmed. The information for each connected input device can be entered in this same manner. Information also can be assigned to input and output devices as the program is being written. If this is the case, the symbol type and description will be shown automatically when a Data File window is opened.

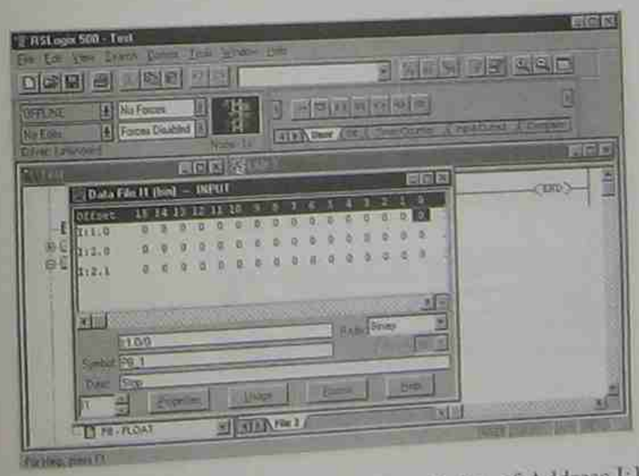


Figure 9–15 Input Data File with Symbol and Description of Address I:1.0/0 Entered

If we close the Data File Input screen and double click on O0-Output, the Output Data File will open as shown in Figure 9–16.

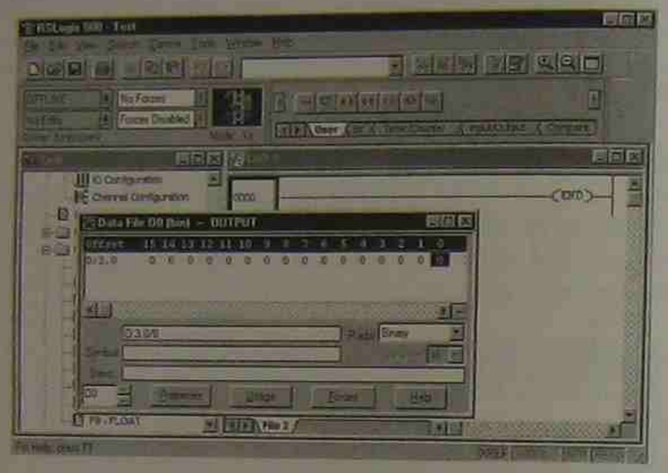


Figure 9–16 Output Data File

From this screen, we can add the symbol and description for each of the 16 outputs represented by bit 0 through 15. Figure 9–17 shows the screen after the symbol and description for output address O:3.0/0 has been typed in. This is the address that will be used for the motor starter that controls the exhaust fan in our STOP/START circuit.

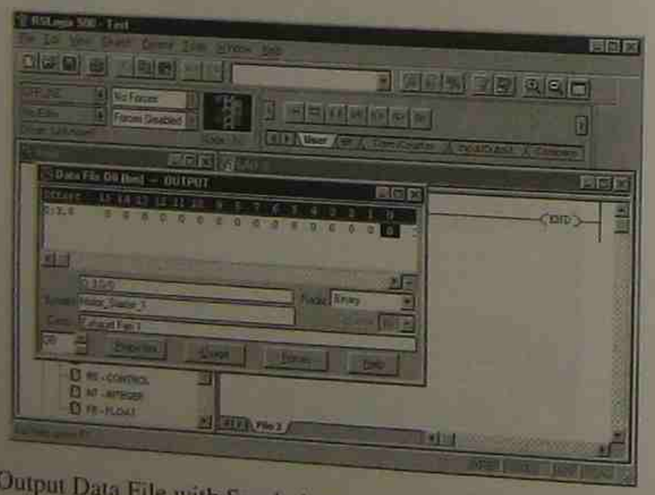


Figure 9–17 Output Data File with Symbol and Description of Address O:3.0/0 Entered

As with the Input Data File, each output device can be given a symbol and a description from this screen.

Again, this brief discussion of the program tree and the information that can be obtained is intended to show the power of the RSLogix software and is not a complete guide to the software.

By closing the Output Data Table screen and maximizing the programming side of the display, the screen now looks like the one in Figure 9–18.

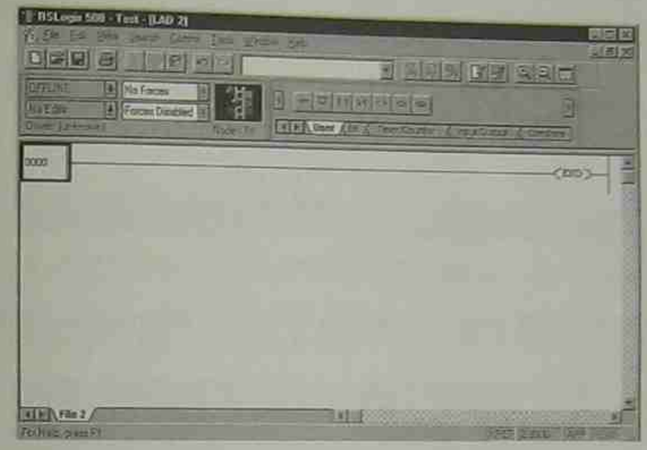


Figure 9–18 Programming Screen

The box at the upper left-hand side of the programming screen has four zeros (0000) inside the box. The four zeros indicate Rung 0000. To start to develop a program, single click on the *New Rung* button above the program area. The *New Rung* button is the first button and uses this symbol: --|—|. Figure 9–19 shows the screen with the new rung added.

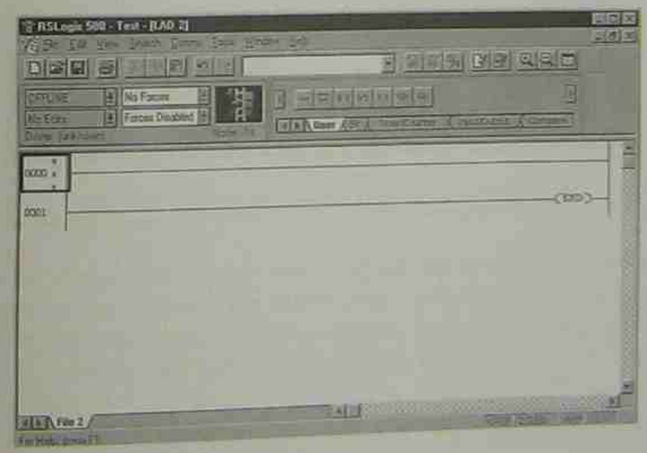


Figure 9–19 Program Area with Rung Added

As shown in Figure 9–19, two rungs now show on the screen. The first rung is 0000; the second rung is 0001. Note that there are three small letter "e"s at Rung 0000. The "e"s indicate that this rung is being edited.

Figure 9–20 shows the simple STOP/START circuit without overloads that will be programmed.

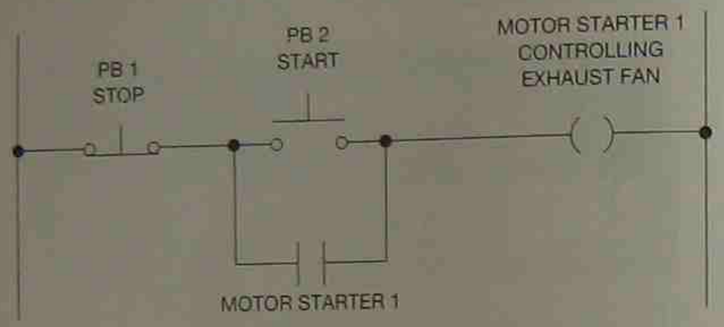


Figure 9–20 Simple STOP/START Circuit Without Overloads

The first step is to program the STOP button. This is accomplished by single clicking on the *Normally Open* contact symbol (—] [—). This is the third symbol in the group of seven symbols. This normally open contact symbol is also referred to as XIC, or examine if closed. Figure 9–21 shows the screen with the normally open contact.

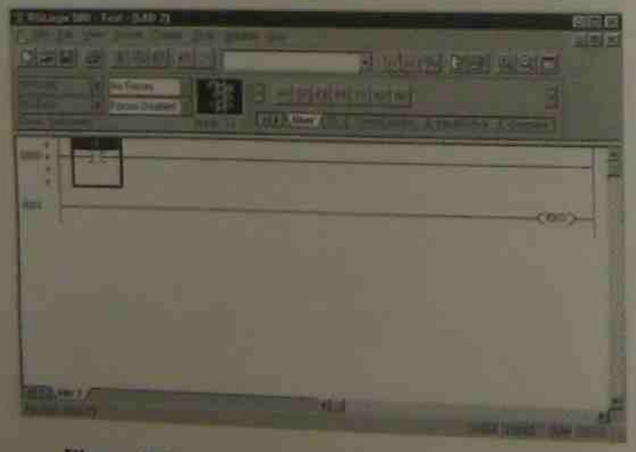


Figure 9–21 Screen with Normally Open Contact

Double click on the (?) above the contact and enter the address of the STOP button. We are using address I:1.0/0. Figure 9–22 shows the screen with the address I:1.0/0 entered above the contact.

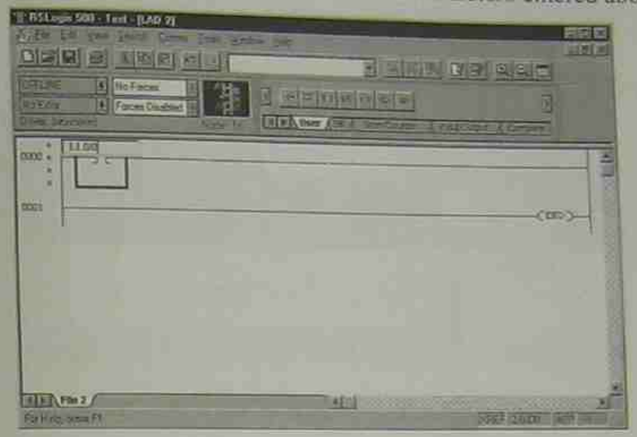


Figure 9–22 Address Entered Above the Normally Open Contact

After typing in the address, hit the *Enter* key. Figure 9–23 shows how the screen looks now.

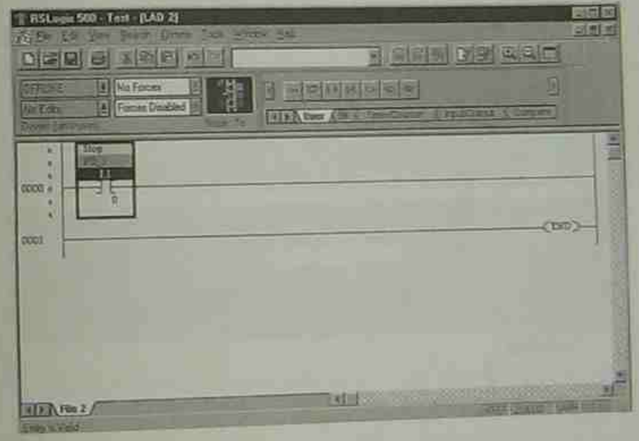


Figure 9–23 Screen with STOP Button Identified

Remember that earlier, the address I:1.0/0 was assigned the symbol PB 1 and the description of STOP. When the address was entered, the information entered for address I:1.0/0 was automatically added to the programmed normally open contacts. Notice also that the address is I:1/0. The program eliminates any unnecessary zeros. So address I:1.0/0 becomes I:1/0 (input device in slot 1 connected to terminal 0).

Note: *When entering the addresses, the semi colon (;) can be used instead of the colon (:). The software will automatically convert the ; to a : in the address.*

The next step in the programming process is to add the START button. Since the START button is in parallel with the holding contacts, single click on the BRANCH START symbol button. The button symbol is ____.

After the BRANCH START symbol is clicked, the screen now appears as shown in Figure 9–24

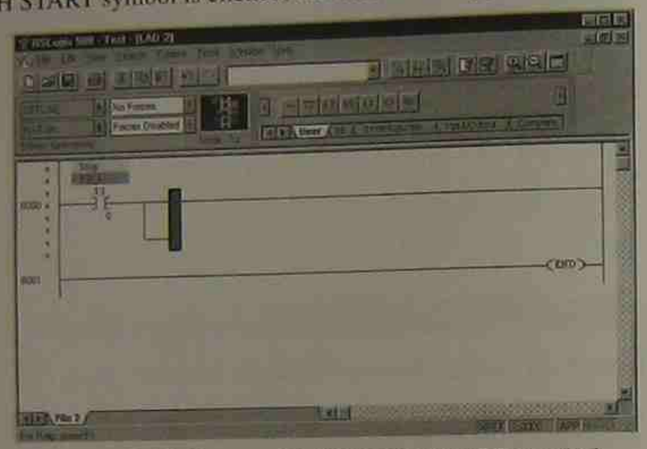


Figure 9–24 Screen with BRANCH START Symbol

Notice that a dark vertical line appears on the right side of the BRANCH START symbol. It is necessary to move this line to the upper left-hand corner of the symbol before contacts can be added. To move the line, point the arrow of the mouse at the upper left-hand corner of the BRANCH START symbol and single click. The Heavy line now moves to the upper left portion of the symbol as shown in Figure 9–25.

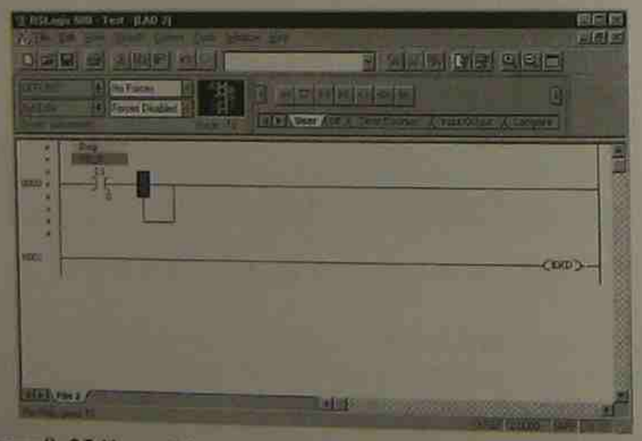


Figure 9–25 Heavy Line Moved to Upper Left-Hand Side of Symbol

Next, single click on the Normally Open symbol to add the next contact. Figure 9–26 shows the screen after the next contact has been added.

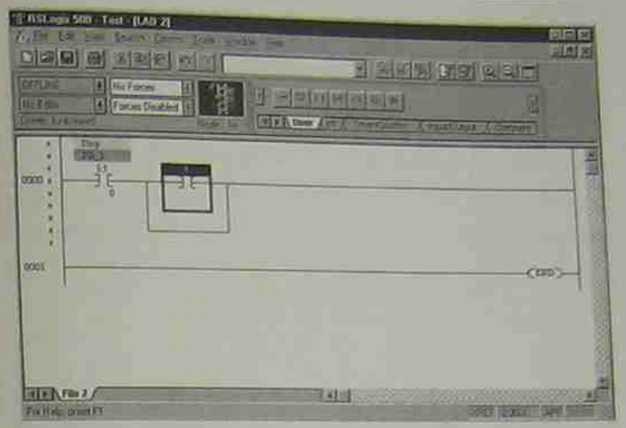


Figure 9–26 Second Contact Added

Because all unnecessary zeros will not be displayed, double click on the (?) and enter the address I:1/1 for the START button. Figure 9–27 shows the START button contacts with the address I:1/1.

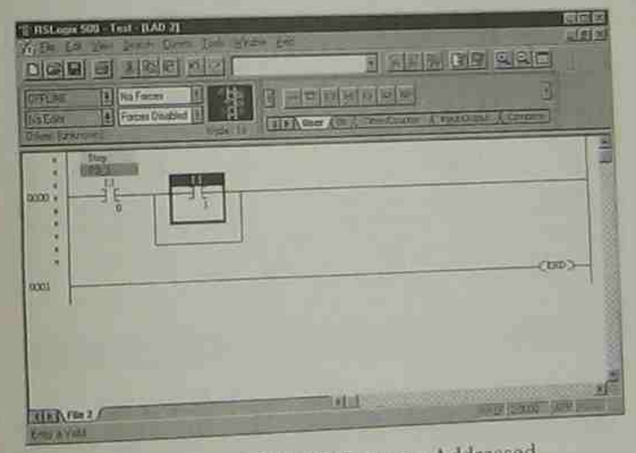


Figure 9–27 START Contacts Addressed

Notice that there is no symbol or description for the address I:1/1 as was the case for address I:1/0. The symbol and description can be entered by opening the Input Data File as was done earlier. However, another way to enter the information, is to single click the right mouse button while the arrow is pointing at the START button symbol. Left button click on *Edit Symbol*. The screen is shown in Figure 9–28.

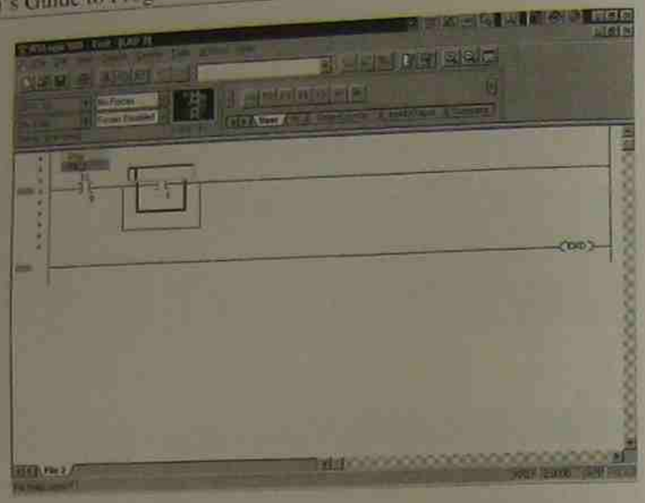


Figure 9–28 Box above START Button for Entering the Symbol

In the box above the START button, type in the assigned symbol PB 2 and press the Enter key. Figure 9–29 shows the screen with the symbol PB 2 added above the START button.

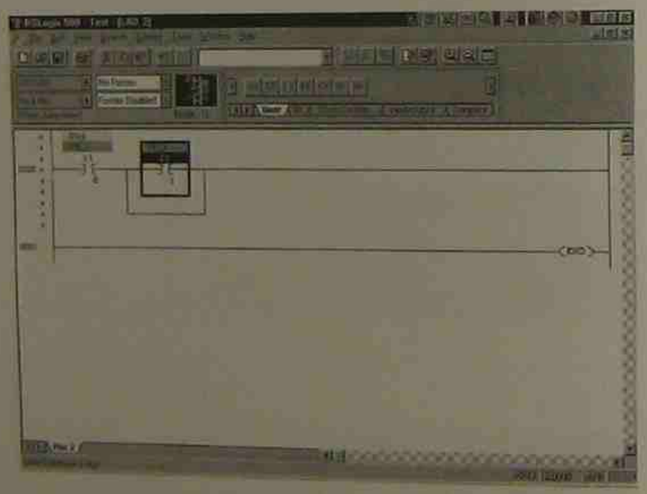


Figure 9–29 PB 2 Symbol Added to START Button

The Description of the input device (START button) can be added by placing the arrow on the START button again, single clicking the right mouse button, and then selecting *Edit Description* with a single click of the left mouse button. Type ***Start*** in the box for the description and then click on OK to enter the information. Figure 9–30 shows the completed START button.

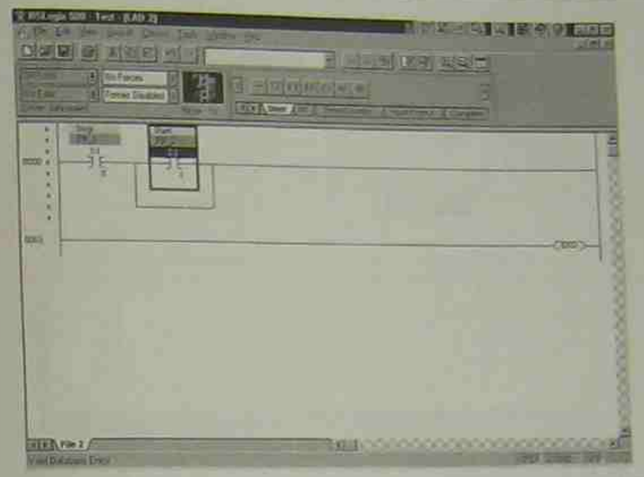


Figure 9–30 START Button with Symbol and Description Notations

To add the holding contact from the Motor Starter, point the arrow at the lower left-hand corner of the BRANCH START instruction and click. This causes the heavy line to appear as shown in Figure 9–31.

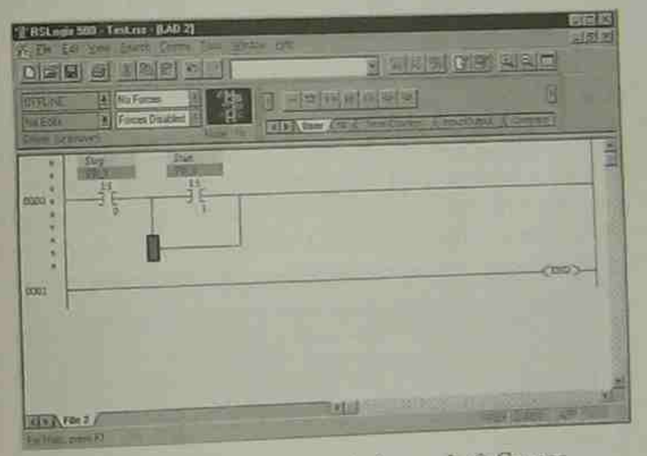


Figure 9–31 Heavy Line in Lower Left Corner

Select and left click on the Normally Open symbol above the program area to add the holding contacts that are in parallel with the START Button. As the holding contacts are controlled by the output, in this case a motor starter connected to output terminal 0 of the Output module in slot 3, the holding contacts will have the same address as the Motor Starter O:3.0/0. As before, the program ignores all unnecessary zeros, so the address can be entered as O:3/0. Figure 9–32 shows the screen after the holding contacts have been added and addressed. Notice also that the symbol notation and the description notation added earlier for address O:3.0/0 is automatically added to the holding contacts. The symbol tells what it is, in this case contacts from Motor Starter 1, that controls Exhaust Fan 1.

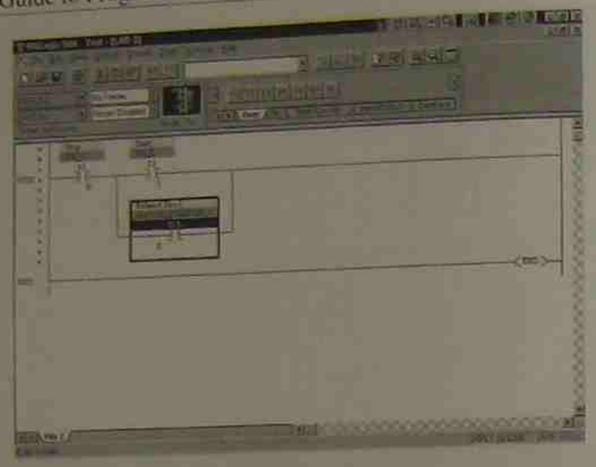


Figure 9–32 Holding Contacts with Address, Symbol, and Description Notation

The last thing necessary to complete our simple STOP/START circuit is to program the output device. The output device in this case is Motor Starter 1. Output address O:3.0/0 was given the symbol notation Motor Starter 1 when the Output Data File was opened earlier. To finish our program, point the mouse arrow at the upper right-hand corner of the parallel contacts and click the left button. This produces a heavy vertical line. Once the line is present, click on the Output symbol button above the programming area. The Output symbol looks like this: —( )—

Figure 9–33 shows the Output symbol with the (?) indicating that the symbol needs an address. Double click the left mouse button on the (?) and enter the assigned address for the output, O:3/0. Remember, there is no need to add any unnecessary zeros because the program will not display them. Figure 9–34 shows the completed STOP/START circuit with all addresses, symbol, and description notations added.

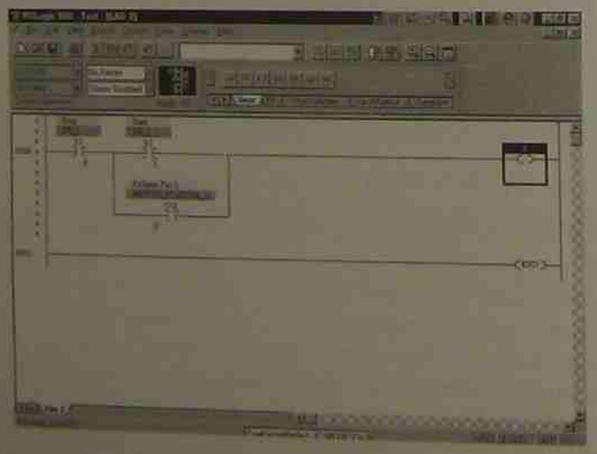


Figure 9–33 Screen with Output Symbol

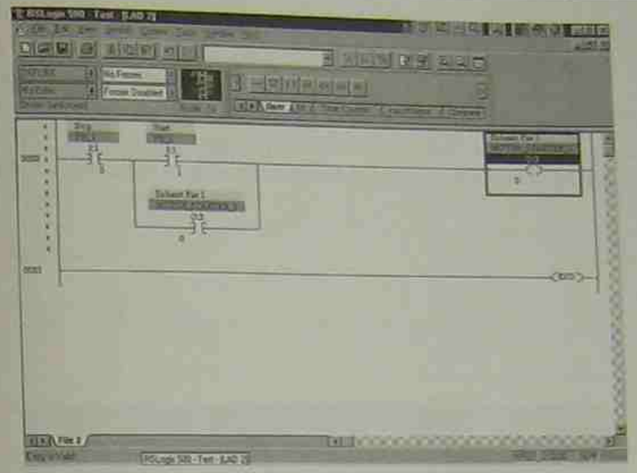


Figure 9–34 Completed STOP/START Circuit

The RSLogix software has been designed to check the finished program to make sure that all addresses and instructions are correct. To verify the correctness of the project (STOP/START Circuit), click on the *Verify Project* button at the top of the program area. The icon for the Verify Project button is a computer with a check mark. Single click on the icon and the verification process begins. If the project has no errors or faults, the program will place a message across the lower left corner of the screen. The message will read "Verify has completed, no errors found". Had there been errors in the program, the errors would be identified by program rung number. Figure 9–35 shows the screen with the verify message.

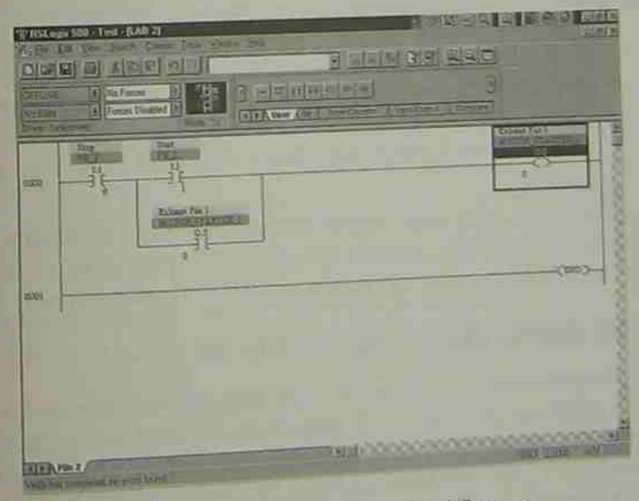


Figure 9–35 Verify Completed Screen

Remember that earlier we opened a Controller Properties screen (Figure 9–8) and one of the things noted was that no Memory Used or Memory Left values were given. That was because we had not developed a program yet. Now that a program has been developed, we can open that screen again and see the number of memory words that were used. Figure 9–36 shows the Controller Properties screen and indicates that our program, or project "TEST," used seven instructional words and 100 Data Table words. The screen shows there are 12281 instruction words left.

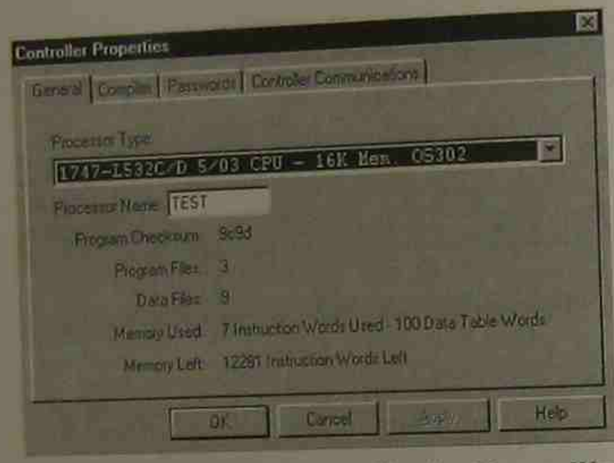


Figure 9–36 Controller Properties Screen Showing Memory Words Used

The seven (7) program words used were for the first contact (START), then a BRANCH START instruction, another contact (STOP), another BRANCH START, another contact (holding contacts), a BRANCH CLOSE instruction, and finally an OUTPUT instruction.

PROGRAMMING USING MNEMONIC STRINGS

Another method of programming can be used with the RSLogix software that reduces the time and keystrokes needed to enter any given program. This second method is sometimes referred to as the **shorthand method** and requires that the electrician or technician know the **mnemonic** names for each program instruction. Table 9–1 gives some of the most commonly used Allen-Bradley instructions and their mnemonic names.

Table 9–1 Mnemonic Names for Commonly Used Instructions

INSTRUCTION	MNEMONIC NAME
Examine if Closed (Examine ON)	XIC
Examine if Open (Examine OFF)	XIO
Output Energized	OTE
Branch Start	BST
Next Branch	NXB
Branch End	BND
Output Latch	OTL
Output Unlatch	OTU
One-Shot Rising	OSR
Timer On-Delay	TON
Timer Off-Delay	TOF
Retentive Timer	RTO
Count Up	CTU
Count Down	CTD
Reset	RES

By using the mnemonic names instead of clicking on the symbol buttons at the top of the programming area, programs can be entered more quickly. The mnemonic names are typed in strings with a space between each mnemonic name and address. Figure 9–37 shows the STOP/START circuit ladder diagram that represents the circuit previously programmed.

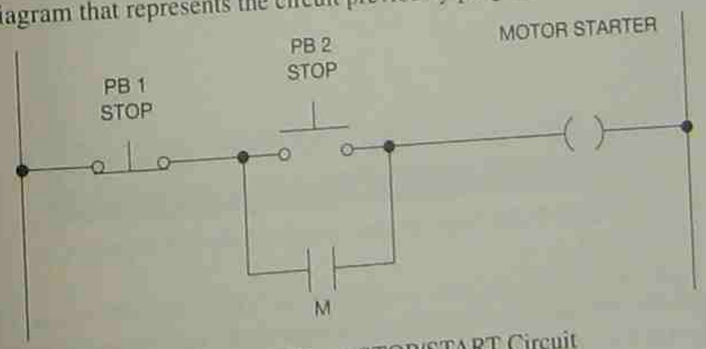


Figure 9–37 Basic STOP/START Circuit

Starting at the left of the ladder diagram and working to the right, the mnemonics needed to duplicate the circuit are:

Normally Open Contact (Examine if Closed)	XIC
Branch Start	BST
Normally Open Contact (Examine if Closed)	XIC
Next Branch OK	NXB
Normally Open Contact (Examine if Closed)	XIC
Branch Close (Branch End)	BND
Output (Output Energized)	OTE

Note: *These are the same seven instructions that were used to create the circuit from the previous programming example, and are the seven instruction words displayed on the Controller Properties screen. In the first programming example, the symbol buttons above the programming area were used instead of the mnemonic names of the symbols.*

Once the mnemonic names have been determined, the mnemonic string, with addresses, can be created.

Using the same addresses used in the previously programmed STOP/START circuit, the mnemonic string would be:

XIC I:1/0 BST XIC I:1/1 NXB XIC O:3/0 BND OTE O:3/0

Remember that a space must be used between each mnemonic name and each address.

Figure 9–38 shows the RSLogix software screen with the programming area maximized and ready to program. The processor type has been entered, a name assigned to the processor (TEST 2), and the input and output modules have been identified for the specific slots in the chassis. For this example, no *Symbols* or *Descriptions* were entered into the Data Table files for the address that will be used.

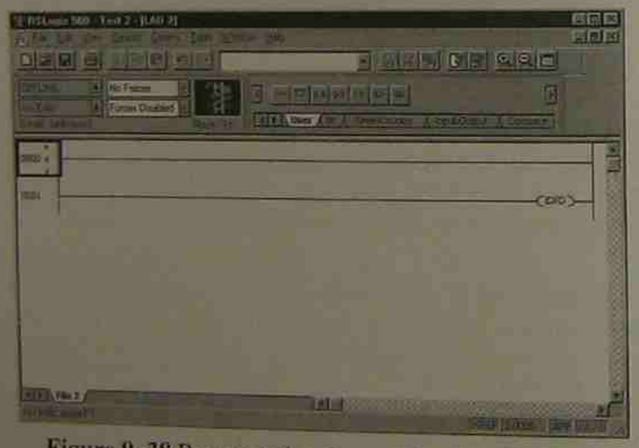


Figure 9–38 Programming Screen Ready to Program

To type in the mnemonic string that was created for the STOP/START circuit, place the mouse arrow in the box with the four zeros (0000) and double click. The elongated box that appears in Figure 9–39 is where the mnemonic string is typed.

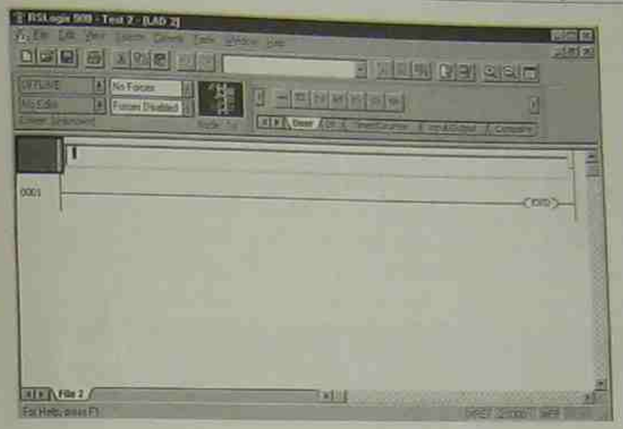


Figure 9–39 Programming Screen Ready to Enter Mnemonic String

Enter the mnemonic string by typing in the information.

XIC I:1/0 BST XIC I:1/1 NXB XIC O:3/0 BND OTE O:3/0

Figure 9–40 shows the programming screen with the mnemonic string entered.

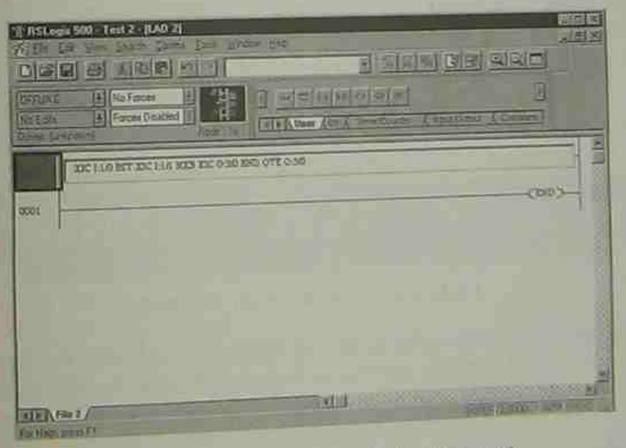


Figure 9–40 Mnemonic String Entered

Enter the program by hitting the *Enter* key on the keyboard. Figure 9–41 shows the screen with the mnemonic string converted into the finished program.

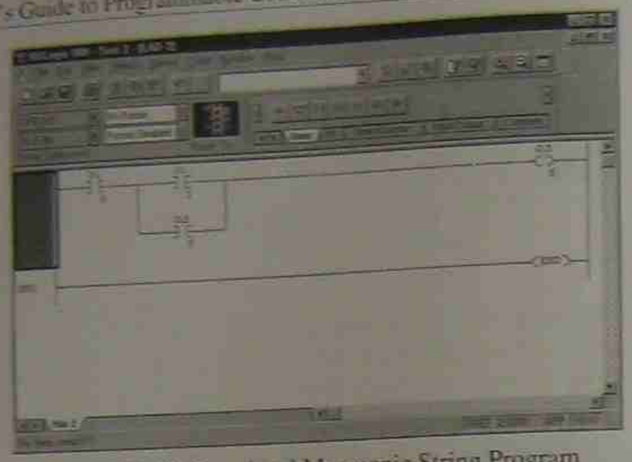


Figure 9–41 Completed Mnemonic String Program

The last step is to click on the *Verify Project* button to verify that the project is complete and free from errors. Figure 9–42 shows the completed circuit and the *Verify has completed, no errors found* message at the bottom of the screen.

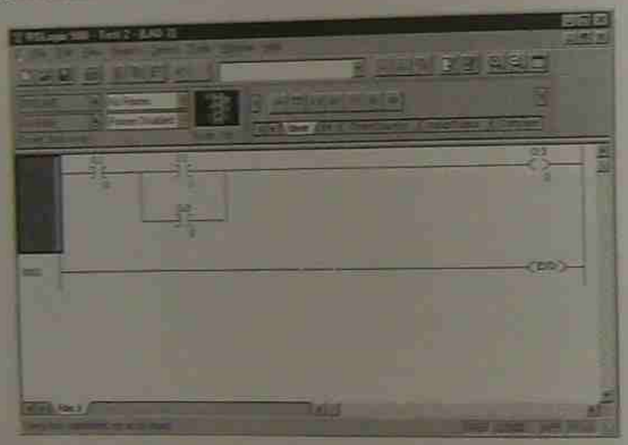


Figure 9–42 Verification Complete

The shorthand method of programming, as the example shows, is much faster. It does, however, require that the programmer know the mnemonic names of the various instructions and understand when to use the BRANCH START, NEXT BRANCH, and BRANCH END instructions.

PERIPHERALS

A communication port or ports is mounted on the back of the programmer, computer, or on the processor for connection to a printer for hard copy printouts of the program, storage registers, or report generation. To use a printer, it must be compatible with the programmer or processor.

Some printers can be connected for **serial** and/or **parallel communication**. In serial communication, the bits are sent one after the other, or sequentially. In parallel communication, groups of bits (a byte) are sent simultaneously. When using a serial printer, the rate of communication, or **baud rate**, between the printer and programming device varies with the printer and/or programming device capabilities. A baud rate of 110 indicates that 10-, 7-, or 8-bit characters can be printed per second; 300 baud indicates 30 characters per second; 1200 baud indicates 120 characters per second, and so on.

Before purchasing a printer, be careful to check the communication baud rate. There is no sense buying a printer capable of printing 120 characters per second (1200 baud rate) if the programming device can only communicate at 300 baud, or to purchase a printer that can only print 30 characters per second (300 baud rate) for a PLC or computer capable of 1200 baud. For printers that do not print as fast as the PLC or computer can communicate, an internal **buffer** often is used. A buffer is a temporary storage area where information is held until the printer catches up.

Chapter Summary

The PLC is programmed by using a dedicated programmer or with a personal computer that uses software that has been created to program a specific PLC. OP codes or operational codes are used during the programming to tell the processor what to do, while addresses are used to tell the processor where to do it. The programming device (programmer) is used to enter, modify, and monitor the user program. The program (ladder diagram) is entered by pushing keys on the keyboard in a prescribed sequence so that the results can be displayed on either the VDT of a dedicated desktop programmer and the screen of a personal computer. Programs are usually developed while the PLC is in the Off-line mode. Once the program is complete, it can be tested in the Off-line mode to ensure correct operation before it is placed in the On-line mode for final testing and verification. Keys or passwords are used to prevent unauthorized use of the PLC. From the programming device, contacts and coils can be forced ON or OFF while the circuit is operational. The FORCE ON, FORCE OFF capability should be restricted to personnel who have a complete understanding of the circuit and the driven equipment. The programming device screen can be used as a troubleshooting aid to test the circuit prior to entry into user memory, or after the circuit is entered into memory and is operational. Contacts and coils are either intensified or displayed in reverse video to indicate true logic or power flow.

Programming a PLC is not difficult, but time must be spent to become familiar with the programming device, the software, and the programming techniques used by the various PLC manufacturers.

Review Questions

1. What does the term *On Line Programming* mean?
2. What is the function of the cursor?
3. What is the FORCE feature used for?
4. T F Timer and counters use words of memory, but contacts, coils, and BRANCH START instructions do not.
5. When is *Off Line Programming* normally used?
6. What are the mnemonic names of the following instructions?
   EXAMINE ON _____
   NEXT BRANCH _____
   TIMER ON-DELAY _____
   BRANCH END _____
7. T F The RSLogix software can tell you if the right power supply has been used based on the number and mix of I/O modules.
8. Describe briefly the shorthand method of programming using the RSLogix software.

CHAPTER

10 Programming Considerations

Objectives

After completing this chapter, you should have the knowledge to:

- Define a *network*.
- Describe the term *dummy relay*.
- Understand the horizontal and vertical contact limits.
- Define the term *nesting*.
- Correctly wire and program STOP buttons.
- Describe the difference between logical and discrete holding contacts.

NETWORK LIMITATIONS

A **network** is defined as a group of connected logic elements used to perform a specific function. Figure 10–1 shows a typical network consisting of seven series contacts and three parallel branches. A network also constitutes one rung of a ladder diagram that starts at the left rail and ends at the right rail.

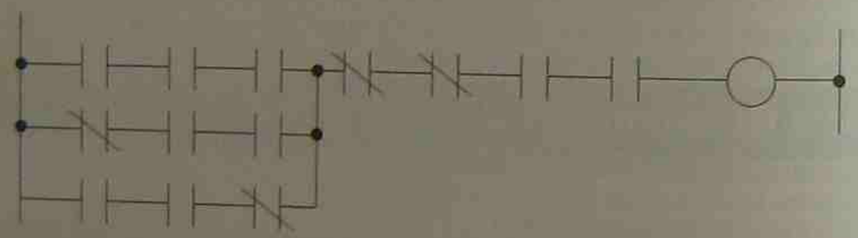


Figure 10–1 Network (Rung)

Some PLC manufacturers have virtually no network limitations, whereas other PLC systems are limited by the number of contacts or other logic symbols that can be included on the horizontal line of a network and the parallel branches (lines) that make up one network. A typical PLC network limitation of 10 series contacts per line and seven parallel lines, or branches, is shown in Figure 10–2. Additionally, some PLCs are further limited because they only allot one output per rung or network, and the output must be on the first line.

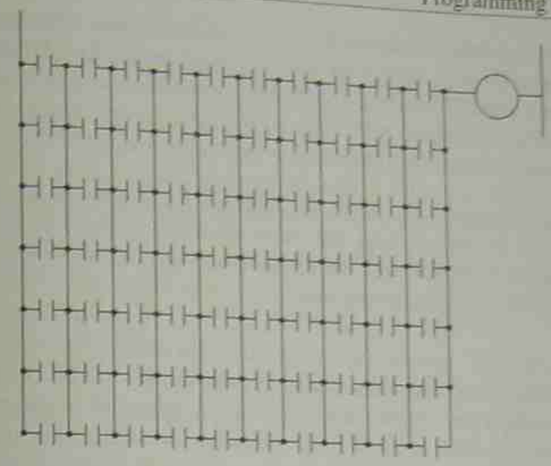


Figure 10–2 Network Limits

Note: *Special functions and other logic symbols alter the network limitations and requirements; check the operation or program manual of the specific PLC for additional information.*

While the number of elements and lines within a network is limited, only the size of user memory limits the number of networks or rungs.

When a circuit requires more series contacts than the network allows (Figure 10–3a), the contacts are split into two rungs (Figure 10–3b). The first rung contains part of the required contacts and is programmed to an internal, or "dummy," relay. Internal relays are actually a bit and word location in storage memory or an unused bit in the I/O table.

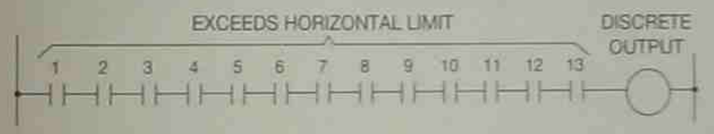


Figure 10–3a Contacts Exceed Horizontal Limit

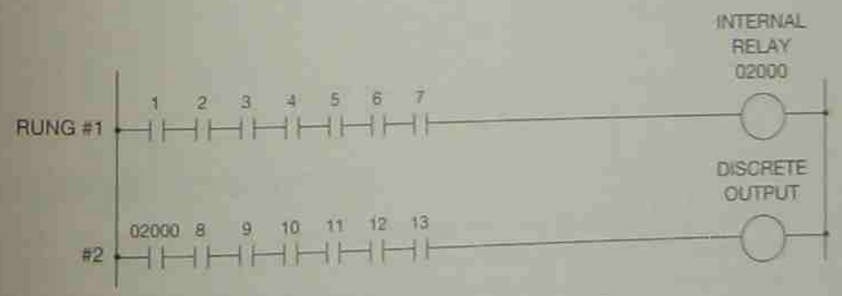


Figure 10–3b Contacts Split into Two Rungs

The address of the internal relay, 02000 in Figure 10–3b, is also the address of the first N.O. contact on the second rung. The remaining contacts (8–13) are programmed, followed by the address of the real-world output device. When the first seven contacts close, the internal output, bit 00 of word 020, is set to 1. This makes the N.O. contacts 02000 of Rung 2 true. If the other six contacts (8–13) are closed, the rung is true, and the discrete output is turned *ON*.

Note: *It is not necessary to split the contacts in any ratio. If the network allows 10 horizontal contacts, 10 could be placed on the top rung, and three could follow the N.O. contacts of the internal relay (02000) on Rung 2. This technique applies not only to N.O. contacts, but also to N.C. (or combinations of N.O. and N.C.) contacts as well.*

The internal or "dummy" relay just used does not exist as a real-world device that has to be hard-wired, but is merely a bit in the storage memory that performs the logic of a relay. In actual programming, internal control relays that do not actually exist, except in the storage memory as bits, are extensively used. The use of these internal relay equivalents is what makes the programmable controller unique, and eliminates hours of hard-wiring that shortens installation and maintenance time.

When a program requires more parallel branches than the network allows, the circuit can be split into two networks, or rungs. The first six parallel contacts are programmed to an internal or dummy relay as shown in Figure 10–4. A contact with the same address as the internal, or dummy, relay is then programmed in parallel with the remaining contacts to control the output.

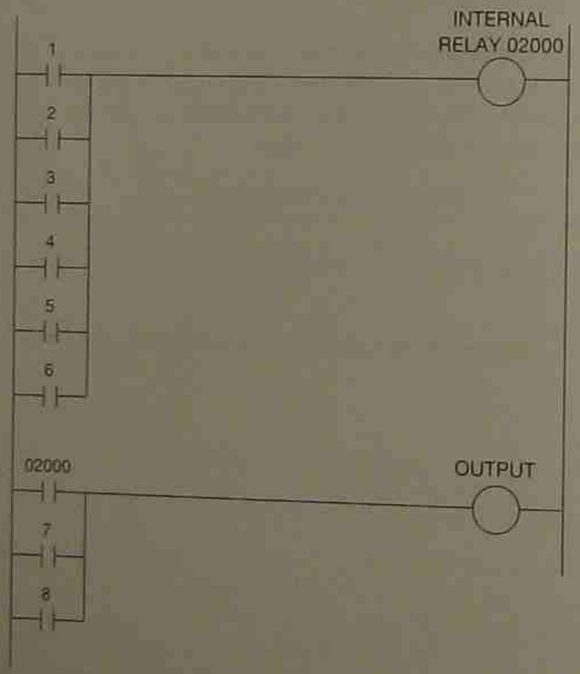


Figure 10–4 Parallel Contacts Split into Two Rungs (Networks)

Sometimes it is not the PLC that limits the number of horizontal or parallel contacts, but the display capability of the VTD. The screen may only be able to display 10 horizontal contacts, even though the PLC can be programmed with an unlimited number of horizontal contacts. While the processor may allow more contacts to be programmed than the VDT can display, it is not a good idea to do so. If the contacts cannot be seen on the screen, it is difficult to troubleshoot the circuit later. The ability to view each contact on the VDT and to know the status (*ON* or *OFF*) of each contact, as well as the status of the output devices, is what makes the PLC such a powerful tool.

Note: *When using a computer as a programming device, most software packages allow the electrician or technician to "shift" the screen to monitor all the instructions, even when the normal screen is filled with instructions.*

Many PLCs have networks that allow for more than one output (parallel outputs) (Figure 10–5). With this parallel output configuration, all of the outputs are *ON* or *OFF* at the same time, based on the network logic.

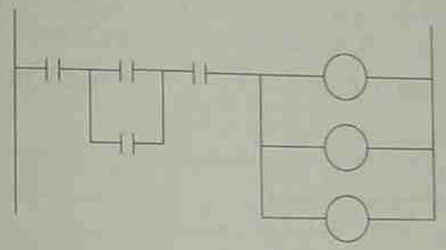


Figure 10–5 Parallel Output Format

Other PLCs, like the Allen-Bradley PLC-5, allow multiple outputs that can be *ON* or *OFF* at different times, depending on the network logic (illustrated in Figure 10–6).

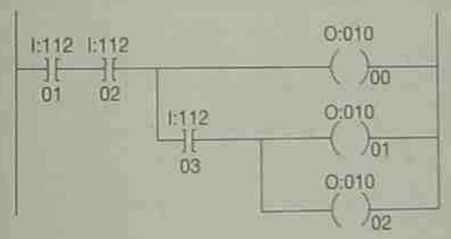


Figure 10–6 Multiple Outputs

PROGRAMMING RESTRICTIONS

In addition to the number of horizontal contacts on one line, and the number of lines in a network or rung, the PLC does not allow for programming vertical contacts (Figure 10–7). In the real world, one could wire the circuit as shown in the figure, but programming restrictions would not allow the PLC to be programmed in this manner.

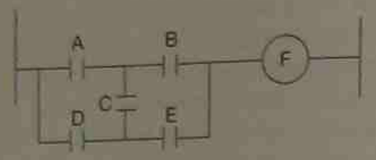


Figure 10–7 Vertical Contacts

If one analyzes the logic of the circuit in Figure 10–7, the circuit logic shows that output F can be energized by any of the following contact combinations: A, B (Figure 10–8a); A, C, E (Figure 10–8b); D, C, B (Figure 10–8c); and D, E (Figure 10-8d).

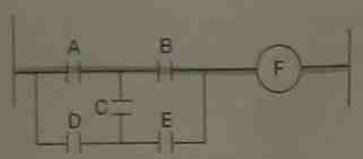


Figure 10–8a Path A, B

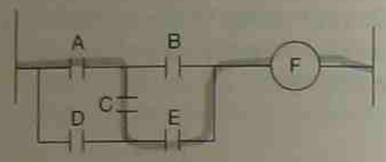


Figure 10–8b Path A, C, E

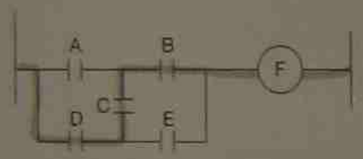


Figure 10–8c Path D, C, B

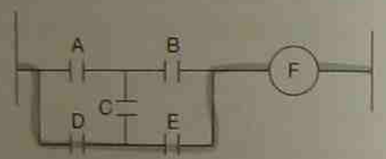


Figure 10–8d Path D, E

To duplicate the logic, the circuit could be programmed as shown in Figure 10–9.

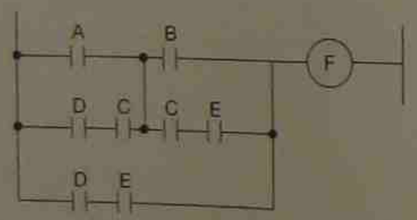


Figure 10–9 Equivalent Circuit Without Vertical Contacts

This circuit maintains the circuit logic. Contact combinations A, B; A, C, E; D, C, B; and D, E all energize output F.

Another limitation to circuit programming is the way in which the processor considers power flow, or logic continuity, when it scans a rung of logic. Flow is from left to right *only*, and vertically *up* or *down*. The processor *never* allows logic continuity (power flow) from right to left.

Normally, relay logic for the circuit shown in Figure 10–10 would indicate the following possible contact combinations to energize output G: A, B, C; A, D, E; F, E; and F, D, B, C.

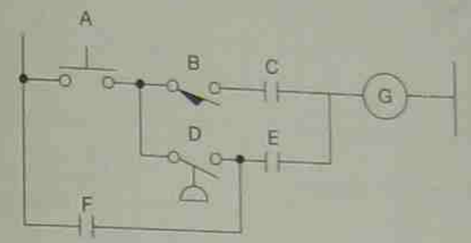


Figure 10–10 Hardwired Circuit

If the circuit shown in Figure 10–10 was programmed into user memory as shown in Figure 10–11a, the processor would ignore contact combination F, D, B, C because it would require power flow (logic continuity) from right to left. If combination F, D, B, C was required, the circuit would be reprogrammed as shown in Figure 10–11b.

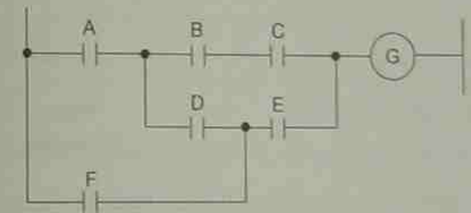


Figure 10–11a Circuit Improperly Programmed

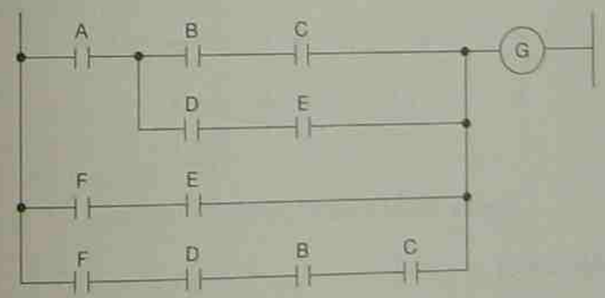


Figure 10–11b Circuit Properly Programmed

The last restriction placed on the programming of circuits into user memory by some—*but not all*—PLCs is the use of "a branch circuit within a branch circuit," or the **nesting** of contacts. Figure 10–12a is an example of a circuit that has nested contacts (L and G) or "a branch within a branch." To obtain the required logic, the circuit is programmed as shown in Figure 10–12b. The duplication of contacts J and K eliminates the nested contacts L and G.

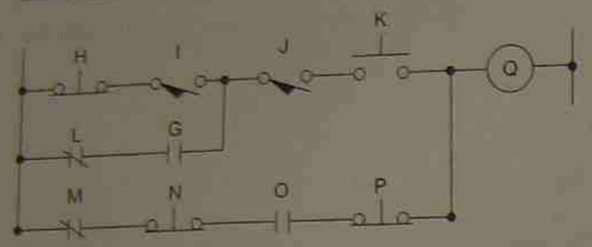


Figure 10–12a Nested Contacts

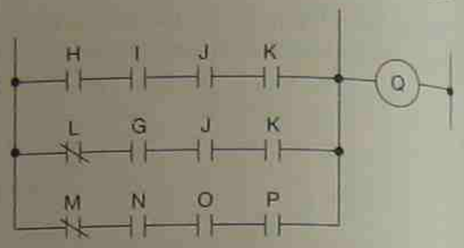


Figure 10–12b Programmed to Eliminate Nested Contacts

Figures 10–13a and 10–13b are other examples of circuits with "a branch within a branch" (in this case "branches within a branch"), and how the circuits are programmed to maintain circuit logic.

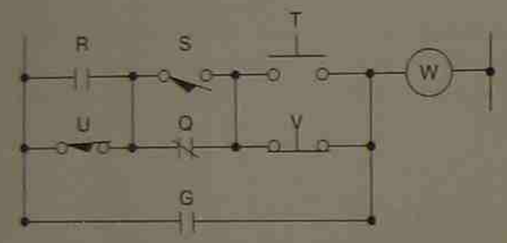


Figure 10–13a Branches Within a Branch

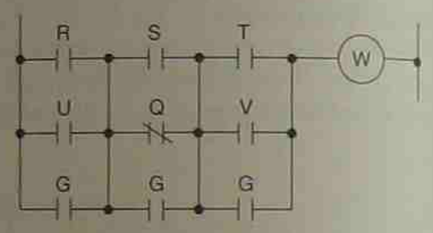


Figure 10–13b Programmed to Eliminate Branches Within a Branch

The easiest way to avoid nesting is to remember that all branches must start at a common point, and all of the branches must end at the same location. Figure 10–13b is actually three parallel branches in series as illustrated in Figure 10–14.

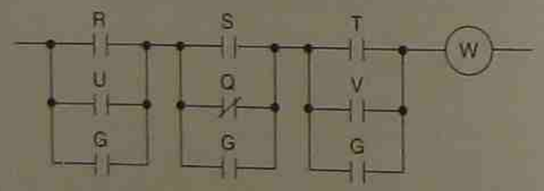


Figure 10–14 Parallel Series Combination

PROGRAM SCANNING

As discussed in Chapter 3, the processor first determines the status of the input devices, then it scans the user program, and then updates (turns *ON* or *OFF*) the outputs. The way the processor scans the program varies from PLC to PLC. One common method is to scan the program from left to right and top to bottom, similar to the way in which a book is read. In this method, the processor scans the first rung of the program from left to right, then the second rung from left to right, and continues in this fashion until all the rungs have been scanned. In the next scan, the processor returns to the first rung and starts all over again, scanning each rung in order from top to bottom.

Figure 10–15 illustrates the order of scanning for a processor that scans from left to right and top to bottom. This is the scanning method used by Allen-Bradley for their family of PLCs.

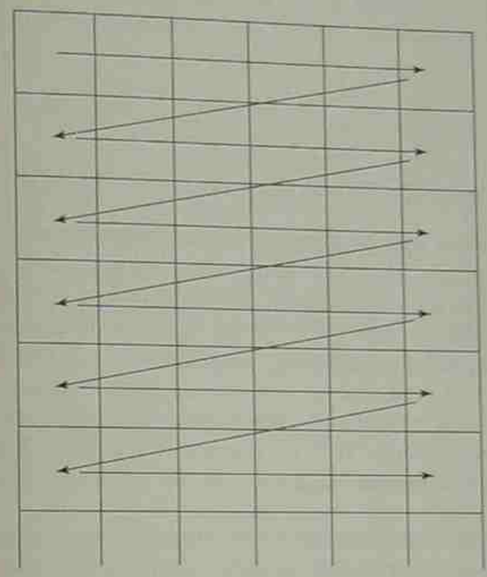


Figure 10–15 Processor Scan Left to Right, Top to Bottom

Look at the circuit shown in Figure 10–16. If S1 is closed, or true, the logic of Rung 1 is true, making the logic of Rung 2 true, which in turn makes the logic of Rung 3 true. Lamps 1, 2, and 3 are turned *ON* at the end of the first scan.

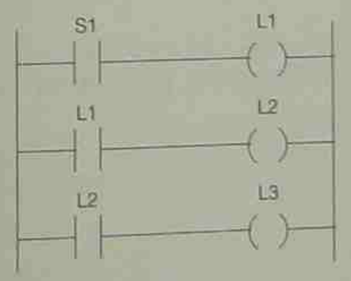


Figure 10–16 One Scan Turns On L3

If, however, the circuit was programmed as shown in Figure 10–17, L3 would *not* turn *ON* until the third scan was completed.

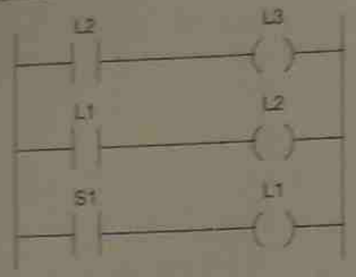


Figure 10–17 Three Scans Required to Turn On L3

In the first scan, Rung 1 is not yet logically true because the processor does not know the status of L2. Rung 2 would also not have logic continuity because the processor does not yet know the status of L1. When the processor scans Rung 3, however, it now is logically true because S1 is closed. The processor, therefore, turns *ON* L1 at the end of the first scan. On the second scan, L2 is still false, so Rung 1 has no logic continuity. Rung 2, however, is now logically true because L1 was turned *ON* after the first scan, and L1 contacts are closed, or true. S1 is still closed which keeps the third rung true, so at the end of the second scan, L1 and L2 are *ON*. It is only during the third scan that Rung 1 becomes true. With L2 now *ON*, the logic of Rung 1 is complete and L3 will be turned *ON* at the end of the third scan.

Each scan took only a matter of milliseconds (msecs) to complete, so the delay in turning on L3 is not perceptible to the naked eye. However, in certain high-speed processes, the time lost due to poor programming may be significant and must be considered.

Another example of how the processor scans is illustrated by duplicating output addresses. If the same output address is inadvertently used twice in one program, the last rung in which the address is used will indicate the status of the output. For example, if the address is used in Rung 5 and the logic of Rung 5 is true (which would tell the processor to turn *ON* the output), and the address is used again in Rung 13 and the logic of Rung 13 is false, the output will not be turned *ON* because the processor scanned Rung 13 last, and the last state of the output (true or false) will be based on the last rung scanned.

Modicon controllers such as the 984 scan and solve the user program by networks. A network consists of 11 columns and 7 rows as shown in Figure 10–18. Column 1 through 10 can contain

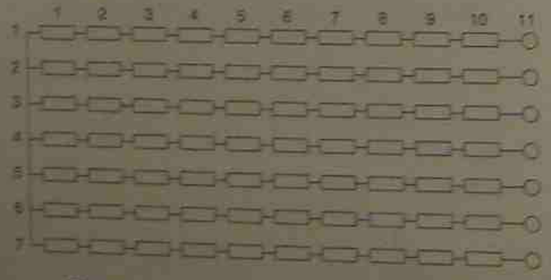


Figure 10–18 Modicon Network Format

any programming instruction, including coils. The 11th column, however, is reserved exclusively for coil elements. Each instruction within the network is referred to as a **node**.

Where Allen-Bradley scans the user program from left to right and top to bottom, the Modicon 984 scans the network vertically from column 1 to column 11 as shown in Figure 10–19.

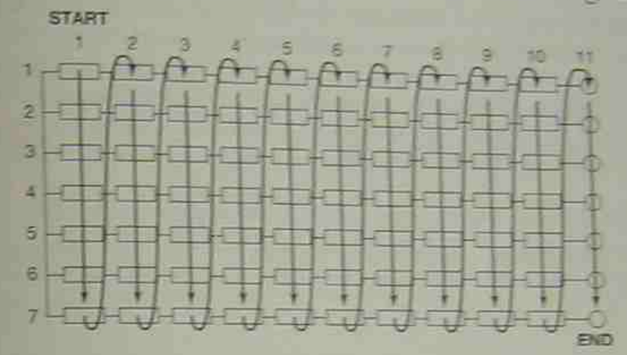


Figure 10–19 Order for Solving Network

The network is solved beginning at the top of Column 1. The instructions are read vertically from the top to the bottom of Column 1 as shown in the figure. When the last instruction in Column 1 has been scanned, the processor goes to the top of the second column and scans from the top to the bottom. This process is followed until all 11 columns have been scanned. The results of the scan are then written into memory.

Although this scheme of program scanning is different from the method used by Allen-Bradley, the logic continuity, or power flow, limitations are the same. Power can flow up and down and from left to right, but *cannot* flow from right to left.

Columns 1 through 10 can contain any program instruction, including an output coil. The 11th column is reserved exclusively for coil elements. Figure 10–20 shows a typical programmed network.

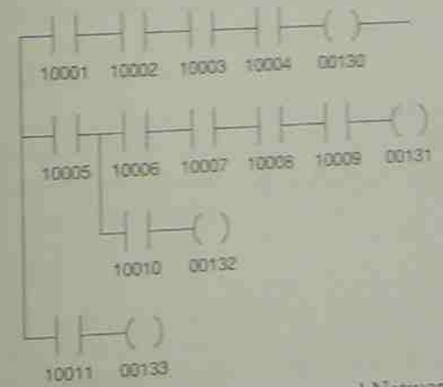


Figure 10–20 Typical Programmed Network

In Row 1, the output coil is shown in Column 5, whereas the remaining coils in the network are shown in Column 6, 3, and 2, respectively. This is the position of the program elements for solving the logic of the program during the program scan.

Figure 10–21 shows the same program but with the coils shown as they would typically be displayed on the programming device. Although the display shows all of the output coils in the 11th column, the coils are scanned in their actual column order 1 through 11.

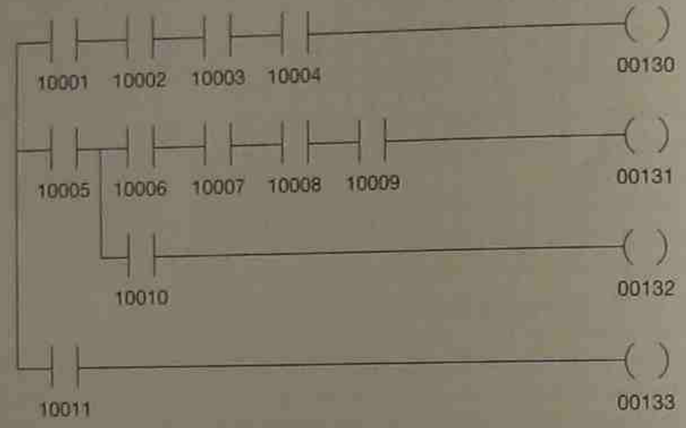


Figure 10–21 Expanded Display

PROGRAMMING STOP BUTTONS

During the early days of programmable controllers, it was common for salespeople to use a demonstrator model that had the STOP buttons wired normally open (N.O.). This technique was used so the salespeople did not have to explain why a STOP button was shown as normally open in a PLC program.

Figure 10–22a shows a standard STOP/START station ladder diagram, and Figure 10–22b shows the equivalent diagram used by some PLC salespeople.

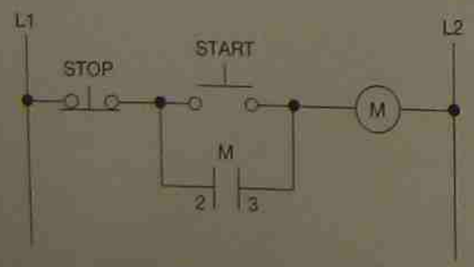


Figure 10–22a Standard STOP/START Ladder Diagram

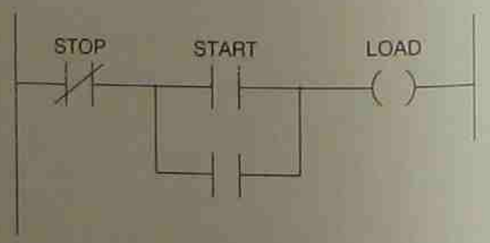


Figure 10–22b PLC Programmed STOP/START Circuit

From an understanding of how a STOP/START circuit works, and an understanding of the EXAMINE ON and EXAMINE OFF instructions, it is easy to see that the only way the circuit could be logically true would be for the STOP button to be wired open. By using a N.O. STOP button, the EXAMINE OFF instruction is true and the circuit energizes when the START button is pressed. The problem is that once the circuit is energized, the only way it can be stopped, or turned *OFF*, is if the STOP button is pushed and the contacts close. While this circuit will work, there is a built in *danger* that must be considered. If a STOP button is wired in a N.O. position, the switch is impossible to close if it becomes jammed, and it would be impossible to deenergize the circuit. Similarly, if a wire breaks on the STOP button, it is possible to complete the logic of the circuit and energize the equipment, but *impossible* to deenergize the equipment. With the wire broken, changes in the status of the STOP button cannot be conveyed to the processor, and the circuit and/or equipment cannot be deenergized.

Safety Note: *All STOP buttons must be wired so that a failure of the switch or a broken wire will automatically break logic continuity and turn the circuit* OFF. *A good programmer will always wire the devices and program the circuit so that if the real-world device fails, it creates a safe condition, not a safety hazard.*

This practice of wiring STOP buttons in the N.O. position was common during the 1980s. If an electrician or technician finds equipment wired in this fashion, he or she should change the stop buttons to N.C. and the PLC program from EXAMINE OFF to EXAMINE ON.

The START button should be wired normally open (N.O.) and programmed with an EXAMINE ON instruction. As a general rule, all input devices, except STOP buttons, are wired normally open and given EXAMINE ON instructions in the program.

LOGICAL HOLDING INSTRUCTIONS

In previous programming examples we have used the output address to address the holding contacts. This method of providing holding logic works well in many applications and eliminates the need to actually wire the holding contacts on the motor starter. When the output address is also used for the holding contacts (logic), the circuit is maintained logically because the output point has been turned *ON*. This is no guarantee, however, that the actual motor starter connected to the output module has been energized.

DISCRETE HOLDING CONTACTS

The only real way to know that the starter has energized is to actually wire the holding contacts of the motor starter to an input module point, and use that address when programming the holding contacts. This method has many advantages, and short of installing a motor sensor, is the best way to verify that the motor starter has been energized.

OVERLOAD CONTACTS

It is common practice *not* to wire the overload contacts to an input module, but instead to wire the overload contacts in series with the starter coil (shown in Figure 10–23). When wired in this manner, a motor overload that opens the overload contacts also opens the circuit to the starter coil, and the starter will drop-out, or deenergize. When the starter deenergizes, the holding contacts (which

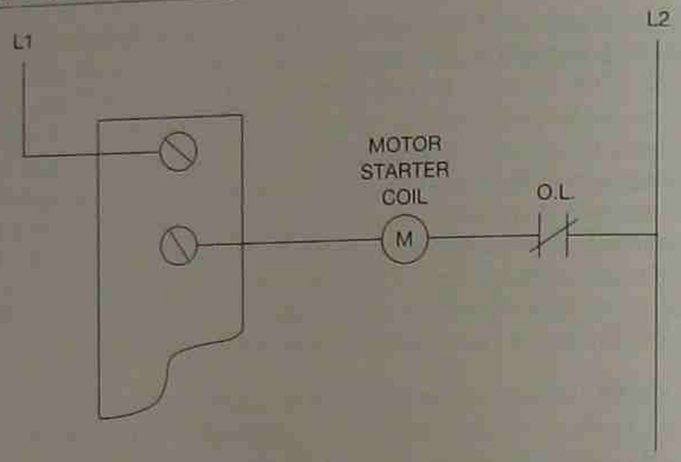


Figure 10–23 Overload Contacts Wired in Series With the Starter Coil

must be wired to an input module and programmed in the PLC circuit) also open, and the PLC circuit loses logic, which in turn, turns *OFF* the point on the output module that is connected to the starter coil. This arrangement only works if the holding contacts are wired to an input module and programmed into the PLC program.

If the holding contacts are not wired to an input module, but instead the output address is used for holding logic, the overloads trip and interrupt the circuit to the starter coil and the starter deenergizes, but the PLC logic is *not* broken. Without the holding contact to open the logic in the PLC program, the output module point would remain *ON*, even though the starter coil circuit is open. This wiring scheme can cause a safety hazard. With the logic remaining true, the motor restarts automatically when the overload is reset.

Not wiring the overloads to an input module and programming them in the circuit can also cause problems in sequential motor circuits, or other automated circuits that require one device to be *ON* before other devices are turned *ON*.

EXAMPLE: On a milling machine it is necessary for the coolant pump to always be running before the cutting wheel can operate. If the overload for the coolant pump is not wired to an input module, but wired in series with the motor starter coil instead, an overload opens the circuit to the coolant pump motor starter and turns *OFF* the coolant pump. The programmed circuit is not aware that the coolant pump has shut down, and allows the cutting wheel to continue to run. If the overload had been wired to an input module, the continuity of the circuit would have been broken when the overload opened, and both the coolant pump and cutting wheel would have been turned *OFF*.

When the decision is made not to wire the holding contacts on the starter, but instead to use the address of the output for logic continuity, the actual overload contacts *must* be wired to an input module and referenced in the PLC program.

Hardwiring the overload contact(s) to an input module and addressing the contact(s) in the PLC program has the added benefit of being able to look at the status of the overload contacts to determine if they have tripped. Any address used in the PLC program can be viewed on the programmer screen to determine if the device is *ON* or *OFF*. By viewing the screen, it can be determined that the overload has tripped. Knowing why the motor starter deenergized is helpful when troubleshooting.

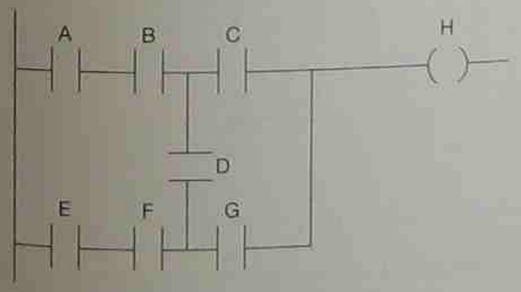
Chapter Summary

Each PLC has a maximum network size, or matrix, that limits the number of horizontal and vertical contacts for any one network, or rung. The only limitation to the number of rungs (networks) is memory size. Since the processor reads power flow (logic) from left to right *only*, and vertically either *up* or *down*, the logic of a relay circuit must be examined carefully to ensure that the logic is maintained when the circuit is programmed into the user memory. The program device does not allow contact to be programmed vertically, but the logic of a ladder diagram with vertical contacts may be duplicated by adding additional contacts. Depending on the PLC, contacts may or may not be programmed as a "branch-within-a-branch," or nested.

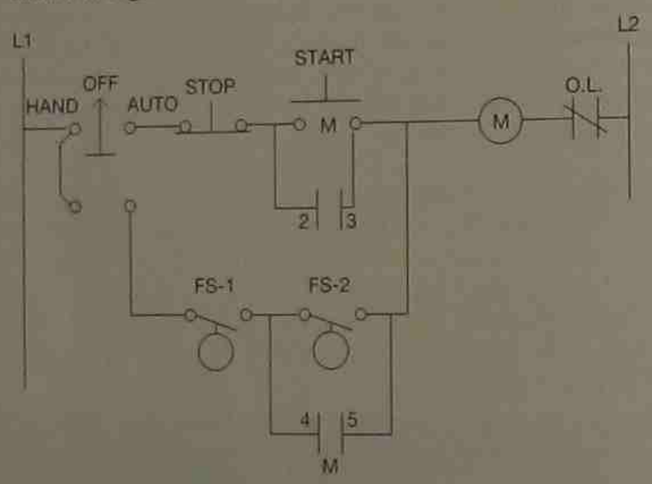
Consideration must be given to the way that the processor scans the program to eliminate unnecessary scans before a line or rung of logic goes true. STOP buttons must always be wired as normally closed and use an EXAMINE ON instruction to work correctly and safely. Holding contacts can be either logical or discrete (real world). The discrete method has the added advantage of verifying that the motor starter has indeed energized. Hard-wiring the overload contacts to an input module has the added benefit of being able to look at the status of the overload by means of a programming device to determine if they have tripped.

Review Questions

1. Define the term *network*.
2. Draw and label a diagram to show how a network that requires 14 series contacts to control a discrete output can be programmed on a PLC that limits series logic elements to 10 on one line.
3. Draw a circuit with nested contacts.
4. Draw a circuit that retains the logic of Figure 10–A, which has a vertical contact (D), so the circuit could be programmed into a PLC.



5. Power flow, or logic, in a PLC is considered to be (check all correct answers):
   a. up to down only
   b. up or down only
   c. left to right
   d. right to left
   e. up or down and from left to right
   f. up, down, and from right to left
   g. up to down only and left to right
   h. up to down only and right to left
   i. up or down and left to right or right to left
6. Write a program for a PLC that does *not* allow nested contacts, for the Hand-Off-Auto circuit shown in Figure 10–B.



7. Explain how a STOP button must always be wired, and why.
8. List two ways that holding contacts can be programmed.
   a. _____
   b. _____
9. Explain one advantage of wiring the overload contacts to an input module and then programming the overload address into the PLC program.

CHAPTER

11 Program Control Instructions

Objectives

After completing this chapter, you should have the knowledge to:

- Write a program using a *latching relay*.
- Understand the term *retentive*.
- Write a program using a *master control relay*.
- Understand the importance of a *safety circuit*.
- Describe how an *immediate input instruction* could be used.
- Describe how an *immediate output instruction* could be used.
- Write a program using the *jump and label instructions*.
- Give a reason for using the *temporary end instruction*.

MASTER CONTROL RELAY INSTRUCTIONS

In standard relay control systems, a master control is often used to control power to the entire circuit or just to selected rungs. This allows selected rungs, or the whole circuit, to be deenergized by turning off the master control relay (MCR). Figure 11–1 shows a typical hardwired master control relay that controls power for the whole circuit.

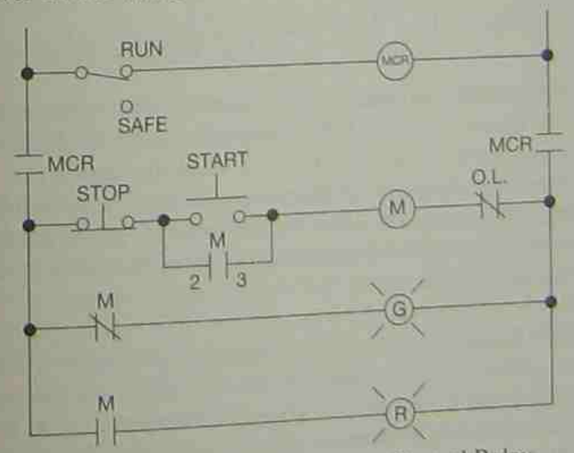


Figure 11–1 Hardwired Master Control Relay

Master control relays are often used with circuits that have off delay timers so the circuit can be shut down completely *without* waiting for the timers to time out.

With PLCs, a master control relay (MCR) function can be programmed to control an entire circuit or just selected rungs of a circuit. When the MCR instruction is programmed as shown in Figure 11–2, any rungs that follow can only become energized if the MCR instruction is energized, or set to a True condition.

Note: *MCRs are sometimes called Master Control Resets because they reset the output to zero or OFF.*

Allen-Bradley PLC-5, SLC 500, and MicroLogix Master Control Reset (MCR) Instruction

As shown in Figure 11–2, when the MCR instruction is *true*, the outputs in the rungs that follow are controlled in a normal fashion by the logic programmed for each rung. If the MCR instruction is *false*, the rungs below the MCR are also *false* and cannot energize even if the programmed logic for each rung is *true*. The exception is latching relay instructions. Latching relay instructions will remain *ON*, or *true*, even when the MCR instruction is *false*. Latching relay instructions are covered later in this chapter.

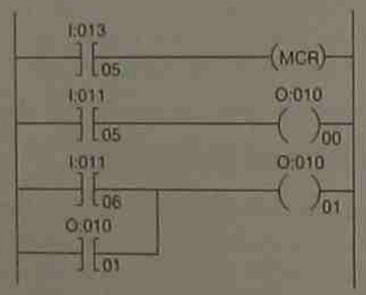


Figure 11–2 Allen-Bradley MCR Instruction

To create an area, or zone, within a program that will turn *OFF* all nonretentive outputs within the area, or zone, two Master Control Reset (MCR) instructions are used. Figure 11–3 shows how the two instructions are programmed. Rung 1 has the first MCR instruction that is controlled by input device I:012/00. Note that the second MCR instruction has been programmed in Rung 5 with no control device preceding the instruction. When an instruction is programmed in this manner, it is said to be programmed unconditionally. This second MCR instruction is used to end the zone controlled by the MCR instruction in Rung 1.

When input device I:012/00 goes *true*, the MCR instruction goes *true*, and the rungs between the two MCRs—Rungs 2, 3, and 4—are controlled by the first MCR. As long as the MCR in Rung 1 remains *true*, the normal logic of the rungs below the instruction will control output devices O:010/00, 01, and 02. When the MCR instruction in Rung 1 goes *false*, all output devices between the MCR in Rung 1 and the MCR in Rung 5 will be turned *OFF*, regardless of the logic of the individual rungs (Rungs 2, 3, and 4). Rung 6 is outside the area, or zone, controlled by the MCR instructions and works independently of the MCRs.

Additional MCR zones can be created within the program using pairs of MCR instructions. MCR zones cannot be nested within another MCR zone.

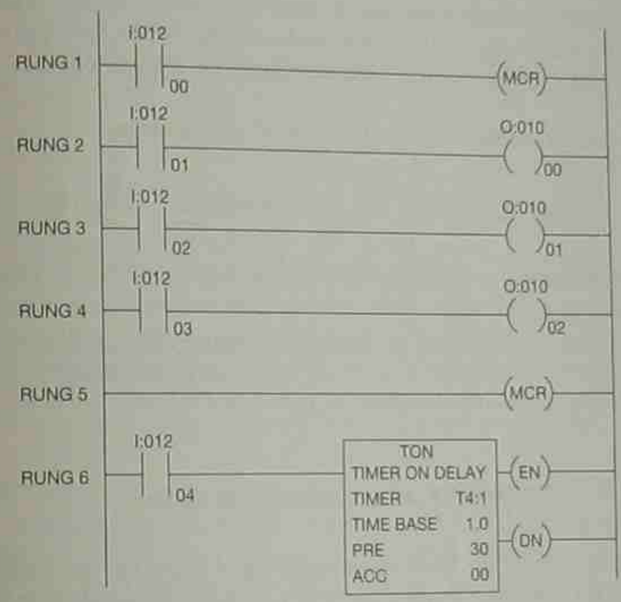


Figure 11–3 Two MCR Instructions Used to Create a Control Zone

Safety Note: *A programmed MCR must never be used to replace a hardwired master control relay that provides emergency shutdown.*

LATCHING RELAY INSTRUCTIONS

Before discussing how a latching relay function is programmed with a PLC, a review of the traditional hardwired latching relays may be helpful.

Latching relays are used when it is necessary for contacts to stay open and/or closed even though the coil is only energized for a short time (40 milliseconds) by a momentary signal. Latching relays are often used for lighting applications where multiple circuits are needed for the lights in a large room or auditorium. Instead of having a switch for each circuit, a multiple contact latching relay is used. Each lighting circuit is wired to a set of contacts on the latching relay. One switch now controls the latching relay, which in turn controls several circuits of the lighting load. The latch and unlatch feature of the relay only requires three wires, so wiring is greatly reduced, and controlling the latching relay from multiple locations is quite simple. Another advantage of using the latching relay occurs when lighting is normally turned *ON* at the start of the work day, and left *ON* until quitting time. Under these circumstances, if a normal relay is used, the coil needs to be

energized all day long to keep the lights *ON*. The latching relay, however, needs only to be momentarily energized to latch, or turn the connected load *ON*, and the relay remains latched even though the coil is no longer energized. By not having the coil energized all day, there is an energy saving.

Latching relays normally use two coils: one to *latch* and one to *unlatch*. There is a mechanical linkage that holds the relay in the latched, or closed, position. When the unlatch coil is energized, the coil action disengages the mechanical latch and allows the relay to open.

Figure 11–4 shows the wiring diagram for a mechanical latching relay.

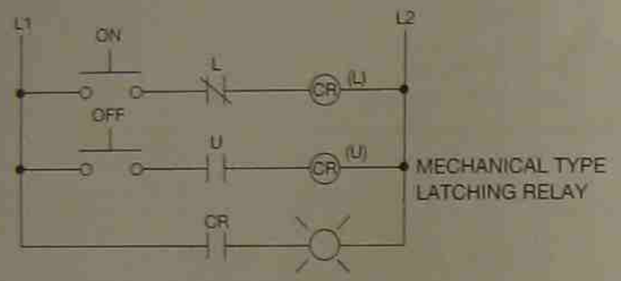


Figure 11–4 Mechanical Latching Relay

When the *ON* button is pushed, the latch coil energizes and opens the N.C. latch (L) contacts and closes the unlatched (U) contacts. Opening the N.C. L contacts deenergizes the L coil. The length of time it took to push the *ON* button and to energize the latch coil (which opened the N.C. L contacts and deenergizes the L coil) required only a fraction of a second. During the short time the latch coil energized, it closed the N.O. CR contacts, completing the circuit to the lamp. CR contacts remain closed even though the latch coil deenergized because of the mechanical latch mechanism. To open the mechanically latched contacts to turn the light *OFF* requires the *OFF* push button be pushed. The U contacts in the unlatch coil circuit are now closed. Pushing the *OFF* button energizes the unlatch coil, which in turn closes the N.C. L contacts, and opens the U contacts, which deenergizes the unlatch coil. For the brief instant that the unlatch coil is energized, it releases the mechanically latched CR contacts so they can open and turn the light *OFF*.

Mechanical latching relays can be replaced by programming bits in the storage memory of the PLC as internal latching instructions, or by programming real-world outputs using the latch and unlatch instructions. Like the dummy relays discussed earlier, the programmed internal latching relay do not exist a real-world devices but can perform all the logic of an actual latching relay.

Programmed latch and unlatch instructions, like their physical real-world counterparts, are **retentive** during a power failure. When the processor loses power, is switched to either the TEST or PROGRAM modes, or detects a major fault, outputs are turned *OFF*; the state of the latch instruction is retained in memory, however, and when power is restored or the processor is switched back to the RUN mode, the outputs that were *ON* previously return to their *ON* state.

Figure 11–5 shows latch and unlatch rungs as they would be programmed using the Allen-Bradley Output Latch (OTL) and Output Unlatch (OTU) instructions used for their PLC-5, SLC 500, and MicroLogix PLCs.

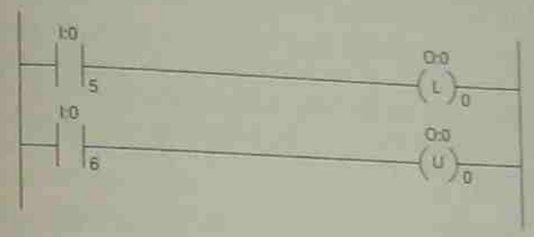


Figure 11–5 Programmed Latch and Unlatch Rungs

The OTL instruction is a retentive output instruction that can be programmed to turn *ON* an output device. This instruction cannot turn an output device *OFF*. Once an output device has been turned *ON* by an OTL instruction, a OTU (output unlatch) instruction must be used to turn the device *OFF*. As these two instructions must be used in pairs, it follows that they will use the same address. Note in Figure 11–5 that both the latch (L) and unlatch (U) coils have the same address (O:0/0).

The address, which is bit 0 of output image table word 0, will be set to 1, or *ON*, when the latch rung is true (input I:0/5 closed), and will be cleared to 0 or turned *OFF* when the unlatch rung is true (input I:0/6 closed). Like normal latching relays, only a momentary closure of input device I:0/5 latches output coil (L) O:0/0, and the output remains latched, or *ON*, until the unlatch coil (U) rung is true by closing input I:0/6.

N.O. and N.C. contacts programmed after the latch and unlatch rungs, and given the same address as the latch and unlatch coils perform the same functions as N.O. and N.C. contacts of a standard real-world latching relay (Figure 11–6).

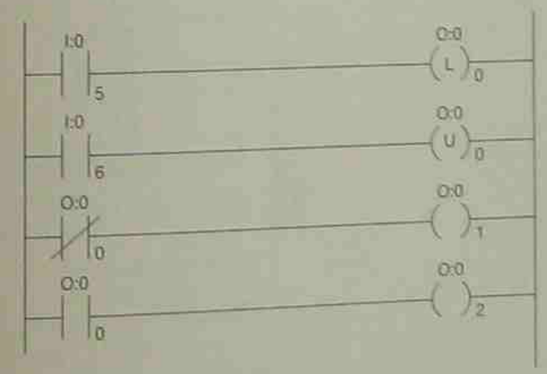


Figure 11–6 Programmed Latching Circuit

Output O:0/1 is *ON* and output O:0/2 is *OFF* when O:0/0 is not latched.

Output O:0/1 goes *OFF* and O:0/2 comes *ON* when O:0/0 is latched.

Normally, an internal storage bit (dummy relay) is used for the latch and unlatch address, rather than an actual discrete output address. If a discrete output address is used, the output, once latched, remains *ON*, even if programmed after an open MCR Rung. When an internal storage bit is used for the latch and unlatch address, the bit is still retentive, but turns *OFF* if programmed *after* an MCR Rung that is open.

Although all PLCs are designed and manufactured to the highest standards and quality, a latching or MCR instruction should not be depended on for machine safety. A hardwired safety circuit should always be added. A safety circuit is recommended by most PLC manufacturers to ensure maximum safety rather than depending on a programmed MCR or latching relay alone.

SAFETY CIRCUIT

The concept of safety circuit has been discussed earlier in the text, and is an important enough subject to be covered again. The National Electrical Manufacturing Association (NEMA) standards for programmable controllers recommends that consideration be given to the use of emergency stop functions which are independent of the programmable controller. The standard reads in part:

> When the operator is exposed to the machinery, such as loading or unloading a machine tool, or where the machine cycles automatically, consideration should be given to the use of an electromechanical override or other redundant means, independent of the controller, for starting or interrupting the cycle.

Figure 11–7 shows how a control relay (CR) and *safe-run switch* is added to interrupt line2 to the discrete output devices of an automatic machine or process.

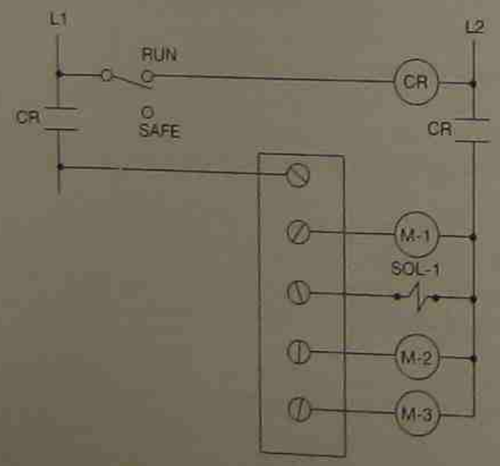


Figure 11–7 Safety Circuit

IMMEDIATE INPUT INSTRUCTION

The immediate input instruction is used when it becomes necessary to update the status of an input word in the input image table prior to the normal scan sequence. For some high-speed processes it may be necessary to know the status of a certain bit (or bits) in a particular input word prior to the next normal input image table update. Allen-Bradley refers to their immediate input instruction with the mnemonic IIN. To use the instruction to interrupt the normal scan cycle and update an input word, the IIN instruction is programmed prior to the rung(s) that include critical input addresses. The address for the IIN instruction is the rack number and module group for the input devices that need to be updated prior to the normal update. The rack number and the module group correspond to a word in the input image table. Figure 11–8 illustrates how the IIN instruction is programmed.

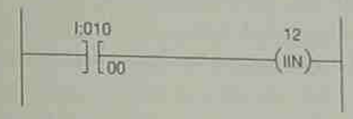


Figure 11–8 Programming an Allen-Bradley PLC-5 Immediate Input Instruction

If input I:010/00 is true, the program scan is interrupted while the input status for input image word 12 is updated (rack 1, module group 2). As soon as the input word has been updated, the scan continues. Figure 11–9 shows the typical scan cycle and how it is interrupted for the immediate update of the status of the input devices in rack 1, module group 2.

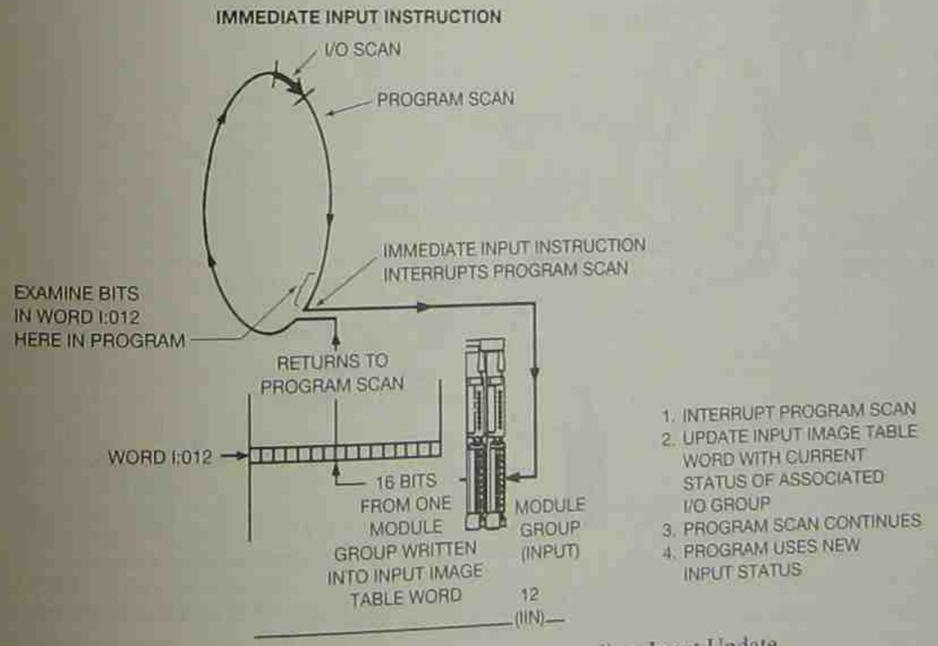


Figure 11–9 Scan Interrupted for Immediate Input Update
(Courtesy of Allen-Bradley)

IMMEDIATE OUTPUT INSTRUCTION

The Immediate Output instruction is used when it is necessary to update the output image table prior to the completion of a normal program scan. Allen-Bradley PLC-5 used the mnemonic IOT. When the immediate output instruction (IOT) is enabled, or true, the instruction will interrupt the normal program scan and update a word in the output image table. Similar to the IIN instruction, the rack number and module group number is used to identify the location of the output devices that are to be immediately updated. Figure 11–10 shows how the instruction is programmed for outputs in rack 1, module group 3 (output word 013).

Figure 11–10 Programming the Immediate Output Instruction

When input I:010/00 is true, the IOT instruction is enabled and the scan is interrupted so that the output devices in rack 1, module group 3 (output word 013) are immediately updated. Figure 11–11 shows the interrupted program scan for the IOT instruction.

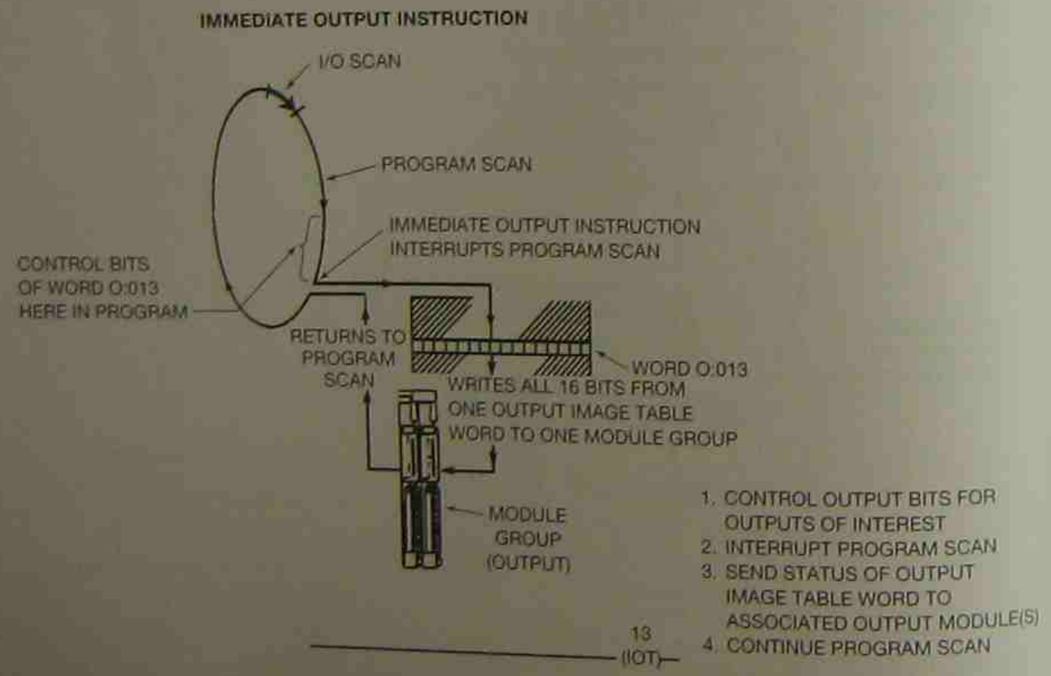


Figure 11–11 Scan Interrupted for Immediate Output Update
(Courtesy of Allen-Bradley)

JUMP AND LABEL INSTRUCTIONS

Used in combination, these two instructions allow for skipping over portions of the program to save program scan time. If there is a portion of the program that is not operational during certain portions of the process, the portion that is not used and/or needed can be jumped over or bypassed until it is needed again. By jumping over parts of the program, the scan time is decreased and more scans can be completed in a given period of time, which, in turn, means more frequent updating of information in the program. The jump instruction (JMP) tells the processor to jump over a portion of the program. Where to jump to is controlled by the label instruction (LBL). Figure 11–12 shows a jump and label instruction in Allen-Bradley PLC-5 format. When the JMP instruction is true, the processor will jump over Rungs 3 and 4 and go directly to Rung 5, as shown.

The JMP instruction is assigned a three-digit number from 000–255, and the rung that the processor is to jump to is given an LBL of the same number. In Figure 11–12 the JMP and LBL instruction is also enabled, and the processor is instructed to jump all successive rungs until it reaches the rung that contains the label instruction with the number 20. In this illustration, only two rungs are jumped. In actual practice, any number of rungs can be jumped.

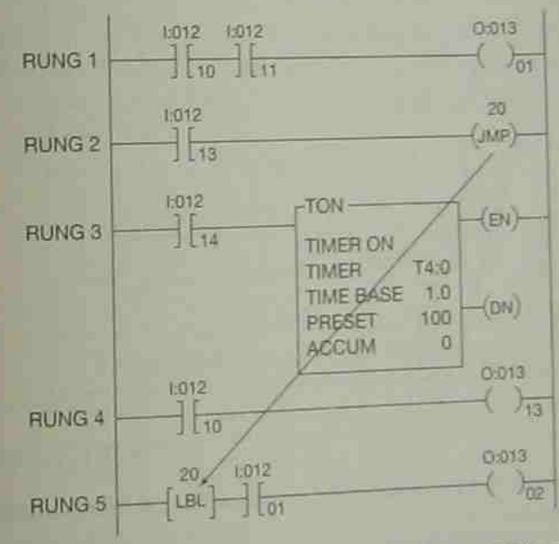


Figure 11–12 Programming the Jump (JMP) and Label (LBL) Instructions

The jump and label instruction can be used to jump forward or backwards in the program, depending on need. Jumping backward adds to the total scan time.

Note: *Jumping backward an excessive number of times could increase the scan time to a point where the watchdog timer will time out (the processor has a watchdog timer that is reset on each scan). If the scan time exceeds the watchdog timer's preset time, the processor goes into a* fault *condition.*

JUMP TO SUBROUTINE, SUBROUTINE, AND RETURN INSTRUCTIONS

The jump to subroutine, subroutine, and return instructions are used to direct the processor to a subroutine file within the program, scan it, return to the program, and continue to scan. The formats used by the various PLC manufacturers to program these instructions varies widely, and for this reason, only instruction blocks are shown in Figure 11–13. The blocks are identified using the Allen-Bradley mnemonics, or label: jump to subroutine (JSR); subroutine (SBR); and return (RET) as used with the SLC 500 and MicroLogix PLCs. Subroutines are very valuable for program organization, and for using blocks of programming logic over and over by simply changing the variables used in the subroutine. To use this group of instructions, consult the programming guide for the PLC system that is being used.

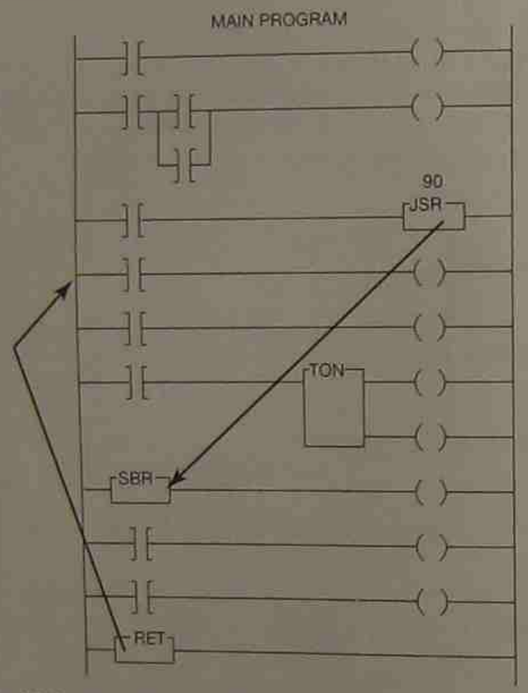


Figure 11–13 Jump to Subroutine, Subroutine, and Return Instructions

TEMPORARY END INSTRUCTION

The temporary end instruction (TND) is used to place a temporary end to the program scan. When the instruction is inserted into the program, the processor stops scanning the program at this point and returns to the start of the program. When the processor returns to the start of the program, the watchdog timer is reset to zero. This instruction is often used when a new program is being debugged for the first time, because it allows for portions of the program to be checked out without running the entire program. Figure 11–14 shows a TND using the Allen-Bradley format. When the TND instruction is true, the processor stops scanning at Rung 3 and does not scan Rungs 4 and 5. The TND instruction is also available with the SLC 500 and MicroLogix PLCs.

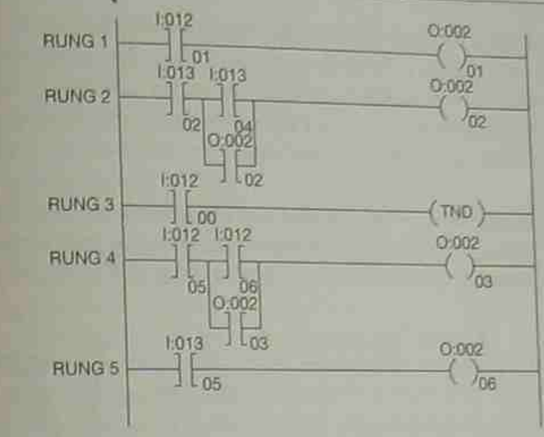


Figure 11–14 Allen-Bradley PLC—Temporary End Instruction (TND)

ALWAYS FALSE INSTRUCTION

The always false instruction (AFI) is also used when debugging a new or modified program. By inserting the always false instruction in a rung, the rung will always be false, regardless of the status of other instructions in the rung. Figure 11–15 shows a rung of logic with the AFI instruction programmed at the start of the rung.

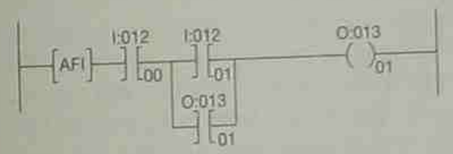


Figure 11–15 Always False Instruction

ONE SHOT INSTRUCTION

The one shot instruction (ONS) is an input instruction that makes the rung true for just one program scan, based on a false-to-true transition of the instruction that precedes the one shot instruction. Figure 11–16 shows a rung of logic with the one shot instruction programmed after input device I:011/04, which is controlling output O:010/12. Again, the Allen-Bradley PLC-5 format is used. This instruction is also used with the SLC 500 and MicroLogix PLCs.

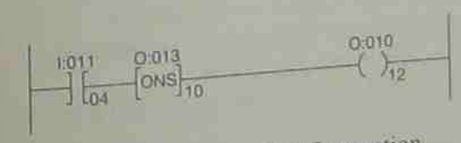


Figure 11–16 One Shot Instruction

When the rung is programmed as shown, the output is turned *ON* for one scan, and one scan *only* when input I:-11/04 closes (makes a false-to-true transition). The output cannot be turned *ON* again until the input device is first opened, then closed again, making a false-to-true transition. With the next false-to-true transition, the output device is again only turned on for one scan. This is a beneficial instruction when an output signal or operation is wanted for only one scan.

Math operations, data or word moves, and the like, are completed only once if a one shot instruction is put in series or "anded" with the instruction. The one shot instruction is often used with timers and counters for changing preset and accumulated values. This technique is discussed further in Chapter 14.

Note: *This chapter has covered some of the basic instructions that are available for programming with a PLC. As the instruction sets vary with each manufacturer, it is necessary that the programming manuals be consulted to determine what instructions are available, what their mnemonics or designations are, and how to properly use them in a program.*

Chapter Summary

Latching and master control instructions can be programmed to serve the same control functions as their real-world counterparts. Where personnel safety is a factor, a safety circuit should be added, instead of depending on latching or master control relay instructions alone. Through the use of various PLC instructions, the programmer can cause the processor to immediately update specific input and output devices, label and jump between specific rungs of logic, jump to subroutines then back to the original rung by using special instruction blocks, place a temporary end statement in the program to limit the amount of program that the processor will scan, use an always false instruction to keep a rung of logic from going true, and program one shot instructions that limit activity to only that one program scan. Although the mnemonics used by the various manufacturers will differ for each of their instruction sets or blocks, the main purpose of each instruction is the same. Once the electrician or technician has mastered programming one type of PLC, the transition to other types becomes easier.

Review Questions

1. Will both programmed and real-world latching relays, if latched, remain latched if power is lost and then restored?
2. Latching relays are normally used when it is necessary for:
   a. contacts to open and/or close only while the coil is energized.
   b. contacts to open and/or close every 30 seconds.
   c. contacts to stay open and/or closed even though the coil is only energized a short time.
   d. none of the above.
3. Explain why a *master control relay* is often used with off delay timers.

4. A master control relay can be used to control:
   a. selected circuit rungs (networks).
   b. entire circuits.
   c. individual contacts within a rung (network).
   d. all of the above.
5. Define the term *unconditional*.
6. When using PLCs, the National Electrical Manufacturing Association (NEMA) recommends that consideration be given to stop functions independent of the PLC. Explain briefly why this recommendation is made.
7. Define the term *retentive*.
8. Give an example of how an *immediate input instruction* is used.
9. What is the primary function of an *immediate output instruction*?
10. What does the *jump and label instruction* do?
11. Give one reason why you might use a *temporary end instruction*.
12. What is the function of the *always false instruction*?
13. What is the function of a *one shot instruction*?

CHAPTER

12 Programming Timers

Objectives

After completing this chapter, you should have the knowledge to:

- Describe how *pneumatic time delay relays* work.
- Write a program using *ON delay* and *OFF delay* timers.
- Describe the difference between an *ON delay timer and a retentive timer*.
- Explain how to extend the time range of timers by *cascading*.

PNEUMATIC TIMERS (GENERAL)

To fully understand how a PLC can be programmed to replace pneumatic time delay relays, both the basic pneumatic time delay relay and the standard symbols used must be understood.

Figure 12–1 shows a complete Allen-Bradley pneumatic timing relay, and Figure 12–2 shows a cutaway view of the contact and timing mechanism.

Figure 12–1 Pneumatic Timing Relay
(Courtesy of Allen-Bradley)

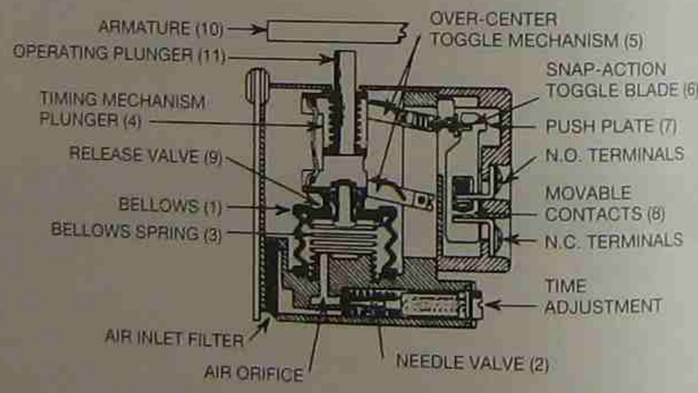


Figure 12–2 Cutaway View of Contact Unit and Timing Mechanism
(Courtesy of Allen-Bradley)

For the timer to time when power is applied (coil energized), the solenoid unit—coil, core piece, and armature—is mounted so that the natural weight of the armature (10) pushes down on the operating plunger (11). This causes the bellows (1) and bellows spring (3) to collapse the bellows, and dispel the air out through the release valve (9). When the coil is energized, the armature is attracted magnetically to the pole pieces, and lifts up and off the bellows assembly. Air now comes in through the air inlet filter, passed the needle valve (2), and fills the bellows with air. The incoming air expands the bellows upward, pushing on the timing mechanism plunger (4). As the plunger rises, it causes the over-center toggle mechanism (5) to move the snap-action toggle blade (6) upward. This picks up the push plate (7) that carries the movable contacts (8) to open the N.C. contact and close the N.O. contact. The time it takes for the bellows to fill with air and activate the contact mechanism is controlled by adjusting the needle valve in the air orifice. The valve is adjusted with a screwdriver as shown in Figure 12–3. A counterclockwise rotation moves the needle valve further into the air orifice, restricting airflow into the bellows, slowing the airflow, and increasing the time it takes for the bellows to expand and operate the contact mechanism. Conversely, clockwise adjustment of the needle valve decreases the time it takes the bellows to fill with air and activate the contacts after the armature has been lifted off the bellows mechanism.

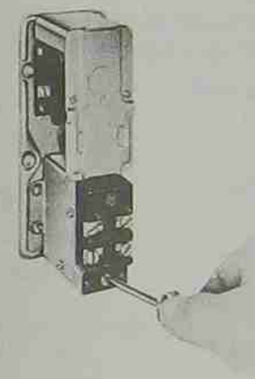


Figure 12–3 Pneumatic Timer Adjustment
(Courtesy of Allen-Bradley)

When the contact action is delayed after the coil has been energized and the armature is lifted up and off the bellows mechanism, it is called ON delay.

When the coil of an **ON delay timer** is deenergized, the armature drops down, pushing on the operating plunger, which in turn pushes down on the bellows expelling air through the release valve. The downward motion of the bellows causes the snap-action toggle blade to instantaneously snap the N.C. contact closed and the N.O. contact open.

To summarize the ON delay timer, the delay in contact operation begins *after* the timer coil has been energized, or turned *ON*. When the timer coil is deenergized, or turned *OFF*, the contacts go back to their normal condition instantly. Figure 12–4 shows a pneumatic timer with the solenoid unit mounted for ON delay.

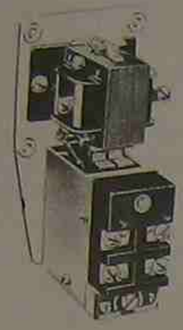


Figure 12–4 OM Delay Timer
(Courtesy of Allen-Bradley)

Figure 12–5 illustrates the electrical symbols used to indicate ON delay contacts.

The arrowhead indicates that movement is up. Since ON delay contacts can only time *after* the armature has lifted up off the bellows, this method of identifying timed contacts is easy to remember. Another common method of identifying timed contacts is shown in Figure 12–6.

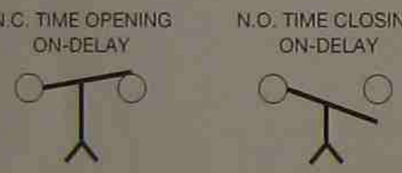


Figure 12–5 ON Delay Symbols

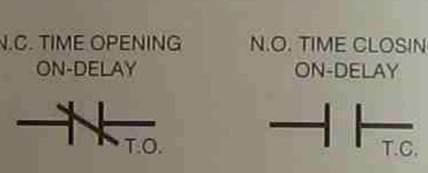


Figure 12–6 ON Delay Symbols

Note: *Remember that normal for contacts is how they are open or closed, with the coil of the relay deenergized and time expired.*

For a pneumatic timer to time when power is removed from the relay coil (**OFF delay**), the solenoid unit is mounted as shown in Figure 12–7. With a spring holding the armature up, no weight is applied to the bellows assembly, and the bellows are filled with air in a fully extended position.

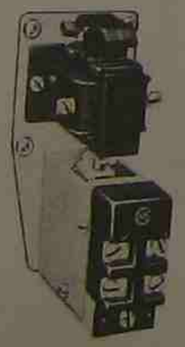


Figure 12–7 OFF Delay Timer *(Courtesy of Allen-Bradley.)*

Note: *Compare Figure 12–4 (the ON delay) with Figure 12–7 (the OFF delay) to clearly see the difference in the mounting of the solenoid assemblies.*

When the coil is energized and the armature moves down, the armature pushes on the operating plunger. The plunger pushes on the bellows assembly, and all air is immediately forced out of the bellows through the release valve. This causes the snap-action contact assembly to instantly open the N.C. contact and close the N.O. contact. The contacts stay in this configuration as long as the coil is energized and the armature is holding the bellows mechanism down (compressed).

When the relay coil is deenergized, or turned *OFF*, the spring on the armature lifts it up and off the operating plunger, which allows the bellows to start to fill with air. The N.C. contact remains open, and the N.O. contact remains closed until the bellows are filled with enough air to activate the snap-action contact mechanism. When the contact mechanism has been activated, the N.C. contacts go closed and the N.O. contacts go open.

Figure 12–8 shows the electrical symbols for OFF delay contacts.

To avoid confusion when reading electrical drawings with OFF delay contacts, it must be remembered that normal refers to the coil *after* it has been deenergized (turned *OFF*), and the time set for the timer has elapsed. The other symbols used for OFF delay contacts are shown in Figure 12–9.

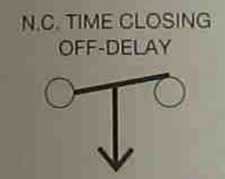


Figure 12–8 OFF Delay Symbols

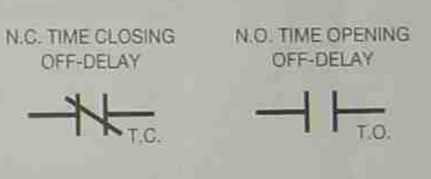


Figure 12–9 OFF Delay Symbols

Figure 12–10 compares both types of symbols used for ON delay and OFF delay timer relays.

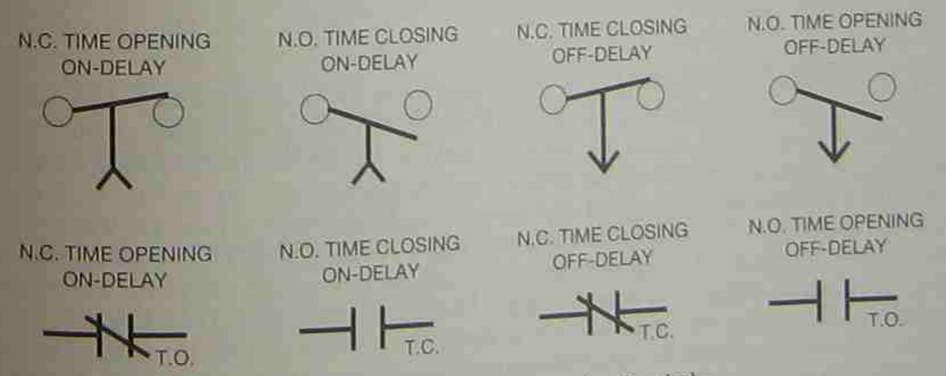


Figure 12–10 ON and OFF Delay Symbols

Reviewing the two types of symbols commonly used in motor control diagrams, an electrician or technician should have no trouble determining the type of timing relay (ON delay or OFF delay) used, or what is normal (open or closed) for the timed contacts.

The basic pneumatic timing relay is designed so that additional instantaneous contacts may be added, as shown in Figure 12–11. The instantaneous contacts operate when the coil is energized or deenergized independent of the timing mechanism. Figure 12–12 shows the electrical symbol for contacts with an asterisk (*) which is sometimes used to indicate instantaneous contacts of a timing relay.

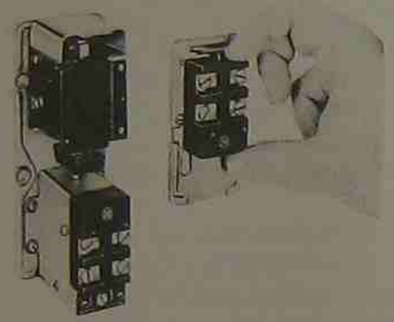


Figure 12–11 Adding Instantaneous Contacts
(Courtesy of Allen-Bradley.)

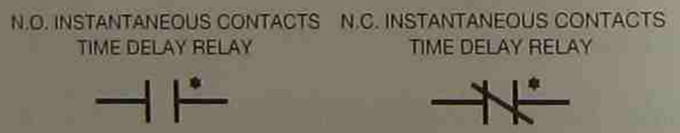


Figure 12–12 Instantaneous Contact Symbols

Figure 12–13a shows a simple light circuit controlled by an ON delay timer set for five seconds. The amount of delay is written near the timer coil on the diagram for understanding and for troubleshooting. Figure 12–13b shows that when S^1 is closed, the coil of the pneumatic timer energizes, lifts the armature up and off the bellows, and the timing starts. Figure 12–13c shows the circuit after three seconds have elapsed (not enough time for the timer to time out) with the lamp circuit still open. After five seconds have elapsed (Figure 12–13d), the N.O. time closing contacts close, and the lamp lights. As long as S^1 remains closed, the timer coil is energized, and the timed contacts stay closed. When S^1 is opened (Figure 12–13e), the coil circuit is broken, and the coil deenergizes. This causes the timed contacts to open, thereby turning OFF the lamp. The timed contacts will open the instant the coil deenergizes because they are timed only when power is applied to the coil.

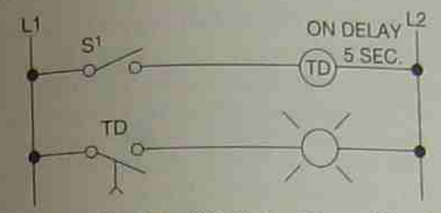


Figure 12–13a ON Delay Timer Circuit

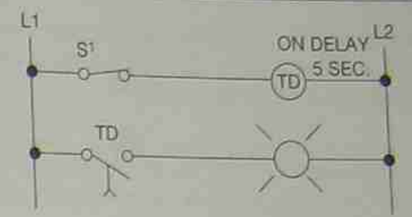


Figure 12–13b Instant S^1 is Closed

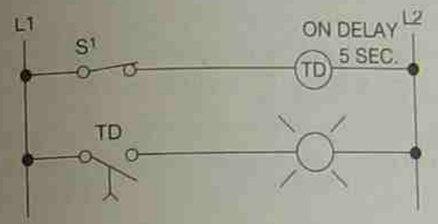


Figure 12–13c Three Seconds After S^1 is Closed

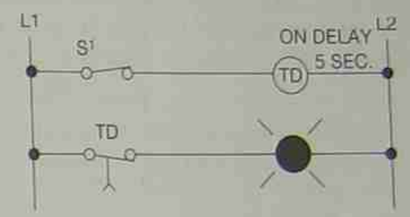


Figure 12–13d Five Seconds After S^1 is Closed

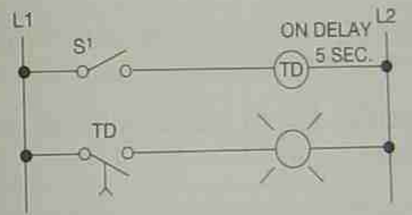


Figure 12–13e Instant S^1 is Opened

Figure 12–14a shows the same circuit but with an OFF delay timer. When S^1 is closed (Figure 12–14b), the TD coil energizes, drawing the armature down and compressing the bellows. This causes the N.O. OFF delay contacts to go closed instantly, and the lamp lights. When S^1 is opened (Figure 12–14c), the TD coil is deenergized, the spring-loaded armature is lifted up and off the bellows, and the five-second timing begins. Figure 12–14d shows the circuit after three seconds have elapsed. The lamp remains energized until the full five seconds have elapsed, and the N.O. contacts time out and open (Figure 12–14e).

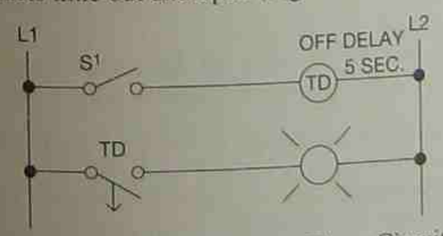


Figure 12–14a OFF Delay Timer Circuit

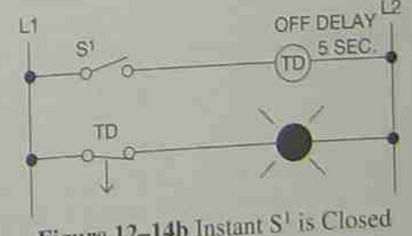


Figure 12–14b Instant S^1 is Closed

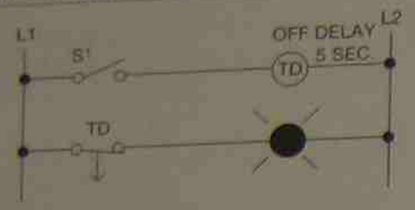


Figure 12–14c Instant S^1 is Opened

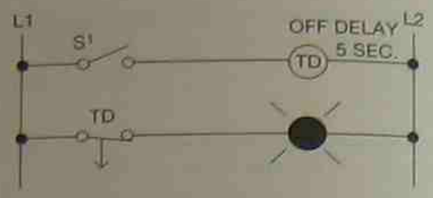


Figure 12–14d Three Secons After S^1 is Opened

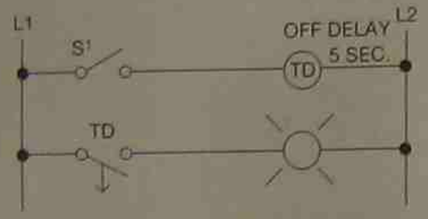


Figure 12–14e Five Seconds After S^1 is Opened

Instead of the bellows assembly, like the pneumatic time delay relay, PLC timers use internal solid state circuitry (clocks) for timing intervals or time base.

The various PLC manufacturers use varying approaches for the actual programming of timers. Several methods which are typical for most PLCs will be discussed.

Because it is an easy transition from pneumatic timer concepts to programming concepts, the Allen-Bradley approach to programming timers is discussed first.

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX TIMERS

Figure 12–15 shows the timer format used by Allen-Bradley. The timer consists of a timing block containing the timer number (address), time base (1 second or 0.01 seconds), and the preset and

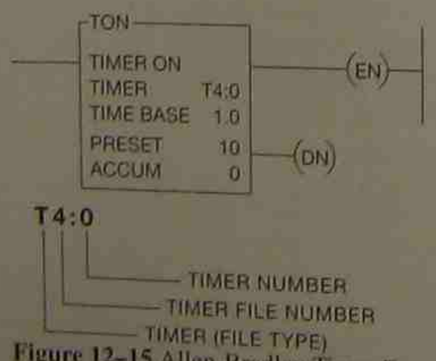


Figure 12–15 Allen-Bradley Timer Format

accumulated times. The preset time can be programmed with any value from 0 to 32,767. If a time base of one second was assigned, 32,767 would equal 9.1 hours (32,767 ÷ [60 × 60] = 9.1); if a time base of 0.01 (one hundredth of a second) was assigned, 32,767 would equal approximately 5.5 minutes (32,767 × 0.01 ÷ 60 = 5.46).

The two lines to the right of the block are the enable (EN) bit and done (DN) bit that indicate the status of the timer. The timer address (T4:0) identifies the timer file number and timer number. T4:0 indicates timer file 4, timer 0.

File 4 is the default file from the data table for timers using the PLC-5, SLC 500, and the MicroLogix family. The PLC-5 can be programmed to use files 3-999 for additional timer files. The SLC 500 can be programmed to use files 9-255 for additional timer files. The MicroLogix 1000 is limited to one timer file which is the default file, file 4.

By only having one timer file, the MicroLogix 1000 is limited to 40 timers (timer files 0 through 39) whereas the SLC 500 can use files 0 through 255. The PLC-5 can have timer numbers from 0 through 999.

The EN bit is set to 1 (or is true) whenever there is a logic path to the timer block. The DN bit is set to 1 (or is true) when the accumulated value equals the preset value, and the timer has timed out. Figure 12–16 shows how the information for a timer is stored. Three words of memory are used for each timer programmed. The first word of memory uses the first 8 bits for internal use and uses bit 13 for the DN bit, bit 14 for the timer timing bit (TT), and bit 15 for the EN bit. The next two words store the preset and accumulated values of the timer.

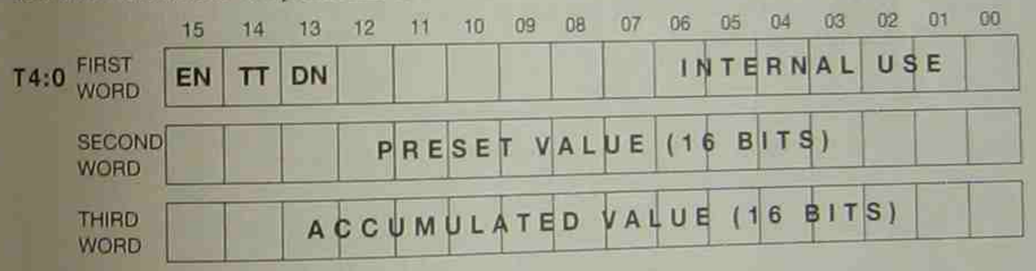


Figure 12–16 Timer Storage Format

Note: *There is no need to remember or memorize the timer bit numbers because the programming software accepts the mnemonics DN, TT, and EN. When addressing timer contacts, the timer number is entered first, followed by the timer bit. For example, T4:1/TT or T4:1.TT addresses the TT bit of timer 1, file 4.*

The timer enable bit, bit 15, is set to 1, or turned ON, when the rung goes true, and remains set until the rung goes false or a reset instruction resets the timer.

Note: *The EN bit can be used as an instantaneous contact.*

The TT bit, bit 14, is set to 1, or turned ON, when the rung goes true, and remains *ON* until: the rung goes false; the DN bit is set to 1 (accumulated value = preset value); timing is completed; or a reset instruction resets the timer.

Note: *The TT bit can be used to control a timer timing light that is only* ON *when the timer is actually timing. Figure 12–17a shows how the TT bit is used to control an indicator light, and Figure 12–17b shows the equivalent circuit using a pneumatic timer.*

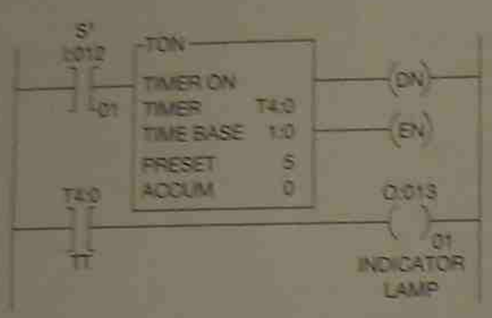


Figure 12–17a TT Bit Used to Control an Indicator Lamp

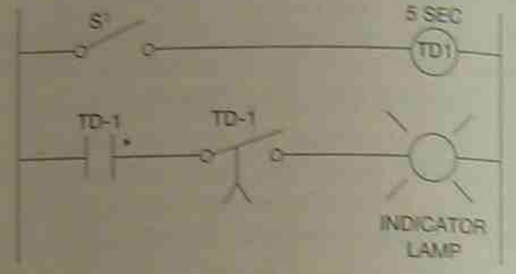


Figure 12–17b Pneumatic Timer Circuit to Control an Indicator Lamp

The DN bit, bit 13, is set to 1 when the accumulated value is equal to the preset value. The DN bit remains set to 1, or *ON*, until the rung goes false or a reset instruction resets the timer.

Note: *The DN bit can be used to control an output, or for other logic within a program.*

Figure 12–18 shows an ON delay timer and how it is programmed to control outputs O:013/01, O:013/02, O:013/03, and O:013/04.

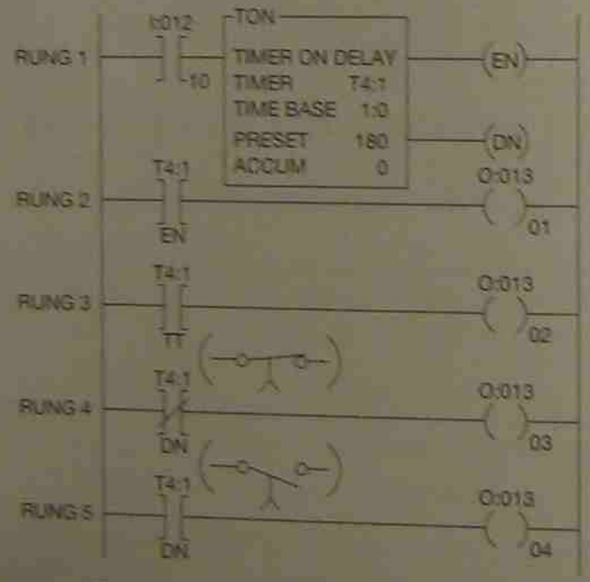


Figure 12–18 Programmed TON Timer

When bit I:012/10 (input device) is true, or set to 1, the timer rung is true, and the processor starts timer T4:0 timing and sets the EN and TT bits to 1. This turns *ON* outputs O:013/01 and O:013/02 in Rungs 2 and 3. The accumulated value increases in one-second intervals. The output in Rung 4, controlled by an EXAMINE OFF instruction, is true as long as the preset is not equal to the accumulated value. The EXAMINE OFF instruction addressed with the timer DN bit acts like a normally closed-time opening contact, and does not open until the accumulated value equals the preset value. The EXAMINE ON instruction in Rung 5 with the DN bit address acts like a normally open-time closing timer contact, and does not close (or go true) until the accumulated value is equal to the preset. When the accumulated time does equal the preset time, the DN bit is set to 1, and output O:013/04 is turned *ON* and output O:013/03 is turned *OFF*. Once the timer instruction has completed timing, the TT bit is reset to 0 and the output (O:013/02) of Rung 3 is turned *OFF*.

Like a pneumatic ON delay timer, when power is removed, the timer is reset to 0. The PLC-5 timer instruction is reset when the input device (I:012/10) is opened.

Figure 12–19 shows a typical timing chart. Notice that when the Rung condition is true (*ON*), the timer will time, but if the Rung goes false (*OFF*), the timer resets to 0, as illustrated, during the first two minutes of the timing diagram.

Figure 12–19 TON Timing Chart

Figure 12–20 shows how an OFF delay timer is programmed.

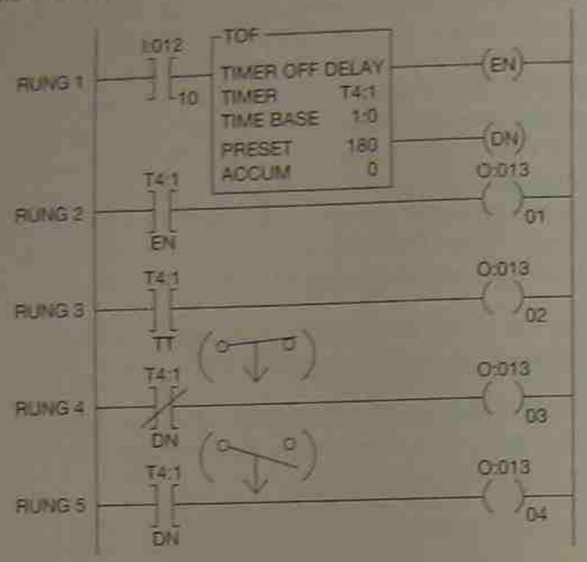


Figure 12–20 Programming an OFF delay Timer (TOF)

In an OFF delay timer, when bit I:012/10 is set to 1, the DN and EN bits are also set to 1. The DN bit acts like the OFF delay contacts of a pneumatic timer, and the EXAMINE OFF (N.C.) instruction in Rung 4 goes false, or open, while the EXAMINE ON instruction (N.O.) in Rung 5 goes true. When input device I:012/10 is reset, or set to 0, Rung 1 goes false, and the timer starts to accumulate time in one-second intervals as long as the Rung remains false. When the accumulated value equals the preset value (180) the timer stops. T4:1.TT was set to 1 while the timer was timing and output O:013/02 in Rung 3 was *ON*. When the accumulated value equaled the preset value and the timer stopped timing, the TT bit was reset to 0 and output O:013/02 was turned *OFF*. When the TT bit is reset to 0, the DN bit (bit 13) is also set to 0, and output O:013/03 in Rung 4 is turned *ON* and output O:013/04 in Rung 5 is turned *OFF*. The TOF instruction is reset by each open-to-closed transition of input device I:012/10. Figure 12–21 shows a typical timing chart for an OFF delay timer.

During the first timing cycle, the timer was only *OFF* for 120 seconds. That was not long enough for the timer to time out, so the outputs controlled by the DN bit did not change. During the second timing cycle, the timer was allowed to time out and the outputs changed states.

Most PLCs also offer a timer that replaces the standard motor-driven timer. A typical motor-driven timer consists of shaft mounted cam(s) that are driven by a synchronous motor. Rotating cam(s) activate (open or close) limit or micro switches. Once power is applied, the motor turns the shaft and cam(s). The positioning of the lobes of the cam(s) and the gear reduction of the motor determine the time it takes for the motor to turn the cam far enough to activate the switches. If power is removed from the motor, the shaft stops. When power is reapplied, the motor continues turning the shaft until the switches are activated. When the timing of a device is not reset due to a loss of power, the timing is said to be retentive.

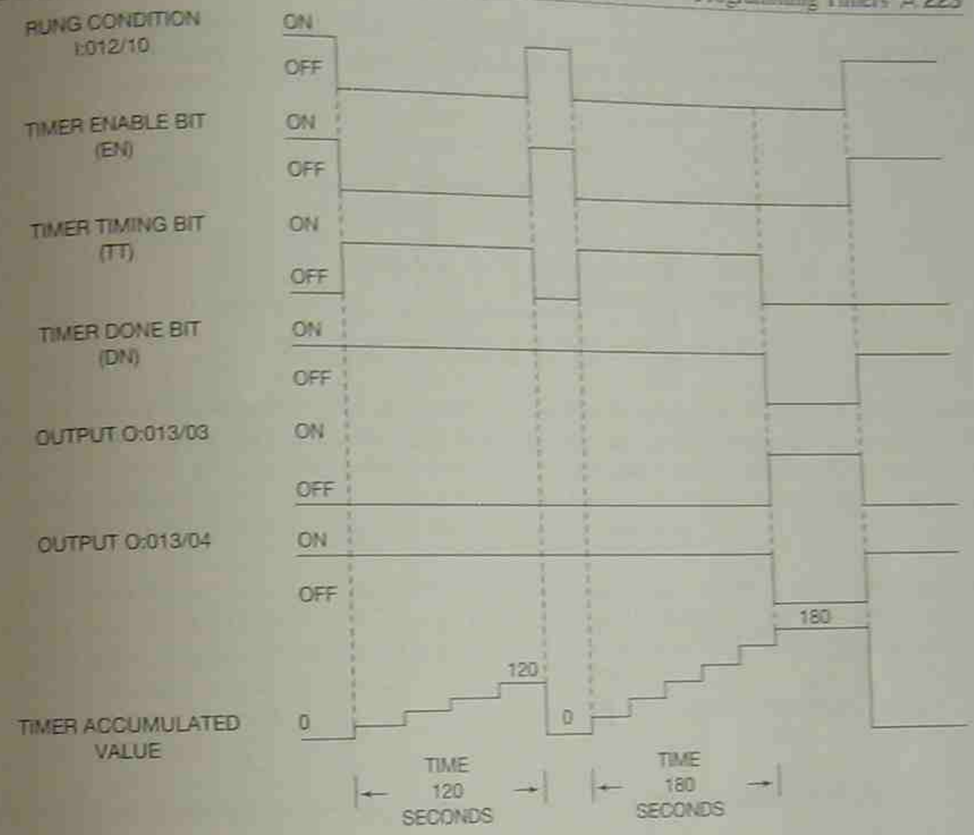


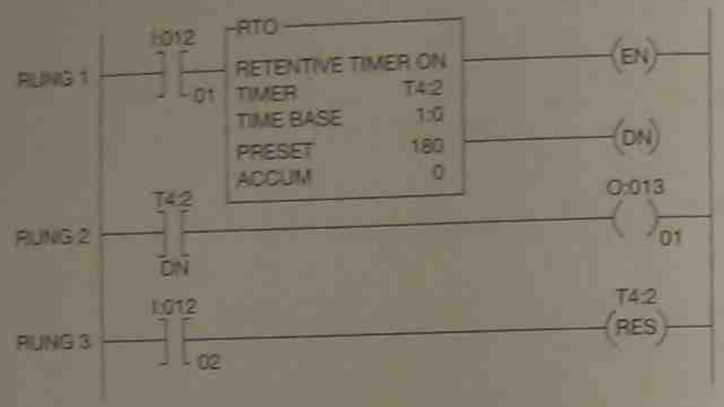
Figure 12–21 Timing Chart for a TOF Timer

Retentive timers (RTO) can be programmed to replace motor-driven timers.

The retentive timer lets the timer START and STOP without resetting the accumulated value to 0. The bits associated with the timer EN, TT and DN function the same as with the TON instruction. The RTO instruction begins timing when the Rung goes true. As long as the Rung remains true, the timer continues to time until the accumulated value reaches the preset value. If the timer Rung goes false, the timer holds the accumulated time, rather than resetting the accumulated value to 0. When the timing Rung goes true again, the count picks up from where it was, and continues to accumulate time. Once the accumulated time is equal to the preset time, the processor will set bit 13 (the DN bit) to 1. The DN bit remains *ON* (or set to 1) as long as the accumulated value is equal to the preset value. Because the retentive timer does not reset to 000 when the timer is deenergized, a reset Rung (RES) must be added. The reset instruction must be given the same address as the retentive timer it is intended to reset.

A common problem in programs that have retentive timers is that the timer is not accumulating time, even though the timer Rung is true. More often than not, the problem is a reset instruction

that is true which prevents the timer from timing. Figure 12–22 shows how the RTO timer is programmed, including the reset Rung (Rung 3), and shows a typical retentive timer timing chart.



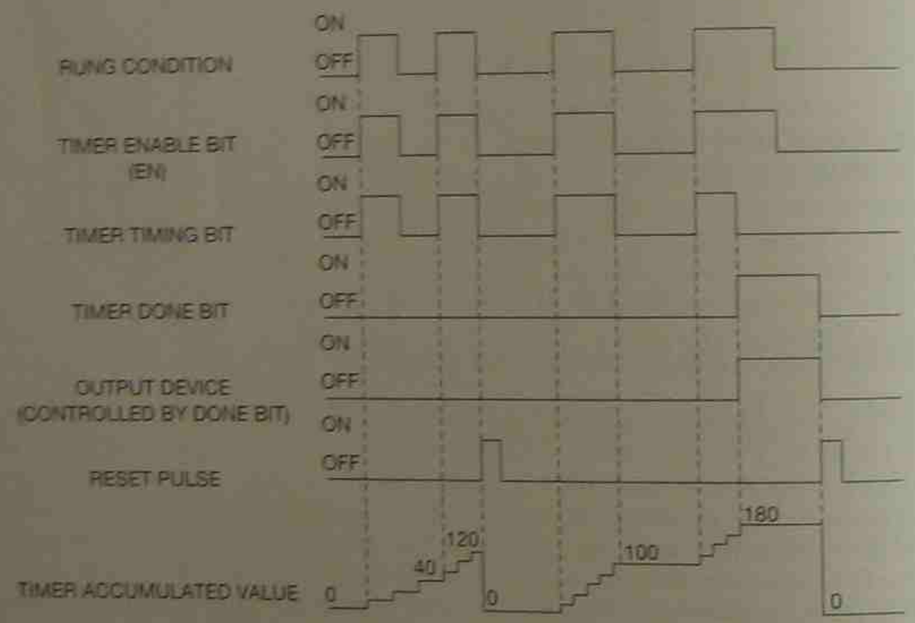


Figure 12–22 Programmed Retentive Timer (RTO) and Timing Chart

MODICON INC. TIMERS

Figure 12–23 shows the timer format typical for timers programmed for the Modicon 984 family of PLCs.

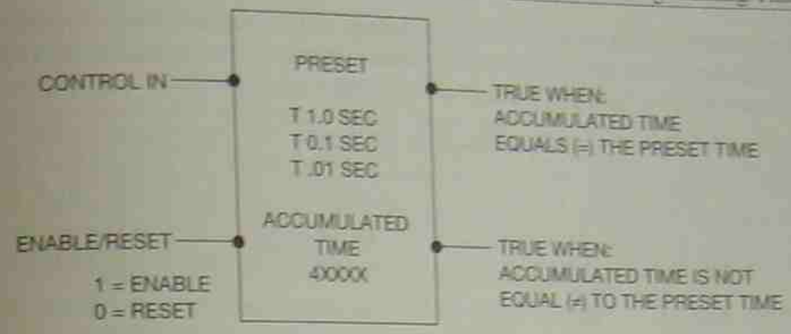


Figure 12–23 Modicon 984 Timer Format

The two lines, or nodes, to the left are control in and enable/reset. The Control In line controls when the timer times, and the Enable/Reset line resets the accumulated value of the timer to 000, or resets the timer. The timer is enabled when both the Control In and the Enable/Reset lines have power flow (are set to 1). The timer stops timing if the Control In line goes false, but the timer is not reset until the Enable/Reset line goes false, or has no power flow.

The timer box holds the preset time (1–999 for some models and 1–9999 on other models). The preset value for the timer is stored in an input register (3XXXX) or in a holding register (4XXXX). The time base is selectable for seconds, tenths of seconds, and hundredths of a second.

T 1.0 = seconds
T 0.1 = tenths of a second
T .01 = hundredths of a second

The timer box also displays the storage register that holds the current or accumulated time.

The two right-hand nodes, or lines, are output lines and provide power to contacts, coils, and the like. The top line provides power only when the timer's accumulated value is *equal* to the preset value; the bottom line provides power as long as the accumulated time is *less* than the preset time (not equal). The output on the bottom line only stops passing power, or is false, when the accumulated and preset values are the same (when the timer has timed out). Figure 12–24 shows an ON delay timer and how it is programmed to control outputs 00107 and 00108.

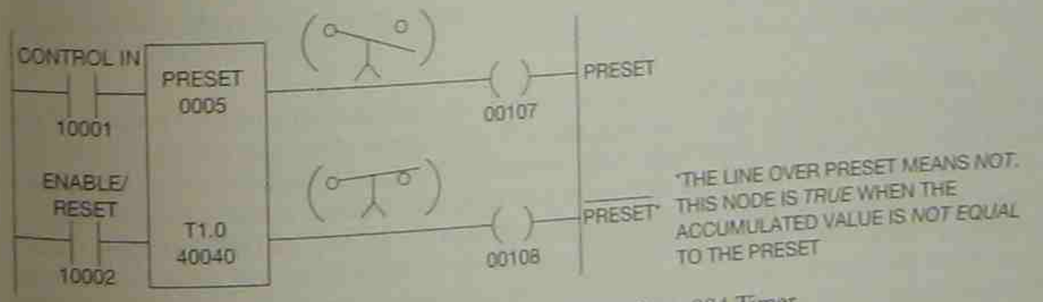


Figure 12–24 Programmed Modicon 984 Timer

If 10002 is closed, the timer is enabled, but not timing, and the value stored in register 40040 is 0. Since the value in register 40040 is 0, the accumulated value is not equal to the preset value of 00005, so output 00107 is false (*OFF*), but output 00108 is true (*ON*).

When 10001 in the control in line is closed, or goes true, the timer starts timing, and register 40040 will begin to accumulate time in one-second intervals. When the accumulated value is equal to the preset value (five), output 00107 becomes true, and output 00108 goes false. The status of the outputs remains the same until input 10002 in the enable/reset line is opened and resets the timer to zero. When the timer is reset, output 00107 is turned *OFF* and output 00108 is turned *ON*.

To duplicate the N.O.T.C. contacts of a pneumatic time delay, contacts with address 00107 should be used. For N.C.T.O. contacts, address 00108 is used.

The Gould 984, like many other PLCs, does not have a dedicated OFF delay instruction. To exactly duplicate the timing action of an OFF delay timer, additional Rungs of logic are required. Figure 12–25 shows the additional Rungs of logic required to obtain N.C.T.C. contacts and N.O.T.O. contacts. A timing chart is used to clarify the timing action.

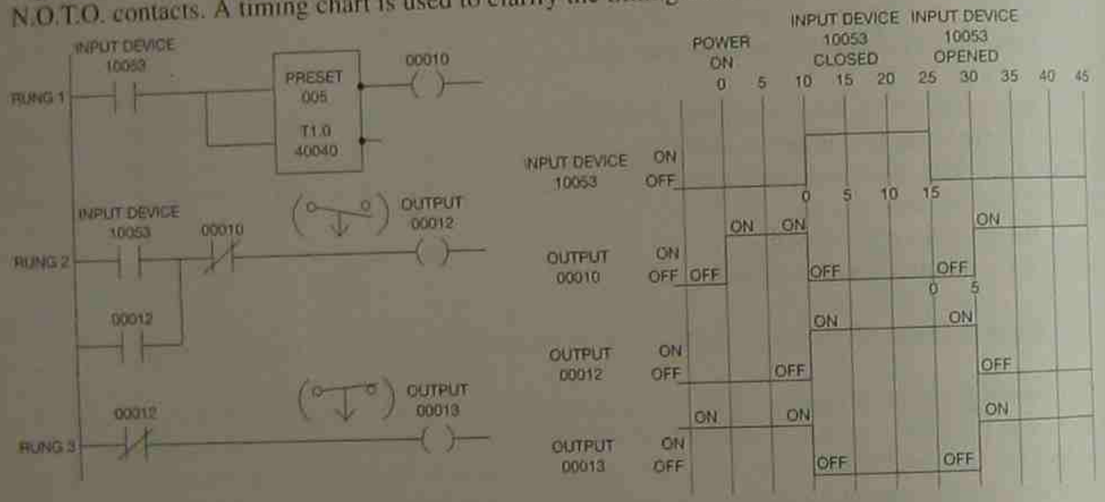


Figure 12–25 Programming a Gould 984 ON Delay Timer to Duplicate the Timing Action of an OFF Delay Timer

Notice that the enable/reset line has been tied into the control in line. This enables and turns the timer *ON* and *OFF* with just one input device (10053).

When power is first applied to the circuit, the timer is enabled and starts to time because of the EXAMINE OFF instruction 10053, which is the N.O. input device. Once the timer has timed out, output 00010 will be set to 1, or be turned ON. Output 00012 in Rung 2 cannot be energized because of the open 10053 input contacts, as well as the now open 00010 output contacts. Output 00013 in Rung 3 is energized as soon as the power is applied through the EXAMINE OFF (normally closed) contacts of output 00012. When the input device is closed, the timer Rung goes false and resets the timer. When the timer is reset, contacts 00010 in Rung 2 go back to their normally closed state. With the input device 10053 and contacts 00010 now closed, output 00012 energizes. When 00012 ener-

gizes, the EXAMINE ON contacts in Rung 2 close and act as holding contacts, while the EXAMINE OFF instruction in Rung 3 goes false and turns output 00013 *OFF*. The circuit remains in this condition until input device 10053 is opened, which activates the timer. When the accumulated time equals the preset time (value in register 40040 equals 0005), the timer will time out, making the EXAMINE OFF instruction in Rung 2 go false, which turns *OFF* output 00012, and causes output 00013 in Rung 3 to again be set to 1 (turn *ON*). This timing action exactly duplicates the timing action of an OFF delay timer. Additionally, normally open contacts (EXAMINE ON instructions) from output 00012 can be programmed for N.O.T.O. timer action, while normally closed contacts (EXAMINE OFF instructions) can be used for N.C.T.C. timer contacts.

For a retentive timer, a different input device is programmed in the control in and reset lines as indicated in Figure 12–26.

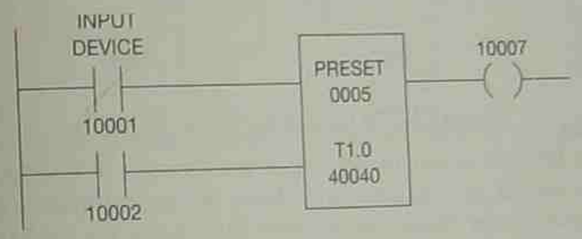


Figure 12–26 Retentive Timer

CASCADING TIMERS

When circuit requirements demand more time than is available from a single timer, two or more timers can be programmed together as shown in Figure 12–27a. Programming two or more timers together to extend the timing range is called **cascading**.

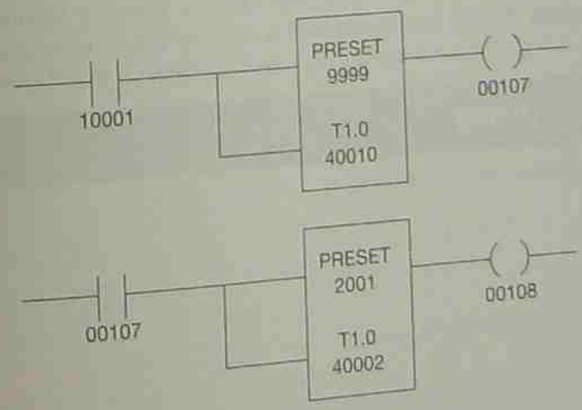


Figure 12–27a Cascading Timers

In this circuit, the first timer is controlled by input device 10001. When the device is true, the timer starts to time. When the accumulated time is equal to the preset time, output 00107 is set to 1, or *ON*. When 00107 is set to 1, the second timer is enabled and starts to time. When the second timer has timed out, output 00108 is turned *ON*. The total time to turn ON output 00108 after input 10001 was closed is 12,000 seconds (9999 + 2001), or 200 minutes.

A second, and perhaps easier method to cascade timers, is to put the two timers in series as shown in Figure 12–27b. This method only works with PLCs that allow timers to be placed in series, such as the Modicon 984. If the timer is considered an output instruction, like the Allen-Bradley PLC families, this method does not work, and the two-rung method shown in Figure 12–27a must be used.

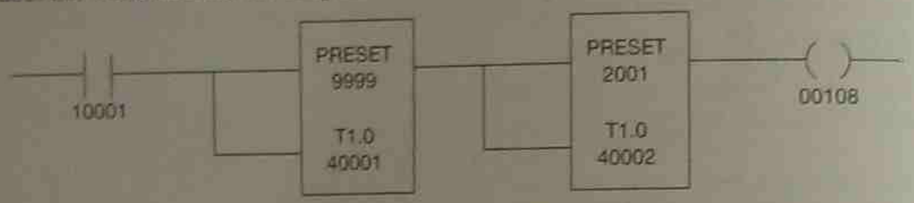


Figure 12–27b Cascading Timers in Series

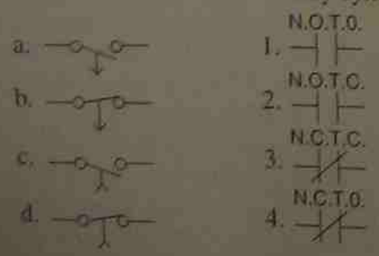
Once the input device 10001 is closed, the first timer starts to time. When the first timer times out, the second timer is activated and starts to time. When the second timer's accumulated value is equal to the preset value, output 00108 will be turned *ON*. Again, the total time to turn on output 00108 after input 10001 closes, is 12,000 seconds (9999 + 2001), or 200 minutes. By cascading timers, virtually any required time can be achieved.

Chapter Summary

Although the format is different for different PLCs, the basic principles are the same. Preset and accumulated times are stored and compared on each processor scan. When the accumulated value equals the preset value, discrete output devices or internal outputs can be turned *ON* or *OFF*. Timers can be programmed for *ON delay*, *OFF delay*, or as *retentive* timers. The only limit to the number of timed and instantaneous contacts that can be programmed is memory size. Programmed timers offer a wider range of time settings and greater accuracy than is possible with hard-wired pneumatic timers.

Review Questions

1. Match the standard time delay symbols.



2. The amount of time for which a timer is programmed is called the:
   a. preset
   b. set point
   c. desired Time (DT)
   d. all of the above
3. T F As scan time increases, so does the accuracy of any programmed timers.
4. When the timing of a device is not reset due to a loss of power, the timer is said to be:
   a. holding
   b. secured
   c. retentive
   d. continuous
5. When more time is needed than can be programmed with one timer, two or more timers can be programmed together. This programming technique is called:
   a. stacking
   b. cascading
   c. doubling
   d. synchronizing
6. When programming timers with Allen-Bradley format, which bit will act as an instantaneous contact?
   a. DN
   b. TT
   c. EN
   d. IN
7. When the accumulated time is equal to the preset time, which bit in the Allen-Bradley PLC-5 family will be true?
   a. DN
   b. TT
   c. EN
   d. IN
8. When programming a Modicon 984, which line is true when the accumulated time is *not* equal to the preset time?
   a. preset line
   b. *not* preset line

CHAPTER

13 Programming Counters

Objectives

After completing this chapter, you should have the knowledge to:
- Write a program using up and down counters.
- Define the terms *increment* and *decrement*.

Programmed counters serve the same function as the mechanical counters used in the past. Programmed counters can count up, count down, or be combined to count up and down. Counters are similar to timers, except they do not operate on an internal clock but instead are dependent on external or program sources for counting.

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX COUNTERS

Allen-Bradley offers two types of counters: up counters (CTU) and down counters (CTD). Both counters are retentive until reset by a reset instruction. Figure 13–1 is a typical Allen-Bradley up counter.

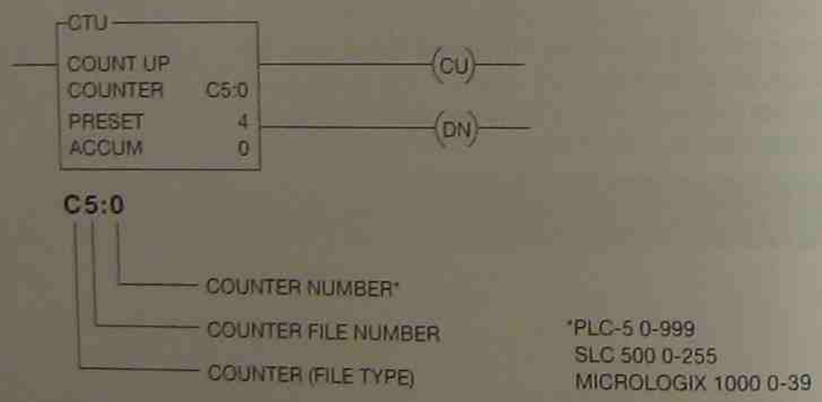


Figure 13–1 Allen-Bradley PLC-5 Counter Format

The Allen-Bradley up counter format is similar to the timer format. The up counter consists of a counter block that contains the up counter address, the preset value, and the accumulated count value which can

be any number from 0 to +32,767. The counter address consists of C for counter, the file number (5 [the default number]), a colon (:), and the counter number. File 5 is the default file from the data table for counters using the PLC-5, SLC 500, and the MicroLogix 1000. The PLC-5 can be programmed to use files 3-999 for additional counter files. The SLC 500 can be programmed to use files 9-255 for additional counter files. The MicroLogix is limited to one counter file, which is the default file, file 5.

By only having one counter file the MicroLogix 1000 is limited to 40 counters (counters 0 through 39), whereas the SLC 500 can use files 0 through 255 for counters. The PLC-5 can have counter numbers from 0 through 999. Each counter requires three words of memory, as shown in Figure 13–2.

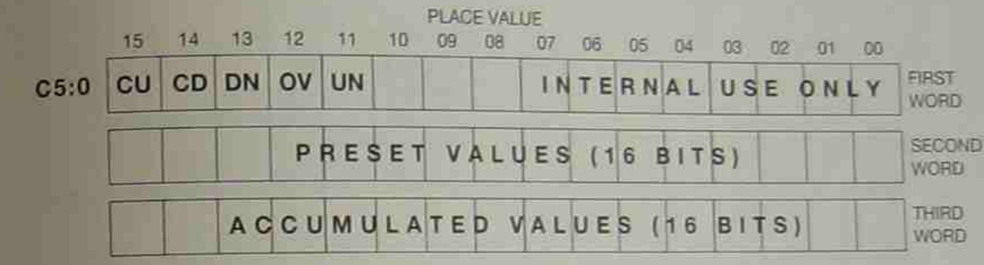


Figure 13–2 Storage Format for Counters

The first word stores the status bits of the counter (bits 11 through 15). The second word holds the preset values, or count, and can range from –32,768 to +32,767. The positive numbers are stored in 16-bit binary, while the negative numbers are stored in the 2s complement. The third word stores the accumulated count, and can be any number from –32,768 to +32,767. 2s complement is covered in Chapter 6.

The status bits for up and down counters that are stored in the first word are as follows:

Count Up Enable Bit (CU) The CU bit (bit 15) is set to 1, or is true, when the rung is true, and remains true as long as the up counter is enabled. The CU bit goes false when the counter is reset or the counter rung goes false. The CU bit is *only used* with up counters.

Count Down Enable Bit (CD) The CD bit (bit 14) is set to 1, or is true, when the rung is true, and remains true as long as the down counter is enabled. The CD bit goes false when the counter is reset or the counter rung goes false. The CD bit is *only used* with down counters.

Count Up Done Bit (DN) The DN bit (bit 13) is set to 1 when the accumulated count equals the preset count, and remains set to 1, or *ON*, as long as the accumulated value (count) is equal to or greater than the preset value.

Count Down Done Bit (DN) The DN bit (bit 13) is *ON*, or set to 1, as long as the accumulated value is equal to or greater than the preset value. The DN bit is only reset to 0, turned *OFF*, when the accumulated count is less than the preset value.

Count Up Overflow Bit (OV) The OV bit (bit 12) is set by the processor to 1, or ON, when the accumulated count exceeds the *upper limit* of (+)32,767. When this limit is reached, the count wraps around to (–)32,767, and the up counter increments from there.

Count Down Underflow Bit (UN) The UN bit (bit 11) is set by the processor to 1, or *ON*, when the accumulated count exceeds the *lower limit* of (–)32,768. It wraps around to (+)32,767, and the CTD instruction counts down from there.

Output O:0/1 is *ON* and output O:0/2 is *OFF* when O:0/0 is not latched.

Output O:0/1 goes *OFF* and O:0/2 comes *ON* when O:0/0 is latched.

Normally, an internal storage bit (dummy relay) is used for the latch and unlatch address, rather than an actual discrete output address. If a discrete output address is used, the output, once latched, remains *ON*, even if programmed after an open MCR Rung. When an internal storage bit is used for the latch and unlatch address, the bit is still retentive, but turns *OFF* if programmed *after* an MCR Rung that is open.

Although all PLCs are designed and manufactured to the highest standards and quality, a latching or MCR instruction should not be depended on for machine safety. A hardwired safety circuit should always be added. A safety circuit is recommended by most PLC manufacturers to ensure maximum safety rather than depending on a programmed MCR or latching relay alone.

SAFETY CIRCUIT

The concept of safety circuit has been discussed earlier in the text, and is an important enough subject to be covered again. The National Electrical Manufacturing Association (NEMA) standards for programmable controllers recommends that consideration be given to the use of emergency stop functions which are independent of the programmable controller. The standard reads in part:

> When the operator is exposed to the machinery, such as loading or unloading a machine tool, or where the machine cycles automatically, consideration should be given to the use of an electromechanical override or other redundant means, independent of the controller, for starting or interrupting the cycle.

Figure 11–7 shows how a control relay (CR) and *safe-run switch* is added to interrupt line2 to the discrete output devices of an automatic machine or process.

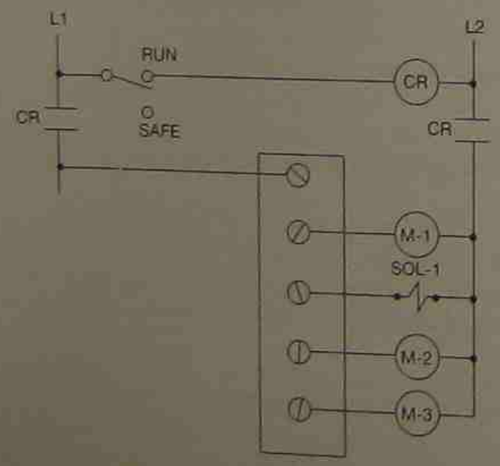


Figure 11–7 Safety Circuit

IMMEDIATE INPUT INSTRUCTION

The immediate input instruction is used when it becomes necessary to update the status of an input word in the input image table prior to the normal scan sequence. For some high-speed processes it may be necessary to know the status of a certain bit (or bits) in a particular input word prior to the next normal input image table update. Allen-Bradley refers to their immediate input instruction with the mnemonic IIN. To use the instruction to interrupt the normal scan cycle and update an input word, the IIN instruction is programmed prior to the rung(s) that include critical input addresses. The address for the IIN instruction is the rack number and module group for the input devices that need to be updated prior to the normal update. The rack number and the module group correspond to a word in the input image table. Figure 11–8 illustrates how the IIN instruction is programmed.

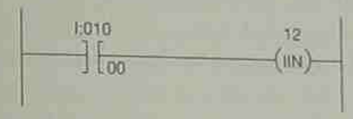


Figure 11–8 Programming an Allen-Bradley PLC-5 Immediate Input Instruction

If input I:010/00 is true, the program scan is interrupted while the input status for input image word 12 is updated (rack 1, module group 2). As soon as the input word has been updated, the scan continues. Figure 11–9 shows the typical scan cycle and how it is interrupted for the immediate update of the status of the input devices in rack 1, module group 2.

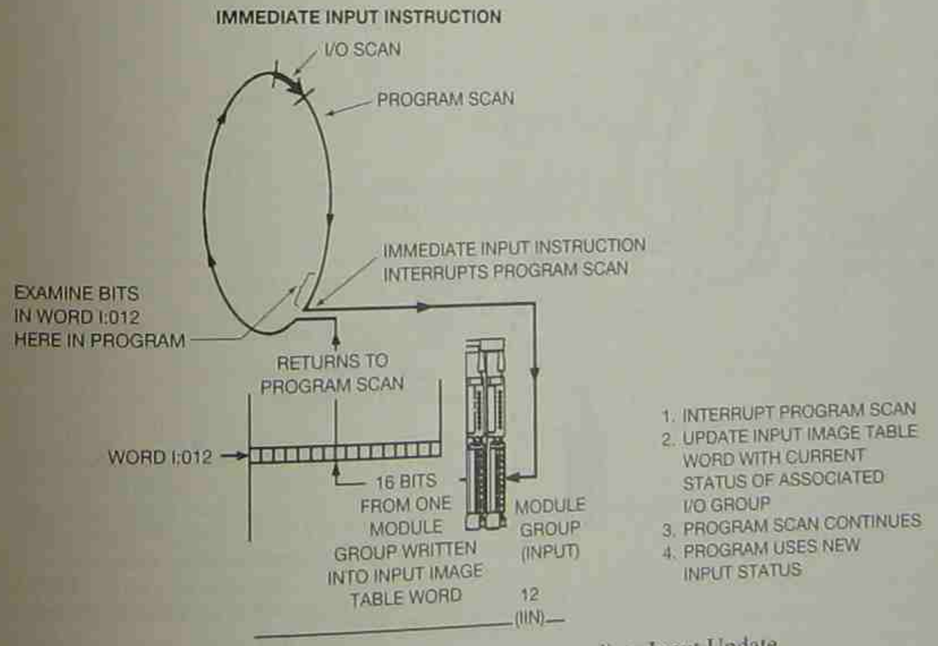


Figure 11–9 Scan Interrupted for Immediate Input Update
(Courtesy of Allen-Bradley)

IMMEDIATE OUTPUT INSTRUCTION

The Immediate Output instruction is used when it is necessary to update the output image table prior to the completion of a normal program scan. Allen-Bradley PLC-5 used the mnemonic IOT. When the immediate output instruction (IOT) is enabled, or true, the instruction will interrupt the normal program scan and update a word in the output image table. Similar to the IIN instruction, the rack number and module group number is used to identify the location of the output devices that are to be immediately updated. Figure 11–10 shows how the instruction is programmed for outputs in rack 1, module group 3 (output word 013).

Figure 11–10 Programming the Immediate Output Instruction

When input I:010/00 is true, the IOT instruction is enabled and the scan is interrupted so that the output devices in rack 1, module group 3 (output word 013) are immediately updated. Figure 11–11 shows the interrupted program scan for the IOT instruction.

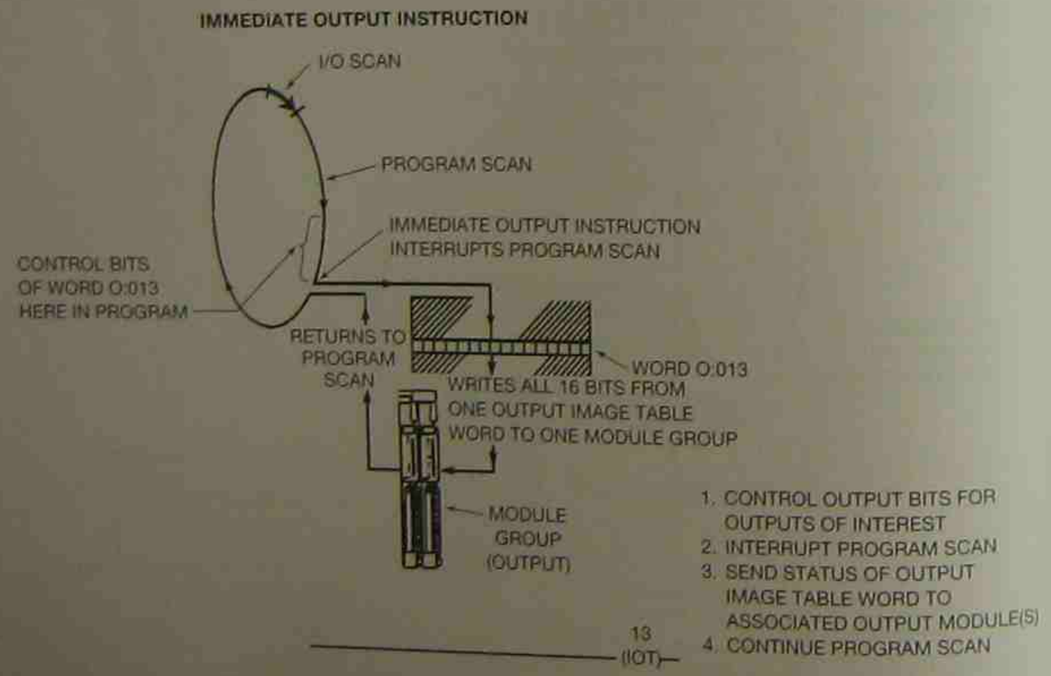


Figure 11–11 Scan Interrupted for Immediate Output Update
(Courtesy of Allen-Bradley)

JUMP AND LABEL INSTRUCTIONS

Used in combination, these two instructions allow for skipping over portions of the program to save program scan time. If there is a portion of the program that is not operational during certain portions of the process, the portion that is not used and/or needed can be jumped over or bypassed until it is needed again. By jumping over parts of the program, the scan time is decreased and more scans can be completed in a given period of time, which, in turn, means more frequent updating of information in the program. The jump instruction (JMP) tells the processor to jump over a portion of the program. Where to jump to is controlled by the label instruction (LBL). Figure 11–12 shows a jump and label instruction in Allen-Bradley PLC-5 format. When the JMP instruction is true, the processor will jump over Rungs 3 and 4 and go directly to Rung 5, as shown.

The JMP instruction is assigned a three-digit number from 000–255, and the rung that the processor is to jump to is given an LBL of the same number. In Figure 11–12 the JMP and LBL instruction is also enabled, and the processor is instructed to jump all successive rungs until it reaches the rung that contains the label instruction with the number 20. In this illustration, only two rungs are jumped. In actual practice, any number of rungs can be jumped.

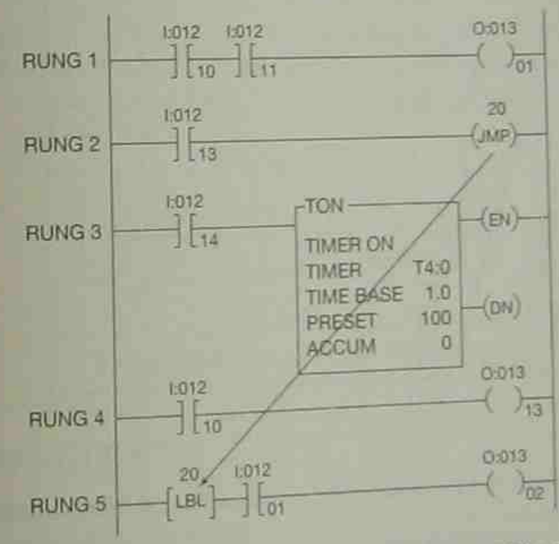


Figure 11–12 Programming the Jump (JMP) and Label (LBL) Instructions

The jump and label instruction can be used to jump forward or backwards in the program, depending on need. Jumping backward adds to the total scan time.

Note: *Jumping backward an excessive number of times could increase the scan time to a point where the watchdog timer will time out (the processor has a watchdog timer that is reset on each scan). If the scan time exceeds the watchdog timer's preset time, the processor goes into a* fault *condition.*

JUMP TO SUBROUTINE, SUBROUTINE, AND RETURN INSTRUCTIONS

The jump to subroutine, subroutine, and return instructions are used to direct the processor to a subroutine file within the program, scan it, return to the program, and continue to scan. The formats used by the various PLC manufacturers to program these instructions varies widely, and for this reason, only instruction blocks are shown in Figure 11–13. The blocks are identified using the Allen-Bradley mnemonics, or label: jump to subroutine (JSR); subroutine (SBR); and return (RET) as used with the SLC 500 and MicroLogix PLCs. Subroutines are very valuable for program organization, and for using blocks of programming logic over and over by simply changing the variables used in the subroutine. To use this group of instructions, consult the programming guide for the PLC system that is being used.

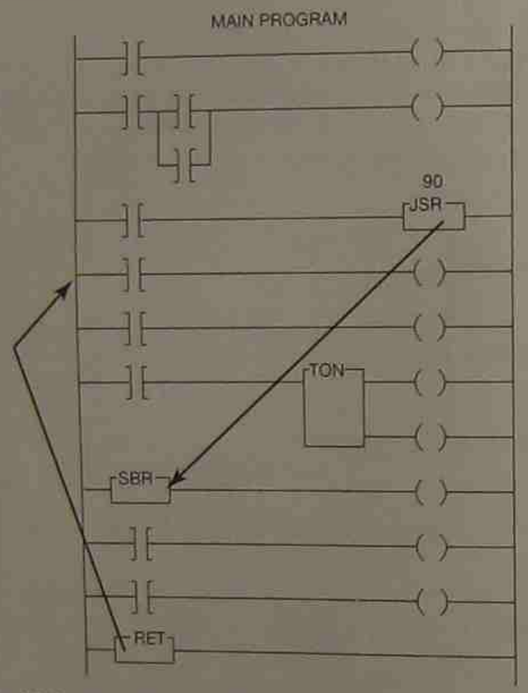


Figure 11–13 Jump to Subroutine, Subroutine, and Return Instructions

TEMPORARY END INSTRUCTION

The temporary end instruction (TND) is used to place a temporary end to the program scan. When the instruction is inserted into the program, the processor stops scanning the program at this point and returns to the start of the program. When the processor returns to the start of the program, the watchdog timer is reset to zero. This instruction is often used when a new program is being debugged for the first time, because it allows for portions of the program to be checked out without running the entire program. Figure 11–14 shows a TND using the Allen-Bradley format. When the TND instruction is true, the processor stops scanning at Rung 3 and does not scan Rungs 4 and 5. The TND instruction is also available with the SLC 500 and MicroLogix PLCs.

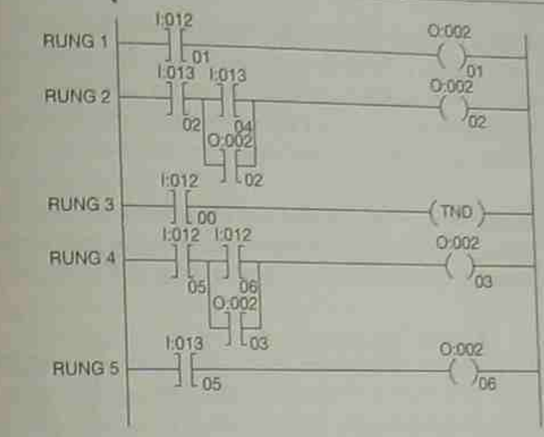


Figure 11–14 Allen-Bradley PLC—Temporary End Instruction (TND)

ALWAYS FALSE INSTRUCTION

The always false instruction (AFI) is also used when debugging a new or modified program. By inserting the always false instruction in a rung, the rung will always be false, regardless of the status of other instructions in the rung. Figure 11–15 shows a rung of logic with the AFI instruction programmed at the start of the rung.

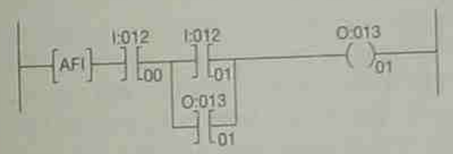


Figure 11–15 Always False Instruction

ONE SHOT INSTRUCTION

The one shot instruction (ONS) is an input instruction that makes the rung true for just one program scan, based on a false-to-true transition of the instruction that precedes the one shot instruction. Figure 11–16 shows a rung of logic with the one shot instruction programmed after input device I:011/04, which is controlling output O:010/12. Again, the Allen-Bradley PLC-5 format is used. This instruction is also used with the SLC 500 and MicroLogix PLCs.

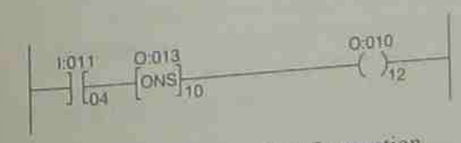


Figure 11–16 One Shot Instruction

When the rung is programmed as shown, the output is turned *ON* for one scan, and one scan *only* when input I:-11/04 closes (makes a false-to-true transition). The output cannot be turned *ON* again until the input device is first opened, then closed again, making a false-to-true transition. With the next false-to-true transition, the output device is again only turned on for one scan. This is a beneficial instruction when an output signal or operation is wanted for only one scan.

Math operations, data or word moves, and the like, are completed only once if a one shot instruction is put in series or "anded" with the instruction. The one shot instruction is often used with timers and counters for changing preset and accumulated values. This technique is discussed further in Chapter 14.

Note: *This chapter has covered some of the basic instructions that are available for programming with a PLC. As the instruction sets vary with each manufacturer, it is necessary that the programming manuals be consulted to determine what instructions are available, what their mnemonics or designations are, and how to properly use them in a program.*

Chapter Summary

Latching and master control instructions can be programmed to serve the same control functions as their real-world counterparts. Where personnel safety is a factor, a safety circuit should be added, instead of depending on latching or master control relay instructions alone. Through the use of various PLC instructions, the programmer can cause the processor to immediately update specific input and output devices, label and jump between specific rungs of logic, jump to subroutines then back to the original rung by using special instruction blocks, place a temporary end statement in the program to limit the amount of program that the processor will scan, use an always false instruction to keep a rung of logic from going true, and program one shot instructions that limit activity to only that one program scan. Although the mnemonics used by the various manufacturers will differ for each of their instruction sets or blocks, the main purpose of each instruction is the same. Once the electrician or technician has mastered programming one type of PLC, the transition to other types becomes easier.

Review Questions

1. Will both programmed and real-world latching relays, if latched, remain latched if power is lost and then restored?
2. Latching relays are normally used when it is necessary for:
   a. contacts to open and/or close only while the coil is energized.
   b. contacts to open and/or close every 30 seconds.
   c. contacts to stay open and/or closed even though the coil is only energized a short time.
   d. none of the above.
3. Explain why a *master control relay* is often used with off delay timers.

4. A master control relay can be used to control:
   a. selected circuit rungs (networks).
   b. entire circuits.
   c. individual contacts within a rung (network).
   d. all of the above.
5. Define the term *unconditional*.
6. When using PLCs, the National Electrical Manufacturing Association (NEMA) recommends that consideration be given to stop functions independent of the PLC. Explain briefly why this recommendation is made.
7. Define the term *retentive*.
8. Give an example of how an *immediate input instruction* is used.
9. What is the primary function of an *immediate output instruction*?
10. What does the *jump and label instruction* do?
11. Give one reason why you might use a *temporary end instruction*.
12. What is the function of the *always false instruction*?
13. What is the function of a *one shot instruction*?

CHAPTER

12 Programming Timers

Objectives

After completing this chapter, you should have the knowledge to:

- Describe how *pneumatic time delay relays* work.
- Write a program using *ON delay* and *OFF delay* timers.
- Describe the difference between an *ON delay timer and a retentive timer*.
- Explain how to extend the time range of timers by *cascading*.

PNEUMATIC TIMERS (GENERAL)

To fully understand how a PLC can be programmed to replace pneumatic time delay relays, both the basic pneumatic time delay relay and the standard symbols used must be understood.

Figure 12–1 shows a complete Allen-Bradley pneumatic timing relay, and Figure 12–2 shows a cutaway view of the contact and timing mechanism.

Figure 12–1 Pneumatic Timing Relay
(Courtesy of Allen-Bradley)

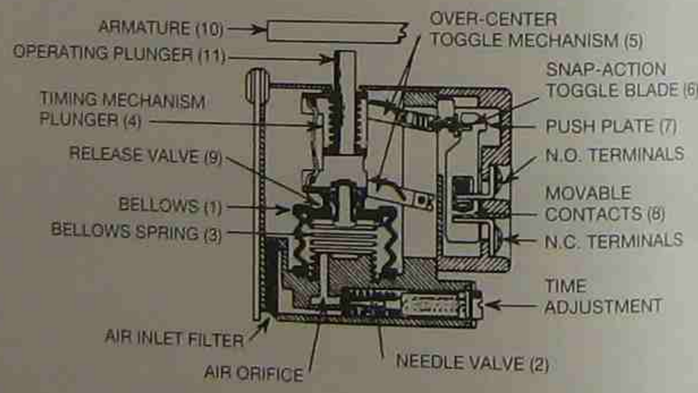


Figure 12–2 Cutaway View of Contact Unit and Timing Mechanism
(Courtesy of Allen-Bradley)

For the timer to time when power is applied (coil energized), the solenoid unit—coil, core piece, and armature—is mounted so that the natural weight of the armature (10) pushes down on the operating plunger (11). This causes the bellows (1) and bellows spring (3) to collapse the bellows, and dispel the air out through the release valve (9). When the coil is energized, the armature is attracted magnetically to the pole pieces, and lifts up and off the bellows assembly. Air now comes in through the air inlet filter, passed the needle valve (2), and fills the bellows with air. The incoming air expands the bellows upward, pushing on the timing mechanism plunger (4). As the plunger rises, it causes the over-center toggle mechanism (5) to move the snap-action toggle blade (6) upward. This picks up the push plate (7) that carries the movable contacts (8) to open the N.C. contact and close the N.O. contact. The time it takes for the bellows to fill with air and activate the contact mechanism is controlled by adjusting the needle valve in the air orifice. The valve is adjusted with a screwdriver as shown in Figure 12–3. A counterclockwise rotation moves the needle valve further into the air orifice, restricting airflow into the bellows, slowing the airflow, and increasing the time it takes for the bellows to expand and operate the contact mechanism. Conversely, clockwise adjustment of the needle valve decreases the time it takes the bellows to fill with air and activate the contacts after the armature has been lifted off the bellows mechanism.

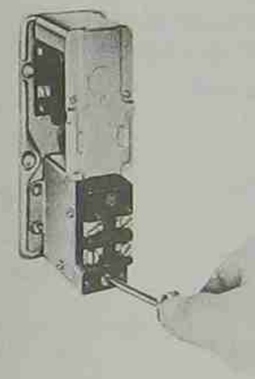


Figure 12–3 Pneumatic Timer Adjustment
(Courtesy of Allen-Bradley)

When the contact action is delayed after the coil has been energized and the armature is lifted up and off the bellows mechanism, it is called ON delay.

When the coil of an **ON delay timer** is deenergized, the armature drops down, pushing on the operating plunger, which in turn pushes down on the bellows expelling air through the release valve. The downward motion of the bellows causes the snap-action toggle blade to instantaneously snap the N.C. contact closed and the N.O. contact open.

To summarize the ON delay timer, the delay in contact operation begins *after* the timer coil has been energized, or turned *ON*. When the timer coil is deenergized, or turned *OFF*, the contacts go back to their normal condition instantly. Figure 12–4 shows a pneumatic timer with the solenoid unit mounted for ON delay.

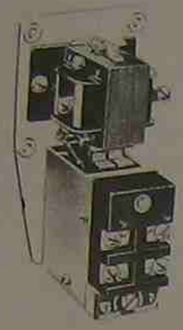


Figure 12–4 OM Delay Timer
(Courtesy of Allen-Bradley)

Figure 12–5 illustrates the electrical symbols used to indicate ON delay contacts.

The arrowhead indicates that movement is up. Since ON delay contacts can only time *after* the armature has lifted up off the bellows, this method of identifying timed contacts is easy to remember. Another common method of identifying timed contacts is shown in Figure 12–6.

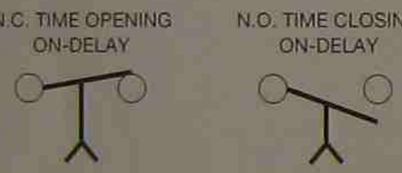


Figure 12–5 ON Delay Symbols

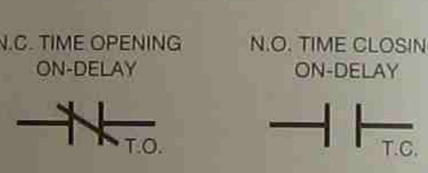


Figure 12–6 ON Delay Symbols

Note: *Remember that normal for contacts is how they are open or closed, with the coil of the relay deenergized and time expired.*

For a pneumatic timer to time when power is removed from the relay coil (**OFF delay**), the solenoid unit is mounted as shown in Figure 12–7. With a spring holding the armature up, no weight is applied to the bellows assembly, and the bellows are filled with air in a fully extended position.

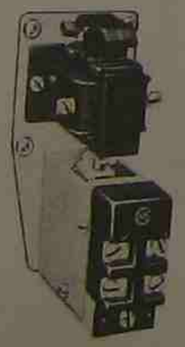


Figure 12–7 OFF Delay Timer *(Courtesy of Allen-Bradley.)*

Note: *Compare Figure 12–4 (the ON delay) with Figure 12–7 (the OFF delay) to clearly see the difference in the mounting of the solenoid assemblies.*

When the coil is energized and the armature moves down, the armature pushes on the operating plunger. The plunger pushes on the bellows assembly, and all air is immediately forced out of the bellows through the release valve. This causes the snap-action contact assembly to instantly open the N.C. contact and close the N.O. contact. The contacts stay in this configuration as long as the coil is energized and the armature is holding the bellows mechanism down (compressed).

When the relay coil is deenergized, or turned *OFF*, the spring on the armature lifts it up and off the operating plunger, which allows the bellows to start to fill with air. The N.C. contact remains open, and the N.O. contact remains closed until the bellows are filled with enough air to activate the snap-action contact mechanism. When the contact mechanism has been activated, the N.C. contacts go closed and the N.O. contacts go open.

Figure 12–8 shows the electrical symbols for OFF delay contacts.

To avoid confusion when reading electrical drawings with OFF delay contacts, it must be remembered that normal refers to the coil *after* it has been deenergized (turned *OFF*), and the time set for the timer has elapsed. The other symbols used for OFF delay contacts are shown in Figure 12–9.

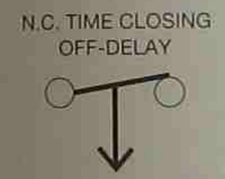


Figure 12–8 OFF Delay Symbols

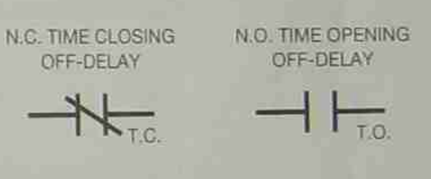


Figure 12–9 OFF Delay Symbols

Figure 12–10 compares both types of symbols used for ON delay and OFF delay timer relays.

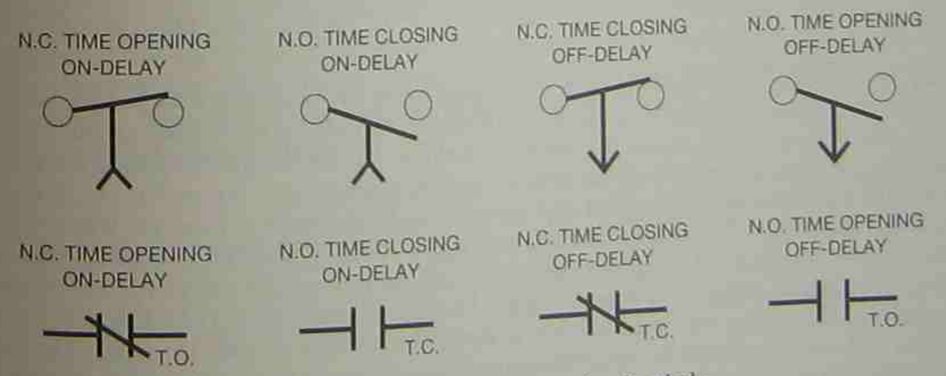


Figure 12–10 ON and OFF Delay Symbols

Reviewing the two types of symbols commonly used in motor control diagrams, an electrician or technician should have no trouble determining the type of timing relay (ON delay or OFF delay) used, or what is normal (open or closed) for the timed contacts.

The basic pneumatic timing relay is designed so that additional instantaneous contacts may be added, as shown in Figure 12–11. The instantaneous contacts operate when the coil is energized or deenergized independent of the timing mechanism. Figure 12–12 shows the electrical symbol for contacts with an asterisk (*) which is sometimes used to indicate instantaneous contacts of a timing relay.

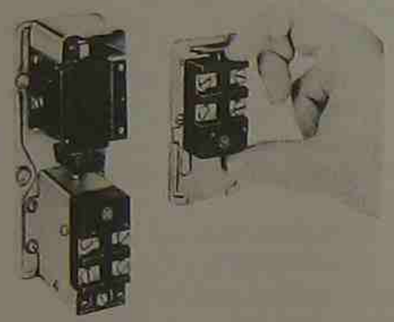


Figure 12–11 Adding Instantaneous Contacts
(Courtesy of Allen-Bradley.)

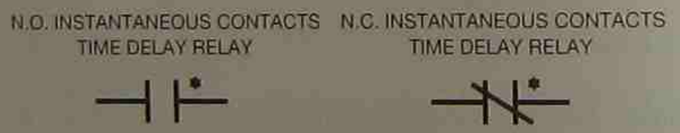


Figure 12–12 Instantaneous Contact Symbols

Figure 12–13a shows a simple light circuit controlled by an ON delay timer set for five seconds. The amount of delay is written near the timer coil on the diagram for understanding and for troubleshooting. Figure 12–13b shows that when S^1 is closed, the coil of the pneumatic timer energizes, lifts the armature up and off the bellows, and the timing starts. Figure 12–13c shows the circuit after three seconds have elapsed (not enough time for the timer to time out) with the lamp circuit still open. After five seconds have elapsed (Figure 12–13d), the N.O. time closing contacts close, and the lamp lights. As long as S^1 remains closed, the timer coil is energized, and the timed contacts stay closed. When S^1 is opened (Figure 12–13e), the coil circuit is broken, and the coil deenergizes. This causes the timed contacts to open, thereby turning OFF the lamp. The timed contacts will open the instant the coil deenergizes because they are timed only when power is applied to the coil.

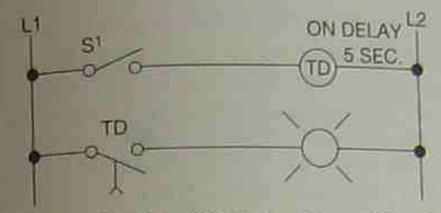


Figure 12–13a ON Delay Timer Circuit

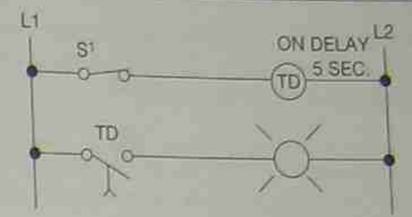


Figure 12–13b Instant S^1 is Closed

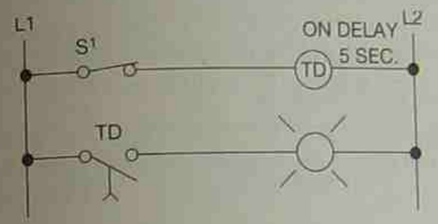


Figure 12–13c Three Seconds After S^1 is Closed

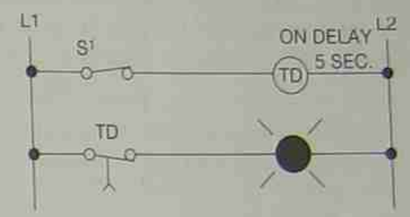


Figure 12–13d Five Seconds After S^1 is Closed

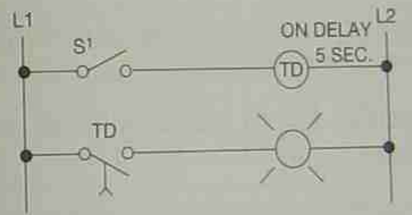


Figure 12–13e Instant S^1 is Opened

Figure 12–14a shows the same circuit but with an OFF delay timer. When S^1 is closed (Figure 12–14b), the TD coil energizes, drawing the armature down and compressing the bellows. This causes the N.O. OFF delay contacts to go closed instantly, and the lamp lights. When S^1 is opened (Figure 12–14c), the TD coil is deenergized, the spring-loaded armature is lifted up and off the bellows, and the five-second timing begins. Figure 12–14d shows the circuit after three seconds have elapsed. The lamp remains energized until the full five seconds have elapsed, and the N.O. contacts time out and open (Figure 12–14e).

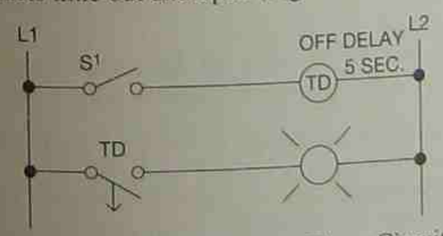


Figure 12–14a OFF Delay Timer Circuit

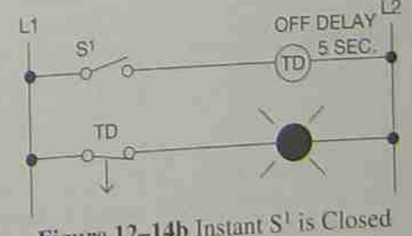


Figure 12–14b Instant S^1 is Closed

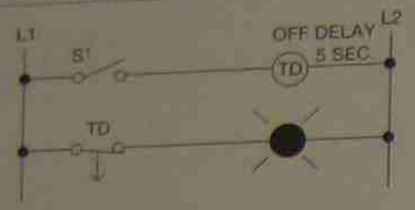


Figure 12–14c Instant S^1 is Opened

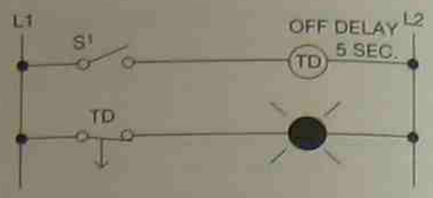


Figure 12–14d Three Secons After S^1 is Opened

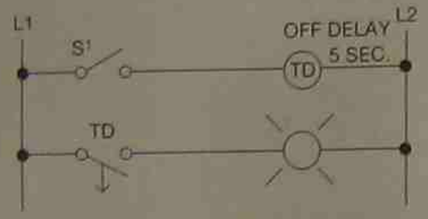


Figure 12–14e Five Seconds After S^1 is Opened

Instead of the bellows assembly, like the pneumatic time delay relay, PLC timers use internal solid state circuitry (clocks) for timing intervals or time base.

The various PLC manufacturers use varying approaches for the actual programming of timers. Several methods which are typical for most PLCs will be discussed.

Because it is an easy transition from pneumatic timer concepts to programming concepts, the Allen-Bradley approach to programming timers is discussed first.

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX TIMERS

Figure 12–15 shows the timer format used by Allen-Bradley. The timer consists of a timing block containing the timer number (address), time base (1 second or 0.01 seconds), and the preset and

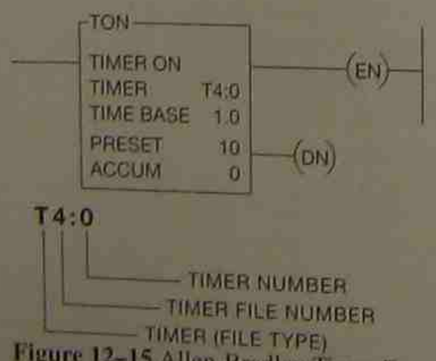


Figure 12–15 Allen-Bradley Timer Format

accumulated times. The preset time can be programmed with any value from 0 to 32,767. If a time base of one second was assigned, 32,767 would equal 9.1 hours (32,767 ÷ [60 × 60] = 9.1); if a time base of 0.01 (one hundredth of a second) was assigned, 32,767 would equal approximately 5.5 minutes (32,767 × 0.01 ÷ 60 = 5.46).

The two lines to the right of the block are the enable (EN) bit and done (DN) bit that indicate the status of the timer. The timer address (T4:0) identifies the timer file number and timer number. T4:0 indicates timer file 4, timer 0.

File 4 is the default file from the data table for timers using the PLC-5, SLC 500, and the MicroLogix family. The PLC-5 can be programmed to use files 3-999 for additional timer files. The SLC 500 can be programmed to use files 9-255 for additional timer files. The MicroLogix 1000 is limited to one timer file which is the default file, file 4.

By only having one timer file, the MicroLogix 1000 is limited to 40 timers (timer files 0 through 39) whereas the SLC 500 can use files 0 through 255. The PLC-5 can have timer numbers from 0 through 999.

The EN bit is set to 1 (or is true) whenever there is a logic path to the timer block. The DN bit is set to 1 (or is true) when the accumulated value equals the preset value, and the timer has timed out. Figure 12–16 shows how the information for a timer is stored. Three words of memory are used for each timer programmed. The first word of memory uses the first 8 bits for internal use and uses bit 13 for the DN bit, bit 14 for the timer timing bit (TT), and bit 15 for the EN bit. The next two words store the preset and accumulated values of the timer.

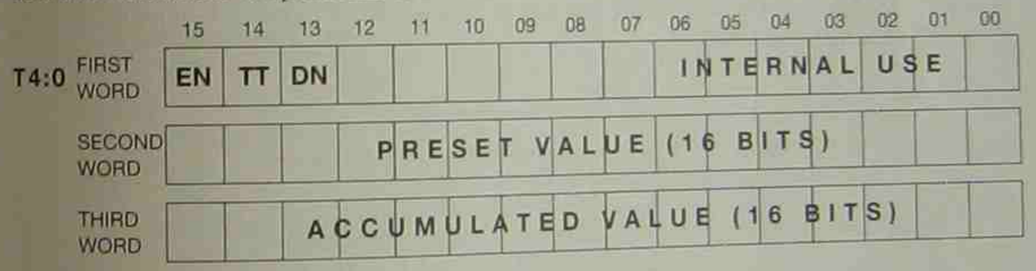


Figure 12–16 Timer Storage Format

Note: *There is no need to remember or memorize the timer bit numbers because the programming software accepts the mnemonics DN, TT, and EN. When addressing timer contacts, the timer number is entered first, followed by the timer bit. For example, T4:1/TT or T4:1.TT addresses the TT bit of timer 1, file 4.*

The timer enable bit, bit 15, is set to 1, or turned ON, when the rung goes true, and remains set until the rung goes false or a reset instruction resets the timer.

Note: *The EN bit can be used as an instantaneous contact.*

The TT bit, bit 14, is set to 1, or turned ON, when the rung goes true, and remains *ON* until: the rung goes false; the DN bit is set to 1 (accumulated value = preset value); timing is completed; or a reset instruction resets the timer.

Note: *The TT bit can be used to control a timer timing light that is only* ON *when the timer is actually timing. Figure 12–17a shows how the TT bit is used to control an indicator light, and Figure 12–17b shows the equivalent circuit using a pneumatic timer.*

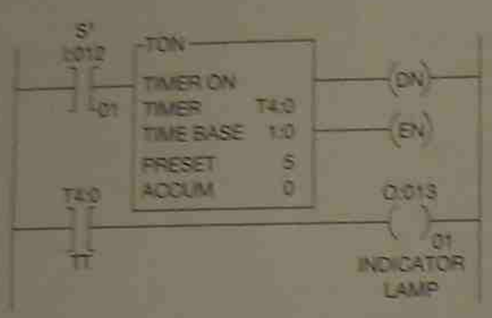


Figure 12–17a TT Bit Used to Control an Indicator Lamp

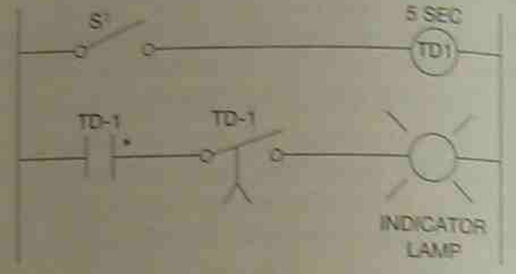


Figure 12–17b Pneumatic Timer Circuit to Control an Indicator Lamp

The DN bit, bit 13, is set to 1 when the accumulated value is equal to the preset value. The DN bit remains set to 1, or *ON*, until the rung goes false or a reset instruction resets the timer.

Note: *The DN bit can be used to control an output, or for other logic within a program.*

Figure 12–18 shows an ON delay timer and how it is programmed to control outputs O:013/01, O:013/02, O:013/03, and O:013/04.

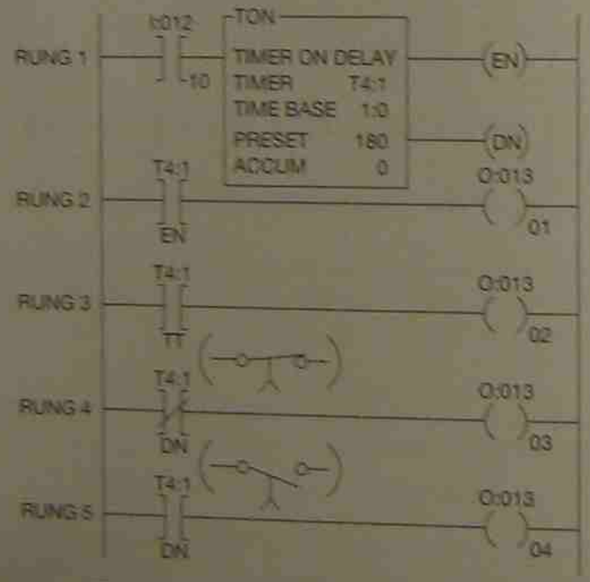


Figure 12–18 Programmed TON Timer

When bit I:012/10 (input device) is true, or set to 1, the timer rung is true, and the processor starts timer T4:0 timing and sets the EN and TT bits to 1. This turns *ON* outputs O:013/01 and O:013/02 in Rungs 2 and 3. The accumulated value increases in one-second intervals. The output in Rung 4, controlled by an EXAMINE OFF instruction, is true as long as the preset is not equal to the accumulated value. The EXAMINE OFF instruction addressed with the timer DN bit acts like a normally closed-time opening contact, and does not open until the accumulated value equals the preset value. The EXAMINE ON instruction in Rung 5 with the DN bit address acts like a normally open-time closing timer contact, and does not close (or go true) until the accumulated value is equal to the preset. When the accumulated time does equal the preset time, the DN bit is set to 1, and output O:013/04 is turned *ON* and output O:013/03 is turned *OFF*. Once the timer instruction has completed timing, the TT bit is reset to 0 and the output (O:013/02) of Rung 3 is turned *OFF*.

Like a pneumatic ON delay timer, when power is removed, the timer is reset to 0. The PLC-5 timer instruction is reset when the input device (I:012/10) is opened.

Figure 12–19 shows a typical timing chart. Notice that when the Rung condition is true (*ON*), the timer will time, but if the Rung goes false (*OFF*), the timer resets to 0, as illustrated, during the first two minutes of the timing diagram.

Figure 12–19 TON Timing Chart

Figure 12–20 shows how an OFF delay timer is programmed.

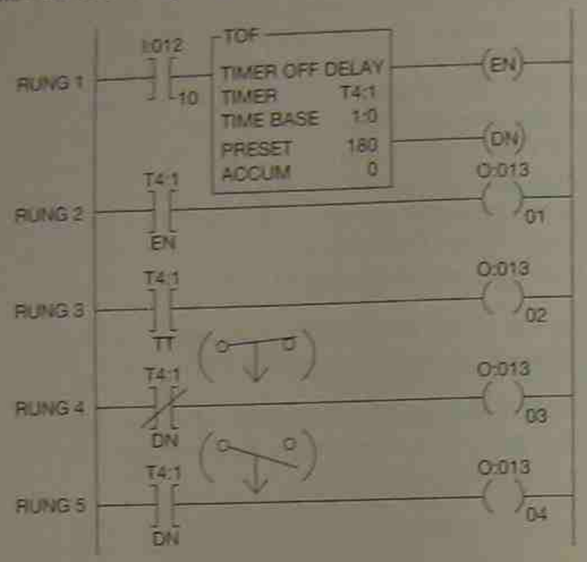


Figure 12–20 Programming an OFF delay Timer (TOF)

In an OFF delay timer, when bit I:012/10 is set to 1, the DN and EN bits are also set to 1. The DN bit acts like the OFF delay contacts of a pneumatic timer, and the EXAMINE OFF (N.C.) instruction in Rung 4 goes false, or open, while the EXAMINE ON instruction (N.O.) in Rung 5 goes true. When input device I:012/10 is reset, or set to 0, Rung 1 goes false, and the timer starts to accumulate time in one-second intervals as long as the Rung remains false. When the accumulated value equals the preset value (180) the timer stops. T4:1.TT was set to 1 while the timer was timing and output O:013/02 in Rung 3 was *ON*. When the accumulated value equaled the preset value and the timer stopped timing, the TT bit was reset to 0 and output O:013/02 was turned *OFF*. When the TT bit is reset to 0, the DN bit (bit 13) is also set to 0, and output O:013/03 in Rung 4 is turned *ON* and output O:013/04 in Rung 5 is turned *OFF*. The TOF instruction is reset by each open-to-closed transition of input device I:012/10. Figure 12–21 shows a typical timing chart for an OFF delay timer.

During the first timing cycle, the timer was only *OFF* for 120 seconds. That was not long enough for the timer to time out, so the outputs controlled by the DN bit did not change. During the second timing cycle, the timer was allowed to time out and the outputs changed states.

Most PLCs also offer a timer that replaces the standard motor-driven timer. A typical motor-driven timer consists of shaft mounted cam(s) that are driven by a synchronous motor. Rotating cam(s) activate (open or close) limit or micro switches. Once power is applied, the motor turns the shaft and cam(s). The positioning of the lobes of the cam(s) and the gear reduction of the motor determine the time it takes for the motor to turn the cam far enough to activate the switches. If power is removed from the motor, the shaft stops. When power is reapplied, the motor continues turning the shaft until the switches are activated. When the timing of a device is not reset due to a loss of power, the timing is said to be retentive.

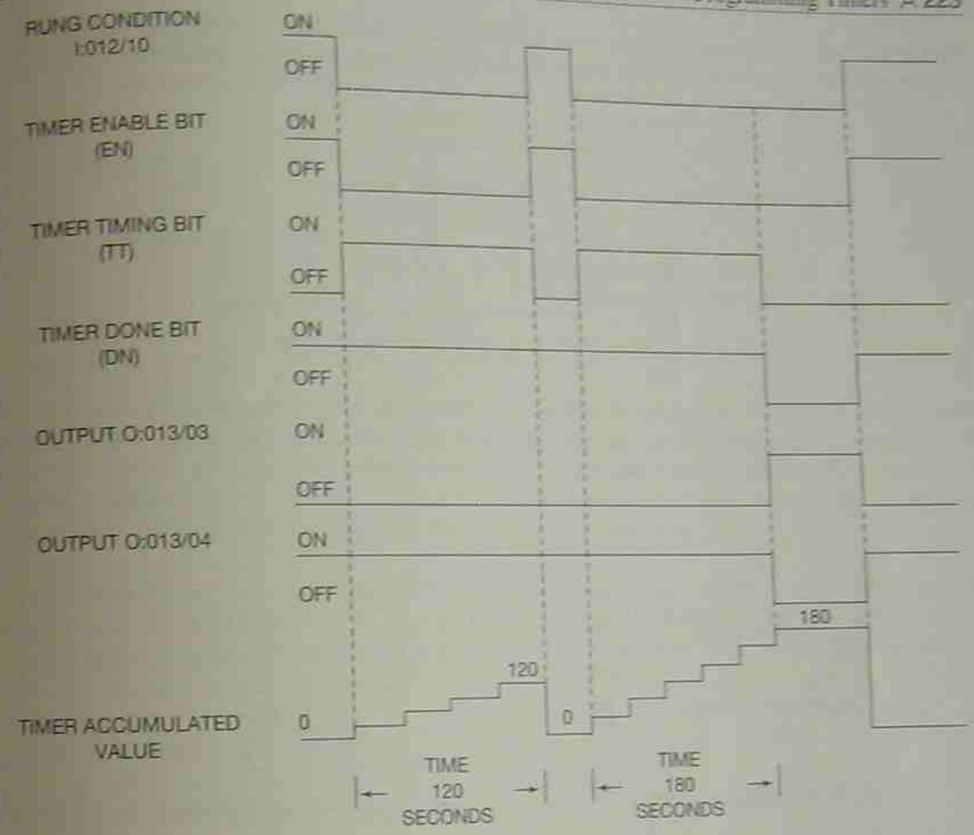


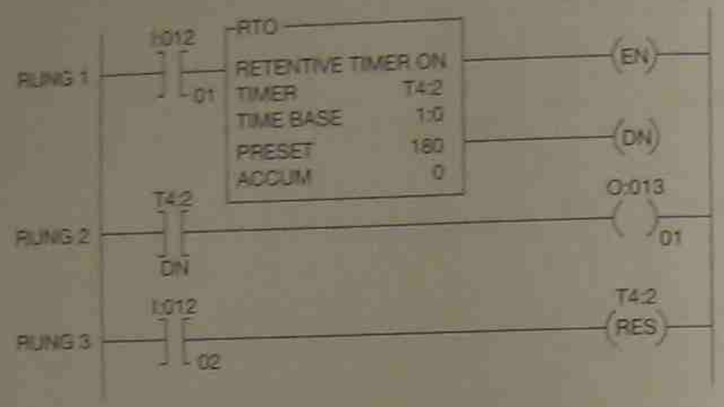
Figure 12–21 Timing Chart for a TOF Timer

Retentive timers (RTO) can be programmed to replace motor-driven timers.

The retentive timer lets the timer START and STOP without resetting the accumulated value to 0. The bits associated with the timer EN, TT and DN function the same as with the TON instruction. The RTO instruction begins timing when the Rung goes true. As long as the Rung remains true, the timer continues to time until the accumulated value reaches the preset value. If the timer Rung goes false, the timer holds the accumulated time, rather than resetting the accumulated value to 0. When the timing Rung goes true again, the count picks up from where it was, and continues to accumulate time. Once the accumulated time is equal to the preset time, the processor will set bit 13 (the DN bit) to 1. The DN bit remains *ON* (or set to 1) as long as the accumulated value is equal to the preset value. Because the retentive timer does not reset to 000 when the timer is deenergized, a reset Rung (RES) must be added. The reset instruction must be given the same address as the retentive timer it is intended to reset.

A common problem in programs that have retentive timers is that the timer is not accumulating time, even though the timer Rung is true. More often than not, the problem is a reset instruction

that is true which prevents the timer from timing. Figure 12–22 shows how the RTO timer is programmed, including the reset Rung (Rung 3), and shows a typical retentive timer timing chart.



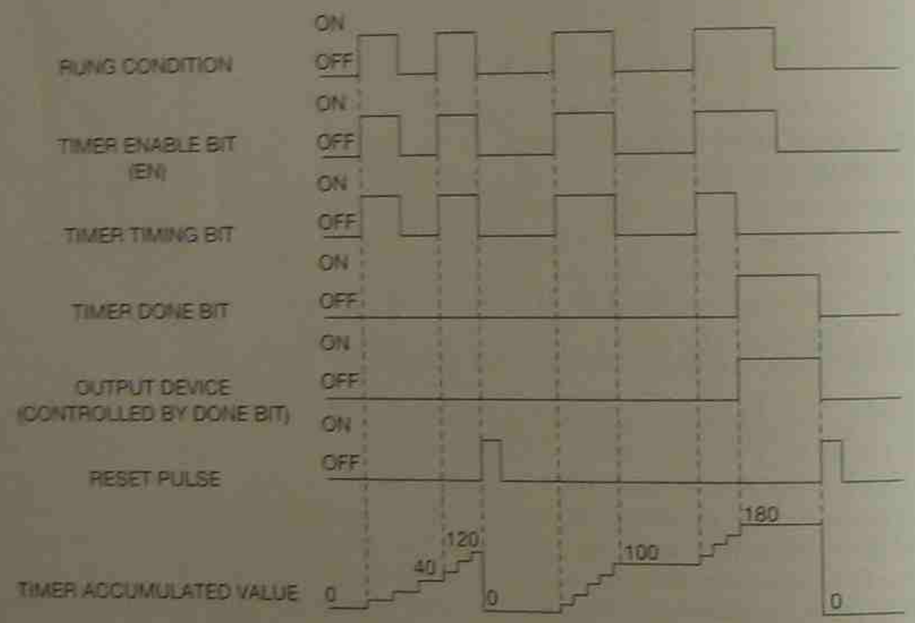


Figure 12–22 Programmed Retentive Timer (RTO) and Timing Chart

MODICON INC. TIMERS

Figure 12–23 shows the timer format typical for timers programmed for the Modicon 984 family of PLCs.

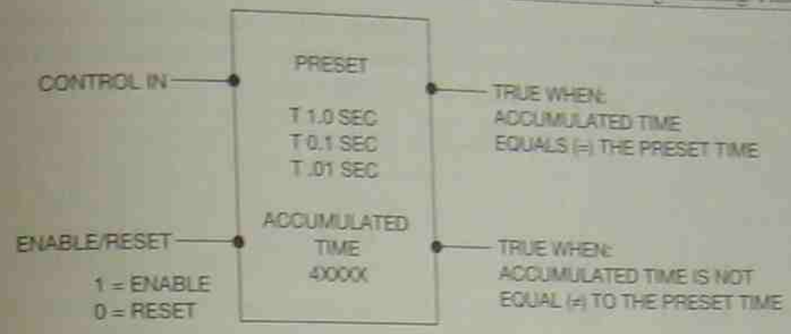


Figure 12–23 Modicon 984 Timer Format

The two lines, or nodes, to the left are control in and enable/reset. The Control In line controls when the timer times, and the Enable/Reset line resets the accumulated value of the timer to 000, or resets the timer. The timer is enabled when both the Control In and the Enable/Reset lines have power flow (are set to 1). The timer stops timing if the Control In line goes false, but the timer is not reset until the Enable/Reset line goes false, or has no power flow.

The timer box holds the preset time (1–999 for some models and 1–9999 on other models). The preset value for the timer is stored in an input register (3XXXX) or in a holding register (4XXXX). The time base is selectable for seconds, tenths of seconds, and hundredths of a second.

T 1.0 = seconds
T 0.1 = tenths of a second
T .01 = hundredths of a second

The timer box also displays the storage register that holds the current or accumulated time.

The two right-hand nodes, or lines, are output lines and provide power to contacts, coils, and the like. The top line provides power only when the timer's accumulated value is *equal* to the preset value; the bottom line provides power as long as the accumulated time is *less* than the preset time (not equal). The output on the bottom line only stops passing power, or is false, when the accumulated and preset values are the same (when the timer has timed out). Figure 12–24 shows an ON delay timer and how it is programmed to control outputs 00107 and 00108.

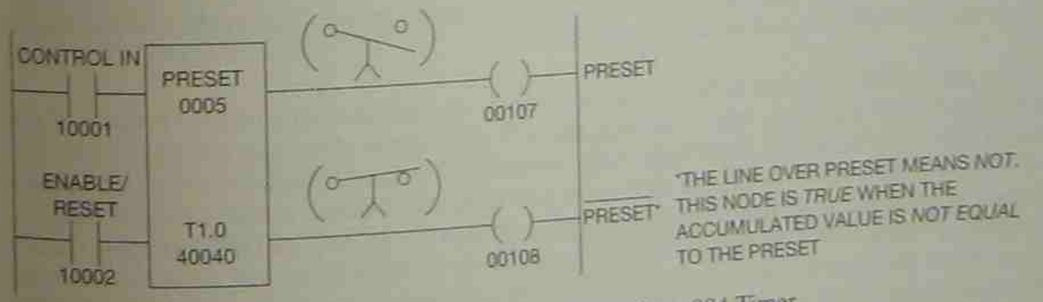


Figure 12–24 Programmed Modicon 984 Timer

If 10002 is closed, the timer is enabled, but not timing, and the value stored in register 40040 is 0. Since the value in register 40040 is 0, the accumulated value is not equal to the preset value of 00005, so output 00107 is false (*OFF*), but output 00108 is true (*ON*).

When 10001 in the control in line is closed, or goes true, the timer starts timing, and register 40040 will begin to accumulate time in one-second intervals. When the accumulated value is equal to the preset value (five), output 00107 becomes true, and output 00108 goes false. The status of the outputs remains the same until input 10002 in the enable/reset line is opened and resets the timer to zero. When the timer is reset, output 00107 is turned *OFF* and output 00108 is turned *ON*.

To duplicate the N.O.T.C. contacts of a pneumatic time delay, contacts with address 00107 should be used. For N.C.T.O. contacts, address 00108 is used.

The Gould 984, like many other PLCs, does not have a dedicated OFF delay instruction. To exactly duplicate the timing action of an OFF delay timer, additional Rungs of logic are required. Figure 12–25 shows the additional Rungs of logic required to obtain N.C.T.C. contacts and N.O.T.O. contacts. A timing chart is used to clarify the timing action.

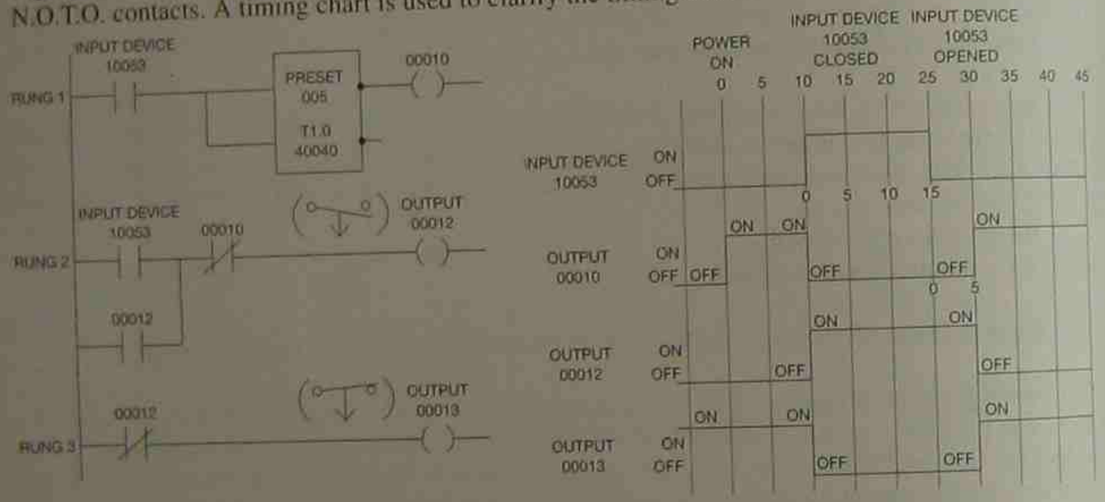


Figure 12–25 Programming a Gould 984 ON Delay Timer to Duplicate the Timing Action of an OFF Delay Timer

Notice that the enable/reset line has been tied into the control in line. This enables and turns the timer *ON* and *OFF* with just one input device (10053).

When power is first applied to the circuit, the timer is enabled and starts to time because of the EXAMINE OFF instruction 10053, which is the N.O. input device. Once the timer has timed out, output 00010 will be set to 1, or be turned ON. Output 00012 in Rung 2 cannot be energized because of the open 10053 input contacts, as well as the now open 00010 output contacts. Output 00013 in Rung 3 is energized as soon as the power is applied through the EXAMINE OFF (normally closed) contacts of output 00012. When the input device is closed, the timer Rung goes false and resets the timer. When the timer is reset, contacts 00010 in Rung 2 go back to their normally closed state. With the input device 10053 and contacts 00010 now closed, output 00012 energizes. When 00012 ener-

gizes, the EXAMINE ON contacts in Rung 2 close and act as holding contacts, while the EXAMINE OFF instruction in Rung 3 goes false and turns output 00013 *OFF*. The circuit remains in this condition until input device 10053 is opened, which activates the timer. When the accumulated time equals the preset time (value in register 40040 equals 0005), the timer will time out, making the EXAMINE OFF instruction in Rung 2 go false, which turns *OFF* output 00012, and causes output 00013 in Rung 3 to again be set to 1 (turn *ON*). This timing action exactly duplicates the timing action of an OFF delay timer. Additionally, normally open contacts (EXAMINE ON instructions) from output 00012 can be programmed for N.O.T.O. timer action, while normally closed contacts (EXAMINE OFF instructions) can be used for N.C.T.C. timer contacts.

For a retentive timer, a different input device is programmed in the control in and reset lines as indicated in Figure 12–26.

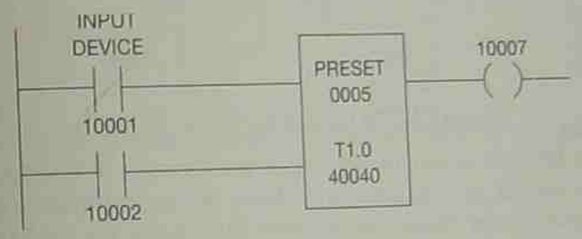


Figure 12–26 Retentive Timer

CASCADING TIMERS

When circuit requirements demand more time than is available from a single timer, two or more timers can be programmed together as shown in Figure 12–27a. Programming two or more timers together to extend the timing range is called **cascading**.

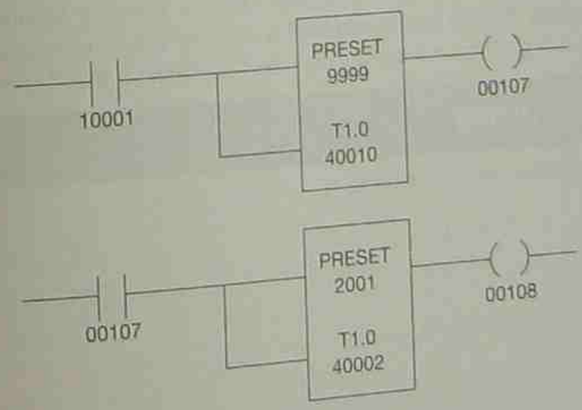


Figure 12–27a Cascading Timers

In this circuit, the first timer is controlled by input device 10001. When the device is true, the timer starts to time. When the accumulated time is equal to the preset time, output 00107 is set to 1, or *ON*. When 00107 is set to 1, the second timer is enabled and starts to time. When the second timer has timed out, output 00108 is turned *ON*. The total time to turn ON output 00108 after input 10001 was closed is 12,000 seconds (9999 + 2001), or 200 minutes.

A second, and perhaps easier method to cascade timers, is to put the two timers in series as shown in Figure 12–27b. This method only works with PLCs that allow timers to be placed in series, such as the Modicon 984. If the timer is considered an output instruction, like the Allen-Bradley PLC families, this method does not work, and the two-rung method shown in Figure 12–27a must be used.

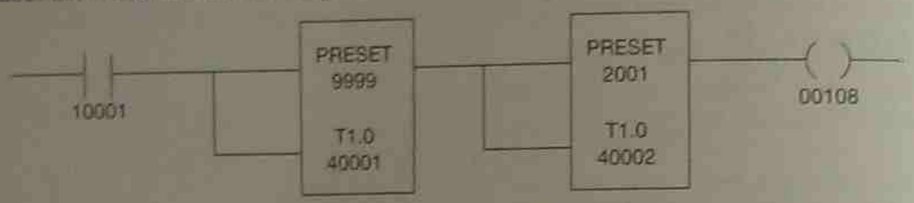


Figure 12–27b Cascading Timers in Series

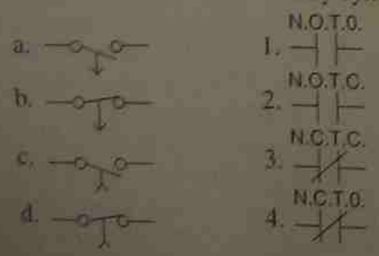
Once the input device 10001 is closed, the first timer starts to time. When the first timer times out, the second timer is activated and starts to time. When the second timer's accumulated value is equal to the preset value, output 00108 will be turned *ON*. Again, the total time to turn on output 00108 after input 10001 closes, is 12,000 seconds (9999 + 2001), or 200 minutes. By cascading timers, virtually any required time can be achieved.

Chapter Summary

Although the format is different for different PLCs, the basic principles are the same. Preset and accumulated times are stored and compared on each processor scan. When the accumulated value equals the preset value, discrete output devices or internal outputs can be turned *ON* or *OFF*. Timers can be programmed for *ON delay*, *OFF delay*, or as *retentive* timers. The only limit to the number of timed and instantaneous contacts that can be programmed is memory size. Programmed timers offer a wider range of time settings and greater accuracy than is possible with hard-wired pneumatic timers.

Review Questions

1. Match the standard time delay symbols.



2. The amount of time for which a timer is programmed is called the:
   a. preset
   b. set point
   c. desired Time (DT)
   d. all of the above
3. T F As scan time increases, so does the accuracy of any programmed timers.
4. When the timing of a device is not reset due to a loss of power, the timer is said to be:
   a. holding
   b. secured
   c. retentive
   d. continuous
5. When more time is needed than can be programmed with one timer, two or more timers can be programmed together. This programming technique is called:
   a. stacking
   b. cascading
   c. doubling
   d. synchronizing
6. When programming timers with Allen-Bradley format, which bit will act as an instantaneous contact?
   a. DN
   b. TT
   c. EN
   d. IN
7. When the accumulated time is equal to the preset time, which bit in the Allen-Bradley PLC-5 family will be true?
   a. DN
   b. TT
   c. EN
   d. IN
8. When programming a Modicon 984, which line is true when the accumulated time is *not* equal to the preset time?
   a. preset line
   b. *not* preset line

CHAPTER

13 Programming Counters

Objectives

After completing this chapter, you should have the knowledge to:
- Write a program using up and down counters.
- Define the terms *increment* and *decrement*.

Programmed counters serve the same function as the mechanical counters used in the past. Programmed counters can count up, count down, or be combined to count up and down. Counters are similar to timers, except they do not operate on an internal clock but instead are dependent on external or program sources for counting.

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX COUNTERS

Allen-Bradley offers two types of counters: up counters (CTU) and down counters (CTD). Both counters are retentive until reset by a reset instruction. Figure 13–1 is a typical Allen-Bradley up counter.

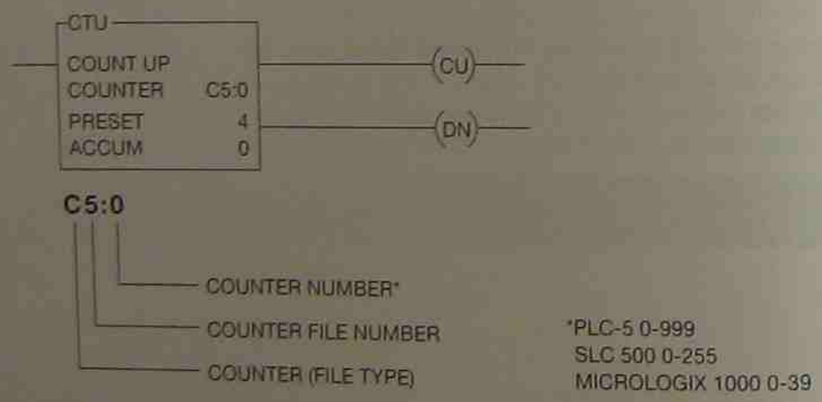


Figure 13–1 Allen-Bradley PLC-5 Counter Format

The Allen-Bradley up counter format is similar to the timer format. The up counter consists of a counter block that contains the up counter address, the preset value, and the accumulated count value which can

be any number from 0 to +32,767. The counter address consists of C for counter, the file number (5 [the default number]), a colon (:), and the counter number. File 5 is the default file from the data table for counters using the PLC-5, SLC 500, and the MicroLogix 1000. The PLC-5 can be programmed to use files 3-999 for additional counter files. The SLC 500 can be programmed to use files 9-255 for additional counter files. The MicroLogix is limited to one counter file, which is the default file, file 5.

By only having one counter file the MicroLogix 1000 is limited to 40 counters (counters 0 through 39), whereas the SLC 500 can use files 0 through 255 for counters. The PLC-5 can have counter numbers from 0 through 999. Each counter requires three words of memory, as shown in Figure 13–2.

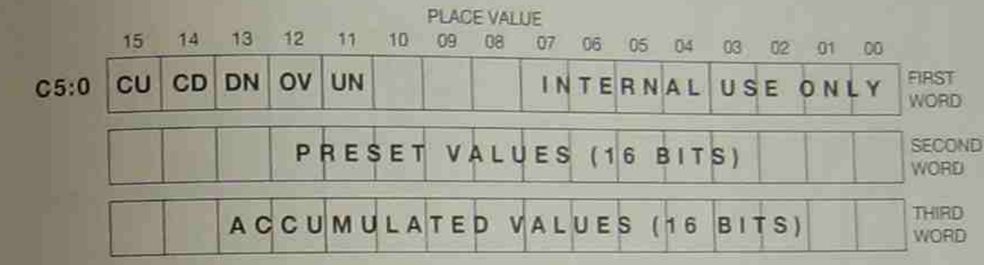


Figure 13–2 Storage Format for Counters

The first word stores the status bits of the counter (bits 11 through 15). The second word holds the preset values, or count, and can range from –32,768 to +32,767. The positive numbers are stored in 16-bit binary, while the negative numbers are stored in the 2s complement. The third word stores the accumulated count, and can be any number from –32,768 to +32,767. 2s complement is covered in Chapter 6.

The status bits for up and down counters that are stored in the first word are as follows:

Count Up Enable Bit (CU) The CU bit (bit 15) is set to 1, or is true, when the rung is true, and remains true as long as the up counter is enabled. The CU bit goes false when the counter is reset or the counter rung goes false. The CU bit is *only used* with up counters.

Count Down Enable Bit (CD) The CD bit (bit 14) is set to 1, or is true, when the rung is true, and remains true as long as the down counter is enabled. The CD bit goes false when the counter is reset or the counter rung goes false. The CD bit is *only used* with down counters.

Count Up Done Bit (DN) The DN bit (bit 13) is set to 1 when the accumulated count equals the preset count, and remains set to 1, or *ON*, as long as the accumulated value (count) is equal to or greater than the preset value.

Count Down Done Bit (DN) The DN bit (bit 13) is *ON*, or set to 1, as long as the accumulated value is equal to or greater than the preset value. The DN bit is only reset to 0, turned *OFF*, when the accumulated count is less than the preset value.

Count Up Overflow Bit (OV) The OV bit (bit 12) is set by the processor to 1, or ON, when the accumulated count exceeds the *upper limit* of (+)32,767. When this limit is reached, the count wraps around to (–)32,767, and the up counter increments from there.

Count Down Underflow Bit (UN) The UN bit (bit 11) is set by the processor to 1, or *ON*, when the accumulated count exceeds the *lower limit* of (–)32,768. It wraps around to (+)32,767, and the CTD instruction counts down from there.

Figure 13–3 shows a CTU counter and how it is programmed to control outputs O:013/01 and O:013/02. Rung 5 is the reset rung that resets the counter's accumulated value to 0000. An output instruction is used to reset the counter. The reset (RES) command must have the same address as the counter to enable it to be reset.

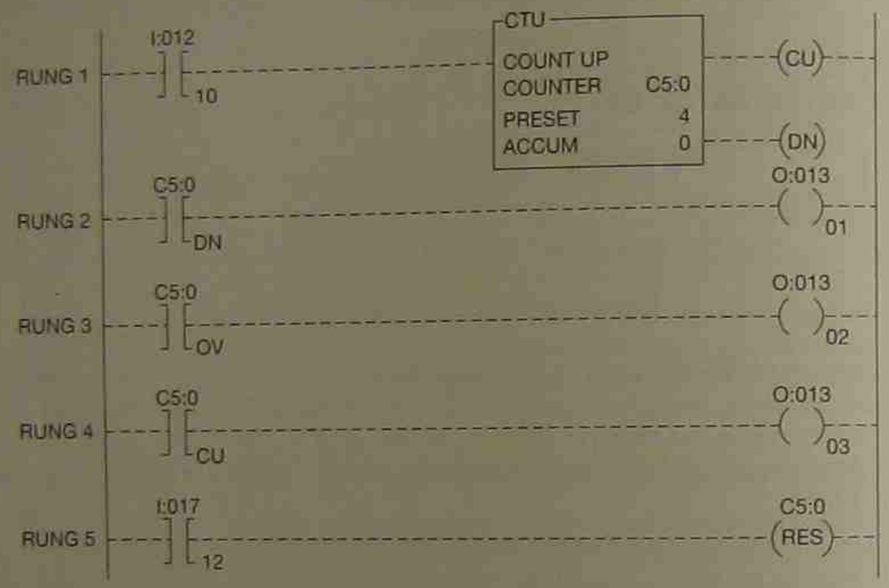


Figure 13–3 Programmed Up Counter (CTU)

Each time input device I:012/10 in Rung 1 makes a transition from false to true, the counter increments, or counts up by one. When the accumulated value (count) is equal to or greater than the preset count, the done (DN) bit is set to 1 by the processor, and Rung 2 becomes true, turning ON output O:013/01. Rung 3 is not true unless the count exceeds the counter's upper limit of (+)32,767. If the count exceeds the limit, output O:013/02 comes *ON* and remains *ON* until the counter is reset by closing and then opening input device I:017/12 in Rung 5. Bit 15, the count up bit (CU) can be programmed and used to indicated that the counter is enabled and that Rung 1 is true. The CU bit in Rung 4 is set to 1 by the processor any time that input device I:012/10 is true, thereby enabling the counter. Bit 15 is reset to 0 when Rung 1 goes false, or the timer is reset.

Figure 13–4 shows a typical counting chart for a CTU timer.

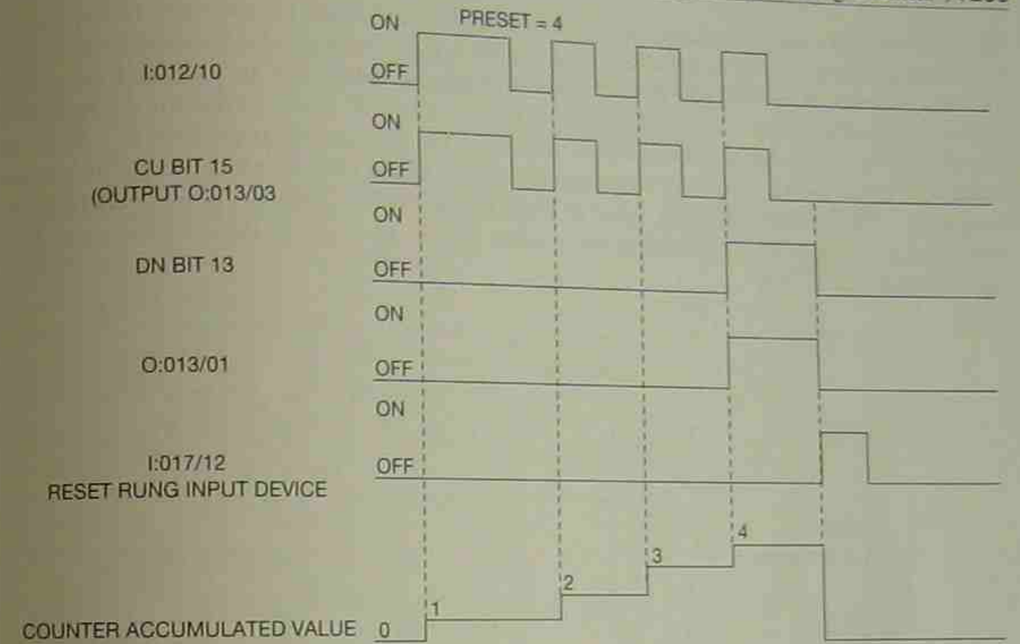


Figure 13–4 CTU Counting Chart

Figure 13–5 shows how a PLC-5 down counter (CTD) is programmed.

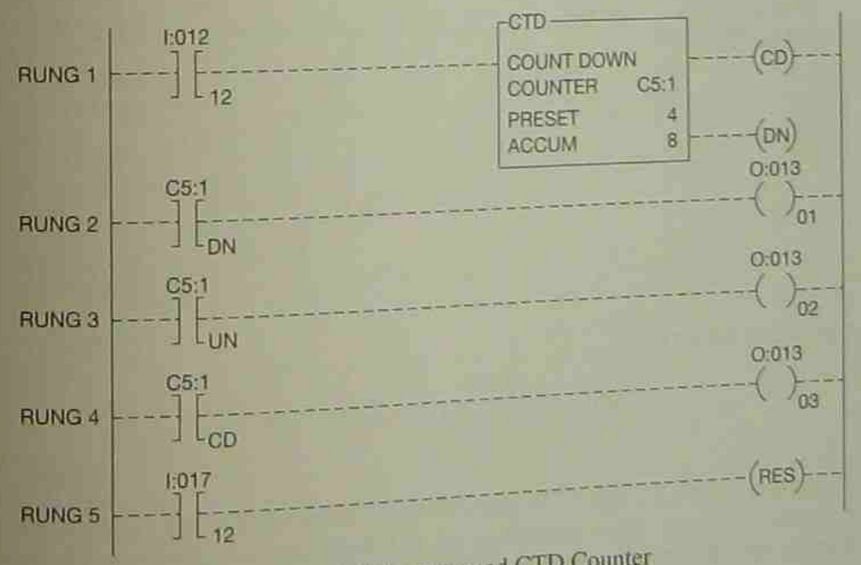


Figure 13–5 Programmed CTD Counter

The CTD counter counts down each time input device I:012/12 in the counter rung (Rung 1) goes from false to true. As long as the accumulated count is equal to or greater than the preset count, the output device (O:013/01) in Rung 2 remains energized. When the accumulated count falls below the preset count of 4, output O:013/01 is set to 0, or *OFF*. Rung 3 contains the underflow bit, which is opposite the overflow bit used with the CTU counter, and is only set to 1 when the count goes below (–)32,768. Rung 4 contains the CD bit and is *ON*, or true, any time the counter is enabled. The CD bit mirrors the status of input device I:012/12. Rung 5 is the reset rung and uses input device I:017/12 for resetting the counter. Figure 13–6 shows the counting chart for a count down timer (CTD).

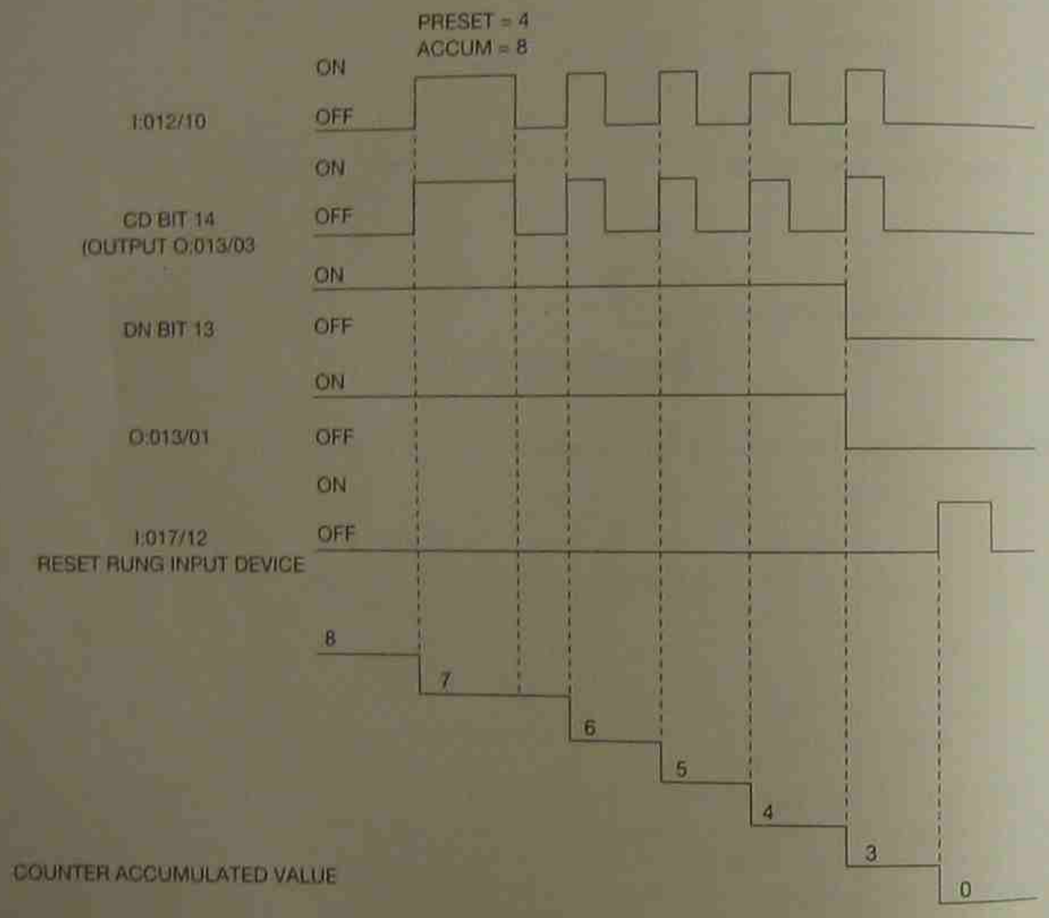


Figure 13–6 CTD Counting Chart

Figure 13–7a shows how CTU and CTD instructions can be combined, and Figure 13–7b is an example of a counting chart. When combining up and down counters, the same counter file and counter number are used for both counters as well as for the reset instruction in Rung 6.

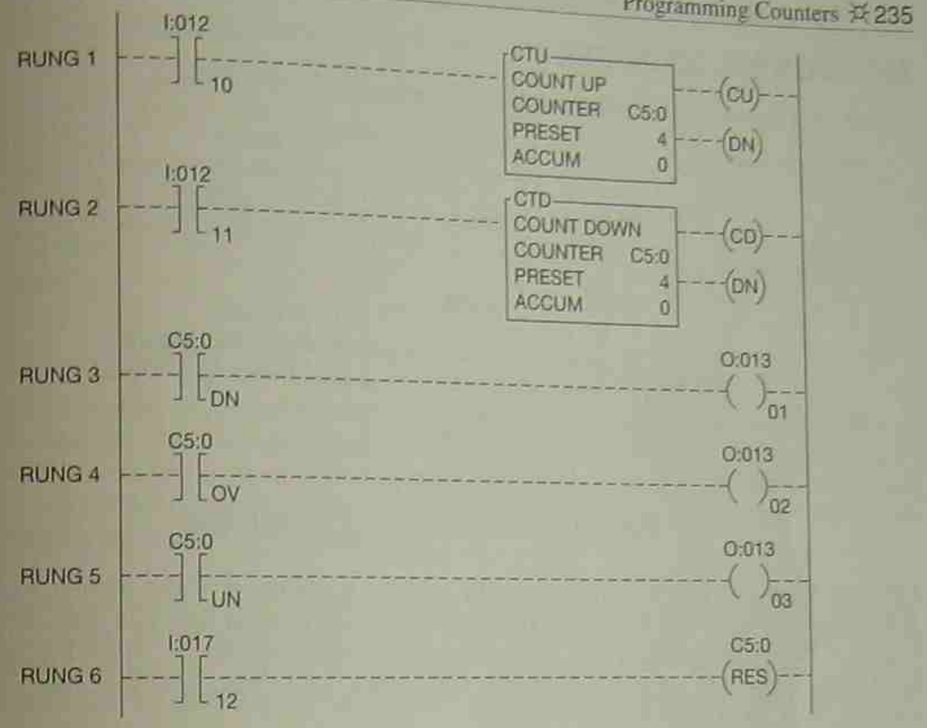


Figure 13–7a Combining CTU and CTD Instructions

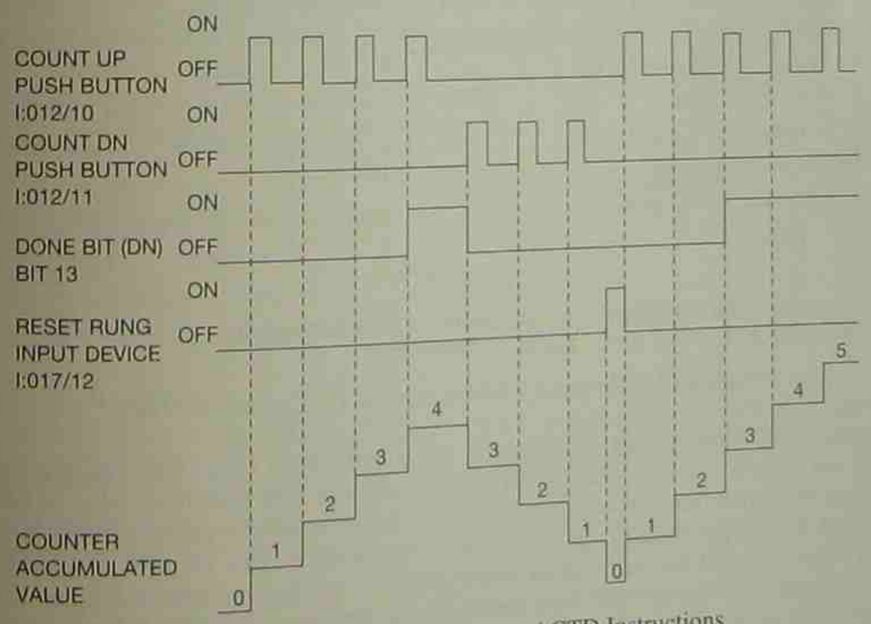


Figure 13–7b Combining CTU and CTD Instructions

MODICON 984 COUNTERS

Figure 13–8 shows a typical Modicon 984 PLC up counter (UCTR) format.

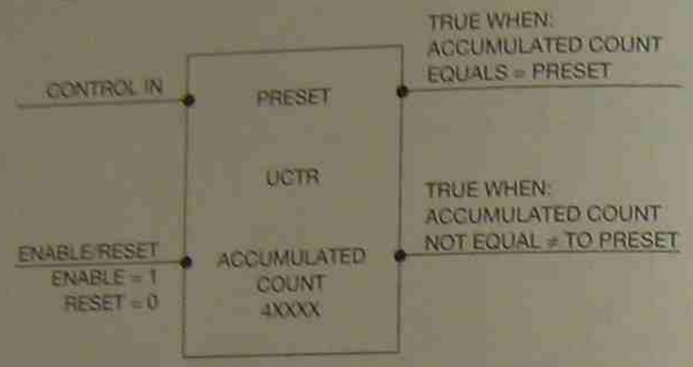


Figure 13–8 Modicon 984 Up Counter Format

The Control In line increments (increases by one) the actual count of the counter each time the line makes a false-to-true transition. The counting line (Control In line) can only increment the counter if the enable circuit is *ON*. The top line on the output side is true *only* when the accumulated count *is* equal to the preset value. The bottom line will be true, or *ON*, as long as the accumulated count is *not* equal to the preset count.

Figure 13–9 illustrates how an up counter (UCTR) is programmed.

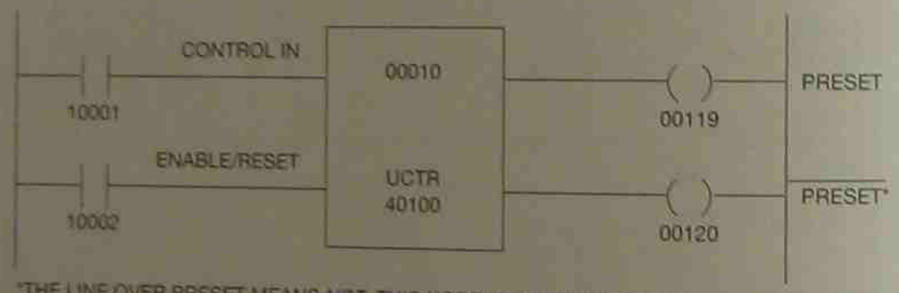


Figure 13–9 Programmed Modicon 984 Up Counter (UCTR)

When input 10002 in the enable circuit is *ON*, the counter starts from 0000 and increments the count by one each time input device 10001 in the Control In line goes from false (*OFF*) to true (*ON*). When the actual (accumulated) count stored in 40100 equals the preset count, output 00119 is energized and output 00120 is deenergized, or turned *OFF*. Any contacts labeled 00119 or 00120 in other rungs of the circuit open and/or close, based on the status of outputs 00119 and 00120. The counter is reset to 0000 when the enable circuit is opened, and is held at 0000 until the enable circuit is again closed, or true.

Figure 13–10 shows a typical Modicon 984 down counter (DCTR).

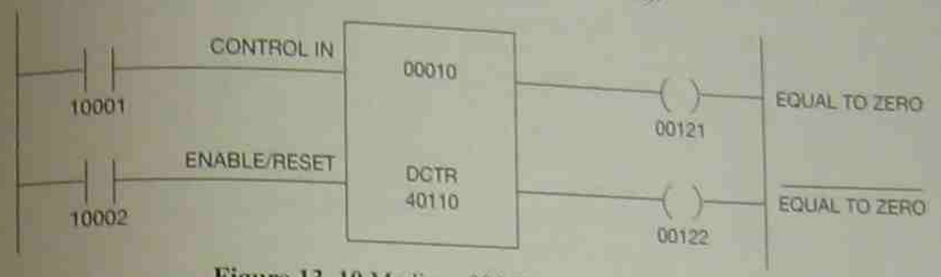


Figure 13–10 Modicon 984 Down Counters (DCTR)

Down counters count from a preset value down to 0000. When the actual count reaches 0000, output 00121 is energized (accumulated count equals zero) and output 00122 is deenergized (accumulated count is not equal to zero). When input 10002 is closed, enabling the timer, the counter counts down (decrements) from its preset value of 00010, each time the input device 10001 in the counting line goes from false (*OFF*) to true (*ON*).

When the enable circuit is opened, the accumulated value in 40110 is reset and held at the preset value of 0010 instead of 0000, like an up counter. When the enable circuit is again energized, each false-to-true transition of the counting line (or circuit) counts down from the preset value to 0000. When the actual (accumulated) count reaches 0000, output 00121 is energized and output 00122 deenergizes, or turns *OFF*.

Up and down counters can be programmed together as shown in Figures 13–11a and 13–11b to count products as they enter a conveyor line (count up) and as they leave the line (count down).

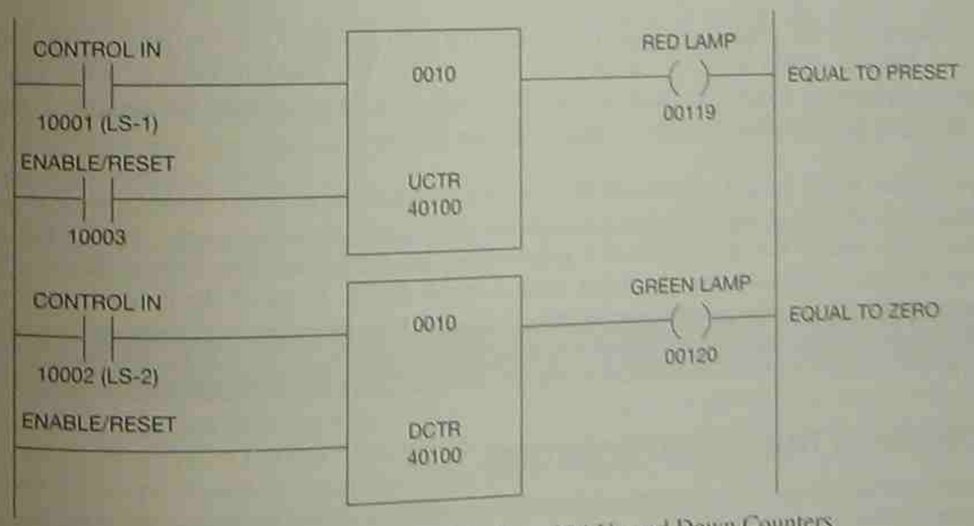


Figure 13–11a Combining Modicon 984 Up and Down Counters

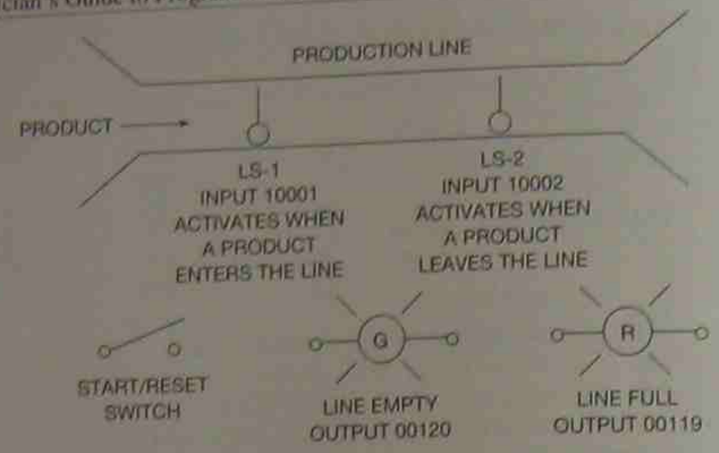


Figure 13–11b Applying Up and Down Counters

Notice that the same holding register (storage word) 40100 is used for both the up counter (output 00119) and down counter (output 00120). By sharing the same holding register, the actual value stored in 40100 of the up counter, which is programmed first, determines the actual value in 40100 of the down counter.

As a product enters the conveyor line, input 10001 is activated (false-to-true), and the actual count in 40100 increments from 0000 to 0001. The next product increments 40100 to 0002, and so on. If eight products entered the line before any were removed, the accumulated count in 40100 would be 0008. The first product to leave the line and activate 10002 decrements the actual count in 40100 (from 0008 to 0007). The next product that left the line and activated 10002 (false-to-true), would again decrease the accumulated count in 40100 from 0007 to 0006.

The indicator lamps, outputs 00120 (green) and 00119 (red) in Figure 13–11b, indicate the condition of the line. When the line is empty, the down counter has counted down to 0000, and the green lamp (output 00120) is ON. When the line is full, and the up counter has counted to 0010, the red lamp (output 00119) is ON to indicate the line is full.

Input device 10003 is used as a START/RESET switch. When the switch is closed, the up counter is enabled, and when the switch is opened, the up counter is reset to zero. Notice that the enable/reset line of the down counter is programmed unconditionally (no control device). This programming technique keeps the down counter enabled at all times. If the down counter was reset, it would reset to the preset value of 10, and because both counters share the same storage, or holding register, the up counter would also be reset to 10. Up and down counters are retentive during power failures.

COMBINING TIMERS AND COUNTERS

Timers can be combined with counters when it is necessary to extend the time of the timer beyond its normal limits. An example of combining a timer with a counter is shown in Figure 13–12.

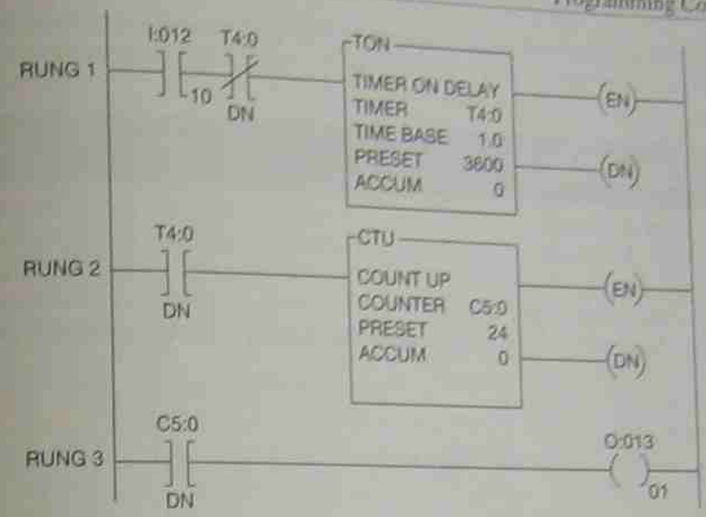


Figure 13–12 Combing a Timer with a Counter

The timer (T4:0) has a time base of 1.0 seconds and a preset value of 3600. The 3600-second preset value is equal to one hour. When input device I:012/10 is closed, the timer starts to time in 1-second increments. When the accumulated time is equal to the preset value, the DN bit (bit 13) is set to 1 and the CTU counter in Rung 2 counts, or increments, by one. (When the DN bit was set to 1, it also acted as a reset for the timer because of the EXAMINE OFF instruction in Rung 1.) The momentary action of bit 13 (false-to-true-to-false) resets the timer to 0, and the timer starts to accumulate time again. When the accumulated time on the timer has reached 3600 seconds, the timer resets itself once again and increments counter C5:0 again. The counter continues to count each time the DN bit makes a false-to-true transition (every 3600 seconds, or one hour) until the accumulated count equals the preset value of 24. When the counter has counted to 24 (24 hours), the counter DN bit is set to 1 and output O:013/01 in Rung 3 is turned *ON*.

Note: *Remember that the length of the program effects scan time, which in turn effects timer accuracy and total time. The actual time it takes for the counter to count to 24 may be 24 hours plus or minus a few minutes.*

When a timer is programmed to reset itself, as in Figure 13–12, it is referred to as a *free wheeling timer*.

Chapter Summary

Programmed counters give added flexibility and control to electrical process equipment and/or driven machinery. Similar to timers, counters store values in binary or binary coded decimal (BCD) format for the preset and accumulated counts. For up counters, the processor compares the preset and accumulated values on each scan, and updates the I/O section on the scan so that the accumulated value equals the preset value. For down counters, the processor updates the I/O section on

the scan so that the accumulated value is 0 for some PLCs, or when the accumulated value is equal to or greater than the preset for other PLCs. Counters count (increment or decrement) when the count rung transition is from false to true.

Review Questions

1. Define the term *increment*.
2. Define the term *decrement*.
3. What is the *preset value* or *count*?
4. What is the *accumulated value* or *count*?
5. In Figure 13–A, switch I:012/10 is now open. When switch I:012/10 is closed, counter C5:0 will:
   a. increment by one
   b. decrement by one
   c. not count, and the accumulated value will remain at 0

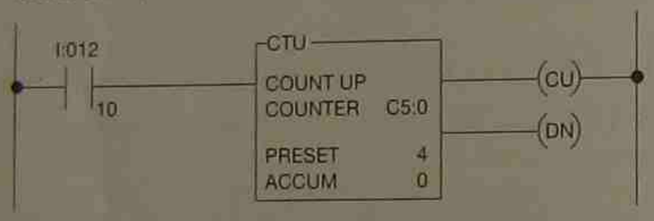


Figure 13–A

6. Output O:013/01, shown in Rung 2 of Figure 13–B, is true:
   a. only when the count is equal to the preset value
   b. when the count is equal to or greater than the preset value
   c. when the count is less than the preset value
   d. when the accumulated value reaches +32,767 and overflows
   e. when the count goes to 011

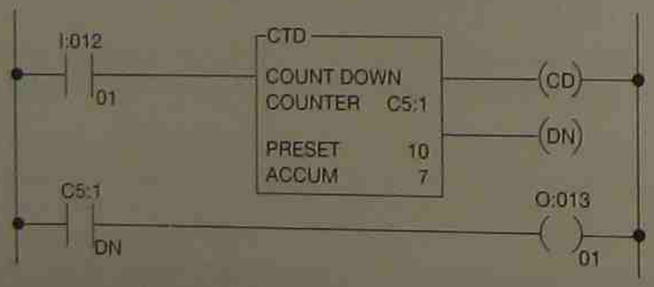


Figure 13–B

7. When the accumulated count is equal to or greater than the preset count, which bit in the Allen-Bradley SLC 500 family will be true?
   a. CU
   b. CD
   c. OV
   d. DN

8. When programming a Modicon 984, which line is true when the accumulated count is not equal to the preset count?
   a. the preset line
   b. *not* the preset line
9. T F Up and down counters can be programmed together to count up and down.
10. The reset rung shown in Figure 13–C resets counter C5:0:
    a. automatically when the count reaches 010
    b. automatically when the count reaches 011
    c. only when the count reaches 32,767
    d. only when switch I:017/12 is closed
    e. only when switch I:017/12 is closed and then opened

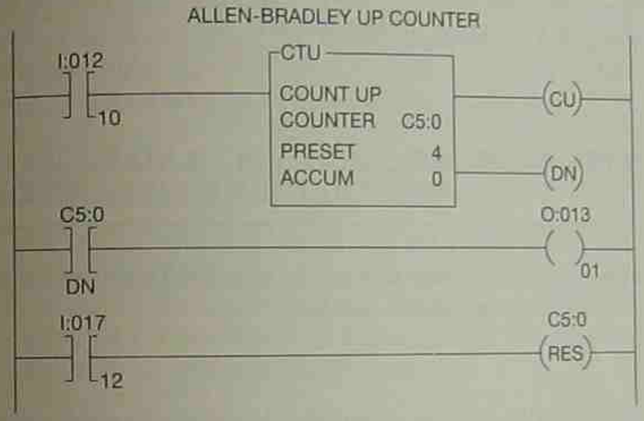


Figure 13–C

11. Define the term *overflow*.
12. Define the term *underflow*.
13. When an up counter accumulated value equals the preset value, the counter will:
    a. reset itself
    b. stop counting
    c. continue to count
    d. continue to count but go into an overflow condition as soon as the accumulated value exceeds the preset value

CHAPTER

14 Data Manipulation

Objectives

After completing this chapter, you should have the knowledge to:

- Explain what data transfer is.
- Define the term *writing over*.
- Write a rung of logic that transfers data from one word to another.
- Identify the standard data compare instructions.
- Write logic that compares data to control an output.

Most PLCs now have the ability to manipulate data that is stored in memory. Data manipulation can be placed in two broad categories: data transfer and data compare.

DATA TRANSFER

Data transfer consists of moving or transferring numeric information stored in one memory word location to another word in a different location. Words in the user memory portion of the processor may be referred to as data table words, holding register, and/or storage register words, depending on the PLC.

Figures 14–1a and 14–1b illustrate the concept of moving numerical data from one word location to another word location. Figure 14–1a shows that numeric (binary) data is stored in word 0 of file N7, and that no information is currently stored in word 1 of file N7.

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	0	1	1	0	1	0	1	1	1	0	1

N7:0 (WORD 0 FROM INTEGER FILE 7)

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

N7:1 (WORD 1 FROM INTEGER FILE 7)

Figure 14–1a Numeric Data Stored in Words 0 and 1 of File N7

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	0	1	1	0	1	0	1	1	1	0	1

N7:0 (WORD 0 FROM INTEGER FILE 7)

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	0	1	0	0	1	1	0	1	0	1	1	1	0	1

N7:1 (WORD 1 FROM INTEGER FILE 7)

Figure 14–1b Data Transferred From Word 0 of File N7 Into Word 1 of File N7

After the data transfer (Figure 14–1b), word 1 of file N7 now holds the exact or duplicate information that is in word 0 of file N7. If word 1 had information already stored, rather than all 0s, the information would have been replaced. When new data replaces existing data in a word after a transfer, it is referred to as **writing over** the existing data.

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX DATA TRANSFER INSTRUCTIONS

The PLC-5, SLC 500, and MicroLogix use a move (MOV) instruction for moving data from one word to another. MOV is an output instruction that copies a value from one word (source address) to another word (destination address). When the rung that holds the MOV instruction is true, the instruction moves data from the source address into the destination address on each processor scan. Figure 14–2 shows the MOV format.

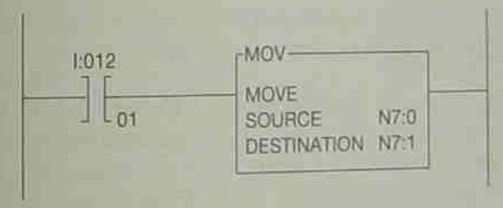


Figure 14–2 PLC-5 MOV Format

When input device I:012/01 is closed and the rung is true, the MOV instruction reads the data from the source address and copies it to the destination address. In this case, the source is integer file N, file number 7, and word 0. The destination is integer file N, file 7, word 1. As long as the rung stays true, the values found in word 0 of N7 is transferred (copied) into word 1 of N7 on each program scan. The integer file (as discussed in Chapter 5) is used for storing whole numbers.

To illustrate how a MOV instruction could be used, consider a machine that produces two types of products. Product A requires a time delay of 10 seconds during the processing and Product B requires a 20-second delay. The ON delay timer in the process program would be programmed as shown in Figure 14–3.

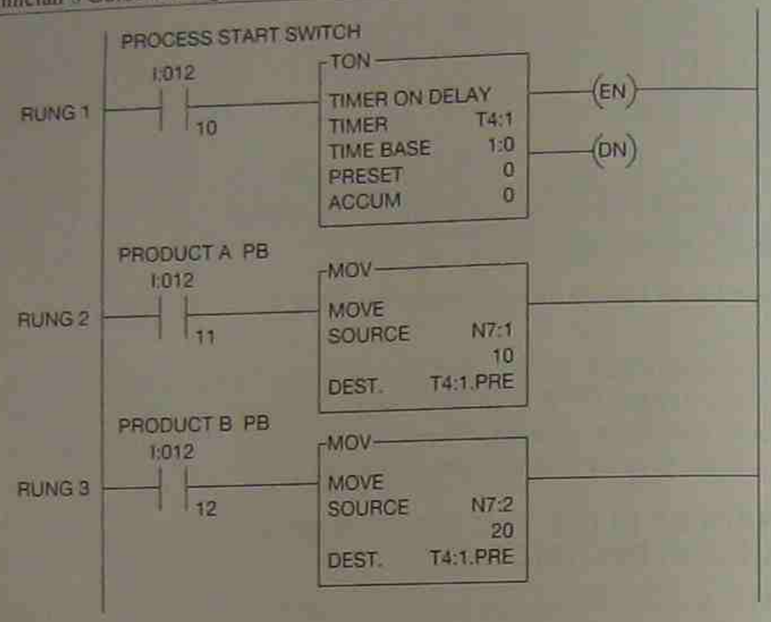


Figure 14–3 MOV Instruction Used to Change Timer Preset Values

The ON delay timer T4:1 is programmed with no preset value as shown in Rung 1. Rung 2 has input device I:012/11 controlling a MOV instruction programmed to move the numeric value found in Word 1 of integer file N7 (Source) into the destination word which is shown as T4:1.PRE. The destination word is the word that holds the preset value for timer T4:1. Rung 3 is programmed so that when the rung is true, the MOV instruction moves the numeric value in Word 2 of file N7 into the word that holds the preset value for timer T4:1.

When Product A is being processed, input device I:012/10 and I:012/11 are activated and the numeric value of 10 from Word 1 of file N7 is moved into the word that holds the preset value of ON delay timer T4:1. This gives timer T4:1 a preset value of 10 seconds. As long as input device I:012/11 remains true, the preset value of T4:1 is 10. When Product B is to be run, input device I:012/11 is opened and input device I:012/12 in Rung 3 is closed. With Rung 3 now true, the value found in Word 2 of file N7 (20) is moved into the word that holds the preset value for ON delay timer T4:1. The value 20, from the Source (N7:1), is moved into the word that holds the preset time for timer T4:1 and overwrites the previous information which was a preset of 10.

MOV instructions can be used to change preset values of timer, preset, or accumulated values of counters, as well as transferring data between any two words to meet program requirements. An example of how a MOV instruction can be used to change preset values of a counter is a program that counts boards in a saw mill.

When the mill is producing 2 x 4s, it wants 400 in a stack. However, when the mill is producing 2 x 6s, only 250 boards are needed for a full stack. Figures 14–4a and 14–4b show how to change the preset value of an up counter for each different lumber size by using pushbuttons.

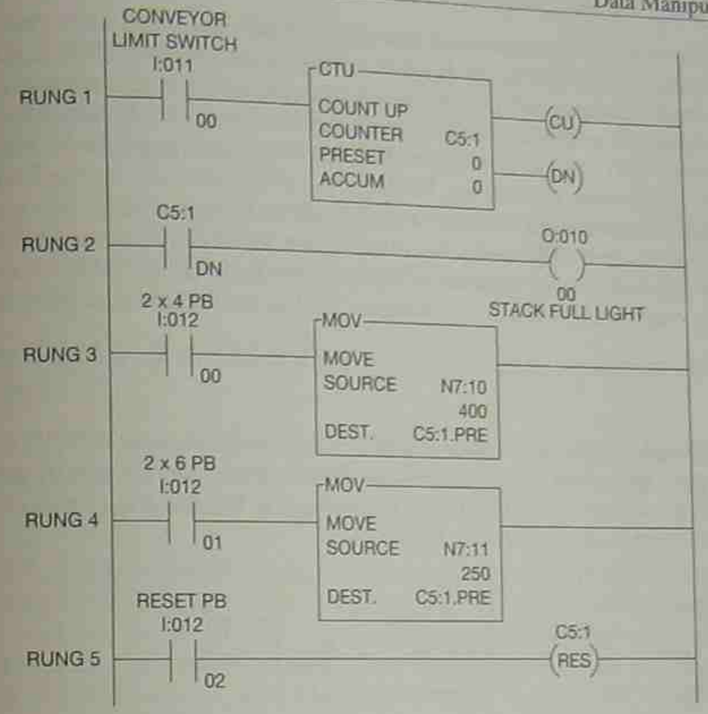


Figure 14–4a Changing Preset Values with a MOV Instruction

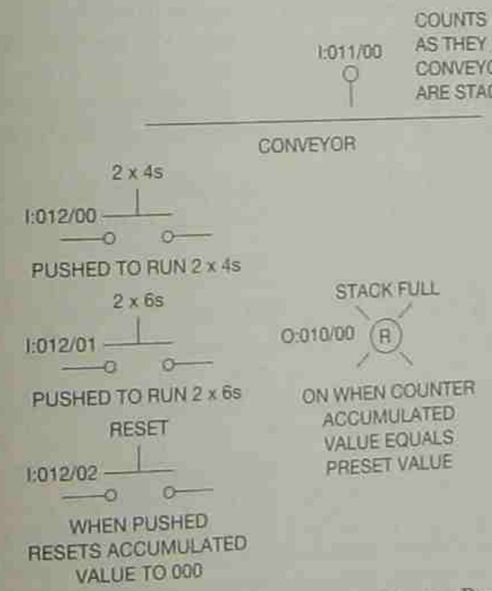


Figure 14–4b Pushbuttons Used to Change Preset Values

Up counter C5:1 is initially programmed with no preset (0). The preset is determined by whichever pushbutton is depressed. If 2 x 4s are to be counted, pushbutton I:012/00 is pushed, enabling the MOV instruction in Rung 3. When this rung is enabled, or true, it tells the processor to MOVE the value 400 stored in word 10 of file N7 into the word that stores the preset value for up counter C5:1. This causes CTU C5:1 to be preset to 400. A pushbutton is used so the rung will go false (open) after the preset value has been set. Holding the button down and keeping Rung 3 true, holds the value at 400, and transitions of input device I:011/00 do not increment the counter. After 400 boards have been counted (PR = AC), bit 13 (the done bit) of the first word of C5:1 in the up counter will be set to 1, and the "Stack Full" light, O:010/00, comes *ON*. After the stack has been moved, counter reset pushbutton I:012/02 is pushed to clear the accumulated value back to 0.

To change the preset value of up counter C5:1 from 400 to 250, the 2 x 6s' pushbutton (I:012/01) is depressed.

Another Allen-Bradley data manipulation instruction is the masked move (MVM) instruction. The MVM is an output instruction that copies a value from a source address to a destination address, but in addition, allows portions of the data to be **masked**, or blocked from being copied. The format for the MVM instruction is shown in Figure 14–5.

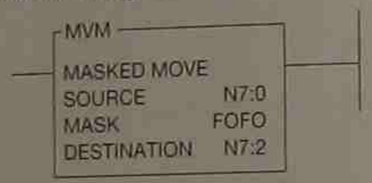


Figure 14–5 Masked Move (MVM) Instruction Format

To program an MVM instruction, a source address and a destination address are required, just as in the MOV instruction. The additional requirement for the MVM instruction is the mask data. For each bit of the destination word that is to be masked, or not copied, a 0 is used. If, on the other hand, it is desired that the data from the source word be written into specific bits of the destination word, a 1 is placed in that bit location. Figure 14–6 clarifies the operation of the MVM instruction.

Figure 14–6 Mask Bits Used to Block Transfer of Data from Source Address Into Destination Address

For each location in the destination word that you want to be overwritten by the data from the source word, a 1 is used. In Figure 14–6, only bits 4 through 7, and bits 12 through 15, are set to 1 in the mask, so only these bits of the destination word will have data transferred in from the source word. Those bits in the destination word that had 0s in the mask (bits 0 through 3 and bits 8 through 11) remain unchanged when the MVM instruction is true.

The bit status for the mask is entered by addressing a word and file that has the desired bit order that is wanted; for example, B100:0 (Binary file, file 100, word 0). The value can also be entered into the instruction using the hexadecimal format. The mask bit pattern shown in Figure 14–6 is F0F0 in hexadecimal.

MODICON PLC

The Modicon 984 PLC uses a **data transfer**, or **DX**, instruction to move data from one word to another. The DX instruction can also be used to make file-to-word moves and word-to-file moves, which will be covered in more detail in Chapter 16.

The MOV, MVM, and DX instructions (as well as other designated instructions by different PLC manufacturers) are all *data transfer instructions*, and their objective is to move numerical information from one word into another. To illustrate this concept, the data transfer for entering accumulated and preset values into counters was used. This is by no means the only application for a data transfer instruction. When using a given PLC, and becoming more familiar with its operation and capabilities, data transfer becomes a powerful tool that has many applications.

DATA COMPARE

Data compare opens a new realm of programming possibilities, and demonstrates why PLCs are rapidly replacing most, if not all, hard-wired control systems.

Data compare instructions, as the name implies, compare the data stored in two or more words and make decisions based on the program instructions.

Numeric values in two words of memory can be compared for *less than* (<), *equal to* (=), *greater than* (>), *less than or equal to* (≤), *greater than or equal to* (≥), and *not equal to* (≠), depending on the PLC.

Data compare concepts were previously used when timers and counters were discussed.

The ON delay timer turns *ON* an output when the accumulated value equals the preset value (AC = PR). What happens is the accumulated numeric data in one memory word is compared to the preset value in another word on each scan of the processor, and when the accumulated value equals the preset value (AC = PR), the output is turned *ON*. Additional programming instructions can compare memory words and turn *ON* outputs when the values are less than (<), equal to (=), greater than (>), and so on.

ALLEN-BRADLEY PLC-5, SLC 500, AND MICROLOGIX DATA COMPARE INSTRUCTIONS

The Allen-Bradley family of programmable controllers has a set of data compare instructions that include *equal* (EQU), *greater than or equal* (GEQ), *greater than* (GRT), *less than or equal* (LEQ), *less than* (LES), and *not equal* (NEQ). Figure 14–7 shows how the EQU instruction is programmed.

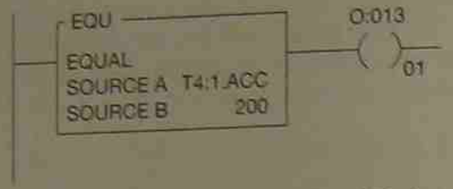


Figure 14–7 Allen-Bradley PLC-5 Equal To Instruction

The EQU instruction is true, and turns *ON* output O:013/01 when the value in Source A is equal to the value in Source B. Source A and B can be either numeric values or addresses that contain values. The value in Source A is the value of address T4:1.ACC (timer file 4, timer 1, accumulated value), whereas the value in Source B is the numeric value 200. To use either the accumulated or preset values of timers and counters, a period is entered after the timer number, followed by ACC or PRE.

Figure 14–8 illustrates how the GEQ instruction operates.

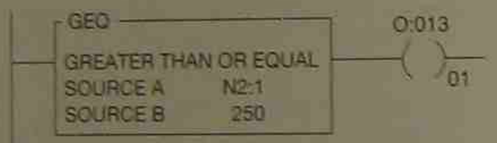


Figure 14–8 Allen-Bradley PLC-5 Greater Than or Equal To Instruction

This instruction becomes true and turns output O:013/01 *ON* when the value in Source A is greater than or equal to the value in Source B. Again, the value that is in Source A or B can be numeric values or addresses that contain values. In this illustration, the value in Source A is the value stored in integer File 7, word 1. The value in Source B is the numeric value of 250.

Another instruction is GRT (greater than) and is programmed as shown in Figure 14–9. This instruction is true when the value in Source A is greater than the value in Source B.

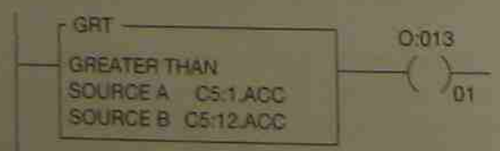


Figure 14–9 Allen-Bradley PLC-5 Greater Than Instruction

The instruction is true as long as the value in Source A is greater than the value in Source B. In Figure 14–9, output O:013/01 is turned *ON* anytime that the accumulated value of counter C5:1 is greater than the accumulated value in counter C5:12. As in the earlier example, the preset and accumulated values of timers and counters can be referenced by typing a period followed by either ACC or PRE after the timer or counter address. In Figure 14–9, Source A is the accumulated value of Counter 1, in counter file 5 (C5:1.ACC). Source B is the ACC value of timer 12, in counter file 5 (C5:12.ACC).

The less than or equal instruction (LEQ) is programmed as shown in Figure 14–10. This instruction is true whenever the value is Source A is less than or equal to the value stored in Source B.

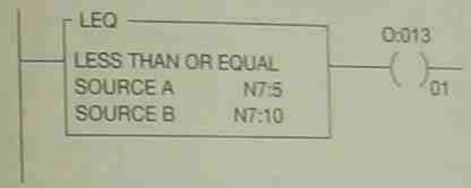


Figure 14–10 Allen-Bradley PLC-5 Less Than or Equal To Instruction

The LEQ instruction is true as long as the value in N7:5 is less than or equal to the value in N7:10. When the value in Source A is less than or equal to the value in Source B, output O:013/01 is turned *ON* by the processor.

The LES (less than) instruction is logically true when the value in Source A is less than the value in Source B. Figure 14–11 shows a LES instruction.

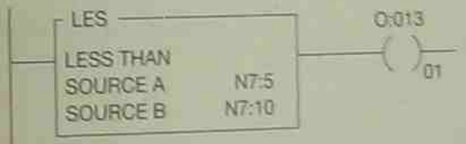


Figure 14–11 Allen-Bradley PLC-5 Less Than Instruction

In Figure 14–11, output O:013/01 is *ON* whenever the value in N7:5 (Source A) is less than the value in N7:10 (Source B). When the value in Source A is equal to or larger than the value in Source B, the instruction is not logically true, and output O:013/01 is set to 0, or *OFF*.

The not equal to instruction (NEQ) is programmed as shown in Figure 14–12. This instruction is true whenever the value in Source A is not equal to the value stored in Source B.

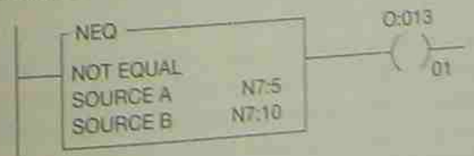


Figure 14–12 Allen-Bradley PLC-5 Not Equal To Instruction

The NEQ instruction will be logically true any time the value in N7:5 (Source A) is not equal to the value in N7:10 (Source B). As long as the values are not equal, output O:013/01 will be turned *ON*. This instruction is logically false only when the value in Source A is equal to the value in Source B.

To graphically demonstrate how data compare instructions can be used, consider the hardwired circuit in Figure 14–13. This circuit uses three pneumatic time delay relays to start up a 4-motor conveyor system in inverse order (4–3–2–1).

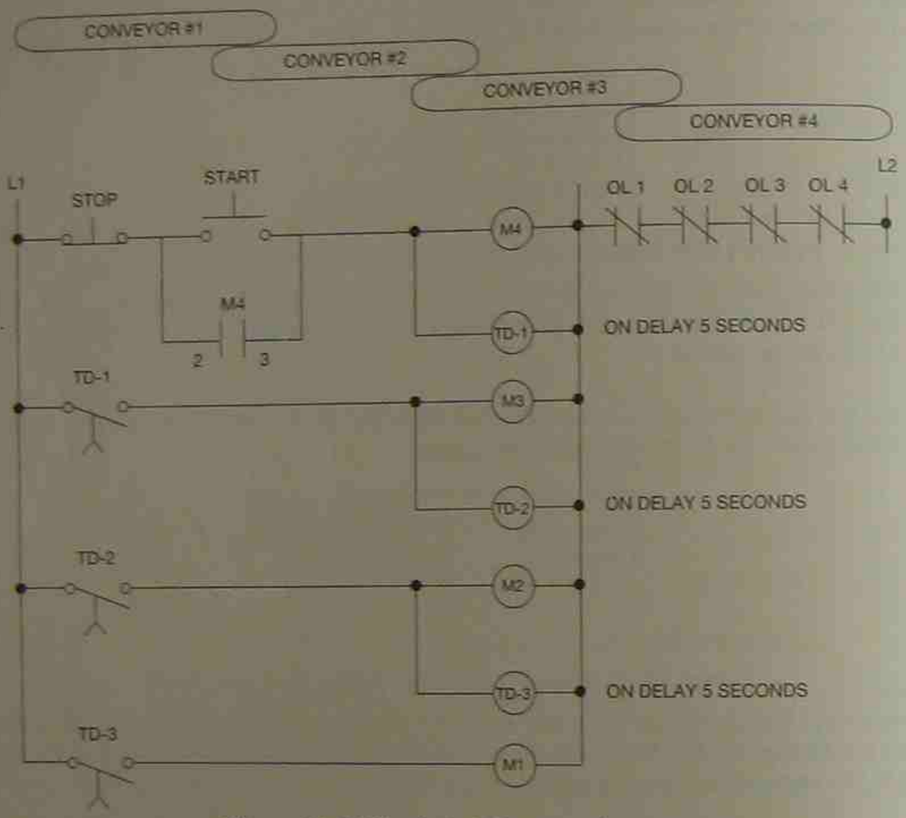


Figure 14–13 Hardwired Conveyor System

The same circuit can be programmed with an Allen-Bradley PLC using only one internal timer and two data compare statements, as shown in Figure 14–14.

Assume that STOP button O:012/00 and overloads O:012/01, O:012/02, O:012/03, and O:012/04 in Rung 1 are closed. When the START button (O:012/05) is pushed, Output M-4 (O:010/04) energizes, and holding contacts O:010/04 close and hold the circuit in. M-4 contacts O:010/04 also close in Rung 2 and enable the timer. The timer has been preset to 15 seconds (1.0 second time base). The accumulated time is stored in timer file T4:1 (remember that timers use three words of memory with the third word holding the accumulated value of the timer).

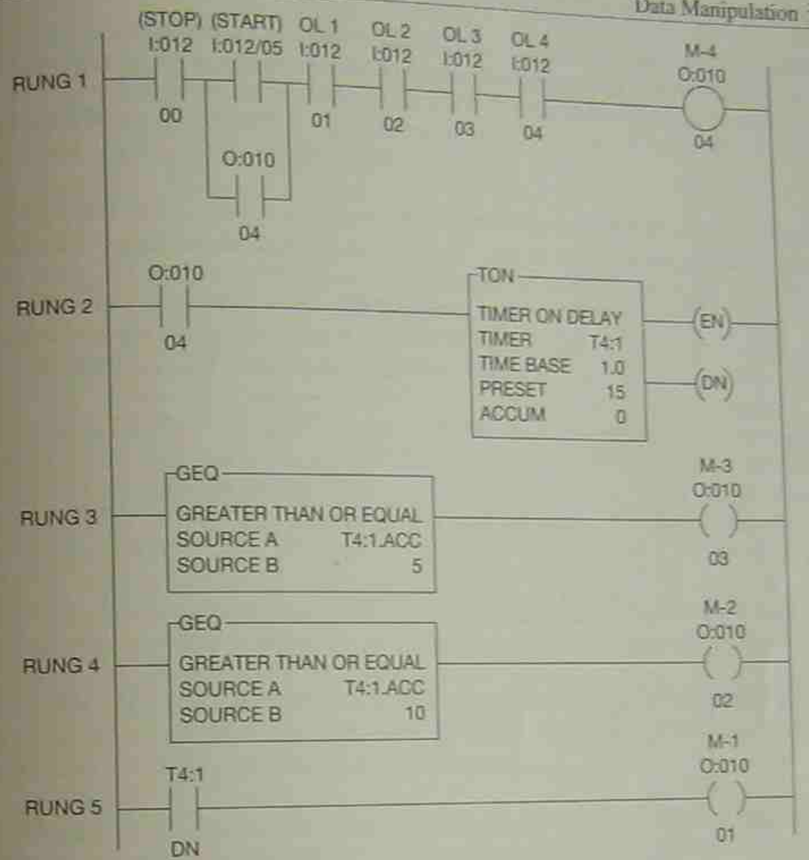


Figure 14–14 Square D Company Data Compare Format

The GEQ instruction in Rung 3 that controls output O:010/03 (Motor 3) is logically true when the accumulated value of timer T4:1 (Source A) is equal to or greater than the constant in Source B (5 seconds). When the accumulated value reaches 5, output O:010/03 (M-3) is turned *ON*. Similarly, when the accumulated value reaches 10, the logic of Rung 4 will be true and output O:010/02 will be turned *ON*. When the accumulated value of the timer reaches 15 and is equal to the preset value of 15, the done bit (DN), bit 13, will be set to 1 and the last motor, Motor 1, will be turned *ON*.

To further illustrate how the DATA COMPARE instructions work, consider the program in Figure 14–15 and the Time Chart for Data Comparisons in Figure 14–16.

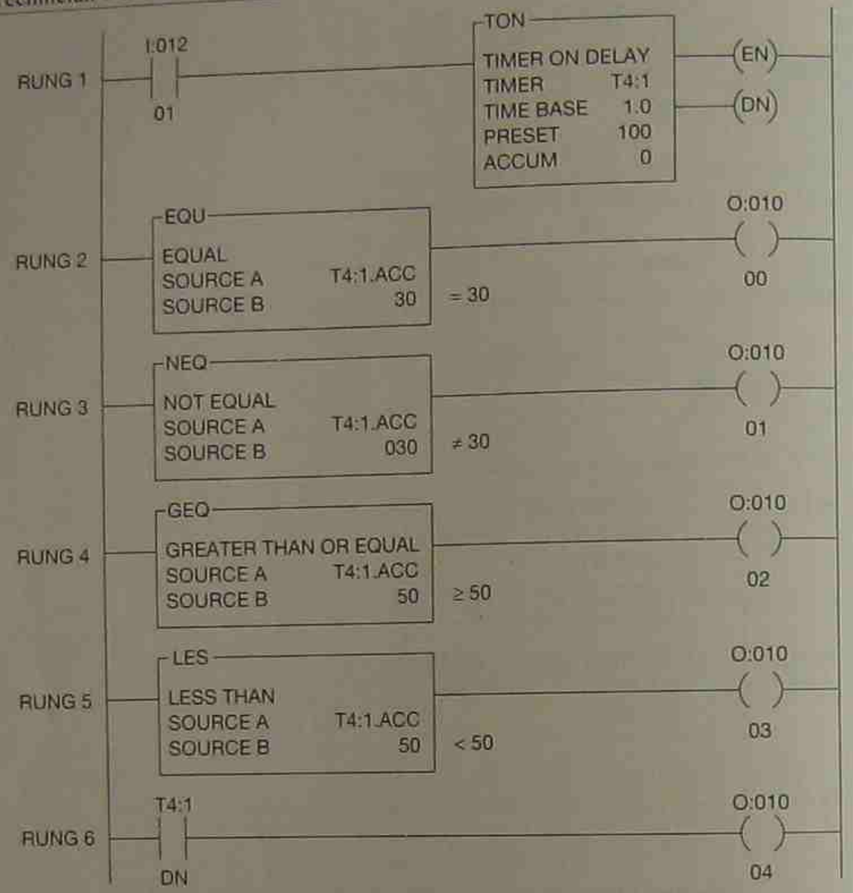


Figure 14–15 Data Compare Instructions

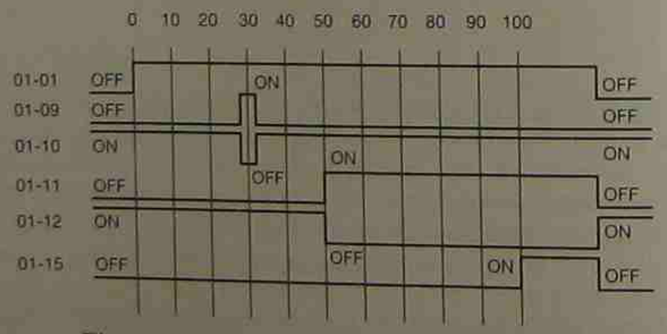


Figure 14–16 Time Chart for Data Comparisons

When power is applied, but before I:012/01 is closed to activate the timer, outputs O:010/01 (Rung 3) and O:010/03 (Rung 5) are energized. The NEQ instruction preceding output O:010/01 is true if the value in Source A is not equal (≠) to 30. With I:012/01 open, timer T4:1.ACC value is 00 and is *not* equal to 30. Output O:010/03 (Rung 5) is energized because the LES instruction is true anytime the accumulated value of T4:1 is less than (<) 50.

When input I:012/01 closes, the timer is enabled and starts to time. At time 30, the EQU instruction preceding O:010/00 goes true because the accumulated value in T4:1 is equal to (=) 30, and output O:010/00 in Rung 2 is energized. This is only true when the accumulated value of timer T4:1 is 30. When it advances to 31, the EQU instruction goes false and O:010/00 goes *OFF*. Output O:010/01, which was *ON* because of the not equal to (≠) instruction, now goes *OFF* for one second because the NEQ instruction was false when the accumulated value of T4:1 was equal to (=) 30.

When the accumulated time reaches 50, output O:010/02 (Rung 4) turns *ON* because the GEQ instruction preceding it goes true when T4:1.ACC is equal to or greater than (≥) 50. The rung is true when the accumulated value in T4:1 is equal to (=) 50 and remains true as long as the accumulated value is 50 or greater. Output O:010/02 remains *ON* until the timer is turned *OFF* and the accumulated value is reset to 00.

Output O:010/03, that was *ON*, now goes *OFF* when the accumulated value of T4:1 reaches 50 because the LES instruction that precedes it is only true when the value of Source A is less than (<) 50.

Output O:010/04, the timer done bit (DN), comes *ON* at 100 when the accumulated value equals the preset value. The time chart in Figure 14–16 illustrates the *ON* and *OFF* states of the outputs in relation to time and the DATA COMPARE instructions.

The Allen-Bradley PLC-5 has an instruction that is a combination of the previous instructions, it is the **compare instruction**, or **CMP**. The CMP instruction is an input instruction that compares values from addresses or files. Figure 14–17 shows a CMP instruction and the compare expression is designated as T4:0.ACC = N7:2.

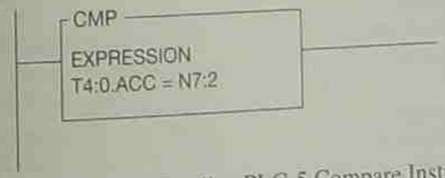


Figure 14–17 Allen-Bradley PLC-5 Compare Instruction

The CMP instruction in this case is true only when the accumulated value of T4:0 is equal to the value found in integer file 7, word 2.

The table in Figure 14–18 shows the different operators (symbols) that the CMP instruction uses. Because standard computer keyboards do not have keys for not equal, less than or equal to, or a greater than or equal to, the CMP instruction uses variations as shown in Figure 14–19.

OPERATOR	DESCRIPTION	EXAMPLE
=	EQUAL TO	TRUE IF A = B
< >	NOT EQUAL TO	TRUE IF A < > B
<	LESS THAN	TRUE IF A < B
< =	LESS THAN OR EQUAL TO	TRUE IF A < = B
>	GREATER THAN	TRUE IF A > B
> =	GREATER THAN OR EQUAL TO	TRUE IF A > = B

Figure 14–18 Available Operators (Symbols) for CMP Instruction

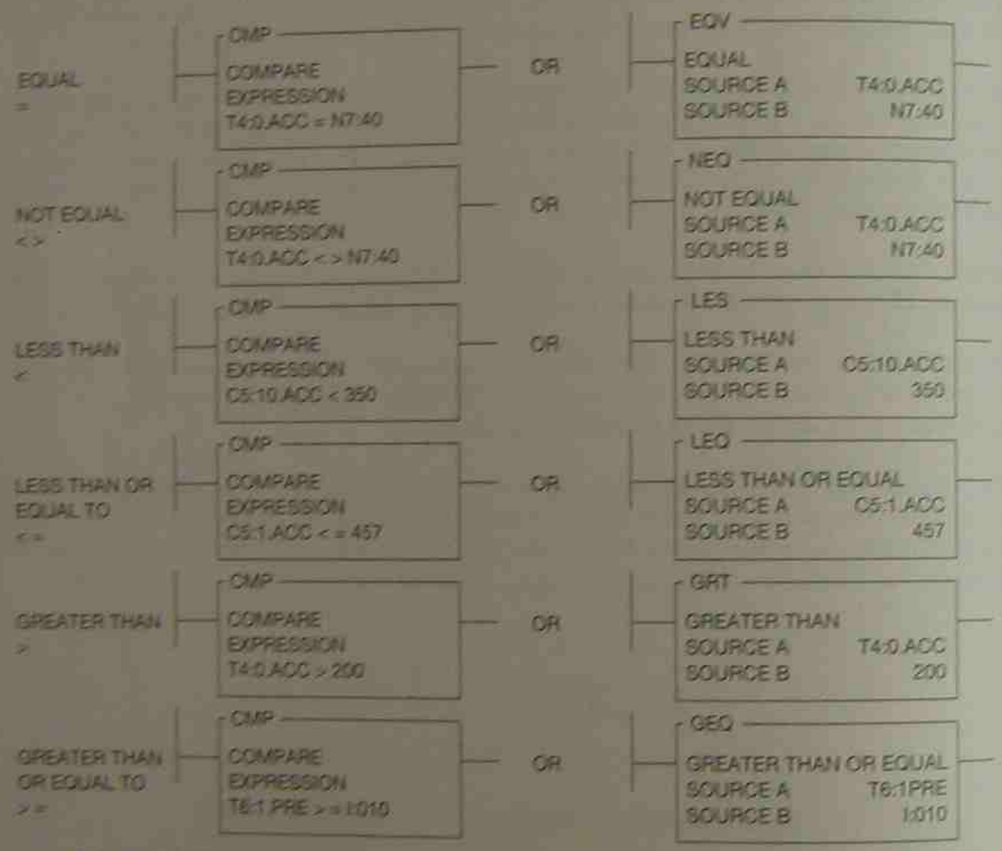


Figure 14–19 Comparing the PLC-5 CMP Instruction to Data Compare Instruction

Note: *While the CMP instruction duplicates the other data compare instructions, the execution time for the CMP instruction is longer than the execution time for equivalent comparison instructions (for example, GRT, LEQ, etc.). A CMP instruction also uses more words per instruction than the equivalent comparison instructions. The advantage, however, is that the CMP instruction does all of the compare functions that are needed, without remembering all the instruction mnemonics.*

The LIM, or limit test instruction, is an input instruction used by Allen-Bradley to test for values inside or outside a specific range. The instruction is false until it detects that the test value is within certain limits. It then goes true. When the instruction detects that the test value has again gone outside the prescribed limits, the instruction goes false. This instruction is perfect for monitoring analog signals and making program decisions based on the analog value(s). Figure 14–20 shows the format for a LIM instruction.

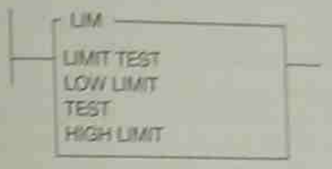


Figure 14–20 PLC-5 Limit Test Instruction Format

To program a LIM instruction, the following information must be provided:

Low Limit
Low limit is a constant, or an address that determines the lower limit of the test range. The value in the low limit can be either integer (whole number) or floating point (number[s] and a decimal).

Test Value
The test value is the address that contains the value examined to determine if it is inside or outside the specified range.

High Limit
High limit is a constant (numeric value) or an address that determines the upper limit of the test range. The value in the high limit can be either integer or floating point value.

An example of how a LIM instruction is used is shown in Figure 14–21. In the example, the LIM instruction is used to turn *ON* an indicator lamp whenever the accumulated value of T4:1 is between the values stored in N7:10 and N7:20.

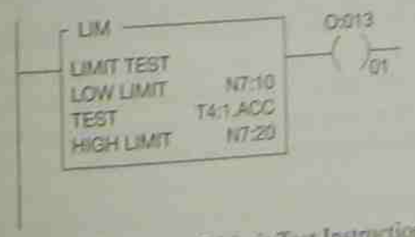


Figure 14–21 PLC-5 Limit Test Instruction

If the lower limit was set to 100 (value stored in N7:10) and the upper limit was set for 300 (value stored in N7:20), the instruction will be false as long as the accumulated value of T4:1 was less than 100 or greater than 300. When the accumulated value of T4:1 reached 100 and became equal to or greater than the low limit, the instruction would become true and indicator lamp (output O:013/01) would be set to 1, or turned *ON*. If the accumulated value of T4:1 became greater than 300, the instruction would again go false and the indicator lamp (O:013/01) would be turned *OFF*. The instruction remains *OFF* as long as the accumulated value of T4:1 is outside the limit test range (100-300) that was established for the LIM instruction.

The values that are used for the low limit and high limit can be entered as numeric values when the instruction is being programmed. Instead of referencing N7:20 (which held a value of 300), a value of 300 could have been entered at the low limit prompt.

Another example of the limit test instruction (LIM) is shown in Figure 14–22.

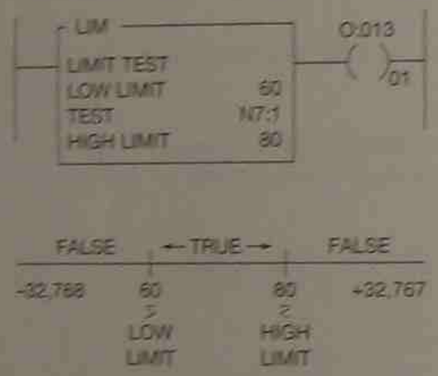


Figure 14–22 LIM Instruction With the Low Limit Value Set at 60

The Low Limit is a constant with a value of 60 whereas the High Limit has a constant value of 80. The Test address is N7:1. This could be the numeric value from an analog input device such as a thermocouple or resistive temperature device (RTD). Programmed this way, the instruction is true, and output device O:013/01 is *ON*, as long as the Test value is equal to or greater than the Low Limit of 60 and less than or equal to the High Limit of 80. Any time the value stored in N7:1 is below 60 or greater than 80, the instruction is false and output device O:013/01 is *OFF*. The output device could be an indicator lamp that is lit as long as the temperature value measured by the thermocouple is within the specified range of 60 to 80 degrees.

If a LIM instruction is programmed with the Low Limit value higher than the High Limit value, as shown in Figure 14–23, the logic of the instruction is reversed.

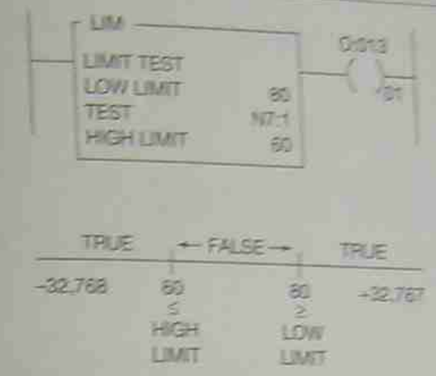


Figure 14–23 LIM Instruction With the Low Limit Value Higher Than the High Limit Value

In this illustration, the Low Limit has been set at 80, and the High Limit has been set at 60. The Test value is still the value stored in Integer file N7 word 1. Programmed this way, the logic of the LIM instruction is true when the value stored in N7:1 is equal to or greater than 80 (Low Limit) and also true when the value is less than or equal to 60 (High Limit). Figure 14–23 illustrates the logic for the instruction. Output device O:013/01 will be *ON* anytime the value stored in N7:1 is equal to or less than 60 and also will be *ON* anytime the value in N7:1 is equal to or greater than 80.

Once the electrician or technician becomes familiar with a specific PLC, the many applications and advantages of using the various data compare instructions become evident.

Chapter Summary

Although formats and instructions vary with each PLC manufacturer, the concepts of data manipulation are the same. Data manipulation enables an operator to transfer data from one word location to another, whereas data comparison allows a value in one word to be compared to another word or a constant value.

Both data transfer and data comparison instructions give new dimension and flexibility to motor-control circuits, and the application of either is only limited by programmer imagination.

Review Questions

1. Define the term *data transfer*.
2. When numerical information replaces data that already exists in a memory location it is referred to as:
   a. exchanging info (data)
   b. replacement programming
   c. blanket move
   d. writing over