

1

Programmable Logic Controllers (PLCs) An Overview



Photo courtesy Rockwell Automation, Inc.

Chapter Objectives

After completing this chapter, you will be able to:

- 1.1** Define what a programmable logic controller (PLC) is and list its advantages over relay systems
- 1.2** Identify the main parts of a PLC and describe their functions
- 1.3** Outline the basic sequence of operation for a PLC
- 1.4** Identify the general classifications of PLCs

This chapter gives a brief history of the evolution of the programmable logic controller, or PLC. The reasons for changing from relay control systems to PLCs are discussed. You will learn the basic parts of a PLC, how a PLC is used to control a process, and the different kinds of PLCs and their applications. The ladder logic language, which was developed to simplify the task of programming PLCs, is introduced.

Each chapter begins with a brief introduction outlining chapter coverage and learning objectives.



Figure 2-49 Human Machine Interfaces (HMIs).

Source: Photo courtesy Omron Industrial Automation, www.ia.omron.com.

time. Through personal computer-based setup software, you can configure display screens to:

- Replace hardwired pushbuttons and pilot lights with realistic-looking icons. The machine operator needs to push
- Show operations in graphic format for easier viewing.
- Allow the operator to change timer and counter presets by touching the numeric keypad graphic on the touch screen.
- Show alarms, complete with time of occurrence and location.
- Display variables as they change over time.

The Allen-Bradley Pico GFX-70 controller, shown in Figure 2-50, serves as a controller with HMI capabilities.



Figure 2-50 Allen-Bradley Pico GFX-70 controller.

Source: Photo courtesy Rockwell Automation, Inc.

Human machine interfacing with Pico controllers added.

of three modular parts: an HMI, processor/power supply, and I/O modules.

The display/keypad can be used as an operator interface or can be linked to control operations to provide real-time feedback. It has the ability to show text, date and time, as well as custom messages and bitmap graphics, allowing operators to acknowledge fault messages, enter values, and initiate actions. Users can create both the control program and HMI functionality using a personal computer with PicoSoft Pro software installed or the controller's on-board display buttons.

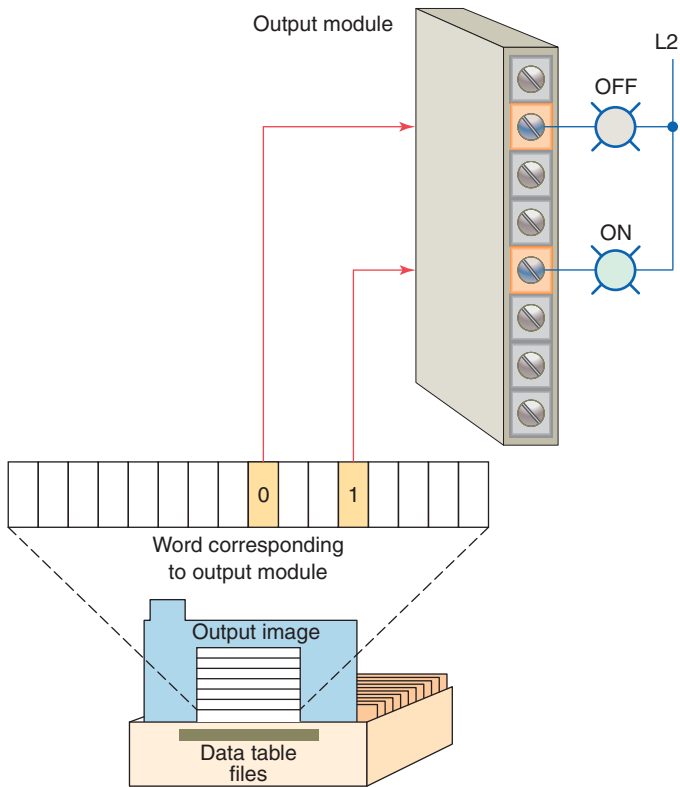


Figure 5-5 Connections of pilot lights to the output image table file through the output module.

connection of two pilot lights to the output image table file through the output module. Its operation can be summarized as follows.

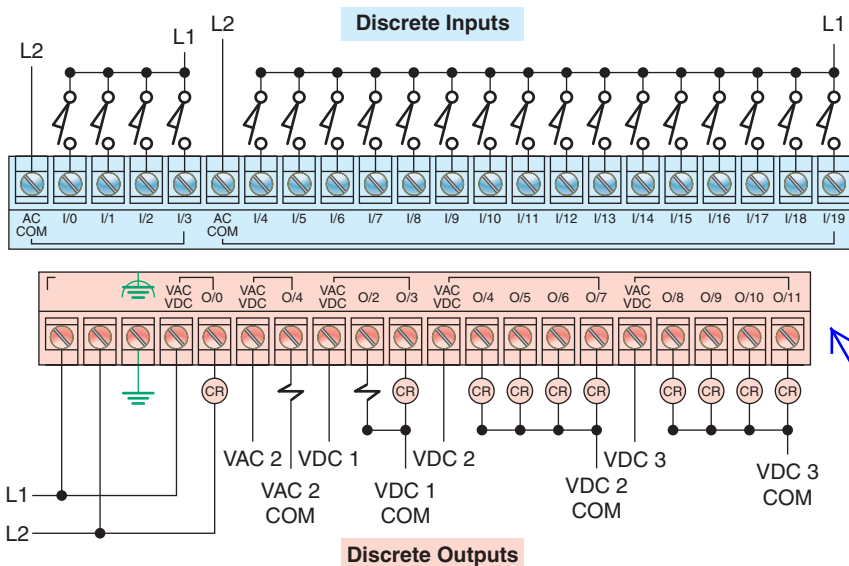
- The status of each light (ON/OFF) is controlled by the user program and is indicated by the presence of 1 (ON) and 0 (OFF).

- Each connected output has a bit in the output image table file that corresponds exactly to the terminal to which the output is connected.
- If the program calls for a specific output to be ON, its corresponding bit in the table is set to 1.
- If the program calls for the output to be OFF, its corresponding bit in the table is set to 0.
- The processor continually activates or deactivates the output status according to the output table file status.

Typically, micro PLCs have a fixed number of inputs and outputs. Figure 5-6 shows the MicroLogic controller from the Allen-Bradley MicroLogic 1000 family of controllers. The controller has 20 discrete inputs with predefined addresses I/0 through I/19 and 12 discrete outputs with predefined addresses O/1 through O/11. Some units also contain analog inputs and outputs embedded into the base unit or available through add-on modules.

5.2 Program Scan

When a PLC executes a program, it must know—in real time—when external devices controlling a process are changing. During each operating cycle, the processor reads all the inputs, takes these values, and energizes or de-energizes the outputs according to the user program. This process is known as a *program scan cycle*. Figure 5-7 illustrates a single PLC operating cycle consisting of the *input scan*, *program scan*, *output scan*, and housekeeping duties. Because the inputs can change at any time, it constantly repeats this cycle as long as the PLC is in the RUN mode.



Addressing of a micro PLC illustrated.

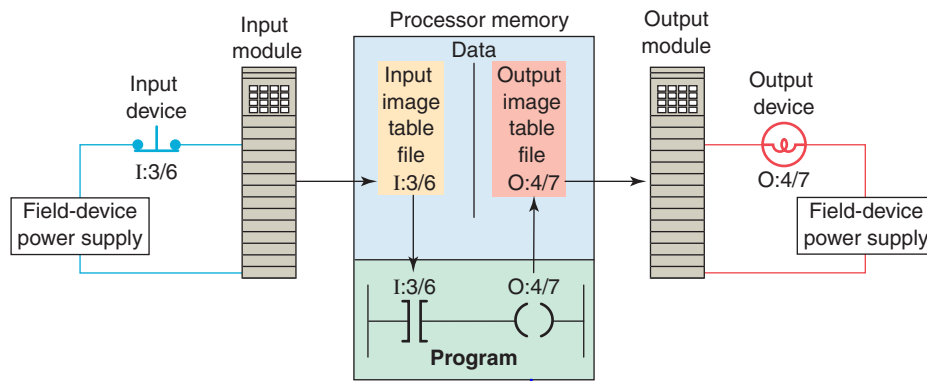


Figure 5-9 Scan process applied to a single rung program.

- During the program scan, the processor examines bit I:3/6 for a 1 (ON) condition.
- In this case, because input I:3/6 is 1, the rung is said to be TRUE or have *logic continuity*.
- The processor then sets the output image table bit O:4/7 to 1.
- The processor turns on output O:4/7 during the next I/O scan, and the output device (light) wired to this terminal becomes energized.
- This process is repeated as long as the processor is in the RUN mode.
- If the input device opens, electrical continuity is lost, and a 0 would be placed in the input image table. As a result, the rung is said to be FALSE due to loss of logic continuity.
- The processor would then set the output image table bit O:4/7 to 0, causing the output device to turn off.

Step 3 The final step of the scan process is to update the actual states of the output devices by transferring the output table results to the output module, thereby switching the connected output devices ON (1) or OFF (0). If the status of

Program scan process explained in greater detail.

processor will not scan.

Each instruction entered into a program requires a certain amount of time for the instruction to be executed. The amount of time required depends on the instruction. For example, it takes less time for a processor to read the status of an input contact than it does to read the accumulated value of a timer or counter. The time taken to scan

Ladder programs process inputs at the beginning of a scan and outputs at the end of a scan, as illustrated in Figure 5-10. For each rung executed, the PLC processor will:

Step 1 Update the input image table by sensing the voltage of the input terminals. Based on the absence or presence of a voltage, a 0 or a 1 is stored into the memory bit location designated for a particular input terminal.

Step 2 Solve the ladder logic in order to determine logical continuity. The processor scans the ladder program and evaluates the logical continuity of each rung by referring to the input image table to see if the input conditions are met. If the conditions controlling an output are met, the processor immediately writes a 1 in its memory location, indicating that the output will be turned ON; conversely, if the conditions are not met a 0 indicating that the device will be turned OFF is written into its memory location.

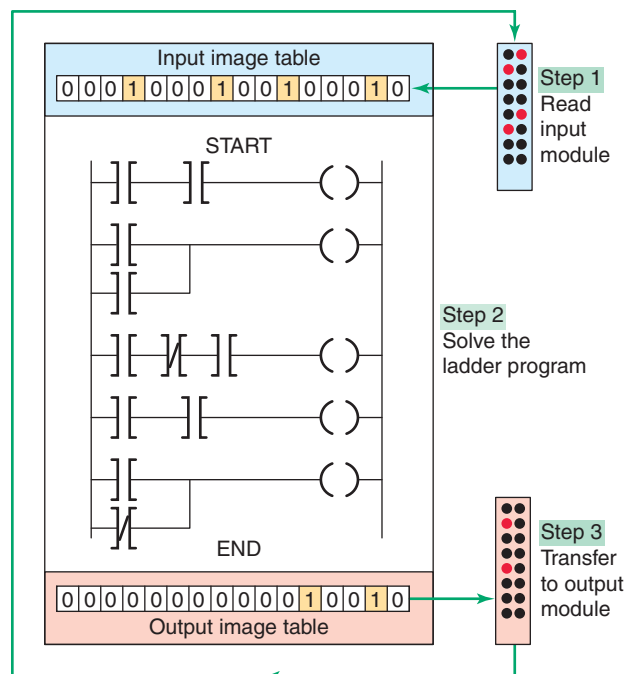


Figure 5-10 Scan process applied to a multiple rung program.



Motor/controller

Wiring of field inputs and outputs to a micro PLC illustrated.

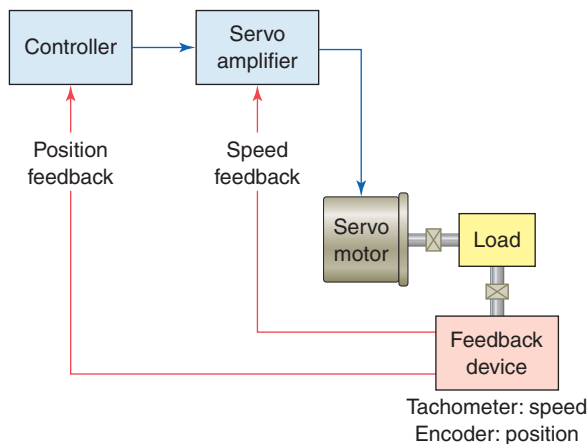


Figure 6-43 Closed-loop servo motor system.
Source: Photos courtesy Omron Industrial Automation, www.ia.omron.com.

The motor stop/start circuit shown in Figure 6-44 is a typical example of a seal-in circuit. The hardwired circuit consists of a normally closed stop button in series with a normally open start button. The seal-in auxiliary contact of the starter is connected in parallel with the start button to keep the starter coil energized when the start button is released. When this circuit is programmed into a PLC, both the start and stop buttons are examined for a closed condition because both buttons must be closed to cause the motor starter to operate.

Figure 6-45 shows a PLC wiring diagram of the motor seal-in circuit implemented using an Allen-Bradley Pico controller. The controller is programmed using ladder logic. Each programming element can be entered directly

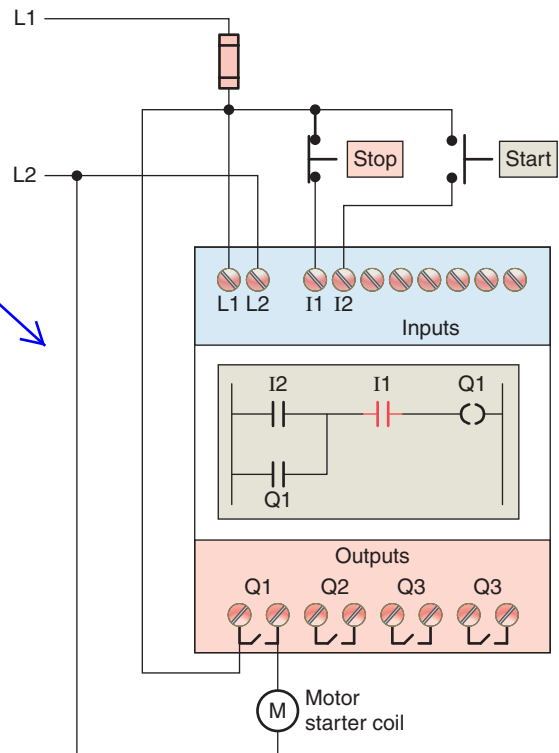


Figure 6-45 Motor seal-in circuit implemented using an Allen-Bradley Pico controller.

via the Pico display. This controller also lets you program the circuit from a personal computer using PicoSoft programming software.

6.9 Latching Relays

Electromagnetic latching relays are designed to hold the relay closed after power has been removed from the coil. Latching relays are used where it is necessary for contacts to stay open and/or closed even though the coil is energized only momentarily. Figure 6-46 shows a latching relay that uses two coils. The *latch* coil is momentarily energized to set the latch and hold the relay in the latched position. The *unlatch* or release coil is momentarily

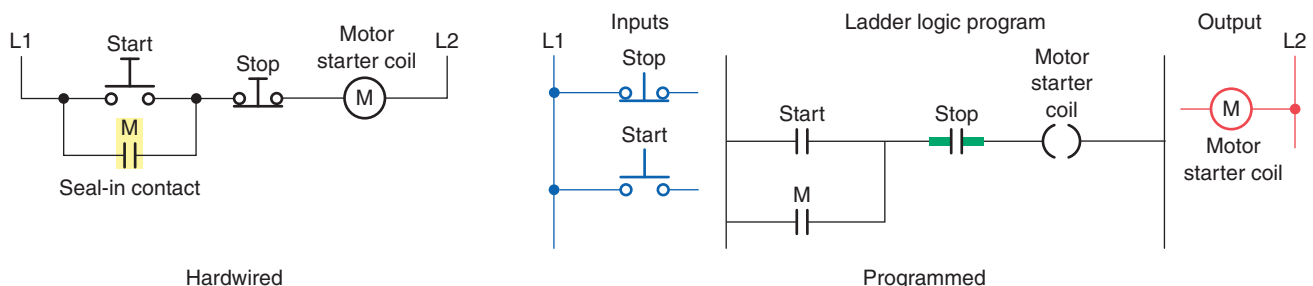


Figure 6-44 Hardwired and programmed seal-in circuit.

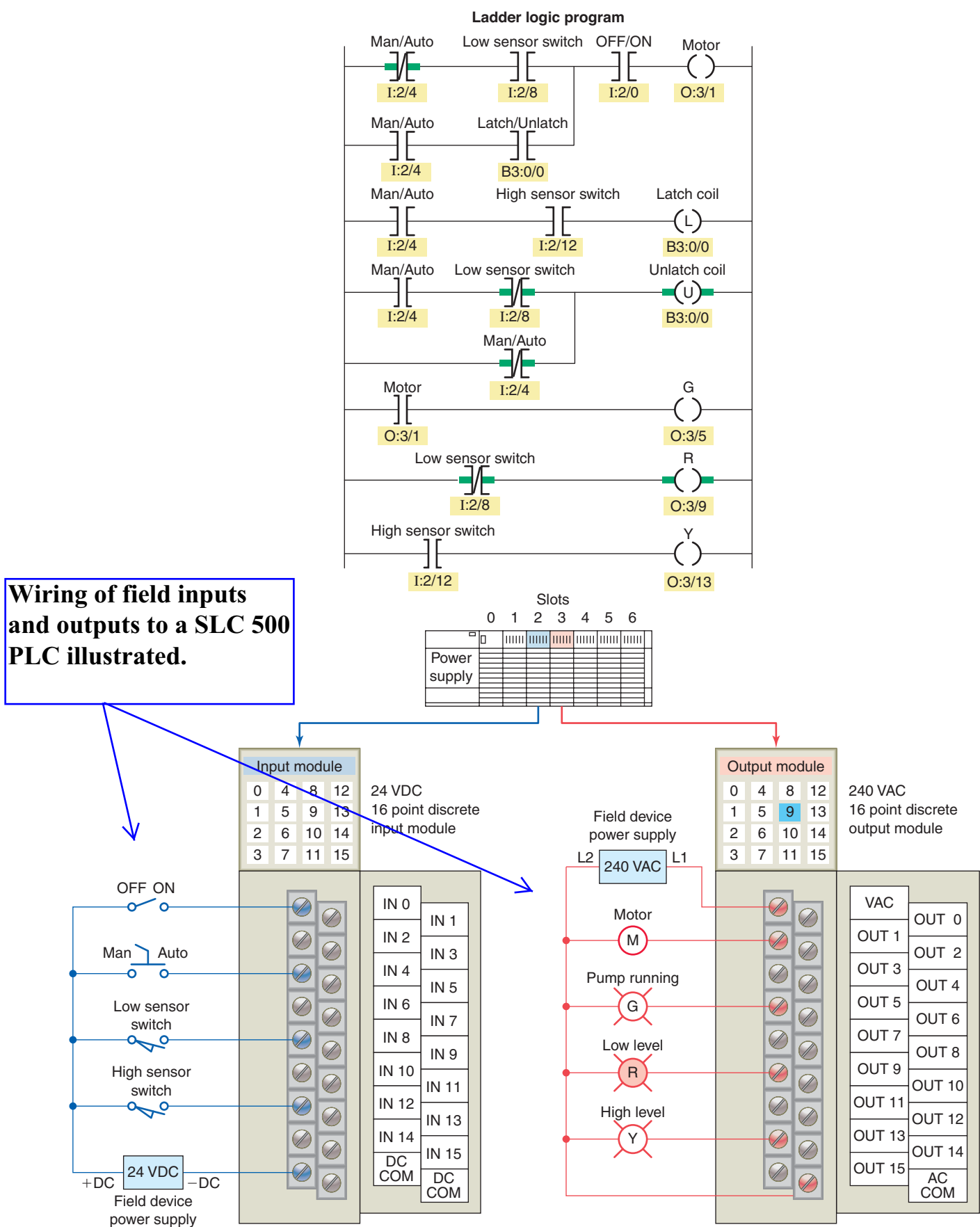


Figure 6-52 Water level control program implemented using an Allen-Bradley modular SLC 500 controller.

EXAMPLE 6-3

Figure 6-67 shows the sketch of a continuous filling operation. This process requires that boxes moving on a conveyor be automatically positioned and filled.

The sequence of operation for the continuous filling operation is as follows:

- Start the conveyor when the start button is momentarily pressed.
- Stop the conveyor when the stop button is momentarily pressed.
- Energize the run status light when the process is operating.
- Energize the standby status light when the process is stopped.
- Stop the conveyor when the right edge of the box is first sensed by the photosensor.
- With the box in position and the conveyor stopped, open the solenoid valve and allow the box to fill. Filling should stop when the level sensor goes true.
- Energize the full light when the box is full. The full light should remain energized until the box is moved clear of the photosensor.

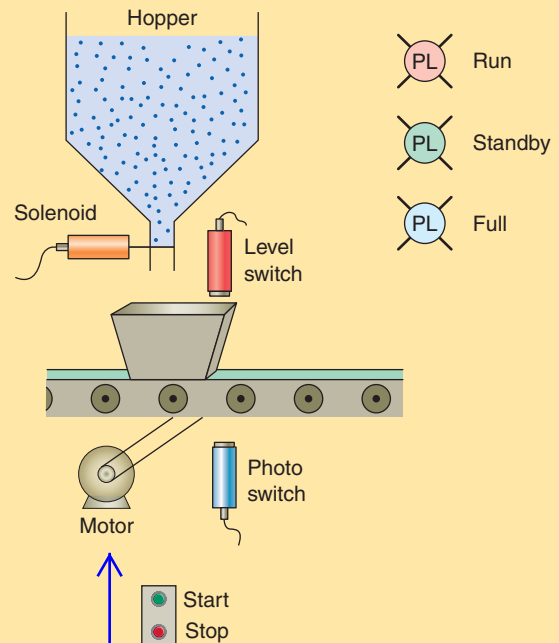


Figure 6-67 Sketch of the continuous filling operation.

Figure 6-68 shows the ladder logic program for the operation.

Examples of programs include a sketch of the process.

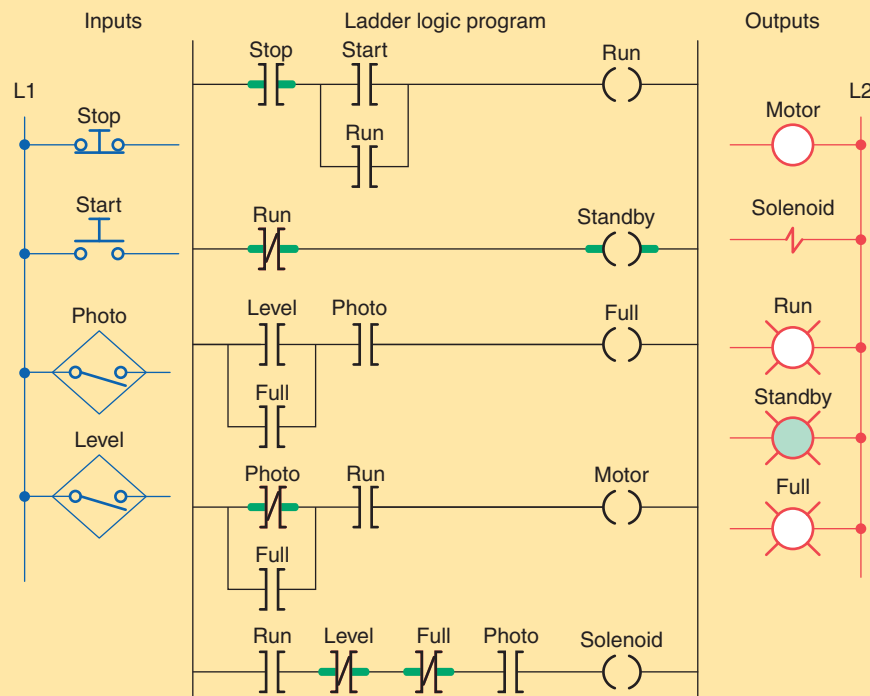


Figure 6-68 Continuous filling operation PLC program.

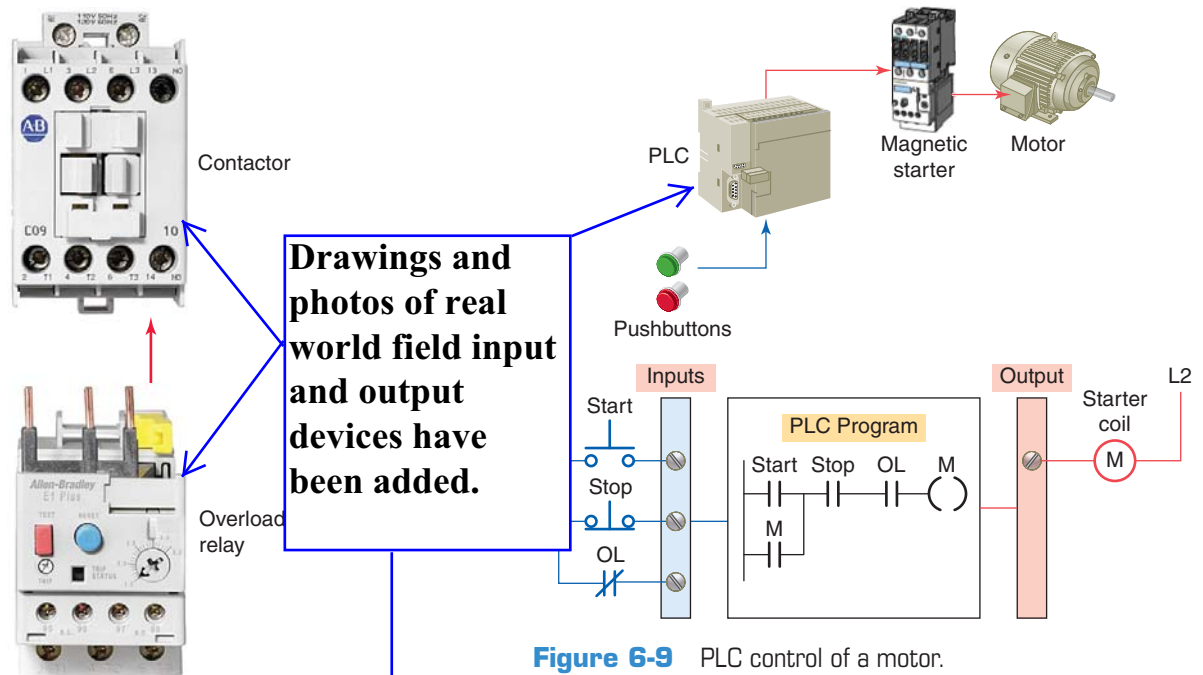


Figure 6-9 PLC control of a motor.

Figure 6-7 Motor starter is a contactor with an attached overload relay.

Source: Photo courtesy Rockwell Automation, Inc.

- Control contact M (across START button) closes to seal in the coil circuit when the START button is released. This contact is part of the *control* circuit and, as such, is only required to handle the small amount of current needed to energize the coil.
- An overload (OL) relay is provided to protect the motor against current overloads. The normally closed relay contact OL opens automatically when

an overload current is sensed to de-energize the M coil and stop the motor.

Motor starters are available in various standard National Electric Manufacturers Association (NEMA) sizes and ratings. When a PLC needs to control a large motor, it must work in conjunction with a starter as illustrated in Figure 6-9. The power requirements for the starter coil must be within the power rating of the output module of the PLC. Note that the control logic is determined and executed by the program within the PLC and not by the hardwired arrangement of the input control devices.

6.4 Manually Operated Switches

Manually operated switches are controlled by hand. These include toggle switches, pushbutton switches, knife switches, and selector switches.

Pushbutton switches are the most common form of manual control. A pushbutton operates by opening or closing contacts when pressed. Figure 6-10 shows commonly used types of pushbutton switches, which include:

- Normally open (NO) pushbutton*, which makes a circuit when it is pressed and returns to its open position when the button is released.
- Normally closed (NC) pushbutton*, which opens the circuit when it is pressed and returns to the closed position when the button is released.
- Break-before-make pushbutton* in which the top section contacts are NC and the bottom section contacts are NO. When the button is pressed, the top contacts open before the bottom contacts are closed.

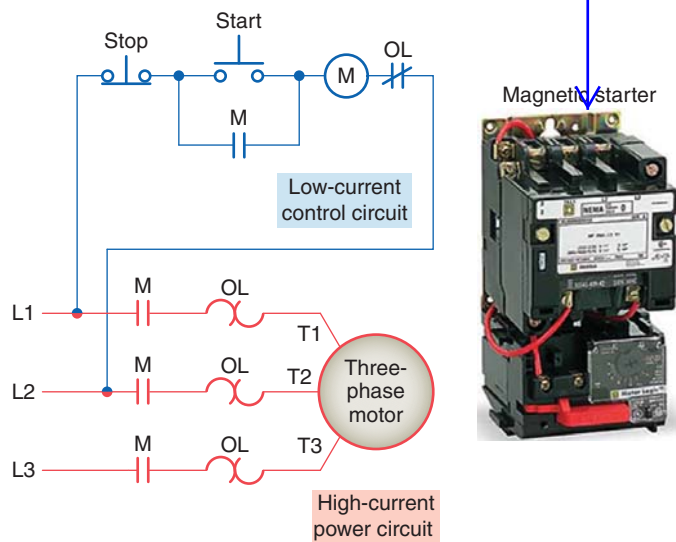


Figure 6-8 Three-phase magnetic motor starter.

Source: This material and associated copyrights are proprietary to, and used with the permission of Schneider Electric.

The relay equivalent of the virtual programmed instruction is explained first, followed by the appropriate PLC program.

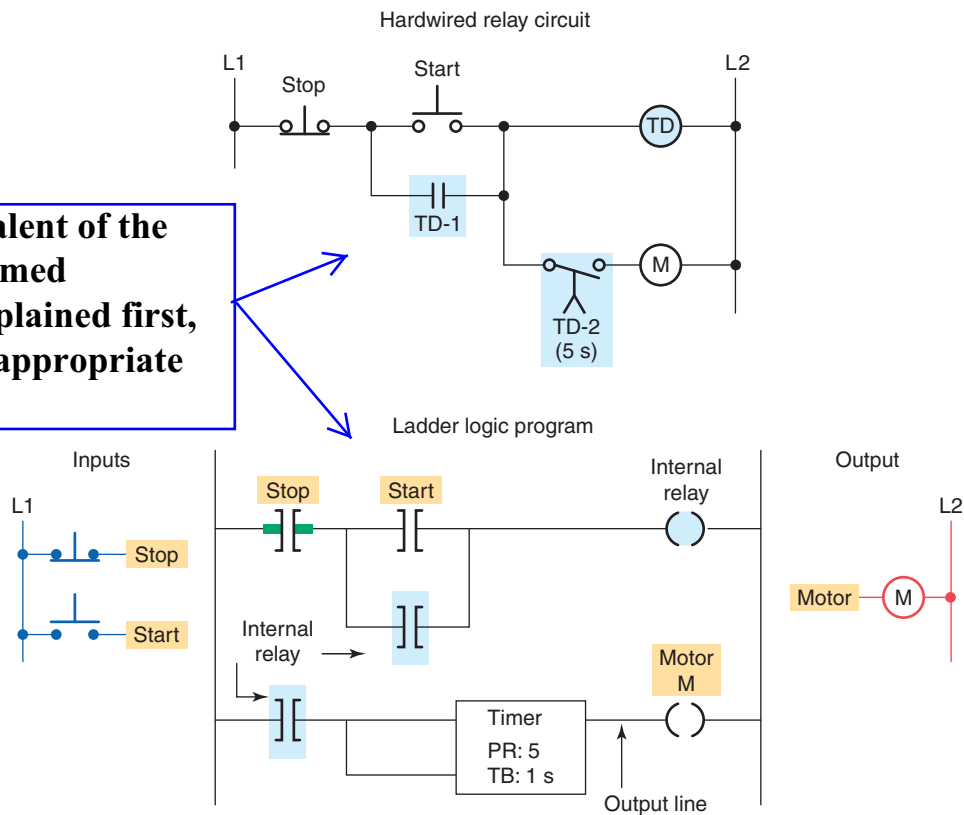


Figure 7-17 Instantaneous contact instruction can be programmed using an internally referenced relay coil.

- Contact TD-1 is the instantaneous contact, and contact TD-2 is the timed contact.
- The ladder logic program shows that a contact instruction referenced to an internal relay is now used to operate the timer.
- The instantaneous contact is referenced to the internal relay coil, whereas the time-delay contact is referenced to the timer output coil.

Figure 7-18 shows an application for an on-delay timer that uses an NCTO contact. This circuit is used as a warning signal when moving equipment, such as a conveyor motor, is about to be started. The operation of the circuit can be summarized as follows:

- According to the hardwired relay circuit diagram, coil CR is energized when the start pushbutton PB1 is momentarily actuated.
- As a result, contact CR-1 closes to seal in CR coil, contact CR-2 closes to energize timer coil TD, and contact CR-3 closes to sound the horn.
- After a 10-s time-delay period, timer contact TD-1 opens to automatically switch the horn off.
- The ladder logic program shows how an equivalent circuit could be programmed using a PLC.

- The logic on the last rung is the same as the timer-timing bit and as such can be used with timers that do not have a timer-timing output.

Timers are often used as part of automatic sequential control systems. Figure 7-19 shows how a series of motors can be started automatically with only one start/stop control station. The operation of the circuit can be summarized as follows:

- According to the relay ladder schematic, lube-oil pump motor starter coil M1 is energized when the start pushbutton PB2 is momentarily actuated.
- As a result, M1-1 control contact closes to seal in M1, and the lube-oil pump motor starts.
- When the lube-oil pump builds up sufficient oil pressure, the lube-oil pressure switch PS1 closes.
- This in turn energizes coil M2 to start the main drive motor and energizes coil TD to begin the time-delay period.
- After the preset time-delay period of 15 s, TD-1 contact closes to energize coil M3 and start the feed motor.
- The ladder logic program shows how an equivalent circuit could be programmed using a PLC.

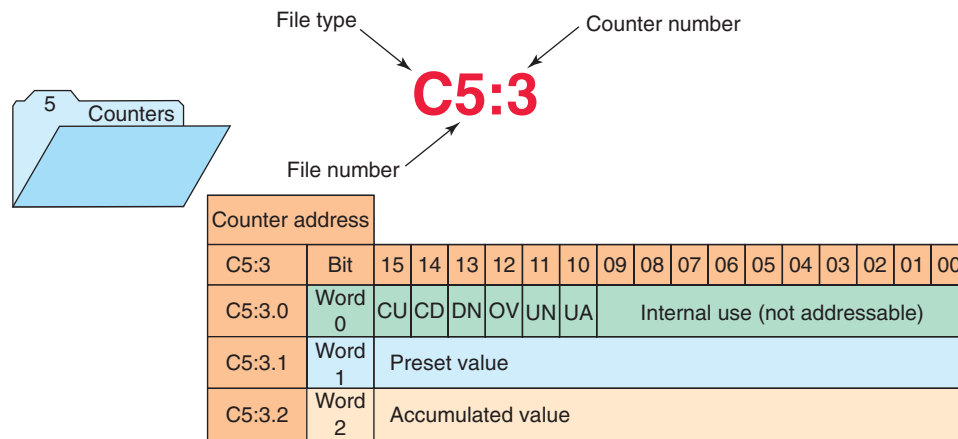


Figure 8-9 SLC 500 counter file.

- Each false-to-true transition of rung 1 increases the counter's accumulated value by 1.
- After 7 pulses, or counts, when the preset counter value equals the accumulated counter value, output DN is energized.
- As a result, the output light is energized.
- At the end of the energized light, the counter resets to zero.
- The counter can resume when rung 4 goes false again.

The Allen-Bradley SLC 500 counter file is file 5 (Figure 8-9). Each counter is composed of three 16-bit words, collectively called a counter element. These three data words are the control word, preset word, and accumulated word. Each of the three data words shares the same base address, which is the address of the counter itself. There can be up to 256 counter elements. Addresses for counter file 5, counter element 3 (C5:3), are listed below.

C5 = counter file 5

:3 = counter element 3 (0–255 counter elements per file)

C5:3/DN is the address for the done bit of the counter.
C5:3/CU is the address for the count-up enable bit of the counter.

C5:3/CD is the address for the count-down enable bit of the counter.

C5:3/OV is the address for the overflow bit of the counter.

C5:3/UN is the address for the underflow bit of the counter.

C5:3/UA is the address for the update accumulator bit of the counter.

Figure 8-10 shows the counter table for the Allen-Bradley SLC 500 controller. The *control word* uses status control bits consisting of the following:

Although the content is of a nature to allow the information to be applied to a variety of PLCs from different manufacturers, this book for the most part uses the Allen-Bradley SLC-500 and ControlLogix controller instruction sets for the programming examples.

Count-Down (CD) Enable Bit—The count-down enable bit is used with the count-down counter and is true whenever the count-down counter instruction is true. If the count-down counter instruction is false, the CD bit is false.

Done (DN) Bit—The done bit is true whenever the accumulated value is equal to or greater than the preset value of the counter for either the count-up or the count-down counter.

Overflow (OV) Bit—The overflow bit is true whenever the counter counts past its maximum value, which is 32,767. On the next count, the counter will wrap around to 32,768 and will continue counting.

Counter Table								
	/CU	/CD	/DN	/OV	/UN	/UA	.PRE	.ACC
C5:0	0	0	0	0	0	0	0	0
C5:1	0	0	0	0	0	0	0	0
C5:2	0	0	0	0	0	0	0	0
C5:3	0	0	0	0	0	0	50	0
C5:4	0	0	0	0	0	0	0	0
C5:5	0	0	0	0	0	0	0	0
Address: C5:3 Table: C5: Counter								

Figure 8-10 SLC 500 counter table.

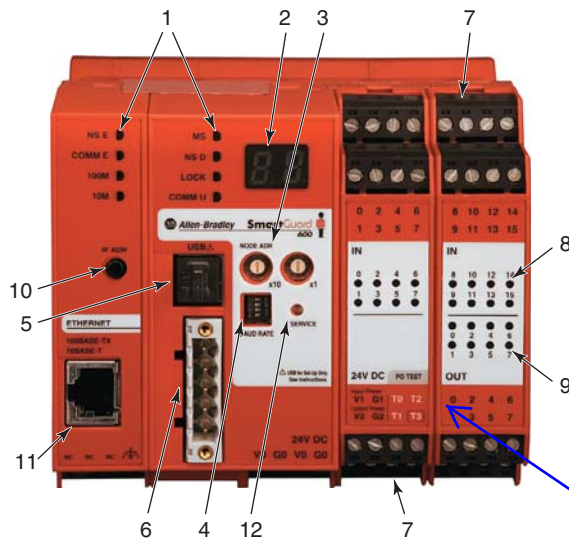


Figure 9-24 Safety PLC.
Source: Image Used with Permission of Rockwell Automation, Inc.

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

The content has been updated and reflects the changes in technology since the publication of the previous edition.

Safety PLCs, such as the one shown in Figure 9-24, are now available for applications that require more advanced safety functionality. A safety PLC is typically certified by third parties to meet rigid safety and reliability requirements of international standards. Both standard and safety PLCs have the ability to perform control functions but a standard PLC was not initially designed to be fault tolerant and fail-safe. That is the fundamental difference.

Some of the differences between standard and safety PLCs include the following:

- A standard PLC has one microprocessor that executes the program, Flash memory area that stores the program, RAM for making calculations, ports for communications, and I/O for detection and control of the machine. In contrast, a safety PLC has redundant microprocessors, Flash and RAM that are continuously monitored by a watchdog circuit, and a synchronous detection circuit. *Redundancy* is duplication. The probability of hazards arising from one malfunction in an electrical circuit can be minimized by creating partial or complete redundancy (duplication).
- Standard PLC inputs provide no internal means for testing the functionality of the input circuitry. By contrast, safety PLCs have an internal output circuit associated with each input for the purpose of testing the input circuitry. Inputs are driven both high and low for very short cycles during runtime to verify their functionality.

- Safety PLCs use power supplies designed specifically for use in safety control systems and redundant backplane circuitry between the controller and I/O modules.

Safety considerations should be developed as part of the PLC program. A PLC program for any application will be only as safe as the time and thought spent on both personnel and hardware considerations make it. One such consideration involves the use of a motor starter **auxiliary seal-in contact**, shown in Figure 9-25, in place of the programmed contact referenced to the output coil instruction. The use of the field-generated starter auxiliary contact status in the program is more costly in terms of field wiring and hardware, but it is *safer* because it provides positive feedback to the processor about the exact status of the motor. Assume, for example, that the OL contact of the starter opens under an overload condition. The motor, of course, would stop operating because power would be lost to the starter coil. If the program was written using an examine-on contact instruction referenced to the output coil instruction as the seal-in for the circuit, the processor would never know that power had been lost to the motor. When the OL was reset, the motor would restart instantly, creating a potentially unsafe operating condition.

Another safety consideration concerns the **wiring of stop buttons**. A stop button is generally considered a safety function as well as an operating function. As such, *all stop buttons should be wired using a normally closed contact programmed to examine for an on*

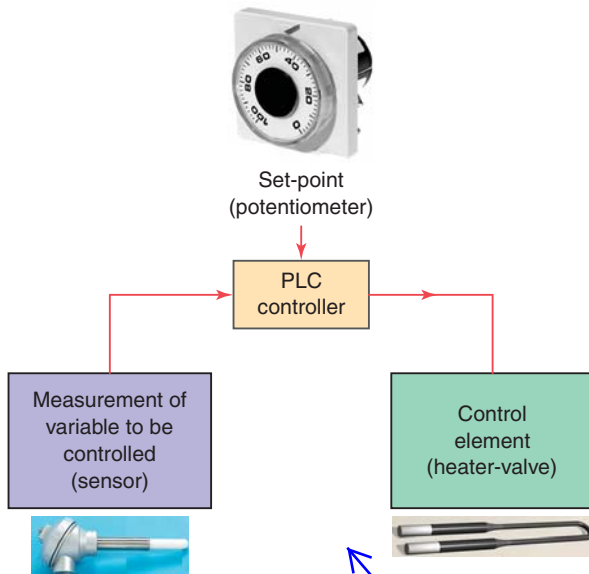


Figure 10-42 Closed-loop control system.

(error) exists between the actual and desired levels, the PLC control program will take the necessary corrective action. Adjustments are made continuously by the PLC until the difference between the desired and actual output is as small as is practical.

With on/off PLC control (also known as *two-position* and *bang-bang control*), the output or final control element is either on or off—one for the occasion when the value of the measured variable is above the set-point and the other for the occasion when the value is below the set-point. The controller will never keep the final element in an intermediate position. Most residential thermostats are on/off type controllers.

On/off control is inexpensive but not accurate enough for most process and machine control applications. On/off control almost always means overshoot and resultant system cycling. For this reason a *deadband* usually exists around the set-point. The deadband or hysteresis of the control loop is the difference between the on and off operating points.

Proportional controls are designed to eliminate the hunting or cycling associated with on/off control. They allow the final control element to take intermediate positions between on and off. This permits *analog control* of the final control element to vary the amount of energy to the process, depending on how much the value of the measured variable has shifted from the desired value.

The process illustrated in Figure 10-43 is an example of a proportional control process. The PLC analog output module controls the amount of fluid placed in the holding tank by adjusting the percentage of valve opening. The valve is initially open 100 percent. As the fluid level in the tank approaches the preset point, the processor modifies

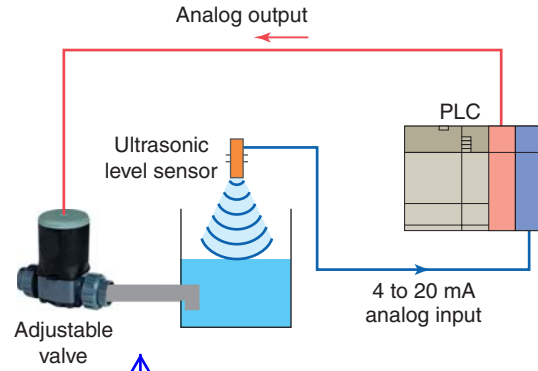


Figure 10-43 Proportional control process.

the output to degrade closing the valve by different percentages, adjusting the valve to maintain a set-point.

Proportional-integral-derivative (PID) control is the most sophisticated and widely used type of process control. PID operations are more complex and are mathematically based. PID controllers produce outputs that depend on the *magnitude, duration, and rate of change* of the system error signal. Sudden system disturbances are met with an aggressive attempt to correct the condition. A PID controller can reduce the system error to 0 faster than any other controller.

A typical PID control loop is illustrated in Figure 10-44. The loop measures the process, compares it to a set-point, and then manipulates the output in the direction which

Analog control covered in more detail.

The terminology can be summarized as follows:

- Operating information that the controller receives from the machine is called the *process variable (PV)* or *feedback*.
- Input from the operator that tells the controller the desired operating point is called the *set-point (SP)*.
- When operating, the controller determines whether the machine needs adjustment by comparing (by subtraction) the set-point and the process variable

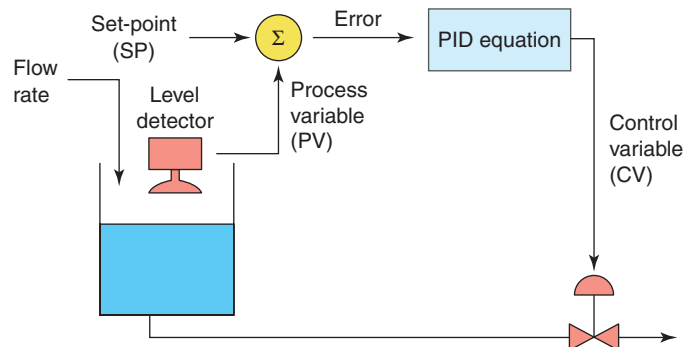


Figure 10-44 Typical PID control loop.

Length—Is the number of steps of the sequencer file starting at position 1. Position 0 is the start-up position. The instruction resets (wraps) to position 1 at each cycle completion. The actual file length will be 1 plus the file length entered in the instruction.

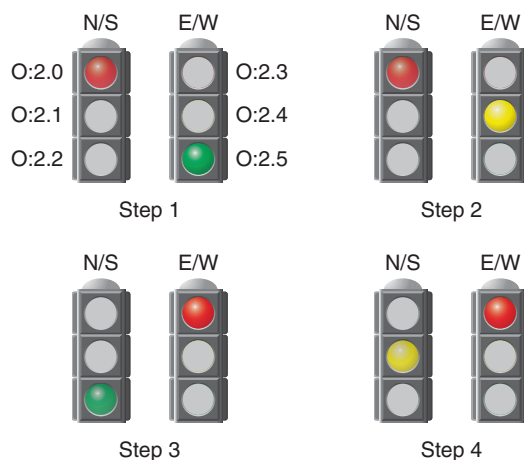
Position—Indicates the step that is desired to start the sequencer instruction. The position is the word location or step in the sequencer file from which the instruction moves data. Any value up to the file length may be entered, but the instruction will always reset to 1 on the true-to-false transition after the instruction has operated on the last position. Before we start the sequence, we need a starting point at which the sequencer is in a neutral position. The start position is all zeros, representing this neutral position; thus, all outputs will be off in position 0.

To program a sequencer, binary information is first entered into the sequencer file or register made up of a series of consecutive memory words. The sequencer file is typically a bit file that contains one bit file word representing the output action required for each step of the sequence. Data are entered for each sequencer step according to the

requirements of the control application. As the sequencer advances through the steps, binary information is transferred from the sequencer file to the output word.

To illustrate the purpose and function of the sequencer file we will examine the operation of the four-step sequence process shown in Figure 12-8. This sequencer is to be used to control traffic in two directions. The operation of the process can be summarized as follows:

- Six outputs are to be energized from one 16-point output module.
- Each light is controlled by one bit address of output word O:2.
- The first 6 bits are programmed to execute the following sequence of light outputs:
 - **Step 1:** Outputs O:2.0 (red) and O:2.5 (green) lights will be energized.
 - **Step 2:** Outputs O:2.0 (red) and O:2.4 (yellow) will be energized.
 - **Step 3:** Outputs O:2.2 (green) and O:2.3 (red) will be energized.
 - **Step 4:** Outputs O:2.1 (yellow) and O:2.3 (red) will be energized.



How Programs Operate
When the operation of a program is called for, a bulleted list is used to summarize its execution.

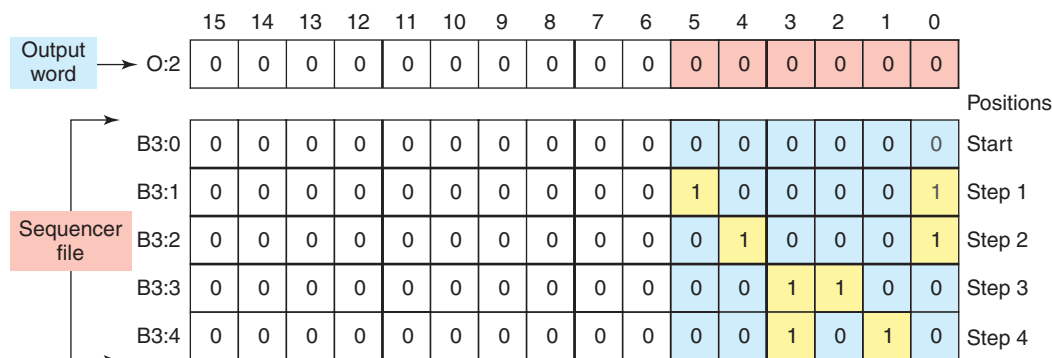


Figure 12-8 Four-step sequencer.

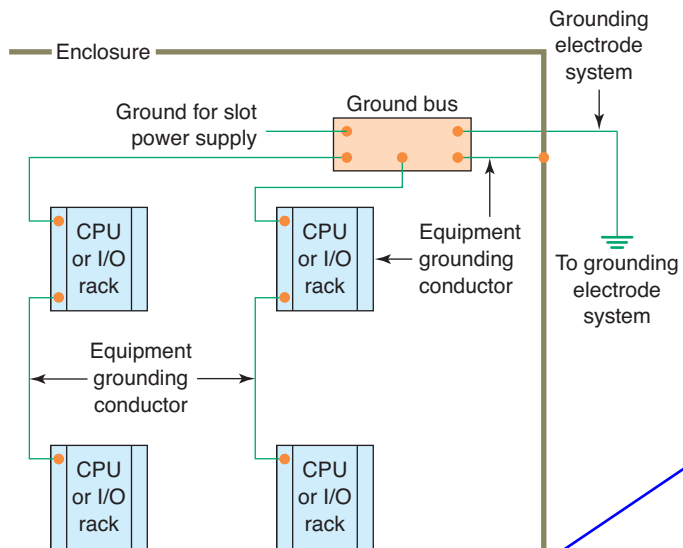


Figure 13-8 PLC grounding system.

of components and connections necessary to provide detailed information for the PLC grounding diagrams. In addition, the diagrams provide detailed information for the PLC grounding methods to use in an enclosure.

Figure 13-8 illustrates a PLC grounding system. A properly installed grounding system will provide a low-impedance path to earth ground. The complete PLC installation, including enclosures, CPU and I/O chassis, and power supplies are all connected to a single low-impedance ground. These connections should exhibit low DC resistance and low high-frequency impedance. A central ground bus bar is provided as a single point of reference inside the enclosure to which all chassis and power supply equipment grounding conductors are connected. The ground bus is then connected to the building's earth ground.

In the event of a high value of ground current, the temperature of the conductor could cause the solder to melt, resulting in interruption of the ground connection. Therefore the grounding path must be permanent (no solder), continuous, and able to conduct safely the ground-fault current in the system with minimal impedance. Paint or other nonconductive material should be scraped away from the area where a chassis makes contact with the enclosure. The minimum ground wire size should be No. 12 AWG stranded copper for PLC equipment grounds and No. 8 AWG stranded copper for enclosure backplane grounds. Ground connections should be made with a star washer between the grounding wire and lug and metal enclosure surface, as illustrated in Figure 13-9.

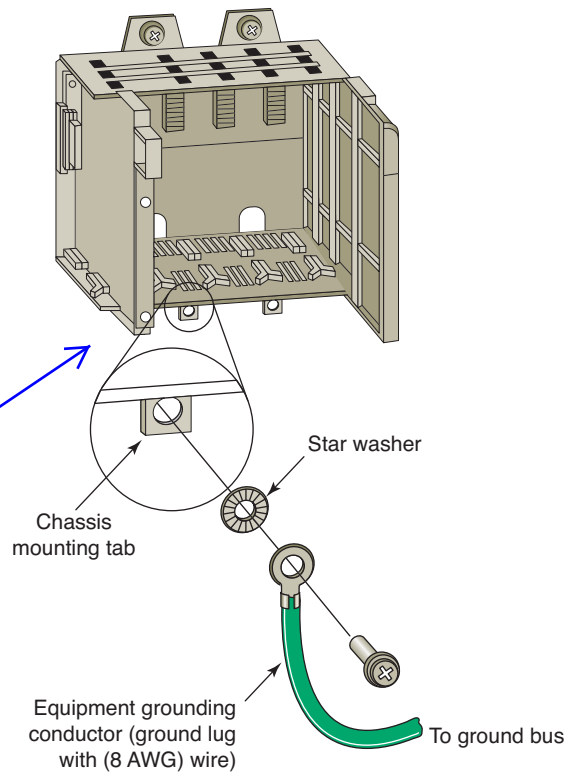


Figure 13-9 Make ground connections using a star washer.

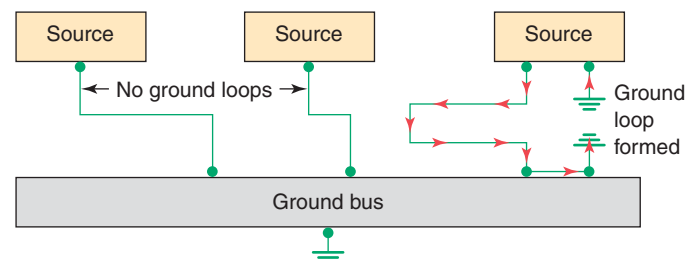


Figure 13-10 Formation of ground loops.

Ground loops can cause problems by adding or subtracting current or voltage from input signal devices. A ground loop circuit can develop when each device's ground is tied to a different earth potential thereby allowing current to flow between the grounds, as illustrated in Figure 13-10. If a varying magnetic field passes through one of these ground loops, a voltage is produced and current flows in the loop. The receiving device is unable to differentiate between the wanted and unwanted signals and, thus, can't accurately reflect actual process conditions. Certain connections require shielded cables to help reduce the effects of electrical noise coupling. Each shield should be grounded at one end only, as a shield grounded at both ends forms a ground loop.

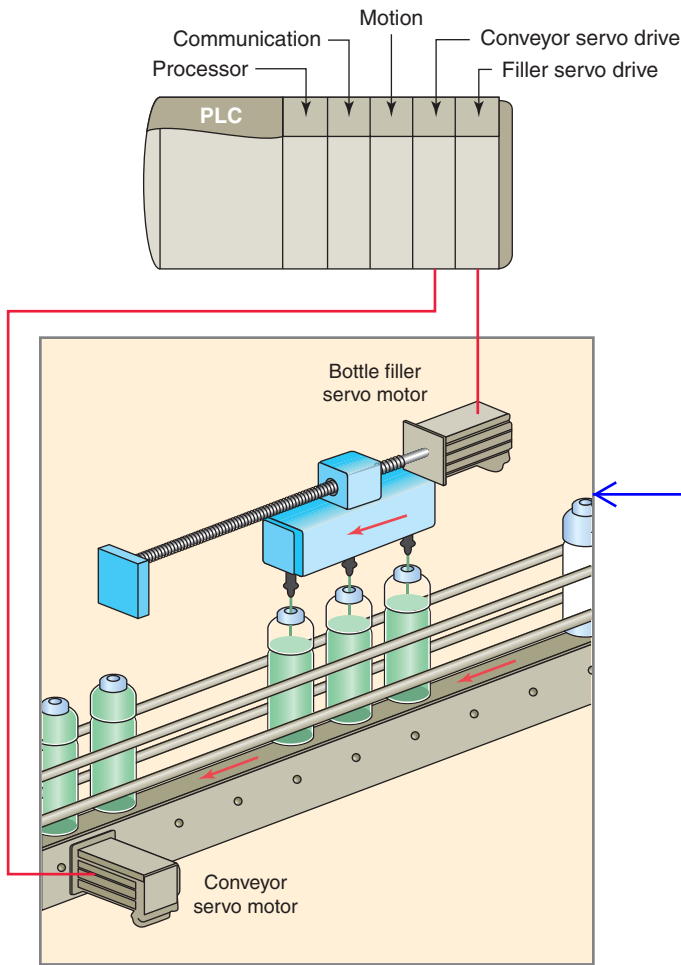


Figure 14-23 Bottle-filling motion control process.

axis of motion. Figure 14-23 illustrates a bottle-filling motion control process. This application requires two axes of motion: the motor operating the bottle filler mechanism and the motor controlling the conveyor speed. The role of each control component can be summarized as follows:

Programmable Logic Controller

- The controller stores and executes the user program that controls the process.
- This program includes motion instructions that control axis movements.
- When the controller encounters a motion instruction it calculates the motion commands for the axis.
- A motion command represents the desired position, velocity, or torque of the servo motor at the particular time the calculations take place.

Motion Module

- The motion module receives motion commands from the controller and transforms them into a compatible form the servo drive can understand.

- In addition it updates the controller with motor and drive information used to monitor drive and motor performance.

Servo Drive

- The servo drive receives the signal provided by the motion module and translates this signal into motor drive commands.
- These commands can include motor position, velocity, and/or torque.
- The servo drive provides power to the servo motors in response to the motion commands.

Fundamentals of PLC motion control have been added.

velocity by use of an encoder mounted on the motor shaft. This feedback information is used within the servo drive to ensure accurate motor motion.

Servo Motor

- The servo motors represent the axis being controlled.
- The servo motors receive electrical power from their servo drive which determines the motor shaft velocity and position.
- The filler motor must accelerate the filler mechanism in the direction the bottles are moving, match their speed, and track the bottles.
- After the bottles have been filled, the filler motor has to stop and reverse direction to return the filler mechanism to the starting position to begin the process again.

A robot is simply a series of mechanical links driven by servo motors. The basic industrial robot widely used today is an *arm* or *manipulator* that moves to perform industrial operations. Figure 14-24 illustrates the motion of

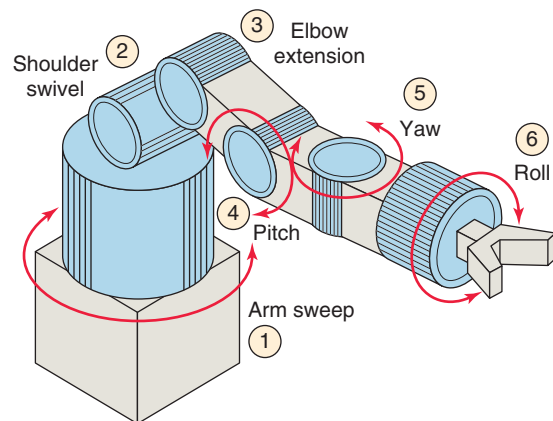
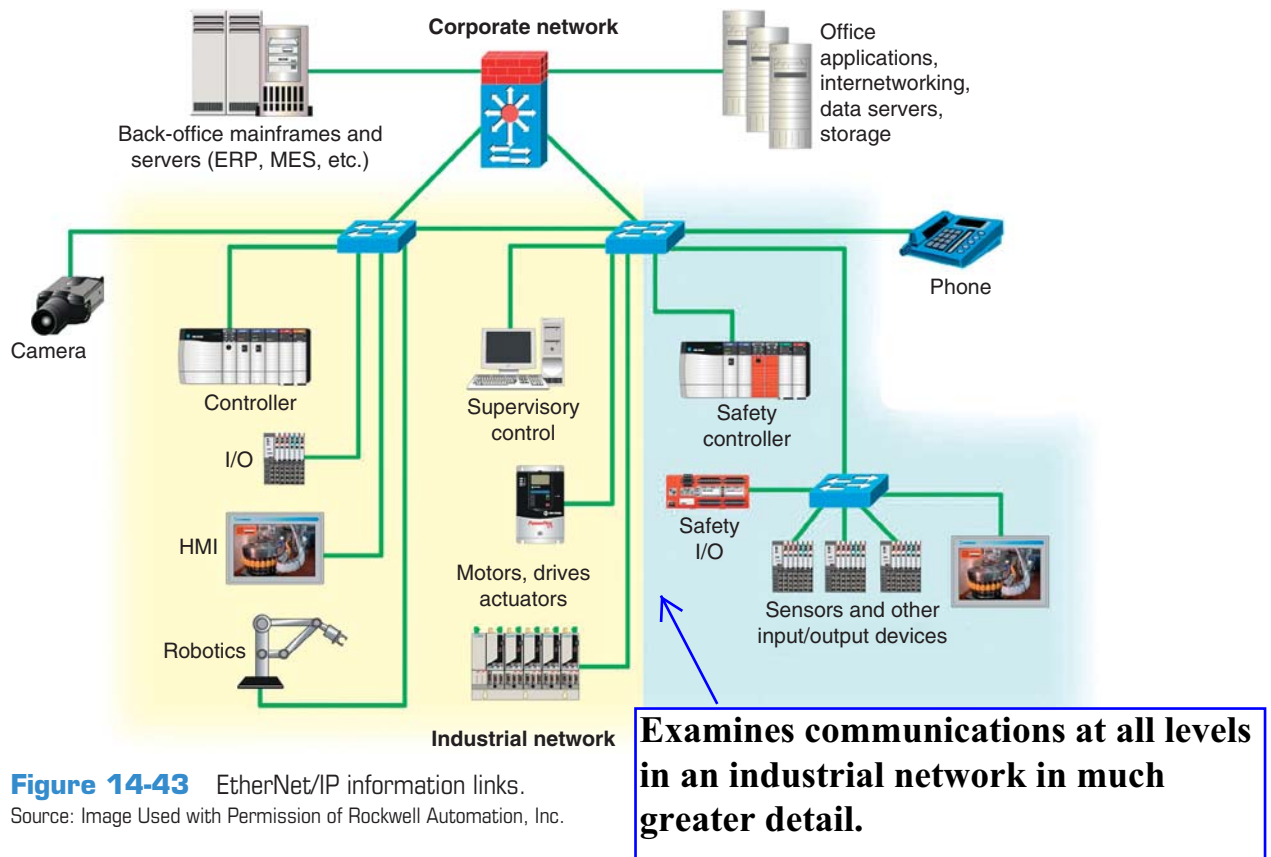


Figure 14-24 Six-axis robot arm.



build into their equipment without having to pay royalties. It has become a standard communications protocol in industry, and is one of the most commonly available means of connecting industrial electronic devices. Figure 14-44 shows an Omron PLC with Modbus-RTU network communication capabilities via RS-232C and RS-422/485 serial ports.



Figure 14-44 Omron PLC with Modbus-RTU network communication capabilities.
Source: Photo courtesy Omron Industrial Automation, www.ia.omron.com.

Fieldbus

Fieldbus is an open, serial, two-way communications system that interconnects measurement and control equipment such as sensors, actuators, and controllers. At the base level in the hierarchy of plant networks, it serves as a network for field devices used in process control applications.

There are several possible topologies for fieldbus networks. Figure 14-45 illustrates the *daisy-chain* topology. With this topology, the fieldbus cable is routed from device to device. Installations using this topology require

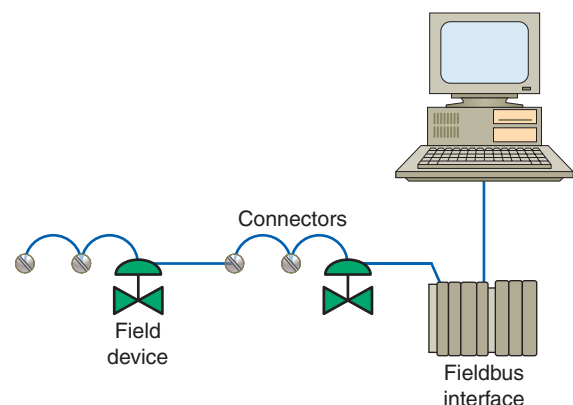


Figure 14-45 Fieldbus implemented using daisy-chain topology.

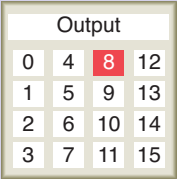




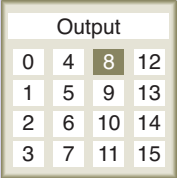




If Your Output Circuit LED Is . . .	And Your Output Device Is . . .	And	Probable Cause
ON 	On/Energized 	Your program indicates that the output circuit is off or the output circuit will not turn off. 	Programming problem: <ul style="list-style-type: none"> - Check for duplicate outputs and addresses. - If using subroutines, outputs are left in their last state when not executing subroutines. - Use the force function to force output off. If this does not force the output off, output circuit is damaged. If the output does force off, then check again for logic/programming problem.
			Output is forced on in program.
			Output circuit wiring or module.
	Off/De-energized 	Your output device will not turn on and the program indicates that it is on. 	Low or no voltage across the load. Output device is incompatible: check specifications and sink/source compatibility (if dc output). Output circuit wiring or module.
OFF 	On/Energized 	Your output device will not turn off and the program indicates that it is off. 	Output device is incompatible. Output circuit off-state leakage current may exceed output device specification. Output circuit wiring or module. Output device is shorted or damaged.
	Off/De-energized 	Your program indicates that the output circuit is on or the output circuit will not turn on. 	Programming problem: <ul style="list-style-type: none"> - Check for duplicate outputs and addresses. - If using subroutines, outputs are left in their last state when not executing subroutines. - Use the force function to force output on. If this does not force the output on, output circuit is damaged. If the output does force on, then check again for logic/programming problem. Output is forced off in program. Output circuit wiring or module.

Figure 13-32

Output troubleshooting guide.

13.10

PLC Programming

Extended coverage of practical troubleshooting techniques.

You must establish a way for your per (PC) software to communicate with the programmable logic controller (PLC) processor. Making this connection is known as *configuring* the communications. The method used to configure the communications varies with each brand of controller. In Allen-Bradley controllers, *RSLogix* software is required to develop and edit ladder programs. A second software package, *RSLink*, is needed to monitor PLC activity, download a program from your PC to your PLC, and upload a program from your PLC into your

projects to the PLC. The PLC will accept the program can consist of multiple subroutine files which can be conditionally called from the main program.

RSLink software is available in multiple packages to meet the demand for a variety of cost and functionality requirements. This software package is used as the driver between your PC and PLC processor. A *driver* is a computer program that controls a device. For example, you must have the correct printer driver installed in your PC



PART 5 REVIEW QUESTIONS

1. Construct a ControlLogix ladder rung with a math instruction that executes when a toggle switch is closed to add the tag named Pressure_A (value 680) to the constant of 50 and store the answer in the tag named Result.
2. Construct a ControlLogix ladder rung with a math instruction that executes when two normally open limit switches are closed to subtract the tag named Count_1 (value 60) from the tag named Count_2 (value 460) and store the answer in the tag named Count_Total.
3. Construct a ControlLogix ladder rung with a math instruction that executes when either one of two normally open pushbuttons is closed to multiply the tag named Cases (value 10) by the constant 24 and store the answer in the tag named Cans.
4. Construct a ControlLogix ladder rung with a compare instruction that will energize a pilot light output anytime the value stored at Data_3 is 60.
5. Construct a ControlLogix ladder rung with a compare instruction that will energize a pilot light output anytime the value stored at Data_2 is not the same as that stored at Data_6.
6. Construct a ControlLogix ladder rung with compare instructions that will energize a pilot light output anytime the pressure of a system goes above 300 psi or below 100 psi.



PART 5 PROBLEMS

1. While checking the operation of the parts tracking system with the Monitor Tags window, you note that the value of Conveyor_Sensor_1 remains at 1 with parts passing by. What can you surmise from this? Why?
2. Three conveyors are delivering the same parts in different packages. A package can hold 12, 24, or 18 parts. Proximity switches installed on each of the conveyor lines are used to advance the accumulated value of the three counters. Write a ControlLogix program that uses multiply and add instructions to calculate the sum of the parts.
3. A single pole switch is used in place of the two pushbuttons for the variable preset timer program. When this switch is closed the timer is to be set for 10 seconds and when open to 15 seconds. Make the necessary changes to the program.

Chapters conclude with a set of review questions and problems. The review questions are closely related to the chapter objectives and require students to recall and apply information covered in the chapter. The problems range from easy to difficult, thus challenging students at various levels of competence.