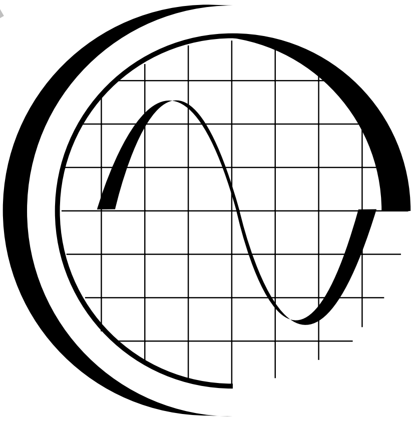
Table of Contents

PROGRAMMABLE LOGIC CONTROLLERS

Lesson One

Introduction to Programmable Logic Controllers



TPC Training Systems

29801

Lesson

1 Introduction to Programmable Logic Controllers

TOPICS

The Electromagnetic Relay
Characteristics of Programmable Controllers
Applications of Programmable Controllers
Limitations of Programmable Controllers
Parts of a Programmable Logic Controller System
The Input Side
The Processor
The Output Side
Programming Devices
Power Supplies

OBJECTIVES

After studying this lesson, you should be able to…

- Describe an electromagnetic relay and define the terms *control circuit*, *power circuit*, *NO* and *NC*.
- Define *programmable logic controller*.
- Describe the general type of application in which a programmable logic controller would best be used, and give examples.
- Define *scan time*.
- Name each of the blocks in a block diagram of a programmable logic controller system and explain how each functions within the system as a whole.
- Define *memory* and explain the different types.

KEY TECHNICAL TERMS

Programmable	1.01	capable of having its operation changed through changes in software as opposed to hardware
Relay	1.02	electromagnetically controlled switch that provides remote control of electric loads with a small control signal
Control circuit	1.04	energizes the electromagnet; controls current flowing in power circuit
Power circuit	1.04	controlled circuit
Energize	1.05	pass current through a load
Modular	1.09	constructed with standardized units for flexibility
Event-driven	1.16	responding to changes in present status
Scan time	1.42	time it takes controller to run through program and return to beginning
Processor	1.47	place where logical decisions are made
Memory	1.48	place where information is stored
Register	1.55	group of consecutive, single-bit memory locations allotted by the programmer for special functions
Bit	1.60	binary digit
Binary	1.60	having only two states

Programmable logic controllers are microprocessor-based computer devices that are designed to control machines and processes automatically. They have had a significant impact on industrial control because of their versatility, reliability, and speed.

The main purpose of this lesson is to show how programmable controllers are typically applied to equipment and processes. You will move step by step through a typical programmable logic controller system, from the input, through the processor, to the output. Power supplies, both internal and external are covered, as well as programming devices. The lesson describes the features and functions of the processor unit, which performs the required calculations and makes program decisions. In particular, it focuses on memory.

The Electromagnetic Relay

1.01 Although it might seem complicated at first, the programmable logic controller (also known simply as a programmable controller or PLC) is not too difficult to understand. In fact, programmable logic controllers are an outgrowth of an extremely simple control mechanism, the *electromagnetic relay*. Figure 1-1 shows a cutaway drawing of a typical electromagnetic relay.

1.02 A *relay* is an electromagnetically operated switch that provides remote control of electric loads with a small control signal. The term relay originated from early applications of electromagnetic switches. These switches were used to strengthen and repeat weak telegraph signals. As you can imagine, relays have been around for some time, and their use has grown considerably over the years. Relays and programmable controllers now form the cornerstone of industrial control.

1.03 An electromagnetic relay is similar in many ways to a manual electric switch. However, the relay's contacts are opened and closed by an electromagnet instead of a hand-operated lever or throw. When a relay is energized, current passes through a coil of wire surrounding an iron core. The core becomes a magnet and attracts the movable armature. As the armature moves, the contacts open or close the other circuit in the same way as they would in a manual switch.

1.04 As you can see in Fig. 1-1, an electromagnetic relay has two separate circuits. One circuit energizes the electromagnet. This circuit is generally referred to as the *control circuit*, because it is the one in control of current flowing in the other circuit. The second circuit is usually called the *power circuit* or *controlled circuit*. When the coil is energized, the armature moves one way, the contacts come together, and the circuit is completed. When the coil is de-energized, the contacts open.

1.05 Associated with each control relay are a coil and some combination of normally open (NO) and normally closed (NC) contacts. When current is passed through the coil, the relay is said to be *energized*. At this point, any NC contacts open and any NO contacts close. When the coil is *de-energized*, the contacts return to their "normal" state.

1.06 The electromagnetic relay in Fig. 1-1 has a normally open contact that closes when current passes through the electromagnetic coil. Schematic symbols for normally open and normally closed

Fig. 1-1. Electromagnetic relay

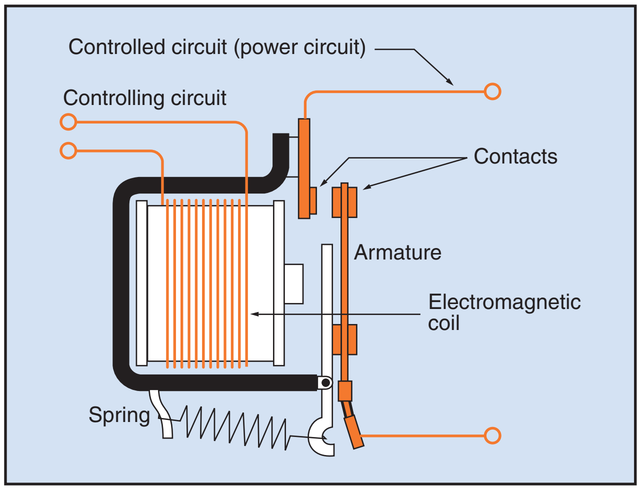
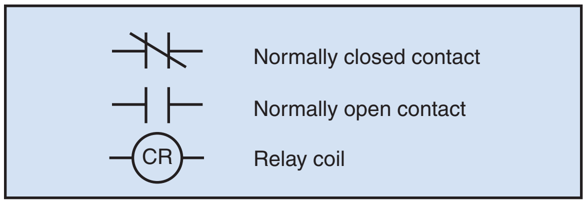


Fig. 1-2. Schematic symbols for a relay



contacts are shown in Fig. 1-2. The symbol for a relay coil is also shown.

1.07 The major advantage of a relay lies in its ability to control a circuit that is *electrically isolated* from the control circuit. It is possible, for instance, to have direct current (dc) in the control circuit, and at the same time have alternating current (ac) in the power circuit. This condition is possible because the two circuits are completely separate. It is also possible to have widely varying voltages and currents in the controlling and controlled circuits.

1.08 Although relays have long been used to control complex processes and events, they do have significant disadvantages.

- Any change in logic means that the control panel wiring must be changed, a difficult and time-consuming process.
- The contacts in an electromagnetic relay wear with each operation and will eventually fail.
- The control panels enclosing relay systems are relatively large and costly.

Characteristics of Programmable Controllers

1.09 There is a wide range of programmable logic controllers available. Some smaller, fixed I/O models contain all components in a single enclosure. These devices are sometimes referred to as "brick" or "lunchbox" systems. Larger, more capable systems are usually modular. Components can be added, substituted, and rearranged to give the system great flexibility. A modular programmable logic controller system has three major components, each of which is an essential part of the system.

1.10 The three main components of a typical system are the *processor unit*, the *input/output section*, and the *programming device*. These functional modules are shown in Fig. 1-3 and will be discussed in detail later in this lesson. The programming device can be portable and detachable, or it can be a permanent fixture. Physically, the input/output section is usually included with the processor unit in the same rack or enclosure.

1.11 One basic similarity between programmable controllers and general-purpose computers is that both require input/output devices, or I/Os. An *I/O* is a device that communicates with the logic circuitry and the process to be controlled. If a device receives data from the outside world and puts it into a form that is understandable to logic circuits, it is an *input device*. If it receives information from internal circuitry and translates it into a form that is useful to the outside world, it is an *output device*.

1.12 Communication with the I/O devices is accomplished through the input/output section of the

Fig. 1-3. Major components of a PLC system

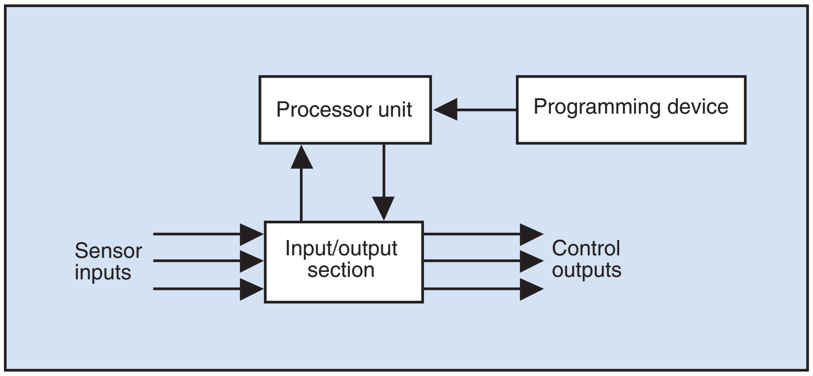
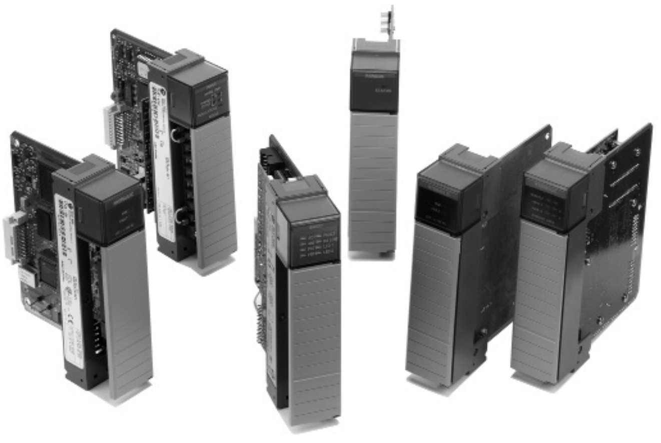


Fig. 1-4. Plug-in I/O modules



system. In a modular PLC, the system can be expanded by adding more I/O modules to this section. The rack, or *chassis*, for this section is a simple, box-like device that serves as a holder and connector to the processor for the plug-in I/O modules. Several of these plug-in modules are shown in Fig. 1-4.

1.13 The chassis is very much like a personal computer case. The processor module and I/O modules slide into slots in the chassis where they connect to a *backplane*. The backplane is simply a series of electric "wires" on a printed circuit board. The backplane connects the various I/O modules, sometimes called I/O *cards*, to the processor module.

1.14 The I/O interface contains both input modules and output modules. The total number of these modules depends on the memory of the processor, its construction, and on the complexity of the system being controlled. Each of these modules is connected by hard wiring to some real world input or output device.

1.15 Input devices, as well as output devices, are sometimes referred to as *actual* or *real world* devices. These terms arise from the fact that signals and commands do not always come from, or go to, places outside the processor circuitry. Sometimes timer, relay, and counter signals originate within the user program and the logic circuitry of the system. Real world I/O devices are always outside the processor unit.

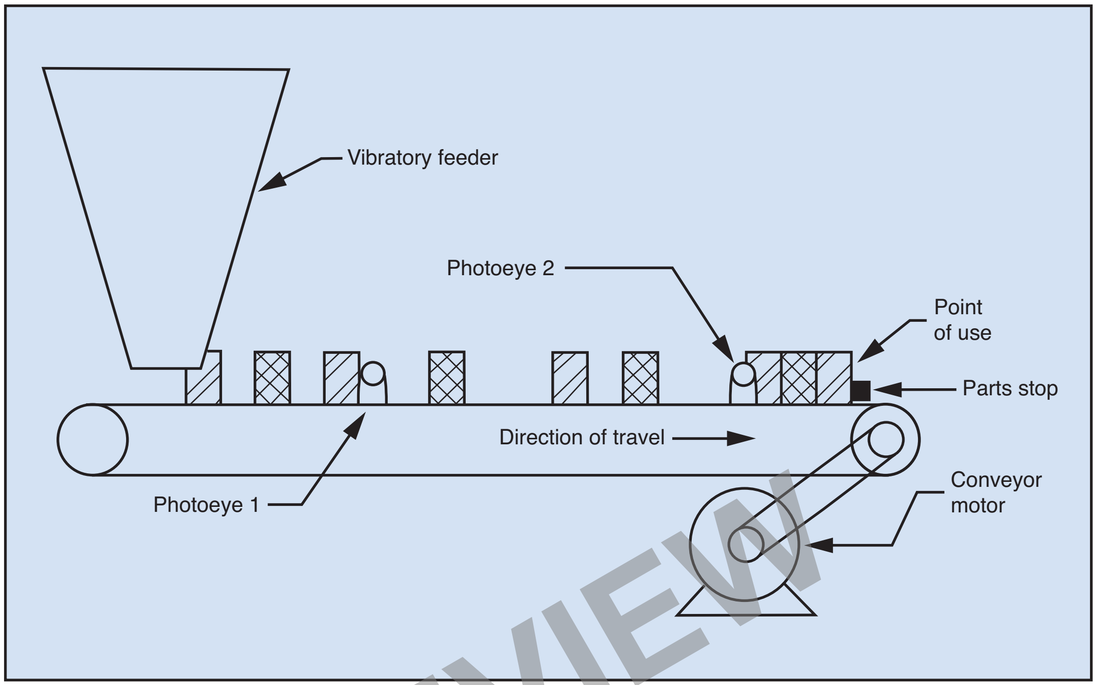
1.16 Most programmable logic controllers are *event-driven*. That is, they respond to changes in the present status of a particular process, system, or device. For example, a programmable controller can respond to a change in the temperature of a fluid in a holding tank as the change occurs. The fact that these controllers are event-driven makes them different from general-purpose computers, which are driven largely by stored information. Programmable controllers are capable of other processes as well. For example, many PLC applications utilize time-driven programming.

1.17 While a general-purpose computer usually operates with a small number of input devices, a programmable logic controller often uses many input devices, including various switches and sensors. Output devices controlled by programmable logic controllers include lights, motor starters, contactors, pumps, valves, relays, alarms, digital readouts.

1.18 The original programmable controllers did nothing more than mimic relay logic. But as solid-state and microprocessor technology became more advanced, these controllers became more powerful. In fact, most of today's programmable logic controllers are built around microprocessors. A *microprocessor* is a large-scale integrated circuit so small that it fits on a single chip. A microprocessor contains an arithmetic unit, control circuitry, and some special registers.

1.19 Just as controller hardware has evolved, so have programming languages. The original languages for programmable logic controllers served only to replace relay logic. Later languages were enhanced with such added refinements as arithmetic, branching, data manipulation, and data transfer.

Fig. 1-5. Conveyor system application



1.20 Later programming languages were made up of selected English words, similar to the BASIC computer language. These languages were much simpler to use and had much of the same flexibility as general-purpose computer languages. Today, ladder logic programming is the most popular programming format for PLCs.

Applications of Programmable Controllers

1.21 Programmable controllers are at home in a real-time environment. A *real-time environment* is one in which decisions and actions must be performed fast enough to respond to changes in the real world. In contrast to a real-time environment is the *offline environment*, where measurements are made and then manipulated without regard to what is happening in the real world at that time. The following paragraphs give examples of typical PLC applications.

1.22 **Infeed conveyor with vibratory feeder.** The conveyor system shown in Fig. 1-5 consists of a vibratory feeder that drops parts onto a conveyor. Vibratory feeders are often used to orient parts correctly before they are used in a process. In this case, parts are placed on a conveyor that transports them to their point of use.

1.23 It would be nearly impossible to feed only the exact number of parts needed, so a feeder is designed to deliver more parts than will be needed in a given time. Therefore, if the vibratory feeder were allowed to run all the time, parts could jam up as the conveyor is filled beyond capacity. Starting and stopping the feeder each time a part is needed would cycle the feeder unnecessarily. So a system is designed to cycle the feeder and keep the conveyor filled between two points, as defined by the photoeye sensors.

1.24 When the system is first started and the conveyor is empty of parts, both the conveyor belt and the vibratory feeder begin running. The conveyor belt carries the parts downstream (to the right in the drawing) to the end of the conveyor where they "stack up" against a stop. A robot or other device removes parts from the end of the conveyor as needed for the process.

1.25 The conveyor's moving belt allows parts to stack up until they continuously block photoeye 1. At that time the vibratory feeder shuts off, thereby feeding no additional parts onto the conveyor. However, the conveyor belt continues to run. As parts are removed from the end of the conveyor, the moving belt keeps the parts stacked up and available for use at the end of the conveyor. Photoeye 1 will soon become uncovered, but the vibratory feeder does not yet restart. Once enough parts are removed so that photo-eye 2 is uncovered, the vibratory feeder restarts and the cycle repeats.

1.26 **Overhead door.** Programmable logic controllers are available in a variety of sizes and configurations to fit almost any application. The cost of traditional relay controls makes new, small PLCs a logical choice for even relatively simple applications. Even the automatic opening and closing of a door becomes a viable candidate for PLC application, as shown in Fig. 1-6.

1.27 A large overhead garage door is to be controlled by a PLC. There is a pushbutton station near the door with three pushbuttons—OPEN, CLOSE, and STOP. There is also a panel nearby in plain view with three lights indicating the door's current condition. Two of the lights indicate when the door is fully open and when it is fully closed. The third light is used to indicate that the door is somewhere between fully open and fully closed, and by flashing, it also can indicate that the door is in motion. The door is operated by a reversible electric motor. The following table indicates the inputs and outputs necessary for the PLC to operate the door. Both the input and the output devices will be "hard wired" directly to the appropriate terminals on the PLC system.

Inputs	**Outputs**
OPEN pushbutton	Motor forward
CLOSE pushbutton	Motor reverse
STOP pushbutton	Door AJAR light
CLOSE limit switch	Door CLOSED light
OPEN limit switch	Door OPEN light

1.28 The door's operation should meet the following conditions.

- Momentarily pressing the OPEN pushbutton will cause the motor to start running in the forward direction, opening the door if it is not already fully open. The opening operation will continue unless the STOP pushbutton is momentarily pressed or until the OPEN limit switch is actuated when the door reaches the full open position. If either the STOP pushbutton or the OPEN limit switch is actuated, door movement will halt immediately.
- Momentarily pressing the CLOSE pushbutton will cause the motor to start running in the reverse direction, closing the door if it is not already fully closed. The closing operation will continue unless the STOP pushbutton is momentarily pressed or the CLOSE limit switch is actuated when the door reaches the full closed position. If either the STOP pushbutton or CLOSE limit switch is actuated, door movement will halt immediately.
- If the door is already fully opened, pressing the OPEN pushbutton will not energize the motor.
- If the door is already fully closed, pressing the CLOSE pushbutton will not energize the motor.

Fig. 1-6. Overhead door application

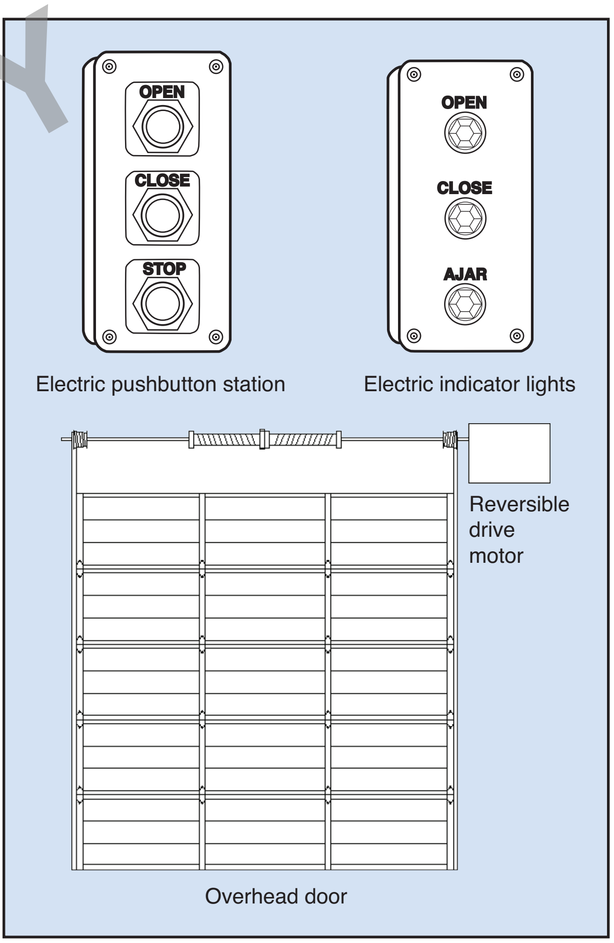
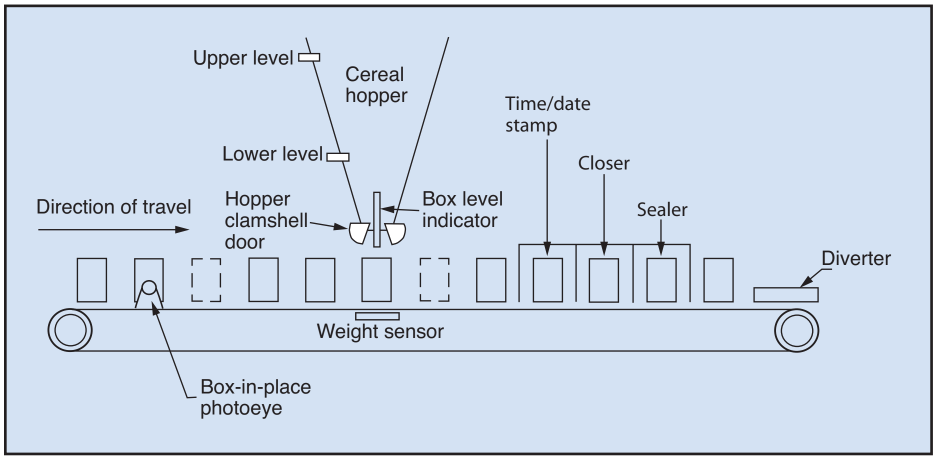


Fig. 1-7. Cereal box filling application



- Under no circumstance will the motor be allowed to energize in both the forward and reverse direction at the same time. So doing would cause serious motor damage.

- The OPEN indicator light will be lit if the door is in the fully open position.

- The CLOSED indicator light will be lit if the door is in the fully closed position.

- The AJAR indicator light will be lit if the door is anywhere between fully closed and fully open. In addition, the AJAR light will flash continuously when the door is in motion.

1.29 The programming for such an operation is relatively simple and straightforward. Additional functions can be added with only small changes in programming. For example, an added safety feature might be to program the PLC to stop the door's motion when any of the pushbuttons are hit. Other pushbutton operations could remain the same, but by allowing any pushbutton to stop a door in motion, a person would not have to be accurate in their aim in an emergency situation in which stopping the door as quickly as possible is desired.

1.30 Stopping the door by pushing any of the pushbuttons would not require any additional physical connections to the PLC. However, it would require one or two more input connections to allow for additional safety features.

- Photodetectors could be added to stop the door if something gets in its path while closing.

- Load-sensing devices, such as current sensors, could be added to stop the motor in case of a jam.

The possibilities are nearly endless.

1.31 **Cereal box filling.** Figure 1-7 illustrates an incremental conveyor system used to fill cereal boxes by weight of contents. The operator selects one of three box sizes on a thumbwheel switch. Boxes may be overweight by up to 5%, but cannot be underweight. The box may not be overfilled by volume.

1.32 An analog weight sensor measures the weight, and an ultrasonic (analog) level sensor tracks the level of contents as the box is filled. *Analog sensors* are sensors that measure a quantity, such as weight or level, in continuous amounts rather than on and off conditions as a switch would. Such quantities must be converted to a digital signal before they can be used by the PLC's program. A solid-state device called an *analog-to-digital (A/D) converter* performs this function. The filling of each box is accomplished by opening a clamshell door on a hopper above the box to be filled.

1.33 A box does not have to be present at every position. The position where boxes are shown in dashed lines indicates this condition. Empty boxes of one size are placed on the conveyor at the left end. As the conveyor moves, the box passes in front of a photo-sensor and the PLC "remembers" that a box is on the conveyor at that point. It is not necessary for a box to be on the conveyor at every position since the PLC automatically tracks whether or not a box is present as the conveyor increments along the way.

1.34 The conveyor moves one position at a time, stopping under the hopper if a box is present. If a box is present, it is filled to the designated weight as determined by the weight sensor under the box. At the same time, input from the ultrasonic box level indicator ensures that the box is not filled to overflowing. Should the box become overfilled by height of product before it reaches the desired weight, the clamshell doors shut and the PLC "remembers" that the box is "bad" and deals with it at a later point in the process. If for any reason filling a box requires more than 5 seconds, the process stops and an alarm sounds.

1.35 Once the box is filled to the correct weight, the clamshell doors shut, and the conveyor can move ahead to the next position. If for some reason the box happens to be overfilled by weight (for example, the clamshell door did not close fast enough), the PLC remembers that the box is "bad" and deals with it at a later point in the process.

1.36 An operator can select one of three box sizes by setting a thumbwheel switch prior to starting the machine. The box sizes, empty box weights, and maximum filling levels are given in the table below.

Selectable fill sizes	**10 oz**	**12 oz**	**16 oz**
Corresponding max. level	8 in.	9 in.	10 in.
Corresponding empty box wt.	1 oz	1.25 oz	1.5 oz

1.37 Conveyor positions may be empty (no box present). Bit shift registers are used to track boxes on the conveyor as well as whether the box is good or bad after the fill/weigh/measure position. A bit shift register is actually an instruction a PLC programmer can use to track items in a system.

1.38 The thumbwheel responds only to exact inputs (10, 12, 16). That is, the machine will not respond (start or run) to sizes different than those values. Once the system is running, the box size cannot be changed unless the system is stopped.

Other operations:

- Time/date stamp if box is present.
- Close and seal if good box is present.
- Close only (no seal) if bad box is present.
- RF seal for 1.5 seconds if box is good.

1.39 After a box has been time/date stamped, closed, and sealed (if it is a good box), it is moved off the end of the conveyor in one of two directions by a diverter gate. Good boxes are sent to packaging, and bad boxes are diverted to a holding area for inspection. If too many bad boxes occur in a given time, the process is halted and an alarm is sounded to call attention according to the following conditions.

Halt operation and sound alarm if:

- Two consecutive bad boxes.
- Three bad boxes in 1 hour.
- Five bad boxes in one 8 hour shift.
- Over 5 seconds on fill operation.

Additional rules of operation:

- Conveyor cannot increment if clamshell door is not closed.
- Conveyor cannot increment if diverter is not in one of the two possible positions.

1.40 The hopper contains two sensors to indicate the level of product available for filling boxes. The output of these sensors is fed to another controller that distributes product to several similar lines. It is not necessary to use these for this part of the process since, if the hopper runs out of product, the fill time limit would stop the process, as described earlier.

Inputs:

- Start (pushbutton)
- Stop (pushbutton)

- Divert to packaging (limit switch)
- Divert to inspection (limit switch)
- Box on conveyor (photoeye)
- Clamshell door shut (limit switch)
- Hopper full (limit switch)
- Hopper low (limit switch)
- Box level sensor (analog input)
- Weight sensor (analog input)
- Thumbwheel switch for package size.

Outputs:

- System run
- Hopper clamshell door open solenoid
- Conveyor increment
- Conveyor in place
- Time/date stamp
- Box closer
- Box sealer
- Divert to packaging/inspection
- Total counter (good boxes)
- System fault
- Seven segment LEDs for package size, in oz
- Alarm.

Limitations of Programmable Controllers

1.41 The previous examples give some indication of the versatility of programmable logic controllers. However, do not think that they have totally taken the place of relays. On the contrary, relays are still needed in industrial control. Small machines frequently use them and, in many cases, programmable logic controllers are used to open and close relays on larger pieces of equipment.

1.42 *Scan time* is the time it takes the programmable logic controller to run through its entire program and return to the beginning. Although PLCs are capable of handling large amounts of complex mathematics and data manipulation, these operations could considerably extend the PLCs scan time. For this reason, complex math is sometimes done by math coprocessors in an offline environment.

1.43 Most standard programmable logic controllers do not have permanent operator consoles. They are designed to be connected to portable keyboards for programming and then immediately disconnected. Once they are programmed, these controllers are allowed to run alone, without operator intervention. Control applications that are highly user interactive will have I/O modules designed for operator interface and durable enough for their environment.

The Programmed Exercises on the next page will tell you how well you understand the material you have just read. Before starting the exercises, remove the Reveal Key from the back of your book. Read the instructions printed on the Reveal Key. Follow these instructions as you work through the Programmed Exercises.

1-1. An electric relay has two separate circuits: the ________ circuit and the ________ circuit.	1-1. CONTROL; POWER or CONTROLLED Ref: 1.04
1-2. One disadvantage of relays is that any change in logic requires that control panel ________ must be changed.	1-2. WIRING Ref: 1.08
1-3. Larger PLC systems are usually ________, meaning that components can be added and rearranged for increased flexibility.	1-3. MODULAR Ref: 1.09
1-4. Name the three main components of a typical PLC system.	1-4. PROCESSOR UNIT, INPUT/OUTPUT SECTION, PROGRAMMING DEVICE Ref: 1.10
1-5. Most PLCs are ________ -driven; they respond to changes in the status of a process or device.	1-5. EVENT Ref: 1.16
1-6. Most of today's PLCs are built around ________.	1-6. MICROPROCESSORS Ref: 1.18
1-7. Today, the most popular programming format for PLCs is ________.	1-7. LADDER LOGIC Ref: 1.20
1-8. The time it takes a PLC to run through its entire program and return to the beginning is called ________ time.	1-8. SCAN Ref: 1.42

Parts of a Programmable Logic Controller System

1.44 Figure 1-8 shows a block diagram of a programmable logic controller system. This lesson will consider each of these blocks, one at a time. As you work through this information, it might be helpful to think of the system as a big logic machine—a system that takes signals from the real world, feeds them through inputs, and makes logical decisions based on the programming that has been entered into its memory. Once the decisions have been made, the programmable logic controller system operates the appropriate output devices.

The Input Side

1.45 An *input device* is a real world device that can be used to give a signal to the input module. A programmable controller can have hundreds or even thousands of input devices. Consider a simple push-button, for example. This device might be used to tell the processor to start a motor. A device such as this is typically wired to an input module.

1.46 The *input module* processes signals from input devices and gives them to the processor in a form that it can understand. A programmable logic controller system can have several input modules. Each module has a specified number of *input points*, meaning that the module can handle that many input devices. The input module is typically plugged into a rack. Generally the processor is plugged into the same rack. Once the input signal is in the processor, logical decisions are made based on the user program.

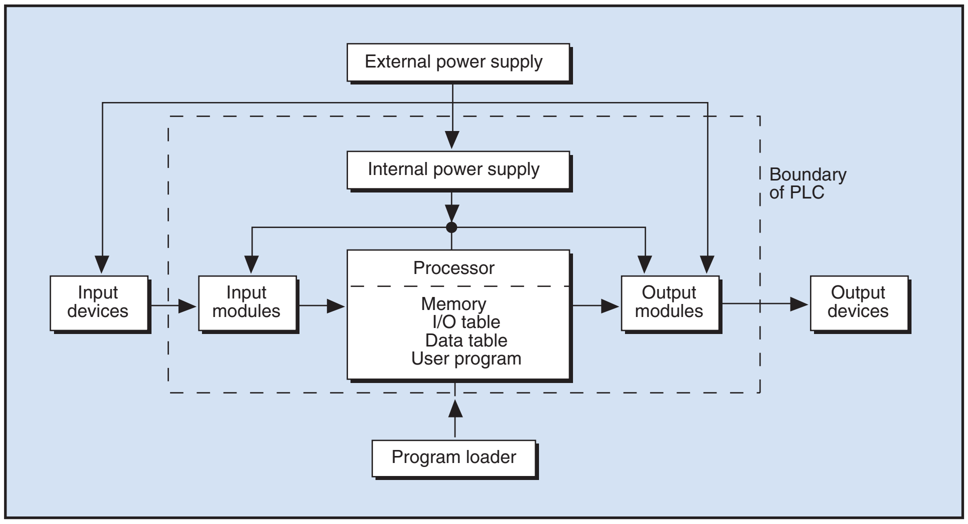
The Processor

1.47 The *processor* is the most complex part of the system. It is here that the logical decisions are made. These decisions are made based on the *program* that has been loaded into the memory. Figure 1-9 shows a typical processor installed in the chassis along with the power supply and I/O modules.

1.48 **Memory.** Just as the name implies, *memory* is a place where information is stored. The programmable logic controller, like all computers, requires memory to operate. There are many different types of memory chips or ICs. Manufacturers design their PLCs with specific memory types suited for the PLC's intended application.

1.49 First, the PLC needs memory to hold its operating system. PLC operating systems are called *firmware*. Firmware contains the instructions that control all other aspects of the computer, such as loading, interpreting, and running the user program. Firmware is written by the PLC manufacturer and is usually stored permanently in the processor section or processor module of the PLC. Memory used for firmware is usually a type referred to as *read-only memory* or *ROM*. ROM can maintain its memory without external power. Memory of this type is considered *nonvolatile*.

Fig. 1-8. Block diagram of a program logic controller system



1.50 Early PLCs used a type of ROM called *PROM* (*programmable read-only memory*) or *EEPROM* (*electrically erasable programmable read-only memory*) to contain the firmware. If firmware upgrades became available, a PROM would have to be replaced with a new one containing the firmware upgrade. EEPROM memory used for firmware would have to be removed, erased, reprogrammed with the new firmware, and reinstalled in the PLC whenever firmware was updated.

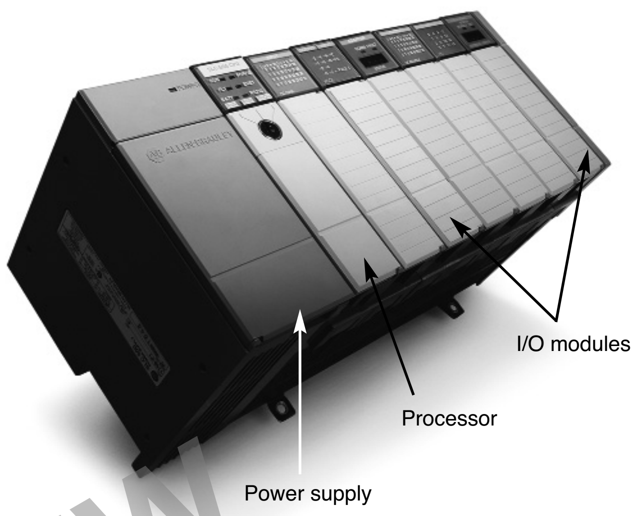
1.51 Many PLCs now use a newer type of nonvolatile memory called *flash memory*. Flash memory is a special type of EEPROM that can be reprogrammed without removing it from the PLC. With these PLCs, the firmware can be updated without disassembling the processor. Usually all that is required is a connection from the PLC to a personal computer and either a CD-ROM containing the upgrade or an internet connection to the PLC manufacturer's website. PLC manufacturers provide specific instructions on how to update firmware in flash memory.

1.52 Secondly, PLCs need memory to store the user program and any data the user program might generate during execution. PLC memory for this purpose is usually *random access memory* (*RAM*). RAM requires constant power to retain its contents. If power is lost, RAM loses whatever had been stored in it. For this reason, RAM is referred to as *volatile memory*. PLCs often use a small battery for backup power so the user program is not lost when power to the PLC is lost or turned off.

1.53 Many PLCs have a socket or slot in the processor module for user program backup memory. This backup memory is a read-only memory device that contains a copy of the user program. If the user program in RAM becomes corrupted or the battery backup does not work, this copy can be automatically loaded into the processor. In addition, the socket or slot can be used to load a new program into the processor if needed. EEPROM and flash memory are commonly used for this purpose. Some newer PLCs use the same CompactFlash® memory used in digital cameras for their backup memory.

1.54 Memory can also be divided by its intended use into program files and data files. *Program files*, as the name implies, are the files that contain the user

Fig. 1-9. Basic components of a PLC system



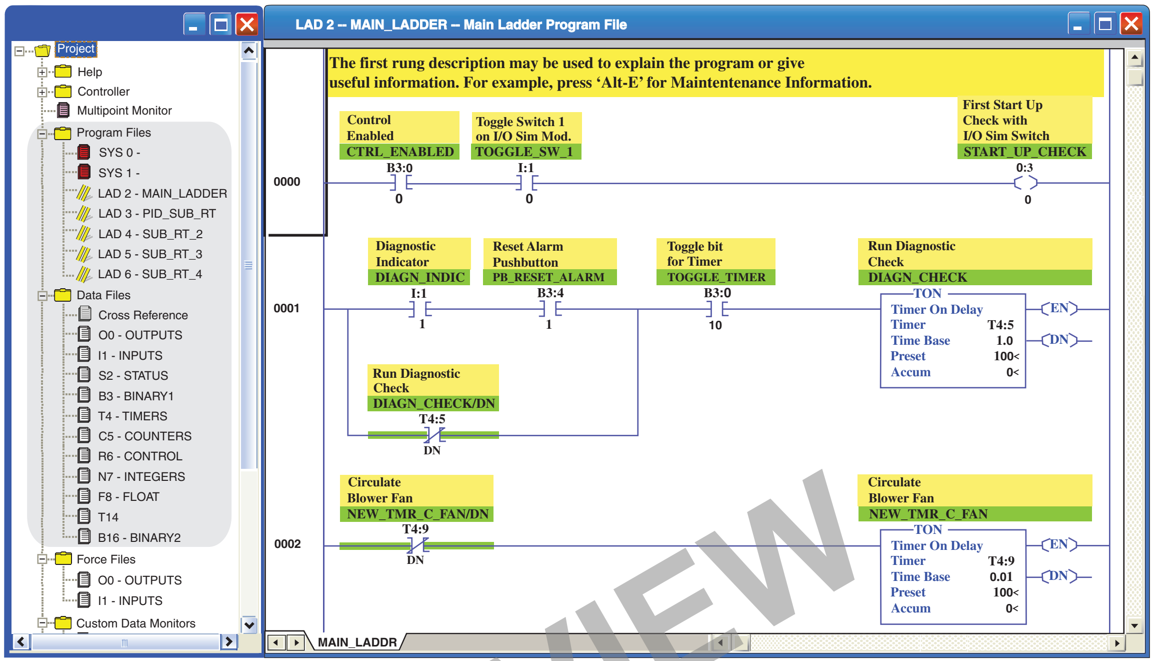
program. There may be many of these files, broken down by subroutines or functions. Together, they contain the instructions the programmer entered that cause the PLC to control the machines to which it is connected. *Data files* are the files that contain the data needed by the program files to operate correctly. Some of the data files may have been constructed by the programmer for use by the program. Other data files may be keeping track of data generated by the program itself. Some data files contain both user-entered data and program-generated data.

1.55 One special type of file used in PLC programs is the register. A *register* is simply a group of consecutive, single-bit memory locations allotted by the programmer for special functions. One common application using registers is the bit shift instruction. *Bit shift instructions* are used to track items through a manufacturing operation. The instruction places a 1 or 0 in the register and moves that bit one memory location at a time through the register as the product moves through the manufacturing operations. By so doing, the register can keep track of the individual product without using numerous expensive sensing devices at every location.

1.56 Figure 1-10, on the following page, shows the program files and data files for a typical program. This arrangement of files is used in a programming/monitoring environment for a popular

CompactFlash® is a registered trademark of SanDisk

Fig. 1-10. Program files and data files


Project
Help
Controller
Multipoint Monitor
Program Files
SYS 0 -
SYS 1 -
LAD 2 - MAIN_LADDER
LAD 3 - PID_SUB_RT
LAD 4 - SUB_RT_2
LAD 5 - SUB_RT_3
LAD 6 - SUB_RT_4
Data Files
Cross Reference
O0 - OUTPUTS
I1 - INPUTS
S2 - STATUS
B3 - BINARY1
T4 - TIMERS
C5 - COUNTERS
R6 - CONTROL
N7 - INTEGERS
F8 - FLOAT
T14
B16 - BINARY2
Force Files
O0 - OUTPUTS
I1 - INPUTS
Custom Data Monitors
LAD 2 – MAIN_LADDER – Main Ladder Program File
The first rung description may be used to explain the program or give useful information. For example, press 'Alt-E' for Maintentenance Information.
0000
Control Enabled CTRL_ENABLED B3:0 0
Toggle Switch 1 on I/O Sim Mod. TOGGLE_SW_1 I:1 0
First Start Up Check with I/O Sim Switch START_UP_CHECK 0:3 0
0001
Diagnostic Indicator DIAGN_INDIC I:1 1
Reset Alarm Pushbutton PB_RESET_ALARM B3:4 1
Toggle bit for Timer TOGGLE_TIMER B3:0 10
Run Diagnostic Check DIAGN_CHECK
TON Timer On Delay Timer T4:5 Time Base 1.0 Preset 100< Accum 0<
EN
DN
Run Diagnostic Check DIAGN_CHECK/DN T4:5 DN
0002
Circulate Blower Fan NEW_TMR_C_FAN/DN T4:9 DN
Circulate Blower Fan NEW_TMR_C_FAN
TON Timer On Delay Timer T4:9 Time Base 0.01 Preset 100< Accum 0<
EN
DN
MAIN_LADDR

PLC series. Most PLC programming is done using personal computers and software similar to that shown. The active window on the left shows the files that make up the total program. You can see one folder with the name PROGRAM FILES. In this folder there are several files, the first two of which are called SYS 0 and SYS 1, short for System File 0 and System File 1. These two files are system files and are not accessible to the user.

1.57 The rest of the files in the PROGRAM FILES folder are actual user program files. Each of these files contains instructions put together by the programmer. In this case, the programming format is called *ladder logic* and has an appearance similar to the relay logic diagrams used in motor control. For this reason, each of these files is called a ladder file and is default labeled as LAD 2 for ladder file 2, LAD 3 for ladder file 3, and so on. It is only necessary to have one ladder file, LAD 2, for a fully functional program. In this case, additional ladder files were created to be used as subroutines, a more advanced programming technique.

1.58 On the right you can see LAD 2, ladder file 2, which has been titled MAIN_LADDER. This is the actual program file, open for editing or viewing in real time as it is running in the PLC.

1.59 Directly under the PROGRAM FILES folder is the DATA FILES folder. This folder contains all the data files used by the program to keep track of data generated by the program or for use by the program.

1.60 At this point, it is necessary to understand a little more about computer terminology. You have probably heard the words bits and bytes in reference to computers, but you might not know just what they mean. A *bit* (short for *bi*nary digi*t*) is the smallest unit of memory in any computing system. Since a PLC is just a specialized industrial computer, it uses bits just like every other computer. A bit can have only two values, zero (0) or one (1). In fact the word *binary* means that there are only two states, zero and one. While a bit can only be a zero or a one, in the PLC it is very powerful. It takes only one bit to turn something like a huge electric motor on or off when that bit is used in a PLC.

1.61 It would be nearly impossible to keep track of all the bits necessary to run a computer without some additional organization, so bits are organized into groups called *bytes*. Eight bits makes a byte. Further, bytes are organized into *words*. Words can have different numbers of bytes, depending on the computer or system. In the case of our programming example, a word consists of two bytes, or 16 bits total. Now we can keep track of the individual bits in groups of 16 bits at a time.

1.62 Looking back at the DATA FILES folder, notice that the first file is called the CROSS REFERENCE file. It contains information on where everything in the rest of the data files appears in the user program. This is very handy information when you are troubleshooting a program.

1.63 The files listed after the CROSS REFERENCE are:

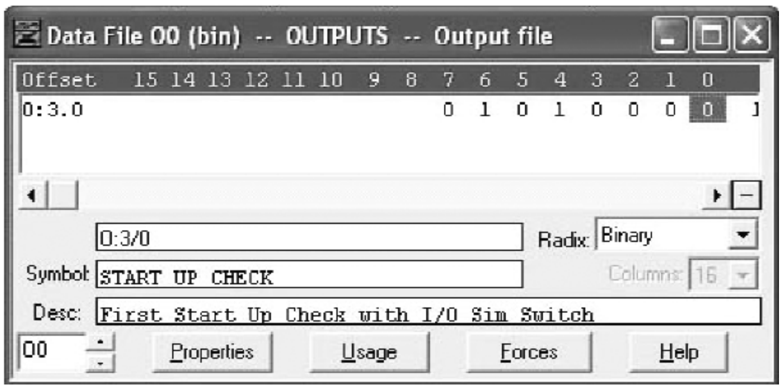
- O0–OUTPUTS
- I1–INPUTS
- S2–STATUS
- B3–BINARY (sometimes called the *bit file*)
- T4–TIMERS
- C5–COUNTERS
- R6–CONTROL
- N7–INTEGERS
- F8–FLOAT (floating-point file).

These nine files are the default files for any program using this PLC. There are two additional files listed in the data files. These were added by the programmer, but have the same structure as their counterparts in the original nine data files.

1.64 **OUTPUTS file.** First we will open the O0–OUTPUTS file, as shown in Fig. 1-11. In the section labeled OFFSET, notice the numbers 0 through 15 across the top from right to left (a total of 16). We usually start numbering anything in computers with 0 (zero), so position 0 is actually the first of anything in computers. These are the 16 bits of a word used for output in this program. Under the word OFFSET you can see the label O:3.0. This means output file word zero for slot three of this PLC. We now know that there is an output module in slot three of the PLC, and we can see there are 8 bits controlling the outputs in that card. The individual bits are bits labeled 0 through 7. Notice that bits 4 and 6 are ones while the rest of the bits are zeros. The reason there are no bits (ones or zeros) in positions 9 through 15 is because the output card in slot 3 has only eight outputs that can be turned on or off with the bits. If this program were running, outputs 4 and 6 in slot 3 of the PLC would be turned on and the other 6 outputs would be off.

1.65 This is a good time to cover the topic of addressing in PLCs. All computers have their memory organized by addressing schemes of one form or another. There is a unique address for every bit in any computer. Whether you are working with inputs and outputs or some other data file, addressing is how you send or get information to or from the right memory bits. You should be aware that different PLC manufacturers handle addressing in slightly different manners. For example, some use the letter X to indicate inputs and the letter Y to indicate outputs. In this course, we will use the letter I to indicate inputs and the letter O to indicate outputs.

Fig. 1-11. OUTPUTS file



Data File O0 (bin) -- OUTPUTS -- Output file

```
Offset   15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
O:3.0                             0  1  0  1  0  0  0  0
```

O:3/0 Radix: Binary

Symbol: START UP CHECK Columns: 16

Desc: First Start Up Check with I/O Sim Switch

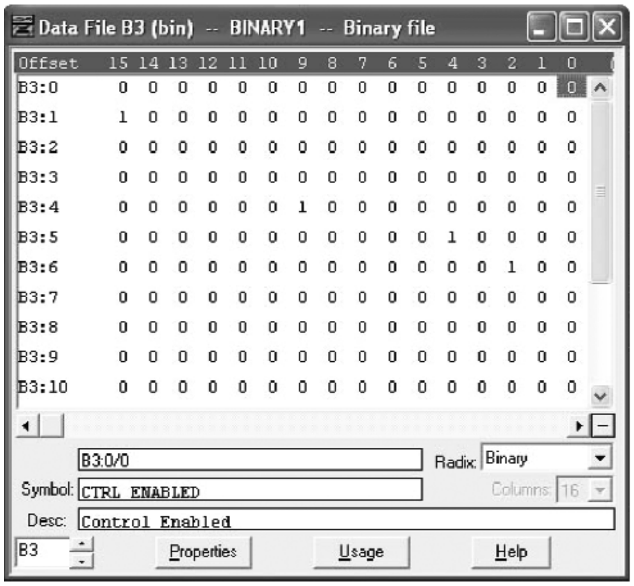


Fig. 1-12. BINARY file

```
Data File B3 (bin) -- BINARY1 -- Binary file
Offset  15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
B3:0     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
B3:1     1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
B3:2     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
B3:3     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
B3:4     0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
B3:5     0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
B3:6     0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
B3:7     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
B3:8     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
B3:9     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
B3:10    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

         B3:0/0                 Radix: Binary
Symbol:  CTRL ENABLED           Columns: 16
Desc:    Control Enabled
```

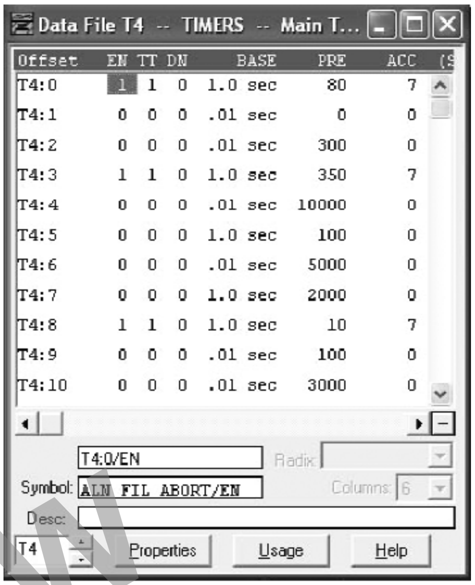


Fig. 1-13. TIMERS file

```
Data File T4 -- TIMERS -- Main T...
Offset   EN TT DN     BASE     PRE    ACC
T4:0      1  1  0  1.0 sec      80      7
T4:1      0  0  0  .01 sec       0      0
T4:2      0  0  0  .01 sec     300      0
T4:3      1  1  0  1.0 sec     350      7
T4:4      0  0  0  .01 sec   10000      0
T4:5      0  0  0  1.0 sec     100      0
T4:6      0  0  0  .01 sec    5000      0
T4:7      0  0  0  1.0 sec    2000      0
T4:8      1  1  0  1.0 sec      10      7
T4:9      0  0  0  .01 sec     100      0
T4:10     0  0  0  .01 sec    3000      0

         T4:0/EN                Radix:
Symbol:  ALN FIL ABORT/EN       Columns: 6
Desc:
```

1.66 The format for addressing this PLC has two basic forms—one for the input and output files and one for the rest of the data files. There can only be one output file and one input file, and all the cards in the PLC I/O rack are addressed through those files.

1.67 As mentioned earlier, output files in this course will always start with the letter O. It will be followed by a colon, then the slot number in which the output module is installed, followed by a period, then the word designation if it has more than 16 bits, followed by a fore slash (/), then an individual bit number. So O:3.0/0 is the first bit in the first word of the output card in slot 3. Since there is only one word in this example, we can eliminate the .0 (dot zero) and simply use the designation O:3/0. Either way is fine.

1.68 **INPUTS file.** The INPUTS file is similar to the OUTPUTS file, except that the designation would be I:2.0/1 for bit one in the first word of the input card in slot 2.

1.69 The field wiring terminals on the I/O modules most often use the same code as the data file address. For example, an input terminal might be labeled I:2.0/1 to indicate its memory location for bit one in the first word of the input card in slot 2. This code assigns a unique input or output address for every terminal and, consequently, for every I/O point. Similarly, an output terminal might be labeled O:3/0 to indicate its specific memory location as the first bit in the first word of the output card in slot 3.

1.70 **BINARY file.** The B3–BINARY file is shown in Fig. 1-12. The BINARY files are often used to simulate relays in PLC programs. Notice there are several words here, each containing 16 bits. These words are called *elements*. Each bit can simulate a relay with a nearly unlimited number of contacts. The addressing of the bits in the BINARY file is made up of the file designation followed by a colon, then the element (word) number followed by a fore slash (/), then the bit number. The highlighted bit, for example, is B3:0/0. In the row (element) labeled B3:4 the bit that is a one would be addressed as B3:4/9. The bits here have the same power to control things as the bits in the OUTPUTS or INPUTS files. This is true of all bits in the data files, though only the INPUTS and OUTPUTS files have a direct relationship to the physical I/O cards in the PLC.

1.71 **TIMERS and COUNTERS files.** The next files, T4–TIMERS and C5–COUNTERS, are the timers and counters files. Once, all counting and timing operations were done with physical electric timers and counters. With PLCs, the computer is capable of very accurate timing and counting functions, eliminating the need for expensive physical units. The PLC can simulate a large number of such devices without any additional expenditure. Refer to the T4–TIMERS file shown in Fig. 1-13.

1.72 The file format looks a little different for the timers and counters. Both timers and counters have a three-word format for each element. That is, a timer, such as the one labeled as T4:0 has three separate words in it. The first word is the *status word*, and contains bits that tell whether the timer is enabled, when it is running, and when it has timed out, among other things. These bits are commonly called by their initials—EN (enable), TT (timer timing, true whenever timer is running), and DN (done, true whenever timer has timed out). They are used rather than actual word and bit numbers to control timing operations in programs.

1.73 The other two words in the timer data element are the PRESET and the ACCUMULATOR. These are the words that are entered into the TIMER instructions when the programmer is writing the program. They are shown in the data file as PRE and ACC and are used as an entire word that holds a numerical value. For example, T4:0 has a PRESET of 80 and the ACCUMULATOR shows 7. If the program were running, that would mean the timer is set to run for 80 seconds and has currently run for 7 seconds. When a programmer wishes to address these words, he or she would use the address designation T4:0.PRE or T4:0.ACC. The individual bits in the status word, such as the DONE bit, would be addressed as T4:0/DN.

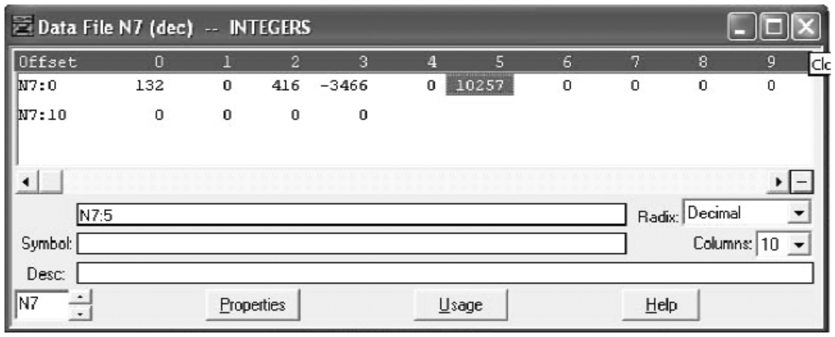
1.74 COUNTERS files, like TIMERS files, are formatted with three words. The main difference is that while the status word contains a DONE bit similar to the TIMERS files, other bits include underflow, overflow, up enable, down enable, and others. The PRESET and ACCUMULATOR words for the COUNTERS files are the same as those in the TIMERS files. The first counter element would be addressed as C5:0. Its DONE bit would be C5:0/DN. Its PRESET would be C5:0.PRE and its ACCUMULATOR would be C5:0.ACC.

1.75 **CONTROL register.** The register R6–CONTROL is used for more advanced instructions and will not be covered in detail here. Simply keep in mind that many advanced instructions require multiple words to keep their data. Rather than create separate files for all these different instructions, as was done with the TIMERS and COUNTERS, a generic file group was created so that elements from that group could be assigned to the advanced instructions as they were needed.

1.76 **INTEGERS file.** The N7–INTEGERS file is designed specifically to hold integer (whole) numbers as needed in a program. Each integer word contains 16 bits, so the range of values that can be represented by each word is limited to –32768 to +32767. An INTEGERS file is shown in Fig. 1-14.

1.77 Element N7:0 contains the value 132 while element N7:3 contains –3466. N7:0 and N7:3 are the addresses of the elements. The contents of those elements are 132 and –3466 respectively. These values may have been entered when the program was written to be used by the program. Or they may be numbers stored by the program as the result of some calculation done by the program. In either case, the example shows how numbers are stored and how they are addressed.

Fig. 1-14. INTEGERS file



Data File N7 (dec) -- INTEGERS

```
Offset      0     1     2      3     4     5     6     7     8     9
N7:0      132     0   416  -3466     0 10257     0     0     0     0
N7:10       0     0     0      0
```

N7:5 Radix: Decimal
Symbol: Columns: 10
Desc:
N7

1.78 **FLOAT file.** The F8–FLOAT (floating-point) file is another file for storing numbers, but may contain fractional values. Rather than storing a value in strict binary format, the floating-point file stores the numbers in a special coded form. It is capable of storing very large and very small numbers, both positive and negative. Its main function is to be used with mathematical instructions in programs that need extreme accuracy. It is unlikely you would need any greater accuracy than that offered by the floating-point file in a PLC operation. The elements are addressed much like the INTEGERS file elements, except that the designation F8 is used instead of N7. A typical floating-point file element address might be F8:0 or F8:17. The contents of a floating-point element might be 1.2436556e+07 or –2.775469e–56.

The Output Side

1.79 A programmable controller can have a large number of *output modules*, depending on the size of the application. The output devices wired to them are energized when the module receives a signal from the processor. The power that drives the output device does not come directly from the output module. The output module is a switch that diverts power from an external power supply to the output device.

1.80 An *output device* is a device that needs a signal to be activated. A programmable logic controller system can have a large number of output devices. A motor starter is a good example of an output device. This device is typically wired to an output module as well as to a motor.

Programming Devices

1.81 The *program loader* is a device that is used to load the program into the processor. It can be a personal computer or a dedicated piece of equipment made by the controller manufacturer. In either case, this program is the most important part of the system. It is the program that makes the programmable controller flexible. Generally, the program loader is not needed during normal operation once the system has been started up and debugged.

1.82 The *programming device* allows the user to create new control programs or edit and verify existing programs. It can also be very helpful to maintenance personnel when troubleshooting a programmable controller system. Types of programming devices include desktop units, hand-held programmers, laptop computers, and other personal computers.

1.83 Most desktop units have a full typewriter keyboard, as well as a built-in monitor. Most keyboards have raised mechanical-action keys, although many have sealed touchpad keys. These sealed touchpads are especially popular in manufacturing environments where dirt can be a problem. Desktop units might also have a row of function keys, called *soft keys*, that enable you to select among several display and programming modes. For example, one of the keys calls up a ladder grid on the screen on which you can develop your program graphically.

1.84 Although simple hand-held programmers have few keys, you will always find the standard symbols of relay logic included on their keyboards. Hand-held programmers are similar in appearance to hand-held calculators. Although typically their editing and display capabilities are limited, their main advantage is their portability. They can be taken into industrial environments, and programs can be changed on-site. Some hand-held programmers have a small LCD screen for displaying ladder logic programs.

1.85 In recent years, personal computers, especially laptop models, have become the most common tools for programming and troubleshooting programmable logic controllers. These devices have largely replaced the desktop and handheld models just described. They can be adapted with an array of program development tools, allowing them to function as program development terminals as well as general-purpose computers. Software for PLCs often has troubleshooting and debugging functions built in to simplify maintenance. Often the personal computer or laptop is connected to the PLC by network wiring, making it possible to observe, debug, and troubleshoot from a distance.

Power Supplies

1.86 The programmable controller has an *internal* power supply that is sometimes part of the controller itself, but might be a plug-in module. The modular power supply allows you the flexibility to support different sized systems. The larger the system, the larger the power supply required. This supply is used to power the components of the PLC control system, not the output devices.

1.87 The *external* power supply is not part of the controller itself. It is a term that is used to describe that portion of the system that powers the entire application. This external supply is also used to energize the output devices.

1-9. The most complex part of the PLC system is the ________; it is here that decisions are made, based on the ________ that has been loaded into memory.	1-9. PROCESSOR; PROGRAM Ref: 1.47
1-10. Memory can be divided by its intended use into ________ files and ________ files.	1-10. PROGRAM; DATA Ref: 1.54
1-11. A bit can have only two values. What are they?	1-11. ZERO (0) or ONE(1) Ref: 1.60
1-12. Bits are organized into groups of eight called ________, which are further organized into ________.	1-12. BYTES; WORDS Ref: 1.61
1-13. All computers have their memory organized by a(n) ________ scheme. This is how you send or get information to or from the proper memory bits.	1-13. ADDRESSING Ref: 1.65
1-14. A(n) ________ word contains bits that tell whether a timer is enabled, when it is running, and when it has timed out.	1-14. STATUS Ref: 1.72
1-15. The most common tools for programming and troubleshooting PLCs are ________.	1-15. PERSONAL COMPUTERS Ref: 1.85
1-16. The ________ power supply powers the entire application and is used to energize the output devices.	1-16. EXTERNAL Ref: 1.87

Answer the following questions by marking an "X" in the box next to the best answer.

1-1. In an electromagnetic relay, the control circuit is the one that

- ☐ a. carries the controlled load
- ☐ b. energizes the electromagnet
- ☐ c. opens when the armature moves
- ☐ d. responds to the electromagnet

1-2. Electromagnetic relays can operate while connected to two different voltages because

- ☐ a. most of them have built-in voltage selectors
- ☐ b. they each have two electromagnetic coils
- ☐ c. they have no contacts
- ☐ d. they have separate control and load circuits

1-3. An I/O device is basically any system component that

- ☐ a. communicates with the programmable controller's internal logic circuits
- ☐ b. converts mechanical motion or position into an electric signal
- ☐ c. establishes final control over the machine or process
- ☐ d. fastens to the controller frame, with or without cable leads

1-4. Because a programmable controller can respond to a change in the temperature of a fluid in a tank as the change occurs, it is said to be

- ☐ a. electrically isolated
- ☐ b. event-driven
- ☐ c. modular
- ☐ d. time-driven

1-5. Programmable controllers are used to best advantage in applications where

- ☐ a. a small machine must be started and stopped
- ☐ b. controls must respond rapidly to changes in the real world
- ☐ c. much data reduction is required
- ☐ d. rapid, real-time control is not possible

1-6. The purpose of an input module is to

- ☐ a. link the inputs and outputs
- ☐ b. make logical decisions
- ☐ c. process signals from an input device
- ☐ d. store information

1-7. The three-word format for each timer element consists of a status word, a(n) ________ word, and a(n) ________ word.

- ☐ a. accumulator; integer
- ☐ b. integer; float
- ☐ c. float; preset
- ☐ d. preset; accumulator

1-8. Which data file stores numbers in coded form to be used when great accuracy is required?

- ☐ a. Binary
- ☐ b. Control
- ☐ c. Floating point
- ☐ d. Integer

1-9. Which of the following can be used as an output device?

- ☐ a. Encoder
- ☐ b. Limit switch
- ☐ c. Motor starter
- ☐ d. Pushbutton

1-10. Which element makes a programmable logic controller system flexible?

- ☐ a. I/O
- ☐ b. Memory
- ☐ c. Power supply
- ☐ d. Program

SUMMARY

Programmable logic controllers often replace control systems that traditionally used electromagnetic relays. A relay is an electromagnetically operated switch that provides remote control of electric loads with a small control signal. Programmable controllers are at home in a real-time environment—one in which decisions and actions must be performed fast enough to respond to changes in the real world. Although these devices are extremely versatile, they have not replaced relays in all applications. In fact, they often operate relays on larger pieces of equipment, especially where high voltage, high current, or both are present.

A PLC system takes signals from the real world, feeds them through inputs, makes decisions based on its program, and then operates the appropriate output devices. An input device is a real-world device that can be used to give a signal to the input module. The input module processes signals from input devices and gives them to the processor in a form it can understand. It is in the processor that the logical decisions are made. These decisions are based on the program that has been loaded into its memory. Memory is a place where data is stored. It is typically divided into program files and data files. A PLC can have many output modules, depending on the size of the application. An output device is a device that needs a signal to be activated.

A program loader is a device used to load the program into the processor. It is most often a personal computer, but it can be a dedicated piece of equipment made by the controller manufacturer. The programming device allows the user to create new programs or edit existing ones. It is also helpful in troubleshooting. A programmable logic controller has an internal power supply, which is sometimes part of the controller itself, and an external power supply, which is not part of the controller.

Answers to Self-Check Quiz

1-1.	b.	Energizes the electromagnet. Ref: 1.04
1-2.	d.	They have separate control and load circuits. Ref: 1.07
1-3.	a.	Communicates with the programmable controller’s internal logic circuits. Ref: 1.11
1-4.	b.	Event-driven. Ref: 1.16
1-5.	b.	Controls must respond rapidly to changes in the real world. Ref: 1.21
1-6.	c.	Process signals from an input device. Ref: 1.46
1-7.	d.	Preset; accumulator. Ref: 1.72, 1.73
1-8.	c.	Floating point. Ref: 1.78
1-9.	c.	Motor starter. Ref: 1.80
1-10 .	d.	Program. Ref: 1.81

Contributions from the following source are appreciated:

Figure 1-4. Used with permission of Rockwell Automation, Inc.
Figure 1-9. Used with permission of Rockwell Automation, Inc.