Programmable Controllers

Theory and Implementation

Second Edition

L.A. Bryan
E.A. Bryan

INDUSTRIAL TEXT

PROGRAMMABLE CONTROLLERS
THEORY AND IMPLEMENTATION

Second Edition

L. A. Bryan
E. A. Bryan

Due to the nature of this publication and because of the different applications of programmable controllers, the readers or users and those responsible for applying the information herein contained must satisfy themselves to the acceptability of each application and the use of equipment therein mentioned. In no event shall the publisher and others involved in this publication be liable for direct, indirect, or consequential damages resulting from the use of any technique or equipment herein mentioned.

The illustrations, charts, and examples in this book are intended solely to illustrate the methods used in each application example. The publisher and others involved in this publication cannot assume responsibility or liability for actual use based on the illustrative uses and applications.

No patent liability is assumed with respect to use of information, circuits, illustrations, equipment, or software described in this text.

CONTENTS

SECTION 1 INTRODUCTORY CONCEPTS

Chapter 1 Introduction to Programmable Controllers

Chapter 2 Number Systems and Codes

Chapter 3 Logic Concepts

SECTION 2 COMPONENTS AND SYSTEMS

Chapter 4 Processors, the Power Supply, and Programming Devices

Chapter 5 The Memory System and I/O Interaction

SECTION 5 ADVANCED PLC TOPICS AND NETWORKS

This page intentionally left blank.

PREFACE

Since the first edition of this book in 1988, the capabilities of programmable logic controllers have grown by leaps and bounds. Likewise, the applications of PLCs have grown with them. In fact, in today's increasingly computer-controlled environment, it is almost impossible to find a technical industry that does not use programmable controllers in one form or another. To respond to these phenomenal changes, we introduce the second edition of *Programmable Controllers: Theory and Implementation*.

This second edition, like the first, provides a comprehensive theoretical, yet practical, look at all aspects of PLCs and their associated devices and systems. However, this version goes one step further with new chapters on advanced PLC topics, such as I/O bus networks, fuzzy logic, the IEC 1131-3 programming standard, process control, and PID algorithms. This new edition also presents revised, up-to-date information about existing topics, with expanded graphics and new, hands-on examples. Furthermore, the new layout of the book—with features like two-tone graphics, key terms lists, well-defined headings and sections, callout icons, and a revised, expanded glossary—makes the information presented even easier to understand.

This new edition has been a labor-intensive learning experience for all those involved. As with any task so large, we could never have done it alone. Therefore, we would like to thank the following companies for their help in bringing this book to press: Allen-Bradley Company—Industrial Computer Group, ASI-USA, B & R Industrial Automation, Bailey Controls Company, DeviceNet Vendors Association, ExperTune Software, Fieldbus Foundation, Hoffman Engineering Company, Honeywell—MicroSwitch Division, LANcity—Cable Modem Division of Bay Networks, Mitsubishi Electronics, Omron Electronics, Phoenix Contact, PLC Direct, PMC/BETA LP, Profibus Trade Organization, Schaevitz Engineering Company, Siemens Automation, Square D Company, Thermometrics, and WAGO.

We hope that you will find this book to be a valuable learning and reference tool. We have tried to present a variety of programmable control operations; however, with the unlimited variations in control systems, we certainly have not been able to provide an exhaustive list of PLC applications. Only you, armed with the knowledge gained through this book, can explore the true limits of programmable logic controllers.

Stephanie Philippo
Editor

ABOUT THE AUTHORS

LUIS BRYAN

Luis Bryan holds a Bachelor of Science in Electrical Engineering degree and a Master of Science in Electrical Engineering degree, both from the University of Tennessee. His major areas of expertise are digital systems, electronics, and computer engineering. During his graduate studies, Luis was involved in several projects with national and international governmental agencies.

Luis has extensive experience in the field of programmable controllers. He was involved in international marketing activities, as well as PLC applications development, for a major programmable controller manufacturer. He also worked for a consulting firm, providing market studies and company-specific consultations about PLCs. Furthermore, Luis has given lectures and seminars in Canada, Mexico, and South America about the uses of programmable controllers. He continues to teach seminars to industry and government entities, including the National Aeronautics and Space Administration (NASA).

Luis is an active member of several professional organizations, including the Institute of Electrical and Electronics Engineers (IEEE) and the IEEE's instrument and computer societies. He is a senior member of the Instrument Society of America, as well as a member of Phi Kappa Phi honor society and Eta Kappa Nu electrical engineering honor society. Luis has coauthored several other books about programmable controllers.

ERIC BRYAN

Eric Bryan graduated from the University of Tennessee with a Bachelor of Science in Electrical Engineering degree, concentrating in digital design and computer architecture. He received a Master of Science in Engineering degree from the Georgia Institute of Technology, where he participated in a special computer-integrated manufacturing (CIM) program. Eric's specialties are industrial automation methods, flexible manufacturing systems (FMS), and artificial intelligence. He is an advocate of artificial intelligence implementation and its application in industrial automation.

Eric worked for a leading automatic laser inspection systems company, as well as a programmable controller consulting firm. His industrial experience includes designing and implementing large inspection systems, along with developing PLC-based systems. Eric has coauthored other publications about PLCs and is a member of several professional and technical societies.

HOW TO USE THIS BOOK

Welcome to *Programmable Controllers: Theory and Implementation*. Before you begin reading, please review the following strategies for using this book. By following these study strategies, you will more thoroughly understand the information presented in the text and, thus, be better able to apply this knowledge in real-life situations.

BEFORE YOU BEGIN READING

- Look through the book to familiarize yourself with its structure.
- Read the table of contents to review the subjects you will be studying.
- Familiarize yourself with the icons used throughout the text:

     Chapter Highlights

     Key Terms

- Look at the appendices to see what reference materials have been provided.

AS YOU STUDY EACH CHAPTER

- Before you start a chapter, read the Chapter Highlights paragraph at the beginning of the chapter's text. This paragraph will give you an overview of what you'll learn, as well as explain how the information presented in the chapter fits into what you've already learned and what you will learn.
- Read the chapter, paying special attention to the bolded items. These are key terms that indicate important topics that you should understand after finishing the chapter.
- When you encounter an exercise, try to solve the problem yourself before looking at the solution. This way, you'll determine which topics you understand and which topics you should study further.

WHEN YOU FINISH EACH CHAPTER

- At the end of each chapter, look over the list of key terms to ensure that you understand all of the important subjects presented in the chapter. If you're not sure about a term, review it in the text.
- Review the exercises to ensure that you understand the logic and equations involved in each problem. Also, review the workbook and study guide, making sure that you can work all of the problems correctly.
- When you're sure that you thoroughly understand the information that has been presented, you're ready to move on to the next chapter.

INTRODUCTORY CONCEPTS

- Introduction to Programmable Controllers
- Number Systems and Codes
- Logic Concepts

This page intentionally left blank.

INTRODUCTION TO PROGRAMMABLE CONTROLLERS

I find the great thing in this world is not so much where we stand as in what direction we are moving.

—Oliver Wendell Holmes

Chapter Highlights

Every aspect of industry—from power generation to automobile painting to food packaging—uses programmable controllers to expand and enhance production. In this book, you will learn about all aspects of these powerful and versatile tools. This chapter will introduce you to the basics of programmable controllers—from their operation to their vast range of applications. In it, we will give you an inside look at the design philosophy behind their creation, along with a brief history of their evolution. We will also compare programmable controllers to other types of controls to highlight the benefits and drawbacks of each, as well as pinpoint situations where PLCs work best. When you finish this chapter, you will understand the fundamentals of programmable controllers and be ready to explore the number systems associated with them.

1-1 DEFINITION

Programmable logic controllers, also called *programmable controllers* or *PLCs*, are **solid-state** members of the computer family, using integrated circuits instead of electromechanical devices to implement control functions. They are capable of storing instructions, such as sequencing, timing, counting, arithmetic, data manipulation, and communication, to control industrial machines and processes. Figure 1-1 illustrates a conceptual diagram of a PLC application.
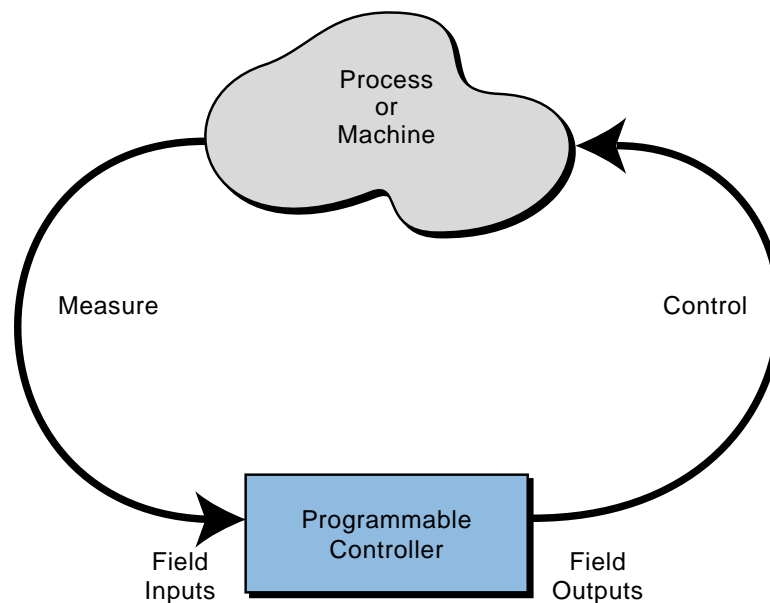
Figure 1-1. PLC conceptual application diagram.

Programmable controllers have many definitions. However, PLCs can be thought of in simple terms as industrial computers with specially designed architecture in both their central units (the PLC itself) and their interfacing circuitry to field devices (input/output connections to the real world).

As you will see throughout this book, programmable logic controllers are mature industrial controllers with their design roots based on the principles of simplicity and practical application.

1-2 A HISTORICAL BACKGROUND

The Hydramatic Division of the General Motors Corporation specified the design criteria for the first programmable controller in 1968. Their primary goal was to eliminate the high costs associated with inflexible, relay-controlled systems. The specifications required a solid-state system with computer flexibility able to (1) survive in an industrial environment, (2) be easily programmed and maintained by plant engineers and technicians, and (3) be reusable. Such a control system would reduce machine downtime and provide expandability for the future. Some of the initial specifications included the following:

- The new control system had to be price competitive with the use of relay systems.

- The system had to be capable of sustaining an industrial environment.

- The input and output interfaces had to be easily replaceable.

- The controller had to be designed in modular form, so that subassemblies could be removed easily for replacement or repair.

- The control system needed the capability to pass data collection to a central system.

- The system had to be reusable.

- The method used to program the controller had to be simple, so that it could be easily understood by plant personnel.

THE FIRST PROGRAMMABLE CONTROLLER

The product implementation to satisfy Hydramatic's specifications was underway in 1968; and by 1969, the programmable controller had its first product offsprings. These early controllers met the original specifications and opened the door to the development of a new control technology.

The first PLCs offered relay functionality, thus replacing the original hardwired **relay logic**, which used electrically operated devices to mechanically switch electrical circuits. They met the requirements of modularity, expandability, programmability, and ease of use in an industrial environment. These controllers were easily installed, used less space, and were reusable. The controller programming, although a little tedious, had a recognizable plant standard: the ladder diagram format.

In a short period, programmable controller use started to spread to other industries. By 1971, PLCs were being used to provide relay replacement as the first steps toward control automation in other industries, such as food and beverage, metals, manufacturing, and pulp and paper.

THE CONCEPTUAL DESIGN OF THE PLC

The first programmable controllers were more or less just relay replacers. Their primary function was to perform the sequential operations that were previously implemented with relays. These operations included ON/OFF control of machines and processes that required repetitive operations, such as transfer lines and grinding and boring machines. However, these programmable controllers were a vast improvement over relays. They were easily installed, used considerably less space and energy, had diagnostic indicators that aided troubleshooting, and unlike relays, were reusable if a project was scrapped.

Programmable controllers can be considered newcomers when they are compared to their elder predecessors in traditional control equipment technology, such as old hardwired relay systems, analog instrumentation, and other types of early solid-state logic. Although PLC functions, such as speed of operation, types of interfaces, and data-processing capabilities, have improved throughout the years, their specifications still hold to the designers' original intentions—they are simple to use and maintain.

TODAY'S PROGRAMMABLE CONTROLLERS

Many technological advances in the programmable controller industry continue today. These advances not only affect programmable controller design, but also the philosophical approach to control system architecture. Changes include both **hardware** (physical components) and **software** (control program) upgrades. The following list describes some recent PLC hardware enhancements:

- Faster scan times are being achieved using new, advanced microprocessor and electronic technology.

- Small, low-cost PLCs (see Figure 1-2), which can replace four to ten relays, now have more power than their predecessor, the simple relay replacer.

- High-density input/output (I/O) systems (see Figure 1-3) provide space-efficient interfaces at low cost.

- Intelligent, microprocessor-based I/O interfaces have expanded distributed processing. Typical interfaces include PID (proportional-

integral-derivative), network, CANbus, fieldbus, ASCII communication, positioning, host computer, and language modules (e.g., BASIC, Pascal).

- Mechanical design improvements have included rugged input/output enclosures and input/output systems that have made the terminal an integral unit.

- Special interfaces have allowed certain devices to be connected directly to the controller. Typical interfaces include thermocouples, strain gauges, and fast-response inputs.

- Peripheral equipment has improved operator interface techniques, and system documentation is now a standard part of the system.

Courtesy of Mitsubishi Electronics, Mount Prospect, IL

Figure 1-2. Small PLC with built-in I/O and detachable, handheld programming unit.
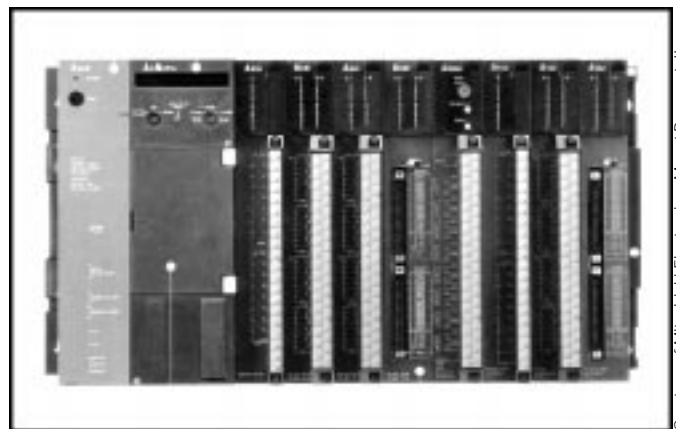
Courtesy of Mitsubishi Electronics, Mount Prospect, IL

Figure 1-3. PLC system with high-density I/O (64-point modules).

All of these hardware enhancements have led to the development of programmable controller families like the one shown in Figure 1-4. These families consist of a product line that ranges from very small "microcontrollers," with as few as 10 I/O points, to very large and

sophisticated PLCs, with as many as 8,000 I/O points and 128,000 words of memory. These family members, using common I/O systems and programming peripherals, can interface to a local communication network. The family concept is an important cost-saving development for users.

Figure 1-4. Allen-Bradley's programmable controller family concept with several PLCs.

Like hardware advances, software advances, such as the ones listed below, have led to more powerful PLCs:

- PLCs have incorporated object-oriented programming tools and multiple languages based on the IEC 1131-3 standard.

- Small PLCs have been provided with powerful instructions, which extend the area of application for these small controllers.

- High-level languages, such as BASIC and C, have been implemented in some controllers' modules to provide greater programming flexibility when communicating with peripheral devices and manipulating data.

- Advanced functional block instructions have been implemented for ladder diagram instruction sets to provide enhanced software capability using simple programming commands.

- Diagnostics and fault detection have been expanded from simple system diagnostics, which diagnose controller malfunctions, to include machine diagnostics, which diagnose failures or malfunctions of the controlled machine or process.

- Floating-point math has made it possible to perform complex calculations in control applications that require gauging, balancing, and statistical computation.

- Data handling and manipulation instructions have been improved and simplified to accommodate complex control and data acquisition applications that involve storage, tracking, and retrieval of large amounts of data.

Programmable controllers are now mature control systems offering many more capabilities than were ever anticipated. They are capable of communicating with other control systems, providing production reports, scheduling production, and diagnosing their own failures and those of the machine or process. These enhancements have made programmable controllers important contributors in meeting today's demands for higher quality and productivity. Despite the fact that programmable controllers have become much more sophisticated, they still retain the simplicity and ease of operation that was intended in their original design.

PROGRAMMABLE CONTROLLERS AND THE FUTURE

The future of programmable controllers relies not only on the continuation of new product developments, but also on the integration of PLCs with other control and factory management equipment. PLCs are being incorporated, through networks, into computer-integrated manufacturing (CIM) systems, combining their power and resources with numerical controls, robots, CAD/CAM systems, personal computers, management information systems, and hierarchical computer-based systems. There is no doubt that programmable controllers will play a substantial role in the factory of the future.

New advances in PLC technology include features such as better operator interfaces, graphic user interfaces (GUIs), and more human-oriented man/machine interfaces (such as voice modules). They also include the development of interfaces that allow communication with equipment, hardware, and software that supports artificial intelligence, such as fuzzy logic I/O systems.

Software advances provide better connections between different types of equipment, using communication standards through widely used networks. New PLC instructions are developed out of the need to add intelligence to a controller. Knowledge-based and process learning–type instructions may be introduced to enhance the capabilities of a system.

The user's concept of the flexible manufacturing system (FMS) will determine the control philosophy of the future. The future will almost certainly continue to cast programmable controllers as an important player in the factory. Control strategies will be distributed with "intelligence" instead of being centralized. Super PLCs will be used in applications requiring complex calculations, network communication, and supervision of smaller PLCs and machine controllers.

1-3 PRINCIPLES OF OPERATION

A programmable controller, as illustrated in Figure 1-5, consists of two basic sections:

- the central processing unit
- the input/output interface system
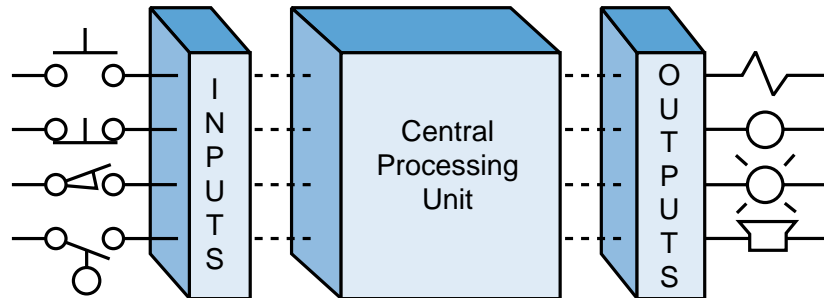
Figure 1-5. Programmable controller block diagram.

The central processing unit (CPU) governs all PLC activities. The following three components, shown in Figure 1-6, form the CPU:

- the processor
- the memory system
- the system power supply
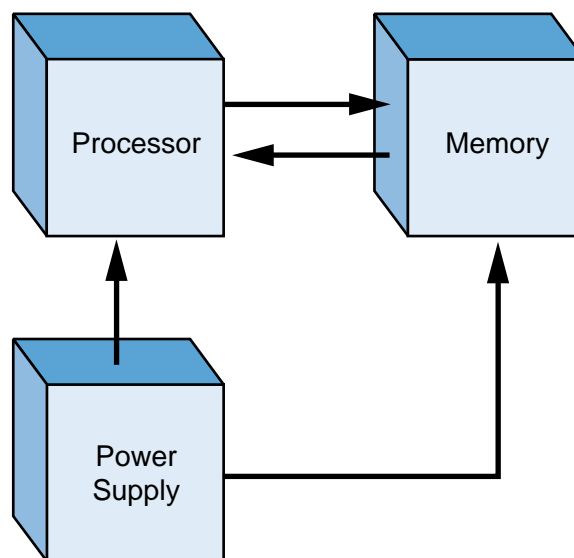
Figure 1-6. Block diagram of major CPU components.

The operation of a programmable controller is relatively simple. The **input/output (I/O) system** is physically connected to the field devices that are encountered in the machine or that are used in the control of a process. These field devices may be discrete or analog input/output devices, such as limit switches, pressure transducers, push buttons, motor starters, solenoids, etc. The I/O interfaces provide the connection between the CPU and the information providers (inputs) and controllable devices (outputs).

During its operation, the CPU completes three processes: (1) it **reads**, or accepts, the input data from the field devices via the input interfaces, (2) it **executes**, or performs, the control program stored in the memory system, and (3) it **writes**, or updates, the output devices via the output interfaces. This process of sequentially reading the inputs, executing the program in memory, and updating the outputs is known as **scanning.** Figure 1-7 illustrates a graphic representation of a scan.

SCAN

READ (1)

EXECUTE (2)

WRITE (3)
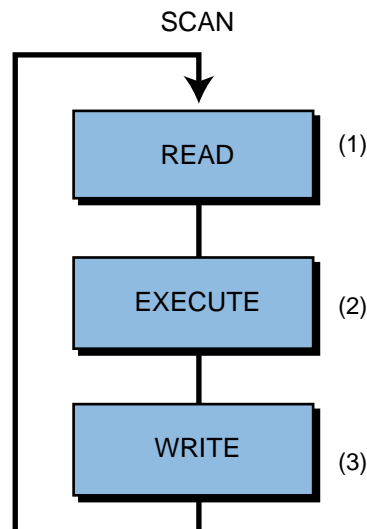
Figure 1-7. Illustration of a scan.

The input/output system forms the **interface** by which field devices are connected to the controller (see Figure 1-8). The main purpose of the interface is to condition the various signals received from or sent to external field devices. Incoming signals from sensors (e.g., push buttons, limit switches, analog sensors, selector switches, and thumbwheel switches) are wired to terminals on the input interfaces. Devices that will be controlled, like motor starters, solenoid valves, pilot lights, and position valves, are connected to the terminals of the output interfaces. The system **power supply** provides all the voltages required for the proper operation of the various central processing unit sections.
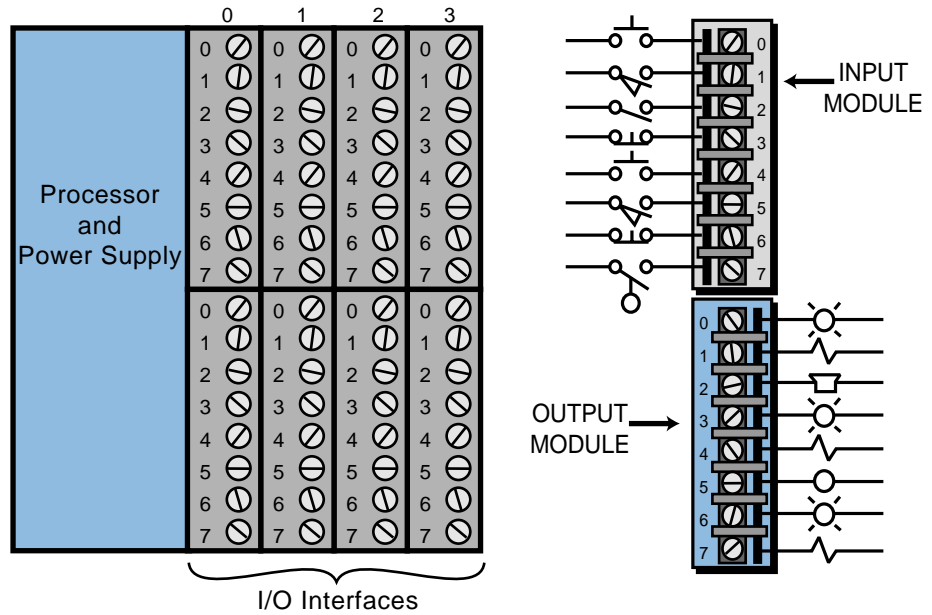
Figure 1-8. Input/output interface.

Although not generally considered a part of the controller, the **programming device**, usually a personal computer or a manufacturer's miniprogrammer unit, is required to enter the control program into memory (see Figure 1-9). The programming device must be connected to the controller when entering or monitoring the control program.

(a)

(b)
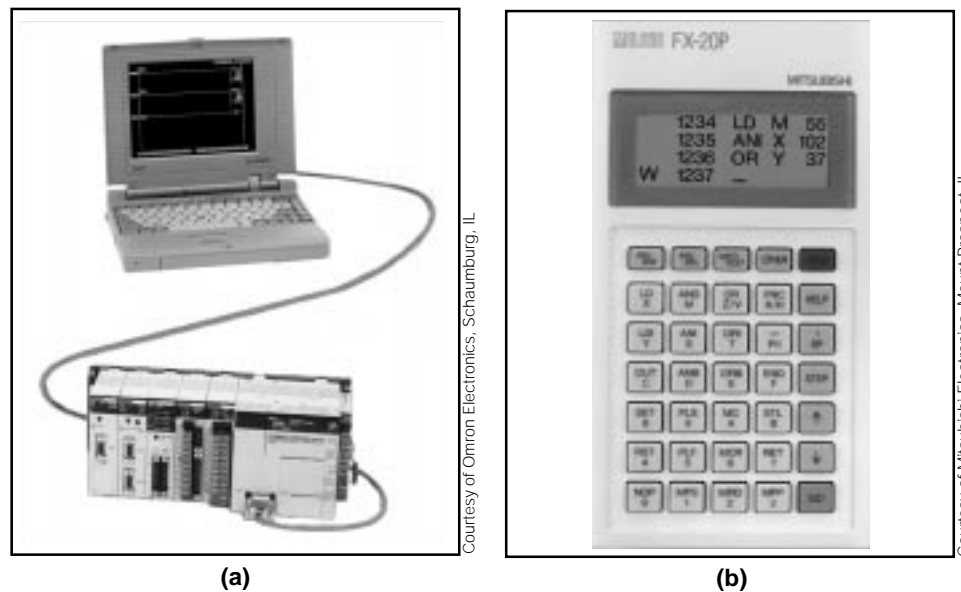
Figure 1-9. (a) Personal computer used as a programming device and **(b)** a mini-programmer unit.

Chapters 4 and 5 will present a more detailed discussion of the central processing unit and how it interacts with memory and input/output interfaces. Chapters 6, 7, and 8 discuss the input/output system.

1-4 PLCs VERSUS OTHER TYPES OF CONTROLS

PLCs VERSUS RELAY CONTROL

For years, the question many engineers, plant managers, and original equipment manufacturers (OEMs) asked was, "Should I be using a programmable controller?" At one time, much of a systems engineer's time was spent trying to determine the cost-effectiveness of a PLC over relay control. Even today, many control system designers still think that they are faced with this decision. One thing, however, is certain—today's demand for high quality and productivity can hardly be fulfilled economically without electronic control equipment. With rapid technology developments and increasing competition, the cost of programmable controls has been driven down to the point where a PLC-versus-relay cost study is no longer necessary or valid. Programmable controller applications can now be evaluated on their own merits.

When deciding whether to use a PLC-based system or a hardwired relay system, the designer must ask several questions. Some of these questions are:

- Is there a need for flexibility in control logic changes?

- Is there a need for high reliability?

- Are space requirements important?

- Are increased capability and output required?

- Are there data collection requirements?

- Will there be frequent control logic changes?

- Will there be a need for rapid modification?

- Must similar control logic be used on different machines?

- Is there a need for future growth?

- What are the overall costs?

The merits of PLC systems make them especially suitable for applications in which the requirements listed above are particularly important for the economic viability of the machine or process operation. A case which speaks for itself, the system shown in Figure 1-10, shows why programmable controllers are easily favored over relays. The implementation of this system using electromechanical standard and timing relays would have made this control panel a maze of large bundles of wires and interconnections.

SECTION
1

*Introductory
Concepts*

*Introduction to
Programmable Controllers*

CHAPTER
1
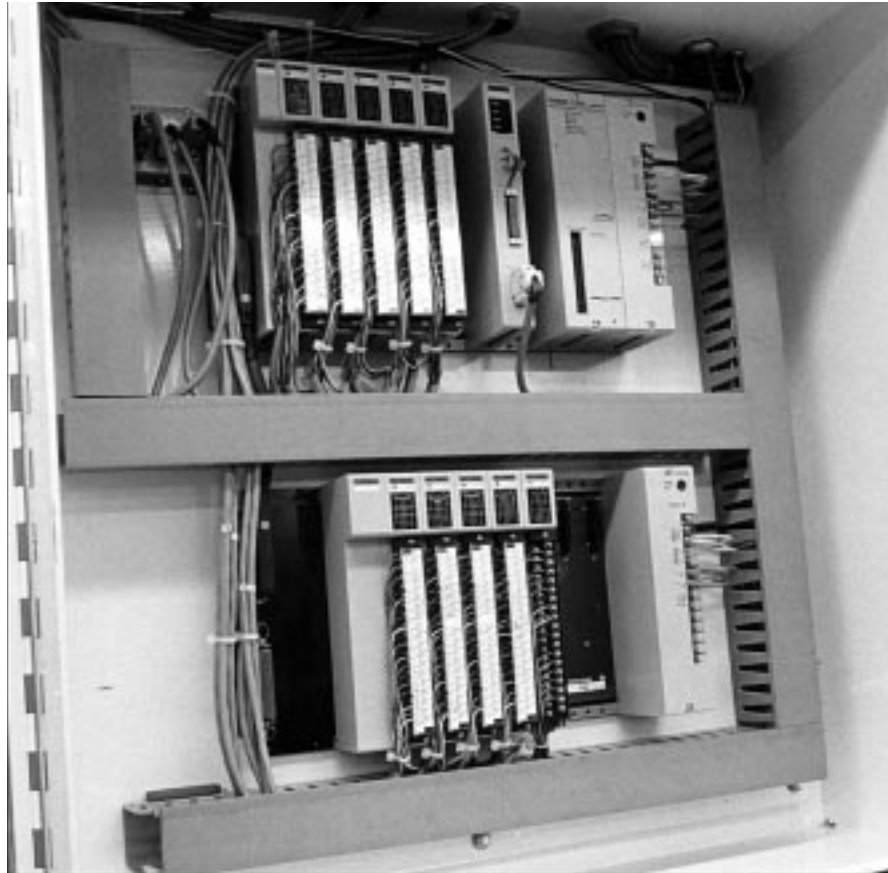
Courtesy of Omron Electronics, Schaumburg, IL

Figure 1-10. The uncluttered control panel of an installed PLC system.

If system requirements call for flexibility or future growth, a programmable controller brings returns that outweigh any initial cost advantage of a relay control system. Even in a case where no flexibility or future expansion is required, a large system can benefit tremendously from the troubleshooting and maintenance aids provided by a PLC. The extremely short cycle (scan) time of a PLC allows the productivity of machines that were previously under electromechanical control to increase considerably. Also, although relay control may cost less initially, this advantage is lost if production downtime due to failures is high.

PLCs VERSUS COMPUTER CONTROLS

The architecture of a PLC's CPU is basically the same as that of a general purpose computer; however, some important characteristics set them apart. First, unlike computers, PLCs are specifically designed to survive the harsh conditions of the industrial environment. A well-designed PLC can be placed in an area with substantial amounts of electrical noise, electromagnetic interference, mechanical vibration, and noncondensing humidity.

A second distinction of PLCs is that their hardware and software are designed for easy use by plant electricians and technicians. The hardware interfaces for connecting field devices are actually part of the PLC itself and are easily connected. The modular and self-diagnosing interface circuits are able to pinpoint malfunctions and, moreover, are easily removed and replaced. Also, the software programming uses conventional relay ladder symbols, or other easily learned languages, which are familiar to plant personnel.

Whereas computers are complex computing machines capable of executing several programs or tasks simultaneously and in any order, the standard PLC executes a single program in an orderly, sequential fashion from first to last instruction. Bear in mind, however, that PLCs as a system continue to become more intelligent. Complex PLC systems now provide multiprocessor and multitasking capabilities, where one PLC may control several programs in a single CPU enclosure with several processors (see Figure 1-11).
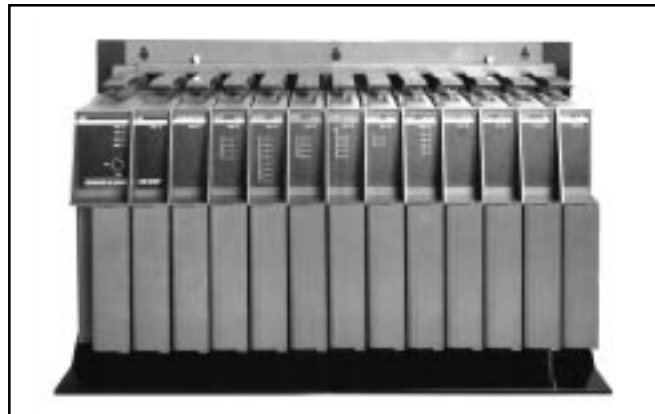
Courtesy of Giddings & Lewis, Fond du Lac, WI

Figure 1-11. PLC system with multiprocessing and multitasking capabilities.

PLCs VERSUS PERSONAL COMPUTERS

With the proliferation of the personal computer (PC), many engineers have found that the personal computer is not a direct competitor of the PLC in control applications. Rather, it is an ally in the implementation of the control solution. The personal computer and the PLC possess similar CPU architecture; however, they distinctively differ in the way they connect field devices.

While new, rugged, industrial personal computers can sometimes sustain midrange industrial environments, their interconnection to field devices still presents difficulties. These computers must communicate with I/O interfaces not necessarily designed for them, and their programming languages may not meet the standards of ladder diagram programming. This presents a problem to people familiar with the ladder diagram standard when troubleshooting and making changes to the system.

The personal computer is, however, being used as the programming device of choice for PLCs in the market, where PLC manufacturers and third-party PLC support developers come up with programming and documentation systems for their PLC product lines. Personal computers are also being employed to gather process data from PLCs and to display information about the process or machine (i.e., they are being used as graphic user interfaces, or GUIs). Because of their number-crunching capabilities, personal computers are also well suited to complement programmable controllers and to bridge the communication gap, through a network, between a PLC system and other mainframe computers (see Figure 1-12).

Main
Computer
System

Personal
Computer

PLC
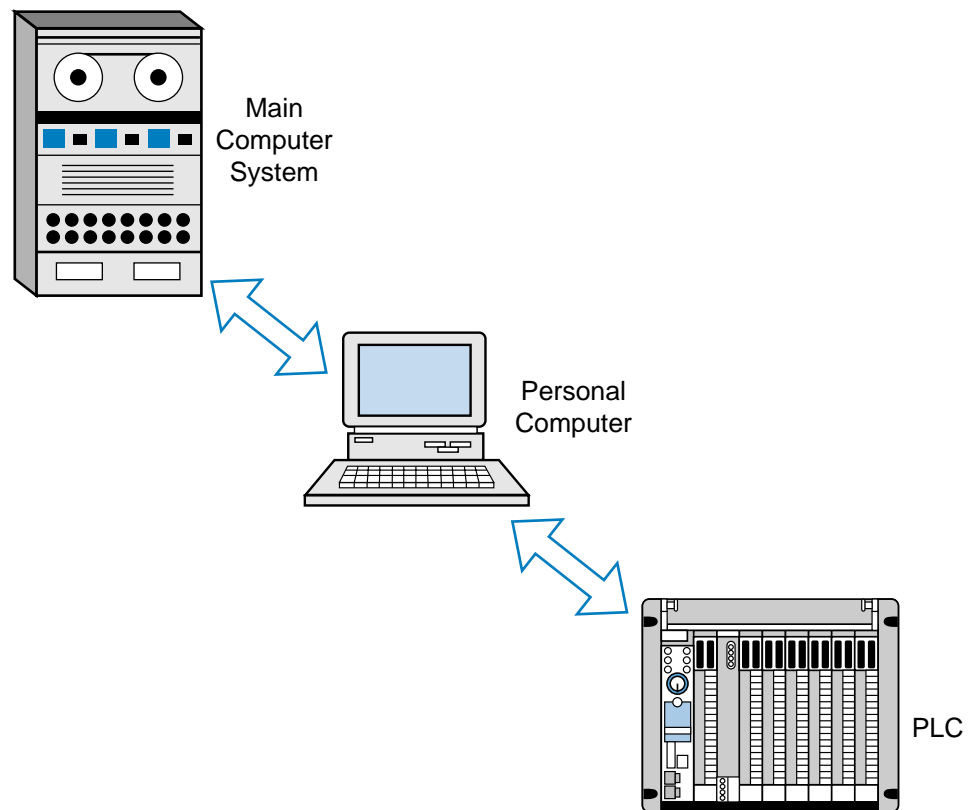
Figure 1-12. A personal computer used as a bridge between a PLC system and a main computer system.

Some control software manufacturers, however, utilize PCs as CPU hardware to implement a PLC-like environment. The language they use is based on the International Electrotechnical Commission (IEC) 1131-3 standard, which is a graphic representation language (sequential function charts) that includes ladder diagrams, functional blocks, instruction lists, and structured text. These software manufacturers generally do not provide I/O hardware interfaces; but with the use of internal PC communication cards, these systems can communicate with other PLC manufacturers' I/O hardware modules. Chapter 10 explains the IEC 1131-3 standard.

TYPICAL AREAS OF PLC APPLICATIONS

Since its inception, the PLC has been successfully applied in virtually every segment of industry, including steel mills, paper plants, food-processing plants, chemical plants, and power plants. PLCs perform a great variety of control tasks, from repetitive ON/OFF control of simple machines to sophisticated manufacturing and process control. Table 1-1 lists a few of the major industries that use programmable controllers, as well as some of their typical applications.

CHEMICAL/PETROCHEMICAL
Batch process
Finished product handling
Materials handling
Mixing
Off-shore drilling
Pipeline control
Water/waste treatment

GLASS/FILM
Cullet weighing
Finishing
Forming
Lehr control
Packaging
Processing

FOOD/BEVERAGE
Accumulating conveyors
Blending
Brewing
Container handling
Distilling
Filling
Load forming
Metal forming loading/unloading
Palletizing
Product handling
Sorting conveyors
Warehouse storage/retrieval
Weighing

LUMBER/PULP/PAPER
Batch digesters
Chip handling
Coating
Wrapping/stamping

MANUFACTURING/MACHINING
Assembly machines
Boring
Cranes
Energy demand
Grinding
Injection/blow molding
Material conveyors
Metal casting
Milling
Painting
Plating
Test stands
Tracer lathe
Welding

METALS
Blast furnace control
Continuous casting
Rolling mills
Soaking pit

MINING
Bulk material conveyors
Loading/unloading
Ore processing
Water/waste management

POWER
Burner control
Coal handling
Cut-to-length processing
Flue control
Load shedding
Sorting
Winding/processing
Woodworking

Table 1-1. Typical programmable controller applications.

Because the applications of programmable controllers are extensive, it is impossible to list them all in this book. However, Table 1-2 provides a small sample of how PLCs are being used in industry.

AUTOMOTIVE

Internal Combustion Engine Monitoring. A PLC acquires data recorded from sensors located at the internal combustion engine. Measurements taken include water temperature, oil temperature, RPMs, torque, exhaust temperature, oil pressure, manifold pressure, and timing.

Carburetor Production Testing. PLCs provide on-line analysis of automotive carburetors in a production assembly line. The systems significantly reduce the test time, while providing greater yield and better quality carburetors. Pressure, vacuum, and fuel and air flow are some of the variables tested.

Monitoring Automotive Production Machines. The system monitors total parts, rejected parts, parts produced, machine cycle time, and machine efficiency. Statistical data is available to the operator anytime or after each shift.

Power Steering Valve Assembly and Testing. The PLC system controls a machine to ensure proper balance of the valves and to maximize left and right turning ratios.

CHEMICAL AND PETROCHEMICAL

Ammonia and Ethylene Processing. Programmable controllers monitor and control large compressors used during ammonia and ethylene manufacturing. The PLC monitors bearing temperatures, operation of clearance pockets, compressor speed, power consumption, vibration, discharge temperatures, pressure, and suction flow.

Dyes. PLCs monitor and control the dye processing used in the textile industry. They match and blend colors to predetermined values.

Chemical Batching. The PLC controls the batching ratio of two or more materials in a continuous process. The system determines the rate of discharge of each material and keeps inventory records. Several batch recipes can be logged and retreived automatically or on command from the operator.

Fan Control. PLCs control fans based on levels of toxic gases in a chemical production environment. This system effectively removes gases when a preset level of contamination is reached. The PLC controls the fan start/stop, cycling, and speeds, so that safety levels are maintained while energy consumption is minimized.

Gas Transmission and Distribution. Programmable controllers monitor and regulate pressures and flows of gas transmission and distribution systems. Data is gathered and measured in the field and transmitted to the PLC system.

Pipeline Pump Station Control. PLCs control mainline and booster pumps for crude oil distribution. They measure flow, suction, discharge, and tank low/high limits. Possible communication with SCADA (Supervisory Control and Data Acquistion) systems can provide total supervision of the pipeline.

Oil Fields. PLCs provide on-site gathering and processing of data pertinent to characteristics such as depth and density of drilling rigs. The PLC controls and monitors the total rig operation and alerts the operator of any possible malfunctions.

Table 1-2. Examples of PLC applications.

GLASS PROCESSING

Annealing Lehr Control. PLCs control the lehr used to remove the internal stress from glass products. The system controls the operation by following the annealing temperature curve during the reheating, annealing, straining, and rapid cooling processes through different heating and cooling zones. Improvements are made in the ratio of good glass to scrap, reduction in labor cost, and energy utilization.

Glass Batching. PLCs control the batch weighing system according to stored glass formulas. The system also controls the electromagnetic feeders for infeed to and outfeed from the weigh hoppers, manual shut-off gates, and other equipment.

Cullet Weighing. PLCs direct the cullet system by controlling the vibratory cullet feeder, weight-belt scale, and shuttle conveyor. All sequences of operation and inventory of quantities weighed are kept by the PLC for future use.

Batch Transport. PLCs control the batch transport system, including reversible belt conveyors, transfer conveyors to the cullet house, holding hoppers, shuttle conveyors, and magnetic separators. The controller takes action after the discharge from the mixer and transfers the mixed batch to the furnace shuttle, where it is discharged to the full length of the furnace feed hopper.

MANUFACTURING/MACHINING

Production Machines. The PLC controls and monitors automatic production machines at high efficiency rates. It also monitors piece-count production and machine status. Corrective action can be taken immediately if the PLC detects a failure.

Transfer Line Machines. PLCs monitor and control all transfer line machining station operations and the interlocking between each station. The system receives inputs from the operator to check the operating conditions on the line-mounted controls and reports any malfunctions. This arrangement provides greater machine efficiency, higher quality products, and lower scrap levels.

Wire Machine. The controller monitors the time and ON/OFF cycles of a wire-drawing machine. The system provides ramping control and synchronization of electric motor drives. All cycles are recorded and reported on demand to obtain the machine's efficiency as calculated by the PLC.

Tool Changing. The PLC controls a sychronous metal cutting machine with several tool groups. The system keeps track of when each tool should be replaced, based on the number of parts it manufactures. It also displays the count and replacements of all the tool groups.

Paint Spraying. PLCs control the painting sequences in auto manufacturing. The operator or a host computer enters style and color information and tracks the part through the conveyor until it reaches the spray booth. The controller decodes the part information and then controls the spray guns to paint the part. The spray gun movement is optimized to conserve paint and increase part throughput.

MATERIALS HANDLING

Automatic Plating Line. The PLC controls a set pattern for the automated hoist, which can traverse left, right, up, and down through the various plating solutions. The system knows where the hoist is at all times.

Table 1-2 continued.

Storage and Retrieval Systems. A PLC is used to load parts and carry them in totes in the storage and retrieval system. The controller tracks information like □ lane numbers, the parts assigned to specific lanes, and the quantity of parts in a particular lane. This PLC arrangement allows rapid changes in the status of parts loaded or unloaded from the system. The controller also provides inventory printouts and informs the operator of any malfunctions.

Conveyor Systems. The system controls all of the sequential operations, alarms, and safety logic necessary to load and circulate parts on a main line conveyor. It also sorts products to their correct lanes and can schedule lane sorting to optimize palletizer duty. Records detailing the ratio of good parts to rejects can be obtained at the end of each shift.

Automated Warehousing. The PLC controls and optimizes the movement of stacking cranes and provides high turnaround of materials requests in an automated, high-cube, vertical warehouse. The PLC also controls aisle conveyors and case palletizers to significantly reduce manpower requirements. Inventory control figures are maintained and can be provided on request.

METALS

Steel Making. The PLC controls and operates furnaces to produce metal in accordance with preset specifications. The controller also calculates oxygen requirements, alloy additions, and power requirements.

Loading and Unloading of Alloys. Through accurate weighing and loading sequences, the system controls and monitors the quantity of coal, iron ore, and limestone to be melted. It can also control the unloading sequence of the steel to a torpedo car.

Continuous Casting. PLCs direct the molten steel transport ladle to the continuous-casting machine, where the steel is poured into a water-cooled mold for solidification.

Cold Rolling. PLCs control the conversion of semifinished products into finished goods through cold-rolling mills. The system controls motor speed to obtain correct tension and provide adequate gauging of the rolled material.

Aluminum Making. Controllers monitor the refining process, in which impurities are removed from bauxite by heat and chemicals. The system grinds and mixes the ore with chemicals and then pumps them into pressure containers, where they are heated, filtered, and combined with more chemicals.

POWER

Plant Power System. The programmable controller regulates the proper distribution of available electricity, gas, or steam. In addition, the PLC monitors powerhouse facilities, schedules distribution of energy, and generates distribution reports. The PLC controls the loads during operation of the plant, as well as the automatic load shedding or restoring during power outages.

Energy Management. Through the reading of inside and outside temperatures, the PLC controls heating and cooling units in a manufacturing plant. The PLC system controls the loads, cycling them during predetermined cycles and keeping track of how long each should be on or off during the cycle time. The system provides scheduled reports on the amount of energy used by the heating and cooling units.

Table 1-2 continued.

Coal Fluidization Processing. The controller monitors how much energy is generated from a given amount of coal and regulates the coal crushing and mixing with crushed limestone. The PLC monitors and controls burning rates, temperatures generated, sequencing of valves, and analog control of jet valves.

Compressor Efficiency Control. PLCs control several compressors at a typical compressor station. The system handles safety interlocks, startup/shutdown sequences, and compressor cycling. The PLCs keep compressors running at maximum efficiency using the nonlinear curves of the compressors.

PULP AND PAPER

Pulp Batch Blending. The PLC controls sequence operation, ingredient measurement, and recipe storage for the blending process. The system allows operators to modify batch entries of each quantity, if necessary, and provides hardcopy printouts for inventory control and for accounting of ingredients used.

Batch Preparation for Paper-Making Processing. Applications include control of the complete stock preparation system for paper manufacturing. Recipes for each batch tank are selected and adjusted via operator entries. PLCs can control feedback logic for chemical addition based on tank level measurement signals. At the completion of each shift, the PLC system provides management reports on materials use.

Paper Mill Digester. PLCs control the process of making paper pulp from wood chips. The system calculates and controls the amount of chips based on density and digester volume. Then, the percent of required cooking liquors is calculated and these amounts are added to the sequence. The PLC ramps and holds the cooking temperature until the cooking is completed.

Paper Mill Production. The controller regulates the average basis weight and moisture variable for paper grade. The system manipulates the steam flow valves, adjusts the stock valves to regulate weight, and monitors and controls total flow.

RUBBER AND PLASTIC

Tire-Curing Press Monitoring. The PLC performs individual press monitoring for time, pressure, and temperature during each press cycle. The system alerts the operator of any press malfunctions. Information concerning machine status is stored in tables for later use. Report generation printouts for each shift include a summary of good cures and press downtime due to malfunctions.

Tire Manufacturing. Programmable controllers are used for tire press/cure systems to control the sequencing of events that transforms a raw tire into a tire fit for the road. This control includes molding the tread pattern and curing the rubber to obtain road-resistant characteristics. This PLC application substantially reduces the space required and increases reliability of the system and the quality of the product.

Rubber Production. PLCs provide accurate scale control, mixer logic functions, and multiple formula operation of carbon black, oil, and pigment used in the production of rubber. The system maximizes utilization of machine tools during production schedules, tracks in-process inventories, and reduces time and personnel required to supervise the production activity and the shift-end reports.

Plastic Injection Molding. A PLC system controls variables, such as temperature and pressure, which are used to optimize the injection molding process. The system provides closed-loop injection, where several velocity levels can be programmed to maintain consistent filling, reduce surface defects, and shorten cycle time.

Table 1-2 continued.

1-5 PLC PRODUCT APPLICATION RANGES

Figure 1-13 graphically illustrates programmable controller product ranges. This chart is not definitive, but for practical purposes, it is valid. The PLC market can be segmented into five groups:

1. micro PLCs

2. small PLCs

3. medium PLCs

4. large PLCs

5. very large PLCs
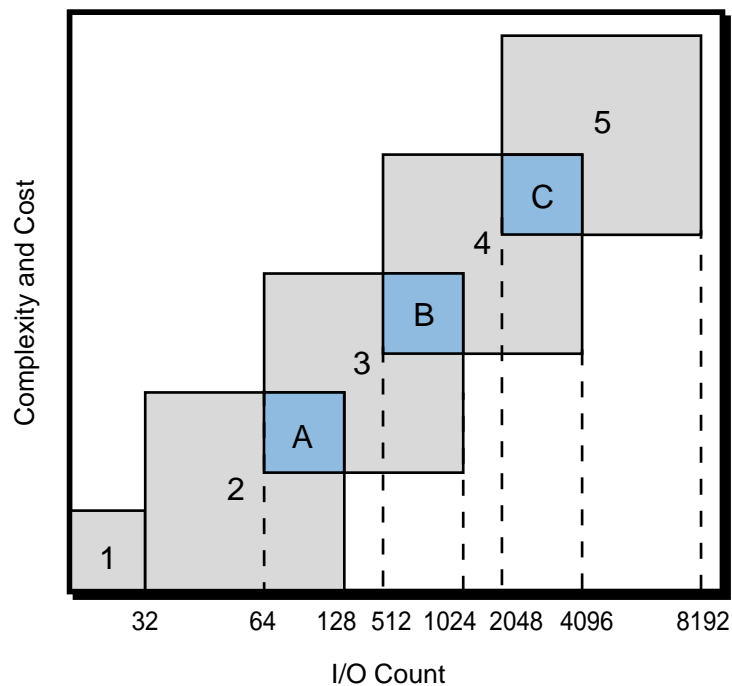
Figure 1-13. PLC product ranges.

Micro PLCs are used in applications controlling up to 32 input and output devices, 20 or less I/O being the norm. The micros are followed by the small PLC category, which controls 32 to 128 I/O. The medium (64 to 1024 I/O), large (512 to 4096 I/O), and very large (2048 to 8192 I/O) PLCs complete the segmentation. Figure 1-14 shows several PLCs that fall into this category classification.

The A, B, and C overlapping areas in Figure 1-13 reflect enhancements, by adding options, of the standard features of the PLCs within a particular segment. These options allow a product to be closely matched to the application without having to purchase the next larger unit. Chapter 20

covers, in detail, the differences between PLCs in overlapping areas. These differences include I/O count, memory size, programming language, software functions, and other factors. An understanding of the PLC product ranges and their characteristics will allow the user to properly identify the controller that will satisfy a particular application.

(a)

Courtesy of Mitsubishi Electronics, Mount Prospect, IL

(b)

Courtesy of PLC Direct, Cumming, GA

(c)

Courtesy of Giddings & Lewis, Fond du Lac, WI

(d)

Courtesy of Allen-Bradley, Highland Heights, OH

(e)

Courtesy of Omron Electronics, Schaumburg, IL

(f)

Courtesy of Allen-Bradley, Highland Heights, OH

Figure 1-14. (a) Mitsubishi's smallest print size PLC (14 I/O), **(b)** PLC Direct DL105 with 18 I/O and a capacity of 6 amps per output channel, **(c)** Giddings & Lewis PIC90 capable of handling 128 I/O with motion control capabilities, **(d)** Allen-Bradley's PLC 5/15 (512 I/O), **(e)** Omron's C200H PLC (1392 I/O), and **(f)** Allen-Bradley's PLC 5/80 (3072 I/O).

1-6 LADDER DIAGRAMS AND THE PLC

The **ladder diagram** has and continues to be the traditional way of representing electrical sequences of operations. These diagrams represent the interconnection of field devices in such a way that the activation, or turning ON, of one device will turn ON another device according to a predetermined sequence of events. Figure 1-15 illustrates a simple electrical ladder diagram.
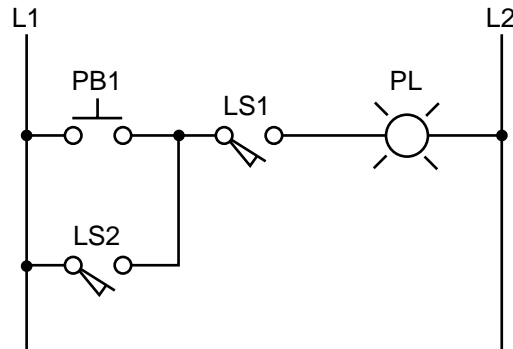
Figure 1-15. Simple electrical ladder diagram.

The original ladder diagrams were established to represent hardwired logic circuits used to control machines or equipment. Due to wide industry use, they became a standard way of communicating control information from the designers to the users of equipment. As programmable controllers were introduced, this type of circuit representation was also desirable because it was easy to use and interpret and was widely accepted in industry.

Programmable controllers can implement all of the "old" ladder diagram conditions and much more. Their purpose is to perform these control operations in a more reliable manner at a lower cost. A PLC implements, in its CPU, all of the old hardwired interconnections using its software instructions. This is accomplished using familiar ladder diagrams in a manner that is transparent to the engineer or programmer. As you will see throughout this book, a knowledge of PLC operation, scanning, and instruction programming is vital to the proper implementation of a control system.

Figure 1-16 illustrates the PLC transformation of the simple diagram shown in Figure 1-15 to a PLC format. Note that the "real" I/O field devices are connected to input and output interfaces, while the ladder program is implemented in a manner, similar to hardwiring, inside the programmable controller (i.e., *softwired* inside the PLC's CPU instead of *hardwired* in a panel). As previously mentioned, the CPU reads the status of inputs, energizes the corresponding circuit element according to the program, and controls a real output device via the output interfaces.

represents input module   represents output module
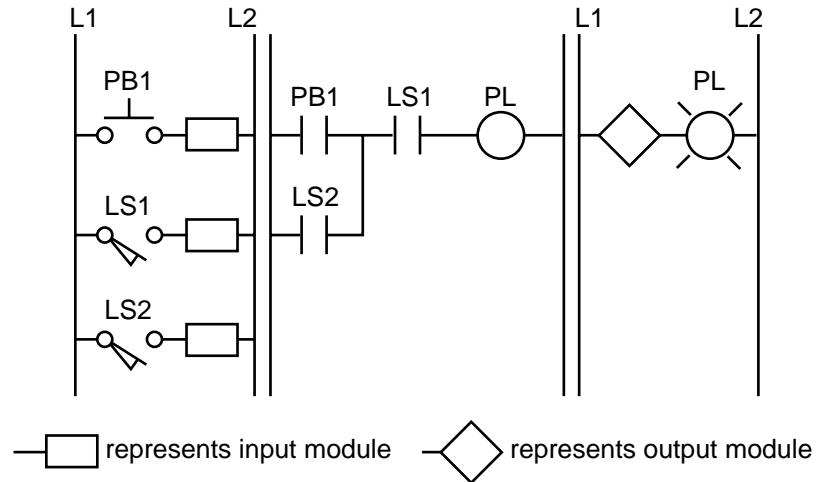
Figure 1-16. PLC implementation of Figure 1-15.

As you will see later, each instruction is represented inside the PLC by a reference **address**, an alphanumeric value by which each device is known in the PLC program. For example, the push button PB1 is represented inside the PLC by the name PB1 (indicated on top of the instruction symbol) and likewise for the other devices shown in Figure 1-16. These instructions are represented here, for simplicity, with the same device and instruction names. Chapters 3 and 5 further discuss basic addressing techniques, while Chapter 6 covers input/output wiring connections. Example 1-1 illustrates the similarity in operation between hardwired and PLC circuits.

EXAMPLE 1-1

In the hardwired circuit shown in Figure 1-15, the pilot light PL will turn ON if the limit switch LS1 closes <u>and</u> if either push button PB1 <u>or</u> limit switch LS2 closes. In the PLC circuit, the same series of events will cause the pilot light—connected to an output module—to turn ON. Note that in the PLC circuit in Figure 1-16, the internal representation of contacts provides the equivalent power logic as a hardwired circuit when the referenced input field device closes or is pushed. Sketch hardwired and PLC implementation diagrams for the circuit in Figure 1-15 illustrating the configurations of inputs that will turn PL ON.

SOLUTION

Figure 1-17 shows several possible configurations for the circuit in Figure 1-15. The highlighted blue lines indicate that power is present at that connection point, which is also the way a programming or monitoring device represents power in a PLC circuit. The last two configurations in Figure 1-17 are the only ones that will turn PL ON.
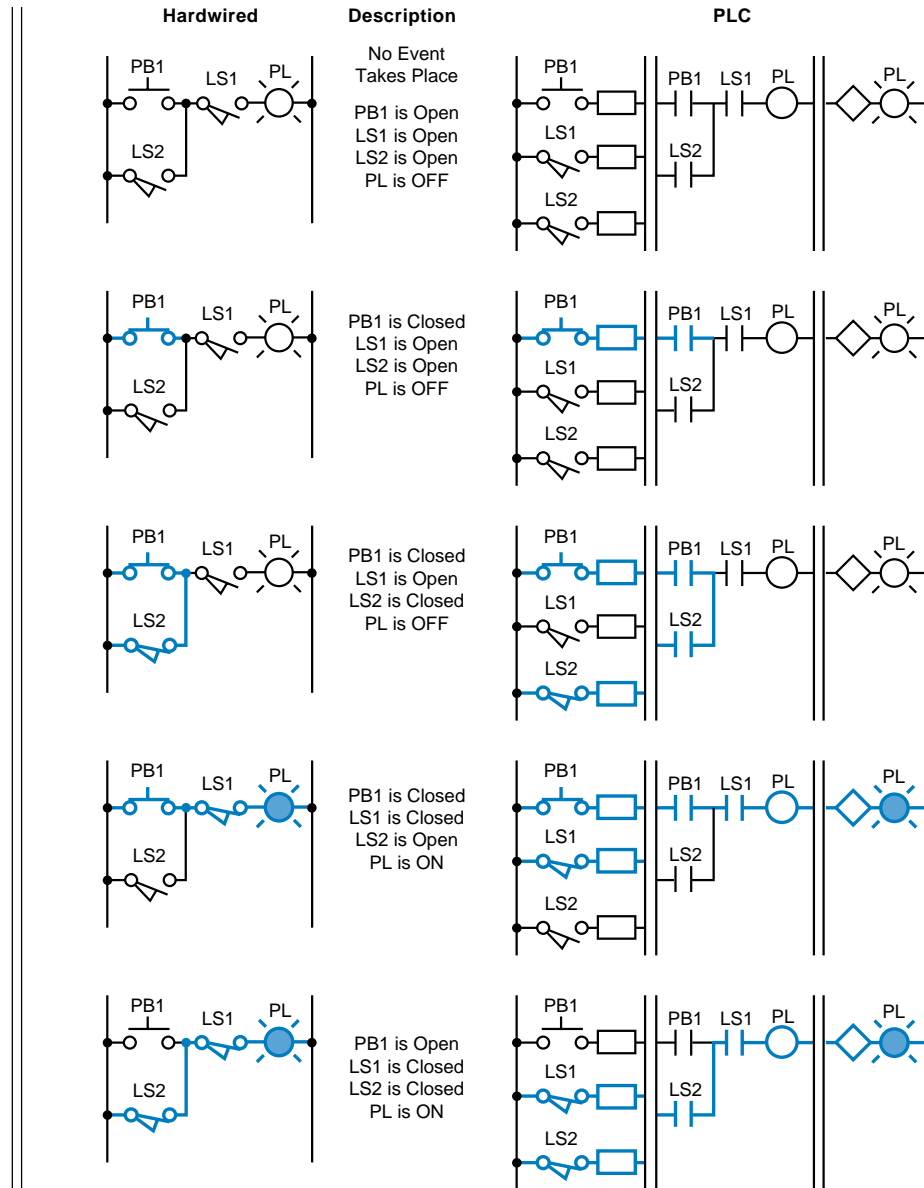
Figure 1-17. Possible configurations of inputs and corresponding outputs.

1-7 ADVANTAGES OF PLCS

In general, PLC architecture is modular and flexible, allowing hardware and software elements to expand as the application requirements change. In the event that an application outgrows the limitations of the programmable controller, the unit can be easily replaced with a unit having greater memory and I/O capacity, and the old hardware can be reused for a smaller application. A PLC system provides many benefits to control solutions, from reliability and repeatability to programmability. The benefits achieved with programmable controllers will grow with the individual using them—the more you learn about PLCs, the more you will be able to solve other control problems.

Table 1-3 lists some of the many features and benefits obtained with a programmable controller.

Inherent Features	Benefits
Solid-state components	• High reliability
Programmable memory	• Simplifies changes • Flexible control
Small size	• Minimal space requirements
Microprocessor-based	• Communication capability • Higher level of performance • Higher quality products • Multifunctional capability
Software timers/counters	• Eliminate hardware • Easily changed presets
Software control relays	• Reduce hardware/wiring cost • Reduce space requirements
Modular architecture	• Installation flexibility • Easily installed • Reduces hardware cost • Expandability
Variety of I/O interfaces	• Controls a variety of devices • Eliminates customized control
Remote I/O stations	• Eliminate long wire/conduit runs
Diagnostic indicators	• Reduce troubleshooting time • Signal proper operation
Modular I/O interface	• Neat appearance of control panel • Easily maintained • Easily wired
Quick I/O disconnects	• Service without disturbing wiring
System variables stored in memory data	• Useful management/maintenance • Can be output in report form

Table 1-3. Typical programmable controller features and benefits.

Without question, the "programmable" feature provides the single greatest benefit for the use and installation of programmable controllers. Eliminating hardwired control in favor of programmable control is the first step towards achieving a flexible control system. Once installed, the control plan can be manually or automatically altered to meet day-to-day control requirements without changing the field wiring. This easy alteration is possible since there are no physical connections between the field input devices and output devices (see Figure 1-18), as in hardwired systems. The only connection is through the control program, which can be easily altered.
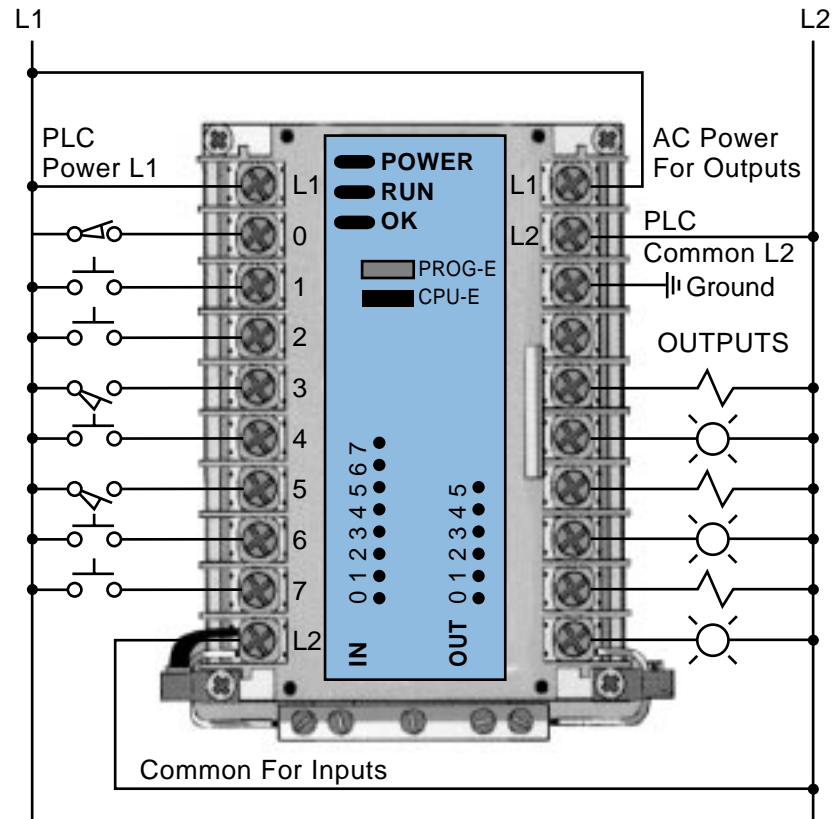
Figure 1-18. Programmable controller I/O connection diagram showing no physical connections between the inputs and outputs.

A typical example of the benefits of softwiring is a solenoid that is controlled by two limit switches connected in series (see Figure 1-19a). Changing the solenoid operation by placing the two limit switches in parallel (see Figure 1-19b) or by adding a third switch to the existing circuit (see Figure 1-19c) would take less than one minute in a PLC. In most cases, this simple program change can be made without shutting down the system. This same change to a hardwired system could take as much as thirty to sixty minutes of downtime, and even a half hour of downtime can mean a costly loss of production. A similar situation exists if there is a need to change a timer preset value or some other constant. A software timer in a PLC can be changed in as little as five seconds. A set of thumbwheel switches and a push button can be easily configured to input new preset values to any number of software timers. The time savings benefit of altering software timers, as opposed to altering several hardware timers, is obvious.

The hardware features of programmable controllers provide similar flexibility and cost savings. An intelligent CPU is capable of communicating with other intelligent devices. This capability allows the controller to be integrated into local or plantwide control schemes. With such a control

configuration, a PLC can send useful English messages regarding the controlled system to an intelligent display. On the other hand, a PLC can receive supervisory information, such as production changes or scheduling information, from a host computer. A standard I/O system includes a variety of digital, analog, and special interface modules, which allow sophisticated control without the use of expensive, customized interface electronics.

HARDWIRED   **PLC**

(a) Series

(b) Parallel

(c) Adding one LS in series
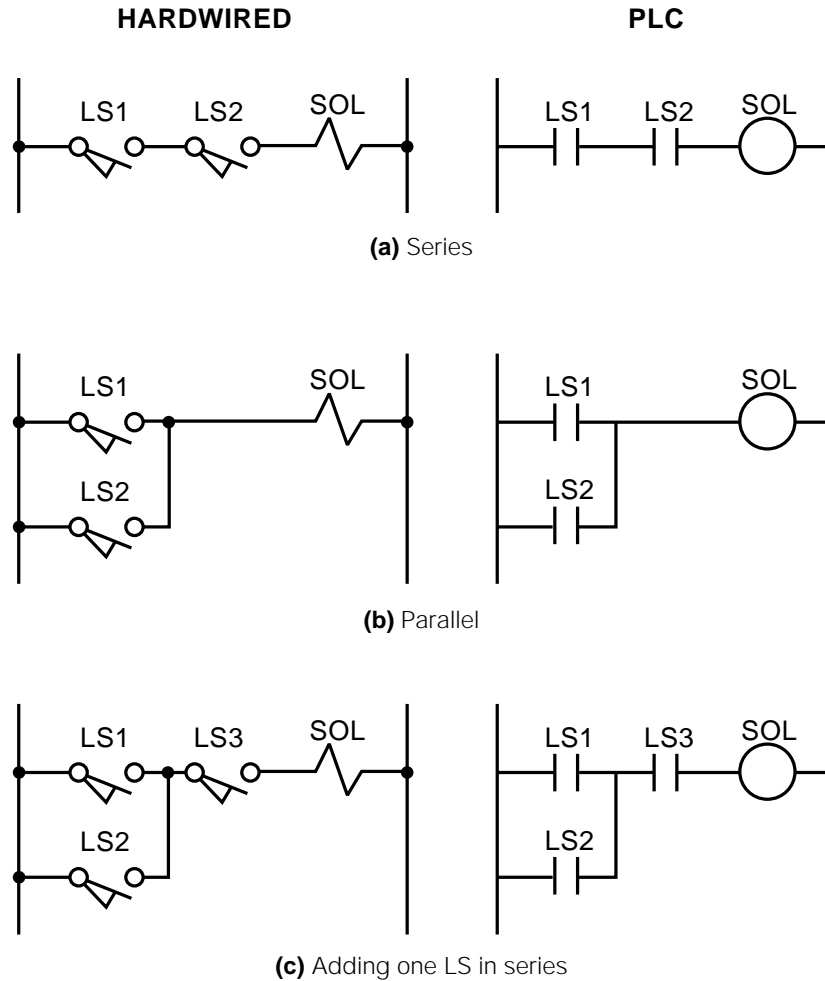
Figure 1-19. Example of hardwiring changes as opposed to softwiring changes.

EASE OF INSTALLATION

Several attributes make PLC installation an easy, cost-effective project. Its relatively small size allows a PLC to be conveniently located in less than half the space required by an equivalent relay control panel (see Figure 1-20). On a small-scale changeover from relays, a PLC's small, modular construction allows it to be mounted in the same enclosure where the relays were located. Actual changeover can be made quickly by simply connecting the input/output devices to the prewired terminal strips.
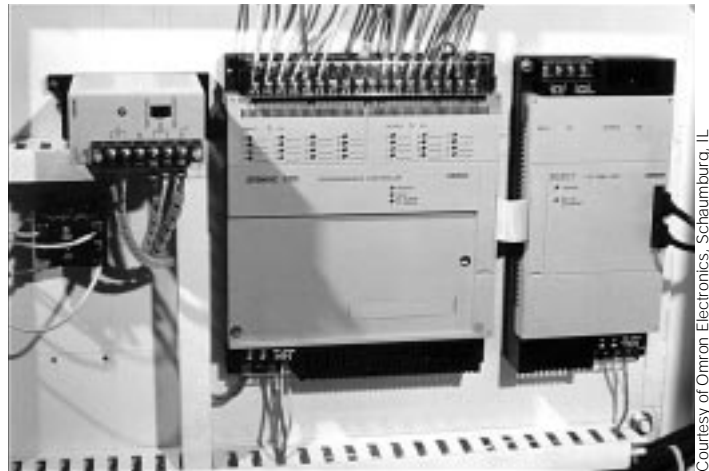
Figure 1-20. Space-efficient design of a PLC.

In large installations, remote input/output stations are placed at optimum locations (see Figure 1-21). A coaxial cable or a twisted pair of wires connects the remote station to the CPU. This configuration results in a considerable reduction in material and labor costs as compared to a hardwired system, which would involve running multiple wires and installing large conduits. The remote subsystem approach also means that various sections of a total system can be completely prewired by an OEM or PLC vendor prior to reaching the installation site. This approach considerably reduces the time spent by an electrician during an on-site installation.
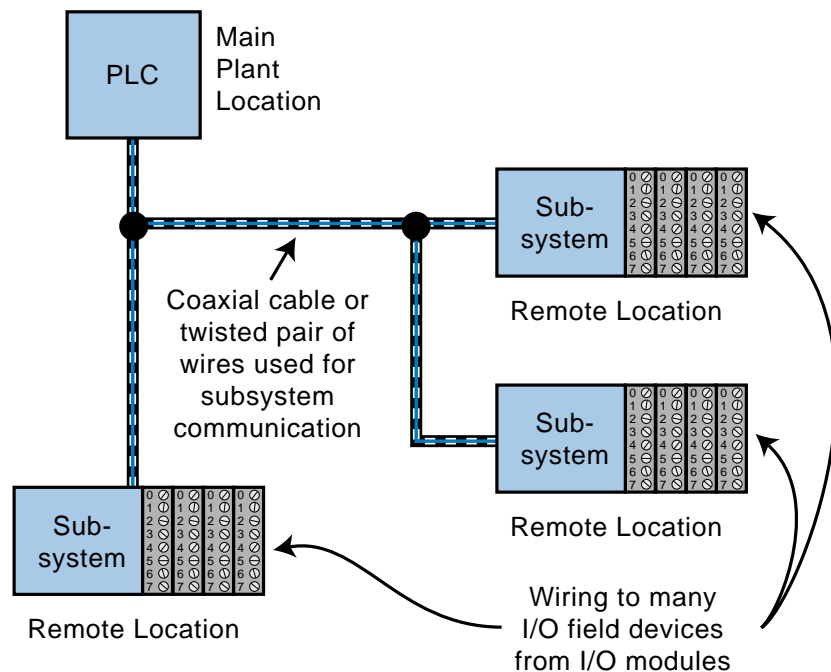
Figure 1-21. Remote I/O station installation.

EASE OF MAINTENANCE AND TROUBLESHOOTING

From the beginning, programmable controllers have been designed with ease of maintenance in mind. With virtually all components being solid-state, maintenance is reduced to the replacement of modular, plug-in components. Fault detection circuits and diagnostic indicators (see Figure 1-22), incorporated in each major component, signal whether the component is working properly or malfunctioning. In fact, most failures associated with a PLC-based system stem from failures directly related to the field input/output devices, rather than the PLC's CPU or I/O interface system (see Figure 1-23). However, the monitoring capability of a PLC system can easily detect and correct these field device failures.

(a)

(b)

Courtesy of Allen-Bradley, Highland Heights, OH
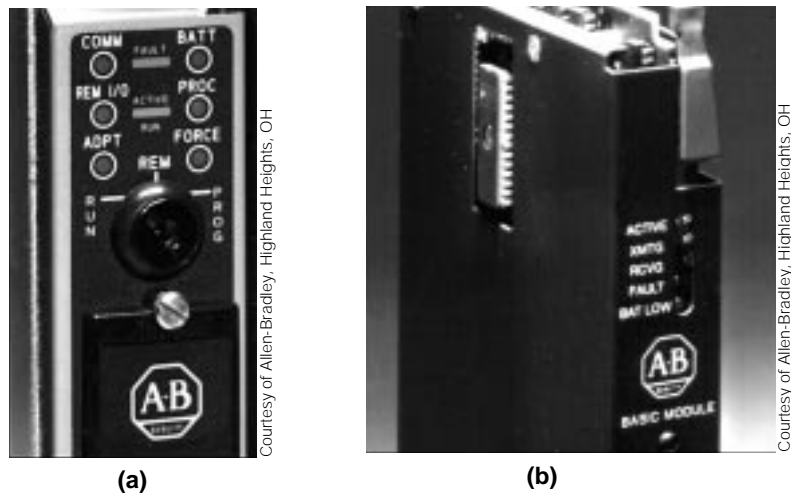
Figure 1-22. (a) A PLC processor and **(b)** an intelligent module containing several status indicators.

CPU
5%

I/O
10%

Field
device
failures
85%
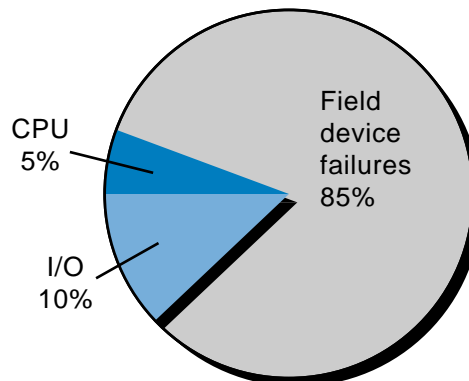
Figure 1-23. Failures in a PLC-based system.

SECTION
1
*Introductory
Concepts*

*Introduction to
Programmable Controllers*
CHAPTER
1

With the aid of the programming device, any programmed logic can be viewed to see if inputs or outputs are ON or OFF (see Figure 1-24). Programmed instructions can also be written to enunciate certain failures.
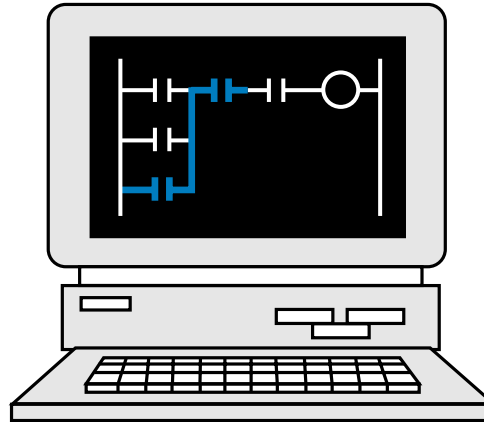
Figure 1-24. A programming device being used to monitor inputs and outputs, with highlighted contacts indicating an ON condition.

These and several other attributes of the PLC make it a valuable part of any control system. Once installed, its contribution will be quickly noticed and payback will be readily realized. The potential benefits of the PLC, like any intelligent device, will depend on the creativity with which it is applied.

It is obvious from the preceding discussion that the potential benefits of applying programmable controllers in an industrial application are substantial. The bottom line is that, through the use of programmable controllers, users will achieve high performance and reliability, resulting in higher quality at a reduced cost.

**KEY
TERMS**

**address
central processing unit (CPU)
execute
hardware
input/output system
interface
ladder diagram
programmable logic controller (PLC)
programming device
read
relay logic
scan
software
solid-state
write**

NUMBER SYSTEMS AND CODES

I have often admired the mystical ways of Pythagoras and the secret magic of numbers.

—Sir Thomas Browne

CHAPTER HIGHLIGHTS

In this chapter, we will explain the number systems and digital codes that are most often used in programmable controller applications. We will first introduce the four number systems most frequently used during input/output address assignment and programming: binary, octal, decimal, and hexadecimal. Then, we will discuss the binary coded decimal (BCD) and Gray codes, along with the ASCII character set and several PLC register formats. Since these codes and systems are the foundation of the logic behind PLCs, a basic knowledge of them will help you understand how PLCs work.

2-1 NUMBER SYSTEMS

A familiarity with number systems is quite useful when working with programmable controllers, since a basic function of these devices is to represent, store, and operate on numbers, even when performing the simplest of operations. In general, programmable controllers use binary numbers in one form or another to represent various codes and quantities. Although these number operations are transparent for the most part, there are occasions where a knowledge of number systems is helpful.

First, let's review some basics. The following statements apply to any number system:

- Every number system has a base or radix.

- Every system can be used for counting.

- Every system can be used to represent quantities or codes.

- Every system has a set of symbols.

The **base** of a number system determines the total number of unique symbols used by that system. The largest-valued symbol always has a value of one less than the base. Since the base defines the number of symbols, it is possible to have a number system of any base. However, number system bases are typically chosen for their convenience. The number systems usually encountered while using programmable controllers are base 2, base 8, base 10, and base 16. These systems are called binary, octal, decimal, and hexadecimal, respectively. To demonstrate the common characteristics of number systems, let's first turn to the familiar decimal system.

DECIMAL NUMBER SYSTEM

The **decimal number system**, which is the most common to us, was undoubtedly developed because humans have ten fingers and ten toes. Thus, the base of the decimal number system is 10. The symbols, or digits, used in this system are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. As noted earlier, the total number of symbols (10) is the same as the base, with the largest-valued symbol being

one less than the base (9 is one less than 10). Because the decimal system is so common, we rarely stop to think about how to express a number greater than 9, the largest-valued symbol. It is, however, important to note that the technique for representing a value greater than the largest symbol is the same for any number system.

In the decimal system, a *place value*, or *weight*, is assigned to each position that a number greater than 9 would hold, starting from right to left. The first position (see Figure 2-1), starting from the right-most position, is position 0, the second is position 1, and so on, up to the last position *n*. As shown in Figure 2-2, the **weighted value** of each position can be expressed as the base (10 in this case) raised to the power of *n* (the position). For the decimal system, then, the position weights from right to left are 1, 10, 100, 1000, etc. This method for computing the value of a number is known as the **sum-of-the-weights method**.

Position $\quad n. \ldots . 3 \quad 2 \quad 1 \quad 0$

Number $\longrightarrow$ Value $\quad V_n. \ldots V_3 \quad V_2 \quad V_1 \quad V_0$

Figure 2-1. Place values.

Position (*n*) $\qquad 3 \quad 2 \quad 1 \quad 0$

Value (*V*) $\qquad V_3 \quad V_2 \quad V_1 \quad V_0 \qquad$ Weight Value = Base ^{Position}
(Base = 10 for decimal)

$10^0 = 1$
$10^1 = 10$
$10^2 = 100$
$10^3 = 1000$
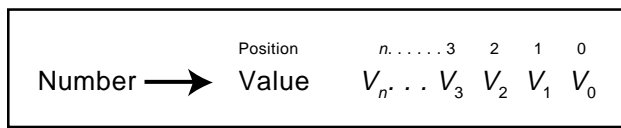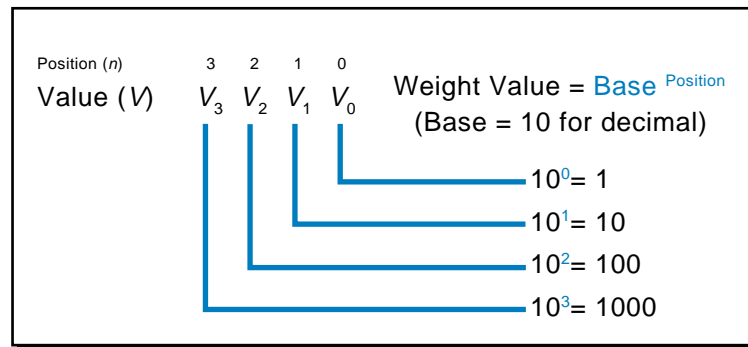
Figure 2-2. Weighted values.

The value of a decimal number is computed by multiplying each digit by the weighted value of its position and then summing the results. Let's take, for example, the number 9876. It can be expressed through the sum-of-the-weights method as:

Position $\quad 3 \quad 2 \quad 1 \quad 0$

Number $\quad$ 9 8 7 6

$6 \times 10^0 = \qquad 6$
$7 \times 10^1 = \qquad 70$
$8 \times 10^2 = \qquad 800$
$9 \times 10^3 = \qquad 9000$
$\overline{\qquad\qquad 9876_{10}}$
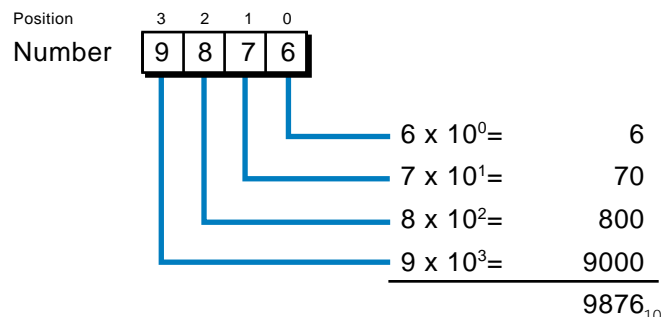
As you will see in other number systems, the decimal equivalent of any number can be computed by multiplying each digit by its base raised to the power of the digit's position. This is shown below:

Position n • • • 3 2 1 0

Number $\boxed{Z_n \mid \cdots \mid Z_3 \mid Z_2 \mid Z_1 \mid Z_0}$ Base = b

$Z_0 \times b^0 = N_0$
$Z_1 \times b^1 = N_1$
$Z_2 \times b^2 = N_2$
$Z_3 \times b^3 = N_3$

• • •
• • •
• • •
• • •

$Z_n \times b^n = N_n$
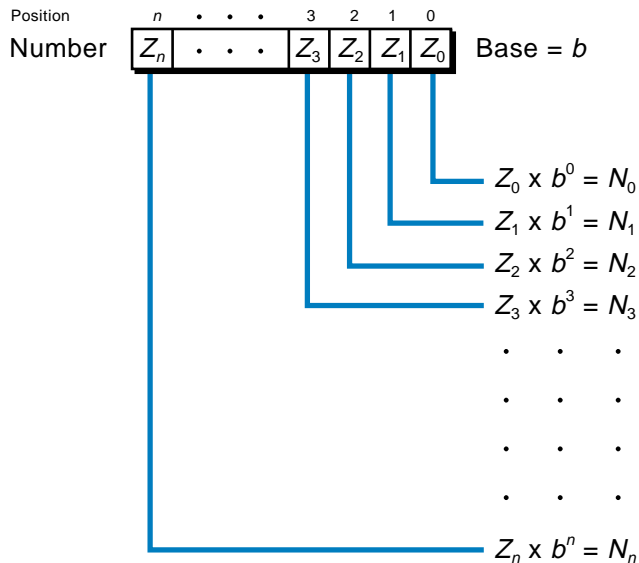
Therefore, the sum of N_0 through N_n will be the decimal equivalent of the number in base b.

BINARY NUMBER SYSTEM

The **binary number system** uses the number 2 as the base. Thus, the only allowable digits are 0 and 1; there are no 2s, 3s, etc. For devices such as programmable controllers and digital computers, the binary system is the most useful. It was adopted for convenience, since it is easier to design machines that distinguish between only two entities, or numbers (i.e., 0 and 1), rather than ten, as in decimal. Most physical elements have only two states: a light bulb is on or off, a valve is open or closed, a switch is on or off, and so on. In fact, you see this number system every time you use a computer—if you want to turn it on, you flip the switch to the 1 position; if you want to turn it off, you flip the switch to the 0 position (see Figure 2-3). Digital circuits can distinguish between two voltage levels (e.g., +5 V and 0 V), which makes the binary system very useful for digital applications.

Figure 2-3. The binary numbers, 1 and 0, on a computer's power switch represent ON and OFF, respectively.

As with the decimal system, expressing binary numbers greater than the largest-valued symbol (in this case 1) is accomplished by assigning a weighted value to each position from right to left. The weighted value (decimal equivalent) of a binary number is computed the same way as it is for a decimal number—only instead of being 10 raised to the power of the position, it is 2 raised to the power of the position. For binary, then, the weighted values from right to left are 1, 2, 4, 8, 16, 32, 64, etc., representing positions 0, 1, 2, 3, 4, 5, 6, etc. Let's calculate the decimal value that is equivalent to the value of the binary number 10110110:

Position   7 6 5 4 3 2 1 0

Number | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |$_2$

$$0 \times 2^0 = 0$$
$$1 \times 2^1 = 2$$
$$1 \times 2^2 = 4$$
$$0 \times 2^3 = 0$$
$$1 \times 2^4 = 16$$
$$1 \times 2^5 = 32$$
$$0 \times 2^6 = 0$$
$$1 \times 2^7 = 128$$
$$182_{10}$$
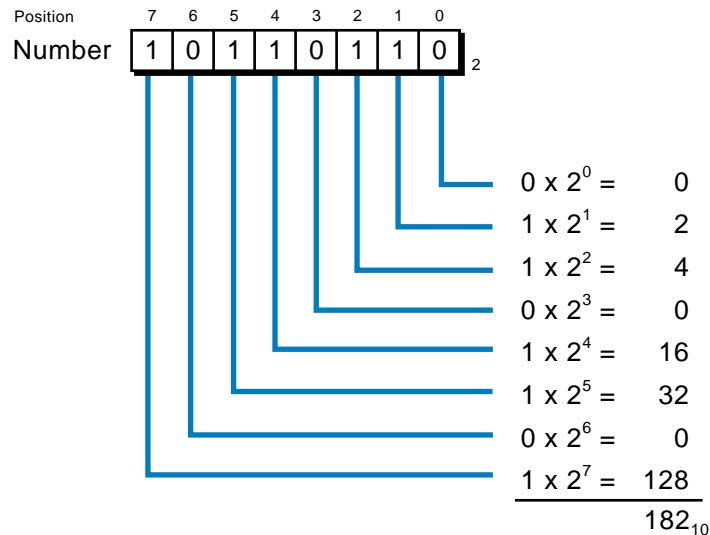
Thus, the binary number 10110110 is equivalent to the number 182 in the decimal system. Each digit of a binary number is known as a **bit**; hence, this particular binary number, 10110110 (182 decimal), has 8 bits. A group of 4 bits is known as a **nibble**; a group of 8 bits is a **byte**; and a group of one or more bytes is a **word**. Figure 2-4 presents a binary number composed of 16 bits, with the **least significant bit (LSB)**, the lowest valued bit in the word, and the **most significant bit (MSB)**, the largest valued bit in the word, identified.

Most Significant Bit (MSB)   Bit   Least Significant Bit (LSB)

| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |

|←—— Byte ——→|←—— Byte ——→|

|←——————— Word ———————→|
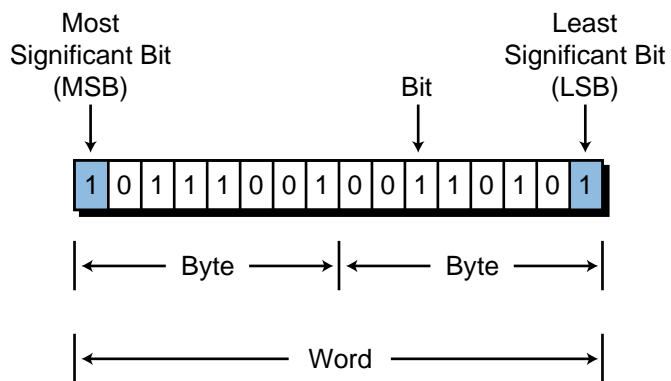
Figure 2-4. One word, two bytes, sixteen bits.

Counting in binary is a little more awkward than counting in decimal for the simple reason that we are not used to it. Because the binary number system uses only two digits, we can only count from 0 to 1—only one change in one digit location (OFF to ON) before a new digit position must be added. Conversely, in the decimal system, we can count from 0 to 9, equaling ten digit transitions, before a new digit position is added.

In binary, just like in decimal, we add another digit position once we run out of transitions. So, when we count in binary, the digit following 0 and 1 is 10 (one-zero, not ten), just like when we count 0, 1, 2…9 in decimal, another digit position is added and the next digit is 10 (ten). Table 2-1 shows a count in binary from 0_{10} to 15_{10}.

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Table 2-1. Decimal and binary counting.

OCTAL NUMBER SYSTEM

Writing a number in binary requires substantially more digits than writing it in decimal. For example, 91_{10} equals 1011011_2. Too many binary digits can be cumbersome to read and write, especially for humans. Therefore, the **octal numbering system** is often used to represent binary numbers using fewer digits. The octal number system uses the number 8 as its base, with its eight digits being 0, 1, 2, 3, 4, 5, 6, and 7. Table 2-2 shows both an octal and a binary count representation of the numbers 0 through 15 (decimal).

Decimal	Binary	Octal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	10
9	1001	11
10	1010	12
11	1011	13
12	1100	14
13	1101	15
14	1110	16
15	1111	17

Table 2-2. Decimal, binary, and octal counting.

Like all other number systems, each digit in an octal number has a weighted decimal value according to its position. For example, the octal number 1767 is equivalent to the decimal number 1015:

Position 3 2 1 0

Number $\boxed{1\ 7\ 6\ 7}_8$

$7 \times 8^0 = \qquad 7$
$6 \times 8^1 = \qquad 48$
$7 \times 8^2 = \qquad 448$
$1 \times 8^3 = \qquad 512$
$\overline{\qquad\qquad 1015_{10}}$
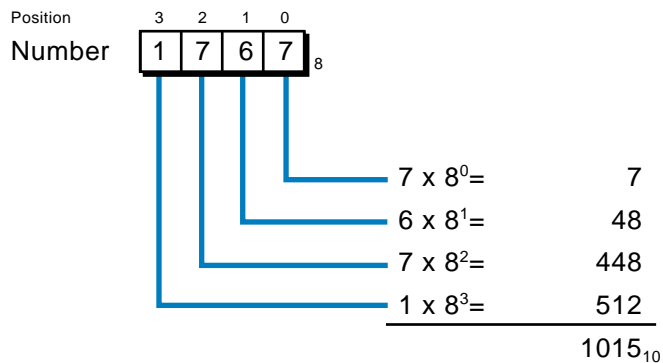
As noted earlier, the octal numbering system is used as a convenient way of writing a binary number. The octal system has a base of 8 (2^3), making it possible to represent any binary number in octal by grouping binary bits in groups of three. In this manner, a very large binary number can be easily represented by an octal number with significantly fewer digits. For example:

| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | Binary Number |

| 1 1 1 | 0 0 0 | 1 1 1 | 1 0 1 | 0 1 1 | 3-Bit Groups |

| 1 | 6 | 1 | 7 | 5 | 3 | Octal Digits |
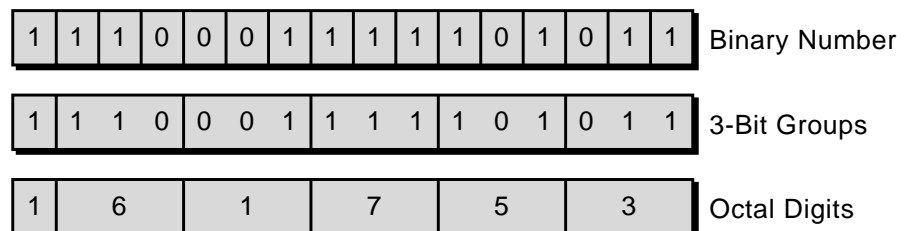
So, a 16-bit binary number can be represented directly by six digits in octal. As you will see later, many programmable controllers use the octal number system for referencing input/output and memory addresses.

HEXADECIMAL NUMBER SYSTEM

The **hexadecimal (hex) number system** uses 16 as its base. It consists of 16 digits—the numbers 0 through 9 and the letters *A* through *F* (which represent the numbers 10 through 15, respectively). The hexadecimal system is used for the same reason as the octal system, to express binary numbers using fewer digits. The hexadecimal numbering system uses one digit to represent four binary digits (or bits), instead of three as in the octal system. Table 2-3 shows a hexadecimal count example of the numbers 0 through 15 with their decimal and binary equivalents.

Binary	Decimal	Hexadecimal
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Table 2-3. Binary, decimal, and hexadecimal counting.

As with the other number systems, hexadecimal numbers can be represented by their decimal equivalents using the sum-of-the-weights method. The decimal values of the letter-represented hex digits A through F are used when computing the decimal equivalent (10 for A, 11 for B, and so on). The following example uses the sum-of-the-weights method to transform the hexadecimal number F1A6 into its decimal equivalent. The value of A in the example is 10 times 16^1, while F is 15 times 16^3. Thus, the hexadecimal number F1A6 is equivalent to the decimal number 61,862:

Position   3   2   1   0

Number | F | 1 | A | 6 |$_{16}$

$6 \times 16^0 =$ 6
$10 \times 16^1 =$ 160
$1 \times 16^2 =$ 256
$15 \times 16^3 =$ 61440
                     61862_{10}
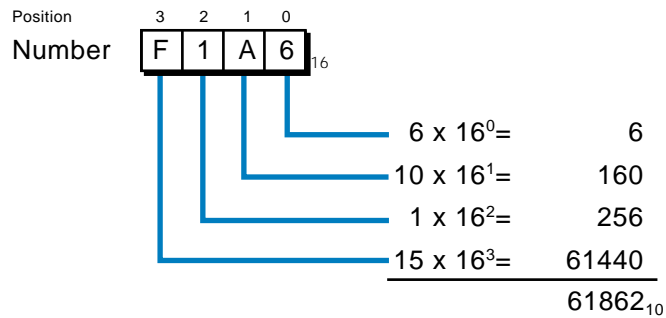
Like octal numbers, hexadecimal numbers can easily be converted to binary without any mathematical transformation. To convert a hexadecimal number to binary, simply write the 4-bit binary equivalent of the hex digit for each position. For example:

| F | 1 | A | 6 |

| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
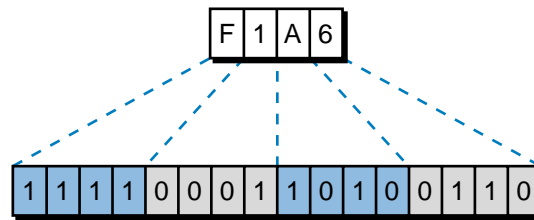
2-2 NUMBER CONVERSIONS

In the previous section, you saw how a number of any base can be converted to the familiar decimal system using the sum-of-the-weights method. In this section, we will show you how a decimal number can be converted to binary, octal, or any number system.

To convert a decimal number to its equivalent in any base, you must perform a series of divisions by the desired base. The conversion process starts by dividing the decimal number by the base. If there is a remainder, it is placed in the **least significant digit** (right-most) position of the new base number. If there is no remainder, a 0 is placed in least significant digit position. The result of the division is then brought down, and the process is repeated until the final result of the successive divisions is 0. This methodology may be a little cumbersome; however, it is the easiest conversion method to understand and employ.

As a generic example, let's find the base 5 equivalent of the number Z (see Figure 2-5). The first division ($Z \div 5$) gives an N_1 result and a remainder R_1. The remainder R_1 becomes the first digit of the base 5 number (the least significant digit). To obtain the next base 5 digit, the N_1 result is again divided

by 5, giving an N_2 result and an R_2 remainder that becomes the second base 5 digit. This process is repeated until the result of the division ($N_n \div 5$) is 0, giving the last remainder R_n, which becomes the **most significant digit** (leftmost digit) of the base 5 number.

Division	Remainder
$Z \div 5 = N_1$	R_1
$N_1 \div 5 = N_2$	R_2
$N_2 \div 5 = N_3$	R_3
$N_3 \div 5 = N_4$	R_4
•	
•	
•	
$N_n \div 5 = 0$	R_n
New base 5 number is $(R_n... R_4 R_3 R_2 R_1)_5$	

Figure 2-5. Method for converting a decimal number into any base.

Now, let's convert the decimal number 35_{10} to its binary (base 2) equivalent using this method:

Division	Remainder
$35 \div 2 = 17$	1
$17 \div 2 = 8$	1
$8 \div 2 = 4$	0
$4 \div 2 = 2$	0
$2 \div 2 = 1$	0
$1 \div 2 = 0$	1

Therefore, the base 2 (binary) equivalent of the decimal number 35 is 100011.

As another exercise, let's convert the number 1355_{10} to its hexadecimal (base 16) equivalent:

Division	Remainder
1355 ÷ 16 = 84	11
84 ÷ 16 = 5	4
5 ÷ 16 = 0	5

Thus, the hexadecimal equivalent of 1355_{10} is $54B_{hex}$ (remember that the hexadecimal system uses the letter *B* to represent the number 11).

There is another method, which is a little faster, for computing the binary equivalent of a decimal number. This method employs division by eight, instead of by two, to convert the number first to octal and then to binary from octal (three bits at a time).

For instance, let's take the number 145_{10}:

Division	Remainder
145 ÷ 8 = 18	1
18 ÷ 8 = 2	2
2 ÷ 8 = 0	2

The octal equivalent of 145_{10} is 221_8, so from Table 2-2, we can find that 221_8 equals 010010001 binary:
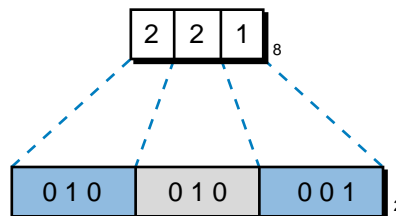
2-3 ONE'S AND TWO'S COMPLEMENT

The one's and two's complements of a binary number are operations used by programmable controllers, as well as computers, to perform internal mathematical calculations. To *complement* a binary number means to change it to a negative number. This allows the basic arithmetic operations of

subtraction, multiplication, and division to be performed through successive addition. For example, to subtract the number 20 from the number 40, first complement 20 to obtain –20, and then perform an addition.

The intention of this section is to introduce the basic concepts of complementing, rather than to provide a thorough analysis of arithmetic operations. For more information on this subject, please use the references listed in the back of this book.

ONE'S COMPLEMENT

Let's assume that we have a 5-bit binary number that we wish to represent as a negative number. The number is decimal 23, or binary:

$$10111_2$$

There are two ways to represent this number as a negative number. The first method is to simply place a minus sign in front of the number, as we do with decimal numbers:

$$-(10111)_2$$

This method is suitable for us, but it is impossible for programmable controllers and computers to interpret, since the only symbols they use are binary 1s and 0s. To represent negative numbers, then, some digital computing devices use what is known as the **one's complement** method. First, the one's complement method places an extra bit (sign bit) in the most significant (left-most) position and lets this bit determine whether the number is positive or negative. The number is positive if the sign bit is 0 and negative if the sign bit is 1. Using the one's complement method, +23 decimal is represented in binary as shown here with the sign bit (0) indicated in bold:

$$\mathbf{0}\ 10111_2$$

The negative representation of binary 10111 is obtained by placing a 1 in the most significant bit position and inverting each bit in the number (changing 1s to 0s and 0s to 1s). So, the one's complement of binary 10111 is:

$$\mathbf{1}\ 01000_2$$

If a negative number is given in binary, its one's complement is obtained in the same fashion.

$$-15_{10} = \mathbf{1}\ 0000_2$$

$$+15_{10} = \mathbf{0}\ 1111_2$$

TWO'S COMPLEMENT

The **two's complement** is similar to the one's complement in the sense that one extra digit is used to represent the sign. The two's complement computation, however, is slightly different. In the one's complement, all bits are inverted; but in the two's complement, each bit, from right to left, is inverted only after the first 1 is detected. Let's use the number +22 decimal as an example:

$$+22_{10} = \mathbf{0}\ 10110_2$$

Its two's complement would be:

$$-22_{10} = \mathbf{1}\ 01010_2$$

Note that in the negative representation of the number 22, starting from the right, the first digit is a 0, so it is not inverted; the second digit is a 1, so all digits after this one are inverted.

If a negative number is given in two's complement, its complement (a positive number) is found in the same fashion:

$$-14_{10} = \mathbf{1}\ 10010_2$$

$$+14_{10} = \mathbf{0}\ 01110_2$$

Again, all bits from right to left are inverted after the first 1 is detected. Other examples of the two's complement are shown here:

$$+17_{10} = \mathbf{0}\ 10001_2$$

$$-17_{10} = \mathbf{1}\ 01111_2$$

$$+7_{10} = \mathbf{0}\ 00111_2$$

$$-7_{10} = \mathbf{1}\ 11001_2$$

$$+1_{10} = \mathbf{0}\ 00001_2$$

$$-1_{10} = \mathbf{1}\ 11111_2$$

The two's complement of 0 does not really exist, since no first 1 is ever encountered in the number. The two's complement of 0, then, is 0.

The two's complement is the most common arithmetic method used in computers, as well as programmable controllers.

2-4 BINARY CODES

An important requirement of programmable controllers is communication with various external devices that either supply information to the controller or receive information from the controller. This input/output function involves the transmission, manipulation, and storage of binary data that, at some point, must be interpreted by humans. Although machines can easily handle this binary data, we require that the data be converted to a more interpretable form.

One way of satisfying this requirement is to assign a unique combination of 1s and 0s to each number, letter, or symbol that must be represented. This technique is called *binary coding*. In general, there are two categories of codes—those that represent numbers only and those that represent letters, symbols, and decimal numbers.

Several codes for representing numbers, symbols, and letters are standard throughout the industry. Among the most common are the following:

- ASCII

- BCD

- Gray

ASCII

Alphanumeric codes (which use a combination of letters, symbols, and decimal numbers) are used when information processing equipment, such as printers and cathode ray tubes (CRTs), must process the alphabet along with numbers and special symbols. These alphanumeric characters—26 letters (uppercase), 10 numerals (0-9), plus mathematical and punctuation symbols—can be represented using a 6-bit code (i.e., $2^6 = 64$ possible characters). The most common code for alphanumeric representation is **ASCII** (the American Standard Code for Information Interchange).

An ASCII (pronounced as-kee) code can be 6, 7, or 8 bits. Although a 6-bit code (64 possible characters) can accommodate the basic alphabet, numbers, and special symbols, standard ASCII character sets use a 7-bit code ($2^7 = 128$ possible characters), which provides room for lower case and control characters, in addition to the characters already mentioned. This 7-bit code provides all possible combinations of characters used when communicating with peripherals and interfaces.

An 8-bit ASCII code is used when parity check (see Chapter 4) is added to a standard 7-bit code for error-checking purposes (note that all eight bits can still fit in one byte). Figure 2-6a shows the binary ASCII code representation of the letter Z (132_8). This letter is generally sent and received in serial form between the PLC and other equipment.

Figure 2-6b illustrates a typical ASCII transmission, again using the character Z as an example. Note that extra bits have been added to the beginning and end of the character to signify the start and stop of the ASCII transmission. Appendix B shows a standard ASCII table, while Chapter 8 further explains serial communication.
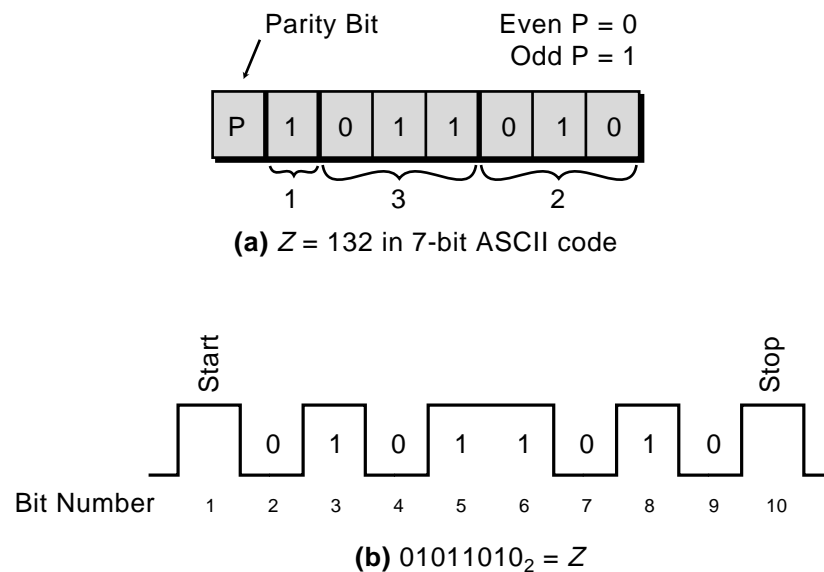
Figure 2-6. (a) ASCII representation of the character Z and **(b)** the ASCII transmission of the character Z.

BCD

The **binary coded decimal (BCD)** system was introduced as a convenient way for humans to (1) handle numbers that must be input to digital machines and (2) interpret numbers that are output from machines. The best solution to this problem was to convert a code readily handled by man (decimal) to a code readily handled by processing equipment (binary). The result was BCD.

The decimal system uses the numbers 0 through 9 as its digits, whereas BCD represents each of these numbers as a 4-bit binary number. Table 2-4 illustrates the relationship between the BCD code and the binary and decimal number systems.

Decimal	Binary	BCD
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001

Table 2-4. Decimal, binary, and BCD counting.

The BCD representation of a decimal number is obtained by replacing each decimal digit with its BCD equivalent. The BCD representation of decimal 7493 is shown here as an example:

$$\begin{array}{ccccc} \text{BCD} & \rightarrow & 0111 & 0100 & 1001 & 0011 \\ \text{Decimal} & \rightarrow & 7 & 4 & 9 & 3 \end{array}$$

Typical PLC applications of BCD codes include data entry (time, volume, weight, etc.) via thumbwheel switches (TWS), data display via seven-segment displays, input from absolute encoders, and analog input/output instructions. Figure 2-7 shows a thumbwheel switch and a seven-segment indicator field device.

Figure 2-7. (a) A seven-segment indicator field device and **(b)** a thumbwheel switch.

Nowadays, the circuitry necessary to convert from decimal to BCD and from BCD to seven-segment is already built into thumbwheel switches and seven-segment LED devices (see Figures 2-8a and 2-8b). This BCD data is converted internally by the PLC into the binary equivalent of the input data. Input and output of BCD data requires four lines of an input/output interface for each decimal digit.
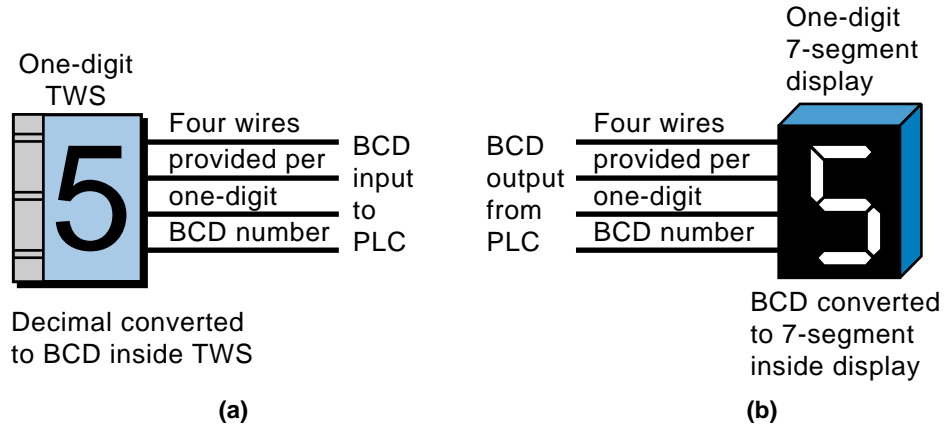
Figure 2-8. (a) Thumbwheel switch converts decimal numbers into BCD inputs for the PLC. **(b)** The seven-segment display converts the BCD outputs from the PLC into a decimal number.

GRAY

The **Gray code** is one of a series of cyclic codes known as *reflected codes* and is suited primarily for position transducers. It is basically a binary code that has been modified in such a way that only one bit changes as the counting number increases. In standard binary, as many as four digits can change when counting with as few as four binary digits. This drastic change is seen in the transition from binary 7 to 8. Such a change allows a great chance for error, which is unsuitable for positioning applications. Thus, most encoders use the Gray code to determine angular position. Table 2-5 shows this code with its binary and decimal equivalents for comparison.

Gray Code	Binary	Decimal
0000	0	0
0001	1	1
0011	10	2
0010	11	3
0110	100	4
0111	101	5
0101	110	6
0100	111	7
1100	1000	8
1101	1001	9
1111	1010	10
1110	1011	11
1010	1100	12
1011	1101	13
1001	1110	14
1000	1111	15

Table 2-5. Gray code, binary, and decimal counting.

An example of a Gray code application is an optical absolute encoder. In this encoder, the rotor disk consists of opaque and transparent segments arranged in a Gray code pattern and illuminated by a light source that shines through the transparent sections of the rotating disk. The transmitted light is received at the other end in Gray code form and is available for input to the PLC in either Gray code or BCD code, if converted. Figure 2-9 illustrates a typical absolute encoder and its output.
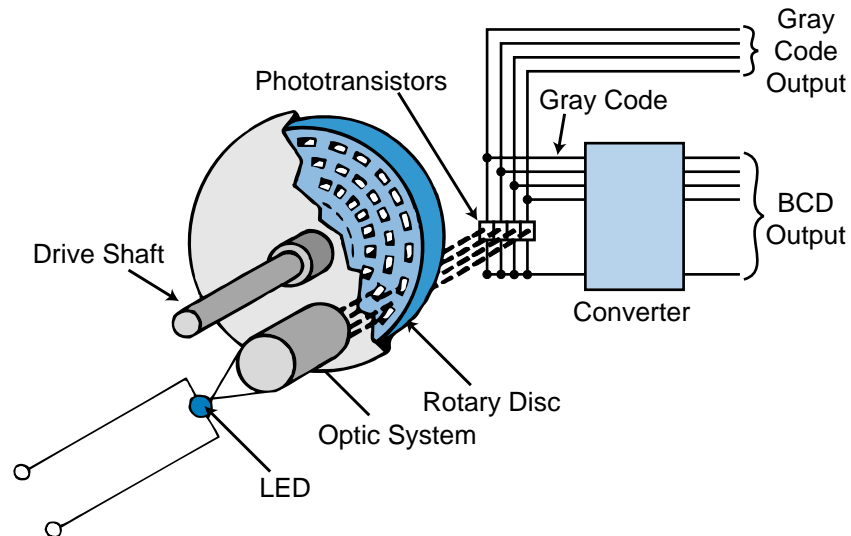
Figure 2-9. An absolute encoder with BCD and Gray outputs.

2-5 REGISTER WORD FORMATS

As previously mentioned, a programmable controller performs all of its internal operations in binary format using 1s and 0s. In addition, the status of I/O field devices is also read and written, in binary form, to and from the PLC's CPU. Generally, these operations are performed using a group of 16 bits that represent numbers and codes. Recall that the grouping of bits with which a particular machine operates is called a word. A PLC word is also called a **register** or *location*. Figure 2-10 illustrates a 16-bit register composed of a two-byte word.

Figure 2-10. A 16-bit register/word.

Although the data stored in a register is represented by binary 1s and 0s, the format in which this binary data is stored may differ from one programmable controller to another. Generally, data is represented in either straight (noncoded) binary or binary coded decimal (BCD) format. Let's examine these two formats.
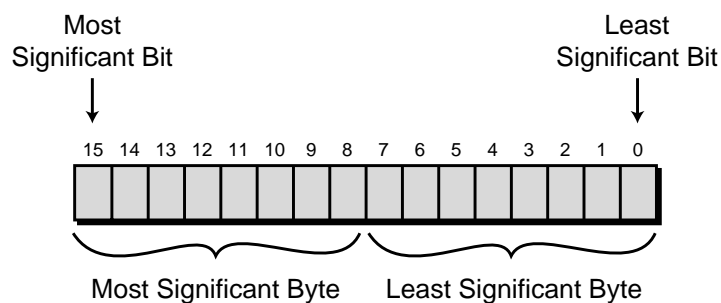
BINARY FORMAT

Data stored in binary format can be directly converted to its decimal equivalent without any special restrictions. In this format, a 16-bit register can represent a maximum value of 65535_{10}. Figure 2-11 shows the value 65535_{10} in binary format (all bits are 1). The binary format represents the status of a device as either 0 or 1, which is interpreted by the programmable controller as ON or OFF. All of these statuses are stored in registers or words.
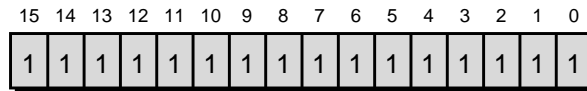
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2-11. A 16-bit register containing the binary equivalent of 65535_{10}.

If the most significant bit of the register in Figure 2-12 is used as a sign bit, then the maximum decimal value that the 16-bit register can store is $+32767_{10}$ or -32767_{10}.
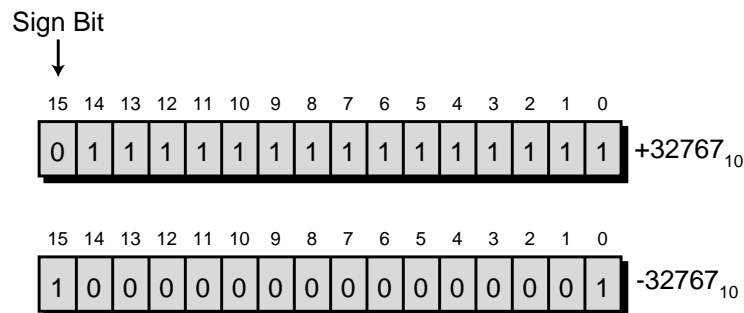
Sign Bit

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

$+32767_{10}$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

-32767_{10}

Figure 2-12. Two 16-bit registers with sign bits (MSB).

The decimal equivalents of these binary representations can be calculated using the sum-of-the-weights method. The negative representation of 32767_{10}, as shown in Figure 2-12, was derived using the two's complement method. As an exercise, practice computing these numbers (refer to Section 2-3 for help).

BCD FORMAT

The BCD format uses four bits to represent a single decimal digit. The only decimal numbers that these four bits can represent are 0 through 9. Some PLCs operate and store data in several of their software instructions, such as arithmetic and data manipulations, using the BCD format.

In BCD format, a 16-bit register can hold up to a 4-digit decimal value, with the decimal values that can be represented ranging from 0000–9999. Figure 2-13 shows a register containing the binary representation of BCD 9999.
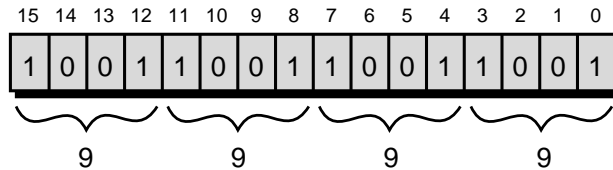

Figure 2-13. Register containing BCD 9999.

In a PLC, the BCD values stored in a register or word can be the result of BCD data input from a thumbwheel switch. A 4-digit thumbwheel switch will use a 16-bit register to store the BCD output data obtained during the read section of the scan (see Figure 2-14).
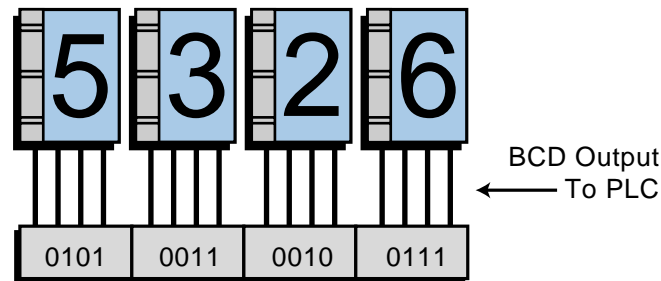
Figure 2-14. A 4-digit TWS using a 16-bit register to store BCD values.

EXAMPLE 2-1

Illustrate how a PLC's 16-bit register containing the BCD number 7815 would connect to a 4-digit, seven-segment display. Indicate the most significant digit and the least significant digit of the seven-segment display.

SOLUTION

Figure 2-15 illustrates the connection between a 16-bit register and a 4-digit, seven-segment display. The BCD output from the PLC register or word is sent to the seven-segment indicator through an output interface during the write, or update, section of the scan.
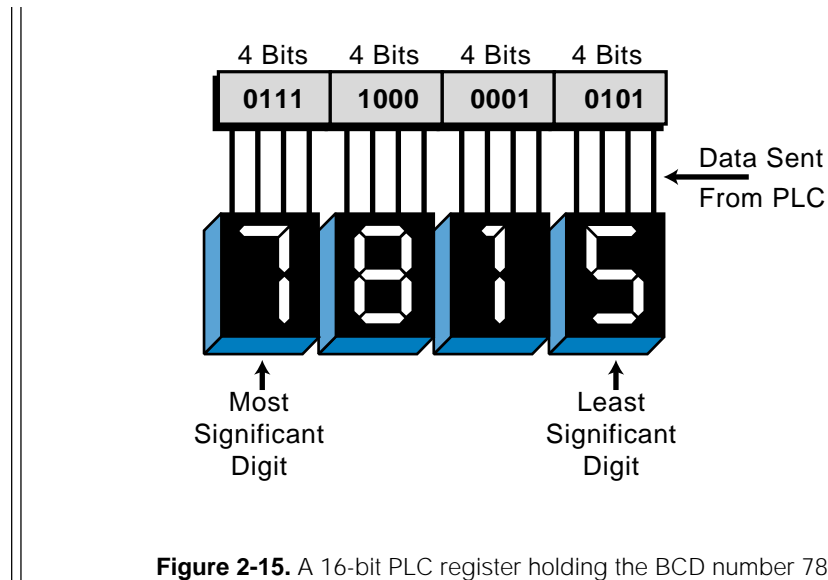
Figure 2-15. A 16-bit PLC register holding the BCD number 7815.

KEY TERMS

alphanumeric code
ASCII
base
binary coded decimal (BCD)
bit
byte
decimal number system
Gray code
hexadecimal number system
least significant bit (LSB)
least significant digit
most significant bit (MSB)
most significant digit
nibble
octal number system
one's complement
register
sum-of-the-weights method
two's complement
weighted value
word

This page intentionally left blank.

CHAPTER
THREE

LOGIC CONCEPTS

Science when well digested is nothing but good sense and reason.

—Leszinski Stanislas

CHAPTER HIGHLIGHTS

To understand programmable controllers and their applications, you must first understand the logic concepts behind them. In this chapter, we will discuss three basic logic functions—AND, OR, and NOT—and show you how, with just these three functions, you can make control decisions ranging from very simple to very complex. We will also introduce you to the fundamentals of Boolean algebra and its associated operators. Finally, we will explain the relationship between Boolean algebra and logic contact symbology, so that you will be ready to learn about PLC processors and their programming devices.

3-1 THE BINARY CONCEPT

The binary concept is not a new idea; in fact, it is a very old one. It simply refers to the idea that many things exist only in two predetermined states. For instance, a light can be on or off, a switch open or closed, or a motor running or stopped. In digital systems, these two-state conditions can be thought of as signals that are present or not present, activated or not activated, high or low, on or off, etc. This two-state concept can be the basis for making decisions; and since it is very adaptable to the binary number system, it is a fundamental building block for programmable controllers and digital computers.

Here, and throughout this book, binary 1 represents the presence of a signal (or the occurrence of some event), while binary 0 represents the absence of the signal (or the nonoccurrence of the event). In digital systems, these two states are actually represented by two distinct voltage levels, +V and 0V, as shown in Table 3-1. One voltage is more positive (or at a higher reference) than the other. Often, binary 1 (or logic 1) is referred to as TRUE, ON, or HIGH, while binary 0 (or logic 0) is referred to as FALSE, OFF, or LOW.

1 (+V)	0 (0V)	Example
Operating	Not operating	Limit switch
Ringing	Not ringing	Bell
On	Off	Light bulb
Blowing	Silent	Horn
Running	Stopped	Motor
Engaged	Disengaged	Clutch
Closed	Open	Valve

Table 3-1. Binary concept using positive logic.

Note that in Table 3-1, the more positive voltage (represented as logic 1) and the less positive voltage (represented as logic 0) were arbitrarily chosen. The use of binary logic to represent the more positive voltage level, meaning the occurrence of some event, as 1 is referred to as **positive logic**.

Negative logic, as illustrated in Table 3-2, uses 0 to represent the more positive voltage level, or the occurrence of the event. Consequently, 1 represents the nonoccurrence of the event, or the less positive voltage level. Although positive logic is the more conventional of the two, negative logic is sometimes more convenient in an application.

1 (+V)	0 (0V)	Example
Not operating	Operating	Limit switch
Not ringing	Ringing	Bell
Off	On	Light bulb
Silent	Blowing	Horn
Stopped	Running	Motor
Disengaged	Engaged	Clutch
Open	Closed	Valve

Table 3-2. Binary concept using negative logic.

3-2 LOGIC FUNCTIONS

The binary concept shows how physical quantities (binary variables) that can exist in one of two states can be represented as 1 or 0. Now, you will see how statements that combine two or more of these binary variables can result in either a TRUE or FALSE condition, represented by 1 and 0, respectively. Programmable controllers make decisions based on the results of these kinds of logical statements.

Operations performed by digital equipment, such as programmable controllers, are based on three fundamental logic functions—AND, OR, and NOT. These functions combine binary variables to form statements. Each function has a rule that determines the statement outcome (TRUE or FALSE) and a symbol that represents it. For the purpose of this discussion, the result of a statement is called an output (Y), and the conditions of the statement are called inputs (A and B). Both the inputs and outputs represent two-state variables, such as those discussed earlier in this section.

THE AND FUNCTION

Figure 3-1 shows a symbol called an AND **gate**, which is used to graphically represent the **AND** function. The AND output is TRUE (1) only if all inputs are TRUE (1).
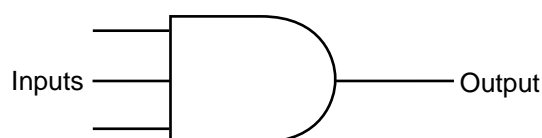

Figure 3-1. Symbol for the AND function.

SECTION
1

Introductory
Concepts

Logic
Concepts

CHAPTER
3

An AND function can have an unlimited number of inputs, but it can have only one output. Figure 3-2 shows a two-input AND gate and its resulting output *Y*, based on all possible input combinations. The letters *A* and *B* represent inputs to the controller. This mapping of outputs according to predefined inputs is called a **truth table**. Example 3-1 shows an application of the AND function.
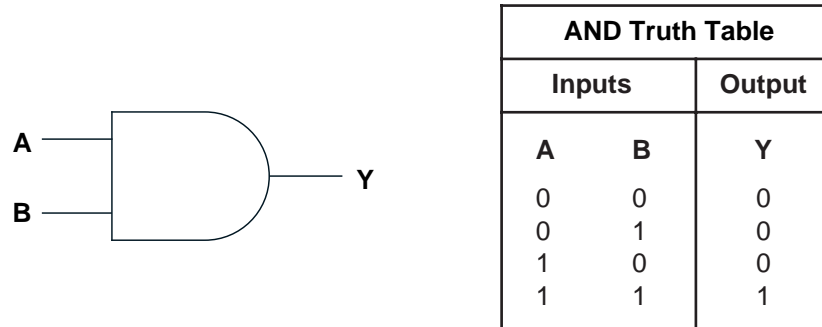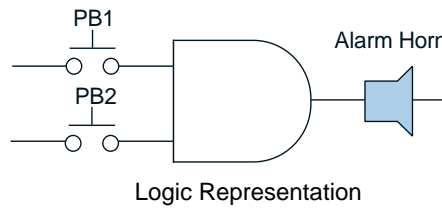

AND Truth Table		
Inputs		**Output**
A	**B**	**Y**
0	0	0
0	1	0
1	0	0
1	1	1

Figure 3-2. Two-input AND gate and its truth table.
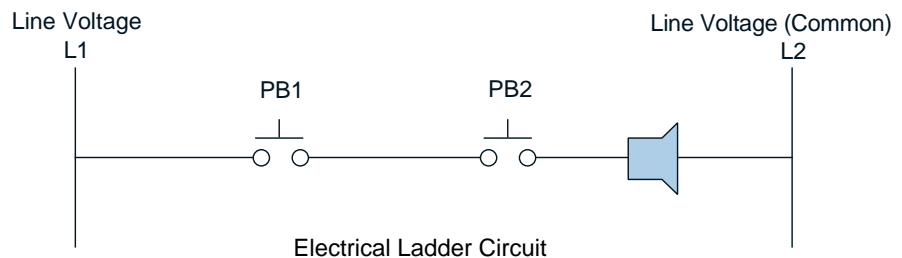
EXAMPLE 3-1

Show the logic gate, truth table, and circuit representations for an alarm horn that will sound if its two inputs, push buttons PB1 and PB2, are 1 (ON or depressed) at the same time.

SOLUTION

Logic Representation

PB1	PB2	Alarm Horn
Not pushed (0)	Not pushed (0)	Silent (0)
Not pushed (0)	Pushed (1)	Silent (0)
Pushed (1)	Not pushed (0)	Silent (0)
Pushed (1)	Pushed (1)	Sounding (1)

Electrical Ladder Circuit

Electrical Circuit

THE OR FUNCTION

Figure 3-3 shows the OR gate symbol used to graphically represent the **OR** function. The OR output is TRUE (1) if one or more inputs are TRUE (1).

Figure 3-3. Symbol for the OR function.

As with the AND function, an OR gate function can have an unlimited number of inputs but only one output. Figure 3-4 shows an OR function truth table and the resulting output Y, based on all possible input combinations. Example 3-2 shows an application of the OR function.

OR Truth Table		
Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Figure 3-4. Two-input OR gate and its truth table.
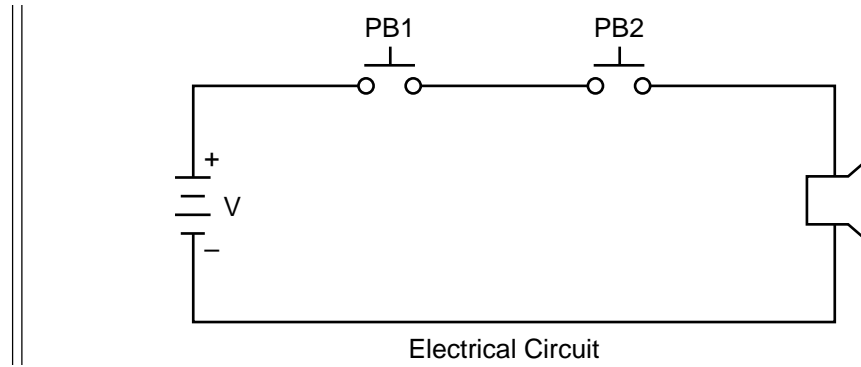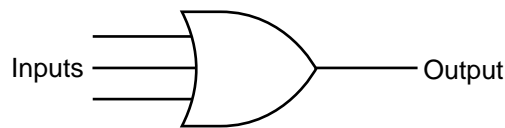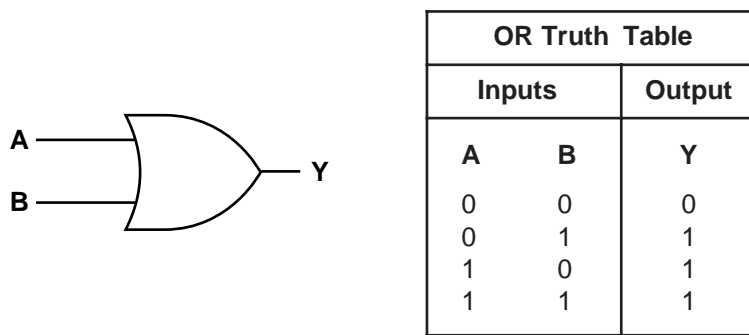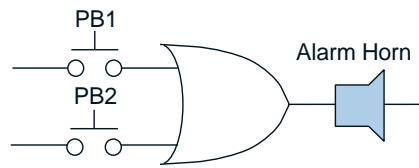
EXAMPLE 3-2

Show the logic gate, truth table, and circuit representations for an alarm horn that will sound if either of its inputs, push button PB1 or PB2, is 1 (ON or depressed).

SECTION
1

*Introductory
Concepts*

*Logic
Concepts*

CHAPTER
3

SOLUTION

Logic Representation

PB1	PB2	Alarm Horn
Not pushed (0)	Not pushed (0)	Silent (0)
Not pushed (0)	Pushed (1)	Sounding (1)
Pushed (1)	Not pushed (0)	Sounding (1)
Pushed (1)	Pushed (1)	Sounding (1)
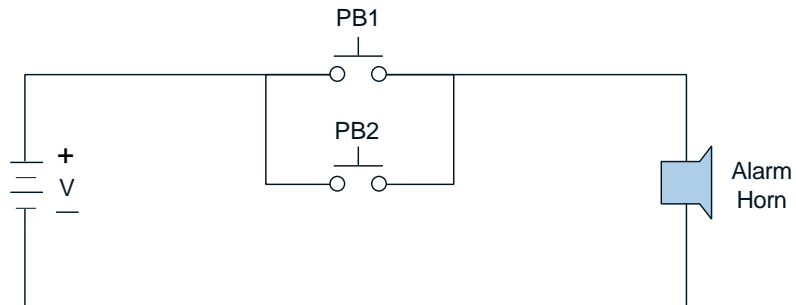
Electrical Circuit
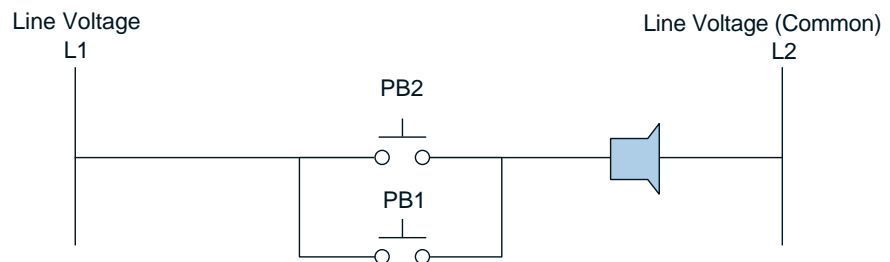
Electrical Ladder Circuit

THE NOT FUNCTION

Figure 3-5 illustrates the NOT symbol, which is used to graphically represent
the **NOT** function. The NOT output is TRUE (1) if the input is FALSE (0).
Conversely, if the output is FALSE (0), the input is TRUE (1). The result of
the NOT operation is always the inverse of the input; therefore, it is
sometimes called an *inverter*.

The NOT function, unlike the AND and OR functions, can have only one input. It is seldom used alone, but rather in conjunction with an AND or an OR gate. Figure 3-6 shows the NOT operation and its truth table. Note that an *A* with a bar on top represents NOT *A*.

Figure 3-5. Symbol for the NOT function.
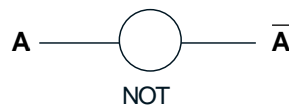
NOT Truth Table	
Input	**Output**
A	**$\overline{A}$**
0	1
1	0

Figure 3-6. NOT gate and its truth table.

At first glance, it is not as easy to visualize the application of the NOT function as it is the AND and OR functions. However, a closer examination of the NOT function shows it to be simple and quite useful. At this point, it is helpful to recall three points that we have discussed:

1. Assigning a 1 or 0 to a condition is arbitrary.

2. A 1 is normally associated with TRUE, HIGH, ON, etc.

3. A 0 is normally associated with FALSE, LOW, OFF, etc.

Examining statements 2 and 3 shows that logic 1 is normally expected to activate some device (e.g., if $Y = 1$, then motor runs), and logic 0 is normally expected to deactivate some device (e.g., if $Y = 0$, then motor stops). If these conventions were reversed, such that logic 0 was expected to activate some device (e.g., if $Y = 0$, then motor runs) and logic 1 was expected to deactivate some device (e.g., $Y = 1$, then motor stops), the NOT function would then have a useful application.

1. A NOT is used when a 0 (LOW condition) must activate some device.

2. A NOT is used when a 1 (HIGH condition) must deactivate some device.

The following two examples show applications of the NOT function. Although the NOT function is normally used in conjunction with the AND and OR functions, the first example shows the NOT function used alone.
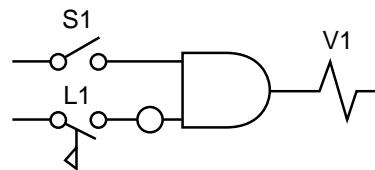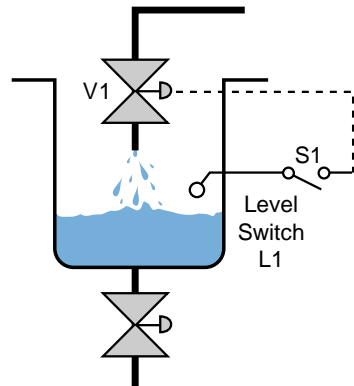
EXAMPLE 3-3

Show the logic gate, truth table, and circuit representation for a solenoid valve (V1) that will be open (ON) if selector switch S1 is ON and if level switch L1 is NOT ON (liquid has not reached level).

SOLUTION

Logic Representation

S1	L1 ($\overline{L1}$)		V1
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0

Truth Table
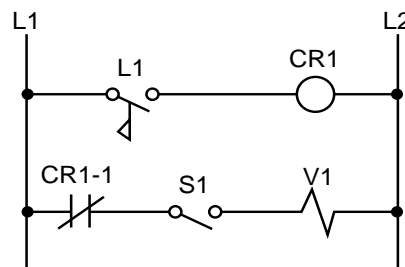
Electrical Ladder Circuit

Note: In this example, the level switch L1 is normally open, but it closes when the liquid level reaches L1. The ladder circuit requires an auxiliary control relay (CR1) to implement the not normally open L1 signal. When L1 closes (ON), CR1 is energized, thus opening the normally closed CR1-1 contacts and deactivating V1. S1 is ON when the system operation is enabled.

EXAMPLE 3-4

Show the logic gate, truth table, and circuit representation for an alarm horn that will sound if push button PB1 is 1 (ON or depressed) and PB2 is NOT 0 (not depressed).

SOLUTION

PB1

PB2

Alarm Horn

Logic Representation

PB1	PB2	Alarm Horn
Not pushed (0)	Not pushed (0)	Silent (0)
Not pushed (0)	Pushed (1)	Silent (0)
Pushed (1)	Not pushed (0)	Sounding (1)
Pushed (1)	Pushed (1)	Silent (0)

PB1   PB2

+
V
−

Electrical Circuit

Line Voltage
L1

Line Voltage (Common)
L2

PB1   PB2

Electrical Ladder Circuit

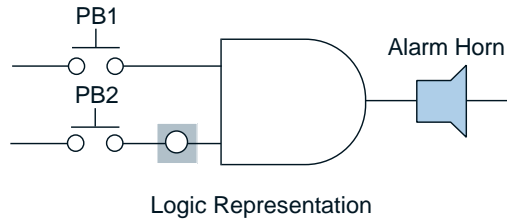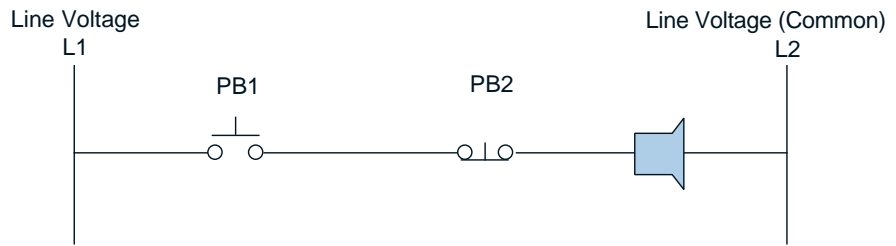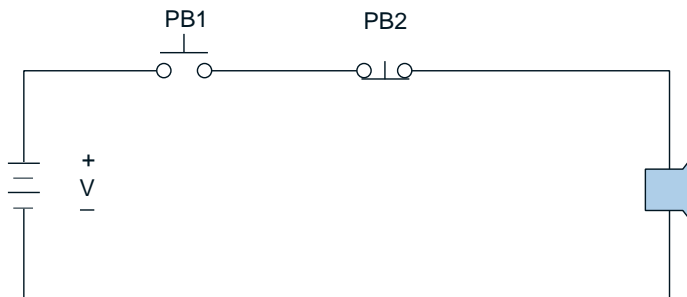**Note:** In this example, the physical representation of a field device element that signifies the NOT function is represented as a normally closed, or not normally open, switch (PB2). In the logical representation section of this example, the push button switch is represented as NOT open by the symbol.

The two previous examples showed the NOT symbol placed at inputs to a gate. A NOT symbol placed at the output of an AND gate will negate, or invert, the normal output result. A negated AND gate is called a **NAND** gate. Figure 3-7 shows its logic symbol and truth table.

NAND Truth Table		
Inputs		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0
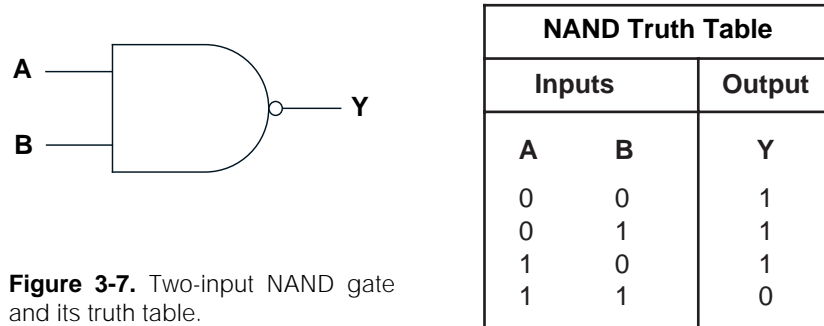
Figure 3-7. Two-input NAND gate and its truth table.

The same principle applies if a NOT symbol is placed at the output of an OR gate. The normal output is negated, and the function is referred to as a **NOR** gate. Figure 3-8 shows its symbol and truth table.

NOR Truth Table		
Inputs		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0
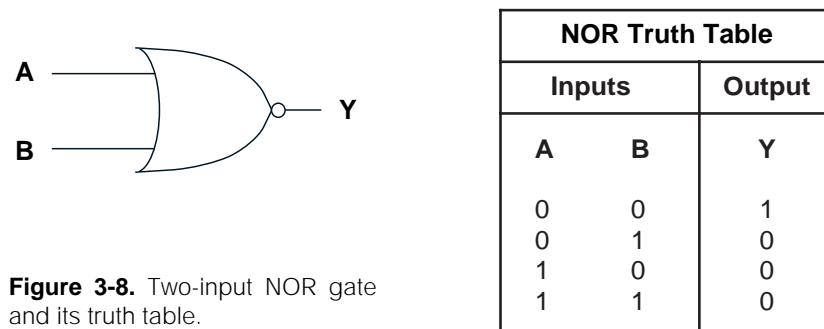
Figure 3-8. Two-input NOR gate and its truth table.

3-3 PRINCIPLES OF BOOLEAN ALGEBRA AND LOGIC

An in-depth discussion of Boolean algebra is not required for the purposes of this book and is beyond the book's scope. However, an understanding of the Boolean techniques for writing shorthand expressions for complex logical statements can be useful when creating a control program of Boolean statements or conventional ladder diagrams.

In 1849, an Englishman named George Boole developed Boolean algebra. The purpose of this algebra was to aid in the logic of reasoning, an ancient form of philosophy. It provided a simple way of writing complicated combinations of "logical statements," defined as statements that can be either true or false.

When digital logic was developed in the 1960s, Boolean algebra proved to be a simple way to analyze and express digital logic statements, since all digital systems use a TRUE/FALSE, or two-valued, logic concept. Because of this

relationship between digital logic and Boolean logic, you will occasionally hear logic gates referred to as Boolean gates, several interconnected gates called a Boolean network, or even a PLC language called a Boolean language.

Figure 3-9 summarizes the basic **Boolean operators** as they relate to the basic digital logic functions AND, OR, and NOT. These operators use capital letters to represent the wire label of an input signal, a multiplication sign (•) to represent the AND operation, and an addition sign (+) to represent the OR operation. A bar over a letter represents the NOT operation.
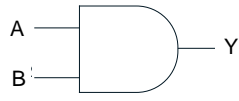
Logical Symbol	Logical Statement	Boolean Equation
A ——[ ]—— Y B	Y is 1 if A AND B are 1	$Y = A \cdot B$ or $Y = AB$
A ——[ ]—— Y B	Y is 1 if A OR B is 1	$Y = A + B$
A ——( )—— Y	Y is 1 if A is 0 Y is 0 if A is 1	$Y = \overline{A}$

Figure 3-9. Boolean algebra as related to the AND, OR, and NOT functions.

In Figure 3-9, the AND gate has two input signals (A and B) and one output signal (Y). The output can be expressed by the logical statement:

Y is 1 if A AND B are 1.

The corresponding Boolean expression is:

$$Y = A \cdot B$$

which is read *Y equals A ANDed with B*. The Boolean symbol • for AND could be removed and the expression written as $Y = AB$. Similarly, if Y is the result of ORing A and B, the Boolean expression is:

$$Y = A + B$$

which is read *Y equals A ORed with B*. In the NOT operation, where Y is the inverse of A, the Boolean expression is:

$$Y = \overline{A}$$

which is read *Y equals NOT A*. Table 3-3 illustrates the basic Boolean operations of ANDing, ORing, and inversion. The table also illustrates how these functions can be combined to obtain any desired logic combination.

1. Basic Gates. Basic logic gates implement simple logic functions. Each logic function is expressed in terms of a truth table and its Boolean expression.

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

NAND

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

OR

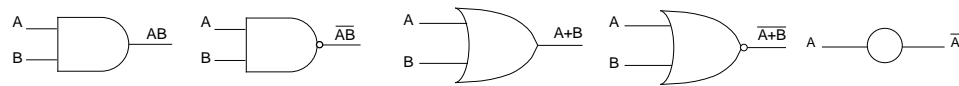| A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

NOR

A	$\overline{A}$
0	1
1	0

NOT

2. Combined Gates. Any combination of control functions can be expressed in Boolean terms using three simple operators: (•), (+), and (‾).

$Y = AB + C$

$Y = \overline{AB} + C$

$Y = (A+B)(C)$

$Y = \overline{(A+B)}(C)$
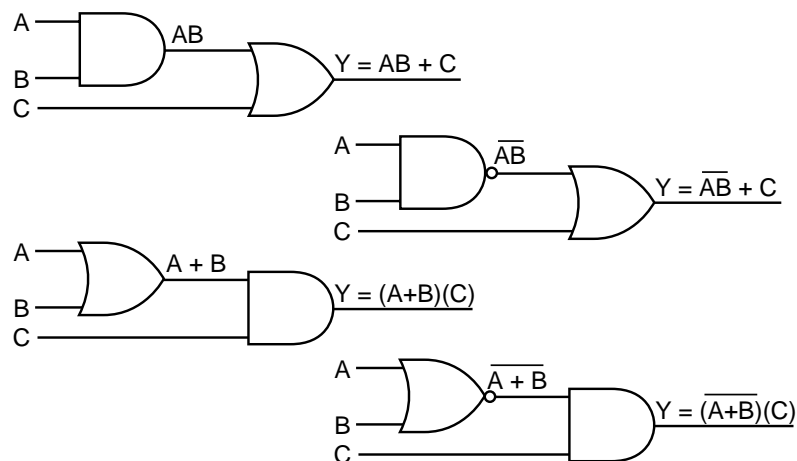
3. Boolean Algebra Rules. Control logic functions can vary from simple to very complex combinations of input variables. However simple or complex the functions may be, they satisfy the following rules. These rules are a result of a simple combination of basic truth tables and may be used to simplify logic circuits.

Commutative Laws

$A + B = B + A$

$AB = BA$

Associative Laws

$A + (B + C) = (A + B) + C$

$A(BC) = (AB)C$

De Morgan's Laws

$\overline{(A+B)} = \overline{A}\,\overline{B}$

$\overline{(AB)} = \overline{A} + \overline{B}$

$\overline{\overline{A}} = A,\ \overline{1} = 0,\ \overline{0} = 1$

$A + \overline{A}B = A + B$

$AB + AC + B\overline{C} = AC + B\overline{C}$

Distributive Laws

$A(B + C) = AB + AC$

$A + BC = (A + B)(A + C)$

Law of Absorption

$A(A + B) = A + AB = A$

Table 3-3. Logic operations using Boolean algebra.

4. Order of Operation and Grouping Signs. The order in which Boolean operations (AND, OR, NOT) are performed is important. This order will affect the resulting logic value of the expression. Consider the three input signals *A*, *B*, and *C*. Combining them in the expression $Y = A + B \cdot C$ can result in misoperation of the output device *Y*, depending on the order in which the operations are performed. Performing the OR operation prior to the AND operation is written $(A + B) \cdot C$, and performing the AND operation prior to the OR is written $A + (B \cdot C)$. The result of these two expressions is not the same.

The order of priority in Boolean expression is NOT (inversion) first, AND second, and OR last, unless otherwise indicated by grouping signs, such as parentheses, brackets, braces, or the vinculum. According to these rules, the previous expression $A + B \cdot C$, without any grouping signs, will always be evaluated only as $A + (B \cdot C)$. With the parentheses, it is obvious that *B* is ANDed with *C* prior to ORing the result with *A*. Knowing the order of evaluation, then, makes it possible to write the expression simply as $A + BC$, without fear of misoperation. As a matter of convention, the AND operator is usually omitted in Boolean expressions.

When working with Boolean logic expressions, misuse of grouping signs is a common occurrence. However, if the signs occur in pairs, they generally do not cause problems if they have been properly placed according to the desired logic. Enclosing two variables that are to be ANDed within parentheses is not necessary since the AND operator would normally be performed first. If two input signals are to be ORed prior to ANDing, they must be placed within parentheses.

To ensure proper order of evaluation of an expression, use parentheses as grouping signs. If additional signs are required brackets [ ], and then braces { } are used. An illustration of the use of grouping signs is shown below:

$$Y1 = Y2 + Y5\,[X1(X2 + X3)] + \{Y3[Y4(X5 + X6)]\}$$

5. Application of De Morgan's Laws. De Morgan's Laws are frequently used to simplify inverted logic expressions or to simply convert an expression into a usable form.
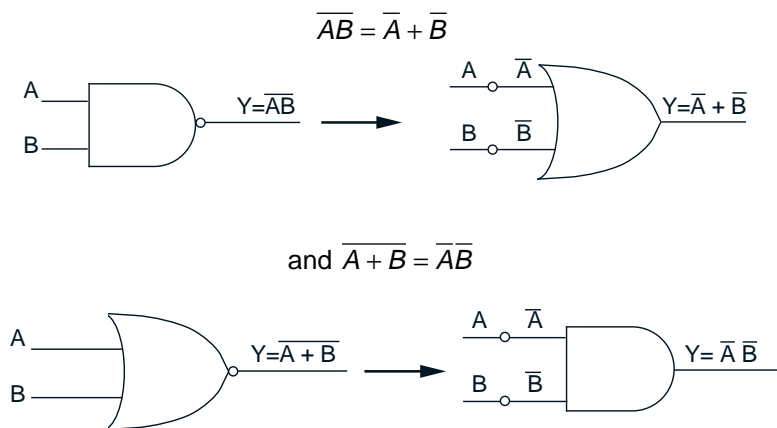
According to De Morgan's Laws:

$$\overline{AB} = \overline{A} + \overline{B}$$

and $\overline{A + B} = \overline{A}\,\overline{B}$

Table 3-3 continued.

<div style="background:#2e2e8f; color:white; padding:6px;">

3-4 PLC CIRCUITS AND LOGIC CONTACT SYMBOLOGY

</div>

Hardwired logic refers to logic control functions (timing, sequencing, and control) that are determined by the way devices are interconnected. In contrast to PLCs, in which logic functions are programmable and easily changed, hardwired logic is fixed and can be changed only by altering the way devices are physically connected or interwired. A prime function of a PLC is to replace existing hardwired control logic and to implement control functions for new systems. Figure 3-10a shows a typical hardwired relay logic circuit, and Figure 3-10b shows its PLC ladder diagram implementation. The important point about Figure 3-10 is not to understand the process of changing from one circuit to another, but to see the similarities in the representations. The ladder circuit connections of the hardwired relay circuit are implemented in the PLC via software instructions, thus all of the wiring can be thought of as being inside the CPU (*softwired* as opposed to hardwired).
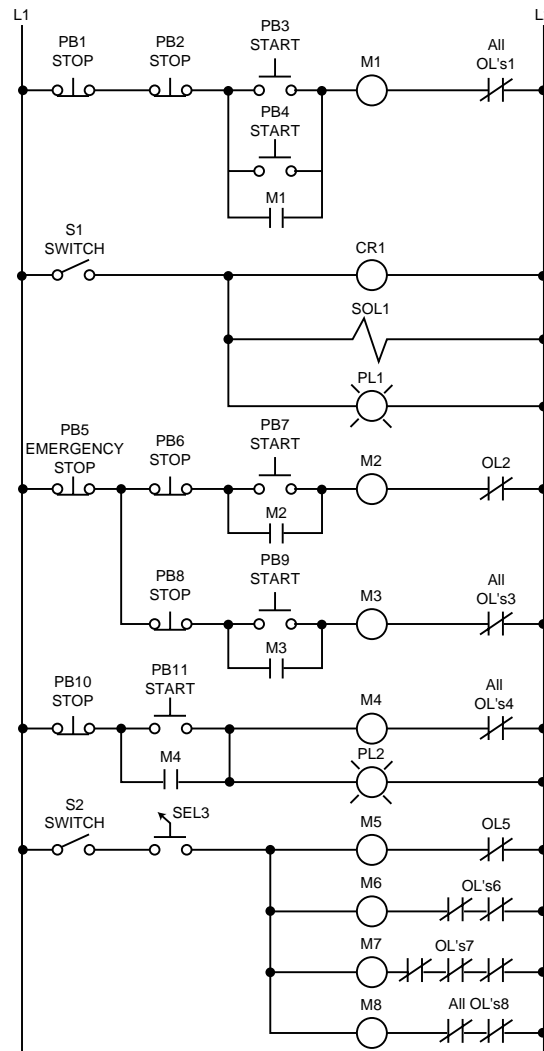
Figure 3-10a. Hardwired relay logic circuit.

The logic implemented in PLCs is based on the three basic logic functions (AND, OR, and NOT) that we discussed in the previous sections. These functions are used either alone or in combination to form instructions that will determine if a device is to be switched on or off. How these instructions are implemented to convey commands to the PLC is called the **language**. The most widely used languages for implementing on/off control and sequencing are ladder diagrams and Boolean mnemonics, among others. Chapter 9 discusses these languages at length.

The most conventional of the control languages is ladder diagram. Ladder diagrams are also called **contact symbology**, since their instructions are relay-equivalent contact symbols (i.e., normally open and normally closed contacts and coils).

Figure 3-10b. PLC ladder diagram implementation of Figure 3-10a.

Contact symbology is a very simple way of expressing control logic in terms of symbols that are used on relay control schematics. If the controller language is ladder diagram, the translation from existing relay logic to program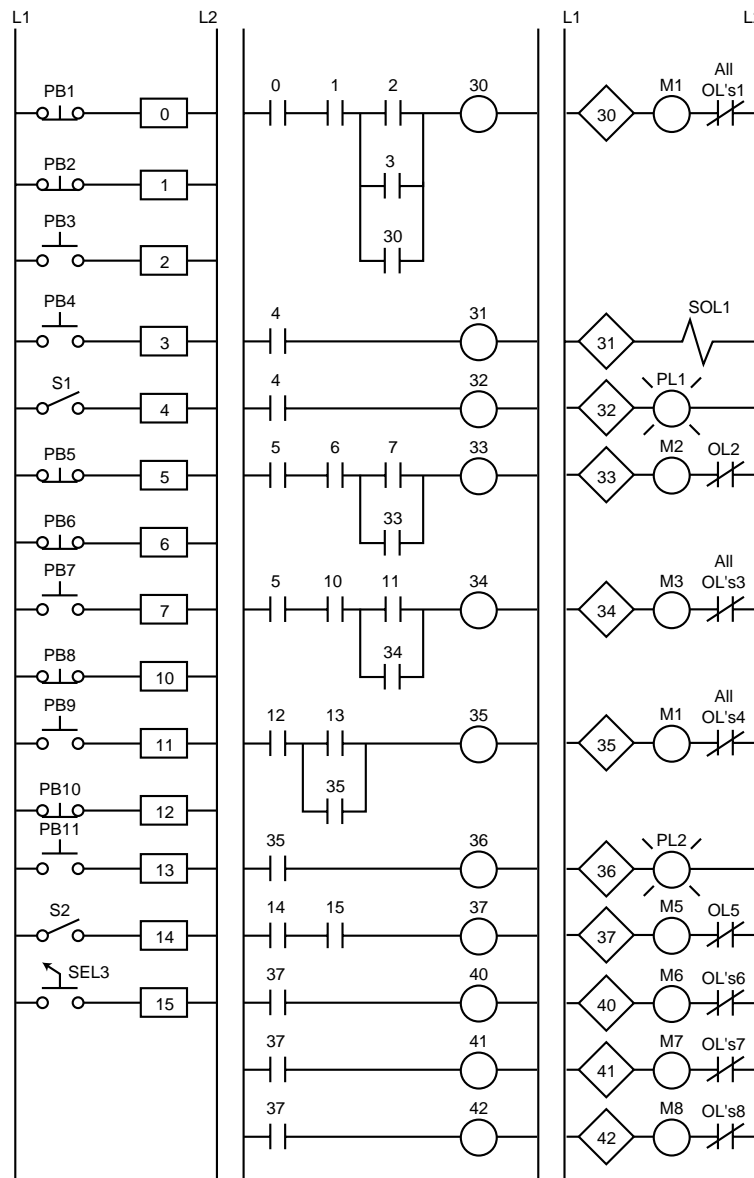med logic is a one-step translation to contact symbology. If the language is Boolean mnemonics, conversion to contact symbology is not required, yet is still useful and quite often done to provide an easily under-stood piece of documentation. Table 3-6a, shown later, provides examples of simple translations from hardwired logic to programmed logic. Chapter 11 thoroughly explains these translations.

The complete ladder circuit, in Figure 3-10, shown earlier, can be thought of as being formed by individual circuits, each circuit having one output. Each of these circuits is known as a **rung** (or network); therefore, a rung is the contact symbology required to control an output in the PLC. Some controllers allow a rung to have multiple outputs, but one output per rung is the convention. Figure 3-11a illustrates the top rung of the hardwired circuit from Figure 3-10, while Figure 3-11b shows the top rung of the equivalent PLC circuit. Note that the PLC diagram includes all of the field input and output devices connected to the interfaces that are used in the rung. A complete PLC ladder diagram program, then, consists of several rungs. Each rung controls an output interface that is connected to an **output device**, a piece of equipment that receives information from the PLC. Each rung is a combination of input conditions (symbols) connected from left to right between two vertical lines, with the symbol that represents the output at the far right.
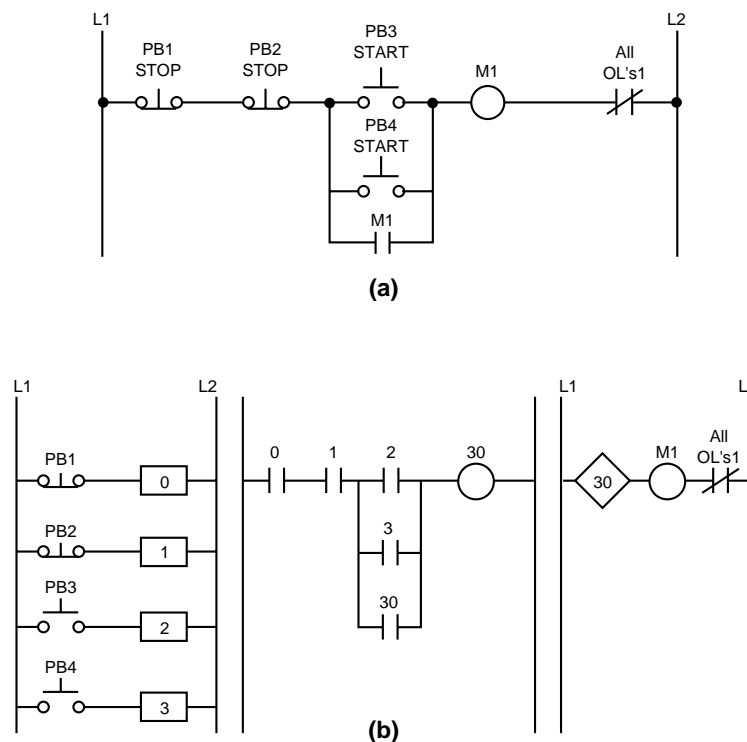

(a)

(b)

Figure 3-11. (a) Top rung of the hardwired circuit from Figure 3-10 and **(b)** its equivalent PLC circuit.

The symbols that represent the inputs are connected in series, parallel, or some combination to obtain the desired logic. These input symbols represent the **input devices** that are connected to the PLC's input interfaces. The input devices supply the PLC with field data. When completed, a ladder diagram control program consists of several rungs, with each rung controlling a specific output.

The programmed rung concept is a direct carryover from the hardwired relay ladder rung, in which input devices are connected in series and parallel to control various outputs. When activated, these input devices either allow current to flow through the circuit or cause a break in current flow, thereby switching the output devices ON or OFF. The input symbols on a ladder rung can represent signals generated by connected input devices, connected output devices, or outputs internal to the controller (see Table 3-4).

Input Devices	Output Devices
Push button	Pilot light
Selector switch	Solenoid valve
Limit switch	Horn
Proximity switch	Control relay
Timer contact	Timer

Table 3-4. ON/OFF input and output devices.

ADDRESSES USED IN PLCS

Each symbol on a rung will have a *reference number*, which is the address in memory where the current status (1 or 0) for the referenced input is stored. When a field signal is connected to an input or an output interface, its address will be related to the terminal where the signal wire is connected. The address for a given input/output can be used throughout the program as many times as required by the control logic. This PLC feature is an advantage when compared to relay-type hardware, where additional contacts often mean additional hardware. Sections 5-4 and 6-2 describe more about I/O interaction and its relationship with the PLC's memory and enclosure placement.

Figure 3-12 illustrates a simple electrical ladder circuit and its equivalent PLC implementation. Each "real" field device (e.g., push buttons PB1 and PB2, limit switch LS1, and pilot light PL1) is connected to the PLC's input and output modules (see Figure 3-13), which have a reference number—the address. Most controllers reference these devices using numeric addresses with octal (base 8) or decimal (base 10) numbering. Note that in the electrical ladder circuit, any complete electrical path (all contacts closed) from left to right will energize the output (pilot light PL1). To turn PL1 ON, then, one of the following two conditions must occur: (1) PB1 must be pressed and LS1 must be closed or (2) PB2 must be pressed and LS1 must be closed. Either of these two conditions will complete the electrical path and cause power to flow to the pilot light.

Electrical Ladder Circuit

PLC Implementation
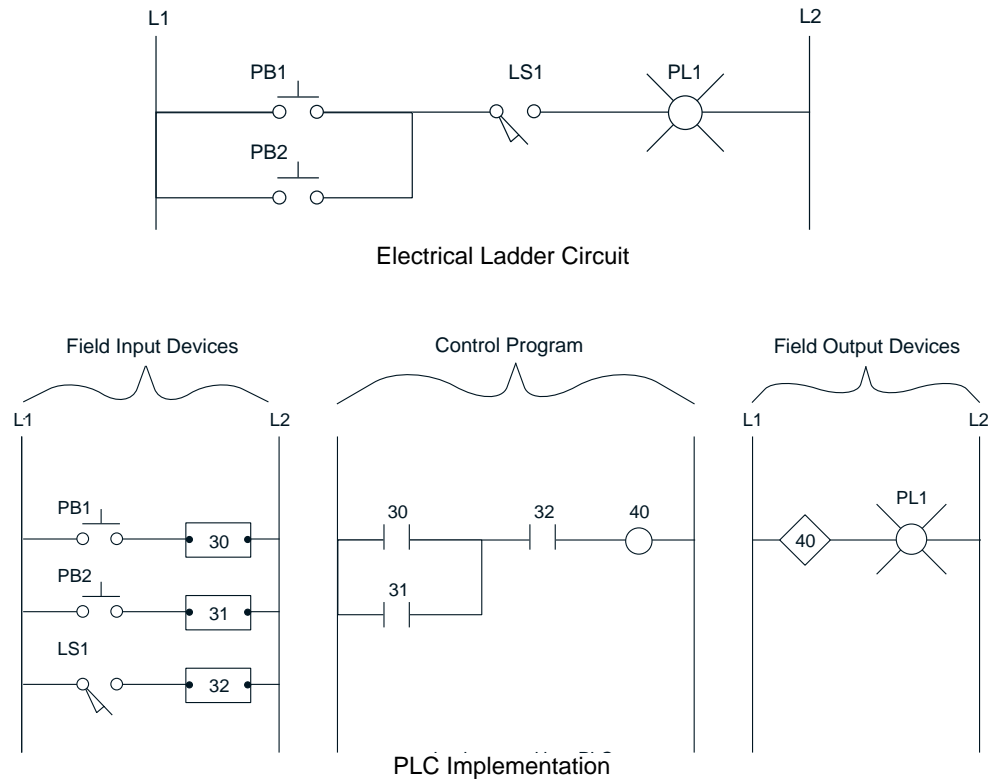
Figure 3-12. Electrical ladder circuit and its equivalent PLC implementation.
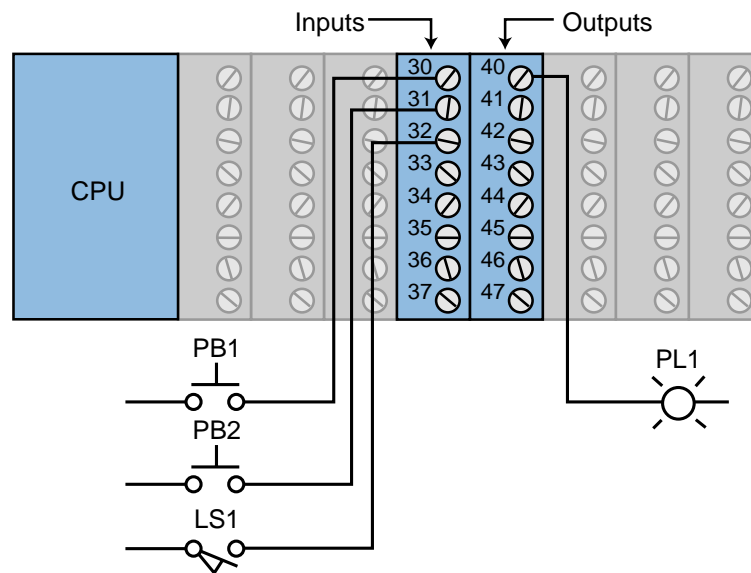

Figure 3-13. Field devices from Figure 3-12 connected to I/O module.

The same logic that applies to an electrical ladder circuit applies to a PLC circuit. In the PLC control program, power must flow through either addresses 30 (PB1) and 32 (LS1) or through addresses 31 (PB2) and 32 (LS1) to turn ON output 40. Output 40, in turn, energizes the light PL1 that is

connected to the interface with address 40. In order to provide power to addresses 30, 31, or 32, the devices connected to the input interfaces addressed 30, 31, and 32 must be turned ON. That is, the push buttons must be pressed or the limit switch must close.

CONTACT SYMBOLS USED IN PLCs

Programmable controller contacts and electromechanical relay contacts operate in a very similar fashion. For example, let's take relay A (see Figure 3-14a) which has two sets of contacts, one **normally open** contact (A-1) and one **normally closed** contact (A-2). If relay coil A is not energized (i.e., it is OFF), contact A-1 will remain open and contact A-2 will remain closed (see Figure 3-14b). Conversely, if coil A is energized, or turned ON, contact A-1 will close and contact A-2 will open (see Figure 3-14c). The blue lines highlighting the coil and contacts denote an ON, or closed, condition.

A — Relay Coil A

A-1 — Contact A-1 (NO)
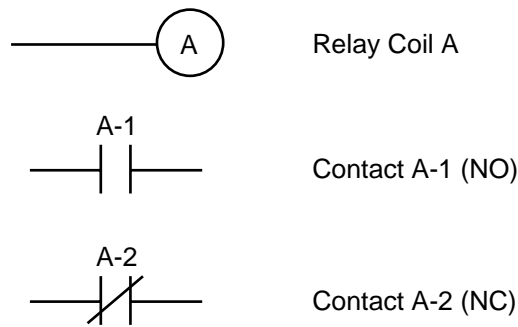
A-2 — Contact A-2 (NC)

(a) Standard configuration for relay coil A with normally open contact A-1 and normally closed contact A-2.

A — OFF
A-1 — Open
A-2 — Closed
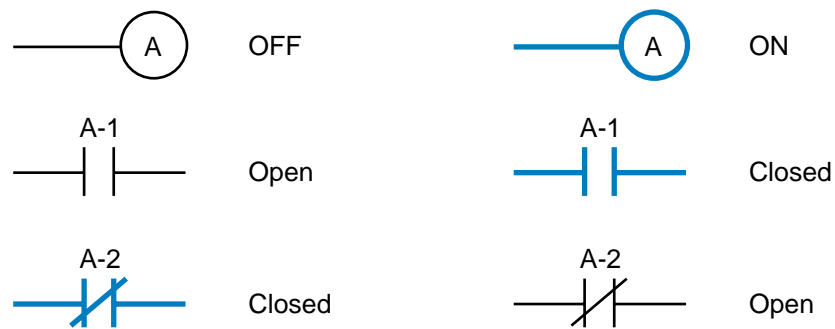
(b) Coil A de-energized.

A — ON
A-1 — Closed
A-2 — Open

(c) Coil A energized.

Figure 3-14. Relay and PLC contact symbols showing a relay coil and normally open and normally closed contacts.

Remember that when a set of contacts closes, it provides power flow, or continuity, in the circuit where it is used. Each set of available coils and its respective contacts in the PLC have a unique reference address by which they are identified. For instance, coil 10 will have normally open and normally closed contacts with the same address (10) as the coil (see Figure 3-15). Note that a PLC can have as many normally open and normally closed contacts as desired; whereas in an electromechanical relay, only a fixed number of contacts are available.
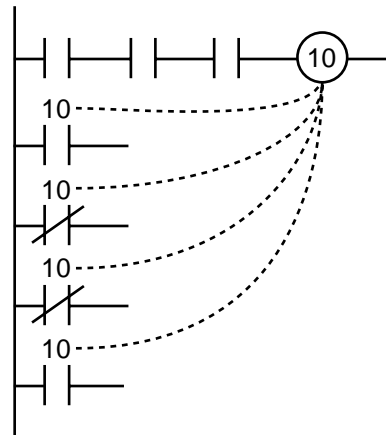
Figure 3-15. Multiple contacts from a PLC output coil.

A programmable controller also allows the multiple use of an input device reference. Figure 3-16 illustrates an example in which limit switch LS1 is connected to reference input module connection 20. Note that the PLC control program can have as many normally open and normally closed reference 20 contacts in as many rungs as needed.

Figure 3-16. Input 20 has multiple contacts in the PLC control program.

The symbols in Table 3-5 are used to translate relay control logic to contact symbolic logic. These symbols are also the basic instruction set for the ladder diagram, excluding timer/counter instructions. Chapter 9 further explains these and more advanced instructions.

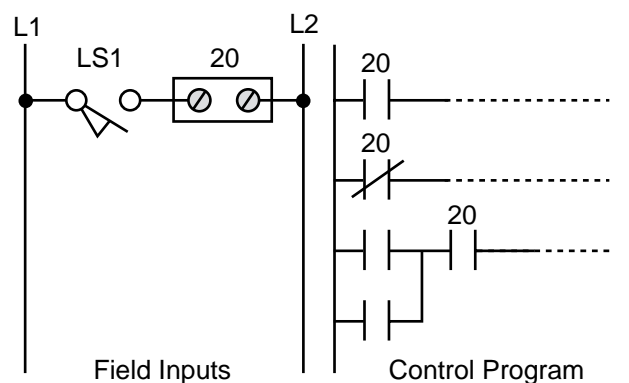| Symbol	Definition and Symbol Interpretation
⊣⊢	**Normally open contact.** Represents any input to the control logic. An input can be a connected switch closure or sensor, a contact from a connected output, or a contact from an internal output. When interpreted, the referenced input or output is examined for an ON condition. If its status is 1, the contact will close and allow current to flow through the contact. If the status of the referenced input/output is 0, the contact will remain open, prohibiting current from flowing through the contact.
⊣/⊢	**Normally closed contact.** Represents any input to the control logic. An input can be a connected switch closure or sensor, a contact from a connected output, or a contact from an internal output. When interpreted, the referenced input/output is examined for an OFF condition. If its status is 0, the contact will remain closed, thus allowing current to flow through the contact. If the status of the referenced input/output is 1, the contact will open, prohibiting current from flowing through the contact.
─○─	**Output.** Represents any output that is driven by some combination of input logic. An output can be a connected device or an internal output. If any left-to-right path of input conditions is TRUE (all contacts closed), the referenced output is energized (turned ON).
─⊘─	**NOT output.** Represents any output that is driven by some combination of input logic. An output can be a connected device or an internal output. If any left-to-right path of input conditions is TRUE (all contacts closed), the referenced output is de-energized (turned OFF).

Table 3-5. Symbols used to translate relay control logic to contact symbolic logic.

The following seven points describe guidelines for translating from hardwired logic to programmed logic using PLC contact symbols:

- **Normally open contact.** When evaluated by the program, this symbol is examined for a 1 to close the contact; therefore, the signal referenced by the symbol must be ON, CLOSED, activated, etc.

- **Normally closed contact.** When evaluated by the program, this symbol is examined for a 0 to keep the contact closed; thus, the signal referenced by the symbol must be OFF, OPEN, deactivated, etc.

- **Output.** An output on a given rung will be energized if any left-to-right path has all contacts closed, with the exception of power flow going in reverse before continuing to the right. An output can control either a connected device (if the reference address is also a termina-

tion point) or an **internal output** used exclusively within the program. An internal output does not control a field device. Rather, it provides interlocking functions within the PLC.

- **Input.** This contact symbol can represent input signals sent from connected inputs, contacts from internal outputs, or contacts from connected outputs.

- **Contact addresses.** Each program symbol is referenced by an address. If the symbol references a connected input/output device, then the address is determined by the point where the device is connected.

- **Repeated use of contacts.** A given input, output, or internal output can be used throughout the program as many times as required.

- **Logic format.** Contacts can be programmed in series or in parallel, depending on the output control logic required. The number of series contacts or parallel branches allowed in a rung depends on the PLC.

Table 3-6a show how simple hardwired series and parallel circuits can be translated into programmed logic. A **series circuit** is equivalent to the Boolean AND operation; therefore, all inputs must be ON to activate the output. A **parallel circuit** is equivalent to the Boolean OR operation; therefore, any one of the inputs must be ON to activate the output. The STR and OUT Boolean statements stand for START (of a new rung) and OUTPUT (of a rung), respectively. Table 3-6b further explains Table 3-6a.

KEY TERMS

AND
Boolean operators
contact symbology
gate
input device
internal output
language
NAND
negative logic
NOR
normally closed
normally open
NOT
OR
output device
parallel circuit
positive logic
rung
series circuit
truth table

(b)

(a) **Series Circuit.** In this circuit, if both switches LS1 and LS2 are closed, the solenoid SOL1 will energize.

(b) **Parallel Circuit.** In this circuit, if either of the two switches LS3 or LS4 closes, the solenoid SOL2 will energize.

(c) &
(d) **Series/Parallel Circuits.** In a series/parallel circuit, the result of ORing two or more inputs is ANDed with one or more series or parallel inputs. In both of these examples, all of the relay circuit elements are normally open and must be closed to activate the pilot lights. Normally open contacts are used in the program.
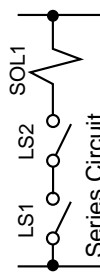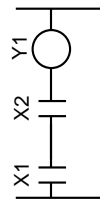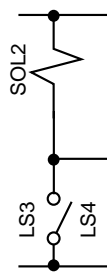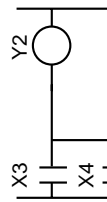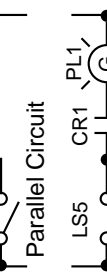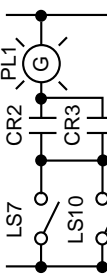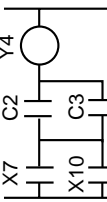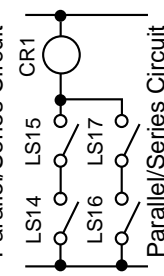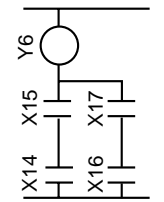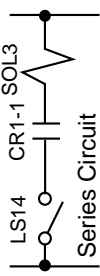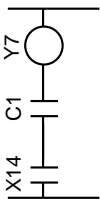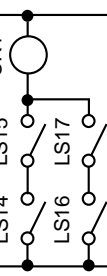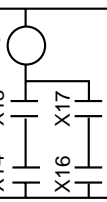
(e) &
(f) **Parallel/Series Circuits.** In a parallel/series circuit, the result of ANDing two or more inputs is ORed with one or more series inputs. In both of these examples, all of the relay circuit elements are normally open and must be closed to activate the output device. Normally open contacts are used in the program.

(f) **Internal Outputs.** Circuit **(f)** controls an electromechanical control relay. Control relays do not normally drive output devices, but rather drive other relays. They are used to provide additional contacts for interlocking logic. The internal output provides the same function in software; however, the number of contacts are unlimited and can be either normally open or normally closed.

(g) **Normally Open Contacts.** In the series circuit **(g)**, the solenoid will energize if LS14 closes and CR1-1 is energized. CR1-1 is a contact from the control relay CR1 in circuit **(f)** and closes whenever CR1 is energized. In the program, CR1 was replaced by the internal output C1; therefore, the program uses a normally open contact from the internal output C1. SOL3 will energize when LS14 closes and C1 is energized.

(h) **Normally Closed Contacts.** In circuit **(h)**, the solenoid will energize if LS14 closes and CR1-1 is not energized. The program uses a normally closed contact from the internal output C1. SOL3 will stay energized as long as the limit switch is closed and C1 is not energized.

Table 3-6. (a) Hardwired relay logic translated into PLC logic using contact symbols and **(b)** an explanation of the translation.

(a)

Relay Ladder Diagram	Contact or Ladder Diagram	Boolean Equation	Boolean Statements
(a) Series Circuit		$Y1 = X1 \cdot X2$	STR X1 / AND X2 / OUT Y1
(b) Parallel Circuit		$Y2 = X3 + X4$	STR X3 / OR X4 / OUT Y2
(c) Series/Parallel Circuit		$Y3 = (X5 + X6) \cdot C1$	STR X5 / OR X6 / AND C1 / OUT Y3
(d) Series/Parallel Circuit		$Y4 = (X7 + X10) \cdot (C2 + C3)$	STR X7 / AND X10 / STR C2 / OR C3 / OUT Y4
(e)		$Y5 = (X11 \cdot X12) + X13$	STR X11 / AND X12 / OR X13 / OUT Y5
(f) Parallel/Series Circuit		$Y6 = (X14 \cdot X15) + (X16 \cdot X17)$	STR X14 / AND X15 / OR X16 / AND X17 / OUT Y6
(g) Series Circuit		$Y7 = X14 \cdot C1$	STR X14 / AND C1 / OUT Y7
(h) Series Circuit		$Y10 = X14 \cdot \overline{C1}$	STR X14 / AND / NOT C1 / OUT Y10

This page intentionally left blank.

COMPONENTS AND SYSTEMS

- Processors, the Power Supply, and Programming Devices
- The Memory System and I/O Interaction
- The Discrete Input/Output System
- The Analog Input/Output System
- Special Function I/O and Serial Communication Interfacing

This page intentionally left blank.

PROCESSORS, THE POWER SUPPLY, AND PROGRAMMING DEVICES

Unity makes strength, and since we must be strong, we must also be one.

—Grand Duke Friedrich von Baden

CHAPTER
HIGHLIGHTS

The processor and the power supply are important parts of the central processing unit. In this chapter, we will take a look at these CPU components, concentrating on their roles and requirements in PLC applications. In addition, we will discuss the importance of CPU subsystem communications, error detection and correction, and power supply loading. Finally, we will present some of the most common programming devices for entering and editing the control program. The next chapter will discuss the other major component of the CPU—the memory system—and will explore the relationship between input/output field devices, memory, and the PLC.

4-1 INTRODUCTION

As mentioned in the first chapter, the central processing unit, or CPU, is the most important element of a PLC. The CPU forms what can be considered to be the "brain" of the system. The three components of the CPU are:

- the processor

- the memory system

- the power supply

Figure 4-1 illustrates a simplified block diagram of a CPU. CPU architecture may differ from one manufacturer to another, but in general, most CPUs follow this typical three-component organization. Although this diagram shows the power supply inside the CPU block enclosure, the power supply may be a separate unit that is mounted next to the block enclosure containing the processor and memory. Figure 4-2 shows a CPU with a built-in power supply. The programming device, not regarded as part of the CPU per se, completes the total central architecture as the medium of communication between the programmer and the CPU.
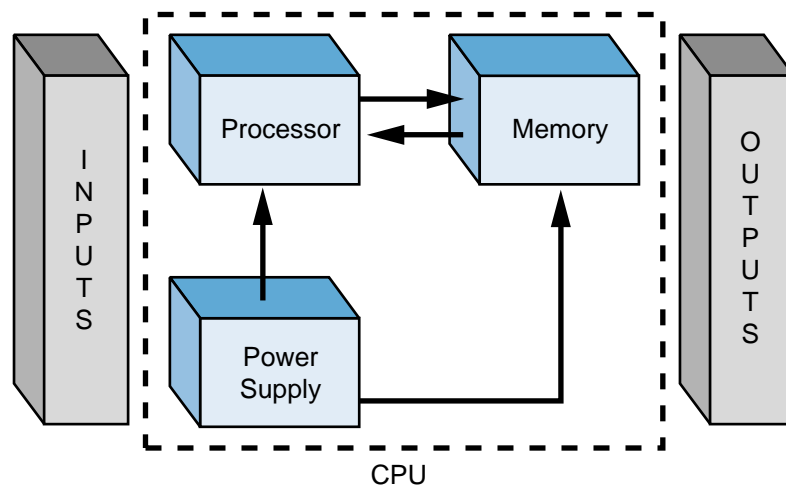
Figure 4-1. CPU block diagram.
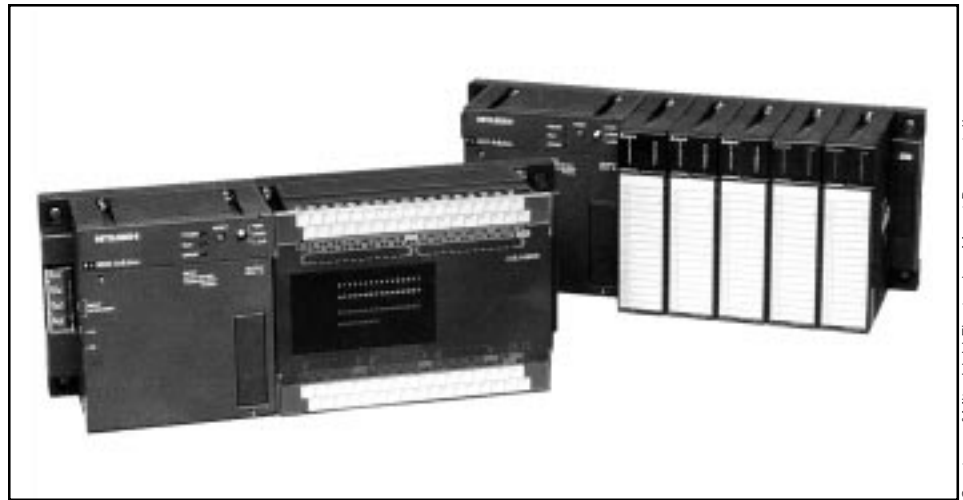
Courtesy of Mitsubishi Electronics, Mount Prospect, IL

Figure 4-2. Two PLC CPUs with built-in power supplies (left with fixed I/O blocks and right with configurable I/O).

The term *CPU* is often used interchangeably with the word *processor*; however, the CPU encompasses all of the necessary elements that form the intelligence of the system—the processor plus the memory system and power supply. Integral relationships exist between the components of the CPU, resulting in constant interaction among them. Figure 4-3 illustrates the functional interaction between a PLC's basic components. In general, the
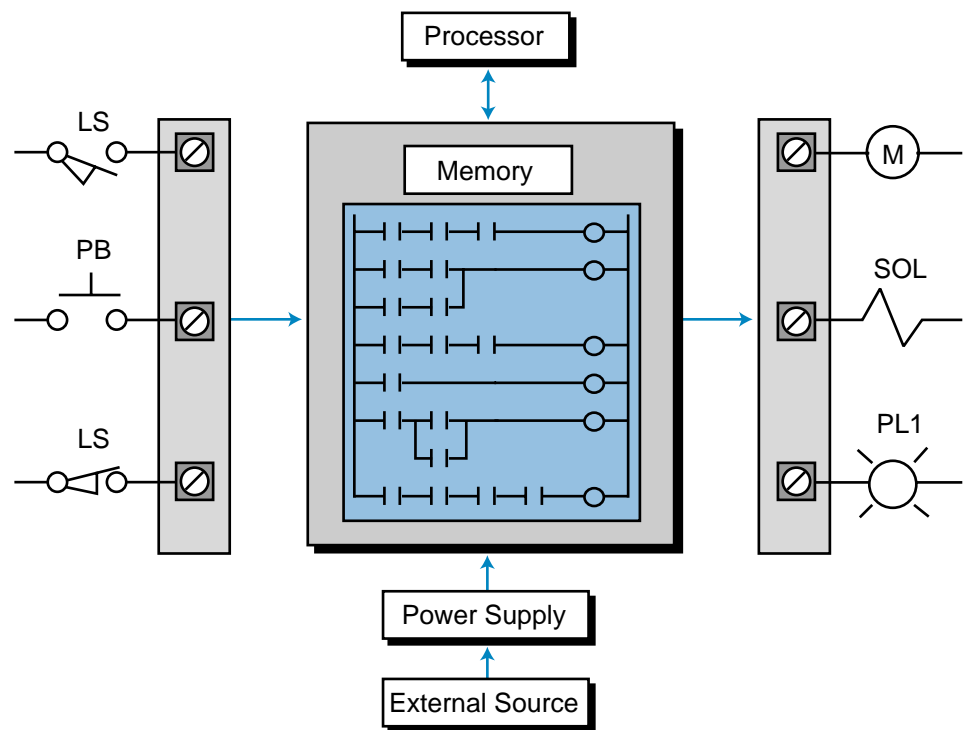
Figure 4-3. Functional interaction of a PLC system.

processor executes the control program stored in the memory system in the form of ladder diagrams, while the system power supply provides all of the necessary voltage levels to ensure proper operation of the processor and memory components.

4-2 PROCESSORS

Very small **microprocessors** (or micros)—integrated circuits with tremendous computing and control capability—provide the intelligence of today's programmable controllers. They perform mathematical operations, data handling, and diagnostic routines that were not possible with relays or their predecessor, the hardwired logic processor. Figure 4-4 illustrates a processor module that contains a microprocessor, its supporting circuitry, and a memory system.

Courtesy of Allen-Bradley, Highland Heights, OH

Figure 4-4. Allen Bradley's PLC processors—models 5/12, 5/15, and 5/25.

The principal function of the processor is to command and govern the activities of the entire system. It performs this function by interpreting and executing a collection of system programs known as the executive. The executive, a group of supervisory programs, is permanently stored in the processor and is considered a part of the controller itself. By executing the executive, the processor can perform all of its control, processing, communication, and other housekeeping functions.

The executive performs the communication between the PLC system and the user via the programming device. It also supports other peripheral communication, such as monitoring field devices; reading diagnostic data from the power supply, I/O modules, and memory; and communicating with an operator interface.

The CPU of a PLC system may contain more than one processor (or micro) to execute the system's duties and/or communications, because extra processors increase the speed of these operations. This approach of using several microprocessors to divide control and communication tasks is known as **multiprocessing**. Figure 4-5 illustrates a multiprocessor configuration.

Power Supply

Basic Computer Processor Module

Main CPU Processor

PID Processor Module
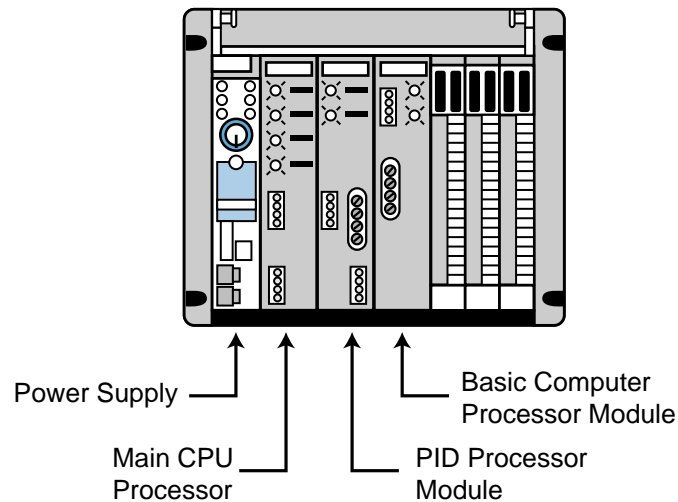
Figure 4-5. A multiprocessor configuration.

Another multiprocessor arrangement takes the microprocessor intelligence away from the CPU, moving it to an intelligent module. This technique uses intelligent I/O interfaces, which contain a microprocessor, built-in memory, and a mini-executive that performs independent control tasks. Typical intelligent modules are proportional-integral-derivative (PID) control modules, which perform closed-loop control independent of the CPU, and some stepper and servo motor control interfaces. Figure 4-6 shows some intelligent I/O modules.

Courtesy of Mitsubishi Electronics, Mount Prospect, IL

Courtesy of Allen-Bradley, Highland Heights, OH

(a)   **(b)**
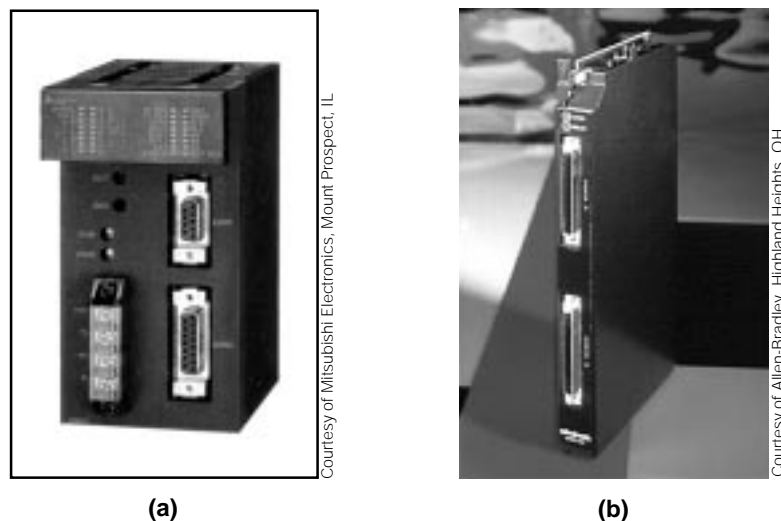
Figure 4-6. (a) A single-axis positioning module and **(b)** a temperature control interface.

The microprocessors used in PLCs are categorized according to their word size, or the number of bits that they use simultaneously to perform operations. Standard word lengths are 8, 16, and 32 bits. This word length affects the speed at which the processor performs most operations. For example, a 32-bit microprocessor can manipulate data faster than a 16-bit micro, since it manipulates twice as much data in one operation. Word length correlates with the capability and degree of sophistication of the controller (i.e., the larger the word length, the more sophisticated the controller).

4-3 PROCESSOR SCAN

The basic function of a programmable controller is to read all of the field input devices and then execute the control program, which according to the logic programmed, will turn the field output devices ON or OFF. In reality, this last process of turning the output devices ON or OFF occurs in two steps. First, as the processor executes the internal programmed logic, it will turn each of its programmed internal output coils ON or OFF. The energizing or de-energizing of these internal outputs will not, however, turn the output devices ON or OFF. Next, when the processor has finished evaluating all of the control logic program that turns the internal coils ON or OFF, it will perform an update to the output interface modules, thereby turning the field devices connected to each interface terminal ON or OFF. This process of reading the inputs, executing the program, and updating the outputs is known as the *scan*.

Figure 4-7 shows a graphic representation of the scan. The scanning process is repeated over and over in the same fashion, making the operation sequential from top to bottom. Sometimes, for the sake of simplicity, PLC manufacturers
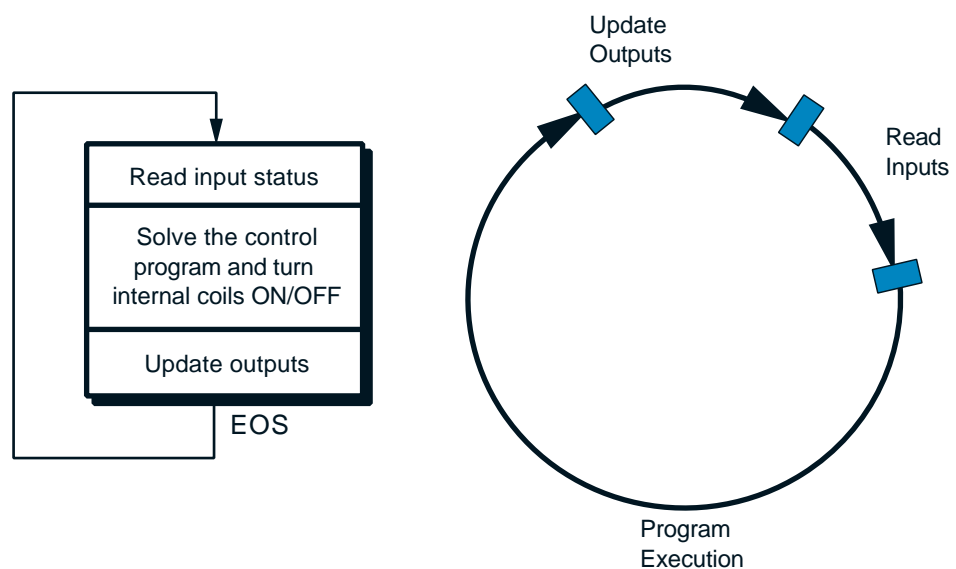
Figure 4-7. PLC total scan representation.

call the solving of the control program the **program scan** and the reading of inputs and updating of outputs the **I/O update scan**. Nevertheless, the total system scan includes both. The internal processor signal, which indicates that the program scan has ended, is called the *end-of-scan* (EOS) signal.

The time it takes to implement a scan is called the **scan time**. The scan time is the total time the PLC takes to complete the program and I/O update scans. The program scan time generally depends on two factors: (1) the amount of memory taken by the control program and (2) the type of instructions used in the program (which affects the time needed to execute the instructions). The time required to make a single scan can vary from a few tenths of a millisecond to 50 milliseconds.

PLC manufacturers specify the scan time based only on the amount of application memory used (e.g., 1 msec/1K of programmed memory). However, other factors also affect the scan time. The use of remote I/O subsystems can increase the scan time, since the PLC must transmit and receive the I/O update from remote systems. Monitoring control programs also adds time to the scan, because the microprocessor must send data about the status of the coils and contacts to a monitoring device (e.g., a PC).

The scan is normally a continuous, sequential process of reading the status of the inputs, evaluating the control logic, and updating the outputs. A processor is able to read an input as long as the input signal *is not* faster than the scan time (i.e., the input signal does not change state—ON to OFF to ON or vice versa—twice during the processor's scan time). For instance, if a controller has a total scan time of 10 msec (see Figure 4-8) and must monitor an input
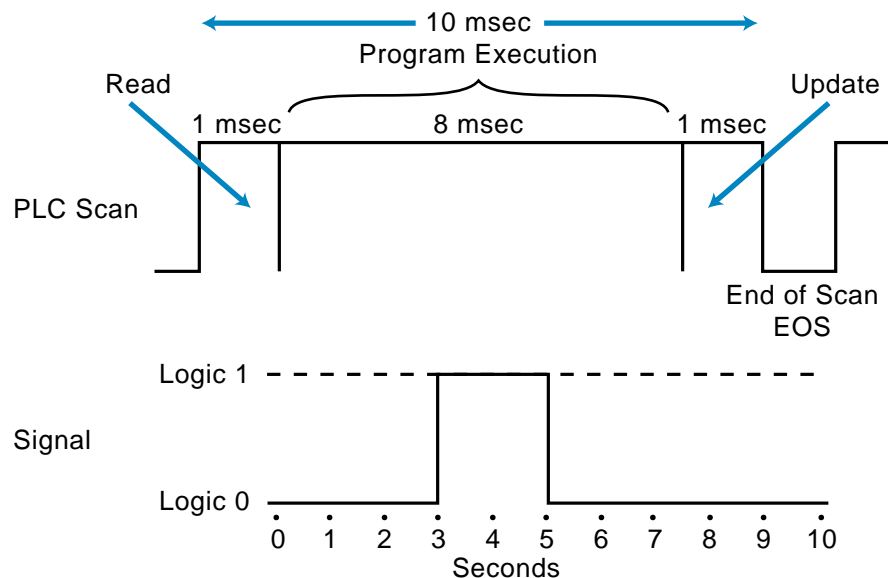
Figure 4-8. Illustration of a signal that will not be detected by a PLC during a normal scan.

signal that changes states twice during an 8 msec period (less than the scan), the programmable controller will not be able to "see" the signal, resulting in a possible machine or process malfunction. This scan characteristic must always be considered when reading discrete input signals and ASCII characters (see the ASCII section in Chapter 8). A programmable controller's scan specification indicates how fast it can react to inputs and still correctly solve the control logic. Chapter 9 provides more information about scan evaluation.

EXAMPLE 4-1

What occurs during the scanning operation of a programmable controller if the signal(s) from an input field device behave as shown in Figures 4-9a and 4-9b?
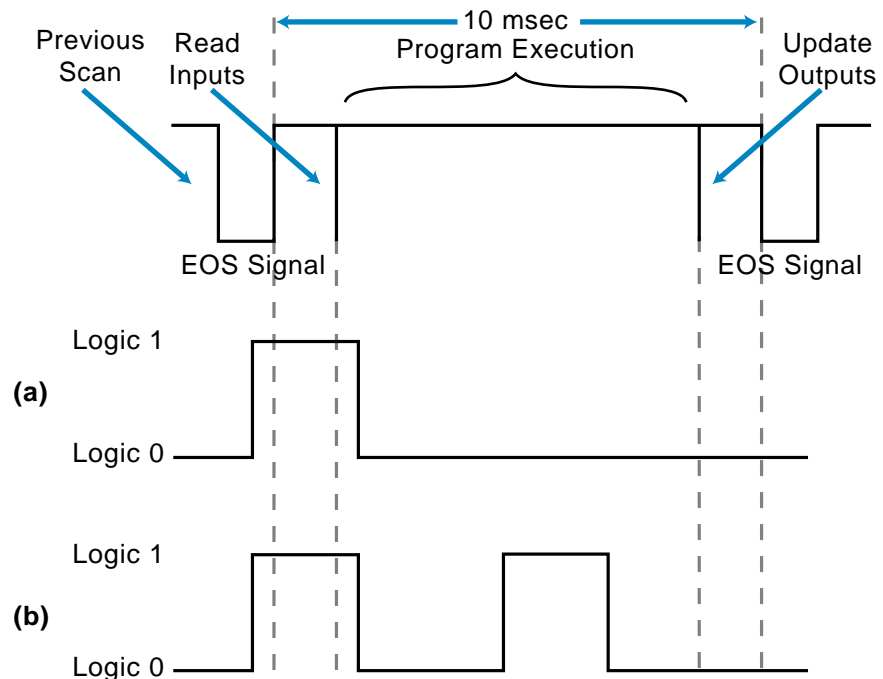
Figure 4-9. (a) Single-pulse and **(b)** double-pulse signals.

SOLUTION

In Figure 4-9a, the PLC will recognize the signal, even though it is shorter than the scan, because it was ON during the read section of the scan. In Figure 4-9b, the PLC will recognize the first signal, but it will not be able to detect the second pulse because this second ON-OFF-ON transition occurred in the middle of the scan. Thus, the PLC can not read it.

> Note that although the signal in Figure 4-9a is shorter than the scan, the PLC recognizes it. However, the user should take precautions against signals that behave like this, because if the same signal occurs in the middle of the scan, the PLC will not detect it.
>
> Also note that the behavior of the signal in Figure 4-9b will cause a misreading of the pulse. For instance, if the pulses are being counted, a counting malfunction will occur. These problems, however, can be corrected, as you will see later.

The common scan method of monitoring the inputs at the end of each scan may be inadequate for reading certain extremely fast inputs. Some PLCs provide software instructions that allow the interruption of the continuous program scan to receive an input or to update an output immediately. Figure 4-10 illustrates how immediate instructions operate during a normal program scan. These immediate instructions are very useful when the PLC must react instantaneously to a critical input or output.
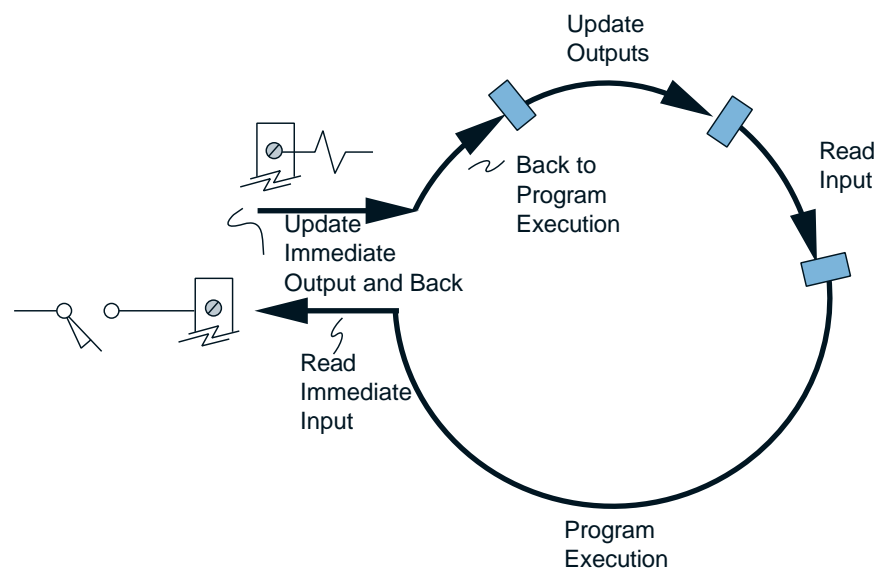
Figure 4-10. PLC scan with immediate I/O update.

Another method for reading extremely fast inputs involves using a *pulse stretcher*, or fast-response module (see Figure 4-11). This module stretches the signal so that it will last for at least one complete scan. With this type of interface, the user must ensure that the signal does not occur more than once per two scans; otherwise, some pulses will be lost. A pulse stretcher is ideal for applications with very fast input signals (e.g., 50 microseconds), perhaps from an instrumentation field device, that do not change state more than once per two scans. If a large number of pulses must be read in a shorter time than the scan time, a high-speed pulse counter input module can be used to read all the pulses and then send the information to the CPU.

One Scan   One Scan

Read

Update

Program Execution

Logic 1

50 µsec
Signal

Logic 0

Logic 1

Logic 0
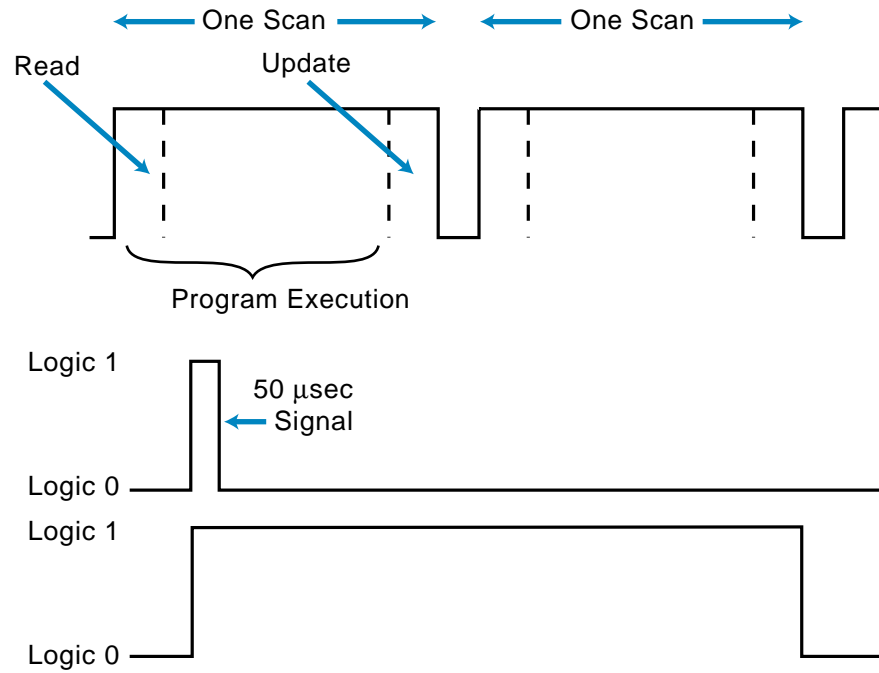
EXAMPLE 4-2

Referencing Figure 4-12, illustrate how, in one scan, **(a)** an immediate instruction will respond to an interrupt input and **(b)** the same input instruction can update an immediate output field device, like a solenoid.

10 msec
Program Execution

Read

Update

End of Scan
EOS

Logic 1

Input
Signal "N"

Logic 0

0   1   2   3   4   5   6   7   8   9   10
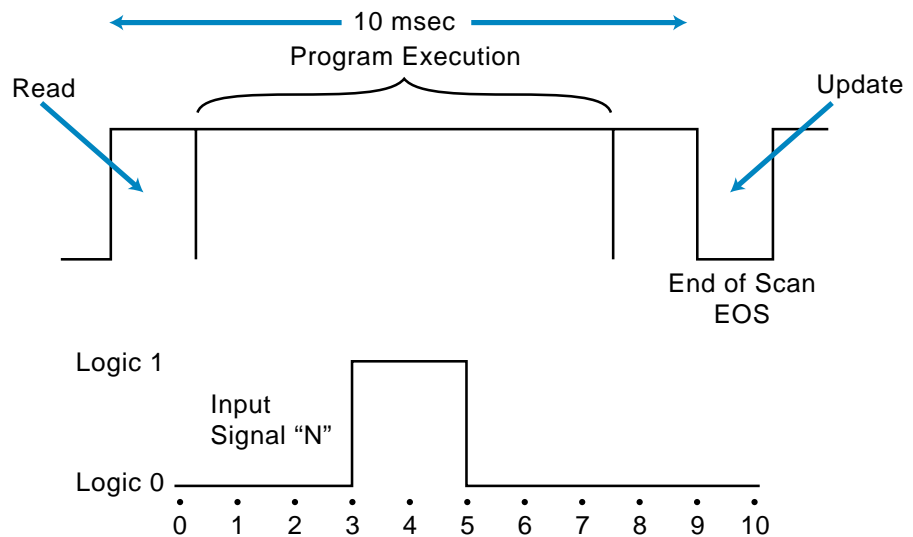
Figure 4-12. Example scan and signal.

SOLUTION

(a) As shown in Figure 4-13, the immediate instruction will interrupt the control program to read the input signal. It will then evaluate the signal and return to the control program, where it will resume program execution and update outputs.
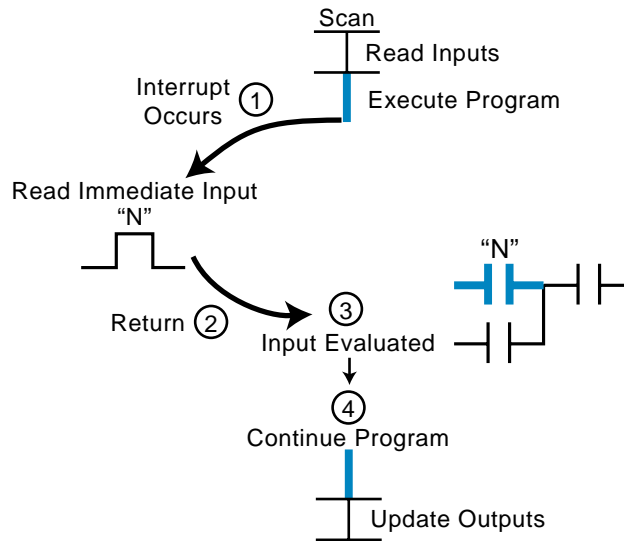
Figure 4-13. Immediate response to an interrupt input.

(b) Figure 4-14 depicts the immediate update of an output. As in part (a), the immediate instruction interrupts the control program to read and evaluate the input signal. However, the output is updated before normal program execution resumes.
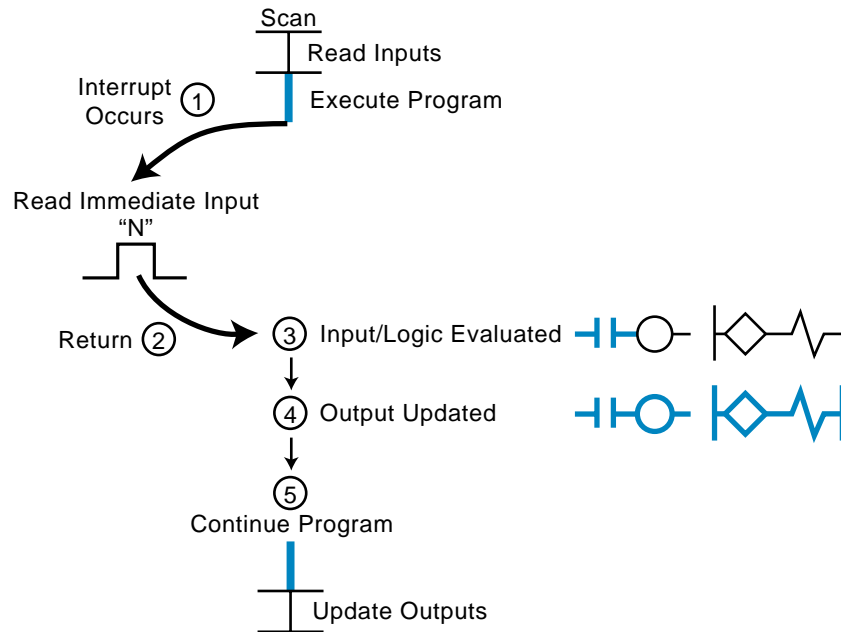
Figure 4-14. Immediate update of an output field device.

4-4 ERROR CHECKING AND DIAGNOSTICS

The PLC's processor constantly communicates with local and remote subsystems (see Chapter 6), or *racks* as they may also be called. I/O interfaces connect these subsystems to field devices located either close to the main CPU or at remote locations. Subsystem communication involves data transfer exchange at the end of each program scan, when the processor sends the latest status of outputs to the I/O subsystem and receives the current status of inputs and outputs. An I/O subsystem adapter module, located in the CPU, and a remote I/O processor module, located in the subsystem chassis or rack, perform the actual communication between the processor and the subsystem. Figure 4-15 illustrates a typical PLC subsystem configuration.
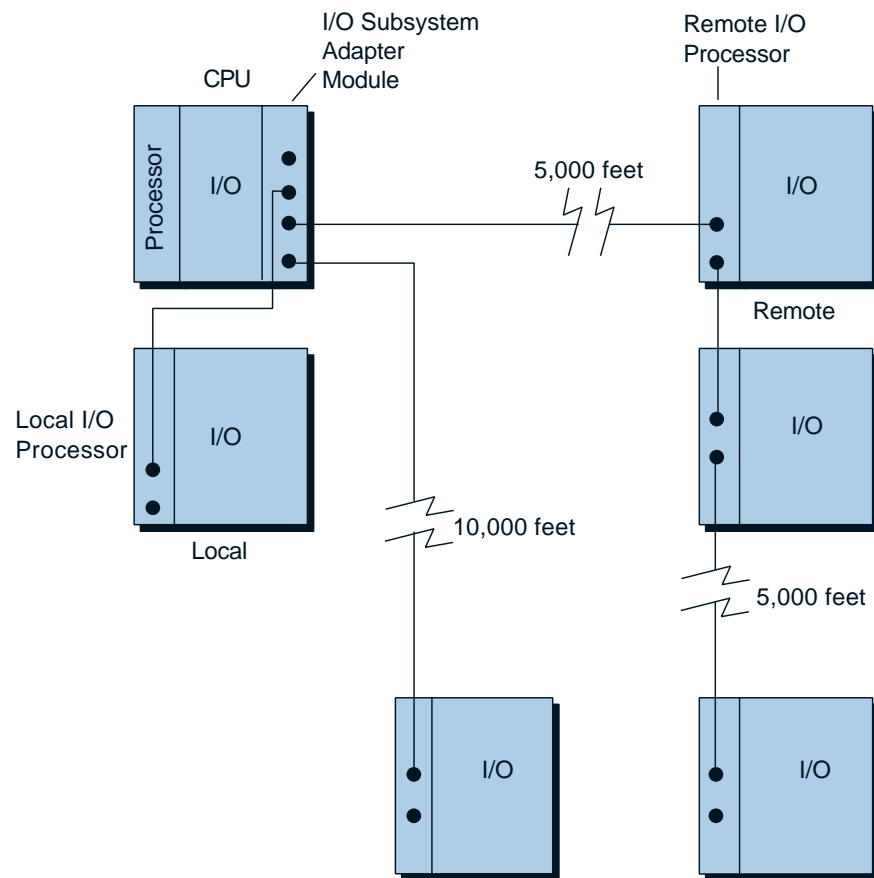
Figure 4-15. Typical PLC subsystem configuration.

The distance between the CPU and a subsystem can vary, depending on the controller, and usually ranges between 1,000 and 15,000 feet. The communication medium generally used is either twisted-pair, twinaxial, coaxial, or fiber-optic cable, depending on the PLC and the distance.

The controller transmits data to subsystems at very high speeds, but the actual speed varies depending on the controller. The data format also varies, but it is normally a serial binary format composed of a fixed number of data bits (I/O status), start and stop bits, and error detection codes.

Error-checking techniques are also incorporated in the continuous communication between the processor and its subsystems. These techniques confirm the validity of the data transmitted and received. The level of sophistication of error checking varies from one manufacturer to another, as does the type of errors reported and the resulting protective or corrective action.

ERROR CHECKING

The processor uses error-checking techniques to monitor the functional status of both the memory and the communication links between subsystems and peripherals, as well as its own operation. Common error-checking techniques include parity and checksum.

Parity. Parity is perhaps the most common error detection technique. It is used primarily in communication link applications to detect mistakes in long, error-prone data transmission lines. The communication between the CPU and subsystems is a prime example of the useful application of parity error checking. Parity check is often called **vertical redundancy check (VRC).**

Parity uses the number of 1s in a binary word to check the validity of data transmission. There are two types of parity checks: *even parity*, which checks for an even number of 1s, and *odd parity*, which checks for an odd number of 1s. When data is transmitted through a PLC, it is sent in binary format, using 1s and 0s. The number of 1s can be either odd or even, depending on the character or data being transmitted (see Figure 4-16a). In parity data transmission, an extra bit is added to the binary word, generally in the most significant or least significant bit position (see Figure 4-16b). This extra bit, called the **parity bit** (P), is used to make each byte or word have an odd or even number of 1s, depending on the type of parity being used.

(a)	1011 0110 1000 1010	Even 1s
	1011 0110 1000 1000	Odd 1s

(b)	Parity→ (P = 0 or 1)	P 1011 0110 1000 1010	Even 1s
		P 1011 0110 1000 1000	Odd 1s
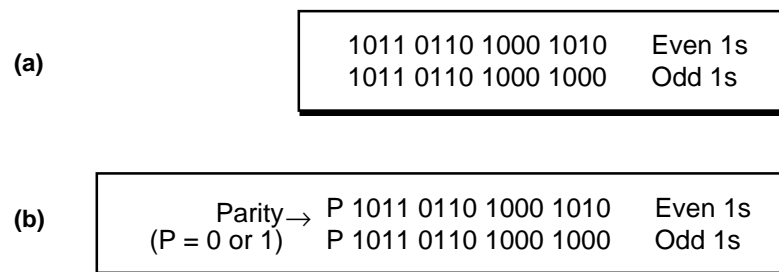
Figure 4-16. (a) A 16-bit data transmission of 1s and 0s and **(b)** the same transmission with a parity bit (P) in the most significant bit position.

Let's suppose that a processor transmits the 7-bit ASCII character *C* (1000011) to a peripheral device and odd parity is required. The total number of 1s is three, or odd. If the parity bit (P) is the most significant bit, the transmitted data will be P1000011. To achieve odd parity, P is set to 0 to obtain an odd number of 1s. The receiving end detects an error if the data does not contain an odd number of 1s. If even parity had been the error-checking method, P would have been set to 1 to obtain an even number of 1s.

Parity error checking is a single-error detection method. If one bit of data in a word changes, an error will be detected due to the change in the bit pattern. However, if two bits change value, the number of 1s will be changed back, and an error will not be detected even though there is a mistransmission.

In PLCs, when data is transmitted to a subsystem, the controller defines the type of parity (odd or even) that will be used. However, if the data transmission is from the programmable controller to a peripheral, the parity method must be prespecified and must be the same for both devices.

Some processors do not use parity when transmitting information, although their peripherals may require it. In this case, parity generation can be accomplished through application software. The parity bit can be set for odd or even parity with a short routine using functional blocks or a high-level language. If a nonparity-oriented processor receives data that contains parity, a software routine can also be used to mask out, or strip, the parity bit.

Checksum. The extra bit of data added to each word when using parity error detection is often too wasteful to be desirable. In data storage, for example, error detection is desirable, but storing one extra bit for every eight bits means a 12.5% loss of data storage capacity. For this reason, a data block error-checking method known as **checksum** is used.

Checksum error detection spots errors in blocks of many words, instead of in individual words as parity does. Checksum analyzes all of the words in a data block and then adds to the end of the block one word that reflects a characteristic of the block. Figure 4-17 shows this last word, known as the **block check character (BCC)**. This type of error checking is appropriate for memory checks and is usually done at power-up.

There are several methods of checksum computation, with the three most common being:

- cyclic redundancy check

- longitudinal redundancy check

- cyclic exclusive-OR checksum

Cyclic Redundancy Check. **Cyclic redundancy check (CRC)** is a technique that performs an addition of all the words in the data block and then stores the resulting sum in the last location, the block check character (BCC). This

Word 1
Word 2
Word 3
⋮
Last Word
Checksum
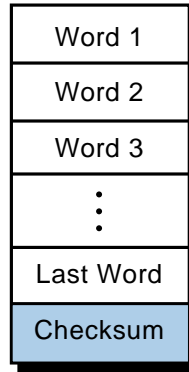
Figure 4-17. Block check character at the end of the data block.

summation process can rapidly reach an overflow condition, so one variation of CRC allows the sum to overflow, storing only the remainder bits in the BCC word. Typically, the resulting word is complemented and written in the BCC location. During the error check, all words in the block are added together, with the addition of the final BCC word turning the result to 0. A zero sum indicates a valid block. Another type of CRC generates the BCC using the remainder of dividing the sum by a preset binary number.

Longitudinal Redundancy Check. **Longitudinal redundancy check (LRC)** is an error-checking technique based on the accumulation of the result of performing an **exclusive-OR (XOR)** on each of the words in the data block. The exclusive-OR operation is similar to the standard OR logic operation (see Chapter 3) except that, with two inputs, only one can be ON (1) for the output to be 1. If both logic inputs are 1, then the output will be 0. The exclusive-OR operation is represented by the $\oplus$ symbol. Figure 4-18 illustrates the truth table for the exclusive-OR operation. Thus, the LRC operation is simply the logical exclusive-OR of the first word with the second word, the result with the third word, and so on. The final exclusive-OR operation is stored at the end of the block as the BCC.

Figure 4-18. Truth table for the exclusive-OR operation.

Exclusive-OR Truth Table		
Inputs		**Output**
A	**B**	**Y**
0	0	0
0	1	1
1	0	1
1	1	0

Cyclic Exclusive-OR Checksum. **Cyclic exclusive-OR checksum (CX-ORC)** is similar to LRC with some slight variations. The operation starts with a checksum word containing 0s, which is XORed with the first word of the block. This is followed by a left rotation of the bits in the checksum word. The next word in the data block is XORed with the checksum word and then

rotated left (see Figure 4-19). This procedure is repeated until the last word of the block has been logically operated on. The checksum word is then appended to the block to become the BCC.

A software routine in the executive program performs most checksum error-detecting methods. Typically, the processor performs the checksum computation on memory at power-up and also during the transmission of data. Some controllers perform the checksum on memory during the execution of the control program. This continuous on-line error checking lessens the possibility of the processor using invalid data.

Bit	7	6	5	4	3	2	1	0	
Data	1	0	1	1	0	1	0	1	Data Before Rotation

| | 0 | 1 | 1 | 0 | 1 | 0 | 1 | | 1 Data During Rotation |

Bit 7 Rotates to Bit 0 Position

| | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | Data After Rotation |
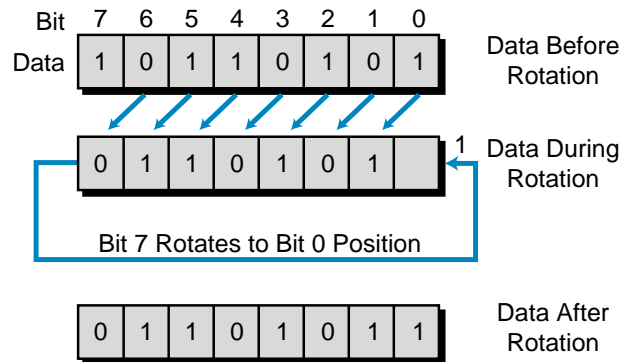
Figure 4-19. Cyclic exclusive-OR checksum operation.

EXAMPLE 4-3

Implement a checksum utilizing **(a)** LRC and **(b)** CX-ORC techniques for the four, 6-bit words shown. Place the BCC at the end of the data block.

```
word 1          1 1 0 0 1 1
word 2          1 0 1 1 0 1
word 3          1 0 1 1 1 0
word 4          1 0 0 1 1 1
```

SOLUTION

(a) Longitudinal redundancy check:

```
word 1          1 1 0 0 1 1
  ⊕                ⊕
word 2          1 0 1 1 0 1
result          0 1 1 1 1 0

  ⊕                ⊕
word 3          1 0 1 1 1 0
result          1 1 0 0 0 0
  ⊕                ⊕
word 4          1 0 0 1 1 1
result          0 1 0 1 1 1
```

LRC data block:

word 1	1 1 0 0 1 1
word 2	1 0 1 1 0 1
word 3	1 0 1 1 1 0
word 4	1 0 0 1 1 1
BCC	0 1 0 1 1 1

(b) Cyclic exclusive-OR check:

Start with checksum word 000000.

CS start	0 0 0 0 0 0
⊕	⊕
<u>word 1</u>	<u>1 1 0 0 1 1</u>
result	1 1 0 0 1 1
left rotate	1 0 0 1 1 1
⊕	⊕
<u>word 2</u>	<u>1 0 1 1 0 1</u>
result	0 0 1 0 1 0
left rotate	0 1 0 1 0 0
⊕	⊕
<u>word 3</u>	<u>1 0 1 1 1 0</u>
result	1 1 1 0 1 0
left rotate	1 1 0 1 0 1
⊕	⊕
<u>word 4</u>	<u>1 0 0 1 1 1</u>
result	0 1 0 0 1 0
left rotate	1 0 0 1 0 0 (final checksum)

CX-ORC data block:

word 1	1 1 0 0 1 1
word 2	1 0 1 1 0 1
word 3	1 0 1 1 1 0
word 4	1 0 0 1 1 1
BCC	1 0 0 1 0 0

Error Detection and Correction. More sophisticated programmable controllers may have an error detection and correction scheme that provides greater reliability than conventional error detection. The key to this type of error correction is the multiple representation of the same value.

The most common error-detecting and error-correcting code is the **Hamming code**. This code relies on parity bits interspersed with data bits in a data word. By combining the parity and data bits according to a strict set of parity equations, a small byte is generated that contains a value that identifies the erroneous bit. An error can be detected and corrected if any bit is changed by any value. The hardware used to generate and check Hamming codes is quite complex and essentially implements a set of error-correcting equations.

Error-correcting codes offer the advantage of being able to detect two or more bit errors; however, they can only correct one-bit errors. They also present a disadvantage because they are bit wasteful. Nevertheless, this scheme will continue to be used with data communication in hierarchical systems that are unmanned, sophisticated, and automatic.

CPU DIAGNOSTICS

The processor is responsible for detecting communication failures, as well as other failures, that may occur during system operation. It must alert the operator or system in case of a malfunction. To do this, the processor performs **diagnostics**, or error checks, during its operation and sends status information to indicators that are normally located on the front of the CPU.

Typical diagnostics include *memory OK*, *processor OK*, *battery OK*, and *power supply OK*. Some controllers possess a set of fault relay contacts that can be used in an alarm circuit to signal a failure. The processor controls the fault relay and activates it when one or more specific fault conditions occur.

The relay contacts that are usually provided with a controller operate in a *watchdog timer* fashion; that is, the processor sends a pulse at the end of each scan indicating a correct system operation. If a failure occurs, the processor does not send a pulse, the timer times out, and the fault relay activates.

In some controllers, CPU diagnostics are available to the user during the execution of the control program. These diagnostics use internal outputs that are controlled by the processor but can be used by the user program (e.g., loss of scan, backup battery low, etc.).

4-5 THE SYSTEM POWER SUPPLY

The system power supply plays a major role in the total system operation. In fact, it can be considered the "first-line manager" of system reliability and integrity. Its responsibility is not only to provide internal DC voltages to the system components (i.e., processor, memory, and input/output interfaces), but also to monitor and regulate the supplied voltages and warn the CPU if something is wrong. The power supply, then, has the function of supplying well-regulated power and protection for other system components.

THE INPUT VOLTAGE

Usually, PLC power supplies require input from an AC power source; however, some PLCs will accept a DC power source. Those that will accept a DC source are quite appealing for applications such as offshore drilling operations, where DC sources are commonly used. Most PLCs, however, require a 120 VAC or 220 VAC power source, while a few controllers will accept 24 VDC.

Since industrial facilities normally experience fluctuations in line voltage and frequency, a PLC power supply must be able to tolerate a 10 to 15% variation in line voltage conditions. For example, when connected to a 120 VAC source, a power supply with a line voltage tolerance of ±10% will continue to function properly as long as the voltage remains between 108 and 132 VAC. A 220 VAC power supply with ±10% line tolerance will function properly as long as the voltage remains between 198 and 242 VAC. When the line voltage exceeds the upper or lower tolerance limits for a specified duration (usually one to three AC cycles), most power supplies will issue a shutdown command to the processor. Line voltage variations in some plants can eventually become disruptive and may result in frequent loss of production. Normally, in such a case, a constant voltage transformer is installed to stabilize line conditions.

Constant Voltage Transformers. Good power supplies tolerate normal fluctuations in line conditions, but even the best-designed power supply cannot compensate for the especially unstable line voltage conditions found in some industrial environments. Conditions that cause line voltage to drop below proper levels vary depending on application and plant location. Some possible conditions are:

- start-up/shutdown of nearby heavy equipment, such as large motors, pumps, welders, compressors, and air-conditioning units

- natural line losses that vary with distance from utility substations

- intraplant line losses caused by poorly made connections

- brownout situations in which line voltage is intentionally reduced by the utility company

A **constant voltage transformer** compensates for voltage changes at its input (the primary) to maintain a steady voltage to its output (the secondary). When operated at less than the rated load, the transformer can be expected to maintain approximately ±1% output voltage regulation with an input voltage variation of as much as 15%. The percentage of regulation changes as a function of the operated load (PLC power supply and input devices)—the higher the load, the more fluctuation. Therefore, a constant voltage trans-

former must be properly rated to provide ample power to the load. The rating of the constant voltage transformer, in units of volt-amperes (VA), should be selected based on the worst-case power requirements of the load. The recommended rating for a constant voltage transformer can be obtained from the PLC manufacturer. Figure 4-20 illustrates a simplified connection of a constant voltage transformer and a programmable controller.
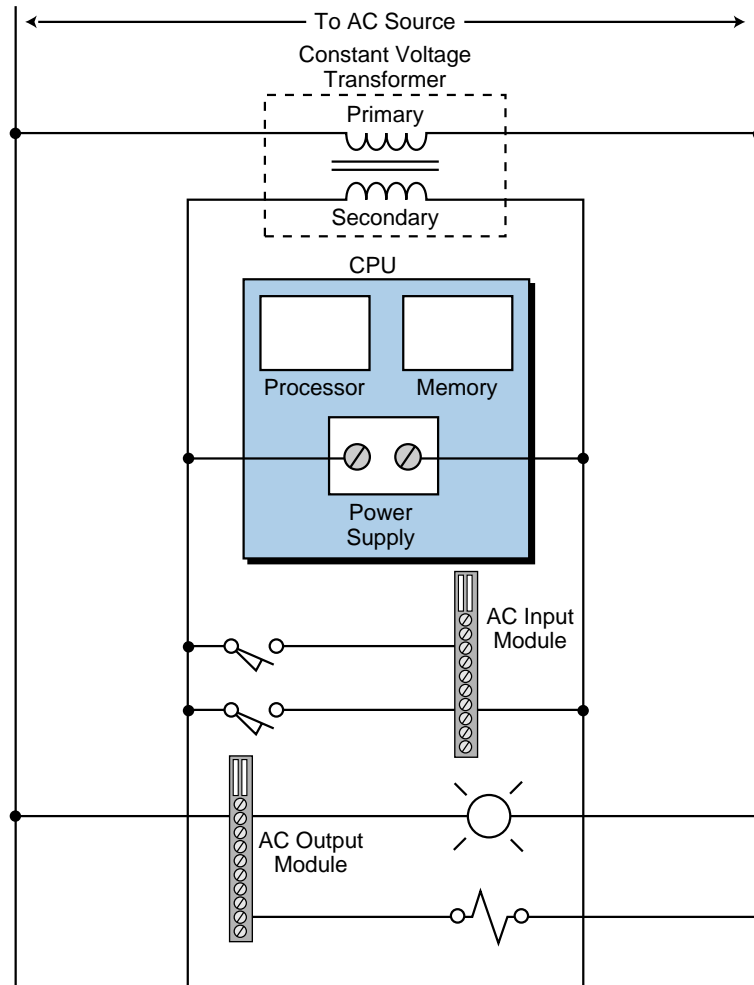
Figure 4-20. A constant voltage transformer connected to a PLC system (CPU and modules).

The Sola CVS standard sinusoidal transformer, or an equivalent constant voltage transformer, is suitable for programmable controller applications. This type of transformer uses line filters to remove high-harmonic content and provide a clean sinusoidal output. Constant voltage transformers that do not filter high harmonics are not recommended for programmable controller applications. Figure 4-21 illustrates the relationship between the output voltage and input voltage for a typical Sola CVS transformer operated at different loads.
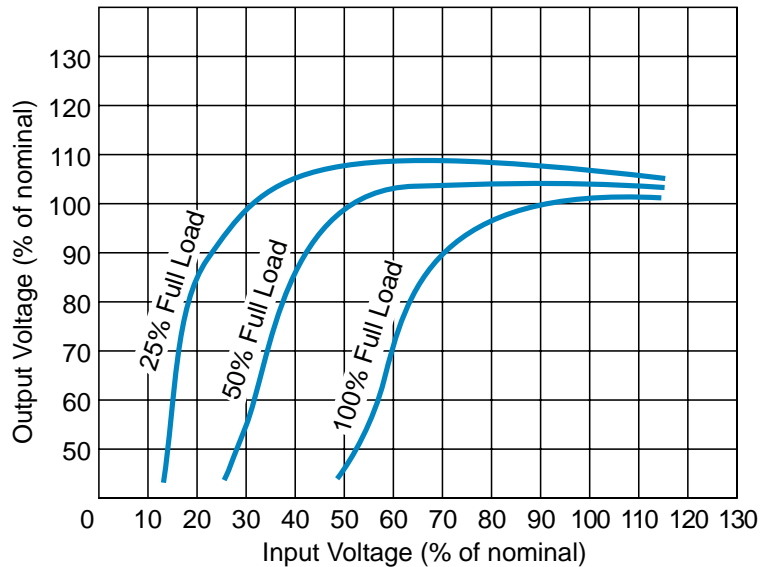
Figure 4-21. Relationship of input versus output voltages for a Sola unit.

Isolation Transformers. Often, a programmable controller will be installed in an area where the AC line is stable; however, surrounding equipment may generate considerable amounts of electromagnetic interference (EMI). Such an installation can result in intermittent misoperation of the controller, especially if the controller is not electrically isolated (on a separate AC power source) from the equipment generating the EMI. Placing the controller on a separate **isolation transformer** from the potential EMI generators will increase system reliability. The isolation transformer need not be a constant voltage transformer; but it should be located between the controller and the AC power source.

LOADING CONSIDERATIONS

The system power supply provides the DC power required by the logic circuits of the CPU and the I/O circuits. The power supply has a maximum amount of current that it can provide at a given voltage level (e.g., 10 amps at 5 volts), depending on the type of power supply. The amount of current that a given power supply can provide is not always sufficient to satisfy the requirements of a mix of I/O modules. In such a case, undercurrent conditions can cause unpredictable operation of the I/O system.

In most circumstances, an undercurrent situation is unusual, since most power supplies are designed to accommodate a mix of the most commonly used I/O modules. However, an undercurrent condition sometimes arises in applications where an excessive number of special purpose I/O modules are used (e.g., power contact outputs and analog inputs/outputs). These special purpose modules usually have higher current requirements than most commonly used digital I/O modules.

Power supply overloading can be an especially annoying condition, since the problem is not always easily detected. An overload condition is often a function of a combination of outputs that are ON at a given time, which means that overload conditions can appear intermittently. When power supply loading limits have been exceeded and overload occurs, the normal remedy is to either add an auxiliary power supply or to obtain a supply with a larger current capability. To be aware of system loading requirements ahead of time, users can obtain vendor specifications for I/O module current requirements. This information should include per point (single input or output) requirements and current requirements for both ON and OFF states. If the total current requirement for a particular I/O configuration is greater than the total current supplied by the power supply, then a second power supply will be required. An early consideration of line conditions and power requirements will help to avoid problems during installation and start-up.

Power Supply Loading Example. Undoubtedly, the best solution to a problem is anticipation of the problem. When selecting power supplies, current loading requirements, which can indicate potential loading problems, are often overlooked. For this reason, let's go over a load estimation example.

Consider an application where a PLC will control 50 discrete inputs and 25 discrete outputs. Each discrete input module can connect up to 16 field devices, while each output module can connect up to 8 field devices. In addition to this discrete configuration, the application requires a special servo motor interface module and five power contact outputs. The system also uses three analog inputs and three analog outputs.

Figure 4-22 illustrates the configuration of this PLC application. The first plug-in module is the power supply, then the processor module, and then the I/O modules.

Slot 00 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

| power supply | processor | contact output | digital output | digital output | digital output | digital output | digital input | digital input | digital input | digital input | servo motor | analog input | analog output | | | | |
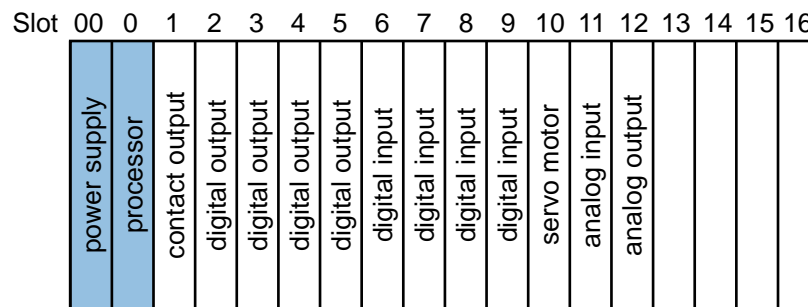
Application Note

Power supply requires one slot (slot 00).
Processor requires one slot (slot 0).
Twelve I/O slots are used, four are spare.
Auxiliary power supplies, if required, must be placed in slot 8.

Figure 4-22. Configuration of an example PLC.

The first step in estimating the load is to determine how many modules are required and then compute the total current requirement of these modules. Table 4-1 lists the module types, current requirements for all inputs and outputs ON at the same time, and the available power supplies for our programmable controller example.

Module Type	I/O Devices Connected	Connections per Module	# of Modules Required	Module Current @ On State	Total Current Required
Discrete in	50	16	4	250 mA	1000 mA
Discrete out	25	8	4	220 mA	880 mA
Contact	5	4	1	575 mA	575 mA
Analog in	3	4	1	600 mA	600 mA
Analog out	3	4	1	1200 mA	1200 mA
Servo motor	1	1	1	400 mA	400 mA
				TOTAL	4655 mA

Processor's current:	1.2 amps	
Power supplies available:	Type A	3 amps
	Type B	5 amps
	Type C	6 amps
Auxiliary power supply:	Type AA	3 amps
(placement in slot 8)	Type BB	5 amps

Table 4-1. Listing of modules and their current requirements.

The total power supply current required by this input/output system is 4655 mA, or 4.655 amps. Adding this current to the 1.2 amps required by the processor results in a total of 5.855 amps, the minimum current the power supply must provide to ensure the proper operation of the system. This total current indicates a worst-case condition, since it assumes that all I/Os are operating in the ON condition (which requires more current than the OFF condition).

For this example, there are several power supply options. These options include using a 6 amp power supply or using a combination of a smaller supply with an auxiliary source. If no expansion is expected, the 6 amp power source will suffice. Conversely, if there is a slight possibility for more I/O requirements, then an auxiliary supply will most likely be needed. The addition of an auxiliary supply can be done either at setup or when required; however, for the controller configuration in Figure 4-22, the auxiliary source must be placed in the eighth slot, resulting in I/O address changes if the auxiliary supply is added after setup. Therefore, the reference addresses in the program will have to be reprogrammed to reflect this change. Also, remember that the larger the power supply, the higher the price in most cases. You must keep all these factors in mind when configuring a PLC system and assigning I/O addresses to field devices.

<div style="background:navy;color:white">

4-6 PROGRAMMING DEVICES

</div>

Although the way to enter the control program into the PLC has changed since the first PLCs came onto the market, PLC manufacturers have always maintained an easy human interface for program entry. This means that users do not have to spend much time learning how to enter a program, but rather they can spend their time programming and solving the control problem.

Most PLCs are programmed using very similar instructions. The only difference may be the mechanics associated with entering the program into the PLC, which may vary from manufacturer to manufacturer. This involves both the type of instruction used by each particular PLC and the methodology for entering the instruction using a programming device. The two basic types of programming devices are:

- miniprogrammers
- personal computers

MINIPROGRAMMERS

Miniprogrammers, also known as *handheld* or *manual programmers*, are an inexpensive and portable way to program small PLCs (up to 128 I/O). Physically, these devices resemble handheld calculators, but they have a larger display and a somewhat different keyboard. The type of display is usually LED (light-emitting diode) or dot matrix LCD (liquid crystal display), and the keyboard consists of numeric keys, programming instruction keys, and special function keys. Instead of handheld units, some controllers have built-in miniprogrammers. In some instances, these built-in programmers are detachable from the PLC. Even though they are used mainly for editing and inputting control programs, miniprogrammers can also be useful tools for starting up, changing, and monitoring the control logic. Figure 4-23 shows a typical miniprogrammer along with a small PLC, in which miniprogrammers are generally used.

Most miniprogrammers are designed so that they are compatible with two or more controllers in a product family. The miniprogrammer is most often used with the smallest member of the PLC family or, in some cases, with the next larger member, which is normally programmed using a personal computer with special PLC programming software (discussed in the next section). With this programming option, small changes or monitoring required by the larger controller can be accomplished without carrying a personal computer to the PLC location.

Miniprogrammers can be intelligent or nonintelligent. Nonintelligent handheld programmers can be used to enter and edit the PLC program with limited on-line monitoring and editing capabilities. These capabilities are

Figure 4-23. A typical miniprogrammer and a small PLC.

limited by memory and display size. Intelligent miniprogrammers are micro-processor-based and provide the user with many of the features offered by personal computers during off-line programming (disconnected from the PLC). These intelligent devices can perform system diagnostic routines (memory, communication, display, etc.) and even serve as an operator interface device that can display English messages about the controlled machine or process.

Some miniprogrammers offer removable memory cards or modules, which store a complete program that can be reloaded at any time into any member of the PLC family (see Figure 4-24). This type of storage is useful in applications where the control program of one machine needs to be duplicated and easily transferred to other machines (e.g., OEM applications).

Figure 4-24. A removable memory card for a miniprogrammer.

PERSONAL COMPUTERS

Common usage of the personal computer (PC) in our daily lives has led to the practical elimination of dedicated PLC programming devices. Due to the personal computer's general-purpose architecture and standard operating system, most PLC manufacturers and other independent suppliers provide the necessary PC software to implement ladder program entry, editing, documentation, and real-time monitoring of the PLC's control program. The large screens of PCs can show one or more ladder rungs of the control program during programming or monitoring operation (see Figure 4-25).

Figure 4-25. A PLC ladder diagram displayed on a personal computer.

Personal computers are the programming devices of choice not so much because of their PLC programming capabilities, but because PCs are usually already present at the location where the user is performing the programming. The different types of desktop, laptop, and portable PCs give the programmer flexibility—they can be used as programming devices, but they can also be used in applications other than PLC programming. For instance, a personal computer can be used to program a PLC, but it may also be connected to the PLC's local area network (see Figure 4-26) to gather and store, on a hard disk, process information that could be vital for future product enhancements. A PC can also communicate with a programmable controller through the RS-232C serial port, thus serving either as the data handler and supervisor of the PLC control or as the bridge between the PLC network and a larger computer system (see Figure 4-27).
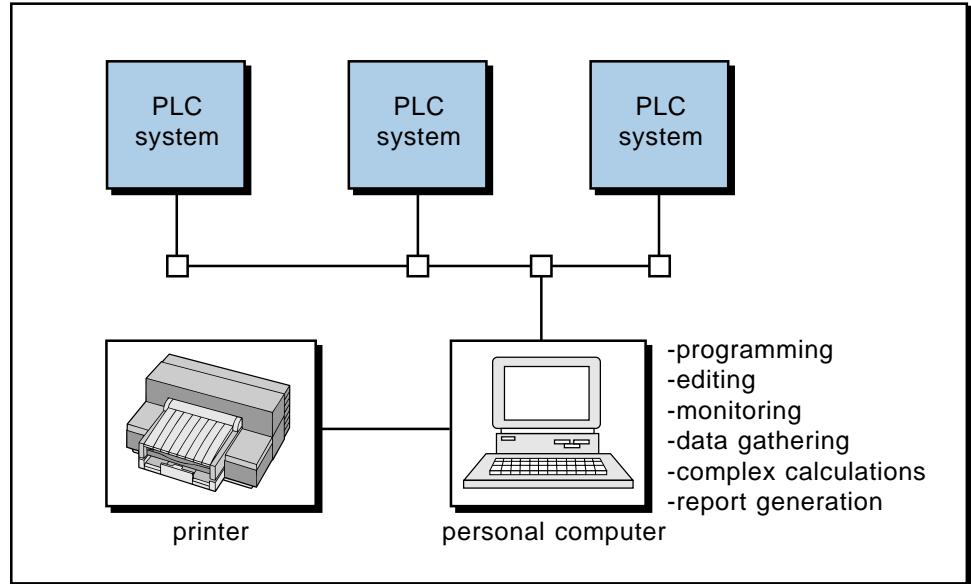
Figure 4-26. A PC connected to a PLC's local area network.
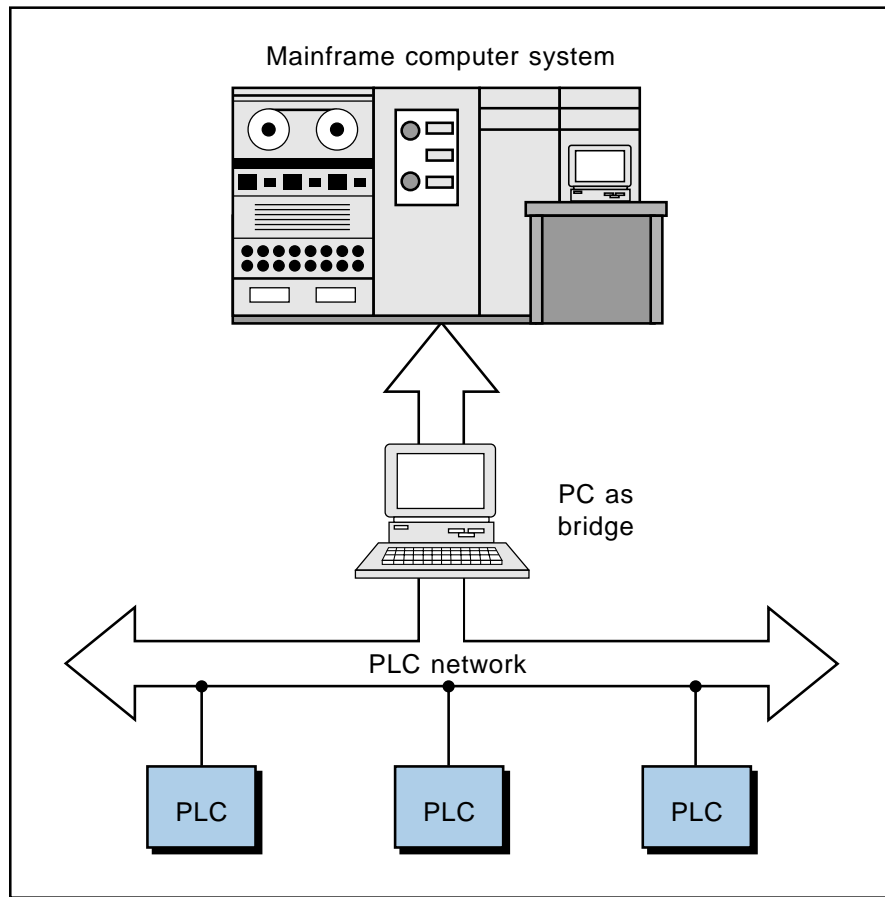
Figure 4-27. A PC acting as a bridge between a PLC network and a mainframe computer system.

In addition to programming and data collection activities, PC software that provides ladder programming capability often includes PLC documentation options. This documentation capability allows the programmer to define the purpose and function of each I/O address that is used in a PLC program. Also, general software programs, such as spreadsheets and databases, can communicate process data from the PLC to a PC via a software bridge or translator program. These software options make the PC almost invaluable when using it as a man/machine interface, providing a window to the inner workings of the PLC-controlled machine or process and generating reports that can be directly translated into management forms.

KEY TERMS

block check character (BCC)
checksum
constant voltage transformer
cyclic exclusive-OR checksum (CX-ORC)
cyclic redundancy check (CRC)
diagnostics
exclusive-OR (XOR)
Hamming code
I/O update scan
isolation transformer
longitudinal redundancy check (LRC)
microprocessor
miniprogrammer
multiprocessing
parity
parity bit
program scan
scan time
vertical redundancy check (VRC)

THE MEMORY SYSTEM AND I/O INTERACTION

The two offices of memory are collection and distribution.

—Samuel Johnson

CHAPTER HIGHLIGHTS

Now that you've learned about the first three major components of the programmable controller, it's time to learn about the last—the memory system. Understanding the PLC's memory system will help you understand why it operates as it does, as well as how it interacts with I/O interfaces. In this chapter, we will discuss the different types of memory, including memory structure and capabilities. Then, we will explore the relationship between memory organization and I/O interaction. Finally, we will explain how to configure the PLC memory for I/O addressing.

5-1 MEMORY OVERVIEW

The most important characteristic of a programmable controller is the user's ability to change the control program quickly and easily. The PLC's architecture makes this programmability feature possible. The **memory** system is the area in the PLC's CPU where all of the sequences of instructions, or *programs*, are stored and executed by the processor to provide the desired control of field devices. The memory sections that contain the control programs can be changed, or reprogrammed, to adapt to manufacturing line procedure changes or new system start-up requirements.

MEMORY SECTIONS

The total memory system in a PLC is actually composed of two different memories (see Figure 5-1):

- the executive memory

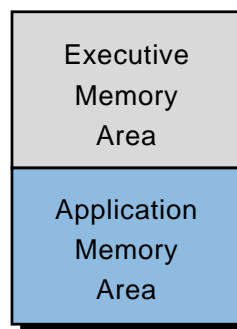- the application memory

Figure 5-1. Simplified block diagram of the total PLC memory system.

The **executive memory** is a collection of permanently stored programs that are considered part of the PLC itself. These supervisory programs direct all system activities, such as execution of the control program and communication with peripheral devices. The executive section is the part of the PLC's

memory where the system's available instruction software is stored (i.e., relay instructions, block transfer functions, math instructions, etc.). This area of memory is not accessible to the user.

The **application memory** provides a storage area for the user-programmed instructions that form the application program. The application memory area is composed of several areas, each having a specific function and usage. Section 5-4 covers the executive and application memory areas in detail.

5-2 MEMORY TYPES

The storage and retrieval requirements for the executive and application memory sections are not the same; therefore, they are not always stored in the same type of memory. For example, the executive requires a memory that permanently stores its contents and cannot be erased or altered either by loss of electrical power or by the user. This type of memory is often unsuitable for the application program.

Memory can be separated into two categories: volatile and nonvolatile. **Volatile memory** loses its programmed contents if all operating power is lost or removed, whether it is normal power or some form of backup power. Volatile memory is easily altered and quite suitable for most applications when supported by battery backup and possibly a disk copy of the program. **Nonvolatile memory** retains its programmed contents, even during a complete loss of operating power, without requiring a backup source. Nonvolatile memory generally is unalterable, yet there are special nonvolatile memory types that are alterable. Today's PLCs include those that use nonvolatile memory, those that use volatile memory with battery backup, as well as those that offer both.

There are two major concerns regarding the type of memory where the application program is stored. Since this memory is responsible for retaining the control program that will run each day, volatility should be the prime concern. Without the application program, production may be delayed or forfeited, and the outcome is usually unpleasant. A second concern should be the ease with which the program stored in memory can be altered. Ease in altering the application memory is important, since this memory is ultimately involved in any interaction between the user and the controller. This interaction begins with program entry and continues with program changes made during program generation and system start-up, along with on-line changes, such as changing timer or counter preset values.

The following discussion describes six types of memory and how their characteristics affect the manner in which programmed instructions are retained or altered within a programmable controller.

READ-ONLY MEMORY

Read-only memory (ROM) is designed to permanently store a fixed program that is not alterable under ordinary circumstances. It gets its name from the fact that its contents can be examined, or *read*, but not altered once information has been stored. This contrasts with memory types that can be read from and written to (discussed in the next section). By nature, ROMs are generally immune to alteration due to electrical noise or loss of power. Executive programs are often stored in ROM.

Programmable controllers rarely use read-only memory for their application memory. However, in applications that require fixed data, read-only memory offers advantages when speed, cost, and reliability are factors. Generally, the manufacturer creates ROM-based PLC programs at the factory. Once the manufacturer programs the original set of instructions, the user can never alter it. This typical approach to the programming of ROM-based controllers assumes that the program has already been debugged and will never be changed. This debugging is accomplished using a random-access memory–based PLC or possibly a computer. The final program is then entered into ROM. ROM application memory is typically found only in very small, dedicated PLCs.

RANDOM-ACCESS MEMORY

Random-access memory (RAM), often referred to as *read/write memory (R/W)*, is designed so that information can be written into or read from the memory storage area. Random-access memory does not retain its contents if power is lost; therefore, it is a volatile type of memory. Random-access memory normally uses a battery backup to sustain its contents in the event of a power outage.

For the most part, today's programmable controllers use RAM with battery support for application memory. Random-access memory provides an excellent means for easily creating and altering a program, as well as allowing data entry. In comparison to other memory types, RAM is a relatively fast memory. The only noticeable disadvantage of battery-supported RAM is that the battery may eventually fail, although the processor constantly monitors the status of the battery. Battery-supported RAM has proven to be sufficient for most programmable controller applications. If a battery backup is not feasible, a controller with a nonvolatile memory option (e.g., EPROM) can be used in combination with the RAM. This type of memory arrangement provides the advantages of both volatile and nonvolatile memory. Figure 5-2 shows a RAM chip.
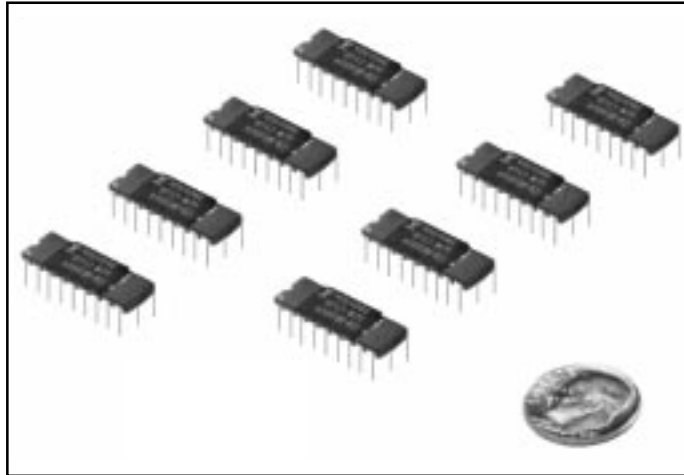
Figure 5-2. A 4K words by 8 bits RAM memory chip.

PROGRAMMABLE READ-ONLY MEMORY

Programmable read-only memory (PROM) is a special type of ROM because it can be programmed. Very few of today's programmable controllers use PROM for application memory. When it is used, this type of memory is most likely a permanent storage backup for some type of RAM. Although a PROM is programmable and, like any other ROM, has the advantage of nonvolatility, it has the disadvantage of requiring special programming equipment. Also, once programmed, it cannot be easily erased or altered; any program change requires a new set of PROM chips. A PROM memory is suitable for storing a program that has been thoroughly checked while residing in RAM and will not require further changes or on-line data entry.

ERASABLE PROGRAMMABLE READ-ONLY MEMORY

Erasable programmable read-only memory (EPROM) is a specially designed PROM that can be reprogrammed after being entirely erased by an ultraviolet (UV) light source. Complete erasure of the contents of the chip requires that the window of the chip (see Figure 5-3) be exposed to a UV light source for approximately twenty minutes. EPROM can be considered a semipermanent storage device, because it permanently stores a program until it is ready to be altered.

EPROM provides an excellent storage medium for application programs that require nonvolatility, but that do not require program changes or on-line data entry. Many OEMs use controllers with EPROM-type memories to provide permanent storage of the machine program after it has been debugged and is fully operational. OEMs use EPROM because most of their machines will not require changes or data entry by the user.
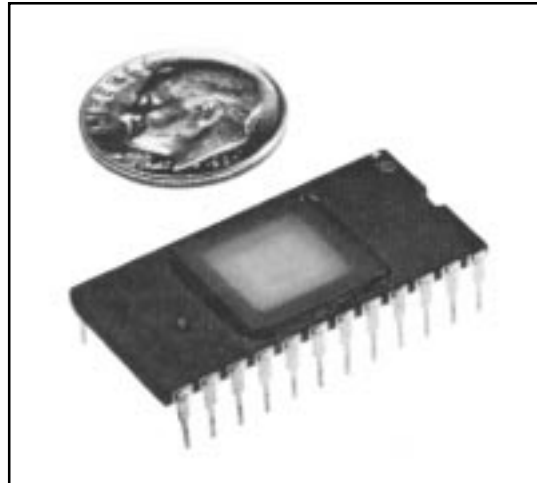
Figure 5-3. A 4K by 8 bits EPROM memory chip.

An application memory composed of EPROM alone is unsuitable if on-line changes or data entry are required. However, many controllers offer EPROM application memory as an optional backup to battery-supported RAM. EPROM, with its permanent storage capability, combined with RAM, which is easily altered, makes a suitable memory system for many applications.

ELECTRICALLY ALTERABLE READ-ONLY MEMORY

Electrically alterable read-only memory (EAROM) is similar to EPROM, but instead of requiring an ultraviolet light source to erase it, an erasing voltage on the proper pin of an EAROM chip can wipe the chip clean. Very few controllers use EAROM as application memory, but like EPROM, it provides a nonvolatile means of program storage and can be used as a backup to RAM-type memories.

ELECTRICALLY ERASABLE PROGRAMMABLE READ-ONLY MEMORY

Electrically erasable programmable read-only memory (EEPROM) is an integrated circuit memory storage device that was developed in the mid-1970s. Like ROMs and EPROMs, it is a nonvolatile memory, yet it offers the same programming flexibility as RAM does.

Several of today's small and medium-sized controllers use EEPROM as the only memory within the system. It provides permanent storage for the program and can be easily changed with the use of a programming device (e.g., a PC) or a manual programming unit. These two features help to eliminate downtime and delays associated with programming changes. They also lessen the disadvantages of electrically erasable programmable read-only memory.

One of the disadvantages of EEPROM is that a byte of memory can be written to only after it has been erased, thus creating a delay. This delay period is noticeable when on-line program changes are being made. Another disadvantage of EEPROM is a limitation on the number of times that a single byte of memory can undergo the erase/write operation (approximately 10,000). These disadvantages are negligible, however, when compared to the remarkable advantages that EEPROM offers.

5-3 MEMORY STRUCTURE AND CAPACITY

BASIC STRUCTURAL UNITS

PLC memories can be thought of as large, two-dimensional arrays of single-unit storage cells, each storing a single piece of information in the form of 1 or 0 (i.e., the binary numbering format). Since each cell can store only one binary digit and *bit* is the acronym for "*bi*nary digi*t*," each cell is called a bit. A bit, then, is the smallest structural unit of memory. Although each bit stores information as either a 1 or a 0, the memory cells do not actually contain the numbers 1 and 0 per se. Rather, the cells use voltage charges to represent 1 and 0—the presence of a voltage charge represents a 1, the absence of a charge represents a 0. A bit is considered to be ON if the stored information is 1 (voltage present) and OFF if the stored information is 0 (voltage absent). The ON/OFF information stored in a single bit is referred to as the *bit status*.

Sometimes, a processor must handle more than a single bit of data at a time. For example, it is more efficient for a processor to work with a group of bits when transferring data to and from memory. Also, storing numbers and codes requires a grouping of bits. A group of bits handled simultaneously is called a *byte*. More accurately, a byte is the smallest group of bits that can be handled by the processor at one time. Although byte size is normally eight bits, this size can vary depending on the specific controller.

The third and final structural information unit used within a PLC is a *word*. In general, a word is the unit that the processor uses when data is to be operated on or instructions are to be performed. Like a byte, a word is also a fixed group of bits that varies according to the controller; however, words are usually one byte or more in length. For example, a 16-bit word consists of two bytes. Typical word lengths used in PLCs are 8, 16, and 32 bits. Figure 5-4 illustrates the structural units of a typical programmable controller memory.
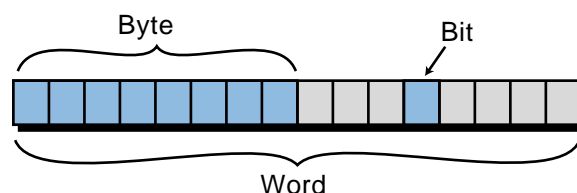

Figure 5-4. Units of PLC memory: bits, bytes, and words.

MEMORY CAPACITY AND UTILIZATION

Memory capacity is a vital concern when considering a PLC application. Specifying the right amount of memory can save the costs of hardware and time associated with adding additional memory capacity later. Knowing memory capacity requirements ahead of time also helps avoid the purchase of a controller that does not have adequate capacity or that is not expandable.

Memory capacity is nonexpandable in small controllers (less than 64 I/O capacity) and expandable in larger PLCs. Small PLCs have a fixed amount of memory because the available memory is usually more than enough to provide program storage for small applications. Larger controllers allow memory expandability, since the scope of their applications and the number of their I/O devices have less definition.

Application memory size is specified in terms of K units, where each K unit represents 1024 word locations. A 1K memory, then, contains 1024 storage locations, a 2K memory contains 2048 locations, a 4K memory contains 4096 locations, and so on. Figure 5-5 illustrates two memory arrays of 4K each; however, they have different configurations—the first configuration uses one-byte words (8 bits) and the other uses two-byte words (16 bits).
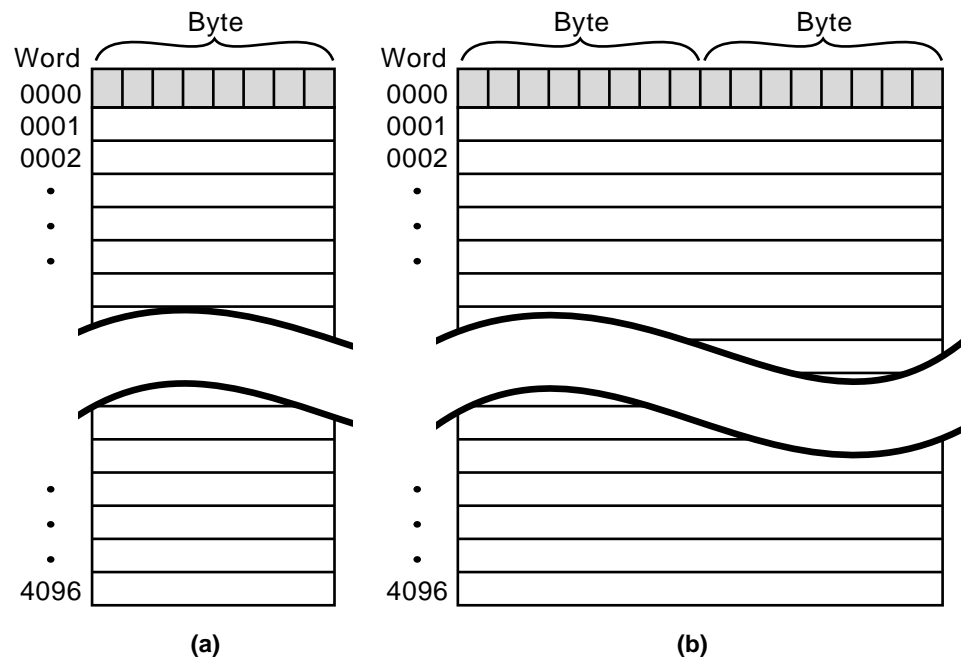

Figure 5-5. Block illustration of **(a)** a 4K by 8 bits storage location and **(b)** a 4K by 16 bits storage location.

The memory capacity of a programmable controller in units of K is only an indication of the total number of storage locations available. Knowing this maximum number alone is not enough to determine memory requirements.

Additional information concerning how program instructions are stored will help to make a better decision. The term *memory utilization* refers to the amount of data that can be stored in one location or, more specifically, to the number of memory locations required to store each type of instruction. The manufacturer can supply this data if the product literature does not provide it.

To illustrate memory capacity, let's refer to Figure 5-5. Suppose that each normally open and normally closed contact instruction requires 16 bits of storage area. With these memory requirements, the effective storage area of the memory system in Figure 5-5a is half that of Figure 5-5b. This means that, to store the same size control program, the system in Figure 5-5a would require 8K memory capacity instead of 4K, as in Figure 5-5b.

After becoming familiar with how memory is utilized in a particular controller, users can begin to determine the maximum memory requirements for an application. Although several rules of thumb have been used over the years, no one simple rule has emerged as being the most accurate. However, with a knowledge of the number of outputs, an idea of the number of program contacts needed to drive the logic of each output, and information concerning memory utilization, memory requirement approximation can be reduced to simple multiplication.

EXAMPLE 5-1

Determine the memory requirements for an application with the following specifications:

- 70 outputs, with each output driven by logic composed of 10 contact elements

- 11 timers and 3 counters, each having 8 and 5 elements, respectively

- 20 instructions that include addition, subtraction, and comparison, each driven by 5 contact elements

Table 5-1 provides information about the application's memory utilization requirements.

Instruction	Words of Memory Required
Examine ON or OFF (contacts)	1
Output coil	1
Add/subtract/compare	1
Timer/counter	3

Table 5-1. Memory utilization requirements.

SOLUTION

Using the given information, a preliminary estimation of memory is:

(a) Control logic = 10 contact elements/output rung
Number of output rungs = 70

(b) Control logic = 8 contact elements/timer
Number of timers = 11

(c) Control logic = 5 contact elements/counter
Number of counters = 3

(d) Control logic = 5 contact elements/math and compare
Number of math and compare = 20

Based on the memory utilization information from Table 5-1, the total number of words is:

(a)	Total contact elements	(70 x 10)	700
	Total outputs	(70 x 1)	70
	Total words		770
(b)	Total contact elements	(11 x 8)	88
	Total timers	(11 x 3)	33
	Total words		121
(c)	Total contact elements	(3 x 5)	15
	Total counters	(3 x 3)	9
	Total words		24
(d)	Total contact elements	(20 x 5)	100
	Total math and compare	(20 x 1)	20
	Total words		120

Thus, the total words of memory required for the storage of the instructions, outputs, timers, and counters is 1035 words (770 + 121 + 24 + 120), or just over 1K of memory.

The calculation performed in the previous example is actually an approximation because other factors, such as future expansion, must be considered before the final decision is made. After determining the minimum memory requirements for an application, it is wise to add an additional 25 to 50% more memory. This increase allows for changes, modifications, and future expansion. Keep in mind that the sophistication of the control program also affects memory requirements. If the application requires data manipulation and data storage, it will require additional memory. Normally, the enhanced instructions that perform mathematical and data manipulation operations

will also have greater memory requirements. Depending on the PLC's manufacturer, the application memory may also include the data table and I/O table (discussed in the next section). If this is the case, then the amount of "real" user application memory available will be less than that specified. Exact memory usage can be determined by consulting the manufacturer's memory utilization specifications.

5-4 MEMORY ORGANIZATION AND I/O INTERACTION

The memory system, as mentioned before, is composed of two major sections—the system memory and the application memory—which in turn are composed of other areas. Figure 5-6 illustrates this memory organization, known as a **memory map**. Although the two main sections, system memory and application memory, are shown next to each other, they are not necessarily adjacent, either physically or by address. The memory map shows not only what is stored in memory, but also where data is stored, according to specific locations called *memory addresses*. An understanding of the memory map is very useful when creating a PLC control program and defining the data table.
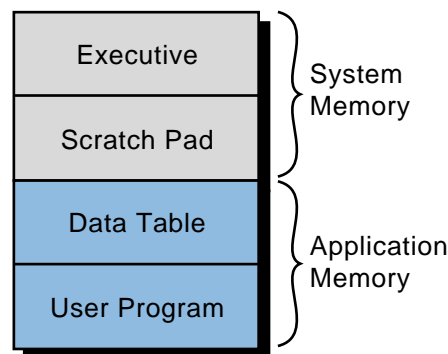

Figure 5-6. A simplified memory map.

Although two different programmable controllers rarely have identical memory maps, a generalized discussion of memory organization is still valid because all programmable controllers have similar storage requirements. In general, all PLCs must have memory allocated for four basic memory areas, which are as follows:

- **Executive Area.** The executive is a permanently stored collection of programs that are considered part of the system itself. These supervisory programs direct system activities, such as execution of the control program, communication with peripheral devices, and other system housekeeping activities.

- **Scratch Pad Area.** This is a temporary storage area used by the CPU to store a relatively small amount of data for interim calculations and control. The CPU stores data that is needed quickly in this memory area to avoid the longer access time involved with retrieving data from the main memory.

- **Data Table Area.** This area stores all data associated with the control program, such as timer/counter preset values and other stored constants and variables used by the control program or CPU. The data table also retains the status information of both the system inputs (once they have been read) and the system outputs (once they have been set by the control program).

- **User Program Area.** This area provides storage for programmed instructions entered by the user. The user program area also stores the control program.

The executive and scratch pad areas are hidden from the user and can be considered a single area of memory that, for our purpose, is called *system memory*. On the other hand, the data table and user program areas are accessible and are required by the user for control applications. They are called *application memory*.

The total memory specified for a controller may include system memory and application memory. For example, a controller with a maximum of 64K may have executive routines that use 32K and a system work area (scratch pad) of 1/4K. This arrangement leaves a total of 31 3/4K for application memory (data table and user memory). Although it is not always the case, the maximum memory specified for a given programmable controller normally includes only the total amount of application memory available. Other controllers may specify only the amount of user memory available for the control program, assuming a fixed data table area defined by the manufacturer. Now, let's take a closer look at the application memory and explore how it interacts with the user and the program.

APPLICATION MEMORY

The application memory stores programmed instructions and any data the processor will use to perform its control functions. Figure 5-7 shows a mapping of the typical elements in this area. Each programmable controller has a maximum amount of application memory, which varies depending on the size of the controller. The controller stores all data in the data table section of the application memory, while it stores programmed instructions in the user program section.
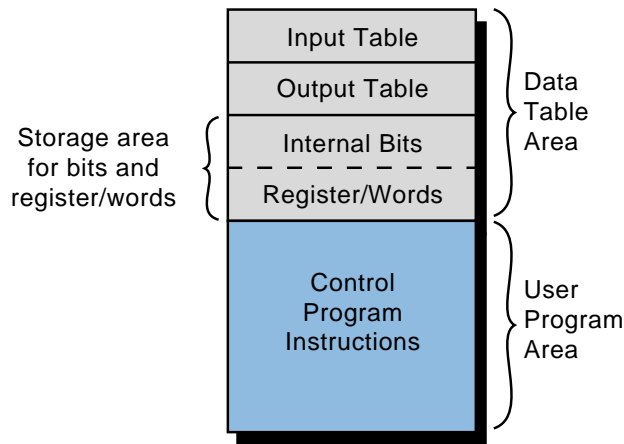
Figure 5-7. Application memory map.

Data Table Section. The data table section of a PLC's application memory is composed of several areas (see Figure 5-7). They are:

- the input table

- the output table

- the storage area

These areas contain information in binary form representing input/output status (ON or OFF), numbers, and codes. Remember that the memory structure contains cell areas, or bits, where this binary information is stored. Following is an explanation of each of the three data table areas.

Input Table. The **input table** is an array of bits that stores the status of digital inputs connected to the PLC's input interface. The maximum number of input table bits is equal to the maximum number of field inputs that can be connected to the PLC. For example, a controller with a maximum of 64 field inputs requires an input table of 64 bits. Thus, each connected input has an analogous bit in the input table, corresponding to the terminal to which the input is connected. The address of the input device is the bit and word location of its corresponding location in the input table. For example, the limit switch connected to the input interface in Figure 5-8 has an address of 13007_8 as its corresponding bit in the input table. This address comes from the word location 130_8 and the bit number 07_8, both of which are related to the module's rack position and the terminal connected to the field device (see Section 6-2). If the limit switch is OFF, the corresponding bit (13007_8) is 0 (see Figure 5-8a); if the limit switch is ON (see Figure 5-8b), the corresponding bit is 1.

During PLC operation, the processor will read the status of each input in the input module and place a value (1 or 0) in the corresponding address in the input table. The input table is constantly changing to reflect the changes of the input module and its connected field devices. These input table changes take place during the reading part of the I/O update.
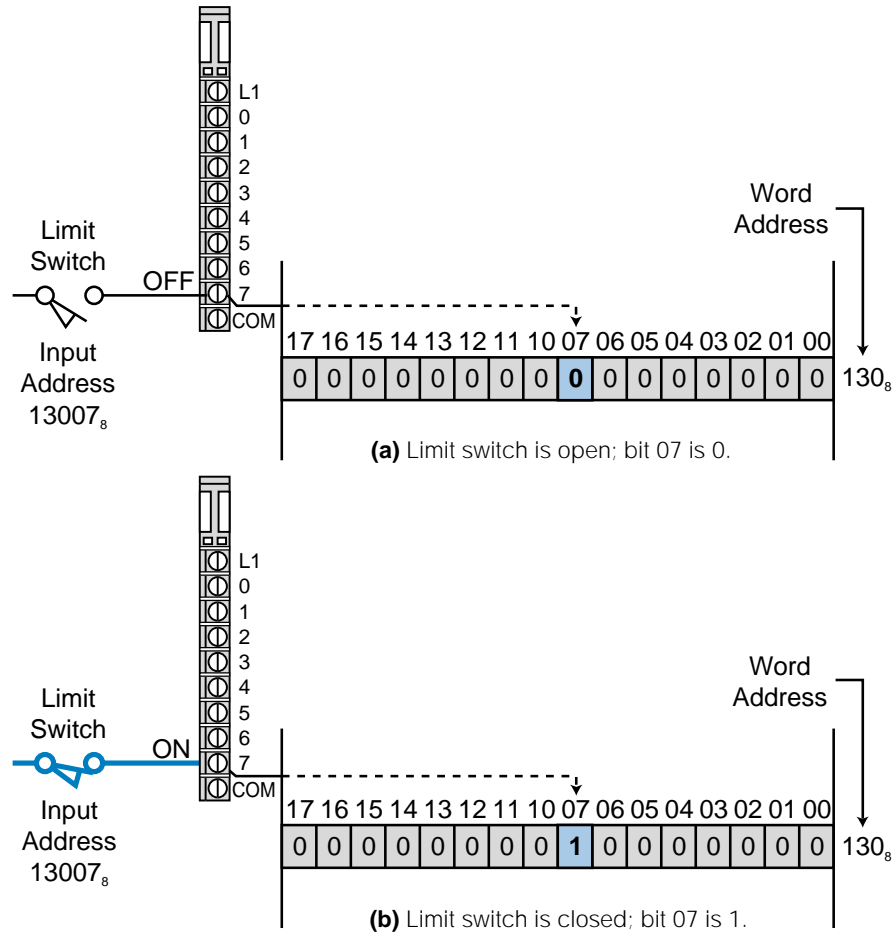
(a) Limit switch is open; bit 07 is 0.

(b) Limit switch is closed; bit 07 is 1.

Figure 5-8. Limit switch connected to a bit in the input table.

<u>Output Table.</u> The **output table** is an array of bits that controls the status of digital output devices that are connected to the PLC's output interface. The maximum number of bits available in the output table equals the maximum number of output field devices that can interface with the PLC. For example, a PLC with a maximum of 128 outputs requires an output table of 128 bits.

Like the input table, each connected output has an analogous bit in the output table corresponding to the exact terminal to which the output is connected. The processor controls the bits in the output table as it interprets the control program logic during the program scan, turning the output modules ON and OFF accordingly during the output update scan. If a bit in the table is turned ON (1), then the connected output is switched ON (see Figure 5-9a); if a bit is cleared, or turned OFF (0), the output is switched OFF (see Figure 5-9b). Remember that the turning ON and OFF of field devices via the output module occurs during the update of outputs after the end of the scan.

<u>Storage Area.</u> The purpose of the **storage area** section of the data table is to store changeable data, whether it is one bit or a word (16 bits). The storage area consists of two parts: an *internal bit storage area* and a *register/word*
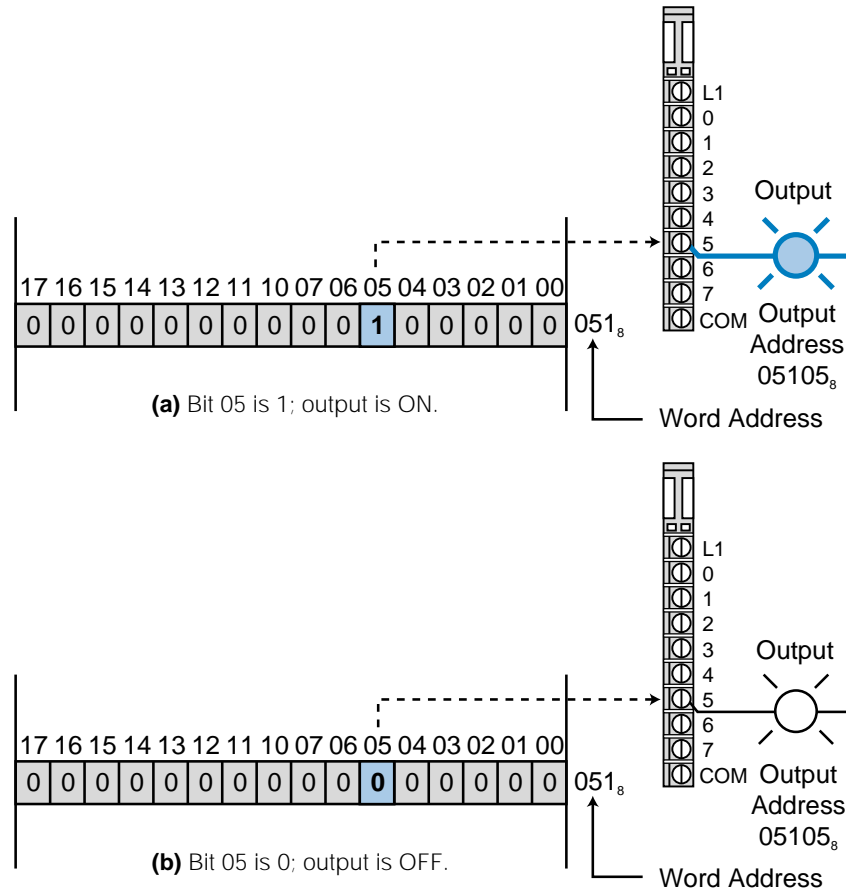
(a) Bit 05 is 1; output is ON.

(b) Bit 05 is 0; output is OFF.

Figure 5-9. Field output connected to a bit in the output table.

storage area (see Figure 5-10). The internal bit storage area contains storage bits that are referred to as either *internal outputs, internal coils, internal (control) relays,* or *internals*. These internals provide an output, for interlocking purposes, of ladder sequences in the control program. Internal outputs do not directly control output devices because they are stored in addresses that do not map the output table and, therefore, any output devices.

When the processor evaluates the control program and an internal bit is energized (1), its referenced contact (the contact with this bit address) will change state—if it is normally open, it will close; if it is normally closed, it will open. Internal contacts are used in conjunction with either other internals or "real" input contacts to form interlocking sequences that drive an output device or another internal output.

The register/word storage area is used to store groups of bits (bytes and words). This information is stored in binary format and represents quantities or codes. If decimal quantities are stored, the binary pattern of the register represents an equivalent decimal number (see Chapter 2). If a code is stored, the binary pattern represents a BCD number or an ASCII code character (one character per byte).
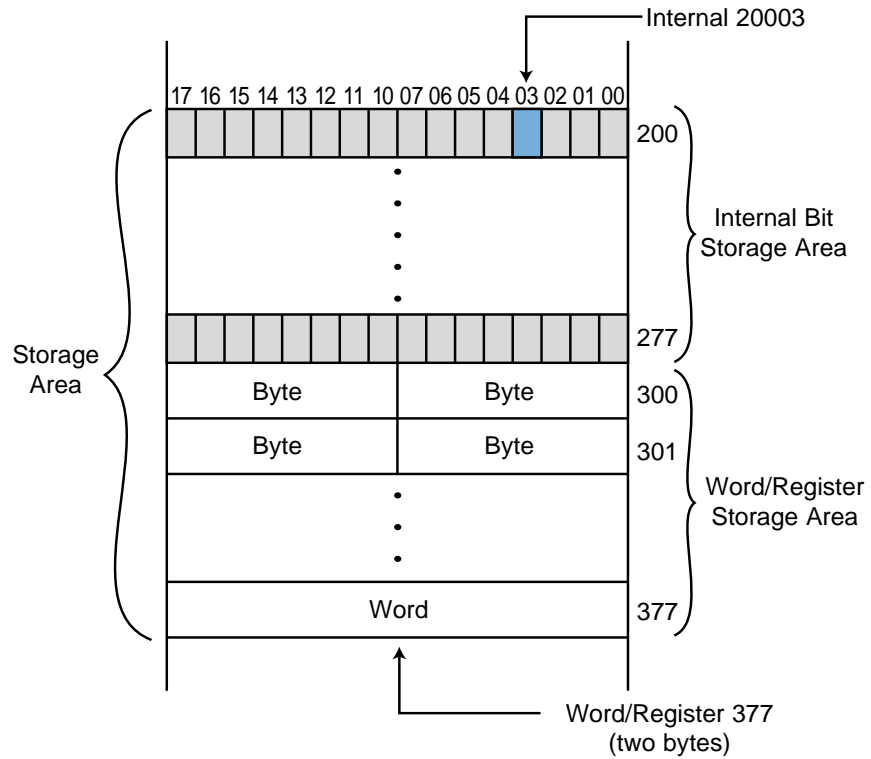
Figure 5-10. Storage area section of the data table.

Values placed in the register/word storage area represent input data from a variety of devices, such as thumbwheel switches, analog inputs, and other types of variables. In addition to input values, these registers can contain output values that are destined to go to output interface modules connected to field devices, such as analog meters, seven-segment LED indicators (BCD), control valves, and drive speed controllers. Storage registers are also used to hold fixed constants, such as preset timer/counter values, and changing values, such as arithmetic results and accumulated timer/counter values. Depending on their use, the registers in the register/word storage area may also be referred to as *input registers, output registers,* or *holding registers*. Table 5-2 shows typical constants and variables stored in these registers.

Constants	Variables
Timer preset values	Timer accumulated values
Counter preset values	Counter accumulated values
Loop control set points	Result values from math operations
Compare set points	Analog input values
Decimal tables (recipes)	Analog output values
ASCII characters	BCD inputs
ASCII messages	BCD outputs
Numerical tables	

Table 5-2. Constants and variables stored in register/word storage area registers.

EXAMPLE 5-2

Referencing Figure 5-11, what happens to internal 2301 (word 23, bit 01) when the limit switch connected to input terminal 10 closes?
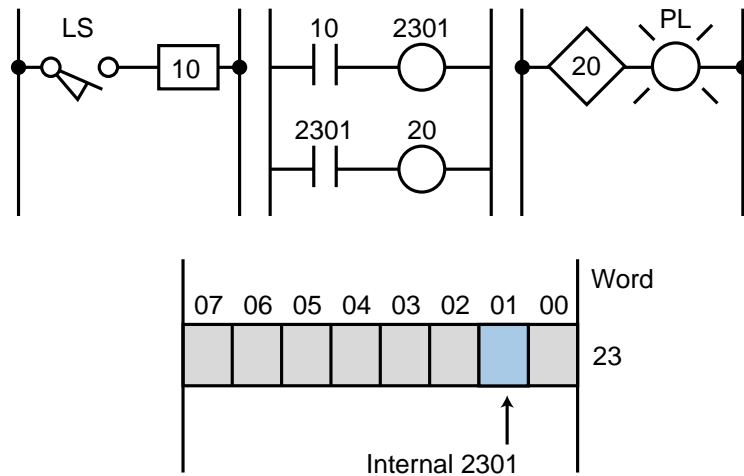

Figure 5-11. Open limit switch connected to an internal output.

SOLUTION

When LS closes (see Figure 5-12), contact 10 will close, turning internal output 2301 ON (a 1 in bit 01 of word 23). This will close contact 2301 ($\overset{2301}{\dashv\vdash}$) and turn real output 20 ON, causing the light PL to turn ON at the end of the scan.
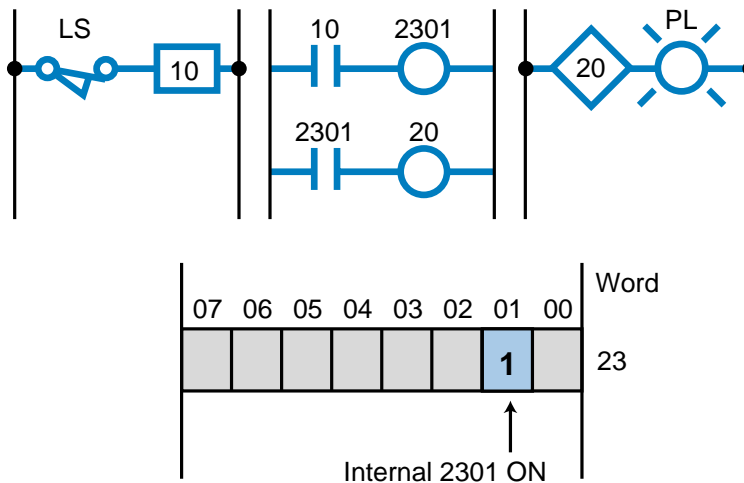
Figure 5-12. Closed limit switch connected to an internal output.

EXAMPLE 5-3

For the memory map shown in Figure 5-13, illustrate how to represent the following numbers in the storage area: **(a)** the BCD number 9876, **(b)** the ASCII character *A* (octal 101) in one byte (use lower byte), and **(c)** the analog value 2257 (1000 1101 0001 binary). Represent these values starting at register 400.
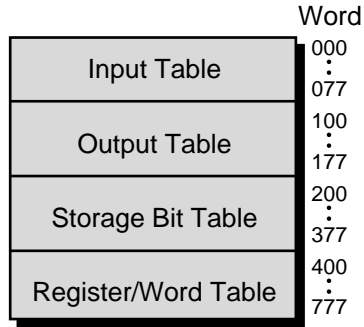

Figure 5-13. Memory map.

SOLUTION

Figure 5-14 shows the register data corresponding to the BCD number 9876, the ASCII character *A*, and the analog value 2257.
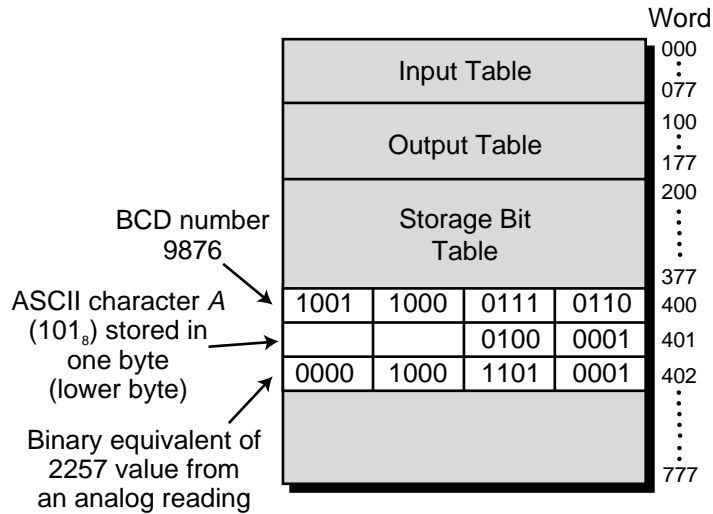

Figure 5-14. Solution for Example 5-3.

User Program Section. The user program section of the application memory is reserved for the storage of the control logic. All of the PLC instructions that control the machine or process are stored in this area. The processor's executive software language, which represents each of the PLC instructions, stores its instructions in the user program memory.

When a PLC executes its program, the processor interprets the information in the user program memory and controls the referenced bits in the data table that correspond to real or internal I/O. The processor's execution of the executive program accomplishes this interpretation of the user program.

The maximum amount of user program memory available is normally a function of the controller's size (i.e., I/O capacity). In medium and large controllers, the user program area is made flexible by altering the size of the data table so that it meets the minimum data storage requirements. In small controllers, however, the user program area is normally fixed. The amount of user program memory required is directly proportional to the number of instructions used in the control program. Estimation of user memory requirements is accomplished using the method described earlier in Section 5-3.

5-5 CONFIGURING THE PLC MEMORY—I/O ADDRESSING

Understanding memory organization, especially the interaction of the data table's I/O mapping and storage areas, helps in the comprehension of a PLC's functional operation. Although the memory map is often taken for granted by PLC users, a thorough understanding of it provides a better perception of how the control software program should be organized and developed.

DATA TABLE ORGANIZATION

The data table's organization, or *configuration* as it is sometimes called, is very important. The configuration defines not only the discrete device addresses, but also the registers that will be used for numerical and analog control, as well as basic PLC timing and counting operations. The intention of the following discussion of data table organization is not to go into detail about configuration, but to review what you have learned about the memory map, making sure that you understand how memory and I/O interact.

First, let's consider an example of an application memory map for a PLC. The controller has the following memory, I/O, and numbering system specifications:

- total application memory of 4K words with 16 bits

- capability of connecting 256 I/O devices (128 inputs and 128 outputs)

- 128 available internal outputs

- capability of up to 256 storage registers, selectable in groups of 8-word locations, with 8 being the minimum number of registers possible (32 groups of 8 registers each)

- octal (base 8) numbering system with 2-byte (16-bit) word length

To illustrate this memory map may seem unnecessary, but at this point, we do not know the starting address of the control program. This does not matter as far as the program is concerned; however, it does matter when determining the register address references to be used, since these register addresses are referred to in the control program (i.e., timer preset and accumulated values).

With this in mind, let's set the I/O table boundaries. Assuming the inputs are first in the I/O mapping, the input table will start at address 0000_8 and end at address 0007_8 (see Figure 5-15). The outputs will start at address 0010_8 and end at address 0017_8. Since each memory word has 16 bits, the 128 inputs require 8 input table words, and likewise for the outputs. The starting address for the internal output storage area is at memory location address 0020_8 and continues through address 0027_8 (8 words of 16 bits each totaling 128 internal output bits). Address 0030_8 indicates the beginning of the register/word storage area. This area must have a minimum of 8 registers, with a possibility of up to 256 registers added in 8-register increments. The first 8 required
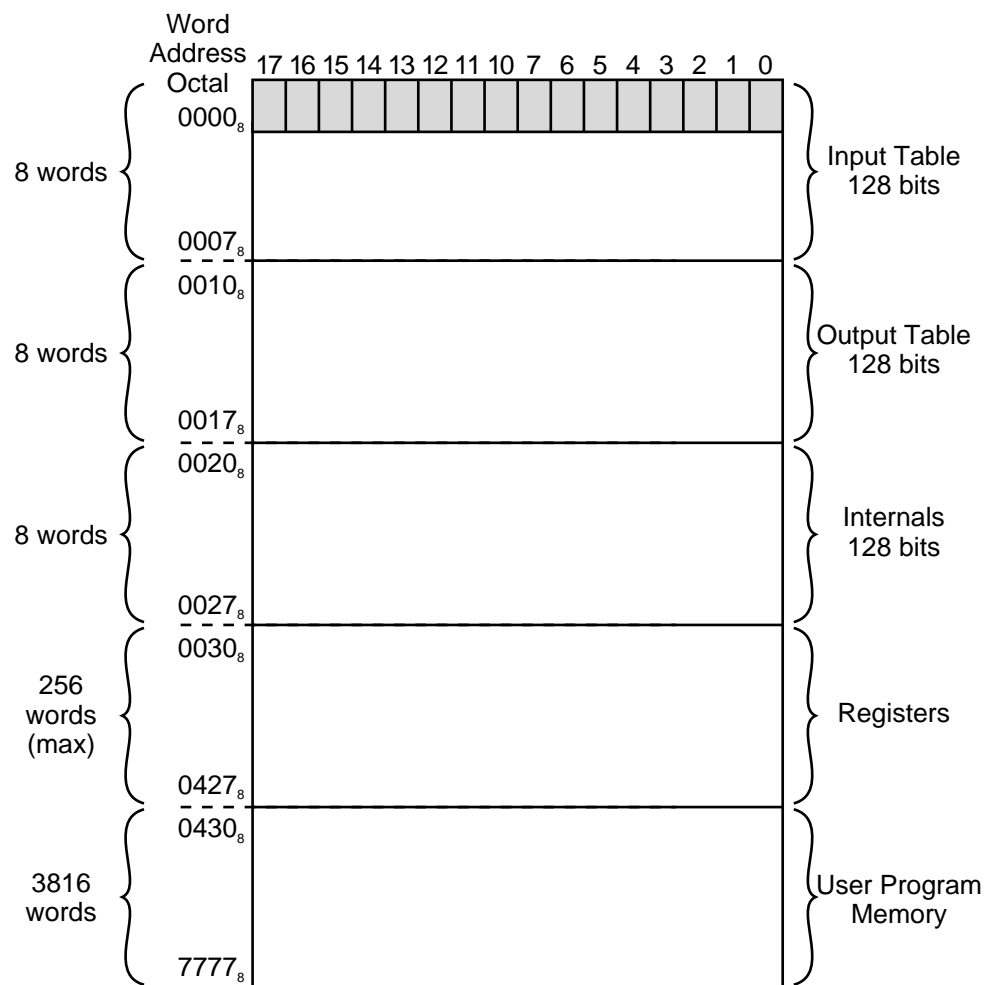

Figure 5-15. I/O table and user memory boundaries.

SECTION
2
Components
and Systems

The Memory System
and I/O Interaction
CHAPTER
5

registers, then, will end at address 0037_8 (see Figure 5-16). Any other 8-register increments will start at 0040_8, with the last possible address being 0427_8, providing a total of 256 registers.

If all available storage registers are utilized, then the starting memory address for the control program will be 0430_8. This configuration will leave 3816 (decimal) locations to store the control software. Figure 5-15 showed this maximum configuration.

Word
Address

Registers
(min)

0030

0037
0040

0047
0050

0057
0060

•
•
•
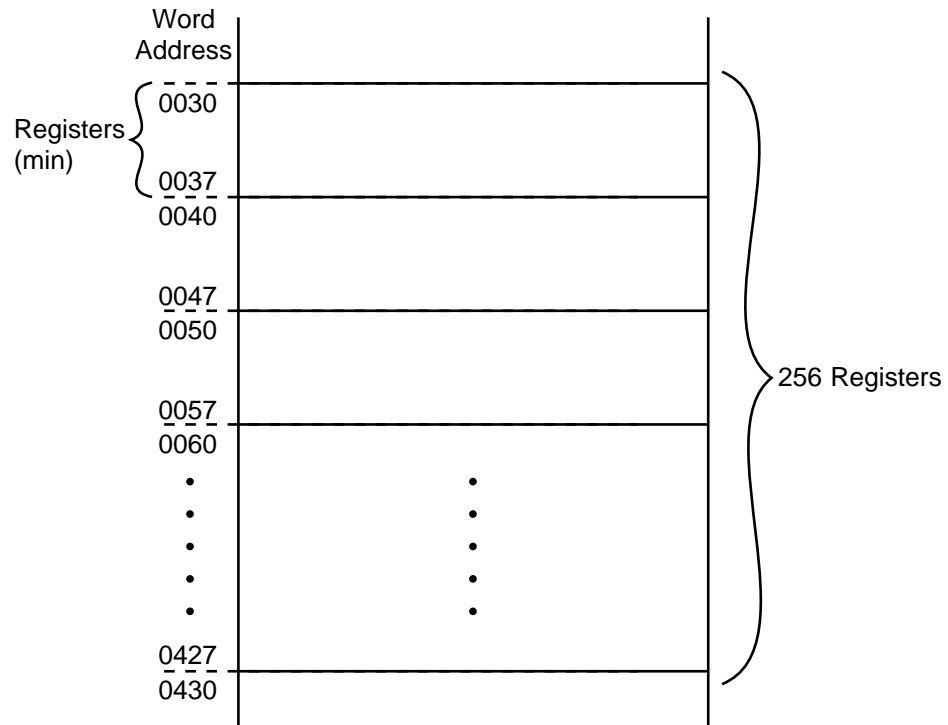•
•

0427
0430

256 Registers

Figure 5-16. Breakdown, in groups of eight, of the register storage area at its maximum capacity.

Most controllers allow the user to change the range of register boundaries without any concern for starting memory addresses of the program. Nonetheless, the user should know beforehand the number of registers needed. This will be useful when assigning register addresses in the program.

I/O ADDRESSING

Throughout this text, we have mentioned that the programmable controller's operation simply consists of reading inputs, solving the ladder logic in the user program memory, and updating the outputs. As we get more into PLC programming and the application of I/O modules, we will review the relationship between the I/O address and the I/O table, as well as how I/O addressing is used in the program.

The input/output structure of a programmable controller is designed with one thing in mind—simplicity. Input/output field devices are connected to a PLC's I/O modules, which are located in the *rack* (the physical enclosure that houses a PLC's supplementary devices). The rack location of each I/O device is then mapped to the I/O table, where the I/O module placement defines the address of the devices connected to the module. Some PLCs use internal module switches to define the addresses used by the devices connected to the module. In the end, however, all of the input and output connections are mapped to the I/O table.

Assume that a simple relay circuit contains a limit switch driving a pilot light (see Figure 5-17). This circuit is to be connected to a PLC input module and output module, as shown in Figure 5-18. For the purpose of our discussion, let's assume that each module contains 8 possible input or output channels and that the PLC has a memory map similar to the one shown previously in Figure 5-15. The limit switch is connected to the number 5 (octal) terminal of the input module, while the light is connected to the number 6 (octal) terminal of the output module.
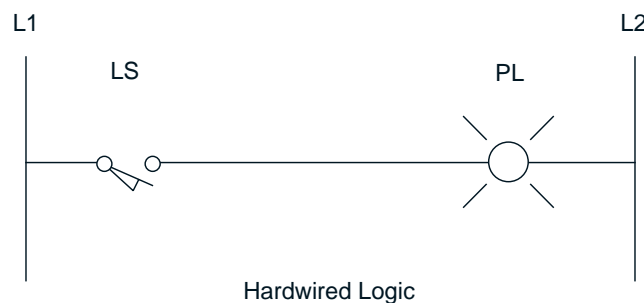

Figure 5-17. A relay circuit with a limit switch driving a pilot light.

Let's assume that, due to their placement inside the rack, the I/O modules' map addresses are word 0000 for the input module and word 0010 for the output module. Therefore, the processor will reference the limit switch as input 000005, and it will reference the light as output 001006 (i.e., the input is mapped to word 0000 bit 05, and the output is mapped to word 0010 bit 06). These addresses are mapped to the I/O table. Every time the processor reads the inputs, it will update the input table and turn ON those bits whose input devices are 1 (ON or closed). When the processor begins the execution of the ladder program, it will provide power (i.e., continuity) to the ladder element corresponding to the limit switch, because its reference address is 1 (see Figure 5-18). At this time, it will set output 001006 ON, and the pilot light will turn ON after *all* instructions have been evaluated and the end of scan (EOS)—where the output update to the module takes place—has been reached. This operation is repeated every scan, which can be as fast as every thousandth of a second (1 msec) or less.
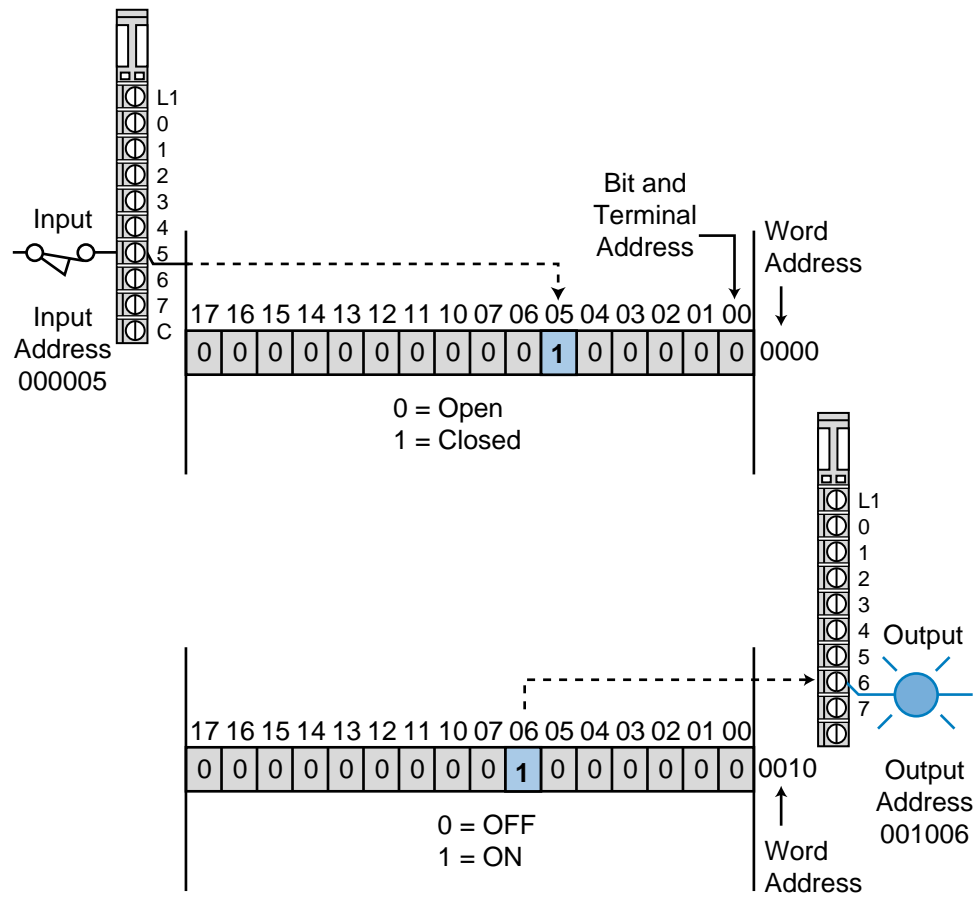
Figure 5-18. Input/output module connected to field devices.

Note that addresses 000005 and 001006 can be used as often as required in the control program. If we had programmed a contact at 001006 to drive internal output 002017 (see Figure 5-19), the controller would turn its internal output bit (002017) to 1 every time output 001006 turned ON. However, this output would not be directly connected to any output device. Note that internal storage bit 002017 is located in word 0020 bit 17.
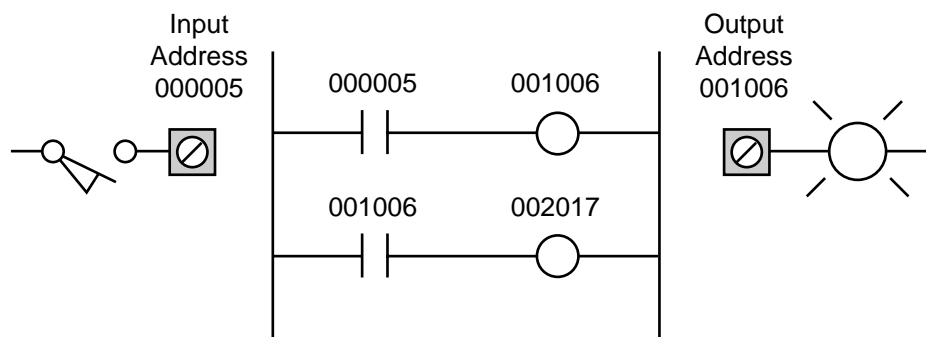

Figure 5-19. PLC ladder implementation of Figure 5-17 using an internal output bit.

5-6   SUMMARY OF MEMORY, SCANNING, AND I/O INTERACTION

So far, you have learned about scanning, memory system organization, and the interaction of input and output field devices in a programmable controller. In this section, we will present an example that summarizes these PLC operations. In this example, we will assume that we have a simple PLC memory, organized as shown in Figure 5-20, and a simple circuit (see Figure 5-21), which is connected to a PLC via I/O interfaces.
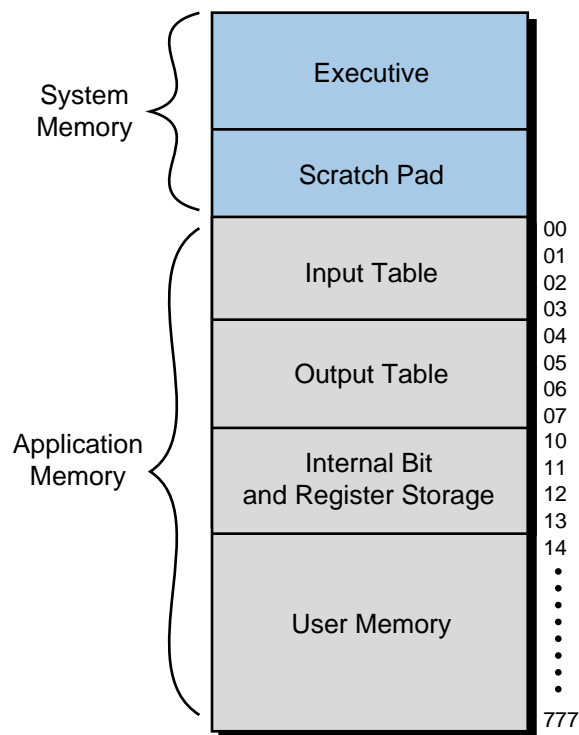

Figure 5-20. An example of a PLC memory map.

The instructions used to represent the simple control program, shown in Figure 5-22, are stored in the user memory section, where specific binary 1s and 0s represent the instructions (e.g., the $\overset{0010}{\dashv\vdash}$ instruction). During the PLC scan, the executive program reads the status of inputs and places this data into the input data table. Then, the programmable controller scans the user memory to interpret the instructions stored. As the logic is solved rung by rung according to the status of the I/O table, the results from the program evaluation are stored in the output table and the storage bit table (if the program uses internals). After the evaluation (program scan), the executive program updates the values stored in the output table and sends commands to the output modules to turn ON or OFF the field devices connected to their respective interfaces. Figure 5-23 on the page 26 shows the steps that will occur during the evaluation of the PLC circuit shown in Figure 5-22.
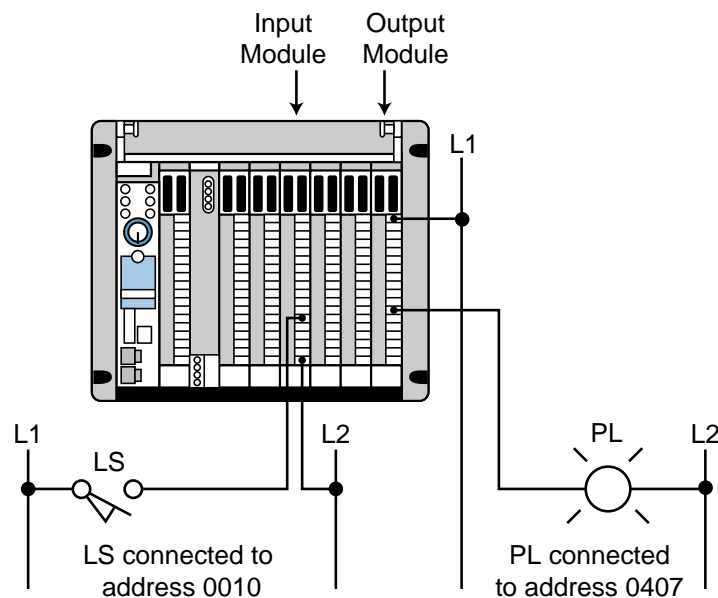
Figure 5-21. A simple circuit connected to a PLC via I/O interfaces.
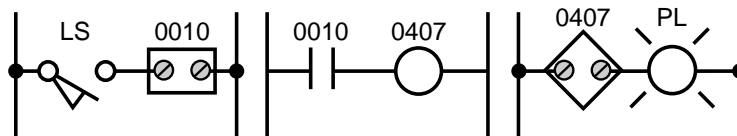

Figure 5-22. Instructions used to represent the control program.

5-7 MEMORY CONSIDERATIONS

The previous sections presented an analysis of programmable controller memory characteristics regarding memory type, storage capacity, organization, structure, and their relationship to I/O addressing. Particular emphasis was placed on the application memory, which stores the control program and data. Careful consideration must also be given to the type of memory, since certain applications require frequent changes, while others require permanent storage once the program is debugged. A RAM with battery support may be adequate in most cases, but in others, a RAM and an optional nonvolatile-type memory may be required.

It is important to remember that the total memory capacity for a particular controller may not be completely available for application programming. The specified memory capacity may include memory utilized by the executive routines or the scratch pad, as well as the user program area.

The application memory varies in size depending on the size of the controller. The total area available for the control program also varies according to the size of the data table. In small controllers, the data table is usually fixed, which
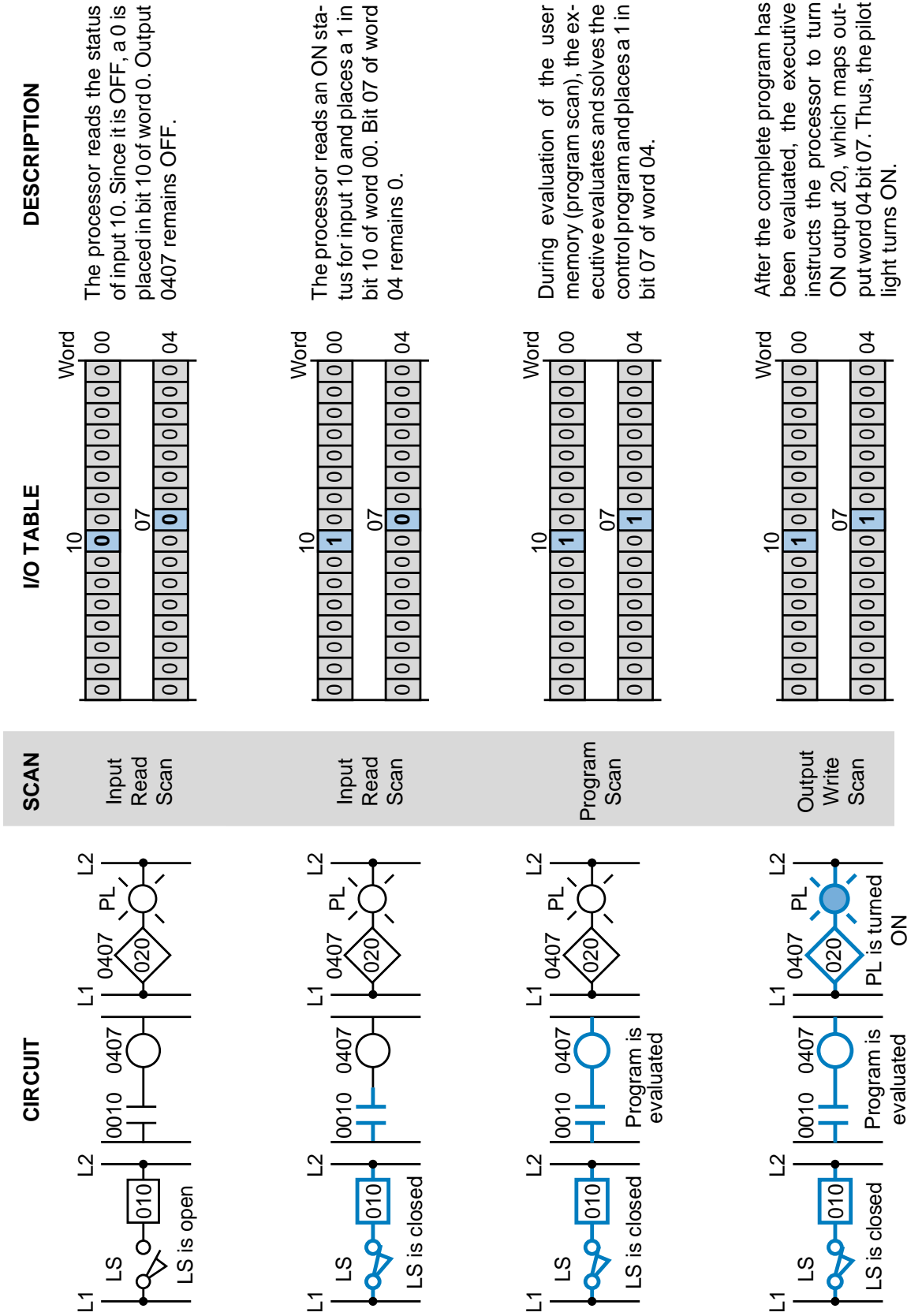
Figure 5-23. Steps in the evaluation of the PLC circuit shown in Figure 5-22.

DESCRIPTION

The processor reads the status of input 10. Since it is OFF, a 0 is placed in bit 10 of word 0. Output 0407 remains OFF.

The processor reads an ON status for input 10 and places a 1 in bit 10 of word 00. Bit 07 of word 04 remains 0.

During evaluation of the user memory (program scan), the executive evaluates and solves the control program and places a 1 in bit 07 of word 04.

After the complete program has been evaluated, the executive instructs the processor to turn ON output 20, which maps output word 04 bit 07. Thus, the pilot light turns ON.

SCAN

Input Read Scan

Input Read Scan

Program Scan

Output Write Scan

CIRCUIT

LS is open

LS is closed

LS is closed · Program is evaluated

LS is closed · Program is evaluated · PL is turned ON

means that the user program area will be fixed. In larger controllers, however, the data table size is usually selectable, according to the data storage requirements of the application. This flexibility allows the program area to be adjusted to meet the application's requirements.

When selecting a controller, the user should consider any limitations that may be placed on the use of the available application memory. One controller, for example, may have a maximum of 256 internal outputs with no restrictions on the number of timers, counters, and various types of internal outputs used. Another controller, however, may have 256 available internal outputs that are restricted to 50 timers, 50 counters, and 156 of any combination of various types of internal outputs. A similar type of restriction may also be placed on data storage registers.

One way to ensure that memory requirements are satisfied is to first understand the application requirements for programming and data storage, as well as the flexibility required for program changes and on-line data entry. Creating the program on paper first will help when evaluating these capacity requirements. With the use of a memory map, users can learn how much memory is available for the application and, then, how the application memory should be configured for their use. It is also good to know ahead of time if the application memory is expandable. This knowledge will allow the user to make sound decisions about memory type and requirements.

KEY TERMS

application memory
data table
electrically alterable read-only memory (EAROM)
electrically erasable programmable read-only memory (EEPROM)
erasable programmable read-only memory (EPROM)
executive memory
input table
memory
memory map
nonvolatile memory
output table
programmable read-only memory (PROM)
random-access memory (RAM)
read-only memory (ROM)
scratch pad memory
storage area
user program memory
volatile memory

This page intentionally left blank.

THE DISCRETE
INPUT/OUTPUT SYSTEM

*All science is concerned with the relationship
of cause and effect.*

—Laurence J. Peter

CHAPTER HIGHLIGHTS Input/output (I/O) systems put the "control" in programmable controllers. These systems allow PLCs to work with field devices to perform programmed applications. This chapter introduces the most common type of I/O system—the discrete interface—and explains its physical, electrical, and functional characteristics. You will learn how discrete I/O systems provide the connection between PLCs and the outside world. In the following two chapters, you will further explore the operation and installation of input/output systems, learning about analog and special function I/O interfaces.

6-1 INTRODUCTION TO DISCRETE I/O SYSTEMS

The discrete input/output (I/O) system provides the physical connection between the central processing unit and field devices that transmit and accept digital signals (see Figure 6-1). **Digital signals** are noncontinuous signals that have only two states—ON and OFF. Through various interface circuits and field devices (limit switches, transducers, etc.), the controller senses and measures physical quantities (e.g., proximity, position, motion, level, temperature, pressure, current, and voltage) associated with a machine or process. Based on the status of the devices sensed or the process values measured, the CPU issues commands that control the field devices. In short, input/output interfaces are the sensory and motor skills that exercise control over a machine or process.
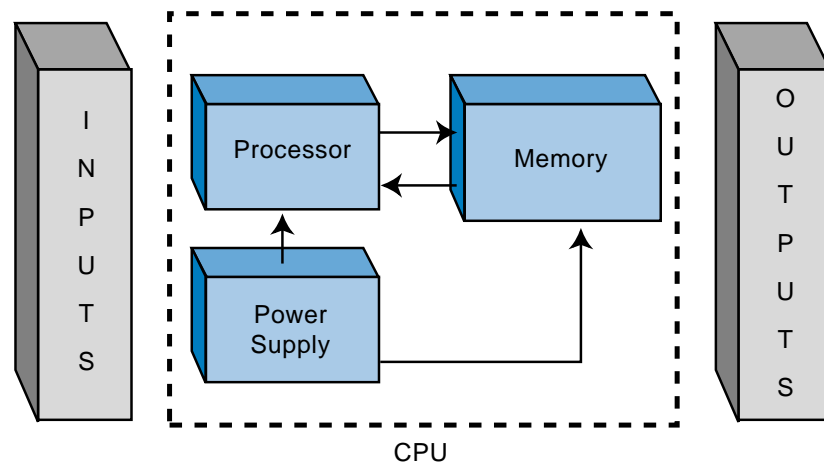

Figure 6-1. Block diagram of a PLC's CPU and I/O system.

The predecessors of today's PLCs were limited to just discrete input/output interfaces, which allowed interfacing with only ON/OFF-type devices. This limitation gave the PLC only partial control over many processes, because many process applications required analog measurements and manipulation of numerical values to control analog and instrumentation devices. Today's controllers, however, have a complete range of discrete and analog interfaces, which allow PLCs to be applied to almost any type of control. Figure 6-2 shows a typical discrete I/O system.

SECTION
2

Components
and Systems

The Discrete
Input/Output System

CHAPTER
6
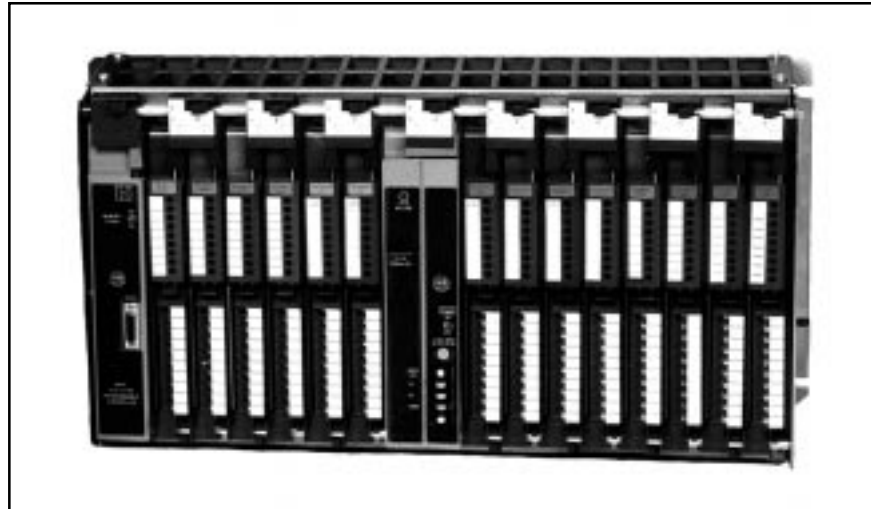
Figure 6-2. Typical discrete input/output system.

6-2 I/O RACK ENCLOSURES AND TABLE MAPPING

An **I/O module** is a plug-in–type assembly containing circuitry that communicates between a PLC and field devices. All I/O modules must be placed or inserted into a **rack enclosure**, usually referred to as a *rack*, within the PLC (see Figure 6-3). The rack holds and organizes the programmable controller's I/O modules, with a module's rack location defining the **I/O address** of its connected device. The I/O address is a unique number that identifies the input/output device during control program setup and execution. Several PLC manufacturers allow the user to select or set the addresses (to be mapped to the I/O table) for each module by setting internal switches (see Figure 6-4).
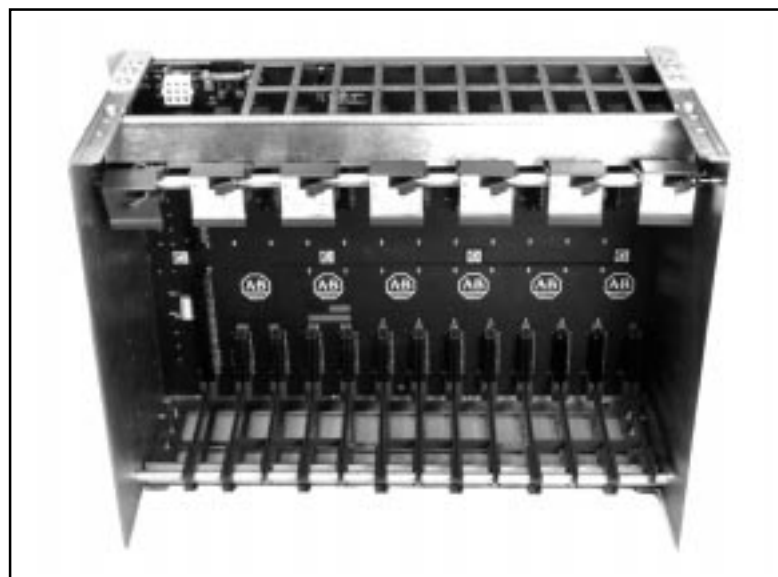
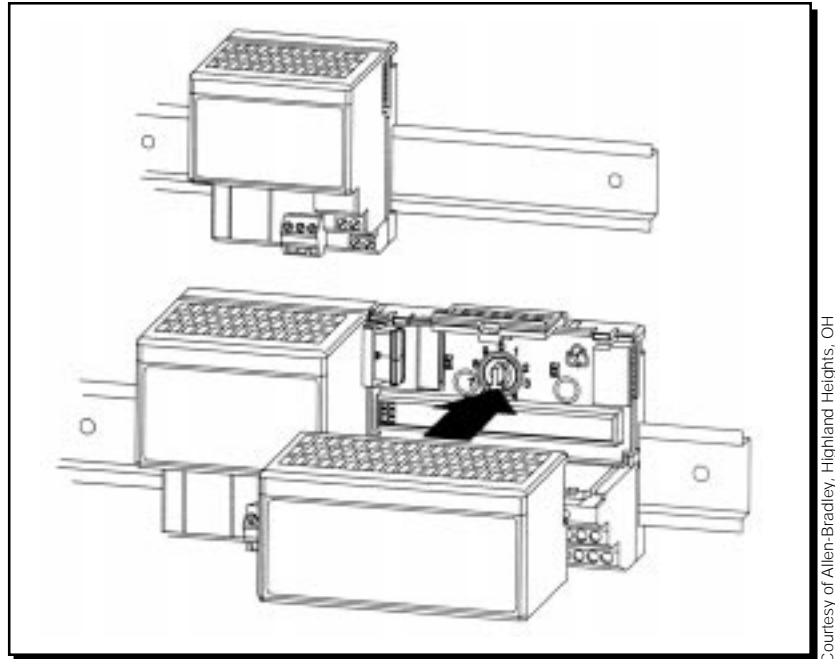**Figure 6-3.** Example of an I/O rack enclosure.

Figure 6-4. Internal switches used to set I/O addresses.

A rack, in general, recognizes the type of module connected to it (input or output) and the class of interface (discrete, analog, numerical, etc.). This module recognition is decoded on the back plane (i.e., the printed circuit board containing the data bus, power bus, and mating connectors) of the rack.

The controller's rack configuration is an important detail to keep in mind throughout system configuration. Remember that each of the connected I/O devices is referenced in the control program; therefore, a misunderstanding of the I/O location or addresses will create confusion during and after the programming stages.

Generally speaking, there are three categories of rack enclosures:

- master racks

- local racks

- remote racks

The term **master rack** (see Figure 6-5) refers to the rack enclosure containing the CPU or processor module. This rack may or may not have slots available for the insertion of I/O modules. The larger the programmable controller system, in terms of I/O, the less likely the master rack will have I/O housing capability.

A **local rack** (see Figure 6-6) is an enclosure, which is placed in the same area as the master rack, that contains I/O modules. If a master rack contains I/O modules, the master rack can also be considered a local rack. In general, a local rack (if not a master) contains a local I/O processor that sends data to and
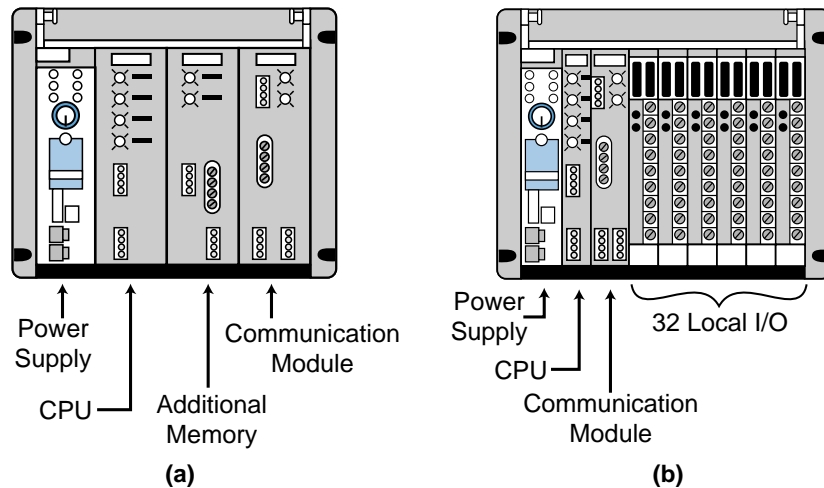
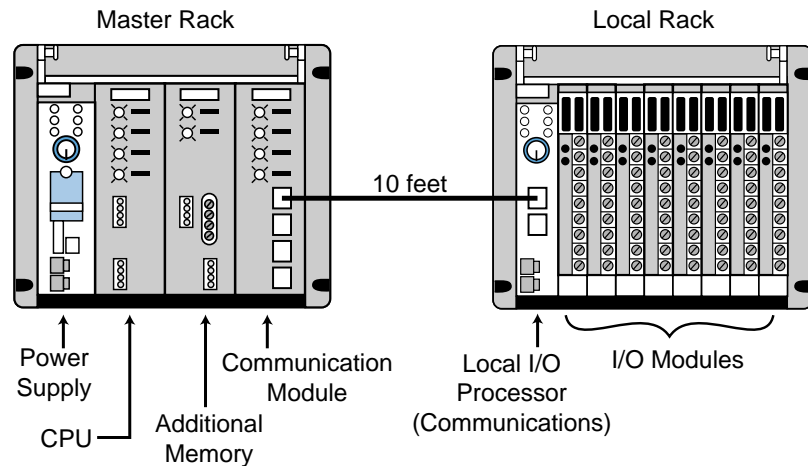**Figure 6-5.** Master racks **(a)** without I/O modules and **(b)** with I/O modules.

Figure 6-6. Local rack configuration.

from the CPU. This bidirectional information consists of diagnostic data, communication error checks, input status, and output updates. The I/O image table maps the local rack's I/O addresses.

As the name implies, **remote racks** (see Figure 6-7) are enclosures, containing I/O modules, located far away from the CPU. Remote racks contain an I/O processor (referred to as a remote I/O processor) that communicates input and output information and diagnostic status just like a local rack. The I/O addresses in this rack are also mapped to the I/O table.

The rack concept emphasizes the physical location of the enclosure and the type of processor (local, remote, or main CPU) that will be used in each particular rack. Every one of the I/O modules in a rack, whether discrete, analog, or special, has an address by which it is referenced. Therefore, each terminal point connected to a module has a particular address. This connec-
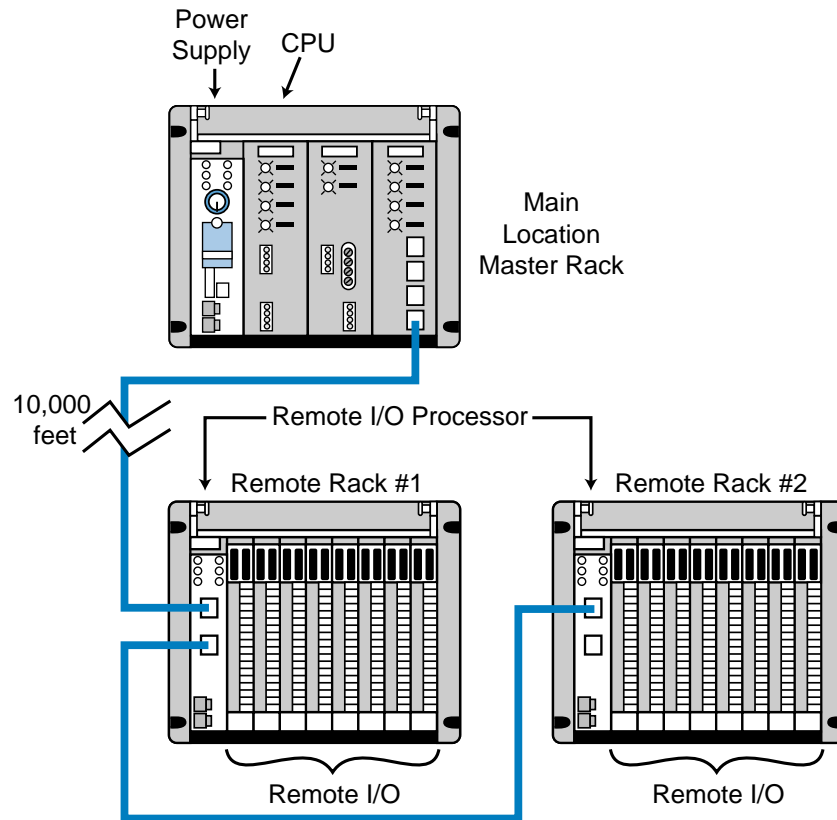
Figure 6-7. Remote rack configuration.

tion point, which ties the real field devices to their I/O modules, identifies each I/O device by the module's address and the terminal point where it is connected. This is the address that identifies the programmed input or output device in the control program.

I/O RACK AND TABLE MAPPING EXAMPLE

PLC manufacturers set specifications for placing I/O modules in rack enclosures. For example, some modules accommodate 2 to 16 field connections, while other modules require the user to follow certain I/O addressing regulations. It is not our intention in this section to review all of the different manufacturers' rules, but rather to explain how the I/O typically maps each rack and to illustrate some possible restrictions through a generic example.

As our example, let's use the PLC I/O placement specifications shown in Table 6-1. As Figure 6-8 illustrates, several factors determine the address location of each module. The type of module, input or output, determines the first address location from left to right (0 for outputs, 1 for inputs). The rack number and slot location of the module determine the next two address numbers. The terminal connected to the I/O module (0 through 7) represents the last address digit.
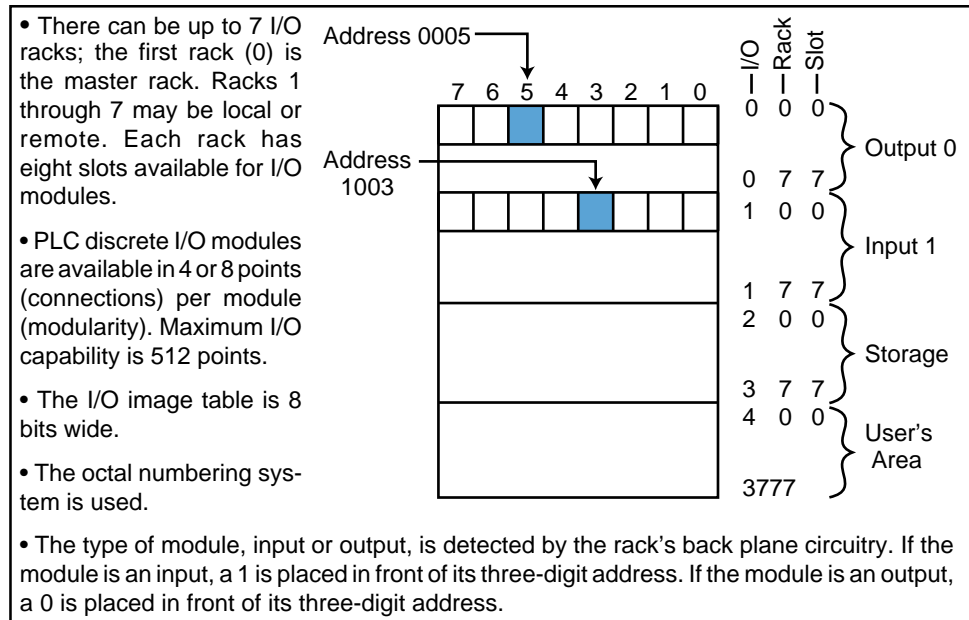
• There can be up to 7 I/O racks; the first rack (0) is the master rack. Racks 1 through 7 may be local or remote. Each rack has eight slots available for I/O modules.

• PLC discrete I/O modules are available in 4 or 8 points (connections) per module (modularity). Maximum I/O capability is 512 points.

• The I/O image table is 8 bits wide.

• The octal numbering system is used.

• The type of module, input or output, is detected by the rack's back plane circuitry. If the module is an input, a 1 is placed in front of its three-digit address. If the module is an output, a 0 is placed in front of its three-digit address.

Table 6-1. Specifications for the I/O rack enclosure example.
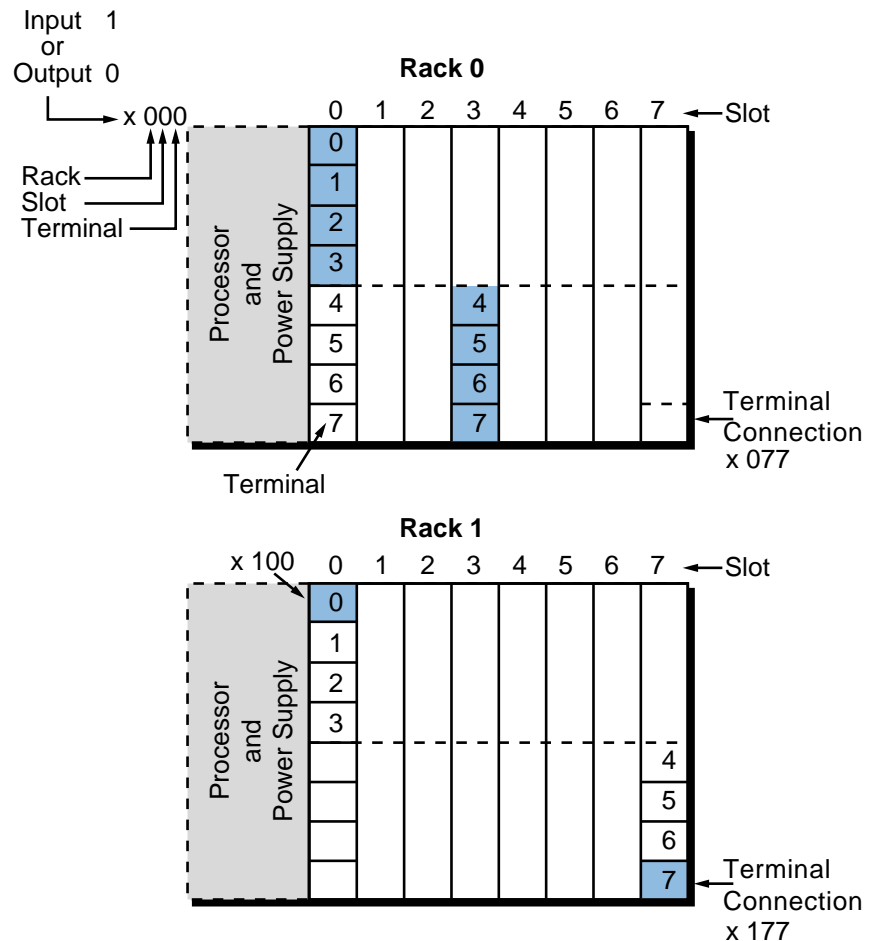
Figure 6-8. Illustration of the example I/O rack enclosure (x = 1 for inputs, 0 for outputs).

The maximum capacity of this system is 512 inputs or 512 outputs, or a total combination of 512 inputs and outputs that do not overlap addresses. The 512 possible inputs come from the following word addresses:

1000_8 (word 100, bit 0)
•
512 input addresses   •
(64 words × 8 bits/word)   to
•
•
1777_8 (word 177, bit 7)

While the 512 possible outputs come from word addresses:

0000_8 (word 000, bit 0)
•
512 output addresses   •
(64 words × 8 bits/word)   to
•
•
0777_8 (word 077, bit 7)

Again, note that the capacity is a total of 512 inputs and outputs together, not 512 each. If one input module takes a slot in the input table, the mirror image slot in the output table is taken by those inputs. The same applies for output modules.

For instance (see Figure 6-9), if a 4-point output module (see Figure 6-9b) is placed in rack 0, slot 0 (terminal addresses 0–3), the output table word 000_8, bits 0–3, represented by the shaded area in Figure 6-9c, will be mapped for outputs. Consequently, the input table image corresponding to the slot location 100_8, bits 0–3 (represented by the word *taken*) will not have a mapped reference input, since it has already been taken by outputs. If an 8-point input module is used in rack 0, slot 2 (see Figure 6-9a), indicating word location 102_8 (input = 1), the whole eight bits of that location in the input table (location 102_8 bits 0–7) would be taken by the mapping; the corresponding address in the output table (word location 002_8, bits 0–7 in Figure 6-9c) would not be able to be mapped. The bits from the output table that do not have a mapping due to the use of input modules could be used as internal outputs, since they cannot be physically connected output field devices (e.g., bits 4–7 of word 000).

For example, in Figure 6-9c, output addresses 0004 through 0007 (corresponding to word 000, bits 4–7 in the I/O table) cannot be physically connected to an output module because their map locations are taken by an input module (at word 100, bits 4–7). Therefore, these reference addresses can only be used as internal coil outputs. The use of these output bits as internal outputs is shown in Figure 6-10, where output 0004 (now used as an internal coil) will be turned ON if its logic is TRUE and contacts from this output can be used in other output rungs.
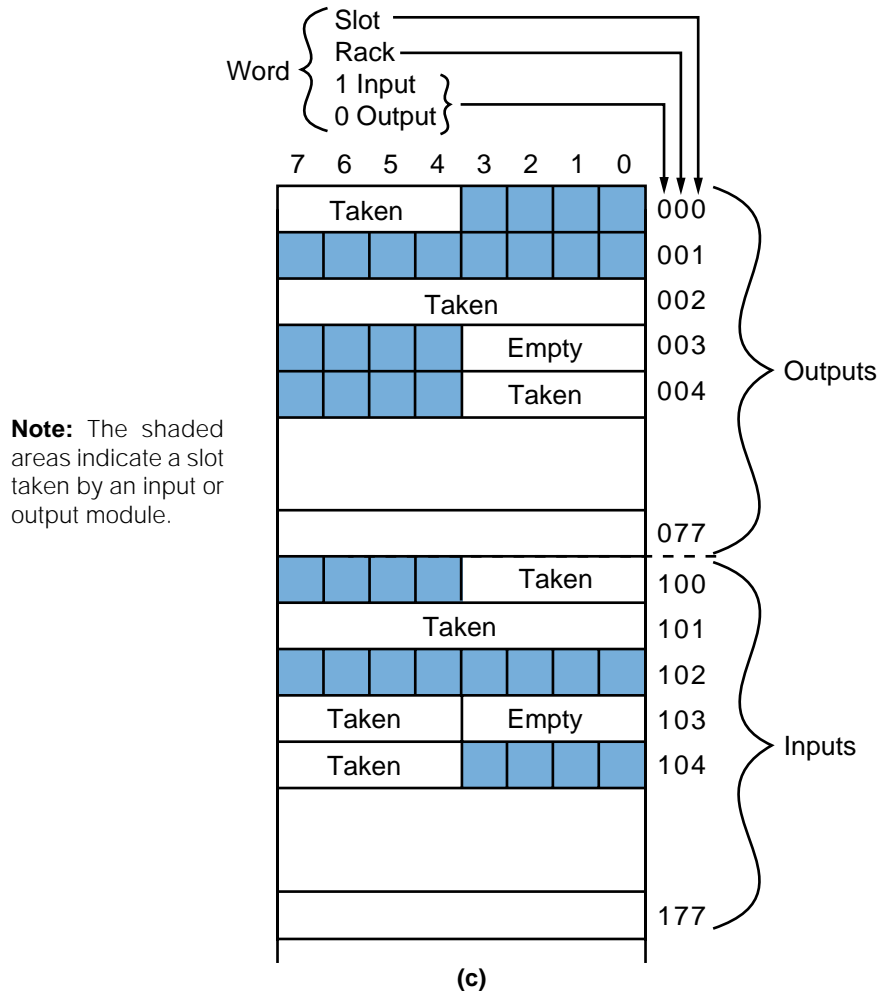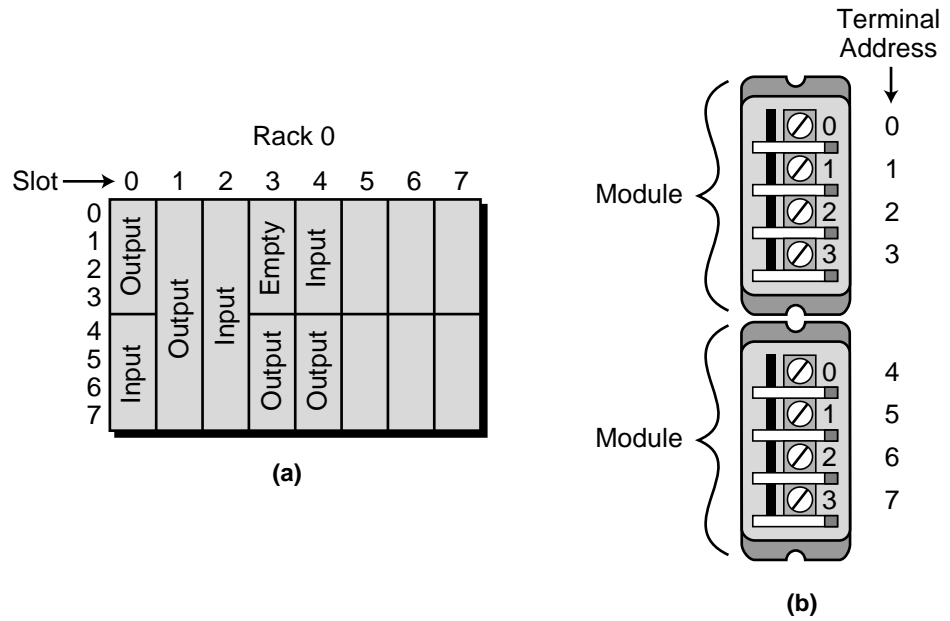
Figure 6-9. Diagrams of **(a)** an I/O table, **(b)** two 4-point I/O modules in one slot, and **(c)** an I/O table mapping.
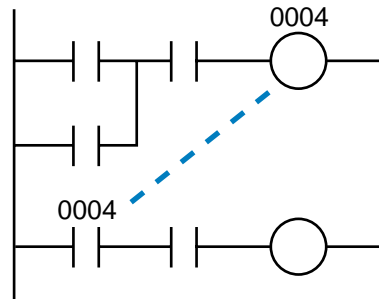
Figure 6-10. Output 0004 used as an internal coil.

6-3 REMOTE I/O SYSTEMS

In large PLC systems (upwards of 512 I/O), input/output subsystems can be located away from the central processing unit. A **remote I/O subsystem** is a rack-type enclosure, separate from the CPU, where I/O modules can be installed. A remote rack includes a power supply that drives the logic circuitry of the interfaces and a remote I/O adapter or processor module that allows communication with the main processor (CPU). The communication between I/O adapter modules and the CPU occurs in serial binary form at speeds of up to several megabaud (millions of bits transmitted per second). This serial information packet contains 1s and 0s, representing both the status of the I/O and diagnostic information about the remote rack.

The capacity of a single subsystem (rack) is normally 32, 64, 128, or 256 I/O points. A large system with a maximum capacity of 1024 I/O points may have subsystem sizes of either 64 or 128 points—eight racks with 128 I/O, sixteen racks with 64 I/O, or some combination of both sizes equal to 1024 I/O. In the past, only discrete interface modules could be placed in the racks of most remote subsystems. Today, however, remote I/O subsystems also accommodate analog and special function interfaces.

Individual remote subsystems are normally connected to the CPU via one or two twisted-pair conductors or a single coaxial cable, using either a *daisy chain, star*, or *multidrop* configuration (see Figure 6-11). The distance a remote rack can be placed away from the CPU varies among products, but it can be as far as two miles. Another approach for connecting remote racks to the CPU is a fiber-optic data link, which allows greater distances and has higher noise resistance.

Remote I/O offers tremendous materials and labor cost savings on large systems where the field devices are clustered at various, distant locations. With the CPU in a main control room or some other central area, only the communication link must be wired between the remote rack and the processor, replacing hundreds of field wires. Another advantage of remote I/O is that

subsystems may be installed and started up independently, allowing maintenance of individual subsystems while others continue to operate. Also, troubleshooting and connection checks become much easier, since hundreds of wires do not need to be checked all the way back to the master rack.
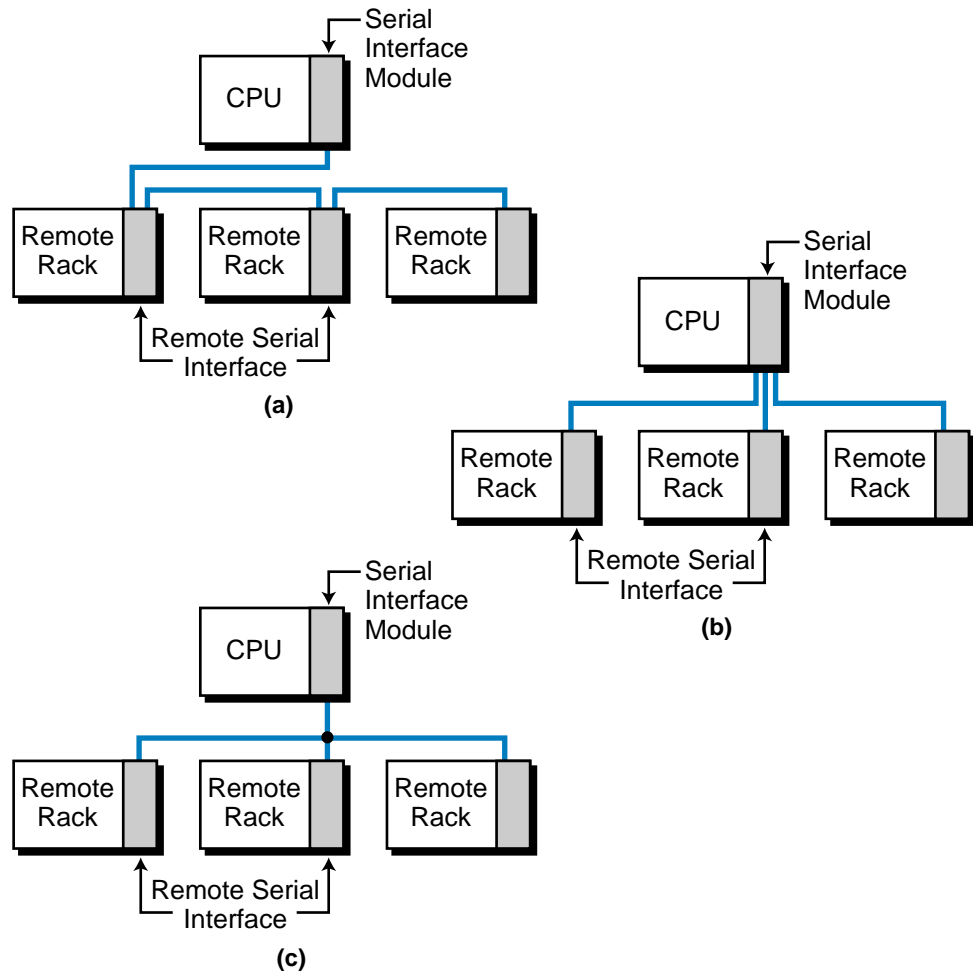
Figure 6-11. Remote I/O configurations: **(a)** daisy chain, **(b)** star, and **(c)** multidrop.

6-4 PLC INSTRUCTIONS FOR DISCRETE INPUTS

The most common class of input interfaces is digital (or discrete). **Discrete input interfaces** connect digital field input devices (those that send noncontinuous, fixed-variable signals) to input modules and, consequently, to the programmable controller. The discrete, noncontinuous characteristic of digital input interfaces limits them to sensing signals that have only two states (i.e., ON/OFF, OPEN/CLOSED, TRUE/FALSE, etc.). To an input interface circuit, discrete input devices are essentially switches that are either open or closed, signifying either 1 (ON) or 0 (OFF). Table 6-2 shows several examples of discrete input field devices.

Field Input Devices
Circuit breakers
Level switches
Limit switches
Motor starter contacts
Photoelectric eyes
Proximity switches
Push buttons
Relay contacts
Selector switches
Thumbwheel switches (TWS)

Table 6-2. Discrete input devices.

Many instructions are designed to manipulate discrete inputs. These instructions handle either *single bits*, which control one field input connection, or *multibits,* which control many input connections. Regardless of whether the instruction controls one discrete input or multiple inputs, the information provided by the field device is the same—either ON or OFF.

During our discussion of input modules, keep in mind the relationship between interface signals (ON/OFF), rack and module locations (where the input device is inserted), and I/O table mapping and addressing (used in the control program). Remember that each PLC manufacturer determines the addressing and mapping scheme used with its systems. Manufacturers may use a 1 for an input and a 0 for an output, or they may simply assign an I/O address for the input or output module inserted in a particular slot of a rack. Figure 6-12 illustrates a simplified 8-bit image table where limit switch LS1 is connected to a discrete input module in rack 0, which can connect 8 field inputs (0–7). Note that LS1 is known as input 014, which stands for rack 0, slot 1, connection 4.

When an input signal is energized (ON), the input interface senses the field device's supplied voltage and converts it to a logic-level signal (either 1 or 0), which indicates the status of that device. A logic 1 in the input table indicates an ON or CLOSED condition, and a logic 0 indicates an OFF or OPEN condition. PLC symbolic instructions, which include the normally open (⊣⊢) and normally closed (⊣/⊢) instructions, transfer this field status information into the input table.

For multibit modules that receive multiple inputs, such as thumbwheel switches used in register (BCD) interfaces, block transfer or get data instructions place input values into the data table (see Figure 6-13). Chapter 9 explains single-bit and multibit instructions in more detail.
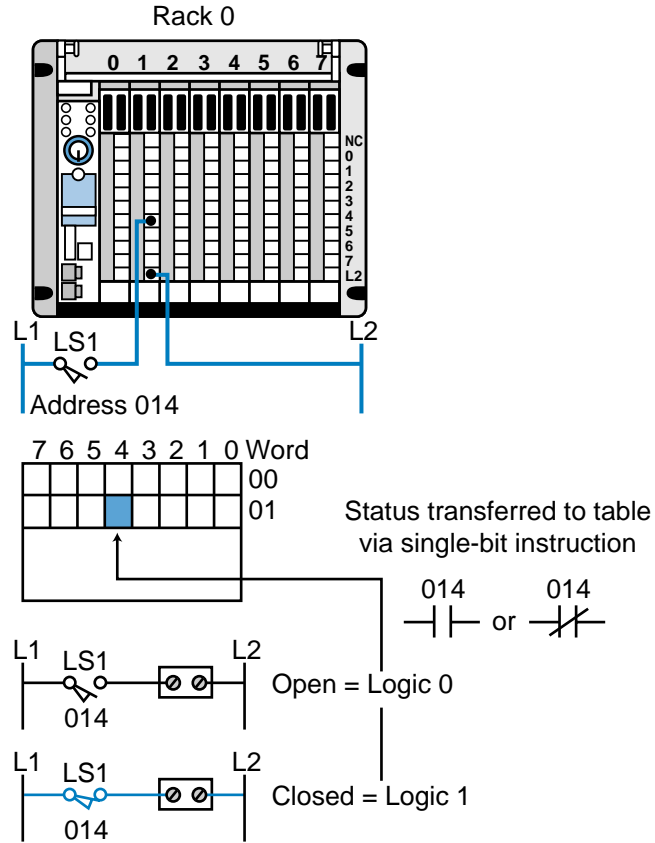
Rack 0
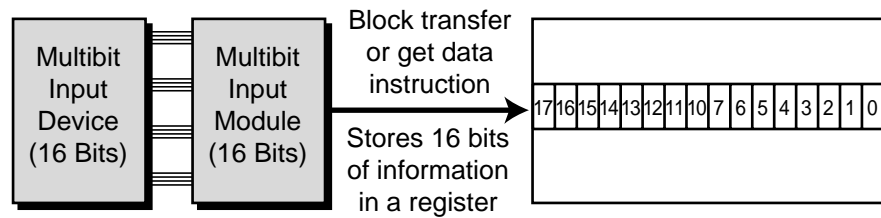

Figure 6-12. An 8-bit input image table.

Figure 6-13. Block transfer and get data instructions transferring multibit input values into the data table.

EXAMPLE 6-1

For the rack configuration shown in Figure 6-14, determine the address for each field device wired to each input connection in the 8-bit discrete input module. Assume that the first four slots of this 64 I/O micro-PLC are filled with outputs and that the second four slots are filled with inputs. Also, assume that the addresses follow a rack-slot-connection scheme and start at I/O address 000. Note that the number system is octal.

SECTION
2

*Components
and Systems*

*The Discrete
Input/Output System*

CHAPTER
6
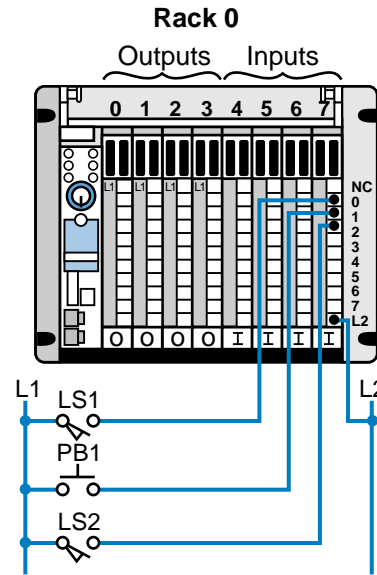
Rack 0

Figure 6-14. Rack configuration for Example 6-1.

SOLUTION

The discrete input module (where the input devices are connected) will have addresses 070 through 077, because it is located in rack 0, slot number 7. Therefore, each of the field input devices will have addresses as shown in Figure 6-15; LS1 will be known as input 070, PB1 as input 071, and LS2 as input 072. The control program will reference the field devices by these addresses. If LS1 is rewired to another connection in another discrete input, its address reference will change. Consequently, the address must be changed in the control program because there can only be one address per discrete field input device connection.
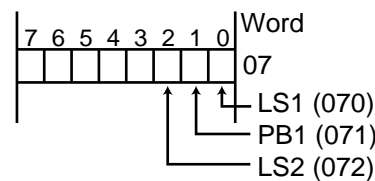
Figure 6-15. Field device addresses for the rack configuration in Example 6-1.

6-5 TYPES OF DISCRETE INPUTS

As mentioned earlier, discrete input interfaces sense noncontinuous signals from field devices—that is, signals that have only two states. Discrete input interfaces receive the voltage and current required for this operation from the back plane of the rack enclosure where they are inserted (see Chapter 4 for loading considerations). The signal that these discrete interfaces receive from

input field devices can be of different types and/or magnitudes (e.g., 120 VAC, 12 VDC). For this reason, discrete input interface circuits are available in different AC and DC voltage ratings. Table 6-3 lists the standard ratings for discrete inputs.

Input Ratings
24 volts AC/DC
48 volts AC/DC
120 volts AC/DC
230 volts AC/DC
TTL level
Nonvoltage
Isolated input
5–50 volts DC (sink/source)

Table 6-3. Standard ratings for discrete input interfaces.

To properly apply input interfaces, you should have an understanding of how they operate and an awareness of certain operating specifications. Section 6-9 discusses these specifications, while Chapter 20 describes start-up and maintenance procedures for I/O systems. Now, let's look at the different types of discrete input interfaces, along with their operation and connections.

AC/DC INPUTS

Figure 6-16 shows a block diagram of a typical **AC/DC input interface** circuit. Input circuits vary widely among PLC manufacturers, but in general, AC/DC interfaces operate similarly to the circuit in the diagram. An AC/DC input circuit has two primary parts:

- the power section
- the logic section

These sections are normally, but not always, coupled through a circuit that electrically separates them, providing isolation.
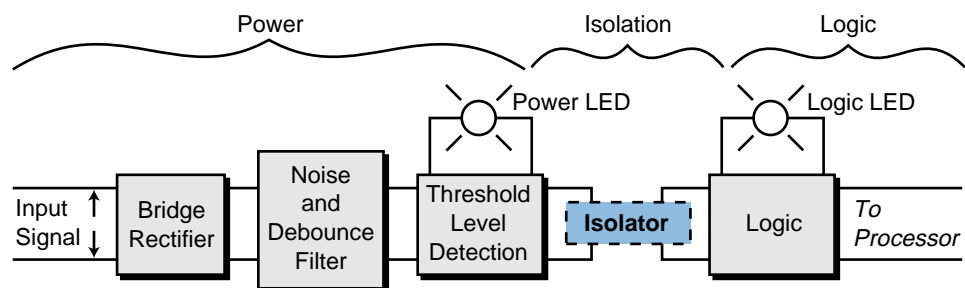

Figure 6-16. Block diagram of an AC/DC input circuit.

The power section of an AC/DC input interface converts the incoming AC voltage from an input-sensing device, such as those described in Table 6-2, to a DC, logic-level signal that the processor can use during the read input section of its scan. During this process, the bridge rectifier circuit of the interface's power section converts the incoming AC signal to a DC-level signal. It then passes the signal through a filter circuit, which protects the signal against bouncing and electrical noise on the input power line. This filter causes a signal delay of typically 9–25 msec. The power section's threshold circuit detects whether the signal has reached the proper voltage level for the specified input rating. If the input signal exceeds and remains above the threshold voltage for a duration equal to the filter delay, the signal is recognized as a valid input.

Figure 6-17 shows a typical AC/DC input circuit. After the interface detects a valid signal, it passes the signal through an isolation circuit, which completes the electrically isolated transition from an AC signal to a DC, logic-level signal. The logic circuit then makes the DC signal available to the processor through the rack's back plane data bus, a pathway along which data moves. The signal is electrically isolated so that there is no electrical connection between the field device (power) and the controller (logic). This electrical separation helps prevent large voltage spikes from damaging either the logic side of the interface or the PLC. An optical coupler or a pulse transformer provides the coupling between the power and logic sections.
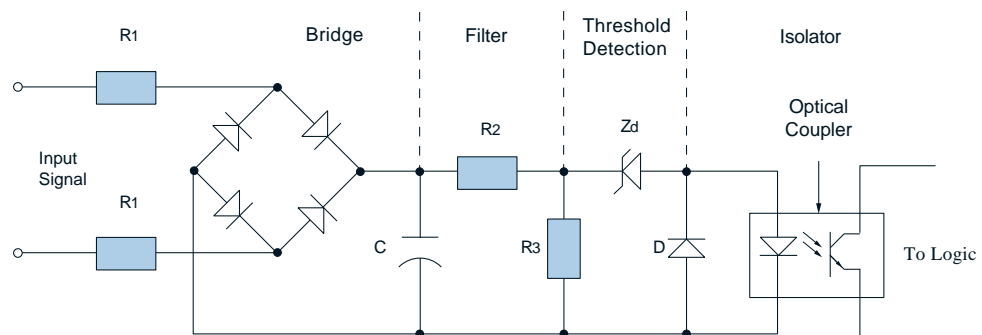

Figure 6-17. Typical AC/DC input circuit.

Most AC/DC input circuits have an LED (power) indicator to signal that the proper input voltage level is present (refer to Figure 6-16). In addition to the power indicator, the circuit may also have an LED to indicate the presence of a logic 1 signal in the logic section. If an input voltage is present and the logic circuit is functioning properly, the logic LED will be lit. When the circuit has both voltage and logic indicators and the input signal is ON, both LEDs must be lit to indicate that the power and logic sections of the module are operating correctly. Figure 6-18 shows AC/DC device connection diagrams.
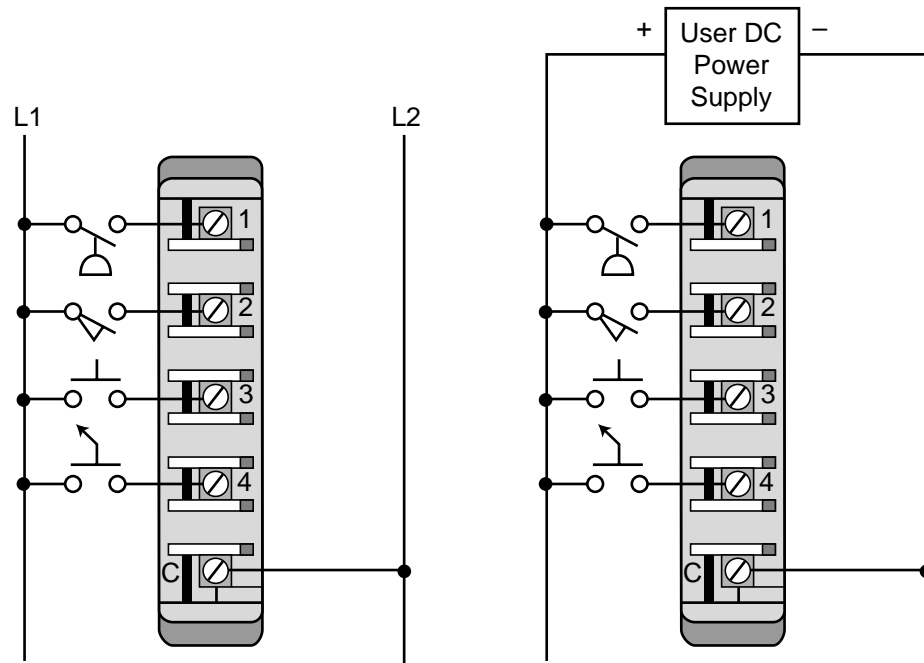
Figure 6-18. Device connections for **(a)** an AC input module and **(b)** a DC input module with common wire connection "C" used to complete the path from hot.

DC INPUTS (SINK/SOURCE)

A **DC input module** interfaces with field input devices that provide a DC output voltage. The difference between a DC input interface and an AC/DC input interface is that the DC input does not contain a bridge circuit, since it does not convert an AC signal to a DC signal. The input voltage range of a DC input module varies between 5 and 30 VDC. The module recognizes an input signal as being ON if the input voltage level is at 40% (or another manufacturer-specified percentage) of the supplied reference voltage. The module detects an OFF condition when the input voltage falls under 20% (or another manufacturer-specified percentage) of the reference DC voltage.

A DC input module can interface with field devices in both **sinking** and **sourcing** operations, a capability that AC/DC input modules do not have. Sinking and sourcing operations refer to the electrical configuration of the circuits in the module and field input devices. If a device *provides* current when it is ON, it is said to be sourcing current. Conversely, if a device *receives* current when it is ON, it is said to be sinking current. There are both sinking and sourcing field devices, as well as sinking and sourcing input modules. The most common, however, are sourcing field input devices and sinking input modules. Rocker switches inside a DC input module may be used to select sink or source capability. Figure 6-19 depicts sinking and sourcing operations and current direction.
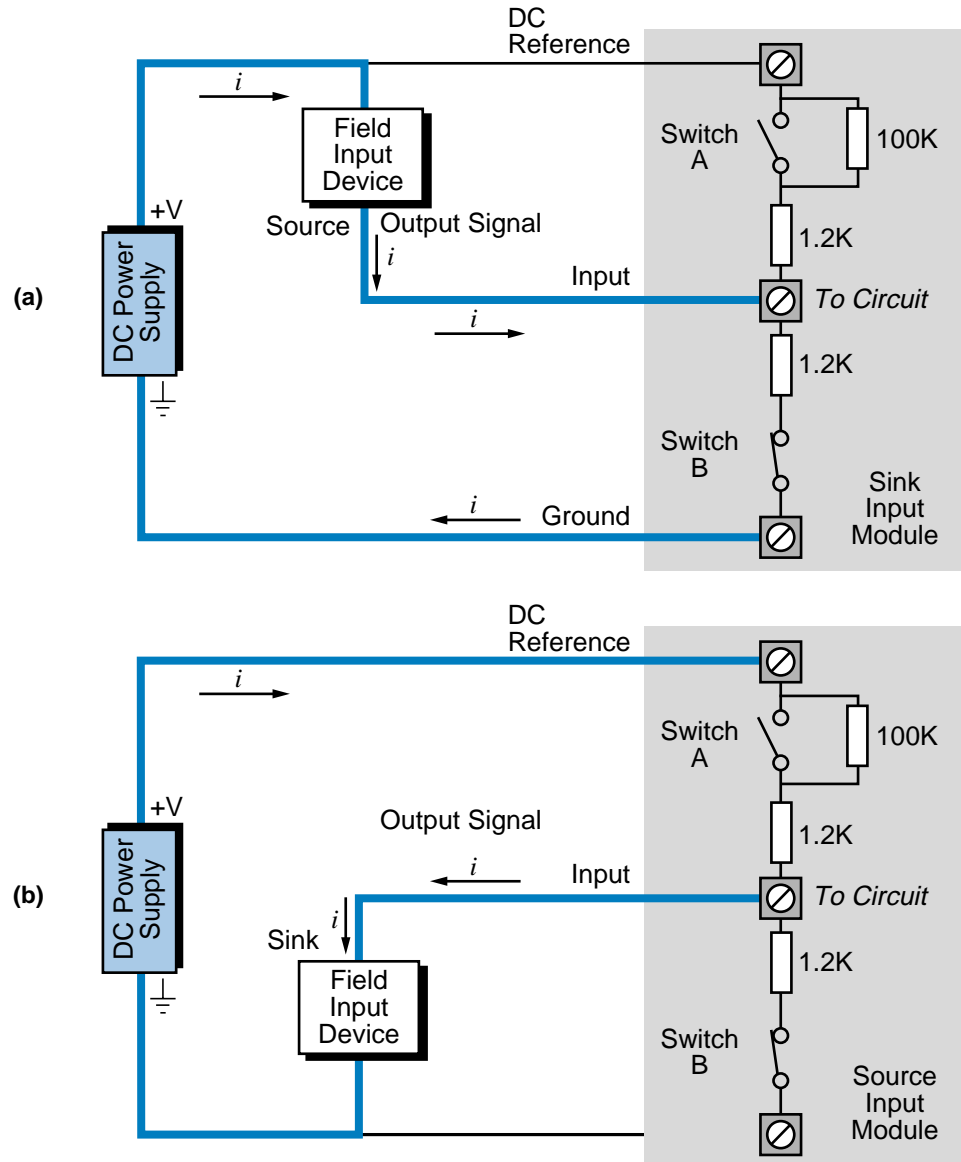
Figure 6-19. Current for **(a)** a sinking input module/sourcing input device and **(b)** a sourcing input module/sinking input device.

During interfacing, the user must keep in mind the minimum and maximum specified currents that the input devices and module are capable of sinking or sourcing. Also, if the module allows selection of a sink or source operation via selector switches, the user must assign them properly. A potential interface problem could arise, for instance, if an 8-input module was set for a sink operation and all input devices except one were operating in a source configuration. The source input devices would be ON, but the module would not properly detect the ON signal, even though a voltmeter would detect a voltage across the module's terminals. Figure 6-20 illustrates three field device connections to a DC input module with both sinking and sourcing input device capabilities.
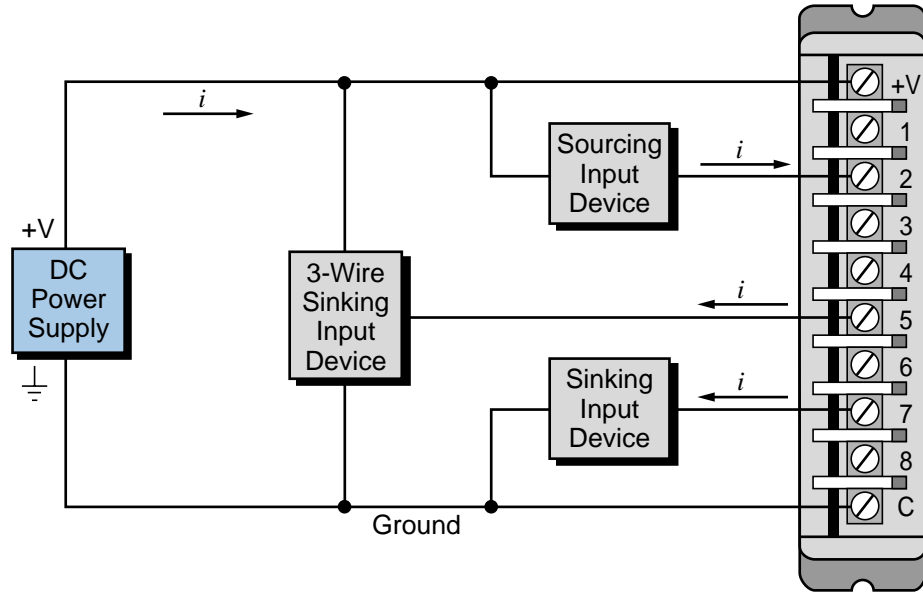
Figure 6-20. Field device connections for a sink/source DC input module.

The majority of DC proximity sensors used as PLC inputs provide a sinking sensor output, thereby requiring a sinking input module. However, if an application requires only one sinking output and the controller already has several sourcing inputs connected to a sourcing input module, the user may use the inexpensive circuit shown in Figure 6-21 to interface the sinking output with the sourcing input module. The sourcing current provided by this input is approximately 50 mA. Note that if the supply voltage (V_S) is increased, the current I_{out} will be greater than 50 mA.
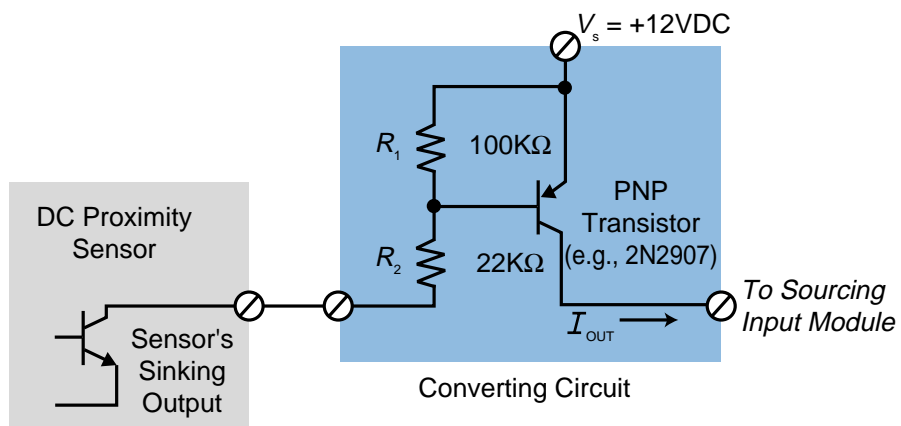
Figure 6-21. Conversion circuit interfacing a sinking output with a sourcing input module.

ISOLATED AC/DC INPUTS

Isolated input interfaces operate like standard AC/DC modules except that each input has a separate return, or *common*, line. Depending on the manufacturer, standard AC/DC input interfaces may have one return line per 4, 8, or 16 points. Although a single return line, provided in standard multipoint input modules, may be ideal for 95% of AC/DC input applications, it may not be suitable for applications requiring individual or isolated common lines. An example of this type of application is a set of input devices that are connected to different phase circuits coming from different power distribution centers. Figure 6-22 illustrates a sample device connection for an AC/DC input isolation interface capable of connecting five input devices.
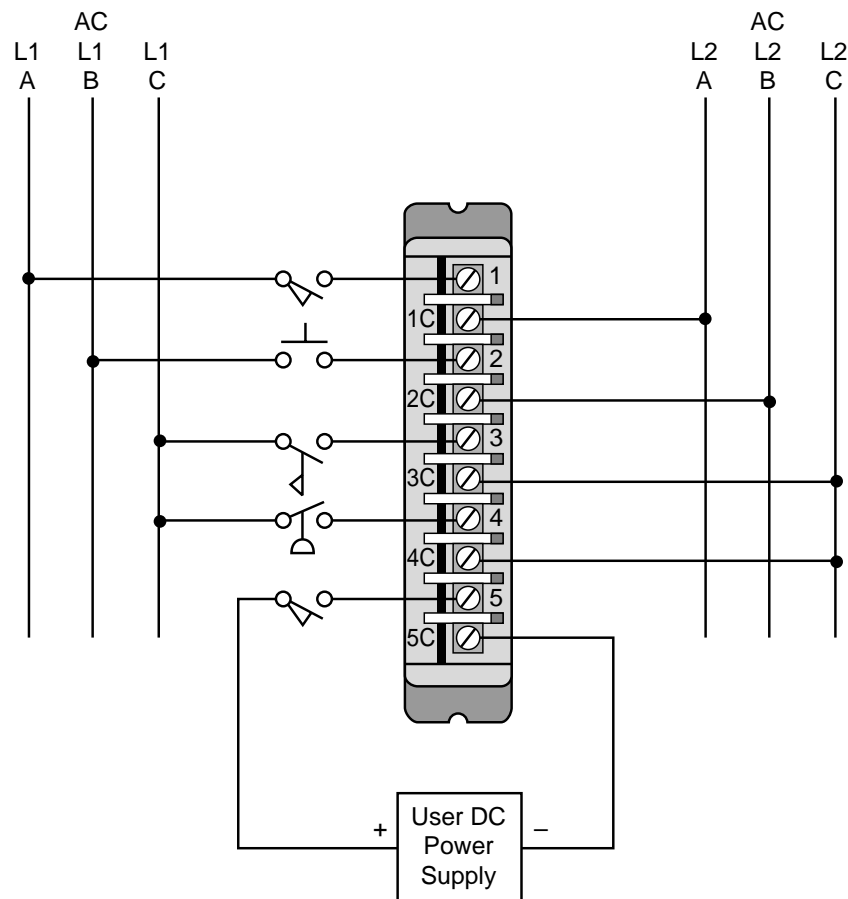
Figure 6-22. Device connection for an AC/DC isolated input interface.

Isolated input interfaces provide fewer points per module than their standard counterparts. This decreased modularity exists because isolated inputs require extra terminal connections to connect each of the return lines.

If isolation modules are not available for an application requiring singular return lines, standard interfaces may be used. However, the standard inter-

faces will lose inputs, because to keep isolation among inputs, they can have only one input line per return line. For example, a 16-point standard module with one common line per four points can accommodate four distinct isolated field input devices (each from a different source). However, as a result, it will lose 12 points. Figure 6-23 illustrates an 8-point module with different commons for every four inputs, thus allowing two possible isolated inputs.
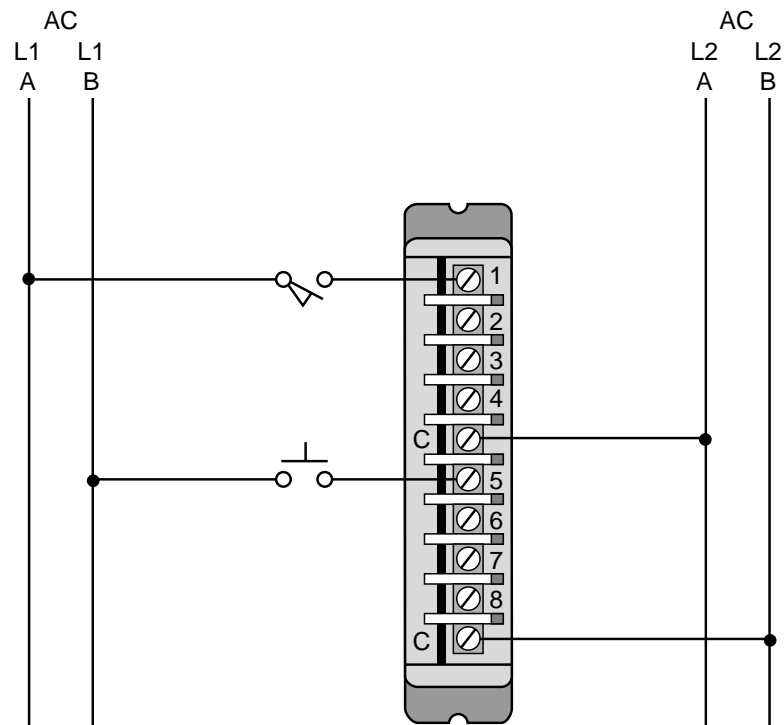

Figure 6-23. An 8-point standard input module used as an isolated module.

TTL INPUTS

Transistor-transistor logic (TTL) input interfaces allow controllers to accept signals from TTL-compatible devices, such as solid-state controls and sensing instruments. TTL inputs also interface with some 5 VDC–level control devices and several types of photoelectric sensors. The configuration of a TTL interface is similar to an AC/DC interface, but the input delay time caused by filtering is much shorter. Most TTL input modules receive their power from within the rack enclosure; however, some interfaces require an external 5-VDC power supply (rack or panel mounted).

Transistor-transistor logic modules may also be used in applications that use BCD thumbwheel switches (TWS) operating at TTL levels. These interfaces provide up to eight inputs per module and may have as many as sixteen inputs (high-density input modules). A TTL input module can also interface

with thumbwheel switches if these input devices are TTL compatible. Figure 6-24 illustrates a typical TTL input module connection diagram with an external power supply.
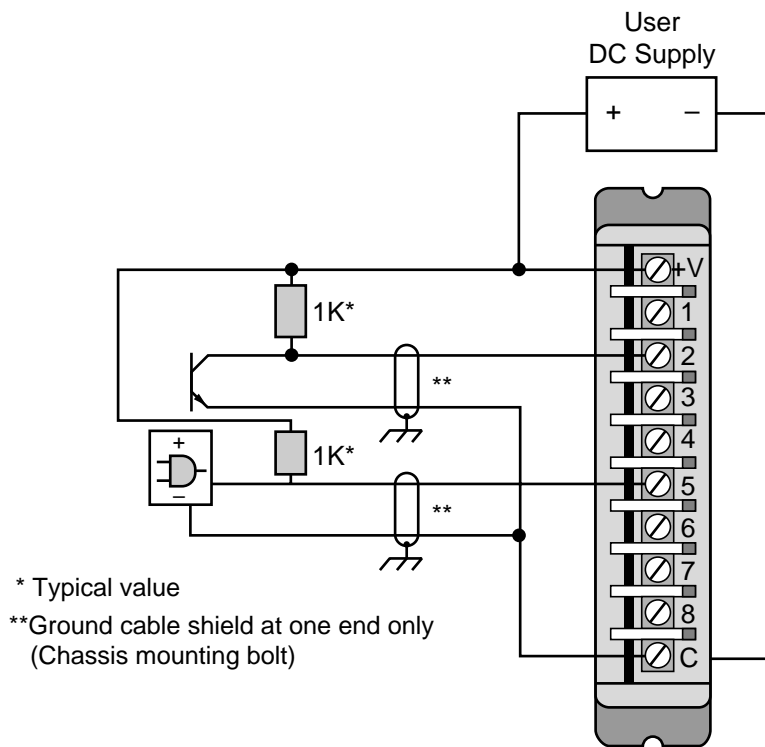

Figure 6-24. TTL input connection diagram.

REGISTER/BCD INPUTS

Multibit **register/BCD input modules** enhance input interfacing methods with the programmable controller through the use of standard thumbwheel switches. This register, or BCD, configuration allows groups of bits to be input as a unit to accommodate devices requiring that bits be in parallel form.

Register/BCD interfaces are used to input control program parameters to specific register or word locations in memory (see Figure 6-25). Typical input parameters include timer and counter presets and set-point values. The operation of register input modules is almost identical to that of TTL and DC input modules; however, unlike TTL input modules, register/BCD interfaces accept voltages ranging from 5 VDC (TTL) to 24 VDC. They are also grouped in modules containing 16 or 32 inputs, corresponding to one or two I/O registers (mapped in the I/O table), respectively. Data manipulation instructions, such as get or block transfer in, are used to access the data from the register input interface. Figure 6-26 illustrates a typical device connection for a register input.
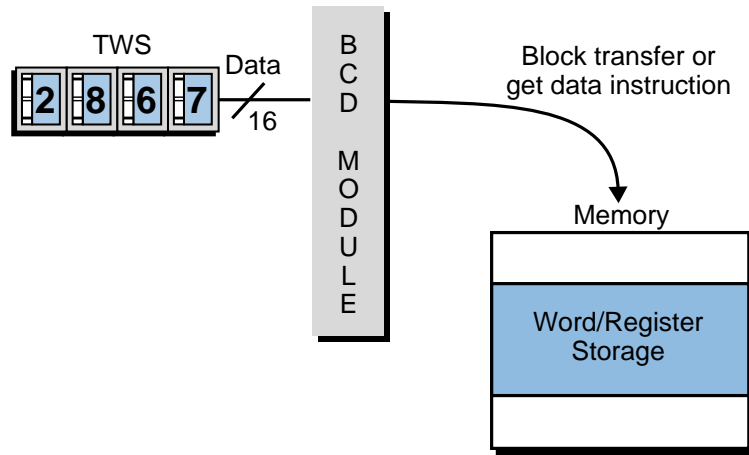
Figure 6-25. BCD interface inputting parameters into register/word locations in memory.
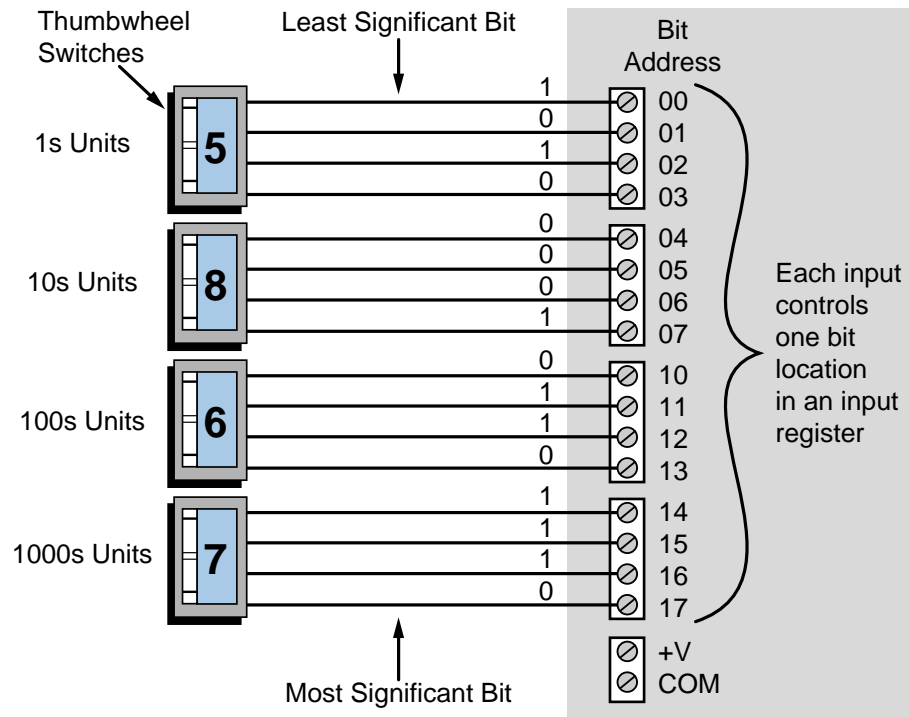

Figure 6-26. Register or BCD input module connection diagram.

Some manufacturers provide **multiplexing** capabilities that allow more than one input line to be connected to each terminal in a register module (see Figure 6-27). This kind of multiplexed register input requires thumbwheel switches that have an enable line (see Figure 6-28). When this line is selected, the TWS provides a BCD output at its terminals; when it is not selected, the TWS does not provide an output. If the TWS set provides four digits with one enable line (see Figure 6-29), then the enable line will make all of the outputs available
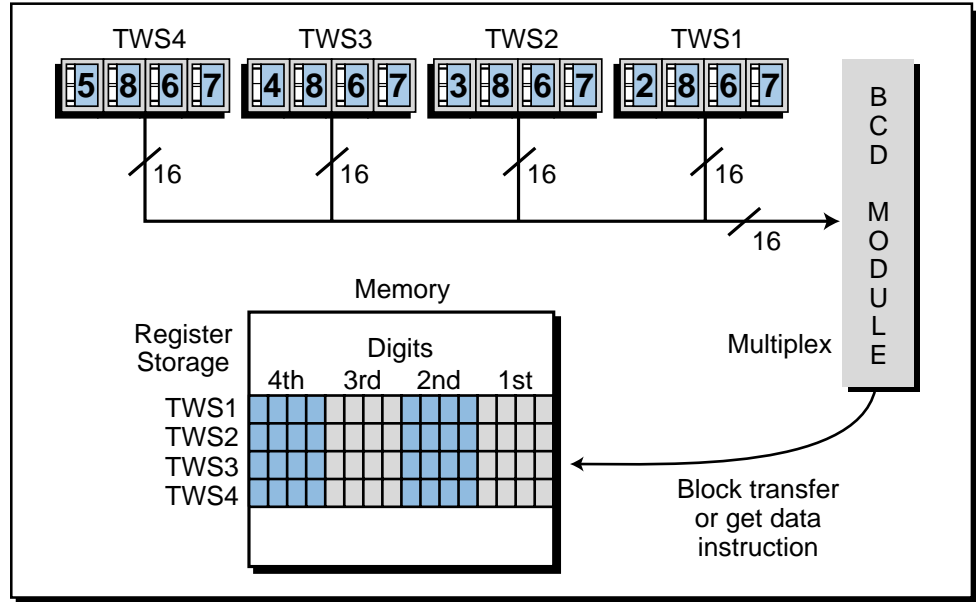
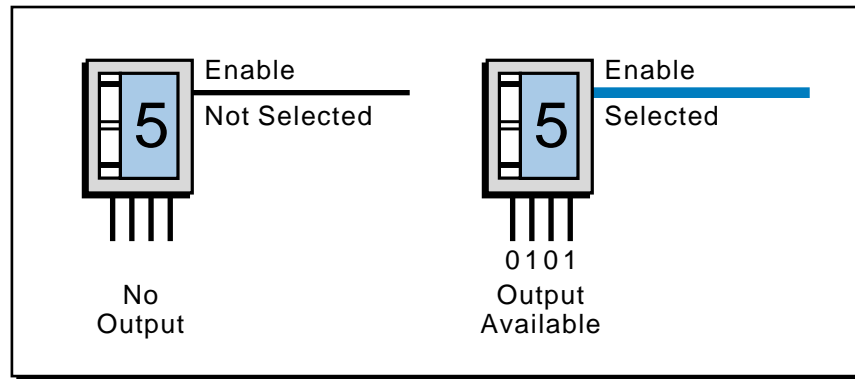**Figure 6-27.** Multiplexing input module connection diagram.

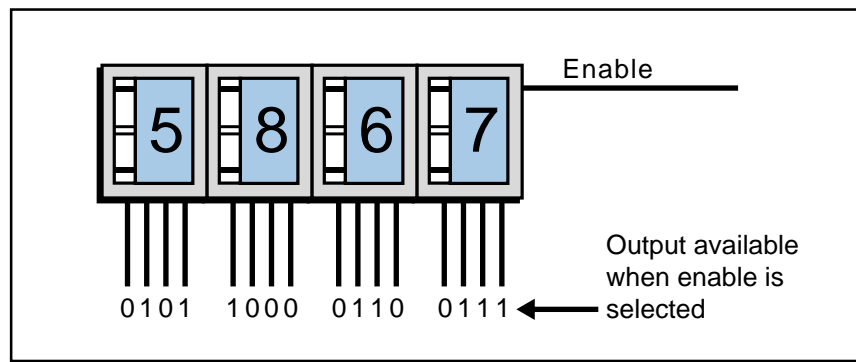**Figure 6-28.** Single-digit TWS with enable line.

Figure 6-29. A 4-digit TWS with one common enable line.

when it is selected. This multiplexing technique minimizes the number of input modules required to read several sets of four-digit TWS. For instance, a 16-bit input module capable of multiplexing 6 input devices ($6 \times 16 = 96$ total inputs) could receive information from six 4-digit thumbwheel switches. The user would not need to decode each of the six sets of 16 input groups, since the multiplexed module enables each group of 16 inputs to be read one scan at a time. However, the user may have to specify the register or word addresses where the 16-bit data will be stored through an instruction that specifies the storage location, along with the length or number of registers to be stored. Figure 6-30 illustrates a block diagram connection for a module capable of multiplexing four 4-digit TWS (four 16-bit input lines).
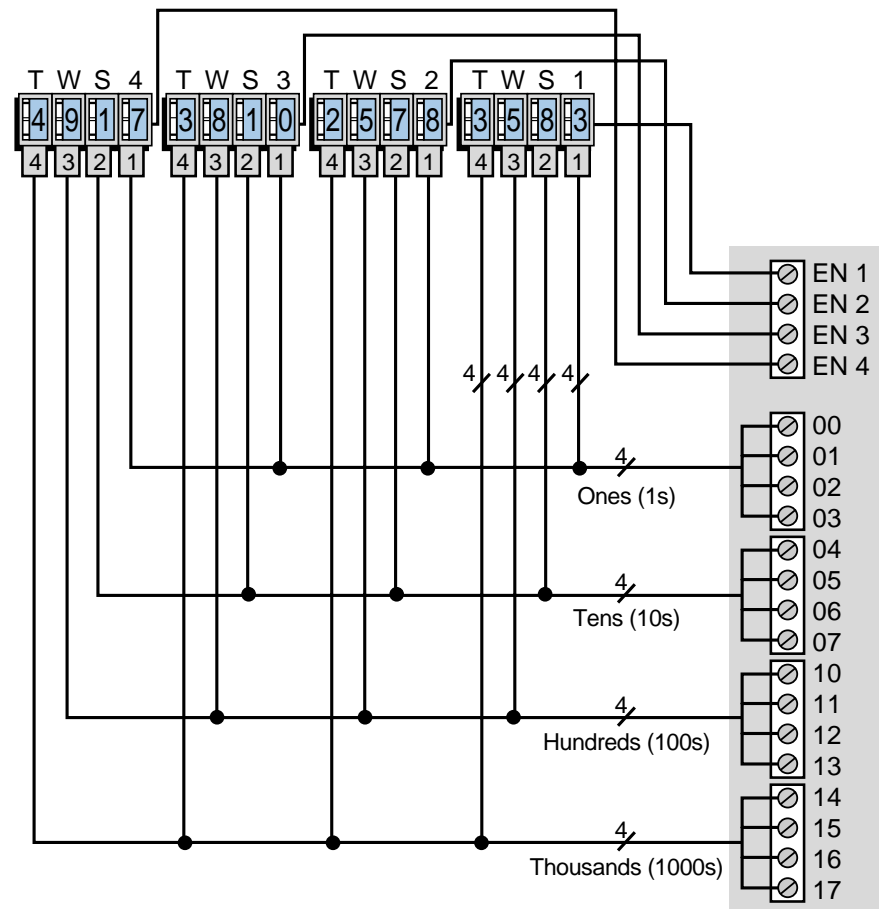

Figure 6-30. Block diagram of a multiplexed input module connected to four 4-digit TWS.

EXAMPLE 6-2

Referencing Figure 6-30, determine the values of the registers (in BCD) after an input transfer is made (in this case via a block transfer input of data). The input has a starting destination register of 4000 and a length of 4 registers (i.e., from registers 4000 to 4003). Assume that TWS set 1 is read first, TWS set 2 is read second, etc.

SOLUTION

The contents of register 4000 in BCD will be the BCD code equivalent of the first set of thumbwheel switches connected to the PLC register input module, and likewise for registers 4001, 4002, and 4003. Figure 6-31 shows the register contents. Note that the contents of each register does not represent the decimal equivalent of the binary pattern stored in that location, but rather the BCD equivalent. To change this number to decimal, you must convert the BCD pattern to its decimal equivalent using other instructions. For instance, the decimal equivalent of the binary (BCD) pattern stored in register 4000 is 13,699, not 3,583, as the TWS (BCD number) indicates.

Word or Register

Contents in BCD

Register																	BCD	
4000	0	0	1	1	0	1	0	1	1	0	0	0	0	0	1	1	3583	Value of 1st Set
4001	0	0	1	0	0	1	0	1	0	1	1	1	1	0	0	0	2578	Value of 2nd Set
4002	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	3810	Value of 3rd Set
4003	0	1	0	0	1	0	0	1	0	0	0	1	0	1	1	1	4917	Value of 4th Set

Storage Table

Figure 6-31. Register contents for Example 6-2.

6-6 PLC INSTRUCTIONS FOR DISCRETE OUTPUTS

Like discrete input interfaces, **discrete output interfaces** are the most commonly used type of PLC output modules. These outputs connect the programmable controller with discrete output field devices. Many single-bit and multibit instructions are designed to manipulate discrete outputs.

During this discussion of output modules, keep in mind the relationship between output interface signals (ON/OFF), rack and module locations (where the output modules are inserted), and I/O table maps and addresses (used in the control program). Figure 6-32 illustrates a simplified 8-bit output image table. The coil of the motor starter (M1) is connected to a discrete output module (slot 7) in rack 0, which can connect 8 field inputs (0–7). Note that the starter will be known as output 077, which stands for rack 0, slot 7, terminal connection 7.

Output interface circuitry switches the supplied voltage from the PLC ON or OFF according to the status of the corresponding bit in the output image table. This status (1 or 0) is set during the execution of the control program and is sent to the output module at the end of scan (output update). If the signal from the processor is 1, the output module will switch the supplied voltage

(e.g., 120 VAC) to the output field device, turning the output ON. If the signal received from the processor is 0, the module will deactivate the field device by switching to 0 volts, thus turning it OFF. Typically, an output coil (—○) instruction, like the one shown in Figure 6-32, activates the output interface when the reference address is logic 1 (ON).
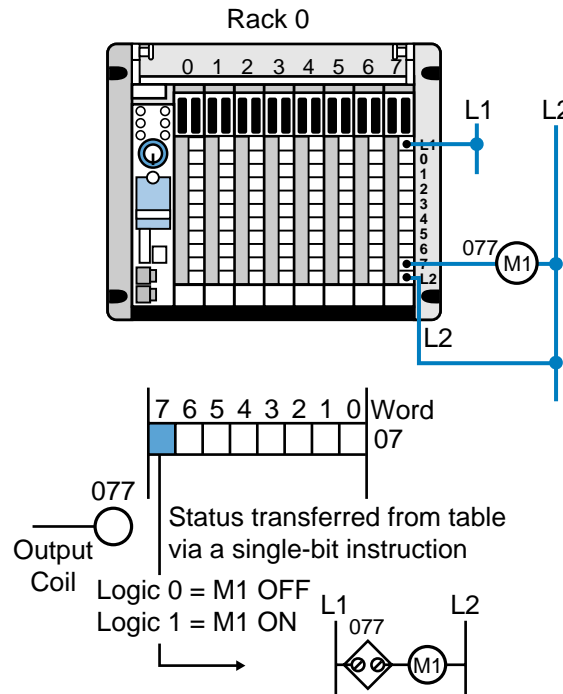

Figure 6-32. An 8-bit output image table with the module's L2 connection completing the path from L1 to L2.

Multibit outputs, such as BCD register outputs, use functional block instructions (e.g., block transfer out) to output a word or register to the module (see Figure 6-33). These instructions, in conjunction with input instructions, are heavily utilized during the programming and control of discrete I/O signals. Chapter 9 provides more information about the use and operation of functional block instructions.
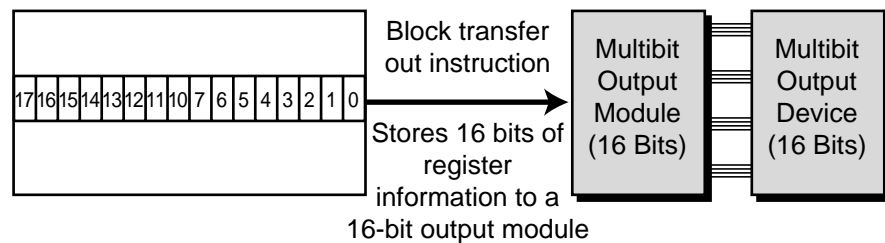

Figure 6-33. Functional block instruction transferring the output register contents to the module.

EXAMPLE 6-3

For the rack configuration shown in Figure 6-34, determine the addresses for each of the output field devices wired to the output connections in the 8-bit discrete input module. Assume that the first four slots of this 64 I/O micro-PLC are filled with outputs and that the second four are filled with inputs. The addressing scheme follows a rack-slot-connection convention (like Example 6-1), which starts at I/O address 000. Note that the number system is octal.
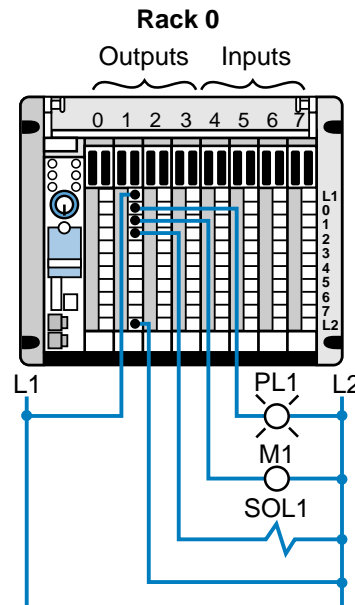
Figure 6-34. Rack configuration for Example 6-3.

SOLUTION

The field devices in this discrete output module will have addresses 010 through 017 because the module is located in rack 0, slot number 1 and the 8 field devices are connected to bits 0 through 7. Therefore, each of the field output devices will have the addresses shown in Figure 6-35—PL1 will be known as output 010, M1 as 011, and SOL1 as 012. Every time a bit address becomes 1, the field device with the corresponding address will be turned ON.
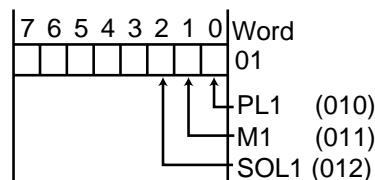
Figure 6-35. Field device addresses for the outputs in Example 6-3.

If M1 is rewired to another connection in another discrete output, the address that turns it ON and OFF will change. Consequently, the control program must be changed, since there can be only one reference address per discrete field output device connection.

6-7 DISCRETE OUTPUTS

Discrete output modules receive their necessary voltage and current from their enclosure's back plane (see Chapter 4 for loading considerations). The field devices with which discrete output modules interface may differ in their voltage requirements; therefore, several types and magnitudes of voltage are provided to control them (e.g., 120 VAC, 12 VDC). Table 6-4 illustrates some typical output field devices, while Table 6-5 lists the standard output ratings found in discrete output applications.

Output Devices
Alarms
Control relays
Fans
Horns
Lights
Motor starters
Solenoids
Valves

Table 6-4. Output field devices.

Output Ratings
12–48 volts AC/DC
120 volts AC/DC
230 volts AC/DC
Contact (relay)
Isolated output
TTL level
5–50 volts DC (sink/source)

Table 6-5. Standard output ratings.

AC OUTPUTS

AC output circuits, like input circuits, vary widely among PLC manufacturers, but the block diagram shown in Figure 6-36 depicts their general configuration. This block configuration shows the main sections of an AC output module, along with how it operates. The circuit consists primarily of the logic and power sections, coupled by an isolation circuit. An output interface can be thought of as a simple switch (see Figure 6-37) through which power can be provided to control an output device.
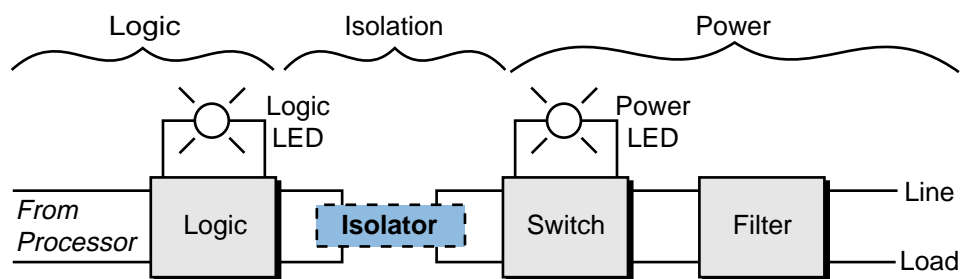

Figure 6-36. AC output circuit block diagram.

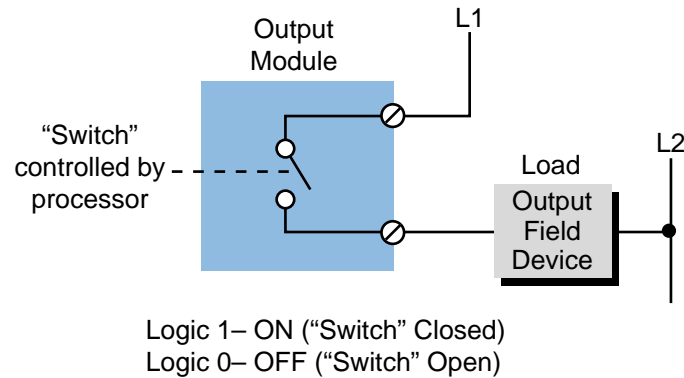Logic 1– ON ("Switch" Closed)
Logic 0– OFF ("Switch" Open)

Figure 6-37. "Switch" function of an output interface.

During normal operation, the processor sends an output's status, according to the logic program, to the module's logic circuit. If the output is to be energized (reflecting the presence of a 1 in the output table), the logic section of the module will latch, or maintain, a 1. This sends an ON signal through the isolation circuit, which in turn, switches the voltage to the field device through the power section of the module. This condition will remain ON as long as the output table's corresponding image bit remains a 1. When the signal turns OFF, the 1 that was latched in the logic section unlatches, and the OFF signal passed through the isolation circuit provides no voltage to the power section, thus de-energizing the output device. Figure 6-38 illustrates a typical AC output circuit.
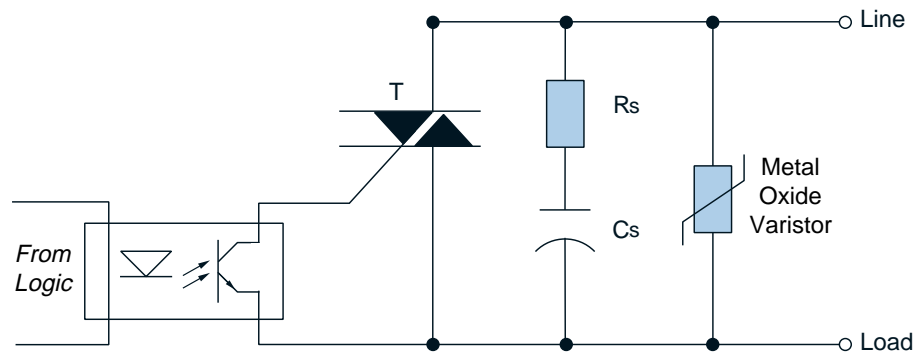

Figure 6-38. Typical AC output circuit.

The switching circuit in the power section of an AC output module uses either a triac or a silicon controlled rectifier (SCR) to switch power. The AC switch is normally protected by an RC snubber and/or a metal oxide varistor (MOV), which limits the peak voltage to some value below the maximum rating. Snubber and MOV circuits also prevent electrical noise from affecting the circuit operation. Furthermore, an AC output circuit may contain a fuse that prevents excessive current from damaging the switch. If the circuit does not contain a fuse, the user should install one that complies with the manufacturer's specifications.

As with input circuits, AC output interfaces may have LEDs to indicate operating logic signals and power circuit voltages. If the output circuit contains a fuse, it may also have a fuse status indicator. Figure 6-39 illustrates an AC output connection diagram. Note that power from the field (L1) supplies the voltage that the module uses to turn ON the output devices. Chapter 20 discusses other considerations for connecting AC outputs.
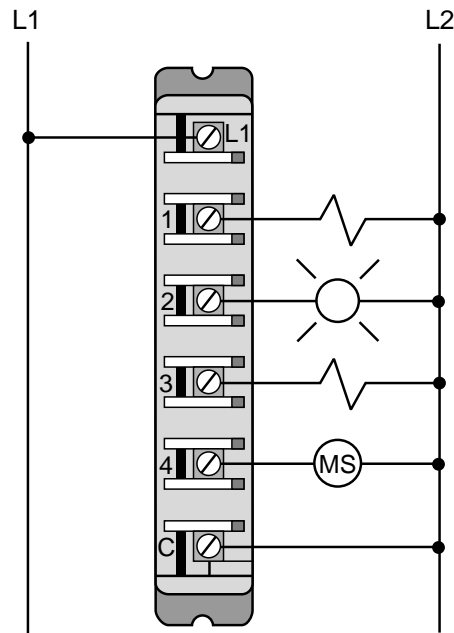

Figure 6-39. AC output module connection diagram.

DC OUTPUTS (SINK/SOURCE)

DC output interfaces control discrete DC loads by switching them ON and OFF. The functional operation of a DC output is similar to that of an AC output; however, the DC output's power circuit employs a power transistor to switch the load. Like triacs, transistors are also susceptible to excessive applied voltages and large surge currents, which can cause overdissipation and short-circuit conditions. To prevent these conditions, a power transistor is usually protected by a freewheeling diode placed across the load (field output device). DC outputs may also incorporate a fuse to protect the transistor during moderate overloads. These fuses are capable of opening, or breaking continuity, quickly before excessive heat due to overcurrents occurs.

As in DC inputs, DC output modules may have either sinking or sourcing configurations. If a module has a sinking configuration, current flows *from the load* into the module's terminal, switching the negative (return or common) voltage to the load. The positive current flows from the load to the common via the module's power transistor.

SECTION
2

Components
and Systems

The Discrete
Input/Output System

CHAPTER
6

In a sourcing module configuration, current flows *from the module* into the load, switching the positive voltage to the load. Figure 6-40 illustrates a typical sourcing DC output circuit, and Figure 6-41 shows device connections for both sourcing and sinking configurations. Note that in sinking output devices, current flows into the device's terminal from the module (the module provides, or sources, the current). Conversely, the current in sourcing output devices flows out of the device's terminal into the module (the module receives, or sinks, the current).
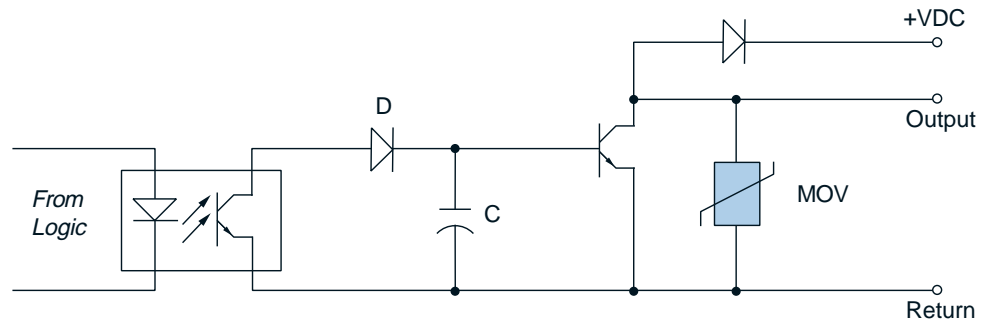

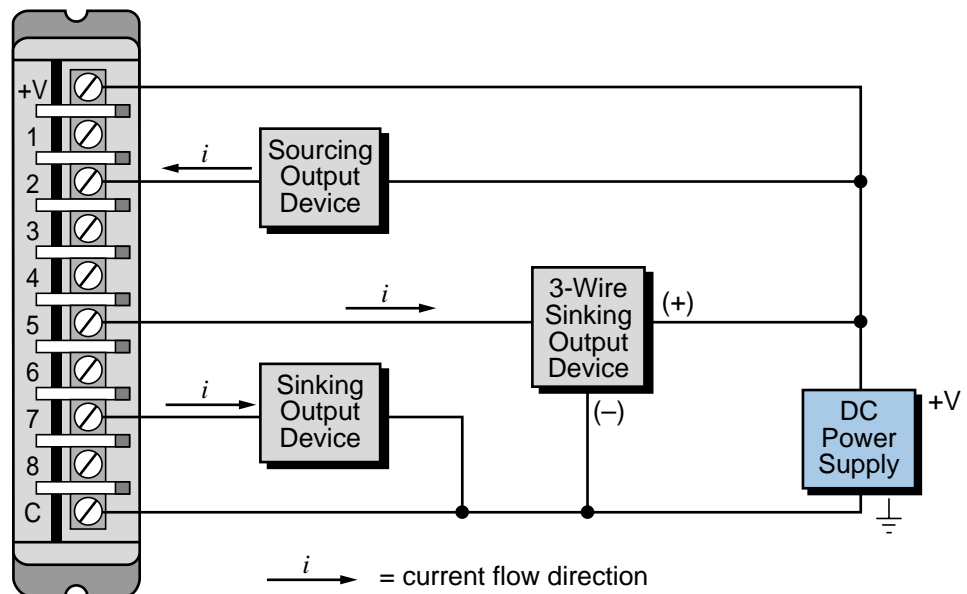**Figure 6-40.** Typical sourcing DC output circuit.

Figure 6-41. Field device connections for a sinking/sourcing DC output module.

ISOLATED AC AND DC OUTPUTS

Isolated AC and DC outputs operate in the same manner as standard AC and DC output interfaces. The only difference is that each output has its own return line circuit (common), which is isolated from the other outputs. This configuration allows the interface to control output devices powered by different sources, which may also be at different ground (common) levels.

A standard, nonisolated output module has one return connection for all of its outputs; however, some modules provide one return line per four outputs if the interface has eight or more outputs. Isolated interfaces provide less modularity (i.e., fewer points per module) than their standard counterparts, because extra terminal connections are necessary for the independent return lines. Figure 6-42 illustrates connections to an isolated AC output interface.
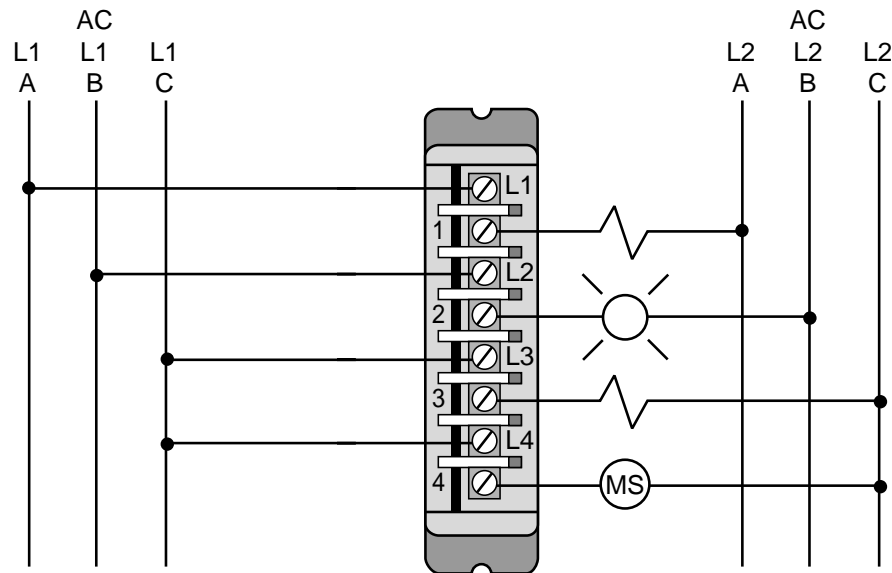

Figure 6-42. Connection diagram for an isolated AC output interface.

TTL OUTPUTS

TTL output interfaces allow a PLC to drive output devices that are TTL compatible, such as seven-segment LED displays, integrated circuits, and 5-VDC devices. Most of these modules require an external 5-VDC power supply with specific current requirements, but some provide the 5-VDC source voltage internally from the back plane of the rack. TTL modules usually have eight available output terminals; however, high-density TTL modules may be connected to as many as sixteen devices at a time. Typical output devices that use high-density TTL modules are 5-volt seven-segment indicators. Figure 6-43 illustrates typical output connections to a TTL output module. A TTL output interface requires an external power supply.

REGISTER/BCD OUTPUTS

Multibit **register/BCD output interfaces** provide parallel communication between the processor and an output device, such as a seven-segment LED display or a BCD alphanumeric display. Register output interfaces may also drive small DC loads with low current requirements (0.5 amps). Register
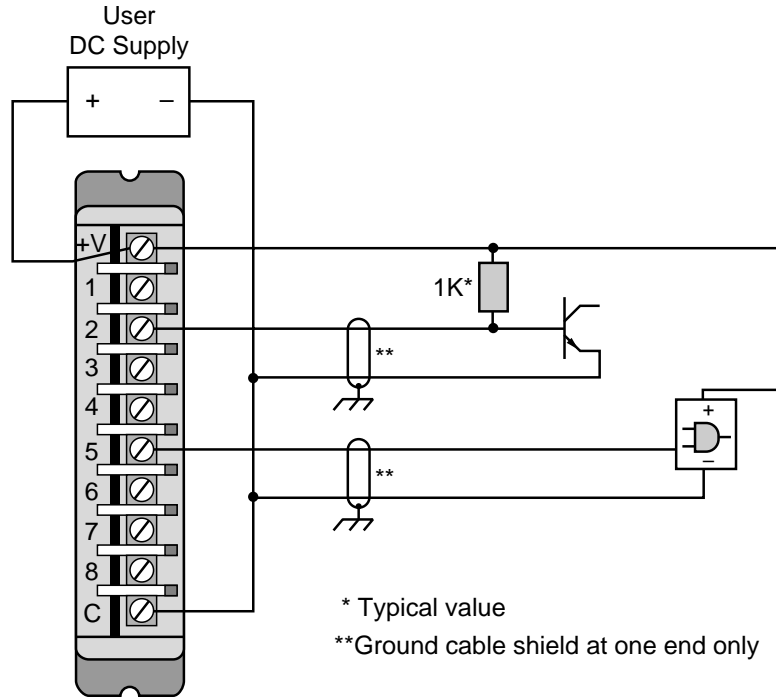
Figure 6-43. Connection diagram for a TTL output module.

output interfaces provide voltages ranging from 5 VDC (TTL level) to 30 VDC and have 16 or 32 output lines (one or two I/O registers). Figure 6-44 illustrates a typical device interface connection for a register output module.
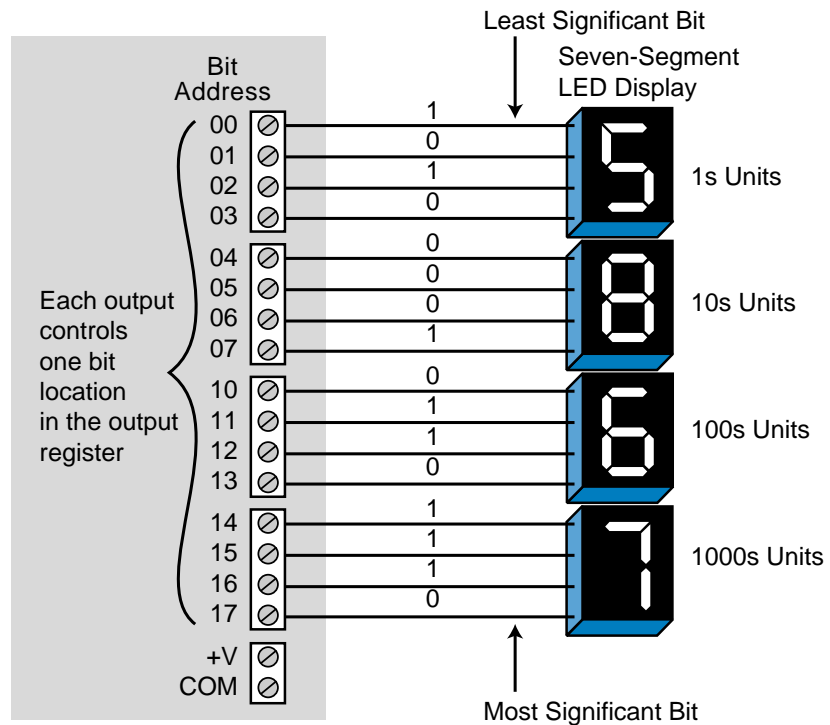

Figure 6-44. Register/BCD output interface connected to seven-segment indicators.

In a register output module, the information sent to the module originates in the register storage data table (see Figure 6-45). A 16-bit word or register is sent from this table to the module address specified by the data transfer or I/O register instruction (e.g., block transfer out). Once the data arrives at the module, it is latched and made available at the output circuits.
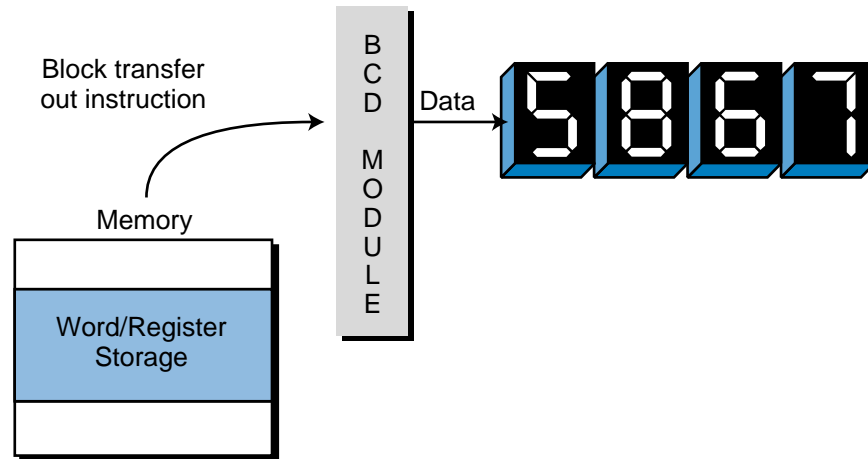

Figure 6-45. Output data table sending a 16-bit word to a register output module.

Register output modules may also have multiplexing capabilities (see Figure 6-46). As is the case with multiplexed inputs, multiplexed output devices (e.g., BCD display digits) require enable line capability to select the BCD display group that will receive the parallel, 16-bit data from the module (see Figure 6-47). A single-digit seven-segment display will be able to receive data if the enable is selected. Conversely, if the enable is not selected, the
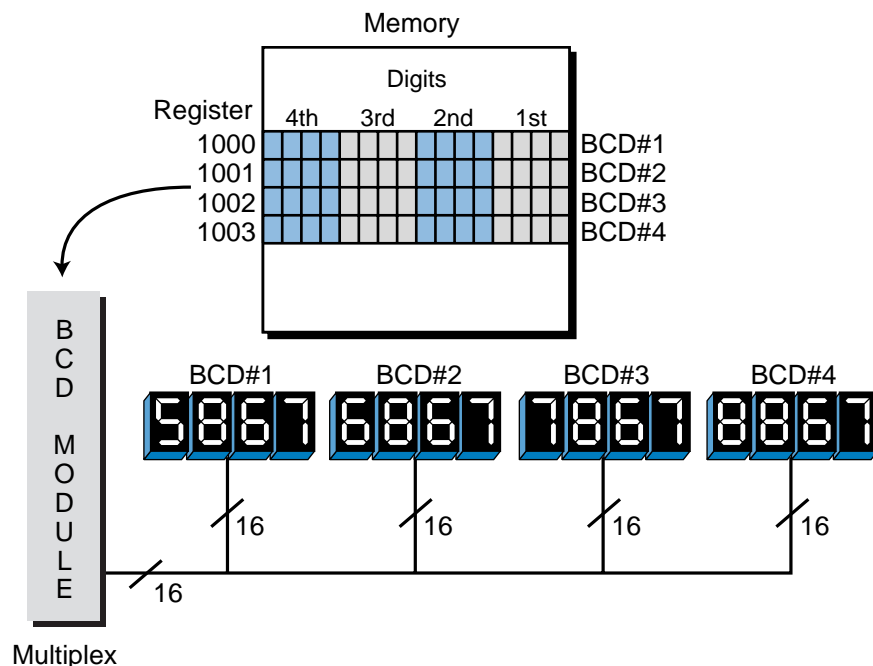

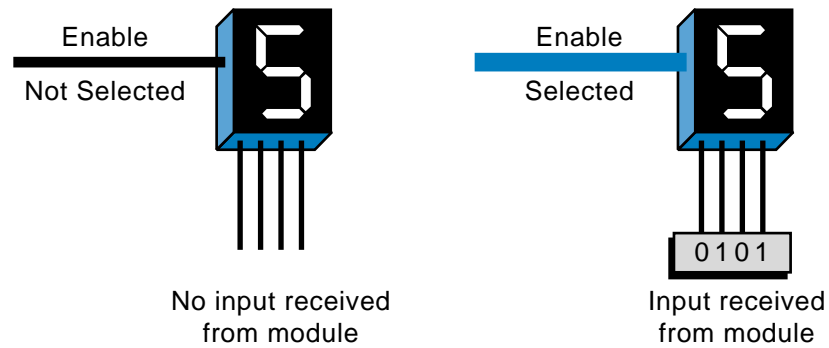**Figure 6-46.** Multiplexed output module.

Figure 6-47. Single-digit seven-segment BCD display with enable line.

display will be blank or will contain the last data that was latched, because it may latch the data until the enable reselected and new data is available. If the BCD display contains four digits and one enable line (see Figure 6-48), the operation will be the same, except that the enable will control all four displays. With this option, one interface can control several groups of 16 or 32 outputs, depending on the modularity. For example, if a multiplexed output can handle four sets of 16-bit outputs, then it can drive up to four sets of 4-digit seven-segment indicators. Register data from the output table is sent to the module once a scan, updating each multiplexed set of output devices.
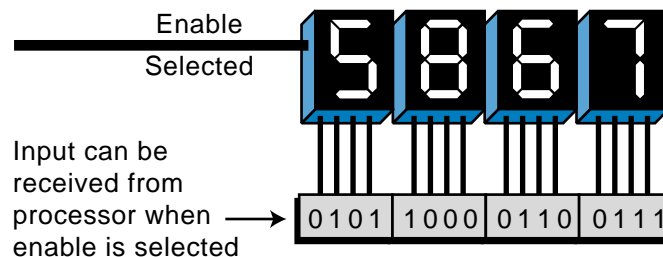

Figure 6-48. A 4-digit seven-segment BCD display with one common enable line.

The use of multiplexed outputs does not require special programming, since there are output instructions that specify the multiplexing operation. The only requirement is that the output devices (e.g., LED displays) must possess enable circuits allowing the module to connect the enable lines to each set of loads controlled by each set of 16 bits. Figure 6-49 shows a block diagram of a multiplexed output module with four sets of seven-segment LED indicators.

If output modules with enable lines are multiplexed, only passive-type output devices (i.e., seven-segment indicators, displays, etc.), as opposed to control-type elements (i.e., low-current solenoids), can be controlled. The reason for this is that while multiplexed outputs are very useful, their output data does not remain static for one channel, or set, of 16 bits or 32 bits; it changes for each circuit that is being multiplexed. The only way to use multiplexed
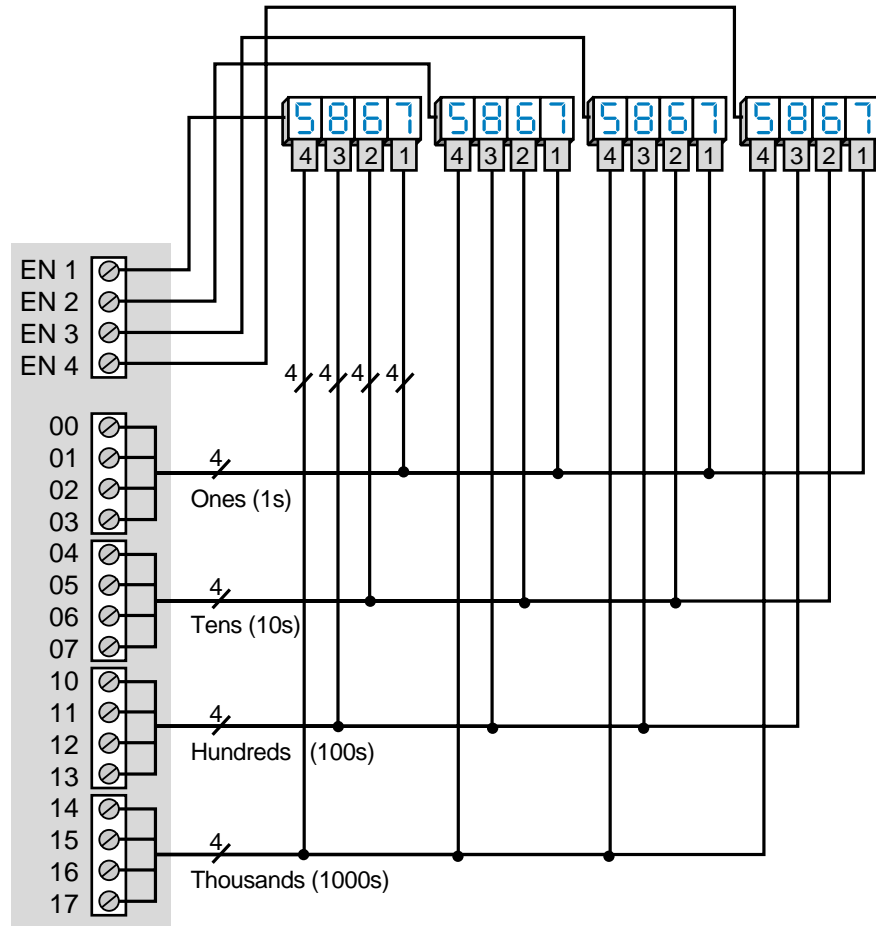
Figure 6-49. A multiplexed output module with four sets of LED displays.

modules and still have correctly operating output devices is to incorporate additional latching/enabling circuits into the output devices' hardware (see Figure 6-50). Such a situation may be encountered in the transmission of parallel data to instrumentation or computing devices that have enable and latching lines for incoming data.
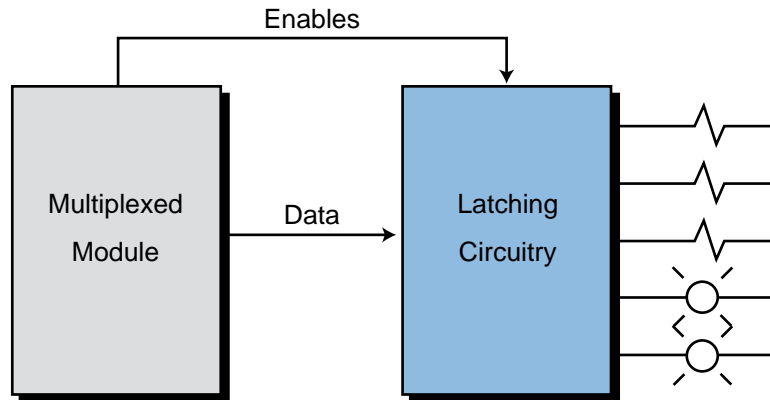

Figure 6-50. Latching/enabling circuit.

EXAMPLE 6-4

Assume that the contents of the registers in the storage table shown in Figure 6-51 are transferred to a BCD multiplexed module and, subsequently, to a BCD display. **(a)** What will be the value displayed on the seven-segment indicators during the third scan as shown in the timing diagram in Figure 6-52? **(b)** Also indicate, using Figure 6-49 as a reference, the lines (e.g., enable bits 0–17) that will be active during the third-scan transfer.

Word of Register																	Contents in BCD
4000	0	0	1	1	0	1	0	1	1	0	0	0	0	0	1	1	3583
4001	0	0	1	0	0	1	0	1	0	1	1	1	1	0	0	0	2578
4002	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	3810
4003	0	1	0	0	1	0	0	1	0	0	0	1	0	1	1	1	4917

Figure 6-51. Storage table for Example 6-4.
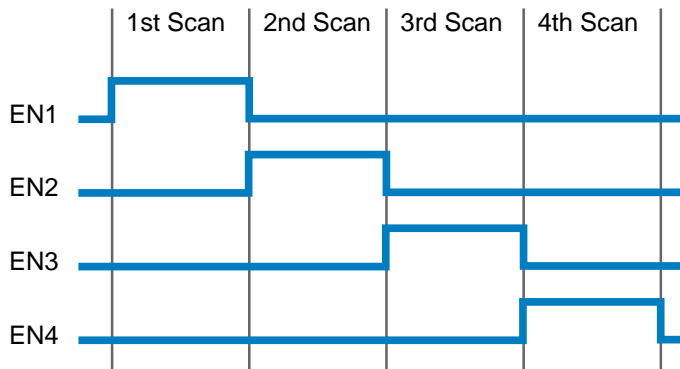
Figure 6-52. Timing diagram of enable signals from a BCD multiplexed module.

SOLUTION

(a) During the third scan (see Figure 6-53), the enable line EN3 will be ON, allowing the BCD data 3810 to go to BCD set #3. The value of register 4002 will be sent to the module through the wires connected to it. Since only BCD set #3 is enabled, it will accept all of the signals.

(b) The active lines, including the enable, are shown in blue. Note that in the other BCD sets, the BCD values from each set's respective register are shown in gray. These values may remain on the display because they have been latched from previous scans. They are not shown in blue because they are not active.

SECTION
2

Components
and Systems

The Discrete
Input/Output System
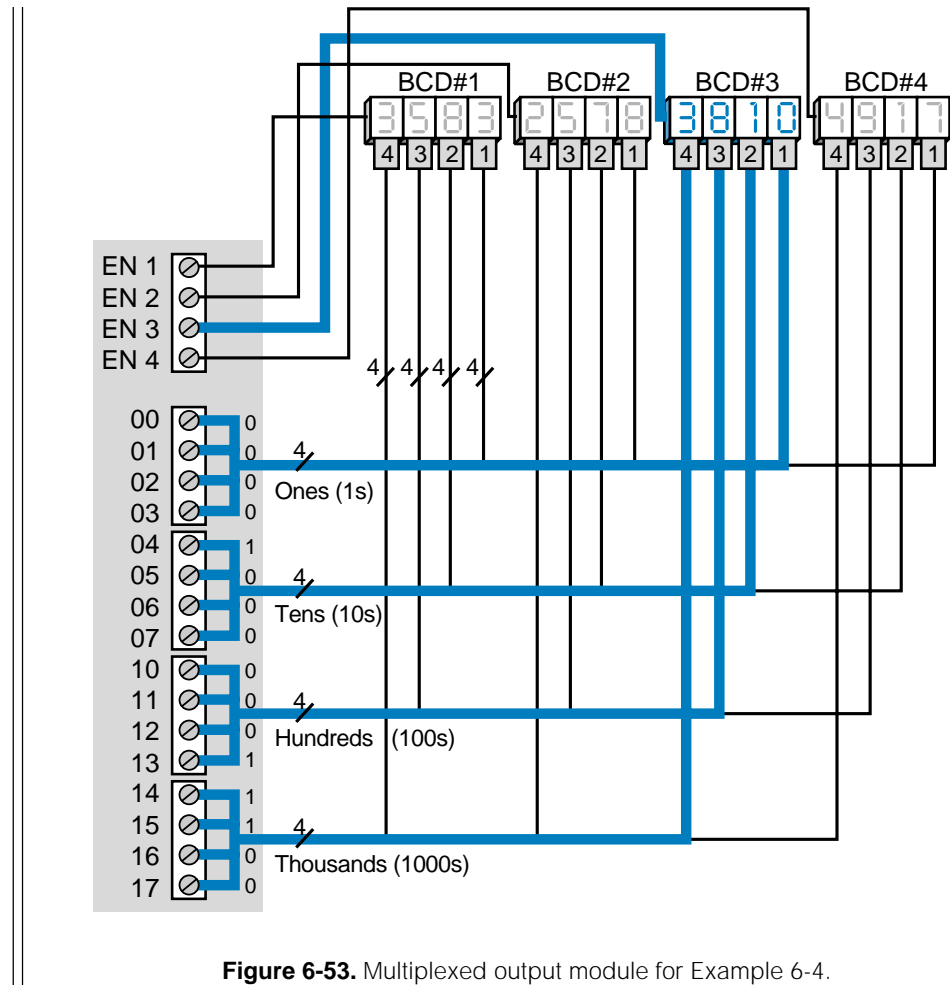
CHAPTER
6

Figure 6-53. Multiplexed output module for Example 6-4.

CONTACT OUTPUTS

Contact output interfaces allow output devices to be switched by normally open or normally closed relay contacts. Contact interfaces provide electrical isolation between the power output signal and the logic signal through separation between contacts and between the coil and contacts. These outputs also include filtering, suppression, and fuses.

The basic operation of contact output modules is the same as that of standard AC or DC output modules. When the processor sends status data (1 or 0) to the module during the output update, the state of the contacts changes. If the processor sends a 1 to the module, normally open contacts close and normally closed contacts open. If the processor sends a 0, no change occurs to the normal state of the contacts.

Contact outputs can be used to switch either AC or DC loads, but they are normally used in applications such as multiplexing analog signals, switching small currents at low voltages, and interfacing with DC drives to control

different voltage levels. High-power contact outputs are also available for applications that require the switching of high currents. Figure 6-54 shows a contact output circuit. The device connection for this output module is similar to an AC output module. In this circuit, one side (1A) goes to L1, while the other (1B) goes to the load.
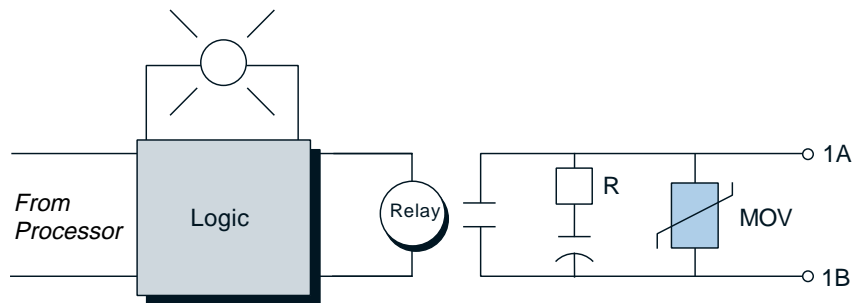
Figure 6-54. Contact output circuit.

Figure 6-55 illustrates an interfacing example where four analog voltage references are connected to a contact output module. These references represent preset speed values, which if connected to a speed drive controller, can be used to switch different motor velocities (e.g., two forward, two reverse). Note that each contact in this interface must be mutually exclusive—that is, only one contact can be closed at a time. Interlocking logic in the control program is necessary to prevent two or more output coils from being energized at the same time.

Figure 6-55. Example of a contact interface connection.

6-8 DISCRETE BYPASS/CONTROL STATIONS

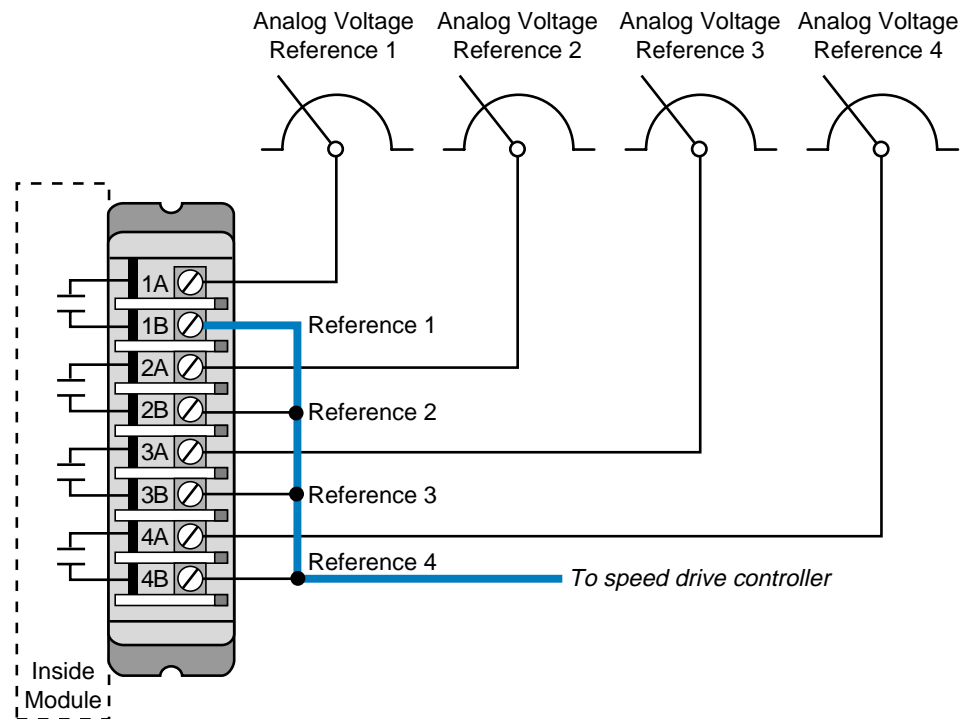**Bypass/control stations** are manual backup devices that are used in PLC systems to allow flexibility during start-up and output failure. By incorporating a selector switch that allows a field output device to be switched ON regardless of the state of its output module, these devices can override a PLC's output signal. Bypass devices can also be configured to place field outputs under PLC output control or to change them to an OFF condition.

Figure 6-56 shows a diagram of a typical bypass device. Bypass units provide 8 to 16 isolated points, each protected by a circuit breaker or fuse, for use with any PLC's discrete output modules. Bypass devices are placed between the PLC's output interface and the digitally controlled element (see Figure 6-57). Indicators, which are incorporated into the control system, show the ON/OFF state of the field device. Bypass units provide a way to control field devices without the PLC. These devices are very useful during maintenance situations, system start-up, and emergency disconnect of particular field devices.
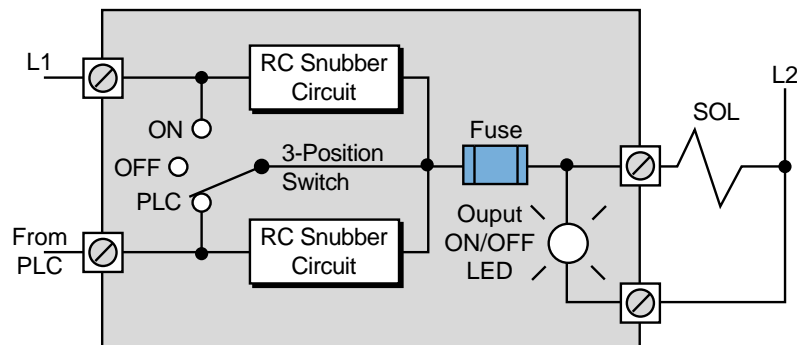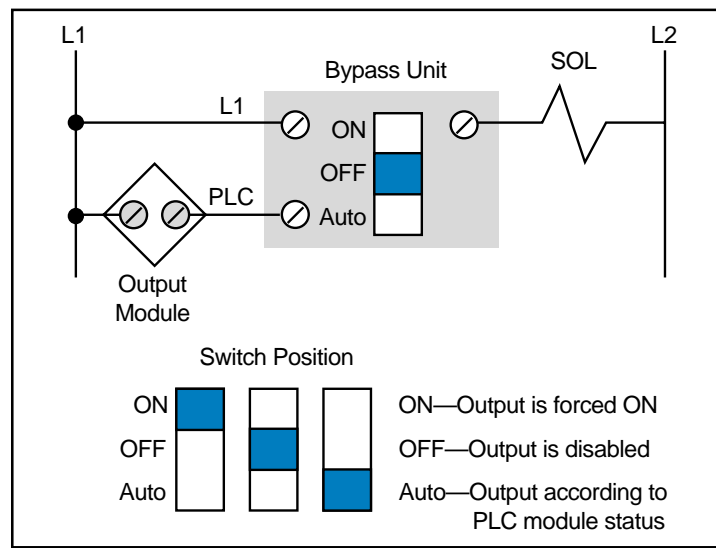
Figure 6-56. Typical bypass device.

Figure 6-57. Bypass unit placed between the PLC and a field device.

6-9 INTERPRETING I/O SPECIFICATIONS

Perhaps with the exception of standard I/O current and voltage ratings, specifications for I/O circuits are all too often treated as a meaningless listing of numbers. Nevertheless, manufacturers' specifications provide valuable information about the correct and safe application of interfaces. These specifications place certain limitations on the module and also on the field equipment that it can operate. Failure to adhere to specifications can result in a misapplication of the hardware, leading to faulty operation or equipment damage. Table 6-6 provides an overview of the electrical, mechanical, and environmental specifications that should be evaluated for each PLC application. Following is a more detailed explanation of each specification. These specifications should also be evaluated for the interfaces covered in the next two chapters (analog and special function).

ELECTRICAL

Input Voltage Rating. This AC or DC value defines the magnitude and type of signal that will be accepted by the circuit. The circuit will usually accept a deviation from this nominal value of ±10–15%. This specification may also be called the *input voltage range*. For a 120 VAC–rated input circuit with a range of ±10%, the minimum and maximum acceptable input voltages for continuous operation will be 108 VAC and 132 VAC, respectively.

Input Current Rating. This value defines the minimum input current at the rated voltage that the input device must be capable of driving to operate the input circuit. This specification may also appear indirectly as the *minimum power requirement*.

Input Threshold Voltage. This value specifies the voltage at which the input signal is recognized as being absolutely ON. This specification is also called the *ON threshold voltage*. Some manufacturers also specify an OFF voltage, defining the voltage level at which the input circuit is absolutely OFF.

Input Delay. The input delay defines the duration for which the input signal must exceed the ON threshold before being recognized as a valid input. This specification is given as a minimum or maximum value. This delay is a result of filtering circuitry provided to protect against contact bounce and voltage transients. The input delay is typically 9–25 msec for standard AC/DC inputs and 1–3 msec for TTL or electronic inputs.

Output Voltage Rating. This AC or DC value defines the magnitude and type of voltage source that the I/O module can control. Deviation from this nominal value is typically ±10–15%. For some output interfaces, the output voltage is also the maximum continuous voltage. The output voltage

ELECTRICAL

Input Voltage Rating. An AC or DC value that specifies the magnitude and type of signal a circuit will accept.

Input Current Rating. The minimum current at the rated voltage an input device must be capable of driving.

Input Threshold Voltage. The voltage at which an input signal is recognized as being ON.

Input Delay. The duration for which an input signal must be ON to be recognized as a valid input.

Output Voltage Rating. An AC or DC value that specifies the magnitude and type of voltage that an I/O module can control.

Output Current Rating. The maximum current that a single output circuit can safely carry under load.

Output Power Rating. The maximum power an output module can dissipate with all circuits energized.

Current Requirements. The current demand that an I/O module places on the system power supply.

Surge Current (Max). The maximum current and duration for which an output circuit can exceed its maximum ON-state current rating.

OFF-State Leakage Current. The maximum leakage current that flows through the triac/transistor during its OFF state.

Output ON-Delay. The response time for an output to turn from OFF to ON after it receives an ON command.

Output OFF-Delay. The response time for an output to turn from ON to OFF after it receives an OFF command.

Electrical Isolation. A maximum value in volts defining the isolation between the I/O circuit and the controller logic.

Output Voltage/Current Ranges. The value of the voltage/current swing of the digital-to-analog converter.

Input Voltage/Current Ranges. The value of the voltage/current swing of the analog-to-digital converter.

Digital Resolution. A measure of how closely the converted analog I/O current or voltage signal approximates the actual analog value.

Output Fuse Rating. The type and rating of fuses that should be used in the interface.

MECHANICAL

Points Per Module. The number of input or output circuits that are on a single module.

Wire Size. The number of conductors and the largest gauge wire the I/O termination points will accept.

ENVIRONMENTAL

Ambient Temperature Rating. The maximum air temperature surrounding the I/O system for ideal operating conditions.

Humidity. The maximum air humidity surrounding the I/O system.

Table 6-6. Summary of I/O specifications.

specification may also be stated as the *output voltage range*, in which case both the minimum and maximum operating voltages are given. An output circuit rated at 48 VDC, for example, can have an absolute working range of 42 to 56 VDC.

Output Current Rating. This specification is also known as the *ON-state continuous current rating*, a value that defines the maximum current that a single output circuit can safely carry under load. The output current rating is a function of the electrical and heat dissipation characteristics of the component. This rating is generally specified at an ambient temperature (typically 0–60°C). As the ambient temperature increases, the output current decreases. Exceeding the output current rating or oversizing the manufacturer's fuse rating can result in a permanent short-circuit failure or other damage.

Output Power Rating. This maximum value defines the total power that an output module can dissipate with all circuits energized. The output power rating for a single energized output is the product of the output voltage rating and the output current rating expressed in volt-amperes or watts (e.g., 120 V × 2 A = 240 VA). This value for a given I/O module may or may not be the same if all outputs on the module are energized simultaneously. The rating for an individual output when all other outputs are energized should be verified with the manufacturer.

Current Requirements. The current requirement specification defines the current demand that a particular I/O module's logic circuitry places on the system power supply. To determine whether the power supply is adequate, add the current requirements of all the installed modules that the power supply supports, and compare the total with the maximum current the power supply can provide. The current requirement specification will provide a typical rating and a maximum rating (all I/O activated). An insufficient power supply current can result in an undercurrent condition, causing intermittent operation of field input and output interfaces.

Surge Current (Max). The surge current, also called the *inrush current*, defines the maximum current and duration (e.g., 20 amps for 0.1 sec) for which an output circuit can exceed its maximum ON-state continuous current rating. Heavy surge currents are usually a result of either transients on the output load or power supply line or the switching of inductive loads. Freewheeling diodes, Zener diodes, or RC networks across the load terminals normally provide output circuits with internal protection. If not, protection should be provided externally.

OFF-State Leakage Current. Typically, this is a maximum value that measures the small leakage current that flows through the triac/transistor during its OFF state. This value normally ranges from a few microamperes to a few milliamperes and presents little problem. It can present problems when switching very low currents or can give false indications when using a sensitive instrument, such as a volt-ohm meter, to check contact continuity.

Output-ON Delay. This specification defines the response time for the output to go from OFF to ON once the logic circuitry has received the command to turn ON. The ON response time of the output circuit affects the total time required to activate an output device. The worst-case time required to turn an output device ON after the control logic goes TRUE is the total of the two program scan times plus the I/O update, output-ON delay, and device-ON response times.

Output-OFF Delay. The output-OFF delay specification defines the response time for the output to go from ON to OFF once the logic circuitry has received the command to turn OFF. The OFF response time of the output circuit affects the total time required to deactivate an output device. The worst-case time required to turn an output device OFF after the control logic goes FALSE is the total of the two program scan times plus the I/O update, output-OFF delay, and device-OFF response times.

Electrical Isolation. This maximum value in volts defines the isolation between the I/O circuit and the controller logic circuitry. Although this isolation protects the logic side of the module from excessive input/output voltages or currents, the power circuitry of the module can still be damaged.

Output Voltage/Current Ranges. This specification is a nominal expression of the voltage/current swing of the D/A converter in analog outputs. This output will always be a proportional current or voltage within the output range. A given analog output module may have several hardware- or software-selectable, unipolar or bipolar ranges (e.g., 0 to 10 V, –10 to +10 V, 4 to 20 mA).

Input Voltage/Current Ranges. This specification defines the voltage/current swing of the A/D converter in analog inputs. This specification will always be a proportional current or voltage within the input range. A given analog input module may have several hardware- or software-selectable, unipolar or bipolar ranges (e.g., 0 to 10 V, –10 to +10 V, 4 to 20 mA).

Digital Resolution. This specification defines how closely the converted analog input/output current or voltage signal approximates the actual analog value within a specified voltage or current range. Resolution is a function of the number of bits used by the A/D or D/A converter. An 8-bit converter has a resolution of 1 part in 2^8 or 1 part in 256. If the range is 0 to 10 V, then the resolution is 10 divided by 256, or 40 mV/bit.

Output Fuse Rating. Fuses are often supplied as a part of the output circuit, but only to protect the semiconductor output device (triac or transistor). The manufacturer carefully selects the fuse that is employed or recommended for the interface based on the fusing current rating of the output switching device. Fuse rating incorporates a fuse opening time along with a current overload rating, which allows opening within a time frame that will avoid damage to the triac or transistor. The recommended specifications should be followed when replacing fuses or when adding fuses to the interface.

MECHANICAL

Points Per Module. This specification defines the number of input/output circuits that are on a single module (encasement). Typically, a module will have 1, 2, 4, 8, or 16 points per module. The number of points per module has two implications that may be of importance to the user. First, the less dense (fewer the number of points) a module is, the greater the space requirements are; second, the higher the density, the lower the likelihood that the I/O count requirements can be closely matched with the hardware. For example, if a module contains 16 points and the user requires 17 points, two modules must be purchased. Thus, the user must purchase 15 extra inputs or outputs.

Wire Size. This specification defines the number of conductors and the largest gauge wire that the I/O termination points will accept (e.g., two #14 AWG). The manufacturer does not always provide wire size specifications, but the user should still verify it.

ENVIRONMENTAL

Ambient Temperature Rating. This value is the maximum temperature of the air surrounding the input/output system for best operating conditions. This specification considers the heat dissipation characteristics of the circuit components, which are considerably higher than the ambient temperature rating itself. The ambient temperature rating is much less than the heat dissipation factors so that the surrounding air does not contribute to the heat already generated by internal power dissipation. The ambient temperature rating should never be exceeded.

Humidity Rating. The humidity rating for PLCs is typically 0–95% noncondensing. Special consideration should be given to ensure that the humidity is properly controlled in the area where the input/output system is installed. Humidity is a major atmospheric contaminant that can cause circuit failure if moisture is allowed to condense on printed circuit boards.

Proper observance of the specifications provided on the manufacturer's data sheets will help to ensure correct, safe operation of control equipment. Chapter 20 discusses other considerations for properly installing and maintaining input/output systems.

6-10 SUMMARY OF DISCRETE I/O

For the most part, all PLC system applications require the types of discrete I/O interfaces covered in this chapter. In addition to discrete interfaces, some PLC applications require analog and special I/O modules (covered in the next two chapters) to implement the required control.

All I/O interfaces accept input status data for the input table and accept processed data from the output table. This information is placed in or written from the I/O table (in word locations) according to the location or address of the modules. This address depends on the module's placement in the I/O rack enclosure; therefore, the placement of I/O interfaces is an important detail to keep in mind.

The software instructions that are generally used with discrete-type interfaces are basic relay instructions (ladder type), although multibit modules use functional block instructions as well as some advanced ladder functions. Chapter 9 explains these software instructions. Figure 6-58 shows several programmable controller input and output modules and enclosures.
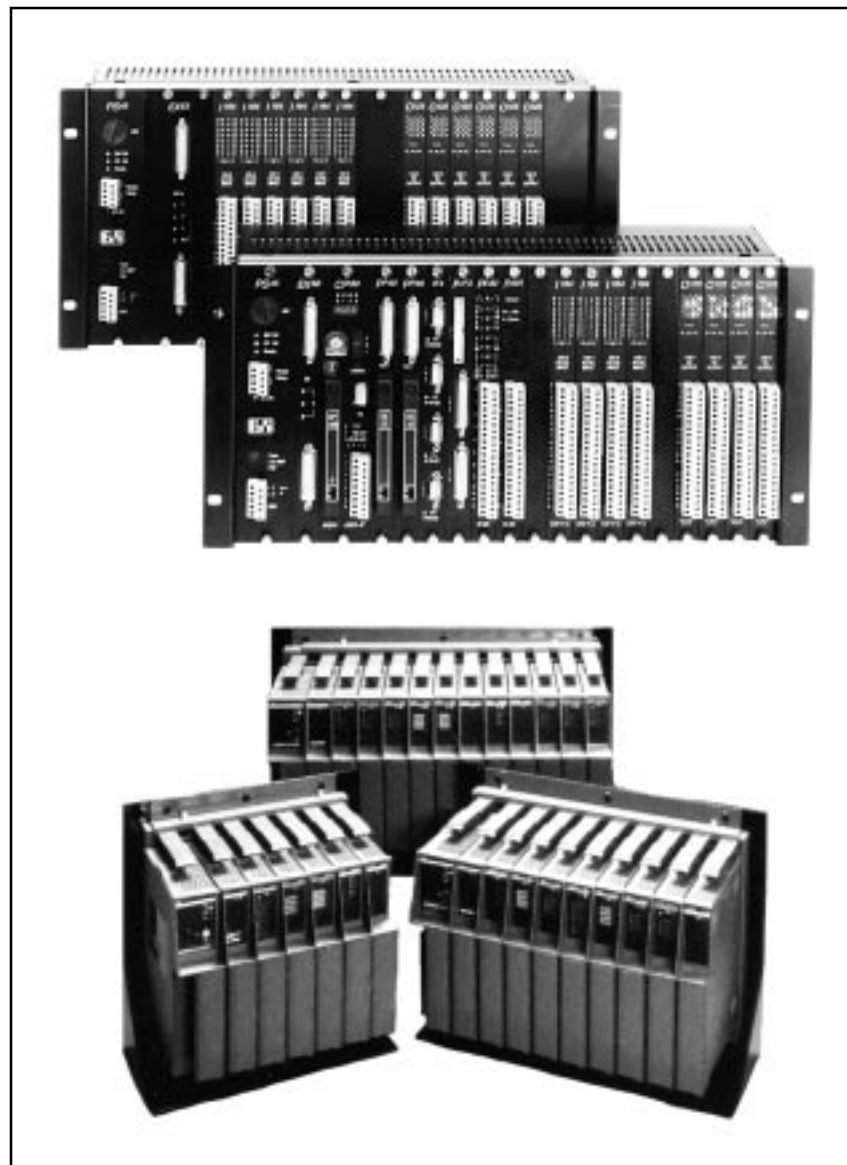
Figure 6-58. PLC families sharing the use of I/O modules and enclosures.

KEY
TERMS

AC/DC I/O interface
bypass/control station
contact output interface
DC I/O interface
digital signal
discrete input interface
discrete output interface
I/O address
I/O module
isolated I/O interface
local rack
master rack
multiplexing
rack enclosure
register/BCD I/O interface
remote I/O subsystem
remote rack
sinking configuration
sourcing configuration
TTL I/O interface

Basic Introduction to PLCs Video Training Series

No matter which brand of PLC you use, this is the perfect introductory series.

Industrial Text & Video's ***Basic Introduction to PLCs Video Training Series*** is a complete PLC course designed for electrical maintenance personnel. It covers the basic operation of PLCs, as well as all the critical maintenance functions, such as troubleshooting the PLC and I/O system and preventive maintenance.

Highlighting the fundamental points of PLCs, this training series is recommended as a prerequisite to either our ***MicroLogix™ 1000 Controller Total Training Series*** or ***Advanced PLC Video Training Series.***

Many of the specifics presented in the video are taken from our industry-standard reference on PLCs—***Programmable Controllers: Theory and Implementation***, 2nd Edition.

This series provides approximately four hours of instructional material and topics covered are equivalent to a 1-day PLC course. Plus, you can train multiple people by simply ordering additional student kits.

Contents:

- **Two Videos.** Explain the basics of PLCs including addressing and programming instructions as well as installation requirements, troubleshooting procedures, and maintenance how-tos.

- **Video Reference Handbook.** Reinforces the material in the videos and provides review questions and answers.

- **Electronic Courseware Series Software.** Tests people on the material in the videos and validates your training program without any extra work for you.

- **General PLC References.**
  Programmable Controllers: Theory and Implementation, 2nd Edition—the best-selling, award-winning book applicable to all PLCs.

  Programmable Controllers: Workbook/Study Guide—the companion workbook to the industry-standard reference on PLCs.

Benefits:

- **Maintenance Focus.** The videos target electrical maintenance personnel by covering what a PLC is, basic relay and timer instructions, the I/O system, motor controls, troubleshooting, and more.

- **Course Components Work Together.** The videos explain the basics of PLCs. When you need to know more, the reference book will take you as far as you want to go—process control, analog I/O, plant networks, and more. Workbooks provide summaries, questions, and answers for a quick-reference tool, and the software helps to evaluate skills as well as the training program in your plant.

- **Convenience and Flexibility.** Employees can watch a tape, take a review quiz, and be back to work in under two hours.

- **Money-Back Guarantee.** Our program is backed by a 30-day, unconditional, money-back guarantee.

Video Program and Reference Handbook

UNDERSTANDING AND APPLYING PLCs IN ELECTRICAL CONTROLS

"We liked Industrial Text & Video's program so much we bought it immediately. They cover everything that we really need to know."

—George Cooros, PLC Training Instructor, US Steel Co.

This program covers it all—from number systems to general operation to introductory programming. You'll learn how to convert an electromechanical (hardwired) system to a PLC-based system, troubleshoot I/O, and much more.

It's packed full of industry secrets, installation hints, and maintenance how-tos—all to make your job easier. This program gives maintenance personnel the skills necessary to troubleshoot the controller and determine whether the problem is with the controller, the wiring, or a field device. And, the video reference handbook acts as an on-the-job reference tool for all your basic PLC questions.

TAPE 1 CONTENTS (31:33min)

- PLC Principles of Operation
- Ladder diagrams and the PLC
- Ladder symbology and power flow: continuity and logic
- Binary concepts and number systems used in PLCs
- Input and Output (I/O) system addressing and structure
- Discrete I/O modules: interface installation and wiring
- Basic relay instructions

TAPE 2 CONTENTS (33:20 min)

- Timer control: ON-delay and OFF-delay timers
- PLC control circuit examples and implementations: start/stop motor circuit (2-wire and 3-wire control), wiring and interfacing to magnetic starters, reversing motor control and interlocking
- Troubleshooting the CPU and I/O system
- Isolating faults
- Sections to leave hardwired for safety reasons
- PLC preventive maintenance

PLCs—What Everyone Needs to Know

✔ If you're using PLCs to control motors, make sure you're implementing low-voltage protection—we'll show you how and help you avoid potential injury to personnel and equipment

✔ Is your PLC system safely and properly installed—learn which sections of the control circuit must be left hardwired for safety reasons

✔ One small wiring mistake during troubleshooting will make things worse—we'll show you the right way to do it

✔ Follow our step-by-step method of converting a hardwired relay system to a PLC—do it quickly and do it right!

✔ Troubleshoot the complete discrete I/O system—see it graphically—from power line problems to wiring connections to I/O modules to the detection of a bad field device

✔ Does everyone on your staff know how to troubleshoot PLCs and I/O—learn how to use the power of the PLC and its monitoring capabilities to help you

Electronic Courseware Series

As one of the steps in the training process, the Electronic Courseware Series (ECS) software lets your people sharpen and test their knowledge on the material covered in the videos. It consists of two parts—the ECS Trainer for testing and the ECS Administrator for reporting. Once your people use this software, you can be sure they are qualified to work with the PLCs in your facility.

ECS Trainer

Use our exclusive interactive software to train and evaluate everyone going through the videos—without any extra work for you.

The ECS Trainer has practical, real-life questions of varying levels of difficulty. You can use it to test employees as well as prescreen new hires.

It's easy. Just watch one of the videos. Then, take a review quiz to see how much was learned.

Instant feedback. Using A-R-M (Automatic Reinforcement Mode) technology, if someone misses a question, they get instant feedback and receive a follow-up question to make sure they understand the explanation.

ECS Administrator

Produce simple reports to assess the performance of each person going through the *Basic Introduction to PLCs Video Training Series* and validate the entire training process.

No Licensing Fees

Unlike most software packages, our ECS software doesn't have any licensing fees. The program is licensed per facility. This means you can install it on one computer or 1,000 computers. Employees can even install it on their computers at home. We do recommend, however, that you order extra student kits for everyone who will be going through the program.

Features at a glance:

- No licensing fees
- Networkable
- 160 questions
- A-R-M technology
- Training and testing modes
- Random question selection on final exams

Benefits of computer testing:

- Take it anytime you want
- Take it anywhere you want
- Immediate feedback
- Great for refresher training
- Reduced learning time
- Self-paced

Minimum system requirements:

- 486 PC (Pentium recommended)
- Windows™ 3.1 and 4MB RAM or Windows™ 95, 98, or NT and 16MB RAM
- 256-color monitor
- 15MB free hard-drive space
- Mouse

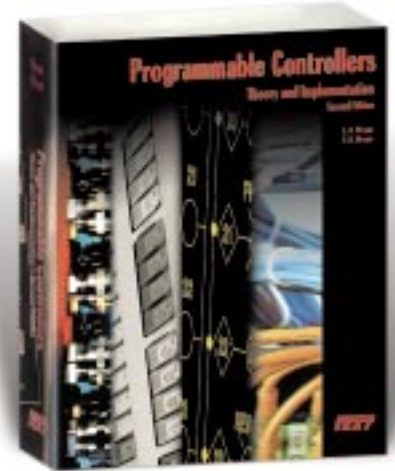*Sample question from the ECS Trainer*
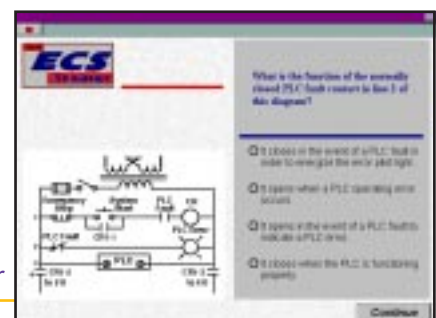
PLC Reference Book

"You covered a huge amount of detail very well. It was very easy to understand."

—Jeff Camp, United Control Corp.

The biggest book on PLCs. Written by industry experts, this book covers important, up-to-date, real-world programmable controller topics and applications. This new edition is completely revised and updated to give you the latest developments and insights from the field. At 5 pounds and 1,035 pages, it puts all the PLC information you need at your fingertips. And, since this is a generic PLC reference, it will help you with all of the different PLCs in your facility.

But, this book is about more than just PLCs—it also thoroughly explains process control, instrumentation, and plant networks. Whether you're already an expert on PLCs or just starting out, our problem-solving approach is guaranteed to help you succeed.

Catalog# ABT-ITV206BOOK *(included in the package)*

21 Chapters of PLC Know-How

TABLE OF CONTENTS

• Valuable Maintenance Tips •

SELECTION, INSTALLATION & SAFETY

✔ Follow our 11 major steps in selecting a PLC for an application and avoid using the wrong controller

✔ Install sinking and sourcing inputs and outputs properly—one wrong wire and it won't work

✔ Implement safety circuits correctly in PLC applications to protect people and equipment

✔ Prevent noise, heat, and voltage variations from ruining your PLC system

✔ Implement a step-by-step static and dynamic start-up checkout to guarantee smooth PLC system operation

✔ Design preventive safety and maintenance into your total control system

TROUBLESHOOTING & MAINTENANCE

✔ Learn no-nonsense troubleshooting procedures to reduce downtime

✔ Troubleshoot analog I/O and avoid undesirable count jumps

✔ Learn 6 preventive maintenance procedures to keep your PLC system running fault free

✔ Learn a step-by-step procedure for finding hidden ground loops

✔ Learn how to deal with leaky inputs

✔ Identify vibration problems and use them for preventive engineering control

✔ Control excessive line voltage and avoid intermittent shutdowns

PROGRAMMING

✔ Learn the number systems and codes used in PLC addressing

✔ Eliminate the confusion of ladder logic programming

✔ Master all types of timers and counters used in real-life applications

✔ Avoid ladder scan evaluation problems

✔ Implement a safe circuit with hardware and software interlocking

Programmable Controllers: Workbook/Study Guide

"Sometimes you think you know it all, but after reading the questions, I oftentimes had to refer back to the theory book."

—Ernest Presto, Electrical Engineer, Polyclad Laminates, Inc.

Imagine having the answers to over 800 PLC problems at your fingertips. That's what you get with *Programmable Controllers: Workbook and Study Guide.* At 334 pages, it's the perfect companion to *Programmable Controllers: Theory and Implementation,* 2nd Edition.

This workbook provides not only valuable summaries of each of the text-book's twenty-one chapters, but also over 800 review questions. And each of the review questions includes a detailed answer and explanation. Use it on the job to brush up on the essentials and to solve any PLC problem.

Whether you're an expert or just learning about PLCs, you'll find plenty to put your skills to the test.

You Will Learn:

Catalog #ABT-ITV206WKBK *(included in the package)*

- Proper address assignment and interfacing
- Basic PLC ladder program implementation
- Data measurement
- Internal coil assignments
- Proper digital and analog interfacing procedures
- Advanced function block programming
- Network protocols
- Analog input and output data handling
- Correct PLC installation

Perfect textbook companion:

- 800 answers to common PLC problems at your fingertips
- Makes a great review tool
- Practice PLC addressing and programming
- Great on-the-job quick-reference guide
- Separate answer section makes quizzing easy
- Valuable chapter summaries
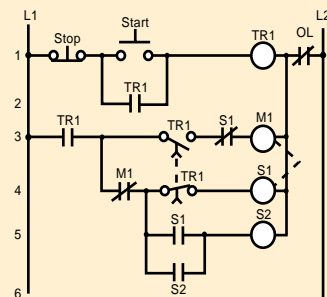
Sample pages from the workbook
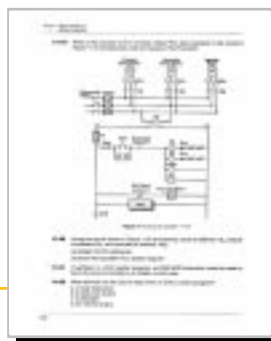
Sample Problem

A sample problem from
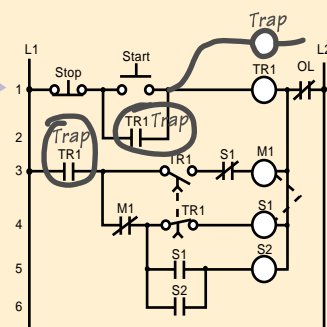Chapter 11 of the workbook:

System Programming and Implementation

Q. Circle the locations where timer traps will be used in the PLC implementation of this reduced-voltage start motor circuit.

A.

• Catalog Numbers •

Description	Catalog #	Price ($US)

Basic Introduction to PLCs Video Training Series includes:
• Two videos
• ECS Software on 3.5" diskettes
• One Student Kit for the *Basic Introduction to PLCs Video Training Series* (ABT-ITV900SK) includes one of each:
  - Video Reference Handbook (ABT-ITV900VRH)
  - Programmable Controllers: Theory and Implementation, 2nd Edition Textbook (ABT-ITV206BOOK)
  - Programmable Controllers: Theory and Implementation, 2nd Edition Workbook/Study Guide (ABT-ITV206WKBK)
  - Course Completion Certificate
	ABT-ITV900D	450

Additional Student Kit for *Basic Introduction to PLCs Video Training Series* includes one of each: • Video Reference Handbook (ABT-ITV900VRH) • Programmable Controllers: Theory and Implementation, 2nd Edition Textbook (ABT-ITV206BOOK) • Programmable Controllers: Theory and Implementation, 2nd Edition Workbook/Study Guide (ABT-ITV206WKBK) • Course Completion Certificate	ABT-ITV900SK	138
Programmable Controllers: Theory and Implementation, 2nd Edition Textbook	ABT-ITV206BOOK	88
Programmable Controllers: Theory and Implementation, 2nd Edition Workbook/Study Guide	ABT-ITV206WKBK	28
Additional Video Reference Handbook for *Basic Introduction to PLCs Video Training Series*	ABT-ITV900VRH	30

Other products available from Industrial Text and Video

Product	See Bulletin #
Advanced PLC Video Training Series Applicable to all PLCs, this series focuses on programming, analog I/O, system implementation, and much more.	**ABT-ITV500**
MicroLogix™1000 Controller Total Training Series All you need to train everyone on the MicroLogix™1000 controller.	**ABT-ITV1761**
Electrical and Motor Controls Video Training Series Get everyone up to speed on electrical maintenance and troubleshooting.	**ABT-ITV700**

For more information on these and other products, please contact your local authorized distributor or visit our website at: www.industrialtext.com

Your authorized distributor is:

All products are backed by a 30-day, unconditional, money-back guarantee.