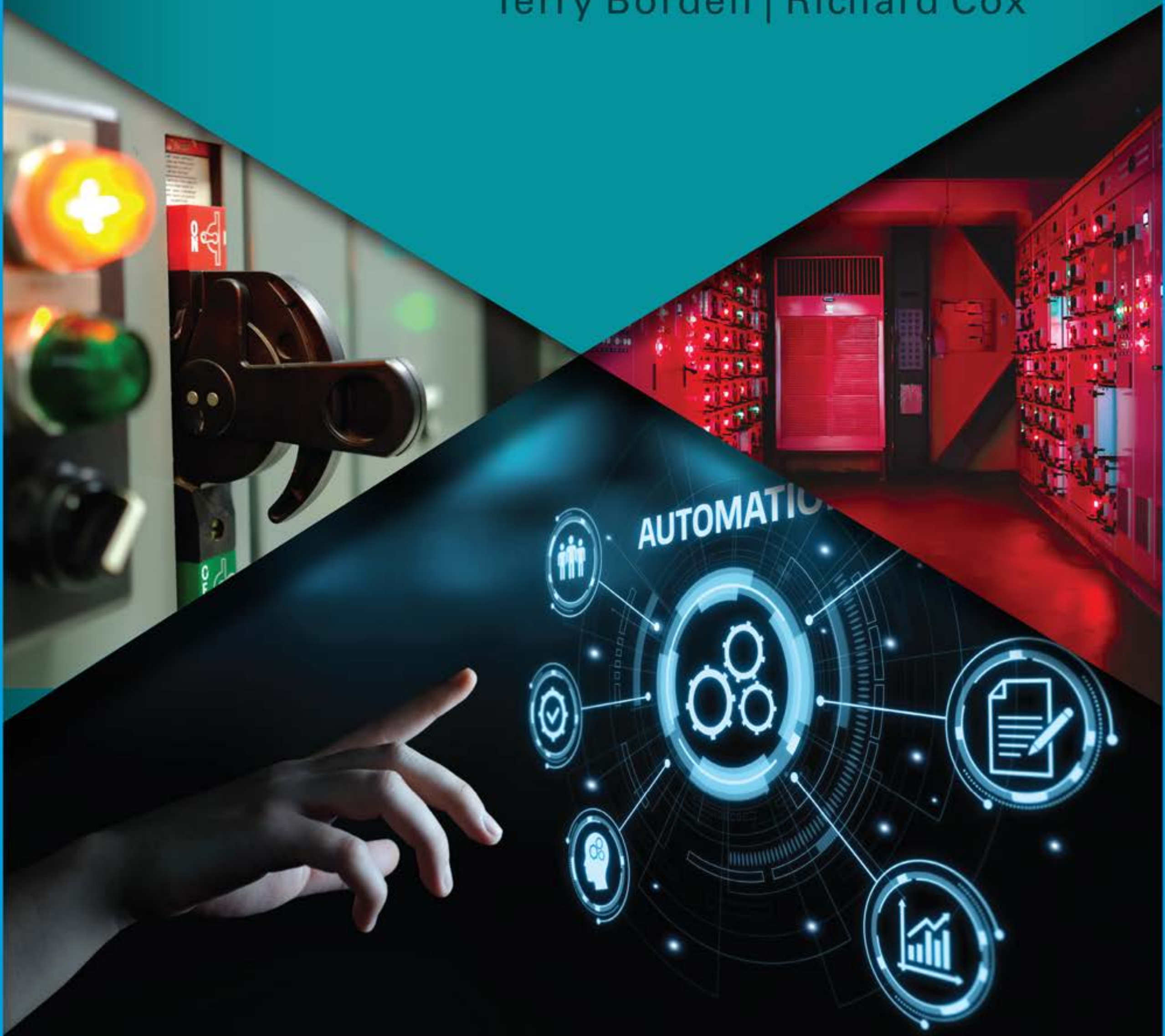
CENGAGE

7th Edition

Technician's Guide to Programmable Controllers

Terry Borden | Richard Cox



7th Edition

Technician's Guide
to Programmable Controllers

Terry R. Borden | Richard A. Cox

CENGAGE

Australia • Brazil • Canada • Mexico • Singapore • United Kingdom • United States

This is an electronic version of the print textbook. Due to electronic rights restrictions, some third party content may be suppressed. Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. The publisher reserves the right to remove content from this title at any time if subsequent rights restrictions require it. For valuable information on pricing, previous editions, changes to current editions, and alternate formats, please visit www.cengage.com/highered to search by ISBN#, author, title, or keyword for materials in your areas of interest.

Important Notice: Media content referenced within the product description or the product text may not be available in the eBook version.

CENGAGE

***Technician's Guide to Programmable Controllers,* 7th Edition**
Terry Borden and Richard Cox

SVP, Higher Education Product Management: Erin Joyner

VP, Product Management, Learning Experiences: Thais Alencar

Product Director: Jason Fremder

Product Manager: Lauren Whalen

Product Assistant: Bridget Duffy

Learning Designer: Elizabeth Berry

Content Manager: Manoj Kumar, Lumina Datamatics, Inc.

Digital Delivery Quality Partner: Elizabeth Cranston

IP Analyst: Ashley Maynard

IP Project Manager: Nick Barrows

Production Service: Lumina Datamatics, Inc

Designer: Felicia Bennett

Cover and Interior Image Source: udorn_1976 /Shutterstock; Alexander Supertramp /Shutterstock; Mr.B-king/Shutterstock

For product information and technology assistance, contact us at **Cengage Customer & Sales Support, 1-800-354-9706 or support.cengage.com.**

For permission to use material from this text or product, submit all requests online at **www.copyright.com.**

Library of Congress Control Number: 2021917612

ISBN: 978-0-357-62249-0

Cengage
200 Pier 4 Boulevard
Boston, MA 02210
USA

Cengage is a leading provider of customized learning solutions with employees residing in nearly 40 different countries and sales in more than 125 countries around the world. Find your local representative at **www.cengage.com.**

To learn more about Cengage platforms and services, register or access your online learning solution, or purchase materials for your course, visit **www.cengage.com.**

Notice to the Reader

Publisher does not warrant or guarantee any of the products described herein or perform any independent analysis in connection with any of the product information contained herein. Publisher does not assume, and expressly disclaims, any obligation to obtain and include information other than that provided to it by the manufacturer. The reader is expressly warned to consider and adopt all safety precautions that might be indicated by the activities described herein and to avoid all potential hazards. By following the instructions contained herein, the reader willingly assumes all risks in connection with such instructions. The publisher makes no representations or warranties of any kind, including but not limited to, the warranties of fitness for particular purpose or merchantability, nor are any such representations implied with respect to the material set forth herein, and the publisher takes no responsibility with respect to such material. The publisher shall not be liable for any special, consequential, or exemplary damages resulting, in whole or part, from the readers' use of, or reliance upon, this material.

Printed in the United States of America
Print Number: 01 Print Year: 2021

Table of Contents

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Chapter 10

Chapter 11

Chapter 12

Chapter 13

Chapter 14

Chapter 15

Chapter 16

Chapter 17

Chapter 18

Chapter 19

Chapter 20

Chapter 21

Chapter 22

PREFACE

Programmable logic controllers, first introduced in 1969, have become an unqualified success. They can be found in almost every industrial and manufacturing facility worldwide. This computer-based device has become the industry standard, replacing the hard-wired electromechanical devices and circuits that had controlled the process machines and driven equipment of industry in the past.

Programmable logic controllers, or **PLCs** as they are referred to, vary in size and sophistication. When PLCs were first introduced, they typically used a dedicated programming device for entering and monitoring the PLC program. The programming device could only be used for programming a specific brand of PLC. These dedicated programmers, while user friendly, were very expensive and could not be used for anything except programming a PLC. Today, personal laptop computers have replaced the dedicated programming devices of the past.

Many electricians and/or technicians seem apprehensive about PLCs and their application in industry. One of the purposes of this text is to explain PLC basics using a plain, easy-to-understand approach so that electricians and technicians with no PLC experience will be more comfortable with their first exposure to PLCs.

Half the battle of understanding any PLC is to first understand the terminology of the PLC world. This text covers terminology, as well as explaining the input/output section, processor unit, memory organization, program instructions, communications, and much more.

A chapter has been included to explain not only ladder diagrams but also Relay Ladder Logic, which is the programming language used in the majority of programmable controllers. Additional chapters are included to cover Function Block, Structured Text, and Sequential Function Chart programming.

Examples of basic programming techniques used with typical PLCs are discussed and illustrated, as are the commonly used commands and functions. There are a variety of PLCs on the market today, and it would be impossible to write a book that explains how they all work and are programmed. Instead, this book is intended to discuss PLCs in a somewhat general, or generic, sense, and to cover the basic concepts of operation that are common to all. All of the examples used in this text are based on the Allen-Bradley Logix 5000 family of PLCs.

New to This Edition: Completely updated with all new color figures and photos, this edition features step-by-step programming examples using the latest Rockwell Automation, Inc. software and hardware. It also features expanded sections on analog configuration and troubleshooting. The introductory chapter on communication networks has been updated to include the latest industrial protocols. Programming instructions along with examples have been included throughout this edition including array and PID instructions.

Although this text only scratches the surface of available instructions and other advanced programming capabilities of the PLCs on the market today, the reader will gain a basic understanding of the most commonly used instructions and how they work. A chapter, “PLC Programming Examples,” will provide the reader with several examples of PLC programming code/logic utilizing many of the instructions covered in previous chapters. The examples are intended to help the reader gain a better understanding of the various PLC instructions and how they can be combined to provide simple control logic solutions.

As with any new skill, a firm base of understanding is required before an electrician or technician can become proficient. After completing the text, the reader will possess a good foundation upon which additional PLC skills and understanding can be built.

The best teacher, of course, is experience, and the only way to really fully understand any given PLC is to work with that PLC and its associated software. If a PLC is not available, the next best thing is a workshop or seminar sponsored by a local PLC distributor. If a workshop or seminar is not available, obtain as much literature and other information as possible from a local electrical distributor, PLC representative, or the Internet.

PLC manufacturers and third-party vendors are continually adding new hardware that can communicate across different platforms. The day of proprietary hardware and communications is all but gone. With the rapid advancements in PLCs and software, the technician without an electronics background need not feel intimidated. The manufacturers are doing everything possible to make the PLC easy to install, program, troubleshoot, and maintain.

Technician’s Guide to Programmable Controllers is organized to take the student from “What is a PLC?” to learning the basic hardware, software, communications, and programming languages used with most PLCs on the market today. Each chapter is designed to build on the knowledge gained from previous chapters. Students that complete the text will have a solid foundation and understanding of what PLCs are, how they are used, and the programming language required to make them perform the automation tasks required in today’s industrial world.

INSTRUCTOR RESOURCES

Additional instructor resources for this product are available online. Instructor assets include an Instructor's Manual, PowerPoint® slides, and a test bank powered by Cognero®. Sign up or sign in at **www.cengage.com** to search for and access this product and its online resources.

NEW USERS

If you're new to Cengage.com and do not have a password, contact your sales representative.

About the Authors

Terry R. Borden is owner of Adept Consulting in Ione, Washington, and recently retired from Seattle City Light as Operations Manager of a 1,000-megawatt hydroelectric project in northeast Washington. Mr. Borden is a former member of the Electrical Maintenance and Automation Department at Spokane Community College in Spokane, Washington. He holds a degree in Industrial Automation and Robotics. Mr. Borden has worked as a control engineer and was a partner in Applied Solutions, LLC.

Richard A. Cox is the executive director of COXCO Training and Consulting in Spokane, Washington, and is a retired member of the Electrical/Robotics Department at Spokane Community College. He holds a Bachelor of Science degree from the University of New York, a Master of Science degree from Eastern Washington University, and is also a retired member of the International Brotherhood of Electrical Workers, Local 73.

The authors wish to acknowledge the cooperation and assistance of Rockwell Automation, Inc. whose product information, photographs, and software are used throughout the text to illustrate PLC concepts and components. Rockwell Automation, Inc. is a world leader in the PLC field, and offers a full line of programmable logic controllers and software for industrial automation. It is not a question of which PLC is best, but rather which PLC best fits your needs and individual applications.

Chapter 1 What Is a Programmable Logic Controller (PLC)?

Learning Objectives

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

- Describe several advantages of a programmable logic controller (PLC) over hardwired relay systems.
- Identify the four major components of a typical PLC.
- Describe the function of the four major components of a typical PLC.
- Define the acronyms PLC, CPU, PC, I/O, ADC, and DAC.
- Define the term *discrete*.
- Define the term *analog*.

Introduction to Programmable Logic Controllers

A programmable logic controller is a solid-state system originally designed to perform the logic functions previously accomplished by components such as electromechanical relays, drum switches, mechanical timers/counters, etc., for the control and operation of manufacturing process equipment and machinery.

The electromechanical relay (control relays, pneumatic timer relays, etc.) had served well for many generations, often under adverse conditions. The ever-increasing sophistication and complexity of modern processing equipment required faster-acting, more reliable control functions than electromechanical relays and/or timing devices could offer. Relays had to be hardwired to perform a specific function, and when the system requirements changed, the relay wiring had to be changed or modified. In extreme cases, such as in the auto industry, complete control panels had to be replaced since it was not economically feasible to rewire the old panels with each model changeover.

It was, in fact, the requirements of the auto industry and other highly specialized, high-speed manufacturing processes that created a demand for smaller, faster-acting, and more reliable control devices. The electrical/electronics industry responded with modular-designed, solid-state electronic devices. These early devices, while offering solid-state reliability, lower power consumption, expandability, and elimination of much of the hardwiring, also brought with them a new language. The language consisted of AND, OR, NOT gates, J-K flip flops, and so on.

What happened to simple relay logic and ladder diagrams? That was the question plant engineers and maintenance electricians/technicians asked the solid-state device manufacturers. The reluctance of the end user to learn a new language and the advent of the microprocessor gave the industry what is now known as the **programmable logic controller (PLC)**. The first PLC was invented in 1969 by Richard (Dick) E. Morley, the founder of the Modicon Corporation.

Internally, there are still AND gates, OR gates, and so forth in the processor, but the design engineers have preprogrammed the PLC so that programs can be entered using Relay Ladder Logic. While Relay Ladder Logic may not have the mystique of other computer languages such as Python, Java, and C++, it is however a high-level, real-world, graphic programming language that is understood by most electricians and technicians. Relay Ladder Logic programming is the most common programming language used today, but other programming languages such as Sequential Function Chart, Structured Text, and Function Block languages can also be found. A brief description of each will be covered in later chapters.

The International Electrotechnical Commission (IEC) IEC61131 defines a programmable controller as:

> A digitally operating electronic system, designed for use in an industrial environment, which uses a programmable memory for internal storage of user-oriented instructions for implementing specific functions such as logic, sequencing, timing, counting and arithmetic, to control, through digital or analogue inputs and outputs, various types of machines or processes. Both the PLC and its associated peripherals are designed so that they can be easily integrated into an industrial control system and easily used in all their intended functions.

What does a PLC consist of, and how is it different from a computer system? The PLC consists of a programming device (computer), processor unit, power supply, memory, and an input/output (**I/O**) interface such as the system illustrated in Figure 1–1. And while there are similarities, there are also some major differences.

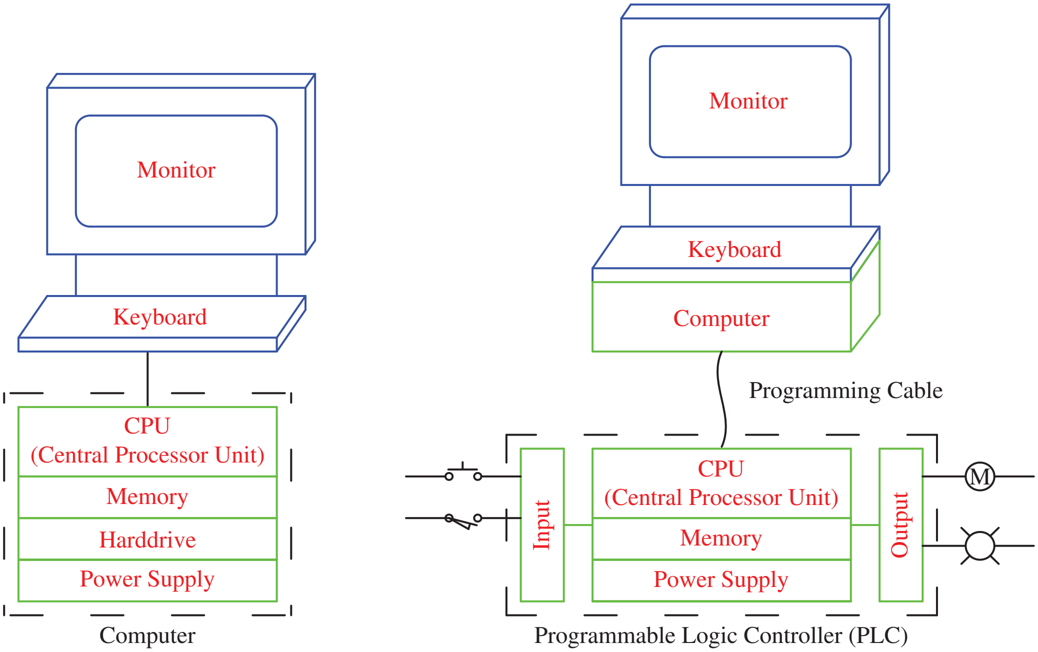


Figure 1–1 Comparison of a Computer System and a PLC

Note: An interface occurs when two systems come together and interact or communicate. In the case of the PLC, the communication or interaction is between the inputs (limit switches, push buttons, sensors, and the like), outputs (coils, solenoids, lights, and so forth), and the processor (CPU). This interface happens when any I/O voltage (AC or DC) or current signal is changed to or from a low-voltage DC signal that the processor uses internally for the decision-making process.

PLCs are designed to be operated and maintained by plant engineers and maintenance personnel with limited knowledge of computers. Like the computer, which has an internal memory for its operation and storage of programs and data, the PLC also has memory for storing the user program, or Logic, as well as memory for storing data. But unlike the computer, the PLC is typically programmed in Relay Ladder Logic, *not* one of the computer languages and is designed for controlling the operation of a process machine or driven equipment.

The PLC is also designed to operate in an industrial environment with wide ranges of ambient temperature, vibration, and humidity, and is not usually affected by the electrical noise that is inherent in most industrial locations.

Note: Electrical noise is discussed in Chapter 3.

Maybe one of the biggest, or at least the most significant difference between the PLC and a computer, is that PLCs have been designed for installation and maintenance by plant electricians who are not required to be highly skilled electronics or computer technicians. Troubleshooting is simplified in most PLCs because they include fault indicators, blown-fuse indicators, I/O status indicators, and written fault information that can be displayed on the PLC and/or programmer.

Although the PLC and the personal computer (PC) are different in many ways, the PC is often used for programming and monitoring the PLC. Using PCs in conjunction with PLCs will be discussed in later chapters.

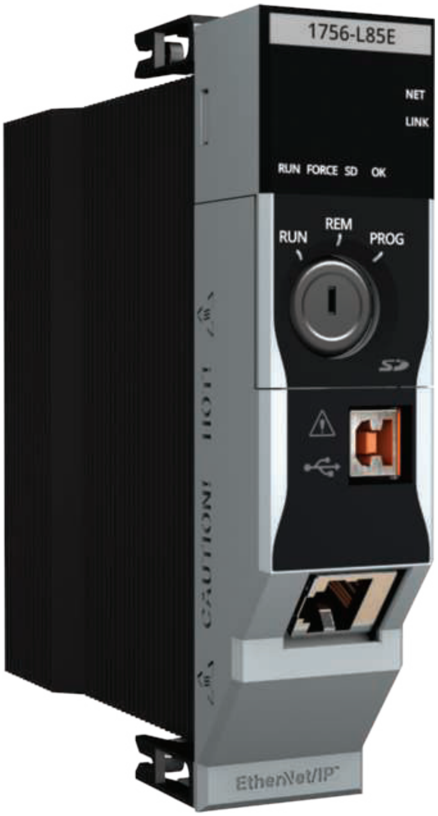
The Main Components of a Programmable Logic Controller System

A typical PLC can be divided into four main components. These components consist of the **processor unit**, **power supply**, input/output section (I/O interface), and the programming device (**programmer**).

The processor unit or central processing unit (**CPU**) houses the processor, which is the decision-maker or “brain” of the system. The brain is a microprocessor-based system that replaces control relays, counters, timers, sequencers, and so forth, and is designed so that the user can enter the desired program in Relay Ladder Logic. The processor then makes all the decisions necessary to carry out the user program, based on the status of the inputs and outputs for control of a machine or process. It can also perform arithmetic functions, data manipulation, and communications between the local I/O section, remotely located I/O sections, and/or other networked PLC systems. Figure 1–2 shows the Allen-Bradley 1756-L85E PLC processor unit.

The power supply is necessary to convert the available power source, 120 or 240 volts AC, 24, 48, or 125 volts DC, to the low-voltage DC required for the logic circuits of the processor, and for the internal power required for the I/O modules. The power supply can be a separate unit as, shown in Figure 1–3, one of modular design that plugs into the processor chassis or rack, or, depending on the manufacturer, one that is an integral part of the processor.

Note: The power supply does not supply power for the actual input or output devices themselves; it only provides the power needed for the internal electronic circuitry of the input and output modules. Power for the input and output devices, if required, must be provided from a separate source.



Courtesy of Rockwell Automation, Inc.

Figure 1–2 Allen-Bradley 1756-L85E PLC Processor Unit

Courtesy of Rockwell Automation, Inc.

Figure 1–3 Allen-Bradley PLC Power Supply

The size or power rating of the power supply is based on the number and type of I/O modules that are to be installed. Power supplies are normally available with output power ratings of 30 to 75 watts.

Note: Consider future needs and the possibility of expansion when initially sizing the power supply. It is cheaper in the long run to install a larger power supply initially than to try to add additional capacity at a later date.

The I/O section consists of I/O modules that are either fixed or modular. The number of inputs and outputs necessary is dictated by the requirements of the equipment that is to be controlled by the PLC. Figure 1–4 shows a chassis or rack with modular I/O modules installed. I/O modules are "plugged in" and added as needed.

I/O modules are where the real-world devices are connected. The real-world **input** (I) **devices** can be push buttons, limit switches, analog sensors, pressure switches, selector switches, etc., while the real-world **output** (O) **devices** can be hardwired motor starter coils, solenoid valves, indicator lights, positioning valves, and the like. The term *real world* is used to distinguish actual devices that exist and must be physically wired to the I/O modules as compared to internal functions of the PLC system that duplicate the function of relays, timers, counters, and so on, even though none physically exists. This may seem a bit strange and hard to understand at this point, but the distinction between what

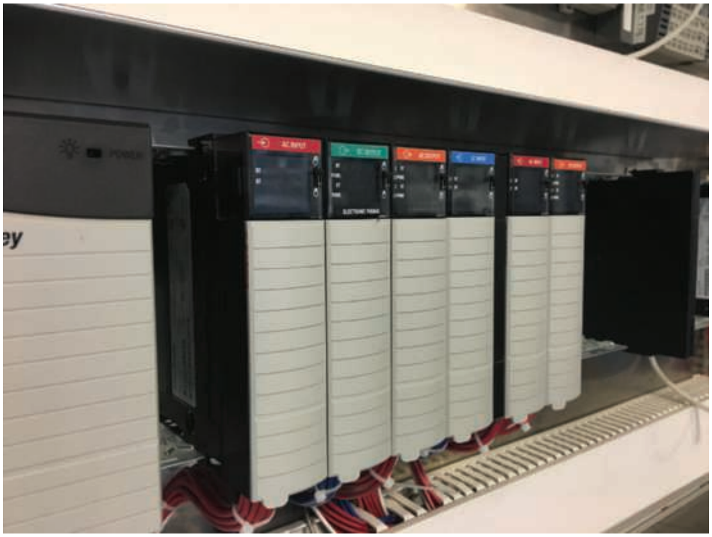


Figure 1–4 PLC Chassis with I/O Modules Installed

the processor can do internally—which eliminates the need for all the previously used control relays, timers, counters, and so forth—will be graphically shown and readily understandable later in the text.

Real-world I/O devices are of two types: discrete and analog. Discrete I/O devices are either *ON* or *OFF*, open or closed, while analog devices have a range of possible values. Examples of discrete devices are limit switches, push buttons, motor starter coils, and indicator lamps. Examples of analog devices are pressure sensors, temperature probes, panel meters, variable speed drive signals, and modulating valves. When reference is made to an I/O device, the terms ***discrete input*** *device*, ***discrete output*** *device*, ***analog input*** *device*, and ***analog output*** *device* are commonly used to describe the type of device.

A reference was made earlier in this chapter to the I/O section as an interface. Although not a common reference, it is an accurate one. The I/O section contains the circuitry necessary to convert input voltages from *discrete input devices* to low-level DC voltages (typically 5V) that the processor uses internally to represent the status or condition (*ON* or *OFF*). Similarly, the I/O section changes low-level DC signals from the processor to AC or DC voltages required to operate the *discrete output devices*. The I/O section also converts varying voltage or current signals from *analog input devices* into corresponding decimal values by way of an Analog-to-Digital converter (ADC). This same process, but reversed via a Digital-to-Analog converter (DAC), is used by the I/O section to convert decimal values into corresponding voltage or current signals necessary to operate *analog output devices*. The field signals from both digital and analog devices are normally isolated from the low-level logic circuitry of the processor by means of optical coupling. This is a brief overview of the I/O section and its function. How I/O devices are wired to I/O modules, optical coupling, and more information about the module circuitry itself is covered in Chapter 3.

The programming device is used to enter the desired program or sequence of operation into the PLC memory. The program is entered using relay ladder logic, or one of the other PLC programming languages, and it is this program that determines the sequence of operation and ultimate control of the process equipment or driven machinery. The programming device is typically a PC.

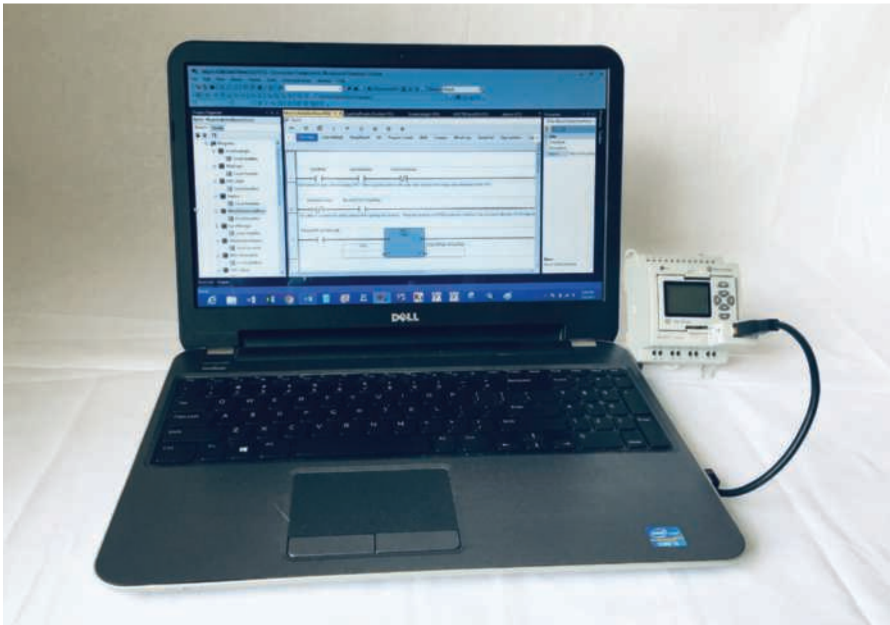


Figure 1–5 Laptop Computer Connected to an Allen-Bradley Micro PLC

A **PC** is used to program nearly all the PLCs on the market today. The PLC programming software that is installed on the PC and a communications cable is sometimes all that is required to program a PLC. At other times special hardware keys and/or communication cards are required to be installed on the PC for it to work successfully as a programming device. The PC provides the benefit of a large viewing screen that allows more of the program to be viewed at one time and makes troubleshooting and memory access much easier. It also provides program storage, as well as runs all the various software packages we have come to depend on today, such as spreadsheets, word processing, etc. Figure 1–5 shows a laptop PC that, with the appropriate software, is used to program and monitor a PLC.

Chapter Summary

Programmable logic controllers (PLCs) have made it possible to precisely control large process machines and driven equipment with less physical wiring and lower installation costs than is required with standard electromechanical relays, pneumatic timers, drum switches, and so on. The programmability allows for fast and easy changes in the Relay Ladder Logic to meet the changing needs of the process or driven equipment without the need for expensive and time-consuming rewiring. Designed to be "technician friendly," the modern PLC is easier to program and can be used by plant engineers and maintenance technicians who have little or no electronic or computer background.

Key Terms

programmable logic controller (PLC)
processor unit
programmer
input device
discrete input
analog input
I/O
power supply
CPU
output device
discrete output
analog output

Review Questions

1. List the four main components of a programmable logic controller.
2. Define the term *interface*.
3. Define the term *real world*.
4. Define the term *discrete*.
5. What do the following acronyms stand for?

   CPU ADC PC
   I/O DAC PLC
6. Define the term *analog*.
7. What is typically used to program and monitor PLCs?
8. Relay Ladder Logic is a high-level graphic computer language.

   T F
9. What is the major advantage of a PLC system over the traditional hardwired control system?

Chapter 2 Numbering Systems

Learning Objectives

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

- Describe decimal, binary, octal, hexadecimal, and binary coded decimal (BCD) numbering systems.
- Convert from one numbering system to another.
- Express negative numbers in 2s complement.
- Add signed numbers.
- Convert a negative binary display to its decimal equivalent.
- Complete a subtraction and addition problem using 2s complement.

Electricians, technicians, or other personnel who are required to program, modify, or maintain a PLC must have a "working" knowledge of the different numbering systems that are used. For example, the input/output addresses may use the octal numbering system; the timer and counter addresses may use the decimal numbering system; accumulated and preset values of the timers and counters may use the binary numbering system; and the hexadecimal system may be used for setting I/O module configurations. The numbering system used in each area discussed varies with the different PLC manufacturers, but it is obvious that to fully understand and program a PLC, an understanding of the various numbering systems is necessary.

Decimal System

Electricians and technicians use the decimal numbering system every day, therefore it is a system they are comfortable with. This system uses 10 unique numbers, or digits, which are 0–9. A numbering system that uses 10 digits is said to have a base of 10. The value of the decimal number depends on the digit(s) used and each digit's place value. Each position can be represented as a power of 10, starting with 10^0, as shown in Figure 2–1. In the decimal system, the first position to the left of the decimal point is called the units place, and any digit from 0–9 can be used. The next position to the left of the units place is the tens place; next is the hundreds place, the

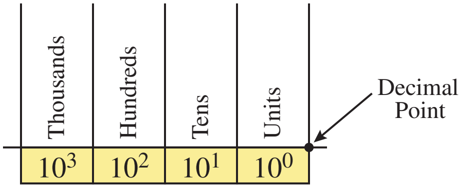


Figure 2–1 Place Value and Corresponding Power of Ten

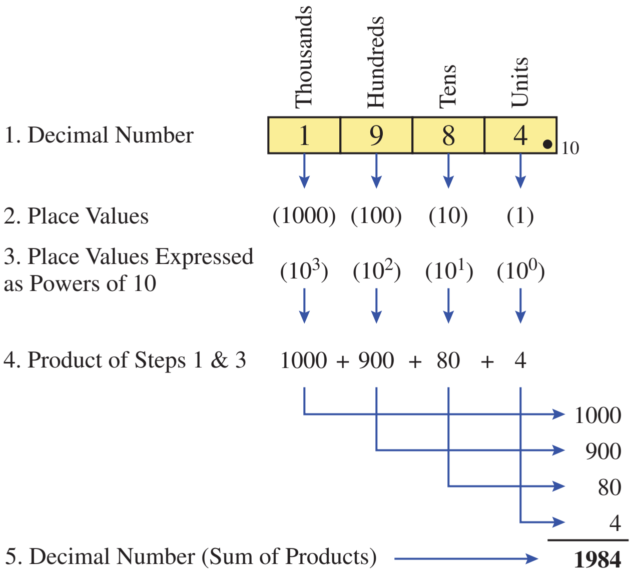


Figure 2–2 Decimal Numbering System

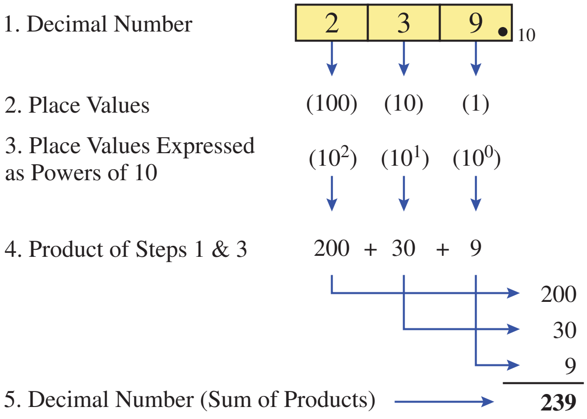

1. Decimal Number
2 3 9 • 10
2. Place Values
(100) (10) (1)
3. Place Values Expressed as Powers of 10
(10^2) (10^1) (10^0)
4. Product of Steps 1 & 3
200 + 30 + 9
200
30
9
5. Decimal Number (Sum of Products)
239

Figure 2–3 Decimal Numbering System

thousands place, and so on, with each place extending the capability of the decimal system by 10, or a power of 10.

Note: Any number that uses an exponent of 0, such as 10^0, has a place value of 1. Exponent 10^0 equals 1.

A specific decimal number can be expressed by adding the place values, as shown in Figure 2–2.

Mathematically, each place value is expressed as a digit number times a power of the base, or 10, in the decimal numbering system. Another example is shown in Figure 2–3 using the decimal number 239.

Binary System

The **binary** system uses only two digits: 1 and 0. Since only two digits are used, this system has a base of 2. Like the decimal system—and all numbering systems for that matter—each digit has a certain place value. The first place to the left of the starting point, or binary point, is the units or 1s location (base 2^0). The next place, to the left of the units place, is the 2s place, or base 2^1, as shown in Figure 2–4. The next place value is the 4s place, or base 2^2, then the 8s place, or base 2^3, and so forth. A binary number is always indicated by placing a 2 in subscript to the right of the unit's digit. Figure 2–4 illustrates how a binary number is converted to a decimal equivalent number. Note the subscripted 2 at the lower right-hand corner of the binary number line that indicates a base 2, or binary number.

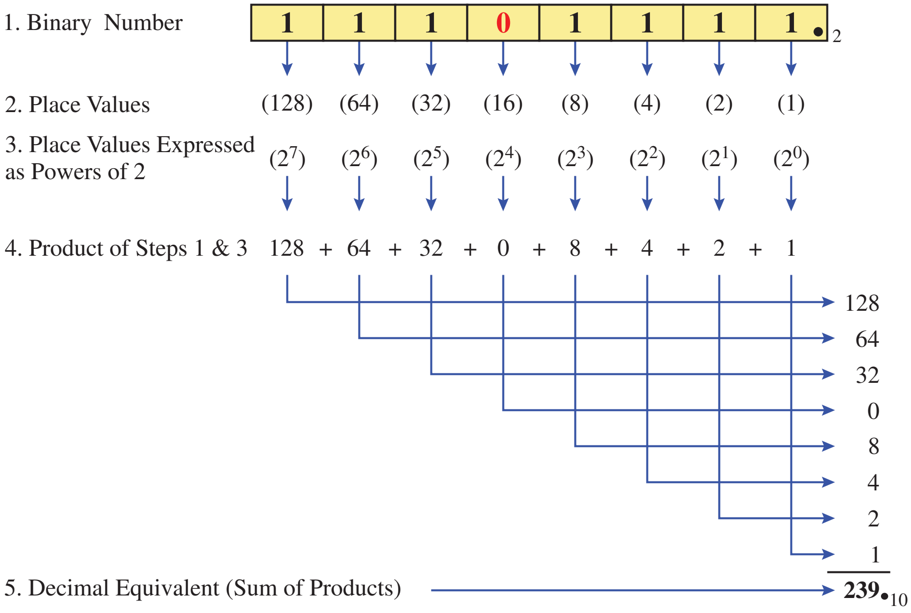


Figure 2–4 Converting a Binary Number to a Decimal Number

To convert a decimal number into a binary number, or to any numbering system for that matter, use the following procedure, as shown in Figure 2–5. Divide the decimal number by the base you wish to convert to, in this case 2. The remainder is the 1s value (see Step 1). Now divide the quotient from the first division again; the remainder becomes the value that is placed in the 2s location (see Step 2). The quotient of each preceding division is then divided by the base 2 until the base can no longer be divided (see Step 8), and the remainder (1) becomes the last digit in the binary number.

It is important to arrange the remainders correctly when making the decimal-to-binary conversion. The first digit placed in the 1s position is called the *least* significant digit, whereas the last digit is called the *most* significant digit. The last digit placed has the highest place value (128s) which is why it is called the most significant digit. This reference to least and most significant digits is common, and refers to the relative position of any given digit within a number.

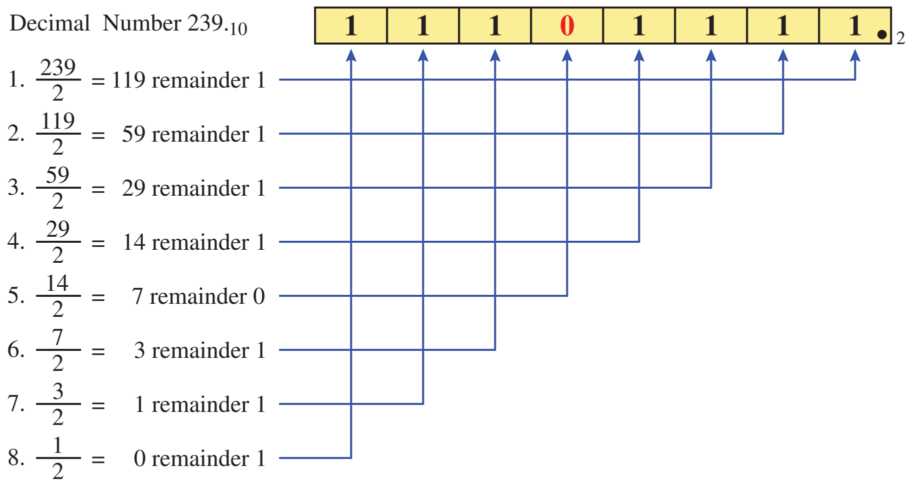


Figure 2–5 Converting a Decimal Number to a Binary Number

The following steps summarize this decimal-to-binary conversion.

Step 1. The decimal number is divided by 2 (base of the binary numbering system). The quotient is listed (119) as well as the remainder (1).

Step 2. Divide the quotient of Step 1 (119) by base 2, and list the new quotient (59) and the remainder (1).

Step 3. Divide the quotient of Step 2 (59) by base 2, and list the new quotient (29) and remainder (1).

Step 4. Divide the quotient of Step 3 (29) by 2, and list the new quotient (14) and the remainder (1).

Step 5. Divide the quotient of Step 4 (14) by 2, and list the new quotient (7) and remainder (0).

Step 6. Divide the quotient of Step 5 (7) by 2, and list the new quotient (3) and remainder (1).

Step 7. Divide the quotient of Step 6 (3) by 2, and list the new quotient (1) and remainder (1).

Step 8. Divide the quotient of Step 7 (1) by 2, and list the new quotient (0) and remainder (1).

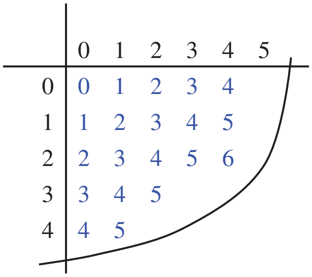
Note: When using a calculator to do the division, the value to the right of the decimal must be multiplied by the base to get the actual remainder. For example, when 239 is divided by 2 (Step 1) on a calculator, the answer is 119.5. To find the actual remainder, the 0.5 is multiplied by 2, the base, to find the remainder 1. This procedure is true for any numbering system. The base times the value to the right of the decimal point equals the actual remainder.

The binary numbering system is used to store information in the processor memory in the form of **bits**.

2s Complement

Virtually all programmable controllers, computers, and other electronic calculating equipment perform counting functions using the binary system. For those PLCs that are programmed to perform arithmetic functions, a method of representing both positive (+) and negative (–) numbers must be used. The most common method is **2s complement**. The 2s complement is simply a convention for binary representation of negative decimal numbers.

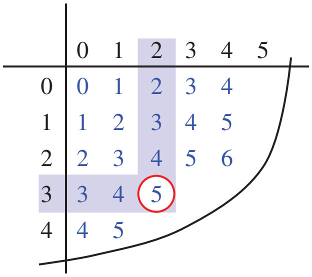
Before going any further with a discussion of 2s complement, a review of adding binary numbers may be helpful. In decimal addition, numbers are added according to an addition table. A partial addition table is shown in Figure 2–6.



	0	1	2	3	4	5
0	0	1	2	3	4	
1	1	2	3	4	5	
2	2	3	4	5	6	
3	3	4	5			
4	4	5				

Figure 2–6 Decimal Addition System

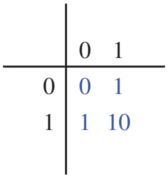
To use the table, the first number to be added is located on the vertical line, and the second number on the horizontal line. The sum, or total, is found where the two imaginary lines intersect. For example, 3 + 2 = 5 (as shown in Figure 2–7).



	0	1	2	3	4	5
0	0	1	2	3	4	
1	1	2	3	4	5	
2	2	3	4	5	6	
3	3	4	5			
4	4	5				

Figure 2–7 Adding 2 and 3

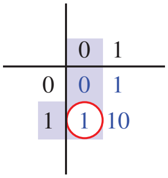
For binary addition, a similar addition table is constructed. The table is small because the binary system only has two digits (1 and 0) (Figure 2–8).



	0	1
0	0	1
1	1	10

Figure 2–8 Binary Addition Table

To use the table, the first number (digit) to be added is located on the vertical line, the second digit is located on the horizontal line. The sum, or total, is found where the two imaginary lines intersect. Figure 2–9 shows an example of adding 1 + 0 = 1.



	0	1
0	0	1
1	1	10

Figure 2–9 Adding Binary 1 and 0

Notice that if 1 and 1 are added, the table shows 1 0, not 2, as might be expected. 1 0 is the binary representation of 2 (Figure 2–10).

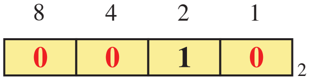


Figure 2–10 Binary Representation of 2

Figure 2–11 shows how binary numbers 1011_2 and 110_2 are added.

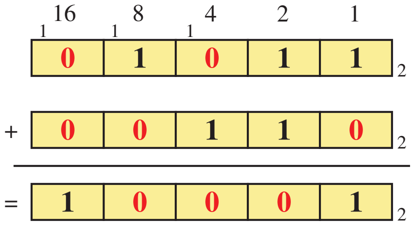


Figure 2–11 Adding Binary Numbers

In the 1s column 1 + 0 = 1.
In the 2s column 1 + 1 = 0, with a carryover of 1.
In the 4s column 1 + 0 + 1 = 0, with a carryover of 1.
In the 8s column 1 + 1 + 0 = 0, with a carryover of 1.
In the 16s column 1 + 0 + 0 = 1.
The sum (total) of 1011_2 and 110_2 is, therefore, 10001_2.

To verify our results we can convert the binary numbers to decimal equivalent numbers and add them, as shown in Figure 2–12.

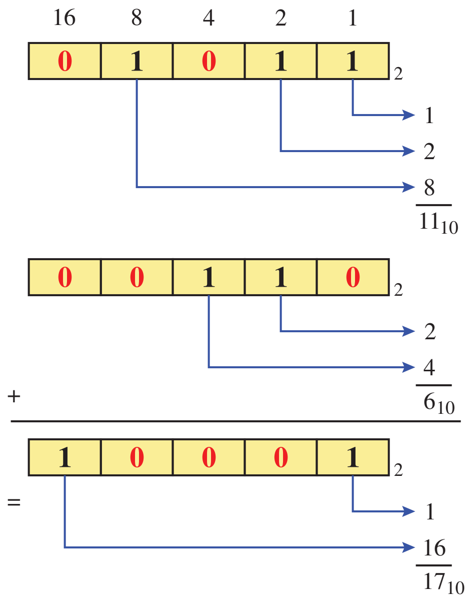


Figure 2–12 Converting Binary Numbers to Decimal Equivalents

Another example of adding binary numbers is shown in Figure 2–13, where 11011_2 and 11_2 are added.

$$\begin{array}{rccccc} & & & 1 & 1 & \\ & 1 & 1 & 0 & 1 & 1 \\ + & & & & 1 & 1 \\ \hline & 1 & 1 & 1 & 1 & 0_2 \end{array}$$

Figure 2–13 Addition of Binary Numbers

In the 1s column 1 + 1 = 0, with a carryover of 1.
In the 2s column 1 + 1 + 1 = 1, with a carryover of 1.

Note: 1 + 1 + 1 = 3. The binary equivalent of 3_{10} is 11_2.

In the 4s column 1 + 0 = 1.
In the 8s column 1 + 0 = 1.
In the 16s column 1 + 0 = 1.
The sum of 11011_2 and 11_2 is 11110_2.

To verify this method, convert the binary numbers to decimal numbers, and add them, as shown in Figure 2–14.

$$\begin{array}{rccccccl} & 1 & 1 & 0 & 1 & 1 & & = 27_{10} \\ + & & & & 1 & 1 & & = \underline{\ 3_{10}} \\ \hline & 1 & 1 & 1 & 1 & 0_2 & & = 30_{10} \end{array}$$

Figure 2–14 Comparing Binary and Decimal Addition

To represent negative numbers using the binary numbering system, one bit is designated as a signed bit. If the designated bit is a 0 (zero), the number is positive, and if the bit is a 1, the number is negative.

Using a 4-bit word length, and using bit 4 as the designated signed bit, 0001_2 represents +1 decimal (see Figure 2–15).

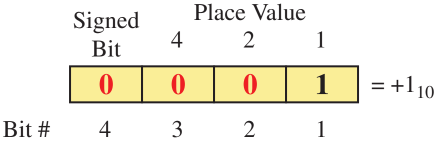


Figure 2–15 4-Bit Word with a Signed Bit

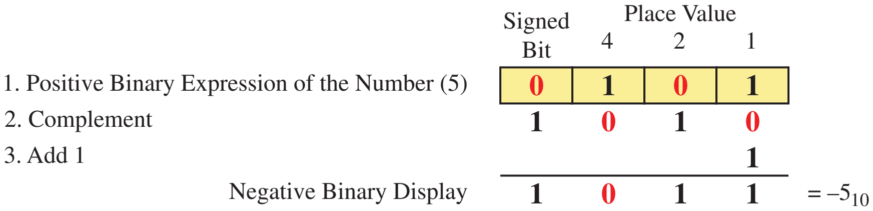
The table in Figure 2–16 shows all of the possible numbers for a 4-bit word using 2s complement.

Binary Number	Decimal
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	0
1111	–1
1110	–2
1101	–3
1100	–4
1011	–5
1010	–6
1001	–7
1000	–8

Figure 2–16 2s Complement Numbers for a 4-Bit Word

Notice that the negative numbers go to –8 while the positive numbers only go to +7. In this case, the signed bit is used for its place value, which is 8. The same holds true for 8-, 16-, and 32-bit words. The maximum negative number is always one number *higher* than the maximum positive number.

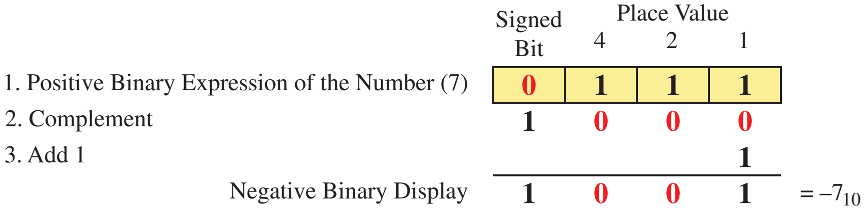
To display a negative binary number requires that the same value positive number be complemented (all 1s changed to 0s and all 0s changed to 1s) and a value of 1 added. The result is the 2s complement of the number. Figure 2–17 shows the steps to express –5 in 2s complement using a 4-bit word.



	Signed	Place Value			
	Bit	4	2	1	
1. Positive Binary Expression of the Number (5)	0	1	0	1	
2. Complement	1	0	1	0	
3. Add 1				1	
Negative Binary Display	1	0	1	1	$= -5_{10}$

Figure 2–17 Expressing –5 in 2s Complement

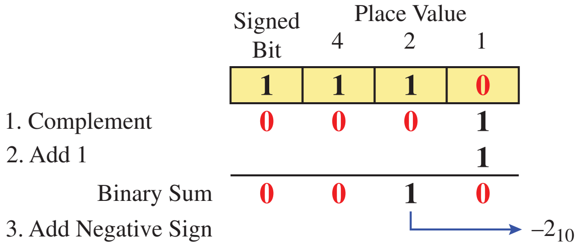
Another example of 2s complement is shown in Figure 2–18 with the steps required to express –7 in 2s complement.



	Signed	Place Value			
	Bit	4	2	1	
1. Positive Binary Expression of the Number (7)	0	1	1	1	
2. Complement	1	0	0	0	
3. Add 1				1	
Negative Binary Display	1	0	0	1	$= -7_{10}$

Figure 2–18 Expressing –7 in 2s Complement

To convert a negative binary number to the decimal equivalent, the negative binary display is complemented, 1 is added, the binary sum is converted to decimal, and the negative sign (–) is added. Figure 2–19 shows what steps are necessary to determine the negative value of 1110_2.



	Signed	Place Value			
	Bit	4	2	1	
	1	1	1	0	
1. Complement	0	0	0	1	
2. Add 1				1	
Binary Sum	0	0	1	0	
3. Add Negative Sign					-2_{10}

Figure 2–19 2s Complement to Decimal Equivalent

An easy way to convert a negative binary number to its equivalent negative decimal number is to subtract the place value of the signed bit from the value of the binary digits. In Figure 2–19, the binary sum 1110 is equal to -2_{10}, which is the decimal number –2. In this example, a 4-bit word is used with bit 4 being the signed bit. The fourth bit normally would have a place value of 8. In this example, the value 8 is subtracted from the value of the binary number 110_2 which is equal to 6, 6 – 8 = –2.

Another example using this method of converting negative binary numbers to their decimal equivalent is shown in Figure 2–20, using an 8-bit word with bit 8 being the signed bit.

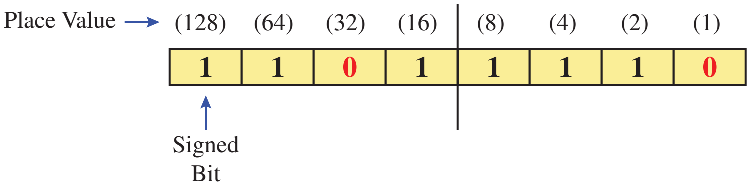


Figure 2–20 8-Bit Word 2s Complement

In this example, the signed bit would have a value of 128 (2^7), whereas the numeric value of the other bits would be 94, as shown in Figure 2–21.

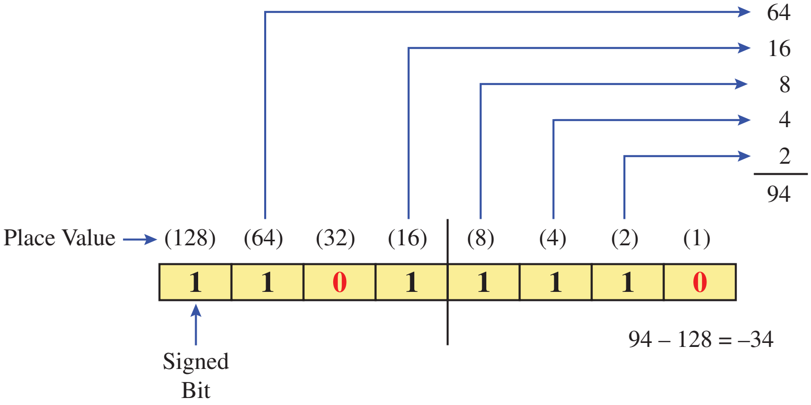


Figure 2–21 8-Bit Word (2s Complement) with Bits Added

Subtracting the place value of the signed bit (128) from the numeric value of the other bits (94) gives us the decimal number: 94 – 128 = –34. To verify this answer, complement the original binary number 1101 1110 to get 0010 0001. Then add 1.

$$\begin{array}{r} 0010\ 0001 \\ 1 \\ \hline 0010\ 0010 \end{array}$$

The answer is $2^5 + 2^1$ or 32 + 2 = 34; adding the negative sign gives us a final answer of –34, the same answer we got when we subtracted the place value of the signed bit (128) from the numeric value of the binary number 94.

If the PLC system uses 2s complement for arithmetic, the highest positive number that a 16-bit word can represent is +32,767, as shown in Figure 2–22. The highest positive number that a 32-bit word can represent is +2,147,483,647.

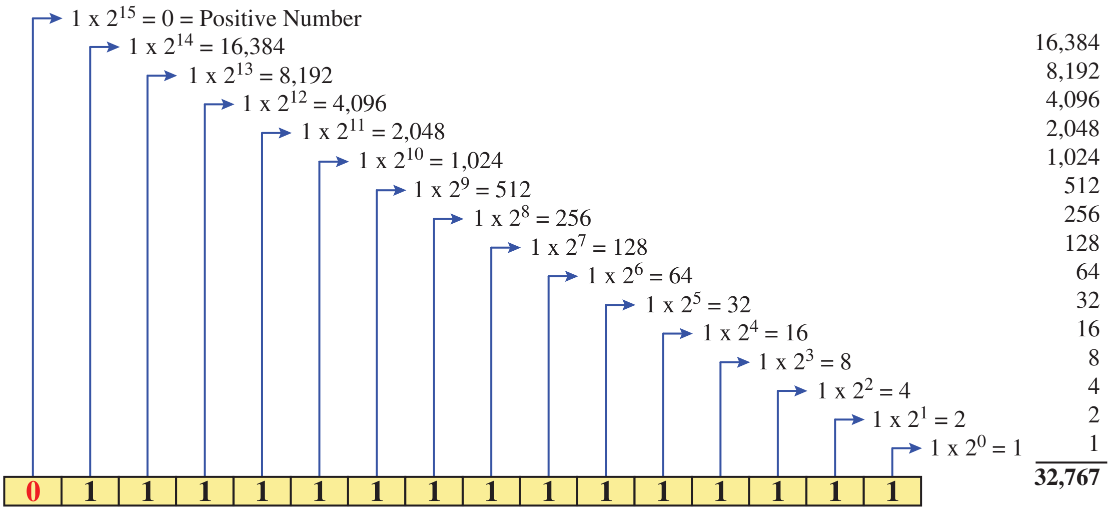
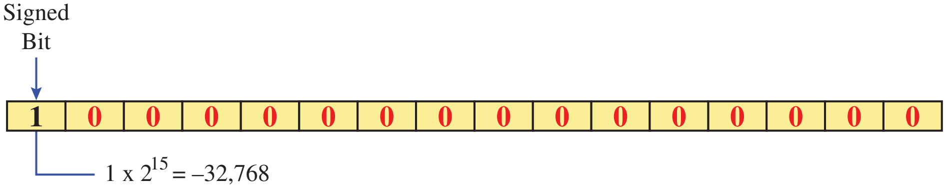


Figure 2–22 Maximum Positive Value of 2s Complement 16-Bit Word

The largest negative number that can be represented by a 16-bit word is –32,768, as shown in Figure 2–23. The largest negative number that can be represented by a 32-bit word is –2,147,483,648.



Rule: Subtract Signed Bit Place Value from the Numeric Value of the Other Bits.

Signed Bit Place Value = 32,768
0 – 32,768 = –32,768

Figure 2–23 Maximum Negative Value of 2s Complement 16-Bit Word

Another method of converting positive numbers to 2s complemented negative numbers is as follows: starting at the least significant bit and working to the left, copy each bit up to and including the first 1 bit, and then complement or change each remaining bit. Figure 2–24 shows this alternate method of expressing –2 in 2s complement using a 4-bit word.

1. Original Positive Number (+2)	0	0	1	0
2. Copy Up to First (1) Bit			1	0
3. Complement the Remaining Bits	1	1	1	0
2s Complement –2	1	1	1	0

Figure 2–24 Alternate Method of 2s Complement

A further example is shown in Figure 2–25 for 2s complementing the value 24 using an 8-bit word.

1. Original Positive Number (+24)	0	0	0	1	1	0	0	0
2. Copy Up to First (1) Bit					1	0	0	0
3. Complement the Remaining Bits	1	1	1	0	1	0	0	0
2s Complement –24	1	1	1	0	1	0	0	0

Figure 2–25 2s Complement of –24 Decimal

By using 2s complement, negative and positive values can now be added. The two steps for adding -7_{10} and $+5_{10}$ using 2s complement with a 4-bit word are shown in Figure 2–26.

		Signed	Place Value			
1. Express –7 In 2s Complement		Bit	4	2	1	
A. Positive Expression of Number (+7)		0	1	1	1	
B. Complement		1	0	0	0	
C. Add 1					1	
Negative Binary Display		1	0	0	1	
2. Add –7 And +5		1	0	0	1	
	+	0	1	0	1	
Binary Sum		1	1	1	0	$= -2_{10}$

Figure 2–26 Adding Positive and Negative Numbers

Note: When adding signed binary numbers, any carryover from the signed bit column is discarded.

Once addition of signed numbers is possible, the other arithmetic functions (subtraction, multiplication, and division) are also possible, because they are achieved by successive addition on a PLC.

Example: Subtracting the number 20 from 26 is accomplished by complementing 20 to obtain –20, and then performing addition.

Subtracting 20 from 26 by complementing 20 and performing addition using an 8-bit word is shown in Figure 2–27.

	Signed			Place Value				
	Bit	64	32	16	8	4	2	1
+26	0	0	0	1	1	0	1	0
+								
–20	1	1	1	0	1	1	0	0
$+6_{10}$	0	0	0	0	0	1	1	0

Figure 2–27 Subtraction by Addition

Octal System

The **octal** system, or base 8, is made up of 8 digits, numbers 0–7. The first digit to the *left* of the octal point is the units place, or 1s, and has a base or power of 8^0. The next place is eights (8s) or base 8^1. The next place is sixty-fours (64s) or base 8^2, followed by five hundred twelves (512s) or base 8^3, and four thousand ninety-sixes or base 8^4, and so on. An octal number will always be expressed by placing an eight in subscript to the right of the unit's digit, as shown in Figure 2–28.

$$357._8$$

Figure 2–28 Octal Number

The method of converting an octal number to a decimal equivalent number is illustrated in Figure 2–29.

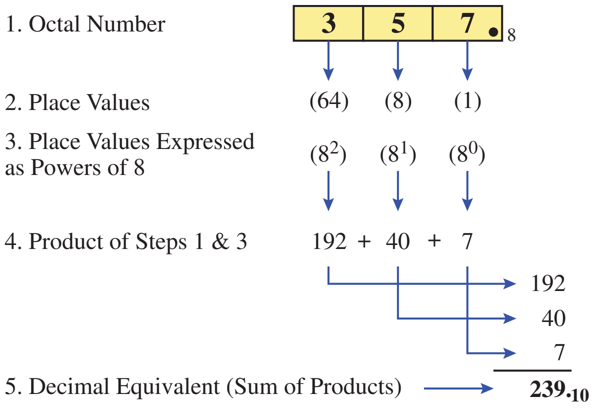

1. Octal Number
3 5 7 •8
2. Place Values
(64) (8) (1)
3. Place Values Expressed as Powers of 8
(8²) (8¹) (8⁰)
4. Product of Steps 1 & 3
192 + 40 + 7
192
40
7
5. Decimal Equivalent (Sum of Products)
239.10

Figure 2–29 Converting an Octal Number to a Decimal Number

The decimal number 239 is converted to an octal number in Figure 2–30.

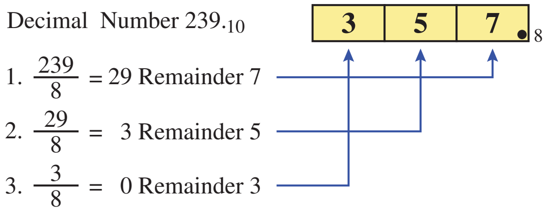


Figure 2–30 Converting a Decimal Number to an Octal Number

Step 1. The decimal number 239 is divided by 8 (base for the octal numbering system). The quotient is listed (29) as well as the remainder (7). A calculator shows the answer as 29.875. The quotient is 29, and the remainder is 0.875 × 8, or 7.

Step 2. Divide the quotient of Step 1 (29) by 8, and list the new quotient (3) and the remainder (5). A calculator gives the answer 3.625. The quotient is 3, and the remainder is 0.625 × 8, or 5.

Step 3. Divide the quotient from Step 2 (3) by 8, and list the new quotient (0) and remainder (3). The quotient 3 divided by 8 equals 0.375. The new quotient is 0, and the remainder is 0.375 × 8, or 3.

The decimal number 239 is the same as the octal number 357.

Since the largest single number that can be expressed using the octal numbering system is seven (7), each octal digit can be represented by using only three (3) binary bits (base 2). Figure 2–31 illustrates how to convert an octal number to a binary number. The figure shows three sets of binary bits and the place value of each bit. For the *least* significant digit (7), a one (1) must be placed in the 1s place, the 2s place, and the 4s place to equal 7. For the middle digit (5), a 1 is placed in the 1s place and the 4s place, while a 0 is placed in the 2s place. This combination equals 5. For the *most* significant digit (3), a 1 is placed in the 1s place and the 2s place, and a 0 is placed in the 4s place. This combination adds up to 3.

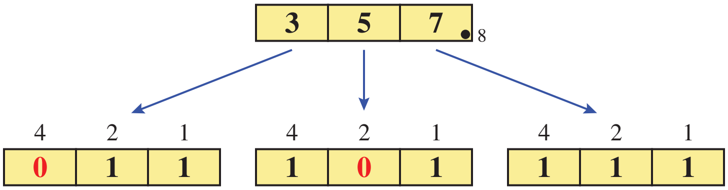
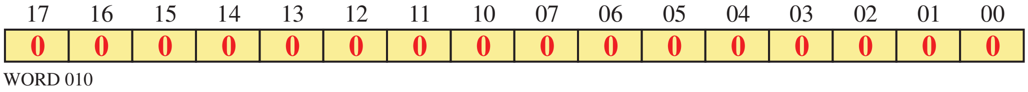


Figure 2–31 Conversion of Octal Number to Binary

Some older PLCs used the octal numbering system for I/O addressing. The terminals of the input and output modules would be labeled 00 through 07 and 10 through 17, rather than 0 through 15 as would be the case with decimal numbering (which is used in many PLCs today). When using the octal numbering system, words are labeled 000–007, 010–017, 020–027, and so forth, whereas the bits are labeled 00–07 and 10–17. Figure 2–32 shows a memory word with the internal bits addressed using the octal numbering system.



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

WORD 010

Figure 2–32 Word and Bit Labeling Using the Octal Numbering System

Hexadecimal System

The **hexadecimal** system, often referred to as HEX, consists of a number system with base 16.

It seems logical that the numbers used in base 16 would be 0–15. However, only numbers 0–9 are used, and the letters A–F represent numbers 10–15, respectively. The place values from the hexadecimal point are 1s—16^0, 16s—16^1, 256s—16^2, 4096s—16^3, and so on.

Each hexadecimal digit is represented by four binary digits. The binary equivalents are shown in the table in Figure 2–33.

Hexadecimal	Binary Number	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Figure 2–33 Hexadecimal Equivalents for Binary and Decimal

The decimal number 4,780 is converted to hexadecimal, as illustrated in Figure 2–34.

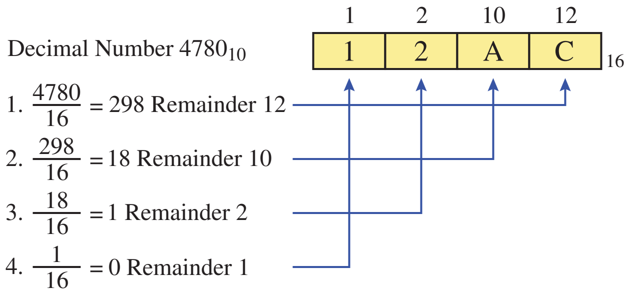

Decimal Number 4780_{10}
1 2 10 12
1 2 A C
16
1. $\frac{4780}{16}$ = 298 Remainder 12
2. $\frac{298}{16}$ = 18 Remainder 10
3. $\frac{18}{16}$ = 1 Remainder 2
4. $\frac{1}{16}$ = 0 Remainder 1

Figure 2–34 Converting a Decimal Number to a Hexadecimal Number

Step 1. The decimal number is divided by 16 (base for the hexadecimal numbering system). The quotient is listed (298) as well as the remainder (12). A calculator provides the answer 298.75. The quotient is 298, and the remainder is 0.75 × 16, or 12.

Step 2. Divide the quotient of Step 1 (298) by 16, and list the new quotient (18) and the remainder (10). The answer is 18.625. The quotient is 18, and the remainder is 0.625 × 16, or 10.

Step 3. Divide the quotient from Step 2 (18) by 16, and list the new quotient (1) and the remainder (2). Eighteen divided by 16 equals 1.125. The quotient is 1, and the remainder is 0.125 × 16, or 2.

Step 4. Divide the quotient from Step 3 (1) by 16, and list the new quotient (0) and the remainder (1). One divided by 16 equals 0.0625. The quotient is 0, and the remainder is 0.0625 × 16, or 1.

Converting a hexadecimal number to a decimal number is illustrated in Figure 2–35.

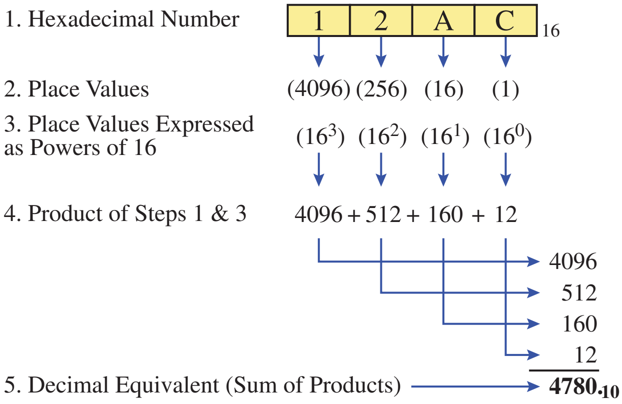


Figure 2–35 Converting a Hexadecimal Number to a Decimal Number

Note: Remember that A is equivalent to 10, and C is equivalent to 12.

The binary equivalent of the hexadecimal number 12AC is shown in Figure 2–36.

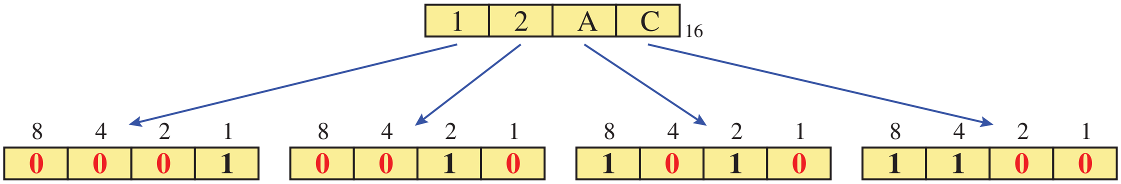


Figure 2–36 Binary Equivalent of a Hexadecimal Number

Since the largest number that can be displayed using the hexadecimal numbering system is 15, or F (as shown in the table in Figure 2–33), only four binary bits are needed to display each hexadecimal digit. The conversion to binary simply places the 1s in the correct binary locations to duplicate the hexadecimal digit (1 through F), as illustrated. The C has a value of 12, so 1s are placed in the 8s and 4s locations, while zeros (0) are placed in the 2s and 1s locations for a total binary value of 12. The same procedure is followed for the remaining digits A (10), 2, and 1.

The HEX system is used when large numbers need to be processed. The hexadecimal system is also used by some PLCs for entering output instructions into a sequencer.

Figure 2–37 shows the conversion of a 16-bit binary number to its hexadecimal equivalent.

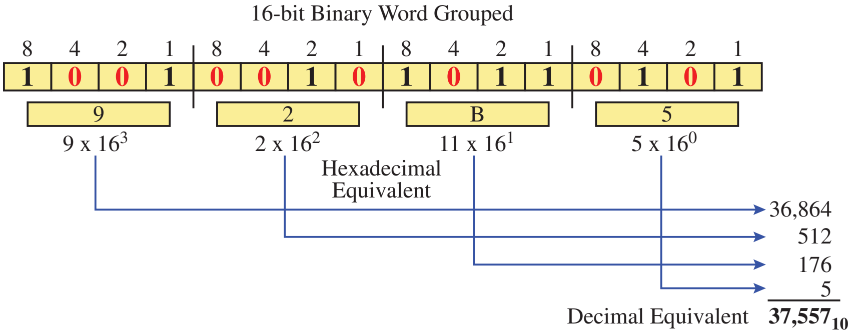


Figure 2–37 16-Bit Binary to Hexadecimal

The first step in converting 16-bit binary to hexadecimal is to group the 16-bit binary word into groups of four (conversion to BCD). Each group of four digits is converted to its hexadecimal equivalent. In Figure 2–37, the hexadecimal number is $92B5_{16}$.

The conversion of $92B5_{16}$ to decimal is $37{,}557_{10}$. Figure 2–38 shows the conversion of the original 16-bit binary number to its decimal equivalent.

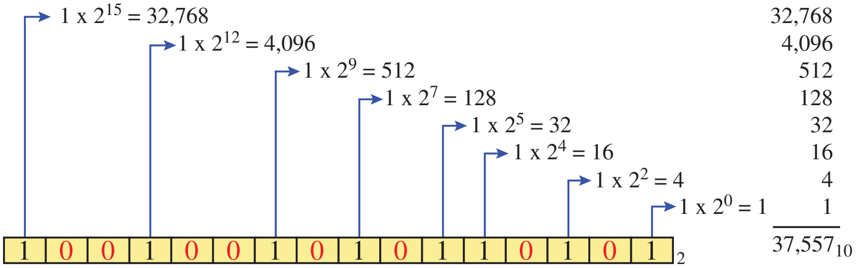


Figure 2–38 Converting a 16-Bit Digital Number to a Decimal Number

BCD System

When large decimal numbers are to be converted to binary for memory storage, the process becomes somewhat cumbersome. To solve this problem and speed conversion, the Binary Coded Decimal (**BCD**) system was devised. In the BCD system, four binary digits (base 2) are used to represent each decimal digit. To distinguish the BCD numbering system from a binary system, the designation BCD is subscripted and placed to the lower right of the units place. Converting a BCD number to a decimal equivalent is shown in Figure 2–39.

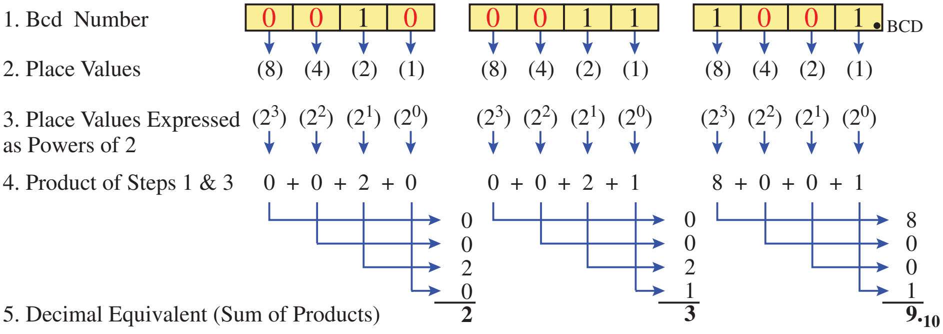
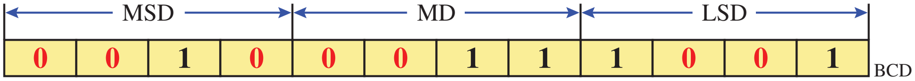


Figure 2–39 Converting a BCD Number to a Decimal Number

When using a BCD numbering system, three decimal numbers may be displayed using 12 bits (3 groups of 4), or 16 bits (4 groups of 4) may be used to represent four decimal numbers or digits.

When only three decimal digits are to be represented, using 12 bits, they are further identified as ***most significant digit*** (MSD), ***middle digit*** (MD), and ***least significant digit*** (LSD) (Figure 2–40).



MSD				MD				LSD				
0	0	1	0	0	0	1	1	1	0	0	1	BCD

Figure 2–40 Significant Digits

When using the BCD system, the largest decimal number that can be displayed by any four binary digits is 9. The table in Figure 2–41 shows the four binary digit equivalents for each decimal number 0–9.

Place Value				Decimal Equivalent
2^3 (8)	2^2 (4)	2^1 (2)	2^0 (1)	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Figure 2–41 Binary to Decimal Equivalents

Using Numbering Systems

The alphanumeric keys of many programming terminals generate standard ASCII characters and control codes. **ASCII** is an acronym for American Standard Code for Information Interchange. The ASCII code uses different combinations of 7-bit binary (base 2) information for communication of data. The data may be communicated to a printer, barcode reader, or be shown on the display of the programmer and/or computer.

Note: ASCII information is often expressed in hexadecimal (base 16). Figure 2–42 shows the 128 standard ASCII control codes and character set with both the binary and hexadecimal numbering systems.

			Most Significant Bit (**MSB**)							
	BINARY		000	001	010	011	100	101	110	111
		HEX	0	1	2	3	4	5	6	7
Least Significant Bit (**LSB**)	0000	0	NUL	DLE	SP	0	@	P	\	p
	0001	1	SOH	DC1	!	1	A	Q	a	q
	0010	2	STX	DC2	"	2	B	R	b	r
	0011	3	ETX	DC3	#	3	C	S	c	s
	0100	4	EOT	DC4	$	4	D	T	d	t
	0101	5	ENQ	NAK	%	5	E	U	e	u
	0110	6	ACK	SYN	&	6	F	V	f	v
	0111	7	BEL	ETB	'	7	G	W	g	w
	1000	8	BS	CAN	(	8	H	X	h	x
	1001	9	HT	EM	)	9	I	Y	i	y
	1010	A	LF	SUB	*	:	J	Z	j	z
	1011	B	VT	ESC	+	;	K	[	k	{
	1100	C	FF	FS	,	<	L	\	l	|
	1101	D	CR	GS	-	=	M	]	m	}
	1110	E	SO	RS	.	>	N	^	n	~
	1111	F	SI	US	/	?	O	–	o	DEL

Figure 2–42 Standard ASCII Control Code and Character Set

The digital or hexadecimal number is determined by first locating the vertical column where the code or character is located, and then the horizontal row.

Example: The letter A is in column 4, row 1. The binary number that transmits the letter A is 100 0001. The hexadecimal number is 41. The symbol # is 010 0011 in binary and 23 in HEX.

An eighth bit is often used by programmers to provide error-checking of information that is transmitted. This eighth bit is called the **parity bit**.

For *even* parity, the parity bit (the eighth bit) is added to the seven bits that represent the ASCII codes and characters so that the number of 1s will always add up to an even number.

Example: The binary number for the # symbol is 010 0011. The 1s add up to three, an odd number. By adding an eighth bit and making it a 1, the total of 1s is now 4, or even, as shown in Figure 2–43.

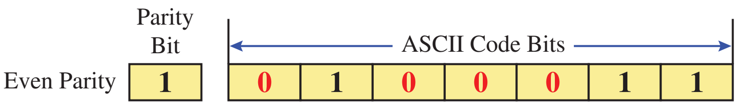


Figure 2–43 Parity Bit Set to 1 for Even Parity

The letter A, which is the binary number 100 0001, has two 1s and is already even. In this case, the parity bit would be a 0, as shown in Figure 2–44.

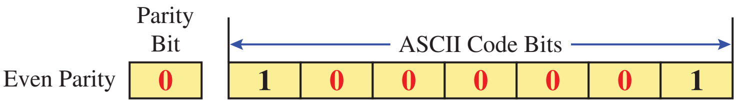


Figure 2–44 Parity Bit Set to 0 for Even Parity

The ASCII control code BS (backspace) is binary number 000 1000. For even parity, a 1 is added for the parity bit, as shown in Figure 2–45.

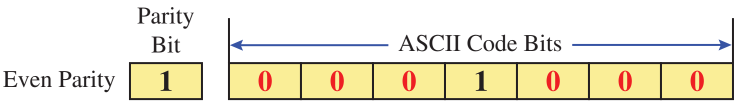


Figure 2–45 Even Parity

By checking each character or control code that is sent for an even number of 1s, transmission errors can be detected when an odd number of 1s is found.

For systems that operate on *odd* parity, the parity bit is used to make the total of 1s add up to an odd number.

Example: The number 5 has a binary number of 011 0101. The 1s add up to 4. The parity bit is set to 1, making the 1s total 5, or an odd number. Figure 2–46 illustrates this concept.

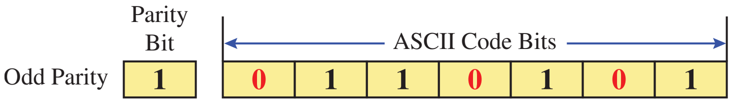


Figure 2–46 Parity Bit Set to 1 for Odd Parity

For systems that do not use a parity bit for error-checking, the 8 bit is always a zero (0).

Chapter Summary

There are several numbering systems that are used to store information in the form of binary digits (bits) into the memory system of a processor. The specific numbering system or the combination of numbering systems used depends on the hardware requirements of the specific PLC manufacturer. The important thing to remember, however, is that no matter which numbering system or systems are used, the information is still stored as 1s and 0s.

For programmable controllers to perform arithmetic functions, a way must be found to represent both positive and negative numbers. One of the most common methods used is called 2s complement. Using 2s complement, negative and positive numbers can be added, subtracted, divided, and multiplied. In reality, however, all arithmetic functions are accomplished by successive addition.

Key Terms

binary
octal
hexadecimal
most significant digit
least significant digit
parity bit
2s complement
BCD
bits
ASCII
middle digit

Review Questions

1. When information is stored using only 1s and 0s, it is called a ________________ system.
2. A *bit* is an acronym for ____.
3. The decimal numbering system uses 10 digits, or a base of 10. List the base for each of the following numbering systems.
   a. binary base ____________________
   b. hexadecimal base ____________________
   c. octal base ____________________
4. Convert *binary* number 11011011 to a *decimal* number.
5. Convert *decimal* number 359 to a *binary* number.
6. Convert *hexadecimal* number 14CD to a *decimal* number.
7. Convert *decimal* number 3247 to a *hexadecimal* number.
8. Convert *decimal* number 232 to an *octal* number.
9. How do we prevent binary numbers 10 and 11 from being confused as decimal numbers?
10. Convert the following *binary* values to *decimal*.
    a. 10011000
    b. 01100101
    c. 10011001
    d. 00010101

11. Convert the following *BCD* values to *decimal*.
    a. 1001 1000
    b. 0110 0101
    c. 1001 1001
    d. 0001 0101
12. The BCD value 1001 0011 0101 is *not*
    a. 935 decimal
    b. 0011 1010 0111 binary
    c. 647 octal
    d. 3A7 hexadecimal
13. The hexadecimal value 2CB is *not*
    a. 715 decimal
    b. 1313 octal
    c. 0010 1100 1011 binary
    d. 0111 0001 0011 BCD
14. Express the following signed decimal numbers in 2s complement. Use 8-bit words. Show all work.
    a. (–)7
    b. (–)4
    c. (–)3
15. Convert the following decimal numbers to 2s complement and add. Use 8-bit words. Show all work.
    a. (+)4
       (–)7
    b. (–)10
       (+)22
    c. (+)22
       (+)33
16. Convert the following negative binary values (8-bit word) to decimal. Show all work.
    a. 10010011
    b. 11001100
    c. 11100111

17. Convert the following decimal numbers to 2s complement and subtract. Use 8-bit words. Show all work.
    a. (+)30
       (+)21
    b. (+)48
       (+)32
    c. (+)67
       (+)116

Chapter 3

Understanding the Input/Output (I/O) Section

Learning Objectives

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

- Describe the I/O section of a programmable logic controller.
- Describe how basic AC and DC I/O modules work.
- Define *optical isolation*.
- Describe why *optical isolation* is used.
- Describe the proper wiring connections for I/O devices and their corresponding modules.
- Explain why a hardwired emergency-stop function is desirable.
- Define the term *interposing*.
- Describe what I/O shielding does.
- List environmental concerns when installing PLCs.

I/O Section

The input/output section (**I/O section**) is the major reason that PLCs are so versatile when used with process machines or driven equipment. The I/O section has the ability to change virtually any type of voltage or current signal into a logic-level signal (typically 5V DC) that is compatible with the PLC processor. The I/O section automatically makes the conversions necessary for the processor to interpret input signals and to activate output devices, even when the I/O devices are of various voltage and current levels.

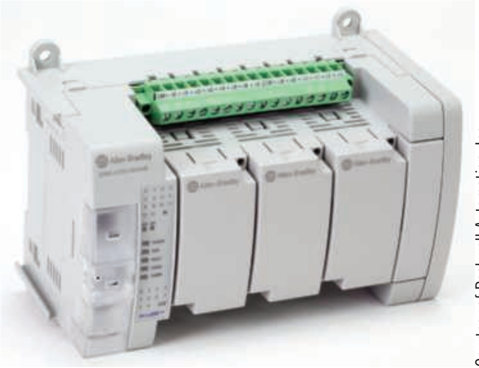
A DC input module, for example, can be used with a 24V DC proximity switch to turn on a 240V AC motor starter coil that is connected to an AC output module. The conversion and interfacing is all accomplished automatically in the I/O section of the PLC, and it is the ease with which the interfacing is accomplished that has made the PLC such a viable tool in industrial and process control.

The input modules of the I/O section provide the status (*ON* or *OFF*) of push buttons, limit switches, proximity switches, and the like, to the processor so decisions can be made to control the machine or process in the proper sequence. Outputs, such as motor starter coils, indicator lights, and solenoids are interfaced to the processor through the output section of the I/O. Once a decision has been made by the processor, a signal is sent to the output section to control the flow of current to the output device. In general, the status of the inputs are relayed to the processor and, based on the logic of the program that has been written, a decision is made to turn the outputs to *ON* or *OFF*. All of the different types and levels of signals (voltages and currents) used in the control process are **interfaced** in the I/O section.

The I/O section generally can be divided into two categories: fixed I/O and modular I/O.

Fixed I/O

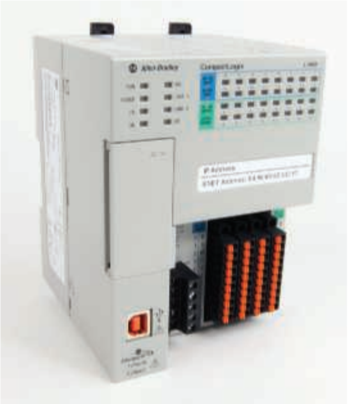
PLCs with fixed I/O typically come in a complete unit that contains the processor, I/O section, and power supply. The I/O section contains a fixed number of inputs and outputs. For example, the Allen-Bradley Micro850 PLC shown in Figure 3–1 has a combination of digital inputs (14) and outputs (10) in a self-contained unit. Like most small PLCs, it can be DIN-rail or panel mounted.



Courtesy of Rockwell Automation, Inc.

Figure 3–1 Allen-Bradley Micro850 PLC with Embedded I/O

If more I/O capability is required or different voltages are needed, expansion units with various I/O configuration can be added to many of the micro and small PLCs on the market today. Figure 3–2 shows an Allen-Bradley 5370 CompactLogix PLC with 16 embedded digital inputs and outputs with the capacity for six optional expansion modules and up to an additional 80 I/O points.



Courtesy of Rockwell Automation, Inc.

Figure 3–2 Allen-Bradley CompactLogix 5370 PLC with Embedded I/O

Micro and small PLCs with fixed I/O typically have a discrete input and output section. As discussed in Chapter 1, discrete-type I/O signals are *ON* or *OFF* and do not vary in level. For example, when a 120V limit switch contact closes or is *ON,* the signal to the input section will be 120V, and the signal will be 0V when the limit switch contact is open (*OFF*). Many manufacturers offer models with a combination of digital and analog inputs and outputs.

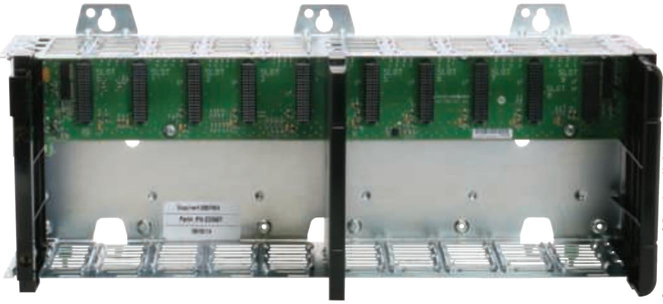
While these PLCs are small in size, they are big on features. Most include full-feature instruction sets, embedded communications ports, real-time clocks, and much more. One should consult the specific dealer for a full list of features.

As the cost of these compact units has decreased, their use has increased. The costs are so competitive that any control processes that uses only a small number of relays and/or timers can now be accomplished using a small micro type PLC. The use of a small PLC not only saves money, but also gives added reliability and flexibility.

Modular I/O

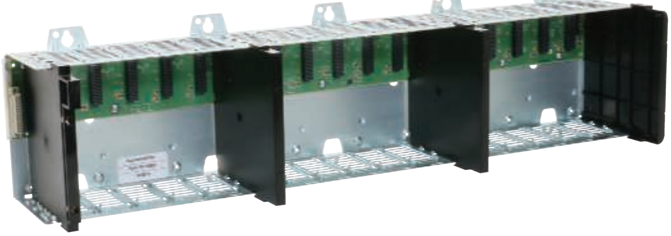
Modular I/O, as the name implies, is modular in nature, more flexible than fixed I/O units, and provides added versatility when it comes to the type and number of I/O devices that can be connected to the system. The various types of I/O modules that make up the I/O section are housed, or installed, in an I/O chassis or rack. Chassis-based I/O modules are specifically designed for a particular PLC controller. You can install the I/O modules locally in the same chassis as the controller, or in some cases remotely through the use of I/O communication networks. In some smaller PLCs, the I/O modules do not actually mount in a physical chassis, rather they connect to each other and the processor to create a rackless design.

The I/O chassis or rack is a framework or housing into which modules are inserted. Figure 3–3a shows a 10-slot I/O chassis and Figure 3–3b a 17-slot chassis. Figure 3–3c shows a chassis with the I/O modules installed to the right of the power supply.



Courtesy of Rockwell Automation, Inc.

Figure 3–3a Allen-Bradley 1756-A10 I/O Chassis



Courtesy of Rockwell Automation, Inc.

Figure 3–3b Allen-Bradley 1756-A17 I/O Chassis

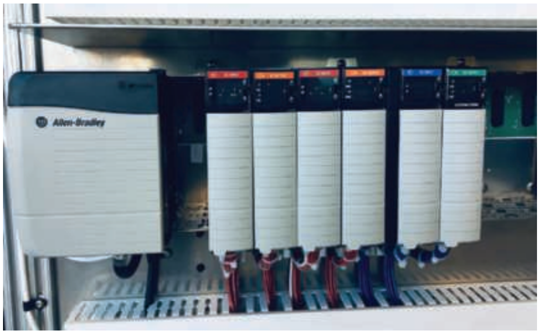


Figure 3–3c I/O Chassis with I/O Modules & Power Supply Installed

Chassis come in many sizes and typically allow for 4 to 17 modules to be inserted. Chassis that contain I/O modules and the PLC controller are referred to as the *local chassis* or local I/O. Chassis that contain I/O modules, remote communication modules, and are mounted separately or away from the PLC controller, are referred to as a *remote chassis* or remote I/O. An advantage of remote I/O is that you can distribute the I/O closer to the field devices (sensors and actuators), reducing wiring costs. The number of remote I/O chassis that a processor can control varies with each manufacturer. The communication between the remote chassis(s) and the PLC controller is accomplished using several different types of I/O communication methods. These methods include coaxial cable, twin axial cable, shielded-twisted pair, or fiber optic cable. If distance or electrical noise are considerations, the fiber optic communication method may be the best option. Figure 3–4 shows a local and remote I/O chassis.

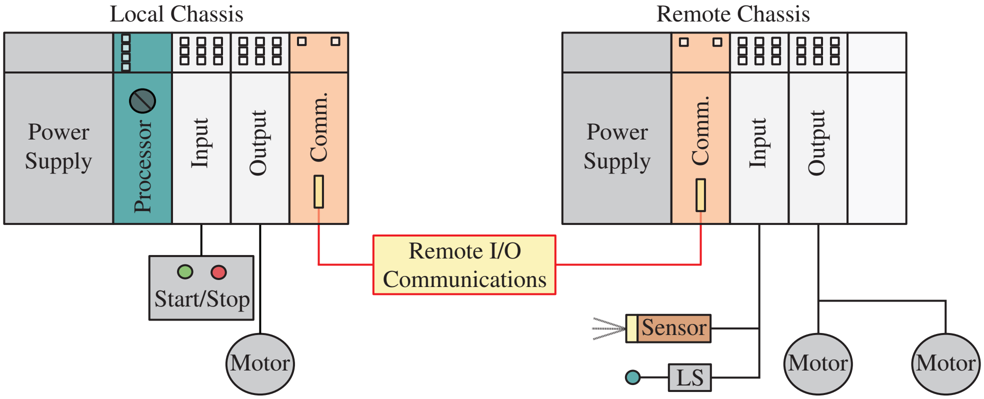


Figure 3–4 Local Chassis and Remote I/O Chassis

Most modern chassis or racks today do not require any additional configurations other than to be mounted and modules installed. On the other hand, older PLC systems required that jumpers or switches had to be set or configured.

I/O modules can be separated into three basic groups: discrete or digital input/output modules, analog input/output modules, and specialized modules.

Discrete I/O Modules

Discrete I/O modules are types of modules that only accept digital or *ON*- and *OFF*-type signals. These modules only recognize these two states or conditions, *ON* or *OFF*. If a discrete device, such as a limit switch, is connected to a digital input module, the module determines the state, or position, of the limit switch, and communicates that status to the processor. If the limit switch contact is open (*OFF*), the module indicates to the processor that the limit switch is *OFF*. This *OFF* condition is stored in the processor memory as a zero (0). Had the limit switch contact been in a closed position, the module would have sent a signal to the processor indicating that the limit switch was *ON,* or closed. The *ON* condition would have been stored in the processor memory as a one (1). All information stored in the processor memory about the status or condition of discrete I/O devices is always in ones and zeros. Understanding this basic concept is critical in learning to program and interpret PLC Ladder Logic.

Discrete modules are the most common type used in a majority of PLC applications and can be divided into two groups: input and output.

Discrete Input Module

The **discrete input module** communicates the status of the various real-world input devices connected to the module (*ON* or *OFF*) to the processor.

Note: As discussed in Chapter 1, the term *real world* is used to indicate that an actual device is involved. As you will learn later in the text, the PLC has the ability to provide timing and counting functions for a machine, even though the timers and counters exist only within the processor, and are not wired into the circuit as with real-world, or actual, devices.

Once the real-world input device is connected, an open or closed electrical circuit exists, depending on the position (open or closed) of the device. The status of the real-world input device is then converted to a logic-level DC electrical signal by the input module and sent to the processor.

Discrete input modules come in a wide range of voltages for various applications. Some of the more common voltages are 120V AC, 240V AC, 24V DC, and 130V DC. Some manufacturers give their modules an AC/DC rating to increase their flexibility and reduce required inventory. It is important to note, however, that while the module may be used with either AC or DC input voltages, the voltages *cannot* be intermixed on the same module.

Input modules can be purchased with a wide range of input terminals or points, which determine the number of individual field devices that can be connected to the module. Common sizes, depending on the manufacturer, are 8, 16, and 32 points. Sixteen- and 32-point modules are often referred to as high-density modules since they are physically the same size as an 8-point module. High-density modules usually provide lower cost per point, or input device, but are also more difficult to wire. The increased difficulty in wiring is caused by the closer proximity of the wiring terminals and the increased number of wires in the wiring harness.

AC Discrete Input Module

Figure 3–5 shows a simplified diagram of one of the input circuits of a typical AC discrete input module. Resistors and capacitors are used to filter and limit the voltage signal. The filter requires that the AC signal be not only of a specific value but also be present for a specific amount of time before the module views it as a real signal and communicates the results to the processor. A valid signal is relayed through an **optically coupled** circuit, across the backplane of the I/O chassis to the processor.

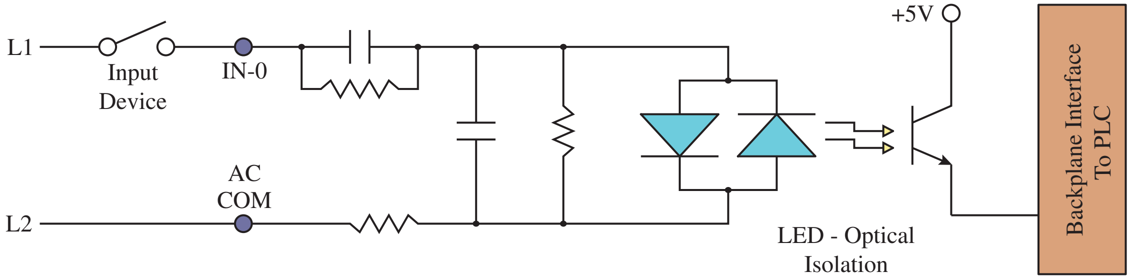


Figure 3–5 Simplified AC Input Module Circuit

The optically coupled circuit uses a light-emitting diode (LED) to turn *ON*, or forward bias, a photo transistor to complete the electrical circuit to the processor. When the LED is turned *ON* to indicate that the actual input device has closed its contact and is providing a voltage to the input terminal, the light from the LED is picked up by the photo transistor, which makes the transistor conduct or switch on, completing a 5V DC logic circuit, and the status of the input is communicated to the processor. This form of optical coupling is also referred to as **optical isolation**. By employing optical coupling or isolation, there is no actual electrical connection between the input device and the processor. This eliminates any possibility of the input line voltage, i.e., 120 or 240V AC, from coming in contact with and damaging the low-voltage DC section of the processor or I/O module. Optical isolation also protects the processor and I/O module from electrical noise, voltage transients, or spikes. In summary, optical isolation prevents any unwanted voltage from the I/O section from reaching the logic section of the processor or I/O module.

Individual status lights are provided on the front of I/O modules for each input terminal (Figure 3–6). The status light is *ON* when the input device is providing a voltage to the input terminal (contact closed) and is *OFF* when a voltage is absent (contact open). With the status lights showing the actual state of the various input devices connected to the input module, they make a valuable troubleshooting aid. The electrician or technician need only look at the input status lights on the input module to determine the state, or status, of any input device.

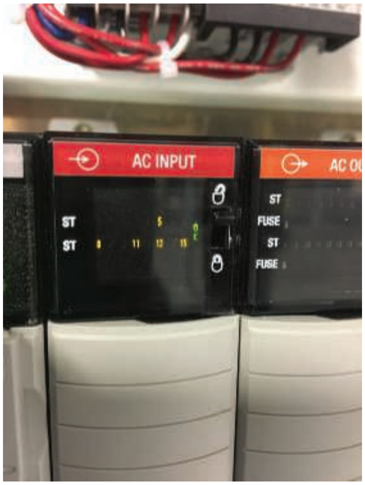
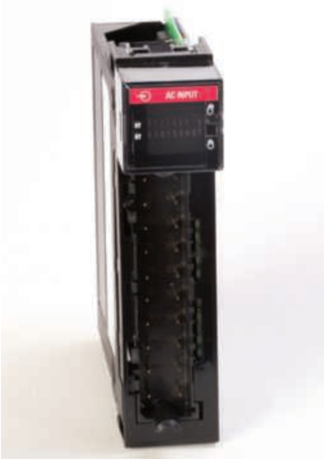


Figure 3–6 AC Input Module with Input Indicator Lights

A typical I/O module consists of two parts: a printed circuit board and a removable terminal block (RTB) assembly. The printed circuit board plugs into a slot, or connector, in the I/O chassis, or in the case of a rackless system, the adjacent module and contains the **solid-state** electronic circuits that interface the I/O devices with the processor. The RTB assembly then attaches to the front edge of the printed circuit board, which may or may not have a protective cover, depending on the manufacturer. Figure 3–7 shows a typical 16-point AC input module *without* the RTB attached.



Courtesy of Rockwell Automation, Inc.

Figure 3–7 16-Point AC Input Module

Figure 3–8 shows how the input module is installed in the I/O chassis.

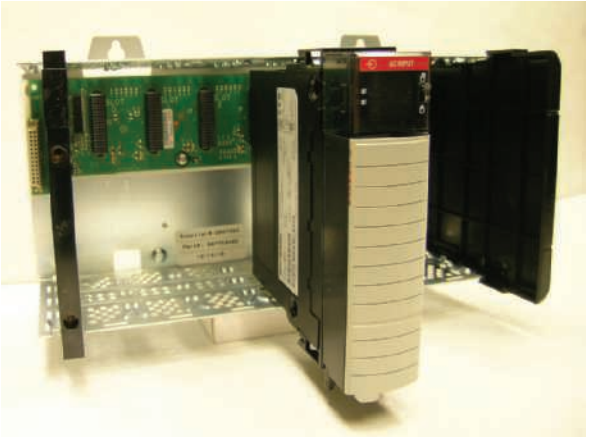


Figure 3–8 Installing I/O Module in I/O Chassis

After the input modules have been installed in the I/O chassis or rack, they are ready to have one side of each input device connected to their RTB (Figure 3–9). Do to the density and size of the terminal blocks, you may find it easier to wire the removable terminal blocks before installing them on the module. In some cases, as an alternative to buying RTBs and connecting the wires yourself, you can buy a wiring system that connects to the I/O modules through prewired and pretested cables.

RTBs allow I/O modules to be replaced without unwiring them. Figure 3–10 shows the RTB installed on the I/O module.

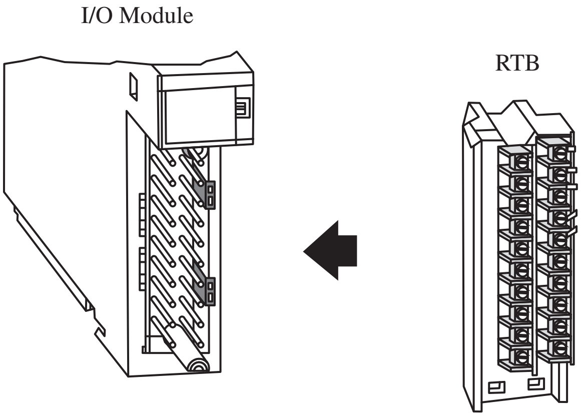


Figure 3–9 Removable Terminal Block (RTB)

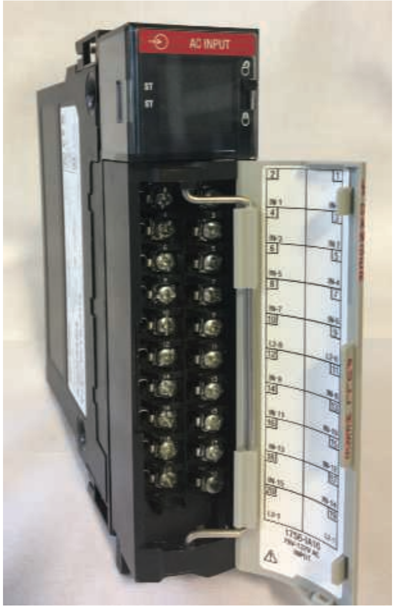


Figure 3–10 Removable Terminal Block (RTB) Installed on I/O Module

While each input device has two wires connected, only one wire is connected directly to the input module. The other wire of each input device is connected to L1 (Figure 3–11a). The same connection scheme is used for 8-, 16-, or 32-point input modules. Figure 3–11b shows the wiring connections for a typical 120V AC 16-point input module.

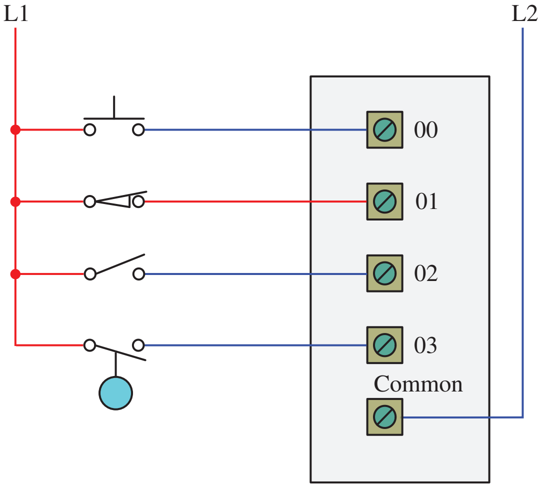


Figure 3–11a Field Wiring for AC Input Module

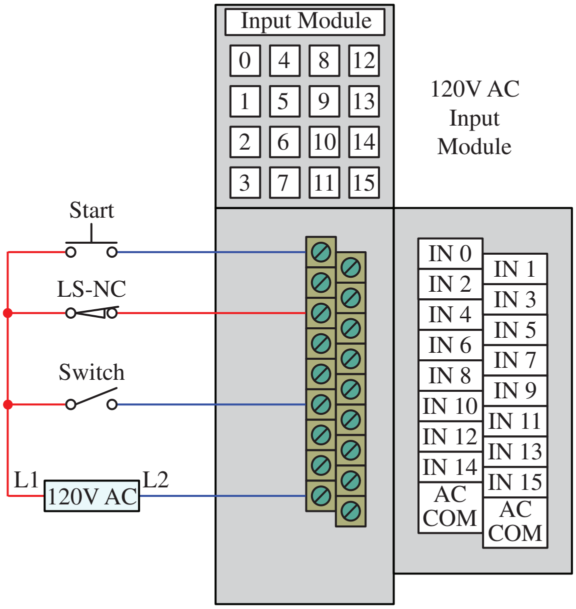


Figure 3–11b Typical Connections for a 120V AC 16-Point Input Module

The wires from the individual devices are referred to as **field wiring**, since the wires are external to the PLC and are connected in the field. On larger PLC systems, the field wiring that is brought into the I/O chassis or rack can consist of hundreds of wires. The basic rule is that one side of each input device is wired to a hot conductor (L1 for AC or + for DC), and the other side of the device is wired to an input terminal on the RTB of the input module. The input module has a common connection for the neutral, or grounded potential (L2), for AC modules, and the negative (–) for DC modules. Consult the manufacturer's literature that comes with each input module to ensure that the correct wiring connections are made.

Figure 3–12 shows two simplified circuits. In the first, or traditional circuit, the input device (single-pole switch) is connected to, and controls, the light. In the PLC circuit, the input device is connected directly to the input module instead of the light. The module converts the AC input signal to 5V DC, and communicates the status of the single-pole switch to the processor which, in turn, controls a light that is connected to an output module.

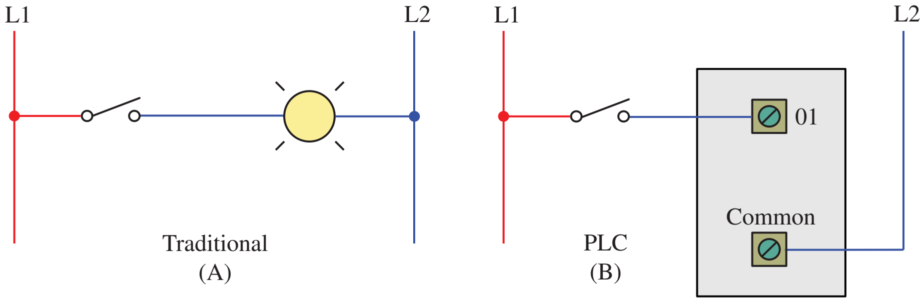


Figure 3–12 Traditional Wiring for Single Pole Switch Compared to PLC Wiring