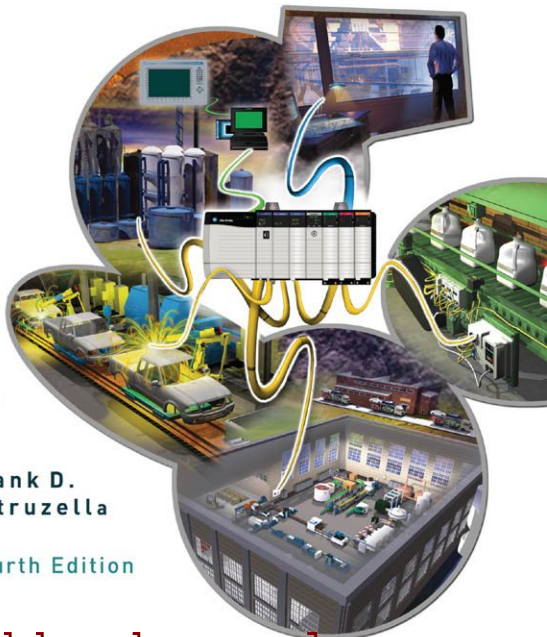


PROGRAMMABLE LOGIC CONTROLLERS

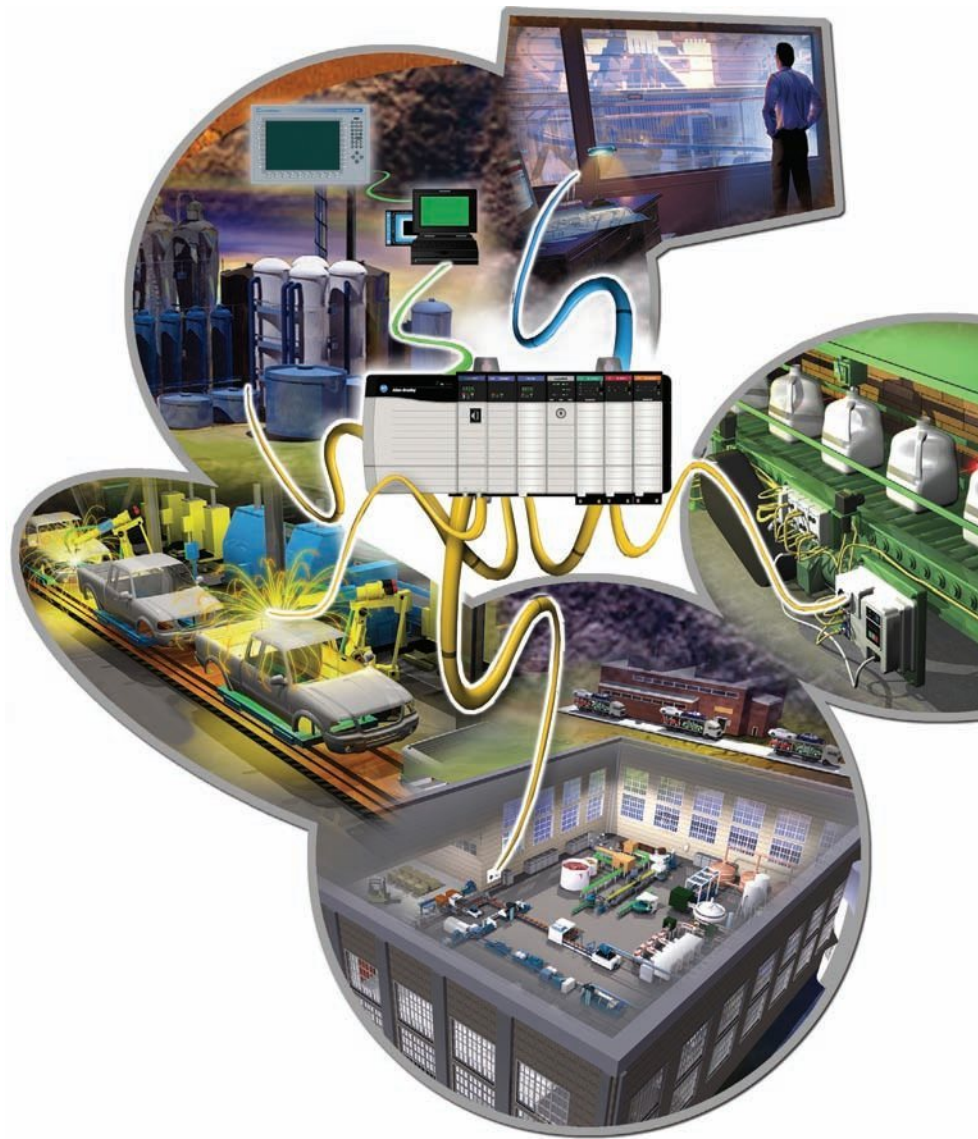


Frank D.
Petruzella

Fourth Edition

.Techbooksyard.com

PROGRAMMABLE LOGIC CONTROLLERS



This page intentionally left blank



PROGRAMMABLE LOGIC CONTROLLERS

Fourth Edition



Frank D.
Petruzella





PROGRAMMABLE LOGIC CONTROLLERS

Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY, 10020. Copyright © 2011 by The McGraw-Hill Companies, Inc. All rights reserved. Previous editions © 1989, 1998, and 2005. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of The McGraw-Hill Companies, Inc., including, but not limited to, in any network or other electronic storage or transmission, or broadcast for distance learning.

Some ancillaries, including electronic and print components, may not be available to customers outside the United States.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 0 DOW/DOW 1 0 9 8 7 6 5 4 3 2 1 0

ISBN 978-0-07-351088-0

MHID 0-07-351088-2

Vice president/Editor in chief: *Elizabeth Haefele*

Vice president/Director of marketing: *John E. Biernat*

Director of Development: *Sarah Wood*

Freelance developmental editor: *Kirsten Guidero*

Editorial coordinator: *Vincent Bradshaw*

Marketing manager: *Kelly Curran*

Lead digital product manager: *Damian Moshak*

Digital development editor: *Kevin White*

Director, Editing/Design/Production: *Jess Ann Kosic*

Project manager: *Jean R. Starr*

Buyer II: *Debra R. Sylvester*

Senior designer: *Srdjan Savanovic*

Senior photo research coordinator: *Lori Hancock*

Photo researcher: *Danny Meldung*

Cover design: *George Kokkonas*

Interior design: *Kay Lieberherr*

Typeface: *11/13 Times*

Compositor: *MPS Limited, A Macmillan Company*

Printer: *R. R. Donnelley*

Cover credit: *Cover Image Used with Permission of Rockwell Automation, Inc.*

Library of Congress Cataloging-in-Publication Data

Petruzella, Frank D.

Programmable logic controllers / Frank D. Petruzella. — 4th ed.

p. cm.

Includes index.

ISBN-13: 978-0-07-351088-0 (alk. paper)

ISBN-10: 0-07-351088-2 (alk. paper)

1. Programmable controllers. I. Title.

TJ223.P76P48 2011

629.8'95—dc22

2010025533

The Internet addresses listed in the text were accurate at the time of publication. The inclusion of a Web site does not indicate an endorsement by the authors or McGraw-Hill, and McGraw-Hill does not guarantee the accuracy of the information presented at these sites.

Contents

Preface	ix	3.8	ASCII Code.	51
Acknowledgments	xii	3.9	Parity Bit.	51
About the Author	xiii	3.10	Binary Arithmetic	51
Walkthrough.	xiv		Review Questions.	55
			Problems	56
Chapter 1 Programmable Logic Controllers (PLCs): An Overview	1	Chapter 4 Fundamentals of Logic	57	
1.1 Programmable Logic Controllers	2	4.1 The Binary Concept	58	
1.2 Parts of a PLC	4	4.2 AND, OR, and NOT Functions.	58	
1.3 Principles of Operation	8	<i>The AND Function</i>	58	
1.4 Modifying the Operation	11	<i>The OR Function</i>	59	
1.5 PLCs versus Computers	11	<i>The NOT Function</i>	60	
1.6 PLC Size and Application.	12	<i>The Exclusive-OR (XOR) Function</i>	61	
Review Questions.	15	4.3 Boolean Algebra	61	
Problems	15	4.4 Developing Logic Gate Circuits from Boolean Expressions	63	
Chapter 2 PLC Hardware Components	17	4.5 Producing the Boolean Equation for a Given Logic Gate Circuit	63	
2.1 The I/O Section.	18	4.6 Hardwired Logic versus Programmed Logic	64	
2.2 Discrete I/O Modules	22	4.7 Programming Word Level Logic Instructions	67	
2.3 Analog I/O Modules	27	Review Questions.	69	
2.4 Special I/O Modules	29	Problems	69	
2.5 I/O Specifications	32			
<i>Typical Discrete I/O Module Specifications</i>	32	Chapter 5 Basics of PLC Programming	71	
<i>Typical Analog I/O Module Specifications</i>	33	5.1 Processor Memory Organization	72	
2.6 The Central Processing Unit (CPU)	33	<i>Program Files</i>	72	
2.7 Memory Design	35	<i>Data Files</i>	72	
2.8 Memory Types	36	5.2 Program Scan	76	
2.9 Programming Terminal Devices	37	5.3 PLC Programming Languages	79	
2.10 Recording and Retrieving Data.	38	5.4 Relay-Type Instructions	81	
2.11 Human Machine Interfaces (HMIs)	38	5.5 Instruction Addressing	84	
Review Questions.	40	5.6 Branch Instructions.	85	
Problems	42	5.7 Internal Relay Instructions	87	
Chapter 3 Number Systems and Codes	43	5.8 Programming Examine If Closed and Examine If Open Instructions.	88	
3.1 Decimal System	44	5.9 Entering the Ladder Diagram	89	
3.2 Binary System.	44	5.10 Modes of Operation	91	
3.3 Negative Numbers.	46	Review Questions.	92	
3.4 Octal System.	47	Problems	93	
3.5 Hexadecimal System.	48			
3.6 Binary Coded Decimal (BCD) System.	48			
3.7 Gray Code.	50			

Chapter 6 Developing Fundamental PLC Wiring Diagrams and Ladder Logic Programs 95

6.1 Electromagnetic Control Relays 96
6.2 Contactors 97
6.3 Motor Starters 98
6.4 Manually Operated Switches 99
6.5 Mechanically Operated Switches 100
6.6 Sensors 101
 Proximity Sensor 101
 Magnetic Reed Switch 104
 Light Sensors 104
 Ultrasonic Sensors 106
 Strain/Weight Sensors 106
 Temperature Sensors 107
 Flow Measurement 107
 Velocity and Position Sensors 108
6.7 Output Control Devices 108
6.8 Seal-In Circuits 110
6.9 Latching Relays 111
6.10 Converting Relay Schematics into PLC Ladder Programs 116
6.11 Writing a Ladder Logic Program Directly from a Narrative Description 119
Review Questions 122
Problems 123

Chapter 7 Programming Timers 125

7.1 Mechanical Timing Relays 126
7.2 Timer Instructions 128
7.3 On-Delay Timer Instruction 129
7.4 Off-Delay Timer Instruction 133
7.5 Retentive Timer 136
7.6 Cascading Timers 140
Review Questions 144
Problems 144

Chapter 8 Programming Counters 149

8.1 Counter Instructions 150
8.2 Up-Counter 152
 One-Shot Instruction 155
8.3 Down-Counter 159
8.4 Cascading Counters 162
8.5 Incremental Encoder-Counter Applications 165
8.6 Combining Counter and Timer Functions 166
Review Questions 171
Problems 171

Chapter 9 Program Control Instructions 176

9.1 Master Control Reset Instruction 177
9.2 Jump Instruction 180
9.3 Subroutine Functions 181
9.4 Immediate Input and Immediate Output Instructions 184
9.5 Forcing External I/O Addresses 187
9.6 Safety Circuitry 190
9.7 Selectable Timed Interrupt 193
9.8 Fault Routine 194
9.9 Temporary End Instruction 194
9.10 Suspend Instruction 195
Review Questions 196
Problems 196

Chapter 10 Data Manipulation Instructions 200

10.1 Data Manipulation 201
10.2 Data Transfer Operations 201
10.3 Data Compare Instructions 209
10.4 Data Manipulation Programs 213
10.5 Numerical Data I/O Interfaces 216
10.6 Closed-Loop Control 218
Review Questions 222
Problems 223

Chapter 11 Math Instructions 226

11.1 Math Instructions 227
11.2 Addition Instruction 227
11.3 Subtraction Instruction 229
11.4 Multiplication Instruction 230
11.5 Division Instruction 231
11.6 Other Word-Level Math Instructions 233
11.7 File Arithmetic Operations 235
Review Questions 237
Problems 238

Chapter 12 Sequencer and Shift Register Instructions 242

12.1 Mechanical Sequencers 243
12.2 Sequencer Instructions 245
12.3 Sequencer Programs 248
12.4 Bit Shift Registers 254
12.5 Word Shift Operations 260
Review Questions 264
Problems 264

Chapter 13 PLC Installation Practices, Editing, and Troubleshooting	268		
13.1	PLC Enclosures	269	
13.2	Electrical Noise	271	
13.3	Leaky Inputs and Outputs	272	
13.4	Grounding	272	
13.5	Voltage Variations and Surges	274	
13.6	Program Editing and Commissioning	275	
13.7	Programming and Monitoring	276	
13.8	Preventive Maintenance	278	
13.9	Troubleshooting	279	
	<i>Processor Module</i>	279	
	<i>Input Malfunctions</i>	279	
	<i>Output Malfunctions</i>	281	
	<i>Ladder Logic Program</i>	281	
13.10	PLC Programming Software	286	
	Review Questions	288	
	Problems	288	
Chapter 14 Process Control, Network Systems, and SCADA	291		
14.1	Types of Processes	292	
14.2	Structure of Control Systems	294	
14.3	On/Off Control	296	
14.4	PID Control	297	
14.5	Motion Control	301	
14.6	Data Communications	303	
	<i>Data Highway</i>	308	
	<i>Serial Communication</i>	308	
	<i>DeviceNet</i>	308	
	<i>ControlNet</i>	310	
	<i>EtherNet/IP</i>	311	
	<i>Modbus</i>	311	
	<i>Fieldbus</i>	312	
	<i>PROFIBUS-DP</i>	313	
14.7	Supervisory Control and Data Acquisition (SCADA)	313	
	Review Questions	315	
	Problems	316	
Chapter 15 ControlLogix Controllers	317		
Part 1 Memory and Project Organization	318		
	Memory Layout	318	
	Configuration	318	
	Project	319	
	Tasks	320	
	Programs	320	
	Routines	321	
	Tags	321	
	Structures	324	
	Creating Tags	325	
	Monitoring and Editing Tags	326	
	Array	326	
	Review Questions	328	
Part 2 Bit-Level Programming	329		
	Program Scan	329	
	Creating Ladder Logic	330	
	Tag-Based Addressing	331	
	Adding Ladder Logic to the Main Routine	332	
	Internal Relay Instructions	334	
	Latch and Unlatch Instructions	334	
	One-Shot Instruction	335	
	Review Questions	336	
	Problems	336	
Part 3 Programming Timers	338		
	Timer Predefined Structure	338	
	On-Delay Timer (TON)	339	
	Off-Delay Timer (TOF)	342	
	Retentive Timer On (RTO)	344	
	Review Questions	346	
	Problems	346	
Part 4 Programming Counters	347		
	Counters	347	
	Count-Up (CTU) Counter	348	
	Count-Down (CTD) Counter	350	
	Review Questions	352	
	Problems	352	
Part 5 Math, Comparison, and Move Instructions	353		
	Math Instructions	353	
	Comparison Instructions	355	
	Move Instructions	357	
	Review Questions	360	
	Problems	360	
Part 6 Function Block Programming	361		
	Function Block Diagram (FBD)	361	
	FBD Programming	365	
	Review Questions	371	
	Problems	371	
	Glossary	373	
	Index	385	

This page intentionally left blank



Preface

Programmable logic controllers (PLCs) continue to evolve as new technologies are added to their capabilities. The PLC started out as a replacement for hardwired relay control systems. Gradually, various math and logic manipulation functions were added. Today PLCs are the controller of choice for the vast majority of automated processes. PLCs now incorporate smaller cases, faster CPUs, networking, and various Internet technologies.

This Fourth Edition of *Programmable Logic Controllers* continues to provide an up-to-date introduction to all aspects of PLC programming, installation, and maintenance procedures. No previous knowledge of PLC systems or programming is assumed. As one reviewer of this edition put it: “I honestly believe that someone with little or no background to PLC systems could take this book and teach themselves PLCs.”

The primary source of information for a particular PLC is always the accompanying user manuals provided by the manufacturer. This textbook is not intended to replace the vendor’s reference material but rather to complement, clarify, and expand on this information. With the current number of different types of PLCs on the market it is not practical to cover the specifics of all manufacturers and models in a single text. With this in mind, the text discusses PLCs in a generic sense. Although the content is of a nature to allow the information to be applied to a variety of PLCs from different manufacturers, this book, for the most part, uses the Allen-Bradley SLC 500 and ControlLogix controller instruction sets for the programming examples. The underlying PLC principles and concepts covered in the text are common to most manufacturers and serve to maximize the knowledge gained through attending PLC training programs offered by different vendors.

The text is written at a level and format understandable to students being introduced to PLCs for the first time. Feedback from instructors indicates that the information is well organized, to the point, and easy to understand. The content of this new Fourth Edition has been updated and reflects the changes in technology since the publication of the previous edition.

Each chapter begins with a brief introduction outlining chapter coverage and learning objectives. When applicable, the relay equivalent of the virtual programmed instruction is explained first, followed by the appropriate PLC instruction. Chapters conclude with a set of review questions and problems. The review questions are closely related to the chapter objectives and require students to recall and apply information covered in the chapter. The problems range from easy to difficult, thus challenging students at various levels of competence.

This new Fourth Edition has been revised to include a number of new features:

How Programs Operate When the operation of a program is called for, a bulleted list is used to summarize its execution. The list is used in place of lengthy paragraphs and is especially helpful when explaining the different steps in the execution of a program.

Representation of I/O Field Devices Recognition of the input and output field devices associated with the program helps in understanding the overall operation of the program. With this in mind, in addition to their symbols, we provide drawings and photos of field input and output devices.

New ControlLogix Chapter Some instructors have felt that students tend to get confused when switching back and forth from SLC 500 Logic to Logix 5000-based programming within the same chapter. For this reason, a ***new Chapter 15*** has been added that is devoted entirely to the Allen-Bradley ControlLogix family of controllers and the RSLogix 5000 software. Each part of the new Chapter 15 is treated as a separate unit of study and includes ControlLogix:

- Memory and Project Organization
- Bit-Level Programming
- Programming Timers
- Programming Counters
- Math, Comparison, and Move Instructions
- Function Block Programming

Chapter changes in this edition include:

Chapter 1

- Drawings and photos of real world field input and output devices have been added.
- 50% more figures have been added to this chapter to increase visual appeal and illustrate key concepts further.
- Most recent photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 2

- Drawings and photos of real world field input and output devices have been added.
- Information on the latest selection of PLC hardware components.
- Human machine interfacing with Pico controllers added.
- Most recent photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 3

- Improvement in sizing and placement of drawings make explanations of the different number systems easier to follow.

Chapter 4

- Improvement in sizing and placement of drawings make explanations easier to follow.
- Drawings and photos of real world field input and output devices have been added to the logic diagrams.

Chapter 5

- Information on the ControlLogix memory organization relocated to chapter 15.
- Program scan process explained in greater detail.
- Extended coverage of relay type instructions.
- Instruction addressing examined in greater detail.
- Addressing of a micro PLC illustrated.
- Revisions to chapter review questions and problems.

Chapter 6

- Drawings and photos of real world field input and output devices have been added.

- Drawings and photos of real world field input and output devices have been included in the ladder logic programs.
- Wiring of field inputs and outputs to a micro PLC illustrated.
- Additional coverage of hardwired motor control circuits and their PLC equivalent.
- Revisions to chapter review questions and problems.

Chapter 7

- Information on the ControlLogix timers relocated to chapter 15.
- Drawings and photos of real world field input and output devices have been included in the ladder logic programs.
- Bulleted lists used to summarize program execution.
- Most recent photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 8

- Information on the ControlLogix counters relocated to chapter 15.
- Drawings and photos of real world field input and output devices have been included in the ladder logic programs.
- Bulleted lists used to summarize program execution.
- Most recent photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 9

- Drawings and photos of real world field input and output devices have been included in the ladder logic programs.
- Forcing of inputs and outputs covered in greater detail.
- Differences between a safety PLC and a standard PLC explained.
- Bulleted lists used to summarize program execution.
- Most recent photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 10

- Drawings and photos of real world field input and output devices have been included in the ladder logic programs.

- Analog control covered in more detail.
- PID control process explained in a simplified manner.
- Bulleted lists used to summarize program execution.
- Most recent photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 11

- Drawings and photos of real world field input and output devices have been included in the ladder logic programs.
- Improvement in sizing and placement of drawings make explanations of the different math instructions easier to follow.
- Bulleted lists used to summarize program execution.
- Most recent photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 12

- Drawings and photos of real world field input and output devices have been included in the sequencer programs.
- Improvements to sequencer line drawings designed to make this instruction easier to follow.
- Bulleted lists used to summarize program execution.
- Most recent photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 13

- Drawings and photos of real world field input and output devices have been added.
- Safety issues examined in greater detail.
- Extended coverage of practical troubleshooting techniques.
- Improvement to PLC grounding diagrams makes this function easier to follow.
- Bulleted lists used to summarize program execution.
- Most recent photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 14

- All pertinent information from Chapters 14 and 15 of the 3rd edition have been incorporated into this chapter.

- Examines communications at all levels in an industrial network in much greater detail.
- Fundamentals of PLC motion control have been added.
- Bulleted lists used to summarize program execution.
- New photographs from major PLC manufactures.
- Revisions to chapter review questions and problems.

Chapter 15

- Completely new chapter that concentrates on the fundamentals of ControlLogix technology.
- Includes Memory and Project Organization, Bit Level Programming, Timers, Counters, Math Instructions, and Function Block Programming.

Ancillaries

- **Activity Manual for Programmable Logic Controllers**, Fourth Edition.

This manual contains:

Tests made up of multiple choice, true/false, and completion-type questions for each of the chapters.

Generic programming hands-on exercises designed to offer students real-world programming experience. These assignments are designed for use with any brand of PLC.

- **LogixPro PLC Lab Manual for use with Programmable Logic Controllers**, Fourth Edition

This manual contains:

LogixPro 500 simulator software CD. The LogixPro simulation software converts the student's computer into a PLC and allows the student to write ladder logic programs and verify their real-world operation.

Over **250 LogixPro student lab exercises** sequenced to support material covered in the text.

- **Instructor's Resource Center** is available to instructors who adopt *Programmable Logic Controllers*, Fourth Edition. It includes:

Textbook answers to all questions and problems.

Activity Manual answers to all tests.

Computer Simulation Lab Manual answers for all programming exercises.

PowerPoint presentations for each chapter.

EZ Test testing software with text-coordinated question banks.

ExamView text-coordinated question banks.

Acknowledgments

I would like to thank the following reviewers for their comments and suggestions:

Wesley Allen
Jefferson State Community College

Bo Barry
University of North Carolina–Charlotte

David Barth
Edison Community College

Michael Brumbach
York Technical College

Fred Cope
Northeast State Technical Community College

Warren Dejardin
Northeast Wisconsin Technical College

Montie Fleshman
New River Community College

Steven Flinn
Illinois Central College

Brent Garner
McNeese State University

John Haney
Snead State Community College

Thomas Heraly
Milwaukee Area Technical College

John Lukowski
Michigan Technical University

John Martini
University of Arkansas–Fort Smith

Steven McPherson
Sauk Valley Community College

Max Neal
Griffin Technical College

Ralph Neidert
NECA/IBEW Local 26 JATC

Chrys Panayiotou
Indian River State College

Don Pelster
Nashville State Technical Community College

Dale Petty
Washtenaw Community College

Sal Pisciotta
Florence-Darlington Technical College

Roy E. Pruett
Bluefield State College

Melvin Roberts
Camden County College

Farris Saifkani
Northeast Wisconsin Technical College

David Setser
Johnson County Community College

Richard Skelton
Jackson State Community College

Amy Stephenson
Pitt Community College

William Sutton
ITT Technical Institute

John Wellin
Rochester Institute of Technology

Last but not least, special thanks to Wade Wittmus of Lakeshore Technical College, not only for his extended help with reviews but also for his outstanding work on the supplements.

Frank D. Petruzella

About the Author

Frank D. Petruzella has extensive practical experience in the electrical control field, as well as many years of experience teaching and authoring textbooks. Before becoming a full time educator, he was employed as an apprentice and electrician in areas of electrical installation and

maintenance. He holds a Master of Science degree from Niagara University, a Bachelor of Science degree from the State University of New York College–Buffalo, as well as diplomas in Electrical Power and Electronics from the Erie County Technical Institute.

Additional coverage of communications and control networks utilizes clear graphics to demonstrate how things work

The scan is normally a continuous and sequential process of reading the status of inputs, evaluating the control logic, and updating the outputs. Figure 5-8 shows an overview of the data flow during the scan process. For each rung executed, the PLC processor will:

- Examine the status of the input image table bits.
- Solve the ladder logic in order to determine logical continuity.
- Update the appropriate output image table bits, if necessary.
- Copy the output image table status to all of the output terminals. Power is applied to the output device if the output image table bit has been previously set to a 1.
- Copy the status of all of the input terminals to the input image table. If an input is active (i.e., there is electrical continuity), the corresponding bit in the input image table will be set to a 1.

Diagrams, such as this one illustrating an overview of the function block programming language, help students put the pieces together

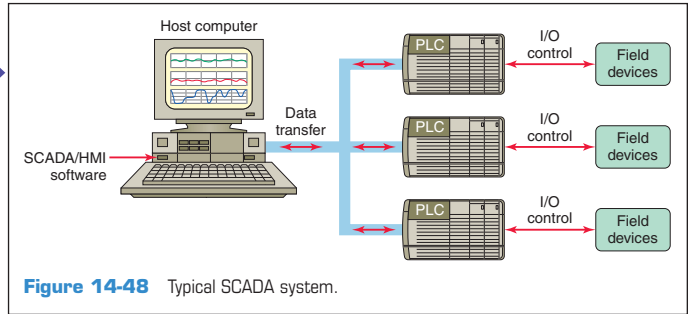


Figure 14-48 Typical SCADA system.

BULLETED LISTS break down processes to helpfully summarize execution of tasks

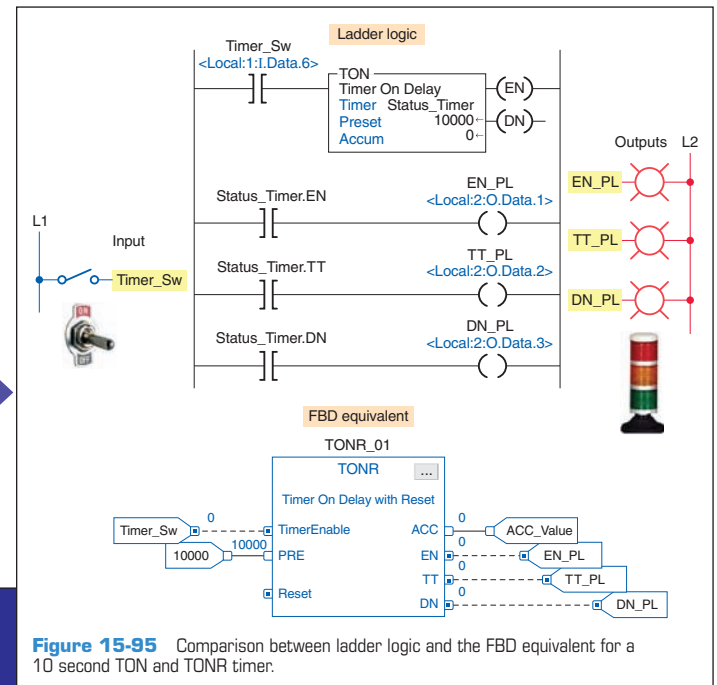


Figure 15-95 Comparison between ladder logic and the FBD equivalent for a 10 second TON and TONR timer.



Figure 15-1 Programmable automation controllers (PACs). Source: Image Used with Permission of Rockwell Automation, Inc.

An entirely new chapter on ControlLogix has been added to familiarize students with the entire Allen-Bradley family of controllers and RSLogix 5000 software

CHAPTER 3 REVIEW QUESTIONS

- Convert each of the following binary numbers to decimal numbers:
 - 10
 - 100
 - 111
 - 1011
 - 1100
 - 10010
 - 10101
 - 11111
 - 11001101
 - 11100011
- Convert each of the following decimal numbers to binary numbers:
 - 7
 - 19
 - 28
 - 46
 - 57
 - 86
 - 94
 - 112
 - 148
 - 230
- Convert each of the following octal numbers to decimal numbers:
 - 36
 - 104
- Convert each of the following hexadecimal numbers to binary numbers:
 - 4C
 - E8
 - 6D2
 - 31B
- Convert each of the following decimal numbers to BCD:
 - 146
 - 389
 - 1678
 - 2502
- What is the most important characteristic of the Gray code?
- What makes the binary system so applicable to computer circuits?
- Define the following as they apply to the binary memory locations or registers:
 - Bit
 - Byte
 - Word
 - LSB
 - MSB
- State the base used for each of the following number systems:

CHAPTER 3 PROBLEMS

- The following binary PLC coded information is to be programmed using the hexadecimal code. Convert each piece of binary information to the appropriate hexadecimal code for entry into the PLC from the keyboard.
 - 0001 1111
 - 0010 0101
 - 0100 1110
 - 0011 1001
- The encoder circuit shown in Figure 3-17 is used to convert the decimal digits on the keyboard to a binary code. State the output status (HIGH/LOW) of A-B-C-D when decimal number
 - 2 is pressed.
 - 5 is pressed.
 - 7 is pressed.
 - 8 is pressed.
- If the bits of a 16-bit word or register are numbered according to the octal numbering system, beginning with 00, what consecutive numbers would be used to represent each of the bits?
- Express the decimal number 18 in each of the following number codes:
 - Binary
 - Octal
 - Hexadecimal
 - BCD

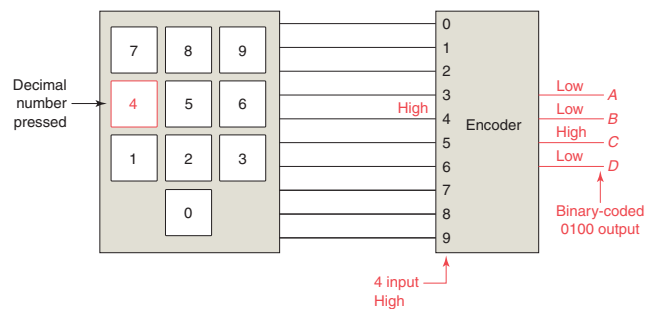


Figure 3-17 Diagram for Problem 2.

EXAMPLE PROBLEMS help bring home the applicability of chapter concepts

ANCILLARIES THAT WORK

Expanded on and updated from the previous edition, this new edition includes an outstanding instructor support package:

- ExamView and EZ Test question test banks for each chapter.
- PowerPoint lessons with animations that help visualize the actual process.
- Activity Manual contains true/false, completion, matching, and multiple-choice questions for every chapter in the text. So that students get a better understanding of programmable logic controllers, the manual also includes a wide range of programming assignments and additional practice exercises.
- On-line Instructor's Resource Center.



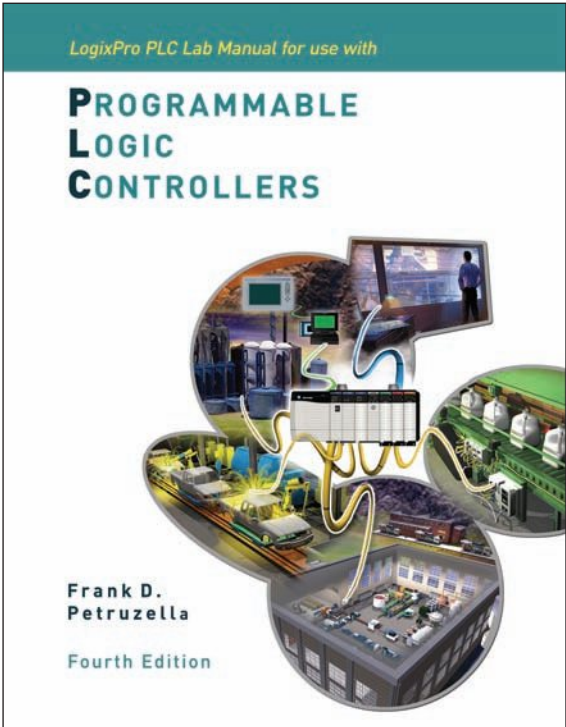
The screenshot shows the Online Learning Center interface for the textbook. It includes a navigation menu with links for 'About the Author', 'Book Preface', 'Sample Chapter', 'Student Learning Aids', 'Supplements', and 'Table of Contents'. The main content area displays the book title 'Programmable Logic Controllers, 4/e' by Frank Petruzella, along with the ISBN (0073510662) and copyright year (2011). A quote from a reviewer is featured: 'This fourth edition of Programmable Logic Controllers continues to provide an up-to-date introduction to all aspects of PLC programming, installation, and maintaining procedures. No previous knowledge of PLC systems or programming is assumed. As one reviewer of this edition put it, "I honestly believe that someone with little or no background to PLC systems could take this book and teach themselves PLCs."' A 'LogixPro LearningCenter' logo is also present. At the bottom, there is a note about obtaining an instructor login and a copyright notice for McGraw-Hill Higher Education.



The cover of the 'Activities Manual to accompany Programmable Logic Controllers, Fourth Edition' by Frank D. Petruzella. The cover features a central graphic of a PLC rack with various components and a person working at a computer. The title 'PROGRAMMABLE LOGIC CONTROLLERS' is prominently displayed in large, bold letters. The author's name 'Frank D. Petruzella' and 'Fourth Edition' are also visible.

In addition, for students, this edition also has available:

- *LogixPro PLC Lab Manual for use with Programmable Logic Controllers* Fourth Edition, with LogixPro PLC Simulator. This manual contains:
 - LogixPro 500 simulator software CD. The LogixPro simulation software converts the student's computer into a PLC and allows the student to write ladder logic programs and verify their real-world operation.
 - Over 250 LogixPro student lab exercises sequenced to support material covered in the text.



The cover of the 'LogixPro PLC Lab Manual for use with Programmable Logic Controllers, Fourth Edition' by Frank D. Petruzella. The cover features a central graphic of a PLC rack with various components and a person working at a computer. The title 'PROGRAMMABLE LOGIC CONTROLLERS' is prominently displayed in large, bold letters. The author's name 'Frank D. Petruzella' and 'Fourth Edition' are also visible.

1

Programmable Logic Controllers (PLCs) An Overview



Image Used with Permission of Rockwell Automation, Inc.

Chapter Objectives

After completing this chapter, you will be able to:

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

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

1.1 Programmable Logic Controllers

Programmable logic controllers (Figure 1-1) are now the most widely used industrial process control technology. A programmable logic controller (PLC) is an industrial grade computer that is capable of being programmed to perform control functions. The programmable controller has eliminated much of the hardwiring associated with conventional relay control circuits. Other benefits include easy programming and installation, high control speed, network compatibility, troubleshooting and testing convenience, and high reliability.

The programmable logic controller is designed for multiple input and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs for the control and operation of manufacturing process equipment and machinery are typically stored in battery-backed or non-volatile memory. A PLC is an example of a real-time system since the output of the system controlled by the PLC depends on the input conditions.

The programmable logic controller is, then, basically a digital computer designed for use in machine control. Unlike a personal computer, it has been designed to operate in the industrial environment and is equipped with special input/output interfaces and a control programming language. The common abbreviation used in industry for these devices, PC, can be confusing because it is also the abbreviation for “personal computer.” Therefore, most manufacturers refer to their programmable controller as a PLC, which stands for “programmable logic controller.”

Initially the PLC was used to replace relay logic, but its ever-increasing range of functions means that it is found in many and more complex applications. Because the structure of a PLC is based on the same principles as those employed in computer architecture, it is capable not only of performing relay switching tasks but also of performing other applications such as timing, counting, calculating, comparing, and the processing of analog signals.

Programmable controllers offer several advantages over a conventional relay type of control. Relays have to be hardwired to perform a specific function. When the system requirements change, the relay wiring has 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. The programmable controller has eliminated much of the hardwiring associated with conventional relay control circuits (Figure 1-2). It is small and inexpensive compared to equivalent relay-based process control systems. Modern control systems still include relays, but these are rarely used for logic.

In addition to cost savings, PLCs provide many other benefits including:

- **Increased Reliability.** Once a program has been written and tested, it can be easily downloaded to other PLCs. Since all the logic is contained in the PLC’s memory, there is no chance of making a logic wiring error (Figure 1-3). The program takes the place of much of the external wiring that would normally be required for control of a process. Hardwiring, though still required to connect field devices, is less intensive. PLCs also offer the reliability associated with solid-state components.
- **More Flexibility.** It is easier to create and change a program in a PLC than to wire and rewire a circuit. With a PLC the relationships between the inputs and outputs are determined by the user program instead of the manner in which they are interconnected (Figure 1-4). Original equipment manufacturers can provide system updates by simply sending out a new program. End users can modify the program in the field, or if desired, security can be provided by hardware features such as key locks and by software passwords.
- **Lower Cost.** PLCs were originally designed to replace relay control logic, and the cost savings have been so significant that relay control is becoming



(a)



(b)

Figure 1-1 Programmable logic controller.

Source: (a–b) Courtesy GE Intelligent Platforms.



(a)



(b)

Figure 1-2 Relay- and PLC-based control panels. (a) Relay-based control panel. (b) PLC-based control panel.

Source: (a) Courtesy Mid-Illini Technical Group, Inc.; (b) Photo courtesy Ramco Electric, Ltd.

obsolete except for power applications. Generally, if an application has more than about a half-dozen control relays, it will probably be less expensive to install a PLC.

- **Communications Capability.** A PLC can communicate with other controllers or computer equipment to perform such functions as supervisory control, data gathering, monitoring devices and process parameters, and download and upload of programs (Figure 1-5).

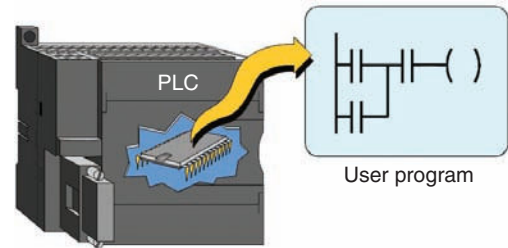


Figure 1-3 All the logic is contained in the PLC's memory.

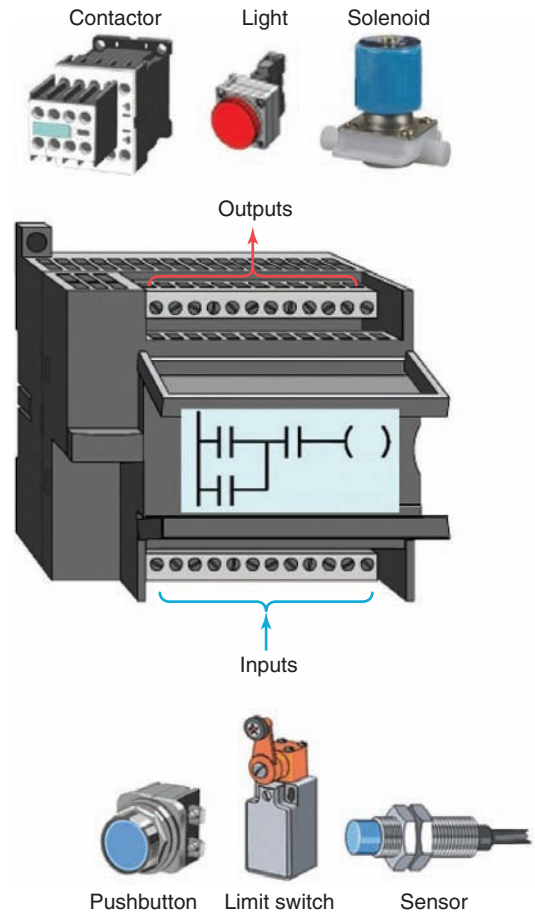


Figure 1-4 Relationships between the inputs and outputs are determined by the user program.

- **Faster Response Time.** PLCs are designed for high-speed and real-time applications (Figure 1-6). The programmable controller operates in real time, which means that an event taking place in the field will result in the execution of an operation or output. Machines that process thousands of items per second and objects that spend only a fraction of a second in front of a sensor require the PLC's quick-response capability.
- **Easier to Troubleshoot.** PLCs have resident diagnostics and override functions that allow users to



Figure 1-5 PLC communication module.
Source: Photo courtesy Automation Direct, www.automationdirect.com.



Figure 1-6 High-speed counting.
Source: Courtesy Banner Engineering Corp.

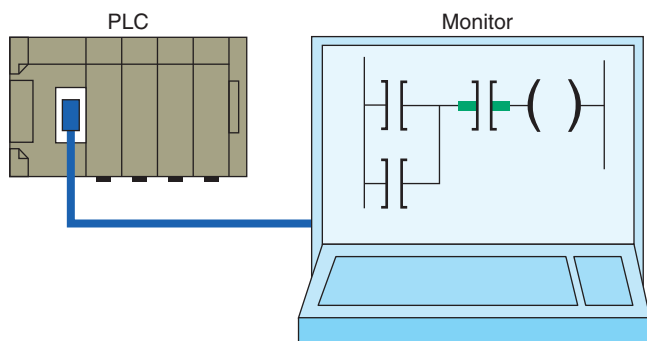


Figure 1-7 Control program can be displayed on a monitor in real time.

easily trace and correct software and hardware problems. To find and fix problems, users can display the control program on a monitor and watch it in real time as it executes (Figure 1-7).

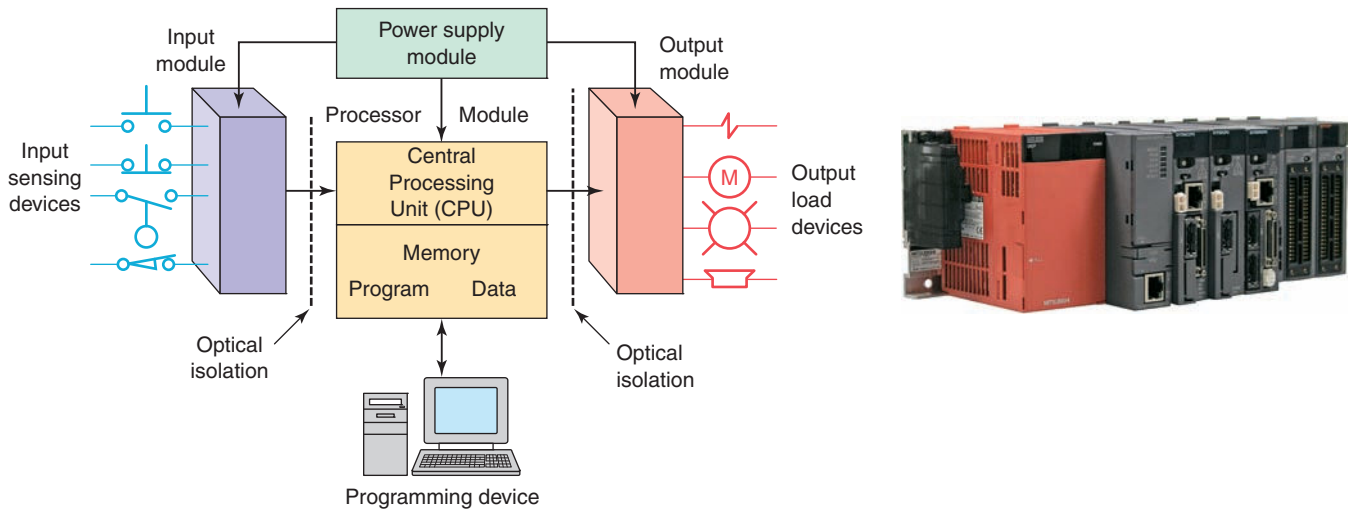
1.2 Parts of a PLC

A typical PLC can be divided into parts, as illustrated in Figure 1-8. These are the *central processing unit (CPU)*, the *input/output (I/O)* section, the *power supply*, and the *programming device*. The term *architecture* can refer to PLC hardware, to PLC software, or to a combination of both. An *open* architecture design allows the system to be connected easily to devices and programs made by other manufacturers. Open architectures use off-the-shelf components that conform to approved standards. A system with a *closed* architecture is one whose design is proprietary, making it more difficult to connect to other systems. Most PLC systems are in fact proprietary, so you must be sure that any generic hardware or software you may use is compatible with your particular PLC. Also, although the principal concepts are the same in all methods of programming, there might be slight differences in addressing, memory allocation, retrieval, and data handling for different models. Consequently, PLC programs cannot be interchanged among different PLC manufacturers.

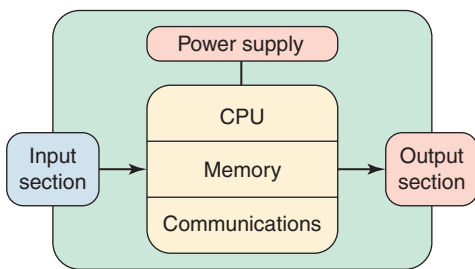
There are two ways in which I/Os (Inputs/Outputs) are incorporated into the PLC: fixed and modular. *Fixed I/O* (Figure 1-9) is typical of small PLCs that come in one package with no separate, removable units. The processor and I/O are packaged together, and the I/O terminals will have a fixed number of connections built in for inputs and outputs. The main advantage of this type of packaging is lower cost. The number of available I/O points varies and usually can be expanded by buying additional units of fixed I/O. One disadvantage of fixed I/O is its lack of flexibility; you are limited in what you can get in the quantities and types dictated by the packaging. Also, for some models, if any part in the unit fails, the whole unit has to be replaced.

Modular I/O (Figure 1-10) is divided by compartments into which separate modules can be plugged. This feature greatly increases your options and the unit's flexibility. You can choose from the modules available from the manufacturer and mix them any way you desire. The basic modular controller consists of a rack, power supply, processor module (CPU), input/output (I/O modules), and an operator interface for programming and monitoring. The modules plug into a rack. When a module is slid into the rack, it makes an electrical connection with a series of contacts called the backplane, located at the rear of the rack. The PLC processor is also connected to the backplane and can communicate with all the modules in the rack.

The *power supply* supplies DC power to other modules that plug into the rack (Figure 1-11). For large PLC systems, this power supply does not normally supply power to the field devices. With larger systems, power to field devices is



(a) Modular type



(b) Fixed type

Figure 1-8 Typical parts of a programmable logic controller.

Source: (a) Courtesy Mitsubishi Automation; (b) Image Used with Permission of Rockwell Automation, Inc.

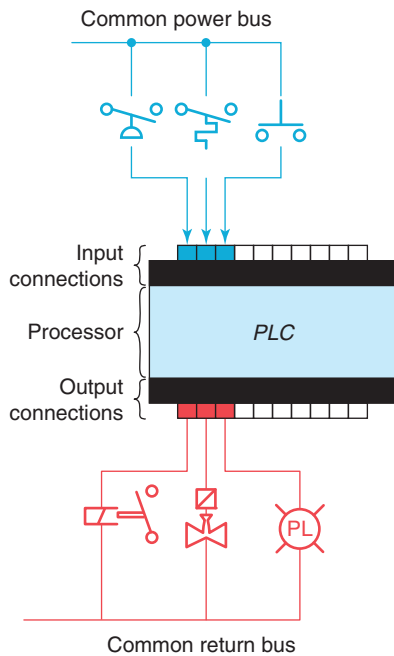


Figure 1-9 Fixed I/O configuration.

provided by external alternating current (AC) or direct current (DC) supplies. For some small micro PLC systems, the power supply may be used to power field devices.

The *processor* (CPU) is the “brain” of the PLC. A typical processor (Figure 1-12) usually consists of a microprocessor for implementing the logic and controlling the communications among the modules. The processor requires memory for storing the results of the logical operations performed by the microprocessor. Memory is also required for the program EPROM or EEPROM plus RAM.

The CPU controls all PLC activity and is designed so that the user can enter the desired program in relay ladder logic. The PLC program is executed as part of a repetitive process referred to as a scan (Figure 1-13). A typical PLC scan starts with the CPU reading the status of inputs. Then, the application program is executed. Once the program execution is completed, the CPU performs internal diagnostic and communication tasks. Next, the status of all outputs is updated. This process is repeated continuously as long as the PLC is in the run mode.

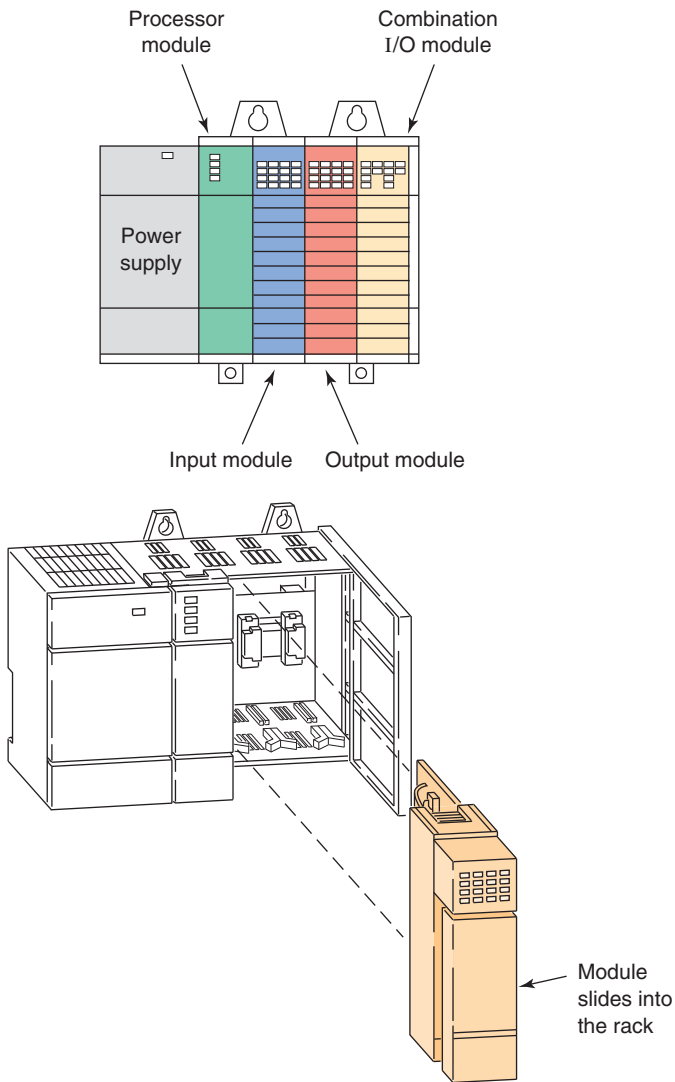


Figure 1-10 Modular I/O configuration.



Figure 1-11 The power supply supplies DC power to other modules that plug into the rack.

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



Figure 1-12 Typical PLC processor modules.
Source: Image Used with Permission of Rockwell Automation, Inc.

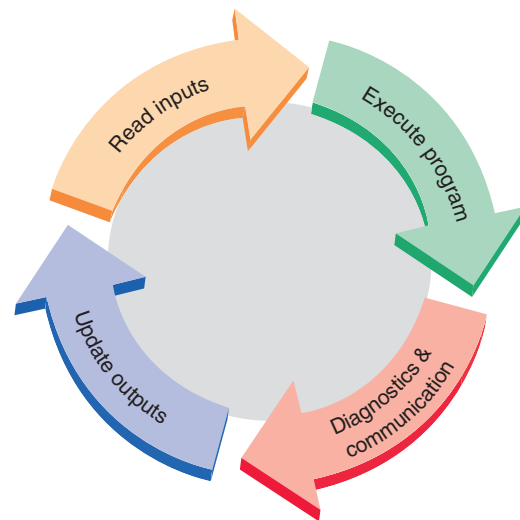


Figure 1-13 Typical PLC scan cycle.

The I/O system forms the interface by which field devices are connected to the controller (Figure 1-14). The purpose of this interface is to condition the various signals received from or sent to external field devices. Input devices such as pushbuttons, limit switches, and sensors are hardwired to the input terminals. Output devices such as small motors, motor starters, solenoid valves, and indicator lights are hardwired to the output terminals. To electrically isolate the internal components from the input and output terminals, PLCs commonly employ an optical isolator, which uses light to couple the circuits together. The external devices are also referred to as “field” or “real-world” inputs and outputs. The terms *field* or *real world* are used to distinguish actual external devices that exist and must be physically wired from the internal user program that duplicates the function of relays, timers, and counters.

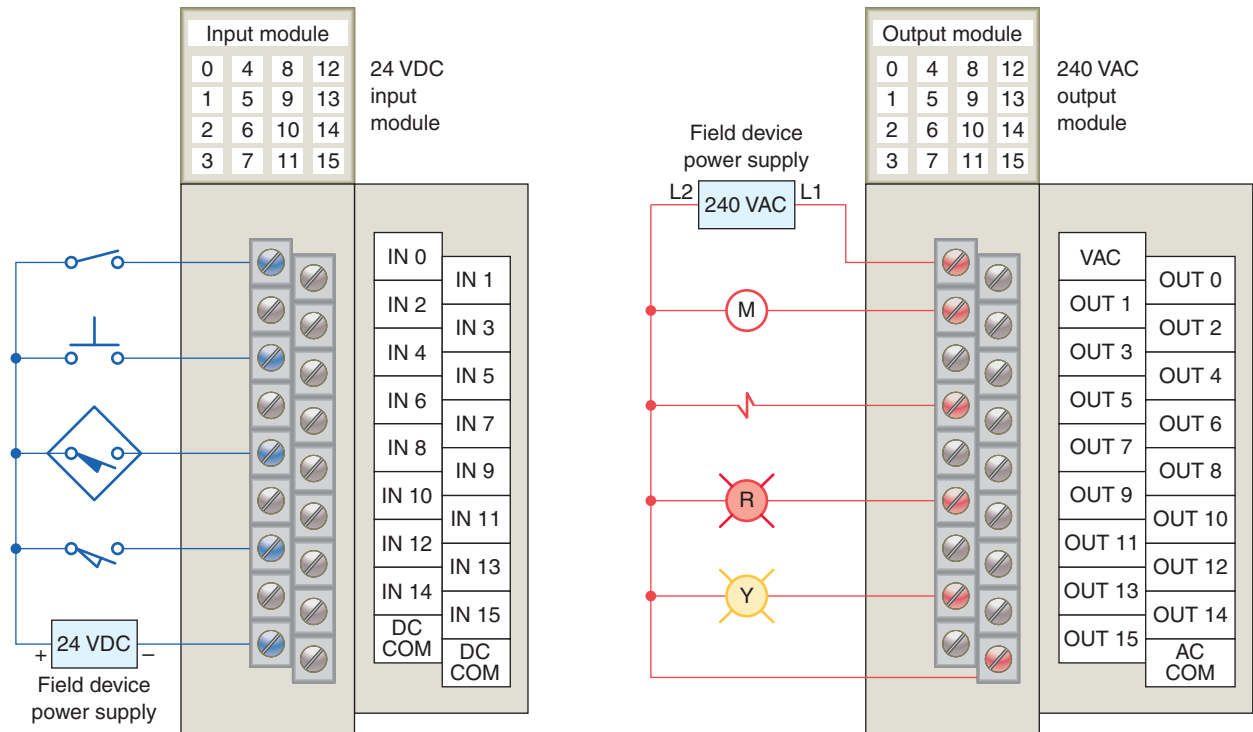


Figure 1-14 Typical PLC input/output (I/O) system connections.

A *programming device* is used to enter the desired program into the memory of the processor. The program can be entered using relay ladder logic, which is one of the most popular programming languages. Instead of words, ladder logic programming language uses graphic symbols that show their intended outcome. A program in ladder logic is similar to a schematic for a relay control circuit. It is a special language written to make it easy for people familiar with relay logic control to program the PLC. Hand-held programming devices (Figure 1-15) are sometimes used to program small PLCs because they are inexpensive and easy to use. Once plugged into the PLC, they can be used to enter and monitor programs. Both compact hand-held units and laptop computers are frequently used on the factory floor for troubleshooting equipment, modifying programs, and transferring programs to multiple machines.

A personal computer (PC) is the most commonly used programming device. Most brands of PLCs have software available so that a PC can be used as the programming device. This software allows users to create, edit, document, store, and troubleshoot ladder logic programs (Figure 1-16). The computer monitor is able to display more logic on the screen than can hand-held types, thus simplifying the interpretation of the program. The personal computer communicates with the PLC processor via a serial or parallel data communications link, or Ethernet. If

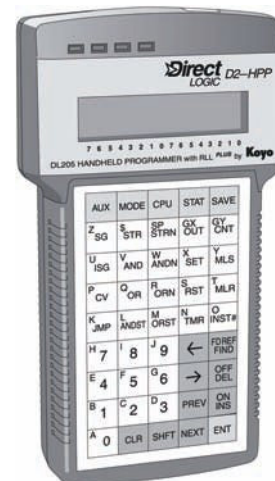


Figure 1-15 Typical hand-held programming device. Source: Photo courtesy Automation Direct, www.automationdirect.com.

the programming unit is not in use, it may be unplugged and removed. Removing the programming unit will not affect the operation of the user program.

A *program* is a user-developed series of instructions that directs the PLC to execute actions. A *programming language* provides rules for combining the instructions so that they produce the desired actions. *Relay ladder logic (RLL)* is the standard programming language used

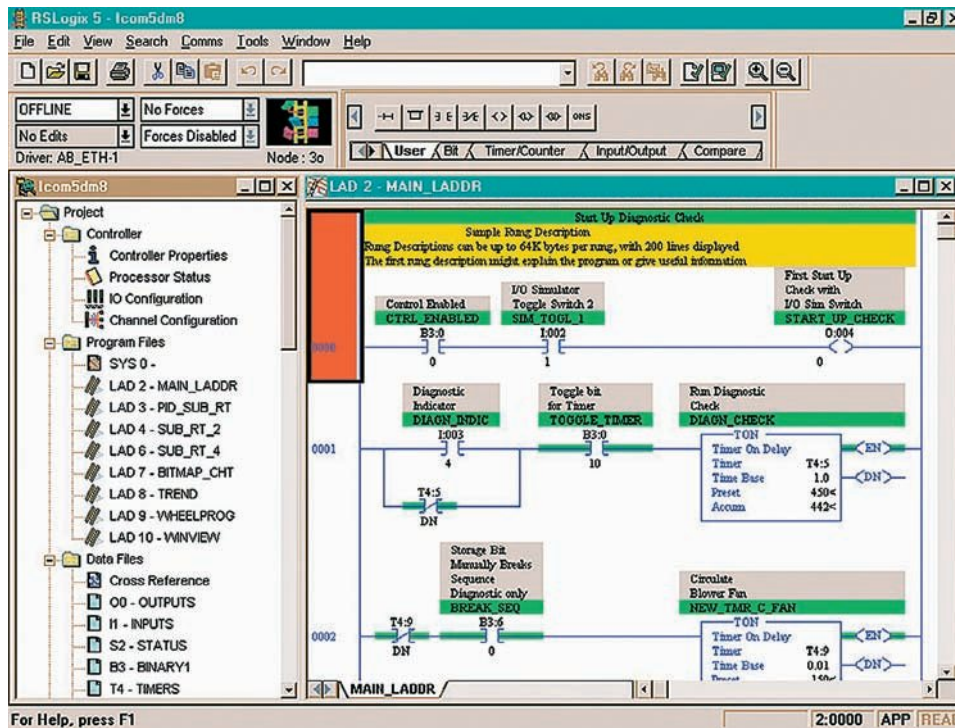


Figure 1-16 Typical PC software used to create a ladder logic program.
Source: Image Used with Permission of Rockwell Automation, Inc.

with PLCs. Its origin is based on electromechanical relay control. The relay ladder logic program graphically represents rungs of contacts, coils, and special instruction blocks. RLL was originally designed for easy use and understanding for its users and has been modified to keep up with the increasing demands of industry's control needs.

1.3 Principles of Operation

To get an idea of how a PLC operates, consider the simple process control problem illustrated in Figure 1-17. Here a mixer motor is to be used to automatically stir the liquid in a vat when the temperature and pressure reach preset values. In addition, direct manual operation of the motor is provided by means of a separate pushbutton station. The process is monitored with temperature and pressure sensor switches that close their respective contacts when conditions reach their preset values.

This control problem can be solved using the relay method for motor control shown in the relay ladder diagram of Figure 1-18. The motor starter coil (M) is energized when both the pressure and temperature switches are closed or when the manual pushbutton is pressed.

Now let's look at how a programmable logic controller might be used for this application. The same input field devices (pressure switch, temperature switch, and pushbutton) are used. These devices would be hardwired to an

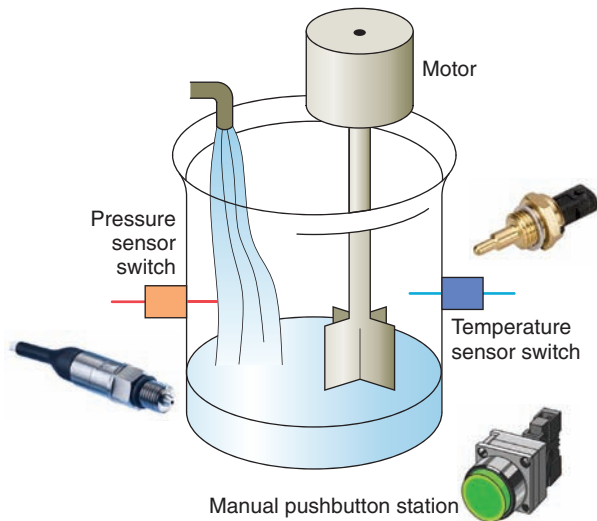


Figure 1-17 Mixer process control problem.

appropriate input module according to the manufacturer's addressing location scheme. Typical wiring connections for a 120 VAC modular configured input module is shown in Figure 1-19.

The same output field device (motor starter coil) would also be used. This device would be hardwired to an appropriate output module according to the manufacturer's addressing location scheme. Typical wiring connections for a 120 VAC modular configured output module is shown in Figure 1-20.

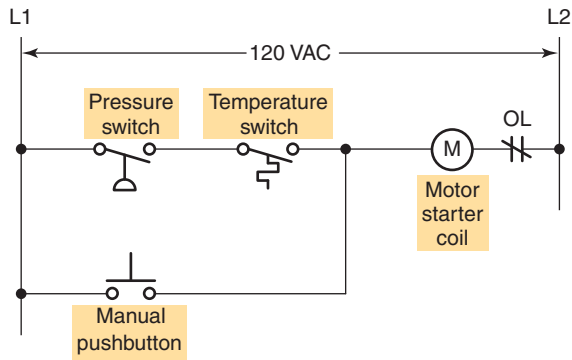


Figure 1-18 Process control relay ladder diagram.

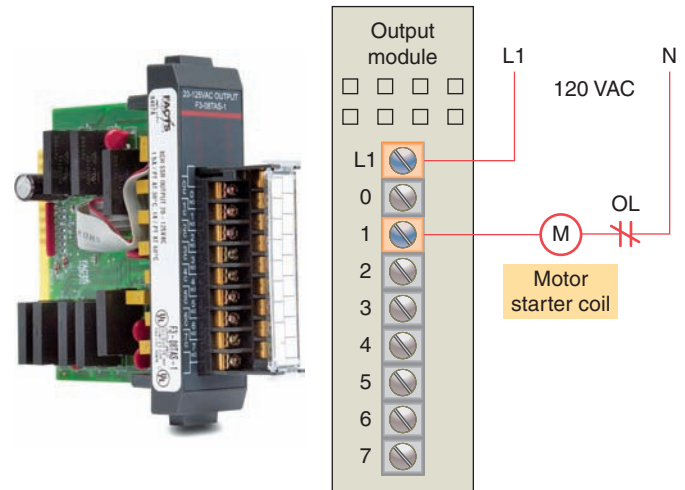


Figure 1-20 Typical wiring connections for a 120 VAC modular configured output module.

Source: Photo courtesy Automation Direct, www.automationdirect.com.

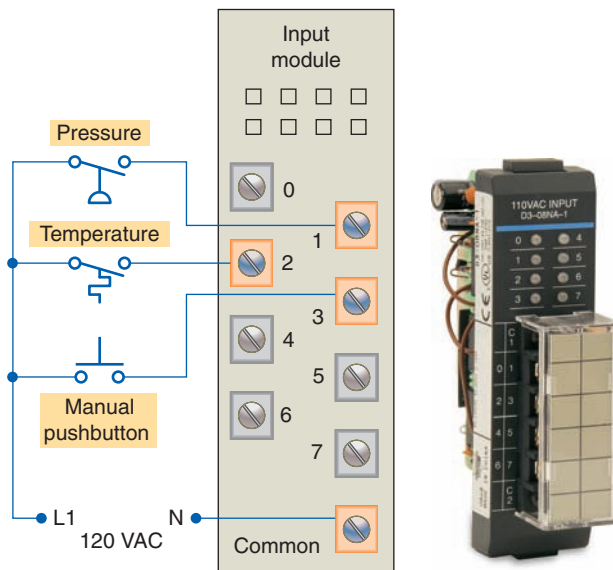


Figure 1-19 Typical wiring connections for a 120 VAC modular configured input module.

Source: Photo courtesy Automation Direct, www.automationdirect.com.

Next, the PLC ladder logic program would be constructed and entered into the memory of the CPU. A typical ladder logic program for this process is shown in Figure 1-21. The format used is similar to the layout of the hardwired relay ladder circuit. The individual symbols represent instructions, whereas the numbers represent the instruction location addresses. To program the controller, you enter these instructions one by one into the processor memory from the programming device. Each input and output device is given an address, which lets the PLC know where it is physically connected. Note that the I/O address format will differ, depending on the PLC model and manufacturer. Instructions are stored in the user program portion of the processor memory. During the program scan the controller monitors the inputs, executes the control program, and changes the output accordingly.

For the program to operate, the controller is placed in the RUN mode, or operating cycle. During each operating cycle, the controller examines the status of input devices, executes the user program, and changes outputs accordingly. Each \parallel symbol can be thought of as a set of normally open contacts. The $()$ symbol is considered to represent a coil that, when energized, will close a set of contacts. In the ladder logic program of Figure 1-21, the coil O/1 is energized when contacts I/1 and I/2 are closed or when contact I/3 is closed. Either of these conditions provides a continuous logic path from left to right across the rung that includes the coil.

A programmable logic controller operates in real time in that an event taking place in the field will result in an operation or output taking place. The RUN operation for the process control scheme can be described by the following sequence of events:

- First, the pressure switch, temperature switch, and pushbutton inputs are examined and their status is recorded in the controller's memory.
- A closed contact is recorded in memory as logic 1 and an open contact as logic 0.
- Next the ladder diagram is evaluated, with each internal contact given an OPEN or CLOSED status according to its recorded 1 or 0 state.
- When the states of the input contacts provide logic continuity from left to right across the rung, the output coil memory location is given a logic 1 value and the output module interface contacts will close.
- When there is no logic continuity of the program rung, the output coil memory location is set to logic 0

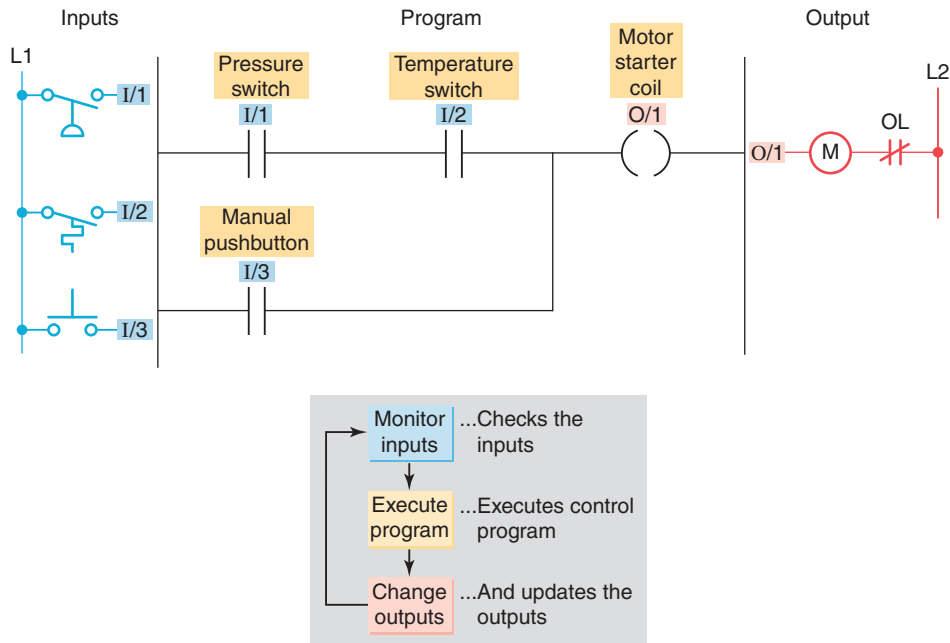


Figure 1-21 Process control PLC ladder logic program with typical addressing scheme.

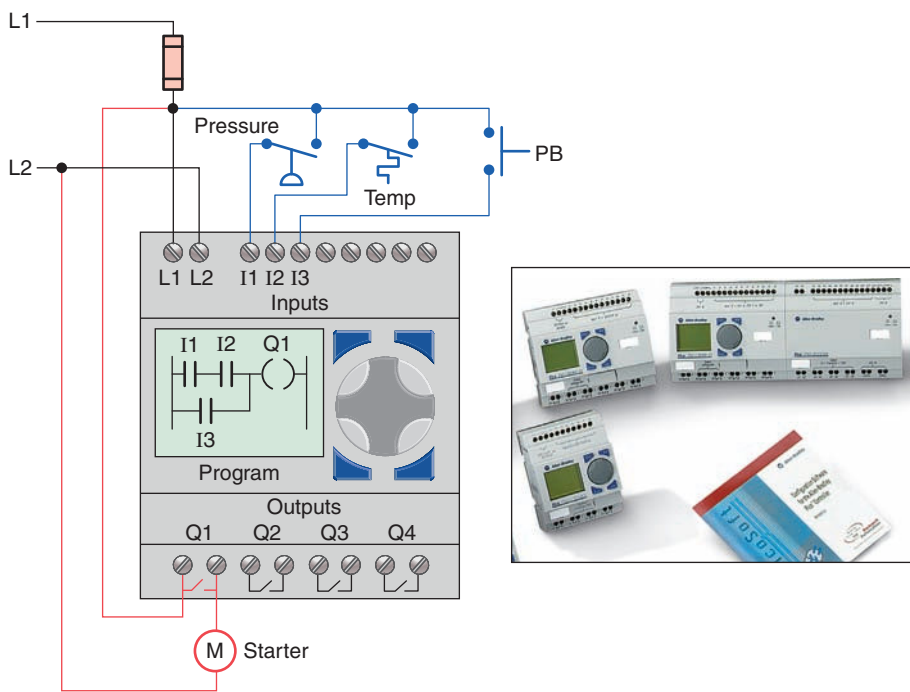


Figure 1-22 Typical wiring required to implement the process control scheme using a fixed PLC controller.

Source: Image Used with Permission of Rockwell Automation, Inc.

and the output module interface contacts will be open.

- The completion of one cycle of this sequence by the controller is called a *scan*. The scan time, the time required for one full cycle, provides a measure of the speed of response of the PLC.

- Generally, the output memory location is updated during the scan but the actual output is not updated until the end of the program scan during the I/O scan.

Figure 1-22 shows the typical wiring required to implement the process control scheme using a fixed PLC

controller. In this example the Allen-Bradley Pico controller equipped with 8 inputs and 4 outputs is used to control and monitor the process. Installation can be summarized as follows:

- Fused power lines, of the specified voltage type and level, are connected to the controller's L1 and L2 terminals.
- The pressure switch, temperature switch, and pushbutton field input devices are hardwired between L1 and controller input terminals I1, I2, and I3, respectively.
- The motor starter coil connects directly to L2 and in series with Q1 relay output contacts to L1.
- The ladder logic program is entered using the front keypad and LCD display.
- Pico programming software is also available that allows you to create as well as test your program using a personal computer.

1.4 Modifying the Operation

One of the important features of a PLC is the ease with which the program can be changed. For example, assume that the original process control circuit for the mixing operation must be modified as shown in the relay ladder diagram of Figure 1-23. The change requires that the manual pushbutton control be permitted to operate at any pressure, but not unless the specified temperature setting has been reached.

If a relay system were used, it would require some rewiring of the circuit shown in Figure 1-23 to achieve the desired change. However, if a PLC system were used, no rewiring would be necessary. The inputs and outputs are still the same. All that is required is to change the PLC ladder logic program as shown in Figure 1-24.

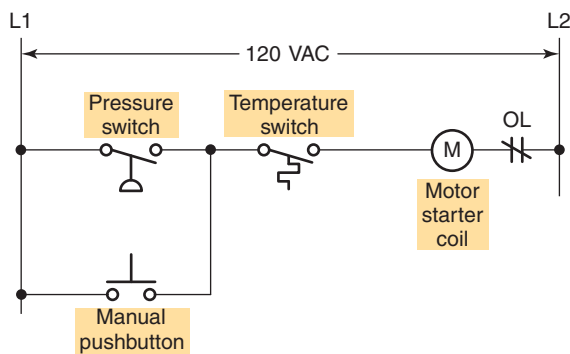


Figure 1-23 Relay ladder diagram for the modified process.

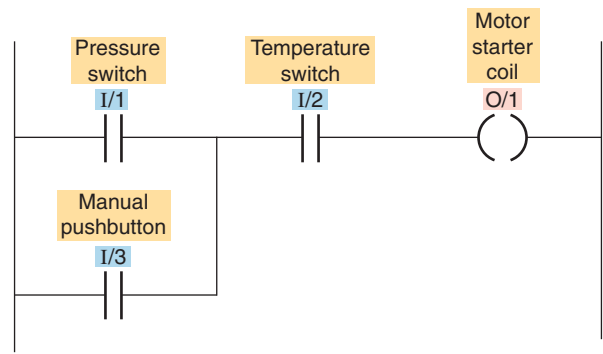


Figure 1-24 PLC ladder logic program for the modified process.

1.5 PLCs versus Computers

The architecture of a PLC is basically the same as that of a personal computer. A personal computer (PC) can be made to operate as a programmable logic controller if you provide some way for the computer to receive information from devices such as pushbuttons or switches. You also need a program to process the inputs and decide the means of turning load devices off and on.

However, some important characteristics distinguish PLCs from personal computers. First, unlike PCs, the PLC is designed to operate in the industrial environment with wide ranges of ambient temperature and humidity. A well-designed industrial PLC installation, such as that shown in Figure 1-25, is not usually affected by the electrical noise inherent in most industrial locations.

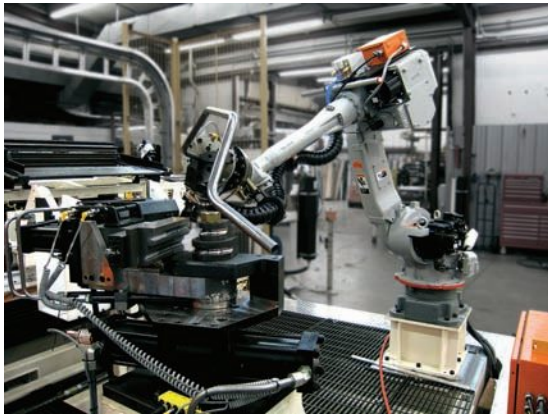
Unlike the personal computer, the PLC is programmed in relay ladder logic or other easily learned languages. The PLC comes with its program language built into its memory and has no permanently attached keyboard, CD drive, or monitor. Instead, PLCs come equipped with terminals for input and output field devices as well as communication ports.

Computers are complex computing machines capable of executing several programs or tasks simultaneously and in any order. Most PLCs, on the other hand, execute a single program in an orderly and sequential fashion from first to last instruction.

PLC control systems have been designed to be easily installed and maintained. Troubleshooting is simplified by the use of fault indicators and messaging displayed on the programmer screen. Input/output modules for connecting the field devices are easily connected and replaced.



(a)



(b)

Figure 1-25 PLC installed in an industrial environment.
Source: (a–b) Courtesy Automation IG.



Figure 1-26 PLC operator interface and monitor.
Source: Courtesy Rogers Machinery Company, Inc.



Figure 1-27 Programmable automation controller (PAC).
Source: Photo courtesy Omron Industrial Automation, www.ia.omron.com.

Software associated with a PLC but written and run on a personal computer falls into the following two broad categories:

- PLC software that allows the user to program and document gives the user the tools to write a PLC program—using ladder logic or another programming language—and document or explain the program in as much detail as is necessary.
- PLC software that allows the user to monitor and control the process is also called a *human machine interface (HMI)*. It enables the user to view a process—or a graphical representation of a process—on a monitor, determine how the system is running, trend values, and receive alarm conditions (Figure 1-26). Many operator interfaces do not use PLC software. PLCs can be integrated with HMIs but the same software does not program both devices.

Most recently automation manufacturers have responded to the increased requirements of industrial control systems

by blending the advantages of PLC-style control with that of PC-based systems. Such a device has been termed a programmable automation controller, or PAC (Figure 1-27). Programmable automation controllers combine PLC ruggedness with PC functionality. Using PACs, you can build advanced systems incorporating software capabilities such as advanced control, communication, data logging, and signal processing with rugged hardware performing logic, motion, process control, and vision.

1.6 PLC Size and Application

The criteria used in categorizing PLCs include functionality, number of inputs and outputs, cost, and physical size (Figure 1-28). Of these, the *I/O count* is the most important factor. In general, the nano is the smallest size with less than 15 I/O points. This is followed by micro types (15 to 128 I/O points), medium types (128 to 512 I/O points), and large types (over 512 I/O points).

Matching the PLC with the application is a key factor in the selection process. In general it is not advisable to



Figure 1-28 Typical range of sizes of programmable controllers.

Source: Courtesy Siemens.

buy a PLC system that is larger than current needs dictate. However, future conditions should be anticipated to ensure that the system is the proper size to fill the current and possibly future requirements of an application.

There are three major types of PLC application: single-ended, multitask, and control management. A *single-ended* or stand-alone PLC application involves one PLC controlling one process (Figure 1-29). This would be a stand-alone unit and would not be used for communicating with other computers or PLCs. The size and sophistication of the process being controlled are obvious factors in determining which PLC to select. The applications could dictate a large processor, but usually this category requires a small PLC.

A *multitask* PLC application involves one PLC controlling several processes. Adequate I/O capacity is a significant factor in this type of installation. In addition, if the PLC would be a subsystem of a larger process and would have to communicate with a central PLC or computer, provisions for a data communications network are also required.

A *control management* PLC application involves one PLC controlling several others (Figure 1-30). This kind



Figure 1-29 Single-ended PLC application.

Source: Courtesy Rogers Machinery Company, Inc.

of application requires a large PLC processor designed to communicate with other PLCs and possibly with a computer. The control management PLC supervises several PLCs by downloading programs that tell the other PLCs what has to be done. It must be capable of connection to all the PLCs so that by proper addressing it can communicate with any one it wishes to.

Memory is the part of a PLC that stores data, instructions, and the control program. Memory size is usually expressed in K values: 1 K, 6 K, 12 K, and so on. The measurement kilo, abbreviated K, normally refers to 1000 units. When dealing with computer or PLC memory, however, 1 K means 1024, because this measurement is based on the binary number system ($2^{10} \nabla 1024$). Depending on memory type, 1 K can mean 1024 bits, 1024 bytes, or 1024 words.

Although it is common for us to measure the memory capacity of PLCs in words, we need to know the number of bits in each word before memory size can be accurately compared. Modern computers usually have a word size of 16, 32, or 64 bits. For example, a PLC that uses 8-bit words has 49,152 bits of storage with a 6 K word capacity ($8 \times 6 \times 1024 = 49,152$), whereas a PLC using 32-bit words has 196,608 bits of storage with the same 6 K memory ($32 \times 6 \times 1024 = 196,608$). The amount

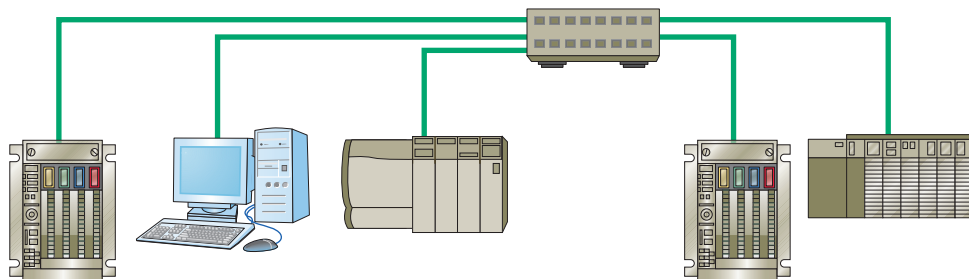


Figure 1-30 Control management PLC application.

Table 1-1 Typical PLC Instructions

Instruction	Operation
XIC (Examine ON)	Examine a bit for an ON condition
XIO (Examine OFF)	Examine a bit for an OFF condition
OTE (Output Energize)	Turn ON a bit (nonretentive)
OTL (Output Latch)	Latch a bit (retentive)
OTU (Output Unlatch)	Unlatch a bit (retentive)
TOF (Timer Off-Delay)	Turn an output ON or OFF after its rung has been OFF for a preset time interval
TON (Timer On-Delay)	Turn an output ON or OFF after its rung has been ON for a preset time interval
CTD (Count Down)	Use a software counter to count down from a specified value
CTU (Count Up)	Use a software counter to count up to a specified value

of memory required depends on the application. Factors affecting the memory size needed for a particular PLC installation include:

- Number of I/O points used
- Size of control program
- Data-collecting requirements

- Supervisory functions required
- Future expansion

The *instruction set* for a particular PLC lists the different types of instructions supported. Typically, this ranges from 15 instructions on smaller units up to 100 instructions on larger, more powerful units (see Table 1-1).



CHAPTER 1 REVIEW QUESTIONS

1. What is a programmable logic controller (PLC)?
2. Identify four tasks in addition to relay switching operations that PLCs are capable of performing.
3. List six distinct advantages that PLCs offer over conventional relay-based control systems.
4. Explain the differences between open and proprietary PLC architecture.
5. State two ways in which I/O is incorporated into the PLC.
6. Describe how the I/O modules connect to the processor in a modular-type PLC configuration.
7. Explain the main function of each of the following major components of a PLC:
 - a. Processor module (CPU)
 - b. I/O modules
 - c. Programming device
 - d. Power supply module
8. What are the two most common types of PLC programming devices?
9. Explain the terms *program* and *programming language* as they apply to a PLC.
10. What is the standard programming language used with PLCs?
11. Answer the following with reference to the process control relay ladder diagram of Figure 1-18 of this chapter:
 - a. When do the pressure switch contacts close?
 - b. When do the temperature switch contacts close?
 - c. How are the pressure and temperature switches connected with respect to each other?
 - d. Describe the two conditions under which the motor starter coil will become energized.
 - e. What is the approximate value of the voltage drop across each of the following when their contacts are open?
 - (1) Pressure switch
 - (2) Temperature switch
 - (3) Manual pushbutton
12. The programmable controller operates in real time. What does this mean?
13. Answer the following with reference to the process control PLC ladder logic diagram of Figure 1-21 of this chapter:
 - a. What do the individual symbols represent?
 - b. What do the numbers represent?
 - c. What field device is the number I/2 identified with?
 - d. What field device is the number O/1 identified with?
 - e. What two conditions will provide a continuous path from left to right across the rung?
 - f. Describe the sequence of operation of the controller for one scan of the program.
14. Compare the method by which the process control operation is changed in a relay-based system to the method used for a PLC-based system.
15. Compare the PLC and PC with regard to:
 - a. Physical hardware differences
 - b. Operating environment
 - c. Method of programming
 - d. Execution of program
16. What two categories of software written and run on PCs are used in conjunction with PLCs?
17. What is a programmable automation controller (PAC)?
18. List four criteria by which PLCs are categorized.
19. Compare the single-ended, multitask, and control management types of PLC applications.
20. What is the memory capacity, expressed in bits, for a PLC that uses 16-bit words and has an 8 K word capacity?
21. List five factors affecting the memory size needed for a particular PLC installation.
22. What does the instruction set for a particular PLC refer to?



CHAPTER 1 PROBLEMS

1. Given two single-pole switches, write a program that will turn on an output when both switch *A* and switch *B* are closed.
2. Given two single-pole switches, write a program that will turn on an output when either switch *A* or switch *B* is closed.

- Given four NO (Normally Open) pushbuttons (*A-B-C-D*), write a program that will turn a lamp on if pushbuttons *A* and *B* or *C* and *D* are closed.
- Write a program for the relay ladder diagram shown in Figure 1-31.

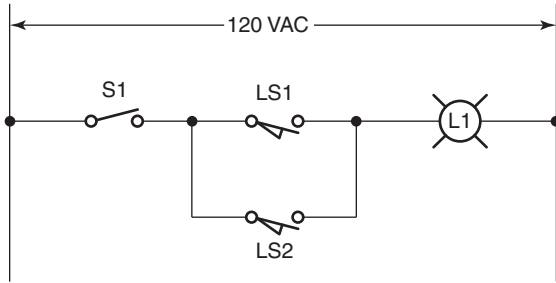


Figure 1-31 Circuit for Problem 4.

- Write a program for the relay ladder diagram shown in Figure 1-32.

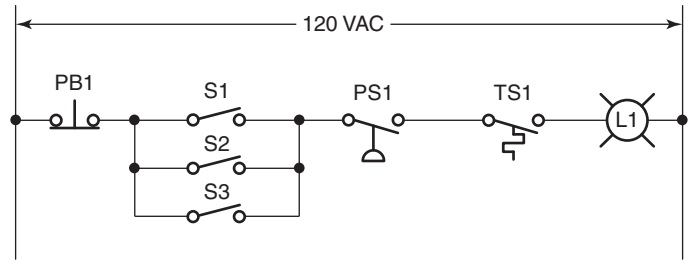


Figure 1-32 Circuit for Problem 5.

2

PLC Hardware Components



Image Used with Permission of Rockwell Automation, Inc.

Chapter Objectives

After completing this chapter, you will be able to:

- 2.1** List and describe the function of the hardware components used in PLC systems
- 2.2** Describe the basic circuitry and applications for discrete and analog I/O modules, and interpret typical I/O and CPU specifications
- 2.3** Explain I/O addressing
- 2.4** Describe the general classes and types of PLC memory devices
- 2.5** List and describe the different types of PLC peripheral support devices available

This chapter exposes you to the details of PLC hardware and modules that make up a PLC control system. The chapter's illustrations show the various subparts of a PLC as well as general connection paths. In this chapter we discuss the CPU and memory hardware components, including the various types of memory that are available, and we describe the hardware of the input/output section, including the difference between the discrete and analog types of modules.

2.1 The I/O Section

The input/output (I/O) section of a PLC is the section to which all field devices are connected and provides the interface between them and the CPU. Input/output arrangements are built into a fixed PLC while modular types use external I/O modules that plug into the PLC.

Figure 2-1 illustrates a rack-based I/O section made up of individual I/O modules. Input interface modules accept signals from the machine or process devices and convert them into signals that can be used by the controller. Output interface modules convert controller signals into external signals used to control the machine or process. A typical PLC has room for several I/O modules, allowing it to be customized for a particular application by selecting the appropriate modules. Each slot in the rack is capable of accommodating any type of I/O module.

The I/O system provides an interface between the hard-wired components in the field and the CPU. The input interface allows *status information* regarding processes to be communicated to the CPU, and thus allows the CPU to

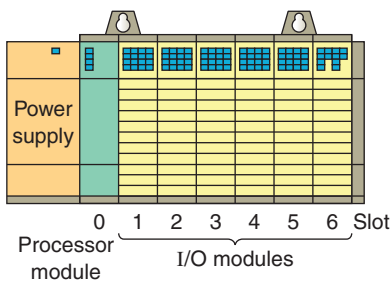


Figure 2-1 Rack-based I/O section.

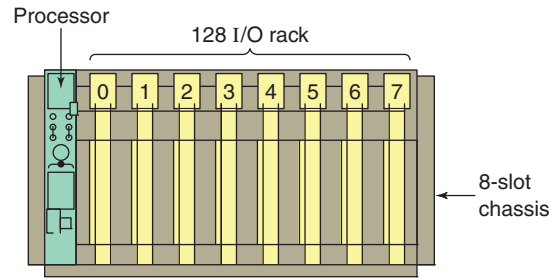


Figure 2-2 Allen-Bradley PLC chassis and rack.

communicate *operating signals* through the output interface to the process devices under its control.

Allen-Bradley controllers make a distinction between a PLC chassis and rack as illustrated in Figure 2-2. The hardware assembly that houses I/O modules, processor modules, and power supplies is referred to as the chassis. Chassis come in different sizes according to the number of slots they contain. In general, they can have 4, 8, 12, or 16 slots.

A *logical rack* is an addressable unit consisting of 128 input points and 128 output points. A rack uses 8 words in the input image table file and 8 words in the output image table file. A word in the output image table file and its corresponding word in the input image table file are called an *I/O group*. A rack can contain a maximum of 8 I/O groups (numbered from 0 through 7) for up to 128 discrete I/O. There can be more than one rack in a chassis and more than one chassis in a rack.

One benefit of a PLC system is the ability to locate the I/O modules near the field devices, as illustrated in Figure 2-3, in order to minimize the amount of wiring

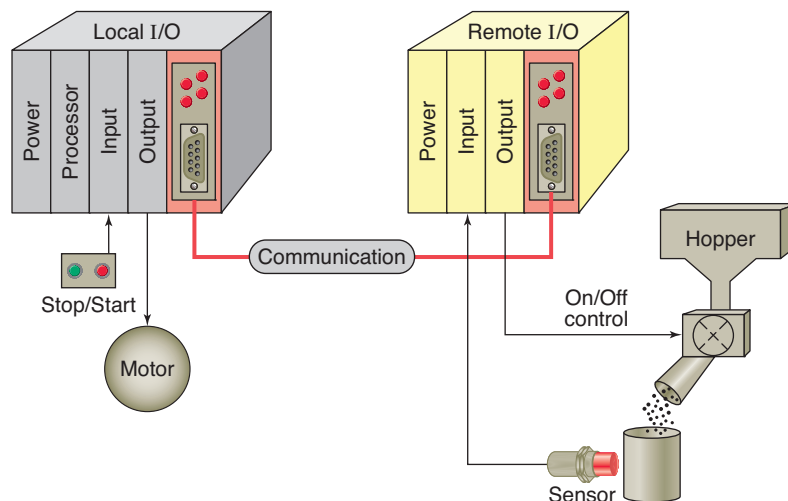


Figure 2-3 Remote I/O rack.

required. The processor receives signals from the remote input modules and sends signals back to their output modules via the communication module.

A rack is referred to as a *remote* rack when it is located away from the processor module. To communicate with the processor, the remote rack uses a special communications network. Each remote rack requires a unique station number to distinguish one from another. The remote racks are linked to the local rack through a *communications module*. Cables connect the modules with each other. If fiber optic cable is used between the CPU and I/O rack, it is possible to operate I/O points from distances greater than 20 miles with no voltage drop. Coaxial cable will allow remote I/O to be installed at distances greater than two miles. Fiber optic cable will not pick up noise caused by adjacent high power lines or equipment normally found in an industrial environment. Coaxial cable is more susceptible to this type of noise.

The PLC's memory system stores information about the status of all the inputs and outputs. To keep track of all this information, it uses a system called *addressing*. An address is a label or number that indicates where a certain piece of information is located in a PLC's memory. Just as your home address tells where you live in your city, a device's or a piece of data's address tells where information about it resides in the PLC's memory. That way, if a PLC wants to find out information about a field device, it knows to look in its corresponding address location. Examples of addressing schemes include *rack/slot-based*, versions of which are used in Allen-Bradley PLC-5 and SLC 500 controllers, *tag-based* used in Allen-Bradley ControlLogix controllers, and PC-based control used in soft PLCs.

In general, rack/slot-based addressing elements include:

Type—The type determines if an input or output is being addressed.

Slot—The slot number is the physical location of the I/O module. This may be a combination of the rack number and the slot number when using expansion racks.

Word and Bit—The word and bit are used to identify the actual terminal connection in a particular I/O module. A discrete module usually uses only one word, and each connection corresponds to a different bit that makes up the word.

With a rack/slot address system the location of a module within a rack and the terminal number of a module to which an input or output device is connected will determine the device's address. Figure 2-4 illustrates the Allen-Bradley PLC-5 controller addressing format. The following are typical examples of input and output addresses:

I1:27/17	Input, file 1, rack 2, group 7, bit 17
O0:34/07	Output, file 0, rack 3, group 4, bit 7
I1:0/0	Input, file 1, rack 0, group 0, bit 0 (Short form blank = 0)
O0:1/1	Output, file 0, rack 0, group 1, bit 1 (Short form blank = 0)

Figure 2-5 illustrates the Allen-Bradley SLC 500 controller addressing format. The address is used by the processor to identify where the device is located to monitor or control it. In addition, there is some means of connecting field wiring on the I/O module housing. Connecting the field wiring to the I/O housing allows easier disconnection and reconnection of the wiring to change modules. Lights are also added to each module to indicate the ON or OFF status of each I/O circuit. Most output modules also have blown fuse indicators. The following are typical

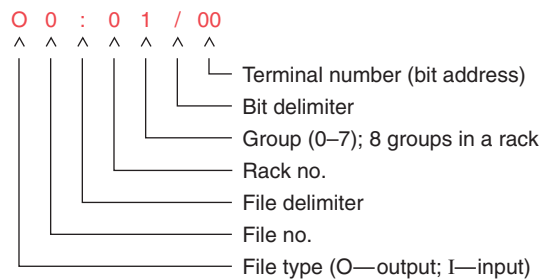


Figure 2-4 Allen-Bradley PLC-5 rack/slot-based addressing format.

Source: Image Used with Permission of Rockwell Automation, Inc.

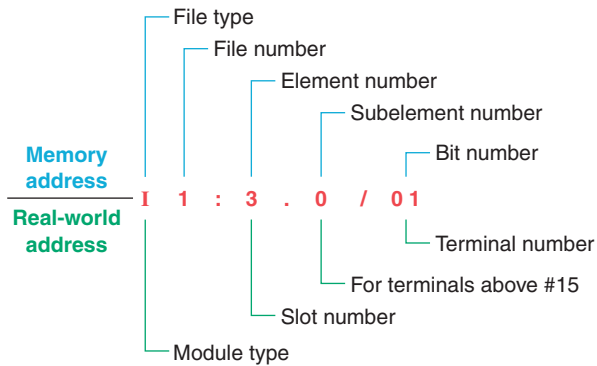


Figure 2-5 Allen-Bradley SLC 500 rack/slot-based addressing format.
Source: Image Used with Permission of Rockwell Automation, Inc.

examples of SLC 500 real-world general input and output addresses:

O:4/15	Output module in slot 4, terminal 15
I:3/8	Input module in slot 3, terminal 8
O:6.0	Output module, slot 6
I:5.0	Input module, slot 5

Every input and output device connected to a discrete I/O module is addressed to a specific *bit* in the PLC's memory. A bit is a binary digit that can be either 1 or 0. Analog I/O modules use a *word* addressing format, which allows the entire words to be addressed. The bit part of the address is usually not used; however, bits of the digital representation of the analog value can be addressed by the programmer if necessary. Figure 2-6 illustrates bit level and word level addressing as it applies to an SLC 500 controller.

Figure 2-7 illustrates the Allen-Bradley ControlLogix tag-based addressing format. With Logix5000 controllers, you use a tag (alphanumeric name) to address data (variables). Instead of a fixed numeric format the tag name itself identifies the data. The field devices are assigned tag names that are referenced when the PLC ladder logic program is developed.

PC-based control runs on personal or industrial hardened computers. Also known as soft PLCs, they simulate the functions of a PLC on a PC, allowing open architecture systems to replace proprietary PLCs. This implementation uses an input/output card (Figure 2-8) in conjunction with the PC as an interface for the field devices.

Combination I/O modules can have both input and output connections in the same physical module as illustrated

in Figure 2-9. A module is made up of a printed circuit board and a terminal assembly. The printed circuit board contains the electronic circuitry used to interface the circuit of the processor with that of the input or output device. Modules are designed to plug into a slot or connector in the I/O rack or directly into the processor. The terminal assembly, which is attached to the front edge of the printed circuit board, is used for making field-wiring connections. Modules contain terminals for each input and output connection, status lights for each of the inputs and outputs, and connections to the power supply used to power the inputs and outputs. Terminal and status light arrangements vary with different manufacturers.

Most PLC modules have plug-in wiring terminal strips. The terminal block is plugged into the actual module as illustrated in Figure 2-10. If there is a problem with a module, the entire strip is removed, a new module is inserted, and the terminal strip is plugged into the new module. Unless otherwise specified, never install or remove I/O modules or terminal blocks while the PLC is powered. A module inserted into the wrong slot could be damaged by improper voltages connected through the wiring arm. Most faceplates and I/O modules are keyed to prevent putting the wrong faceplate on the wrong module. In other words, an output module cannot be placed in the slot where an input module was originally located.

Input and output modules can be placed anywhere in a rack, but they are normally grouped together for ease of wiring. I/O modules can be 8, 16, 32, or 64 point cards (Figure 2-11). The number refers to the number of inputs or outputs available. The standard I/O module has eight inputs or outputs. A *high-density* module may have up to 64 inputs or outputs. The advantage with the high-density module is that it is possible to install up to 64 inputs or outputs in one slot for greater space savings. The only disadvantage is that the high-density output modules cannot handle as much current per output.

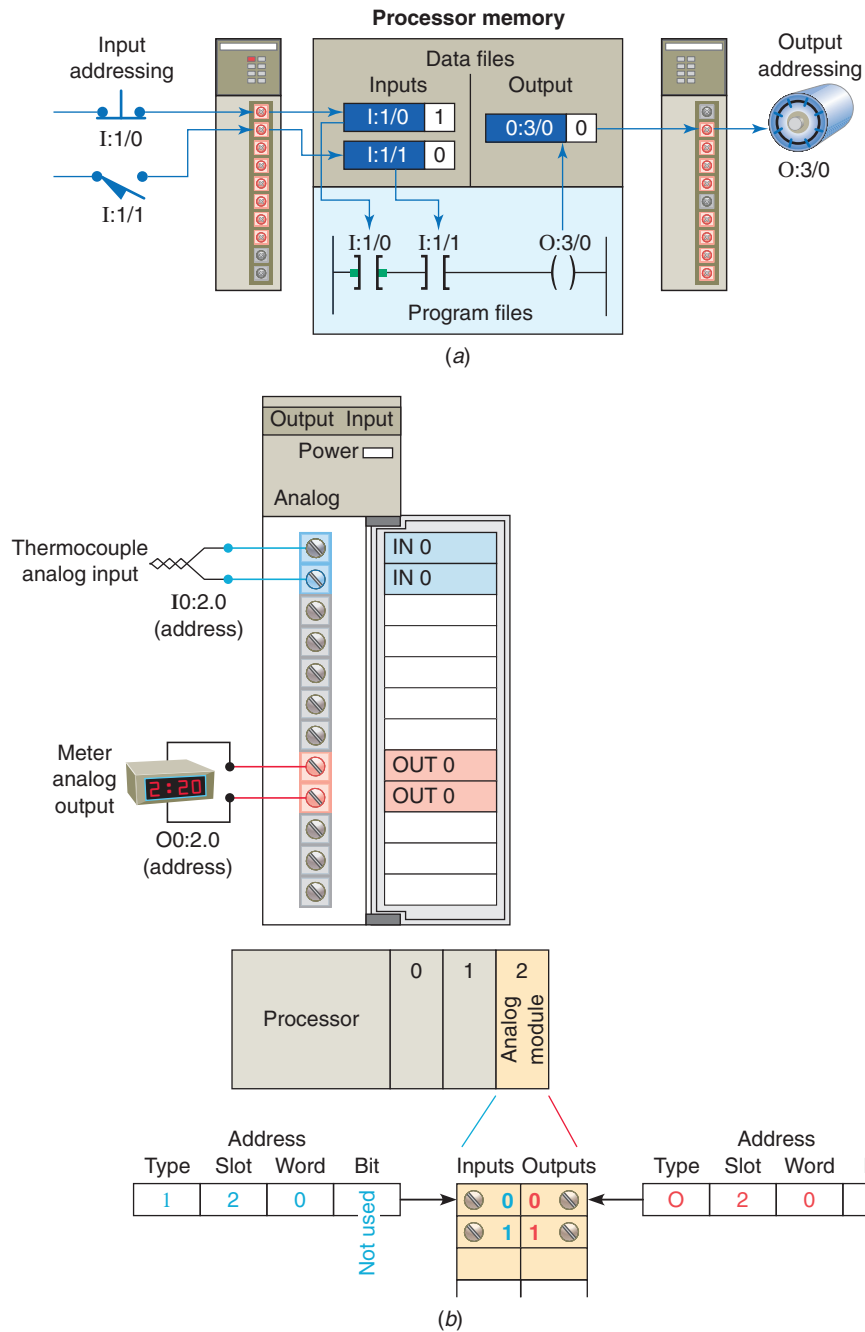


Figure 2-6 SLC 500 bit level and word level addressing. (a) Bit level addressing. (b) Word level addressing.

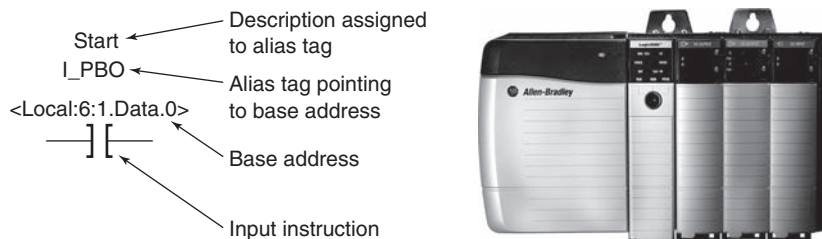


Figure 2-7 Allen-Bradley ControlLogix tag-based addressing format.

Source: Image Used with Permission of Rockwell Automation, Inc.



Figure 2-8 Typical PC interface card.
Source: Photo © Beckhoff Automation GmbH.

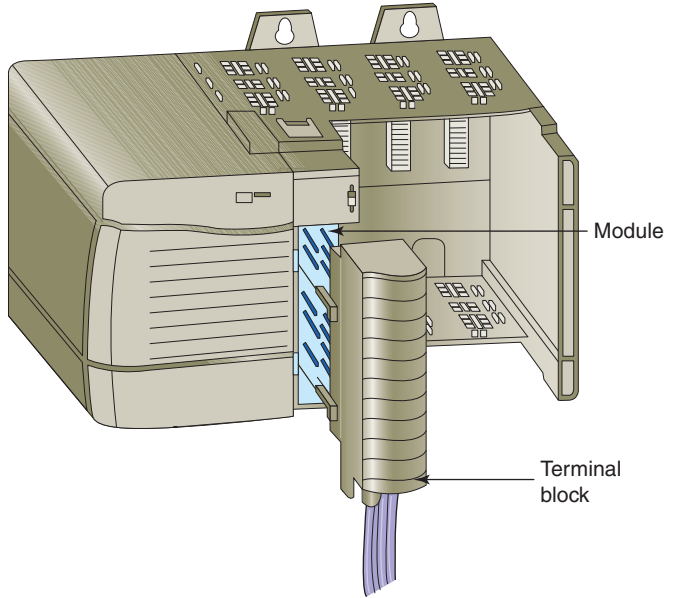


Figure 2-10 Plug-in terminal block.

2.2 Discrete I/O Modules

The most common type of I/O interface module is the *discrete* type (Figure 2-12). This type of interface connects field input devices of the ON/OFF nature such as selector switches, pushbuttons, and limit switches. Likewise, output control is limited to devices such as lights, relays, solenoids, and motor starters that require simple ON/OFF

switching. The classification of discrete I/O covers *bit-oriented* inputs and outputs. In this type of input or output, each bit represents a complete information element in itself and provides the status of some external contact or advises of the presence or absence of power in a process circuit.

Each discrete I/O module is powered by some *field-supplied* voltage source. Since these voltages can be of different magnitude or type, I/O modules are available at various AC and DC voltage ratings, as listed in Table 2-1.

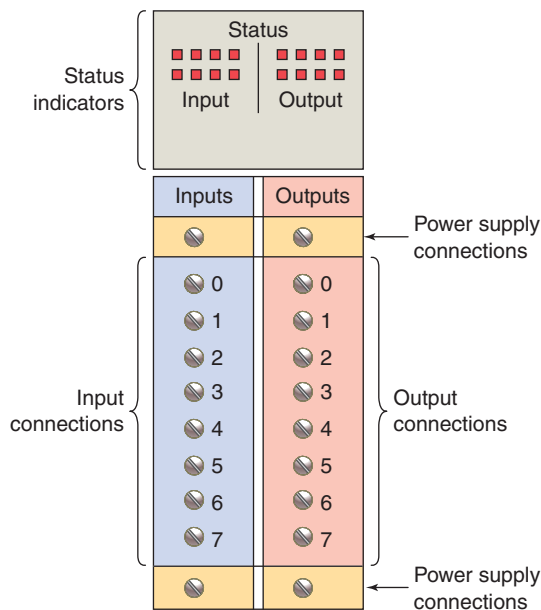


Figure 2-9 Typical combination I/O module.
Source: Image Used with Permission of Rockwell Automation, Inc.





Figure 2-11 16, 32, and 64 point I/O modules.
 Source: (a//) Photos courtesy Omron Industrial Automation, www.ia.omron.com.

The modules themselves receive their voltage and current for proper operation from the backplane of the rack enclosure into which they are inserted, as illustrated in Figure 2-13. Backplane power is provided by the PLC module power supply and is used to power the electronics that reside on the I/O module circuit board. The relatively higher

Table 2-1 Common Ratings for Discrete I/O Interface Modules

Input Interfaces	Output Interfaces
12 V AC/DC /24 V AC/DC	12–48 V AC
48 V AC/DC	120 V AC
120 V AC/DC	230 V AC
230 V AC/DC	120 V DC
5 V DC (TTL level)	230 V DC
	5 V DC (TTL level)
	24 V DC

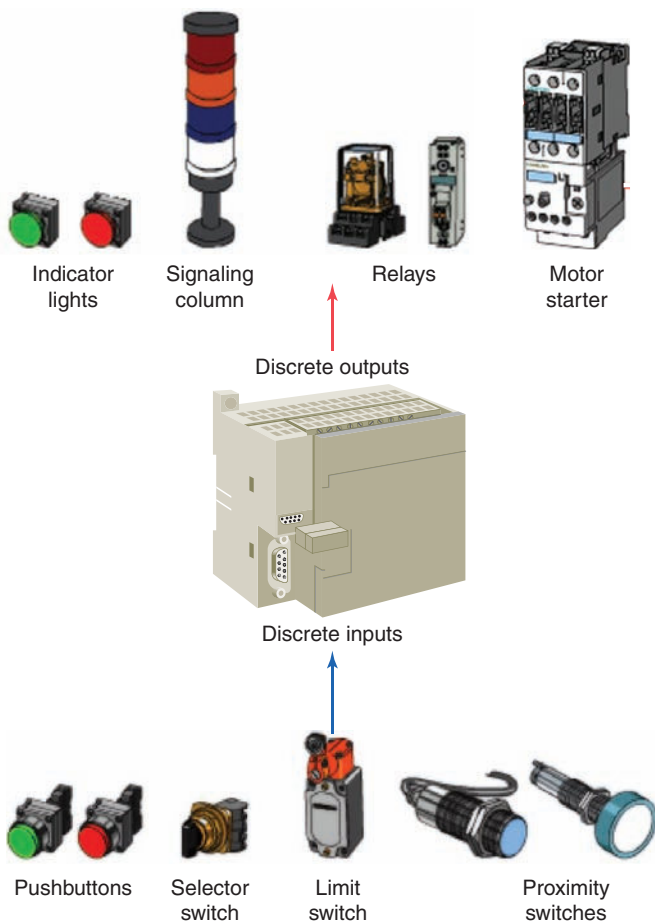


Figure 2-12 Discrete input and output devices.

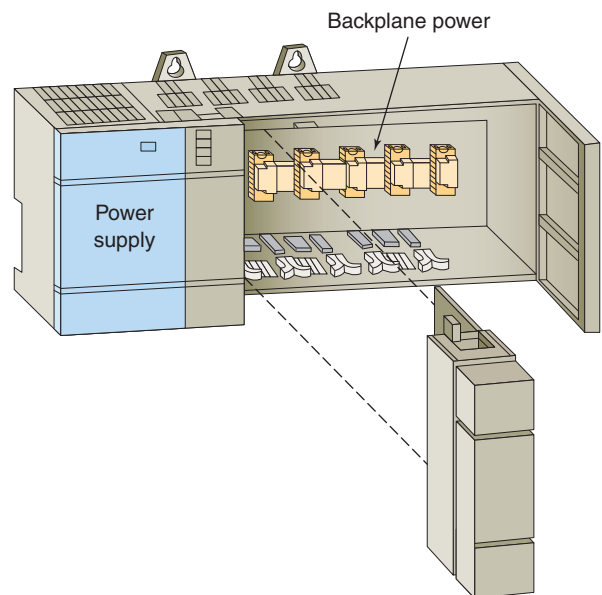


Figure 2-13 Modules receive their voltage and current from the backplane.

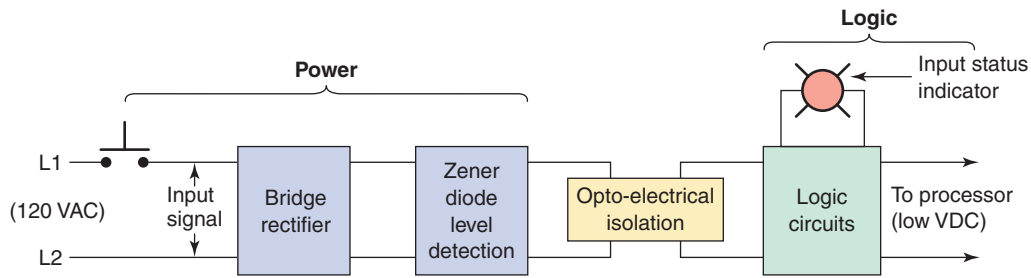


Figure 2-14 Discrete AC input module block diagram.

currents required by the loads of an output module are normally provided by user-supplied power. Module power supplies typically may be rated for 3 A, 4 A, 12 A, or 16 A depending on the type and number of modules used.

Figure 2-14 shows the block diagrams for one input of a typical alternating current (AC) *discrete input module*. The input circuit is composed of two basic sections: the *power* section and the *logic* section. An optical isolator is used to provide electrical isolation between the field wiring and the PLC backplane internal circuitry. The input LED turns on or off, indicating the status of the input device. Logic circuits process the digital signal to the processor. Internal PLC control circuitry typically operates at 5 VDC or less volts.

A simplified diagram for a single input of a discrete AC input module is shown in Figure 2-15. The operation of the circuit can be summarized as follows:

- The input noise filter consisting of the capacitor and resistors R1 and R2 removes false signals that are due to contact bounce or electrical interference.
- When the pushbutton is closed, 120 VAC is applied to the bridge rectifier input.
- This results in a low-level DC output voltage that is applied across the LED of the optical isolator.

- The zener diode (Z_D) voltage rating sets the minimum threshold level of voltage that can be detected.
- When light from the LED strikes the phototransistor, it switches into conduction and the status of the pushbutton is communicated in logic to the processor.
- The optical isolator not only separates the higher AC input voltage from the logic circuits but also prevents damage to the processor due to line voltage transients. In addition, this isolation also helps reduce the effects of electrical noise, common in the industrial environment, which can cause erratic operation of the processor.
- For fault diagnosis, an input state LED indicator is on when the input pushbutton is closed. This indicator may be wired on either side of the optical isolator.
- An AC/DC type of input module is used for both AC and DC inputs as the input polarity does not matter.
- A PLC input module will have either all inputs isolated from each other with no common input connections or groups of inputs that share a common connection.

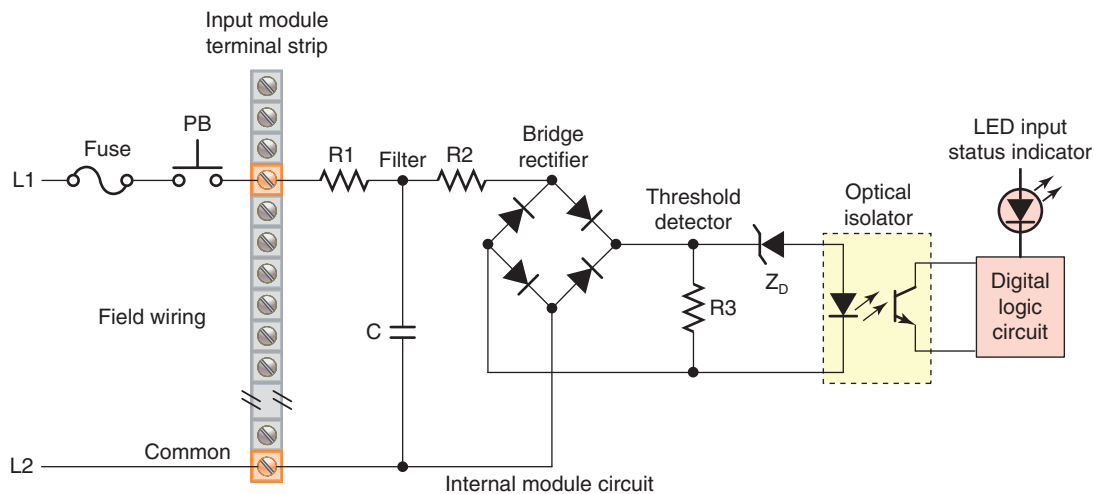


Figure 2-15 Simplified diagram for a single input of a discrete AC input module.

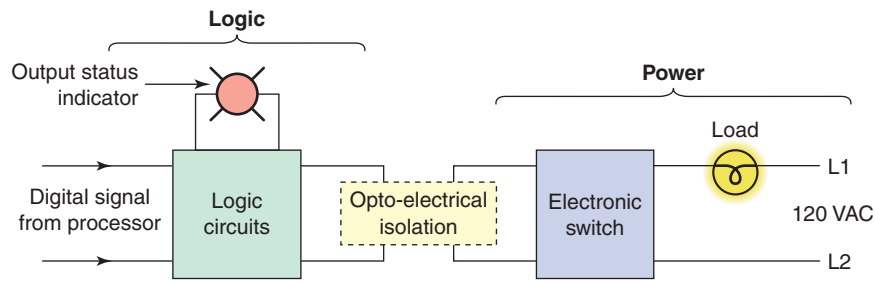


Figure 2-16 Discrete AC output module block diagram.

Discrete input modules perform four tasks in the PLC control system. They:

- Sense when a signal is received from a field device.
- Convert the input signal to the correct voltage level for the particular PLC.
- Isolate the PLC from fluctuations in the input signal's voltage or current.
- Send a signal to the processor indicating which sensor originated the signal.

Figure 2-16 shows the block diagram for one output of a typical discrete output module. Like the input module, it is composed of two basic sections: the power section and the logic section, coupled by an isolation circuit. The output interface can be thought of as an electronic switch that turns the output load device on and off. Logic circuits determine the output status. An output LED indicates the status of the output signal.

A simplified diagram for a single output of a discrete AC output module is shown in Figure 2-17. The operation of the circuit can be summarized as follows:

- As part of its normal operation, the digital logic circuits of the processor sets the output status according to the program.

- When the processor calls for an output load to be energized, a voltage is applied across the LED of the opto-isolator.
- The LED then emits light, which switches the phototransistor into conduction.
- This in turn triggers the triac AC semiconductor switch into conduction allowing current to flow to the output load.
- Since the triac conducts in either direction, the output to the load is alternating current.
- The triac, rather than having ON and OFF status, actually has LOW and HIGH resistance levels, respectively. In its OFF state (HIGH resistance), a small leakage current of a few milliamperes still flows through the triac.
- As with input circuits, the output interface is usually provided with LEDs that indicate the status of each output.
- Fuses are normally required for the output module, and they are provided on a per circuit basis, thus allowing for each circuit to be protected and operated separately. Some modules also provide visual indicators for fuse status.
- The triac cannot be used to switch a DC load.

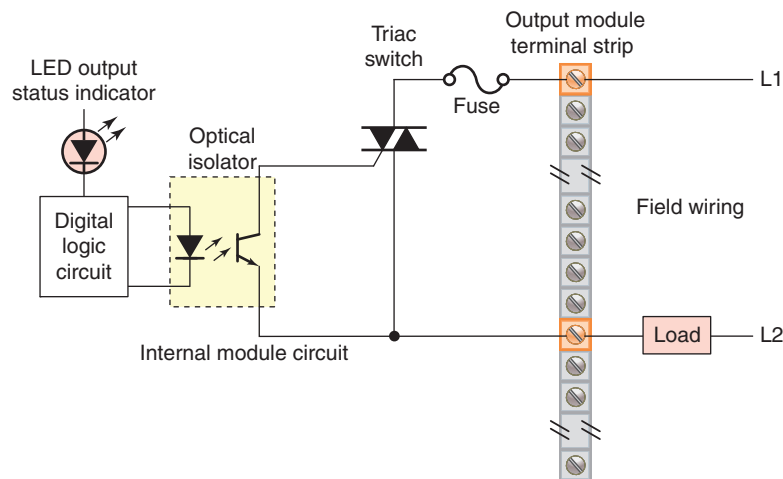


Figure 2-17 Simplified diagram for a single output of a discrete AC output module.

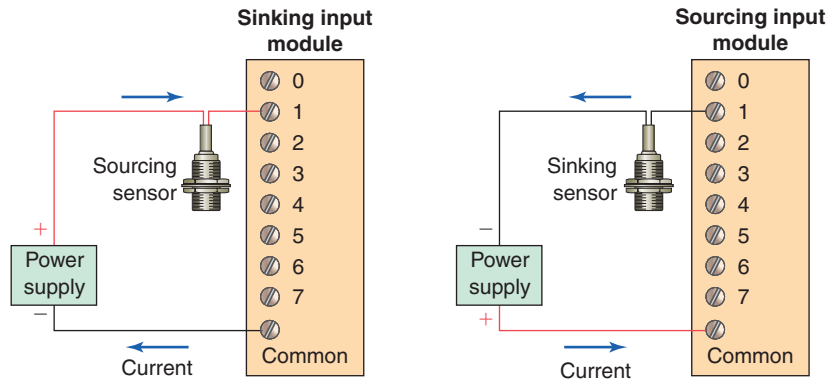


Figure 2-20 Sinking and sourcing inputs.

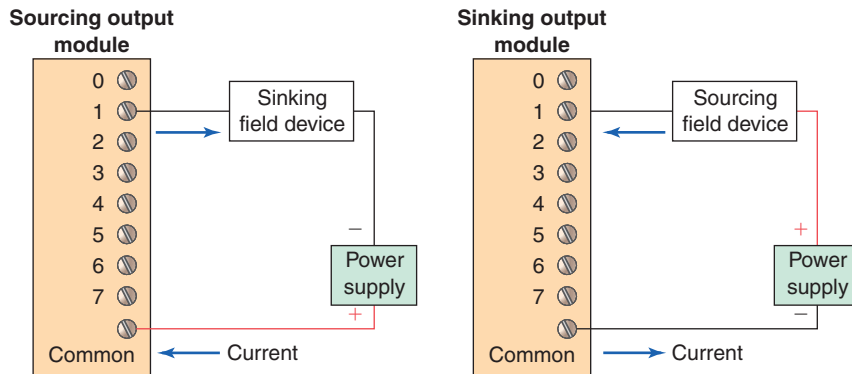


Figure 2-21 Sinking and sourcing outputs.

solid-state circuitry that needs a DC signal voltage to function. Field devices connected to the positive (+) side of the field power supply are classified as sourcing field devices. Conversely, field devices connected to the negative (-) side or DC common of the field power supply are sinking field devices.

2.3 Analog I/O Modules

Earlier PLCs were limited to discrete or digital I/O interfaces, which allowed only on/off-type devices to be connected. This limitation meant that the PLC could have only partial control of many process applications. Today, however, a complete range of both discrete and analog interfaces are available that will allow controllers to be applied to practically any type of control process.

Discrete devices are inputs and outputs that have only two states: on and off. In comparison, analog devices represent physical quantities that can have an infinite number of values. Typical analog inputs and outputs vary from 0 to 20 milliamps, 4 to 20 milliamps, or 0 to 10 volts. Figure 2-22 illustrates how PLC analog input and output modules are used in measuring and displaying the level of fluid in a tank. The analog input interface module contains the circuitry necessary to accept an analog voltage or

current signal from the level transmitter field device. This input is converted from an analog to a digital value for use by the processor. The circuitry of the analog output module accepts the digital value from the processor and converts it back to an analog signal that drives the field tank level meter.

Analog input modules normally have multiple input channels that allow 4, 8, or 16 devices to be interface to the PLC. The two basic types of analog input modules are *voltage* sensing and *current* sensing. Analog sensors measure a varying physical quantity over a specific range and generate a corresponding voltage or current signal. Common physical quantities measured by a PLC analog module include temperature, speed, level, flow, weight, pressure, and position. For example, a sensor may

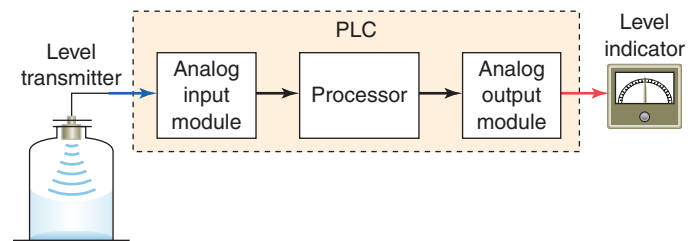


Figure 2-22 Analog input and output to a PLC.

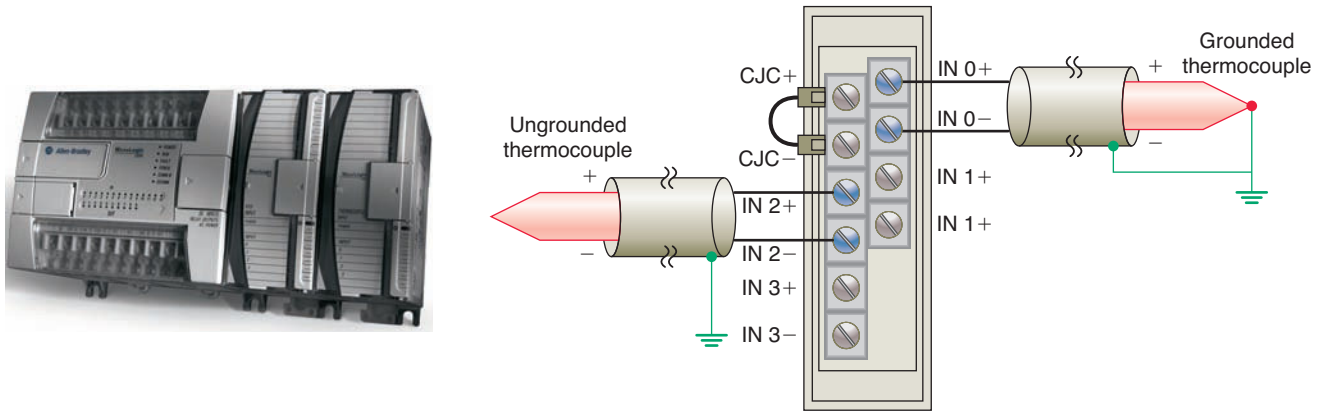


Figure 2-23 MicroLogix 4-channel analog thermocouple input module.
Source: Image Used with Permission of Rockwell Automation, Inc.

measure temperature over a range of 0 to 500°C, and output a corresponding voltage signal that varies between 0 and 50 mV.

Figure 2-23 illustrates an example of a voltage sensing input analog module used to measure temperature. The connection diagram applies to an Allen-Bradley Micro-Logic 4-channel analog thermocouple input module. A varying DC voltage in the low millivolt range, proportional to the temperature being monitored, is produced by the thermocouple. This voltage is amplified and digitized by the analog input module and then sent to the processor on command from a program instruction. Because of the low voltage level of the input signal, a twisted shielded pair cable is used in wiring the circuit to reduce unwanted electrical noise signals that can be induced in the conductors from other wiring. When using an ungrounded thermocouple, the shield must be connected to ground at the module end. To obtain accurate readings from each of the channels, the temperature between the thermocouple wire and the input channel must be compensated for. A cold junction compensating (CJC) thermistor is integrated in the terminal block for this purpose.

The transition of an analog signal to digital values is accomplished by an analog-to-digital (A/D) converter, the main element of the analog input module. Analog voltage input modules are available in two types: unipolar and bipolar. *Unipolar* modules can accept an input signal that varies in the positive direction only. For example, if the field device outputs 0 V to +10 V, then the unipolar modules would be used. Bipolar signals swing between a maximum negative value and a maximum positive value. For example, if the field device outputs -10 V to +10 V a bipolar module would be used. The *resolution* of an analog input channel refers to the smallest change in input signal value that can be sensed and is based on the number of bits used in the digital representation. Analog input

modules must produce a range of digital values between a maximum and minimum value to represent the analog signal over its entire span. Typical specifications are as follows:

Span of analog input	Bipolar	10 V	-10 to +10 V
		5 V	-5 to +5 V
	Unipolar	10 V	0 to +10 V
		5 V	0 to +5 V
Resolution			0.3 mV

When connecting voltage sensing inputs, close adherence to specified requirements regarding wire length is important to minimize signal degrading and the effects of electromagnetic noise interference induced along the connecting conductors. Current input signals, which are not as sensitive to noise as voltage signals, are typically not distance limited. Current sensing input modules typically accept analog data over the range of 4 mA to 20 mA, but can accommodate signal ranges of -20 mA to +20 mA. The loop power may be supplied by the sensor or may be provided by the analog output module as illustrated in Figure 2-24. Shielded twisted pair cable is normally recommended for connecting any type analog input signal.

The *analog output interface module* receives from the processor digital data, which are converted into a proportional voltage or current to control an analog field device. The transition of a digital signal to analog values is accomplished by a digital-to-analog (D/A) converter, the main

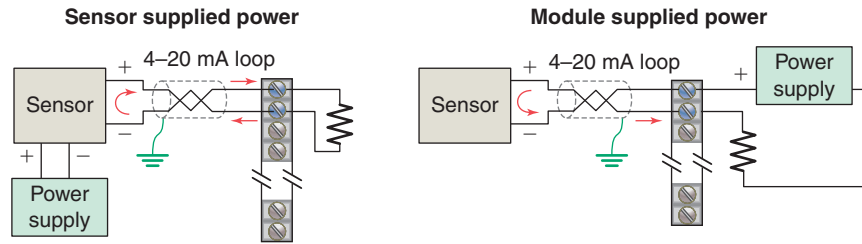


Figure 2-24 Sensor and analog module supplied power.

element of the analog output module. An analog output signal is a continuous and changing signal that is varied under the control of the PLC program. Common devices controlled by a PLC analog output module include instruments, control valves, chart recorder, electronic drives, and other types of control devices that respond to analog signals.

Figure 2-25 illustrates the use of analog I/O modules in a typical PLC control system. In this application the PLC controls the amount of fluid placed in a holding tank by adjusting the percentage of the valve opening. The analog output from the PLC is used to control the flow by controlling the amount of the valve opening. The valve is initially open 100 percent. As the fluid level in the tank approaches the preset point, the processor modifies the output, which adjusts the valve to maintain a set point.

2.4 Special I/O Modules

Many different types of I/O modules have been developed to meet special needs. These include:

HIGH-SPEED COUNTER MODULE

The high-speed counter module is used to provide an interface for applications requiring counter speeds that surpass the capability of the PLC ladder program. High-speed counter modules are used to count pulses (Figure 2-26) from sensors, encoders, and switches that operate at very high speeds. They have the electronics needed to count independently of the processor. A typical count rate

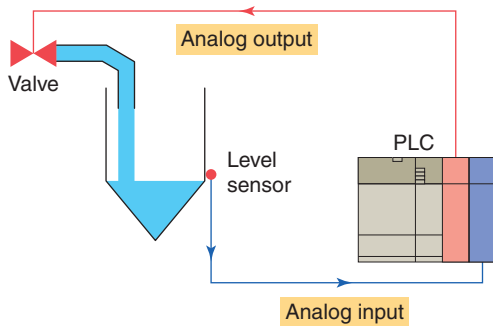


Figure 2-25 Typical analog I/O control system.

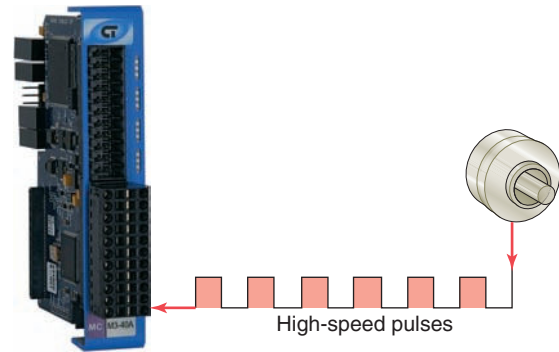


Figure 2-26 High-speed counter module.

Source: Courtesy Control Technology Corporation.

available is 0 to 100 kHz, which means the module would be able to count 100,000 pulses per second.

THUMBWHEEL MODULE

The thumbwheel module allows the use of thumbwheel switches (Figure 2-27) for feeding information to the PLC to be used in the control program.

TTL MODULE

The TTL module (Figure 2-28) allows the transmitting and receiving of TTL (Transistor-Transistor-Logic) signals. This module allows devices that produce TTL-level signals to communicate with the PLC's processor.



Figure 2-27 Thumbwheel switch.

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



Figure 2-28 TTL module.
Source: Courtesy Control Technology, Inc.

ENCODER-COUNTER MODULE

An encoder-counter module allows the user to read the signal from an encoder (Figure 2-29) on a real-time basis and stores this information so it can be read later by the processor.

BASIC OR ASCII MODULE

The BASIC or ASCII module (Figure 2-30) runs user-written BASIC and C programs. These programs are independent of the PLC processor and provide an easy, fast interface between remote foreign devices and the PLC



Figure 2-29 Encoder.
Source: Photo courtesy of Allied Motion Technologies, Inc.



Figure 2-30 BASIC module.
Source: Image Used with Permission of Rockwell Automation, Inc.

processor. Typical applications include interfaces to bar code readers, robots, printers, and displays.

STEPPER-MOTOR MODULE

The stepper-motor module provides pulse trains to a stepper-motor translator, which enables control of a stepper motor (Figure 2-31). The commands for the module are determined by the control program in the PLC.

BCD-OUTPUT MODULE

The BCD-output module enables a PLC to operate devices that require BCD-coded signals such as seven-segment displays (Figure 2-32).

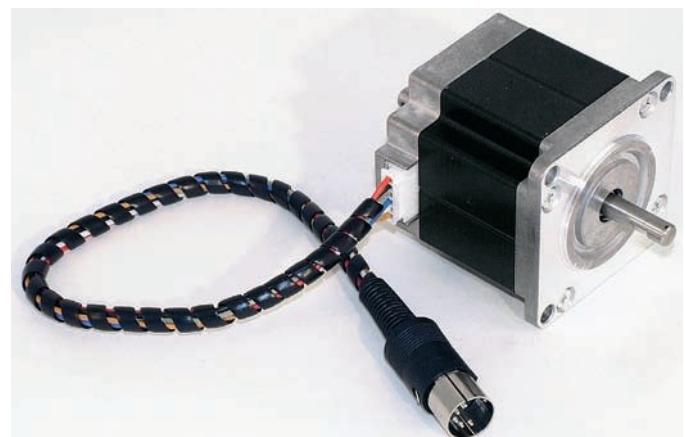


Figure 2-31 Stepper-motor.
Source: Courtesy Sherline Products.



Figure 2-32 Seven-segment display.
Source: Courtesy Red Lion Controls.

Some special modules are referred to as *intelligent I/O* because they have their own microprocessors on board that can function in parallel with the PLC. These include:

PID MODULE

The proportional-integral-derivative (PID) module (Figure 2-33) is used in process control applications that incorporate PID algorithms. An algorithm is a complex program based on mathematical calculations. A PID module allows process control to take place outside the CPU. This arrangement prevents the CPU from being burdened with complex calculations. The basic function of this module is to provide the control action required to maintain a process variable such as temperature, flow, level, or speed within set limits of a specified set point.

MOTION AND POSITION CONTROL MODULE

Motion and position control modules are used in applications involving accurate high-speed machining and packaging operations. Intelligent position and motion control modules permit PLCs to control stepper and servo motors.



Figure 2-33 PID module.
Source: Courtesy Red Lion Controls.

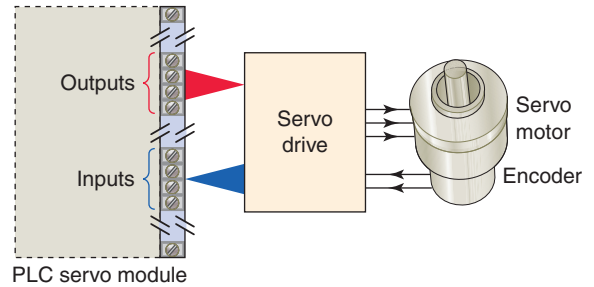


Figure 2-34 PLC servo module.

These systems require a drive, which contains the power electronics that translate the signals from the PLC module into signals required by the motor (Figure 2-34).

COMMUNICATION MODULES

Serial communications modules (Figure 2-35) are used to establish point-to-point connections with other intelligent devices for the exchange of data. Such connections are normally established with computers, operator stations, process control systems, and other PLCs. Communication modules allow the user to connect the PLC to high-speed local networks that may be different from the network communication provided with the PLC.



Figure 2-35 Serial communications module.
Source: Photo courtesy Automation Direct, www.automationdirect.com.

2.5 I/O Specifications

Manufacturers' specifications provide information about how an interface device is correctly and safely used. These specifications place certain limitations not only on the I/O module but also on the field equipment that it can operate. Some PLC systems support *hot swappable* I/O modules designed to be changed with the power on and the PLC operating. The following is a list of some typical manufacturers' I/O specifications, along with a short description of what is specified.

Typical Discrete I/O Module Specifications

NOMINAL INPUT VOLTAGE

This discrete input module voltage value specifies the magnitude (e.g., 5 V, 24 V, 230 V) and type (AC or DC) of user-supplied voltage that a module is designed to accept. Input modules are typically designed to operate correctly without damage within a range of plus or minus 10 percent of the input voltage rating. With DC input modules, the input voltage may also be expressed as an operating range (e.g., 24–60 volts DC) over which the module will operate.

INPUT THRESHOLD VOLTAGES

This discrete input module specification specifies two values: a minimum ON-state voltage that is the minimum voltage at which logic 1 is recognized as absolutely ON; and a maximum OFF-state voltage which is the voltage at which logic 0 is recognized as absolutely OFF.

NOMINAL CURRENT PER INPUT

This value specifies the minimum input current that the discrete input devices must be capable of driving to operate the input circuit. This input current value, in conjunction with the input voltage, functions as a threshold to protect against detecting noise or leakage currents as valid signals.

AMBIENT TEMPERATURE RATING

This value specifies what the maximum temperature of the air surrounding the I/O modules should be for best operating conditions.

INPUT ON/OFF DELAY

Also known as *response time*, this value specifies the maximum time duration required by an input module's circuitry to recognize that a field device has switched ON (input ON-delay) or switched OFF (input OFF-delay). This delay is a result of filtering circuitry provided to protect against contact bounce and voltage transients.

This input delay is typically in the 9 to 25 millisecond range.

OUTPUT VOLTAGE

This AC or DC value specifies the magnitude (e.g., 5 V, 115 V, 230 V) and type (AC or DC) of user-supplied voltage at which a discrete output module is designed to operate. The output field device that the module interfaces to the PLC must be matched to this specification. Output modules are typically designed to operate within a range of plus or minus 10 percent of the nominal output voltage rating.

OUTPUT CURRENT

These values specify the maximum current that a single output and the module as a whole can safely carry under load (at rated voltage). This rating is a function of the module's components and heat dissipation characteristics. A device drawing more than the rated output current results in overloading, causing the output fuse to blow. As an example, the specification may give each output a current limit of 1 A. The overall rating of the module current will normally be less than the total of the individuals. The overall rating might be 6 A because each of the eight devices would not normally draw their 1 A at the same time. Other names for the output current rating are *maximum continuous current* and *maximum load current*.

INRUSH CURRENT

An inrush current is a momentary surge of current that an AC or DC output circuit encounters when energizing inductive, capacitive, or filament loads. This value specifies the maximum inrush current and duration (e.g., 20 A for 0.1 s) for which an output circuit can exceed its maximum continuous current rating.

SHORT CIRCUIT PROTECTION

Short circuit protection is provided for AC and DC output modules by either fuses or some other current-limiting circuitry. This specification will designate whether the particular module's design has individual protection for each circuit or if fuse protection is provided for groups (e.g., 4 or 8) of outputs.

LEAKAGE CURRENT

This value specifies the amount of current still conducting through an output circuit even after the output has been turned off. Leakage current is a characteristic exhibited by solid-state switching devices such as transistors and triacs and is normally less than 5 milliamperes. Leakage current is normally not large enough to falsely trigger an output device but must be taken into consideration when switching very low current sensitive devices.

ELECTRICAL ISOLATION

Recall that I/O module circuitry is electrically isolated to protect the low-level internal circuitry of the PLC from high voltages that can be encountered from field device connections. The specification for electrical isolation, typically 1500 or 2500 volts AC, rates the module's capacity for sustaining an excessive voltage at its input or output terminals. Although this isolation protects the logic side of the module from excessive input or output voltages or current, the power circuitry of the module may be damaged.

POINTS PER MODULE

This specification defines the number of field inputs or outputs that can be connected to a single module. Most commonly, a discrete module will have 8, 16, or 32 circuits; however, low-end controllers may have only 2 or 4 circuits. Modules with 32 or 64 input or output bits are referred to as *high-density* modules. Some modules provide more than one common terminal, which allows the user to use different voltage ranges on the same card as well as to distribute the current more effectively.

BACKPLANE CURRENT DRAW

This value indicates the amount of current the module requires from the backplane. The sum of the backplane current drawn for all modules in a chassis is used to select the appropriate chassis power supply rating.

Typical Analog I/O Module Specifications

CHANNELS PER MODULE

Whereas individual circuits on discrete I/O modules are referred to as points, circuits on analog I/O modules are often referred to as channels. These modules normally have 4, 8, or 16 channels. Analog modules may allow for either single-ended or differential connections. *Single-ended* connections use a single ground terminal for all channels or for groups of channels. *Differential* connections use a separate positive and negative terminal for each channel. If the module normally allows 16 single-ended connections, it will generally allow only 8 differential connections. Single-ended connections are more susceptible to electrical noise.

INPUT CURRENT/VOLTAGE RANGE(S)

These are the voltage or current signal ranges that an analog input module is designed to accept. The input ranges must be matched accordingly to the varying current or voltage signals generated by the analog sensors.

OUTPUT CURRENT/VOLTAGE RANGE(S)

This specification defines the current or voltage signal ranges that a particular analog output module is designed to output under program control. The output ranges must be matched according to the varying voltage or current signals that will be required to drive the analog output devices.

INPUT PROTECTION

Analog input circuits are usually protected against accidentally connecting a voltage that exceeds the specified input voltage range.

RESOLUTION

The resolution of an analog I/O module specifies how accurately an analog value can be represented digitally. This specification determines the smallest measurable unit of current or voltage. The higher the resolution (typically specified in bits), the more accurately an analog value can be represented.

INPUT IMPEDANCE AND CAPACITANCE

For analog I/Os, these values must be matched to the external device connected to the module. Typical ratings are in Megohm ($M\Omega$) and picofarads (pF).

COMMON-MODE REJECTION

Noise is generally caused by electromagnetic interference, radio frequency interference, and ground loops. Common-mode noise rejection applies only to differential inputs and refers to an analog module's ability to prevent noise from interfering with data integrity on a single channel and from channel to channel on the module. Noise that is picked up equally in parallel wires is rejected because the difference is zero. Twisted pair wires are used to ensure that this type of noise is equal on both wires. Common-mode rejection is normally expressed in decibels or as a ratio.

2.6 The Central Processing Unit (CPU)

The central processing unit (CPU) is built into single-unit fixed PLCs while modular rack types typically use a plug-in module. CPU, controller, and processor are all terms used by different manufacturers to denote the same module that performs basically the same functions. Processors vary in processing speed and memory options. A processor module can be divided into two sections: the CPU section and the memory section (Figure 2-36). The CPU section executes the program and makes the decisions needed by the PLC to operate and communicate

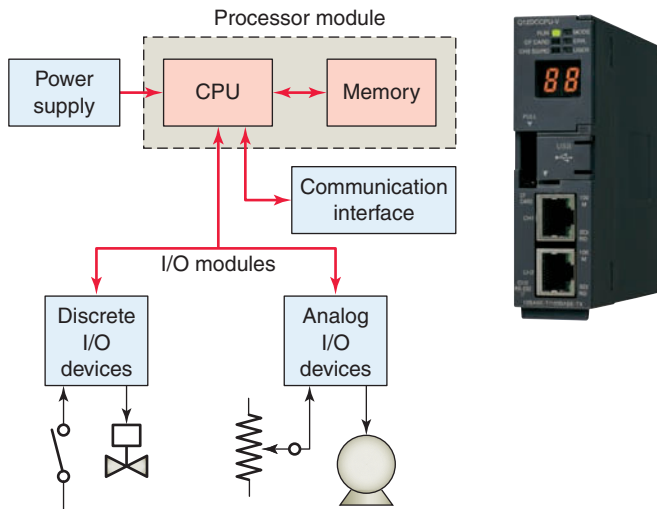


Figure 2-36 Sections of a PLC processor module.
Source: Courtesy Mitsubishi Automation.

with other modules. The memory section electronically stores the PLC program along with other retrievable digital information.

The PLC power supply provides the necessary power (typically 5 VDC) to the processor and I/O modules plugged into the backplane of the rack (Figure 2-37). Power supplies are available for most voltage sources encountered. The power supply converts 115 VAC or 230 VAC into the usable DC voltage required by the CPU, memory, and I/O electronic circuitry. PLC power supplies are normally designed to withstand momentary losses of power without affecting the operation of the PLC. *Hold-up time*, which is the length of time a PLC can tolerate a power loss, typically ranges from 10 milliseconds to 3 seconds.

The CPU contains the similar type of microprocessor found in a personal computer. The difference is that the program used with the microprocessor is designed to facilitate industrial control rather than provide general-purpose computing. The CPU executes the operating system, manages memory, monitors inputs, evaluates the

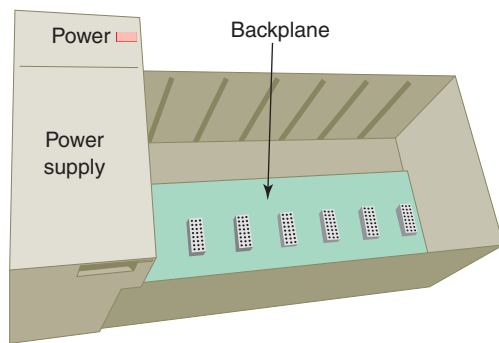


Figure 2-37 PLC power supply.

user logic (ladder program), and turns on the appropriate outputs.

The CPU of a PLC system may contain more than one processor. One advantage of using multiprocessing is that the overall operating speed is improved. Each processor has its own memory and programs, which operate simultaneously and independently. In such configurations the scan of each processor is parallel and independent thus reducing the total response time. Fault-tolerant PLC systems support dual processors for critical processes. These systems allow the user to configure the system with *redundant* (two) processors, which allows transfer of control to the second processor in the event of a processor fault.

Associated with the processor unit will be a number of status LED indicators to provide system diagnostic information to the operator (Figure 2-38). Also, a keyswitch may be provided that allows you to select one of the following three modes of operation: RUN, PROG, and REM.

RUN Position

- Places the processor in the Run mode
- Executes the ladder program and energizes output devices
- Prevents you from performing online program editing in this position
- Prevents you from using a programmer/operator interface device to change the processor mode

PROG Position

- Places the processor in the Program mode
- Prevents the processor from scanning or executing the ladder program, and the controller outputs are de-energized
- Allows you to perform program entry and editing
- Prevents you from using a programmer/operator interface device to change the processor mode

REM Position

- Places the processor in the Remote mode: either the REMote Run, REMote Program, or REMote Test mode
- Allows you to change the processor mode from a programmer/operator interface device
- Allows you to perform online program editing

The processor module also contains circuitry to communicate with the programming device. Somewhere on the module you will find a connector that allows the PLC to be connected to an external programming device. The decision-making capabilities of PLC processors go far beyond simple logic processing. The processor performs

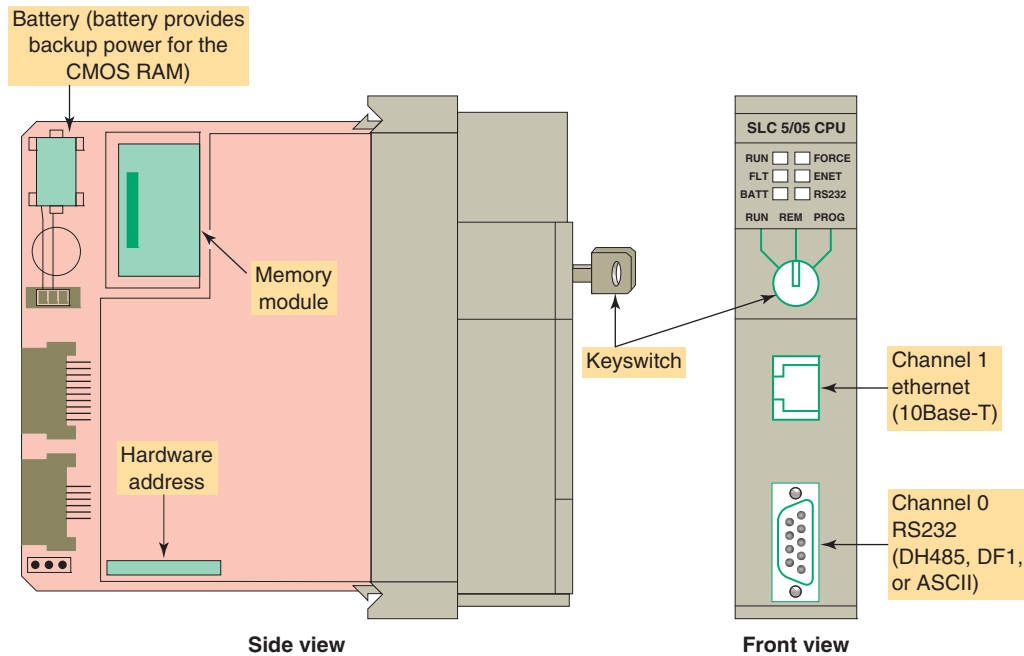


Figure 2-38 Typical processor module.

other functions such as timing, counting, latching, comparing, motion control and complex math functions.

PLC processors have changed constantly due to advancements in computer technology and greater demand from applications. Today, processors are faster and have additional instructions added as new models are introduced. Because PLCs are microprocessor based, they can be made to perform tasks that a computer can do. In addition to their control functions, PLCs can be networked to do supervisory control and data acquisition (SCADA).

Many electronic components found in processors and other types of PLC modules are sensitive to *electrostatic* voltages that can degrade their performance or damage them. The following static control procedures should be followed when handling and working with static-sensitive devices and modules:

- Ground yourself by touching a conductive surface before handling static-sensitive components.
- Wear a wrist strap that provides a path to bleed off any charge that may build up during work.
- Be careful not to touch the backplane connector or connector pins of the PLC system (always handle the circuit cards by the edge if possible).
- Be careful not to touch other circuit components in a module when you configure or replace its internal components.
- When not in use, store modules in its static-shield bag.
- If available, use a static-safe work station.

2.7 Memory Design

Memory is the element that stores information, programs, and data in a PLC. The user memory of a PLC includes space for the user program as well as addressable memory locations for storage of data. Data are stored in memory locations by a process called *writing*. Data are retrieved from memory by what is referred to as *reading*.

The complexity of the program determines the amount of memory required. Memory elements store individual pieces of information called *bits* (for *binary digits*). The amount of memory capacity is specified in increments of 1000 or in “K” increments, where 1 K is 1024 bytes of memory storage (a byte is 8 bits).

The program is stored in the memory as 1s and 0s, which are typically assembled in the form of 16-bit words. Memory sizes are commonly expressed in thousands of words that can be stored in the system; thus 2 K is a memory of 2000 words, and 64 K is a memory of 64,000 words. The memory size varies from as small as 1 K for small systems to 32 MB for very large systems (Figure 2-39). Memory capacity is an important prerequisite for determining whether a particular processor will handle the requirements of the specific application.

Memory location refers to an address in the CPU’s memory where a binary word can be stored. A word usually consists of 16 bits. Each binary piece of data is a bit and eight bits make up one byte (Figure 2-40). *Memory utilization* refers to the number of memory locations required to store each type of instruction. A rule of thumb

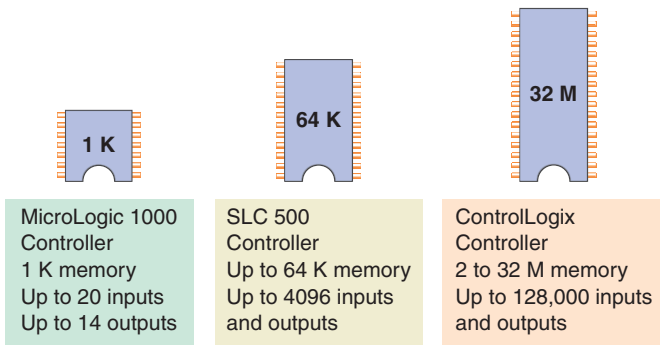


Figure 2-39 Typical PLC memory sizes.

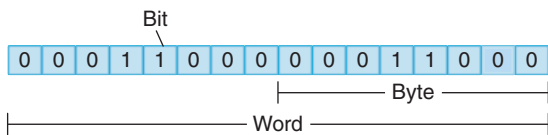


Figure 2-40 Memory bit, byte, and word.

for memory locations is one location per coil or contact. One K of memory would then allow a program containing 1000 coils and contacts to be stored in memory.

The memory of a PLC may be broken into sections that have specific functions. Sections of memory used to store the status of inputs and outputs are called input status files or tables and output status files or tables (Figure 2-41). These terms simply refer to a location where the status of an input or output device is stored. Each bit is either a 1 or 0, depending on whether the input is open or closed. A closed contact would have a binary 1 stored in its respective location in the input table, whereas an open contact would have a 0 stored. A lamp that is ON would have a 1 stored in its respective location in the output table, whereas a lamp that is OFF would have a 0 stored. Input and output image tables are constantly being revised by the CPU. Each time a memory location is examined, the table changes if the contact or coil has changed state.

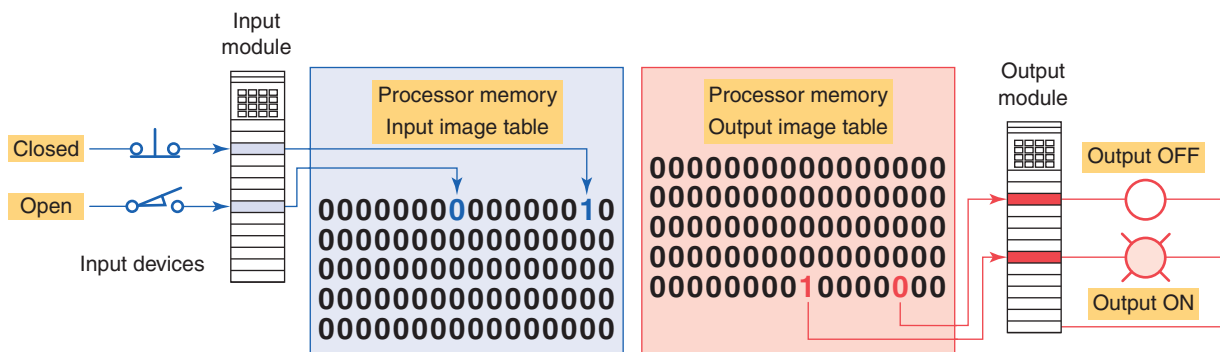


Figure 2-41 Input and output tables.

PLCs execute memory-checking routines to be sure that the PLC memory has not been corrupted. This memory checking is undertaken for safety reasons. It helps ensure that the PLC will not execute if memory is corrupted.

2.8 Memory Types

Memory can be placed into two general categories: volatile and nonvolatile. Volatile memory will lose its stored information if all operating power is lost or removed. Volatile memory is easily altered and is quite suitable for most applications when supported by battery backup.

Nonvolatile memory has the ability to retain stored information when power is removed accidentally or intentionally. As the name implies, programmable logic controllers have programmable memory that allows users to develop and modify control programs. This memory is made nonvolatile so that if power is lost, the PLC holds its programming.

Read Only Memory (ROM) stores programs, and data cannot be changed after the memory chip has been manufactured. ROM is normally used to store the programs and data that define the capabilities of the PLC. ROM memory is nonvolatile, meaning that its contents will not be lost if power is lost. ROM is used by the PLC for the operating system. The operating system is burned into ROM by the PLC manufacturer and controls the system software that the user uses to program the PLC.

Random Access Memory (RAM), sometimes referred to as *read-write (R/W) memory*, is designed so that information can be written into or read from the memory. RAM is used as a temporary storage area of data that may need to be quickly changed. RAM is volatile, meaning that the data stored in RAM will be lost if power is lost. A battery backup is required to avoid losing data in the event of a power loss (Figure 2-42). Most PLCs use CMOS-RAM technology for user memory. CMOS-RAM chips have very low current draw and can maintain memory with a lithium battery for an extended time, two to five

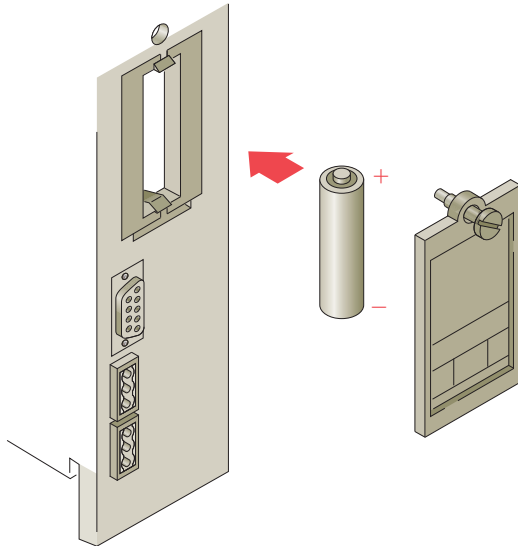


Figure 2-42 Battery used to back up processor RAM.

years in many cases. Some processors have a capacitor that provides at least 30 minutes of battery backup when the battery is disconnected and power is OFF.

Erasable Programmable Read-Only Memory (EPROM) provides some level of security against unauthorized or unwanted changes in a program. EPROMs are designed so that data stored in them can be read, but not easily altered without special equipment. For example, UV EPROMs (ultraviolet erasable programmable read-only memory) can only be erased with an ultraviolet light. EPROM memory is used to back up, store, or transfer PLC programs.

Electrically erasable programmable read-only memory (EEPROM) is a nonvolatile memory that offers the same programming flexibility as does RAM. The EEPROM can be electrically overwritten with new data instead of being erased with ultraviolet light. Because the EEPROM is nonvolatile memory, it does not require battery backup. It provides permanent storage of the program and can be changed easily using standard programming devices. Typically, an EEPROM memory module is used to store, back up, or transfer PLC programs (Figure 2-43).

Flash EEPROMs are similar to EEPROMs in that they can only be used for backup storage. The main difference comes in the flash memory: they are extremely fast at saving and retrieving files. In addition, they do not need to be physically removed from the processor for reprogramming; this can be done using the circuitry within the processor module in which they reside. Flash memory is also sometimes built into the processor module (Figure 2-44), where it automatically backs up parts of RAM. If power fails while a PLC with flash memory is running, the PLC

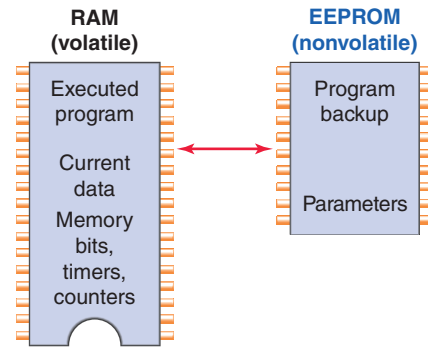


Figure 2-43 EEPROM memory module is used to store, back up, or transfer PLC programs.

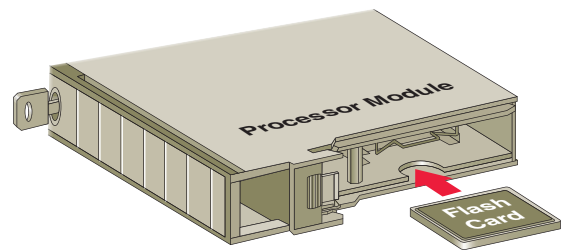


Figure 2-44 Flash memory card installed in a socket on the processor.

will resume running without having lost any working data after power is restored.

2.9 Programming Terminal Devices

A programming terminal device is needed to enter, modify, and troubleshoot the PLC program. PLC manufacturers use various types of programming devices. The simplest type is the hand-held type programmer shown in Figure 2-45. This proprietary programming device has a connecting cable so that it can be plugged into a PLC's programming port. Certain controllers use a plug-in panel rather than a hand-held device.

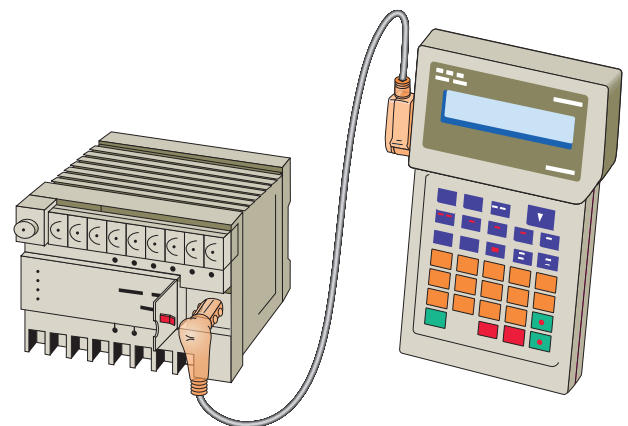


Figure 2-45 Hand-held programming terminal.

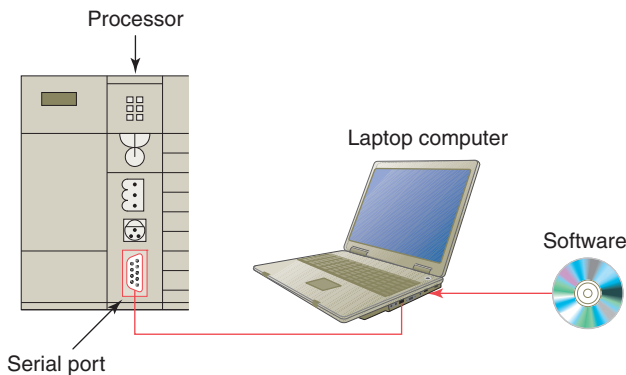


Figure 2-46 Personal computer used as the programming device.

Hand-held programmers are compact, inexpensive, and easy to use. These units contain multifunction keys and a liquid-crystal display (LCD) or light-emitting diode (LED) window. There are usually keys for instruction entering and editing, and navigation keys for moving around the program. Hand-held programmers have limited display capabilities. Some units will display only the last instruction that has been programmed, whereas other units will display from two to four rungs of ladder logic. So-called intelligent hand-held programmers are designed to support a certain family of PLCs from a specific manufacturer.

The most popular method of PLC programming is to use a personal computer (PC) in conjunction with the manufacturer's programming software (Figure 2-46). Typical capabilities of the programming software include online and offline program editing, online program monitoring, program documentation, diagnosing malfunctions in the PLC, and troubleshooting the controlled system. Hard-copy reports generated in the software can be printed on the computer's printer. Most software packages will not allow you to develop programs on another manufacturer's PLC. In some cases, a single manufacturer will have multiple PLC families, each requiring its own software to program.

2.10 Recording and Retrieving Data

Printers are used to provide hard-copy printouts of the processor's memory in ladder program format. Lengthy ladder programs cannot be shown completely on a screen. Typically, a screen shows a maximum of five rungs at a time. A printout can show programs of any length and analyze the complete program.

The PLC can have only one program in memory at a time. To change the program in the PLC, it is necessary either to enter a new program directly from the

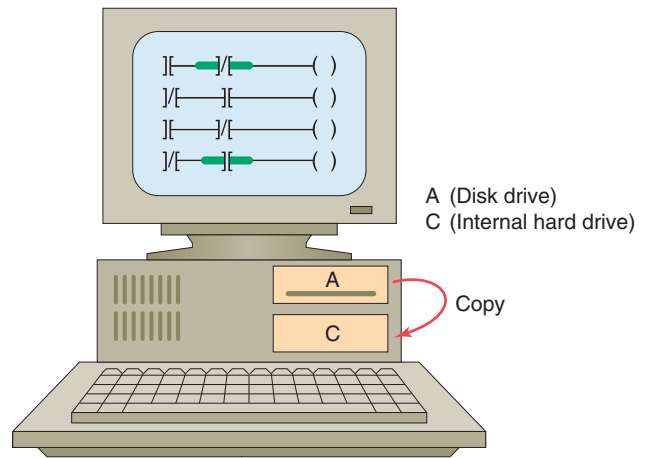


Figure 2-47 Copying programs to a computer hard drive.

keyboard or to download one from the computer hard drive (Figure 2-47). Some CPUs support the use of a memory cartridge that provides portable EEPROM storage for the user program (Figure 2-48). The cartridge can be used to copy a program from one PLC to another similar type PLC.

2.11 Human Machine Interfaces (HMIs)

A *human machine interface (HMI)* can be connected to communicate with a PLC and to replace pushbuttons, selector switches, pilot lights, thumbwheels, and other operator control panel devices (Figure 2-49). Luminescent touch-screen keypads provide an operator interface that operates like traditional hardwired control panels.

Human machine interfaces give the ability to the operator and to management to view the operation in real

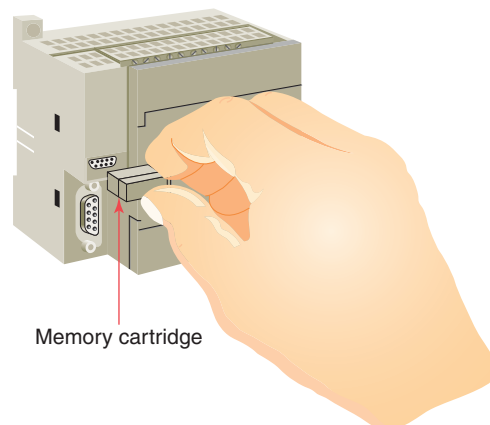


Figure 2-48 Memory cartridge provides portable storage for user program.



Figure 2-49 Human Machine Interfaces (HMIs).
Source: Photo courtesy Omron Industrial Automation, www.ia.omron.com.

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

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

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



Figure 2-50 Allen-Bradley Pico GFX-70 controller.
Source: Image Used with Permission of Rockwell Automation, Inc.

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

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



CHAPTER 2 REVIEW QUESTIONS

1. What is the function of a PLC input interface module?
2. What is the function of a PLC output interface module?
3. Define the term *logical rack*.
4. With reference to a PLC rack:
 - a. What is a *remote rack*?
 - b. Why are remote racks used?
5. How does the processor identify the location of a specific input or output device?
6. List the three basic elements of rack/slot-based addressing.
7. Compare bit level and word level addressing.
8. In what way does tag-based addressing differ from rack/slot-based addressing?
9. What do PC-based control systems use to interface with field devices?
10. What type of I/O modules have both inputs and outputs connected to them?
11. In addition to field devices what other connections are made to a PLC module?
12. Most PLC modules use plug-in wiring terminal strips. Why?
13. What are the advantage and the disadvantage of using high-density modules?
14. With reference to PLC discrete input modules:
 - a. What types of field input devices are suitable for use with them?
 - b. List three examples of discrete input devices.
15. With reference to PLC discrete output modules:
 - a. What types of field output devices are suitable for use with them?
 - b. List three examples of discrete output devices.
16. Explain the function of the backplane of a PLC rack.
17. What is the function of the optical isolator circuit used in discrete I/O module circuits?
18. Name the two distinct sections of an I/O module.
19. List four tasks performed by a discrete input module.
20. What electronic element can be used as the switching device for a 120 VAC discrete output interface module?
21. With reference to discrete output module current ratings:
 - a. What is the maximum current rating for a typical 120 VAC output module?
 - b. Explain one method of handling outputs with larger current requirements.
22. What electronic element can be used as the switching device for DC discrete output modules?
23. A discrete relay type output module can be used to switch either AC or DC load devices. Why?
24. With reference to sourcing and sinking I/O modules:
 - a. What current relationship are the terms *sourcing* and *sinking* used to describe?
 - b. If an I/O module is specified as a current-sinking type, then which type of field device (sinking or sourcing) it is electrically compatible with?
25. Compare discrete and analog I/O modules with respect to the type of input or output devices with which they can be used.
26. Explain the function of the analog-to-digital (A/D) converter circuit used in analog input modules.
27. Explain the function of the digital-to-analog (D/A) converter circuit used in analog output modules.
28. Name the two general sensing classifications for analog input modules.
29. List five common physical quantities measured by a PLC analog input module.
30. What type of cable is used when connecting a thermocouple to a voltage sensing analog input module? Why?
31. Explain the difference between a unipolar and bipolar analog input module.
32. The resolution of an analog input channel is specified as 0.3 mV. What does this tell you?
33. In what two ways can the loop power for current sensing input modules be supplied?
34. List three field devices that are commonly controlled by a PLC analog output module.
35. State one application for each of the following special I/O modules:
 - a. High-speed counter module
 - b. Thumbwheel module
 - c. TTL module

- d. Encoder-counter module
 - e. BASIC or ASCII module
 - f. Stepper-motor module
 - g. BCD-output module
36. List one application for each of the following intelligent I/O modules:
 - a. PID module
 - b. Motion and position control module
 - c. Communication module
 37. Write a short explanation for each of the following discrete I/O module specifications:
 - a. Nominal input voltage
 - b. Input threshold voltages
 - c. Nominal current per input
 - d. Ambient temperature rating
 - e. Input ON/OFF delay
 - f. Output voltage
 - g. Output current
 - h. Inrush current
 - i. Short circuit protection
 - j. Leakage current
 - k. Electrical isolation
 - l. Points per module
 - m. Backplane current draw
 38. Write a short explanation for each of the following analog I/O module specifications:
 - a. Channels per module
 - b. Input current/voltage range(s)
 - c. Output current/voltage range(s)
 - d. Input protection
 - e. Resolution
 - f. Input impedance and capacitance
 - g. Common-mode rejection
 39. Compare the function of the CPU and memory sections of a PLC processor.
 40. With reference to the PLC chassis power supply:
 - a. What conversion of power takes place within the power supply circuit?
 - b. Explain the term *hold-up time* as it applies to the power supply.
 41. Explain the purpose of a redundant PLC processor.
 42. Describe three typical modes of operation that can be selected by the keyswitch of a processor.
 43. State five other functions, in addition to simple logic processing, that PLC processors are capable of performing.
 44. List five important procedures to follow when handling static-sensitive PLC components.
 45. Define each of the following terms as they apply to the memory element of a PLC:
 - a. writing
 - b. reading
 - c. bits
 - d. location
 - e. utilization
 46. With reference to the I/O image tables:
 - a. What information is stored in PLC input and output tables?
 - b. What is the input status of a closed switch stored as?
 - c. What is the input status of an open switch stored as?
 - d. What is the status of an output that is ON stored as?
 - e. What is the status of an output that is OFF stored as?
 47. Why do PLCs execute memory-checking routines?
 48. Compare the memory storage characteristics of volatile and nonvolatile memory elements.
 49. What information is normally stored in the ROM memory of a PLC?
 50. What information is normally stored in the RAM memory of a PLC?
 51. What information is normally stored in an EEPROM memory module?
 52. What are the advantages of a processor that utilizes a flash memory card?
 53. List three functions of a PLC programming terminal device.
 54. Give one advantage and one limitation to the use of hand-held programming devices.
 55. What is required for a personal computer to be used as a PLC programming terminal?
 56. List four important capabilities of PLC programming software.
 57. How many programs can a PLC have stored in memory at any one time?
 58. Outline four functions that an HMI interface screen can be configured to perform.



CHAPTER 2 PROBLEMS

1. A discrete 120 VAC output module is to be used to control a 230 VDC solenoid valve. Draw a diagram showing how this could be accomplished using an interposing relay.
2. Assume a thermocouple, which supplies the input to an analog input module, generates a linear voltage of from 20 mV to 50 mV when the temperature changes from 750°F to 1250°F. How much voltage will be generated when the temperature of the thermocouple is at 1000°F?
3. With reference to I/O module specifications:
 - a. If the ON-delay time of a given discrete input module is specified as 12 milliseconds, how much is this expressed in seconds?
 - b. If the output leakage current of a discrete output module is specified as 950 μ A, how much is this expressed in amperes?
 - c. If the ambient temperature rating for an I/O module is specified as 60°C, how much is this expressed in degrees Fahrenheit?
4. Create a five-digit code using the SLC 500 rack/slot-based addressing format for each of the following:
 - a. A pushbutton connected to terminal 5 of module group 2 located on rack 1.
 - b. A lamp connected to terminal 3 of module group 0 located on rack 2.
5. Assume the triac of an AC discrete output module fails in the shorted state. How would this affect the device connected to this output?
6. A personal computer is to be used to program several different PLCs from different manufacturers. What would be required?

3

Number Systems and Codes



Image Used with Permission of Rockwell Automation, Inc.

Chapter Objectives

After completing this chapter, you will be able to:

- 3.1** Define the decimal, binary, octal, and hexadecimal numbering systems and be able to convert from one numbering or coding system to another
- 3.2** Explain the BCD, Gray, and ASCII code systems
- 3.3** Define the terms *bit*, *byte*, *word*, *least significant bit (LSB)*, and *most significant bit (MSB)* as they apply to binary memory locations
- 3.4** Add, subtract, multiply, and divide binary numbers

Using PLCs requires us to become familiar with other number systems besides decimal. Some PLC models and individual PLC functions use other numbering systems. This chapter deals with some of these numbering systems, including binary, octal, hexadecimal, BCD, Gray, and ASCII. The basics of each system, as well as conversion from one system to another, are explained.

3.1 Decimal System

Knowledge of different number systems and digital codes is quite useful when working with PLCs or with most any type of digital computer. This is true because a basic requirement of these devices is to represent, store, and operate on numbers. In general, PLCs work on binary numbers in one form or another; these are used to represent various codes or quantities.

The *decimal system*, which is most common to us, has a base of 10. The radix or base of a number system determines the total number of different symbols or digits used by that system. For instance, in the decimal system, 10 unique numbers or digits—i.e., the digits 0 through 9—are used: the total number of symbols is the same as the base, and the symbol with the largest value is 1 less than the base.

The value of a decimal number depends on the digits that make up the number and the place value of each digit. A place (weight) value is assigned to each position that a digit would hold from right to left. In the decimal system the first position, starting from the rightmost position, is 0; the second is 1; the third is 2; and so on up to the last position. The weighted value of each position can be expressed as the base (10 in this case) raised to the power of the position. For the decimal system then, the position weights are 1, 10, 100, 1000, and so on. Figure 3-1 illustrates how the value of a decimal number can be calculated by multiplying each digit by the weight of its position and summing the results.

3.2 Binary System

The *binary system* uses the number 2 as the base. The only allowable digits are 0 and 1. With digital circuits it is easy to distinguish between two voltage levels (i.e., +5 V and 0 V), which can be related to the binary digits 1 and 0 (Figure 3-2). Therefore the binary system can be applied quite easily to PLCs and computer systems.

Since the binary system uses only two digits, each position of a binary number can go through only two

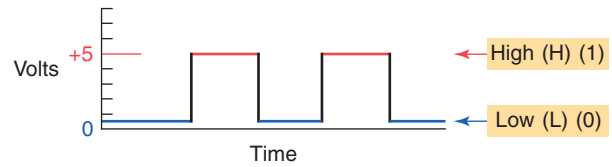


Figure 3-2 Digital signal waveform.

changes, and then a 1 is carried to the immediate left position. Table 3-1 shows a comparison among four common number systems: decimal (base 10), octal (base 8), hexadecimal (base 16), and binary (base 2). Note that all numbering systems start at *zero*.

The decimal equivalent of a binary number can be determined in a manner similar to that used for a decimal number. This time the weighted values of the positions are 1, 2, 4, 8, 16, 32, 64, and so on. The weighted value, instead of being 10 raised to the power of the position, is 2 raised to the power of the position. Figure 3-3 illustrates how the binary number 10101101 is converted to its decimal equivalent: 173.

Each digit of a binary number is known as a *bit*. In a PLC the processor-memory element consists of hundreds or thousands of locations. These locations, or registers,

Table 3-1 Number System Comparisons

Decimal	Octal	Hexadecimal	Binary
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111
16	20	10	10000
17	21	11	10001
18	22	12	10010
19	23	13	10011
20	24	14	10100

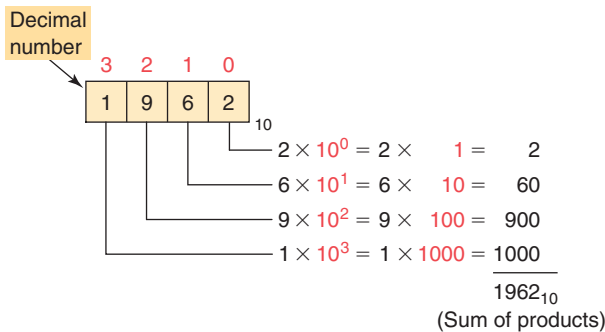


Figure 3-1 Weighted value in the decimal system.

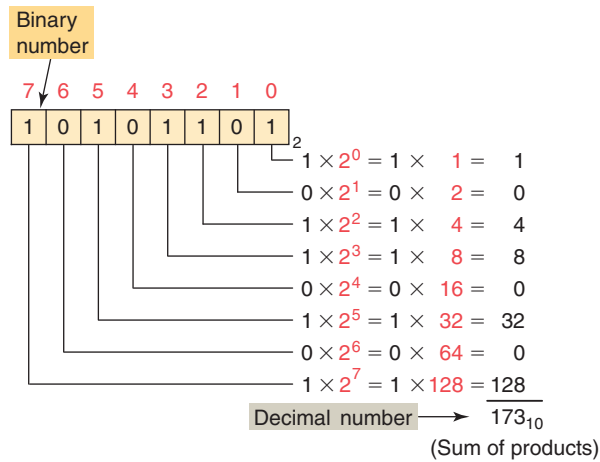


Figure 3-3 Converting a binary number to a decimal number.

are referred to as words. Each *word* is capable of storing data in the form of binary digits, or bits. The number of bits that a word can store depends on the type of PLC system used. Sixteen-bit and 32-bit words are the most common. Bits can also be grouped within a word into bytes. A group of 8 bits is a byte, and a group of 2 or more bytes is a word. Figure 3-4 illustrates a 16-bit word made up of 2 bytes. The least significant bit (LSB) is the digit that represents the smallest value, and the most significant bit (MSB) is the digit that represents the largest value. A bit within the word can exist only in two states: a logical 1 (or ON) condition, or a logical 0 (or OFF) condition.

PLC memory is organized using bytes, single words, or double words. Older PLCs use 8-bit or 16-bit memory words while newer systems, such as the ControlLogix platform from Allen-Bradley, use 32-bit double words. The size of the programmable controller memory relates to the amount of user program that can be stored. If the memory size is 1 K word (Figure 3-5), it can store 1024 words or 16,384 (1024×16) bits of information using 16-bit words, or 32,768 (1024×32) bits using 32-bit words.

To convert a decimal number to its binary equivalent, we must perform a series of divisions by 2. Figure 3-6 illustrates the conversion of the decimal number 47 to binary. We start by dividing the decimal number by 2. If there is a remainder, it is placed in the LSB of the binary

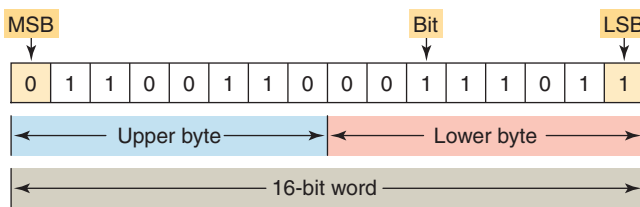


Figure 3-4 A 16-bit word.

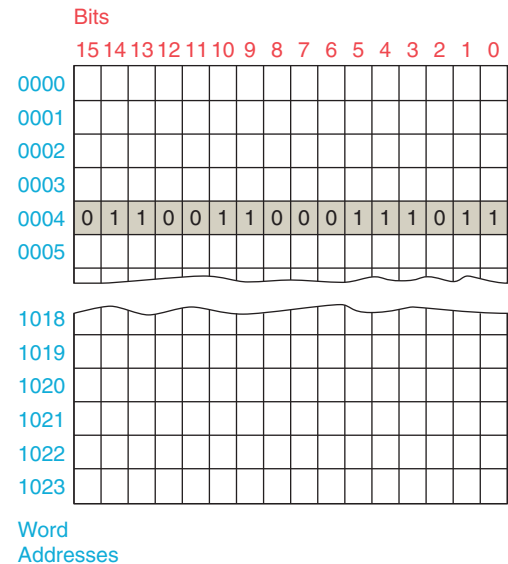


Figure 3-5 1-K word memory.

number. If there is no remainder, a 0 is placed in the LSB. The result of the division is brought down and the process is repeated until the result of successive divisions has been reduced to 0.

Even though the binary system has only two digits, it can be used to represent any quantity that can be represented in the decimal system. All PLCs work internally in the binary system. The processor, being a digital device, understands only 0s and 1s, or binary.

Computer memory is, then, a series of binary 1s and 0s. Figure 3-7 shows the output status file for an Allen-Bradley SLC 500 modular chassis, which is made up of single bits grouped into 16-bit words. One 16-bit output file word is reserved for each slot in the chassis. Each bit represents the ON or OFF state of one output point. These points are numbered 0 through 15 across the top row from

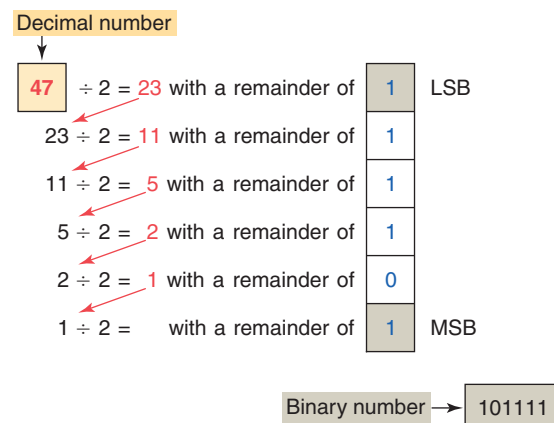


Figure 3-6 Converting a decimal number to a binary number.

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

Figure 3-7 SLC 500 output status file.

right to left. The column on the far right lists the output module address. Although the table in Figure 3-7 illustrates sequentially addressed output status file words, in reality a word is created in the table only if the processor finds an output module residing in a particular slot. If the slot is empty, no word will be created.

3.3 Negative Numbers

If a decimal number is positive, it has a plus sign; if a number is negative, it has a minus sign. In binary number systems, such as used in a PLC, it is not possible to use positive and negative symbols to represent the polarity of a number. One method of representing a binary number as either a positive or negative value is to use an extra digit, or *sign bit*, at the MSB side of the number. In the sign bit position, a 0 indicates that the number is positive, and a 1 indicates a negative number (Table 3-2).

Another method of expressing a negative number in a digital system is by using the complement of a binary number. To complement a binary number, change all the

1s to 0s and all the 0s to 1s. This is known as the 1's complement form of a binary number. For example, the 1's complement of 1001 is 0110.

The most common way to express a negative binary number is to show it as a 2's complement number. The 2's complement is the binary number that results when 1 is added to the 1's complement. This system is shown in Table 3-3. A zero sign bit means a positive number, whereas a 1 sign bit means a negative number.

Using the 2's complement makes it easier for the PLC to perform mathematical operations. The correct sign bit is generated by forming the 2's complement. The PLC knows that a number retrieved from memory is a negative number if the MSB is 1. Whenever a negative number is entered from a keyboard, the PLC stores it as a 2's complement. What follows is the original number in true binary followed by its 1's complement, its 2's complement, and finally, its decimal equivalent.

Table 3-2 Signed Binary Numbers

Magnitude Sign	Decimal Value
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

Same as binary numbers

Table 3-3 1's and 2's Complement Representation of Positive and Negative Numbers

Signed Decimal	1's Complement	2's Complement
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
0	0000	0000
-1	1110	1111
-2	1101	1110
-3	1100	1101
-4	1011	1100
-5	1010	1011
-6	1001	1010
-7	1000	1001

Same as binary numbers

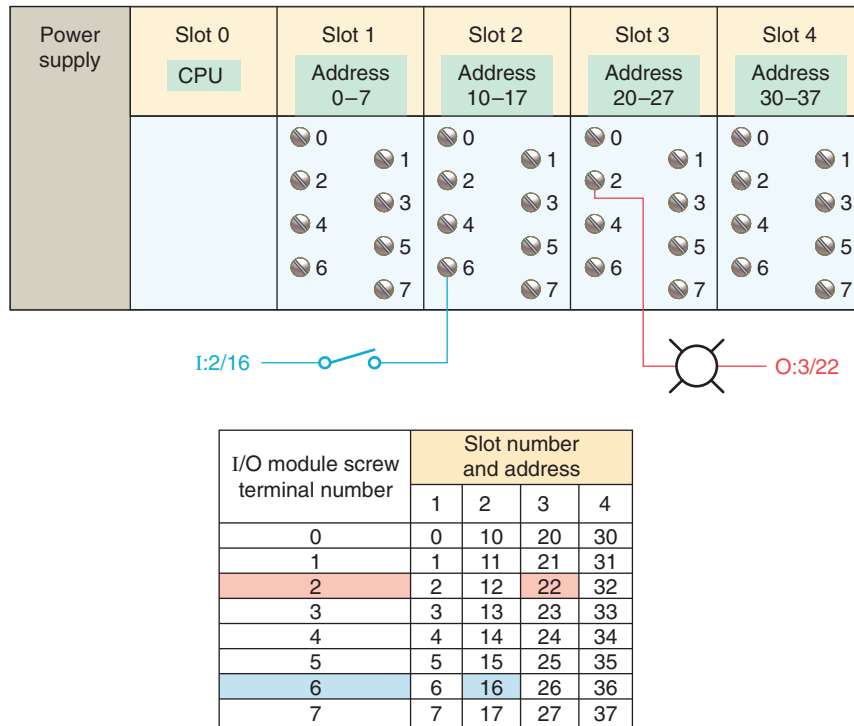


Figure 3-8 Addressing of I/O modules using the octal numbering system.

3.4 Octal System

To express the number in the binary system requires many more digits than in the decimal system. Too many binary digits can become cumbersome to read or write. To solve this problem, other related numbering systems are used.

The *octal numbering system*, a base 8 system, is used because 8 data bits make up a byte of information that can be addressed. Figure 3-8 illustrates the addressing of I/O modules using the octal numbering system. The digits range from 0 to 7; therefore, numbers 8 and 9 are not allowed. Allen-Bradley PLC-5 processors use octal-based I/O addressing while the SLC 500 and Logix controllers use decimal-base 10 addressing.

Octal is a convenient means of handling large binary numbers. As shown in Table 3-4, one octal digit can be used to express three binary digits. As in all other numbering systems, each digit in an octal number has a weighted decimal value according to its position. Figure 3-9 illustrates how the octal number 462 is converted to its decimal equivalent: 306.

Octal converts easily to binary equivalents. For example, the octal number 462 is converted to its binary equivalent by assembling the 3-bit groups, as illustrated in Figure 3-10. Notice the simplicity of the notation: the octal 462 is much easier to read and write than its binary equivalent is.

Table 3-4 Binary and Related Octal Code

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

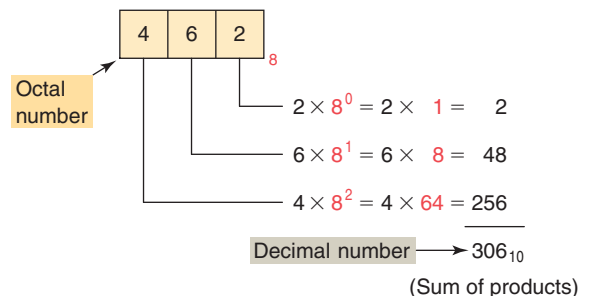


Figure 3-9 Converting an octal number to a decimal number.

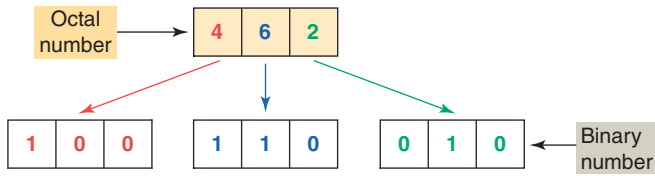


Figure 3-10 Converting an octal number to a binary number.

3.5 Hexadecimal System

The hexadecimal (hex) numbering system is used in programmable controllers because a word of data consists of 16 data bits, or two 8-bit bytes. The hexadecimal system is a base 16 system, with A to F used to represent decimal numbers 10 to 15 (Table 3-5). The hexadecimal numbering system allows the status of a large number of binary bits to be represented in a small space, such as on a computer screen or PLC programming device display.

The techniques used when converting hexadecimal to decimal and decimal to hexadecimal are the same as those used for binary and octal. To convert a hexadecimal number to its decimal equivalent, the hexadecimal digits in the columns are multiplied by the base 16 weight, depending on digit significance. Figure 3-11 illustrates how the conversion would be done for the hex number 1B7.

Hexadecimal numbers can easily be converted to binary numbers. Conversion is accomplished by writing the

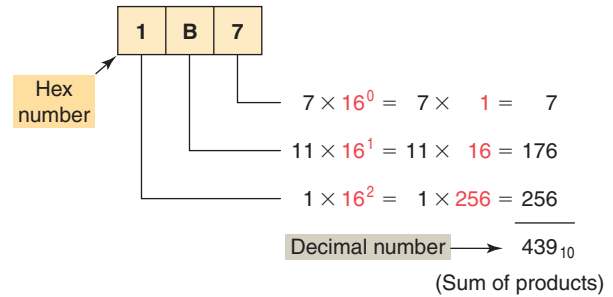


Figure 3-11 Converting a hexadecimal number to a decimal number.

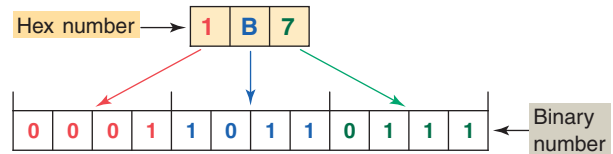


Figure 3-12 Converting a hexadecimal number to a binary number.

4-bit binary equivalent of the hex digit for each position, as illustrated in Figure 3-12.

3.6 Binary Coded Decimal (BCD) System

The binary coded decimal (BCD) system provides a convenient way of handling large numbers that need to be input to or output from a PLC. As you can see from looking at the various number systems, there is no easy way to go from binary to decimal and back. The BCD system provides a means of converting a code readily handled by humans (decimal) to a code readily handled by the equipment (binary). PLC thumbwheel switches and LED displays are examples of PLC devices that make use of the BCD number system. Table 3-6 shows examples of numeric values in decimal, binary, BCD, and hexadecimal representation.

The BCD system uses 4 bits to represent each decimal digit. The 4 bits used are the binary equivalents of the numbers from 0 to 9. In the BCD system, the largest decimal number that can be displayed by any four digits is 9.

The BCD representation of a decimal number is obtained by replacing each decimal digit by its BCD equivalent. To distinguish the BCD numbering system from a binary system, a BCD designation is placed to the right of the units digit. The BCD representation of the decimal number 7863 is shown in Figure 3-13.

A thumbwheel switch is one example of an input device that uses BCD. Figure 3-14 shows a single-digit BCD thumbwheel. The circuit board attached to the thumbwheel has one connection for each bit's weight plus

Table 3-5 Hexadecimal Numbering System

Hexadecimal	Binary	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

Table 3-6 Numeric Values in Decimal, Binary, BCD, and Hexadecimal Representation

Decimal	Binary	BCD	Hexadecimal
0	0	0000	0
1	1	0001	1
2	10	0010	2
3	11	0011	3
4	100	0100	4
5	101	0101	5
6	110	0110	6
7	111	0111	7
8	1000	1000	8
9	1001	1001	9
10	1010	0001 0000	A
11	1011	0001 0001	B
12	1100	0001 0010	C
13	1101	0001 0011	D
14	1110	0001 0100	E
15	1111	0001 0101	F
16	1 0000	0001 0110	10
17	1 0001	0001 0111	11
18	1 0010	0001 1000	12
19	1 0011	0001 1001	13
20	1 0100	0010 0000	14
126	111 1110	0001 0010 0110	7E
127	111 1111	0001 0010 0111	7F
128	1000 0000	0001 0010 1000	80
510	1 1111 1110	0101 0001 0000	1FE
511	1 1111 1111	0101 0001 0001	1FF
512	10 0000 0000	0101 0001 0010	200

a common connection. The operator dials in a decimal digit between 0 and 9, and the thumbwheel switch outputs the equivalent 4 bits of BCD data. In this example, the number eight is dialed to produce the input bit pattern of

1000. A four-digit thumbwheel switch, similar to the one shown, would control a total of 16 (4×4) PLC inputs.

Scientific calculators are available to convert numbers back and forth between decimal, binary, octal, and

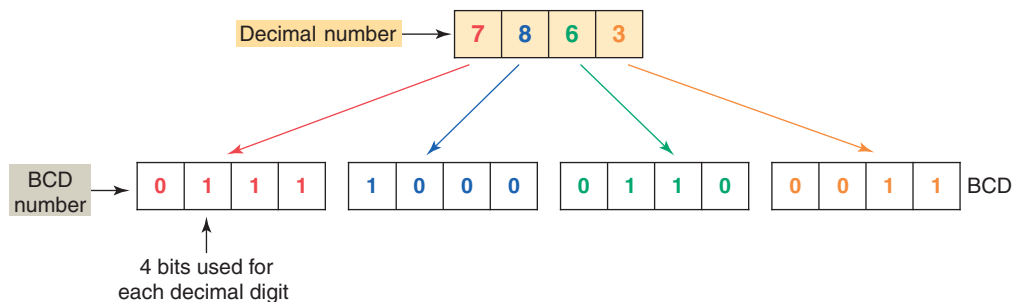


Figure 3-13 The BCD representation of a decimal number.

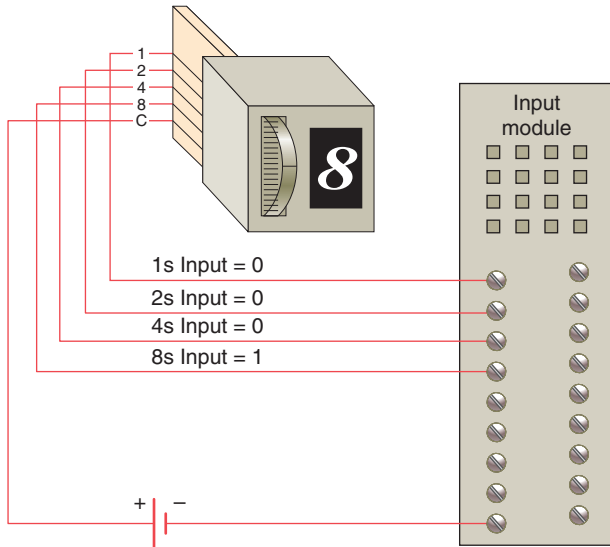


Figure 3-14 BCD thumbwheel switch interfaced to a PLC.

hexadecimal. In addition, PLCs contain number conversion functions such as illustrated in Figure 3-15. BCD-to-binary conversion is required for the input while binary-to-BCD conversion is required for the output. The PLC convert-to-decimal instruction will convert the binary bit pattern at the source address, N7:23, into a BCD bit pattern of the same decimal value as the destination address, O:20. The instruction executes every time it is scanned, and the instruction is true.

Many PLCs allow you to change the format of the data that the data monitor displays. For example, the change radix function found on Allen-Bradley controllers allows you to change the display format of data to binary, octal, decimal, hexadecimal, or ASCII.

3.7 Gray Code

The Gray code is a special type of binary code that does not use position weighting. In other words, each position does not have a definite weight. The Gray code is set up

Table 3-7 Gray Code and Binary Equivalent

Gray Code	Binary
0000	0000
0001	0001
0011	0010
0010	0011
0110	0100
0111	0101
0101	0110
0100	0111
1100	1000
1101	1001
1111	1010
1110	1011
1010	1100
1011	1101
1001	1110
1000	1111

so that as we progress from one number to the next, only one bit changes. This can be quite confusing for counting circuits, but it is ideal for encoder circuits. For example, absolute encoders are position transducers that use the Gray code to determine angular position. The Gray code has the advantage that for each “count” (each transition from one number to the next) only one digit changes. Table 3-7 shows the Gray code and the binary equivalent for comparison.

In binary, as many as four digits could change for a single “count.” For example, the transition from binary 0111 to 1000 (decimal 7 to 8) involves a change in all four digits. This kind of change increases the possibility for error in certain digital circuits. For this reason, the Gray code is considered to be an error-minimizing code. Because only one bit

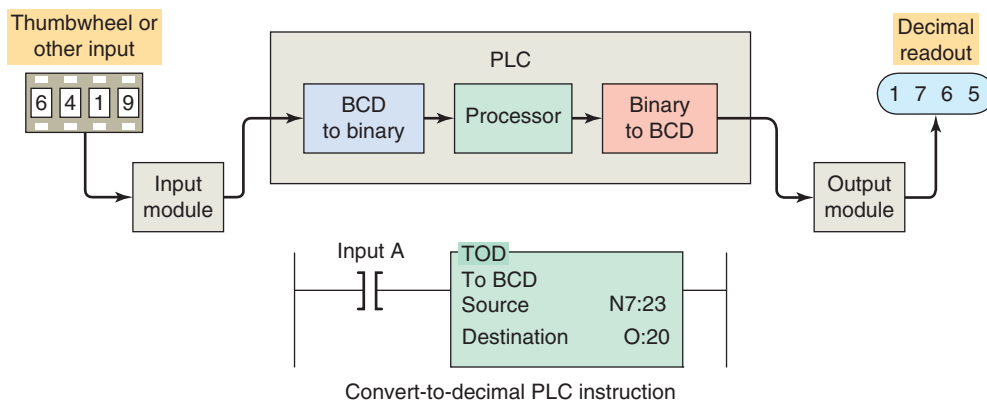


Figure 3-15 PLC number conversion.

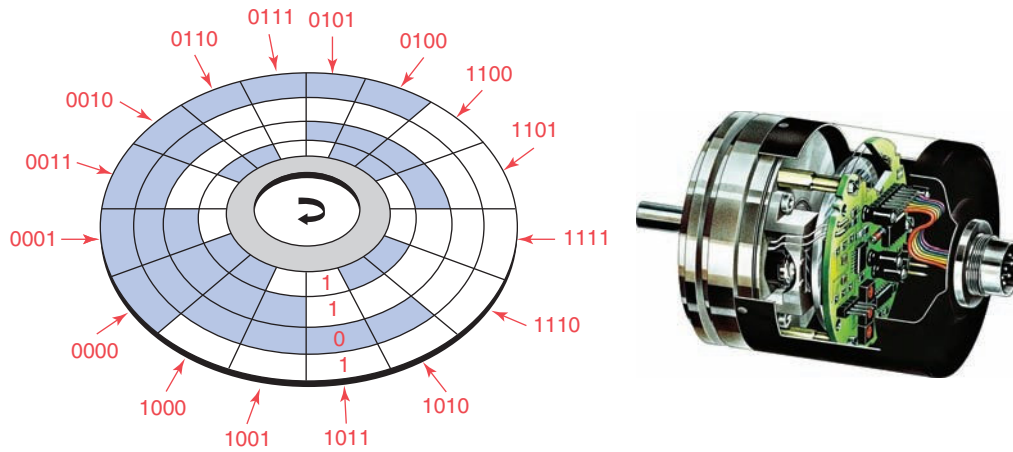


Figure 3-16 Optical encoder disk.
Source: Photo courtesy Baumer Electric.

changes at a time, the speed of transition for the Gray code is considerably faster than that for codes such as BCD.

Gray codes are used with position encoders for accurate control of the motion of robots, machine tools, and servomechanisms. Figure 3-16 shows an optical encoder disk that uses a 4-bit Gray code to detect changes in angular position. In this example, the encoder disk is attached to a rotating shaft and outputs a digital Gray code signal that is used to determine the position of the shaft. A fixed array of photo diodes senses the reflected light from each of the cells across a row of the encoder path. Depending on the amount of light reflected, each cell will output a voltage corresponding to a binary 1 or 0. Thus, a different 4-bit word is generated for each row of the disk.

3.8 ASCII Code

ASCII stands for American Standard Code for Information Interchange. It is an alphanumeric code because it includes letters as well as numbers. The characters accessed by the ASCII code include 10 numeric digits; 26 lowercase and 26 uppercase letters of the alphabet; and about 25 special characters, including those found on a standard typewriter. Table 3-8 shows a partial listing of the ASCII code. It is used to interface the PLC CPU with alphanumeric keyboards and printers.

The keystrokes on the keyboard of a computer are converted directly into ASCII for processing by the computer. Each time you press a key on a computer keyboard, a 7- or 8-bit word is stored in computer memory to represent the alphanumeric, function, or control data represented by the specific keyboard key that was depressed. ASCII input modules convert ASCII code input information from an external device to alphanumeric information that the PLC can process. The communication interfacing is done through either an RS-232 or RS-422 protocol. Modules

are available that will transmit and receive ASCII files and that can be used to create an operator interface. The user writes a program in the BASIC language that operates in conjunction with the ladder logic as the program runs.

3.9 Parity Bit

Some PLC communication systems use a binary digit to check the accuracy of data transmission. For example, when data are transferred between PLCs, one of the binary digits may be accidentally changed from a 1 to a 0. This can happen because of a transient or a noise or because of a failure in some portion of the transmission network. A parity bit is used to detect errors that may occur while a word is moved.

Parity is a system in which each character transmitted contains one additional bit. That bit is known as a parity bit. The bit may be a binary 0 or binary 1, depending on the number of 1s and 0s in the character itself. Two systems of parity are normally used: odd and even. Odd parity means that the total number of binary 1 bits in the character, including the parity bit, is odd. Even parity means that the number of binary 1 bits in the character, including the parity bit, is even. Examples of odd and even parity are shown in Table 3-9.

3.10 Binary Arithmetic

Arithmetic circuit units form a part of the CPU. Mathematical operations include addition, subtraction, multiplication, and division. Binary addition follows rules similar to decimal addition. When adding with binary numbers, there are only four conditions that can occur:

$$\begin{array}{cccc}
 0 & 1 & 0 & 1 \\
 +0 & +0 & +1 & +1 \\
 \hline
 0 & 1 & 1 & 0 \text{ carry } 1
 \end{array}$$

Table 3-8 Partial Listing of ASCII Code

Character	7-Bit ASCII	Character	7-Bit ASCII
A	100 0001	X	101 1000
B	100 0010	Y	101 1001
C	100 0011	Z	101 1010
D	100 0100	0	011 0000
E	100 0101	1	011 0001
F	100 0110	2	011 0010
G	100 0111	3	011 0011
H	100 1000	4	011 0100
I	100 1001	5	011 0101
J	100 1010	6	011 0110
K	100 1011	7	011 0111
L	100 1100	8	011 1000
M	100 1101	9	011 1001
N	100 1110	blank	010 0000
O	100 1111	.	010 1110
P	101 0000	,	010 1100
Q	101 0001	+	010 1011
R	101 0010	-	010 1101
S	101 0011	#	010 0011
T	101 0100	(010 1000
U	101 0101	%	010 0101
V	101 0110	=	011 1101
W	101 0111		

The first three conditions are easy because they are like adding decimals, but the last condition is slightly different. In decimal, $1 + 1 = 2$. In binary, a 2 is written 10. Therefore, in binary, $1 + 1 = 0$, with a carry of 1 to the next most significant place value. When adding

larger binary numbers, the resulting 1s are carried into higher-order columns, as shown in the following examples.

Table 3-9 Odd and Even Parity

Character	Even Parity Bit	Odd Parity Bit
0000	0	1
0001	1	0
0010	1	0
0011	0	1
0100	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1

Decimal **Equivalent binary**

$$\begin{array}{r}
 5 \\
 +2 \\
 \hline
 7
 \end{array}
 \qquad
 \begin{array}{r}
 101 \\
 + 10 \\
 \hline
 111
 \end{array}$$

$$\begin{array}{r}
 10 \\
 + 3 \\
 \hline
 13
 \end{array}
 \qquad
 \begin{array}{r}
 \text{carry} \\
 1 \\
 10 \ 10 \\
 + \quad | 11 \\
 \hline
 11 \ 01
 \end{array}$$

$$\begin{array}{r}
 26 \\
 +12 \\
 \hline
 38
 \end{array}
 \qquad
 \begin{array}{r}
 \text{carry} \quad \text{carry} \\
 1 \quad 1 \\
 1 \quad 1 \\
 1010 \\
 + \quad | 1100 \\
 \hline
 1 \ 0 \ 0110
 \end{array}$$

In arithmetic functions, the initial numeric quantities that are to be combined by subtraction are the minuend

and subtrahend. The result of the subtraction process is called the difference, represented as:

$$\begin{array}{r} A \text{ (minuend)} \\ -B \text{ (subtrahend)} \\ \hline C \text{ (difference)} \end{array}$$

To subtract from larger binary numbers, subtract column by column, borrowing from the adjacent column when necessary. Remember that when borrowing from the adjacent column, there are now two digits, i.e., 0 borrow 1 gives 10.

EXAMPLE

Subtract 1001 from 1101.

$$\begin{array}{r} 1101 \\ -1001 \\ \hline 0100 \end{array}$$

Subtract 0111 from 1011.

$$\begin{array}{r} 1011 \\ -0111 \\ \hline 0100 \end{array}$$

Binary numbers can also be negative. The procedure for this calculation is identical to that of decimal numbers because the smaller value is subtracted from the larger value and a negative sign is placed in front of the result.

EXAMPLE

Subtract 111 from 100.

$$\begin{array}{r} 111 \\ -100 \\ \hline -011 \end{array}$$

Subtract 11011 from 10111.

$$\begin{array}{r} 11011 \\ -10111 \\ \hline -00100 \end{array}$$

There are other methods available for doing subtraction:

- 1's complement
- 2's complement

The procedure for subtracting numbers using the 1's complement is as follows:

- Step 1 Change the subtrahend to 1's complement.
- Step 2 Add the two numbers.
- Step 3 Remove the last carry and add it to the number (end-around carry).

Decimal	Binary
$\begin{array}{r} 10 \\ -6 \\ \hline 4 \end{array}$	$\begin{array}{r} 1010 \\ -0110 \\ \hline 100 \end{array}$
	$\begin{array}{r} 1010 \\ +1001 \\ \hline 10011 \\ \hline 100 \end{array}$
	<p style="margin-left: 100px;">End-around carry $\rightarrow +1$</p>

When there is a carry at the end of the result, the result is positive. When there is no carry, then the result is negative and a minus sign has to be placed in front of it.

EXAMPLE

Subtract 11011 from 01101.

$\begin{array}{r} 01101 \\ + \cancel{0}00100 \\ \hline 10001 \end{array}$	<p>The 1's complement</p> <p>There is no carry, so we take the 1's complement and add the minus sign:</p> <p><u>-01110</u></p>
---------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------

For subtraction using the 2's complement, the 2's complement is added instead of subtracting the numbers. In the result, if the carry is a 1, then the result is positive; if the carry is a 0, then the result is negative and requires a minus sign.

EXAMPLE

Subtract 101 from 111.

$\begin{array}{r} 111 \\ + \cancel{0}011 \\ \hline 1010 \end{array}$	<p>The 2's complement</p> <p>The first 1 indicates that the result is positive, so it is disregarded:</p> <p><u>010</u></p>
----------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------

EXAMPLE

Subtract 101 from 111.

$$\begin{array}{r}
 111 \\
 + \textcircled{0}11 \\
 \hline
 1010
 \end{array}$$

The 2's complement

The first 1 indicates that the result is positive, so it is disregarded:

$$\underline{010}$$

Binary numbers are multiplied in the same manner as decimal numbers. When multiplying binary numbers, there are only four conditions that can occur:

$$\begin{array}{l}
 0 \times 0 = 0 \\
 0 \times 1 = 0 \\
 1 \times 0 = 0 \\
 1 \times 1 = 1
 \end{array}$$

To multiply numbers with more than one digit, form partial products and add them together, as shown in the following example.

Decimal	Equivalent binary
5	101
$\times 6$	$\times 110$
30	000
	101
	$\underline{101}$
	11110

The process for dividing one binary number by another is the same for both binary and decimal numbers, as shown in the following example.

Decimal	Equivalent binary
$\frac{7}{2} \overline{)14}$	$\frac{111}{10} \overline{)1110}$
	$\underline{10}$
	11
	$\underline{10}$
	10
	$\underline{10}$
	00

The basic function of a comparator is to compare the relative magnitude of two quantities. PLC data comparison instructions are used to compare the data stored in two words (or registers). At times, devices may need to be controlled when they are less than, equal to, or greater than other data values or set points used in the application, such as timer and counter values. The basic compare instructions are as follows:

$$\begin{array}{l}
 A = B \text{ (A equals B)} \\
 A > B \text{ (A is greater than B)} \\
 A < B \text{ (A is less than B)}
 \end{array}$$



CHAPTER 3 REVIEW QUESTIONS

- Convert each of the following binary numbers to decimal numbers:
 - 10
 - 100
 - 111
 - 1011
 - 1100
 - 10010
 - 10101
 - 11111
 - 11001101
 - 11100011
- Convert each of the following decimal numbers to binary numbers:
 - 7
 - 19
 - 28
 - 46
 - 57
 - 86
 - 94
 - 112
 - 148
 - 230
- Convert each of the following octal numbers to decimal numbers:
 - 36
 - 104
 - 120
 - 216
 - 360
 - 1516
- Convert each of the following octal numbers to binary numbers:
 - 74
 - 130
 - 250
 - 1510
 - 2551
 - 2634
- Convert each of the following hexadecimal numbers to decimal numbers:
 - 5A
 - C7
 - 9B5
 - 1A6
- Convert each of the following hexadecimal numbers to binary numbers:
 - 4C
 - E8
 - 6D2
 - 31B
- Convert each of the following decimal numbers to BCD:
 - 146
 - 389
 - 1678
 - 2502
- What is the most important characteristic of the Gray code?
- What makes the binary system so applicable to computer circuits?
- Define the following as they apply to the binary memory locations or registers:
 - Bit
 - Byte
 - Word
 - LSB
 - MSB
- State the base used for each of the following number systems:
 - Octal
 - Decimal
 - Binary
 - Hexadecimal
- Define the term *sign bit*.
- Explain the difference between the 1's complement of a number and the 2's complement.
- What is ASCII code?
- Why are parity bits used?
- Add the following binary numbers:
 - $110 + 111$
 - $101 + 011$
 - $1100 + 1011$
- Subtract the following binary numbers:
 - $1101 - 101$
 - $1001 - 110$
 - $10111 - 10010$

18. Multiply the following binary numbers:

- a. 110×110
- b. 010×101
- c. 101×11

19. Divide the following unsigned binary numbers:

- a. $1010 \div 10$
- b. $1100 \div 11$
- c. $110110 \div 10$

CHAPTER 3 PROBLEMS

1. The following binary PLC coded information is to be programmed using the hexadecimal code. Convert each piece of binary information to the appropriate hexadecimal code for entry into the PLC from the keyboard.

- a. 0001 1111
- b. 0010 0101
- c. 0100 1110
- d. 0011 1001

2. The encoder circuit shown in Figure 3-17 is used to convert the decimal digits on the keyboard to a binary code. State the output status (HIGH/LOW) of A-B-C-D when decimal number

- a. 2 is pressed.
- b. 5 is pressed.

- c. 7 is pressed.
- d. 8 is pressed.

3. If the bits of a 16-bit word or register are numbered according to the octal numbering system, beginning with 00, what consecutive numbers would be used to represent each of the bits?

4. Express the decimal number 18 in each of the following number codes:

- a. Binary
- b. Octal
- c. Hexadecimal
- d. BCD

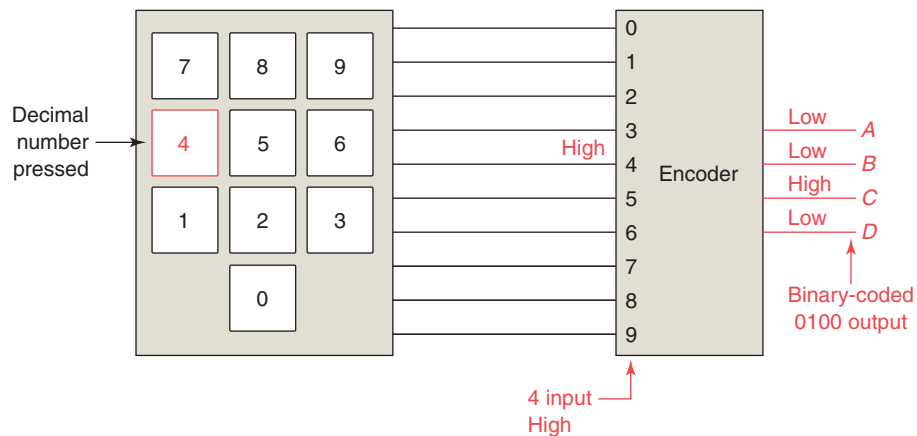


Figure 3-17 Diagram for Problem 2.

4

Fundamentals of Logic



Image Used with Permission of Rockwell Automation, Inc.

Chapter Objectives

After completing this chapter, you will be able to:

- 4.1** Describe the binary concept and the functions of gates
- 4.2** Draw the logic symbol, construct a truth table, and state the Boolean equation for the AND, OR, and NOT functions
- 4.3** Construct circuits from Boolean expressions and derive Boolean equations for given logic circuits
- 4.4** Convert relay ladder schematics to ladder logic programs
- 4.5** Develop elementary programs based on logic gate functions
- 4.6** Program instructions that perform logical operations

This chapter gives an overview of digital logic gates and illustrates how to duplicate this type of control on a PLC. Boolean algebra, which is a shorthand way of writing digital gate diagrams, is discussed briefly. Some small hand-held programmers have digital logic keys, such as AND, OR, and NOT, and are programmed using Boolean expressions.

4.1 The Binary Concept

The PLC, like all digital equipment, operates on the binary principle. The term *binary principle* refers to the idea that many things can be thought of as existing in only one of two states. These states are 1 and 0. The 1 and 0 can represent ON or OFF, open or closed, true or false, high or low, or any other two conditions. The key to the speed and accuracy with which binary information can be processed is that there are only two states, each of which is distinctly different. There is no in-between state so when information is processed the outcome is either yes or no.

A *logic gate* is a circuit with several inputs but only one output that is activated by particular combinations of input conditions. The two-state binary concept, applied to gates, can be the basis for making decisions. The high beam automobile lighting circuit of Figure 4-1 is an example of a logical AND decision. For this application, the high beam light can be turned on only when the light switch AND the high beam switch are closed.

The dome light automobile circuit of Figure 4-2 is an example of a logical OR decision. For this application, the dome light will be turned on whenever the passenger door switch OR the driver door switch is activated.

Logic is the ability to make decisions when one or more different factors must be taken into account before an action is taken. This is the basis for the operation of the PLC, where it is required for a device to operate when certain conditions have been met.

4.2 AND, OR, and NOT Functions

The operations performed by digital equipment are based on three fundamental logic functions: AND, OR, and NOT. Each function has a rule that will determine

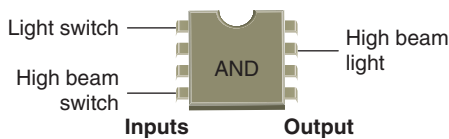


Figure 4-1 The logical AND.

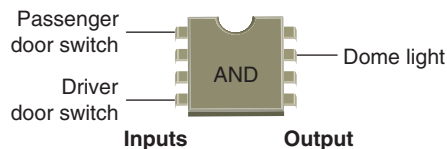


Figure 4-2 The logical OR.

the outcome and a *symbol* that represents the operation. For the purpose of this discussion, the outcome or output is called *Y* and the signal inputs are called *A*, *B*, *C*, and so on. Also, binary 1 represents the presence of a signal or the occurrence of some event, and binary 0 represents the absence of the signal or nonoccurrence of the event.

The AND Function

The symbol drawn in Figure 4-3 is that of an AND gate. An AND gate is a device with two or more inputs and one output. The AND gate output is 1 only if all inputs are 1. The AND truth table in Figure 4-3 shows the resulting output from each of the possible input combinations.

Logic gate *truth tables* show each possible input to the gate or circuit and the resultant output depending upon the combination of the input(s).

Since logic gates are digital ICs (Integrated Circuits) their input and output signals can be in only one of two possible digital states, i.e., logic 0 or logic 1. Thus, the logic state of the output of a logic gate depends on the logic states of each of its individual inputs. Figure 4-4

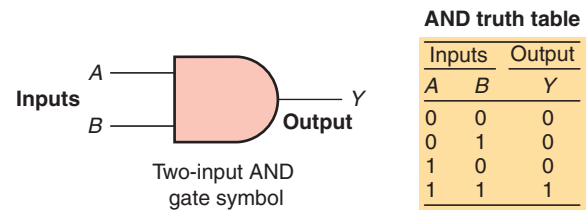


Figure 4-3 AND gate.

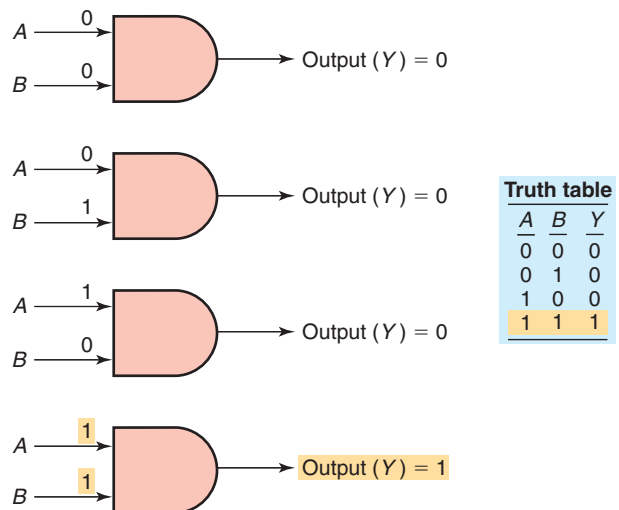


Figure 4-4 AND logic gate digital signal states.

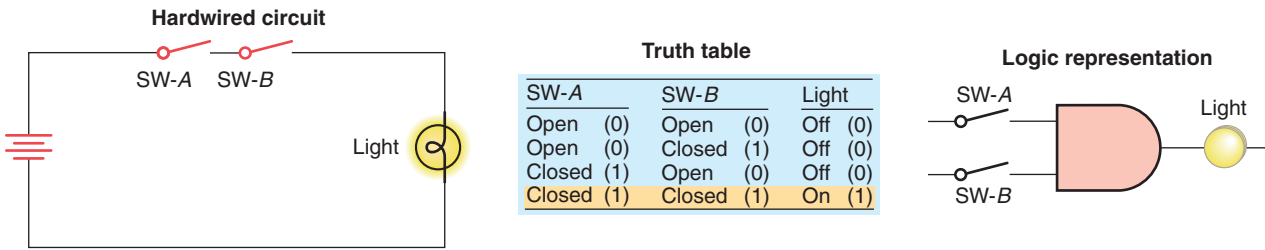


Figure 4-5 AND logic gate operates similarly to control devices connected in series.

illustrates the four possible combinations of inputs for a 2-input AND gate. The basic rules that apply to an AND gate are:

- If all inputs are 1, the output will be 1.
- If any input is 0, the output will be 0.

The AND logic gate operates similarly to control devices connected in *series*, as illustrated in Figure 4-5. The light will be on only when both switch A and switch B are closed.

The OR Function

The symbol drawn in Figure 4-6 is that of an OR gate. An OR gate can have any number of inputs but only one output. The OR gate output is 1 if one or more inputs are 1. The truth table in Figure 4-6 shows the resulting output Y from each possible input combination.

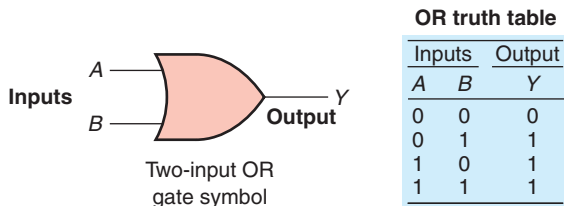


Figure 4-6 OR gate.

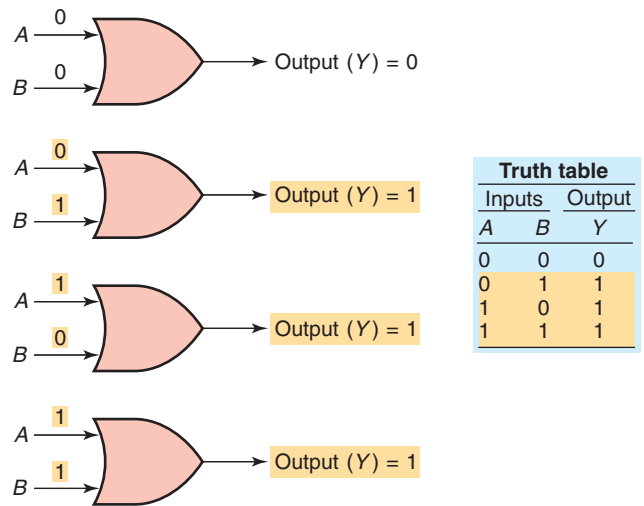


Figure 4-7 OR logic gate digital signal states.

Figure 4-7 illustrates the four possible combinations of inputs for a 2-input OR gate. The basic rules that apply to an OR gate are:

- If one or more inputs are 1, the output is 1.
- If all inputs are 0, the output will be 0.

The OR logic gate operates similarly to control devices connected in *parallel*, as illustrated in Figure 4-8. The light will be on if switch A or switch B or both are closed.

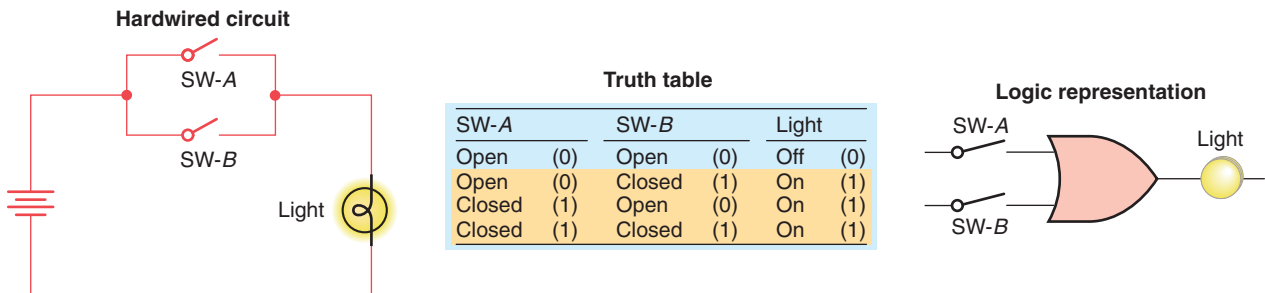


Figure 4-8 OR logic gate operates similarly to control devices connected in parallel.

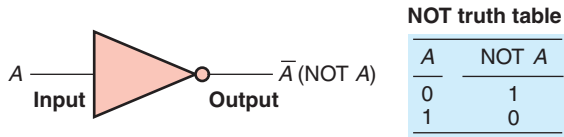


Figure 4-9 NOT function.

The NOT Function

The symbol drawn in Figure 4-9 is that of a NOT function. Unlike the AND and OR functions, the NOT function can have *only one* input. The NOT output is 1 if the input is 0. The output is 0 if the input is 1. The result of the NOT operation is always the inverse of the input, and the NOT function is, therefore, called an *inverter*. The NOT function is often depicted by using a bar across the top of the letter, indicating an inverted output. The small circle at the output of the inverter is termed a state indicator and indicates that an inversion of the logical function has taken place.

The logical NOT function can be performed on a contact input simply by using a normally closed instead of a normally open contact. Figure 4-10 shows an example

of the NOT function constructed using a normally closed pushbutton in series with a lamp. When the input pushbutton is *not* actuated, the output lamp is ON. When the input pushbutton is actuated, the output lamp switches OFF.

The NOT function is most often used in conjunction with the AND or the OR gate. Figure 4-11 shows the NOT function connected to one input of an AND gate for a low-pressure indicator circuit. If the power is on (1) and the pressure switch is not closed (0), the warning light will be on (1).

The NOT symbol placed at the output of an AND gate would invert the normal output result. An AND gate with an inverted output is called a *NAND* gate. The NAND gate symbol and truth table are shown in Figure 4-12. The NAND function is often used in integrated circuit logic arrays and can be used in programmable controllers to solve complex logic.

The same rule about inverting the normal output result applies if a NOT symbol is placed at the output of the OR gate. The normal output is inverted, and the function is referred to as a *NOR* gate. The NOR gate symbol and truth table are shown in Figure 4-13.

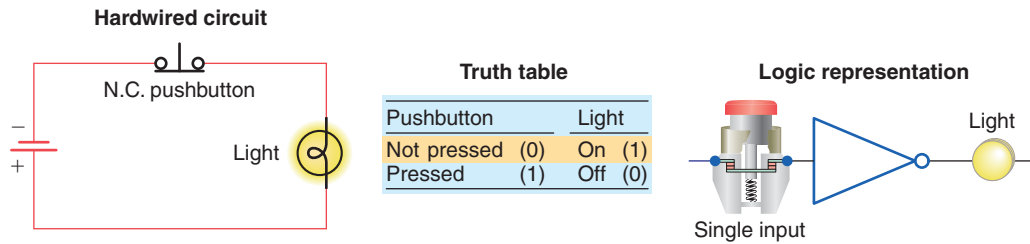
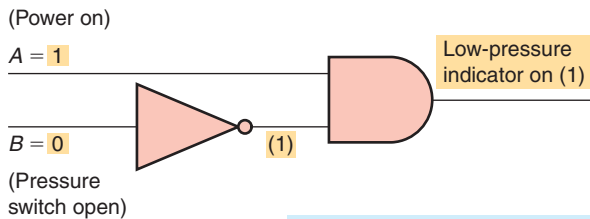


Figure 4-10 NOT function constructed using a normally closed pushbutton.



Pressure switch	Power	Pressure indicator
0	1	1
1	1	0

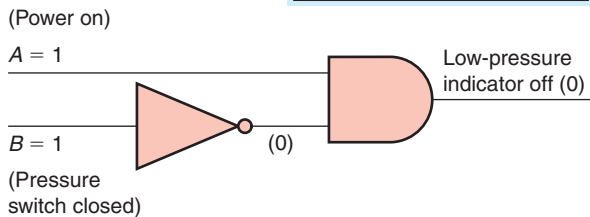


Figure 4-11 NOT function is most often used in conjunction with an AND gate.

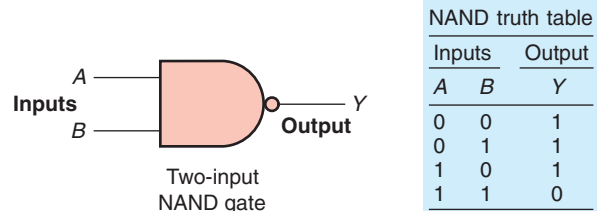


Figure 4-12 NAND gate symbol and truth table.

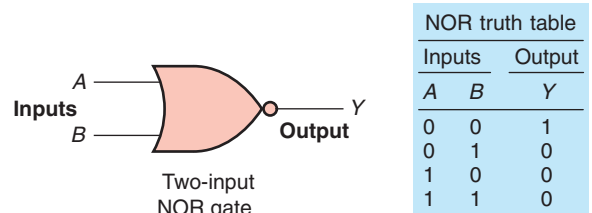


Figure 4-13 NOR gate symbol and truth table.

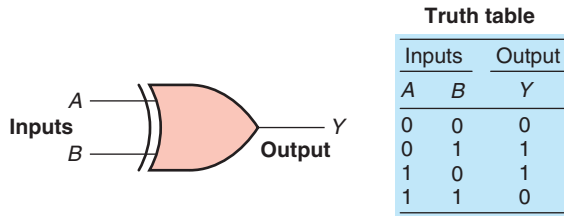


Figure 4.14 The XOR gate symbol and truth table.

The Exclusive-OR (XOR) Function

An often-used combination of gates is the exclusive-OR (XOR) function. The XOR gate symbol and truth table are shown in Figure 4-14. The output of this circuit is HIGH

only when one input or the other is HIGH, but not both. The exclusive-OR gate is commonly used for the *comparison* of two binary numbers.

4.3 Boolean Algebra

The mathematical study of the binary number system and logic is called *Boolean algebra*. The purpose of this algebra is to provide a simple way of writing complicated combinations of logic statements. There are many applications where Boolean algebra could be applied to solving PLC programming problems.

A typical Boolean instruction list (also referred to as a statement list) is shown in Table 4-1. The instructions

Table 4-1 Typical Boolean Instruction or Statement List

Boolean Instruction and Function	Graphic Symbol
Store (STR)–Load (LD) Begins a new rung or an additional branch in a rung with a normally open contact.	
Store Not (STR NOT)–Load Not (LD NOT) Begins a new rung or an additional branch in a rung with a normally closed contact.	
Or (OR) Logically ORs a normally open contact in parallel with another contact in a rung.	
Or Not (OR NOT) Logically ORs a normally closed contact in parallel with another contact in a rung.	
And (AND) Logically ANDs a normally open contact in series with another contact in a rung.	
And Not (AND NOT) Logically ANDs a normally closed contact in series with another contact in a rung.	
And Store (AND STR)–And Load (AND LD) Logically ANDs two branches of a rung in series.	
Or Store (OR STR)–Or Load (OR LOAD) Logically ORs two branches of a rung in parallel.	
Out (OUT) Reflects the status of the rung (on/off) and outputs the discrete (ON/OFF) state to the specified image register point or memory location.	
Or Out (OR OUT) Reflects the status of the rung and outputs the discrete (ON/OFF) state to the image register. Multiple OR OUT instructions referencing the same discrete point can be used in the program.	
Output Not (OUT NOT) Reflects the status of the rung and turns the output OFF for an ON execution condition; turns the output ON for an OFF execution condition.	

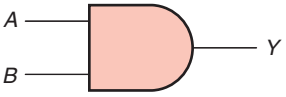
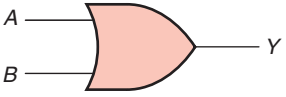
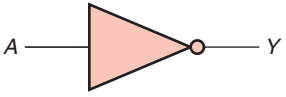
Logic symbol	Logic statement	Boolean equation	Boolean notations												
	Y is 1 if A and B are 1	$Y = A \cdot B$ or $Y = AB$	<table border="0"> <tr> <td>Symbol</td> <td>Meaning</td> </tr> <tr> <td>•</td> <td>and</td> </tr> <tr> <td>+</td> <td>or</td> </tr> <tr> <td>–</td> <td>not</td> </tr> <tr> <td>◦</td> <td>invert</td> </tr> <tr> <td>=</td> <td>result in</td> </tr> </table>	Symbol	Meaning	•	and	+	or	–	not	◦	invert	=	result in
Symbol	Meaning														
•	and														
+	or														
–	not														
◦	invert														
=	result in														
	Y is 1 if A or B is 1	$Y = A + B$													
	Y is 1 if A is 0 Y is 0 if A is 1	$Y = \bar{A}$													

Figure 4-15 Boolean algebra as related to AND, OR, and NOT functions.

are based on the basic Boolean operators of AND, OR, and NOT. Although these instructions are programmed in list format similar to BASIC and other text languages, they implement the same logic as relay ladder logic.

Figure 4-15 summarizes the basic operators of Boolean algebra as they relate to the basic AND, OR, and

NOT functions. Inputs are represented by capital letters A, B, C, and so on, and the output by a capital Y. The multiplication sign (×) or dot (•) represents the AND operation, an addition sign (+) represents the OR operation, the circle with an addition sign ⊕ represents the exclusive-OR operation, and a bar over the letter \bar{A} represents the NOT operation. The Boolean equations are used to express the mathematical function of the logic gate.

PLC digital systems may be designed using Boolean algebra. Circuit functions are represented by Boolean equations. Figure 4-16 illustrates how logic operators AND, NAND, OR, NOR, and NOT are used singly to form logical statements. Figure 4-17 illustrates how basic logic operators are used in combination to form Boolean equations.

An understanding of the technique of writing simplified Boolean equations for complex logical statements is a useful tool when creating PLC control programs. Some laws of Boolean algebra are different from those of ordinary algebra. These three basic laws illustrate the close comparison between Boolean algebra and

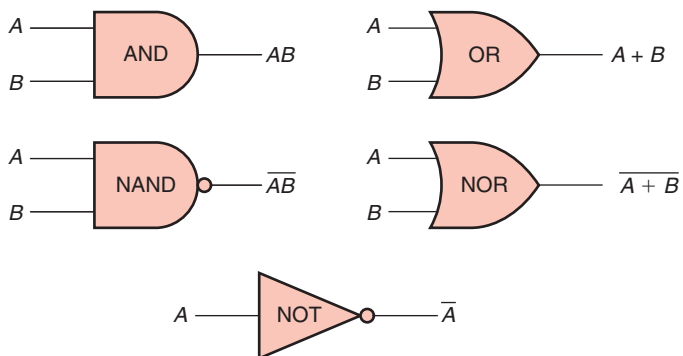


Figure 4-16 Logic operators used singly to form logical statements.

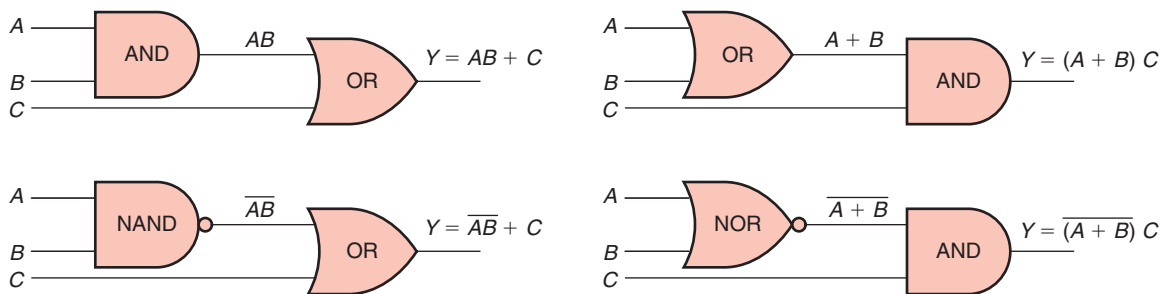


Figure 4-17 Logic operators used in combination to form Boolean equations.

ordinary algebra, as well as one major difference between the two:

COMMUTATIVE LAW

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$

ASSOCIATIVE LAW

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

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

DISTRIBUTIVE LAW

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

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

This law holds true only in Boolean algebra.

4.4 Developing Logic Gate Circuits from Boolean Expressions

As logic gate circuits become more complex, the need to express these circuits in Boolean form becomes greater. Figure 4-18 shows a logic gate circuit developed from the Boolean expression $Y = AB + C$. The procedure is as follows:

Boolean expression: $Y = AB + C$

Gates required: (by inspection)

- 1 - AND gate with input A and B
- 1 - OR gate with input C and output from previous AND gate

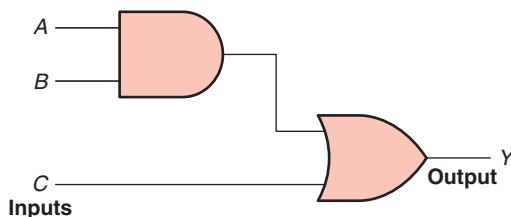


Figure 4-18 Logic gate circuit developed from the Boolean expression $Y = AB + C$.

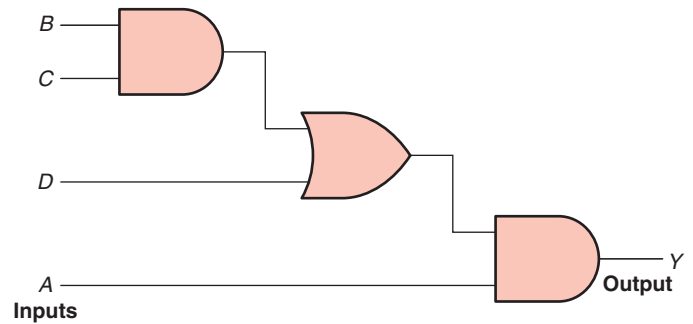


Figure 4-19 Logic gate circuit developed from the Boolean expression $Y = A(BC + D)$.

Figure 4-19 shows a logic gate circuit developed from the Boolean expression $Y = A(BC + D)$. The procedure is as follows:

Boolean expression: $Y = A(BC + D)$

Gates required: (by inspection)

- 1 - AND gate with input B and C
- 1 - OR gate with inputs B, C, and D
- 1 - AND gate with inputs A and the output from the OR gate

4.5 Producing the Boolean Equation for a Given Logic Gate Circuit

A simple logic gate is quite straightforward in its operation. However, by grouping these gates into combinations, it becomes more difficult to determine which combinations of inputs will produce an output. The Boolean equation for the logic circuit of Figure 4-20 is determined as follows:

- The output of the OR gate is $A + B$
- The output of the inverter is \bar{D}
- Based on the input combination applied to the AND gate the Boolean equation for the circuit is $Y = C \bar{D}(A + B)$

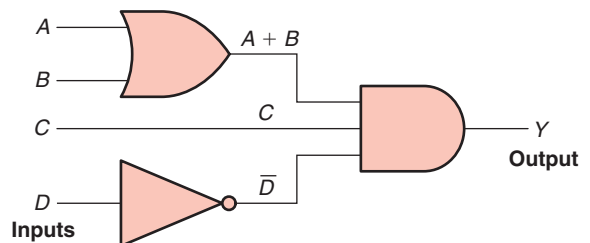


Figure 4-20 Determining the Boolean equation for a logic circuit.

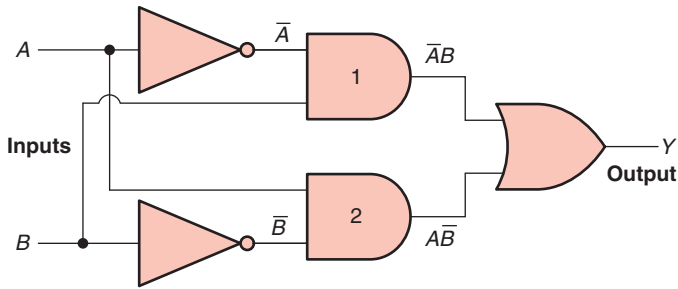


Figure 4-21 Determining the Boolean equation for a logic circuit.

The Boolean equation for the logic circuit of Figure 4-21 is determined as follows:

- The output of AND gate 1 is $\bar{A}B$
- The output of AND gate 2 is $A\bar{B}$
- Based on the combination of inputs applied to the OR gate the Boolean equation for the circuit is $Y = \bar{A}B + A\bar{B}$

4.6 Hardwired Logic versus Programmed Logic

The term *hardwired logic* refers to logic control functions that are determined by the way devices are electrically interconnected. Hardwired logic can be implemented using relays and relay ladder schematics. Relay ladder schematics are universally used and understood in industry. Figure 4-22 shows a typical relay ladder schematic of a motor stop/start control station with pilot lights. The control scheme is drawn between two vertical supply lines. All the components are placed between these two lines,

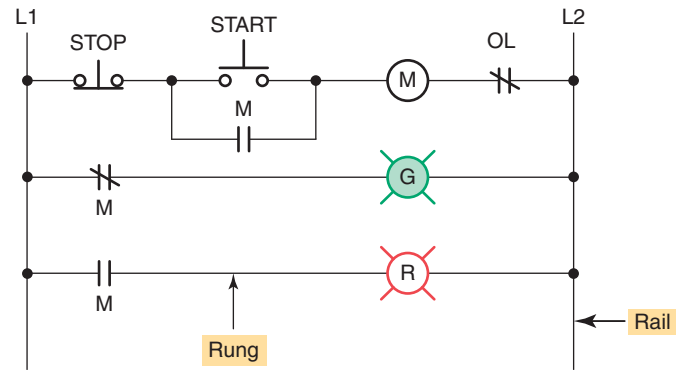


Figure 4-22 Motor stop/start relay ladder schematic.

called rails or legs, connecting the two power lines with what look like rungs of a ladder—thus the name, relay ladder schematic.

Hardwired logic is fixed; it is changeable only by altering the way devices are electrically interconnected. In contrast, programmable control is based on the basic logic functions, which are programmable and easily changed. These functions (AND, OR, NOT) are used either singly or in combinations to form instructions that will determine if a device is to be switched on or off. The form in which these instructions are implemented to convey commands to the PLC is called the *language*. The most common PLC language is *ladder logic*. Figure 4-23 shows a typical *ladder logic program* for the motor start/stop circuit. The instructions used are the relay equivalent of normally open (NO) and normally closed (NC) contacts and coils.

PLC contact symbolism is a simple way of expressing the control logic in terms of symbols. These symbols are

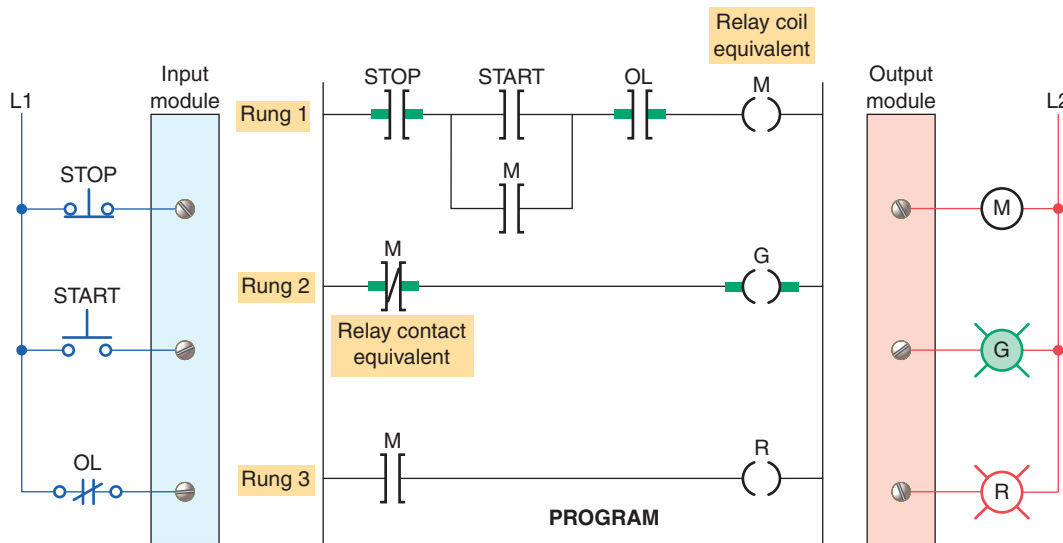
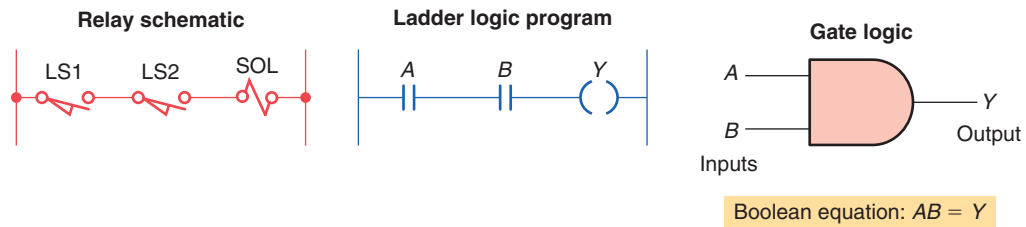


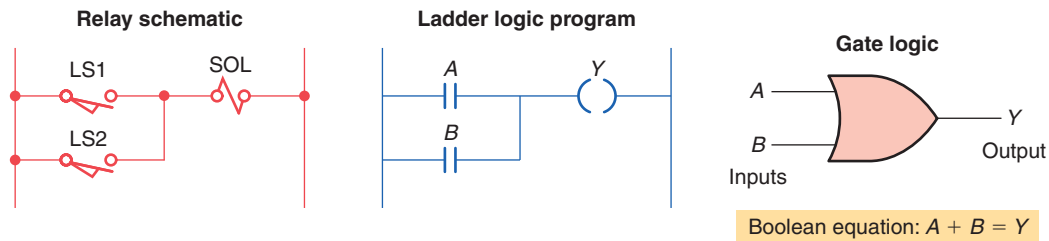
Figure 4-23 Motor stop/start ladder logic program.

basically the same as those used for representing hard-wired relay control circuits. A rung is the contact symbolism required to control an output. Some PLCs allow a rung to have multiple outputs while others allow only one output per rung. A complete ladder logic program then consists of several rungs, each of which controls an output. In programmed logic all mechanical switch contacts are represented by a software contact symbol and all electromagnetic coils are represented by a software coil symbol.

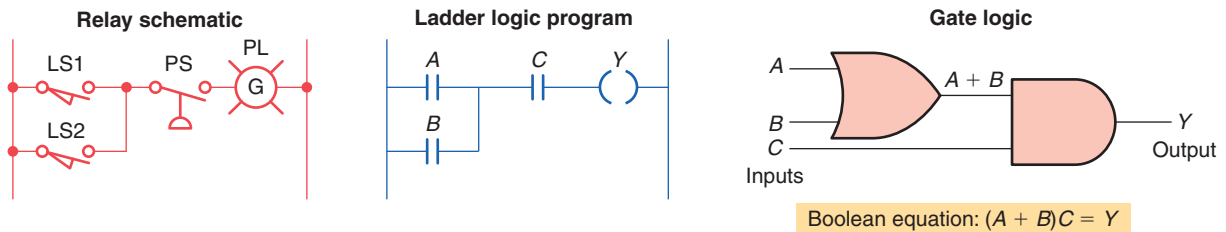
Because the PLC uses ladder logic diagrams, the conversion from any existing relay logic to programmed logic is simplified. Each rung is a combination of input conditions (symbols) connected from left to right, with the symbol that represents the output at the far right. The symbols that represent the inputs are connected in series, parallel, or some combination of the two to obtain the desired logic. The following group of examples illustrates the relationship between the relay ladder schematic, the ladder logic program, and the equivalent logic gate circuit.



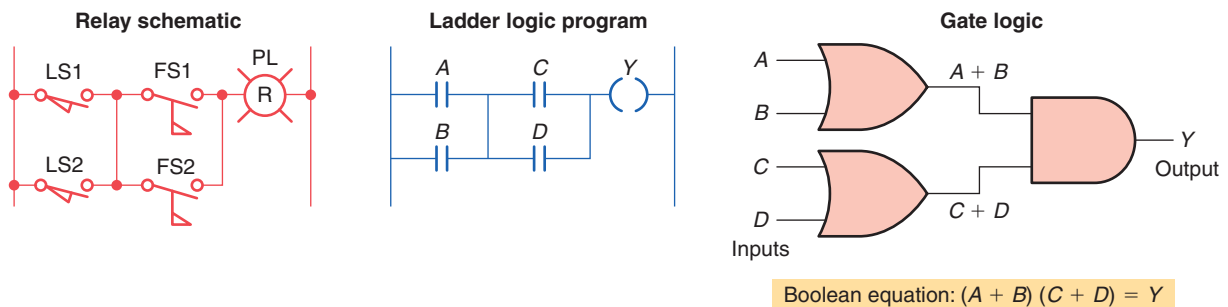
Example 4-1 Two limit switches connected in series and used to control a solenoid valve.



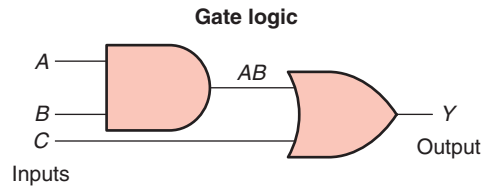
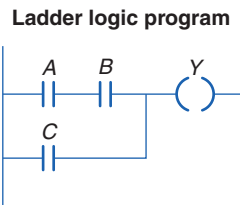
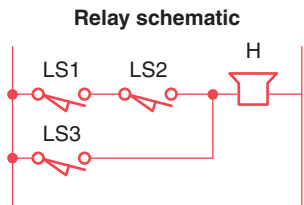
Example 4-2 Two limit switches connected in parallel and used to control a solenoid valve.



Example 4-3 Two limit switches connected in parallel with each other and in series with a pressure switch.

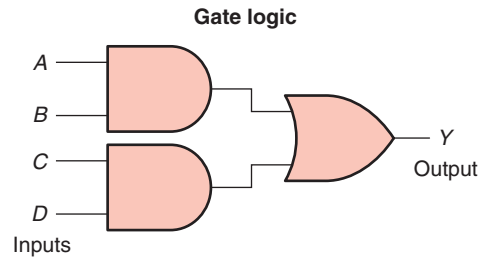
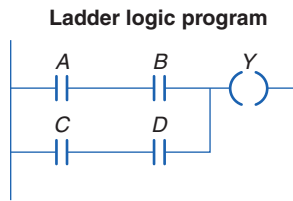
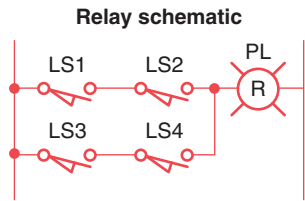


Example 4-4 Two limit switches connected in parallel with each other and in series with two sets of flow switches (that are connected in parallel with each other), and used to control a pilot light.



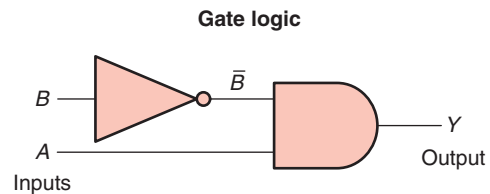
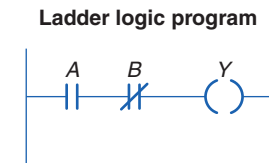
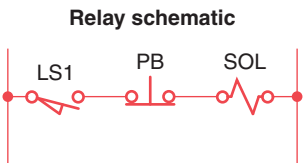
Boolean equation: $(AB) + C = Y$

Example 4-5 Two limit switches connected in series with each other and in parallel with a third limit switch, and used to control a warning horn.



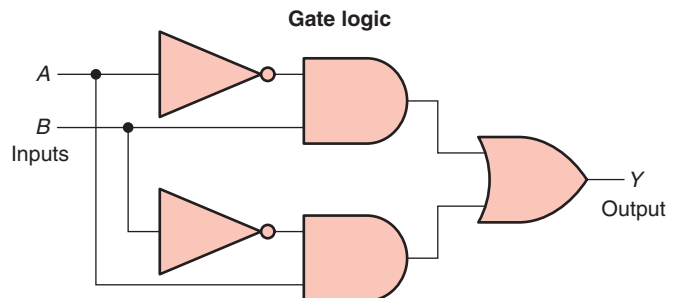
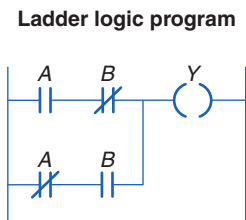
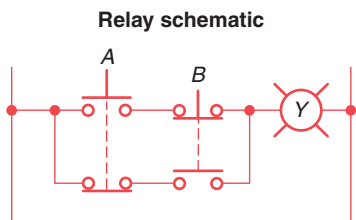
Boolean equation: $(AB) + (CD) = Y$

Example 4-6 Two limit switches connected in series with each other and in parallel with two other limit switches (that are connected in series with each other), and used to control a pilot light.



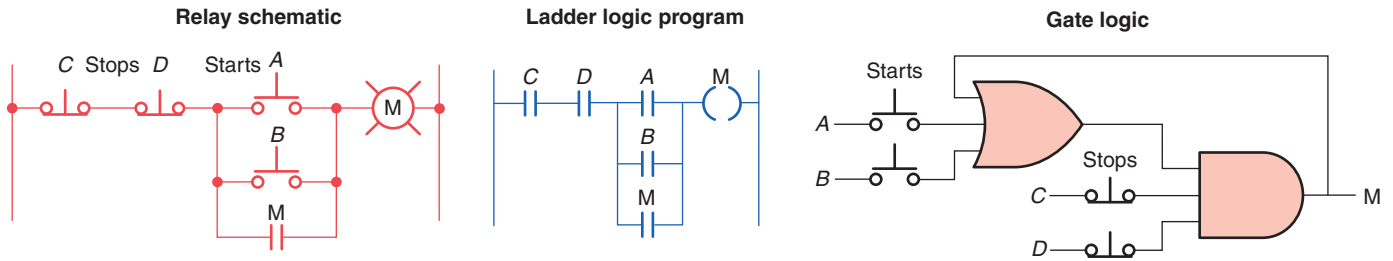
Boolean equation: $A\bar{B} = Y$

Example 4-7 One limit switch connected in series with a normally closed pushbutton and used to control a solenoid valve. This circuit is programmed so that the output solenoid will be turned on when the limit switch is closed and the pushbutton is *not pushed*.



Boolean equation: $\bar{A}B + A\bar{B} = Y$
 $A \oplus B = Y$

Example 4-8 Exclusive-OR circuit. The output lamp of this circuit is ON only when pushbutton A or B is pressed, but not both. This circuit has been programmed using only the normally open A and B pushbutton contacts as the inputs to the program.



Example 4-9 A motor control circuit with two start/stop buttons. When either start button is depressed, the motor runs. By use of a seal-in contact, it continues to run when the start button is released. Either stop button stops the motor when it is depressed.

4.7 Programming Word Level Logic Instructions

Most PLCs provide word-level logic instructions as part of their instruction set. Table 4-2 shows how to select the correct word logic instruction for different situations.

Figure 4-24 illustrates the operation of the AND instruction to perform a word-level AND operation using the bits in the two source addresses. This instruction tells the processor to perform an AND operation on B3:5 and

Table 4-2 Selecting Logic Instructions

If you want to use this instruction.
Know when matching bits in two different words are both ON	AND
Know when one or both matching bits in two different words are ON	OR
Know when one or the other bit of matching bits in two different words is ON	XOR
Reverse the state of bits in a word	NOT

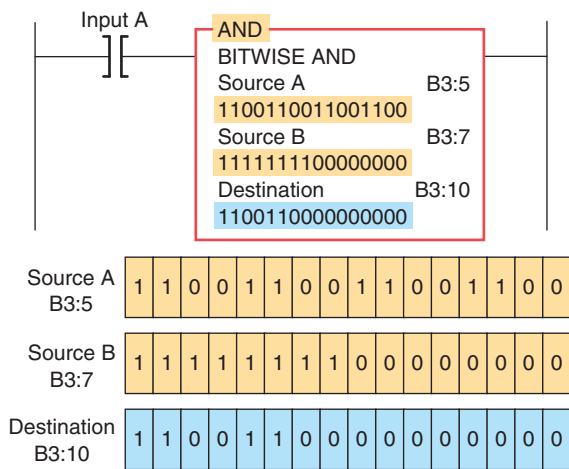


Figure 4-24 Word-level AND instruction.

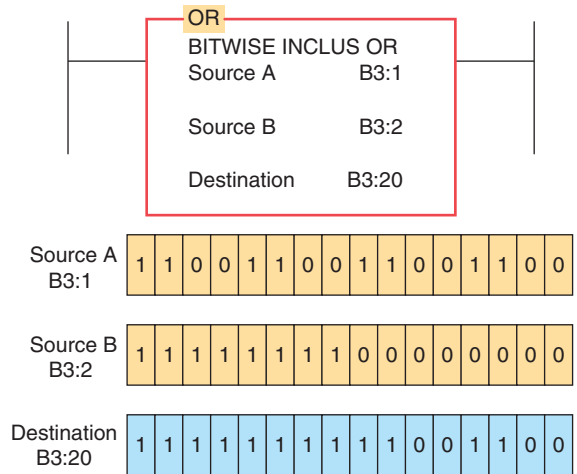


Figure 4-25 Word-level OR instruction.

B3:7 and to store the result in destination B3:10 when input device A is true. The destination bits are a result of the logical AND operation.

Figure 4-25 illustrates the operation of a word-level OR instruction, which ORs the data in Source A, bit by bit, with the data in Source B and stores the result at the destination address. The address of Source A is B3:1, the address of Source B is B3:2, and the destination address is B3:20. The instruction may be programmed conditionally, with input instruction(s) preceding it, or unconditionally, as shown, without any input instructions preceding it.

Figure 4-26 illustrates the operation of a word-level XOR instruction. In this example, data from input I:1.0 are compared, bit by bit, with data from input I:3.0. Any mismatches energize the corresponding bit in word O:4.0. As you can see, there is a 1 in every bit location in the destination corresponding to the bit locations where Source A and Source B are different, and a 0 in the destination where Source A and Source B are the same. The XOR is often used in diagnostics, where real-world inputs, such as rotary cam limit switches, are compared with their desired states.

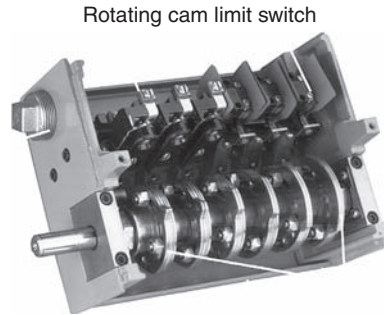


Figure 4-26 Word-level XOR instruction.
Source: Image Used with Permission of Rockwell Automation, Inc.

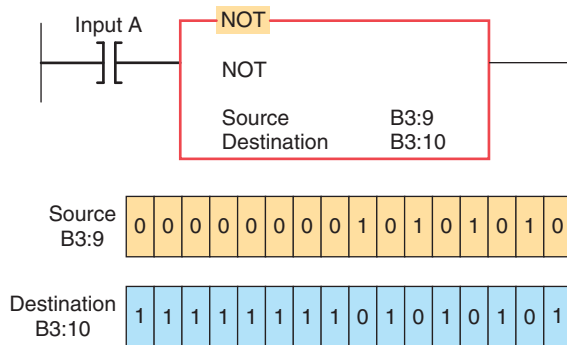


Figure 4-27 illustrates the operation of a word-level NOT instruction. This instruction inverts the bits from the source word to the destination word. The bit pattern in B3:10 is the result of the instruction being true and is the inverse of the bit pattern in B3:9.

For 32-bit PLCs, such as the Allen-Bradley Control-Logix controller, the source and destination may be a SINT (one-byte integer), INT (two-byte integer), DINT (four-byte integer), or REAL (four-byte floating decimal point) value.

Figure 4-27 Word-level NOT operation.



CHAPTER 4 REVIEW QUESTIONS

- Explain the binary principle.
- What is a logic gate?
- Draw the logic symbol, construct a truth table, and state the Boolean equation for each of the following:
 - Two-input AND gate
 - NOT function
 - Three-input OR gate
 - XOR function
- Express each of the following equations as a ladder logic program:
 - $Y = (A + B)CD$
 - $Y = A\bar{B}C + \bar{D} + E$
 - $Y = [(\bar{A} + \bar{B})C] + DE$
 - $Y = (\bar{A}\bar{B}C) + (D\bar{E}F)$
- Write the ladder logic program, draw the logic gate circuit, and state the Boolean equation for the two relay ladder diagrams in Figure 4-28.
- Develop a logic gate circuit for each of the following Boolean expressions using AND, OR, and NOT gates:
 - $Y = ABC + D$
 - $Y = AB + CD$
 - $Y = (A + B)(\bar{C} + D)$

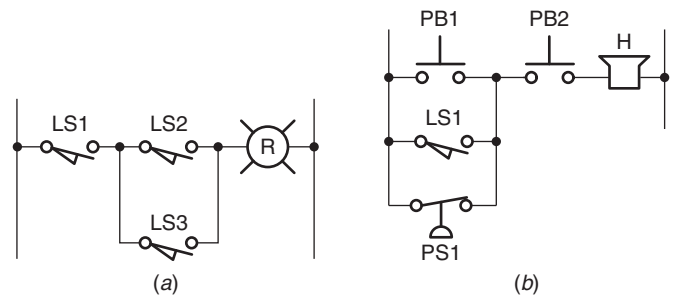


Figure 4-28 Question 5 relay ladder diagrams.

- $Y = \bar{A}(B + CD)$
 - $Y = \bar{A}B + C$
 - $Y = (ABC + D)(\bar{E}\bar{F})$
- State the logic instruction you would use when you want to:
 - Know when one or both matching bits in two different words are 1.
 - Reverse the state of bits in a word.
 - Know when matching bits in two different words are both 1.
 - Know when one or the other bit of matching bits, but not both, in two different words is 1.



CHAPTER 4 PROBLEMS

- It is required to have a pilot light come on when all of the following circuit requirements are met:
 - All four circuit pressure switches must be closed.
 - At least two out of three circuit limit switches must be closed.
 - The reset switch must not be closed.
 Using AND, OR, and NOT gates, design a logic circuit that will solve this hypothetical problem.
- Write the Boolean equation for each of the logic gate circuits in Figure 4-29a–f.
- The logic circuit of Figure 4-30 is used to activate an alarm when its output Y is logic HIGH or 1. Draw a truth table for the circuit showing the resulting output for all 16 of the possible input conditions.

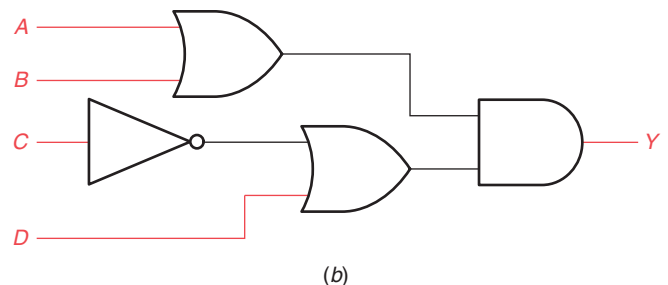
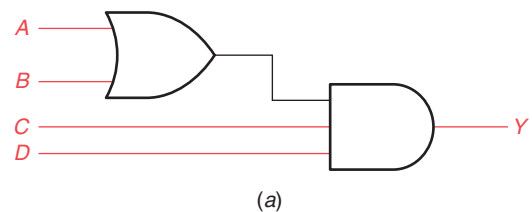


Figure 4-29 Logic gate circuits for Problem 2.

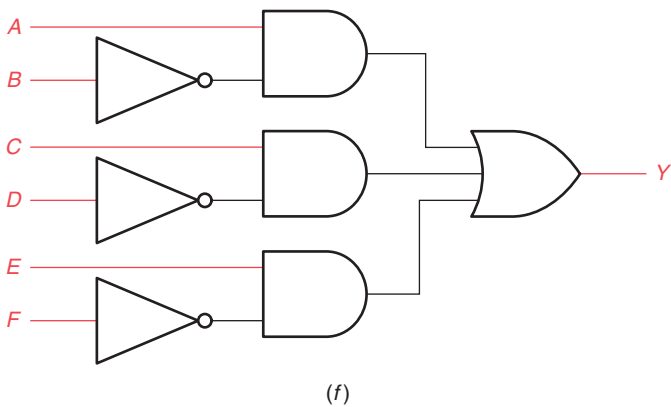
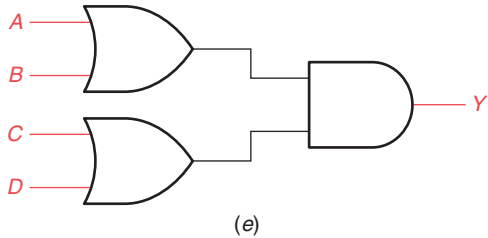
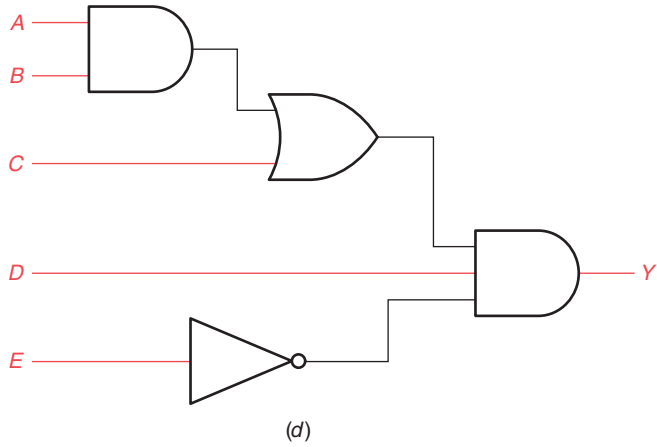
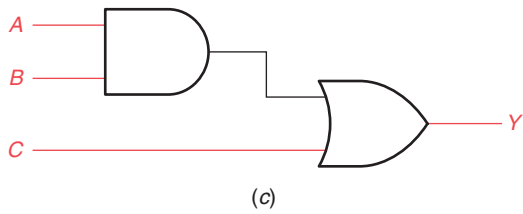


Figure 4-29 (Continued)

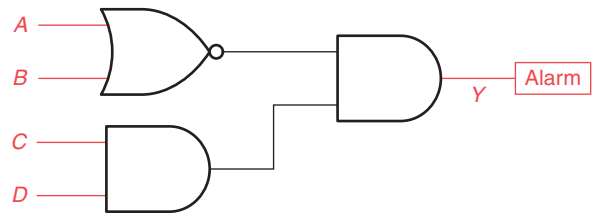


Figure 4-30 Logic circuit for Problem 3.

Source A	0	0	0	0	0	0	0	0	1	0	1	0	1	0	
Source B	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1
Destination															

Figure 4-31 Data for Problem 4.

4. What will be the data stored in the destination address of Figure 4-31 for each of the following logical operations?
 - a. AND operation
 - b. OR operation
 - c. XOR operation
5. Write the Boolean expression and draw the gate logic diagram and typical PLC ladder logic diagram for a control system wherein a fan is to run only when all of the following conditions are met:
 - Input A is OFF
 - Input B is ON or input C is ON, or both B and C are ON
 - Inputs D and E are both ON
 - One or more of inputs F, G, or H are ON

5

Basics of PLC Programming



Image Used with Permission of Rockwell Automation, Inc.

Chapter Objectives

After completing this chapter, you will be able to:

- 5.1** Define and identify the functions of a PLC memory map
- 5.2** Describe input and output image table files and types of data files
- 5.3** Describe the PLC program scan sequence
- 5.4** Understand how ladder diagram language, Boolean language, and function chart programming language are used to communicate information to the PLC
- 5.5** Define and identify the function of internal relay instructions
- 5.6** Identify the common operating modes found in PLCs
- 5.7** Write and enter ladder logic programs

Each input and output PLC module terminal is identified by a unique address. In PLCs, the internal symbol for any input is a contact. Similarly, in most cases, the internal PLC symbol for all outputs is a coil. This chapter shows how these contact/coil functions are used to program a PLC for circuit operation. This chapter covers only the basic set of instructions that perform functions similar to relay functions. You will also learn more about the program scan cycle and the scan time of a PLC.

5.1 Processor Memory Organization

While the fundamental concepts of PLC programming are common to all manufacturers, differences in memory organization, I/O addressing, and instruction set mean that PLC programs are never perfectly interchangeable among different makers. Even within the same product line of a single manufacturer, different models may not be directly compatible.

The memory map or structure for a PLC processor consists of several areas, some of these having specific roles. Allen-Bradley PLCs have two different memory structures identified by the terms *rack-based* systems and *tag-based* systems. The memory organization for rack-based systems will be covered in this chapter and that for tag-based systems in a later chapter.

Memory organization takes into account the way a PLC divides the available memory into different sections. The memory space can be divided into two broad categories: *program files* and *data files*. Individual sections, their order, and the sections' length will vary and may be fixed or variable, depending on the manufacturer and model.

Program files are the part of the processor memory that stores the user ladder logic program. The program accounts for most of the total memory of a given PLC system. It contains the ladder logic that controls the machine operation. This logic consists of instructions that are programmed in a ladder logic format. Most instructions require one word of memory.

The data files store the information needed to carry out the user program. This includes information such as the status of input and output devices, timer and counter values, data storage, and so on. Contents of the data table can be divided into two categories: status data and numbers or codes. Status is ON/OFF type of information represented by 1s and 0s, stored in unique bit locations. Number or code information is represented by groups of bits that are stored in unique byte or word locations.

The memory organizations of the rack-based Allen-Bradley PLC-5 and SLC 500 controllers are very similar. Figure 5-1 shows the program and data file organization for the SLC 500 controller. The contents of each file are as follows.

Program Files

Program files are the areas of processor memory where ladder logic programming is stored. They may include:

- **System functions (file 0)**—This file is always included and contains various system-related information and user-programmed information such as processor type, I/O configuration, processor file name, and password.

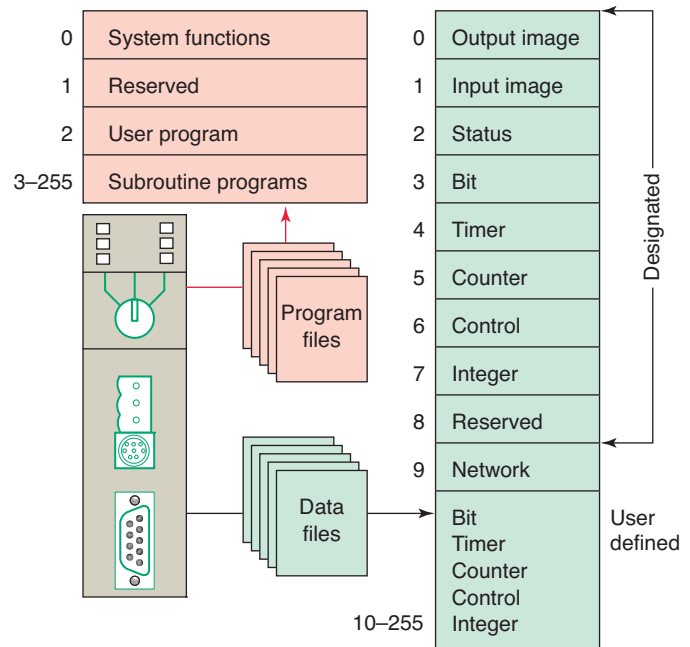


Figure 5-1 Program and data file organization for the SLC 500 controller.

- **Reserved (file 1)**—This file is reserved by the processor and is not accessible to the user.
- **Main ladder program (file 2)**—This file is always included and contains user-programmed instructions that define how the controller is to operate.
- **Subroutine ladder program (files 3–255)**—These files are user-created and are activated according to subroutine instructions residing in the main ladder program file.

Data Files

The data file portion of the processor's memory stores input and output status, processor status, the status of various bits, and numerical data. All this information is accessed via the ladder logic program. These files are organized by the type of data they contain and may include:

- **Output (file 0)**—This file stores the state of the output terminals for the controller.
- **Input (file 1)**—This file stores the status of the input terminals for the controller.
- **Status (file 2)**—This file stores controller operation information and is useful for troubleshooting controller and program operation.
- **Bit (file 3)**—This file is used for internal relay logic storage.
- **Timer (file 4)**—This file stores the timer accumulated and preset values and status bits.

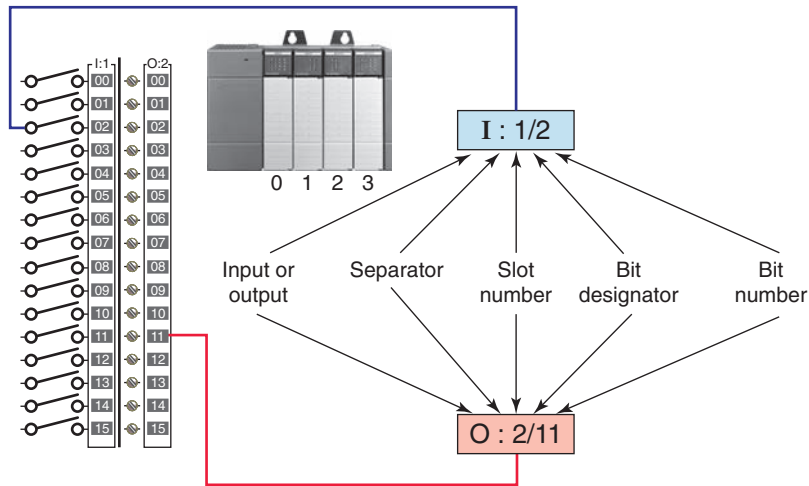


Figure 5-2 I/O address format for the SLC family of PLCs.

Source: Image Used with Permission of Rockwell Automation, Inc.

- **Counter (file 5)**—This file stores the counter accumulated and preset values and status bits.
- **Control (file 6)**—This file stores the length, pointer position, and status bit for specific instructions such as shift registers and sequencers.
- **Integer (file 7)**—This file is used to store numerical values or bit information.
- **Reserved (file 8)**—This file is not accessible to the user.
- **Network communications (file 9)**—This file is used for network communications if installed or used like files 10–255.
- **User-defined (files 10–255)**—These files are user-defined as bit, timer, counter, control, and/or integer data storage.

The I/O address format for the SLC family of PLCs is shown in Figure 5-2. The format consists of the following three parts:

Part 1: I for input, and a colon to separate the module type from the slot.

O for output and a colon to separate the module type from the slot.

Part 2: The module slot number and a forward slash to separate the slot from the terminal screw.

Part 3: The screw terminal number.

There are about 1000 program files for an Allen-Bradley PLC-5 controller. These program files may be set up in two ways: either (1) standard ladder logic programming, with the main program in program file 2 and program files 3 through 999 assigned, as needed, to subroutines; or (2) in sequential function charts in which files 2 through 999 are assigned steps or transitions, as required. With the

processor set up for standard ladder logic, the main program will always be in program file 2, and program files 3 through 999 will be subroutines. In either case, the processor can store and execute only one program at a time.

Figure 5-3 shows a typical data file memory organization for an Allen-Bradley PLC-5 controller. Each data file is made up of numerous *elements*. Each element may be one, two, or three words in length. Timer, counter, and control elements are three words in length; floating-point elements are two words in length; and all other elements are a single word in length. A *word* consists of 16 bits, or binary digits. The processor operates on two different data types: integer and floating point. All data types, except the floating-point files, are treated as integers or whole numbers. All element and bit addresses in the output and input data files are numbered octally. Element and bit addresses in all other data files are numbered decimally.

The PLC-5 and SLC 500 store all data in global data tables and are based on 16-bit operations. You access these data by specifying the address of the data you want. Typical addressing formats for the PLC-5 controller are as follows:

- The addresses in the output data file and the input data file are potential locations for either input modules or output modules mounted in the I/O chassis:
 - The address O:012/15 is in the output image table file, rack 1, I/O group 2, bit 15.
 - The address I:013/17 is in the input image table file, rack 1, I/O group 3, bit 17.
- The **status data file** contains information about the processor status:
 - The address S:015 addresses word 15 of the status file.
 - The address S:027/09 addresses bit 9 in word 27 of the status file.



Address range		Size, in elements
O:000		
O:037	Output image file	32
I:000		
I:037	Input image file	32
S:000		
S:031	Processor status	32
B3:000		
B3:999	Bit file	1–1000
T4:000		
T4:999	Timer file	1–1000
C5:000		
C5:999	Counter file	1–1000
R6:000		
R6:999	Control file	1–1000
N7:000		
N7:999	Integer file	1–1000
F8:000		
F8:999	Floating-point file	1–1000
	Files to be assigned file nos. 9–999	1–1000 per file

Figure 5-3 Data file memory organization for an Allen-Bradley PLC-5 controller.
Source: Image Used with Permission of Rockwell Automation, Inc.

- The **bit data file** stores bit status. It frequently serves for storage when using internal outputs, sequencers, bit-shift instructions, and logical instructions:
 - The address B3:400 addresses word 400 of the bit file. The file number (3) must be included as part of the address. Note that the input, output, and status data files are the only files that do not require the file number designator because there can only be one input data, one output data, and one status data file.
 - Word 2, bit 15 is addressed as B3/47 because bit numbers are always measured from the beginning of the file. Remember that here, bits are numbered decimally (not octally, as the word representing the rack and slot).
- The **timer file** stores the timer status and timer data. A timer element consists of three words: the control word, preset word, and accumulated word. The addressing of the timer control word is the assigned timer number. Timers in file 4 are numbered starting with T4:0 and running through T4:999. The addresses for the three timer words in timer T4:0 are:

The enable-bit address in the control word is T4:0/EN, the timer-timing-bit address is T4:0/TT, and the done-bit address is T4:0/DN.

- The **counter file** stores the counter status and counter data. A counter element consists of three words: the control word, preset word, and accumulated word. The addressing of the counter control is the assigned counter number. Counters in file 5 are numbered beginning with C5:0 and running through C5:999. The addresses for the three counter words in counter C5:0 are:

Control word:	C5:0
Preset word:	C5:0.PRE
Accumulated word:	C5:0.ACC

The count-up-enable-bit address in the control word is C5:0/CU, the count-down-enable-bit address is C5:0/CD, the done-bit address is C5:0/DN, the overflow address is C5:0/OV, and the underflow address is C5:0/UN.

- The **control file** stores the control element's status and data, and it is used to control various file instructions. The control element consists of three words: the control word, length word, and position word. The addressing of the control's control word

Control word:	T4:0
Preset word:	T4:0.PRE
Accumulated word:	T4:0.ACC

is the assigned control number. Control elements in control file 6 are numbered beginning with R6:0 and running through R6:999. The addresses for the three words in control element R6:0 are:

Control word:	R6:0
Length:	R6:0.LEN
Position:	R6:0.POS

There are numerous control bits in the control word, and their function depends on the instruction in which the control element is used.

- The **integer file** stores integer data values, with a range from $-32,768$ through $32,767$. Stored values are displayed in decimal form. The integer element is a single-word (16-bit) element. As many as 1000 integer elements, addressed from N7:000 through N7:999, can be stored.
 - The address N7:100 addresses word 100 of the integer file.
 - Bit addressing is decimal, from 0 through 15. For example, bit 12 in word 15 is addressed N7:015/12.
- The **floating-point file** element can store values in the range from $\pm 1.1754944e-38$ to $\pm 3.4028237e+38$. The floating-point element is a two-word (32-bit) element. As many as 1000 elements, addressed from F8:000 through F8:999, can be stored. Individual words or bits cannot be addressed in the floating-point file.
- Data files 9 through 999 may be assigned to different data types, as required. When assigned to a certain type, a file is then reserved for that type and cannot be used for any other type. Additional input, output, or status files cannot be created.

The bit file, integer file, or floating-point file can be used to store status or data. Which of these you use depends on the intended use of the data. If you are dealing with status rather than data, the bit file is preferable. If you are using very large or very small numbers and require a decimal point, the floating-point file is preferable. The floating-point data type may have a restriction, however, because it may not interface well with external devices or with internal instructions such as counters and timers, which use only 16-bit words. In such a situation, it may be necessary to use the integer file type.

The **input image table file** is that part of the program memory allocated to storing the on/off status of connected discrete inputs. Figure 5-4 shows the connection of an open and closed switch to the input image table file

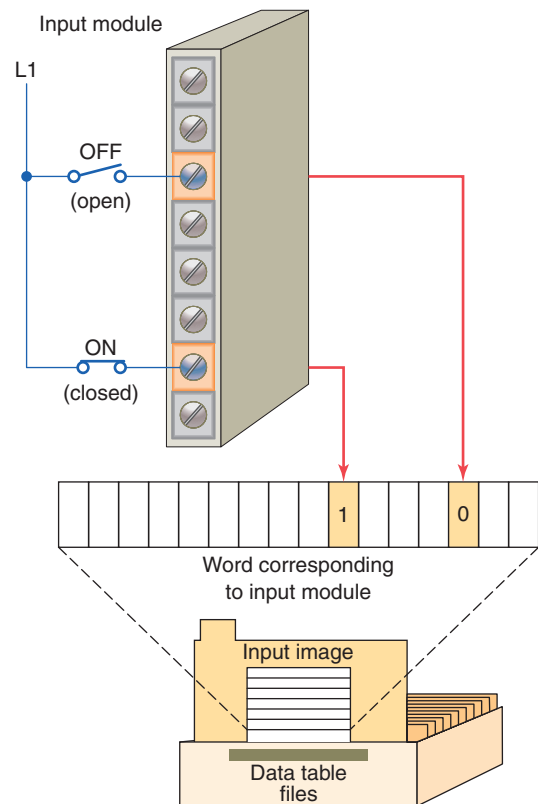


Figure 5-4 Connection of an open and closed switch to the input image table file through the input module.

through the input module. Its operation can be summarized as follows.

- For the switch that is closed, the processor detects a voltage at the input terminal and records that information by storing a binary 1 in its bit location.
- For the switch that is open, the processor detects no voltage at the input terminal and records that information by storing a binary 0 in its bit location.
- Each connected input has a bit in the input image table file that corresponds exactly to the terminal to which the input is connected.
- The input image table file is changed to reflect the current status of the switch during the I/O scan phase of operation.
- If the input is on (switch closed), its corresponding bit in the table is set to 1.
- If the input is off (switch open), the corresponding bit is cleared, or reset to 0.
- The processor continually reads the current input status and updates the input image table file.

The **output image table file** is that part of the program memory allocated to storing the actual on/off status of connected discrete outputs. Figure 5-5 shows a typical

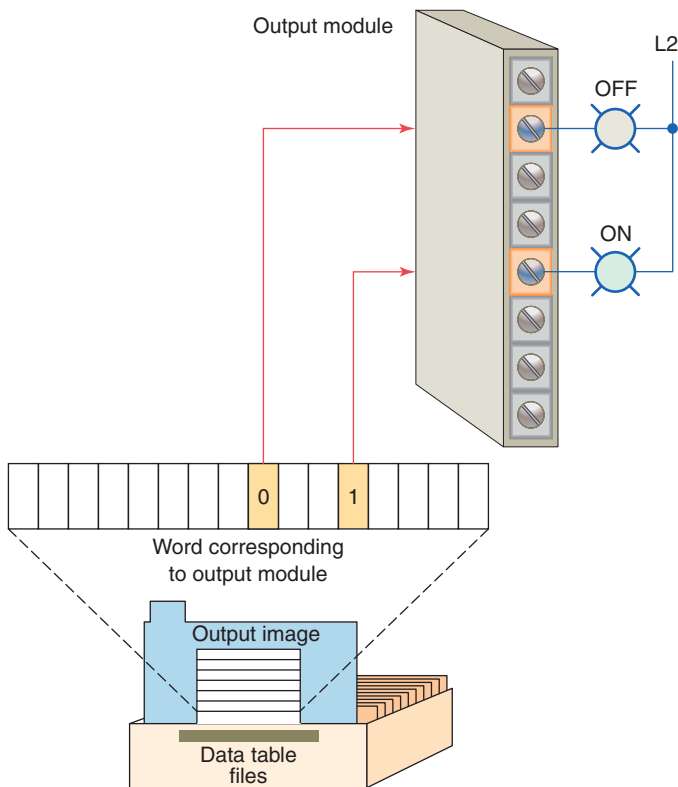


Figure 5-5 Connections of pilot lights to the output image table file through the output module.

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

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

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

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

5.2 Program Scan

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

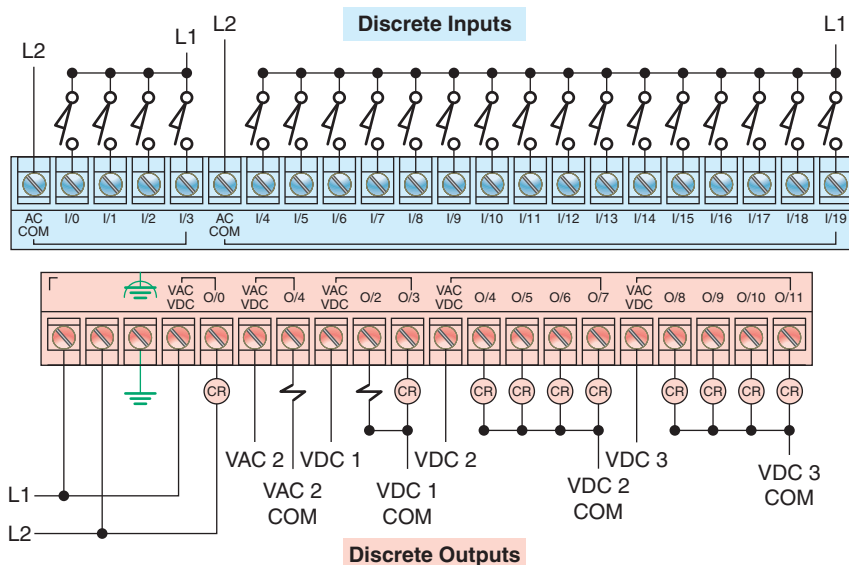


Figure 5-6 Typical micro PLC with predefined addresses.

Source: Image Used with Permission of Rockwell Automation, Inc.

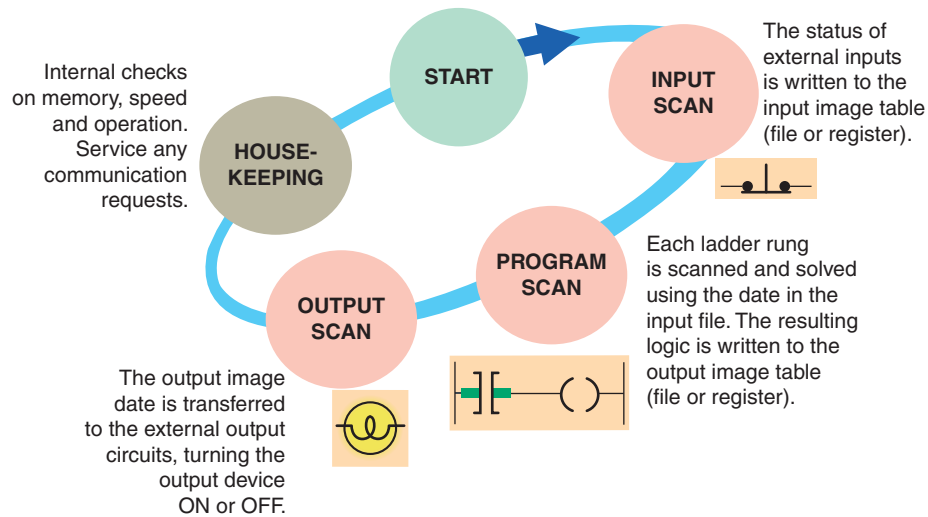


Figure 5-7 PLC program scan cycle.

The time it takes to complete a scan cycle is called the scan cycle time and indicates how fast the controller can react to changes in inputs. The time required to make a single scan can vary from about 1 millisecond to 20 milliseconds. If a controller has to react to an input signal that changes states twice during the scan time, it is possible that the PLC will never be able to detect this change. For example, if it takes 8 ms for the CPU to scan a program, and an input contact is opening and closing every 4 ms, the program may not respond to the contact changing state. The CPU will detect a change if it occurs during the update of the input image table file, but the CPU will not respond to every change. The scan time is a function of the following:

- The speed of the processor module
- The length of the ladder program
- The type of instructions executed
- The actual ladder true/false conditions

The actual scan time is calculated and stored in the PLC's memory. The PLC computes the scan time each time the END instruction is executed. Scan time data can be monitored via the PLC programming. Typical scan time data include the maximum scan time and the last scan time.

The scan is normally a continuous and sequential process of reading the status of inputs, evaluating the control logic, and updating the outputs. Figure 5-8 shows an overview of the data flow during the scan process. For each rung executed, the PLC processor will:

- Examine the status of the input image table bits.
- Solve the ladder logic in order to determine logical continuity.

- Update the appropriate output image table bits, if necessary.
- Copy the output image table status to all of the output terminals. Power is applied to the output device if the output image table bit has been previously set to a 1.
- Copy the status of all of the input terminals to the input image table. If an input is active (i.e., there is electrical continuity), the corresponding bit in the input image table will be set to a 1.

Figure 5-9 illustrates the scan process applied to a simple single rung program. The operation of the scan process can be summarized as follows:

- If the input device connected to address I:3/6 is closed, the input module circuitry senses *electrical continuity* and a 1 (ON) condition is entered into the input image table bit I:3/6.

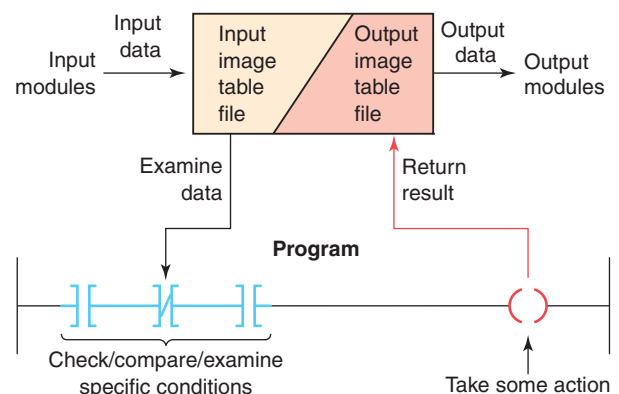


Figure 5-8 Overview of the data flow during the scan process.

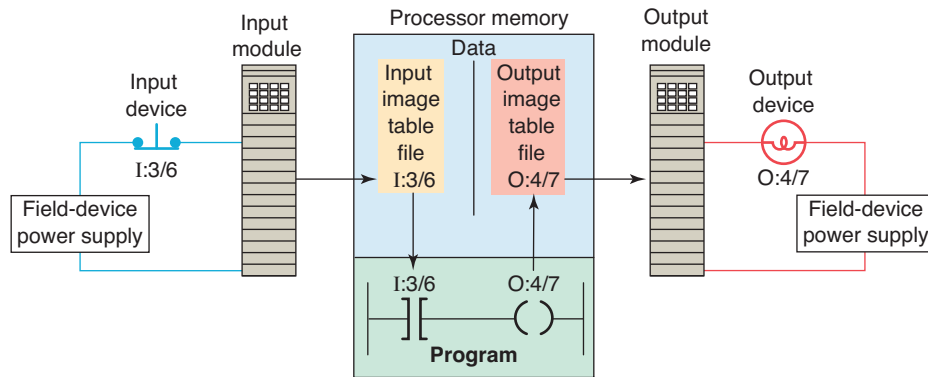


Figure 5-9 Scan process applied to a single rung program.

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

Step 3 The final step of the scan process is to update the actual states of the output devices by transferring the output table results to the output module, thereby switching the connected output devices ON (1) or OFF (0). If the status of any input devices changes when the processor is in step 2 or 3, the output condition will not react to them until the next processor scan.

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

- Step 1** Update the input image table by sensing the voltage of the input terminals. Based on the absence or presence of a voltage, a 0 or a 1 is stored into the memory bit location designated for a particular input terminal.
- Step 2** Solve the ladder logic in order to determine logical continuity. The processor scans the ladder program and evaluates the logical continuity of each rung by referring to the input image table to see if the input conditions are met. If the conditions controlling an output are met, the processor immediately writes a 1 in its memory location, indicating that the output will be turned ON; conversely, if the conditions are not met a 0 indicating that the device will be turned OFF is written into its memory location.

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

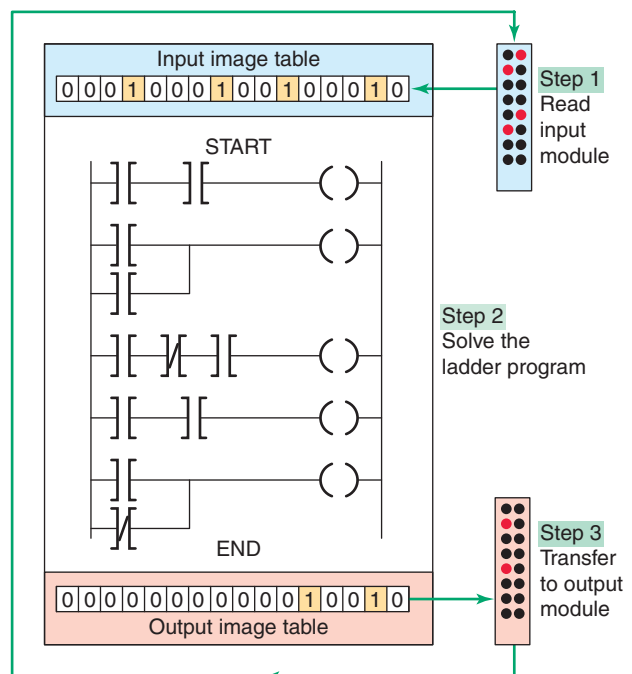


Figure 5-10 Scan process applied to a multiple rung program.

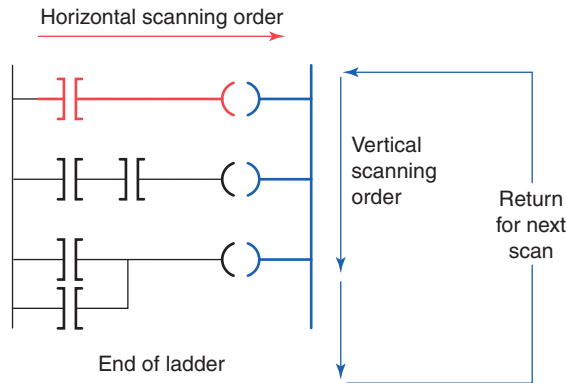


Figure 5-11 Scanning can be vertical or horizontal.

the user program is also dependent on the clock frequency of the microprocessor system. The higher the clock frequency, the faster is the scan rate.

There are two basic scan patterns that different PLC manufacturers use to accomplish the scan function (Figure 5-11). Allen-Bradley PLCs use the *horizontal* scan by rung method. In this system, the processor examines input and output instructions from the first command, top left in the program, horizontally, rung by rung. Modicon PLCs use the *vertical* scan by column method. In this system, the processor examines input and output instructions from the top left command entered in the ladder diagram, vertically, column by column and page by page. Pages are executed in sequence. Both methods are appropriate; however, misunderstanding the way the PLC scans a program can cause programming bugs.

5.3 PLC Programming Languages

The term *PLC programming language* refers to the method by which the user communicates information to the PLC. The standard IEC 61131 (Figure 5-12) was established to standardize the multiple languages associated with PLC programming by defining the following five standard languages:

- **Ladder Diagram (LD)**—a graphical depiction of a process with rungs of logic, similar to the relay ladder logic schemes that were replaced by PLCs.
- **Function Block Diagram (FBD)**—a graphical depiction of process flow using simple and complex interconnecting blocks.
- **Sequential Function Chart (SFC)**—a graphical depiction of interconnecting steps, actions, and transitions.
- **Instruction List (IL)**—a low-level, text-based language that uses mnemonic instructions.
- **Structured Text (ST)**—a high-level, text-based language such as BASIC, C, or PASCAL specifically developed for industrial control applications.

Ladder diagram language is the most commonly used PLC language and is designed to mimic relay logic. The ladder diagram is popular for those who prefer to define control actions in terms of relay contacts and coils, and other functions as block instructions. Figure 5-13 shows a comparison of ladder diagram programming and instruction list programming. Figure 5-13a shows

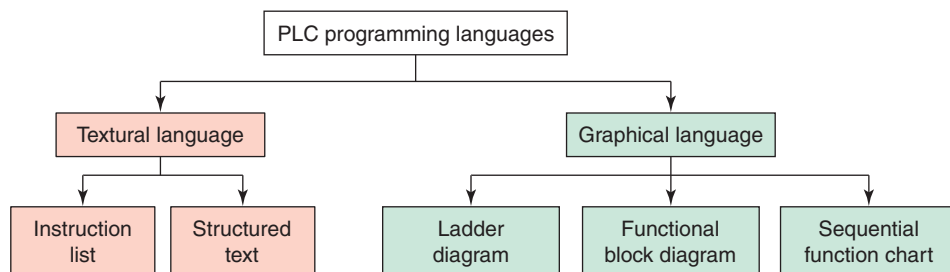


Figure 5-12 Standard IEC 61131 languages associated with PLC programming.

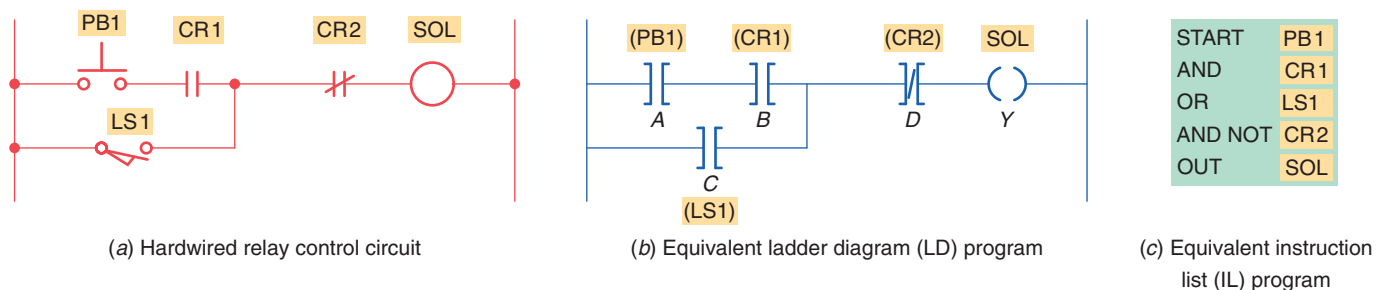


Figure 5-13 Comparison of ladder diagram and instruction list programming.

the original relay hardwired control circuit. Figure 5-13b shows the equivalent logic ladder diagram programmed into a controller. Note how closely the ladder diagram program closely resembles the hardwired relay circuit. The input/output addressing is generally different for each PLC manufacturer. Figure 5-13c show how the original hardwired circuit could be programmed using the instruction list programming language. Note that the instructional list consists of a series of instructions that refer to the basic AND, OR, and NOT logic gate functions.

Functional block diagram programming uses instructions that are programmed as blocks wired together on screen to accomplish certain functions. Typical types of function blocks include logic, timers, and counters. Functional block diagrams are similar in layout to electrical/electronic block diagrams used to simplify complex systems by showing blocks of functionality. The primary concept behind a functional block diagram is data flow. Function blocks are linked together to complete a circuit that satisfies a control requirement. Data flow on a path from inputs, through function blocks or instructions, and then to outputs.

The use of function blocks for programming of programmable logic controllers (PLCs) is gaining wider acceptance. Rather than the classic contact and coil representation of ladder diagram or relay ladder logic programming, function blocks present a graphical image to the programmer with underlying algorithms already defined. The programmer simply completes needed information within the block to complete that phase of the program. Figure 5-14 shows function block diagram equivalents to ladder logic contacts.

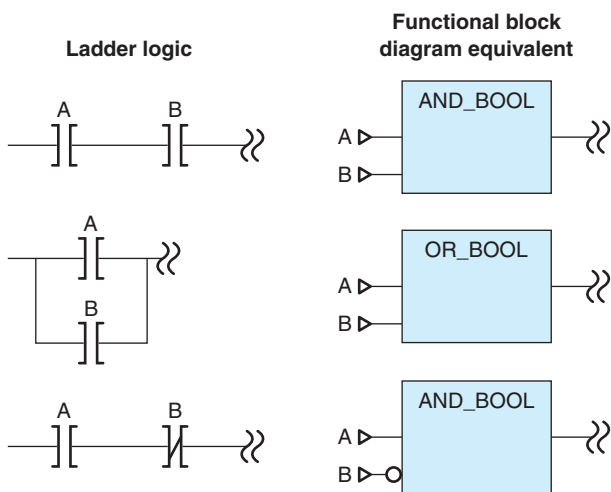


Figure 5-14 Function block diagram equivalents to ladder logic contacts.

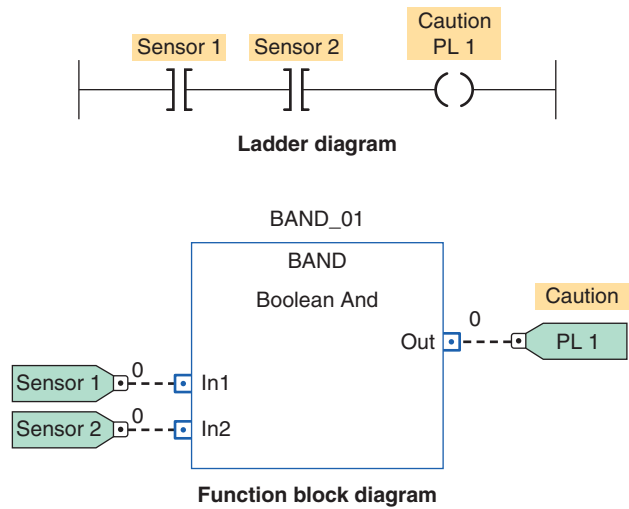


Figure 5-15 PLC ladder and equivalent function block diagram.

Figure 5-15 illustrates how ladder diagram and functional block diagram programming could be used to produce the same logical output. For this application, the objective is to turn on caution pilot light PL 1 whenever both sensor switch 1 and sensor switch 2 are closed. The ladder logic consists of a single rung across the power rails. This rung contains the two input sensor instructions programmed in series with the pilot light output instruction. The function block solution consists of a logic *Boolean And* function block with two input reference tags for the sensors and a single output reference tag for the pilot light. Note there are no power rails in the function block diagram.

Sequential function chart programming language is similar to a flowchart of your process. SFC programming is designed to accommodate the programming of more advanced processes. This type of program can be split into steps with multiple operations happening in parallel branches. The basic elements of a sequential function chart program are shown in Figure 5-16.

Structured text is a high level text language primarily used to implement complex procedures that cannot be easily expressed with graphical languages. Structured text uses statements to define what to execute. Figure 5-17 illustrates how structured text and ladder diagram programming could be used to produce the same logical output. For this application, the objective is to energize SOL 1 whenever either one of the two following circuit conditions exists:

- Sensor 1 and Sensor 2 switches are both closed.
- Sensor 3 and Sensor 4 switches are both closed and Sensor 5 switch is open.

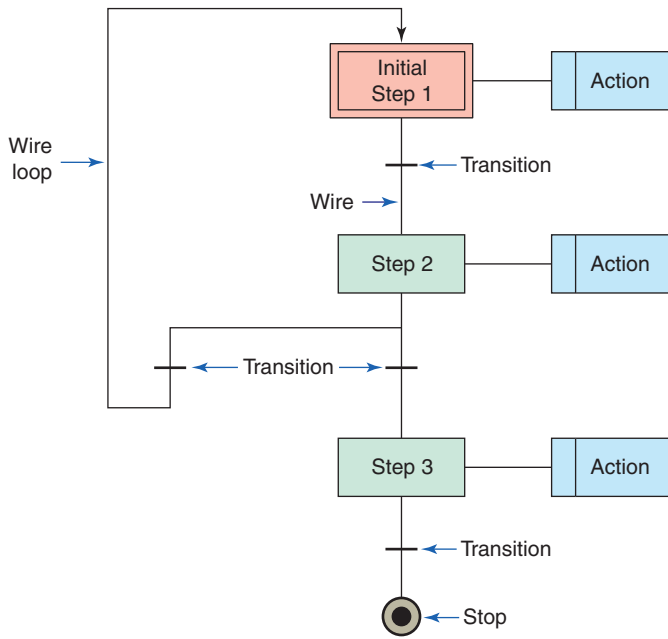
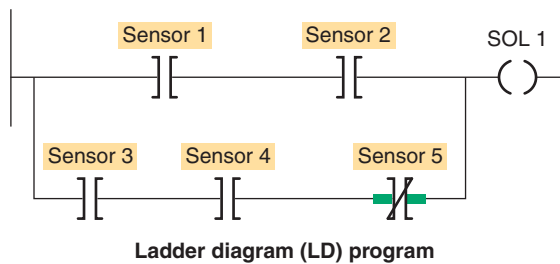


Figure 5-16 Major elements of a sequential function chart program.



```

IF Sensor_1 AND Sensor_2 THEN
  SOL_1 := 1;
ELSEIF Sensor_3 AND Sensor_4 AND NOT Sensor_5 THEN
  SOL_1 := 1;
END_IF;

```

Structured text (ST) program

Figure 5-17 PLC ladder and equivalent structured text program.

5.4 Relay-Type Instructions

The ladder diagram language is basically a *symbolic* set of instructions used to create the controller program. These ladder instruction symbols are arranged to obtain the desired control logic that is to be entered into the memory of the PLC. Because the instruction set is composed of contact symbols, ladder diagram language is also referred to as *contact symbology*.

Representations of contacts and coils are the basic symbols of the logic ladder diagram instruction set. The three fundamental symbols that are used to translate relay control

logic to contact symbolic logic are Examine If Closed (XIC), Examine If Open (XIO), and Output Energize (OTE). Each of these instructions relates to a single bit of PLC memory that is specified by the instruction's address.

The symbol for the *Examine If Closed (XIC)* instruction is shown in Figure 5-18. The XIC instruction, which is also called the Examine-on instruction, looks and operates like a normally open relay contact. Associated with each XIC instruction is a memory bit linked to the status of an input device or an internal logical condition in a rung. This instruction asks the PLC's processor to examine if the contact is *closed*. It does this by examining the bit at the memory location specified by the address in the following manner:

- The memory bit is set to 1 or 0 depending on the status of the input (physical) device or internal (logical) relay address associated with that bit.
- A 1 corresponds to a true status or on condition.
- A 0 corresponds to a false status or off condition.
- When the Examine-on instruction is associated with a physical input, the instruction will be set to 1 when a physical input is present (voltage is applied to the input terminal), and 0 when there is no physical input present (no voltage applied to the input terminal).
- When the Examine-on instruction is associated by address with an internal relay, then the status of the

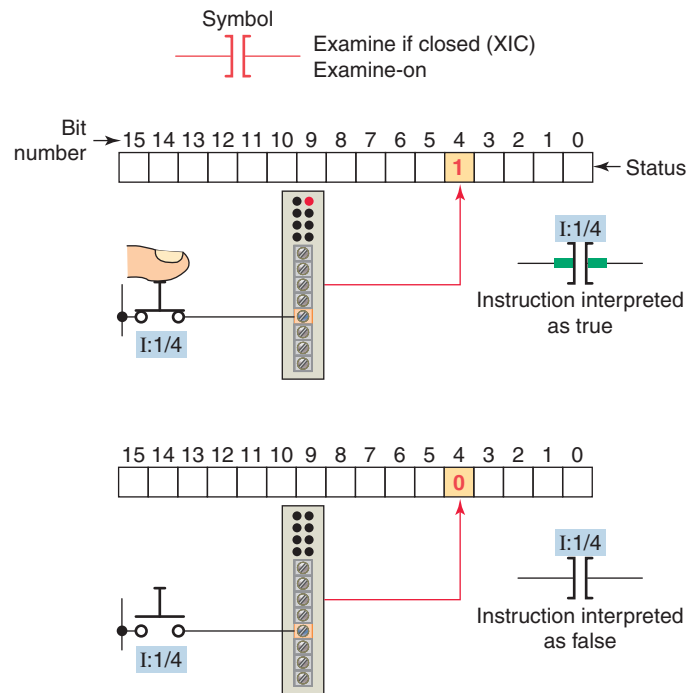


Figure 5-18 Examine If Closed (XIC) instruction.

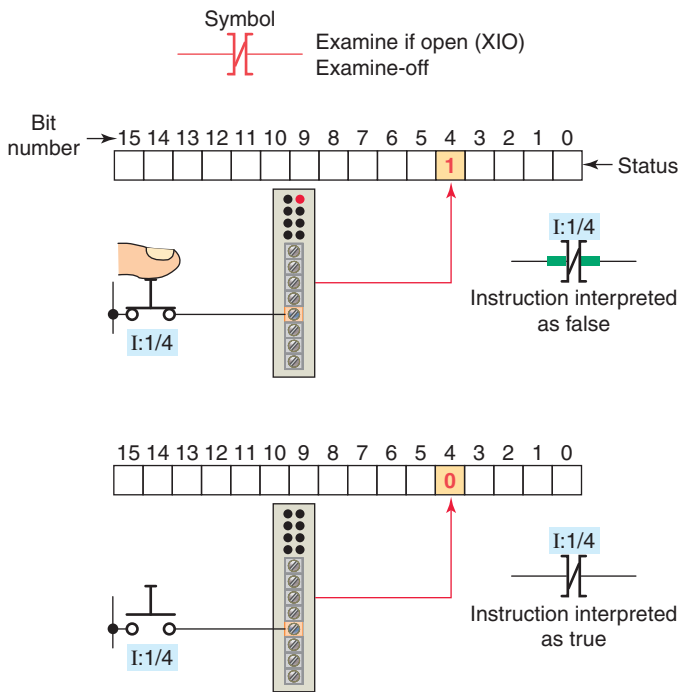


Figure 5-19 Examine If Open (XIO) instruction.

bit is dependent on the logical status of the internal bit with the same address as the instruction.

- If the instruction memory bit is a 1 (true) this instruction will allow rung continuity through itself, like a closed relay contact.
- If the instruction memory bit is a 0 (false) this instruction will not allow rung continuity through itself and will assume a normally open state just like an open relay contact.

The symbol for the *Examine If Open (XIO)* instruction is shown in Figure 5-19. The XIO instruction, which is also called the Examine-off instruction, looks and operates like a normally closed relay contact. Associated with each XIO instruction is a memory bit linked to the status of an input device or an internal logical condition in a rung. This instruction asks the PLC's processor to examine if the contact is *open*. It does this by examining the bit at the memory location specified by the address in the following manner:

- As with any other input the memory bit is set to 1 or 0 depending on the status of the input (physical) device or internal (logical) relay address associated with that bit.
- A 1 corresponds to a true status or on condition.
- A 0 corresponds to a false status or off condition.
- When the Examine-off instruction is used to examine a physical input, then the instruction will be

interpreted as false when there is a physical input (voltage) present (the bit is 1) and will be interpreted as true when there is no physical input present (the bit is 0).

- If the Examine-off instruction were associated by address with an internal relay, then the status of the bit would be dependent on the logical status of the internal bit with the same address as the instruction.
- Like the Examine-on instruction, the status of the instruction (true or false) determines if the instruction will allow rung continuity through itself, like a closed relay contact.
- The memory bit always follows the status (true = 1 or false = 0) of the input address or internal address assigned to it. The interpretation of that bit, however, is determined by which instruction is used to examine it.
- Examine-on instructions always interpret a 1 status as true and a 0 status as false, while Examine-off instructions interpret a 1 status as false and a 0 status as true.

The symbol for the *Output Energize (OTE)* instruction is shown in Figure 5-20. The OTE instruction looks and operates like a relay coil and is associated with a memory bit. This instruction signals the PLC to energize (switch on) or de-energize (switch off) the output. The processor makes this instruction true (analogous to energizing a coil) when there is a logical path of true XIC and XIO instructions in the rung. The operation of the Output Energize instruction can be summarized as follows:

- The status bit of the addressed Output Energize instruction is set to 1 to energize the output and to 0 to de-energize the output.
- If a true logic path is established with the input instructions in the rung, the OTE instruction is energized and the output device wired to its terminal is energized.
- If a true logic path cannot be established or rung conditions go false, the OTE instruction is de-energized and the output device wired to it is switched off.

Sometimes beginner programmers used to thinking in terms of hardwired relay control circuits tend to use the same type of contact (NO or NC) in the ladder logic program that corresponds to the type of field switch wired to the discrete input. While this is true in many instances, it is not the best way to think of the concept. A better approach is to separate the action of the field device from

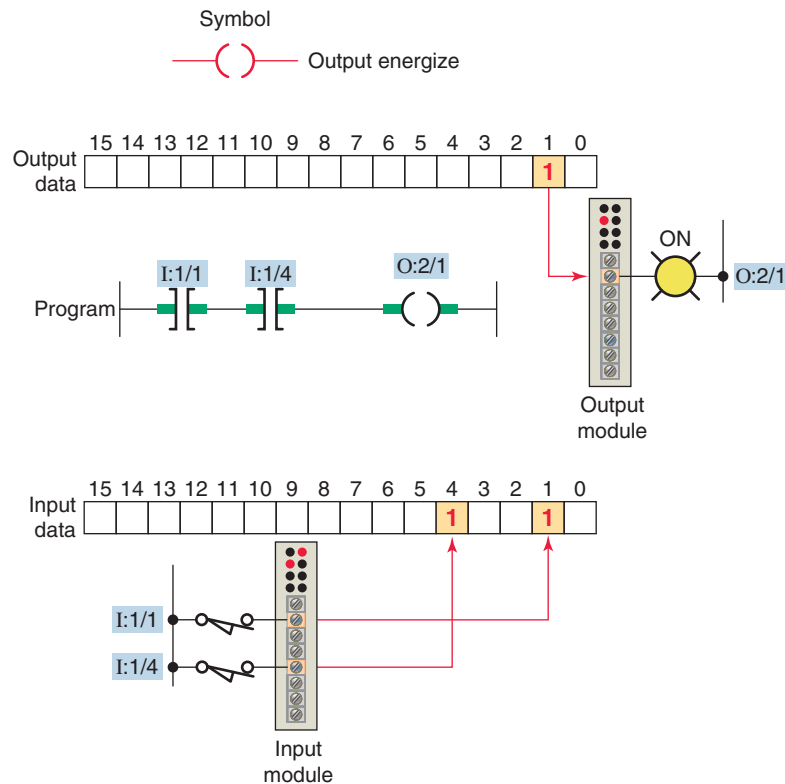


Figure 5-20 Output Energize (OTE) instruction.

the action of the PLC bits as illustrated in Figure 5-21. A signal present makes the NO bit (1) true; a signal absent makes the NO bit (0) false. The reverse is true for an NC bit. A signal present makes the NC bit (1) false; a signal absent makes the NO bit (0) true.

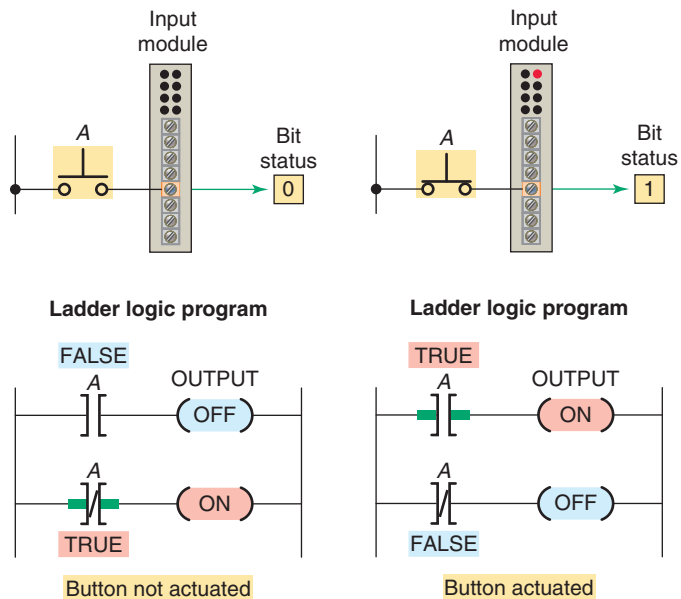


Figure 5-21 Separating the action of the field device and PLC bit.

The main function of the ladder logic diagram program is to control outputs based on input conditions, as illustrated in Figure 5-22. This control is accomplished through the use of what is referred to as a ladder rung. In general, a rung consists of a set of input conditions, represented by contact instructions, and an output instruction at the end of the rung, represented by the coil symbol. Each contact or coil symbol is referenced with an address that identifies what is being evaluated and what is being controlled. The same contact instruction can be used throughout the program whenever that condition needs to be evaluated. The number of ladder logic relays and input and output instructions is limited only by memory size. Most PLCs allow more than one output per rung.

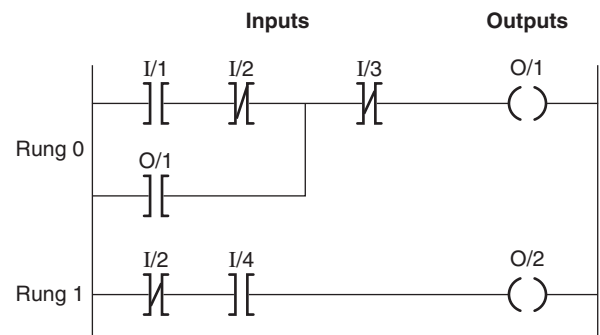


Figure 5-22 Ladder logic diagram rungs.

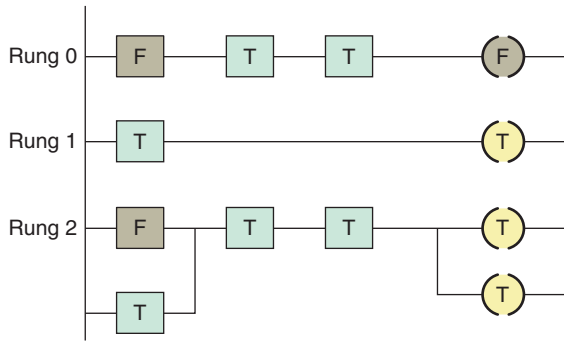


Figure 5-23 Logical continuity.

For an output to be activated or energized, at least one left-to-right true logical path must exist, as illustrated in Figure 5-23. A complete closed path is referred to as having logical continuity. When logical continuity exists in at least one path, the rung condition and Output Energize instruction are said to be true. The rung condition and OTE instruction are false if no logical continuity path has been established. During controller operation, the processor evaluates the rung logic and changes the state of the outputs according to the logical continuity of rungs.

5.5 Instruction Addressing

To complete the entry of a relay-type instruction, you must assign an address to each instruction. This address indicates what PLC input is connected to what input device and what PLC output will drive what output device.

The addressing of real inputs and outputs, as well as internals, depends on the PLC model used. Addressing formats can vary from one PLC family to another as well as for different manufacturers. These addresses can be represented in decimal, octal, or hexadecimal depending on the number system used by the PLC. The address identifies the function of an instruction and links it to a particular bit in the data table portion of the memory. Figure 5-24 shows the addressing format for an Allen-Bradley SLC 500 controller. Addresses contain the slot number of the module where input or output devices are connected. Addresses are formatted as file type, slot number, and bit.

The assignment of an I/O address can be included in the I/O connection diagram, as shown in Figure 5-25. Inputs and outputs are typically represented by squares and diamonds, respectively.

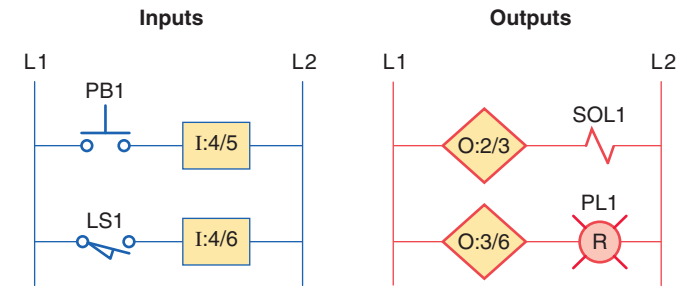


Figure 5-25 I/O connection diagram.

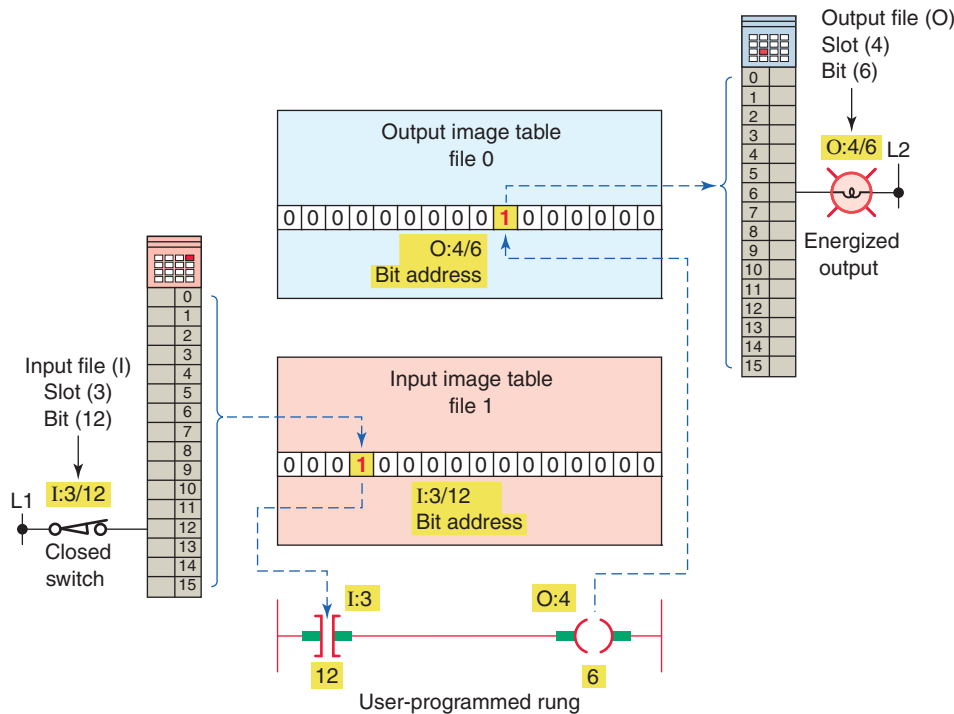


Figure 5-24 Addressing format for an Allen-Bradley SLC 500 controller.

5.6 Branch Instructions

Branch instructions are used to create parallel paths of input condition instructions. This allows more than one combination of input conditions (OR logic) to establish logic continuity in a rung. Figure 5-26 illustrates a typical branch instruction. The rung will be true if either instruction *A* or *B* is true.

Input branching by formation of parallel branches can be used in your application program to allow more than one combination of input conditions. If at least one of these parallel branches forms a true logic path, the rung logic is true and the output will be energized. If none of the parallel branches complete a logical path, logic rung continuity is not established and the output will not be de-energized. In the example shown in Figure 5-27, either *A* and *B*, or *C* provides logical continuity and energizes output *D*.

On most PLC models, branches can be established at both input and output portions of a rung. With output branching, you can program parallel outputs on a rung to allow a true logic path to control multiple outputs, as illustrated in Figure 5-28. When there is a true logic path, all parallel outputs become true. In the example shown, either *A* or *B* provides a true logical path to all three output instructions: *C*, *D*, and *E*.

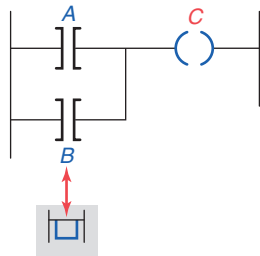


Figure 5-26 Typical branch instruction.

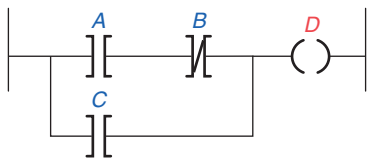


Figure 5-27 Parallel input branches.

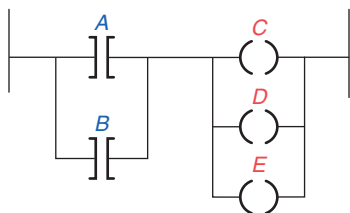


Figure 5-28 Parallel output branches.

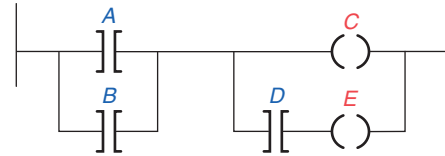


Figure 5-29 Parallel output branching with conditions.

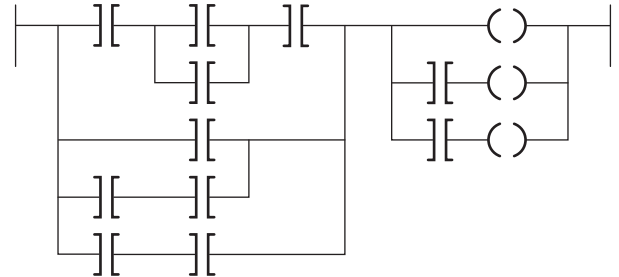


Figure 5-30 Nested input and output branches.

Additional input logic instructions (conditions) can be programmed in the output branches to enhance conditional control of the outputs. When there is a true logic path, including extra input conditions on an output branch, that branch becomes true. In the example shown in Figure 5-29, either *A* and *D* or *B* and *D* provide a true logic path to *E*.

Input and output branches can be *nested* to avoid redundant instructions and to speed up processor scan time. Figure 5-30 illustrates nested input and output branches. A nested branch starts or ends within another branch.

In some PLC models, the programming of a branch circuit within a branch circuit or a *nested* branch cannot be done directly. It is possible, however, to program a logically equivalent branching condition. Figure 5-31 shows an example of a circuit that contains a nested contact *D*. To obtain the required logic, the circuit would be programmed as shown in Figure 5-32. The duplication of contact *C* eliminates the nested contact *D*. Nested branching can be converted into non-nested branches by repeating instructions to make parallel equivalents.

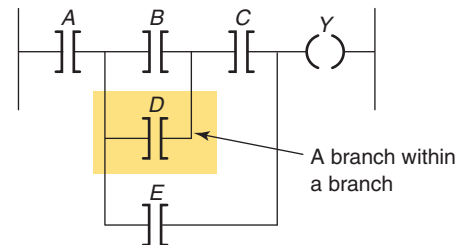


Figure 5-31 Nested contact program.

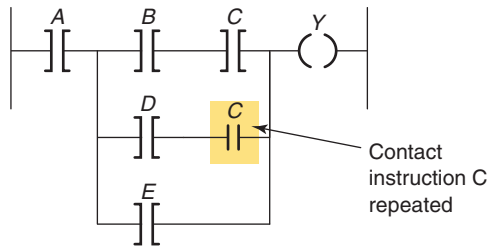
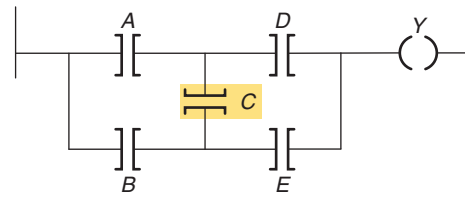


Figure 5-32 Program required to eliminate nested contact.

Some PLC manufacturers have virtually no limitations on allowable series elements, parallel branches, or outputs. For others, there may be limitations to the number of series contact instructions that can be included in one rung of a ladder diagram as well as limitations to the number of parallel branches. Also, there is an additional limitation with some PLCs: only one output per rung and the output must be located at the end of the rung. The only limitation on the number of rungs is memory size. Figure 5-33 shows the matrix limitation diagram for a typical PLC. A maximum of seven parallel lines and 10 series contacts per rung is possible.

Another limitation to branch circuit programming is that the PLC will not allow for programming of vertical contacts. A typical example of this limitation is contact *C* of the user program drawn in Figure 5-34. To obtain the required logic, the circuit would be reprogrammed as shown in Figure 5-35.

The processor examines the ladder logic rung for logic continuity from left to right *only*. The processor never allows for flow from right to left. This situation presents a problem for user program circuits similar to that shown in Figure 5-36. If programmed as shown, contact combination *FDBC* would be ignored. To obtain the required logic, the circuit would be reprogrammed as shown in Figure 5-37.



Boolean equation: $Y = (AD) + (BCD) + (BE) + (ACE)$

Figure 5-34 Program with vertical contact.

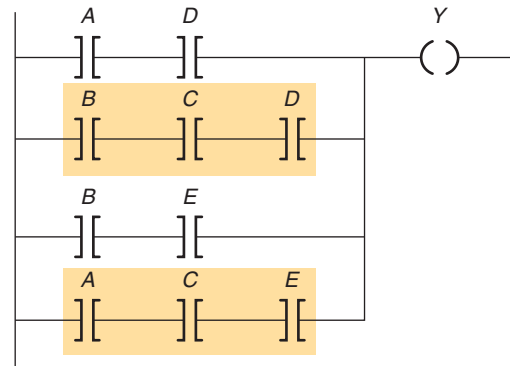
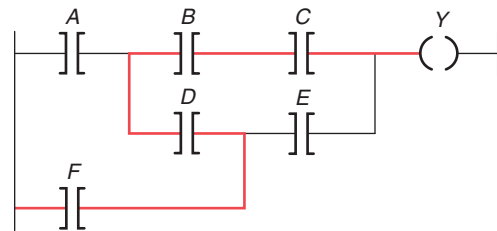


Figure 5-35 Reprogrammed to eliminate vertical contact.



Boolean equation: $Y = (ABC) + (ADE) + (FE) + (FDBC)$

Figure 5-36 Original circuit.

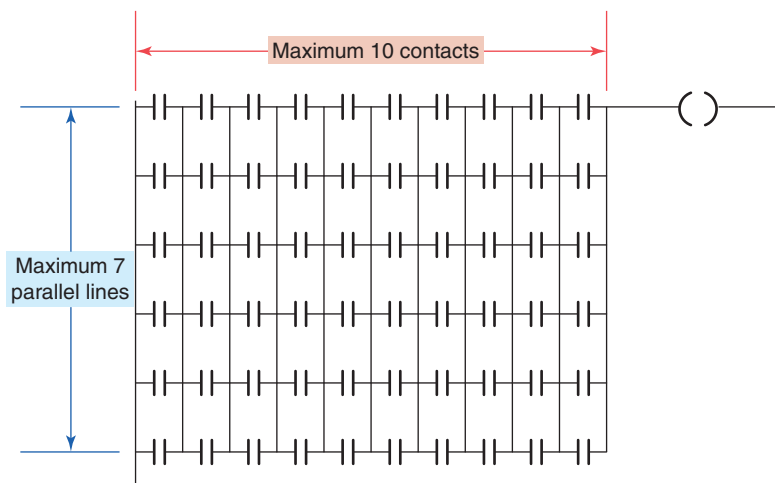


Figure 5-33 PLC matrix limitation diagram.