
REAL-TIME MICROCOMPUTER SYSTEM DESIGN

An Introduction

**Peter D. Lawrence
Konrad Mauch**

REAL-TIME MICROCOMPUTER SYSTEM DESIGN

An Introduction

**Peter D. Lawrence
Konrad Mauch**

*Department of Electrical Engineering
University of British Columbia
Canada*

McGraw-Hill Book Company

New York St. Louis San Francisco Auckland Bogotá Hamburg
Johannesburg London Madrid Mexico Milan Montreal New Delhi
Panama Paris São Paulo Singapore Sydney Tokyo Toronto

This book was set in Times Roman.

The editor was Sanjeev Rao;

the cover was designed by Rafael Hernandez;

the production supervisor was Phil Galea.

Project supervision was done by The Total Book.

R.R. Donnelley & Sons Company was printer and binder.

REAL-TIME MICROCOMPUTER SYSTEM DESIGN

An Introduction

Copyright © 1987 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

1234567890 DOCDOC 8943210987

ISBN 0-07-036731-0

Library of Congress Cataloging-in-Publication Data

Lawrence, Peter D. (Peter Donald)

Real-time microcomputer system design.

(McGraw-Hill series in electrical engineering)

Includes bibliographies and index.

1. Automatic control. 2. Microcomputers.

3. Real-time data processing. 4. System design.

I. Mauch, Konrad. II. Title. III. Series.

TJ223.M53L39 1987 629.8'95 86-15361

ISBN 0-07-036731-0

ISBN 0-07-036732-9 (solutions manual)

PREFACE

Microcomputer-based real-time systems are used not only by electrical and computer engineers but also by engineers in many other disciplines. Mechanical engineers, for example, design robots and other automated machinery controlled by microcomputers. Plants designed by chemical engineers and mineral processing engineers are controlled and monitored by microcomputers. Civil engineers gather hydrological and other environmental data using microcomputer-based data-acquisition systems. Scientists in both the physical and biological sciences are also using microcomputer-based real-time systems to gather and record data from experiments and to control experimental apparatus.

USERS OF THIS BOOK

This book was primarily developed for engineering students in a broad range of disciplines. Practicing engineers and scientists working in the physical, biological, and applied sciences who are seeking a self-contained introduction to the design of real-time microcomputer systems should also find this book useful.

Even though this book is not aimed specifically at electrical engineering students, we believe that it has a place in the electrical engineering curriculum. Used either as a supplement or as the primary text, it provides a high-level perspective on the design process. References after every chapter are provided for readers who may want to pursue a topic in greater depth.

APPROACH

The objective of the book is to present a systems design methodology that involves all the components of a real-time computing system, including sensors, actuators, conditioning electronics, and interfaces as well as the CPU. To

reflect a common practice in industry and to a great extent in scientific research, we have chosen to concentrate on the development of *board-based* real-time microcomputer systems.

We also emphasize the use of *high-level programming languages* to produce software for real-time systems. It is our experience that high-level programming languages can be used effectively in all but the most time-critical real-time systems. When assembly-language programming is required, it can often be reduced to a few short subroutines called by the high-level-language program.

Because of our emphasis on board-level hardware design and the use of high-level programming languages, and because of the nature of the audience that we are writing for, this book differs in content and approach from the standard textbooks on microcomputers used by electrical engineering students. Our approach is mostly top-down, starting with a look at the design process for the real-time system as a whole and then discussing the design of the various components of the system.

We do not discuss the design of microcomputer systems at the chip level in any detail. Our discussion of assembly-language programming is brief and oriented toward how to incorporate short assembly-language modules into a program written in a high-level language and write the necessary assembly-language components of an interrupt handler. However, we include material on important topics such as microcomputer-development systems, real-time operating systems, and transducers and actuators, which are usually not covered in the standard introductory textbooks on microcomputers.

ORGANIZATION

We use this book as a text for an introductory course on microcomputers. Our students are in the senior year of an engineering discipline other than electrical engineering. The majority are mechanical engineering students, but we also have bioresource, chemical, civil, engineering physics, geological, metallurgical, mining and mineral process engineering students. All students have had at least one high-level language programming course (either FORTRAN or Pascal) and an introductory electrical circuits course. Many of the students have also had an introductory electronics course which includes some exposure to digital logic. The material on number systems and digital logic elements in the first two appendixes is assigned to the students for self-study at the beginning of the course to ensure that they all have the required background information.

While we are covering high-level design and software design in lectures, our students are programming C language applications (using an auxiliary C language text) in the laboratory. We then cover assembly language and finally the hardware topics, calling this a top-down software-before hardware organization.

Our course is heavily laboratory-oriented, which is important for any course based on this text, since the general concepts introduced must be reinforced by practical experience with a real microcomputer system. Our microcomputer laboratory consists of a multiuser microcomputer development system (Tektronix 8560), which is linked to Motorola MC6800-based laboratory microcomputers. Each laboratory microcomputer is equipped with a standard backplane bus (STD bus) which contains boards that interface the computer to the apparatus which it monitors and controls. The use of a board-based microcomputer system in the laboratory reinforces the board-based hardware design approach of the text.

Our students develop programs for the laboratory computers on the microcomputer development system. The majority of the programs are written entirely in a high-level programming language (the language C).

The laboratory sequence starts with some programming exercises to introduce the student to the UNIX operating system used on the development system and to programming in C. Then the students develop a program which makes use of pumps, liquid-level sensors, and agitators in the laboratory apparatus to emulate a washing machine. This procedure introduces students to the use of microcomputers in simple sequential control (i.e., programmable controller applications) and shows them very early in the course how a computer can control physical systems. The next sequence of laboratory exercises involves the measurement and control of water temperature, which introduces students to analog interfaces, sensor linearizing, and control via computer. Students then develop a program for a real-time clock. This exercise involves the use of interrupts and the development of a short assembly-language interrupt service routine linked with the main C program. The last laboratory in the sequence varies, depending on the time available and the interests of the instructors and students, but it has included the control of a small robot arm.

OTHER COURSE ORGANIZATIONS

A second approach is to have a bottom-up software-before-hardware organization in which assembly language would be covered first. A suggested chapter order for this approach is: introduction (1), computer structure (2), assembly language (7), high-level languages (5), high-level software example (6), high-level design (3, 4), hardware topics (8 to 14). In the laboratory, the students could do assembly language first, high-level language next, and then programs interacting with interfaces, sensors, actuators, etc.

A third approach is to have a bottom-up hardware-before-software order. In that case the chapters could be ordered as hardware (2, 8, 9, 10 to 13), assembly language (7), high-level language (5, 6), high-level design (3, 4), system-design example (14), development systems (15), and operating systems (16). With this approach it is more difficult to have laboratories early in the

course (since software is required to interact with the hardware and the software is not covered until later in the sequence). It might be used when there are no laboratories in the first semester of a two-semester sequence.

Although our laboratory is centered on a commercial microcomputer development system, many other arrangements are possible. A mainframe computer or minicomputer could be used to develop the software for the laboratory computers. Both Tektronix and Hewlett-Packard, for example, support Pascal and C language cross-compilers and assemblers for many microprocessors. This cross-software runs on the multiuser DEC VAX computers and the HP-9000 series computers. Alternatively, a completely decentralized system could be designed using personal computers such as an IBM PC. The personal computer can be used as a software-development station, and for downloading software to a laboratory experiment station containing a bus-based system; or the personal computer can be enhanced with interface boards itself and be used as the real-time system. The essential requirements are that the development system support software development in a high-level language and that the microcomputer be readily interfaced to transducers and actuators in the lab apparatus.

The book by David Auslander and Paul Sagues entitled *Microprocessors for Measurement and Control* (Osborne—McGraw-Hill, 1981) also contains many laboratory exercises well suited for a course based on our book.

ACKNOWLEDGMENTS

We would like to thank the many engineering students who used drafts of this book over the past three years and who gave us valuable criticism, found errors, and carried us along with their interest in the technology. Jim Dukarm of RSI Robotic Systems International contributed very useful comments during the drafting of the book. There were many helpful discussions with Eric Jackson of International Submarine Engineering.

To Réal Frenette who worked out the solutions to the exercises at the end of each chapter, we would like to express our gratitude.

We are also indebted to the reviewers of this book whose many constructive suggestions we have attempted to incorporate into the text.

Finally, we would like to thank our families and friends for their constant encouragement and support.

Peter D. Lawrence
Konrad Mauch

CONTENTS

Preface

xxi

Part 1 Preliminaries

1	Real-Time Computer Systems	3
1.1	What Is a Real-Time System?	3
1.1.1	Analog Systems	4
1.1.2	Dedicated Digital Systems	5
1.1.3	Computer-Based Systems	7
1.2	An ASM Description of a Real-Time System	8
1.3	Components of a Real-Time System	12
1.3.1	Sensor Characteristics	13
1.3.2	Signal Conditioning	15
1.3.3	Computer Input	17
1.3.4	The Processor	20
1.3.5	Computer Output	21
1.3.6	Output Conditioning and Power Control	22
1.3.7	Actuators	23
1.4	Example of a Real-Time Computer System	23
	Summary	25
	Exercises	26
	Bibliography	28
2	Computer Structure and Function	29
2.1	The Basic Elements of a Computer	29
2.2	The Memory	30
2.2.1	Instructions in Memory	31
2.2.2	Data in Memory	34

x CONTENTS

2.3	The CPU	36
2.3.1	Arithmetic and Logical Unit (ALU)	37
2.3.2	Registers	37
2.3.3	Internal Processor Bus	40
2.3.4	Controller	40
2.4	Buses	45
2.5	Computer Input and Output	45
	Summary	46
	Exercises	47
	Bibliography	48
3	Prelude to the Design Process	50
3.1	The System Components	50
3.2	The Design Specification	53
3.2.1	Functional Specifications	53
3.2.2	Performance	54
3.2.3	Characteristics and Constraints	54
3.3	The Development Environment	56
3.4	Hardware Development	56
3.4.1	Board-Level Hardware	57
3.5	System Software	61
3.5.1	Operating Systems and Application Programs	61
3.5.2	High-Level Language and Assembly Language	62
3.5.3	Development Systems	63
	Summary	64
	Exercises	64
	Bibliography	66

Part 2 High-Level Design

4	Design of Real-Time Systems	69
4.1	The System-Development Cycle	69
4.2	Analysis of System Requirements	71
4.2.1	Requirements Document	71
4.2.2	Response-Time Specification	73
4.2.3	Specification of the Human Interface	73
4.3	Preliminary System Design	74
4.3.1	Block Diagram	74
4.3.2	Representation of Control Flow	75
4.3.3	Representation of Data Flow	78
4.3.4	Functional Decomposition	80
4.3.5	Relationships among Functions	83
4.4	Division into Modules	83
4.4.1	Definition of a Module	83
4.4.2	Advantages of a Modular Design	86
4.5	Preliminary Estimates of Development Cost and System Performance	86

4.5.1	Required Development Time	86
4.5.2	Estimating Program Length	87
4.5.3	Estimating Memory Requirements	89
4.5.4	Estimating Execution Speed	90
4.6	Division between Software and Hardware	90
4.6.1	Hardware-Software Tradeoffs	91
4.6.2	Hardware-Software Tradeoff Example	91
4.7	Software Design	92
4.7.1	Structured Flowcharts	92
4.7.2	Nesting and Stepwise Refinement	93
4.8	Design Review	95
4.9	Translating the Design into a Program	95
4.10	Testing the Module	96
4.10.1	Limitations on Testing in Real-Time Systems	96
4.10.2	Planning for Testing	96
4.10.3	Test Software: Drivers and Stubs	98
4.10.4	Program Instrumentation	98
	Summary	98
	Exercises	100
	References	100
	Bibliography	101
5	Programming Languages	103
5.1	Machine and Assembly Language	103
5.1.1	Machine-Language Programming	103
5.1.2	Assembly-Language Programming	104
5.1.3	Format of Assembly Language Programs	105
5.1.4	The Assembly Process	107
5.1.5	Relocating Assemblers	110
5.1.6	Native and Cross Assemblers	112
5.1.7	Macroassemblers	112
5.1.8	Structured Assemblers	113
5.1.9	Limitations of Assembly Language	114
5.2	High-Level Languages	114
5.2.1	Compiled, Interpreted, and Intermediate Code Languages	115
5.2.2	Systems Programming Languages and Applications Languages	117
5.3	Choosing between Assembly and High-Level Languages	118
5.3.1	Advantages of High-Level Languages	118
5.3.2	Disadvantages of High-Level Languages	120
5.4	Requirements for Real-Time High-Level Languages	121
5.4.1	Speed and Efficient Use of Memory	121
5.4.2	Benchmarks	122
5.4.3	Interface to Assembly Language	123
5.4.4	Compiler Utilities	123
5.4.5	Producing ROMable Programs	125
5.4.6	Access to the Computer's Hardware	125
5.4.7	Bit-Manipulation Operations	126
5.4.8	Reentrancy	126

5.5	Some Real-Time Programming-Languages	127
5.5.1	BASIC	127
5.5.2	FORTRAN	128
5.5.3	FORTH	129
5.5.4	PL/M	130
5.5.5	Pascal	131
5.5.6	Ada	132
5.5.7	C	133
5.6	Choosing a Language	133
	Summary	134
	Exercises	135
	References	136
	Bibliography	136
6	Software Design Example: An I/O Driver for a Simple Peripheral Device	137
6.1	The Pro-Log 7303 Keypad/Display Card	138
6.2	Display Handler	139
6.2.1	Programmer's Model of the Display	139
6.2.2	Design of Display Handler Function	140
6.3	Keypad Handler	144
6.3.1	Programmer's Model	144
6.3.2	Keypad Scanning	144
6.3.3	Keypad Debouncing	146
6.3.4	Keypad Handler Design	147
	Summary	152
	Exercises	153
	Bibliography	153

Part 3 Lower-Level Considerations

7	Integrating Assembly-Language Components	157
7.1	Processor Background Information Required	157
7.1.1	Programming Model of the CPU	158
7.1.2	Representation of Data in Memory	162
7.1.3	Representation of Instructions in Memory	165
7.1.4	Instruction Format in Assembly Language	166
7.1.5	Addressing Modes	167
7.1.6	Instruction Set Description	182
7.2	Assembly-Language Programming	182
7.3	Subroutine Calls and Returns	183
7.4	Subroutine Parameters	186
7.4.1	Parameters in Registers	186
7.4.2	Parameters in Dedicated Memory	187
7.4.3	In-Line Parameter Area	188
7.4.4	Parameter Passing on the Stack	189
7.5	Linking High- and Low-Level Programs	192
7.6	Startup Routines	194

7.7	Computer Interrupt Systems and Service Routines	195
7.7.1	Interrupt Systems and Events	196
7.7.2	Example of an Interrupt System	206
7.7.3	The Interrupt Service Routine	209
	Summary	211
	Exercises	212
	Bibliography	214
8	Fundamental Technological Alternatives	216
8.1	Semiconductor Technologies	217
8.1.1	Field Effect Transistors	217
8.1.2	MOS Logic Circuits	218
8.1.3	Bipolar Transistors	220
8.1.4	Bipolar Junction Transistor Logic	220
8.1.5	Other Technologies	221
8.1.6	Comparison of Main Technologies	221
8.2	Choice of Integration Level	222
8.2.1	Turnkey Systems	223
8.2.2	Prepackaged Systems	223
8.2.3	Board-Level Design	224
8.2.4	Component-Level Design	224
8.2.5	Semicustom Integrated-Circuit Design	226
8.2.6	Full Custom Design	227
8.3	Memory Technologies	228
8.3.1	Random Access Memory (RAM)	229
8.3.2	Read-Only Memory (ROM)	229
8.3.3	Structure of a Memory Chip	230
8.3.4	Selection Factors	232
8.4	Microprocessor Technology	233
8.4.1	Performance	234
8.4.2	Architectural Features	235
	Summary	235
	Exercises	236
	References	237
	Bibliography	237
9	Interfaces to External Signals and Devices	238
9.1	Parallel Input Interfaces	240
9.2	Parallel Output Interfaces	242
9.3	Digital-to-Analog Conversion Interface	243
9.3.1	Digital Representation of Analog Voltage	243
9.3.2	Full-Scale Voltage	245
9.3.3	Other Specifications	245
9.4	Analog-to-Digital Conversion Interface	247
9.4.1	Successive Approximation A/D	248
9.4.2	Dual Slope A/D	248
9.4.3	Flash Converter	249
9.4.4	Sample-and-Hold Circuit	249
9.4.5	The Multiplexer	249

9.5	Real-Time Clock Interfaces	250
9.6	Direct Memory Access Interfaces	252
9.6.1	Interface Description	252
9.6.2	DMA Interface to a Floppy Disk	255
	Summary	260
	Exercises	261
	Reference	262
	Bibliography	262

Part 4 Connected Systems

10	Serial Communications	265
10.1	The EIA RS-232-C Interface Standard	270
10.1.1	Mechanical Specifications	271
10.1.2	Functional Specifications	272
10.1.3	Electrical Specifications	273
10.1.4	Other Related Standards	274
10.2	The IEEE-488 Interface Standard	275
10.2.1	Mechanical Specifications	277
10.2.2	Functional Specifications	278
10.2.3	Electrical Specifications	285
	Summary	288
	Exercises	288
	Bibliography	289
11	Input Systems	291
11.1	Sensors with Binary-State Outputs	292
11.2	Sensors That Produce Binary Pulse Trains	295
11.3	Sensors That Produce Continuous Analog Signals	297
11.3.1	Error Sources	298
11.3.2	Error Analysis	299
11.3.3	Integrated Semiconductor Pressure Sensors	300
11.3.4	Potentiometers	302
11.3.5	Resistance Temperature Detector (RTD)	304
11.3.6	Thermistor	305
11.3.7	Integrated Semiconductor Temperature Sensor	306
11.4	Signal Conditioning Circuits	307
11.4.1	Differential Amplifier	307
11.4.2	Instrumentation Amplifier	309
11.4.3	Analog Isolation Amplifiers	311
11.4.4	Digital Isolation	312
11.4.5	Low-Pass Filtering	313
11.4.6	Gain Conversions and Level Shifting	314
11.5	Transmission Circuitry	315
11.5.1	Short-Range Analog Signal Transmission	316
11.5.2	Medium-Range Analog Signal Transmission	316
11.5.3	Long-Range Analog Signal Transmission	317

11.6	Bus-Compatible Input Systems	318
	Summary	319
	Exercises	319
	Bibliography	321
12	Output Systems	322
12.1	Output Systems Involving Two-State Actuators	322
	12.1.1 Output Port Current and Voltage Ratings	323
	12.1.2 Low- and Medium-Power Switches	324
	12.1.3 Electrically Isolated Switches	327
	12.1.4 Influence of the Actuator on Switch Ratings	331
12.2	Output Systems with Continuous Actuators	334
	12.2.1 Low to Medium-Power Amplifiers	334
	12.2.2 Pulse-Width Modulated Amplifiers	336
12.3	Examples of Actuator Systems	337
	12.3.1 Stepping Motors	337
	12.3.2 Dc Servo Motors	347
	Summary	354
	Exercises	355
	Bibliography	355
<hr/>		
Part 5	Board-Level Design	
13	Board-Based Microcomputer Systems	359
13.1	The Backplane Bus	361
	13.1.1 Address and Data Lines	363
	13.1.2 Lines to Control Data Transfer	364
	13.1.3 Interrupt Control Lines	367
	13.1.4 Bus Request Arbitration	368
	13.1.5 Adapting Microprocessor Control Lines to Bus Control Lines	369
	13.1.6 Power-Supply Lines	369
	13.1.7 Bus Electrical Characteristics	370
	13.1.8 Bus Physical Characteristics	372
13.2	Boards for Bus Systems	373
	13.2.1 CPU Boards	373
	13.2.2 Memory Boards	374
	13.2.3 Peripheral Device Controller and I/O Boards	375
13.3	Development of Backplane Bus Standards	376
13.4	Some Important Backplane Bus Standards	377
	13.4.1 STD (IEEE P961) Bus	377
	13.4.2 IBM Personal Computer Bus	378
	13.4.3 S-100 (IEEE S696) Bus	380
	13.4.4 Multibus (IEEE S796)	381
	13.4.5 VME (IEEE P1014) Bus	382
13.5	Selecting a Bus System	385
	Summary	386
	Exercises	386
	Bibliography	388

14	Example of a Board-Level System Design	390
14.1	A Torque Wrench for Industrial Robots	390
14.1.1	System Operating Cycle	391
14.1.2	System Components	392
14.2	System Specifications	392
14.3	Basic System Design	397
14.3.1	Flow of Control: Algorithmic State-Machine Diagram	397
14.3.2	Decomposition into Functions—Tree diagram	397
14.3.3	Estimating Program Length	397
14.3.4	Module Specification	399
14.4	Hardware Specifications	399
14.4.1	Microcomputer and System Bus	399
14.4.2	Analog-to-Digital Converter	399
14.4.3	Solenoid Drivers	401
14.4.4	Serial Communications Interface	403
14.4.5	Power Supply	403
14.4.6	Timer Function	403
14.5	Hardware Selection	404
14.5.1	Microcomputer Board	404
14.5.2	Analog-to-Digital Converter	406
14.5.3	Solenoid Drivers	407
14.5.4	Serial Communications Interface	409
14.5.5	Power Supply	410
14.5.6	Packaging	412
14.6	Hardware Configuration	413
14.6.1	Power Supply	414
14.6.2	Microcomputer Board	415
14.6.3	Analog-to-Digital Converter Board	419
14.6.4	Solenoid Driver Interface	420
14.6.5	Serial Interface	421
14.7	Software Design I: High-Level Routines	421
14.7.1	State Sequencer	423
14.7.2	WAIT State	425
14.7.3	GRIP and ACK States	426
14.7.4	FASTEN State	427
14.7.5	FAULT State	430
14.8	Software Design II: I/O Driver Routines	431
14.8.1	Solenoid Drivers	431
14.8.2	Analog-to-Digital Converter	432
14.8.3	Serial Interface	434
14.8.4	Real-Time Clock	440
14.9	Software Design III: Assembly-Language Functions and System Initialization	443
14.9.1	Programmer's Model of the Microprocessor	444
14.9.2	System Initialization: Stack Pointer	446
14.9.3	System Initialization: Interrupt System	447
14.9.4	Interrupt Handling	448
14.9.5	Linking the Software Modules	450
14.9.6	Interface and Variable Initialization	450

Summary	451
Exercises	451
Bibliography	452

Part 6 Implementation Tools

15	Development Systems	457
15.1	Peripheral Devices	458
15.2	Operating System	459
15.3	Editor	459
15.4	Object-Code Generation	460
15.5	Software Integration	460
15.5.1	Downloading	460
15.5.2	EPROM Programming	462
15.6	Software Debugging	463
15.7	Other Software-Development-System Capabilities	466
15.8	Examples of Available Development Systems	467
15.9	Selection of a Development System	471
	Summary	472
	Exercises	473
	References	474
	Bibliography	474
16	Operating Systems	475
16.1	Introductory Concepts and Terminology	475
16.1.1	Basic Elements of an Operating System	475
16.1.2	Operating-System Facilities	477
16.2	Operating Systems for Microcomputer Development Systems	479
16.2.1	The CP/M Operating System	480
16.2.2	The UNIX Operating System	481
16.3	Real-Time Operating Systems	482
16.3.1	Tasks and Task Scheduling	483
16.3.2	Task Synchronization and Data Transfer	489
16.4	Factors in Selecting a Real-Time Operating System	494
	Summary	495
	Exercises	496
	Bibliography	498

Appendixes

A	Binary Numbers and Codes	499
A.1	Binary Numbers	499
A.1.1	Unsigned Binary Numbers	499
A.1.2	Representation of Signed Binary Numbers	500
A.2	Binary-Coded Decimal Numbers	501
A.3	Hexadecimal and Octal Numbers	502
A.4	Kbytes and Mbytes	503

A.5	Character Codes	504
A.6	Logical Operations	504
	Exercises	506
B	Basic Logic Devices	508
B.1	Logic Functions	508
B.2	Combinational Logic Devices	509
B.3	Sequential Logic Elements	512
B.4	Logic Electronics and Limitations	514
C	Motorola MC68000 Family Instruction Set	519
D	Intel 8086/8 Family Instruction Set	521
E	The C Programming Language	523
E.1	Basic Elements of a C Program	523
E.1.1	Functions	523
E.1.2	Statements	525
E.1.3	Blocks	525
E.1.4	Constants	526
E.1.5	Variables	526
E.1.6	Comments	526
E.2	Variable Names and Constants	526
E.2.1	Identifiers	526
E.2.2	Numerical Constants	528
E.2.3	Character Constants	529
E.3	Declarations and Variable Types	529
E.3.1	The auto Storage Class	530
E.3.2	The static Storage Class	531
E.3.3	The extern Class	531
E.3.4	Integer Variable Types	532
E.3.5	Real Variable Types	532
E.3.6	Character Type	532
E.3.7	Arrays	533
E.4	Operators and Expressions	533
E.4.1	Arithmetic Operators	533
E.4.2	Assignment Operators	534
E.4.3	Increment and Decrement Operators	534
E.4.4	Bitwise Logical Operators	535
E.4.5	Relational Operators	537
E.4.6	Logical Connective Operators	537
E.4.7	Precedence of Operators	537
E.4.8	Expressions with Mixed-Variable Types	538
E.5	Statements and Control Structures	539
E.5.1	The if else Statement	540
E.5.2	The switch Statement	541
E.5.3	The while and do while Statements	542
E.5.4	The for statement	543