

EMBEDDED SYSTEMS

A Contemporary Design Tool



JAMES K. PECKOL



Embedded Systems

A Contemporary Design Tool

James K. Peckol, Ph.D.

University of Washington



JOHN WILEY & SONS, INC.

Contents

Preface	v
0 Foreword: Introduction to Embedded Systems	xxxv
0.0	Introducing Embedded Systems xxxv
0.1	Philosophy xxxv
0.2	Embedded Systems xxxvi
0.2.1	What Is an Embedded System? xxxvi
0.2.2	Building an Embedded System xxxvii
0.3	The Embedded Design and Development Process xl
0.4	Summary vl
0.5	Review Questions xlv
0.6	Thought Questions xlv
Part One Hardware and Software Infrastructure	
1 The Hardware Side—Part 1: An Introduction	1
1.0	Introduction 1
1.1	The Hardware Side—Getting Started 2
1.2	The Core Level 3
1.2.1	The Microprocessor 5
1.2.2	The Microcomputer 6
1.2.3	The Microcontroller 6
1.2.4	The Digital Signal Processor 7
1.3	Representing Information 8
1.3.1	Word Size 8
1.4	Understanding Numbers 9
1.4.1	Resolution 9
1.4.2	Propagation of Error 10
1.5	Addresses 12
1.6	Instructions 13
1.7	Registers—A First Look 15
1.8	Embedded Systems—An Instruction Set View 17
1.8.1	Instruction Set—Instruction Types 18
1.8.2	Data Transfer Instructions 18
1.8.3	Execution Flow 25
1.9	Embedded Systems—A Register View 32
1.9.1	The Basic Register 33
1.9.2	Register Operations 34
	Write to a Register 34
	Read from a Register 34

1.10	Register Transfer Language 35
1.11	Register View of a Microprocessor 35
1.11.1	The Datapath 36
1.11.2	Processor Control 37
	Fetch 37
	Decode 38
	Execute 38
	Next 38
1.12	Summary 44
1.13	Review Questions 44
1.14	Thought Questions 45
1.15	Problems 46

2 The Hardware Side—Part 2: Combinational Logic—A Practical View 54

2.0	Introduction 54
2.1	A Look at Real-World Gates—Part 1: Signal Levels 55
2.1.1	Logic Levels 55
2.1.2	A First Look Inside the Logic Gate 58
2.1.3	Fan-In and Fan-Out 59
	Logic 0 State 60
	Logic 1 State 60
2.2	A Look at Real-World Gates—Part 2: Time 62
2.2.1	Rise and Fall Times 62
2.2.2	Propagation Delay 63
2.2.3	Race Conditions and Hazards 65
2.3	A Look at Real-World Gates—Part 3: The Legacy of Early Physicists 68
2.3.1	Resistors 68
2.3.2	Capacitors 71
2.4	Logic Circuits and Parasitic Components 73
2.4.1	First-Order RC Circuits 73
2.4.2	Second-Order RLC Circuits 77
2.5	Testing Combinational Circuits—Introduction and Philosophy 79
2.6	Modeling, Simulation, and Tools 79
2.7	Structural Faults 80
2.7.1	Stuck-at Faults 80
2.7.2	Open Circuit Faults 83
2.7.3	Bridging Faults 84
2.8	Functional Faults 87
	Hazards and Race Conditions 87
2.9	Summary 87
2.10	Review Questions 88
2.11	Thought Questions 89
2.12	Problems 90

3 The Hardware Side—Part 3: Storage Elements and Finite-State Machines—A Practical View 96

3.0	Introduction 96
3.1	The Concepts of State and Time 97
	Time 97
	State 97

	State Changes 97
3.2	The State Diagram 98
3.3	Finite-State Machines—A Theoretical Model 99
3.4	Designing Finite-State Machines—Part 1: Registers 101
3.4.1	Storage Registers 101
3.4.2	Shift Registers 101
3.4.3	Linear Feedback Shift Registers 106
3.5	Designing Finite-State Machines: Part 2—Counting and Dividing 108
3.5.1	Dividers 109
	Divide by Two 109
3.5.2	Asynchronous Dividers and Counters 109
3.5.3	Synchronous Dividers and Counters 111
3.5.4	Johnson Counters 112
3.6	Practical Considerations—Part 1: Timing in Latches and Flip-Flops 115
3.6.1	Gated Latches 115
3.6.2	Flip-Flops 115
3.6.3	Propagation Delays 116
3.6.4	Timing Margin 116
3.7	Practical Considerations—Part 2: Clocks and Clock Distribution 118
3.7.1	The Source 118
	Frequency 118
	Precision and Stability 120
3.7.2	Designing a Clock System 121
3.8	Testing Sequential Circuits 124
3.8.1	The Finite-State Machine Model Revisited 125
3.8.2	Sequential Circuit Test—A First Look 125
	Defining Homing and Transfer Sequences 127
3.8.3	Scan Design Techniques 130
3.8.4	Boundary Scan—Extending Scan-Path Techniques 131
3.9	Summary 134
3.10	Review Questions 134
3.11	Thought Questions 135
3.12	Problems 137

4 Memories and the Memory Subsystem

146

4.0	Introduction 146
4.1	Classifying Memory 147
4.2	A General Memory Interface 148
4.3	ROM Overview 149
4.3.1	Read Operation 150
4.4	Static RAM Overview 150
4.4.1	Write Operation 150
4.4.2	Read Operation 151
4.5	Dynamic RAM Overview 152
4.5.1	Read Operation 152
4.5.2	Write Operation 152
4.5.3	Refresh Operation 152
4.6	Chip Organization 154
4.7	Terminology 154
4.8	A Memory Interface in Detail 156
4.9	A SRAM Design 156

	The Memory Array 156
	Reading and Writing 158
	Write 158
	Read 158
4.10	A DRAM Design 159
4.10.1	DRAM Timing Analysis 160
	Core Components 160
4.10.2	DRAM Refresh 161
4.11	The DRAM Memory Interface 161
4.11.1	Refresh Timing 162
4.11.2	Refresh Address 162
4.11.3	Refresh Arbitration 164
4.12	The Memory Map 166
4.13	Memory Subsystem Architecture 167
4.14	Basic Concepts of Caching 168
4.14.1	Locality of Reference 168
4.14.2	Cache System Architecture 169
4.15	Designing a Cache System 169
4.15.1	A High-Level Description 170
4.16	Caching—A Direct Mapped Implementation 170
4.17	Caching—An Associative Mapping Cache Implementation 173
4.18	Caching—A Block-Set Associative Mapping Cache Implementation 175
4.19	Dynamic Memory Allocation 176
4.19.1	Swapping 177
4.19.2	Overlays 177
4.19.3	Multiprogramming 178
	Fixed 178
	Variable Number 179
4.20	Testing Memories 179
4.20.1	RAM Memory 180
4.20.2	ROM Memory 183
4.21	Summary 184
4.22	Review Questions 184
4.23	Thought Questions 186
4.24	Problems 187

5 An Introduction to Software Modeling

190

5.0	Introduction 190
5.1	An Introduction to UML 191
5.2	UML Diagrams 192
5.3	Use Cases 193
	Writing a Use Case 194
5.4	Class Diagrams 194
	Class Relationships 195
5.4.1	Inheritance or Generalization 196
5.4.2	Interface 196
5.4.3	Containment 197
5.5	Dynamic Modeling with UML 198
5.6	Interaction Diagrams 198
5.6.1	Call and Return 199

5.6.2	Create and Destroy	199
5.6.3	Send	199
5.7	Sequence Diagrams	200
5.8	Fork and Join	201
5.9	Branch and Merge	202
5.10	Activity Diagram	203
5.11	State Chart Diagrams	203
5.11.1	Events	203
5.11.2	State Machines and State Chart Diagrams	204
5.12	Dynamic Modeling with Structured Design Methods	207
5.12.1	Brief Introduction to the Structured Design Philosophy	207
5.12.2	Data and Control Flow Diagrams	208
5.13	Summary	210
5.14	Review Questions	211
5.15	Thought Questions	212
5.16	Problems	213

6 The Software Side—Part 1: The C Program

215

6.0	Introduction	215
6.1	Software and Its Manifestations	215
6.1.1	Combining Hardware and Software	216
6.1.2	High-Level Language	217
6.1.3	Preprocessor	217
6.1.4	Cross Compiler	218
6.1.5	Assembler	218
6.1.6	Linker and Loader	218
6.1.7	Storing	220
6.2	An Embedded C Program	220
6.2.1	A Program	220
6.2.2	Developing Embedded Software	221
	Abstraction	221
6.3	C Building Blocks	222
6.3.1	Fundamental Data—What's in a Name?	222
	Identifiers in C	222
6.3.2	Defining Variables—Giving Them a Name and a Value	223
6.3.3	Defining Variables—Giving Them a Type, Scope, and Storage Class	224
6.4	C Program Structure	238
6.4.1	Separate Compilation	239
6.4.2	Translation Units	240
6.4.3	Linking and Linkage	240
6.4.4	Where C Finds Functions	242
6.4.5	Makefiles	242
6.4.6	Standard and Custom Libraries	243
6.4.7	Debug and Release Builds	243
6.5	Summary	244
6.6	Review Questions	244
6.7	Thought Questions	245
6.8	Problems	246

7 The Software Side—Part 2: Pointers and Functions **250**

- 7.0 Introduction 250
- 7.1 Bitwise Operators 251
 - 7.1.1 Bit Manipulation Operations 251
 - 7.1.2 Testing, Resetting, and Setting Bits 252
 - 7.1.3 Arithmetic Operations 255
- 7.2 Pointer Variables and Memory Addresses 256
 - 7.2.1 Getting Started 256
 - 7.2.2 Simple Pointer Arithmetic 260
 - Pointer Comparison 263
 - 7.2.3 Const Pointers 263
 - 7.2.4 Generic and NULL Pointers 265
- 7.3 The Function 266
 - Function Header 266
 - Function Name 266
 - Arguments or Parameter List 267
 - Return 267
 - The Function Body 267
 - 7.3.1 Using a Function 267
 - 7.3.2 Pass by Value 271
 - 7.3.3 Pass by Reference 273
 - 7.3.4 Function Name Scope 274
 - 7.3.5 Function Prototypes 274
 - 7.3.6 Nesting Functions 276
- 7.4 Pointers to Functions 276
- 7.5 Structures 280
 - 7.5.1 The Struct 281
 - 7.5.2 Initialization 283
 - 7.5.3 Access 283
 - 7.5.4 Operations 284
 - 7.5.5 Structs as Data Members 284
 - 7.5.6 Pointers to Structs 288
 - 7.5.7 Passing Structs and Pointers to Structs 289
- 7.6 The Interrupt 290
 - 7.6.1 The Interrupt Control Flow 290
 - 7.6.2 The Interrupt Event 290
 - 7.6.3 The Interrupt Service Routine—ISR 291
 - 7.6.4 The Interrupt Vector Table 292
 - 7.6.5 Control of the Interrupt 293
- 7.7 Summary 295
- 7.8 Review Questions 295
- 7.9 Thought Questions 296
- 7.10 Problems 298

Part Two Developing the Foundation

8	Safety, Reliability, and Robust Design	301
8.0	Introduction 301	
8.1	Safety 302	
8.2	Reliability 302	
8.3	Faults, Errors, and Failures 304	
8.4	Another Look at Reliability 305	
8.5	Some Real-World Examples 306	
	Big Word . . . Small Register 306	
	It's My Turn—Not Yours 307	
	Where Do I Put My Stuff? 307	
8.6	Single-Point and Common Mode Failure Model 308	
8.7	Safe Specifications 308	
8.8	Safe and Robust Designs 309	
	8.8.1 Understanding System Requirements 309	
	8.8.2 Managing Essential Information 310	
	8.8.3 The Review Process 311	
	8.8.4 Bug Lists 311	
	8.8.5 Errors and Exceptions 312	
	8.8.6 Use the Available Tools 314	
8.9	Safe and Robust Designs—The System 315	
8.10	System Functional Level Considerations 316	
	8.10.1 Control and Alarm Subsystems 316	
	8.10.2 Memory and Bus Subsystems 316	
	8.10.3 Data Faults and the Communications Subsystem 316	
	8.10.4 Power and Reset Subsystems 316	
	8.10.5 Peripheral Device Subsystems 317	
	8.10.6 Clock Subsystem 317	
8.11	System Architecture Level Considerations 317	
	8.11.1 Fail Operational ² /Fail Operational Capability 317	
	8.11.2 Reduced Capability 319	
8.12	Busses—The Subsystem Interconnect 320	
	8.12.1 The Star Configuration 320	
	8.12.2 The Multidrop Bus Configuration 321	
	8.12.3 The Ring Configuration 321	
8.13	Data and Control Faults—Data Boundary Values 322	
	8.13.1 Type Conformance 322	
	8.13.2 Boundary Values 322	
8.14	Data and Control Faults—The Communications Subsystem 323	
	8.14.1 Damaged Data 323	
	8.14.2 Managing Damaged Data 324	
8.15	The Power Subsystem 333	
	8.15.1 Full Operation 334	
	8.15.2 Reduced Operation 334	
	8.15.3 Backup Operation 335	
8.16	Peripheral Devices—Built-In Self-Test (BIST) 335	
	8.16.1 Self-Tests 336	
	8.16.2 Busses 337	
	8.16.3 ROM Memory 339	
	8.16.4 RAM Memory 339	
8.17	Failure Modes and Effects Analysis 340	

- 8.18 Summary 342
- 8.19 Review Questions 342
- 8.20 Thought Questions 343
- 8.21 Problems 344

9 Embedded Systems Design and Development

346

- 9.0 Introduction 346
- 9.1 System Design and Development 348
 - 9.1.1 Getting Ready—Start Thinking 348
 - 9.1.2 Getting Started 349
- 9.2 Life-Cycle Models 350
 - 9.2.1 The Waterfall Model 351
 - 9.2.2 The V Cycle Model 352
 - 9.2.3 The Spiral Model 353
 - 9.2.4 Rapid Prototyping—Incremental 354
- 9.3 Problem Solving—Five Steps to Design 354
- 9.4 The Design Process 355
- 9.5 Identifying the Requirements 356
- 9.6 Formulating the Requirements Specification 358
 - 9.6.1 The Environment 359
 - 9.6.2 The System 360
- 9.7 The System Design Specification 366
 - 9.7.1 The System 367
 - 9.7.2 Quantifying the System 367
- 9.8 System Specifications versus System Requirements 375
- 9.9 Partitioning and Decomposing a System 376
 - 9.9.1 Initial Thoughts 376
 - 9.9.2 Coupling 378
 - 9.9.3 Cohesion 379
 - 9.9.4 More Considerations 380
- 9.10 Functional Design 380
- 9.11 Architectural Design 384
 - 9.11.1 Mapping Functions to Hardware 384
 - 9.11.2 Hardware and Software Specification and Design 386
- 9.12 Functional Model versus Architectural Model 388
 - 9.12.1 The Functional Model 388
 - 9.12.2 The Architectural Model 389
 - 9.12.3 The Need for Both Models 389
- 9.13 Prototyping 389
 - 9.13.1 Implementation 390
 - 9.13.2 Analyzing the System Design 390
- 9.14 Other Considerations 392
 - 9.14.1 Capitalization and Reuse 392
 - Capitalization 392
 - Reuse 392
 - 9.14.2 Requirements Traceability and Management 393
 - Requirements Traceability 393
 - Requirements Management 393
- 9.15 Archiving the Project 393
- 9.16 Summary 395
- 9.17 Review Questions 395
- 9.18 Thought Questions 396
- 9.19 Problems 398

10 Hardware Test and Debug**401**

- 10.0 Introduction 401
- 10.1 Some Vocabulary 401
- 10.2 Putting Together a Strategy 402
- 10.3 Formulating a Plan 403
- 10.4 Formalizing the Plan—Writing a Specification 405
- 10.5 Executing the Plan—The Test Procedure and Test Cases 406
- 10.6 Applying the Strategy—Egoless Design 406
- 10.7 Applying the Strategy—Design Reviews 407
- 10.8 Applying the Strategy—Module Debug and Test 407
 - 10.8.1 Black Box Tests 408
 - 10.8.2 White Box Tests 409
 - 10.8.3 Gray Box Tests 409
- 10.9 Applying the Strategy—The First Steps 409
 - 10.9.1 The Parts 410
 - 10.9.2 Initial Tests and Measurements—Before Applying Power 411
 - 10.9.3 Initial Tests and Measurements—Immediately after Applying Power 411
- 10.10 Applying the Strategy—Debugging and Testing 412
 - 10.10.1 The Reset System 412
 - 10.10.2 The Clocks and Timing 412
 - 10.10.3 The Inputs and Outputs 413
 - 10.10.4 Sudden Failure during Debugging 413
- 10.11 Testing and Debugging Combinational Logic 415
- 10.12 Path Sensitizing 415
 - 10.12.1 Single Variable—Single Path 416
 - Testing 416
 - Debugging 416
 - 10.12.2 Single Variable—Two Paths 417
- 10.13 Masking and Untestable Faults 418
- 10.14 Single Variable—Multiple Paths 419
- 10.15 Bridge Faults 420
- 10.16 Debugging—Sequential Logic 421
- 10.17 Scan Design Testing 423
- 10.18 Boundary-Scan Testing 426
- 10.19 Memories and Memory Systems 427
- 10.20 Applying the Strategy—Subsystem and System Test 427
- 10.21 Applying the Strategy—Testing for Our Customer 428
 - 10.21.1 Alpha and Beta Tests 428
 - 10.21.2 Verification Tests 428
 - 10.21.3 Validation Tests 428
 - 10.21.4 Acceptance Tests 429
 - 10.21.5 Production Tests 429
- 10.22 Self-Test 429
 - 10.22.1 On Demand 429
 - 10.22.2 In Background 429
- 10.23 Summary 430
- 10.24 Review Questions 430
- 10.25 Thought Questions 431
- 10.26 Problems 432

Part Three Doing the Work

11	Real-Time Kernels and Operating Systems	433
11.0	Introduction 433	
11.1	Tasks and Things 434	
11.2	Programs and Processes 435	
11.3	The CPU Is a Resource 436	
11.3.1	Setting a Schedule 437	
11.3.2	Changing Context 438	
11.4	Threads—Lightweight and Heavyweight 438	
11.4.1	A Single Thread 439	
11.4.2	Multiple Threads 439	
11.5	Sharing Resources 440	
11.5.1	Memory Resource Management 441	
11.5.2	Process-Level Management 442	
11.5.3	Reentrant Code 442	
11.6	Foreground/Background Systems 442	
11.7	The Operating System 443	
11.8	The Real-Time Operating System (RTOS) 444	
11.9	Operating System Architecture 444	
11.10	Tasks and Task Control Blocks 445	
11.10.1	The Task 445	
11.10.2	The Task Control Block 446	
11.10.3	A Simple Kernel 448	
11.10.4	Interrupts Revisited 451	
11.11	Memory Management Revisited 455	
11.11.1	Duplicate Hardware Context 456	
11.11.2	Task Control Blocks 457	
11.11.3	Stacks 457	
	Runtime Stack 458	
	Application Stacks 459	
	Multiprocessing Stacks 459	
11.12	Summary 460	
11.13	Review Questions 460	
11.14	Thought Questions 461	
11.15	Problems 461	
12	Tasks and Task Management	463
12.0	Introduction 463	
12.1	Time, Time-Based Systems, and Reactive Systems 464	
12.1.1	Time 464	
12.1.2	Reactive and Time-Based Systems 464	
12.2	Task Scheduling 466	
12.2.1	CPU Utilization 467	
12.2.2	Scheduling Decisions 467	
12.2.3	Scheduling Criteria 468	
12.3	Scheduling Algorithms 469	
12.3.1	Asynchronous Interrupt Event Driven 470	
12.3.2	Polled and Polled with a Timing Element 470	

12.3.3	State Based	471
12.3.4	Synchronous Interrupt Event Driven	471
12.3.5	Combined Interrupt Event Driven	472
12.3.6	Foreground–Background	472
12.3.7	Time-Shared Systems	472
12.3.8	Priority Schedule	473
12.4	Real-Time Scheduling Considerations	475
12.5	Algorithm Evaluation	476
12.5.1	Deterministic Modeling	476
12.5.2	Queuing Models	477
12.5.3	Simulation	478
12.5.4	Implementation	478
12.6	Tasks, Threads, and Communication	478
12.6.1	Getting Started	478
12.6.2	Intertask/Interthread Communication	479
12.6.3	Shared Variables	479
12.6.4	Messages	483
12.7	Task Cooperation, Synchronization, and Sharing	487
12.7.1	Critical Sections and Synchronization	488
12.7.2	Flags	491
12.7.3	Token Passing	493
12.7.4	Interrupts	493
12.7.5	Semaphores	494
12.7.6	Process Synchronization	495
12.7.7	Spin Lock and Busy Waiting	496
12.7.8	Counting Semaphores	496
12.8	Talking and Sharing in Space	497
12.8.1	The Bounded Buffer Problem	497
12.8.2	The Readers and Writers Problem	499
12.9	Monitors	501
12.9.1	Condition Variables	502
12.9.2	Bounded Buffer Problem with Monitor	503
12.10	Starvation	504
12.11	Deadlocks	504
12.12	Summary	505
12.13	Review Questions	505
12.14	Thought Questions	506
12.15	Problems	506

13 Deadlocks

510

13.0	Introduction	510
13.1	Sharing Resources	510
13.2	System Model	511
13.3	Deadlock Model	512
	Necessary Conditions	512
13.4	A Graph Theoretic Tool—The Resource Allocation Graph	513
13.5	Handling Deadlocks	515
13.6	Deadlock Prevention	516
13.6.1	Mutual Exclusion	516
13.6.2	Hold and Wait	516
13.6.3	No Preemption	516

- 13.6.4 Circular Wait 517
- 13.7 Deadlock Avoidance 517
 - 13.7.1 Algorithms Based on the Resource Allocation Graph 518
 - 13.7.2 Banker's Algorithm and Safe States 519
- 13.8 Deadlock Detection 522
 - 13.8.1 Detection in a Single-Instance Environment 522
 - 13.8.2 Deadlock Recovery 522
- 13.9 Summary 524
- 13.10 Review Questions 524
- 13.11 Thought Questions 524
- 13.12 Problems 525

14 Performance Analysis and Optimization

528

- 14.0 Introduction 528
- 14.1 Getting Started 529
- 14.2 Performance or Efficiency Measures 529
 - 14.2.1 Introduction 529
 - 14.2.2 The System 530
 - 14.2.3 Some Limitations 531
- 14.3 Complexity Analysis—A High-Level Measure 531
- 14.4 The Methodology 533
 - 14.4.1 A Simple Experiment 533
 - 14.4.2 Working with Big Numbers 534
 - 14.4.3 Asymptotic Complexity 534
- 14.5 Comparing Algorithms 535
 - 14.5.1 Big-O Notation 535
 - 14.5.2 Big-O Arithmetic 536
- 14.6 Analyzing Code 537
 - 14.6.1 Constant Time Statements 537
 - 14.6.2 Looping Constructs 538
 - For loops 538
 - While Loops 539
 - 14.6.3 Sequences of Statements 539
 - 14.6.4 Conditional Statements 540
 - 14.6.5 Function Calls 540
- 14.7 Analyzing Algorithms 541
 - 14.7.1 Analyzing Search 541
 - 14.7.2 Analyzing Sort 542
- 14.8 Analyzing Data Structures 543
 - 14.8.1 Array 544
 - 14.8.2 Linked List 544
- 14.9 Instructions in Detail 545
 - 14.9.1 Getting Started 545
 - 14.9.2 Flow of Control 546
 - Sequential 546
 - Branch 546
 - Loop 547
 - Function Call 547
 - 14.9.3 Analyzing the Flow of Control—Two Views 547
 - Sequential Flow 548
 - Branch 548

	Loop	549
	Function Call	549
	Co-routine	552
	Interrupt Call	552
14.10	Time, Etc.—A More Detailed Look	553
	Metrics	553
14.11	Response Time	554
	14.11.1 Polled Loops	554
	14.11.2 Co-routine	555
	14.11.3 Interrupt-Driven Environment	555
	Preemptive Schedule	556
	Nonpreemptive Schedule	556
14.12	Time Loading	556
	14.12.1 Instruction Counting	557
	14.12.2 Simulation	557
	14.12.3 Timers	558
	14.12.4 Instrumentation	558
14.13	Memory Loading	559
	14.13.1 Memory Map	559
	14.13.2 Designing a Memory Map	560
	Instruction/Firmware Area	560
	RAM Area	560
	Stack Area	561
14.14	Evaluating Performance	561
	Early Stages	562
	Midstages	562
	Later Stages	562
14.15	Thoughts on Performance Optimization	562
	Questions to Ask	563
14.16	Performance Optimization	563
	Common Mistakes	563
14.17	Tricks of the Trade	564
14.18	Hardware Accelerators	569
14.19	Optimizing for Power Consumption	569
	14.19.1 Software	570
	14.19.2 Hardware	571
14.20	Caches and Performance	574
14.21	Trade-offs	576
14.22	Summary	576
14.23	Review Questions	576
14.24	Thought Questions	577
14.25	Problems	578

Part Four Interacting with the Physical World

15	Working Outside of the Processor I: A Model of Interprocess Communication	581
15.0	Communication and Synchronization with the Outside World 581	
15.1	First Steps: Understanding the Problem 582	
15.2	Interprocess Interaction Revisited 583	
15.3	The Model 585	
15.3.1	Information 585	
15.3.2	Places 585	
15.3.3	Control and Synchronization 586	
15.3.4	Transport 586	
15.4	Exploring the Model 587	
15.4.1	The Transport Mechanism 587	
15.4.2	Control and Synchronization 592	
15.4.3	Information Flow 592	
15.4.4	Places 594	
15.5	Summary 595	
15.6	Review Questions 595	
15.7	Thought Questions 595	
16	Working Outside of the Processor I: Refining the Model of Interprocess Communication	597
16.0	Communication and Synchronization with the Outside World 597	
16.1	The Local Device Model 598	
16.1.1	Control, Synchronization, and Places 598	
16.1.2	Information—Data 601	
16.1.3	Transport 601	
16.2	Implementing the Local Device Model—A First Step 601	
16.2.1	An Overview 601	
16.2.2	Main Memory Address Space—Memory-Mapped I/O 603	
	Address Bus 603	
	Data Bus 604	
	Control Bus 604	
	Read 604	
	Write 605	
	Bidirectional Bus 605	
16.2.3	I/O Ports—Program-Controlled I/O 607	
16.2.4	The Peripheral Processor 607	
16.3	Implementing the Local Device Model—A Second Step 608	
16.3.1	Information Interchange—An Event 609	
16.3.2	Information Interchange—A Shared Variable 609	
16.3.3	Information Interchange—A Message 610	
16.4	Implementing an Event-Driven Exchange—Interrupts and Polling 610	
16.4.1	Polling 610	
16.4.2	Interrupts 612	
16.4.3	Masking Interrupts 617	
16.5	A Message 618	
16.5.1	Asynchronous Information Exchange 619	
	Strobes 619	
	Strobe with Acknowledge 619	

	Full Handshake 620
	Resynchronization 620
	Analysis 621
16.5.2	Synchronous Information Exchange 622
16.6	The Remote Device Model 625
16.6.1	Places and Information 626
16.6.2	Control and Synchronization 626
16.6.3	Transport 626
16.7	Implementing the Remote Device Model—A First Step 627
16.7.1	The OSI and TCP/IP Protocol Stacks 628
	OSI—Physical Layer 628
	OSI—Data Link Layer 628
	TCP/IP—Host to Network 629
	OSI—Network Layer 629
	TCP/IP—Internet Layer 629
	OSI—Transport Layer 630
	TCP/IP—Transport Layer 630
	OSI—Session Layer 630
	OSI—Presentation Layer 630
	OSI—Application Layer 630
	TCP/IP—Application Layer 631
16.7.2	The Models 631
16.8	Implementing the Remote Device Model—A Second Step 633
16.8.1	The Messages 633
16.8.2	The Message Structure 633
16.8.3	Message Control and Synchronization 634
16.9	Working with Remote Tasks 635
16.9.1	Preliminary Thoughts on Working with Remote Tasks 635
	Local vs. Remote Addresses and Data 636
	Repeated Task Execution 636
	Node Failure, Link Failure, Message Loss 637
16.9.2	Procedures and Remote Procedures 637
16.9.3	Node Failure, Link Failure, Message Loss 641
16.10	Group Multicast Revisited 643
	Atomic Multicast 643
	Reliable Multicast 643
16.11	Connecting to Distributed Processes—Pipes, Sockets, and Streams 643
16.11.1	Pipes 644
16.11.2	Sockets 644
16.11.3	Stream Communication 645
16.12	Summary 646
16.13	Review Questions 646
16.14	Thought Questions 647
16.15	Problems 648

17 Working Outside of the Processor II: Interfacing to Local Devices

649

17.0	Shared Variable I/O—Interfacing to Peripheral Devices 649
17.1	The Shared Variable Exchange 650
17.2	Generating Analog Signals 650
17.2.1	Binary Weighted Digital-to-Analog Converter 650
17.2.2	R/2R Ladder Digital-to-Analog Converter 653

17.3	Common Measurements 655
17.3.1	Voltage 655
17.3.2	Current 655
17.3.3	Resistance 656
17.4	Measuring Voltage 656
17.4.1	Dual Slope Analog-to-Digital Conversion 656
17.4.2	Successive Approximation Analog-to-Digital Conversion 660
	Sample and Hold 662
17.4.3	VCO Analog-to-Digital Conversion 663
17.5	Measuring Resistance 664
17.6	Measuring Current 665
17.7	Measuring Temperature 666
17.7.1	Sensors 666
17.7.2	Making the Measurement 667
17.7.3	Working with Nonlinear Devices 668
17.8	Generating Digital Signals 670
17.8.1	Motors and Motor Control 670
17.8.2	DC Motors 670
17.8.3	Servo Motors 671
17.8.4	Stepper Motors 671
17.9	Controlling DC and Servo Motors 672
17.9.1	DC Motors 672
17.9.2	Servo Motors 673
17.9.3	Controlling Stepper Motors 674
17.9.4	Motor Drive Circuitry 676
17.9.5	Motor Drive Noise 678
17.10	LEDs and LED Displays 678
17.10.1	Individual LEDs 678
17.10.2	Multi-LED Displays 679
17.11	Measuring Digital Signals 681
17.11.1	The Approach 681
17.11.2	Working with Asynchronous Signals 682
17.11.3	Buffering Input Signals 683
17.11.4	Inside vs. Outside 683
17.11.5	Measuring Frequency and Time Interval 683
17.12	Summary 687
17.13	Review Questions 688
17.14	Thought Questions 688
17.15	Problems 689

18 Working Outside of the Processor III: Interfacing to Remote Devices

693

18.0	Common Network-Based I/O Architectures 693
18.1	Network-Based Systems 694
18.2	RS-232/EIA-232—Asynchronous Serial Communication 694
18.2.1	Introduction 694
18.2.2	The EIA-232 Standard 695
	What It Is . . . 696
	What They Think It Is . . . 696
18.2.3	EIA-232 Addressing 698
18.2.4	Asynchronous Serial Communication 698
18.2.5	Configuring the Interface 698

	18.2.6	Data Recovery and Timing	699
	18.2.7	EIA-232 Interface Signals	700
	18.2.8	An Implementation	700
18.3		The Universal Serial Bus—Synchronous Serial Communication	701
	18.3.1	Background	702
	18.3.2	The Universal Serial Bus Architecture	702
	18.3.3	The Universal Serial Bus Protocol	702
	18.3.4	USB Devices	704
	18.3.5	Transfer Types	704
	18.3.6	Device Descriptors	704
	18.3.7	Network Configuration	705
	18.3.8	USB Transactions	706
	18.3.9	USB Interface Signals	707
	18.3.10	The Physical Environment	708
		Low-speed Cables	708
		High-speed Cables	708
		Cables and Cable Power	709
	18.3.11	Detecting Device Attachment and Speed	709
	18.3.12	Differential Pair Signaling	710
	18.3.13	Implementation	710
18.4		I ² C—A Local Area Network	710
	18.4.1	The Architecture	711
	18.4.2	Electrical Considerations	712
	18.4.3	Basic Operation	712
	18.4.4	Flow of Control	713
	18.4.5	Multiple Masters	713
		Arbitration	714
		Synchronization	714
	18.4.6	Using the I ² C Bus	715
18.5		The Controller Area Network—The CAN Bus	715
	18.5.1	The Architecture	715
	18.5.2	Electrical Considerations	716
	18.5.3	Message Types	716
	18.5.4	Message Format	717
	18.5.5	Basic Operation	718
		Synchronization	718
		Error Management	718
		Transmission and Arbitration	719
	18.5.6	Using the CAN Bus	719
18.6		Summary	720
18.7		Review Questions	720
18.8		Thought Questions	721
18.9		Problems	721

19 Programmable Logic Devices

723

19.0	Introduction	723
19.1	Why Use Programmable Logic Devices?	724
19.2	Basic Concepts	725
19.3	Basic Configurations	728
	19.3.1	The (P)ROM 728
	19.3.2	Programmable Array Logic—PAL 728
	19.3.3	Programmable Logic Array—PLA 729

19.3.4	Programmable Logic Sequencer—PLS	729
19.3.5	PLA vs. PAL vs. (P)ROM	729
19.4	Programmable and Reprogrammable Technologies	730
19.4.1	Programmable Technologies	730
19.4.2	Reprogrammable Technologies	730
19.5	Architectures	732
19.5.1	PLDs	732
19.5.2	CPLD	734
19.5.3	FPGAs	735
19.6	Example PLD Devices	737
19.6.1	Xilinx XC 9500XL™ Family	737
19.6.2	Altera Flex 10K™ FPGA	739
19.6.3	Input/Output and Internal Interconnection	743
19.6.4	Cypress CY8C21x23 PSOC™ ⁷⁴⁴	
	The CPU Core	745
	The Digital Subsystem	745
	The Analog Subsystem	745
	The System Resources	746
19.7	The Design Process	746
19.8	Design Examples	747
19.8.1	Lane Departure Detection Implementation and Acceleration	747
19.8.2	Local Area Tracking System—LATS with uCLinux™ OS	750
19.9	Summary	752
19.10	Review Questions	752
19.11	Thought Questions	753

Appendix A Verilog Overview: The Verilog Hardware Description Language 754

A.0	Introduction	754
A.1	An Overview of a Verilog Program	755
A.2	Creating a Verilog Program	755
A.2.1	Some Concepts in a Verilog Source File	756
A.2.2	Modules	756
	Module Name	756
	Inputs and Outputs Declarations	757
	Nets and Variables	757
	Declaring Multi-Bit Signals	758
	Subsets of Multi-Bit Expressions	759
	\$display and \$monitor Statements	759
	\$stop and \$finish Statements	760
	\$time Statement	760
A.3	Three Models—The Gate Level, the Dataflow, and the Behavioral	761
A.3.1	The Structural/Gate-Level Model	761
	Combinational Logic	762
A.3.2	The Dataflow Model	767
A.3.3	The Behavioral Model	772
A.4	Testing and Verifying the Circuit	781
A.4.1	The Circuit Module	782
A.4.2	The Test Module	782
	Clocks and Resets	784
A.4.3	The Test Bench	784
A.4.4	Performing the Simulation	784
A.5	Summary	785

Appendix B Laboratory Projects

Available for instructors on the book companion site

- B.0 Introduction
- B.1 Getting Started
- B.2 Developing Skills
- B.3 Bringing It Together

References

786

Index

793



Embedded Systems Design!

From operating our cars, to controlling the elevators we ride, to doing our laundry or cooking our dinner, the special computers we call embedded systems are quietly and unobtrusively doing their jobs. Embedded systems give us the ability to put increasingly large amounts of capability into ever-smaller devices. *Embedded Systems: A Contemporary Design Tool* introduces you to the theoretical hardware and software foundations of these systems and shows you how to apply embedded systems concepts to design practical applications that solve real-world challenges.

Taking the user's problem and needs as your starting point, you'll delve into each of the key theoretical and practical aspects to consider when designing an application. Author James Peckol walks you through the formal hardware and software development process, covering:

- How to break the problem down into major functional blocks;
- Planning the digital and software architecture of the system;
- Designing the physical world interface to external analog and digital signals;
- Debugging and testing throughout the development cycle;
- Improving performance.

Stressing the importance of safety and reliability in the design and development of embedded systems and providing a balanced treatment of both the hardware and software aspects, *Embedded Systems* gives you the right tools for developing safe, reliable, and robust solutions in a wide range of applications.

James K. Peckol is Senior Lecturer in the Department of Electrical Engineering at the University of Washington - Seattle, where he has twice been named Teacher of the Year. He is also the founder of Oxford Consulting, Ltd., a product design and development consulting firm.



WILEY

1807-2007 KNOWLEDGE FOR GENERATIONS

www.wiley.com/college/peckol

ISBN 978-0-471-72180-2



9 780471 721802

90000

