

Contents

About the Authors	x
Introduction	xiii
Chapter 1: Basic Embedded Programming Concepts	1
1.1 Numbering Systems	2
1.2 Signed Binary Numbers	5
1.3 Data Structures	13
1.4 Communications Protocols	29
1.5 Mathematics	37
1.6 Numeric Comparison	46
1.7 State Machines	59
1.8 Multitasking	74
Chapter 2: Device Drivers	85
2.1 In This Chapter	85
2.2 Example 1: Device Drivers for Interrupt-Handling	89
2.3 Example 2: Memory Device Drivers	110
2.4 Example 3: Onboard Bus Device Drivers	134
2.5 Board I/O Driver Examples	143
2.6 Summary	168
Chapter 3: Embedded Operating Systems	169
3.1 In This Chapter	169
3.2 What Is a Process?	175
3.3 Multitasking and Process Management	177
3.4 Memory Management	213
3.5 I/O and File System Management	230
3.6 OS Standards Example: POSIX (Portable Operating System Interface)	232
3.7 OS Performance Guidelines	235
3.8 OSes and Board Support Packages (BSPs)	237
3.9 Summary	239

Chapter 4: Networking	241
Introduction to the RCM3200 Rabbit Core	243
Introduction to the Dynamic C Development Environment	244
Brief Introduction to Dynamic C Libraries	246
Memory Spaces in Dynamic C	247
How Code Is Compiled and Run	256
Setting Up a PC as an RCM3200 Development System	259
Time to Start Writing Code!	259
Embedded Networks	274
Dynamic C Support for Networking Protocols	275
Typical Network Setup	279
Setting Up a Core Module's Network Configuration	282
Project 1: Bringing Up a Rabbit Core Module for Networking	288
The Client Server Paradigm	293
The Berkeley Sockets Interface	294
Using TCP versus UDP in an Embedded Application	298
Important Dynamic C Library Functions for Socket Programming	300
Project 2: Implementing a Rabbit TCP/IP Server	303
Project 3: Implementing a Rabbit TCP/IP Client	311
Project 4: Implementing a Rabbit UDP Server	322
Some Useful (and Free!) Networking Utilities	328
Final Thought	331
Chapter 5: Error Handling and Debugging	333
The Zen of Embedded Systems Development and Troubleshooting	333
Avoid Debugging Altogether—Code Smart	340
Proactive Debugging	341
Stacks and Heaps	342
Seeding Memory	344
Wandering Code	346
Special Decoders	347
MMUs	348
Conclusion	349
Implementing Downloadable Firmware with Flash Memory	350
The Microprogrammer	350
Advantages of Microprogrammers	351
Disadvantages of Microprogrammers	351
Receiving a Microprogrammer	352
A Basic Microprogrammer	354
Common Problems and Their Solutions	355
Hardware Alternatives	362
Memory Diagnostics	364
ROM Tests	365

5.20 RAM Tests	367
5.21 Nonvolatile Memory	372
5.22 Supervisory Circuits	372
5.23 Multibyte Writes	374
5.24 Testing	375
5.25 Conclusion	375
5.26 Building a Great Watchdog	377
5.27 Internal WDTs	381
5.28 External WDTs	381
5.29 Characteristics of Great WDTs	381
5.30 Using an Internal WDT	381
5.31 An External WDT	391
5.32 WDTs for Multitasking	391
5.33 Summary and Other Thoughts	391
Chapter 6: Hardware/Software Co-Verification	39
6.1 Embedded System Design Process	39
6.2 Verification and Validation	40
6.3 Human Interaction	40
6.4 Co-Verification	40
Chapter 7: Techniques for Embedded Media Processing	44
7.1 A Simplified Look at a Media Processing System	44
7.2 System Resource Partitioning and Code Optimization	45
7.3 Event Generation and Handling	45
7.4 Programming Methodology	45
7.5 Architectural Features for Efficient Programming	45
7.6 Compiler Considerations for Efficient Programming	46
7.7 System and Core Synchronization	47
7.8 Memory Architecture—the Need for Management	47
7.9 Physics of Data Movement	48
7.10 Media Processing Frameworks	49
7.11 Defining Your Framework	50
7.12 Asymmetric and Symmetric Dual-Core Processors	50
7.13 Programming Models	51
7.14 Strategies for Architecting a Framework	51
7.15 Other Topics in Media Frameworks	51
Chapter 8: DSP in Embedded Systems	52
8.1 Overview of Embedded Systems and Real-Time Systems	52
8.2 Real-Time Systems	52
8.3 Hard Real-Time and Soft Real-Time Systems	52
8.4 Efficient Execution and the Execution Environment	52

5	Challenges in Real-Time System Design	542
6	Summary	553
7	Overview of Embedded Systems Development Life Cycle Using DSP	554
8	The Embedded System Life Cycle Using DSP	554
9	Optimizing DSP Software	580
10	What Is Optimization?	580
11	The Process	581
12	Make the Common Case Fast	584
13	Make the Common Case Fast—DSP Architectures	584
14	Make the Common Case Fast—DSP Algorithms	587
15	Make the Common Case Fast—DSP Compilers	588
16	An In-Depth Discussion of DSP Optimization	595
17	Direct Memory Access	595
18	Using DMA	596
19	Loop Unrolling	604
20	Software Pipelining	610
21	More on DSP Compilers and Optimization	620
22	Programmer Helping Out the Compiler	633
23	Profile-Based Compilation	646
24	References	653

Chapter 9: Practical Embedded Coding Techniques..... 655

1	Reentrancy	655
2	Atomic Variables	656
3	Two More Rules	658
4	Keeping Code Reentrant	659
5	Recursion	661
6	Asynchronous Hardware/Firmware	661
7	Race Conditions	662
8	Options	664
9	Other RTOSes	665
10	Metastable States	666
11	Firmware, Not Hardware	668
12	Interrupt Latency	671
13	Taking Data	674
14	Understanding Your C Compiler: How to Minimize Code Size	677
15	Modern C Compilers	677
16	Tips on Programming	687
17	Final Notes	695
18	Acknowledgments	696

Chapter 10: Development Technologies and Trends 6

10.1	How to Choose a CPU for Your System on Chip Design	6
10.2	Emerging Technology for Embedded Systems Software Development	6
10.3	Making Development Tool Choices	6
10.4	Eclipse—Bringing Embedded Tools Together	6
10.5	Embedded Software and UML	6
10.6	Model-based Systems Development with xtUML	6
10.7	The Future	6

Index 7