CONTENTS

Preface xv
Acknowledgments xxi

1 Fundamentals of Real-Time Systems 1
   1.1 Concepts and Misconceptions, 2
       1.1.1 Definitions for Real-Time Systems, 2
       1.1.2 Usual Misconceptions, 14
   1.2 Multidisciplinary Design Challenges, 15
       1.2.1 Influencing Disciplines, 16
   1.3 Birth and Evolution of Real-Time Systems, 16
       1.3.1 Diversifying Applications, 17
       1.3.2 Advancements behind Modern Real-Time Systems, 19
   1.4 Summary, 21
   1.5 Exercises, 24
       References, 25

2 Hardware for Real-Time Systems 27
   2.1 Basic Processor Architecture, 28
       2.1.1 Von Neumann Architecture, 29
       2.1.2 Instruction Processing, 30
       2.1.3 Input/Output and Interrupt Considerations, 33
   2.2 Memory Technologies, 36
       2.2.1 Different Classes of Memory, 36
       2.2.2 Memory Access and Layout Issues, 38
       2.2.3 Hierarchical Memory Organization, 41
2.3 Architectural Advancements, 43
  2.3.1 Pipelined Instruction Processing, 45
  2.3.2 Superscalar and Very Long Instruction Word Architectures, 46
  2.3.3 Multi-Core Processors, 48
  2.3.4 Complex Instruction Set versus Reduced Instruction Set, 50
2.4 Peripheral Interfacing, 52
  2.4.1 Interrupt-Driven Input/Output, 53
  2.4.2 Direct Memory Access, 56
  2.4.3 Analog and Digital Input/Output, 58
2.5 Microprocessor versus Microcontroller, 62
  2.5.1 Microprocessors, 62
  2.5.2 Standard Microcontrollers, 64
  2.5.3 Custom Microcontrollers, 66
2.6 Distributed Real-Time Architectures, 68
  2.6.1 Fieldbus Networks, 68
  2.6.2 Time-Triggered Architectures, 71
2.7 Summary, 73
2.8 Exercises, 74
References, 76

3 Real-Time Operating Systems 79

3.1 From Pseudokernels to Operating Systems, 80
  3.1.1 Miscellaneous Pseudokernels, 82
  3.1.2 Interrupt-Only Systems, 87
  3.1.3 Preemptive Priority Systems, 90
  3.1.4 Hybrid Scheduling Systems, 90
  3.1.5 The Task Control Block Model, 95
3.2 Theoretical Foundations of Scheduling, 97
  3.2.1 Scheduling Framework, 98
  3.2.2 Round-Robin Scheduling, 99
  3.2.3 Cyclic Code Scheduling, 100
  3.2.4 Fixed-Priority Scheduling: Rate-Monotonic Approach, 102
  3.2.5 Dynamic Priority Scheduling: Earliest Deadline First Approach, 104
3.3 System Services for Application Programs, 106
  3.3.1 Linear Buffers, 107
  3.3.2 Ring Buffers, 109
  3.3.3 Mailboxes, 110
  3.3.4 Semaphores, 112
  3.3.5 Deadlock and Starvation Problems, 114
  3.3.6 Priority Inversion Problem, 118
3.3.7 Timer and Clock Services, 122
3.3.8 Application Study: A Real-Time Structure, 123
3.4 Memory Management Issues, 127
  3.4.1 Stack and Task Control Block Management, 127
  3.4.2 Multiple-Stack Arrangement, 128
  3.4.3 Memory Management in the Task Control Block Model, 129
  3.4.4 Swapping, Overlaying, and Paging, 130
3.5 Selecting Real-Time Operating Systems, 133
  3.5.1 Buying versus Building, 134
  3.5.2 Selection Criteria and a Metric for Commercial Real-Time Operating Systems, 135
  3.5.3 Case Study: Selecting a Commercial Real-Time Operating System, 138
  3.5.4 Supplementary Criteria for Multi-Core and Energy-Aware Support, 140
3.6 Summary, 142
3.7 Exercises, 143
References, 146

4 Programming Languages for Real-Time Systems 149
  4.1 Coding of Real-Time Software, 150
    4.1.1 Fitness of a Programming Language for Real-Time Applications, 151
    4.1.2 Coding Standards for Real-Time Software, 152
  4.2 Assembly Language, 154
  4.3 Procedural Languages, 156
    4.3.1 Modularity and Typing Issues, 156
    4.3.2 Parameter Passing and Dynamic Memory Allocation, 157
    4.3.3 Exception Handling, 159
    4.3.4 Cardelli’s Metrics and Procedural Languages, 161
  4.4 Object-Oriented Languages, 162
    4.4.1 Synchronizing Objects and Garbage Collection, 162
    4.4.2 Cardelli’s Metrics and Object-Oriented Languages, 164
    4.4.3 Object-Oriented versus Procedural Languages, 165
  4.5 Overview of Programming Languages, 167
    4.5.1 Ada, 167
    4.5.2 C, 169
    4.5.3 C++, 170
    4.5.4 C#, 171
    4.5.5 Java, 172
    4.5.6 Real-Time Java, 174
    4.5.7 Special Real-Time Languages, 177
4.6 Automatic Code Generation, 178
   4.6.1 Toward Production-Quality Code, 178
   4.6.2 Remaining Challenges, 180
4.7 Compiler Optimizations of Code, 181
   4.7.1 Standard Optimization Techniques, 182
   4.7.2 Additional Optimization Considerations, 188
4.8 Summary, 192
4.9 Exercises, 193

References, 195

5 Requirements Engineering Methodologies, 197
5.1 Requirements Engineering for Real-Time Systems, 198
   5.1.1 Requirements Engineering as a Process, 198
   5.1.2 Standard Requirement Classes, 199
   5.1.3 Specification of Real-Time Software, 201
5.2 Formal Methods in System Specification, 202
   5.2.1 Limitations of Formal Methods, 205
   5.2.2 Finite State Machines, 205
   5.2.3 Statecharts, 210
   5.2.4 Petri Nets, 213
5.3 Semiformal Methods in System Specification, 217
   5.3.1 Structured Analysis and Structured Design, 218
   5.3.2 Object-Oriented Analysis and the Unified
        Modeling Language, 221
   5.3.3 Recommendations on Specification Approach, 224
5.4 The Requirements Document, 225
   5.4.1 Structuring and Composing Requirements, 226
   5.4.2 Requirements Validation, 228
5.5 Summary, 232
5.6 Exercises, 233
5.7 Appendix 1: Case Study in Software Requirements
        Specification, 235
   5.7.1 Introduction, 235
   5.7.2 Overall Description, 238
   5.7.3 Specific Requirements, 245
   References, 265

6 Software Design Approaches, 267
6.1 Qualities of Real-Time Software, 268
   6.1.1 Eight Qualities from Reliability to Verifiability, 269
6.2 Software Engineering Principles, 275
   6.2.1 Seven Principles from Rigor and Formality
        to Traceability, 275
   6.2.2 The Design Activity, 281

References, 265
6.3 Procedural Design Approach, 284
   6.3.1 Parnas Partitioning, 284
   6.3.2 Structured Design, 286
   6.3.3 Design in Procedural Form Using Finite State Machines, 292
6.4 Object-Oriented Design Approach, 293
   6.4.1 Advantages of Object Orientation, 293
   6.4.2 Design Patterns, 295
   6.4.3 Design Using the Unified Modeling Language, 298
   6.4.4 Object-Oriented versus Procedural Approaches, 301
6.5 Life Cycle Models, 302
   6.5.1 Waterfall Model, 303
   6.5.2 V-Model, 305
   6.5.3 Spiral Model, 306
   6.5.4 Agile Methodologies, 307
6.6 Summary, 311
6.7 Exercises, 312
6.8 Appendix 1: Case Study in Designing Real-Time Software, 314
   6.8.1 Introduction, 314
   6.8.2 Overall Description, 315
   6.8.3 Design Decomposition, 316
   6.8.4 Requirements Traceability, 371
References, 375

7 Performance Analysis Techniques

7.1 Real-Time Performance Analysis, 380
   7.1.1 Theoretical Preliminaries, 380
   7.1.2 Arguments Related to Parallelization, 382
   7.1.3 Execution Time Estimation from Program Code, 385
   7.1.4 Analysis of Polled-Loop and Coroutine Systems, 391
   7.1.5 Analysis of Round-Robin Systems, 392
   7.1.6 Analysis of Fixed-Period Systems, 394
   7.1.7 Analysis of Nonperiodic Systems, 396
7.2 Applications of Queuing Theory, 398
   7.2.1 Single-Server Queue Model, 398
   7.2.2 Arrival and Processing Rates, 400
   7.2.3 Buffer Size Calculation, 401
   7.2.4 Response Time Modeling, 402
   7.2.5 Other Results from Queuing Theory, 403
7.3 Input/Output Performance, 405
   7.3.1 Buffer Size Calculation for Time-Invariant Bursts, 405
   7.3.2 Buffer Size Calculation for Time-Variant Bursts, 406
7.4 Analysis of Memory Requirements, 408
   7.4.1 Memory Utilization Analysis, 408
   7.4.2 Optimizing Memory Usage, 410
7.5 Summary, 411
7.6 Exercises, 413
References, 415

8 Additional Considerations for the Practitioner 417

8.1 Metrics in Software Engineering, 418
   8.1.1 Lines of Source Code, 419
   8.1.2 Cyclomatic Complexity, 420
   8.1.3 Halstead’s Metrics, 421
   8.1.4 Function Points, 423
   8.1.5 Feature Points, 427
   8.1.6 Metrics for Object-Oriented Software, 428
   8.1.7 Criticism against Software Metrics, 428
8.2 Predictive Cost Modeling, 429
   8.2.1 Basic COCOMO 81, 429
   8.2.2 Intermediate and Detailed COCOMO 81, 431
   8.2.3 COCOMO II, 433
8.3 Uncertainty in Real-Time Systems, 433
   8.3.1 The Three Dimensions of Uncertainty, 434
   8.3.2 Sources of Uncertainty, 435
   8.3.3 Identifying Uncertainty, 437
   8.3.4 Dealing with Uncertainty, 438
8.4 Design for Fault Tolerance, 438
   8.4.1 Spatial Fault-Tolerance, 440
   8.4.2 Software Black Boxes, 443
   8.4.3 N-Version Programming, 443
   8.4.4 Built-in-Test Software, 444
   8.4.5 Spurious and Missed Interrupts, 447
8.5 Software Testing and Systems Integration, 447
   8.5.1 Testing Techniques, 448
   8.5.2 Debugging Approaches, 454
   8.5.3 System-Level Testing, 456
   8.5.4 Systems Integration, 458
   8.5.5 Testing Patterns and Exploratory Testing, 462
8.6 Performance Optimization Techniques, 465
   8.6.1 Scaled Numbers for Faster Execution, 465
   8.6.2 Look-Up Tables for Functions, 467
   8.6.3 Real-Time Device Drivers, 468
8.7 Summary, 470
8.8 Exercises, 471
References, 473
9 Future Visions on Real-Time Systems 477

9.1 Vision: Real-Time Hardware, 479
   9.1.1 Heterogeneous Soft Multi-Cores, 481
   9.1.2 Architectural Issues with Individual Soft Cores, 483
   9.1.3 More Advanced Fieldbus Networks and Simpler Distributed Nodes, 484

   9.2.1 One Coordinating System Task and Multiple Isolated Application Tasks, 486
   9.2.2 Small, Platform Independent Virtual Machines, 487

9.3 Vision: Real-Time Programming Languages, 488
   9.3.1 The UML++ as a Future “Programming Language”, 489

9.4 Vision: Real-Time Systems Engineering, 491
   9.4.1 Automatic Verification of Software, 491
   9.4.2 Conservative Requirements Engineering, 492
   9.4.3 Distance Collaboration in Software Projects, 492
   9.4.4 Drag-and-Drop Systems, 493

9.5 Vision: Real-Time Applications, 493
   9.5.1 Local Networks of Collaborating Real-Time Systems, 494
   9.5.2 Wide Networks of Collaborating Real-Time Systems, 495
   9.5.3 Biometric Identification Device with Remote Access, 495
   9.5.4 Are There Any Threats behind High-Speed Wireless Communications?, 497

9.6 Summary, 497
9.7 Exercises, 499

References, 500

Glossary 503

About the Authors 535

Index 537