Real-Time Systems

Second Edition

Real-Time Systems Series

Series Editor

John A. Stankovic
University of Virginia, Virginia, USA

For further volumes:
http://www.springer.com/series/6941

Hermann Kopetz

Real-Time Systems

Design Principles for Distributed Embedded Applications

Second Edition

Springer

Hermann Kopetz
Vienna University of Technology
Department of Computer Engineering
Real Time Systems Group
Treitlstrasse 3, 3rd floor
1040 Wien, Austria
hk@vmars.tuwien.ac.at

ISSN 1867-321X e-ISSN 1867-3228
ISBN 978-1-4419-8236-0 e-ISBN 978-1-4419-8237-7
DOI 10.1007/978-1-4419-8237-7
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2011925551

© Springer Science+Business Media, LLC 2011
All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.
The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The primary objective of this book is to serve as a textbook for students who take a senior undergraduate or a first-year graduate course on real-time embedded systems, also called *cyber-physical systems*. The structure of the book – the material is organized into 14 chapters – maps to the 14 weeks of a semester. The book is also intended for practitioners in industry who want to learn about the state of the art in real-time embedded system design and need a reference book that explains the fundamental concepts of the field. More than 1,000 students used the first edition of this book, published about 14 years ago, as a text for the real-time systems course at the Vienna University of Technology. The feedback from these students and many new developments in this dynamic field of embedded real-time systems have been incorporated in this fundamentally revised second edition of the book. The focus of the book is on the design of distributed real-time systems at the architecture level. While a significant part of the established computer science literature abstracts from the progression of real-time, real-time system designers cannot get away with such an abstraction. In this book, the progression of physical time is considered a first-order citizen that shapes many of the relevant concepts. The book explains the fundamental concepts related to the progression of time on a number of practical insightful examples from industry. The conceptual model of a distributed real-time distributed system has been extended and precise definitions of important time-related concepts, such as *sparse time*, *state*, *temporal accuracy of real-time data*, and *determinism* are given.

Since the evolving cognitive complexity of large computer systems is a topic of utmost concern, a new chapter on *simplicity* has been included in this second edition. This chapter builds on some of the recent insights from the field of cognition – concerning concept formation, understanding, human simplification strategies and model building – and formulates seven principles that lead to the design of *simple* systems. These principles are followed in the remaining 12 chapters of the book. The other two new chapters, one on *energy and power awareness*, and one on the *Internet of things* cover topics of increasing importance in the enormous market of mobile devices. The chapters on *communication*, *dependability*, *system design*, and *validation* have been substantially revised with a focus on *component-based* and *model-based design*. The chapter on dependability includes new sections on *security* and *safety*. The final chapter describes the *time-triggered architecture* that integrates all presented concepts into a coherent framework for the development of dependable embedded real-time systems. Since the first edition of the book has been published, a visible paradigm shift from the event-triggered to the time-triggered design methodology for dependable distributed real-time systems has taken place in a number of applications.

It is assumed that the reader of this book has a background in basic computer science or computer engineering or has some practical experience in the design or implementation of embedded systems.

The glossary at the end of the book is an integral part of the book, providing definitions for many of the technical terms that are used throughout the book. If the reader is not sure about the meaning of a term, she/he is advised to refer to the glossary.

Acknowledgements

It is impossible to name all students, colleagues from industry and fellow scientists who have contributed to this second edition of the book by asking intelligent questions or making constructive comments over the last decade – thanks to all of you. In the final stages of finishing the manuscript of this second edition, in October 2010, I have given a course at Vanderbilt University, organized by Janos Sztipanovits, and got valuable comments form an unbiased audience. I am especially grateful to Christian Tessarek who did the artwork, and the following persons who have read part or all of the evolving manuscript and made many valuable suggestions for improvement: Sven Bünte, Christian El-Salloum, Bernhard Frömel, Oliver Höftberger, Herbert Grünbacher, Benedikt Huber, Albrecht Kadlec, Roland Kammerer, Susanne Kandl, Vaclav Mikolasek, Stefan Poledna, Peter Puschner, Brian Randell, Andreas Steininger, Ekarin Suethanuwong, Armin Wasicek, Michael Zolda, and the following students from Vanderbilt: Kyoungho An, Joshua D. Carl, Spencer Crosswy, Fred Eisele, Fan Qui, and Adam C. Trewyn.

Vienna, Austria
January 2011

Hermann Kopetz

Contents

1 The Real-Time Environment 1

1.1 When Is a Computer System Real-Time? 2
1.2 Functional Requirements 3
1.2.1 Data Collection 3
1.2.2 Direct Digital Control 5
1.2.3 Man–Machine Interaction 6
1.3 Temporal Requirements 6
1.3.1 Where Do Temporal Requirements Come from? 6
1.3.2 Minimal Latency Jitter 9
1.3.3 Minimal Error-Detection Latency 10
1.4 Dependability Requirements 10
1.4.1 Reliability 10
1.4.2 Safety 11
1.4.3 Maintainability 12
1.4.4 Availability 12
1.4.5 Security 13
1.5 Classification of Real-Time Systems 13
1.5.1 Hard Real-Time System Versus Soft Real-Time System 13
1.5.2 Fail-Safe Versus Fail-Operational 15
1.5.3 Guaranteed-Response Versus Best-Effort 16
1.5.4 Resource-Adequate Versus Resource-Inadequate 16
1.5.5 Event-Triggered Versus Time-Triggered 16
1.6 The Real-Time Systems Market 17
1.6.1 Embedded Real-Time Systems 18
1.6.2 Plant Automation Systems 20
1.6.3 Multimedia Systems 21

1.7 Examples of Real-Time Systems 22
1.7.1 Controlling the Flow in a Pipe 22
1.7.2 Engine Control 23
1.7.3 Rolling Mill 24
Points to Remember 25
Bibliographic Notes 26
Review Questions and Problems 27

2 Simplicity 29

2.1 Cognition 30
2.1.1 Problem Solving 30
2.1.2 Definition of a Concept 32
2.1.3 Cognitive Complexity 33
2.1.4 Simplification Strategies 34
2.2 The Conceptual Landscape 35
2.2.1 Concept Formation 36
2.2.2 Scientific Concepts 37
2.2.3 The Concept of a Message 38
2.2.4 Semantic Content of a Variable 39
2.3 The Essence of Model Building 40
2.3.1 Purpose and Viewpoint 41
2.3.2 The Grand Challenge 42
2.4 Emergence 43
2.4.1 Irreducibility 43
2.4.2 Prior and Derived Properties 44
2.4.3 Complex Systems 44
2.5 How Can We Achieve Simplicity? 45
Points to Remember 47
Bibliographic Notes 49
Review Questions and Problems 49

3 Global Time 51

3.1 Time and Order 52
3.1.1 Different Orders 52
3.1.2 Clocks 53
3.1.3 Precision and Accuracy 55
3.1.4 Time Standards 56
3.2 Time Measurement 57
3.2.1 Global Time 58
3.2.2 Interval Measurement 59
3.2.3 π/Δ-Precedence 60
3.2.4 Fundamental Limits of Time Measurement 61
3.3 Dense Time Versus Sparse Time 62
3.3.1 Dense Time-Base 63
3.3.2 Sparse Time-Base 64

3.3.3 Space-Time Lattice 64
3.3.4 Cyclic Representation of Time 65
3.4 Internal Clock Synchronization.................................... 66
3.4.1 The Synchronization Condition............................ 66
3.4.2 Central Master Synchronization 68
3.4.3 Fault-Tolerant Synchronization Algorithms................ 69
3.4.4 State Correction Versus Rate Correction.................. 72
3.5 External Clock Synchronization.................................. 73
3.5.1 External Time Sources.................................... 73
3.5.2 Time Gateway.. 74
3.5.3 Time Formats 75
Points to Remember... 76
Bibliographic Notes 77
Review Questions and Problems................................. 77

4 Real-Time Model 79

4.1 Model Outline.. 80
4.1.1 Components and Messages 80
4.1.2 Cluster of Components.................................. 81
4.1.3 Temporal Control Versus Logical Control 82
4.1.4 Event-Triggered Control Versus Time-Triggered Control 84
4.2 Component State... 84
4.2.1 Definition of State... 84
4.2.2 The Pocket Calculator Example 85
4.2.3 Ground State 86
4.2.4 Database Components 88
4.3 The Message Concept... 88
4.3.1 Message Structure... 88
4.3.2 Event Information Versus State Information............. 89
4.3.3 Event-Triggered Message.................................. 90
4.3.4 Time-Triggered Message 91
4.4 Component Interfaces... 92
4.4.1 Interface Characterization................................ 93
4.4.2 Linking Interface.. 94
4.4.3 Technology Independent Control Interface................ 94
4.4.4 Technology Dependent Debug Interface 95
4.4.5 Local Interfaces 95
4.5 Gateway Component.. 95
4.5.1 Property Mismatches...................................... 96
4.5.2 LIF Versus Local Interface of a Gateway Component.... 97
4.5.3 Standardized Message Interface 98
4.6 Linking Interface Specification................................. 99
4.6.1 Transport Specification 99
4.6.2 Operational Specification 100
4.6.3 Meta-Level Specification 101

4.7 Component Integration 102
4.7.1 Principles of Composability 102
4.7.2 Integration Viewpoints 104
4.7.3 System of Systems 104
Points to Remember 107
Bibliographic Notes 108
Review Questions and Problems 109

5 Temporal Relations 111

5.1 Real-Time Entities 112
5.1.1 Sphere of Control 112
5.1.2 Discrete and Continuous Real-Time Entities 112
5.2 Observations 113
5.2.1 Untimed Observation 113
5.2.2 Indirect Observation 114
5.2.3 State Observation 114
5.2.4 Event Observation 114
5.3 Real-Time Images and Real-Time Objects 115
5.3.1 Real-Time Images 115
5.3.2 Real-Time Objects 116
5.4 Temporal Accuracy 116
5.4.1 Definition 116
5.4.2 Classification of Real-Time Images 119
5.4.3 State Estimation 120
5.4.4 Composability Considerations 121
5.5 Permanence and Idempotency 122
5.5.1 Permanence 122
5.5.2 Duration of the Action Delay 123
5.5.3 Accuracy Interval Versus Action Delay 124
5.5.4 Idempotency 124
5.6 Determinism 125
5.6.1 Definition of Determinism 125
5.6.2 Consistent Initial States 127
5.6.3 Non-deterministic Design Constructs 128
5.6.4 Recovery of Determinism 130
Points to Remember 130
Bibliographic Notes 132
Review Questions and Problems 132

6 Dependability 135

6.1 Basic Concepts 136
6.1.1 Faults 136
6.1.2 Errors 138
6.1.3 Failures 139

6.2 Information Security 141
6.2.1 Secure Information Flow 142
6.2.2 Security Threats 143
6.2.3 Cryptographic Methods 145
6.2.4 Network Authentication 148
6.2.5 Protection of Real-Time Control Data 149
6.3 Anomaly Detection 150
6.3.1 What Is an Anomaly? 150
6.3.2 Failure Detection 151
6.3.3 Error Detection 152
6.4 Fault Tolerance 153
6.4.1 Fault Hypotheses 154
6.4.2 Fault-Tolerant Unit 155
6.4.3 The Membership Service 158
6.5 Robustness 160
6.5.1 The Concept of Robustness 160
6.5.2 Structure of a Robust System 160
6.6 Component Reintegration 161
6.6.1 Finding a Reintegration Point 162
6.6.2 Minimizing the Ground-State 162
6.6.3 Component Restart 163
Points to Remember 164
Bibliographic Notes 165
Review Questions and Problems 166

7 Real-Time Communication 167

7.1 Requirements 168
7.1.1 Timeliness 168
7.1.2 Dependability 168
7.1.3 Flexibility 171
7.1.4 Physical Structure 171
7.2 Design Issues 172
7.2.1 A Waistline Communication Model 172
7.2.2 Physical Performance Limitation 174
7.2.3 Flow Control 174
7.2.4 Thrashing 176
7.3 Event-Triggered Communication 178
7.3.1 Ethernet 179
7.3.2 Controller Area Network 179
7.3.3 User Datagram Protocol 180
7.4 Rate-Constrained Communication 180
7.4.1 Token Protocol 181
7.4.2 Mini-slotting Protocol ARINC 629 181

7.4.3 Avionics Full Duplex Switched Ethernet 182
7.4.4 Audio Video Bus 182
7.5 Time-Triggered Communication 183
7.5.1 Time-Triggered Protocol 184
7.5.2 Time-Triggered Ethernet 186
7.5.3 FlexRay 187
Points to Remember 187
Bibliographic Notes 188
Review Questions and Problems 188

8 Power and Energy Awareness 191

8.1 Power and Energy 192
8.1.1 Basic Concepts 192
8.1.2 Energy Estimation 193
8.1.3 Thermal Effects and Reliability 197
8.2 Hardware Power Reduction Techniques 200
8.2.1 Device Scaling 200
8.2.2 Low-Power Hardware Design 201
8.2.3 Voltage and Frequency Scaling 202
8.2.4 Sub-threshold Logic 202
8.3 System Architecture 203
8.3.1 Technology-Agnostic Design 203
8.3.2 Pollack's Rule 204
8.3.3 Power Gating 205
8.3.4 Real Time Versus Execution Time 206
8.4 Software Techniques 207
8.4.1 System Software 207
8.4.2 Application Software 208
8.4.3 Software Tools 209
8.5 Energy Sources 209
8.5.1 Batteries 209
8.5.2 Energy Harvesting 210
Points to Remember 211
Bibliographic Notes 212
Review Questions and Problems 212

9 Real-Time Operating Systems 215

9.1 Inter-Component Communication 216
9.1.1 Technology Independent Interface 216
9.1.2 Linking Interface 216
9.1.3 Technology Dependent Debug Interface 217
9.1.4 Generic Middleware 217
9.2 Task Management 218
9.2.1 Simple Tasks 218

9.2.2 Trigger Tasks 220
9.2.3 Complex Tasks 220
9.3 The Dual Role of Time 221
9.3.1 Time as Data 222
9.3.2 Time as Control 223
9.4 Inter-task Interactions 223
9.4.1 Coordinated Static Schedules 224
9.4.2 The Non-blocking Write Protocol 224
9.4.3 Semaphore Operations 225
9.5 Process Input/Output 226
9.5.1 Analog Input/Output 226
9.5.2 Digital Input/Output 227
9.5.3 Interrupts 228
9.5.4 Fault-Tolerant Actuators 229
9.5.5 Intelligent Instrumentation 231
9.5.6 Physical Installation 232
9.6 Agreement Protocols 232
9.6.1 Raw Data, Measured Data, and Agreed Data 233
9.6.2 Syntactic Agreement 233
9.6.3 Semantic Agreement 233
9.7 Error Detection 234
9.7.1 Monitoring Task Execution Times 234
9.7.2 Monitoring Interrupts 235
9.7.3 Double Execution of Tasks 235
9.7.4 Watchdogs 235
Points to Remember 235
Bibliographic Notes 236
Review Questions and Problems 237

10 Real-Time Scheduling 239

10.1 The Scheduling Problem 240
10.1.1 Classification of Scheduling Algorithms 240
10.1.2 Schedulability Test 241
10.1.3 The Adversary Argument 242
10.2 Worst-Case Execution Time 243
10.2.1 WCET of Simple Tasks 243
10.2.2 WCET of Complex Tasks 246
10.2.3 Anytime Algorithms 246
10.2.4 State of Practice 247
10.3 Static Scheduling 248
10.3.1 Static Scheduling Viewed as a Search 248
10.3.2 Increasing the Flexibility in Static Schedules 250
10.4 Dynamic Scheduling 251
10.4.1 Scheduling Independent Tasks 251
10.4.2 Scheduling Dependent Tasks 253

10.5 Alternative Scheduling Strategies 255
10.5.1 Scheduling in Distributed Systems 255
10.5.2 Feedback Scheduling 256
Points to Remember 256
Bibliographic Notes 257
Review Questions and Problems 257

11 System Design 259

11.1 System Design 260
11.1.1 The Design Process 260
11.1.2 The Role of Constraints 261
11.1.3 System Design Versus Software Design 262
11.2 Design Phases 263
11.2.1 Purpose Analysis 264
11.2.2 Requirements Capture 264
11.2.3 Architecture Design 265
11.2.4 Design of Components 265
11.3 Design Styles 266
11.3.1 Model-Based Design 266
11.3.2 Component-Based Design 267
11.3.3 Architecture Design Languages 268
11.3.4 Test of a Decomposition 269
11.4 Design of Safety-Critical Systems 271
11.4.1 What Is Safety? 272
11.4.2 Safety Analysis 274
11.4.3 Safety Case 276
11.4.4 Safety Standards 279
11.5 Design Diversity 281
11.5.1 Diverse Software Versions 281
11.5.2 An Example of a Fail-Safe System 282
11.5.3 Multilevel System 283
11.6 Design for Maintainability 284
11.6.1 Cost of Maintenance 284
11.6.2 Maintenance Strategy 285
11.6.3 Software Maintenance 287
Points to Remember 287
Bibliographic Notes 289
Review Questions and Problems 289

12 Validation 291

12.1 Validation Versus Verification 292
12.2 Testing Challenges 293
12.2.1 Design for Testability 293
12.2.2 Test Data Selection 294

12.2.3 Test Oracle 296
12.2.4 System Evolution 296
12.3 Testing of Component-Based Systems 297
12.3.1 Component Provider 297
12.3.2 Component User 298
12.3.3 Communicating Components 298
12.4 Formal Methods 299
12.4.1 Formal Methods in the Real World 299
12.4.2 Classification of Formal Methods 300
12.4.3 Benefits of Formal Methods 301
12.4.4 Model Checking 302
12.5 Fault Injection 302
12.5.1 Software-Implemented Fault Injection 303
12.5.2 Physical Fault Injection 303
12.5.3 Sensor and Actuator Failures 305
Points to Remember 305
Bibliographic Notes 306
Review Questions and Problems 306

13 Internet of Things 307

13.1 The Vision of an Internet-of-Things 308
13.2 Drivers for an IoT 309
13.2.1 Uniformity of Access 309
13.2.2 Logistics 309
13.2.3 Energy Savings 310
13.2.4 Physical Security and Safety 310
13.2.5 Industrial 311
13.2.6 Medical 311
13.2.7 Life Style 311
13.3 Technical Issues of the IoT 311
13.3.1 Internet Integration 311
13.3.2 Naming and Identification 312
13.3.3 Near Field Communication 313
13.3.4 IoT Device Capabilities versus Cloud Computing 314
13.3.5 Autonomic Components 314
13.4 RFID Technology 315
13.4.1 Overview 316
13.4.2 The Electronic Product Code 316
13.4.3 RFID Tags 317
13.4.4 RFID Readers 318
13.4.5 RFID Security 318
13.5 Wireless Sensor Networks 320
Points to Remember 321
Bibliographic Notes 322
Review Questions and Problems 322

14 The Time-Triggered Architecture 325
14.1 History of the TTA 326
14.1.1 The MARS Project 326
14.1.2 The Industrial TTA Prototype 327
14.1.3 The GENESYS Project 327
14.2 Architectural Style 328
14.2.1 Complexity Management 328
14.2.2 Component Orientation 329
14.2.3 Coherent Communication 330
14.2.4 Dependability 331
14.2.5 Time Aware Architecture 332
14.3 Services of the TTA 332
14.3.1 Component-Based Services 332
14.3.2 Core System Services 333
14.3.3 Optional System Services 334
14.4 The Time-Triggered MPSoC 336
Points to Remember 337
Bibliographic Notes 338
Review Questions and Problems 338

Abbreviations 341

Glossary 343

References 359

Index 369