

Contents

Preface	13
1 Introduction to Real-time Systems	15
1.1 Historical Background	15
1.2 Elements of a Computer Control System	18
1.3 Real-time Systems – Definition	24
1.4 Classification of Real-time Systems	26
1.4.1 Clock-based Tasks (cyclic, periodic)	27
1.4.2 Event-based Tasks (aperiodic)	27
1.4.3 Interactive Systems	28
1.5 Time Constraints	28
1.6 Classification of Programs	31
1.6.1 Sequential	32
1.6.2 Multi-tasking	32
1.6.3 Real-time	33
1.7 Summary	34
Exercises	35
2 Concepts of Computer Control	36
2.1 Introduction	36
2.1.1 Batch	37
2.1.2 Continuous	37
2.1.3 Laboratory Systems	38
2.1.4 General Embedded Systems	38
2.2 Sequence Control	39
2.3 Loop Control (Direct Digital Control)	44
2.3.1 PID Control	45
2.3.2 DDC Applications	46
2.3.3 Adaptive Control	50
2.4 Supervisory Control	53
2.5 Centralised Computer Control	55

2.6	Hierarchical Systems	57
2.7	Distributed Systems	61
2.8	Human-Computer Interface (HCI)	63
2.9	The Control Engineer	64
2.10	Economics and Benefits of Computer Control Systems	65
2.11	Summary	66
	Exercises	67
3	Computer Hardware Requirements for Real-time Applications	68
3.1	Introduction	68
3.2	General Purpose Computer	68
	3.2.1 Central Processing Unit	70
	3.2.2 Storage	71
	3.2.3 Input and Output	72
	3.2.4 Bus Structure	72
3.3	Single-chip Microcomputers and Microcontrollers	73
3.4	Specialised Processors	74
	3.4.1 Parallel Computers	74
	3.4.2 Digital Signal Processors	76
3.5	Process-related Interfaces	76
	3.5.1 Digital Signal Interfaces	77
	3.5.2 Pulse Interfaces	81
	3.5.3 Analog Interfaces	83
	3.5.4 Real-time Clock	85
3.6	Data Transfer Techniques	86
	3.6.1 Polling	87
	3.6.2 Interrupts	88
	3.6.3 Direct Memory Access	101
	3.6.4 Comparison of Data Transfer Techniques	101
3.7	Communications	102
	3.7.1 Asynchronous and Synchronous Transmission Techniques	103
	3.7.2 Local- and Wide-area Networks	106
3.8	Standard Interfaces	109
3.9	Summary	111
	Exercises	112
4	DDC Algorithms and Their Implementation	115
4.1	Introduction	115
4.2	Implementation of the Basic PID Algorithm	117
4.3	Synchronisation of the Control Loop	118
	4.3.1 Polling	119
	4.3.2 Ballast Code	120

4.3.3	External Interrupt	120
4.3.4	Real-time Clock	121
4.4	Bumpless Transfer	126
4.4.1	Method 1 – Preset Change-over Value	127
4.4.2	Method 2 – Tracking of Operator Setting	127
4.4.3	Method 3 – Velocity Algorithm	127
4.4.4	Comparison of Position and Velocity Algorithms	128
4.5	Saturation and Integral Action Wind-up	129
4.5.1	Fixed Limits	131
4.5.2	Stop Summation	132
4.5.3	Integral Subtraction	133
4.5.4	Velocity Algorithm	133
4.5.5	Analytical Approach	135
4.6	Tuning	137
4.7	Choice of Sampling Interval	139
4.8	Plant Input and Output	140
4.8.1	Noise	140
4.8.2	Actuator Control	142
4.8.3	Computational Delay	143
4.9	Improved Forms of Algorithm for Integral and Derivative Calculation	144
4.10	Implementation of Controller Designs Based on Plant Models	145
4.10.1	The PID Controller in Z-transform Form	146
4.10.2	Direct Method 1	148
4.10.3	Direct Method 2	149
4.10.4	Cascade Realisation	149
4.10.5	Parallel Realisation	150
4.10.6	Discretisation of Continuous Controllers	151
4.11	Summary	151
	Exercises	152
5	Languages for Real-time Applications	155
5.1	Introduction	155
5.1.1	Security	157
5.1.2	Readability	158
5.1.3	Flexibility	158
5.1.4	Simplicity	159
5.1.5	Portability	159
5.1.6	Efficiency	159
5.2	Syntax Layout and Readability	160
5.3	Declaration and Initialisation of Variables and Constants	163
5.3.1	Declarations	163
5.3.2	Initialisation	164

5.3.3	Constants	165
5.4	Modularity and Variables	166
5.4.1	Scope and Visibility	166
5.4.2	Global and Local Variables	167
5.4.3	Control of Visibility and Scope	169
5.4.4	Modularity	169
5.5	Compilation of Modular Programs	171
5.6	Data Types	172
5.6.1	Sub-range Types	173
5.6.2	Derived Types	174
5.6.3	Structured Types	175
5.6.4	Pointers	176
5.7	Control Structures	178
5.8	Exception Handling	181
5.9	Low-level Facilities	186
5.10	Coroutines	190
5.11	Interrupts and Device Handling	191
5.12	Concurrency	193
5.13	Run-time Support	194
5.14	Overview of Real-time Languages	195
5.15	Application-oriented Software	196
5.15.1	Table-driven Approach	198
5.15.2	Block-structured Software	200
5.15.3	Application Languages	202
5.16	CUTLASS	203
5.16.1	General Features of CUTLASS	203
5.16.2	Data Typing and Bad Data	206
5.16.3	Language Subsets	207
5.16.4	Scope and Visibility	207
5.16.5	Summary	209
5.17	A Note on BASIC	209
5.18	Summary	211
	Exercises	211
6	Operating Systems	212
6.1	Introduction	212
6.2	Real-time Multi-tasking Operating Systems	215
6.3	Scheduling Strategies	219
6.3.1	Cyclic	219
6.3.2	Pre-emptive	219
6.4	Priority Structures	221
6.4.1	Interrupt Level	221
6.4.2	Clock Level	223

6.4.3	Cyclic Tasks	223
6.4.4	Delay Tasks	225
6.4.5	Base Level	226
6.5	Task Management	227
6.5.1	Task States	227
6.5.2	Task Descriptor	228
6.6	Scheduler and Real-time Clock Interrupt Handler	230
6.6.1	System Commands Which Change Task Status	230
6.6.2	Dispatcher – Search for Work	231
6.7	Memory Management	236
6.8	Code Sharing	241
6.8.1	Serially Reusable Code	242
6.8.2	Re-entrant Code	242
6.9	Resource Control: an Example of an Input/Output Subsystem (IOSS)	244
6.9.1	Example of an IOSS	248
6.9.2	Output to Printing Devices	250
6.9.3	Device Queues and Priorities	252
6.10	Task Co-operation and Communication	254
6.11	Mutual Exclusion	254
6.11.1	Semaphore	256
6.11.2	The Monitor	260
6.11.3	Intertask Communication	261
6.11.4	Task Synchronisation Without Data Transfer	261
6.12	Data Transfer (the Producer–Consumer Problem)	262
6.12.1	Data Transfer Without Synchronisation	262
6.12.2	Synchronisation With Data Transfer	269
6.13	Liveness	269
6.14	Minimum Operating System Kernel	270
6.15	Example of Creating an RTOS Based on a Modula-2 Kernel	270
6.16	Summary	276
	Exercises	277
7	Design of Real-time Systems – General Introduction	278
7.1	Introduction	278
7.2	Specification Document	284
7.3	Preliminary Design	286
7.3.1	Hardware Design	286
7.3.2	Software Design	287
7.4	Single-program Approach	288
7.5	Foreground/Background System	290
7.6	Multi-tasking Approach	296
7.7	Mutual Exclusion	297

7.7.1	Condition Flags	299
7.7.2	Semaphores	303
7.7.3	Notes on Using Semaphores	309
7.8	Monitors	310
7.9	Rendezvous	313
7.10	Summary	318
	Exercises	319
8	Real-time System Development Methodologies – 1	320
8.1	Introduction	320
8.2	Yourdon Methodology	324
8.3	Requirements Definition for Drying Oven	324
8.4	Ward and Mellor Method	327
8.4.1	Building the Essential Model – the Environmental Model	328
8.4.2	Building the Essential Model – the Behavioural Model	331
8.4.3	Behavioural Model – Rules and Conventions	334
8.4.4	Process Specifications	337
8.4.5	Control Specifications	337
8.4.6	Checking the Essential Model	341
8.4.7	Building the Implementation Model	341
8.4.8	Enhancing the Model	341
8.4.9	Allocation of Resources	343
8.5	Hatley and Pirbhai Method	345
8.5.1	Requirements Model	345
8.5.2	Architecture Model	349
8.6	Comments on the Yourdon Methodologies	350
8.7	Summary	352
9	Real-time System Development Methodologies – 2	353
9.1	MASCOT	353
9.2	Basic Features of MASCOT	354
9.2.1	Simple Example	355
9.2.2	Communication Methods	356
9.3	General Design Approach	356
9.4	Textual Representations of MASCOT Designs	359
9.5	Other Features of MASCOT	362
9.5.1	Constants	362
9.5.2	Direct Data Visibility	362
9.5.3	Qualifiers	362
9.6	Development Facilities	363
9.7	The MASCOT Kernel	364
9.8	Summary of MASCOT	365

<i>Contents</i>	11
9.9 Formal Methods	366
9.10 The PAISLey System for Real-time Software Development Method	366
9.10.1 A Simple System	367
9.10.2 Exchange Functions	371
9.10.3 Timing Constraints	371
9.11 PAISLey Summary	372
9.12 Summary	375
10 Design Analysis	376
10.1 Introduction	376
10.2 Petri Nets	376
10.2.1 Basic Ideas	377
10.2.2 Modelling Mutual Exclusion	380
10.3 Analysing Petri Nets	384
10.3.1 Reachability Tree	386
10.4 Scheduling	387
10.5 General Approaches to the Scheduling Problem	389
10.6 On-line Scheduling – Independent Tasks	389
10.6.1 Schedulability	390
10.6.2 Scheduling Algorithms – Pre-emptive, Priority Based	393
10.6.3 Scheduling Algorithms – Other Types	394
10.7 Pre-run-time Scheduling	395
10.8 Scheduling – Including Task Synchronisation	396
10.9 Summary	397
Exercises	398
11 Dependability, Fault Detection and Fault Tolerance	399
11.1 Introduction	399
11.2 Use of Redundancy	399
11.3 Fault Tolerance in Mixed Hardware–Software Systems	402
11.3.1 Mechanisms and Measures	403
11.3.2 Exceptions	404
11.4 Fault Detection Measures	404
11.4.1 Replication Checks	404
11.4.2 Expected Value Methods	405
11.4.3 Watch-dog Timers	406
11.4.4 Reversal Checks	407
11.4.5 Parity and Error Coding Checks	407
11.4.6 Structural Checks	407
11.4.7 Diagnostic Checks	408
11.5 Fault Detection Mechanisms	408
11.6 Damage Containment and Assessment	409

11.7	Provision of Fault Tolerance	409
11.7.1	Redundancy	409
11.7.2	Deadline Mechanisms	410
11.7.3	Bad Data Marking	411
11.7.4	Recovery Measures – Check Points	412
11.8	Summary	413
	Bibliography	414
	Index	425

Preface

Over the past 30 years digital control of industrial processes has changed from being the exception to the commonplace. Each succeeding year sees an increase in the range of applications and each advance in hardware design widens the potential application areas. Microprocessors are now a normal component employed in a wide range of electronic systems.

Computers in one form or another now form an integral part of most real-time control systems; such computers are generally referred to as *embedded real-time computers* and an understanding of how to design and build systems containing embedded computers is an essential requirement for a systems engineer. The knowledge required covers both hardware and software design and construction, and of the two the software engineering is the most difficult and least understood. The difficulties of specifying, designing and building real-time software have led in recent years to intensive work on development methodologies and one of the major objectives of this book is to introduce the reader to the fundamental ideas underlying these methodologies.

Traditional computing courses for engineers have typically emphasised the hardware and programming language aspects of using computers. An understanding of the basic principles of operation of computer hardware and of programming languages is of course important; however, these are details with respect to the overall system design problem. Also in the past it has frequently been assumed that real-time control software will be written in an assembly language. It is of course true that for many engineering applications use of assembly languages, and languages such as FORTH and real-time BASIC, are legitimate, economical means of producing a viable solution. However, for large-scale, complex and safety-critical systems such approaches are no longer adequate and in this book I attempt to cover some of the important and fundamental ideas underlying a software engineering approach to specifying, designing and building real-time software for computer control applications.

The book is intended for final-year undergraduate students and practising engineers, and is also suitable as a text for computer science students requiring an introduction to real-time software from an application viewpoint as opposed to an operating system viewpoint. It assumes that the reader has some familiarity with at

least one high-level programming language, with basic ideas about computer hardware and an understanding of simple feedback control concepts. It does not require familiarity with control design techniques. Examples in the text are given using Modula-2 and should be easily followed by anyone familiar with Pascal.

In Chapters 1 and 2 I introduce basic concepts relating to real-time systems and their characteristics and provide an overview of computer control applications. Chapter 3 provides a brief summary of the important hardware building blocks for computers used for control. Chapter 4 introduces some of the practical problems of implementing control algorithms. I use the simple and widely used PID (Proportional + Integral + Derivative) controller as an example and I explain its operation in detail. The section of this chapter which uses z-transform notation and deals with the implementation of controllers designed using discrete control techniques can be omitted by readers who have no background in control systems. In the final part of Chapter 4, I introduce in general terms how to deal with systems which involve more than just the control algorithm.

Chapter 5 on real-time languages and Chapter 6 on operating systems and concurrency provide basic information about two essential tools that will be required by the real-time system builder. In Chapters 7, 8 and 9 I deal with the methodologies that have been developed in recent years to help in the specification, design and construction of real-time software and real-time systems. I concentrate mainly on two methodologies, MASCOT and the Hatley & Pirbhai variant of the Yourdon method. The standard methodologies provide very little guidance for the actual implementation of the software design and in Chapter 10 I describe some of the implementation problems and possible approaches to solving them. The final chapter deals with the vital topic of building dependable software.

Many people assisted me in producing the first edition of this book: colleagues in the Department of Automatic Control and Systems Engineering at the University of Sheffield, in particular Steve White. In preparing this second edition I have received much useful comment from Les Woolliscroft and from two reviewers.

Stuart Bennett
Sheffield, UK
April 1993