

Contents

Preface xi

1 Programming embedded systems in C 1

- 1.1 Introduction 1
- 1.2 What is an embedded system? 1
- 1.3 Which processor should you use? 2
- 1.4 Which programming language should you use? 7
- 1.5 Which operating system should you use? 9
- 1.6 How do you develop embedded software? 12
- 1.7 Conclusions 15

2 Introducing the 8051 microcontroller family 17

- 2.1 Introduction 17
- 2.2 What's in a name? 17
- 2.3 The external interface of the Standard 8051 18
- 2.4 Reset requirements 20
- 2.5 Clock frequency and performance 21
- 2.6 Memory issues 23
- 2.7 I/O pins 29
- 2.8 Timers 29
- 2.9 Interrupts 30
- 2.10 Serial interface 32
- 2.11 Power consumption 32
- 2.12 Conclusions 34

3 Hello, embedded world 35

- 3.1 Introduction 35
- 3.2 Installing the Keil software and loading the project 36

3.3 Configuring the simulator	37
3.4 Building the target	39
3.5 Running the simulation	39
3.6 Dissecting the program	43
3.7 Aside: Building the hardware	55
3.8 Conclusions	56

4 Reading switches 57

4.1 Introduction	57
4.2 Basic techniques for reading from port pins	58
4.3 Example: Reading and writing bytes	60
4.4 Example: Reading and writing bits (simple version)	61
4.5 Example: Reading and writing bits (generic version)	62
4.6 The need for pull-up resistors	67
4.7 Dealing with switch bounce	69
4.8 Example: Reading switch inputs (basic code)	70
4.9 Example: Counting goats	75
4.10 Conclusions	80

5 Adding structure to your code 81

5.1 Introduction	81
5.2 Object-oriented programming with C	82
5.3 The Project Header (MAIN.H)	88
5.4 The Port Header (PORT.H)	94
5.5 Example: Restructuring the 'Hello Embedded World' example	96
5.6 Example: Restructuring the goat-counting example	103
5.7 Further examples	111
5.8 Conclusions	111

6 Meeting real-time constraints 113

6.1 Introduction	113
6.2 Creating 'hardware delays' using Timer 0 and Timer 1	116
6.3 Example: Generating a precise 50 ms delay	120

- 6.4 Example: Creating a portable hardware delay 124
- 6.5 Why not use Timer 2? 129
- 6.6 The need for ‘timeout’ mechanisms 129
- 6.7 Creating loop timeouts 130
- 6.8 Example: Testing loop timeouts 133
- 6.9 Example: A more reliable switch interface 134
- 6.10 Creating hardware timeouts 136
- 6.11 Example: Testing a hardware timeout 140
- 6.12 Conclusions 142

7 Creating an embedded operating system 143

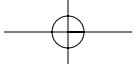
- 7.1 Introduction 143
- 7.2 The basis of a simple embedded OS 147
- 7.3 Introducing sEOS 152
- 7.4 Using Timer 0 or Timer 1 161
- 7.5 Is this approach portable? 166
- 7.6 Alternative system architectures 166
- 7.7 Important design considerations when using sEOS 172
- 7.8 Example: Milk pasteurization 174
- 7.9 Conclusions 187

8 Multi-state systems and function sequences 189

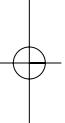
- 8.1 Introduction 189
- 8.2 Implementing a Multi-State (Timed) system 192
- 8.3 Example: Traffic light sequencing 192
- 8.4 Example: Animatronic dinosaur 198
- 8.5 Implementing a Multi-State (Input/Timed) system 204
- 8.6 Example: Controller for a washing machine 205
- 8.7 Conclusions 215

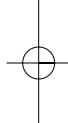
9 Using the serial interface 217

- 9.1 Introduction 217
- 9.2 What is RS-232? 217
- 9.3 Does RS-232 still matter? 218


x Contents

- 9.4 The basic RS-232 protocol 218
- 9.5 Asynchronous data transmission and baud rates 219
- 9.6 Flow control 220
- 9.7 The software architecture 220
- 9.8 Using the on-chip UART for RS-232 communications 222
- 9.9 Memory requirements 224
- 9.10 Example: Displaying elapsed time on a PC 225
- 9.11 The Serial-Menu architecture 237
- 9.12 Example: Data acquisition 237
- 9.13 Example: Remote-control robot 252
- 9.14 Conclusions 253


10 Case study: Intruder alarm system 255

- 10.1 Introduction 255
 - 10.2 The software architecture 257
 - 10.3 Key software components used in this example 257
 - 10.4 Running the program 258
 - 10.5 The software 258
 - 10.6 Conclusions 283
- 

11 Where do we go from here 285

- 11.1 Introduction 285
- 11.2 Have we achieved our aims? 285
- 11.3 Suggestions for further study 286
- 11.4 *Patterns for Time-Triggered Embedded Systems* 288
- 11.5 *Embedded Operating Systems* 288
- 11.6 Conclusions 289

Index 291

Licensing Agreement 295

