

EMBEDDED TECHNOLOGY™

SERIES

Embedded Microprocessor Systems: Real World Design

THIRD EDITION

Stuart R. Ball, P.E.

Embedded Microprocessor Systems
Real World Design

Embedded Microprocessor Systems Real World Design

Third Edition

Stuart R. Ball




An imprint of Elsevier Science

Amsterdam Boston London New York Oxford Paris San Diego
San Francisco Singapore Sydney Tokyo

Newnes is an imprint of Elsevier Science.

Copyright © 2002, Elsevier Science (USA). All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

 Recognizing the importance of preserving what has been written, Elsevier Science prints its books on acid-free paper whenever possible.

Library of Congress Cataloging-in-Publication Data

Ball, Stuart R., 1956–

Embedded microprocessor systems : real world design / Stuart R. Ball.—3rd ed.
p. cm.

ISBN 0-7506-7534-9 (pbk. : alk. paper)

1. Embedded computer systems—Design and construction. 2. Microprocessors.

I. Title.

TK7895.E42 B35 2002

621.39'16—dc21

2002071917

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

The publisher offers special discounts on bulk orders of this book.

For information, please contact:

Manager of Special Sales

Elsevier Science

200 Wheeler Road

Burlington, MA 01803

Tel: 781-313-4700

Fax: 781-313-4880

For information on all Newnes publications available, contact our World Wide Web home page at: <http://www.newnespress.com>

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Contents

Introduction xi
Special Introduction to the Third Edition xiv

1 System Design 1

- Requirements Definition 3
- Processor Selection 5
- Development Environment 17
- Development Costs 19
- Hardware and Software Requirements 20
- Hardware/Software Partitioning 22
- Distributed Processor Systems 24
- Specifications Summary 25
- A Requirements Document Outline 26
- Communication 28

2 Hardware Design 1 29

- Single-Chip Designs 29
- Multichip Designs 31
- Wait States 35
- Memory 38
- Types of PROM 39
- RAM 45
- I/O 54
- Peripheral ICs 58
- Data Bus Loading 68
- Nonvolatile Memory 70
- Microwire 73
- DMA 74

<i>Watchdog Timers</i>	81
<i>In-Circuit Programming</i>	83
<i>Internal Peripherals</i>	85
<i>Design Shortcuts</i>	85
<i>EMC Considerations</i>	86
<i>Microprocessor Clocks</i>	90
<i>Hardware Checklist</i>	92

3 Hardware Design 2 **95**

<i>Dynamic Bus Sizing</i>	95
<i>Fast Cycle Termination</i>	95
<i>Bus Sizing at Reset</i>	96
<i>Clock-Synchronized Buses</i>	97
<i>Built-in Dynamic Ram Interface</i>	99
<i>Combination ICs</i>	100
<i>Digital-to-Analog Converters</i>	101
<i>Analog-to-Digital Converters</i>	103
<i>SPI/Microwire in Multichip Designs</i>	106
<i>Timer Basics</i>	107
<i>Example System</i>	115
<i>Hardware Specifications Outline</i>	115

4 Software Design **119**

<i>Data Flow Diagram</i>	120
<i>State Diagram</i>	121
<i>Flowcharts</i>	123
<i>Pseudocode</i>	123
<i>Partitioning the Code</i>	125
<i>Software Architecture</i>	129
<i>The Development Language</i>	131
<i>Microprocessor Hardware</i>	135
<i>Hard Deadlines Versus Soft Deadlines</i>	138
<i>Dangerous Independence</i>	138
<i>Software Specifications</i>	140
<i>Software Specifications Outline</i>	141

5 Interrupts in Embedded Systems **143**

<i>Interrupt Basics</i>	143
<i>Interrupt Vectors</i>	144
<i>Edge- and Level-Sensitive Interrupts</i>	146

<i>Interrupt Priority</i>	146
<i>Interrupt Hardware</i>	146
<i>Interrupt Bus Cycles</i>	148
<i>Daisy-Chained Interrupts</i>	148
<i>Other Types of Interrupts</i>	149
<i>Using Interrupt Hardware</i>	150
<i>Interrupt Software</i>	155
<i>Interrupt Service Mechanics</i>	155
<i>Nested Interrupts</i>	157
<i>Passing Data to or from the ISR</i>	158
<i>Some Real World Dos and Don'ts</i>	159
<i>Minimizing Low-Priority Interrupt Service Time</i>	166
<i>When to Use Interrupts</i>	168

6 Adding Debug Hardware and Software **171**

<i>Action Codes</i>	172
<i>Hardware Output</i>	173
<i>Write to ROM</i>	175
<i>Read from ROM</i>	176
<i>Software Timing</i>	177
<i>Software Throughput</i>	177
<i>Circular Trace Buffers</i>	178
<i>Monitor Programs</i>	179
<i>Logic Analyzer Breakpoints</i>	180
<i>Memory Dumps</i>	181
<i>Serial Condition Monitor</i>	182

7 System Integration and Debug **189**

<i>Hardware Testing</i>	190
<i>Software Debug</i>	191
<i>Debugging in RAM</i>	193
<i>Functional Test Plan</i>	194
<i>Stress Testing</i>	196
<i>Problem Log</i>	197
<i>A Real-World Example</i>	198
<i>Emulators/Debuggers</i>	201

8 Multiprocessor Systems **203**

<i>Communication Between Processors</i>	205
<i>Dual-Port RAM (DPRAM)</i>	212

9	Real-Time Operating Systems	235
	Multitasking	238
	Keeping Track of Tasks	242
	Communication Between Tasks	243
	Memory Management	244
	Resource Management	245
	RTOS and Interrupts	247
	Typical RTOS Communication	247
	Preemption Considerations	248
	Applicability of RTOS	250
	Debuggers	253

10	Industry-Standard Embedded Platforms	255
	Advantages of Using a PC Platform	255
	Drawbacks of Using a PC Platform	258
	Some Solutions to These Problems	260
	ISA- and PCI-Based Embedded Boards	261
	Other Platforms for Embedded Systems	262
	Example Real-Time PC Application	267

11	Advanced Microprocessor Concepts	271
	Pipeline (Prefetch) Queue	271
	Interleaving	272
	DRAM Burst Mode	273
	SDRAM	274
	High-Speed, High-Integration Processors and Multiple Buses	277
	Cache Memory	278
	Processors with Multiple Clock Inputs and Phase-Locked Loops	279
	Multiple-Instruction Fetch and Decode	280
	Microcontroller/FPGA Combinations	281
	On-Chip Debug	282
	Memory Management Hardware	284
	Application-Specific Microcontrollers	286

	Appendix A: Example System Specifications	287
	System Description	287
	User Interface	287
	Setting Time	288

<i>Water Low</i>	288
<i>Example System Hardware Specifications</i>	288
<i>Example System Software Description</i>	290
<i>Example System Software Pseudocode</i>	292

Appendix B: Number Systems **303**

<i>Number Bases</i>	303
<i>Converting Numbers Between Bases</i>	306
<i>Math with Binary and Hex Numbers</i>	307
<i>Negative Numbers and Computer Representation of Numbers</i>	308
<i>Number Suffixes</i>	310
<i>Floating Point</i>	311

Appendix C: Digital Logic Review **315**

<i>Basic Logic Functions</i>	316
<i>Registers and Latches</i>	320

Appendix D: Basic Microprocessor Concepts **325**

<i>A Simple Microprocessor</i>	325
<i>A More Complex Microprocessor</i>	333
<i>Addressing Modes</i>	337
<i>Code Formats</i>	340

Appendix E: Embedded Web Sites **343**

<i>Organizations and Literature</i>	343
<i>Manufacturers</i>	343
<i>Software, Operating Systems, and Emulators</i>	344

<i>Glossary</i>	345
-----------------	-----

<i>Index</i>	350
--------------	-----

Introduction

Imagine this scene: You get into your car and turn the key on. You take a 3.5" floppy disk from the glove compartment, insert it into a slot in the dashboard, and drum your fingers on the steering wheel until the operating system prompt appears on the dashboard liquid crystal display (LCD). Using the cursor keys on the center console, you select the program for the electronic ignition, then turn the key and start the engine. On the way to work you want to listen to some music, so you insert the program compact disc (CD) into the player, wait for the green light to flash indicating that the digital signal processor (DSP) in the player is ready, then put in your music CD.

You get to work and go to the cafeteria for a pastry. Someone has borrowed the mouse from the microwave but has not unplugged the microwave itself, so the operating system is still up. You can heat your breakfast before starting work.

What is the point of this inconvenient scenario? This is how the world would work if we used microprocessor technology without having *embedded microprocessors*. Every microprocessor-based appliance would need a disk drive, some kind of input device, and some kind of display.

Embedded microprocessors are all around us. Since the original Intel 8080 was pioneered in the 1970s, engineers have been embedding microprocessors in their designs. They even are embedded in general-purpose computers; if you own a variation of the IBM PC/AT, there is an embedded microprocessor in the keyboard. Virtually all printers have at least one microprocessor in them, and no car on the market is without at least one under the hood. Embedded microprocessors may control the automatic processing equipment that cans your soup or the controls of your microwave oven. Basically, we can define an embedded microprocessor as having the following characteristics:

- Dedicated to controlling a specific real-time device or function.
- Self-starting, not requiring human intervention to begin. The user cannot tell if the system is controlled by a microprocessor or by dedicated hardware.
- Self-contained, with the operating program in some kind of nonvolatile memory.

Of course, there are exceptions to this general description, which we will get to eventually, but this definition will serve us for now.

An embedded microprocessor system usually contains the following components:

- A microprocessor
- RAM (random access memory)
- Nonvolatile storage: erasable programmable read-only memory (EPROM), read-only memory (ROM), flash memory, battery-backed RAM, and so on
- I/O (some means to monitor or control the real world)

If you have seen textbooks describing general computer systems, this description fits those as well. The difference is in the details. A general-purpose computer, such as the one this book was written on, may have many megabytes of RAM, whereas an embedded system may have less than 256 bytes (that is bytes, not megabytes) of RAM. Your PC at home or at the office may have a 10GB IDE hard drive with DOS, Windows, and several other applications.

An embedded system usually contains its entire program in a few thousand bytes of EPROM. The most important difference between the two is the application. Your home personal computer (PC) runs a word processor, then you switch over to the money management program to balance your checkbook, then to the spreadsheet to work on the family budget, then back to the word processor. The embedded system does just a limited number of tasks, such as making sure your toast does not burn or timing the cook cycle in your microwave.

Why would anyone want to use a microprocessor? The main reasons are:

- **Cost.** The cost of developing firmware for an embedded system can be very high, but it is a *nonrecurring expense*, only spent once to develop the product. The actual cost of the finished product can be very low. On the other hand, the product cost of a system such as a microwave oven controller, if implemented in discrete hardware, can be very high by comparison.
- **Flexibility.** Say a typical microwave oven manufacturer gets a contract from a very large discount store for microwave ovens, but the contract specifies certain changes in the way the user controls the device. In a hardware-based system, the control electronics would need to be redesigned. In a microprocessor-based system, the only change may be a few lines of code.
- **Programmability.** You may want to program a robotic arm to paint car doors one day and trunk lids the next. An embedded microcontroller permits you to have the same hardware perform different tasks. Of course, this also could be implemented in discrete hardware but at much higher cost.
- **Adaptability.** A system may need to adapt to its environment or to a user's needs. A typical example of this is an automobile's "smart" automatic transmission, which remembers your driving patterns and adjusts its shift points for optimum

comfort, economy, or even reliability. You *could* implement this sort of feature with dedicated hardware, but a microprocessor makes the job much easier.

This book will take you step by step through the procedures involved in designing an embedded control system. Many of the tricks I have learned in my 20 years in the field will be passed on, as well as some pitfalls to avoid. Along the way, we will use as an example of a simple embedded control system, a swimming pool pump timer, to illustrate these concepts.

The book is aimed primarily at students, new graduates who will be moving into the embedded processor field, and engineers working in another field who want to switch to embedded microprocessors. It assumes that the reader has a basic knowledge of software concepts, binary and hexadecimal number systems, and a basic understanding of digital logic. A review of this material is included in the appendixes at the end.