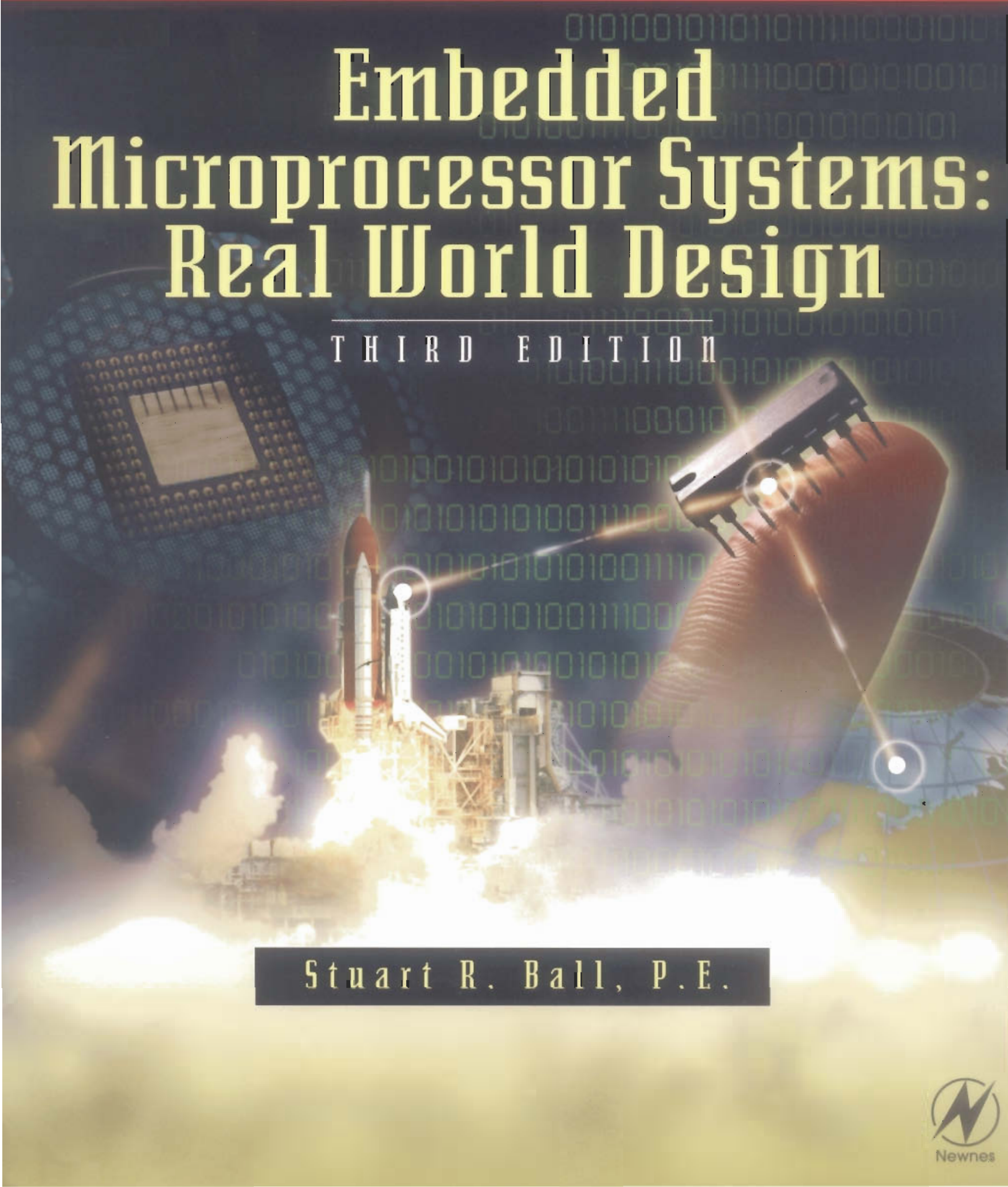
EMBEDDED TECHNOLOGY™
SERIES

Embedded Microprocessor Systems: Real World Design

THIRD EDITION



Stuart R. Ball, P.E.

Newnes

Embedded Microprocessor Systems

Real World Design

Embedded Microprocessor Systems
Real World Design

Third Edition

Stuart R. Ball

Newnes
An imprint of Butterworth-Heinemann

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.

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.

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Contents

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.