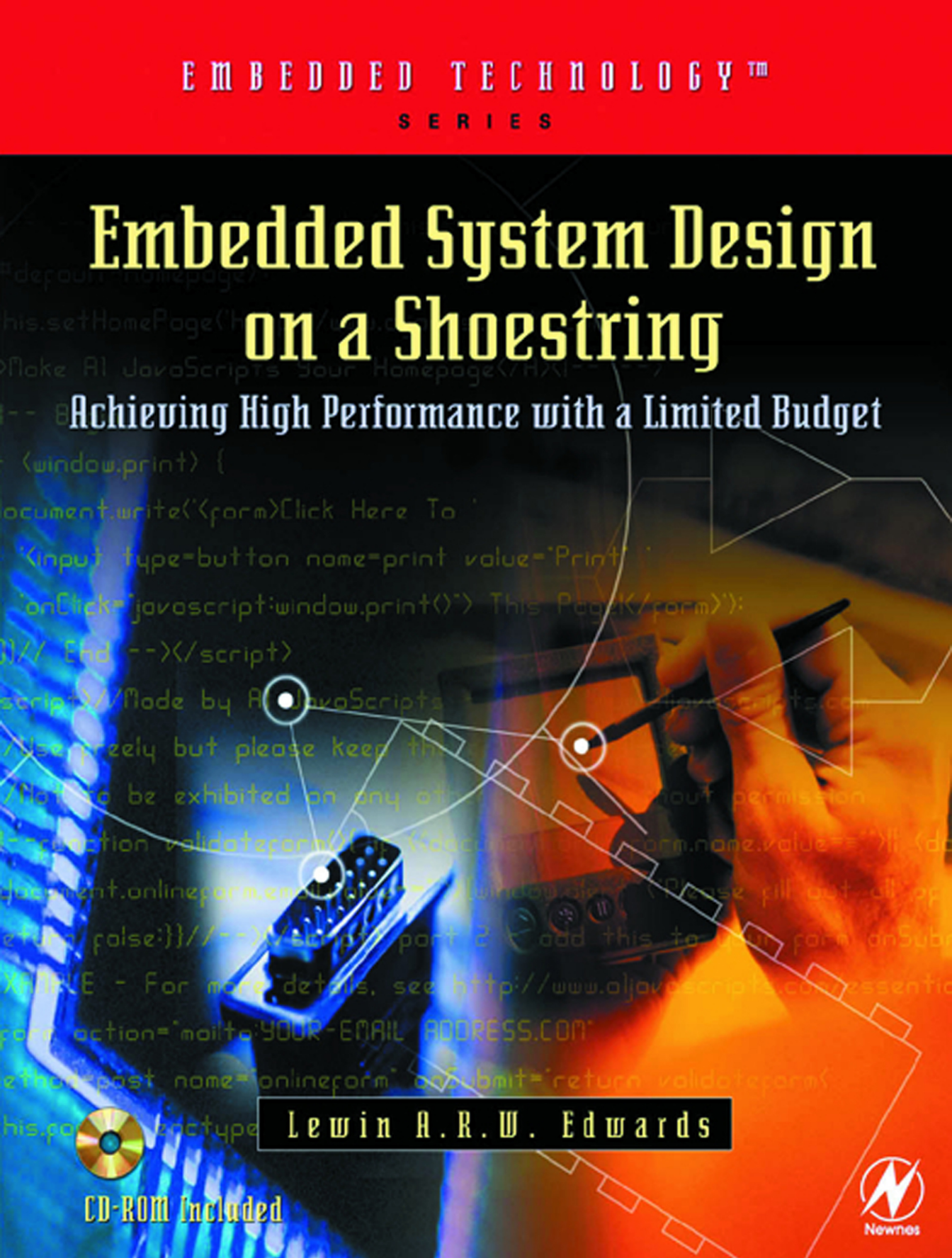
EMBEDDED TECHNOLOGY™

SERIES

# Embedded System Design on a Shoestring

## Achieving High Performance with a Limited Budget



Lewin A.R.W. Edwards

CD-ROM Included

Newnes

# *Embedded System Design on a Shoestring*

*Achieving High Performance with a Limited Budget*

This Page Intentionally Left Blank

# *Embedded System Design on a Shoestring*

*Achieving High Performance with a Limited Budget*

***by Lewin A.R.W. Edwards***


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

Newnes is an imprint of Elsevier Science.



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**

ISBN: 0-7506-7609-4

**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: 781-313-4882

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

This Page Intentionally Left Blank

# Acknowledgments

The author would like to extend his sincere thanks to the following individuals and corporations who have contributed directly and indirectly to the publication of this book:

- Atmel developer support
- Cadsoft Computer, Inc.
- Cirrus Logic developer support
- Michael Barr
- Don McKenzie of dontronics.com
- Spehro Pefhany
- Rob Severson of USBmicro
- Sharp Microelectronics developer support

In keeping with the open-source nature of this book's subject matter, the manuscript of this work was developed entirely using the free open-source OpenOffice.org office productivity suite, under Red Hat Linux 8.0.

This Page Intentionally Left Blank

# Introduction

There exists a large body of literature focused on teaching both general embedded systems principles and design techniques, and tips and tricks for specific microcontrollers. The majority of this literature is targeted at small 8-bit microcontrollers such as the Microchip PIC, Atmel AVR and the venerable 8051, principally because these devices are inexpensive and readily available in small quantity, and development hardware is available from a variety of sources at affordable prices. Historically, higher-performance 16- and 32-bit parts have been hard to obtain in small quantities, their development toolchains have been prohibitively expensive, and the devices themselves have been difficult to design around, with tight electrical and timing requirements on external circuitry necessitating very careful hardware design. A dearth of royalty-free, open-source operating system and library code for these processors also meant that developing a new project was a huge from-the-ground-up effort.

However, over the past few years we have simultaneously seen the size and price of 16- and 32-bit cores fall, and the development of many highly integrated parts that enable the easy development of almost single-chip 32-bit systems. In addition, many readily available appliances now contain a well-documented 32-bit microcontroller with ample flash memory, RAM and a variety of useful peripherals such as color LCDs, network interfaces and so forth, which can be exploited by the cunning embedded developer as a ready-made hardware platform. Cross-platform assemblers, high-level language compilers and debugging tools are available free for the downloading and will run satisfactorily on the average desktop PC; it is no longer nec-

essary to spend tens of thousands of dollars on proprietary compilers and special workstations on which to run them.

As these systems have increased in complexity, to a certain extent the degree of specialization required to develop them has decreased. This might sound paradoxical, but consider the fact that high-end 32-bit embedded systems, and the tools used to develop for them, are effectively converging with the low-end mainstream PC. The skills required to develop an application for embedded Linux, NetBSD or Windows CE are by intention not radically different from the skills used in developing applications for the desktop equivalents of these operating systems (though of course different coding best practices usually apply in embedded environments). In most cases there are mature off-the-shelf operating systems available ready-to-run for the common hardware reference designs and manufacturer-supplied evaluation boards, so we are usually spared even the initial bring-up phase and much of the effort required to debug device drivers.

Given a working hardware platform with reasonably well-documented components, the only task for which traditional embedded expertise is absolutely necessary is to create the necessary bootstrap and "glue" code to get a C run-time working on the target platform, and perhaps create drivers for some peripherals (and as discussed above, even this step can often be skipped if you are building around a reference platform). From that point on, most of the programming work to be done runs in the application layer and can be accomplished using high-level languages. There is a large workforce available almost ready-trained for this type of coding.

The end result of this evolutionary process is that it is now well within the financial and logistical reach of a small company or even an individual hobbyist or student to develop (or at least repurpose) advanced embedded systems with exciting functionality provided by these high-performance parts. Unfortunately, however, device vendors' support infrastructures are still geared towards large-scale commercial developers. This raises two major obstacles: