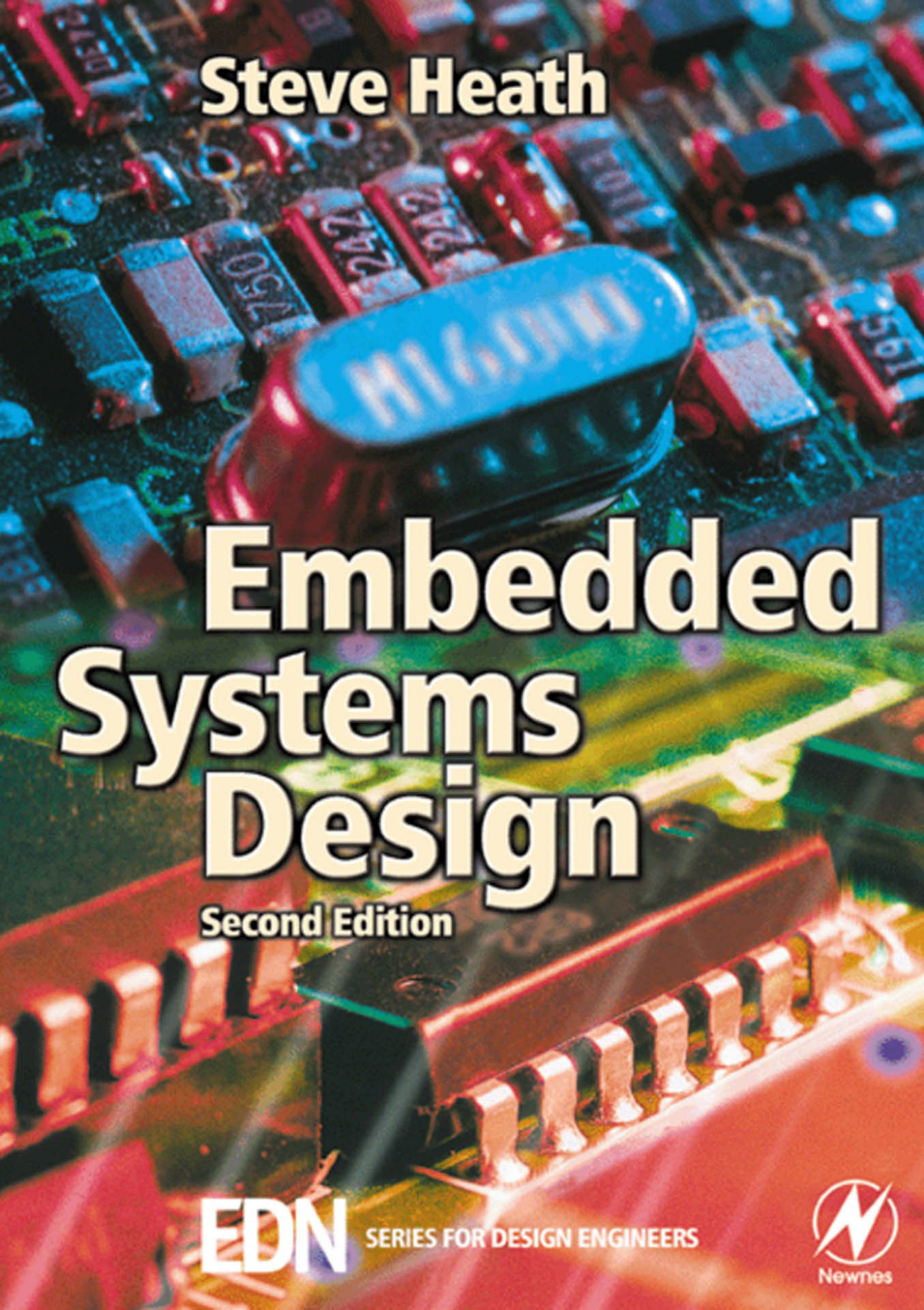
Steve Heath
Embedded Systems Design
Second Edition
EDN
SERIES FOR DESIGN ENGINEERS
Newnes

Embedded Systems Design

By the same author

VMEbus: a practical companion

Newnes UNIX™ Pocket Book

Microprocessor architectures: RISC, CISC and DSP

Effective PC networking

PowerPC: a practical companion

The PowerPC Programming Pocket Book

The PC and MAC handbook

The Newnes Windows NT Pocket Book

Multimedia Communications

Essential Linux

Migrating to Windows NT

All books published by Butterworth-Heinemann

About the author:

Through his work with Motorola Semiconductors, the author has been involved in the design and development of microprocessor-based systems since 1982. These designs have included VMEbus systems, microcontrollers, IBM PCs, Apple Macintoshes, and both CISC- and RISC-based multiprocessor systems, while using operating systems as varied as MS-DOS, UNIX, Macintosh OS and real-time kernels.

An avid user of computer systems, he has had over 60 articles and papers published in the electronics press, as well as several books.

Embedded Systems Design

Second edition

Steve Heath

OXFORD AMSTERDAM BOSTON LONDON NEW YORK
PARIS SAN DIEGO SAN FRANCISCO SINGAPORE SYDNEY TOKYO

Newnes
An imprint of Elsevier Science
Linacre House, Jordan Hill, Oxford OX2 8DP
200 Wheeler Road, Burlington MA 01803

First published 1997
Reprinted 2000, 2001
Second edition 2003

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Cataloguing in Publication Data
A catalogue record for this book is available from the Library of Congress

ISBN 0 7506 5546 1

Typeset by *Steve Heath*

Contents

This Page Intentionally Left Blank

Preface

The term embedded systems design covers a very wide range of microprocessor designs and does not simply start and end with a simple microcontroller. It can be a PC running software other than Windows and word processing software. It can be a sophisticated multiprocessor design using the fastest processors on the market today.

The common thread to embedded systems design is an understanding of the interaction that the various components within the system have with each other. It is important to understand how the hardware works and the restraints that using a certain peripheral may have on the rest of the system. It is essential to know how to develop the software for such systems and the effect that different hardware designs can have on the software and vice versa. It is this system design knowledge that has been captured in this book as a series of tutorials on the various aspects of embedded systems design.

Chapter 1 defines what is meant by the term and in essence defines the scope of the rest of the book. The second chapter provides a set of tutorials on processor architectures explaining the different philosophies that were used in their design and creation. It covers many of the common processor architectures ranging from 8 bit microcontrollers through CISC and RISC processors and finally ending with digital signal processors and includes information on the ARM processor family.

The third chapter discusses different memory types and their uses. This has been expanded in this edition to cover caches in more detail and the challenges associated with them for embedded design. The next chapter goes through basic peripherals such as parallel and serial ports along with timers and DMA controllers. This theme is continued in the following chapter which covers analogue to digital conversion and basic power control.

Interrupts are covered in great detail in the sixth chapter because they are so essential to any embedded design. The different types that are available and their associated software routines are described with several examples of how to use them and, perhaps more importantly, how not to use them.

The theme of software is continued in the next two chapters which cover real-time operating systems and software development. Again, these have a tremendous effect on embedded designs but whose design implications are often not well understood or explained. Chapter 9 discusses debugging and emulation techniques.

The remaining five chapters are dedicated to design examples covering buffer and data structures, memory and processor performance trade-offs and techniques, software design examples including using a real-time operating system to create state machines and finally a couple of design examples. In this edition, an example real-time system design is described that uses a non-real-time system to create an embedded system. The C source code is provided so that it can be run and experimented with on a PC running MS-DOS.

Steve Heath

Acknowledgements

By the nature of this book, many hardware and software products are identified by their tradenames. In these cases, these designations are claimed as legally protected trademarks by the companies that make these products. It is not the author's nor the publisher's intention to use these names generically, and the reader is cautioned to investigate a trademark before using it as a generic term, rather than a reference to a specific product to which it is attached.

Many of the techniques within this book can destroy data and such techniques must be used with extreme caution. Again, neither author nor publisher assume any responsibility or liability for their use or any results.

While the information contained in this book has been carefully checked for accuracy, the author assumes no responsibility or liability for its use, or any infringement of patents or other rights of third parties which would result.

As technical characteristics are subject to rapid change, the data contained are presented for guidance and education only. For exact detail, consult the relevant standard or manufacturers' data and specification.

This Page Intentionally Left Blank

1 What is an embedded system?

Whenever the word microprocessor is mentioned, it conjures up a picture of a desktop or laptop PC running an application such as a word processor or a spreadsheet. While this is a popular application for microprocessors, it is not the only one and the fact is most people use them indirectly in common objects and appliances without realising it. Without the microprocessor, these products would not be as sophisticated or cheap as they are today.

The embedding of microprocessors into equipment and consumer appliances started before the appearance of the PC and consumes the majority of microprocessors that are made today. In this way, embedded microprocessors are more deeply ingrained into everyday life than any other electronic circuit that is made. A large car may have over 50 microprocessors controlling functions such as the engine through engine management systems, brakes with electronic anti-lock brakes, transmission with traction control and electronically controlled gearboxes, safety with airbag systems, electric windows, air-conditioning and so on. With a well-equipped car, nearly every aspect has some form of electronic control associated with it and thus a need for a microprocessor within an embedded system.

A washing machine may have a microcontroller that contains the different washing programs, provides the power control for the various motors and pumps and even controls the display that tells you how the wash cycles are proceeding.

Mobile phones contain more processing power than a desktop processor of a few years ago. Many toys contain microprocessors and there are even kitchen appliances such as bread machines that use microprocessor-based control systems. The word control is very apt for embedded systems because in virtually every embedded system application, the goal is to control an aspect of a physical system such as temperature, motion, and so on using a variety of inputs. With the recent advent of the digital age replacing many of the analogue technologies in the consumer world, the dominance of the embedded system is ever greater. Each digital consumer device such as a digital camera, DVD or MP3 player all depend on an embedded system to realise the system. As a result, the skills behind embedded systems design are as diverse as the systems that have been built although they share a common heritage.

What is an embedded system?

There are many definitions for this but the best way to define it is to describe it in terms of what it is not and with examples of how it is used.

An embedded system is a microprocessor-based system that is built to control a function or range of functions and is not designed to be programmed by the end user in the same way that a PC is. Yes, a user can make choices concerning functionality but cannot change the functionality of the system by adding / replacing software. With a PC, this is exactly what a user can do: one minute the PC is a word processor and the next it's a games machine simply by changing the software. An embedded system is designed to perform one particular task albeit with choices and different options. The last point is important because it differentiates itself from the world of the PC where the end user does reprogram it whenever a different software package is bought and run. However, PCs have provided an easily accessible source of hardware and software for embedded systems and it should be no surprise that they form the basis of many embedded systems. To reflect this, a very detailed design example is included at the end of this book that uses a PC in this way to build a sophisticated data logging system for a race car.

If this need to control the physical world is so great, what is so special about embedded systems that has led to the widespread use of microprocessors? There are several major reasons and these have increased over the years as the technology has progressed and developed.

Replacement for discrete logic-based circuits

The microprocessor came about almost by accident as a programmable replacement for calculator chips in the 1970s. Up to this point, most control systems using digital logic were implemented using individual logic integrated circuits to create the design and as more functionality became available, the number of chips was reduced.

This was the original reason for a replacement for digital systems constructed from logic circuits. The microprocessor was originally developed to replace a mass of logic that was used to create the first electronic calculators in the early 1970s. For example, the early calculators were made from discrete logic chips and many hundreds were needed just to create a simple four function calculator. As the integrated circuit developed, the individual logic functions were integrated to create higher level functions. Instead of creating an adder from individual logic gates, a complete adder could be bought in one package. It was not long before complete calculators were integrated onto a single chip. This enabled them to be built at a very low cost compared to the original machines but any changes or improvements required that a new