EMBEDDED SYSTEMS

A Contemporary Design Tool

JAMES K. PECKOL

Embedded Systems
A Contemporary Design Tool

James K. Peckol, Ph.D.
University of Washington

BICENTENNIAL 1807 WILEY 2007 BICENTENNIAL

JOHN WILEY & SONS, INC.

Executive Publisher	*Don Fowley*
Associate Publisher	*Dan Sayre*
Acquisitions Editor	*Catherine Shultz*
Project Editor	*Gladys Soto*
Editorial Assistant	*Chelsee Pengal*
Marketing Manager	*Chris Ruel*
Production Editor	*Lea Radick*
Cover Designer	*Michael St. Martine*
Cover Image	*©Megumi Takamura/Dex Image/Getty Images*
Bicentennial Logo Design	*Richard J. Pacifico*

This book was set in *Times Ten* by *Preparé* and printed and bound by *Hamilton Printing*. The cover was printed by *Phoenix Color*.

Library of Congress Cataloging-in-Publication Data

Peckol, James K.
Embedded system: a contemporary design tool / James K. Peckol.
p. cm.
ISBN 978-0-471-72180-2 (cloth)
1. Embedded computer system. 2. Object oriented methods (Computer science)
I. Title.
TK7895.E42P43 2008
004.16--dc22

2007017870

ISBN 978-0-471-72180-2

Risk = Probability of Failure • Severity

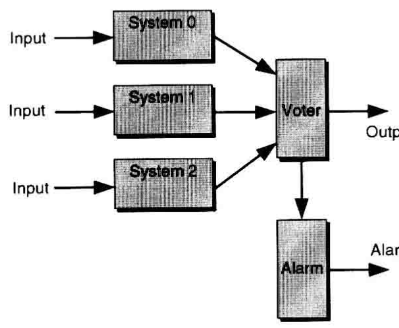
Increased Risk → Decreased Safety

- Find out what the customer wants
- Think of a way to give them what they want
- Prove what you've done by building and testing it
- Build a lot of them to prove that it wasn't an accident
- Use the product to solve the customer's problem

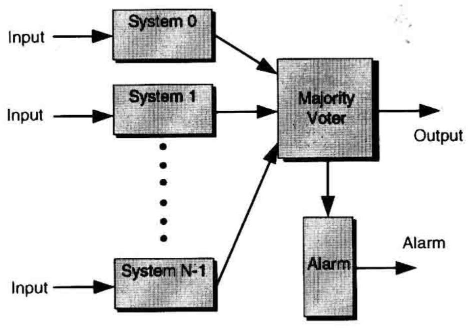
Fundamentals of Design

- Requirements Definition
- System Specification
- Functional Design
- Architectural Design
- Prototyping

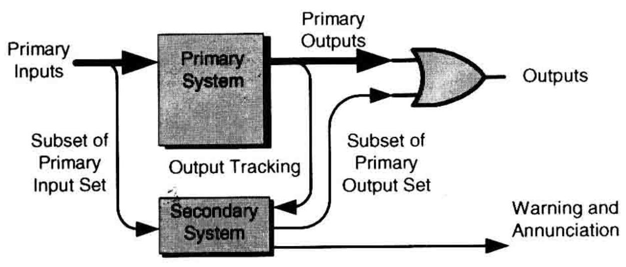
Five Steps to a Successful Design



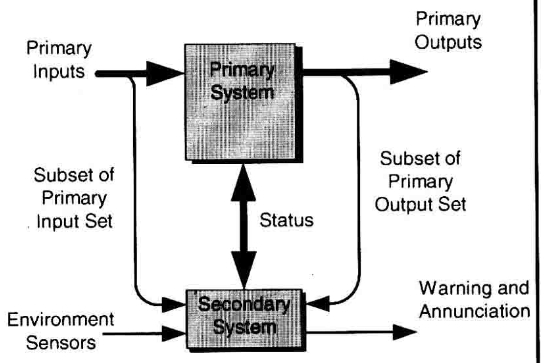
Triple Module Redundancy



N Module Redundancy



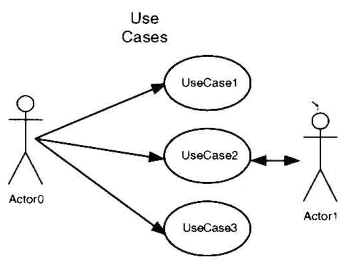
Lightweight Redundancy

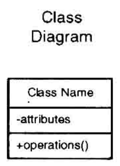


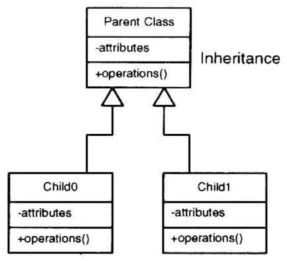
Monitor Only System Configuration

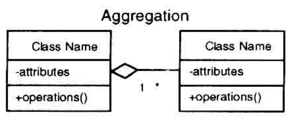
UML Diagrams

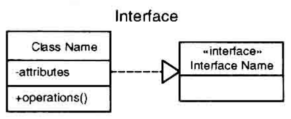
Static Relationships

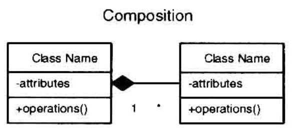




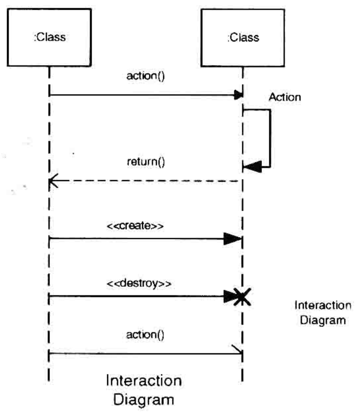


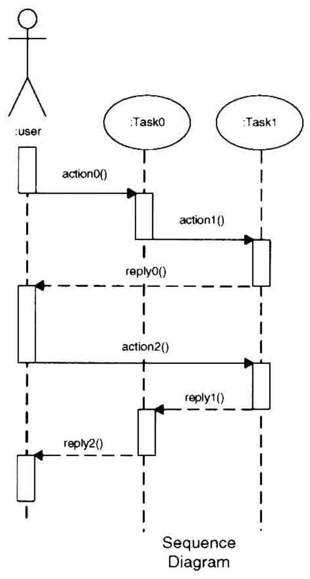


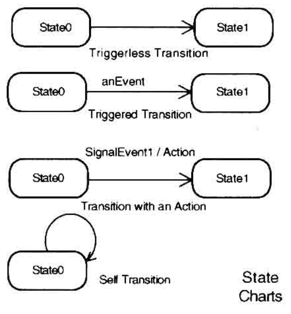


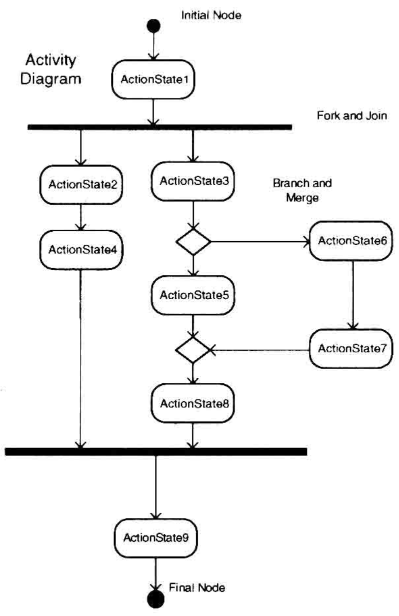


Dynamic Relationships

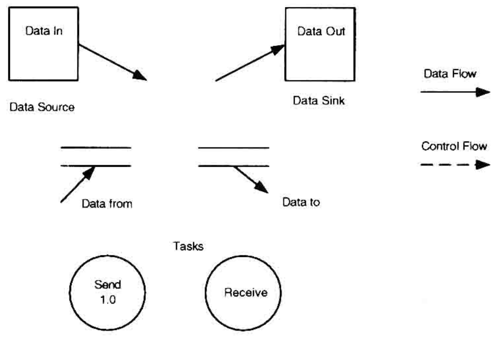








Data and Control Flow Diagrams



Embedded Systems

A Contemporary Design Tool

BICENTENNIAL 1807 WILEY 2007 BICENTENNIAL

THE WILEY BICENTENNIAL–KNOWLEDGE FOR GENERATIONS

Each generation has its unique needs and aspirations. When Charles Wiley first opened his small printing shop in lower Manhattan in 1807, it was a generation of boundless potential searching for an identity. And we were there, helping to define a new American literary tradition. Over half a century later, in the midst of the Second Industrial Revolution, it was a generation focused on building the future. Once again, we were there, supplying the critical scientific, technical, and engineering knowledge that helped frame the world. Throughout the 20th Century, and into the new millennium, nations began to reach out beyond their own borders and a new international community was born. Wiley was there, expanding its operations around the world to enable a global exchange of ideas, opinions, and know-how.

For 200 years, Wiley has been an integral part of each generation's journey, enabling the flow of information and understanding necessary to meet their needs and fulfill their aspirations. Today, bold new technologies are changing the way we live and learn. Wiley will be there, providing you the must-have knowledge you need to imagine new worlds, new possibilities, and new opportunities.

Generations come and go, but you can always count on Wiley to provide you the knowledge you need, when and where you need it!

William J. Pesce | Peter Booth Wiley

WILLIAM J. PESCE
PRESIDENT AND CHIEF EXECUTIVE OFFICER

PETER BOOTH WILEY
CHAIRMAN OF THE BOARD

Preface

INTRODUCING EMBEDDED SYSTEMS

Less than 150 years ago, shipping a new product, petroleum, down the Mississippi in barges was viewed with skepticism and fear of possible explosion. Fifty years later, electricity and electric lights were viewed as marvels of modern technology available only to a few. Another 50 years subsequent, someone suggested that the world would need at most three to four computers. Our views continue to change. Today we ship petroleum (still with concern) all over the world. Electricity has become so common that we are surprised if a switch is not available to turn on a light when we enter a room. The need for three to four computers has grown to hundreds of millions, perhaps billions, of installed computers worldwide.

This book presents a contemporary approach to the design and development of a kind of computer system that most of us will never see—those that we call embedded systems. The approach brings together a solid theoretical hardware and software foundation with real-world applications. Why do we need such a thing? A good question, let's take a look.

Today we interact with an embedded computer in virtually every aspect of our everyday life. From operating our car to riding an elevator to our office to doing our laundry or cooking our dinner, a computer is there, quietly, silently doing its job. We find the microprocessor—microcomputer—microcontroller—everywhere. Today these machines are ubiquitous. Like the electric light, without thought, we expect the antilock braking system in our car to work when we use it. We expect our mobile phone to operate like the stationary one in our home. We carry a computer in our pocket that is more powerful than the ones the original astronauts took into space.

Today we have the ability to put an increasingly larger number of hardware pieces into diminishingly smaller spaces. Software is no longer relegated to a giant machine in an air-conditioned room; our computer and its software go where we go. This ability gives engineers a new freedom to creatively put together substantially more complex systems with titillating functionality, systems that only science fiction writers thought of a few years ago. Such an ability also gives us the opportunity to solve bigger and more complex problems than we have ever imagined in the past—and to put those designs into smaller and smaller packages. These are definitely the most fun problems, the exciting kinds of things that we are challenged to work on. Okay, where do we begin?

The embedded field started almost by accident not too many years ago. In the early 70s Federico Faggin and many others at Intel and Motorola introduced the 4004, 8008, and 6800 microprocessors to the engineering world. Originally intended for use in calculators and in calculator-like applications, today, driven by evangelists like Faggin, the microprocessor has become a fundamental component of virtually everything we touch. With such widespread application, the ensured safety and reliability of such systems are absolutely essential.

The embedded systems field has grown virtually overnight from nonexistent several years ago to encompass almost every aspect of modern electrical engineering and computing science. Embedded systems are almost unique in this respect. Although certainly other disciplines within electrical engineering and computing science utilize the knowledge of other fields, it is essential for those studying and working with embedded systems to develop multidisciplinary skills, particularly in the areas of digital hardware and software. Electrical and computer engineers, working with embedded systems, contribute to all aspects of the development process from planning and design to manufacturing and marketing.

The embedded systems field is also a bit of an enigma. Unlike the fields of mathematics, physics, or chemistry, embedded systems have evolved from the engineer's workbench rather than from the scientist's research lab. Much of our formal theory has roots in the efforts of skilled engineers and computer scientists whose work has been quickly adapted to the factory floor. The field of embedded systems is more like a large umbrella. The systems designed under that umbrella require skills from many diverse fields. Without those skills, embedded systems cannot exist. Herein lies one of the dilemmas of trying to write a book on the field. Finding the right balance between depth and breadth can be a significant challenge. Hopefully, we have approached a good and useful balance.

This text is based on a vast store of theoretical and practical knowledge gained in developing safe, highly reliable embedded applications over the years for the aerospace, commercial, and medical industries. We endeavor to present the material in interesting, exciting, and challenging ways. We hope that we have succeeded and that this text will create lots of opportunities for you to explore and to learn further.

SELECTING A LANGUAGE AND TOOLS

This book contains a rich collection of real-world hardware and software examples. In both areas, we have a variety of ways through which we can turn our ideas and designs into real-world hardware and software components. Perhaps someday we will develop a universal language in which we can express all applications from business to science to engineering. Perhaps someday we will talk to our computer, and it will effortlessly perform our requests—maybe even making suggestions along the way. We're not there yet. The hardware and software concepts we study here are largely language independent. In this book, as we take the step from concept to realization, we will use the Verilog language as a modeling and synthesis tool to express the hardware implementation, the Unified Modeling Language (UML) and structured design to model the software designs, and the C language to affect the software implementation. Although beyond the scope of this text, modeling the hardware and software functions of our design is central to the developing field called hardware-software co-design. Moving to other implementation languages and tools should be rather straightforward. For those readers not fully versed in Verilog or C, we provide a good introduction to and overview of the fundamentals of both tools.

ORGANIZING THE BOOK

It is often all too easy to hack together a one-off embedded application that works. Trying to replicate a million or more copies of such a design (with tight time constraints) very quickly runs into the real-world gremlins that are waiting for us. A solid, robust, reliable design must always be based on the underlying theory and a disciplined design approach. Such methods are growing increasingly important as we continue to push the design envelop.

This book takes a developer's perspective to teaching embedded systems concepts. It examines, in detail, each of the important theoretical and practical aspects that one must consider when designing today's applications. These include the formal hardware and software development process (stressing safety and reliability); the digital and software architecture of the system; the physical world interface to external analog and digital signals; the debug and test throughout the development cycle; and finally improving the system's performance.

THE CHAPTERS

Introduction and Background

The *Foreword* gives an introductory overview, some of the vocabulary that is part of the embedded world, a bit of background and history, and a few contemporary examples.

Hardware and Software Infrastructure

With a preliminary background set, the next several chapters cover the essential aspects of the hardware and software necessary for the design and development of contemporary embedded systems. The Verilog hardware design language, Unified Modeling Language, and structured design models are introduced as tools in support of the development process.

Chapter 1 provides the first formal look at embedded systems and introduces some basic concepts, approaches, and vocabulary. The chapter begins with the hardware and computing core, which is usually manifest as a microprocessor, microcomputer, or microcontroller, and follows with an introduction to and discussion of the classic von Neumann and Harvard architectures.

Next, at the opposite end of the system hierarchy, methods by which the bits, bytes, and volts can be interpreted as the various and essential kinds of information (numbers, characters, addresses, and instructions) found inside of an embedded system are studied. Building on the instructions, the instruction set architecture (ISA) and register transfer (RTL) levels of the computer are introduced and studied.

Chapters 2 and 3 address a portion of the hardware side of embedded system design. The material provides a solid basis for practical aspects of working with digital circuits and systems in the embedded world. The Verilog hardware design language is used as a modeling tool in the design and synthesis of combinational and sequential logics. Time constraints and related issues at the hardware level are introduced as critical considerations in embedded applications. Difficulties with and solutions to problems of asynchronous system I/O are examined. Effective clocking schemes for the design of robust digital hardware move the reader into the synchronous world.

Embedded systems work and sometimes fail in the real world. As part of a recurring emphasis on the need for safe, robust, and reliable designs, several of the more common failure modes in combinational and sequential hardware as well as methods for testing for such failures are presented. Those who already have a good background in digital design or hardware design languages can still benefit from going over the material on timing, time constraints, and the effects of parasitic devices or reviewing the Verilog examples.

Chapter 4 looks at how memory is used in embedded systems. This section begins with an examination of registers and cache and studies several of the more commonly used cache organizations and schemes. Next, the static and dynamic allocation of memory and their impact on performance in real-time embedded designs are studied. Finally, the stack data type and how it is used in multitasking design is examined.

Chapter 5 presents the major UML modeling diagrams that are relevant to the material subsequently presented in the text and later moves to the data and control flow diagram from the structured design approach to system modeling and development. The chapter introduces UML-based static and dynamic models of the software. The static view, which begins from outside the system, is refined to increasing levels of detail to capture the comprising modules, their relationships, and their communication paths.

The dynamic view expresses the behavior of the system while it is performing its intended tasks and provides information about interactions among tasks. Concurrent task operation and persistence are introduced and discussed as two of the more important dynamic considerations in anticipation of subsequent studies of tasks, intertask communication, scheduling, and the operating system.

Chapters 6 and 7 provide a review of the core elements of the C language as well as of several of the more commonly used data structures and algorithms necessary for developing embedded applications. Whether you are an experienced programmer or know just enough to get into trouble, this material guides you through developing software for an embedded environment. The chapters introduce the C basics with specific coverage of variables, storage types, scopes, addresses, pointers, and structs. Bit operations are presented as an essential tool for working with hardware signals. Functions, function calls, and pointers to functions are introduced and discussed in the context of embedded applications.

Developing the Foundation

The next few chapters present the embedded system development process based on the need to deliver a safe and reliable design. The development section closes as it does in the real world with the debug and test processes.

Chapter 8 introduces the basic concepts of safety, reliability, and robustness in embedded applications, formulates definitions for each, and identifies their differences. Several real-world examples in which minor oversights have led to either significant or potentially significant and costly failures are examined. After establishing some of the relevant vocabulary and the need for robust and reliable applications, several design approaches are presented to help to ensure those needs are met. The chapter concludes with the introduction of some tools and techniques that can be used to detect and manage problems that may occur during system operation.

Chapter 9 formalizes the embedded systems design and development process. Several different manifestations of the development life cycle are presented, studied, and analyzed. The reader is introduced to several traditional approaches to system design. Such approaches utilize models and model-based development, both of which are becoming increasingly critical in the design of today's highly complex systems. The primary tools in such discussions will be Verilog models, structured design techniques, and UML. Approaches for assessing and criticizing the quality and robustness of a design are presented and discussed. The chapter concludes, as the design must also, with an examination of the core elements in a design release package.

Chapter 10 contains concepts and material that are always relevant. Though certainly no substitute for a sound design process, debug, test, and troubleshooting are essential components throughout the process of developing embedded systems. This chapter begins by motivating the need for testing in both hardware and software. Then, starting with the predebug phase of a project, the presentation moves through module, subsystem, and system debug and test. Included are discussions of test process and associated specifications, test case design, alpha and beta testing, then production test as well as self-test and agency-driven testing.

Doing the Work

The next chapters build on the foundation established earlier to develop the application as a collection of interacting tasks under the management of a real-time operating system. Deadlock problems arising from such designs are examined. Prior to moving outside of the microprocessor in the following section, methods for analyzing and optimizing the performance of an embedded application are presented.

Chapters 11 and 12 provide an introduction to and motivation for tasks, multitasking, and the control of an embedded application. Beginning with the necessary terminology, the material examines the critical role of time in developing and deploying many embedded applications, and presents a first look at time-based and reactive systems. The chapter identifies the central responsibilities of an operating system, examines the characteristics and capabilities that distinguish a Real-Time Operating System (RTOS), and then examines the core set of requirements of the OS as embodied in the kernel.

Study then shifts to the fundamentals of flow of control, communication, and detailed timing in embedded applications. The discussion begins with event-driven control schema based on simple polling, interrupts, and associated handling mechanisms. Topics of interest include intertask communication, data and resource sharing, and task synchronization through semaphores and monitors. Scheduling, scheduling algorithms, and methods for evaluating scheduling algorithms in a real-time context round out the topic.

Chapter 13 continues the study of schedulers by examining the problem of deadlocks and starvation in multitasking embedded applications. Several methods for avoiding, preventing, identifying, and resolving deadlocks, as well as ensuring progress through the system, are described and discussed.

Chapter 14 examines performance and the quantification and evaluation of performance in embedded designs. To begin the study, several different metrics are introduced and discussed. An analysis of several important metrics—response times, time loading, and memory loading in embedded applications—follows. The chapter also studies the evaluation and optimization of time and power consumption aspects of performance. By looking at the opposite side of performance, several common errors in analysis of performance measures are explored and evaluated.

Interacting with the Physical World

Continuing the design and development of an embedded application, the scope is expanded first to local peripheral devices and then to more remote ones. The next several chapters move outside of the processor and into the physical world that includes working with a wide variety of different kinds of signals. First, a model of the interaction is developed as an extension of that developed earlier in Chapter 12, and then specific applications are examined in the context of that model.

Chapters 15 and 16 open the study by exploring how an embedded application can interact with the external world. The internal interprocess and communication model developed earlier is expanded to include information, control and synchronization, and addressing in the external world and is extended to include a transport component. Following the introductory discussion, each component is studied in detail from the points of view of a shared variable (local) and a message-based (remote) model of information exchange. The objective of these chapters is to establish the basic infrastructure and various implementation architectures for both the local and remote models of external world interaction.

Chapter 17 focuses on the typically local analog and digital I/O interface to the external world. The chapter begins with several different methods for generating analog output signals and then looks at how various physical world analog input signals can be converted into a digital form. Three specific conversion algorithms—dual slope, successive approximation, and voltage to frequency—are studied. Because the outputs of the various sensors and transducers are often nonlinear, the problem of working with such signals is examined.

The chapter next introduces the topic of generating digital signals as control inputs to several different kinds of small motors, including stepper and servo motors, and as information that must be displayed. The discussion of digital I/O concludes by studying how time and frequency parameters of digital signals can be measured.

Chapter 18 examines the world in which interaction with the external world devices takes place via a network. The chapter introduces four different, commonly used network-based input/output designs. The study of each begins with the problems that motivated the development of the interface. Analysis of each design includes the transport mechanism, the control and synchronism scheme used, and the identification of message senders and receivers in the context of the model of intertask communication and synchronization developed earlier.

The chapter opens with the traditional RS-232 standard asynchronous serial interface, follows with a synchronous approach utilized by the Universal Serial Bus, and then examines the I^2C bus and the CAN bus. The objective is to establish the basic infrastructure and various implementation architectures for both local and remote models of external world interaction.

Chapter 19 provides an introduction to programmable logic devices (PLDs). The chapter begins with a brief discussion motivating the use of such devices in embedded systems and then examines the underlying logical concepts that have led to their development and widespread use. Next, the commonly used technologies for implementing programmable devices are examined. The basic structure of the components, variations on I/O configurations, and the fundamental architectures for the Complex Programmable Logic Device (CPLD) and the field programmable gate array (FPGA) are then presented.

As representative examples of PLD architectures, two of the more commonly used components—the CPLD and the Gate Array—as well as a more general-purpose device called a Programmable System on a Chip are presented. The chapter concludes with a look at several applications.

Supporting and Background Material

The first appendix is an introductory Verilog tutorial. The second provides a number of laboratory projects of increasing complexity that can be used to reinforce the practical application of the theory underlying the design of embedded systems.

Appendix A introduces the Verilog language and presents the important features and capabilities it then used in this book. The material begins with the basic components and organization of a Verilog program; examines the behavioral, dataflow, and gate-level or structural models for combinational logic circuits, and follows with similar models for sequential circuits. Design is only one element of the product development; each design must also be tested to confirm that it meets specified requirements. To that end, each section also discusses how one can formulate test suites to verify the proper operation. The material on testing will lay the foundation to guide the developer in building test cases for performing testing to the desired level. It is beyond the scope of this text to present a comprehensive treatise on testing.

Appendix B, found on the text's companion website, www.wiley.com/college/peckol, gives a number of lab exercises that are classified into three categories: *Getting Started, Developing Skills,* and *Bringing It Together*. The exercises in the first category suggest some basic projects that introduce some of the fundamental requirements of an embedded system such as bringing information into the microprocessor, using that information in an application, and producing some outputs. Projects in the second category are more complex. Many of these require a multitasking implementation, although they do not require an operating system. They utilize many of the peripheral devices commonly found in an embedded microprocessor, microcomputer, or microcontroller-based design. Projects in the third category represent simplified examples of real-world applications. These projects cover the complete product development life cycle from identifying requirements through design and test.

Additional Materials

A great variety of additional support material is available on the book's companion website, www.wiley.com/college/peckol. This includes information freely available to everyone such as the latest errata and additional background tutorials covering the basics of digital design and the C language fundamentals.

On the instructor's portion of the site, among other things, we include Power Point slides of all of text's figures and Appendix B which was described previously.

THE AUDIENCE

The book is intended for students with a broad range of background and experience and also serves as a reference text for those working in the field. The core audience should have at least one quarter to one semester of study in logic design, facility with a high-level programming language such as C, C++, or Java, and some knowledge of operating systems, and should be an upper-level junior or senior or lower-level graduate student.

NOTES TO THE INSTRUCTOR

This book can be a valuable tool for the student in the traditional undergraduate electrical engineering, computer engineering, or computer science programs as well as for the practicing engineer who wishes to review the basic concepts. Here the student may study the five essential aspects of the development of contemporary embedded systems, and is notably given a solid presentation of hardware and software architecture fundamentals, a good introduction to the design process and formal methods (including safety and reliability), the study of contemporary real-time kernel and operating system concepts, a comprehensive presentation of the interface to local and distributed external world devices, and finally debug and test of the designs.

Key to the presentation is a substantial number of worked examples illustrating fundamental ideas as well as how some of the subtleties in application go beyond basic concepts. Each chapter opens with a list of *Things to Look For* that highlight the more important material in the chapter and concludes with review questions and thought questions. The review questions are based directly on material covered in the chapter and mirror and expand on the *Things to Look For* list. They provide the student a self-assessment of their understanding and recall of the material covered. Though based on the material covered in the chapter, the thought questions extend the concepts as well as provide a forum in which the student can synthesize new ideas based on those concepts. Most chapters also include an extensive set of problems to permit the student to begin to apply the theory. These do not require laboratory support; however, they could be easily extended into basic lab projects. Included in *Appendix B*, found on the text's companion website, www.wiley.com/college/peckol, are 23 in-depth laboratory exercises.

The text is written and organized much as one would develop a new system, from the top down, building on the basics. Ideas are introduced and then revisited throughout the text, each time to a greater depth or in a new context. Busses may appear in the first few paragraphs to introduce the idea, later used to interconnect system components, and analyzed at a detailed level as the concepts of critical timing and data movement are studied. Safety and reliability are absolutely essential components in the development of any kind of system today. Such material is placed near the front of this text to emphasize its importance. The goal is to have the student think about such issues as he or she learns about and designs embedded applications.

As we stated in the opening of this Preface, finding a good balance between depth and breadth in an embedded systems text is a challenge. To that end, a couple of decisions were made at the outset. First, the text is not written around a specific microprocessor. Rather, the material is intended to be relevant to (and has been used to develop) a wide variety of applications running on many different kinds of processors. Second, the embedded field is rapidly changing even as this sentence is being typed and read. In lieu of trying to pursue and include today's latest technologies, the focus is on the basics that apply to any of the technologies. It is the underlying philosophy of this book that the student well grounded in the fundamentals will be comfortable working with and developing state-of-the-art systems utilizing the newest ideas. Ohm's law hasn't changed for many years; the field of electrical engineering has.

The core material has been taught as a one-quarter senior-level course in embedded systems development for approximately nine years. Roughly two-thirds of the material has been successfully taught for several years as a three-quarter on-site and distance learning outreach program to a population of students with rather diverse backgrounds. The outreach students have typically been working in industry for at least five years post-bachelor's degree.

Based on student background, the text is sufficiently rich to provide material for a two- to three-quarter or two-semester course in embedded systems development at the junior to senior level in a traditional four-year college or university. Beyond the core audience, the sections covering the assumed foundation topics can provide a basis on which the student with a limited hardware or software background can progress to the remainder of the material. The logic and software sections are not sufficiently deep to replace the corresponding one- or two-quarter courses in the topics. For those with adequate background in such areas, the material can either be skipped or serve as a brief refresher. Students with a Java background may find the material on pointers, bitwise operators, and structs to be particularly useful. The same holds for portions of the material on operating systems; such material is not intended to replace a formal, in-depth operating systems course. As deemed appropriate, the material may be skipped, used as a good refresher, or serve to introduce topics unique to embedded applications.

THE AUTHOR

The author's background spans over 40 years as an engineer and educator in the field of software, digital, and embedded systems design and development. As an engineer in the aerospace, commercial, and medical electronics industries, the author has worked on test systems for military aircraft navigation systems and radar systems, the *Apollo* color camera, various weather satellites, the Mars *Viking Lander*, flight control systems for a number of commercial aircraft, production of high-quality electronic test instruments and measurement systems, and several defibrillation systems. Academic experience spans more than 20 years of developing and teaching software, digital design, networking, and embedded systems design courses for students with experience ranging from limited hardware or software background to those at the junior, senior, and graduate levels.

ABOUT THE COVER

The umbrella on the cover is based upon an original photograph by Megumi Takamura. The image was chosen because it expresses the idea that the embedded systems field covers applications that utilize knowledge and skills from almost every discipline in engineering and computing science. Like the cover on this book, if we open the cover of most contemporary products, we will find embedded designs being used as tools to enhance their features and capabilities.

ACKNOWLEDGMENT

Over the years, as I've collected the knowledge and experiences to bring this book together, there have been many, many people with whom I have studied, worked, and interacted. Our discussions, debates, and collaborations have led to the ideas and approach to design presented on the pages that follow.

While there are far too many to whom I owe a debt of thanks to try to list each here, I do want to give particular thanks to David L. Johnson, Corrine Johnson, Greg Zick, Tom Anderson, David Wright, Gary Anderson, Patrick Donahoo, Steve Swift, Paul Lantz, Mary Kay Winter, Kasi Bhaskar, Brigette Huang, Jean-Paul Calvez, Gary Whittington, Olivier Pasquier, Charles Staloff, Gary Ball, John Davis, Patrick F. Kelly, Margaret Bustard, and Donna Karschney for all they've done over the years. William Hippe and Alex Talpalatskiy, who spent many hours proofreading, commenting, and making valuable suggestions to improve the early versions of the text, deserve a special thank you.

From John Wiley, I want to thank Bill Zobrist who supported the original idea of publishing this text and especially Gladys Soto, the Project Editor who carried the development forward, the Production Editors Lea Radick and Lisa Wojick who brought everything together, and the unknown copyeditors, compositors, and others whose efforts on and contributions to this project have been invaluable.

In any project, design reviews are an essential part of producing a quality product. I wish to express my appreciation and thanks to this project's many reviewers for their evaluations and constructive comments, which helped guide its development.

John Acken	Oklahoma State University, Stillwater
Farrokh Attarzadeh	University of Houston
Saad Biaz	Auburn University
Phillip De Leon	University of Colorado-Boulder
Michael Eastman	Rochester Institute of Technology
Richard Fryer	Cal Poly, San Luis Obispo
Subra Ganesan	Oakland University
Adam Hoover	Clemson University
Kenneth Jacker	Appalachian State University
Phillip Laplante	Pennsylvania State University
Al Liddicoat	Cal Poly, San Luis Obispo
Tulin Mangir	California State University, Long Beach
Michael Morrow	University of Wisconsin-Madison
Brad Naegle	US Naval Postgraduate School
Kamesh Namuduri	Wichita State University

Kimberly Newman	University of Denver
Mitchell Nielsen	Kansas State University
Dan Phillips	Rochester Insituite of Technology
HR Shih	Jackson State University
Tom Stuart	University of Toledo
Jindong Tan	Michigan TechnologicalUniversity
Richard Wall	University of Idaho-Moscow
Hong Zhao	Embry Riddle Aeronautical University
Peixin Zhong	Michigan State University

Finally, I extend a thank you to my many teachers, friends, colleagues, and students who I've had the pleasure of knowing and working with over the years.

James K. Peckol, Ph.D.