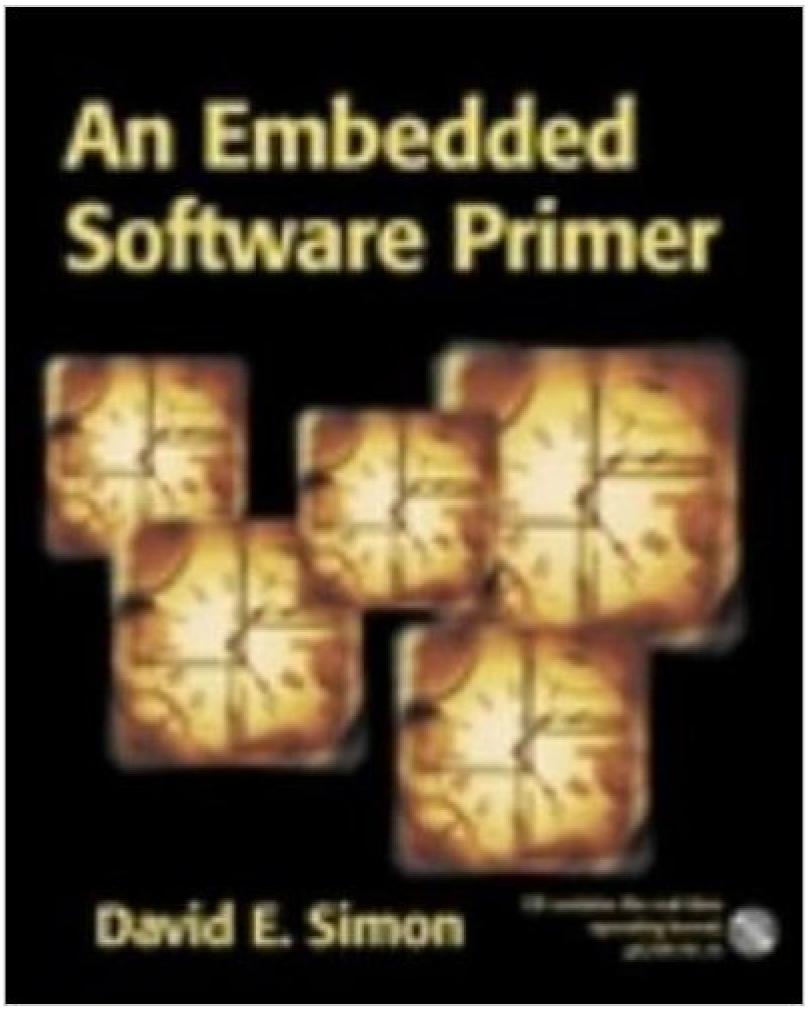
An Embedded Software Primer



Synopsis

Â "I sincerely wish (this book) had been available when I had to learn all this stuff the hard way."--Steve Vinoski   "An excellent job of introducing and defining the jargon associated with embedded systems. This makes the text extremely easy to read." --David Cuka   An Embedded Software Primer is a clearly written, insightful manual for engineers interested in writing embedded-system software. The example-driven approach puts you on a fast track to understanding embedded-system programming and applying what you learn to your projects. This book will give you the necessary foundation to work confidently in this field. Building on a basic knowledge of computer programming concepts, this book will help you to: Learn core principles and advanced techniques of embedded-system software. Find out what a real-time operating system (RTOS) does and how to use one effectively. Experiment with sample code and the Â C/OS RTOS version 1.11 (on the accompanying CD). Apply what you learn, no matter which microprocessor or RTOS you use.   After reading this book, you will be able to tackle the challenges of embedded system programming and quickly reap the benefits of your new skills.

Book Information

Paperback: 448 pages
Publisher: Addison-Wesley Professional; 1 edition (August 15, 1999)
Language: English
ISBN-10: 020161569X
ISBN-13: 978-0201615692
Product Dimensions: 7.3 x 1.2 x 8.9 inches
Shipping Weight: 1.3 pounds (View shipping rates and policies)
Average Customer Review: 4.4 out of 5 starsÂ Â See all reviewsÂ (35 customer reviews)
Best Sellers Rank: #99,672 in Books (See Top 100 in Books)   #7 inÂ Books > Computers & Technology > Hardware & DIY > Microprocessors & System Design > Embedded Systems   #110 inÂ Books > Computers & Technology > Business Technology > Software > Enterprise Applications   #113 inÂ Books > Textbooks > Computer Science > Software Design & Engineering

Customer Reviews

This is a great book for begineers. The author touches upon just about all aspects of software development for embedded systems. This information is usually learned the hard way - on the job. This book will give begineers a head start with the numerous examples of how to do things. And how NOT to do things. I have made it required reading for my software engineers - new and experienced.In addition, the book is easy reading. The author keeps things somewhat brief and to the point.

I started working on an embedded platform 1 year ago. Although now I have a good understanding of a lot of the aspects of the system, I never get the big picture.This book provides exactly what I needed.If you are a working software engineer and have spent years doing high level programming, the first few chapters will give you a good review of low-level-close-to-the-machine things that you need to know, which is also very useful for students as these are very important concepts that they need to understand to have a solid foundation to conquer higher level, more abstract CS subjects.In embedded system, bugs in task code can bring down the device. Author has done a very good job explaining how to protect shared data using mechanism provided by a RTOS. The communication between interrupt/task and among tasks are also discussed thoroughly.Unlike the other reviewer, I found the use of the C!! language in this book a very clever way to abstract away the hardware dependent code from the point being discussed. Consider it pseudo comment if you will.BTW, the excellent typesetting and use of fonts also makes it a very pleasant experience reading this book.Looking forward to see a more advanced text on embedded system from this author in the future.

I shouldn’t need to say more; If you’re interested in learning about writing software for embedded systems, this book will take you by the hand and get you there without a lot of fuss or difficulty in trying to figure out advanced topics from a 50,000ft perspective. It drills into each discussion using simplified but not simpleton characteristics of this writer’s excellent skill. I recommend this book to every software engineer starting on the path to developing embedded systems. The use of uCOS-II is a decent idea, especially for everyone interested in doing a little embedded systems development using PC hardware...however, I believe that it is a less-than perfect choice because of the rather limited compiler choices for uCOS-II, which should at least include GNU’s gcc, but doesn’t last time I checked.Update:Since the time of this review, uC/OS-II has continued to expand support for a wide variety of compilers/tool chains and dozens (if not hundreds) of micro choices.

Particularly useful for those just starting out in embedded software or for students at the bachelors level. Contains many real world-type examples and illustrates some common errors that we will all tend to make. Also, I know that electrical engineering students often don’t get exposure to operating system concepts and computer science students lack exposure to hardware issues. This book helps to smooth over what you may be missing in your education. For the practicing engineer, I recommend it, not so much for reference, but for re-enforcement and clarification of essential concepts. Also it will add quality to your code.

The book provides very useful information for anyone that wishes to learn embedded software from the ground up. Great for entry level engineers, or professionals wishing to make a lateral move into embedded systems. The book has two clearly written chapters dedicated to hardware fundamentals; describing I/O .vs. memory mapping, how interrupts function, memory types including PROM’s, and microprocessor basics. A chapter is dedicated to one of the single most inportant issues in embedded systems SHARED DATA and how to prevent corrupting it. Chapter 5 discusses four basic software methods to servicing interrupts. Chapters 6,7, and 8 clearly introduce the concept of an RTOS (Real Time Operating System). Chapter 9 is a must read for those new to embedded systems. It discusses development tools and the steps required to get your final code onto the target system. It explains what a cross-compiler is and why they are used in embedded systems. This is a very good book for engineers with C skills!

For someone starting up into embedded systems this book is excellent. For an experienced developer this book makes for a nice bedtime read. The hardware overview is something that all software developers should understand but many don’t take the time. Who needs an RTOS ? In todays fragmented embedded world, where we have 8051’s at one end, and StrongArms or Power PC’s at the other end of the spectrum, David Simon does a good job of presenting the available options. Table 5-1, "Characteristics of Various Software Architectures" is worth the price of the book. My only real complaint is that on the cover of the book the word ’kernel’ has been spelled as ’kernal’.

The book is well organized, clearly written and full of real code examples. It addresses the common and not-so-common pitfalls, and you can tell it has been written by a "seasoned" engineer. The coverage on RTOS’s is excellent.

Download to continue reading...

Systems (Embedded Technology) Analog Interfacing to Embedded Microprocessor Systems, Second Edition (Embedded Technology Series) Real-Time UML Workshop for Embedded Systems, Second Edition (Embedded Technology) Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers (Embedded Technology) TCP/IP Embedded Internet Applications (Embedded Technology) Linux for Embedded and Real-time Applications, Third Edition (Embedded Technology) Linux for Embedded and Real-time Applications (Embedded Technology) Linux for Embedded and Real-time Applications, Second Edition (Embedded Technology) Embedded Linux Primer: A Practical Real-World Approach (2nd Edition) Making Embedded Systems: Design Patterns for Great Software Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development Embedded Linux Systems with the Yocto Project (Prentice Hall Open Source Software Development) Make: Arduino Bots and Gadgets: Six Embedded Projects with Open Source Hardware and Software (Learning by Discovery) Embedded System Design: A Unified Hardware/Software Introduction Real-Time Software Design for Embedded Systems Better Embedded System Software