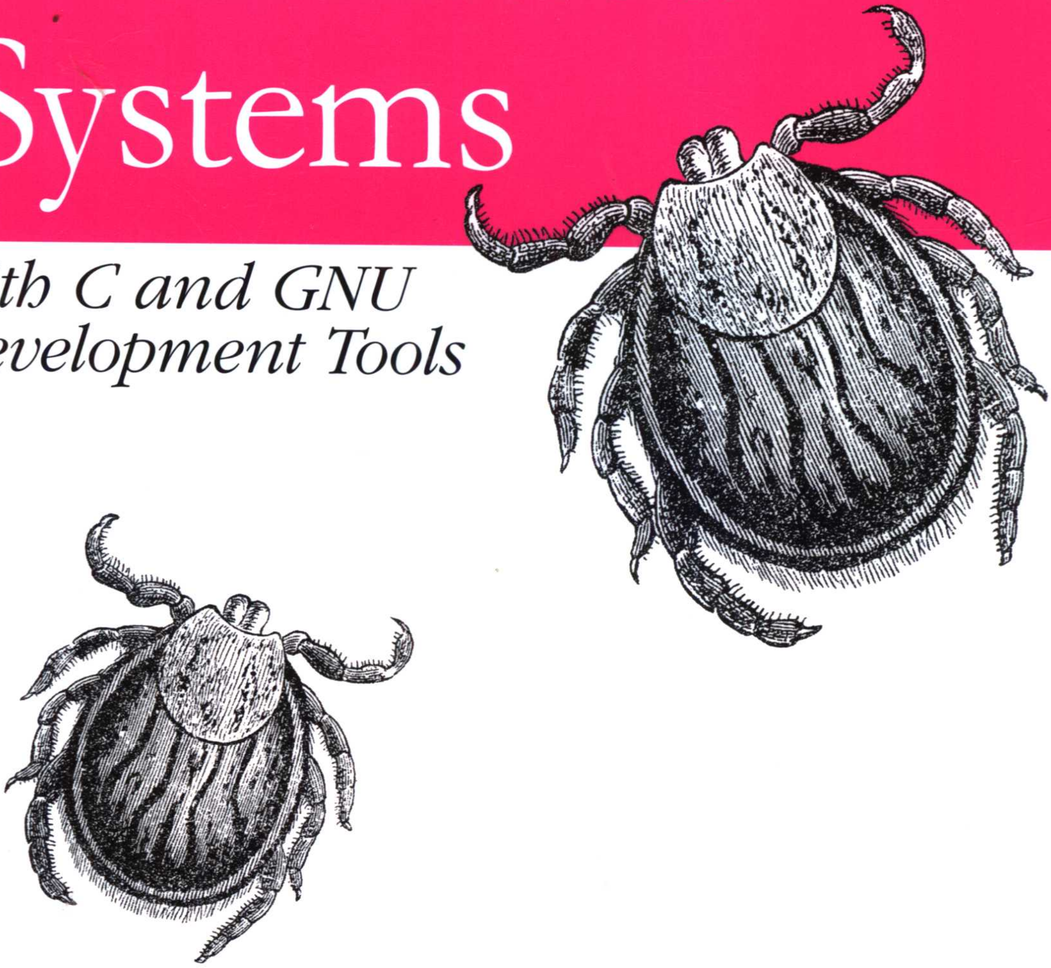
嵌入式系统编程（影印版）

2nd Edition
Includes Real-Time and Linux Examples

Programming Embedded Systems

with C and GNU Development Tools



O'REILLY®
東南大學出版社

Michael Barr & Anthony Massa 著
Jack Ganssle 序

第二版

嵌入式系统编程(影印版)

Programming Embedded Systems

Michael Barr and Anthony Massa

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

嵌入式系统编程：第2版：英文/（美）巴尔（Barr, M.），（美）马萨（Massa, A.）著.—影印本.—南京：东南大学出版社，2007.6

书名原文：Programming Embedded Systems, Second Edition

ISBN 978-7-5641-0772-7

Ⅰ.嵌… Ⅱ.①巴… ②马… Ⅲ.微型计算机－系统设计－英文 Ⅳ.TP360.21

中国版本图书馆 CIP 数据核字（2007）第 072072 号

江苏省版权局著作权合同登记

图字：10-2007-091 号

书　　名/ 嵌入式系统编程 第二版（影印版）
责任编辑/ 张烨
封面设计/ Edie Freedman，张健
出版发行/ 东南大学出版社（press.seu.edu.cn）
地　　址/ 南京四牌楼 2 号（邮政编码 210096）
印　　刷/ 扬中市印刷有限公司
开　　本/ 787 毫米 × 980 毫米　16 开本　20.5 印张
版　　次/ 2007 年 6 月第 1 版　2007 年 6 月第 1 次印刷
印　　数/ 0001-3000 册
书　　号/ ISBN 978-7-5641-0772-7/TP · 120
定　　价/ 45.00 元（册）

嵌入式系统编程（影印版）

Programming Embedded Systems

O'Reilly Media, Inc.介绍

O'Reilly Media, Inc.是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为二十世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc.一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc.是最稳定的计算机图书出版商 —— 每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc.具有深厚的计算机专业背景，这使得O'Reilly Media, Inc.形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc.所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc.还有许多固定的作者群体 —— 他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc.依靠他们及时地推出图书。因为O'Reilly Media, Inc.紧密地与计算机业界联系着，所以O'Reilly Media, Inc.知道市场上真正需要什么图书。

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc.达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的影印版图书，包括：

For my son, Vikram.

—Michael Barr

This book is dedicated to
my wonderful daughters, Katie and Ashley,
and my beautiful wife, Deanna.
You mean everything to me. I love you.

—Anthony Massa

Foreword

If you mention the word *embedded* to most people, they'll assume you're talking about reporters in a war zone. Few dictionaries—including the canonical *Oxford English Dictionary*—link *embedded* to computer systems. Yet embedded systems underlie nearly all of the electronic devices used today, from cell phones to garage door openers to medical instruments. By now, it's nearly impossible to build anything electronic without adding at least a small microprocessor and associated software.

Vendors produce some nine billion microprocessors every year. Perhaps 100 or 150 million of those go into PCs. That's only about one percent of the units shipped. The other 99 percent go into embedded systems; clearly, this stealth business represents the very fabric of our highly technological society.

And use of these technologies will only increase. Solutions to looming environmental problems will surely rest on the smarter use of resources enabled by embedded systems. One only has to look at the network of 32-bit processors in Toyota's hybrid Prius to get a glimpse of the future.

Though prognostications are difficult, it is absolutely clear that consumers will continue to demand ever-brainier products requiring more microprocessors and huge increases in the corresponding software. Estimates suggest that the firmware content of most products doubles every 10 to 24 months. While the demand for more code is increasing, our productivity rates creep up only slowly. So it's also clear that the industry will need more embedded systems people in order to meet the demand.

What skills will these people need? In the PC world, one must be a competent C/C++ programmer. But embedded developers must have a deep understanding of both the programming languages and the hardware itself; no one can design, code, and test an interrupt service routine, for instance, without knowing where the interrupts come from, how the hardware prioritizes them, the tricks behind servicing that hardware, and machine-level details about saving and preserving the system's context. A firmware developer must have detailed insight into the hardware implementation of his system's peripherals before he can write a single line of driver code.

In the PC world, the magic of the hardware is hidden behind an extensive API. In an embedded system, that API is always written by the engineers that are developing the product.

In this book, Michael Barr and Anthony Massa show how the software and hardware form a synergistic gestalt. They don't shy away from the intricacies of interrupts and I/O, or priority inversion and mutexes.

The authors appropriately demonstrate building embedded systems using a variety of open source tools, including the GNU compiler suite, which is a standard tool widely used in this industry. eCos and Linux, both free/open source products, are used to demonstrate small and large operating systems.

The original version of this book used an x86 target board, which has been replaced in this edition by an ARM-based product. Coincidently, as this volume was in production, Intel made an end-of-life announcement for all of its embedded x86 processors. Readers can be assured that the ARM will be around for a very long time, as it's supported by an enormous infrastructure of vendors.

The hardware is inexpensive and easily available; the software is free. Together they represent the mainstream of embedded systems development. Readers can be sure they'll use these tools in the future.

Buy the development kit, read the book, and execute the examples. You'll get the hands-on experience that employers demand: building and working with real embedded applications.

—Jack Ganssle

Preface

First figure out why you want the students to learn the subject and what you want them to know, and the method will result more or less by common sense.
—Richard Feynman

Embedded software is in almost every electronic device in use today. There is software hidden away inside our watches, DVD players, mobile phones, antilock brakes, and even a few toasters. The military uses embedded software to guide missiles, detect enemy aircraft, and pilot UAVs. Communication satellites, deep-space probes, and many medical instruments would've been nearly impossible to create without it.

Someone has to write all that software, and there are tens of thousands of electrical engineers, computer scientists, and other professionals who actually do. We are two of them, and we know from our personal experiences just how hard it can be to learn the craft.

Each embedded system is unique, and the hardware is highly specialized to the application domain. As a result, embedded systems programming can be a widely varying experience and can take years to master. However, one common denominator across almost all embedded software development is the use of the C programming language. This book will teach you how to use C in any embedded system.

Even if you already know how to write embedded software, you can still learn a lot from this book. In addition to learning how to use C more effectively, you'll also benefit from the detailed explanations and source code associated with common embedded software problems. Among the advanced topics covered in the book are memory testing and verification, device driver design and implementation, real-time operating system internals, and code optimization techniques.

Why We Wrote This Book

Each year, globally, approximately one new processor is manufactured per person. That's more than six billion new processors each year, fewer than two percent of which are the Pentiums and PowerPCs at the heart of new personal computers. You may wonder whether there are really that many computers surrounding us. But we bet that within five minutes you can probably spot dozens of products in your own home that contain processors: televisions, stereos, MP3 players, coffee makers, alarm clocks, VCRs, DVD players, microwaves, dishwashers, remote controls, bread machines, digital watches, and so on. And those are just the personal possessions—many more such devices are used at work. The fact that every one of those products contains not only a processor, but also software, is the impetus for this book.

One of the hardest things about this subject is knowing when to stop writing. Each embedded system is unique, and we have therefore learned that there is an exception to every rule. Nevertheless, we have tried to boil the subject down to its essence and present the things that programmers definitely need to know about embedded systems.

Intended Audience

This is a book about programming embedded systems in C. As such, it assumes that the reader already has some programming experience and is at least familiar with the syntax of the C language. It also helps if you have some familiarity with basic data structures, such as linked lists. The book does not assume that you have a great deal of knowledge about computer hardware, but it does expect that you are willing to learn a little bit about hardware along the way. This is, after all, a part of the job of an embedded programmer.

While writing this book, we had two types of readers in mind. The first reader is a beginner—much as we were once. He has a background in computer science or engineering and a few years of programming experience. The beginner is interested in writing embedded software for a living but is not sure just how to get started. After reading the first several chapters, he will be able to put his programming skills to work developing simple embedded programs. The rest of the book will act as a reference for the more advanced topics encountered in the coming months and years of his career.

The second reader is already an embedded systems programmer. She is familiar with embedded hardware and knows how to write software for it but is looking for a reference book that explains key topics. Perhaps the embedded systems programmer has experience only with assembly language programming and is relatively new to C. In that case, the book will teach her how to use the C language effectively in an embedded system, and the later chapters will provide advanced material on real-time operating systems, peripherals, and code optimizations.

Whether you fall into one of these categories or not, we hope this book provides the information you are looking for in a format that is friendly and easily accessible.

Organization

The book contains 14 chapters and 5 appendixes. The chapters can be divided quite nicely into two parts. The first part consists of Chapters 1 through 5 and is intended mainly for newcomers to embedded systems. These chapters should be read in their entirety and in the order that they appear. This will bring you up to speed quickly and introduce you to the basics of embedded software development. After completing Chapter 5, you will be ready to develop small pieces of embedded software on your own.

The second part of the book consists of Chapters 6 through 14 and discusses advanced topics that are of interest to inexperienced and experienced embedded programmers alike. These chapters are mostly self-contained and can be read in any order. In addition, Chapters 6 through 12 contain example programs that might be useful to you on a future embedded software project.

Chapter 1, *Introduction*
: Explains the field of embedded programming and lays out the parameters of the book, including the reference hardware used for examples

Chapter 2, *Getting to Know the Hardware*
: Shows how to explore the documentation for your hardware and represent the components you need to interact with in C

Chapter 3, *Your First Embedded Program*
: Creates a simple blinking light application that illustrates basic principles of embedded programming

Chapter 4, *Compiling, Linking, and Locating*
: Goes over the ways that embedded systems differ from conventional computer systems during program building steps, covering such issues as cross-compilers

Chapter 5, *Downloading and Debugging*
: Introduces the tools you'll need in order to iron out problems in both hardware and software

Chapter 6, *Memory*
: Describes the different types of memory that developers choose for embedded systems and the issues involved in using each type

Chapter 7, *Peripherals*
: Introduces the notion of a device driver, along with other coding techniques for working with devices

Chapter 8, *Interrupts*
: Covers this central area of working with peripherals

Chapter 9, *Putting It All Together*
: Combines the concepts and code from the previous chapter with convenience functions and a main program, to create a loadable, testable application

Chapter 10, *Operating Systems*
: Introduces common operating system concepts, including tasks (or threads) and synchronization mechanisms, along with the reasons for adding a real-time operating system

Chapter 11, *eCos Examples*
: Shows how to use some features of the eCos real-time operating system

Chapter 12, *Embedded Linux Examples*
: Accomplishes the same task as the previous chapter, but for the embedded Linux operating system

Chapter 13, *Extending Functionality*
: Describes options for adding buses, networking, and other communication features to a system

Chapter 14, *Optimization Techniques*
: Describes ways to decrease code size, reduce memory use, and conserve power

Appendix A, *The Arcom VIPER-Lite Development Kit*
: Describes the board used for the examples in this book and how to order one for yourself

Appendix B, *Setting Up Your Software Development Environment*
: Gives instructions for loading the software described in this book on your host Windows or Linux computer

Appendix C, *Building the GNU Software Tools*
: Shows you how to compile the GNU development tools

Appendix D, *Setting Up the eCos Development Environment*
: Shows you how to build an eCos library appropriate for your embedded system so you can compile programs to run on your system

Appendix E, *Setting Up the Embedded Linux Development Environment*
: Describes how to install the embedded Linux tools for your Arcom system and build and run a program on it

Throughout the book, we have tried to strike a balance between specific examples and general information. Whenever possible, we have eliminated minor details in the hope of making the book more readable. You will gain the most from the book if you view the examples, as we do, primarily as tools for understanding important concepts. Try not to get bogged down in the details of any one circuit board or chip. If you understand the general C programming concepts, you should be able to apply them to any embedded system you encounter.

To focus the book's example code on specific concepts, we intentionally left it incomplete—for example, by eliminating certain include files and redundant variable declarations. For complete details about the code, refer to the full example source code on the book's web site.

Conventions, Typographical and Otherwise

The following typographical conventions are used throughout the book:

Italic
: Indicates names of files, programs, methods, and options when they appear in the body of a paragraph. Italic is also used for emphasis and to introduce new terms.

`Constant Width`
: In examples, indicates the contents of files and the output of commands. In regular text, this style indicates keywords, functions, variable names, classes, objects, parameters, and other code snippets.

`Constant Width Bold`
: Indicates commands and options to be typed literally. This style is used in examples only.

`Constant Width Bold Italic`
: Indicates text to be replaced with user values; for example, a filename on your system. This style is used in examples only.

This symbol is used to indicate a tip, suggestion, or general note.

This symbol is used to indicate a warning.

Other conventions relate to gender and roles. With respect to gender, we have purposefully used both "he" and "she" throughout the book. With respect to roles, we have occasionally distinguished between the tasks of hardware engineers, embedded software engineers, and application programmers. But these titles refer only to roles played by individual engineers, and it should be noted that it can and often does happen that a single individual fills more than one of these roles on an embedded-project team.

Obtaining the Examples Online

This book includes many source code listing, and all but the most trivial snippets are available online. These examples are organized by chapter number and include build instructions (makefiles) to help you recreate each of the executables. The complete archive is available at *http://examples.oreilly.com/embsys2*.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Programming Embedded Systems with C and GNU Development Tools*, Second Edition, by Michael Barr and Anthony Massa. Copyright 2007 O'Reilly Media, Inc., 978-0-596-00983-0."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Comments and Questions

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, code examples, and any additional information. Corresponding files for code examples are mentioned on the first line of the example. You can access this page at:

http://www.oreilly.com/catalog/embsys2

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Safari® Enabled

Personal Comments and Acknowledgments

From Michael Barr

For as long as I can remember, I have been interested in writing a book or two. But now that I have written several, I must confess that I was naive when I started. I had no idea how much work it would take, or how many other people would have to get involved in the process. Another thing that surprised me was how easy it was to find a willing publisher. I had expected that to be the hard part.

I continue to be thankful to all of the following people for sharing their ideas and reviewing my work on the first edition: Toby Bennett, Paul Cabler (and the other great folks at Arcom), Mike Corish, Kevin D'Souza, Don Davis, Steve Edwards, Mike Ficco, Barbara Flanagan, Jack Ganssle, Stephen Harpster, Jonathan Harris, Jim Jensen, Mark Kohler, Andy Kollegger, Jeff Mallory, Ian Miller, Henry Neugauss, Chris Schanck, Brian Silverman, John Snyder, Jason Steinhorn, Ian Taylor, Lindsey Vereen, Jeff Whipple, and Greg Young.

I would also like to thank our editor, Andy Oram. Without his enthusiasm for my initial proposal, overabundant patience, and constant encouragement, neither the first nor the second edition of this book would have been completed.

And, of course, I am extremely thankful to Anthony Massa. Anthony's interest in updating this book with additional materials, new hardware and examples, and a change to the GNU tools came at just the right time. It has been difficult to watch someone else update a first edition that I felt good about and that sold so surprisingly well. But the new book is significantly better for Anthony's tireless efforts. This second edition would not exist if not for Anthony's hard work and dedication to the project.

From Anthony Massa

This is my second adventure in the realm of book writing. I thought writing a second edition would be a lot less work because most of the material was already finished. Boy, was I wrong. The second edition was as bit of a struggle and took more effort and time than I expected, but I think the book turned out better as a result.

I am very thankful to our editor, Andy Oram. His feedback was fantastic, he was a guiding light to push the book to completion, he always provided the needed spark to pull things together, and he even stepped in to test the code when needed. The second edition of this book is much better because of him and would not have been possible without his support and determination.

I would like to thank Michael Barr for the opportunity to work with him on this project. I know how attached a writer can become to such a project; thank you for entrusting me with the new edition. Michael provided extremely helpful input and helped me guide the text in the right direction. There were some struggles getting things just right, but I think that working through them has improved the book. Michael is truly a great mind in the embedded software development community.

Thanks to the folks at Arcom that so graciously provided the very impressive and top-notch development system for this book. A big thank you to Glen Middleton, who was always there to make sure I got whatever I needed. And thanks to Arcom's extremely helpful development team of Ian Campbell, Martyn Blackwell, and David Vrabel.

I am very fortunate that the following people gave their valuable time to help make this book better by sharing ideas and reviewing the second edition. This outstanding team was made up of Michael Boerner, John Catsoulis, Brian Jepson, Nigel Jones, Alfredo Knecht, Jon Masters, Tony Montiel, Andrea Pellegrini, Jack Quinlan, Galen Seitz, and David Simon. A special thanks to Jonathan Larmour for being there in the clutch when I had a question for you—you came through for me, again.

A special thanks to my A-1 review crew of Greg Babbitt, my brother Sean Hughes, Brian Kingston, Anthony Taranto, and Joseph Terzoli.

I would like to thank two great people for all their support throughout my life—Nonno and Nonna. They were always there for me with love and guidance.

Thanks to my brother, Laurie, and my sister, Catherine, for their support. I am grateful that both of you are in my life.