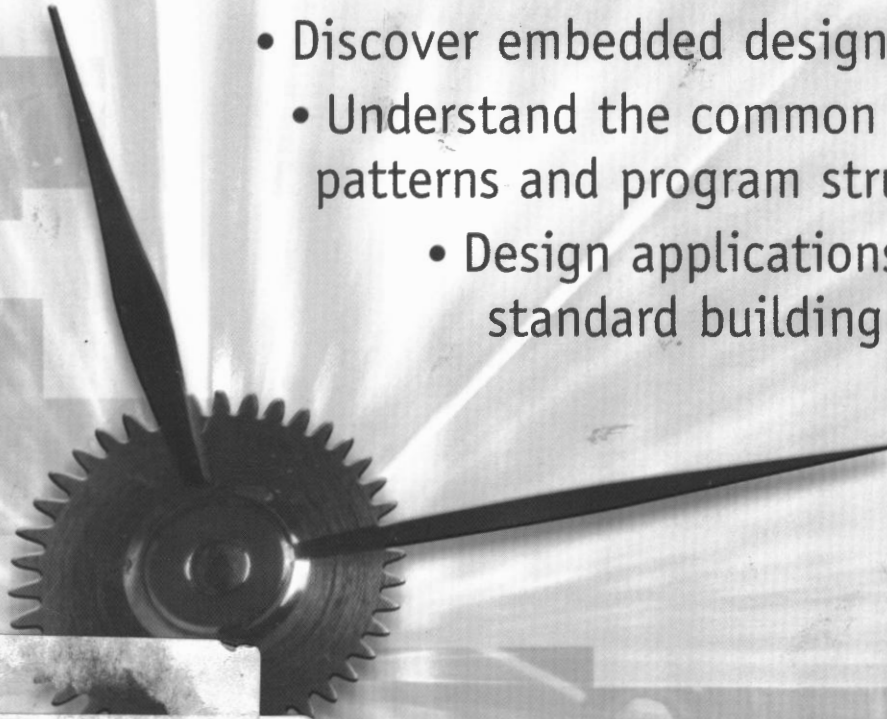


Real-Time Concepts for Embedded Systems

- Discover embedded design issues
- Understand the common design patterns and program structures
- Design applications using standard building blocks



CMPBooks

Qing Li

with Caroline Yao

INTEL EXHIBIT 1010 Page 1

IPR2023-00787

THIRTIETH BOOKS

Real-Time Concepts for Embedded Systems

Qing Li

with Caroline Yao

CMPBooks

San Francisco, CA • New York, NY • Lawrence, KS

Published by CMP Books
an imprint of CMP Media LLC
Main office: 600 Harrison Street, San Francisco, CA 94107 USA
Tel: 415-947-6615; fax: 415-947-6015
Editorial office: 1601 West 23rd Street, Suite 200, Lawrence, KS 66046 USA
www.cmpbooks.com
email: books@cmp.com

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where CMP Books is aware of a trademark claim, the product name appears in initial capital letters, in all capital letters, or in accordance with the vendor's capitalization preference. Readers should contact the appropriate companies for more complete information on trademarks and trademark registrations. All trademarks and registered trademarks in this book are the property of their respective holders.

Copyright © 2003 by Wind River Systems, Inc., except where noted otherwise. Published by CMP Books, CMP Media LLC. All rights reserved. Printed in the United States of America. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

The programs in this book are presented for instructional value. The programs have been carefully tested, but are not guaranteed for any particular purpose. The publisher does not offer any warranties and does not guarantee the accuracy, adequacy, or completeness of any information herein and is not responsible for any errors or omissions. The publisher assumes no liability for damages resulting from the use of the information in this book or for any infringement of the intellectual property rights of third parties that would result from the use of this information.

Technical editors:	Robert Ward and Marc Briand
Copyeditor:	Catherine Janzen
Layout design & production:	Madeleine Reardon Dimond and Michelle O'Neal
Managing editor:	Michelle O'Neal
Cover art design:	Damien Castaneda

Distributed to the book trade in the U.S. by:
Publishers Group West
1700 Fourth Street
Berkeley, CA 94710
1-800-788-3123

Distributed in Canada by:
Jaguar Book Group
100 Armstrong Avenue
Georgetown, Ontario M6K 3E7 Canada
905-877-4483

For individual orders and for information on special discounts for quantity orders, please contact:
CMP Books Distribution Center, 6600 Silacci Way, Gilroy, CA 95020
Tel: 1-800-500-6875 or 408-848-3854; fax: 408-848-5784
email: cmp@rushorder.com; Web: www.cmpbooks.com

Library of Congress Cataloging-in-Publication Data
Li, Qing, 1971-

Real-time concepts for embedded systems / Qing Li ; with Caroline Yao.
p. cm.

Includes bibliographical references and index.

ISBN 1-57820-124-1 (alk. paper)

1. Embedded computer systems. 2. Real-time programming. I. Yao, Caroline. II. Title.
Tk7895.E42L494 2003
004'.33—dc21

2003008483

Printed in the United States of America
03 04 05 06 07 5 4 3 2 1

ISBN: 1-57820-124-1

CMPBooks

To my wife, Huaying, and my daughter, Jane, for their love, understanding, and support.

*To my parents, Dr. Y. H. and Dr. N. H. Li, and my brother, Dr. Yang Li,
for being the exemplification of academic excellence.*

"The book has a very good balance between the theory and practice of real-time embedded system designs. Most other books on real-time systems concentrate on theory without practical experience or have a focus on design examples without the theoretical background."

—Jun-ichiro Itojun Hagino, Ph.D., Research Laboratory, Internet Initiative Japan Inc., IETF IPv6 Operations Working Group (v6ops) co-chair

"A clear, concise, and comprehensive introduction to embedded and pervasive computing. This book should be required reading for any engineer or product manager entering or looking to grow in the embedded marketplace. Beginning with basic concepts, Qing Li provides exceptional detail taking the new embedded reader to whatever level of granularity they require or would find interesting. Qing Li introduces the concept of embedded designs and clearly illustrates the differences between non-real-time and soft and hard real-time requirements. Taking the RTOS concept from its component structure (e.g., scheduler, objects, and services), he delves into great detail with each sub-component. His sections on I/O, exception handling, and protocol stacks create a comprehensive presentation for the embedded 'newbee' as well as the intermediate-level embedded developer."

—Jerry Krasner, Ph.D., MBA., vice-president, industry analyst, and consultant, Embedded Market Forecasters

"This book has comprehensive, practical, updated, and easy-to-understand content about real-time embedded computer systems. I strongly believe that the knowledge, theory, and concepts explained in this book are quite important not only for the engineers who design the real-time systems and students in computer science and robotics departments, but also for every user of real-time embedded systems—especially a communication network operator, system administrator, or machinery engineer."

—Dr. Shin Miyakawa, senior research manager, IPv6 Group, Innovative IP Architecture Center, NTT Communications Corporation

TABLE OF CONTENTS

FOREWORD	IX
-----------------------	-----------

PREFACE	XI
----------------------	-----------

Audience for this Book	xii
Acknowledgments	xii

CHAPTER 1 INTRODUCTION	1
-------------------------------------	----------

1.1 Real Life Examples of Embedded Systems	2
1.2 Real-Time Embedded Systems	10
1.3 The Future of Embedded Systems	16
1.4 Points to Remember	17

CHAPTER 2 BASICS OF DEVELOPING FOR EMBEDDED SYSTEMS . . .	19
--	-----------

2.1 Introduction	19
2.2 Overview of Linkers and the Linking Process	20
2.3 Executable and Linking Format	23
2.4 Mapping Executable Images into Target Embedded Systems	27
2.5 Points to Remember	34

CHAPTER 3 EMBEDDED SYSTEM INITIALIZATION	35
---	-----------

3.1 Introduction	35
3.2 Target System Tools and Image Transfer	36
3.3 Target Boot Scenarios	39

3.4 Target System Software Initialization Sequence	46
3.5 On-Chip Debugging	51
3.6 Points to Remember	52
CHAPTER 4 INTRODUCTION TO REAL-TIME OPERATING SYSTEMS . . .	53
4.1 Introduction	53
4.2 A Brief History of Operating Systems	54
4.3 Defining an RTOS	55
4.4 The Scheduler	57
4.5 Objects	61
4.6 Services	62
4.7 Key Characteristics of an RTOS	62
4.8 Points to Remember	64
CHAPTER 5 TASKS	65
5.1 Introduction	65
5.2 Defining a Task	65
5.3 Task States and Scheduling	67
5.4 Typical Task Operations	72
5.5 Typical Task Structure	76
5.6 Synchronization, Communication, and Concurrency	77
5.7 Points to Remember	77
CHAPTER 6 SEMAPHORES	79
6.1 Introduction	79
6.2 Defining Semaphores	79
6.3 Typical Semaphore Operations	84
6.4 Typical Semaphore Use	87
6.5 Points to Remember	95
CHAPTER 7 MESSAGE QUEUES	97
7.1 Introduction	97
7.2 Defining Message Queues	97
7.3 Message Queue States	99
7.4 Message Queue Content	100
7.5 Message Queue Storage	101
7.6 Typical Message Queue Operations	101
7.7 Typical Message Queue Use	105
7.8 Points to Remember	110

CHAPTER 8 OTHER KERNEL OBJECTS	111
8.1 Introduction.	111
8.2 Pipes	111
8.3 Event Registers	118
8.4 Signals	121
8.5 Condition Variables.	126
8.6 Points to Remember.	130
 CHAPTER 9 OTHER RTOS SERVICES	 133
9.1 Introduction.	133
9.2 Other Building Blocks	133
9.3 Component Configuration.	139
9.4 Points to Remember.	141
 CHAPTER 10 EXCEPTIONS AND INTERRUPTS	 143
10.1 Introduction.	143
10.2 What are Exceptions and Interrupts?	144
10.3 Applications of Exceptions and Interrupts	145
10.4 A Closer Look at Exceptions and Interrupts	146
10.5 Processing General Exceptions	150
10.6 The Nature of Spurious Interrupts.	163
10.7 Points to Remember.	165
 CHAPTER 11 TIMER AND TIMER SERVICES	 167
11.1 Introduction.	167
11.2 Real-Time Clocks and System Clocks	168
11.3 Programmable Interval Timers.	169
11.4 Timer Interrupt Service Routines	171
11.5 A Model for Implementing the Soft-Timer Handling Facility.	171
11.6 Timing Wheels.	176
11.7 Soft Timers and Timer Related Operations	182
11.8 Points to Remember.	185
 CHAPTER 12 I/O SUBSYSTEM	 187
12.1 Introduction.	187
12.2 Basic I/O Concepts.	188
12.3 The I/O Subsystem.	192
12.4 Points to Remember.	197

CHAPTER 13	MEMORY MANAGEMENT	199
13.1	Introduction	199
13.2	Dynamic Memory Allocation in Embedded Systems	200
13.3	Fixed-Size Memory Management in Embedded Systems	208
13.4	Blocking vs. Non-Blocking Memory Functions	209
13.5	Hardware Memory Management Units	211
13.6	Points to Remember	212
CHAPTER 14	MODULARIZING AN APPLICATION FOR CONCURRENCY	213
14.1	Introduction	213
14.2	An Outside-In Approach to Decomposing Applications	214
14.3	Guidelines and Recommendations for Identifying Concurrency	217
14.4	Schedulability Analysis—Rate Monotonic Analysis	225
14.5	Points to Remember	229
CHAPTER 15	SYNCHRONIZATION AND COMMUNICATION	231
15.1	Introduction	231
15.2	Synchronization	231
15.3	Communication	236
15.4	Resource Synchronization Methods	238
15.5	Critical Section Revisited	240
15.6	Common Practical Design Patterns	241
15.7	Specific Solution Design Patterns	247
15.8	Points to Remember	258
CHAPTER 16	COMMON DESIGN PROBLEMS	259
16.1	Introduction	259
16.2	Resource Classification	260
16.3	Deadlocks	260
16.4	Priority Inversion	273
16.5	Points to Remember	280
APPENDIX A	REFERENCES	281
ABOUT THE AUTHORS		285
INDEX		287

FOREWORD

We live in a world today in which software plays a critical part. The most critical software is not running on large systems and PCs. Rather, it runs inside the infrastructure and in the devices that we use every day. Our transportation, communications, and energy systems won't work if the embedded software contained in our cars, phones, routers and power plants crashes.

The design of this invisible, embedded software is crucial to all of us. Yet, there has been a real shortage of good information as to effective design and implementation practices specific to this very different world. Make no mistake, it is indeed different and often more difficult to design embedded software than more traditional programs. Time, and the interaction of multiple tasks in real-time, must be managed. Seemingly esoteric concepts, such as priority inversion, can become concrete in a hurry when they bring a device to its knees. Efficiency—a small memory footprint and the ability to run on lower cost hardware—become key design considerations because they directly affect cost, power usage, size, and battery life. Of course, reliability is paramount when so much is at stake—company and product reputations, critical infrastructure functions, and, sometimes, even lives.

Mr. Li has done a marvelous job of pulling together the relevant information. He lays out the issues, the decision and design process, and the available tools and methods. The latter part of the book provides valuable insights and practical experiences in understanding application development, common design problems, and solutions. The book will be helpful to anyone embarking on an embedded design project, but will be of particular help to engineers who are experienced in software development but not yet in real-time and embedded software development. It is also a wonderful text or reference volume for academic use.

The quality of the pervasive, invisible software surrounding us will determine much about the world being created today. This book will have a positive effect on that quality and is a welcome addition to the engineering bookshelf.

Jerry Fiddler
Chairman and Co-Founder, Wind River

In this chapter...

- *Real Life Examples of Embedded Systems* 2
- *Real-Time Embedded Systems* 10
- *The Future of Embedded Systems*... 16
- *Points to Remember*.....17

CHAPTER 1

INTRODUCTION

In ways virtually unimaginable just a few decades ago, embedded systems are reshaping the way people live, work, and play. Embedded systems come in an endless variety of types, each exhibiting unique characteristics. For example, most vehicles driven today embed intelligent computer chips that perform value-added tasks, which make the vehicles easier, cleaner, and more fun to drive. Telephone systems rely on multiple integrated hardware and software systems to connect people around the world. Even private homes are being filled with intelligent appliances and integrated systems built around embedded systems, which facilitate and enhance everyday life.

Often referred to as *pervasive* or *ubiquitous* computers, embedded systems represent a class of dedicated computer systems designed for specific purposes. Many of these embedded systems are reliable and predictable. The devices that embed them are convenient, user-friendly, and dependable.

One special class of embedded systems is distinguished from the rest by its requirement to respond to external events in real time. This category is classified as the *real-time embedded system*.

As an introduction to embedded systems and real-time embedded systems, this chapter focuses on:

- examples of embedded systems,
- defining embedded systems,
- defining embedded systems with real-time behavior, and
- current trends in embedded systems.

3. creating necessary stacks for RTOS
4. initializing additional RTOS extensions, such as:
 - TCP/IP stack
 - file systems
5. starting the RTOS and its initial tasks

The components of an RTOS (for example, tasks, semaphores, and message queues) are discussed in more detail in later chapters of this book. For now, note that the RTOS abstracts the application code from the hardware and provides software objects and services that facilitate embedded-systems application development.

3.4.3 Application Software Initialization

After the RTOS is initialized and running with the required components, control is transferred to a user-defined application. This transfer takes place when the RTOS code calls a predefined function (that is RTOS dependent) which is implemented by the user-defined application. At this point, the RTOS services are available. This application also goes through initialization, during which all necessary objects, services, data structures, variables, and other constructs are declared and implemented. For a simple, user application such as the “hello world” application, all the work can be done in this function. This user-defined application (maybe the “hello world” application) might finally produce its impressive output. On the other hand, for a complex application, it will create task or tasks to perform the work. These application-created tasks will execute once the kernel scheduler runs. The kernel scheduler runs when this control-transfer function exits.

3.5 On-Chip Debugging

Many silicon vendors recognize the need for built-in microprocessor debugging, called on-chip debugging (OCD). BDM and JTAG are two types of OCD solutions that allow direct access and control over the microprocessor and system resources without needing software debug agents on the target or expensive in-circuit emulators. As shown in Figure 3.1, the embedded processor with OCD capability provides an external interface. The developer can use the external interface to download code, read or write processor registers, modify system memory, and command the processor to execute one instruction and halt, thus facilitating single-step debugging. Depending on the selected processor, it might be possible to disable the on-chip peripherals while OCD is in effect. It might also be possible to gain a near real-time view of the executing system state. OCD is used to solve the chicken-and-egg problem often encountered at the beginning development stage—if the monitor is the tool for debugging a running program, what debugs the monitor while it’s developed? The powerful debug capabilities offered by the OCD combined with the quick turnaround time required to set up the connection means that software engineers find OCD solutions invaluable when writing hardware initialization code, low-level drivers, and even applications.

JTAG stands for Joint Test Action Group, which was founded by electronics manufacturers to develop a new and cost-effective test solution. The result, produced by the JTAG consortium, is sanctioned by the IEEE1149.1 standard.

BDM stands for background debug mode. It refers to the microprocessor debug interface introduced by Motorola and found on its processor chips. The term also describes the non-intrusive nature (on the executing system) of the debug method provided by the OCD solutions.

An OCD solution is comprised of both hardware and software. Special hardware devices, called personality modules, are built for the specific processor type and are required to connect between the OCD interface on the target system and the host development system. The interface on the target system is usually an 8- or 10-pin connector. The host side of the connection can be the parallel port, the serial port, or the network interface. The OCD-aware host debugger displays system state information, such as the contents of the processor registers, the system memory dump, and the current executing instruction. The host debugger provides the interface between the embedded software developer and the target processor and its resources.

3.6 Points to Remember

Some points to remember include the following:

- Developers have many choices for downloading an executable image to a target system. They can use target-monitor-based, debug-agent-based, or hardware-assisted connections.
- The boot ROM can contain a boot image, loader image, monitor image, debug agent, or even executable image.
- Hardware-assisted connections are ideal, both when first initializing a physical target system as well as later, for programming the final executable image into ROM or flash memory.
- Some of the different ways to boot a target system include running an image out of ROM, running an image out of RAM after copying it from ROM, and running an image out of RAM after downloading it from a host.
- A system typically undergoes three distinct initialization stages: hardware initialization, OS initialization (RTOS), and application initialization.
- After the target system is initialized, application developers can use this platform to download, test, and debug applications that use an underlying RTOS.

MICROPROCESSORS / OPERATING SYSTEMS

A clear, concise, and comprehensive introduction to embedded and pervasive computing. This book should be required reading for any engineer or product manager ...
—**Jerry Krasner, Ph.D., MBA.**, vice-president, industry analyst, and consultant, Embedded Market Forecasters

... a very good balance between the theory and practice of real-time embedded system designs.
—**Jun-ichiro Itojun Hagino, Ph.D.**, Research Laboratory, Internet Initiative Japan Inc., IETF IPv6 Operations Working Group (v6ops) co-chair

Master the fundamental concepts of real-time embedded system programming and jumpstart your embedded projects with effective design and implementation practices. This book bridges the gap between higher-level abstract modeling concepts and the lower-level programming aspects of embedded systems development. You gain a solid understanding of real-time embedded systems with detailed practical examples and industry wisdom on key concepts, design processes, and the available tools and methods.

Delve into the details of real-time programming so you can develop a working knowledge of the common design patterns and program structures of real-time operating systems (RTOS). The objects and services that are a part of most RTOS kernels are described and real-time system design is explored in detail. You learn how to decompose an application into units and how to combine these units with other objects and services to create standard building blocks. A rich set of ready-to-use, embedded design "building blocks" is also supplied to accelerate your development efforts and increase your productivity.

Experienced developers new to embedded systems and engineering or computer science students will both appreciate the careful balance between theory, illustrations, and practical discussions. Hard-won insights and experiences shed new light on application development, common design problems, and solutions in the embedded space. Technical managers active in software design reviews of real-time embedded systems will find this a valuable reference to the design and implementation phases.

CMP Books is an imprint of CMP Media LLC
Other fine publications in the CMP family include:

Embedded Systems
PROGRAMMING

C/C++ Users Journal™
Advanced Solutions for C/C++ Programmers



CMP

United Business Media

CMP Books

INTEL EXHIBIT

1010 Page 149

IPR2023-00787

\$10.99
867003
AO -
273 - Y
No Exchange Books Computers
value village
Communications Corporation

Qing Li is a senior architect at Wind River Systems, Inc., and the lead architect of the company's embedded IPv6 products. Qing holds four patents pending in the embedded kernel and networking protocol design areas. His 12+ years in engineering include expertise as a principal engineer designing and developing protocol stacks and embedded applications for the telecommunications and networks arena. Qing was one of a four-member Silicon Valley startup that designed and developed proprietary algorithms and applications for embedded biometric devices in the security industry.

Caroline Yao has more than 15 years of high tech experience ranging from development, project and product management, product marketing, business development, and strategic alliances. She is co-inventor of a pending patent and recently served as the director of partner solutions for Wind River Systems, Inc.

In Canada

\$44.95 \$77.95 UK £34.95
ISBN 1-57820-124-1



781578 201242