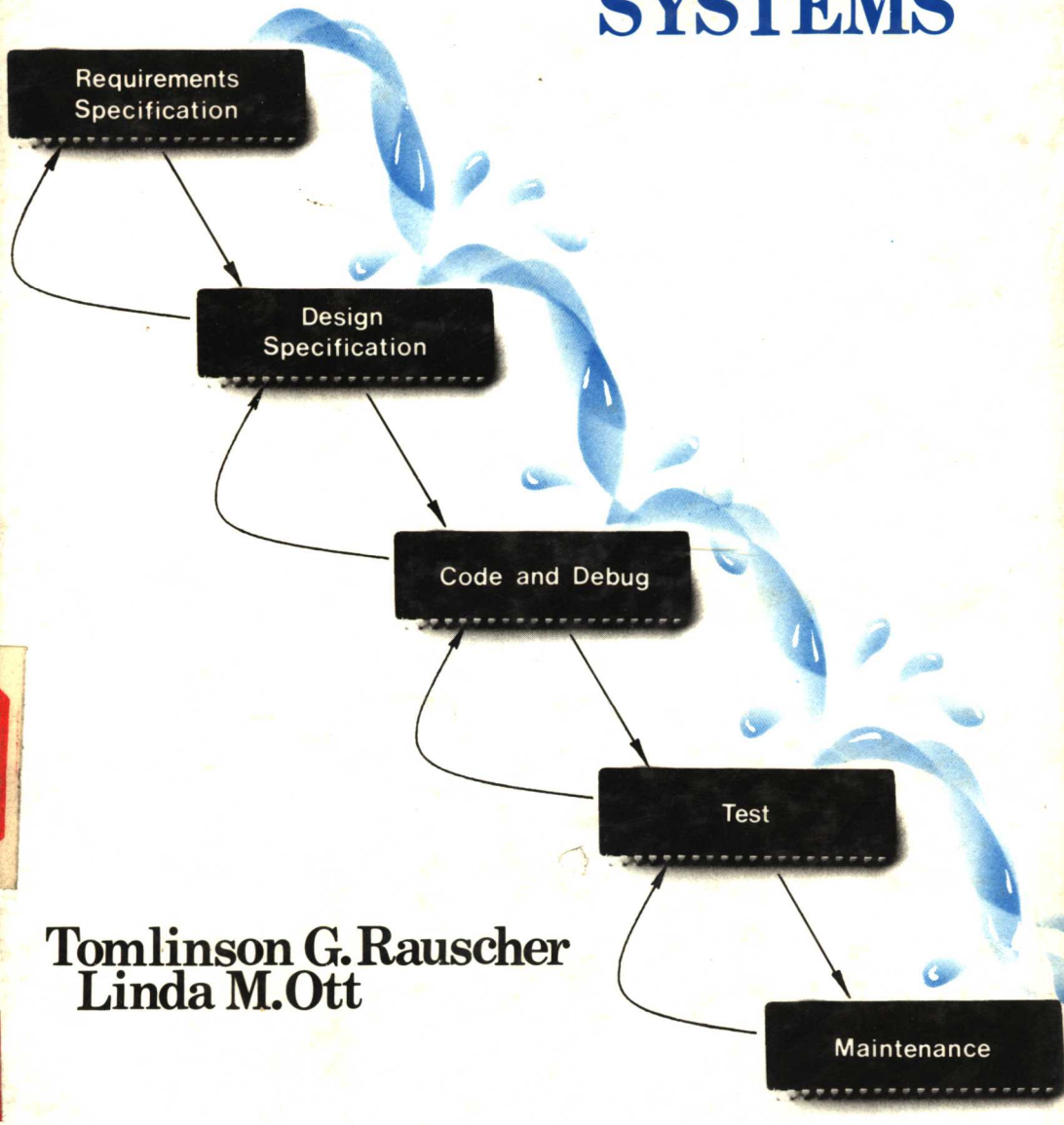
SOFTWARE DEVELOPMENT and MANAGEMENT for MICROPROCESSOR-BASED SYSTEMS



Tomlinson G. Rauscher
Linda M. Ott

Software Development and Management For Microprocessor-Based Systems

Tomlinson G. Rauscher
Xerox Corporation
Rochester, New York

Linda M. Ott
Michigan Technological University
Houghton, Michigan

Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

RAUSCHER, TOMLINSON G.
Software development and management for microprocessor-based systems.

Includes index.
1. Computer software—Development.
2. Microprocessors—Programming. I. Ott, Linda M., 1950-19 II. Title.
QA76.76.D47R38 1987 005.26 86-30477
ISBN 0-13-822933-3

Cover design: *Photo Plus Art*
Manufacturing buyer: *Gordon Osbourne*

To David, Tasha, April, and Nathan

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-822933-3 025

PRENTICE-HALL INTERNATIONAL (UK) LIMITED, *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL CANADA INC., *Toronto*
PRENTICE-HALL HISPANOAMERICANA, S.A., *Mexico*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*
EDITORA PRENTICE-HALL DO BRAZIL, LTDA., *Rio de Janeiro*

PREFACE

The purpose of our book is to describe a set of techniques for developing software for large microprocessor-based systems, and for managing such development activities. The book is practical in its orientation; it describes a particular set of techniques that have demonstrably improved the effectiveness of microprocessor development and management.

The topics of microprocessor software development and management have grown in interest for several reasons:

- Microprocessors are being used in a large and increasing number of applications to provide greater functionality at a lower cost: consumer products, appliances, automobiles, industrial control, personal computers, office equipment, and so forth.
- The software systems for microprocessors in such applications are increasing in size, complexity, and sophistication, thus creating a demand for more effective software development and management.
- Many software developers of microprocessor systems have little training in developing the large software systems that typify modern microprocessor-based products.
- Many managers of microprocessor-based system products have little knowledge or experience in managing large software projects.
- The software development cost for microprocessor-based systems is large (often tens or hundreds of man-years), typically larger than the cost of developing the electronic hardware on which the software runs, and almost always larger than it should be.

Our book is intended for two principal audiences:

- Students at the upper class or graduate level who study microprocessor software in electrical engineering, computer

engineering, computer science, or related programs of study. Such students seldom have the opportunity to work on large projects, and thus lack the knowledge of a practical development environment to which many of them will graduate.

- Industrial practitioners who want to develop and manage significant microprocessor software activities. These professionals may have experience writing small programs on microprocessors or writing programs for large data processing systems. This basis coupled with the information presented in our book provides strong capabilities for developing and managing microprocessor-based systems.

We have used material in this book with hundreds of people in these two groups, effecting substantial improvements in personal and corporate capabilities.

We approach the subject of microprocessor software development assuming that the reader has some knowledge of microprocessors and programming. Readers should have a working knowledge of microprocessors, have written small assembler language programs for a particular microprocessor, and have done small programming projects with a higher level language to understand the fundamentals of programming concepts. There are a number of books that address such topics; our book is a second book addressing advanced topics for large microprocessor-based systems. Using introductory knowledge as a starting point, we describe in Chapters 1 through 10 specific techniques for engineering significant microprocessor software systems. The techniques cover the phases of software engineering – requirements specification, design specification, coding and debugging, testing, and maintenance – concentrating on microprocessor applications with emphasis on real-time aspects. To illustrate techniques, the examples used are short and address typical microprocessor applications, so that they can be grasped quickly. The book describes specific techniques that address the entire software development life cycle for microprocessor applications, in contrast to surveying a large number of techniques concentrating on one phase of software development, or orienting the techniques toward business data processing applications.

In a similar manner, we approach the subject of microprocessor software management assuming that the reader has some knowledge of management activities such as general project planning, PERT methods, task assignments, and delegation of authority. In the second part of the book, Chapters 11 through 19, we address software management for microprocessor-based systems by extending relevant aspects of traditional managerial concepts such as planning, staffing, organizing, and reviewing

to facilitate developing software for microprocessor-based products. Facilities required for developing microprocessor software are described. The special problems of managing embedded microprocessor software are examined in detail. Recommendations are given for developing software technology from both the practical aspects of architecture, tools, and techniques, and the personal aspects of professionals who constitute a staff. The book concludes with software management lessons that address problems typically encountered when microprocessors play a significant role in the product development repertoire in a company. The software development of a real microprocessor-based product in a large company is reviewed and critiqued.

In writing the book we have incorporated several concepts to increase its utility to students and professionals. The book uses a practical technique orientation to software development and management for microprocessors, based on real product development experiences; it is not a survey of theoretical concepts. The book concentrates on microprocessor examples; it is not a general data processing book. It addresses life cycle aspects of software development and management, not just the details of a methodology to support one phase of product development. It describes techniques that are integrated with one another throughout the development process; it is not a collection of independent articles addressing specific issues within the development process. The book is organized to present material in the order engineers and managers will be comfortable reading, building on coding and debugging knowledge and using this as motivation for design, requirements, testing, and maintenance. As the coding and debugging phase of software development is the phase with which engineers and managers are the most familiar, we present it first, even though it is neither the first phase nor the most important phase in microprocessor software development. With a thorough understanding of coding and debugging, readers will better understand the need for the design specification phase and the requirements specification phase, the most important phase in software development. The book also includes a chapter that summarizes techniques for microprocessor development in chronological order (Chapter 10) and a chapter that summarizes specific management directions (Chapter 19).

Many people assisted us in the preparation of the book. Steve Palumbo, Dan Auman, and John Kemp of Xerox Corporation helped with the preparation of several figures on software design. William Stumbo of Xerox assisted in the preparation of the index. Andrea and Roger Hopkins of Publishease prepared many of the figures. Jeanne Paige of Xerox Corporation typed several drafts of the book over an extended period; her

patient competence made preparation of the manuscript enjoyable. Marie Vasapolli and Nancy Taylor of Xerox made noteworthy contributions as well. Finally we would like to thank our colleagues at Xerox Corporation and Michigan Technological University for their encouragement and support during this work. Their reviews and comments improved not only the presentation but the substance of the book as well. Michael Nekora of Xerox made several insightful comments, which we hope we have conveyed to the reader.

CONTENTS

1

INTRODUCTION TO MICROPROCESSOR SOFTWARE SYSTEMS DEVELOPMENT

1.1 The Role of Software in the Microprocessor Revolution

The decade of the seventies will surely be remembered for witnessing the commercial introduction and utilization of microprocessors. Not only did their deployment revolutionize the use of electronics in a multitude of applications, but their continued improvement demonstrated incredible technological maturation. From the simple characteristics of the first four-bit microprocessors to the sophisticated features of modern sixteen-bit and thirty-two-bit microcomputers, the capabilities provided by these microelectronic marvels have increased dramatically. While capabilities have increased, the cost of employing these "computers-on-a-chip" has, surprisingly, decreased when examined on a cost-per-function basis. The increasing availability of capable, inexpensive microprocessors has facilitated the development of increasingly sophisticated systems, ranging from simple calculators to real-time systems that simultaneously support several activities and input/output devices.

The ability of microprocessors to support complex applications has been made possible not only by improved hardware capabilities such as greater functionality of instructions, larger on-chip memories, and faster instruction execution times, but also by larger, more powerful software systems. Whereas the size of early microprocessor programs ranged only up to a few thousand bytes, modern systems frequently reach fifty thousand

bytes and more. The effort involved in developing software systems of such capabilities requires technological knowledge, engineering discipline, and management direction. The development of these capabilities often proceeds at a slower rate than that of the microprocessor hardware technology. When this is coupled with the realization that developing large software systems for microprocessors is exponentially more complex than developing small systems (with which people typically experience some success in initial attempts), it is easy to see why software development has been criticized for products that are expensive, late, and unreliable.

Microprocessor software systems development need not deserve this reputation. With the appropriate application of engineering and management tools and techniques, microprocessor software systems development can be a rewarding and successful enterprise. The purpose of this book is to assist in the accomplishment of this goal.

It is appropriate to examine the historical background which has led to the poor opinion in which software is often held today. Early microprocessor software development activities tended to emphasize the differences from traditional software development, citing justifications such as

- Close functional relationship to hardware being replaced
- Small program sizes
- Use of low-level languages, particularly machine and assembler languages

The current trend in microprocessor systems deviates from these viewpoints:

- Microprocessors are being used where flexibility and expandability are required; specially designed VLSI chips and "programmable logic arrays" are now being used to replace hardware of moderate complexity where microprocessors were formerly used.
- As mentioned earlier, large programs are as much the rule as the exception.
- Higher level languages can, should, and are being used in most microprocessor applications.

It is useful to note, however, that microprocessor software often exhibits differences from the typical data processing programs, which constitute a significant portion of traditional software systems. These differences include the following:

- Microprocessor software frequently represents only a small part of a system composed of several electrical, mechanical, and other components. The end user need not (and often is not) aware that a microprocessor is integrated in the system.
- Microprocessor manufacturers seldom supply extensive support system software (although this trend is changing), and, as a result, developers frequently must design and implement all the microprocessor software for their system. This ranges from low-level operating system tasks such as interrupt and I/O handling to high-level application specific tasks.
- Microprocessors often support real-time systems, wherein immediate response to asynchronous external events characterize normal requirements.
- The principal memory for program storage in many microprocessor systems is read-only memory (ROM), due to its cost savings over other memory types, such as electrically programmable ROM (EPROM) and read/write random access memory (RAM). The cost per system aspect is especially important when considering that microprocessor products often have sales or usage volumes in the tens and hundreds of thousands (such as calculators, automobiles, cash registers, and games).
- Another aspect of products with imbedded microprocessors is that companies often produce product families in which microprocessor use may vary in only minor ways among products in the same family.

These variations on traditional software systems result in the use of tools and techniques that are peculiar to microprocessor software development. Much of the development activity for microprocessor software, however, adapts techniques used in traditional software development.

It has been interesting to follow the technology for developing software for microprocessor systems because history has, to a large extent, repeated itself. The use of machine language, assemblers, macroassemblers, compilers, and debuggers bears a strong resemblance to their development for larger computers two decades earlier. The reasons for this are technological and organizational:

- The first developers of microprocessors were technologists whose principal expertise was in microelectronics, not software.