REAL-TIME MICROCOMPUTER SYSTEM DESIGN

An Introduction

Peter D. Lawrence
Konrad Mauch

REAL-TIME MICROCOMPUTER SYSTEM DESIGN

An Introduction

Peter D. Lawrence
Konrad Mauch

Department of Electrical Engineering
University of British Columbia
Canada

ık Company

New York St. Louis San Francisco Auckland Bogotá Hamburg
Johannesburg London Madrid Mexico Milan Montreal New Delhi
Panama Paris São Paulo Singapore Sydney Tokyo Toronto

This book was set in Times Roman.
The editor was Sanjeev Rao;
the cover was designed by Rafael Hernandez;
the production supervisor was Phil Galea.
Project supervision was done by The Total Book.
R.R. Donnelley & Sons Company was printer and binder.

REAL-TIME MICROCOMPUTER SYSTEM DESIGN
An Introduction

1234567890 DOCDOC 8943210987

ISBN 0-07-036731-0

Library of Congress Cataloging-in-Publication Data

Lawrence, Peter D. (Peter Donald)
Real-time microcomputer system design.

(McGraw-Hill series in electrical engineering)
Includes bibliographies and index.
1. Automatic control. 2. Microcomputers.
3. Real-time data processing. 4. System design.
I. Mauch, Konrad. II. Title. III. Series.
TJ223.M53L39 1987 629.8'95 86-15361
ISBN 0-07-036731-0
ISBN 0-07-036732-9 (solutions manual)

PREFACE

Microcomputer-based real-time systems are used not only by electrical and computer engineers but also by engineers in many other disciplines. Mechanical engineers, for example, design robots and other automated machinery controlled by microcomputers. Plants designed by chemical engineers and mineral processing engineers are controlled and monitored by microcomputers. Civil engineers gather hydrological and other environmental data using microcomputer-based data-acquisition systems. Scientists in both the physical and biological sciences are also using microcomputer-based real-time systems to gather and record data from experiments and to control experimental apparatus.

USERS OF THIS BOOK

This book was primarily developed for engineering students in a broad range of disciplines. Practicing engineers and scientists working in the physical, biological, and applied sciences who are seeking a self-contained introduction to the design of real-time microcomputer systems should also find this book useful.

Even though this book is not aimed specifically at electrical engineering students, we believe that it has a place in the electrical engineering curriculum. Used either as a supplement or as the primary text, it provides a high-level perspective on the design process. References after every chapter are provided for readers who may want to pursue a topic in greater depth.

APPROACH

The objective of the book is to present a systems design methodology that involves all the components of a real-time computing system, including sensors, actuators, conditioning electronics, and interfaces as well as the CPU. To

reflect a common practice in industry and to a great extent in scientific research, we have chosen to concentrate on the development of *board-based* real-time microcomputer systems.

We also emphasize the use of *high-level programming languages* to produce software for real-time systems. It is our experience that high-level programming languages can be used effectively in all but the most time-critical real-time systems. When assembly-language programming is required, it can often be reduced to a few short subroutines called by the high-level-language program.

Because of our emphasis on board-level hardware design and the use of high-level programming languages, and because of the nature of the audience that we are writing for, this book differs in content and approach from the standard textbooks on microcomputers used by electrical engineering students. Our approach is mostly top-down, starting with a look at the design process for the real-time system as a whole and then discussing the design of the various components of the system.

We do not discuss the design of microcomputer systems at the chip level in any detail. Our discussion of assembly-language programming is brief and oriented toward how to incorporate short assembly-language modules into a program written in a high-level language and write the necessary assembly-language components of an interrupt handler. However, we include material on important topics such as microcomputer-development systems, real-time operating systems, and transducers and actuators, which are usually not covered in the standard introductory textbooks on microcomputers.

ORGANIZATION

We use this book as a text for an introductory course on microcomputers. Our students are in the senior year of an engineering discipline other than electrical engineering. The majority are mechanical engineering students, but we also have bioresource, chemical, civil, engineering physics, geological, metallurgical, mining and mineral process engineering students. All students have had at least one high-level language programming course (either FORTRAN or Pascal) and an introductory electrical circuits course. Many of the students have also had an introductory electronics course which includes some exposure to digital logic. The material on number systems and digital logic elements in the first two appendixes is assigned to the students for self-study at the beginning of the course to ensure that they all have the required background information.

While we are covering high-level design and software design in lectures, our students are programming C language applications (using an auxiliary C language text) in the laboratory. We then cover assembly language and finally the hardware topics, calling this a top-down software-before hardware organization.

Our course is heavily laboratory-oriented, which is important for any course based on this text, since the general concepts introduced must be reinforced by practical experience with a real microcomputer system. Our microcomputer laboratory consists of a multiuser microcomputer development system (Tektronix 8560), which is linked to Motorola MC6800-based laboratory microcomputers. Each laboratory microcomputer is equipped with a standard backplane bus (STD bus) which contains boards that interface the computer to the apparatus which it monitors and controls. The use of a board-based microcomputer system in the laboratory reinforces the board-based hardware design approach of the text.

Our students develop programs for the laboratory computers on the microcomputer development system. The majority of the programs are written entirely in a high-level programming language (the language C).

The laboratory sequence starts with some programming exercises to introduce the student to the UNIX operating system used on the development system and to programming in C. Then the students develop a program which makes use of pumps, liquid-level sensors, and agitators in the laboratory apparatus to emulate a washing machine. This procedure introduces students to the use of microcomputers in simple sequential control (i.e., programmable controller applications) and shows them very early in the course how a computer can control physical systems. The next sequence of laboratory exercises involves the measurement and control of water temperature, which introduces students to analog interfaces, sensor linearizing, and control via computer. Students then develop a program for a real-time clock. This exercise involves the use of interrupts and the development of a short assembly-language interrupt service routine linked with the main C program. The last laboratory in the sequence varies, depending on the time available and the interests of the instructors and students, but it has included the control of a small robot arm.

OTHER COURSE ORGANIZATIONS

A second approach is to have a bottom-up software-before-hardware organization in which assembly language would be covered first. A suggested chapter order for this approach is: introduction (1), computer structure (2), assembly language (7), high-level languages (5), high-level software example (6), high-level design (3, 4), hardware topics (8 to 14). In the laboratory, the students could do assembly language first, high-level language next, and then programs interacting with interfaces, sensors, actuators, etc.

A third approach is to have a bottom-up hardware-before-software order. In that case the chapters could be ordered as hardware (2, 8, 9, 10 to 13), assembly language (7), high-level language (5, 6), high-level design (3, 4), system-design example (14), development systems (15), and operating systems (16). With this approach it is more difficult to have laboratories early in the

course (since software is required to interact with the hardware and the software is not covered until later in the sequence). It might be used when there are no laboratories in the first semester of a two-semester sequence.

Although our laboratory is centered on a commercial microcomputer development system, many other arrangements are possible. A mainframe computer or minicomputer could be used to develop the software for the laboratory computers. Both Tektronix and Hewlett-Packard, for example, support Pascal and C language cross-compilers and assemblers for many microprocessors. This cross-software runs on the multiuser DEC VAX computers and the HP-9000 series computers. Alternatively, a completely decentralized system could be designed using personal computers such as an IBM PC. The personal computer can be used as a software-development station, and for downloading software to a laboratory experiment station containing a bus-based system; or the personal computer can be enhanced with interface boards itself and be used as the real-time system. The essential requirements are that the development system support software development in a high-level language and that the microcomputer be readily interfaced to transducers and actuators in the lab apparatus.

The book by David Auslander and Paul Sagues entitled *Microprocessors for Measurement and Control* (Osborne—McGraw-Hill, 1981) also contains many laboratory exercises well suited for a course based on our book.

ACKNOWLEDGMENTS

We would like to thank the many engineering students who used drafts of this book over the past three years and who gave us valuable criticism, found errors, and carried us along with their interest in the technology. Jim Dukarm of RSI Robotic Systems International contributed very useful comments during the drafting of the book. There were many helpful discussions with Eric Jackson of International Submarine Engineering.

To Réal Frenette who worked out the solutions to the exercises at the end of each chapter, we would like to express our gratitude.

We are also indebted to the reviewers of this book whose many constructive suggestions we have attempted to incorporate into the text.

Finally, we would like to thank our families and friends for their constant encouragement and support.

Peter D. Lawrence
Konrad Mauch

CONTENTS