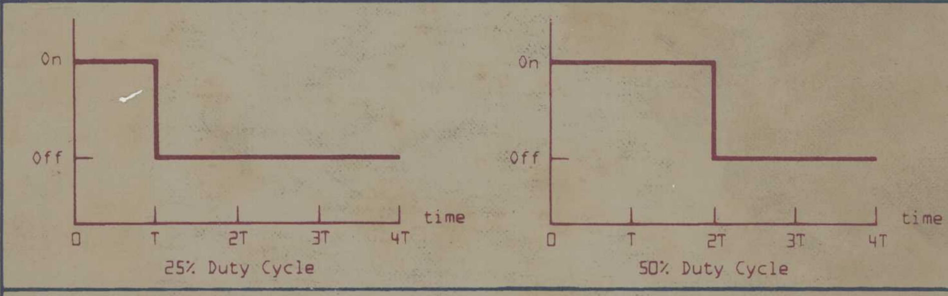
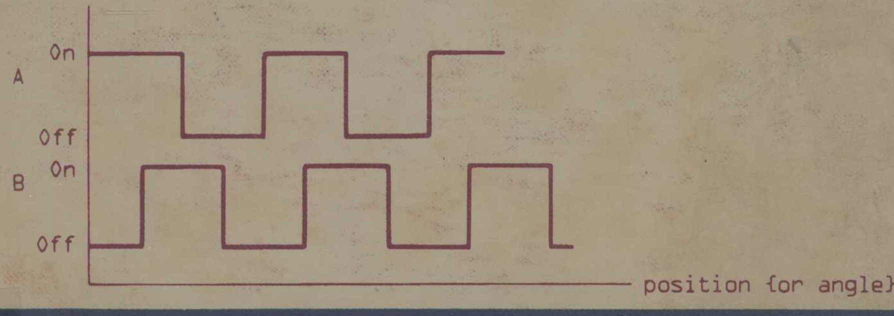
DISK INCLUDED

DAVID M. AUSLANDER
CHENG H. THAM

REAL-TIME SOFTWARE FOR CONTROL





PROGRAM EXAMPLES IN C

TP31
A932

9064405

REAL-TIME SOFTWARE FOR CONTROL: PROGRAM EXAMPLES IN C

南京工大学 图书馆藏书

David M. Auslander
University of California, Berkeley

Cheng H. Tham
Sherpa Corporation

E9064405

Prentice Hall
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Auslander, David M.
Real time software for control : program examples in C / David M. Auslander, Cheng H. Tham.
p. cm.
Includes bibliographical references.
ISBN 0-13-762824-2
1. Real-time control--Computer programs. 2. C (Computer program language) I. Tham, Cheng H. (Cheng Haam) II. Title.
TJ217.7.A87 1989
629.8'955133--dc20 89-36478
CIP

Editorial/production supervision
and interior design: *Jacqueline A. Jeglinski*
Cover design: *Ben Santora*
Manufacturing buyer: *Denise Duggan/Mary Ann Gloriande*

© 1990 by Prentice-Hall, Inc.
A division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

The publisher offers discounts on this book when ordered in bulk quantities. For more information, write:

Special Sales/College Marketing
Prentice-Hall, Inc.
College Technical and Reference Division
Englewood Cliffs, NJ 07632

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-762824-2

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*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

REAL-TIME SOFTWARE FOR CONTROL: PROGRAM EXAMPLES IN C

PREFACE

Engineers from all disciplines must be able to conceptualize, design, and prototype systems that depend on computers as operational components. The software in these computers is *real-time*, in the sense that its operation must be synchronized with events occurring in the physical system and with time.

Performance, reliability, and cost are products of the integral design of a machine and its control intelligence. Part count reductions, self-diagnostic ability, adaptation to changing environments, faster operation—these are all benefits of appropriate replacement of hardware with software. Our hypothesis is that *all* engineers need to know the fundamentals of intergrated real-time software, whether they are designing, supervising design, purchasing, or using sophisticated engineering systems.

Real-time software functions in an environment in which the various system components operate asynchronously. The events associated with changes in software state, changes in state of the physical system, and time, do not repeat in a predictable way. This environment puts a premium on good design practice, since system "bugs" cannot be reproduced at will for diagnosis as they usually can in purely numerical programs. Furthermore, the asynchronous nature of the system increases the likelihood that a system will contain very low probability bugs, bugs that don't show up in laboratory testing, but could appear in production versions of the system long after its initial release.

Our approach to real-time software emphasizes design practices that result in fewer bugs in the first place: modular programs, data hiding, mutual exclusion,

此为试读,需要完整PDF请访问: www.ertongbook.com

task isolation, simulation. The text uses a graduated approach, starting with strictly synchronous software (although the complete system remains asynchronous), adding interrupts, simple scheduling, and then event-driven scheduling.

All of the text material is illustrated with extensive examples, with complete code in C included. The text presents solutions in a language-independent form, with C-specific discussion of the examples following the text material in each chapter.

Motor-driven systems are used as the physical example throughout the text. Motors are ubiquitous in the engineering world, and they also can be small, inexpensive, and safe—ideal properties for use in a teaching laboratory.

The material in this text is the subject of a one-semester graduate course in the Mechanical Engineering Department, University of California at Berkeley. It is supplemented with a C-language tutorial. The course has no formal prerequisites, and is successfully completed by students from Mechanical Engineering as well as other engineering departments. In particular, students are not expected to have any prior C or real-time experience.

The course is heavily laboratory-based. In the early part of the course, example programs from the text are used as the basis for lab exercises. The students are asked to test, modify, or enhance these programs. This provides a functioning starting point for students, and helps to minimize the frustration commonly associated with development of real-time software from scratch. It also helps teach an incremental style of design—test, enhance the design, retest, etc.—which we feel allows students to build confidence in their ability to get a job done.

The text can be used even in situations in which an actual lab is not available, since the use of simulations is encouraged throughout. Simulations are carried all the way to real-time operation, with the physical system existing as a simulation in a separate task module.

The book is also intended for use as a professional self-study text. In that case, some prior study in C would be useful. Because the example code includes simulation-based programs, completion of the text does not require an extensive laboratory.

Real-time Software for Control: Program Examples in C

CONTENTS

此为试读，需要完整PDF请访问：www.ertongbook.com

CHAPTER 1

SYNCHRONOUS PROGRAMMING

Real-time computer systems must interact with the outside world on terms that are dictated by events taking place there. The computations that are done in response to those events must not only produce the correct results, but they must also produce those results at the right time. Unlike a real-time system, the success of a scientific or engineering computation is rarely related to when the result appears, although the user's patience and total computing expenses are related to the computation time. A further distinction in real-time computing is that the total computing environment consists of many semi-independent tasks that must be synchronized properly.

Many varieties of computers and systems qualify as "real-time." In this text, our concerns will focus on engineering systems in which there are interactions between a computer and some form of physical system. There are also often interactions with an operator. The physical system usually contains several measuring devices, which the computer must interrogate to get information, and several actuators, which receive signals from the computer to control their actions. Some systems have only one or the other of sensors or actuators, while most have both (Fig. 1.1). The computer (or computers) used can range from thumbnail size to room size (microprocessors to superminis), but the basic techniques for designing effective real-time systems are the same: careful conceptual design, systematic implementation, exhaustive validation, and thoughtful choice of software and hardware development tools. A major focus here will be on the use of high-level computing languages for implementation of real-time systems.

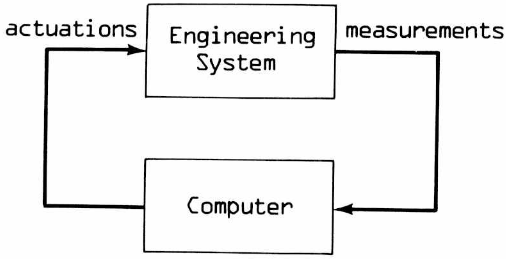


Figure 1.1

1.1 MOTOR SPEED CONTROL

We have chosen the control of electric motors as our theme. Motors are widely used and appear in so many different kinds of engineering systems that they cross virtually all disciplinary boundaries. When different methods of actuation and speed and position measurement are considered, motors also offer examples of situations that are typical of almost any real-time system. Motor systems are also easy and inexpensive to build in a laboratory, and so offer an excellent learning environment. On the other hand, the programs developed in the course of exploring the theme of motor control are generic to other control problems, and could be applied to many of them with little or no change.

A simple motor control system is shown schematically in Fig. 1.2. From the point of view of real-time system design, the simplicity of the job, even for this very simple-looking physical system, will depend on how much we demand of the computer. If the analog-to-digital (A/D) and digital-to-analog (D/A) converters can operate with little or no intervention from the computer, if the only interaction with the operator takes place at the beginning and end of an experiment, and if the algorithm chosen for computing the output signal to the power amplifier as a function of the measured motor speed depends only on the most recent measurement, then the real-time system will also be quite simple. With these restrictions, we can embark on our first example.

1.2 THE CONTROL ALGORITHM

At the heart of most real-time computation systems there are usually some key calculations. This could be a trend analysis of incoming data, spectral analysis for recognizing changes in system characteristics, generation of waveforms for system excitation, or, in this case, computation of the actuation signal on the basis of the measured motor velocity. Although these calculations are absolutely critical to proper system operation, the actual amount of program code devoted to them is usually embarrassingly small!

Control of motor speed is accomplished by increasing the voltage to the power amplifier if the speed is too low, and decreasing it if the speed is too high. A simple rule for doing this is to make the change in actuation voltage proportional to the velocity error, the difference between the actual velocity and the

此为试读，需要完整PDF请访问：www.ertongbook.com