Contents

Preface

Over the past 30 years digital control of industrial processes has changed from being the exception to the commonplace. Each succeeding year sees an increase in the range of applications and each advance in hardware design widens the potential application areas. Microprocessors are now a normal component employed in a wide range of electronic systems.

Computers in one form or another now form an integral part of most real-time control systems; such computers are generally referred to as *embedded real-time computers* and an understanding of how to design and build systems containing embedded computers is an essential requirement for a systems engineer. The knowledge required covers both hardware and software design and construction, and of the two the software engineering is the most difficult and least understood. The difficulties of specifying, designing and building real-time software have led in recent years to intensive work on development methodologies and one of the major objectives of this book is to introduce the reader to the fundamental ideas underlying these methodologies.

Traditional computing courses for engineers have typically emphasised the hardware and programming language aspects of using computers. An understanding of the basic principles of operation of computer hardware and of programming languages is of course important; however, these are details with respect to the overall system design problem. Also in the past it has frequently been assumed that real-time control software will be written in an assembly language. It is of course true that for many engineering applications use of assembly languages, and languages such as FORTH and real-time BASIC, are legitimate, economical means of producing a viable solution. However, for large-scale, complex and safety-critical systems such approaches are no longer adequate and in this book I attempt to cover some of the important and fundamental ideas underlying a software engineering approach to specifying, designing and building real-time software for computer control applications.

The book is intended for final-year undergraduate students and practising engineers, and is also suitable as a text for computer science students requiring an introduction to real-time software from an application viewpoint as opposed to an operating system viewpoint. It assumes that the reader has some familiarity with at

least one high-level programming language, with basic ideas about computer hardware and an understanding of simple feedback control concepts. It does not require familiarity with control design techniques. Examples in the text are given using Modula-2 and should be easily followed by anyone familiar with Pascal.

In Chapters 1 and 2 I introduce basic concepts relating to real-time systems and their characteristics and provide an overview of computer control applications. Chapter 3 provides a brief summary of the important hardware building blocks for computers used for control. Chapter 4 introduces some of the practical problems of implementing control algorithms. I use the simple and widely used PID (Proportional + Integral + Derivative) controller as an example and I explain its operation in detail. The section of this chapter which uses z-transform notation and deals with the implementation of controllers designed using discrete control techniques can be omitted by readers who have no background in control systems. In the final part of Chapter 4, I introduce in general terms how to deal with systems which involve more than just the control algorithm.

Chapter 5 on real-time languages and Chapter 6 on operating systems and concurrency provide basic information about two essential tools that will be required by the real-time system builder. In Chapters 7, 8 and 9 I deal with the methodologies that have been developed in recent years to help in the specification, design and construction of real-time software and real-time systems. I concentrate mainly on two methodologies, MASCOT and the Hatley & Pirbhai variant of the Yourdon method. The standard methodologies provide very little guidance for the actual implementation of the software design and in Chapter 10 I describe some of the implementation problems and possible approaches to solving them. The final chapter deals with the vital topic of building dependable software.

Many people assisted me in producing the first edition of this book: colleagues in the Department of Automatic Control and Systems Engineering at the University of Sheffield, in particular Steve White. In preparing this second edition I have received much useful comment from Les Woolliscroft and from two reviewers.

Stuart Bennett
Sheffield, UK
April 1993