Real-time Computer Control

STUART BENNETT

Real-time Computer Control

STUART BENNETT

Department of Control Engineering,
University of Sheffield, UK

PRENTICE HALL

NEW YORK • LONDON • TORONTO • SYDNEY • TOKYO

First published 1988 by
Prentice Hall International (UK) Ltd,
66 Wood Lane End, Hemel Hempstead,
Hertfordshire, HP2 4RG
A division of
Simon & Schuster International Group

Printed and bound in Great Britain at
The University Press, Cambridge

Library of Congress Cataloging-in-Publication Data

Bennett, S. (Stuart)
Real-time computer control.

(Prentice-Hall International series in systems and control engineering)
Bibliography: p.
Includes index.
1. Digital control systems. 2. Real-time data processsing. I. Title. II. Series.
TJ223.M53B46 1988 629.8'95 87-29274
ISBN 0-13-762485-9

2 3 4 5 92 91 90 89

ISBN 0-13-762485-9
ISBN 0-13-762501-4 PBK

Real-time Computer Control

Prentice Hall International Series
in Systems and Control Engineering

M.J. Grimble, Series Editor

BANKS, S.P., *Control Systems Engineering: modelling and simulation, control theory and microprocessor implementation*

BANKS, S.P., *Mathematical Theories of Nonlinear Systems*

BENNETT, S., *Real-time Computer Control: an introduction*

CEGRELL, T., *Power Systems Control*

COOK, P.A., *Nonlinear Dynamical Systems*

PATTON, R., CLARK, R.N., FRANK, P.M. (editors), *Fault Diagnosis in Dynamic Systems*

SÖDERSTRÖM, T., STOICA, P., *System Identification*

Preface

Over the past 25 years the application of digital control to industrial processes has changed from being the exception to the commonplace. The growth in the applications of computer control has been brought about largely by the rapid advances in hardware design and the reduction in costs: this is most clearly demonstrated by the extent to which the microprocessor has become a normal component of a wide range of electronic systems.

The development of design and production techniques for the software necessary to operate computer systems has not kept pace with the advances in hardware. The problems of software design and production are well-known and there are major research and development programs aimed at remedying the position. These programs are concerned both with developing techniques for design and with the provision of software tools to support both design and production. It is now clearly understood that the creation of software for real-time systems, i.e. systems which have to respond in real-time to events in the outside world, is one of the most difficult areas of software design and production.

The difficulty faced by both engineering students and by experienced engineers is that traditional computing courses for engineers have emphasized the hardware and programing language aspects of computer systems. The languages usually taught – FORTRAN and more recently Pascal – do not have any support for real-time concurrent processing or for direct manipulation of the computer hardware. Applications used to illustrate the teaching have usually been restricted to stand-alone scientific programs. Attempts to go beyond this have required instruction in the use of assembly languages. The majority of books which deal with real-time applications and with digital control assume that the software will be written in an assembly language. They concentrate almost entirely on the techniques for coding algorithms and are therefore concerned with speed of execution, memory usage, and the effects of word length on accuracy. These are, of course, important problems in many applications; however, there are applications for which a different approach, based on the use of high-level languages is appropriate and this book attempts to address this area.

The book is intended for final year undergraduate students and practising engineers. It assumes that the reader will have some familiarity with at least one high-level language and possibly with an assembly language as well. The first part of the book (Chapters 1, 2 and 3) provides a general overview of the subject including definitions and classifications of real-time systems, computer control configurations and hardware requirements.

Chapter 4 deals with methods of implementing a controller on a digital computer. The problems are illustrated by considering the implementation of the traditional three-term (PID) controller. A brief coverage of the implementation of a controller given in z-transform notation, obtained either by the use of discrete design methods or through discretization of a continuous controller design, is given. Knowledge of z-transforms is not required in order to understand the section.

Software design techniques for real-time systems are introduced in Chapter 5. The principles underlying two methods, MASCOT and real-time structured design, are covered. Emphasis is given to dividing the software into modules and to the use of multi-tasking. The traditional approach to implementing multi-tasking – the use of a real-time operating system – is covered in Chapter 6. The general features of operating systems are introduced by describing a simple single-user, single-task, operating system (CP/M 80). The various additional requirements for real-time multi-tasking are then introduced.

In Chapter 7 the approach to supporting multi-tasking based on the use of a real-time language with minimum operating system support are considered. The basic ideas of concurrent programming, use of semaphore, signals and monitors are described. The general language requirements for real-time programming are covered in Chapter 8 and a brief comparison of a number of languages is given in Chapter 9. The examples in the book are given in several languages, but predominantly in Modula-2.

Many people have assisted in producing this book and I am grateful to all of them. Particular thanks are due to Steve White, a former colleague; to the many students who have assisted in developing my understanding both through class discussion and through the project work which they have carried out, and to the technical staff of the Department of Control Engineering, University of Sheffield. I wish also to thank the staff of Prentice Hall, in particular Glen Murray and Andrew Binnie, for their advice and support.

S.B.

Acknowledgements

The Publishers wish to thank the following for permission to reproduce extracts from published material:

Peter Peregrinus Ltd, for Figures 2.1, 2.5, 2.14, 9.29 reproduced from S. Bennett and D.A. Linkens (1984), *Real-time Computer Control* and for Figure 2.8, reproduced from S. Bennett and D.A. Linkens (1982), *Computer Control of Industrial Processes*.

The Institution of Electrical Engineers for Figures 8.4, 8.5, 8.6, 8.7 reproduced from B.S. Hoyle (1984) 'Engineering microprocessor software', *Electronics and Power*, **30**.

The American Society of Mechanical Engineers for Figure 1.1 redrawn from G.S. Brown and D.P. Campbell (1950) 'Instrument engineering: its growth and promise in process control problems', *Mechanical Engineering*, **72**, p. 124.

Ellis Horwood for Figure 7.7 reproduced from S.J. Young (1982), *Real-time Languages*.

Van Nostrand Reinhold for Figure 9.24 reproduced from D.A. Mellichamp (ed.) (1983), *Real-time Computing*.

Contents

1

Introduction to Real-time Systems

1.1 HISTORICAL BACKGROUND

The earliest proposal to use a computer operating in 'real time' as part of a control system was made in a paper by Brown and Campbell [1950]. The paper contains a diagram (see Figure 1.1) which shows a computer in both the feedback and feed-forward loops. Brown and Campbell assumed that analog computing elements were the most likely to be used but they did not rule out the use of digital computing elements. The first digital computers developed specifically for real-time control were for airborne operation, and in 1954 a Digitrac digital computer was successfully used to provide an automatic flight and weapons control system.

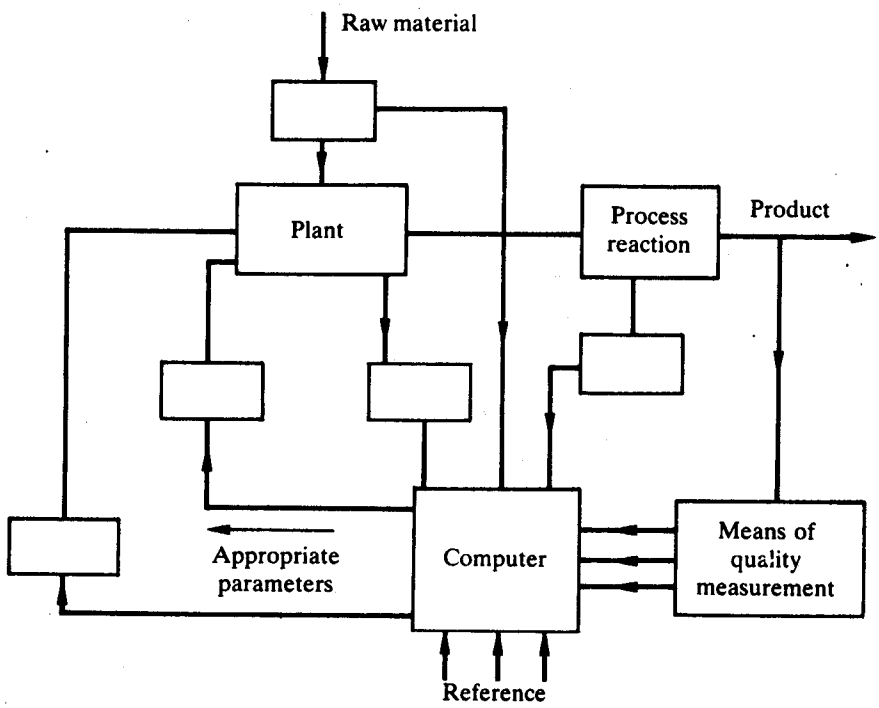

Fig. 1.1 Computer used in control of plant (redrawn from Brown and Campbell *Mechanical Engineering*, **72**, 1950).

The application of digital computers to industrial control began in the late 1950s. The initiative came, not from the process and manufacturing industries, but from the computer and electronic systems manufacturers who were looking to extend their markets and to find outlets for equipment which had failed to be adopted by the military [Williams 1983]. The first industrial installation of a computer system occurred in September 1958 when the Louisiana Power and Light Company installed a Daystrom computer system for plant monitoring at their power station in Sterling, Louisiana. This was not a control system: the honor of the first industrial computer control installation went to the Texaco Company who installed an RW-300 (Ramo-Wooldridge Company) system at their Port Arthur refinery in Texas, which achieved closed-loop control on March 15, 1959 [Anon 1959].

During 1957–8 the Monsanto Chemical Company in cooperation with the Ramo-Wooldridge Company, studied the possibility of using computer control and in October 1958 decided to implement a scheme on the ammonia plant at Luling, Louisiana. Commissioning of this plant began on January 20, 1960 and closed-loop control was achieved on April 4, 1960 after an almost complete rewrite of the control algorithm part of the program and considerable problems with noise on the measurement signals. This scheme, like the system installed by the B.F. Goodrich Company on their acrylanite plant at Calvert City, Kentucky in 1959–60, and some 40 other systems based on the RW-300, were supervisory control systems used for steady-state optimization calculations to determine the set-points for standard analog controllers; that is, the computer did not control directly the movement of the valves or other plant actuators.

The first *direct digital control (DDC)* computer system was the Ferranti Argus 200 system installed in November 1962 at the ICI ammonia-soda plant at Fleetwood, Lancashire, the planning for which had begun in 1959 [Burkitt 1965]. It was a large system with provision for 120 control loops and 256 measurements, of which 98 and 224 respectively were used on the Fleetwood system. In 1961 the Monsanto Company also began a DDC project for a plant in Texas City and a *hierarchical* control scheme for the petrochemical complex at Chocolate Bayou.

The Ferranti Argus represented a change in computer hardware design in that the control program was held in a ferrite core store rather than on a rotating drum store as used by the RW-300 computer. The program was held in a programmable read-only memory; it was loaded by physically inserting pegs into a plug board, each peg representing one bit in the memory word. Although laborious to set up initially, the system proved to be very reliable in that destruction of the memory contents could only be brought about by the physical dislodgment of the pegs. In addition, security was enhanced by using special power supplies and switch-over mechanisms to protect information held in the main core store. This information was classified as follows:

1. *Set points* Loss most undesirable;
2. *Valve demand* Presence after controlled stoppage allows computer to gain control of plant immediately and without disturbance: *bumpless transfer*;
3. *Memory calculations* Loss is tolerable, soon will be updated and only slight disturbance to plant; and

4. *Future development calculation* Extension to allow for optimization may require information to be maintained for long periods of time.

In addition to improved reliability the Argus system provided more rapid memory access than the drum stores of the RW-300 and similiar machines and as such represented the beginning of the second phase of application of computers to real-time control.

The computers used in the early 1960s combined magnetic core memories and drum stores, the drums eventually giving way to hard disk drives. They included the General Electric 4000 series, IBM 1800, CDC 1700, Foxboro FOX 1 and 1A, the SDS and Xerox SIGMA series, Ferranti Argus series and Elliot Automation 900 series. The attempt to resolve some of the problems of the early machines led to an increase in the cost of systems: the increase was such that frequently their use could be justified only if both DDC and supervisory control were performed by the one computer. A consequence of this was the generation of further problems particularly in the development of the software. The programs for the early computers had been written by specialist programmers using machine code; and this was manageable because the tasks were clearly defined and the quantity of code relatively small. In combining DDC and supervisory control, not only had the quantity of code for a given application increased, but the complexity of the programming also increased in that the two tasks had very different time-scales; and the DDC control programs had to be able to interrupt the supervisory control programs. The increase in the size of the programs meant that not all the code could be stored in core memory: provision had to be made for the swapping of code between the drum memory and core.

The solution appeared to lie in the development of general purpose operating systems and high level languages. In the late 1960s real-time operating systems were developed and various PROCESS FORTRAN compilers made their appearance. The problems and the costs involved in attempting to do everything in one computer led users to retreat to smaller systems for which the newly developing minicomputer (DEC PDP-8 PDP-11, Data General Nova, Honeywell 316, etc.) were to prove ideally suited. The cost of the minicomputer was small enough to avoid the need to load a large number of tasks onto one machine; indeed by 1970 it was becoming possible to consider having two computers on the system, one simply acting as a stand-by in the event of failure.

The advent of the microprocessor in 1974 led to a further reappraisal of approach and the development of *distributed systems*. These developments are considered in more detail in Chapter 2.

1.2 ELEMENTS OF A COMPUTER CONTROL SYSTEM

As an example we shall consider a simple plant, a 'hot-air blower' as shown in Figure 1.2. A centrifugal fan blows air over a heating element and into a tube. A thermistor bead is placed at the outlet end of the tube and forms one arm of a bridge circuit. The

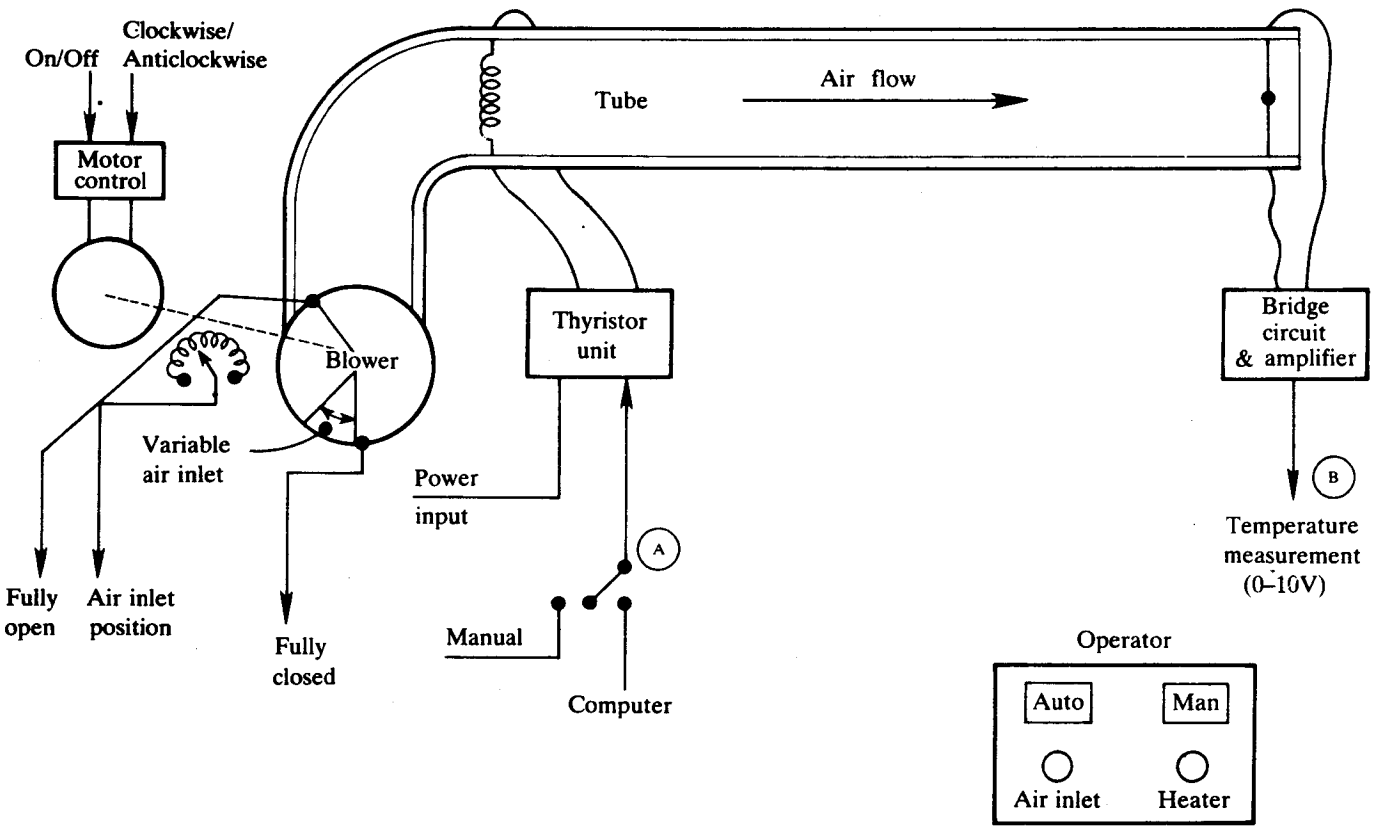

Fig 1.2 A simple plant: a hot-air blower.