DESIGN OF A MONITOR FOR THE DEBUGGING AND DEVELOPMENT OF MULTIPROCESSING PROCESS CONTROL SYSTEMS

A thesis presented
in partial fulfilment of the requirements
for the degree of
Master of Technology in Computing Technology
at
Massey University

GILLIAN DOBBIE

February 1987

ACKNOWLEDGEMENTS

There are a number of people I wish to thank for the parts they have played, either directly or indirectly, in the production of this thesis.

I would like to sincerely thank my supervisors, Dr. Tim Hesketh and Dr. Bob Chaplin. Tim developed the RTS process control system and hence provided the incentive for the production of this thesis. Bob guided me through the masterate, offering much welcome criticism where necessary, throughout all the stages of the work reported in this thesis.

I am also extremely grateful to the other members of the Production Technology Department and to members of the Computer Science Department. They showed interest, gave support and listened. My particular thanks goes to Paul Lyons whose constructive criticism was invaluable in shaping this thesis.

Finally I would like to take this opportunity to thank my family and friends whose encouragement and support throughout my years at Massey University provided much incentive towards the completion of this thesis.

CONTENTS

List of Figures

ABSTRACT

This thesis describes the design of a general purpose tool for debugging and developing multimicroprocessor process control systems. With the decreasing price of computers, multimicroprocessors are increasingly being used for process control. However, the lack of published information on multiprocessing systems and distributed systems has meant that methodologies and tools for debugging and developing such systems have been slow to develop. The monitor designed here is system independent, a considerable advantage over other such tools that are currently available.

Chapter 1

Introduction

This thesis is divided into five chapters. This first chapter is the introduction, giving the background of the work to be done.
Chapter Two considers tools for debugging multiprocessing systems and looks at existing multiprocessing process control systems.
Chapter Three outlines multiprocessing considerations and decisions which were encountered in the design of the monitor.
Chapter Four outlines the monitor requirements and design specifications, giving a scenario of activities for which the monitor may be used.
Chapter Five descibes the implementation of the monitor. It outlines the hardware and software used and how they fit together.
Finally, Chapter Six contains the conclusions and a proposal for furthur research which has stemmed from this thesis.

1.1 Process Controllers

Digital industrial controllers, with control functions realised in software are now common place. They must meet stringent reliability and availability specifications. This has led to the design of real-time systems with software designed with the requirements of reliability, safety and management in mind.

However a control system encompasses much more than the process control functions themselves. Indeed it can be argued that the following support functions:

- process interface,
- user interface,
- alarm monitoring,
- data logging,
- start-up and shut-down,
- sequence control

are as important as the controller. Certainly they are inordinate consumers of computing resources.

The traditional approach to servicing the above functions has been to use a bigger and faster monoprocessor computer, but with the advent of low cost powerful microcomputers the multiprocessor solution can be economically explored. Many different machine architectures and software packages [3,8,24,25,26,30,39,42,47] have been proposed. The difficulties of deciding which structures are optimal or even suitable for a given control system are not only caused by implementation costs but also by the lack of suitable tools for fault tracing and performance assessment in multiprocessor systems.

RTS, a small digital controller will be used to illustrate the associated problems, and later to assist in the design and testing of a multiprocessor.

1.2 RTS

RTS (real-time system) is a real-time process control system which was designed and the program written by Dr. T. Hesketh [19]. It has been used in a number of practical applications including controlling the atmosphere in the climate rooms at the DSIR and the drum temperature at a wool scourer. The program runs on a single Z80 processor and is written mainly in PL/I with some routines in Z80 assembly language. It has been written so that it is relatively easy to alter.

RTS is a general purpose process controller which may be set up in a specialised configuration when it is to control a specific situation. It assumes multi-independent single variable loops. These are loops where a given manipulated variable affects only its own controlled variable and there is no interaction between the loops. RTS implements three of the five functions mentioned above:

- user interface function,
- process interface function,
- control interface function.

The other functions were not incorporated because they would overload the processor. An 8-bit microprocessor has a limited capability and so priorities must be given to the activities which are considered for implementation. The control function has the highest priority and the data logging the least.

Start-up and shut-down actions do not occur automatically in RTS. The sequences can be carried out manually by the operator. This is possible in this situation as the operator using the system would be skilled. This gives the operator more flexibility in the operation of the process.

1.2.1 User Interface Function

The user interface function has been minimised to reduce processor loading. This function in RTS includes:

- menu display,
- process, state and set-up displays,
- user responses,
- plotting and printing.

1.2.2 Process Interface Function

The process interface function includes:

- A/D and D/A operations,
- filtering operations,
- alarm monitoring.

1.2.3 Control Interface Function

The control interface function incorporates the following five sections:

- measurement of the process behaviour,
- determination of the model,
- design of the controller,
- subsequent definition of feedback gains by the controller,

- use of feedback gains to obtain a control signal.

Measurement of the Process Behaviour

In this phase, system-specific information must be gathered. This information comprises:

- sampling and control time intervals,
- the signals and their interconnections (loops).

There are three different time intervals:

- the interrupt time interval (usually in the order of milli-seconds),
- measurement sample rate, set by the user,
- actuation sample rate, also set by the user.

A signal in RTS is a data structure for the definition of external variables i.e. set-point, measurement, actuation. Loops are hierarchically superior data structures which are incorporated in the system to associate signals in groups.

Determination of the Model

The system identification involves a recursive procedure whereby the data which becomes available at the beginning of each sampling interval are used to improve the previous estimate of the system model. The technique of extended least squares [2] is used to determine the model and a Kalman Filter [23] is used to obtain parameter estimates also recursively.

The plant can be operated at user-selected operating points and have pseudo-random disturbances created by RTS. The subsequent measurements are used in the system identification to estimate model parameters.

Design of Controller

In RTS there are a number of different controller design methods used. They are designed for both on- and off-line operation and include the following controller designs:

- optimal controller,
- self-tuning controller,
- user defined controller,
- three term controller.

The design of an optimal controller uses the Ricatti Equation [27]. To determine the gains of the system it is necessary to iteratively solve an initial value problem for a Ricatti Equation.

The self-tuning controller works well but as it operates on-line, it can begin to model the noise if the system remains in this mode once the system has settled.

The user defined controller allows the user to define and design a controller.

It is necessary to consider the categories of variables which exist.

1.3 Variables in Process Control

There are four important categories of variables related to process control [41]:

- manipulated variables,
- disturbances,
- controlled variables,
- intermediates.

The values of manipulated variables can be adjusted by the control system. Examples include input raw material flow rate, steam pressure etc.
Variables whose values affect the operation of the process but are not subject to adjustment via the control system are known as disturbances (e.g. composition of raw material, ambient air temperature etc.)
Control variables are those whose values measure the performance of the plant, and as such are those which the control system must keep at some set-point (e.g. production rate, production quality etc.)
The intermediate variables appear at some intermediate stage in the process and can not normally be measured.

The general control problem is to adjust the manipulated variables so as to maintain the controlled variables at their set-point in the face of disturbances. The set-point is the value at which the user wants the control variables to operate.

The intermediates may be used advantageously, if they can be measured, in determining what control action should be taken.
One major problem with process plants is the difficulty in deriving a mathematical model of the process from its characteristics. The characteristics depend firstly on the level of the plant operation (the plant often cannot be descibed using a linear model) and secondly, even under constant operating conditions the plant's characteristics change with time. The abilities of a computer make it attractive in this situation where it can collect large quantities of data, analyse it and make logical decisions based on the results.

1.4 Real-Time Programming

The term "real-time" has been described as "any information processing activity or system which has to respond to externally generated input stimuli with a finite and specifiable delay" [33].
For real-time process control systems the delay must not only be specifiable, it must also be constant as the theory of the identification and controller design is based on the assumption of regular measurement and actuation.
Software for real-time process control systems must respond to "clock tick" interrupts, user requests and changes in the process, so there are time critical components. The software must interact with the dynamic properties of the industrial system and must react to stochastically occurring events.

There are two different types of events:

1. events which must be serviced whenever a time interval boundary is reached,
2. events which are serviced as and when required.

With both types of event, the control software must react within a certain time and parallel tasks must also be synchronised within the software system.

1.5 Multiprocessing vs Multiprogramming

To alleviate the problems in RTS and indeed any other processor bound process control systems, it is possible to implement the system as either a multiprogramming system or a multiprocessing system.

A multiprogramming system appears to execute many tasks at the same time. However there is always only ever one set of instructions being executed at a particular time. A multiprogramming system time-slices, devoting a set amount of time to all the processes that are currently being run. In the case of RTS, the use of multiprogramming would not increase the speed of operation. RTS cannot have other functions added as it is presently processor bound (section 1.2).

Multiprogramming provides no relief from this situation as the implementation of multiprogramming would take up more processing power, and hence make RTS slower. In real-time process control systems, the interrupts must be serviced immediately. Interrupt systems like this would complicate a multiprogramming operating system and the software would be difficult to write.

The definition of multiprocessing differs from paper to paper. In many books only tightly coupled systems (section 3.6) are considered multiprocessing systems. In this thesis, a broader interpretation will be used, in which a multiprocessing system consists of both tightly and loosely coupled systems. A multiprocessing system consists of two or more processors linked together to perform one function. It was thought that RTS would run more efficiently as a multiprocessing system as it is presently processor bound, and there is a lot of code that could be executed in parallel, hence saving time. RTS can be divided into four functional subunits which could be executing at the same time. They are:

- the user interface function,
- the processor interface function,
- the control interface function where this could be implemented as an identification function and a control function.

1.6 Objectives of this Project

As mentioned previously (section 1.1), the advent of microprocessors has generated interest in alternative, multiprocessor-based methods of providing process control functions. The main objective of this project has been to design a monitor (a tool for debugging systems) for use in the development of such multiprocessor-based systems. This monitor was developed for the conversion of RTS from a single processor to a multiprocessor system. Every effort has been made to keep the monitor as general as possible. To this end, as few assumptions as possible about the nature of the multiprocessing system have been made. Any design decision that was made was considered from a general view-point and if a decision could not be reached then the logic and the structure of RTS was taken into account in reaching the decision.