Control system architectures for distributed manipulators and modular robots

This item was submitted to Loughborough University's Institutional Repository by the/author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/12467](https://dspace.lboro.ac.uk/2134/12467)

Publisher: © Terence William Thatcher

Please cite the published version.
CONTROL SYSTEM ARCHITECTURES FOR DISTRIBUTED
MANIPULATORS AND MODULAR ROBOTS

by

TERENCE WILLIAM THATCHER

A Doctoral Thesis
submitted in partial fulfilment of the
requirements for the award of
Doctor of Philosophy
of Loughborough University of Technology
Department of Engineering Production

October 1987
© by Terence William Thatcher
DECLARATION

No part of the work described in this Thesis has been submitted in support of an application for any other degree or qualification of this or any other University or other Institution of learning.
The author wishes to thank:

Professor R.H. Weston, Dr P.R. Moore, Mr R. Harrison, Mr J.D. Gascoigne, Mr C.M. Sumpter, Mr G.P. Charles, Mr N.D. Carpenter, Mr D. Walters, Dr R. Jones, Mr L. Liddard, Mr G. Morgan, Mr E.E.P. Hard, and Mr R. Brigginsaw for their continued help, interest, encouragement, and friendship,

My parents for their moral support and practical contributions,

The SERC, Martonair Ltd, and Nutek (London) Ltd for their financial and technical assistance.
SYNOPSIS

Keywords: Distributed manipulators, Modular robots, Pneumatic servo-control, Hierarchical control, Distributed multiprocessor control, Robot programming, Robot control.

This thesis outlines the evolution of computer hardware and software architectures which are suitable for the programming and control of modular robots and distributed manipulators.

Fundamental aspects of automating manufacturing functions are considered and the use of flexible machines, constructed from components of a family of mechanical modules and associated control system elements, are proposed. Many of the features of these flexible machines can be identified with those of conventional industrial robots. However a broader class of manufacturing machine is represented in as much as the industrial user defines the kinematics and dynamics of the manipulator. Such flexible machines can be referred to as "modular robots" or, where the mechanical modules are arranged in concurrently operating but mechanically decoupled groups, as "distributed manipulators".

The main body of the work reported centred on the design of a family of computer control system elements which can serve a range of distributed manipulator and modular robot forms. These control system elements, whose cost is commensurate with the size and complexity of the manipulator's mechanical configuration, necessarily have many of the features found in robot controllers but also require properties of reconfigurability, programmability, and control system performance for the considerable array of manipulator configurations which can be constructed.

The role of distributed manipulators and modular robots, and the design principles involved in constructing control systems for this type of flexible machine are discussed, including system architecture, supervisory control requirements, programming methods, servo-control approaches, interface circuit design, and inter-processor communications.

It is shown that in advancing to the availability and industrial use of commercial control system elements, the principles established in the conceptual design and prototyping phases required some refinement and modification, but were fundamentally sound. A resulting product, known as a Single Axis Controller, is marketed worldwide by Martonair Ltd and is finding increasing application in many areas of manufacturing industry.
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>i</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>ii</td>
</tr>
<tr>
<td>Synopsis</td>
<td>iii</td>
</tr>
<tr>
<td>Contents</td>
<td>iv</td>
</tr>
<tr>
<td>CHAPTER I - INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER II - THE IMPACT OF AUTOMATION TECHNOLOGY</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Automation in a Non-Economic World</td>
<td>4</td>
</tr>
<tr>
<td>2.3 The International Spread of Automation</td>
<td>6</td>
</tr>
<tr>
<td>2.4 The Future of British Manufacturing Industry</td>
<td>7</td>
</tr>
<tr>
<td>CHAPTER III - TYPES OF AUTOMATION</td>
<td>9</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Hard (Dedicated) Automation</td>
<td>9</td>
</tr>
<tr>
<td>3.3 Soft (Flexible) Automation</td>
<td>11</td>
</tr>
<tr>
<td>3.4 Other Automation Types</td>
<td>14</td>
</tr>
<tr>
<td>3.5 Summary of the Use of Industrial Robots in Automatic Production Equipment</td>
<td>15</td>
</tr>
<tr>
<td>CHAPTER IV - CONVENTIONAL MODULAR AND DISTRIBUTED ROBOTICS</td>
<td>17</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>17</td>
</tr>
<tr>
<td>4.2 Conventional Robots</td>
<td>17</td>
</tr>
<tr>
<td>4.3 The Concepts of Modular Robotics</td>
<td>17</td>
</tr>
<tr>
<td>4.4 Distributed Manipulators</td>
<td>18</td>
</tr>
<tr>
<td>4.5 The Advantages of Distributed Manipulators</td>
<td>24</td>
</tr>
<tr>
<td>4.6 Further Implications of the Concepts of Distributed Manipulators</td>
<td>25</td>
</tr>
<tr>
<td>4.7 Note on Terminology</td>
<td>28</td>
</tr>
</tbody>
</table>
5.1 Introduction

5.2 The Requirements of a Distributed Manipulator Control System

5.2.1 Minimisation of Costs
5.2.2 Control of Distributed Manipulators
5.2.3 The Number of Controlled Axes
5.2.4 Programming Systems
5.2.5 Actuation
5.2.6 Fast Program Changeover
5.2.7 Servo-control Function
5.2.8 General Plant Interface and Control
5.2.9 Integration into CIM Systems
5.2.10 Processor Intensive Activities
5.2.11 Enhancements

5.3 Possible Architectures

5.3.1 Single Processor with Analogue Servo-control
5.3.2 Single Processor
5.3.3 Multiprocessor
5.3.4 Distributed Multiprocessor

5.4 The Choice of a Distributed Multiprocessor Architecture

5.4.1 Single Processor with Analogue Servo-control
5.4.2 Single Processor
5.4.3 Multiprocessor
5.4.4 Distributed Multiprocessor
5.4.5 Hierarchy of Distributed Manipulator Control

5.5 Requirements of the General Manufacturing Machine Controller

5.6 Architectures for the General Manufacturing Machine Controller

5.7 Pneumatically-Actuated Distributed Manipulator System

5.7.1 Limitation of Objectives
5.7.2 Pneumatic Actuation
5.7.3 Controller Architecture
5.7.4 Enhancements

5.8 Note on Terminology
CHAPTER VI - CONVENTIONAL ROBOTICS

6.1 Introduction

6.2 An Historical Perspective

6.3 The State-of-the-Art

6.4 Application Areas

6.5 Mechanical Design

6.5.1 For Conventional Robots

6.5.2 For Distributed Manipulators

6.6 Actuation Systems

6.6.1 For Conventional Robots

6.6.2 For Distributed Manipulators

6.7 Controller Design

6.7.1 Control Systems Architectures

6.7.2 Electronic Hardware

6.7.3 Software

6.7.4 Control Systems for Distributed Manipulators

CHAPTER VII - ROBOT PROGRAMMING SYSTEMS

7.1 Introduction

7.2 Purpose of Robot Programming

7.3 For End-stop and Multi-Stop Robots

7.4 Programming by Teaching

7.4.1 Teach by Pendant Features and Applications

7.4.2 Teach by Showing Features and Applications

7.5 Language-based Approaches

7.5.1 Reasons for Language-based Approaches

7.5.2 Requirements

7.5.3 History

7.5.4 Current Trends

7.5.5 Standardisation

7.6 Off-line Programming

7.7 Task Orientated Methods

7.8 Distributed Manipulator Programming
CHAPTER VIII - SUPERVISORY CONTROL

8.1 Introduction 84

8.2 Supervisory Controller Requirements 84

8.2.1 Supervisor to SCC/SAC Communications 84

8.2.2 Programming and Set-point Teaching 84

8.2.3 General Plant Interface and Control Functions 84

8.2.4 Computer Integrated Manufacture (CIM) 85

8.2.5 Upgradability 85

8.3 Supervisor Hardware 86

8.4 Current Approaches to the Control of Flexible Automated Production Plant 86

8.4.1 Programmable Logic Controllers 87

8.4.2 Industrial Computers 88

8.4.3 Ladder Diagram and Language Based Programming 88

8.4.4 Industrial Process Controllers 90

8.5 Programmable Logic Controllers & Industrial Computers as Supervisors 92

8.5.1 PLC Supervisory Controllers 92

8.5.2 Industrial Computers 92

8.6 Development of Industrial Computer Based Supervisors 93

CHAPTER IX - THE EVOLUTION OF PNEUMATIC MOTION CONTROL

9.1 Introduction 94

9.2 Early Servo-control Approaches 96

9.3 Pneumatic Proportional Servo-valves 96

9.4 Pneumatic Servo-control Modelling 98

9.5 Control Strategy Testing and Evaluation 98

9.5.1 Standard Evaluation Criteria 101

9.5.2 Hardware Design and Choice of Processor 101

9.5.3 Performance Assessment Software 102

9.6 Control Strategy Implementation 106

9.7 Performance Characteristics 106

9.8 Algorithm Execution Speed 108
CHAPTER X - SERVO-VALVE INTERFACE DESIGN

10.1 Introduction 110
10.2 The Control Valve Interface 110
10.3 Digital to Analogue Conversion (DAC) 111
  10.3.1 DAC Selection 111
  10.3.2 Output Range 113
  10.3.3 Pulse Width Modulation (PWM) 113
10.4 Servo-Amplifier Design 114
  10.4.1 Requirements 114
  10.4.2 Servo-amplifier Development 114
  10.4.3 Mark 3 Version Servo-amplifier 115
  10.4.4 Resistor Value Selection 115
  10.4.5 Mark 3 Version Testing 115
10.5 Hysteresis and Dither 125
  10.5.1 Experimental Results and Conclusions 125
  10.5.2 Summary of Dither/Hysteresis Aspects 127

CHAPTER XI - POSITION VELOCITY AND ACCELERATION MEASUREMENT

11.1 Introduction 129
11.2 Selection of the Position Sensor 129
11.3 Operation of Optical Incremental Encoders 129
11.4 Counting in Software 131
11.5 Counting in Hardware 133
  11.5.1 Preliminary Experiments 135
  11.5.2 Secure Encoder Counter Circuits 138
  11.5.3 Counter Circuit Initialisation and Load 138
  11.5.4 Other Secure Encoder Counter Circuits 142
11.6 Velocity and Acceleration Measurement 142
  11.6.1 Velocity and Acceleration Transducers 142
  11.6.2 Position Differentiation 142
11.7 Single Chip Incremental Encoder Counters 146
14.2 Industrial Collaborator's Viewpoint
14.3 Pneumatically-actuated Motion Elements and Servo-valves
14.4 Electronics Construction Capability
14.5 Division of Responsibility
14.6 Architecture and Packaging
14.7 Hardware Design
14.8 Software Design
  14.8.1 Hardware Test Programs
  14.8.2 Run-time Software
14.9 Construction
  14.9.1 Hardware
  14.9.2 Software
  14.9.3 Hardware/Software Integration
14.10 Tests and Results

CHAPTER XV - SAC PRE-PRODUCTION AND PRODUCTION VERSIONS
15.1 Introduction
15.2 Architecture and Packaging
15.3 User Interface and Programming/Teach System
15.4 Supervisor/SAC Interfaces
15.5 Servo-control
15.6 Hardware Design
  15.6.1 74LS2000-based Encoder Counters
  15.6.2 Division
15.7 Software Design
15.8 Hardware Test Program Design
15.9 Run-time Software Design and Structure
  15.9.1 Requirements
  15.9.2 Structure
  15.9.3 ISRO and the Background Task Loops
  15.9.4 Timer Interrupt Service Routine (ISR3)
15.9.5 Byte Received Interrupt Service Routine (ISR4) 221
15.9.6 Battery-backed RAM Integrity Checking 223
15.9.7 Operation Mode Changeover 223
15.9.8 Microscribe User Interface 225
15.9.9 PLC Interface 226
15.9.10 Supervisor Serial Interface Instruction Execution 226
15.9.11 SAC Status and Error Conditions 226

15.10 Construction 228

15.10.1 Hardware 228
15.10.2 Software 228
15.10.3 Hardware/Software Integration 230

15.11 Results 230

CHAPTER XVI - CONCLUSIONS AND RECOMMENDATIONS 232

GLOSSARY OF TERMS 238

REFERENCES 239

APPENDICES

APPENDIX I - SERVO-VALVE RESISTOR VALUE SELECTION 253

APPENDIX 2 - OPERATIONAL DETAILS OF THE ENCODER COUNTER CIRCUIT 260

APPENDIX 3 - THE TEXAS INSTRUMENTS TMS 9900 FAMILY OF MICROPROCESSORS 268

A3.1 Introduction 268
A3.2 Architecture 268
A3.3 The TMS 9900 Microprocessor 269
  A3.3.1 Memory Map, Addressing and Word Length 270
  A3.3.2 Interrupts 270
  A3.3.3 I/O Interfaces 271
A3.4 The TMS 9995 Microprocessor 271
  A3.4.1 On-chip RAM 271
  A3.4.2 On-chip Timer/event Counter 271
  A3.4.3 Interrupts 272
  A3.4.4 I/O Interfaces 272
  A3.4.5 Pin out 274
  A3.4.6 Execution Speed 274
A3.5 TMS 9902 Asynchronous Communications Controller 274
APPENDIX 4 - SAC PRE-PRODUCTION AND PRODUCTION VERSION HARDWARE TEST PROGRAM

APPENDIX 5 - SAC PRE-PRODUCTION AND PRODUCTION VERSION RUN-TIME SOFTWARE
CHAPTER I

INTRODUCTION

Much has been written and spoken in recent years of the increasingly significant role that automation and robotics will play in the coming period. Learned and well informed authors have foretold almost unmanned factories and example show piece manufacturing plant of this type has already been implemented. Politicians and economists have advised that, to enable successful competition with overseas producers, increased levels of automation will be essential in the United Kingdom.

Although reduction in labour cost is a much used reason for automating, other less tangible benefits are equally important and it can be argued objectively that automation is beneficial to both manufacturing industry and society as a whole. Why then is the UK often accused of applying too little automation too late? The examples of the UK's motor cycle and machine tool industries would suggest that the accusations have foundation. The reasons for these problems are complex but undoubtedly the technical complexity of developing flexible production plant capable of manufacturing a sufficiently large range and consequently volume of products to justify the plant's cost is one.

Although the restricted view of some currently applied justification schemes may be partially at fault, if the development of flexible production equipment could be facilitated and its cost reduced, more automation would result. Research and development carried out since 1981 by staff at the Department of Engineering Production, Loughborough University of Technology (LUT) and Martonair UK Ltd, under the auspices of an SERC collaborative grant, has had the objective of facilitating flexible automation and reducing the cost of some of the equipment required.
The route chosen to attain this goal was through the concept of distributed modularity in robots (Chapter IV, Section 4.3). This idea, although readily accepted today, was at its infancy in early 1981. Since this time many companies, including Martonair, have introduced pneumatically actuated end-stop modular robotic systems, the technology required for these developments being relatively straightforward and largely well understood. The combined LUT and Martonair research team proceeded to examine the feasibility of providing modular units with fully software programmable positioning capability and pneumatic actuation and thus increase the scope of the possible application of modular robotic systems.

As this work progressed the modular robot concept was developed to encompass distributed manipulators employing hydraulically, pneumatically and/or electrically actuated positioning modules as required. Consideration of possible control architectures for such systems gave rise to examinations of the requirements of the family of control system elements necessary for the control of the general class of manipulators.

Clearly the number of technical and academic disciplines required for this effort was to be considerable. Mechanical and pneumatic engineering skills were required for the design of the modules themselves. Mathematical control theory was needed to indicate possible control approaches which could then be implemented in computer hardware and software and consequently tested. Electronics were required to interface the controller to the mechanical and pneumatic elements. Further electronics, software, and interface design would be essential for the linking of the microprocessor controllers to higher level control elements and for reducing the complexity of application by an appropriate design of man-machine interface.

To address a problem of this magnitude a considerable number of researchers have been involved and the efforts of other members of the team must be acknowledged.
Some of the work required to attain the research team's objectives is more fully described elsewhere. The particular purposes of this thesis being to develop the concepts of distributed modular robotics and to show how innovative research led to the development of fully servo-controlled, pneumatically-actuated, distributed modular robotic systems and an associated family of control system elements packaged in an industrially applicable form, which are now commercially available throughout the world. Also some of the general manipulation concepts evolved during the work but not expanded upon have been taken up so that the use of electrically and pneumatically actuated manipulators are being developed and their control systems requirements investigated.

The thesis divides into three parts. The first of these (Chapters 1-5) deals with background, evolution, and the fundamental concepts of distributed modular robotics. The second discusses the enabling technologies required to bring about the development of such systems. In this second part Chapters 6-8 review conventional robotics and its relevance to distributed modular systems whilst Chapters 9-12 describe the fundamental enabling research. The third part (Chapters 13-15) shows how these enabling technologies were drawn together to develop a commercial distributed modular robotic system. The concluding chapter discusses the results of this project and makes suggestions for further work.
2.1 INTRODUCTION

This chapter considers automation in the broad socio-economic environment. The probability of the further spread of automation is discussed in both international and national contexts.

2.2 AUTOMATION IN A NON-ECONOMIC WORLD

To objectively discuss the desirability of automation it is helpful to envisage a world without international economic commercial competition. In such a world it is possible to consider the social and manufacturing advantages and disadvantages of automation technology in isolation.

For many years it has been appreciated that if automation were applied widely, within even a non-economic framework, the level of skill, in a psychological sense of dexterity and decision making ability, would be considerably increased (1). Furthermore less unskilled/semi-skilled labour, in the industrial sense of formally craft trained, and more capital investment would be required to manufacture more and a wider variety of products. The problem of the employment of the workers no longer required arises and by what means one may ask would the wealth generated by this new manufacturing industry be distributed?

A considerable amount has been written by engineers, politicians and economists of the likely personnel requirements of manufacturing industry in the wake of large scale automation. It has been suggested that the application of the so-called new technology will itself contribute to labour requirements (2) and thus assist in alleviating the problem, although this argument can be re-emphasised (3) to suggest that the staff required will
be more highly skilled designers, maintainers and engineers. By contrast a recent study (4) indicates that the so-called sunrise industries, have not, are not, and will not, greatly contribute to levels of employment. The role of service sector industries as increasing users of labour as also been examined (5). These questions are perhaps best addressed by sociologists, politicians and economists (6). However should any of these theories prove correct then acceptable levels of employment (7) would be maintained and the wealth generated spread as those benefiting directly from automation spend their earnings increasingly in the services and consumer markets.

If the staffing requirement of manufacturing industry falls dramatically in real terms, as it is already showing signs of doing (8), and no other major source of employment emerges, then society must address the problem of unemployment and wealth distribution. Here the realms of political ideology and dogma are encountered which are beyond the scope of this thesis. However the destruction of often boring and tedious jobs (9) must be seen as a positive contribution to society, freeing man to explore his/her own interests in art, sport, leisure and science, although it must be noted that the effects of the removal of the work ethic in both psychological and sociological terms are uncertain.

From the manufacturing point of view the greater flexibility of some forms of automation should result in a wider range of products, including the manufacture of one-off items, with improved quality and delivery times (10). As more computer-based equipment is integrated into manufacturing plants (the so-called CIM - Computer Integrated Manufacture) a greater variety, quality and quantity of information on the manufacturing processes becomes available (11). Thus management has greater scope for the use of more advanced techniques and more economical practices so that lower levels of work in progress can be attained, faults found and corrected more rapidly, and "just in time" production achieved (12).
These developments in CIM should in turn offer to consumers products which more nearly match their requirements in terms of specification, delivery, quality, and price.

From a strictly production engineering point of view greater automation is therefore definitely desirable. The social and economic benefits are however less clear and it is thus not easy to argue in favour of the wider use of automation technology without considering commercial influences.

2.3 THE INTERNATIONAL SPREAD OF AUTOMATION

Thus far a non-economic world has been considered in which it has been assumed that the nations en-masse have the choice of adopting automation technology or not. While this has been useful in gaining a picture of the non-commercial effects of automation it is neither truly realistic nor practical.

No examples of past global commercial co-operation exist and indeed history can relate few examples of even two nations working together towards a common commercial goal. Thus on a purely practical level it would appear unlikely that any degree of global collective policy towards automation is likely to emerge. If this is the case then it would seem that the actions of each nation, and of individual companies within nations, are not likely to be guided by the potential benefits or otherwise to mankind considered in the previous section. In adopting more or less automation what then are likely to be the guiding influences? Clearly commercial and economic competitiveness will be of paramount importance.

As indicated in the previous section, automation can offer considerable advances in terms of product quality, batch size, delivery and price. If a company can offer better performance in some or all of these areas than a rival then clearly it will have a competitive edge in the market place. It would therefore seem likely that for this reason alone companies involved in manufacturing throughout the world will be looking to apply greater
levels of automation in their manufacturing processes and some companies, many of which are Japanese, are already moving in this direction (13).

2.4 THE FUTURE OF BRITISH MANUFACTURING INDUSTRY

In the event of a scenario such as that proposed in the previous section, in which companies throughout the world are increasing their levels of automation and consequently their economic competitiveness, what is a probable or indeed a desirable course for British manufacturing industry?

For nearly the past two centuries Great Britain has based its prosperity on manufacturing industry and Empire (14). The concepts of Empire have given way to arguably more humane and civilised systems and for the future Britain must look to its manufacturing base for economic survival and wealth or devise some other approach to international existence.

If Britain were to abandon manufacturing industry it may be possible that the means to economic survival could be found in the improved agricultural and service sector industries, including finance and banking. The desirability of being totally reliant on other nations for even manufactured consumer goods is highly questionable, as is the viability of a service industry without a sound manufacturing foundation. Another system might be developed based on Britain as a job-shop specialising in one-off production or alternatively Britain could endeavour to develop a radically new social structure based on new values and economic patterns.

All of these non-manufacturing or batch-of-one solutions to the problem of economic survival are of doubtful desirability and would seem highly unlikely to emerge in even the medium term future. In this case Britain must compete in the international market for manufactured goods and, as has been argued here (Section 2.3) and elsewhere (15), to achieve this Britain must look to its manufacturing industry to employ much greater levels of automation (16) than it does at present (17)
and to find solutions to possible side-effects such as high unemployment and unacceptably uneven wealth distribution (18).
CHAPTER III

TYPES OF AUTOMATION

3.1 INTRODUCTION

The likely advance of automation internationally has been suggested and the consequent necessity for all countries including the U.K. to further the use of automation in their own manufacturing industries has been established. This chapter examines the possible routes to this automation goal and compares their relative merits.

Although some authors (19) have considered more groupings, in the interests of simplicity and brevity two categories of automation only are discussed in detail, namely hard or dedicated automation, and soft or flexible automation. Any extra types, which may be thought to more or less reside in one or other of these two categories, are outlined in Section 3.4.

3.2 HARD (DEDICATED) AUTOMATION

For the production of large numbers of exactly similar products high levels of capital investment may be justified. In these cases it is often appropriate to use automatic manufacturing equipment dedicated to the production of these products. Such equipment is called hard or dedicated automation being specially designed to produce one product and not easily or quickly changed to produce other products. Because of its dedicated application it is usually possible to optimise cycle times and thus enable high output rates. Automation of this type has been in use in increasing quantities since before the Second World War. During this period its use has increased steadily so that generally the principles involved are now well understood (20). A modern example of this type of equipment is the car body multi-welder, each machine costing typically hundreds of thousands of pounds and being dedicated to the welding of one
particular car body shape. The welding of a different body section is not possible as the machine has been specifically designed in an attempt to optimise the manufacture of a particular shape. It is interesting to note that the needs of the automotive industry have been prominent in promoting the requirement for hard automation and indeed subsequently for flexible automation.

The electronics industry presents another example of hard automation where large numbers of components with standard physical dimensions, such as resistors, capacitors, and dual-in-line (DIL) packaged integrated circuits (ICs), are required to be inserted into printed circuit boards (PCBs). For very large batch runs of a particular PCB automatic insertion machines (21) are economically viable. These costly machines are capable of inserting standard components at rates of greater than one per second and although flexible in the sense of being able to be reprogrammed for different PCB configurations, this operation usually requires human intervention and is lengthy relative to the period needed to fill a single PCB. Thus if a small number of a PCB is to be manufactured these insertion operations are often performed manually. Also automatic insertion machines are not used for transistors, transformers and other components having less common packages. The primary problem is not that these components are technically more difficult to handle and insert, although this may be a contributory factor, but that they are not usually used on PCBs in such large numbers and therefore the cost of buying or developing machines and associated tools and fixtures for their automatic insertion is not justified.

In summary dedicated automation has the advantages of being fast and economical for very large or infinite batch production. Its disadvantages are high initial cost and lack of flexibility/reconfigurability making it inappropriate for the manufacture of products with several substantially different versions or goods subject to design changes, a fundamental problem when the trend towards reduced product life is considered.
3.3 SOFT (FLEXIBLE) AUTOMATION

For the reasons described above hard automation systems are frequently inappropriate for the automatic manufacture of products in small batches. The cost of the equipment is not usually reduced but the number of items over which it can be spread is lessened considerably. Until recent years manual manufacture has been the only solution but a second possibility is now available with the advent of reprogrammable devices, industrial robots representing a particular class of such devices.

The problem, in essence is to design a production facility which is mechanically capable of producing a variety of items whose combined batch sizes are sufficient to justify the facility's cost. To achieve this end the production equipment must have the flexibility to change both mechanical manipulation and operation sequences. Mechanical flexibility is difficult and expensive to achieve. For instance the movements required together with the necessary tools and fixtures to assemble a motor car are obviously vastly different to those involved in the assembly of a television set. Thus the necessary range of manipulations and tools is usually limited by restricting the variety of items produced to those with only small differences such as different sizes or specifications of the same product or members of a family of closely related products (22).

One route to this goal of limited manipulation flexibility is to construct the plant from primary actuation, sensory and control elements. That is to purchase conveyors, electric motors, lead screws, micro-switches, incremental encoders, computers, and such equipment, and from these build a complete system (23). This approach has much to recommend it. Performance can be optimised and good cycle times achieved; unfortunately this route is also likely to incur large development costs.

An alternative route is to do the minimum amount of new construction and to use off-the-shelf items of
flexible production equipment such as industrial robots, whose primary raison d'être is flexible automation. Also, being themselves relatively mass produced items, the cost of an industrial robots is less than re-designing a similar device from its constituent parts. However, jigs, fixings and other tooling peculiar to the product to be manufactured must still be designed and made. Furthermore good cycle times will be less easily achieved.

Even a family of very similar products will require for their manufacture, not only different movements, but also different sequences of movements and other operations. Thus the plant control equipment will be required to be able to change between different sequences when the desired output product alters.

If these different products are to be manufactured in small batches then the changeover time must be minimised if the percentage down time is not to seriously impair productivity. This requirement adds to the automatic production equipment specification. Clearly very little or no manual mechanical adjustments must be necessary. Also if the plant controllers are to change sequences rapidly the new sequence should not have to be loaded manually (as with a peg board controller for instance) nor preferably be read electro-mechanically (from paper tape for example). Ideally the sequence controllers should merely have to switch between programs already loaded or receive them automatically and electronically from a remote store.

Modern microprocessor based industrial robots can meet these requirements being able to store several sequence programs locally, given sufficient memory, and many are designed with communications interfaces through which they can receive new sequence programs from a central computer. Programmable controllers and small computers are ideally suited to the control function having excellent local storage and communications capabilities, as well as being available in ruggedised packages and in a price range from a few hundred pounds upwards.
Many products are not susceptible to a hard automation approach either having many different variations or being subject to frequent design or styling changes. An example in the first category is the production of pneumatic cylinders and valves which usually have many variations on essentially similar themes, but each variation selling in relatively small quantities. In the case of pneumatic cylinders, besides the complication suggested above, one-off orders for particular stroke lengths and even bore diameter are commonplace. The textile and shoe manufacturing industries are obviously susceptible to seasonal and trend changes and do not therefore have the length of production runs necessary to justify hard automation.

Examples of flexible automation equipment designed and built from primary elements are uncommon owing to the time and cost involved in such work. However as early as 1980 the limitations of hard automation and robot based flexible assembly were recognised by Rosted (23). This work considered the manufacture of pressure fed beer taps with a purpose-built manipulator. Hard automation being too expensive for the output volume required and not sufficiently flexible. Furthermore the assembly operations needed were too simple to justify the use of an industrial robot. It is interesting to note that Rosted also recognised the desirability of being able to utilise robot modules.

A more recent, but as yet undocumented example of this type of flexible automation equipment has been built by ICL who needed to be able to unload pallet delivered PCBs into a flow soldering machine and reload after washing. The use of hard automation being precluded by the variety of PCB and pallet sizes and shapes. The difficulty of the handling task and the separation of the load and unload stations meant that industrial robots were technically unsuitable and hard to cost justify.

At their Greenock plant in Scotland IBM have constructed a flexible production line manufacturing the keyboards used in their personnel computer and terminal
ranges. In this application industrial robots have been used where possible as a flexible, expedient and cost effective means of manipulating and assembling parts although considerable effort and cost has also been incurred in the design and construction of the jigs, tools, transportation elements, and fixings needed to assist in the manufacture(24). A total of five IBM SCARA configuration industrial robots are used, all of which are programmed using the AML/E programming language (a subset of AML, Section 7.5.4). The AML/E programs are prepared and compiled using IBM PCs and the resulting object code downloaded to the robot's dedicated controllers so that the PCs may be disconnected and removed at run-time. An interesting hierarchical control approach has also been adopted in which an IBM Series1 minicomputer is responsible for the overall control of the line. This supervisory computer receives the daily schedule from the mainframe based production planning and control system and then downloads the different control sequences for the different keyboard configurations to the dedicated industrial robot controllers as required and a rapid change over between batches is effected.

The characteristics of flexible automation may be summarised as the ability to manufacture several different products or variations of products with very rapid changeover between products. Thus flexible automation facilitates automatic manufacture of small batches of products, reduction of work-in-progress and the increase of product quality, which in turn enables the manufacturer to respond more quickly to customer demand, reduce product cost and provide better quality goods with longer, better guarantees(25).

3.4 OTHER AUTOMATION TYPES

Before the advent of microprocessors made possible truly flexible automation, standard modules of production equipment had become available, and from these off-the-shelf items production equipment could be constructed using a building block approach, this came be known as
modular automation (26,27).

These standard elements, such as conveyors, part feeders, rotary indexing tables, are available as stock items in ranges of sizes and capacities so that, although the production line remains essentially dedicated, most of the advantages usually associated with a modular approach are obtained. Plant design and hardware costs are reduced, development times are shortened, maintenance is eased, and modules are reusable when products and production lines are altered. This final point gives modular automation a degree of flexibility as, if product changes are small, it may be possible to design the plant so that with the substitution of only a few modules, production of the new item may begin using basically the same production line.

Obviously there is no reason why several different types of automation and manual operations cannot be utilised together. When a combination of manual, hard, soft and modular equipment work on a single product or family of products the plant is described as hybrid automation (28).

As previously discussed truly flexible automation has become available only in recent years and some authors (29) are already discussing the next generation of automatic production equipment. This, it is envisaged, will be fitted with sophisticated sensors through which vision and force data may be fed back. Plant controllers will be able to utilise this information intelligently and will thus be capable of handling a much wider range of products on the same production line and recover from many more errors without manual assistance. However before integration on this scale becomes commonplace many problems associated with interfacing, information transfer and processing, remain to solved.

3.5 SUMMARY OF THE USE OF INDUSTRIAL ROBOTS IN AUTOMATIC PRODUCTION EQUIPMENT

Industrial robots have been employed in both dedicated and flexible production lines. They have found
frequent application as items of hard automation where, although not taking full advantage of all of the industrial robot's capabilities, the plant designers exploit the device's good manipulation abilities and relatively low cost when compared with the alternative of engineering a specially designed manipulator. In Section 3.2 reference was made to the multi-welders used to weld car body sections together and although these machines are still much used, as is manual spot welding, the industrial robot has been applied in large numbers (30) to perform spot welding of car bodies on production lines dedicated to one car model. A second example is provided by Plessey who have used a Unimation PUMA industrial robot for testing complete telephone handset microphones on a line producing several million units annually (31).

The use of industrial robots in flexible automation has been discussed in Section 3.3, and here full use is made of the machines's abilities to change manipulations and sequences of manipulations and other operations very rapidly.

Why, one may ask, is so comparatively little use made of automation especially in Europe? Whatever blame can be attached to management and low levels of capital investment, two fundamentally technical problems exist with the industrial robot, both of which make robot-based production equipment more costly and hence more difficult to justify. Being general purpose machines industrial robots are not optimised in performance to one particular task, thus a single industrial robot is often unable to achieve a required cycle time. Furthermore being designed to possess the range of manipulative abilities necessary to perform a wide variety of tasks they are often over complex for simple operations. These two points are further considered in the following chapter.
4.1 INTRODUCTION

The primary types of automation, have been considered and the role of industrial robots in each has been discussed.

This chapter deals with the conceptual differences between conventional industrial robots and modular devices. It must be emphasised that the ideas involved in the modularity of robotics and the possibilities for building distributed mechanical systems are central to this thesis.

4.2 CONVENTIONAL ROBOTS

This term is used herein to refer to pedestal mounted manipulators which usually demonstrate between 3 and 6 servo-controlled degrees of freedom (dof) arranged in an arm-like sequential kinematic chain. Chapter VI, Section 6.5.1 discusses the more common configurations on which some variations are found. However such manipulators may be mounted on a moving track or overhead gantry, or there may be twin arm versions.

4.3 THE CONCEPTS OF MODULAR ROBOTICS

That there are tasks too demanding for conventional robots is self-evident. However many tasks exist whose low level of complexity make their automation technically well within the capability of even the simpler conventional robots. The reasons that these operations have not been automated is not well-documented but probably lies in cost justification difficulties (32). Although more imaginative justification schemes are emerging (33,34) a fundamental problem exists as the conventional robot is overly sophisticated and consequently unnecessarily costly for a significant number of tasks. If a simpler, less
expensive alternative existed then cost justification would be facilitated and higher levels of automation would result. The commercial availability of some modular robots (35, 36) have implicitly recognised these advantages and more recent cost analyses for assembly applications (37) have endeavoured to quantify them. However it is considered that a number of other, at least equally important, advantages accrue.

An example is the most effective way to examine these points. Figure 4.3.1 shows a conventional 6 degree of freedom (dof) revolute robot (Chapter VI, Section 6.5.1) loading a 2-dimensional pallet. Tasks of this type (38) are essentially 2 dof and frequently require only moderate repeatability (no better than 0.5 mm) when positioning but good cycle times (less than 5 secs per part). Thus the conventional machine illustrated with 6 degrees of freedom and capable of perhaps 0.1mm repeatability represents a considerable overkill, possibly without meeting the speed requirements.

Figure 4.3.2 shows a modular robot tackling the same problem. The designers of this solution needed to purchase only two servo-controlled axes, one end-stop module and an appropriate control system. Matching the performance of the modules, in terms of reach and repeatability, to the task specifications has been possible by selecting from the manufacturer's off-the-shelf range and so system cost has been minimised.

However the use of modular robots does present some difficulties. Engineering effort is required to design and construct a suitable configuration of axes and possibly a controller. Furthermore the very flexibility of the system necessitates the production of an appropriate control program.

4.4 DISTRIBUTED MANIPULATORS

A second modular solution to this task is shown in Figure 4.4.1. This diagram illustrates the concept of the distributed manipulator in which, although not mechanically connected, the servo-controlled modules still act
FIGURE 4.3.1

CONVENTIONAL ROBOT PERFORMING PALLET LOADING
FIGURE 4.3.2 MODULAR ROBOT PERFORMING PALLET LOADING
FIGURE 4.4.1 DISTRIBUTED MANIPULATOR PERFORMING PALLET LOADING
together to perform the pallet loading function. This
distributed manipulator concept may be extended as shown
in Figure 4.4.2. Here ten servo controlled and one end-
stop modular units are unloading pallet delivered parts,
adding attachments, also delivered by pallet, and putting
the final assemblies into further pallets for despatch.
Although perhaps not particularly realistic in detail
this example serves to illustrate some significant
aspects of the application of distributed manipulator
systems. The ten servo-controlled and one end-stop axes
can be thought of as one 14 degree of freedom device,
eleven single degree of freedom machines or any intermed-
iate combination. This suggests the concept of the "Axis
Group" or "Robot Axis Group" in which axes clearly acting
together to achieve a well-defined single objective,
although not necessarily mechanically linked, are thought
of, from the point of view of the controller supervising
the activities of the whole system (Chapter VIII), as a
single manipulator with the appropriate number of degrees
of freedom.

Although not strictly necessary in this example, the
use of a physically distributed manipulator presents the
system designer with the opportunity to improve several
mechanical aspects. Since it may be possible to decouple
some of the links, the problems associated with having to
remotely locate the axis drives, and cumulative joint
errors do not arise or are less pronounced. Thus improved
power, power/weight ratio and stiffness are achievable.

The example in Figure 4.4.2 also shows how modular
systems can be used to achieve the required cycle times
by adding extra robot axis groups at critical or slow
stations. In the application shown the assembly of the
attachment and base requires a considerable amount of
manipulation and is consequently a slow activity compared
to the unload and load functions. The robot axis group
performing this task has therefore been duplicated,
doubling the through-put at this bottle-neck. Such a
course of action is possible with a conventional
industrial robot but will usually be prohibitively
Servo-controlled axes 1
Servo-controlled axes 3
Servo-controlled axes 4
Assembly station 2
Servo-controlled axes 7
Servo-controlled axes 8
End-stop axes 1
Assembly dispatch
Assembly station 1
Transport conveyors
Parts delivery
Servo-controlled axes 2
Servo-controlled axes 5
Servo-controlled axes 6
Servo-controlled axes 9
Servo-controlled axes 10

A possible assignment of axes into robot axis groups (RAGs) is:
- Unload RAG: Servo-controlled axes 7-8-9-10
- 1st Assembly RAG: Servo-controlled axes 5-6
- 2nd Assembly RAG: Servo-controlled axes 3-4
- Dispatch RAG: Servo-controlled axes 1-2
End-stop axes 1

FIGURE 4.4.2 DISTRIBUTED MANIPULATOR PERFORMING SIMPLE ASSEMBLY WITH PARALLEL MANIPULATORS FOR IMPROVED CYCLE TIMES
expensive requiring the installation of an extra complete six degree of freedom machine not just, as in this case, two extra modular units.

This feature of course adds to the potential flexibility of the system, as, should a second attachment operation or some inspection be required by a product enhancement or the introduction of a new product, it may be possible to merely install as many modular axes as required by cycle time and complexity considerations together with suitable jigs and fixings. An existing industrial robot may of course perform the extra operation but this approach would compromise cycle times. Alternatively an extra conventional machine might be employed but the cost of this method would be high.

The modular robot may be thought of as a special case of the distributed manipulator, in fact a robot axis group with zero distribution. Therefore modular robots can be considered a sub-group of the distributed manipulator group and thus throughout this thesis the term "Distributed Manipulator" will be used to refer to both types of machine.

4.5 THE ADVANTAGES OF DISTRIBUTED MANIPULATORS

It is noteworthy that a special machine could also be designed to perform the palletizing and other functions needed in the examples above. However the use of an off-the-shelf family of modules offers several potential advantages:

i) reduced systems engineering,

ii) the modules' proven reliability,

iii) increased flexibility, as through the use and re-programming of a suitable supervisory controller (Chapter VIII), the manipulator could perform functions other than those for which it was originally designed.

Although there are clearly many applications for which the conventional industrial robot or special machine is appropriate, there are situations in which the distributed manipulator will have the advantages of:
i) easier cost justification,
ii) greater flexibility,
iii) easier, less expensive upgrades,
v) enhanced mechanical performance.

4.6 FURTHER IMPLICATIONS OF THE CONCEPTS OF DISTRIBUTED MANIPULATORS

Any machine with a number (from one upwards) of motion elements of any type or actuation may be considered as a general manipulator. These motions may or may not be physically linked and some groups of motions (connected or not) may be considered to perform sub-tasks necessary to the achievement of the machine's overall purpose. For instance most, if not all, hard automation machines may be included within the classification.

The above definition of the general manipulator is in fact merely a restatement of the concept of the distributed manipulator (Section 4.4) and therefore consideration of the distributed manipulator is identical to that of the general manipulator. As was suggested in the previous section, modular robots form a sub-class of the distributed manipulator classification. Also conventional robots (Section 4.2) are in turn a sub-class of the modular robot classification. Figure 4.6.1 illustrates the relationships between these machine groupings. Thus any results, especially related to control, arising from studies of distributed manipulators will also be applicable, usually in simplified or particular forms, to the modular robot and in turn the conventional robot. This point is further explored in Chapter V, Sections 5.5 and 5.6.

A greater control generalisation is also possible. Consider the machines shown in Figures 4.4.2 and 4.6.2(a) which both belong to the same class of machine (distributed or general manipulators), differing only in structure, the control of both machines is therefore conceptually identical. As the conventional robot may, from the control point of view, be considered a special case of
General Manufacturing Machine

Distributed manipulator or General manipulator

Modular robots

Conventional robots
FIGURE 4.6.2  FURTHER CONTROL IMPLICATIONS OF THE CONCEPTS OF DISTRIBUTED MANIPULATORS
the distributed manipulator, its inclusion is legitimate. The equipment shown in Figure 4.6.2(b) which comprises, besides the distributed manipulator of Figure 4.6.2(a), manufacturing machines of several other classes. It can be argued that, although at the detailed level the control of each of these different classes of machine is very different, at higher levels the concept is very similar to that of controlling a distributed manipulator. All machines must be instructed of their required actions and will wish to report their status and the success or otherwise of their actions. Also the sequencing of the activities of the various machines is necessary. It may therefore be postulated that at certain levels the theory and practice of the control of the general manufacturing machine is similar, if not identical to, the control of the distributed manipulator or general manipulator and much of more general significance may be learnt from the study of the control of the distributed manipulator. These ideas are explored further in the following chapter.

4.7 NOTE ON TERMINOLOGY

It is worthwhile to draw together some of the terminology developed in this chapter as it is used throughout this thesis.

**Conventional Robot**
Any pedestal mounted industrial robot which is available in only one or at most a few configurations of kinematic chain.

**Modular Robot**
A robot, constructed from a family of modular elements, and having any number of axes of motion all mechanically connected.

**Distributed Manipulator or General Manipulator**
A robotic device constructed from a family of modular elements but not necessarily all mechanically connected.
Robot Axis Group
A collection of modular elements not necessarily mechanically connected (that is part of a distributed manipulator) operating to achieve a well-defined goal.

General Manufacturing Machine
Any automatic machine used in an automated production facility.

Figure 4.6.1 aids the understanding of the relationships between these terms.
CHAPTER V

CONTROLLER ARCHITECTURES FOR
DISTRIBUTED MANIPULATOR SYSTEMS

5.1 INTRODUCTION
This chapter considers architectures appropriate to the control of the distributed manipulator systems proposed in Chapter IV and in the light of the conclusions drawn, discusses approaches to general manipulator and general manufacturing machine control.

5.2 THE REQUIREMENTS OF A DISTRIBUTED MANIPULATOR CONTROL SYSTEM

5.2.1 Minimisation of Costs
It was argued in the previous chapter that, particularly in low complexity applications areas, a possibly distributed, modular approach can yield a less expensive manipulator as complexity matching would then be feasible. Thus where possible it is appropriate to minimise the cost of the controller.

5.2.2 Control of Distributed Manipulators
The distributed manipulator concept (Chapter IV, Section 4.4) requires that axes neither mechanically connected nor even necessarily physically close be controlled.

5.2.3 The Number of Controlled Axes
A further example in Section 4.4 examined the cycle time advantages accruing from the application of larger numbers of axes to certain tasks. Extrapolation of this idea indicates that the maximum number of axes which may be used in a single application is not necessarily small and therefore a control scheme for a distributed modular system must be able to accommodate not only applications...
using one or two axes but also tasks requiring ten or more axes.

5.2.4 Programming Systems

A detailed discussion of the user interface and programming needs of a distributed modular system is undertaken in Chapter VII; for the present it will suffice to notice that any controller design must potentially offer programming interfaces able to handle applications requiring large and small numbers of axes. Furthermore these programming systems must maintain a high level of user-friendliness despite the obvious sophistication necessitated by the requirement to control a machine in which NOT ONLY IS THE NUMBER OF MOTIONS VARIABLE BUT ALSO THE CONFIGURATION OF THESE MOTIONS IS NOT PREDETERMINED.

5.2.5 Actuation

The actuation of a distributed manipulator system is discussed in Chapter VI, Section 6.6. For the present it will suffice to note that the control system architecture must be able to encompass the use of electrically, hydraulically and/or pneumatically actuated modules.

5.2.6 Fast Program Changeover

Chapter III, Section 3.3 discussed the desirability of automatic production equipment being able to switch between the manufacture of one product and another in very short times. This is also a requirement of distributed manipulator controllers.

5.2.7 Servo-control Function Flexibility

Most fundamentally the controller of a distributed manipulator must be able to safely, and with adequate static and dynamic performance, servo control each axis motion which requires a positioning capability.

5.2.8 General Plant Interface and Control

No industrial robot operates in isolation and the
distributed manipulator controller must therefore be able to interface to, and in some cases control associated equipment such as end-stop modules, tools, sensors, and other machines.

5.2.9 Integration into CIM Systems

The advantages of greater integration of manufacturing companies' computer systems has already been mentioned (Chapter II, Section 2.2) and distributed manipulator controllers will therefore be required to be integrated into CIM systems.

5.2.10 Processor Intensive Activities

Included in this requirement is any function which places a large load on the processor. For instance graphics displays aiding the machine operators or programmers (Chapter VII, Section 7.6) require considerable real-time processing power.

Also, in order that an end effector may trace straight lines and other paths and/or be synchronized with the velocity of moving objects (on a conveyor for example), it is necessary for the controller to be able to solve, possibly in real-time, the kinematic equations representing the distributed manipulator's configuration (39). This is frequently a highly processor intensive task (40) and is potentially even more complex in this case as the manipulator's configuration is variable and expandable.

5.2.11 Enhancements

One of the major advantages of any modular system is that it is at least theoretically possible to add to the range of the modules available as new applications and needs are found and as new discoveries make faster, more accurate, and more intelligent modules feasible. Although this is a difficult target because of its lack of definition, it is clearly highly desirable that the control system architecture selected should be able to encompass as many future developments as can be forseen.
without compromising the cost or adding unnecessarily to the complexity of the controllers currently required.

5.3 POSSIBLE ARCHITECTURES

The possible architectural schemes are listed below and are discussed in the following sections.

i) Single processor with analogue servo-control (Figure 5.3.1)

ii) Single processor (Figure 5.3.2)

iii) Multiprocessor
   a) using processors of equal power (Figure 5.3.3)
   b) using processors of unequal power (Figure 5.3.4)

iv) Distributed multiprocessor (Figure 5.3.5)

5.3.1 Single processor with Analogue Servo-control (Figure 5.3.1)

Analogue servo-controls together with a single, possibly off-the-shelf, programmable controller type user interface (Chapter XIII, 8.4.1) may provide adequate performance at an optimum price.

5.3.2 Single Processor (Figure 5.3.2)

The second possible scheme utilises a single processor possessing sufficient power to provide the required level of servo-control, operator interface, and other functions.

5.3.3 Multiprocessor (Figures 5.3.3 & 5.3.4)

If more processing capability is required than can be provided by a single processor it may be necessary to use a multiprocessor architecture (41). However, using conventional processors, performance gains are not achieved as might be naively expected in proportion to the number of processors used, the problem being the extra load imposed by the communications overhead.

In some multiprocessor systems two or more processors of similar power are used and divide the load equally. Alternatively a number of less powerful processors perform tasks requiring appropriate resources and a
FIGURE 5.3.1  SINGLE PROCESSOR WITH ANALOGUE SERVO-CONTROL
FIGURE 5.3.3  MULTIPROCESSOR ARCHITECTURE USING PROCESSORS OF EQUAL POWER
FIGURE 5.3.4 MULTIPROCESSOR ARCHITECTURE USING PROCESSORS OF UNEQUAL POWER
further machine organises their activities and performs the remaining functions.

A recent development which facilitates the construction of multiprocessor systems is the Transputer, released in the latter part of 1985 by Inmos (42). Essentially this device is a conventional microprocessor with four built-in high speed serial communication interfaces and an appropriately designed instruction set which facilitates the design of very high power multiprocessor systems which do not suffer the problems of inter-processor communications overhead.

5.3.4 Distributed Multiprocessor (Figure 5.3.5)

A multiprocessor system is described as being "distributed" if all processors are not physically close so that their inter-connections have to leave an electrically well protected environment and therefore their inter-communication requires more sophisticated error checking hardware and software.

5.4 THE CHOICE OF A DISTRIBUTED MULTIPROCESSOR ARCHITECTURE

5.4.1 Single Processor with Analogue Servo-control

To achieve the performance required of today's manipulators it is necessary to use modern servo-control approaches whose implementation is only cost effective, or even possible, with digital, computer-based hardware. Thus the approach outlined in Section 5.3.1 may be discarded.

5.4.2 Single Processor

Consider firstly the resource requirements of the servo-control function. These programs are typically short in length and manipulate only small quantities of data and thus require but little ROM or RAM memory. However their essential function is to control the static and dynamic behaviour of mechanisms operating at speeds often in excess of 1 metre/second and frequently being
Figure 5.3.5 Distributed Multi-Processor Architecture
required to position without overshoot to better than 0.1mm. The controller is therefore required to execute completely the servo-control programs of the axes in a few milliseconds. If other than small amounts of 16-bit or floating point arithmetic are involved in the control algorithm then an 8-bit device may well have insufficient capacity, and even 16-bit processors may be able to handle less than 6 axes (Chapter IX, Section 9.8).

Because of the processor intensive nature of the servo-control function, even for fast 32-bit processors there will be a maximum number of servo-controlled axes that can be handled by a single processor without recourse to complex and costly interface electronics to reduce processor load. Even in this case only limited processor time would be available for a good, user-friendly programming interface and other requirements (Section 5.2). Also the scope for possible future enhancements would be severely restricted.

Thus although the single processor architecture does represent a useful approach to the control of systems with only a limited number of axes (a conventional robot for example) it has limited upgradability and does not solve the general distributed manipulator control problem.

5.4.3 Multiprocessor

Multiprocessor approaches using processors of equal power have been employed successfully in commercially available conventional robots (43), the ill-effects of the inter-processor communication overhead being minimised by the use of a high speed parallel bus (Chapter VI, Section 6.7.2).

Such a bus-based multiprocessor approach could control a wide range of manipulator configurations, but whilst offering a considerable measure of flexibility and upgradability, it would be unnecessarily complex and consequently costly when used to control just one or two axes. The fundamental problem with this approach is that it requires that the size of the complete task is known.
and therefore the amount of computing power is calculable. This is not the case with a distributed manipulator system in which the number of axes requiring servo-positioning and their configuration can change, even after installation.

A common distributed multiprocessor approach of the second type suggested in Section 5.3.3 is to devote a separate processor to the control of each axis (44,45) together with an extra processor acting as system supervisor. This approach has been facilitated by the development of inexpensive single chip microcomputers offering memory, input/output, timer, and interrupt handling (Chapter VI, Section 6.7.2). Other divisions of labour between processors have been considered and implemented (46). For instance where processor price and performance have necessitated, the servo-control of more than one axis has been allocated to each processor (47).

An architecture of this type using a supervisory computer (SC) and one or more servo-control computers (SCC) is clearly appropriate as the amount of processing power available increases with the servo-control requirement, provided that servo-control computers are purchased together with servo-controlled axes. The question of the exact number of axes to be servo-controlled by one SCC will for the present remain unanswered but each SCC will have only sufficient resources (processing power, memory and I/O) to perform its servo-control functions and any essential communications overhead.

Should the supervisory computers be required to do large numbers of floating point calculations and other time consuming operations associated with trajectory planning and co-ordinate transformations in real-time (Section 5.2.10), it may be necessary to employ mathematical co-processors or floating point accelerator units (48). Alternatively two or more ordinary processors may be used in the supervisory computer and in extreme cases co-processors may again be added. Thus the supervisory computer itself might require a multiprocessor architecture.
5.4.4 Distributed Multiprocessor

Section 5.2.2 required that motion elements, in some applications separated by several metres, be controlled. The question therefore arises as to whether the units providing the servo-control function should be located together at some convenient central position or each one physically close to the manipulator axis it controls. If all were gathered centrally the analogue control signals from the servo-amplifier (Chapter X) to the servo-valve (Chapter IX, Section 9.3) would have to be transmitted over long cable runs. The potential for signal attenuation and interference in the electrically-noisy environment often encountered on the shop-floor makes this an inappropriate approach and favours an architecture in which at least the servo-control units can be distributed to positions adjacent to the axes they control. This is a restatement of the reasons why digital signalling using error checking and correcting hardware and software (Chapter XII, Section 12.5.3) is usually preferred to analogue for fast, accurate communications.

Thus the discussion in this and the previous section strongly favours a distributed multiprocessor approach.

5.4.5 Hierarchy of Distributed Manipulator Control

Figure 5.4.2 shows how the control of the distributed manipulator originally illustrated in Figure 4.6.2(a) (and reproduced here for convenience as Figure 5.4.1) might be achieved.

The distributed multiprocessor architecture of Sections 5.3.4 and 5.4.4 is used with a noteworthy supplement. The single dof motion elements in the top right-hand corner of Figure 5.4.1 are required to act together so that the relative motions of the object and tool they manipulate are always controlled, that is contoured rather than just point-to-point motion is necessary. To achieve these contoured motions considerable 16-bit arithmetic must be performed and to off load this processor intensive task from the supervisor (SC), it being beyond the capabilities of the servo-control
FIGURE 5.4.1 A DISTRIBUTED MANIPULATOR SYSTEM
computers (SCC) (Section 5.4.3), an extra computer (called here a "contouring controller") is installed.

Section 5.4.3 proposed that the supervisor (SC) organise the activities of the servo-control computers (SCCs) and thus implied a control hierarchy of two levels. The scheme illustrated in Figure 5.4.2 has a more developed, 3 level hierarchy as shown in Figure 5.4.3. This example demonstrates the power and flexibility of the distributed multiprocessor approach as more levels, higher or intermediate, may be employed without necessarily disturbing the architectural concept or the operation of existing systems.

Also in Figure 5.4.2, six of the motion elements are contained in a conventional robot for which the purpose-built controller is utilised. This controller may itself have an internal multiprocessor architecture and, as shown in Figure 5.4.3, it also fits into the control hierarchy.

A hierarchical distributed multiprocessor architectural framework is thus most appropriate to the control of the general or distributed manipulator.

Although this hierarchy implies that the SCCs are subservient to the SC, no media access control mechanism has been specified.

5.5 REQUIREMENTS OF THE GENERAL MANUFACTURING MACHINE CONTROLLER

Chapter IV, Section 4.6 introduced a discussion of the general manufacturing machine controller whose requirements are briefly listed below:

i) Minimisation of cost
ii) Control of physically distributed machines
iii) Control of few or many machines
iv) Programming system
v) Fast program changeover
vi) Different actuation methods
vii) Servo-control capability and flexibility
viii) General plant interface and control
ix) Integration into CIM systems
x) Processor intensive activities
xi) Upgradability.

It will be observed that in essence these are very similar to the requirements of the distributed manipulator or controller with some small differences. Points ii), iii), and iv) now deal with a wider class of machine and vii) is concerned with not only the control of motion elements but also with other systems, atmospheric controls for instance. Although usually performed by local machine controllers, the potential for general plant interface and control must remain, as must the ability to perform processor intensive activities, although the nature of these could now be very different (works scheduling for instance).

5.6 ARCHITECTURES FOR THE GENERAL MANUFACTURING MACHINE CONTROLLER

As the requirements of the General Manufacturing Machine controller are so similar to those of the distributed manipulator then the line of reasoning of Sections 5.3 and 5.4 can again be applied with a similar result, that a hierarchical distributed multiprocessor approach is most appropriate. There are however some differences:

i) The servo control computer equivalent may be required to be much more powerful and to undertake considerably different functions

ii) The interprocessor communications links may need to be different and will perhaps not be homogeneous. That is different approaches being used in various situations as demanded by cost/performance trade-off.

Figure 5.6.2 illustrates these comments, showing a possible architecture for the control of the machine in Figure 4.6.2(b) (reproduced here for convenience as Figure 5.6.1).
The following points are also noteworthy:

i) Multiple distributed machines of different types are now controlled

ii) Controlled machines now include a distributed manipulator controller

iii) Different communication systems are used as appropriate

iv) The distributed manipulator controller now regards the General Manufacturing Machine controller as its link to CIM

v) Distributed manipulator and pneumatic press controllers interface directly to their own associated equipment

vi) Programming interface now has a very broad specification

vii) Closed-loop control of dryer is undertaken directly by the General Manufacturing Machine controller

viii) The conveyor controller, for instance, could be regarded as the "servo-control computer" (SCC) of the General Manufacturing Machine controller

ix) Levels have been added to the hierarchical control structure (Figure 5.6.3).

5.7 PNEUMATICALLY-ACTUATED DISTRIBUTED MANIPULATOR SYSTEM

5.7.1 Limitation of Objectives

Important though the considerations of the control of the general manufacturing machine and general or distributed manipulator are, limitations of time, manpower, and other resources prevented their development in the very general terms of this and the previous chapter. As the concepts of the distributed manipulator were central, this was the most important area for application and consequent study. In order that such a system could be installed in an industrial environment it was necessary to construct a family of modules and associated control system elements.

Application being the major objective, completion of the design and build phase was vital and thus the size of
the family of modules and controllers needed to be well within resource limitations. Also these products were required to have a high likelihood of industrial acceptance. However, at all times during these practical developments, their role as a part of a greater whole was fully considered.

5.7.2 **Pneumatic Actuation**

It was decided that the distributed manipulator system to be studied would be based on a family of pneumatically actuated, servo-controlled modules. The reasons for this decision being that:

i) Research in the Dept. of Engineering Production at Loughborough University had indicated that, in laboratory conditions the use of micro-electronics could result in achieving industrially attractive price/performance from servo-controlled pneumatics. The relative merits of pneumatic actuation are discussed in depth in Chapter VI, Section 6.6.

ii) The industrial collaborators, Martonair Ltd (a company with considerable expertise and experience in the manufacture and industrial application of pneumatic components) wished, with the support of SERC funding to produce novel programmable positioning products. This involvement offered several potential benefits:

a) Martonair planned to produce a range of pneumatically actuated, end-stop modular units which might offer an expedient route to the design and manufacture of servo-controlled modules.

b) With a well-established industrial presence Martonair could contribute requirements knowledge and at the formative stages were able to support intuitive feelings that a system aimed at comparatively less complex applications would find most ready industrial acceptance.

c) The prototype products resulting from this work would have a potential route to production versions and thus industrial application, so
important for the study of the distributed manipulator concept.

5.7.3 Controller Architecture

As an, albeit restricted, distributed manipulator system, the reasoning of the earlier parts of this chapter which favoured a distributed hierarchical multi-processor design is valid.

5.7.4 Enhancements

As a practical development of a larger concept the need to consider system upgrade and enhancement was of importance. For instance the use of electrically driven motion elements was anticipated which would permit contoured motions whereas first generation pneumatically actuated modules would only necessitate a point-to-point positioning capability.

5.8 NOTE ON TERMINOLOGY

It is worthwhile to draw together some of the terminology and abbreviations defined in this chapter and used henceforth in this work.

i) Servo-control Computer: SCC
A physically small, technically simple, and relatively low cost microprocessor-based device performing the servo-control function for at most three axes.

ii) Single Axis Controller: SAC
An SCC controlling only one axis.

iii) Supervisory Computer, Supervisor: SC
A probably more complex device designed to organise the activities of a number (possibly tens) of SCCs and/or SACs together with any associated tools, jigs, fixings, sensors etc.

5.9 CONTROL SYSTEM ARCHITECTURE DESIGN SUMMARY

The design of a controller for a distributed
A manipulator system should be based on a distributed hierarchical multiprocessor approach in which one computer of minimum complexity and cost is allocated to the servo-control of at most a few axes and that another computer system (Chapter VIII) should supervise and organise the activities of the servo-control computers.

In order to progress towards a pneumatically actuated distributed manipulator system a wide range of technologies needed to be gathered together. Fortunately some of these can be drawn from conventional robotics, the relevant areas of which are reviewed in the following 2 chapters, whilst Chapters VIII to XI discuss the subjects which necessitated research and development within this project.
6.1 INTRODUCTION

Much of the technology used in the conventional robot is relevant to distributed modular systems and the fundamentals thereof are thus worthy of greater consideration. This chapter and the next review conventional robotics with particular reference to distributed manipulators.

6.2 AN HISTORICAL PERSPECTIVE

The continuing industrial revolution gave the development of hydraulic and pneumatic power, electric motors, fluidics, thermionic valves and transistors. These together with the advance in servo-control technology inspired by World War II led to the development of, amongst other automatic machines, the first commercially available industrial robot by the Unimation Corporation in 1961 (49). This device was hydraulically actuated and controlled by electronic hardware constructed from discrete components. Its major areas of application were in dirty and hazardous environments (49). By the early 1970s other industrial robots were appearing. All of these early devices were fast and powerful, their major drawback lay not in their servo control systems which achieved acceptable performance, but in their programming systems which needed highly-skilled staff for their use and required of the order of days of programming to achieve even the relatively simple manufacturing tasks which their less than 100 step memories could contain.

Machines with electric d.c. motors began to appear in the 1970s and thus removed some of the problems associated with hydraulics (including noise, physical bulk, and the need for regular maintenance) which were previously limiting the application of industrial robots.
This era also saw the introduction of LSI (large scale integration) technology and the birth of microprocessor families, resulting in much decreased cost of processing power and, in industrial robotics, greatly improved programming systems.

6.3 THE STATE-OF-THE-ART

As the servo-controlled industrial robot developed, so too did its simple, frequently pneumatically actuated, "end-stop" relative (50). These machines can be configured with up to six axes but are only able to halt the motion of each axis at either of the axis extremities and in some cases at a limited number of selectable intermediate mechanical stops. As such these machines can be considered to offer limited mechanical flexibility making them most suitable for low complexity positioning applications or where job changes are infrequent (a job change possibly involving manual set-up of end-stops).

The simple control requirements and relatively low cost of these end-stop machines has ensured their rapid development and application, and indeed it is these devices that account for the very high Japanese robot population figures (51).

Returning to servo-control devices, the continuing reduction in the price of processing power and memory capability has resulted in powerful, frequently multi-processor controllers (Section 5.4.3) which have facilitated high speed/high repeatability manipulators with the capability of performing axis transformations in real-time. This ability to change between co-ordinate frames has enabled industrial robots having configurations not aligned with any convenient co-ordinate system to move their end-effectors in arbitrarily positioned and orientated cartesian frames, often with controlled velocity.

Programming systems, dealt with more fully in the following chapter, have also benefited, and machines with high-level language programming front-ends are now available, enabling, at least in principle, much greater use of sensor feedback which in turn facilitates the use of
industrial robots in applications of greater complexity. Advances in the mechanical aspects of industrial robots are discussed in Section 6.5

6.4 APPLICATION AREAS

The first robots to find industrial application were used primarily in spot welding and, to a lesser extent, paint spraying. Load/unload and materials handling applications were also common, particularly in hazardous or unpleasant environments where safety legislation facilitated cost justification (49).

The greater sophistication of the current generation of industrial robots has naturally resulted in a greater range of applications, notably assembly and arc welding (52).

Even a cursory examination shows that industrial robots are applied only where their technical capabilities and ease of programming closely match requirements. For instance spot welding requires good repeatability, manoeuvrability, and fairly high load capability, but does not need straight line interpolation or the use of large quantities of sensor feedback. The specifications of industrial robots such as the Unimation 2000 series (53) are well matched to these requirements and also have programming systems which can be readily used by skilled manual spot welders so that these workers can carry out robot programming with a minimum of re-training and a good transference of human skills is obtained. These comments, particularly the last, apply equally to robot paint spraying.

Assembly robots are high speed/high repeatability machines with considerable contouring and sensor feedback capabilities. Their sophisticated programming facilities require computer programming skills not easily or quickly acquired by manual assemblers and thus the use of assembly robots involves not only the cost of the complex machine itself and associated jigs, fixings, tools and sensors, but also the cost of the skilled programming staff.
6.5 MECHANICAL DESIGN

Of all the disciplines of industrial robotics this aspect has seen the least striking developments, arguably because the design of the actuation and mechanical systems of early industrial robots was fundamentally sound.

6.5.1 For Conventional Robots

Conventional robots are usually designed to be fixed in one location and perform their tasks with an arm having usually from four to six axes, each axis consisting of one link and one joint. Robots are usually designed in one of four configurations usually referred to as Cartesian, Spherical, Cylindrical and Revolute. These four arrangements each have their own advantages and disadvantages in terms of working envelope, maximum reach, payload, speed, repeatability, and actuation system (Section 6.6). These have been discussed in considerable depth by other authors (49,54).

Although six axes are required to achieve maximum flexibility some machines have less. The SCARA (Selective Compliance Assembly Robot Arm) configuration (55) which has become common in recent years being the prime example. These devices are aimed primarily at assembly tasks where studies have indicated that their selective compliance feature is well matched to the problems of the very common vertical insertion activity (56).

6.5.2 For Distributed Manipulators

Whilst it is clearly possible to arrange modular systems with configurations similar to those of conventional robots, some aspects of the static and dynamic performance of such systems cannot be expected to achieve the same levels as those of conventional machines. Indeed it is to remove the built in constraints in terms of reach, load, and configuration placed on the applier of industrial robots by the very design of conventional robots that the distributed modular approach is in part
intended. As has been suggested (Chapter IV, Sections 4.3 and 4.4), any number of axes may be employed in any physical arrangement with the length and load carrying capacity of any one axis only being dependent on cost, resolution, and accuracy or repeatability requirements. Also, for simple manipulation problems, only the required number of axes need be used and paid for, whereas for the conventional industrial robot the mechanical complexity and consequently cost is predetermined. This is not to suggest that the conventional machine does not have its place but endeavours to indicate that there are frequently applications where a well-designed distributed system is to be preferred in terms of either cost, mechanical suitability or both.

6.6 ACTUATION SYSTEMS

Three basic power sources are used for industrial robot actuation, electric, hydraulic, and pneumatic, each having advantages and disadvantages (57,58). In the first of the following two sections the characteristics of each type of system are outlined and considered with reference to conventional robots. The second section discusses their application in distributed modular systems.

6.6.1 For Conventional Robots

The electric direct current (dc) motor is now the most common robot actuator, although advances in dc brushless (59) and other areas (60) have brought greater application of these devices. The electric dc motor offers good, readily achievable control of speed and position and the wealth of different types now available has meant that specifications in terms of torque, speed, acceleration and efficiency can usually be met. Electric motors of all types being essentially quiet and clean in operation are environmentally acceptable and indeed brushless forms can be used in flammable environments. Although generally efficient, electric motors of all types together with their controllers and power amplifiers, usually cost upwards of several hundred pounds and
are not therefore conducive to the production of low cost machines. The power to weight ratio of electric drives and their associated transmission and feedback systems tends to be low compared to the hydraulic and pneumatic systems discussed below. Whilst causing no difficulty in lower limbs, this presents the industrial robot designer with the problem of how to actuate the limbs near to the end-effector. This has led to many ingenious transmission systems involving rods, steel belts, and chains, all of which, although solving the primary problem, usually led to some deterioration in positioning performance. Electric motor driven systems are also not usually employed in the actuation of high load machines (greater than 90 kg) as, although motors capable of this performance are available, their size and weight usually aggravate the aforementioned problem.

It is in precisely these high load applications that hydraulics is usually employed. Working at high pressures compact hydraulic cylinders are capable of generating considerable force and with a remote primary power source (the hydraulic pump) the actuator weight required to be moved by the arm is kept within acceptable bounds. Also, being an almost constant volume medium with a well-developed servo-valve technology (61), good control of position and velocity can be achieved. However the nonlinearities (which can vary significantly with time) and actual compressibility of the working medium ultimately limit factors such as the deadband, repeatability, linearity, hysteresis, and response time. Another contrast between electric and hydraulic actuation lies in the essentially rotary nature of electric motors and the linear nature of hydraulic cylinders. Whilst electrically actuated linear drives and hydraulic rotary motors exist they usually require increased bulk and weight or have other drawbacks limiting their use in industrial robotics.

Pneumatic cylinders also conveniently provide linear actuation and a number of the above comments concerning hydraulic actuation apply equally to pneumatic. Although
considerable interest within the research community has centred on the modelling and design of pneumatic motion control systems (Chapter IX, Section 9.4), until recently the lack of suitable quantities of appropriately priced processing power has meant that the complexities of the resulting highly non-linear control equations has prevented potential appliers of such theories achieving servo controlled positioning of payloads in general industrial application areas. Consequently the qualities of pneumatic actuation, such as relatively low cost, good power to weight ratio, simplicity, intrinsic safety and cleanliness have only been widely utilised in systems requiring end-stop motions (Section 6.3). The advents of the transistor, integrated circuit (IC), and microprocessor have largely removed this obstacle and whilst end-stop axes are still frequently appropriate, servo-controlled pneumatically actuated systems are now a practical proposition (Chapter IX).

6.6.2 For Distributed Manipulators

Clearly any of the three forms of actuation discussed in the previous section could provide a distributed manipulator system with the desired flexibility. Some reasons for the choice of pneumatic actuation for the motion elements discussed in this thesis have already been given (Chapter V, Section 5.7.2), more technical arguments are herein outlined.

The higher cost of the electric prime-mover, the motor, compared to that of the pneumatic cylinder, the most common pneumatic prime-mover, is a clear argument against an electrically actuated system. Further intuitive feelings suggested that modular units having linear rather than rotary motion would find greater numbers of simpler applications. For systems having many coupled axes the weight of the electric motors actuating the last joints in the manipulator chain would adversely affect the performance of those at the beginning. Although these effects are very difficult to quantify in the general case, being inevitably functions of application and
dynamic conditions, it is probable that the performance gains usually associated with the use of electric devices would not be fully achieved.

Hydraulic actuation shares all of the advantages mentioned in the previous paragraph with compressed air and in distributed systems aimed at high load applications it undoubtedly has advantages. However, for lower load applications, pneumatic actuation offers a less expensive system with comparable speed and positioning performance.

It should also be noted that another potential advantage of the distributed manipulator approach is that, should some high performance or high load motions be required in a particular situation, appropriate electrically or hydraulically actuated modules might be applied.

Although the cost of the pneumatic prime-mover and control components (valves) is low it is not usual for pneumatics to be associated with high energy efficiency and therefore low running costs (62).

6.7 CONTROLLER DESIGN

In contrast to developments in industrial robot mechanical and actuation systems' design, which have been slow but sure, progress in all aspects of controller design has been as rapid as the advances in the microprocessor-based hardware and software from which industrial robot controllers are now constructed. Early controllers were based on analogue designs (the Unimate 2000 series (53) for instance) which, although necessarily restricting the complexity and sophistication of the servo-control approaches adopted, still gave satisfactory static and dynamic performance for many applications. Unfortunately the user interfaces or programming systems were limited (Section 6.2) and severely restricted the application of these early industrial robots.

The development of 8-bit microprocessors in the mid-1970s changed this situation. While the real-time performance of these devices was not usually sufficient to
improve greatly on analogue servo-control performance, their ability to facilitate the construction of user-friendly programming systems greatly increased the ease and consequently the range of application of the industrial robot.

6.7.1 Control Systems Architecture

This subject has previously been discussed in Chapter V, Sections 5.3 and 5.4.

6.7.2 Electronic Hardware

The design of robot controllers has been considerably influenced by developments in all aspects of the electronics which form the controller hardware.

Advances in the design of Large Scale Integration (LSI) and Very Large Scale Integration (VLSI) devices together with the computer aided software tools which

![Diagram of silicon fabrication progress]

FIGURE 6.7.1 PROGRESS IN SILICON FABRICATION
assist these processes (63,64) have led to large increases in the numbers of gates available on a single chip (Figure 6.7.1).

These developments have resulted in an almost yearly doubling of the storage capacity of RAM and ROM devices. Greater integration of microprocessor and peripheral devices has also occurred so that facilities previously available only from a multi chip set (eg Intel 8086, 8284, 8253, 8237, 8259) are now provided on a single device (eg Intel 80186 (65)). As new microprocessors and their associated IC families have been introduced the wisdom with which designers have used the resources have been debated(66). For instance the true worth of Intel's guaranteed upwards software compatibility has been contrasted with the arguably better internal architecture and more elegant instruction set that has been achieved by the designers of the Motorola 68000 microprocessor family (67).

Mathematical co-processors and single chip microcomputers have been mentioned in Chapter V, Sections 5.4.3 and 5.4.4. The former processors provide hardware for the performance of, amongst other tasks, floating point, 64-bit integer and Binary Coded Decimal (BCD) arithmetic, and certain functions such as square root, tangent, arctangent, logarithm and exponent. In this way the general purpose processor is relieved of these highly time consuming calculations.

Single chip microcomputers (68,69) have been manufactured in response to the desire to put microprocessor-based computing power and decision making capabilities into an even wider range of application areas. These devices which contain, besides the CPU, limited amounts of memory (RAM and in some cases ROM/EPROM) and input/output, have made possible the design of small and inexpensive circuits that provide large amounts of control capability for such machines as video recorders, televisions, washing machines, motor cars, laboratory instruments, and, not least, industrial robot controllers. Perhaps of the currently available single chip
computers the most powerful is the 16 bit Intel 8096 (70).

An ever increasing range of other special purpose ICs is now available and includes local area network (LAN) interfaces (Intel 82586 and 82501) (71), controllers for various electric motors (72), and specialised sensor interfaces (eg the incremental encoder interface outlined in Section 11.7).

All of the IC's previously mentioned in this section are fabricated in either PMOS or NMOS (73). These devices interface at TTL levels and although not individually high consumers of power in mains terms, can present energy consumption problems when battery-backed circuits are being designed. This and the heat dissipation of large numbers of TTL ICs has caused the development of devices fabricated in Complementary Metal Oxide Semiconductor (CMOS) material. The traditional CMOS disadvantages of slow operation and high manufacturing cost have now been largely overcome and most common IC families, memories, and microprocessor families are available in CMOS (74).

Advances have also taken place in the design and manufacture of printed circuit boards (PCBs) with many computer aids now available to the PCB designers and manufacturers (75,76).

PCB design tasks are normally highly labour intensive and, requiring skilled personnel, are costly. Thus for small batch PCB production the minimisation of these costs is essential. Unfortunately appropriate computer aided facilities require large sophisticated software packages and very considerable processing power. Packages of this type are available for some of the larger personal computers but tend to be slow in execution and frequently omit more sophisticated facilities.

If an industrial robot controller requires special circuits and consequently PCBs to be constructed then the aforementioned problems associated with their manufacture cannot be avoided. However a large range of "standard" PCBs are available from an almost equally wide range of
suppliers and it is possible to design microprocessor-based systems around off-the-shelf boards, many of which are designed to fit standard backplane buses. A detailed discussion of the range of standard boards available is beyond the scope of this thesis but processor, memory, I/O, and combination boards of many variations are manufactured for every IC maker's hardware. Details of inter-connection buses are also outside this discussion but are rather less numerous than off-the-shelf PCBs, some standarisation having been forced by the leading electronics manufacturers (77), but internationally recognised standards have only recently come into being (78) and there remain several well-established de facto bus defintions, Q-bus(79), G-64 bus(80), VME bus(81), Multibus(82) and others primarily aimed at 32-bit applications (83).

Whilst this route to the production of industrial robot controllers avoids the costs of PCB design and manufacture, it has a number of major disadvantages. Firstly each off-the-shelf PCB has a price ranging from a hundred to a few thousand pounds and secondly, despite the wide selection, certain combinations of memory, processor, and I/O may not be achievable without considerable redundancy and thus unnecessary cost. Also if it is necessary only to have a controller of the minimum complexity, and thus a price below that of a single "standard" board, PCB build may be unavoidable. Furthermore off-the-shelf PCBs are normally available in one of a range of "standard" sizes and thus small or odd shaped packages may not be possible.

Surface mount technology which, whilst finding application in the manufacture of high volume analogue circuits, has only comparatively recently been used in the production of digital equipment (84). Using these devices IC package sizes are considerably reduced and consequently more ICs can be placed on a single board. Further space is gained as both sides of the board can be used. Also the number of holes through the board is greatly reduced, this being of special significance to
the designers of multi-layer PCBs. The size of surface mounted components is so small and the placing requirements so precise that generally assembly is only possible by automatic means. This fact has meant that as yet only relatively high volume products contain surface mounted components but as the range of devices available in surface mount packages becomes greater and the assembly, layout, and soldering techniques better understood this technology will have an increasingly significant role to play in the construction of industrial robot controllers.

6.7.3 Software

As the needs of industrial robot controllers have become larger, more complex, and more sophisticated their software content has followed similar trends. Fortunately software engineering methodologies and tools have also advanced.

The benefits of top-down software design and structured software are well understood (86,87). These techniques are now widely used. For example MASCOT (Modular Approach to Systems Construction, Operation, and Test) (88) philosophy is much employed by the Ministry of Defence in Great Britain. Approaches of this type are essential for the management of large software projects so that all-encompassing specification documents can be translated into reasonably sized modules which can be comprehended in detail by one programmer and thus coded. Furthermore a proper top-down design technique facilitates the testing of such modules and groups of modules and enables their coalition into a complete software package without major module interfacing problems. This is achieved by forcing designers and programmers to use well-defined module interfaces. For instance the MASCOT technique uses the ACP (Activities Channels Pools) diagram for overall design, the Activities representing the executable modules of code, the Pools the structure of the database, and the Channels the module interfaces. During this work early controller software designs
utilised a MASCOT approach (89).

Although such sophisticated methodologies may not be necessary for the programs used in smaller and simpler industrial robot controllers, the modularity intrinsic to such an approach remains, if not vital, at least most useful. The production of modular software involves coding a larger program as a series of smaller parts, each module containing a procedure or procedures or other code performing separately identifiable functions. Each module is at most a few pages in length and is sufficiently simple to be easily comprehended by one person. Furthermore each module is only allowed to interact with any other module via a well-defined interface. These measures ensure that not only are the modules readily assembled into a whole system, but also that it is easy to change certain functions in the system by merely modifying the modules responsible for that function. This work may be carried out after the system has started operation even by programmers who do not fully comprehend the functioning of the system as a whole, provided, of course, that the module interfacing rules are observed.

Many comprehensive top-down design philosophies have associated with them whole computer-aided facilities for their implementation (90,91). Others, including MASCOT and ADA have operating systems and languages designed to assist their use. The MASCOT operating system, known as the "Kernel", is strictly a multitasking real-time executive which enables the modules of code to be run concurrently in such a way that the programmer does not have to be aware of the design of the kernel. The inter-module communication via channels is also handled by the kernel. Similar facilities exist in all comparable multi-tasking systems (eg iRMX (92)) and all to some extent suffer from the problem of unpredictable response times in certain circumstances. This may be overcome by the use of interrupts provided that the number of stimuli requiring immediate attention is small but will always be a problem when attempting to design systems with
"adequate" response times to large numbers of signals.

If however the size of the required programs is not sufficient to necessitate the use of such a methodology, then there may be benefits in designing the software to run on a "bare" machine. In this situation no assistance is available to the applications programmer in terms of how each module is executed and in what order, or how they are to interface to each other. This has the benefit of not requiring the memory overhead of such an operating system or executive and the system hardware may be designed with a minimum configuration at minimum cost. Furthermore the applications programmer may be able to design a system with much more predictable response times without having to understand in detail the workings of a complex executive. However such code as is required to organise the sequence of execution of the modules and their inter-communication will have to be written and tested by the applications programmer, the size, elapsed time and consequently the cost of such an exercise being dictated by the sophistication required by these functions.

A large variety of languages are now available for most development systems and microprocessors (93) although if a MASCOT or ADA type of overall approach is used the applications programmer may be forced to use, or may find many advantages in using one particular language. For the production of large pieces of software the use of a compiled high-level language such as PASCAL (94) will probably be preferred as this type of language speeds the writing of code and, with the aid of a good compiler, facilitates fault finding. Unfortunately this high-level, compiled language approach tends to produce executable machine code that is optimised in neither speed of execution nor memory occupancy. The use of an interpretive high-level language for very large systems may reduce memory occupancy but is not to be recommended for truly real-time systems as interpreted programs have slow execution speeds. Thus for small industrial robot controllers where software is to be run on a minimum
hardware configuration it is likely that software may have to be written in assembly language, this having the advantage of giving the programmer full control over the executable machine code produced and enabling fast, compact programs to be written. Assembly language is not suitable for large software systems since it is slow to write and therefore requires large amounts of skilled and thus expensive manpower for its production.

In recent years practical applications of Artificial Intelligence (AI) (95) have been seen and the so-called AI languages such as PROLOG (96) and LISP (97), which enable non-sequential programs to be written, have been more widely used. The most common application being in so-called "Expert Systems" (98) where an expert in a particular field inputs details of his/her knowledge and experience into a database. If a non-expert now interrogates the system about a set of new circumstances then the system can apply the same deductive processes as would have been used by the expert to come to conclusions regarding the circumstances. Furthermore such an expert system can use its inference capabilities to find new rules and deductive processes. It is this "learning" capacity which makes the application of AI so interesting and potentially powerful.

Examples of expert systems currently in existence include those for medical diagnosis (99) and the analysis of geological data (100). Some applications in manufacturing engineering have also been proposed (101).

Although no commercial industrial robot controllers yet utilise this artificial intelligence approach, it is clearly a most powerful technique for the future production of software for sophisticated controllers which are required to make decisions and plan end-effector trajectories based on vision data, force feedback, and inputs from other advanced sensors.

6.7.4 Control Systems for Distributed Manipulators

The work involved in the design and construction of the distributed manipulator control system which the
remainder of this thesis describes has made full use of the hardware and software methodologies discussed above where this has been both timely and economically/technically appropriate.
CHAPTER VII

ROBOT PROGRAMMING SYSTEMS

7.1 INTRODUCTION

A review of the technologies of conventional robotics and their applicability to distributed manipulator systems begun in Chapter VI is herein continued with a discussion of robot programming systems in which area there has been rapid progress since the introduction of the industrial robot, this being largely attributable to the reduced cost of processing power.

The purpose of robot programming systems and the methods appropriate to limited stop robots are outlined. Teaching and language-based approaches are then reviewed. Off-line and "Task Orientated" systems are outlined and the chapter concludes by considering the programming of distributed manipulators.

7.2 PURPOSE OF ROBOT PROGRAMMING

All robots are flexible manipulators capable of being programmed to carry out a large range of tasks. In order that an individual robot can become specialised in the performance of a single task a means of programming the robot is required. In much the same way a computer, a flexible data manipulator, is programmed to handle one specific data set.

However only rarely is a robot acting in isolation and usually there will be other equipment which must be interacted with, including:

i) the robot end-effector

ii) computers which transmit and receive production information, including robot commands and new programs

iii) other equipment including machine tools and robots working together in the execution of the overall task

iv) tools, jigs and fixings in the robot's workplace
workstation sensors of varying complexity, from simple binary switches to vision systems. The programming system must allow the specification of the robot's interaction with all of these devices. Furthermore it is important that interference with production is minimised whilst reprogramming takes place and that systems are not overly complex or sophisticated in order that:

i) programming is as simple as possible to understand and perform

ii) all programming costs including labour are minimised

iii) programs are easy to change, upgrade and generally (in the software sense) maintain.

7.3 FOR END-STOP AND MULTI-STOP ROBOTS

Robots of this type, being essentially binary in nature, require only simple control systems capable of binary I/O and appropriate decision making. Control systems can thus utilise pneumatic logic, relays, hard-wired electronics, and computer based systems including Programmable Logic Controllers (PLCs) (Chapter VIII). Pneumatic logic, relays, and hard-wired electronic controllers possess desirable features but share the problem of holding the control program in the physical arrangement of the components and thus being inflexible.

Computer based systems provide the advantage of relatively easy reprogrammability. However language-based facilities, whilst offering a large range of programming features are not readily supported by normal production plant staff and thus require specialists for their use and maintenance. As many of these features are not normally required for the control of non-servoed robots, the ladder diagram programming which is frequently offered by PLCs is appropriate as this is readily used, understood, and maintained by normal plant electrical staff. Because of this PLCs are a common choice for the control and programming of end-stop industrial robots. A further advantage of computer based devices is that communication with other computer systems and thus
integration into CIM systems (Chapter VIII, Section 8.2.4) is facilitated.

7.4 PROGRAMMING BY TEACHING

By the early 1980s two distinct types of robot, known as "point-to-point" and "continuous path", had emerged. Both having similiar mechanical and control structures, their major difference was and is in programming approach. It is noteworthy that the range of applications of each type matches well the features offered by their respective programming system, illustrating that conventional robot application is frequently governed by programming method as much as by any other parameter.

Point-to-point machines utilise a "teach by pendant" approach. Here the programmer moves each robot axis until the robot takes up the required spatial positions using a hand-held "teach-box" or "pendant" on which are mounted suitable controls. Some systems also enable the movement of the robot end-effector in real-world or other x-y-z co-ordinate frames. When a desired position is reached the pendant may be used to instruct the controller to store that position, commonly in joint coordinates, but possibly in real-world co-ordinates. Interaction with other equipment may also be programmed using the pendant.

Continuous path devices use a "teach by showing" method in which a human operator guides the robot tool around the path required. Whilst in "teaching" mode the controller samples (typically every 10-100 milliseconds) and stores the position of each robot joint. When a program has been taught the machine can be switched to "run" mode in which the joint positions are retrieved from the controller's memory at the same frequency as they were stored and then despatched to each of the axis servo control systems so that the robot retraces the path originally taught and at the same speed.
7.4.1 Teach by Pendant Features and Applications

This type of system enables programs to be taught which require the robot to return to a number of set points with high repeatability (typically 0.1mm or better) set points. However the control of the motion of the robot end-effector between these set points is not normally easily achieved, the avoidance of obstacles often necessitating the insertion of extra set points. Simple interactions with other equipment, including robot tools, are usually readily achievable although interfacing to analogue devices is often more difficult if not impossible. Programming constructs such as loops and conditional jumps are achievable, although facilities for data structuring and subroutine calls are not easily used.

Being conceptually relatively simple, programming robots of this type can be performed by staff skilled in the task to which the robot is to be applied and a good transfer of human skills may be achieved. Editing and debugging can be complex and time consuming especially for longer programs.

The features of the teach by pendant approach are well matched to the requirements of the applications of point-to-point conventional robots (small number of good repeatability set points, no control of motion of the end effector between these set points). Thus included in the range of applications are pick and place, load/unload, and spot welding, which is the most common conventional robot application (30). Simple, highly jigged assembly can also be performed.

7.4.2 Teach by Showing Features and Applications

The teach by showing approach enables the conventional robot to follow geometrically complex paths with moderate repeatability (approximately 1mm). Large amounts of memory are required to store the taught path and thus magnetic tape is a frequently chosen medium. Small amounts of interaction with other plant, particularly the robot tools and parts present sensors can also be
inserted either at program time or in some cases as a post-programming edit. However editing the taught path is normally not possible and even small modifications will necessitate complete reprogramming.

Good transfer of human skills to the robot are achieved and of course the robot reliably repeats the taught contour so that the user always obtains the performance of "the best worker on his best day".

Applications requiring complex continuous, but not high precision, contours and little interaction with other equipment are clearly most appropriate to machines programmed by this method, the most common being paint spraying. Other tasks undertaken by machines programmed by this method include glue and wax spreading, and sand blasting.

7.5 LANGUAGE-BASED APPROACHES

Since the late 1970s there has been considerable academic and industrial interest in programming robots by language based methods. This section looks at the reasons for this interest, briefly examines its history, outlines the current trends and briefly discusses standardisation.

7.5.1 Reasons for Language-based Approaches

Teaching approaches to robot programming possess several desirable features, particularly simplicity and good transfer of human skills, but are, in many respects severely limited for more complex applications. These limitations include:

i) difficulties of incorporating sensor feedback and of specifying how decisions should be made based on this sensor derived information

ii) difficulty of specifying mathematically and changing (at plan or run time) the manipulator's trajectory

iii) difficulty of specifying interactions with other intelligent machines including other shop floor equipment (machine tools, other robots etc) and supervisory computers

iv) need to have access to the robot whilst programming
(i.e. be on-line) and thus loss of production whilst programming.

Most attempts to solve these problems and to make robot programming user friendly in more complex situations have used language based approaches, analogous to computer programming.

7.5.2 Requirements

A robot programming language would ideally resolve the above difficulties and possess at least some of the following features:

i) structuring of program and data to at least the same level as computer programming languages such as PASCAL or ADA

ii) much reduced or zero down time due to reprogramming

iii) standardisation so that personnel need only to be trained in the use of one system.

iv) interpretation and compilation so that the advantages of both may be obtained.

v) ability to specify concurrent robot control and other programs

vi) Operating systems to execute such programs.

7.5.3 History

The amount of work in this field is vast. A detailed coverage is beyond the scope of this thesis and the reader is referred to one of the comprehensive review articles for a wider coverage (102,103).

The work on robot programming languages began in the late 1960s and was concerned mainly with specifying manipulator trajectories (Section 7.5.1, item (ii)). Researchers at the Stanford Artificial Intelligence Laboratory (SAIL) in the USA firstly defined a low level language WAVE (104), and recognising the limitations of such languages developed AL (105) based on the computer high level language ALGOL. The IBM Research Centre (Yorktown Heights USA) was also active in the field developing languages such as ML (106) and AUTOPASS (107).

All of the activity until the mid 1970s had
primarily experimental objectives and as the systems all required considerable computer power no true industrial applications resulted. The first programming language commercially available, VAL (108), whose development was based on the Stanford research was released in 1978 by Unimation Inc. for use with their PUMA robot.

By the early 1980s other commercial and academic organisations in several countries were active. MAL (109) was developed at the Milan Polytechnic and Robex (110) in Germany. In France a version of a language originally designed at MIT in the USA known as LAMA-S (111) was implemented. DEA in Turin developed the HELP language for use with their multi-arm PRAGMA cartesian robots (112). Eastern Europe countries also showed considerable interest in the subject (113, 114).

Most of this work concentrated on the problems of specifying manipulator trajectories (eg WAVE, AL) or where robot configurations eased this problem, on providing easy to use BASIC-like languages (eg HELP, ML). No single development even approached the criteria for an ideal system (Section 7.5.2). However useful theoretical and practical experience was gained. Perhaps most widely used industrially was VAL, which was recognised (115) as providing good facilities for the programming of manipulator motion trajectories, but whose BASIC-like approach necessitated computer science trained staff without providing the programming constructs essential for the truly user friendly incorporation of the sensor feedback vital in complex applications.

Some systems provided for multi-arm configurations (HELP) with concurrent program operation but the need for further parallelism to permit on-line trajectory planning and other real-time features were identified but not implemented.

7.5.4 Current Trends

Current languages which more closely approximate the ideal of Section 7.5.2 use one of three distinct approaches. Recognising the deficiencies of VAL, Uni-
formation Inc. developed VAL-II (45) which, like its predecessors WAVE, AL, and VAL, is a specially designed language and operating system aimed at industrial robot control. The VAL-II system retains its good trajectory control features while offering in addition a PASCAL-like structured language, good sensor integration facilities, parallel execution of motion control, trajectory planning and background process control program, and also purports to address the wider needs of manufacturing applications including integration into CIM systems (116). The PASCAL-like LM (117) developed at Grenoble represents a similar start-from-scratch approach as does the rather less sophisticated BASIC-like AR-SMART now available with the Reflex Robot (118).

Contrastingly AML (119), the result of IBM's continuing efforts, is a newly defined language targeted at general manufacturing applications but containing instructions appropriate to the control of manipulators.

Paul, at Purdue University, continued his earlier work with a sophisticated motion descriptor language PAL (120), and has more recently incorporated the concepts therein developed into RCCL (121), a set of robot motion control procedures written in 'C' which can be incorporated into any 'C' program running under a Unix operating system. Other authors have suggested similar additions to other suitable high-level languages such as Concurrent PASCAL (122) and ADA (123).

It is claimed (124) that this approach offers all of the advantages of a well-known and developed language and operating system including I/O facilities, concurrency, good documentation, readily available trained staff together with the full range of data manipulation and structuring features of an established computer programming language.

7.5.5 Standardisation

The previous sections have mentioned some of the large range of robot programming systems developed in the last 20 years. Despite the advantages these systems
offer, few have found industrial application. Although the reasons for this are complex, one undoubtedly contributory factor is that each robot programming language offered is different and consequently requires specially trained staff for its use.

The need for standardisation is well understood (125) and much work has been done, some by international consortiums (126). Standard high and low level languages have been proposed (127, 128) as have standard interfaces between such languages and the robot controllers (129). However there is much disagreement as to the best approach. Also robot manufacturers show no inclination to move away from their own de facto standards and as a result there is little likelihood of standards emerging in even the medium term.

7.6 OFF-LINE PROGRAMMING

As batch sizes decrease the frequency of robot reprogramming increases and it becomes more desirable to program off-line as much as possible. This is in part possible (set points need to be taught on-line) with current language based methods such as VAL-II and AML. However the loss of the real-world as a visualisation aid presents severe difficulties. Systems are now available (130, 131) which utilise state-of-the-art solid modelling and graphics techniques in attempting to simulate the real-world. Unfortunately these products are somewhat slow in operation (that is not real-time), expensive to purchase, and may always be limited by their necessarily 2-D representations of a 3-D world.

7.7 TASK ORIENTATED METHODS

All of the robot programming methods discussed in this chapter have been of the "explicit" type in which the programming is in terms of exactly which motions are required for the desired task, for example:

Move(pickup)
Grip
Departs(pickup,0,0,10)
Approach(hole,0,0,10)
Moves(hole)
Ungrip....... 

An equivalent but "implicit" program might read:

Put peg in hole

The power and simplicity (to the programmer) of such an approach, which also offers the advantage of being off-line (Section 7.6) have made them the subject of considerable research interest.

A complete treatment being beyond the scope of this thesis the reader is referred to Ambler's paper (132) for a review of the issues involved. It should however be noted that few even experimental task orientated systems exist, one such being RAPT (133) developed by the University of Edinburgh, and that much of the research activity has been of a theoretical nature (134,135).

If work in this area meets with eventual success it would seem reasonable to argue that extrapolation of these ideas might lead to even less explicit programming methods. Indeed if a product were designed using CAD systems and the dimensions stored in the CAD database it might be possible, using artificial intelligence (AI) techniques, for the off-line programming system to work out for itself the actions required for the manufacture of that product.

7.8 DISTRIBUTED MANIPULATOR PROGRAMMING

Chapter IV introduced the concept of distributed manipulators, the essential features of which were:

i) any number of axes

ii) manipulator mechanical structure motion dynamics undefined at commissioning

iii) use of robot axis groups (Section 4.4)

iv) manipulator could be reconfigured after initial installation by the addition of extra axes and/or
the redistribution of axes at plant rebuild.

Chapter V discussed the design of distributed manipulator controllers, identified the need for a programming system (Section 5.2) and selected a distributed multiprocessor architecture. The programming or user interface function was allocated to the supervisory controller (Section 5.4.3).

The distributed manipulator programming system requirements extra to those of the conventional manipulator can be identified as:

1) capability of encompassing manipulators with any number of axes in any mechanical configuration
2) allowing the user to reconfigure the control and programming system to suite the configuration
3) allowing the user to define and program robot axis groups
4) permitting the communication between and concurrent execution of such robot axis group programs.

Clearly all of the discussion in the preceding sections is relevant, with the possible exception of the task orientated methods which appear somewhat over the horizon at present. Thus a distributed manipulator programming system comprising a well-structured language and set point teaching facility would appear as technically achievable and meeting user requirements.

Although some systems have addressed the problems of multiple arms (e.g. HELP) no work has been performed which tackles the special problems of programming a manipulator whose structure is undefined at manufacture and may be altered several times after initial installation. However the underlying theory of VAL-II, originally due to the Stanford group, is, at least in principle capable of generalisation and extension to manipulators with any number of degrees of freedom in any configuration.

Until such efforts, which will be considerable, are made it is not possible to predict the problems likely to be encountered, except to anticipate the size and complexity of any resulting computational systems. Due to resource constraints no further work in this area has
been performed within this project and it was therefore not feasible to foresee exactly the requirements the design of the programming system would place on the distributed manipulator controller. All that could be said was that no aspect of the controller design should impede any future developments in programming system. The choice of a distributed multiprocessor architecture is, from this point of view advantageous as it is feasible to select new supervisory systems (hardware and/or software) as developments take place without effecting installed SCCs/SACs provided that interface standards are maintained.

It is noteworthy that as any programming system developed for distributed manipulators must handle devices with any number of axes in any configuration, including their synchronisation, then it will also be appropriate to conventional robots and any general manipulator (Chapter IV, Section 4.6) and it is thus possible to see how developments in distributed manipulator programming might have implications for the designers, makers, and users of many flexible automation devices and may even aid the efforts towards robot programming system standards.
CHAPTER VIII

SUPERVISORY CONTROL

8.1 INTRODUCTION

Chapter V discussed the selection of a distributed hierarchical multiprocessor architectural framework and identified the need for a supervisory computer system which is now considered. The terminology defined in Chapter V and summarised in Section 5.8 will be used extensively throughout this chapter.

8.2 SUPERVISORY CONTROLLER REQUIREMENTS

8.2.1 Supervisor to SCC/SAC Communications

Communications between supervisor and SCC/SAC are necessary so that, for instance, positions to be controlled to (set points) can be transmitted from supervisor to SCCs/SACs and "in-position" signals sent in the opposite direction. Furthermore a SCC/SAC may be required to transmit its actual position to the supervisor. As information concerning position requires at least one 16-bit number (Chapter XI) the need to send and receive digital data of at least 16-bit length to any number of SCCs/SACs (from 1 upwards) is identified.

8.2.2 Programming and Set-point Teaching

These requirements have already been discussed in detail in Section 7.8 where the major points were:

i) the need for a language based interface
ii) capability of handling any number of axes
iii) the need for user reconfigurability
iv) the need to define and execute concurrently robot axis group programs.

8.2.3 General Plant Interface and Control Functions

All industrial robot workstations contain not only
the robot but also other equipment, for instance conveyors, binary valves, electric motors, and simple optical, proximity and other sensory devices. The integrated operation of this peripheral equipment and the industrial robot(s) themselves requires organisation and sequencing. In the case of workstations formed from distributed manipulators this responsibility must be assumed by the supervisory controller.

Thus the supervisor must provide general plant interfaces to analogue, digital, serial and parallel devices, and make decisions based on both mathematical and logic functions.

8.2.4 Computer Integrated Manufacture (CIM)

The advantages of greater integration of manufacturing companies' computer systems performing a variety of manufacturing functions has already been outlined (Section 2.2). The subject of shop floor computer networking has received considerable recent attention during which many approaches have been considered, including MAP (Manufacturing Applications Protocol), originally developed in the USA by General Motors and now the subject of considerable worldwide standardisation efforts. This subject is beyond the scope of this thesis and the reader is referred to Gascoigne's paper (136) for a detailed discussion. However, it can be concluded that supervisory computers are required to connect to both current and future inter-computer communications systems and to be a part of the company's hierarchical factory control system.

8.2.5 Upgradability

A supervisory computer system will probably be completed in a piecemeal fashion and will even then be subject to technological advances. Thus a supervisor must be upgraded, preferably without needing to completely redesign and rebuild.
8.3 SUPERVISOR HARDWARE
The usual three routes to the development of electronic hardware are available:

i) build from chip level
ii) build from board level
iii) purchase and possibly customise off-the-shelf hardware.

As (iii) will usually be less expensive and will always require less manpower, it is be considered in the first instance. To this end the following sections review the current approaches to the control of flexible automated production plant.

8.4 CURRENT APPROACHES TO THE CONTROL OF FLEXIBLE AUTOMATED PRODUCTION PLANT
A large range of computers are currently in use for the control of manufacturing plant and these can be classified within the categories shown in Figure 8.4.1. Controlled environment computers are machines which need to be installed in situations where the electrical, mechanical and atmospheric conditions are more closely controlled than is usually possible on the factory floor. As it is anticipated that supervisory controllers will normally be situated on the shop floor these machines are considered unsatisfactory.

Ruggedised computers are machines which are designed to withstand "harsh" environments in which:

i) vibrations and other mechanical disturbances are present

ii) electrical and electromagnetic interference and noise are present

iii) conditions of extreme temperature and/or humidity occur

iv) water, oil, and other chemicals may be present.

The difference between Programmable Logic Controllers (PLCs) and industrial computers is most readily explained by briefly reviewing their history
8.4.1 Programmable Logic Controllers (PLCs)

The relay control of automated plant, much used in the post war period, offered the ability to make decisions based on the logic functions such as AND, OR and NOR. The advent of the transistor and subsequent development of the integrated circuit (IC) facilitated greater logic control although relays were still common particularly where power switching was required. Planning and documentation of all systems still employed the so-called ladder diagrams originally used for controllers containing only relays.

The further development of the IC and particularly the microprocessor made available the potential for more complex levels of logic control, as well as offering the possibilities of lower cost and reprogrammability. Clearly general purpose equipment could be designed and manufactured that would be able to be programmed to control many different types of plant. The staff responsible for installing and maintaining these controllers were accustomed to relay systems with their ladder diagram planning and documentation, and it was appropri-
ate to select for this new generation of reprogrammable controllers a programming interface familiar to the intended users, namely ladder diagrams. Manufacturers such as Allen-Bradley therefore offered their earlier Programmable Logic Controllers (PLCs), as this type of general purpose control equipment was called, with ladder diagram programming.

As IC technology progressed, PLCs offered more memory and therefore larger and more complex programs, together with a range of features enhanced beyond the simple logic operators and including the arithmetic functions and serial interfaces. For these and trademark reasons many suppliers retitled their products "Programmable Controllers" (PCs). However to avoid confusion with the ubiquitous Personal Computer the term "PLC" will be used henceforth to describe these microprocessor based devices with other than language-like programming systems, most frequently ladder diagrams (137,138).

8.4.2 Industrial Computers

More widespread familiarity with computer programming languages, particularly BASIC, reduced the need for staff re-training and has meant that industrially packaged computers with a language programming interface have become more acceptable and a wide range of these devices, referred to henceforth as industrial computers, is now available (139). Examples of industrial computers include ruggedised versions of the IBM PC and DEC MicroVAX.

Thus this thesis distinguishes between these 2 types of computer based industrial controllers, Programmable Logic Controllers with ladder diagram programming, and Industrial Computers with language programming.

8.4.3 Ladder Diagram and Language Based Programming

Although a detailed comparison of ladder diagram and language programming is beyond the scope of this work an outline of the advantages and disadvantages of each is necessary. The examples in this discussion use the Pascal computer language (94) and Allen-Bradley Mini PLC-2
a) PROGRAM conditions;
VAR A,B,C,D,E : BOOLEAN;
BEGIN
WHILE TRUE DO
BEGIN
IF (A and B) THEN C:=TRUE;
IF (D or (NOT E)) THEN F:=TRUE;
END
END.

b) A B C

D F

E

FIGURE 8.4.2 SIMPLE CONDITION MONITORING PROGRAMS
WITH BINARY I/O:
a) PASCAL, b) LADDER DIAGRAM
ladder programming (140).

The Pascal program shown in Figure 8.4.2(a) activates output C when both A and B are "on". Output F is activated when D is "on" or E is "off". All inputs and outputs are binary. The ladder program in Figure 8.4.2(b) performs the same function. Clearly for this simple task both methods are equally applicable.

In the Pascal program of Figure 8.4.3(a) binary output E is activated when the sum of analogue inputs A and B multiplied by analogue input C is less than analogue input D. Analogue output H is set equal to analogue input I when boolean input D is "on" or E is "off". The functionally equivalent ladder program (Figure 8.4.3(b)) is more complex, not easy to read, nor in any sense self-documenting. Thus, although ladder programming is much used for manipulating arithmetic functions and handling analogue I/O, language programming is much less cumbersome in these circumstances.

The example programs in the section above all deal with the events which are to occur under certain input conditions regardless of the order in which the conditions arise. In many cases industrial robot workstations require that actions take place in strict sequences. Whilst sequence programs can be thought of in terms of a series of input conditions and therefore programmed in ladder diagrams, sequences can be more easily achieved with a programming language (141).

8.4.4 Industrial Process Controllers

The third category of ruggedised computer in Figure 8.4.1 is the industrial process controller. Although functionally these controllers can often be identified with high levels of sophistication, generally they are used to control long time constant processes and are not directly an appropriate choice of control element for distributed manipulator systems. There is none-the-less an overlap in functionality between many commercial PLCs and industrial process controllers, where, for both types, sequencing facilities (usually associated with the
a) PROGRAM ana_conditions;

    TYPE one_to_twenty = 1..20;
    VAR A, B, C, D, H, I : one_to_twenty;
        E, F, G : BOOLEAN;

    BEGIN
    WHILE TRUE DO
    BEGIN
        IF ((A+B)*C)<D THEN E:=TRUE;
        IF (F or (NOT G)) THEN H:=I;
    END
    END.

b) | A   B   M  |
    | 001 002 010 |

   +-----------------------+-
   | 010 003 011 012 |   |
   +-----------------------+-

   +-----------------------+-
   | 012 004 005 |   |
   +-----------------------+-

   +-----------------------+-
   | F   P  |
   +-----------------------+-

   +-----------------------+-
   | 006 013 |   |
   +-----------------------+-

   +-----------------------+-
   | G   |
   +-----------------------+-

   +-----------------------+-
   | 005 |   |
   +-----------------------+-

   +-----------------------+-
   | P   I   H   |
   +-----------------------+-

   +-----------------------+-
   | 013 007 006 |   |
   +-----------------------+-

---

FIGURE 8.4.3 CONDITION MONITORING PROGRAMS WITH ANALOGUE I/O
   a)PASCAL, b)LADDER DIAGRAM

91
former) and continuous-loop control functions (typically identified with the later) can be found.

8.5 PROGRAMMABLE LOGIC CONTROLLERS (PLCs) AND INDUSTRIAL COMPUTERS AS SUPERVISORS

Section 8.2 identified and discussed the requirements of Supervisory Controllers. Section 8.4 outlined the range of currently available machines with a view to their use as supervisory controllers from which it is clear that the types classified as PLCs and Industrial Computers are the primary candidates.

8.5.1 PLC Supervisory Controllers

Some limitations of PLC ladder diagram programming systems, particularly with reference to manipulation of analogue values and sequence control, have been discussed (Section 8.4.3). Many PLCs have a serial I/O capability and have the potential to be linked to networked communications systems, but these features are not easily handled by ladder diagram programming. Thus PLCs, although capable of acting as supervisory controllers, are not ideal.

However their continuing popularity in manufacturing industry eventually necessitated the design of a special interface to facilitate their use as supervisory controllers (Chapter XV, Section 15.4). Furthermore the particular needs of the collaborating company, Martonair, were directed towards ensuring the commercial availability of pneumatic motion controllers which could be supervised by off-the-shelf PLCs.

8.5.2 Industrial Computers

Although build from the board or chip level could give an optimum hardware configuration, the timescales involved (both with respect to the author's contribution and the collaborating company's required launch dates) dictated that the benefits of such an approach were not likely to be commensurate with extra effort involved. Thus the use of industrial computers, which offer a
technically acceptable off-the-shelf route to supervisory hardware, was established as the best choice of supervisory computer.

8.6 DEVELOPMENT OF INDUSTRIAL COMPUTER BASED SUPERVISORS

A complete supervisory computer must offer to the user a full range of tools for writing, compiling/interpreting, and executing the programs controlling distributed manipulators together with a set point teaching facility (Section 8.2.2). To achieve this in reasonable timescales the use of off-the-shelf software will be of considerable benefit, particularly with reference to a real-time multi-tasking operating system and standard tools such as text editors.

The provision of language programming and teaching facilities should be a phased development with the initial stages providing a simple teach by pendant system and suitable command primitives (eg Move_to_postion).

Phase two would provide a VALII-like structured programming language. The development of a new language and associated compiler is not advised for reasons of non-standardisation and the resources required.

The adaptation of existing robot programming languages is obviously attractive in terms of convenience and conforming to de facto industrial standards but presents several problems. Even if the necessary source code, together with associated support hardware and software could be procured, the problems associated with changing databases might have such far reaching repercussions within the code that the very convenience of this route might be jeopardised.

The recommended route is that discussed in Section 7.5.4 where a library of motion control and other procedures is written in an existing language and may be incorporated into supervisor control programs produced by the user in that same language. Succeeding phases should provide for upgrade and the inclusion of improving technologies, for example the use of graphics programming.
CHAPTER IX

THE EVOLUTION OF PNEUMATIC MOTION CONTROL

9.1 INTRODUCTION

Chapter V discussed the selection of a distributed hierarchical multiprocessor architectural framework to organise and control the operation of distributed manipulators and identified the primary function of the SAC (Single Axis Controller) or SCC (Servo Control Computer) as being the servo-controlled movement of an axis or axes.

The principal elements necessary to achieve motion control of the pneumatic servo-drives utilised in this project are shown in Figure 9.1.1. It should be clearly stated that, at the commencement of this study, there were very few examples of the industrial use of pneumatic servo systems. The use of air as a working fluid presents major problems in control engineering as it's compressibility and inherent time delays are unpredictable and non-linear. The methodologies employed in this project to overcome these limitations utilise microprocessor based controls to achieve the point-to-point positioning of a family of modules which vary significantly with respect to a number of mechanical characteristics. Thus the object of this aspect of the project was to evolve a robust real-time control algorithm.

Three distinct areas of activity were involved. Martonair (UK) Ltd were responsible for much pneumo-mechanical design and construction, including that of the servo-valve briefly discussed in Section 9.3. Modelling and control engineering work was also necessary and is outlined in Section 9.4. The author was primarily responsible for the implementation of control algorithms together with software to evaluate their performance. It is these aspects of the study that this chapter details.
Figure 8.1.1
Principal Elements of Pneumatic Servo-Control Systems
9.2 EARLY SERVO-CONTROL APPROACHES

Early studies centred on the use of on-off valving. This work was terminated when a servo-valve (Section 9.3) was developed by Martonair, but buying on-off valving promising performance characteristics had been achieved (repeatability<1mm). However, in order to achieve adequate dynamic characteristics (underdamped response being generally unacceptable in robotic/machine control applications) it was necessary to achieve positioning at low speeds (<0.1m/sec) for short moves (<0.1m). The other major disadvantages of these approaches using on/off valving, particularly in terms of commercial implementation were:

i) the complexity of the necessary mechanical and pneumatic control circuit elements led to relatively high cost drives

ii) the control strategy employed was generally specific to a particular mechanical configuration and/or load, that is the control algorithms derived were not particularly robust.

9.3 PNEUMATIC PROPORTIONAL SERVO-VALVES

The relative merits offered by different types of servo-valve have been previously reviewed (142). The advantages of using a servo-valve approach in general and of the Martonair servo-valve in particular (Plate 9.3.1), can be summarised as:

i) simpler and less expensive mechanical and pneumatic circuit elements can be employed

ii) improved control algorithm robustness and flexibility can be achieved and the resulting drives are simpler and easier to manufacture.

The operation of the Martonair servo-valve is relatively simple. Increased current in the solenoid results in a greater force on the spool, this in turn working against the return spring and diverting, in a proportional manner, air from one port to the other. A solenoid rated at 24v, 1A, and 24.6 ohms with a suitably matched spring gave the response characteristics of
Plate 9.3.1: Servo-control Valve
of Figure 9.3.1, which show the valves narrow active range. This characteristic also serves to show that the valve is fairly non-linear in this range, while outside the active range, valve saturation occurs. It is also evident that the valve exhibits considerable hysteresis.

By incorporating air passageways into the valve body, pilot air pressures can act on each end of the spool in such a way as to increasingly oppose the force producing the spool movement. Such a pressure compensated valve is of higher cost but can also be identified with a larger active range (Figure 9.3.2). It is this second valve which is referred to in Chapter X, Section 10.4.1, as the 0-1A active range valve. However, resulting studies showed that the simpler valve performed adequately so that the higher cost, pressure compensated valve was developed no further.

9.4 PNEUMATIC SERVO-CONTROL MODELLING

Generally the servo control methodologies employed were experimentally based with a comprehensive evaluation facility designed and produced to allow a variety of performance criteria to be assessed for each algorithm devised. Based on earlier work (142,143) a theoretical approach was attempted (144) utilising small perturbation linearised models. However the models devised were not found to be satisfactory due to the highly non-linear nature of the elements involved, so that only broad indications of control methods could be gained from a theoretical approach (145).

9.5 CONTROL STRATEGY TESTING AND EVALUATION

The complexity inherent in the motion control of pneumatic drives (Section 9.1) necessitated a microprocessor based system of considerable power, flexibility, and ease of use for the prototyping and evaluation of the control strategies devised. It should be emphasised that the possibility of accomplishing pneumatic motion control for general industrial use has only become possible through the availability of low cost LSI elements.

98
a) Proportional servo-valve pressure/current characteristic
for increasing current (inlet pressure = 5 bar)

Pressure (Bar)

Port A

Port B

Increasing Current (mA)

b) Proportional servo-valve pressure/current characteristic
for decreasing current (inlet pressure = 5 bar)

Pressure (Bar)

Port A

Port B

Decreasing Current (mA)

FIGURE 9.3.1 SERVO - VALVE CHARACTERISTICS
FIGURE 9.3.2 0–1A ACTIVE RANGE PRESSURE COMPENSATED SERVO-VALVE CHARACTERISTICS

Note: Inlet pressure = 5 bar
9.5.1 Standard Evaluation Criteria

So that different control strategies, implementations, modules, control hardware, and pneumatic arrangements could be compared it was necessary to establish a set of performance criteria. This activity was of particular importance to Martonair who clearly wished to monitor project progress, make suggestions and assess the stage at which this research effort had evolved commercially exploitable algorithms.

Furthermore, there was the need, when the commercialisation stage was reached, to obtain statistically valid performance data thereby allowing product descriptions to be developed which could be of value to potential industrial users.

Such a set of static performance measures were agreed, based largely on BS 4151:1967, "Method of Evaluating Pneumatic Valve Positioners with input signal of 3 to 15 psi (gauge)", these comprised:

i) average departure from linearity
ii) hysteresis
iii) repeatability
iv) accuracy
v) dead band
vi) step resolution.

As dynamic performance is less relevant to pneumatic positioners, BS 4151:1967 was not so useful in suggesting dynamic performance criteria. Thus the dynamic performance measures employed were based on those commonly used in control systems, as listed below:

i) maximum velocity
ii) average positioning velocity
iii) total time to position
iv) time lag before response
v) time to reach peak response
vi) maximum overshoot.

9.5.2 Hardware Design and Choice of Processor

Prototype control strategies were implemented and evaluated using a 16-bit Texas Instruments TMS 9900.
microprocessor (146). Software was written in TMS 9900 assembler (147) using a Texas Instruments FS990/4 Microprocessor Development System (148) running under the TXDS operating system (149). The evolution of the I/O interfaces are considered in depth by Chapters X and XI of this thesis.

The TMS 9900 processor was selected for this prototyping work for the following reasons:

i) its 16-bit architecture could achieve the required processing performance,

ii) I/O interfaces were readily built and used from the program

iii) comprehensive hardware and software development facilities were already in existence at Loughborough

iv) Loughborough staff possessed significant expertise in the use of these facilities.

It should be noted that none of these points precluded the use of other processors in the commercial implementations of the control strategies evolved and Chapter XIII discusses at length the choice of processor for subsequent commercialisation.

9.5.3 Performance Assessment Software

Evaluation software was written, tested and run on TMS 9900 based single board microcomputers (150). This software demonstrated modularity and allowed the integration of real-time control algorithms corresponding to each control strategy devised, thus evaluation testing using the criteria outlined in Section 9.5.1. was facilitated. The features of the evaluation software are summarised below:

i) the modification of test variables such as parameters relating to control algorithms, and any specified sequence of axis set points defining the target motion positions

ii) the ability to conduct tests in either "single step" or "automatic" mode. "Single step" mode returning control to the operator after each set point is achieved, whereas in "automatic" mode a complete set
A point sequence (as previously defined) is executed an operator-defined number of times (typically 10-20 as required to achieve statistically valid testing) before further operator input is required.

iii) Data collection during an experimental run, usually in "automatic" mode.

iv) Assistance in the statistical computation of the values of some of the performance measures.

The TMS 9900 assembly language instruction set possesses only integer arithmetic commands and no library of real-valued arithmetic functions was readily available. Several problems are associated with calculations using integer arithmetic: mainly associated with truncation errors and overflow. For instance, in obtaining variance values (standard deviation squared) it is necessary to sum squared errors, which in experimental systems of this type were frequently large. It is possible to perform these calculations using the value of the \((N-1)\)th variance to arrive at the \(N\)th and thus avoid the build up of large numbers. Unfortunately, obtaining a variance value requires a division which, in integer arithmetic, produces a truncation error which in turn introduces an error into any succeeding calculations based on that value. It can be shown that in this case the error builds up as successive variance values are calculated from one another and thus this method is not appropriate here. No true solution was found to this problem and it was only possible to ensure that, when programming calculations such as variance, any possible sources of overflow error were trapped and the operator informed. The value of the evaluation software was not seriously impaired as overflows only occurred when the results of a particular experimental run were conspicuously poor.

The performance assessment software package is driven by simple menus, the levels of which are shown in Figure 9.5.1.

The software control algorithm implementation ran as the real-time control interrupt service routine with the
Note: All menus contain a return to previous menu option

FIGURE 9.5.1 SERVO-CONTROL PERFORMANCE ASSESSMENT SOFTWARE PACKAGE HIGH-LEVEL MENUS
FIGURE 9.5.2 SERVO – CONTROL PERFORMANCE ASSESSMENT SOFTWARE PACKAGE STRUCTURE
remaining software operations as a background task. Communication between the two fundamental elements being via common data areas. This situation is illustrated by Figure 9.5.2.

9.6 CONTROL STRATEGY IMPLEMENTATION

The essence of the control strategy (145) embodied within the commercial pneumatic servo-controllers (SCCs/SACs) (Chapters XIV and XV) is shown in Figure 9.6.1. This complete strategy can be characterised as including control algorithm scheduling, gain scheduling, and the use of minor loop compensation. The software implementing this strategy formed a clock interrupt service routine. Thus a real-time control loop was executed at an interrupt frequency which could be changed via the operator menus (Section 9.5.3) but was typically 2 milliseconds.

However, Figure 9.6.1 is somewhat oversimplified, control algorithm scheduling being based on the value of the position error (=Yd-Y). For large values only the proportional term was used, whilst for other values of error this term was replaced by a value from a look-up table. In order that these ranges might be changed and the relative effects of the terms altered for experimental purposes, and to suit various load/move length conditions, a considerable amount of arithmetic manipulation was called for. Whilst the mathematical manipulations involved were not complex, great care was necessary in the implementation to ensure that the TMS 9900's integer arithmetic produced well-conditioned results without overflow of intermediate or final values.

9.7 PERFORMANCE CHARACTERISTICS

The results of comprehensive testing of the commercialised control algorithm for pneumatic motion control are summarised by Moore et al (151) where it is concluded that a significant improvement in dynamic response and a dramatic increase in drive stiffness (resulting in good accuracy and repeatability) had been achieved. In a
Set Point (Yd) → Proportional term $K_p(Y_d - Y)$ → Include velocity term $K_p(Y_d - Y) + K_v \dot{Y}$ → Include Acceleration term $K_p(Y_d - Y) + K_v \dot{Y} + K_a \ddot{Y}$ → Valve and Actuator

1. Calculate Velocity ($\dot{Y}$)
2. Calculate Acceleration ($\ddot{Y}$)
3. Position Feedback ($Y$)

MINOR LOOP COMPENSATION

Figure 96.1
concurrent research study alternative approaches had led to the reporting of equally beneficial effects (152). However deficiencies remain, mainly associated with variable time lags (due to the influence of friction), particularly for short moves (<0.1m), and unacceptable overshoots with change in load. Work at Loughborough is in progress which promises to solve many of these problems (153).

9.8 ALGORITHM EXECUTION SPEED

In order to assess the processing requirements of the servo-control algorithm two methods were available:

i) direct measurement of the execution time

ii) calculation of the execution time.

The first of these was not considered fruitful as it would have necessitated exhaustive testing of the associated numerous routes through the code and in any case the relative frequency of execution of each was very dependent on both software and physical parameters (eg gain values, control regions, move length, load). As the algorithm contains no loops it was possible to obtain a worst case estimate of execution time by simply summing the time to execute every instruction using manufacturer's data (146). This exercise was carried out for the TMS 9900 processor and its successor, the TMS 9995. Table 9.8.1 shows the results of the calculations from which it can be observed that:

i) TMS 9900 based designs can adequately control 1 and only 1 axis

ii) TMS 9995 based designs could control up to 5 axes in the most advantageous circumstances (Case 3) and, even in a more feasible situation, (Case 2) 3 axes could be servoed with over 20% spare processor capacity.
<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
<th>Case 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processor</strong></td>
<td>TMS 9900</td>
<td>TMS 9995</td>
<td>TMS 9995</td>
<td>TMS 9995</td>
</tr>
<tr>
<td><strong>Conditions</strong></td>
<td>No wait states</td>
<td>No wait states</td>
<td>No wait states</td>
<td>No wait states</td>
</tr>
<tr>
<td></td>
<td>Opcodes &amp; all oper-</td>
<td></td>
<td>Opcodes &amp; all oper-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ands off-chip</td>
<td></td>
<td>ands on-chip</td>
<td></td>
</tr>
<tr>
<td><strong>Total no. of clock</strong></td>
<td>3664</td>
<td>1558</td>
<td>1013</td>
<td>1283</td>
</tr>
<tr>
<td><strong>cycles</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total Time</strong></td>
<td>1221.33</td>
<td>519.33</td>
<td>337.67</td>
<td>427.67</td>
</tr>
<tr>
<td><strong>(microsecs)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>assuming 3 Mhz clock</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>For 2 msec sampling</strong></td>
<td>61.07</td>
<td>25.97</td>
<td>16.88</td>
<td>21.38</td>
</tr>
<tr>
<td><strong>rate, percentage</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>time occupied</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>For 2 msec sampling</strong></td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td><strong>rate, max no. of</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>axes controllable</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>by one processor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** In all calculations full account has been taken of addressing mode.

**TABLE 9.8.1 : ALGORITHM EXECUTION SPEED CALCULATIONS**
CHAPTER X

SERVO-VALVE INTERFACE DESIGN

10.1 INTRODUCTION

Two essential electronic hardware elements of the axis control function are interfaces to facilitate conversion and conditioning of the drive signal (to the solenoid operated pneumatic servo-valve) and the feedback signal (from the incremental encoder). The author was involved in significant work in designing, constructing, and testing these interface elements, primarily to facilitate research into pneumatic servo-control (Chapter IX), but also with a view to incorporation into future commercial products (Chapters XIV & XV). The design of the pneumatic servo-valve interface electronics demanded a novel approach as the valve was conceived and developed during the research and there were, at that time, no commercial alternatives.

This chapter and the next deal with the servo-valve and position feedback interfaces respectively.

10.2 THE CONTROL VALVE INTERFACE

The purpose of the control valve interface is to provide conversion and amplification of a multi-bit TTL signal into an equivalent analogue current, to drive the valve solenoid against the return spring and hence control the flow of air to the pneumatic actuator. This interface comprises two principle stages, digital to analogue conversion (DAC), and voltage to current conversion. The latter usually being referred to as the "servo-amplifier".

The general approach to the design of the complete servo-valve interface is shown in Figure 10.2.1. The inclusion of a dither input is discussed in Section 10.5.
10.3 DIGITAL TO ANALOGUE CONVERSION (DAC)

Figure 10.3.1 shows the DAC circuit used. In the design of this stage there were 3 primary requirements:
   i) Minimisation of cost
   ii) Good resolution to compensate for valves with small active ranges
   iii) 0-10v output range.

10.3.1 DAC Selection

A considerable range of integrated circuits (ICs) which perform digital-to-analogue conversion are commercially available and are manufactured with wide ranges of input (including 8, 10, 12, or 16 bit binary numbers) and output voltage capabilities.

Table 10.3.1 shows the relationship between DAC resolution and price. The required resolution was intuitively thought to lie somewhat above that offered by an 8-bit DAC. Whilst a 12 or 16 bit DAC would have been preferable from a performance point of view, the price of these devices and the possible cost implications of their
<table>
<thead>
<tr>
<th>Input</th>
<th>Resolution</th>
<th>Resolution (absolute)</th>
<th>Price (1985)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1 in 256</td>
<td>39.06 mV</td>
<td>£3-5</td>
</tr>
<tr>
<td>10</td>
<td>1 in 1024</td>
<td>9.76 mV</td>
<td>£7-12</td>
</tr>
<tr>
<td>12</td>
<td>1 in 4096</td>
<td>2.44 mV</td>
<td>£15-25</td>
</tr>
<tr>
<td>16</td>
<td>1 in 65536</td>
<td>0.15 mV</td>
<td>£25-50</td>
</tr>
</tbody>
</table>

Note: Resolution (absolute) figure assumes a DAC output range of 0-10v

TABLE 10.3.1: DAC INPUT CAPABILITY, RESOLUTION AND TYPICAL PRICE

extra I/O requirements necessitated a compromise. As the resolution of the 10-bit DAC was not expected to unduly degrade the servo-control performance this was selected.

10.3.2 Output Range
As the servo-valve is a non-centre null, single-solenoid, asymmetric valve there is considerable convenience, in terms of servo-amplifier design, in having a DAC stage with asymmetric (about 0v) output range

10.3.3 Pulse Width Modulation (PWM)
This approach to servo-valve interfacing has the characteristics of:

i) applying dither (Section 10.5) without additional hardware
ii) more easily compensating for shift of valve null
iii) heavily loading the processor or requiring special-purpose hardware.
This approach was not pursued as processor time was likely to be at a premium (Chapter IX, Section 9.8), the manpower and expertise to design PWM hardware was not readily available and also because additional hardware would add to the cost of any resulting products.
10.4 SERVO-AMPLIFIER DESIGN

This stage provides power amplification and essentially converts the DAC voltage output to a current capable of driving the servo-valve solenoid. Although designs were based on standard approaches, the requirements (detailed below) of this application, particularly the very restricted active range of the valve and its consequent sensitivity to even small drifts or instabilities, necessitated the development of circuits with a novel range of characteristics.

10.4.1 Requirements

To avoid possible duplication of effort it was required that the servo-amplifier be able to drive both the then available servo-valve (0 - 0.7 A active range) and any future designs with active ranges of up to 0-1A (Chapter IX, Section 9.3).

Due to the sensitivity of the servo-valve and effects of gravity on vertically mounted cylinders the exact null of the valve/cylinder pair, was and is, difficult to predict, and it was therefore necessary to provide a means of adjusting the null output condition of the servo-amplifier when in operation. To achieve this it was required that a suitably positioned potentiometer should be provided. The precise requirements in terms of relationships and sizes of the output current and zero or null adjust ranges are considered in Appendix 1.

Other obvious, but none-the-less essential requirements, were that the servo-amplifier should be low cost, linear, have adequate frequency response, and be unaffected by temperature or drift under the normal operating conditions of 0-70C.

10.4.2 Servo-amplifier Development

Several designs of servo-amplifier were constructed and tested during which the following problems were encountered:

i) drift (apparently mainly thermal in origin)

ii) insufficient output current range to drive a valve
with 0-1A active range.

iii) interaction of the zero adjust and output range adjust functions

v) instability in the zero adjust when using a potential divider circuit

vi) instability in the final drive stage.

10.4.3 Mark 3 Version Servo-amplifier

The final design (referred to as the Mark 3 Version), shown in Figure 10.4.1, resolved these problems and has the following features:

i) zener diode stabilised zero adjust

ii) d.c. decoupled dither input

iii) 30V load supply

iv) decoupled zero and range adjusts

v) high stability, low cost operational amplifiers

vi) double transistor "Darlington Pair" final drive stage with feedback stabilisation

vii) single load transistor with very good temperature coefficient over the required range.

It is this circuit which was used in the research into the pneumatic servo-control approach (Chapter IX) and on which the servo-amplifier circuits of the commercial distributed manipulator controllers are based (Chapters XIV and XV).

10.4.4 Resistor Value Selection

Appendix 1 considers in detail the values of the resistors R1, R2, Rz, and Rvar in the design of the Mark 3 Version servo-amplifier (Figure 10.4.1).

10.4.5 Mark 3 Version Testing

With the exception of the last 2 drift experiments, all tests were carried out using a 24.3 ohms (nominal), 48 watt resistive load. No dither was used.

Linearity (Figure 10.4.2) was found to be good for both settings of DAC input and output current range adjust. A maximum load current of over 1A was achieved.
Resistive load = 24.3 ohms
Room temperature
Load supply = 30V
Zero adjusted to give 0V at point A

For DAC input=10V and range adjusted to give 1A load current

Gain 1 = \(\frac{\Delta I_1}{\Delta V_1}\) = \(\frac{600}{3.05}\) = 196mA/V

Gain 2 = \(\frac{\Delta I_2}{\Delta V_2}\) = \(\frac{700}{7.05}\) = 99mA/V

\(\Delta V_1 = 3.05V\)
\(\Delta V_2 = 7.05V\)
\(\Delta I_1 = 600\) mA
\(\Delta I_2 = 700\) mA

For DAC input=5V and range adjusted to give 1A load current

OAC Incut = 10V
OAC Input voltage = 5V
DAC Input voltage (V)
The frequency response tests (Figure 10.4.3) were conducted using a sine wave signal generator with d.c offset level facility connected to the DAC input. A dual beam oscilloscope was used to monitor the input and output signal amplitudes. Tests were carried out with d.c. offset levels of 3v and 5v. In both cases the amplifier was found to have a constant gain up to 45 kHz and a bandwidth of over 44 kHz.

For the temperature test (Figure 10.4.4) both amplifier and resistive load were subjected to a rising ambient temperature by insertion into an oven. A change of only 1-2 mA in output current was observed up to 70°C but from 70°C to 90°C a small but distinct negative temperature characteristic was discernable. To ensure that no lags due to thermal conductivity times were present the temperature was maintained at over 89°C for 30 minutes. During this period no change in output current was observed.

Some low temperature testing was carried out but its significance must be doubted as when the amplifier was switched on initial warm up took place quickly, and it was difficult to be certain that the temperature values being recorded were correct. It was considered that to perform the test more effectively would require the amplifier being operated in a cooled environment, either a large freezer compartment or preferably a liquid nitrogen cooled thermos, and then allowed to reach steady temperature state before each reading was made. This procedure would have proved expensive in both equipment and time. Furthermore it was thought to be of limited use as the crude test conducted at least demonstrated that at temperatures below 0°C the load current was not dramatically different to those at normal ambients.

Three drift test were performed:

a) Resistive load, 900 mA current  (Figure 10.4.5)
b) Solenoid load, 500 mA current  (Figure 10.4.6)
c) Solenoid load, maximum current  (Figure 10.4.7)

In drift test (a) the amplifier was observed to be very
**Figure 10.43**

**Response Characteristic**

**MKA Servo-Amplifier: Frequency**

- Input dc offset = 3V
- Input amplified = 2V (pk-pk)
- Resisitive load = 24.3 ohms (nominal)

Bandwidth calculated from graph = 52.2 Hz

Bandwidth calculated from graph = 44.7 Hz
Notes:
1) Resistive load, nominal value 24.3 ohms
2) Temperature measured with Phillips PM2517X Multimeter and probe
3) System allowed to settle for 15 minutes before test
4) Time to ramp temperature 30°C - 90°C = 45 minutes

Set-up:
Zero adjusted to give 0V at Point A (Fig 10.4.1). Range adjusted to give 0.9A at Point B (Fig 10.4.1) with DAC input at 9.0V. DAC then adjusted to 5V.
FIGURE 10.4.5
MK3 SERVO-AMPLIFIER DRIFT TEST (WITH RESISTIVE LOAD) AND 900mA LOAD CURRENT

Notes:
1) Resistive load of 24.3 ohms (nominal)
2) Temperature varied between 22.8°C (t = 0) and 27.3°C (t = 350)
3) Test started at 9.40am

Set-up:
Zero adjusted to give 0v at Point A (Fig 10.4.1) with DAC input at 9.0v. Range adjusted to give 900mA (approx) at Point B (Fig 10.4.1). Set up time approximately 2 minutes.

Load current (mA)

Time (minutes)

-15, 0, +15v power supply interrupted momentarily

Remained at this level for readings at 165, 180, 313, and 350 mins
Notes: 1) Solenoid (Parmeko) load (24.3 ohms 11.4w)
2) Solenoid situated in normal sound proofed box
3) Temperature measured in sound proofed box with temperature probe and DVM
4) Normal air flow through valve (at 5 bar) permitted

Set-up: DAC adjusted to 5v. Range adjusted to give minimum current at Point B (Figure 10.4.1). Zero adjusted to give 500mA (approx) at Point B (Figure 10.4.1). Set up time approximately 2 minutes.
**Notes:**

Solenoid (Parmeko) load of 24.6 ohms, 11.4 watts
Solenoid situated in normal sound-proof housing and temperature inside the housing measured using a DVM and probe

**Set-up:**

Zero adjusted to give 0v at Point A (Fig 10.4.1)
DAC output adjusted to 10v at Point B (Fig 10.4.1) (Range adjust no effect)
Set-up performed overnight. Measurements from switch on at time t=0

**Graph:**

- **Load Current**
- **Temperature**

**Axes:**
- **Time (minutes)**: 0, 10, 20, 30, 40, 50, 60, 97, 126, 230, 260
- **Load Current (mA)**: 700, 750, 800, 850
- **Temperature (°C)**: 20, 30, 40, 50, 60, 70, 80
stable over a six hour period after a warm-up period of some 3-5 minutes during which a 4mA drift occurred. The reason for this initial drift is not clear but may be associated with the warm-up of the zener diodes or the resistive load. If the former then higher specification components could be employed without serious cost penalty.

Drift test (b) was performed with a solenoid controlled valve (24v, 1A, 11.4 watt) connected as load and taking an initial current of 500mA. The valve was located in its normal sound proof box with compressed air connected and flowing as in normal operation. Again the amplifier was observed to be very stable over a five and a half hour period, despite a considerable temperature rise (16C) in the sound-proofed box.

The final drift test was also performed with a solenoid load but with the saturated amplifier output current (>1A). Considerable drift occurred over a short period and the load current had fallen by 25% after 1 hour. A large temperature rise in the sound-proofed box had also taken place during the same period. However this was not considered to be the cause of the drift. A more likely explanation being a temperature induced rise in the resistance of the solenoid windings.

The Mark 3 version was considered to have more than adequate linearity, frequency response, drift and temperature characteristics for the control of the flow valve over the 0-70C temperature range, provided that the rate of execution of the control algorithm, and consequently the frequency with which the servo-amplifier is being driven, does not exceed around 30 Hz. The only problem, and that not necessarily insurmountable or particularly significant, being that of warm-up drift.

This version of the amplifier is appropriate to the control of valves with 0-1A active ranges except when operating at the high current limit of the active range where the increased solenoid resistance reduces the load current and the amplifier being saturated is unable to compensate. However this was considered unlikely to be a
major problem as, in normal operation, valves are not fully open/closed for more than a few seconds at most. Software protection would need to be incorporated into the control program to prevent this situation occurring for any length of time due to malfunction.

10.5 HYSTERESIS AND DITHER

The existence of significant hysteresis in the servo-valve/actuator has been previously mentioned (Chapter IX, Section 9.3). To reduce similar effects in hydraulic servo-systems an oscillating signal, usually referred to as "dither", is often superimposed upon the control signal.

Informal experiments at Loughborough University had shown qualitatively that dither produced similar benefits in this pneumatic-mechanical case, but unfortunately resources did not permit an investigation of the relationship between hysteresis and dither. However it was reasoned that if values of dither amplitude and frequency could be found which minimised the hysteresis in laboratory bench tests, then these same values would be equally appropriate to the same design of valve being used with any control algorithm. That is, the optimum values of dither amplitude and frequency are functions only of the pneumatic-mechanical system to which they are applied and not of the controlling algorithm, processor or servo-amplifier. With the object of establishing these optimum values a series of experiments was performed.

10.5.1 Experimental Results and Conclusions

A series of experiments was performed which examined the effect on hysteresis of varying dither amplitudes and frequencies (154). Figure 10.5.1 summarises the results with Figure 10.5.1(a) showing that the minimum hysteresis for a low dither amplitude is obtained at 20 Hz. However Figure 10.5.1(a) is considered only to provide an indication of the general shape of the curve and its minimum between 10Hz and 50Hz. Figure 10.5.1(b) shows that generally hysteresis decreases with increasing
FIGURE 10.5.1 SUMMARY OF DITHER/HYSTERESIS EXPERIMENTS: HYSTERESIS AS FUNCTION OF a) DITHER FREQUENCY AND b) DITHER AMPLITUDE

Note: All experiments (except where indicated) used sine wave inputs.
dither amplitude. Although there are comparatively few points on the graph it appears that this inverse relationship has no limits and that the higher bound on the value of the dither amplitude will be governed by other considerations such as noise level, and increased valve wear.

All experiments, with one exception, were carried out using sine waveforms. A square waveform of 20 Hz frequency and 15mA rms amplitude gave results very similar to those obtained for a comparable sinewave, indicating that the hysteresis level is not a function of waveform. However the benefit of employing the sine waveform is that if a frequency of 50Hz could be used without significantly reducing the effects of applying dither, it is attractive, in cost and simplicity terms, to employ a suitable mains derived dither source.

10.5.2 Summary of Dither/Hysteresis Aspects

The following conclusions have been drawn:

i) in terms of reducing hysteresis, dither has a beneficial effect

ii) dither has the disadvantages of increasing noise and valve wear (although neither of these effects has been quantified)

iii) dither frequencies in the range 10-50Hz are most beneficial

iv) increased dither amplitude reduces hysteresis. No upper bound to this trend was established, although in practice would be limited by excessive wear on the valve spool

However, several possibly important questions, remained to be answered:

i) are the curves equivalent to those in Figures 10.5.1 (a) and (b) the same for different servo-valves?

ii) are the indications that hysteresis is not a function of waveform correct?

iii) is it possible to formulate a quantitative theory of hysteresis that might yield information as to
whether similar or improved hysteresis dither characteristics exist for higher dither frequencies? Thus is it possible to trade off the beneficial effects of dither frequency against amplitude (and noise)?

As the resolution of these questions was beyond the means of the project it was decided that 50 Hz mains derived dither would be used as standard as this provided an inexpensive source of a suitable waveform at an appropriate frequency. The magnitude of the dither amplitude would be determined pragmatically in each case by acceptable noise levels.
11.1 INTRODUCTION
Chapter IX dealt with servo-control approaches and established the need to be able to obtain the position, velocity, and acceleration of a controlled axis. This chapter discusses the choice of the optical incremental encoder as the position measurement device, together with the design and implementation of suitable interfaces. Techniques for obtaining velocity and acceleration information are also considered.

11.2 SELECTION OF THE POSITION SENSOR
The wide range of available position transducers, including potentiometers, variable transformers, resolvers, and synchros, and optical encoders, has been widely reviewed (54, 155, 156). The following factors lead to the adoption of the rotary optical incremental encoder in this project:

i) ease of fitting to existing end-stop modules
   (Chapter XIV, Section 14.3)
ii) low cost
iii) ease of interfacing.

11.3 OPERATION OF OPTICAL INCREMENTAL ENCODERS
Figure 11.3.1(a) shows schematically the structure of the optical incremental encoder. An optical disc engraved with a number of lines (typically 250 to 2000) is mounted on a shaft which is in turn connected to the system whose position is to be measured, either directly or via gearing to rotary devices, alternatively using rack and pinion to linear movements.

The disc rotates between the two LED/photodiode pairs to produce, with the aid of suitable electronics (Figure 11.3.1(b)), two pulse streams. In the "forwards"
FIGURE 11.3.1  OPERATION OF OPTICAL INCREMENTAL ENCODERS
direction stream A lags stream B by 90 degrees (Figure 11.3.1(c)), in "reverse" stream A leads stream B (Figure 11.3.1(d)). If it is arranged that each rising edge of stream A is counted to give a position measure then the condition of stream B gives the direction of motion. For instance in Figure 11.3.1(c) and (d) the following holds:

- B high = "forwards"
- B low = "reverse"

Thus to be of use, the optical incremental encoder requires pulse counting and direction discrimination circuitry.

The fundamental problem of designing this sensor's interface was to select the optimal level of hardware/software trade-off with the objectives of minimal hardware complexity, and hence cost, without unacceptable degradation of software performance.

11.4 COUNTING IN SOFTWARE

Figure 11.4.1 shows a method of interfacing an incremental encoder to the TMS 9900 processor using the TMS 9901 Programmable Systems Interface IC (146). In this approach the CPU is interrupted on every rising edge only of every pulse from one encoder line. The service routine for this interrupt then inspects the state of the second encoder line to determine the direction of motion and either increments or decrements the stored position value.

The fundamental problems of this approach are those of processor load and response times. Consider an actuator moving with a velocity of 1 metre/sec whose position is monitored by an incremental encoder mounted so as to give a resolution of 0.025mm (approximately 1/1000 inches). If encoder pulses are counted by the TMS 9900 system suggested above at this speed and resolution, interrupts will be received by the processor at a rate of 40,000 per second (40 kHz), that is one every 25 microseconds. As the required interrupt service routine (Figure 11.4.1) occupies approximately 50 microseconds, some interrupts and consequently some pulses will be
FIGURE 11.4.1 SIMPLE INCREMENTAL ENCODER/MICROPROCESSOR INTERFACE AND ASSOCIATED SOFTWARE FLOW DIAGRAM
missed making the position value held by the microprocessor inaccurate. Thus the processor determines the best position resolution.

Even for resolutions of 0.1mm, 50% of the processor's time would be occupied in merely counting pulses. This activity would have to be at higher priority than all others, including the servo-control function which, taking up to 61% of the processor's time itself (Table 9.8.1), may be impaired.

The timing figures above apply to the TMS 9900 microprocessor (Chapter XIII, Section 13.5.1). For the somewhat faster TMS 9995 microprocessor (Chapter XIII, Section 13.5.3) the interrupt service routine execution time is approximately halved. Thus the use of this processor appears to largely resolve the problem. However, difficulties such as those described in Section 11.5.1 have not been considered and as larger interrupt service routines would be required for their solution, pulse counting in software was decided to be unwise unless the processor could be almost, if not entirely, devoted to the task.

Improved resolution was produced by inserting the circuit shown in Figure 11.4.2 into the one illustrated in Figure 11.4.1 as indicated. This enhancement provided a 2-fold increase in resolution by allowing the counter to register each change of state (rising and falling edges) of the encoder outputs compared to only the low to high transition. However, this improved resolution also doubled the incoming pulse rate, consequently doubling the processor load and aggravating the problem discussed in the previous paragraph, and was thus pursued no further.

11.5 COUNTING IN HARDWARE

The solution to the above problems is to perform pulse counting in hardware, thereby reducing the processor load. However this approach has the disadvantage of increasing hardware complexity and cost.
Encoder

Clear from microprocessor

Count pulse stream

Direction pulse stream

Position count before x2 resolution

Position count after x2 resolution

FIGURE 11.4.2 ENCODER x2 RESOLUTION ENHANCEMENT CIRCUIT
11.5.1 Preliminary Experiments

The first designs to perform incremental encoder counting in hardware were based on the circuits shown in Figure 11.5.1.

Whilst this counter circuit functioned correctly under test conditions, incorrect position values occurred when connected to operational modules. Due to the difficulty and expense of procuring appropriate equipment (most importantly a logic trace analyser) and the highly inconsistent nature of this fault, it was not possible for its cause to be positively identified. The ICs used in the design are individually capable of operating at in excess of 10MHz, this being a considerably greater frequency than the 40kHz produced by an encoder connected to a module moving at 1 metre/sec and resolving 0.025mm (Section 11.4), or indeed the maximum operating frequency of the encoder's optoelectronics.

However, it was suggested that vibration could have been the source of error. Consider the situation in which the module stops very close to the rising edge (C) of the count pulse stream shown in Figure 11.5.2(a). If vibration of any frequency (regular or otherwise) begins about this point, but does not exceed the limits A and E then the encoder counter will "see" the waveform pair shown in Figure 11.5.2(b), so that on each rising edge of the waveform the direction is perceived as being positive and thus the counter increments the although motion is actually vibratory. Vibrations of this nature could readily occur when the motion element strikes a mechanical stop.

If the vibrations were of larger amplitude so that the extremities A or E in Figure 11.5.2 were passed then correct counting would occur and so improving the resolution by mechanical gearing or the use of circuits such as that shown in Figure 11.4.2 (Section 11.4) at least reduces the chances of error arising. However, counting all 4 edges (rising and falling) of both pulse streams, using a scheme such as that shown in Figure 11.5.3, eliminates this problem.
Notes

1) All power rail layouts and decoupling devices MUST conform to low power Schmitt TTL noise requirements.
2) Capacitors C2 and C5 decouple the power rails close to the connectors.
3) Capacitors C3-C4 are to be placed between each pair of ICs as closely as possible.
4) All inputs held high or low should use active pull up or pull down.
5) C2 and C5 have value 1uF (Ferraz Xeramic).
6) C3-C4 have value 0.1uF (Ferraz Xeramic).

FIGURE 11.5.1 PRELIMINARY ENCODER COUNTER CIRCUIT
FIGURE 11.5.2 POSSIBLE SOURCE OF HARDWARE COUNTER ERROR
11.5.2 Secure Encoder Counter Circuits

As a result of these considerations the circuit shown in Figure 11.5.3 was designed. This circuit counts all 4 edges of both pulse streams as suggested in the previous section and thus is intended to eliminate erroneous count problems and achieve 4-times resolution enhancement by using a pulse direction enabling feature.

The operation of this complex circuit is outlined in Figure 11.5.4 and detailed in Appendix 2. The most important aspects of the circuit are:

i) The counting of erroneous pulse streams on a single line is prevented.

ii) Count is always 1 pulse in error in the reverse direction (Figure 11.5.5)

iii) Well-defined edge-delay-direction-delay-count sequence (Figure 11.5.5)

The well-defined operational sequence ((iii) above) and the delays contained therein, set the maximum operational frequency of the circuit at approximately 4.5Mhz, this being in excess of the maximum switching frequency of the encoders optoelectronics. It also ensures that the operation of the circuit does not rely upon the characteristics of the constituent ICs being well within their quoted ranges.

As the origin of the problems associated with the preliminary encoder counter circuits had not been definitely established, the success of these circuits were not guaranteed. However, hours of correct operation gave considerable confidence in the design.

11.5.3 Counter Circuit Initialisation and Load

The microprocessor signals necessary to initialise and zero the counter are given in Appendix 2. The counter circuit shown in Figure 11.5.3 may be loaded with any desired value by the injection, by the microprocessor, of a serial pulse stream into the counter whilst the encoder pulse stream itself is disabled and an appropriate direction signal set. Parallel load versions of the circuit were also designed and built.
FIGURE 11.5.3: SECURE ENCODER COUNTER CIRCUIT
Notes:

1. All power rail inputs and decoupling devices MUST conform to low power supply TIL noise requirements.
2. Capacitors C1, C2, and C3 decouple the power rails close to the connectors.
3. Capacitors C1-C11 are to be spaced between each pair of line blocks as possible.
4. All values must be high or low should use active pull up or pull down.
5. C1-C10 are to be spaced between each pair of line blocks as possible.
6. Capacitors C1-C11 are to be spaced between each pair of line blocks as possible.
7. C1-C11 are to be spaced between each pair of line blocks as possible.
8. C1-C11 are to be spaced between each pair of line blocks as possible.
9. C1-C11 are to be spaced between each pair of line blocks as possible.
10. C1-C11 are to be spaced between each pair of line blocks as possible.
11. C1-C11 are to be spaced between each pair of line blocks as possible.
12. C1-C11 are to be spaced between each pair of line blocks as possible.
FIGURE 11.5.3: SECURE ENCODER COUNTER CIRCUIT
Initialisation from processor

Wait for edge (rising or failing) from either encoder output

Expected edge (Figure 11.5.5)?

- Yes: Delay 100 nanoseconds
  - Latch appropriate direction (Figure 11.5.5) to counter and pulse enabling circuits
  - Delay 100 nanoseconds
  - Clock signal to counter and pulse enabling circuits
  - Increment / decrement counter and rotate pulse enabling circuits to next expected count / direction
  - Pulse pairs

- No (Ignore)

FIGURE 11.5.4 OUTLINE OF THE OPERATION OF THE ENCODER COUNTER CIRCUIT
FIGURE 11.5.5  PRINCIPLE OF OPERATION OF THE SECURE ENCODER COUNTER CIRCUIT
11.5.4 Other Secure Encoder Counter Circuits

Other designs of encoder counter have been suggested (157) but because of the unexplained nature of the problems experienced with preliminary designs (Section 11.5.1) and the confidence in the circuit of Figure 11.5.3 it was considered unnecessary to pursue these.

11.6 VELOCITY AND ACCELERATION MEASUREMENT

The servo-control algorithms, whose evolution has been described in Chapter IX, required the use of velocity and acceleration feedback and it is thus necessary to be able to measure these quantities. Two approaches are possible, direct measurement by appropriate transducers or derivation from other primary measurements.

11.6.1 Velocity and Acceleration Transducers

The ranges of devices available and their relative merits are well-documented (156). However they all possess the problem of increasing system complexity and cost. It was therefore considered that software derivation techniques would be more appropriate.

11.6.2 Position Differentiation

First order differencing is the most obvious method of deriving velocity from position. Using this technique position is sampled prior to every execution of the control algorithm and velocity calculated as the difference between this most recent position and its immediate predecessor. Thus

\[ V_i = \frac{(P_i - P_{i-1})}{t} - (1) \]

where

- \( V_i \) = Velocity at sample \( i \)
- \( P_i \) = position at sample \( i \)
- \( P_{i-1} \) = position at sample \( (i-1) \)

The value \( V \) represents the average velocity over the most recently completed sampling interval and thus the problem often associated with averaging, that the data is possibly highly inaccurate, is also associated with 1st order differencing (Figure 11.6.1). More frequent sampl-
ing eases this problem. The other major disadvantage of this approach is poor resolution at low velocities due to quantisation. Unfortunately this is worsened by increased sampling frequency.

Figure 11.6.2(a) shows the actual velocity of a linear motion element with incremental encoder position feedback resolving approximately 0.025mm as it approaches a stopping point. The results of measuring this velocity profile using the essentially digital incremental encoder and first order differencing is shown in Figure 11.6.2(b). Considerable quantisation has occurred resulting in the velocity value obtained being in error by as much as 1 pulse count, which represents a large percentage error at low velocities (e.g. 20% at 5 pulses/sampling interval).

Nth order differencing is defined (using the notation of Section 11.6.2) by:

\[ V_i = P_i - P_{(i-N)} \]

Figure 11.6.2(c) shows the results of 5th order differencing on the velocity profile of Figure 11.6.2(a). Quantisation has been much reduced but as 5th order differencing is in fact merely an averaging over a longer period the problems illustrated by Figure 11.6.1 are now more pronounced. These problems are also shown by Figure 11.6.2 where the velocity peak occurs later in (c) than in (a) or (b).

One other problem with Nth order differencing is that velocity values are larger, not necessarily a problem at low speed, but at higher velocities very large numbers may result. This may be rectified by scaling down, for instance dividing by 5 would for 5th order differencing reduce values to the same level as 1st order differencing. Figure 11.6.2(d) shows the outcome of this approach. Whilst the magnitude of the speed values has been reduced, the effects of quantisation reappear and thus resolution is poor. Also data is again late/inaccurate.

Other variations on these schemes have been suggested (158), but all suffer the problems of poor resolution.
<table>
<thead>
<tr>
<th>Time (secs)</th>
<th>0</th>
<th>0.01</th>
<th>0.02</th>
<th>0.03</th>
<th>0.04</th>
<th>0.05</th>
<th>0.06</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position (mm)</td>
<td>1.0</td>
<td>1.0</td>
<td>1.5</td>
<td>2.5</td>
<td>4.5</td>
<td>8.5</td>
<td>12.5</td>
</tr>
</tbody>
</table>

Actual velocity at time $t = 0.06$ secs

$$v = \frac{8}{0.02} = 400 \text{ mm/sec}$$

Average velocity measured over period 0 - 0.6 secs

$$\frac{12.5}{0.06} = 208 \text{ mm/sec}$$

Average velocity measured over period 0.03 - 0.6 secs (effectively increased sampling frequency)

$$\frac{10}{0.03} = 333 \text{ mm/sec}$$

**FIGURE 11.6.1** AVERAGING PROBLEMS IN VELOCITY DERIVATION USING POSITION DIFFERENCING
FIGURE 11.6.2  VELOCITY MEASUREMENT BY VARIOUS POSITION DIFFERENCING TECHNIQUES

Notes:
1) 10 counts/sampling interval ≈ 30 mm/sec
2) 1 sampling interval ≈ 200 milliseconds
due to quantisation effects at low velocities together with late and/or inaccurate data due to averaging.

All of the preceding discussion has concerned velocity derivation. Exactly similar approaches can be used for the acceleration feedback with very similar or more pronounced consequences.

A more detailed consideration of these problems necessitates the use of discrete time control (159) which is beyond the scope of this work together with an understanding of how the problems are likely to effect the performance of the servo control algorithm. In the first instance the largely pragmatic consideration of simplicity suggested the use of 1st order differencing and as results of the required level were obtained, more complex approaches were not investigated.

11.7 SINGLE CHIP INCREMENTAL ENCODER COUNTERS

During the course of the project an IC (74LS2000) performing all the functions of the secure encoder counter circuit was introduced by Texas Instruments (160). Although this device was not used during the development phase of this project, the potential to reduce hardware cost and complexity of any commercial implementations was made available (Chapter XIV, Section 15.6.1)

The 74LS2000 has several features and can be used in modes whereby the width or frequency of pulses from the encoder can be measured. The width measurement facility readily yields velocity measurements and so in principle the 74LS2000 could be used as a velocity transducer interface. Unfortunately it cannot be used in pulse counting and width measurement modes simultaneously and therefore two would be needed for position and velocity feedback. Although the expense of an extra velocity transducer would be alleviated, direct velocity feedback would still incur increased hardware complexity and cost. For this reason and because adequate performance had been achieved using 1st order differencing, the use of the 74LS2000 IC in velocity measurement was not pursued.
12.1 INTRODUCTION

Chapter V proposed a distributed hierarchical multi-processor architectural framework for the control systems of distributed manipulators. In this scheme servo-control of at most a few axes was performed by a small computer (SCC/SAC) and the activities of these SCCs/SACs were organised by a supervisory computer. Chapter VIII proposed the use of Programmable Logic Controllers (PLCs) and Industrial Computers as machines which could perform this supervisory function. Thus it is necessary that the servo control computer (SCC/SAC) should possess communication systems capable of interfacing to both of these types of supervisor. This chapter considers the requirements and design of this communications system.

12.2 REQUIREMENTS

Chapter VIII described some of the commands and their associated data that might be required to be transmitted between supervisor and SCC/SAC and vice-versa to achieve point-to-point motion control of pneumatic, servo-controlled axes. Table 12.2.1 gives a comprehensive list, from which it is clear that the total message length (command plus data) is variable.

Table 12.2.1 also shows that only 24 commands were required and that the allocation of 1 byte to encode the commands would be sufficient and allow for expansion of up to a total of 128 such commands. Thus (from Table 12.2.1) the maximum message length (command plus data) is 3 bytes, assuming a 16-bit position word (Chapter XI).

If the controlled pneumo-mechanical system is assumed to have a natural frequency which will not exceed 20Hz, then in achieving point-to-point motion control,
### Supervisor to SCC/SAC commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Data</th>
<th>SCC/SAC to Supervisor replies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialise</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load set point</td>
<td>set point (2)</td>
<td></td>
</tr>
<tr>
<td>Go to set point</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request status</td>
<td></td>
<td>status (2)</td>
</tr>
<tr>
<td>Request position</td>
<td></td>
<td>position (2)</td>
</tr>
<tr>
<td>ROM to RAM download</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Axis to home pos'n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initialise counter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zero counter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load counter</td>
<td>c'ter value (2)</td>
<td></td>
</tr>
<tr>
<td>Null valve</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brake on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brake off</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P'meter change no.</td>
<td>P'meter no. (2)</td>
<td></td>
</tr>
<tr>
<td>New parameter value</td>
<td>P'meter Val (2)</td>
<td></td>
</tr>
<tr>
<td>Activate teach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deactivate teach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teach fast mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teach slow mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teach move forward</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teach move reverse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dither on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dither off</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emergency stop</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: 1) Figures in parentheses in the second and third columns give data length in bytes.
2) SCC/SAC instigation of communication (ie SCC/SAC generated commands are not required).

**TABLE 12.2.1 : LIST OF COMMANDS AND ASSOCIATED DATA**

The communications system is required to transmit/receive a message and any associated reply (comprising a maximum of 5 bytes) together with any protocol overhead in less than 25 milliseconds, assuming sampling rate must be at least twice the natural frequency to avoid data loss.

So that the communications system will be easy for the programmer of the Supervisor to use both the protocol and data format must not be overly complex.

Good data integrity is also required although this is not usually easily achieved without considerable protocol overhead. This suggests two levels of system, one with a simple protocol and minimal integrity checking.
and a second with a more complex protocol and greater integrity.

Furthermore as identified in Section 5.2.1 distributed manipulators are required to be low cost thus the communications system must not introduce significant cost penalty or be difficult to program or implement in terms of electronic hardware, software, or testing.

The requirements of the communications system may therefore be summarised as:

i) bi-directional transfers of maximum length 5 bytes
ii) maximum time of 25 milliseconds for transfer of each 5 byte message/reply and associated overhead
iii) easy for the Supervisory computer programmer to use
iv) opportunity to utilise differing levels of integrity checking
v) usable by PCs and Industrial Computers (Section 12.1)
vi) minimum cost.

12.3 THE ISO/OSI 7 LAYER MODEL

The internationally recognised ISO/OSI reference model uses the technique of layering to divide the interconnection problem into manageable subproblems. The layers form a hierarchy with each layer defining a subset of functions necessary for open systems interconnection (OSI). Seven layers were considered to be sufficient to divide the problem into structurally similar but logically distinct functions. Figure 12.3.1 illustrates this layered structure. Various standards are available which meet the requirements of each layer and in the construction of a communications system designers may select one standard for each layer.

For less sophisticated systems one or more of the upper layers (3 and above) are inappropriate or application dependent. This is true of the tightly coupled system under discussion, so that layers 1 and 2 are of primary interest. In this case it is appropriate to
User Programs

<table>
<thead>
<tr>
<th>Layer 7</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 6</td>
<td>Presentation</td>
</tr>
<tr>
<td>Layer 5</td>
<td>Session</td>
</tr>
<tr>
<td>Layer 4</td>
<td>Transport</td>
</tr>
<tr>
<td>Layer 3</td>
<td>Network</td>
</tr>
<tr>
<td>Layer 2</td>
<td>Data link</td>
</tr>
<tr>
<td>Layer 1</td>
<td>Physical</td>
</tr>
</tbody>
</table>

**FIGURE 12.3.1** ISO/OSI SEVEN LAYER REFERENCE MODEL

'embody any necessary functionality of the remaining layers within the user program (Figure 12.3.1), which, for this application, is the software considered in Chapters XIV and XV.

Thus a simpler approach is well-suited to this distributed manipulator system in which only point-to-point servo-control is required but greater functionality may be needed where manipulators must be integrated within the wider context of CIM.

The following two sections discuss layers 1 and 2, the Physical link and Data link layers respectively.
12.4 PHYSICAL LAYER

Brief reference was made in Chapter V, Section 5.4.4 to the advantages of digital communications where fast, accurate data transfer was required and this discussion will not therefore encompass analogue links.

12.4.1 Serial and Parallel Communication

Digital communications systems can be classified as serial or parallel (Figure 12.4.1 a) and b)). Fundamentally parallel links are the faster, but involve more hardware costs in terms of line drivers, receivers and cabling. As full bi-directional serial links can be implemented using only 3 wires, this approach is attractive in terms of cost minimisation.

12.4.2 Serial Communications Standards

Many digital devices communicate using serial links and many standards are now in existence. These standards often specify the number of wires involved, the nature of the signals carried over each wire, and the electrical characteristics of the cable which in turn usually define the maximum length of the link and the number of devices which may communicate over it. Furthermore, integrated circuits (ICs) are frequently available which are designed to provide a direct microprocessor interface to one of the standard serial interfaces so that the processor merely has to load information to be transmitted into the IC or unload data received using simple I/O instructions. All aspects of the communication are then organised by these ICs, including data rate determination, addition of parity and start/stop bits, and voltage levels (161). For this project, from the point of view of ease of implementation, the use of readily available ICs would be beneficial and, if the range of devices capable of acting as supervisory controllers were to be as large as possible, the use of a very common serial communications standard was essential.
a) Transmission of hexadecimal 9CA7 using 1 serial link and four time intervals

Time Interval for the transmission of one byte

b) Transmission of hexadecimal 9CA7 using 4 parallel links and one time interval

FIGURE 12.4.1 PARALLEL AND SERIAL COMMUNICATIONS
12.4.3 RS-232-C

The most frequently encountered of these standards is RS-232-C and this is therefore worthy of closer examination. As fully specified (162), RS-232-C requires a 25-way link, but is usually implemented as a subset and can readily be used with just three wires, namely transmit, receive, and signal ground. Furthermore RS-232-C defines the electrical characteristics of and voltage levels applied to each wire including the logic levels of +/-12volts. The nature of the electrical characteristics specified mean that strictly, each RS-232-C transmitter is capable of driving only one receiver and that over a maximum distance of at most a few metres. Also the system has little intrinsic immunity to electrical noise.

12.4.4 RS-422-A

In some applications the supervisory controller may be required to communicate with several SCCs/SACs. If each SCC/SAC requires a dedicated serial port at the supervisor (Figure 12.4.2a), this clearly increases the hardware complexity and probably the cost of the supervisor. A preferable approach is to use just one serial port at the supervisor and have all SACs connected to that port. This approach is referred to "multi-drop" (Figure 12.4.2b). The most commonly used serial standard with this multi-drop capability is RS-422-A (163) which also possesses the capability of driving over hundreds of metres and has transmitter/receiver ICs readily available. Furthermore RS-422-A is a balanced system, that is each signal has associated with it a complete circuit consisting of two wires, and consequently has greater intrinsic electrical noise immunity. This approach then is applicable in the potentially electrically noisy shopfloor environment and assists in achieving cost minimisation.

12.5 DATA LINK LAYER

This layer has two distinct aspects, media access control and link layer control, which are discussed in
a) Using RS-232-C

b) RS-422-A

FIGURE 12.4.2 COMMUNICATIONS WITH MULTIPLE SCCs/SACs
Sections 12.5.1 and 12.5.2 respectively.

12.5.1 Media Access Control (MAC)

Many communications systems have no master devices to organise the use of the communications medium, so that any device attached to the system has to establish that the communications medium is not in use and only then is it permitted to send its message. If the medium is active then the device must repeatedly examine the system until it is found to be free. For lightly loaded systems of this type the approach has the benefits of being simple and high speed. As traffic increases, the likelihood of the medium being active (collision) increases and more time is wasted in resolving the contentions. In many situations these time delays, which might amount to a few tenths of seconds at most, and the fact that they are probabilistic (that is of unpredictable magnitude) are not important. A particular implementation of this media access control method is commonly referred to as "Carrier Sense Multiple Access/Collision Detect" (CSMA/CD) and an example of such a system is the Ethernet local area network (164). Ethernet is frequently used in office situations where, although communications speed is important, the exact length of a delay in accessing the transmissions medium is not.

However, in the real-time conditions that exist in many manufacturing environments, where much can occur in 0.1 seconds it is important to have a deterministic communications system in which, even if the communication is not particularly fast, it has at least a calculable slowness.

It is for precisely this reason that the original specifiers of the Manufacturing Applications Protocol (MAP) (165), General Motors (GM) selected token passing media access control, which in the case of commercial broadband implementations can lead to bus access times of in excess of one second (166), but is at least always predictable.

Token passing MAC, as used with MAP Version 2.2,
achieves predictability by transferring data in an allocated slot time. That is each device connected to the system is allocated times during which it may access the medium and thus contention is avoided. The techniques required to achieve this synchronisation can involve significant complexity in the hardware and software of the communicating devices. As simplicity and predictable response times were requirements of the distributed manipulator communications system (Section 12.2), neither token passing nor CSMA/CD media access control was considered ideal.

An alternative method of achieving predictable access times is to assign one communicating device as "Master" and the remainder as "Slaves". In a relationship of this type slaves only send data when requested to do so by the master and so no synchronisation protocol or any associated cabling is required. This simple, low cost approach is well suited to communications between Supervisors and SCCs/SACs in distributed manipulator systems with the supervisor assigned as "master" and the SCCs/SACs as "slaves". Here a natural master/slave relationship exists in which the supervisor has a monitoring/co-ordinating role.

The major disadvantage of a master/slave approach is that slaves cannot instigate communications. In normal operation this facility is not required, SCC/SACs being merely subservient devices performing the actions requested of them by the supervisor. Only when a SCC/SAC detects a serious fault and wishes to shut down its axis does a SCC/SAC have a requirement to originate communications. To alleviate this situation the programmer of the supervisory system must be sufficiently disciplined to check the status of each SCC/SAC at regular intervals.

12.5.2 Link Layer Control

As indicated in Section 12.2 data transfers of different lengths are required. Two approaches are commonly used, fixed and flexible formatting.

In a flexible formatting system both sender and
receiver are aware of the length of each command transfer and are therefore able, at send/receive time, to adjust the lengths of their data transmission/reception. This has the benefit of restricting to the minimum the amount of data flowing and hence maximising the data rate.

In fixed formatting the number of bytes required by the longest command is always transferred, those not used (by shorter commands) being filled with dummy data. Clearly this impairs the data rate but does have the advantage of very simple communications software in both sender and receiver.

In a multi-drop configuration (Section 12.4.4), in order that only the desired SCC/SAC responds to a supervisor command, each SCC/SAC must be associated with an address, most conveniently by the setting of switches in the SCC/SAC. If each instruction is then prefixed with an address, only the required axis controller will respond. Thus the command format discussed above now requires an extra byte in which will be placed the address of the SCC/SAC for whom the message is intended. Accordingly a maximum of 128 SCCs/SACs axes can be organised and monitored by a single supervisor, this number not being considered to be a limit to industrial application. A supervisor-SCC/SAC fixed message length (address plus command plus data) of 4 bytes was therefore applicable (Figure 12.5.1).

When a supervisory command is a request to a SCC/SAC to transmit data, the origin of the information received by the supervisor will be known. As it is clearly not sensible for the supervisor to ask several SCCs/SACs to send simultaneously, contention will not arise. Thus no reply address overhead is incurred and a SCC/SAC-supervisor fixed length reply of 2 bytes is suggested.

As software simplicity, especially for the supervisory computer programmer, is a requirement (Section 12.2), fixed format link layer control was considered the optimum choice with message formatting as shown in Figure 12.5.1.
12.5.3 Error Detection and Recovery

This is a complex area and an in-depth coverage is beyond the scope of this thesis. In sophisticated communications systems each layer includes a measure of error detection and recovery. At its simplest, parity checking provides a low level of error detection, although it is trivial to envisage error situations which would not be detected by parity (for instance any even number of bits in a single byte being reversed).

Other methods of increasing complexity (167) are capable of detecting increasing percentages of errors. Perhaps the most sophisticated of these is CRC (Cyclic Redundancy Check) which is capable of trapping almost 100% of errors. The disadvantages of these methods lie in the time overhead involved and their complexity to applications programmers.
12.6 COMMUNICATION SYSTEM SELECTION

Having stated the requirements of the distributed manipulator communications system (Section 12.2) and reviewed the standards and approaches available in relation to the ISO/OSI reference model (Sections 12.4 and 12.5) it is now possible to specify the communication system.

Section 12.4.1 and 12.4.4 respectively proposed serial communications using the RS-422-A standard. Section 12.5.1 suggested master/slave media access control (MAC) should be adopted with a fixed format approach being used for link layer control (Section 12.5.2).

12.6.1 RS-422-A Hardware

In a system containing a supervisor (master) communicating with several SCCs/SACs (slaves) using multi-drop RS-422-A links as shown in Figure 12.6.1, each RS-422-A transceiver may be enabled/disabled. This is vital for multi-drop systems as, if SCC1/SAC1 sends whilst the transmitter of SCC2/SAC2 is enabled, the SCC1/SAC1 transmitter will be driving into SCC2/SAC2's transmitter. If there is no risk of such contention (where there is only 1 SCC/SAC) then the SCC/SAC's RS-422-A transceiver may be permanently enabled.

12.6.2 Combined RS-422-A/RS-232-C Systems

Neither the RS-232-C nor the RS-422-A definitions (162,163) specify the nature of the communicated data nor the media access control method. Thus for systems in which the multi-drop high noise immunity features of RS-422-A are not required, RS-232-C could be utilised. Such an RS-232-C interface would be of use for testing and in applications areas requiring a single SCC/SAC.

In a system where transmit and receive are the only circuits used the hardware shown in Figure 12.6.2 may be employed. Here a standard UART (universal asynchronous receiver/transmitter) interfaces to both RS-232-C and RS-422-A line drivers and receivers. Therefore the SCC/SAC software need not be aware of which standard is in use
Notes:
- D = Driver
- R = Receiver
- DE = Drive enable (active high)
- RE = Receive enable (active low)
- RTS = Request to send (active high)

FIGURE 12.6.1  RS-422-A MULTI-DROP COMMUNICATIONS USING SN75176 DIFFERENTIAL TRANCEIVERS
FIGURE 126.2

SCC / SAC CPU
(eg TMS 9995)

RS-232-C
and RS-422-A

Notes:

IC1  is SN75188(1/4)
IC2  is SN75189(1/4)
IC3 and IC4 are SN75178
IC5  is 74LS14(1/6)

D = Driver
R = Receiver
DE = Drive enable
(Active high)
RE = Receive enable
(Active low)
RTS = Request to send (Active
low)

Xout = UART transmit
Rin = UART receive

GND

IC4

IC5

RS-422-A

D
DE

RS-232-C

IC2

Tx
Rx

IC1

IC3

Tx-
Tx+

Rx+
Rx-

IC4

UART
(eg TMS 9902)

Xout
Rin

RTC

GND

1C5

RS-232-C

Rx

Driver

Receiver

Drive enable

Receive enable

Request to send

Transmit

Receive

Enable
for inter-processor communications, and indeed only simple jumpering is required for selection.

This approach offers flexibility and choice to the user and incurs only a small cost penalty as both RS-232-C and RS-422-A line driver/receiver ICs are inexpensive and physically small.

12.6.3 Data Rates
Section 12.2 identified the need for data rates such that a complete message/reply sequence could be transmitted in at most 25 milliseconds. Section 12.5.2 suggested fixed format messages of 4 bytes and replies of 2 bytes, a total of 6 bytes. Serial RS-232-C/RS-422-A hardware operating at 9600 bits/sec with 1 start bit, 1 stop bit, and 2 parity bits will achieve this in 6.88 milliseconds, well within the required period, even when allowance is made for SCC/SAC response time.

12.6.4 Summary
Having considered the requirements of the distributed manipulator communications system to achieve point-to-point positioning control, the necessary features of the system can be summarised as:

1) serial physical layer operating at 9600 baud and using the RS-422-A standard for good noise immunity and/or multiple SCC/SAC applications, and the RS-232-C standard for testing or simple single SCC/SAC applications

ii) requiring only simple jumpering for the selection of RS-232-C or RS-422-A communications

iii) media access control (MAC) using simple master(supervisor)/slave(SCC/SAC) relationships

vi) link layer control using fixed formatting including 1 byte addressing.
CHAPTER XIII

MICROPROCESSOR SELECTION

13.1 INTRODUCTION

Chapter IX discussed the essential research work performed to achieve adequate servo-control of the pneumatic actuator. The use of the Texas Instruments 9900 microprocessor for this work was also described. This chapter discusses the selection of the TMS 9995 microprocessor for the prototype and production versions of the servo-control computer (SCC).

13.2 THE DISADVANTAGES OF THE TMS 9900 MICROPROCESSOR

The TMS 9900, although still technically a suitable device for the servo-control function, has several disadvantages as a microprocessor on which to base a low cost production printed circuit board (PCB). Having a 64 pin package, it occupies a considerable board area, and is, in semi-conductor terms, a processor design of some age. The architecture, although arguably better suited to real-time control, is very different to that of the more recently designed and now more common Intel 8086 and Motorola 68000 processors. More importantly, the manufacturer's development systems and software support was, in 1983, deteriorating. This situation has since become more coherent. Not unnaturally concerned for the longevity of their proposed product, Martonair wished to examine the possibilities and penalties of the use of other processors. To this end a Department of Trade and Industry sponsored "Microprocessor Applications Study" (MAPCON) was commissioned with which the author was intimately involved, being concerned with information supply in general and the specification of hardware and software requirements in particular.
13.3 THE MAPCON STUDY

The Cranfield Product Engineering Centre (CPEC) undertook this activity. CPEC is a commercial offshoot of Cranfield Institute of Technology specialising in the design and construction of microprocessor-based products, with, at the time in question, experience mainly in the use of the Zilog Z80 8-bit microprocessor.

13.3.1 Approach

The terms of reference given to the consultants was to study "The application of microprocessors to the commercial exploitation" of the pneumatically actuated programmable positioners which would be utilised within the distributed manipulator system whose design has been considered in this thesis. The consultant's report (168) is now examined in some detail.

The specifications of the proposed product were considered in two parts. Firstly, a facility was needed for up to three axes of point-to-point servo-controlled motion. Secondly, additional functions were required to make this servo-positioning capability available to the user. For instance some means of loading positions would be needed, as would the ability to interface the controller to other equipment. These functions were collectively called the "Operator Interface" and this term will be used henceforth in this chapter.

In the first instance five different approaches were considered:

i) Single TMS 9900 microprocessor performing both servo-control and operator interface functions
ii) TMS 9900 based servo-control with Z80 performing the operator interface function
iii) Two separate Z80 microprocessors, one performing servo-control and the other the operator interface function
iv) A single Z80 performing both functions
v) Z80 based operator interface with Intel 8751 single chip microcomputers dedicated to the servo-control of each axis.
Only the second, third and fourth options were considered further. The first was rejected as "neither Martonair nor CPEC have any experience of the 9900 and purchasing the development tools would be expensive" and the last as being "not favoured by any of the Cranfield departments who design and use servo-systems, as the inter-device communication requirements add(ed) unnecessarily to the complexity of the system."

The costs of the three remaining options were considered by examining the construction of, from the component level:

i) Z80 based operator interface unit

ii) Z80 based combined operator interface and servo unit

and from the board level,

iii) Z80 based operator interface

iv) Z80 based combined operator interface and servo unit

v) Z80 servo unit

vi) TMS 9900 based servo unit.

and then adding the appropriate partial costs to arrive at a total cost for each option.

13.3.2 Conclusions

Table 13.3.1 gives the partial costs arrived at by the MAPCON study. These costs are then used to arrive at the total product costs shown in Table 13.3.2. From these figures it is clear that even the option with the lowest product cost, E, was above the original target cost of £1250. Option B offered the best compromise of product and development costs.

Despite recognising earlier in the study that "Re-designing the servo would not take as long again as experience has been built up, but it would be a major task due to the constraints of real time software performance" it was concluded that option E was to be preferred as the convenience for Martonair in only working with one supplier would probably more than balance the higher
development costs.

Although it does not appear to have been taken into account in the conclusions, another of the study's earlier passages is noteworthy:

"In deciding which microprocessor to use, and in what form, the past experience of LUT and Martonair is very important. Even if a microprocessor not used before were technically the best, the cost and effort needed to develop and support it may out-weigh any disadvantage in compromising on performance and using current microprocessor technology"

<table>
<thead>
<tr>
<th>Component</th>
<th>Assembly &amp; Test</th>
<th>Development Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>From component level-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i) Z80 based operator i/f unit</td>
<td>£670</td>
<td>£250</td>
</tr>
<tr>
<td>ii) Z80 based combined operator i/f and servo unit</td>
<td>£1150</td>
<td>£300</td>
</tr>
<tr>
<td>From board level-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iii) Z80 based operator i/f unit</td>
<td>£1800</td>
<td>£300</td>
</tr>
<tr>
<td>iv) Z80 based combined operator i/f and servo unit</td>
<td>£2650</td>
<td>£600</td>
</tr>
<tr>
<td>v) Z80 servo unit</td>
<td>£1800</td>
<td>£300</td>
</tr>
<tr>
<td>vi) TMS 9900 servo unit</td>
<td>£1200</td>
<td>£250</td>
</tr>
</tbody>
</table>

The bracketed figure in the "Development Costs" column gives the elapsed time in weeks

Note 1: Assuming servo-algorithm defined by Martonair

TABLE 13.3.1: MAPCON PARTIAL COSTINGS (1983 prices)
<table>
<thead>
<tr>
<th>Option</th>
<th>Total Product Cost (Cmpnts, Assm &amp; Test)</th>
<th>Development Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>£3550</td>
<td>£54k+ (Note 1)</td>
</tr>
<tr>
<td>B</td>
<td>£2370</td>
<td>£60k+ (Note 1)</td>
</tr>
<tr>
<td>C</td>
<td>£4200</td>
<td>£74k (Note 2)</td>
</tr>
<tr>
<td>D</td>
<td>£3020</td>
<td>£80k (Note 2)</td>
</tr>
<tr>
<td>E</td>
<td>£1450</td>
<td>£75k</td>
</tr>
<tr>
<td>F</td>
<td>£3250</td>
<td>£80k</td>
</tr>
</tbody>
</table>

Note 1: Cost of developing TMS 9900 based servo not included
Note 2: Assuming servo algorithm defined by Martonair

**TABLE 13.3.2: MAPCON PRODUCT AND DEVELOPMENT COSTINGS**

(1983 prices)
13.3.3 Criticisms

Unfortunately the study only attempted to cost Z80 based designs. This device, having an 8-bit architecture was, and is, less suitable for performing the real-time 16-bit arithmetic required by the servo-control function than an equivalent 16-bit processor. This omission must be considered most regrettable as it prevented accurate cost comparisons being made with routes based on any of the common 16-bit Intel or Motorola microprocessors, or more importantly with products based on the TMS 9900 microprocessor family. Why these options were not considered is not clear, perhaps lack of time or lack of experience with these 16-bit devices are reasons. If so, this would clearly call into question the basis of the MAPCON scheme, as reports with these restrictions are very close to being normal commercial suppliers quotations, not as supposedly intended, studies into the best way of implementing a potential microprocessor product.

13.3.4 The Impact of the MAPCON Study

At the time of the study Martonair were a company with relatively little experience in the design and construction of microprocessor-based systems and in this context some valuable indications were provided for Martonair by the MAPCON study.

The development and product costs involved in designs based on 8-bit microprocessors were now clear. Although no quantitative evidence had been offered by the report of the benefits of remaining with a TMS 9900 family based design, as this option had been discarded by the study at an early stage (Section 13.3.1), some of the cost implications of rewriting the assembler derived servo-control software had been recognised (Section 13.3.2). Thus by inference the problems of working with other 16-bit microprocessors could be understood and some basis gained for balancing these problems against the disadvantages of the TMS 9900 (Section 13.2).
13.4 SELECTION OF A TMS 9900 FAMILY MICROPROCESSOR

As a result of the MAPCON study it was concluded that development based on 8-bit microprocessors was inappropriate as it provided a product of too greater price and incurred development costs above the level which were considered to be commensurate with the anticipated return. This left a choice between the TMS 9900 family and other 16-bit microprocessors, with the following major points to be considered:

i) Disadvantages of the TMS 9900 processor
   (Section 13.2), mainly support, age, compatibility, and pin count.

ii) High development costs involved in the transport­ation of already available software and hardware designs to other microprocessors.

iii) Loss of LUT expertise and support.

It was decided that points ii) and iii) out-weighed i). Particularly relevant was the consideration that, if a design were based on processors other than the TMS 9900 family, LUT would be less able to offer relatively impartial advice. To a company, such as Martonair, venturing into the field of microprocessor based products for the first time, this availability of a sympathetic (to their cause) party able to act as design authority and provide informed comment on any contracted work was most useful.

The following section considers the choice of the TMS 9995 microprocessor. Appendix 3 gives more details of the operation and design of the TMS 9900 family of microprocessors and support devices relevant to this work. Complete coverage is given in the TMS 9900 Family Data Book (146).

13.5 SELECTION OF THE TMS 9995 MICROPROCESSOR

The Texas Instruments TMS 9900 family were introduced in the late 1970s and at that time were the first commonly available 16-bit devices. The family has four variants, the TMS 9900 (Section 13.5.1), the TMS 9980A/9981 (Section 13.5.2), and the TMS 9995 (Section
13.5.3). Their principle features are shown and contrasted by Table 13.5.1. All four processors are fabricated in NMOS silicon, have a memory-to-memory architecture (Appendix 3, Section A3.2) and are assembler code upwards compatible. Support devices are available (Table 13.5.2), including the TMS 9902 asynchronous serial communications interface (Appendix 3, Section A3.5).

13.5.1 TMS 9900 (Appendix A3.3)
This is the original member of the family and, although proven technically capable of the required functions (Chapter IX), suffered two major disadvantages for a compact, low cost application:
  i) High pin count and thus physically large package
  ii) 3 power supplies required.

13.5.2 TMS 9980A/9981
These processors, although of lower pin count than the TMS 9900, were not suitable as they
  i) required multiple power supplies and
  ii) possessed a possibly restrictive 16k byte memory addressing capability

13.5.3 TMS 9995 (Appendix 3, Section A3.4)
This processor is very similar to the TMS 9900 but being of a later design utilises the improvements in silicon fabrication that have been made to incorporate on-board a clock generator, 256 bytes of RAM (Section A3.4.1), and a timer/decrementer (Section A3.4.2). Also only one power supply (5v) is needed and instruction pre-fetch enables greater operating speed. The TMS 9995 is upwards code compatible with the TMS 9900 and has two extra instructions, signed multiply (MPYS) and signed divide (DIVS).

Several features make the TMS 9995 attractive for this application:
  i) fast processor, with good interrupt response time
  and the potential for the servo-control of up to 3 axes (Chapter IX, Section 9.8)
ii) reasonably low pin-count reducing hardware cost

iii) good on-chip facilities which would also lessen

hardware cost and could increase execution speed.

As the extra facilities of the TMS 9900, such as the
ability to handle 16 interrupts, were not required, the
TMS 9995 was selected and in fact is ideally matched to
the application as is evident from the similarity
(excepting in I/O and ROM features) of the Intel 8096
(70) which was designed originally for automotive applica-
tions whose requirements were very much akin to those of
this project.

The use of a Texas Instruments TMS 7000 series 8-bit
microcomputer (146) was also considered but not pursued
as the TMS 7000 was:

i) not available in EPROM versions (until 1986)

ii) not TMS 9900 code compatible.
<table>
<thead>
<tr>
<th>TMS Family No.</th>
<th>9900</th>
<th>9980A</th>
<th>9981</th>
<th>9995</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Word Length</td>
<td>16-bit</td>
<td>16-bit</td>
<td>16-bit</td>
<td>16-bit</td>
</tr>
<tr>
<td>Memory Address Space</td>
<td>64k bytes</td>
<td>16k bytes</td>
<td>16k bytes</td>
<td>64k bytes</td>
</tr>
<tr>
<td>Speed</td>
<td>3Mhz</td>
<td>3Mhz</td>
<td>3Mhz</td>
<td>3Mhz</td>
</tr>
<tr>
<td>I/O bus</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Interrupt bus</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No. of General Purpose Regs</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>No. Ext Ints</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Prior'isation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Programmed I/O</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DMA I/O</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Data bus width</td>
<td>16-bits</td>
<td>8-bits</td>
<td>8-bits</td>
<td>8-bits</td>
</tr>
<tr>
<td>Pin count</td>
<td>64</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>PSUs req'd (as well as Gnd)</td>
<td>3(+12v)(+/-5v)</td>
<td>3(+12v)(+/-5v)</td>
<td>2 (+5v)(+12v)</td>
<td>1 (+5v)</td>
</tr>
<tr>
<td>On-chip Oscillator</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Extra features</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Instruction Pre-fetch On-chip facilities (incl RAM)</td>
</tr>
<tr>
<td>Reference (146)</td>
<td>1-1</td>
<td>1-75</td>
<td>1-75</td>
<td>1-117</td>
</tr>
</tbody>
</table>

**TABLE 13.5.1**: COMPARISON OF THE FEATURES OF THE TMS 9900 MICROPROCESSOR FAMILY
### TABLE 13.5.2: TEXAS INSTRUMENTS TMS 9900 MICROPROCESSOR FAMILY SUPPORT DEVICES

<table>
<thead>
<tr>
<th>TMS Family Number</th>
<th>Description</th>
<th>Reference (146)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9901</td>
<td>Programmable Systems Interface (PSI)</td>
<td>Page 2-1</td>
</tr>
<tr>
<td>9902</td>
<td>Asynchronous Communications Controller (ACC)</td>
<td>Page 2-23</td>
</tr>
<tr>
<td>9903</td>
<td>Synchronous Communications Controller (Note 1)</td>
<td>Page 2-57</td>
</tr>
<tr>
<td>9904</td>
<td>4 Phase Clock Generator/Driver</td>
<td>Page 1-355</td>
</tr>
<tr>
<td>9909</td>
<td>Floppy Disc Controller (FDC)</td>
<td>Page 2-83</td>
</tr>
<tr>
<td>9911</td>
<td>Direct Memory Access Controller (DMAC)</td>
<td>Page 2-147</td>
</tr>
<tr>
<td>9914</td>
<td>IEEE 488 GPIB Interface Controller</td>
<td>Page 2-187</td>
</tr>
<tr>
<td>9918A</td>
<td>Video Display Processor (VDP)</td>
<td>Page 2-225</td>
</tr>
<tr>
<td>9927</td>
<td>Video Timer/Controller</td>
<td>Page 2-265</td>
</tr>
</tbody>
</table>

**Note 1:** Some operational and design details are given in Appendix 3, Section A3.5

**Note 2:** This is in fact a general purpose serial interface device capable of handling asynchronous as well as synchronous communications.
CHAPTER XIV

SAC PROTOTYPE VERSION

14.1 INTRODUCTION

This chapter discusses the prototyping investigation undertaken to determine the technical and commercial feasibility of a range of distributed manipulator products based on the pneumatic servo-control and associated technologies which have been discussed earlier.

14.2 INDUSTRIAL COLLABORATOR'S VIEWPOINT

Prior to this work Martonair Ltd were engaged in the manufacture and marketing of pneumatic and pneumomechanical products. This proposed range of products would be the company's first based on microprocessor technology. Before fully committing themselves Martonair decided that a prototype should be built to establish that adequate servo-control performance could be achieved at a suitable price. Furthermore, the functioning of any external interfaces needed to be proven and any technical problems highlighted. Marketing had to be satisfied that packaging would be suitable and that the unit could be made easy-to-use from all points of view. This cautious approach to such a new venture was both understandable and prudent.

The strategy employed in this prototype design and construction phase was to achieve valid results with minimum costs and maximum speed so that important decisions could be expedited. Thus the servo-control algorithm already developed (Chapter IX) was to be used with as few changes as possible to provide the required level of positioning performance. Also any materials and/or techniques, wire wrapping for instance (Section 14.4), were to be used that allowed principles to be displayed but which minimised construction costs.
14.3 PNEUMATICALLY-ACTUATED MOTION ELEMENTS AND SERVO-VALVES

At the outset of the author's research study the industrial collaborator was manufacturing a pneumatically actuated end-stop, conventional robot, the M35 (Plate 14.3.1). During this study this product was later replaced by a range of pneumatically actuated, end-stop modular units (Plates 14.3.2, 14.3.3, and 14.3.4). These units were used in the evolution and testing of the prototype pneumatic motion controllers discussed in Chapter IX.

An obvious route to the availability of a range of servo-controlled mechanical units was to adapt these end-stop modules. This approach offered the advantages of:

i) proven design of an existing product which was intended for module-to-module interconnections.

ii) lessening the required design effort.

iii) obviating the set up of new production facilities.

iv) proven servo-control capability with this pneumo-mechanical arrangement.

v) immediately available, compatible family of end-stop modules.

For these reasons it was decided that the family of pneumatically-actuated servo-controlled modules would be adapted from the existing Martonair end-stop range by the addition of a pneumatic lock and an optical incremental rotary encoder (Chapter XI), with the linear motion of the module being converted by a rack and pinion formed by a portion of a toothed-belt and suitable wheel as shown in Figure 14.3.1.

To control the range of modified motion elements a servo-valve was required. The adoption of the valve already used in the servo-algorithm work described in Chapter IX was decided upon as having the advantages of:

i) proven servo-control performance

ii) very similar to existing, Martonair on-off, solenoid operated, spring return, 5-port, spool valve
Plate 14.3.1: Martonair M35 Industrial Robot

Plate 14.3.2
Martonair Modular Units
Plate 14.3.3
Manipulator with cylindrical configuration constructed from Martonair Modular Units

Plate 14.3.4
Gantry manipulator constructed from Martonair Modular Units
FIGURE 14.3.1
MARTONAIR MODULAR UNIT MODIFIED FOR SERVO-CONTROL

- Toothed belt (attached to moving slide)
- Feedback sensor (Encoder)
- Gear
- Locking device (air operated clamp acting on slide)
iii) minimum effort involved in the set up of manufacturing facilities

iv) simple, low cost and reliable.

Thus effort was concentrated on the design and construction of the control system elements described in the remainder of this chapter.

14.4 ELECTRONICS CONSTRUCTION CAPABILITY

As neither LUT nor Martonair possessed an in-house electronic hardware manufacture capability it was necessary to appoint a contractor to undertake these responsibilities. This took place prior to the completion of design, so that the contractor could make "design for manufacture" contributions.

The search was therefore conducted for a company, ideally with the following credentials:

i) established reputation in hardware design and manufacture

ii) TMS 9900 family experience

iii) ability and willingness to make contributions at the design stage from the manufacturing point of view.

iv) good geographical location, preferably close to Martonair's Farnham base.

v) willingness to proceed to full manufacturing runs

A number of companies were invited to provide quotations for the development work and the manufacture of a batch of up to 10 prototypes. Taking only the companies whose perceived competence, and thus probability of delivering on time, to specification, and within budget, was high, all quotations were considerably in excess of the funds available. Following negotiations one supplier, Nutek (London) Ltd., agreed that 6 units could be produced within the financial limits prevailing. To achieve this the units would be manufactured by wire-wrapping the circuit boards and thus avoiding, at this stage, the high development costs of PCB layout and mask production.
14.5 DIVISION OF RESPONSIBILITY

In order that the three very different organisations involved (LUT, Martonair, and Nutek) would perform and interact in such a way as to manufacture the product on time and to specification, well defined divisions of responsibility were needed.

Martonair managed the project and assumed the role of design authority with inputs from Nutek and LUT. Nutek designed and manufactured the electronic hardware and its housing with design contributions from LUT (circuitry) and Martonair (packaging). LUT, in the person of the author, produced the software necessary, and provided assistance in transferring the technology embodied within that software to both Martonair and Nutek personnel.

14.6 ARCHITECTURE AND PACKAGING

The architecture of distributed manipulator control systems has been considered in detail in Chapter V where it was concluded that a small computer containing a single processor should assume responsibility for the control of at most a few servo positioning axes with a larger, more powerful supervisory device (Chapter VIII) organising the activities of these axis controllers. Chapter IX, Section 9.8 examined the demands on processor time made by the control algorithm to be employed in this prototype version and concluded that the TMS 9995 would be able to handle up to 3 axes with sufficient spare capacity (approximately 25%) to also perform communications (Chapter XII) and other activities necessary to make the servo control facilities useful to the industrial user.

However other considerations, such as power supply configurations, packaging, and their impact on cost, influenced how many axes would be controlled by each axis processor.
<table>
<thead>
<tr>
<th>Scheme and Details</th>
<th>Costs of controlling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 axis</td>
</tr>
<tr>
<td><strong>Scheme I</strong> : 3 axes per processor</td>
<td></td>
</tr>
<tr>
<td>consisting of:</td>
<td></td>
</tr>
<tr>
<td>1 processor board controlling 3 axes (Cost 60)</td>
<td>220</td>
</tr>
<tr>
<td>I/O board(s) for interfacing 3 axes (Cost 100)</td>
<td></td>
</tr>
<tr>
<td>Power supply unit capable of driving up to 3 axes (Cost 60)</td>
<td></td>
</tr>
<tr>
<td><strong>Scheme II</strong> : 1 axis per processor</td>
<td></td>
</tr>
<tr>
<td>consisting of:</td>
<td></td>
</tr>
<tr>
<td>1 processor board controlling 1 axis (Cost 60)</td>
<td>140</td>
</tr>
<tr>
<td>I/O board(s) for interfacing 1 axis (Cost 40)</td>
<td></td>
</tr>
<tr>
<td>Power supply unit capable of driving only 1 axis (Cost 40)</td>
<td></td>
</tr>
</tbody>
</table>

Note: Cost figures are relative and are intended only to enable comparisons and **not** necessarily to represent actual costings.

**TABLE 14.6.1 : POSSIBLE ARCHITECTURAL AND PACKAGING SCHEMES**
Two possible schemes were considered as shown in Table 14.6.1, these representing the most obvious extremes of the range of architectural possibilities available. It was considered that the largest early market for the product would be in simpler applications, frequently requiring only single axis configurations, and Scheme II was therefore clearly attractive, offering the best cost for the most frequent use.

The cost of the Scheme I approach could be reduced for 1 and 2 axis configurations by non-population of the unrequired I/O, but the physical size of the package was also a disadvantage as it was considered important that the electronics should fit into standard 19" instrument racks. Also the product should be perceived to be of minimal complexity by ensuring that the electronics package was physically small, at most 3U (132.5mm) high and single euro-card (approx 200mm) in depth. Furthermore, the Scheme I approach necessitated more complex communications software as effectively 3 individually addressable servo-control software units would be contained in each servo-control computer (SCC). Although it is difficult at the design stage to predict the impact of increased software complexity, possible effects were:

i) increase in memory requirement and hence cost
ii) lengthening of software design, code, and test phases.

Point (i) directly opposes a primary requirement of distributed manipulator controllers (Section 5.2.1) and point (ii) jeopardises the speedy achievement of results crucial to the overall strategy of this phase (Section 14.2). Scheme II was thus selected for the prototyping phase.

This design of servo-control computer, controlling one axis, is referred to henceforth as a "Single Axis Controller" (SAC).

14.7 HARDWARE DESIGN

To comply with the size requirements discussed above the electronic hardware was divided between three boards:
A) Processor, memory, & serial I/O - Figure 14.7.1
B) Rotary optical incremental encoder interface and counter - Figure 14.7.2
C) DAC, servo-amplifier, brake actuator and home sensor interfaces - Figure 14.7.3

These boards were connected using a simple specially designed bus.

The circuits used on board A were designed by Nutek using standard approaches. The board B circuit was based on the encoder circuit discussed in Chapter XI (Figure 11.5.3) and is almost identical to it. The board C circuit uses a standard approach for the 24v dc brake actuator and home sensor interfaces, while the designs of the DAC and servo-amplifier closely follow those given in Chapter X (Figure 10.4.1). The purpose of the home sensor was to provide a reference point for the automatic resetting of the encoder counter circuit.

The serial communications channel would have a twofold purpose:

i) Hardware test using simple test software and a VDU

ii) Demonstration of the SAC's ability to communicate with and be organised by a supervisory computer.

Both RS-232-C (Section 12.4.3) and RS-422-A (Section 12.4.4) links were incorporated and the communications system outlined in Section 12.6 utilised.

14.8 SOFTWARE DESIGN

As appropriate development systems and expertise were already in existence at Loughborough, all code was written by the author in TMS 9900 assembler language (147). This approach was further suggested by the poor support of the relevant Pascal compiler and the need to ensure that the code was small in size and efficient in speed of operation.

Hardware test programs were required in addition to the run-time software.

In the interests of achieving the desired time scales it was decided that no attempt would be made to
FIGURE 14.7.1 : PASC PROTOTYPE VERSION
BOARD A : CPU, MEMORY,
AND SERIAL I/O CIRCUITS
FIGURE 14.7.1 - S & C PROTOTYPE VERSION
BOARD A: CPU, MEMORY, AND SERIAL I/O CIRCUITS
FIGURE 14.7.2: SAC PROTOTYPE VERSION
BOARD B: ENCODER INTERFACE
AND COUNTER CIRCUIT
FIGURE 14.7.3 : SAC PROTOTYPE VERSION
BOARD C : DAC, SERVO-AMPLIFIER, BRAKE ACTUATOR AND HOME SENSOR INTERFACE CIRCUITS
produce fully documented, maintainable software. Thus the software would be of an essentially temporary nature. As a result no listings of this prototype version software are given.

However a modular structured approach was taken to the design of both hardware test programs and run-time software, as this, besides facilitating the production of maintainable code, also expedites its production (Chapter VI, Section 6.7.3).

14.8.1 Hardware Test Program

The structure of the hardware test programs is as shown by Figure 14.8.1 and the accompanying notes. Each individual test is contained in a single procedure which is itself located in one separately assembled code module. All of the test procedures are called from the central module, which is also located in a single, separately assembled code module. A further module contains utility procedures called by other procedures, including those for initialisation and I/O.

User interface is via a simple menu-driven system. The tests were required only to provide proof of functioning.

14.8.2 Run-time Software

The run-time control software consists of service routines for three interrupts:

- level 0 - power-up and processor reset (Fig 14.8.2)
- level 3 - on-board timer (Fig 14.8.3)
- level 4 - external interrupt connected to the UART "received byte" interrupt request. (Fig 14.8.4 a & b)

The design follows a similar modular structured approach to that of the hardware test programs.

The power-up/reset interrupt service routine performs SAC initialisation (once only at power-up) and then forms an infinite execution loop during which background activities are carried out.
START LEVEL 0
INTERRUPT (RESET)
SERVICE ROUTINE [1]

DISABLE ALL MASKABLE
INTERRUPTS [2]

INITIALISE SERIAL
PORT [3]

OUTPUT HEADER AND
MENU TO VDU [4]

INPUT
CHARACTER [5]

Accompanying notes and
brief description of tests are
given on the following page

B?

C?

D?

E?

L?

S?

T?

OUTPUT
INVALID
INPUT
MESSAGE

[4]

[8]

[7]

[9]

[10]

[11]

[12]

BRAKE TEST
COUNTER TEST
DAC TEST
DITHER TEST
DISPLAY TEST
SERIAL I/O TEST
TIMER TEST

FIGURE 14.8.1 HARDWARE TEST PROGRAMS
Figure 14.8.1: Accompanying Notes (including brief description of the hardware tests)

1) Forms an infinite execution loop
2) Interrupt only allowed when explicitly enabled during individual procedures
3) TMS 9902 UART (Appendix A3, Section 3.5) set to 1 stop bit, even parity, 8-bit chars, 9600 bits/sec transmit and receive
4&5) Standard message output and input routines, very similar to TIBUG (150) XOPs 14 and 13 respectively
6) Brake Test Taking into account brake type (active-on active-off) alternatively actuates and de-actuates the brake with an interval of approximately 1 second
7) Counter Test The counter is initialised and zeroed. If this operation fails it is repeated, otherwise the counter is read and the value displayed at approx 0.33 sec intervals
8) DAC Test The DAC output is ramped up and down, cyclically with a period of of approximately 10 seconds
9) Dither Test The dither is switched on and off with an interval of 1 sec
10) Display Test In the interests of expediency, (Section 14.2) displays (LEDs etc.) were not implemented and consequently this is a null procedure
11) Serial I/O Test The TMS 9902 is re-initialised to provide interrupt 4 on its "receive buffer" being filled and the port tested in this mode. On test termination the port is returned to normal mode (no interrupt)
12) Timer Test The on board timer is initialised to provide interrupt every 2 msec. The testers terminal receives 5 sec and 1 min markers. On test termination the timer is de-activated

General During all tests the user may cause termination at any sensible time by pressing "control C".

189
Notes

[1] To allow for 'active on' and 'active off' brakes
[2] In the interests of expediency (Section 14.3) the displays (LEDs etc) were not implemented
[3] To 9600 baud (bits/sec) transmit and receive with 1 stop bit, even parity, and 8-bit characters
[4] Execution following the interrupt service routine commences at the instruction after 'Idle'

FIGURE 14.8.2  RUN - TIME SOFTWARE - START UP AND BACKGROUND LOOP
Notes: [1] The evolution of this algorithm is described in Chapter IX.
Start level 4 byte received interrupt

- Clear all input buffers
- Increment byte received count
- Get data

- Data valid?
  - Yes
    - Addr byte?
      - No
        - Set 'for me' flag
      - Yes
        - This SAC's addr [1]?
          - No
            - Clear 'for me' flag
          - Yes
            - Set 'for me' flag
  - No
    - Set received error flag
    - Clear received error flag

Return from Interrupt

Note: [1] 1st byte in 4 byte protocol (Chapter XII, Section 12.5.2)

FIGURE 14.8.4 (a) RUN-TIME SOFTWARE: MESSAGE RECEIVED INTERRUPT AND INSTRUCTION EXECUTION
Note: [1] This SAC being addressed
[2] 2nd byte in 4 byte protocol
[3] 3rd byte in 4 byte protocol
[4] See Table 15.4.1
The relationship between the level 3, timer interrupt service routine (ISR) performing servo-control and the level 4 ISR is real-time, asynchronous and consequently complex. However the following points, illustrated by Figure 14.8.5, are the most essential:

i) Interaction using dual access data areas is a conventional and relatively simple approach in systems without multitasking operating system support, but which requires considerable care by the programmer to ensure appropriate timing and access rights.

ii) ISR3 has the sole purpose of servo-controlling the module to the position contained in the set point store using the servo-control parameters and velocity profile.

iii) ISR4 handles all communications with the supervisor, including address recognition, data decoding, and execution of the required instructions including those which manipulate the set point store and servo-control parameters. Thus the supervisor is able to control the module's position.

The instructions available to the supervisor programmer to control the SAC form a subset of those provided with the pre-production version (Chapter XV, Section 15.9.10). An example demonstrates the following:

i) The method of Supervisor programming

ii) Some points of the operation of the software

iii) The advantages and disadvantages of the approach

Figure 14.8.6 contains the block diagram of a Supervisor program controlling a single SAC/module to 2 successive set points. The following are worthy of note:

i) At start up the "complete initialisation" instruction must be sent to reset the SAC's RAM-held parameters from ROM, bring the module to the home position, establish the valve null, and correctly zero the position counter. This instruction is in fact a succession of other instructions, each of which is available to the supervisor individually.
RUN-TIME SOFTWARE IN THE SAC PROTOTYPE VERSION RELATIONSHIP BETWEEN ISR3 AND ISR4

**Figure 14.8.5**


- **ISer 3**: DAC, Servo Valve Module, Servo Controller.

- **ISR 3**: RAM data area, Velocity parameters, Profile, ROM parameters, ROM data area, RAM to ROM load, Change data, Parameter instruction, Request status, Other instructions, Teach instructions, RAM data area, Interval timer, Interrupts, Servo control parameters, SAC state, Set point selection, Servo algorithm, DAC, Servo valve module, Servo controller.
FIGURE 14.8.6 EXAMPLE SUPERVISOR PROGRAM FOR MOVING A SINGLE MODULE TO TWO SET POINTS SUCCESIVELY
ii) Each instruction conforms to the protocol discussed in Chapter XII, some requiring data (the set point value for the "load set point" instruction) and some not ("complete initialisation" for instance).

iii) To command the SAC to move its module to a set point requires that two instructions be sent, "load set point" and "go to set point". In this case the latter is superfluous but if broadcast to all SACs (using the "all SACs" address) in multi-SAC systems, it facilitates the synchronisation of move starts.

iv) The SAC software maintains a "Status word" (Figure 14.8.7), which contains limited information concerning the SAC and its module, most significantly in this example, whether the module has achieved position. Thus by requiring the SAC to transmit the status word the Supervisor can establish whether the new set point has been achieved. Alternatively the Supervisor could request the SAC's actual position.

v) Although the necessary SAC instructions exist, it is non-trivial to set up a "teach" function from the Supervisor. Therefore numerical set point values must be estimated, or measured and calculated.

vi) Although Figure 14.8.6 represents a sizeable program to achieve a simple objective, the use of the procedure "move" as suggested in Figure 14.8.6 and shown in Figure 14.8.8 has the advantages of lessening program length and programming effort, and increased eligibility. It is thus possible to create Supervisor systems very similar to conventional robot programming languages such as VAL II (Chapter VII).

14.9 CONSTRUCTION

14.9.1 Hardware

As hardware construction was carried out by Nutek no details of this activity are included. Plates 14.9.1 and 14.9.2 show the individual boards and complete system respectively.
FIGURE 14.8.7 SAC PROTOTYPE VERSION STATUS WORD

<table>
<thead>
<tr>
<th>Bit</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Axis in position</td>
</tr>
<tr>
<td>1</td>
<td>Not used</td>
</tr>
<tr>
<td>2</td>
<td>Axis at 'home' position</td>
</tr>
<tr>
<td>3</td>
<td>Error in received data</td>
</tr>
<tr>
<td>4</td>
<td>Dither on</td>
</tr>
<tr>
<td>5-15</td>
<td>Not used</td>
</tr>
</tbody>
</table>

True when bit set (=1)

FIGURE 14.8.8 USE OF THE "MOVE" PROCEDURE
Plate 14.9.1: SAC Prototype Version Hardware.
(From left to right CPU/Memory(A), encoder counter(B), and DAC/Servo-amplifier(C) PCBs. Letters are those used in Section 14.7)

Plate 14.9.2: Complete Assembled Prototype Version SAC
14.9.2 Software

As is frequently the case software and hardware production were parallel activities and, as hardware was not initially available for software testing, software was developed in the first instance for the TMS 9900 based Texas Instruments' 990/101M single board computer (SBC) (150) using a Texas Instruments' FS 9900/4 microprocessor development system (MDS) (148) with associated tools (TXEDIT text editor and TXMIRA assembler (149), TXSLNK link editor (169) and TMS 9900 real-time in-circuit-emulation (ICE) (170,171)).

This approach simplified the fault finding of the software on proven, reliable hardware so that when work had progressed sufficiently to demonstrate the working of the design, the code was transported to a TMS 9995 based SBC, the MPE IOPROC-A (172), and converted to the use of the TMS 9995's on-board RAM and timer (Appendix A3, Section A3.4). This process was facilitated by the code compatibility of the 2 processors. Reconfiguration of the software for the SAC prototype hardware was then relatively straightforward.

Although the FS990/4 MDS was still suitable for code production using the same editor, assembler and link editor, testing required ICE for the TMS 9995 processor. The Texas Instruments XDS/22 (173) was used for this purpose. The importance of the real-time ICE facility cannot be over emphasised in the development of real-time, interrupt driven, asynchronous software for small devices without operating system support.

14.9.3 Hardware/Software Integration

This activity, in which hardware and software developed by different personnel, and in this case different organisations, comes together, is frequently problematic. Here however, due in no small measure to good design, liaison, and comprehensive logic analyser and ICE facilities, all major difficulties were resolved in less than one day.
14.10 TESTS AND RESULTS

When a number of prototype version SACs were functioning as intended (all six were never fully commissioned due to the problems of fault finding wire-wrapped boards), testing was carried out at LUT, Martonair and jointly. Most of these tests were subjective, being concerned with issues such as ease of use, ease of interface, applicability, and marketability. Discussions regarding ease of manufacture also took place. The results of these activities can be summarised as:

i) concept, basic design, and architecture were sound
ii) packaging was acceptable but could be more modular
iii) servo-performance was acceptable for the anticipated application, but algorithm tuning for various models and situations needed improvement
iv) the RS-232-C/RS-422-A interface worked satisfactorily with occasional data loss problems. It was however considered an over-complex approach in some situations, particularly where a smaller PLC was acting as supervisor, and not sufficiently sophisticated from an error detection point of view in other applications
v) the cost of hardware manufacture even in reasonable quantities and using modern pcb layout and production equipment was likely to be a little too high. Thus simplifications needed to be considered.
vi) Some programming and general user interface problems existed, mainly concerned with "teaching" set points and using such points in supervisory programs.

The overall outcome of the prototype design and build exercise was that in November 1984 Martonair made the decision to market a pneumatically actuated servo-positioning addition to their range of distributed manipulator units with launches in continental Europe at the Hanover Fair in April 1985 and in the UK at the Automan Exhibition in May 1985, thus placing tight time constraints on the development team.
CHAPTER XV

SAC PRE-PRODUCTION AND PRODUCTION VERSIONS

15.1 INTRODUCTION

The team to develop SAC pre-production and production versions comprised Loughborough University, in the person of the author who was responsible for software development, Nutek (London) Ltd, undertaking hardware design and manufacture, and Martonair (UK) Ltd, responsible for project management and ultimate system design. However, Martonair were supported in their responsibility through considerable input from LUT and, to a lesser extent, from Nutek. This chapter describes the work which took place in this context.

Martonair's initial commercial agreement with Nutek was for a hardware batch of 100 single axis units of which between 20 and 30 would be required in the "Pre-Production" form, for the purposes of launch demonstrations, with the remainder delivered, without major hardware modification, as orders arrived. These later units would only be subject to minor software alterations (that is not to overall structure or servo-control implementation) in response to marketing feedback and will henceforth be referred to as the "Production Version". These arrangements were intended to minimise unit costs by enabling Nutek to purchase components in 100-off quantities.

15.2 ARCHITECTURE AND PACKAGING

Chapter XIV (Section 14.6) considered that the architecture and packaging of the control system for the prototype distributed manipulator was sound. However during the pre-launch period, applications feedback had indicated that, whilst installations with single servo-controlled axes would be most frequent, 2 and 3 servoed axes would not be uncommon. Using the arrangement
described in Chapter XIV (one axis per processor per PSU) the cost of multi-servoed axis controllers would be high (Table 14.6.1). To achieve a suitable cost compromise architectural and packaging scheme III of Table 15.2.1 was chosen for the SAC pre-production/production versions: this approach having the additional advantage of allowing the use of either 19-inch rack or stand-alone packaging for single, double, or treble SAC configurations (Figure 15.2.1, Plates 15.2.1 and 15.2.2).

15.3 USER INTERFACE AND PROGRAMMING/TEACH SYSTEM

The only method of interfacing the user to the prototype SAC in general and of teaching set points in particular was via the Supervisor. This approach required, as a pre-requisite, the effort of writing suitable software which would be run in the supervisor. Also previous investigatory studies had not found this approach to be user-friendly (Chapter XIV, Section 14.10).

To resolve this difficulty a decision was made to provide the SAC with its own, simple, user interface and set point teaching system. However no more sophisticated facilities would be offered in addition to those of the prototype version as this would be:

i) impractical in the timescales available
ii) probably not cost-effective
iii) not consistent with the overall system philosophy of the SAC's being responsible only for positioning, with the Supervisor undertaking responsibility for decision making (Chapter V Section 5.4.3).

Such user interfacing and teaching facilities would necessitate the provision of:

i) a user terminal
ii) user interface software
iii) teaching software
iv) local set point storage area
v) battery-backing for at least (iv)
vi) sufficient extra memory to accommodate (ii), (iii) - ROM and (iv) - RAM.
<table>
<thead>
<tr>
<th>Scheme and Details</th>
<th>Costs for controlling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 axis</td>
</tr>
<tr>
<td>Scheme I : 3 axes per processor consisting of:</td>
<td></td>
</tr>
<tr>
<td>1 processor board controlling 3 axes (Cost 60)</td>
<td>220</td>
</tr>
<tr>
<td>I/O board(s) for interfacing 3 axes (Cost 100)</td>
<td></td>
</tr>
<tr>
<td>Power supply unit capable of driving up to 3 axes (Cost 60)</td>
<td></td>
</tr>
<tr>
<td>Scheme II : 1 axis per processor consisting of:</td>
<td></td>
</tr>
<tr>
<td>1 processor board controlling 1 axis (Cost 60)</td>
<td>140</td>
</tr>
<tr>
<td>I/O board for interfacing 1 axis (Cost 40)</td>
<td></td>
</tr>
<tr>
<td>Power supply unit capable of driving only 1 axis (Cost 40)</td>
<td></td>
</tr>
<tr>
<td>Scheme III : 1 axis per processor with large PSU consisting of:</td>
<td></td>
</tr>
<tr>
<td>1 processor board controlling 1 axis (Cost 60)</td>
<td>160</td>
</tr>
<tr>
<td>I/O board for interfacing 1 axis (Cost 40)</td>
<td></td>
</tr>
<tr>
<td>Power supply unit capable of driving up to 3 axes (Cost 60)</td>
<td></td>
</tr>
</tbody>
</table>

Note: Cost figures are relative and intended only to enable comparisons and not necessarily to represent actual costings.

TABLE 15.2.1 : POSSIBLE ARCHITECTURAL/PACKAGING SCHEMES
Components

Single SAC in stand-alone package
(Plate 15.2.1)

Single SAC in 19 Inch rack

Double SACs in stand-alone package
(Plate 15.2.2)

Triple SACs in 19 Inch rack

FIGURE 15.2.1 SAC PRE-PRODUCTION/PRODUCTION VERSION HARDWARE MODULARITY
Plate 15.2.1: Single SAC in Stand-alone Package

Plate 15.2.2: Double SAC in 19-inch Rack
The user interface would include features facilitating the tuning of the servo-control for particular modules and applications. The terminal was needed to act as a "teach pendant" and had therefore to be hand-held. The single line, 16 character display "Microscribe" (Plate 15.2.3), being the least expensive (at the time costing approximately £200 in small quantities) with reasonable weight and shape, was selected.

Battery-backing introduced a small cost penalty but presented no other major difficulties and was readily implemented by Nutek. A single 28 pin 2732 EPROM IC provided 4K bytes of ROM in the prototype version. No extra board cost would be incurred in providing the production version with up to 32k bytes as the 27256 EPROM also has 28 pins and would require only two additional PCB tracks. However the cost of the 27256 (and 27128) was (in 1984) considerably in excess of the 2764 EPROM and this was the factor determining the ROM memory size of 8k bytes.

Besides providing a much improved user interface and set point teaching system, these changes also made possible a range of operational procedures:

i) transfer of set points back to the supervisor and their use as in the SAC Prototype version
ii) use of set points, in the SAC, as taught
iii) manipulate (ie edit) the set points in the SAC and use the resulting set point sequences from the SAC
iv) transfer set points to the supervisor, manipulate, and reload into the SAC's local set point store.

To fully implement (iii) would over complicate the SAC's user interface and necessitate considerable extra software, but as this would adversely effect memory size and development time, it was rejected. Facilities for procedures (i) and (iv) were included by adding appropriately to the SAC's instruction set (Section 15.9.10).

Procedure (ii) could be achieved by a supervisor requesting the SAC to move to a locally held set point referred to by number (eg move to set point 1, move to set point 2) Such instructions could be transmitted using
the serial interface or by a simple address/control bus interface.

15.4 SUPERVISOR/SAC INTERFACES

The approach taken in the prototype version had been found to be basically sound, but three modifications were required (Chapter XIV, Section 14.10) to:

i) correct occasional data loss

ii) provide more sophisticated error detection

iii) provide a simpler interface.

The first was accomplished through a modification to the software structure (Sections 15.9.2 and 15.9.5), whereas the second of the above points could be addressed in later production versions as this required just additional software and is needed only in more sophisticated applications.

Many industrial PLCs possess serial communications interfaces but as the use of such interfaces is at least cumbersome (Chapter VIII, Section 8.5.1) it was consider-
ed highly desirable to offer a more suitable interface for PLCs, albeit less flexible and sophisticated. The use of a SAC teaching system together with local set point storage as described in the previous section offered a means of achieving such a Supervisor/SAC interface. The principle of the simple parallel bus interface chosen is shown in Figure 15.4.1 and has the following features:

i) readily used by (but not confined to) PLC supervisors
ii) simple to understand
iii) simple and safe to program using ladder diagrams
iv) requiring only small additions to the original SAC hardware and software
v) capable of referring to a maximum of 32 SAC stored set points
vi) sequence of use not restricted to order of teaching thereby obviating the need for full SAC set point editing facilities. (Section 15.3)
vii) only available to instruct the SAC to initialise, move to set points, and to receive "in position" signals.
viii) all lines active at 24v dc levels to facilitate use over several metres and in industrial environments
ix) restricted to 8 lines to reduce complexity and minimise hardware costs.

15.5 SERVO-CONTROL

As the algorithm evolved in the earlier work (Chapter IX) and incorporated into the SAC prototype (Chapter XIV, Section 14.2) had offered adequate performance, this would again be used. There being only minimal software effort required for this course of action the meeting of timescales was assisted.

15.6 HARDWARE DESIGN

The hardware design also followed that of the prototype version being based on the circuits developed at LUT and laid out by Nutek. The following additions were required to achieve the enhancements discussed in
### Table: Supervisor to SAC Signal Significance

<table>
<thead>
<tr>
<th>Direction to SAC</th>
<th>Signal Name</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLCIF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLCIF2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLCIF3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLCIF4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLCIF5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLCIF6</td>
<td></td>
<td>Move to set point</td>
</tr>
<tr>
<td>PLCIF7</td>
<td></td>
<td>Initialise SAC command</td>
</tr>
</tbody>
</table>

**SAC to Supervisor**

| PLCIF8           | SAC module at set point |

#### Diagram:

- **PLCIF1 - 5**
  - Valid SAC set point address

- **PLCIF8**
  - Valid SAC set point address

- **PLCIF7**

- **PLCIF8**
  - Module in motion
  - Module arrives at set point
  - Module in motion
  - Module arrives at set point

---

**Note 1:** SAC will not move module to a second set point until a 'low' appears on PLCIF6 and it is necessary to ensure that PLCIF6 does not go 'high' until PLCIF8 has gone 'low'.

---

**FIGURE 15.4.1** SUPERVISOR / SAC PLC INTERFACE
Sections 15.3 and 15.4:
  i) Additional ROM
  ii) PLC interface circuits
  iii) Battery-backing

Furthermore, as the need to allow for the potential use of the 0-1A active range valves (Chapter X, Section 10.4.1) could now be dispensed with, some components were derated, lowering hardware cost and allowing a 24v power supply to replace the previous 30v one. This simplification also assisting with packaging (Section 15.2).

The interface to the "home" reference proximity sensor was retained, as were both the RS-232-C and RS-422-A communications interfaces. The former being intended for use both in SAC/Supervisor communications and in interfacing to the user terminal. Changeover between "run" and "teach" modes for both Supervisor interfaces would be achieved using a front-panel switch. Selection of the supervisor interface itself would be via jumpers on the CPU/memory PCB.

15.6.1 74LS2000-based Encoder Counters

As suggested in Chapter XI, Section 11.7, the introduction of the 74LS200 IC provided the opportunity to considerably reduce SAC hardware cost. This approach was proven using the circuit shown in Figure 15.6.1 as a modification to the prototype SAC and was then adopted.

15.6.2 Division

The 74LS2000-based encoder counter allowed the hardware to be fitted on to 2 PCBs, compared to 3 for the prototype version:

A) Figure 15.6.2: CPU, ROM, Battery-backed RAM, mode selection switch, 74LS2000 encoder, supervisor selection jumpers, SAC address setting switches, serial interface UART, RS-232-C and RS-422-A transceivers

B) Figure 15.6.3: DAC, servo-amplifier (power amplifier mounted on accompanying heat-sink, Plate 15.10.2), all 24v dc interfaces (including PLC interface lines, home sensor and brake actuator)
FIGURE 15.6.1: 74LS2000-BASED ENCODER COUNTER TEST BOARD
FIGURE 15.6.2: SAC PRE-PRODUCTION/PRODUCT-ION BOARD A: CPU AND MEMORY CIRCUITS
FIGURE 15.6.3: SAC PRE-PRODUCTION/PRODUCT-ION BOARD B: I/O CIRCUITS
FIGURE 15.6.3: SAC PRE-PRODUCTION/PRODUCTION BOARD B: I/O CIRCUITS
15.7 SOFTWARE DESIGN

All software was written in TMS 9900 Assembler for exactly the reasons discussed in the SAC prototype version (Section 14.8). Also the experience gained in the prototyping phase and some of the code produced were of direct relevance at this stage.

Modular structured methods were used in the design of all software. The advantages of such approaches have been previously discussed in Chapter VI, Section 6.7.3, and may be summarised as:

i) division of large complex pieces of software into comprehensible, and generally manageable modules of code
ii) separate production and testing of such modules
iii) ability to change modules without the potential hazards of knock-on effects and unforeseen consequences
iv) production of software on time, in budget, and to specification
v) Post-commissioning/installation maintainability.

In the interests of ease of maintenance, care was also taken in order that, within the limits of the six characters allowed by TMS 9900 assembler, procedures, jump/branch destinations, data locations and areas, had titles representing their function.

These techniques have been proved to have been successfully applied in this case as the Martonair staff responsible for the software have been able to carry out successful software maintenance and upgrading.

15.8 HARDWARE TEST PROGRAM DESIGN

The purpose and structure of this test program was exactly similar to that of the prototype version (Section 14.8.1). Each test is contained within a separately assembled module. The module containing the user interface menu driver calls the procedures implementing the various tests as required. The purposes of the tests are similar to those of the prototype version with some additional facilities. Table 15.8.1 lists the modules
<table>
<thead>
<tr>
<th>Module (Note 1)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELCTT</td>
<td>Power-up ISR, System initialisation, and menu driven operator interface</td>
</tr>
<tr>
<td>ADRMAP</td>
<td>Off-chip RAM memory and I/O address map</td>
</tr>
<tr>
<td>BBRTST</td>
<td>Battery-backing test (note 2)</td>
</tr>
<tr>
<td>BRKTST</td>
<td>Brake test</td>
</tr>
<tr>
<td>CNTTST</td>
<td>74LS2000 based incremental encoder interface and counter test</td>
</tr>
<tr>
<td>CPURUN</td>
<td>CPU running indicator test (indicator (omitted for production versions)</td>
</tr>
<tr>
<td>DACTST</td>
<td>DAC test</td>
</tr>
<tr>
<td>GENP</td>
<td>General utility procedures</td>
</tr>
<tr>
<td>INITST</td>
<td>Initialisation button test</td>
</tr>
<tr>
<td>HEXIO</td>
<td>Hexadecimal I/O procedures</td>
</tr>
<tr>
<td>MEMTST</td>
<td>Off-chip RAM test (note 2)</td>
</tr>
<tr>
<td>OPMSEL</td>
<td>Operation mode selection switch (run/teach) and jumpers (supervisor i/f) test</td>
</tr>
<tr>
<td>OVRTMP</td>
<td>Valve current overload sensor test (omitted for production versions)</td>
</tr>
<tr>
<td>PLCADR</td>
<td>PLC interface address &quot;bus&quot; test (Figure 15.4.1 PLCIF 1-5)</td>
</tr>
<tr>
<td>PLCTST</td>
<td>PLC interface control &quot;bus&quot; test (Figure 15.4.1 PLCIF 6-8)</td>
</tr>
<tr>
<td>REFPRX</td>
<td>Home reference proximity sensor test</td>
</tr>
<tr>
<td>SACADR</td>
<td>SAC address DIL switches test</td>
</tr>
<tr>
<td>SIOTST</td>
<td>Serial I/O port and &quot;byte received&quot; ISR (level 4) test</td>
</tr>
<tr>
<td>TIMTST</td>
<td>TMS 9995 on-board timer ISR (level 3) test</td>
</tr>
</tbody>
</table>

Notes
1) Software listing are given in Appendix 4 and Table A4.1 gives the page numbers of these modules.
2) The "MEMTST" module tests the memory retention capability of each bit of the off-chip RAM whilst mains power remains on, whereas the "BBRTST" module tests retention through mains power-down.

**TABLE 15.8.1 : SAC PRE-PRODUCTION VERSION HARDWARE TEST PROGRAM CONSTITUENT MODULES AND THEIR FUNCTIONS**
which constitute the pre-production version tests and outlines their functions.

15.9 RUN-TIME SOFTWARE DESIGN AND STRUCTURE

15.9.1 Requirements

For the pre-production and production version SACs the requirements of the run-time software can be summarised as being the original functions provided by the prototype version, that is:

i) servo-control

ii) supervisory communications protocol

iii) execution of supervisory commands for
- SAC and module initialisation
- set point download
- returning of SAC in-position, position, and status data,

which should be complemented by the following additional functions:

i) correction of the data loss problem (Chapter XIV, Section 14.10)

ii) integrity checking of battery-backed RAM

iii) provision of the PLC interface (Section 15.4)

iv) Microscribe user interface and set point teaching facility (Section 15.3)

v) operational mode switching (between run/teach and PLC/serial interfaces)

vi) the execution of extra supervisory commands for the manipulation of the locally (to the SAC) stored set points by the supervisor

vii) improvements to SAC/module error detection and the provision of error report requesting instructions.

15.9.2 Structure

For the pre-production and production version SACs the structure of the run-time software was similar to that of the prototype version, with three interrupt service routines at levels 0, 3, and 4:
**Level 0** (Power-up/reset)
initialisation and infinite execution background

task loop
**Level 3** (Timer)
servo-control at regular, but programmable intervals
**Level 4** (Byte received)
reception of each byte from the supervisor via the
serial interface and UART.

Figure 15.9.1 illustrates schematically this structure
and is to be contrasted with Figure 14.8.5 which gives
the structure of the prototype version run-time software.
The following three sections (15.9.3 to 15.9.5) give
overviews of these interrupt service routines, whilst
sections 15.9.6 to 15.9.11 briefly describe the function-
ality available to the user that is in addition to that
of the prototype version. More details of the run-time
software, its design, operation, and listings, are given
in Appendix 5.

15.9.3 **ISR0 and the Background Task Loop**

On receipt of the power-up/CPU reset interrupt
(level 0), initialisation is performed followed by the
indefinite execution of the background task loop, whose
functions are performed by directly or indirectly called
procedures.

Figure 15.9.2 contains the flow chart of the
background task loop, although it should be noted that
the code which it represents is both of considerable size
and complexity, and thus the flow chart gives only an
overview.

15.9.4 **Timer Interrupt Service Routine (ISR3)**

This service routine contains the servo-control
software implementing motion control (Chapter IX). It is
in essence identical to that of the prototype version
(Figure 14.8.3) and communicates with the background task
loop in exactly the same fashion as for the prototype
version, that is using common data areas containing set
point and in-position signals.
Start
ISR0

Hardware/software/mechanical initialisation

Operational mode termination, initialisation & selection (Section 15.9.7)

Check reference proximity sensor and update status word bit

Selected op. mode valid?

Yes

Operational Mode Execution

Supervisor Run

Instr present in FIFO (Section 15.9.5)?

Yes

Execute supervisor instruction

Supervisor Teach

PLC run

Perform PLC "bus" protocol (Section 15.4 & Figure 15.4.1)

PLC Teach

Run Microscribe user interface, receive and user commands (Section 15.9.8)

Operational Mode Execution

No

Initial button operated?

Yes

Perform mechanical initialisation for selected operational mode

No

Operational Mode Execution

Note: This flowchart essentially describes the code contained in the module "SCAN" (Appendix 5)

FIGURE 15.9.2 OVERVIEW FLOWCHART OF ISR3 AND THE BACKGROUND TASK LOOP
15.9.5 Byte Received Interrupt Service Routine (ISR4)

The occasional data loss during serial communications was caused by the structure of the prototype version software (Chapter XIV, Section 14.8.2), in which the interrupt service routine handling the serial communications (ISR4) also executed the instructions received, so that if the supervisor sent a second instruction before the first had been fully processed by the SAC, part or all of the second instruction was lost by the SAC (Figure 15.9.3(a)).

This problem was solved by a change in structure so that, in the pre-production run-time software, ISR4 handles only the communications protocol and puts each received instruction into a first-in first-out (FIFO) buffer. These instructions are then removed and executed by the background task. The effect of this structure on instruction reception is shown in Figure 15.9.3(b).

The disadvantage of this approach is that, because the FIFO buffer is emptied and instructions executed by a background task, it is possible for the buffer to overflow. Even though the buffer has been set to hold a maximum of 30 supervisor instructions, some discipline is required on the part of the supervisory programmer to ensure that buffer overflow does not occur. The supervisory programmer is however assisted in this, as, whenever an instruction is issued requiring a SAC reply (eg request status), the supervisor would normally wait before proceeding and thus the buffer will be emptied before more instructions are received: thus overflow will not occur.

This FIFO buffer represents the means of communication between ISR4 (receive instruction) and the background task loop. As such, it is a commonly accessible data area and care has therefore been taken to ensure that reading of the buffer is prevented when an instruction is being received (Appendix 5, module "READMS").
a) SAC Prototype version serial communications data loss

b) SAC Pre-production version serial communications avoiding data loss

FIGURE 15.9.3  EFFECT OF SOFTWARE STRUCTURE ON THE SERIAL DATA LOSS PROBLEM
15.9.6 Battery-backed RAM Integrity Checking

Although all of the off-chip RAM is battery backed, in the interests of speed of operation, the integrity of only the servo-control parameters and the locally stored set points is validated.

Each update of this data is accompanied by a recalculation of the checksum of the area so that at power up the checksum may again be calculated and compared to that at power down.

15.9.7 Operation Mode Changeover

As shown in Figure 15.9.2 a number of operational modes are available, so that either an industrial computer or a PLC supervisor (Chapter VIII) may be used and "teach" or "run(set)" modes may be selected for each.

The supervisor/interface type is selected by jumpering on the processor/memory board (Section 15.6.2 board A, Figure 15.6.2) so that a change of supervisor may only be accomplished by power down and board removal. By contrast teach/run(set) mode is selected by a front panel switch so that the user may utilise the Microscribe interface (Section 15.9.8) to teach the required set points and then change to run mode and move to these locally held set points using instructions issued via the PLC or serial interfaces.

As shown by Figure 15.9.2 the operation mode selection procedure is called from the background task loop. The flow diagram of this procedure is given in Figure 15.9.4. Mode termination and initialisation provide for the tidying of the serial byte received interrupt and other software aspects.

This operational mode procedure returns to the main background task loop with pointers to the appropriate mechanical initialisation and execution procedures so the background task loop itself can call and execute them as shown in Figure 15.9.2.

Although Figure 15.9.2 shows only four possible execution modes, there are in fact 8 available. The four additional ones being "teach" and "run(set)" in each of
Note: This flowchart essentially describes the code contained in the module 'OPMODE' (Appendix 5)
the "Simple PLC" and "PLC Intelligent Terminal" modes.

As some fears were expressed that even the PLC bus interface might be too complex in some situations, the first of these modes provides for a simpler PLC interface. This simple interface consists of only one line in each direction, the SAC moving to the next locally held set point on each rising edge of the Supervisor to SAC line, and signalling in-position on the other.

The "PLC Intelligent Terminal" mode provides for the use of the PLC bus in commanding the movement of the SAC/module to set points (that is in run(set) mode) together with the use of an industrial computer based device in teach mode so that the complete range of SAC facilities (eg set point editing) are available to users of the PLC bus interface for more complex situations.

As these operational modes have only been infrequently used they are omitted from Figure 15.9.2.

15.9.8 Microscribe User Interface

Provision of a Microscribe user interface represents the most significant software change in generating the SAC pre-production and production version software (Appendix 5, modules "MENU", "MSFBI", "MSPARA", "TCHDSP", "TCHMEN", "TCHSET", "TEACH1", "TEACH2", and "VDUIO"). A menu driven, user friendly interface is provided and used by both main run(set) execution modes to perform teaching (Figure 15.9.2). The following functions are offered:

i) initialisation of the module, including bringing the module to the "home" position, and zeroing the encoder counter

ii) tuning of the servo-control by the display and update of the algorithm parameters and subsequent testing of the module performance

iii) teaching locally held set points, including storing, deleting, and moving to set points either individually or cyclically

iv) inspection and alteration of the SAC's internal condition (Section 15.9.11) including, ROM to
RAM servo-control parameter download, examination and resetting of inhibits, display of absolute module position, position error, and status word (and thus brake on/off, in position, and home/not home).

15.9.9 PLC Interface

The PLC interface procedure (Appendix 5, module "PLC") implements the timing diagram of Figure 15.4.1 and also includes code for the "Simple PLC" interface described in Section 15.9.7.

15.9.10 Supervisor Serial Interface Instruction Execution

This function is performed by the module "READMS" (Appendix 5), which removes instructions from the FIFO buffer (Section 15.9.5 and Appendix 5, module "RXINSB") and calls the procedures required for processing them.

A total of 41 instructions are provided for the supervisory programmer. These are grouped by function as shown in Table 15.9.1. This categorisation was useful in facilitating the enabling/disabling of each group so that any group of instructions that were inappropriate in a given situation (e.g., groups 4 and 8 before initialisation) could be disallowed.

15.9.11 SAC Status and Error Conditions

The SAC status is contained in a 16-bit word in exactly the same way as that used in the prototype version. However, error reporting is more sophisticated to allow Supervisors using the serial interface to detect and intelligently recover from errors. More details of the use of the bits in the status word and the error conditions reported are contained in Appendix 5 (modules "STATUS" and "ERROR" respectively).
<table>
<thead>
<tr>
<th>Group No.</th>
<th>Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initialisation</td>
<td>Complete startup initialisation, software only initialisation, counter reset and load, ROM to RAM download</td>
</tr>
<tr>
<td>2</td>
<td>Parameter</td>
<td>Servo-control parameter change</td>
</tr>
<tr>
<td>3</td>
<td>Feedback</td>
<td>For requesting the SAC to return data including status (and thus in-position), errors, and actual position</td>
</tr>
<tr>
<td>4</td>
<td>Teach I</td>
<td>For providing teach facilities via the supervisor and loading/manipulating set points stored locally to the SAC (some duplication with Group 8)</td>
</tr>
<tr>
<td>5 &amp; 6</td>
<td>Teach II &amp; III</td>
<td>Reserved for additional teach instructions</td>
</tr>
<tr>
<td>7</td>
<td>Equipment control</td>
<td>Direct control of actuator, servo-valve, and brake (ie without servo-control)</td>
</tr>
<tr>
<td>8</td>
<td>Servo-control &amp; set-point</td>
<td>Activate/de-activate servo-control, local set-point manipulation (some duplication with Group 4), and set-point download</td>
</tr>
<tr>
<td>9</td>
<td>SAC internal control</td>
<td>For clearing errors and any associated inhibits</td>
</tr>
<tr>
<td>10-15</td>
<td>Not defined</td>
<td>Not used</td>
</tr>
<tr>
<td>16</td>
<td>Permanently enabled</td>
<td>The emergency stop instruction which may not be disabled</td>
</tr>
</tbody>
</table>

**TABLE 15.9.1 : SUPERVISOR SERIAL INTERFACE INSTRUCTIONS**

227
15.10 CONSTRUCTION

15.10.1 Hardware
This activity was again carried out by Nutek and thus no details are included. Plates 15.10.1 and 15.10.2 show the construction of the individual PCBs and the power amplifier heat sink discussed in Section 15.6.2. The packaging of the SACs is shown above in Plates 15.2.1 and 15.2.2.

15.10.2 Software
All code was again produced using the Texas Instruments FS9900/4 microprocessor development system (148) and associated tools, as described for the prototype version (Chapter XIV, Section 14.9.2) and took place in two stages.

The hardware test programs were written in the first place and themselves tested as far as possible on the existing prototype version hardware using the Texas Instruments XDS/22 TMS 9995 processor ICE (173). The modular design methodology meant that the memory and I/O maps could be located in a separate module, thus removing the need for multiple copies of addresses. This facilitated the testing of the test software using the prototype version hardware with its radically different I/O memory map, as transfer to the pre-production version hardware merely required the replacement of one module and re-linking. This procedure carried considerable certainty of correct functioning, even though retests with pre-production version hardware were of course impossible.

These programs were put into EPROM and used in hardware construction. Taking into account that it was not possible to comprehensively test the test programs the fact that they were used extensively by the hardware engineers almost without modification during construction bears witness to the success of the modular structured design approach and the friendliness of the user interface.
Plate 15.10.1: Pre-Production/Production Board A: CPU and Memory Circuits

Plate 15.10.2: Pre-Production/Production Board B: I/O Circuits

Note: "Board A" and "Board B" refer to the notation used in Section 15.6.2
The run-time software was produced in the second stage. As production version hardware was not available until the later phases, much work was performed using the prototype hardware. The XDS/22 ICE was invaluable during all stages of the software production.

15.10.3 Hardware/Software Integration

As with the prototype version, due to the comprehensive hardware test programs, good design, and proper liaison this phase was completed in less than one day, despite the software and hardware developers being a considerable distance apart and being able to meet previously on only one occasion. Doubtlessly the prototyping exercise also had considerable benefits in this phase.

15.11 RESULTS

The SAC pre-production and production versions were completed on time and to specification. From the software viewpoint, the problems of the prototyping version had been successfully addressed. The data loss problem solved, battery-backed RAM facilities provided, including local set point storage, and a PLC interface made available.

Since the product launch in April 1985 the Microscribe user interface has proved popular with SAC operators, and both the serial and PLC interfaces have been found to function well and to be easy to use.

Of the total range of facilities provided, the instruction grouping capabilities have not been used, nor yet have supervisor programs been written with sufficient "intelligence" to require the SAC's level of error reporting. However, neither of these points have hindered any other aspect of the SAC's functioning.

The comparatively short time in which the pre-production version SAC software was evolved from the prototype software affected adversely only the quality of documentation of the code. The structure of the code, even with considerable hindsight, has been found to be
good, and indeed the soundness of the structure and the use of meaningful names within the code and general design/coding approach has greatly facilitated the retrospective familiarisation and documentation exercise carried out by Martonair
CHAPTER XVI

CONCLUSIONS AND RECOMMENDATIONS

This project has been concerned with concepts central to distributed manipulators and modular robots. To investigate these ideas a range of servo-controlled pneumo-mechanical modules has been designed and built. Correspondingly, a suitable controller architecture has been devised and implemented in which motion control of each single degree of freedom is accomplished by a microprocessor based single axis controller (SAC) whilst a supervisory computer organises and monitors the activities of the user defined manipulator through these SACs.

The demands of a project of this magnitude and scope were beyond a single individual's efforts and the work of other members of the research group must be recognised.

This thesis has itself been directly concerned with a study of the features of distributed manipulators/modular robots, the design of an appropriate control architecture, the parts of this structure concerned particularly with point-to-point servo-control of pneumatic drives, and the methods involved in implementing motion controllers of this type in a low cost, microcomputer with user-friendly interfaces.

The work reached a successful conclusion in April 1985 with Martonair's worldwide launch of a range of servo-controlled pneumatically actuated modules which have since facilitated the construction of a number of distributed manipulators/modular robots within various manufacturing companies. Over the last two 2 years these modules have been used in a number of application areas, the range and sophistication of which has been somewhat limited due mainly to non-technical circumstances. However, in so far as the concepts have been tested, they have been proven, and simple manipulator forms of this type are finding wider acceptance.
The author was intimately concerned with devising a suitable control hierarchy and evolving pneumatic motion control elements over a three year period. A pivotal role was also assumed during the commercialisation of some of the prototype designs evolved during the research period. These commercial forms of the SAC hardware and software were completed on time, in a short period, and performed the functions for which they were designed. This functionality included a PLC interface which has been thoroughly tested by, and proved popular with, industrial users. The more sophisticated RS-232-C/RS-422-A serial link has not been so comprehensively tested in industrial applications but the protocol is being shown to function well and to be easy to use. Furthermore, little software maintenance has been necessary, pleasantly surprising when it is considered that many of the SAC functions were largely untested at launch in April 1985. Thus from the technical viewpoint this study has been a success in that it has provided a major UK company with a new product range (the first of its type worldwide) and it has made available a vehicle for the company to enable them to move into higher technology markets. Moreover, pneumatic-mechanical and control system elements have been made available with which it has been possible to begin to further explore the concepts of modular robotics and distributed manipulators which were introduced in this thesis.

Inevitably, associated with the evolution of new, complex products, there are technical and other problem areas which must be addressed before pneumatic motion controls rival fully their electric and hydraulic counterparts or before distributed manipulators/modular robots become a universal solution in constructing flexible manufacturing machinery. Here we will concern ourselves with the technical issues.

Work in a number of areas, directly concerned with pneumatic motion control, needs to be pursued. Pneumatic servo-control can be significantly improved in terms of both dynamic and static performance by achieving an
optimum choice of motion control parameters. The complex non-linearities associated with pneumatic drives necessitates complex control algorithms. To facilitate more widespread industrial acceptance, the tuning of a particular algorithm in a user-defined physical arrangement can be made more user friendly by reducing the number of parameters needing to be specified, possibly by the use of suitable software running on a supervisory computer to which a user would only need to give meaningful parameters such as load, desired speed/cycle time, and repeatability.

The control level of pneumatic motion control corresponds only to the point-to-point positioning of payloads. This thesis has illustrated the complex nature of the algorithms required to achieve this control. In application areas requiring point-to-point positioning, pneumatic motion controllers can demonstrate good performance/cost ratios compared to their electric and hydraulic equivalents. However, during the prototype design stages of this work it was not possible to achieve good speed control, and algorithms capable of accomplishing both position and velocity control would widen the application area for pneumatic servos and permit the full emergence of pneumatic servo drives.

Of lesser significance, a better theoretical basis for the determination of the values of dither frequency and amplitude is required as these have been chosen pragmatically and, to some extent, for convenience (Chapter X).

In terms of the current microprocessor based implementation of the pneumatic servo-control, several areas for advancement can be identified, including improved communication protocol (Chapter XIV) in order that the integrity lapses likely in the shopfloor environment can be automatically detected and corrected. Also other packaging arrangements might have benefits, particularly if miniaturisation, possibly using surface mount technology (SMT), could enable the electronic hardware to be built into or on to the mechanical element. Furthermore,
16-bit integer arithmetic limits the stroke length/resolution which can be offered, and the cost/benefit trade-off of using floating point calculations and 32-bit position feedback interfaces are worthy of investigation.

With reference to the emergence of distributed manipulators/modular robots as a universal solution in constructing flexible machines, there is a need for a comprehensive family of mechanical modules exhibiting a wide variation of factors such as length of stroke, load carrying capability, nature of actuation (rotary or linear), and resolution. It can be reasonably anticipated that such a family would include dc electrically and hydraulically actuated modules together with a variety of lower order mechanical primitives such as the so-called "intelligent cylinders". Some examples of the latter have been devised by the Loughborough University/Martonair consortium based on cylinders (some rodless) with integral position feedback sensors, but not comprising load bearing slideways. Such devices represent a lower order primitive form than a conventional pneumatic module and allow reduced cost per axis and/or more appropriate mechanical properties such as increased stroke length (provided that position resolution remains acceptable). Clearly however, a wide variety of mechanical modules already exist from various commercial sources which, with the addition of suitable feedback sensors, can complement the Martonair range.

A major deficiency in the current implementation of modules and controls is the lack of a readily available supervisory controller. The issues associated with the very complex area of the design and construction of a supervisor have been discussed. Chapters IV and V suggested that the control of distributed manipulators had much in common with the control of general manufacturing machines, and in an ideal world an overall scheme, perhaps along the lines outlined in Chapter V, ought to be investigated in the first place. Programming systems both for distributed manipulators and more general manufacturing machines would be required. Because of the
complexities of such systems, the route suggested in Chapter VIII of enhancing existing operating and programming languages (for instance UNIX and C) has much to recommend it. Because of the undefined structure of the distributed manipulator, the need for user reconfigurations, and the requirement for n-axis synchronisation (including software cams and straight line and other interpolated motions) of these manipulators demonstrating user defined kinematics, this work will be complex. As a result it is probable that such advances will be implemented in a piecemeal fashion and thus of considerable importance in the design is the requirement to anticipate change and upgrade. For instance 3-D solid modelling and associated graphics systems may, in the not too distant future, become available for distributed manipulators as indeed they are already for more conventional robots.

Many machine design problems are associated with configuring mechanisms which are mechanically optimised. This design problem is simplified to some extent through the use of conventional off-the-shelf industrial robots, but with inherent performance penalties resulting from a lack of mechanical optimisation. Computer based design facilities are required to ease these design problems when using distributed manipulators/modular robots for which many mechanical structures can be evolved to accomplish a specific manufacturing function. Studies have been initiated within the Loughborough University modular robots research group aimed at addressing some of these problems and include the use and development of solid modelling simulator/graphic facilities of the type mentioned in the previous paragraph to achieve manipulator and workplace kinematic design. Such facilities should eventually encompass kinematic and dynamic design and also address the problems of configuring appropriate control system structures from new generations of control system primitives (both hardware and software) as they evolve.

This project, which has been associated with the
activities of the distributed manipulator/modular robotics research group at Loughborough University, has provided a methodology in the design of control structures for distributed manipulators/modular robots and has resulted in the commercial launch of a range of such products. Many of the recommendations for further work discussed herein are already underway and will, it is hoped, result in wider use of the distributed manipulator and its associated ideas and thus the greater use of flexible automation with accompanying benefits to UK manufacturing industry and society in general.
GLOSSARY OF TERMS

Conventional Robot
Any pedestal mounted industrial robot which is available in only one or at most a few configurations of kinematic chain.

Distributed Manipulator or General Manipulator
Robotic device constructed from a family of modular elements but not necessarily all mechanically connected.

General Manufacturing Machine
Any automatic machine used in an automated production facility.

Modular Robot
Robot having any number of axes all mechanically connected and constructed from a family of modular elements.

Robot Axis Group
Collection of modular elements not mechanically connected (that is part of a distributed manipulator) operating to achieve a well-defined goal.

Servo-control Computer: SCC
Physically small, technically simple, and relatively low cost microprocessor-based device performing the servo-control function for at most three axes.

Single Axis Controller: SAC
SCC controlling only one axis.

Supervisory Computer, Supervisor: SC
Device which organises the activities of a number (possibly tens) of SCCs and/or SACs together with any associated tools, jigs, fixings, sensors etc.
REFERENCES

1 : Crossman, E.R.F.W.

2 : Head, B.

3 : Black, G.

4 : Preston, P. et al

5 : Sleigh, J. et al.

6 : Williams, S.

7 : Lobley, D.T.

8 : Foster, D.F.

9 : Miles, K.E.

10 : Wildish, M.

11 : Lawerence, A.

12 : Garrett, A.

13 : Kochan, A.

14 : Churchill, W.S.
15 : Chilver, Sir Henry et al

16 : Ashdown, P. et al

17 : Head, B.

18 : Fields, T.

19 : Riley, F.J.

20 : Riley, F.J.

21 : Walton, J. Page

22 : Wild, R.

23 : Rosted, Ib.

24 : McKillop, A.J.

25 : Potts, D.

26 : Richardson, B.A.
27 : Labanche, E.F.  

28 : Davison, R.G.  

29 : Pham, D.T.  

30 : Rooks, B.  

31 : Truman, R.  

32 : Cassidy, J.  

33 : Owen, A.E.  
"Automated assembly can equate with short payback periods", Proc. 3rd Int. Conf. on Assembly Automation, Stuttgart, West Germany, May 1982.

34 : Primrose, P.L. and Leonard, L  

35 : Kanum, L.J.  
"Recent applications of modular technology robots", Proc. 13th Int. Sym. on Industrial Robots, Chicago, April 1983, pages 11-66 to 11-72

36 : Lindstrom, B.  

37 : Bartl, P. et al.  

38 : Harrison, R. et al  
39: Denavit, J. and Hartenberg, R.S.

40: Paul, R.P.

41: Plessmann, K.W.

42: Anning, N.

43: Adept Technology Inc.
"Adeptone service manual", March 1985, Chapter 3 - Principles of operation.

44: Mitchell, P.

45: Larson, T.M. and Coppola, A.

46: Horden, I. and Ryan, D.
"Building a structured system with microcontrollers", New Electronics, 18 February, 1986.

47: Anon.

48: Allan, B.

49: Engelberger, J.F.

50: Anon.

51: Astrop, A.
52: Davey, P.G. and Wiejak, J.S.  

53: Anon.  

54: Shoham, M.  

55: Makyho, H. and Furuga, N.  
"Selective Compliance Assembly Robot Arm", Proc 1st Int. Conf. on Assembly Automation, Brighton, March 1980, 77-86.

56: Schraft, R.D. and Warner, R.C.  

57: Archer, J.R. and Blenkinsop, R.T.  

58: Walker, D.G.  

59: Barber, N.  
"Coming to terms with brushless servo drives", Electric Drives and Controls, April/May 1985.

60: McCelland, S.  
"Motivating developments in electric drives for robots", Industrial Robot, March 1987, 44.

61: McCloy, D. and Martin, H.R.  

62: The Staff of Martonair Ltd  

63: Foulk, P.W.  
64 : Bennetts, B.

65 : Anon.

66 : Allan, B.

67 : Allan, B.

68 : Cole, B.C.

69 : Shimada, S.

70 : Anon.

71 : Dahlberg, B.

72 : Tal, J.

73 : Anon.

74 : Kokkonen, K. and Pashley, R.

75 : Lund, P.

76 : Ward, M.A.
77 : MacWilliams, P.D. et al.

78 : Connell, R.

79 : O'Mohundro, K.

80 : Wilson, P. and Cordingdale, C.

81 : Mackenna, C. et al.

82 : Harrison, N.

83 : Edwards, R.

84 : McCarthy, J.P.

85 : Goodman, M.

86 : Yourdon, E. and Constantine, L.L.

87 : McGowan, C.L. and Kelly, J.R.

88 : Mascot Suppliers' Association

89 : Thatcher, T.W. et al.

90 : Bishop, T.
91 : Enticknap, N.  

92 : Anon.  

93 : Philpotts, P.  

94 : Wilson, I.R. and Addyman, A.M.  

95 : Leadbetter, P.A.  
"Towards a pragmatic philosophy of artificial intelligence", Interfaces in Computing, 2, 1984, 269-277

96 : Clocksin, W.F. and Mellish, C.S.  

97 : Barr, A. and Feignbaum, E.A.  
"The Handbook of Artificial Intelligence - Vol II", Pitman, 1982

98 : Mills, P. and Jones, R.  

99 : Teacher, D. and Morton, B.A.  

100 : Duda, R.O.  

101 : Matthews, A. and Swift, K.  

102 : McLellan, E.  

103 : Gruver, W.A. et al.  
104: Paul, P.

105: Shahid Mujtaba, M.


107: Lieberman, L.I. and Wesley, M.A.

108: Shimano, B.E.

109: Giuse, D. and Guida, G.

110: Wirtsch, D. et al.

111: Falek, D. and Parent, M.

112: Donato, G. and Camera, A.

113: Novatchenko, S.I. et al.

114: Nemes, L.
115: Shimano, B.E.

116: Braganza, C.A.J. and Sholl, P.

117: Latombe, J-C. and Mazer, E.

118: Cooke, I.E. and Heys, J.D.

119: Taylor, R.H. et al.

120: Takase, K. et al.

121: Paul, R.P. et al.
"Advanced Industrial Robot Control Systems", Proc. 10th National Science Foundation Conf. on Production Research and Technology, Detroit, February 1983, 167-175.

122: Steiger-Carcao, A. and Camarinha-Matos, L.M.

123: Volz, R.A. et al.

124: Paul, R.P. and Hayward, V.

125: Weck, M. and Niehaus, T.

126: Lucas, A.J.

248
127: Arai, T. and Matsumoto, A.
"Intermediate language for robot controllers", Proc 4th Int. Conf. on Assembly Automation, Tokyo, October 1983, 55-66.

128: Arai, T. et al

129: D'Souza, C. et al.
"Aspects to Achieve Standardized Programming Interfaces for Industrial Robots."

130: Kawabe, S. et al.
"Interactive graphic programming for industrial robots", Proc 15th Int. Sym. on Industrial Robots, Tokyo, October 1985, 699-705.

131: Sata, T. et al.
"Robot simulation as a task programming tool", Proc. 11th Int. Sym. on Industrial Robots, Tokyo, October 1981, 595-602.

132: Ambler, P.

133: Popplestone, R.J. and Ambler, P.

134: Shaket, E. and Freedy, A.

135: Colombetti, M. and Pagello, E.
"A logical framework for robot programming", 2nd Int. CISM-IFToMM Sym. on the Theory and Practice of Robot and Manipulators, Warsaw, Poland, 1976

136: Gascoigne, J.D. et al.

137: Sharpe, C.

138: MacGregor, M.
139: Miller, R.L.

140: Allen-Bradley Co.

141: Anon

142: Burrows, C.R.

143: Shearer, J.L.

144: Weston, R.H. et al

145: Moore, P.R. et al

146: The Engineering Staff of the TI Semiconductor Group
"TMS 9900 Family Data Book", Texas Instruments, 1981.

147: Anon.

148: Anon.

149: Anon.

150: Anon.

151: Moore, P.R. et al.
152 : Schwenzer, R.

153 : Nagarajan, R. and Weston, R.H.

154 : Thatcher, T.W.

155 : Coiffet, P. and Chirouze, M.

156 : Hordeski, H.F.

157 : Ashley, W.R.

158 : Drazan, P.

159 : Franklin, G. and Powell, J.

160 : Anon.
"Type SN74LS2000 (Direction Discriminator)", TTL Data Book, Texas Instruments, 1985, Pages 7-726 to 7-734.

161 : Seyer, M.D.

162 : EIA Engineering Dept.
163 : EIA Engineering Dept.

164 : Weston, R.H. et al.

165 : Weston, R.H. et al.

166 : Bux, W.

167 : Salvi, G.

168 : Cox-Smith, P.
"Feasibility study into the application of microprocessors to the commercial exploitation of Martonair's pneumatic robot system", Cranfield Product Engineering Centre (CPEC), August 1983.

169 : Anon.

170 : Anon.

171 : Anon.

172 : Anon.

173 : Anon.
SERVO-VALVE RESISTOR VALUE SELECTION

This appendix considers the values of resistors R1, R2, Rz and Rvar in the design of the Mark 3 version servo-amplifier shown originally in Figure 10.4.1 and reproduced here for convenience as Figure A1.1.

The design requirements for the range and zero adjust functions are:

i) For a DAC input range of 0-10v to be able to adjust the output current range to match the active range of the particular valve in use and thus to obtain maximum resolution.

ii) For a particular valve at a DAC input voltage of 5v, to be able to use the zero adjust to null the valve (no net flow to either side of the actuator). Thus it is required that for a DAC input voltage of 5v the zero adjust range is sufficient to move the cylinder in both directions, possibly under the influence of gravity in either direction.

From Figure A1.1

Output current \( I_o = \frac{V_o}{R_{load}} \)

where \( R_{load} = 4.7 \text{ ohms} \) and \( V_o \) is the voltage at point C

Also \( V_o = \left[ \frac{-10}{10} V_{dac} \right] \left[ \frac{-10}{Rr} \right] + \left[ \frac{-10}{Rz} V_z \right] \)

where \( R_r = R_1 + R_2 + R_{var} \)

Thus \( V_o = 10 \left[ \frac{V_{dac} - V_z}{Rr} \right] \)

therefore \( I_o = \frac{10 \left( \frac{V_{dac} - V_z}{Rr} \right)}{4.7 \left( \frac{V_{dac} - V_z}{Rz} \right)} \) \hfill (1)

Note that: \(-3.6v < V_z < 3.6v\)

\(0v < V_{dac} < 10v\)

and that \( R_r \) and \( R_z \) have units of kilohms.

253
Now define Zero Range start by:

\[ ZR_{\text{start}} = Io(\text{min}) = Io \{ V_{\text{dac}}=5\text{v} \} \{ V_{z}=3.6\text{v} \} \]

and Zero Range end by:

\[ ZR_{\text{end}} = Io(\text{max}) = Io \{ V_{\text{dac}}=5\text{v} \} \{ V_{z}=-3.6\text{v} \} \]

thus Zero adjust range, \( ZR \), is given by:

\[ ZR = Io(\text{max}) - Io(\text{min}) \]

and so from (1) obtain:

\[ ZR_{\text{start}} = 10 \left[ \frac{5 - 3.6}{4.7 R_{r} / R_{z}} \right] \quad - (2) \]

\[ ZR_{\text{end}} = 10 \left[ \frac{5 + 3.6}{4.7 R_{r} / R_{z}} \right] \quad - (3) \]

and

\[ ZR = \frac{72}{4.7 R_{z}} \quad - (4) \]

Define Output current range start by:

\[ R_{\text{start}} = Io'(\text{min}) = Io \{ V_{\text{dac}}=0\text{v} \} \{ V_{z}=0\text{v} \} \]

and Output current range end by:

\[ R_{\text{end}} = Io'(\text{max}) = Io \{ V_{\text{dac}}=10\text{v} \} \{ V_{z}=0\text{v} \} \]

thus Output current range, \( R \), is given by

\[ R = Io'(\text{max}) - Io'(\text{min}) \]

and so from (1) obtain:

\[ R_{\text{start}} = 0 \quad - (5) \]

and

\[ R = R_{\text{end}} = \frac{100}{4.7 R_{r}} \quad - (6) \]

Also Zero adjust range centre

\[ \frac{ZR_{\text{end}} - ZR_{\text{start}}}{2} + ZR_{\text{start}} \]

\[ = \frac{50}{4.7 R_{r}} \quad [\text{using (2), (3) and (6)}] \]

Similarly Output current range adjust centre

\[ \frac{1 R}{2} = \frac{50}{4.7 R_{r}} \]

255
Thus the centres of the zero adjust and the output current ranges always coincide for all values of $R_z$ and $R_r$.

It is now possible, using (2), (3), (5), (6) to calculate the zero adjust and output current ranges available with the Mark 3 version servo-amplifier with its original resistor values of:

- $R_z = 12$ kohms  
- $R_{var} = 0 - 20$ kohms
- $R_1 = 82$ ohms  
- $R_2 = 2.2$ kohms

The results of these calculations are shown in Figure A1.2 where it is clear that most of the output current range adjust is wasted and that the zero adjust range has an unnecessarily large scope, reducing its sensitivity.

More appropriate resistor values are now calculated. For the $0 - 0.7$A active range valve require that:

$$\frac{R_{end}}{R_r} = \frac{0.7}{4.7}$$

therefore

$$R_r = 30.4 \text{ kohms}$$

and for $0-1$A active range valves that:

$$\frac{R_{end}}{R_r} = \frac{1}{4.7}$$

therefore

$$R_r = 21.3 \text{ kohms}$$

Thus $R_r$ must be variable over at least the range

$$21.3 < R_r < 30.4$$

but

$$R_r = R_1 + R_2 + R_{var}$$

so select

$$R_{var} = 20 \text{ kohms}$$

$$R_1 = 2.2 \text{ kohms}$$

and

$$R_2 = 15 \text{ kohms}$$

obtaining

$$17.2 < R_r < 37.2$$

and so

$$0.57 \text{ A} < I_0'(\text{max}) \{V_{dac=10v}\} < 1.24 \text{ A}$$

$$V_z=0v$$

256
Required maximum value of Range (R)

Zero adjust range start (ZRstart)

Zero adjust range end (ZRend)

Range = R = Rend = 10max

Output current (Amps)

Range adjust resistor (Rvar) value (kohms)

Note: For values of –

Rz = 12 kohms
R1 = 82 ohms
R2 = 2.2 kohms
Rvar = 0 – 20 kohms

FIGURE A1.2 MK3 SERVO-AMPLIFIER OUTPUT AND ZERO ADJUST RANGES AS FUNCTIONS OF RANGE ADJUSTING RESISTOR VALUE (Rvar)
Thus the required output current ranges are available with sufficient capacity to drive valves of up to 0-1A active range.

To set up for the 0 - 0.7A valve, Rvar must be initially adjusted to 13.2 kohms, and for 0-1A valves to 4.1 kohms.

If 600mA is the necessary zero adjust range require that:

zero adjust range, $ZR = \frac{72}{4.7Rz} > 0.6$ A [using (4)]

therefore $Rz < 25.5$ kohms

Thus select $Rz = 22$ kohms

And so for the 0 - 0.7A valves with $Rz=22$ kohms and $Rvar=13.2$ kohms ($Rr=30.4$ kohms) obtain:

$ZR_{start} = 2$ mA [from (2)]  $ZR_{end} = 698$ mA [from (3)]

Similarly for 0-1A valves with $Rz=22$ kohms and $Rvar=4.1$ kohms ($Rr=21.3$ kohms) obtain:

$ZR_{start} = 151$ mA [from (2)]  $ZR_{end} = 848$ mA [from (3)]

These figures were generally satisfactory for both cases. If however lack of sensitivity in the 0 - 0.7A valve case should ever prove a handicap it may be necessary to switch the value of $Rz$ to 47 kohms. Using a Mark 3 version servo-amplifier with suitable values of $Vz$, $Vdac$, and $Rvar$ this theoretical approach was verified. The results are shown in Table A1.1. Thus the following resistor values were selected:

$R1 = 15$ kohms  $R2 = 2.2$ kohms

$Rz = 22$ kohms  $Rvar = 20$ kohms (variable)

with $Rvar$ set initially at 13.2 kohms and 4.1 kohms for the 0 - 0.7A and 0-1A valves respectively.
<table>
<thead>
<tr>
<th>Vz (volts)</th>
<th>Vdac (volts)</th>
<th>Rvar (kohms)</th>
<th>Icalc (mA)</th>
<th>Iexp (mA)</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>20.0</td>
<td>570</td>
<td>539</td>
<td>Minimum of the o/p current adj range</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>3.9</td>
<td>1008*</td>
<td>984</td>
<td>o/p saturation</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>0</td>
<td>1240*</td>
<td>984</td>
<td>Maximum of the o/p current adj range</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>4.1</td>
<td>1000*</td>
<td>981</td>
<td>Io'(max) required for 0-1A valves</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>13.2</td>
<td>700</td>
<td>686</td>
<td>Io(max) requ'd for 0 - 0.7 A valves</td>
</tr>
<tr>
<td>-3.6</td>
<td>5</td>
<td>13.2</td>
<td>698</td>
<td>704</td>
<td>ZRend )</td>
</tr>
<tr>
<td>3.6</td>
<td>5</td>
<td>13.2</td>
<td>2</td>
<td>3</td>
<td>ZRstart ) 0 - 0.7 A valves</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>13.2</td>
<td>348</td>
<td>346</td>
<td>ZRcentre )</td>
</tr>
<tr>
<td>-3.6</td>
<td>5</td>
<td>4.1</td>
<td>848</td>
<td>853</td>
<td>ZRend )</td>
</tr>
<tr>
<td>3.6</td>
<td>5</td>
<td>4.1</td>
<td>151</td>
<td>154</td>
<td>ZRstart ) 0-1A valves</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>4.1</td>
<td>499</td>
<td>495</td>
<td>ZRcentre )</td>
</tr>
</tbody>
</table>

* without o/p stage saturation being taken into account

Notes: Tests performed with resistive load of 24.3 ohms (nominal) and with R1=15 kohms, R2=2.2 kohms, Rvar = 0-20 kohms, and Rz=12 kohms. Vz, Vdac, Iexp as shown in Figure A1.1. Icalc is the value of Io calculated from the accompanying theory.

**TABLE A1.1 : RESISTOR VALUE SELECTION FOR MARK 3 SERVO-AMPLIFIER, THEORY COMPARED WITH PRACTICE**
APPENDIX 2

OPERATIONAL DETAILS OF THE ENCODER COUNTER CIRCUIT

Chapter XI described the need for the design of the secure optical incremental encoder interface and counter circuit, shown in Figure 11.5.3, which is reproduced for convenience here as Figure A2.1. This appendix details the functioning of this circuit.

Figure A2.2 and Table A2.1 detail the purpose of each functional block of the circuit whilst Tables A2.2 and A2.3 describe the inter-function block and microprocessor interfaces respectively. The following general notes apply:

i) **Signal levels** all interfaces at TTL levels

ii) **Counter Serial Load** may be achieved as described in function block 4 (Table A2.1).

iii) **Maximum Operating Speed** minimum edge separation of 208 nanoseconds (Figure A2.3), setting maximum operating speed above the maximum switching frequency of the encoder optoelectronics.

The complete initialisation of the counter circuit is achieved in three stages, initialise shift register, zero counter, clock buffer (counter read). Only two processor outputs are needed to perform these functions and the required sequences are given in Table A2.4.

These sequences may be used consecutively to achieve a complete initialisation, as required at power up, or individually.
FIGURE A2.1: SECURE ENCODER COUNTER CIRCUIT (Copied from Figure 11.5.3)
FIGURE A2.1: SECURE ENCODER COUNTER CIRCUIT (Copied from Figure 11.5.3)
FIGURE A2.2 ENCODER COUNTER CIRCUIT FUNCTION BLOCKS

16 bit count data to microprocessor (at TTL levels)

FB5

FB4

FB3

FB2

FB1

Free incremental encoder quadrature outputs (at TTL levels)

Microprocessor Control Signals (at TTL levels) (Table A2.3)
FB1: Encoder Interface and Edge Enable

These circuits take two quadrature pulse streams from the incremental encoder and provide a direction signal to the latch circuit (FB3).

A four times increase in resolution is achieved over normal incremental encoder counting techniques by counting each edge of both pulse streams. To facilitate this and to prevent multiple edge problems occurring only one edge is enabled at each instance. A shift register (IC3) provides this feature and is supplied with direction and clock signals from the outputs of the direction latch (FB3) and pulse delay (FB2) respectively to allow it to enable the correct edge at the correct time.

FB2: Pulse Delay

Takes rising edge signals from the encoder interface (FB1) and delays for approximately 104 nanoseconds before outputting a rising edge to the direction latch (FB3). After a further delay of 104 nanoseconds a second rising edge is output to the 16-bit counter (FB4) and edge enable (FB1) circuits. Outputs from this circuit may be disabled by MCS3 to allow external serial counter loads.

FB3: Direction Latch

Takes direction signal from encoder interface (FB1), latches and outputs it to 16-bit counter (FB4). Signal to latch is provided by pulse delay circuit (FB2) 104 nanoseconds after this circuit has received rising edge and 104 nanoseconds before pulse delay circuit (FB2) clocks 16-bit counter (FB4). This ensures that direction signals are established before counting takes place.

Outputs may be disabled by MCS3 to allow external serial loads.

FB4: 16-Bit Up/Down Counter

Takes pulse streams from the pulse delay circuit (FB2) and counts them in the direction given by the pulse latch circuit (FB3). The counter may be loaded with zero by suitable signals on MCS5. By disabling the normal pulse stream with MCS3 it may be loaded serially with any number by a direction signal on MSC1 and a pulse stream of suitable length on MCS2.

FB5: Buffer

This circuit takes the 16-bit value from the counter (FB4) and on a signal from MCS4 latches it so that it may be read by the microprocessor.

| TABLE A2.1 : FUNCTION BLOCK DESCRIPTIONS |
**I/F1** Provides a rising edge from the encoder enabling circuitry (FB1) to the pulse delay circuit (FB2).

**I/F2** Provides a direction indication from the encoder enable interface (FB1) to the direction latch (FB3).

**I/F3** This connection provides a rising edge to the direction latch (FB3).

**I/F4** Transmits the rising edge coming indirectly from the encoder via the encoder enable interface (FB1) and pulse delay circuit (FB2) to the counter (FB4).

**I/F5** Provides a direction signal to the 16-bit counter (FB4) (high = count down, low = count up) from the direction latch (FB3).

**I/F6** These sixteen lines transmit the binary value of the sixteen bit counter (FB4) to the buffer (FB5).

<table>
<thead>
<tr>
<th>TABLE A2.2 : FUNCTION BLOCK INTERFACES</th>
</tr>
</thead>
</table>
**MCS1** Selects the direction of an external serial load of the counter (from the microprocessor) as follows:
- Low = count up
- High = count down

**MCS2** Provides a square wave signal from the microprocessor which constitutes an external serial load of the counter. The counter will register one count on every low to high transition of the square wave stream.

**MCS3** Allows the microprocessor to disable the normal counting mode (counting pulses from the encoder) and also allows the enabling of an external serial load from the microprocessor. The following states apply:
- high = serial load enable/normal operation disabled
- low = serial load disabled/normal operation enabled

**MCS4** Clocks the 16-bit buffer (FB5) on a low to high transition. It thus allows the microprocessor to perform a 16 bit counter read.

**MCS5** Allows the microprocessor to reset the 16-bit counter to zero. Load with zero occurs when this line is taken low. It should therefore normally be maintained high by the microprocessor.

**MCS4 & 5** These two lines are used together to reset the encoder counter circuit in general, and the encoder interface (FB1) shift register (IC3) in particular.

**TABLE A2.3 : MICROPROCESSOR CONTROL SIGNALS**

265
From encoder interface circuits (FB1, IC14)

IC 8
monostable multivibrator (74LS121)

IC 10
monostable multivibrator (74LS121)

$\bar{q}$

Notes:
$t_1, t_2$ governed by $C_1R_1$ and $C_2R_2$ respectively.

From figure A3.1.1.

$C_1 = C_2 = 100\text{pF}$

$R_1 = R_2 = 1.5\text{kohms}$

and thus

$t_1 + t_2 = C_1 R_1 \ln (2)$

$= 0.693 C_1 R_1 = 104 \text{nanoseconds}$

to direction latch (FB3)

to counter (FB4)
and edge enabling circuits (FB1, IC3)
<table>
<thead>
<tr>
<th>Initialise shift register (FB1, IC3)</th>
<th>Zero Counter (FB4)</th>
<th>Clock Buffer i/ps to o/ps (FB5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS4 (D14)</td>
<td>MCS4 (D14)</td>
<td>MCS4 (D14)</td>
</tr>
<tr>
<td>MCS5 (D15)</td>
<td>MCS5 (D15)</td>
<td>MCS5 (D15)</td>
</tr>
<tr>
<td>?</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>?</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Load with 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Read</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE A2.4: MICROPROCESSOR CONTROL SIGNALS SEQUENCES**
THE TEXAS INSTRUMENTS TMS 9900 FAMILY OF MICROPROCESSORS

A3.1 INTRODUCTION

This appendix gives details of the design and operation of the TMS 9900 microprocessor family which are relevant to this project. For complete coverage the reader is referred to the TMS 9900 Family Data Book (146).

A3.2 ARCHITECTURE

The architecture of the TMS 9900 family is somewhat unusual in having on-board, besides the customary arithmetic logic unit (ALU) and address/data/IO/control bus circuits, just 3 other special purpose registers. These are the Program Counter (PC), containing the address of the next instruction to be executed, the status word (ST), and the Workspace Pointer (WP) which holds the address of the current workspace or general purpose register area that usually resides in ordinary off-chip memory. Thus the TMS 9900 family processors do not possess on-chip accumulators or general purpose registers. Their only equivalents are the sixteen off-chip registers pointed to by the workspace pointer. These registers are not as fast to access as on-chip ones and therefore simple register-to-register arithmetic is slightly slower than with processors having on-chip registers, such as the Intel 8086. However, benefits accrue when any context switch, most significantly interrupt response, is required. Thus for applications in which fast interrupt responses are necessary the TMS 9900 has a speed advantage. Also procedure calls and returns are executed more rapidly. Although this treatment is somewhat over-simplified the TMS 9900 is well suited to real-time applications requiring good interrupt handling together with moderate amounts of arithmetic and data manipulation.
A3.3 TMS 9900 MICROPROCESSOR

This is the original member of the family and many of its principles of design and operation are common to other group members.

A3.3.1 Memory Map, Addressing, and Word Length

With 15 address and 16 data lines the TMS 9900 addresses and handles up to a maximum of 32k words of 16-bit length. A 16th address line, used internally by the processor further divides this into 64k bytes and facilitates byte as well as word operands.

A3.3.2 Interrupts

The TMS 9900 processor handles up to 16 interrupts (level 0=highest priority, is reserved for CPU reset) with on-chip prioritisation and masking. A peripheral device wishing to interrupt the processor signals its level (0=highest priority, 15=lowest priority) on the level encoding lines (IC0-IC4) and activates interrupt request (INTREQ) (Figure A3.1).

On detecting an interrupt request the level is compared to the software set interrupt mask. If allowed (level <= mask) a context switch to the appropriate interrupt service routine takes place in which the Workspace Pointer (WP) and Program Counter (PC) are fetched and the old Workspace Pointer, Program Counter, and Status Register (ST) including the interrupt mask, are stored in the new workspace registers 13, 14, and 15. With only 2 pieces of data to fetch and 3 to store, interrupt response is fast compared to processors with non memory-to-memory architectures.

The interrupt mask is then set to the current level less one, thereby masking out interrupts with levels higher (priorities lower) than the current interrupt. The interrupt service routine must be terminated with a "Return with Workspace" (RTWP) instruction which restores the original Program Counter, Workspace Pointer, and Status (and hence interrupt mask) so that the processor returns to its original task.
<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin 1</th>
<th>Pin 2</th>
<th>Pin 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vcc</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAIT</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOAD</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOLD</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAQ</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PH1CLK</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PH2CLK</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A14</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A13</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A12</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A11</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A10</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A8</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PH4CLK</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vss</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vdd</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PH3CLK</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBIN</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUOUT</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUIN</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTREQ</td>
<td>32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Underlined signals are active low

FIGURE A3.1 | TMS 9900 & TMS 9995 PIN DESCRIPTIONS

270
A3.3.3 I/O Interfaces

Besides the usual memory mapped interfacing of peripheral devices, the TMS 9900 also possesses a simple I/O bus, the Communications Register Unit (CRU). This consists of 3 dedicated lines (CRUOUT, CRUIN, CRUCLK) together with part of the address bus (A3-A14) permitting the use of 4096 lines of both input and output. Special software instructions enable the use of these facilities either in single bits or in multiples of up to 16 bits.

A3.4 TMS 9995 MICROPROCESSOR

The TMS 9995 is architecturally identical to the TMS 9900 but with an instruction pre-fetch facility which in most situations considerably decreases the length of the instruction fetch/execute cycle and hence increases execution speed.

A3.4.1 On-chip RAM

The TMS 9995 incorporates 256 bytes of RAM of which 252 are available to the user. For applications requiring small quantities of RAM this reduces the hardware required (and hence also the cost). Even in situations needing more than 252 bytes of RAM, a considerably increased execution speed is achieved, provided it can be arranged that the most frequently used instructions and data are loaded into this on-chip area as the time needed to fetch data from on-chip is considerably less than from off-chip (by approximately a factor of 2). Exact calculation of the benefits of the use of on-chip RAM are somewhat lengthy but Table A3.1 gives some examples.

A3.4.2 On-Chip Timer/Event Counter

These features may be used to generate real-time clock interrupts at level 3 with a best resolution of sixteen times the crystal fundamental period (for example 1.33 microseconds for a 12Mhz crystal). The use of this facility requires no additional hardware.
A3.4.3 Interrupts
These are handled by the TMS 9995 in a fashion exactly similar to that of the TMS 9900 including prioritorisation. However, for service routines with workspaces in on-chip RAM, context switches occur faster than for the TMS 9900, and thus interrupt response is achieved more rapidly. Other architectural and fabric-ation advances which enhance the general execution speed of the TMS 9995 also further benefit its speed of interrupt response compared to that of the TMS 9900.

A3.4.4 I/O Interfaces
These are similar to those of the TMS 9900. However 32k bits of each of input and output are available of which 16 of both are internal, that is on-board the TMS 9995 chip itself. Of these 5 input and 5 output have predefined uses and the remainder may be used for internal software signalling.

A3.4.5 Pin Out
The TMS 9995 is contained in a 40 pin package. As shown by Figure A3.1, the reduction from the 64 pin package of the TMS 9900 is mainly achieved by the following means:

i) Reducing the number of data bus lines from 16 to 8
ii) Simplification of clock timing I/O signals
iii) Reduction in number of power supplies needed
iv) Reduction in the number of external available interrupts
v) Double use of pins (eg A15/CRUOUT)

It should be emphasised that although only 8 data lines are brought out from the processor, the TMS 9995 has a full internal 16-bit architecture. This reduction in external data bus might be thought to indicate a reduction in the speed of memory access compared to the TMS 9900, as 2 memory cycles are needed to achieve the read/write of a complete 16-bit word. However the somewhat earlier construction of the TMS 9900 also necessitated the use of 2 clock cycles for a full 16-bit memory access.
MOV R1,R2 1.00 2.00 4.00
MOV R1,@LABEL 1.33 2.67 5.33
MOV @LABEL1,@LABEL2 1.67 3.33 6.67

*1 Opcodes, symbolic (direct) addresses, workspace registers and symbolic (direct) operands - On-chip

*2 As *1 but all Off-chip and with 0 wait states off-chip

*3 As *2 but with 1 wait state off-chip

All timings are in microseconds and assume 12MHz clock

**TABLE A3.1 : EXECUTION SPEED BENEFITS OF ON-CHIP RAM**

<table>
<thead>
<tr>
<th></th>
<th>TMS 9995 (*1 below)</th>
<th>TMS 9995 (*2 below)</th>
<th>TMS 9900</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV R1,R2</td>
<td>1.33</td>
<td>2.00</td>
<td>4.67</td>
</tr>
<tr>
<td>MOV R1,@LABEL</td>
<td>2.33</td>
<td>2.67</td>
<td>7.33</td>
</tr>
<tr>
<td>MOV @LABEL1,@LABEL2</td>
<td>3.00</td>
<td>3.33</td>
<td>10.00</td>
</tr>
<tr>
<td>Interrupt Response</td>
<td>5.67</td>
<td>7.33</td>
<td>7.33</td>
</tr>
<tr>
<td>RTWP</td>
<td>2.33</td>
<td>3.33</td>
<td>4.67</td>
</tr>
</tbody>
</table>

*1 Workspace registers - On-chip, all other opcodes, addresses, operands, and interrupt vectors - Off-chip

*2 All workspace registers, opcodes, addresses, operands, and interrupt vectors - Off-chip

All timings are in microseconds and assume a 12MHz clock and 0 wait states in all cases

**TABLE A3.2 : EXAMPLE COMPARISON OF THE EXECUTION SPEEDS OF THE TMS 9900 AND TMS 9995**
A3.4.6 Execution Speed

The instruction pre-fetch and on-board RAM features together with improvements in silicon fabrication technology, have resulted in a considerable increase in the execution speed of the TMS 9995 compared to that of the TMS 9900. Because of the differing effects of each of these factors in various circumstances it is impossible to numerically evaluate improvements in the general case. However Table A3.2 provides an example.

A3.5 TMS 9902 ASYNCHRONOUS COMMUNICATIONS CONTROLLER

The TMS 9902 ACC provides an interface between TMS 9900 family processors and serial asynchronous communications channels, including RS-232-C and RS-422-A. The following features are pertinent to this project:

i) programmable character length (5-8 bits))
ii) programmable number of stop bits (1, 1.5, 2)
iii) programmable parity (odd, even, none)
iv) programmable transmit and receive data rates
v) readily interfaced to TMS 9900 family CPUs via the CRU
vi) programmable initiation of CPU interrupts including "byte received".
This appendix contains the TMS 9995 assembler listings of the SAC pre-production version hardware test program introduced in Chapter XV, Section 15.8.

Figure A4.1 contains the map of the object code produced by the linking tool "TXSLNK" (169), the first column of which is a catalogue of all the modules forming the complete program. The page numbers of the modules' listings, which conclude this appendix, are given in Table A4.1.

The program in the module "SELCTT" forms an infinite execution loop from which all tests requested by the operator are called. The modules "ADRMAP", "GENP", and "HEXIO" contain the I/O address map, general utility procedures (such as delays, character transmit and receive), and the hexadecimal I/O procedures respectively. The tests called by "SELCTT" are contained in the remainder of the modules.

"SIOTST" and "TIMTST" test the serial I/O byte received and timer interrupts respectively and contain, not only the procedures called from "SELCTT" which perform appropriate hardware initialisation and VDU handling, but also the interrupt service routines.
<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELCTT</td>
<td>278</td>
</tr>
<tr>
<td>ADRMAP</td>
<td>280</td>
</tr>
<tr>
<td>BBRTST</td>
<td>281</td>
</tr>
<tr>
<td>BRKTST</td>
<td>282</td>
</tr>
<tr>
<td>CNTTST</td>
<td>283</td>
</tr>
<tr>
<td>CPURUN</td>
<td>283</td>
</tr>
<tr>
<td>DACTST</td>
<td>283</td>
</tr>
<tr>
<td>GENP</td>
<td>285</td>
</tr>
<tr>
<td>INITST</td>
<td>286</td>
</tr>
<tr>
<td>HEXIO</td>
<td>287</td>
</tr>
<tr>
<td>MEMTST</td>
<td>288</td>
</tr>
<tr>
<td>OPMSEL</td>
<td>289</td>
</tr>
<tr>
<td>OVRTMP</td>
<td>290</td>
</tr>
<tr>
<td>PLCADR</td>
<td>290</td>
</tr>
<tr>
<td>PLCST</td>
<td>291</td>
</tr>
<tr>
<td>REFPRX</td>
<td>292</td>
</tr>
<tr>
<td>SACADR</td>
<td>292</td>
</tr>
<tr>
<td>SIOTST</td>
<td>293</td>
</tr>
<tr>
<td>TIMTST</td>
<td>293</td>
</tr>
</tbody>
</table>

Table A4.1: PAGE NUMBERS OF THE LISTINGS OF THE MODULES OF THE SAC PRE-PRODUCTION AND PRODUCTION VERSION HARDWARE TEST PROGRAM
ull procedure

GENERAL PROCEDURES

PAGE 9801

0081 101 "GEMP"
0083 101 "GEMP"
0086 101 "GEMP"
0085 101 "GEMP"
0087 101 "GEMP"
0089 101 "GEMP"
0090 101 "GEMP"
0091 101 "GEMP"
0092 101 "GEMP"
0093 101 "GEMP"
0094 101 "GEMP"
0095 101 "GEMP"
0096 101 "GEMP"
0097 101 "GEMP"
0098 101 "GEMP"
0099 101 "GEMP"
0100 101 "GEMP"
0101 101 "GEMP"
0102 101 "GEMP"
0103 101 "GEMP"
0104 101 "GEMP"
0105 101 "GEMP"
0106 101 "GEMP"
0107 101 "GEMP"
0108 101 "GEMP"
0109 101 "GEMP"
0110 101 "GEMP"
0111 101 "GEMP"
0112 101 "GEMP"
0113 101 "GEMP"
0114 101 "GEMP"
0115 101 "GEMP"
0116 101 "GEMP"
0117 101 "GEMP"
0118 101 "GEMP"
0119 101 "GEMP"
0120 101 "GEMP"
0121 101 "GEMP"
0122 101 "GEMP"
0123 101 "GEMP"
0124 101 "GEMP"
0125 101 "GEMP"
0126 101 "GEMP"
0127 101 "GEMP"
0128 101 "GEMP"
0129 101 "GEMP"
0130 101 "GEMP"
0131 101 "GEMP"
0132 101 "GEMP"
0133 101 "GEMP"
0134 101 "GEMP"
0135 101 "GEMP"
0136 101 "GEMP"
0137 101 "GEMP"
0138 101 "GEMP"
0139 101 "GEMP"
0140 101 "GEMP"
0141 101 "GEMP"
0142 101 "GEMP"
0143 101 "GEMP"
0144 101 "GEMP"
0145 101 "GEMP"
0146 101 "GEMP"
0147 101 "GEMP"
0148 101 "GEMP"
0149 101 "GEMP"
0150 101 "GEMP"
0151 101 "GEMP"
0152 101 "GEMP"
0153 101 "GEMP"
0154 101 "GEMP"
0155 101 "GEMP"
0156 101 "GEMP"
0157 101 "GEMP"
0158 101 "GEMP"
0159 101 "GEMP"
0160 101 "GEMP"
0161 101 "GEMP"
0162 101 "GEMP"
0163 101 "GEMP"
0164 101 "GEMP"
0165 101 "GEMP"
0166 101 "GEMP"
This appendix contains notes on the detailed coding techniques employed in the SAC pre-production and production version run-time software, introduced in Chapter XV, Section 15.9, together with the TMS 9995 assembler program listings.

In order to make maximum use of on-chip RAM and its associated access speed benefits, some modules have reduced workspaces. For instance, the module "START" possesses only registers 12-15. This is achieved by reserving only the required space:

```
DSEG
WS    BSS  8
DEND
```

but declaring the workspace to begin 24 bytes before it actually does:

```
STRTWS DATA WS-24
```

This of course required disciplined programming on the part of the author but did ensure RAM, especially that on the TMS 9995 chip, was not wasted.

In order to identify to the linker data that needed to be placed in the on-chip and off-chip RAM areas, assembler directives were used. All data in all modules between the assembler directives "CSEG" and "CEND" is placed together continuously in memory by the linking tool "TXSLNK" (169) to produce the "common" block in the "Link Map" (Figure A5.2). The linking tool also associated this common block with address hexadecimal 4000 corresponding to the start address of the off-chip RAM area.
Similarly all data between "DSEG" and "DEND" (The "$DATA" elements in the "Link Map" in Figure A5.2) are placed together continuously in the memory area corresponding to the on-chip RAM. Unchanging data (constants) is, like all executable code, not bracketed by assembler directives and is thus contained, with the executable code, in the EPROM area.

Inter-procedure communication is achieved using one of the techniques shown in Figure A5.1. The first technique illustrates the passing of one fixed value parameter to a procedure. The second shows the transmission of one variable value parameter from the calling to the called procedure using one of the callers' own workspace registers. The last example accomplishes a similar transmission but in the reverse direction, from the called to the calling procedure, but using an extra RAM data location. Techniques II and III may each be used to pass data in either direction. All three techniques may be used together or multiply to effect multiple parameter passing in either or both directions using private data areas that are only known and available to the two communicating procedures.

Figure A5.2 contains the map of the object code produced by the linking tool "TXSLNK" (169), the first column of which is a catalogue of all the modules forming the complete program. The page numbers of the modules' listings, which conclude this appendix, are given in Table A5.1.
### RAM Data Area

<table>
<thead>
<tr>
<th>Segment</th>
<th>Size</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSEG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS</td>
<td>32</td>
<td>BSS</td>
</tr>
<tr>
<td>PROCWS</td>
<td>32</td>
<td>BSS</td>
</tr>
<tr>
<td>PARAM</td>
<td>2</td>
<td>BSS</td>
</tr>
<tr>
<td>CEND</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Calling Software

<table>
<thead>
<tr>
<th>Technique I</th>
<th>Called Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLWP @PROC1</td>
<td>PC1 MOV *R14+,R1 R1=100</td>
</tr>
<tr>
<td>DATA 100</td>
<td>Process using value 100</td>
</tr>
<tr>
<td>PROC1 DATA PROCWS RTWP</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique II</th>
<th>Called Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>LI R8,10</td>
<td>PC2 MOV *R14+,R11 R1=10</td>
</tr>
<tr>
<td>BLWP @PROC2</td>
<td>MOV *R11,R1 Process using value 10</td>
</tr>
<tr>
<td>DATA R8*2+WS</td>
<td>..........</td>
</tr>
<tr>
<td>PROC2 DATA PROCWS RTWP</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique III</th>
<th>Called Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLWP @PROC3</td>
<td>PC3 MOV *R14+,R11 Calculate reply into R1</td>
</tr>
<tr>
<td>DATA PARAM</td>
<td>..........</td>
</tr>
<tr>
<td>PROC3 DATA PROCWS RTWP</td>
<td></td>
</tr>
</tbody>
</table>

### FIGURE A5.1

TMS 9995 ASSEMBLER PARAMETER PASSING TECHNIQUES
<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>300</td>
<td>PLC</td>
<td>318</td>
</tr>
<tr>
<td>BBCKSM</td>
<td>301</td>
<td>PPROC</td>
<td>319</td>
</tr>
<tr>
<td>CRUMAP</td>
<td>301</td>
<td>READMS</td>
<td>319</td>
</tr>
<tr>
<td>DELAY</td>
<td>301</td>
<td>RTC</td>
<td>320</td>
</tr>
<tr>
<td>EQPINS</td>
<td>301</td>
<td>RTCINS</td>
<td>322</td>
</tr>
<tr>
<td>ERROR</td>
<td>303</td>
<td>RXINSB</td>
<td>323</td>
</tr>
<tr>
<td>FDBINS</td>
<td>305</td>
<td>RXMS</td>
<td>323</td>
</tr>
<tr>
<td>GPROC</td>
<td>306</td>
<td>SCAN</td>
<td>324</td>
</tr>
<tr>
<td>ICINS</td>
<td>307</td>
<td>STATUS</td>
<td>324</td>
</tr>
<tr>
<td>INITS</td>
<td>307</td>
<td>SYSPRM</td>
<td>325</td>
</tr>
<tr>
<td>INSTR</td>
<td>309</td>
<td>TCHDSP</td>
<td>326</td>
</tr>
<tr>
<td>IPROC</td>
<td>311</td>
<td>TCHMEN</td>
<td>327</td>
</tr>
<tr>
<td>LTABLE</td>
<td>312</td>
<td>TCHSET</td>
<td>328</td>
</tr>
<tr>
<td>MENU</td>
<td>312</td>
<td>TEACH1</td>
<td>330</td>
</tr>
<tr>
<td>MSFDBI</td>
<td>313</td>
<td>TEACH2</td>
<td>331</td>
</tr>
<tr>
<td>MSPARA</td>
<td>315</td>
<td>TX</td>
<td>332</td>
</tr>
<tr>
<td>NMIINS</td>
<td>317</td>
<td>VDUIO</td>
<td>333</td>
</tr>
<tr>
<td>OPMODE</td>
<td>317</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A5.1: Page numbers of the listings of the modules of the SAC Pre-production and Production Version Run-time Software
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0082</td>
<td>0014</td>
</tr>
<tr>
<td>0082</td>
<td>0014</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
<tr>
<td>0099</td>
<td>0006</td>
</tr>
</tbody>
</table>
0166 IBM 8300 UNIVPC LINE B

0167 IBM 8300

0168 IBM 8300

0169 IBM 8300

0170 IBM 8300

0171 IBM 8300

0172 IBM 8300

0173 IBM 8300

0174 IBM 8300

0175 IBM 8300

0176 IBM 8300

0177 IBM 8300

0178 IBM 8300

0179 IBM 8300

0180 IBM 8300

0181 IBM 8300

0182 IBM 8300

0183 IBM 8300

0184 IBM 8300

0185 IBM 8300

0186 IBM 8300

0187 IBM 8300

0188 IBM 8300

0189 IBM 8300

0190 IBM 8300

0191 IBM 8300

0192 IBM 8300

0193 IBM 8300

0194 IBM 8300

0195 IBM 8300

0196 IBM 8300

0197 IBM 8300

0198 IBM 8300

0199 IBM 8300

0200 IBM 8300

0201 IBM 8300

0202 IBM 8300

0203 IBM 8300

0204 IBM 8300

0205 IBM 8300

0206 IBM 8300

0207 IBM 8300

0208 IBM 8300

0209 IBM 8300

0210 IBM 8300

0211 IBM 8300
Firstly it contains the procedures required to activate and deactivate the brake. Both procedures take no parameters and send no replies.

When called the procedures check the system brake configuration. Four modes are possible encoded in the word 'BRKSYS' as below:

- BRKSYS = 0 Brake node with active HI brake
- BRKSYS = 1 Brake node with active LO brake
- BRKSYS = 2 Active node with active HI brake
- BRKSYS = 3 Active node with active LO brake

- Node brake mode implies that the brake is not to be applied at any point, in particular, not when the axis reaches position.

The system is said to have an active brake when the brake is actuated by the application of a logic 0 to the brake port. The "BRKSYS" procedure is required by the lower level C-1-SI service routine in "RIG/RC" and to avoid reversion problem it is available to this module with a different namespace but used code by using the call to "BRKSYS".

Secondly it contains a procedure (RCOUNT) to read the current value of the incremental encoder counter, that is the axis position. This procedure requires one parameter, the address of the location in which the counter value is to be placed, and this must be placed immediately after the procedure call.

Errors Reported : None

END OF DESCRIPTION

DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP

GENERAL CONTROL PROCEDURES MODULE

DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP

DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP

DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP

DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP

DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP

DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP

DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP

DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
DEF BRAHCP, BRAHCP, BRAHCP, BRAHCP
INTERNAL CONTROL INSTRUCTIONS MODULE

BEGIN

- Clear all 8 bit

- Clear all errors

---

END
PLC TIA00A V36227 RA

PLC RUN MODE PROCEDURE MODULE
Errors reported: None

Note: the instruction bit in the status word is not set during teach mode.

END OF DESCRIPTION

DEF...
### SEND DESCRIPTION

- **Description**: This module is part of the interrupt service routine. It handles the reception of data packets from the transmit buffer.
- **Function**: Receives data from the transmit buffer and passes it to the appropriate module for further processing.
- **Input**:
  - A complete packet (up to 256 bytes) or a partial packet.
- **Output**:
  - Transfers data to the appropriate module.

### Key Functions

- **Receive Buffer**: Reads data from the transmit buffer and checks for the end of the packet.
- **Packet Check**: Validates the packet and handles any errors.
- **Packet Format**: Determines the format of the received packet.

### Error Handling

- **Error Conditions**:
  - **Receive Error**: Indicates an error in receiving the packet.
  - **Packet Error**: Indicates an error in the packet format.
- **Error Messages**:
  - **Receive Error**: Indicates the type of error and the packet count.
  - **Packet Error**: Provides additional details about the error.

### Module Integration

- The module integrates with other components to ensure smooth data transmission and reception.
- Proper handling of errors is crucial to maintain data integrity.

### Further Reading

- Detailed documentation on error handling and packet formats can be found in the project's technical documentation.
**DESCRIPTION:** This module is called by "SAMS/VSC" immediately after power-up initialization and runs thereafter as a background task monitoring the condition of the overtemperature sensor, reference proximity sensor, initialization signal, and taking action accordingly.

**Features:**
- It monitors the operation node jumpers and switches by calling an appropriate procedure in "SAMS/SRC" and sets off the required operation node initiation procedure.
- This module is also responsible for flushing the CPU running indicator.

**Note:**
- The inputs from the overtemperature sensor, reference proximity sensor, and initialization signal are normally high (1) and the presence of a low (0) indicates the signal active.
- Errors Reported: 13 Servo-valve current high "over-temperature."

**END OF DESCRIPTION**

---

**SCAN**  IBM030  9134227 PA  PAGE 0001

---

**SCAN**  IBM030  9134227 PA  PAGE 0002

---

**STATUS**  IBM030  9134227 PA  PAGE 0003

---

**REASONS:**
- Bit 0: Axis in position
- Bit 1: Brake activated
- Bit 2: Axis at home
- Bit 3: Teach error
- Bit 4 to 9: not used
- Bit 10: Initializing axis
- Bit 11: Emergency stop
- Bit 12: Error word 1 not clear
- Bit 13: Error word 2 not clear
- Bit 14: Error word 3 not clear
- Bit 15: Error word 4 not clear

**Note:**
- Bit set/reset: status true
- Bit clear/reset: status false

**Errors Reported:** None

---

**ONLY AVAILABLE**

---

**END**