This item was submitted to Loughborough's Research Repository by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

## The role of a modular multi-microcomputer controller in position control systems

PLEASE CITE THE PUBLISHED VERSION

PUBLISHER
© Luiz Eduardo Lopes

LICENCE
CC BY-NC-ND 4.0

## REPOSITORY RECORD

Lopes, Luiz E.. 2019. "The Role of a Modular Multi-microcomputer Controller in Position Control Systems". figshare. https://hdl.handle.net/2134/13769.

## This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.Iboro.ac.uk/) under the following Creative Commons Licence conditions.

## (c) creative

C O M M O N S D E E D

Attribution-NonCommercial-NoDerivs 2.5

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

BY Attribution. You must attribute the work in the manner specified by the author or licensor


Noncommercial. You may not use this work for commercial purposes.

No Derivative Works. You may not alter, transform, or build upon this work

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Leqal Code (the full license).
Disclaimer $\left.{ }^{[ }\right]$

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

-000.4968.02


${ }^{\circ} 000 \times 4968 \cdot 02$

-

THE ROLE OF A MODULAR
MULTI-MICROCOMPUTER CONTROLLER
IN POSITION CONTROL SYSTEMS
by

LUIZ EDUARDO LOPES

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

Loughborough University
Department of Engineering Production August, 1982
(C) by Luiz Eduardo Lopes


To my wife DIVA and my daughter RAQUEL for their love and understanding

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.

## ACKNOWLEDGMENTS

The author wishes to thank

Professor R. Bell, for his kind supervision and support.

Mr G.P. Charles, Mr N.D. Carpenter and Mr D. Walters, for helping with the use of the laboratory facilities.

Mr P.G. Leaney, for making available the hydraulic rig and for interesting discussions about hydraulic systems.

Moog and Sperry-Vickers, for donating part of the equipment for the hydraulic rig.

Mr J. Manning, for repairing the slideway-table when it was necessary.

Mr A.A. de Queiroz, Mr S.C. Silva and Mr N. Bilalis, my friends, for sharing the problems of a research work.

Mrs S. Johnson, for carefully typing this work.

The INSTITUTO DE PESQUISAS TECNOLÓGICAS DO ESTADO DE SÃO PAULO S.A. (Brasil), through its Divisão de Engenharia Mecânica, for providing financial support.

## SUMMARY

The work reported in this thesis is concerned with the application of digital control techniques to position control. The major emphasis is devoted to the use of modular multi-microcomputers as real-time controllers. Minicomputers and microcomputers do not always meet the technical requirements of modern system design; the modular multi-microcomputer controller is the most appropriate choice in many high performance systems.

The specification, design and construction of a modular multimicrocomputer controller is described. The controller has been applied to an electro-hydraulic cylinder position control system, where emphasis has been given to the control of load damping. The control system design is based on the use of state-space methods. Optimal control and observers are used.

Experimental tests are reported for open and closed loop. The open-loop results show the particular characteristics of an asymmetric cylinder drive. The closed-loop results show the behaviour of the drive under a simple digital proportional controller and a state-variable controller. Variables such as position, velocity, cylinder pressures and piston thrust are displayed for several test conditions.

## CONTENTS

page
DECLARATION ..... i
ACKNOWLEDGMENTS ..... ii
SUMMARY ..... iii
CONTENTS ..... iv
ABBREVIATIONS ..... $\mathbf{x}$

1. INTRODUCTION1:
2. LITERATURE SURVEY ..... 3
2.1 Introduction ..... 3
2.2 Digital Control ..... 3
2.2.1 General Aspects ..... 3
2.2.2 Process Digital Contro1 ..... 4
2.2.3 Servo Digital Control ..... 5
2.3 Multiprocessor Systems in Control Applications ..... 9
3. ASPECTS OF STATE-VARIABLE DIGITAL CONTROL ..... 12
3.1 Introduction ..... 12
3.2 Continuous and Discrete State Equations ..... 12
3.2.1 Continuous State Equations ..... 12
3.2.2 Discrete State Equations ..... 14
3.3 Controller Design Sequence ..... 17
3.4 Design of the Control Law ..... 18
3.4.1 Pole-Placement Design Method ..... 19
3.4.2 Optimal Control Design Method ..... 21
3.5 Design of the Observer ..... 22
3.5.1 Ful1-Order Observer ..... 23
3.5.2 Reduced-Order Observer ..... 25
4 DIGITAL CONTROLLER COMPUTER AIDED DESIGN AND CLOSED LOOP SIMULATION ..... 28
4.1 Introduction ..... 28
4.2 Continuous to Discrete State Equations - STATE Program ..... 28
4.3 Feedback Gains by Pole-Placement Design - ACKER Program ..... 32
4.4 Feedback Gains by Optimal Control Design - SWEEP Program ..... 34
4.5 Closed-Loop Simulation - TIMELOCUS Program ..... 37
4. THE MODULAR MULTI-MICROCOMPUTER ANALYSIS ..... 44
5.1 Introduction ..... 44
5.2 Multi-Microcomputer Systems ..... 44
5.3 Multi-Microcomputer Structures ..... 48
5.3.1 General Structures ..... 48
5.3.2 Modularity ..... 50
5.4 Multi-Microcomputer Structure Selection ..... 51
5.4.1 Structures for Real-Time Control ..... 51
5.4.2 Central Memory Structure ..... 51
5.4.3 Global (Common) Bus Structure ..... 52
5.4.4 Modified Common Bus Structure ..... 54
5.4.5 Common-Bus Arbitration ..... 565.4.6 Common-Bus Selection
5. DESIGN OF A MODULAR MULTI-MICROCOMPUTER ..... 60
6.1 Introduction ..... 60
6.2 The Microcomputer ..... 60
6.2.1 The Microprocessor ..... 60
6.2.2 Microcomputer Selection ..... 61
6.2.3 The Texas TM990/100M-1 Microcomputer ..... 63
6.2.3.1 The Microcomputer Board ..... 63
6.2.3.2 TIBUG Software Interface ..... 67
6.2.3.3 The Texas TMS9901 Programmable Interface ..... 69
6.2.4 Memory Expansion - Necessity ..... 73
6.3 The General Mapped Communication Structure ..... 75
6.3.1 The Microcomputer's Communications Global Memory Area ..... 75
6.3.2 Externally Referenced Conmunication Areas ..... 79
6.3.3 The Number of Supported Modules ..... 83
6.3.4 The Role of the System Controller ..... 86
6.4 General Logical Design ..... 87
6.4.1 The Interrupt Controlled Arbitration Link ..... 88
6.4.2 Slave Definition Phase ..... 93
6.4.3 Slave Acceptance Phase ..... 95
6.4.4 Communication Phase ..... 98
6.4.5 End of Communication Phase ..... 101
6.5 The Design of a Modular System ..... 102
6.5.1 Integrated Circuits ..... 102
6.5.2 Dynamic Memory Restriction ..... 102
6.5.3 Buffering Signals on the System Bus ..... 103
6.5.4 Logic Diagrams ..... 105
6.6 Functional Discussion of the Designed Logic ..... 105
6.6.1 The Master-Slave Identification Address Decoder ..... 105
6.6.2 The Slave-Identification Address Decoder ..... 110
6.6.3 Slave Definition Handshake ..... 111
6.6.4 Master-Slave Communication ..... 114
6.7 The Timing Relationship Diagrams ..... 118
6. THE MULTI-MICROCOMPUTER'S HARDWARE IMPLEMENTATION and Software communication design ..... 126
7.1 Introduction ..... 126
7.2 The Hardware Design Implementation ..... 126
7.2.1 Board Manufacturing Process ..... 126
7.2.2 Card Size and Edge Connectors ..... 128
7.2.3 The Functional Distribution ..... 129
7.2.4 The Lay-Out Design and the Final Boards ..... 129
7.2.5 Line Termination for the System Bus ..... 136
7.2.6 The Racking System and Board Arrangement ..... 136
7.3 Memory Expansion Board - General Test ..... 137
7.4 The System Bus and Microcomputer ..... 143
7.4.1 Interrupt Controlled Arbitration Link and Microcomputer Connections ..... 143
7.4.2 Slave Definition Process and the Microcomputer ..... 144
7.4.3 The Handshake Connections ..... 144
7.4.4 CRU and Interrupt Lines - Final Definition ..... 145
7.4.5 Reset Action through the System Bus ..... 145
7.4.6 The Internal Bus Separation ..... 146
7.4.7 Final Interconnections and the System Bus ..... 147
7.5 Power Supply ..... 151
7.6 System Bus Testing - Partial Communication Software ..... 151
7.7 The General Communication Software ..... 155
7.7.1 The Controlled Process and Message Exchange ..... 155
7.7.2 Message Identification - The Transfer Vector ..... 157
7.7.3 Message Transfer Modes - Transparency ..... 158
7.7.4 Single and Global Message Definition ..... 159
7.7.5 Word and Block Transfer Routines ..... 162
7.7.6 Complementary Routines ..... 163
7.7.7 Considerations ..... 167
7. THE CONTROLLER TO ELECTROHYDRAULIC DRIVE INTERFACE ..... 169
8.1 Introduction ..... 169
8.2 The Hydraulic Rig ..... 169
8.2.1 Load Tables ..... 169
8.2.2 Hydraulic Cylinder ..... 173
8.2.3 The Servovalve ..... 173
8.2.4 Power Supply ..... 175
8.3 The Servo Amplifier ..... 176
8.4 Analog Input/Output Board ..... 178
8.4.1 The Analog I/O Necessity ..... 178
8.4.2 The Board's Choice ..... 179
8.4.3 System Configuration ..... 183
8.4.4 Analog Output ..... 185
8.4.5 Analog Inputs ..... 186
8.5 Position Transducer ..... 187
8.5.1 The Optical Linear Incremental Position Transducer ..... 187
8.5.2 Controller to Position Transducer Interface ..... 189
8.5.3 Detailed Interface Design ..... 190
8.5.4 Timing Diagrams and Interface Discussion ..... 192
8.5.5 Final Interface Board and Interconnections ..... 197
8. THE STRUCTURE OF THE CONTROLLER SOFTWARE AND ITS IMPLEMENTATION ..... 204
9.1 Introduction ..... 204
9.2 The General Software Structure ..... 204
9.3 Loop Closing Functions ..... 205
9.4 Loop Closing Routine ..... 205
9.4.1 Sampling Frequency Control ..... 205
9.4.2 Data Acquisition ..... 209
9.4.3 Position Control ..... 209
9.4.3.1 Feedback Counter ..... 209
9.4.3.2 Actual Position ..... 210
9.4.3.3 Position Following Error ..... 210
9.4.3.4 Feedback Increment ..... 211
9.4.3.5 Observer and Control Function ..... 213
9.4.4 Emergency Action ..... 214
9.5 Position Reference Generation ..... 216
9.5.1 General Considerations ..... 216
9.5.2 Ramp Generation Routine ..... 217
9.6 Data Acquisition and Operator's Communication ..... 220
9.6.1 Operator's Options ..... 220
9.6.2 Sampling Period ..... 223
9.6.3 Ramp Input Options ..... 224
9.6.4 Storing Variables ..... 226
9.6.5 Displaying Stored Values ..... 228
9.6.6 Restarting the System ..... 228
9.7 Task Division Between Modules ..... 229
9.7.1 General Division ..... 229
9.7.2 Master-Slave Communication ..... 229
9.7.3 Auxiliary Functions ..... 233
9.7.4 Initialisation Routines ..... 236
9.7.5 Linking Structure ..... 236
9. STATE-VARIABLE DIGITAL CONTROL OF THE ELECTROHYDRAULIC CYLINDER DRIVE ..... 240
10.1 Introduction ..... 240
10.2 Model Identification ..... 240
10.2.1 Load Natural Frequency ..... 241
10.2.2 Steady-State Characteristics ..... 244
10.2.3 Analytical Model ..... 248
10.2.4 Open-Loop Dynamic Test Results ..... 249
10.2.5 Open-Loop Steady-State Test Results ..... 254
10.2.6 The Identified Model ..... 259
10.3 Controller Design ..... 265
10.3.1 Model Equations in State-Space Representation ..... 265
10.3.2 Sampling Period and Delay ..... 268
10.3.3 Discrete State-Equations ..... 270
10.3.4 Control Law Definition ..... 274
10.3.5 Optimal Controller ..... 275
10.3.6 Velocity Estimation ..... 280
10.3.7 Reduced-Order Observer Design ..... 286
10.3.7.1 Matrix Partition and Observability ..... 286
10.3.7.2 Observer Design by Pole-Placement Method ..... 288
10.3.8 Combined Control Law and Observer ..... 291
10.3.9 Closed Loop Simulation ..... 294
10.4 Controller Equations - Software Implementation ..... 297
10. EXPERIMENTAL TESTS AND RESULTS ..... 299
11.1 Introduction ..... 299
11.2 Open Loop Gain ..... 299
11.2.1 DAC and Servo Amplifier Combined Gain ..... 299
11.2.2 Servovalve and Hydraulic Cylinder Combined Gain ..... 300
11.2.3 Gain Correction by Software ..... 304
11.2.4 Total Open Loop Gain ..... 304
11. 3 Coefficients for the Controller Equations ..... 306
11.4 Experimental Tests and Results ..... 306
11.4.1 Experimental Tests ..... 306
11.4.2 Position Ramp Input ..... 306
11.4.2.1 Digital Proportional Controller (P) ..... 310
11.4.2.2 State-Variable Controller (SV) ..... 310
11.4.2.3 Position Response ..... 323
11.4.3 Position Step Input ..... 323
11.5 Discussions ..... 331
12. CONCLUSIONS AND RECOMMENDATIONS ..... 336
REFERENCES ..... 339
APPENDICES
4.1 PSIMAl Program ..... 347
4.2 STATE Program ..... 350
4.3 ACKER Program ..... 353
4.4 SWEEP Program ..... 356
4.5 Matrices Arithmetic ..... 358
4.6 OBSEQU Program ..... 362
4.7 TIMELOCUS Program ..... 364
6.1 The 990/100M-1 Logic Diagrams and Pin Assignment ..... 370
6.2.1 TMS 9900 Memory Bus Timing ..... 378
6.2.2 TMS 9900 HOLD Timing ..... 379
6.2.3 Memory Expansion ..... 380
6.2.4 List and Description of Components for
Memory Expansion Board ..... 383
6.5.1 List and Description of Components for Address Bus Board ..... 384
6.5.2 List and Description of Components for Data Bus Board ..... 385
7.2 Open Collector Bus Drivers and Line Termination ..... 386
7.3 Memory Test Program ..... 388
7.5 Power Supply Specification ..... 390
7.6.1 System Bus First Test Program ..... 391
7.6.2 System Bus Second Test Program ..... 393
7.7.1 Word and Block Transfer Routines ..... 395
7.7.3 Acknowledge Master Routine ..... 401
7.7.4 Interrupt Slave Routine ..... 404
7.7.5 Interrupt Controller Routine ..... 406
8.2.1 Series 76 (E076-102) Moog Servovalve ..... 407
8.4.1 Specifications for I/O Analog Board ..... 410
8.4.2 Analog I/O Board and Microcomputer Interconnections ..... 411
8.4.3 Analog I/O Board and Ribbon Cable Interconnections ..... 412
8.5.1 Linear Transducer Signal Diagram ..... 413
8.5.2 Digitizer Feedback Board Components List ..... 414
8.5.3 Digitizer Feedback Board Pin-Function Assignment ..... 415
8.5.4 Digitizer Feedback Board and Microcomputer Interconnections ..... 417
8.5.5 EXE Unit and Digitizer Feedback Board Interconnections ..... 418
9.1 Loop Closing Routine ..... 418
9.2 Emergency Deceleration Routine ..... 425
9.3 Ramp Generation Routine ..... 426
9.4 Sampling Frequency ..... 434
9.5 Displaying Values Routine ..... 436
9.6 Module 2 Communication Software ..... 438
9.7 Module 1 Communication Software ..... 441
9.8 Arithmetic Routines ..... 443
9.9 Auxiliary Routines ..... 453
9.10 Module I Initialisation Routine ..... 456
9.11 Module 2 Initialisation Routine ..... 460
9.12 Process Software ..... 466
9.13 Module 1 Object Code Program ..... 467
9.14 Module 2 Object Code Program ..... 469
10.1 Control Function to Velocity Transfer Function Analysis ..... 471
10.2 Discrete State Equations for Extending Direction ..... 480
10.3 Discrete State Equations for Retracting Direction ..... 481
10.4 Sample of the SWEEP Program ..... 482
10.5 Observer Pole-Placement Design ..... 483
10.6 Matrix Coefficients for Observer Equations ..... 484
10.7 Estimated Variables ..... 485
10.8 Control Equation ..... 487
10.9 Observer Equations ..... 491

## ABBREVIATIONS

AC Alternating Current
A/D Analogue to Digital Converter
Amp Amplifier
ASCII American Standard Code for Information Interchange
CPU Central Processing Units
CRU Communication Register Unit
DAC (D/A) Digital to Analog Converter
DC Direct Current
DMA Direct Memory Access
EPROM Erasable Programable Read-Only Memory
GM Global Message
IC Integrated Circuit
ID Identification
I/0 Input-Output
HP Hewlett Packard
LSI Large Scale Integration
MCGMA Module's Communication Global Memory Area
MOS Metal Oxide Semiconductor
OA Operational Amplifier
P Proportional
PC Program Counter
PI Proportional and Integral
PID Proportional, Integral and Differential
RAM Random Access Memory
ROM Read Only Memory
SM Single Message
ST Status
SV State Variable
TTL Transistor-Transistor Logic
VDU Video Display Unit
WP Workspace Pointer
XOP Extended Operation
1K One Thousand
N.B.: $\quad A^{T} \quad$ Transpose of Matrix $A$

The work related in this thesis started as part of a new major approach to servo-control design. A group with particular interest in servodrives started working, in this Department, in the spring of 1980, towards computer aided design for D.C. and electro-hydraulic servo-drives. The use of microprocessors and digital control techniques was a major emphasis in the work.

The use of more complex methods of digital control, such as statevariable and adaptive control, imposed too much pressure on the capability of a single microcomputer. The use of modular multimicrocomputers was thought to be a reasonably inexpensive approach to control flexibility and computing power and, actually, a necessary control tool.

This research was to have been the first phase of a more comprehensive programme of work but the necessary SERC funds were not made available. The experimental work was aided by the help of manufacturers such as MOOG and VICKERS who donated equipment. One minor drawback in the work is that the output of the power supply, available in the laboratory, did not match the requirements of the experimental drive.

There was a choice between the use of a four-way electro-hydraulic servovalve controling either a symmetric or an asymmetric cylinder. The asymmetric cylinder was chosen as it raised the wider range of control problems.

Complete priority was given to the use of a position transducer as the only source of information available to the controller. All the nonavailable state-variables would have to be estimated using only position as the system output.

The thesis starts with a literature survey of works on the two main related subjects: digital control and multi-processors. On the digital control side, works of basic value were sought and emphasis was given to servo-digital control. In the area of multi-processors, the search looked for industrial applications with particular interest in servo-control.

The basic aspects of state-variable digital control techniques are treated in Chapter 3. There, two methods for controller design are presented and reduced-order observers are introduced as estimators for the non-available state-variables. Chapter 4 presents a set of programs, written in FORTRAN, which implement the design methods described in Chapter 3. With provision for closed-loop simulation, both controller and observer designs can be verified.

Multi-processors are discussed in Chapter 5 and a particular multimicrocomputer structure is chosen to be implemented. The design of a modular multi-microcomputer, based on the chosen structure, is presented in Chapter 6. The design includes many aspects, from the choice of a particular microcomputer board to the final modular multimicrocomputer hardware configuration. The hardware design implementation is discussed in Chapter 7. The general software communication is developed.

The interface of the modular multi-microcomputer controller to the electro-hydraulic cylinder drive is described in Chapter 8. The design for the position transducer interface is given.

The general closed-loop position control software is developed in Chapter 9, where a modular approach is taken, and a task division is made for the multi-microcomputer implementation. All software was written in assembly-level language.

The state-variable digital control of the electro-hydraulic cylinder drive is discussed in Chapter 10. A model is identified and an optimal controller with reduced-order observer is designed. The controller and observer equations are implemented.

Comparison has been made between the behaviour of the electrohydraulic cylinder drive under a simple digital proportional controller and the state-variable controller. The experimental results, and discussions, are presented in Chapter 11.

Conclusions and recommendations for future works are given in Chapter 12.

## CHAPTER 2

## LITERATURE SURVEY

### 2.1 INTRODUCTION

There are two basic subjects related to this work: digital control and multiprocessors.

With the advent of the microprocessor, digital control has expanded to areas which were just unthinkable before. This expansion made the number of published works increase tenfold. There is not any technical periodical nowadays which does not display at least one article about microprocessor application. When someone talks about microprocessor application he is often talking about digital control and sometimes about servo control. With this in mind we decided to include in this survey, publications related to process control and servo control. Process control is included to show that this area has been the test bed for many new control applications which are then extended to other areas.

Section 2.2.1 includes basic references about sampled data control and digital control. In section 2.2 .2 some representative works about process digital control are presented.

In Section 2.2.3, details are given rabout recent and new developments in the servo control area.

The use of multiprocessors has become quite normal in control applications, especially in machine tools. Section 2.3 looks for such applications as they are the ones which give emphasis to servo control.

DIGITAL CONTROL

### 2.2.1 General Aspects

The sampling action associated to any kind of digital control makes the study of sampled data control systems at least desirable. A complete set of references in this area is given by Lopes (1).

About digital control there are three recently published books which give, under certain restrictions, a reasonable source of recent papers published in this area. The first book to be
mentioned is the one by Franklin and Powell (2), first published in 1980. This is a concise book with emphasis on state-space methods which gives a good insight into many aspects of digital control. By the way, Franklin was the co-author, with Ragazzini, of the well known book: "Sampled Data Control Systems". The second book, by Katz (3), follows the same lines as the first with emphasis on microprocessor implementation. The third book to be mentioned is by Isermann (4) and was first published, in English, in 1981. This book is more comprehensive than the two mentioned before with great emphasis on industrial process control. It covers practically all aspects of digital control but it is difficult to read at times.

### 2.2.2 Process Digital Control

The use of digital control for industrial processes is well established and it is there that new developments in control theory or just application of already available theoretical results, are first tried. Fisher and Seborg (5) have compiled a number of papers where they show many aspects in the modelling, simulation and control design used in the chemical industry. Takahashi et al ( $6,7,8$ ) present interesting algorithms to be used in process control by implementation in stand-alone microprocessor based controllers.

Dahlin (9) presented a method for designing digital controllers, in single and cascade loop applications, based on the selection of a single tuning parameter which defined closed loop response and disturbance suppression. In (10), Dahlin et al extended the ideas developed in (9) to a multivariable control. Lopez et al (11) showed how to tune proportional digital controllers using error criteria.

Floersch (12) designed a digital control for temperature during plastic parts fabrication. The extreme non linearity of the heating fluid dynamics and the integrating amplifier is minimised by leaving the fluid loop under analog PI control. The use of self-tuning is shown by McGreavy and Awda (13) for a heat exchanger using the extended Kalman filter for parameter - updating. The delay in the control action is expressed in the form of a Padé approximation which was used by Smith (14) in his pioneer work in controlling processes with dead-time.

Using state-space methods in process industries can be made difficult by the traditional industry's approach to control system design. Borer (15) proposes a way out of this situation by introducing a control strategy with a mastercontroller microprocessor structure.

An interesting feasibility study of the application of microprocessors to self-tuning controllers was made by Clarke et al (16).

### 2.2.3 Servo Digital Control

The use of digital filters for servo compensation has been analysed by Stevenson (17) using frequency techniques. McDonnel (18) showed how to implement difference equations in microcomputers for servo compensation, before the very large scale integration stage was reached in the electronic industry.

A microprocessor speed control was developed by Lin (19) to control the speed of a chopper-fed DC motor. In this work an Intel 8080 microcomputer is used and digital measurement of speed is achieved through a digitizer. The pulses are counted one by one by the software routine and the microcomputer introduces only proportional compensation in the loop. Hing (20) used proportional plus integral (PI) control to stabilise the current loop and proportional plus integral plus derivative (PID) in the velocity loop of a DC thyristor driven motor. It is found that the direct control of speed is better than the cascaded due to limit cycles (non linearities).

An interesting closed loop digital controller is shown by Doherty et al (21). It uses an Intel 8080 microprocessor as the central processor and is intended to be applicable to a broad spectrum of power drives. In order to avoid position orders from the master computer at an excessively high rate, the number of samples per second is reduced to 20 . The largesignal and linear control modes are features of this controller which, with its feed forward compensation, indicates the way to flexible closed loop control.

A digital method for $D C$ motor speed control is given by Maloney and Alvarado (22). Using a digital tachometer the actual
speed is measured and then compared with a pre-set value. The achieved resolution is not good and the inclusion of a microprocessor in the process is necessary. Burger and Ronchinsky (23) introduce a microprocessor in a well established closed loop design. The microprocessor generates a pulse train to the digital servo and the limitations are quite apparent. The prediction of increased take over of loop control functions by the microprocessor came true.

Matsumoto (24) closes the position and velocity loops using a microcomputer and introduces prediction filters to compensate for the delays introduced by computation and sampling operations. Two different sampling periods are used for the position and velocity loops. Using the integral of squared errors he tries to show the effect of filter constants but does not show great improvement. The state-space method is used but notation is very poor.

The use of a microcomputer for closing the position loop around a thyristor controlled DC drive is described by Ernsberger (25). The interesting point about it is the possibility to work in a multi-computer environment by the use of serial links.

An illustration of a successful application of microprocessors to servo drives is given in reference 26 . The microprocessor controlled AC motor feed drive is intended to be used in a multidrive arrangement and therefore the necessary communication link is provided. The EPROM memory chip contains all the necessary functions and gives good flexibility in any updating. The functions normally implemented are closed-loop control, torque monitoring, communication with the system management, faults reporting and diagnosis of problems.

The general position servo control through software is discussed by Smith (27). Oriented towards machine tool application, he discusses the following error calculation and the introduction of gain break points by software.

Plant et al (28) introduce a microprocessor control of position or speed of SCR DC motor drive which uses a half-wave single phase supply. The firing angle is used as the control function. The state-space analysis and optimum controller design using a look-up table for implementation, show the use of digital control and microprocessor at their best.

The use of sliding mode control in a multiple-element mechanical linkage is reported by Klein and Maney(29). The microprocessor causes the system to move or slide along a predetermined line in phase-space. The control can cope with non linearities but the effects of chatter must be evaluated as lags in the control path are sometimes inevitable.

The use of an on-off controller for position control is presented by Hartman et al (30). The system is made adaptive by a learning process for the switching action related to position.

The design and application of a PID predictor controller using a microprocessor is given by Aylor et al (31). The controller is applied to a manipulator. For reasons of computer throughput the control function is not calculated at each sampling interval, but predicted by a simple linear extrapolation before each sampling instant. The integration effect provided by the actuator allows the removal of the integral from the controller action and stability curves are provided to find the gains for the controller.

Schnieder (32) uses a microprocessor for direct control of thyristors on a DC drive. The firing control algorithm is applied to a three phase thyristor AC/DC converter. Speed and current loops are closed by the microprocessor and PI control action with output limiting is used in both loops. Current signal is obtained by A/D conversion but speed measurement is performed by digital techniques. Oumanar et al (33) show the design of an optimal adaptive current loop for a DC motor supplied by a dual converter (two thyristor bridges in antiparallel). The adaptation is achieved by having a varying gain which is defined at each operating point for continuous and discontinuous mode of operation. The speed control of an induction motor using a microprocessor and optimal control design method is discussed by Tsuchiya (34). Coristraints such as input dead-time and limitation of input and state variable values are imposed on the design. In order to simplify the general optimal regulator problem, several performance indices are introduced. According to the imposed constraints the microprocessor chooses the required path to follow and achieve a sub-optimal control.

The most recent developments in the area of digital control applied to servo drives and new developments on the drives themselves are pictured by Bollinger et al (35). It is emphasized the increasing role the microprocessor must have in this area and the flexibility required by the new control systems to cope with ever increasing demand. The digital control of the velocity loop is indicated as a new development together with the direct digital control of the power amplifier. The use of state-variable control is advocated and optimisation of design by simulation is indicated as a required tool. Identification by a model-fitting using a recursive leastsquare algorithm is shown as the first step to adaptive control. The use of microprocessor controlled servos in hierarchical control systems is described by Brussel et al (36). The concept of hierarchical control systems applied to manufacturing processes is discussed. Two level interpolation is described and a microcomputer controlled feed drive, in which the thyristor is directly triggered by the controller, has its implementation given in some detail. Brussel is one of the co-authors of reference 35 .

Brussel and Vastmans (37) raise the point that the general trend of decentralising the control function makes direct digital control of feed drives interesting. Some identification schemes to be implemented in microprocessors are discussed and simple adaptive algorithms based upon real-time identification are described.

Duffie and Bollinger (38) describe a technique for directly controlling SCR's for the closed-loop position and velocity control of DC drives. For moderate performance it is possible to control both velocity and position loops using a simple encoder. Position and velocity resolution is discussed and the limitations on the digital measurement of velocity are raised. . The position loop is just proportional and the velocity loop has got a PID controller which has its gains optimised by minimising a summation of absolute velocity errors.

The change from the traditional PID controller to a state-variable parameter optimised one is well represented by Stute and Hesselbach (39). They call it a more efficient feedback control strategy.

The position loop of a CNC machine is closed through a statevariable feedback controller which minimises a quadratic performance index in position and control function. As only position and velocity are available by direct measurement, an observer is used to estimate acceleration. It is good to mention that an analog velocity loop is maintained around the drive. The control strategy is developed for applications where a high dynamic of the position control loop is required or the control of a robot. In high speed machining, dynamic errors cannot be neglected. If for reasons of design the mechanical structure cannot be rigid enough, the situation gets worse. Stute et al (40) describe a control method by means of which positioning can be improved in these cases. The method is based on the same lines followed in reference 39 and a second order observer is used. The results, which are presented, show a real improvement in positioning performance for the optimum state-variable controller over the simple proportional controller. The restriction of following errors less than 0.1 mm and velocities around corners of not less than $1 \mathrm{~m} / \mathrm{min}$ is achieved by the optimum controller.

In the area of low powered servos Rao (41) gives a good description of the electronics design and signal. processing involved in each loop for the three interdependent servos in a quadruplex colour video-tape recorder. His measure and optimise technique is in line with state-space methods.

### 2.3 MULTIPROCESSOR SYSTEMS IN CONTROL APPLICATIONS

The use of multicomputer numerical control systems is well established and has its origins with the advent of the microprocessor. One of the first commercial controllers with this characteristic is described by Schmitt and Chang (42). The system uses a common memory block and machine tool control tasks are distributed between microcomputers. Popa (43) and Prasad (44) developed a multicomputer numerical control system based on the Motorola 6800 microprocessor. It used a DMA bus structure and could cope with synchronous and asynchronous operation.

A totally modular multiprocessor control system is described by Stute et al $(45,46)$. The MPST system has a parallel bus which
couples all used modules. The hardware and software interfaces are standardised and communication through the bus is transparent to the user. The application of the MPST system to control some types of machine tool is described by Stute and Klemm (47). The splitting of the overall NC function is done into five independent function blocks. Each function block performs a separate, self-contained task of the system. This minimises the data transfer between function blocks. Each function block is then associated to one microcomputer, or more if needed, and the complete system is defined by the particular application. .. Examples of application to a lathe, gear hobbing machine, gear shaping machine and a measuring machine are given.

Ilic et al (48) analysed different architectures of multiprocessor systems for application to control systems of machine tools and robots. Consideration was given to the interprocessor comminication regarding flexibility, modularity, self-testing capability and price. The final decision is to use a common memory block accessible by a common bus and dedicated interrupt lines for synchronisation between processors.

Dalzell et al (49) have designed a microprocessor hierarchical control system for machine tools based on the Motorola 6800 microprocessor. The system consists of a delegator computer connected by an IEEE-488 bus to a number of slave computers which control or monitor system variables. The system was applied to a lathe, moved by stepper motors, where no feedback is used.

Luh and Lin (50) describe the use of multiprocessor controllers for mechanical manipulators. The complex equations and the imposed minimum sampling frequency (bandwidth) are conflicting. Solution is found by the use of a multiprocessor. An algorithm is presented which arranges the schedule of computation of a task and distributes the computational load efficiently among the processors so that the total computing time is minimum.

The control of special purpose machine tools by multi-microprocessor is described by Kuisma et al (51). A hierarchical control system is used for low level position and speed control and high level work cycle control. Hierarchical computer structure for quality control is discussed by Rembold (52). A multi-bus system is proposed in order to cope with increasing distances between data acquisition equipment and central computer.

The structure of a microprocessor based distributed system for the control of generating plant is discussed by Maples and Jervis (53). Jervis (54) discusses general characteristics of the communication system and control nodes that are required to control a generating plant. Each computer is at the node of the communications network.

## CHAPTER 3

## ASPECTS OF STATE-VARIABLE DIGITAL CONTROL

### 3.1 INTRODUCTION

This chapter is concerned with some aspects of state-variable digital control which are relevant to this work. It deals only with linear and time-invariant systems.

The state-variable approach requires the continuous system equations to be represented in the standard form $\dot{x}=A x+B y$. For a linear and time-invariant system, defined by a set of linear differential equations, it is always possible to find its representation in standard form. The choice of appropriate state-variables is not treated in this work but physical variables are highly recomended. This aspect of representation is treated in Section 3.2 . 1 with mention of linear transformations and canonical forms.

The digital controller sees the continuous system through the sampling operation and, therefore, a discrete system representation is necessary. Section 3.2.2 deals with this aspect of representation.

Section 3.3 describes the design sequence adopted in this work and assumes the existence of a system model, from where the discrete stateequations can be obtained.

With the assumption that all the state-variables are available, two methods for the control law design are discussed. The first, given in Section 3.4.1, is based on pole-placement design and the second, given in Section 3.4 .2 , is based on optimal control.

The non-available state-variables must be estimated before the control law can be implemented. In this work, estimation is achieved by using observers and Section 3.5 deals with their design. Emphasis is given to the reduced-order observers.
3.2 CONTINUOUS AND DISCRETE STATE EQUATIONS

### 3.2.1 Continuous State Equations

For a system defined by a set of continuous, linear and constant coefficient differential equations, it is always possible to find a correspondent set of first order matrix differential equations as

$$
\dot{x}(t)=A x(t)+B_{u}(t)
$$

where $x(n \times 1)$ is the state vector and $\mu(1 \times 1)$ is the control function. Matrices $A$ and $B$ have dimensions ( $n \times n$ ), ( $n \times 1$ ) respectively and are independent of time, $t$. The output can normally be represented as a linear combination of the state, $x$, as

$$
y(t)=C x(t)
$$

where matrix $C$ is time-invariant and has dimensions (1xn). The System has single input and output.

The representations given by 3.1 and 3.2 are not unique of course as any non-singular linear transformation $T_{1}$ allows an alternative representation of the same system. For instance, defining a new alternative state as

$$
\theta(t)=T_{1} x(t)
$$

where $\mathrm{T}_{1}$ is non-singular, the alternative system equations are given by

$$
\begin{array}{ll}
\dot{\theta}(t)=E \theta(t)+F u(t) & 3.4 \\
y(t)=D x(t) & 3.5
\end{array}
$$

where

$$
\begin{aligned}
& \mathrm{E}=\mathrm{T}_{1} A T_{1}^{-1} \\
& \mathrm{~F}=\mathrm{T}_{1} \mathrm{~B}
\end{aligned}
$$

and

$$
\mathrm{D}=\mathrm{CT}_{1}^{-\mathrm{l}}
$$

The use of linear transformation makes some design procedures easier as it allows the representation of the system equations in specially structured forms called canonical forms (4, 62).

### 3.2.2 Discrete State Equations

The way a digital controller sees a continuous system, represented by 3.1 and 3.2, can be represented as in Figure 3.2.1.


T - Sampling Period
Figure 3.2.1 Continuous System and Sampling Operation

Basically, the digital controller samples the output $y(t)$, operates on the sequence $y(k)$, by a set of difference equations and sends out a sequence of numbers $u(k)$ as inputs to the continuous system. In order to relate the samples of the output $y(k)$ to the samples of the control input $u(k)$, we must solve equation 3.1.

Before we solve equation 3.1, a generalisation is necessary. This concerns the delay in the control action. The digital controller will almost certainly introduce a computation delay which by itself or added to delay in the controlled system must be accounted for in the controller design. Introducing a delay in the control action, representation 3.1 gives

$$
\dot{x}(t)=A x(t)+B u\left(t-T_{D}\right)
$$

where $T_{D}$ is the delay.

Solving 3.6, we have (55)

$$
x(t)=e^{A(t-t 0)} x(t 0)+\int_{0}^{t} e^{A(t-\zeta)} B u\left(\zeta-T_{D}\right) d \zeta
$$

We wish this solution over one sampling period, to obtain a difference equation, therefore we make

$$
t=k T+T
$$

and

$$
\text { to }=\mathrm{kT}
$$

where $k$ is an integer and $T$ is the sampling period. Using 3.8 in 3.7 gives

$$
x(k T+T)=e^{A T} x(k T)+\int_{k T}^{k T+T} e^{A(k T+T-\zeta)} B u\left(\zeta-T_{D}\right) d \zeta
$$

If we substitute $\eta=k T+T-\zeta$ for $\zeta$ in the integral (2) we find

$$
x(k T+T)=e^{A T} x(k T)+\int_{0}^{T} e^{A \eta}{ }_{B} \mu\left(k T+T-T D^{-\eta) d \eta}\right.
$$

The delay, $T_{D}$, can now be separated into an integral number of sampling periods plus a fraction by defining

$$
T_{D}=a T-m
$$

where $a \geqslant 0$

$$
0 \leqslant m<T
$$

With this substitution 3.10 gives

$$
x(k T+T)=e^{A T} x(k T)+\int_{0}^{T} e^{A \eta} B u(k T+T-a T+m-\eta) d \eta
$$

Assuming a zero-order hold, as indicated in Figure 3.2.1, the integral in 3.12 can be broken into two parts as follows:

$$
\begin{align*}
x(k T+T)= & e^{A T} x(k T)+\int_{0}^{m} e^{A \eta} B d \eta u(k T-a T+T)+ \\
& +\int_{m}^{T} e^{A \eta} B d \eta u(k T-a T)
\end{align*}
$$

Simplifying the representation we write

$$
x(k T+T)=\Phi x(k T)+L_{l u}(k T-a T)+L_{2} u(k T-a T+T)
$$

where

$$
\Phi^{\prime \prime}=\mathrm{e}^{\mathrm{AT}}
$$

$$
\begin{align*}
& \mathrm{L}_{1}=\int_{m}^{\mathrm{T}} \mathrm{e}^{\mathrm{An}} \mathrm{Bdn} \\
& \mathrm{~L}_{2}=\int_{0}^{m} e^{\mathrm{An}} \mathrm{Bdn}
\end{align*}
$$

In order to represent 3.14 in standard state-space form, it is necessary to consider the cases for $a=0, a=1$ and $a>1$. For $a=0$ and $m \neq 0$ we have $T_{D}=-m$, according to 3.11 , which implies no delay but prediction. This is equivalent to the modified $z$-transform as described by Tou (56) and is not discussed in this work.

For $a=1$ equation 3.14 , with the sampling period left aside for easier representation, gives:

$$
x(k+1)=\Phi^{*} x(k)+L_{1} u(k-1)+L_{2} u(k)
$$

The standard state-space representation requires the elimination of $u(k-1)$ from the right hand side which is done by defining a new state as

$$
x_{n+1}(k) .=u(k-1)
$$

With increased dimension the state-equations are

$$
\left[\begin{array}{r}
x(k+1) \\
x_{\eta+1}(k+1)
\end{array}\right]=\left[\begin{array}{ll}
\Phi^{\prime} & L_{1} \\
0 & 0
\end{array}\right]\left[\begin{array}{l}
x(k) \\
x_{\eta+1}(k)
\end{array}\right]+\left[\begin{array}{l}
L_{2} \\
1
\end{array}\right] u(k)
$$

and

$$
y(k)=\left[\begin{array}{ll}
C & 0
\end{array}\right]\left[\begin{array}{r}
x(k) \\
x_{\eta+1}^{(k)}
\end{array}\right]
$$

For $a>1$ the equations are

$$
x(k+1)=\Phi^{*} x(k)+L_{1} u(k-a)+L_{2} \mu(k-a+1)
$$

and we must now eliminate the past controls up to $u(k)$.
Introducing new state-variables as

$$
\begin{gathered}
x_{\eta+1}(k)=u(k-a) \\
x_{\eta+2}(k)=\mu(k-a+1) \\
\vdots \\
x_{\eta+a-1}(k)=u(k-2) \\
x_{n+a}(k)=u(k-1)
\end{gathered}
$$

the state-equations are transformed to

$$
\left[\begin{array}{c}
x_{(k+1)} \\
x_{n+1}(k+1) \\
x_{n+2}(k+1) \\
\vdots \\
\vdots \\
x_{n+a-1}(k+1) \\
x_{n+a}(k+1)
\end{array}\right]=\left[\begin{array}{cccccc}
\Phi^{*} & L_{1} & L_{2} & 0 & \ldots & 0 \\
0 & 0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 \\
& & & & & \\
0 & 0 & 0 & 0 & \ldots & 1 \\
0 & 0 & 0 & 0 & \ldots & 0
\end{array}\right]\left[\begin{array}{c}
x_{(k)} \\
x_{n+1}(k) \\
x_{n+2}(k) \\
\vdots \\
x_{n+a-1}(k) \\
x_{n+a}(k)
\end{array}\right]+\left[\begin{array}{c}
0 \\
0 \\
0 \\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\\
0(k)
\end{array}\right]
$$

and

$$
y(k)=\left[\begin{array}{ll}
c & 0
\end{array}\right]\left[\begin{array}{l}
x(k) \\
x_{n+1}(k) \\
x_{n+2}(k) \\
x_{n+a-1}(k) \\
x_{n+a}(k)
\end{array}\right]
$$

The implementation of matrices $\Phi^{*}, L_{1}$ and $L_{2}$, as defined by equations 3.15, is discussed in Chapter 4.

### 3.3 CONTROLLER DESIGN SEQUENCE

The design of digital controllers generally follows the sequence (4) given in Figure 3.3.1.


## Figure 3.3.1 Design of Control Algorithm

Depending on the design method and the application, exact or approximate mathematical models are used as the basis for design. At this moment, it is assumed that a model of the system to be controlled is available and that it is represented in the state-space standard form as given by equations 3.17 and 3.18 .

The use of state-space methods for controller design is attractive for the design is done in two independent steps. In the first step it is assumed that all the state-variables are available and a control law (controller) can be found. Normally we do not have all the statevariables available for several reasons like difficult measurement or inaccessibility and the cost of measurement instruments for all variables. Therefore, the second step is the design of an observer which will estimate all variables or just the ones that are not available. First we will discuss the design of the control law and after the design of the observer.

### 3.4 DESIGN OF THE CONTROL LAW

In this work we discuss two design methods for the state-variable controller. The first one is based on pole-placement and the second is based on optimal control. For the pole-placement method a certain characteristic equation for the controlled system is prescribed. On the optimal control method a:quadratic performance index, based on the
state-variables and control function, is minimised.
The control law for stace-variable control is the feedback of a linear combination of all the states, that is

$$
u(k)=-K x(k)
$$

where $K$ is the feedback gain matrix of dimension (lxn), as 3.20 , which must be found by the design method.

$$
K=\left[\begin{array}{llll}
K_{1} & K_{2} & \cdots & K_{n}
\end{array}\right]
$$

It is assumed, in this work, that the system is controllable (57). Using Figure 3.2 .1 , the control law can be represented as in Figure 3.4.1


Figure 3.4.1 State-Variable Feedback
3.4.1 Pole Placement Design Method

A general form for the extended state-equations 3.17 and 3.18 can be written as
$x(k+1)=\Phi x(k)+L u(k)$

Substituting equation 3,19 on 3.21 we have
$x(k+1)=\Phi x(k)-L K x(k)$

The $z$-transform of 3.22 is
$(z I-\Phi+L K) x(z)=0$

Thus the characteristic equation of the controlled system, in closed loop, is
$\operatorname{det}[z I-\Phi+L K]=0$
The pole-placement design consists of finding the elements of K so that the roots of 3.24 are in specified locations. If the desired characteristic equation is given by
$\alpha_{c}(z)=\left(z-\beta_{1}\right)\left(z-\beta_{2}\right)\left(z-\beta_{3}\right) \ldots=0$
then the required elements of $K$ are found by matching coefficients in 3.24 and 3.25 or
$\alpha_{c}(z)=\operatorname{det}[z I-\Phi+L K]$
with
$\alpha_{c}(z)=z^{\eta}-\alpha_{1} z^{n-1}-\ldots-\alpha_{n}$
An interesting formula which has been derived by Ackermann (58) and the proof repeated by Franklin (2), is used in this work to find the elements of K . It is based on the transformation of the system equations to a canonical form and the use of the Cayley-Hamilton theorem. The relation is
$K=\left[\begin{array}{llll}0 & \ldots & 0 & 1\end{array}\right]\left[\begin{array}{lllll}\mathrm{L} & \Phi \mathrm{L} & \Phi^{2} \mathrm{~L} & \ldots & \Phi^{\eta-1} \mathrm{~L}\end{array}\right]^{-1}{ }_{\alpha_{c}}(\Phi)$
where $C_{0}=\left[\begin{array}{ll}L & \Phi L \ldots\end{array}\right]$ is called the controllability matrix, $\eta$ is the order of the system and we change $\Phi$ for $z$ on 3.27 to have
$\alpha_{c}(\Phi)=\Phi^{\eta}-\alpha_{1} \Phi^{\eta-1}-\alpha_{2} \Phi^{\eta-2} \ldots-\alpha_{\eta} I$
where the $\alpha_{i}$ 's are the coefficients of the desired characteristic equation as given by 3.27.

Equation 3.28, or Ackermann's formula, is implemented in Chapter 4 as an interactive program which calculates the elements of $K$ for a characteristic equation specified by the user.

### 3.4.2 Optimal-Control Design Method

The optimal control design method has an advantage over the pole-placement design method because it can handle multi-input systems with no difficulty. This work is concerned with single-output systems but further work probably will require handling of multi input/output systems and this method is recommended.

The important feature of optimal control is the establishment of an analytic index of performance for the system and design of the controller so as to optimise the index selected (60). The choice of the performance index is made by the designer and in this work, as we are dealing with linear systems, a quadratic performance index (59) is used.

With the discrete system equations defined by
$x(k+1)=\Phi x(k)+L u(k)$
$\mathrm{u}(\mathrm{k})$ is found by minimising the performance index
$I P=\sum_{k=0}^{N}\left[\frac{1}{2} x^{T}(k) Q_{1} x(k)+\frac{1}{2} \mu^{T}(k) Q_{2} u(k)\right]$
$Q_{1}$ and $Q_{2}$ are symmetric weighting matrices to be selected by the designer based on his choice of the relative importance of the various states and controls. $Q_{l}$ is positive semi-definite and $Q_{2}$ is positive definite.

In our case, for a single-input system, $Q_{2}$ is a scalar and different from zero, as some weight is almost always given to the control, to avoid large components in the control gain and possibly actuator saturation.

The problem of minimizing 3.31 subject to the constraints imposed by the system equations 3.30 , can be solved using the method of Lagrange multipliers.

For convenience (59) the augmented performance index.is written

$$
I P=\sum_{k=0}^{N}\left\{\frac{1}{2} x^{T}(k) Q_{1} x(k)+\frac{1}{2} u^{T}(k) Q_{2} u(k)+\lambda^{T}(k+1)[-x(k+1)+\Phi x(k)+L u(k)]\right\}
$$

The minimisation process gives

$$
\begin{align*}
& \frac{\delta I P}{\delta \mu(k)}=u^{T}(k) Q_{2}+\lambda^{T}(k+1) L=0 \\
& \frac{\delta I P}{\delta \lambda(k+1)}=-x(k+1)+\Phi x(k)+L u(k)=0
\end{align*}
$$

and

$$
\frac{\delta I P}{\delta x(k)}=x^{T}(k) Q_{1}-\lambda^{T}(k)+\lambda^{T}(k+1) \Phi=0
$$

Combining 3.33, 3.34 and 3.35 , we have a set of coupled difference equations defining the optimal solution of $x(k), \lambda(k)$ and $\mu(k)$ provided the initial or final conditions are known. Initial conditions on $x(k)$ must be given but usually $\lambda(0)$ would not be known and the end point ( $k=N$ ) is used to establish a final condition. From equations 3.30 and 3.31 it is clear that $u(N)$ will be zero for the minimum IP because $u(N)$ has no effect on $\lambda(N)$. So equation 3.33 indicates that $\lambda(N+1)=0$ and equation 3.35 gives a suitable condition

$$
\lambda(N)=Q_{1} x(N)
$$

With initial and final conditions defined, the solution to the optimal control problem is completely specified. This is a two-point boundary-value problem and one method to solve it is called sweep method by Bryson and Ho (59) and is used in this work. Its implementation is given in Chapter 4 as an interactive program which calculates the elements of $K$ for the optimal solution based on the user's choice of weighting matrices $Q_{1}$ and $Q_{2}$.

### 3.5 DESIGN OF THE OBSERVER

The assumption made, when designing the control law, that all the statevariables were available is not always true. For reasons already mentioned, some or even all the state-variables are not available or one just does not want to use them because the measured signals are too noisy. In our case, for example, we decided to use only the position. transducer for feedback purposes. The cost of transducers, in general, is high and if good results can be achieved just with a position transducer the total cost of the controller will be much lower.

In order to be able to implement the control law, the non-available state-variables have to be estimated and there are two basic state estimation techniques: the Kalman filter and the Luenberger observer (5). The Kalman filter gives an estimate of the state which is a solution to an optimal estimation problem and will not be discussed in more detail in this work. The Luenberger observer is the estimation technique used in this work and we will discuss it in some detail. Luenberger (61) was the pioneer of this estimation technique which has got his name. Further developments are given by Luenberger ( 62,63 ) and Gopinath (64).

### 3.5.1 Full-Order Observer

The basic idea behind the observer technique is the design of a linear system $\left(S_{2}\right)$ which is constructed in such a way that its state $\hat{\boldsymbol{x}}$ can easily be observed and such that the state of $\mathrm{S}_{2}$ tends to the state of the real controlled system ( $S_{1}$ ). System $S_{2}$ will consist of a model of $S_{1}$ driven by atunction which is equal to the sum of the input to a weighted error term which is the difference between the state of $S_{1}$ and that of $S_{2}$, as shown in Figure 3.5.1(64).


Figure 3.5.1 State Estimation by Observer

It is clear from Figure 3.5 .1 that we have an estimation, $\hat{x}$, for all the state-variables, $x$. In most cases some of the statevariables are available by direct measurement so that they do not need to be estimated and a reduced order observer can be used (64). Before we discuss the reduced order observer, the observer design according to Figure 3.5.1 is presented.

According to Figure 3.5.1 the observer system $\left(S_{2}\right)$ is defined by
$\hat{x}(k+1)=\Phi \hat{x}(k)+\operatorname{LOC}[x(k)-\hat{x}(k)]+\operatorname{Lu}(k)$

If the state error is defined as
$\tilde{x}(k)=\boldsymbol{x}(k)-\hat{x}(k)$ 3.38
equations 3.21 and 3.37 give
$\tilde{x}(k+1)=\Phi \tilde{x}-\operatorname{LOC} \tilde{x}$ 3.39

We want $\tilde{x}$ to decrease to zero according to some dynamics, that is, the characteristic equation of 3.39 , or
$\operatorname{det}[\Phi-\operatorname{LOC}]=0$
should be some prescribed polynomial. This is the same problem as imposed on the pole-placement design method to find $K$, but now we are looking for LO such that 3.40 is a prescribed polynomial. If the controlled system, of order $\eta$, given by
$x(k+1)=\Phi x(k)+L u(k)$
$y(k)=C x(k)$
is completely observable (57), that is
$\operatorname{rank}\left[C^{T} \quad L^{T} C^{T}=\cdots \quad\left(L^{T}\right)^{n-1} C^{T}\right]=n$
then it is possible to find LO to match 3.40 to any prescribed polynomial.

### 3.5.2 Reduced-Order Observer

When some of the state-variables are available, a reduced order observer can be designed (64).

Calling $x_{a}$ the available state-variables and $x_{b}$ the variables to be estimated, equations 3.41 can be written as
$\left[\begin{array}{l}x_{a}(k+1) \\ x_{b}(k+1)\end{array}\right]=\left[\begin{array}{ll}\Phi_{a a} & \Phi_{a b} \\ \Phi_{b a} & \Phi_{b b}\end{array}\right]\left[\begin{array}{l}x_{a}(k) \\ x_{b}(k)\end{array}\right]+\left[\begin{array}{l}L a \\ L b\end{array}\right] \quad \mu(k)$

Then
$x_{a}(k+1)=\Phi_{a} x_{a}(k)+\Phi_{a b} x_{b}(k)+L_{a} u(k)$
and
$x_{b}(k+1)=\Phi_{b a} x_{a}(k)+\Phi_{b b} x_{b}(k)+L_{b} u(k)$

Since $x_{a}$ does not give any information about $x_{b}$ the only information available is given by equation 3.44 . Writing 3.45 and 3.44 in other forms, we have
$x_{b}(k+1)=\Phi_{b b} x_{b}(k)+\Phi_{b a} x_{a}(k)+L_{b} u(k)$
$x_{a}(k+1)-\Phi_{a a_{a}}(k)-L_{a} u(k)=\Phi_{a b} x_{b}(k)$

Equations 3.46 and 3.47 have the same configuration of equations 3.41 , for the states $x_{b}$. The left hand side of 3.47 is available by measurement and the right hand side of 3.46 contains an expanded and known input term, or
$(\mathrm{Lu}) *=\Phi_{\mathrm{ba}} x_{a}(\mathrm{k})+\mathrm{L}_{\mathrm{b}} \mathrm{u}(\mathrm{k}) \quad$ known input
$y^{*}=x_{a}(k+1)-\Phi_{a a} x_{a}(k)-L_{a} u(k) \quad$ known output

By similarity, we can apply the observer equations, 3,37 , and find an observer for the system defined by equations 3.46 and 3.47 . The observed $x_{b}$ value, $\hat{X}_{b}$, is given by
$\hat{x}_{b}(k+1)=\Phi_{b b} \hat{x}_{b}+\operatorname{LO\Phi }{ }_{a b}\left[x_{b}(k)-\hat{x}_{b}(k)\right]+$

$$
+\phi_{b a} x_{a}(k)+L_{b} u(k)
$$

Then $\tilde{x}_{b}(k)=x_{b}(k)-\hat{x}_{b}(k)$ implies
$\tilde{x}_{b}(k+1)=x_{b}(k+1)-\hat{x}_{b}(k+1)=\left[\Phi_{b b}-L O \Phi_{a b}\right] \tilde{x}_{b}(k)$

Gopinath (64) proved that if the system given by equation 3.41 : is completely observable then the system given by 3.46 and 3.47 is completely observable. Therefore we can choose LO in 3.49 to match any characteristic equation and make the state error $\tilde{x}_{b} \longrightarrow 0$ as fast as we want.

The characteristic equation of 3.49 can be written as
$\operatorname{det}\left[z I-\Phi_{b b}+L O \Phi_{a b}\right]=0$
Comparing 3.50 to 3.24 we see that the feedback gains LO and $K$ are in different positions in relation to the output matrices $\Phi_{a b}$ and L. As the characteristic equation is not changed by transposition we can write 3.50 as
$\operatorname{det}\left[z I-\Phi_{b b}^{T}+\Phi_{a b}^{T} L 0^{T}\right]=0$
and now 3.24 and 3.51 have the same configuration. This modification, and the fact that we assume a single output system, allow us to use the program developed for pole-placement design of the control law to design the observer. A sample of this procedure can be seen in Appendix 10.5.

After the observer feedback gain, LO, has been found the observer equations can be arranged in a convenient way to be solved by the controller.

Using equation 3.44 , equation 3.48 gives

$$
\begin{align*}
\hat{x}_{b}(k+1) & =\left[\Phi_{b b}-L 0 \Phi_{a b}\right] \hat{x}_{b}(k)+\left[\Phi_{b a}-L 0 \Phi_{a a}\right] x_{a}(k) \\
& +\left[L_{b}-L O L_{a}\right] u(k)+L O x_{a}(k+1)
\end{align*}
$$

Introducing the idea developed by Gopinath (64), we define a variable $\omega$ such as
$\omega(k+1)=\hat{x}_{b}(k+1)-L 0 x_{a}(k+1)$
which, for initial conditions set to zero, gives
$\hat{x}_{b}(k)=\omega(k)+L 0 x_{a}(k)$

Using equations 3.53 and 3.54 , in equation 3.52 , the observer equation can be written as

$$
\begin{align*}
\omega(k+1) & =\left[\Phi_{b b}-L 0 \Phi_{a b}\right] \omega(k)+ \\
& +\left[\left[\Phi_{b a}-L 0 \Phi_{a a}\right]+\left[\Phi_{b b}-L 0 \Phi_{a b}\right] L 0\right] x_{a}(k)+ \\
& +\left[L_{b}-\text { LO } L_{a}\right] u(k)
\end{align*}
$$

with

$$
\omega(k)=\hat{x}_{b}(k)-L 0 x_{a}(k)
$$

It is clear from equations 3.55 and 3.54 that the observer equations are solved in two stages. At each sampling interval the estimated variables are obtained from equation 3.54 and variable $\omega$ is updated for the next sampling period according to 3.55. The use of these equations can be seen in Chapter 10. The coefficient matrices in equation 3.55 are calculated by a program called OBSEQU which FORTRAN listing is given in Appendix 4.6 and sample run given in Appendix 10.6.

The controller obtained by combining the control law with the observers, as designed in this chapter, is basically a regulator. The design reduces the impact of disturbances upon the output. The incorporation of reference input tracking (2, 4, 65) is not discussed in this work.

## CHAPTER 4

> DIGITAL CONTROLLER COMPUTER AIDED DESIGN AND CLOSED-LOOP SIMULATION

### 4.1 INTRODUCTION

This chapter is concerned with the implementation of a computer aid for the design methods developed in Chapter 3.

The use of state-space methods, in digital control, requires the representation of the system state-equations in discrete form. This is basically achieved by using the exponential of a matrix, $e^{A T}$, and Section 4.2 describes the computer program developed to this end. It accepts the continuous state-equations, sampling period and delay, and gives the matrices for the discrete stateequations.

The pole-placement design method is implemented by the ACKER program, which is developed in Section 4.3. This program accepts the discrete-state equations and the desired pole-locations, on the s-plane for easier reference, and gives the feedback gain matrix as the output.

The optimal-control design method is implemented in Section 4.4 by the SWEEP program. This program solves the two-point boundary value problem, discussed in Chapter 3, by the sweep method and its output is the time varying feedback gain matrix.

The closed-loop simulation, for the final system, can be done by using the TIMELOCUS program. It is developed in Section 4.5, and offers the possibility of displaying time-response, with step or ramp input, for any system variable and the root locus for the closed-loop system.

### 4.2 CONTINUOUS TO DISCRETE STATE-EQUATIONS - STATE PROGRAM

The use of the design techniques described in Chapter 3, requires the representation of the system continuous state-space equations in discrete form, as represented by equations 3.17 and 3.18 .

Repeating here equation 3.17 , for reference, we have

$$
\left[\begin{array}{c}
x_{0}(k+1) \\
x_{n+1}(k+1)
\end{array}\right]=\left[\begin{array}{ll}
\Phi^{*} & L_{1} \\
0 & 0
\end{array}\right]\left[\begin{array}{c}
x(k) \\
\cdot \\
x_{\eta+1}(k)
\end{array}\right]+\left[\begin{array}{l}
L_{2} \\
1
\end{array}\right] u(k)
$$

where

$$
\begin{align*}
& \Phi^{*}=e^{A T} \\
& L_{1}=\int_{m}^{T} e^{A \eta_{B d \eta}} \\
& L_{2}=\int_{0}^{m} e^{A \eta_{B} B d \eta}
\end{align*}
$$

Therefore, matrices $\Phi^{*}, L_{1}$ and $L_{2}$ must be found by solving equations 4.2, 4.3 and 4.4. As we assume a time-invariant system matrix $B$ can be taken out of the integrals.

Making $\sigma=n-m$ we can convert $L_{1}$ to a form similar to the integral $\mathrm{L}_{2}$ as

$$
L_{1}=\int_{0}^{T-m} e^{A(m+\sigma)} d \sigma B
$$

From equation 3.11 we have the delay

$$
\mathrm{T}_{\mathrm{D}}=\mathrm{aT}-\mathrm{m}
$$

where $T$ : sampling period

$$
\text { a: integer } \geqslant 0
$$

$$
0 \leqslant m<T
$$

With $m$ as a constant we can write equation 4.5 as

$$
L_{1}=e^{A m} \int_{0}^{T-m} e^{A \sigma} d \sigma B
$$

For easier representation we define

$$
\stackrel{\Phi^{*}(b)}{ }\left(e^{A b}\right.
$$

and

$$
\Psi(b)=\frac{1}{b} \int_{0}^{b} e^{A \sigma} d \sigma
$$

Using equations 4.8 and $4.9, \stackrel{*}{\Phi}, L_{1}$ and $L_{2}$ are given by

$$
\begin{array}{ll}
\Phi^{*}=\ddot{\Phi}(T) & 4.10 \\
L_{1}=\ddot{\Phi}(\mathrm{m}) \Psi[(T-m)](T-m) B & 4.11 \\
\mathrm{~L}_{2}=\Psi(\mathrm{m}) \mathrm{mB} & 4.12
\end{array}
$$

Now, in order to find $\Phi^{*}, \mathrm{~L}_{1}$ and $\mathrm{L}_{2}$ we must solve equations 4.8 and 4.9.

Equation 4.8 is a basic matrix exponential which series expansion is defined as

$$
\Phi^{*}(b)=e^{A b}=I+A b+\frac{A^{2} b^{2}}{2!}+\frac{A^{3} b^{3}}{3!}
$$

Writing equation 4.13, in closed form we have

$$
\ddot{\Phi}(b)=e^{A b}=\sum_{k=0}^{\infty} \frac{A^{k} b^{k}}{k!}
$$

Using equation 4.14 , equation 4.9 can be found as

$$
\Psi(b)=\frac{1}{b} \int_{0}^{b_{\infty}} \sum_{k=0} \frac{A^{k}{ }^{k}}{k!} d \sigma
$$

Changing the order for summation and integral, the integral can be found and

$$
\Psi(b)=\frac{1}{b} \sum_{k=0}^{\infty} \frac{A^{k}}{k!} \frac{b^{k+1}}{k+1}
$$

With the constant moved to the right side of the summation, equation 4.16 gives

$$
\Psi(b)=\sum_{k=0}^{\infty} \frac{A^{k_{k}} b^{k}}{(k+1)!}
$$

Equation 4.17 is important, for if we change the lower end of the summation, in equation 4.14 , we get

$$
\stackrel{*}{\Phi}(b)=I+\sum_{k=1}^{\infty} \frac{A^{k} b^{k}}{k!}
$$

where, if we let $k=j+1$

$$
\ddot{\Phi}(b)=I+\sum_{j=0}^{\infty} \frac{A^{j_{b}}{ }^{j}}{(j+1)!} b A
$$

Using equation 4.17 in equation 4.19

$$
\Phi^{*}(b)=I+\Psi(b) b A
$$

which indicates that only $\Psi$ will have to be found to solve equations 4.10, 4.11 and 4.12.

Moler and Loan (66) discuss several methods to find the exponential of a matrix and the method used here, to solve equation 4.17 , is based on a scaling and squaring technique (68). This technique is based on the simple fact that

$$
\left[e^{A \frac{T}{2}}\right]^{2}=e^{A T}
$$

If $T$ is too large we can compute the series for $T / 2^{k}$, which is not too large, and square the result $k$ times.

The value of $k$ can be selected (2) such that the size of AT divided by $2^{k}$ is less than 1 . In this case the series AT/2 ${ }^{k}$ converges. The rule is to select $k$ such that

$$
2^{k}>\|A T\|=\max _{j} \sum_{i=1}^{\eta}|A i j| T
$$

Taking the $\log$ of both sides

$$
\mathrm{k}>\log _{2}\|\mathrm{AT}\|
$$

from which we select, for

$$
\begin{align*}
& \log _{2}\|\mathrm{AT}\|<0 \longrightarrow \mathrm{~m}=0 \\
& \log _{2}\|\mathrm{AT}\|>0 \longrightarrow \mathrm{~m}=\log _{2}\|\mathrm{AT}\|
\end{align*}
$$

To square the result $k$ times we use the general formula

$$
\Psi(b) \Psi(b)=\Psi(2 b)=\left[I+\frac{T A}{2} \Psi(b)\right] \Psi(b)
$$

The truncation of the series given by equation 4.17 is done at $k=30$ and the block diagram to solve it is given in Figure 4.2.1. The program was written in FORTRAN and can be used in stand-alone mode or as a subroutine. It is called PSIMAl and its listing is given in Appendix 4.1

With the use of ISIMA1, equations $4.10,4.11$ and 4.12 can be solved by using equation 4.20. This program was checked against the test matrices given by Ward (67) and the results were good.

The program to find $\Phi^{*}, L_{1}$ and $L_{2}$, and assemble them in one of the forms given by equations 3.17 and 3.18 , according to the delay, is called STATE. It accepts matrices $A \& B$, the sampling period and the delay, as inputs, and gives $\Phi^{*}, L_{1}, L_{2}$ and the extended forms, when delay is present. The FORTRAN listing for this program is given in. Appendix 4.2 and a sample of its use can be seen in Appendix 10.2 .

### 4.3 FEEDBACK GAINS BY POLE-PLACEMENT DESIGN - ACKER PROGRAM

The pole-placement design method, as described in Chapter 3, gives the feedback gains for a specified characteristic equation.

The gains are obtained by solving equation 3.28 which is repeated here.

$$
K=\left[\begin{array}{lllll}
0 & \cdot & 0 & 1
\end{array}\right]\left[\begin{array}{llll}
\mathrm{L} & \Phi \mathrm{~L} & \Phi^{2} L & \cdots
\end{array} \Phi^{\eta^{-1}} L\right]^{-1} \cdot \alpha_{c}(\Phi) \quad 4.24
$$

Matrix $C_{0}=\left[\begin{array}{lll}\mathrm{L} & \Phi \mathrm{L} & \Phi^{2} \mathrm{~L}\end{array} \ldots \Phi^{\mathrm{n}-1} \mathrm{~L}\right]$ is the controllability matrix. As it is easier to solve a specific set of equations than to find the inverse, we define a vector $d^{*}$ so that

$$
[0 . . .0 \quad 1] C_{o}^{-1}=d^{T}
$$



Figure 4.2.1 Program to find $\Psi(b)$

Solving

$$
\mathrm{d}^{\mathrm{T}} \mathrm{C}_{\mathrm{o}}=\left[\begin{array}{lllll}
0 & \cdot & 0 & 0 & 1
\end{array}\right]
$$

we can then find $K$ by solving

$$
\mathrm{K}=\mathrm{d}^{\mathrm{T}}{\alpha_{c}}(\Phi)
$$

The block diagram for the solution of equation 4.26 is given in Figure 4.3.1. Equation 4.25 is solved by using the NAG routines FO1 BTF and F04.AYF. These routines solve a set of real linear equations with multiple right hand sides by Gaussian elimination. The FORTRAN listing for this program, which is called ACKER, is given in Appendix 4.3. A sample of its use can be seen in Appendix 10.5.

### 4.4 FEEDBACK GAINS BY OPTIMAL CONTROL DESIGN - SWEEP PROGRAM

The solution to the two-point boundary value problem, discussed in Chapter 3, is found by using the sweep method by Bryson and Ho (59). The equations to be solved to find a minimum solution are repeated here for easier reference. They are

$$
\begin{array}{ll}
\mathrm{u}^{\mathrm{T}}(\mathrm{k}) \mathrm{Q}_{2}+\lambda^{\mathrm{T}}(\mathrm{k}+1) \mathrm{L}=0 & 4.27 \\
x(\mathrm{k}+1)-\Phi x(\mathrm{k})+\mathrm{Lu}(\mathrm{k})=0 & 4,28
\end{array}
$$

and

$$
x^{T}(k) Q_{1}-\lambda^{T}(k)+\lambda^{T}(k+1) \Phi=0
$$

The initial conditions on $x(k)$ are given and the final condition is specified by

$$
\lambda(N)=Q_{1} x(N)
$$

The sweep method assumes

$$
\lambda(k)=S(k) x(k)
$$


which transforms equation 4.27 to

$$
Q_{2} u(k)=-L^{T} S(k+1) x(k+1)
$$

or

$$
\mathrm{Q}_{2} \mathrm{u}(\mathrm{k})=-\mathrm{L}^{\mathrm{T}} \mathrm{~S}(\mathrm{k}+1)[\Phi x(\mathrm{k})+\mathrm{Lu}(\mathrm{k})]
$$

Solving for $u(k)$ we have

$$
u(k)=-\left[Q_{2}+L^{T} S(k+1) L\right]^{-1} L^{T} S(k+1) \Phi x(k)
$$

where for shorter representation we call

$$
R=\left[Q_{2}+L^{T} S(k+1) L\right]
$$

Using equation 4.31 we can eliminate $\lambda$ from equation 4.29 to get

$$
S(k) x(k)=\Phi^{T} S(k+1) x(k+1)+Q_{1} x(k)
$$

where using the system equations we have

$$
S(k) x(k)=\Phi^{T} S(k+1)[\phi x(k)+L u(k)]+Q_{1} x(k)
$$

Substituting equation 4.32 into equation, 4.33 and collecting all terms on one side

$$
\left[S(k)-\Phi^{T} S(k+1) \Phi+\Phi^{T} S(k+1) L^{-1} L^{T} S(k+1) \Phi-Q_{l}\right] x(k)=0 \quad 4.34
$$

Equation 4.34 must hold for all values of $x(k)$ therefore it follows a backward equation in $S(k)$.

$$
S(k)=\Phi^{T}\left[S(k+1)-S(k+1) L R^{-1} L^{T} S(k+1)\right] \Phi+Q_{1}
$$

which is normally. rewritten as

$$
S(k)=\Phi^{T} M(k+1) \Phi+Q_{1}
$$

where

$$
M(k+1)=S(k+1)-S(k+1) L R^{-1} L^{T} S(k+1)
$$

$$
4.37
$$

Equation 4.36 must be solved backwards with the initial condition given by equations 4.30 and 4.31 , as

$$
S(N)=Q_{1}
$$

To solve for $u(k)$, we use 4.32 to obtain

$$
u(k)=-K(k) x(k)
$$

where the optimal feedback gains are given by

$$
\mathrm{K}(\mathrm{k})=\left[\mathrm{Q}_{2}+\mathrm{L}^{\mathrm{T}} \mathrm{~S}(\mathrm{k}+1) \mathrm{L}\right]^{-1} \mathrm{~L}^{\mathrm{T}} \mathrm{~S}(\mathrm{k}+1) \Phi
$$

Equation 4.40 is time varying but we will be using only the steadystate solution.

The implementation of the solution requires only matrices manipulation and is given in the block diagram of Figure 4.4.1, which FORTRAN solution is shown in Appendix 4.4. The subroutines for matrices manipulation are given in Appendix 4.5. A sample of a SWEEP program run is shown in Appendix 10.4 .

### 4.5 CLOSED-LOOP SIMULATION - TIMELOCUS PROGRAM

The closed-loop state equations must be found before any simulation program is written.

The open-loop state equations for system and observer, as described in Chapter 3, can be written from equations 3.43 and 3.55 as
$x(k+1)=\left[\begin{array}{c}x_{a}(k+1) \\ x_{b}(k+1) \\ w(k+1)\end{array}\right]=\left[\begin{array}{ccc}\Phi_{a a} & \Phi_{a b} & 0 \\ \Phi_{b a} & \Phi_{b b} & 0 \\ \Omega & 0 & \theta\end{array}\right]\left[\begin{array}{c}x_{a}(k) \\ x_{b}(k) \\ w(k)\end{array}\right]+\left[\begin{array}{c}L_{a} \\ L_{b} \\ M\end{array}\right] u(k) \quad 4.41$

where

$$
\begin{aligned}
& \Omega=\Phi_{b a}-L O \Phi_{a a}+\left(\Phi_{b b}-L O \Phi_{a b}\right) L O \\
& \theta=\Phi_{b b}-L O \Phi_{a b}
\end{aligned}
$$

and

$$
M=L_{b}-L_{a}
$$

According to this expanded state, the observer estimated state-variables equation, given by equation 3.56 , is written as

$$
\hat{x}_{b}(k)=\left[\begin{array}{lll}
L 0 & 0 & I
\end{array}\right]\left[\begin{array}{c}
x_{a}(k) \\
x_{b}(k) \\
w(k)
\end{array}\right]
$$

where we define matrix LOBLW as

$$
\text { LOBLW }=\left[\begin{array}{lll}
\text { LO } & 0 & I
\end{array}\right]
$$

The control law, as given by equation 3.19 , cannot be written, directly, as a function of the state vector, $x$, given by equation 4.41. It is a function of estimated $\hat{x}_{b}$, which is not part of that state-vector。 Introducing a reference input on the control law it is written, in terms of available and estimated variables, as

$$
u(k)=K_{r} r(k)-\left[K_{a} x_{a}(k)+K_{b} \hat{x}_{b}(k)\right]
$$

where $K_{i}$ and $K_{b}$ come from the original control law

$$
u_{0}(k)=-K x_{0}(k)
$$

'with

$$
u_{0}(k)=\left[\begin{array}{ll}
K_{a} & -K_{b}
\end{array}\right]\left[\begin{array}{c}
x_{a}(k) \\
x_{b}(k)
\end{array}\right]
$$

in order to make the software representation easier we decided to use the complete state-vector $x_{\text {, }}$ instead of just $x_{a}$, in equation 4.45. The result is

$$
u(k)=K_{r} r(k)-\left[K M_{1} x(k)+K M_{2} \hat{x}_{b}(k)\right]
$$

where

$$
K M_{1} x(k)=\left[\begin{array}{lll}
K_{a} & 0 & 0
\end{array}\right]\left[\begin{array}{c}
x_{a}(k) \\
x_{b}(k) \\
w(k)
\end{array}\right] \quad K M_{2}=K_{b}
$$

Using equations 4.43 and 4.44 in equation 4.47 we find

$$
\mathrm{u}(\mathrm{k})=\mathrm{K}_{\mathrm{r}} \mathrm{r}(\mathrm{k})-\left[\mathrm{KM}_{1} x(\mathrm{k})+\mathrm{KM}_{2} \operatorname{LOBLW} x(\mathrm{k})\right]
$$

Equation 4.49 is the final form for the control function with reference input.

Writing equations 4.41 in the general form

$$
x(k+1)=\phi x(k)+\operatorname{Lu}(k)
$$

the state-equations for the closed-loop system are found, by using 4.49, as

$$
x(k+1)=\left[\Phi-\mathrm{LKM}_{1}-\mathrm{LKM}_{2} \text { LOBLW }\right] x(k)+\mathrm{LK}_{r} r(\mathrm{k})
$$

The time-response, for the closed-loop system, is found by solving equation 4.51 , which is a simple recurrence formula. Initial conditions can be introduced but our simulation program assumes all initial conditions to be zero.

Taking the $z$-transform of 4.51 we have

$$
\left[\mathrm{zI}-\left(\Phi-\mathrm{LKM}_{1}-\mathrm{LKM}_{2} \mathrm{LOBLW}\right)\right] x(z)=\mathrm{LK}_{\mathrm{r}} \mathrm{r}(\mathrm{z})
$$

Therefore, the closed-loop characteristic equation is given by

$$
\operatorname{det}\left[z I-\Phi_{E}\right]=0
$$

where

$$
\Phi_{E}=\Phi-L K M_{1}-L K M_{2} L O B L W \quad 4.54
$$

Equation 4.53 indicates the characteristic polynomial of $\Phi_{E}$ and the roots of this polynomial are the eigenvalues of ${ }^{\Phi_{E}}$.

The root-locus for the closed-loop system is implemented by finding the eigen values of matrix $\Phi_{E}$ through a suitable algorithm. In this work we use the NAG routine FO2BCF which calculates selected eigenvalues of a real unsymmetric matrix by reduction to Hessenberg form, the $Q R$ algorithm and inverse iteration.

The simulation program includes the time-response and root-locus techniques and is based on the general diagram shown in Figure 4.5.1. The FORTRAN implementation is given in Appendix 4.7.


Figure 4.5.1 Simulation by TIMELOCUS Program


Figure 4.5.1 (continued)

## CHAPTER 5

### 5.1 INTRODUCTION

The use of advanced modes of control imposes severe restrictions on the utilisation of a single microcomputer. Multi-microcomputers offer a relatively inexpensive and very flexible way to implement such control modes (91, 92).

This chapter is concerned with the analysis and choice of a multimicrocomputer structure to be implemented in this work.

The definition of multi-microcomputer systems with its advantages and disadvantages is discussed in Section 5.2. General structures for multi-microcomputer implementation, with particular emphasis on real-time control applications, are discussed in Section 5.3. Modularity, as a major characteristic of multi-microcomputer structures, has been analysed in this section.

The selection of a multi-microcomputer structure, to be implemented in this work, is described in Section 5.4. Industrial experience shows that some structures are more favourable for real-time control than others. The central memory and global bus structure are suitable for such applications and a mixture of both offers greater flexibility (5.4.4). The use of a common bus causes contention and arbitration is necessary (5.4.5). Selection of a common bus, to be implemented in this work, is explained at the end of Section 5.4.

### 5.2 MULTI-MICROCOMPUTER SYSTEMS

A multi-micro is a system of two or more microcomputers connected either through shared memory or via high or low speed data links. The shared memory may be a multi-ported main memory, cache memory or a multi-ported disc.

The data path may be either a bit-serial or parallel bus connecting I/O ports of two microcomputers or a shared bus to which two or more computers are interconnected in various ways. Data may be broadcasted onto the bus and intercepted by the receiving microcomputer or, where the communication link is a daisy chain, each connecting microcomputer takes the message off the link. The advantages of
each of these interconnect schemes depend on the application.
Multi-microcomputer systems that employ the shared-memory interconnect approach have been named "tightly coupled". It means that all the processors in the system can reach all the memories and execute code out of them (69); $1 / O$ and other system resources are shared by the processors and the interprocessor latency is low due to the access time being limited only by the actual memory access time. "Tightly coupled" and "loosely' coupled" systems differ in that a loosely coupled system has separate primary or main memory address space. This means that, at the hardware level, there has to be an explicit communications interface between the microcomputers. This communications interface implies that there is a higher latency of communications between processors than would be if they shared primary memory.

Relating to timing, tightly coupled systems generally require synchronization between processes while in loosely coupled systems concurrent processes may be performed asynchronously.

Changes in architecture are usually not easily achieved in tightly coupled systems. Figure 5.2 .1 shows the structure for both tightly and loosely coupled systems. In the latter the microcomputer stands by itself with its own main memory. Variations on the structures shown in Figure 5.2.1 are common and some of them will be discussed later on.

The use of multi-microcomputer systems is adequate for many applications for several reasons. They usually make it easier for the user to access the system, provide increased performance through resource sharing and often increase the availability of a system. A network of microcomputers can quite often duplicate the capability of one large expensive system at lower total cost. Multi-microcomputer systems can provide adaptability and rapid reconfiguration with the system working at different times as a very large and complex problem solver or as a network of smaller machines, each dedicated to a unique task, or as something in between. They can usually provide increased reliability since the total system can continue to operate despite individual microcomputer failure, with limited capability; if some of the communication links remain intact. Redundancy can be achieved at lower cost for each micrcomputer can be replaced by others in a homogeneous system.


Tightly Coupled System


Loosely Coupled System

Figure 5.2.1 Loosely and Tightly Coupled Systems

Distributed multi-microcomputer systems can provide increased distributed processing power and responsiveness because it can be closely tailored to the application. A properly designed distributed microcomputer system threatened by overload can be expanded by simply adding more microcomputers.

The disadvantages and advantages of using multi-microcomputers can only be assessed according to the application. On the disadvantage side the designer may be faced with increased software complexity, and the software for distributed systems may be more costly to develop than for centralised systems. In contrast to a single central processor based system with only one executive, a distributed system normally requires each microcomputer to contain its own individual executive that must be capable of communicating with all the other executives in the total system. This will also require that each individual executive provides a task handling capability where tasks resident in various processors can communicate with each other and, in case of local software errors, diagnostic capability exists to find the cause.

Finally, the design and development of a multi-microcomputer system may require unique expertise both in hardware and software areas. The advantages and disadvantages of multi-microcomputer systems are summarised in Table 5.1 (70).

| MULTI-MICROCOMPUTER SYSTEMS |  |
| :--- | :--- |
| Advantages | Disadvantages |
| Increased: | Increased software com- <br> Reliability <br> Processing Power <br> Responsiveness <br> Modularity |
| Expandability in |  |
| smaller incremed difficulty for |  |
| system test and diagnosis |  |$\quad$| More dependence on commu- |
| :--- |
| nications technology |
| Snique expertise needed |
| during design and |
| development phase |,

Table 5.1

### 5.3.1 General Structures

Any multi-microcomputer structure is the result of a series of decisions and the decision space can be considered to be a tree, as proposed by Anderson and Jensen (71). The decision tree is shown in Figure 5.3.1 and in its identification there are three major hardware elements involving the transfer of information. They are the microcomputer itself, the path and the switching element. A path is the medium by which a message is transferred between the other system elements. Some examples of paths are wires or buses, radio links and memories. The transmission of a message over a path results in no alteration of the message. The switching element is an entity which affects the destination of message. This can be done by changing the destination address, by routing the message to one of a number of alternative paths or by both actions.

The decision tree has got four levels and alternative system architectures represented as leaves. The root of the tree is the decision to interconnect a number of microcomputers for complete intercommuncation. Below this are decision levels representing choice of message transfer strategy, the method of controlling transfers and choice of the type of path over which transfer is to be made. The first two levels areconcerned with policy and the third and fourth with implementation.

The first choice is between direct transmission of messages from source to destination and indirect transmission in which an intervening operation is required. Intervening repeater or storage elements are simply instances of paths and do not affect the directness of the communication; an intervenor that alters the message, by any means mentioned before, is effecting an indirect communcation. Another way to make this distinction is to determine whether control information is contained or sent to the intervenor.

When indirect communication is chosen a further decision concerning the switching method must be made. The alternatives are centralisation and decentralisation in which a single or a number of intervenors are used to switch the messages.

|  | Transfer <br> Strategy | Interconnection for Communication |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Direct |  |  |  | Indirect |  |  |  |  |  |
|  | Transfer <br> Control <br> Method | (none) |  |  |  | Centralised <br> Routing |  |  | Decentralised <br> Routing |  |  |
|  | Transfer <br> Path <br> Structure | Dedicated Path |  | Shared <br> Path |  | Dedicated <br> Path |  | Shared <br> Path | Dedicated Path |  | Shared <br> Path |
|  | System <br> Archi- <br> tecture | Loop |  | Central memory | Global <br> bus | Star | Loop <br> with central switch | Bus <br> with central <br> switch | $\left\lvert\, \begin{gathered} \text { Regular } \\ \text { net- } \\ \text { work } \end{gathered}\right.$ |  | Bus |

Figure 5.3.1 Multi-Microcomputer Structures

The third level, in the decision process, involves the choice of dedicated or shared transfer paths. The shared path is defined as one which is accessible from more than two points. Contention does not occur in unidirectional dedicated paths but can happen in bidirectional dedicated paths and is a major consideration in bidirectional shared paths. Redundant paths, for fault tolerance or bandwidth reasons, are considered logically singular.

The final level of decision is concerned with specific system designs and before any choice can be taken the different characteristics must be evaluated. As emphasized by Anderson and Jensen, quantitative characteristics of systems, such as bandwidths and throughputs, are only representative of rapidly changing technology (1) and must be evaluated for each particular application.

### 5.3.2 Modularity

Modularity, the ability to make incremental changes in system capability, is a major characteristic and the only one to be discussed in this work. In instances where a specific design is to be configured for a variety of applications, it is often desirable to vary the number of processors according to the computational requirements of the particular problem. One measure of system modularity is the incremental cost of adding an element, such as a processor. If this cost is simply that of the element then the system is indeed modular; but if the addition of the nth processor requires the addition of $n-1$ interconnection paths, then the system is not so modular. At the third level of the tree, some decisions involving this cost modularity measure have already been made. For instance, selection between direct and indirect paths involves trade offs between the poorer cost modularity of dedicated paths and the vulnerability of shared paths to bottlenecking.

Another measure of modularity is the degree to which the location and function of the incremental element is restricted. For instance, in a given design there may be particular places where a resource (microcomputer, switch or path) could be easily added
to produce a specific performance increase, and other types of performance increase which are difficult or impossible to obtain in a modular fashion. This place modularity characteristic of indirect centralised systems is poor with respect to the central switch. Replication of the central switch to achieve an increase in throughput changes the basic architecture to indirect decentralised. A place restriction can also occur in any nonhomogeneous indirect architecture, since a special purpose processor which must be added to the system usually cannot occupy a place that must perform a switching function.

### 5.4 MULTI-MICROCOMPUTER STRUCTURE SELECTION

5.4.1 Structures for Real-Time Control

The industrial experience, as shown by some of the references in Chapter 2, indicates that requirements imposed by the realtime control of processes, restrict the decision in the choice of a multi-microcomputer structure to just one level on the tree structure. The transfers need not be indirect, which eliminates the necessary transfer control, and dedicated paths are difficult to justify economically,

With these assumptions we are left, according to Figure 5.3.1, with just two system structures. They are the global bus and the central memory types.

### 5.4.2 Central Memory Structure

The central memory structure, shown in Figure 5.4 .1 , is the most common way to interconnect computer systems. In this structure two or more processors communicate by leaving messages for one another in a commonly accessible memory. The key characteristic to this structure is that the memory is, or can be, used as a path rather than only as storage.

The place modularity of the central memory structure is very good as it is possible to add processors arbitrarily, since they are not distinguished by the structure. It is also possible to increase capacity.for message storage by simply increasing the size of the memory.

The cost modularity of central memory systems depends almost exclusively on the path structure by which the processors access the memory. If each processor is provided with a direct path, then cost modularity can be poor since an incremental processor can possibly bring the total number to be bigger than the number of available memory ports. If the memory is accessed by a single bus with a suitable allocation mechanism, cost modularity can be very good. The common memory systems are quite vulnerable to a bottleneck in which the memory's bandwidth becomes a restriction on communication rates. Cost modularity in this case is poorer as it is expensive to increase bandwidth of the memory or the access path.

Logical complexity in common memory systems is low and the failure reconfiguration characteristic is good in case of processor failure but poor in case of failure of the central memory unit.

### 5.4.3 Global. (Common) Bus Structure

The global bus structure, shown in Figure 5.4.2, is formed by a number of processing elements interconnected by a common, or global bus. Access to this bus is shared among the processors by some allocation scheme, and messages are sent directly from the source, through the bus, to be recognised and accepted by the proper destination or destinations.

Both the cost and the place modularity of global bus structures are good with respect to the processing elements. Depending on the choice of bus allocation scheme it is possible to add a processor to the system in any position with little or no effect on the other processing elements. The cost and place modularity of the communications path are poor as it is not possible to increase the bandwidth easily and performance cannot be improved only when needed. To increase performance it is necessary to change the implementation of the entire bus or to replicate it, alternatives which have a great impact on the bus interfaces of all the processing elements. The failure-reconfiguration characteristics for global bus systems are very good with respect to the bus. Failures of the bus are catastrophic and


Figure 5.4.1 Central Memory Structure


Figure 5.4.2 Global Bus Structure
replication is required if the structure is to be maintained. The global bus is of course a potential bandwidth bottleneck.

### 5.4.4 Modified Common Bus Structure

For real-time control applications it is not necessary for the processors, in a multiprocessor structure, to share a central and unique memory. The possibility to describe the system in separate tasks which communicate not very frequently and only transfer data between them, allows a mixture of both structures described before. This mixture has got the advantages of both structures and avoids the bottleneck imposed by a central and unique memory on real-time control systems.

The use of asynchronous input/output global bus, as discussed by Popa(43), has proved to be too low, to preserve the expandability, for requiring real-time control. The use of higher speed bus structures is necessary and the global bus and common memory structure, shown in Figure 5.4.3, is normally used. This structure has not been successful for systems with more than 4 computers, as suggested by Popa, and its modularity is poor compared with that of a global bus structure as shown in Figure 5.4.2. Now, there is a new module in the system which is different from the computer modules: thememory module. The interfaces to the bus are different for the computer and memory modules.

A simple modification in the structure, shown in Figure 5.4.4, allows an almost return to the original global bus structure in terms of modules hardware but maintaining the common memory facility. The idea is to split the common memory block, shown in Figure 5.4.3, into small blocks which are now mapped on the address space of each computer (microcomputer). This structure is shown in Figure 5.4.4. Communications, according to this modified structure, can now take place by direct memory access and profit from it.

The structure shown in Figure 5.4 .4 is the one used in this work for the multi-microcomputer. It is slightly modified,


Figure 5.4.3 Global Bus and Common Memory Structure


Figure 5.4.4 Globaĺ Bus and Distributed Common Memory
as discussed and shown in Chapter 6, as the common memory block, in each microcomputer, is just a fraction of its address space size.

### 5.4.5 Common-Bus Arbitration

The use of a global bus, from now on called common bus, to interconnect several microcomputers, generates contention and some sort of arbitration must exist.

The bus arbitration is the way by which a particular microcomputer can request and obtain control of the bus and then transmit data over it. In the arbitration process a bus request is generated by the microcomputer which wants to use the bus and it is received by the bus arbitrator. If conditions allow, the bus is granted to the microcomputer and, when communication over the bus has finished, the arbitrator takes over the bus again.

The bus arbitrator must resolve the conflict between pending requests in order to allow just one microcomputer to use the bus at a given time. Popa (43) discussed arbitration schemes in detail and here for design purposes we only mention the salient aspects of centralised arbitration.

Centralised arbitration is usually associated to a processor which plays a central role in the system. It can be implemented with inexpensive hardware and have high speed but its position-modularity is poor and some implementations require a considerable number of lines.

For real-time control applications, the multi-microcomputer will be tailored to the control task and a priority of bus usage quite certainly is known at the design stage. This allows us to exclude from this discussion such centralised schemes as polling and static arbitration, as discussed by Popa.

The independent requests arbitration scheme as shown in Figure 5.4.6 is quite attractive and was seriously considered to be implemented in this work. Its main advantage of course is the flexibility it gives to have priorities modified while the system is running. The only disadvantage, we see in this implementation, is the number


Figure 5.4.5 Centralised Daisy Chain Arbitration


Figure 5.4.6 Independent Requests Arbitration
of necessary lines. The arbitration is not a problem as it can easily be solved by prioritized interrupts.

At the end, we decided to implement the simpler solution given by a centralised daisy chain arbitration, as in Figure 5.4.5 for we only have two microcomputers in our actual system and, with this number of microcomputers, either of the schemes have basically the same configuration. The advantages of a daisy chain link are first, the simplicity of implementation and second, the possibility to have several daisy chains, each with its own assigned priority. The main disadvantage is that the priorities are rigidly fixed to the position of a microcomputer on the chain. Highest priority is associated to the microcomputer nearest to the arbitrator. Implementation of such arbitration scheme can be seen in Chapter 6, where it is called interrupt controlled arbitration link.

### 5.4.6 Common-Bus Selection

The multi-microcomputer structure chosen to be implemented in this work, as shown in Figure 5.4.4, has a common bus (global bus) which interconnects all the microcomputers in the system. The structure assumes that transfer of data is achieved by direct memory access. Therefore, the bus must contain data, address and control lines (parallel bus), even if multiplexed (72). The implementation of this parallel bus was based on simplicity of interconnection of any microcomputer to the bus. At this point the design had to move to a device dependent direction. The only microprocessor supported by the facilities available is the Texas TMS 9900, which is a 16 -bit microprocessor. For direct memory access we must therefore have 16 data lines or a multiplexed data bus. Any multiplexing requires extra hardware, for the TMS 9900 has not any multiplexing facility. This extra hardware would add to the hardware necessary to interface each microcomputer to the common bus and we decided that multiplexing was out of the question. This eliminated from considerations the standard IEEE-488 bus $(73,77)$.

The buses available commercially, when this work started, were supplied by other makers of microprocessors and, even if we
could afford to buy any of them the interface to the Texas TMS 9900 microprocessor would still be necessary.

The only alternative left to be considered was a proposed IEEE standard which could cope with 16 -bit microcomputers. This proposed standard specifies the $\mathrm{S}-100$ bus (75). This is a parallel bus with 100 lines to be implemented in a backplane. Among these lines provision is made for 16 data lines and 16 or 24 address lines. The communication between microcomputers takes place on a master-slave configuration and there is a permanent master of the bus. The permanent master transfers control of the bus to any microcomputer wishing to use it, and this microcomputer is then the temporary master. The bus is completely specified including the interconnection board mechanical parameters. The argument to go for a standard bus implementation was strong but we did not need such a complex bus as the $S-100$ and the argument against it, in our case, is the same raised when we discarded the commercially available buses: the interfaces for the TMS 9900 would still be necessary because they were not available when this work started and they were still not available when we finished.

The common bus implemented in this work is based on the Texas TMS 9900 microprocessor data, address and control signal lines and its complete implementation is given in Chapters 6 and 7. At the end of this work, Texas made available the E-Bus (90), which is a multiplexed bus. It was basically designed to be used by 8-bit microprocessors and allows the use of the new generation of 16 -bit multiplexed devices. Even at the real end of this work, no interfaces were provided by the manufacturer unless for the 8-bit devices.

## CHAPTER 6

## DESIGN OF A MODULAR MULTI-MICROCOMPUTER

### 6.1 INTRODUCTION

The design of a modular multi-microcomputer, based on the considerations made in previous chapters, is described. The basic element in the multi-microcomputer structure is the microcomputer itself. The choice of such element is discussed in Section 6.2. There we can see the microprocessor's dictating rules and the limitations sometimes imposed by non-technical factors.

A general mapped communication structure, used in this work, is discussed in Section 6.3. The necessary amount of communication area is discussed and the number of supported modules is chosen. The use of a common communication bus is bound to have contentious problems so a controller module is used.

All the logic involved in the communication process is discussed in Section 6.4. A priority link between microcomputers is used and the master-slave communication approach is divided in various phases.

In a modular design, flexibility must be first. Section 6.5 shows how this was achieved and the resulting logic diagrams.

The logic design at integrated circuit level is not easily understood, so Section 6.6 discusses the functions of the designed logic.

To complete the design, Section 6.7 shows the various timing relationship diagrams within the system.

### 6.2 THE MICROCOMPUTER

### 6.2.1 The Microprocessor

The choice of a microprocessor for a specific application is not always a matter of technical specifications. At the time this work started, the Texas 9900 Microprocessor was not in the top of the 16 -bit microprocessor specifications, but its availability, its support family chips, its technical back up, and its powerful set of instructions made it a reasonable choice. The availability in the Department of a Texas Full Development System (1) and the experience acquired in using the Texas TMS 9900 Microprocessor, within the Control Laboratory since 1978 , made its choice not only
reasonable, but the only one possible.

The TMS 9900 Microprocessor is a single chip 16 -bit CPU produced using $N$-channel silicon gate MOS technology. It has a l5-bit wide address bus which of course can access directly 32268 words ( 16 bits) of memory. The data bus is 16 -bits wide and the instruction set enables both word and byte ( 8 bits) operands.

The instruction set for the TMS 9900 is compatible with the full minicomputer Texas TM 990 family, including multiply and divide (unsigned). The microprocessor's normal working frequency is 3 MHz which gives a clock cycle of about $0.33 \mu s$ and an average instruction execution time of $5 \mu \mathrm{~s}$ (76).

### 6.2.2 Microcomputer Selection

The basic element in multi-microcomputer system is clearly the microcomputer itself. By microcomputer it is meant a small computer system where the CPU is a microprocessor.

The availability from Texas Instruments of well tested and reasonably priced single board microcomputers made us decide to go for the off-the-shelf option. A very strong point favouring this decision was that, based on some other developments in the Department, it would take at least six months to develop a microcomputer from scratch and the final product would still have to prove itself in terms of reliability, so we decided to buy off-the-shelf microcomputer boards.

There were two 16 -bit single board microcomputers offered by Texas at the time this work started. They were the TM 990/100M and TM 990/101M single board microcomputers. The specifications for each one are given in Table 6.2.1. It is seen there that the basic differences are: memory available on board, number of $1 / 0$ interface chips ( $/$ /O ports) and the possibility of direct memory access to the memory on the board on the TM 990/101M microcomputer.

For the multi-microcomputer structure desired, any of the boards would give the minimum required characteristics and, of course, more facilities available, more flexibility. It is good to remember that these boards are stand alone microcomputers and can be used elsewhere when not used on the multi-microcomputer structure. So, the best choice seemed to be the TM 990/101M at this stage. The next stage was to compare prices. The TM 990/101M was £617.00
and the TM 990/100M £443.00 excluding VAT. The minimum configuration for the multi-microcomputer would require 2 microcomputer boards plus power supply and racking system. The amount of money allocated to this project was $£ 1200.00$ in total, so that the TM 990/101M microcomputer could not be chosen anyway.

It is worth mentioning here that, sometimes, what are considered to be minor things can cost a lot when on limited budget. For instance, $3 \%$ of the total budget was spent on 5 edge connectors. The power supply and racking system cost another $14 \%$.

|  |  | MICROCOMPUTER |  |
| :---: | :---: | :---: | :---: |
|  |  | TM990/100M | TM990/101M |
|  | CPU | TMS990 Microprocessor | TMS990 Microprocessor |
|  | Interrupts | 16 levels - 15 external | 16 levels - 15 external |
|  | Interval Timers | two | three |
|  | EPROM (board) | up to 4 K | up to 4 K |
|  | RAM (board) | up to $1 / 2 \mathrm{~K}$ | up to 2 K |
|  | off board | up to 32 K | up to 32 K |
|  | Paralle 1 | 16 bits - up to 4 K | 16 bits - up to 4K |
|  | Serial | TMS9902-Asynchronous <br> TMS9903-Synchronous | TMS9902-Asynchronous <br> TMS9903-Synchronous |
|  | Bus | TTL compatible | TTL compatible |
|  | Paralle1 | TTL compatible | TTL compatible |
|  | Serial | RS232C or 20 mA current loop | Port A - RS 232 C or 20 mA current loop <br> Port B - RS 232 C mode m |
|  | Software | Tibug Monitor in EPROM | Tibug Monitor in EPROM |
|  | Power <br> Requirements <br> $\pm 3 \%$ | $\begin{array}{r} 5 \mathrm{~V}-1.3 \mathrm{~A} \\ 12 \mathrm{~V}-0.2 \mathrm{~A} \\ -12 \mathrm{~V}-0.1 \mathrm{~A} \end{array}$ | $\begin{aligned} 5 \mathrm{~V} & -1.6 \mathrm{~A} \\ 12 \mathrm{~V} & -0.2 \mathrm{~A} \\ -12 \mathrm{~V} & -0.2 \mathrm{~A} \end{aligned}$ |

Table 6.2.1

### 6.2.3 The Texas TM 990/100M-1 Microcomputer

### 6.2.3.1 The Microcomputer Board

The Texas TM 990/100M-1 is a microcomputer module, assembled in a single printed circuit board, which CPU is the Texas TMS-9900 microprocessor. The basic hardware configuration of the TM 990/ l00M microcomputer family, to which the TM 990/100M-1 belongs, is

1 TMS 9900 Microprocessor
1 Clock TIM 9904
256 words of 16 bits of RAM memory expandable on board to 512 words 1 K words of 16 bits of EPROM expandable on board to 2 K words norma1ly and to 4 K words using the jumper-selectable TMS 2716

1 TMS 9901 Programmable Systems Interface
1 TMS 9902 Asynchronous Communications Controller
These elements and their relationship can be seen in Figure 6.2.1 where P1, P2, P3 and P4 are connectors accessible on the board.

The TM 990/100M-1 microcomputer comes with its EPROM memories already expanded to 2 K words where it is pre-programmed the TIBUG monitor software, which will have its features commented on later. The TMS 9900 microprocessor, which pin assignment is given in Appendix 6.1, has a memory word 16 bits long and the instruction set allows both word and byte ( 8 bits) operands. Thus, all memory locations are on even address boundaries and byte instructions can address either the even or odd byte. The memory space is 65536 bytes or 32768 words. The basic configuration for the TM 990/100M-1 has the memory address map shown in Figure 6.2.2.

The communications register unit (CRU) is the normal input/output data interface for the TM 990/100-1 microcomputer. When CRU instructions are executed data is written or read through the CRUOUT or CRUIN pins of the TMS 9900 microprocessor. The devices from which data is read or written to are addressed via the address bus of the microprocessor. The CRU address is maintained in register 12 of the work space register area. Only bits 3 through 14 of register 12 are interpreted by the CPU for the desired CRU address and this 12 bit value is called the CRU base address. This value multiplied by 2 is called the base address and it is the value normally loaded in register 12 to generate the CRU base address desired.


Figure 6.2.1 TM 990/100M Microcomputer Block Diagram
(2000

Figure 6.2.2 TM 990/100M-1 Memory Address Map

The devices accessed by the microprocessor CRU lines in the microcomputer module board are the TMS 9901 parallel interface and the TMS 9902 serial interface which have the base and CRU base addresses as given in Table 6.2.2 in hexadecimal representation.

|  | BASE ADDRESS <br> R12 (0-15) | CRU BASE ADDRESS <br> R12 (3-14) |
| :--- | :---: | :---: |
| TMS 9902 | 0080 | 0040 |
| TMS 9901 | 0100 | 0080 |

Table 6.2.2

The TMS 9902 acts as the controller for the asynchronous serial interface. The character length can be 5 to 8 bits and the Baud rate 75 to 38400 bits per second. Through this chip, two interfaces are supported: ELA and 20 mA neutral current loop TTY. The microcomputer board is delivered with a 25 pin RS 232 type female connector and is jumper selectable to support EIA or TTY operation.

The TMS 9901 handles the paralle1 I/O operation including all 15 external interrupts. It provides 22 lines which can be programmed as input/output or interrupt lines. A complete description of TMS 9901 is given later on in this chapter.

From Figure 6.2.2 it is seen that the memory area where the interrupt context switch vectors are stored is an EPROM area. This disables one to change the contents of these vectors through software which would give more flexibility when using this module as a prototype. In order to overcome this difficulty, Texas delivers the microcomputer with some values already written in the EPROM interrupt vector locations. The vectors for interrupt 0 are used as the reset function which places the microcomputer under the TIBUG monitor control. The vectors for interrupt 3 and 4 are given in Table 6.2.3, which shows that the memory contents indicate addresses in the RAM area of memory.

|  | MEMORY ADDRESS | MEMORY CONTENTS |
| :---: | :---: | :---: |
| INTERRUPT <br> 3 | 000 C | FF68 |
|  | 000E | FF88 |
|  | 0010 | FF86 |
|  | 0012 | FFAC |

Table 6.2.3

So, as the microcomputer is delivered, only interrupts 3 and 4 can be used. In order to achieve that, the user normally will have to use a general branch instruction written in the location indicated by the program counter of the desired interrupt. This branch instruction (or jump) will transfer the execution to the desired area of memory where the user program is written.

If more than two interrupts are necessary the user must be able, through a programmer module, to write (EPROM - writing operation) values in the interrupt vector locations of other interrupts and then use the same indirect jump procedure as described above or a direct jump to the interrupt routine. This is achieved by writing the address of the routine in the program counter vector of the interrupt.

Both TMS 9901 and TMS 9902 provide interval timers. The time resolution for TMS 9901 is $21.3 \mu \mathrm{~s}$ with a maximum interval of 349 ms and the interval end generates interrupt 3. For the TMS 9902 the resolution is $64 \mu \mathrm{~s}$ with maximum interval of 16.4 ms and generation of interrupt 4 through the TMS 9901. The use of TMS 9901 time interval is shown later on.

There are five signal lines on the Texas TM 990/100M microcomputer which control the exchange of data between the $C P U$ and any external hardware in direct memory access mode. We are not concerned here with detection of illegal operation codes.

The first set of lines set the direction of data exchange. DBIN (data bus IN) when active (HIGH) indicates that the TMS 9900 microprocessor has disabled its output buffers to allow memory to place memory read data on the data bus during $\overline{\text { MEMEN }}$. DBIN remains LOW in all other cases except when HOLDA is active. $\overline{\text { MEMEN }}$ (memory enable) when active (LOW), indicates that the address bus contains memory address. $\overline{W E}$ (write enable) when active (LOW) indicates that
memory write data is available from the TMS 9900.
The second set of lines cater for delayed communications. READY when active (HIGH) indicates that memory will be ready to read or write during the next clock cycle. When not ready is indicated during a memory operation the TMS 9900 microprocessor enters a wait state and suspends internal operation until the memory indicates that it is ready. WAIT when active (HIGH) indicates that the microprocessor has entered a wait state because of a not ready condition from memory. The timing relationship between these lines is shown in Appendix 6.2.1.

For direct memory access in block and cycle stealing modes the hold state is normally used. Associated with it are 2 lines. $\overline{H O L D}$ when active (LOW) indicates to the processor that an external controller wishes to utilise the address and data but to transfer data to or from memory. The microprocessor enters the hold state following a hold signal when it has completed its present memory cycle. The processor then places the address and data bus in the high impedance state (along with $\overline{W E}, \overline{M E M E N}$ and DBIN) and sets the HOLDA signal to acknowledge the hold state. When HOLD is removed the processor returns to normal operation. HOLDA (hold acknowledge) when active (HIGH) indicates that the processor is in the hold state. Timing for these lines is shown in Appendix 6.2.2.

Looking back to Appendix 6.1 we can see that, as soon as the hold state is acknowledged by the microprocessor, that is HOLDA is set HIGH, two things happen: the control lines are disabled, going to a state dependable on the output connections to the open collector gates, and the address and data lines are disabled going to a high impedance state (three-state output of IC 74LS 243N). From this observation it is obvious that no direct memory access is possible to memory inside the TM 990/100M microcomputer board.

### 6.2.3.2 TIBUG Software Interface

The TIBUG is a debug monitor which provides an interactive interface between the user and the microcomputer. It comes on two 2708 EPROMs and occupies memory space from memory address 0080. The software utilises four workspaces in 40 words of RAM memory where also are the restart vectors which initialise the monitor following single step execution of instructions. All communication with

TIBUG is through a 20 mA current loop or RS 232 device. When the reset push button is pressed the TIBUG monitor is initialised through interrupt 0 . In order to set the Baud rate for the terminal device used, the character "A" must be entered after the reset. TIBUG uses this input to measure the width of the start bit and set the TMS 9902 to the correct Baud rate. The acceptable Baud rates are: 110, 300, 1200 and 2400 Baud.

The TIBUG monitor commands are given in Table 6.2 .4 and their descriptions can be seen in Reference 77.

| COMMAND | RESULT |
| :--- | :--- |
| B | Execute Under Breakpoint |
| C | CRU Inspect/Change |
| D | Dump Memory to Cassette/Paper Tape |
| E | Execute |
| F | Find Word/Byte in Memory |
| H | Hex Arithmetic |
| L | Load Memory from Cassette/Paper Tape |
| M | Memory Inspect/Change |
| R | Inspect/Change User WP, PC and ST |
| S | Execute in Step Mode |
| T | 1200 Baud Terminal |
| W | Inspect/Change current User Workspace |

Table 6.2.4
Together with the interactive commands TIBUG provides seven software routines, as user accessible utilities, that accomplish special tasks. These routines are called in user programs as extended operations (XOPs) and their numbers and functions are shown in Table 6.2.5. A character is represented in ASCII code.

| XOP | RESULT |
| :---: | :--- |
| 8 | Write 1 hexadecimal character to terminal |
| 9 | Read hexadecimal word from terminal |
| 10 | Write 4 hexadecimal characters to terminal |
| 11 | Read and write 1 character |
| 12 | Write 1 character to terminal |
| 13 | Read 1 character from terminal |
| 14 | Write message to terminal |

Table 6.2.5

### 6.2.3.3 The Texas TMS 9901 Programable Interface

The TMS 9901 programmable systems interface is a multifunctional component designed to provide interrupts, input/output ports and realtime clock in a Texas TMS 9900 microprocessor system as seen in Figure 6.2.3.


Figure 6.2.3 The TMS 9901 Programmable Interface


Figure 6.2.4 The TMS 9901 in a TMS 9900 System

The TMS 9901 interfaces to the CPU through the commnications register unit (CRU) and the interrupt control lines as can be seen in Figure 6.2.4.

The possible interface to the TMS 9901 consists of 22 pins that can be divided functionally in three groups as indicated in Figure 6.2.3. The 6 pins in Group $1(\overline{\text { INT1 }}$ - $\overline{\text { INT6 }}$ ) are normally dedicated to interrupt inputs (active low) but may also be used as input ports (true data in). Group 2 of pins ( $\overline{\mathrm{INT}} / \mathrm{P} 15$ $\overline{\text { INT15/P7) }}$ consists of 9 pins which can be individually programmed as interrupt inputs (active low), input ports (true data in) or output ports (true data out). Group 3 of pins ( $\mathrm{PO}-\mathrm{P} 6$ ) can be used either as input or output ports.

The first stage in programing the TMS 9901 arises from the possibility of masking (disable), through its masking register, all the input lines that we want to use as true data in and not as interrupts. As we can see from Figure 6.2.3 there are 15 lines in this condition. This means that if we disable (mask) interrupt 1 , in the mask register of TMS 9901, and the INT1 line becomes low, the microprocessor interfaced to the TMS 9901 will not receive an interrupt. If it is necessary, the CRU line associated with $\overline{\text { INT1 }}$ can be read into the microprocessor. If one masks (disable) all the interrupts in the mask register, he will be able to use all 22 pins as input lines. It is stressed here that Group 1 of pins cannot be used as output ports, as shown in Figure 6.2.3, which enables a maximum of 16 output lines.

The TMS 9901 interfaces to the CPU through the CRU and being so, when the microprocessor system is designed, a base address is assigned to the 9901 chip. This base address is the address of the first bit that can be reached in the TMS 9901. In order to reach other bits we must add to this base address, the address the bit has inside the TMS 9901. The CRU bit assignments inside the programmable interface are given in Figure 6.2.5.

The first bit that can be reached in the TMS 9901, shown in Figure 6.2.5, is bit 0 ; its address is given by the base address. This bit is called the control bit and it gives the possibility to work in interrupt mode (a) or clock mode (b).

| SELECT BIt | S, S, S: S, S. | CRU Resd Data | CRU Write Oata |
| :---: | :---: | :---: | :---: |
| 0 | 0000000 | CONTAOL BITil | CONTAOL BIT ${ }^{\text {(1) }}$ |
| 1 | 00000001 | WNT1/CLK12] | Mask 1/CLKif3) |
| 2 | 000010 | INT 2 CLK 2 | Mask 2/CLK 2 |
| 3 | $\begin{array}{llllll}0 & 0 & 0 & 1 & 1\end{array}$ | INT3/CLK3 | Mask 3/CLK3 |
| 4 | 0 0 01100 | INT4/CLK4 | Mask 4/CLK4 |
| 5 | $\begin{array}{llllll}0 & 0 & 1 & 0 & 1\end{array}$ | INT5/CLK5 | *ask 5/CLK5 |
| 6 | 0 0-110 | INTGiCLK6 | Mask 6/CLK6 |
| 7 | $\begin{array}{llllll}0 & 0 & 1 & 1 & 1\end{array}$ | INT7CLK | Mask 7/CLKJ |
| 8 | 010000 | iñalclks | SAask 8/CLK8 |
| 9 | 0 1 1 0 0 0 1 | INT9/CLK9 | Mask 9/CLK9 |
| 10 | 0 0 1 00010 | INT10/CLK10 | Mask 10/CLK 10 |
| 11 | $\begin{array}{llllll}0 & 1 & 0 & 1 & 1\end{array}$ | INT11/CLK11 | Mask 11/CLK 11 |
| 12 | $\begin{array}{lllll}0 & 1 & 1 & 0 & 0\end{array}$ | INT12/CLK 12 | Mask 12/CLK 12 |
| 13 | 0 1 1 1 001 | INTI3/CLK13 | Mask 13iCLKi3 |
| 14 | 0 1 110 | 何T:4/CLK 14 | Mask 14/CLK14 |
| 15 | $\begin{array}{lllll}0 & 1 & 1 & 1 & 1\end{array}$ | INT15/INTREQ ${ }^{(7)}$ | Mask 15/8ST2 ${ }^{(4)}$ |
| 16 | 10000 | PO input ${ }^{\text {(5) }}$ | PO Outsut ${ }^{(6)}$ |
| 11 | 100001 | Pi input | P1 Output |
| 18 | 10010 | P2 input | P2 Ouiput |
| 19 | 100011 | P3 Input | P3 Outour |
| 20 | 10100 | P4 input | P4 Ourput |
| 21 | $1 \begin{array}{lllll}1 & 0 & 1 & 0 & 1\end{array}$ | PS input | P5 Output |
| 22 | 100110 | P6 Input | P6 Output |
| 23 | $1 \begin{array}{lllll}1 & 0 & 1 & 1 & 1\end{array}$ | Pj Inout | P7 Output |
| 24 | 110000 | P8 Input | P8 Ouiput |
| 25 | 1100001 | P9 Input | P9 Outpur |
| 26 | 11010 | Pioinqut | 910 Outpur |
| 27 | 1110011 | P11 Input | P11 Output |
| 28 | 1 1 1000 | P12 Inpus | P12 Output |
| 29 | 1111001 | P13 input | P13 Outpue |
| 30 | 11110 | Pi4input | P14 Outpui |
| 31 | $1 \begin{array}{lllll}1 & 1 & 1 & 1\end{array}$ | P15 input | P15 Output |

Figure 6.2.5 TMS 9901 CRU bit assignments


Figure 6.2.6 TMS 9901 real-time clock

## a) The Interrupt Mode

When bit 0 (control bit) is set to zero value the TMS 9901 will give the user two possibilities: 1. (Reading) The user can read CRU bits 1 to 15 into the microprocessor regardless of the mask register value on TMS 9901. It is remembered that, if the interrupt line is not disabled in the mask register, care must be taken because, if the line goes low, one interrupt will be requested to the microprocessor and, depending on the interrupt mask value on the microprocessor itself, this interrupt will be accepted or not. This first possibility enables the use of all 22 pins as true data input lines. 2. (Writing) The user can enable or disable interrupt lines writing ones or zeros to the CRU bits 1 to 15. The written data will go to the TMS 9901 mask register as can be seen in Figure 6.2.3. This mask value will remain the same unless it is changed by another writing instruction.

## b) The Clock Mode

The TMS 9901 has an internal real-time clock that can be used as an interval or event timer. The configuration of this real-time clock can be seen in Figure 6.2.6. When bit 0 (control bit) is set to value one we will have two possibilities: 1. (Writing) When the user writes data to CRU bits 1 to 14 this data will go to the clock register and simultaneously to the clock decrementer. As soon as the last bit is transferred the clock decrementer starts to decrement. When it reaches zero the clock generates an interrupt ( (NT3) which can be used as an interval timer. The frequency that decrements the clock decrementer is the input frequency divided by 64. If the input frequency is 3 MHz (main frequency) a maximum interval of 349 ms with a resolution ( 14 bits) of $21.3 \mu \mathrm{~s}$ can be used.

When the clock interrupt is active the clock mask (mask bit 3) must be written into, with either one or zero, to clear the interrupt. If a timer value other than that initially programmed is required a new 14 bit clock start value is loaded into the clock, executing a CRU write operation. Normally the clock is disabled by "powerup clear" or by writing a zero value into the clock register.
Enabling the clock programs the third priority interrupt ( $\overline{\text { INT3 }}$ ) as the clock interrupt and disables generation of interrupts from the $\overline{\text { INT3 }}$ input pin. It is important to note that, for maintaining
the system integrity all interrupts should be disabled when accessing the clock. 2. (Reading) The current value of the clock decrementer can be transferred to the read register whenever the control bit is switched to the clock mode. Reading the value, in this register (reading in the clock mode) which is proportional to the elapsed time between the clock enabling and reading, means using the clock as an event timer. A software reset can be performed by writing one to the control but followed by writing a one to bit 15 , which forces all I/O ports to the input mode. The status of the TMS 9901 can be known reading the control bit (bit 0 ) and reading bit 15 (INTREQ) shows if the TMS 9901 has sent an interrupt request to the microprocessor. We stress here that, if the interrupt is enabled in the mask register and the corresponding interrupt line goes low, the interrupt will reach the microprocessor regardless of the value in the control bit.

The TMS 9901 real-time clock is very convenient to implement sampling operations without extra hardware as we will see later on.

### 6.2.4 Memory Expansion - Necessity

As was mentioned before and can be seen in Appendix 6.1, the TM 990/100M-1 microcomputer board does not allow DMA to the memory board. Basically, the control lines come out of the board through unidirectional buffers. So, any attempt to use this board in a DMA configuration will require first a memory expansion accessible to DMA. A general configuration for such expansion is given in Figure 6.2.7.


* accessible to DMA

Figure 6.2.7 General Direct Memory Access Configuration

This memory expansion board is intended for data commication between microcomputers so the best possible choice of memory type, for this application, is RAM (read and write possible).

The design for this memory expansion board was based on a general application idea that would have to be simple and cheap. The first necessary decision to be taken was about the size of memory block this board will contain. The first point to consider was the maximum physical size the printed circuit board could have. This was limited firstly by the etching facilities available in the Department and secondly by the available edge connectors and space inside the racking system. At the end, the space inside the racking system and the standard edge connector available restricted the area of the printed circuit board to $200 \times 114 \mathrm{~mm}$ and the number of connection pins to 86 in a double sided card with 254 mm between pins on the 114 mm side.

A general sketch of the board, with the necessary decoder and buffer logic, showed that limiting the spacing between chips to a minimum, up to 16 memory chips could be accommodated on the board. The minimum spacing would mean of course thin connection lines and the possibility of problems when etching the boards. This risk was thought to be acceptable and it was decided to design the board for 16 memory chips.

The readily available memory chips of the MOS static RAM type at the time the board was designed were the ones given in Table 6.2.2.

| Type (TMS) | Number of Bits | Organisation |
| :---: | :---: | :---: |
| 4033 | 1024 | $1024 \times 1$ |
| 4034 | 1024 | $1024 \times 1$ |
| 4035 | 1024 | $1024 \times 1$ |
| 4036 | 512 | $64 \times 8$ |
| 4039 | 1024 | $256 \times 4$ |
| $2111 / 4042$ | 1024 | $256 \times 4$ |
| 4043 | 1024 | $256 \times 4$ |
| 2114 | 4096 | $1024 \times 4$ |

Table 6.2.2

To have a reasonable amount of memory to work with the TMS 4039, 2111, 4043 and 2114 were the only ones to be considered. The Microcomputer board already used the 2111 type on its RAM area and so the choice was made between the 2111 (TMS 4042) and 2114. Using the TMS 4042 the board would have ( 16 chips) a capacity of 1 K words with a minimum of 256 words ( 4 chips ). Now, using the TMS 2114, the capacity would be 4 K words ( 16 chips) with a minimum of lK words ( 4 chips).

The final decision, on our tight budget, was taken based on a minimum price for a workable expansion. The minimum requirement for a workable expansion was 4 chips in each case. As the price of the TMS 2111 was half of the TMS 2114 , the first was chosen for the memory expansion design. Details of the memory expansion board design are given in Appendix 6.2.3 and the finished board is shown in Figure 6.2.8. The list and description of components for the board is given in Appendix 6.2.4.
6.3 THE GENERAL MAPPED COMMUNICATION STRUCTURE
6.3.1 The Microcomputer's Communications Global Memory Area

Tightly coupled multi-microcomputers with uniform access, by all processing elements, to all main memory, have a switching structure whose cost grows as the product of the number of processors and the number of memory units. Thus, the processor/memory interconnect becomes prohibitively expensive as the number of processing elements and memory modules grows over certain limit (69).

A requirement set on the Cm* computer system was that each processor is able to address directly all main memory, rather than require a message transmission for access to remote units as in a network. Uniformly fast access to all of memory by each processor was not, however, considered necessary either for system performance or for generality of experimentation. Based on the success of cache memories it was verified that a processor's memory references tend to cluster in a small portion of its address space.

Results presented by Fuller et al (69) indicated that for the processors used in $C \mathrm{~m}^{*}$, instructions and temporary data usually account for between 90 and 99 per cent of the memory references. When a task is subdivided so that several processors may perform different parts of it in parallel, the shared global data accessed

MEMORY ‘EXPANSION BOARD
by many or all of the processors often accounts for most of the total main memory required by the task. However, the results indicate that these global locations are accessed so infrequently that it makes little difference if their access times are substantially longer than those for instruction and temporary data.

In a multi-microcomputer structure for real-time control, it is not reasonable to assume that a microcomputer will execute instruction codes not stored in its local memory. If the same instructions are needed by two or more microcomputers, it is assumed that local memory will be available to duplicate the instructions. Even if for some special reason the memory available is not sufficient, it is always possible, with a fast communication link, to send a complete set of program instructions to a certain microcomputer in order to have it executed in local memory.

This is one of the key points in this design. The communication process is used basically to transfer DATA (Global) between microcomputers. We used the term basically because at a certain stage it will probably be necessary to transfer a complete program, although this will be done so infrequently that it will not affect the system's throughput.

In our design, in order to keep the hardware reasonably simple and cheap, it was decided that each microcomputer should be able to address directly only its own address space of 32 K words ( 16 bits). If the microcomputer is able to access only its own address space, and microcomputers do not share all the same address space, in order to communicate with others, part of this address space must be dedicated to communications. Accepting the results mentioned before, it is assumed that up to 5 per cent of the address space can be used for inter-microcomputer communications. In our case it was decided to have 2 K words dedicated to inter-microcomputer communications, which accounts for approximately 6 per cent of the available address space of 32 K . This 2 K words block also will help, later on, to achieve a reasonably organised communication software.

So, 2 K words of a microcomputer's address space will be made accessible to all other microcomputers as can be seen in Figure
6.3.1. Microcomputer module or just MODULE is used from now on
in this work to define a microcomputer board, as the Texas TM 990/100M-1, and its associated expansions and interfaces, linked by the microcomputer's internal bus, which allow it to operate in a stand alone mode or as a module in the multimicrocomputer structure.

It is quite clear that, if a microcomputer access is a certain amount of memory located in other microcomputer's address space using its own addressing capability, this memory is really almost a virtual memory and a switching scheme will be necessary. This scheme will switch the virtual address to the real physical location or, better, switch from local to external memory. Local memory is the name used from now on to designate the memory accessed by the microcomputer without the use of the common bus. If memory is accessed using the common bus it is called external memory.

The access time of a microcomputer to the conmunication common memory locations (external memory) does not need to be the same as if it were local memory. Having this in mind and keeping the simplicity of the design, it was decided to use the microcomputer's facility to be held inoperative ( $\overline{\mathrm{HOLD}}$ state) while communication takes place in its address space commanded by another microcomputer. This of course means that any communication process takes really the time of two microcomputers to complete. The time of the microcomputer that controls the communication process is from now on called the MASTER and the time of the microcomputer that is held inoperative, is called from now on SLAVE. If communication takes place only when the slave agrees, this doubled time can be thought of to be the same as it would be having the external memory with a longer access time than local memory.

In order to make clearer the distinction between buses, we shall call the multi-microcomputer's common bus the SYSTEM BUS and the module's internal bus the INTERNAL BUS.

At the outset of this work it was decided that only elements capable of commanding the transfer of information would be connected to the system bus. Elements not capable of doing so, such as extra memory banks and terminals, would not be connected
to the system bus directly. There was no reason to believe that such elements should be accessible to all modules on the system and not just one. So, any of these elements, when necessary, would be connected to the system bus via a microcomputer. As we shall see later on, the hardware designed is very simple and this restriction can be relaxed enabling the connection of passive elements to the system bus in a straightforward way.

Also, it was decided that the design should make possible that every module connected to the system bus should be able to behave as master or slave. This implies that the interface hardware and software should cater for it, as we will see later on. The exact location of the communications global memory area on the address space of each module is decided later on in this chapter.

### 6.3.2 Externally Referenced Communication Areas

The microcomputer's communications global memory area, as seen in Figure 6.3.1, is accessible by all the modules sharing the system bus. As modules can act as masters and slaves, every module will have in its memory space a block reserved to this end. Trying to make it clearer, Figure 6.3.2 shows a configuration with only three modules and the accessibility to the module's own communications global memory areas (MCGMA).

From the considerations already made, it is quite clear that there are as many MCGMA as there are modules in the system and that for a module to gain access to an MCGMA that lies in another module, the first must be a master. So, a master has the possibility to access externally as many MCGMA as the number of modules connected to the system bus, minus one. The master's own. MCGMA is a local memory of course.

Each MCGMA externally referenced by the master must be uniquely identified for a reliable communication process. As each master has got access only to the microcomputer's own address space, the only possibility for unique identification is that each externally referenced MCGMA must be mapped in different locations on the address space. This requires a mapping of 2 K word blocks for each externally referenced MCGMA. When the master refers to one


Figure 6.3.1 Microcomputer's Global Communication Area


Figure 6.3.2 Accessibility to Module's Global Memory Areas
of these blocks there is a unique association to a certain module's communication global memory area.

Figure 6.3.3 shows this general mapping of communication areas. There, it is shown very clearly how the external MCGMA are accessed by module 2. For example, when module 2 refers to block 1 on its own address space, it is really accessing the communication global memory area of module 1. The MCGMA of module 2 is accessed by the other modules using the same mapping structure.

The approach discussed above solves the unique identification problem but does not cope with software organisation problems. The position of each block of 2 K words externally referenced changes from module to module and any software program must track in each module to where a certain 2 K block is referring.

A simple solution to this problem is achieved by identifying each module in the system with a determined number. This is done simply by numbering the modules starting from 1 . Then we have modules: module 1 , module $2 \ldots$..... module $n$.

At this moment the other reason for the choice of a 2 K word block for global communication purposes will become clearer. Dividing the microcomputer's address space in blocks of 2 K words gives exactly 16 blocks ( 32 K total). Starting from address and going upwards, 16 blocks of 2 K words start when the most significant digit in the hexadecimal address is incremented by one. This indicates a simple way to identify the blocks using the most significant digit in the hexadecimal address, as shown in Table 6.3.1

| Memory Locations (Hex) | Block (2K words) |
| :---: | :---: |
| 0000 to 0 FFE | Block $\emptyset$ |
| 1000 to 1 FFE | Block 1 |
| 2000 to 2 FFE | Block 2 |
| 3000 to 3 FFE | Block 3 |
| 4000 to 4 FFE | Block 4 |
| , | 1 |
| , | , |

Table 6.3.1
module 1
module 2

module 4

Figure 6.3.3. General Mapping of Communication Areas
lock $\emptyset$ on the RM 990/100M-1 microcomputer address space can be considered a special one for the dedicated application of certain memory locations inside this block. Memories $\emptyset \emptyset \emptyset \emptyset$ to $\emptyset \emptyset 7 E$ are dedicated to interrupt and extended operations vectors (77). In the author's view, this area must not be made accessible to other modules (made an MCGMA) in order to maintain the module integrity in case of failure in the communcation process.

If we associate the block number to the module number starting from 1, we have an easy way to know which module external reference is being made to. For instance, if a master module 2 wants to access the communication area on slave module 5 , the master has only to refer to the block of memory from address 500016 to $5 \mathrm{FFE}_{16}$; slave module 1 requires reference to addresses $1000_{16}$ to $1 \mathrm{FFE}: \quad$ It is clear that if master module 2 refers to addresses 200016 to 2 FFE , it is trying to communicate with itself and, if the memory addressed physically exists,it will be a local reference. Figure 6.3 .4 shows the mapping of communication areas on the address space of a typical module.

### 6.3.3 The Number of Supported Modules

From Figure 6.3 .7 it is quite clear that, as the number of modules with which a typical module can communicate increases, the available space for local memory decreases on the address space. It is pointed out here, and also shown in Figure 6.3.4, that the communications global memory area must be mapped as local memory on the module's available address space.

At this point the question of how many modules the designed system will support arises. In the multi-microprocessor structures already mentioned applied to real-time control, the maximum number of processors sharing a parallel common bus is found to be around eight. In our particular case, at the moment, we have only 2 microcomputers and are looking for advanced control modes of positioning systems. Assuming that only three axes of movement will require advanced control in a general positioning system, a minimum of four modules should be supported by the design. As we will see later on, to this minimum number, one module is added to care for the system bus management. So if the system is

Address space


Figure 6.3.4 Mapping of Communication Areas for a Typical Module
designed to support ten modules it will cope with a general position control system.

The system bus is made of course of real transmission signal lines. The total transmission path length between connected modules must be made electrically short in order to make the transmission line propagation delay not important. If a transmission line delay is to be maintained less than 2.0 nsec the system bus must have a maximum length of less than 600 mm . Each module must have mechanical access to the system bus so dividing the maximum bus length by the number of modules gives the distance between each connection to the bus. If each module was made of just one board we could go to the mechanically workable standard of 1 in ( 25.4 mm ) and have up to 24 modules connected to the maximum bus length in an equally spaced manner. In our case, of course, this is not possible. Our modules are made of many different boards and, in order to keep a short distance between the module and the system bus, the spacing between connections must be increased. Increasing the spacing means of course a decrease in the possible number of modules. For a ten module system the space between equally spaced connections is almost 60 mm and modules formed by two boards can be placed very close to the system bus. As we will show later on, this configuration could not be achieved due to a restriction on the size of boards possible to be manufactured. Even so, we kept the idea of supporting ten modules.

Having fixed the number of modules, we can now decide the position of the communications global memory area in the address space. There are four points which lead to the final decision. The first point is that, for a better organisation of software programs, the area should be in the same address space location for every module. The second point is that, supporting ten modules and not using for communication purposes the area from addresses 0000 to OFFE (Interrupt and XOP Vectors), the communications area must lie between addresses BOOO to FFFE. The third point is that, being a 2 K word block, the communication global memory area is easily identified if it lies starting on addresses B000, C000, D000 or F000. The fourth point is that, to avoid any awkward partition in the memory space in case less than ten
modules exist in the system and the local memory needs to be expanded over the non used externally referenced areas, the communications global memory area is best placed starting at address F000. This is the chosen position for the communications global memory area in every module's address space and it is indicated in Figure 6.3.4.

### 6.3.4 The Role of the System Controller

The use of a common system bus, as the means of communication between modules, makes it a possible source of contention problems. Being a communication link there should be only two modules connected to it at the same time: one that is sending information and one that is receiving (master-slave). If two modules want to use the bus to send information, there should be an organised way to provide them with access without any interference.

The simplest way to organise access to the system bus is to have a system controller. This system controller has, or could have, many functions associated to it:

1. Arbitration - solves the contention problem arising when two modules want to use the bus at the same time.
2. Bus Granting - bus must be allocated to only one module at a time.
3. Bus Sensing - senses when the system bus is being used or not.
4. Detecting failures on the bus and associated circuitry.
5. System Initialisation
6. Bus Management
7.. Diagnostic Functions

It is clear that the first three functions mentioned above could be executed by a piece of specialised hardware. The other functions point towards the flexibility of a software controlled hardware or a microcomputer.

In order to have this flexibility and to maintain the modularity concept, it was decided to use one of the modules in the system as the system controller. So, the modularity concept requires that
any module in the system must be able to act as the system controller and the design must provide for that.

When only two modules are connected to the system bus, as in our system, one of the modules is the system controller but there is really no necessity for it. In this case there is no contention problem for the system controller to solve, but all the other functions can be implemented.

The system controller defined here, in relation to the contention problem, can be compared to the permanent master defined by the S-100 bus interface standard. All the other modules in the system ( temporary masters) must request the system controller (permanent master) to use the system bus.

From now on, to simplify our writing, the system controller will be called just controller.

### 6.4 General Logical Design

The design approach to the multi-microcomputer system hardware will be based on the communication sequence.

When a module wants to change information with another one using the system bus there are well defined phases on the communication sequence through which it must go. They are:

1. Bus Grant Phase

First thing before a module can use the bus is to ask permission from the controller. When this permission is granted, the module can proceed with the communication. This phase could be considered as a master definition phase.
2. Slave Definition Phase

In this phase, the module which has command of the bus, in this work called the MASTER, must define to which microcomputer (SLAVE) it wants to communicate.
3. Slave Acceptance Phase

When the slave accepts the definition it signals back to the master indicating that communication can take place.
4. Communication Phase

In this phase, data is transferred between MASTER and SLAVE commanded of course by the MASTER.

## 5. End of Communication Phase

After communication has finished, the MASTER must signal back to the slave and the controller to inform it.

The hardware design will try to follow the communication phases as often as possible.

### 6.4.1 The Interrupt Controlled Arbitration Link

The bus grant phase was implemented by using an interrupt controlled chain link. The link will set a priority level to each microcomputer. This priority level will dictate which microcomputer will use the bus in case of contention. The physical location of each microcomputer on the chain, in relation to the controller, is associated with the priority leve1. Highest priority will be nearest the controller.

The general configuration of such a link can be seen in Figure 6.4.1. Its simplicity is the great advantage, for it has only three lines which we will call INTCTL (interrupt controller) RACKO (received acknowledge) and BUSBUSY (bus busy).


Lower Priority

$$
\text { see paç. } 17
$$

Figure 6.4.1 Interrupt Controlled Arbitration Link


Figure 6.4.2 Interrupt Controlled Arbitration Link Design


The INTCTL line is set by any microcomputer as soon as it wants to use the system bus. As Figure 6.4 .1 shows, there must be the possibility of all the microcomputers setting this line at the same time. As soon as INTCTL is set, the controller will be interrupted, depending on the acceptance of the interrupt and, if the system bus is free, it will set the RACKO line. This line carries back to the microcomputers, the grant to use the system bus.

If a microcomputer has asked permission to use the system bus, the switch shown on Figure 6.4.l must be open. This will guarantee that the first time the RACKO line is sensed set by the microcomputer it will take over control of the system bus. The priority associated with the physical location on the link is clearly understood by now.

As soon as a microcomputer finishes using the system bus, the switch associated with it will be closed again allowing the RACKO line to serve the microcomputers down the link.

A point must be raised at this stage in relation to the modularity concept raised before. It is seen in Figure 6.4.1 that the link lines have different interpretations for a normal microcomputer and a controller microcomputer. So, in order to assure modularity, a simple switching structure must be provided for on the interface design.

The final configuration for the link design. is given in Figure 6.4.2 and a timing diagram in Figure 6.4.3.

There are some implicit aspects of the design which require further consideration.

All the necessary lines for the link will be operated through the TMS 9901 interface already described. When the microcomputer is reset all the output lines from the TMS 9901 go to a high level. So, this condition is assumed to be the non-operational condition. That is why an inverter is used for the SICTL (set interrupt control) line.

An interrupt line on the TMS 9901 interface is active when it goes low, so the INTMST (interrupt master) line is properly buffered.

The switches shown on Figure 6.4 .2 are related to the controller function. They will have to be set properly to define if a microcomputer is a controller or not. It is good to remember that there should be only one controller for the system bus. Even so, the setting is very simple and straightforward.

Table 6.4.1 gives the necessary connections to be made in order to have proper operation.

| MICROCOMPUTERS |  |  |  |
| :---: | :---: | :---: | :---: |
| Controller |  | Others |  |
| Connect | Disconnect | Connect | Disconnect |
| $g-\mathrm{h}$ | $\mathrm{w}-\mathrm{y}$ | $\mathrm{w}-\mathrm{y}$ | $\mathrm{g}-\mathrm{h}$ |
| $x-\mathrm{v}$ | $\mu-\mathrm{v}$ | $\mathrm{u}-\mathrm{v}$ | $x-\mathrm{v}$ |
| $x_{1}-\mu_{1}$ | $\mathrm{v}_{1}-\mu_{l}$ | $\mathrm{v}_{1}-\mathrm{u}_{\mathrm{l}}$ | $x_{l}-\mathrm{u}_{1}$ |
| $\mathrm{a}-\mathrm{b}$ | $\ell-\mathrm{m}$ | $\ell-\mathrm{m}$ | $\mathrm{a}-\mathrm{b}$ |

Table 6.4.1

The gates driving the INCTL and the BUSBUSY lines are open collector types and the pull up resistors are implemented on the controller side by the pair of resistors which will act also as line terminators.

The switch $w-y$ is only implemented to show the possibility of having the correspondent line free for any other use, having in mind that the TMS 9901 has a limited number of interrupt lines available and that some $I / O$ and interrupt lines are mutually exclusive. This could be implemented of course in all other dual function lines, but to avoid misuse of the system bus each line must have only one possible function assigned to it.

The BUSBUSY is basically a line to indicate that the systems bus is being used. This line should be set by the microcomputer controlling the system bus when it takes over and reset as soon as it finishes using the bus. There are two main functions associated with this line. The first one is to indicate that the system is being used so that the controller can reset the RACKO line and the second and most important one is to indicate to the controller that the master has not been able to take over the system bus for
any reason. If, after the bus is granted by the controller, the bus busy line is not set there will be problems on the interrupt link or on the microcomputer which required the use of the bus. Software design must take account of this problem. The TACKO (transmitted acknowledge) line shown in Figure 6.4.2 is the continuation of the RACKO line after the switch shown on Figure 6.4.1. As soon as the TACKO line leaves the microcomputer where it originates, it becomes the RACKO line for the next module on the chain.

The use of certain types of gates will become clearer when the total hardware design is finished. Basically we tried to utilise a minimum number of integrated circuits on the overall design.

The timing diagram of Figure 6.4 .3 shows the two possible sequences on the interrupt link. On the first portion of the diagram, the microcomputer has set the SITCTL line (low) asking permission to use the system bus. As soon as the bus is granted (RACKO-low) the microcomputer takes over. It is noted that the TACKO line, that is, the transmitted RACKO line, remains inoperative (high) and the grant is accepted only by the microcomputer with the highest priority on the link.

The second portion of the timing diagram shows the sequence of events when the microcomputer has not asked permission to use the bus (SITCTL-high). When the bus is granted (RACKO-1ow) the grant message is just passed over, through the TACKO line, to the next microcomputer on the link.

### 6.4.2 Slave Definition Phase

After the system bus has been granted to a certain microcomputer, the next phase on the communication process is the definition to where the microcomputer wishes to communicate.

As was mentioned before, the communication is always on a masterslave structure and the master is always in command of the bus during the exchange of information.

The first question that arises is to how many slaves a microcomputer can communicate. This question has been answered before, when we decided to have 2 K words allocated to each microcomputer on the
memory map. So, the number of slaves is expected to be up to ten.

There are many ways by which a master could indicate to a certain slave its desire to communicate. The only restriction is that the indication must be precise and clear to avoid any problems.

The design solution is to use a memory mapped procedure to achieve a fast and clear slave definition. The master will have mapped on its memory different addresses for each possible slave and when these addresses are accessed the correspondent slaves receive an indication that communication with them is required. The advantage of this procedure is that no extra lines are required between the microcomputers, for the system address bus will be used to carry the information. The main disadvantage is the overhead in software that will be necessary for the purposes of master identification. This overhead, estimated here to be of one extra word to be transmitted, is so small in execution time, when using DMA, that it is thought to be of negligible consequence to the final design.

With the possibility of ten slaves at least four lines on the address bus must be used for coding the identification. According to the memory map already described, it is possible to have the most significant four lines on the system address bus to convey the slave identification code.

Each module will have associated to it a switch detemined address. When a certain microcomputer must be defined as slave, its correspondent address must be referred to. The addresses chosen to identify the possible ten microcomputers are given in Table 6.4.2. If is better to have them sequentially arranged for easy reference when software writing.

Table 6.4.2

| MODULE | ADDRESS <br> (HEX) |
| :---: | :---: |
| 1 | 1 FFE |
| 2 | 2 FFE |
| 3 | 3 FFE |
| 4 | 4 FFE |
| 5 | 5 FFE |
| 6 | 6 FFE |
| 7 | 7 FFE |
| 8 | 8 FFE |
| 9 | 9 FFE |
| 10 | AFFE |

The addresses given on Table 6.4 .2 must be decoded to indicate which slave is being defined. The safest way to achieve it is to decode the main part of it on the master module and only the remaining four most significant address lines on the slave module. This will avoid the use of all system address lines.

When decoding, it was decided that only when writing to the addresses should the decoder be activated. This would help decrease the risk of undesirable decoding. The decoder main configuration and working principle is shown in Figure 6.4.4.

After the main part of the slave address has been decoded, the remaining part is decoded by all decoders receiving information from the lines AO-A3 on the system bus. The general configuration is shown on the right side of Figure 6.4.4.

The final design will be shown later on, after we have discussed the slave acceptance phase for there are details that interlock both phases.

### 6.4.3 S1ave Acceptance Phase

After the master has indicated to a particular slave its wish to communicate, the slave must signal back to ensure a complete handshake. When the master receives back the acceptance signal, it will hold the slave and process communication. It is at this point that the design uses the facility of the hold state provided by the TMS 9900 microprocessor.

As soon as the microprocessor is in the hold state it signals with a HOLDA (hold acknowledge) line. Having in mind design simplicity we are going to use the HOLDA line for two purposes. The first to signal the master and second that the addressed slave is already in the necessary state for direct memory access. In order to be able to achieve this double sided objective, the master must set the $\overline{H O L D}$ line as soon as it defines the slave - see last section.

So, this phase will require two more lines on the system bus. They are, the $\overline{H O L D} . B$ and HOLDA.B lines. Figure 6.4 .5 shows these lines on the system bus and the necessary inputs and outputs to the microcomputers. It is clear that because any. microcomputer can act as master or slave, depending on the circumstances, it is necessary to have four $1 / 0$ lines to which the HOLD.B and HOLDA.B

Microcomputer Internal
Address lines


Figure 6.4.4 Slave Address Decoding


Figure 6.4.5 The Master-Slave Hold-Acknowledge Bus
system bus lines will be decoded. A master will use the HOLDSET and RECHOLDA lines to set and receive the $\overline{H O L D} . B$ and HOLDA.B lines respectively. A slave will receive the $\overline{H O L D} . B$ line through HOLD and will acknowledge through the HOLDA line.

The ENA $\overline{H O L D}$ line shown in Figure 6.4 .5 is a necessary one to avoid any problems due to the hold state. As was mentioned before, the microcomputer's hold state is only tied to the execution of a memory cycle. As soon as the actual memory cycle is finished, the microcomputer will enter a hold state, when the $\overline{\mathrm{HOLD}}$ line is held down. This means that any important process requiring attention at that time will suffer from discontinuity if nothing is done to avoid it. The ENAHOLD line solves this problem. The $\overline{H O L D} . B$ line from the system bus will only be connected to the $\overline{H O L D}$ line on the internal bus when the ENAFOLD line allows it

### 6.4.4 Communication Phase

After a master has received the information that the addressed slave is in the hold state, the transfer of information can take place.

This will be achieved by the master taking control of the system bus data and address lines. It is clear, as was mentioned before, that the communication is only on a one to one basis: one master, one slave.

Figure 6.4 . 6 shows the general configuration for the logic control on the communication phase. The duality of functions is there of course as each microcomputer can act as master or slave.

It is clearly seen in Figure 6.4 .6 that the access to the system bus is controlled by the three buffers on the address, data and control lines. Due to the bidirectional use of the system bus, the buffers will be controlled by two lines. One line sets the direction of flow through the buffers and the other enables or disables them, that is, closing or opening the communication between the microcomputer's internal bus and the system bus.

For the sake of better understanding, we will start discussing the slave side of the design.

The slave will have its communcation memory accessed by the master and being so, the slave's buffers on the address and control lines


Figure 6.4.6 The General Bus Control Logic on Communications
will have a definite direction set on them: from system bus to the internal bus.

On the data side the direction on the buffer will be determined by the master's intention to read or write from comunication memory. If the master wants to read, then the direction of data flow will be from internal bus to system bus and vice-versa if it wants to write. The control of direction on the address and control lines buffers will be done using the HOLDA line. When high it indicates that the microcomputer is in the hold state and therefore is a slave. The control of direction on the data lines will be done using two lines: the HOLDA line and the DBIN line. We will explain this point and the enabling of the buffers later on when we talk about the master side of design. The address lines received by the slave must be decoded of course. As the area of memory ( 2 K words) dedicated to communication is fixed on the memory map there is no reason to input all the address lines on the slave module. The most significant four lines of the address bus ( $\mathrm{AO}-\mathrm{A} 3$ ) on the slave internal bus will be generated by the communication address area generator shown in Figure 6.4.6. The HOLDA line will enable or disable the communication address block depending on the module being a slave or not. The master side on the design is a little more involved. When the master addresses a specific address that lies on the slave module, the decode shown on Figure 6.4 .6 will recognise the address as being an external one and will enable the address and data buffers. The HOLDA line comes to the decoder to cater for the slave behaviour. As soon as the slave is in the hold state (HOLDA - high) the buffers will be enabled. The direction on the address bus buffer will be from the internal to the system bus. On the data bus buffer the direction will be dealt with using the HOLDA and the DBIN lines as on the slave side. If the module is a master, reading data will mean from system to internal bus and writing data will mean from internal to system bus. In the slave, data being read will mean from internal to system bus and writing will mean from system to internal. Table 6.4 .3 shows the direction on the buffers for different situations.

|  |
| :--- |
| Direction* |
| Buffers |
| Control |
| Master |
| Data |

Table 6.4.3

### 6.4.5 End of Communication Phase

There are two aspects concerning the end of communication phase. One is related to the slave and the other to the controller.

The master-slave end of communication is a necessary measure. Using the hold state to achieve the transfer mades the recognition by the slave of the end of transmission not easy if time is not to be lost. In order to maintain the number of lines in the system bus to a minimum and exploring the flexibility given by the already defined interfaces, the master - slave end of communication will be achieved using a software flag.

The master-controller end of communication is necessary to indicate to the controller that the system bus is free again and can be allocated to other users. There are two ways to achieve this in the already defined interfaces. The first, and obvious, is using the BUSBUSY line of the arbitration link shown in Figure 6.4.1. The master would reset this line at the end of transmission and the controller would recognise it. The second is using the slave definition process to signal the controller in the same way the master deals with the slave, as mentioned before. Depending on which input lines we use, both options are equivalent and this decision will be taken later on.

After the general logical design has been gone through, we are now able to proceed to the hardware design for the system bus interface.

### 6.5.1 Integrated Circuits

The use of any TTL integrated circuit at the moment calls for a low-power Schottky series device. It has a combined low propogation delay and power dissipation with a maximum working frequency well above the microcomputer's clock frequency. The range of functions available is quite wide although sometimes devices from other TTL series have to be used.

In our case, there was another restriction when trying to achieve a certain logical function. As was mentioned before there was no money left after the microcomputers and power supply were bought, so any necessary IC had to come from the available stock maintained at the Control Laboratory. This stock was not supposed to be a designer's one but a simple maintenance facility for some of the existing equipment.

The use of the mentioned stock did not restrict the design as much as one would expect on the logical side but on the number of ICs available. For the system minimum configuration, all the hardware interfaces must be executed twice: one for each microcomputer. Sometimes there was available a specific IC but not in the necessary quantity. The design went around using other ICs doing the same job in a more complicated and inefficient way sometimes, but with the same functional result.

As will be seen later on, the design came to be very simple and substitution of any IC could be done very easily, on paper at least. Another point to mention is that we did not think that the used ICs were so important at the end if they worked properly. It is quite clear that with the fast advances in microelectronics, any detail in IC logical function today can be obsolete tomorrow.

We tried to prove that the general concept for the system communication works. If in a shorttime just one IC will do the same job as the several ones we used, it will be wonderful. But if we just wait for it, nothing will be done, solets work.

### 6.5.2 Dynamic Memory Restriction

Memory applications requireing large bit storage can use dynamic RAMs for low cost, low power consumption and high density. The
drawback of such memories is that they must be refreshed periodically to avoid the loss of stored data.

There are several dynamic memory refresh techniques which can be used with the Texas TMS 9900 microprocessor if at least one cell of each row in the RAM matrix is not accessed every 2 milliseconds. If this happened, refresh would not be ncessary. The three most used techniques are the block, cycle stealing and transparent mode. All of them require special hardware control logic. If direct memory access is attempted to this type of memory, the hardware control logic will have to be even more special to ensure that refreshing is properly achieved.

Trying to design a system as simple as possible, it was decided that the communications global memory areas should be made using only STATIC RAMs. If for other parts of the memory dynamic RAM is necessary, proper buffering should be provided on the module's internal bus in order to have continuous refreshing to this area of memory even when the module goes to a hold state (slave). This can be achieved using the HOLDA line to control the buffers to the dynamic RAM area. A small step in this direction has been taken already in our design . When we look to the design of the memory expansion board. There, the control lines on the internal bus are buffered through a TN 74LS241 device controlled by the HOLDA line. This buffer really separates the internal control bus into two: the part that serves the communications global memory area and the part that serves the other elements in the module. This separation helps the buffering for dynamic RAM areas of memory but we will not take it further away in this work.

### 6.5.3 Buffering Signals on the System Bus

As was seen on the general logical design section, the transfer of information between the microcomputer's internal bus and the system bus is done through elements which we call buffers.

The buffers have two main functions. The first is to isolate the internal bus from the system bus and vice-versa when transfer is not necessary. The second is to provide routing for the information transfer.

If we exclude the arbitration link already designed, it is quite clear that all the system bus lines are bi-directional. This
means of course that information can travel on the lines in both directions, so a line reaching a buffer from the system bus could carry information from and to the buffer.

The isolation between buses can be achieved in two ways using the logic available. The first one is using buffers with threestate outputs. When disabled the buffers will set the output lines to a high impedance state and basically the state on the lines is defined by other devices connected to them. The second way to isolate buses is by using open-collector gates. These gates are not disabled like a three-state device but their outputs can be driven by an input line to a high state which is a pseudo diabled state. If other gates connected to the same line are diabled the state on the line is dictated by the enabled gate.

When lines on the system bus require a well defined state, all the time, even when not driven by any buffer, the open-collector gates offer the possibility directly. The pull-up resistor necessary for the gate proper operation will try to define a high state at the output all the time. Indirectly, the three-state buffer can provide such possibility. Using a pair of resistors between the line and the power rails, like in a termination scheme, it is possible to match the output voltage of a three-state device and maintain the line in a defined state (high).

The availability in the Control Laboratory of Texas SN 74LS245 octal bus transceivers made them the first choice for the buffers. They are bi-directional transceivers with three-state outputs and hysteresis at inputs. The control is done through two pins: the enable $(\bar{G})$ pin and the direction (DIR) pin. Table 6.5.1 is the function table for the device.

| ENABLE $(\bar{G})$ | DIRECTION <br> (DIR) | OPERATION |
| :---: | :---: | :--- |
| $L$ | L | $B_{i}$ to $A_{i}$ |
| $L$ | $H$ | $A_{i}$ to $B_{i}$ |
| $H$ | $X$ | Isolation |

H - High level
L - Low level
X - Irrelevant
$A_{i}, B_{i},-i n p u t$ pins
$\mathbf{i}=1-8$

Tab1e 6.5.1

These devices match very closely the logical requirements already mentioned. The general internal configuration for just one line is shown in Figure 6.5.1.


Figure 6.5.1 Texas SN 74LS245 Transceiver General Configuration

### 6.5.4 Logic Diagrams

The final hardware design is given in Figures 6.5.2 and 6.5.3. All the lines associated with the system bus are identified by the same names used on the microcomputer's internal bus but a suffix . $B$ is added (. $B=$ system bus). The list and description of components for each board is given in Appendices 6.5.1 and 6.5.2.

### 6.6 FUNCTIONAL DISCUSSION OF THE .DESIGNED LOGIC

### 6.6.1 The Master-Slave Identification Address Decoder

As was mentioned before, the slaves are identified by addresses 1FFE, 2FFE, ...., AFFE. If a master wants to identify a slave, having control of the system bus, it will have to write to the correspondent slave's address. As soon as the master writes to the identification address, the address pattern appears on the internal bus address lines AO to A14. At the same time, the $\overline{W E}$ line is set accordingly by the microcomputer indicating a write cycle.



The decoder recognises the correct address and being a write cycle enables the buffers on the AO - A3 lines. The lines AO.B to A3.B on the system bus are set to the required pattern and this information is transmitted to all possible slaves including the master itself.

As can be seen on Figure 6.5.2, the main decoder function is executed by IC's 1A and 6A. Devices 7A, 8A and 9A are just buffering the address lines. It is quite clear here that the decoder is not working in the way first mentioned. In order to simplify the design, and due to the non-availability of required IC's, only address lines AO to All are decoded. This means that slaves will be identified by addresses 1 FFO to 1 FFE , 2 FFO to 2 FFE , ....... AFFO to AFFE. The consequence of this is that there will be a loss of 15 words on the 2 K words available for communication purposes in each microcomputer. This loss is thought to be not critical in such a development stage.

The use of HOLDA and $\overline{H O L D}$ lines on the decoder area will be explained later on as they are related to the communication phase. Device 1 A is a Texas SN 74150 data selection/multiplexer. It selects one of sixteen data sources and its functions table is given in Table 6.6.1.

The choice of which codes are to be allowed through is made by the levels connected to data inputs EO to E15. If the data input is set to HIGH level, the code will go through, enabling, if other conditions are satisfied, the buffers on lines AO to A3, and viceversa. In the way the design is connected, codes 0001 (1) to 1010 (A) on lines A0 - A3 will go through.

The timing diagram for the master-slave identification scheme is shown in Figure 6.6.1


HOLDA


Figure 6.6.1. The Master-Slave Identification Timing Diagram

| INPUTS |  |  |  |  |  | OUTPUT |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| DATA | Select |  |  |  | STROBE | W |
|  | D | C | B | A |  |  |
| ANY | X | X | X | X | H | H |
| EO | L | L | L | L | L | EO |
| El | L | L | L | H | L | E1 |
| E2 | L | L | H | L | L | $\overline{\mathrm{E} 2}$ |
| E3 | L | L | H | H | L | E3 |
| E4 | L | H | L | L | L | E4 |
| E5 | L | H | L | H | L | E5 |
| E6 | L | H | H | L | L | $\overline{\mathrm{E}}$ |
| E7 | L | H | H | H | L | $\overline{\mathrm{E} 7}$ |
| E8 | H | L | L | L | L | E8 |
| E9 | H | L | L | H | L | E9 |
| E10 | H | L | H | L | L | $\overline{\mathrm{E} 10}$ |
| E11 | H | L | H | H | L | E11 |
| E12 | H | H | L | L | L | $\overline{\mathrm{E} 12}$ |
| E13 | H | H | L | H | L | E13 |
| E14 | H | H | H | L | L | $\overline{E 14}$ |
| E15 | H | H | H | H | L | $\overline{\mathrm{E} 15}$ |

X - Irrelevant
L - Low Leve1
H - High Level
$\overline{\mathrm{E}}_{\mathrm{i}}-$ Complement of Input
Table 6.6.1

### 6.6.2 The Slave Identification Address Decoder

The master identifies the slave through system bus lines AO.B to A3.B, so each slave must have a decoder connected to these lines which will enable a determined address associated to it to be recognised.

The decoding is done again using the SN 74150 device which was already discussed. It is device 2A on Figure 6.5.2. The choice of the slave's address is made through connections to the data inputs E1 to E10. If a certain data input is set HIGH the
slave will be recognised by the address associated to the data input.

As can be seen in Figure 6.6.1, the slave address is valid only for a short period ( $\simeq 300 \mathrm{~ns}$ ), so some device with memory facility must be used between the decoder and slave microcomputer. This device will store the master definition information until the slave accepts the definition. After the slave has accepted the definition, it clears the information on the device's memory enabling new definitions to be received.

The memory facility in this case is provided by a SN 7473 dual J-K flip flop with clear.

The timing diagram for the slave identification design is given in Figure 6.6.2. The slave's action, clearing the flip flop after the definition has been accepted, is indicated by the FLIPCLR line (clear flip-flop). The use of Schmitt trigger inverters was thought to be helpful in such a possible noisy environment.

### 6.6.3 Slave Definition Handshake

As mentioned before, the master sets the $\overline{H O L D}$ line as soon as it defines the slave. The acceptance of the definition by the slave is indicated by the HOLDA (hold acknowledge) line.

The handshake procedure is accomplished through device 7D shown in Figure 6.5.3. A more detailed view is shown in Figure 6.6.3. Looking from the master's side, it is seen that the HOLDSET line is self enabling. As soon as it is set low by the master, it will go through gate 4 on device 7 D and will set the $\overline{H O L D}$. $B$ line, LOW, on the system bus.

The HOLD.B line, after inversion, is received by gate 1 on the same device. This gate is controlled by the slave's ENAHOLD line. It is quite clear that the $\overline{H O L D}$ is set LOW by the $\overline{\mathrm{HOLD}} . \mathrm{B}$ line only when enabled by a low level on ENAHOLD line.

After the $\overline{H O L D}$ line goes LOW and a maximum of 3 consecutive memory cycles have been executed, the slave goes into the hold state and the HOLDA line is set to HIGH. This line is self enabling on device 7D and, as soon as it is set to high, it makes the HOLDA.B


Figure 6.6.2. The Slave Identification Timing Diagram


Figure 6.6.3 The Handshake Connections through Device 7D
line on the system bus go LOW.
The HOLDA.B line after inversion goes through gate 3 on device 7D and is received by the master on RECHOLDA line. This is the signal the master has been waiting for to complete the handshake of the slave definition procedure. It indicates to the master that the slave is in a hold state and that communication can take place by direct memory access mode.

A timing diagram for this handshake procedure is shown in Figure 6.6.4. The first line on this figure, the INTSLAV line, indicates that the slave has been correctly identified by the master through the master-slave identification procedure.

Events marked $M$ and $S$ are initiated respectively by the MASTER and SLAVE. Also shown is the sequence of events that occurs after the master has finished communication and resets the HOLDSET line. As soon as $\overline{H O L D}$ is reset (HIGH) the slave returns to normal operation.

### 6.6.4 Master-Slave Communication

When the master receives the hold acknowledge signal indicating that the slave is in a hold state, communication can take place between the master and the slave's communication memory area.

It is appropriate to discuss first what happens at theslave's side of the communication link and then the master's.

## 1. Relating to the Slave

When the slave enters the hold state, all its communication buffers are set accordingly. The direction on the address and control buffers, devices 4 A and 3A, in Figure 6.5.2 and 6D in $F$ Figure 6.5.3, is set to be from the system bus to the internal bus. The reason for this has already been discussed. The HOLDA line is used for this purpose. After inversion, it sets the proper direction on the buffers. The enabling action on the same buffers is controlled by the HOLDA and $\overline{H O L D}$ lines. When HOLD goes low, indicating that the module is a slave, it enables the HOLDA line to control the ENA line. As soon as HOLDA goes high (hold state) the buffers are enabled. It is clear that as far as the HOLD line remains low, that is, the module is a slave, the address buffers and the control buffer are enabled.


Figure 6.6.4 Timing for the Complete Handshake Procedure

As can be seen on Figure 6.5 .2 and has already been mentioned, the four most significant lines on the system address bus do not convey information to the slave's internal bus. The information on these lines for the internal bus is generated by the slave module itself, through the commication address area generator.

This is accomplished by using an octal buffer with three-state outputs. The device, a SN 74LS241, is identified by code 5A on Figure 6.5.2. In the disabled state (input 2G is low) its data outputs are in high impedance state. When the input 2G is set to high the data inputs (2Ai) are enabled through to data outputs (2Yi).

When the slave enters the hold state (HOLDA - high) after being asked for ( $\overline{\text { HOLD }}$ - Low), the input (2G) to device 5A goes to a high state and the four most significant lines on the slave's internal address bus are set accordingly to the data inputs (2Ai) to device 5A.

The choice of where the 2 K words communication block will 1ie on the slave's memory map is made through the data inputs to device 5A. The connections shown in Figure 6.5.2, with all data inputs (2A1, 2A2, 2A3, 2A4) made HIGH, indicates that the communication block lies between addresses FOOO and FFFE.

The buffers on the data lines are controlled by lines DBIN and HOLDA. When the master wants to read from the slave's memory, data flows from the slave's internal bus to the system bus. When the master is writing to the slave's memory, the flow is from system bus to internal bus. The logic necessary is achieved by an exclusive-or (2C - Figure 6.5.3) followed by an inverter (3B). Table 6.6.2 gives the logical levels and the direction of data flow through the buffers when HOLDA line is high or the module is a slave.

| HOLDA | DBIN | DATDIR | DATA FLOW DIRECTION |
| :---: | :---: | :---: | :---: |
| HIGH | HIGH | HIGH | Internal to System Bus |
| HIGH | LOW | LOW | System to Internal Bus |

Table 6.6.2
2. Relating to Master

After receiving the hold acknowledge signal (RECHOLDA) the master knows that everything is ready on the slave's side for the communication to take place. In order to avoid any loss of time, the communication should take place as soon as the RECHOLDA signal is received. Software design must cover this requirement.

When the master refers to a memory location, correspondant to a certain slave, the most significant four lines of the internal address bus will contain the slave's identification code. The decoder 1A on Figure 6.5 .2 receives this code and, if it is a possible slave address, will recognise and through output $W$ will enable the address data and control buffers on the master module, through the ENA line, see Figures 6.5.2 and 6.5.3. It is quite clear that the same decoder used for slave identification purposes is used now for switching the buffers to an external referenced memory. The difference is that, as far as no commnication attempt is maḍe to the memories reserved for identification purposes, the buffers (11A), on the slave identification lines, will remain disabled. This should be kept in mind when software is written.

After the buffers on the master module are enabled, the slave's memory communication area looks like an extension of the master's own memory. There is a certain timing restriction related to the memory access time on the slave module which we will discuss later on.

Table 6.6.3 gives the logical levels and the direction of data flow through the data buffers when HOLDA line is low, or the module is a master.

| HOLDA | DBIN | DATDIR | DATA FLOW DIRECTION |
| :---: | :---: | :---: | :--- |
| LOW | HIGH | LOW | System to Internal Bus |
| LOW | LOW | HIGH | Internal to System Bus |

Table 6.6.3

Tables 6.6.3 and 6.6 .2 complement each other in relation to DBIN as it should be.

### 6.7 THE TIMING RELATIONSHIP DIAGRAMS

The timing diagrams for the complete comminication system are not able to be put together in an easy way to understand. We will go by parts and hope the ideas will get clearer as we proceed.

The first timing diagram, given in Figure 6.7.1, indicates what happens on the slave module just after it enables the $\overline{H O L D} . B$ signal in by the use of ENAHOLD. The maximum latency time between the hold request and the hold acknowledge, is equal to three clock cycles plus three memory cycles. The minimum latency time is equal to one clock cycle. At 3 MHz and no wait cycles, the maximum time is nine clock cycles or 3 microseconds and the minimum time of one clock cycle is 333 nanoseconds. This indicates that the RECHOLDA line on the master's side should be used as an interrupt line if time is not to be lost. At the other end of this timing diagram the delay between the slave's release by $\overline{H O L D}$ and the slave's return to normal operation is clearly shown. This delay is mainly dictated by the microprocessor's operation which requires two $\phi 1$ clock cycles for releasing of HOLDA. The maximum of three clock cycles is meant for the non-synchronisation of $\overline{H O L D}$ and $\phi 1$. The delay of three clock cycles is not much of a problem because the master will take at least 10 clock cycles to execute the next instruction. In the meanwhile, within three clock cycles, the HOLDA line is released but the ENA $\overline{H O L D}$ line is still set to enable HOLD and it takes the slave at least 12 clock cycles to reset this line. While this line is not reset the $\overline{H O L D}$ line on the slave's internal bus is in direct communcation with the system bus line $\overline{H O L D} . \mathrm{B}$ and, being so, the buffers on the slave module are still under direct command of this line. Depending on the communication software used, this will not be a serious drawback. . The master, after communication is finished, would check the RECHOLDA line and would not proceed until this line indicated that the slave had resumed operation. It would take only an extra simple instruction or 12 clock cycles. If the slave's first executed instruction after a hold state is a reset of ENAMOLD line this instruction would be executed practically while the master is checking the RECHOLDA line.

Based on this approach and trying to keep the design with the minimum number of $I C^{\prime} s$ it was decided not to implement the J-K flip flop disabling logic on lines $\overline{\mathrm{HOLD}}$ and ENA $\overline{H O L D}$ as shown in Figure 6.7 .2 with


Figure 6.7.1. The Handshake Timing Diagram
its timing diagram. When the master resets the HOLD.B line on the system bus it clocks the J-K flip flop through $\overline{\text { HOLD }}$ and disables the NAND gate maintaining HOLD high independently of $\overline{H O L D}$. . The state of the flip flop changes again when the slave resets the ENA $\overline{H O L D}$ but the NAND gate remains disabled until a new slave definition is required. The implementation of this logic is recommended on a multi-microcomputer configuration with more than two microcomputer modules. In our case, as we only have two microcmputers, there are no drawbacks in not having it implemented and that is what we have done.

The timing diagrams for the communication phase are shown in Figure 6.7.3. Read and write cycles are shown for completeness. As these diagrams are quite self explanatory, the important thing to discuss is the time delay introduced by all the logic involved in the process related to memory access time.

When communication is made, at one end of the communication link there is a microprocessor and at the other a memory device. The microprocessor timing is controlled by its clock and anything to be read or written should be available at the right time. If the speed of any device is not compatible with the microprocessor's own speed, there is the possibility of using the READY input line, already discussed.

We decided not to use wait states, for there is no reason to believe that the 2 K words of memory on the communication block could not be implemented using fast memory devices compatible with the system bus speed requirements. The other point favouring this decision is the question of price: slow and fast devices are falling dramatically in price and there is always a tendency to have bigger reductions in price on the faster devices due to the market demand for them.

The delay introduced on the communication process is made of many different parts. The first is the delay on the generation of the ENA line on the master module. The second is the delay on the system bus buffers. The third is the delay on the slave's address decoder. Table 6.7 .1 shows all the IC's involved in the process with respective delay intervals.


Figure 6.7.2 The HOLDA Delay Logic Protection


|  |  | MASTER ENA GENERATION |  |  |  |  | BUS <br> BUFFERS | $\begin{gathered} \text { SLAVE } \\ \text { DECODING } \end{gathered}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Device |  | LS243 | LS08 | 150 | LSO4 | LS08 | LS245 | LS245 | LS20 |
| Delay | typ | 12 | 10 | 21 | 10 | 10 | 12 | 12 | 10 |
|  | max | 18 | 20 | 30 | 15 | 20 | 18 | 18 | 15 |
| Qty |  | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 |
| Delay | typ | 12 | 10 | 21 | 10 | 10 | 24 | 12 | 10 |
|  | max | 18 | 20 | 30 | 15 | 20 | 36 | 18 | 15 |
| Total | type | 109 ns |  |  |  |  |  |  |  |
| Delay | max |  |  |  | 172 |  |  |  |  |

Table 6.7.1

It takes typically 109 ns and a maximum of 172 ns for the address generated by the master to get to the slave's communication area.

With a system clock frequency of 3 MHz , figure 6.7 .4 gives the timing between the clock phases and other microprocessors' inputs and outputs. Using Figures 6.7 .3 and 6.7 .4 we can say that:

1. Master Reading: when rise and fall times for the signals and set up time for the data are computed, the memory device should have an access time of about 490 ns or less from valid address.
2. Master Writing: counting rise and fall times plus data hold time the cycle time should be less than about 600 ns from valid address.

Using the timing restrictions imposed by the microprocessor and the delay introduced on the communication link, we can say that for the worst case condition and reliable read and write cycles, the memory device must have an access time of less than 318 ns from valid address if the maximum delay on the link is assumed. It is clear that the memory device used on the memory expansion board working under its maximum access time from valid address ( 450 ns ) would not suit our requirements. With the experience acquired on the use of these devices, it is known that they normally work much faster than the expected worst conditions. Under the same view, the delay in the communication link is expected to be the typical one or 109 ns .


|  | PARAMETER | MIN | NOM | MAX | UNIT |
| :---: | :---: | :---: | :---: | :---: | :---: |
| ${ }^{\mathrm{c}} \mathrm{c}(\mathrm{o})$ | Clock cycle time | 0.3 | 0.333 | 0.5 | $\mu s$ |
| ripl | Clock rise time | 10 | 12 |  | ns |
| tf(0) | Clock fall time | 10 | 12 |  | ns |
| $\mathrm{t}_{\mathrm{w}}(0)$ | Pulse width, any clock high | 40 | 45 | 100 | ns |
| 101 L .62 H | Delay time, clock 1 low to clock 2 high (time between clock pulses) | 0 | 5 |  | ns |
| ! 2L, ${ }^{\text {3H}}$ | Delay time, clock 2 low to clock 3 high (time between clock pulses) | 0 | 5 |  | ns |
| ¢ $03 \mathrm{~L}, 04 \mathrm{H}$ | Delay time, clock 3 low to clock 4 high (time between clock pulses) | 0 | 5 |  | ns |
| $1.94 \mathrm{~L}, 01 \mathrm{H}$ | Delay time, clock 4 low to clock 1 high (time between clock pulses) | 0 | 5 |  | ns |
| \% $\phi 1 \mathrm{H}, \phi 2 \mathrm{H}$ | Delay time. clock 1 high to clock 2 high (time between leading edges) | 70 | 80 |  | $n s$ |
| ${ }^{1} \bigcirc 2 \mathrm{H}, 93 \mathrm{H}$ | Delay time, clock 2 high to clock 3 high (time between leading edges) | 70 | 80 |  | ns |
| ${ }^{1} \phi 3 \mathrm{H}, \phi 4 \mathrm{H}$ | Delay time, clock 3 high to clock 4 high (time between leading edges) | 70 | 80 |  | ns |
| $\mathrm{t}_{6} 4 \mathrm{H}$ 人 1 H | Delay time, clock 4 high to clock 1 high (time beiween leading edges) | 70 | 80 |  | ns |
| $t_{\text {su }}$ | Data or control setup time before clock 1 | 30 |  |  | ns |
| th | Data hold time after clock 1 | 10 |  |  | ns |


| PARAMETER | TEST CONDITIONS | MIN | TYP | MAX |
| :---: | :---: | :---: | :---: | :---: |
| TPLH Or TPHL Propagation delay time, clocks to outputs | $\mathrm{C}_{\mathrm{L}}=200 \mathrm{pF}$ | 20 | 40 | ns |

Figure 6.7.4 TMS 9900 Microprocessor Signal Timing

Using this delay, the memory access time should be less than 391 ns . This value is only 60 ns less than the 450 ns for maximum access time and we assume, at this point, that the memory devices will be capable of reaching such speed. If not, we will have to change to faster memories or use wait states.

## CHAPTER 7

THE MULTI-MICROCOMPUTER'S HARDWARE IMPLEMENTATION AND SOFTWARE COMMUNICATION DESIGN

### 7.1 INTRODUCTION

This chapter is concerned with the hardware implementation of the logical design, treated in Chapter 6, and the necessary communication software.

All the aspects relating to the hardware involved are treated in Section 7.2 .

For the communication software, the memory expansion board plays a key role. Section 7.3 deals with the testing of this important element.

In Section 7.4, the interconnections between the system bus interfaces and the microcomputer are discussed and chosen. The internal bus separation provided by the memory expansion board and the reset action provided by the system bus are discussed.

Section 7.5 describes the power supply available to the system.
As the system bus is formed by well defined sectors, testing is achieved by writing software that works through these sectors as independently as possible. Section 7.6 is concerned with this software and the system bus testing.

The general communication software for the multi-microcomputer system has its design given in Section 7.7. Several concepts are discussed there, such as transfer vector and single and global message. Transparency to the system user is made possible through the use of word and block transfer modes.

### 7.2 THE HARDWARE DESIGN IMPLEMENTATION

### 7.2.1 Board Manufacturing Process

The implementation of the logical design to the integrated circuit board level has gone through several stages.

In the first stage, the question of wire wrapped or printed circuit board arises. The wire wrapped version gives more flexibility in a prototype system like ours. The use of proper wire wrapping connectors and tools ensures no problems relating to bad connections.

Any modifications on the board can be achieved with little effort and the increase on the number of integrated circuits can be easily accounted for if physical space is available on board. This possibility had to be dropped from the start due to the nonavailability in the Department of wire wrapping connectors for integrated circuits. For some time, the Department has provided reasonable facilities for printed circuit board manufacturing and all the connectors and tools available are related to it.

The printed circuit board manufacturing facility includes three main baths: the photo resistive coating bath, the developer bath and the etching bath. After the board is coated with the photo resistive material, it is subjected to an ultra-violet light source and the desired mask is transferred to the coated board. When the board goes to the developer bath, the ultra-violet light affected areas are chemically transformed and can be washed away with running water. Then the board goes to the etching bath and the ferric chloride attacks and removes all the metallic areas non-protected by the resistive coating. After a good wash with rumning water, the board is ready to go to the next step, that is the assembly process.

The first problem in the process appears in the photo resistive coating. The board's metallic cover must be thoroughly cleaned to make sure the coating is effective. Any dust particle will break the coating layer and the board has to be cleaned again. To ensure the metallic surface is really clean one has to rub on polishing powder until it is spotless. After removing all the remaining polishing powder and drying, the board can go to the coating bath. The actual coating takes 10 minutes but the coating layer takes 24 hours to dry out before it can be used again.

The mask for each different board has of course to be manufactured and we will talk about it later on. The time under the ultra-violet light can be adjusted through a timer and this is the only automated part in the process. After some trial runs the time under the light for the particular resistive material is set without much difficulty.

The second problem in the process is the one related to the developer bath. The time it takes for the developer to act varies
and one has to keep a close eye on the process. Even working very carefully, the developed areas sometimes do not come very clear and the process has to be started again from scratch. The third problem in the process appears in the etching bath. Even with a good developed masking pattern, for reasons not well understood but probably due to faulty board's metallic cover, the bath is not able to clean all the unprotected areas. The idea of leaving more time in the bath is just a remedy for, as long as the board remains in the bath, the ferric chloride is acting, even on the protected areas, and at the end destroys thin connection lines and the process has to start again from scratch.

When a good board comes out of the photochemical process, the next step is to cut it to size and gold plate the edge connector side. This measure is really necessary to ensure that there will be no change in the contact characteristics due to metal surface changes.

The drilling of the board is the next step in the process. To this end there is a special high speed mini drilling machine. Using whole carbide helical drills, there is no problem making holes, except for the occasional tool break down and the reminder that tools are expensive.

There is no possibility in the Department to plate through holes and the necessity for a double sided board means that through pins must be used and extra holes provided for them.

### 7.2.2 Card Size and Edge Connectors

The baths of the photo chemical process already mentioned are all of the same size and the maximum board size, which can be manufactured without extra problems, is $140 \times 230 \mathrm{~mm}$.

The connector available in stock to suit boards within this size is the 43 way double sided edge connector with 0.1 in pitch and position 37 fitted with polarising key (wire wrapping connector). This connector limits the connector edge in the board to 114 mm and the total number of connections to $84(42+42)$ in a double sided board. In order to be able to fit the connector properly
into the racking system, and providing space for expansion, the final board size has to be $114 \times 202 \mathrm{~mm}$. These dimensions are compatible with the maximum possible to be manufactured.

### 7.2.3 The Functional Distribution

Looking back to Figures 6.5.2 and 6.5.3, it is noticed that for the implementation of the total logic hardware design, it is necessary to have available at least 99 pins ( $22+15+34+28$ ). This of course cannot be achieved using just one board with 82 $(41+41)$ available pins.

With two boards it is possible to implement the design and it was done so. It was decided to have implemented in each board all the possible set of logical decisions associated with each sector of the system bus.

The final arrangement is the same as shown in Figures 6.5.2 and 6.5.3 where the functions associated with the address system bus, correspondent to Figure 6.5.2, are grouped in the first board and the other functions, correspondent to Figure 6.5.3, in the second board.

### 7.2.4 The Lay-Out Design and the Final Boards

The board associated with the address system bus is called from now on the address bus board. Its manufacturing mask is shown in Figure 7.2.1 and its pin-signal assignment in Table 7.2.1. The side with the integrated circuits in it has odd numbered pins and the locating key is positioned at pin 73.

The second board which we call data, control and handshake buses board or just data bus board, has its mask shown in Figure 7.2.2 looking from the component side. The pin-signal assignment for this board is shown in Table 7.2 .2 with odd pin numbers for the component side.

The boards in their final configuration are shown in Figures 7.2.3 and 7.2.4.



| ADDRESS BUS BOARD PIN-SIGNAL ASSIGNMENT |  |  |  |
| :---: | :--- | :--- | :--- |
| PIN | SIGNAL | PIN | SIGNAL |
| 1 | GND | 50 | $\overline{\text { MEMEN. BF }}$ |
| 2 | GND | 52 | HOLD |
| 3 | +5V | 54 | HOLDA |
| 4 | +5V | 56 | A14 |
| 10 | INTSLA | 58 | A13 |
| 12 | FLIPCLR | 60 | Al2 |
| 14 | ENA (To Data) | 62 | A11 |
| 16 | WE.BF | 64 | Al0 |
| 18 | A14.B | 66 | A9 |
| 20 | A13.B | 68 | A8 |
| 22 | A12.B | 70 | A7 |
| 24 | A11.B | 72 | A6 |
| 26 | A10.B | 76 | A5 |
| 28 | A9.B | 78 | A4 |
| 30 | A8.B | 80 | A3 |
| 32 | A7.B | 82 | A2 |
| 34 | A0.B | 84 | A1 |
| 36 | A1.B | 86 | A0 |
| 38 | A2.B |  |  |
| 40 | A3.B |  |  |
| 42 | A4.B |  |  |
| 44 | A5.B |  |  |
| 46 | A6.B |  |  |
|  |  |  |  |

Table 7.2.1


FIGURE 7.2.2 MANUFACTURING MASK FOR DATA BUS BOARD


| DATA CONTROL AND HANDSHAKE BUSES PIN-SIGNAL ASSIGNMENT |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| PIN | SIGNAL | PIN | SIGNAL | PIN | SIGNAL |
| 1 | GND | 69 | INTMST | 2 | HOLD. B |
| 3 | +5v | 77 | SETRACKO | 4 | HOLDA.B |
| 5 | HOLDSET | 79 | RBUSBUSY | 6 | $\overline{\text { ¢1. }}$ B |
| 7 | ENAMOLD | 81 | RINTCTL | 8 | DBIN. ${ }^{\text {B }}$ |
| 9 | HOLDA | 85 | RECPRES | 10 | MEMEN. ${ }^{\text {B }}$ |
| 11 | HOLD | 86 | SITCTL | 12 | $\overline{\text { WE. }} \mathrm{B}$ |
| 13 | RECHOLDA | 24 | READY. BF | 14 | READY. ${ }^{\text {b }}$ |
| 35 | D7 | 26 | $\overline{\text { WE. }}$ BF | 38 | Do.B |
| 37 | D6 | 28 | MEMEN. ${ }^{\text {BF }}$ | 40 | DI.B |
| 39 | D5 | 30 | DBIN. BF | 42 | D2.B |
| 41 | D4 | 32 | $\overline{\text { ¢1. }} \mathrm{BF}$ | 44 | D3.B |
| 43 | D3 |  |  | 46 | D4.B |
| 45 | D2 | 36 | ENA | 48 | D5.B |
| 47 | D1 |  | (From Address) | 50 | D6.B |
| 49 | D0 |  |  | 52 | D7.B |
| 53 | D15 |  |  | 54 | D8.B |
| 55 | D14 |  |  | 56 | D9.B |
| 57 | D13 |  |  | 58 | D10.B |
| 59 | D12 | 72 | RACKO | 60 | D11.B |
| 61 | D11 | 78 | тACKO | 62 | D12.B |
| 63 | D10 | 82 | BUSBUSY | 64 | D13.B |
| 65 | D9 | 83 | $\widehat{\text { PRES }}$ | 66 | D14.B |
| 67 | D8 | 84 | INTCTL | 68 | D15.B |

Table 7.2.2

### 7.2.5 Line Termination for the System Bus

The termination of all lines in the system bus is a necessary measure when all the possible ten modules are connected to it. In this case the bus lines can extend up to 600 mm and the analogy to transmission lines has to be considered. Any of the already mentioned termination techniques can be used and the $5-100$ bus approach (75) is the more appropriate in a conscious power dissipation design.

In an ideal situation the system bus would be implemented as a back plane bus and the terminations would be set on this plane at one or both ends for each uni or bi-directional line respectively. In our case, it is not necessary and there is not really any possibility, as is shown later on, to implement such a back plane bus. As we have only two modules, at the moment, to be connected to the system bus, it is simpler and cheaper to wire up the bus lines. The terminations on the lines driven by other than open-collector drivers are not executed, for there is no necessity as the final bus length is less than 150 mm .

For the open-collector driven lines, the necessary pull-up resistors are implemented as line termination Thevenin equivalent network. Appendix 7.2.1 shows how the equivalent network is found for the different lines and the implications of the choice of line drivers.

### 7.2.6 The Racking System and. Board Arrangement

With the memory expansion and the interface boards ready we were able to wire wrap all the necessary communication links.

The position of the microcomputer boards on the racking system was not easily chosen. The horizontal position was chosen finally based on the idea that the emulation cable, from the Texas development system, could reach the microcomputer on the top, avoiding the use of extension cards, at least for this board. The 7.5 in length in extension board greatly increases the length of the transmission lines and it should be avoided.

The stacking of the microcomputer boards in the horizontal plane is not of course the most favourable one for natural cooling. For our system with two boards this was thought to be not damaging
and, if the number of modules increases, it is quite certain that forced cooling will be necessary and then horizontal or vertical positions will have practically the same effects.

The microcomputer boards being on the top of the racking system, in the horizontal plane, do not leave much space for all the other boards to be on the top, so they are placed on the bottom part of the rack. Figure7.2.5 shows the racking system with the two microcomputer boards on the top and their correspondent memory expansion and bus communication boards (2) on the bottom.

At this moment the drawback of not being able to manufacture printed circuit boards in bigger sizes appears very clearly. The distance between the microcomputers and the interface boards is practically over the limit for a transmission line without proper termination. We just accepted these facts and hoped the system would behave properly.

### 7.3 MEMORY EXPANSION BOARD - GENERAL TEST

The first links to be wired up were between the microcomputers and the memory expansion boards. Having in mind that this memory board is the key element in the systems communication process, we must first check for any malfunction in the manufactured boards. Table 7.3 gives the pin connections and assignment for these links.

The memory expansion board was tested using the functions which show up in the general diagram given in Figure 7.3.1 and are discussed below. The first function is implemented by the decoder. The control lines and part of the address lines go to the decoder and, if any memory block under its control is addressed, the block must be enabled and no other one. The second function is achieved by the remaining address lines. These lines must address univocally each independent memory cell in each block. The third function is executed by the data bus lines. Data must be transferred to and from the memory cells without error. The fourth and last function is executed by the memory cell itself. The memory cell must be able to store the data without any change.

The block diagram for the memory test program is shown in Figure 7.3.2 and its implementation in assembly level language is given in Appendix 7.3. The message display is achieved using one of the TIBUG monitor extended operation routines (XOPs).

## -

(1) MICROCOMPUTER I
(2) MICROCOMPUTER 2
(3) INPUT/OUTPUT

BOARD

(4), (7) ADDRESS BUS

BOARD
$(5),(8)$ DATA BUS
BOARD
(6). (9) MEMORY

EXPANSION
BOARD


The global error message indicates one of the following possible source of errors:

1. no memory chip on the addressed location
2. decoder logic is not correct or is faulty
3. short-circuit on the enable lines
4. short-circuit on the remaining address lines
5. bad contact somewhere

Using the TIBUG monitor, it is possible to display the values stored on memory and verify where the global test failed (the memory does not contain its own address). Then using the source of errors mentioned above, find out which one is affecting the system.

The AAAA test fail message indicates that one of the memory cells cannot accept the binary pattern 1010101010101010. There are two main reasons for this:

1. one of the memory chips is faulty
2. there is a short-circuit on the data lines

The 5555 test fail message indicates that one of the memory cells cannot accept the binary pattern 0101010101010101 . The reasons for this are the same as explained above for the AAAA test.

The test program was loaded using the Tibug load facility and the memory expansion tested. As was mentioned before, due to the very compact design used for the board (Appendix 6.3), many broken and short-circuited lines were found. The test program helped to find where the simple problem was or at least indicated the possible cause for more complex ones.

Using the test program and the emulation and tracing facilities available on the Texas development system, all the faults in the memory expansion board were corrected and we proceeded to the next stages of assembling and testing.

| INTERCONNECTIONS <br> MICROCOMPUTER AND MEMORY EXPANSION BOARD |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| PIN |  | SIGNAL | PIN |  | SIGNAL |
| Memory | Micro |  | Memory | Micro |  |
| 17 | 48 | D15 | 48 | 57 | AO |
| 19 | 47 | D14 | 46 | 58 | A1 |
| 21 | 46 | D13 | 44 | 59 | A2 |
| 23 | 45 | D12 | 42 | 60 | A3 |
| 25 | 44 | D11 | 38 | 61 | A4 |
| 27 | 43 | D10 | 36 | 62 | A5 |
| 29 | 42 | D9 | 40 | 63 | A6 |
| 31 | 41 | D8 | 34 | 64 | A7 |
| 49 | 40 | D7 | 35 | 65 | A8 |
| 51 | 39 | D6 | 37 | 66 | A9 |
| 53 | 38 | D5 | 39 | 67 | A10 |
| 55 | 37 | D4 | 41 | 68 | All |
| 57 | 36 | D3 | 43 | 69 | A12 |
| 59 | 35 | D2 | 45 | 70 | A13 |
| 61 | 34 | D1 | 47 | 71 | A14 |
| 63 | 33 | D0 |  |  |  |
| 2 | 90 | READY | 6 | 78 | $\overline{\text { WE }}$ |
| 4 | 22 | $\overline{\text { ¢ } 1}$ | 8 | 80 | $\overline{\text { MEMEN }}$ |
| 10 | 82 | DBIN | 12 | 86 | HoLDA |

Table 7.3


Figure 7.3.1 $\frac{\text { Memory Board }- \text { General }}{\underline{\text { Diagram }}}$


### 7.4 THE SYSTEM BUS AND THE MICROCOMPUTER

### 7.4.1 Interrupt Controlled Arbitration Link and the Microcomputer Connections

As can be seen on Figures 6.4.2 and 6.5.3, the interrupt controlled arbitration link requires in total 6 communication lines to any microcomputer. There is no distinction made here between controller, master or slave module.

According to the direction of flow of information in each line, we can separate them in input or output lines when the information is received or transmitted to the microcomputer. This separation allows us to write Table 7.4.1.


## Table 7.4.1

Among the input lines there are only two which really require connection to microcomputer interrupt lines to have the fastest possible operation. They are the INTMST and the RINTCTL lines. All the other lines will be connected to CRU I/O ports through the TMS 9901 which has already been described.

The priorities to be associated to each interrupt line depend on the final system configuration and how important the communication process is when compared to other ones. It is quite clear that, in a multi-microcomputer real-time control environment for certain modules, there will be processes with much higher priority than the communication one. On the other hand, for other modules, or within a defined task, the communication process will be vital. It is assumed here that the final choice will be made according to the actual environment and, at this moment, we are only interested in checking the hardware design and writing a communication software which will be independent of associated priorities.

Just for convenience, it was decided to associate INTMST to interrupt level 4 and RINTCTL to interrupt level 5.

The final interconnections between interrupt controller arbitration link lines and the microcomputer and the associated software design are shown in the next sections.

### 7.4.2 Slave Definition Process and the Microcomputer

The slave definition process, as was mentioned before, and which hardware implementation is indicated in Figure 6.5.2, requires one input and one output line to the microcomputer when the correspondent module is acting as a slave. Table 7.4 .2 shows these lines and the direction associated to them.

| LINE |  |
| :---: | :---: |
| Output | Input |
| FLIPCLR | INTSLAV |

Table 7.4.2

The INTSLAV line, which name already gives the indication, is going to be connected to an interrupt line. In our case interrupt level 3 is used for this line.

The FLIPCLR line is connected to a CRU output line through the TMS 9901 interface chip.

### 7.4.3 The Handshake Connections

The master-slave handshake procedure, which hardware implementation is indicated in Figure 6.5.3, requires 2 output and 1 input line. The direction associated with each one is indicated in Table 7.4.3.

| LINE |  |
| :---: | :---: |
| Output | Input |
| $\overline{\text { HOLDSET }}$ | RECHOLDA |
| ENAHOLD |  |

Table 7.4.3

The $\overline{H O L D S E T}$ and ENA $\overline{H O L D}$ lines will be connected to CRU output lines.

The RECHOLDA line can be connected either to a CRU or an interrupt line. In a general multi-microcomputer environment it should be connected to an interrupt line. The reason for this is that the master would not have to check this line to receive the $\overline{\mathrm{HOLD}}$ acknowledge signal from the slave. It would come normally as an interrupt. In our case, for the two modules environment, this line can be connected to a normal CRU input line. The master will have to keep checking it to receive the acknowledgement signal. For our environment we will see that for most cases this set up is almost as fast as it would be if the interrupt option were used. The interrupt context switch takes twice the average instruction execution time.

### 7.4.4 CRU and Interrupt Lines - Final Definition

The interconnections for the $\mathrm{I} / \mathrm{O}$ and interrupt lines described above are shown in Table 7.4.4. The CRU bit number associated with each CRU line is also shown for ease of reference when software writing. The RECHOLDA line indicates both options for CRU and interrupt operation.

### 7.4.5 Reset Action through the System Bus

The signal lines SETPRES and RECPRES, shown in Figure 6.5.3 and Table 7.4.4, originate from the system bus line $\overline{\text { PRES }}$ and they provide the means to reset all themodules on the system through a software command by the controller module. When line SETPRES is set LOW by the controller, a microprocessor RESET action is initiated in all other modules connected to the system bus. The line must be maintained LOW for at least three clock cycles and when it is released a level-zero interrupt sequence takes place. Means should beprovided for the modules to initiate the correct sequence.

|  | INTERCONNECTIONS |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Interface Boards |  | Microcomputer |  |  |  |
|  | Signal | Pin | Pin | $\begin{gathered} \text { Connec- } \\ \text { tor } \\ \hline \end{gathered}$ | $\begin{aligned} & \text { CRU } \\ & \text { bit } \end{aligned}$ | Signal |
|  | SITCTL <br> RINTCTL <br> SBUSBUSY <br> RBUSBUSY <br> INTMST <br> SETRACKO <br> HOLDSET <br> ENAMOLD <br> RECHOLDA <br> SETPRES <br> REC $\overline{P R E S}$ | 86 81 79 80 69 77 5 7 13 85 83 |  | P4 <br> P1 <br> P4 <br> P4 <br> P1 <br> P4 <br> P4 <br> P4 <br> P4 <br> P4 <br> P1 | 19 <br> - <br> 21 <br> 24 <br> - <br> 20 <br> 17 <br> 18 <br> 22 <br> 23 | $\begin{gathered} \frac{\mathrm{P} 3}{} \\ \overline{\text { INT5 }} \\ \text { P5 } \\ \text { P8 } \\ \overline{\text { INT4 }} \\ \text { P4 } \\ \text { P1 } \\ \text { P2 } \\ \text { P6( } \overline{\text { INT6 }}) \\ \overline{\text { P7 }} \\ \overline{\text { PRES }} \end{gathered}$ |
|  |  |  |  |  |  |  |
|  | INTSLA FLIPCLR | $\begin{aligned} & 10 \\ & 12 \end{aligned}$ | $\begin{aligned} & 15 \\ & 20 \end{aligned}$ | $\begin{aligned} & \text { P1 } \\ & \text { P4 } \end{aligned}$ | 16 | $\begin{gathered} \overline{\text { INT3 }} \\ \text { PO } \end{gathered}$ |

Table 7.4.4

### 7.4.6 The Internal Bus Separation

As mentioned in Chapter 6, the control lines on the internal bus are split in two parts: one that serves the communication global memory area and the other that serves the module's remaining parts. The separation is made by a buffer controlled by the HOLDA line. The control lines coming from the microcomputer maintain their original names. Between the separation buffer, memory cells and system bus buffer, they receive a suffix . BF (BUFFER) as indicated in Figure 7.4.1


Figure 7.4.1 Internal Bus Separation

### 7.4.7 Final Interconnections and the System Bus

With the explanation given above and the help of Figures 6.4.6 and 7.4.1, we are now able to make the remaining internal interconnections for each module. Tables 7.4 .5 and 7.4 .6 show the pinassignment for these interconnections. The connections to the memory expansion board are repeated, when necessary, to show that there is a direct link between the boards and due to the racking system configuration, the wires come from the memory board. The system bus can now have its lines connected between our two modules.

The module correspondent to the microcomputer on top of the racking system is identified from now on as MODULE 1 and the other module as MODULE 2. The system bus, in our case, extends between the address and data bus boards of each module. Table 7.4 .7 shows the pin connections for the system bus. It is clear that there are 42 lines in total.

|  | INTERCONNECTIONS |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Microcomputer Board |  | Memory Expansion Board |  | Address Bus Board |  | Data Bus Board |  |
|  | Pin | Signal | Pin | Signal | Pin | Signal | Pin | Signal |
|  | $\begin{aligned} & 86 \\ & 92 \end{aligned}$ | $\begin{aligned} & \text { HOLDA } \\ & \overline{\text { HOLD }} \end{aligned}$ | $\begin{array}{r} 9 \\ 15 \\ 11 \\ 2 \\ 13 \\ 12 \end{array}$ | $\overline{\text { MEMEN. }} \mathrm{BF}$ $\overline{\mathrm{WE}} . \mathrm{BF}$ $\mathrm{DBIN} . \mathrm{BF}$ $\mathrm{READY} . \mathrm{BF}$ $\overline{\phi 1} . \mathrm{BF}$ HOLDA | 50 <br> 54 <br> 52 | MEMEN.BF <br> $\overline{\text { WE }} . \mathrm{BF}$ <br>  <br> HOLDA <br> $\overline{\text { HOLD }}$ | $\begin{array}{r} 28 \\ 26 \\ 30 \\ 24 \\ 32 \\ 9 \\ 11 \end{array}$ | $\overline{\text { MEMEN }} \mathrm{BF}$ $\overline{\mathrm{WE}} \cdot \mathrm{BF}$ DBIN.BF READY.BF $\overline{\phi 1} . \mathrm{BF}$ HOLDA $\overline{\text { HOLD }}$ |
|  |  |  |  |  | 14 | ENA | 36 | ENA |
|  | 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 | AO <br> Al <br> A2 <br> A3 <br> A4 <br> A5 <br> A6 <br> A7 <br> A8 <br> A9 <br> A10 <br> All <br> A12 <br> A13 <br> A14 | 48 46 44 42 38 36 40 34 35 37 39 41 43 45 47 | A0 <br> A1 <br> A2 <br> A3 <br> A4 <br> A5 <br> A6 <br> A7 <br> A8 <br> A9 <br> A10 <br> Al1 <br> A12 <br> A13 <br> A14 | 86 84 82 80 78 76 72 70 68 66 64 62 60 58 56 | A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 |  |  |

Table 7.4.5

|  | INTERCONNECTIONS |  |  |
| :---: | :---: | :---: | :---: |
|  | Microcomputer Board | Memory Expansion Board | Data <br> Bus <br> Board |
| SIGNAL | PIN | PIN | PIN |
| DO | 33 | 63 | 49 |
| D1 | 34 | 61 | 47 |
| D2 | 35 | 59 | 45 |
| D3 | 36 | 57 | 43 |
| D4 | 37 | 55 | 41 |
| D5 | 38 | 53 | 39 |
| D6 | 39 | 51 | 37 |
| D7 | 40 | 49 | 35 |
| D8 | 41 | 31 | 67 |
| D9 | 42 | 29 | 65 |
| D10 | 43 | 27 | 63 |
| D11 | 44 | 25 | 61 |
| D12 | 45 | 23 | 59 |
| D13 | 46 | 21 | 57 |
| D14 | 47 | 19 | 55 |
| D15 | 48 | 17 | 53 |

Table 7.4.6

|  | SYSTEM BUS |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | INTERCONNECTIONS |  |  |  |  |  |  |
|  | MODULE 1 |  | MODULE 2 |  | MODULE 1 | MODULE 2 |  |
| SIGNAL | DATA BUS BOARD |  |  |  | ADDRESS BUS BOARD |  |  |
| $\overline{\text { PRES }}$ | 83 |  | 83 |  | 34 | 34 | A0. ${ }^{\text {B }}$ |
| INTCTL | 84 |  | 84 |  | 36 | 36 | A1. ${ }^{\text {B }}$ |
|  | SIGNAL | PIN | PIN | SIGNAL | 38 | 38 | A2. ${ }^{\text {B }}$ |
|  | TACKO | 78 | 72 | RACKO | 40 | 40 | A3. ${ }^{\text {B }}$ |
| BUSBUSY | 82 |  | 82 |  | 42 | 42 | A4. B |
| $\overline{\mathrm{HOLD}} . \mathrm{B}$ | 2 |  | 2 |  | 44 | 44 | A5. B |
| HOLDA.B | 4 |  | 4 |  | 46 | 46 | A6. $\mathrm{B}^{\text {A }}$ |
| $\overline{\text { MEMEN }}$. ${ }^{\text {S }}$ | 10 |  | 10 |  | 32 | 32 | A7. ${ }^{\text {B }}$ |
| $\overline{\text { WE. }} \mathrm{B}$ | 12 |  | 12 |  | 30 | 30 | A8. B |
| DBIN. B | 8 |  | 8 |  | 28 | 28 | A9.B |
| $\overline{\text { ¢1 }}$. $B$ | 6 |  | 6 |  | 26 | 26 | A10.B |
| READY.B | 14 |  | 14 |  | 24 | 24 | A11. ${ }^{\text {A }}$ |
| DO.B | 38 |  | 38 |  | 22 | 22 | A12.B |
| D1.B | 40 |  | 40 |  | 20 | 20 | A13.B |
| D2.B | 42 |  | 42 |  | 18 | 18 | A14.B |
| D3.B | 44 |  | 44 |  |  |  |  |
| D4.B | 46 |  | 46 |  |  |  |  |
| D5.B | 48 |  | 48 |  |  |  |  |
| D6.B | 50 |  | 50 |  |  |  |  |
| D7. B | 52 |  | 52 |  |  |  |  |
| D8.B | 54 |  | 54 |  |  |  |  |
| D9.B | 56 |  | 56 |  |  |  |  |
| D10.B | 58 |  | 58 |  |  |  |  |
| D11.B | 60 |  | 60 |  |  |  |  |
| D12.B | 62 |  | 62 |  |  |  |  |
| D13.B | 64 |  | 64 |  |  |  |  |
| D14.B | 66 |  | 66 |  |  |  |  |
| D15.B | 68 |  | 68 |  |  |  |  |

Table 7.4.7

## 7.5 <br> POWER SUPPLY

The power to the system is supplied by two DC power supplies. Due to lack of space, one of them is installed on the rack itself, behind the identification plate, and the other behind the rack. The specifications for these two power supplies are given in Appendix 7.5. All connections between the rack and elements outside it are made through easily removable sockets because there is very limited access to the back of the rack and any modification or maintenance will require the removal of the rack from the box.

### 7.6 SYSTEM BUS TESTING - PARTIAL COMMUNICATION SOFTWARE

We are now able to check the system bus. It was decided to have at first the simplest piece of software capable of exercising as many functions as possible on the system hardware and then to develop the complete communication software.

As we have only two modules the simpler way to check the system is to have two separate tests. The first will check the interrupt arbitration link and handshake control and the second will check the slave identification and also the handshake control. Both of them, at this stage, will only check a limited amount of the address and data transfer capability.

For the first test we need to define a CONTROLLER for the system. Module 1 , as defined in a previous section, was chosen to act as the system controller. There is no special reason for this choice. Any of the modules could have been chosen.

The choice of a controller module implies that the switches on the data bus boards (Figure 6.5.3) for module 1 and 2 must be set accordingly. Chapter 6 indicates which connections must be made.

With the connections properly made, the software test program can now be developed.

The first test software design is based on the block diagram of Figure 7.6.1. The assembly level language implementation is shown in Appendix 7.6.1. There, we can easily see that, when module 2 sends the message to the controller, in this case module 1 , the message is sent to address location 1000 which is the first location on the global communication area associated with module 1 , as defined in Chapter 6.


Figure 7.6.1 Block Diagram for First Test

With the help of the emulator (only for module l) and tracing facilities, the programs were run and, after a few corrections, like missing and short circuited lines, the system was approved and ready to go to the next test.

The second test checks the slave identification process. The identification of each slave on the system bus is made through inputs El to E10 on device 2A, Figure 6.5.2. For modules 1 and 2, as we have, the recommended connections as given by section 6.4.2, are the ones shown in Table 7.6.1.

|  | INPUT LEVELS |  |  |  |  |  |  |  |  |  |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | ---: |
| INPUT | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 |
| MODULE 1 | H | L | L | L | L | L | L | L | L | L |
| MODULE 2 | L | H | H | H | H | H | H | H | H | H |

L - Low Level
H - High Leve 1
Table 7.6.1

So module 1 is identified by slave address 1 FFE and module 2 by slave address 2 FFE .

The second test is based on the block diagram, shown in Figure 7.6.2. The assembly level language implementation is shown in Appendix 7.6.2. The interrupt routine is, of course, not common to both modules but only represented as such for convenience in Figure

Module 1 identifies module 2 as a slave by writing to address 2 FFE. Then it sends the message to its internal address 2000 which is received by module 2 of address $F 000$ (see Chapter 6). Module 2 identifies module 1 as a slave by writing to address 1 FFE . Then the message is sent to its internal address 1000 which is received by module 1 at address F000. The slave identification process proved to be the most sensitive among all the other processes. Noise on the lines was one of the many sources of problems. The final design, for the slave identification decoder, shown in Figure 6.5.2, is the result of many hours of emulation and tracing and several modifications on the original design. The impossibility to emulate both modules at the same time, proved daunting.


Figure 7.6.2 Block Diagram for Second Test

In the end, after a long and lonely time, the design reached a quite satisfactory performance, with all the initial problems solved but not forgotten.

In order to check the system's reliability, the memory test program was added to the second test. Now, instead of a master just sending one word of message to the slave, it would check the whole available slave communication area. The program was run in closed form, with both modules acting as masters or slaves, for 8 consecutive hours on many occasions and no error was displayed in any of them.

### 7.7 THE GENERAL COMMUNICATION SOFTWARE

### 7.7.1 The Controlled Process and Message Exchange

The communication software for our multi-microcomputer system is based on two fundamental facts:

1. The system is going to be used as a real-time controller and not as a general computer.
2. The software has to be module oriented in order to maintain the flexibility for easy upgrading.

The real-time controller function implies that there is a process to be controlled with well defined characteristics. The controlling function, for this well defined process, will probably be realised by the execution of many different but interrelated tasks. The idea of a well defined process is really important in order to take advantage of a multi-microcomputer environment and be able to find which different tasks there are.

It is clear that a defined process will require a finite number of tasks to be executed as indicated in Figure 7.7.1.

PROCESS


Figure 7.7.1 Tasks for a Defined

The other almost clear feature of such a process is that it is wellknown how the tasks interrelate and which, if any, has higher priority over the others.

The interrelation among tasks, as they are executed in the multimicro environment, is basically the exchange of information between them. This exchange is achieved simply by the transfer of messages between the tasks. In Figure 7.7.2, four tasks transfer messages between them. The arrows indicate the direction in which the message flows (sender-receiver). The title on each arrow is the message ID card:


This title is directly related to a reliable communication software. In such software means must be provided in order to have messages univocally identified. The first and last numbers on the title provide the means to achieve reliability. The first number indicates the origin of the message and the last to what the message relates.


Figure 7.7.2 Message Exchange between Tasks

Different tasks do not necessarily mean that there are as many modules in the control system. Tasks can be grouped and executed in one module. It is clear that the exchange of information between tasks executed on the same module do not require the use of the system bus. The system bus will be used only for the exchange of messages between tasks executed on different modules.

### 7.7.2 Message Identification - The Transfer Vector

In our case, for a unified message identification scheme, each message contains a well defined identifier added to it. This identifier is a 16 -bit word with 2 defined fields: the most significant byte contains the master identification (sender) and the least significant byte contains the message identification.

| most | 8 bits | 8 bits | least |
| :---: | :---: | :---: | :---: |
|  | MASTER ID | MESSAGE |  |
|  | $\longleftarrow$ most byte $\longrightarrow$ | least byte |  |

## Figure 7.7.3 Transfer Vector

This identifier is called from now on a transfer vector and its general structure is shown in Figure 7.7.3. As the exchange of information is achieved on a master-slave basis, the slave is identified for management purposes on the system controller. The value to be given on the master identification field is related to the module acting as master, so the module number will be used. The message identification depends on the total process and will have to be.chosen accordingly. If for instance module 4 acting as the master transfers message 3 to slave 2 (module 2), the transfer vector contains:


When in possession of the transfer vector, the slave knows from which module the message is coming and to what it relates. The system controller can use the transfer vector to achieve many of the functions mentioned in Chapter 6.

### 7.7.3 Message Transfer Modes - Transparency

The coumunication software is designed to achieve modularity. It is structured in a way that it can be used by any module in the system. It allows any module to transfer messages within the system and is totally transparent to the system user in relation to the system bus hardware.

In our coumunication software a message can be transferred in two basic ways:

1. word transfer
2. block transfer

The word transfer mode allows the user to transfer a single word with a message identification extra word if necessary. The block transfer mode transfers one block of sequentially arranged words. Figure 7.7.4 illustrates both modes.


Word Transfer Mode


Block Transfer Mode

Figure 7.7.4 Word and Block Transfer Mode

The word and block transfer modes are accessible as independent routines. The extended operation instructions (XOP) are not fast but are quite simple and elegant and are used to implement the word and block transfer routines.

Before we develop the mentioned routines we will make some definitions which will help their understanding.

### 7.7.4 Single and Global Message Definition

When a module uses the system bus it must do so efficiently. For this reason we say that within the period the master has control of the bus, it is transferring a global message. This global message is formed by several single messages. Each single message is related to a particular slave and there will not be two single messages to the same slave within a global message.

Each single message (SM) will contain its own message identifier and will be separated from other single messages by an SM terminator. Within a single message there can be as many calls to the word or block transfer routines as necessary.

The end of a global message (GM) will be indicated by a GM terminator. This terminator indicates to the master that the system bus can be released.

The software design for the word and block transfer routines is based on the diagram shown in Figure 7.7.5. This diagram shows only actions taken by the master. The controller and slave actions are shown later on.

For management purposes, the global message has a global identifier which is transferred to the system controller as the first single message.

As we are interested in a general software design, the RECHOLDA line is assumed to be connected to an interrupt line.

The block diagram of Figure 7.7 .5 indicates very clearly that the preparation of the single message tables holds the key for the whole operation. Details about it are shown on what follows.


Figure 7.7.5 Master's Communication Actions

The global message general structure used in this work is:

## GLOBAL MESSAGE

```
MESSAGE 1 CALL Word Transfer Routing (WTRANS)
SINGLE MESSAGE TERMINATION
MESSAGE 2 CALL Word Transfer Routine (WTRANS)
    MESSAGE ID (source)
    Address on Slave (destination)
    Word 1 (source)
    Word l,l (destination)
    Word 2 (source)
    Word 2,1 (desgination)
    CALL Block Transfer Routine (BTRANS)
    Block Address 1 (source)
    Number of words
    Destination Address 1
SINGLE MESSAGE TERMINATION
GLOBAL MESSAGE TERMINATION
```

It is easily seen that words not sequentially arranged in memory can be transferred by just one call to the word transfer routine. This avoids the time wasted on the XOP call. It is very important to maintain the sequence: first source and then destination specification.

Assuming that nothing is going to be transferred to or from memory locations 0000 and 0001 , the single message termination will be indicated by 0000 and the global message termination by 0001.

The single message tables to be created are:

| TABLES |  |  |
| :---: | :---: | :---: |
| SLAVE | MODE | SOURCE ADD for ADD |
| Controller | word | source 1 |
|  |  | SM END |
| $\mathrm{Sl}^{\text {ave }}$ i | word | source 2 |
|  | word | source 3 |
|  | block | source 4 |
|  |  | number |
|  |  | SM END |
| Slave ${ }^{\text {j }}$ | block | source 5 |
|  |  | number |
|  |  | SM END |
|  |  | GM END |

7.7.5 Word and Block Transfer Routines

The block diagram for the word and block transfer routines are shown in Figures 7.7.6, 7.7.7 and 7.7.8. Their implementation in assembly level language is shown in Appendix 7.7.1.

When the system bus is granted to the master, the interrupt master routine, which source program is seen in Appendix 7.7.2, takes over and prepares the module for the slave acknowledgement signal (RECHOLDA).


Figure 7.7.6 Global End Routine
7.7.6 Complementary Routines

The acknowledge master routine, shown in Appendix 7.7.3, interrupts and sends messages to all the necessary slaves up to completion of the global message. It.is triggered by the RECHOLDA line as indicated in Figure 7.7.5.

On the slave side, the communication is triggered by the INTSLV line (INT3), and the actions taken are shown in Figure 7.7.9.

It is clear that the INTSLV line serves two purposes. The first is to signal the slave the start and end of message transmission and the second is to signal the controller the start and end of bus use. The block, identify and serve message is not developed in full here because it depends on the application. The source program for Figure 7.7.9 is shown in Appendix 7.7.4.


Figure 7.7.7 Word Transfer Block Diagram


Figure 7.7.8 Block Transfer Routine Diagram


Figure 7.7.9 Slave Interrupt Actions

On the controller side, the communication is triggered by the RINTCTL line. The actions taken by the controller are shown in Figure 7.7.10.


Figure 7.7.10 Controller RINTCTL Action

This diagram indicates that a flag is passed to the INTSLV routine indicating when there is a controller action involved. The Assembly level language for this routine is shown in Appendix 7.7.5.

### 7.7.7 Considerations

It is clear, from what we have developed, that in order to have a communication software as general as possible a price must be paid in terms of execution.time. Even writing the software in assembly level language, we can feel the overhead imposed by the general approach. For our system (2 modules) a much simplified version
for the communication software is used. It allows communication without overhead from general software design. A description of its operation mode can be seen in the next chapters.

The real throughput for the general software could not be tested on its whole configuration as we only have 2 modules on the system. The test made using dummy slaves is not really a good one for checking overall performance. We know that must of the overall performance will depend on the particular application and hope somebody will go through these tests in the future and make the possible necessary tuning on the general communication software.

## CHAPTER 8

## THE CONTROLLER TO ELECTROHYDRAULIC DRIVE INTERFACE

### 8.1 INTRODUCTION

This chapter is concerned with the interface between the modular multimicrocomputer, designed in Chapter 6, to act as the controller for the electrohydraulic cylinder drive.

The hydraulic rig was designed as a general testing facility and Section 8.2 gives a description of it. Details of the tables, hydraulic cylinder, electrohydraulic servovalve and power supply are given there.

Section 8.3 discusses the servo amplifier used to drive the servovalve and sets up the necessary components to match the valve's electrical characteristics.

The analog nature of signals on the drive requires an analog input/ output interface in the controller interface. Section 8.4 discusses the choice of such interface and sets up the system configuration. The control of position implies a position transducer and Section 8.5 is concerned with it. There, an optical position transducer is chosen and the necessary interface to the controller is designed and discussed.

Throughout this chapter all the necessary interconnections between controller and rig and internal to the controller are specified in detail to make sure anybody can continue this work from where it has stopped.

### 8.2 THE HYDRAULIC RIG

The hydraulic rig used in this work was designed as a flexible tool in control systems research. A general view of the hydraulic rig is shown in Figures 8.2.1 and 8.5.12. Figure 8.2.1A shows details on the hydraulic cylinder and transducers.

### 8.2.1 Load Tables

The primary objective of a servo drive in position control systems is to drive a certain load within specified constraints. This load has in general two main characteristics, i.e. mass and friction; nonlinear friction can be considered to be the
(1) GUIDE HAY FOR FRICTIONLESS TABLE
(2) SLIDE WAY TABLE BUFFER
(3) CAST IRON BED
(4) CAST IRON TABLE
(5) LINEAR POSITION TRANSDUCER (POT.)
(6) LINEAR VELOCITY TRANSDUCER

(7) DIFFERENTIAL PRESSURE TRANSDUCER
(8) ELECTROHYDRAULIC SERVOVALVE
(9) SUPPLY LINE TO SERVOVALVE
(10) DIFFERENTIAL

PRESSURE
TRANSDUCER


FIGURE 8.2.1 HYDRAULIC RIG - GENERAL VIEW

(1) DIFFERENTIAL PRESSURE TRANSDUCER
(2) ELECTROHYDRAULIC SERVOVALVE
(3) LINEAR VELOCITY TRANSDUCER
(4) LINEAR POSITION TRANSDUCER(POTENTIOMETER)
(5) ASYMMETRIC HYDRAULIC CYLINDER
(6) LINEAR OPTICAL INCREMENTAL POSITION TRANSDUCER

summation of viscous friction, static friction and coulomb friction components. These characteristics are always present on real systems, even if in levels that can be sometimes neglected. In order to be able to emulate a certain load, the rig must provide means to vary the load characteristics over the range practically found. This range must include the possibility of certain characteristics having null values as, for example, in a frictionless load.

The most significant load characteristics in practical systems are mass, static friction and coulomb friction. Static and coulomb friction are inherent in many mechanical systems. Due to their nonlinear behaviour they are reduced to the lowest possible level and in high precision positioning systems their influence must be neutralised. Mass is a characteristic which, for most systems, is directly related to friction and in a test rig there must be a way to vary it without undermining the other characteristics.

In order to represent the most common real loads found in practice, and to cope with the possibility of changing characteristics, the rig was designed with two main loads or, as called from now on, two main tables. The two tables can be driven by the hydraulic cylinder independently or coupled together.

The first table is mounted on a plain slideway with dovetail and tapered gib as seen in Figure 8.2.2, Section A-A. The table weight is 90 kg . It is really the table of a vertical milling machine.

The second table is mounted on ball bushings, of the open type, with provision to have extra plates attached on top of it. The table and bearings weigh 32 kgf and each extra plate 15 kgf .

When the two tables are coupled together, it is possible to increase the inertia load, without undermining friction levels, by attaching extra plates to the frictionless ball bushing guided table. To emulate a frictionless load, the frictionless table can be disconnected from the other table and driven independently by the hydraulic cylinder.

### 8.2.2 Hydraulic Cylinder

The hydraulic cylinder used in the rig is of the asymmetric or single rod type, as seen in Figure 8.2.2, Section B-B. The piston dimensions and maximum stroke are given below:

|  | Hydraulic Piston |
| :--- | :--- |
| Bore | 38.1508 mm (1.502") |
| Rod | 26.7716 mm (1.054") |
| Stroke | $533.40 \quad \mathrm{~mm}$ (21.0") |

In order to cope with possible misalignment between the rod centre line and the table guideway, the rod is hollow and provided with an inner flexure rod which has limited lateral flexibility, see Figure 8.2.2, Section B-B. This all round flexibility copes with the residual misalignment between the line of action of the cylinder and the load guideways.

The piston is connected to the tables by the use of a metal arm which is bolted down to either or both tables. The rod's inner shaft is not fixed to this metal arm against a small shoulder, as seen in Figure 8.2.2. The metal arm is a welded element which is also used as the force transfer member between the hydraulic cylinder and the buffers in case the end of the fixed stroke is reached.

The cylinder is first fixed to metal plate by bolts through its end blocks and then this plate is bolted down to the cast iron bed. The use of dowels is necessary to avoid the longitudinal movement of cylinder in relation to bed.

### 8.2.3 The Servovalve

The valve available in the rig is a MOOG Series 76, Two-Stage Fluid Control Servovalve. This servovalve is a 4 -way valve and operates with force feedback in the spool position loop. The first stage or pilot stage is a symmetrical double nozzle and flapper, driven by a double air gap, dry torque motor. The valve type is E076-102 and, because it is a key element in the control of the rig, more details about it are given in Appendix 8.2.1.

Section B-B


Section A-A


FIGURE 8.2.2 Detail of Sliding Table and Hydraulic Cylinder Arrangement

### 8.2.4 Power Supply

Power to the hydraulic rig is supplied by a power pack whose circuit is shown in Figure 8.2.3.


Figure 8.2.3 Hydraulic Rig Power Pack Arrangement

The pump is a Sperry Vickers PVB5. This is an axial piston pump with variable displacement and pressure compensation. At 1800 rpm it delivers $191 / \mathrm{min}$ and its maximum working pressure is 210 bar. The electric motor driving the piston pump is a 3 phase and 5 HP device with a speed of 1440 rpm . This motor is of course not matched to the piston pump relating to their operating speeds. The pump delivery will be much lower than the $19 \mathrm{l} / \mathrm{min}$ which means that we have to accept the maximum cylinder speed related to it.

The filters on the high pressure side are of the high pressure type and filter 2 has a 3 marticles retention capability. Filter 3 is $10 \mu \mathrm{~m}$ nominal and has a bypass at 15 psi .

The cooler is water operated and is capable of maintaining the temperature at approximately $37^{\circ} \mathrm{C}$ for normal operating conditions when the tap is fully open.

### 8.3 THE SERVO AMPLIFIER

The servo amplifier used to drive the hydraulic servovalve is a commercial MOOG DC servocontroller, Model $82-300$. This unit is completely solid state using silicon semi-conductors and integrated circuit operational amplifiers.

A schematic diagram for the MOOG servocontroller is shown in Figure 8.3.1. There, it is easily seen that there are two operational amplifiers on the circuit driving the valve.

The first, OAl, is an operational amplifier input stage. The second, OA2, is associated with the current driven output stage. The input stage sums several dc signals and provides a range of voltage gains (R16). There are reasonable controls for setting input signal sensitivity and offset bias. The output stage uses the operational amplifier and two current boosting transistors. Current feedback is used so that servocontroller characteristics should not be affected by servovalve coil inductance and resistance.

The input associated with signal $e_{3}$ has fixed sensitivity. Signal $e_{2}$ can be balanced through a range of $+/-10 \%$. Signal $e_{1}$ can be adjusted from 0 to $100 \%$.

Removing R15 and using capacitors C 11 and C 6 , integral analog control can be achieved with appropriate settings.

With proportional control and R7 fully clockwise the output voltage of the OAl stage will be

$$
\begin{equation*}
\mathrm{ea}=-\left[\mathrm{R} 15+\mathrm{R} 16+\frac{\mathrm{R} 15 . \mathrm{R} 16}{\mathrm{R} 17}\right]\left[\frac{\mathrm{e} 1}{\mathrm{R} 6}+\frac{\mathrm{e} 2}{\mathrm{R} 3+\mathrm{BAL}}+\frac{\mathrm{e} 3}{\mathrm{R} 2}+\frac{\mathrm{e} 4}{\mathrm{Z} 2}\right] \tag{v}
\end{equation*}
$$

The associated current from the output state will be

$$
\text { io }=\frac{1000}{\mathrm{R} 19}\left[1+\frac{\mathrm{R} 22}{\mathrm{R} 23}\right] \text {. ea (mA) }
$$

```
R2 to R6 = 100 k\Omega
R15 = 47 k\Omega
R16 = 100 k\Omega
R17 = 1 k\Omega
```



Figure 8.3.1 Schematic Diagram for the MOOG 82.300 Servocontroller

Using the details for the servovalve given in the last section we can already set the output stage of the amplifier to match the necessary characteristics.

According to MOOG's recommendation we should have:
a) voltage across R23 (VR23) equal to 0.5 V when the rated current is applied to the servovalve
b) resistors R 19 and R 22 chosen so that with 10 V out of OA1, the voltage across R23 is about 0.5 V .

With rated current of 15 mA and using the equation for the current at the output stage, the recommendations impose the relations:

$$
\begin{aligned}
& 15 \times 10^{-3} \cong \frac{10}{\mathrm{R} 19}+\frac{\mathrm{VR} 23}{\mathrm{R} 23} \\
& \text { and } \frac{\mathrm{R} 22}{\mathrm{R} 19} \cong \mathrm{VR} 23
\end{aligned}
$$

There are only two equations for three variables, but if we assume that all the rated current goes through R23, the value for it should be approximately $33 \Omega$. Assuming $\mathrm{R} 23=18 \Omega$ almost all the rated current will go through it and R19 can be chosen with a large value making the current coming from the input stage very small. The values chosen were R19 $=33 \mathrm{k} \Omega$ and $\mathrm{R} 22=1 \mathrm{k} \Omega$. These values will give a rated current of 17 mA and a voltage across R 23 equal to 0.3 V when the output of OA1 is 10V.

The connections to the input stage will be discussed later on when the digital to analog converter is discussed.

### 8.4 ANALOG INPUT/OUTPUT BOARD

### 8.4.1 The Analog I/O Necessity

The use of a typical electrohydraulic servovalve to control the movement in the hydraulic rig means that the input to the servo amplifier must be an analog signal. Assuming, in our case, that this signal is going to be the control function generated by the digital controller, there must be a way to convert digital information into analog signals. This is the function, of course, of a digital to analog converter (D/A). It is mentioned here that there
is no simpler way to generate a control function for this type of system configuration.

A digital to analog converter costs money and ours had already finished by this time. The great help came from money available to build the rig. With this help we were able to choose and buy what we mostly needed to finish interfacing the controller to the rig:
a) a digital to analog converter and
b) a linear incremental position transducer.

Following on, we will discuss the digital to analog converter and in the next section, the position transducer.

When we started looking for a digital to analog converter, we kept in mind that besides the incremental position transducer, many other transducers have an analog output signal. For data acquisition or feedback control using this type of transducer, an analog to digital converter (A/D) is also necessary, so we started looking for both converters.

On the digital to analog side for our rig, we need at least one output channel to control the servovalve. On the analog to digital side, in order to have some flexibility, we need at least three input channels.

### 8.4.2 The Board's Choice

Having in mind that there is more to look for on $D / A$ and $A / D$ than just resolution and speed, we came across a quite reasonable device which fulfilled our needs and did not require any hardware interfacing to one of our modules.

This device is the RTI - 1241 S analog input/output subsystem from Analog Devices. It is totally compatible with the Texas TM9900/ 100M microcomputer board and the only hardware interfacing required is connections between the pins on the board's 100 pin edge connector.

The RTI - 1241 S is a complete, 12 bit resolution analog $\mathrm{I} / 0$ sub system which functional block diagram is shown in Figure 8.4 There are on the board 16 single-ended or 8 differential input channels and 2 output channels. The general specifications for


Figure 8.4 RTI-1241 S Block Diagram
the RTI - 1240 family is given in Appendix 8.4.1.
The RTI - 1241 S was chosen due to its software programmable gains on the input channel. This allows more flexibility in dealing with various signal sources.

The RTI - 1241 S appears to the controlling microcomputer as a block of eight contiguous memory locations in the microcomputer's address space (memory mapped). A11 control and data transfer operations are accomplished by writing into or reading from, one or another of the eight words exactly as would be done with read/ write memory. Each word has a preassigned function, and the eight words taken as a group comprise the memory map of the RTI - 1241S. Figure 8.4.1 shows this memory map. Details of words and functions associated with this map are important for this work and are discussed below. More details can be seen in Reference 79

The words in the memory map form a contiguous block of eight memory word locations. This block can be positioned anywhere in the microcomputer's address space by jumper selection of a base address. For convenience all word addresses in the memory map are referenced to the base address as seen in Figure 8.4.1 Some words are read only or write only functions and if an illegal operation is performed no useful data will result. The important words, for our work, are:

DAC2 (BASE ADDRESS + 0): Data written into this word is converted into an analog signal by the analog output channel DAC2. The 12 bit data converted is right-justified in the 16 bit microcomputer word; the four most significant digits are ignored and can therefore have any value. This is a write only address and any writing instruction can be used to write to it.

DAC1 (BASE ADDRESS + 2): This word functions in exactly the same way as DAC2 above, but produces analog output on the DAC1 output channe1.

GAIN (BASE ADDRESS + 6): The two least significant bits of this word set the gain of the instrumentation amplifier. The codes for each gain are:


Figure 8.4.1 Address Map for RTI-1241 S

| GO | 0 | 1 | 0 | 1 |
| :---: | :---: | :---: | :---: | :---: |
| G1 | 0 | 0 | 1 | 1 |
| GAIN | 1 | 2 | 4 | 8 |

This word is read/write.
MPX (BASE ADDRESS + 8): The eight least significant bits of this word select the analog input channel during data acquisition operations. The word is read/write. If it contains value 3 data is taken on input channel 3.

ADC DATA (BASE ADDRESS + E) : The results of analog to digital conversion are available in this word. Data will be valid until a new conversion is made. The 12 bit ADC output data is right-justified in the 16 bit microcomputer word. The four most significant bits are set to zero when in unipolar or offset binary coding and are set to the sign of the 12 bit value when in two's complement coding. This word is read only.

### 8.4.3 System Configuration

Before any settings on the Analog $1 / 0$ board are made, its position on the multi-microcomputer controller must be chosen.

The best configuration for our system, in terms of reliability, as we mentioned before, is shown in Figure 8.4.2. There is just one module directly connected to the rig (controlling) which minimises the risk caused by any system bus failure.

With this configuration the analog I/O board is mapped into the address space of module 2. To avoid any overlapping of positions already mapped on this address space, the base address was chosen as:

| Analog I/O Board |  |
| :---: | :---: |
| BASE ADDRESS | B000 |

With this selected address the address jumper locations were set as:


Figure 8.4.2 System Configuration

|  | X1 |  |  |  | X2 |  |  |  |  | X3 |  |  |  |  |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| OPEN | $8-1$ |  | $6-3$ | $5-4$ |  |  |  |  |  |  |  |  |  |  |
| CLOSED |  | $7-2$ |  |  | $8-1$ | $7-2$ | $6-3$ | $5-4$ | $8-1$ | $7-2$ | $6-3$ | $5-4$ |  |  |

An open connection means a high level signal on the input of the decoder.

The analog I/O board was placed on the racking system, below the microcomputer belonging to module 2 , as can be seen in Figure 8.4.2. The interconnections between the analog $1 / 0$ board and the microcomputer relating to pin assignment and function is shown in Appendix 8.4.2. It is clear that the analog I/O board is connected directly to the microcomputer's internal bus.

### 8.4.4 Analog Output

The output to the servovalve amplifier was defined as DACl just for convenience. From connector P5, which function and pin assignments are given in Reference 79 a two-way screened cable was brought to the back of the controller box where a DIN 5 -way $180^{\circ}$ socket was inserted. The connections made on the controller box are:

| ANALOG I/O BOARD |  | CABLE | DIN 180 <br> 5-WAY SOCKET |
| :--- | :---: | :---: | :---: |
| Function | PIN(P5) | Colour | PIN |
| DACI (out) | 13 | red | 1 |
| Analog <br> Common | 12 | blue | 5 |

With these connections made we have the link between the controller and the servovalve almost finished. It is only now necessary to establish to which of the servo amplifier inputs the analog I/O board signal is going to be connected.

For bi-directional operation of the servovalve, we need a bipolar output from DACl. The output range was chosen to be 20 V ( $+/-10 \mathrm{~V}$ ) which is the biggest one available on the analog I/O board. This range will give a better definition when small inputs to the servovalve are used, especially in the presence of noise. The output code is in 2's complement.

The maximum output current for the RTI - 1241 S is $+/-5 \mathrm{~mA}$, for the chosen range. For this value of current the minimum value for the resistor at the input of the servo amplifier should be $2 \mathrm{k} \Omega$. So with the resistors already existent on the amplifier, as can be seen in Figure 8.3.1, any input can be used. It is clear that the sensitivity of each input can be changed without much difficulty but we decided to maintain them as they are and use the input to the amplifier correspondent to signal $e_{2}$ (Input 2), which has a balance potentiometer for sensitivity adjustment. In order to have the possibility to move the servo amplifier around without any cabling problems, a DIN $180^{\circ}$ 5-way socket was provided on the back of the servo amplifier box to receive the command signal from the controller. The connections made on the servo amplifier box are:

| SERVO AMP BOARD |  | CABLE | DIN 180 <br> 5-WAY SOCKET |
| :---: | :---: | :---: | :---: |
| Function | Input | Colour | PIN |
| $\mathrm{e}_{2}$ | 2 | red | 1 |
| Ground | 6 | blue | 5 |

With the manufacturing of a 2 m long, 2 -way screened cable, with DIN $180^{\circ} 5$-way plug in both ends, the link between the controller and servo amplifier, carrying the servovalve command signal, is completed. The lay-out for this link can be seen in Figure 8.5.11

### 8.4.5 Analog Inputs

The analog inputs to the controller are provided by connector P3 on the analog $I / O$ board. There are 8 full differential input channels which, being differential, are effective in reducing the effects of noise and bias current.

From connector P 3 on the analog I/O board a 20-way ribbon cable with edge connector brings several input channels to the back of the controller box. There, a dIN $240^{\circ}$ socket (only available) was inserted and through it provision is made for two full differential analog inputs. The interconnections between the ribbon cable and connector P3 are shown in Appendix 8.4.3. For the provided analog inputs, the interconnections and assignments are:

| INPUT CHANNELS |  |  |  |
| :--- | :--- | :--- | :---: |
| ANALOG I/O BOARD |  | CABLE | DIN 2400 <br> 5-WAY SOCKET |
| Function | PIN(P3) | Colour | PIN |
| CHO HI | 5 | white | 5 |
| CHO LO | 4 | white | 4 |
| CH1 HI | 7 | grey | 3 |
| CH1 LO | 8 | grey | 2 |

With the analog $I / O$ board in place on the racking system we were able to test it and adjust its analog outputs according to procedure given in Reference 79 Using a digital voltmeter the offset and gain of both digital to analog converters were adjusted to give $+9.9951 V$, for maximum positive output and -10.000 V for maximum negative output. Apart from this necessary set up the analog $I / 0$ board worked perfectly well.

It is mentioned here that the analog I/O board is operating with necessary jumpers to be reset whenever a system reset occurs. Thus, if the RESET switch on the TM990/100 microcomputer is actuated (at power up or to recover from an error) the output of both DACl and DAC2 will become OV. This acts as a starting and safety mechanism for it will close the servovalve (zero input) whenever the RESET switch is actuated.

### 8.5 POSITION TRANSDUCER

8.5.1 The Optical Linear Incremental Position Transducer

The measurement of position for systems operated by linear actuators is easily achieved by the use of linear transducers, for there is no need for special gearing or coupling between system and transducer.

In our case, using a hydraulic cylinder, it is possible to provide a stiff coupling between actuator and load in a way that, in terms of dynamic effects, it is practically the same to measure position of the load (direct) or position of the actuator (indirect). As it is easier to measure the position of the load, that is what is normally done and we decided to do this.

The use of analog position transducers together with digital controllers is common and must not be discarded a priori. There are many design aspects which sometimes will avoid the use of a digital type transducer but there are two points to consider before choosing an analog transducer:
a) The use of an analog position transducer will most certainly require an analog to digital converter.
b) The resolution is directly related to the number of bits used for conversion and noise badly affects small resolutions.

Point a) will of course affect the price on the decision process but it is point b) which probably will indicate the chosen transducer, as discussed below.

In our case we have a stroke of approximately 500 mm . Using a normal 12 bits analog to digital converter, the smallest increment we could measure would be 0.122 mm . This resolution is not enough for most practical needs. Increasing the number of bits to 16 we could go down in resolution to $7.6 \mu \mathrm{~m}$, but if we assume a 20 V range, the voltage step would go down to 0.3 mV which would be certainly affected by any real working environment.

The position transducer resolution we are looking for is around $1 \mu \mathrm{~m}$ and an analog to digital converter of at least 20 bits resolution would be necessary. So for our application, a digital position transducer is really necessary.

The chosen transducer, to suit our needs, is a HEIDENHAIN LS 701 Linear Transducer. This transducer is an optical linear incremental position transducer of the Moire-fringe type. The grating pitch of the transducer is $20 \mu \mathrm{~m}$ and the chosen version has a pulse shaping electronics (EXE) with 5-fold evaluation (cycles of the solar cells are subdivided 5 -fold such that $1 / 5$ of the grating pitch is achieved).

The pulse shaping electronics is independent of the transducer and can also be used with a compatible optical angular transducer. In addition to the two by $90^{\circ}$ phase shifted signal trains, inverted signals for noise suppression are also available at the output of the pulse shaping electronics.

The block diagram showing the solar cells and the pulse shaping electronics (EXE) is shown in Appendix 8.5.1, with the signal diagram for the available outputs. Signal Ua ${ }_{o}$ is associated with a pulse generated at the middle of the transducer stroke. The output signals from the EXE unit are TTL compatible with a fan out maximum of 2 standard TTL loads. The recommended slope clearance in any of the output signals is to be bigger than $0.5 \mu s$.

### 8.5.2 Controller to Position Transducer Interface

The transducer resolution we are trying to achieve is, as mentioned before, $1 \mu \mathrm{~m}$. It is quite clear that we must condition the signals coming out of EXE unit as they only represent a resolution of $4 \mu \mathrm{~m}$ $(20 \div 5)$. So a 4 -fold subdivision is necessary to achieve $1 \mu \mathrm{~m}$. Beside it, there is the direction indication which is still associated to the $90^{\circ}$ phase-angle between signals Ual and Ua 2 .

The general interface between the position transducer and controller will follow the guidelines already discussed by the author (1) using a register buffered indirect counting as shown in Figure 8.5.1.


Figure 8.5.1 Register Buffered Indirect Counting
The operations load and clear indicated on Figure 8.5.1 are the most critical in this interface design, as any loss of pulse means an error in position measurement. In order to achieve the end results, the up and down pulses must be synchronized with the microcomputer's clock and with the sampling instants. Details of this will be shown later.

### 8.5.3 Detailed Interface Design

The detailed interface design is based on the diagram shown in Figure 8.5.2. The functions associated with each block in this figure will be discussed now.

The differential line receiver is used on the input lines coming from the transducer's pulse shaping electronics (EXE). These lines are already in a suitable configuration to use a differential receiver as they are available in pairs with inverted couples. The EXE unit available comes with a lm cable and will be enough for our needs. If an extension is needed, the receivers will cope with at least 20 m of cable.

The chosen receiver is the SN 75182 dual differential line receiver (80). It has a $+/-15 \mathrm{~V}$ common-mode input voltage range and $+/-15 \mathrm{~V}$ differential input range. The response time control capacitance was chosen as 100 pF , which gives a protection against a maximum pulse width, applied differentially, of 60 ns .

After the line receiver the input lines have been combined to give: Ua1 and $\overline{\mathrm{Ua} 1}$ give CH1 (channel 1), Ua2 and Ua2 give CH2 (channe1 2) and Ua3 and $\overline{\mathrm{Ua} 3}$ give CH3 (channe1 3).

The use of one of the microcomputer's clock phase as the synchronizing clock in our design was not possible. As seen in Chapter 6, the clock phase has a cycle which is not balanced in relation to LOW and HIGH periods. For instance, $\overline{\phi 1}$ has a cycle with a LOW period of 45 ns which will be very restrictive, as can be seen later on, due to propagation delays through the logical gates.

The clock generation block in Figure 8.5.2 indicates the generation of a better balanced clock cycle to be used as our synchronization clock, or just CLOCK.

The up and down generation block synchronizes CH 1 and CH 2 pulses with the CLOCK and generates up and down pulses to be used by the counter. These pulses are now synchronized with the CLOCK pulses. The clear and load generation block uses the CLOCK signal and a signal associated with the microcomputer's sampling interval, SAMPLE, to generate signals that will load the register with the value on counter (load) and after will clear the counter (clear). It is


Figure 8.5.2 Position Transducer Interface Diagram
quite clear that the use of the synchronizing CLOCK is necessary to avoid loss of pulses when the counter is cleared.

The decoder block indicates that the interface will be memory mapped on the microcomputer's address space which allows a fast and direct access to position data.

The final and complete logic diagram for the transducer interface is shown in Figure 8.5.3. The description and list of components is given in Appendix 8.5.2. The detailed analysis of all the logic involved is now given.

### 8.5.4 Timing Diagram and Interface Discussion

The CLOCK generation is achieved using the microcomputer's clock phases $\overline{\phi 1}$ and $\overline{\phi 3}$ through a flip-flop, SN74LS74, which is device number 22 in Figure 8.5.3. Before going to the flip-flop $\overline{\phi 1}$ is inverted (device 18) and $\overline{\phi 3}$ is just buffered (device 17). The timing diagram resulting from this logic is given in Figure 8.5.4.


Figure 8.5.4 CLOCK Generation Timing Diagram

It is clear on Figure 8.5 .4 that CLOCK is well balanced having a duty cycle of almost $50 \%$.

Before we can proceed to the next stages in our design, the timing on the transducer input signals ( $\mathrm{Ua}_{\mathrm{i}}$ ) must be analysed.

The maximum traversing speed permissible by the transducer is $30 \mathrm{~m} / \mathrm{min}$ for $1 \mu \mathrm{~m}$ resolution ( 81 ). Our design will try not to degrade this limitation.


Remembering that the output from the pulse shaping electronics (EXE) has each cycle associated to $4 \mu \mathrm{~m}$ it is easy to deduce the timing diagram for CH 1 and CH 2 when the transducer is moving at $30 \mathrm{~m} / \mathrm{min}$ (etc). This diagram is shown in Figure 8.5.5.


Figure 8.5.5 Input Signals at Maximum Speed

It is clear that for a $1 \mu \mathrm{~m}$ resolution, the cycles in CH and CH 2 must generate 2 pulses each. Associating each pulse with the positive and negative going edges of each cycle this is not difficult to be achieved. The restriction lies in the timing between 2 consecutive pulses. From Figure 8.5 .5 it is seen that at $30 \mathrm{~m} / \mathrm{min}$, we have $2 \mu s$ between 2 consecutive pulses. So any logic to load the register and clear the counter must work within $1 \mu \mathrm{~s}$ in order to have a good safety margin. Talking about CLOCK cycles the logic must work within 3 cycles ( 990 ns ).

All the logic in the interface is interrelated, of course, but first we will talk about the UP and DOWN pulses generation.

The UP and DOWN pulses generation is associated to devices 1,2 , 14 and 16, Figure 8.5.3. Assuming that the SAMPLE line is kept high the timing diagram for the UP and DOWN pulses generation is shown in Figures 8.5.6 and 8.5.7. From these figures it is quite clear that 2 pulses are generated for each channel cycle giving the necessary 4 -fold subdivision to achieve the desired $1 \mu \mathrm{~m}$ resolution. The relationship between CLOCK 2 and CLOCK will become clear later on. The up and down binary counter associated with devices 4, 5, 6, and 7 is actuated on the positive going edge of input signals UP and DOWN which according to Figures 8.5 .6 and 8.5 .7 is related to a negative


Figure 8.5.6 Timing Diagram for UP Pulses Generation


Figure 8.5.7 Timing Diagram for DOWN Pulses Generation
going edge of CLOCK 2. This relationship is used in the following discussion to achieve the timing imposed by the maximum speed. When the sampling instant is reached the microcomputer will set the SAMPLE line to a LOW level. This signal will automatically load the register and clear the counter and the logic associated to it avoids any loss of pulses. The logic involved is easily understood through the timing diagram shown in Figure 8.5.8. The register is loaded on the positive going edge of LOADREGISTER and the counter is cleared on the positive going edge of CLEARCOUNTER.

From Figure 8.5.8 it is seen that the required time of $1 \mu \mathrm{~s}$ is achieved. In the worst possible condition a pulse has been accepted with the last negative going edge of CLOCK 2 before the break interval. This pulse will go to the counter and it will have plenty of time to settle down ( 330 ns ) before the LOADREGISTER signal goes high. This signal acts as the clock signal for the register latches. At this time all the data stored on the counter is transferred to the register and then the counter can be cleared, as it is 160 ns later. At $30 \mathrm{~m} / \mathrm{min}$ and in the worst possible condition the next pulse will arrive only $2 \mu \mathrm{~s}$ after the negative going edge of CLOCK 2 on the break. So, the designed logic is fast enough to cope with the maximum speed and worst possible conditions avoiding any loss of pulses.

The register, on our design, is implemented using IC SN74 LS374 which has 3 -state outputs. The ENA line, in this design, is connected to the register control lines. When the ENA line goes LOW the outputs are enabled and data stored on the register is available on the internal data bus. As we mentioned before, the interface is mapped on the microcomputer address space. The address associated to it is chosen by switches available at the inputs to device 10 on Figure 8.5.3. This device together with devices 11 and 12 form a decoder with the same characteristics as the one for the slave definition phase mentioned in Chapter 6.

### 8.5.5 Final Interface Board and Interconnections

All the logic given in Figure 8.5.3 was implemented in a single printed circuit board. For easy reference we called it digitizer feedback board and its manufacturing mask is shown in Figure 8.5.9. The finished board is shown in Figure 8.5.10 and its associated table of functions and pin assignment is given in Appendix 8.5.3.


Figure 8.5.8 Timing Diagram for Load and Clear



The digitizer feedback board is located, on the rack, beside the data bus board of module 2 and its interconnections with the microcomputer are shown in Appendix 8.5.4. In order to avoid the use of long connection cables the necessary signals are brought from the nearest available sources. The SAMPLE line was connected to a CRU line through the TMS 9901, bit 23. The MIDDLE line which indicates a middle stroke for the transducer movement was connected to the microcomputer's INT 2 line just to be tested, without any consideration to priority at this stage, for it can be changed very easily.

The interconnections between the digitizer feedback board and the transducer's pulse shaping electronics (EXE) are made in two stages. First the necessary signals are taken to the back of the controller box and then to the EXE unit as shown in Figure 8.5.11. In order to make the rack totally independent from the controller box, a 15-way socket was fixed on the rack and the signals from the digitizer feedback board taken to it. To this socket is also connected the leads to supply power to the EXE unit ( 0 and +5 V ). At the back of the controller box was inserted a 12 -way socket compatible with the cable and plug supplied with the EXE unit. To connect the 9 -way socket at the back of the controller box to the 15-way socket on the rack a screened multi-way cable with a 15-way plug at one end was manufactured and installed. The final interconnections for the links discussed above are shown in Appendix 8.5.5 and the lay-out for the external connections to the controller is shown in Figure 8.5.11.


Figure 8.5.11 Lay-Out of External Connections


The digitizer feedback board was tested and worked perfectly well. The software to drive this board is discussed in Chapter 9, when the general software for the controller actions is developed.

A complete view of the hydraulic rig with controller and related testing equipment is shown in Figure 8.5.12.

## CHAPTER 9

### 9.1 INTRODUCTION

This chapter is concerned with the structure and implementation of the controller software. Section 9.2 shows the general software structure through its different functions. The functions related to loop closing are detailed in Section 9.3.

The design of the routine that closes the loop is given in Section 9.4 and Section 9.5 describes the position reference generation。

The interactive mode between operator and controller is discussed in Section 9.6. There, the data acquisition facility is discussed.

In Section 9.7, the controller tasks are divided between the modules and the necessary communications defined. This section also describes the auxiliary functions and the linking structure for the controller software on both modules.

The controllersoftware was written in assembly level language and details of implementation are given throughout the chapter.

### 9.2 THE GENERAL SOFTWARE STRUCTURE

The modular multi-microcomputer designed in previous chapters has only the TIBUG software available on its two modules up to this stage. To act as a controller the necessary software must be developed and this is done in what follows.

The main objective for the controller, in our application, is the control of position. This does not mean that other variables will not receive any attention but only that, basically, we will have a dominant position loop. Looking back to Figure 8.5.11 we see this dominant position loop represented by the path through module 2 , DAC, hydraulic-actuator and digitizer.

A software for position control as the main objective does not mean that only following error and references need be catered for. This would be enough in simple applications but not flexible enough to achieve our end results: the implementation of state-variable control with the possibility of easy changing and updating as needed in a changing research environment.

The availability of an electrohydraulic cylinder drive, for testing purposes, must not in any way dictate the rules. The software must be capable of dealing with different types of servo drives and the peculiarities of each drive easily accounted for.

The software design is based on the general structure shown in Figure 9.2.1. There, each block accounts for several functions and will be discussed in detail later on. In Figure 9.2.1 no mention is made to which system's module a particular block is associated, unless for the loop closing block. It is partially associated with module 2 which is part of the closed position loop path. The division of tasks between the modules will become clear when we discuss details of implementation.

The system initialisation comes on top of the general structure but, as normally happens, will be written and discussed after all the other functions have been implemented. The initialisation of variables requires first their definition.

The discussion of the functions associated with each block in Figure 9.2.1 will try to follow, as closely as possible, the top-down direction starting with the loop closing block.

### 9.3 LOOP CLOSING FUNCTIONS

The functions associated with the loop closing block are shown in Figure 9.3.1. The loop closing functions are implemented by the general loop closing routine which flow diagram is shown in Figure 9.3.2.

### 9.4 LOOP :CLOSING ROUTINE

9.4.1 Sampling Frequency Control

The sampling frequency is controlled by the TMS 9901 internal clock, as described in Chapter 6. The possibility to change the sampling frequency, while running, is a powerful way to achieve a good control action ( tuning) and can be used to achieve certain types of adaptive action, when required. The change in sampling frequency is associated with a flag called NEWFRE which will have to be tested each sampling interval. As mentioned in Chapter 6, the internal clock, when enabled, generates an interrupt level 3. The loop closing routine is the most important to achieve position



Figure 9.3.1 Loop Closing Functions

control, so it is directly associated to Interrupt 3, that is, it is interrupt driven.

### 9.4.2 Data Acquisition

The system variables acquisition in Figure 9.3 .1 is represented in Figure 9.3 .2 by the input variables block. We have restricted our work to single input systems but not to single output systems. Assuming that position is always available as an output, the controller should provide means to have access to all the available outputs. This does not mean that it is necessary to have more than one output to achieve the desired control action.

As discussed in Chapter 8, the position transducer and the digitizer feedback board provide position information and it is memory mapped on the address map. Assuming that any other system output is available as an analog signal, the simpler way to gain access to this signal is through the analog to digital converter in the $1 / 0$ board. There are 8 differential analog input channels provided on the controller. In this work we are only concerned with the use of position information and that is the only system output for which the software has provided access. For any analog input very simple software additions have to be made.

### 9.4.3 Position Control <br> 9.4.3.1 Feedback Counter

The counter and register provided on the digitizer feedback board are both 16 -bits wide including sign and can of course count and store to $\pm 32768$ pulses. In our case, with a position resolution of $1 \mu \mathrm{~m}$ this means that we can move only $\pm 32768 \mu \mathrm{~m}$ around null which is, of course, not enough. The requirement is to move the table within $\pm 250 \mathrm{~mm}$ or $\pm 250000 \mu \mathrm{~m}$ 。 For this value, we need to read position information in incremental form and to store position in double precision.

The position information in incremental form is achieved by reading and clearing the counter at each sampling interval. The problem of loss of pulses, when doing this, has already been solved when we designed the digitizer feedback board.

### 9.4.3.2 Actual Position

At each sampling interval, the actual position value must be updated by a summation process. The position value at sampling interval $k$ is the position value at interval ( $k-1$ ) plus the feedback counter value at interval $k$, as:

$$
\operatorname{ACPOS}(k)=\operatorname{ACPOS}(k-1)+\operatorname{DACPOS}(k) \quad 9.1
$$

As mentioned before, position cannot be represented internally by a single word, so ACPOS is a double precision value or ACPOS (ACPOS1, ACPOS2) where ACPOS1 is the most significant 16 bits word. To deal with double precision values, we need special routines and they will be discussed later on.

For position control the actual position value is not really important, when closing the loop, if we have access to the following error value. The actual position value is necessary as a display and checking value for operator communication and for position reference generation.

### 9.4.3.3 Position Following Error

Position following error is defined as:

$$
\text { ERROR }=\text { POSREF }- \text { ACPOS }
$$

where ERROR is the following error, POSREF is the position reference cormmand and ACPOS is the actual position. POSREF and ACPOS are both represented as double precision values but ERROR does not necessarily need to be represented as double precision. It is clear of course that the maximum expected following error will depend on the gain levels inside the position loop. It is assumed here that these gain levels are such that there is no saturation within the expected range of working conditions and that the maximum following error, for a resolution of $1 \mu \mathrm{~m}$, can be represented as a single precision value, or $\pm 32768 \mu \mathrm{~m}$.

If the position reference is represented in incremental form, which is updated every sampling interval, at interval $k$ we have:

$$
\operatorname{POSREF}(k)=\operatorname{POSREF}(k-1)+\operatorname{DPOSREF}(k) \quad 9.3
$$

where $\operatorname{DPOSREF}(\mathrm{k})$ is the increment in position reference at interval k .

Using equations $9.1,9.2$ and 9.3 , we can write for the following error at interval $k$ :

```
ERROR(k) = POSREF(k) - ACPOS(k)
and
```

ERROR(k) = POSREF(k-1) - ACPOS(k-1) + DPOSREF(k) - DACPOS(k) 9.5

```
From equation 9.4 we can modify 9.5 to give:
\(\operatorname{ERROR}(k)=\operatorname{ERROR}(k-1)+\operatorname{DPOSREF}(k)-\operatorname{DACPOS}(k)\)

Defining the incremental error at interval \(k\) as:
```

DERROR(k) = DPOSREF(k) - DACPOS(k)9.7

```
we have
\(\operatorname{ERROR}(k)=\operatorname{ERROR}(k-1)+\operatorname{DERROR}(k)\)

Equations 9.7 , and 9.8 are solved in the controller software to find the following error. The value for ERROR is represented as single precision and the conditions imposed are: there is no saturation within the working range and the position reference is calculated in incremental form as indicated in equation 9.3. Looking back to equations 9.7 and 9.8 , we verify that only two incremental values are necessary, after initialisation, to calculate the position following error at any interval. They are the position reference increment (DPOSREF) and the actual position increment (DACPOS). The position reference increment generation is dealt with on the loop reference generation section and the actual position increment is just the feedback counter value at the sampling instants.

\subsection*{9.4.3.4 Feedback Increment}

The digitizer feedback board hardware interface has already been described in Chapter 8 and now we will describe the necessary
software to acquire the actual position increment. The feedback register is memory mapped at address BFF2 which we call DPOS in the controller software. The line that triggers the feedback hardware is connected to bit 23 on the TMS 9901 CRU interface.


\section*{Figure 9.4.1 Feedback Sampling Action}

The necessary software to acquire the actual position increment is shown in Figure 9.4.1, where the increment value is stored at address called DXK. The importance of the digitizer feedback board is indicated by the simplicity achieved in the necessary software interface.

At this point it seems important to look at the possibility of saturation on the feedback counter. For \(1 \mu \mathrm{~m}\) resolution the maximum speed allowed by the position transducer is \(30 \mathrm{~m} / \mathrm{min}\) or \(500 \mathrm{~mm} / \mathrm{sec}\). If the software is to cope with. this maximum speed and no saturation on the feedback counter is admissible, there must be a restriction on the minimum sampling frequency for reliable operation. Calling the sampling frequency \(f\) and the sampling period \(T\), we can write
\[
\begin{array}{ll}
f=\frac{1}{T} & F\{H z\} \\
T\{s
\end{array}
\]

In general, the maximum counter value at a particular sampling instant can be written as
\[
\text { DACPOS }_{\max }=\frac{V_{\max }}{\mathrm{f}}
\]
where Vmax is the maximum speed.

In our case, for a resolution of \(1 \mu \mathrm{~m}\) and a 16 -bit wide, \(u p\) and down counter, the sampling frequency is limited to
\[
\mathrm{f}>\frac{\mathrm{Vmax}}{\text { DACPOS }_{\max }}=\frac{500 \times 10^{3}}{2^{15}} \cong 15.3 \mathrm{~Hz}
\]

So if we expect a speed of \(30 \mathrm{~m} / \mathrm{min}\), the sampling frequency should not be less than 16 Hz for correct operation.

It is clear that, for a maximum speed smaller than \(30 \mathrm{~m} / \mathrm{min}\), the sampling frequency can be decreased accordingly to equation 9.10.

Assuming that the sampling frequency is 6 times bigger than the load natural frequency on the servodrive and that the expected load natural frequencies are between 5 to \(100 \mathrm{~Hz}(84)\), it is clear that the counter will always operate correctly.

\subsection*{9.4.3.5 Observer and Control Function}

The blocks labelled observer, control function and updating are related exclusively to internal operations on the controller and were not generalised in this work. The observer and control function blocks are associated to.independent routines which are called by the general loop closing routine when necessary. In this way, the observer and/or the control action routines can be changed without any change on the main loop closing routine. These routines will be described later on in Chapter 10.

After the control function has been calculated, it is sent out of the controller through the digital to analog converter (DAC) on the I/O board. The DAC is memory mapped at address BOO2 (DAC1) and is called DAC1 on the controller software. A simple move instruction that moves the control function to location DACl is sufficient to send the control signal to the servo amplifier.

\subsection*{9.4.4 Emergency Action}

The loop closing function shownin Figure 9.3.1, named emergency action, is not shown explicitly on the flow diagram of Figure 9.3.2, but it is an important part of it. At each stage in the calculations, where an overflow is possible, a check is done and a branch is made to a called emergency deceleration routine, when conditions require.

The control function in our case (servo amplifier signal) is a direct command to the electrohydraulic servovalve and so proportional to actuator velocity. When conditions require the emergency deceleration routine will reduce the control function towards null by equal steps (constant deceleration). The value of the decrementing step depends, of course, on the maximum wanted deceleration. Ca1ling the decrementing step DEMU, the control function (U) at each sampling interval, when in emergency condition, is given by:
\[
U(k)=U(k-1)-\operatorname{sign}\{U(k-1)\} \quad x \text { DEMU }
\]
for \(|U(k-1)|>\) DEMU or
\[
\mathrm{U}(\mathrm{k})=0
\]
for \(\quad|\mathrm{U}(\mathrm{k}-1)|\) < DEMU

The flow diagram for the emergency deceleration routine is shown in Figure 9.4.2 and its implementation in assembly level language in Appendix 9.2.

The emergency deceleration routine is associated with two different flags on the main routine. These flags indicate which kind of action is necessary after the emergency has been served.

The critical point is related to the following error calculation. If there is an overflow it means that an error has occurred on the feedback path and the information on the actual position is not reliable. When this occurs, the flag FEFLAG is raised and no attempt is made to update the actual position value.


When overflow occurs in any other calculation the flag EMFLAG is raised and after the emergency routine has been executed the actual position is updated normally, for there is no indication of error on the feedback information.

The assembly level language implementation for the general loop closing routine which general flow diagram is shown in Figure 9.3.2 is given in Appendix 9.1. The comments make the routine selfexplanatory. There are some points about it which will become clearer as we proceed in our discussions.

\subsection*{9.5 POSITION REFERENCE GENERATION}
9.5.1 General Considerations

The position reference generation for machine tools requires linear and circular interpolation within restricted error band for several axes of movement. It is one of the most requiring in practical terms and, in order to maintain a certain direction of movement, the controller must relate the axes movement. Popa (43) has discussed all the possible alternatives and best algorithms for contour and feedrate control generation for numerical control. In our case we restrict ourselves to a single axis of movement and try to maintain the software as simple as possible for there is a severe limitation of memory available on the system.

The functions associated with the reference generation block in Figure 9.2.1 are indicated in Figure 9.5.1.


Figure 9.5.1 Reference Generation Functions

The step and ramp position. reference commands were thought to be enough to use in the test environment available.

As mentioned before, the position reference is calculated in incremental form as indicated in equation 9.3 which is repeated here for reference.
\[
\operatorname{POSREF}(k)=\operatorname{POSREF}(k-1)+\operatorname{DPOSREF}(k) \quad 9.3
\]

From this equation much useful information can be taken for the reference generation software.

The step position reference commands, from a steady-state condition, is generated by any non zero value of DPOSREF at any instant \(k\). If a step of height STEP is necessary at instant \(k\), out of \(a\) steady-state condition, we write
and
```

POSREF(k-1) = POSREF(k-2)
DPOSREF(k-1)=0
DPOSREF(k) = STEP
DPOSREF}(k+1)=
POSREF(k) = POSREF (k-1) + STEP

Equation 9.12 indicates very clearly that, out of a steady-state condition, any change in position reference means a step into the system as the position reference can only change by integer numbers of the position resolution. It is clear that using equation 9.12 we can generate steps with heights from position resolution up to saturation levels.

### 9.5.2 Ramp Generation Routine

The ramp position reference command requires the specification of two variables for complete definition. The first variable is the velocity and the second the distance to move.

If we call the velocity $V$ and the sampling period $T$, we can write for the incremental change in position reference

$$
\text { DPOSREF }=V \times T
$$

This incremental change is applied continuously up to when the required distance has been covered.

For a ramp starting at instant $k$ with defined initial conditions and a distance $D$ to move, we have for the initial and final conditions
initial: $\quad \operatorname{POSREF}(\mathrm{k}) \quad \operatorname{DPOSREF}(\mathrm{k})=0$
final: $\quad \operatorname{POSREF}_{\text {final }}=\operatorname{POSREF}(\mathrm{k})+\mathrm{D}$

The software has to calculate, at each subsequent sampling interval, the incremental change in position reference and this is given by equation 9.13. If the distance to move is an integer multiple of the value given by that equation, the problem is completely solved. For the majority of cases this is not true and the software will have to care for it. In this work the ramp is started with the increment defined by 9.13 and the last increment to complete the movement is the remaining distance to be moved. Defining the distance still to move as a variable, we can write using the ramp conditions specified before,

```
DISTA(k) = D
DISTA \((k+1)=\) DISTA \((k)-V \times T \quad 9.14\)
```

where DISTA is the distance still to move on the ramp movement. It is easy to show that $\operatorname{DISTA}(\mathrm{k})$ and $V \mathrm{x} T$ have the same sign. If DISTA( $k+1$ ) has not got the common sign it means that the distance still to move is smaller than the ramp position increment and so the next position increment must be the remaining distance to move.

The flow diagram for the ramp generation is shown in Figure 9.5.2. This routine is flexible enough to allow the generation of limited position steps when properly used.

When a step input is required the step height is the same as the distance to move on the ramp routine. If the ramp velocity is chosen so that the increment per sampling period given by equation 9.13 is bigger than the step height, the ramp routine will make the position reference increment equal the required step, for there will be an overflow on solving equation 9.14.


Figure 9.5.2 General Ramp Generation Routine

The condition for a step input using the ramp generation software can be written

```
STEP \leqslant V x T9.15
```

Using equation 9.9 , we have

$$
\operatorname{STEP} \leqslant V / f
$$

where f is the sampling frequency.

The maximum allowed velocity for the linear transducer is $30 \mathrm{~m} / \mathrm{min}$ or $500 \mathrm{~mm} / \mathrm{sec}$. Using this as the maximum allowed velocity by the software, for sampling frequencies varying between 100 and 500 Hz , the maximum step input will be between 1 to 5 mm as given by equation 9.16.

The range of step inputs allowed by the ramp generation software is thought to be suitable for our needs and no special software is written for step generation. Step and ramp are generated by the same software.

The assembly level language implementation for the ramp generation routine is shown in Appendix 9.3. It also includes other functions which have not yet been discussed.

### 9.6 DATA ACQUISITION AND OPERATOR'S COMMUNICATION

The functions associated with data acquisition and storage and operator communication, as shown in Figure 9.2.1, are interrelated in this work. The restriction in memory size, for data storage, made this quite inevitable. The restriction in memory is imposed on the operator who must choose which kind of data is to be stored.

The functions linked to operator communication are shown in Figure 9.6.1.

### 9.6.1 Operator's Options

The best way to achieve communication with the operator is using interactive routines. To start with, a table of possible lines of action must be presented and a choice made. The table of options must be self-explanatory and, most important of all, easy to be changed and/or upgraded, when necessary.


Figure 9.6.1 General Operator Communication Functions

In this work, the options offered to the operator are:

Options 1. Change sampling frequency
2. Ramp input
3. Display stored values
4. Stop and re-initialise

This table of options is sufficient to operate the system without heavy use of memory and can be, as mentioned before, upgraded very easily.

Each option is written as an independent routine and, when a choice is made, a branch to the correspondent routine is executed. This allows easy upgrading of the software.


The flow diagram for the table of options routine is shown in Figure 9.6.2 and its implementation in assembly level language is given in Appendix 9.10, where it is part of the MODIN1 routine.

### 9.6.2 Sampling Period

The choice of the change the sampling frequency option will cause a branch to a routine written under the name NEWFRE. The flow diagram for this routine is shown in Figure 9.6.3.

As mentioned in Chapter 6, the minimum frequency allowed by the TMS 9901 internal clock is 3 Hz . The software cares for it and would not allow frequencies below 5 Hz . With a range of frequencies from 5 to 999 Hz , the controller will cope with the expected range of load natural frequencies.


Figure 9.6.3 Changing Sampling Frequency Routine (NEWFRE)

At 3 MHz each TMS 9901 clock unit (resolution) equals $21.3 \mu \mathrm{~s}$. For a sampling frequency, f, the value to be loaded on the clock is given by

$$
\text { PERIOD (clock units) }=\frac{10^{6}}{21.3 \times \mathrm{f}}
$$

which is approximated, in the calculations, to

$$
\text { PERIOD }=\frac{46948}{F}
$$

This approximation implies an error which is negligible for the range of sampling frequencies expected.

The implementation for the changing sampling frequency routine is shown in Appendix 9.4.

### 9.6.3 Ramp Input Options

The ramp input option choice will cause a branch to a routine called RAMP1. This routine includes the ramp generation, already discussed, and here we discuss the operator's communication interactive facility provided by it.

For the ramp generation, the two main necessary inputs are the velocity and distance to move.

The velocity is indicated by its modulus (speed) and its direction which will be associated with the sign on the distance to move. The message asking input of speed is shown in Appendix 9.3.

The limitation on the maximum speed has already been explained and will not be repeated here. The minimum speed allowed by the software is determined by the sampling frequency. It was decided to allow only speeds down to 1 position resolution ( $1 \mu \mathrm{~m}$ ) per sampling period.

Using equations 9.13 and 9.9 , the minimum speed is given by

$$
V_{\min }=1 \text { Position Resolution } \times f
$$

For a position resolution of $1 \mu \mathrm{~m}$ and a practical range of sampling frequencies from 50 to 500 Hz , the minimum speeds would be within 3 to $30 \mathrm{~mm} / \mathrm{min}$.


It is clear that the minimum attainable speed increases with the increase in sampling frequency. For high moving speeds this restriction was thought to be not important and the software was developed according to the flow diagram shown in Figure 9.6.4. The distance to move, necessary for the ramp generation, is associated with the input of a new position command. It was decided, for better understanding between operator and controller, to have position represented as absolute values. The zero reference is a fixed position in space but the controller assumes that the zero reference is the actual position when the system is initialised. This allows the absolute reference to float within the possible limits.

The use of absolute reference requires the input of plus and minus signs and this is dealt with by a routine called SIGRED, shown later on. The distance to move is calculated as

$$
\text { DISTA }=\text { COMPO }- \text { ACTUL } \quad 9.20
$$

where DISTA is the distance to move, COMPO is the new commanded position and ACTUL is the actual position. All three variables involved in equation 9.20 are represented as double precision and to solve it, and other double precision equations, special routines were developed. Routines such as TWOCP2 and DBADD are described later on.

Using equations 9.13 and 9.9 the position increment per period is given by

$$
\begin{array}{lll}
\text { DPOSREF }=\frac{100 \times \mathrm{V}}{6 \times \mathrm{f}} & \begin{array}{l}
\mathrm{V}\{\mathrm{~mm} / \mathrm{min}\} \\
\mathrm{f}\{\mathrm{~Hz}\}
\end{array} & 9.21
\end{array}
$$

where V is speed and f sampling frequency.
With V and f represented as single precision values, equation 9.21 can be solved using only normal instructions.

### 9.6.4 Storing Variables

The next interactive function on the ramp input software is related to variables which the operator probably would like to have stored during operation for later use. This function is directly related
to data acquisition. The restriction in memory size, already mentioned, allows only one variable to be stored at any time. Our interest lies specially in the system behaviour during transient conditions and so, within our limitations, 112 words were allocated for storage purposes. This block of words is filled with the first 112 sampled values of the chosen variable. If the ramp movement takes less than 112 sampling intervals detailed information for all the movement is obtained. If it takes more than 112 sampling intervals only the initial transient part of the movement is represented. As we will see later on, in our experimental work, the restriction in memory did not impair the acquisition of representative information for various testing conditions.

The table of variables which can be stored by the software at operator's choice is

Variable 1) Actual Position (least word)
2) Following Error
3) Control Function
4) Speed
5) Acceleration
6) Transient Acceleration

Apart from the actual position, all the other variables can be represented as single precision values when properly scaled. The actual position is a double precision value and to avoid further reduction in the available storage space only the least significant word is stored. It is clear that this information will only be a true representation of actual position when the movement is restricted to $+/-32.768 \mathrm{~mm}$ around the absolute zero reference.

The variables chosen to be included on the table mentioned above were thought to be the most representative for our case. It is clear that any other variable which can be directly measured or estimated can be included on the table with no difficulty whatsoever.

After the variable to be stored is chosen the controller will ask the operator if the movement is to be executed or not. With a negative answer the software will branch to the start of the table of options routine, as indicated in Figure 9.6.4. When an affirmative answer is given the software will continue to the ramp generation routine which has already been described.

Details of implementation for the communication software described above can be seen in Appendix 9.3.

### 9.6.5 Displaying Stored Values

The display stored values option is directly associated with the ramp input data acquisition process. After the ramp movement is finished the software returns and displays again the table of options. If the operator wants to display the stored values he will choose the correspondent option and a routine called DISPLA is executed. This routine, which assembly level language implementation is shown in Appendix 9.5, displays 112 values with 8 values in each line. It converts two's complement hexadecimal stored values to signed decimals before they are displayed. For this a special routine called HEXDEC was written and is described in the auxiliary functions section.

### 9.6.6 Restarting the System

The operator's stop and reinitialise option is mainly to be used as a restart procedure when something goes wrong. It is clear that for a software restart, under operator control, at least part of the hardware must be working properly. The emergency condition, already discussed, caused by overflow errors, requires system reinitialisation after it has occurred.

The system bus provides the way to achieve a complete reinitialisation procedure. Through the appropriate lines the bus controller can reset all the modules connected to the system bus. A complete reinitialisation is not always necessary even in our case with only two modules depending on the task division between them. As mentioned before the microcomputer boards have a reset push-bution which allows any module to be restarted independently from the others. For a system as ours, in constant development, the software restart means a certain loss of flexibility as it requires the use of restart vectors blown in non volatile memory. Any rearrangement in software inside the memory would require new vectors and extra work which we think would be wasted in our case.

After some initial developments the stop and reinitialise option was left aside but the option maintained on the table of options
to remind the author, and any future user, that some thought had been given to it and when really necessary the required guidelines can be taken from here.

If it is necessary to stop and reinitialise our system the reset push-buttons on the microcomputer boards will be actuated.

### 9.7 TASK DIVISION BETWEEN MODULES

### 9.7.1 General Division

The software developed up to now is totally independent of any hardware implementation, which is necessary to achieve maximum modularity. At this stage we decided to assign tasks to both system modules based on the control software developed so far. This will allow, as mentioned at the end of Chapter 7, a much simpler implementation for the necessary communication software.

For reasons already discussed, module 2 is part of the position control loop and this indicated a task subdivision which was used in this work. This subdivision is not unique and, in the author's opinion, probably not the best, but that is what is interesting about it all: flexibility. Any new subdivision can be implemented and tested very easily.

It was decided to implement on module 2 the functions closely related to the position control loop and on module 1 the loop reference generation and operator communication functions. If there are several loops to be controlled, with one module associated to closing each loop, this is the most reasonable task subdivision in the author's opinion. The tasks associated with each module are shown in Figure 9.7.1.

### 9.7.2 Master-Slave Communication

As clearly shown in Figure 9.7.1, the inter-micro communication task is associated only with module 2. This suits our needs and simplifies significantly the necessary communication software. It does not mean that module 1 has not got.part of the communication software but only that we are restricting module 1 to behave only as a slave on the system bus. All the necessary transfer of information between modules is commanded by module 2.


Figure 9.7.1 Task Subdivision

The functions associated with the communication software are shown in Figure 9.7.2. The distinction between data and flag is used to allow a general definition for they are both really data as far as the system bus and memories are concerned. The controller software must distinguish between them and can take advantage of such distinction.


Figure 9.7.2 General Communication Functions

For module 2 , acting as the master, the general communication software is based on the flow diagram shown in Figure 9.7.3. The assembly implementation is given in Appendix 9.6 where the routine is called COMMUN.

It is of course clear that this routine is served every sampling interval and comes just after the loop closing routine when all variables were input or estimated. At this stage the position increment for the next sampling interval must be available or must have already been calculated by module 1.

If the flag for a new frequency is set, module 2 will load the clock with the new period at the next sampling interval. The flag location on module 1 is reset to indicate that action has already taken place and to avoid unnecessary clock loadings.

For safety reasons, module 2 clears the position increment location on module 1 as soon as it is transferred. If module 1 is not able to calculate the next increment in time there will not be harmful consequences.


Position Increment New Frequency Flag Period

Variables

Reset Frequency Flag Clear Increment Command End of Message

Figure 9.7.3 Module 2 Communication Software

As module 1 does not use its internal TMS 9901 clock there is no problem of interference with the slave definition process which uses interrupt level 3 ( $\overline{\text { INT3 }}$ ).

The necessary communication software on module 1 is quite simple and its flow diagram is shown in Figure 9.7.4. The routine is driven by interrupt 3 and its implementation is shown in Appendix 9.7.


Figure 9.7.4 Module 1 Communication software

### 9.7.3 Auxiliary Functions

The auxiliary functions developed for this work are shown in Figure 9.7.5.

All routines are written to be used with the branch and link instruction as this is the most time effective way to make calls to any auxiliary routine. Return to the calling program depends on the auxiliary routine and detailed instructions are given on the correspondent assembly listings.


## Figure 9.7.5 Auxiliary Functions

The arithmetic routines, written for this work, relate to integer or fixed point numbers. The use of double precision number was inevitable as position is represented as such. Triple precision routines were not really thought to be necessary but they were made available in case truncation errors caused any problems. As can be seen in Figure 9.7.6 instead of having addition and subtraction it was decided to have addition and two's complement routines as this makes shorter the complete set of routines. The multiplication routines are available for signed numbers as the microprocessor set of instructions only has unsigned multiplication of single precision numbers.

The assembly level language implementation for the arithmetic routines is shown in Appendix 9.8. They are linked under MATHS $3 / O B J$.

To cope with the input of positive and negative numbers, specially related to the position command, a special routine called SIGRED was written. It reads the sign, input from the terminal, and returns a 0000 or $\operatorname{FFFF}$ for a positive and negative sign, respectively. Appendix 9.9. shows the listing for this routine.


Figure 9.7.6 Arithmetic Routines

The input of numbers is the central part on the operator's commication interactive process. To simplify it, the routine NUMBE2 was written. It reads a character from the terminal, assembles the number and checks if it is a valid number from 0 to 9. If it is an invalid character, it.returns to the normal address. When a carriage return is input, it skips one word on return. For a valid number it skips two words on return. Appendix 9.9 also shows the listing for this routine. The display of stored values already discussed requires the
conversion of hexadecimal (really binary) to decimal numbers. The conversion is restricted to single precision signed hexadecimal numbers and is executed by the HEXDEC routine which listing is given in Appendix 9.9.

### 9.7.4 Initialisation Routines

The initialisation routines for both modules are quite simple and are given in Appendices 9.10 and 9.11. For module 1 the routine initialises all variables and flags and the sampling frequency to 200 Hz . Then it goes into the operator communication process displaying the table of options already discussed.

For module 2 the routine initialises all the necessary variables and flags, sets the deceleration rate for the emergency routine and loads the TMS 9901 clock with a period correspondent to a sampling frequency of 200 Hz . On return from the loop closing routine, the inter-micro communication takes place. This is achieved by positioning the COMMUN routine just after MATIN on the linking procedure.

### 9.7.5 Linking Structure

A11 the routines developed so far must be linked to achieve the end product, that is, the controller software. This is done, independently for each module, using the linking routing (TXLINK) available on the Texas Development System.

As there are several routines to be linked and TXLINK only allows three routines to be linked at one time, the partial link (P) option must be used.

The linking arrangement for modules 1 and 2 are shown in Figures 9.7.7 and 9.7.8. It is recomended that this arrangement be followed to avoid any problems.

The resulting object code programs, MOD1P4/OBJ and M0D2P4/OBJ, are shown in Appendices 9.13 and 9.14. For more flexibility, they are assembled as relocatable object code and can be loaded into the respective modules using the load option in the TIBUG software.

In our application, the MOD1P4/OBJ program is loaded in module 1 starting at address F120 and the MOD2P4/OBJ program loaded in module 2 starting at address $F 000$.


Figure 9.7.7 Linking Arrangement for Module 1 Routines

MODULE 2


Figure 9.7.8 Linking Arrangement for Module 2 Routines

## CHAPTER 10

## STATE-VARIABLE DIGITAL CONTROL OF THE ELECTROHYDRAULIC CYLINDER DRIVE

### 10.1 INTRODUCTION

In Chapter 8, the hardware interface, between the multi-microcomputer controller and the electrohydraulic drive, has been designed and its implementation is described. The general software interface, discussed in Chapter 9, is now applied to the electrohydraulic cylinder drive.

The computer aided design for the state-variable feedback controller needs, at least to start with, an effective model of the system. The system modelling, which discussion is shown in Section 10.2, has been strongly influenced by earlier published work ( $83,84,85$ ).

The controller design for the electrohydraulic cylinder drive is described in Section 10.3. There, the design methods and simulation facilities, described in Chapters 3 and 4, are used.

The controller equations, including control law and observer, are scaled and have their implementation in assembly-level language described in Section 10.4.

### 10.2 MODEL IDENTIFICATION

The schematic diagram given in Figure 10.2 . 1 gives an indication of the system we are trying to control


Figure 10.2.1 Electrohydraulic Cylinder Drive

The use of an asymmetric cylinder with a four-way symmetric valve is not common and merits some discussion before the model is developed.

### 10.2.1 Load Natural Frequency

From Figure 10.2.1, we can write that the oil compliance of the hydraulic cylinder, on either side of the piston, is (line volume effects are considered to be trivial)

$$
\frac{1}{\mathrm{k}_{1}}=\frac{\mathrm{L}_{1}}{\mathrm{~A}_{1} \mathrm{~B}} \quad \frac{1}{\overline{\mathrm{k}}_{2}}=\frac{\mathrm{L}_{2}}{\overline{\mathrm{~A}_{2} \mathrm{~B}}}
$$

which for a series arrangement gives a total compliance

$$
\frac{1}{k}=\frac{1}{B\left(\frac{\mathrm{~A}_{1}}{\mathrm{~L}_{1}}+\frac{\mathrm{A}_{2}}{\mathrm{~L}_{2}}\right)}
$$

where $B$ is the $B u l k$ modulus, $A_{1}$ and $A_{2}$ the areas on both sides. The maximum compliance occurs when

$$
\frac{\mathrm{L}_{1}}{\mathrm{~L}}=\frac{1}{1+\sqrt{\frac{\mathrm{A}_{2}}{\mathrm{~A}_{1}}}}
$$

where $L=L_{1}+L_{2}$, giving

$$
\left(\frac{1}{k}\right) \max =\frac{L}{A_{1} B\left(1+\sqrt{\frac{A_{2}}{A_{1}}}\right)^{2}}
$$

10.4

Using equation 10.2 and defining a normalised compliance, Figure 10.2 .2 can be drawn. It indicates the variation of compliance with stroke ( $L_{1} / L$ ), for our case, where $A_{1}=2 A_{2}$. For this situation the maximum compliance is reached where $L_{1} / L=0.59$


Figure 10.2.2 Oil Compliance Variation with Stroke

For easy reference, the characteristics of the hydraulic cylinder and load are repeated here.

| Piston Areas | $A_{1}=1143.1 \mathrm{~mm}^{2}$ <br> $A_{Z}=$ |
| :--- | :--- |
| Stroke Length | $L=500 \mathrm{~mm}$ |
| Load Mass | $M=90 \mathrm{~kg}$ |

Table 10.1

Assuming a bulk modulus of 8000 bar the minimum natural frequency, relating oil compliance and load mass, is found using 10.4 as
$w_{o}=\sqrt{\frac{k \min }{M}}=\sqrt{\frac{A_{1} B\left(1+\sqrt{\frac{A_{2}}{A_{1}}}\right)^{2}}{L M}}$

Using Table 10.1, the minimum natural frequency for load and oil compliance is approximately
$w_{o}=\sqrt{\frac{1143.1 \times 10^{-6} \times 8 \times 10^{8} \times(1+\sqrt{0.5})^{2}}{0.5 \times 90}} \cong 243 \frac{\mathrm{rad}}{\mathrm{sec}} \quad 10.6$

As shown in Figure 10.2.1, the connection between the piston and load was represented as having a stiffness $\mathrm{k}_{3}$. This is mainly due to the flexure-rod described in Chapter 8. The characteristics of the flexure-rod are repeated here in Table 10.2.

Rod | Diameter | $\mathrm{d}=14.224 \mathrm{~mm}$ |
| :--- | :--- |
| Length | $\ell=686.0 \mathrm{~mm}$ |

Table 10.2

For a cylindrical bar the stiffness, for a load applied axially at one end, is given by

$$
\begin{equation*}
k_{B}=\frac{E A_{B}}{L_{B}} \tag{1}
\end{equation*}
$$

where $E$ is the Young's modulus, $A_{B}$ the area and $L_{B}$ the length of the bar.

Assuming a Young's modulus of $2.1 \times 10^{11} \frac{\mathrm{~N}}{\mathrm{~m}^{2}}$ and using Table 10.2, stiffness $k_{3}$ is given by
$k_{3}=\frac{2.1 \times 10^{11} \times 158.9 \times 10^{-6}}{0.686}=486.6 \times 10^{5} \frac{\mathrm{~N}}{\mathrm{~m}}$
Turning back to equation 10.4 the oil spring stiffness can be calculated and found as
$\mathrm{k}=53.1 \times 10^{5} \frac{\mathrm{~N}}{\mathrm{~m}}$

Associating the oil spring stiffness given by equation 10.9 and the flexure-rod stiffness given by equation 10.8 , it is clear that the oil spring stiffness will dominate the load-drive mass-spring system. So the natural frequency given by equation 10.6 is a reasonable indication of the load natural frequency.

### 10.2.2 Steady-State Characteristics

The equilibrium of forces applied to the load and the continuity equations for the flow through the electrohydraulic valve's orifices will show an interesting situation when the piston is made to move from one direction to the other.


Figure 10.2.3 Asymmetric Cylinder and 4-way Valve
As shown in Figure 10.2.3, we define the total load $\Sigma \mathrm{F}$ as

$$
\Sigma F=M \ddot{y}+c \dot{y}+F_{c}+F_{e}
$$

where $y$ is the piston displacement, $M$ is the load mass, $c$ is the coefficient of viscous friction, $\mathrm{F}_{\mathrm{c}}$ the coulomb friction and $\mathrm{F}_{\mathrm{e}}$ the load's external forces. Applying Newton's second law to the forces on the piston we have

$$
P_{1} A_{1}-P_{2} A_{2}=\Sigma F=\ddot{M} \ddot{y}+c \dot{y}+F_{c}+F_{e}
$$

where $P_{1}, A_{1}$ and $P_{2}, A_{2}$ are the pressures and areas on the sides of the piston.

The continuity equation applied to both sides of the cylinder, using Figure 10.2.3, gives
$\dot{y}=\frac{a_{1}}{A_{1}} C_{d} \sqrt{\frac{2}{\rho}} \sqrt{P_{s}-P_{1}}-\frac{a_{2}}{A_{1}} C_{d} \sqrt{\frac{2}{\rho}} \sqrt{P_{1}}-\frac{L_{1}}{B} \dot{P}_{1}$
and
$\dot{y}=\frac{a_{3}}{A_{2}} c_{d} \sqrt{\frac{2}{\rho}} \sqrt{P_{2}}-\frac{a_{4}}{A_{2}} c_{d} \sqrt{\frac{2}{\rho}} \sqrt{P_{s}-P_{2}}+\frac{L_{2}}{B} \dot{P}_{2}$ 10.12
where $C_{d}$ is the discharge coefficient, $\rho$ the oil mass density, $a_{1}, a_{2}, a_{3}$ and $a_{4}$ the valve's orifice areas and $P_{R}=0$.

Assuming an ideal symmetric 4 -way critical centre valve with matched orifices, we have
$a_{1}=a_{3}>0$ and $a_{2}=a_{4}=0$
or
$a_{2}=a_{4}>0$ and $a_{1}=a_{3}=0$
10.14

For a steady state condition, pressures $P_{1}$ and $P_{2}$ are calculated as follows.

For $a_{1}=a_{3}>0$ and $\dot{y}>0$ we have
$P_{1} A_{1}-P_{2} A_{2}=\Sigma F_{S S}$
and
$\frac{1}{A_{1}} \sqrt{P_{s}-P_{1}}=\frac{1}{A_{2}} \sqrt{P_{2}}$

For $a_{2}=a_{4}>0$ and $\dot{y}<0$ we have
$P_{1} A_{1}-P_{2} A_{2}=\sum F_{S S}$
and

$$
\frac{1}{A_{1}} \sqrt{P_{1}}=\frac{1}{A_{2}} \sqrt{P_{s}-P_{2}}
$$

Assuming $A_{1}=2 A_{2}$, equations $10.15, .16$ and $10.17, .18$, can be solved to give, for

$$
\begin{align*}
& \dot{y}>0 \quad P_{1}=\frac{1}{9} P_{s}+\frac{8}{9} \frac{\sum F_{s s}}{A_{1}} \\
& P_{2}=\frac{2}{9} P_{S}-\frac{2}{9} \frac{\Sigma F_{S S}}{A_{I}} \\
& \dot{\mathrm{y}}<0 \quad \mathrm{P}_{1}=\frac{4}{9} \mathrm{P}_{\mathrm{s}}+\frac{8}{9} \frac{\sum \mathrm{~F}_{\mathrm{SS}}}{\mathrm{~A}_{1}} \\
& P_{2}=\frac{8}{9} P_{s}-\frac{2}{9} \frac{\sum F_{S S}}{A_{1}}
\end{align*}
$$

Solution of equations 10.19 and 10.20 is shown in Figure 10.2.4 for both directions of movement.


Figure 10.2.4 Cylinder Pressures v Steady State Load

From Figure 10.2.4, it is quite clear that there is a jump on the cylinder pressures when the piston reverses direction. The height of the jump, on both pressures, does not change with the load but is only shifted vertically by it.

This pressure jump can cause, subject to the level of load friction, non-smooth operation around null speed ( $\dot{y}=0$ ).

This is a characteristic of the system we have and we did not have an asymmetric 4 -way valve available, as suggested by Viersma (86), to avoid the pressure jumps.

The linearised analysis for symmetric cylinders given by Merritt (87) and Ertan (83) show the influence of valve opening and pressure levels on the damping, through the flow-pressure coefficient. Here we only talk about the influence of different valve openings. Turning back to equations 10.12, for a steady state condition, the modulus of velocity for both directions is given by (ideal value)
$\dot{y}>0 \quad|\dot{y}|=\frac{C_{d}}{A_{1}} \sqrt{\frac{2}{\rho}} \sqrt{P_{s}-P_{1}} a_{1}$
$\dot{\mathrm{y}}<0 \quad|\dot{\mathrm{y}}|=\frac{\mathrm{C}_{\mathrm{d}}}{\mathrm{A}_{1}} \sqrt{\frac{2}{\rho}} \sqrt{\mathrm{P}_{1}} \mathrm{a}_{2}$
Assuming the same speed in both directions, we have the relation

$$
\frac{a_{2}}{a_{1}}=\sqrt{\frac{P_{s}-\left(P_{1}\right) \dot{y}>0}{\left(P_{1}\right) \dot{y}<0}}
$$

Using Figure 10.2.3, for a null load (EF), equation 10.21 gives

$$
\frac{a_{2}}{a_{1}}=\sqrt{\frac{1-0.11}{0.44}} \cong 1.42
$$

So, for the same speed, the valve opening for an extending asymmetric cylinder ( $\mathrm{A}_{1}=2 \mathrm{~A}_{2}$ ) is 1.42 times smaller than the one when it is retracting. This-means that we should expect less damped behaviour when the cylinder is extending.

The other aspect shown by equation 10.22 is related to the gain associated with different directions of motion. Assuming a linear relationship between current and servovalve opening the velocity/current gain for both directions will be probably related by the value given by equation 10.22 ,

$$
\frac{\left(K_{v c y}\right) \dot{y}>0}{\left(K_{v c y}\right) \dot{y}<0}=1.42
$$

at least for operation outside the null region.
Reviewing what we have discussed so far, for the symmetric valveasymmetric cylinder combination, we can say:
a) there are pressure jumps around null speed and smooth operation in this region might be impaired,
b) the predicted damping is dependent on direction of motion,
c) the velocity/current gain is also different when extending and retracting.

For very small valve openings, the valve lap dominates its behaviour. In our system, in this region, we also have the pressure jumps, so we will await the experimental results to see how it behaves.

Outside the null region, alternatives b) and c), shown above, indicate the association of two different models, one for each direction of movement. It is expected that the models will only differ on the load damping associated with them.

### 10.2.3 Analytical Mode1

The model definition in this work follows the approach taken by Bell and de Pennington (84) on the analysis of lightly damped electro-hydraulic cylinder drives.

The transfer function between the servo-amplifier command, $u$, as shown in Figure 10.2.1, and the valve spool displacement, w, is generally given as

$$
\frac{{ }^{w}(s)}{u_{(s)}}=\frac{K_{A} \cdot K_{M}}{\left(1+s T_{M}\right)\left(\frac{s^{2}}{w_{v}^{2}}+\frac{2 \rho_{v}}{w_{v}} s+1\right)}
$$

where $K_{A}$ is the amplifier gain and $K_{M}$ the current to spool displacement gain.

Neglecting Coulomb friction and external forces, the resultant transfer function, relating output velocity to servo-amplifier command, is written

$$
\frac{s y(s)}{u(s)}=\frac{K_{v}}{\left(1+s T_{M}\right)\left(\frac{s^{2}}{w_{V}^{Z}}+\frac{2 \rho v}{w_{v}} s+1\right)\left(\frac{s^{2}}{w_{\ell}^{Z}}+\frac{2^{\rho}{ }_{\ell}}{w_{\ell}} s+1\right)}
$$

$K_{v}$ :(velocity to control)total gain
where $\rho_{\ell}$ and $w_{\ell}$ are load damping and natural frequency respectively. For systems with a load natural frequency lower than the natural frequency of the servovalve, but comparable with the break frequency of the torque motor coil, it is possible to simplify the transfer function, given by 10.25 , introducing a lag with a time constant $T_{A V}$ which fits transfer function 10.24 in the region of interest. It is recommended to use $T_{A V} 20$ to $50 \%$ greater than the torque motor coil time constant, $T_{M}$. The transfer function reduced to its simpler form is then

$$
\frac{s y(s)}{u(s)}=\frac{K_{v}}{\left(1+s T_{A V}\right)\left(\frac{s^{2}}{w_{\ell}^{2}}+\frac{2 \rho}{w_{\ell}} s+1\right)}
$$

If the $\operatorname{lag} \mathrm{T}_{\mathrm{AV}}$ is determined experimentally, it should be chosen as the $45^{\circ}$ lag frequency of the amplifier and unloaded servovalve response.

The transfer function shown by 10.26 has been normalised and its behaviour for varying parameters studied in Appendix 10.1. Using that Appendix and the experimental results, we will find the parameters $\rho_{\ell}, w_{\ell}$ and $T_{A V}$ which give the best fit to the electro-hydraulic cylinder drive available.

### 10.2.4 Open-Loop Dynamic Test Results

The test on the electro-hydraulic cylinder drive, which diagram
is represented in Figure 10.2 .1, was carried out using basically a function generator and a storage oscilloscope. The signals from the transducers available on the rig were stored on the oscilloscope screen and then photographed with a polaroid camera. The various tests made on the rig are documented in what follows.

First and most important of all, we tried to get data in order to be able to identify, as we have already discussed, a suitable transfer function for the drive. This was done in open-loop condition by using a square-wave at the amplifier input, $u$, as shown in Figure 10.2.1.

In order to get an indication of parameter variation with valve opening, different levels of input signal were used in the tests. Due to the open-loop test condition, the short length of cylinder travel and mismatch between power pack and electro-hydraulic servovalve, the maximum valve opening we achieved, safely, was correspondent to approximately $40 \%$ rated current ( 15 mA ). The tests were carried out with the piston at mid-stroke as this point lies in the region of minimum hydraulic stiffness.

The results for three representative levels of input signal are shown in Figures 10.2.5, 10.2.6, and 10.2.7. In each figure, the time responses for velocity, thrust, pressures $\mathrm{P}_{1}$ and $P_{2}$ are shown. On the bottom of each $P_{1}, P_{2}$ and thrust displays, the velocity signal is also shown to indicate direction of movement. Thrust is defined here as (total load)

$$
\text { Thrust }=P_{1} A_{1}-P_{2} A_{2}
$$

where $P_{1}$ and $P_{2}$ are the pressures and $A_{1}$ and $A_{2}$ the piston areas as indicated in Figure 10.2.2.

As expected, Figures $10.2 .5,10.2 .6$ and 10.2 .7 show an increase in damping for increasing valve opening. The difference in damping for both directions is also clearly shown with the extending direction displaying less damping for a given speed.


Figure 10.2.5
Open Loop Response
(5\% rated current)

LS


Thrust $x$ time


Figure 10.2.6
Open Loop Response
(20\% rated current)


Velocity x time

$P_{1} \times$ time


Thrust x time

$\mathrm{P}_{2} \times$ time

Figure 10.2.7

Open Loop Response ( $40 \%$ rated current)

The jump in pressure, as predicted earlier, is an interesting characteristic, as shown in these figures. The behaviour of the pressures, independently, is remarkable, but their combination generates a thrust without any particular features. As the velocity behaviour is consistent with published work in this field and well understood, for our model identification purposes, this is what we really need. We accept the many non-linear characteristics shown in these figures as a fact, and hope that future works on modelling and simulation will explain the non-clear aspects and benefit from the documentation provided in this work.

### 10.2.5 Open-Loop Steady-State Test Results

The second set of tests carried out relate to steady-state behaviour, for analysis of threshold and gain nonlinearities. For such tests, a slow sinewave input is used to avoid any influence of dynamic effects. In our case, the effect of a very short stroke complicated the tests twofold. In order to get the servo moving back and forward without hitting the end stops, the function generator bias was used. Even for low levels on the input signal, the minimum achieved frequency on the sinewave was not the one desired but we could not do much about it. Some dynamic effects are visible in our results.

For an input level correspondent to $10 \%$ rated current on the servovalve, the velocity versus current plot is shown in Figure 10.2.8. Two important characteristics are clearly indicated there. The first, as we predicted before, is that there is a different gain (velocity/current) for both directions. For the same speed a larger current is necessary when the cylinder is retracting. The second characteristic relates to the behaviour around null velocity. The effect of non-linearities is clearly shown causing a servo deadband. For higher input levels, the frequency of the sinewave had to be increased in order to restrict the maximum displacement.

Figure 10.2.9 shows the plot for velocity versus current for an input level correspondent to $100 \%$ rated current. As soon


Figure 10.2.8 Velocity x Current (10\% rated)


Figure 10.2.9 Velocity $x$ Current
(Power Pack Saturation)
as the movement is initiated, the supply pressure starts to drop, up to the moment when it reaches the accumulator pressure. At this moment, the system stalls and this is clearly indicated on the nose shaped appendix on Figure 10.2.9. This happens first, as one would expect, on the extending direction, for it requires a bigger flow to maintain the same speed.

The maximum achieved input level, without noticeable interaction between power pack and load, was correspondent to $40 \%$ rated current ( 15 mA ) and, as mentioned before, this was the same level used in order to maintain a safety margin from the end stops. Figure 10.2 .10 shows the velocity versus current plot for $40 \%$ rated current. The null region in this plot is shown enlarged in Figure 10.2.11. The input signal frequency was 1 Hz .

The effect of high friction levels on the performance of positioning systems is totally undesirable. Sometimes, for design reasons, friction cannot be reduced below certain levels but one thing the designer must have in mind: if the friction level cannot be reduced, at least allow it to be a"well behaved non-linear function". One difficulty experienced in this work was that the gib fixture used in our rig, as shown in Chapter 8, did not behave properly at first. The effects of this were that, basically, the same tests, under the "same conditions" would give totally different results. With modifications proposed by the author, the gib and gib fixture were remachined and the behaviour of the system load was much improved. The stick-slip effect practically disappeared and the test results were repeatable at last.

The friction level after the gib modification, at mid-stroke position, is indicated by Figure 10.2.12. It is the result of a sinewave input with a frequency of 0.05 Hz .

In order to show the effect of friction along the stroke, a test was made with a speed of $500 \mathrm{~mm} / \mathrm{min}$ moving in both
directions. Figure 10.2 .13 shows the test result where


Figure 10.2.10 Velocity x Current
(40\% rated)
extending

$10 \mathrm{~mm} / \mathrm{s}$

Figure 10.2.11 Velocity $x$ Current ( $40 \%$ rated)



Figure 10.2.12 $\frac{\text { Thrust } x \text { Speed }}{\text { (steady state) }}$


Figure 10.2.13 $\frac{\text { Thrust } x \text { Cylinder }}{\text { Stroke }}$
position signal was taken from the linear potentiometer position transducer. It is good to remember here that the friction we are talking about is the combined friction action on the piston and slideway table. Figures 10.2.14 and 10.2.15 show the necessary thrust for the same input conditions to move only the piston and piston plus table. A sinewave input signal was used. To show the pressure levels on both sides, pressure $\mathrm{P}_{1}$ is also shown in these figures.

The friction levels, considering good lubrication, are not as low as one would like to have but are the lowest we could achieve under the circumstances.

### 10.2.6 The Identified Model

The tests done so far show that, apart from the particular behaviour around null, the velocity response does not show any striking difference from symmetric electro-hydraulic cylinder drives, if we consider both directions independently. With this in mind we decided to use equation 10.24 , and the results given in Appendix 10.1, and find two models that would fit the drive response for each direction. It is of course clear, from Figures $10.2 .5,10.2 .6$ and 10.2 .7 , that we would need a model for each particular operating condition. As we are not dealing, in this work, with adaptive systems, we will choose a model for a particular condition and the control action will have to make the system insensitive to parameters variation within the operating conditions range.

The chosen operating condition was the one correspondent to $5 \%$ of rated current, input to the valve. In our case this already gives a speed of almost $2 \mathrm{~m} / \mathrm{min}$ which was thought to be a reasonable reference for practical applications.

In order to be able to use the results given in Appendix 10.1, two extra tests were carried out. They were step inputs applied to the servo-amplifier commanded by the multi-microcomputer controller. With the drive at rest, using TIBUG on Module 2, by simply writing to memory location BOO2, a command is sent to the servo-amplifier and a step generated.

With the piston at mid-stroke position, both directions of movement were tested and the results are presented in Figures 10.2.16 and 10.2.18. As the same command signal was asked for, the different gain levels for each direction of movement can be


Figure 10.2.14 Thrust, $\mathrm{P}_{1} \mathrm{x}$ time
(unloaded cylinder)


Figure 10.2.15 Thrust, $\mathrm{P}_{1} \times$.time
(loaded cylinder)


Figure 10.2.16 $\frac{\text { Velocity Step Response }}{\text { (Test Result) }}$


Figure 10.2.17 Velocity Step Response (MODEL: $\alpha=0.5 \rho=0.07$ )


Figure 10.2.18 Velocity Step Response (Test Result)


Figure 10.2.19 Velocity Step Response (MODEL: $\alpha=0.5 \rho=0.13$ )
clearly seen in those figures.
Using Appendix 10.1 and Figures 10.2 .16 and 10.2.18, the parameters for the model given by equation 10.26 were found as

$$
\begin{aligned}
& \mathrm{w}_{\ell}=28 \mathrm{~Hz}(176 \mathrm{rad} / \mathrm{sec}) \\
& \alpha=\mathrm{w}_{\ell} \times \mathrm{T}_{\mathrm{AV}}=0.5
\end{aligned}
$$

cylinder extending

$$
\rho_{\ell}=0.07
$$

cylinder retracting

$$
\rho_{\ell}=0.13
$$

Using these values, the model step response was found for both damping factors and the results are shown in Figures 10.2.17 and 10.2.19. The comparison between Figures 10.2 .16 and 10.2 .17 and Figures 10.2 .18 and 10.2 .19 shows an adequate match between the time responses of model and real system. So, the parameter values defined above will be used to define the controlled system when designing the state-variable feedback controller, later on.

In order to illustrate the effect of sudden valve closure on the velocity response, a test with a step to null velocity was done for both directions. The input was generated through module 2 using a delay loop to get the desired timing. The results are shown in Figures 10.2 .20 and 10.2.21. There, it is clearly seen that on valve closure the velocity response exhibits a much more damped behaviour for both directions. At the end of this work, the electro-hydraulic servovalve, used on the drive, was tested at the manufacturer's testing facilities according to reference 88 . The frequency response, for valve no-load flow output to current input, is shown in Figure 10.2.22. There, we see that for $45^{\circ}$ phase-lag, the correspondent frequency is 60 Hz . This proves that the recommendation made by Bell and de Pennington (84) gives a
cylinder extending

cylinder retracting


Figure 10.2.21 Valve Closure Effect on Velocity Response
good fitting for the lag ( $\mathrm{T}_{\mathrm{AV}}$ ) on equation 10.26 and that our model was chosen properly.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | - |  |  |  | Fre | quency |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 10= | $0=$ |  | 5 | + |  |  | 1 | po | atem |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | -10, | - | t- |  |  |  |  |  |  | + |  |
|  | AR: | (D) | B) | - |  |  |  |  |  |  |  |  | $\bigcirc \mathrm{PHAS}$ | E $6^{\circ}$ |
|  |  |  |  | --1 | -2-: |  |  | - | - |  |  |  | 0 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | $\cdots$ | - | T+ | - | t. |  |  |  | 1-2 |  |  |
|  |  |  | $=$ |  |  |  |  |  |  |  |  | - - | 80 |  |
|  |  |  | \% | - |  | + | - | E- |  |  | , | - | - |  |
|  |  |  | - | - |  |  | - | -2 |  |  |  | F=- |  |  |
|  |  |  |  |  |  |  |  | :- |  |  |  |  |  |  |
|  | 7- |  | - | \% | + +2 |  |  | \% |  |  | $\bigcirc$ | - | 10 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | $\square$ | - |  | - | $\because$ |  |  | A | I- AR- |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | --60 |  |
|  |  |  |  |  | :- |  |  |  |  |  |  |  | $=-$ | - |
| $\underline{ }$ |  |  | - | $=$ | Ever | C7\% |  | \% |  |  | \% | $\bigcirc 5$ | $\bigcirc$ | T |
|  |  |  |  | - | --7. |  |  |  |  |  |  |  | - |  |
| $\square$ | - |  |  | - | E- |  |  |  | - |  | 4 | - - - - |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | - | - |  |  | 7 |  |  |  |  |  |  |
|  |  |  | 36 |  |  |  |  |  |  |  |  | -1. | $=40$ |  |
|  |  |  | - | - 1 | - |  |  |  |  |  | - | IL $=:$ |  |  |
|  |  |  |  |  |  |  | - |  | S |  |  | - |  |  |
| 1 |  |  |  | - | - |  |  | $\cdots$ |  |  |  | + |  |  |
| , |  |  |  | $\cdots$ | I |  |  |  |  |  |  | E: | 30 |  |
|  |  |  |  |  | + |  |  |  |  |  |  | - |  |  |
|  |  |  | -7. | 10, | +1 | T-7 |  |  | - | \% | - | E:= = | $\bigcirc$ |  |
|  |  |  | 8 | . |  |  |  |  |  |  |  | +1-1- | 20 |  |
|  |  |  |  |  |  |  |  | T |  |  |  | - | = |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | , |  |  |  |  |  |  |  | - |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | - |  |  | , | - | - | - | E |  |  | \% | $\cdots$ |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | . |  |
|  |  |  |  | 5: | Erat | + |  |  |  |  |  | - |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  | - | Q |  |

Figure 10.2.22 No-load. flow output to current input frequency response (Moog E076-102)

### 10.3 CONTROLLER DESIGN

### 10.3.1 Model Equations in State-Space Representation

The use of state-space methods requires the system representation in the stardard matrix form

$$
\begin{align*}
& \dot{x}=A x+B u \\
& y=C x
\end{align*}
$$

where $u$ is the control input, $x$ the state variables and $y$ the output.

For the digital control problem equations 10.27 and 10.28 are transformed to difference equations using the arrangement shown in Figure 10.3.1.


Figure 10.3.1 Controlled System and Sampling

In our case the controlled system and the hold element are represented as in Figure 10.3.2. The hold action is applied by the digital to analogue converter (DAC) while it holds the control input constant during the sampling period.


Figure 10.3.2 Servodrive and Hold

The DAC has a gain, of course, relating maximum output voltage to number of bits. Calling one bit as a DAC Unit or just Unit the DAC gain for the range set-up in the rig is given by

$$
K_{D A C}=\frac{10}{2^{11}} \quad \frac{V}{U n i t}
$$

or

$$
\mathrm{K}_{\mathrm{DAC}} \cong 4.9 \mathrm{mV} / \mathrm{Unit}
$$

This gain can be incorporated into the gain for the drive and load, $K_{v}$, and equation 10.26 can be rewritten as

$$
\frac{s y(s)}{u(s)}=\frac{v(s)}{u(s)}=\frac{K}{\left(1+s^{T} A V\right)\left(\frac{s^{2}}{w^{2}}+\frac{2 \rho_{\ell}}{w_{\ell}} s+1\right)} \quad 10.30
$$

where

$$
\mathrm{K}=\mathrm{K}_{\mathrm{DAC}} \times \mathrm{K}_{\mathrm{v}}
$$

As mentioned before, the only output we want to use is position so equation 10.30 is modified to give the required transfer function, between position ( $y$ ) and control input ( $u$ ), as

$$
\frac{y(s)}{u(s)}=\frac{K}{s\left(1+s T_{A V}\right)\left(\frac{s^{2}}{w_{\ell}^{2}}+\frac{2 \rho_{\ell}}{w_{\ell}} s+1\right)}
$$

Defining state-variables as

$$
\begin{align*}
\text { position: } & x_{1}=y \\
\text { velocity: } & x_{2}=\frac{\dot{x}_{1}}{w_{\ell}} \\
\text { acceleration: } & x_{3}=\frac{\dot{x}_{2}}{w_{\ell}^{2}} \\
\text { transient } & x_{4}=\frac{\dot{x}_{3}}{w_{\ell}^{3}}
\end{align*}
$$

and variable

$$
\alpha=w_{\ell} \times T_{A V}
$$

the transfer function given by 10.31 can be transformed to standard matrix form $\dot{x}=A x+B u$ to give

$$
\frac{1}{\mathrm{w}}\left[\begin{array}{c}
\dot{x}_{1} \\
\dot{x}_{2} \\
\dot{x}_{3} \\
\dot{x}_{4}
\end{array}\right]=\left[\begin{array}{cccc}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & -\frac{1}{\alpha} & -\left(\frac{2 \rho}{\alpha}+1\right)-\left(\frac{1}{\alpha}+2 \rho\right)
\end{array}\right]\left[\begin{array}{l}
x_{1} \\
x_{2} \\
x_{3} \\
x_{4}
\end{array}\right]+\left[\begin{array}{l}
0 \\
0 \\
0 \\
\frac{K}{w \alpha}
\end{array}\right]
$$

The subscripts under the load natural frequency $w_{l}$ and damping $\rho_{\ell}$ have been taken out for easier representation. They are now represented simply as $w$ and $\rho$.

Representation 10.32 can be simply written as

$$
\frac{1}{w} \dot{x}=A x+B u^{*}
$$

where $u^{*}=\left(\frac{K}{w} u\right)$ and

$$
\mathrm{B}=\left[\begin{array}{c}
0 \\
0 \\
0 \\
\frac{1}{\alpha}
\end{array}\right]
$$

Equation 10.33 is in the appropriate form to be used with the STATE Program as defined in Chapter 4 to find the system's discrete state equations as

$$
x(k+l)=\Phi x(k)+L u(k)
$$

with output given by

$$
y(k)=C x(k)
$$

### 10.3.2 Sampling Period and Delay

The use of the STATE Program requires the definition of two variables: the sampling period, $T$, and the delay on the control function, $T_{D}$. If a change in time scale is made such that

$$
t^{*}=\omega t
$$

$T$ and $T_{D}$ are changed to

$$
T^{*}=w T
$$

and

$$
T_{D}^{*}=W T_{D}
$$

For the sampling frequency, $f$,

$$
\mathrm{f}=\frac{1}{\mathrm{~T}}
$$

and the load natural frequency, w,

$$
\dot{\mathrm{w}}=2 \pi \mathrm{f}_{\ell}(\mathrm{rad} / \mathrm{sec}) \quad 10.40
$$

where $f_{\ell}(\mathrm{Hz})$. Equation 10.37 can be rewritten as

$$
T^{*}=2 \pi \frac{\mathbf{f}_{\ell}}{\mathrm{f}}
$$

Assuming10.42
we have

$$
\mathrm{f}>6 \mathrm{f}_{\ell}
$$

which means that the sampling frequency is more than 6 times the load natural frequency ( Hz ) . According to results shown by Middleditch (89) and Stute (39) equation 10.42 is compatible with a wide range of drive characteristics, but we do not try to prove it in this work.

Equation 10.42 introduces a sampling period given by

$$
T \leqslant \frac{1}{w}
$$

which in our case for $w=176 \mathrm{rad} / \mathrm{sec}$ gives

$$
T \leqslant 5.7 \mathrm{msec}
$$

For maximum computing time between samples, the sampling period should be as long as possible and so we decided to use, in this work,

This would give, in our case, a sampling period of around 5.7 msec which is compatible with the software for closing the loop, developed so far, which runs in approximately 3 msec .

The other advantage in using equation 10.45 will become clear when the estimation of velocity from position information is described later on.

The delay on the control function, $T_{D}$, is used to account for the computation delay in the digital controller, but can be made to include any delay of control action on the controlled system itself. In our case, the results presented earlier do not show any significant delay on the controlled system and so $T_{D}$ represents only computational delay.

In this work we assumed a delay equal to a complete sampling period, or

$$
T_{D}=T
$$

for two main reasons. The first, and most important, relates to the actual computation delay. A change in the controller software will probably change the computation delay but if the delay is kept smaller than a sampling period, the system difference equations still hold and no adverse effects should be expected, because the controller equations are related to the system difference equations. The second reason is that the choice of delay given by relation 10.46 simplifies the representation and solution of system and observer difference equations, as will be clearly seen later on.

### 10.3.3 Discrete State-Equations

Using equations 10.45 and 10.46 , the discrete state equations are found, applying the STATE Program, as developed in Chapter 4, to the continuous state equation 10.33.

When the cylinder drive is extending we have

$$
\rho=0.07
$$

and

$$
\alpha=0.5
$$

which gives matrices $A_{1}$ and $B_{1}$ as

$$
\begin{gather*}
A_{1}=\left[\begin{array}{cccc}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & -2 & -1.28 & -2.14
\end{array}\right] \\
B_{1}=\left[\begin{array}{l}
0 \\
0 \\
0 \\
2
\end{array}\right]
\end{gather*}
$$

10.48

Using equations $10.45,10.46$, 10.47 and 10.48 as inputs to the STATE Program, the resulting discrete state equations are given in the form

$$
x_{e}(k+1)=\Phi_{1} x(k)+L_{1} u^{*}(k)
$$

With $\Phi_{1}$ and $L_{1}$ taken from Appendix 10.2 , we have

|  | 1.0000 | Ø. 9451 | 0.4531 | ø. 0984 | 0.0549 | 10.50 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Ø. øøøø | 0.8033 | 0.8192 | 0.2426 | ø. 1967 |  |
| $\Phi_{1}=$ | Ø. $0 \varnothing \square \varnothing$ | -0.4853 | 0.4927 | 0.3000 | 0.4853 |  |
|  | 0.0000 | -0.6000 | - 0.8693 | -0.1493 | 0.6000 |  |
|  | Ø. $0 \varnothing \varnothing \square$ | ø.0øøø | Ø.øøøø | 0.0000 | 0.0000 |  |
|  |  |  |  |  |  |  |

and
where the state vector, $x_{e}(k)$, is now

$$
x_{e}(k)=\left[\begin{array}{c}
x_{1}(k) \\
x_{2}(k) \\
x_{3}(k) \\
x_{4}(k) \\
u *(k-1)
\end{array}\right]
$$

When the cylinder is retracting we have

$$
\rho=0.13
$$

and

$$
\alpha=0.5
$$

Matrices $A_{2}$ and $B_{2}$ for equation 10.33 are now represented as

$$
\begin{aligned}
\mathrm{A}_{2}= & {\left[\begin{array}{cccc}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & -2 & -1.52 & -2.26
\end{array}\right] } \\
\mathrm{B}_{2}=\left[\begin{array}{l}
0 \\
0 \\
0 \\
2
\end{array}\right] & 10.53
\end{aligned}
$$

Using equations $10.45,10.46,10.53$ and 10.54 , as inputs to the STATE Program, the resulting discrete state equations are given in the form

$$
x_{r}(k+1)=\Phi_{2} x(k)+L_{2} u^{\star}(k)
$$

With $\Phi_{2}$ and. $L_{2}$ taken from Appendix 10.3, we have
and

$$
L_{2}=\left[\begin{array}{l}
0.0000 \\
0.0000 \\
0.0000 \\
0.0000 \\
1.0000
\end{array}\right]
$$

where the state vector, $x_{r}(k)$, is now

$$
x_{r}(k)=\left[\begin{array}{c}
x_{1}(k) \\
x_{2}(k) \\
x_{3}(k) \\
x_{4}(k) \\
u *(k-1)
\end{array}\right]
$$

In this work, as mentioned before, we only use position information available through the linear optical position transducer. We assume that. the other state-variables such as velocity and acceleration are not available by direct measurement. For both directions of movement, the output equation is so given by

$$
y(k)=\left[\begin{array}{lllll}
1 & 0 & 0 & 0 & 0
\end{array}\right] x(k) \quad 10.59
$$

where $x(k)$ is either $x_{e}(k)$ or $x_{r}(k)$ as defined by equations 10.52 and 10.58 .

### 10.3.4 Control Law Definition

For the control law as a linear combination of all the states, we have

$$
u(k)=-K x(k)
$$

where the gain matrix $K$ is found in this work by using optimal control. Equation 10.60 is shown, applied to our case, in the diagram given in Figure 10.3.3. There, it is assumed that all the state-variables are available for direct measurement and the sampler indicates the discrete action of the digital controller. In our case, we do not have all the state-variables available but the separation principle allows us to find the feedback gains $K_{i}$, independently of any estimation involved (observer).


Figure 10.3.3 State-Variable Feedback

The performance index for the optimal controller is defined, as in Chapter 3, by
$I P=\frac{1}{2} \sum_{k=0}^{N}\left[x^{T}(k) Q_{1} x(k)+u^{T}(k) Q_{2} u(k)\right]$

The feedback gains must be chosen in order to minimise IP subject to the constraints imposed by equation 10.34 (state-equation).

The solution for this two-point boundary-value problem has already been discussed and the sweep method developed by Bryson and Ho (59) is used. The program called SWEEP (Chapter 4) helps us find the possible candidates for the minimum solution but first we must decide about the weighting matrices $Q_{1}$ and $Q_{2}$.

For the single-input system, dealt with in this work, matrix $Q_{2}$ is $1 \times 1$ or a scalar. Some weight is always selected for the control function, otherwise the solution will include large components in the control gain, which is bound to cause saturation. Results from the simulation will indicate the best choice for $Q_{2}$ in terms of a reasonable control effort.

Matrix $Q_{1}$ indicates the relative importance given to the various states. As we are primarily interested in improving load damping the most indicative of the state-variables is velocity ( $x_{2}$ ). Since a good velocity behaviour almost certainly indicates well behaved acceleration and transient acceleration, it is only necessary to give weight to velocity.

As position is just an effect of other state-variables and control function, as shown by equations 10.55 and 10.56 , it is really not necessary to feed it back just to achieve load damping as will be clearly shown when its weight on $Q_{1}$ is set to zero. With an improved load damping, we will close an external position loop around the load and will have the chance to compare the behaviour of a simple digital proportional loop with that of a digital proportional loop with state-variable feedback improving load damping.

### 10.3.5 Optimal Controller

Using the already defined discrete-state equations, for both directions of movement, as inputs to the SWEEP program, defined in Chapter 4, we can find the possible solutions for the optimal
feedback gains. A typical SWEEP sample is shown in Appendix 10.4. As shown there, the interactive style allows many solutions to be obtained very quickly.

As recommended by Bryson and Ho (59) the weight on the velocity variable for matrix $Q_{1}$ was chosen as follows:

The state-variable $x_{2}$ is defined as

$$
x_{2}=\frac{v}{w}
$$

where $v$ is velocity and $w$ the load natural frequency. Assuming a maximum speed of around $200 \mathrm{~mm} / \mathrm{sec}$, we have

$$
x_{2(\max )} \stackrel{\cong}{\cong} 1
$$

and so

$$
Q_{1}(2,2)=\frac{1}{x_{2}^{2}(\max )}=1
$$

For matrix $Q_{1}$ defined as
$Q_{1}=\left[\begin{array}{lllll}0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0\end{array}\right]$

The steady-state results using SWEEP for both directions of movement are shown in Tables 10.1 and 10.2. The feedback gains are related to the defined state-variables as shown in Figure 10.3.3.

|  | Extending Cylinder |  |  |  |  |  |
| :--- | ---: | :---: | :---: | :---: | :---: | :---: |
| Test | $Q_{2}$ | $\mathrm{~K}_{1}$ | $\mathrm{~K}_{2}$ | $\mathrm{~K}_{3}$ | $\mathrm{~K}_{4}$ | $\mathrm{~K}_{5}$ |
| 1 | 0.01 | 0.0 | -0.55 | 1.05 | 0.67 | 1.50 |
| 2 | 0.1 | 0.0 | -0.58 | 0.47 | 0.38 | 0.98 |
| 3 | 0.5 | 0.0 | -0.48 | 0.12 | 0.18 | 0.57 |
| 4 | 1 | 0.0 | -0.40 | 0.04 | 0.12 | 0.41 |
| 5 | 5 | 0.0 | -0.20 | -0.03 | 0.03 | 0.16 |
| 6 | 10 | 0.0 | -0.13 | -0.03 | 0.02 | 0.10 |
| 7 | 20 | 0.0 | -0.08 | -0.02 | 0.01 | 0.06 |
| 8 | 100 | 0.0 | -0.02 | -0.01 | 0.00 | 0.01 |

Table 10.1

|  | Retracting Cylinder |  |  |  |  |  |
| ---: | ---: | ---: | :---: | :---: | :---: | :---: |
| Test | $Q_{2}$ | $K_{1}$ | $K_{2}$ | $K_{3}$ | $K_{4}$ | $K_{5}$ |
| 9 | 0.01 | 0.0 | -0.39 | 1.08 | 0.64 | 1.45 |
| 10 | 0.1 | 0.0 | -0.45 | 0.49 | 0.36 | 0.93 |
| 11 | 0.5 | 0.0 | -0.38 | 0.15 | 0.17 | 0.52 |
| 12 | 1 | 0.0 | -0.31 | 0.06 | 0.11 | 0.36 |
| 13 | 5 | 0.0 | -0.14 | -0.01 | 0.03 | 0.13 |
| 14 | 10 | 0.0 | -0.08 | -0.01 | 0.02 | 0.07 |
| 15 | 20 | 0.0 | -0.05 | -0.01 | 0.01 | 0.04 |
| 16 | 100 | 0.0 | -0.01 | -0.00 | 0.00 | 0.01 |

Table 10.2

The root locus for both sets of tests are shown in Figures 10.3.4 and 10.3.5. The improvement in load damping is quite clear.

The influence of $Q_{2}$ on the speed of response is clearly seen as the load poles move, initially, towards higher dampings. Increasing $Q_{2}$ makes the response slower as it imposes a greater restriction on the control effort.

TEST 1


Gain Incremen
0.05

TEST 4


Figure 10.3.4 Root Locus for Optimal Control

TEST 9


Gain Increment
0.05

TEST 14
cylinder retracting
TEST 10


Figure 10.3.5 Root Locus for Optimal Control

The velocity and control function time response for a step input using the optimum gains, given in Tables 10.1 and 10.2, are shown in Figures 10.3.6, 10.3.7, 10.3 .8 and 10.3.9. For small values of $Q_{2}$, as mentioned before, the velocity response is fast but the control function exhibits quite large overshoots, which means a large control effort. Such behaviour can cause saturation and should be avoided. For higher values of $Q_{2}$ the control effort is reduced but the velocity response becomes less and less damped.

The compromise between speed of response and control effort, with an acceptable level of load damping, made us decide to use the optimum gains associated with tests 4 and 12 as the starting point for the state-variable feedback controller.

### 10.3.6 Velocity Estimation

Among the defined state-variables, only position is available by direct measurement, as was already mentioned. The implementation of a control law, as given by equation 10.60 , requires all the state-variables and so the non-available variables must be estimated. In this work estimation is achieved by the use of observers as described in Chapter 3.

The non-available variables in our case are velocity, acceleration and transient acceleration. Before we proceed to the observer design we will discuss the use of position information for velocity estimation.

The use of a linear optical position transducer makes the feedback signal free of noise, for practical purposes. This noiseless signal can be differentiate to provide information about velocity, without expecting the type of problem which is normal with pure analog differentiators. Looking carefully at the position information provided by an incremental transducer it is easy to see that, what we really have is a certain increment in position within a certain time or, more explicitly, velocity. Position information is achieved by adding (integration) position increments as is done by the counter in our digitizer feedback board. The sampling interval fixes the time in which the summation is done and relates position increment

TEST 1



TEST 2



Figure 10.3.6 Velocity Step Response

TEST 4



TEST 6



TEST 9


$$
\times 1 a^{-4}
$$





Figure 10.3.8 Velocity Step Response


$\frac{t}{T}$

TEST 14


$\frac{t}{T}$
$\frac{t}{T}$
Figure 10.3.9 Velocity Step Response
to velocity.
At sampling instant $k$, the average velocity over the sampling interval is given by

$$
v(k)=\frac{C_{t}(k) R_{e}}{T}
$$

```
where \(v=\) average.velocity (mm/sec)
    \(\mathrm{T}=\) sampling period (msec)
    \(R_{e}=\) feedback resolution ( \(\mu \mathrm{m}\) )
    \(C_{t}=\) total counter units (unit)
```

Equation 10.64 indicates that the velocity resolution, or the smallest (non-null) increment in velocity which can be measured, results from a change in the counter of one unit, that is

$$
R_{v}=\frac{R_{e}}{T}
$$

Velocity resolution improves as the sampling period increases or the position resolution is reduced.

In our case, for a feedback resolution of $1 \mu \mathrm{~m}$ and a sampling period of around 5 msec , the velocity resolution is

$$
R_{v}=\frac{1}{5}=0.2 \frac{\mathrm{umm}}{\mathrm{sec}}
$$

For velocities higher than $10 \mathrm{~mm} / \mathrm{sec}$ this resolution implies an error of less than $5 \%$ on the measured velocity and was thought to be an acceptable error in this work.

Using equation 10.64 , state-variable $x_{2}$, at instant $k$, can be written

$$
x_{2}(k)=\frac{v(k)}{w}=\frac{C_{t}(k)}{w T}
$$

for a feedback resolution ( $R_{e}$ ) of $1 \mu \mathrm{~m}$. Using equation 10.45 we have

$$
x_{2}(k)=\frac{v(k)}{w}=c_{t}(k)
$$

which means that $x_{2}(k)$ equals the position counter contents at instant $k$ without any manipulation. This is a very simple and reasonably accurate way to estimate the velocity related state-variable and is therefore used in this work.

### 10.3.7 Reduced-Order Observer Design

### 10.3.7.1 Matrix Partition and Observability

With state-variable $x_{2}$ defined by equation 10.67 , in terms of observer design it can be assumed as an available variable. The observer will be designed to give an estimate of acceleration and transient acceleration using $x_{2}$ as a measurable output. The observer design will be completely described for the extending cylinder model and only results will be given for the retracting direction, for the procedures are the same.

Removing $x_{1}$ from state-vector $x_{e}(10.52)$ and ordering with the available state-variables on top, we have a modified state-verctor

$$
x_{e m}(k)=\left[\begin{array}{c}
x_{2}(k) \\
u *(k-1) \\
x_{3}(k) \\
x_{4}(k)
\end{array}\right]
$$

with state-equations defined by
$\Phi_{1 \mathrm{~m}}=\left[\begin{array}{rrrr}0.8033 & 0.8192 & 0.2496 & 0.1967 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.4853 & 0.4927 & 0.3000 & 0.4853 \\ -.0 .6000 & -0.8693 & -0.1493 & 0.6000\end{array}\right]$
and

$$
\mathrm{L}_{\mathrm{lm}}=\left[\begin{array}{c}
0.0 \\
1.0 \\
0.0 \\
0.0
\end{array}\right]
$$

10.70

The output is now given by
$y_{m}(k)=\left[\begin{array}{llll}1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0\end{array}\right] x_{e m}(k)$
which indicates a multi-output system due to the delay action. Writing $\Phi_{1 m}$ and $L_{1 m}$ in the partitioned form, for reduced-order observer design, as
$\Phi_{1 \mathrm{~m}}=\left[\begin{array}{cc}\Phi_{\mathrm{aa}} & \Phi_{\mathrm{ab}} \\ \Phi_{\mathrm{ba}} & \Phi_{\mathrm{bb}}\end{array}\right]$
and

$$
\mathrm{L}_{1 \mathrm{~m}}\left[\begin{array}{l}
\mathrm{L}_{\mathrm{a}} \\
\mathrm{~L}_{\mathrm{b}}
\end{array}\right]
$$

we have
$\begin{array}{ll}\Phi_{\mathrm{aa}} & =\left[\begin{array}{rr}0.8033 & 0.8192 \\ 0.0000 & 0.0000\end{array}\right] \quad \Phi_{\mathrm{ab}}=\left[\begin{array}{ll}0.2496 & 0.1967 \\ 0.0000 & 0.0000\end{array}\right] \\ \Phi_{\mathrm{ba}}=\left[\begin{array}{ll}-0.4853 & 0.4927 \\ -0.6000 & -0.8693\end{array}\right] \quad \Phi_{\mathrm{bb}}=\left[\begin{array}{ll}0.3000 & 0.4853 \\ -0.1493 & 0.6000\end{array}\right]\end{array}$
and

$$
\mathrm{L}_{\mathrm{a}}=\left[\begin{array}{l}
0 \\
1
\end{array}\right] \quad \mathrm{L}_{\mathrm{b}}=\left[\begin{array}{l}
0 \\
0
\end{array}\right]
$$

The check for observability shows that

$$
\operatorname{rank}\left[\Phi_{a b}^{T} \quad \Phi_{b b}^{T} \Phi_{a b}^{T}\right]=2
$$

Therefore an observer of second order exists.

### 10.3.7.2 Observer Design by Pole-Placement Method

The observer feedback gain matrix, LO, Chapter 3 , is set to satisfy the chosen characteristic equation

$$
\alpha_{e}(z)=\left|z I-\Phi_{b b}+L 0 . \Phi_{a b}\right|
$$

The particular characteristic of matrix $\Phi_{a b}$ allows a simplification on the feedback gain matrix, LO, which allows us to use the pole-placement design through the ACKER program, defined in Chapter 4, as if the system were a single-output one. From equations $10.73, \Phi_{a b}$ is given in general form as

$$
\Phi_{a b}=\left[\begin{array}{cc}
m & n \\
0 & 0
\end{array}\right]
$$

For LO given by

$$
\mathrm{LO}=\left[\begin{array}{ll}
\mathrm{LO}_{11} & \mathrm{LO}_{12} \\
\mathrm{LO}_{21} & \mathrm{LO}_{22}
\end{array}\right]
$$

the product $L O . \Phi_{a b}$ is

$$
\mathrm{LO} . \Phi_{a b}=\left[\begin{array}{ll}
\mathrm{mLO}_{11} & \mathrm{nLO}_{11} \\
\mathrm{mLO}_{21} & \mathrm{nLO}_{21}
\end{array}\right]
$$

which indicates that the second row of $\Phi_{a b}$ does not have any influence on the characteristic equation $\alpha_{e}$. Therefore we can reduce the order of $\Phi_{a b}$ to $1 \times 2$ to have

$$
\Phi_{\mathrm{abr}}=\left[\begin{array}{ll}
\mathrm{m} & \mathrm{n}
\end{array}\right]
$$

from the general form of $\Phi_{a b}$. This reduces the system to single output and the pole-placement design can now be done using the ACKER program.

The characteristic equation, given by 10.75 , is first modified to read

$$
\alpha_{e}(z)=\left|z I-\Phi_{b b}^{T}+\Phi_{a b r}^{T} \cdot L 0_{r}^{T}\right|
$$

where

$$
{ }^{\mathrm{LO}} \mathrm{r}=\left[\begin{array}{c}
\mathrm{LO}_{11} \\
\mathrm{LO}_{21}
\end{array}\right]
$$

and it indicates the input format for the ACKER program. The output from this program is the observer feedback gain matrix ${ }^{\mathrm{L}}{ }_{r}$.

For our application, we decided to place the observer poles at zero location on the $z$-plane, that is, to have a deadbeat observer and verify its behaviour under noise and modelling errors. The pole-placement design program (ACKER) asks for desired poles on the s-plane and, for a deadbeat design on the $z-$ plane, we need $s \rightarrow-\infty$. The use of the ACKER program showed that the feedback gains reach steady-state values for increasing negative $s$ values long before $s \longrightarrow-\infty$.

Using the matrices as defined by 10.80 , in the ACKER program, as in Appendix 10.5, gives

$$
\mathrm{LO}_{\mathrm{r}}=\left[\begin{array}{r}
0.7107 \\
-0.9845
\end{array}\right]
$$

With

$$
\begin{aligned}
\mathrm{LO}_{12} & =0 \\
\mathrm{LO}_{22} & =0
\end{aligned}
$$

the complete observer feedback gain matrix is written

$$
L O=\left[\begin{array}{rr}
0.7107 & 0.0000 \\
-0.9845 & 0.0000
\end{array}\right]
$$

If the modified state vector (10.68) is written

$$
x_{e m}(k)=\left[\begin{array}{l}
x_{a}(k) \\
x_{b}(k)
\end{array}\right]
$$

and the output

$$
y_{m}(k)=\left[\begin{array}{ll}
I & 0
\end{array}\right]\left[\begin{array}{l}
x_{a}(k) \\
x_{b}(弓)
\end{array}\right]
$$

with

$$
x_{a}(k)=\left[\begin{array}{c}
x_{2}(k) \\
u *(k-1)
\end{array}\right], \quad x_{b}(k)=\left[\begin{array}{l}
x_{3}(k) \\
x_{4}(k)
\end{array}\right]
$$

the observer equations are given by

$$
\begin{align*}
w(k+1) & =\left[\Phi_{b b}-\text { LO } \Phi_{a b}\right] w(k)+ \\
& +\left[\left(\Phi_{b a}-\text { LO } \Phi_{a a}\right)+\left(\Phi_{b b}-L 0 \Phi_{a b}\right) L 0\right] x_{a}(k)+ \\
& +\left[L_{b}-L 0 L_{a}\right] u^{*}(k)
\end{align*}
$$

and

$$
\hat{x}_{b}(k)=w(k)+\operatorname{LO} x_{a}(k)
$$

where $\hat{x}_{b}$ is the observer estimated value for vector $x_{b}$ (nonavailable variables).

The coefficient matrices for equation 10.85 are found using $10.69,10.70$ and 10.82 on program OBSEQU (Chapter 4) which in our case, according to Appendix 10.6, gives

$$
\begin{aligned}
w(k+1)= & {\left[\begin{array}{cc}
-0.0895 & 0.1276 \\
-0.0628 & 0.0895
\end{array}\right] \mathrm{w}(\mathrm{k})+\left[\begin{array}{rr}
-1.2454 & 0.3455 \\
0.0581 & 0.7937
\end{array}\right] \mathrm{x}_{\mathrm{a}}(\mathrm{k}) } \\
& +\left[\begin{array}{c}
0.0 \\
0.0
\end{array}\right] \mathrm{u}^{*}(\mathrm{k})
\end{aligned}
$$

Now, the complete set of observer equations, for the extending cylinder, can be written in expanded form as
$w_{1}(k+1)=-0.0895 w_{1}(k)+0.1276 w_{2}(k)-1.2454 x_{2}(k)+0.3455 u *(k-1)$
$w_{2}(k+1)=-0.0628 w_{2}(k)+0.0895 w_{2}(k)+0.0581 x_{2}(k)+0.7937 u *(k-1)$
and

$$
\begin{aligned}
& \hat{x}_{3}(k)=w_{1}(k)+0.7107 x_{2}(k) \\
& \hat{x}_{4}(k)=w_{2}(k)-0.9845 x_{2}(k)
\end{aligned}
$$

Using the same procedures described for the extending cylinder, the observer equations for the retracting cylinder are found as
$w_{1}(k+1)=-0.0745 w_{1}(k)+0.1227 w_{2}(k)-1.1785 x_{2}(k)+0.3382 u *(k-1)$
$w_{2}(k+1)=-0.0453 w_{1}(k)+0.0746 w_{2}(k)+0.1866 x_{2}(k)+0.7530 u *(k-1)$
and
$\hat{x}_{3}(k)=w_{1}(k)+0.6623 x_{2}(k)$
$\widehat{x}_{4}(k)=w_{2}(k)-1.0477 x_{2}(k)$

### 10.3.8 Combined Control Law and Observer

After designing the observers, we are now able to simulate and compare the behaviour of a digital proportional position loop with and without state-variable feedback for load damping.

The general structure for the controller comes out from a slight modification on Figure 10.3 .3 to include the observer and external position loop. Figure $\mathbf{1 0 . 3} \mathbf{. 1 0}$ shows this modified structure. From this figure it is clear that a simple digital proportional position loop, without state-variable feedback for load damping, is achieved by simply setting gains $K_{2}$ to $K_{5}$ equal to zero.

In order to simulate the complete system, the state-space must be expanded to include the new variables introduced by the observer. The general state-equations in the expanded state-space can be written, using equations $10.49,10.83$ and 10.85 , as

where $\Phi_{a a}, \Phi_{a b}, \Phi_{b a}, \Phi_{b b}$, and $L_{a}$ and $L_{b}$ are as defined in 10.72 , and $\Phi_{11}$ and $\Phi_{12}$ come from 10.49 or 10.55 .


Figure 10.3.10 Digital Proportional Position Loop with State-Variable Feedback and Observer

From 10.85, we have
$\Omega=\Phi_{b a}-L O \Phi_{a a}+\left(\Phi_{b b}-L O \Phi_{a b}\right) L O$
$\theta=\Phi_{b b}-L 0 \Phi_{a b}$
and
$M=L_{b}-L 0 L_{a}$

The observer estimated state-variable equations, given by 10.86 , can be written in the new expanded space as
$x_{b}(k)=\left[\begin{array}{lll}0 & \text { LO } & 0\end{array}\right]\left[\begin{array}{l}x_{1}(k) \\ x_{a}(k) \\ x_{b}(k) \\ w(k)\end{array}\right]$
where we define matrix

LOBLW $=\left[\begin{array}{llll}0 & \text { LO } & 0 & I\end{array}\right]$ 10.94
to be used as input on the simulation program.
The control function, as indicated in Figure 10.3.10, is now given by
$u *(k)=K_{1} r(k)-\left[K_{1} x_{1}(k)+K_{2} x_{2}(k)+K_{3} \hat{x}_{3}(k)+K_{4} \hat{x}_{4}(k)+K_{5} u *(k-1)\right]$ 10.95

Equation 10.95 shows that $u^{*}$ cannot be represented as a simple linear combination of state-variables in the expanded space. For this reason, we separate the gain matrix into two, for the available and estimated variables. The control function is therefore given as
$u *(k)=K_{1} r(k)-\left[K M_{1} x_{x}(k)+K M_{2} \hat{x}_{b}(k)\right]$
where

$$
x_{x}(k)=\left[\begin{array}{c}
x_{1}(k) \\
x_{a}(k) \\
x_{b}(k) \\
w(k)
\end{array}\right]
$$

and in our case

$$
\mathrm{KM}_{1}=\left[\begin{array}{lllllll}
\mathrm{K}_{1} & \mathrm{~K}_{2} & \mathrm{~K}_{5} & 0 & 0 & 0 & 0
\end{array}\right]
$$

and

$$
\mathrm{KM}_{2}=\left[\begin{array}{ll}
\mathrm{K}_{3} & \mathrm{~K}_{4}
\end{array}\right]
$$

### 10.3.9 Closed-Loop.Simulation

The simulation was executed only for the extending cylinder model as we are really interested in the behaviour of the real system, which is available for test. Simulation will help to show if there are problems on the controller design and if any improvement is achieved by the control action on the linear model.

The position loop was first simulated only with a proportional digital controller and the root-locus and velocity response for a ramp input (fixed displacement) are shown in Figure 10.3.11. From the root-locus it is clear that, for a gain $K_{1} \cong 0.4$, the drive will become unstable. The sensitivity to variation in the location of the load poles is quite high as these poles move very close to the unit circle.

The position loop simulation results for controller with statevariable feedback is shown in Figure 10.3.12.

The sensitivity to variation in the location of the load poles is reduced and now the system has a dominant pair of poles which slow it down but improve the velocity response. For gain $K_{1}>0.1$ the position loop will have a natural frequency ( $w_{p}$ ) given approximately by

$$
W_{p} T=0.3
$$

where $T$ is the sampling period. As we started with a load




VAR
VAR
TIME


$$
x_{10} 0^{-4} \quad K_{1}=0.3
$$

$$
\text { VAR }=\text { Velocity }
$$

$$
\operatorname{TIME}=\frac{\mathrm{t}}{\mathrm{~T}}
$$



Figure 10.3.11 Proportional Position Loop (Simple)


Figure 10.3.12 Proportional Position Loop and State-Variable Feedback
natural frequency given by $w T=1.0$, we see that the position loop will have one third of the open-loop natural frequency.

### 10.4 CONTROLLER EQUATIONS - SOFTWARE IMPLEMENTATION

The controller software can now be completed with the implementation of the control function and observer equations. They are developed here in the order they are required in the closing the loop routine.

After the available state-variables have been sampled by the controller, the non-available variables are estimated by the observer through the general equation 10.86 . This must be done before the control function is calculated, for the non-available variables are also necessary.

In our case, the estimated values are given in general form as

$$
\begin{align*}
& \hat{x}_{3}(k)=\frac{K A 1}{K A 2}\left[K W 1 w_{1}(k)+K A x_{2}(k)\right] \\
& \hat{x}_{4}(k)=\frac{K D A 1}{K D A 2}\left[K W 2 w_{2}(k)+\mathrm{KDAx}_{2}(k)\right]
\end{align*}
$$

where KA1, KA2, KDA and KDA2 are integers, which allow constants KW1, KA, KW2 and KDA to have modulus less or equal to 1 . With this procedure the fractional constants are represented as single precision fixed point (15) binary numbers in the controller memory. The implementation, in assembly level language, of equations 10.99 and 10.100 , is shown in Appendix 10.7.

After the estimated variables are calculated, the control equation can be solved. As already described, in the loop closing routine, the position error is updated every sampling interval and according to Figure 10.3.10 it is

$$
e(k)=r(k)-x_{1}(k)
$$

Using the same procedure as for the estimated values, equation 10.95 is given in general form as
$u^{*}(k):=\frac{K P 1}{K P 2}\left[K E e(k)+K V x_{2}(k)+K A \hat{\mathbf{x}}_{3}(k)+K A T \hat{x}_{4}(k)+K U u *(k-1)\right]$ 10.102
where $K P 1$ and $K P 2$ are integers.
The assembly level language implementation of equation 10.102 is shown in Appendix 10.8.

After the control function is found and sent out through the DAC, the observer state-equation, (10.85) must be solved for the next sampling period. According to equations 10.87 and 10.88 , the general forms for our case are
$w_{1}(k+1)=\frac{\text { KOB1A }}{\text { KOB1B }}\left[K 1 W 1 w_{1}(k)+K 2 W 1 w_{2}(k)+K 1 V x_{2}(k)+K 1 U K u^{*}(k-1)\right]$
$w_{2}(k+1)=\frac{K O B 2 A}{\text { KOB2B }}\left[K 2 W 1 w_{1}(k)+K 2 W 2 w_{2}(k)+K 2 V x_{2}(k)+K 2 U K u *(k-1)\right]$
10.103
which implementation is shown in Appendix 109.
Before we finish this chapter, the relationship between the control functions $u$ and $u^{*}$ must be discussed. As for equation 10.33

$$
u^{\star}=\frac{K}{w} u \quad \text { or } \quad u=\frac{w}{K} u^{\star}
$$

where $w$ is the load natural frequency and $K$ the open-loop gain. As the actual command function is $u$, we have to modify the output of equation 10.102 before it is sent out to the DAC. This modification allows us also to increase the open-loop gain by controller action. This is done by making

$$
u(k)=\frac{K U C O R 1}{\operatorname{KUCOR} 2} u^{*}(k)
$$

where KUCOR1 and KUCOR2 are integers. The modification action is shown by the diagram in Figure 10.3.13.


Figure 10.3.13 Control Function Modification

## CHAPTER 11

## EXPERIMENTAL TESTS AND RESULTS

### 11.1 INTRODUCTION

This chapter describes the tests carried out to compare the behaviour of the electro-hydraulic cylinder drive under a simple digital proportional controller and a state-variable controller.

First, the open-loop gain is determined and then the coefficients for the controller equations are found.

Two types of position reference input are used: ramp and step input. The results obtained, for both controller actions, are shown in Section 11.4. Section 11.5 contains discussions about the results obtained.

### 11.2 OPEN-LOOP GAIN

The control function, as shown in Chapter 10 , requires the open-loop gain in order to define its coefficients. The results, already obtained for the open-loop tests, could be used for this end but the required constant can be obtained more directly for the whole range of operations using the controller software.

If we exclude the gain of the controller (software) the open-loop gain is given by the product of the three main components

$$
K=K_{D A C} \times K_{A m P} \times K_{V C y}
$$

where
$K_{\text {DAC }}\left[\frac{\mathrm{V}}{\text { unit }}\right]$ digital-analog converter gain
$K_{\text {Amp }}\left[\frac{m A}{V}\right]$ servo-amplifier gain
$K_{v c y}\left[\frac{m \mathrm{~m} / \mathrm{sec}}{\mathrm{mA}}\right]$ servovalve-cylinder combined gain

### 11.2.1 DAC and Servo-Amplifier Combined Gain

The gains in the digital to analog converter and the servoamplifier are linked by the fact that maximum voltage out of the DAC must correspond to maximum velocity out of the drive
and, for normal electro-hydraulic drive, maximum current supplied to the servovalve.

As shown in Chapter 10, our electro-hydraulic drive is not a normal system as there is a mismatch between power pack and drive elements. The mismatch is evident at speeds in excess of $13 \mathrm{~m} / \mathrm{min}$, i.e. at $50 \%$ of rated current.

With this in mind we decided to set the servo-amplifier gain to a level which would give around 8 mA for maximum output from the DAC.

With this set-up and using open-loop, the bias in the servoamplifier was used to cancel any movement in the drive, with the DAC output set to zero. Under these conditions a test to verify the gain associated to the DAC and servo-amplifier and linearity was applied. Using the microcomputer, the input to the DAC was varied and the current out to the servovalve measured using a digital multimeter. As the bias current to stop the asymmetric cylinder can be an important element for simulation purposes, the results are shown in two graphs. Figure 11.2 .1 shows the complete range and Figure 11.2 .2 the expanded null region. The results show good linearity and a bias current of around 0.14 mA .

The combined DAC and servo-amplifier gain is given by

$$
K_{\text {DAC }} \times K_{\text {Amp }} \cong 0.0044 \frac{\mathrm{~mA}}{\text { unit }}
$$

(maximum DAC output corresponds to (+/-) 2048 units).

### 11.2.2 Servovalve and Hydraulic Cylinder Combined Gain

The combined servovalve and cylinder gain, $K_{v c y}$, was verified under closed-loop conditions. Using a simple proportional loop, the ramp input program was asked to store velocity information and the current to the electro-hydraulic servovalve was measured by a digital multimeter.

For both directions of movement, the results are shown in Figure 11.2.3, bias current taken out. As predicted before, the gain for the extending direction is higher when we move outside the null region.


(

For conditions outside the null region, the gains are
cylinder extending $K_{v c y}=48 \frac{\mathrm{~mm} / \mathrm{sec}}{\mathrm{mA}} \quad 11.3$
cylinder retracting $K_{v c y}=31 \frac{\mathrm{~mm} / \mathrm{sec}}{\mathrm{mA}}$

### 11.2.3 Gain Correction by Software

The gains for both directions of movement, being different, cause the following errors to be different. To compensate for this, the software correction shown in Figure 11.2 .4 is used. It is added to the modification shown in Figure 10.3.13.

It is clear that such correction does not include the variation in gain around null but this could be included without much difficulty. For our application, this correction was thought to be enough.

The correction gain used in this work is

$$
\frac{\mathrm{KCORR1}}{\mathrm{KCORR} 2}=1.5
$$

### 11.2.4 Total Open-Loop Gain

After the correction gain, given by equation 11.5, has been introduced, we can go back to the open-loop gain of equation 11.1 and verify its value.

Using results from 11.2 and 11.3 , as the retracting gain is corrected to this value, we have

$$
K=0.0044 \times 48=0.21 \frac{\mathrm{~mm} / \mathrm{sec}}{\text { unit }}
$$

In our case, as mentioned before, the position resolution is $1 \mu \mathrm{~m}$ and we decided to maintain it through the following error. This gives an internal position - DAC gain of 1 unit/ $\mu \mathrm{m}$. Using this value on 11.6, we have the open-loop gain, in the proper units, as

$$
\mathrm{K}=210 \mathrm{sec}^{-1}
$$



Figure 11.2.4 Gain Modification for Retracting Direction

### 11.3 COEFFICIENTS FOR THE CONTROLLER EQUATIONS

Before any experiment can be carried out, all the coefficient values, used in the control and observer equations, must be loaded in their respective memory locations. As these equations are solved by module 2 the memory locations refer to its memory map. Loading is done using TIBUG and hexadecimal.representation. It is a manual operation but simple and can be done quite quickly. Table ll. 1 is used in this operation and the values shown there refer to the extending cylinder control and observer equations. As shown in this table, provision is made for coefficients required if the delay in the control action is greater than one sampling period. The equations developed in Chapter 10 admit a delay of one complete sampling period in the control action. That is why the coefficient values for control actions older than one sampling period are set to zero in Table 11.1.

### 11.4 EXPERIMENTAL TESTS AND RESULTS

### 11.4.1 Experimenta1 Tests

The experimental tests, which results are shown in this chapter, were basically carried out to compare the behaviour of the electro-hydraulic cylinder drive under the two controller actions described in Chapter 10. The first control action is related to a digital proportional position loop and the second is related to the same proportional position loop but now with an internal state-variable feedback action, in order to. improve load damping, as shown in Figure 10.3.10. A typical experimental session is shown in Figure 11.4.1.

### 11.4.2 Position Ramp Input

For the ramp response test (step in velocity) the drive was asked to move a certain distance with a constant velocity. With the memory available for data acquisition, the actual position was stored by the controller and the velocity, pressures and thrust signals, stored in an HP storage oscilloscope.

The velocities chosen for the tests are given in Table 11.2 with respective percentages of rated current supplied to the servovalve, for the retracting direction, as given by Figure 11.2.3.


FIGURE 11.4.1 EXPERIMENTAL SESSION

|  | Coefficients for Control and Observer Equations |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | Memory <br> Location | Coefficient | Decimal Value | Hexadecimal Value |
|  | FE80 | KERRK | 0.125 | 1000 |
|  | FE82 | KUK1 | - 0.410 | CB86 |
|  | FE84 | KUK2 | 0.000 | 0000 |
|  | FE86 | KUK3 | 0.000 | 0000 |
|  | FE88 | KVOK | $+0.400$ | 3333 |
|  | FE8A | KAOK | - 0.036 | FB65 |
|  | FE8C | KDAOK | -0.120 | FOA4 |
|  | FE8E | KPMAXA | 1 | 0001 |
|  | FE90 | кРМАХВ | 1 | 0001 |
|  | FEA6 | K1W1K | - 0.0719 | F6CC |
|  | FEA8 | K1W2K | + 0.1025 | ODIE |
|  | FEAA | K1Vk | - 1.0000 | 8001 |
|  | FEAC | K1UK1 | + 0.2744 | 2381 |
|  | FEAE | K1UK2 | 0.0000 | 0000 |
|  | FEBO | K1UK3 | 0.0000 | 0000 |
|  | FEB2 | K2W1K | - 0.0628 | F7F7 |
|  | FEB4 | K2W2K | + 0.0895 | OB74 |
|  | FEB6 | K2vk | + 0.0581 | 076F |
|  | FEB8 | K2UK1 | + 0.7937 | 6597 |
|  | FEBA | K2UK2 | 0.0000 | 0000 |
|  | FEBC | K2UK3 | 0.0000 | 0000 |
|  | FECE | K0B1A | 12454 | 3046 |
|  | FEDO | K0B1B | 10000 | 2710 |
|  | FED2 | K0B2A | 1 | 0001 |
|  | FED4 | Kов2B | 1 | 0001 |
| $<x^{\circ}$ | FEEC | K3vok | 0.7107 | 5AF8 |
|  | FEEE | K4VOK | -0.9845 | 81FC |
|  | FE72 | KCORR2 | 10 | 000A |
|  | FE74 | KCORR1 | 15 | 0005 |
| . | FE6C | KUCOR1 | 1 | 0001 |
|  | FE6E | KUCOR2 | 1 | 0001 |

Table 11.1

| Velocity <br> (min/min) | \% rated <br> current |
| :---: | :---: |
| 500 | $1.7 \%$ |
| 1500 | $4.3 \%$ |
| 3000 | $9.3 \%$ |
| 6000 | $20 \%$ |
| 12000 | $41 \%$ |

Table 11.2

The conventions used in the presentation of the results are given in Figure 11.4.2.


Thrust $=\left[P_{1}-\frac{P_{2}}{2}\right] \cdot A_{1}$
$P_{1}, P_{2}=$ pressures
$A_{1}, A_{2}=$ piston areas

Figure 11.4.2 Convention for Results

A11 tests were carried out starting from middle-stroke position. The results for velocity, pressure and thrust response are presented separately for each control type.

The position loop gain was set to $K E R R K=0.125$ as it gives a velocity gain of around $30 \mathrm{sec}^{-1}$ and the simple position loop shows no overshoot.

The sampling frequency, according to our design in Chapter 10 , is set to

$$
\mathbf{f}=|\mathbf{w}|=176 \mathrm{~Hz}
$$

where $w$ is the load natural frequency at middle-stroke (rad/s). This is simply done by using the change the sampling frequency option in the interactive software.

### 11.4.2.1 Digital Proportional Controller (P)

The results for the simple digital proportional position loop control are shown in Figures 11.4 .3 and 11.4.4. The results are shown for the variables which presented meaningful displays on the oscilloscope screen. The thrust traces show considerable variation due to friction and the asymmetric characteristic, after the drive has stopped. Even so, the thrust displays that are included can give reasonable information for a simulation study.

### 11.4.2.2 State-Variable Controller (SV)

Before any result is shown for the controller with state-variable feedback something must be said about the dual model system we used for designing purposes.

The effect of the two different sets of feedback gains, as chosen in Chapter 10, was checked by moving the system in one direction with the proper observer, but using the set of gains for the opposite direction. As expected, the difference in gains from one set to the other is so small that quantization effects and drive non-linearities make its effect imperceptible.

The different set of observer equations was checked by moving in one direction with the proper set of feedback gains but using the observer equations for the opposite direction. Here again, there was no major change in the drive response for now we have an even bigger quantization effect as more equations are solved.

With the results mentioned above we decided to use only the feedback gains and observer equations for the extending cylinder, as they are designed for a less"damped"model.

The experimental results for the controller with state-variable feedback are shown in Figures 11.4.5, .6, .7, .8, .9, for the different input conditions as specified in Table 11.2. Again, only meaningful results are shown. We draw attention to the fact that the time scale on the pressure and thrust displays are different from the velocity ones as we tried to show the characteristic behaviour of those variables after the drive had stopped.


Figure 11.4.3 Digital Proportional Position Loop


Figure 11.4.4 Digital Proportional Position Loop
Responses to Ramp Input (Velocity Pulse)


Figure 11.4.5 State-Variable Controller
Velocity
$500 \mathrm{~mm} / \mathrm{min}$

Figure 11.4.5 (continuation)

(-)


(+)


Velocity
$1500 \mathrm{~mm} / \mathrm{min}$

100 ms

Thrust

160 N


Figure 11.4.6 State-Variable Controller
Ramp Input Responses(Velocity Pulse)


(-)


(+)


Thrust

160 N


Figure 11.4.7 State-Variable Controller
Ramp Input Responses(Velocity Pulse)



(-)



Velocity $6000 \mathrm{~mm} / \mathrm{min}$
(+)


Figure 11.4.8 State-Variable Controller
Ramp Input Responses(Velocity Pulse)


(-)



Velocity
$12000 \mathrm{~mm} / \mathrm{min}$
(+)


Figure 11.4.9 State-Variable Controller
Ramp Input Responses(Velocity Pulse)


### 11.4.2.3 Position Response

The ramp position response was obtained by asking the controller to store the actual position during the movement. As only 112 memory locations are available, the distance to move had to be restricted to 32 mm . For high speeds, this distance was not enough to get out of transient conditions and we, therefore, went to store the following error, when necessary.

A typical figure shows the response, under certain conditions, split into two sections. The first shows the transient during acceleration and the second during deceleration. For easier representation the initial time on the deceleration transient is set to zero. Details of representation are given in Figure 11.4.10.

The dialogue between operator and controller is shown in Figure 11.4.11 with the printout for a particular test condition.

The data was collected from printouts like the one shown in Figure 11.4 .11 and manually entered into files on the Prime computer. The final position response was obtained by submitting those files to the GRAPH.DIALOG program provided by the Computer Centre.

The most representative results are shown in Figures 11.4 .12 to 11.4.16. The first four figures show movements out of the middlestroke position in the retracting direction, back to middle-stroke position, then out in the extending direction and back to middlestroke. As mentioned before, the middle-stroke position is the reference for our absolute coordinates. Figure 11.4 .16 shows the following error for a ramp input with a velocity of $12000 \mathrm{~mm} / \mathrm{min}$, for both directions of movement, from middle-stroke position.

### 11.4.3 Position Step Input

The step response test was carried out using a facility provided by the ramp generation program. This gives a step in position conmand when the distance to be moved is smaller than the increment in position per sampling period. Asking for the maximum speed allowed by the controller ( $30 \mathrm{~m} / \mathrm{min}$ ), the maximum step input that can be generated, for a sampling frequency of around 200 Hz , is 2.5 mm . With this facility, we chose to test the position loop under step input heights of 1 and 2 mm .

Position




Figure 11.4.10 Basis of Tests to show Response to Velocity Pulse

| oftions：1）change sampling freduency <br> 2）RAMF INFUT <br> 3）OISPLAY STURED YRLUES <br> 4）STOF RND REINT MODE |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 22 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| UARIAELE：1）AGTUAL FOSITIUN（LEAST HORO） <br> 2）FOLLUMIHG ERROR（MICRONS： <br>  <br> 4 SFEED <br> 5）ACEELERATIUN <br> G）TEmISIENT bccelemation <br> $\because 2$ |  |  |  |  |  |  |  |
| MOUEくYEs（1）NOT2）？ 1 |  |  |  |  |  |  |  |
| OFTIUNS：BCHAREE SRMFLING FEEQUENGY <br> 2）REIIF IMPUT <br> ЭDISFLHY STARED VRLUES <br> 4）ETOF GNO EEINT MUD2 |  |  |  |  |  |  |  |
| 33 |  |  |  |  |  |  |  |
| －60011 | －66011 | 01681 | 62173 | 9254 | 64184 | 6485\％ | 65cte |
| 05595 | 6575 | 65973 | 66151 | 60207 | 86549 | Deese | 06749 |
| 66812 | 6658 | Q6896 | 66.92 | 66956 | 66979 |  | 67624 |
| Q6043 | Q66E1 | 47674 | 91683 | 67769 | 617191 | Q1760 | 47116 |
| Q7123 | Qrice | 0，72E | Qifer | 077124 | 67123 | 01201 | 可 312 C |
| 07120 | Q 721 | 07121 | 6？121 | 517121 | Q1721 | －172c | 47124 |
| 07125 | 07126 | 07126 | 67125 | 67124 | 0.7122 | arlet | 0.7119 |
| 07116 | 07114 | Q1720 | 07111 | 07116 | 67116 | 67110 | 6iP111 |
| Q113 | Q116 | Q7117 | 67129 | 07121 | 97122 | 4.7123 | 07122 |
| 07120 | 07118 | 6， 3115 | 时113 | 6， 9113 | 67112 | 07111 | WT110 |
| 67168 | 07166 | 0,7164 | 671日3 | 67142 | 6142 | 9716？ | 631073 |
| 0716 | Q193 | 0.7103 | 07164 | 07165 | 67105 | 67105 | 07105 |
| 07105 | 07105 | 03164 | Q1703 | 67102 | 6，7101 | 6， 1010 | 07104 |
| Q 7109 | 07101 | 07102 | 日アta？ | 07104 | 6106 | ail6？ | 616958 |

Figure 11．4．11 Operator－Controller Interactive Program

## Cylinder retracting: $1500 \mathrm{~mm} / \mathrm{min}$




Figure 11.4.12 Position Ramp Response

Cylinder extending: $1500 \mathrm{~mm} / \mathrm{min}$


cylinder retracting: $3000 \mathrm{~mm} / \mathrm{min}$



Figure 11.4.14 Position Ramp Response



Figure 11.4.15 Position Ramp Response

## cylinder retracting: $12000 \mathrm{~mm} / \mathrm{min}$


cylinder extending: $12000 \mathrm{~mm} / \mathrm{min}$


Figure 11.4.16 Position Following Error (Ramp Input)

The data shown here was obtained from the controller output listings and the graphs were obtained using the GRAPH.DIALOG program, as mentioned before.

For the step input of 1 mm , the drive was asked to move from middle-stroke to 1 mm in the retracting direction and then back to middle-stroke position. The results are shown in Figure 11.4.17. From middle-stroke position, it was then asked to move to 1 mm in the extending direction and back to middle-stroke position again. The results from this test are shown in Figure 11.4.18.

For the step input of 2 mm , the drive was asked to move from middle-stroke position to 2 mm in the retracting direction and then back to middle-stroke position (extending direction). The results from this test are shown in Figure 11.4.19.

### 11.5 DISCUSSIONS

The introduction of the software gain correction for the retracting direction, as given by equation 11.5 , has achieved its objectives as seen in Figures 11.4 .12 and 11.4.13. There, the following errors are practically the same for both directions of movement. The correction introduces a higher gain for the retracing direction, around the null region, as is clearly indicated in Figures 11.4 .3 and 11.4.5 for a velocity of $500 \mathrm{~mm} / \mathrm{min}$.

The improvement in the velocity response introduced by the statevariable controller is quite clear from the presented results. Around null, the overshoot in velocity under the state-variable controller can be considered acceptable as this region shows the lowest damping. Under accelerating or decelerating conditions, the velocity behaviour is much improved by the state-variable controller. The deterioration in velocity response for high velocities, such as $12000 \mathrm{~mm} / \mathrm{min}$, is due to the mismatch between power pack and drive.

The behaviour of the electro-hydraulic cylinder drive under the simple proportional. controller shows a good match to the simulation presented in Chapter 10, Figure 10.3.11. The frequency of oscillation, on the velocity response, is still dominated by the poles introduced by the load. The behaviour under the state-variable controller also shows a good match with the simulation results in Figure 10.3.12. For this



Figure 11.4.17 Position Step Response
( 1 mm )



Figure 11.4.18 Position Step Response
( 1 mm )



Figure 11.4.19 Position Step Response
controller action, the poles introduced by the load are much faster and damped. The drive behaviour is now dominated by the pole introduced by the servovalve.

The position step response results show that, under state-variable feedback, there is an almost critically damped behaviour. The response under the simple proportional controller cannot be acceptable in some applications.

The more complex computations required by the state-variable controller introduces an effect which can cause position error sometimes bigger than the one introduced by the simpler controller. This could be easily compensated by introducing a higher position loop gain for the state-variable controller, as it would still give a nearly critically damped response. The gains were maintained the same for both controllers just to have a fair basis for comparison.

Tests for positioning accuracy showed that, under the state-variable controller, the position error is within $\pm 0.08 \mathrm{~mm}$. Without changing the controller action a reasonable improvement can be made, in our case, by decreasing the servovalve cylinder combined gain. Looking back to equation 11.1, we see that, for the same open-loop gain, $K$, decreasing $\mathrm{K}_{\mathrm{vcy}}$ would allow a higher gain in the servo-amplifier and therefore an improvement in resolution against friction and servovalve non-linearities. A maximum speed of around $12 \mathrm{~m} / \mathrm{min}$, as in our case, and a rated current of 15 mA , would call for a servovalve cylinder combined gain of around $13 \frac{\mathrm{~mm} / \mathrm{sec}}{\mathrm{mA}}$ which is almost 3 times less than the gain we have. Maintaining the same open-loop gain, by increasing the servo-amplifier gain, a reasonable improvement in position accuracy could be achieved.

## CONCLUSIONS AND RECOMMENDATIONS

The work reported in this thesis leads to a number of significant conclusions. The choice of an electro-hydraulic cylinder drive, with an asymmetric cylinder and a 4-way servovalvè, imposed an extra degree of difficulty on the control task but introduced factors which led to a heightened awareness of the problems associated with digital control. The work reported in this thesis, however, allows a number of conclusions to be drawn on the wider implications of the research.

## CONCLUSIONS

1. The digital control strategy, discussed and designed in Chapter 10 , produced a controlled system of good performance with relaxation of primary parameters, i.e. stroke, area and mass. The same approach would also yield higher performance if an equal area cylinder were used.
2. The use of an asymmetric cylinder has been shown to be accommodated by digital controller, with arguably greater flexibility than the used controllers based on classical concepts, which require the use of different gains for different directions of movement as a minimum complication.
3. The variations in natural frequency and damping factor which were observed did not form an obstacle to the implementation of the control strategy.
4. The influence of non-linearities on the selection of observers is evident. The use of models for different operating conditions has been shown to be effective, but more work is necessary in this area.
5. The use of state-variable digital control with the controller obtained by optimal control methods has been shown to be viable for this application and allowed a simple closed form design procedure.
6. Quantization in the controller software can introduce steadystate errors on the controlled variable, through the estimation equations. Care must be taken when scaling and implementing the controller equations.
7. The choice of a single transducer, i.e. a displacement sensor, as the source of all the state signals has been shown to be effective. However, this should not be assumed to constitute a general solution. In certain high performance designs the selection of transducers will need careful consideration.
8. The actuator has a natural frequency which varies from 27 to 30 Hz . This is a relatively low range of values if a high performance specification is sought. The results presented in Chapter 11 are considered to show digital control to be a satisfactory alternative to classical analogue techniques.
9. The results presented in Chapter 11 show a system performance which is practically the same as that presented by Schmutz (82) using state-variable analogue techniques, for a system with an equal area jack. The results lead to the conclusion that the flexibility and ease of implementation of the digital approach has more to offer than the analogue method.
10. The results obtained in this work indicate that digital control can add to the performance of electro-hydraulic drives. However, there is a clear case for further research on its application.
11. The common bus direct memory access structure used in this work gives an inexpensive, reliable and fast communication link between microcomputers. It allows the implementation of parallel processing, for complex algorithms, very easily.
12. Standardised buses for multi-microcomputer structures are not yet attractive to general applications as interfaces are not readily available for most of the microcomputers.
13. The modular approach to the multi-microcomputer design gives good expandability and the use of modules with the same hardware configuration allows easy maintenance and cheaper replacement.
14. The communication through the common bus can be made transparent to the user. When using just a few modules, communications can be faster for a master-slave configuration by the reduction in the necessary handshake time.
15. Memory mapped techniques, for multi-microcomputer interfacing to control tasks, reduce processing time and simplify programming.
16. The use of assembly level language enables writing efficient programs for fast execution but must only be used for the real controlling tasks.
17. Division of the total control task in subtasks, which need few transfers of information betweenthem, allows efficient allocation of tasks to the available modules.

## RECOMMENDATIONS

1. Implementation, in high level language, of the design methods developed in this work, to run on the multi-microcomputer controller itself.
2. Design of an identification algorithm to be executed by the controller, on-line.
3. Study on the influence of using different sampling frequencies for different valve openings.
4. Design of a high gain loop by introducing feed-forward techniques.
5. Study on the use of adaptive control such as model reference and self-tuning.
6. Lopes, Luiz Eduardo
"A Study on the Application of Microprocessors to Real-Time Control", MSc Thesis, Loughborough University of Technology, January 1980.
7. Franklin, G.F. and Powell, J.D.
"Digital Control of Dinamic Systems", Addison-Wesley Company, USA, 1980.
8. Katz, Paul
"Digital Control Using Microprocessors", Prentice/Hall Int., USA, 1981.
9. Isermann, Rolf
"Digital Control Systems", Springer-Verlag, Germany, 1981.
10. Fisher, D.G. and Seborg, D.E.,
"Multivariable Computer Control - A Case Study", NorthHolland Publishing Company, Netherlands, 1976.
11. Takahashi, Y. et al
"Simple Finite-Time Settling Control and Manipulated-Variable Softening for Reverse Reaction, Overshoot and Oscillatory Processes", Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control, 100, March 1978, 50-58.
12. Takahashi, Y. et al
"Direct Digital Process Control: Practice and Algorithms for Microprocessor Application", Proceedings of the IEEE, 66, 2, February 1978, 199-208.
13. Takahashi, Y. et al
"Simple Discrete Control of Industrial Processes (Finite Time Settling Control) Algorithm for Single-Loop Digital Controller", ASME, Journal of Dynamic Systems, Measurement and Control, December 1975, 354-361.
14. Dahlin, E.B.
"Designing and Tuning Digital Controllers, Part I", Instruments and Control Systems, 41, June 1968, 77-83.
15. Dahlin, E.B. et al
"Designing and Tuning Digital Controllers, Part II", Instruments and Control Systems, 41, July 1968, 87-91.
16. Lopez, A.M. et al
"Optimal Tuning of Proportional Digital Controllers", Instruments and Control Systems, 41, October 1968, 97-102.
17. Floersch, R.
"A Digital Controller for Cyclic Temperature Control", Control Engineering, October 1978, 58-61.
18. McGreavy, C. and Awda, F.
"A Robust General Purpose Algorithm for Digital Process Control', IEE Int. Conference on Control and Its Applications, University of Warwick, UK, 23-25 March 1981, 128-133.
19. Smith, Otto J.M.
"A Controller to Overcome Dead Time", ISA Journa1, 6, 2, February 1959, 28-33.
20. Borer, J.R.
"A Multi-Variable Master Controller", Measurement and Control, 13, Apri1 1980, 126-132.
21. Clarke, D.W. et al
"Feasibility Study of the Application of Microprocessors to Self-Tuning Controllers", University of Oxford, Department of Engineering Science, Report 1137/75, 1975.
22. Stevenson, J.W.
"Servo Compensation with Digital Filters", Control Engineering, November 1971, 71-75.
23. McDonnel, D.
"Microcomputers for Digital Servo Systems", Microprocessing, 8, 20, October 1975, 74 and 78.
24. Lim, A.K. and Koepse1, W.W.
"A Microprocessor Speed Control System", IEEE Transactions on Ind. Elect. and Control Inst., 24, 3, August 1977, 241-247.
25. Hing, S.A.
"Etude Théorique de la Stabilité d'une Machine à Courant Continu Soumise à un Reglage Digital", Bull Assoc. Suisse Electr., 68, 22, November 1977.
26. Dowerty, D.W. and Wells, E.J.
"Digital Power Drive Controller Dynamics are Characterised by ROMS", Control Engineering, October 1978, 58-61.
27. Maloney, T.J. and Alvarado, F.L.
"A Digital Method for DC Motor Speed Control", IEEE Transactions on Indust. Elect. and Control Inst., 23, 1, February 1976, 44-46.
28. Burger, P. and Ronchinsky, S.
"A Microprocessor Driven Digital Servo System", Proceedings Indust. Elect. and Control Inst., March 21-23, 1977, 159163.
29. Matsumoto, Y.
"Evaluation of the Digital Prediction Filter Applied to Control a Class of Servomotor by Microcomputers", IEEE Transactions on Indust. Elect. and Control Inst., 23, 4, 1976, 359-363.
30. Ernsberger, G.W.
"A Microprocessor Controlled Positioning System Utilizing Thyristors in a Reversible DC Drive.!, Proceedings Indust. Elect. and Control Inst., March 21-23, 1977, 152-158.
31. Anon
"Microprocessor-Controlled AC Servo Drive System for Machine Feed Slides", Engineer's Digest, 41, 3, March 1980, 15-19.
32. Smith, L.C.
"Servo-Control through Software", Machine Design, May 25, 1978, 70-75.
33. Plant, J.B. et al
'Microprocessor Control of Position or Speed of an SCR DC Motor Drive", IEEE Transactions on Ind. Elect. and Control Inst., 27, 3, August 1980, 228-234.
34. Klein, C.A. and Maney, J.J.
"Real-Time Control of a Multiple-Element Mechanical Linkage with a Microcomputer", IEEE Transactions on Ind. Elect. and Control Inst., 26, 4, November 1979, 227-234.
35. Hartman, J.L. et al
"Adaptive Position Controller Avoids Complex Algorithms", Control Engineering, May 1981, 71-74.
36. Aylor, J.H. et al
"Design and Application of a Microprocessor PID Predictor Controller", IEEE Transactions on Ind. Elect. and Control Inst., 27, 3, August 1980, 133-137.

Schnieder, E.
"Control of DC Drives by Microprocessors", IFAC Symposium on Control in Power Elect. and Elect. Drives, Dusseldorf, October 1977, 603-608.
33. Oumanar, A. et al
"Design of an Optimal, Autoadaptive Current Loop for DC Motor. Realization with an Hybrid Device Including a Microprocessor", IFAC Symposium on Control in Power Elect. and Elect. Drives, Dusseldorf, October 1977, 593-601.
34. Tsuchiya, Takeshi
"Basic Considerations on Applications of Optimal Control and the Microprocessor to the Induction Motor Speed Control System", Int. J. Control, 31, 2, 1980, 285-302.
35. Bollinger, J.G. et al
"Digital Control and Feed Drives State-of-the-Art and New Developments", Annals of the CIRP, Vol. 29/2/1980, 497-506.
36. Brusse1, H. Van et al
"Microprocessors in Hierarchical Control Systems", Annals of the CIRP, Vol. 27/1/1978, 265-269.
37. Brussel, H. Van and Vastmans, L.
"Direct Digital Control of Feed Drives", 13th CIRP International Seminar "Microprocessors.in Manufacturing Systems", Leaven, 23-24 June 1981, 103-117.
38. Duffie, N.A. and Bollinger J.G.
"Direct Microprocessor Control of SCR Feed Drives", 13th -CIRP International Seminar "Microprocessors in Manufacturing Systems", Leaven, 23-24 June 1981, 161-175.
39. State, G. and Hesselbach, J.
"Discrete-Time Position Control at NC Machines Parameter and System Optimization", Annals of the CIRP, Vol. 28/1/1979, 257-261.
40. State, G. et al
"Digitate Lageregelung an Numerisch Gesteuerten Maschinen fur Hochgeschwindigkeitsbearbeitungen', 13th CIRP International Seminar "Microprocessors in Manufacturing Systems", Leaven, 23-24 June 1981, 79-90.
41. Rio, G.V.
"Complex Digital Control Systems", Van Nostrand Reinhold Company, New York, 1979.
42. Schmitt, G.H. and Chang, G.C.C.
"Avionics Microprocessor is Heart of Distributed Microcomputer System for Machine Tool Control", Proc. IEEE, Naecon, 1977, 701-704.
43. Pop, F.
"The Design and Implementation of a Multicomputer Numerical Control System ", PhD Thesis, UMIST, April 1979.
44. Prąsad, A.
"The Design and Development of a Microprocessor Based Controllex of Stepping Motor Drives", PhD Thesis, UMIST, March 1979.
45. Stupe, G. and Worn, H.
\% "Principle and Examples of a Modular Multiprocessor NC System", Proc. 19th MTDR Conference, September 1978, 13-17.
$\stackrel{t}{k}$
46. Stuťe, G. et al
"A Multiprocessor Control System as a Universal Modular System for the Design of Machine Tool Controllers", Procs. and IFAC/ IFIP Symposium, Stuttgart, 22-24 October 1979, 37-46.
47. Stute, G. and Klemm, P.
"The Application of a Modular Multi-processor NC System", Proc. 22nd MTDR Conference, UMIST, 1981, 215-222.
48. Ilic, V.B. et al
"Interprocessor Communication in the Multi-microprocessor CNC Systems", 13th CIRP Int. Seminar on Manufact. Systems, Leuven, 23-24 June, 1981, 193-207.
49. Dalze11, D. et al
"A Microprocessor Based Hierarchical Control System for Machine Tools", 13th CIRP Int. Seminar on Manufact. Systems, Leuven, 23-24 June, 1981, 177-191.
50. Luh, J.S. and Lin, C.S.
"Multiprocessor-Controllers for Mechanical Manipulators", 3rd Int. Conf. Computer Software and Applications, Chicago, November 1979, 458-463.
51. Kuisma, A. et al
"Multimicroprocessor Control System for Special Purpose Machine Tools", Proc. IFAC Int. Symposium, Tokyo, October 1977, 189-193.
52. Rembold, U.
"Recent Advances in the Use of Computers in Quality Control", Proc. 2nd IFAC/IFIP Symposium, Stuttgart, October 1979, 265-277.
53. Maples, G.C. and Jervis, P.
"The Structure of Microprocessor Based Control Systems", Central Elect. Generating Board, RD/LM/CONT/O13, 1977.
54. Jervis, P.
"A Study of Microprocessor Based Distributed Control Networks", Central Elect. Generating Board, RD/L/N63/77, 1977.
55. Kuo, B.C.
"Automatic Control Systems", Prentice/Hall Inc., 1975.
56. Tou, J.T.
"Digital \& Sampled Data Control Systems", McGraw-Hill, 1959.
57. Gilbert, E.G.
"Controllability and Observability in Multivariable Control Systems", J.S.I.A.M. Control, A, 1, 2, 1963, 128-150.
58. Ackermann, J.
"Der Entwurf Linearer Regelungssysteme im Zustandsraum", Regelungstechnik und Prozessdatenverarbeitung, 7, 1972, 297-300.
59. Bryson, E.A. and Ho, Y.
"Applied Optimal Control" Blaisdell Publishing Company, USA, 1969.
60. Citron, S.J.
"Elements of Optimal Control", Holt-Rinehart and Winston, Inc., USA, 1969.
61. Luenberger, D.G.
"Observing the State of a Linear System", IEEE Trans. on Military Electronics, 8, April 1964, 74-80.
(62). Luenberger, D.G.
"Observers for Multivariable Systems", IEEE Trans. on Automatic Control, 11, 2, April 1966, 190-197. 7
63. Luenberger, D.G.
"An Introduction to Observers", IEEE Trans. On Automatic Control, 16, 6, December 1971, 596-602. - 6
(64) Gopinath, B.
"On the Control of Linear Multiple Input-Output Systems", The Bell Systems Technical Journal, 50, 3, March 1971, 10631081.
65. Wellstead, P.E. and Zanker, P.
"Servo Self-Tuners", Int. J. Control, 30, 1, 1979, 27-36.
66. Moler, C. and Loan, C.Van
"Nineteen Dubious Ways to Compute the Exponential of a Matrix',' SIAM Review, 20, 4, October 1978, 801-836.
67. Ward, R.C.
"Numerical Computation of the Matrix Exponential with Accuracy Estimate", SIAM J. Numer. Anal., 14, 4, September 1977, 601-610.
68. Kallstrom, C.
"Computing $\exp (\mathrm{A})$ and $\exp (A s) \mathrm{ds} "$, Report 7309, Division of Automatic Control, Lund Inst. of Techn, Lund, Sweden, 1973.
69. Fuller, S.H. et al
"Multi-Microprocessors: An Overview and Working Example", Proceedings of the IEEE, 66, 2, 1978, 216-228.
70. Weitzman, C .
"Distributed Micro/Minicomputer Systems - Structure, Implementation and Application", Prentice/Hall, USA, 1980.
71. Anderson, G.A. and Jensen, E.D.
"Computer Interconnection Structures: Taxonomy, Characteristics and Examples", Computing Surveys, 7, 4, 1975, 197-213.
72. Borril1, P.L.
"Microprocessor Bus Structures and Standards", Microprocessor Systems, North-Holland Publishing Company, 1980, 285-296.
73. Shere, J.W.
"An Introduction to the IEEE-488 Bus", Electronic Technology, 15, February 1981, 30-31.
74. Fisher, E. and Jensen, C.W.
"PET and the IEEE-488 Bus (GPIB)", McGraw-Hill, USA, 1980.
75. Elmquist, K.A. et al
"Standard Specification for.S-100 Bus Interface Devices", IEEE, Computer Magazine, July 1979, 28-52.
76. Anon
"TMS 9900 Microprocessor Data Manual", Texas Instruments, 1976.
77. Anon
"TM 990/100M Microcomputer User's Guide", Texas Instruments, 1978.
78. Anon
"Mode1 990 Computer TMS 9900 Microprocessor - Assembly Language Progranmer's Guide", Texas Instruments, March 1978.
79. Anon
"RTI-1240/1241 User's Reference Manual", Analog Devices Inc. 1980.
80. Anon
"The Interface Circuits Data Book", Texas Instruments, 1978.
81. Anon
"Heidenhain 7010 Digital Readout", Heidenhain (G.B.) Ltd.
82. Schmutz, R.H.
"State-Variable Feedback Control of Hydraulic Feed Drives", Annals of the CIRP, 29, 1, 1980.
83. Ertan, U.
"Contributions to the Modelling and Design of High Performance Electro-hydraulic Cylinder Drives used in Position Control", PhD Thesis, UMIST, April 1976.
84. Bell, R. and de Pennington. A.
"Active Compensation of Lightly Damped Electrohydraulic Cylinder Drives using Derivative Signals", Proc. Instn. Mechanical Engineers, 184, 4, 1969-70, 83-94.
85. Bell, R. and de Pennington, A.
"The Design of Active Damping for Electrohydraulic Cylinder Feed Drives", Proc. 9th MTDR Conf. Birmingham, 1968, 13091324.
86. Viersma, T.J.
"Analysis, Synthesis and Design of Hydraulic Servosystems and Pipelines", Elsevier Scientific Publishing Company, The Netherlands, 1980.
87. Merritt, H.E.
"Hydraulic Control Systems", John Wiley \& Sons, Inc., USA, 1967.
88. Thayer, W.J.
"Specification Standards for Electro-hydraulic Flow Control Servovalves", MOOG INC., Technical Bulletin 17, 1962.
89. Middleditch, A.E.
"Design Criteria for Multi-Axis Closed Loop Computer Control Systems', Trans. of the ASME, Journal of Dynamic Systems, Measurement and Control, March 1974, 36-40.
90. Althoff, H. et al
"E-Bus System Design Handbook", Texas Instruments, September 1981.
91. Farrar, F.A.
"Microprocessor Implementation of Advanced. Control Modes", Conf. in Computer Simulation, Chicago, July 1977, 339-342.
92. Irving, E.
"Implicit Reference Model and Optimal Aim Strategy for Electrical Generator Adaptive Control", Int. Symposium on Adaptive Systems, Ruhr University, Bochum, 20-21 March, 1980.

APPENDICES

## APPENDIX 4.1

## PSLMAI PROGRAM

```
    SUBROUTINE PSIMA1(N,P,VARI,PSIl,PSI2)
    DOUBLE PRECISION P(10,10),VAR1,P1(10),PlMAX,VAR2,IDENT(10,10)
    DOUBLE PRECISION APSI(10,10), PSIl(10,10),PSI2(10,10),PSI3(10,10)
    DOUBLE PRECISION P2MAX,PT(10,10),P2(10,10)
    INTEGER N
C CALCULATE P*VARI
C
    I=\emptyset
    IF(I.EQ.0) GO TO 444
    WRITE(1,100)
10\emptyset FORMAT(1H,'ENTER ORDER N')
    READ(1,33) N
33 FORMAT(1H,Il)
    WRITE(1,102)
102 FORMAT(1H ,'ENTER MATRIX A BY LINES')
    READ(l,*)((P (I,J),J=l,N),I=1,N)
    WRITE(1,104)
104 FORMAT(1H ,'ENTER VARIABLE VAR1')
    READ(1,*) VARI
    DO 66 I=1,N
    DO 66 J=1,N
66 P2(I,J)=DABS (P(I,J))
444 DO lØ0\emptyset I=l,N
1Ø\emptyset\emptyset P1(I)=\emptyset.\emptyset
    DO 200\emptyset J=1,N
    DO 200\emptyset I=1,N
2\emptysetด\emptyset Pl(J)=Pl(J)+P2(I,J)
    DO 2777 I=1,N
2777 Pl(I)=Pl(I)*VAR1
    P2MAX=0.0
    DO 20ø2 I=1,N
    P1MAX=P1(I)-P 2MAX
    IF(PIMAX.LE.g.\emptyset) GO TO 20Ø2
    P2MAX=P1(I)
2\emptyset\emptyset2 CONTINUE
456 FORMAT(1H ,'P2MAX=',F12.5)
    IF(P2MAX.EQ.\emptyset.\emptyset) GO TO 2\emptyset\emptyset5
    P 2MAX=DLOG 2 (P 2MAX)
    IF(P2MAX.LE.\emptyset.\emptyset) GO TO 2\emptyset\emptyset5
    P2MAX=DINT (P2MAX)
    KPSI=IDINT (P2MAX) +1
43 FORMAT(1H ,I4)
45 FORMAT(1H ,F20.4)
    GO TO 999
20\emptyset5 KPSI=Ø
C CALCULATE PSI(T/2TO K)
999 VAR2=VAR1/2.0**KPSI
    DO 2006 I=1,N
    DO 2006 J=1,N
    PSI2(I,J)=\emptyset
    APSI(I,J)=Ø.\emptyset
    PSI3(I,J)=\varnothing. Ø
2Ø06 IDENT(I,J)=\emptyset.\emptyset
    DO 2007 I=1,N
```

```
2007 IDENT(I,I)=1.\emptyset
    DO 2008 I=1,N
    DO 2008 J=1,N
2008 PSIl(I,J)=IDENT(I,J)
4444 FORMAT(1H ,I2)
    L=3}
    L2=L
2009 DO 2015 I=1,N
    DO 2015 J=1,N
    PSI2(I,J)=\varnothing
2ø15 PT(I,J)=P(I,J)*VAR2/L
    DO 20lØ I=l,N
    DO 2010 J=1,N
    DO 2ø10 K=1,N
2ø10 PSI2(I,J)=PT(I,K)*PSII(K,J)+PSI2(I,J)
    DO 2011 I=1,N
    DO 2011 J=1,N
2ø11 PSIl(I,J)=PSI2(I,J)+IDENT(I,J)
    L=L-1
    IF(L.GT.1) GO TO 2ø09
C EXP PSI
    K2=KPSI
    IF(K2.EQ.ø) GO TO 90ø5
7000 DO 7050 I=l,N
    DO 7050 J=1,N
705\emptyset PT(I,J)=P(I,J)*VAR1/2.0**(K2+1)
    DO 3000 I=1,N
    DO 300ø J=1,N
    DO 3000 K=1,N
3øø\emptyset APSI(I,J)=APSI(I,J)+PT(I,K)*PSII(K,J)
    DO 4|ø\emptyset I=1,N
    DO 4000 J=1,N
4øøø APSI(I,J)=APSI (I,J)+IDENT(I,J)
    DO 500ø I=1,N
    DO 5000 J=1,N
    DO 50ø\emptyset K=1,N
5øøø PSI3(I,J)=PSI3(I,J)+APSI(I,K)*PSII(K,J)
    DO 6000 I=1,N
    DO 60øø J=1,N
    PSIl(I,J)=PSI3(I,J)
    PSI3(I,J)=Ø.ø
    APSI (I,J)=\varnothing.\emptyset
6000
    *
        K2=К2-1
    IF(K2.GT.ø) GO TO 7øøø
9000 FORMAT(1H , 3F20.4)
9005 DO 32ø I=1,N
    DO 320 J=l,N
    PSI2(I,J)=\varnothing.\emptyset
32ø PT(I,J)=P(I,J)*VARI
    DO 330 I=1,N
    DO 330 J=1,N
    DO 330 K=1,N
    PSI2(I,J)=PSI2(I,J)+PSIl(I,K)*PT(K,J)
```

$$
\text { DO } 543 \mathrm{I}=1, \mathrm{~N}
$$

$$
\text { DO } 543 \mathrm{~J}=1, \mathrm{~N}
$$

543

```
PSI2(I,J)=PSI2(I,J)+IDENT(I,J)
```

RETURN
END

## APPENDIX 4.2

## STATE PROGRAM

```
DOUBLE PRECISION A(10,1\emptyset),B(1\emptyset),T,TD,TM,T1
DOUBLE PRECISION PHIT(10,10), PSI2(10,10), PSIl(10,10)
DOUBLE PRECISION BTM(10),C(10,10), PHITM(10,10),IDENT(10,10)
DOUBLE PRECISION L2(10),Ll(10), PHIT2(10,10),L(10)
```

C
C

20

$$
\text { DO } 6 \quad I=1, N
$$

CALCULATING PHIT
CALL PSIMAI (N, A, T, PSII, PSI2)
DO $10 \mathrm{I}=1, \mathrm{~N}$
DO $1 \emptyset J=1, N$
L2 (I) = Ø. $\varnothing$
LI (I) $=0.0$
$\operatorname{PHIT}(I, J)=\operatorname{PSI} 2(I, J)$
CALCULATING L2 AND PHITM FOR LI
CALL PSIMAI (N, A, TM, PSI1, PSI2)
DO 20 I=1, $N$
WRITE (1, 71)
CALL TNOUA('ENTER NUMBER OF STATES $=1,25$ )
READ (1,*) N
WRITE (1, 3)
FORMAT (1H,/,'ENTER MATRIX A BY LINES')
WRITE (1, 71)
$\operatorname{READ}(1, *)((A(I, J), J=1, N), I=1, N)$
WRITE $(1,4)$
FORMAT (1H, /,'ENTER MATRIX B BY COLUMN')
WRITE (1, 71)
$\operatorname{READ}(1, *)(B(I), I=1, N)$
WRITE (1, 71)
CALL TNOUA ('ENTER SAMPLING PERIOD $T(M S)=1,29)$
READ (1,*) T

WRITE (1,71)
CALL TNOUA('ENTER DELAY TD (MICROSECOND) $=1,3 \emptyset)$
$\operatorname{READ}(1, *) \mathrm{TD}$
TD=TD*.$\varnothing 0 \emptyset \varnothing \emptyset 1$
Tl=DINT (TD/T) $+1 . \emptyset$
TM=T1*T-TD
IF (TM.LT.T) GO TO $30 \emptyset$
Tl=T1-1. $\quad$ D
$\mathrm{TM}=\varnothing$
TD=T
GO TO 40ø
$T D=T-T M$
DO $5 \mathrm{I}=1, \mathrm{~N}$
DO $5 \mathrm{~J}=\mathrm{I}, \mathrm{N}$

$$
\operatorname{IDENT}(I, I)=1 . \theta
$$

$\mathrm{BTM}(\mathrm{I})=\mathrm{B}(\mathrm{I}){ }^{*} \mathrm{TM}$
DO $30 I=1, N$

```
        DO \(30 \mathrm{~J}=1, \mathrm{~N}\)
        \(C(I, J)=\varnothing . \emptyset\)
        PHITM (I, J) = PSI2 (I, J)
        30
    \(\mathrm{L} 2(\mathrm{I})=\mathrm{L} 2(\mathrm{I})+\mathrm{PSI} 1(\mathrm{I}, \mathrm{J}) * \operatorname{BTM}(\mathrm{~J})\)
C
C CALCULATING L1
C
    CALL PSIMAI (N, A, TD, PSI1,PSI2)
    DO \(40 \quad I=1, N\)
    \(\mathrm{BTM}(\mathrm{I})=\mathrm{B}(\mathrm{I}) * T D\)
    DO \(50 \mathrm{I}=1, \mathrm{~N}\)
    DO \(50 \mathrm{~J}=1, \mathrm{~N}\)
    DO \(50 \mathrm{~K}=1, \mathrm{~N}\)
    \(C(I, J)=C(I, J)+\operatorname{PHITM}(I, K) * \operatorname{PSII}(K, J)\)
    DO 60 I=l,N
    DO \(6 \emptyset \mathrm{~J}=1, \mathrm{~N}\)
\(60 \quad \mathrm{LI}(\mathrm{I})=\mathrm{LI}(\mathrm{I})+\mathrm{C}(\mathrm{I}, \mathrm{J}) * \operatorname{BTM}(\mathrm{~J})\)
    \(\mathrm{N} 2=\mathrm{IDINT}(\mathrm{T} 1)\)
    \(\mathrm{N} 3=\mathrm{N}+\mathrm{N} 2\)
    DO 5 ØØ \(I=1\), N3
    DO \(5 \emptyset \emptyset \mathrm{~J}=1, \mathrm{~N} 3\)
    PHIT2 (I, J) = Ø. Ø
    DO 6ดØ \(I=1, N\)
    DO \(60 \emptyset \mathrm{~J}=1, \mathrm{~N}\)
\(60 \emptyset \quad \operatorname{PHIT}(I, J)=\operatorname{PHIT}(I, J)\)
    \(J=N+1\)
    DO 7ØØ I=1,N
    PHIT2 (I, J) =Ll(I)
    IF (N2.GT.1) GO TO 1ØØØ
    \(\mathrm{L}(\mathrm{N}+1)=1 . \emptyset\)
    DO 8øØ I=1,N
\(8 \emptyset \emptyset \quad \mathrm{~L}(\mathrm{I})=\mathrm{L} 2(\mathrm{I})\)
    GO TO 2øø口
\(1 \varnothing \emptyset \emptyset \quad \mathrm{~N} 3=\mathrm{N}+\mathrm{N} 2-1\)
    DO \(101 \varnothing \mathrm{I}=1, \mathrm{~N} 3\)
    \(1010 \quad L(I)=\varnothing\)
    \(L(N+N 2)=1 . \varnothing\)
    \(\mathrm{I}=\mathrm{N}\)
    \(\mathrm{K}=\mathrm{N}+1\)
    N3 \(=\) N \(2-1\)
    DO \(1020 \mathrm{~J}=1, \mathrm{~N} 3\)
1020 PHIT2 (I +J, K \(+J\) ) \(=1 . \emptyset\)
C
C OUTPUT
C
2ØØD WRITE (1,71)
    CALL TNOUA ('MATRIX PHI', lØ)
    WRITE (1, 71)
    DO \(70 \mathrm{I}=1, \mathrm{~N}\)
    WRITE (1, 71)
71 FORMAT (1H)
    WRITE (1, 72) (PHIT (I, J) , J=1, N)
\(7 \emptyset\) CONTINUE
72 FORMAT (1H,10ดF1Ø.4)
```

```
    WRITE(1,71)
    CALL TNOUA('MATRIX LI',9)
    WRITE(1,71)
    WRITE(1,80)(Ll(I),I=1,N)
8\emptyset FORMAT(1H ,/,F20.4)
    WRITE(1,71)
    CALL TNOUA('MATRIX L2',9)
    WRITE(1, 8\emptyset)(L2(I),I=1,N)
    WRITE(1,71)
    CALL TNOUA('MATRIX PHIT2',12)
    WRITE(1,71)
    N}=\textrm{N}+\textrm{N}
    DO }82\textrm{I}=1,
    WRITE(1,71)
    WRITE(1, 72)(PHIT2(I,J),J=1,N)
    WRITE(6, 72)(PHIT2(I,J),J=1,N)
82. CONTINUE
    WRITE (1,71)
    CALL TNOUA('MATRIX L',8)
    WRITE(1,8Ø)(L(I),I=1,N)
    WRITE (6, 8Ø) (L(I),I=1,N)
    STOP
    END
```


## ACKER PROGRAM

DOUBLE PRECISION PHIl(6,6), GAMMA(6),T,IDENT $(6,6), \operatorname{ALFA}(6,6)$
DOUBLE PRECISION A, B,Al, Bl, PHITWO $(6,6), \operatorname{AlPHI}(6,6)$
DOUBLE PRECISION A2IDEN $(6,6), \operatorname{CX}(6,6), \operatorname{CX1}(6), \operatorname{CX2}(6)$
DOUBLE PRECISION P(10), DP,ALFA1 $(6,6)$
INTEGER IA,N,IFAIL,IR,IB
C ACCEPT DESIRED POLE LOCATIONS ON S-PLANE
WRITE (1,5)
FORMAT (1H)
CALL TNOUA ('ENTER NUMBER OF STATES = ',25)
$\operatorname{READ}(1,1 \varnothing) \mathrm{N}$
FORMAT(1H , Il)
WRITE $(1,5)$
CALL TNOUA ('ENTER SAMPLING PERIOD(SEC) $=\mathbf{~ ' , 2 9 ) ~}$
$\operatorname{READ}(1, *) T$
WRITE (1, 2ø)
$2 \sigma$ FORMAT(1H ,/,'ENTER MATRIX PHII BY LINE')
$\operatorname{READ}(1, *)((\operatorname{PHI} 1(I, J), J=1, N), I=1, N)$
WRITE (1,30)
30 FORMAT(1H ,/,'ENTER MATRIX GAMMA BY COLUMN')
$\operatorname{READ}(1, *)(\operatorname{GAMMA}(\mathrm{I}), \mathrm{I}=1, \mathrm{~N})$
DO $4 \emptyset \mathrm{I}=1, \mathrm{~N}$
DO $40 \mathrm{~J}=1, \mathrm{~N}$
PHITWO (I, J) $=\varnothing . \varnothing$
Cx(I,J) $=\varnothing . \varnothing$
CX1 (I) = Ø. $\varnothing$
CX2 (I) = Ø. $\varnothing$
DO $5 \emptyset \mathrm{I}=1, \mathrm{~N}$
$\operatorname{IDENT}(I, I)=1 . \sigma$
DO $60 \mathrm{I}=1, \mathrm{~N}$
DO $60 \mathrm{~J}=1, \mathrm{~N}$
$\operatorname{ALFA}(\mathrm{I}, \mathrm{J})=\operatorname{IDENT}(\mathrm{I}, \mathrm{J})$
DO $7 \emptyset \mathrm{I}=1, \mathrm{~N}$
DO $7 \emptyset \mathrm{~J}=1, \mathrm{~N}$
DO $7 \varnothing \mathrm{~K}=1, \mathrm{~N}$
$\mathrm{Kl}=\mathrm{N}$
IF (K1.LE. Ø) GO TO 255
WRITE $(1,5)$
CALL TNOUA('ENTER DESIRED POLE S-PLANE REAL= ',33)
$\operatorname{READ}(1, *)$ A
CALL TNOUA (' IMAG= ',33)
$\operatorname{READ}(1, *) \mathrm{B}$
IF (B.EQ..$\varnothing$ ) GO TO 130
$A 1=-2 \cdot \varnothing * \operatorname{DEXP}(A * T) * \operatorname{COS}(B * T)$
$\mathrm{Bl}=\operatorname{DEXP}\left(2 . \boldsymbol{\theta}^{*} \mathrm{~A}^{*} \mathrm{~T}\right)$
DO $9 \varnothing I=1, N$
DO $9 \varnothing \mathrm{~J}=1, \mathrm{~N}$
$\operatorname{ALFAl}(\mathrm{I}, \mathrm{J})=\varnothing . \emptyset$
$\operatorname{AlPHI}(I, J)=\operatorname{PHITWO}(I, J)+A l * P H I 1(I, J)+B l * I D E N T(I, J)$
DO $1 \varnothing \varnothing \mathrm{I}=1, \mathrm{~N}$
DO $1 \varnothing \varnothing K=1, N$

```
        DO \(1 \emptyset \emptyset \mathrm{~J}=1, \mathrm{~N}\)
\(110 \quad \operatorname{ALFA}(I, J)=\operatorname{ALFAI}(I, J)\)
    \(\mathrm{Kl}=\mathrm{Kl}-2\)
    GO TO 120
\(130 \quad \mathrm{Al}=\mathrm{DEXP}(\mathrm{A} * \mathrm{~T})\)
    DO \(140 \mathrm{I}=1, \mathrm{~N}\)
    DO \(140 \mathrm{~J}=1, \mathrm{~N}\)
    ALEAl (I, J) \(=\varnothing . \emptyset\)
    AlPHI (I, J) \(=\) PHII(I,J)-AI*IDENT (I,J)
    DO \(150 \mathrm{I}=1, \mathrm{~N}\)
    DO \(150 \mathrm{~J}=1, \mathrm{~N}\)
    DO \(150 \mathrm{~K}=1, \mathrm{~N}\)
\(150 \quad \operatorname{ALFAl}(I, J)=\operatorname{ALFAl}(I, J)+A L F A(I, K) * A l P H I(K, J)\)
    DO \(160 \mathrm{I}=1, \mathrm{~N}\)
    DO \(160 \mathrm{~J}=1, \mathrm{~N}\)
    ALFA (I, J) =ALFAI (I, J)
    Kl=Kl-l
    GO TO 120
C
C CONTROLLABILITY MATRIX CX
C
255 DO \(170 \quad \mathrm{I}=1, \mathrm{~N}\)
\(17 \emptyset \quad\) CXI (I)=GAMMA (I)
    \(\mathrm{K}=1\)
\(22 \emptyset\) DO \(180 \mathrm{I}=1, \mathrm{~N}\)
\(180 \quad \mathrm{CX}(\mathrm{I}, \mathrm{K})=\mathrm{CXI}\) (I)
    \(K=K+1\)
    IF (K.GT.N) GO TO 257
    DO \(190 \mathrm{I}=1, \mathrm{~N}\)
    DO \(190 \mathrm{~J}=1, \mathrm{~N}\)
    CX2 (I) =CX2 (I) +PHI1 (I, J) * CX1 (J)
    DO \(200 \mathrm{I}=1, \mathrm{~N}\)
    CX1 (I) \(=\) CX2 (I)
\(2 \emptyset \emptyset \quad\) CX2 (I) \(=\varnothing . \emptyset\)
    GO TO \(22 \emptyset\)
C
C SCALING BEFORE FOlBTF
C
250 DO \(666 \mathrm{I}=1, \mathrm{~N}\)
    DO \(666 \mathrm{~J}=1, \mathrm{~N}\)
666 AlPHI (I,J) \(=\operatorname{DABS}(C X(I, J))\)
    \(A=A 1 P H I(1,1)\)
    DO 2øØØ \(I=1, N\)
    DO \(2000 \mathrm{~J}=1, \mathrm{~N}\)
    \(B=A 1 P H I(I, J)-A\)
    IF (B.GE. Ø. Ø) GO TO 2øøø
    \(\mathrm{A}=\mathrm{AlPHI}(\mathrm{I}, \mathrm{J})\)
2øøØ CONTINUE
    DO \(1060 \mathrm{I}=1, \mathrm{~N}\)
    DO \(1 \emptyset 6 \emptyset J=1, N\)
\(1 \varnothing 6 \emptyset \quad C X(I, J)=C X(I, J) / A\)
```


## c

C SOLVING CONTROL LAW
c
257 DO 260 I=1,N
CX2 (I) = Ø. $\varnothing$
$260 \quad \operatorname{cx1}(\mathrm{I})=\varnothing . \emptyset$
$\operatorname{CX1}(N)=1 . \emptyset$
DO $270 \mathrm{I}=1, \mathrm{~N}$
DO $270 \mathrm{~J}=1, \mathrm{~N}$
$27 \sigma \quad \operatorname{A2IDEN}(I, J)=C X(J, I)$
11 FORMAT(1H,2F30.10)
IA $=6$
$\mathrm{Nl}=\mathrm{N}$
IFAIL=1
CALL FØ1BTF(N1,A2IDEN, IA, P, DP, IFAIL)
IF (IFAIL.EQ.O) GO TO 290
WRITE(1,280) IFAIL
280 FORMAT(1H ,'ERROR FØ1BTF IFAIL=',I2)
STOP
290 IR=1
$I B=6$
$\mathrm{N} 1=\mathrm{N}$
$I A=6$
CALL Fø4AYF(N1,IR,A2IDEN,IA, P, CXI,IB,IFAIL)
DO $3 \varnothing \square$ I=1,N
DO $300 \mathrm{~J}=1, \mathrm{~N}$
$\operatorname{CX2}$ (I) $=\operatorname{CX2}(\mathrm{I})+\operatorname{CXI}(\mathrm{J}) * \operatorname{ALFA}(\mathrm{~J}, \mathrm{I})$
WRITE (1,310)
310 FORMAT(1H ././.' K')
$\operatorname{WRITE}(1,32 \theta)(\operatorname{CX2(I)}, I=1, N)$
$32 \varnothing$
FORMAT(1H ,F12.4)
STOP
END

```
    DOUBLE PRECISION PHIT(10,10),PHI(10,10),L(10,10)
    DOUBLE PRECISION Q2,Q1(10,10),SLT(10,10),LT(10,1\varnothing)
    DOUBLE PRECISION SLT2(10,10),MM(10,10),MM1(1\varnothing,10)
    DOUBLE PRECISION SLT3(10,10),SL(10,10),SLT1(10,10)
    DOUBLE PRECISION S(10,10),AAl
    INTEGER MA,KCOUNT,I,J,K
    WRITE(1,5)
    FORMAT(/)
    CALL TNOUA('ENTER ORDER N =',15)
    READ(1,*) N
    WRITE (1,5)
    WRITE(1,30)
    FORMAT('ENTER MATRIX PHI BY LINE')
    READ(5,*)((PHI (I,J),J=1,N),I=1,N)
    WRITE(1,40)
    FORMAT('ENTER MATRIX L BY COLUMN')
    READ(5,*)(L(I,l),I=l,N)
    WRITE (1,5)
    WRITE(1,50)
    FORMAT('ENTER MATRIX QL BY LINE')
    WRITE (1,5)
    READ(l,*)((Ql(I,J),J=1,N),I=l,N)
    WRITE(1,60)
    FORMAT(lH ,/,'ENTER VALUE Q2')
    WRITE(1,5)
    READ(1,*) Q2
    CALL MATCOP(N,N,Q1,S)
    WRITE(1,5)
    CALL TNOUA('ENTER NUMBER OF STEPS =',23)
    READ(1,*) KCOUNT
    WRITE (1,5)
    MA=1
    CALL MATRAN(N,MA,L,LT)
    CALL MATRAN(N,N,PHI,PHIT)
l11 CALL MATMUL(N,N,MA,S,L,SL)
    CALL MATMUL(MA,N,MA,LT,SL,SLT)
    AAl=Q2+SLT (1,1)
    AAl=1.9/AA1
    CALL MATMUL(MA,N,N,LT,S,SLTI)
    CALL MATMUL(N,MA,N,SL,SLT1,SLT2)
    CALL MATVRB(N,N,SLT2,AA1,SLT2)
    CALL MATSUB(N,N,S,SLT2,MM)
    CALL MATMUL(MA,N,N,SLT1,PHI,SLTT3)
    CALL MATVRB(MA,N,SLT3,AAI,SLT3)
    CALL MATMUL(N,N,N,MM,PHI,MMI)
    CALL MATMUL(N,N,N,PHIT,MMI,MM)
    CALL MATADD(N,N,MM,Q1,S)
    WRITE(1, 200)(SLT3(1,I),I=1,N)
20ø FORMAT(1H ,6F12.4)
KCOUNT=KCOUNT-1
IF(KCOUNT.GE.0) GO TO 111
WRITE(1,5)
CALL TNOUA('CHANGE Q1(1),CHANGE Q2(2),STOP(3) ',33)
READ(1,*) DAD
```

IF (DAD.EQ.1.ø) GO TO 45
IF (DAD.EQ.2. $\varnothing$ ) GO TO 46
STOP
END

## MATRICES ARITHMETIC

C C C C C C C C
************MATRICES OPERATIONS PACKAGE*
***SUBROUTINE FOR MATRICES MULTIPLICATION

$$
C(N, M)=A(N, L) * B(L, M)
$$

SUBROUTINE MATMUL(N,L,M,A,B,C)

$$
\text { DOUBLE PRECISION A }(1 \varnothing, 1 \theta), B(1 \theta, 1 \theta), C(1 \varnothing, 1 \theta)
$$

$$
\text { DO } 1 \varnothing \mathrm{I}=1, \mathrm{~N}
$$

$$
\text { DO } 10 \mathrm{~J}=1, \mathrm{M}
$$

$$
C(I, J)=\varnothing . \varnothing
$$

$$
\text { DO } 2 \sigma \quad \mathrm{I}=1, \mathrm{~N}
$$

$$
\text { DO } 2 \varnothing \mathrm{~J}=1, \mathrm{M}
$$

$$
\text { DO } 2 \varnothing \mathrm{~K}=1, \mathrm{~L}
$$

$$
C(I, J)=C(I, J)+A(I, K) * B(K, J)
$$

RETURN
END
***SUBROUTINE FOR MATRICES ADDITION
$C(N, M)=A(N, M)+B(N, M)$
SUBROUTINE MATADD(N,M,A,B,C)
DOUBLE PRECISION A $(1 \varnothing, 1 \theta), B(1 \varnothing, 1 \varnothing), C(1 \theta, 1 \theta)$
DO $1 \varnothing \mathrm{I}=1, \mathrm{~N}$
DO $1 \varnothing \mathrm{~J}=1, \mathrm{M}$
$C(I, J)=A(I, J)+B(I, J)$
RETURN
END
***SUBROUTINE FOR MATRICES SUBTRACTION
$C(N, M)=A(N, M)-B(N, M)$
SUBROUTINE MATSUB( $N, M, A, B, C$ )
DOUBLE PRECISION A(1 $0,1 \varnothing), B(1 \varnothing, 1 \varnothing), C(1 \varnothing, 1 \varnothing)$
DO $1 \varnothing I=1, N$
DO $1 \varnothing \mathrm{~J}=1, \mathrm{M}$
$C(I, J)=A(I, J)-B(I, J)$
RETURN
END

```
O
C
C
    DO l\emptyset I=l,N
    DO 10 J=1,M
    B(I,J)=A(I,J)
        RETURN
        END
    ***SUBROUTINE FOR IDENTITY MATRIX
    A(N,N)=IDENTITY MATRIX
    SUBROUTINE IDENTY(N,A)
    DOUBLE PRECISION A(1\emptyset,l\emptyset)
    DO 10 I=l,N
    DO 10 J=1,N
    A(I,J)=\emptyset., (
    DO 2\emptyset I=1,N
    A(I,I)=1.\emptyset
    RETURN
    END
C
C
C
C
C
    ***SUBROUTINE FOR TRANSPOSE OF A MATRIX
    B(M,N) TRANSPOSE OF A(N,M)
    SUBROUTINE MATRAN(N,M,A,B)
    DOUBLE PRECISION A(1\varnothing,1\varnothing),B(1\emptyset,1\emptyset)
    DO 1\emptyset I=1,N
    DO 1\emptyset J=1,M
    B(J,I)=A(I,J)
    RETURN
    END
***SUBROUTINE FOR INVERSE OF A SQUARE MATRIX
AINV(N,N) INVERSE OF \(A(N, N)\)
```

```
C
SUBROUTINE MATINV(N,A,AINV)
C
        DOUBLE PRECISION A(10,10),AINV(10,10),P(20),DP
        DOUBLE PRECISION BMAINV(10,10),ABMINV(10,10),EMAX,E
        DOUBLE PRECISION IDENT2(10,10),AMTINV(10,10)
    C
    C CROUT DECOMPOSITION
C
        CALL MATCOP(N,N,A,AMTINV)
        IA=1|
        IFAIL=1
        CALL FOIBTF(N,AMTINV,IA, P,DP,IFAIL)
        IF(IFAIL.EQ.\sigma) GO TO 2\varnothing
        WRITE(1,10) IFAIL
        FORMAT(lH ,'ERROR ON FglBTF IFAIL=',I2)
        STOP
C IDENTITY MATRIX
20 CALL IDENTY(N,IDENT2)
        CALL IDENTY(N,BMAINV)
        IR=N
        IB=6
C
C APPROXIMATE SOLUTION OF LINEAR EQUATIONS
C
    CALL FG4AYF(N,IR,AMTINV,IA,P,BMAINV,IB,IFAIL)
C
C ITERATION FOR PRECISION
C
EMAX=\varnothing. Øøøøø\emptysetø01
llø CALL MATMUL(N,N,N,A,BMAINV,ABMINV)
        CALL MATSUB(N,N,IDENT2,ABMINV,ABMINV)
        DO 80 I=l,N
        DO 8\emptyset J=1,N
        E=DABS (ABMINV (I,J))
        IF(E.LT.EMAX) GO TO 80
        GO TO 90
        CONTINUE
        GO TO 120
90 CALL F04AYF(N,IR,AMTINV,IA,P,ABMINV,IB,IFAIL)
        CALL MATADD(N,N,BMAINV,ABMINV,BMAINV)
        GO TO llD
C RESULT
12ø CALL MATCOP(N,N,BMAINV,AINV)
        RETURN
        END
C
C
C
C
SUBROUTINE FOR MATRIX TIMES VARIABLE
C
C B(N,M)=VARIAB*A(N,M)
C
SUBROUTINE MATVRB(N,M,A,VARIAB,B)
```

```
DOUBLE PRECISION A \((1 \varnothing, 1 \varnothing), B(1 \varnothing, 1 \varnothing)\), VARIAB
DO \(10 \mathrm{I}=1, \mathrm{~N}\)
DO \(1 \varnothing \mathrm{~J}=1, \mathrm{M}\)
\(B(I, J)=V A R I A B * A(I, J)\)
RETURN
END
```


## APPENDIX 4.6

OBSEQU PROGRAM
C OBSERVER EQUATIONS
DOUBLE PRECISION PHI $(10,10), L(10,10), \operatorname{PAA}(10,10), \operatorname{PAB}(10,10)$
DOUBLE PRECISION PBB(10,10), $\operatorname{PBA}(10,1 \theta), L A(1 \theta, 10), L B(10,1 \theta)$

DOUBLE PRECISION PLOPAB ( $10,1 \emptyset$ ), PLOPAA $(10,1 \sigma), L B L O L A(10,10)$
DOUBLE PRECISION PAALO $(10,10), \operatorname{PLOLO}(10,10)$
INTEGER N,M,NN,MM1, I, J, Ml
WRITE (1,5)
FORMAT (/)
$1 \emptyset$ CALL TNOUA('ENTER SYST ORDER= ', 18)
$\operatorname{READ}(1, *) N$
WRITE (1,5)
CALL TNOUA('ENTER OBSERVER ORDER= ',22)
$\operatorname{READ}(1, *) \quad M$
$\mathrm{Ml}=\mathrm{N}-\mathrm{M}$
IF (Ml.LE. Ø) GO TO 1Ø
WRITE (1, 20)
$2 \emptyset$ FORMAT (1H, /,'ENTER ORDERED SYSTEM MATRIX BY LINE')
$\operatorname{READ}(1, *)((\operatorname{PHI}(I, J), J=1, N), I=1, N)$
WRITE (1, 3ø)
FORMAT (IH ,/,'ENTER ORDERED INPUT MATRIX BY COLUMN')
$\operatorname{READ}(1, *)(L(I, 1), I=1, N)$
DO $4 \emptyset I=1$, M1
DO $4 \emptyset \mathrm{~J}=1, \mathrm{Ml}$
$\operatorname{PAA}(I, J)=\operatorname{PHI}(I, J)$
40 CONTINUE
MMI $=$ MI +1
DO $50 \mathrm{I}=1, \mathrm{Ml}$
DO $50 \mathrm{~J}=\mathrm{MM1}, \mathrm{~N}$
$\mathrm{Jl}=\mathrm{J}-\mathrm{Ml}$
$\operatorname{PAB}(I, J I)=\operatorname{PHI}(I, J)$
50 CONTINUE
DO 60 I=MMI,N
Il=I-Ml
DO $6 \emptyset \mathrm{~J}=1, \mathrm{Ml}$
$\operatorname{PBA}(I 1, J)=\operatorname{PHI}(I, J)$
CONTINUE
DO $7 \emptyset$ I=MMI, N
I $1=I-M I$
DO $70 \mathrm{~J}=\mathrm{MML}, \mathrm{N}$
$\mathrm{J} 1=\mathrm{J}-\mathrm{Ml}$
$\operatorname{PBB}(I I, J 1)=\operatorname{PHI}(I, J)$
DO $80 \mathrm{I}=1, \mathrm{MI}$
$\mathrm{LA}(\mathrm{I}, 1)=\mathrm{L}(\mathrm{I}, 1)$
CONTINUE
DO $9 \emptyset$ I=MMI, N
II $=1-M I$
$\operatorname{LB}(I I, 1)=L(I, 1)$
90 CONTINUE
WRITE (1,100)
100
FORMAT(1H ,/,'ENTER OBSERVER FEED GAINS BY LINE')
$\operatorname{READ}(1, *)((L O(I, J), J=1, M 1), I=1, M)$

CALL MATMUL (M, MI, M, LO , PAB, LOPAB)
CALL MATMUL (M, MI, MI, LO, PAA, LOPAA)
$\mathrm{NN}=1$
CALL MATMUL (M, M1, NN, LO, LA, LOLA)
CALL MATSUB (M, M, PBB, LOPAB, PLOPAB)
CALL MATSUB (M, Ml, PBA, LOPAA, PLOPAA)
NN=1
CALL MATSUB (M, NN, LB, LOLA, LBLOLA)
CALL MATMUL (M, M, MI, PLOPAB, LO, PLOLO)
CALL MATADD(M,M1, PLOPAA, PLOLO, PAALO)
WRITE (1,11Ø)
110 FORMAT(1H ,/,'PBB-LO*PAB')
DO $120 \mathrm{I}=1, \mathrm{M}$
WRITE ( $1,13 \theta$ ) (PLOPAB (I, J), J=1, M)
12g CONTINUE
130 FORMAT (1H,6F12.4)
WRITE (1,180)
180 FORMAT (1H , /, (PBA-LO*PAA) +(PBB-LO*PAB)*LO')
DO $140 \quad \mathrm{I}=1, \mathrm{M}$
WRITE (1, 13Ø) (PAALO (I, J) , J=1, M1)
140 CONTINUE
WRITE (1, 159)
150 FORMAT (1H ,/,'LB-LO*LA')
DO $160 \quad \mathrm{I}=1, \mathrm{M}$
WRITE ( 1,130 ) LBLOLA $(1,1)$
160
CONTINUE
STOP
END

TIME RESPONSE

```
DOUBLE PRECISION PHIX(10,10), PHI(10,10),LU(1\emptyset,10)
DOUBLE PRECISION VAR,XPLUS(10,10),U(20\emptyset),XMINUS(10,10)
DOUBLE PRECISION L(10,1\emptyset)
DOUBLE PRECISION HEIGHT,G1
INTEGER I,J,K,M,NN1
DOUBLE PRECISION KM1(10,10),KM2(10,10),G2(10,10)
DOUBLE PRECISION LOBLW(10,10)
DOUBLE PRECISION RPHILG(10,10),RLG(10,10), BB(20,15)
DOUBLE PRECISION RR(10),RI(10),VR(15,10),UU(30),VV(10)
DOUBLE PRECISION VI (15,10),KC,KCMIN,KCMAX,KCSTEP,RLB,RUB
DOUBLE PRECISION XI,TW,XIØ,E,X,Y
REAL RR1(19),RI1(10),VXBEG,VYBEG,VXEND,VYEND, X1,Y1
INTEGER NXINTS,NYINTS,IA, IB, IFAIL,IVI,IVR,J,MM,MMM,N
LOGICAL CC(10)
DOUBLE PRECISION G(10,10),LG(10,10),LINPUT(10,10)
DOUBLE PRECISION PHILG(10,10)
INTEGER INTGER(12),ICNT(12)
REAL AAAl,BBBI
REAL DKSTEP,YY1
INTEGER KSTEP,KSTEP2,KSTEP3,NSTEP
DOUBLE PRECISION Z,ZI,WNT,TETA
```

FORMAT (1H)
CALL TNOUA('ENTER ORDER $N=$ ', 15)
$\operatorname{READ}(1, *) N$
$\mathrm{NN}=\mathrm{N}$
WRITE (1, 30)
FORMAT (1H ,/,'ENTER MATRIX PHI BY LINE')
WRITE (1,10)
$\operatorname{READ}(5, *)((\operatorname{PHI}(I, J), J=1, N), I=1, N)$
WRITE (1, 4Ø)
40 FORMAT (1H,$/,{ }^{\prime} E N T E R$ MATRIX L BY COLUMN')
$\operatorname{READ}(5, *)(L(1,1), I=1, N)$
WRITE (1,10)
CALL TNOUA('ENTER OBSERVER ORDER =', 22)
$\operatorname{READ}(1, *) \mathrm{NOB}$
IF (NOB.EQ. Ø) GO TO 49
WRITE (1, 42)
FORMAT (1H ,/,'ENTER FEEDBACK GAIN MATRIX KM1 BY LINE')
WRITE (1,1ø)
$\operatorname{READ}(1, *)(\operatorname{KM1}(1, I), I=1, N)$
WRITE (1, 44)
44 FORMAT (1H,/,'ENTER FEEDBACK GAIN MATRIX KM2 BY LINE')
WRITE (1,10)
$\operatorname{READ}(1, *)(\operatorname{KM} 2(1, I), I=1, N O B)$
WRITE (1, 46)
FORMAT (1H, /,'ENTER OBSERVER MATRIX LOBLW BY LINE')
WRITE ( 1,10 )
$\operatorname{READ}(I, *)((\operatorname{LOBLW}(I, J), J=1, N), I=1, N O B)$
$M=1$

```
    CALL MATMUL(M, NOB,N,KM2, LOBLW,G2)
    CALL MATADD(M,N,G2, KM1,G)
    GO TO 70
49 WRITE(1,50)
50 FORMAT(1H ,/,'ENTER MATRIX G BY LINE')
    WRITE(1,10)
    READ(1,*)(G(1, I),I=1,N)
7Ø WRITE(1,60)
60 FORMAT(1H,/,'ENTER INPUT GAIN GI')
    WRITE(1,10)
    READ(1,*) Gl
    J=l
    CALL MATVRB(N,J,L,Gl,LINPUT)
    J=1
    CALL MATMUL(N,J,N,L,G,LG)
    CALL MATSUB(N,N,PHI,LG, PHILG)
7\emptyset\emptyset\emptyset CALL TNOUA('ROOT LOCUS(1),TIMERESPONSE(2) ',30)
    READ(l,*) IA
    IF(IA.EQ.l) GO TO 305ø
    WRITE(1,1Ø)
    CALL TNOUA('ENTER SAMPLING PERIOD T(MS)=',28)
    READ(1,*) T
    T=T*Ø.\emptyset\emptyset1
    WRITE(1,10)
    CALL TNOUA('NUMBER OF STEPS(SOLUTION)=',26)
    READ(1,*) KSTEP
    WRITE(1,10)
    CALL TNOUA('STEP(1) OR RAMP(2) ',19)
    READ(1,*) I
    IF(I.EQ.2) GO TO 15
    WRITE(1,10)
    CALL TNOUA('ENTER STEP HEIGHT (MM)=', 22)
    READ(1,*) HEIGHT
    HEIGHT=HEIGHT*Ø.\emptyset\emptysetl
    M=1
    DO 1Ø\emptyset K=1,KSTEP
10\emptyset U(K)=HEIGHT
    GO TO 30\emptyset
15 WRITE (1,10)
    CALL TNOUA('ENTER SPEED (MM/SEC)=',21)
    READ(1,*) ACCELE
    ACCELE=ACCELE*Ø. Ø\emptyset1
    KSTEP2=KSTEP/2+1
    U(1)=0
    DO 2ØØ K=2,KSTEP2
2\emptyset\emptyset U(K)=(K-1)*T*ACCELE
    KSTEP3=KSTEP2+1
    DO 21\emptyset K=KSTEP3,KSTEP
2l\emptyset U(K)=U(KSTEP2)
3\emptyset\emptyset CALL T401\emptyset
    CALL PICCLE
    CALL DEVEND
    CALL TNOUA('PRINT STATE VARIABLE=',21)
    READ(1,*) KSTATE
```

```
        CALL TNOUA('mAX VALUE VAR=',14)
        READ(1,*) XMAX
        CALL TNOUA('MIN VALUE VAR=',14)
        READ(l,*) XMIN
        DKSTEP=FLOAT(KSTEP)
        Yl=120.0
        CALL T4010
        W=100.0
        V=50.0
        CALL AXIPLO(\emptyset,W,V,l,l,KSTEP,1\emptyset,\emptyset,DKSTEP,XMIN,XMAX,'TIME',4,'VAR',
        *)
        CALL GRID ( }3,\varnothing,\varnothing
        NSTEP=1
        DO 40Ø K=1,N
        XMINUS (K,M)=\varnothing
        NSTEP=NSTEP+1
        IF(NSTEP.LE.KSTEP) GO TO 350
        CALL CHAPOS(\emptyset,Y1)
701 WRITE(1,7ø0)
70\emptyset FORMAT(1H ,'CHANGE OBSERVER(1),GAIN(2),STOP(3),CONTINUE(4)')
        READ(1,*) I
        CALL PICCLE
        IF(I.EQ.l) GO TO 4l
        IF(I.EQ.2) GO TO 49
        IF(I.EQ.3) GO TO 30øø
        IF(I.EQ.4) GO TO 3øø
        GO TO 701
350 M=1
        CALL MATMUL(N,N,M,PHILG,XMINUS,PHIX)
        VAR=U (NSTEP-1)
        CALL MATVRB(N,M,LINPUT,VAR,LU)
        CALL MATADD(N,M,PHIX,LU,XPLUS)
        YY1=SNGL(XPLUS(KSTATE,M))
        DNSTEP=FLOAT(NSTEP-1)
        IF(XMAX-YY1) 50\emptyset,501,501
501 IF(YY1-XMIN) 500,501,502
502
        CALL GRAMOV(DNSTEP,YYI)
        CALL DOT(3)
        CALL MATCOP(N,M,XPLUS,XMINUS)
        GO TO 5ø\emptyset
3000 CALL DEVEND
        STOP
C *****************
C ROOT LOCUS
C
C
3ø50 CALL T4ø10
    CALL PICCLE
        CALL DEVEND
4øø\emptyset CALL TNOUA('ENTER MIN AND MAX GAINS (KCMIN,KCMAX) ',4Ø)
        READ(1,*) KCMIN,KCMAX
        CALL TNOUA('ENTER GAIN STEP= ',17)
```

```
    READ(1,*) KCSTEP
    CALL TNOUA('ENTER RANGE ON REAL AXIS (MIN,MAX,STEPS) ',40)
    READ(1,*) VXBEG,VXEND,NXINTS
    CALL TNOUA('ENTER RANGE ON IMAG AXIS (MIN,MAX,STEPS) •,40)
    READ(1,*) VYBEG,VYEND,NYINTS
    KC=KCMIN
    CALL T4010
    CALL AXIPLO(1,90.0,90.0, 2,2,NXINTS,NYINTS,VXBEG,VXEND,VYBEG,
    *VYEND,'REAL', },\mathrm{ 'IMAG', Ø)
    CALL GRAMOV(1.ø.\emptyset.ø)
    TW=\varnothing.\emptyset
4045 Xl=SNGL(DCOS(TW))
    Yl=SNGL(DSIN(TW))
    CALL GRALIN(X1,Y1)
    TW=TW+.02
    IF(TW.LE.6.3ø) GO TO 4045
4040 CALL MATVRB(N,N,LG,KC,RLG)
    CALL MATSUB(N,N,PHI,RLG,RPHILG)
    RLB=\varnothing.\emptyset
    RUB=10. }
    MM=NN1
    IA=1\varnothing
    IB=2\emptyset
    IVR=15
    IVI=15
    IFAIL=1
    CALL F\emptyset2BCF(RPHILG,IA,NN1,RLB,RUB,MM,MMM,RR,RI,VR,IVR,VI,IVI,
    *INTGER,ICNT,CC,BB,IB,UU,VV,IFAIL)
    IF(IFAIL.EQ.\emptyset) GO TO 40lø
    WRITE(1,4005) IFAIL
4005 FORMAT(1H ,'ERROR Fø2BCF',I2)
    STOP
401\emptyset DO 402\emptyset I=1,MMM
    AAA1=SNGL(RR(I))
    RR1 (I)=AAA1
    IF(AAAI.GT.VXEND) GO TO 4030
    IF(AAAI.LT.VXBEG) GO TO 4030
4012 BBBl=SNGL(RI(I))
    RIl(I)=BBBI
    IF(BBBI.GT.VYEND) GO TO 4030
    IF(BBBI.LT.VYBEG) GO TO 4ø3ø
4020 CONTINUE
    CALL GRASYM(RRI,RI1,MMM, 3, Ø)
403ø KC=KC+KCSTEP
    IF(KC.LE.KCMAX) GO TO 4040
409ø CALL GRASPA(VXBEG,VYBEG,XSP,YSP)
    YSP=YSP-10. }
    CALL SPAGRA(XSP,YSP,XGR,YGR)
    CALL CHAPOS( }0.0,1\varnothing.|
    WRITE(1,4095)
4095 FORMAT(1H ,'CONTINUE(1),CHANGE FEED GAINS(2),CTE DAMP(3)')
    CALL TNOUA('CTE WNT(4),CTE XIWNT(5),STOP(6) 1,32)
    READ(1,*) IA
    IF(IA.EQ.6) GO TO 60øø
```

```
    IF(IA.EQ.5) GO TO 7010
    IF(IA.EQ.2) GO TO 49
    IF(IA.EQ.3) GO TO 4050
    IF(IA.EQ.4) GO TO 5010
    CALL CHAPOS(0.0,138.0)
    GO TO 40\emptyset\emptyset
4050 XI=0.1
4070 TW=\emptyset.\emptyset
    CALL GRAMOV(1.0, Ø. Ø)
4|6\emptyset XI\emptyset=XI/(DSQRT(1.0-XI**2))
    E=DEXP(- (XI\sigma*TW))
    X=E*DCOS (TW)
    Y=E *DS IN (TW )
    Xl=SNGL(X)
    Yl=SNGL(Y)
    CALL GRALIN(XI,Yl)
    TW=TW+. . }
    IF(TW.LE.3.15) GO TO 4060
    XI=XI+.l
    IF(XI.LE.1.0) GO TO 407\emptyset
    GO TO 4090
5010 WNT=\emptyset.\emptyset
5\emptyset2\emptyset WNT=WNT+.25
    IF(WNT.GT.3.25) GO TO 4090
    TETA=\varnothing.\emptyset
    X=DEXP(- (WNT*(DCOS (TETA))))
    Xl=SNGL(X)
    CALL GRAMOV(X1,0.0)
5Ø30 TETA=TETA+.01
    IF(TETA.GT.1.58) GO TO 502\emptyset
    Z=(DEXP(-(WNT*(DCOS(TETA)))))
    Zl=WNT*DSIN(TETA)
    IF(Zl.GT.3.14) GO TO 5ø2\emptyset
    X=Z*(DCOS(Zl))
    Y=Z**2-X**2
    Y=DSQRT (Y)
    Xl=SNGL(X)
    YI=SNGL(Y)
    CALL GRALIN(XI,YI)
    GO TO 5\emptyset30
C LOCI OF ROOTS OF CTE XI.WN.T
701Ø WNT=1.\emptyset
7020 WNT=WNT-.1
    IF(WNT.LT.\emptyset.1) GO TO 4ø9\emptyset
    TETA=\varnothing.D
    X=WNT*DCOS (TETA)
    Y=DSQRT (WNT**2-X**2)
    XI =SNGL (X)
    CALL GRAMOV(X1,0.0)
7Ø30 TETA=TETA+.ø2
    IF(TETA.GT.3.15) GO TO 7020
    X=WNT *DCOS (TETA)
    Y=DSQRT (WNT**2-X**2)
    X1=SNGL(X)
```

Yl =SNGL (Y)
CALL GRALIN(X1, Y1)
GO TO 7030
6øøø CALL DEVEND
STOP
END

TM 990/100M-1 LOGIC DIAGRAMS AND PIN ASSIGNMENT

L1.- - - $+0 v$



EPROM MEMORY

ram memory


DATA/ADDRESS BUS BUFFERS



MULTI-DROP INTERFACE (

| $\begin{aligned} & \text { PI } \\ & \text { PIN } \end{aligned}$ | SIGNAL | $\begin{aligned} & \text { PI } \\ & \text { PIN } \end{aligned}$ | SIGNAL | $\begin{aligned} & \text { P1 } \\ & \text { PIN } \end{aligned}$ | SIGNAL |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 33 | DO.B | 71 | A14.8 | 12 | INTi3 8 |
| 34 | 01.8 | 72 | A15.8 | 11 | INT14.B |
| 35 | 02.8 | 22 | 97.8 | 14 | $\overline{\text { INTITS.B }}$ |
| 36 | D3.B | 24 | 93. ${ }^{\text {B }}$ | 28 | EXTCUX $B$ |
| 37 | D4.B | 92 | H0LO. ${ }^{\text {a }}$ | 3 | $+5 \mathrm{~V}$ |
| 38 | 05.8 | 86 | HOLDA. | 4 | +5V |
| 39 | D6.E | 82 | DEIN. 8 | 97 | +5V |
| 40 | D7.B | 26 | CLK. 8 | 98 | $+5 \mathrm{~V}$ |
| 41 | D8.B | 80 | MEMEN. 3 | 75 | +12V |
| 42 | 09.8 | 84 | ME! ${ }^{\text {MCTC.B }}$ | 76 | +12V |
| 43 | D10.B | 78 | WE. B | 73 | -12V |
| 44 | D11.8 | 90 | READY. ${ }^{\text {che }}$ | 74 | -12V |
| 45 | D12.8 | 87 | CRUCLK. | 1 | GND |
| 46 | D13.8 | 30 | CRUOUT.B | 2 | GND |
| 47 | D14.8 | 29 | CRUIN B | 21 | GND |
| 48 | D15.B | 19 | IAQ. ${ }^{\text {P }}$ | 23 | GND |
| 57 | AO.B | 94 | PRES $B$ | 25 | GND |
| 58 | A1. B | 88 | TिलST. | 27 | GND |
| 59 | A2.B | 16 | INTT. 8 | 31 | GND |
| 60 | A3.B | 13 | (NT2.B | 77 | GNO |
| 61 | A4. $\mathrm{B}^{\text {A }}$ | 15 | INT3. ${ }^{\text {a }}$ | 79 | GND |
| 62 | A5.B | 18 | [iNT4. 8 | 81 | GND |
| 63 | A6.B | 17 | [ NT T5. B | 83 | GND |
| 64 | A7.B | 20 | $\sqrt{\text { NT6 }} \mathrm{B}$ | 85 | GND |
| 65 | A8. ${ }^{\text {A }}$ | 6 | INT7. 8 | 89 | GNO |
| 66 | A9.B | 5 | INTE 8 | 91 | GNO |
| 67 | A10.B | 8 | INTG. ${ }^{\text {d }}$ | 99 | GND |
| 68 | Al1. ${ }^{\text {A }}$ | 7 | $\overline{\text { INT }} \overline{10} \mathrm{~B}$ | 100 | GNO |
| 69 | A12.B | 10 | NTIT: ${ }^{\text {N }}$ | 93 | RESTART. 8 |
| 70 | A13.B | 9 | INT12.8 |  |  |


| P4 PIN | SIGNAL | $\begin{aligned} & \text { P4 } \\ & \text { PIN } \end{aligned}$ | SIGNAL |
| :---: | :---: | :---: | :---: |
| 20 | PO | 17 | GND |
| 22. | P1 | 15 | GND |
| 14 | P2 | 13 | GND |
| 16 | P3 | 11 | GND |
| 18 | P4. | 9 | GND |
| 10 | P5 | 39 | GND |
| 12 | P6 | 37 | GND |
| 24 | $\sqrt{N}+15$ or P7 | 35 | GND |
| 26 | INT14 or Pg | 33 | GND |
| 28 | inti 3 or P9 | 31 | GNO |
| 30 | INT 12 or P10 | 29 | GND |
| 32 | INT1T or P11 | 27 | GND |
| 34 | INTTO or P12 | 25 | GNO |
| 36 | INT9 or P13 | 23 | GND |
| 38 | INTS or P14 | 21 | GND |
| 40 | [NT7 or P15 | 19 | GND |
| 7 | INT 6 | 1-6 | Spares |


$R D=R E A D D A T A$

APPENDIX 6.2.1


APPENDIX 6.2.2



MANUFACTURING MASK FOR MEMORY EXPANSION BOARD

APPENDIX 6.2.3 (cont.)

MEMORY EXPANSION BOARD PIN-SIGNAL ASSIGNMENT

| MEMORY EXPANSION BOARD PIN-SIGNAL ASSIGNMENT |  |  |  |
| :---: | :---: | :---: | :---: |
| PIN | SIGNAL | PIN | SIGNAL |
| 1 | GND | 2 | READY |
| 3 | $+5 \mathrm{v}$ | 4 | $\overline{\text { ¢ } 1}$ |
| 9 | MEMEN. BF | 6 | $\overline{\mathrm{WE}}$ |
| 11 | DBIN. BF | 8 | MEMEN |
| 13 | $\overline{\text { ¢1 }}$. BF | 10 | DBIN |
| 15 | $\overline{\mathrm{WE}} . \mathrm{BF}$ | 34 | A7 |
| 17 | D15 | 35 | A8 |
| 19 | D14 | 36 | A5 |
| 21 | D13 | 37 | A9 |
| 23 | D12 | 38 | A4 |
| 25 | D11 | 39 | Al0 |
| 27 | D10 | 40 | A6 |
| 29 | D9 | 41 | All |
| 31 | D8 | 42 | A3 |
| 49 | D7 | 43 | A12 |
| 51 | D6 | 44 | A2 |
| 53 | D5 | 45 | A13 |
| 55 | D4 | 46 | A1 |
| 57 | D3 | 47 | A14 |
| 59 | D2 | 48 | AO |
| 61 | D1 | 12 | HOLDA |
| 63 | D0 | 83 | GND |
|  |  | 85 | +5v |


| ID | TYPE | DESCRIPTION |
| :--- | :---: | :---: |
| al | D2111AL-4 | 256-word by 4-bit Static RAM |
| a2 | $"$ | $"$ |
| a3 | $"$ | $"$ |
| a4 | $"$ | $"$ |
| b1 | D211AL-4 | 256-word by 4-bit Static RAM |
| b2 | $"$ | $"$ |
| b3 | $"$ | $"$ |
| b4 | $"$ | $"$ |
| c1 | D2111AL-4 | 256-word by 4-bit Static RAM |
| c2 | $"$ | $"$ |
| c3 | $"$ | $"$ |
| c4 | $"$ | $"$ |
| d1 | D2111AL-4 | 256-word by 4-bit Static RAM |
| d2 | $"$ | $"$ |
| d3 | $"$ | $"$ |
| d4 | $"$ | $"$ |
| e | SN74LS345N | Octal Bus Transceiver (3-state) |
| f | SN74LS345N | Octal Bus Transceiver (3-state) |
| g | SN74LS10N | Trip1e 3-Input Positive NAND |
| h | SN74LS241N | Octal Buffer (3-state) |
| i | SN74LS245N | Octal Bus Transceiver (3-state) |
| m | SN74LS245N | Octal Bus Transceiver (3-state) |
| n | SN74LS74N | Dual D-Type F1ip F1op |
| p | SN74LS20N | Dual 4-Input Positive NAND |
| q | SN74LS155N | Dual 2-1ine-to-4-1ine Decoder |
| r | SN74LS04N | Hex Inverters |

## APPENDIX 6.5.1

## LIST AND DESCRIPTION OF COMPONENTS

FOR ADDRESS BUS BOARD

| ID | TYPE | DESCRIPTION |
| :---: | :--- | :---: |
| 1A | SN74150 | 1-of-16 Data Selector Multiplexer |
| 2A | SN74150 | $"$ |
| 3A | SN74LS245 | Octal Bus.Transceiver (3-state) |
| 4A | SN74LS245 | Octal Bus Transceiver (3-state) |
| 5A | SN74LS241 | Octal Buffer (3-state) |
| 6A | SN74LS30 | 8-Input Positive NAND Gate |
| 7A | SN73LS08 | Quadruple 2-Input Positive AND |
| 8A | SN74LS08 | $"$ |
| 9A | SN74LS08 | $"$ |
| 10A | SN74LS08 | " |
| 11A | SN74LS26 | Quadruple 2-Input NAND (Open) |
| 12A | SN74LS14 | Hex Schmitt Trigger Inverter |
| 13A | SN74LS73 | Dual J-K F1ip Flop with Clear |
| 14A | SN74LS08 | Quadruple 2-Input Positive AND |
| 15A | SN74LS14 | Hex Schmitt Trigger Inverter |
| 16A | SN74LS08 | Quadruple 2-Input Positive AND |

## APPENDIX 6.5.2

LIST AND DESCRIPTION OF COMPONENTS
FOR DATA BUS BOARD

| ID | TYPE | DESCRIPTION |
| :---: | :---: | :---: |
| 6A | SN74LS00 | Quadruple 2-Input Positive NAND |
| 8A | SN74LS01 | Quadruple 2-Input NAND (Open) |
| 2C | SN74LS86 | Quadruple 2-Input Exclusive OR |
| 5C | SN74LS14 | Hex Schmitt Trigger Inverter |
| 6C | SN74LS14 | $"$ |
| 7C | SN74LS14 | $"$ |
| 8C | SN74LS14 |  |
| 4D | SN74LS245 | Octal Bus Transceiver (3-state) |
| 5D | SN74LS245 | " |
| 6D | SN74LS245 | " |
| 7D | SN74LS26 | Quadruple 2-Input NAND (Open) |

## APPENDIX $7: 2$

## OPEN COLLECTOR BUS DRIVERS AND LINE TERMINATION

Open collector drivers are used for the slave identification bus, HOLDA.B and HOLD.B lines, the interrupt controller (INTCTL) line and the bus busy signal line.

The necessary pull-up resistors for the open collector drivers are replaced by two resistors RL1 to +5 V and RL2 to ground. These resistors are chosen so that their Thevenin equivalent network has a voltage generator of +3 V , and an impedance equal to the driven line impedance ( $Z 0$ ). Assuming a characteristic line impedance of $180 \Omega$ ( $\mathrm{S}-100$ ) resistors RLl and RL2 are calculated according to:

$$
\begin{aligned}
3 & =5\left(\frac{\mathrm{RL} 2}{\mathrm{RL} 1+\mathrm{RL} 2}\right) \\
180 & =\left(\frac{\mathrm{RL} 1 . \mathrm{RL} 2}{\mathrm{RL} 1+\mathrm{RL} 2}\right)
\end{aligned}
$$

which gives RL1 $=300 \Omega$ and RL2 $=470 \Omega$ (standard).
In the case of the uni-directional lines as the interrupt controller and busbusy line, which configuration is shown in Figure 6.5.3, each NAND gate in the worst possible condition must be capable of sinking the current coming from the resistor network and the inverter low level input. For a 74 LS 14 , these currents add up to $17 \mathrm{~mA}(16.6+0.4 \mathrm{~mA})$ and the open collector NAND driver must be the 7438 or 74 S 38 with a low level output current of 48 mA .

For the bi-directional lines, as the slave identification bus and others, shown in Figure 6.5.2, there must be termination at both ends. In this case, each NAND gate must sink the current coming from the two resistor networks and ten inverter low level inputs. For 74LS14 inverters, these currents add up to $37.2 \mathrm{~mA}(2 \times 16.6+10 \times 0.4 \mathrm{~mA})$. This current is compatible with the maximum low level current for the 74 S 38 device.

## Configuration for unidirectional lines:

- Interrupt Controller and Bus Busy Line


Configuration for bi-directional lines:


## APPENDIX 7.3

## MEMORY TEST PROGRAM

2.3.0 78.244 00:16:23 01/01/00

F'AGE 0001

TXMIFA
MORY TEST ROUTINE

0028 5555
$002 A$ C043
002C C444
002E 8444
0030 160A
0032 C445
0034 8445
0036160 A
0038 05C1
003A 8081
OO3C 1BGA
OOBE 1 GF 6

| **** |  |  |  |
| :---: | :---: | :---: | :---: |
| MEMTE | LI | $\mathrm{Fi} 1,7 \mathrm{FOOD}$ | STAFTING ADDFESS |
|  | MOV | Fil, F3 | SAVE ADDFESS |
|  | XOF' | @LFCR, 14 | LFCF' |
|  | LI | F2, >F3FE | END ADDFESS |

*************************************

* GLOEAL TEST
* 
* WRITES ON LOCATION ITS ON ADDFESS * *************************************

| MOU | MOV | F1, *F1+ | WFITE ADDRESS ON |
| :---: | :---: | :---: | :---: |
|  | C | F1, F 2 | FINISHED? |
|  | JH | OUT |  |
|  | JMF | MOV |  |
| OUT | MOU | F3, R1 | CHECKING |
| COMF | C | F1, *R1+ | COMF'AFE |
|  | JNE | MESS1 | MESSAGE ON ERROR |
|  | C | R1, F 2 | FINISHED? |
|  | JH | QUT2 |  |
|  | JMF' | COMF |  |

*******************************************

* AAAA AND 5555 TEST
* 
* WRITES UALUES AAAA AND 5555 ON LOCATION * *******************************************
OUT2 LI R4, >AAAA
LI K (, >5555


```
**************************************************
IDT 'MEMTE'
DEF MEMTE
信 *THE STAFTING ADDFESS MUST EE GIVEN ON LOCATION * * 日OD2 AND THE FINAL ADDKESS ON LOCATION BODC. * *WHEN AN EKFOR IS FQUND A COFRESFONDENT MESSAGE * *IS DISFLAYED. IT USES FEGISTEFS FI TO FKS AND IF * *THERE IS AN EFFOF FEGISTEF FI WILL CONTAIN THE *
```

MOU K1, R3 SAVE ADDRESS

LI F 2, PF3FE END ADDFESS

COMFAFE
MESSAGE ON EFROR
FINISHED?

K3, R1
MOUZ MOU R4, *R1
C KA, *FI
JNE MESS?
MOV RS, *R1
C FiS, *FI
JNE MESS3
INCT R1
C $\mathrm{Ki}, \mathrm{Ki}$
JH MESS4
JMF MOUZ

STARTING ADDFESS WKITE AAAA
COMF'AFE
MESSAGE ON EFROR
WRITE 5555
COMF'AFE
MESSAGE ON EFROR
NEXT ADDFESS
FINISHED?
NO EFFORS


## APPENDIX 7.5

POWER SUPPLY SPECIFICATION

| POWER SUPPLY 1 |  |
| :---: | :---: |
| Source | Code |
| QUARNDON | QMS PSU1 |
| Voltage Range <br> (V) | Max Current (A) $40^{\circ} \mathrm{C}$ Ambient |
| $\begin{aligned} & 5 \pm 5 \% \\ & +12 \\ & -12 \\ & -\quad 5 \end{aligned}$ | $\begin{aligned} & 6.0 \\ & 1.0 \\ & 1.0 \\ & 0.1 \end{aligned}$ |
| Located at the | ack of the Rack |


| POWER SUPPLY 2 |  |
| :---: | :---: |
| Source | Code |
| VERO SPEED | $89-2665 \mathrm{G}$ |
| Voltage Range <br> (V) | Max Current (A) <br> $40^{\circ} \mathrm{C}$ Ambient |
| Output 1 <br> Output 2 | $5.5 \%$ |

APPENDIX 7.6.1

| FSSTI | TXMIFA | 2.3 .0 | 78.244 | $00: 18: 55$ | $01 / 01 / 00$ | F.AGE DOQ1 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| E FIFST TEST ROUTINE 1 |  |  |  |  |  |  |

0001 0002 0003 0004 0005 0006 0007 0009 0010 0011
0012 001300000300 00020000 00140004 020C 00060100 00080200 000 A 2000 0016 000C 31C0 0017 000E 0720 0010 F000
001800120300 00140005 001900160340 002000180460 001A 0080
3021
0022
0023
0024001 C 020 C $001 E 0100$ 00200200 0022 FOOO 00260024 1E00 00270026 1E14 00280028 1F18 0029 002A 16FE 0030 002C $1 E 12$ 0031 002E C050 00320030 16FE 003300321012 $003400341 F 18$ 00350036 13FE 003600380380 0037

## ****************************************** <br> *FIFST TEST * <br> *THIS FFIOGFAM WAS WFITTEN FOF CONTROLLEF * *MODULE. IT HELFS CHECKING THE INTERFUUT * *AKEITRATION LINK AND HANDSHAKE CONTFOL * *THE INTCTL LINE IS SEFUED EY INTS. * ****************************************** IDT 'FIFST1'

***************
*MAIN FROGFAM *
***************
FIFST1 LIMI 0
LI Fi12, >100
LI FO, >2000
LDCF $F 0,7$
SETO @ PF OQO
LIMI 5
IDLE
E @ 180
********************** *INTEFFUFT 5 ROUTINE $*$ **********************
INTS LI R12, >10日
LI $\mathrm{FO}, \mathrm{YFODO}$
SEZ 0
SEZ 20
TB 24
JNE LA
SEZ 18
MOU *FG, R1
JNE CA
SE:O 18
TE 24
JEQ TA
FTWF
END

DISAEILE ALL INTEFFUFPTS
TMS 9901 ADDRESS
ENAELE ONLY INTS

ON TMS 9901
FLAG FOF MESSAGE
ENAELE INTS ON MICRO
WAIT INTEFFUFT FETUKN TO MONITOK

TMS 9991 ADDFESS
LOCATION FOF MESSAGE
INTEFFIUFT MODE
SET RACKO LINE
EUS EUSY?
NO! FETUFN TO LA
ENAELE HOLD IN
MESSAGE TRANSMITTED?
NO!
DISAELE HOLD IN
EUS FREE?
NO!
RETUFN TO MAIN


0001 0002 3003 0004 0005 0006 0007 0009 0010 0011

001300000300
0002 0000
0014 0004 020C
00060100
00080200
000A 1000
0016 000C 31C0
0017 000E 1 E13
00.1800100300

00120004
001900140340
002000160460
0018 0080

| 0021 |  |  |
| :--- | :--- | :--- |
| 0022 |  |  |
| 0023 |  |  |
| 0024 | $001 A$ | $020 C$ |
|  | 001 C | 0100 |
| 0025 | $001 E$ | 0200 |
|  | 0020 | 0000 |
| 0026 | 0022 | 1013 |
| 0027 | 0024 | $1 E 11$ |
| 0028 | 0026 | $1 F 16$ |
| 0029 | 0028 | $16 F E$ |
| 0030 | $002 A$ | $C 800$ |
|  | $002 C$ | 1060 |
| 0031 | $002 E$ | 1011 |
| 0032 | 0030 | $1 F 16$ |
| 0033 | 0032 | $13 F E$ |
| 0034 | 0034 | 0380 |

******************************************
*FIRST TEST
*THIS FROGRAM WAS WRITTEN FOR MODULE 2 *
*IT HELFS CHECKING THE INTERFUFT *
*AFEITRATION LINK AND HANDSHAKE CONTROL *
*THE INTMST LINE IS SERVED EYY INTA. *
******************************************

## IDT 'FIFST2'

***************
*MAIN FROGRAM *
***************
FIRST2 LIMI 0
LI R12, $>100$
LI RO, 71000
LDCR RG, 7
SEZ 19
LIMI 4
IDLE
E @ 880
**********************
*INTEFFUFT 4 ROUTINE *
**********************
INT4 LI Fil2, >100
LI FiO, O
SEO 19
SEZ 17
TE: 22
JNE EA
MOV $F O, 0>1000$
SEO 17
TE 22
JEQ VA RTWF
END

DISAELE ALL INTEFRUF'TS
TMS 9961 ADDRESS
ENAELE ONLY INT4

ON TMS 9901
SEND COMMUNICATION WISH
ENAELE INT4 ON MICRO
WAIT INTEFFULTT
RETURN TO MONITOF

TMS 9901 ADDFESS
MESSAGE TO EE TRANSMITTED
CLEAF: WISH
HOLD CONTROLLER CONTFOLLER STOFFED?
NO! RETUFN TO EA
SEND MESSAGE
RELEASE CONTROLLER
RELEASED?
NO! FETURN TO UA
RETURN TO MAIN

## SYSTEM BUS SECOND TEST PROGRAM

CON1 TXMIFA $2.3 .0 \quad 7 日 .24400: 20: 46$ $01 / 01 / 00$ FAGE 0日01

E SECOND TEST ROUTINE 1

```
*****************************************
*THIS FROGRAM WAS WRITTEN FOR MODULE 1 *
*IT HELFS CHECKING THE SLAVE IDENTIFI- *
*CATION F'FOCESS. THE INTSLA LINE IS *
*SEFUED EY INT3 *
*****************************************
    IDT 'SECON1'
***************
*MAIN FROGRAM *
***************
*******MASTEF OFEFATION***
SECON1 LIMI O DISAELEE ALL INTERFUFTS
            LI F12, >100 TMS 9901 ADDFESS
            LI FO,G MESSAGE TO E:E TFANSMITTED
            SEZ 16 FESET FLIF-FLOF
            SEO 16
            MOU FII, O>2FFE DEFINE SLAVE(WFITE TO)
            SEZ 17 HOLD SLAVE
            TE 22
            JNE MA
            MOU FO, @>2OOO SEND MESSAGE
            SEO 17 RELEASE SLAvE
            TE 22 FELEASED
            JEQ TA NO!
```

*******SLAUE OFEFATION****
LI FQ, >0800 ENAELEE ONLY INT3
LDCF $\mathrm{FG}, 7$ ON TMS 9901
SETO @ $)$ FOOO
FLAG FDR MESSAGE
LIMI 3 ENAELE INTS ON MICRO
IDLE WAIT INTEFFUFT
E e80 FETURN TO MONITOR
**********************
*INTERFUFT 3 FOUTINE *
**********************
LI $\mathrm{K} 12,>100$
SEZ 16
TMS 9901 ADDRESS
RESET FLIF-FLOF
SEO 16
SEZ 18
$X A$ MOU @ $>F O O Q$, Fi
ENAELE HOLD IN
MESSAGE TRANSMITTED?
JNE XA NO!
SEO 18 DISAELE HOLD IN
FTWF FETUFN TO MAIN
END
CON2 TXMIFA $2.3 .0 \quad 78.24400: 21: 32$ 01/01/00 FAGE 0001

E SECOND TEST ROUTINE 2
0001
00022
0003
0004
0005
0006
0008
0009
0010
0011
0012
00130000
0300
00020000

00140004 020C 00060100 00080200 000A 0800 0016 000C 31C0 0017 OOOE 1E10 001.800101010 001900120300 00140003 00160340 0021 00220018 CBOC

OOIA 1FFE
0023001 C 0200
$001 E$ 0000
$002400201 E 11$ $002500221 F 16$ 00260024 16FE $00270026 \mathrm{C800}$ 00281000 0028 002A 1D11 0029 002C 1F16 0030 002E $13 F E$ 003100300460 00320080
0032 0033 0034 00350034 020C 00360100 003600381 E 10 0037 003A 1D10 3038 003C 1 E 12 0039 003E C060 0040 F000 3040004216 FD 004100441012 004200460380 0043

********************** *INTERFUFT 3 FRUUTINE * **********************

LI Fi2, 1100
SEZ 16
SEO 16
SEZ 18
MOV @ $\operatorname{MFODO,R1}$
JNE XA
SEO 18
RTWF.
END

TMS 9901 ADDEESS
RESET FLIF-FLOF
FLIF CLR
ENAELE HOLD IN
MESSAGE TRANSMITTED?
NO!
DISAELE HOLD IN
FETURN TO MAIN
FANS
$000 C 0000$
0024 DOOE 05C2
00250010 04E2
00120000
00260014 022E
00160004
00270018 C 01 E
0028 001A 130A
0029001 C 0980
0030001 E 0280
0020 002D
0031002216 F 0
00320024 C 38 E
00330026 C801
0028 0002
0034 002A CB 02
$002 \mathrm{C} 000{ }^{\circ}$
0035
0036
0037
0038
00390030 022E
0032 FFFC
0034 CODE
00360983
004200380603
0043 003A 1120
0044 003C 0223
$003 E$ FFF7
004500401510

TXMIFA
2. 3. 0 78. 244 00:22:43
$01 / 01 / 00$
FAGE 0001 WORD TRANSFER ROUTINE *******
*************************************************
*

* THIS ROUTINE WAS DESIGNED TO EE USED AS XOF $6 * *$
* IT FREFARES AND/OR UPDATES TAELES TO EE USED
* ON THE COMMUNICATION FROCESS. THE TAELES ARE
* SLAUES, MODES, SOURCES OF ADDRESS. WHEN IT IS
* FINISHED AND A GLOEAL MESSAGE END IS FOUND IT
* SENDS A COMMUNICATION WISH TO THE CONTROLLER
* THROUGH USE OF GELEND FOUTINE.
* 

*************************************************
IDT 'WTFANS'
DEF WTFANS
REF GELEND, ERFDR, SRCTEL, MODTELL, SLUTEL REF SFICCNT, MODCNT, SLUCNT
************************************************* * WORD TRANSFER ROUTINE *
*************************************************
WTRANS MOU @SFCCNT, RI GET SOURCE TAELE COUNTEF
INCFI INCT R1 INCFEMENT COUNTEF EY TWO-
MOU R11, @SRCTEL(Fi1) SAVE ADDFESS AT SFCTEL
MOU @MODCNT, F2 GET MODE COUNTEF
INCT FR
CLF MMODTEL(FR) WOFD MODE
AI RI1,4 SKIF TWO WORDS
MOU *R11,FD GET WORD
JEQ SNGEND
SRL RG, 8
CI RG, >2D
JNE INCRI
MOU R11, R14
MOU R1, 巴SFCCNT
MOU R2, @MODCNT
FTWP
**********************

* SINGLE MESSAGE END *
**********************
SNGEND AI R11,-4 EACK TWO WORDS
MOU *R11, R3
SRL R3, 8
DEC R3
JLT INTEF
AI R3, -9
JGT INTER

END OF SINGLE MESSAGE
MASK OFF
XOF INSTRUCTION?
NO!
CHANGE RETURN ADDRESS
save soufice counter

- MODE •

RETURN

GET WORD
MASK OFF
LESS THAN ONE?
INTEFNAL ADDFESS
MORE THAN TEN?

INTERNAL ADDEESS


0046 0047 0048 004900420223
$0044000 A$
0046 C 120 0048 0000
DOAA 日5C4 004C OAC3 004E 0223日050 0FFF 0054 0052 0903 00540000
$00550056 \quad 0501$
0056 0058 04E1
00SA 0008'
005C 022E:
005 E 0004
$00580060 \quad 15 E$
005900620285
00640001
$0060 \quad 00661602$
$00610068 \quad 0460$
$006 A$ 0000
0062
0063
0064
0065006 C C801
OOGE 日O28'
0070 C802
0072 002C'
0074 C804
0076 0048'
0078 C38E
007A 0380
0070
0071
0072
0073 007C 022E
007E ロロロ?
0074 0080 CODE:
007500820983
007600840603
007700861104
00780088 0223
Q日BA FFF7
0079 008C 1501
0080 008E 1009
0081
0082
0083
0084
0085

| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| :---: | :---: | :---: | :---: |
| ＊UPDATE SLAUE TAELE＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
|  |  |  |  |
| SLUNME： | AI | F3， 10 | RESTOFE |
|  | mov | ESLUCNT，Fi4 | get slave counter |
|  | INCT | R4 |  |
|  | SLA | F3， 12 | SET SLAVE ADDFESS |
|  | AI | F3， 7 EFFF |  |
|  | MOV | R3，©SLUTEL（F4） | save at slave taEle |
|  | INCT | Fil |  |
|  | CLF | ＠SFCTEL（R1） | END OF MESSAGE |
|  | AI | Fi1， 4 | SKIF TWO WORDS |
|  | － |  |  |
|  | mov | ＊ $\mathrm{F} 11, \mathrm{~F} 5$ | GET WORD |
|  | CI | F5，＞0001 | END OF GLOEAL MESSAGE？ |
|  | JNE | NOTEND | NO！ |
|  | E： | ＠GELEND | ERANCH TO GLOEAL END |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| ＊NOT A GLOEAFL END＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
|  |  |  |  |
| NOTEND | MロV | F1，＠SFiCCNT | SAVE SOUFCE COUNTEF： |
|  | MOV | R2，＠MODCNT | －MODE |
|  | MOV | F4，DSLUCNT | －slave |
|  | Mov | R11，Fil | CHANGE RETUFN ADDFESS |
|  | RTWF＇ |  | RETURN |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| ＊INTERNAL FEFEFENCE ONLY ？＊ |  |  |  |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| INTER | AI | F11，2 | SKIF WORD |
|  | MOV | ＊R11， 3 | GET WOFiD |
|  | SFL | Fi3， 8 | MASK OFF |
|  | DEC | R3 | LEESS THAN ONE |
|  | JL．T | ERR | ERROR！ |
|  | AI | Fi3，－9 | MORE THAN TEN |
|  | JGT | ERF＇ | EFKOR！ |
|  | JMF | SLUNME | CONTINUE |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| ＊ONLY INTEFNAL FEFEFENCES＊ |  |  |  |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| ERR | E | ＠ERROR | ERANCH TO EFFFOR |
| END |  |  |  |

RANS ***** 0001 0002 0003 0004 0005 0006 0007 0008 0009 0010 0012 0013 0014 0015 0016 0017 $00180000 \mathrm{C060}$ 00020000 0019000405 Cl 0020 0006 C84E 00080000 0021 0GOA COAO $000 C 0000$ 0022 DODE 05C2 002300100200 00120001 $00240614 \mathrm{CB80}$ 0016 0000 0018 022E 001 A 0006 0026 001C C01E 0027 001E 130A 002800200980 002900220280 0024 002D 0030002616 EE 00310028 C38E 3032 002A C801 002C 0002' 0033 002E C802 $0030000 C^{\prime}$ 003400320380

TXMIRA
2.3.0 78.244 00:24:03 01/01/00

F'AGE 0001 gLOCK TRANSFER ROUTINE *******
 *************************************************

IDT 'ETFANS'
DEF ETTFANS
FIEF GELEND, EFFIOR, SFICTEL, MODTEIL
REF SLUTEL, SRCCNT, MODCNT, SLUCNT
************************************************* * ELOCK TFiANSFER FOUTINE *
*************************************************

ETFANS MOU @SFCCNT, 下1
GET SOUFCE TAELE COUNTEF
INCFI INCT R1 INCFEMENT COUNTEF E:Y TWO
MOU Fi11, @SFCTEL(Fi) SAVE ADDFESS AT SFCTEL
MOU @MODCNT,FI GET MODE COUNTEF:
INCT K2
LI $\quad \mathrm{F} 0,1$
MOU FO, MMODTE:L KR2) SET
AI Fi1,6 SKIF 3 WOKDS
MOV *R11, FRO
JEQ SNGEND
SFL Fib, 8
CI FiO, >2D
JNE INCFi 1
MOU K11, R14
MOU R1, @SRCCNT
MOV F2, ©MODCNT
RTWF
**********************

* SINGLE MESSAGE END *
**********************
SNGEND AI $\mathrm{F} 11,-6$
MOU *R11, R3
SRL Fi3, 8
DEC R3
JLT INTEF
AI R3,-9
JGT INTEF

GET WORD
END OF SINGLE MESSAGE
MASK OFF
XOF INSTFUCTION?

NO!
CHANGE RETURN ADDRESS
SAVE SOUFICE COUNTEF
-MODE *
RETURN

EACK THREE WORDS
GET WORD
MASK OFF.
LESS THAN ONE?
INTERNAL ADDFESS
MOFE THAN TEN?
INTEFNAL ADDRESS

TXMIRA
2.3.0 78.244 00:24:03 01/01/00

FAGE 00日2 BLOCK TFANSFER FOUTINE *******
0070 C801
0072 002C
0074 C802
0076 0030
3066
0067
0068
0069
0070
0071
00720080 0225
00820004
00730084 CODE
007400860983
307500880603
0076 008A 1104
3077 008C 0223
008E FFF7
007800901501
007900921009
0080
081
0882
008300940460
0096 0000

| * UFDATE SLAVE TAELE * ********************** |  |  |  |
| :---: | :---: | :---: | :---: |
|  |  |  |  |
| SLUNME: | AI | F3, 10 | REStore |
|  | mov | @SLUCNT, F4 | get slave counter |
|  | INCT | K4 |  |
|  | SLA | F3, 12 | SEt Slave address |
|  | AI | R3, >0FFF |  |
|  | mov | R3, @SLUTEL(R4) | save at slave taele |
|  | INCT | F1 |  |
|  | CLR | @SFCTEL ( K 1 ) | END OF message |
|  | AI | F11, 4 | SKIF TWO WORDS |
|  | MOV | *R11, R5 | GET WORD |
|  | CI | R5, >0001 | END OF GLOEAL MESSAGE? |
|  | JNE | NOTEND | NO! |
|  | E: | @GELEND | ERANCH to gloeal end |
| ******************** |  |  |  |
| * NOT A GLOEAL END * |  |  |  |
| ******************** |  |  |  |
| NOTEND | MOV | F1, @SRCCNT | save source counter |
|  | MOV | R2, @MODCNT | - mode |
|  | mov | R4, @SLUCNT | - slave |
|  | mov FTWF | F11, R14 | CHANGE RETUFN ADDRESS RETUFN |
| *************************** |  |  |  |
| * Interinal reference only * |  |  |  |
| *************************** |  |  |  |
| INTER | AI | F11, 4 | SKIF TWO WORDS |
|  | MOV | *F11, 3 | GET WORD |
|  | SFL | R3, 8 | MASK OFF |
|  | DEC | R3 | LESS THAN ONE |
|  | JLT | ERF | ERROR! |
|  | AI | R3, -9 | MORE THAN TEN |
|  | JGT | ERR | ERFOR! |
|  | JMP | SLUNME | CONTINUE |
| **************************** |  |  |  |
| * ONLY INTERNAL REFERENCES * |  |  |  |
| **************************** |  |  |  |
| ERR | E: | @EFFROR | ERANCH TO ERROR |
| END |  |  |  |

**********************

* UFDATE SLAVE TAELE *
**********************
MOU ESLUCNT, R4
SLA K3, 12
AI R3, ADFFF
moU R3, @SLUTEL(R4) SAUE at sLave taEle
INCT FI
CLR @SRCTEL(R1) END OF MESSAGE
SKIF TWO WORDS
GET WOFD
END OF GLOEAL MESSAGE?
NO!
ERANCH TO GLOE:AL END
save sourice counter

CHANGE RETURN ADDRESS RETUFN

SKIF TWO WORDS
GET WORD MASK OFF
LESS. THAN ONE
ERROR!
MORE THAN TEN
ERFOR!
CONTINUE

## ****************************

* ONLY INTERNAL REFERENCES *
****************************
ERR E @EFROR ERANCH TO EFROR
END

0001 0002 0003 0004 0005

00040001
$00200006 \mathrm{C840}$
0008 0000
$0021000 A$ 020C
$000 C 0100$
*********************************************

* THIS ROUTINE IS USED EY WTFANS AND ETRANS *
* IT CLOSES THE GLOEAL MESSAGE TAELES AND * * ..... * ..... *
*********************************************
IDT 'GELEND'FEF SFETEL, WISHFG
*********************************************
* GLOEAAL END FOUTINE ..... *
*********************************************
GEILEND INCT Fil INCFEMENT COUNTER* CLOSING SOURCE TAELE (GEMEND) **********************************
LI $F \mathfrak{O},>0001$
MOU F'O, @SFCTEL(Fi1) GLOEAL MESSAGE END
LI FiL2, 1000901 EASE
*******************************
* SET COMMUNICATION WISH FLAG *CLR @WISHFG SET WISH FLAG
LIMI 4 ENAELE INT 4
***************************
* INTEFRUFTING CONTROLLER *
***************************
SEZ 19 INTEFFUFT CONTFOLLEF
RTWF
END

FAGE 0001



## APPENDIX 7.7.3

ACKNOWLEDGE MASTER ROUTINE

OODA 0000
000C 05C1
OQDE COA1
00100000
0012 1612
$001 C 0000$
0035 001E 1309
0036 0020 C133
0037 0022 C153
0038 0024 C554
20390026 05C2
00400028 C0E2
002A 001C'
日02C 1302
OO2E 0642
0043 0030 10ED
0044 0032 C802
0034 0016
00361012
0042


## *

* 
* 

TXMIRA
2.3.0 78.244 00:27:10
$01 / 01 / 00$
F'AGE 0002 ***** ACKNOWLEDGE MASTEF fOUTINE $* * * * * * *$

0049 0050 6051
00520038 COAD
0日3A 0034'
0053 003C 05c2
OD5 4 OQ3E COE2
0040 002A'
00550042 13F7
$00560044 \quad C 133$
00570046 C1E3
$00580048 \quad \mathrm{C} 153$
0059 004A CD74
$0060004 C 0606$
0061 004E 16FD
$0062005005 C 2$
0063 0052 COE?
00540040
0064005616 DA
$00650058 \quad 0642$
0066 005A 1 0EE
0067
0068
0069
0070
0071 005C 0801
OOSE OODA'
0060 COAO
0062 003A'
00730064 05c2
0074 0066 COE?
0068 0054'
006A 1311
0076
0077
0078
0079
00800060 020C
006E 0100
0081 0070 1011
008200720642
00830074 C060
00760000
0078 C802
007A 0062'
0085 OOTC COA1
007E 0000
$00860080 \quad 0712$
0087 0082 05c1
00880084 COA1
0086 007E'
$00890088 \quad 0712$
0090 00BA 1E11
0091 日08C 0380


* ELOCK MODE *
**************
EMODE MOU @SFCCT1,R2 SOUFCE COUNTER
INCT R2
MOU @SFCTELL(F2), FB GET WOFD
JEQ SNGEND END OF MESSAGE
SOUFEE
NUMEEF OF WOFDS
DESTINATION
SEND MESSAGE
FINISHED?
NO!
ADDFESS SOUFICE
END
FESTOFE
TEST GLOE:AL END
SAVE MODE COUNTER
SOURCE COUNTEF
MESSAGE
GLOEAL END
SLAVE
FESTORE COUNTER
SLAVE COUNTER
SAVE COUNTER
SLAVE ID
SIGNAL END OF MESSAGE
NEXT SLAVE ADDRESS
INTEFFUPT SLAVE
HOLD SLAVE
RETUFN


| 0092 |  |  |
| :--- | :--- | :--- |
| 0093 |  |  |
| 0094 |  |  |
| 0095 | $008 E$ | $04 E 0$ |
|  | 0090 | $007 A^{\prime}$ |
| 0096 | 0092 | $04 E 0$ |
|  | 0094 | $005 E$ |
| 0097 | 0096 | $04 E 0$ |
|  | 0098 | 0076 |
| 0098 | $009 A$ | $020 C$ |
|  | $009 C$ | 0100 |
| 0099 | $009 E$ | 0720 |
|  | $00 A 0$ | 0000 |
| 0100 | $00 A 2$ | 1015 |
| 0101 | $00 A 4$ | 0380 | 0101 00A4 0380 0102

## ***************

* FEELEASE EUSS *
***************
FELESE CLF @SRCCT1 CLEAF COUNTEFS
CLK @MODCT 1

CLF eSLUCT1
LI $\mathrm{Fi} 12,7100 \quad 9901 \mathrm{EA} \mathrm{ASE}$
SETO @CTL SIGNAL CONTFOLLEF(END)
SE:O 21 FESET -EUS EUUSY LINE
FTWF FETUFN

TXMIFA
2. 3. 0 78.244 00: 28:48
$01 / 01 / 00$
FAGE 0001 INTERFUFT SLAUE ROUTINE *******
000 A 0000
0023 BOOC 130C
0024 OOOE C020
00100000
00121304
0026
0027
0028
0029
0030
0031
00320014 O4E0
$00160010^{\circ}$
0033
0034
0035
0036
0037
0038 001C 0720
$001 E 0016^{\prime}$
0020 06A0
00220000
00240380
0041
0042
0043
00440026 C 020
0028 001E
g62A 1308
0046
0047
0048
*************************************************
*

* THIS FOUTINE IS USED EiY A SLAVE OF CONTROLLER *
* MODULE TO ACCEFT AND CHECK MESSAGES TRANSMIT- *
* TED TO THEM. IT IS A TWO FASS FOUTINE. ON THE *
* FIRST TIME IT SETS A FLAG TO INDICATE A *
* MASTEF'S WISH TO COMMUNICATE. ON THE SECOND *
* FASS THE MESSAGE IS ALFEADY FINISHED AND CAN *
* EE CHECKED, IT IS DESIGNED TO EE INTERRUFT *
* DRIUEN.
* 

*************************************************
IDT 'INTSLU'
DEF INTSLU
FEF MSGFLG, IDMESL, CTLFLG, IDMECT
*************************************************

* INTEFFUFT SLAUE FOUTINE *
*************************************************
INTSLULI Fi12, >10日 9901 EASE
SEO 18 DISAELLE HOLD IN
SEZ 16 CLEAF FLIF-FLOF
MOV ECTLFLG,FO CONTFOLLEF MODULE?
JEQ CONTLF YES!
MOU EMSGFLG, FO END OF MESSAGE?
JEQ MESEND YES!
******************
* SLAUE ACTION *
****************
* 
* STAFT OF mESSAGE
* 

CLR EMSGFLG

SE:Z 18
RTWF RETURN
SET FLAG FOR MESSAGE
ENAELE HOLD IN
*

* END OF MESSAGE
* 

MESEND SETO @MSGFLG
EL @IDMESL
RESET FLAG
IDENTIFY AND SEFVE

RETURN
*********************

* CONTFROLLEF ACTION * *********************
CONTLF MOU @MSGFLG, FG
JEQ EUSFRE
* 
* EUS IS EEEN USED
* 



| 0049 | 002 C | 1 E 00 |  | SE:Z | 0 | DISAELE INTS ON 9901 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 0050 | 002E | 1E05 |  | SER | 5 |  |
| 0051 | 0030 | 04E0 |  | CLR | CMSGFLG | SEt flag |
|  | 0032 | 0028' |  |  |  |  |
| 0052 | 0034 | 06AO |  | EL | ©IDMECT | IdENTIFY AND SERUE |
|  | 0036 | 0000 |  |  |  |  |
| 0053 | 0038 | 0720 |  | SETO | @MSGFLG | feset flag |
|  | 003A | 0032' |  |  |  |  |
| 0054 |  |  | * |  |  |  |
| 0055 |  |  | * EUS | IS FRE |  |  |
| 0056 |  |  | * |  |  |  |
| 0057 | $003 C$ | 0720 | EUSFFE | SETO | @CTLFLG | RESET CONTROLLEF FLAG |
|  | 003E | 000A' |  |  |  |  |
| 0058 | 0040 | 1E0日 |  | SE:Z | 0 | INTERRUPT MODE |
| 0059 | 0042 | 1005 |  | SEO | 5 | ENAELE INTERRUFT 5 |
| 0060 | 0044 | 0300 |  | LIMI | 5 |  |
|  | 0046 | 0005 |  |  |  |  |
| 0061 | 0048 | 0380 |  | RTWF |  | FETURN |
| 0062 |  |  |  | END |  |  |

## APPENDIX 7.7.5

## INTERRUPT CONTROLLER ROUTINE

TXMIFA
2.3.0 78.244 00:29:45
$01 / 01 / 00$
FAGE 0001


APPENDIX 8.2.1

## SERIES 76 (E076-102) MOOG SERVOVALVE

The rated flow for the available valve is $191 / \min$ ( 5 gpm ) at 70 bar ( 1000 psi ). The flows for other supply pressures are given in Figure 1.


Figure 1 Change in Rated Flow with Pressure

The flow gain nonlinearity is most severe in the null region due to variations in the spool null cut. Within $\pm 5 \%$ of rated current input the valve flow gain may range from 50 to $200 \%$ of the normal flow gain. The rated flow tolerance is $\pm 10 \%$.

The pressure gain at null exceeds $30 \%$ of supply pressure for $1 \%$ of rated current and can be as high as $80 \%$. The expected variation in other parameters relating to valve performance are given below.

|  | Parameter | Variation (\%RC) |
| :---: | :---: | :---: |
| Symmetry |  | < 10\% |
| Hysteresis |  | $<3 \%$ |
| Threshold |  | < $\frac{1}{2} \%$ |
| 易 | Temperature | $< \pm 4 \%$ |
|  | Acceleration | $< \pm 2 \%$ |
|  | Supply Pressure | $< \pm 2 \%$ |
|  | Quiescent Current | $< \pm 2 \%$ |
|  | Back Pressure | $< \pm 2 \%$ |

* RC: Rated Current

The electrical characteristics for the valve are as follows:

| Nominal <br> Resistance <br> per Coil <br> at $20^{\circ} \mathrm{C}$ <br> $\Omega$ | Recommended <br> Rated Current <br> (mA) |  | DF,PL,SC | Series <br> Coil | SC |  |  |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 200 | 15 | 10 | 0.72 | Approximate Coil <br> Inductance - (Henrys) |  |  |
|  |  | 1.1 | 2.2 | 0.59 |  |  |  |

DF - Differential Coils
PL - Parallel Coils
SC - Single Coil

The frequency response for the series 76 servovalves is shown in Figure

The variation in response with supply pressure, as expressed by the change in frequency of the $90^{\circ}$ phase point, is given in Figure 2.

A typical transient step response for the series 76 servovalves is given in Figure 2. The straight-1ine portion of the response represents saturation flow from the pilot stage which will increase with higher supply pressures.


FIGURE 2

## SPECIFICATIONS FOR I/O ANALOG BOARD



APPENDIX 8.4.2

ANALOG I/O BOARD AND MICROCOMPUTER INTERCONNECTIONS

| ANALOG I/O AND MICROCOMPUTER INTERCONNECT |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Function | Analog | Micro(P1) | Function | Ana1og | Micro(P1) |
| D0 | 33 | 33 | A0 | 57 | 57 |
| D1 | 34 | 34 | A1 | 58 | 58 |
| D2 | 35 | 35 | A2 | 59 | 59 |
| D3 | 36 | 36 | A3 | 60 | 60 |
| D4 | 37 | 37 | A4 | 61 | 61 |
| D5 | 38 | 38 | A5 | 62 | 62 |
| D6 | 39 | 39 | A6 | 63 | 63 |
| D7 | 40 | 40 | A7 | 64 | 64 |
| D8 | 41 | 41 | A8 | 65 | 65 |
| D9 | 42 | 42 | A9 | 66 | 66 |
| D10 | 43 | 43 | A10 | 67 | 67 |
| D11 | 44 | 44 | A11 | 68 | 68 |
| D12 | 45 | 45 | A12 | 69 | 69 |
| D13 | 46 | 46 | A13 | 70 | 70 |
| D14 | 47 | 47 | A14 | 71 | 71 |
| D15 | 48 | 48 |  |  |  |
|  |  |  |  |  |  |
| WE | 78 | 78 | $\overline{\text { MEMCYC }}$ | 84 | 84 |
| DBIN | 82 | 82 | $\overline{\text { IORST }}$ | 88 | 88 |
| READY | 90 | 90 |  |  |  |

## APPENDIX 8.4.3

## ANALOG I/O BOARD AND RIBBON CABLE <br> INTERCONNECTIONS

| ANALOG I/O BOARD |  | CABLE | ANALOG I/O BOARD |  | CABLE |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Function | Pin (P3) | Colour | Function | Pin (P3) | Colour |
| СНО HI | 5 | white | CHO LO | 4 | white |
| CH1 HI | 7 | grey | CH1 LO | 8 | grey |
| CH2 HI | 11 | violet | CH2 LO | 10 | violet |
| CH3 HI | 13 | blue | CH3 LO | 14 | blue |
| CH4 HI | 17 | green | CH4 L0 | 16 | green |
| CH5 HI | 19 | yellow | CH5 L0 | 20 | yellow |
| CH6 HI | 23. | orange | CH6 LO | 22 | orange |
| CH7 HI | 25 | red | CH7 LO | 26 | red |
| CH8 HI | 29 | brown | CH8 L0 | 28 | brown |
| CH9 HI | 31 | black | CH9 LO | 32 | black |

(measuring system)

$\underset{\omega}{\underset{\omega}{\oplus}}$

APPENDIX 8.5.2

DIGITIZER FEEDBACK BOARD COMPONENTS LIST

| DIGITIZER FEEDBACK BOARD COMPONENTS |  |  |
| :---: | :---: | :---: |
| ID | Device | Description |
| 1 | SN74LS95 | 4-bit shift register |
| 2 | SN74LS42 | 4-1ine-to-10 line decoder |
| 3 | SN74LS42 | " |
| 4 | DM74LS 193 | Synchronous up/down counter |
| 5 | DM74LS193 |  |
| 6 | DM74LS193 | " |
| 7 | DM74LS 193 | " |
| 8 | SN74LS374 | Octal D-type flip-flop |
| 9 | SN74LS374 | Octal D-type (3-state output) |
| 10 | DM74154 | 4-line-to-16 line decoder |
| 11 | SN74LS245 | Octal bus transceiver (3-state) |
| 12 | SN74LS30 | 8 -input NAND gate |
| 13 | SN74LS32 | Quadruple 2-input OR gate |
| 14 | SN74LS10 | Triple 3-input NAND gate |
| 15 | SN74LS00 | Quadruple 2-input NAND |
| 16 | SN74LS08 | Quadruple 2-input AND |
| 17. | SN74LS08 | " |
| 18 | SN74LS14 | Nex Schmitt-Trugger inverter |
| 19 | SN74LS14 | " |
| 20 | SN74LS74 | Dual. D-type f1ip-flop |
| 21 | SN74LS74 |  |
| 22 | SN74LS74 | " |
| 23 | SN74LS02 | Quadruple 2-input NOR gate |
| 24 | SN75182 | Dual differential line receiver |
| 25 | SN75182 | " |
| - | C1-C2-C3 | Polyester capacitor $0.01 \mu \mathrm{~F}$ |
| - | C4-C5-C6 | Ceramic capacitor 100 pF |

APPENDIX 8.5.3

DIGITIZER FEEDBACK BOARD PIN-FUNCTION ASSIGMENT

| DIGITIZER FEEDBACK BOARD |  |  |  |
| :---: | :---: | :---: | :---: |
| Function | Pin | Function | Pin |
| D0 | 20 | AO | 68 |
| D1 | 22 | Al | 66 |
| D2 | 26 | A2 | 64 |
| D3 | 24 | A3 | 62 |
| D4 | 36 | A4 | 58 |
| D5 | 38 | A5 | 56 |
| D6 | 42 | A6 | 54 |
| D7 | 40 | A7 | 52 |
| D8 | 34 | A8 | 50 |
| D9 | 32 | A9 | 48 |
| D10 | 28 | A10 | 46 |
| D11 | 30 | All | 44 |
| D12 | 10 | Sample | 60 |
| D13 | 8 | DBIN | 70 |
| D14 | 12 | $\overline{\text { MEMEN }}$ | 72 |
| D15 | 14 | $\overline{\text { ¢ }} 1$ | 76 |
| GND | 1 | ¢ ${ }^{\text {3 }}$ | 71 |
| GND | 85 | Ua1 | 77 |
| +5V | 3 | $\overline{\text { Ua1 }}$ | 78 |
| MIDDLE | 69 | Ua2 | 79 |
|  |  | $\overline{\mathrm{Ua} 2}$ | 80 |
|  |  | Ua3 | 81 |
|  |  | Ua3 | 82 |

## DIGITIZER FEEDBACK BOARD AND MICROCOMPUTER INTERCONNECTIONS

| PIN INTERCONNECTION <br> Function |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Databus <br> Board | Digitizer <br> Board | Function | Address <br> Bus Board | Digitizer <br> Board |  |
| D0 | 49 | 20 | AO | 68 | 86 |
| D1 | 47 | 22 | A1 | 66 | 84 |
| D2 | 45 | 26 | A2 | 64 | 82 |
| D3 | 43 | 24 | A3 | 62 | 80 |
| D4 | 41 | 34 | A4 | 58 | 78 |
| D5 | 39 | 32 | A5 | 56 | 76 |
| D6 | 37 | 28 | A6 | 54 | 72 |
| D7 | 35 | 30 | A7 | 52 | 70 |
| D8 | 67 | 36 | A8 | 50 | 68 |
| D9 | 65 | 38 | A9 | 48 | 66 |
| D10 | 63 | 42 | A10 | 46 | 64 |
| D11 | 61 | 40 | A11 | 44 | 62 |
| D12 | 59 | 10 | Function | Micro | Digitizer |
| Board |  |  |  |  |  |
| D13 | 57 | 8 | $\overline{\phi 1}$ | 22 (P1) | 76 |
| D14 | 55 | 12 | $\overline{\phi 3}$ | 24 (P1) | 71 |
| D15 | 53 | 14 | SAMPLE | 24 (P4) | 60 |
| DBIN | 30 | 70 | MIDDLE | 13 (P1) | 69 |
| MEMEN | 28 | 72 |  |  |  |

## EXE UNIT AND DIGITIZER FEEDBACK BOARD

INTERCONNECTIONS

| DIGITIZER <br> FEEDBACK <br> BOARD |  | 15-WAY SOCKET (RACK) | SCREENED CABLE WITH 15-WAY PLUG | $\begin{aligned} & \text { EXE UNIT } \\ & \text { 12-WAY } \\ & \text { SOCKET } \end{aligned}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Function | Pin | Pin | Colour | Function | Pin |
| $\begin{aligned} & \frac{\mathrm{Ua} 1}{\overline{\mathrm{Ua} 1}} \\ & \mathrm{Ua} 2 \\ & \overline{\mathrm{Ua} 2} \\ & \mathrm{Ua} 3 \\ & \overline{\mathrm{Ua} 3} \\ & \text { chassi } \end{aligned}$ | $\begin{aligned} & 77 \\ & 78 \\ & 79 \\ & 80 \\ & 81 \\ & 82 \\ & - \end{aligned}$ | $\begin{aligned} & 1 \\ & 2 \\ & 3 \\ & 4 \\ & 5 \\ & 6 \\ & 7 \end{aligned}$ | brown <br> green <br> grey <br> pink <br> orange <br> black <br> yellow | $\begin{aligned} & \frac{\text { Ual }}{} \\ & \hline \text { Ua1 } \\ & \text { Ua2 } \\ & \hline \text { Ua2 } \\ & \text { Uao } \\ & \hline \text { Uao } \end{aligned}$ screen | 6 8 1 3 4 9 |
| +5V | 3 | 9-10-11-12 | white-red <br> white-blue <br> white <br> pale blue | +5V | 12 |
| GND | 1 | 13-14-15 | red <br> red-brown <br> red-black | OV | 10 |

TXMIFA 2.3.0 78.244 00:32:29
$01 / 01 / 00$
F'AGE 0001 NERAL CLOSING THE LOOF ROUTINE
***********************

* FEEAD FEEDEACK UALUE *
***********************

*********
* START *
*********
LOOFIN LIMI 0
LWPI WOFKK1
LI R12, $>100$
****************************
* NEW SAMFLING FREQUENCY ? *
****************************
MOV @NEWFRE, R1 NEW SAMFLING?
JNE MASKIN NO!
LDCR @MASKO, 7 FROTECTS 9900
LDCR @PERIOD, 15
SETO @NEWFFE
MASKIN LDCR @MASK 1,7
MASK INTERRUFTS
LOAD FOINTER
9901 EASE

LOAD CLOCK
RESET FLAG
ENAELE INT 9901

SE:Z 23


004800241017 004900260300 00280002 0050 002A C820 002C 0000 $002 E$ 0000

0051 0052 0053 0054
$0030 \mathrm{C060}$ 0032 0000 005500341604 0056 0036 06A0 0038 0000 0057 003A 0460 003C 016A'
0058
0059
0060
0061
$003 E$ C060
00400000
006200421604
00630044 06A0
$00460038^{\prime}$
006400480460
004A 018E'
0065
0066
0067
0068 004C COAO
OBAE OODO
0069
0070
0071
0050 60AB
0052 002E'
00541903
0073
0074
0075
0076
005604 E 0
0058 0040
0077 00SA 10F4
0078 005C A802
OOSE OOOD
0079
0080
0081
0082 0BGO COAD
0062 005E'
$00830064 \quad 0742$
$00840066 \quad 0282$
00680000
$0085006 A 1502$
0086 006C 04E0

SEO 23
LIMI 2 ENAELE HIGHEK INT.
MOU RDPAS, RDXK READ FEED. INCFEM.

```
*************************
* EMERGENCY CONDITION ? *
*************************
    MOU @EMFLAG,F1
    JNE FECHEK
    EL @EMDECE
    E @UF'DAFO
```

******************************

* FEEDE:ACK EFFOR CONDITION ? *
******************************
FECHEK MOU eFEFLAG, Fi FEEDEACK EFFROR?
JNE UFDEFRO NO!
EMER EL ©EMDECE GO TO DECELE. FOUTINE
E: @FETUFN


## ***************************

* UFDATING FOSITION EFFROR * ***************************
UFDERO MOU QDCFK, F2
*FOF FOSITION OFEN LOOF INSEFT 1 ORO ON
*NEXT 6 MEMORY LOCATIONS
5 ODXK, FI INCF, ON FOS. EFFKOF
JNO UFEFRO
****************** * FEEDEACK EFFROF * ******************
SETFE CLF @FEFLAG

JMF EMEF
UFEFFO A F'ב, @EFRK
*******************

* EFFFOF EANDWIDTH * *******************

MOU @EFFK, RI2 EFFKOR
AES FI2 AESOLUTE
CI $\mathrm{K} 2,70000$ OO MICFONS
JGT CALCUL
CLR @ERKK

SET FEEDEACK ERKOK FLAG

GO TO DECELE, FOUTINE UF'DATE EFFFOF

CLEAF ERROR

TXMIFA
2．3．0 78．24400：32：29 01／01／00
FAGE 0003 NERAL CLOSING THE LOOF FOUTINE

006E 0062＇
0087 0088 0089 0090 0091 $00920070 \mathrm{C060}$

0072 0000
00930074 A060
$00760052^{\prime}$
009400780821
$0095007 A$ C801
$007 C 0000$
0096 007E C820
$00800076^{\prime}$
$0082007 C^{\prime}$
0097 0098 0099 0100 0101 0102 0103 0104

0084 C060 0086 9082＇
010500886060 008A 0000 $008 \mathrm{C801}$ 0日8E 0000

0107
0108 0109 0110 0111 0112

|  |  |
| :--- | :--- |
|  |  |
|  |  |
|  |  |
| 0084 | 0060 |
| 0086 | 0082 |
| 0088 | 6060 |
| $008 A$ | 0000 |
| $008 C$ | 2801 |
| $008 E$ | 0000 |

0121
0122
0123
0124 DOAB OGAD

| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| :---: | :---: | :---: | :---: |
| ＊ESTIM | MATED | SPEED＊ |  |
| ＊SFEED | ／WN | XK－XK1）／（1＊T＊WN）＊ | ＊ |
| ＊（WN＊T | ）$=1$ | ＊ | ＊ |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| CAL．CUL | MOV | ＠ХイMXK4，R1 | UFDATE DISTANCE INCFE |
|  | A | ＠DXK，R1 |  |
|  | SKA | F1， 2 | DIUIDE EY 4 |
|  | MOV | R1，mu01K | SAVE SFFED |
|  | MOV | ＠DXK，＠UO1K | SAUE SFEED（ 15TEF AVEK） |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| ＊DIFFERENTIAL ESTIMATION＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
|  |  |  |  |
| ＊ACCELEFATION <br> ＊A／WN＊＊2＝（V01K－V01K1）／（WN＊T） <br> ＊（WN＊T）$=1$ |  |  | ＊ |
|  |  |  | ＊ |
|  |  |  | ＊ |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
|  | MOV | ＠ソ01K，下ı | GET ACTUAL SFEED |
|  | 5 | ＠UO1K1，下1 | SUETFACT LAST SFEED |
|  | MOV | Fi，©AO1K | SAVE ACCELEFATION |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| ＊TFAANSIENT |  | ACCELERATION | ＊ |
| ＊DA／WN＊＊3＝（VO1K－2＊VOIK1＋U01K2） |  |  | ）／（WN＊T）＊＊2＊ |
| $*(W N * T)=1$ |  |  | ＊ |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| MOV |  | ＠U01K，Ri1 | GET ACTUAL SF＇EED |
| A |  | ＠UO1K2，K1 | ADD LAST－1 SFEED |
| 5 |  | ＠VO1K1，Fi | SUE：TFACT LAST SFEED |
|  | S | ＠U01K1，Fi | TWICE |
|  | MOV | Fi，©DAOIK | SAUE TFiANSIENT ACCELE． |
| ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
| ＊OESEFVED UALUES＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ |  |  |  |
|  |  |  |  |
| ELL |  | ＠DOES1 | CALCUL．OESERV．VALUES |

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ ＊CONTFOL FUNCTION＊
$* * * * * * * * * * * * * * * * * * * *$
EL eDUCONT
$* * * * * * * * * * * * * * * * * * * * ~$
EL CDUCONT

CALCUL．CONTROL FUNCTION OOAA EOGO


OFIN TXMIFA $2.3 .0 \quad 78.24400: 32: 29 \quad 01 / 01 / 00 \quad$ FAGE 0005 NERAL CLOSING THE LOOF FOUTINE

OOFE 0000
$01660100 \mathrm{C801}$ 0102 0000
0167
0168
0169
01700104 06A0 01060000
0171
0172
0173
01740108 c 820
010A ODOD
010 COOD
$0175010 \mathrm{CB20}$
0110 ODFE＇
0112 D10A＇
$01760114 \mathrm{C820}$
0116 0日FA＇
$01180110^{\prime}$
0177
0178 011A C820
011C 0000
$011 E 0000$
$01790120 \quad 0820$
01220000
$0124011 C^{\prime}$
$0180 \quad 0126 \quad 6820$
$01280080^{\prime}$
012A 0122＇
0181 012C 04E6
$012 \mathrm{E} 0072^{\prime}$
01820130 A820
0132 012A＇
0134 012E
01830136 A820
$01380124^{\prime}$
013A 0134＇
0184013 C A820
013E 011E＇
$0140013 A^{\prime}$
0185
$01860142 \mathrm{C820}$
0144 009E＇
$01460096^{\prime}$
01870148 C820
$014 \mathrm{~A} 0092^{\prime}$
$014 \mathrm{C} 0144^{\prime}$
0188
0189014 E C820
01500000
01520000
$01900154 \mathrm{C820}$
01560000

MOU Fi，eDUK
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊OESERUER FUNCTION＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
EL セDOBS2
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊UPDATING FUNCTIONS＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
MOU＠UK2，叩UKろ

MOU＠UK1，＠UK2

MOV ©UK，セUK1
＊FOF SFEED
MOU＠DXK2，＠DXKЗ

MOU＠DXK1，＠DXK2

MOU＠DXK，＠DXK1

CLE＠
A＠DXK1，＠XKMXK4

A 巴DXK2，巴 $\times K M \times K 4$

A＠ロXK3，＠XKM×K4
＊FOF DIFFEFENTIAL ESTIMATION
MOU＠UO1K1，＠UOIK2

MOV eVO1K，eVOIK1 U（K）TO U（K－1）

SAVE RESULT

SOLVE OESERVER EQUAT．
$U(K-2)$ T0 U（K－3）

U（K－1）TOU（K－2）

U（K）TO U（K－1）

V（K－1）TO V K－2）
＊FOF OESEFUEF
MOV＠W11KF，©W11K

MOU＠W12KF，＠W12K
01580000

0191 015A C820 $015 C 0000$ 015 E 000
$01920160 \mathrm{C820}$ 01620000 0164 0000 0168 004E＇

0195
0196
0197
0198016 C C120
$016 \mathrm{C} 0128^{\prime}$
0199016 E 1102 $0200017004 C 3$ 020101721001 $02020174 \cdot 0703$
02030176 COAD 0178 0000
0204 017A C060 017C 0000
$0205017 E$ 06A0 01800000
020601820466 $01840056^{\prime}$
$02070186 \quad \mathrm{C} 801$ $0188017 C^{\prime}$ 0208018 A C 002 $018 \mathrm{C} 0178^{\prime}$

0209
0210
0211
0212018 E 02E0 0190 FF8A 021301920380 0214

MロU＠W21KF，巴W21K

MOU＠W22KF，＠W22K
＊FOF SAFETY
CLR 巴DCFK CLEAR COMMAND INCKEMENT
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊UFDATING AESSOLUTE FOSITION＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
UFDAFO MOU＠DXK，R4 FOSITION INCR．
JLT NEDXK MAKE IT DOUELE FRECISION
CLR Fi3 POSITIUE
JMF UF＇DA1
NEDXK SETO R3 NEGATIUE
UFDA1 MOU＠ACFOS2，K2 LATE FOSITION（LEAST）
MOU＠ACFOS1，FI（MOST）
EL ODEADD UFDATE FGSITION

E＠SETFE THEFE IS DUEFFLOW！
MOU Fi ，＠ACFOS1 ．SAVE FOSITION（MOST）
MOU F2，＠ACFOS2（LEAST）
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊INT 3 EFFOM AFEA WF＝$=>F F 8 A$＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
KETUFN LWFI＞FFBA
FTWF＇
END

TXMIFA
2．3．0 78．244 00：32：29
$01 / 01 / 00$
F＇AGE
0007 NERAL CLOSING THE LOOF ROUTINE

| $E$ | ACPOS1 | 0188 | $E$ | ACFOS2 | 018 C | E | A01K | 008E | ， | CALCUL | 0070 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| E | DAC1 | OEF6 | E | DAO1K | 00A2 | E | DEADD | 0180 | E | DCFK | 0168 |
| E | DOES 1 | 00A6 | E | DOES2 | 0106 | $E$ | DPOS | 002 C | E | DUCONT | D日AA |
| E | DUK | 0102 | E | DXK | 016 C | E | DXK1 | 0132 | E | DXK2 | 0138 |
| E | DXK3 | 013 E | E | EMDECE | 0046 | ， | EMEF | 0044 | E | EMFLAG | 0032 |
| E | EFRK | 006E | ， | FECHEK | 003 E | E | FEFLAG | 0058 | E | KCOFR1 | DOE： 6 |
| $E$ | KCORF2 | OQEA | E | KUCOR1 | OOEO | E | KUCOF2 | OEE4 | D | LOOFIN | 0000 |
| E | MASK0 | 0014 | E | MASK1 | 0020 | ， | MASKIN | 001 E | ， | NEDXK | 0174 |
|  | NEG12 | O日DC | $E$ | NEWFRE | 001 C | ， | N011 | OODA | ， | NO12 | OOEC |
| ， | NOCOFR | $00 \mathrm{C2}$ | E | PERIOD | 0018 |  | F＇0 | 0000 |  | F1 | 0001 |
|  | Filo | O日EA |  | R11 | OGOE |  | F12 | 000c |  | F13 | 000D |
|  | R14 | OOOE |  | R15 | 000F |  | F2 | 0002 |  | F3 | 0003 |
|  | F4 | 0004 |  | Fi5 | 0005 |  | R6 | 0006 |  | R7 | 0007 |
|  | Fi8 | 0008 |  | F9 | 0009 | ， | FETUFN | 018 E | ， | SETFE | 0056 |
|  | ST12 | OQEE | E | UK | 0116 | E | UK1 | 0118 | E | UK2 | 0112 |
| E | UK3 | 010 C | E | UKCOFR | 00c8 | E | UKOUT | OOF4 | ， | UF＇DA1 | 0176 |
| ， | UFDAFO | 016 A | ， | UFDERO | 004C | ， | UF＇ERFRO | 005c | $E$ | VO1K | 014 A |
| E | VOIK1 | 014 C | E | VO1K2 | 0146 | $E$ | W11K | 0152 | E | W11 KF | 0150 |
| $E$ | W12K | 0158 | E | W12KF | 0156 | E | W21K | $015 E$ | E | W21kF | $015 C$ |
| E | W22K | 0164 | E | W22KF | 0162 | E | WOFK1 | 0006 | E | XKMXK4 | 0140 |

OQD ERFORS



ERGENCY DECELERATION ROUTINE
0014 000A'
003000161701
0031001804 Cl
0032
0033
0034
$0035001 A \mathrm{CBO1}$
$001 C 0000$
0036
0037
0038
0039 001E C801
00200002
0022045 E
0041

```
*************************************************
*
                                *
    * THIS FOUTINE DECELEFRATES THE CONTFOL FUNCTION *
    * TO ZEFO WITH A FATE GIVEN EY UALUE ON *
    * LOCATION 'DEMU'. IT SENDS THE CONTROL OUT *
    * TO DAC1. *
    * *
*************************************************
    IDT 'EMDECE'
    REF UK, DEMU, DACI
    DEF EMDECE
******************
* CONTFOL NULL ? *
******************
EMDECE MDV @UK,F1
            JEQ RETURN
            JLT ADDIT
************
* FOSITIVE *
************
            S @DEMU,FI SUETFACT INCFEM.
            JNC OUT
            CLFF FI
            JMF QUT
************
* NEGATIUE *
************
ADDIT A @DEMU,F1 ADD INCREM.
    JNC OUT
    CLF Fi1
********************
* SEND COMMAND QUT *
********************
OUT MOU F1, EDAC1
****************
* SAVE COMMAND *
****************
    MOU R1, ©UK
FETUKN E *R11
    END
```

CONTFOL VALUE

CONTFOL NULL NEGATIUE

SUETFACT INCFEM.

ADD INCREM.

CAFIFY?
NULLL

SEND COMMAND

SAVE COMMAND

RETURN

## APPENDIX 9.3

## RAMP GENERATION ROUTINE

```
MF1
```

MFI
TXMIFA
2.3.0 78. 244 00:37:42
$01 / 01 / 00$
F'AGE 0001 MFI FUNCTION COMMAND ROUTINE

0001 0002 0003 0004 0005 0006 0007 0008 0010 0011 0012 0013 0014 0015 0016 0017 0018 0019 0020 0021 0022 0023 $002400002 F A 0$ 0002 022C'
00250004 2FA0
0006 0230
00260008 04C2 0027 000A 04C4 0028 000C 04c6 0029 OODE 04C8 00300010 04CA
$0031001206 A 0$
00140000
$0032001610 F 4$ 003300181006 0034 001A C 084 0035 001C C106 0036 001E C188 00370020 C20A 00380022 C 281 00390024 10F6
0040
0041 0042 0043 0026 38A0 0028 0000
0044 002A 3920
$002 C$ 0000
0045 002E 39A0 00300000
00460032 3A20 0034 0000 00470036 A24A
****************************************************

* RAMF ** THIS ROUTINE GENERATES A RAMF FOSITION
* INFUT (CTE UELOCITY). IT IS AN INTERACTIVE ROUTINE
* WHERE UARIARLES SUCH AS UELOCITY AND DISTANCE TO
* MOUE ARE ASKED FOR. FOR TRANSIENT STUDIES UAFIA -
* 
* ELES RELATED TO THE MOUEMENT CAN EE STORED AT
* USERS CHOICE.
****************************************************
IDT 'RAMF1'
DEF RAMF
REF OPT, MESSAG, DCFK, NUMEE2, SIGFED
REF ACFOS2, EFRK, UK
REF STORE1, FREQ
REF TENS1, TENS2
FEF TEN4, TEN3, TEN2, TEN1
REF DEADD, TWOCF 2, DSMUL, EFRORI
REF DISTA1, DISTA2, COMFO1, COMFO2
REF SFEED1, SFEED2, ACTUL1, ACTUL 2
REF INCFE1, VOIK, AO1K, DAO1K
**********************
* READ SFEED(MM/MIN) *
**********************
FAMF XOF @LFCK, 14 LFCF

|  | XOF. | @SFEED, 14 | disflay sfeed message |
| :---: | :---: | :---: | :---: |
|  | CLE | R2 |  |
|  | CLR | F4 |  |
|  | CLR | F6 |  |
|  | CLR | F8 | CLEAR REGISTERS |
|  | CLF | F10 |  |
| RESFED | EL | @NUMEE2 | READ CHARAC |
|  | JMF | RAMP | EFFROR! |
|  | JMP | ASSSPE |  |
|  | MOV | R4, R2 | ASSEMELE NUMEER |
|  | mov | R6, R 4 |  |
|  | mov | R8, R6 |  |
|  | Mov | R10, R8 |  |
|  | MOV <br> JMF | R1, R10 RESFED | NEW CHARAC |

*********************************

* ASSEMELING EINARY SFEED ON R3 *
*********************************
ASSSPE MFY @TEN4, R2 R2*10000

| MFY @TEN3,R4 | F4*1000 |
| :--- | :--- |
| MFY @TEN2,R6 | F6*100 |
| MFY CTEN1, F8 | R8*10 |

MFY CTEN1, R8 • R8*10
$\mathrm{R} 10+\mathrm{Fi} 9$
$\mathrm{F} 9+\mathrm{R} 7$

| 0049 | $003 A$ | $A 147$ |
| :--- | :--- | :--- |
| 0050 | $003 C$ | $A 0 C 5$ |
| 0051 | $003 E$ | $C 803$ |
|  | 0040 | 0000 |
| 0052 | 0042 | 1602 |
| 0053 | 0044 | 0460 |
|  | 0046 | 0000 |

0054
0055
0056
00570048 2FA0
004A 022C'
0058 004C 2FAO
004E 0251'
$0059005006 A 0$
0052 0000
0060005410 F 9
0061 0056 04C2
$0062005804 C 4$
0063 005A 04C6
0064 005C 04C8
0065 O05E 日4CA
0066 0060 04CC
0067 0062 OGAO
0064 0014'
$0068 \quad 006610 F 0$
006900681007
0070 006A C084
0071 006C 106
0072 006E C188
0073 0070 C20A
$00740072 \mathrm{C28C}$
$00750074 \quad \mathrm{C} 301$
$0076 \quad 0076$ 10F5
0077
0078
9079
0080
$00810078 \quad \mathrm{C802}$
007A FF2A
007C C820
007E 0000
0080 FF22
$00830082 \mathrm{C820}$
0084 0000
0086 FF24
00840088 02E日
OOBA FF20
-085 008C 06A0
$008 E 0000$
00900460
00920000
00870094 02EE
0096 FFGO

| A | K7，KS | R7＋K5 |
| :--- | :--- | :--- |
| A | FS，R3 | RS＋R3 |
| MOV | R3，＠SFEED1 | SAUE SFEED |
|  |  |  |
| JNE | DIST1 | NOT NULL！ |
| B | OOFT | RETURN TO MAIN |

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊READ DISTANCE $+/-$ MICFONS ）＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
DIST1 XOF＠LFCF， 14
LFCF

DISFLLAY DISTANCE MESSAGE

READ SIGN

NOT A SIGN！
CLEAR FEG

READ CHAFAC

EFRFOR！

ASSEMELLE NUMEEF
MOV Fib，F44
MOU Fi8，FiG
MOU FIO，FB
MOU Fis，Fig
MOV Fi1，Fi2
JMF NEXT 1 NEXT CHAFIAC
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊ASSEMELING EINARY DISTANCE＊
＊（SIGN IS ON FiG）＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
ASSDIS MOU Fi2，＠＞FF2A F゙2＊100000
MOU＠TENS1，＠）FF22 CTE 100000

MOU＠TENS2，＠ MFF 24

LWFI＞FF2O WOFKSF
EL＠DSMUI MTFY
E：＠EFFROFI EFFiOF！
LWFI $3 F F G O$


009A FF26
0089 009C COAO 009E FF24
0090 00AD 3920 00A2 0028＇ 0091 00A4 39AD 00A6 002C＇ 0092 00AB 3A20 00AA 0030＇ 0093 ODAC 3AAD gOAE 0034＇ 0094 OOEO A2CC 0095 00E：2 A24B 0096 00E4 A1C9 0097 00E：6 04C6 0098 00E8 02E0 DOEA FFOG OO99 OOEC OGAD OOEE 0 000 0100 OOCO 0460 00 C 2 0092＇
0101 00C4 02E0 00C6 FF02
0102 日0C8 06AO OGCA OOEE
0103 0日CC 0460 00cE 日＠C2＇

0104
0105
0106
0107 OODO 02E0 BOD2 FFGO 0108 00D4 C042 0109 00D6 C083 0110 00D8 CO00 0111 00DA 1304
01.12 OODC OGAD OODE OOOD
0113 OQED 0460
00E2 ODCE＇
0114 OOE4 C102
0115 00E6 COC1
0116 00EB COGO OOEA OODO
0117 gOEC COAD OOEE BOOD
0118 00FO C803 00 F 2 OODO
0119 00F4 C804 00 F 6 ODOD
0120 DOF8 O6A0 OOFA ODDE
0121 0日FC 0460

MOV＠ $\operatorname{MFF} 26, \mathrm{~F} 3$
MOV＠ $\operatorname{PFF24,~R2}$
MFY＠TEN4，R4 R4＊10000
MFY＠TEN3，R6 R6＊1000
MF＇Y ETEN2， FB FB＊100
MFY＠TEN1，F10 R10＊10
A K12，F11 $1+10$
A K11， $\mathrm{F} 9 \quad 1+10+100$
A $\mathrm{FQ}, \mathrm{Fi} 7 \mathrm{~F} \quad 1+10+100+1000$

CLE R6
LWFI＞FFG6
EL＠DEADD
E CERFORI
LWFI＞FFE2
EL＠DEADD
B EEFROR1
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊INCFEMENTAL DISTANCE＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
LWFI＞FFG日
MOU R2，F1
MOU R3， Fi 2
MOU KO，KO
JEQ FOSDIS
EL CTWOCF2
E CERROR1
FOSDIS MOV R2，F4
mov Fi，K3
MOU セACTUL1，F1
MOU＠ACTUL2， K 2
MOU F3，＠COMFO1
MOU R．4，eCOMP02
EL ©TWOCF2 SUETK
B＠EFFORI EFROR

RESTORE WORKSF
SIGNED RESULT

FOSITIVE

EFROR
COMFO－ACFOS
ACTUAL FOSITION

SAVE NEW COMMAND


TXMIRA
2．3．0 78． 244 00：37：42
$01 / 01 / 00$
FAGE ODOS IMFI FUNCTION COMMAND FOUTINE


0163 0150 10FA 01640152 06A0 $01540172^{\prime}$
0165
0166
0167
016801560001
016901580000
$0170015 A 0002$
$0171015 C 0000$
$0172015 E$ 0003
017301600000
017401620004
017501640000
017601660005
017701680000
0178 016A 0006
0179016 C 000
0180016 E 000
0181
0182
0183
0184
01850170 O5CE
01860172 C29E
01870174 13E4
01860176 803E
0189.0178 16FE
0190017 A C29E
0191
0192
0193
0194 017C 2FA0
017E 022C'
$019501802 F A 0$
$01820344^{\prime}$
$0196018406 \mathrm{AD}^{0}$
$01860148^{\prime}$
$0197018810 F 9$
0198018 A 1002
0199018 C 001
Q200 018E 10FA
020101900280
01920001
020201941305
020301960280
01980002
0204 019A 16F0
$0205019 C 0460$
$019 E$ 0046'
0206
0207
0208

|  | MOU | Fi，FO |
| :--- | :--- | :--- |
|  | JMF | FE2 |
| CHE2 | ELL | ＠SFCH2 |

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊SEARCH TAELE＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
DATA $>0001$
DATA ACFOS2
DATA $>0 日 O 2$
DATA ERFK
DATA＞0日ロ3
DATA UK
DATA $700 G A$
DATA VOIK
DATA 7 DGOS
DATA AOIK
DATA $>0006$
DATA DAOIK
DATA $>0000$

＊SEARCH FOUUTINE＊
＊UAFIAELE ADDFESS ON Fig＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
SFICHO INCT FII
SFCH2 MOU＊Fil，Filg
JEQ OFT1
C $\quad$＊R1It， $\mathrm{F} G$
JTE SFiCHO
MOV＊FI1，FiG
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ ＊MOUE OR FETUFN TO MAIN＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
MOU1 XOF ELFCR， 14
XOF GMOUE， 14
FEE3 EL ENUMEES
JMF MOU1
JMF CHE3
MOU Fi，FiO
JMF FEZ
CHE3 CI FOD， 1
JEQ MOVEI
CI FO， 2
JNE MOU1
E OOFT
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊SYNCHFONIZE MOUEMENT＊
求 $\boldsymbol{c}^{*} * * * * * * * * * * * * * * * * * * * *$

WAIT CF
SEARCH FOUTINE

FIFSS OFTION

## END OF TAELE

UFDATE FOINTER
SEAFCH FAIL？
YES！
MATCH？
NO！
UAKIAELE ADDFESS

DISFLAY MOVE MESSAGE

FEAD CHAFAC

CHECK
save
WAIT CF
MOUE？

YES！
NO？
EFFROR！

| MFF 1 | TXMIRA |  |  |  | 78.244 00:37:42 01/01/00 |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| MF' 1 | FUNCT | ION CO | MMAND | OUTINE |  |  |
| 0209 | 01 A 0 | 0340 | movei | IDLE |  | SYNCHR INT3 |
| 0210 | 01A2 | 0205 |  | LI | $\mathrm{F} 5,112$ | STORE 112 WORDS |
|  | 01A4 | 0070 |  |  |  |  |
| 0211 | 01a6 | 0206 |  | LI | R6, store 1 | LOCATION |
|  | 01 A8 | 0000 |  |  |  |  |
| 0212 | 01 AA | 0720 |  | SETO | @MESSAG | set message flag |
|  | 01AC | 0000 |  |  |  |  |
| 0213 | O1AE | 0340 | WAIT3 | IDLE |  |  |
| 0214 |  |  | ****** | ****** | ******************** | ************ |
| 0215 |  |  | * FRAMF | MOVEM | MENT | * |
| 0216 |  |  | * STOF | ES FIR | RST 112 SAMPLES OF | CHOSEN UAR * |
| 0217 |  |  | ****** | ***** | ********************* | ************ |
| 0218 | 01 EO | CoEd | MOVE2 | MOV | @DISTA1, F 3 ( | DISTANCE TO MOUE |
|  | $01 \mathrm{E2}$ | 010A' |  |  |  |  |
| 0219 | 0184 | C120 |  | mov | @DISTA2, K 4 |  |
|  | 01 E 6 | 010E' |  |  |  |  |
| 0220 | 01E8 | 1603 |  | JNE | VA1 | NOT EQUAL |
| 0221 | O1EA | cec3 |  | MOU | R3, F 3 |  |
| 0222 | 01EC | 1601 |  | JNE | UA1 N | NOT EQUAL. |
| 0223 | 01EE | 102E |  | JMP | LATA |  |
| 0224 | 01C0 | 04C1 | UA1 | CLR | Fi | -DOUELE FREC INCREM |
| 0225 | 01 C 2 | coab |  | mov | @INCRE1, F 2 | INCEEM |
|  | 01c4 | 013C' |  |  |  |  |
|  | 01c6 | 1502 |  | JGT | NEG1 |  |
| 0227 | $01 \mathrm{C8}$ | 0742 |  | AES | Fi 2 N | Neg to fos |
| 0228 | 01CA | 1002 |  | JMF- | ADDB | ADD |
| 0229 | 01CC | 0701 | NEG1 | SETO | F1 F | FOS TO NEG |
| 0230 | O1CE | 0502 |  | NEG | F2 |  |
| 0231 | 0100 | 66A0 | ADD8 | EL | CDEADD A | ADD |
|  | 0102 | 0102' |  |  |  |  |
| 0232 | 0104 | 0460 |  | E | @EFFROF1 E | EFFROR |
|  | 01D6 | 0106 |  |  |  |  |
| 0233 |  |  | ****** | ****** | ******************** | *** |
| 0234 |  |  | * INCR | EM GRE | EATER THAN DISTAN ? | * |
| 0235 |  |  | * IF ( | DISTA | -INCRE) CHANGES SIGN | N * |
| 0236 |  |  | * THEN | INCFE | E GT DISTA | * |
| 0237 |  |  | ****** | ****** | ******************** | *** |
| 0238 | 0108 | C1C3 |  | MOV | F3, F7 |  |
| 02390240 | Q1DA | 1102 |  | JLT | NEGS |  |
|  | 01DC | 0467 |  | CLE | F7 7 |  |
| 0241 | O1DE | 1001 |  | JmF' | FOSS |  |
| 0242 | $01 \mathrm{E} 日$ | 0707 | NEG5 | SETO | F7 |  |
| 0243 | 01 E 2 | C201 | FOSS | MOV | F1, F 8 |  |
| 0244 | B1E4 | 1102 |  | JLT | NEG6 |  |
| 0245 | 0156 | 0468 |  | CLE | F8 |  |
| 0246 | 01E8 | 1001 |  | JMF | FOS6 |  |
| 0247 | 01EA | 0708 | NEG6 | SETO | F88 |  |
| 0248 | 81EC | 61 CB | F'OS6 | 5 | F8, F7 7 |  |
| 0249 | Q1EE | 1307 |  | JEQ | CON 1 |  |
| 0250 | $01 F 0$ | C804 |  | MOV | F4, ©DCF'K S | SEND LAST INCREMENT |
|  | 01F2 | 0000 |  |  |  |  |
| 0251 | 01F4 | O4E0 |  | CLR | @DISTAI C | Cleak distan |
|  | 01F6 | 01E2' |  |  |  |  |
| 0252 | 0178 | 04E0 |  | CLR | @dIstaz |  |

TXMIFA
2.3.0 78. 244 00:37:42 01/01/00

FAGE 0007 AMFI FUNCTION COMMAND ROUTINE

01FA 01E6
0202 01F2'
02550204 C 001
$020601 \mathrm{~F}^{\prime}$
02560208 C802
020A 01FA'
0257 020C C145
0258 020E 1302
025902100605
02600212 CD9A
02610214 10CC
02620216 C145
026302181301
0264 021A 10FA
0265
0266
0267
0268 021C C820
021E 日0F2'
0220 GOEA'
02690222 C 820
0224 00F6'
0226 GUEE
027002280460
022A 019E'
0271
0272
0273
0274 022C 0ADD
0275022 EA
$022 F \quad 00$
0276023045
02770250 00
$0278 \quad 0251$ 4D
02790271 00
0280027256
$028102990 A$
029A 0D
0282 029E 20
0283 G2EF OA
92C0 OD
0284 日2C1 20
0285 02EG OA
02E7 0D
0286 02E8 20
0287 02F9 $0 A$
02FA 0D
0288 G2FE: 20
02890313 0A
0314 0D
0290031520


TXMIRA
2.3.0 78.244 00:37:42
$01 / 01 / 00$
FAGE 0008 AMFI FUNCTION COMMAND ROUTINE

```
0291 0337 0A
        0338 0D
02920339 20
0293 0343 00
02940344 4D
0295 0358 00
        EYTE \OA, >OD
        TEXT , ?'
        EMTE 0
        MOVE TEXT 'MOVE(YES(1),NO(2))?'
        EYTE 0
        ********
0 2 9 6
0 2 9 7
END
```

TXMIRA
2.3.0 78.244 00:37:42
$01 / 01 / 00$
FAGE 0009
MFI FUNCTION COMMAND ROUTINE


APPENDIX 9.4

SAMPLING FREQUENCY ROUTINE


0001 0002 0003 0004 0005 0006 0007 0008 0010 0011 0012 0013 0014 0015 0016 0017 $001800002 F A 0$ 00020054
0019 OOO4 2FAO 0006 0058 00200008 04C2 0021 000A 04C4 0022 日00C 04C
0023
0924
0025
0026
0027001210 F 6
002800141004 $00290016 \mathrm{C084}$ 00300018 C106 0031 001A C181
0032001 C 10 FB
0033
0034
0035
0036 001E 38AO 0020 0000
003700223920
00240000
0038 0026 A146
00390028 A0C5
0040 002A 0283 002 C 0005
0041 002E 11 E 8
$00420030 \mathrm{C803}$ 0032 0000
******************************************************

* NEWFRE. THIS ROUTINE READS THE SAMFLING FREQUENCY *
* FFOM A TEFMINAL IN dECIMAL FOFM. IT TRANSFORMS THE *
* FREQUENCY in a corresfondent feriod acceftable ey *
* TMS 9901 AND STORES IT AT LOCATION FERIOD. IF THE *
* FEEIOD IS different from the alfeady existent the *
* ROUTINE SETS THE FLAG FEFLAG (NULL). *
******************************************************
IDT 'NEWFRE'
DEF NEWFRE
FEF FERIOD, FEFLAG, NUMEEZ
REF OFT, FREQ
REF TEN3, TEN2, TEN1
*******************
* DISFLAY MESSAGE *
*******************
NEWFFE XOF QLFCF, 14
LFCR
DISFRE XOF @FREQ1,14
DISFLAY FREQ MESSAGE
CLR F 2
CLEAR REGISTERS
CLR K4
CLF RG
****************** * FEAD FFEQUENCY * ******************
REAFFE EL GNUMEE2 READ CHAFACTEF

| JMF | NEWFRE | EFROR! |
| :--- | :--- | :--- |
| JMF | ASSEM1 |  |
| MOU | F4, R2 | ASSEMELE NUMEEE: |

MOU RG, Fi4
MOU R1, R6
JMP REAFRE
********************************

* ASSEMELING EINARY FREQ ON R3 *
********************************
ASSEM1 MFY @TEN2, R2 R2*100
MFY @TEN1,R4 F4*10
A R6, FS K6 FE

A R5,R3 R5+R3
CI R3, 5 LESS THAN 5 HZ ?
JLT NEWFRE YES!
MOU R3, ©FREQ SAUE FREQ.

* CONUERTING FREG TO FERIOD *
* PEEFIOD(CLOCK UNITS)=10**6/(F*21.3) *
* FERIOD=46948/F
* 

**************************************


## APPENDIX 9.5

## displaying values routine



0001 0002 0003 0004 0006 0007 0008 0009 0010 0011 $0012000004 C 1$ 001300020209 O日0 4 FFF8 00140006 2FA日 0008 0000 0015 000A COA1 $000 C 0000$ $0016000 E 0742$ 001700101103 001800120200 00142000 001900161002 002000180200

001 A 2 DGO 0021 OO1C DGAD
$001 E$ 0000
$002200202 F 80$
$002300222 \mathrm{ED2}$ 00240024 GAC3 00250026 0A84 00260028 0A45 0027 002A A146 0028 002C A105 0029 002E ADC4 003000302 E 83 0031 0032 05C1 003200340281 0036 D0E0 003300381305 0034 003A 0589 $0035003 C$ 13E2 0036 O03E 2FAD 0040 0048 $0037004210 E 3$ 003800440460 00460000

004200482020 0043 004A 2000 0044
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊THIS FOUTINE DISFLAYS 112 SAMFLES OF CHOSEN＊ ＊UARIAELE STORED EY THE FAMF INFUT OFTION＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
IDT＇DISFLA＇
DEF DISFLA
FEF OFT，STOFE 1，HEXDEC，LFCR
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊DISFLAY TAELE OF UALUES＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ $\mathfrak{c}^{*} * * * * *$

DISPLA CLF RI
R98 LI $\quad \mathrm{F} 9,-8$

XOF＠LFCR， 14

TAVA MOU＠STORE1（Fi1），FI2
AES FIS
JLT NEG
LI $\mathrm{FO}, 12000$
JMF FOS
NEG LI F： $\mathrm{LI},>2 \mathrm{DOG}$
FOS EL＠HEXDEC

XOF $\mathrm{KQ}, 14$
XOF KN， 8
SLA $\mathrm{Fi} 3,12$
SLA FR4， 8
SLAA FSS， 4
A R6，RE
A RS，R4
A FA，R3
XOF RZ，10
INCT Fil
CI R1，224
JEQ RET
INC FO
JEQ R98
XOF＠SFACE3， 14
JMF TAUA
RET E GOFT RETURN TO MAIN

## COUNTEF <br> LINE COUNTER

LFCF

GET UALUE

AES UALUE
NEGATIUE！
SFACE FOF FOSITIUE

NEGATIUE SIGN

HEX－DEC CONUERT

DISFLAY SIGN
R2＊100日0
FREFARE FOR DISFLAY

DISPLAY NUMEEF
INCFEM COUNTEF：
END？

FETURN
LINE

3 SF＇ACES
＊＊＊＊＊＊＊＊＊＊＊
＊MESSAGE＊
＊＊＊＊＊＊＊＊＊＊＊
SFACEZ DATA＞2020
DATA $>2000$
END

|  | LA | TXMIFA |  |  | 2.3.0 | 78.244 00:41:50 |  |  | $01 / 01 / 00$ |  | F'AGE 0002 |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | AY SAM | ED | - | OF CH | SEN UAR |  |  |  |  |  |  |  |
| D | DISFLA | 0000 | E | HEXDEC | 001E | E | LFCR | 0008 | , | NEG | 0018 |  |
| $E$ | OF'T | 0046 | , | FOS | O01C |  | F0 | 0000 |  | R1 | 0001 |  |
|  | R10 | O日GA |  | Fil | OOOE |  | R12 | 000C |  | R13 | 0000 |  |
|  | R14 | OQOE |  | F15 | O00F |  | R2 | 0002 |  | R3 | 0003 |  |
|  | F4 | 0004 |  | RS | 0005 |  | K6 | 0006 |  | K7 | 0007 |  |
|  | F8 | 0008 |  | F9 | 0009 | , | F98 | 0002 | , | RET | 0044 |  |
| , | SFACE3 | 0048 | $E$ | STOKE1 | 000C | , | tava | OODA |  |  |  |  |

MMMUN OMMUNICATION ROUTINE FOR LOOF CLOSING

0001 0002 0003 0004 0005 0006 0007 0009 0010 0011 0012 0013 0014 0015 0016 0017 0018 0019 0020 0021 0022 0023 0024 0025 0026 0027 0028 0029 0030 0031 0032 0033 0034 003500000300 00020000

0004 C800 0006 1FFG

|  |  |
| :--- | :--- |
| 0008 | $020 C$ |
| $000 A$ | 0100 |
| $000 C$ | $1 E 11$ |
| $000 E$ | $1 F 16$ |
| 0010 | $16 F E$ |

$1 F F 0$
1000
1002
1004
1006
1008
$100 A$
$100 C$
$100 E$
1010
1012
1014
1016
0300
0000
*****************

* READ COMMANDS *
*****************
MOU @M1DCFK, @DCFK FEAD COMMAND INCREM.


## **********************

* INTEFFUFT MODULE 1 *
**********************
MOV FO, CMODI
***********************
* HOLD (CHECK)MODULE 1 * ***********************

LI R12, $>100 \quad 9901$ EASE ADDFESS
SEZ 17 - SET HOLD LINE
TRY TE 22 HAS IT STOFFED?
JNE TRY
*******************************************************

*     * 
* THIS IS THE SYSTEM EUS COMMUNICATION FOUTINE TO *
* EE USED EY MODULE 2 . IT INTERRUPTS MODULE 1 (SLAUE) * * AND holds it up to message comfletion. *
*     * 



| IDT | 'COMMUN' |
| :--- | :--- |
| DEF | COMMUN, M1DCFK, M1FEFG, M1FEF' |
| DEF M1ACF'1,M1ACF'2, M1ERFO, M1UK |  |
| DEF M1MESS,M1UO1K, M1AO1K, MDAOIK |  |
| FEF | DCFK, FROCES, NEWFFE, FERIOD |
| REF ACFOS1, ACFOS2, EFFK, UK |  |
| REF MESSAG, VOIK, AO1K, DAO1K |  |

***************************************************

* MODULE 1 IS REACHED ON ADDFESSES >100日 TO >IEFE *
* WHICH CORFESFOND TO ADDFESSES $\quad>F Q O Q$ TO >FEFE *
* ON mODULE 1
* 

***************************************************
MODI EQU $\triangle 1 F F G \quad$ ADDFESS FOR MODI INT.
MIDCFK EQU $>100 日$ DCFK ON MOD 1
NEWFRE ON MOD 1.
FEFIIOD ON MODI
ACFOS1 ON MOD1
ACFOS2 ON MOD1
EFFKK ON MODI
UK ON MODI
VELOCITY ON MODI
ACCEL. ON MODI
TRANS ACCEL. ON MODI
MESSAGE INDEX
END OF MESSAGE FLAG
JUST FOR SAFETY

INT MODI(WRITE TO IT)

NO? WAIT

00141000
00160000


CLR @M1DCFK
MOU @M1FEFG, ©NEWFRE

SETO @M1FEFG
MOU @M1FER, CFERIOD

## MOU @M1MESS, @MESSAG




XMIRA
2. 3. 0 78.244 00:42:36

## APPENDIX 9.7

## MODULE 1 COMMUNICATION SOFTWARE

IT3M1 TXMIFA 2．3．0 78．244 00：43：47 日1／01／00 FAGE 日0日1 ITEFFRUF＇T 3 ROUTINE FOF MODULE 1

0001 0002 0003 0004 0005 0006 0007 0099 0010 0011 0012 0013 0014 00150000020 C 00020100 00160004 1E10 0017 0006 1D10 0018 0019 0020 $002100081 E 12$ 0022 0023 0024 0025 000A C2E0 000C 0000 0026 B0日E $16 F D$ 0027 0028 0029 003000101012 0031 0032 0033 003400120720

0014 000c＇ 003500160380 0036
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊THIS IS THE SYSTEM EUS COMMUNICATION FOR MODULE 1 ＊
＊IT IS INTERFUFT DFIUEN EY INTEFFUFT 3 ．＊
＊AFTEF THE INTEFRUFT IS CLEARED ON THE FLIF－FLOF＊
＊THE HOLD LINE IS ENAELED IN．THE END OF MESSAGE＊ ＊FLAG MUST EE CLEAFED AT THE END OF COMMUNICATION＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

IDT＇INT3M1＇
DEF INT3M1
FEF ENMEFG
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊FESET FLIF－FLOF＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
INT3M1 LI FK12， 1009901 EASE ADDFESS

SEZ 16
SEO 16
CLF INT3
FESET
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊ENAEILE HOLD IN＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
SE：Z 18
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊END OF MESSAGE ？＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
CONT1 MOU GENMEFG，R11 END OF MESSAGE？
JNE CONT1 NQ！
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊DISAELE HOLD IN＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
SEO 18 DISAELE HOLD IN
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ ＊FESET END OF MESSAGE FLAG＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
SETO＠ENMEFG FIESET FLAG

RTWF FIETURN
END

| VT3M1 | TXMIRA |  |  | 2．3．0 | 78．244 00：43：47 |  |  | $01 / 01 / 00$ | F＇AG |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| STERFUFT | ROUT | F | OR MODUL |  |  |  |  |  |  |
| ，CONT1 | 日日0A | $E$ | ENMEFG | 0014 | D | INT3M1 | 0000 | Fio | 0000 |
| R1 | 0001 |  | R10 | O日OA |  | F11 | OOOE | R12 | 000c |
| R13 | O日0D |  | F14 | O日GE |  | R15 | OOEF | R2 | 0002 |
| R3 | 0003 |  | R4 | 0004 |  | Fi5 | 0005 | R66 | 0006 |
| R7 | 0007 |  | R8 | 0008 |  | F9 | 0009 |  |  |

## APPENDIX 9.8

## ARITHMETIC ROUTINES


JOCF2 TXMIFA 2.3.0 78.244 00:46:13 01/01/00 FAGE 0002 JO"S COMFLEMENT DOUELE FRECISION FOUTINE

| A | 0001 | E | 0002 | LINK | 000E |  | R 0 | 0000 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Ril | 0001 | R10 | OODA | R11 | OQOE |  | R12 | OODC |
| Fi'3 | 0000 | Fil4 | OOOE | R15 | 000F |  | R2 | 0002 |
| R3 | 0003 | R4 | 0004 | K5 | 0005 |  | R6 | 0006 |
| F\%7 | 0007 | F8 | 0008 | Fi9 | 0009 |  | RETURN | 000c |

D TWOCF2 0000


IOCF 3
TXMIFA 2．3．0 78．244 00：46：51
$01 / 01 / 00$
FAGE 0002 IO＇S COMFLEMENT TRIFLE FRECISION FOUTINE

| A | 0001 |  | E | 0002 | C | 0003 | LINK | O日⿹E |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| FO | 0000 |  | Fil | 0001 | R10 | ODOA | Fi1 | ODOE |
| F12 | OOOC |  | F13 | OOOD | F14 | OOOE | F15 | O日OF |
| F2 | 0002 |  | Fi3 | 0003 | F4 | 0004 | FS | 0005 |
| F66 | 0006 |  | F7 | 0007 | F\％8 | 0008 | F9 | 0009 |
| FETURN | 0012 | D | TWOCP3 | 0000 |  |  |  |  |






0001 0002 0003 0004 0005 0006 0007 0008 0009 0010 0011 0012 0013 0015 O日16 0017 0018 0019 0020 0021 0022 0023 0024 0025 0026 0027 00280000 C80E 0002 FEFB
********************************************** *

* THIS FOUTINE MULTIFLIES A SINGLE FFECISION * * EIY A SINGLE FFECISION(FIXED FOINT )NUMEER **
* BOTH NUMEERS ARE IN TWO'S COMFLEMENT FORM *
* IT IS ASSUMED NUMEEFS AFE ON FEGISTERS 1 *
* AND REGISTER 3. FESULT IS FETUFNED *
* ON REGISTEFS 1,2. IF THEFE IS NO EF:ROR *
* FETUKN SKIF'S TWO WORDS. *
* IT USES FOUTINE TWOCF2 AND REGISTEFS 1 TO *
* 5. LOCATION FEF\& IS USED TO STOFE THE *
* RETURN ADDFESS. *
**********************************************
IDT 'SSMUL'
FEF TWOCF'2
DEF SSMUL
A1 EQU 1
E11 EQU 2
A2 EQU 3
SA1 EQU 4
SAT EQU 5
LINK EQU 11
SALINI EQU >FEFB
***********************
* SAUE RETURN ADDFESS *
*** $* * * * * * * * * * * * * * * * * * * *$
SSMUL MOU LINK, @SALIN1
SAUE FETUFN ADDFESS
********************* * NEGATIUE NUMEEF ? * *********************
MOV A1, A1 SIGN?

JGT FOSIT1 FOSITIVE
JLT NEGAT1 NEGATIVE
CLF A1
CLFE EII
JMF F•OSIT3
NEGAT1 SETO SAI
NEG A1
JMF DEALA2
FOSIT1 CLF SA1
*********************

* NEGATIUE NUMEER ? *
*********************
DEALAS MOU A2, A2
JGT FOSIT2
JLT NEGAT2
JMF NULL
NEGAT2 SETO SAZ
NEG A2
JMF MULT
FOSIT2 CLF SA2
******************
* MULTIFLY A2*A1 *

|  | MOV | A1, A1 |
| :---: | :---: | :--- |
|  | JGT | FOSIT1 |
|  | NULL | JLT |
|  | NEGAT | A1 |
|  | CLF | E11 |
|  | JMF | FOSIT3 |

NULL FEESULT

SET NEGATIUE SIGN TWO'S A1

SET FOSITIUE SIGN

SIGN?<br>FOSITIUE<br>NEGATIUE<br>NULL FESULT<br>SET NEGATIUE SIGN TWO'S COMFL. A2<br>SET FOSITIVE SIGN

FAGE OOD1


0001
0002
0003
0004
0005
0006
0007
0008 0009 0010 0011 0012 0013

0002 FEF8

0032
0033
0034
00350004 C 041
0036 0006 150A
003700081103
0038 000A C082
0039 日日GC 1324
0040 OODE 1006
004100109706

00450012 06A0
00140000

```
0 0 4 6
0047
0048
0049 0016 0460
    0018 0066'
0050 001A 1001
0051 001C 04C6
```

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

*     * 
* THIS ROUTINE mULTIFLIES A DOUELE PFECISION *
* EY A SINGLE FRECISION(FIXED FOINT )NUMEER *
* EOTH NUMEERS ARE IN TWO'S COMPLEMENT FORM *
* It is assumed numeers are on kegisters 1,2 *
* (A1+E1) AND REGISTER 5 (A2). IF THERE *
* IS NO ERROR RETURN SKIFS TWO WORDS. *
* result is feturned on fegistefs 1, 2 and 3. *
* IT USES ROUTINES TWOCF2 , TWOCF3 AND *
* REGISTEFS 1.TO 7. LOCATION FEF8 IS USED TO *
* STORE THE FETURN ADDRESS.
* 

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
IDT 'DSMU1'
REF TWOCF 2, TWOCF3
DEF DSMU1
A1 EQU 1
E11 EQU 2
E1 EQU 3
E12 EQU 4
A2 EQU 5
SA1E1 EQU 6
SA2 EQU 7
LINK EQU 11
SALIN1 EQU >FEF8
***************

* Save return *
***************
DSMU1 MOU LINK, 巴SALIN1 SAVE RETURN
*********************
* NEGATIUE NUMEER ? *
*********************
MOV A1, A1
JGT FOOSITI
JLT NEGATI
MOU E:11, E11
JEQ NULL
JMF FOSITI
NEGATI SETO SAIEI
********************
* TWO's CDMFLEMENT *
********************
EL @TWOCP2 TWO'S COMPLE. A1E1
*********
* EFRFOR *
*********
E: ©OUERRO
JMF DEALA2
FOSITI CLR SA1E1
*********************


## SIGN？

fositive NEGATIUE NULL？

SET NEGATIUE SIGN

TWO＇S COMPLE．A1EI

GO TO TIEUG ON ERROR

SET FOSITIUE SIGN

| SMU1 TXMIFA | 2.3 .0 | $78.24400: 49: 51$ | $01 / 01 / 00$ | FAGE 0日02 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| OUELE-SINGLE MULTIFLICATION |  |  |  |  |


| 0053 |  |  | * NEGATIVE NUMEER ? * ********************* |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 0054 |  |  |  |  |  |  |
| 0055 | 001E | C145 | DEALA2 | mov | A2, A2 | SIGN? |
| 0056 | 0020 | 1505 |  | JGT | FOSIT2 | fositive |
| 0057 | 0022 | 1101 |  | JLT | NEGAT2 | NEGATIUE |
| 0058 | 0024 | 1318 |  | JEQ | NULL | NULL RESULT |
| 0059 | 0026 | 0707 | NEGAT2 | SETO | SA2 | SEt Negative sign |
| 0060 | 0028 | 0505 |  | NEG | A2 | TWO'S COMFL. A2 |
| 0061 | 002A | 1001 |  | JMF' | MULT |  |
| 0062 | 002C | $04 \mathrm{C7}$ | FOSIT2 | CLR | SA2 | SET POSITIUE SIGN |
| 0063 |  |  | ****** | ***** | ******* |  |
| 0064 |  |  | * MULT | IFLY | A2*E1 * |  |
| 0065 |  |  | ******* | ***** | ******* |  |
| 0066 | 002E | COC2 | MULT | MOV | E:11, E1 | SAVE EII |
| 0067 | 0030 | 3855 |  | MFY | A2, E1 | MULTY EII EY A2 |
| 0068 |  |  | ******* | ***** | ******* |  |
| 0069 |  |  | * MULT | IFLY | A2*A1 * |  |
| 0070 |  |  | ******* | ****** | ******* |  |
| 0071 | 0032 | 3845 | NOSIGN | MFPY | A2, A1 | multy al Ey A2 |
| 0072 | 0034 | A083 |  | A | Ei1, Eil | ADD E1 T0 E11 |
| 0073 | 0036 | 1703 |  | JNC | SIGN1 | CARRY? |
| 0074 | 0038 | 6581 |  | INC | A1 |  |
| 0075 | B03A | 1901 |  | JNO | SIGN1 |  |
| 0076 | $003 C$ | 1014 |  | JMF | OUEFRO |  |
| 0077 | 003E | coc4 | SIGN1 | Mov | E12, E1 | ASSEMELE FESULT |
| 0078 | 0040 | 6187 |  | 5 | SA2, SA1E1 | FESULT SIGN |
| 0079 | 0042 | 1304 |  | JEQ | FOSIT3 | Fositive |
| 0080 | 0044 | G6ag |  | ELL | @TWOCF3 | TWO'S COMPL, FESULT |
|  | 0046 | 0000 |  |  |  |  |
| 0081 | 0048 | 0460 |  | E | @OVEFRRO | go to tieug on erfor |
|  | 004A | 0066' |  |  |  |  |
| 0082 | 0045 | C2EO | FOSIT3 | mov | @SALIN1, LINK |  |
|  | 004 E | FEF8 |  |  |  |  |
| 0083 | 0050 | 022E |  | AI | LINK, 4 | FETURN WITH NO EFFROF |
|  | 0052 | 0004 |  |  |  |  |
| 0084 | 0054 | 645E |  | E | *LINK |  |
| 0085 | 0056 | 04C1 | NULL | CLF | A1 | NULL FESULT |
| 0086 | 0058 | 04C2 |  | CLR | E11 |  |
| 0087 | 005A | 0453 |  | CLF | E1 1 |  |
| 0088 | 005 C | C2E0 |  | Mov | @SALIN1, LINK |  |
|  | $005 E$ | FEFB |  |  |  |  |
| 0089 | 0060 | 022E: |  | AI | LINK, 4 | FETURN WITH NO EFROF |
|  | 0062 | 0004 |  |  |  |  |
| 0090 | 0064 | 045E |  | E | *LINK |  |
| 0091 | 0066 | C2EO | OUERRO | Mov | @SALIN1, LINK |  |
|  | 0068 | FEF8 |  |  |  |  |
| 0092 | 006A | 845E |  | E | *LINK |  |
| 0093 |  |  |  | END |  |  |


| Smu1 | TXMIFA |  |  | 2.3.0 | 78.244 00:49:51 |  |  | 01/01/00 | FAG |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| OUELE-SINC | MU | - | ATION |  |  |  |  |  |  |
| A1 | 0001 |  | A2 | 0005 |  | E:1 | 0003 | E:11 | 0002 |
| E12 | 0004 | , | dealaz | 001 E | D | DSMU1 | 0000 | LINK | 000E |
| MULT | 602E | , | negat1 | 0010 | , | negat2 | 0026 | - NOSIGN | 0032 |
| NULL | 0056 |  | OUERRO | 0066 | , | FOSIT1 | 001 C | FOSIT2 | 002 C |
| Fosit3 | 004C |  | F | 0000 |  | K1 | 0001 | F10 | 000A |
| R11 | 000E |  | F12 | 000C |  | R13 | 0000 | R14 | 000 E |
| F15 | 000 F |  | F2 | 0002 |  | F3 | 0003 | F4 | 0004 |
| RS | 0005 |  | F6 | 0006 |  | R7 | 0007 | R8 | 0008 |
| R9 | 0009 |  | SA1E1 | 0006 |  | SA2 | 0007 | SALIN1 | FEF8 |
| SIGN1 | 003 E | E | TWOCP2 | 0014 | E | TWOCF3 | 0046 |  |  |

0000 ERFORS
$0118 \mathrm{MATHS3}$ A0000E0541E0502E1703E0581E1901E045EE022EE0004E045E7F247F $0012 \mathrm{E} 0541 \mathrm{E} 0542 \mathrm{E} 0503 \mathrm{E} 1705 \mathrm{E} 0582 \mathrm{E} 1703 \mathrm{E} 0581 \mathrm{E} 1901 \mathrm{EO45EE} 022 \mathrm{EE} 00047 \mathrm{~F} 31 \mathrm{DF}$ 0028EG45EEA084E1703EG581E1901E1005EA043E1901E1002E022EEODO47F306F 003EE045EEAOC6E1705EO582E1703E0581E1901E100AEA085E1703E05817F2CCF 0054E1901E1005EAO44E1901E10R2EO22EEOGO4E045EEC80ECODAAEC0417F2D4F OO6AE1507E1103E04C1E04C2E1013E6704EO501E1001E04C4EC0C3E15057F2F1F 0080E1101E10F5E0705E0503E1001E04C5E3843E6105E1304E06A0C00007F31EF $0096 E 0460 C 00 A 4 E C 2 E 0 C 0 G A E 022 E E 0004 E 045 E E C 2 E O C O G A A E O 45 E E 00007 F 299 F$ OUACECBOEEFEFBEC041E150AE1103EC082E1324E1006E0706E06A0C00007F291F 00C2E0460C0112E1001E04C6EC145E1505E1101E1318E0707E0505E10017F31DF OQD8EO4C7ECQC2E38C5E384EEAQB3E1703E0581E1901E1014ECGC4E61877F293F GOEEE13G4E06AOCOO12E0460C0112EC2E0EFEF8E022EE0004E045EE04C17F296F $0104 E 04 C 2 E 04 C 3 E C 2 E 0 E F E F 8 E 022 E E 0004 E 045 E E C 2 E 0 E F E F 8 E B 45 E 7 F 346 F$ OQOOTWOCF250012TWOCF35002ADEADD 50040TFADD 50064SSMUL 7F2ADF OQACDSMU1 7FD26F

MATHS3 01/01/00 00:19:16 TXLINK 2.3.日

## APPENDIX 9.9

## AUXILIARY ROUTINES

TXMIRA
2．3．0 78．244 00：53：40
$01 / 01 / 00$
FAGE OOO1
ODULE TO READ THE SIGN OF A NUMEEF

0001
0002
0003
0004
0085
0006
0007
0008
0010
0011
0012
0013 OOOQ 2ECO 001400020280 0004 2E00 001500061602 00160008 04C0 0017 000A 1004 $0018000 C 0280$ ロロOE 2DO日 001900101602 002000120700 OQ21 0014 GSCE $00220016045 E$ 0023

```
************************************************
* FOUTINE TO READ THE SIGN OF A NUMEEEF . *
* SIGN IS FEAD ON FEGISTEF FO USING TIEUGG *
* XOF 11.FOF NEGATIVE SIGN FO IS SETO AND FOK *
* FOSITIUE IT IS CLEAFED.FETUFN SKIF'S A WOFDD *
* IF THEFE IS NO EFROR, *
************************************************
*
    IDT 'SIGFED'
    DEF SIGFED
*
SIGFED XOF FO, 11
            CI KO, >2EBO
            JNE NEG
            CLFR FO
            JMF}\mathrm{ FET
            CI FO, \2DOQ
            JNE EFRKOF
            SETO FO
            INCT FIII
            E: *Fi11
```

            END
    | I GFED | $01 / 01 / 00$ | FAGE ORO？ |
| :---: | :---: | :---: |
| ODULE |  |  |


| ERFOR | 0016 | ，NEG | 000c |  | Fio | 0000 |  | R1 | 0001 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| F10 | OQOA | F11 | goge |  | R12 | $000 C$ |  | K13 | 0000 |
| R14 | GOQE | F15 | O日GF |  | R 2 | 0002 |  | R3 | 0003 |
| R4 | 0004 | RS | 0005 |  | R6 | 0006 |  | R7 | 0007 |
| F8 | 0008 | F9 | 0009 |  | FET | 00.14 | D | SIGRED | 0000 |

$01 / 01 / 00$
FAGE 0001


UMEE2
TXMIFA
2.3.0
78.244 00:54:14
$01 / 01 / 00$
FAGE 0002
EADING A DECIMAL NUMEEF

| ERFOR | 001 A | D | NUMEE2 | 0000 | Fio | 0000 | F1 | 0001 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| F10 | O00A |  | F11 | 000E | F12 | 0000 | F13 | GEOD |
| F14 | 000E |  | F15 | $000 F$ | F2 | 0002 | F3 | 0003 |
| Fi4 | 0004 |  | FS | 0005 | Fi6 | 0006 | R7 | 0007 |
| Fi8 | 0008 |  | R'9 | 0009 | T1 | 0018 |  |  |


| EXDEC | TXMIRA | 2.3 .0 | 78.244 | $00: 54: 46$ | $01 / 01 / 00$ |
| :--- | :--- | :--- | :--- | :--- | :--- |$\quad$ F：AGE OOD：

0001
0002
0003
0004
0005 0007
0008 0009 0010 0011 0012
$00130000 \mathrm{COC2}$
0014 0002 04C2
00150004 3CAO
00060000
$0016 \quad 0008 \quad \mathrm{C103}$
0017 000A 04C3
0018 OOOC 3CEO
000E 0000
$0019 \quad 0010 \quad \mathrm{C} 144$
$0020001204 C 4$
00210014 3D20
00160000
00220018 C 185
0023 001A 04C5
$0024001 C$ 3D60
001E 0000
00250020 045E:
0026

```
**************************************************
* THIS RDUTINE CONUEFTS AN HEXA NUME:ER GIUEN *
* ON R2, INTO A DECIMAL, IT USES REGISTERS FI2 TO *
* R6. RESULT:R2*10000+R3*1000+R4*100+F5*10+R6*1 *
*************************************************
    IDT 'HEXDEC'
    DEF HEXDEC
    REF TEN4, TEN3, TEN2, TEN1
**********************
* CONUERSION ROUTINE *
**********************
HEXDEC MOU Fi2,R3 TWO WORDS
    CLF F2
    DIV OTEN4,R2 F゙2*10000
    MOV R3,F4 FEMAINDEF
    CLF FO
    DIV ETEN3,FI3 F3*1000
    MOU FR4,FE FEMAINDEF
    CLF F'4
    DIV ETEN2,F44 F4*100
    MOU FSS, FiG
    CLF
    DIV ETEN1,FSS FS*10(FR6*1)
    E: *F11
    END
```

TWO WOFDS
F゙2＊10000
FEMAINDEF
F3＊1000

FEMAINDEF
Fi4＊100

F゙られ10（F゙ $6 * 1$ ）

```
END
```

EXDEC TXMIFA 2．3．0 78．244 00：54：46 01／01／00 FAGE 日GO2

EXDEC CONVERT FOUTINE


| ODINI | TXMIKA | 2.3.0 | 78,244 | 00:55:19 | 01/01/00 | F.AGE 0001 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |

0001 0002 0003 0005 0006 0007 0008 0009 0010 0011 0012 0013 0014 0015 0016 0017 0018 0019 0020 0021 0022 0023 0024 0025 0026 0027 0028 0029 0030 0031 0032 0033 0034 0035 0036 0037 0038 0039 0040 0041 0042 0043 0044 0045 0046 004700000300

00020000 00480004 02E0 0006 FFO 00490008 020C日OOA 0100 0050 000C 1E10 0051 OOOE 1D10 005200100201
*****************************************************

* MODULE 1 INITIALIZATION FOUTINE
*****************************************************
IDT 'MODIN1'
DEF MODIN1, OFT, STOFE1
DEF MESSAG
DEF DCFK, FEFLAG, FERIOD, ENMEFG
DEF ACF'OS1, ACFOS2, EFFK, UK, VO1K, AO1K, DAO1K
DEF INCFE1, ACTUL1, ACTUL2, SFEED1
DEF TEN4, TEN3, TEN2, TEN1, LFCR
DEF TENS1, TENS2, EFFIOF1, FREQ
DEF DISTA1, DISTA2, COMFO1, COMF'02
FEF NEWFFE, FAAMF', NUMEEZ
FEF DISPLA, INTZM1
DEF REINT

| * COMMUNICATION LOCATIONS * *************************** |  |  |  |
| :---: | :---: | :---: | :---: |
|  |  |  |  |
| DCF'K | EQU | >F000 |  |
| FEFLAG | EQU | >FOO2 |  |
| F'EFIOD | EQU | >F904 |  |
| ACFOS 1 | EQU | >FOO6 |  |
| ACFOS2 | ENU | >F008 |  |
| EFFFK | EQU | >FOGA |  |
| UK | EQU | 3FOOC |  |
| V01K | EQU | >FOQE |  |
| A01K | EQU | >F010 |  |
| DAOIK | EQU | >F012 |  |
| MESSAG | EQU | >F014 |  |
| ENMEFG | EQU | >FO16 | END OF MESSAGE FLAG |
| DISTA1 | EQU | >F018 | DISTANCE TO MOUE |
| DISTA2 | EQU | >FO1A |  |
| COMPO1 | EQU | 3F01C | COMMAND POSITION |
| COMPO2 | EQU | >FO1E |  |
| ACTUL1 | EQU | >F020 | ACTUAL FOSITION |
| ACTUL2 | EQU | >F022 |  |
| FREQ | EQU | >F024 | SAMPLING FEEQ. |
| SFEED 1 | EQU | >F926 |  |
| INCRE1 | EQU | >FG28 |  |
| FEINT | EQU | >F02A |  |
| STORE1 | EQU | >F030 | ELOCK OF 100 WOFDS |
| *THIS | LOCK | GOES FFOM >F030 | T0 >F130 |
| ******************** |  |  |  |
| * RESET INTERRUFTS * |  |  |  |
| ******************** |  |  |  |
| MODIN1 | LIMI | 0 | MASK DFF ALL INT |
|  | LWFI | >FFO日 | WOFKKSPACE FOINTER |
|  | LII | F12, 1100 | 9901 EASE ADDRESS |
|  | SEZ | 16 | RESET FLIF-FLOF INT3 |
|  | SED | 16 |  |
|  | LI | F1, >0800 | MASK EUUT 3 |

## 00120800

00530014 31C1
0054
0055
0056
005700160201
00180460
0058 001A $\mathrm{CBO1}$ 001 C FF88
0059001 E 021 00200000
$00600022 \mathrm{CBO1}$ 0024 FFBA
0061
0062
0063
00640026 04E0 0028 FDO日
0065 002A 04E0 $002 C$ F006
0066 OO2E O4E0 0030 F008
00670032 G4E 0
0034 FOOA
00680036 OAEO 0038 FBOC
0069 003A BAE0
003C F018
0070 003E 04E0
0040 FO1A
00710042 04E
0044 FO1C
00720046 EAE0
0048 F01E
0073 004A 04E0
$004 C$ FO20
0074004 E 04 E 0
0050 F022
007500520720
0054 F002
$0076 \quad 0056 \quad 0720$
0058 F016
0077 005A 0201
$005 C 0177$
0078 005E C801
0060 F004
007900620300
00640003

0080
0081
0082
008300662 FAD 0068 009E' 0084 00GA 2FAD

LDCK $\mathrm{Fi} 1,7$
LOAD CRU
**********************************

* INITIALIZE INTERRUF'T ROUTINES * **********************************

LI R1, >460 INIT INT3
MOU F 1, 巴 $3 F F 88$
LI R1, INTBMI

************************

* INITIALIZE VARIAELES * ************************

CLE @DCFK
cle eacfos
CLF @ACFOS2
CLE ©EFKK
CLE eUk
CLF @DISTA1
CLF DDISTA2
CLF @COMFO1
CLE @COMFO2
CLE ©ACTUL 1
cle eactula
SETO eFEFLAG RESET FLAG
SETO @ENMEFG RESET FLAG(END MESSAGE)
LI R1, $>177$ INIT FREQ TO 250 HZ
MOU Fi, CFFERIOD
LIMI 3
****************************

* DISFLAY TAELE OF OFTIONS *
****************************
OFT XOP @LFCR, 14
XOF @OFTAE1,14

CLEAR UARIAELES
save
ENAELE INT3(COMMUN)


| ODIN1 |  | TXMIKA |  |  | 2.3.0 |
| :---: | :---: | :---: | :---: | :---: | :---: |
| OD1 INITIALIZATION FOUTINE |  |  |  |  |  |
| 0131 |  |  | ****** | ***** |  |
| 0132 |  |  | * CONS | TANTS | * |
| 0133 |  |  | ****** | ***** |  |
| 0134 | 0124 | 0001 | TENS1 | DATA | >0001 |
| 0135 | 0126 | 86A0 | TENS2 | DATA | >86A0 |
| 0136 | 0128 | 2710 | TEN4 | DATA | 10000 |
| 0137 | 012A | 03E8 | TEN3 | DATA | 1000 |
| 0138 | 012 C | 0064 | TEN2 | DATA | 100 |
| 0139 | 812E | 000A | TEN1 | DATA | 10 |
| 0140 | 0130 | 0000 | ERFOR1 | DATA | 10000 |
| 01.41 |  |  |  | END |  |


| ODIN1 | TXMIFA | 2.3.0 | 78. 244 | 00:55:19 | $01 / 01 / 00$ | FAGE |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |


| D | ACFOS 1 | F006 | D | ACFOS2 | F008 | D | ACTUL 1 | F020 | D | ACTUL2 | F022 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| D | A01K | F010 | , | CHE 1 | 007A | D | COMFOI | F01. | D | COMF'O2 | FO1E |
| D | DAOIK | F012 | D | DCFK | FOOO | E | DISFLA | 0088 | D | DISTA1 | F018 |
| D | DISTA2 | F01A | D | ENMEFG | FO16 | D | EFFK | FgOA | D | EFFORK1 | 0130 |
| D | FEFLAG | F002 | D | FFEQ | F024 | D | INCFE 1 | F028 | E | INT3M1 | 0020 |
| D | LFCF | 009E | D | MESSAG | F014 | D | MODINI | 0000 | E | NEWFFE | 0080 |
| E | NUMEES | 0070 | D | OF'T | 0066 | , | OFTAE: 1 | O日A2 | , | OF'TAE? | 0008 |
| , | OFTAE:4 | OQDF | , | OF'TAES | 0101 | , | OFTAE:6 | 0121 | D | FEFIOD | F004 |
|  | Fig | 0000 |  | Fil | 0001 |  | F10 | 000A |  | F11 | Q00E |
|  | F12 | O00C |  | F13 | 0000 |  | F14 | OOOE |  | R15 | $00 \cdot \mathrm{~F}$ |
|  | Fi' | 0002 |  | Fi3 | 0003 |  | F4 | 0004 |  | FS | 0005 |
|  | F6 | 0006 |  | F7 | 0007 |  | F8 | 0008 |  | F'9 | 0009 |
| $E$ | FAMF' | 0084 | , | FE1 | $006 E$ | D | REINT | FO2A | D | SFEED 1 | F026 |
| , | SFCHO | 0090 | , | SFCH1 | 0092 | D | STOFE 1 | F030 | D | TEN1 | 01.2 E |
| D | TEN? | 012 C | D | TEN3 | 012A | D | TEN4 | 01.28 | D | TENS 1 | 0124 |
| D | TENS2 | 0126 | D | UK | FOOC | D | VO1K | FOOE |  |  |  |

## MODULE 2 INITIALIZATION ROUTINE

TXMIRA
2.3.0 78.244 00:56:57
$01 / 01 / 00$
F'AGE 0001 NITIALIZATION FOR LOOF CLOSING(MOD2)

0001
0002
0003
0004
0005
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055
****************************************************
*
*

* MODULE 2 INITIALIZATION FOUTINE *
*     * 

****************************************************
IDT 'MATIN'
FEF LOOFIN
DEF MATIN, WAIT3
DEF EFFiK, UK, UK1, VO1K, VO2K, AO1K, AO2K
DEF VOIK1, VO1K?
DEF UK2, UK3, DAOIK, DAO2K, WIIK, W12K
DEF W21K, W22K, W11KF, W12KF, W21KF, W22KF
DEF KIW1K, K1W2K, K1UK,K1UK1, K1UK2, K1UK 3
DEF K2W1K, K2W2K, K2UK, K2UK1, K2UK2, K2UK3
DEF KOEIA, KOE1E, KOE2A, KOESE
DEF KZUOK, KAVOK
DEF NEWFFE, FEFIOD, MASK1, MASKE, DCF'K
DEF DEMU, EMFLAG, FEFLAG, DAC1, DFOS
DEF KEFFK, KUK1, KVOK, KAOK
DEF KUK2, KUK $3, K D A O K, K F M A X A, K F M A X E: ~$
DEF WOFK1, WORK22, WOFK 33, WOKK44, WOFKES
DEF MESSAG, ACFOS1, ACFOS2
DEF SALIN1, SALIN2, SALINA
DEF DXK, DXK1, DXK2, DXK3, DUK, XKMXK4
DEF KCOFFi, KCOFF2, KUCOFI, KUCOF:
DEF UKCOFF, UKDUT
******************************************

* CONSTANTS AND UAFIAELES INITIALIZATION *
******************************************
*CONTROL FUNCTION
FEBO
FE82
FE84
FE86
FE88
FE8A
FFEA
FEBC KDADK EQU $P F E 8 C$
FE8E KFMAXA EQU $>F E 8 E$
FE9O KFMAXE: EQU $\angle F E 90$
FE92 EFRK EQU $>F E 92$
FE94 UK1 EQU 1 SFE94
FE96 UK2 EQU >FE96
FE98 UK3 EQU >FE98
FE9A VOIK EQU $P F E 9 A$
FE9C VO2K EQU $\angle F E 9 C$
FEGE AOIK EQU $\triangle F E Q E$
FEAO AO2K EQU $>F E A D$
FEA2 DAOIK EQU $\angle F E A 2$
FEA4 DAO2K EQU >FEA4
*****QESEFVEF
FEAG KIWIK EQU >FEAG.
FEA8 KIW2K EQU >FEA8
FEAA KIUK EQU $>F E A A$
FEAC KIUKI EQU $>F E A C$


0110
0111
0112
0113

| FEF2 | MESSAG EQU >FEF2 |  |  |
| :---: | :---: | :---: | :---: |
|  | ******************* |  |  |
|  | * ESTIMATED SF'EED * |  |  |
|  | ******************* |  |  |
| FEF4 | DXK1 | EQU | >FEF4 |
| FEF6 | DXK2 | EQU | >FEF6 |
| FEF8 | DXK3 | EQU | ) 2 EF 8 |
| FEFA | XKM $\times$ K4 | EQU | >FEFA |
| FEFC | SALIN2 | EQU | >FEFC |
| FEFE | SALIN4 | EQU | >FEFE |
| FE7E | SALIN1 | EQU | >FE7E |
| FE7C | DXK | EQU | >FE7C |
| FE7A | DUK | ERU | >FE7A |

*************************** * DIFFERENTIAL ESTIMATION * ***************************
FE78 VO1K1 EQU $>F E 78$
FE76 VO1K2 ERU >FE76
********************************* * GAIN MODIF. FOF FETFACT. DIF. * *********************************
FE74 KCOFiFi EQU >FE74
FET2 KCOFF? EQU $>F E T 2$
FE70 UKCOFF ERU $>F E 70$
** $\boldsymbol{c}^{*} * * * * * * * * * * * * * * * * * * * * *$

* GAIN COMMAND FUNCTION *
*************************
FEGE KUCOF1 EQU $>F E G E$
FEGC KUCOF2 EQU $>F E 6 C$
FEGA UKDUT EQU $>F E G A$
******************************* * DIGITAL TO ANALOG CONUEFTEF $*$ *******************************
EOQ2 DAC1 EQU $>E D Q 2$
*********************
* FEEDEACK FIEGISTEF *
*********************
EFF2 DFOS EQU >EFF2
***************** * WOFKING AFEAS * *****************
FFOQ WORK1 EQU $>F F Q 日$
FFG4 WORK22 EQU >FFO4

FFG8 WORK33 ERU $>F F Q 8$
FFGC WORK44 EQU $\angle F F Q C$
FFIO WORK55 EQU $>F F 10$ ********
MATIN LIMI 0
LWFI $>F F O Q$
MASK OFF ALL INT
LOAD WDFKSFACE FOINTEF:
**************************

* INITIALIZING UAFIAELES *
**************************

TXMIFA
2．3．0 78．244 00：56：57
$01 / 01 / 00$
FAGE OOOA NITIALIZATION FOR LOOF CLOSING（MOD2）

```
01620008 0201
    GOOA DOQE
0163 000C 04E1
    ODOE FEEE
01640010 0641
01650012 1101
0166 0014 10FE:
0167 00160201
    0018 0012
0168 001A 04E1
    001C FE92
0169 001E 0641
0170 0020 1101
0171 0022 10FE:
0172 0024 04EO
    0026 FEF4
0173 0028 04E0
    002A FEF6
0174 002C 04EO
    002E FEF8
0.75 0030 04E0
    0032 FEFA
0176 0034 04E0
    0036 FEDA
0177 0038 64E0
    OQZA FE7C
0178 003C 04E0
    003E FEDC
0179 0040 04E0
    0042 FED6
0180 0044 G4E0
    0046 FED8
0181 0048 04E0
    004A FE78
0182 004C B4E0
    004E FE76
0183 0050 04E0
    0052 FE70
0184 0054 04E0
    OQ56 FEGA
0185
0186
0187
0188 0058 0201
    005A 0460
0189 005C C801
    OOSE FFAA
0190 0060 0201
    0062 0000
0191 0064 C801
    0066 FFAC
0192
0 1 9 3
0194
\begin{tabular}{|c|c|c|c|c|}
\hline & LI & R1， 14 & CLEAR 8 CONSE & Cutive \\
\hline \multirow[t]{4}{*}{CLTAE1} & CLR & せW11K（F゙1） & MEMORY LOCATIO & \\
\hline & DECT & Fil & STARTING AT W1 & \\
\hline & JLT & OUT1 & & \\
\hline & JMF & CLTAEI 1 & & \\
\hline OUT1 & LI & F1， 18 & CLEAR 10 CONSE & CUT I UE \\
\hline \multirow[t]{4}{*}{CLTAES} & CLR & ＠ERFK（F1） & MEMORY LOCATID & \\
\hline & DECT & Fil & STARTING AT ER & FK \\
\hline & JL．T & OUT2 & & \\
\hline & JMF & CLTAE：2 & & \\
\hline \multirow[t]{13}{*}{OUT2} & CLF & セDXK1 & \multicolumn{2}{|l|}{\multirow[t]{4}{*}{CLEAR SFEED}} \\
\hline & CLR & ＠DXK2 & & \\
\hline & CLF & ＠DXK3 & & \\
\hline & CLR & ＠ХКМХК4 & & \\
\hline & CLF & ＠DCFK & \multicolumn{2}{|l|}{\multirow[t]{3}{*}{CLEAF COMMAND INCFE}} \\
\hline & CLF： & ＠DXK & & \\
\hline & CLF & ＠UK & & \\
\hline & CLF： & ＠ACFOS 1 & \multicolumn{2}{|l|}{\multirow[t]{6}{*}{}} \\
\hline & CLEF & CACFOS2 & & \\
\hline & CLER & ＠UO1K1 & & \\
\hline & CLR & ＠UO1K2 & & \\
\hline & CLF & QUKCOFF & & \\
\hline & CLR & ＠UKOUT & & \\
\hline \multicolumn{5}{|l|}{＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊} \\
\hline \multicolumn{5}{|l|}{\multirow[t]{2}{*}{＊INITIALIZE EFANCH TO INT3＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊}} \\
\hline & & & & \\
\hline & LI & F1，＞460 & \multicolumn{2}{|l|}{\multirow[t]{4}{*}{EFFANCH}} \\
\hline & MOV & F1，©＞FFAA & & \\
\hline & LI & Fi，LOOFIN & & \\
\hline & MOV & Fi 1 ，©＞FFAC & & \\
\hline \multicolumn{5}{|l|}{＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊} \\
\hline \multicolumn{5}{|l|}{＊CLEAF FLAGS＊ ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊} \\
\hline
\end{tabular}
```

```
01950068 0720
    006A FEEA
0196 006C 0720
    006E FEE8
0197 0070 0720
    0072 FEDE
0 1 9 8
0 1 9 9
0200
0201 0074 0201
    0076 0041
0202 0078 C801
    007A FEE6
0203
0204
0205
0206 007C 0201
    007E 7800
0207 0080 c801
    0082 FEE4
0208 0084 04E0
    0086 FEE2
```

0209
0210
0211
0212
021300880201
008A 01D7
0214 008C C801
ODBE FEFG
02150090020 C
00920100
02160094 1E17
021700961017
02180998 1E10
0219 009A 1D10
0220009 C 31E0
009E FEE2
0221 GOAD 33 ED
ODAS FEFD
0222 00A4 31E0
0日AG FEE4
0223 00A8 0300
OOAA 0003
0224
0225
0226
0227 00AC 0340
0227

TXMIFA
2.3.0 78.244 00:56:57
$01 / 01 / 00$
F'AG
0006 NITIALIZATION FOR LOOF CLOSING(MOD2)

| D | ACFOS 1 | FEDG | D | ACFOS 2 | FED8 | D | A01K | FE9E | D | A02K | FEAg |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| , | cltaei | 000c | , | CLTAE: 2 | 001A | D | DAC1 | E002 | D | DAO1K | FEA2 |
| D | DA02K | FEA4 | D | DCFK | FEDA | D | DEMU | FEE6 | D | DFOS | EFF2 |
| D | DUK | FE7A | D | DXK | FE7C | D | DXK1 | FEF4 | D | DXK2 | FEF6 |
| D | D×K3 | FEF8 | D | EMFLAG | FEE8 | D | ERFK | FE92 | D | FEFLAG | FEEA |
| D | K1UK1 | FEAC | D | K1UK2 | FEAE | D | K1UK3 | FEEG | D | K1UK | FEAA |
| D | K1W1K | FEA6 | D | K1W2K | FEAB | D | K2Uk1 | FEE8 | D | K2UK2 | FEEA |
| D | K2UK3 | FEEC | D | K2Uk | FEE6 | D | K2W1k | FEE2 | D | K2W2K | FEEA |
| D | к3VOK | FEEC | D | к4VOK | FEEE | D | KAOK | FE8A | D | KCORR1 | FE74 |
| D | KCORF2 | FE72 | D | KDAOK | FE8C | D | KERRK | FE80 | D | koE:1a | FECE |
| D | Kobie | FEDO | D | KOE2A | FED2 | D | KOE:2E: | FED4 | D | KFimaxa | FE8E |
| D | KPMAXE | FE90 | D | KUCOR1 | FE6E | D | KUCOR2 | FE6C | D | KUK1 | FE82 |
| D | KUK2 | FE84 | D | Kuk3 | FE86 | D | KUOK | FE88 | E | Loofin | 0062 |
| D | MASK0 | FEE2 | D | MASK1 | FEE4 | D | matin | 0000 | D | messag | FEF2 |
| D | NEWFRE | FEDE |  | OUT1 | 0016 |  | OUT2 | 0024 |  | FEFEO | FEFE |
| D | FERIOD | FEED |  | Fib | 0000 |  | F1 | 0001 |  | F10 | O00A |
|  | R11 | 000E |  | F12 | 000c |  | R13 | 000D |  | F14 | OOUE |
|  | F15 | 000F |  | R2 | 0002 |  | R3 | 0003 |  | F4 | 0004 |
|  | Fis | 00es |  | R6 | 0006 |  | F7 | 0007 |  | R8 | 0008 |
|  | R9 | 0009 | D | SALIN1 | FETE | D | SALIN2 | FEFC | D | SALIN4 | FEFE |
| D | UK | FEDC | D | UK1 | FE94 | D | UK2 | FE96 | D | UK3 | FE98 |
| D | UKCORR | FE70 | D | ukout | FE6A | D | voik | FE9A | D | V01K1 | FE78 |
| D) | V01K2 | FE76 | D | vo2k | FE9C | D | W11k | FEEE | D | W11kF | FEC6 |
| D | W12k | FECO | D | W12kF | FEC8 | D | W21k | FEC2 | D | W21KF | FECA |
| D | W2ak | FEC4 | D | W22kF' | FECC | D | Walt3 | GOAC | D | WORK1 | FFO日 |
| D | WORK22 | FF04 | D | WORK33 | FF08 | D | WDEKK4 | FFOC | D | WORK5S | FF10 |

0000 ERFORS


APPENDIX 9.13

MODULE 1 OBJECT CODE PROGRAM

06D8MOD1F4 A0000E0300E：0000E02E0EFFG0B020CED100E1E10B1D10E02017F243F

 OO3EEG4E0RF01AE04E0EFO1CE04E0EFO1EE04EOEFO20E04EGE：G22E07207F264F $0054 \mathrm{EFO02E0720EF016E0201E0177EC801EF004E0300E0003E2FA0C009E7F2D5F}$ ODGAE2FA日C0日A2EG6A0CO6A4E10F9E1B02EC001E10FAEO6AOCOD92E：00017F2ABF $0080 \mathrm{C} 05 \mathrm{~A} 4 \mathrm{E} 0002 \mathrm{C} 0132 \mathrm{E} 0003 \mathrm{C} 061 \mathrm{ER} 0004 \mathrm{EF} 02 \mathrm{AE} 0000 \mathrm{E} 05 \mathrm{CEEC05EB} 13 \mathrm{EB7F} 2 \mathrm{C} 4 \mathrm{~F}$ 0096E803EE16FEEC2DEE045EEOAODEQADOE：4F50ES449E：4F4EESS3AE20317F241F OQACE2943E4841E：4E47E4520B5341E：4D50E4C49E4E47E：2046ES245E51557F29EF 00C2E454EE4359EOAODE2020E2020E2020E：2020E2032E2952E414DE50207F2F 4F 00D8E494EES05SES40AE0D20E2020E：2020E2020E2020E3329E4449E53507F2EEF OOEEE4C41E5920E5354E4F52E4544E2056E414CE5545E530AE：0D20E20207F2AAF 0104E2020E2020E2020E3429ES354E4F50E2041E4E44E2052E4549E4E547F2F6F 011 AR204DE $4 F 44 E 320 A E O D 3 F E 0000 E 0001 E 86 A 0 E 2710 E 03 E 8 E 0064 E 000 A 7 F 2 E E F$ $0130 E 0000 E 2 F A 0 C 035 E E 2 F A 0 C 0362 E 04 C 2 E 04 C 4 E 04 C 6 E 04 C 8 E 04 C A E 06 A 07 F 2 B 3 F$ $0146 \mathrm{CO} 6 \mathrm{~A} 4 \mathrm{B1} 10 \mathrm{~F} 4 \mathrm{E} 1006 \mathrm{ECO84EC} 106 \mathrm{EC} 188 \mathrm{EC} 20 \mathrm{AEC281E10F6E38A0C01287F283F}$ 015CE3920C012AE39A0C012CE3A20C012EEA24AEA1C9EA147EA0C5ECB037F259F 0172EF026E1602E0460C0066E2FA0C035EE2FA0C0383E06A0C06C0E10F97F29AF 0188E04C2E04C4E04C6EO4CBEDACAE04CCEO6ABC06A4E1GFGE1007EC0847F26EF Q19EEC106EC188EC2OAEC28CEC301E10F5EC802EFF2AEC820C0124EFF227F21EF 01E4EC820C0126EFF24E02E0EFF20E06A0C0538E0460C0130E02E0EFF007F283F 01CAECOEOEFF26ECOADEFF 24E3920CD128E39AOCO12AE3A20C012CE3AAG7F23DF ＠1E0C012EEA2CCEA24EEA1C9E04C6E02EDEFF06E：06AOCO4E6E0460C01307F24AF 01F6E02E0EFFO2E06A0C04E6E0460C0130E02E0EFF00EC042EC083EC0007F280F 020CE1304E06AOCO48CEO46GCO13OEC1B2ECOC1ECO6OEFO20ECOAOEFO227F2A3F 0222EC803EF01CECB04EFO1EEO6A0C048CE0460C0130E06A0C04E6E04607F295F 0238 C 0130 EC 01 EF 018 EC O2EFO1AE04C0EC041E1101E1001E0700EC1207F2CFF 024EEFG26E3920C012CEC1A0EF024E0208E：00B6E3988E3D07E1902E04607F2E3F $0264 \mathrm{C} 0132 \mathrm{ECOOQE} 1301 \mathrm{E0504ECBO4EF028E2FA0C035EE2FA0C03A4E06A07F29FF}$
 Q290E0003EF00CE0004EF00EE0005EF010E0006EF0．2E0000E05CEEC29B7F2E7F Q2A6E13E4EB03EE16FEEC29EE2FADC035EE2FA日C0476ED6A0C06A4E10F97F22EF O2ECE1002RC001E10FAEO280E0001E1305E0280E0002E16F0EO460C00667F2F8F Q2D2E0340E0205B0070E0206EF030E0720EF014E0340EC0E0EFO18EC1207F2DCF 02E8EF01AE1603EC0C3E1601E102BE04C1ECOADEFG28E1502E0742E10027F2A7F 02FEE0701E0502E06A0C04E6E0460C0130EC1C3E1102E04C7E1001E07077F2D7F $0314 \mathrm{EC201E1102EG4C8E1001E0708E61C8E1307EC804EFO日0E04EGEFG187F2C6F}$ $032 A E 04 E 0 E F 01$ AE1007ECB20EF02BEF000EC801EF018ECB02EFO1AEC1457F263F $0340 \mathrm{E} 1302 \mathrm{E} 0605 \mathrm{ECD} 9 \mathrm{AE} 10 \mathrm{CCEC} 145 \mathrm{E} 1301 \mathrm{E} 10 \mathrm{FAECB2OEFO} 1 \mathrm{CEFF} 20 \mathrm{EC} 207 \mathrm{~F} 26 \mathrm{CF}$ 0356EF01EEF022E：0460COO66E0AODEGAD0E454EE5445ES220E5350E45457F2ESF 036CE4428B3135E2054E4F20E3330E3030E3020E：4D4DE2F4DE：494EE293D7F295F 0382E004DE：4F56E4520E544FE2028E5550E2054E4F20E2E2FE2D32E34307F2A7F 0398E2E30B3030E204DE4D29E2058E3D00ES641E5249E4142E4C45E3A207F2E6F 03AEE3129E4143E5455E414CE2050E4F53E4954E494FE4E28E4C45E41537F289F 03 C 4 E 5420 E 54 FE 5244 E 290 AE 0 D 20 E 2020 B 2020 E 2020 E 2020 E 2032 E 29467 F 2 FAF 03DAE4F4CE4C4FE5749E4E47E2045E5252E4F52E284DE4943E：524FE：4E537F239F 03F0E290AB0D20E2020E2020E2020R2020E2033E2943E4F4EE5452E4F4C7F2C9F 0406E2046E554EE4354E494FE4E28E4D49E：4352E4F4EES329E0ADDE20207F27DF 041CE2020E2020B2020E：2020E3429ES350B4545E440AEOD20E：2020E20207F322F 0432E2020E2020E2035E2941E4343E454CE4552E4154E494FE4E0AE0D207F2D4F 0448E2020E2020E2020E2020E2036E2954E5241E4E53E4945E4ES4E20417F307F 045EE4343E454CE4552E4154E494FE4E0AEOD20E2020E2020E2020E203F7F2EAF 0474 E 200 EE 4 D 4 FE 5645 E 2859 E 4553 E 2831 E 292 CE 4 E 4 FE 2832 E 2929 E 3 F 207 F 291 F 048AE0000E0541E0502E1703E0581E1901E045EE022EE0004E045EE05417F306F 04A0E0542R0503E1705E0582E1703E0581E1901E045EE022EE0004E045E7F2FAF O4CCEAOC6E1705E0582E1703E0581E1901E100AEA085E1703E0581E19017F2CAF O4E2E1005EA044E1901E1002ER22EE0004E045EEC80EC0536EC041E15077F2D4F 04F8E1103E04C1E：04C2E1013E0704E0501E1001E04C4ECOC3E1505E11017F2F0F OS0EE10F5E0705E0503E1001E04C5E3843E6105E1304E06A0C048CE04607．F2E3F $0524 C 0530 \mathrm{EC} 2 \mathrm{E} 0 \mathrm{C} 0536 \mathrm{E} 02 \mathrm{EE} 0004 \mathrm{E} 045 \mathrm{EEC2E} 0 \mathrm{C} 0536 \mathrm{E} 045 \mathrm{EE} 0000 \mathrm{ECBOE} 7 \mathrm{~F} 2 \mathrm{AFF}$ OS3AEFEF8EC041E150AE1103EC082E1324B1006E0706E06A0C048CE04607F2A0F $0550 C 059 E E 1001 \mathrm{EO4C6EC145E1505E1101E1318E0707E0505E1001E04C77F2F5F}$ 0566EC0C2E38C5E3845EA083E1703E0581E1901E1014EC0C4E6187E13047F2E4F $057 C E 06 A 0 C 049 E E 0460 C 059 E E C 2 E 0 E F E F 8 E 022 E E 0004 E 045 E E 04 C 1 E 04 C 27 F 252 F$
 OSABE2FA日C05FCEO4C2EO4C4E04C6E06A0C06A4E10F6E1004EC084EC1067F25DF OSEEEC181E1GF8E38AOC012CE3920CO12EEA146EAOCEEO283EOOOSE11E87F27AF 05D4EC803EF024E04C0E0201EE764E3C03E0A10E0580E0300E0000EC8007F2C8F
 0600E4C49E4E47E：2046E5245E5155E454EE4359E2835E2054E4F20E39397F2EOF O616E3920E4B5AE293DE0DODE04C1E0209EFFF8E2FA0C009EEC0A1EF0307F260F 062CE0742E1103E0200E2000E1002E0200E2D00E06A0C0682E2F80E2E027F305F O642E0AC3E：OAB4EOA45EA146EA105EA0C4E2E83E05C1E0281EOUEOE13057F289F 0658E0589E13E2E2FA0C0666E10E3E0460C0066E2020E2000E020CE01007F2E6F O66EE1E10E1D10E1E12EC2E0EFO16E16FDE1D12E0720EF016E0380ECOC27F262F 0684E04C2E3CAOCO12BEC103EO4C3E3CEOC012AEC144E04C4E3D20C012C7F27CF O69AEC185E04C5E3D60C012EE045EE2EC1E0281E0D00E1308E06C1E02217F28FF 06EOEFFD0E1105E0281E0009E1502E05CEE05CEE045EE2EC0E0280E2E007F280F O6C6E1602E04CEE1004E02B0E2D00E1602E0700E05CEE：045E7F4FDF OOOOMODIN1500660FT 6F030STORE16F014MESSAG6F000DCFK 7F2CCF F0＠2FEFLAG6F0日4FER1OD6FG16ENMEFG6F006ACFOS16F0日8ACFOS27F209F
 F028INCRE16F020ACTUL 16F022ACTUL26F026SFEED150128TEN4 7F28DF CO12ATEN3 5012CTEN2 5012ETEN1 5009ELFCR 50124TEN51 7F3EAF O126TENS2 50130EFFOK16FG24FFEN GFO18DISTA16F01ADISTA2フF2A9F F01CCOMF016F日1ECOMPO2505A4NEWFRE50132FAMF 506A4NUMEE2TF22EF O61EDISFLAS066AINT3M16F02AFEINT S06C0SIGRED504E6DEADD 7F240F ：048CTWOCF250538DSMU1 5049ETWOCF3504CCTF＇ADD 504F0SSMUL 7F221F 6682HEXDEC7FD13F
MOD1F4 01／01／00 00：04：54 TXLINK 2．3．0

[^0]MODULE 2 OBJECT CODE PROGRAM


 OU3EEFEDCEOAEOEFEDGEO4E0EFED8EOUEOEFE78E04E0EFE76EO4EOBFE7O7F19EF O054E04E0EFE6AE0201E0460EC801EFFAAE0201C0374EC801EFFACE07207F250F H06AEFEEAE0720EFEE8E0720EFEDEE0201E0041EC801EFEE6E0201E78007F220F O680EC801EFEE4EO4E0EFEE2E0201E01D7EC801EFEFQE020CE0100E1E177F23FF OQ96E1D17E1E10E1D10E31EOEFEE2E33E0EFEF0E31E0EFEE4E0300E00037F23FF
 МOC2E1000EFEDAEO4EOE1000EC820E1002EFEDEE0720E1002EC820E10047F290F OOD8EFEEGEC820E1014EFEF2EC820EFED6E1006EC820EFED8E1008EC8207F 1FDF НOEEEFE92E10日AEC820EFEDCE10日CEC820EFE9AE100EEC820EFE9EE10107F1DDF 0104EC820EFEA2E1012E04E0E1016E1D11E0300E0003E0460C0118E04607F2F1F O11ACOOACE1000EC80BEFEFCECOEOEFE9AECO6OEFEECEO6AOCOS9OEO4607F 1EFF $0130 C 0508 E O A 11 E C 0 E 0 E F E E E A 043 E C 801 E F E 9 E E C O E O E F E G A C O 6 O E F E E F F 1 E E F$ 10146E06AOCO590E0460C0508EOA11ECOEOEFEC2EA043EC801EFEADEC2E07F252F $015 C E F E F C E O 45 E E C O E E F E F E E C O E Q E F A G E C Q 60 E F E E E E O G A O C Q 590 E O 4607 F 1$ A $2 F$ 0172C0508E02EOEFFOAECOEOEFEA8ECO6OEFEC2EO6AOCOS90EO460C05087F250F 0188E02EOEFFOBEC060EFEAAECOEOEFE9AE06AOC0590E0460C0508E02E07F22EF O19EEFFOCECO6OEFEACECOEOEFE94EO6AOCO590EO46OCOSO8EO2EOEFFO87F2O3F Н1E4E06A0C0556E0460C0508E02E0EFF04E06A0C0556E0460C0508E02E07F2EDF O1CAEFFOOEO6AOCOS56E0460C0508EOA11EOA12E1801E1002E0221E00017F2E7F O1E日ECO41E1502E1103E100CE＠4C5E1002EG70SEOSO1ECOEGEFECEE38437F2ASF


 O238E0460CO508E02EOEFFGCECO6OEFEEBECDEOEFE94EO6AOCO590E04607F249F Ұ024EC0508E02E0EFF08EO6A0C0556EO460C0508EO2EOEFFG4EO6A0CO5567F28FF 4264E0460C0508E02E0EFFG0EO6AOC0S56E0460C0508E0A11EOA12E18017F2DAF O27AE1002E0221E0001EC041E1502E1103E100CE04C5E1002E0705E05017F325F H290ECOEOEFED2E3843E3C60EFED4EC145E1301E0501EC801EFECAEC2EOTF21OF O2A6EFEFEE045EEC8OEEFETEECOGOEFE92ECOEOEFEBOEO6AOCOS9OEO46O7F1DAF 102ECCOS08E02E0EFFO4ECO60EFE94ECOEGEFE82E06A0C0590E0460C05087F24AF O2D2E02EOEFF08ECO6OEFE9AECOEOEFE88ED6AOC0590EO460CO508E02E07F239F 102E8EFF OCECO6OEFE9EECOEOEFEBAE日6AOCO590EO460C0508ED2EGEFF107F204F
 $0314 \mathrm{COS56EO460C0508EO2EOEFFO8EO6A0COS56EO460C0508EO2EDEFF047F2AFF}$ 1032AE06A0C0556E0460C0508E02E0EFF00E06A0C0556E0460C0508EOA1．17F2C6F 10340E0A12E1801E1002EO221E0001EC041E1502E1103E100CE04C5E10027F32CF $10356 E 0705 E 0501 E C 0 E 0 E F E 8 E E 3843 E 3 C 60 E F E 90 E C 145 E 1301 E 0501 E \mathrm{E} 0017 F 27 C F$ 1036CEFEDCEC2EOEFE7EE045EE0300E0000EO2EOEFFOOEO20CE0100ECO607F243F ， 382 EFEDEE 16 O6E31E0EFEE2E33EOEFEEGEO72OEFEDEE31EDEFEE4E1E177F1E6F 10398E1D17E0300EGOO2EC820EEFF2EFE7CECD60EFEE8E1604E06AOCOS087F249F OUAEE0460CO4DEECO60EFEEAE1604E06ADCO508EO460COSO2ECOADEFEDA7F231F IOЗC4E6OAOEFETCE1903EO4EOEFEEAE1GF 4EA802EFE92ECOAOEFE92EG7427F1F2F 103DAE0282E0000B1502E04E0EFE92EC060EFEFAEA060EFE7CE0821EC8017F239F IO3FOEFE9AEC820EFE7CEFE9AECO6OEFE9AE6060EFE78EC801EFE9EEC0607F18EF 10406EFE9AEA060EFE76E6060EFE78E6060EFE78BC801EFEA2E06AOC011C7F201F 041 CEO OAOCO2AAECO6OEFEDCE1308E1107E3860EFE74E3C60EFE72EOA127F233F O432E1701E0581EC801EFE70ECO60EFE70B1108B3860EFE6EE3C60EFE6C7F238F 0448E0A12E1701E0581E1009E0501E3860EFE6EE3C60EFE6CEOA12E17017F293F O45EEOS81EO501EC801EFE6AEC820EFE6AEEOD2ECO6OEFEDCE6060EFE947F200F 0474 EC 01 EFE 7 AE 06 A 0 CO 160 EC 220EFE96EFE98EC82GEFE94EFE96EC8207F1E6F Q48AEFEDCEFE94EC820EFEF 6EFEF8EC820EFEF 4EFEF 6EC820EFE7CEFEF47F10EF O4AOEOAEOEFEFAEA82OEFEF AEFEFAEA82日EFEF $6 E F E F A E A 82 O E F E F 8 E F E F A 7 F 11$ GF

04E6EC820EFE78EFE76EC820EFF9AEFE78EC820EFEC6EFEEEEC820EFEC87F155F 04CCEFECOBCB20EFECAEFEC2EC820EFECCEFEC4EG4E0EFEDAEC120EFE7C7F12AF 04E2E1102E04C3E1001E0703EC0A0EFED8ECO60EFED6E06A0C0556E04607F272F 04F8C03CAECB01E:FED6EC802EFED8E:02E0E:FF8AE0380EC060EFEDCE130E7F1E:8F 0S0EE1105E6060EFEE6E1706E04C1E1004EA060EFEE6E1701E04C1EC8017F27AF 0524 EE 002 ECBO 1 EFEDCE 045 EE 0541 EOSO 2 E 1703 E 0581 E 1901 E 045 EE 022 E 7 F 2 A 4 F 053 AE 0004 E 045 EE 0541 E 0542 E 0503 E 1705 E 0582 E 1703 E 0581 E 1901 E 045 E 7 F 302 F $0550 E 022 E E 0004 E: 04 E E A G 84 E 1703 E 0581 E 1901 E 1005 E A 043 E 1901 \mathrm{E} 10027 F 366 F$ $0566 \mathrm{E} 02 \mathrm{EE} 0004 \mathrm{E} 045 \mathrm{ER} \mathrm{A} 日 \mathrm{C} 6 \mathrm{E} 1705 \mathrm{E} 0582 \mathrm{E} 1703 \mathrm{E} 0581 \mathrm{E} 1901 \mathrm{E} 100 \mathrm{AEA0B5} 7 \mathrm{~F} 2 \mathrm{D} 2 \mathrm{~F}$ 057CE1703E0581E1901E1005EA044E1901E1002E022EE0004E045EEC80E7F2E0F 0592C05D6EC041E1507E1103E04C1E04C2E1013E0704E0501E1001E04C47F2F4F 05ABECOC3E1505E1101E10FSE0705E0503E1001E04CSE3843E6105E13047F2E9F 05EEE06A0C052CE0460C05D0EC2E0C05D6E022EE0004E045EEC2E0C05D67F267F 05D4E045EE0000EC80EEFEFGEC041E150AE1103EC082E1324E1006E07067F295F 05EAE06A0C052CE0460C063EE1001E04C6EC145E1505E1101E1318E07077F2C7F O600E0505E1001E04C7ECOC2B38C5E3845EA083E1703E0581E1901E10147F2DDF $0616 \mathrm{ECOC4E} 6187 \mathrm{E} 1304 \mathrm{E} 06 \mathrm{~A} 0 \mathrm{CO} 03 \mathrm{EE} 0460 \mathrm{CO} 3 \mathrm{EEC2E} 0 \mathrm{EFEF} 8 \mathrm{E} 022 \mathrm{EE} 00047 \mathrm{~F} 272 \mathrm{~F}$ $062 C E 045 B E 04 C 1 E 04 C 2 E 04 C 3 E C 2 E 0 E F E F 8 E 022 E E 0004 E 045 E E C 2 E 0 E F E F 87 F 216 F$ 0642E045E7FD9FF
O374LOOFINS0000MATIN 500ACWAIT3 6FE92ERFK 6FEDCUK 7F2EEF
FE94UK1 GFE9AVO1K GFE9CVO2K GFE9EAO1K GFEAQAO2K 7F325F

FE78V01K1 FEA4DAO2K FEC6W11KF FEABK 1 W 2 K FEE2K2W1K FEECK2UK3 FEECK $\cup \cup O K$ FEE2MASK0 E002DAC1 FEBAKAOK FF10WORK556FEF2MESSAG6FED6ACFOS16FED8ACFOS26FE7ESALIN17F154F FEFCSALIN26FEFESALIN46FE7CDXK 6FEF4DXK1 6FEF6DXK2 7F234F FEF8DXK3 6FE7ADUK 6FEFAXKMXK46FE74KCOFF16FE72KCORF27F1E9F FEGEKUCOR16FEGCKUCOR2GFETOUKCORFGFEGAUKOUT SOQAECOMMUNTF110F $1000 M 1 D C F K 61002 M 1 F E F G 61004 M 1 F E F 61006 M 1 A C F 161008 M 1$ ACF $27 F 320 F$ 100AM1ERFO6100CM1UK $61014 \mathrm{M} 1 \mathrm{MESSG100EM1VO1K61010M1A01K7F2C4F}$ $1012 \mathrm{MDAO} \mathrm{K} 50118 \mathrm{FROCESSO11CDOES} 1 \mathrm{SO590SSMUL}$ S0508EMDECETF288F 0160DOES2 50556DEADD 502AADUCONT5052CTWOCF25053ETWOCF37F259F O56CTFADD 505D8DSMU1 7FA89F

MOD2F'4 01/01/00 00:09:39 TXLINK 2.3.0

## APPENDIX 10.1

CONTROL FUNCTION TO VELOCITY TRANSFER FUNCTION ANALYSIS

The transfer function analysed in this Appendix is

$$
\frac{s y(s)}{u(s)}=\frac{1}{(1+s T)\left(\frac{s^{2}}{w^{2}}+\frac{2 \rho}{w} s+1\right)}
$$

making $s y(s)=v(s)$, the time response for a unit step input on the command function $u$, or

$$
\begin{aligned}
& u(t)=0 \text { for } t<0 \\
& u(t)=1 \text { for } t>0
\end{aligned}
$$

can be found as ( $t \geqslant 0$ ).

$$
\begin{aligned}
v(t)= & 1-\frac{e^{-\rho w t}}{\beta \rho^{2}(\beta-2)+1}\left[\beta \rho^{2}(\beta-2) \cos \left(\sqrt{1-\rho^{2}} w t\right)+\right. \\
+ & \left.\frac{\beta \rho\left[\rho^{2}(\beta-2)+1\right]}{1-\rho^{2}} \sin \left(\sqrt{1-\rho^{2}} w t\right)\right]- \\
& \frac{e^{-\frac{w t}{\alpha}}}{\beta \rho^{2}(\beta-2)+1}
\end{aligned}
$$

where

$$
\alpha=w T
$$

and $\quad \beta=\frac{1}{\rho \alpha}$

Changing the time scale to $t^{*}=w t$, equation 2 can be solved for different values for $\rho$ and $\alpha$. Figure 1 shows the percentage overshoot (PO) function of $\alpha$ with $\rho$ as a parameter. Figures 2 to 8 show the normalised step response for different pairs $\rho$ and $\alpha$.


Figure 1. Percentage Overshoot $\mathrm{x} \alpha$








Figure 8 Normalized Time Response

DISCRETE STATE EQUATIONS FOR EXTENDING DIRECTION
ENTER NUMBER OF STATES $=4$
ENTER MATRIX A BY LINES
$\theta, 1, \varnothing, \varnothing$
$0,0,1,0$
$\theta, 0,0,1$
Ø，－2，－1．28，－2． 14
ENTER MATRIX B BY COLUMN
$\varnothing, \varnothing, \varnothing, 2$
ENTER SAMPLING PERIOD $T(M S)=1 \emptyset \emptyset \emptyset$
ENTER DELAY TD（MICROSECOND）$=1 \varnothing \varnothing \emptyset \emptyset \emptyset \emptyset ~$
MATRIX PHI

| $1 . \square 00 \square$ | $\emptyset .9451$ | 0.4531 | 0.0984 |
| :---: | :---: | :---: | :---: |
| ø． 0 ¢øб | 0.8033 | 0.8192 | 0.2426 |
| Ø． 0 ¢øб | － 0.4853 | 0.4927 | 0.3000 |
| Ø． 0000 | －0．6000 | －0．8693 | －0．149 |

MATRIX L1
MATRIX L2

| 0.0549 | の． $00 \square 0$ |
| :---: | :---: |
| 0.1967 | $0.00 \varnothing 0$ |
|  | Ø．0のØロ |
| 0.4853 |  |
| 0.6000 | Ø．ØØロロ |

MATRIX PHIT2

| $1.90 \square \square$ | 0.9451 | 0.4531 | 0.0984 | 0.0549 |
| :---: | :---: | :---: | :---: | :---: |
| Ø．Øøøø | 0.8033 | 0.8192 | Ø． 2426 | Ø． 1967 |
| Ø．Øøøø | －Ø． 4853 | 0.4927 | 0.3009 | 0.4853 |
| Ø．ØØØロ | $-0.600 \square$ | －0．8693 | －Ø． 1493 | 0.6000 |
| Ø．Øøøø | Ø． 0 のøø | $0.000 \square$ | $0.00 \square 0$ | 0.0000 |

MATRIX L

> 0.0000
> 0.0000
> 0.0000
> 0.0090
> 1.0000

## DISCRETE STATE EQUATIONS FOR RETRACTING DIRECTION

ENTER NUMBER OF STATES $=4$
ENTER MATRIX A BY LINES
$\theta, 1, \theta, \theta$
$0,0,1,0$
$\varnothing, \varnothing, \varnothing, 1$
Ø，－2，－1．52，－2． 26
ENTER MATRIX B BY COLUMN
$\varnothing, \varnothing, \varnothing, 2$
ENTER SAMPLING PERIOD $T(M S)=1 \emptyset \emptyset \emptyset$
ENTER DELAY TD（MICROSECOND）$=1 \varnothing 0 \emptyset \emptyset \emptyset \emptyset$
MATRIX PHI

| $1.0 \emptyset \square \square$ | 0.9464 | 0.4478 | 0.0953 |
| :---: | :---: | :---: | :---: |
| 0.9000 | 0.8093 | 0.8015 | 0.2323 |
| 0.0000 | －0． 0645 | 0.4563 | 0.2766 |
| $0.000 \square$ | $-0.5532$ | $-\varnothing .8850$ | － 0.1688 |

MATRIX LI

| 9.0536 | Ø．0000 |
| :---: | :---: |
| 0.1907 | Ø． $0 \varnothing \square \varnothing$ |
| $\emptyset .4645$ | Ø．ØøØワ |
| 0.5532 | $\emptyset . \emptyset \varnothing \emptyset \emptyset$ |

MATRIX PHIT2

| $1.000 \square$ | 0.9464 | 0.4478 | 0.0953 | 0.0536 |
| :---: | :---: | :---: | :---: | :---: |
| Ø．00øロ | 0.8093 | 0.8015 | Ø． 2323 | 0.1907 |
| Ø． $00 \square \square$ | －0． 0.4645 | 0.4563 | 0.2766 | 0.4645 |
| $0.00 \square \square$ | －0． 5532 | $-0.8850$ | －0．1688 | 0.5532 |
| Ø． 0000 | 0.0000 | $0 . \square \square 0 \square$ | Ø． 0090 | Ø．000ワ |

MATRIX L
$0.000 \square$
0.0090

0.9000
1.0000

## APPENDIX 10.4

## SAMPLE OF THE SWEEP PROGRAM

ENTER ORDER $N=4$
ENTER MATRIX PHI BY LINE
ENTER MATRIX L BY COLUMN
ENTER MATRIX QI BY LINE
$1, \theta, 0,0$
Ø, $0,0,0$
$0,0,0,0$
Ø. $\varnothing, \varnothing, \varnothing ~$
ENTER VALUE Q2
.01
ENTER NUMBER OF STEPS $=3 \varnothing$

| 0.0000 | 0.0000 | 0.0090 | 0.0000 |
| :---: | :---: | :---: | :---: |
| 0.6340 | 3.3265 | 1.5238 | 2.6945 |
| -0.2529 | 1.2777 | 0.7089 | 1.5196 |
| -0.3875 | 1.8768 | 0.6400 | 1.4452 |
| -0.3867 | 1.0831 | 0.6430 | 1. 4505 |
| -Ø.388Ø | 1.8799 | 0.6417 | 1.4485 |
| -0.3882 | 1.0795 | 0.6415 | 1.4483 |
| -ø. 3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.8795 | 0.6415 | 1.4483 |
| -0. 3882 | 1.8795 | 0.6415 | 1.4483 |
| -0. 3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.0795 | 9. 6415 | 1.4483 |
| -0. 3882 | 1.8795 | 0.6415 | 1.4483 |
| -0.3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.9795 | 0.6415 | 1.4483 |
| -0.3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.8795 | 0.6415 | 1.4483 |
| -0.3882 | 1.8795 | 0.6415 | 1.4483 |
| -0. 3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.6795 | 0.6415 | 1.4483 |
| -0.3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.0795 | 0.6415 | 1.4483 |
| -Ø. 3882 | 1.0795 | 0.6415 | 1.4483 |
| -0.3882 | 1.0795 | 0.6415 | 1.4483 |
| -0. 3882 | 1.0795 | 0.6415 | 1.4483 |
| -0. 3882 | 1.0795 | 0.6415 | 1.4483 |
| -ø. 3882 | 1.0795 | 0.6415 | 1.4483 |
| -ø. 3882 | 1.0795 | 0.6415 | 1.4483 |

CHANGE Q1(1),CHANGE Q2(2),STOP(3)

ок,

## APPENDIX 10.5

OBSERVER POLE-PLACEMENT DESIGN
cylinder extending
ENTER NUMBER OF STATES $=2$
ENTER SAMPLING PERIOD (SEC) $=1$
ENTER MATRIX PHII BY LINE
. 4927, -. 8693
. 3,-. 1493
ENTER MATRIX GAMMA BY COLUMN
. 8192,. 2426
ENTER DESIRED POLE S-PLANE REAL= $-2000 \varnothing$ IMAG $=\varnothing$

ENTER DESIRED POLE S-PLANE REAL= -2ØØØØ IMAG $=\varnothing$

> K
> 0.7107
> -0.9845
**** STOP
OK,

```
cylinder retracting
    ENTER NUMBER OF STATES = 2
    ENTER SAMPLING PERIOD(SEC) = 1
    ENTER MATRIX PHII BY LINE
    .4563,-. 885
    . 2766, -. 1688
    ENTER MATRIX GAMMA BY COLUMN
    .8015,. 2323
    ENTER DESIRED POLE S-PLANE REAL= -2Ø\emptyset\emptyset\emptyset
    IMAG= Ø
    ENTER DESIRED POLE S-PLANE REAL= -2øø\emptyset\emptyset
                                    IMAG= }
```

                K
            0.6623
            -1. 0477
    OK,

## MATRIX COEFFICIENTS FOR OBSERVER EQUATIONS

```
cylinder extending
    ENTER SYST ORDER= 4
    ENTER OBSERVER ORDER= 2
    ENTER ORDERED SYSTEM MATRIX BY LINE
    .8033,.1967,.8192,.2426
    0,0,0,\varnothing
    -.4853,.4853,.4927,. 3
    -.6,.6,-.8693,-. 1493
    ENTER ORDERED INPUT MATRIX BY COLUMN
    Ø,1,0,0
    ENTER OBSERVER FEED GAINS BY LINE
    .7107.0
    -.9845,0
    PBB-LO*PAB
    -0.0895 0.1276
    -0.0628 Ø.0895
    (PBA-LO*PAA)+(PBB-LO*PAB)*LO
    -1.2454 Ø. 3455
        0.0581 0.7937
    LB-LO*LA
        0.0000
        0.0000
cylinder retracting
    ENTER SYST ORDER= 4
    ENTER OBSERVER ORDER= 2
    ENTER ORDERED SYSTEM MATRIX BY LINE
    .8093,.1907,.8015,.2323
    0,0,0,0
    -.4645,.4645,.4563,.2766
    -.5532,.5532,-.8850,-. 1688
    ENTER ORDERED INPUT MATRIX BY COLUMN
    \emptyset,1,0,\varnothing
    ENTER OBSERVER FEED GAINS BY LINE
    .6623,0
    -1.0477.0
    PBB-LO*PAB
        -Ø.0745 0.1227
        -Ø.0453 Ø.9746
(PBA-LO*PAA)+(PBB-LO*PAB)*LO
    -1.1785 0.3382
        \emptyset.1866 0.7530
LB-LO*LA
    0.øøøø
    0.0000
```

APPENDIX 10.7

ESTIMATED VARIABLES
OOBS 1

TXMIFA 2.3.0 78.244 00:02:09 01/01/00
F'AGE 0001 Calculating observer variables


TXMIFA
2.3.078.244 00:02:09
$01 / 01 / 00$
PAGE
0002 CALCULATING OESERUER UARIAELES

| $E$ | A01K | OO1E | E | A02K | 0000 | E | DAO1K | 0038 | E | DA02K | 0000 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| E | DE:ADD | 0000 | D | DOES 1 | 0000 | E | EMDECE | 002E | E | K3VOK | 0000 |
| $E$ | K4VOK | 0026 |  | LINK | O日GE: |  | F0 | 0000 |  | F1 | 0001 |
|  | R10 | 000A |  | R11 | 000E |  | F12 | O00C |  | R13 | 0000 |
|  | Fi14 | QOOE |  | R15 | 000F |  | Fi2 | 0002 |  | F3 | 0003 |
|  | F4 | 0004 |  | F5 | 0005 |  | R6 | 0006 |  | R7 | 0007 |
|  | Fi8 | 0008 |  | F'9 | 0009 | , | FETUR | 003A | E | SALIN2 | 003C |
| E | SSMUL | 002A | E | V01K | 0022 | E | V02K | 0000 | E | W11K | O01A |
| E | W12K | 0000 | E | W21K | 0034 | $E$ | W22K | 0000 |  |  |  |

0000 ERRORS

CONTROL EQUATION

| TUCONT | TXIFA | 2.3 .0 | 78.244 | $00: 02: 57$ | $01 / 01 / 00$ | FAGE OQO1 |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: |
| CONTKOL FUNCTION－FOSITION FORM |  |  |  |  |  |  |

0001
0002
0003
0004
0005
0006
9007
0008
0009
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024 0000 C80E
00020000
0025
0026
0027
00280004 C060
0006 0000
00290008 COED
OQDA ODOD
OO3O OOOC OGAO
OOOE 0000
003100100460
00120000
0032 0014 02E0
00160000
0033
0035
00360018 C060
001A 0000
0037 OOIC COEO
001E 0000
0038 0020 日6A0
0022 OODE＇
0039.00240460
$00260012^{\prime}$
00400028 02E0
002A 0000
0041
0042
0043
0044 002C 2060

＊＊＊＊＊＊＊＊＊
＊STAFT＊
＊＊＊＊＊＊＊＊＊
DUCONT MOV Fi11，巴SALIN1 SAVE FEETUFN
＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊K＊E（K）＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊
MOU EEFFK，Fi1 E（K）
MOU QKEFFK，R3 CTE EIS
EL セSSMUL MTFY
E＠EMDECE
LWFI WORK22 MOUE FOINTER
＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊K＊U（K－1）＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊
MOU＠UK1，R1

MOU＠KUK1，F33 CTE
EL＠SSMUL
E＠EMDECE
LWFI WORK 33
MQUE FQINTER
＊K＊U／W＊
＊＊＊＊＊＊＊＊＊＊＊＊＊＊
MOU＠VOIK，R1



| E | A01K | 0042 | E | A02K | 0000 | , | CACA | 009C | E | DAO1K | 0056 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| E | DAD2K | 0000 | E | DEADD | 008E | E | DSMU1 | 0000 | D | DUCONT | 0000 |
| E | EMDECE | 0092 | E | ERRK | 0006 | E | KAOK | 0046 | E | KDAOK | O05A |
| E | KEFRK | OODA | E | KFMAXA | 00E:2 | E | KFMAXE: | OOES | E | KUK1 | 001 E |
| E | KUK2 | 0000 | E | KUK3 | 0000 | E | KVOK | 0032 | , | NEGAT | GOAC |
| , | POSIT | 00A8 |  | Fio | 0000 |  | R1 | 0001 |  | F10 | OOBA |
|  | R11 | 000B |  | R12 | 000c |  | F13 | OQOD |  | F14 | DODE |
|  | F15 | 000F |  | R2 | 0002 |  | R3 | 0003 |  | F4 | 0004 |
|  | Fi5 | 0005 |  | F6 | 0006 |  | F7 | 0007 |  | F8 | 0008 |
|  | K9 | 0009 | ' | FESUL 1 | O日AD | $E$ | SALIN1 | 00 c 6 | , | SAV | OOC0 |
| , | SCA | OQEO | E | SSMUL | $005 E$ | E | UK | Q日c2 | E | UK1 | 001A |
| E | UK2 | 0000 | E | UK3 | 0000 | E | V01K | 002E | E | VO2K | 0000 |
| E | WORK1 | 008A | $E$ | WOFK22 | Q07E | E | WORK33 | 0072 | E | WOFK44 | 0066 |
| E | WORK55 | 0052 |  |  |  |  |  |  |  |  |  |

## APPENDIX 10.9

## OBSERVER EQUATIONS

$$
01 / 01 / 00
$$

FAGE 0001 ORSERUER EQUATIONS

| 0001 | $* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ~$ |
| :--- | :--- | :--- |

MOU @UK1, F:3
EL DSSMUL
B ©EMDECE
LWFI WORK33

BL. @DEADD
B @EMDECE
LWFI WOFK22
EL ©DEADD
B REMDECE
LWFI WOFK1
EL ©DEADD

E @EMDECE
SLA K1, 1
SLA Fi2, 1
JOC CACA1
JMF CACAS
AI Ki, 1

MOV Kil, Fil
JGT FOSIT1
JLT NEGAT1
JMF SAU1
FOSIT1 CLF FSS
JMF SCA1
NEGAT1 SETO FS
NEG Fil
*FESULT*SCALING AND FFOF. GAIN
SCA1 MOU GKOEIA,FK
MF'Y FB, Fi1
DIU RKOBIE, R1
MOU $F S, F E$
JEQ SAVI
NEG Fil
MOU Fi, WW11KF
************

* W2(K+1) *
************
MOU @K2W1K, 下ろ3

| 0076 | D0ab | C060 | MOV | CW11K，K1 |
| :---: | :---: | :---: | :---: | :---: |
|  | DOAA | BROA＇ |  |  |
| 0077 | goac | 06A0 | EL | ＠SSMUL |
|  | D日AE | 004A＇ |  |  |
| 0078 | 00E0 | 0460 | E． | ＠EmDECE |
|  | 00E2 | $0072{ }^{\prime}$ |  |  |
| 0079 | 00E4 | 02E0 | LWFI | WORK22 |
|  | 00E6 | 005E＇ |  |  |
| 0080 | 00 E 8 | COEO | MOU | ＠к2W2K，R3 |
|  | ODEA | 0000 |  |  |
| 0081 | O日EC | c060 | MOV | ＠W21K，R1 |
|  | QOEE | $001 E^{\prime}$ |  |  |
| 0082 | 日日c0 | O6A0 | ELL | ＠SSMUL |
|  | 00c2 | 00aE＇ |  |  |
| 0083 | 00c4 | 0460 | E | ＠EmDECE |
|  | 日0C6 | 00E2＇ |  |  |
| 0084 | 0ecs | 02E0 | LWFI | WORK33 |
|  | OOCA | 0052＇ |  |  |
| 0085 | O日CC | C060 | mov | ＠к2UK，R1 |
|  | O日CE | 0000 |  |  |
| 0086 | O日DO | COE0 | MOV | ＠VO1K，F3 |
|  | 00D2 | $0032^{\prime \prime}$ |  |  |
| 0087 | 00D4 | OSAO | EL | ©SSMUL |
|  | 0006 | gecz＇ |  |  |
| 0088 | 90D8 | 0460 | E | ＠EMDECE |
|  | GEDA | 00C6＇ |  |  |
| 0089 | OODC | g2E0 | LWF＇I | WOEK44 |
|  | OODE | 003E＇ |  |  |
| 0090 | G0E0 | C060 | MOV | ＠R2UK1，F1 |
|  | O日E2 | 0000 |  |  |
| 0091 | 00E 4 | COEO | MOV | ＠UK1，\％3 |
|  | 00E6 | 0046 |  |  |
| 0092 | 00E8 | 66A0 | EL． | ＠SSMUL |
|  | ODEA | 00D6＇ |  |  |
| 0093 | O日EC | 0460 | E | ＠EMDECE |
|  | OQEE | goda＇ |  |  |
| 0094 | G日Fb | 02E0 | LWF＇I | WORK33 |
|  | 00F2 | O日CA＇ |  |  |
| 0095 | Q日F4 | OGAO | ELL | CDEADD |
|  | Q6F6 | OB6E＇ |  |  |
| 0096 | 0日F8 | 0460 | E | ＠EMDECE |
|  | OOFA | O日EE |  |  |
| 0097 | OQFC | 02E0 | LWFFI | WORK22 |
|  | Q0FE | 00E6＇ |  |  |
| 0098 | 0100 | 06A0 | EL | CDEADD |
|  | 0102 | 00F6＇ |  |  |
| 0099 | 0104 | 0460 | E | ＠EmDECE |
|  | 0106 | 日6FA＇ |  |  |
| 0100 | 0108 | 02E0 | L．WF＇I | WORK1 |
|  | 010A | 006A＇ |  |  |
| 0101 | 010c | OSAO | ELL | CDEADD |
|  | 010E | 0102 |  |  |
| 0102 | 0110 | 0460 | E | ＠EMDECE |
|  | 0112 | $0106^{\prime}$ |  |  |


| 0103 | 0114 | $0 A 11$ |
| :--- | :--- | :--- |
| 0104 | 0116 | $0 A 12$ |
| 0105 | 0118 | 1801 |
| 0106 | $011 A$ | 1002 |
| 0107 | $011 C$ | 0221 |
|  | $011 E$ | 0001 |
| 0108 | 0120 | $C 041$ |
| 0109 | 0122 | 1502 |
| 0110 | 0124 | 1103 |
| 0111 | 0126 | $100 C$ |
| 0112 | 0128 | $04 C 5$ |
| 0113 | $012 A$ | 1002 |
| 0114 | $012 C$ | $\theta 705$ |
| 0115 | $012 E$ | 0501 |
| 0116 |  |  |
| 0117 | 0130 | $C 0 E 0$ |
|  | 0132 | 0000 |
| 0118 | 0134 | 3843 |
| 0119 | 0136 | $3 C 60$ |
|  | 0138 | 0000 |
| 0120 | $013 A$ | $C 145$ |
| 0121 | $013 C$ | 1301 |
| 0122 | $013 E$ | 0501 |
| 0123 | 0140 | $C 801$ |
|  | 0142 | 0000 |
| 0124 | 0144 | $C 2 E 0$ |
|  | 0146 | 0002 |
| 0125 | 0148 | $045 E$ |


|  | SLA | F1， 1 |  |
| :---: | :---: | :---: | :---: |
|  | SLA | K2， 1 |  |
|  | JOC | CACA3 |  |
|  | JMF | CACA4 |  |
| CACA3 | AI | F1， 1 |  |
| CACA4 | MOU | F1，Fil |  |
|  | JGT | POSIT2 |  |
|  | JLT | NEGAT2 |  |
|  | JMF | SAV2 |  |
| FOSIT2 | CLF | R5 |  |
|  | JMF | SCA2 |  |
| NEGAT2 | SETO | FS |  |
|  | NEG | R1 |  |
| ＊FESULT | ＊SCA | ING AND FFROF． | GAIN |
| SCA2 | MOU | ＠KOE2A， F 3 |  |
|  | MPY | F3， F 1 |  |
|  | DIU | ＠KOE2E， F 1 |  |
|  | MOV | FS，FS |  |
|  | JEQ | SAVZ |  |
|  | NEG | F1 |  |
| SAVZ | MOV | F1，以W21KF |  |
|  | MOV | ＠SALIN4，F11 |  |
|  | E | ＊F11． |  |
| ＊＊＊＊ |  |  |  |
|  | END |  |  |


| ， | cacal | 0075 | ， | cacas | 0080 | ， | cacaz | 011 C |  | cacal | 0120 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| E | DEADD | 010E | D | DOES2 | 0000 | E | DSmu1 | 0000 | E | EMDECE | 0112 |
| E | K1UK1 | 0042 | E | K1UK2 | 0000 | E | k1uk3 | 0000 | E | K1UK | 002 E |
| E | K1W1K | 0006 | E | K1W2k | B01A | E | K2Uk 1 | 00e2 | E | K2Uk2 | 0000 |
| E | K2UK3 | 0000 | E | kauk | goce | E | K2W1K | 00Ab | E | к2W2k | OOEA |
| E | KOE：A | 0092 | E | KOE1E： | 0098 | E | koe：2a | 0132 | E | KOE2E： | 0138 |
| ， | NEGAT1 | 008C | ， | NEGAT2 | 012 C | ， | FOSIT1 | 0088 |  | FOSIT2 | 0128 |
|  | F 0 | 0000 |  | F1 | 0001 |  | R10 | 000A |  | F11 | O日GE |
|  | F12 | 000c |  | F13 | 0000 |  | F14 | 000e |  | R15 | 000F |
|  | R2 | 0002 |  | F3 | 0003 |  | K4 | 0004 |  | FS 5 | 0005 |
|  | Fi6 | 0006 |  | F7 | 0007 |  | F8 | 0008 |  | F9 | 0009 |
| E | SALIN4 | 0146 | ， | Savi | OQAE | ， | Savz | 0140 | ， | SCA1 | 0090 |
|  | SCA2 | 8130 | E | SSMUL | OOEA | E | UK1 | B0E6 | E | UK2 | 0000 |
| E | UK3 | 0000 | E | voik | 0002 | E | vo2k | 0000 | E | W11K | DgAA |
| E | W11 KF | $00 \mathrm{A2}$ | E | W12k | 0000 | E | W12KF | 0000 | E | W21K | GOEE |
| E | W21kF | 0142 | E | W22K | 0000 | E | W22KF | 0000 | E | WORK 1 | 010A |
| E | WOFK22 | 00FE | E | WORK33 | OOF2 | E | WORK44 | O日DE | E | WORKS | 0000 |

DODO ERRORS


[^0]:    APPENDIX 9.14

