OPTIMAL ELECTRO-HYDRAULIC CONTROL OF MINING BOOMS

TO MY PARENTS

OPTIMAL ELECTRO-HYDRAULIC CONTROL OF MINING BOOMS

by

Chun Wah CHUEN, AP(HK), MSc, AMIMechE

A thesis submitted for the degree of

Doctor of Philosophy

in

Mechanical Engineering

at

THE UNIVERSITY OF ASTON IN BIRMINGHAM

May 1983

OPTIMAL ELECTRO-HYDRAULIC CONTROL OF MINING BOOMS

by

Chun Wah CHUEN

PhD 1983

SUMMARY

Mining boom rippers are used in the UK coal industry to drive tunnels, which provide ventilation and access to the coalseams. The present tunnelling machines are open-loop controlled; a better positional control can be obtained by introducing feedback, and a microprocessor based system provides automatic operation.

A dual microprocessor system based on the 8085A 8-bit microprocessor has been built and tested on a boom ripper. Poppet valves are used in this system for their ease in interfacing to the microprocessor and their high dirt tolerance. A bang-bang (on-off) control philosophy is adopted to provide the optimal positional control: the valves are fully opened to give the boom the maximum acceleration moving towards the desired position and they will be closed completely to provide the maximum deceleration when a pre-determined criterion has been satisfied.

Two methods in providing the switching signals have been studied: the signals are determined by simply comparing the desired and the current boom positions in one method, or they can be determined by evaluating the switching function which accounts for the valve characteristics and other system properties. Both methods give a positional accuracy limited by the ADC (approximately ±1 cm at the cutter end, ±0.4% of the full range). However, the switching function evaluation method shortens the overall time to change the boom positions at high speed, while increasing the complexity in the circuit and software design.

It has been demonstrated that an APU can increase the speed and simplify the software in handling mathematical operations. An APU board has been developed, although it is not used in the simple comparison method. The APU will be particularly useful if the system is going to compensate for the machine chassis movement when the signals for the machine's position and orientation can be obtained.

Keywords.

Microprocessor Application.
Mining Boom Ripper Positioning.
Bang-Bang (On-Off) Control.
Optimal Switching.

ACKNOWLEDGEMENTS

The author wishes to thank all persons for their assistance in carrying out this project. The varied nature of the matters dealt with has entailed references to many sources, including books, journals, proceedings, learned societies and business concerns, and to all of these the author gladly acknowledges his indebtedness for the information and assistance they have provided.

In particular, the author gladly expresses his sincere gratitude to his supervisor, Professor K Foster, for providing the opportunity to study this subject, for his invaluable help, suggestions, encouragement and guidance.

The author is also indebted to Mr G A Jones for his permission to use the equipment in developing the microprocessor system, to Mr J H Knight for his companionship in this project and help in building the microprocessor hardware, to Mr G Seet for his help in reading the experimental results, to Mr J Leahy and Mr J D Kibble of the National Coal Board for sponsoring this project.

It is also a pleasure to acknowledge the assistance of various members of the staff at The University of Aston. The author is grateful to the following technical personnel in the Department of Mechanical Engineering: Mr A G Evitts, Mr T Horton, Mr A Jones and Mr N Moss.

The author is indebted to Miss P A Blower for carefully and patiently typing this thesis. He also wants to thank The University of Aston in Birmingham for the financial support in the form of a research studentship award.

C O N T E N T S

	Page
SUMMARY	i
ACKNOWLEDGEMENTS	ii
LIST OF CONTENTS	iii
LIST OF FIGURES	viii
LIST OF TABLES	xii
NOTATION	xiii
SUBSCRIPTS	xvi
NOTE ON THE LAYOUT OF THESIS	xvii
CHAPTER ONE: INTRODUCTION	
1.1 Coal Mining Operations	2
1.2 Tunnelling	4
1.3 Stiffness of a Mining Boom Ripper	8
1.4 Control Principles	10
1.5 Microprocessor Control	12
1.6 Organisation of Thesis	13
CHAPTER TWO: BRIEF REVIEW OF PAST WORK DONE IN APPLYING THE CONTROL THEORIES TO A HYDRAULIC SYSTEM	
2.1 Conventional Control	16
2.2 Optimal Control by Closing of Valves	24
2.3 Application to the Current Project	27
CHAPTER THREE: DERIVATION OF THE SWITCHING FUNCTIONS	
3.1 Introduction	29
3.2 Mathematical Model	31
3.3 Reference Quantities for Non-dimensionalisation	
3.3.1 Reference Position	34
3.3.2 Reference Length	35
3.3.3 Reference Area	35
3.3.4 Reference Time	35

CONTENTS (Cont)

		Page
	3.3.5 Reference Pressure	36
	3.3.6 Reference Flow Rate	37
3.4	Non-dimensionalised Equations	37
3.5	Possibility for Cavitation to Occur	38
3.6	Derivation of the Switching Function for a Symmetric System	
	3.6.1 Analysis	40
	3.6.2 Ideal On-Off Valves	42
	3.6.3 On-Off Valves with Time Delay	43
	3.6.4 Linearly Operated Valves	45
	3.6.5 Effects of Externally Applied Load	47
3.7	Derivation of the Switching Function for an Asymmetric System	48
3.8	Simulation Studies	50
CHAPTER FOUR: ARITHMETIC PROCESSING		
4.1	Introduction	63
4.2	Function to be Evaluated	63
4.3	Timing Test Program	64
4.4	Explanation of the Test Program	
	4.4.1 Set Up the 'Arithmetic Processor'	66
	4.4.2 Timing	66
	4.4.3 Format Conversion	67
4.5	Binary Floating Point Format	67
4.6	Operation Required	
	4.6.1 FPAL	72
	4.6.2 APU	72
4.7	Results	74
4.8	Boom Moving Rate	74
4.9	Comments	76

CONTENTS (Cont)

Page

CHAPTER FIVE: CONTROL ALGORITHMS AND PROGRAM STRUCTURE

5.1 Introduction 79

5.2 General Structure of the Control Algorithm 83

5.3 Initialisation 84

5.4 Profile Trimming 86

5.5 Cavity Cutting 90

5.6 Boom Moving 94

5.7 Sorting and Expanding the Profile Data 96

5.8 Routine for Moving the Boom Cutter Slowly --- 'SMOVE" 101

5.9 Mode Changing 102

CHAPTER SIX: CONTROL CIRCUIT DESIGN

6.1 Introduction 104

6.2 Main Microprocessor and Monitor Board 107

6.2.1 Clock Generation 107

6.2.2 $\overline{\text{RESET IN}}$ and Power On 109

6.2.3 Address Decoding 109

6.2.4 First Monitor and I/O Ports 110

6.2.5 Low Order Address Latching 110

6.2.6 Buffers 111

6.2.7 Others 113

6.3 Expansion RAM and EPROM Board 115

6.4 Arithmetic Processing Unit Board 118

6.5 Interface to the Data Capturing System 120

CONTENTS (Cont)

Page

CHAPTER SEVEN: TESTS AND RESULTS

7.1 Introduction 124

7.2 Control Strategies

7.2.1 Simple Comparison with Slow Speed 124

7.2.2 Switching Function Evaluation 125

7.3 Boom Moving at High Speed

--- 'FMOVE' Routine 127

7.4 Test Results

7.4.1 Boom Control by Simple Comparison with Speed Limitation 133

7.4.2 Boom Control with Switching Function Evaluation 135

CHAPTER EIGHT: CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions 145

8.2 Recommendations 147

8.3 Suggestions for Further Work 148

REFERENCES 151

APPENDICES

APPENDIX A Derivation of Switching Function for On-Off Valves with Time Delay 158

APPENDIX B Derivation of Switching Function for Linearly Operated Valves with Operating Time t_c 161

APPENDIX C Computer Programs for the Simulation Studies 165

APPENDIX D Stiffness Analysis of a Mining Boom 184

CONTENTS (Cont) Page

APPENDIX E	Listing of the Assembly Programs for Testing the Time Taken to Evaluate the Simplified Switching Function Using Different Methods	193
APPENDIX F	Function Approximations Using the Chebyshev Series	216
APPENDIX G	Listing of the Assembly Language Programs for the Boom Positional Control by Simple Comparison with Slow Speed Limitation	228
APPENDIX H	Listing of the Assembly Language Programs for the 'FMOVE' Routine with Switching Function Evaluation	262
APPENDIX I	Brief Description of the Hydraulic Circuit for the Boom Ripper under Test	275

LIST OF FIGURES

Fig.	Title	Page
1.1	Artistic Illustration of the Mining Operation Near/at the Coal Face.	1
1.2	Diagram Showing the Seven Degrees of Freedom of a Drilling Boom used in Coalmining.	3
1.3	Diagram Showing the Two Degrees of Freedom of a Mining Boom Ripper.	5
2.1	Basic Form of a Valve Controlled Hydraulic Actuator.	15
2.2	Block Diagram of a Basic Hydraulic Servo.	17
3.1	A Hydraulic Actuator with Poppet Valves.	30
3.2	Non-dimensional Ram Displacement - Non-dimensional Time Response for Symmetrical Systems with Different Inertial Masses.	55
3.3	Non-dimensional Ram Displacement - Dimensional Time Response for Symmetrical Systems with Different Inertial Masses.	55
3.4	Non-dimensional Ram Displacement - Non-dimensional Time Response for Asymmetrical Systems with Different Inertial Masses.	56
3.5	Non-dimensional Ram Displacement - Dimensional Time Response for Asymmetrical Systems with Different Inertial Masses.	56
3.6	Non-dimensional Ram Displacement - Non-dimensional Time Response for Symmetrical Systems with Different Damping Factors.	57
3.7	Non-dimensional Ram Displacement - Non-dimensional Time Response for Asymmetrical Systems with Different Damping Factors.	57
3.8	Non-dimensional Ram Displacement - Non-dimensional Time Response for Symmetrical Systems with ON-OFF Valves of Various Time Delay.	58
3.9	Non-dimensional Ram Displacement - Non-dimensional Time Response for Asymmetrical Systems with ON-OFF Valves of Various Time Delay.	58
3.10	Non-dimensional Ram Displacement - Non-dimensional Time Response for Symmetrical Systems with 'Linear' Valves of Different Operating Times.	59
3.11	Non-dimensional Ram Displacement - Non-dimensional Time Response for Asymmetrical Systems with 'Linear' Valves of Different Operating Times.	59

Fig.	Title	Page
3.12	Fluid Flow Rate - Non-dimensional Time Response for an Asymmetrical System with 'Linear' Valves.	60
3.13	Non-dimensional Fluid Flow Rate 1- Non-dimensional Time Response for Asymmetrical Systems with 'Linear' Valves of Different Operating Times.	61
3.14	Non-dimensional Fluid Flow Rate 2 - Non-dimensional Time Response for Asymmetrical Systems with 'Linear' Valves of Different Operating Times.	61
3.15	Non-dimensional Ram Displacement - Non-dimensional Time Response for Symmetrical Systems with Different Externally Applied Forces.	62
3.16	Non-dimensional Ram Displacement - Non-dimensional Time Response for Asymmetrical Systems with Different Externally Applied Forces.	62
4.1	Block Structure of the Timing Test Program.	65
4.2	Normalised Lifting Ram Displacement - Time Response.	75
5.1	The Boom Ripper Assembly under test.	78
5.2	A Hypothetical Profile.	80
5.3	Flow Diagram of the General Structure of the Control Algorithm.	82
5.4	Flow Diagram of the 'Trim Profile' Routine.	85
5.5	A Sample Tunnel Profile.	87
5.6	A Sample Cavity Cutting.	89
5.7	Flow Diagram for the 'Cut Cavity' Routine.	91
5.8	Flow Diagram for the 'Move Boom' Routine.	93
5.9	Flow Diagram for the Data Sorting and Expanding Procedures.	95
5.10	Sample Discrete Profile Data Captured.	98
5.11	Flow Diagram for the 'SMOVE' Routine.	100
6.1	Block Structure of the Proposed Dual Microprocessor Control System for the Mining Boom Positions.	103
6.2	Proposed Dual Microprocessor Control System for Boom Cutter Position.	105
6.3	Flow Diagram for the Actions Taken by the two Microprocessors in Communicating with each other.	106

Fig.	Title	Page
6.4	Schematic Diagram for the Main Microprocessor Board.	108
6.5	Interrupt Instruction Port Supplying the RESTART 7 Instruction.	114
6.6	Schematic Diagram for Expansion RAM and EPROM Board.	117
6.7	Schematic Diagram for the APU Expansion Board.	119
6.8	Schematic Diagram for Interfacing the Main Control System with the Data Capturing one.	121
7.1	Flow Diagram for the 'FMOVE' Routine using Simple Comparison and the 'SMOVE' Routine.	126
7.2	Flow Diagram for the 'FMOVE' Routine using Switching Function and the 'SMOVE' Routine.	128
7.3	Trace of the Boom Cutter Movement in Trimming the Profile.	132
7.4	Trace of the Boom Cutter Movement in Cutting the Cavity.	134
7.5	Boom Overshoot - Initial Slewing Velocity at Switching.	136
7.6	Boom Overshoot - Initial Lifting Velocity at Switching.	136
7.7	Boom Lowering - Time Response (Analogue Trace).	141
7.8	Boom Lowering - Time Response (Digital Trace).	141
7.9	Error - Boom Slewing Velocity.	143
7.10	Error - Boom Lifting Velocity.	143
C.1	Block Structure of the Simulation Program.	166
D.1	Diagrammatic Representation of a Boom-Ripper.	184
D.2	A Boom-Ripper.	186
D.3	Diagram to show the Arrangement to Provide Vertical Movement of the Boom.	186
D.4	The Profile.	187
D.5	Diagrammatic Representation of the Mechanism of the Boom-Ripper.	187
D.6	Cutter Cuts a Hard Rock and Stops.	192
F.1	Flow Diagram for the Algorithm in Evaluating the Chebyshev Series.	218

Fig.	Title	Page
I.1	Schematic Diagram for the Hydraulic Power Supply Circuit.	275
I.2	Arrangement for One of the Hydraulic Actuators in the Boom Ripper under Test.	275
I.3	Cross-section of the Solenoid Operated Pilot Valve used in the System under Test.	277
I.4	Cross-section of the Main Poppet Valve used in the System under Test.	277

LIST OF TABLES

Table	Title	Page
4.1	Examples of the Number Formats	69
4.2	Time Taken to Evaluate the Simplified Switching Function under Various Operating Conditions	73
6.1	Pin Assignment on the Back Plate of the Main Control System	112
6.2	Pin Assignment on the Back Plate for the Interface Card	122
F.1	The First Ten Chebyshev Polynomials $T_n(x)$	217

NOTATION :

The following notations are used throughout the thesis. If they are only used in the local text then the notations will be re-defined.

A_1	Effective area of the ram in chamber 1.
A_2	Effective area of the ram in chamber 2.
a	Area ratio = $\frac{A_2}{A_1}$.
B	Effective external viscous damping of the system.
b_1	Effective width of valve 1.
b_2	Effective width of valve 2.
C_{D_1}	Flow discharge coefficient for valve 1.
C_{D_2}	Flow discharge coefficient for valve 2.
C_L	Internal leakage coefficient across the ram.
F	External applied force to resist positive movement of the ram.
K	Effective stiffness of the system other than the hydraulic stiffness.
L_r	Reference length.
$\mathcal{L}$	Laplace transform operator.
M	Effective inertial mass of the actuator.
P_s	System supply pressure.
P_e	System exhaust pressure.
P_1	Pressure in chamber 1.
P_2	Pressure in chamber 2.

Notation (cont)

P'_1 Pressure 'external' to chamber 1.

$$\begin{cases} = P_s & \text{when } X_{1v} > 0 \\ = P_e & \text{when } X_{1v} < 0 \end{cases}$$

P'_2 Pressure 'external' to chamber 2.

$$\begin{cases} = P_e & \text{when } X_{2v} > 0 \\ = P_s & \text{when } X_{2v} < 0 \end{cases}$$

p_1, p'_1, p_2, p'_2 Non-dimensional pressures for P_1, P'_1, P_2, P'_2

Q_1 Fluid flow rate into chamber 1.

Q_2 Fluid flow rate into chamber 2.

Q_r Reference fluid flow rate.

q_1, q_2 Non-dimensional fluid flow rates Q_1 and Q_2 respectively.

S_1 Hydraulic stiffness of the fluid column in chamber 1.

S_2 Hydraulic stiffness of the fluid column in chamber 2.

T Time.

t Non-dimensional time.

T_r Reference time.

t_d Non-dimensional time delay for On-Off valves.

t_c Non-dimensional time for valve operation (linear type).

V_1 Volume of the fluid between the ram face and the valve in chamber 1.

V_2 Volume of the fluid between the ram face and the valve in chamber 2.

Notation (cont)

V_T	Total volume of fluid in the circuit when the ram is at its reference position.
V_{1r}	Volume of the fluid between the ram face and the valve in chamber 1 when the ram is at its reference position.
V_{2r}	Volume of the fluid between the ram face and the valve in chamber 2 when the ram is at its reference position.
V_{1L}, V_{2L}	Volume of the fluid in line on side 1 and 2 respectively.
V_{1C}, V_{2C}	Clearance volume of the fluid in chamber 1 and 2 respectively.
X_v	Maximum valve opening from the close position.
X_{1v}	Displacement of valve 1 from its close position.
X_{2v}	Displacement of valve 2 from its close position.
x_{1v}, x_{2v}	Non-dimensional quantities for X_{1v} and X_{2v} respectively.
Y	Ram displacement from its reference position.
y	Non-dimensional quantity for Y.
Y_{st}	Piston stroke measured between the two extremes.
$\ddot{y}_o, \dot{y}_o, y_o$	Non-dimensional acceleration, velocity and displacement of the ram at the application of the switching signal to the valves.
β	Bulk modulus of the fluid
ρ	Density of the fluid.
ν	Ratio $= \frac{\beta}{P_s}$
ω_r	Natural hydraulic frequency at the reference position

Notation (cont)

ζ Damping factor $= \frac{BT_r}{2M}$

ξ Constant $= \frac{1}{Q_r} C_{D_1} b_1 X_v \sqrt{\frac{P_s}{\rho}}$

Subscripts

The following subscripts normally have these meanings unless specifically defined.

s	System supply when used with pressure. Spool valve when used with displacement.
r	Reference.
T	Total.
1	Upstream or chamber without piston rod.
2	Downstream or chamber with piston rod.
L	Leakage.
C	Clearance.
e	System exhaust when used with pressure. Equilibrium when used with other quantities.
v	Valve.
st	Stroke.

NOTE ON THE LAYOUT OF THESIS

All figures and tables are included in the text to facilitate reference to them. They are put on the pages facing the text making reference to them whenever possible. Except that the figures referred to in Section 3.8 are grouped together and placed at the end of Chapter 3.

CHAPTER 1

INTRODUCTION

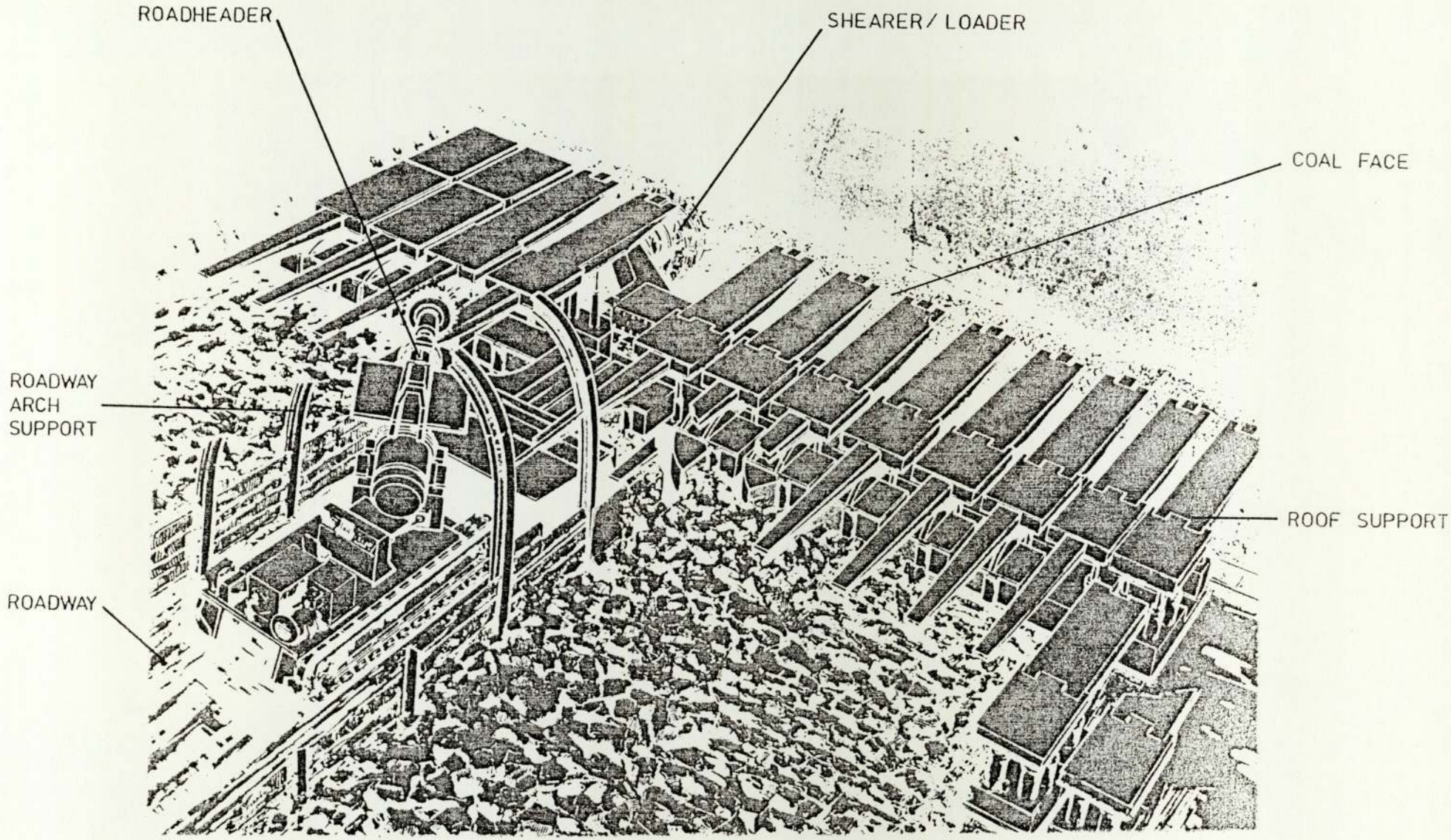


Figure 1.1 Artistic Illustration of the Minining Operations near/at the Coal Face.

1.1. COAL MINING OPERATIONS

Coal mining is one of the traditional industries in the United Kingdom. After centuries of development, it has attained a high level of mechanisation[1]. Nowadays, electro-hydraulic machineries are heavily used in this industry to increase the output from coalmines and the efficiency[2]. Longwall mining is the system used in the UK, some European countries and China; while the room-and-pillar system is generally used in America, Australia and South Africa[3]. Figure 1.1. shows an artistic illustration of the typical coal mining operations in a longwall system near and at the coal face.

Essential operations in longwall mining can be briefly described below. Roadways are built at each end of the coal face, which is normally about 200 metres long, to give access to the coal seam, to facilitate the transportation of coal, machineries and personnel and to provide ventilation. Coal is cut along the seam by electrically powered cutters such as shearer-loaders, and is transported through the roadway. The cavity created by coal extraction along the seam is supported by chocks (hydraulic roof supports) which will be advanced after the coal face has been cut.

The advance in technology has given rise to some degree of automation in the coal industry. The National Coal Board (NCB) has developed a minicomputer system known as Mine Operating System (MINOS) to monitor and control some mining operations[4]. MINOS is used in some mines to automatically

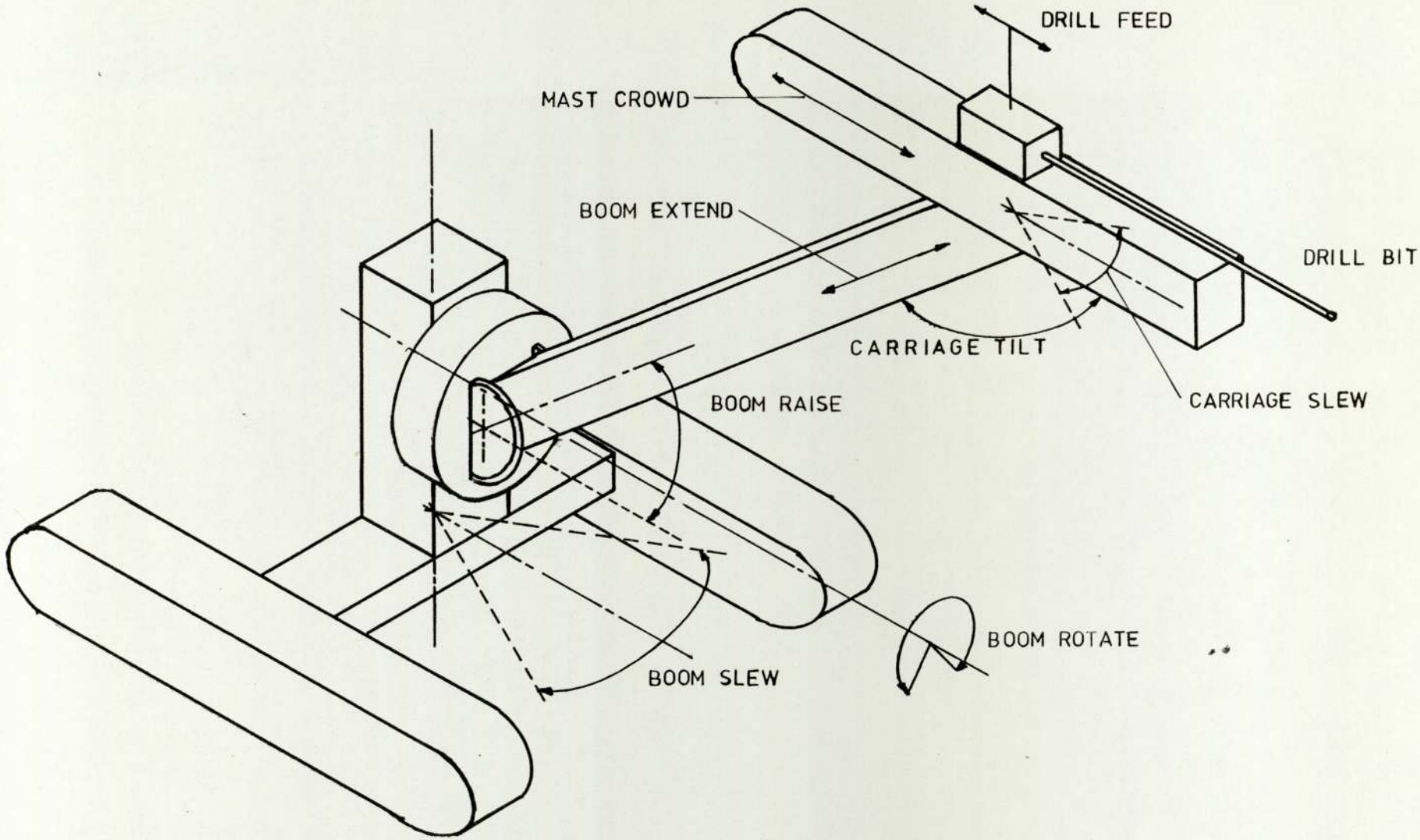


FIG. 1.2 Diagram Showing The Seven Degrees Of Freedom Of A Drilling Boom Used In Coalmining

monitor and control belt conveyors and bunker systems for coal transport, to monitor and record the mine environment for ventilation, to monitor and record the performance of a coal face, to operate fixed plants such as main pumps and fans, and to provide useful management information.

1.2. TUNNELLING

Roadways are made by first driving tunnels then by erecting arches as roadway supports. Setting of arches will be easier if a better profile of the tunnel is obtained. In the UK, tunnels are driven either by shotfiring or by direct cutting of rocks. Mechanisation of this process is highly desirable, therefore ripping machines, which are also known as roadheaders, were introduced to the UK coal industry in 1961. The high speed tunnelling techniques in civil engineering have some influences upon the development and design of ripping machines used in the coal industry[5].

The traditional way to excavate tunnels in UK coalmines is the shotfiring method. In this method, a pattern of holes are drilled for placing explosives and then blasted. Conventionally, this drilling of pattern is a two-man operation: one operator stands at the drilling end and indicates the position of the drilling head to the other, who is operating the control valves for the hydraulic actuators of the drilling boom. Electrohydraulic remote control valves are developed to simplify this operation. However, a drilling machine used in coalmining normally has seven degrees of freedom (Figure 1.2) and is operated

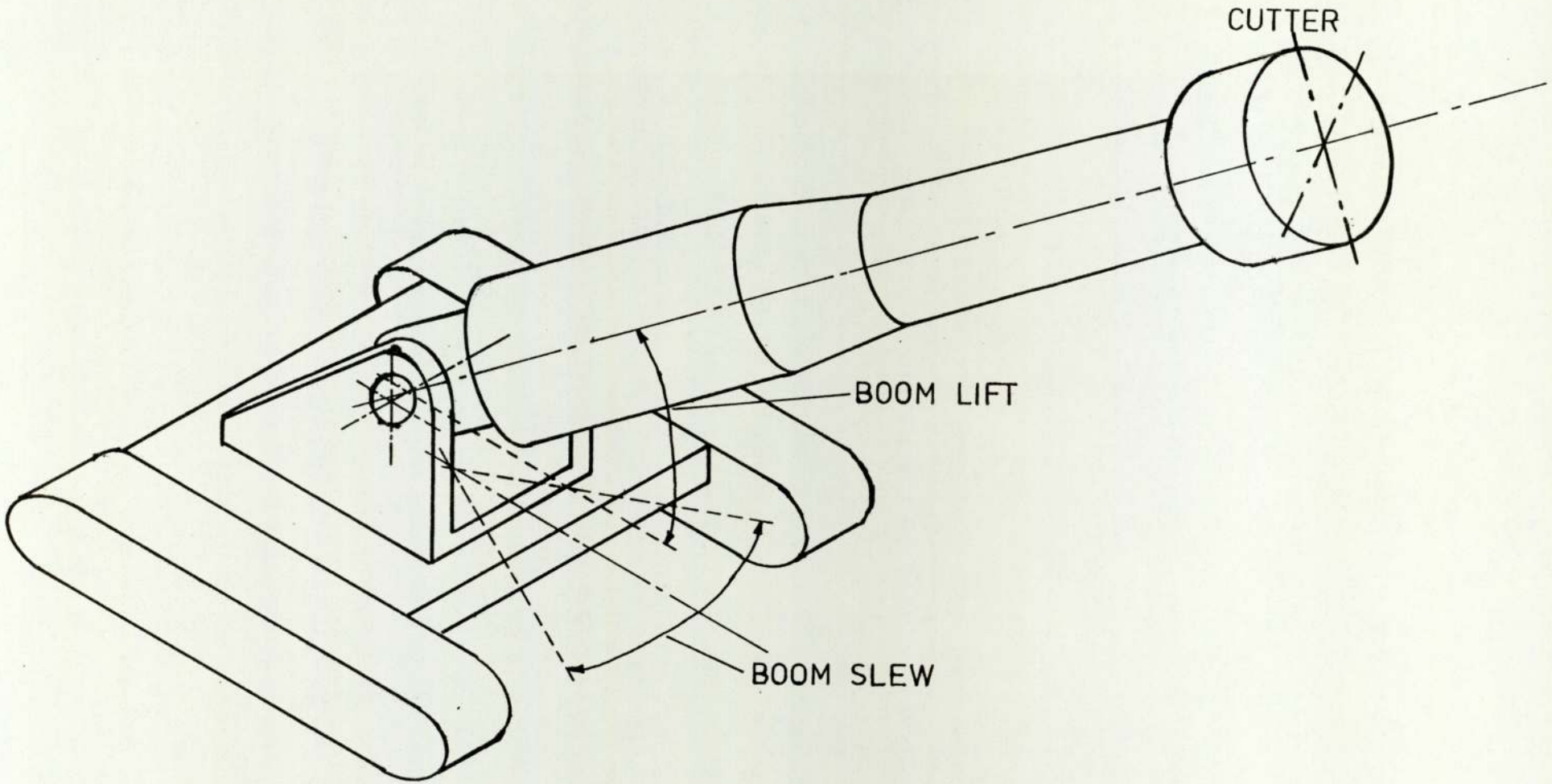


FIG. 1.3 Diagram Showing The Two Degrees Of Freedom Of A Boom Ripper

manually using open-loop control, thus a drilling operation demands high skill and a considerable amount of training for efficient operation. This method is highly labour-intensive and continuous processing is impossible because the machine has to be retracted to permit the blasting of rocks.

Direct cutting of rocks is considered to be an alternative to drilling and blasting. A ripper with a hydraulically powered impact unit mounted at the end of a boom similar to that of a drilling machine has been developed[6]. This impact type ripper can be used to drive the tunnel and trim the profile. This machine provides continuous operation in driving the tunnels, but a large amount of time is required to manipulate the impact breaker boom because of the large number of movements of the hydraulic actuators.

There is another type of roadheader used in UK coalmines, the pick type ripper[7]. This machine has a cutting head comprising of picks mounted and rotated at the end of a boom, which is installed on a machine chasis. The movement of this boom is reduced to have only lifting and slewing, (Figure 1.3). This arrangement greatly reduces the time required to manipulate the boom. Early pick type machines are restricted to operation on rocks of about 70 MN/m^2 compressive stress.

Shield tunnelling used in civil engineering leads to the development of shielded machines. These machines are being tested in some mines, Gascoigne Wood being one of them.

Full face tunnel boring machines are also being developed. However, these machines suffer from many disadvantages including high cost, long installation and dismantling times and inflexibility in coping with variable ground conditions. A heavy-duty boom ripper is being developed so that harder rocks can be cut. This in turn increases the size and weight of the machine. Cutting picks assisted by high pressure water jets are used in heavy-duty rippers to improve their cutting ability[8].

Although the main aim in coal mining is to produce coal, a speedy operation in tunnelling will facilitate transporting materials and machineries, and hence increase the coal production. Therefore, in the UK coal industry, the MRDE (Mining Research and Development Establishment) of the NCB and many mining machinery manufacturers are trying to develop high performance tunnelling machines and improve the performance of existing machines.

The average conditions of strata encountered in UK coal-mines make it more economical to use boom rippers in most areas. While shotfiring is still used in areas with hard rocks, e.g. Six Bells Colliery in South Wales. Shielded machines and full face boring machines may be used in areas where high flexibility is not required. Therefore, the aim of this project is to develop a better positional control system for the existing boom rippers. This will result in deskilling the manipulation of a ripping boom and enable further development in automation of the tunnelling process.

It can be seen in the figures (Figures 1.2. and 1.3) that a boom ripper has fewer degrees of freedom than a drilling boom. This is because a boom ripper is used to cut a predetermined profile of simple shape; while a drilling boom is required to drill a prescribed pattern which demands positioning and orientating the drilling bit mounted on the machine. Although the control system described in this thesis has only two degrees of freedom, it can easily be modified to include more degrees of freedom by considering the machine geometry, and hence the same principle can be applied to the positional control of a drilling boom.

1.3. STIFFNESS OF A MINING BOOM RIPPER

If the stiffness of a mining boom ripper is defined as the ratio of change in force to the change in displacement of the working end in the direction of the applying force, then better position stability of a boom ripper will be achieved if its stiffness is higher. Although this stiffness depends upon both structural and hydraulic stiffnesses of the ripper, the hydraulic one has a greater effect because the boom is supported by hydraulic actuators. Hence a study has been carried out for the hydraulic stiffness of a boom ripper (Appendix D).

It was found that the stiffness of a system will be higher if the supporting hydraulic actuator is connected to the boom closer to the cutter for a given actuator cross-sectional area. However, the stiffness may be kept within a range if the cross-sectional area of the hydraulic actuator is adjusted accordingly.

Unity positional feedback to the hydraulic actuator valves will have a significant improvement on the stiffness for excitation forces of high frequencies, while the stiffness cannot be much increased if the disturbing force is oscillating at a low frequency.

If the maximum force is the force acting tangentially to the boom[7], and components of the resultant force not acting in the direction of the actuator are resisted by another means, then the boom will have a greater stiffness when the cutter position is higher. This is because the component of the resultant force in the direction of the trunk lifting actuator is less when it is more vertical. Similarly, the situation will be the worst when the boom is in a horizontal position cutting an extremely hard rock and fails to break it. If this happens, the machine will 'climb' over the rock and cause a bounce effect. Vertical supports at far ends will help stabilising the machine's sliding movement on the ground.

Although the performance of a boom will be improved by increasing its stiffness, a different structure and arrangement will be required. Since the aim of this project is to improve the performance of existing boom rippers with minimum modification, alternation of boom arrangement to give a better stiffness is not considered. Alternatively, a better positional control of boom ripper is studied.

As reported in this thesis, the valves of the hydraulic actuator are closed simultaneously based on the switching

function. However, it can be shown that a delay in closing the supply valve from the closure of the exhaust valve will improve the effective stiffness of a boom. This is because that cavitation in the supply chamber, caused by rapid closure of the valve, can be avoided, and hence the actuator will have less oscillation before it rests. This is being studied in a parallel project by J H Knight in the Mechanical Engineering Department at the University of Aston.

1.4. CONTROL PRINCIPLES

At present, most of the working time of a boom ripper is incurred in manipulating the boom. Therefore the overall performance of a roadheader will be improved by shortening the time required to position its cutter. A possible way to achieve this is introducing closed-loop control to the system. Since the movement of a boom is caused by the associated hydraulic actuators, closed-loop control will be developed for these actuators.

Hydraulic actuators are basically integrators[9] and the displacement of their rams depend on the fluid flow through the valves, hence one of the possible ways to control these actuators is to control their valve displacements. The conventional system used in this type of control is the hydraulic servomechanism, therefore the basic mathematical model employed in this closed-loop design of the ripping boom is that used for hydraulic servos. Analysis of the model will help developing the

positional control strategy (the switching function) for the mining boom.

Safety in coalmine operations demands the use of fire-resistant fluid in hydraulic machinery. There is a trend to use 5/95 diluted emulsion in UK coalmining machinery because of its fire-resistant property, low cost and non-toxity. The low viscosity of this fluid increases the leakage of sliding type valves, therefore traditional spool valves are not suitable and poppet valves are being developed. The conventional method in proportional control of the spool valves may not be applied because the displacement of a poppet valve is difficult to control using direct proportional control. But the control can be done by using pulse width modulation (PWM) to give an equivalent opening. PWM demands frequent operation of the valves and may cause fatigue problems.

The operation of poppet valves is basically an on-off mode, and in the current project, PWM is not considered but a 'bang-bang' philosophy is employed so that maximum power is used to accelerate and decelerate the hydraulic ram. To achieve a good accuracy, an optimal control strategy is adopted. Switching functions have been developed so that equilibrium can be attained by one switching from acceleration to deceleration, i.e., to close the valves at a point to give a final position within the neighbourhood of the desired position.

1.5. MICROPROCESSOR CONTROL

Optimal control may need a large number of logical decisions, and the conventional way of achieving these operations is the use of expensive computers in which mobility is also restricted. An alternative is to use fluidic elements which are again bulky and expensive. Microprocessors are becoming more commonly used in modern control systems because of their compactness and relatively cheap hardware cost. Development in microelectronics increases the reliability and speed of microprocessor systems and advance in the large scale integration (LSI) technology further reduces the physical size of a microprocessor system.

Microtechnology has a rapid development in the mining industry for the measurement, information and control of mining operations[1]. Therefore it is feasible to develop and use microprocessor control for the current project. The requirements for speed of data handling in capturing and analysing the input and feedback, suggest the use of a multi-microprocessor control system. In this control system, two microprocessors running in parallel are proposed where one is used to capture the input and feedback data so that 'updated' information can be provided for the use of the other microprocessor system, whilst the second is used to analyse the data so that appropriate control signals can be sent to the valves. A special processor, the arithmetic processing unit (APU), is employed to further increase the speed of operation, so

that more sophisticated mathematical analysis can be performed faster.

The microprocessor control system takes care of the appropriate hydraulic actuator movements in positioning the boom cutter, and this will in effect deskill the manipulation of the boom, allowing the operator to concentrate on the quality of the profile. The proposed control strategy can be modified to suit all mining booms including the drilling boom which has seven degrees of freedom. Although the proposed control system is semi-automatic, full automation may be achieved if the necessary sensors have been developed, e.g. a machine position sensor will enable auto-correction of the cutter position.

1.6. ORGANISATION OF THESIS

The current chapter is an introduction to this project and gives a brief description of the coalmining operations incurred in the UK and the importance of positional control in roadheaders. Since the control principles and models employed to develop the boom positional control system are those used for servomechanisms, a brief review of previous researches on this subject is presented in the following chapter. In Chapter 2, a brief review on optimal control is also given, it will be seen that optimal control is normally done by controlling the closure of valves rather than the opening, i.e. control the deceleration rather than the acceleration. Switching functions are derived

for the linear actuators with simple valves in Chapter 3. Since it is desired to employ microprocessor control, arithmetic processing of data is an essential operation in the control system. A decision on using either hardware or software for this process has to be made, a simple comparison between hardware and software arithmetic processing is presented in Chapter 4 to help realising the choice. The main control algorithm and structure of the control programs are given in Chapter 5. The circuits for the microprocessor and associated boards for the main control system are described in Chapter 6. The control algorithm has been tested and the results are presented in Chapter 7. Chapter 8 is the conclusions and recommendations based upon the findings of this project. Details and listings of the programmes may be found in the appendices.

CHAPTER 2

BRIEF REVIEW OF PAST WORK DONE IN APPLYING THE CONTROL THEORIES TO A HYDRAULIC SYSTEM

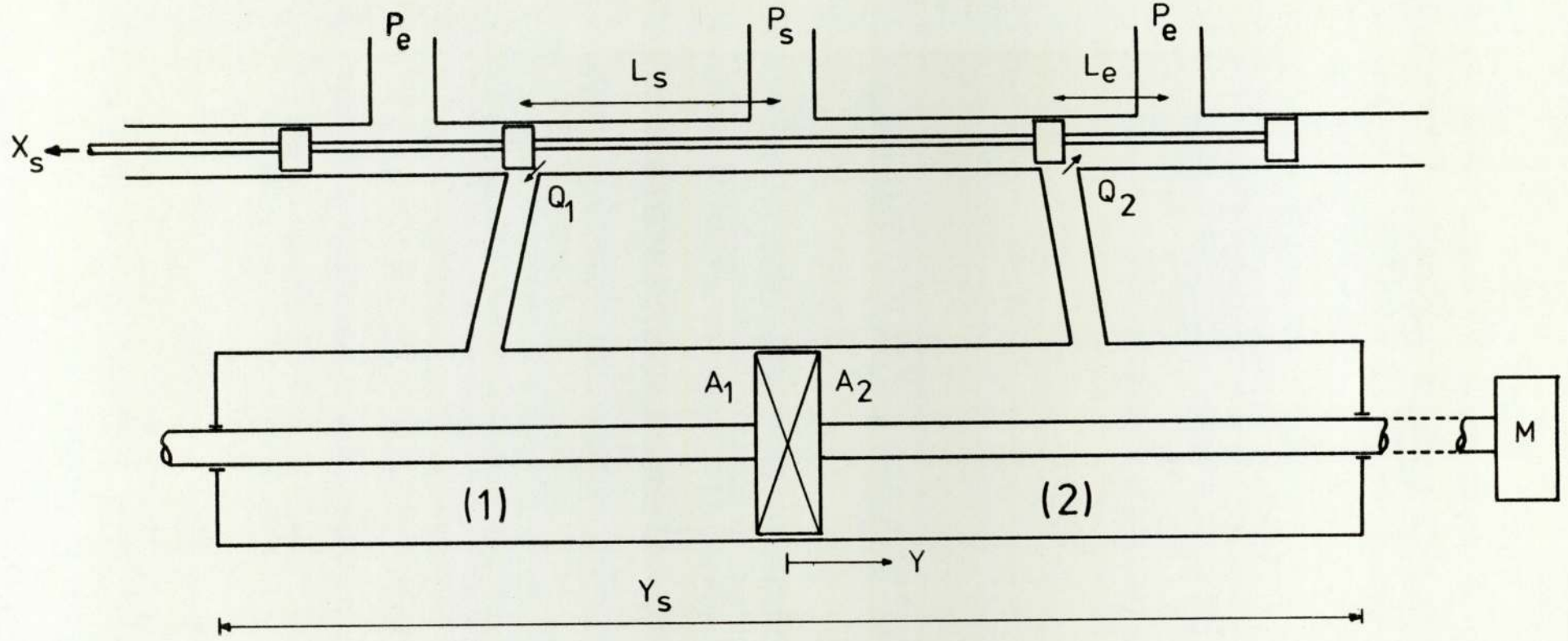


FIG. 2.1 Basic Form Of a Valve Controlled Hydraulic Actuator

2.1. CONVENTIONAL CONTROL

The movement of a hydraulic actuator is commonly controlled by the position of valves and the open-loop system is referred to as a hydraulic relay, but called a servo-mechanism when there is a feedback. Traditionally, the valves used in hydraulic servomechanisms are spool valves which are either mechanically or electrically operated. These valves normally have a movement proportional to the input signal and so control both the magnitude and direction of the fluid flow passing through them. The basic form of a hydraulic actuator with spool valves can be represented by the diagram in Figure 2.1 , and Figure 2.2 shows a block diagram for a hydraulic servo.

A hydraulic relay is an integrating device when used in a position control system because the output displacement of the actuator will be directly proportional to the time integral under ideal conditions[9]. Mathematical modelling of such a hydraulic system will result in a set of nonlinear differential equations which is difficult to analyse, so analysis based on linearisation is commonly used [10,11].

The flow through the valve ports is turbulent and its rate may be assumed to be proportional to the product of the valve opening area and the square root of the pressure drop across the valve as in the case of flow through an orifice[10]. This results in the familiar flow equation (2.1).

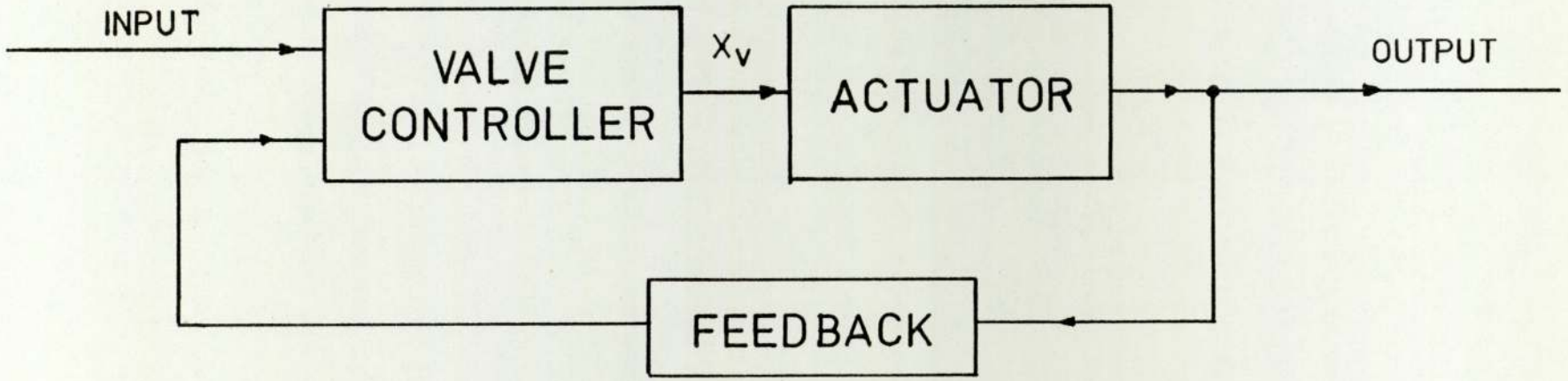


FIG. 2.2 Block Diagram Of A Basic Hydraulic Servo

$$Q = C_d \, a_v \sqrt{\frac{2\Delta p}{\rho}} \qquad (2.1)$$

where Q = fluid flow rate through the valve.

C_d = flow discharge coefficient of the valve.

a_v = valve opening area.

Δp = pressure drop across the valve.

ρ = fluid density.

Analysis of the system is done after combining this flow equation (2.1), the flow equations within the actuator chambers, the equation for the dynamics of the system and the equation for feedback if any.

Since valves are used extensively in hydraulic servo-mechanisms, their performance, behaviour and patterns of stability have been studied in the past. Lee and Blackburn found that there were steady axial and transient flow forces acting on a 4-way spool valve[12]. They also demonstrated that the stability of a piston type control valve would depend upon the damping lengths, L_s and L_e as defined in Figure 2.1. The length L_s is normally made greater than L_e to give a small positive damping on the spool valve to eliminate instability. Taft and Twill had similar findings for a 3-way underlapped spool valve[13]. Harpur used a linearised analysis to show that the oil compressibility could cause instability[11].

Notton and Turnbull showed with the aid of describing functions that a spool valve could have continuous

oscillation if the damping in the spool was not sufficient[14]. They also used a graphical method to predict the frequency and amplitude of this oscillation and suggested that the stability could be improved by complete reaction-force compensated spool valves. Foster and Kulkarni used a linearisation technique and the approximate method of Kryloff and Bogoliuboff to study the spool oscillations and found that the frequency of self-excited oscillation of a spool might be controlled by varying the supply pressure[15]. They also showed that the amplitudes of these oscillations could be controlled by changing the fluid energy losses in the systems and that this type of valve was not suitable for operating as an oscillator because of the oil compressibility and resonant frequency effects at high pressures. Burton, Ukrainetz and Nikiforuk studied an inertially loaded valve with the describing function technique and found that the existence and stability of the limit cycle oscillations could be predicted as a function of the system parameters[16].

Poppet valves attract engineers' attention because of the simplicity in manufacturing, less leakage for fluid with low viscosity (since there is no sliding seal) and less sensitivity to dirts. Poppets are often used as pressure regulators and relief valves in the past. Stone showed that the normal force acting on a poppet face was due to the pressure distribution while the tangential force was viscous in nature. He also reported that the stability could be improved by eliminating the flow forces and that

the downstream configuration had a larger effect on these forces[17]. McCloy and Martin found that the flow through a poppet valve was parallel to the poppet face[18]. Mansfeld investigated electro-hydraulic actuator control using fast switching on-off valves[19]. He found that pulse width modulation (PWM) of these valves could be used to achieve an unsteady fluid flow control, and that PWM might be used for the time-optimal control of electro-hydraulic servomechanisms.

Researches had been carried out for the performance of a hydraulic actuator by numerous people in the past. In the early studies, neglecting the oil compressibility and assuming symmetry were commonly adopted for analysis. Turnbull proposed a graphical method based on the isoclines to find the step response of a symmetrical hydraulic servomechanism[20]. He neglected the oil compressibility and pointed out that the load effect was to increase the response time of the servo. McCloy developed a graphical method to study the step response of a closed-loop system[21]. He also neglected the oil compressibility but considered the effects of frictional forces, inertial and spring loads. Butler provided analytical and graphical solutions for various kinds of inputs: step, ramp and sinusoidal[9]. He neglected the compressibility effects and derived the solutions under certain assumptions for special cases. His assumptions were (i) incompressible fluid and (ii) no leakage and no dynamic friction (viscous damping effects had been neglected). The graphical solutions were presented in the velocity-displacement phase plane.

Cook and Heaps presented a series analysis method to study a closed-loop system with unity feedback[22]. Their analysis was based on the assumption that the oil compressibility effect could be ignored. They also investigated systems with velocity feedforward, velocity and error feedbacks. In this method, they expressed the error term in a series and compared the coefficients of like terms.

Royle derived the response equations of a symmetrical actuator for sinusoidal inputs[23]. He considered the oil compressibility but neglected the leakage across the ram and found an expression for the ram velocity in terms of the load factor and the input frequency. The load factor is a measure of the inertial load on the system. He presented a curve for the relation between the maximum load factor without cavitation and the driving frequency. As the driving frequency was increased to the actuator's natural frequency, the static maximum load factor was $1/\sqrt{2}$. This load factor could be increased by introducing an exhaust valve lap to avoid or delay cavitation.

Lambert and Davies reported that the oil compressibility accounted for damped oscillatory motion[24,25]. The damping ratio is a function of leakage if no other forms of damping are present. However, the most important stabilising effect in most physical systems is the damping effect caused by Coulomb friction. They also derived a method to evaluate the equivalent damping ratio for a sinusoidal oscillation. They derived an open-loop transfer

function assuming that the movement of the ram is restricted to a small value so that the (displacement x area) term could be neglected, the equilibrium position was taken to be the position at mid-stroke. Their analysis was done for a closed-loop system with unity feedback.

Wang showed that an electro-hydraulic servomechanism could be represented by reasonably simple and accurate mathematical models with suitable assumptions[26]. He demonstrated this with an investigation into the response of a symmetrical actuator to step and pulsed inputs. His method can also be used for time-domain design and near time-optimal design of electro-hydraulic servomechanisms.

Martin had investigated a hydraulic servo with unsymmetrical fluid volume conditions using a linearisation (small perturbation) technique. He found that the marginal stability gain was the least for symmetrical oil volumes, therefore asymmetric oil volumes would give a better stability with faster response and enhanced damping[27].

Montgomery and Lichtarowicz presented a theoretical investigation into the effects of asymmetrical valve lap conditions on the open-loop performance of a valve-controlled hydraulic symmetrical actuator[28]. Their studies were carried out in non-dimensional terms and the valves were driven with a sinusoidal excitation. It was reported that valve lap conditions would have a larger effect at low driving frequencies, while these lap effects would decrease when the driving frequency was near and

above the natural frequency of the oil column of the cylinder. Combination of exhaust overlap and supply underlap will give a higher peak pressure and decrease the amplitude of oscillation, and hence reduce the cavitation threshold. Inertial load effect will be significant when the valve-actuator system is used as a vertical oscillator. In these cases, an extra volume term in the compressibility part of the flow equations will help to stabilise the system and stop the ram shifting.

Cavitation reduces the efficiency of a hydraulic servo and may cause undesired damage and instability, therefore work has been done on this subject[9,23]. Cavitation may cause high pressure within the chamber and severe attenuation of the ram displacement. Cavitation may occur when the inlet flow of fluid into a chamber cannot fill up the cavity formed, e.g. there may be cavitation caused by the rapid closure of valves. Wang and Ma had derived the approximate cavitation conditions for a valve controlled symmetrical hydraulic actuator under sinusoidal and discontinuous inputs by neglecting the leakage across the ram and the oil compressibility[29].

McCloy reported that cavitation had larger influence on the pressure dynamic response than on the displacement and ram dynamic responses, that damage caused by cavitation in a cylinder was less than that due to orifice cavitation[30,31]. Therefore cavitation mode operation of hydraulic servos might be permitted. He also showed that simplified models, which did not take cavitation into account, might adequately

describe the dynamic perfomance of a system. This technique eases the analytical studies considerably. He showed that an increase in the natural frequency of an actuator was minimised by cavitation while the linear analysis predicted that the frequency would increase as the piston moved towards the end of the cylinder. He also pointed out that high pressures were usually developed in a system with on-off valves. McCloy and Martin investigated the generation of high pressures in an asymmetrical system due to the rapid valve closure and developed a

damping using the linearised analysis. Their model assumes no damping, thus an energy conserved system has been modelled. In their paper, they also studied the effects of cavitation and derived an expression for the cavitation conditions[32].

2.2. OPTIMAL CONTROL BY CLOSING OF VALVES

Many engineers and researchers had studied optimal control to minimise the response time, overshoot or undershoot within the limits of a system. Oldenburger developed the optimum control transient for the aircraft engine-propellor systems and reported his findings in 1957[33]. He also developed some optimal control switching functions for an electrical control system with linear controlling elements[34].

Bang-bang controller was shown to be the optimal if the control variables were linear[35]. A bang-bang controller is optimal because the control always operates at either

the upper or the lower limit. This bang-bang strategy is also adopted in designing the optimal electro-hydraulic servomechanisms. However, literature on the optimal control of an electro-hydraulic servo is very limited.

Wang used a simplified model to develop an optimal controller for an electro-hydraulic servo[36]. In his system, the valves are opened fully until a predetermined position has been reached, then the valves are closed under control to avoid cavitation. Wang avoided cavitation because he wanted to minimise the damage caused by the actuator cavitation and to use simplified mathematical models.

Davies developed Wang's method further, by not only maintaining the non-cavitating conditions but also introducing a transition stage[37]. Davies first fully opened the valves to give the maximum acceleration, and then allowed the system to enter a transition zone by fully closing the valves until a point when cavitation was about to occur. The valves were again opened followed by controlled valve closure to ensure maximum non-cavitating deceleration.

In Wang and Davies models, the leakage across the ram was assumed to be negligible and the oil compressibility effect has been ignored in the regions where the pressure changes were low.

However, many authors studied the optimal controllers operating under cavitation conditions[38,39,40]. Since

it had been reported that the mechanical damage caused by the actuator cavitation was less severe than that due to the orifice cavitation, the more complexed sub-controllers, which were required to avoid cavitation, might be simplified. Wilson, Nikiforuk and Lepp developed a graphical method to find the transient response of a time-optimised hydraulic servo operating under cavitation conditions[38].

Armstrong and McCloy had developed an optimal controller with an imcompressible, symmetrical and leakage-free model. Then they constructed a sub-optimal controller aiming to give a zero positional error. In designing this controller, they took the oil compressibility into account. They demonstrated that a lightly damped servo would use the sub-optimal controller, while the simple optimal controller was quite adequate for a heavily damped servo. They also suggested that the system performance could be improved by using either a time-optimal controller developed using a more accurate model or a more complex sub-optimal controller[39]. They had developed some optimal on-off controllers and dual-mode controllers. They argued that although the cavitating conditions might be permitted, a better performance would be achieved if there was no cavitation[40]. They suggested the use of a correction function evaluated from the steady-state errors to improve the performance of the hydraulic servo.

2.3. APPLICATION TO THE CURRENT PROJECT

In order to study the system behaviour of the hydraulic actuator, Conway's mathematical model[10] of the flow through a valve is adopted, and all the equations are linearised by series expansion. The control valves used will be poppet valves, which can be treated as a single spool valve, if the exhaust and supply valves are operating simultaneously. Therefore, McCloy's finding for the flow pattern through a poppet valve[18] is applied to evaluate the equivalent flow area used in the equation (2.1).

The performance of the valves, which are being studied in a parallel project to this one by J H Knight, is not investigated in this thesis. However, the valves are assumed to be simple valves, and the switching functions have been derived for pure on-off valves, on-off valves with time delay and 'linearly' operated valves.

The criterion for the asymmetric actuator cavitation due to the rapid closure of valves[32] is used to check the possibility of cavitation in this control system. A simplified system model is adopted in which only inertial, viscous damping and external loads are assumed. The hydraulic actuators used have no experimentally measureable internal leakage, therefore the leakage across the ram has been neglected in deriving the switching functions. The optimal controllers, mentioned in the above literatures, control the valve openings during decelerations[36,37,38,39,40]. However, in this project, the proposed control is done by

controlling the moment to initiate the valve switching instead of the way it switches. This is because of the difficulty in controlling the flow area of poppet valves and the requirement for less complex controllers. Since actuator cavitation may be allowed[31,38], this control strategy will ease the analysis and design of the controller.

The use of a correction function to improve the accuracy of the actuator positioning has been considered. However, the steady state errors, predicted by the computer simulations, are so small that the complexity in using the correction function is not justified.

CHAPTER 3

DERIVATION OF THE SWITCHING FUNCTIONS

3.1. Introduction

Hydraulic mechanisms are commonly used in positional control with closed-loop arrangements. However, in some applications, conventional feedback techniques may not be effective then alternative methods should be used.

An improved form of control for a ripping boom can be obtained by using a microprocessor based control system. The microprocessor, being a digital device, requires all analogue feedback signals to be converted into digital forms. Output signals from the microprocessor control system will also be in 'ON-OFF' form and hydraulic or mechanical hardware capable of operating directly from these signals would have advantages.

The change of the boom position can be achieved in a minimum time if the maximum available power is used. An 'ON-OFF' mechanism can serve this purpose by driving the ram at the maximum speed, then braking it with the maximum retardation. In fact, the optimal control theory shows that the time used will be a minimum if the valves of a hydraulic actuator are closed fully when the ram has reached its desired position, provided that the fluid is incompressible[40].

Because of the fluid compressibility, the valves cannot be switched off when the ram is at the desired position, they should be closed at some other position so that the ram rests at the required position. This requires the setting up of a switching function to account for the fluid

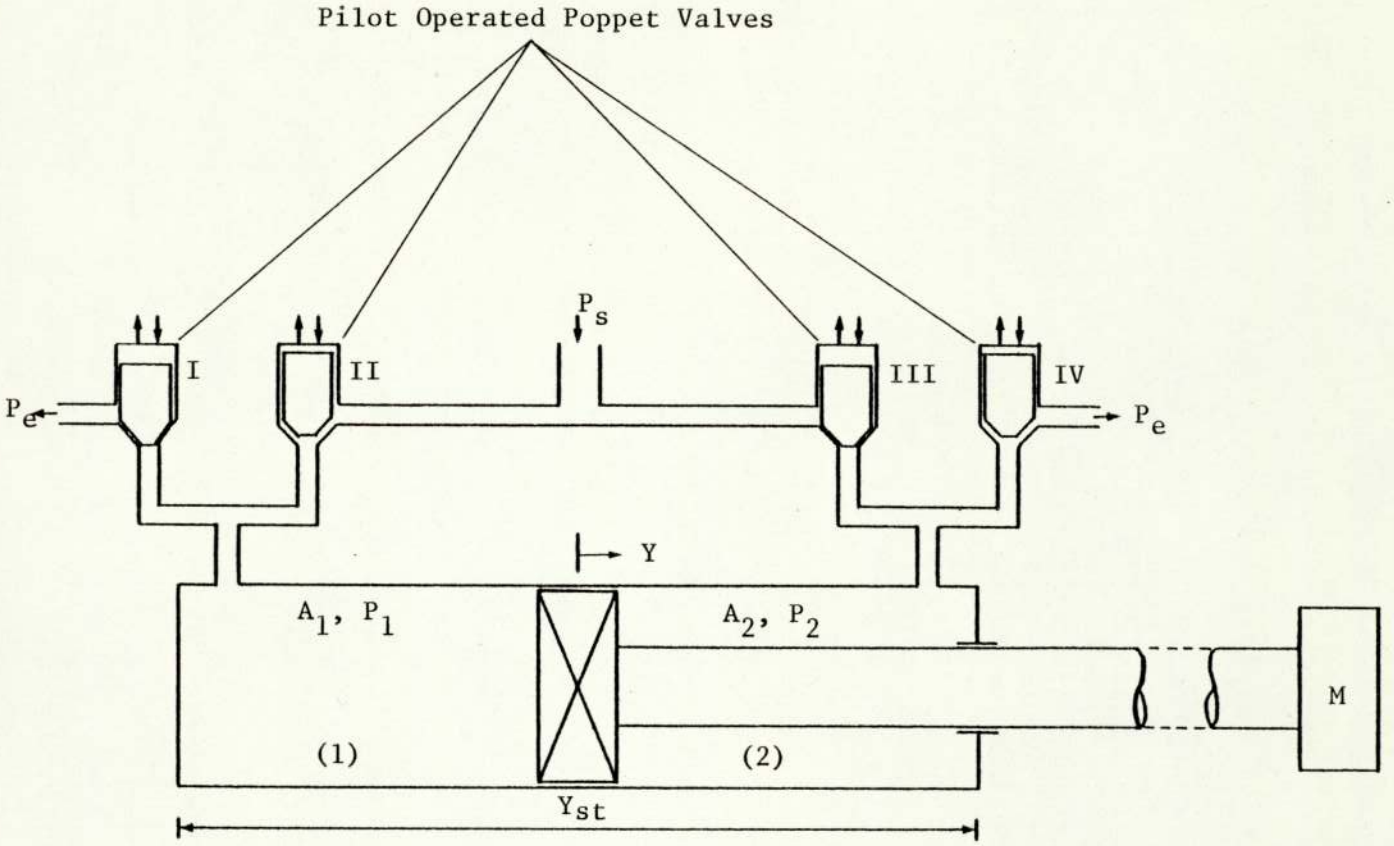


Fig 3.1 A Hydraulic Actuator with Poppet Valves

compressibility. This switching function will depend not only on the desired position but also upon the velocity and acceleration of the ram at switching as well as some other physical properties of the system. However, the coefficients associated with the switching function will remain the same for a particular system, therefore they have to be evaluated only once.

3.2 Mathematical Model

The basic form of a hydraulic actuator used in the current project is shown in Figure 3.1. The poppet valves (I) and (IV) are connected to the system exhaust, while the poppets (II) and (III) are connected to the system supply. In this configuration, the poppets are operated in pairs: (I) and (III) or (II) and (IV). A brief description of the hydraulic system for the boom ripper under test is given in Appendix I. In this study, the poppet valve pairs will be operated simultaneously, they can be treated as a spool valve without any lapping as shown in Figure 2.1. The behaviour of this hydraulic actuator can then be described by the set of differential equations (3.1) to (3.5).

Flow

Valves: $$Q_1 = C_{D_1} b_1 |X_{1v}| \sqrt{\frac{2}{\rho} |P_1' - P_1|} \text{ Sign } (P_1' - P_1) \qquad (3.1)$$

$$Q_2 = C_{D_2} b_2 |X_{2v}| \sqrt{\frac{2}{\rho} |P_2' - P_2|} \text{ Sign } (P_2' - P_2) \qquad (3.2)$$

where Q_1 = Fluid flow rate through the valve at chamber 1.

Q_2 = Fluid flow rate through the valve at chamber 2.

C_{D_1} = Flow discharge coefficient for the valve at chamber 1.

C_{D_2} = Flow discharge coefficient for the valve at chamber 2.

b_1 = Width of the valve at chamber 1.

b_2 = Width of the valve at chamber 2.

X_{1v} = Opening of the valve at chamber 1.

X_{2v} = Opening of the valve at chamber 2.

P_1' = Pressure 'external' to chamber 1. $\begin{cases} = P_s & \text{when } X_{1v} \geq 0 \\ = P_e & \text{when } X_{1v} < 0 \end{cases}$

P_2' = Pressure 'external' to chamber 2. $\begin{cases} = P_e & \text{when } X_{2v} \geq 0 \\ = P_s & \text{when } X_{2v} < 0 \end{cases}$

P_1 = Pressure in chamber 1.

P_2 = Pressure in chamber 2.

ρ = Fluid density.

The signum functions are present to account for any reverse flow. The flow is considered to be positive when it flows into the chamber and the displacement is positive when the volume in chamber 2 is being decreased.

Chambers:

$$Q_1 = A_1 \frac{dY}{dT} + C_L(P_1 - P_2) + \frac{V_1}{\beta} \frac{dP_1}{dT} \qquad (3.3)$$

$$Q_2 = A_2 \frac{dY}{dT} + C_L(P_2 - P_1) + \frac{V_2}{\beta} \frac{dP_2}{dT} \qquad (3.4)$$

where

A_1 = Sectional area of the piston face in chamber 1.

A_2 = Sectional area of the piston face in chamber 2.

C_L = Internal leakage coefficient across the ram.

V_1 = Volume of the fluid between the ram face and the valve in chamber 1.

V_2 = Volume of the fluid between the ram face and the valve in chamber 2.

Y = Ram displacement.

T = Time.

β = Bulk modulus of the fluid.

Dynamics:

$$P_1A_1 - P_2A_2 = M \frac{d^2Y}{dt^2} + B \frac{dY}{dT} + KY + F \qquad (3.5)$$

where

M = Inertial mass of the moving parts.

B = Effective viscous damping on the system.

K = Effective external spring stiffness of the system.

F = Effective external force applied to the system.

3.3. Reference Quantities for Non-dimensionalisation

3.3.1. *Reference Position*

When the piston is at this position, the total volume of the fluid in the circuit is V_T with V_{1r} and V_{2r} as the volumes of the fluid between the valve and the piston face in chambers 1 and 2 respectively.

$$V_T = V_{1r} + V_{2r} \tag{3.6}$$

The reference position is chosen when:

$$\frac{V_{2r}}{V_{1r}} = \frac{A_2}{A_1} = a \tag{3.7}$$

$$\therefore V_T = V_{1r}(1 + a) \tag{3.8}$$

If this is satisfied when the ram face in chamber 1 is Y_r from its mid-stroke position then

$$\begin{aligned} V_{1r} &= A_1(Y_r + \tfrac{1}{2}Y_{st} + Y_{10}) \\ &= \tfrac{1}{2} A_1 (Y_{st} + 2Y_r + 2Y_{10}) \end{aligned} \tag{3.9}$$

$$\begin{aligned} V_{2r} &= A_2 (\tfrac{1}{2}Y_{st} + Y_{20} - Y_r) \\ &= \tfrac{1}{2} aA_1 (Y_{st} + 2Y_{20} - 2Y_r) \end{aligned} \tag{3.10}$$

$$\therefore Y_{st} + 2Y_{20} - 2Y_r = Y_{st} + 2Y_{10} + 2Y_r$$

$$4Y_r = 2(Y_{20} - Y_{10})$$

$$Y_r = \tfrac{1}{2}(Y_{20} - Y_{10}) \tag{3.11}$$

where $$Y_{iO} = \frac{V_{iL} + V_{iC}}{A_i} \qquad i=1,2$$

V_{iL} = Volume of the fluid in the hose on side i $\quad i=1,2$

V_{iC} = Clearance volume of the fluid in chamber i $\quad i=1,2$

Y_{st} = Piston stroke measured between the two extremes

3.3.2. *Reference Length*

The reference length is chosen such that

$$L_r = \frac{V_{2r}}{A_2} = \frac{V_{1r}}{A_1}$$

$$= \tfrac{1}{2}(Y_{st} + 2Y_{20} - 2Y_r)$$

$$= \tfrac{1}{2}(Y_{st} + 2Y_{10} + 2Y_r) \qquad (3.12)$$

3.3.3. *Reference Area*

The reference area is chosen to be the area of the piston face in the chamber 1.

i.e. $\text{Area}_{\text{reference}} = A_1$

3.3.4. *Reference Time*

When the piston is at its reference position, the stiffness of the fluid column in chamber 1 will be given by:

$$S_{1r} = \frac{\beta A_1^2}{V_{1r}} = \frac{\beta A_1^2}{A_1 L_r} = \frac{\beta A_1}{L_r}$$

The stiffness of the fluid column in chamber 2 is:

$$S_{2r} = \frac{\beta A_2^2}{V_{2r}} = \frac{\beta A_2^2}{A_2 L_r} = \frac{\beta A_2}{L_r} = a \frac{\beta A_1}{L_r} = aS_{1r}$$

$\therefore$ The total stiffness of the fluid columns at this position

$$S_r = S_{1r} + S_{2r} = (1 + a) \frac{\beta A_1}{L_r} \qquad (3.13)$$

The natural hydraulic frequency at this position is:

$$\omega_r = \sqrt{\frac{S_r}{M}} = \sqrt{\frac{\beta A_1}{ML_r} (1 + a)} \qquad (3.14)$$

Therefore the natural hydraulic frequency will vary with the physical configuration of the ram and cylinder (including the inertial mass, stroke, piston areas, clearance, etc.) as well as the compressibility of the fluid, and this frequency is a characteristic of a particular system. On non-dimensionalising the quantities, the reference time is chosen to be the inverse of this natural hydraulic frequency:

$$T_r = \frac{1}{\omega_r} = \sqrt{\frac{ML_r}{\beta A_1 (1 + a)}} \qquad (3.15)$$

3.3.5. *Reference Pressure*

The reference pressure is chosen to be the system supply pressure, P_s.

3.3.6. *Reference Flow Rate*

The reference flow rate is chosen to be the reference volume divided by the reference time:

$$Q_r = \frac{V_{1r}}{T_r} = \frac{A_1 L_r}{T_r}$$

$$= A_1 L_r \sqrt{\frac{\beta A_1}{ML_r}(1 + a)} \qquad (3.16)$$

where V_{1r} = Reference volume = $A_1 L_r$

3.4. Non-dimensionalised Equations

After the equations have been non-dimensionalised with the reference quantities, the set of differential equations will become:

$$q_1 = \frac{Q_1}{Q_r} = \frac{A_1 T_r L_r}{A_1 L_r T_r}\frac{dy}{dt} + \frac{P_s}{Q_r}(p_1 - p_2)C_L + \frac{V_{1r} + A_1 L_r y}{\beta Q_r}\frac{P_s}{T_r}\frac{dp_1}{dt}$$

$$= \frac{dy}{dt} + \frac{P_s C_L}{Q_r}(p_1 - p_2) + \frac{P_s}{\beta}(1 + y)\frac{dp_1}{dt} \qquad (3.17)$$

$$q_2 = \frac{Q_2}{Q_r} = -a\frac{dy}{dt} + \frac{P_s C_L}{Q_r}(p_2 - p_1) + \frac{P_s}{\beta}(1 - y)a\frac{dp_2}{dt} \qquad (3.18)$$

$$p_1 - ap_2 = \frac{ML_r}{P_s A_1 T_r^2}\frac{d^2y}{dt^2} + \frac{BL_r}{P_s A_1 T_r}\frac{dy}{dt} + \frac{KL_r}{P_s A_1}y + \frac{F}{P_s A_1} \qquad (3.19)$$

Also

$$q_1 = \frac{1}{Q_r} C_{D_1} b_1 X_v |x_1| \sqrt{\frac{2P_s}{\rho}|p_1' - p_1|}\ \mathrm{Sign}(p_1' - p_1) \qquad (3.20)$$

$$q_2 = \frac{1}{Q_r} C_{D_2} b_2 X_v |x_2| \sqrt{\frac{2P_s}{\rho}|p_2' - p_2|}\ \mathrm{Sign}(p_2' - p_2) \qquad (3.21)$$

3.5. Possibility for Cavitation to occur

Fluid cannot sustain tension, therefore when the pressure falls below the vapour pressure, cavities form behind the moving ram. A necessary condition for cavitation to occur is that the pressure in either chamber takes on the bound, vapour pressure, for a finite time of duration. Both Wang and McCloy pointed out that rapid braking of hydraulic actuators may induce undesirable cavitation in the working fluid of the actuator [32,36]. Wang also stated that non-cavitation condition is always satisfied during acceleration[36].

McCloy and Martin [32] found that the conditions for cavitation to accur during rapid braking of a hydraulic actuator might be:

$$\frac{dy}{dt}_{max} \geq \sqrt{1 - y_d} \quad \frac{P_s}{\beta} p_{1e} \qquad (3.22)$$

where y_d is the ram displacement from its reference position divided by the stroke;

p_{1e} is the non-dimensional equilibrium pressure in chamber 1.

If there is no load other than the inertial load in the system, then the equilibrium pressure will be equal to half the supply pressure.

In many applications, the ram has achieved its terminal velocity before the rapid braking of the hydraulic actuator occurs, then this velocity will become the maximum.

For a symmetrical system with the following parameters:

Inertial mass = 2000 kg

System supply pressure = 137 bar gauge.
(2000 psig approx.)

System exhaust pressure = 0 bar gauge.

Fluid density = 1000 Kg/m^3 (5/95 diluted emulsion may be treated as water).

Compressibility = 2.05×10^9 N/m^2.

Dimensions of the rectangular valve = 6 mm x 6 mm.

Working stroke = 1 m.

Effective piston diameter = 15 cm.

Ratio of total volume in circuit at the reference position to the swept volume = 1.1.

Flow discharge coefficient for the valve ports = 0.65.

Maximum permissible velocity can be obtained from (3.22):

$$\left(\frac{dy}{dt}\right)_{max} = 1.30 \times 10^{-3}$$

Terminal velocity can be obtained from (3.17):

$$\left(\frac{dy}{dt}\right)_{terminal} = 1.098 \times 10^{-3}$$

Therefore with these physical parameters, cavitation is unlikely to occur.

3.6. Derivation of the Switching Function for a Symmetric System

3.6.1. *Analysis*

Subtracting (3.18) from (3.17) gives,

$$q_1 - q_2 = 2\frac{dy}{dt} + \frac{2P_s C_L}{Q_r}(p_1 - p_2) + \frac{P_s}{\beta}\frac{d(p_1 - p_2)}{dt} + \frac{P_s}{\beta}\frac{d(p_1 + p_2)}{dt}\, y$$

since a = 1 for a symmetric system.

Furthermore, for a symmetric system,

$$\frac{d(p_1 + p_2)}{dt} = 0$$

$$\therefore q_1 - q_2 = 2\frac{dy}{dt} + \frac{2P_s C_L}{Q_r}(p_1 - p_2) + \frac{P_s}{\beta}\frac{d(p_1 - p_2)}{dt} \qquad (3.23)$$

If the internal leakage is very small so that it can be neglected, equation (3.23) can further be reduced to

$$q_1 - q_2 = 2\frac{dy}{dt} + \frac{P_s}{\beta}\frac{d(p_1 - p_2)}{dt} \qquad (3.24)$$

If there is no external force and no external spring, then equation (3.19) gives

$$p_1 - p_2 = \frac{ML_r}{P_s A_1 T_r^2}\frac{d^2y}{dt^2} + \frac{BL_r}{P_s A_1 T_r}\frac{dy}{dt} \qquad (3.25)$$

It can easily be shown that if there is no cavitation then, by neglecting the compressibility effect, an approximation can be obtained [36]:

$$q_1 = -q_2 = K_1 |x| \sqrt{|p_1' - p_1|}\ \text{Sign}(p_1' - p_1)$$

where $K_1 = \frac{1}{Q_r} C_{D_1} b_1 X_v \sqrt{\frac{2P_s}{\rho}}$

and the valves do not have any lapping.

With the system exhaust pressure approximately equals to zero gauge, the following relationship can be derived:

$$p_1' \begin{cases} = 1 \text{ for } x \geq 0 \\ = 0 \text{ for } x < 0 \end{cases};$$

$$p_2' \begin{cases} = 0 \text{ for } x \geq 0 \\ = 1 \text{ for } x < 0 \end{cases}$$

Therefore,

$$p_1 = \tfrac{1}{2}(1 + p_1 - p_2)$$

$$p_2 = \tfrac{1}{2}(1 - p_1 + p_2)$$

With these substitutions, equation (3.20) is expanded using Taylor's series for $x > 0$, the expression becomes,

$$q_1 = K_1 x \sqrt{\tfrac{1}{2}\left[1 - \tfrac{1}{2}(p_1 - p_2)\right]} \qquad (3.26)$$

when the terms of order higher than one are being neglected.

Combining (3.24), (3.25) and (3.26) gives,

$$2K_1 x \sqrt{\tfrac{1}{2}}\left[1 - \tfrac{1}{2}\left(\frac{ML_r}{P_s A_1 T_r^2}\frac{d^2y}{dt^2} + \frac{BL_r}{P_s A_1 T_r}\frac{dy}{dt}\right)\right]$$

$$= 2\frac{dy}{dt} + \frac{P_s}{\beta}\left[\frac{ML_r}{P_s A_1 T_r^2}\frac{d^3y}{dt^3} + \frac{BL_r}{P_s A_1 T_r}\frac{d^2y}{dt^2}\right]$$

Rearranging becomes,

$$\frac{ML_r}{\beta A_1 T_r^2}\frac{d^3y}{dt^3} + \left[\frac{BL_r}{\beta A_1 T_r} + \frac{ML_r}{P_s A_1 T_r^2} K_1 x \sqrt{\tfrac{1}{2}}\right]\frac{d^2y}{dt^2} + \left[2 + \frac{BL_r}{P_s A_1 T_r} K_1 x \sqrt{\tfrac{1}{2}}\right]\frac{dy}{dt}$$

$$= 2K_1 x \sqrt{\tfrac{1}{2}} \qquad (3.27)$$

Let $\zeta = \frac{BT_r}{2M}$, $\nu = \frac{\beta}{p_s}$, and $\xi = K_1 \sqrt{\frac{1}{2}}$, then the above equation can be re-written as:

$$\frac{d^3y}{dt^3} + (2\zeta + \nu\xi x) \frac{d^2y}{dt^2} + (1 + 2\nu\zeta\xi x) \frac{dy}{dt} = \xi x \qquad (3.28)$$

3.6.2. *Ideal On-Off Valves*

If the valves used can be approximated to ideal ON-OFF valves, then the non-dimensional valve displacement can be one of the following: -1, 0 and 1. The following discussion will be on the cases when the ram is required to move in the positive direction, similar analysis can be used for movement in the other direction.

When the valves are opened, (3.28) can be written as:

$$\frac{d^3y}{dt^3} + (2\zeta + \nu\xi) \frac{d^2y}{dt^2} + (1 + 2\nu\zeta\xi) \frac{dy}{dt} = \xi$$

When the valves are closed, this becomes,

$$\frac{d^3y}{dt^3} + 2\zeta \frac{d^2y}{dt^2} + \frac{dy}{dt} = 0 \qquad (3.29)$$

After the valves have been closed, the ram displacement can be obtained by solving the differential equation (3.29). Integrating this gives,

$$\frac{d^2y}{dt^2} + 2\zeta \frac{dy}{dt} + y = R_o \qquad (3.30)$$

where $R_o = \ddot{y}_o + 2\zeta \dot{y}_o + y_o$

y_o = Initial displacement of the ram on closing the valves.

$\dot{y}_o$ = Initial ram velocity on closing the valves.

$\ddot{y}_o$ = Initial ram acceleration on closing the valves.

Solution to (3.30) is found to be:

$$y(t) = \frac{1}{s_1 - s_2}\left(\dot{y}_o + 2\zeta y_o - s_1 y_o - \frac{R_o}{s_1}\right) e^{-s_1 t} - \left(\dot{y}_o + 2\zeta y_o - s_2 y_o - \frac{R_o}{s_2}\right) e^{-s_2 t}$$

$$+ \frac{R_o}{s_1 s_2}$$

where $-s_1$ and $-s_2$ are the roots of the characteristic equation:

$$s^2 + 2\zeta s + 1 = 0$$

As $t \to \infty$, $y \to \frac{R_o}{s_1 s_2} = R_o = \ddot{y}_o + 2\zeta \dot{y}_o + y_o$

$\therefore$ The ram will rest at the desired position, y_d, if the valves are switched off when R_o is equal to y_d. Hence,

$$\ddot{y}_o + 2\zeta \dot{y}_o + y_o = y_d$$

is the switching function for this type of valve.

3.6.3. *On-Off Valves with Time Delay*

The actual operation of electromagnetic valves can be very fast, but there may be a time delay due to the residual electromegnetism. This type of valve can then be approximated to on-off valves with time delay, t_d. For this type of valve, over the period between the application of 'close' signal and the actual closure of the valves, equation (3.28) can be written as:

$$\frac{d^3y}{dt^3} + (2\zeta + \nu\xi)\frac{d^2y}{dt^2} + (1 + 2\nu\zeta\xi)\frac{dy}{dt} = \xi \qquad (3.31)$$

Integrating this with respect to t gives,

$$\frac{d^2y}{dt^2} + (2\zeta + \nu\xi)\frac{dy}{dt} + (1 + 2\nu\zeta\xi)y = \xi t + R_1$$

where $R_1 = \ddot{y}_o + (2\zeta + \nu\xi)\dot{y}_o + (1 + 2\nu\zeta\xi)y_o$

Solution to this equation is:

$$y(t) = \frac{\xi t}{\phi_1\phi_2} + \frac{\xi}{\phi_1 - \phi_2}\left(\frac{e^{\phi_1 t}}{\phi_1^2} - \frac{e^{\phi_2 t}}{\phi_2^2}\right) + \frac{\phi_1 + \phi_2}{\phi_1^2\phi_2^2}\xi + y_o$$

$$+ \frac{\dot{y}_o}{\phi_1 - \phi_2}\left[\frac{\phi_1}{\phi_2}e^{\phi_2 t} - \frac{\phi_2}{\phi_1}e^{\phi_1 t} - \frac{(\phi_1 + \phi_2)(\phi_1 - \phi_2)}{\phi_1\phi_2}\right]$$

$$+ \frac{\ddot{y}_o}{\phi_1 - \phi_2}\left[\frac{e^{\phi_1 t}}{\phi_1} - \frac{e^{\phi_2 t}}{\phi_2} + \frac{\phi_1 - \phi_2}{\phi_1\phi_2}\right]$$

where $(s - \phi_1)(s - \phi_2) = s^2 + (2\zeta + \nu\xi)s + (1 + 2\nu\zeta\xi)$

At t_d, the valves are completely closed and the following is required at this instant (Section 3.6.2):

$$y_d = \ddot{y}_{t_d} + 2\zeta\,\dot{y}_{t_d} + y_{t_d}$$

Therefore, the switching function is:

$$y_d = \frac{\ddot{y}_o}{\phi_1 - \phi_2}\left[(\phi_1 + 2\zeta + \frac{1}{\phi_1})e^{\phi_1 t_d} - (\phi_2 + 2\zeta + \frac{1}{\phi_2})e^{\phi_2 t_d} + \frac{\phi_1 - \phi_2}{\phi_1\phi_2}\right]$$

$$+ \frac{\dot{y}_o}{\phi_1 - \phi_2}\left[\phi_1(\phi_2 + 2\zeta + \frac{1}{\phi_2})e^{\phi_2 t_d} - \phi_2(\phi_1 + 2\zeta + \frac{1}{\phi_1})e^{\phi_1 t_d} - \frac{\phi_1^2 - \phi_2^2}{\phi_1\phi_2}\right]$$

$$+ \frac{\xi}{\phi_1\phi_2}\left[(2\zeta + t_d + \frac{\phi_1 - \phi_2}{\phi_1\phi_2})\right] + y_o$$

$$+ \frac{\xi}{\phi_1 - \phi_2}\left[\frac{1}{\phi_1}(\phi_1 + 2\zeta + \frac{1}{\phi_1})e^{\phi_1 t_d} - \frac{1}{\phi_2}(\phi_2 + 2\zeta + \frac{1}{\phi_2})e^{\phi_2 t_d}\right]$$

3.6.4. *Linearly Operated Valves*

In practice, most of the spool valves dynamics can often be approximated to linear operations [29]. If the time taken to operate the valve is t_c, then on closing the valve, displacement can be described as: valve displacement (non-dimensional) = $x = 1 - \frac{t}{t_c}$ where t is the time after the application of the closing signal but before the complete closure of the valve. Over the period between the application of the closing signal and the complete closure of valves, equation (3.28) can be written as:

$$\frac{d^3y}{dt^3} + \left[2\zeta + (1 - \frac{t}{t_c})\nu\xi\right]\frac{d^2y}{dt^2} + \left[1 + 2\nu\zeta\xi\ (1 - \frac{t}{t_c})\right]\frac{dy}{dt} = \xi(1 - \frac{t}{t_c}) \qquad (3.32)$$

The solution to this equation can be found by applying Laplace transform and then inversing the Laplace transform.

With the following substitution:

$$(s - \theta_1)(s - \theta_2)(s - \theta_3) = s^3 + (2\zeta + \nu\xi)s^2 + (1 + 2\nu\zeta\xi + \frac{2\nu\xi}{t_c})s + \frac{2\nu\zeta\xi}{t_c}$$

the expression for the non-dimensional ram displacement can be simplified.

The solution is:

$$Y(t) = \frac{\ddot{y}_o}{(\theta_1-\theta_2)(\theta_2-\theta_3)(\theta_3-\theta_1)}\left[(\theta_3-\theta_2)e^{\theta_1 t} + (\theta_1-\theta_3)e^{\theta_2 t} + (\theta_2-\theta_1)e^{\theta_3 t}\right]$$

$$+ \frac{\dot{y}_o}{(\theta_1-\theta_2)(\theta_2-\theta_3)(\theta_3-\theta_1)}\begin{bmatrix}(2\zeta+\nu\xi+\theta_1)(\theta_3-\theta_2)e^{\theta_1 t}\\ +(\theta_2+\nu\xi+2\zeta)(\theta_1-\theta_3)e^{\theta_2 t}\\ +(2\zeta+\nu\xi+\theta_3)(\theta_2-\theta_1)e^{\theta_3 t}\end{bmatrix}$$

$$+ \frac{y_o}{(\theta_1-\theta_2)(\theta_2-\theta_3)(\theta_3-\theta_1)}\begin{bmatrix}\left\{1+2\nu\zeta\xi+\frac{\nu\xi}{t_c}+(2\zeta+\nu\xi)\theta_1+\theta_1^2\right\}e^{\theta_1 t}(\theta_3-\theta_2)\\ +\left\{1+2\nu\zeta\xi+\frac{\nu\xi}{t_c}+(2\zeta+\nu\xi)\theta_2+\theta_2^2\right\}e^{\theta_2 t}(\theta_1-\theta_3)\\ +\left\{1+2\nu\zeta\xi+\frac{\nu\xi}{t_c}+(2\zeta+\nu\xi)\theta_3+\theta_3^2\right\}e^{\theta_3 t}(\theta_2-\theta_1)\end{bmatrix}$$

$$+ \frac{\xi}{(\theta_1-\theta_2)(\theta_2-\theta_3)(\theta_3-\theta_1)}\begin{bmatrix}(\theta_3-\theta_2)\left(\frac{1}{\theta_1}-\frac{1}{t_c\theta_1^2}\right)e^{\theta_1 t}\\ +(\theta_1-\theta_3)\left(\frac{1}{\theta_2}-\frac{1}{t_c\theta_2^2}\right)e^{\theta_2 t}\\ +(\theta_2-\theta_1)\left(\frac{1}{\theta_3}-\frac{1}{t_c\theta_3^2}\right)e^{\theta_3 t}\\ -\left(1-\frac{t}{t_c}\right)\left(\frac{\theta_3-\theta_2}{\theta_1}+\frac{\theta_1-\theta_3}{\theta_2}+\frac{\theta_2-\theta_1}{\theta_3}\right)\\ +\frac{1}{t_c}\left(\frac{\theta_3-\theta_2}{\theta_1^2}+\frac{\theta_1-\theta_3}{\theta_2^2}+\frac{\theta_2-\theta_1}{\theta_3^2}\right)\end{bmatrix}$$

As in Section 3.6.2, the requirement for the ram to rest at the desired position, y_d, is that at complete closure of valves the following is satisfied:

$$\ddot{y}_{t_c} + 2\zeta\dot{y}_{t_c} + y_{t_c} = y_d$$

This gives the switching function for this type of valve.

3.6.5. *Effects of Externally Applied Load*

If there is an externally applied load, then equation (3.19) will hold and the equation (3.26) becomes,

$$q_1 = K_1 x \sqrt{\tfrac{1}{2}}\left[1 - \tfrac{1}{2}\left(\frac{ML_r}{P_s A_1 T_r^2}\frac{d^2y}{dt^2} + \frac{BL_r}{P_s A_1 T_r}\frac{dy}{dt} + \frac{F}{P_s A_1}\right)\right] \qquad (3.33)$$

Combining (3.19), (3.24) and (3.33) gives,

$$2K_1 x \sqrt{\tfrac{1}{2}}\left[1 - \tfrac{1}{2}\left(\frac{ML_r}{P_s A_1 T_r^2}\frac{d^2y}{dt^2} + \frac{BL_r}{P_s A_1 T_r}\frac{dy}{dt}\right) - \frac{F}{P_s A_1}\right]$$

$$= 2\frac{dy}{dt} + \frac{P_s}{\beta}\left[\frac{ML_r}{P_s A_1 T_r}\frac{d^3y}{dt^3} + \frac{BL_r}{P_s A_1 T_r}\frac{d^2y}{dt^2}\right]$$

Rearranging gives,

$$\frac{ML_r}{\beta A_1 T_r^2}\frac{d^3y}{dt^3} + \left[\frac{BL_r}{\beta A_1 T_r} + \frac{ML_r}{P_s A_1 T_r^2} K_1 x \sqrt{\tfrac{1}{2}}\right]\frac{d^2y}{dt^2} + \left[2 + \frac{BL_r}{P_s A_1 T_r} K_1 x \sqrt{\tfrac{1}{2}}\right]\frac{dy}{dt}$$

$$= 2K_1 x \sqrt{\tfrac{1}{2}}\left(1 - \tfrac{1}{2}\frac{F}{P_s A_1}\right)$$

$$\therefore \frac{d^3y}{dt^3} + (2\zeta + \nu\xi x)\frac{d^2y}{dt^2} + (1 + 2\nu\zeta\xi x)\frac{dy}{dt} = \xi\left(1 - \tfrac{1}{2}\frac{F}{P_s A_1}\right)x \qquad (3.34)$$

By comparing this equation with equation (3.28), it can be seen that the externally applied load only affects the coefficient of the constant term in the switching function. However, it does not have any effect on the switching function for ideal ON-OFF valves since the valves will be closed immediately on the application of the switching signal.

3.7. Derivation of the Switching Function for an Asymmetric System

If the ram is at its reference position, i.e. y = O, and is non-cavitative, then with a small change in the ram position due to the fluid flow will be described by:-

$$q_1 = K_1 x \sqrt{1 - p_1} \; ; \quad q_2 = -K_1 x \sqrt{p_2}$$

and

$$q_1 = \dot{y} + \frac{P_s}{\beta} \dot{p}_1 \qquad (y \approx 0 \text{ for small movement})$$

$$q = -a\dot{y} + \frac{P_s}{\beta} a\dot{p}_2$$

the dot (•) represents the operator $\frac{d}{dt}$

therefore,

$$q_1 - q_2 = (1 + a)\dot{y} + \frac{P_s}{\beta} (\dot{p}_1 - a\dot{p}_2)$$

Equation (3.19) can be written as the following by neglecting the externally applied force and external springs:

$$p_1 - a\, p_2 = \alpha_m \ddot{y} + \alpha_b \dot{y}$$

where $\alpha_m = \dfrac{ML_r}{P_s A_1 T_r^2}$

$$\alpha_b = \frac{BL_r}{P_s A_1 T_r}$$

$$\therefore q_1 - q_2 = (1 + a)\, \dot{y} + \frac{P_s}{\beta} (\alpha_m \dddot{y} + \alpha_b \ddot{y}) \qquad (3.35)$$

Because of the small movement of the ram, an approximation can be obtained by neglecting the compressibility effect[32], and the following relationship is obtained:-

$$aq_1 \approx -q_2$$

$$\therefore aK_1 x \sqrt{1 - p_1} \approx K_1 x \sqrt{p_2}$$

$$\therefore p_1 + \frac{p_2}{a^2} = 1$$

With these substitutions for p_1 and p_2, from (3.35) gives,

$$q_1 - q_2 \approx K_1 x \sqrt{1 - p_1}\,(1 + a) \qquad (3.36)$$

$$= \frac{1 + a}{\sqrt{1 + a^3}} K_1 x \left[1 - (p_1 - a p_2)\right]$$

By Taylor's series expansion gives,

$$(1 + a)\dot{y} + \frac{P_s}{\beta}(\alpha_m \dddot{y} + \alpha_b \ddot{y}) = \frac{1 + a}{\sqrt{1 + a^3}} K_1 x (1 - \tfrac{1}{2}\alpha_m \ddot{y} - \tfrac{1}{2}\alpha_b \dot{y})$$

Rearranging to obtain:

$$\frac{P_s}{\beta}\alpha_m \dddot{y} + \left[\frac{P_s}{\beta}\alpha_b + \frac{1 + a}{\sqrt{1 + a^3}} K_1 x \tfrac{1}{2}\alpha_m\right]\ddot{y} + \left[\frac{1}{\sqrt{1 + a^3}} K_1 x \alpha_b \tfrac{1}{2} + 1\right](1 + a)\dot{y}$$

$$= \frac{1 + a}{\sqrt{1 + a^3}} K_1 x \qquad (3.37)$$

For the symmetric case, a = 1, (3.37) can be reduced to:

$$\frac{P_s}{\beta}\alpha_m \dddot{y} + \left[\frac{P_s}{\beta}\alpha_b + \frac{1}{\sqrt{2}} K_1 x \alpha_m\right]\ddot{y} + \left[\frac{1}{\sqrt{2}} K_1 x \alpha_b + 2\right]\dot{y} = \frac{2}{\sqrt{2}} K_1 x$$

this is the same as (3.27).

Hence the switching functions for various types of valves can be obtained by the similar procedures as in the symmetric case but with the coefficients:

$$\xi = \frac{1 + a}{2\sqrt{1+a^3}} K_1$$

$$\nu = \frac{\beta}{P_s}$$

$$\zeta = \frac{BT_r}{2M}$$

3.8. Simulation Studies

The system behaviours have been simulated on a desk-top computer, HP 9845B, with the switching functions for the valve types. A modified Euler method is used to solve the differential equations [41], the computer program for this simulation study can be found in Appendix C.

Figures 3.2 and 3.3 predict the non-dimensional ram displacement* responses of the symmetric systems with ideal ON-OFF valves and the inertial masses varying from 500 kg to 2000 kg in the non-dimensional time domain** and the physical time domain respectively. The ram will take approximately the same amount of physical time to reach the same desired position. This can be explained by the fact that the same amount of fluid has been displaced to give the same ram displacement, although the response for a lighter system is faster than the one with a higher inertial mass. The dimensional ram velocity and acceleration are slightly smaller in a heavier system and the switching of the valves has to occur earlier with reference to the desired position. However, the non-dimensional times taken are significantly different because the reference time varies with the actuator's natural hydraulic frequency, which depends upon the inertial mass (the heavier the mass, the lower is the frequency).

---

* Recall that the ram displacement is non-dimensionalised by the reference length, L_r which will be half-stroke in the case of a symmetrical system. See Section 3.3.2.

** The time domain is non-dimensionalised by the reference time, T_r, which is the inverse of the natural hydraulic frequency of the ram at its reference position. See Section 3.3.4.

The switching function for an ideal ON-OFF valve system is:

$$\ddot{y} + 2\zeta\dot{y} + y = y_d,$$

therefore, the damping factor, ζ, plays an important role in determining the switching of the valves. If one keeps the damping factor constant (in the non-dimensional unit), the switching will be determined by the ram velocity and its acceleration for the same desired position. However, these two quantities will vary according to the inertial mass: the higher the loading, the lower will be these quantities (in the physical units) for the same given system supply and exhaust pressures. This is because the pressure differences across the valves will be reduced to give a higher driving force in moving a heavier ram, hence a smaller fluid flow rate will result. It should be recalled that the non-dimensional time is also inertial load dependant (the reference time will increase with the inertial mass), therefore the non-dimensional velocity and acceleration will be higher for a system with a heavier inertial load, this is one of the reasons why the switching for the system with a heavier inertial load has to occur earlier. It can also be seen that the overshoot is higher for a larger inertial mass system since the kinetic energy is higher at the switching point for a heavier system, hence the switching occurs earlier for a heavier system. Figures 3.4 and 3.5 review the similar behaviours for the asymmetrical systems.

The effects of damping on the system behaviour can be seen in Figures 3.6 and 3.7: the smaller is the damping, the more will be the overshoot and the oscillations before the

ram reaches its equilibrium state. According to the switching functions, the higher the damping, the earlier the valves should be closed for the same desired position with the same initial conditions, if the other quantities remain unchanged. The distance of travel is normally long enough for the ram to achieve its terminal velocity before closing the valves, therefore the determination of the switching point depends highly upon the velocity term, $2\zeta\dot{y}$. Although the terminal velocity will decrease with an increase in the damping factor, the effect of damping has larger influence on the switching point if the system is underdamped. As the damping increases, the effects of lower velocity will become more significant. This can be reviewed from the graphs (Figures 3.6 and 3.7) that apparantly an exponential curve might be constructed through the switching points.

If one examines the definition of the damping factor, ζ, he will find that the damping factor can be increased by decreasing the inertial mass, hence an increase in the inertial loading is expected to have a similar effect as decreasing the damping factor. However, this damping effect is outweighed by the effects of the fluid flow rate, though a system with a heavier mass will take less non-dimensional time to reach the switching point, since the reference time is different.

The effects of the time delay on the ON-OFF valves are shown in Figures 3.8 and 3.9 for the symmetric and asymmetric systems respectively. It will take a longer time

to reach the desired position for a system with valves of a longer time delay, since these valves will take longer to respond to the switching signals. Consequently, the switching signal has to be applied much earlier for the valves with a longer time delay.

Figures 3.10 and 3.11 show the similar effects of 'linear' valves with various operating times. However, the intermediate oscillations are fewer because the valves are closed gradually and the pressures within the chambers do not change rapidly. The graphs indicate that the final errors for these systems are higher, it may be explained by the fact that the switching functions are derived based on the assumption that the higher order terms in the Taylor's series expansion can be neglected. Therefore, in deriving the switching functions, the estimated fluid flow is less than that obtained in the simulation. In other words, on deriving the switching functions, the flow through the valve is approximately linear, while the simulated flow is non-linear. This can be seen from Figure 3.12 that the non-linearity is more significant on the side with the piston rod (i.e. the side with a smaller sectional area). Figures 3.13 and 3.14 show that the deviation of the simulated flow from the linearity becomes larger as the operating time increases. This is why the final error increases as the valves take longer time to operate. However, the magnitudes of these errors are in the order of 10^{-2} of the reference length, and are insignificant for this application: mining boom positional control. Therefore, correction functions are not introduced so as to keep the switching functions simple.

Figures 3.15 and 3.16 show the effects of the externally applied force: the initial overshoot will be higher when the force is lower, and it takes a longer time to reach the desired position if the force is higher. This is because the force is a measure of the resistance to the ram movement: the higher the force, the lower will be the velocity and acceleration. Consequently, the switching should occur earlier (with reference to the desired position) for a system with a lower externally applied force.

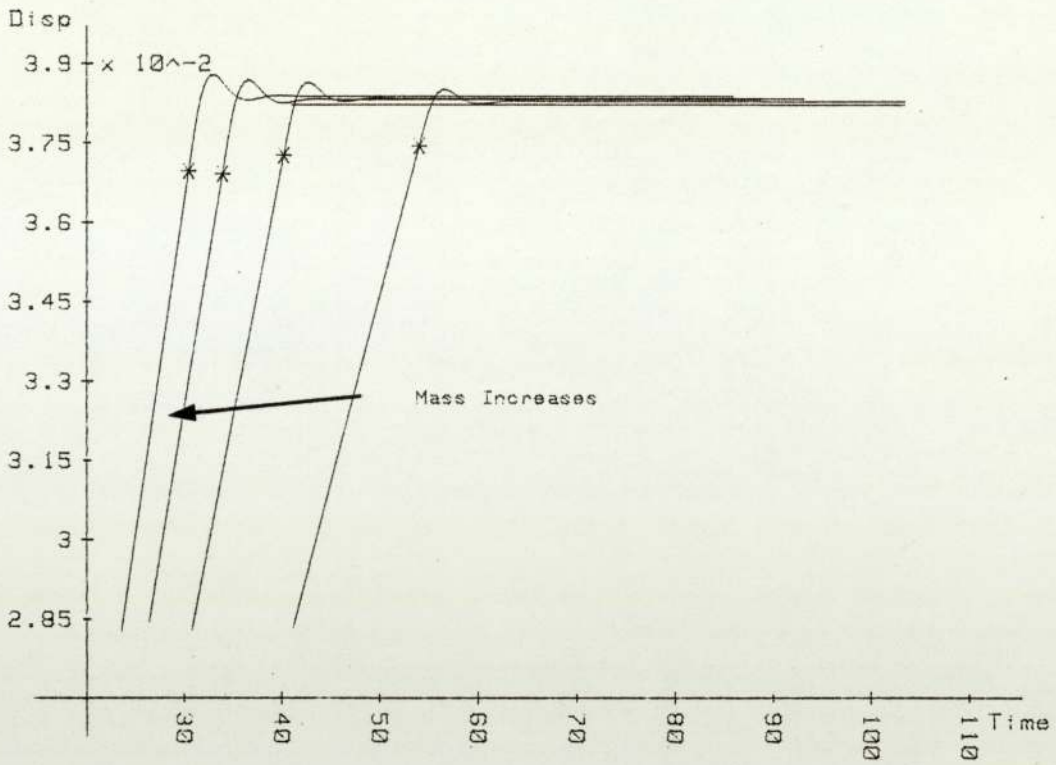


Fig. 3.2 Non-dimensional Ram Displacement - Non-dimensional Time Response
Symmetrical Systems With Different Inertial Masses

ON-OFF Valves With Time-Delay = 0.0000E+00 (0.0 mSec.)
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

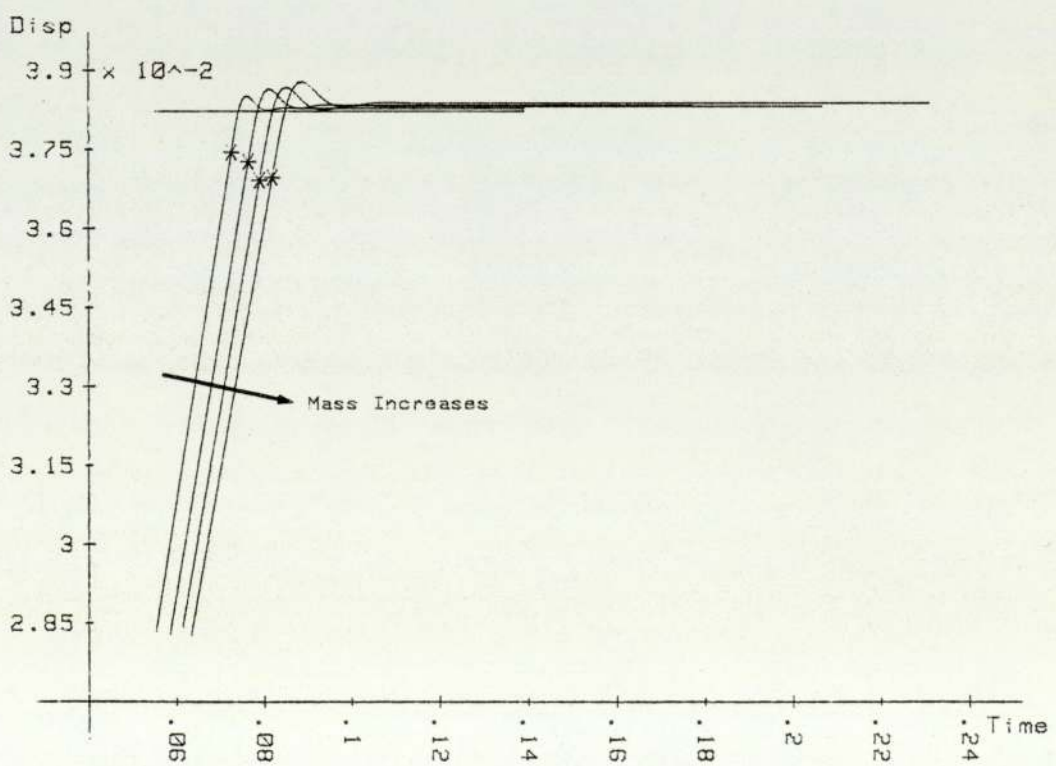


Fig. 3.3 Non-dimensional Ram Displacement - Dimensional Time Response
Symmetrical Systems With Different Inertial Masses

ON-OFF Valves With Time-Delay = 0.0 mSec.
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

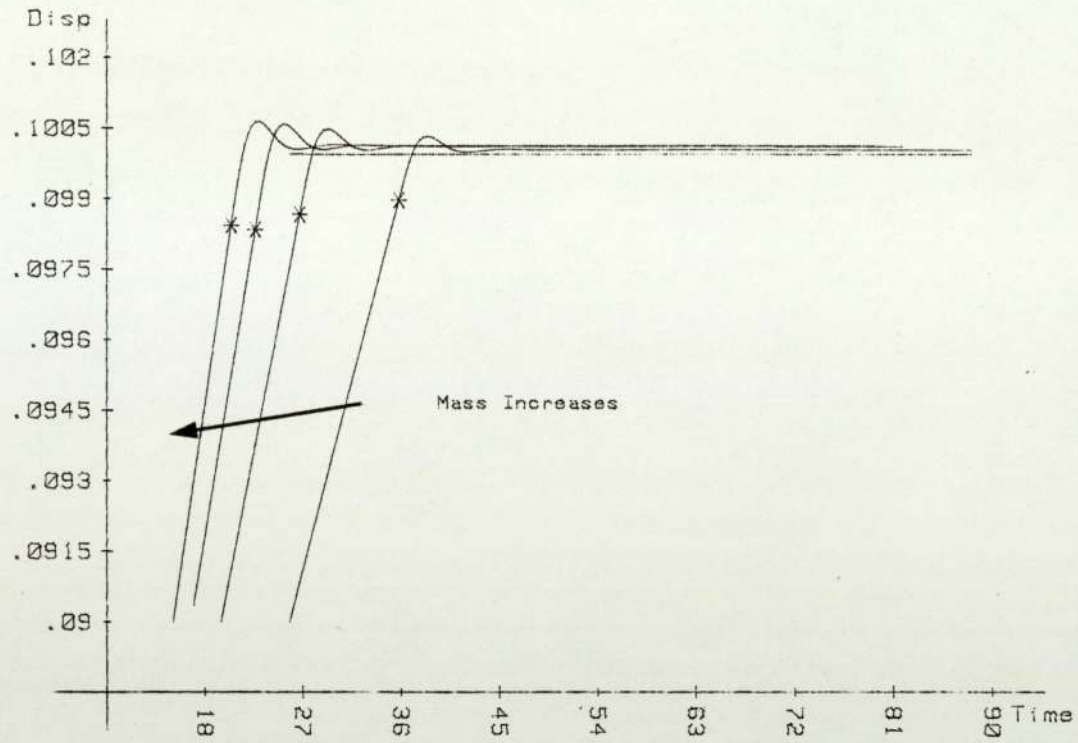


Fig. 3.4 Non-dimensional Ram Displacement - Non-dimensional Time Response
Asymmetrical Systems With Different Inertial Masses
Area Ratio = 6.400E-01
ON-OFF Valves With Time-Delay = 0.0000E+00 (0.0 mSec.)
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

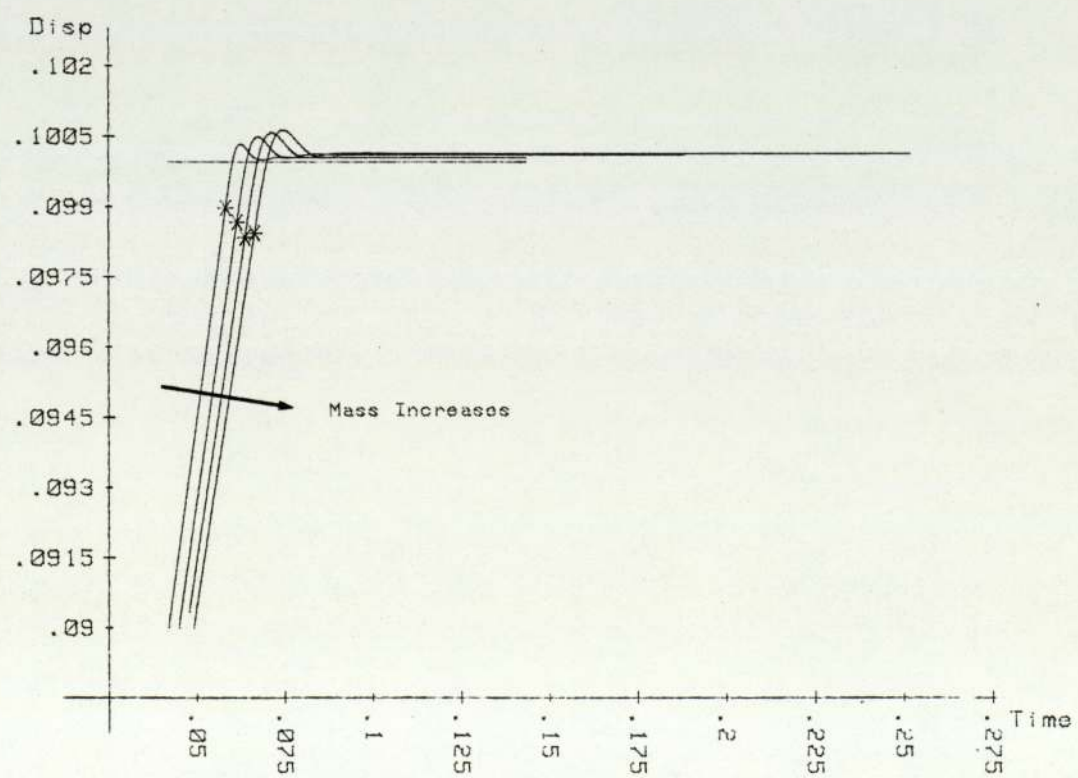


Fig. 3.5 Non-dimensional Ram Displacement - Dimensional Time Response
Asymmetrical Systems With Different Inertial Masses
Area Ratio = 6.400E-01
ON-OFF Valves With Time-Delay = 0.0 mSec.
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

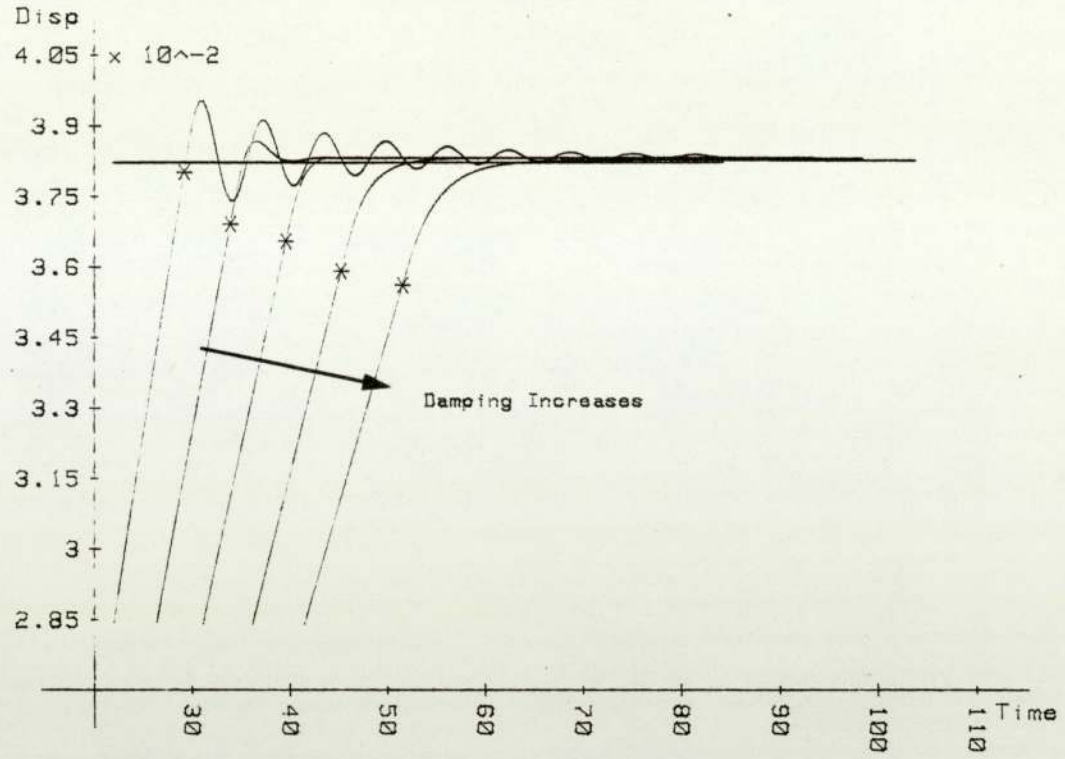


Fig. 3.6 Non-dimensional Ram Displacement - Non-dimensional Time Response
Symmetrical Systems With Inertial Masses = 1500 Kg.

ON-OFF Valves With Time-Delay = 0.0000E+00 (0.0 mSec.)
External Force Applied = 0.0000E+00 Effective Damping Factor Varies

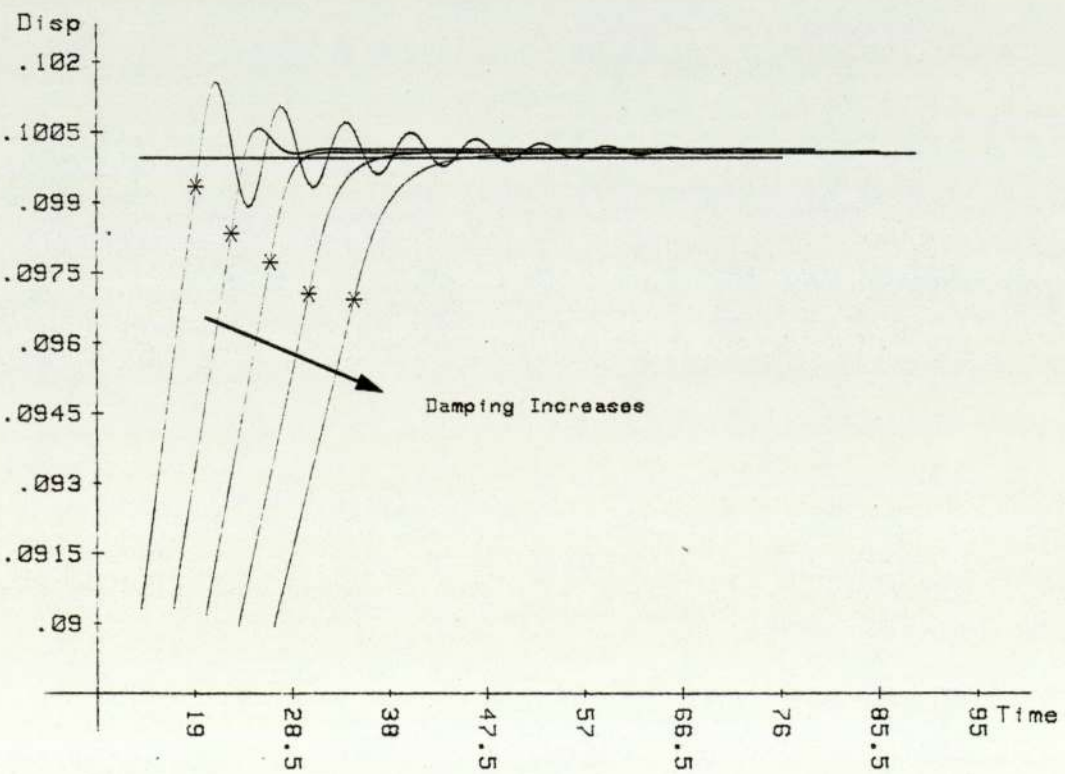


Fig. 3.7 Non-dimensional Ram Displacement - Non-dimensional Time Response
Asymmetrical Systems With Inertial Masses = 1500 Kg.
Area Ratio = 6.400E-01
ON-OFF Valves With Time-Delay = 0.0000E+00 (0.0 mSec.)
External Force Applied = 0.0000E+00 Effective Damping Factor Varies

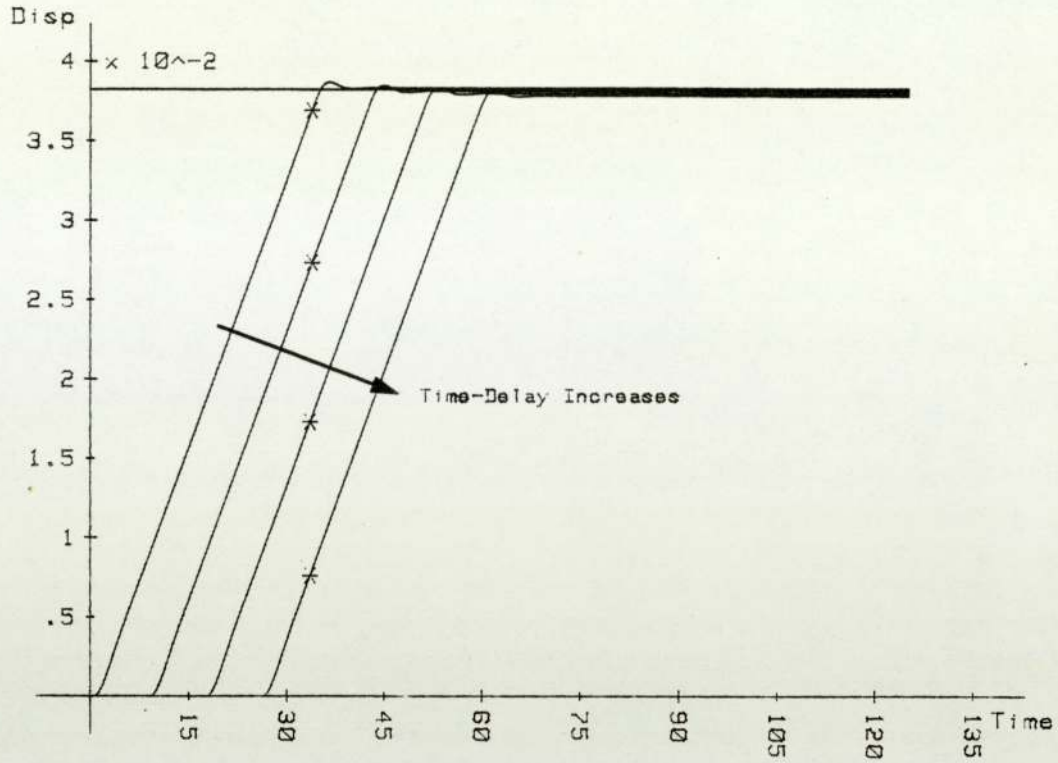


Fig. 3.8 Non-dimensional Ram Displacement - Non-dimensional Time Response
Symmetrical Systems With Inertial Mass = 1500 Kg.

ON-OFF Valves With Different Time-Delay
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

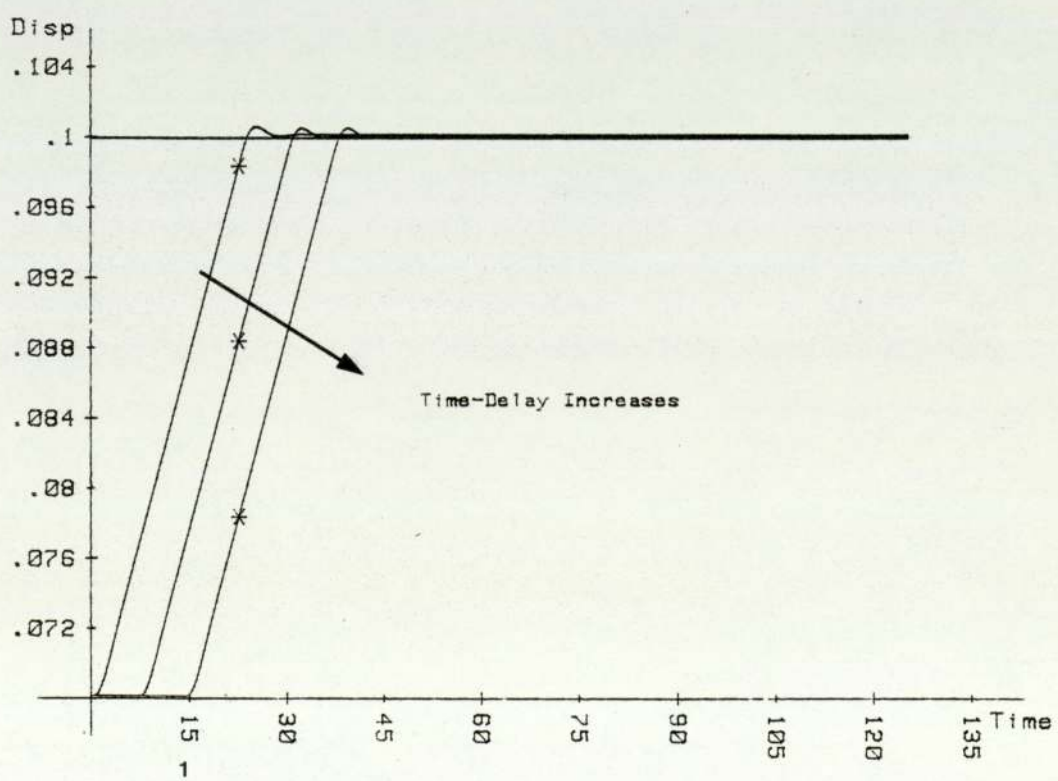


Fig. 3.9 Non-dimensional Ram Displacement - Non-dimensional Time Response
Asymmetrical Systems With Inertial Mass = 1500 Kg.
Area Ratio = 6.400E-01
ON-OFF Valves With Different Time-Delay
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

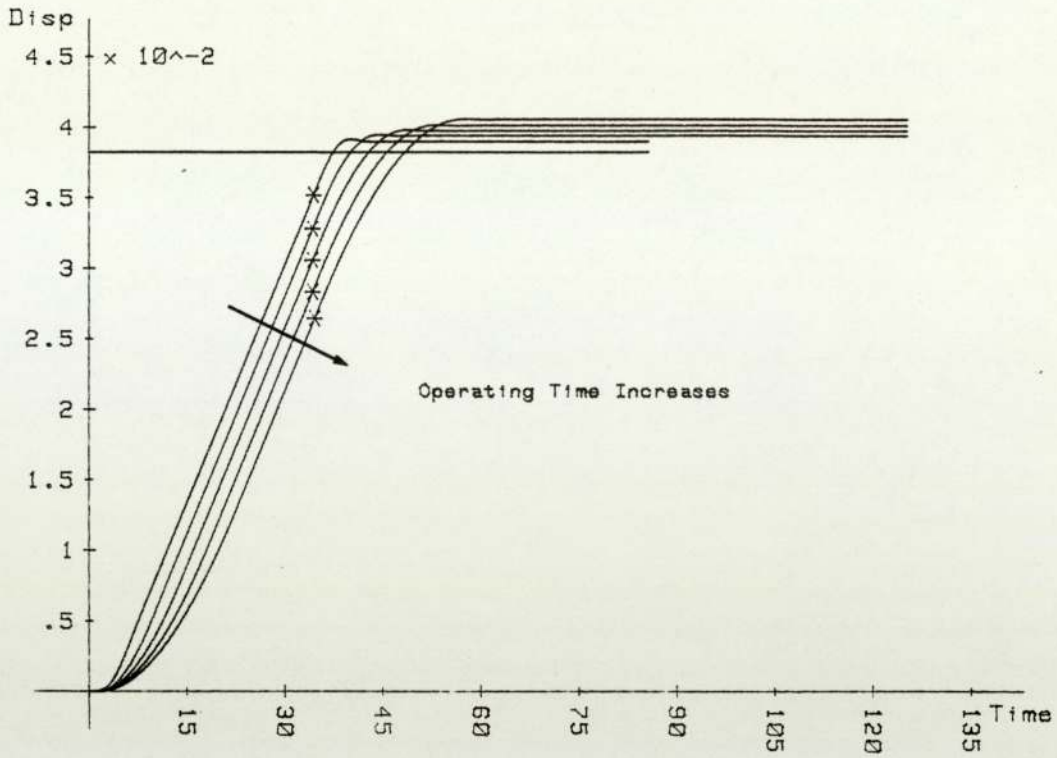


Fig. 3.10 Non-dimensional Ram Displacement - Non-dimensional Time Response
Symmetrical Systems With Inertial Mass = 1500 Kg.

Linear Valves With Different Operating Times
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

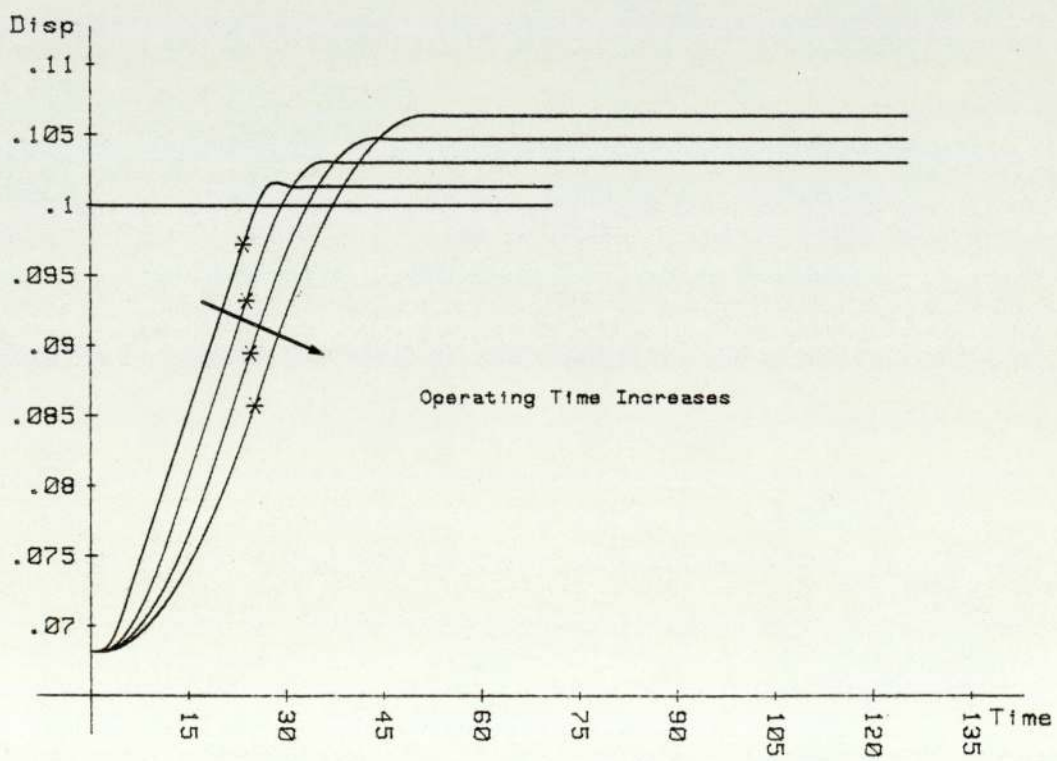


Fig. 3.11 Non-dimensional Ram Displacement - Non-dimensional Time Response
Asymmetrical Systems With Inertial Mass = 1500 Kg.
Area Ratio = 6.400E-01
Linear Valves With Different Operating Times
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

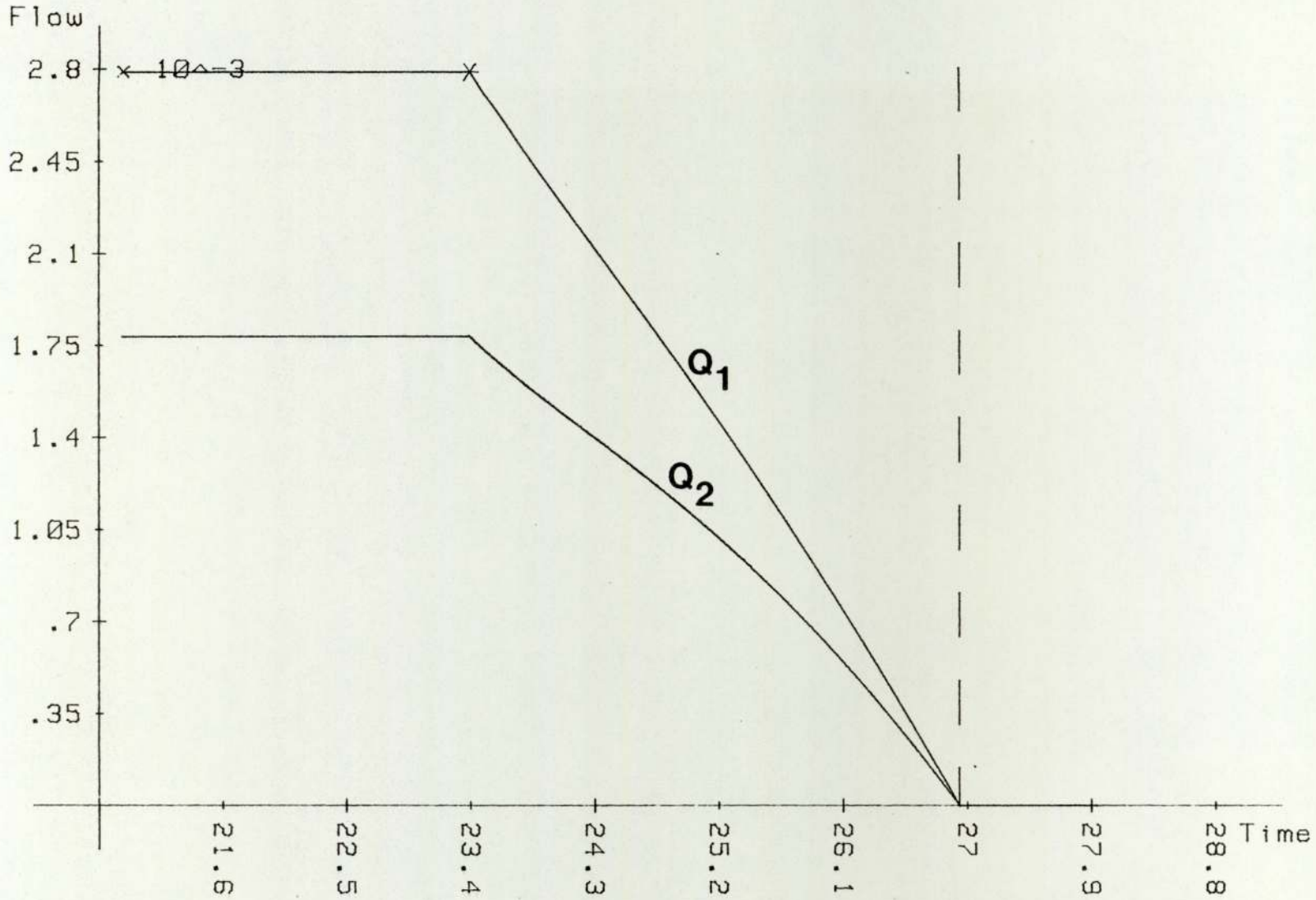


Fig. 3.12 Fluid Flow Rate - Time Response (Non-dimensional) (Mass = 1500 Kg)
Asymmetrical System With Area Ratio = 6.400E-01
Linearly Operated Valves With Time = 3.5497E+00 (10.0 mSec.)
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

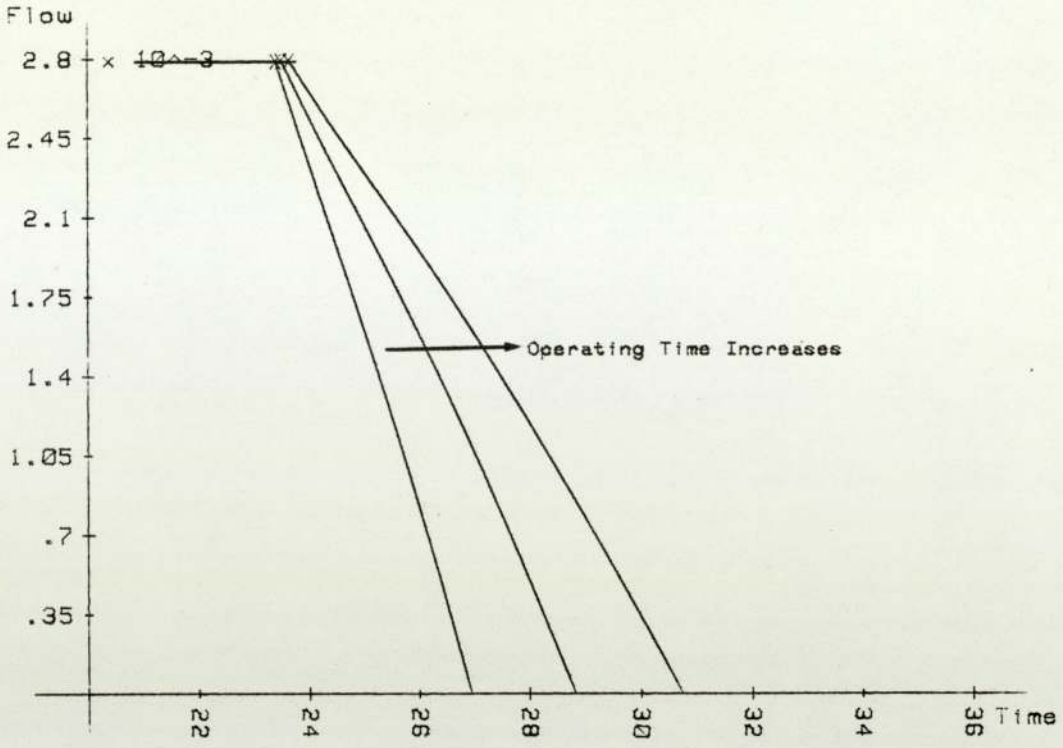


Fig. 3.13 Fluid Flow Rate Through Valve 1 - Time Response (Non-dimensional)
Asymmetrical Systems With Inertial Mass = 1500 Kg.
Area Ratio = 6.400E-01
Linear Valves With Different Operating Times
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

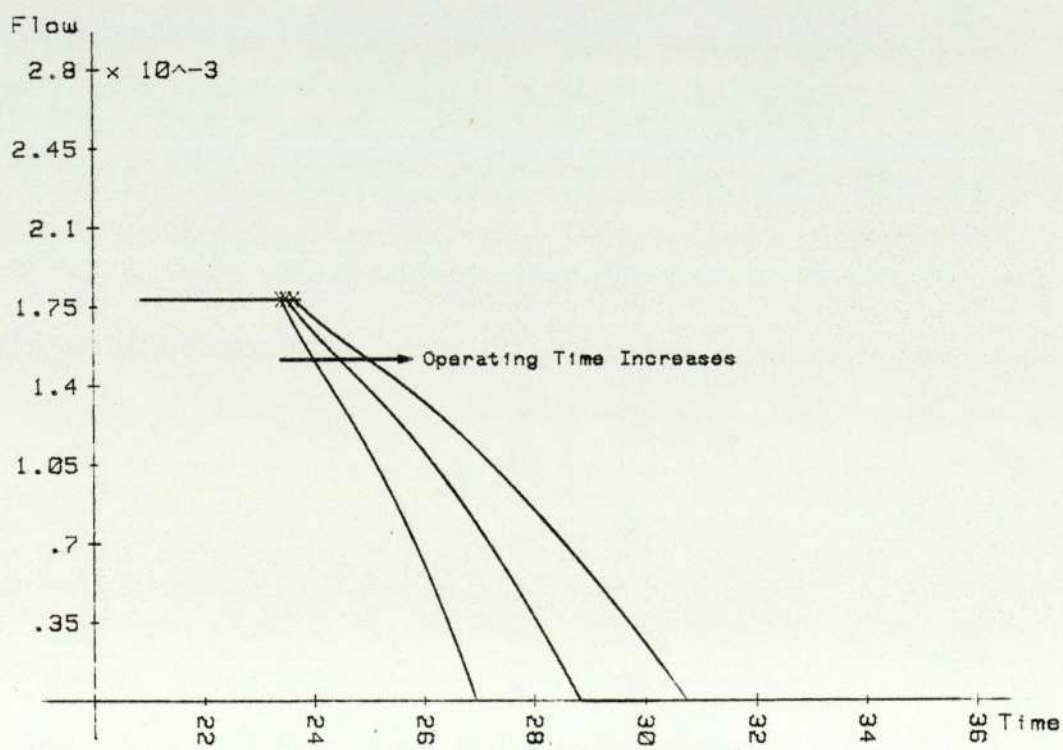


Fig. 3.14 Fluid Flow Rate Through Valve 2 - Time Response (Non-dimensional)
Asymmetrical Systems With Inertial Mass = 1500 Kg.
Area Ratio = 6.400E-01
Linear Valves With Different Operating Times
External Force Applied = 0.0000E+00 Effective Damping Factor = .50

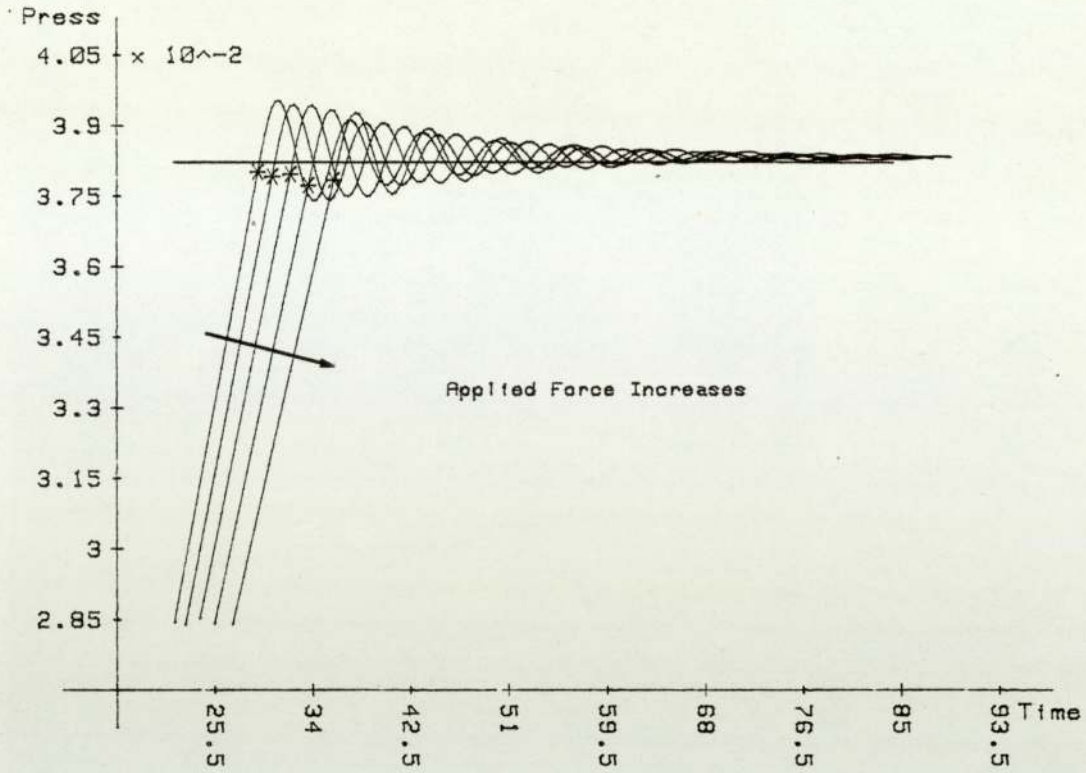


Fig. 3.15 Non-dimensional Ram Displacement - Non-dimensional Time Response
Symmetrical Systems With Inertial Masses = 1500 Kg.

ON-OFF Valves With Time-Delay = 0.0000E+00 (0.0 mSec.)
External Force Applied Varies Effective Damping Factor = .10

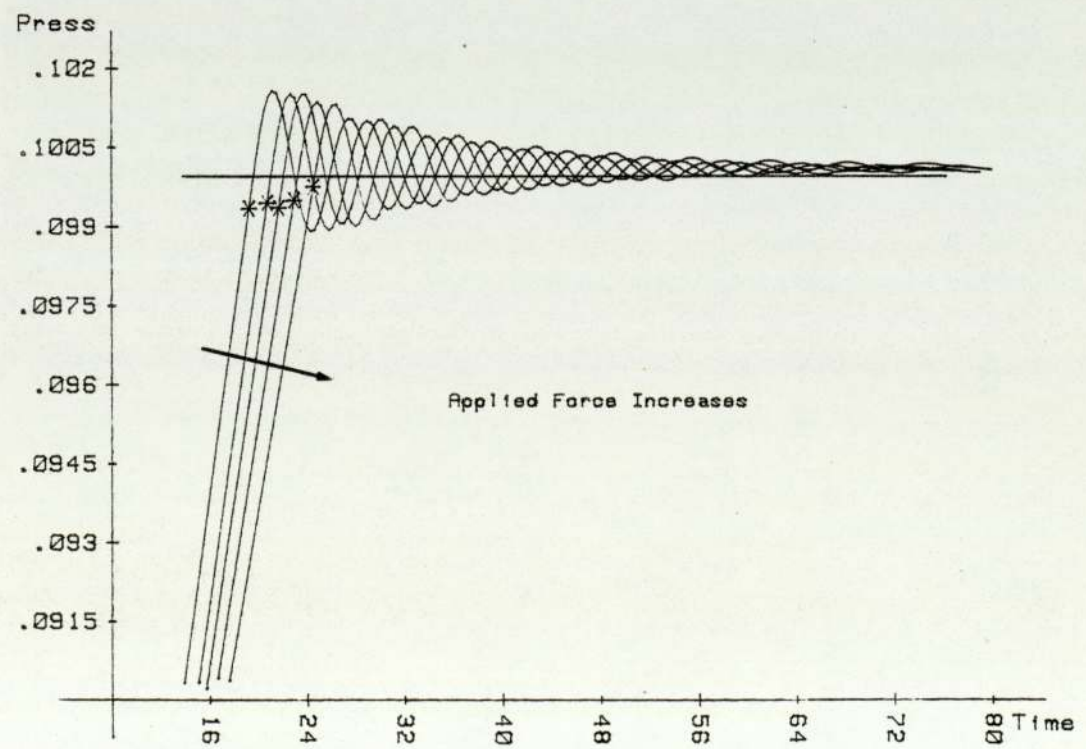


Fig. 3.16 Non-dimensional Ram Displacement - Non-dimensional Time Response
Asymmetrical Systems With Inertial Masses = 1500 Kg.
Area Ratio = 6.400E-01
ON-OFF Valves With Time-Delay = 0.0000E+00 (0.0 mSec.)
External Force Applied Varies Effective Damping Factor = .10

CHAPTER 4

ARITHMETIC PROCESSING

4.1. Introduction

Arithmetic processing can be performed in either hardware or software. Hardware operation has definite advantage over software in speed. On the other hand, if the software for the mathematical operations is available and is ready for programming into read only memories (ROMs), then a special arithmetic processing unit (APU) is more expensive than the ROMs. For example, in 1982, an APU (Intel-8231A) costs £120 while a 2K erasable programmable read only memory (EPROM), 2716, can be bought for £3.

Using the Intel system design kit, SDK-85, a comparison between the hardware arithmetic processing and the software mathematical operation has been made to see whether the software arithmetic processing is suitable for this boom positional control.

4.2. Function to be Evaluated

In order to account for the fluid compressibility effects, a switching function has been set up to give the positional control. This function is simple if there are no significantly variable forces acting on the system (please refer to Chapter 3). This function is used for each pair of the hydraulic rams: the trunk lifting rams and the arcing rams.

For any pair of rams, closing (switching) of the valves should be taken place when:

$$fn(y_d, y, \dot{y}, \ddot{y}) = \lambda_1 \ddot{y} + \lambda_2 \dot{y} + \lambda_3 y + \lambda_4 y_d + \lambda_5 = 0 \qquad (4.1)$$

where y_d = required ram position for the desired boom position.

y = current ram position.

$\dot{y}$ = current ram velocity.

$\ddot{y}$ = current ram acceleration.

λ_i = coefficients determined by the system parameters; $i = 1,2,3,4,5$.

All the coefficients in (4.1) are dependant upon the system properties (e.g. ram diameters, inertial load, damping, valve characteristics, etc.) and their expressions are complicated. However, they are constants for a particular system and can be evaluated and converted into the binary numbers when the boom is installed. Therefore, in calculating the switching function, there is no need to re-evaluate these coefficients.

4.3. Timing Test Program

The time taken by an APU to evaluate the above function (4.1) is compared with that used by the software evaluation. The actual switching function encountered will be more complicated than the one given above because of the geometry of the boom (since the aim is to control the boom cutter position). This simplified switching function is used to give an insight into the differences in the performance between the two techniques. The block structure of this test program

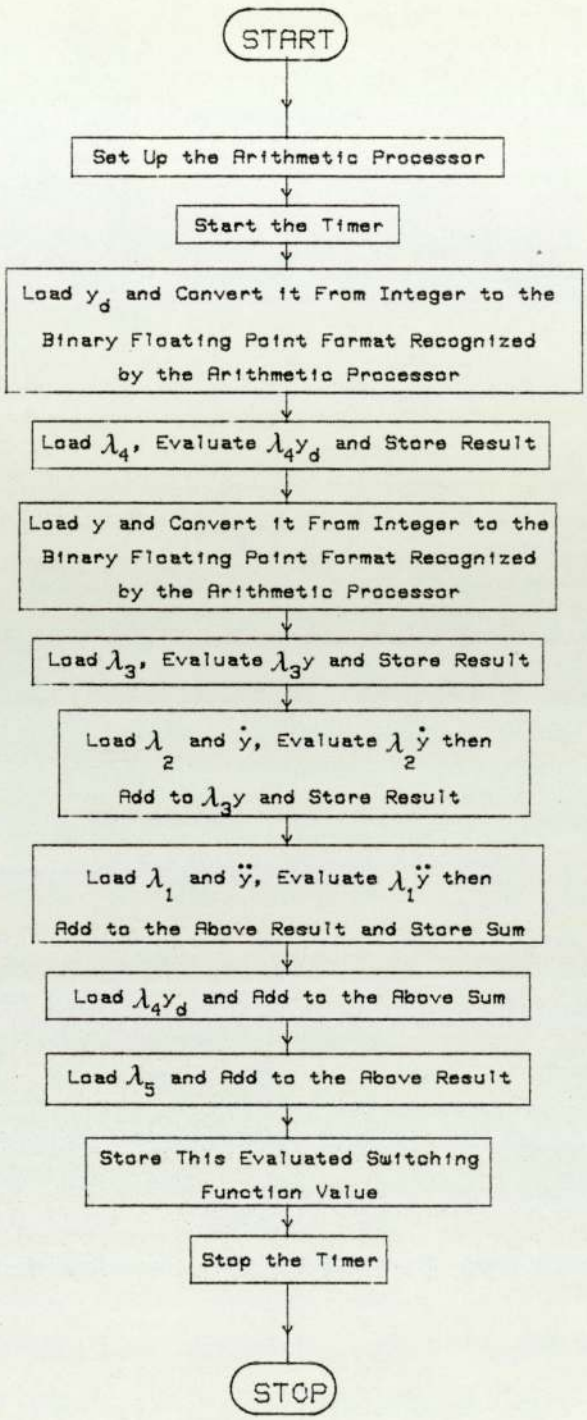


FIG 4.1 Block Structure of the Timing Test Program

is given in Fig. 4.1. It is similar for both arithmetic processors*, hence only one diagram is presented. The actual assembly language programs are given in Appendix E.

4.4. Explanation of the Test Program

4.4.1. *Set up the 'Arithmetic Processor'*

For the hardware arithmetic processor, the APU, the set up procedure is simply applying a reset signal to the RESET pin of this APU [42]. This operation can be synchronised when the system is reset, since the microprocessor will issue a RESET OUT signal on recognising an active RESET IN signal. On the other hand, in case of the software arithmetic processor, the FPAL, setting is done by calling a set up procedure to initialise the floating point record (FPR) [43].

4.4.2. *Timing*

It would be unfair to start the timing at the beginning of the program, because of the difference in the set up procedures, therefore the following timing scheme is used. When the actual procedure for evaluating the switching function is called, the timing will be started and it will be stopped on exit from that procedure. Hence only the time taken for evaluating the simplified switching is measured. This timing is done using the hardware timer resided in the 8155 on the SDK-85.

---

* A term arithmetic processor is introduced to represent either the special arithmetic processing unit (APU) for hardware operations or the Intel 8080/85 floating point library (FPAL) routines for software operations.

4.4.3. *Format Conversion*

In the data capturing system, all the positional information is provided as single byte integers (OOH to FFH, or OD to 255D), therefore conversion from the integer format into the binary floating point format recognised by the arithmetic processor must be performed before any floating point arithmetic processing can be applied. However, all the other information is provided in the binary floating point format.

4.4.3. *Evaluation of the Switching Function*

The function evaluation is done as in the expression (4.1) for the function $fn(y_d, y, \dot{y}, \ddot{y})$ and the intermediate results are stored in temporary memory locations. This temporary storing of the results is required by the FPAL because it does not have any stack for keeping the intermediate results. On the other hand, the APU has a 16-byte data stack, therefore this storing operation may be omitted to increase the speed provided that the arithmetic operation performed does not destroy the current data stack contents [42].

4.5. Binary Floating Point Format

Floating point representation (format) is more commonly known as the scientific notation in the decimal number system. The reason for using the floating point notation is to increase the number range for the calculations so that very large or small numbers can be handled. The floating point format consists of three fields, namely, the sign, mantissa and exponent. The sign field indicates

the sign of the number represented, while the mantissa and the exponent fields give the magnitude of this number. For example, in the decimal number system, the number, 123400, can be represented as +1.234E 5 in the scientific notation. In this format, the sign field is '+' indicating a positive number with a mantissa of '1.234' giving the four significant figures and the exponent field '5', meaning that the decimal point should be shifted five places to the right yielding the actual number's magnitude, i.e. the actual magnitude of this number is obtained by multiplying 10^5 to the mantissa.

Since the microprocessor is a binary digital device, the binary number system must be used. In a binary number system, all the fields are in a binary format with the exponent field indicating the power of two to be multiplied in order to obtain the actual magnitude. Both the APU and the FPAL recognise a 32-bit (4-byte) floating point format consisting of these three fields. These fields appear in this sequence:

Sign exponent mantissa

A floating point zero is represented by thirty two 0-bits in both the APU and FPAL floating point formats. However, non-zero numbers are represented differently in these two formats.

For non-zero numbers, in both formats, the sign field is the most significant bit (i.e. bit-31). A zero in this bit position indicates a positive or non-negative number while a one means that the number is negative.

Decimal Number	Decimal Floating Point Format	APU's Binary Floating Point Format			FPAL's Binary Floating Point Format		
		Sign 1 Bit	Exponent 7 Bits	Mantissa 24 Bits	Sign 1 Bit	Exponent 8 Bits	Mantissa 23 Bits
256	2.56 E 2	0	0001001	10000000 00000000 00000000	0	10000111	0000000 00000000 00000000
-256	-2.56 E 2	1	0001001	10000000 00000000 00000000	1	10000111	0000000 00000000 00000000
0.015625	1.5625 E -2	0	1111011	10000000 00000000 00000000	0	01111001	0000000 00000000 00000000
-0.015625	-1.5625 E -2	1	1111011	10000000 00000000 00000000	1	01111001	0000000 00000000 00000000
11	1.1 E 1	0	0000100	10110000 00000000 00000000	0	10000010	0110000 00000000 00000000

Table 4.1 Examples of the Number Formats

There are seven bits in the exponent field of the APU's floating point format representing the power of two to be multiplied to give the magnitude of this number. These seven bits are given in two's complement for the negative power of 2. The exponent field of the FPAL recognised format consists of 8 bits and is offset by a bias of $2^7 - 1$ (i.e. the indicated exponent is 127D larger than the actual value).

Both formats have 24 significant bits. In the APU format, the mantissa field consists of the remaining 24 bits and represents the fraction field beginning with a 1-bit, and a binary point is assumed to preceed the most significant fraction bit (bit-23 of the 4-byte format), i.e., the mantissa in APU's format is a 24-bit binary fraction number in the range [0.1B, 1.0B) or [0.5D, 1.0D). On the other hand, in the FPAL format, an implicit one bit is assumed at the left of the binary point of the fraction field, therefore only 23 bits are needed to be specified for this 24-bit mantissa. The number range of this mantissa field is therefore [1.0B, 10.0B) or [1.0D, 2.0D). A few examples have been given in Table 4.1.

These floating point formats can better be illustrated by the following examples:-

(a) The decimal number 55D can be represented as +5.5E 1 in the decimal floating point format. In the binary number system, it is 110111 B and can be represented in binary floating point format either as $1.10111\ B*10^{101}\ B$

or $0.110111B*10^{110}$ B. Hence in the APU format, 55D is represented as 0 0000110 11011100 00000000 00000000; while in the FPAL format, the exponent field is offset by 127D to the value, 132D, therefore this number (55D) is represented as

0 10000100 1011100 00000000 00000000.

(b) The decimal number 0.3125D can be represented as +3.125E -1 in the decimal floating point format. As a binary number, it is represented as 0.0101B and can be put in the binary floating point format as either $0.101B*10^{-1}$ B or $1.01B*10^{-10}$ B. Hence in the APU format, 0.3125D is represented as 0 1111111 10100000 00000000 00000000; while in the FPAL format, it is 0 01111101 0100000 00000000 00000000 (the exponent field is offset by 127D).

Since the magnitude of a floating point number is obtained by scaling the mantissa with the exponent, the size of the exponent determines the number range which can be represented in the format. The range of the exponent field of the APU format is from -1000000B to 111111B (-64D to 63D), therefore the number range represented in the APU format is

$(-1.0*2^{63}D, -0.5*2^{-64}D], 0, [0.5*2^{-64}D, 1.0*2^{63}D)$.

While the range of the numbers represented in the FPAL format is

$(-2.0*2^{128}D, -1.0*2^{-127}D], 0, [1.0*2^{-127}D, 2.0*2^{128}D)$,

since the range of the exponent field is from -127D to 128D.

4.6 Operation Required

4.6.1 *FPAL*

The evaluation of the switching function is done with the floating point arithmetic routines provided by the FPAL, therefore the mathematical operations are performed by calling the appropriate subroutines. The routines called upon include: two conversions, four multiplications, four additions, four storing operations and two loadings.

4.6.2 *APU*

All the mathematical operations are performed in hardware within the APU. Every mathematical operation is initiated by issuing (writing) a command code to the APU's command port. Since the APU has a 16-byte data stack, storing of intermediate results may be omitted if these mathematical operations do not destroy the current stack contents. However, every operand should be loaded onto the data stack before an operation can be initiated, so there are at least nine loadings to be performed. The numbers of conversions, additions and multiplications are the same as those required in using the FPAL. It has to be pointed out that the time taken for transferring the data from memory to the APU data stack depends upon the hardware configuration of the system. Instructions in the data transfer group are used for a memory mapped APU, while input and output instructions are used for an I/O mapped APU. Normally, the time taken by an instruction in the data transfer group is less than that used by an I/O instruction. For example, an 8085A microprocessor requires ten states to execute an

Operating Condition	Number of Clock Cycles Used	Time Used in mSecs for a 3.072 MHz System Clock	Ratio = Time Used in Software Evaluation / Time Used Under Given Operating Condition
FPAL	13619 D	4.433	1.000
I/O Mapped APU With Storing of Intermediate Results	4706 D	1.532	2.894
I/O Mapped APU Without Storing of Intermediate Results	3276 D	1.066	4.157
Memory Mapped APU With Storing of Intermediate Results	4529 D	1.474	3.007
Memory Mapped APU Without Storing of Intermediate Results	3162 D	1.029	4.307

Table 4.2 Time Taken to Evaluate the Simplified Switching Function Under Various Operating Conditions

IN instruction while it takes only seven states to perform the instruction, MOV A,M.

4.7. Results

Since the actual time used in an arithmetic operation also depends on the values of the operands for both arithmetic processors, the same numerical values were used for this timing test. Timing has been done for both the memory mapped and the I/O mapped APU and the results are presented in Table 4.2. These timings confirm that the hardware operations are considerably faster than the software operations.

4.8. Boom Moving Rate

The time taken for the boom lifting has been measured and the normalised result is shown in Figure 4.2. The boom took 5.5 seconds to move down from the uppest level to the bottom one. The ram displacement was measured using a linear displacement transducer and then this analogue signal is converted into a digital signal by an analogue-to-digital converter (A/D converter). If an 8-bit A/D converter is used, the resolution will be $1/(2^8-1)$ or 1/255 of the full range. If the range is adjusted to cover only the ram stroke, then for every bit change of the A/D reading, the analogue signal should have changed by 1/255 of the whole range. For this particular boom arrangement, the time taken to give this change after the ram has reached its terminal velocity is 21.76 mSec which is comparatively much longer than the time taken to evaluate the simplified switching function.

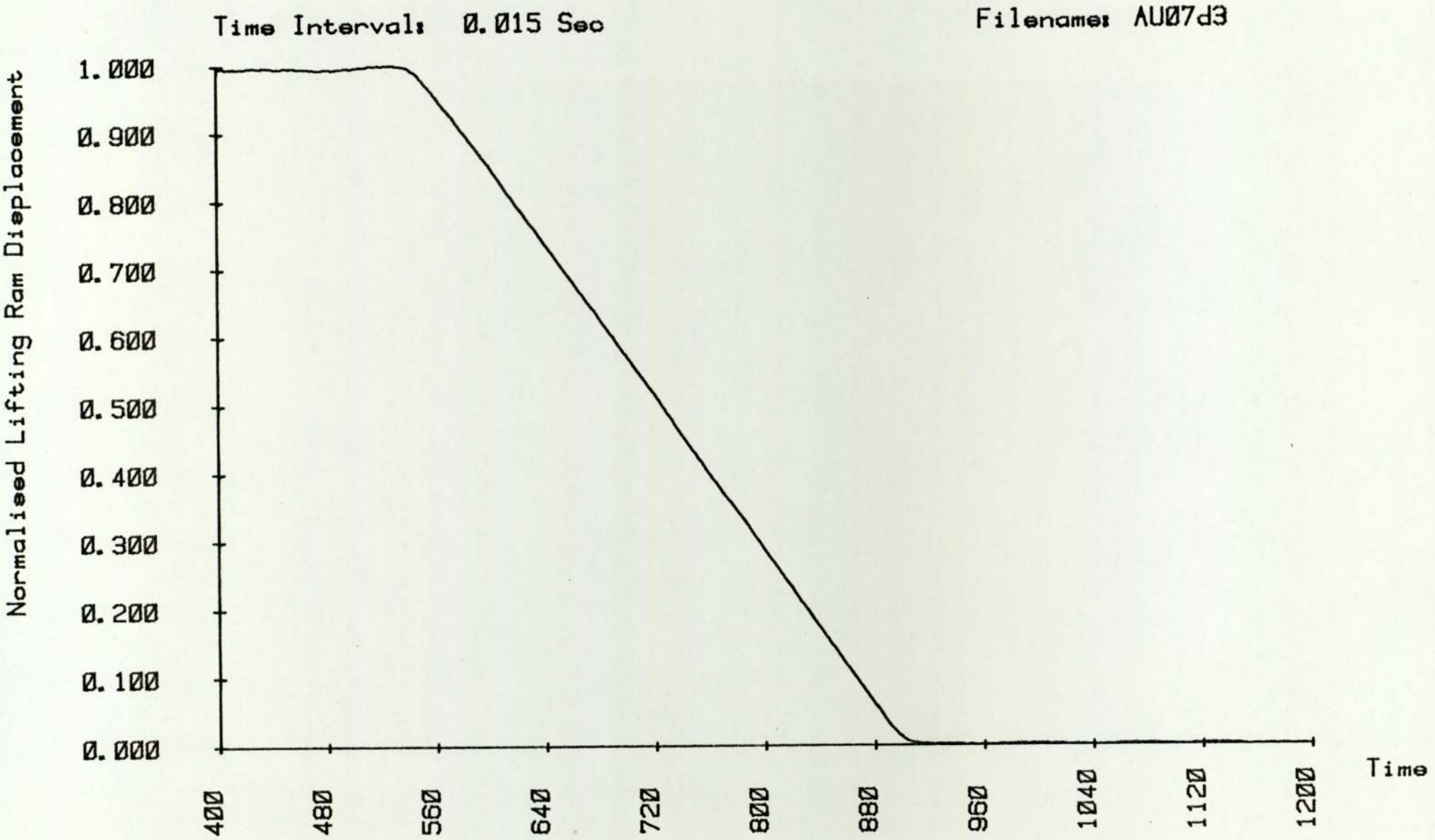


FIG. 4.2 NORMALISED LIFTING RAM DISPLACEMENT - TIME RESPONSE

4.9. Comments

Should the switching function be that simple, the function evaluation can either be done in software or in hardware. Since the hardware operation has no 'apparent' advantage over the software in speed and is more expensive, the software treatment is recommended.

However, in the actual boom system, the desired position will be given for the boom cutter and the required ram position are to be evaluated according to its geometry. This will greatly increase the time taken to calculate the switching function. If more complicated mathematical operations are required, a considerable amount of time will be saved by the hardware APU. A timing test has been done for evaluating the sine function of an angle, the time has been reduced from 45.9 mSec to 1.36 mSec by using the M/M APU (i.e. the APU operates about 34 times faster than the software routines).

Considerable amounts of memories have also been saved when using the APU, even though it is memory mapped at the suggested addresses which amount to 0.5K (512D bytes). For example, the program length of these simple timing tests are 1945D for the software evaluation, 230D bytes for the I/O mapped APU hardware evaluation and 214D bytes for the memory mapped APU hardware operation (in addition to the 0.5K bytes).

Furthermore, with larger sized valves or a higher system supply pressure the boom moving speed can easily be increased. Therefore high speed mathematical operations provided by an APU are strongly recommended for the real system.

CHAPTER 5

CONTROL ALGORITHMS

AND

PROGRAM STRUCTURE

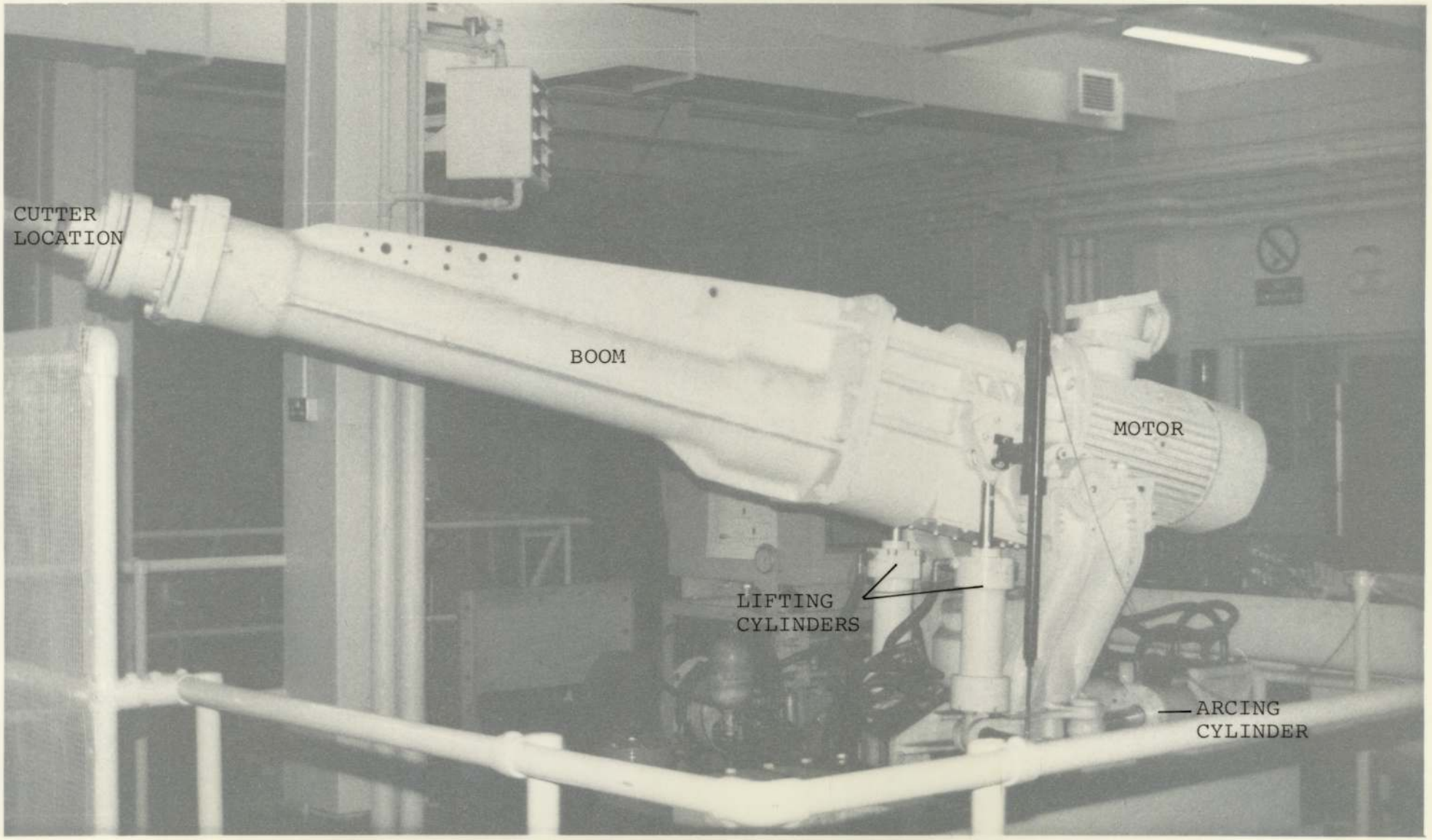


Fig 5.1 The Boom Ripper Assembly under Test

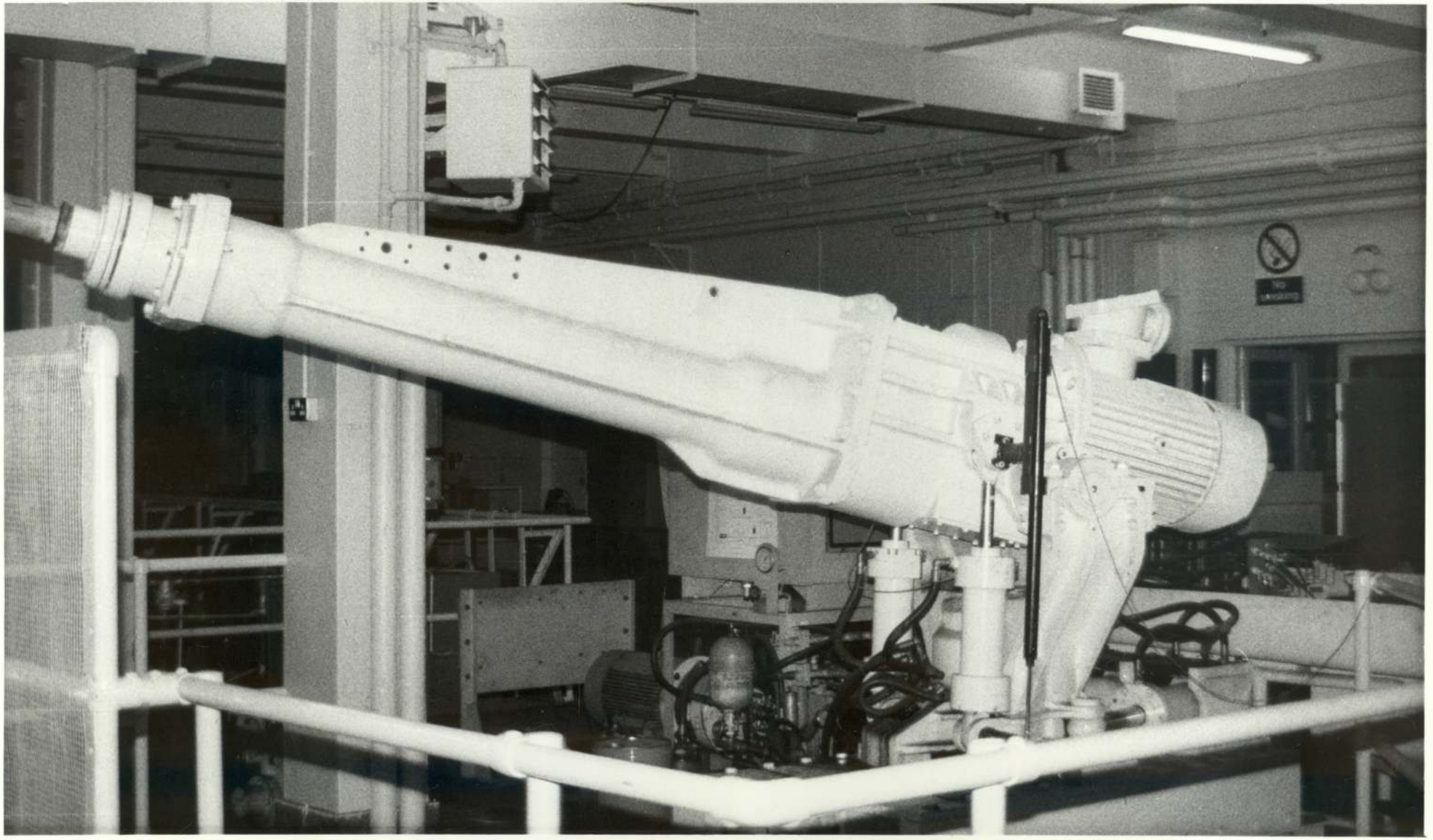


Fig 5.1 The Boom Ripper Assembly under Test

5.1. INTRODUCTION

It should be recalled that the aim of this project is to improve the control of the cutter position of a boom ripper, thus to deskill the tunnelling operation and to increase the overall performance of the roadheader. It would be helpful in deriving the control algorithm, if one has a rough idea of how the tunnel is driven using a boom ripper. In general, the cavity of the roadway is first cut layer by layer from the lowest one, the tunnel profile is then trimmed and followed by the setting up of the support arches. Hence the proposed system should be able to provide the cutting and trimming facilities in addition to moving the boom cutter from one position to another.

It can be seen from Figure 5.1 that the boom ripper has two degrees of freedom, namely slewing and lifting, provided by the paired hydraulic rams. When the machine is operating underground, it advances using its caterpillar tracks (see Figure 1.1) after one face has been cut, but this advancing of the roadheader is not considered in the current study. However, this can easily be included in the control system if the necessary sensors and mechanisms are available.

Because there are two degrees of freedom associated with the boom movement, it will be more convenient if the tunnel profile is defined in a coordinate system that resembles these degrees of freedom. Therefore, in this positional control system, the profile data are captured in pairs which describe the altitude and slew position of a point on the profile boundary, these data pairs will later be referred

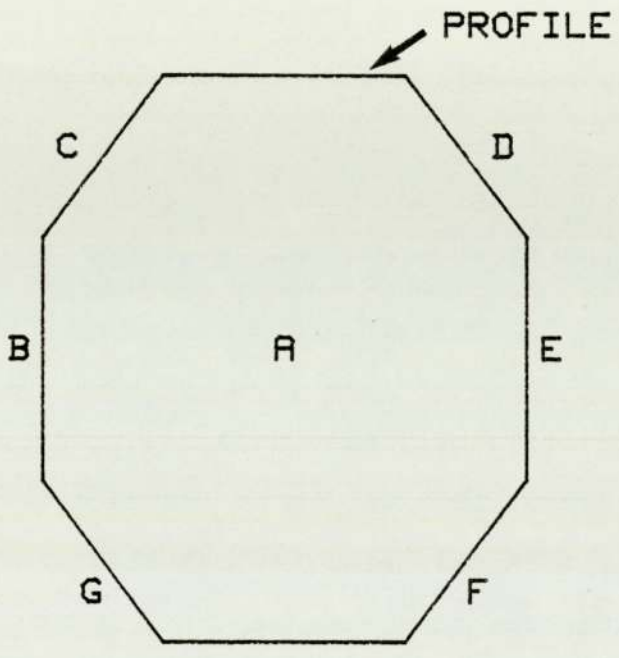


FIG 5.2 A Hypothetical Profile

to as altitude-slew pairs. Similarly, the desired position and feedback data will also be supplied in altitude-slew pairs.

It is undesirable for the boom to move outside the defined profile boundary, an algorithm has been developed to restrict the boom movement. Considering the hypothetical profile shown in Figure 5.2, the following actions will be required. When the current position of the cutter is at (A), it is free from the profile boundary, therefore its movement in all directions is allowed. When the cutter is currently at (B), it will only be allowed to move vertically and/or towards the right; if it is currently at (E), the boom cutter movement will be restricted to vertical and/or left only. In addition to the downward movement permitted, the cutter is allowed to move towards the right while it is currently at (C), but towards the left if its current position is at (D). Similarly, the cutter is allowed to move up and towards the left at (F); while it is permitted to move up and towards the right at (G). In other words, the cutter is only permitted to move to a position without crossing the profile boundary.

The proposed positional control system is composed of the mechanical, hydraulic, electrical and micro-electronic sections. The control program resides in the micro-electronic section of the system. In this chapter, the structure of the control algorithm is presented and the listing of the assembly language programs can be found in Appendix G.

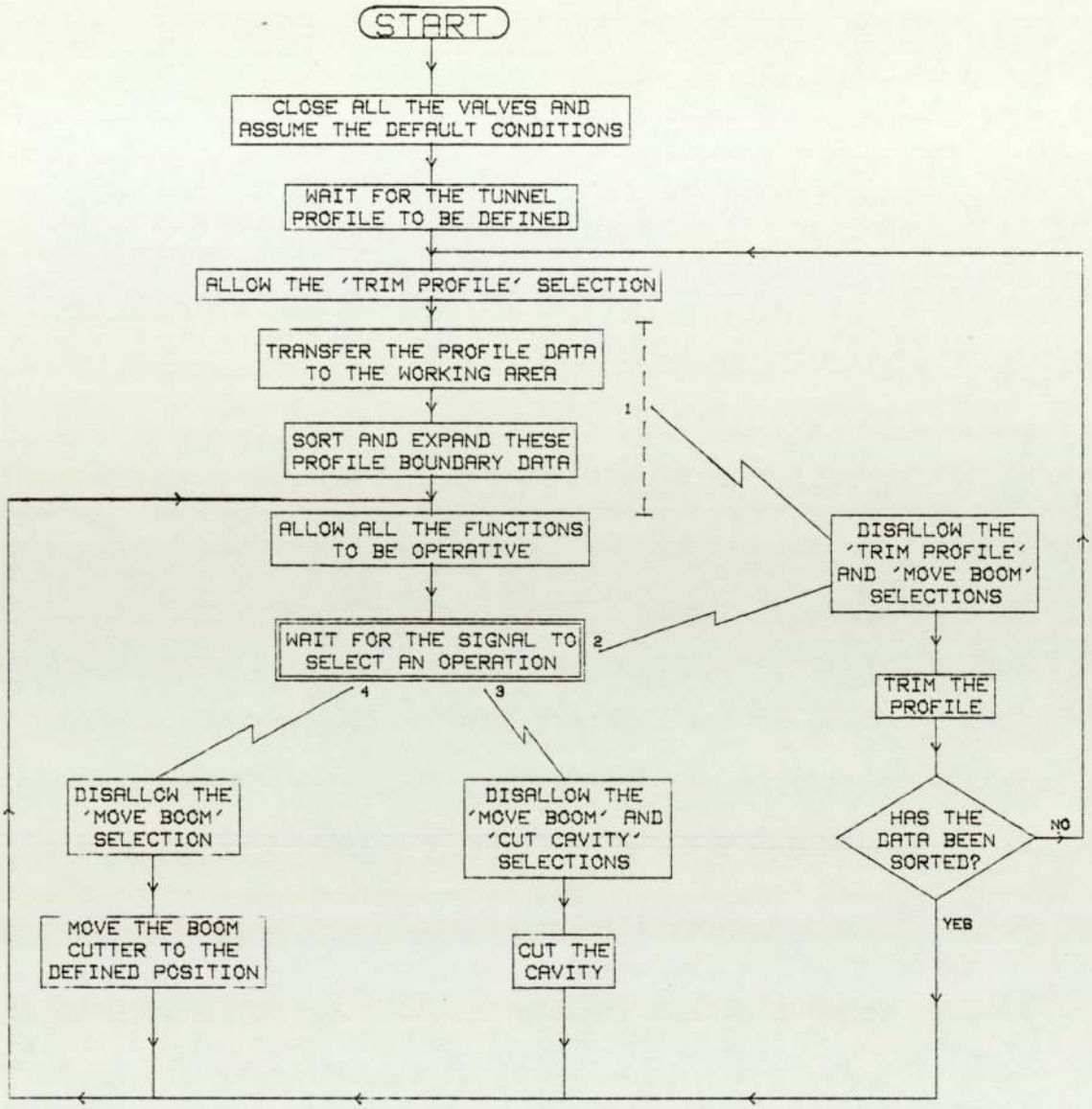


FIG 5.3 Flow Diagram of the General Structure of the Control Algorithm

5.2. GENERAL STRUCTURE OF THE CONTROL ALGORITHM

There is normally an initialisation procedure which will be performed on a fresh (sometimes known as a cold) start; and it is often to begin the operation with all the hydraulic valves closed. Therefore the control program will first ensure that all the valves are closed, assume the default conditions and then wait for the tunnel profile to be defined. There are three functions available in the system: moving the boom cutter from the current position to that defined by the operator (Move Boom), cutting the roadway cavity (Cut Cavity) and trimming the tunnel profile (Trim Profile) defined by the operator. However, for the reasons mentioned later, there will be four branching possibilities: the first and the second for trimming the profile, the third for cutting the cavity and the fourth for moving the boom cutter. A flow diagram for the general structure of this control algorithm is shown in Figure 5.3.

'Trim Profile' is the first operation that can be selected as soon as the tunnel profile has been defined. If this operation is not selected, the profile data will then be transferred to the data manipulating area (so that the original data can be preserved), where they will be sorted and expanded into a table in order to establish the profile boundary. This table consists of data sets, which contain the minimum and maximum slew bounds for every altitude from the lowest level to the highest one. Hence the table enables the cutting of the cavity layers and the checking to see if the boom cutter is moving within the specified profile boundary.

After the data have been sorted and expanded, all the function selections will be enabled. However, re-selection of a function will not be permitted once it has commenced its operation, since there is no need to select the same function twice. Hence the function selection is just an initiating procedure for that activity. After the completion of an operation, its function selection will again be enabled to permit a second trial if desired.

The 'Move Boom' control has two functions: selecting the 'Move Boom' operation and controlling the boom moving speed (there are two speeds available, fast and slow). It is undesirable to move the boom cutter to another position (away from its path) when it is cutting the cavity or is trimming the profile. Hence the 'Move Boom' selection control will be disabled if any one of these operations is selected unless that operation has been stopped or completed. During these two operations, the boom cutter movement is controlled by the program.

5.3. INITIALISATION

This is the first segment to be executed on starting the operation or resetting the system. All the input/output (I/O) ports are first declared so that the appropriate signals can be sent to or received from these ports. All the directional valves will then be closed. Initially, all the function selections are disabled, therefore no operation can be performed while the system is waiting for the tunnel profile to be defined by the operator. The system will also assume all the default conditions: stop

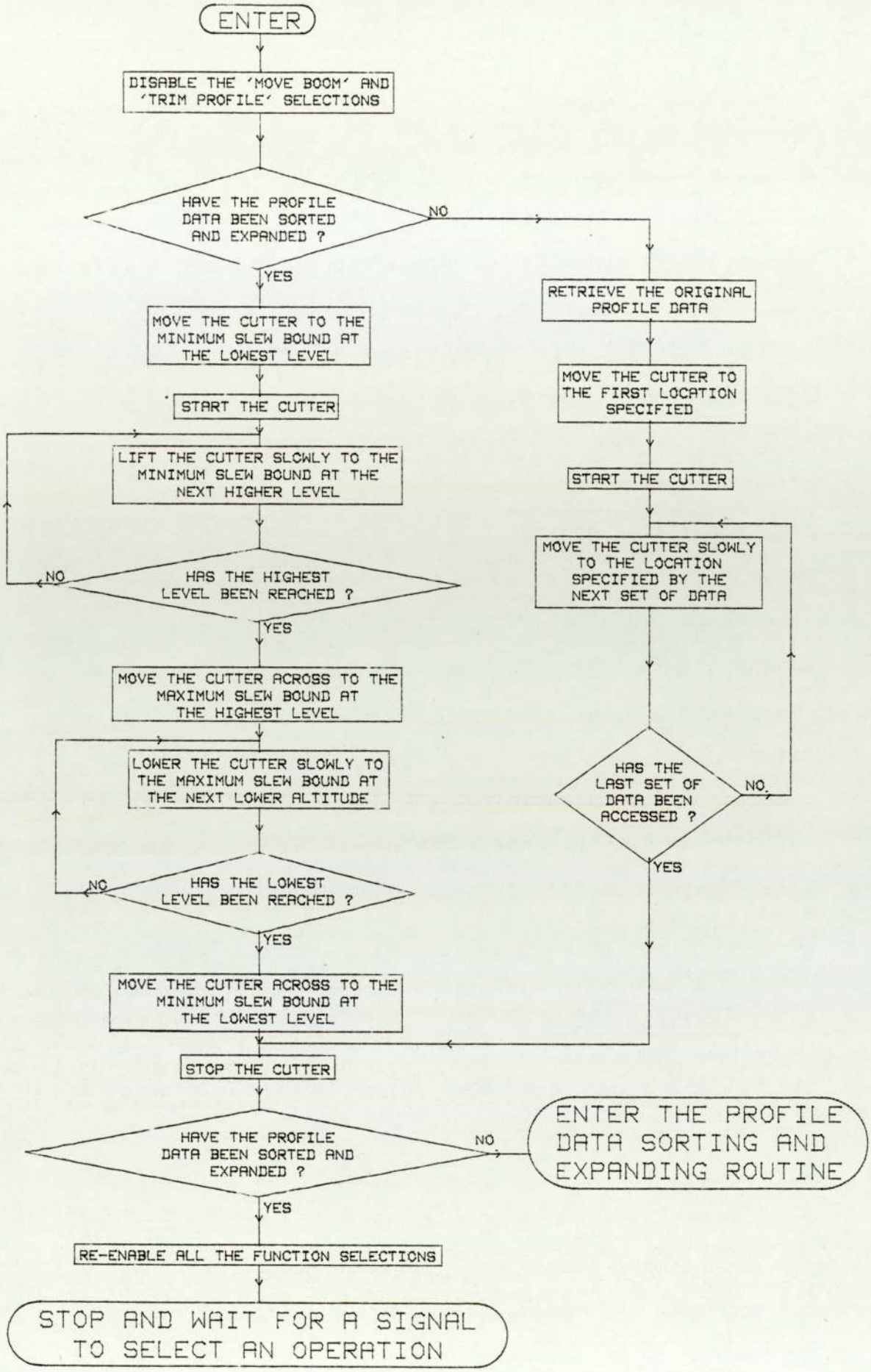


FIG 5.4 Flow Diagram for the 'Trim Profile' Routine

the boom cutter, assume slow speed for the 'Move Boom' mode and set up the appropriate entry locations for the various routines when the selection signals for different functions are recognised.

5.4. PROFILE TRIMMING

After the profile has been defined by the operator, the 'Trim Profile' operation may be selected. Although in normal practice, this operation will not be performed until the cavity has been excavated. However, since the profile has been defined, it can be trimmed if desired. It has been mentioned in Section 5.2 that, as soon as the profile has been defined, the program will call up the data sorting and expanding procedures which take a finite amount of time to complete. The 'Trim Profile' operation can therefore be selected before or after these sorting and expanding procedures have been completed (indicated as the branching at 1 or 2 in Figure 5.3). Different actions will therefore be taken to trim the profile depending on when the branching is originated. A flow diagram for this program segment is shown in Figure 5.4.

If the 'Trim Profile' operation is selected after the profile data have been sorted and expanded, then the boom will be moved to the minimum slew bound at the lowest level and the cutter will be started. The profile trimming process then commences with the cutter being moved slowly along the defined profile. On the other hand, if it is selected before the data sorting and expanding procedures have been completed, the original profile data

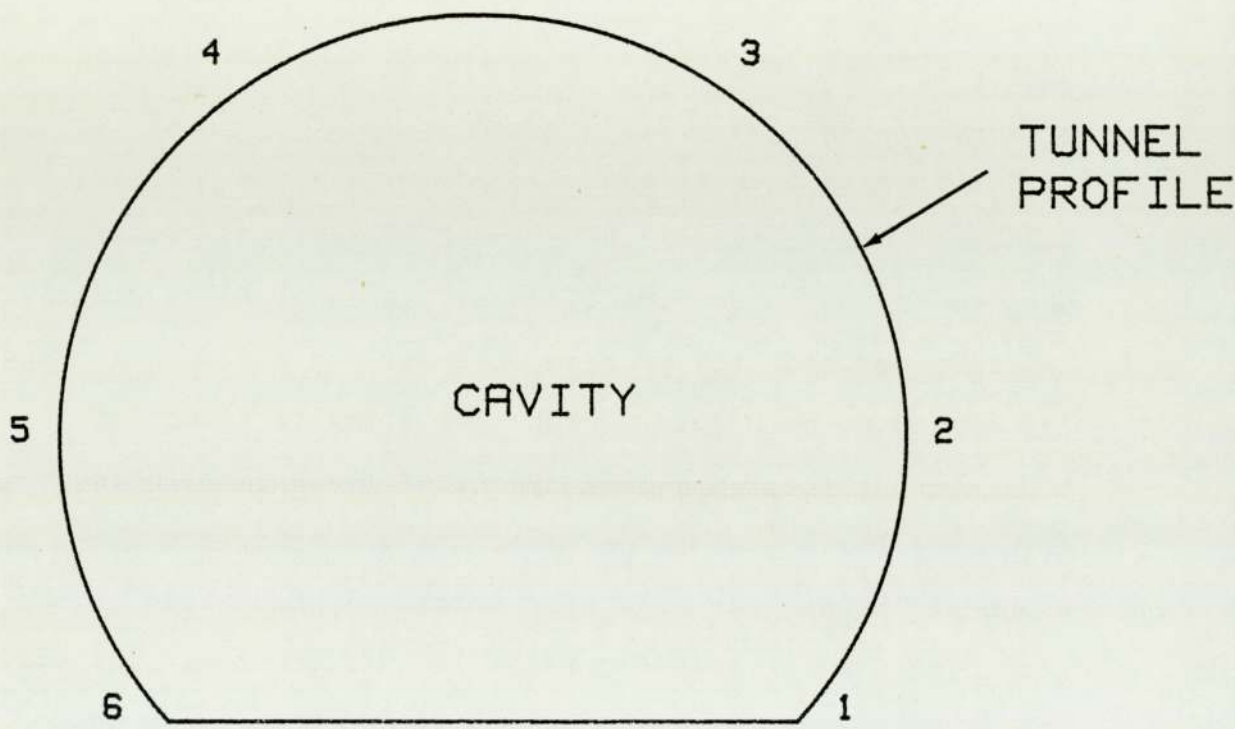


FIG 5.5 A Sample Tunnel Profile

are retrieved from the data capturing system, the boom cutter will then be moved to the first position defined by the operator in specifying the tunnel profile. The cutter will also be started and moved slowly along the profile. The boom cutter movement will follow the way in which the profile is defined by the operator.

If the sorting and expanding procedures have not been completed, the profile data will be sorted and expanded after the completion of this profile trimming process. Otherwise, the system will then wait for the signal to select an operation.

It has been mentioned that there are two ways in moving the boom cutter along the profile: one for the original data and the other for the sorted and expanded data. Figure 5.5 shows a sample tunnel profile which has been defined as the path: 1-2-3-4-5-6-1.

The algorithm for moving the boom cutter along the profile using the unsorted data is relatively simple. The profile data are first transferred to the working area in the form as they are captured. Then the cutter will be moved to the location specified by the first data pair (the profile data are captured as altitude-slew pairs). It will be started and moved slowly to the locations specified by the data pairs in the order as they are captured. After the boom cutter has been moved to the location defined by the last pair of the profile data, it will then be stopped. In other words, the boom cutter will follow the path defined by the operator (the path: 1-2-3-4-5-6-1 in Fig. 5.5).

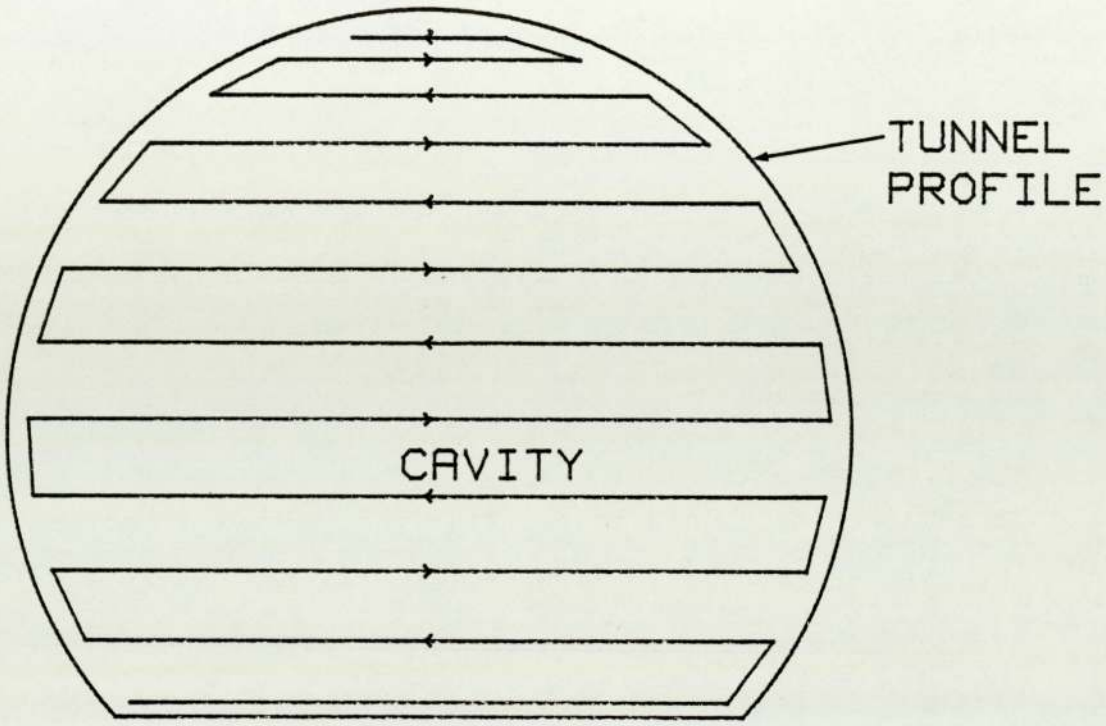


FIG 5.6 A Sample Cavity Cutting

The algorithm for moving the cutter along the profile using the sorted and expanded data is different. The profile data have been sorted and expanded into a table that contains the minimum and maximum slew bounds for every altitude from the lowest level to the highest one. Therefore the boom cutter can be moved to the location where is the minimum slew bound at the lowest altitude. After the cutter has been positioned at this location, it will then be started and raised slowly to the minimum slew bound at the next higher level, until the cutter reaches the highest altitude. Then it will be moved across from the minimum slew bound to the maximum one at this altitude. Afterwards it will be lowered to the maximum slew bound at the next lower level, until the lowest one has been accessed. The cutter will finally be moved across this lowest boundary from the maximum slew location to the minimum one, before it is stopped. Effectively, the cutter has been moved along the defined profile (along the path: 6-5-4-3-2-1-6 as shown in Figure 5.5).

5.5. CAVITY CUTTING

The 'Cut Cavity' operation cannot be selected until the tunnel profile has been defined and the data have been sorted and expanded. After this operation has been selected, the boom cutter movement is controlled by the program. A sample path of this movement is shown in Figure 5.6. Similar to the 'Trim Profile' operation, the 'Move Boom' and the 'Cut Cavity' selections are disabled once this routine has been entered. A flow diagram for

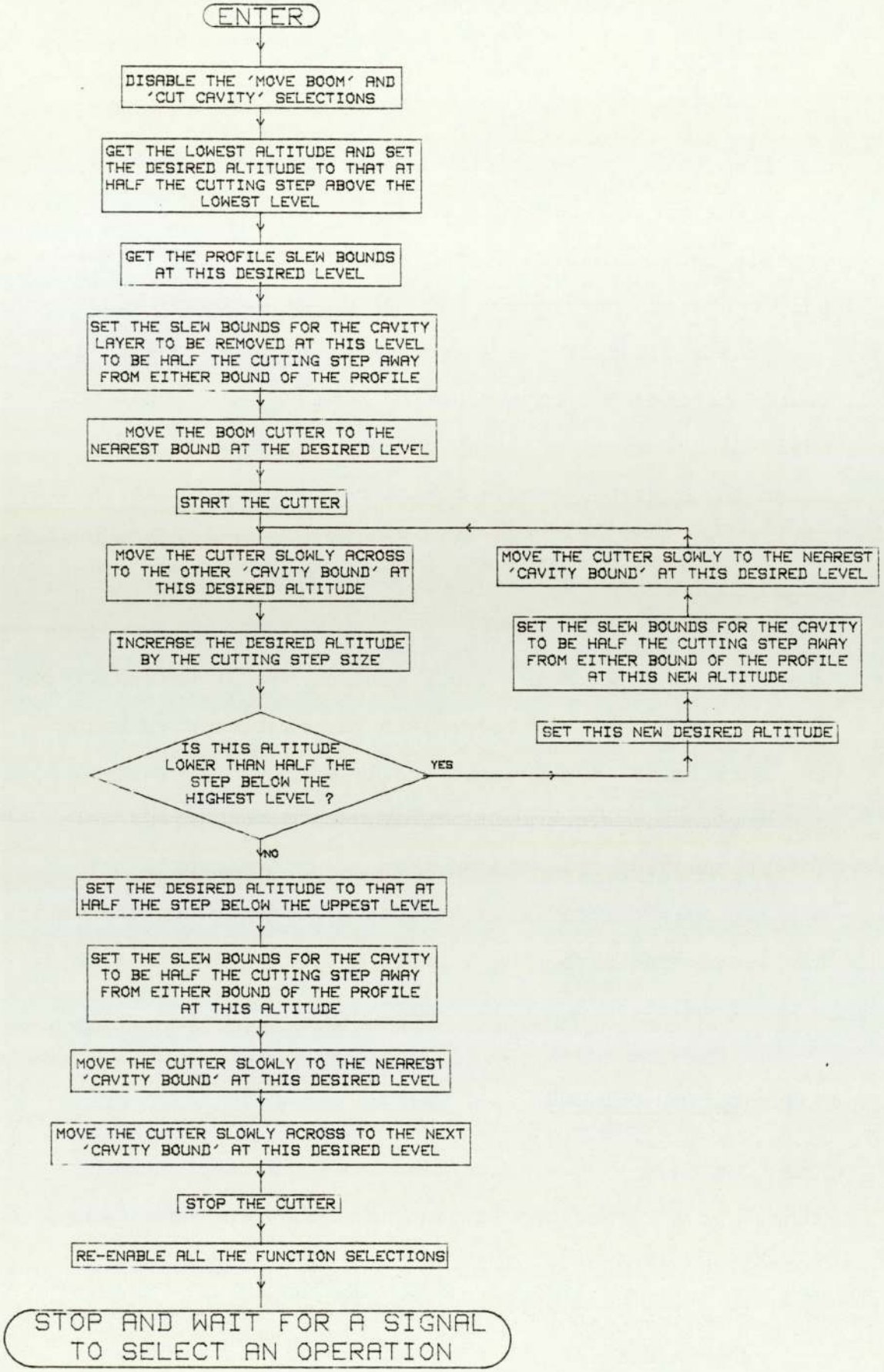


FIG 5.7 Flow Diagram for the 'Cut Cavity' Routine

this routine is shown in Figure 5.7.

After this operation has been selected, the highest and the lowest levels of the cavity layers to be excavated by this operation are first evaluated. These cavity levels* are chosen to be half the desired vertical depth of cut away from but contained in the actual profile boundary. The slew bounds at this lowest cavity altitude are similarly reduced. The boom cutter will be located to the nearest cavity slew bound at the lowest cavity altitude. After the cutter has been moved to this position, it is started and the 'Cut Cavity' process commences. The cutter is now moved slowly across to the other reduced slew bound so that this cavity layer will be removed. After this layer has been cut, the slew bounds at the level, which is one desired vertical depth of cut higher, are similarly reduced (half the depth of cut smaller on both sides than the actual profile slew bounds). The cutter will be advanced to the reduced slew bound on the same side at that higher level. The movement of the cutter is restricted to within the prescribed profile boundary. Then this cavity layer will be cut. The process is repeated until the highest cavity level has been removed.

After the completion of this operation, a cavity, which is smaller than the actual profile by half the vertical

---

* In this section, when a term is preceded by the word cavity, it refers to the reduced value. Therefore cavity slew bound is the same as reduced slew bound. When a term follows the word profile, it means the actual value. Hence the profile boundary is another way of referring to the actual boundary.

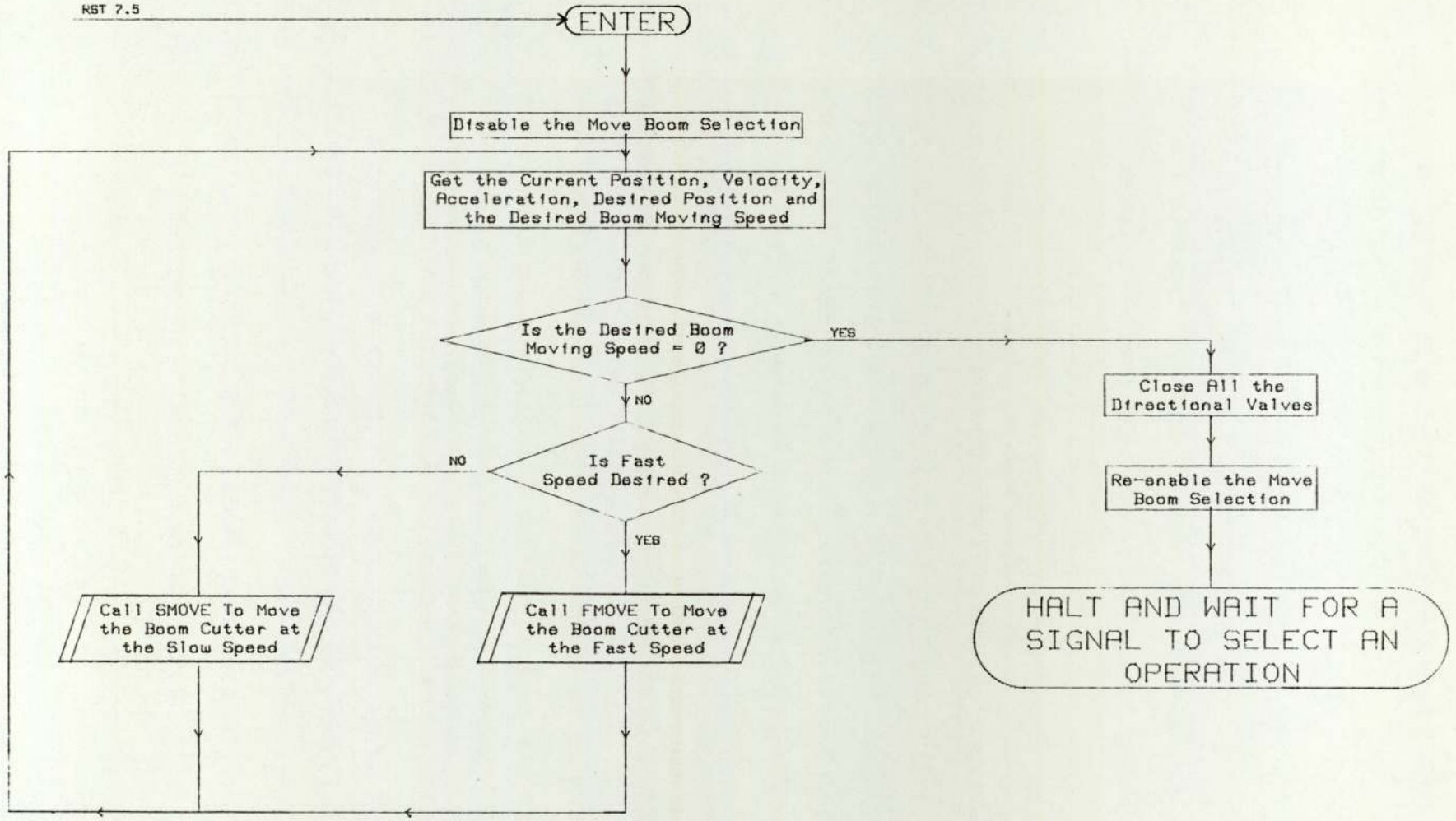


FIG 5.8 A Flow Diagram for the 'Move Boom' Routine

cutting step, will be formed. All the three function selections will then be enabled. The final profile boundary can then be removed by the 'Trim Profile' operation.

5.6. BOOM MOVING

After the tunnel profile has been defined and the completion of the profile data sorting and expanding procedures, the 'Move Boom' operation can be selected while the cutter is neither cutting the cavity nor trimming the tunnel profile. It is desirable to have the 'Move Boom' and the 'Boom Moving Speed' selectors combined into a single control. When this control is operated, it first sends a selection signal to put the system into the 'Move Boom' mode and continuously informs the system the chosen boom moving speed. A flow diagram of this 'Move Boom' control algorithm is given in Fig. 5.8.

After the 'Move Boom' selection signal has been recognised, this function will only act as the 'boom moving speed' selector. It selects one of these three possibilities: fast, slow and no-move. On entering this routine, the main control system will collect the information on the current and desired positions as well as the chosen speed for the boom cutter (from the 'boom moving speed' selector). The appropriate subroutine will be called for moving the boom cutter at the desired speed. The subroutine, SMOVE, for moving the boom cutter slowly, will be described in a later section. The subroutine, FMOVE, for the fast moving of the cutter will be discussed in Chapter 7. This process will

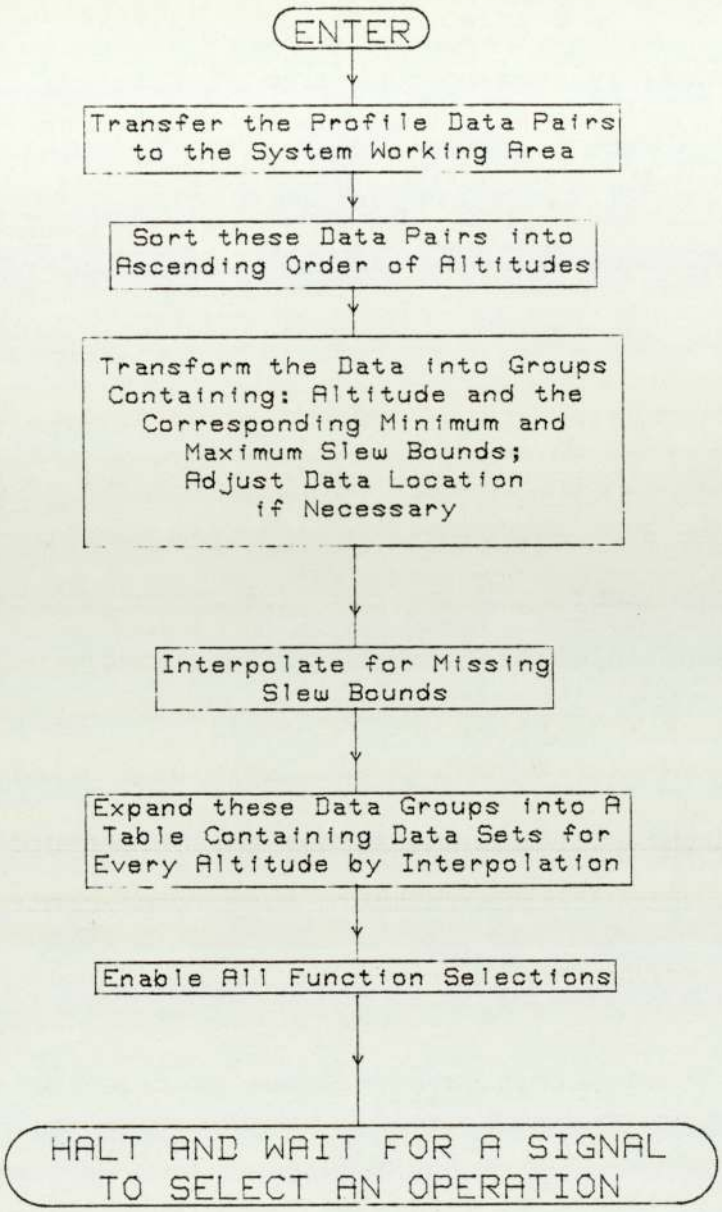


FIG 5.9 A Flow Diagram for the Data Sorting and Expanding Procedures

be repeated until the 'boom moving speed' selector selects a 'no-move' mode. When this 'no-move' selection is recognised, all the directional valves will be closed and all the function selections will also be re-enabled. This allows the system to wait for a signal to select an operation.

5.7. SORTING AND EXPANDING THE PROFILE DATA

These procedures are performed immediately after the tunnel profile has been defined. When the data are being sorted and expanded, only the 'Trim Profile' operation can be selected. Fig. 5.9 shows the flow diagram for these routines. The captured data are first transferred to the working area so that the original data can be preserved. These data will then be sorted in the ascending order of the profile altitudes. The sorted data will be transformed into groups containing the altitudes with their two slew bounds for the defined profile. Therefore, the regrouped data will be in sets of three and will appear as:

lowest altitude, min. slew bound, max. slew bound, ...,
highest altitude, min. slew bound, max. slew bound.

The original data are captured in altitude-slew pairs when the operator defines the tunnel profile. This data capturing is done using the system developed by J H Knight in a parallel project at the University of Aston. He built into this data capturing system an algorithm to avoid capturing incorrectly specified profile data. These data can be either the duplicating data pairs (because the data sampling rate is faster than the speed that the operator

can define the profile), or the data 'mistakenly' generated by the analogue-to-digital converter due to its accuracy. A tolerance has been introduced in this algorithm, it is therefore possible to miss out one of the slew bounds during the data capturing process.

In the transformed data groups, there are three data per group instead of four to define the two slew bounds at any altitude. If only one profile slew bound has been captured, the system will generate the 'missing' slew bound by assuming the same value as the captured data. An algorithm has been derived to prevent the system from loss of data in the data re-grouping process: all the remaining data are copied to their next higher locations if there will be a 'crash' resulted from any data generating process.

After the data have been transformed, any data group with the same left and right slew bounds (i.e. only one slew bound has been captured for that altitude), except the groups for the lowest and the highest altitudes, can be recognised as deficient in one data. The missing information will be interpolated from the adjacent data sets (data groups with the next lower and higher altitudes), to give the appropriate slew bound.

The system can only detect single data point, but cannot determine whether the maximum slew bound or the minimum value is missing without additional checks, therefore interpolation will be performed for both bounds whenever the data group contains only one slew bound. Fig. 5.10

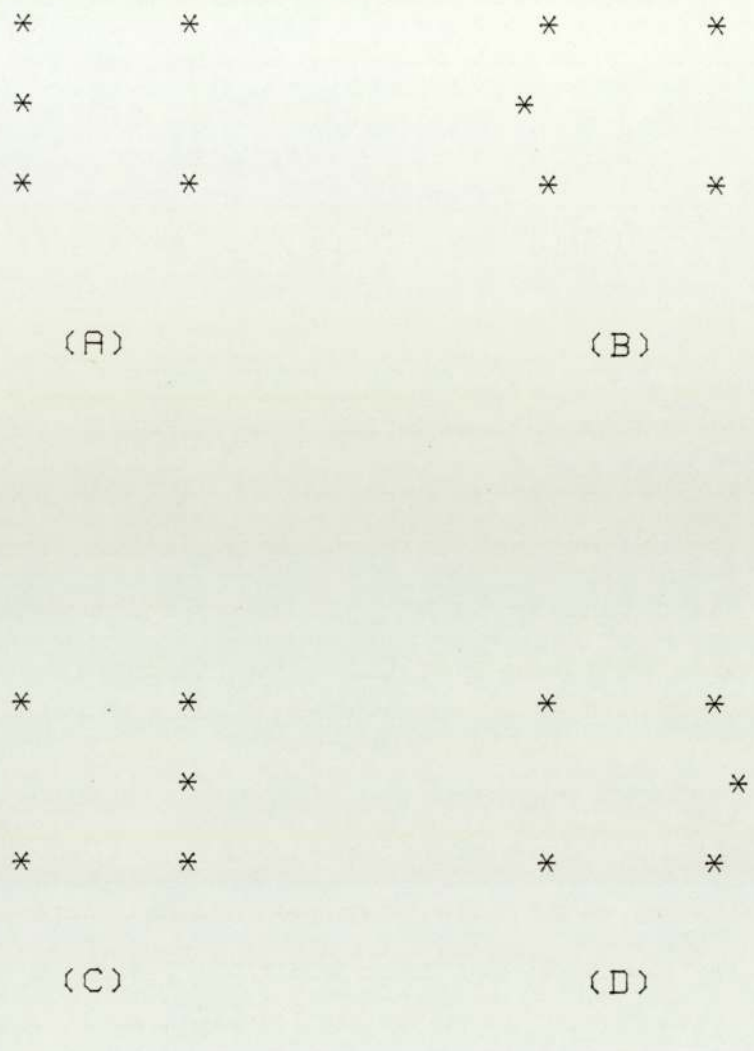


FIG 5.10 Sample Discrete Profile Data Captured

gives the four possibilities of single data point being captured. In (A) and (B), the maximum slew bound can be found by interpolating the maximum slew bounds of the two adjacent data groups. In (C) and (D), the minimum slew value can be found by interpolating the minimum slew bounds of the two adjacent data sets. However, in (B), the minimum value obtained by interpolation will be greater than the captured data; while in (D) the interpolated maximum slew bound will be less than the captured value. Therefore, a simple algorithm for interpolating the data can be derived: the minimum slew bound can be found by taking the smaller of the captured minimum and the interpolated minimum, while the maximum slew bound can be found by taking the larger of the captured maximum and the interpolated maximum.

The table will finally be constructed to contain data group at every altitude. However, because of the way that the profile data are captured, there may be missing data sets at some altitudes. These missing data sets, if any, will be generated by interpoloation for their altitudes and the slew bounds. After this table has been formed, all the function selections will be allowed, and the system will wait for a signal to select one of the three operations available.

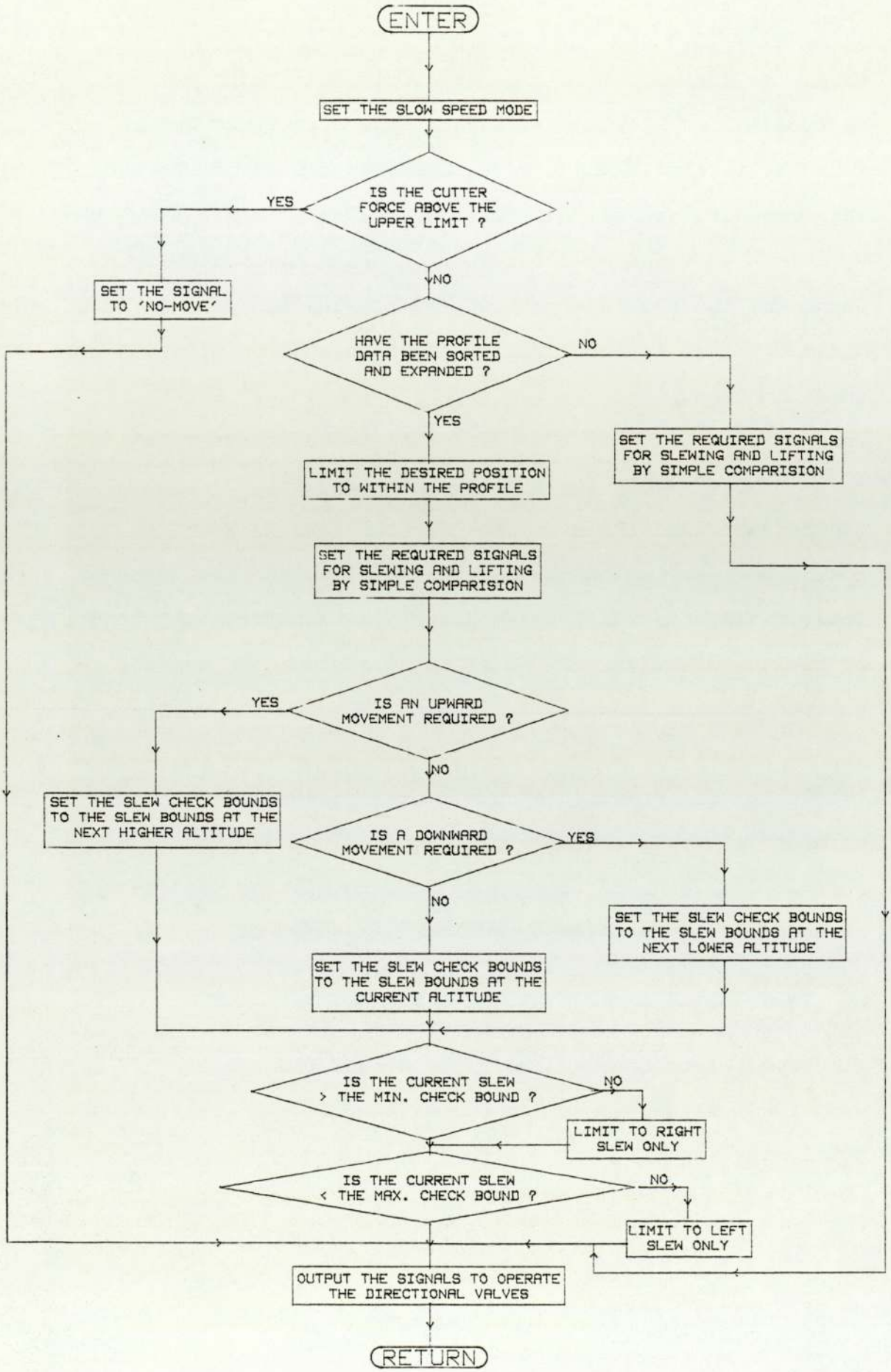


FIG 5.11 A Flow Diagram for the 'SMOVE' Subroutine

5.8. ROUTINE FOR MOVING THE BOOM CUTTER SLOWLY - 'SMOVE'

When the boom cutter is being moved slowly, the signals for the directional valves are determined by purely comparing the desired and the current positions of the cutter. A flow diagram for this routine is shown in Fig. 5.11.

This routine is employed whenever the boom cutter is required to move slowly. The checking for the cutter movement to stay within the profile can only be possible after the data table has been established. Hence this checking will not be exercised if the profile data have not been sorted and expanded. There is therefore a possibility for the cutter to move outside the prescribed profile boundary. However, this 'no-check' movement is only performed when the 'Trim Profile' operation is selected before the completion of the data sorting and expanding procedures; during this operation, the cutter itself is moving along the profile and the overshoot, if any, is not significant after the "smoothing" action of the cutter.

If the desired cutter movement is being checked to ensure that the boom cutter will move within the profile boundary, the following algorithm will be used. The desired altitude is first checked to ensure that it is within the profile. The desired slew at the required altitude will then be checked to ensure that it is not outside the defined profile boundary. The desired altitude is compared with the current altitude to determine the required movement: upward,

downward or horizontal. The comparison between the desired and the current slew values will give the required arcing movement: left, right and no slewing. After these signals have been determined, the effect of the desired movement will then be checked against the profile data table using the algorithm described in Section 5.1. This additional checking will prevent the boom cutter being moved outside the defined profile boundary.

In addition to the positional checks, the cutter force (indicated by the cutter motor current) is also examined: if the force is above the permissible upper limit, then the boom cutter movement will be completely stopped. The finalised signals will then be sent to operate the directional valves for the required boom cutter movement.

5.9. MODE CHANGING

Since the 'Move Boom' selection will be disabled when either the 'Trim Profile' or the 'Cut Cavity' operation is selected, a facility, which stops the current operation and enables the operator to select the 'Move Boom' operation, may be desirable. (The operator may want to move the boom cutter away from a hard stratum which causes the cutter to stop). On selecting this 'Change Mode' operation, all the directional valves will be closed. The data sorting and expanding procedures will be executed if the profile data table has not yet been established. Otherwise, the system will wait for the operator to select the desired operation.

CHAPTER 6

CONTROL CIRCUIT DESIGN

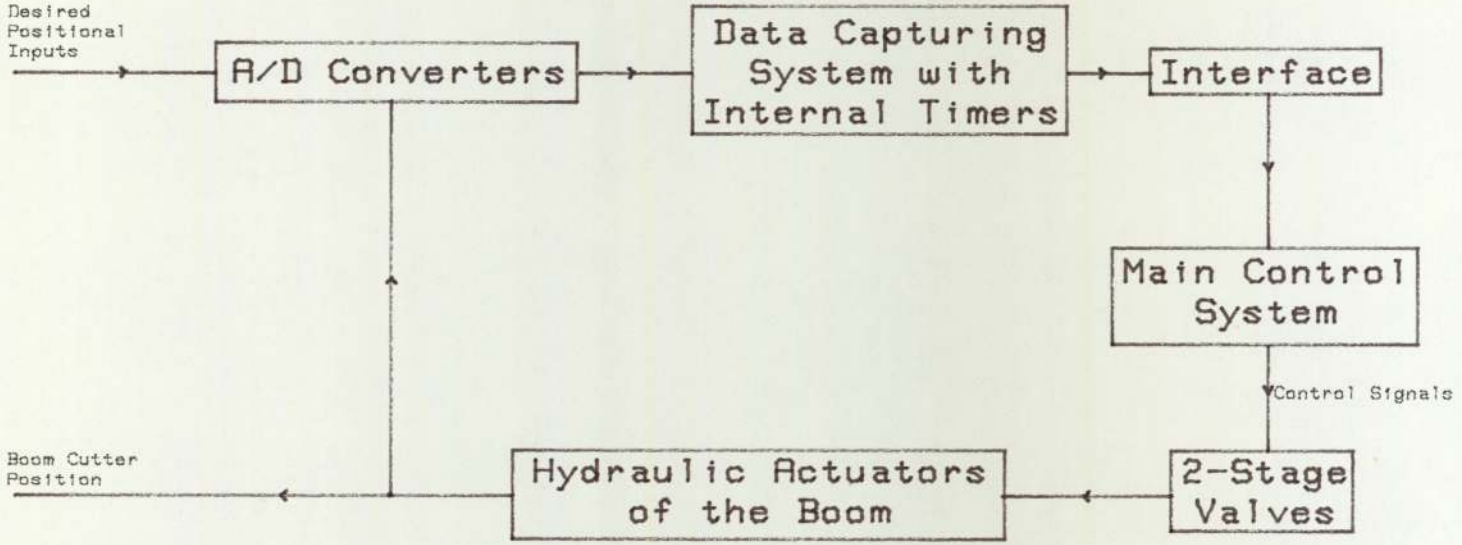


FIG 6.1 Block Structure of the Proposed Dual Microprocessor Control System for the Mining Boom Positions

6.1. INTRODUCTION

It has been decided to control the position of a mining boom cutter with a microprocessor based system. There are many microprocessors available on the market. However, the one commonly used in the Department of Mechanical Engineering at the University of Aston is Intel-8085, hence this microprocessor is chosen for the proposed control system.

A microprocessor by itself can do nothing, it needs other supporting elements, such as address decoders, random access memories (RAMS), read only memories (ROMs), buffers, logic gates, input/output (I/O) ports and other accessories to perform the appropriate operations.

The proposed system is a dual microprocessor system with a main system for the control algorithms and a second one for capturing the system performance and input data ready for use by the main system. Since the two microprocessors are running in parallel, this arrangement will enable a high speed operation. The block diagrams of this control system are shown in Figures 6.1 and 6.2. The 'hand-shaking' facility is provided by an interface to enable the main control system to get data from the data capturing system. Figure 6.3 shows the procedures taken by the two systems in communicating with each other.

In this chapter, a brief account is given on the design of the electronic circuit for the main microprocessor system and the interface to the data capturing system. The data

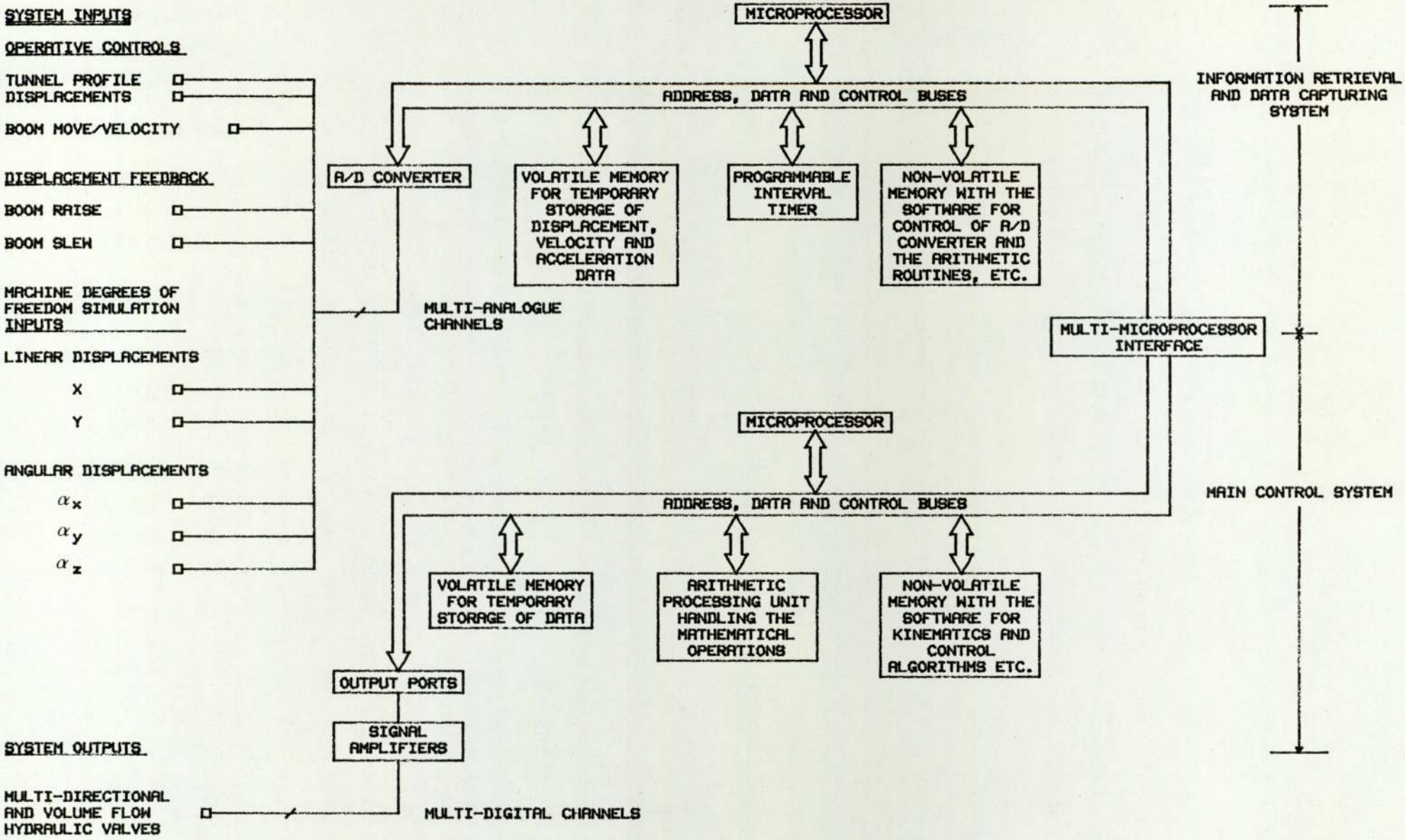


FIG 6.2 Proposed Dual-Microprocessor Control System for Boom Cutter Position

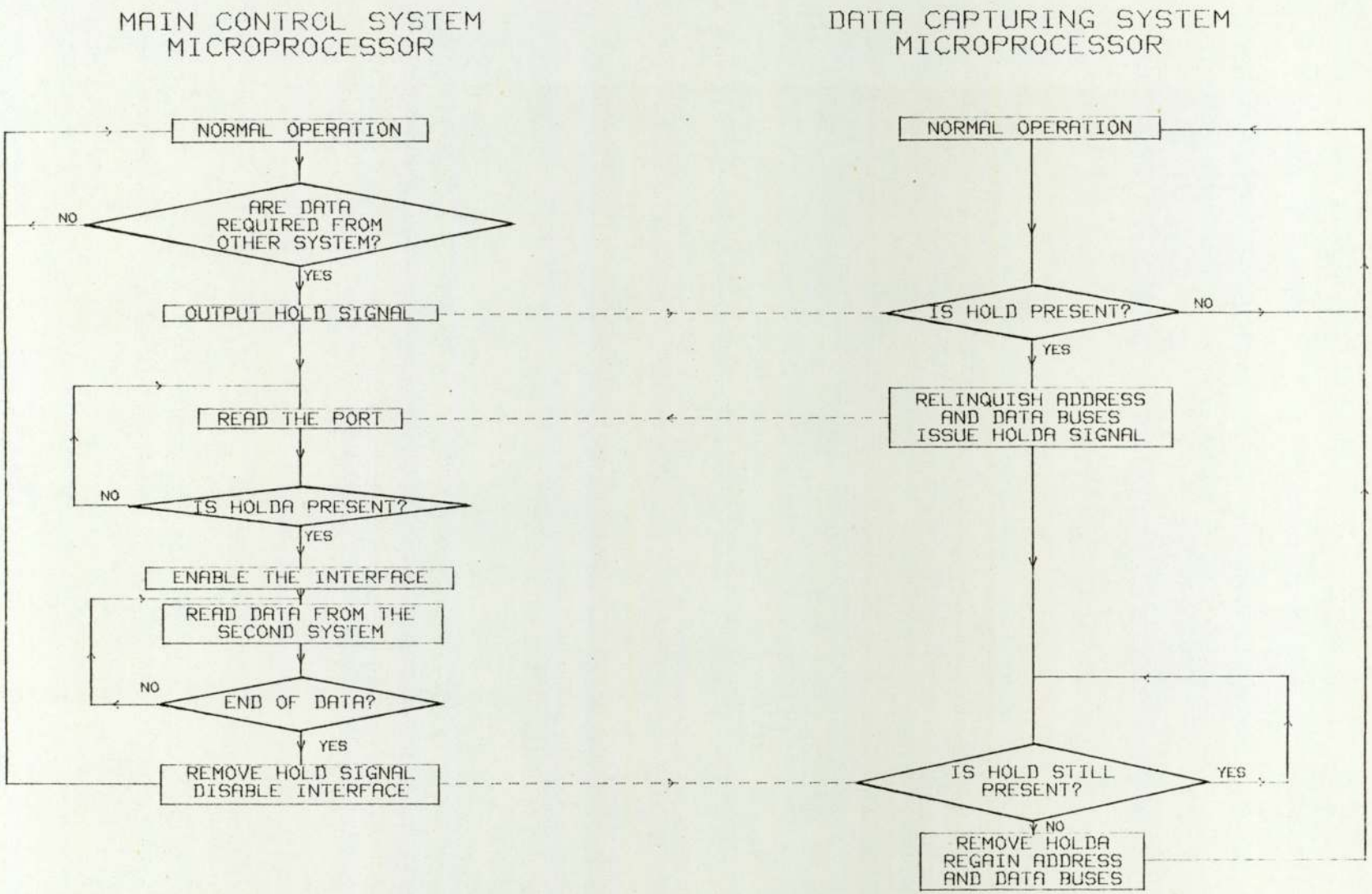


FIG 6.3 Flow Diagram for the Actions Taken by the Two Microprocessors in Communicating With Each Other

capturing microprocessor system and the interface to the pilot valves of the hydraulic actuators are developed by J H Knight in a parallel project at Aston University.

6.2. MAIN MICROPROCESSOR AND MONITOR BOARD

The first board in the main microprocessor control system consists of the microprocessor, an address decoder, the first monitor ROM, two 8-bit I/O ports, buffers, the crystal and logic gates. A schematic diagram of this board is given in Figure 6.4.

The Intel-8085 is an 8-bit parallel central processing unit (CPU). It is designed with N-channel depletion loads, and requires a single +5v supply. Multiplexed lower order address and data bus reduces the number of lines required for transferring the data. Detailed functional description of this CPU can be found in the Intel publication [42].

6.2.1. *Clock Generation*

A CPU is normally dynamic and requires a clock signal to operate. The Intel-8085 CPU has an internal clock generator which derives the timing inputs for the microprocessor from the 6.144 MHz crystal. It generates an internal clock frequency of 3.072 MHz with a clock cycle period of about 326 nSec. It also makes this clock (3.072 MHz) available on the clock output pin of the CPU so that this signal can be used as the system clock.

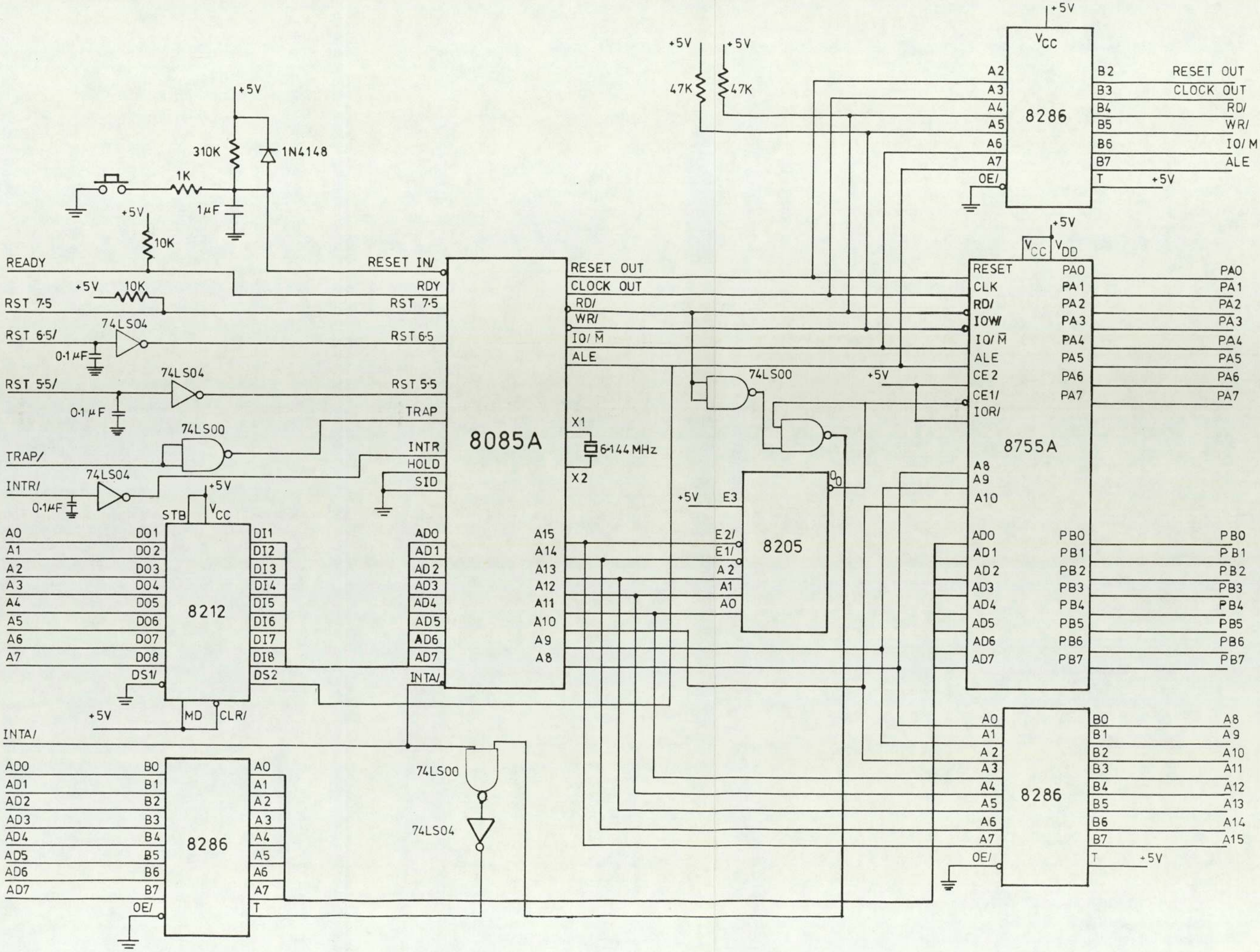


FIG. 6.4 SCHEMATIC DIAGRAM FOR THE MAIN MICROPROCESSOR BOARD

6.2.2. *$\overline{\text{RESET IN}}$ and Power On*

This system can be reset by closing the reset switch to low activate the $\overline{\text{RESET IN}}$ input (pin 36) of the CPU. This CPU will also be automatically reset when the power is switched on. The Intel-8085 CPU employs a special internal circuit, a substrate bias generator, to increase its speed and needs time for this circuit to stabilise, therefore it is not guaranteed to work until 10 mSec after the supply voltage (V_{CC}) has reached +4.75v. A RC (resistor-capacitor) circuit is introduced to keep the $\overline{\text{RESET IN}}$ signal low during this period [44] so that the CPU can be reset at power on.

On recognising the $\overline{\text{RESET IN}}$ low signal, the CPU will generate the high RESET OUT signal which can be used to reset the whole system. Since various signal lines and buses will be floating during the reset process, the low active control signal outputs (e.g. $\overline{\text{RD}}$, $\overline{\text{WR}}$) are connected to pull up resistors to ensure the inactive state during this resetting period.

6.2.3. *Address Decoding*

The address is decoded using a 3-to-8 line decoder, Intel-8205 or 74LS138 (a low power Schottky transistor-transistor-logic integrated circuit). The decoding is performed in 2K (2048D bytes) blocks by examining the address lines A_{11} to A_{15}. The decoder will drive the selected output low, therefore this output can be used for any device that requires a low active chip select input.

6.2.4. *First Monitor and I/O Ports*

The first monitor program resides in the erasable programmable read only memory (EPROM), Intel-8755A, which has a memory of 2048D x 8 bits. Following the reset process, the CPU will start executing instructions from the location with address equal to 0, therefore this monitor EPROM is configured to have its addresses from 0 to 7FF H (i.e. the first 2K block). The Intel-8755A is compatible with the Intel-8085 CPU and has two general purpose 8-bit I/O ports, it facilitates the construction of the control system.

The boom is moved by operating two pairs of hydraulic actuators: one for lifting the trunk and the other for arcing (slewing) it. Each pair of these actuators use four valves to control the direction of the ram movement, hence require four bits of the output port. This means that a total of eight bits should be made available for output. These are provided by assigning one of the two I/O ports of the Intel-8755A to be an 8-bit output port. Three bits of the other port are used for interfacing to the data capturing system. This arrangement leaves 5 bits of that I/O port free for other purposes if required. In fact, these five bits are used as outputs for resetting the APU, setting the boom moving speed (one bit for slewing and one bit for lifting), starting and stopping the boom cutter revolving.

6.2.5. *Low Order Address Latching*

Since the CPU uses a multiplexed low order address and data bus, the low order address has to be latched, if the low

order address is required, throughout the machine cycle of the CPU. This address latching is done by employing an 8-bit latch, 8212. The low order address is put onto the bus during the first part of a machine cycle and is latched to the output of the 8212 by the address latch enable (ALE) signal when this ALE signal goes from high to low.

6.2.6. *Buffers*

Buffers are installed to provide the expansion facilities. The latched low order address has already been buffered through the latch, 8212. The control signals, the high order address bus and the multiplexed low order address/ data bus are buffered by the octal bus transceivers, Intel-8286.

The buffer for the control signals is configured to transmit signals from side A to side B of the transceiver. These buffered signals are: IO/$\overline{M}$, $\overline{WR}$, $\overline{RD}$, ALE, CLK OUT and RESET OUT. Similar arrangement is made for the high order address bus. For these transceivers, the $\overline{OE}$ inputs are tied low to enable the data transmission through the buffer, while the T inputs are connected to +5v (high) to configure the transceivers B_0 - B_7 as outputs with its A_0 - A_7 as inputs.

The buffer for the multiplexed low order address and data bus is connected with $\overline{OE}$ low to enable data transfer through the buffer. The buffer's transmit control signal is configured so that this signal is normally high for

PIN	MNEMONIC	DESCRIPTION
1a	GND	SIGNAL GROUND
2a	+5V	+5 VOLTS
3a	A0	DEMULTIPLEXED LOW ORDER ADDRESS BIT 0
4a	A1	DEMULTIPLEXED LOW ORDER ADDRESS BIT 1
5a	A2	DEMULTIPLEXED LOW ORDER ADDRESS BIT 2
6a	A3	DEMULTIPLEXED LOW ORDER ADDRESS BIT 3
7a	A4	DEMULTIPLEXED LOW ORDER ADDRESS BIT 4
8a	A5	DEMULTIPLEXED LOW ORDER ADDRESS BIT 5
9a	A6	DEMULTIPLEXED LOW ORDER ADDRESS BIT 6
10a	A7	DEMULTIPLEXED LOW ORDER ADDRESS BIT 7
11a	PA0	CONTROL SIGNALS FOR THE DIRECTIONAL VALVES FOR THE BOOM ELEVATION
12a	PA1	
13a	PA2	
14a	PA3	
15a	PA4	CONTROL SIGNALS FOR THE DIRECTIONAL VALVES FOR THE BOOM SLEW
16a	PA5	
17a	PA6	
18a	PA7	
19a	READY	READY SIGNAL
20a	RESET OUT	SYSTEM RESET SIGNAL FROM THE MICROPROCESSOR
21a	CLOCK OUT	SYSTEM CLOCK OUTPUT [3.072 MHz]
22a	HOLD B	HOLD THE DATA CAPTURING SYSTEM [HIGH ACTIVE]
23a	PB7	HOLD ACKNOWLEDGE FROM THE DATA CAPTURING SYSTEM
24a	PB6	ENABLE THE INTERFACE BETWEEN THE TWO SYSTEMS [LOW ACTIVE]
25a	PB5	INVERTED HOLD B
26a	PB4	APU RESET SIGNAL
27a	PB3	STOP THE CUTTER [LOW ACTIVE]
28a	PB2	START THE CUTTER [LOW ACTIVE]
29a	PB1	FAST/SLOW SPEED CONTROL SIGNALS [0 FOR FAST]
30a	PB0	
31a	+5V	+5 VOLTS
32a	GND	SIGNAL GROUND
1c	GND	SIGNAL GROUND
2c	+5V	+5 VOLTS
3c	AD0	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 0
4c	AD1	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 1
5c	AD2	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 2
6c	AD3	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 3
7c	AD4	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 4
8c	AD5	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 5
9c	AD6	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 6
10c	AD7	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 7
11c	A8	HIGH ORDER ADDRESS BIT 8
12c	A9	HIGH ORDER ADDRESS BIT 9
13c	A10	HIGH ORDER ADDRESS BIT 10
14c	A11	HIGH ORDER ADDRESS BIT 11
15c	A12	HIGH ORDER ADDRESS BIT 12
16c	A13	HIGH ORDER ADDRESS BIT 13
17c	A14	HIGH ORDER ADDRESS BIT 14
18c	A15	HIGH ORDER ADDRESS BIT 15
19c	IO/$\overline{M}$	I/O PORT OR MEMORY CONTROL SIGNAL
20c	RD/	READ CONTROL [LOW ACTIVE]
21c	ALE	ADDRESS LATCH ENABLE
22c	TRAP/	INVERTED TRAP INTERRUPT
23c	RST 7.5	RESTART 7.5 INTERRUPT (FOR MOVING THE BOOM)
24c	RST 6.5/	INVERTED RESTART 6.5 INTERRUPT (FOR CUTTING THE CAVITY)
25c	RST 5.5/	INVERTED RESTART 5.5 INTERRUPT (FOR TRIMMING THE PROFILE)
26c	WR/	WRITE CONTROL [LOW ACTIVE]
27c	INTR/	INVERTED INTERRUPT FOR STOPPING THE BOOM [LOW ACTIVE]
28c	INTA/	INTERRUPT ACKNOWLEDGE [LOW ACTIVE]
29c	RESET IN/	SYSTEM RESET INPUT [LOW ACTIVE]
30c	+12V	+12 VOLTS
31c	+5V	+5 VOLTS
32c	GND	SIGNAL GROUND

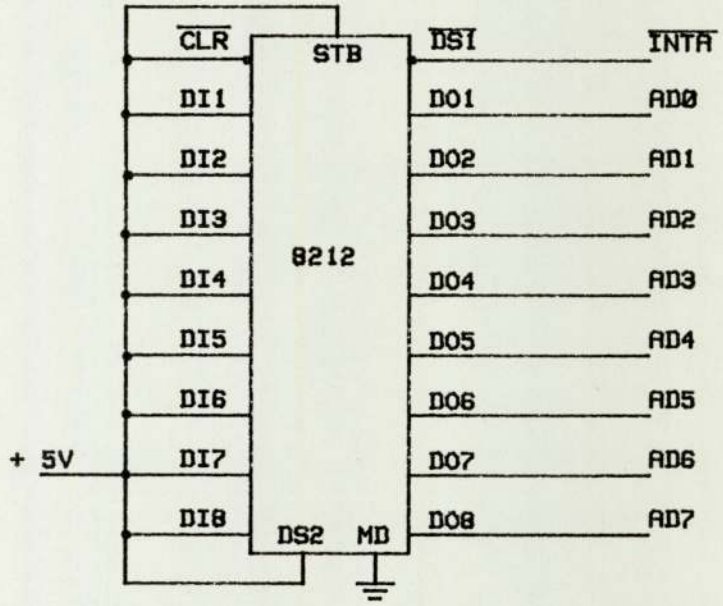
TABLE 6.1 PIN ASSIGNMENT ON THE BACK PLATE OF THE MAIN CONTROL SYSTEM

transmitting information from side A to side B of the transceiver, while the data flow direction will be reversed if the CPU is requiring (reading) data from a device not residing on the CPU board. In order to avoid bus contention, the controlling of this T signal must be done by some logic. Since the only on-board device is the monitor EPROM, the chip select signal for the monitor can be used with other control signals to generate the T signal for this buffer. The CPU will read in data if it performs one of the following operations: reading data from memory, inputting from an input port, or getting an instruction on recognising an interrupt. This T signal generation logic will be done by the NAND gates (74LS00) and the inverter (74LS04). T will be high whenever the CPU requires information from the monitor EPROM and its ports, or writes information onto the data bus. T will become low if the CPU sends out a low $\overline{\text{INTA}}$ signal, or a low $\overline{\text{RD}}$ signal with a high monitor EPROM chip select signal.

6.2.7. *Others*

The CPU ready line input is connected to the back plate of the system and is normally pulled high through a pull up resistor unless a low is sent from a device indicating a busy status. Table 6.1 shows the pin assignment on this back plate for the main control system. The built-in hardware interrupts of the Intel-8085 are constructed in the following way:

FIG 6.5 INTERRUPT INSTRUCTION PORT SUPPLYING THE RESTART 7 INSTRUCTION



TRAP - This interrupt has got the highest priority and is used to continue the system operations after the tunnel profile has been defined.

RST 7.5 - This interrupt is used to execute the routine for moving the boom cutter to the position defined by the operator with the desired speed (fast or slow).

RST 6.5 - This interrupt is used to execute the routine for cutting the tunnel cavity layer by layer.

RST 5.5 - This interrupt is used to execute the routine for trimming the tunnel profile.

INTR - This interrupt is used to stop the execution of the current program segment so that other operations can be selected. This therefore provides a 'change mode' facility to the operator. Fig. 6.5 shows the schematic diagram for the RST 7 instruction port.

All other unused high active input control signals (e.g. HOLD, SID) are grounded to enable normal operation.

6.3. EXPANSION RAM AND EPROM BOARD

This board contains the address decoder, EPROMs and RAMs. There are three 2K EPROMs with addresses from 8000 H to 97FF H and two 1K RAMs with addresses from B800 H to BFFF H. Similar to the first board, the address decoding is done in blocks of 2K from 8000 H to BFFF H. Since the

microprocessor uses the high order addresses for both memories and I/O ports, the address decoder is enabled only when the IO/$\overline{M}$ signal is low to avoid accidental selection of the RAM or the EPROM on this board. A schematic diagram of this board is shown in Fig. 6.6.

The EPROMs and RAMs used are 2716 and Intel-8185 respectively. Since they have standby facilities, they may be disabled when the chips are not selected to reduce the power consumption. These chips are forced to enter a standby state by applying high signals to their $\overline{CS}$ inputs, therefore the use of Intel-8205 (or 74LS138) provides this facility.

The Intel-8185 is an 1K 8-bit static RAM with a multiplexed address and data bus, therefore the buffered AD_0 - AD_7 and ALE of the microprocessor are connected to the corresponding pins of this RAM. These RAMs are selected when the address is within the 2K block from B800 H. However, only one chip is enabled by the buffered A_{10} which is connected to $\overline{CE}_1$ of one chip and to CE_2 of the other. On the contrary, the 2716 EPROMs are selected purely on the chip-select ($\overline{CS}$) signals, since they are 2K EPROMs, and the $\overline{OE}$ pins are enabled by the buffered $\overline{RD}$ signal of the CPU.

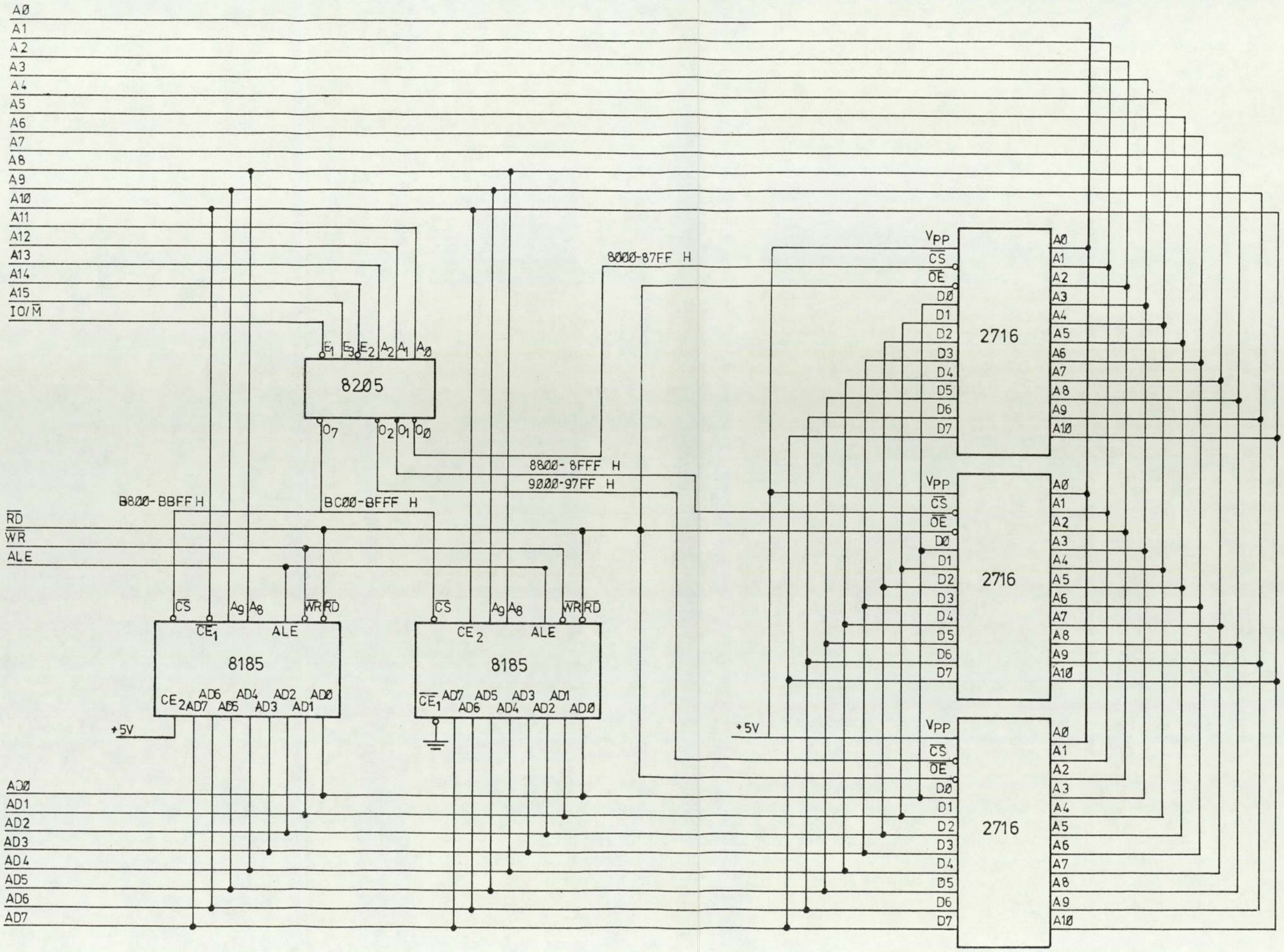


FIG. 6.6 Schematic Diagram For Expansion RAM and EPROM Board

6.4. ARITHMETIC PROCESSING UNIT BOARD

Mathematical operations can be carried out either by software (program) or by hardware (arithmetic processing unit, APU). Hardware operation will speed up the performance and enable the valve control signals to be given at a high frequency. A schematic diagram for this APU board is given in Fig. 6.7.

The APU chosen is the Intel-8231A for its high operation speed and ease of interfacing to the CPU. This APU is selected with the high order address FD H and FC H for its command port and its on-chip data stack respectively. It can therefore be selected as an I/O port or a memory.

Memory mapping technique can increase the flexibility by enabling more instructions to be used for communication with the APU at the expense of the memory addresses FC00 H to FD00 H ($\frac{1}{2}$ K) which cannot be used for ordinary memories.

The APU's ready signal output is connected to the CPU so that the CPU can be halted by the 8231A when this APU is busy and data transfer is requested by the CPU. The ready line is pulled low when the APU is busy and will be pulled high again when it is ready for data transfer.

Since the APU can operate on a maximum frequency of 4 MHz, it is driven by the 3.072 MHz clock output of the CPU employed. Similar to the CPU, all the unused high active control inputs are grounded, while those low active control inputs are tied to high (+5v).

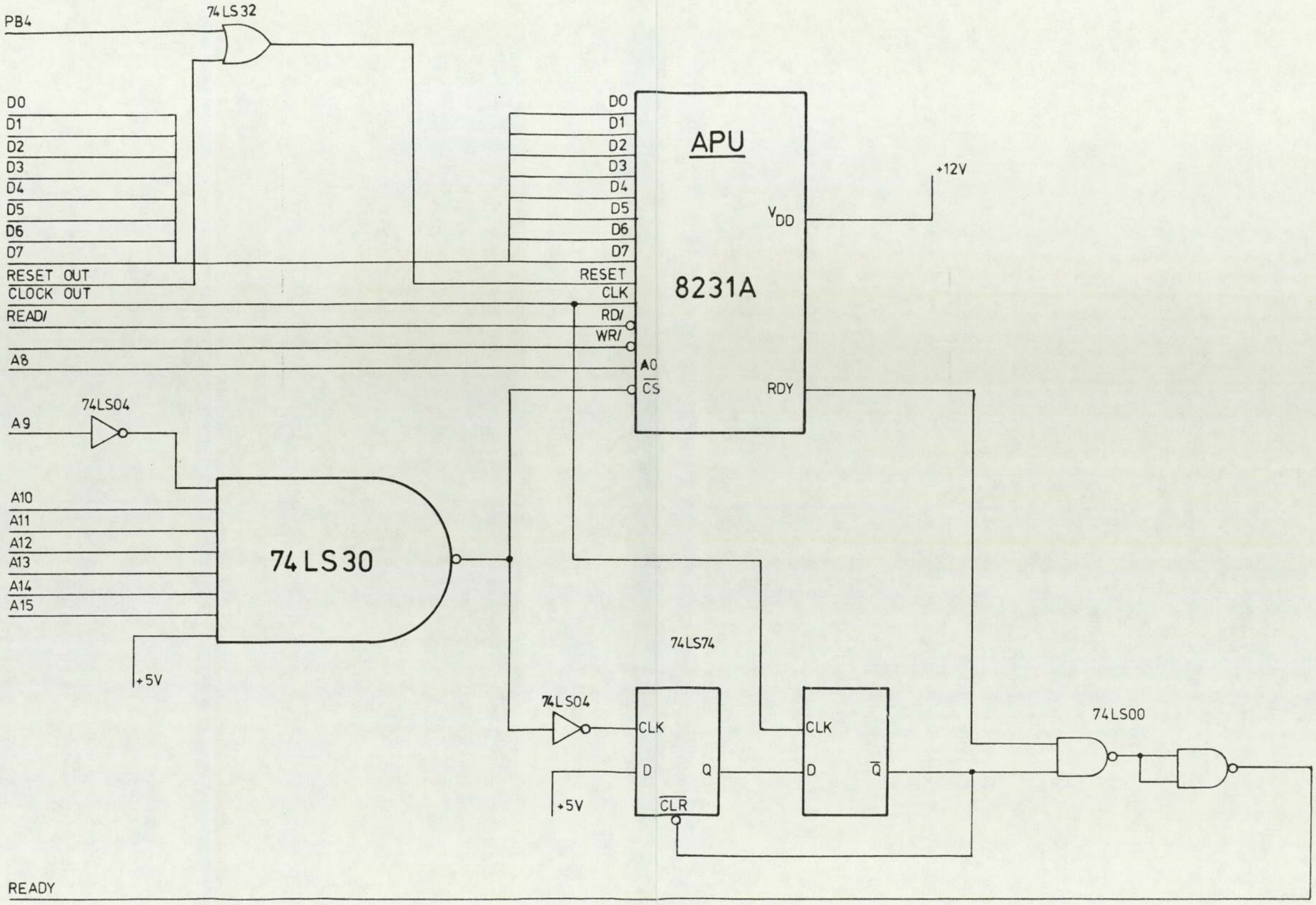


FIG. 6.7 SCHEMATIC DIAGRAM FOR THE APU EXPANSION BOARD

6.5. INTERFACE TO THE DATA CAPTURING SYSTEM

The proposed system is a dual microprocessor one, therefore an interface must be provided to enable the communication between these two parts. A schematic diagram for this interface is presented in Fig. 6.8.

Both the microprocessors will be running in parallel unless data transfer from one to another is performed. When the main control microprocessor wants data from the data capturing system, it latches a low signal onto bit 5 of port 1 and this signal is inverted and then transmitted to the HOLD input of the data capturing system's CPU. The inverter is installed to ensure normal operation at power on. On recognising this signal, the second CPU will issue a HOLDA (hold acknowledge) signal onto its HOLDA output and relinquish the address and data buses as soon as the completion of the current bus transfer. The $\overline{RD}$, $\overline{WR}$ and IO/$\overline{M}$ control lines of this CPU will also be tristated. On receipt of the HOLDA signal, the main microprocessor can start to read data from the data capturing system.

The buffers for address, data and control lines in this interface card normally have tristated outputs unless data transfer is requested. This tristate arrangement enables the two systems to operate normally without any undesirable interference. The data and address buffers (8286s) are enabled by pulling the output enable ($\overline{OE}$) lines low; when the buffer (8216) for the control lines is enabled by pulling the $\overline{CS}$ signal low. The $\overline{RD}$ signal is connected to the T

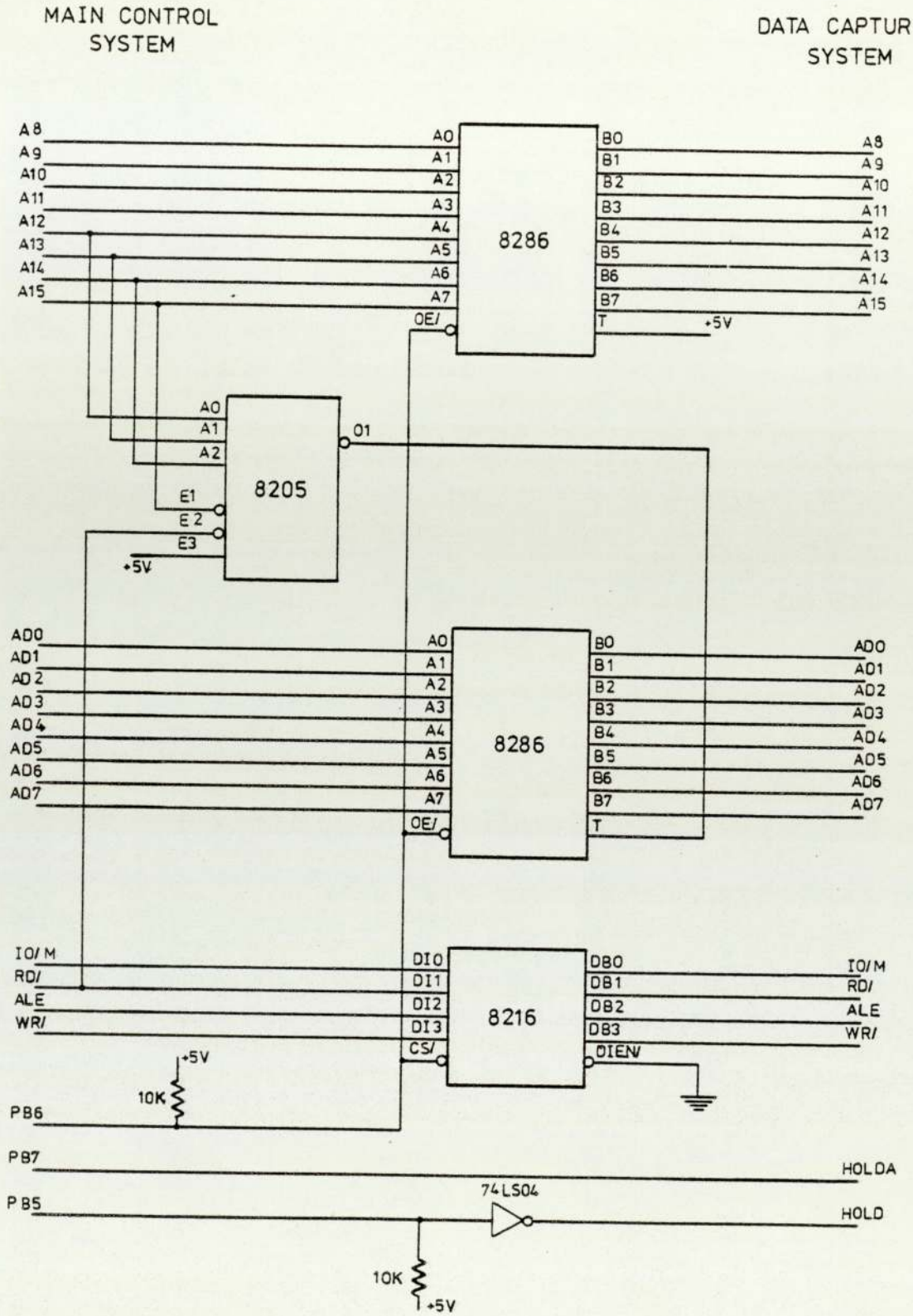


FIG. 6·8 SCHEMATIC DIAGRAM FOR INTERFACING THE MAIN CONTROL SYSTEM WITH THE DATA CAPTURING ONE

PIN	MNEMONIC	DESCRIPTION
1c	GND	SIGNAL GROUND
2c	+5V	+5 VOLTS
3c	AD0	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 0 (Main System)
4c	AD1	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 1 (Main System)
5c	AD2	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 2 (Main System)
6c	AD3	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 3 (Main System)
7c	AD4	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 4 (Main System)
8c	AD5	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 5 (Main System)
9c	AD6	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 6 (Main System)
10c	AD7	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 7 (Main System)
11c	A8	HIGH ORDER ADDRESS BIT 8 (Main System)
12c	A9	HIGH ORDER ADDRESS BIT 9 (Main System)
13c	A10	HIGH ORDER ADDRESS BIT 10 (Main System)
14c	A11	HIGH ORDER ADDRESS BIT 11 (Main System)
15c	A12	HIGH ORDER ADDRESS BIT 12 (Main System)
16c	A13	HIGH ORDER ADDRESS BIT 13 (Main System)
17c	A14	HIGH ORDER ADDRESS BIT 14 (Main System)
18c	A15	HIGH ORDER ADDRESS BIT 15 (Main System)
19c	IO/$\overline{M}$	I/O PORT OR MEMORY CONTROL SIGNAL (Main System)
20c	RD/	READ CONTROL SIGNAL (LOW ACTIVE) (Main System)
21c	ALE	ADDRESS LATCH ENABLE (Main System)
22c	TRAP/	INVERTED TRAP INTERRUPT (Main System)
23c	RST 7.5	RESTART 7.5 INTERRUPT FOR MOVING THE BOOM (Main System)
24c	-	
25c	-	
26c	WR/	WRITE CONTROL SIGNAL (LOW ACTIVE) (Main System)
27c	-	
28c	-	
29c	-	
30c	-	
31c	+5V	+5 VOLTS
32c	GND	SIGNAL GROUND
1a	GND	SIGNAL GROUND
2a	+5V	+5 VOLTS
3a	AD0	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 0 (Data System)
4a	AD1	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 1 (Data System)
5a	AD2	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 2 (Data System)
6a	AD3	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 3 (Data System)
7a	AD4	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 4 (Data System)
8a	AD5	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 5 (Data System)
9a	AD6	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 6 (Data System)
10a	AD7	MULTIPLEXED LOW ORDER ADDRESS/DATA BIT 7 (Data System)
11a	A8	HIGH ORDER ADDRESS BIT 8 (Data System)
12a	A9	HIGH ORDER ADDRESS BIT 9 (Data System)
13a	A10	HIGH ORDER ADDRESS BIT 10 (Data System)
14a	A11	HIGH ORDER ADDRESS BIT 11 (Data System)
15a	A12	HIGH ORDER ADDRESS BIT 12 (Data System)
16a	A13	HIGH ORDER ADDRESS BIT 13 (Data System)
17a	A14	HIGH ORDER ADDRESS BIT 14 (Data System)
18a	A15	HIGH ORDER ADDRESS BIT 15 (Data System)
19a	IO/$\overline{M}$	I/O PORT OR MEMORY CONTROL SIGNAL (Data System)
20a	RD/	READ CONTROL (LOW ACTIVE) (Data System)
21a	ALE	ADDRESS LATCH ENABLE (Data System)
22a	HOLD	HOLD THE DATA CAPTURING MICRO (HIGH ACTIVE)
23a	HOLDA	HOLD ACKNOWLEDGE FROM THE DATA CAPTURING MICRO
24a	PB6	ENABLE THE INTERFACE BETWEEN THE 2 SYSTEMS (LOW ACTIVE)
25a	PB5	INVERTED HOLD SIGNAL (FROM MAIN SYSTEM)
26a	WR/	WRITE CONTROL (LOW ACTIVE) (Data System)
27a	-	
28a	-	
29a	-	
30a	-	
31a	+5V	+5 VOLTS
32a	GND	SIGNAL GROUND

TABLE 6.2 PIN ASSIGNMENT ON THE BACK PLATE FOR THE INTERFACE CARD

input of the multiplexed low order address and data buffer to enable the read data operation. When T is high, signal flow is from side A to side B, hence passes the AD_0 - AD_7 onto the respective bus of the second system and addresses the appropriate memory. When $\overline{RD}$ becomes low, the signal flow direction will be reversed from side B to side A of the buffer, then the host microprocessor can read in data through this buffer. The other two buffers are configured to have information flowing from the main microprocessor system to the data capturing one when they are enabled. The flow diagram for this interfacing procedure has been given in Fig. 6,3, and the pin assignment on the back plate for this interface card is shown in Table 6.2.

CHAPTER 7

TESTS AND RESULTS

7.1. Introduction

The control algorithms and the control circuits have been derived and built. It has been mentioned in Chapter 3 that a switching function can be derived to determine when the valves should be closed. The use of this switching function will be expected to reduce the time taken in positioning the boom ripper. It has also been shown in Chapter 4 that the use of an APU will facilitate the mathematical operations. However, this will mean a more expensive system. Tests have been done using various control strategies: (A) control based on simply comparing the feedback position with the desired position, and (B) the more complicated control employing the switching function evaluations.

7.2. Control Strategies

7.2.1. *Simple Comparison With Slow Speed*

The simplest control strategy in this boom positional control is to include a unity feedback. In a parallel project, J H Knight of the Mechanical Engineering Department at Aston University has developed a data capturing and information retrieval system, which provides the positional information on the desired input and the actual output. These data are supplied in the digital forms and are also arranged to give a unity feedback.

In this simple comparison method, the input and the output data are compared in determining the required signals sent to the electro-hydraulic valves to give the desired boom

movement. It has been mentioned in Chapter 3 that the control by simply comparing the input and the output data would not be good enough because the physical properties (e.g. the inertial load, the fluid compressibility, the valve characteristics etc.) of the system might cause excessive overshoot. However, the inherent limitation in the accuracy of the 8-bit analogue-to-digital converter (ADC), ± 1 least significant bit (LSB), suggests that this simple comparison method will be adequate when the boom is moving at the slow speed.

7.2.2. *Switching Function Evaluation*

It has been shown in Chapter 3 that the employment of the switching function, which determines the switching signals depending on the physical properties and the current conditions of the system, would reduce the overshoot and hence would improve the positional control of the boom ripper. This will be particularly helpful if the boom is moving at its fast speed, therefore tests have been done using the switching function for the boom fast move mode.

The information on the system performance is provided by the data capturing system developed by J H Knight; while the mathematical operations are handled by the APU to increase the speed of operation and to reduce the complexity in the software of the control system. The valve performance has been tested by J H Knight and his results suggest that the valves can be approximated to 'On-Off' valves with time delays. Therefore the switching functions tested are those derived for this valve type (i.e. 'On-Off' valve with time delay).

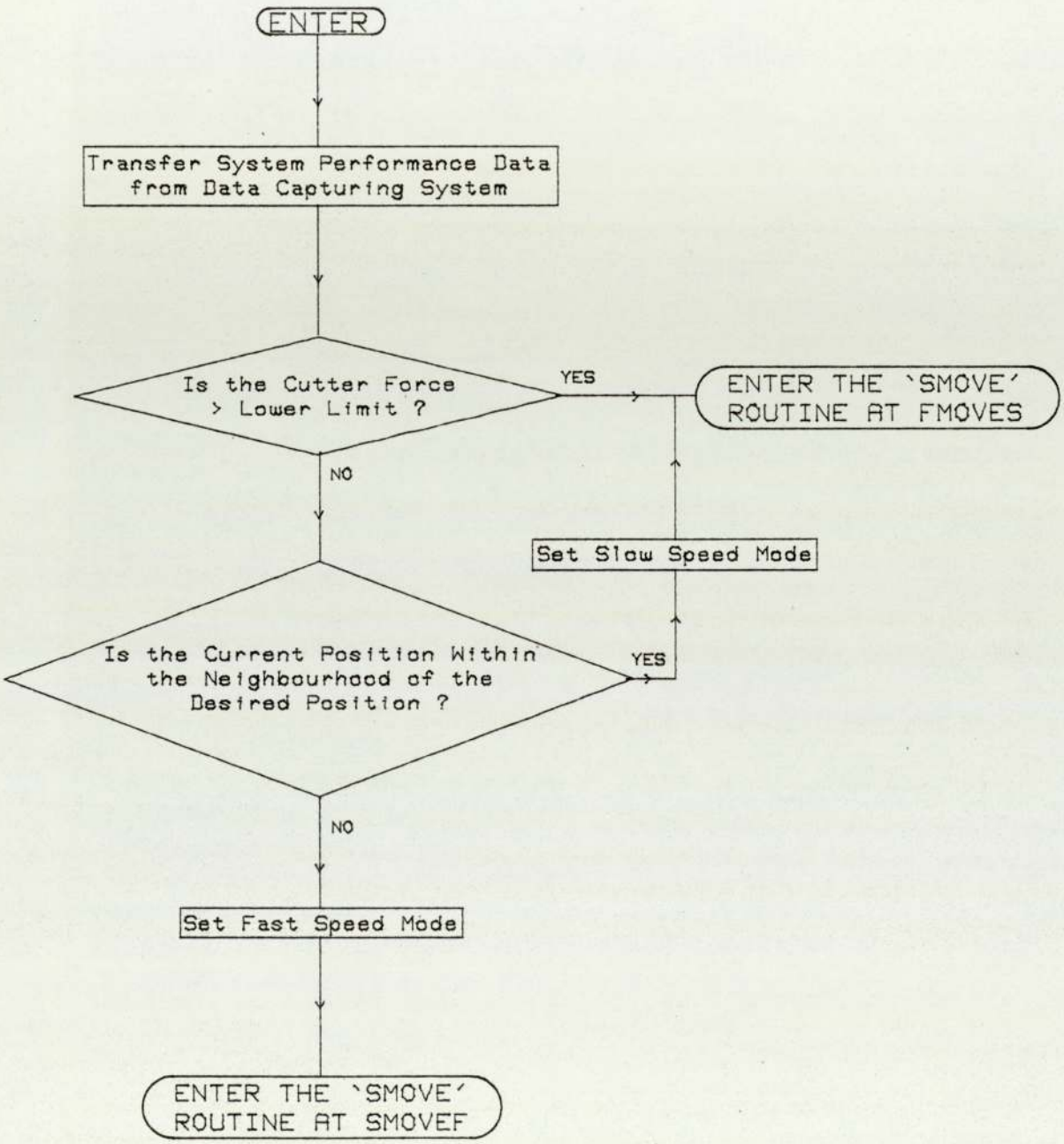


FIG 7.1 Flow Diagram for the `FMOVE' Routine Using Simple Comparison and the `SMOVE' Routine

7.3. Boom Moving at High Speed - 'FMOVE' Routine

It has been mentioned that two control strategies have been tested for the positional control of the boom ripper. The simple comparison method may cause an undesirable overshoot if the boom is moving at a high speed. Therefore, in the simple comparison method, the boom movement will be restricted to the slow move mode, if the cutter is within a specified neighbourhood of the desired position, to avoid any undesirable overshoot. Figure 7.1 shows the flow diagram for this control method, and the listing of the assembly language program for this routine is included in Appendix G.

When this routine is entered, the necessary information will be transferred from the data capturing system. The current boom slew position is checked against the desired slew value: if it is within the specified neighbourhood of the desired slew position, then the slew movement will be set to the slow move mode. Similarly, the current boom altitude is compared with the desired elevation and the slow move mode will be set if it is within the neighbourhood of the desired altitude. After this checking for the permitted speed mode has been done, the 'SMOVE' routine will be entered (please refer to the listing in Appendix G). The determination of the appropriate switching signals and the other checkings will be done in the 'SMOVE' routine by simple comparison. A detailed description of the 'SMOVE' routine has been given in Chapter 5.

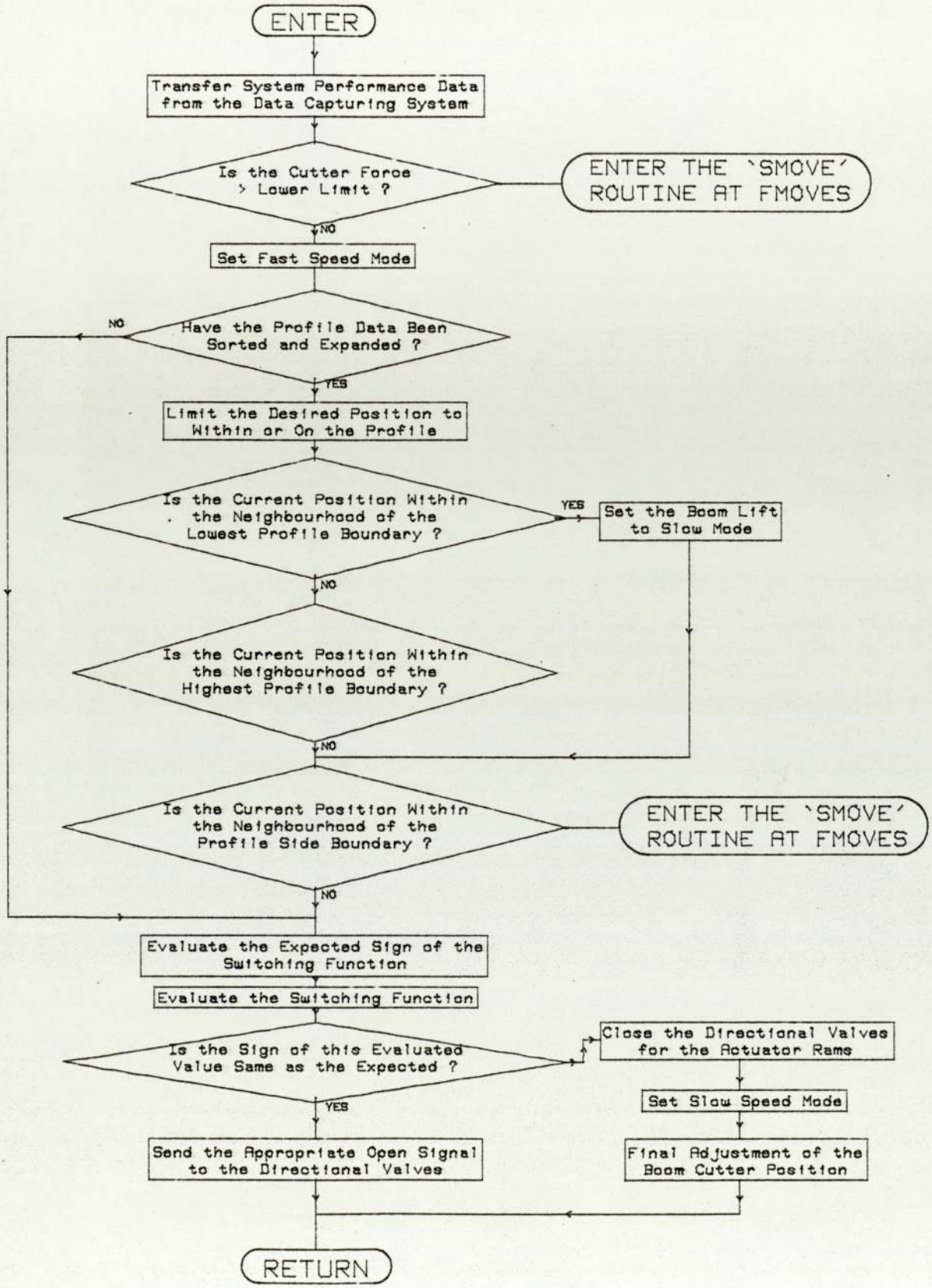


FIG 7.2 Flow Diagram for the `FMOVE' Routine Using the Switching Function and the `SMOVE' Routine

Another control strategy can be used for this boom movement at its high speed is the employment of the switching functions. Figure 7.2 shows the flow diagram for this algorithm and the listing of the assembly language program for this routine is given in Appendix H. Similar to the other routine, the system performance data are transferred from the data capturing system on entering this routine. If the cutter force is greater than the upper limit for the high speed movement, then the 'SMOVE' routine will be entered at FMOVES so that the boom can be moved only at the slow speed. Otherwise, the fast speed will be set. If the profile data have not been sorted and expanded, then the program will go to continue its execution from determining the switching signals.

However, if the profile data table has been properly established as described in Chapter 5, the following actions will be taken before it evaluates the switching function. Similar to the 'SMOVE' routine, the desired position is limited to within or on the profile boundary defined by the operator. The current position is examined: if it is close to either the lowest profile boundary or the highest one, then the slow move mode will be set for the vertical movement of the boom; if it is close to either side of the profile boundary, then the 'SMOVE' routine will be entered at FMOVES. The reason for restricting the boom movement to its slow mode, when it is currently near the profile boundary, is to make the checking for permissible directions of the boom movement easier (please refer to Chapter 5). In this system, the criterion for being close to the boundary is 16D (10H) bits.

If either the current boom position is free from the profile side boundary or the profile data have not been sorted and expanded, then the 'FMOVE' routine will continue its operation. It will determine the switching signals for the two movements (i.e. the lifting and the slewing) separately using similar procedures. The expected sign of the switching function will be determined by simply using the current and the desired positions. Then the captured data will be converted into the APU recognised binary floating point (BFP) format. These BFP data are then transferred to the APU stack and the appropriate switching function will be evaluated. The result will be used to determine the signal: if the evaluated function value has the same sign as the expected, then a signal will be sent to operate the pilots of the directional valves (the direction of movement will depend on the sign of the function value); otherwise, it indicates a change in the sign of the switching function (i.e. the criterion fn=0, for switching off the valves has been satisfied), then the directional valves will be closed. After all the directional valves (both valve sets for the lifting and the slewing) have been closed, the final position will be adjusted in the slow speed mode (this is similar to the 'inching' process used by the operator with an open-loop control system).

Because of the hardware configuration of the hydraulic circuit for the boom ripper under consideration, the heavy mass of the boom causes (i) the lowering movement having preference over the boom slewing and (ii) the slewing

movement having preference over the boom raising. It has been observed in the experiment that the boom would lower down whenever the fast mode is set and both sets of valves (i.e. the valves for both the vertical and the slewing actions) are operating. The boom will lower down even though the raise up signal has been sent to the pilots of the directional valves. Therefore an algorithm has been built into the system for the vertical movement to have preference over the slewing whenever the fast speed mode is set.

Another consideration has been the time delay in the valve operation. Because of the time delay, the boom will neither move nor stop immediately after the signal has been sent to the directional valves. If the boom is near the desired position (say 2 bits from it) and the correction signal has been sent to the valves, then the boom will not move until the valves are opened and its movement cannot be detected unless a two-bit change in the boom position has been observed (the data capturing system regards a change within 1 bit as 'no change' because the ADC has an accuracy of ± 1 LSB). As soon as this boom movement has been detected, the switching off signal will be sent to close the directional valves. However, because of the time delay in the valve operation, the boom will not stop immediately, therefore an overshoot results and another correction signal will then be sent to the valves. Consequently, the boom will be hunting about this desired position. Therefore, the boom movement is restricted to its slow move mode if it is near the desired position (in this system, the criterion is ± 4 bits from the desired position) to prevent hunting.

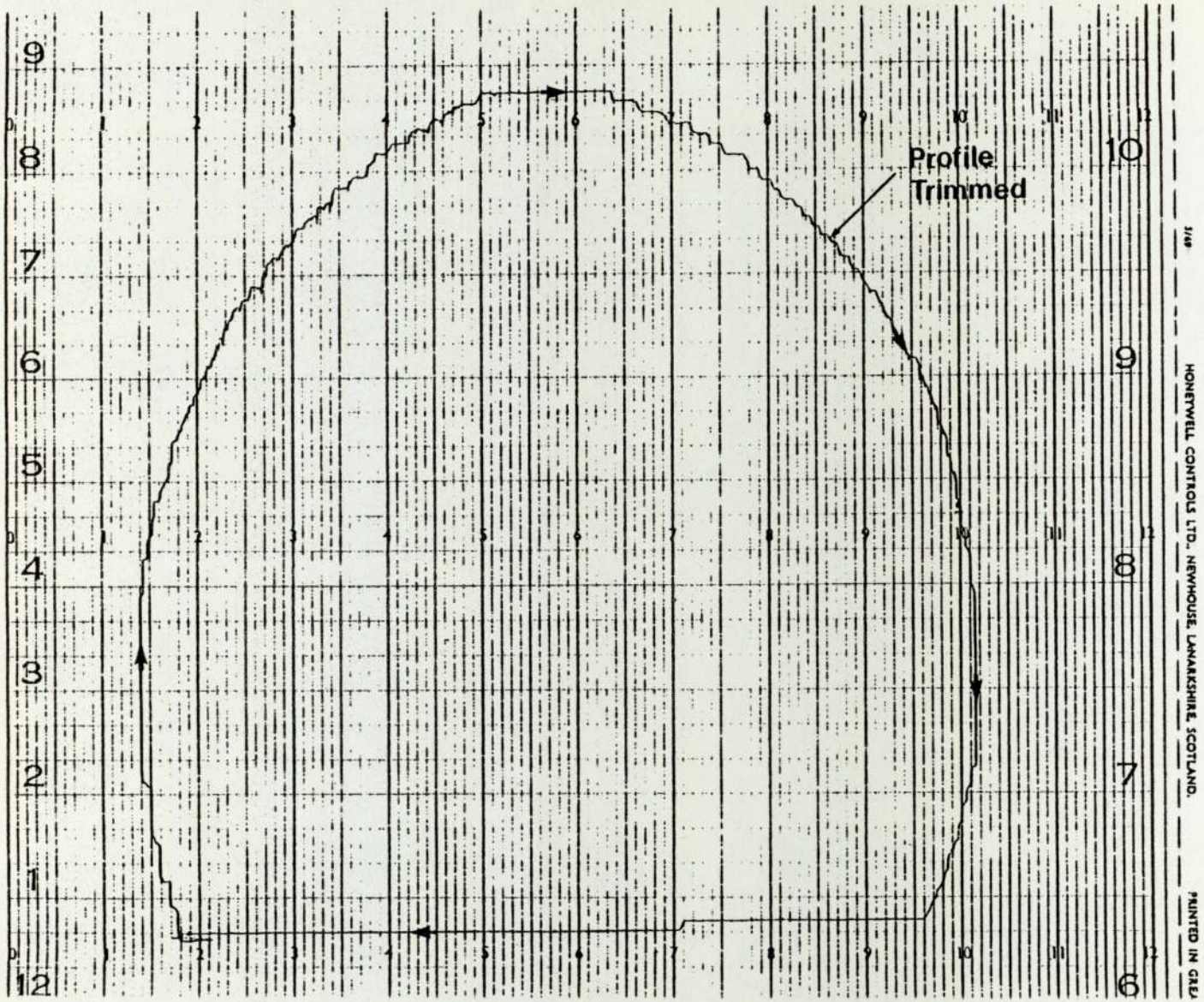


Fig. 7.3 Trace of the Boom Cutter Movement in Trimming the Profile

7.4. Test Results

7.4.1. *Boom Control by Simple Comparison With Speed Limitation*

The control strategy, using simple comparison between the current and the desired boom positions, has been coded (programmed) and tested. In this method, the slow move mode will be set when the comparison reveals that the two positions are near to each other. The results are shown in Figures 7.3 and 7.4 for the profile trimming and the cavity cutting processes respectively. It can be seen from these traces that this control strategy is satisfactory: the accuracy is within the ADC tolerance (± 1 LSB). It can be noted from Figure 7.3 that the downward movement of the boom in trimming the profile boundary has got a larger step than the upward movement, this is because the dynamics of the boom ripper are different in both directions. The heavy mass of the boom causes it to lower more before the valve can be closed completely, while this heavy mass will help in stopping the boom from its upward movement.

It can also be seen from these figures that the boom movement is within the defined profile boundary. This can be easily observed when the cutter is ripping the upper cavity layers. As described in Chapter 5, the starting and ending points of the cavity layer to be cut are half the vertical cutting step size away from the profile boundary. In order to simplify the control program, the cutter movement is restricted to within or on the defined profile boundary for all movements. Therefore the cutter will advance to the next layer along the profile boundary instead of a distance

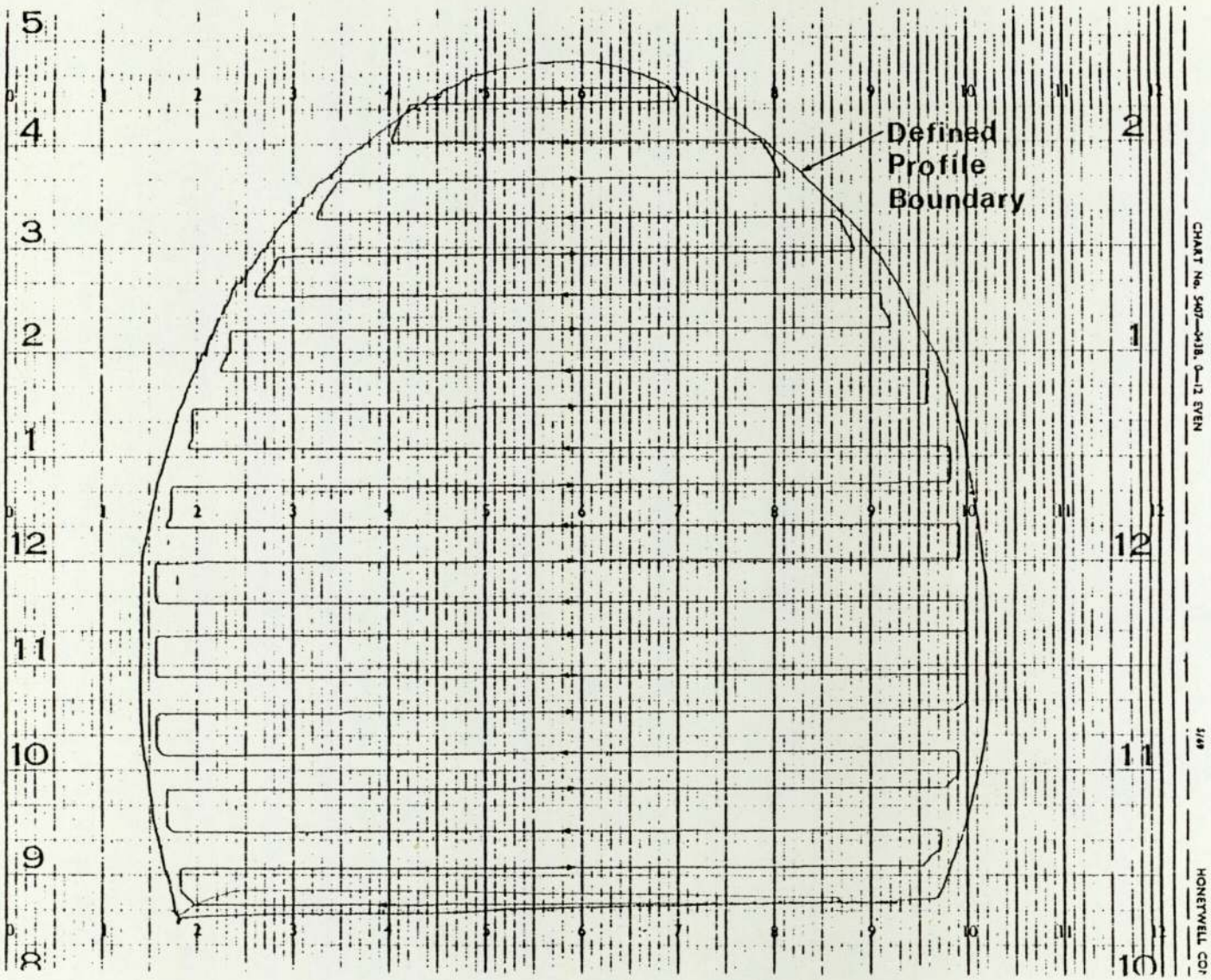


Fig. 7.4 Trace of the Boom Cutter Movement in Cutting the Cavity

away from it when the cutter is ripping the uppermost cavity layers and hits the profile boundary.

It is difficult to demonstrate the 'Move Boom' facility in a still figure (or recording), therefore only the observations are reported. When the boom is required to move fast to a position, it will move at the high speed to a region near the desired position and then it is restricted to moving slowly towards that and stop when it reaches this position. If the boom starts from a position near the profile boundary, it will move slowly away from the boundary until it enters the 'free' space where it can move at high speed if the cutter load permits. The boom will not move outside the defined profile boundary.

7.4.2. *Boom Control with Switching Function Evaluation*

It can be seen from the simulation studies in Chapter 3 that the time delay in the valve operation, the external force and the ram velocity are the dominant factors in determining the instant to send the switching off (closing) signals to the valves.

The time delay in the valve operation has been measured by Knight to be approximately 390 mSec with a system supply pressure of 138 bars. However, this time delay will depend on the system supply pressure and the pressure difference across the valve. In the system under test, a metering out technique is used to provide the different boom moving speeds, therefore the time delay in the valve operation will vary if the speed changes. This time variation

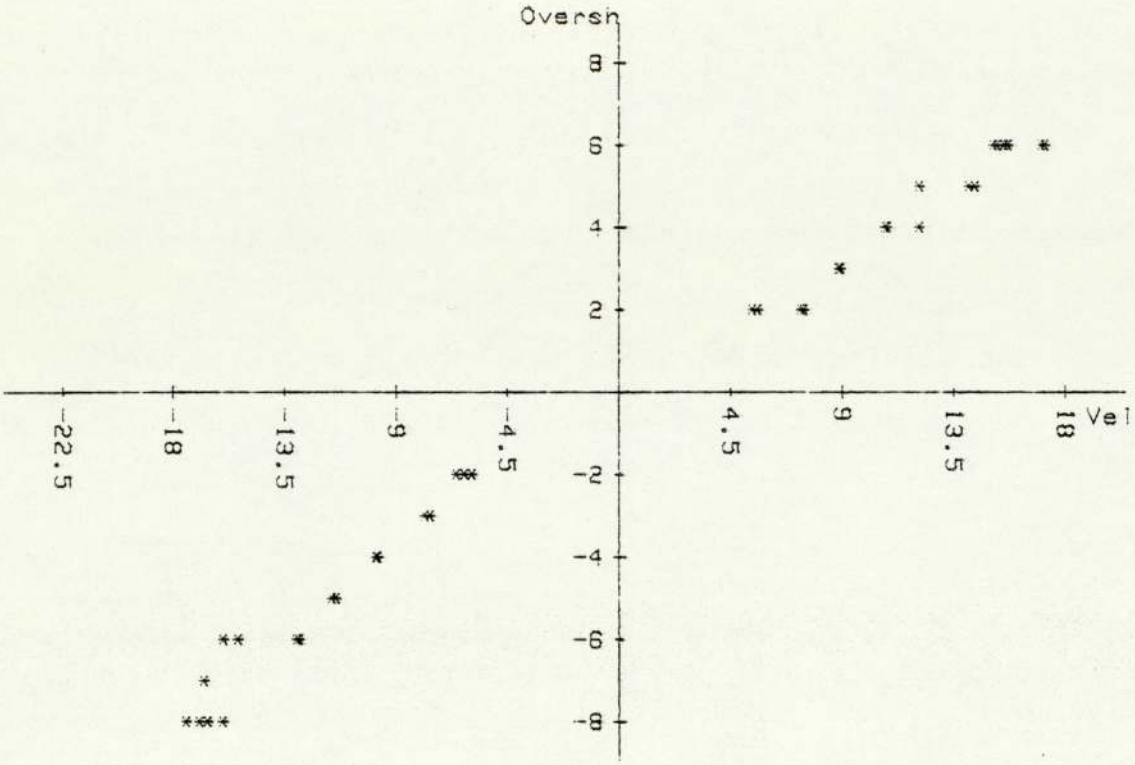


Fig 7.5 Boom Overshoot (Bits) - Initial Slewing Velocity at Switching (Bits/Sec)

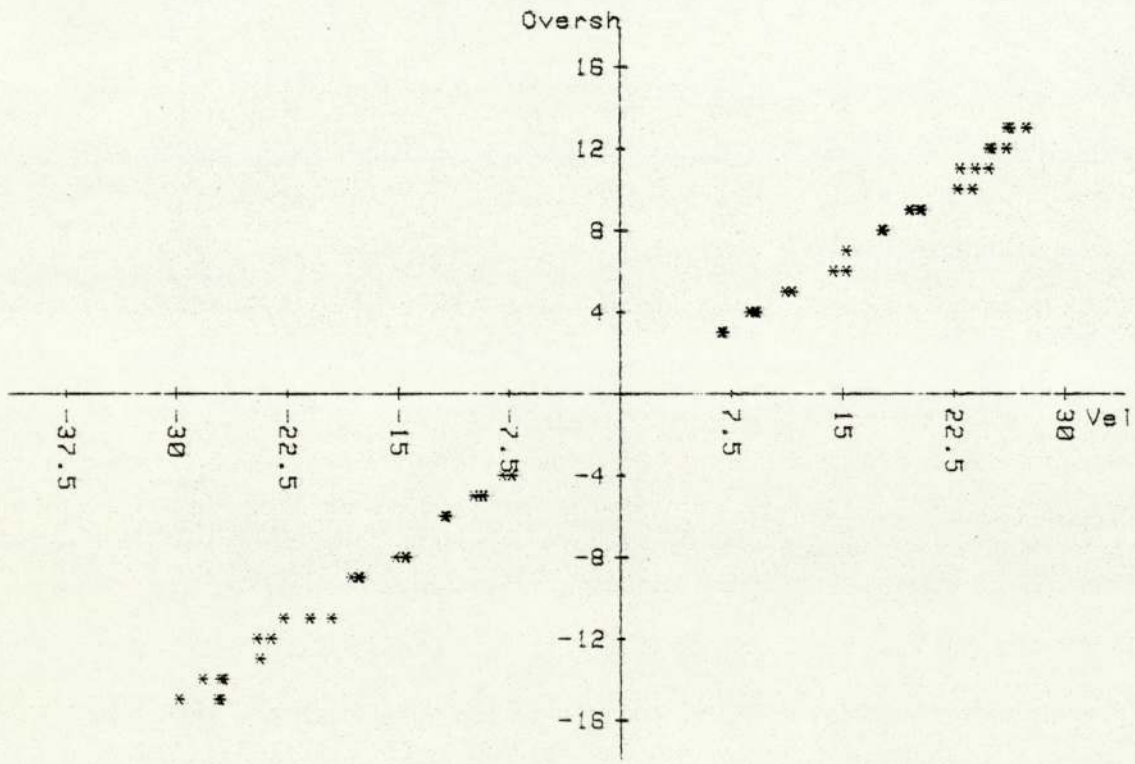


Fig 7.6 Boom Overshoot (Bits) - Initial Lifting Velocity at Switching (Bits/Sec)

and the complexity in the system dynamics make it difficult to theoretically determine the coefficients in the switching functions for this system, and experimental determination of the coefficients has been used.

The external force is mainly due to the heavy weight of the boom. It has been mentioned in Chapter 3 that the effect of the external force is mainly on the constant term in the switching function, while the time delay in valve operation also affects the coefficient of the velocity term in this function, and the acceleration is so small that its contribution to the switching function can be neglected (because the steady state can normally be achieved before the switching signals are sent). Therefore a plot of the amount of overshoot (the boom travel since the application of the closing signal before it comes to a rest) against the velocity at which the boom is moving when the switching signal is sent, will give the required coefficients. Figures 7.5 and 7.6 are obtained for the horizontal and vertical (lifting) movements in the units used by the data capturing system. The horizontal slew and the vertical elevation ranges are both scaled from 0 to 255D bits in the data capturing system. In physical units, the horizontal slew is from $-52^{\circ}57'$ to $52^{\circ}57'$, while the vertical elevation is from $-34^{\circ}12'$ to $44^{\circ}46'$.

These graphs show that the overshoots and the initial boom velocities (when the closing valve signals are sent) have got approximately linear relationships. Therefore

the slopes of these graphs will give the coefficients for the velocity terms in the switching functions and the y-intercepts will yield the constant terms.

Although it is difficult to theoretically determine the coefficients for the switching functions, an attempt has been made to estimate the coefficients. The following parameters are used for the lifting rams:

Mass of Boom = 3070 kg (Calculated from the measured boom dimensions and assumed densities).

Mass of Ram = 7 kg.

Effective External Force = 40 kN (due to the weight of boom, this ranges from 37.65 to 43.67 kN depending on the boom elevation).

Valve Dimensions = 6 mm x 3 mm.

On-Off Valve with Time Delay = 390 mSec.

From the switching function in Section 3.6.3 with the parameter adjustment (for an asymmetric system with external force) mentioned in Sections 3.6.5 and 3.7, the switching function for this lifting ram has been reduced to the form as Equation (4.1):-

$$\lambda_1 \ddot{y} + \lambda_2 \dot{y} + \lambda_3 y + \lambda_4 y_d + \lambda_5 = 0$$

where $\lambda_1 = 0.998$

$\lambda_2 = 2472$ (mainly affected by the valve time delay)

$\lambda_3 = 1$

$\lambda_4 = -1$

$\lambda_5 = -0.04568$ (both the applied force and the valve time delay will contribute to this).

The magnitude of the acceleration term is very small compared with the others in this function, therefore the switching function can be reduced to:

$$\lambda_2 \dot{y} + (y - y_d) + \lambda_5 = 0$$

After these coefficients have been converted into the units used by the control system. The coefficients for the velocity (λ_2) is 27113, and the constant term (λ_5) is -0.188. The experimentally determined values (from Figure 7.6) are 29179 and -0.5 respectively. This suggests that the time delay in the system is longer than that used in the theoretical prediction, while the actual load acting on the hydraulic rams is higher. The reasons for a longer time delay in the system are thought to be an increase in the chamber pressures due to the metering out technique used and the actual working system pressure has been set at 110 Bars. The theoretically determined coefficient for the velocity term differs from the experimental value by 0.8%. Because of the complexity in the dynamics of the system, estimation of the external force cannot be accurate. If the quoted weight of the boom (3.25 tons) by the manufacturer has been assumed, the external force will range from 42.2 to 47.1 kN. If the average value (45 kN) of this force range is used, then the theortical prediction will give 27310 and -0.161 for λ_2

and λ_5 respectively. When the experimental accuracy of the data capturing system is considered: the 8-bit ADC used has an error of ± 1 LSB and the boom velocity will be calculated whenever a positional change of three bits or more has been detected. Therefore an absolute positional error of 1 bit and an absolute error of 2 bits in the velocity calculated are expected for the worst cases. Since the accuracy in the 16-bit timer is $1/(2^{16}-1)$, the error in the velocity calculation is mainly due to the ADC, this is expected to be ± 2/3 (± 66.67%). If these have been considered, then the slopes (λ_2) will lie between 29200 and 27090, while the y-intercepts (λ_5) will have values between 6.16 and -6.9. The theoretical values based on the 390 mSec time-delay on-off valve and a 45 kN external force also fall within this region.

Similarly the theoretically predicted values of λ_2 and λ_5 for the slewing rams are 23443 and 0.01 based on on-off valves with 390 mSec time-delay. However, Figure 7.5 shows that the relationships between the overshoots and the velocities can be described by two distinct lines depending on the direction of the boom slew. Since the boom slewing is controlled by two different sets of poppet valves, the time-delay for valve operation may be different. The experimentally obtained coefficients (from Figure 7.5) are: λ_2 equals to 24894 and λ_5 is -0.59 when the boom slews to the right, but λ_2 is 30303 and λ_5 equals to 0.98 if the boom slews to the left. The slopes of these lines will give the approximate time-delays in the valves: 414 mSec for the valves giving the right

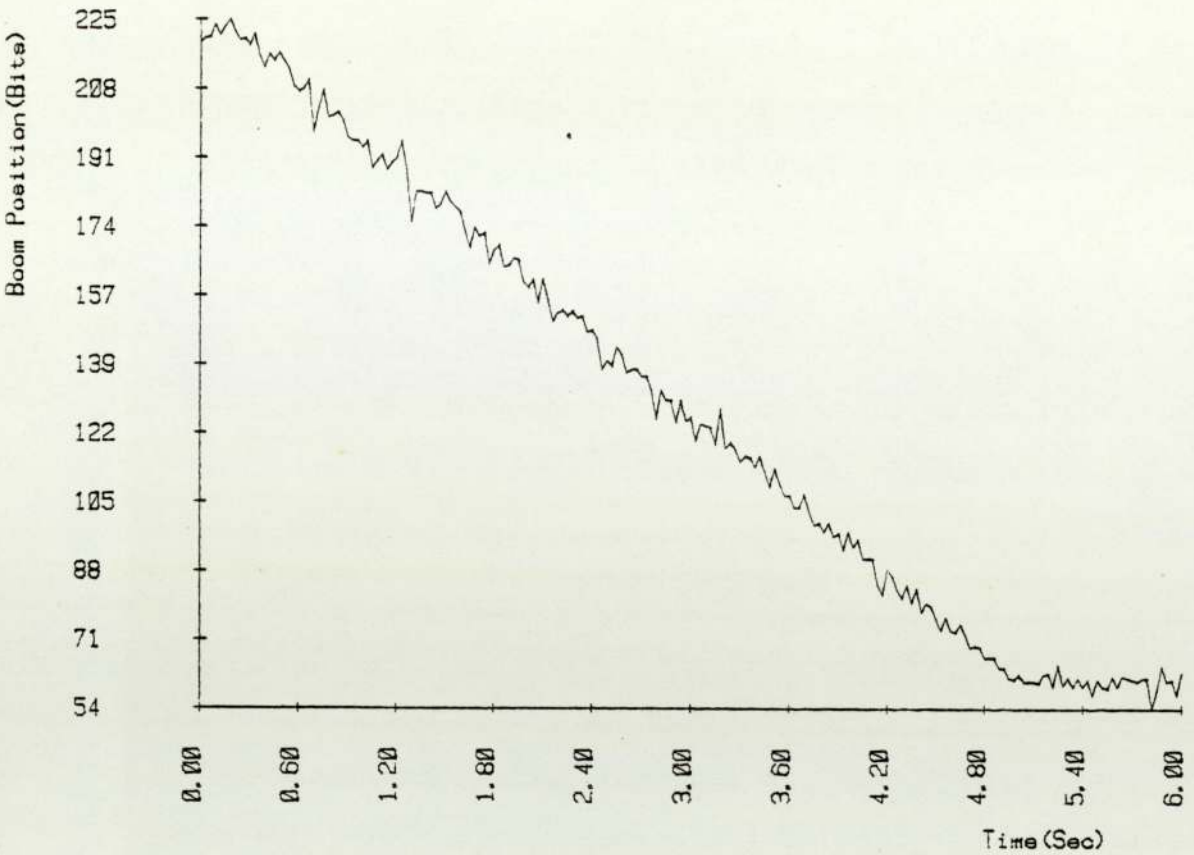


FIG 7.7 Boom Lowering - Time Response

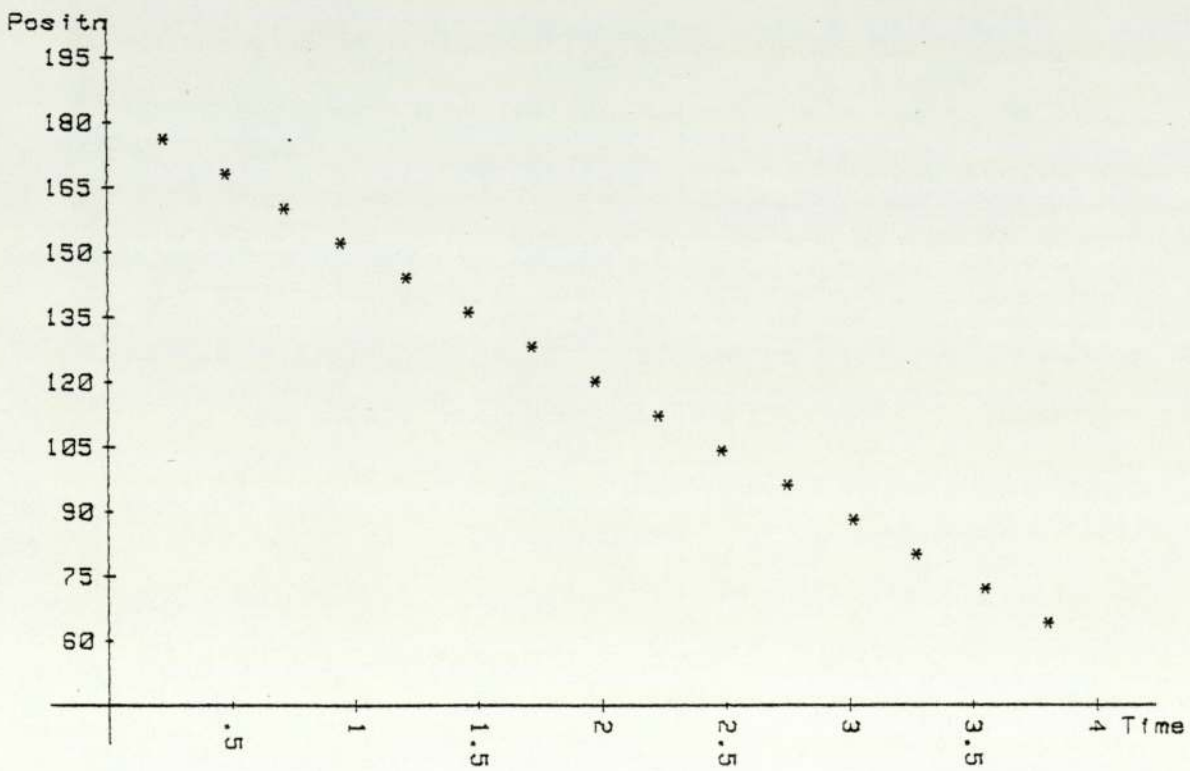


Fig 7.8 Boom Lowering Position (Bits) - Time (Sec) Response

slew, and 505 mSec for the valves giving the left slew (these valves are slower than the measured value for the lifting rams --- 390 mSec). If these values are used in re-calculating the switching coefficients. The following values are obtained: 24049 for λ_2 and -.05 for λ_5 if the boom slews to the right, 30065 for λ_2 and .06 for λ_5 if the boom slews to the left. If the experimental errors in the data capturing system have been taken into account, the values for λ_2 can lie between 18182 and 90910 for the right slew valves, while they are within the range 14936 and 74682 for the left slew valves. Similarly, the ranges for the coefficients, λ_5, are: -1.59 to 0.41 for the valves slewing the boom to the right, -0.02 to 1.98 for those causing the boom to slew towards the left.

Figure 7.7 shows an analogue recording of the time response for the boom lowering while Figure 7.8 shows the equivalent recording by the data capturing system. It can be seen from the analogue trace that the signal is very noisy. However, the magnitude of this noise is so small that it can hardly be detected by the data capturing process. Therefore, the values of the velocities and the positions captured can be used in evaluating the switching function without any correction (adjustment) to compensate for the noise.

Figures 7.9 and 7.10 show the results of the tests on the boom slewing and lifting using the simplified switching functions as shown on P.139. The initial and final errors together with the velocities, at which the boom was moving when the 'close' signals

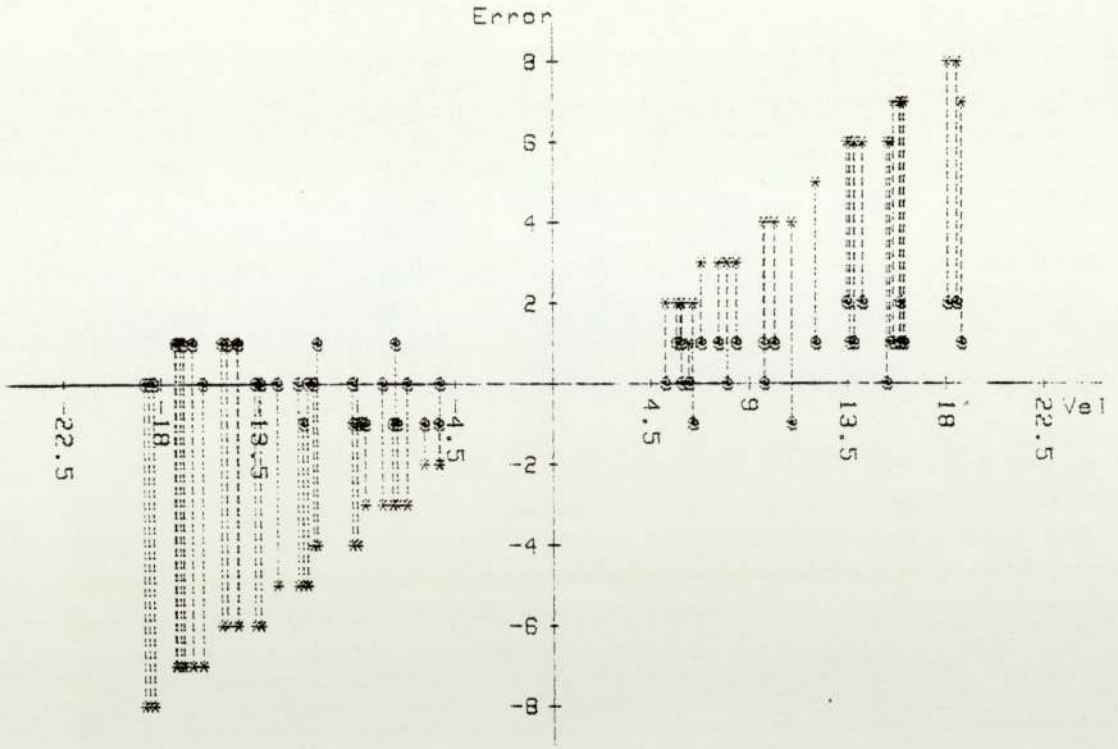


Fig 7.9 Error (Bits) - Boom Slewing Velocity (Bits/Sec)

* --- Initial Error When the Switching Signal is Sent
@ --- Final Error When the Boom Comes to a Rest

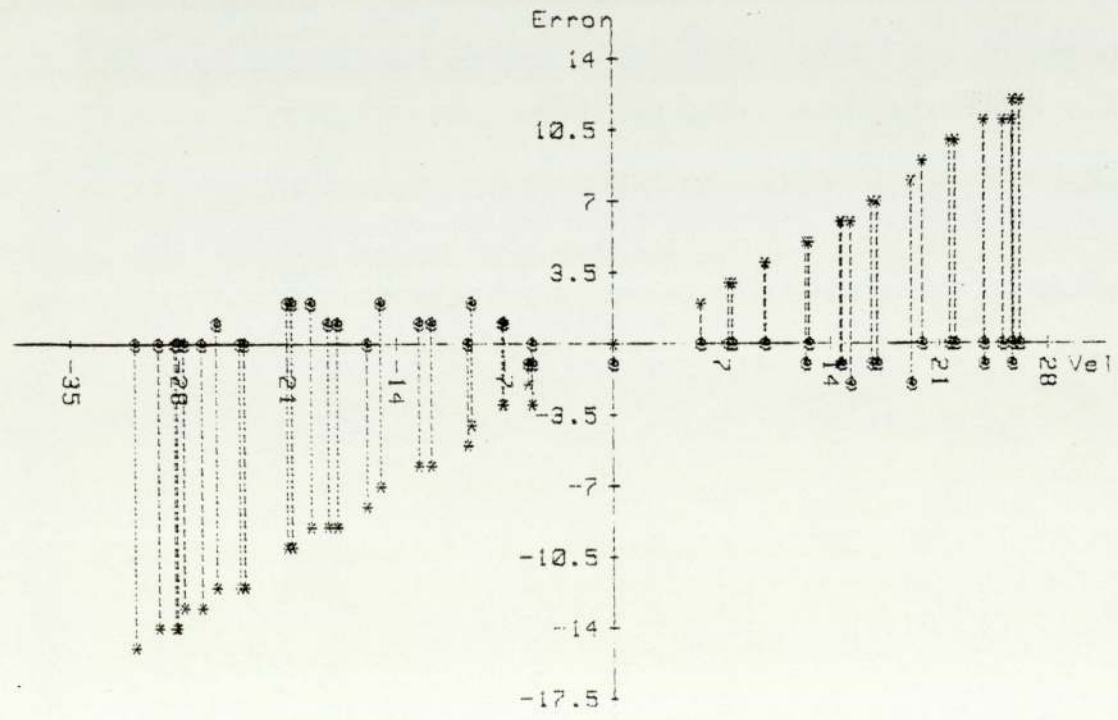


Fig 7.10 Error (Bits) - Boom Lifting Velocity (Bits/Sec)

* --- Initial Error When the Switching Signal is Sent
@ --- Final Error When the Boom Comes to a Rest

were sent to the valves, are presented in these graphs. It can be observed that the absolute final errors are never more than 2 bits. Since the ADC has an accuracy of ± 1 LSB, the worst error due to this ADC limitation is expected to be ± 2 LSB (± 1 LSB when the switching signal is determined coupled with ± 1 LSB when the final boom positional reading is taken). This confirms that the switching function can determine the switching point giving a final position within the expected accuracy of the system.

The switching functions have been programmed and tested for the boom positional control in the three activities (Move Boom, Trim Profile and Cut Cavity). Similar results to those obtained using the comparison method with slow speed limitation (Figures 7.3 and 7.4) have been obtained, therefore the recordings of these tests are not included. The only difference between these two methods is that the overall time taken for the boom to move by the comparison method (with slow speed limitation) is longer than the other one. This is because the band, in which only the slow speed mode is permitted, is larger when the comparison method is used (± 18H bits compared with ± 4 bits). This band is included in the switching function method to prevent the boom from hunting when it is initially very near the desired position.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

8.1. Conclusions

The test results in Chapter 7 show that the positional control of a boom ripper can be obtained. This involves the use of a dual microprocessor system, which provides a high operation speed. This is because the two microprocessors are running in parallel: one microprocessor system can concentrate on capturing and updating the system performance and the required data, while the other system can analyse these data and determine the control signals. Furthermore, the use of an APU will simplify the software and increase the speed in handling the mathematical operations.

Poppet valves have advantages over the conventional spool valves for this control application. Firstly, poppet valves are less sensitive to dirt and therefore are ideal for the mining environment. Secondly, poppet valves have two states: they can either be fully opened or closed. Therefore poppet valves can be operated directly on the 'On-Off' (or digital) signals sent by the microprocessor which is a digital device itself. Thirdly, poppet valves are easier and less expensive to manufacture. They also have less leakage for a fluid with a low viscosity compared with the conventional spool valves which have sliding seals. All these features make the poppet valves suitable for use in this mining boom ripper positional control system.

The optimal control theory shows that the response time can be minimised by maximising the acceleration and carefully

controlling the retardation. However, a 'bang-bang' control philosophy is adopted in this project instead of the normal way of controlling the valve closure. In this 'bang-bang' control strategy, simple On-Off signals are sent to the electro-hydraulically operated poppet valves. Two methods can be used in determining the control signals. They are both satisfactory in their performance in this application.

The simple comparison method is the simpler of the two. In this method, the control signals are determined by comparing the desired with the current boom positions: the valves will be opened if the two positions are not the same, but they will be closed when the two data match. In order to prevent any undesirable overshoot when the boom is moving at its high speed, it is necessary to restrict the speed of the boom to its slow mode when it is near the desired position. This region should be large enough to prevent overshooting when the boom is travelling at its highest speed. Because of this band, the response time is not a minimum when the boom changes its position at high speed.

The second method is to use the switching function evaluation in determining the signals sent to the control valves. Various switching functions have been derived in Chapter 3. J H Knight has studied the performance of the valves used in this control system, his results suggest that the valves can be approximated to On-Off valves with time delays. Because of the complexity in the system geometry and the

dynamics, it is extremely difficult to theoretically determine the coefficients in the switching function. So experimental evaluations have been used, which correlate favourably with the theory. These switching functions are then tested individually for each axis of movement (i.e. lifting and slewing). The results in Chapter 7 show that the use of this switching function can shorten the overall response time and hence an optimal control will be obtained when the boom is travelling at its high speed. It is worth noting that the data capturing system cannot detect a velocity when the boom moves slowly because of the limitation in the accuracy of the ADC and the capacity of the timer installed. Therefore, the evaluation of the switching function at the slow speed can effectively be reduced to a simple comparison.

8.2. Recommendations

The results in Chapter 7 show that the simple comparison method with speed limitation near the desired position can provide the necessary activities and adequate control. The simple comparison method yields a simpler and less expensive control system, which consumes less energy compared with a system using the switching function evaluation method. In this particular application, minimising the time, taken by the boom ripper to move from one position to another, is not very important. On the contrary, the intrinsic safety requirement demands a control circuit with less power consumption. Therefore, the simple comparison method with slow speed limitation near the desired position is recommended for this system.

A dual microprocessor system can provide a fast and continuous monitoring of the electro-hydraulic valves. The control signals sent to the valves can be obtained in a much shorter time, when the simple comparison method is used instead of the switching function evaluation method. This is because the function evaluation method requires more complicated mathematical operations, which need more time to handle, although the data manipulation time can be shortened by using an APU. However, the requirement of a wide region, in which only the slow speed mode is allowed for the boom to travel, makes the total time taken to fast move the cutter from one position to another longer in the simple comparison method. Therefore, if a fast response in moving the boom at its high speed is desired, the switching function evaluation method is recommended.

8.3. Suggestions for Further Work

A microprocessor based system can simplify and increase the facilities in the automatic operations for a boom ripper. Since the microprocessor is a digital device, it must operate on digital signals. This demands that all the analogue signals should be converted into the digital forms before they can be processed by the microprocessors. The accuracy of the system performance therefore depends highly upon the resolution power of the ADC installed. For the boom ripper under consideration, the accuracy in positioning the boom ripper at its cutter end, is approximately 1 cm. If a finer positional control is desired for the boom ripper, then an ADC with a higher

resolution power should be used, and the associated hardware and software have to be developed. For example, a 12-bit ADC with 1-bit accuracy can increase the resolution power from 1/255 (for an 8-bit ADC) to 1/4095.

A basic or the simplest control system has been developed, and extra facilities can be built around this system. One example is to construct a selector that can choose, from a variety of patterns, the method used to cut the cavity.

In the control system under consideration, no provisions have been made for the machine chassis movement. However, it will be easy to include the necessary correction, if the current machine position related to the tunnel frame is known, by considering the kinematics of the boom ripper. This can be done by converting all the positional data (desired and the current positions) into a common reference co-ordinate system and then determining the control signals with the transformed data. However, this will require a high amount of arithmetic operations, and the use of an APU will definitely facilitate these. Since the APU and its associated circuit board have been included in this study, the software development will not present a great difficulty. The main problem is to provide the information on the machine position and its orientation, therefore further work in developing a system to give these data is suggested.

The control system studied in this project has been designed for a mining boom ripper. This can be extended to provide control to the more sophisticated drilling boom machine by considering the machine geometry. Because the drilling machine has seven degrees of freedom, more mathematical operations will be required. Hardware and software development based on the APU is strongly recommended to provide a faster and simpler control system.

It has been observed that the hydraulic configuration of the boom ripper under consideration makes the fluid flow through the path with less resistance. In this project, an algorithm has been built into the system to give preference to the boom's vertical movement at high speed. Further work is suggested to develop a fluid flow divider so that the maximum available fluid power can be utilised.

REFERENCES

Abbreviations used:

AIEE	American Institution of Electrical Engineers.
ASME	American Society of Mechanical Engineers.
Appl.	Applied.
BHRA	British Hydrodynamics Research Association.
Conf.	Conference on.
Engng.	Engineering.
Hyd.	Hydraulic.
IChemE	Institution of Chemical Engineers.
IEE	Institution of Electrical Engineers.
IEEE	Institution of Electrical and Electronic Engineers.
IMechE	Institution of Mechanical Engineers.
IMinE	Institution of Mining Engineers.
Int.	International.
J.	Journal of.
JDSMC	Journal of Dynamic Systems, Measurement and Control.
Mech.	Mechanical or Mechanics.
MRDE	Mining Research and Development Establishment.
NCB	National Coal Board.
Proc.	Proceedings of.
Pt.	Part.
Sci.	Science.
Symp.	Symposium.
Trans.	Transactions of.

Year of publication is enclosed in parethesis ( ).

1. EZRA, Sir Derek

"Applying New Technology in a Traditional Industry - Coal Mining". Electronics and Power (1981), Vol.27, No.1, pp. 36-40.

2. BRIERLEY, F H P, PROBERT, M J, ROBINSON, A

"Electro-hydraulic Controls in Mining". Proc. Sixth Int. Fluid Power Symp. (1981), Paper G6, pp. 373-392, BHRA. Cambridge.

3. MORGAN, D

"In Support of Longwalling". Colliery Guardian (1980), Vol. 228, No. 7, pp. 28-33.

4. KELLY, S M

"MINOS : Overview". Computer Systems Paper No. 30, MRDE, NCB, July 1980.

5. WORT, K G

"Tunnelling Machinery for Coal Mines in the 1980's". Colliery Guardian (1982), Vol. 230, No. 3, pp. 120-124.

6. RODFORD, I G

"Experience with Impact Units". Proc. Conf. Fluid Power Equipment in Mining Quarrying and Tunnelling (1974), pp. 57-66. IMechE and IMinE, London.

7. MORIS, A H

"Boom Ripping". Proc. Conf. Fluid Power Equipment in Mining Quarrying and Tunnelling (1974), pp. 57-66. IMechE and IMinE, London.

8. MORIS, A H

"Roadheaders - Tough Cutting". Colliery Guardian (1982). Vol. 230, No. 3, pp. 116-120.

9. BUTLER, R

"A Theoretical Analysis of the Response of a Loaded Hydraulic Relay". Proc. IMechE (1959), Vol. 173, pp. 429-455.

10. CONWAY, H G and COLLINSON, E G

"An Introduction to Hydraulic Servomechanism Theory". Proc. Conf. Hyd. Servomechanism (1953), pp. 1-11. IMechE, London.

11. HARPUR, N F

"Some Design Considerations of Hydraulic Servos of Jack Type". Proc. Conf. Hyd. Servomechanism (1953), pp. 41-50. IMechE, London.

12. LEE, S Y and BLACKBURN, J F

"Contributions to Hydraulic Control (Pt. 1) Steady-state Axial Forces on Control Valve Pistons, (Pt. 2) Transient Flow Forces and Valve Instability". Trans. ASME (1952), Vol. 74, pp. 1005-1016.

13. TAFT, C K and TWILL, J P

"An Analysis of the Three-way Underlapped Hydraulic Spool Servo-valve". JDSMC, Trans. ASME (1978), Vol. 100, Series G, pp. 117-123.

14. NOTTON, G J and TURNBULL, D E

"Some Factors Influencing the Stability of Piston-type Control Valves". Proc. IMechE (1958), Vol. 172, pp. 1065-1081.

15. FOSTER, K and KULKARNI, M M

"Steady-amplitude, Self-excited Oscillations of Hydraulic Spool Valves". J. Mech. Engng. Sci. (1968), Vol. 10, pp. 306-318.

16. BURTON, R T, UKRAINETZ, P R and NIKIFORUK, P N

"Analytical Prediction of Self-sustained Oscillations in an Inertially Loaded Hydraulic Control Valve". Proc. Third Int. Fluid Power Symp. (1973), Paper E4, pp. E4.49-E4.60. BHRA, Turin.

17. STONE, J A

"Discharge Coefficients and Steady State Flow Forces for Hydraulic Poppet Valves". Trans. ASME (1960), Vol. 82, Series D, pp. 144-154.

18. McCLOY, D and MARTIN, H R

"The Control of Fluid Power". Longman (1973).

19. MANSFELD, G

"Fast Switching Ball Valves as Digital Control Elements for an Electro-hydraulic Servo Actuator". Proc. Sixth Int. Fluid Power Symp. (1981), pp. 335-348. BHRA, Cambridge.

20. TURNBULL, D E

"The Response of a Loaded Hydraulic Servomechanism". Proc. IMechE (1959), Vol. 173, pp. 16-30.

21. McCLOY, D

"Graphical Analysis of the Step Response of a Hydraulic Servo-mechanism". Proc. IMechE (1967-68), Vol. 182, Pt. 1, pp. 243-250.

22. COOK, A E and HEAPS, H S

"Series Analysis of a Closed-loop System Containing a Loaded Hydraulic Relay". Proc. IMechE (1964-65), Vol. 179, Pt. 1, pp. 847-855.

23. ROYLE, J K

"Inherent Non-linear Effects in Hydraulic Control Systems with Inertia Loading". Proc. IMechE (1959), Vol. 173, pp. 257-269.

24. LAMBERT, T H and DAVIES, R M

"Investigation of the Response of an Hydraulic Servomechanism with Inertial Load". J. Mech. Engng. Sci. (1963), Vol. 5, pp. 281-289.

25. LAMBERT, T H and DAVIES, R M

"Transient Response of an Hydraulic Servomechanism Flexibly Inertial Load". J. Mech. Engng. Sci. (1964), Vol. 6, pp. 32-37.

26. WANG, P K C

"Mathematical Models for Time-Domain Design of Electrohydraulic Servomechanisms". Trans. AIEE (1961), Vol. 80, pp. 252-260.

27. MARTIN, K F

"Stability and Step Response of a Hydraulic Servo with Special Reference to Unsymmetrical Oil Volume Conditions". J. Mech. Engng. Sci. (1970), Vol. 12, pp. 331-338.

28. MONTGOMERY, J and LICHTAROWICZ, A

"Asymmetrical Lap and Other Non-linearities in Valve-controlled Hydraulic Actuators". Proc. IMechE (1968-69), Vol. 183, Pt. 1, pp. 663-675.

29. WANG, P K C and MA, J T S

"Cavitation in Valve-controlled Hydraulic Actuators". Trans. ASME (1963), J. App. Mech., Vol. 85, pp. 537-546.

30. McCLOY, D

"Cavity Formation in Valve Controlled Hydraulic Cylinder". Proc. IMechE (1969-70), Vol. 184, Pt. 1, pp. 969-977.

31. McCLOY, D

"Cavity Effects in On-off Controlled Hydraulic Servos". Trans. ASME (1972), JDSMC, Series G, pp. 50-56.

32. MARTIN, H A and McCLOY, D

"Pressure Transients Generated During the Rapid Braking of Asymmetric Hydraulic Actuators". J. Mech. Engng. Sci. (1979), Vol. 21, pp. 93-103.

33. OLDENBURGER, R

"Optimum Non-linear Control". Trans. ASME (1957), Vol. 79, pp. 527-546.

34. OLDENBURGER, R

"Optimal Control". Holt, Rinehart and Winston Inc., (1966).

35. TEREN, F

"Solution of Transient Optimisation Problems by Using an Algorithm Based on Non-linear Programming". Proc. Joint Automatic Control Conference (1977), Vol. 2, pp. 1549-1560, IEEE, San Francisco.

36. WANG, P K C

"Analytical Design of Electro-hydraulic Servomechanisms with Near Time-optimal Response". IEEE Transaction on Automatic Control (1963), Vol. AC-8, pp. 15-27.

37. DAVIES, R M

"Analytical Design for Time Optimum Transient Response of Hydraulic Servomechanisms". J. Mech. Engng. Sci. (1965), Vol. 7, pp. 8-14.

38. NIKIFORUK, P N, WILSON, J N and LEPP, E M

"Transient Responses of a Time-optimised Hydraulic Servo Operating Under Cavitation Conditions". Proc. IMechE (1970-71), Vol. 185, pp. 837-846.

39. ARMSTRONG, P J and McCLOY, D

"Optimisation of Hydraulic Servo Using On-off Controllers". Proc. Second Fluid Power Symp. (1971), Paper B4, pp. B4.69-B4.84. BHRA, Guildford.

40. ARMSTRONG, P J and McCLOY, D

"Design of High Speed Hydraulic Position Servomechanisms Using Optimisation Techniques". Proc. IMechE (1973), Vol. 187, pp. 775-786.

41. FAZARINC, Z

"Minicomputer as a Circuit-Design Tool". Simulation (1972), Vol. 18, No. 2, pp. 47-56.

42. INTEL CORPORATION

"Component Data Catalogue". January (1981).

43. INTEL CORPORATION

"8080/85 Floating-Point Arithmetic Library User's Manual". (1979).

44. INTEL CORPORATION

"MCS - 80/85 Family User's Manual". October (1979).

45. CHUEN, C W

"Computer Aided Dynamic Design of Hydraulic System Components". MSc Dissertation, Dept of Mechanical Engineering, The University of Aston in Birmingham, October (1978).

46. CLENSHAW, C W

"Chebyshev Series for Mathematical Functions". Mathematical Tables, Vol. 5, HMSO (1962).

APPENDICES

APPENDIX A

Derivation of Switching Function for ON-OFF Valves with Time Delay

After the application of switching signal but before the actual operation of valve, the governing equation is given by

$$\frac{d^3y}{dt^3} + (2\zeta + \nu\xi)\frac{d^2y}{dt^2} + (1 + 2\nu\zeta\xi)\frac{dy}{dt} = \xi \qquad (3.31)$$

Integrating with respect to t gives,

$$\frac{d^2y}{dt^2} + (2\zeta + \nu\xi)\frac{dy}{dt} + (1 + 2\nu\zeta\xi)y = \xi t + R_1$$

$$\text{where } R_1 = \ddot{y}_o + (2\zeta + \nu\xi)\dot{y}_o + (1 + 2\nu\zeta\xi)y_o$$

Applying Laplace transform to both sides of this equation gives,

$$s^2\mathcal{L}y - s\dot{y}_o - y_o + (2\zeta + \nu\xi)(s\mathcal{L}y - y_o) + (1 + 2\nu\zeta\xi)\mathcal{L}y = \frac{\xi}{s^2} + \frac{R_1}{s}$$

Rearranging give,

$$y = \frac{1}{s^2+(2\zeta+\nu\xi)s+(1+2\nu\zeta\xi)}\left[\frac{\xi}{s^2} + \frac{R_1}{s} + sy_o + \dot{y}_o + (2\zeta+\nu\xi)y_o\right]$$

$$= \frac{1}{s^2+(2\zeta+\nu\xi)s+(1+2\nu\zeta\xi)}\left[\frac{\xi}{s^2} + \frac{\ddot{y}_o}{s} + \frac{2\zeta+\nu\xi}{s}\dot{y}_o + \dot{y}_o\right] + \frac{y_o}{s}$$

Let $(s - \phi_1)(s - \phi_2) = s^2 + (2\zeta + \nu\xi)s + (1 + 2\nu\zeta\xi)$

$$= s^2 - (\phi_1 + \phi_2)s + \phi_1\phi_2$$

$$\therefore\ y = \frac{1}{\phi_1-\phi_2}\left[\frac{1}{s-\phi_1} - \frac{1}{s-\phi_2}\right]\left[\frac{\xi}{s^2} + \frac{\ddot{y}_o}{s} - \frac{\phi_1+\phi_2}{s}\dot{y}_o + \dot{y}_o\right] + \frac{y_o}{s}$$

Applying inverse Laplace transform to this equation gives,

$$y(t) = \frac{\xi t}{\phi_1\phi_2} + \frac{\xi}{\phi_1-\phi_2}\left(\frac{e^{\phi_1 t}}{\phi_1^2} - \frac{e^{\phi_2 t}}{\phi_2^2}\right) + \frac{\phi_1+\phi_2}{\phi_1^2\phi_2^2}\xi + y_o$$

$$+ \frac{\dot{y}_o}{\phi_1-\phi_2}\left[\frac{\phi_1}{\phi_2}e^{\phi_2 t} - \frac{\phi_2}{\phi_1}e^{\phi_1 t} - \frac{(\phi_1^2-\phi_2^2)}{\phi_1\phi_2}\right]$$

$$+ \frac{\ddot{y}_o}{\phi_1-\phi_2}\left[\frac{e^{\phi_1 t}}{\phi_1} - \frac{e^{\phi_2 t}}{\phi_2} + \frac{\phi_1-\phi_2}{\phi_1\phi_2}\right]$$

Expression of $\dot{y}(t)$ can be obtained by differentiating $y(t)$ with respect to t, the non-dimensional time. Hence, $\dot{y}(t) =$

$$\frac{dy}{dt} = \frac{\xi}{\phi_1\phi_2} + \frac{\xi}{\phi_1\ \phi_2}\left(\frac{e^{\phi_1 t}}{\phi_1} - \frac{e^{\phi_2 t}}{\phi_2}\right) + \frac{\dot{y}_o}{\phi_1\ \phi_2}\left[\phi_1 e^{\phi_2 t} - \phi_2 e^{\phi_1 t}\right]$$

$$+ \frac{\ddot{y}_o}{\phi_1-\phi_2}\left[e^{\phi_1 t} - e^{\phi_2 t}\right]$$

Differentiating $\frac{dy}{dt}$ with respect to t gives the expression for $\ddot{y}(t)$,

$$\ddot{y}(t) = \frac{d^2y}{dt^2} = \frac{\xi}{\phi_1-\phi_2}\left(e^{\phi_1 t} - e^{\phi_2 t}\right) + \frac{\phi_1\phi_2\dot{y}_o}{\phi_1-\phi_2}\left(e^{\phi_2 t} - e^{\phi_1 t}\right)$$

$$+ \frac{\ddot{y}_o}{\phi_1\ \phi_2}\left(\phi_1 e^{\phi_1 t} - e^{\phi_2 t}\right)$$

After the time delay, t_d, the system behaviour will be described by the same set of differential equations as the ideal ON-OFF case. Therefore, for the ram to rest at the desired position, y_d, the valves have to be switched so that at t_d after the application of switching (closing) signal the following will be satisfied:

$$y_d = \ddot{y}_{t_d} + 2\zeta\dot{y}_{t_d} + y_{t_d}$$

Therefore the switching function is:

$$y_d = \frac{\ddot{y}_o}{\phi_1 - \phi_2}\left(\phi_1 + 2\zeta + \frac{1}{\phi_1}\right)e^{\phi_1 t_d} - \left(\phi_2 + 2\zeta + \frac{1}{\phi_2}\right)e^{\phi_2 t_d} + \frac{\phi_1 - \phi_2}{\phi_1\phi_2}$$

$$+ \frac{\dot{y}_o}{\phi_1 - \phi_2}\left[\phi_1\left(\phi_2 + 2\zeta + \frac{1}{\phi_2}\right)e^{\phi_2 t_d} - \phi_2\left(\phi_1 + 2\zeta + \frac{1}{\phi_1}\right)e^{\phi_1 t_d} - \frac{\phi_1^2 - \phi_2^2}{\phi_1\phi_2}\right]$$

$$+ \frac{\xi}{\phi_1\phi_2}\left(2\zeta + t_d + \frac{\phi_1 + \phi_2}{\phi_1\phi_2}\right) + y_o$$

$$+ \frac{\xi}{\phi_1 - \phi_2}\left[\frac{1}{\phi_1}\left(\phi_1 + 2\zeta + \frac{1}{\phi_1}\right)e^{\phi_1 t_d} - \frac{1}{\phi_2}\left(\phi_2 + 2\zeta + \frac{1}{\phi_2}\right)e^{\phi_2 t_d}\right]$$

APPENDIX B

Derivation of Switching Function for Linearly Operated Valves with Operating Time t_c

After the application of switching signal (closing signal) but before the complete closure of valves, the governing equation is given by:

$$\frac{d^3y}{dt^3} + [2\zeta + \nu\xi(1 - \frac{t}{t_c})]\frac{d^2y}{dt^2} + [1 + 2\nu\zeta\xi(1 - \frac{t}{t_c})]\frac{dy}{dt} = \xi(1 - \frac{t}{t_c}) \quad \ldots \quad (3.32)$$

Applying Laplace transform to both sides of this equation gives,

$$s^3\mathcal{L}y - s^2y_o - s\dot{y}_o - \ddot{y}_o + (2\zeta+\nu\xi)(s^2\mathcal{L}y - sy_o - \dot{y}_o) + \frac{\nu\xi}{t_c}(2s\mathcal{L}y - y_o)$$
$$+ (1 + 2\nu\zeta\xi)(s\mathcal{L}y - y_o) + \frac{2\nu\zeta\xi}{t_c}\mathcal{L}y = \xi(\frac{1}{s} - \frac{1}{t_c s^2})$$

Rearranging gives,

$$y = \frac{1}{s^3 + (2\zeta+\nu\xi)s^2 + (1 + 2\nu\zeta\xi + \frac{2\nu\xi}{t_c})s + \frac{2\zeta\xi\nu}{t_c}} *$$

$$\left\{ \begin{array}{l} \ddot{y}_o + (s + 2\zeta + \nu\xi)\dot{y}_o + [s^2 + (2\zeta + \nu\xi)s + 1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c}]y_o \\ + \xi(\frac{1}{s} - \frac{1}{t_c s^2}) \end{array} \right\}$$

Taking the inverse Laplace transform gives the expression for $y(t)$,

$$y(t) = \frac{1}{(\Theta_1 - \Theta_2)(\Theta_2 - \Theta_3)(\Theta_3 - \Theta_1)} *$$

$$\left[\begin{array}{l} y_o \left[(\Theta_3 - \Theta_2) e^{\Theta_1 t} + (\Theta_1 - \Theta_3) e^{\Theta_2 t} + (\Theta_2 - \Theta_1) e^{\Theta_3 t} \right] \\ \\ + \dot{y}_o \left[\begin{array}{l} (2\zeta + \nu\xi + \Theta_1)(\Theta_3 - \Theta_2) e^{\Theta_1 t} + (2\zeta + \nu\xi + \Theta_2)(\Theta_1 - \Theta_3) e^{\Theta_2 t} \\ + (2\zeta + \nu\xi + \Theta_3)(\Theta_2 - \Theta_1) e^{\Theta_3 t} \end{array} \right] \\ \\ + y_o \left[\begin{array}{l} \{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta + \nu\xi)\Theta_1 + \Theta_1^2\}(\Theta_3 - \Theta_2) e^{\Theta_1 t} \\ + \{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta + \nu\xi)\Theta_2 + \Theta_2^2\}(\Theta_1 - \Theta_3) e^{\Theta_2 t} \\ + \{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta + \nu\xi)\Theta_3 + \Theta_3^2\}(\Theta_2 - \Theta_1) e^{\Theta_3 t} \end{array} \right] \\ \\ + \xi \left[\begin{array}{l} (1 - \frac{1}{t_c \Theta_1}) \frac{\Theta_3 - \Theta_2}{\Theta_1} e^{\Theta_1 t} + (1 - \frac{1}{t_c \Theta_2}) \frac{\Theta_1 - \Theta_3}{\Theta_2} e^{\Theta_2 t} \\ + (1 - \frac{1}{t_c \Theta_3}) \frac{\Theta_2 - \Theta_1}{\Theta_3} e^{\Theta_3 t} - (1 - \frac{t}{t_c})(\frac{\Theta_3 - \Theta_2}{\Theta_1} + \frac{\Theta_1 - \Theta_3}{\Theta_2} + \frac{\Theta_2 - \Theta_1}{\Theta_3}) \\ + \frac{1}{t_c}(\frac{\Theta_3 - \Theta_2}{\Theta_1^2} + \frac{\Theta_1 - \Theta_3}{\Theta_2^2} + \frac{\Theta_2 - \Theta_1}{\Theta_3^2}) \end{array} \right] \end{array} \right]$$

$$\therefore \dot{y}(t) = \frac{dy}{dt} = \frac{1}{(\theta_1 - \theta_2)(\theta_2 - \theta_3)(\theta_3 - \theta_1)} *$$

$$\left[\begin{array}{l} \ddot{y}_o \left[\theta_1(\theta_3 - \theta_2)e^{\theta_1 t} + \theta_2(\theta_1 - \theta_3)e^{\theta_2 t} + \theta_3(\theta_2 - \theta_1)e^{\theta_3 t} \right] \\ + \dot{y}_o \left[\begin{array}{l} \theta_1(2\zeta+\nu\xi+\theta_1)(\theta_3-\theta_2)e^{\theta_1 t} + \theta_2(2\zeta+\nu\xi+\theta_2)(\theta_1-\theta_3)e^{\theta_2 t} \\ + \theta_3(2\zeta+\nu\xi+\theta_3)(\theta_2-\theta_1)e^{\theta_3 t} \end{array} \right] \\ + y_o \left[\begin{array}{l} \theta_1\{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta+\nu\xi)\theta_1 + \theta_1^2\}(\theta_3 - \theta_2)e^{\theta_1 t} \\ + \theta_2\{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta+\nu\xi)\theta_2 + \theta_2^2\}(\theta_1 - \theta_3)e^{\theta_2 t} \\ + \theta_3\{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta+\nu\xi)\theta_3 + \theta_3^2\}(\theta_2 - \theta_1)e^{\theta_3 t} \end{array} \right] \\ + \xi \left[\begin{array}{l} (1 - \frac{1}{t_c\theta_1})(\theta_3 - \theta_2)e^{\theta_1 t} + (1 - \frac{1}{t_c\theta_2})(\theta_1 - \theta_3)e^{\theta_2 t} \\ + (1 - \frac{1}{t_c\theta_3})(\theta_2 - \theta_1)e^{\theta_3 t} + \frac{1}{t_c}\left(\frac{\theta_3 - \theta_2}{\theta_1} + \frac{\theta_1 - \theta_3}{\theta_2} + \frac{\theta_2 - \theta_1}{\theta_3}\right) \end{array} \right] \end{array} \right]$$

and $$\ddot{y}(t) = \frac{d^2y}{dt^2} = \frac{1}{(\theta_1 - \theta_2)(\theta_2 - \theta_3)(\theta_3 - \theta_1)} *$$

$$\left[\begin{array}{l} \ddot{y}_o \left[\theta_1^2(\theta_3 - \theta_2)e^{\theta_1 t} + \theta_2^2(\theta_1 - \theta_3)e^{\theta_2 t} + \theta_3^2(\theta_2 - \theta_1)e^{\theta_3 t} \right] \\ + \dot{y}_o \left[\begin{array}{l} \theta_1^2(2\zeta+\nu\xi+\theta_1)(\theta_3 - \theta_2)e^{\theta_1 t} + \theta_2^2(2\zeta+\nu\xi+\theta_2)(\theta_1 - \theta_3)e^{\theta_2 t} \\ + \theta_3^2(2\zeta + \nu\xi + \theta_3)(\theta_2 - \theta_1)e^{\theta_3 t} \end{array} \right] \\ + y_o \left[\begin{array}{l} \theta_1^2\{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta+\nu\xi)\theta_1 + \theta_1^2\}(\theta_3 - \theta_2)e^{\theta_1 t} \\ + \theta_2^2\{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta+\nu\xi)\theta_2 + \theta_2^2\}(\theta_1 - \theta_3)e^{\theta_2 t} \\ + \theta_3^2\{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta+\nu\xi)\theta_3 + \theta_3^2\}(\theta_2 - \theta_1)e^{\theta_3 t} \end{array} \right] \\ + \xi \left[\begin{array}{l} \theta_1(1 - \frac{1}{t_c\theta_1})(\theta_3 - \theta_2)e^{\theta_1 t} + \theta_2(1 - \frac{1}{t_c\theta_2})(\theta_1 - \theta_3)e^{\theta_2 t} \\ + \theta_3(1 - \frac{1}{t_c\theta_3})(\theta_2 - \theta_1)e^{\theta_3 t} \end{array} \right] \end{array} \right]$$

Similarly the switching function can be obtained by satisfying the following requirement when the valves are completely closed, i.e. at time, t_c, after the application of switching signal:

$$y_d = \ddot{y}_{t_c} + 2\zeta\dot{y}_{t_c} + y_{t_c}$$

Therefore, the switching function is:

$$y_d = \frac{1}{(\theta_1 - \theta_2)(\theta_2 - \theta_3)(\theta_3 - \theta_1)} *$$

$$\Bigg[\ddot{y}_o \Big[(1 + 2\zeta\theta_1 + \theta_1^2)(\theta_3 - \theta_2)e^{\theta_1 t_c} + (1 + 2\zeta\theta_2 + \theta_2^2)(\theta_1 - \theta_3)e^{\theta_2 t_c} + (1 + 2\zeta\theta_3 + \theta_3^2)(\theta_2 - \theta_1)e^{\theta_3 t_c} \Big]$$

$$+ \dot{y}_o \Big[(1 + 2\zeta\theta_1 + \theta_1^2)(2\zeta + \nu\xi + \theta_1)(\theta_3 - \theta_2)e^{\theta_1 t_c} + (1 + 2\zeta\theta_2 + \theta_2^2)(2\zeta + \nu\xi + \theta_2)(\theta_1 - \theta_3)e^{\theta_2 t_c} + (1 + 2\zeta\theta_3 + \theta_3^2)(2\zeta + \nu\xi + \theta_3)(\theta_2 - \theta_1)e^{\theta_3 t_c} \Big]$$

$$+ y_o \Big[(1 + 2\zeta\theta_1 + \theta_1^2)\{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta + \nu\xi)\theta_1 + \theta_1^2\}(\theta_3 - \theta_2)e^{\theta_1 t_c} + (1 + 2\zeta\theta_2 + \theta_2^2)\{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta + \nu\xi)\theta_2 + \theta_2^2\}(\theta_1 - \theta_3)e^{\theta_2 t_c} + (1 + 2\zeta\theta_3 + \theta_3^2)\{1 + 2\nu\zeta\xi + \frac{\nu\xi}{t_c} + (2\zeta + \nu\xi)\theta_3 + \theta_3^2\}(\theta_2 - \theta_1)e^{\theta_3 t_c} \Big]$$

$$+ \xi \Big[(1 + 2\zeta\theta_1 + \theta_1^2)(1 - \frac{1}{t_c\theta_1})\frac{\theta_3 - \theta_2}{\theta_1}e^{\theta_1 t_c} + (1 + 2\zeta\theta_2 + \theta_2^2)(1 - \frac{1}{t_c\theta_2})\frac{\theta_1 - \theta_3}{\theta_2}e^{\theta_2 t_c} + (1 + 2\zeta\theta_3 + \theta_3^2)(1 - \frac{1}{t_c\theta_3})\frac{\theta_2 - \theta_1}{\theta_3}e^{\theta_3 t_c}$$

$$+ \frac{1}{t_c}\left(\frac{\theta_3 - \theta_2}{\theta_1^2} + \frac{\theta_1 - \theta_3}{\theta_2^2} + \frac{\theta_2 - \theta_1}{\theta_3^2}\right) - \left(1 - \frac{t}{t_c} - \frac{2\zeta}{t_c}\right)\left(\frac{\theta_3 - \theta_2}{\theta_1} + \frac{\theta_1 - \theta_3}{\theta_2} + \frac{\theta_2 - \theta_1}{\theta_3}\right) \Big] \Bigg]$$

APPENDIX C

Computer Programs for the Simulation Studies

The simulation studies were carried out using the modified Euler method [41] to solve the differential equations. This method has a long term stability which is desirable for continuous simulations over an extended period. It evaluates the forward states using the derivatives which are currently updated. The block structure of this program is shown in Figure C.1. The system data have been written onto a file before this program is executed. When this program is executed, the system data are first read and modified if necessary. Then the coefficients of the function will be evaluated according to the appropriate equations (Sections 3.6 and 3.7). The roots of the polynominal are found by the subprogram "Roots" which was described by Chuen[45]. The initial states and the desired position are then entered into the computer. The program will then evaluate the switching function and determine the switching signal. The system behaviours are then evaluated by the modified Euler method and stored if necessary for a small step increment of time. If the simulation is to be continued, then it will repeat from the switching function evaluation; otherwise it will modify the system parameters and repeat the whole simulation process if required.

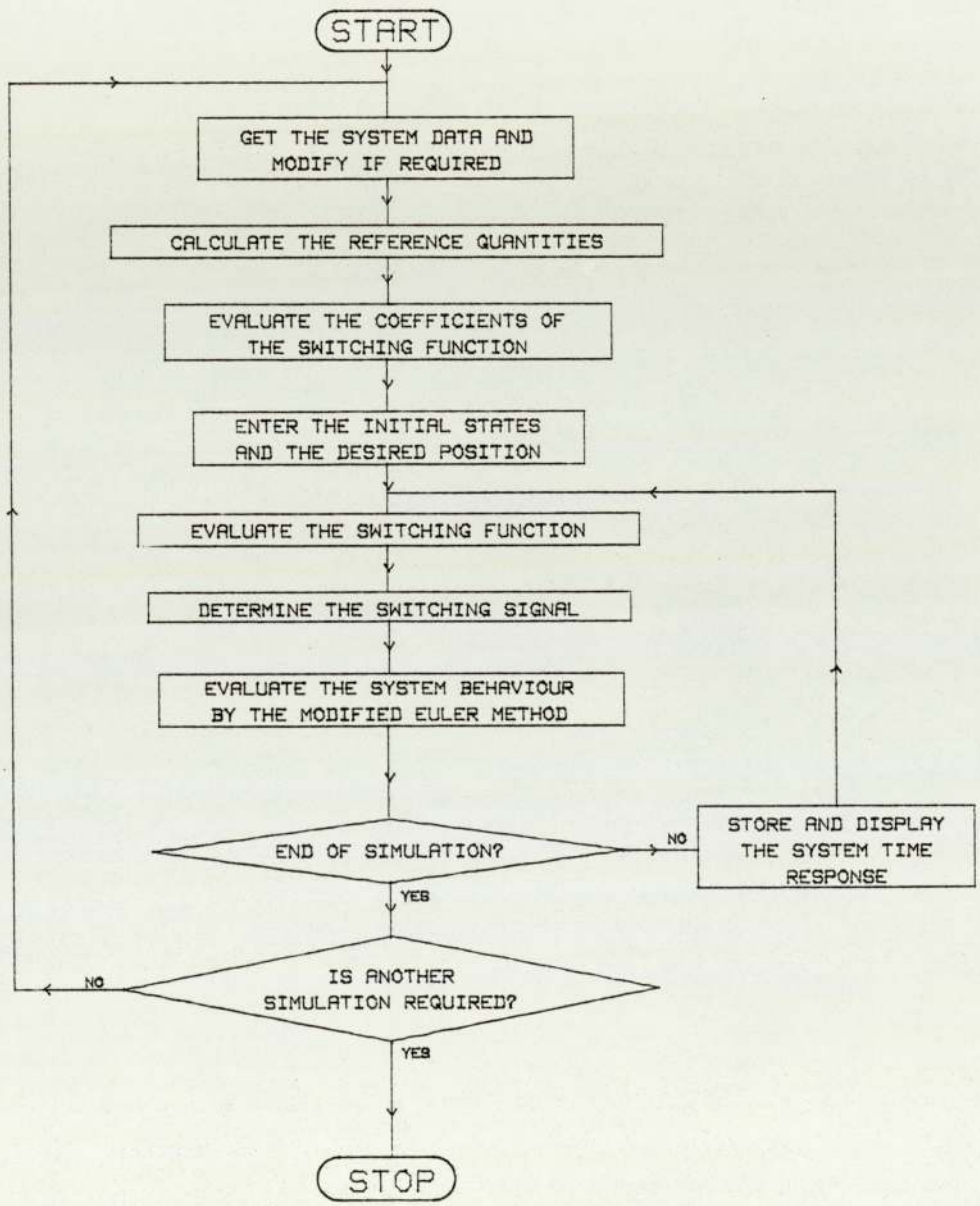


FIG C.1 BLOCK STRUCTURE OF THE SIMULATION PROGRAM

```
20    OPTION BASE 1
30    COM Thetaf,Thetai,Tlimit,Veli,Zmin,Zmax,Ep(500),Z(500)
40    COM Tauc(500),Fo,Zeta,Damping,P1i,P2i,Wtval,Change$[3]
50    COM Deltai,Deltaf,Sety,Tdummyc,M,Tauclose,Type,Sym,Aratio
60    COM INTEGER Rerun,Npoints,Again,Plotter,Dump,No,Printer,
      Pswitch(20),Psws
70    DIM Coefficient(4),Solution(3,2),Result(2),Answer(2)
80    DIM Operand(2),Operand1(2),Operand2(2),Tempres1(2)
90    DIM Tempres2(2),Coeffdum(4),Ramp1(500),Ramp2(500)
100   INTEGER Divide,Multiply,Subtract,Add,Rtorder,Rtorder1
110   SHORT Shortdummy1,Shortdummy2,Shortdummy3
120   PRINTER IS 16
130   PRINT PAGE,"SELECT MASS STORAGE DEVICE WITH THESE CODES:"
140   PRINT
150   PRINT SPA(20),"1 - for Flexible Disk"
160   PRINT
170   PRINT SPA(20),"2 - for Right Cartridge marked T15"
180   PRINT
190   PRINT SPA(20),"3 - for Left Cartridge marked T14"
200   INPUT Selectcode
210   IF (Selectcode>0) AND (Selectcode<4) THEN 240
220   BEEP
230   GOTO 200
240   IF Selectcode=1 THEN Msus$=":F8"
250   IF Selectcode=2 THEN Msus$=":T15"
260   IF Selectcode=2 THEN Msus$=":T14"
270   MASS STORAGE IS Msus$
280   PRINT PAGE
290   READ Divide,Multiply,Subtract,Add
300   DATA 4,3,2,1
310   ASSIGN #5 TO "SOLN"&Msus$,Check
320   IF Check=0 THEN 390
330   IF Check=1 THEN 360
340   DISP "FILE SOLN ERROR:";
350   GOTO 630
360   DISP "CREATE FILE: SOLN"
370   CREATE "SOLN",5
380   GOTO 310
390   BUFFER #5
400   Deltam=Deltas=0
410   Again=No=Deltafo=0
420   ! ****** RE-ENTER AT THIS POINT FOR RE-RUN ****** !
430   Npoints=Pmin=Pmax=Psws=0
440   Zmin=Wmin=Amin=9E99
450   Zmax=Wmax=Amax=-9E99
460   ! ****** INITIALIZATION OF SYSTEM PARAMETERS **** !
470   Printer=16
480   IF Again THEN 560
490   INPUT "PRINTER DEVICE CODE: 0 for Thermal Printer  16
      for CRT",Printer
500   INPUT "DO YOU WANT TO SPECIFY DURATION OF SIMULATION ",A$
510   Tlimit=9E99
520   Rerun=1
530   IF POS(UPC$(A$),"N") THEN 560
540   INPUT "NON-DIMENSIONAL TIME DURATION",Tlimit
550   Tdummyc=Tlimit
560   Sym=Onoff=Epsilo=Tau=Signal=Print=Sw=0
570   Dratio=Aratio=1
580   Va=Vb=.5
590   ASSIGN #1 TO "SPDATA"&Msus$,Check
600   IF Check=0 THEN 680
610   DISP "FILE SPDATA ERROR: ";
620   IF Check=1 THEN 660
630   DISP "WRONG FILE TYPE, OR FILE IS PROTECTED"
640   BEEP
```

```
650   STOP
660   DISP "FILE IS MISSSING"
670   GOTO 640
680   BUFFER #1
690 Readfile: ! **** READING DATA FROM FILE "SPDATA" **** !
700   READ #1;Sym,Onoff,Time,Cd1,B1,Xs,Ps,Pe,Rho,Beta,Ys
710   READ #1;D1,A1,Ratio,Vr,Cl,M,Bd,Ko,Fo
720   Cd2=Cd1
730   D2=D1
740   A2=A1
750   B2=B1
760   IF Sym THEN 780
770   READ #1;Cd2,B2,Dratio,Aratio,Va
780   READ #1;Zeta
790   READ #1,1    ! MOVE THE FILE POINT TO THE FIRST RECORD
800   Thetas=Thetas+Deltas  ! ADJUST Theta values on Re-run
810   Thetaf=Thetas
820   IF NOT Again THEN 850
830   A$=Change$
840   GOTO 950
850   PRINT PAGE,"Sym = ";Sym;"Onoff = ";Onoff;"Time = ";Time;"B1
       = ";B1;"Xs = ";Xs
860   PRINT "Cd1 = ";Cd1;"Ps = ";Ps;"Pe = ";Pe;"Rho = ";Rho;"Beta
       = ";Beta
870   PRINT "Ys = ";Ys;"D1 = ";D1;"Ratio = ";Ratio;"Vr = ";Vr
880   PRINT "Cl = ";Cl;"M = ";M;"Bd = ";Bd;"Ko = ";Ko;"Fo = ";Fo
890   PRINT "Zeta = ";Zeta;"Cd2 = ";Cd2;"B2 = ";B2
900   PRINT "Dratio = ";Dratio;"Va = ";Va
910   PRINT LIN(5),"Note that if Zeta was specified then Bd would
       be calculated automatically"
920   IF Again THEN 970
930   Change$="NO"
940   INPUT "ANY CHANGE IN THE ABOVE QUANTITIES? (YES/NO)",A$
950   IF POS(UPC$(A$),"N") THEN 1090
960   IF Again THEN 850
970   DISP "MAKE THE CHANGES THEN PRESS <CONT>"
980   PAUSE
990   Aratio=Dratio*Dratio
1000  A1=PI*D1*D1/4E4           ! D1 IS IN cm
1010                            ! PRINT NEW DATA INTO FILE
1020  PRINT #1;Sym,Onoff,Time,Cd1,B1,Xs,Ps,Pe,Rho,Beta,Ys
1030  PRINT #1;D1,A1,Ratio,Vr,Cl,M,Bd,Ko,Fo
1040  IF Sym THEN 1060
1050  PRINT #1;Cd2,B2,Dratio,Aratio,Va
1060  PRINT #1;Zeta,END
1070  PRINT LIN(2),"Will there be ANY MORE CHANGES when rerun
       with different ",LIN(1),"initial & final conditions?"
1080  INPUT Change$
1090  D2=Dratio*D1
1100  Type=Onoff
1110  Aratio=Dratio*Dratio
1120  A2=Aratio*A1
1130  ON KEY #4 GOTO Endcal
1140  ON KEY #2 GOSUB Changeprinter
1150  ASSIGN * TO #1
1160  PRINTER IS Printer
1170  IF Sym THEN 1240
1180  PRINT "SYSTEM PARAMETERS",LIN(2),"SPOOL:"
1190  PRINT "    DISCHARGE COEFFICIENT FOR SPOOL VALVE 1:";Cd1
1200  PRINT "    DISCHARGE COEFFICIENT FOR SPOOL VALVE 2:";Cd2
1210  PRINT "                        WIDTH OF SPOOL VALVE 1:";B1;"mm."
1220  PRINT "                        WIDTH OF SPOOL VALVE 2:";B2;"mm."
1230  GOTO 1270
1240  PRINT "SYSTEM PARAMETERS (SYMMETRIC)",LIN(2),"SPOOL:"
1250  PRINT "      DISCHARGE COEFFICIENT FOR SPOOL VALVE:";Cd1
1260  PRINT SPA(22);"WIDTH OF SPOOL VALVE:";B1;"mm."
```

```
1270  PRINT SPA(7);"MAX. SPOOL DISPLACEMENT FROM CENTRE:";Xs;"mm."
1280  IF Onoff THEN PRINT SPA(10);"TIME DELAY FOR THIS ON-OFF
      SPOOL:";Time;"mSec."
1290  IF NOT Onoff THEN PRINT SPA(9);"TIME REQUIRED FOR SPOOL
      OPERATION:";Time;"mSec."
1300  PRINT "ACTUATOR:"
1310  Vt=Vr*Ratio
1320  PRINT " AT MID POSITION, TOTAL VOLUME OF FLUID IN CIRCUI
      T:";Vt;"cu.m."
1330  Var=Vbr=Ratio*Va
1340  IF Sym THEN PRINT "AT MID POSITION, VOLUME OF FLUID IN EACH
      CHAMBER:";Va*Vt;"cu.m."
1350  IF Sym THEN 1450
1360  PRINT "     AT MID POSITION, VOLUME OF FLUID IN CHAMBER
      1:";Va*Vt;"cu.m."
1370  Vb=1-Va
1380  Vbr=Ratio*Vb
1390  PRINT "     AT MID POSITION, VOLUME OF FLUID IN CHAMBER
      2:";Vb*Vt;"cu.m."
1400  PRINT SPA(21);"PISTON DIAMETER (CHAMBER 1):";D1;"cm."
1410  PRINT SPA(21);"PISTON DIAMETER (CHAMBER 2):";D2;"cm."
1420  PRINT SPA(25);"PISTON AREA (CHAMBER 1):";A1;"sq.m."
1430  PRINT SPA(25);"PISTON AREA (CHAMBER 2):";A2;"sq.m."
1440  GOTO 1470
1450  PRINT SPA(33);"PISTON DIAMETER:";D1;"cm."
1460  PRINT SPA(37);"PISTON AREA:";A1;"sq.m."
1470  PRINT SPA(34);"WORKING STROKE:";Ys;"m."
1480  PRINT SPA(20);"INTERNAL LEAKAGE COEFFICIENT:";Cl;"(Litre
      /sec)/Bar"
1490  PRINT SPA(13);"EFFECTIVE EXTERNAL SPRING STIFFNESS:";Ko;
      "kN/m"
1500  PRINT "OTHERS:",LIN(1),"    SYSTEM SUPPLY PRESSURE:";Ps;
      "Bar"
1510  PRINT "   SYSTEM EXHAUST PRESSURE:";Pe;"Bar"
1520  PRINT "  EFFECTIVE EXTERNAL FORCE:";Fo;"kN (ASSUMED CONS
      TANT)"
1530  PRINT SPA(8);"FLUID BULK MODULUS:";Beta;"GN/m^2"
1540  PRINT SPA(13);"FLUID DENSITY:";Rho;"kg/m^3"
1550  ! ****** CONVERSION TO S.I. ****** !
1560  Time=Time/1000       ! Time from mSec
1570  B1=B1/1000           ! B1 from mm
1580  B2=B2/1000           ! B2 from mm
1590  Xs=Xs/1000           ! Xs from mm
1600  Ps=Ps*1E5            ! Ps from Bar
1610  Pe=Pe*1E5            ! Pe from Bar
1620  Beta=Beta*1E9        ! Beta from GN/sq.m.
1630  D1=D1/100            ! D1 from cm
1640  D2=D2/100            ! D2 from cm
1650  Ko=Ko*1000           ! Ko from kN/m
1660  Fo=Fo*1000           ! Fo from kN
1670  ! ******    END OF CONVERSIONS    ****** !
1680  ! ****** CALCULATION OF CONSTANTS ****** !
1690  Refposn=-Vt*((1+Aratio)*Va-1)/(2*A1*Aratio)  ! Ref position
1700  Lref=Va*Vt/A1+Refposn                        ! Ref length
1710  S1=Beta*A1/Lref         ! Hydraulic stiffness column 1
1720  S2=S1
1730  IF Sym THEN 1750
1740  S2=Beta*A2/Lref         ! Hydraulic stiffness column 2
1750  S=S1+S2       !Effective hydraulic stiffness at ref position
1760  Omega=SQR(S/M)          ! Natural hydraulic frequency
1770  PRINT "AT REF POSITION,HYDRAULIC FREQUENCY OF FLUID COLU
      MN:";Omega;"Rad/sec"
1780  PRINT SPA(49);"=";Omega/(2*PI);"Hz"
1790  Tr=1/Omega       ! Ref time = time for 1 radian
1800  Qr=A1*Lref/Tr    ! Ref flow rate
1810  IF Zeta=0 THEN 1890
```

```
1820  Damping=Zeta
1830  Bd=2*M*Zeta*Omega      ! External Damping in physical units
1840  PRINT SPA(11);"EFFECTIVE EXTERNAL DAMPING COEFFICIENT:";
       Bd;"N/(m/sec)"
1850  PRINT SPA(44);"Zeta =";Damping
1860  Tauclose=Time/Tr       ! Non-dimensional valve operation time
1870  PRINT LIN(2),SPA(8);"  NON-DIMENSIONAL TIME FOR CLOSING
       VALVES: ";Tauclose
1880  GOTO 1920
1890  Zeta=Bd/(2*M*Omega)
1900  Damping=Zeta
1910  GOTO 1840
1920  Bdtm=Bd*Tr/M
1930  Pe=Pe/Ps          ! Non-dimensionalize exhaust pressure
1940  K1=Cd1*B1*Xs*SQR(2*Ps/Rho)/Qr
1950  Zeta3=Cd1*B1*Xs*(1+Aratio)/2*SQR(2*Ps/Rho/(1+Aratio*Arat
       io*Aratio))/Qr
1960  Alpha=Beta/Ps
1970  Rtorder=3
1980  IF Onoff THEN Rtorder=2
1990  Rtorder1=Rtorder+1
2000  REDIM Coefficient(Rtorder1),Solution(Rtorder,2),Coeffdum
       (Rtorder1)
2010  IF Onoff THEN Funonoff
2020  Coefficient(1)=2*Alpha*Zeta*Zeta3/Tauclose
2030  Coefficient(2)=1+2*Alpha*Zeta*Zeta3+2*Alpha*Zeta3/Tauclose
2040  Coefficient(3)=2*Zeta+Alpha*Zeta3
2050  Coefficient(4)=1
2060  GOTO Cont1
2070 Funonoff: ! EVALUATE ROOTS(ZEROES) FOR CHARACTERISTIC EQU
      ATION FOR ON-OFF
2080  Coefficient(1)=1+2*Zeta*Alpha*Zeta3
2090  Coefficient(2)=2*Zeta+Alpha*Zeta3
2100  Coefficient(3)=1
2110 Cont1: ! Re-enter at this point
2120  MAT Coeffdum=Coefficient
2130  FOR Idummy=1 TO Rtorder+1
2140  PRINT "Coefficient for y^";Idummy-1;" term = ";Coefficie
       nt(Idummy)
2150  NEXT Idummy
2160  CALL Roots(Coefficient(*),Rtorder,#5)
2170  PRINTER IS 16
2180  READ #5,1
2190  FOR Idummy=1 TO Rtorder
2200  READ #5;Solution(Idummy,1),Solution(Idummy,2)
2210  NEXT Idummy
2220  READ #5,1
2230  MAT Coefficient=Coeffdum
2240  IF Onoff THEN Function1
2250  REM To evaluate the coefficients for the switching function
2260  REM applied to linearly-operated valves
2270  RAD             ! Declare angular units in radians
2280  Operand1(1)=Solution(1,1)-Solution(2,1)
2290  Operand1(2)=Solution(1,2)-Solution(2,2)
2300  Operand2(1)=Solution(2,1)-Solution(3,1)
2310  Operand2(2)=Solution(2,2)-Solution(3,2)
2320  CALL Complexfn(Operand1(*),Operand2(*),Answer(*),Multiply)
2330  Operand1(1)=Solution(3,1)-Solution(1,1)
2340  Operand1(2)=Solution(3,2)-Solution(1,2)
2350  CALL Complexfn(Operand1(*),Answer(*),Result(*),Multiply)
2360  MAT Tempres1=ZER
2370  Idummy=1
2380 Lb1: Idummy1=(Idummy+1) MOD 4+((Idummy+1) DIV 4>0)
2390      Idummy2=(Idummy+2) MOD 4+((Idummy+2) DIV 4>0)
2400  Operand1(1)=Solution(Idummy,1)
2410  Operand1(2)=Solution(Idummy,2)
```

```
2420  CALL Fun1(Operand1(*),Answer(*),Zeta,Tauclose)
2430  Operand1(1)=Solution(Idummy2,1)-Solution(Idummy1,1)
2440  Operand1(2)=Solution(Idummy2,2)-Solution(Idummy1,2)
2450  CALL Complexfn(Operand1(*),Answer(*),Operand2(*),Multiply)
2460  Tempres1(1)=Tempres1(1)+Operand2(1)
2470  Tempres1(2)=Tempres1(2)+Operand2(2)
2480  Idummy=Idummy+1
2490  IF Idummy<=3 THEN Lb1
2500  CALL Complexfn(Tempres1(*),Result(*),Answer(*),Divide)
2510  Coeff3=Answer(1)
2520  MAT Tempres1=ZER
2530  Idummy=1
2540 Lb2: Idummy1=(Idummy+1) MOD 4+((Idummy+1) DIV 4>0)
2550       Idummy2=(Idummy+2) MOD 4+((Idummy+2) DIV 4>0)
2560  Operand1(1)=Solution(Idummy,1)
2570  Operand1(2)=Solution(Idummy,2)
2580  CALL Fun1(Operand1(*),Answer(*),Zeta,Tauclose)
2590  Operand1(1)=Operand1(1)+2*Zeta+Zeta3*Alpha
2600  CALL Complexfn(Operand1(*),Answer(*),Operand2(*),Multiply)
2610  Operand1(1)=Solution(Idummy2,1)-Solution(Idummy1,1)
2620  Operand1(2)=Solution(Idummy2,2)-Solution(Idummy1,2)
2630  CALL Complexfn(Operand1(*),Operand2(*),Answer(*),Multiply)
2640  Tempres1(1)=Tempres1(1)+Answer(1)
2650  Tempres1(2)=Tempres1(2)+Answer(2)
2660  Idummy=Idummy+1
2670  IF Idummy<=3 THEN Lb2
2680  CALL Complexfn(Tempres1(*),Result(*),Answer(*),Divide)
2690  Coeff2=Answer(1)
2700  MAT Tempres1=ZER
2710  MAT Tempres2=ZER
2720  Idummy=1
2730 Lb3: Idummy1=(Idummy+1) MOD 4+((Idummy+1) DIV 4>0)
2740       Idummy2=(Idummy+2) MOD 4+((Idummy+2) DIV 4>0)
2750  Operand1(1)=Solution(Idummy,1)
2760  Operand1(2)=Solution(Idummy,2)
2770  Operand2(1)=Solution(Idummy2,1)-Solution(Idummy1,1)
2780  Operand2(2)=Solution(Idummy2,2)-Solution(Idummy1,2)
2790  CALL Complexfn(Operand2(*),Operand1(*),Answer(*),Divide)
2800  Tempres1(1)=Tempres1(1)+Answer(1)
2810  Tempres1(2)=Tempres1(2)+Answer(2)
2820  CALL Fun1(Operand1(*),Answer(*),Zeta,Tauclose)
2830  CALL Complexfn(Operand2(*),Answer(*),Operand1(*),Multiply)
2840  Sqdummy=2*Zeta/(Solution(Idummy,1)*Solution(Idummy,1)+So
      lution(Idummy,2)*Solution(Idummy,2))
2850  Operand2(1)=1+Sqdummy*Solution(Idummy,1)
2860  Operand2(2)=-1*Sqdummy*Solution(Idummy,2)
2870  CALL Complexfn(Operand1(*),Operand2(*),Answer(*),Multiply)
2880  Tempres2(1)=Tempres2(1)+Answer(1)
2890  Tempres2(2)=Tempres2(2)+Answer(2)
2900  Idummy=Idummy+1
2910  IF Idummy<=3 THEN Lb3
2920  Answer(1)=1-Alpha*Zeta3/Tauclose*(Tempres2(1)-Tempres1(1))
2930  Answer(2)=-1*(Alpha*Zeta3/Tauclose)*(Tempres2(2)-Tempres
      1(2))
2940  Coeff1=Answer(1)
2950  MAT Tempres1=ZER
2960  MAT Tempres2=ZER
2970  Idummy=1
2980 Lb4: Idummy1=(Idummy+1) MOD 4+((Idummy+1) DIV 4>0)
2990       Idummy2=(Idummy+2) MOD 4+((Idummy+2) DIV 4>0)
3000  Operand1(1)=Solution(Idummy,1)
3010  Operand1(2)=Solution(Idummy,2)
3020  CALL Fun2(Operand1(*),Operand2(*),Tauclose)
3030  CALL Fun1(Operand1(*),Answer(*),Zeta,Tauclose)
3040  CALL Complexfn(Operand2(*),Answer(*),Operand1(*),Multiply)
3050  Operand2(1)=Solution(Idummy2,1)-Solution(Idummy1,1)
```

```
3060  Operand2(2)=Solution(Idummy2,2)-Solution(Idummy1,2)
3070  CALL Complexfn(Operand1(*),Operand2(*),Answer(*),Multiply)
3080  Tempres1(1)=Tempres1(1)+Answer(1)
3090  Tempres1(2)=Tempres1(2)+Answer(2)
3100  Idummy=Idummy+1
3110  IF Idummy<=3 THEN Lb4
3120  MAT Tempres2=ZER
3130  Idummy=1
3140 Lb5: Idummy1=(Idummy+1) MOD 4+((Idummy+1) DIV 4>0)
3150      Idummy2=(Idummy+2) MOD 4+((Idummy+2) DIV 4>0)
3160  Operand2(1)=Solution(Idummy,1)
3170  Operand2(2)=Solution(Idummy,2)
3180  Operand1(1)=Solution(Idummy2,1)-Solution(Idummy1,1)
3190  Operand1(2)=Solution(Idummy2,2)-Solution(Idummy1,2)
3200  CALL Complexfn(Operand1(*),Operand2(*),Answer(*),Divide)
3210  Tempres2(1)=Tempres2(1)+Answer(1)
3220  Tempres2(2)=Tempres2(2)+Answer(2)
3230  Idummy=Idummy+1
3240  IF Idummy<=3 THEN Lb5
3250  Tempres1(1)=Tempres1(1)+2*Zeta/Tauclose*Tempres2(1)
3260  Tempres1(2)=Tempres1(2)+2*Zeta/Tauclose*Tempres2(2)
3270  MAT Tempres2=ZER
3280  Idummy=1
3290 Lb6: Idummy1=(Idummy+1) MOD 4+((Idummy+1) DIV 4>0)
3300      Idummy2=(Idummy+2) MOD 4+((Idummy+2) DIV 4>0)
3310  Operand1(1)=Solution(Idummy2,1)-Solution(Idummy1,1)
3320  Operand1(2)=Solution(Idummy2,2)-Solution(Idummy1,2)
3330  Operand2(1)=Solution(Idummy,1)^2-Solution(Idummy,2)^2
3340  Operand2(2)=2*Solution(Idummy,1)*Solution(Idummy,2)
3350  CALL Complexfn(Operand1(*),Operand2(*),Answer(*),Divide)
3360  Tempres2(1)=Tempres2(1)+Answer(1)
3370  Tempres2(2)=Tempres2(2)+Answer(2)
3380  Idummy=Idummy+1
3390  IF Idummy<=3 THEN Lb6
3400  Tempres1(1)=Tempres1(1)+Tempres2(1)/Tauclose
3410  Tempres1(2)=Tempres1(2)+Tempres2(2)/Tauclose
3420  CALL Complexfn(Tempres1(*),Result(*),Answer(*),Divide)
3430  Answer(1)=Answer(1)*Zeta3*(1-Fo/Ps/A1/2)
3440  Answer(2)=Answer(2)*Zeta3
3450  Coeff0=Answer(1)
3460  GOTO Funend
3470 Pause:  DISP "PAUSE At Pause"
3480         PAUSE
3490 Function1:  REM To evaluate the coefficients for switching
     function for ON-OFF valves with/without time delay
3500  RAD
3510  Coeff0=0
3520  Coeff1=1
3530  Coeff2=2*Zeta
3540  Coeff3=1
3550  GOTO Funend
3560  IF (Tauclose=0) AND (Fo=0) THEN Funend
3570  Idummy=1
3580 La1:          ! ********************
3590  Operand1(1)=1
3600  Operand1(2)=0
3610  Operand2(1)=Solution(Idummy,1)
3620  Operand2(2)=Solution(Idummy,2)
3630  CALL Complexfn(Operand1(*),Operand2(*),Answer(*),Divide)
3640  Answer(1)=Solution(Idummy,1)+2*Zeta+Answer(1)
3650  Answer(2)=Solution(Idummy,2)+Answer(2)
3660  IF Idummy>=2 THEN La2
3670  Tempres1(1)=Answer(1)
3680  Tempres1(2)=Answer(2)
3690  Idummy=Idummy+1
3700  GOTO La1
```

```
3710 La2:          ! ********************
3720  Tempres2(1)=Answer(1)
3730  Tempres2(2)=Answer(2)
3740  Operand1(1)=EXP(Tauclose*Solution(1,1))*COS(Tauclose*Sol
      ution(1,2))
3750  Operand1(2)=EXP(Tauclose*Solution(1,1))*SIN(Tauclose*Sol
      ution(1,2))
3760  Operand2(1)=Solution(2,1)
3770  Operand2(2)=Solution(2,2)
3780  CALL Complexfn(Operand1(*),Operand2(*),Answer(*),Multiply)
3790  CALL Complexfn(Answer(*),Tempres1(*),Result(*),Multiply)
3800  Operand1(1)=EXP(Tauclose*Solution(2,1))*COS(Tauclose*Sol
      ution(2,2))
3810  Operand1(2)=EXP(Tauclose*Solution(2,1))*SIN(Tauclose*Sol
      ution(2,2))
3820  Operand2(1)=Solution(1,1)
3830  Operand2(2)=Solution(1,2)
3840  CALL Complexfn(Operand1(*),Operand2(*),Answer(*),Multiply)
3850  CALL Complexfn(Answer(*),Tempres2(*),Operand1(*),Multiply)
3860  CALL Complexfn(Operand1(*),Result(*),Answer(*),Subtract)
3870  Operand1(1)=Solution(1,1)
3880  Operand1(2)=Solution(1,2)
3890  Operand2(1)=Solution(2,1)
3900  Operand2(2)=Solution(2,2)
3910  CALL Complexfn(Operand1(*),Operand2(*),Result(*),Subtract)
3920  CALL Complexfn(Answer(*),Result(*),Operand1(*),Divide)
3930  Coeff2=Operand1(1)+Coefficient(2)/Coefficient(1)
3940  Imaginary=Operand1(2)
3950  PRINT "Coefficient for velocity term [Real] = ";Coeff2
3960  PRINT SPA(25),"[Imaginary] = ";Imaginary
3970  Operand2(1)=EXP(Tauclose*Solution(1,1))*COS(Tauclose*Sol
      ution(1,2))
3980  Operand2(2)=EXP(Tauclose*Solution(1,1))*SIN(Tauclose*Sol
      ution(1,2))
3990  CALL Complexfn(Operand2(*),Tempres1(*),Answer(*),Multiply)
4000  Operand2(1)=EXP(Tauclose*Solution(2,1))*COS(Tauclose*Sol
      ution(2,2))
4010  Operand2(2)=EXP(Tauclose*Solution(2,1))*SIN(Tauclose*Sol
      ution(2,2))
4020  CALL Complexfn(Operand2(*),Tempres2(*),Operand1(*),Multiply)
4030  CALL Complexfn(Answer(*),Operand1(*),Operand2(*),Subtract)
4040  CALL Complexfn(Operand2(*),Result(*),Operand1(*),Divide)
4050  Coeff3=Operand1(1)+1/Coefficient(1)
4060  Imaginary=Operand1(2)
4070  PRINT "Coefficient for acceleration term [Real]: ";Coeff3
4080  PRINT SPA(29),"[Imaginary]: ";Imaginary
4090  Operand2(1)=EXP(Tauclose*Solution(1,1))*COS(Tauclose*Sol
      ution(1,2))
4100  Operand2(2)=EXP(Tauclose*Solution(1,1))*SIN(Tauclose*Sol
      ution(1,2))
4110  CALL Complexfn(Operand2(*),Tempres1(*),Operand1(*),Multiply)
4120  Operand2(1)=Solution(1,1)
4130  Operand2(2)=Solution(1,2)
4140  CALL Complexfn(Operand1(*),Operand2(*),Tempres1(*),Divide)
4150  Operand2(1)=EXP(Tauclose*Solution(2,1))*COS(Tauclose*Sol
      ution(2,2))
4160  Operand2(2)=EXP(Tauclose*Solution(2,1))*SIN(Tauclose*Sol
      ution(2,2))
4170  CALL Complexfn(Operand2(*),Tempres2(*),Operand1(*),Multiply)
4180  Operand2(1)=Solution(2,1)
4190  Operand2(2)=Solution(2,2)
4200  CALL Complexfn(Operand1(*),Operand2(*),Tempres2(*),Divide)
4210  CALL Complexfn(Tempres2(*),Tempres1(*),Answer(*),Subtract)
4220  CALL Complexfn(Answer(*),Result(*),Tempres1(*),Divide)
4230  Tempres2(1)=Zeta3*Tempres1(1)
4240  Tempres2(2)=Zeta3*Tempres1(2)
```

```
4250  Coeff0=(1-Fo/(2*Ps*A1))*Zeta3/Coefficient(1)*(2*Zeta+Tau
       close-Coefficient(2)/Coefficient(1))+Tempres2(1)
4260  Imaginary=Tempres2(2)
4270  PRINT "Coefficient for constant term [Real]: ";Coeff0
4280  PRINT SPA(25),"[Imaginary]: ";Imaginary
4290 Funend:       REM Coefficients have been evaluated
4300  PRINTER IS 0   ! Print these onto paper
4310  PRINT "Coeff. for Accel term = ";Coeff3
4320  PRINT "Coeff. for Vel   term = ";Coeff2
4330  PRINT "Coeff. for Disp  term = ";Coeff1
4340  PRINT "Coeff. for Const term = ";Coeff0
4350  PRINTER IS Printer
4360  K2=K1
4370  IF Sym THEN 4390
4380  K2=K1*(B2/B1)*(Cd2/Cd1)
4390  K3=Tr*Tr*Ps*A1/(M*Lref)     ! Evaluate non-dimensional qu
       antities
4400  Bd=Bd*Lref/(Ps*A1*Tr)
4410  Ko=Ko*Lref/(Ps*A1)
4420  Fo=Fo/(Ps*A1)
4430  Velr=Lref/Tr
4440  Accelr=Velr/Tr
4450  K4=Cl*Ps/Qr
4460  K5=Beta/Ps
4470  Thic=K5*K3*2/Var
4480  Epsilopi=0
4490  IF Onoff THEN 4510
4500  Epsilopi=Tr/Time            ! Rate of change of valve opening
4510  G=K1*Beta*A1*Tr*Tr/(M*Vr*Va*Lref*2)
4520  IF Onoff THEN G=2*G
4530  ! ****** INPUT INITIAL CONDITIONS ****** !
4540  IF Again THEN 4640
4550  BEEP
4560  INPUT "INITIAL DISPLACEMENT OF PISTON FROM CENTRE IN m",Y
4570  Y=Y-Refposn
4580  IF ABS(Y)<=Ys/2 THEN 4610
4590  BEEP
4600  GOTO 4550
4610  INPUT "INCREMENT FOR INITIAL DISPLACEMENT IN m",Deltai
4620  Deltai=Deltai/Lref
4630  GOTO 4650
4640  Y=(Deltai+Thetai)*Lref
4650  PRINT "INITIAL CONDITIONS:"
4660  PRINT "  PISTON DISPLACEMENT FROM CENTRE :";Y+Refposn;"m."
4670  Theta=Y/Lref     !Non-dimensional Displacement
4680  Thetai=Theta     !Initial Displacement
4690  IF NOT Again THEN 4720
4700  Vel=Veli
4710  GOTO 4740
4720  BEEP
4730  INPUT "INITIAL RAM VELOCITY IN m/s",Vel
4740  Veli=Vel
4750  PRINT SPA(22);"RAM VELOCITY:";Vel;"m/sec"
4760  Thetapi=Vel/Velr   !Non-dimensionalize velocity
4770  IF Veli<>0 THEN 4820
4780  Aratio3=Aratio*Aratio*Aratio
4790  P1=Ps*(Aratio3+Fo)/(1+Aratio3)*1E-5
4800  P2=Aratio*Aratio*(Ps-P1*1E5)*1E-5
4810  GOTO 4890
4820  IF NOT Again THEN 4860
4830  P1=P1i
4840  P2=P2i
4850  GOTO 4890
4860  BEEP
4870  INPUT "INITIAL PRESSURE IN CHAMBER 1 IN Bar",P1
4880  P1i=P1
```

```
4890  PRINT SPA(13);"PRESSURE IN CHAMBER 1:";P1;"Bar"
4900  P1=P1*1E5/Ps     !Non-dimensionalize P1
4910  IF Again OR (Vel=0) THEN 4950
4920  BEEP
4930  INPUT "INITIAL PRESSURE IN CHAMBER 2 IN Bar",P2
4940  P2i=P2
4950  PRINT SPA(13);"PRESSURE IN CHAMBER 2:";P2;"Bar"
4960  P2=P2*1E5/Ps
4970  PRINT LIN(1),"REFERENCE QUANTITIES:"
4980  PRINT "         TIME:";Tr*1000;"mSec."
4990  PRINT "     PRESSURE:";Ps/1E5;"Bar"
5000  PRINT "    FLOW RATE:";Qr*1000;"Litre/sec"
5010  PRINT "       LENGTH:";Lref;"m."
5020  PRINT "     POSITION:";Refposn;"m. from Mid-stroke position"
5030  PRINT "         AREA:";A1;"sq.m."
5040  PRINT "       VOLUME:";A1*Lref;"cu.m."
5050  PRINT "     VELOCITY:";Velr;"m/s"
5060  PRINT "ACCELERATION:";Accelr;"m/s^2"
5070  IF NOT Again THEN 5180
5080  Thetaf=Thetaf+Deltaf
5090  Yf=Lref*Thetaf
5100  IF ABS(Thetaf*Lref+Refposn)<=.5*Ys THEN 5270
5110  DISP "Thetaf is now greater than .5 on either side"
5120  PRINTER IS 16
5130  PRINT PAGE,LIN(5),SPA(20),"Please remove the disc from
      the drive"
5140  PRINT LIN(1),SPA(23),"and place it in the pocket marke
      d:",LIN(2)
5150  PRINT SPA(40),"Flexible Disc"
5160  BEEP
5170  STOP
5180  ! ****** INPUT FINAL DESIRED POSITION ****** !
5190  BEEP
5200  INPUT "FINAL DESIRED POSITION FROM CENTRE IN m.",Yf
5210  IF ABS(Yf)<=Ys/2 THEN 5230     ! Check if within the stroke
5220  GOTO 5180
5230  Yf=Yf-Refposn       ! Position relative to the Ref position
5240  INPUT "INCREMENT OF FINAL DESIRED POSITION IN m",Deltas
5250  Deltas=Deltas/Lref     ! Non-dimensionlize this quantity
5260  Deltaf=Deltas
5270  PRINT LIN(1),"FINAL DESIRED POSITION FROM REF POSN.:";Yf
      ;"m.",LIN(1)
5280  Thetas=Yf/Lref
5290  Thetaf=Thetas
5300  Thi2=Bdtm*Bdtm*Thetaf*(4*Thic-1)/4
5310  Thetaf2=Thetaf
5320  Thetar=Thetas-Theta
5330  REM Alpha1=Alpha2*Thetaf-1
5340  Alpha2=2*Ps*A1*Tr*Tr/((Ratio*Va-Thetaf)*M*Ys)
5350  Qrootsq=(Alpha2-Alpha1*Alpha1)/4
5360  Alpha3=Thetaf*(Qrootsq-Alpha2)
5370  Wtval=1
5380  ! ****** CALCULATE TIME INCREMENT ****** !
5390  Deltat=.02*PI
5400  PRINT "THE TIME INCREMENT IS CHOSEN TO BE:";Deltat*1E6;"
      microSec.",LIN(6)
5410  IMAGE " ERROR    S",2X,5(10A,4X)
5420  PRINTER IS 16
5430  ! Print messages onto the CRT
5440  PRINT PAGE,SPA(25),"Please DO NOT Interrupt",LIN(2)
5450  PRINT SPA(25),"Please USE other machine if available",
      LIN(2)
5460  PRINT SPA(10),"PERMISSION HAS BEEN GIVEN TO LEAVE COMPUTER
      RUNNING OVERNIGHT",LIN(5)
5470  PRINT USING 5410;"THETA","THETA-PI","THETA-2-PI","Signal
      F","EPSILO"
```

```
5480  ! End of print message segment
5490  ! Print information on paper
5500  PRINTER IS 0
5510  PRINT LIN(2),"Mass of moving parts = ";M;" Kg","Damping
       Coeff = ";Damping
5520  PRINT "Final desired position: ";Thetas;" ( ";Thetas*Lre
       f+Refposn;" m. from centre)"
5530  PRINT "External Force Applied: ";Fo;" ( ";Fo*Ps*A1/1000;"
       kN)"
5540  PRINT USING 5550;Tauclose,Tauclose*Tr*1000
5550  IMAGE #,"Tau close : ",MD.3DE,2X,"( ",3D.D,2X,"mSec."
5560  IF Onoff THEN PRINT "ON-OFF Valves)",LIN(2)
5570  IF NOT Onoff THEN PRINT "Linearly Operated Valves)",LIN(2)
5580  PRINT USING 5590;"Tau","Theta","Thetapi","Theta2pi","Fun
       ction"
5590  IMAGE 3X,6(10A,4X)
5600  PRINTER IS 16
5610  PLOTTER IS 13,"GRAPHICS"
5620  Tausc=80
5630  IF Tlimit>1E99 THEN 5650
5640  Tausc=1.1*Tlimit
5650  Dummymin=MIN(Theta,Thetaf)
5660  Dummymax=MAX(Theta,Thetaf)
5670  Dummyrange=Dummymax-Dummymin
5680  Dummymin=Dummymin-.1*Dummyrange
5690  Dummymax=Dummymax+.1*Dummyrange
5700  Tausc=Tausc*MAX(INT(ABS(Dummyrange/.05)),1)
5710  IF Dummyrange<>0 THEN 5740
5720  SCALE -Tausc/10,Tausc,Thetaf-.01,.1+Thetaf
5730  GOTO 5750
5740  SCALE -Tausc/10,Tausc,Dummymin,Dummymax
5750  AXES
5760  MOVE 0,Thetaf
5770  LINE TYPE 7
5780  DRAW Tausc,Thetas
5790  LINE TYPE 1
5800  MOVE Tausc/2,Thetaf+(Thetaf-Theta)/4
5810  LORG 6
5820  LABEL "FINAL DESIRED POSITION: "&VAL$(Thetas)
5830  LORG 5
5840  MOVE 0,0
5850  ON KEY #3 GOSUB Graph
5860  ON KEY #4 GOTO Endcal
5870  ON ERROR GOTO Error5
5880  ON KEY #5 GOSUB Chsignal
5890  ON KEY #2 GOSUB Changeprinter
5900  GOSUB Deriv
5910  GOSUB Signchk
5920  IF Signal<>Presign THEN Switch_valve
5930  GOSUB Intrgl
5940  IF (Tau>=Tlimit) OR (Npoints>=400) THEN Endcal
5950  IF Sw<>2 THEN 5900
5960  IF Pscor THEN 5900
5970  Sw=Sw+2
5980  IF Scor THEN Sw=10
5990  Sw=Sw+2
6000  GOTO 5900
6010 Switch_valve:  REM SWITCHING TAKES PLACE NOW
6020  PRINTER IS 0  ! Print information at the point of switching
6030  PRINT USING 6040;Tau,Theta,Thetapi,Theta2pi,Signalfn
6040  IMAGE 2X,5(MD.5DE,2X),7X
6050  PRINTER IS 16
6060  IF Thetaf<>Thetaf1 THEN Sw=0
6070  IF Sw THEN 6110
6080  Scor=0
6090  IF Signal=SGN(Thetae) THEN Scor=1
```

```
6100  Sorg=Signal
6110  Sw=Sw+1      ! COUNT THE NUMBER OF SWITCHINGS
6120  IF Sw=1 THEN 6150
6130  Pscor=0
6140  IF Presign1=Signal THEN Pscor=1
6150  IF Sw<>5 THEN 6170
6160  IF Signal=Sorg THEN Sw=Sw+2
6170  IF Onoff THEN Switch=1
6180  IF Signal<0 THEN 6210
6190  LABEL "*"
6200  GOTO 6220
6210  LABEL "+"
6220  Psws=Psws+1
6230  Pswitch(Psws)=Npoints
6240  MOVE Tau,Theta
6250  IF Time=0 THEN 6430
6260  Deltatd=Deltat
6270  Deltatr=Time/Tr/100
6280  Deltat=MIN(Deltatr,Deltatd)
6290  Taui=Tau
6300  Tauf=Tau+Time/Tr-Deltat
6310  IF Onoff THEN Pureonoff
6320  GOSUB Intrgl
6330  GOSUB Deriv
6340  IF Tau<Tauf THEN 6320
6350  LABEL "#"
6360  IF ABS(Epsilo)<.5 THEN Signal=0
6370  GOTO 5930
6380 Pureonoff:  Taue=0
6390  GOSUB Intrgl
6400  GOSUB Deriv
6410  IF Tau<Tauf THEN 6380
6420  LABEL "#"
6430  Switch=0
6440  GOSUB Intrgl
6450  IF ABS(Epsilo)<.5 THEN Signal=0
6460  GOTO 5940
6470 Endcal: PRINTER IS 0
6480  PRINT USING 6040;Tau,Theta,Thetapi,Theta2pi,Signalf1
6490  PRINT LIN(2),"PRESSSURE IN CHAMBER 1 : ",P1*Ps/1E5;" Bar"
6500  PRINT "PRESSSURE IN CHAMBER 2 : ",P2*Ps/1E5;" Bar"
6510  PRINT "RESULTS HAVE BEEN WRITTEN ON FILE NUMBERED: ";No,
      LIN(5)
6520  PRINTER IS Printer
6530  ASSIGN #6 TO "SMDA"&VAL$(No)&Msus$,Check
6540  IF Check=0 THEN 6570
6550  CREATE "SMDA"&VAL$(No)&Msus$,20
6560  GOTO 6520
6570  PRINT #6;Fo,Damping,Wtval,Sety,M,Tauclose,Tr,Type,Sym,Ar
      atio,Npoints,Psws
6580  FOR Dummy=1 TO Psws
6590  PRINT #6;Pswitch(Dummy)
6600  NEXT Dummy
6610  FOR Dummy=1 TO Npoints-1
6620  Shortdummy1=Tauc(Dummy)
6630  Shortdummy2=Z(Dummy)
6640  Shortdummy3=Ep(Dummy)
6650  PRINT #6;Shortdummy1,Shortdummy2,Shortdummy3
6660  NEXT Dummy
6670  PRINT #6;END
6680  ASSIGN * TO #6
6690  No=No+1
6700  ASSIGN #6 TO "SMDA"&VAL$(No)&Msus$,Check
6710  IF Check=0 THEN 6740
6720  CREATE "SMDA"&VAL$(No)&Msus$,20
6730  GOTO 6700
```

```
6740  PRINT #6;Fo,Damping,Wtval,Sety,M,Tauclose,Tr,Type,Sym,Ar
       atio,Npoints,Psws
6750  FOR Dummy=1 TO Psws
6760  PRINT #6;Pswitch(Dummy)
6770  NEXT Dummy
6780  FOR Dummy=1 TO Npoints-1
6790  Shortdummy1=Tauc(Dummy)
6800  Shortdummy2=Ramp1(Dummy)
6810  Shortdummy3=Ramp2(Dummy)
6820  PRINT #6;Shortdummy1,Shortdummy2,Shortdummy3
6830  NEXT Dummy
6840  PRINT #6;END
6850  ASSIGN * TO #6
6860  EXIT GRAPHICS
6870  IF Again THEN 6910
6880  IF ERRN THEN PRINT ERRM$
6890  INPUT "INPUT 1 to run again, OTHERWISE INPUT 0 to stop
       ",Again
6900  IF NOT Again THEN STOP
6910  No=No+1
6920  REM IF INT(No/2)*2-No=0 THEN Tlimit=Tlimit+Tdummyc
6930  Tlimit=Tlimit+Tdummyc
6940  GOTO 420
6950  STOP
6960 Graph:  REM ENTERING GRAPHIC MODE
6970  GRAPHICS
6980  RETURN
6990 Changeprinter:  REM CHANGE PRINTER DEVICE
7000  Printer=16-Printer
7010  PRINTER IS Printer
7020  RETURN
7030  REM PROGRAMME STORED IN "SW2809" ON 1ST October 1981 ON
       DISC #51
7040 Signchk:  REM CHECK SIGNAL
7050  Presign=Signal
7060  IF Sw<>0 THEN 7220
7070  IF ABS(Thetaf-Theta)<1E-6 THEN 7100
7080  IF (Thetapi<>0) OR (Theta2pi<>0) THEN 7220
7090  IF ABS(Thetaf-Theta)>1E-6 THEN 7120
7100  Signal=Presign
7110  RETURN
7120  Signalfn=Coeff3*Theta2pi+Coeff2*Thetapi+Coeff1*(Theta-Th
       etaf)+SGN(Thetapi)*Coeff0
7130  IF Epsilo<0 THEN 7160
7140  Signalfn=Signalfn+(1+Onoff)*Time/Tr/2*Thetapi
7150  GOTO 7170
7160  Signalfn=Coeff3*Theta2pi+Coeff2*Thetapi+Coeff1*Theta-The
       taf+Coeff0
7170  Signal=SGN(-Signalfn)
7180  RETURN
7190 Signsw:  Signal=-Signal
7200  Sw=100
7210  RETURN
7220  Signalf1=Coeff3*Theta2pi+Coeff2*Thetapi+Coeff1*(Theta-Th
       etaf)+SGN(Thetapi)*Coeff0
7230  IF Epsilo<0 THEN 7660
7240  Signalf1=Signalf1+(1+Onoff)*Time/Tr/2*Thetapi
7250  GOTO 7270
7260  Signalf1=Coeff3*Theta2pi+Coeff2*Thetapi+Coeff1*Theta-The
       taf+Coeff0
7270  IF Sw>7 THEN Set0s
7280  Signal=SGN(-Signalf1)
7290  Signalfn=Signalf1
7300  RETURN
7310 Chsignal:  Signal=-Signal
7320  Signalfn=Signalf1
```

```
7330  RETURN
7340 Set0s: Epsilo=Presign=Signal=0
7350  RETURN
7360 Deriv:  REM CALCULATE DERIVATIVES
7370  Thetae=Thetaf-Theta
7380  Ppi1=1
7390  Ppi2=Pe
7400  IF Epsilo>=0 THEN Skippi
7410  Ppi1=Pe
7420  Ppi2=1
7430 Skippi:  REM CALCULATE FLOW RATES
7440  Q1=K1*ABS(Epsilo)*SQR(ABS(Ppi1-P1))*SGN(Ppi1-P1)
7450  Q2=K2*ABS(Epsilo)*SQR(ABS(P2-Ppi2))*SGN(P2-Ppi2)
7460  P1pi=K5*(Q1-Thetapi-K4*(P1-P2))/(1+Theta)
7470  P2pi=K5*(K4*(P1-P2)+Aratio*Thetapi-Q2)/((1-Theta)*Aratio)
7480  Theta2pi=K3*(P1-Aratio*P2-Bd*Thetapi-Ko*Theta-Fo)
7490  IF Print THEN Noprint
7500  PRINT USING Prtim;Thetae,Signal,Theta,Thetapi,Theta2pi,S
      ignalf1,Epsilo
7510  IF Npoints>1 THEN 7530
7520  IF ABS(Theta-Thetas)>.1 THEN 7590
7530  Npoints=Npoints+1
7540  Tauc(Npoints)=Tau
7550  Z(Npoints)=Theta
7560  Ep(Npoints)=Thetaf
7570  Ramp1(Npoints)=P1
7580  Ramp2(Npoints)=P2
7590  DISP Tau;Npoints;Thetaf,P1*Ps/1E5;P2*Ps/1E5
7600 Noprint:  Print=Print+1
7610  PLOT Tau,Theta
7620  IF Print>=5 THEN Print=0
7630 Prtim:  IMAGE +,MD.DE ,X,MD,X,4(MD.5DE,2X),MD.5DE
7640  RETURN
7650 Intrgl:  REM CALCULATE INTEGRALS
7660 Presign1=-SGN(Signalf1)
7670  Signalf1=Coeff3*Theta2pi+Coeff2*Thetapi+Coeff1*(Theta-Th
      etaf)+Coeff0
7680  IF Epsilo<0 THEN 7710
7690  Signalf1=Signalf1+(1+Onoff)*Time/Tr/2*Thetapi
7700  GOTO 7720
7710  Signalf1=Signalf1+(1+Onoff)*Time/Tr/2*Thetapi*Aratio
7720  Thetaf1=Thetaf
7730  Thetaf2=Thetaf2+Deltaf
7740  IF ABS(Thetaf2)>=ABS(Thetas) THEN Thetaf2=Thetas
7750  Tau=Tau+Deltat
7760  Thetaf=Thetaf1
7770  Theta=Theta+Deltat*Thetapi
7780  Thetapi=Thetapi+Deltat*Theta2pi
7790  Epsilo=Epsilo+Epsilopi*Deltat*Signal+Signal*Onoff*(Switch=0)
7800  IF ABS(Epsilo)>1 THEN Epsilo=SGN(Epsilo)
7810  P1=P1+Deltat*P1pi
7820  P2=P2+Deltat*P2pi
7830  IF P1<=0 THEN P1=0
7840  IF P2<=0 THEN P2=0
7850  RETURN
7860 Error5:  REM ERROR DURING EXECUTION
7870  DISP ERRM$
7880  BEEP
7890  OFF ERROR
7900  GOTO Endcal
7910  END
7920  SUB Roots(K(*),INTEGER N,#5)
7930  REM   ****************************
7940  REM   * PROGRAM STORED IN "ROOTS" *
7950  REM   ****************************
7960  OPTION BASE 1
```

```
7970  REM ************************************************
7980  REM * FINDS THE ROOTS OF POLYNOMIAL                 *
7990  REM * VERSION 3/DECEMBER 1981                       *
8000  REM *     N --- ORDER OF POLYNOMIAL (LESS THAN 3)   *
8010  REM *     K --- COEFFICIENTS OF POLYNOMIAL          *
8020  REM * ROOTS ARE STORED IN THE FILE ASSIGNED TO #5   *
8030  REM * THIS FILE IS CALLED "SOLN"                    *
8040  REM ************************************************
8050  H1=N                    !Conserve N
8060  IF N<=0 THEN 9370
8070  DIM A(4),B(4),X(4)
8080  FOR I=1 TO N+1
8090  A(I)=K(N+2-I)
8100  B(I)=A(I)
8110  NEXT I
8120  PRINT
8130  C1=0
8140  D1=4
8150  O=0
8160  PRINT
8170  PRINT
8180  PRINT
8190  PRINT "   The roots for the characteristic equation are:"
8200  PRINT
8210  PRINT TAB(14);"Real Part";TAB(32);"Imag Part"
8220  PRINT
8230  IF N<=2 THEN 8940
8240  IF A(N+1)=0 THEN 9030
8250  IF N/2-INT(N/2)=0 THEN 8280
8260  GOSUB 9120
8270  GOTO 8230
8280  IF ABS(A(N-1))<1E-60 THEN 8320
8290  P=A(N)/A(N-1)
8300  Q=A(N+1)/A(N-1)
8310  GOTO 8340
8320  P=A(N)
8330  Q=A(N+1)
8340  FOR I=1 TO N+1
8350  X(I)=A(I)
8360  NEXT I
8370  GOSUB 9070
8380  FOR I=1 TO N-1
8390  B(I)=X(I)
8400  NEXT I
8410  R=X(N)
8420  S=A(N+1)-P*X(N)-Q*X(N-1)
8430  GOSUB 9070
8440  X(N)=P*X(N-1)-Q*X(N-2)
8450  D=X(N-1)^2-X(N)*X(N-2)
8460  IF ABS(D)>1E-60 THEN 8490
8470  PRINT "Solution unobtainable with this Program"
8480  GOTO 9370
8490  P1=P+(R*X(N-1)-S*X(N-2))/D
8500  Q1=Q+(S*X(N-1)-R*X(N))/D
8510  IF ABS(P)>1E-60 THEN 8550
8520  IF ABS(P1)>1E-60 THEN 8550
8530  IF ABS(Q)>1E-60 THEN 8560
8540  GOTO 8570
8550  IF ABS(P1/P-1)>.000001 THEN 8570 !Relative error allowed
       1E-6
8560  IF ABS(Q1/Q-1)<.000001 THEN 8710
8570  P=P1
8580  Q=Q1
8590  C1=C1+1
8600  IF C1=D1*25 THEN 8620
8610  GOTO 8340
```

```
8620  PRINT
8630  PRINT "The solution did not converge after";C1;"iterations"
8640  PRINT "To CONTINUE for 25 more iterations TYPE 1 otherwise"
8650  PRINT "TYPE 0"
8660  INPUT K1
8670  IF K1=1 THEN 8690
8680  GOTO 9370
8690  D1=D1+1
8700  GOTO 8340
8710  FOR I=2 TO N-1
8720  A(I)=B(I)
8730  NEXT I
8740  N=N-2
8750  D=P*P-4*Q
8760  IF D<0 THEN 8860
8770  D=SQR(D)
8780  PRINT USING 8810;(-P+D)/2,0
8790  PRINT USING 8810;(-P-D)/2,0
8800  PRINT #5;(-P+D)/2,0,(-P-D)/2,0
8810  IMAGE 4X,2(6X,MD.5DXE),6X,MD.5D
8820  C1=0
8830  D1=4
8840  IF N-2>0 THEN 8240
8850  GOTO 8940
8860  D=SQR(-D)
8870  Z1=P/2/SQR((P/2)^2+(D/2)^2)
8880  PRINT USING 8810;-P/2,D/2
8890  PRINT USING 8810;-P/2,-D/2
8900  PRINT #5;-P/2,D/2,-P/2,-D/2
8910  C1=0
8920  D1=4
8930  IF N-2>0 THEN 8240
8940  IF N=1 THEN 9000
8950  IF N=0 THEN 9370
8960  P=B(2)/B(1)
8970  Q=B(3)/B(1)
8980  N=0
8990  GOTO 8750
9000  PRINT USING 8810;-B(2)/B(1),0
9010  PRINT #5;-B(2)/B(1)
9020  GOTO 9370
9030  PRINT USING 8810;0,0
9040  PRINT #5;0,0
9050  N=N-1
9060  GOTO 8230
9070  X(2)=X(2)-P*X(1)
9080  FOR I=3 TO N
9090  X(I)=X(I)-P*X(I-1)-Q*X(I-2)
9100  NEXT I
9110  RETURN
9120  IF B(2)=0 THEN 9150
9130  X=-B(2)/B(1)
9140  GOTO 9160
9150  X=-B(N+1)/B(1)
9160  F=0
9170  F1=0
9180  FOR I=1 TO N+1
9190  J=N-I+2
9200  IF B(J)=0 THEN 9240
9210  F=B(J)*X^(I-1)+F
9220  IF I-1=0 THEN 9240
9230  F1=(I-1)*B(J)*X^(I-2)+F1
9240  NEXT I
9250  X1=X-F/F1
9260  IF ABS(X/X1-1)<.000001 THEN 9290
9270  X=X1
```

```
9280  GOTO 9160
9290  PRINT USING 8810;X1,0
9300  PRINT #5;X1,0
9310  N=N-1
9320  FOR I=2 TO N+1
9330  A(I)=B(I)+X1*A(I-1)
9340  B(I)=A(I)
9350  NEXT I
9360  RETURN
9370  N=H1
9380  SUBEND
9390 Lcomp:   SUB Complexfn(Operand1(*),Operand2(*),Answer(*),
      INTEGER Operator)
9400  REM  ********************************************
9410  REM  * SUB-PROGRAMS TO OPERATE ON COMPLEX NUMBERS *
9420  REM  ********************************************
9430  OPTION BASE 1
9440  DIM A(2),B(2),C(2)
9450  FOR I=1 TO 2
9460  A(I)=Operand1(I)
9470  B(I)=Operand2(I)
9480  NEXT I
9490  ! ***********************************
9500  ! * Operator = 1 for Addition       *
9510  ! * Operator = 2 for Subtraction    *
9520  ! * Operator = 3 for Multiplication *
9530  ! * Operator = 4 for Division       *
9540  ! ***********************************
9550  ON Operator GOTO Add,Subtract,Multiply,Divide
9560  DISP "OPERATOR INCORRECTLY SPECIFIED"
9570  BEEP
9580  STOP
9590 Add:  REM Add two complex numbers together
9600  C(1)=A(1)+B(1)
9610  C(2)=A(2)+B(2)
9620  GOTO Complete
9630 Multiply:  REM Multiply two complex numbers together
9640  C(1)=A(1)*B(1)-A(2)*B(2)
9650  C(2)=A(1)*B(2)+A(2)*B(1)
9660  GOTO Complete
9670 Subtract:  REM Subtract two complex numbers: A - B
9680  C(1)=A(1)-B(1)
9690  C(2)=A(2)-B(2)
9700  GOTO Complete
9710 Divide:  REM Divide two complex numbers: A / B
9720  Dummy=B(1)*B(1)+B(2)*B(2)
9730  C(1)=(A(1)*B(1)+A(2)*B(2))/Dummy
9740  C(2)=(A(2)*B(1)-A(1)*B(2))/Dummy
9750 Complete: ! Operation completed
9760  Answer(1)=C(1)
9770  Answer(2)=C(2)
9780  SUBEXIT
9790  SUBEND
9800 Lfun1:  SUB Fun1(Operand(*),Answer(*),Zeta,Tauclose)
9810  OPTION BASE 1
9820  ! ******************************************************
9830  ! * Programme to evaluate:                             *
9840  ! * (1+2*Zeta*Operand+Operand^2)*EXP(Operand*Tauclose) *
9850  ! ******************************************************
9860  RAD
9870  DIM A(2),B(2)
9880  A(1)=1+2*Zeta*Operand(1)+Operand(1)*Operand(1)-Operand(2
       )*Operand(2)
9890  A(2)=2*Zeta*Operand(2)+2*Operand(1)*Operand(2)
9900  B(1)=EXP(Operand(1)*Tauclose)*COS(Operand(2)*Tauclose)
9910  B(2)=EXP(Operand(1)*Tauclose)*SIN(Operand(2)*Tauclose)
```

```
9920  CALL Complexfn(A(*),B(*),Answer(*),3)
9930  SUBEXIT
9940  SUBEND
9950 Lfun2:  SUB Fun2(Operand(*),Answer(*),Tauclose)
9960  ! ************************************************
9970  ! * To evaluate the expression:                  *
9980  ! *    (Operand*Tauclose-1)/(Tauclose*Operand)/Operand *
9990  ! ************************************************
10000 !  1-1/(Tauclose*Operand)=(Tauclose*Operand-1)/(Tauclose
      *Operand)
10010 OPTION BASE 1
10020 DIM A(2),B(2)
10030 MAT A=(Tauclose)*Operand
10040 B(1)=A(1)-1
10050 B(2)=A(2)
10060 CALL Complexfn(B(*),A(*),Answer(*),4)
10070 MAT A=Operand
10080 MAT B=Answer
10090 CALL Complexfn(B(*),A(*),Answer(*),4)
10100 SUBEXIT
10110 SUBEND
```

APPENDIX D

STIFFNESS ANALYSIS OF A MINING BOOM

D.0. STIFFNESS ANALYSIS OF A BOOM

One may define the stiffness of a boom of a mining machinery as the ratio of change in force to the change in displacement of the working end in the direction of the applying force. Although this quantity depends on both the structural and hydraulic stiffnesses, only the hydraulic stiffness is investigated because the boom arms are normally supported by the hydraulic rams.

Suppose there is a boom-ripper as shown in Figure D.1.

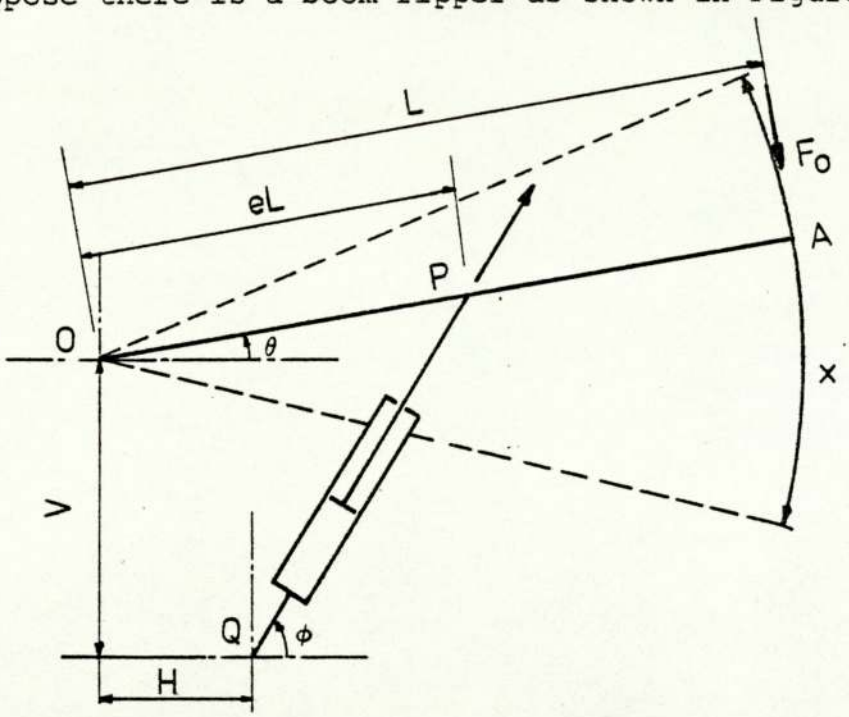


Figure D.1. Diagrammatic Representation of a Boom-ripper

The movement of this boom (OA) in the vertical plane is provided by the hydraulic ram (QP). The analysis is simplified by assuming that all movements in other planes are risisted by other means. The worst case of this system will be that when the cutter at boom-end attempts to cut a hard rock, the force resisting will lift up the

boom so that the cutter and boom "climb" over the rock. This force will be a maximum when it is cutting in a perpendicular direction to the boom. By considering the geometry of the system, the hydraulic properties of the oil in the ram, the maximum force, F_o, and applying the virtual work principle, we have,

$$\text{Stiffness} = \frac{d}{dx} F_o = \frac{e^2 \beta A^2 \sin(\theta-\phi)}{V}$$

where β = Bulk modulus of the fluid in the ram

V = Volume of the fluid column

A = Cross-sectional area of the piston

e, θ, ϕ as defined in Figure D.1

It can be shown that, if the ram's cross-sectional area is ajdusted so that the pressure force within the cylinder can just resist the maximum external force when the ram is fully extended, the stiffness will only vary slightly with the location of the ram; while there is a substantial change in stiffness if the area has been kept constant.

D.1. STIFFNESS OF A DOSCO BOOM-RIPPER

The DOSCO boom-ripper has a slightly different configuration from that shown in Figure D.1. It can be diagrammatically represented in Figure D.2.

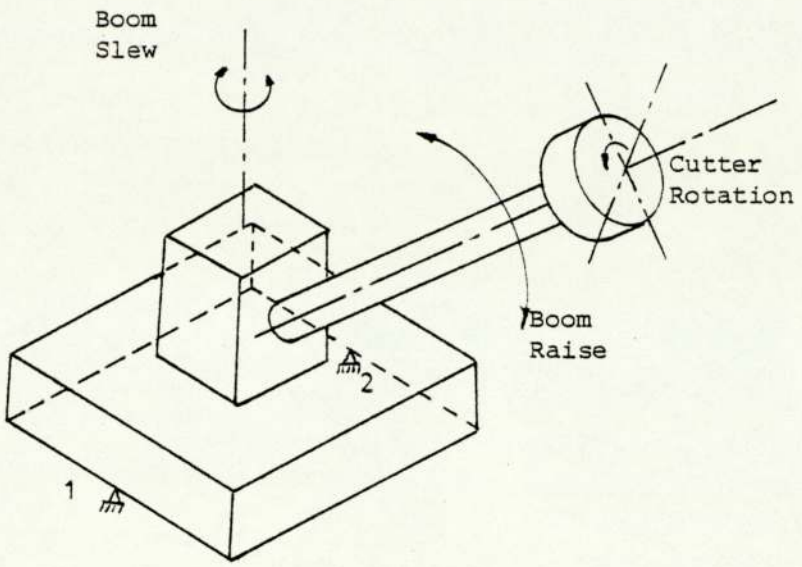


Figure D.2. A Boom-ripper

The vertical movement of the boom can be represented as in Figure D.3.

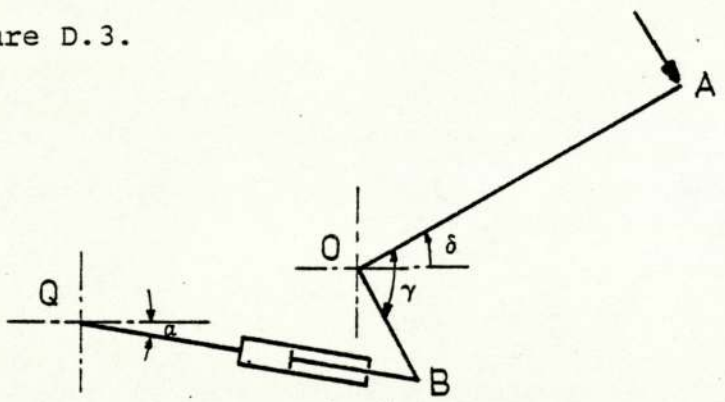


Figure D.3. Diagram to Show the Arrangement to Provide Vertical Movement of the Boom

The boom OA is rotating about O in the vertical plane when the link AOB may be considered as rigid. The hydraulic ram PB provides the movement in any vertical plane. Since a fixed configuration is considered, the stiffness is investigated with the angle of elevation varying.

If the profile to be cut is considered and has a radius, R, then the position of the cutter can be represented as a function of R and the angle of rotation, θ, between the line joining the profile centre point (as in Figure D.4), and the horizontal.

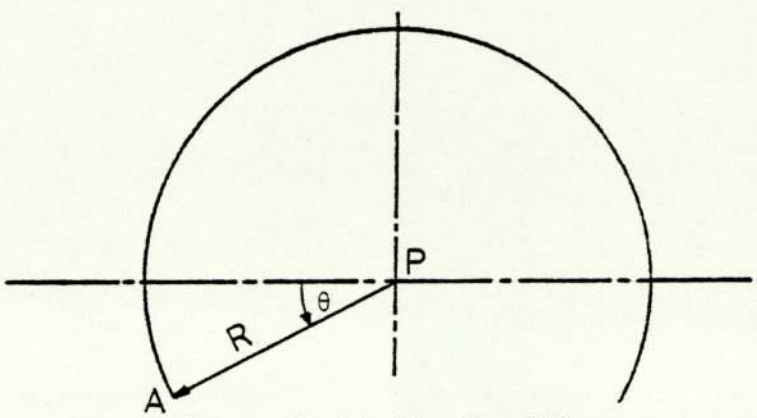


Figure D.4. The Profile

There will be a certain relationship among these quantities and other physical dimensions of the machine. The mechanism of the machine can be represented by Figure D.5.

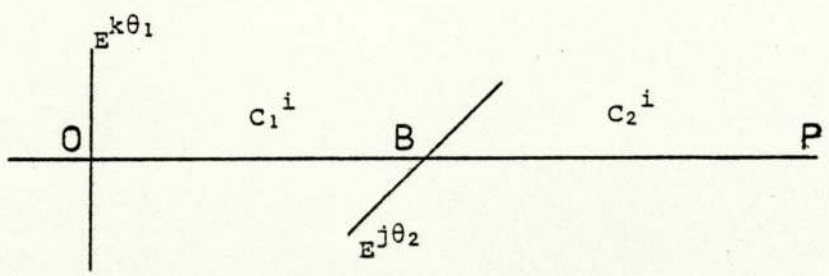


Figure D.5. Diagrammatic Representation of the Mechanism of the Boom-ripper

Figure D.5. indicates that the links $C_1{}^i$ and $C_2{}^i$ will rotate about O in a horizontal plane while the link $C_2{}^i$ will rotate about B in a vertical plane that contains $C_2{}^i$. Therefore any position of P can be found by applying this model.

The stiffness for this type of boom is found to be:-

$$\text{Stiffness} = \left[\frac{a \sin(\gamma-\delta-\alpha)}{L \cos \delta}\right]^2 S$$

where δ = Angle of elevation of the boom

α = Angle of inclination of the ram to horizontal

γ = Angle AOB

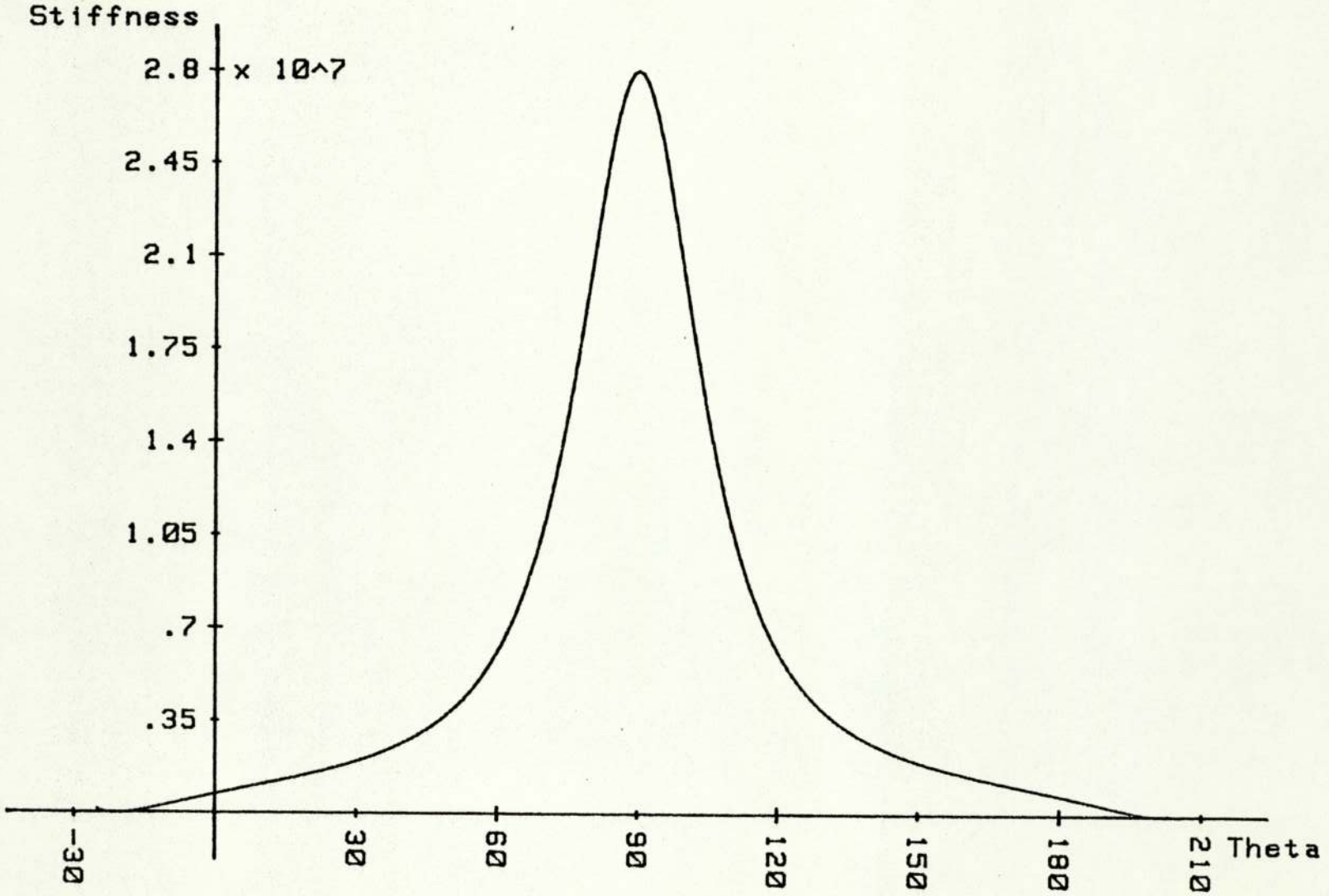
a = Length of the link OB

L = Length of the link OA (the boom)

S = Hydraulic stiffness of the ram

All the above quantities are defined with reference to Figure D.3.

This stiffness is then evaluated and plotted against the angle θ as defined in Figure D.4. (Graph D.1). The graph shows that the stiffness is higher when the boom is cutting the top-part of the profile. This is because of the assumption that the reaction force is acting in a tangential direction to the cutter, i.e. when the cutter is cutting the top-part of the profile, this force has a small vertical component and the horizontal force component is resisted by other means which are assumed "rigid" that can take up a large force. In this type of boom ripper, the hydraulic rams, which provide the rotational movement, have fairly constant stiffnesses, therefore the rotational stiffness hardly varies and has less effect on the variation of the boom's stiffness than the lifting trunks have.



Graph D.1 Boom Stiffness - Profile Angle (Theta) Relationship

D.2. EFFECT OF FEEDBACK ON STIFFNESS

The above two sections show that the effective stiffness of the system will depend on the hydraulic stiffness of the rams. Therefore any improvement in this hydraulic stiffness will improve the system stiffness. Analysis of a simple control system with a proportional position feedback will review this effect. Suppose that the hydraulic ram under consideration is a simple servo-mechanism with a proportional position feedback.

The stiffness of this ram

$$= \frac{1}{\omega}\sqrt{\frac{1}{2}\left(\frac{K_1}{A}\right)^2 + \left(\frac{\omega}{A}\right)^2\left(A - \frac{\omega^2 MV}{2\beta A}\right)^2} \quad S_o$$

where ω = Exciting frequency of disturbing force

A = Area of the ram piston

M = Inertial load

V = Volume of the oil column

β = Bulk modulus of the fluid in the ram

S_o = Hydraulic stiffness of the oil column in open-loop condition

$K_1 = C\ WK\sqrt{\frac{2P_s}{\rho}}$

C_D = Discharge coefficient of the valve

W = Width of the valve

K = Positional gain

P_s = System supply pressure

ρ = Fluid density

This stiffness will be increased to a high value as the exciting frequency decreases. If the position gain is increased, the stiffness can also be increased.

D.3. RIGIDITY (STABILITY OF MACHINE ON GROUND)

The machine is supported on ground by two simple supports 1 and 2 which are X_1 and X_2 apart from the vertical plane that contains the axis of slew (this plane faces the profile to be cut, Figure D.2).

The reactions and frictions of these supports can be estimated with reference to the position of the boom (Angle, θ, as defined in Figure D.4). The reactions will be the least when the angle is in the range of 90° to 110°, this is because that, in this range, the cutter is cutting horizontally. The force/reaction is maximum in this range, therefore the machine will have minimum resistance to slide movement, so sliding of the machine is likely to occur when the machine is cutting in this position. The force over reaction ratio can be reduced by making the supports as far apart as possible, therefore vertical supports at the far ends can help in stabilising the machine on the ground.

If the rock is too hard for the cutter, then a different story will result. Suppose that the ripper is cutting in a horizontal position and the rock is too hard, then there will be a grinding effect at the boom end and the large reaction at this point will cause the machine to slew.

It can be shown that the chance for the machine to be lifted in this situation is very small. Suppose that the boom in Figure D.2 is in a horizontal position and is pointing forward without any slew, and it cuts a hard rock which causes the rotation of the cutter as in Figure D.6.

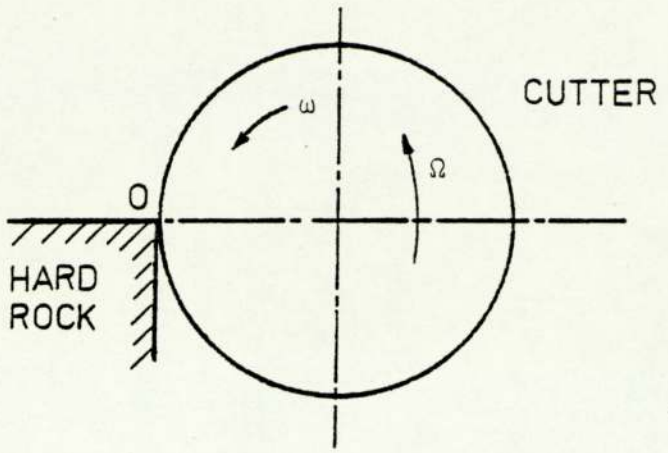


Figure D.6. Cutter cuts a Hard Rock and Stops

By considering the conservation of angular momentum and the impulse force created by the impact, the angular velocity of the machine about O (Fig. D.6) in the vertical plane was found to be in the order of 10^{-2} rad/sec. Therefore it is unlikely for the machine to lift off the ground in this situation.

D.4. DRILLING BOOM

The stiffness of the boom can be analysed in the same way as in Section D.0, but with more degrees of freedom, therefore it is not studied here. The stiffness of this machine will affect the machine performance in a different way from that of the boom-ripper. In this machine, the "static" stiffness will be more significant while the "dynamic" stiffness will be more significant in the boom-rippers. Hence for this kind of machine, the stiffness of the machine will contribute to holding the drill in position and is subject to very small disturbance only. The problem associated with this machine is the positioning of the drilling boom rather than keeping the drill in position while it is drilling.

APPENDIX E

Listing of the Assembly Programs for testing the time taken to evaluate the simplified switching function using different methods.

NAME OF PROGRAM SEGMENT	FUNCTION
PTEST	Driver for testing time used to evaluate the function in software.
MMTEST	Driver for testing the time used to evaluate the function in hardware.
PSWITCH	Routine to time and evaluate function in software with floating point arithmetic library (FPAL).
SWITCH	Routine to time and evaluate function in hardware without storing intermediate results using memory mapped arithmetic processing unit (M/M APU).
MAPSW	Routine to time and evaluate function in hardware with intermediate results stored using M/M APU.
ASWITCH	Routine to time and evaluate function in hardware without storing intermediate results using input or output mapped APU (I/O mapped APU).
APUSW	Routine to time and evaluate function in hardware with intermediate results stored by I/O mapped APU.
RDSND	Routines for data transfer between system memories and I/O mapped APU.
MRDSND	Routines for data transfer between system memories and M/M APU.

METHOD USED TO EVALUATE THE SIMPLIFIED SWITCHING FUNCTION	PROGRAM SEGMENTS TO BE COMBINED	LENGTH OF FINAL PROGRAM IN BYTES
Arithmetic processing in software using FPAL.	PTEST, PSWITCH, FPAL (Supplied by Intel Corporation).	1945 D.
I/O mapped APU with intermediate results stored	MMTEST, APUSW, RDSND.	260 D.
I/O mapped APU without storing of intermediate results.	MMTEST, ASWITCH, RDSND.	230 D.
M/M APU with intermediate results stored.	MMTEST, MAPSW, MRDSND.	244 D.
M/M APU without storing of intermediate results.	MMTEST, SWITCH, MRDSND.	214 D.

IDENTIFIER IN THE ASSEMBLY PROGRAM LISTING	NAME IN THE FLOW DIAGRAM FIGURE 4.1
COEFF3	λ_1
COEFF2	λ_2
COEFF1	λ_4
COEFFϕ	λ_5
COEFFD	λ_3
ACCEL	$\ddot{y}$
VELO	$\dot{y}$
POSN	y
DESIR	y_d
TEMP	Result
RESUL	Evaluated Switching Function Value.

```
ASM80 :F1:PTEST.SRC TITLE('TIMING TEST FOR FPAL TO EVALUATE SWITCHING FUNCTION')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         PTEST    PAGE    1
TIMING TEST FOR FPAL TO EVALUATE SWITCHING FUNCTION

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1 ;*************************************************************
                       2 ;*     A PROGRAMME TO FIND THE TIME TAKEN TO EVALUATE THE     *
                       3 ;*     SWITCHING FUNCTION BY FLOATING POINT ARITHMETIC LIBRARY *
                       4 ;*     SOFTWARE REQUIRED: PSWTCH.OBJ,FPAL.LIB,PLM80.LIB       *
                       5 ;*     HARDWARE REQUIRED: SDK-85 BOARD WITH SECOND 8155       *
                       6 ;*                        EXPANSION ROMs AND RAMs             *
                       7 ;*     WRITTEN BY: C W CHUEN                                  *
                       8 ;*     DATE:       1 JUNE 1982                                *
                       9 ;*************************************************************
                      10 NAME   PTEST
                      11 $INCLUDE(:F1:COEFF.PAL)
                    = 12 ;*************************************************
                    = 13 ;*     SEGMENT TO CONTAIN THE COEFFICIENTS AND    *
                    = 14 ;*     OTHER  RAM AREA  DEFINITIONS               *
                    = 15 ;*     TO BE USED WITH THE ROUTINE: PSWTCH        *
                    = 16 ;*************************************************
                    = 17 PUBLIC   COEFF3,COEFF2,COEFF1,COEFF0,COEFFD,ACCEL,VELO,POSN
                    = 18 PUBLIC   DESIR,TEMP,RESUL
                    = 19 ASEG
  2000              = 20 ORG      02000H                  ; LOCATE STARTING  RAM  ADDRESS
  2000 00           = 21 COEFF3:  DB     00H,00H,80H,3FH  ; COEFFICIENT FOR ACCELERATION TERM
  2001 00           =
  2002 80           =
  2003 3F           =
  2004 00           = 22 COEFF2:  DB     00H,00H,80H,3FH  ; FOR VELOCITY TERM
  2005 00           =
  2006 80           =
  2007 3F           =
  2008 00           = 23 COEFF1:  DB     00H,00H,80H,3FH  ; FOR DISPLACEMENT TERM
  2009 00           =
  200A 80           =
  200B 3F           =
  200C 00           = 24 COEFF0:  DB     00H,00H,80H,3FH  ; CONSTANT TERM
  200D 00           =
  200E 80           =
  200F 3F           =
  2010 00           = 25 COEFFD:  DB     00H,00H,80H,3FH  ; DESIRED POSITION TERM
  2011 00           =
  2012 80           =
  2013 3F           =
  2014 00           = 26 ACCEL:   DB     00H,00H,00H,40H  ; ACCELERATION
  2015 00           =
  2016 00           =
  2017 40           =
  2018 00           = 27 VELO:    DB     00H,00H,80H,40H  ; VELOCITY
  2019 00           =
  201A 80           =
  201B 40           =
  201C 00           = 28 POSN:    DB     00H,00H,00H,00H  ; CURRENT POSITION (INTEGER 00 - FFH)
  201D 00           =
  201E 00           =
  201F 00           =
  2020 00           = 29 DESIR:   DB     00H,00H,00H,00H  ; DESIRED POSITION (INTEGER 00 - FFH)
  2021 00           =
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

2022 00           =
2023 00           =
2024 00           =  30 TEMP:    DB      00H,00H,00H,00H  ; TEMPORARY STORAGE FOR INTERMEDIATE RESULT
2025 00           =
2026 00           =
2027 00           =
2028 00           =  31 RESUL:   DB      00H,00H,00H,00H  ; MEMORY STORAGE FOR FINAL RESULT
2029 00           =
202A 00           =
202B 00           =
                  =  32 ;
                  =  33 ;*************************************************
                  =  34 ;*       END OF SEGMENT                          *
                  =  35 ;*************************************************
                  =  36 ;
                     37 EXTRN  PSWTCH,FSET       ; STARTING ADDRESSES FOR SWITCH AND FPAL ROUTINES
                     38 PUBLIC FPR
C000                 39 STKPT  EQU     0C000H    ; ASSIGN STACK POINTER ADDRESS
                     40 $INCLUDE(:F1:SET.SR1)
                  =  41 ;*****************************************************************
                  =  42 ;*     A PROGRAMME SEGMENT FOR TIME TESTING PROGRAMMES           *
                  =  43 ;*     THIS HAS TO BE INSERTED IN THE BEGINNING OF THE           *
                  =  44 ;*     TEST PROGRAMME                                            *
                  =  45 ;*****************************************************************
202C              =  46 TIME:  DS   2             ; DEFINE STORAGE AREA FOR TIME COUN
202E              =  47 TIN:   DS   2             ; ORIGINAL COUNT VALUE
2030              =  48 HALFT: DS   2             ; ORIGINAL COUNT VALUE / 2
                  =  49 CSEG
0000 3100C0       =  50 START: LXI  SP,STKPT      ; LOCATE STACK POINTER TO STKPT
0003 3EFF         =  51        MVI  A,0FFH        ; INITIALIZE COUNTER IN 8155 FOR SINGLE SHOT MODE
0005 D32C         =  52        OUT  02CH          ; AND LOAD COUNTER WITH HIGHEST VALUE 3FFFH
0007 6F           =  53        MOV  L,A           ; TRANSFER TO H,L-REGISTER PAIR
0008 3EBF         =  54        MVI  A,0BFH
000A D32D         =  55        OUT  02DH
000C E63F         =  56        ANI  03FH          ; MASK OUT THE MODE BITS (6,7) AND CLEAR CARRY
000E 67           =  57        MOV  H,A
000F 222E20       =  58        SHLD TIN           ; STORE THIS INITIAL COUNT VALUE
0012 1F           =  59        RAR                ; HALFING INITIAL COUNT VALUE
0013 67           =  60        MOV  H,A
0014 7D           =  61        MOV  A,L
0015 1F           =  62        RAR
0016 6F           =  63        MOV  L,A
0017 223020       =  64        SHLD HALFT         ; STORE THIS VALUE
001A 0100B8          65        LXI  B,FPR         ; SET UP THE FLOATING POINT ROUTINES
001D C5              66        PUSH B
001E 010000          67        LXI  B,0
0021 CD0000   E      68        CALL FSET
0024 CD0000   E      69        CALL PSWTCH        ; CALL THE SWITCHING FUNCTION
                     70 $INCLUDE(:F1:SET.SR2)
                  =  71 ;*****************************************************************
                  =  72 ;*     A PROGRAMME TO EVALUATE THE TIME-CYCLE COUNT               *
                  =  73 ;*     WRITTEN BY: C W CHUEN                                     *
                  =  74 ;*     DATE:       09 MAY 1982                                   *
                  =  75 ;*****************************************************************
0027 DB2D         =  76 SET2:  IN   02DH          ; GET AND STORE HIGH ORDER BYTE OF TIMER VALUE
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         PTEST    PAGE    3
TIMING TEST FOR FPAL TO EVALUATE SWITCHING FUNCTION

  LOC  OBJ          LINE        SOURCE STATEMENT

  0029 E63F          =  77        ANI  03FH          ; MASK OUT THE TIMER MODE BITS 6,7 AND CLEAR CY
  002B 1F            =  78        RAR
  002C 47            =  79        MOV  B,A
  002D DB2C          =  80        IN   02CH          ; GET AND STORE LOW ORDER BYTE
  002F 1F            =  81        RAR
  0030 4F            =  82        MOV  C,A
  0031 D23A00     C  =  83        JNC  CONT          ; NO ADJUSTMENT IF NO CARRY
  0034 2A3020        =  84        LHLD HALFT         ; SEE P.6-24 OF MCS 80/85
  0037 09            =  85        DAD  B
  0038 44            =  86        MOV  B,H
  0039 4D            =  87        MOV  C,L
  003A 78            =  88 CONT:  MOV  A,B           ; EVALUATE ACTUAL CYCLE COUNT
  003B 2F            =  89        CMA
  003C 47            =  90        MOV  B,A
  003D 79            =  91        MOV  A,C
  003E 2F            =  92        CMA
  003F 4F            =  93        MOV  C,A
  0040 03            =  94        INX  B             ; TWO'S COMPLEMENT OF B,C
  0041 2A2E20        =  95        LHLD TIN           ; GET INITIAL COUNT VALUE
  0044 09            =  96        DAD  B             ; ACTUAL CYCLE COUNT EVALUATED AND
  0045 222C20        =  97        SHLD TIME          ; STORED
  0048 CF            =  98        RST  1             ; DISPLAY -8085
  B800               99 FPR       EQU  0B800H        ; DEFINE STORAGE FOR FLOATING POINT RECORD
  0000            C  100 END      START

PUBLIC SYMBOLS
ACCEL  A 2014    COEFF0 A 200C    COEFF1 A 2008    COEFF2 A 2004    COEFF3 A 2000    COEFFD A 2010    DESIR  A 2020
FPR    A B800    POSN   A 201C    RESUL  A 2028    TEMP   A 2024    VELO   A 2018

EXTERNAL SYMBOLS
FSET   E 0000    PSWTCH E 0000

USER SYMBOLS
ACCEL  A 2014    COEFF0 A 200C    COEFF1 A 2008    COEFF2 A 2004    COEFF3 A 2000    COEFFD A 2010    CONT   C 003A
DESIR  A 2020    FPR    A B800    FSET   E 0000    HALFT  A 2030    POSN   A 201C    PSWTCH E 0000    RESUL  A 2028
SET2   C 0027    START  C 0000    STKPT  A C000    TEMP   A 2024    TIME   A 202C    TIN    A 202E    VELO   A 2018

ASSEMBLY COMPLETE,    NO ERRORS
```

```
ASM80 :F1:MMTEST.SRC TITLE('TIMING TEST FOR APU TO EVALUATE SWITCHING FUNCTION')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MMTEST   PAGE    1
TIMING TEST FOR APU TO EVALUATE SWITCHING FUNCTION

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1 ;*************************************************************
                       2 ;*     A PROGRAMME TO FIND THE TIME TAKEN TO EVALUATE THE     *
                       3 ;*     SWITCHING FUNCTION BY THE MEMORY MAPPED APU - 8231A    *
                       4 ;*     SOFTWARE REQUIRED: MAPS2.OBJ                           *
                       5 ;*     HARDWARE REQUIRED: SDK-85 BOARD,APU BOARD WITH MEMORY  *
                       6 ;*                        ADDRESSES: 0FC00H - 0FDFFH          *
                       7 ;*     WRITTEN BY: C W CHUEN                                  *
                       8 ;*     DATE:       1 JUNE 1982                                *
                       9 ;*************************************************************
                      10 NAME   MMTEST
                      11 $INCLUDE(:F1:COEFF.APU)
                    = 12 ;*************************************************
                    = 13 ;*     SEGMENT TO CONTAIN THE COEFFICIENTS AND    *
                    = 14 ;*     OTHER  R A M  DEFINITIONS                  *
                    = 15 ;*     TO BE USED WITH THE ROUTINE: SWITCH        *
                    = 16 ;*************************************************
                    = 17 PUBLIC   COEFF3,COEFF2,COEFF1,COEFF0,COEFFD,ACCEL,VELO,POSN
                    = 18 PUBLIC   DESIR,TEMP,RESUL
                    = 19 ASEG
  2000              = 20 ORG      02000H                  ; LOCATE STARTING  RAM  ADDRESS
  2000 00           = 21 COEFF3:  DB     00H,00H,80H,01H  ; COEFFICIENT FOR ACCELERATION TERM
  2001 00           =
  2002 80           =
  2003 01           =
  2004 00           = 22 COEFF2:  DB     00H,00H,80H,01H  ; FOR VELOCITY TERM
  2005 00           =
  2006 80           =
  2007 01           =
  2008 00           = 23 COEFF1:  DB     00H,00H,80H,01H  ; FOR DISPLACEMENT TERM
  2009 00           =
  200A 80           =
  200B 01           =
  200C 00           = 24 COEFF0:  DB     00H,00H,80H,01H  ; CONSTANT TERM
  200D 00           =
  200E 80           =
  200F 01           =
  2010 00           = 25 COEFFD:  DB     00H,00H,80H,01H  ; DESIRED POSITON TERM
  2011 00           =
  2012 80           =
  2013 01           =
  2014 00           = 26 ACCEL:   DB     00H,00H,80H,02H  ; ACCELERATION
  2015 00           =
  2016 80           =
  2017 02           =
  2018 00           = 27 VELO:    DB     00H,00H,80H,03H  ; VELOCITY
  2019 00           =
  201A 80           =
  201B 03           =
  201C 00           = 28 POSN:    DB     00H,00H          ; CURRENT POSITION (INTEGER 00 - FFH)
  201D 00           =
  201E 00           = 29 DESIR:   DB     00H,00H          ; DESIRED POSITION (INTEGER 00 - FFH)
  201F 00           =
  2020 00           = 30 TEMP:    DB     00H,00H,00H,00H  ; TEMPORARY STORAGE FOR INTERMEDIATE RESULT
  2021 00           =
```

TIMING TEST FOR APU TO EVALUATE SWITCHING FUNCTION

```
LOC  OBJ          LINE        SOURCE STATEMENT

2022 00           =
2023 00           =
2024 00           =   31 RESUL:   DB     00H,00H,00H,00H  ; MEMORY STORAGE FOR FINAL RESULT
2025 00           =
2026 00           =
2027 00           =
                  =   32 ;
                  =   33 ;*************************************************
                  =   34 ;*       END OF SEGMENT                          *
                  =   35 ;*************************************************
                  =   36 ;
                      37 EXTRN  SWITCH            ; LOCATE SWITCH ROUTINE ADDRESS
20C2                  38 STKPT  EQU   020C2H      ; ASSIGN STACK POINTER ADDRESS
                      39 SET.SR1)
                  =   40 ;*************************************************************
                  =   41 ;*     A PROGRAMME SEGMENT FOR TIME TESTING PROGRAMMES       *
                  =   42 ;*     THIS HAS TO BE INSERTED IN THE BEGINNING OF THE       *
                  =   43 ;*     TEST PROGRAMME                                        *
                  =   44 ;*************************************************************
2028              =   45 TIME:  DS   2            ; DEFINE STORAGE AREA FOR TIME COUN
202A              =   46 TIN:   DS   2            ; ORIGINAL COUNT VALUE
202C              =   47 HALFT: DS   2            ; ORIGINAL COUNT VALUE / 2
                  =   48 CSEG
0000 31C220       =   49 START: LXI  SP,STKPT     ; LOCATE STACK POINTER TO STKPT
0003 3EFF         =   50        MVI  A,0FFH       ; INITIALIZE COUNTER IN 8155 FOR SINGLE SHOT MODE
0005 D32C         =   51        OUT  02CH         ; AND LOAD COUNTER WITH HIGHEST VALUE 3FFFH
0007 6F           =   52        MOV  L,A          ; TRANSFER TO H,L-REGISTER PAIR
0008 3EBF         =   53        MVI  A,0BFH
000A D32D         =   54        OUT  02DH
000C E63F         =   55        ANI  03FH         ; MASK OUT THE MODE BITS (6,7) AND CLEAR CARRY
000E 67           =   56        MOV  H,A
000F 222A20       =   57        SHLD TIN          ; STORE THIS INITIAL COUNT VALUE
0012 1F           =   58        RAR               ; HALFING INITIAL COUNT VALUE
0013 67           =   59        MOV  H,A
0014 7D           =   60        MOV  A,L
0015 1F           =   61        RAR
0016 6F           =   62        MOV  L,A
0017 222C20       =   63        SHLD HALFT        ; STORE THIS VALUE
001A CD0000   E   =   64        CALL SWITCH       ; CALL THE SWITCHING FUNCTION
                      65 $INCLUDE(:F1:SET.SR2)
                  =   66 ;*************************************************************
                  =   67 ;*     A PROGRAMME TO EVALUATE THE TIME-CYCLE COUNT           *
                  =   68 ;*     WRITTEN BY: C W CHUEN                                 *
                  =   69 ;*     DATE:       09 MAY 1982                               *
                  =   70 ;*************************************************************
001D DB2D         =   71 SET2:  IN   02DH         ; GET AND STORE HIGH ORDER BYTE OF TIMER VALUE
001F E63F         =   72        ANI  03FH         ; MASK OUT THE TIMER MODE BITS 6,7 AND CLEAR CY
0021 1F           =   73        RAR
0022 47           =   74        MOV  B,A
0023 DB2C         =   75        IN   02CH         ; GET AND STORE LOW ORDER BYTE
0025 1F           =   76        RAR
0026 4F           =   77        MOV  C,A
0027 D23000   C   =   78        JNC  CONT         ; NO ADJUSTMENT IF NO CARRY
002A 2A2C20       =   79        LHLD HALFT        ; SEE P.6-24 OF MCS 80/85
002D 09           =   80        DAD  B
```

TIMING TEST FOR APU TO EVALUATE SWITCHING FUNCTION

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 002E 44           =  81        MOV  B,H
 002F 4D           =  82        MOV  C,L
 0030 78           =  83 CONT:  MOV  A,B          ; EVALUATE ACTUAL CYCLE COUNT
 0031 2F           =  84        CMA
 0032 47           =  85        MOV  B,A
 0033 79           =  86        MOV  A,C
 0034 2F           =  87        CMA
 0035 4F           =  88        MOV  C,A
 0036 03           =  89        INX  B            ; TWO'S COMPLEMENT OF B,C
 0037 2A2A20       =  90        LHLD TIN          ; GET INITIAL COUNT VALUE
 003A 09           =  91        DAD  B            ; ACTUAL CYCLE COUNT EVALUATED AND
 003B 222820       =  92        SHLD TIME         ; STORED
 003E CF           =  93        RST  1            ; DISPLAY -8085
 0000           C     94 END     START
```

PUBLIC SYMBOLS

ACCEL	A 2014	COEFF0	A 200C	COEFF1	A 2008	COEFF2	A 2004	COEFF3	A 2000	COEFFD	A 2010	DESIR	A 201E
POSN	A 201C	RESUL	A 2024	TEMP	A 2020	VELO	A 2018						

EXTERNAL SYMBOLS

SWITCH	E 0000										

USER SYMBOLS

ACCEL	A 2014	COEFF0	A 200C	COEFF1	A 2008	COEFF2	A 2004	COEFF3	A 2000	COEFFD	A 2010	CONT	C 0030
DESIR	A 201E	HALFT	A 202C	POSN	A 201C	RESUL	A 2024	SET2	C 001D	START	C 0000	STKPT	A 20C2
SWITCH	E 0000	TEMP	A 2020	TIME	A 2028	TIN	A 202A	VELO	A 2018				

ASSEMBLY COMPLETE, NO ERRORS

```
ASM80 :F1:PSWTCH.SRC TITLE('FPAL EVALUATION OF SWITCHING FUNCTION')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         PSWTCH   PAGE    1
FPAL EVALUATION OF SWITCHING FUNCTION

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1 ;*************************************************************
                      2 ;*     A ROUTINE TO EVALUATE THE SWITCHING FUNCTION BASED ON     *
                      3 ;*     ACCEL,VEL,POSITION AND OTHER COEFFICIENTS USING THE       *
                      4 ;*     FLOATING POINT ARITHMETIC LIBRARY  (FPAL.LIB)             *
                      5 ;*                                                               *
                      6 ;*     SWITCHING FUNCTION = COEFF3*ACCEL + COEFF2*VELO +         *
                      7 ;*                          COEFF1*POSN + COEFF0*DESIR + COEFF0  *
                      8 ;*                                                               *
                      9 ;*          DESIR AND POSN ARE SINGLE PRECISION INTEGERS (2-BYTE) *
                     10 ;*          OTHERS ARE FLOATING POINT NUMBERS  (BFP NOS.)         *
                     11 ;*                                                               *
                     12 ;*     SOFTWARE REQUIRED: FPAL.LIB,PLM80.LIB,SYSTEM.LIB          *
                     13 ;*     HARDWARE REQUIRED: SDK-85,EXPANSION ROMs AND RAMs         *
                     14 ;*                                                               *
                     15 ;*     WRITTEN BY: C W CHUEN                                     *
                     16 ;*     DATE:       01 JUNE 1982                                  *
                     17 ;*************************************************************
                     18 NAME     PSWTCH
                     19 EXTRN    FPR,FLOAD,FMUL,FADD,FLTDS,FSTOR,FSUB
                     20 EXTRN    COEFF3,COEFF2,COEFF1,COEFF0,COEFFD
                     21 EXTRN    ACCEL,VELO,POSN,DESIR,TEMP,RESUL
                     22 PUBLIC   PSWTCH
                     23 CSEG
0000 F5              24 PSWTCH:  PUSH     PSW       ; SAVE REGISTERS
0001 E5              25          PUSH     H
0002 C5              26          PUSH     B
0003 3EC0            27          MVI      A,0C0H    ; START THE COUNT DOWN MODE FOR TIMER
0005 D328            28          OUT      028H
0007 010000    E     29          LXI      B,FPR     ; LOCATE FLOATING POINT RECORD
000A 110000    E     30          LXI      D,DESIR   ; CONVERT DESIR INTO
000D CD0000    E     31          CALL     FLTDS     ; BINARY FLOATING POINT
0010 110000    E     32          LXI      D,COEFFD  ;
0013 CD0000    E     33          CALL     FMUL      ; COEFFD*DESIR
0016 110000    E     34          LXI      D,RESUL
0019 CD0000    E     35          CALL     FSTOR     ; <----
001C 110000    E     36          LXI      D,POSN    ; CONVERT POSN INTO
001F CD0000    E     37          CALL     FLTDS     ; BINARY FLOATING POINT FORMAT
0022 110000    E     38          LXI      D,COEFF1
0025 CD0000    E     39          CALL     FMUL      ; *
0028 110000    E     40          LXI      D,TEMP    ; COEFF1*POSN
002B CD0000    E     41          CALL     FSTOR     ; <---
002E 110000    E     42          LXI      D,COEFF3  ; COEFF3
0031 CD0000    E     43          CALL     FLOAD     ; ^
0034 110000    E     44          LXI      D,ACCEL   ; ACCEL
0037 CD0000    E     45          CALL     FMUL      ; COEFF3*ACCEL
003A 110000    E     46          LXI      D,TEMP
003D CD0000    E     47          CALL     FADD      ; +
0040 CD0000    E     48          CALL     FSTOR     ; <---
0043 110000    E     49          LXI      D,COEFF2  ; COEFF2
0046 CD0000    E     50          CALL     FLOAD     ; ^
0049 110000    E     51          LXI      D,VELO    ; VELO
004C CD0000    E     52          CALL     FMUL      ; COEFF2*VELO
004F 110000    E     53          LXI      D,TEMP
0052 CD0000    E     54          CALL     FADD      ; +
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0          PSWTCH   PAGE    2
FPAL EVALUATION OF SWITCHING FUNCTION

  LOC  OBJ          LINE        SOURCE STATEMENT

  0055 110000   E     55          LXI       D,COEFF0 ; COEFF0
  0058 CD0000   E     56          CALL      FADD     ; +
  005B 110000   E     57          LXI       D,RESUL
  005E CD0000   E     58          CALL      FADD     ; DESIR ^
  0061 CD0000   E     59          CALL      FSTOR    ; <----
  0064 3E40           60          MVI       A,040H   ; STOP TIMER
  0066 D328           61          OUT       028H
  0068 C1             62          POP       B        ; AND RECOVER REGISTERS SAVED
  0069 E1             63          POP       H        ; BEFORE RETURN TO THE CALLING PROGRAMME
  006A F1             64          POP       PSW
  006B C9             65          RET
                      66 ;
                      67 END


PUBLIC SYMBOLS
PSWTCH C 0000

EXTERNAL SYMBOLS
ACCEL  E 0000    COEFF0 E 0000    COEFF1 E 0000    COEFF2 E 0000    COEFF3 E 0000    COEFFD E 0000    DESIR  E 0000
FADD   E 0000    FLOAD  E 0000    FLTDS  E 0000    FMUL   E 0000    FPR    E 0000    FSTOR  E 0000    FSUB   E 0000
POSN   E 0000    RESUL  E 0000    TEMP   E 0000    VELO   E 0000

USER SYMBOLS
ACCEL  E 0000    COEFF0 E 0000    COEFF1 E 0000    COEFF2 E 0000    COEFF3 E 0000    COEFFD E 0000    DESIR  E 0000
FADD   E 0000    FLOAD  E 0000    FLTDS  E 0000    FMUL   E 0000    FPR    E 0000    FSTOR  E 0000    FSUB   E 0000
POSN   E 0000    PSWTCH C 0000    RESUL  E 0000    TEMP   E 0000    VELO   E 0000

ASSEMBLY COMPLETE,    NO ERRORS
```

```
ASM80 :F1:SWITCH.SRC TITLE('SWITCHING FUNCTION FOR M/M APU')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         SWITCH   PAGE    1
SWITCHING FUNCTION FOR M/M APU

 LOC  OBJ          LINE        SOURCE STATEMENT

                      1 ;*************************************************************************
                      2 ;*     A ROUTINE TO CALCULATE THE SWITCHING FUNCTION BASED ON ACCEL,     *
                      3 ;*     VEL, POSITION AND OTHER COEFFICIENTS USING THE APU 8231A WITH     *
                      4 ;*     MEMORY MAPPING TECHNIQUE                                          *
                      5 ;*     TEMPORARY STORING OF INTERMEDIATE RESULTS ARE OMITTED             *
                      6 ;*                                                                       *
                      7 ;*     SWITCHING FUNCTION = COEFF3*ACCEL + COEFF2*VELO + COEFF1*POSN     *
                      8 ;*                        + COEFF0 + COEFFD*DESIR                        *
                      9 ;*                                                                       *
                     10 ;*     WRITTEN BY: C W CHUEN                                             *
                     11 ;*     DATE:       1 JUNE 1982                                           *
                     12 ;*************************************************************************
                     13 NAME     SWITCH
                     14 $INCLUDE(:F1:APUCMD.ASM)
                   = 15 $NOLIST
                   = 66 $LIST
                     67 EXTRN    SEND2,SEND4,READ2,READ4
                     68 EXTRN    COEFF3,COEFF2,COEFF1,COEFF0,COEFFD
                     69 EXTRN    ACCEL,VELO,POSN,DESIR,TEMP,RESUL
                     70 PUBLIC   SWITCH
                     71 CSEG
0000 F5              72 SWITCH:  PUSH     PSW       ; SAVE REGISTERS
0001 E5              73          PUSH     H
0002 C5              74          PUSH     B
0003 3EC0            75          MVI      A,0C0H    ; START THE COUNT DOWN MODE FOR TIMER
0005 D328            76          OUT      028H
0007 010000    E     77          LXI      B,DESIR   ; CONVERT DESIR FROM 2-BYTE INTEGER INTO
000A CD0000    E     78          CALL     SEND2     ; 4-BYTE BINARY FLOATING POINT FORMAT
000D 26FD            79          MVI      H,APUC    ; LOAD H WITH APU COMMAND PORT ADDRESS
000F 361D            80          MVI      M,FLTS    ; ISSUE THE CONVERSION COMMAND TO APU
0011 010000    E     81          LXI      B,COEFFD  ; LOAD APU DATA STACK WITH COEFFD
0014 CD0000    E     82          CALL     SEND4
0017 3612            83          MVI      M,FMUL    ; COEFFD*DESIR
0019 010000    E     84          LXI      B,POSN    ; CONVERT POSN FROM 2-BYTE INTEGER INTO
001C CD0000    E     85          CALL     SEND2     ; 4-BYTE BINARY FLOATING POINT FORMAT
001F 361D            86          MVI      M,FLTS    ; RECOGNIZED BY APU
0021 010000    E     87          LXI      B,COEFF1  ; LOAD APU DATA STACK WITH COEFF1
0024 CD0000    E     88          CALL     SEND4
0027 3612            89          MVI      M,FMUL    ; COEFF1*POSN
0029 3610            90          MVI      M,FADD    ; COEFFD*DESIR + COEFF1*POSN
002B 010000    E     91          LXI      B,COEFF3  ; COEFF3
002E CD0000    E     92          CALL     SEND4     ; ^
0031 010000    E     93          LXI      B,ACCEL   ; ACCEL
0034 CD0000    E     94          CALL     SEND4
0037 3612            95          MVI      M,FMUL    ; COEFF3*ACCEL
0039 3610            96          MVI      M,FADD    ; COEFF3*ACCEL + COEFF1*POSN + COEFFD*DESIR
003B 010000    E     97          LXI      B,COEFF2  ; COEFF2
003E CD0000    E     98          CALL     SEND4     ; ^
0041 010000    E     99          LXI      B,VELO    ; VELO
0044 CD0000    E    100          CALL     SEND4
0047 3612            101         MVI      M,FMUL    ; COEFF2*VELO
0049 3610            102         MVI      M,FADD    ; +
004B 010000    E     103         LXI      B,COEFF0  ; COEFF0
004E CD0000    E     104         CALL     SEND4     ; ^
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        SWITCH   PAGE    2
SWITCHING FUNCTION FOR M/M APU

  LOC  OBJ          LINE        SOURCE STATEMENT

  0051 3610           105           MVI     M,FADD  ; +
  0053 010000   E     106           LXI     B,RESUL
  0056 CD0000   E     107           CALL    READ4   ; <----
  0059 3E40           108           MVI     A,040H  ; STOP TIMER BEFOR RETURN TO
  005B D328           109           OUT     028H    ; THE CALLING PROGRAMME
  005D C1             110           POP     B       ; RECOVER REGISTERS SAVED
  005E E1             111           POP     H
  005F F1             112           POP     PSW
  0060 C9             113           RET
                      114 ;
                      115 END


PUBLIC SYMBOLS
SWITCH C 0000

EXTERNAL SYMBOLS
ACCEL   E 0000    COEFF0  E 0000    COEFF1  E 0000    COEFF2  E 0000    COEFF3  E 0000    COEFFD  E 0000    DESIR   E 0000
POSN    E 0000    READ2   E 0000    READ4   E 0000    RESUL   E 0000    SEND2   E 0000    SEND4   E 0000    TEMP    E 0000
VELO    E 0000

USER SYMBOLS
ACCEL   E 0000    ACOS    A 0006    APUC    A 00FD    APUD    A 00FC    ASIN    A 0005    ATAN    A 0007    CHSD    A 0034
CHSF    A 0015    CHSS    A 0074    COEFF0  E 0000    COEFF1  E 0000    COEFF2  E 0000    COEFF3  E 0000    COEFFD  E 0000
COS     A 0003    DADD    A 002C    DDIV    A 002F    DESIR   E 0000    DMUL    A 002E    DMUU    A 0036    DSUB    A 002D
EXP     A 000A    FADD    A 0010    FDIV    A 0013    FIXD    A 001E    FIXS    A 001F    FLTD    A 001C    FLTS    A 001D
FMUL    A 0012    FSUB    A 0011    LN      A 0009    LOG     A 0008    POPD    A 0038    POPF    A 0018    POPS    A 0078
POSN    E 0000    PTOD    A 0037    PTOF    A 0017    PTOS    A 0077    PUPI    A 001A    PWR     A 000B    READ2   E 0000
READ4   E 0000    RESUL   E 0000    SADD    A 006C    SDIV    A 006F    SEND2   E 0000    SEND4   E 0000    SIN     A 0002
SMUL    A 006E    SMUU    A 0076    SQRT    A 0001    SSUB    A 006D    SWITCH  C 0000    TAN     A 0004    TEMP    E 0000
VELO    E 0000    XCHD    A 0039    XCHF    A 0019    XCHS    A 0079

ASSEMBLY COMPLETE,   NO ERRORS
```

```
ASM80 :F1:MAPSW.SRC TITLE( 'ROUTINE FOR M/M APU TO EVALUATE SWITCHING FUNCTION')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         MAPSW    PAGE    1
ROUTINE FOR M/M APU TO EVALUATE SWITCHING FUNCTION

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1 ;*****************************************************************
                      2 ;*     A ROUTINE TO CALCULATE THE SWITCHING FUNCTION BASED ON ACCEL,   *
                      3 ;*     VEL, POSITION AND OTHER COEFFICIENTS USING THE APU 8231A WITH   *
                      4 ;*     MEMORY MAPPING TECHNIQUE                                        *
                      5 ;*                                                                     *
                      6 ;*     SWITCHING FUNCTION = COEFF3*ACCEL + COEFF2*VELO + COEFF1*POSN   *
                      7 ;*                          + COEFF0 + COEFFD*DESIR                    *
                      8 ;*                                                                     *
                      9 ;*     WRITTEN BY: C W CHUEN                                           *
                     10 ;*     DATE:       1 JUNE 1982                                         *
                     11 ;*****************************************************************
                     12 NAME     MAPSW
                     13 $INCLUDE(:F1:APUCMD.ASM)
                   = 14 $NOLIST
                   = 65 $LIST
                     66 EXTRN    SEND2,SEND4,READ2,READ4
                     67 EXTRN    COEFF3,COEFF2,COEFF1,COEFF0,COEFFD
                     68 EXTRN    ACCEL,VELO,POSN,DESIR,TEMP,RESUL
                     69 PUBLIC   SWITCH
                     70 CSEC
0000 F5              71 SWITCH:  PUSH    PSW       ; SAVE REGISTERS
0001 E5              72          PUSH    H
0002 C5              73          PUSH    B
0003 3EC0            74          MVI     A,0C0H    ; START THE COUNT DOWN MODE FOR TIMER
0005 D328            75          OUT     028H
0007 010000    E     76          LXI     B,DESIR   ; CONVERT DESIR FROM 2-BYTE INTEGER INTO
000A CD0000    E     77          CALL    SEND2     ; 4-BYTE BINARY FLOATING POINT FORMAT
000D 26FD            78          MVI     H,APUC    ; LOAD H WITH APU COMMAND PORT ADDRESS
000F 361D            79          MVI     M,FLTS    ; ISSUE THE CONVERSION COMMAND TO APU
0011 010000    E     80          LXI     B,COEFFD  ; LOAD APU DATA STACK WITH COEFFD
0014 CD0000    E     81          CALL    SEND4
0017 3612            82          MVI     M,FMUL    ; *
0019 010000    E     83          LXI     B,RESUL   ; STORE  COEFFD*DESIR
001C CD0000    E     84          CALL    READ4
001F 010000    E     85          LXI     B,POSN    ; CONVERT POSN FROM 2-BYTE INTEGER INTO
0022 CD0000    E     86          CALL    SEND2     ; 4-BYTE BINARY FLOATING POINT FORMAT
0025 361D            87          MVI     M,FLTS    ; RECOGNIZED BY APU
0027 010000    E     88          LXI     B,COEFF1  ; LOAD APU DATA STACK WITH COEFF1
002A CD0000    E     89          CALL    SEND4
002D 3612            90          MVI     M,FMUL    ; *
002F 010000    E     91          LXI     B,TEMP    ; COEFF1*POSN
0032 CD0000    E     92          CALL    READ4     ; <---
0035 010000    E     93          LXI     B,COEFF3  ; COEFF3
0038 CD0000    E     94          CALL    SEND4     ; ^
003B 010000    E     95          LXI     B,ACCEL   ; ACCEL
003E CD0000    E     96          CALL    SEND4
0041 3612            97          MVI     M,FMUL    ; COEFF3*ACCEL
0043 010000    E     98          LXI     B,TEMP
0046 CD0000    E     99          CALL    SEND4
0049 3610           100          MVI     M,FADD    ; +
004B CD0000    E    101          CALL    READ4     ; <---
004E 010000    E    102          LXI     B,COEFF2  ; COEFF2
0051 CD0000    E    103          CALL    SEND4     ; ^
0054 010000    E    104          LXI     B,VELO    ; VELO
```

ROUTINE FOR M/M APU TO EVALUATE SWITCHING FUNCTION

```
LOC  OBJ          LINE      SOURCE STATEMENT

0057 CD0000   E    105        CALL    SEND4
005A 3612          106        MVI     M,FMUL    ; COEFF2*VELO
005C 010000   E    107        LXI     B,TEMP
005F CD0000   E    108        CALL    SEND4
0062 3610          109        MVI     M,FADD    ; +
0064 010000   E    110        LXI     B,COEFF0  ; COEFF0
0067 CD0000   E    111        CALL    SEND4     ; ^
006A 3610          112        MVI     M,FADD    ; +
006C 010000   E    113        LXI     B,RESUL
006F CD0000   E    114        CALL    SEND4     ; COEFFD*DESIR ^
0072 3610          115        MVI     M,FADD    ; +
0074 CD0000   E    116        CALL    READ4     ; <----
0077 3E40          117        MVI     A,040H    ; STOP TIMER BEFOR RETURN TO
0079 D328          118        OUT     028H      ; THE CALLING PROGRAMME
007B C1            119        POP     B         ; RECOVER REGISTERS SAVED
007C E1            120        POP     H
007D F1            121        POP     PSW
007E C9            122        RET
                   123 ;
                   124 END
```

PUBLIC SYMBOLS
SWITCH C 0000

EXTERNAL SYMBOLS

ACCEL	E 0000	COEFF0	E 0000	COEFF1	E 0000	COEFF2	E 0000	COEFF3	E 0000	COEFFD	E 0000	DESIR	E 0000
POSN	E 0000	READ2	E 0000	READ4	E 0000	RESUL	E 0000	SEND2	E 0000	SEND4	E 0000	TEMP	E 0000
VELO	E 0000												

USER SYMBOLS

ACCEL	E 0000	ACOS	A 0006	APUC	A 00FD	APUD	A 00FC	ASIN	A 0005	ATAN	A 0007	CHSD	A 0034
CHSF	A 0015	CHSS	A 0074	COEFF0	E 0000	COEFF1	E 0000	COEFF2	E 0000	COEFF3	E 0000	COEFFD	E 0000
COS	A 0003	DADD	A 002C	DDIV	A 002F	DESIR	E 0000	DMUL	A 002E	DMUU	A 0036	DSUB	A 002D
EXP	A 000A	FADD	A 0010	FDIV	A 0013	FIXD	A 001E	FIXS	A 001F	FLTD	A 001C	FLTS	A 001D
FMUL	A 0012	FSUB	A 0011	LN	A 0009	LOG	A 0008	POPD	A 0038	POPF	A 0018	POPS	A 0078
POSN	E 0000	PTOD	A 0037	PTOF	A 0017	PTOS	A 0077	PUPI	A 001A	PWR	A 000B	READ2	E 0000
READ4	E 0000	RESUL	E 0000	SADD	A 006C	SDIV	A 006F	SEND2	E 0000	SEND4	E 0000	SIN	A 0002
SMUL	A 006E	SMUU	A 0076	SQRT	A 0001	SSUB	A 006D	SWITCH	C 0000	TAN	A 0004	TEMP	E 0000
VELO	E 0000	XCHD	A 0039	XCHF	A 0019	XCHS	A 0079						

ASSEMBLY COMPLETE, NO ERRORS

```
ASM80 :F1:ASWTCH.SRC TITLE('ROUTINE FOR IO APU TO EVALUATE SWITCHING FUNCTION')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         ASWTCH   PAGE    1
ROUTINE FOR IO APU TO EVALUATE SWITCHING FUNCTION

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1 ;*****************************************************************
                      2 ;*     A ROUTINE TO CALCULATE THE SWITCHING FUNCTION BASED ON ACCEL,    *
                      3 ;*     VEL, POSITION AND OTHER COEFFICIENTS USING THE APU 8231A WITH    *
                      4 ;*     I/O MAPPING TECHNIQUE                                           *
                      5 ;*     TEMPORARY STORING OF INTERMEDIATE RESULTS ARE OMITTED           *
                      6 ;*                                                                     *
                      7 ;*     SWITCHING FUNCTION = COEFF3*ACCEL + COEFF2*VELO + COEFF1*POSN   *
                      8 ;*                          + COEFF0 + COEFFD*DESIR                    *
                      9 ;*                                                                     *
                     10 ;*     WRITTEN BY: C W CHUEN                                           *
                     11 ;*     DATE:       1 JUNE 1982                                         *
                     12 ;*****************************************************************
                     13 NAME     ASWTCH
                     14 $INCLUDE(:F1:APUCMD.ASM)
                =    15 $NOLIST
                =    66 $LIST
                     67 EXTRN    SEND2,SEND4,READ2,READ4
                     68 EXTRN    COEFF3,COEFF2,COEFF1,COEFF0,COEFFD
                     69 EXTRN    ACCEL,VELO,POSN,DESIR,TEMP,RESUL
                     70 PUBLIC   SWITCH
                     71 CSEG
0000 F5              72 SWITCH:  PUSH     PSW       ; SAVE REGISTERS
0001 E5              73          PUSH     H
0002 C5              74          PUSH     B
0003 3EC0            75          MVI      A,0C0H    ; START THE COUNT DOWN MODE FOR TIMER
0005 D328            76          OUT      028H
0007 010000    E     77          LXI      B,DESIR   ; CONVERT DESIR FROM 2-BYTE INTEGER INTO
000A CD0000    E     78          CALL     SEND2     ; 4-BYTE BINARY FLOATING POINT FORMAT
000D 3E1D            79          MVI      A,FLTS    ; ISSUE THE CONVERSION COMMAND TO APU
000F D3FD            80          OUT      APUC      ; COMMAND PORT
0011 010000    E     81          LXI      B,COEFFD  ; LOAD APU DATA STACK WITH COEFFD
0014 CD0000    E     82          CALL     SEND4
0017 3E12            83          MVI      A,FMUL    ; COEFFD*DESIR
0019 D3FD            84          OUT      APUC
001B 010000    E     85          LXI      B,POSN    ; CONVERT POSN FROM 2-BYTE INTEGER INTO
001E CD0000    E     86          CALL     SEND2     ; 4-BYTE BINARY FLOATING POINT FORMAT
0021 3E1D            87          MVI      A,FLTS    ; RECOGNIZED BY APU
0023 D3FD            88          OUT      APUC
0025 010000    E     89          LXI      B,COEFF1  ; LOAD APU DATA STACK WITH COEFF1
0028 CD0000    E     90          CALL     SEND4
002B 3E12            91          MVI      A,FMUL    ; COEFF1*POSN
002D D3FD            92          OUT      APUC
002F 3E10            93          MVI      A,FADD    ; COEFFD*DESIR + COEFF1*POSN
0031 D3FD            94          OUT      APUC
0033 010000    E     95          LXI      B,COEFF3  ; COEFF3
0036 CD0000    E     96          CALL     SEND4     ; ^
0039 010000    E     97          LXI      B,ACCEL   ; ACCEL
003C CD0000    E     98          CALL     SEND4
003F 3E12            99          MVI      A,FMUL    ; COEFF3*ACCEL
0041 D3FD           100          OUT      APUC
0043 3E10           101          MVI      A,FADD    ; COEFF3*ACCEL + COEFF1*POSN + COEFFD*DESIR
0045 D3FD           102          OUT      APUC
0047 010000    E    103          LXI      B,COEFF2  ; COEFF2
004A CD0000    E    104          CALL     SEND4     ; ^
```

ROUTINE FOR IO APU TO EVALUATE SWITCHING FUNCTION

```
LOC  OBJ          LINE        SOURCE STATEMENT

004D 010000   E    105          LXI     B,VELO    ; VELO
0050 CD0000   E    106          CALL    SEND4
0053 3E12          107          MVI     A,FMUL    ; COEFF2*VELO
0055 D3FD          108          OUT     APUC
0057 3E10          109          MVI     A,FADD    ; +
0059 D3FD          110          OUT     APUC
005B 010000   E    111          LXI     B,COEFF0  ; COEFF0
005E CD0000   E    112          CALL    SEND4     ; ^
0061 3E10          113          MVI     A,FADD    ; +
0063 D3FD          114          OUT     APUC
0065 010000   E    115          LXI     B,RESUL
0068 CD0000   E    116          CALL    READ4     ; <----
006B 3E40          117          MVI     A,040H    ; STOP TIMER BEFOR RETURN TO
006D D328          118          OUT     028H      ; THE CALLING PROGRAMME
006F C1            119          POP     B         ; RECOVER REGISTERS SAVED
0070 E1            120          POP     H
0071 F1            121          POP     PSW
0072 C9            122          RET
                   123 ;
                   124 END
```

PUBLIC SYMBOLS
SWITCH C 0000

EXTERNAL SYMBOLS

ACCEL	E 0000	COEFF0	E 0000	COEFF1	E 0000	COEFF2	E 0000	COEFF3	E 0000	COEFFD	E 0000	DESIR	E 0000
POSN	E 0000	READ2	E 0000	READ4	E 0000	RESUL	E 0000	SEND2	E 0000	SEND4	E 0000	TEMP	E 0000
VELO	E 0000												

USER SYMBOLS

ACCEL	E 0000	ACOS	A 0006	APUC	A 00FD	APUD	A 00FC	ASIN	A 0005	ATAN	A 0007	CHSD	A 0034
CHSF	A 0015	CHSS	A 0074	COEFF0	E 0000	COEFF1	E 0000	COEFF2	E 0000	COEFF3	E 0000	COEFFD	E 0000
COS	A 0003	DADD	A 002C	DDIV	A 002F	DESIR	E 0000	DMUL	A 002E	DMUU	A 0036	DSUB	A 002D
EXP	A 000A	FADD	A 0010	FDIV	A 0013	FIXD	A 001E	FIXS	A 001F	FLTD	A 001C	FLTS	A 001D
FMUL	A 0012	FSUB	A 0011	LN	A 0009	LOG	A 0008	POPD	A 0038	POPF	A 0018	POPS	A 0078
POSN	E 0000	PTOD	A 0037	PTOF	A 0017	PTOS	A 0077	PUPI	A 001A	PWR	A 000B	READ2	E 0000
READ4	E 0000	RESUL	E 0000	SADD	A 006C	SDIV	A 006F	SEND2	E 0000	SEND4	E 0000	SIN	A 0002
SMUL	A 006E	SMUU	A 0076	SQRT	A 0001	SSUB	A 006D	SWITCH	C 0000	TAN	A 0004	TEMP	E 0000
VELO	E 0000	XCHD	A 0039	XCHF	A 0019	XCHS	A 0079						

ASSEMBLY COMPLETE, NO ERRORS

```
ASM80 :F1:APUSW.SRC TITLE('ROUTINE TO EVALUATE SWITCHING FUNCTION USING APU')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0          APUSW     PAGE     1
ROUTINE TO EVALUATE SWITCHING FUNCTION USING APU

 LOC  OBJ          LINE        SOURCE STATEMENT

                      1 ;*************************************************************
                      2 ;*      A ROUTINE TO CALCULATE THE SWITCHING FUNCTION BASED ON ACCEL,    *
                      3 ;*      VEL, POSITION AND OTHER COEFFICIENTS USING THE APU 8231A WITH    *
                      4 ;*      MEMORY MAPPING TECHNIQUE                                         *
                      5 ;*                                                                       *
                      6 ;*      SWITCHING FUNCTION = COEFF3*ACCEL + COEFF2*VELO + COEFF1*POSN    *
                      7 ;*                           + COEFF0 + COEFFD*DESIR                     *
                      8 ;*                                                                       *
                      9 ;*      WRITTEN BY: C W CHUEN                                            *
                     10 ;*      DATE:       1 JUNE 1982                                          *
                     11 ;*************************************************************
                     12 NAME     APUSW
                     13 $INCLUDE(:F1:APUCMD.ASM)
                  =  14 $NOLIST
                  =  65 $LIST
                     66 EXTRN    SEND2,SEND4,READ2,READ4
                     67 EXTRN    COEFF3,COEFF2,COEFF1,COEFF0,COEFFD
                     68 EXTRN    ACCEL,VELO,POSN,DESIR,TEMP,RESUL
                     69 PUBLIC   SWITCH
                     70 CSEG
0000 F5              71 SWITCH:  PUSH    PSW       ; SAVE REGISTERS
0001 E5              72          PUSH    H
0002 C5              73          PUSH    B
0003 3EC0            74          MVI     A,0C0H    ; START THE COUNT DOWN MODE FOR TIMER
0005 D328            75          OUT     028H
0007 010000    E     76          LXI     B,DESIR   ; CONVERT DESIR FROM 2-BYTE INTEGER INTO
000A CD0000    E     77          CALL    SEND2     ; 4-BYTE BINARY FLOATING POINT FORMAT
000D 3E1D            78          MVI     A,FLTS    ; ISSUE THE CONVERSION COMMAND TO APU
000F D3FD            79          OUT     APUC
0011 010000    E     80          LXI     B,COEFFD  ; LOAD APU DATA STACK WITH COEFFD
0014 CD0000    E     81          CALL    SEND4
0017 3E12            82          MVI     A,FMUL    ; *
0019 D3FD            83          OUT     APUC
001B 010000    E     84          LXI     B,RESUL   ; STORE  COEFFD*DESIR
001E CD0000    E     85          CALL    READ4
0021 010000    E     86          LXI     B,POSN    ; CONVERT POSN FROM 2-BYTE INTEGER INTO
0024 CD0000    E     87          CALL    SEND2     ; 4-BYTE BINARY FLOATING POINT FORMAT
0027 3E1D            88          MVI     A,FLTS    ; RECOGNIZED BY APU
0029 D3FD            89          OUT     APUC
002B 010000    E     90          LXI     B,COEFF1  ; LOAD APU DATA STACK WITH COEFF1
002E CD0000    E     91          CALL    SEND4
0031 3E12            92          MVI     A,FMUL    ; *
0033 D3FD            93          OUT     APUC
0035 010000    E     94          LXI     B,TEMP    ; COEFF1*POSN
0038 CD0000    E     95          CALL    READ4     ; <---
003B 010000    E     96          LXI     B,COEFF3  ; COEFF3
003E CD0000    E     97          CALL    SEND4     ; ^
0041 010000    E     98          LXI     B,ACCEL   ; ACCEL
0044 CD0000    E     99          CALL    SEND4
0047 3E12           100          MVI     A,FMUL    ; COEFF3*ACCEL
0049 D3FD           101          OUT     APUC
004B 010000    E    102          LXI     B,TEMP
004E CD0000    E    103          CALL    SEND4
0051 3E10           104          MVI     A,FADD    ; +
```

ROUTINE TO EVALUATE SWITCHING FUNCTION USING APU

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0053 D3FD          105          OUT     APUC
 0055 CD0000    E   106          CALL    READ4    ; <---
 0058 010000    E   107          LXI     B,COEFF2 ; COEFF2
 005B CD0000    E   108          CALL    SEND4    ; ^
 005E 010000    E   109          LXI     B,VELO   ; VELO
 0061 CD0000    E   110          CALL    SEND4
 0064 3E12          111          MVI     A,FMUL   ; COEFF2*VELO
 0066 D3FD          112          OUT     APUC
 0068 010000    E   113          LXI     B,TEMP
 006B CD0000    E   114          CALL    SEND4
 006E 3E10          115          MVI     A,FADD   ; +
 0070 D3FD          116          OUT     APUC
 0072 010000    E   117          LXI     B,COEFF0 ; COEFF0
 0075 CD0000    E   118          CALL    SEND4    ; ^
 0078 3E10          119          MVI     A,FADD   ; +
 007A D3FD          120          OUT     APUC
 007C 010000    E   121          LXI     B,RESUL
 007F CD0000    E   122          CALL    SEND4    ; DESIR ^
 0082 3E10          123          MVI     A,FADD   ; +
 0084 D3FD          124          OUT     APUC
 0086 CD0000    E   125          CALL    READ4    ; <----
 0089 3E40          126          MVI     A,040H   ; STOP TIMER BEFOR RETURN TO
 008B D328          127          OUT     028H     ; THE CALLING PROGRAMME
 008D C1            128          POP     B        ; RECOVER REGISTERS SAVED
 008E E1            129          POP     H
 008F F1            130          POP     PSW
 0090 C9            131          RET
                    132 ;
                    133 END
```

PUBLIC SYMBOLS
SWITCH C 0000

EXTERNAL SYMBOLS

ACCEL	E	0000	COEFF0	E	0000	COEFF1	E	0000	COEFF2	E	0000	COEFF3	E	0000	COEFFD	E	0000
DESIR	E	0000	POSN	E	0000	READ2	E	0000	READ4	E	0000	RESUL	E	0000	SEND2	E	0000
SEND4	E	0000	TEMP	E	0000	VELO	E	0000									

USER SYMBOLS

ACCEL	E	0000	ACOS	A	0006	APUC	A	00FD	APUD	A	00FC	ASIN	A	0005	ATAN	A	0007
CHSD	A	0034	CHSF	A	0015	CHSS	A	0074	COEFF0	E	0000	COEFF1	E	0000	COEFF2	E	0000
COEFF3	E	0000	COEFFD	E	0000	COS	A	0003	DADD	A	002C	DDIV	A	002F	DESIR	E	0000
DMUL	A	002E	DMUU	A	0036	DSUB	A	002D	EXP	A	000A	FADD	A	0010	FDIV	A	0013
FIXD	A	001E	FIXS	A	001F	FLTD	A	001C	FLTS	A	001D	FMUL	A	0012	FSUB	A	0011
LN	A	0009	LOG	A	0008	POPD	A	0038	POPF	A	0018	POPS	A	0078	POSN	E	0000
PTOD	A	0037	PTOF	A	0017	PTOS	A	0077	PUPI	A	001A	PWR	A	000B	READ2	E	0000
READ4	E	0000	RESUL	E	0000	SADD	A	006C	SDIV	A	006F	SEND2	E	0000	SEND4	E	0000
SIN	A	0002	SMUL	A	006E	SMUU	A	0076	SQRT	A	0001	SSUB	A	006D	SWITCH	C	0000
TAN	A	0004	TEMP	E	0000	VELO	E	0000	XCHD	A	0039	XCHF	A	0019	XCHS	A	0079

ASSEMBLY COMPLETE, NO ERRORS

```
ASM80 :F1:RDSND.SRC TITLE('I/O BYTE TRANSFER FOR APU')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         RDSND    PAGE    1
I/O BYTE TRANSFER FOR APU

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1 ;*****************************************
                       2 ;*     ROUTINES TO SEND/READ BYTES TO/FROM   *
                       3 ;*     APU STACK                             *
                       4 ;*          I.E.  MEMORY <----> APU STACK    *
                       5 ;*     ON ENTRY:  REGISTERS B,C CONTAIN      *
                       6 ;*                STARTING MEMORY ADDRESS    *
                       7 ;*     LEAST SIGNIFICANT BYTE FIRST          *
                       8 ;*                                           *
                       9 ;*     S/R NAME   FUNCTION                   *
                      10 ;*     --------   --------                   *
                      11 ;*     SEND2      SEND  2  BYTES             *
                      12 ;*     SEND4      SEND  4  BYTES             *
                      13 ;*     READ2      READ  2  BYTES             *
                      14 ;*     READ4      READ  4  BYTES             *
                      15 ;*                                           *
                      16 ;*     WRITTEN BY :  C W CHUEN               *
                      17 ;*     DATE :        08 MAY 1982             *
                      18 ;*****************************************
                      19 NAME    RDSND
                      20 $INCLUDE(:F1:APUCMD.ASM)
                    = 21 $NOLIST
                    = 72 $LIST
                      73 PUBLIC  SEND2,SEND4,READ2,READ4
                      74 CSEG
  0000 E5           75 SEND4:  PUSH    H        ; SAVE REGISTERS
  0001 2E04         76         MVI     L,04H    ; LOAD COUNTER FOR NUMBER OF BYTES
  0003 C5           77 L1:     PUSH    B        ; SAVE B,C-REGISTERS
  0004 F5           78         PUSH    PSW      ; SAVE PROCESSOR STATUS WORD
  0005 0B           79         DCX     B        ; PREPARE MEMORY ADDRESS FOR DATA
  0006 03           80 L2:     INX     B
  0007 0A           81         LDAX    B        ; GET DATA BYTE FROM MEMORY
  0008 D3FC         82         OUT     APUD     ; AND SEND TO APU DATA STACK
  000A 2D           83         DCR     L        ; BYTE AND READY FOR TRANSFER TO APU
  000B C20600    C  84         JNZ     L2       ; LOOP BACK FOR MORE BYTES IF ANY
  000E F1           85         POP     PSW      ; RECOVER REGISTERS BEFORE RETURN
  000F C1           86         POP     B
  0010 E1           87         POP     H
  0011 C9           88         RET              ; RETURN TO CALLING PROGRAMME
  0012 E5           89 SEND2:  PUSH    H        ; SAME AS SEND4 BUT WITH A BYTE COUNT OF 2
  0013 2E02         90         MVI     L,02H
  0015 C30300    C  91         JMP     L1
                    92         ;
                    93         ;
                    94         ;
  0018 E5           95 READ4:  PUSH    H        ; ROUTINE TO READ AND STORE 4 BYTES FROM APU
  0019 210404       96         LXI     H,0404H  ; LOAD COUNT VALUE FOR BYTE TRANSFER
  001C F5           97 L3:     PUSH    PSW      ; SAVE PROCESSOR STATUS WORD
  001D 03           98 L4:     INX     B        ; LOCATE MEMORY ADDRESS TO THE MOST SIGNIFICANT
  001E 25           99         DCR     H        ; BYTE AND READY FOR DATA TRANSFER
  001F C21D00    C 100         JNZ     L4
  0022 DBFC        101 L5:     IN      APUD     ; GET DATA BYTE FROM APU DATA STACK
  0024 0B          102         DCX     B
  0025 02          103         STAX    B        ; STORE THIS BYTE INTO MEMORY
  0026 2D          104         DCR     L        ; DECREMENT COUNTER
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

  0027 C22200   C    105          JNZ     L5        ; LOOP BACK IF NOT YET ZERO
  002A F1            106          POP     PSW       ; RECOVER REGISTERS BEFORE RETURNING TO CALLING
  002B E1            107          POP     H         ; PROGRAMME
  002C C9            108          RET               ; RETURN TO CALLING SEGMENT
  002D E5            109 READ2:   PUSH    H         ; SAME AS READ4 BUT WITH 2 BYTES TO BE
  002E 210202        110          LXI     H,0202H   ; TRANSFERRED
  0031 C31C00   C    111          JMP     L3
                     112 END


PUBLIC SYMBOLS
READ2  C 002D    READ4  C 0018    SEND2  C 0012    SEND4  C 0000

EXTERNAL SYMBOLS


USER SYMBOLS
ACOS   A 0006    APUC   A 00FD    APUD   A 00FC    ASIN   A 0005    ATAN   A 0007    CHSD   A 0034    CHSF   A 0015
CHSS   A 0074    COS    A 0003    DADD   A 002C    DDIV   A 002F    DMUL   A 002E    DMUU   A 0036    DSUB   A 002D
EXP    A 000A    FADD   A 0010    FDIV   A 0013    FIXD   A 001E    FIXS   A 001F    FLTD   A 001C    FLTS   A 001D
FMUL   A 0012    FSUB   A 0011    L1     C 0003    L2     C 0006    L3     C 001C    L4     C 001D    L5     C 0022
LN     A 0009    LOG    A 0008    POPD   A 0038    POPF   A 0018    POPS   A 0078    PTOD   A 0037    PTOF   A 0017
PTOS   A 0077    PUPI   A 001A    PWR    A 000B    READ2  C 002D    READ4  C 0018    SADD   A 006C    SDIV   A 006F
SEND2  C 0012    SEND4  C 0000    SIN    A 0002    SMUL   A 006E    SMUU   A 0076    SQRT   A 0001    SSUB   A 006D
TAN    A 0004    XCHD   A 0039    XCHF   A 0019    XCHS   A 0079

ASSEMBLY COMPLETE,   NO ERRORS
```

```
ASM80 :F1:MRDSND.SRC TITLE('BYTE TRANSFER BETWEEN MEMORY AND M/M APU STACK')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MRDSND   PAGE    1
BYTE TRANSFER BETWEEN MEMORY AND M/M APU STACK

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1 ;*****************************************************************
                      2 ;*     ROUTINES TO SEND/READ BYTES TO/FROM APU STACK              *
                      3 ;*          I.E.  MEMORY <----> APU STACK                         *
                      4 ;*     ON ENTRY:  REGISTERS B,C CONTAIN STARTING MEMORY ADDRESS   *
                      5 ;*                WITH LEAST SIGNIFICANT BYTE FIRST              *
                      6 ;*     THE APU 8231 IS MEMORY MAPPED WITH ADDRESSES FC00H - FDFFH *
                      7 ;*                                                               *
                      8 ;*                S/R NAME   FUNCTION                            *
                      9 ;*                --------   --------                            *
                     10 ;*                SEND2      SEND  2  BYTES                     *
                     11 ;*                SEND4      SEND  4  BYTES                     *
                     12 ;*                READ2      READ  2  BYTES                     *
                     13 ;*                READ4      READ  4  BYTES                     *
                     14 ;*                                                               *
                     15 ;*     WRITTEN BY :  C W CHUEN                                   *
                     16 ;*     DATE :        1 JUNE 1982                                 *
                     17 ;*****************************************************************
                     18 NAME    MRDSND
                     19 $INCLUDE(:F1:APUCMD.ASM)
                =    20 $NOLIST
                =    71 $LIST
                     72 PUBLIC  SEND2,SEND4,READ2,READ4
                     73 CSEG
0000 E5              74 SEND4:  PUSH    H        ; SAVE H,L-REGISTERS
0001 2104FC          75         LXI     H,100H*APUD+04H  ; LOAD APU DATA STACK ADDRESS AND COUNT
                     76                                  ; VALUE FOR BYTE TRANSFER
0004 C5              77 L1:     PUSH    B        ; SAVE B,C-REGISTERS
0005 F5              78         PUSH    PSW      ; SAVE PROCESSOR STATUS WORD
0006 0B              79         DCX     B        ; GET MEMORY ADDRESS PREPARED
0007 03              80 L2:     INX     B
0008 0A              81         LDAX    B        ; GET BYTE FROM MEMORY AND
0009 77              82         MOV     M,A      ; SEND BYTE TO APU DATA STACK
000A 2D              83         DCR     L
000B C20700    C     84         JNZ     L2
000E F1              85         POP     PSW      ; RECOVER REGISTERS BEFORE RETURN
000F C1              86         POP     B
0010 E1              87         POP     H
0011 C9              88         RET              ; RETURN TO CALLING PROGRAMME
0012 E5              89 SEND2:  PUSH    H        ; SAME AS SEND4 BUT WITH A BYTE COUNT OF 2
0013 2102FC          90         LXI     H,100H*APUD+02H  ; LOAD APU DATA STACK ADDRESS AND COUNT
                     91                                  ; VALUE FOR BYTE TRANSFER
0016 C30400    C     92         JMP     L1
                     93         ;
                     94         ;
                     95         ;
0019 E5              96 READ4:  PUSH    H        ; ROUTINE TO READ AND STORE 4 BYTES FROM APU
001A 210404          97         LXI     H,0404H  ; LOAD COUNT VALUE FOR BYTE TRANSFER
001D F5              98 L3:     PUSH    PSW      ; SAVE PROCESSOR STATUS WORD
001E 03              99 L4:     INX     B        ; LOCATE MEMORY ADDRESS TO THE MOST SIGNIFICANT
001F 25             100         DCR     H        ; BYTE AND READY FOR DATA TRANSFER
0020 C21E00    C    101         JNZ     L4
0023 26FC           102         MVI     H,APUD   ; LOAD H-REGISTER WITH APU DATA STACK ADDRESS
0025 7E             103 L5:     MOV     A,M      ; GET BYTE FROM APU DATA STACK AND
0026 0B             104         DCX     B        ; STORE DATA INTO MEMORY USING THE B,C-POINTER
```

BYTE TRANSFER BETWEEN MEMORY AND M/M APU STACK

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0027 02            105         STAX    B
 0028 2D            106         DCR     L           ; DECREMENT BYTE COUNTER
 0029 C22500    C   107         JNZ     L5          ; LOOP BACK IF NOT YET ZERO
 002C F1            108         POP     PSW         ; RECOVER REGISTERS BEFORE RETURNING TO CALLING
 002D E1            109         POP     H           ; PROGRAMME
 002E C9            110         RET                 ; RETURN TO CALLING SEGMENT
 002F E5            111 READ2:  PUSH    H           ; SAME AS READ4 BUT WITH 2 BYTES TO BE TRANSFERRED
 0030 210202        112         LXI     H,0202H
 0033 C31D00    C   113         JMP     L3
                    114 END
```

PUBLIC SYMBOLS

```
READ2  C 002F    READ4  C 0019    SEND2  C 0012    SEND4  C 0000
```

EXTERNAL SYMBOLS

USER SYMBOLS

```
ACOS   A 0006    APUC   A 00FD    APUD   A 00FC    ASIN   A 0005    ATAN   A 0007    CHSD   A 0034    CHSF   A 0015
CHSS   A 0074    COS    A 0003    DADD   A 002C    DDIV   A 002F    DMUL   A 002E    DMUU   A 0036    DSUB   A 002D
EXP    A 000A    FADD   A 0010    FDIV   A 0013    FIXD   A 001E    FIXS   A 001F    FLTD   A 001C    FLTS   A 001D
FMUL   A 0012    FSUB   A 0011    L1     C 0004    L2     C 0007    L3     C 001D    L4     C 001E    L5     C 0025
LN     A 0009    LOG    A 0008    POPD   A 0038    POPF   A 0018    POPS   A 007B    PTOD   A 0037    PTOF   A 0017
PTOS   A 0077    PUPI   A 001A    PWR    A 000B    READ2  C 002F    READ4  C 0019    SADD   A 006C    SDIV   A 006F
SEND2  C 0012    SEND4  C 0000    SIN    A 0002    SMUL   A 006E    SMUU   A 0076    SQRT   A 0001    SSUB   A 006D
TAN    A 0004    XCHD   A 0039    XCHF   A 0019    XCHS   A 0079
```

ASSEMBLY COMPLETE, NO ERRORS

APPENDIX F

FUNCTION APPROXIMATIONS USING THE CHEBYSHEV SERIES

It is extremely difficult if not impossible for a digital computer to evaluate the exact value of a mathematical function (e.g. sine, cosine), so various methods are used to obtain the approximation to a function. The use of power series is commonly adopted in these approximation methods. The Chebyshev series evaluation, which is developed from a power series expansion, is commonly used in digital computer library routines because it is easy and economical to use this series. The Intel-8231 arithmetic processing unit uses internally the Chebyshev series to evaluate the approximation to a mathematical function. In order to compare the function evaluation times in hardware and software, the Chebyshev series approximations to certain mathematical functions have been developed.

A Chebyshev polynomial, which is orthogonal, can be expressed in one of the two forms below:-

$$\text{a)}\quad T_n(x) = \mathrm{Cos}(n\mathrm{Cos}^{-1}x) \qquad n = 0,1,2,3,\ldots \qquad -1 \leqslant x \leqslant 1 \qquad \text{(F.1)}$$

and

$$\text{b)}\quad T^*_n(x) = T_n(2x - 1) = \mathrm{Cos}\{n\mathrm{Cos}^{-1}(2x - 1)\} \qquad \text{(F.2)}$$

$$n = 0,1,2,3,\ldots \qquad 0 \leqslant x \leqslant 1$$

Any finite range can be transformed to the range ($-1 \leqslant x \leqslant 1$) by a linear change in the independant variable, therefore (F.1) is normally used. However, if it is easier and more natural in transforming to the range ($0 \leqslant x \leqslant 1$), e.g. the function $\ln(1+x)$, then (F.2) is used.

It can readily be seen from (F1) that the Chebyshev polynominals have got the recursive property (F3) which is easy to implement in a digital calculating machine.

$$T_{r+1}(x) = 2xT_r(x) - T_{r-1}(x)$$
$$r = 1,2,3 \ldots. \qquad (F3)$$
$$T_o(x) = 1$$
$$T_1(x) = x$$

The first ten explicit expressions in x for these polynominals are listed in Table F1.

TABLE F1 THE FIRST TEN CHEBYSHEV POLYNOMINALS $T_n(x)$

n	Term	Explicit Expression in x
0	$T_o(x)$	1
1	$T_1(x)$	x
2	$T_2(x)$	$2x^2 - 1$
3	$T_3(x)$	$4x^3 - 3x$
4	$T_4(x)$	$8x^4 - 8x^2 + 1$
5	$T_5(x)$	$16x^5 - 20x^3 + 5x$
6	$T_6(x)$	$32x^6 - 48x^4 + 18x^2 - 1$
7	$T_7(x)$	$64x^7 - 112x^5 + 56x^3 - 7x$
8	$T_8(x)$	$128x^8 - 256x^6 + 160x^4 - 32x^2 - 1$
9	$T_9(x)$	$256x^9 - 576x^7 + 432x^5 - 120x^3 - 9x$

A continuous function in the region $(-1 \leqslant x \leqslant 1)$ can be expressed in a Chebyshev polynominal:

$$f(x) = a_o + a_1T_1(x) + a_2T_2(x) + \ldots.$$
$$= \sum_{r=o}^{\infty} a_rT_r(x) \qquad (F4)$$

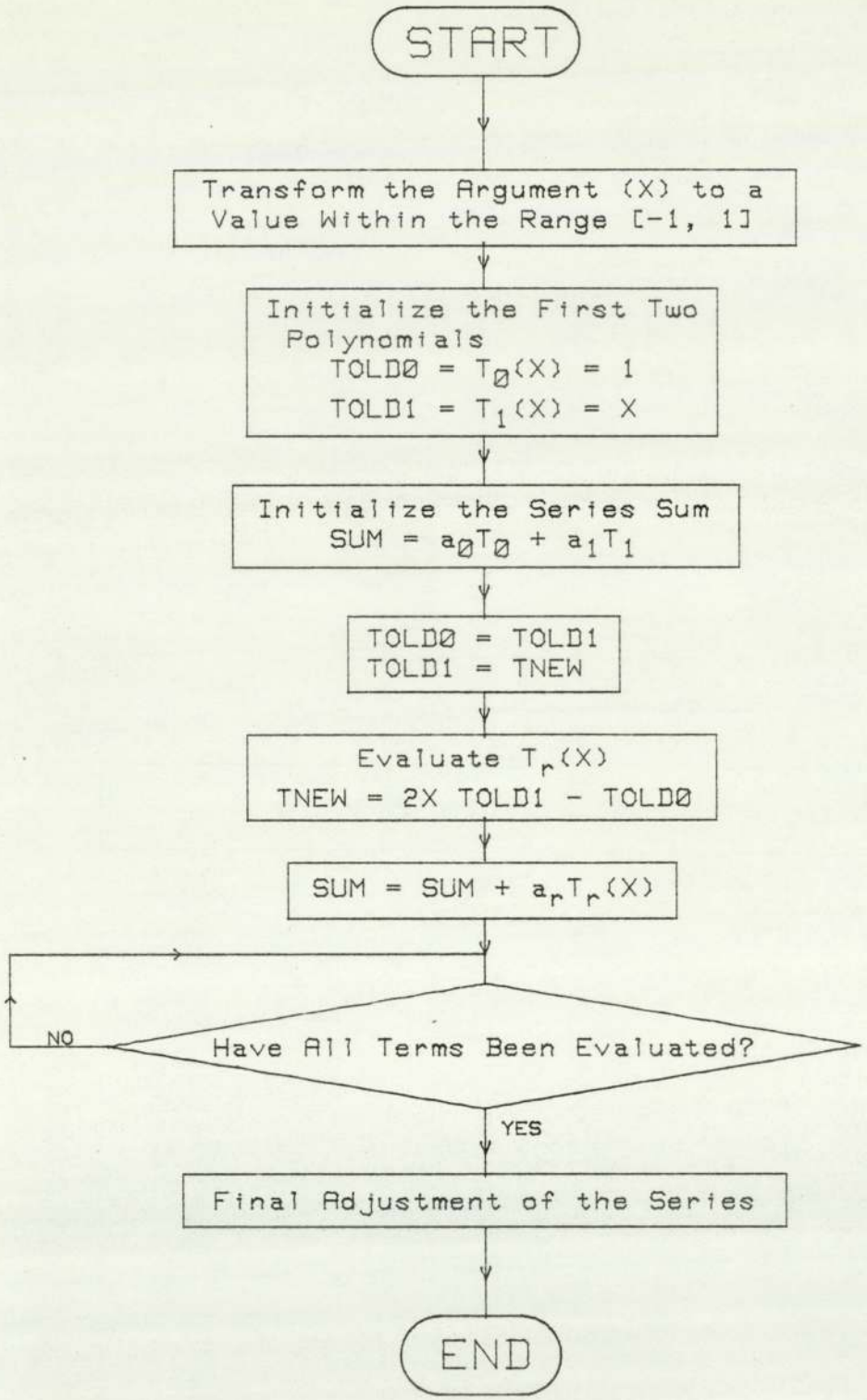


FIG F.1 Flow Diagram for the Algorithm in Evaluating the Chebyshev Series

In fact, (F4) represents the Fourier cosine series expansion of the function f(Cos θ) and is periodic. This periodic function converges rapidly, the coefficients are in general readily evaluated and the approximations of any specified accuracy can be obtained by mere truncation of the series. An algorithm in evaluating the Chebyshev series can be obtained from (F3), the flow diagram for this algorithm can be represented in Figure F1. The coefficients, a_r, of the Chebyshev series are taken from the Mathematical Tables by Clenshaw [46].

The sine function is used as an example to illustrate how this algorithm works. This series evaluation uses the Intel floating point arithmetic library (FPAL) and works in binary floating point numbers.

The following FPAL routines are involved:

Name	Function
FSET	Set up the arithmetic library routines.
FLOAD	Load the binary number from the memory to the floating point accumulator (FAC).
FSTOR	Store the binary number in the FAC to the memory.
FMUL	Multiply the binary number in the FAC to that in the memory and put the result in the FAC.
FDIV	Divide the binary number in the FAC by that in the memory and put the result in the FAC.
FADD	Add the binary number in the FAC by that in the memory and put the result in the FAC.
FSUB	Subtract the binary number in the FAC by that in the memory and put the result in the FAC.
FZTST	Check whether the number in the FAC is less than, equal to or greater than zero.

FNEG	Change the sign of the number in the FAC.
FCMPR	Compare the number in the FAC to that in the memory.

The following sub-routines are constructed in evaluating the Chebyshev series,

Name	Function	Code Lines in CHEBY
INCHEB	Initialise the first two terms in the Chebyshev series.	20 - 43
NEWX	Evaluate the new term by F3. $T_{r+1}(x) = 2xT_r(x) - T_{r-1}(x)$.	44 - 59
MOVE	Move T_r to T_{r-1}. T_{r+1} to T_r.	60 - 73
SUMCH	Sum the Chebyshev series. SUM = SUM + FAC.	74 - 79

The program for evaluating the function, sin(x), is also listed. The argument is first reduced to a value within the range $[-2\pi,0]$ and then transformed to a value within the range $[-1,1]$. The approximation is then carried out by the algorithm in Figure F1, with the following Chebyshev series:

$$\sin \tfrac{1}{2}\pi x = x \sum_{r=0}^{\infty} a_{2r}\, T_{2r}(x) \quad \text{where } -1 \leqslant x \leqslant 1$$

```
ASM80 :F1:CHEBY.SRC MOD85 TITLE('CHEBYSHEV SERIES')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         CHEBY    PAGE    1
CHEBYSHEV SERIES

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1 ;*******************************************
                       2 ;*   ROUTINE TO CALCULATE CHEBYSHEV SERIES  *
                       3 ;*       ON ENTRY:  B,C POINT TO FPR        *
                       4 ;*                  PSW HAS BEEN PUSHED     *
                       5 ;*                  ONTO STACK BEFORE CALL- *
                       6 ;*                  ING THIS ROUTINE        *
                       7 ;*******************************************
                       8 NAME    CHEBY
                       9 PUBLIC  INCHEB,NEWX,MOVE,SUMCH
                      10 EXTRN   ONE,FSTOR,FLOAD,FMUL,FSUB,FADD
                      11 $INCLUDE(:F1:CHEB.STK)
  0018             =  12 XSTOR   EQU     24D ; DEFINE XSTOR TO BE 24D
  0014             =  13 TWOX    EQU     XSTOR - 4
  0010             =  14 TOLD0   EQU     TWOX  - 4
  000C             =  15 TOLD1   EQU     TOLD0 - 4
  0008             =  16 TNEW    EQU     TOLD1 - 4
  0004             =  17 SUM     EQU     TNEW  - 4
                   =  18 $LIST
                      19 CSEG
  0000 211800         20 INCHEB: LXI   H,XSTOR ; ROUTINE TO INITIALIZE FIRST
  0003 39             21         DAD   SP      ; THREE TERMS IN THE CHEBYSHEV
  0004 EB             22         XCHG          ; SERIES
  0005 CD0000   E     23         CALL  FSTOR   ; ARGUMENT (X) IS IN FAC WHEN
  0008 210C00         24         LXI   H,TOLD1 ; INCHEB IS CALLED
  000B 39             25         DAD   SP
  000C EB             26         XCHG
  000D CD0000   E     27         CALL  FSTOR   ; T(1) = X
  0010 211000         28         LXI   H,16D   ; FORM 2*ARGU BY ADDING ONE TO
  0013 09             29         DAD   B       ; THE EXPONENT BYTE OF FAC
  0014 7E             30         MOV   A,M     ;     THIS BYTE IS 16D BYTE
  0015 3C             31         INR   A       ;     THE LOW BYTE OF FPR
  0016 77             32         MOV   M,A
  0017 211400         33         LXI   H,TWOX  ; STORE THIS INTO TWOX
  001A 39             34         DAD   SP
  001B EB             35         XCHG
  001C CD0000   E     36         CALL  FSTOR
  001F 110000   E     37         LXI   D,ONE   ; MAKE  T(0)  =  1
  0022 CD0000   E     38         CALL  FLOAD
  0025 211000         39         LXI   H,TOLD0
  0028 39             40 LIN1:   DAD   SP
  0029 EB             41         XCHG
  002A CD0000   E     42         CALL  FSTOR
  002D C9             43         RET
  002E 211400         44 NEWX:   LXI   H,TWOX
  0031 39             45         DAD   SP
  0032 EB             46         XCHG
  0033 CD0000   E     47         CALL  FLOAD   ; LOAD 2*ARGU INTO FAC
  0036 210C00         48         LXI   H,TOLD1 ; PERFORM 2*X*T(N)
  0039 39             49         DAD   SP
  003A EB             50         XCHG
  003B CD0000   E     51         CALL  FMUL
  003E 211000         52         LXI   H,TOLD0 ; PERFORM 2*X*T(N) - T(N-1)
  0041 39             53         DAD   SP
  0042 EB             54         XCHG
```

CHEBYSHEV SERIES

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0043 CD0000   E     55          CALL  FSUB
 0046 210800         56          LXI   H,TNEW  ; STORE THIS T(N+1) INTO TNEW
 0049 39             57          DAD   SP
 004A EB             58          XCHG
 004B CD0000   E     59          CALL  FSTOR
 004E 210C00         60 MOVE:    LXI   H,TOLD1 ; MOVE TOLD1 TO TOLD0
 0051 39             61          DAD   SP
 0052 EB             62          XCHG
 0053 CD0000   E     63          CALL  FLOAD
 0056 211000         64          LXI   H,TOLD0
 0059 39             65          DAD   SP
 005A EB             66          XCHG
 005B CD0000   E     67          CALL  FSTOR
 005E 210800         68          LXI   H,TNEW  ; MOVE TNEW TO TOLD1
 0061 39             69          DAD   SP
 0062 EB             70          XCHG
 0063 CD0000   E     71          CALL  FLOAD
 0066 210C00         72          LXI   H,TOLD1
 0069 C32800   C     73 JUMPL:   JMP   LIN1
 006C 210400         74 SUMCH:   LXI   H,SUM   ; PERFORM SUMMATION FOR SERIES
 006F 39             75          DAD   SP      ; AND STORE SUM IN FAC AND SUM
 0070 EB             76          XCHG
 0071 CD0000   E     77          CALL  FADD
 0074 CD0000   E     78          CALL  FSTOR
 0077 C9             79          RET
                     80 END


PUBLIC SYMBOLS
INCHEB C 0000    MOVE   C 004E    NEWX   C 002E    SUMCH  C 006C

EXTERNAL SYMBOLS
FADD   E 0000    FLOAD  E 0000    FMUL   E 0000    FSTOR  E 0000    FSUB   E 0000    ONE    E 0000

USER SYMBOLS
FADD   E 0000    FLOAD  E 0000    FMUL   E 0000    FSTOR  E 0000    FSUB   E 0000    INCHEB C 0000    JUMPL  C 0069
LIN1   C 0028    MOVE   C 004E    NEWX   C 002E    ONE    E 0000    SUM    A 0004    SUMCH  C 006C    TNEW   A 0008
TOLD0  A 0010    TOLD1  A 000C    TWOX   A 0014    XSTOR  A 0018

ASSEMBLY COMPLETE,   NO ERRORS
```

```
ASM80 :F1:CONST.SRC


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        CONST    PAGE    1

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1 ;*****************************************
                       2 ;*  SEGMENTS CONTAIN CONSTANTS :          *
                       3 ;*        ONE, PI, 2PI, PI/2 ,ZERO        *
                       4 ;*****************************************
                       5 NAME  CONST
                       6 CSEG
                       7 PUBLIC   ONE,ZERO,PI,TWOPI,PI2
  0000 00              8 ZERO:    DB   000H,000H,000H,000H ; BFP FOR 0
  0001 00
  0002 00
  0003 00
  0004 00              9 ONE:     DB   000H,000H,080H,03FH ; BFP FOR 1
  0005 00
  0006 80
  0007 3F
  0008 DB             10 PI:      DB   0DBH,00FH,049H,040H ; BFP FOR PI
  0009 0F
  000A 49
  000B 40
  000C DB             11 TWOPI:   DB   0DBH,00FH,0C9H,040H ; BFP FOR 2PI
  000D 0F
  000E C9
  000F 40
  0010 DB             12 PI2:     DB   0DBH,00FH,0C9H,03FH ; BFP FOR PI/2
  0011 0F
  0012 C9
  0013 3F
                      13 END

PUBLIC SYMBOLS
ONE    C 0004    PI     C 0008    PI2    C 0010    TWOPI  C 000C    ZERO   C 0000

EXTERNAL SYMBOLS


USER SYMBOLS
ONE    C 0004    PI     C 0008    PI2    C 0010    TWOPI  C 000C    ZERO   C 0000

ASSEMBLY COMPLETE,   NO ERRORS
```

```
ASM80 :F1:TSINE3.SRC MOD85 MACROFILE(:F1:) TITLE('SINE ROUTINE')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         SIN      PAGE    1
SINE ROUTINE

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1 ;*********************************************
                       2 ;*    A ROUTINE FOR THE SINE FUNCTION          *
                       3 ;*       THIS REQUIRES: FPAL.LIB               *
                       4 ;*                      MYLIB                  *
                       5 ;*                      CHEBY.OBJ              *
                       6 ;*    ON ENTRY:  B,C CONTAIN FPR ADDRESS       *
                       7 ;*               D,C CONTAIN ARGUMENT ADDRESS  *
                       8 ;*    ON EXIT :  FAC CONTAINS THE VALUE OF     *
                       9 ;*                 SINE(ARGUMENT)              *
                      10 ;*    THIS REQUIRES AN EXTRA 24D-BYTE STACK    *
                      11 ;*    FOR TEMPORARY STORAGE OF RESULTS         *
                      12 ;*********************************************
                      13 NAME    SIN
                      14 EXTRN   FLOAD,FSTOR,FMUL,FCLR,FNEG,FZTST,FADD
                      15 EXTRN   FSUB,FDIV,ONE,PI,TWOPI,INCHEB,NEWX,SUMCH
                      16 PUBLIC  SIN,SINE1
                      17 $INCLUDE(:F1:CHEB.STK)
  0018            =   18 XSTOR   EQU     24D ; DEFINE XSTOR TO BE 24D
  0014            =   19 TWOX    EQU     XSTOR - 4
  0010            =   20 TOLD0   EQU     TWOX  - 4
  000C            =   21 TOLD1   EQU     TOLD0 - 4
  0008            =   22 TNEW    EQU     TOLD1 - 4
  0004            =   23 SUM     EQU     TNEW  - 4
                  =   24 $LIST
                      25 CSEG
  0000 F5             26 SIN:    PUSH    PSW ; SAVE REGISTERS ON ENTRY
  0001 C5             27         PUSH    B
  0002 D5             28         PUSH    D
  0003 E5             29         PUSH    H
  0004 21E8FF         30         LXI     H,-24D ; LOCATE STACK POINTER TO 24D
  0007 39             31         DAD     SP     ; BYTE BELOW THE CURRENT
  0008 F9             32         SPHL           ; LOCATION
  0009 CD0000   E     33         CALL    FLOAD  ; LOAD ARGUMENT INTO FAC
  000C CD0000   E     34 SINE1:  CALL    FZTST  ; CHECK IF ARGUMENT IS -VE, 0 OR +VE
  000F F5             35         PUSH    PSW    ; SAVE STATUS WORD FOR LATER USE
  0010 17             36         RAL
  0011 DACA00   C     37         JC      XIS0   ; JUMP IF ARGUMENT IS ZERO
  0014 17             38         RAL
  0015 DA1B00   C     39         JC      XGT0   ; JUMP IF ARGUMENT IS POSITIVE
  0018 CD0000   E     40         CALL    FNEG   ; TAKE ABSOLUTE VALUE OF AGRUMENT
  001B 110000   E     41 XGT0:   LXI     D,TWOPI ; REDUCE TO RANGE [-2PI, 0]
  001E CD0000   E     42 MINUS:  CALL    FSUB
  0021 CD0000   E     43         CALL    FZTST
  0024 17             44         RAL
  0025 DACA00   C     45         JC      XIS0    ; JUMP IF ARGUMENT IS MULTIPLE OF 2PI
  0028 17             46         RAL
  0029 DA1E00   C     47         JC      MINUS   ; NEEDS FURTHER REDUCTION
  002C 110000   E     48         LXI     D,PI    ; FORM -(Y + PI)/PI
  002F CD0000   E     49         CALL    FADD
  0032 CD0000   E     50         CALL    FZTST   ; CHECK IF ARGUMENT IS MULTIPLE OF PI
  0035 17             51         RAL
  0036 DACA00   C     52         JC      XIS0
  0039 CD0000   E     53         CALL    FDIV
  003C CD0000   E     54         CALL    FNEG
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 003F CD0000   E    55          CALL    INCHEB  ; INITIALIZE THE CHEBYSHEV SERIES
 0042 11D800   C    56          LXI     D,A0    ; INITIALIZE THE SUM
 0045 CD0000   E    57          CALL    FLOAD
 0048 210200        58          LXI     H,SUM-2
 004B 39            59          DAD     SP
 004C EB            60          XCHG
 004D CD0000   E    61          CALL    FSTOR
 0050 CD0000   E    62          CALL    NEWX
 0053 11DC00   C    63          LXI     D,A2
 0056 CD0000   E    64          CALL    FMUL
 0059 CD0000   E    65          CALL    SUMCH
                    66          IRP     X,<A4,A6,A8,A10,A12,A14>
 -                  67          CALL    NEWX
 -                  68          CALL    NEWX
 -                  69          LXI     D,X
 -                  70          CALL    FMUL
 -                  71          CALL    SUMCH
                    72          ENDM
 005C CD0000   E    73+         CALL    NEWX
 005F CD0000   E    74+         CALL    NEWX
 0062 11E000   C    75+         LXI     D,A4
 0065 CD0000   E    76+         CALL    FMUL
 0068 CD0000   E    77+         CALL    SUMCH
 006B CD0000   E    78+         CALL    NEWX
 006E CD0000   E    79+         CALL    NEWX
 0071 11E400   C    80+         LXI     D,A6
 0074 CD0000   E    81+         CALL    FMUL
 0077 CD0000   E    82+         CALL    SUMCH
 007A CD0000   E    83+         CALL    NEWX
 007D CD0000   E    84+         CALL    NEWX
 0080 11E800   C    85+         LXI     D,A8
 0083 CD0000   E    86+         CALL    FMUL
 0086 CD0000   E    87+         CALL    SUMCH
 0089 CD0000   E    88+         CALL    NEWX
 008C CD0000   E    89+         CALL    NEWX
 008F 11EC00   C    90+         LXI     D,A10
 0092 CD0000   E    91+         CALL    FMUL
 0095 CD0000   E    92+         CALL    SUMCH
 0098 CD0000   E    93+         CALL    NEWX
 009B CD0000   E    94+         CALL    NEWX
 009E 11F000   C    95+         LXI     D,A12
 00A1 CD0000   E    96+         CALL    FMUL
 00A4 CD0000   E    97+         CALL    SUMCH
 00A7 CD0000   E    98+         CALL    NEWX
 00AA CD0000   E    99+         CALL    NEWX
 00AD 11F400   C   100+         LXI     D,A14
 00B0 CD0000   E   101+         CALL    FMUL
 00B3 CD0000   E   102+         CALL    SUMCH
 00B6 211600       103          LXI     H,XSTOR-2
 00B9 39           104          DAD     SP
 00BA EB           105          XCHG
 00BB CD0000   E   106          CALL    FMUL
 00BE F1           107          POP     PSW     ; CHECK IF ADJUSTMENT IS REQUIRED
 00BF 17           108          RAL
 00C0 17           109          RAL
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        SIN      PAGE    3
SINE ROUTINE

  LOC  OBJ          LINE        SOURCE STATEMENT

  00C1 DACE00   C    110         JC      NOADJ
  00C4 CD0000   E    111         CALL    FNEG    ; SINCE  SIN(-X) = -SIN(X)
  00C7 C3CE00   C    112         JMP     NOADJ
  00CA CD0000   E    113 XIS0:   CALL    FCLR    ; CLEAR FAC SINCE SIN(0) = 0
  00CD F1            114         POP     PSW
  00CE 211800        115 NOADJ:  LXI     H,24D   ; RECOVER STACK POINTER LOCATION
  00D1 39            116         DAD     SP
  00D2 F9            117         SPHL
  00D3 E1            118         POP     H  ; RECOVER REGISTERS BEFORE RETURNING
  00D4 D1            119         POP     D
  00D5 C1            120         POP     B
  00D6 F1            121         POP     PSW
  00D7 C9            122         RET             ; RETURN TO CALLING PROGRAMME
  00D8 BD            123 A0:     DB      0BDH,07BH,0ACH,03FH ; 2.695052629347980342462
  00D9 7B
  00DA AC
  00DB 3F
  00DC 61            124 A2:     DB      061H,03EH,0C7H,0BFH ;-1.556591256628969313 08
  00DD 3E
  00DE C7
  00DF BF
  00E0 A6            125 A4:     DB      0A6H,01AH,064H,03EH ; 0.222757911817011171 52
  00E1 1A
  00E2 64
  00E3 3E
  00E4 7E            126 A6:     DB      07EH,08AH,068H,0BCH ;-0.014193174148537272 56
  00E5 8A
  00E6 68
  00E7 BC
  00E8 84            127 A8:     DB      084H,031H,006H,03AH ; 0.000511907274554114 54
  00E9 31
  00EA 06
  00EB 3A
  00EC 37            128 A10:    DB      037H,08AH,047H,0B7H ;-0.000011893504653342 08
  00ED 8A
  00EE 47
  00EF B7
  00F0 A5            129 A12:    DB      0A5H,03DH,04FH,034H ; 0.000000193008036063 80
  00F1 3D
  00F2 4F
  00F3 34
  00F4 5C            130 A14:    DB      05CH,0EBH,01EH,0B1H ;-0.000000002312581057 81
  00F5 EB
  00F6 1E
  00F7 B1
  00F8 BC            131 A16:    DB      0BCH,064H,0BBH,02DH ; 0.000000000021304174 39
  00F9 64
  00FA BB
  00FB 2D
                     132 END

PUBLIC SYMBOLS
SIN     C 0000    SINE1   C 000C
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0         SIN      PAGE    4
SINE ROUTINE

EXTERNAL SYMBOLS
FADD    E 0000    FCLR    E 0000    FDIV    E 0000    FLOAD   E 0000    FMUL    E 0000    FNEG    E 0000    FSTOR   E 0000
FSUB    E 0000    FZTST   E 0000    INCHEB  E 0000    NEWX    E 0000    ONE     E 0000    PI      E 0000    SUMCH   E 0000
TWOPI   E 0000

USER SYMBOLS
A0      C 00D8    A10     C 00EC    A12     C 00F0    A14     C 00F4    A16     C 00F8    A2      C 00DC    A4      C 00E0
A6      C 00E4    A8      C 00E8    FADD    E 0000    FCLR    E 0000    FDIV    E 0000    FLOAD   E 0000    FMUL    E 0000
FNEG    E 0000    FSTOR   E 0000    FSUB    E 0000    FZTST   E 0000    INCHEB  E 0000    MINUS   C 001E    NEWX    E 0000
NOADJ   C 00CE    ONE     E 0000    PI      E 0000    SIN     C 0000    SINE1   C 000C    SUM     A 0004    SUMCH   E 0000
TNEW    A 0008    TOLD0   A 0010    TOLD1   A 000C    TWOPI   E 0000    TWOX    A 0014    XGT0    C 001B    XIS0    C 00CA
XSTOR   A 0018

ASSEMBLY COMPLETE,   NO ERRORS
```

APPENDIX G

LISTING OF THE ASSEMBLY LANGUAGE PROGRAMS FOR THE BOOM POSITIONAL CONTROL BY SIMPLE COMPARISON WITH SLOW SPEED LIMITATION

```
ASM80 :F1:INIT1.SRC MOD85 DEBUG MACROFILE XREF TITLE('CONTROL WITH SIMPLE COMPARISION') PRINT(:LP:)


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0          CNTRL     PAGE     1
CONTROL WITH SIMPLE COMPARISION

  LOC  OBJ          LINE        SOURCE STATEMENT

                      1 ;*****************************************************************
                      2 ;*      THE INITIALIZATION PROGRAM FOR BOOM POSITION CONTROL      *
                      3 ;*                                                                *
                      4 ;*      HARDWARE INTERRUPTS ARE USED IN THE FOLLOWING WAYS:       *
                      5 ;*                                                                *
                      6 ;*               INTERRUPT         CONDITION                    *
                      7 ;*               ---------         ---------                    *
                      8 ;*                TRAP             THE TUNNEL PROFILE HAS BEEN  *
                      9 ;*                                 DEFINED BY THE OPERATOR      *
                     10 ;*                RST 5.5          TO CUT THE TUNNEL PROFILE    *
                     11 ;*                RST 6.5          TO EXCAVATE THE CAVITY       *
                     12 ;*                RST 7.5          TO MOVE THE BOOM             *
                     13 ;*****************************************************************
                     14 NAME    CNTRL
                     15 ;*****************************************************************
                     16 PUBLIC  STKPTR,CUTSTP,SORTFL,INTMSK,SIGNAL
                     17 ;*****************************************************************
                     18 ;*      THE INITIALIZATION                                        *
                     19 ;*****************************************************************
                     20         ASEG              ; DECLARE THE PROGRAM ORIGIN
0000                 21         ORG  00000H       ; INITIALIZE THE PROGRAM CODE COUNTER
                     22 START:                    ; PROGRAM STARTS RUNNING HERE WHEN EITHER THE POWER IS SWITCHED
                     23                           ; ON OR THE RESET BUTTON HAS BEEN PRESSED
0000 310000    D     24         LXI  SP,STKPTR    ; INITIALIZE THE STACK POINTER
0003 3E1F            25         MVI  A,01FH       ; SET THE INITIAL INTERRUPT MASK TO DISABLE ALL INTERRUPTS
0005 6F              26         MOV  L,A          ; COPY THIS INFORMATION TO L
0006 2608            27         MVI  H,08H        ; DEFINE THE DEFAULT STEP SIZE FOR THE CUTTER ADVANCEMENT
0008 30              28 COLDST: SIM               ; SET THE INTERRUPT MASK
0009 220000    D     29         SHLD INTMSK       ; STORE THE ABOVE INFORMATION
                     30                           ; THE INTERRUPT MASK AT   INTMSK
                     31                           ; THE CUTTER STEP SIZE AT CUTSTP
000C 3EFF            32         MVI  A,0FFH       ; OUTPUT THE SIGNAL 'FF H' TO THE PORT 0 FOR CLOSING THE
000E D300            33         OUT  00H          ; DIRECTIONAL VALVES
                     34                           ;    BITS 7,6,5,4 ---- SLEWING RAMS
                     35                           ;    BITS 3,2,1,0 ---- LIFTING RAMS
0010 D302            36         OUT  02H          ; DECLARE ALL THE 8 BITS OF PORT 0 TO BE OUTPUTS
                     37                           ; BY SENDING 0FFH TO ITS D.D.R. (PORT 2)
                     38                           ; PORT 0 IS USED FOR CONTROLLING THE DIRECTION VALVES
                     39                           ; CLOSE A VALVE BY SENDING AN ONE TO THE ASSOCIATE PORT
                     40                           ; OPEN A VALVE BY SENDING A ZERO TO THE ASSOCIATE PORT
0012 3E67            41         MVI  A,067H       ; OUTPUT THE INITIAL BIT PATTERN TO PORT 01H
0014 D301            42         OUT  01H          ;       THIS IS  0110 0111  B
                     43 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0016 3E7F           44          MVI  A,07FH     ; DECLARE THE APPROPRIATE I/O BITS OF PORT 1
0018 D303           45          OUT  03H        ; BY SETTING ITS D.D.R. (PORT 3) WITH THIS PATTERN : 01111111 B
001A 321800    D    46          STA  DDR01      ; STORE THIS INFORMATION
                    47                          ; PORT 1 IS USED IN THE FOLLOWING WAY:
                    48                          ;       PB0,PB1 --- FAST/SLOW CONTROL (DEFAULTED TO 1)
                    49                          ;                   --- 1 FOR SLOW MOVE
                    50                          ;                   --- 0 FOR FAST MOVE
                    51                          ;               PB2 --- START CUTTER (DEFAULTED TO 1)
                    52                          ;               PB3 --- STOP CUTTER CUTTER (DEFAULTED TO 0)
                    53                          ;               PB4 --- RESET THE APU (DEFAULTED TO 0)
                    54                          ;                   THIS IS HIGH ACTIVE
                    55                          ;               PB5 --- 0 TO HOLD THE OTHER SYSTEM
                    56                          ;                   --- 1 NORMAL OPERATION (DEFAULT CONDITION)
                    57                          ;               PB6 --- 0 TO ENABLE THE INERFACE BUFFERS
                    58                          ;                   --- 1 NORMAL OPERATION (DEFAULT CONDITION)
                    59                          ;               PB7 --- HOLDA FROM THE OTHER SYSTEM (INPUT)
                    60                          ;           PATTERN: 10000000 --- I=INPUT, O=OUTPUT
                    61                          ; SET THE INITIAL PATTERN FOR PORT 1:   01100111B
                    62                          ;       BOOM SLOW MOVE MODE IS THE DEFAULT SETTING
                    63                          ;       DATA ARE NOT REQUIRED FROM THE OTHER SYSTEM
                    64                          ;       THE APU IS NOT RESETTED
                    65                          ;       THE CUTTER IS STOPPED
001D 97             66          SUB  A          ; SET THE INITIAL SORT PROFILE FLAG TO ZERO
001E 320200    D    67          STA  SORTFL'    ; STORE THIS FLAG
0021 76             68          HLT             ; WAIT FOR TRAP INTERRUPT TO INDICATE THAT THE TUNNEL
                    69                          ; PROFILE HAS BEEN DEFINED BY THE OPERATOR
0024                70          ORG  0024H
0024 C32B03         71 TRAP:    JMP  RST45      ; HARDWARE TRAP INTERRUPT ENTRY POINT
002C                72          ORG  002CH
002C C3B500         73 RST55:   JMP  PROFIL     ; HARDWARE RST 5.5 INTERRUPT ENTRY POINT
0034                74          ORG  0034H
0034 C31802         75 RST65:   JMP  CAVITY     ; HARDWARE RST 6.5 INTERRUPT ENTRY POINT
0038                76          ORG  0038H
0038 C35900         77          JMP  RST38      ; HARDWARE INTR INTERRUPT ENTRY POINT
003C                78          ORG  003CH
003C C3DC02         79 RST75:   JMP  MOVE       ; HARDWARE RST 7.5 INTERRUPT ENTRY POINT
                    80 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    81 ;*****************************************************************
                    82 ;*      ROUTINES TO SET UP AND DIS-ENGAGE THE DATA LINK BETWEEN   *
                    83 ;*      THE MAIN CONTROL SYSTEM AND THE DATA CAPTURING SYSTEM     *
                    84 ;*      NAME      FUNCTION                                        *
                    85 ;*      ----      --------                                        *
                    86 ;*      SETLNK    SET UP THE DATA LINK BETWEEN THE TWO SYSTEMS    *
                    87 ;*      OFFLNK    DIS-ENGAGE THE DATA LINK                        *
                    88 ;*                                                                *
                    89 ;*      BIT 7 OF PORT 1 IS USED FOR THE HOLD ACKNOWLEDGE SIGNAL   *
                    90 ;*      BIT 6 OF PORT 1 IS USED FOR ENABLING THE BUFFERS          *
                    91 ;*      BIT 5 OF PORT 1 IS USED FOR HOLDING THE DATA SYSTEM       *
                    92 ;*      THEREFORE BIT 7 OF PORT 1 IS AN INPUT                     *
                    93 ;*                BITS 5 AND 6 OF PORT 1 ARE OUTPUTS              *
                    94 ;*      THE PROCESSOR STATUS WORD WILL BE DESTROYED               *
                    95 ;*****************************************************************
                    96 PUBLIC  SETLNK,OFFLNK
0040                97         ORG  0040H
                    98 SETLNK:                  ; A ROUTINE TO SET UP THE INTERFACE CONDITION FOR DATA TRANSFER
                    99                          ; FROM THE DATA CAPTURING SYSTEM
                   100                          ; THE PROCESSOR STATUS WORD WILL BE DESTROYED
0040 DB01          101         IN   01H         ; GET CURRENT SETTING OF PORT 1
0042 E6DF          102         ANI  0DFH        ; SET THE APPROPRIATE BIT VALUES FOR HOLDING THE ADDRESS AND DATA
                   103                          ; BUSES OF THE SECOND SYSTEM BY ANDING IT WITH 11011111B
                   104                          ; THE PATTERN IS: XX0XXXXX B
0044 D301          105         OUT  01H         ; OUTPUT THIS PATTERN TO PORT 1
0046 DB01          106         IN   01H         ; READ THE PORT 1 SIGNAL
0048 17            107         RAL              ; CHECK WHETHER THE SECOND SYSTEM HAS ACKNOWLEDGED THE HOLD SIGNAL
0049 D24600        108         JNC  $-3         ; KEEP CHECKING IF NOT YET ACKNOWLEDGED
004C 1F            109         RAR              ; RECOVER THE ORIGINAL PORT 1 READING
004D E6BF          110         ANI  0BFH        ; MAKE BIT 6 ZERO BY ANDING IT WITH 1011111 B
004F D301          111         OUT  01          ; SEND IT TO PORT 01
0051 C9            112         RET              ; RETURN TO THE CALLING PROGRAM
                   113 OFFLNK:                  ; A ROUTINE TO DIS-ENGAGE THE SECOND SYSTEM
                   114                          ; THE PROCESSOR STATUS WORD WILL BE DESTROYED
0052 DB01          115         IN   01H         ; GET CURRENT PORT 1 BYTE VALUE
0054 F660          116         ORI  060H        ; SET THE 'DIS-ENGAGE' BIT-PATTERN: X11XXXXX B
                   117                          ; BY ORING IT WITH 01100000 B
0056 D301          118         OUT  01H         ; OUTPUT THIS PATTERN
0058 C9            119         RET
                   120                          ;
                   121 RST38:                   ; MODE CHANGE IS REQUIRED
0059 DB01          122         IN   01H         ; SET THE SPEED TO SLOW MODE
005B F603          123         ORI  003H        ;      PATTERN: XXXX XX11
005D D301          124         OUT  01H
005F 3EFF          125         MVI  A,0FFH      ; CLOSE ALL THE DIRECTIONAL VALVES
0061 D300          126         OUT  00H
0063 3A0200    D   127         LDA  SORTFL      ; CHECK IF THE PROFILE DATA HAVE BEEN SORTED
0066 17            128         RAL
0067 D22B03        129         JNC  RST45       ; SORT AND EXPAND THEM IF NOT
006A 3E18          130         MVI  A,018H
006C 30            131         SIM              ; OTHERWISE, RESET THE INTERRUPT MASK
006D 320000    D   132         STA  INTMSK
0070 310000    D   133         LXI  SP,STKPTR   ; RE-LOCATE THE STACK POINTER
0073 FB            134         EI
0074 76            135         HLT              ; WAIT FOR THE SIGNAL TO INITIATE AN OPERATION
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    136 $INCLUDE(:F1:TRNPFL.SRC)
                  = 137 ;*****************************************************************
                  = 138 ;*      A ROUTINE TO TRANSFER PROFILE DATA FROM THE SOURCE TO     *
                  = 139 ;*      THE DESTINATION                                           *
                  = 140 ;*      THE DATA MUST BE STORED IN CONTIGUOUS MEMORIES FOLLOWING  *
                  = 141 ;*      THE TWO BYTES SPECIFYING THE LAST ADDRESS OF DATA         *
                  = 142 ;*      i.e.  LAST DATA ADDRESS, DATA                             *
                  = 143 ;*      ON ENTRY: REGISTERS B,C POINT TO THE SOURCE DATA BLOCK     *
                  = 144 ;*                REGISTERS D,E POINT TO THE DESTINATION BLOCK     *
                  = 145 ;*      ON EXIT:  TRANSFER COMPLETED WITH THE LAST DATA ADDRESS    *
                  = 146 ;*                AT THE DESTINATION BLOCK STORED IN THE TWO       *
                  = 147 ;*                BYTES BELOW THE DATA AS IN THE SOURCE            *
                  = 148 ;*****************************************************************
                  = 149 PUBLIC  TRNPFL
0075 F5           = 150 TRNPFL: PUSH PSW        ; SAVE THE REGISTERS ON ENTRY
0076 E5           = 151         PUSH H
0077 C5           = 152         PUSH B
0078 D5           = 153         PUSH D
0079 69           = 154         MOV  L,C        ; GET THE LAST ADDRESS OF THE SOURCE DATA
007A 60           = 155         MOV  H,B
007B 4E           = 156         MOV  C,M        ; AND PUT IT IN THE REGISTERS B,C
007C 23           = 157         INX  H
007D 46           = 158         MOV  B,M
007E 13           = 159         INX  D
007F 23           = 160 LOOP1:  INX  H          ; ADJUST THE ADDRESSES OF BOTH THE DESTINATION AND SOURCE DATA
0080 13           = 161         INX  D
0081 7E           = 162         MOV  A,M        ; MOVE THE SOURCE DATA TO THE ACCUMULATOR
0082 12           = 163         STAX D          ; THEN STORE IT IN THE DESTINATION MEMORY
0083 7C           = 164         MOV  A,H        ; CHECK IF THE LAST DATA HAS BEEN TRANSFERRED
0084 B8           = 165         CMP  B
0085 C27F00       = 166         JNZ  LOOP1      ; LOOP BACK IF NOT
0088 7D           = 167         MOV  A,L
0089 B9           = 168         CMP  C
008A DA7F00       = 169         JC   LOOP1      ; LOOP BACK IF NOT
008D E1           = 170         POP  H          ; GET THE POINTER FOR STORING THE LAST DATA ADDRESS AT DESTINATION
008E E5           = 171         PUSH H
008F 73           = 172         MOV  M,E        ; STORE THE ADDRESS
0090 23           = 173         INX  H
0091 72           = 174         MOV  M,D
0092 D1           = 175         POP  D          ; RECOVER THE REGISTERS
0093 C1           = 176         POP  B
0094 E1           = 177         POP  H
0095 F1           = 178         POP  PSW
0096 C9           = 179         RET
                    180 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    181 $INCLUDE(:F1:TRANSF.SRC)
                  = 182 ;*************************************************************
                  = 183 ;*      A ROUTINE TO TRANSFER DATA FROM SOURCE TO DESTINATION  *
                  = 184 ;*      FOR X BYTES                                            *
                  = 185 ;*      ON ENTRY: REGISTERS B,C --- STARTING SOURCE ADDRESS    *
                  = 186 ;*                REGISTERS D,E --- STARTING DESTINATION ADDRESS*
                  = 187 ;*      ON EXIT:  REGISTERS HAVE BEEN PRESERVED                *
                  = 188 ;*************************************************************
                  = 189 PUBLIC  TRANSF
                  = 190 ;*************************************************************
0097 F5           = 191 TRANSF: PUSH PSW        ; PUSH THE BYTE COUNTER ONTO THE STACK BEFORE CALLING THIS ROUTINE
                  = 192                         ; MAXIMUM NUMBER OF BYTES TRANSFERRED: 0FFFFH (64K)
0098 E5           = 193         PUSH H          ; SAVE THE REGISTERS
0099 C5           = 194         PUSH B
009A D5           = 195         PUSH D
009B 210A00       = 196         LXI  H,10D      ; ADJUST THE STACK POINTER TO GET THE BYTE COUNT
009E 39           = 197         DAD  SP         ; THIS BYTE COUNTER IS 10D BYTE ABOVE THE CURRENT STACK POINTER
009F 5E           = 198         MOV  E,M        ; TRANSFER THIS BYTE VALUE (LOW ORDER) TO REGISTER E
00A0 23           = 199         INX  H          ; GET THE HIGH ORDER BYTE AND TRANSFER TO REGISTER D
00A1 56           = 200         MOV  D,M
00A2 E1           = 201         POP  H          ; GET THE STARTING DESTINATION ADDRESS
00A3 E5           = 202         PUSH H
00A4 EB           = 203         XCHG            ; PUT THIS ADDRESS BACK TO REGISTERS D,E AND THE BYTE COUNTER TO H,L
00A5 0A           = 204 LOOP3:  LDAX B          ; GET THE BYTE VALUE FROM SOURCE
00A6 12           = 205         STAX D          ; STORE THIS TO THE DESTINATION
00A7 03           = 206         INX  B          ; INCREMENT THE ADDRESSES
00A8 13           = 207         INX  D
00A9 2B           = 208         DCX  H          ; DECREMENT THE COUNTER
00AA 97           = 209         SUB  A          ; CLEAR THE ACCUMULATOR
00AB B4           = 210         ORA  H
00AC B5           = 211         ORA  L
00AD C2A500       = 212         JNZ  LOOP3      ; LOOP BACK IF THE TRANSFER HAS NOT BEEN COMPLETED
00B0 D1           = 213         POP  D          ; RECOVER THE REGISTERS
00B1 C1           = 214         POP  B
00B2 E1           = 215         POP  H
00B3 F1           = 216         POP  PSW
00B4 C9           = 217         RET
                    218 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    219 $INCLUDE(:F1:PROFIL.SRC)
                  = 220 ;*************************************************************
                  = 221 ;*      A ROUTINE TO CUT THE TUNNEL PROFILE                  *
                  = 222 ;*      THIS ROUTINE IS INVOKED BY THE RST 5.5 INTERRUPT     *
                  = 223 ;*************************************************************
                  = 224 PUBLIC  PROFIL
                  = 225 ;*************************************************************
00B5 3EFF         = 226 PROFIL: MVI   A,0FFH    ; CLOSE THE DIRECTIONAL VALVES BY SENDING A 'FF'
00B7 D300         = 227         OUT   00H       ; TO PORT 0
00B9 DB01         = 228         IN    01H       ; STOP THE CUTTER
00BB E6F7         = 229         ANI   0F7H      ; PATTERN: XXXX 01XX
00BD F604         = 230         ORI   004H
00BF D301         = 231         OUT   01H
00C1 310000     D = 232         LXI   SP,STKPTR ; LOCATE THE STACK POINTER TO MAKE THE WHOLE STACK AVAILABLE
00C4 3A0000     D = 233         LDA   INTMSK
00C7 F61D         = 234         ORI   01DH      ; SET THE INTERRUPT MASK TO DISABLE THE 'CUT PROFIL' AND
00C9 320000     D = 235         STA   INTMSK    ; THE 'MOVE' CONTROLS. PATTERN IS XXX111X1
00CC 30           = 236         SIM
00CD 3A0200     D = 237         LDA   SORTFL    ; CHECK IF THE PROFILE DATA HAVE BEEN SORTED
00D0 17           = 238         RAL
00D1 D2B001       = 239         JNC   NOTSRT    ; JUMP TO THE NO SORT ROUTINE IF NOT
00D4 2A0400     D = 240         LHLD  STARCV    ; GET THE FIRST SET OF DATA
00D7 44           = 241         MOV   B,H
00D8 4D           = 242         MOV   C,L
00D9 0A           = 243         LDAX  B         ; FIRST THE RAISE DATA
00DA 321B00     D = 244         STA   RSDESR
00DD 03           = 245         INX   B         ; INCREMENT THE B,C TO POINT AT THE MIN. SLEW BOUND
00DE 0A           = 246         LDAX  B         ; GET THE SLEW VALUE
00DF 321000     D = 247         STA   FIRSL
00E2 321A00     D = 248         STA   SLDESR    ; STORE THIS DESIRED SLEW VALUE
00E5 0B           = 249         DCX   B         ; RECOVER THE REGISTER B,C PAIR
00E6 CDAE80       = 250 FASTS:  CALL  FMOVE     ; FAST MOVE TO THIS STARTING POINT
00E9 DB00         = 251         IN    00H       ; CHECK IF THE DIRECTIONAL VALVES HAVE BEEN CLOSED
00EB 2F           = 252         CMA
00EC B7           = 253         ORA   A
00ED C2E600       = 254         JNZ   FASTS     ; LOOP BACK IF NOT
00F0 CD6F07       = 255         CALL  CHKPOS    ; CHECK IF THE DESIRED POSITION HAS BEEN REACHED
00F3 17           = 256         RAL
00F4 D2E600       = 257         JNC   FASTS     ; LOOP BACK IF NOT YET REACHED
00F7 DB01         = 258 FASTS1: IN    01H       ; OTHERWISE, ISSUE THE START CUTTER SIGNAL
00F9 E6FB         = 259         ANI   0FBH
00FB F608         = 260         ORI   008H
00FD D301         = 261         OUT   01H       ; PATTERN IS: XXXX 10XX
00FF 3EFF         = 262         MVI   A,0FFH
0101 321900     D = 263         STA   CUTTER    ; STORE THE FLAG INDICATING THAT THE CUTTER HAS BEEN STARTED
0104 2A0400     D = 264         LHLD  STARCV
0107 44           = 265         MOV   B,H
0108 4D           = 266         MOV   C,L
0109 2A0600     D = 267 CTPRO1: LHLD  LASTCV
010C 78           = 268         MOV   A,B       ; CHECK IF THE UPPEST LEVEL IS BEING CUT
010D BC           = 269         CMP   H
010E DA1901       = 270         JC    NOTUP
0111 C23301       = 271         JNZ   OVER1
0114 79           = 272         MOV   A,C
0115 BD           = 273         CMP   L
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0116 D23301         = 274          JNC  OVER1
 0119 03             = 275 NOTUP:   INX  B          ; IF NOT, GET THE NEXT SLEW BOUND AND RAISE LEVEL DATA
 011A 0A             = 276          LDAX B
 011B 321000     D   = 277          STA  FIRSL
 011E 0B             = 278          DCX  B
 011F 3A1000     D   = 279          LDA  FIRSL
 0122 321A00     D   = 280          STA  SLDESR
 0125 0A             = 281          LDAX B
 0126 321B00     D   = 282          STA  RSDESR
 0129 CD4007         = 283          CALL SMOVE1    ; CUT TO THIS POINT (MOVE ALONG THE LEFT BOUNDARY)
 012C 03             = 284          INX  B
 012D 03             = 285          INX  B
 012E 03             = 286          INX  B
 012F 03             = 287          INX  B
 0130 C30901         = 288          JMP  CTPRO1
 0133 2A0600     D   = 289 OVER1:   LHLD LASTCV    ; CUT THE UPPER MOST LEVEL
 0136 44             = 290          MOV  B,H
 0137 4D             = 291          MOV  C,L
 0138 CD7805         = 292          CALL SHORT
 013B 0A             = 293          LDAX B
 013C 321B00     D   = 294          STA  RSDESR
 013F 03             = 295          INX  B
 0140 0A             = 296          LDAX B
 0141 321000     D   = 297          STA  FIRSL
 0144 03             = 298          INX  B
 0145 0A             = 299          LDAX B
 0146 321100     D   = 300          STA  FIRSL+1
 0149 0B             = 301          DCX  B
 014A 0B             = 302          DCX  B
 014B 3A1000     D   = 303          LDA  FIRSL
 014E 321A00     D   = 304          STA  SLDESR
 0151 CD4007         = 305          CALL SMOVE1
 0154 3A1100     D   = 306          LDA  FIRSL+1
 0157 321A00     D   = 307 CTPRO2:  STA  SLDESR
 015A CD4007         = 308          CALL SMOVE1    ; MOVE ALONG THE RIGHT BOUNDARY
 015D 2A0400     D   = 309          LHLD STARCV
 0160 78             = 310          MOV  A,B
 0161 BC             = 311          CMP  H         ; CHECK IF THE LOWEST LEVEL HAS BEEN CUT
 0162 DA8201         = 312          JC   UNDER1
 0165 C27001         = 313          JNZ  NOTLOW
 0168 79             = 314          MOV  A,C
 0169 BD             = 315          CMP  L
 016A DA8201         = 316          JC   UNDER1
 016D CA8201         = 317          JZ   UNDER1
 0170 0B             = 318 NOTLOW:  DCX  B
 0171 0B             = 319          DCX  B
 0172 0A             = 320          LDAX B         ; GET THE SLEW BOUND
 0173 321000     D   = 321          STA  FIRSL
 0176 0B             = 322          DCX  B
 0177 0B             = 323          DCX  B
 0178 0A             = 324          LDAX B         ; AND THE RAISE LEVEL
 0179 321B00     D   = 325          STA  RSDESR
 017C 3A1000     D   = 326          LDA  FIRSL
 017F C35701         = 327          JMP  CTPRO2    ; LOOP BACK TO CUT THE PROFILE SIDE BOUNDARY
 0182 2A0400     D   = 328 UNDER1:  LHLD STARCV    ; CUT THE LOWEST LEVEL
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0185 44           = 329         MOV  B,H
 0186 4D           = 330         MOV  C,L
 0187 0A           = 331         LDAX B
 0188 321B00    D  = 332         STA  RSDESR
 018B CD7B05       = 333         CALL SHORT
 018E 3A1000    D  = 334         LDA  FIRSL
 0191 321A00    D  = 335         STA  SLDESR
 0194 CD4007       = 336         CALL SMOVE1
 0197 3A1100    D  = 337         LDA  FIRSL+1
 019A 321A00    D  = 338         STA  SLDESR
 019D CD4007       = 339         CALL SMOVE1
 01A0 DB01         = 340         IN   01H          ; OUTPUT THE 'STOP CUTTER' SIGNAL
 01A2 E6F7         = 341         ANI  0F7H
 01A4 F604         = 342         ORI  004H
 01A6 D301         = 343         OUT  01H
 01A8 3E18         = 344         MVI  A,018H       ; RESET THE INTERRUPT MASK
 01AA 30           = 345         SIM
 01AB 320000    D  = 346         STA  INTMSK
 01AE FB           = 347         EI
 01AF 76           = 348         HLT               ; WAIT FOR THE SYSTEM INTERRUPT SIGNAL TO START AN OPERATION
                   = 349 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                  = 350 NOTSRT:                 ; THE PROFILE DATA HAVE NOT BEEN SORTED AND EXPANDED
01B0 01FE1B       = 351         LXI   B,PFLEND  ; LOCATE THE REGISTERS B,C POINTING TO THE SOURCE BLOCK
01B3 11FEBB       = 352         LXI   D,LASTAD  ; LOCATE THE REGISTERS D,E POINTING TO THE DESTINATION BLOCK
01B6 CD4000       = 353         CALL  SETLNK    ; SET UP THE LINK BETWEEN THE TWO SYSTEMS
01B9 CD7500       = 354         CALL  TRNPFL    ; GET THE PROFILE DATA FROM THE DATA CAPTURING SYSTEM
01BC CD5200       = 355         CALL  OFFLNK    ; RELEASE THE LINK BETWEEN THE TWO SYSTEMS
01BF 2100BC       = 356 FAST1:  LXI   H,STADD   ; LOAD THE STARTING PROFILE DATA ADDRESS TO H,L
01C2 56           = 357         MOV   D,M       ; LOAD THE DESIRED 'ELEVATION' VALUE TO THE REGISTER D
01C3 23           = 358         INX   H
01C4 5E           = 359         MOV   E,M       ; LOAD THE DESIRED 'SLEW' VALUE TO THE REGISTER E
01C5 EB           = 360         XCHG            ; EXCHANGE D,E WITH H,L
01C6 221A00     D = 361         SHLD  SLDESR    ; STORE THESE INFORMATIONS INTO THE APPROPRIATE LOCATIONS
01C9 EB           = 362         XCHG            ; RECOVERS THE REGISTERS D, E, H, AND L
01CA CDAE80       = 363         CALL  FMOVE     ; CALL FAST MOVE
01CD DB00         = 364         IN    00H       ; GET THE CURRENT VALVE SETTING
01CF 2F           = 365         CMA
01D0 B7           = 366         ORA   A         ; CHECK IF ALL VALVES ARE CLOSED
01D1 C2BF01       = 367         JNZ   FAST1     ; LOOP BACK IF THIS POISITION HAS NOT YET BEEN REACHED
01D4 CD6F07       = 368         CALL  CHKPOS    ; CHECK IF THE DESIRED POSITION HAS BEEN REACHED
01D7 17           = 369         RAL
01D8 D2BF01       = 370         JNC   FAST1     ; LOOP BACK IF NOT
01DB DB01         = 371         IN    01H       ; OTHERWISE, GET THE CURRENT PATTERN OF PORT 1 AND THEN
                  = 372                         ; ISSUE THE START CUTTER COMMAND
                  = 373                         ;     ( BIT 2 = 0 TO START THE CUTTER
                  = 374                         ;       BIT 3 = 0 TO STOP THE CUTTER)
                  = 375                         ; AND SET THE SLOW MOVE MODE
                  = 376                         ; THE PATTERN IS XXXX1011B AND IS OBTAINED
01DD E6FB         = 377         ANI   0FBH      ;       BY FIRST ANDING IT WITH 11111011B
01DF F60B         = 378         ORI   00BH      ;       AND THEN  ORING IT WITH 00001011B
01E1 D301         = 379         OUT   01H       ; SEND THE SIGNAL TO THE PORT
01E3 3EFF         = 380         MVI   A,0FFH    ; STORE THE CUTTER STATUS
01E5 321900     D = 381         STA   CUTTER
01E8 E5           = 382         PUSH  H         ; SAVE H,L
01E9 2AFEBB       = 383         LHLD  LASTAD
01EC 4D           = 384         MOV   C,L
01ED 44           = 385         MOV   B,H       ; LOAD THE REGISTERS B,C WITH THE LAST ADDRESS FOR THE PROFILE DATA
01EE E1           = 386         POP   H         ; RECOVER H,L
01EF 03           = 387         INX   B         ; INCREMENT THE FINAL ADDRESS BY 1
01F0 23           = 388 LOOP:   INX   H         ; INCREMENT PROFILE DATA ADDRESS
01F1 7C           = 389         MOV   A,H       ; CHECK IF LAST DATA HAS BEEN REACHED
01F2 B8           = 390         CMP   B
01F3 DAFB01       = 391         JC    CONTP1    ; CONTINUE OPERATION IF NOT
01F6 7D           = 392         MOV   A,L       ; CHECK THE LOW ORDER BYTE
01F7 B9           = 393         CMP   C
01F8 CA0902       = 394         JZ    STOFP1    ; STOP IF THE COMPLETE PROFILE HAS BEEN TRIMMED
01FB 56           = 395 CONTP1: MOV   D,M       ; GET THE RAISE DATA
01FC 23           = 396         INX   H
01FD 5E           = 397         MOV   E,M       ; GET THE SLEW DATA
01FE EB           = 398         XCHG
01FF 221A00     D = 399         SHLD  SLDESR    ; STORE THESE DATA IN THE APPROPRIATE MEMORIES
0202 EB           = 400         XCHG
0203 CD4007       = 401         CALL  SMOVE1    ; MOVE SLOWLY TO THIS POSITION
0206 C3F001       = 402         JMP   LOOP      ; LOOP BACK FOR THE NEXT PROFILE POSITION
0209 DB01         = 403 STOFP1: IN    01H       ; ISSUE THE STOP CUTTER COMMAND: XXXX01XXB
020B E6F7         = 404         ANI   0F7H      ; BY  ANDING IT WITH 11110111B
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

020D F604         = 405         ORI  004H       ; THEN ORING IT WITH 00000100B
020F D301         = 406         OUT  01H
0211 97           = 407         SUB  A          ; CLEAR THE ACCUMULATOR
0212 321900     D = 408         STA  CUTTER     ; STORE THIS CUTTER STATUS
0215 C32B03       = 409         JMP  RST45      ; JUMP TO SORT THE PROFILE DATA OUT
                    410 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    411 $INCLUDE(:F1:CAVITY.SRC)
                  = 412 ;*************************************************************************
                  = 413 ;*       THE CUT CAVITY PROGRAM                                          *
                  = 414 ;*       A PROGRAM TO EXCAVATE THE CAVITY OF THE TUNNEL                  *
                  = 415 ;*       THIS PROGRAM IS INVOKED BY THE RST 6.5 INTERRUPT                *
                  = 416 ;*************************************************************************
                  = 417 PUBLIC  CAVITY
                  = 418 ;*************************************************************************
0218 3EFF         = 419 CAVITY: MVI   A,0FFH     ; CLOSE ALL THE DIRECTIONAL VALVES
021A D300         = 420         OUT   00H
021C 310000     D = 421         LXI   SP,STKPTR ; LOCATE THE STACK POINTER TO THE INITIAL VALUE
021F 3E1E         = 422         MVI   A,01EH     ; SET THE CURRENT INTERRUPT MASK TO MASK OUT THE RST 7.5
                  = 423                          ; AND THE RST 6.5 INTERRUPTS WHILE UNMASK THE RST 5.5 INTERRUPT
                  = 424                          ; RST 5.5 --- 'TRIM PROFILE' BUTTON
                  = 425                          ; RST 6.5 --- 'CUT CAVITY' BUTTON
                  = 426                          ; RST 7.5 --- 'MOVE' CONTROL LEVER
                  = 427                          ;  01EH = 0001 1110 B
0221 320000     D = 428         STA   INTMSK     ; STORE THIS NEW INTERRUPT MASK SETTING
0224 30           = 429         SIM              ; SET THE INTERRUPT MASK
0225 2A0100     D = 430         LHLD CUTSTP      ; GET THE CUTTER STEP SIZE
0228 7D           = 431         MOV   A,L        ; REDUCE THE CUTSTP BY HALF
0229 FE02         = 432         CPI   02H        ; CHECK WHETHER THE CUTSTP IS GREATER THAN 2
022B DA2F02       = 433         JC    $+4        ; MARGIN WILL BE EQUAL TO THE STEP SIZE IF LESS
022E 1F           = 434         RAR
022F 321600     D = 435         STA   HALFST     ; AND STORE THIS IN HALFST
0232 2600         = 436         MVI   H,00H      ; MULTIPLY THIS BY 4
0234 29           = 437         DAD   H
0235 29           = 438         DAD   H
0236 221200     D = 439         SHLD FOURST      ; STORE THIS RESULT IN FOURST
0239 FB           = 440         EI               ; ENABLE THE SYSTEM INTERRUPT
023A 2A0400     D = 441         LHLD STARCV      ; LOAD THE REGISTERS B,C WITH THE STARTING DATA ADDRESS
023D 23           = 442 REDUST: INX   H          ; INCREASE THE START CAVITY DATA ADDRESS
023E 23           = 443         INX   H
023F 23           = 444         INX   H
0240 23           = 445         INX   H
0241 3D           = 446         DCR   A          ; BY HALF BLOCKS
0242 C23D02       = 447         JNZ   REDUST
0245 44           = 448         MOV   B,H        ; COPY THIS ADDRESS TO B,C
0246 4D           = 449         MOV   C,L
0247 2A0600     D = 450         LHLD LASTCV      ; LOAD THE REGISTERS H,L WITH THE END ADDRESS FOR THE EVELOPE
024A 3A1600     D = 451         LDA   HALFST     ; GET THE HALF STEP SIZE
024D 2B           = 452 REDULS: DCX   H          ; DECREMENT THE LAST CAVITY DATA ADDRESS
024E 2B           = 453         DCX   H
024F 2B           = 454         DCX   H
0250 2B           = 455         DCX   H
0251 3D           = 456         DCR   A
0252 C24D02       = 457         JNZ   REDULS
0255 221400     D = 458         SHLD LASTCT      ; STORE THIS ADDRESS
0258 EB           = 459         XCHG             ; THEN PUT IT ONTO D,E
0259 CDB605       = 460         CALL  REDUCE     ; GET THE FIRST SLEW BOUND
025C 0A           = 461         LDAX  B          ; GET THE DESIRED RAISE DATA
025D 321B00     D = 462         STA   RSDESR     ; STORE IT
0260 3A1000     D = 463         LDA   FIRSL      ; GET THE SLEW BOUND
0263 321A00     D = 464         STA   SLDESR     ; AND STORE IT
0266 CDAE80       = 465 CALSMV: CALL  FMOVE
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0269 DB00          = 466          IN   00H        ; GET THE DIRECTIONAL VALVES SETTING
026B 2F            = 467          CMA            ; CHECK IF THEY ARE CLOSED
026C B7            = 468          ORA  A
026D C26602        = 469          JNZ  CALSMV    ; LOOP BACK IF THE POSITION HAS NOT BEEN REACHED
0270 CD6F07        = 470          CALL CHKPOS    ; CHECK IF THE DESIRED POSITION HAS BEEN REACHED
0273 17            = 471          RAL
0274 D26602        = 472          JNC  CALSMV    ; LOOP BACK IF NOT
0277 DB01          = 473          IN   01H       ; START THE CUTTER IF YES
0279 F608          = 474          ORI  08BH      ; PATTERN   XXXX 10XX B
027B E6FB          = 475          ANI  0FBH
027D D301          = 476          OUT  01H
027F 3A1100    D   = 477 CONTC5:  LDA  FIRSL+1   ; GET THE NEXT SLEW BOUND DATA
0282 321A00    D   = 478          STA  SLDESR
0285 CD4007        = 479          CALL SMOVE1    ; MOVE AND CUT TO THIS BOUND
0288 2A1200    D   = 480          LHLD FOURST    ; GET THE STEP IN THE TABLE
028B 09            = 481          DAD  B         ; THEN ADDED TO CURRENT ADDRESS FOR 'RAISE' DATA
028C 7C            = 482          MOV  A,H       ; CHECK IF THE LASTCV HAS BEEN REACHED
028D BA            = 483          CMP  D         ; FIRST THE HIGH ORDER BYTE
028E DA9902        = 484          JC   CONTC0    ; CONTINUE IF NOT
0291 C2AE02        = 485          JNZ  CONTC2    ; ASSUME LASTCV IF GONE PASS
0294 7D            = 486          MOV  A,L       ; THEN THE LOW ORDER BYTE
0295 BB            = 487          CMP  E
0296 D2AE02        = 488          JNC  CONTC2    ; ASSUME LASTCV IF GONE PASS
0299 44            = 489 CONTC0:  MOV  B,H
029A 4D            = 490          MOV  C,L
029B CDB605        = 491          CALL REDUCE
029E 0A            = 492          LDAX B
029F 321B00    D   = 493          STA  RSDESR
02A2 3A1000    D   = 494          LDA  FIRSL
02A5 321A00    D   = 495          STA  SLDESR
02A8 CD4007        = 496          CALL SMOVE1
02AB C37F02        = 497          JMP  CONTC5
02AE 2A1400    D   = 498 CONTC2:  LHLD LASTCT    ; ASSUME LASTCV BLOCK
02B1 7E            = 499          MOV  A,M       ; GET THIS RAISE DATA
02B2 321B00    D   = 500          STA  RSDESR
02B5 44            = 501          MOV  B,H
02B6 4D            = 502          MOV  C,L
02B7 CDB605        = 503          CALL REDUCE
02BA 3A1000    D   = 504          LDA  FIRSL
02BD 321A00    D   = 505          STA  SLDESR
02C0 CD4007        = 506          CALL SMOVE1
02C3 3A1100    D   = 507          LDA  FIRSL+1
02C6 321A00    D   = 508          STA  SLDESR
02C9 CD4007        = 509          CALL SMOVE1
02CC DB01          = 510          IN   01H       ; STOP THE CUTTER
02CE E6F7          = 511          ANI  0F7H      ; PATTERN  XXXX 01XX B
02D0 F604          = 512          ORI  04H
02D2 D301          = 513          OUT  01H
02D4 3E18          = 514          MVI  A,018H    ; RESET THE INTERRUPT MASK TO ENABLE ALL INTERRUPTS
02D6 320000    D   = 515          STA  INTMSK    ; PATTERN: 0001 1000 B
02D9 30            = 516          SIM
02DA FB            = 517          EI
02DB 76            = 518          HLT            ; WAIT FOR ANOTHER SYSTEM INTERRUPTTARTING SOURCE ADDRESS
                   519 $EJECT
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    520 $INCLUDE(:F1:MOVE.SRC)
                  = 521 ;*************************************************************************
                  = 522 ;*       PROGRAM SEGMENT TO SERVICE THE RST 7.5 INTERRUPT                *
                  = 523 ;*************************************************************************
                  = 524 PUBLIC  MOVE
                  = 525 ;*************************************************************************
                  = 526 ;*       SYMBOLS FOR DATA STORAGE IN THE DATA CAPTURING SYSTEM           *
                  = 527 ;*       SYMBOL                                                          *
                  = 528 ;*       ------                                                          *
                  = 529 ;*       CRRTSL        CURRENT SLEW POSITION         (1 BYTE)          *
                  = 530 ;*       MOVEST        CURRENT MOVE LEVER STATUS     (1 BYTE)          *
                  = 531 ;*       DESRSL        DESIRED SLEW POSITION         (1 BYTE)          *
                  = 532 ;*                                                                      *
                  = 533 ;*       SYMBOLS USED FOR DATA STORAGE IN THE CURRENT SYSTEM             *
                  = 534 ;*       SYMBOL                                                          *
                  = 535 ;*       ------                                                          *
                  = 536 ;*       SLDESR        DESIRED SLEW POSITION         (1 BYTE)          *
                  = 537 ;*       SLCRRT        CURRENT SLEW POSITION         (1 BYTE)          *
                  = 538 ;*       INTMSK        INTERRUPT MASK SETTING        (1 BYTE)          *
                  = 539 ;*************************************************************************
02DC F3           = 540 MOVE:   DI                ; ENSURE THAT THE SYSTEM INTERRUPT IS DISABLED
02DD 3E1C         = 541         MVI  A,01CH       ; RESET THE INTERRUPT MASK TO MASK OUT THE 'MOVE' INTERRUPT
02DF 30           = 542         SIM
02E0 320000     D = 543         STA  INTMSK
02E3 210900       = 544         LXI  H,9D         ; SET THE COUNTER FOR THE BYTE TRANSFER TO 9
02E6 E3           = 545         XTHL              ; AND SCRATCH THE RETURN ADDRESS
02E7 01DA1B       = 546 MOVE2:  LXI  B,DESRSL     ; SET UP THE ADDRESSES FOR DATA TRANSFER
02EA 111A00     D = 547         LXI  D,SLDESR
02ED CD4000       = 548         CALL SETLNK       ; SET UP THE INTERFACE CONDITIONS BETWEEN THE
                  = 549                           ; TWO MICROPROCESSOR SYSTEMS
02F0 CD9700       = 550         CALL TRANSF       ; CALL THE DATA TRANSFER ROUTINE
02F3 3AA21B       = 551         LDA  CRRTSL       ; GET AND STORE THE CURRENT BOOM POSITION
02F6 322300     D = 552         STA  SLCRRT
02F9 3AAD1B       = 553         LDA  CRRTRS
02FC 322400     D = 554         STA  RSCRRT
02FF 3ADC1B       = 555         LDA  MOVEST       ; GET THE MOVE STATUS BYTE AND
0302 F5           = 556         PUSH PSW          ; SAVE THIS DATA ONTO THE STACK
0303 CD5200       = 557         CALL OFFLNK       ; RELEASE THE ADDRESS AND DATA BUSES
0306 F1           = 558         POP  PSW          ; OTHERWISE, GET THE MOVE STATUS BYTE FROM THE STACK
0307 FE10         = 559         CPI  010H         ; COMPARE WITH THE LOWER BOUND FOR MOVE
0309 DA1C03       = 560         JC   NOMOVE       ; STOP IF LOWER THAN THIS BOUND
030C 17           = 561         RAL
030D DA1603       = 562         JC   MOVE1        ; JUMP TO FAST MOVE THE BOOM
0310 CDEA05       = 563         CALL SMOVE        ; SLOW MOVE THE BOOM
0313 C3E702       = 564         JMP  MOVE2
0316 CDAE80       = 565 MOVE1:  CALL FMOVE        ; FAST MOVE THE BOOM
0319 C3E702       = 566         JMP  MOVE2
031C 3EFF         = 567 NOMOVE: MVI  A,0FFH       ; CLOSE ALL THE DIRECTIONAL VALVES BY SENDING 'FF'
031E D300         = 568         OUT  00H          ; TO PORT 0
0320 3E18         = 569         MVI  A,018H       ; RESET THE RST 7.5 INTERRUPT AND UNMASK ALL OTHER INTERRPUTS
0322 30           = 570         SIM               ; SET INTERRUPT MASK
0323 320000     D = 571         STA  INTMSK       ; STORE THIS INTERRUPT MASK INFORMATION
0326 310000     D = 572         LXI  SP,STKPTR    ; RE-LOCATE THE STACK POINTER
0329 FB           = 573         EI                ; ENABLE THE SYSTEM INTERRUPTS
032A 76           = 574         HLT               ; WAIT FOR AN INTERRUPT SIGNAL TO START A PROCESS
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    575 ;*****************************************************************
                    576 ;*      THE ROUTINE ON RECOGNIZING THE TRAP INTERRUPT AFTER THE   *
                    577 ;*      PROFILE DATA HAS BEEN ENTERED                            *
                    578 ;*      BOTH THE 'STOP CUTTER' AND 'START CUTTER' LEDs  ARE ON   *
                    579 ;*      INDICATING THAT ONLY THE 'CUT PROFILE' IS OPERATIVE      *
                    580 ;*                                                               *
                    581 ;*      ON EXIT: ONLY THE 'STOP CUTTER' LED IS ON                *
                    582 ;*               ALL INTERRUPTS ARE ENABLED                      *
                    583 ;*                                                               *
                    584 ;*      FUNCTION: TO SORT THE PROFILE ENVELOPE                   *
                    585 ;*                INTERPOLATE THESE DATA AND EXPAND TO COVER EVERY *
                    586 ;*                BIT CHANGE IN ALTITUDE IF NECESSARY             *
                    587 ;*****************************************************************
                    588 PUBLIC  RST45,STARCV,LASTCV,LASTSO,DMIN1,LEVEL
                    589 ;*****************************************************************
 032B F3            590 RST45:  DI                  ; ENSURE THAT THE SYSTEM IS DISABLED
 032C 310000    D   591         LXI  SP,STKPTR      ; RE-LOCATE THE STACK POINTER
 032F 3EFF          592         MVI  A,0FFH         ; CLOSE ALL THE DIRECTIONAL VALVES
 0331 D300          593         OUT  00H
 0333 DB01          594         IN   01H            ; LIGHT THE 'START' AND 'STOP' LEDs SIMULTANEOUSLY TO
 0335 E6F3          595         ANI  0F3H           ; INDICATE THAT ONLY THE 'TRIM PROFILE' IS ENABLED
 0337 D301          596         OUT  01H            ; AND THAT THE PROFILE DATA ARE BEING SORTED AND EXPANDED
 0339 3E1E          597         MVI  A,01EH         ; RESET THE INTERRUPT MASK TO 0001 1110 TO UNMASK ONLY
 033B 30            598         SIM                 ; THE 'TRIM PROFILE' CONTROL BUTTOM INTERRUPT
 033C 320000    D   599         STA  INTMSK         ; AND STORE THIS INFORMATION
 033F 01FE1B        600         LXI  B,PFLEND       ; LOAD THE REGISTERS B,C WITH THE STARTING SOURCE ADDRESS
                    601                             ; WHICH POINTS TO THE LOCATION WHERE THE 'END' ADDRESS IS STORED
 0342 11FEBB        602         LXI  D,LASTAD       ; LOAD THE REGISTERS D,E WITH THE STARTING DESTINATION ADDRESS
                    603                             ; WHICH POINTS TO THE LOCATION WHERE THE 'END' ADDRESS IS STORED
 0345 CD4000        604         CALL SETLNK
 0348 CD7500        605         CALL TRNPFL         ; GET THE PROFILE DATA
 034B CD5200        606         CALL OFFLNK
 034E 97            607         SUB  A              ; SET SORTFL TO ZERO INDICATING THAT THE PROFILE HAS NOT
 034F 320200    D   608         STA  SORTFL         ; BEEN SORTED
 0352 FB            609         EI                  ; ENABLE THE 'CUT PROFILE' INTERRUPT ONLY
 0353 0100BC        610         LXI  B,STADD        ; LOAD REGISTERS B,C WITH THE STARTING PROFILE DATA ADDRESS
 0356 2AFEBB        611         LHLD LASTAD         ; LOAD REGISTERS H,L WITH THE ENDING PROFILE DATA ADDRESS
 0359 EB            612         XCHG                ; PUT THIS ADDRESS INTO REGISTERS D,E
 035A CDDD03        613         CALL SORT           ; SORT THE PROFILE ENVELOPE
 035D EB            614         XCHG                ; STORE THE LAST DATA ADDRESS OF THE SORTED ENVELOPE
 035E 2B            615         DCX  H
 035F 22FEBB        616         SHLD LASTAD         ; SAVE THIS ADDRESS IN THE LOCATION LASTAD
 0362 44            617         MOV  B,H            ; COPY THIS INFORMATION TO B,C
 0363 4D            618         MOV  C,L
 0364 2AFEBB        619         LHLD STADD-2        ; GET THE TWO DATA BELOW THE LOCATION STADD
 0367 E5            620         PUSH H              ; SAVE THIS ONTO THE STACK
 0368 0B            621         DCX  B              ; MAKE B,C POINT AT THE LAST 'RAISE' DATA
 0369 0B            622         DCX  B
 036A C5            623         PUSH B              ; SAVE THIS ONTO THE STACK FOR LATER USE
 036B 2100BC        624         LXI  H,STADD        ; GET THE START DATA ADDRESS
 036E 220400    D   625         SHLD STARCV         ; STORE THIS INTO THE LOCATION STARCV
 0371 44            626         MOV  B,H            ; COPY THIS TO B,C
 0372 4D            627         MOV  C,L
 0373 0B            628         DCX  B              ; COPY THE MIN. AND MAX. DATA ONTO THE TWO LOCATION BELOW THE FIRST
 0374 23            629         INX  H
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0375 23            630          INX  H          ; NOW H,L IS POINTING AT THE MAX.
0376 7E            631          MOV  A,M
0377 02            632          STAX B          ; STORE THE MAX.
0378 2B            633          DCX  H          ; NOW H,L IS POINTING AT THE MIN.
0379 0B            634          DCX  B
037A 7E            635          MOV  A,M        ; GET THIS AND
037B 02            636          STAX B          ; STORE IT
037C 2B            637          DCX  H          ; NOW H,L POINT AT THE 'RAISE' DATA
037D 110C00    D   638          LXI  D,DMIN1    ; LOAD D,E WITH THE ADDRESS DMIN1
0380 C1            639 CONTR1:  POP  B          ; GET THE LAST DATA ADDRESS FOR CHECKING
0381 C5            640          PUSH B          ; PUT IT BACK ONTO THE STACK
0382 7C            641          MOV  A,H        ; CHECK IF THE LAST DATA SET HAS BEEN SORTED
0383 B8            642          CMP  B          ; FIRST THE HIGH ORDER BYTE
0384 DA8C03        643          JC   CONTR2     ; CONTINUE IF NOT YET FINISHED
0387 7D            644          MOV  A,L        ; CHECK FOR THE LOW ORDER BYTE IF THE HIGH BYTE ARE EQUAL
0388 B9            645          CMP  C
0389 D2C003        646          JNC  COMPL      ; JUMP OUT IF THE INTERPOLATING HAS BEEN COMPLETED
038C 44            647 CONTR2:  MOV  B,H        ; COPY THE CURRENT 'RAISE' DATA ADDRESS TO B,C
038D 4D            648          MOV  C,L
038E 0B            649          DCX  B          ; THEN DECREMENT IT TO POINT AT THE MIN. DATA
038F 0B            650          DCX  B
0390 23            651          INX  H          ; GET THE CURRENT MIN. DATA AND STORE IT IN DMIN1
0391 7E            652          MOV  A,M
0392 12            653          STAX D          ; D,E POINT AT THE LOCATION DMIN1
0393 23            654          INX  H          ; GET THE CURRENT MAX. DATA AND STORE IT IN ONE LOCATION
0394 13            655          INX  D          ; ABOVE DMIN1
0395 7E            656          MOV  A,M
0396 12            657          STAX D
0397 23            658          INX  H          ; POINT H,L AT THE 'RAISE' OF THE NEXT SET OF DATA
0398 1B            659          DCX  D          ; POINT D,E AT THE DMIN1
0399 23            660          INX  H          ; POINT H,L AT THE MIN. OF NEXT DATA SET OF DATA
039A 0A            661          LDAX B          ; GET THE MIN. BELOW
039B 86            662          ADD  M          ; AND ADD TO THE MIN. ABOVE
039C 1F            663          RAR             ; DIVIDE THE RESULT BY TWO
039D EB            664          XCHG            ;   H,L <---> D,E
039E BE            665          CMP  M          ; COMPARE WITH THE MIN. SETTING
039F D2A303        666          JNC  $+4
03A2 77            667          MOV  M,A        ; RESET THIS MIN. IF THE ACCUMULATOR < M
03A3 EB            668          XCHG            ;  H,L <---> D,E
03A4 23            669          INX  H          ; POINT H,L AT THE MAX. OF NEXT SET OF DATA
03A5 13            670          INX  D          ; POINT D,E AT THE LOCATION ABOVE DMIN1
03A6 03            671          INX  B          ; POINT B,C AT THE MAX. BELOW THE CURRENT DATA SET
03A7 0A            672          LDAX B          ; GET THIS DATA
03A8 86            673          ADD  M          ; AND ADD TO THE NEXT MAX.
03A9 1F            674          RAR             ; FIND THE AVERAGE
03AA EB            675          XCHG            ;  H,L <---> D,E
03AB BE            676          CMP  M          ; COMPARE WITH THE CURRENT MAX. SETTING
03AC DAB003        677          JC   $+4
03AF 77            678          MOV  M,A        ; RESET THE MAX. IF THE ACCUMULATOR > M
03B0 EB            679          XCHG            ;  H,L <---> D,E
03B1 1B            680          DCX  D          ; POINT D,E AT DMIN1 AGAIN
03B2 03            681          INX  B          ; B,C POINT AT THE CURRENT 'RAISE' DATA
03B3 03            682          INX  B          ; B,C POINT AT THE CURRENT MIN SLEW DATA
03B4 1A            683          LDAX D          ; GET DMIN1
03B5 02            684          STAX B          ; AND STORE IT AT THE MIN SLEW POSITION
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

03B6 03           685          INX  B          ; GO ON FOR THE MAX. SLEW
03B7 13           686          INX  D
03B8 1A           687          LDAX D          ; GET DMIN1 + 1 (I.E. THE MAX)
03B9 02           688          STAX B          ; AND STORE IT IN THE CURRENT MAX. SLEW DATA POSITION
03BA 1B           689          DCX  D          ; POINT D,E AT DMIN1 AGAIN
03BB 2B           690          DCX  H          ; POINT H,L AT THE 'RAISE' DATA ADDRESS
03BC 2B           691          DCX  H
03BD C38003       692          JMP  CONTR1     ; LOOP BACK FOR FURTHER SORTING AND CHECKING
03C0 C1           693 COMPL:   POP  B
03C1 E1           694          POP  H          ; RECOVER THE TWO BYTE DATA BELOW STADD
03C2 22FEBB       695          SHLD STADD-2
03C5 CD0905       696          CALL EXPAND     ; CALL THE TABLE EXPANSION ROUTINE
03C8 3EFF         697          MVI  A,0FFH     ; RESET THE SORTFL TO 0FFH
03CA 320200    D  698          STA  SORTFL
03CD 3E18         699          MVI  A,018H     ; RESET THE INTERRUPT MASK TO ENABLE ALL THE FUNCTIONS
03CF 320000    D  700          STA  INTMSK     ; STORE THIS NEW PATTERN
03D2 30           701          SIM             ; PATTERN IS: 0001 1000
03D3 DB01         702          IN   01H        ; RESET THE LED LIGHT PATTERN TO 'STOP CUTTER'
03D5 E6F7         703          ANI  0F7H       ; PATTERN: XXXX 01XX
03D7 F604         704          ORI  004H
03D9 D301         705          OUT  01H
03DB FB           706          EI              ; ENABLE THE SYSTEM INTERRUPT AND
03DC 76           707          HLT             ; WAIT
                  708 $EJECT
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    709 ;*****************************************************************
                    710 ;*     NAME OF ROUTINE: SORT                                     *
                    711 ;*     THIS IS THE ROUTINE SEGMENT TO SORT THE PROFILE DATA      *
                    712 ;*                                                               *
                    713 ;*     ON ENTRY:  REGISTERS B,C --- START PROFILE DATA ADDRESS   *
                    714 ;*                REGISTERS D,E --- LAST PROFILE DATA ADDRESS    *
                    715 ;*     ON EXIT :  REGISTERS B,C --- START ADDRESS OF SORTED DATA  *
                    716 ;*                REGISTERS D,E --- LAST ADDRESS OF SORTED DATA + 1 *
                    717 ;*                                                               *
                    718 ;*     PROFILE DATA ARE STORED IN THE ORDER:                     *
                    719 ;*                ELEVATION, MIN. SLEW, MAX. SLEW, ELEVATION, ...  *
                    720 ;*****************************************************************
                    721 PUBLIC   SORT           ; ROUTINE OPEN TO OTHER SEGMENTS
                    722 ;*****************************************************************
03DD F5             723 SORT:    PUSH  PSW      ; SAVE THE REGISTERS ON ENTRY
03DE E5             724          PUSH  H
03DF C5             725          PUSH  B
03E0 13             726          INX   D
03E1 D5             727          PUSH  D
03E2 C5             728          PUSH  B        ; ADDRESS FOR DESTINATION OF TRANSFER (FOR SORTED DATA)
03E3 C5             729          PUSH  B        ; ADDRESS FOR CURRENT PROFILE DATA (ELEVATION) TO BE CHECKED
03E4 21FDFF         730          LXI   H,-3     ; INITIALIZE THE COMPARING KEY AND STORE IT TO THE MEMORY
03E7 39             731          DAD   SP       ; BELOW THE STACK
03E8 E5             732          PUSH  H        ; SAVE THE ADDRESS FOR THIS KEY
03E9 3600           733          MVI   M,0      ; START THE SORTING FROM ZERO
03EB 78             734 LOOP11:  MOV   A,B      ; CHECK IF THE LAST DATA HAS BEEN SORTED
03EC BA             735          CMP   D        ; BY FIRST LOOKING AT THE HIGH ORDER BYTE OF THE CURRENT ADDRESS
03ED D21A04         736          JNC   CONT11   ; JUMP TO CHECK THE LOW ORDER BYTE IF NOT SMALLER
03F0 E1             737 CONT41:  POP   H        ; GET THE POINTER FOR THE KEY DATA
03F1 0A             738          LDAX  B        ; GET THE CURRENT PROFILE ELEVATION DATA
03F2 BE             739          CMP   M        ; IS KEY EQUAL TO PROFILE?
03F3 C21104         740          JNZ   CONT21   ; JUMP TO CONT2 IF NOT
03F6 03             741          INX   B        ; ADVANCE THE PROFILE DATA POINTER TO
03F7 0A             742          LDAX  B        ; GET THE DATA FOR SLEW POSITION AND
03F8 5F             743          MOV   E,A      ; MOVE IT TO A BUFFER STORE (E)
03F9 0B             744          DCX   B        ; DECREMENT B,C FOR ELEVATION DATA
03FA 0A             745          LDAX  B        ; AND MOVE IT TO ANOTHER BUFFER (D)
03FB 57             746          MOV   D,A
03FC E1             747          POP   H        ; GET THE 'DESTINATION' ADDRESS FROM STACK
03FD E1             748          POP   H
03FE 7E             749          MOV   A,M      ; EXCHANGE DATA AT THE 'DESTINATION' AND THE CURRENT DATA
03FF 02             750          STAX  B        ; BY EXCHANGE   (H,L) <---> (B,C)
0400 72             751          MOV   M,D      ; FIRST THE ELEVATION DATA
0401 23             752          INX   H
0402 03             753          INX   B        ; THEN THE SLEW DATA
0403 7E             754          MOV   A,M
0404 02             755          STAX  B
0405 73             756          MOV   M,E
0406 23             757          INX   H        ; INCREMENT 'DESTINATION' ADDRESS
0407 03             758          INX   B        ; INCREMENT CURRENT ADDRESS FOR CHECKING
0408 D1             759          POP   D        ; GET THE END ADDRESS FOR THE PROFILE DATA LIST
0409 D5             760          PUSH  D        ; PUT THE ADDRESSES ONTO THE STACK
040A E5             761          PUSH  H
040B C5             762          PUSH  B
040C 3B             763          DCX   SP       ; LOCATE THE STACK POINTER TO POINT AT THE MEMORY FOR THE
```

```
 LOC  OBJ        LINE        SOURCE STATEMENT

 040D 3B          764          DCX   SP       ; KEY ADDRESS
 040E C3EB03      765          JMP   LOOP11   ; LOOP BACK FOR OTHER CHECKS
 0411 C1          766 CONT21:  POP   B        ; INCREMENT THE CURRENT ADDRESS FOR NEXT PROFILE DATA
 0412 03          767          INX   B
 0413 03          768          INX   B
 0414 C5          769          PUSH  B        ; STORE THIS NEW VALUE ONTO STACK
 0415 3B          770          DCX   SP       ; ADJUST THE STACK POINTER
 0416 3B          771          DCX   SP
 0417 C3EB03      772          JMP   LOOP11   ; LOOP BACK FOR OTHER CHECKS
 041A C22204      773 CONT11:  JNZ   CONT31   ; JUMP IF ADDRESS > LAST ADDRESS
 041D 79          774          MOV   A,C      ; CHECK IF THE LOW ORDER BYTE OF THE CURRENT DATA POINTER
 041E BB          775          CMP   E        ; IS LESS THAN THAT OF THE LAST DATA ADDRESS
 041F DAF003      776          JC    CONT41   ; JUMP TO CHECK FOR KEY IF YES
 0422 E1          777 CONT31:  POP   H        ; OTHERWISE INCREMENT THE KEY VALUE FOR COMPARISON
 0423 34          778          INR   M
 0424 CA3404      779          JZ    STOP1    ; RE-STORE THE SORTED ENVELOPE IF 0256D STEPS HAVE BEEN CHECKED
 0427 E5          780          PUSH  H        ; OTHERWISE PUT THE ADDRESS FOR KEY ONTO THE STACK
 0428 E1          781          POP   H
 0429 E1          782          POP   H
 042A E1          783          POP   H        ; GET THE 'DESTINATION' ADDRESS AND
 042B E5          784          PUSH  H
 042C E5          785          PUSH  H        ; DUPLICATE FOR CURRENT ADDRESS USED IN CHECKING
 042D 4D          786          MOV   C,L      ; AND PUT THIS ADDRESS TO B,C
 042E 44          787          MOV   B,H
 042F 3B          788          DCX   SP       ; ADJUST THE STACK POINTER
 0430 3B          789          DCX   SP
 0431 C3EB03      790          JMP   LOOP11   ; LOOP BACK FOR CHECKING
 0434 D1          791 STOP1:   POP   D        ; SORTING COMPLETED
 0435 D1          792          POP   D
 0436 D1          793          POP   D        ; RECOVER REGISTERS TO CONTINUE
 0437 C1          794          POP   B        ; GET THE START ADDRESS OF THE PROFILE DATA LIST
 0438 C5          795          PUSH  B        ; RE-SAVE THE REGISTERS B,C
 0439 D5          796          PUSH  D        ; LAST ADDRESS OF THE PROFILE DATA + 1
 043A C5          797          PUSH  B        ; ADDRESS FOR DESTINATION OF TRANSFER
 043B C5          798          PUSH  B        ; ADDRESS FOR CURRENT PROFILE TO BE CHECKED
 043C 21F7FF      799          LXI   H,-9     ; INITIALIZE THE COMPARING KEY AND STORE
 043F 39          800          DAD   SP       ; IT TO THE ADDRESS BELOW THE STACK
 0440 E5          801          PUSH  H        ; SAVE THIS KEY ADDRESS
 0441 0A          802 LOOP12:  LDAX  B        ; GET DATA FOR ELEVATION
 0442 77          803          MOV   M,A      ; PUT IT INTO THE WORKING STORE
 0443 2B          804          DCX   H        ; INITIALIZE THE MINIMUM AND MAXIMUM DATA
 0444 36FF        805          MVI   M,0FFH   ; CURRENT MINIMUM
 0446 2B          806          DCX   H
 0447 3600        807          MVI   M,000H   ; CURRENT MAXIMUM
 0449 E1          808 CONT42:  POP   H        ; SEGMENT TO ADJUST MINIMUM AND MAXIMUM
 044A C1          809          POP   B
 044B 0A          810          LDAX  B
 044C BE          811          CMP   M        ; COMPARE ELEVATION DATA
 044D C28604      812          JNZ   CONT12   ; JUMP IF DATA HAS CHANGED
 0450 2B          813          DCX   H
 0451 03          814          INX   B
 0452 0A          815          LDAX  B        ; GET CURRENT MINIMUM VALUE
 0453 BE          816          CMP   M
 0454 D25804      817          JNC   CONT22   ; JUMP IF NOT LESS
 0457 77          818          MOV   M,A      ; SET THIS NEW MINIMUM
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0458 2B            819 CONT22: DCX   H
0459 BE            820         CMP   M        ; COMPARE WITH CURRENT MAXIMUM
045A DA5E04        821         JC    CONT32   ; JUMP IF LESS
045D 77            822         MOV   M,A      ; SET THIS NEW MAXIMUM
045E 03            823 CONT32: INX   B        ; INCREMENT CURRENT ADDRESS FOR CHECK
045F C5            824         PUSH  B        ; STORE THIS CURRENT ADDRESS ON STACK
0460 3B            825         DCX   SP       ; ADJUST STACK POINTER
0461 3B            826         DCX   SP
0462 78            827         MOV   A,B
0463 BA            828         CMP   D
0464 DA4904        829         JC    CONT42   ; SKIP IF LESS
0467 C26F04        830         JNZ   STOP2    ; STOP IF GREATER
046A 79            831         MOV   A,C      ; COMPARE LOW ORDER BYTE ADDRESS
046B BB            832         CMP   E
046C DA4904        833         JC    CONT42   ; SKIP IF LESS
046F E1            834 STOP2:  POP   H        ; GET 'SOURCE' ADDRESS
0470 C1            835         POP   B
0471 C1            836         POP   B        ; GET 'DESTINATION' ADDRESS
0472 3B            837         DCX   SP       ; ADJUST THE STACK POINTER
0473 3B            838         DCX   SP
0474 3B            839         DCX   SP
0475 3B            840         DCX   SP
0476 3B            841         DCX   SP
0477 3B            842         DCX   SP
0478 CDB104        843         CALL  CSTORE   ; STORE THE 'CAVITY PROFILE' DATA
047B 50            844 CONT62: MOV   D,B      ; ASSIGN END ADDRESS TO D,E
047C 59            845         MOV   E,C
047D 210800        846         LXI   H,8      ; ADJUST THE STACK POINTER
0480 39            847         DAD   SP
0481 F9            848         SPHL
0482 C1            849         POP   B        ; RECOVER REGISTERS BEFORE RETURNING
0483 E1            850         POP   H
0484 F1            851         POP   PSW
0485 C9            852         RET
                   853 $EJECT
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    854 ;*****************************************************
                    855 ;*       DATA FOR ELEVATION HAS CHANGED              *
                    856 ;*****************************************************
 0486 C1            857 CONT12: POP   B         ; GET CAVITY PROFILE ADDRESS
 0487 3B            858         DCX   SP        ; ADJUST THE STACK POINTER
 0488 3B            859         DCX   SP
 0489 3B            860         DCX   SP
 048A 3B            861         DCX   SP
 048B 3B            862         DCX   SP
 048C 3B            863         DCX   SP
 048D CDB104        864         CALL  CSTORE    ; STORE THE CAVITY PROFILE DATA
 0490 78            865         MOV   A,B       ; CHECK IF THE LAST PROFILE DATA HAS BEEN STORED BY FIRST
 0491 BA            866         CMP   D         ; LOOKING AT THE HIGH ORDER BYTE
 0492 DA9D04        867         JC    CONT52    ; SKIP IF THE HIGH ORDER BYTE IS LESS
 0495 C27B04        868         JNZ   CONT62    ; STOP IF GREATER
 0498 79            869         MOV   A,C       ; CHECK IF THE LOW ORDER BYTE IS LESS
 0499 BB            870         CMP   E
 049A D27B04        871         JNC   CONT62    ; STOP IF NOT
 049D 210600        872 CONT52: LXI   H,6       ; ADJUST THE STACK POINTER
 04A0 39            873         DAD   SP
 04A1 F9            874         SPHL
 04A2 C5            875         PUSH  B         ; THEN PUSH THE NEW 'DESTINATION' ADDRESS ONTO THE STACK
 04A3 21FCFF        876         LXI   H,-4      ; ADJUST THE STACK POINTER
 04A6 39            877         DAD   SP
 04A7 F9            878         SPHL
 04A8 E1            879         POP   H         ; RECOVER THE KEY ADDRESS
 04A9 C1            880         POP   B         ; GET THE CURRENT ADDRESS
 04AA 3B            881         DCX   SP        ; ADJUST THE STACK POINTER
 04AB 3B            882         DCX   SP
 04AC 3B            883         DCX   SP
 04AD 3B            884         DCX   SP
 04AE C34104        885         JMP   LOOP12    ; LOOP BACK FOR THIS NEW ELEVATION VALUE
                    886 $EJECT
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    887 ;*****************************************************************
                    888 ;*       ROUTINE TO STORE THE SORTED CAVITY PROFILE DATA IN THE     *
                    889 ;*       FORM OF:    ELEVATION, MIN. SLEW, MAX. SLEW,              *
                    890 ;*       ON ENTRY:   REGISTERS B,C --- DESTINATION ADDRESS FOR THE  *
                    891 ;*                                    ELEVATION DATA               *
                    892 ;*                   REGISTERS H,L --- ADDRESS OF THE ELEVATION TO  *
                    893 ;*                                    BE STORED                    *
                    894 ;*       ON EXIT:    REGISTERS B,C --- NEXT AVAILABLE DESTINATION   *
                    895 ;*                                    ADDRESS FOR ELEVATION DATA   *
                    896 ;*****************************************************************
 04B1 F5            897 CSTORE:  PUSH  PSW     ; SAVE THE PROCESSOR STATUS WORD ON ENTRY
 04B2 210600        898          LXI   H,6     ; CHECK IF DUPLICATION OF DATA IS REQUIRED
 04B5 39            899          DAD   SP      ; FIRST GET THE ADDRESS FOR THE NEXT PROFILE DATA TO
 04B6 F9            900          SPHL          ; BE CHECKED AND COMPARE THIS WITH THE 'DESTINATION ADDRESS'
 04B7 E1            901          POP   H       ;      DUPLICATION WILL BE REQUIRED IF THERE IS NOT ENOUGH GAP
 04B8 03            902          INX   B       ; INCREMENT THE 'DESTINATION ADDRESS' TO GIVE UPPER BOUND
 04B9 03            903          INX   B
 04BA 78            904          MOV   A,B     ; COMPARE THE HIGH ORDER BYTE
 04BB BC            905          CMP   H
 04BC DAE604        906          JC    NODUP   ; NO DUPLICATION WILL BE REQUIRED IF LESS
 04BF C2C704        907          JNZ   DUPLI   ; DUPLICATION IS REQUIRED IF GREATER
 04C2 79            908          MOV   A,C     ; COMPARE THE LOW ORDER BYTE IF HIGH ORDER BYTES EQUAL
 04C3 BD            909          CMP   L
 04C4 DAE604        910          JC    NODUP   ; NO DUPLICATION WILL BE REQUIRED IF LESS
 04C7 33            911 DUPLI:   INX   SP      ; INCREMENT THE LAST PROFILE DATA ADDRESS AND STORE IT
 04C8 33            912          INX   SP      ; ONTO THE STACK
 04C9 33            913          INX   SP
 04CA 33            914          INX   SP
 04CB 13            915          INX   D
 04CC D5            916          PUSH  D
 04CD 1B            917          DCX   D
 04CE 1A            918 DUPLI1:  LDAX  D       ; COPY DATA TO THE LOCATION JUST ABOVE IT
 04CF 13            919          INX   D
 04D0 12            920          STAX  D
 04D1 1B            921          DCX   D       ; CHECK IF THE LAST DATA HAS BEEN DUPLICATED
 04D2 1B            922          DCX   D
 04D3 7C            923          MOV   A,H     ; COMPARE THE HIGH ORDER ADDRESS BYTE
 04D4 BA            924          CMP   D
 04D5 DACE04        925          JC    DUPLI1  ; LOOP BACK FOR FURTHER DUPLICATION
 04D8 C2E004        926          JNZ   DUPLI2  ; JUMP OUT IF EXCEED
 04DB 7D            927          MOV   A,L     ; COMPARE THE LOW ORDER ADDRESS BYTE IF THE HIGH ORDER
 04DC BB            928          CMP   E       ; ADDRESS BYTES ARE EQUAL
 04DD DACE04        929          JC    DUPLI1  ; LOOP BACK IF NOT YET COMPLETED THE DUPLICATION
 04E0 23            930 DUPLI2:  INX   H       ; INCREMENT THE NEXT CHECK DATA ADDRESS
 04E1 3B            931          DCX   SP
 04E2 3B            932          DCX   SP
 04E3 E5            933          PUSH  H       ; PUSH THIS ADDRESS ONTO THE STACK
 04E4 33            934          INX   SP      ; ADJUST THE STACK POINTER
 04E5 33            935          INX   SP
 04E6 3B            936 NODUP:   DCX   SP      ; ADJUST THE STACK POINTER
 04E7 3B            937          DCX   SP
 04E8 3B            938          DCX   SP
 04E9 3B            939          DCX   SP
 04EA E1            940          POP   H       ; RECOVER THE H,L REGISTERS FROM THE STACK
 04EB C1            941          POP   B
```

```
LOC  OBJ       LINE      SOURCE STATEMENT

04EC C1          942       POP   B       ; GET THE STARTING 'DESTINATION ADDRESS'
04ED D1          943       POP   D       ; GET THE LAST PROFILE DATA ADDRESS
                 944       REPT  12D
                 945       DCX   SP      ;; RE-ADJUST THE STACK POINTER BY 12 DECREMENTS
                 946       ENDM
04EE 3B          947+      DCX   SP
04EF 3B          948+      DCX   SP
04F0 3B          949+      DCX   SP
04F1 3B          950+      DCX   SP
04F2 3B          951+      DCX   SP
04F3 3B          952+      DCX   SP
04F4 3B          953+      DCX   SP
04F5 3B          954+      DCX   SP
04F6 3B          955+      DCX   SP
04F7 3B          956+      DCX   SP
04F8 3B          957+      DCX   SP
04F9 3B          958+      DCX   SP
04FA 7E          959       MOV   A,M     ; GET THE FIRST SOURCE DATA (ELEVATION)
04FB 02          960       STAX  B       ; STORE THIS DATA INTO THE DESTINATION
04FC 2B          961       DCX   H       ; DECREMENT THE SOURCE DATA POINTER
04FD 03          962       INX   B       ; INCREMENT THE DESTINATION DATA POINTER
04FE 7E          963       MOV   A,M     ; TRANSFER DATA
04FF 02          964       STAX  B
0500 2B          965       DCX   H       ; MOVE SOURCE DATA POINTER TO THE MAXIMUM SLEW
0501 03          966       INX   B       ; INCREMENT THE CAVITY PROFILE POINTER
0502 7E          967       MOV   A,M
0503 02          968       STAX  B       ; DATA TRANSFER COMPLETED
0504 03          969       INX   B       ; INCREMENT THE DESTINATION ADDRESS
0505 23          970       INX   H       ; RECOVER H,L TO POINT AT THE 'RAISE' DATA KEY
0506 23          971       INX   H
0507 F1          972       POP   PSW     ; RECOVER THE PROCESSOR STATUS WORD ON EXIT
0508 C9          973       RET
                 974 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   975 ;*****************************************************************
                   976 ;*      A ROUTINE TO EXPAND THE TABLE OF SORTED PROFILE ENVELOPE  *
                   977 ;*      ON ENTRY:  LASTAD CONTAINS THE LAST ADDRESS FOR THE DATA  *
                   978 ;*                 STADD IS THE STARTING ADDRESS FOR THE DATA     *
                   979 ;*      ON EXIT:   LASTCV CONTAINS THE LAST ADDRSS OF THE TABLE   *
                   980 ;*                 STARCV CONTAINS THE START ADDRESS OF THE TABLE *
                   981 ;*                 SORTFL IS SET TO 0FFH                         *
                   982 ;*****************************************************************
                   983 PUBLIC  EXPAND
0509 F5            984 EXPAND: PUSH PSW        ; SAVE THE REGISTERS ON ENTRY
050A C5            985         PUSH B
050B D5            986         PUSH D
050C E5            987         PUSH H
050D 2AFEBB        988         LHLD LASTAD     ; LOAD H,L WITH THE LAST ADDRESS OF THE SORTED ENVELOPE
0510 2B            989         DCX  H
0511 2B            990         DCX  H          ; D,E POINT TO THE 'RAISE' DATA
0512 7E            991         MOV  A,M        ; GET THIS 'RAISE' DATA
0513 6F            992         MOV  L,A        ; MULTIPLY THIS DATA BY 4 USING 'DAD' STATEMENT
0514 2600          993         MVI  H,000H
0516 29            994         DAD  H
0517 29            995         DAD  H
0518 0100BC        996         LXI  B,STADD    ; EVALUATE THIS LAST 'CAVITY RAISE' DATA ADDRESS
051B 09            997         DAD  B
051C 220600    D   998         SHLD LASTCV     ; STORE THIS ADDRESS TO LASTCV
051F 23            999         INX  H          ; LOCATE H,L TO THE STORAGE FOR MAX. SLEW
0520 23           1000         INX  H
0521 EB           1001         XCHG            ; AND EXCHANGE WITH THE D,E PAIR
0522 2AFEBB       1002         LHLD LASTAD     ; GET THE ADDRESS FOR THE 'UNEXPANDED' DATA
0525 7E           1003 EXPA1: MOV  A,M        ; GET THE DATA
0526 12           1004         STAX D          ; STORE IN THE NEW LOCATION
0527 2B           1005         DCX  H          ; STORE THE MIN. SLEW DATA
0528 1B           1006         DCX  D
0529 7E           1007         MOV  A,M
052A 12           1008         STAX D
052B 2B           1009         DCX  H          ; STORE THE 'RAISE' DATA
052C 1B           1010         DCX  D
052D 7E           1011         MOV  A,M
052E 12           1012         STAX D
052F F5           1013         PUSH PSW        ; CHECK IF THE FIRST DATA SET HAS BEEN TRANSFERRED
0530 0100BC       1014         LXI  B,STADD
0533 7C           1015         MOV  A,H
0534 B8           1016         CMP  B
0535 C23D05       1017         JNZ  EXPA2
0538 7D           1018         MOV  A,L
0539 B9           1019         CMP  C
053A CA6E05       1020         JZ   EXPAF
053D F1           1021 EXPA2: POP  PSW
053E 2B           1022         DCX  H
053F 2B           1023         DCX  H
0540 2B           1024         DCX  H
0541 96           1025 EXPA3: SUB  M          ; CHECK IF INTERPOLATION IS NEEDED IN EXPANDING
0542 FE01         1026         CPI  01H        ; DIFFERENCE = 1?
0544 C24E05       1027         JNZ  EXPA4
0547 23           1028         INX  H
0548 23           1029         INX  H
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0549 1B           1030          DCX  D
 054A 1B           1031          DCX  D
 054B C32505       1032          JMP  EXPA1
 054E E5           1033 EXPA4:   PUSH H          ; SAVE THE REGISTERS H,L
 054F C5           1034          PUSH B          ; AND B,C
 0550 23           1035          INX  H          ; POINT H,L AT THE MIN.
 0551 13           1036          INX  D          ; POINT D,E AT THE MIN.
 0552 1A           1037          LDAX D          ; GET THIS VALUE
 0553 86           1038          ADD  M          ; AND ADD TO THE MIN. OF THE SET BELOW
 0554 1F           1039          RAR             ; AVERAGING
 0555 4F           1040          MOV  C,A        ; STORE THIS IN C
 0556 23           1041          INX  H          ; DO THE SAME FOR THE MAX.
 0557 13           1042          INX  D
 0558 1A           1043          LDAX D
 0559 86           1044          ADD  M          ; ADD THE MAXIMUMS TOGETHER
 055A 1F           1045          RAR             ; THEN AVERAGING
 055B 47           1046          MOV  B,A        ; STORE THIS IN B
 055C 1B           1047          DCX  D          ; POINT D,E AT THE NEXT DATA SET BELOW
 055D 1B           1048          DCX  D
 055E 1A           1049          LDAX D          ; GET THIS 'RAISE' DATA SETTING
 055F 1B           1050          DCX  D
 0560 1B           1051          DCX  D          ; NOW AT THE MAX.
 0561 EB           1052          XCHG            ; H,L <---> D,E
 0562 70           1053          MOV  M,B        ; SET THE MAX.
 0563 2B           1054          DCX  H
 0564 71           1055          MOV  M,C        ; SET THE MIN.
 0565 2B           1056          DCX  H
 0566 3D           1057          DCR  A
 0567 77           1058          MOV  M,A        ; SET THE 'RAISE' DATA
 0568 EB           1059          XCHG            ; H,L <---> D,E
 0569 C1           1060          POP  B          ; RECOVER THE REGISTERS
 056A E1           1061          POP  H
 056B C34105       1062          JMP  EXPA3
 056E F1           1063 EXPAF:   POP  PSW
 056F EB           1064          XCHG
 0570 220400    D  1065          SHLD STARCV
 0573 E1           1066          POP  H
 0574 D1           1067          POP  D
 0575 C1           1068          POP  B
 0576 F1           1069          POP  PSW
 0577 C9           1070          RET
                   1071 $EJECT
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                   1072 ;*********************************************************************
                   1073 ;*      NAME OF ROUTINE: SHORT                                       *
                   1074 ;*      A SUB-ROUTINE TO FIND THE NEAREST STARTING POINT FROM THE    *
                   1075 ;*      CURRENT CUTTER POSITION AS FAR AS THE SLEW IS CONCERNED      *
                   1076 ;*      ON ENTRY: B,C POINT TO THE 'RAISE' DATA OF THE BLOCK         *
                   1077 ;*      ON EXIT: THE REQUIRED SLEW POSITIONS ARE SORTED AND STORED   *
                   1078 ;*               IN FIRSL WITH THE FIRST OCCUPYING THE LOW POSITION  *
                   1079 ;*********************************************************************
                   1080 PUBLIC  SHORT,FIRSL,FOURST
                   1081 ;*********************************************************************
 0578 F5           1082 SHORT:  PUSH PSW        ; SAVE THE REGISTERS ON ENTRY
 0579 E5           1083         PUSH H
 057A D5           1084         PUSH D
 057B 69           1085         MOV  L,C        ; COPY REGISTERS B,C TO H,L
 057C 60           1086         MOV  H,B
 057D F3           1087         DI              ; DISABLE THE SYSTEM INTERRUPT DURING DATA TRANSFER
 057E CD4000       1088         CALL SETLNK     ; GET THE CURRENT SLEW DATA
 0581 3AA21B       1089         LDA  CRRTSL
 0584 5F           1090         MOV  E,A        ; AND STORE IT IN THE REGISTER E
 0585 CD5200       1091         CALL OFFLNK
 0588 FB           1092         EI              ; ENABLE THE SYSTEM INTERRUPT AFTER THE DATA TRANSFER
 0589 7B           1093         MOV  A,E        ; GET THE CURRENT SLEW POSITION
 058A 322300   D   1094         STA  SLCRRT     ; AND STORE IT IN THE MEMORY LOCATION 'SLCRRT'
 058D 23           1095         INX  H          ; FIND THE ABSOLUTE DISTANCE BETWEEN CURRENT AND THE NEXT MIN.
 058E 96           1096         SUB  M          ; EVALUATE THE DISTANCE (DIFFERENCE IN SLEW)
 058F D29405       1097         JNC  $+5
 0592 2F           1098         CMA             ; EVALUATE THE ABSOLUTE BY TWO'S COMPLEMENT
 0593 3C           1099         INR  A
 0594 57           1100         MOV  D,A        ; STORE THE ABSOLUTE DISTANCE IN REGISTER D
 0595 23           1101         INX  H          ; FIND THE ABSOLUTE DISTANCE BETWEEN CURRENT AND THE NEXT MAX.
 0596 7B           1102         MOV  A,E        ; GET THE CURRENT SLEW POSITION FROM E
 0597 96           1103         SUB  M          ; EVALUATE THE DISTANCE (DIFFERENCE IN SLEW)
 0598 D29D05       1104         JNC  $+5
 059B 2F           1105         CMA
 059C 3C           1106         INR  A
 059D BA           1107         CMP  D          ; COMPARE TO CHECK WHICH IS THE SHORTEST
 059E 111000   D   1108         LXI  D,FIRSL    ; GET THE ADDRESS FOR STORING THE FIRST SLEW DATA
 05A1 DAAC05       1109         JC   CONT5
 05A4 2B           1110         DCX  H          ; THE PREVIOUS ONE IS THE SHORTEST
 05A5 7E           1111         MOV  A,M
 05A6 12           1112         STAX D          ; FIRST REQUIRED SLEW IS FOLLOWED BY THE SECOND ONE
 05A7 23           1113         INX  H
 05A8 13           1114         INX  D
 05A9 C3B005       1115         JMP  CONT6
 05AC 7E           1116 CONT5:  MOV  A,M        ; THE CURRENT ONE IS THE NEAREST TO THE CUTTER
 05AD 12           1117         STAX D
 05AE 13           1118         INX  D
 05AF 2B           1119         DCX  H
 05B0 7E           1120 CONT6:  MOV  A,M
 05B1 12           1121         STAX D
 05B2 D1           1122         POP  D          ; RECOVER THE REGISTERS BEFORE RETURNING
 05B3 E1           1123         POP  H
 05B4 F1           1124         POP  PSW
 05B5 C9           1125         RET
                   1126 ;
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                  1127 ;*************************************************************
                  1128 ;*     A ROUTINE TO REDUCE THE ENVELOPE TO A SMALLER SIZE BY HALF   *
                  1129 ;*     THE CUTTER ADVANCE STEP (I.E. HALF OF CUTSTP)                *
                  1130 ;*************************************************************
05B6 F5           1131 REDUCE: PUSH PSW      ; SAVE THE REGISTERS ON ENTRY
05B7 E5           1132         PUSH H
05B8 D5           1133         PUSH D
05B9 CD7805       1134         CALL SHORT    ; ON ENTRY:  B,C POINT TO THE 'RAISE' DATA
05BC 2A1000   D   1135         LHLD FIRSL    ; GET THE SLEW BOUNDS
05BF EB           1136         XCHG          ;   H,L <---> D,E
05C0 7A           1137         MOV  A,D      ; CHECK IF REDUCTION IS REQUIRED
05C1 93           1138         SUB  E        ; BY COMPARING THE ABSOLUTE DIFFERENCE BETWEEN BOUNDS AND
05C2 D2C705       1139         JNC  $+5      ; THE CUTTER ADVANCE STEP  (CUTSTP)
05C5 2F           1140         CMA
05C6 3C           1141         INR  A
05C7 210100   D   1142         LXI  H,CUTSTP ; POINT H,L TO THE CUTSTP FOR COMPARISION
05CA BE           1143         CMP  M
05CB DAE205       1144         JC   NORED    ; SKIP THE REDUCTION SINCE THE SEPERATION IS LESS
05CE 211600   D   1145         LXI  H,HALFST ; OTHERWISE REDUCE THE BOUNDARY DATA
05D1 7A           1146         MOV  A,D      ; CHECK THE WAY OF REDUCING THE BOUNDS
05D2 BB           1147         CMP  E
05D3 DADD05       1148         JC   REDU1    ; JUMP TO REDUCE E AND INCREASE D
05D6 96           1149         SUB  M        ; OTHERWISE REDUCE D AND INCREASE E BY HALFST
05D7 57           1150         MOV  D,A      ; NO CHECKING FOR OVERFLOW/UNDERFLOW SINCE DATA ARE AUTOMATICALLY
05D8 7B           1151         MOV  A,E      ; WITHIN THE BOUNDS
05D9 86           1152         ADD  M
05DA C3E105       1153         JMP  REDU2    ; JUMP TO CONTINUE
05DD 86           1154 REDU1:  ADD  M        ; INCREASE D AND REDUCE E
05DE 57           1155         MOV  D,A
05DF 7B           1156         MOV  A,E
05E0 96           1157         SUB  M
05E1 5F           1158 REDU2:  MOV  E,A
05E2 EB           1159 NORED:  XCHG          ; PUT THE DATA BACK TO H,L
05E3 221000   D   1160         SHLD FIRSL    ; STORE THEM BACK TO THE LOCATIONS
05E6 D1           1161         POP  D
05E7 E1           1162         POP  H
05E8 F1           1163         POP  PSW
05E9 C9           1164         RET
                  1165 $EJECT
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

                    1166 RC)
                   =1167 ;*************************************************************************
                   =1168 ;*      A ROUTINE TO MOVE THE BOOM SLOWLY TO THE DESIRED POSITION        *
                   =1169 ;*************************************************************************
                   =1170 PUBLIC  SMOVE,SMOVE1,SMOVEF
                   =1171 ;*************************************************************************
 05EA F5           =1172 SMOVE:  PUSH PSW      ; SAVE THE REGISTERS
 05EB E5           =1173         PUSH H
 05EC D5           =1174         PUSH D
 05ED C5           =1175         PUSH B
 05EE DB01         =1176 FMOVES: IN   01H      ; GET THE CURRENT VALVE STATUS
 05F0 F603         =1177         ORI  003H     ; SET THE SPEED CONTROL VALVE TO SLOW MODE BY ANDING IT WITH 00000011B
 05F2 321700    D  =1178 SMOVEF: STA  MDSIGN   ; RESET THIS PORT PATTERN
 05F5 3A1D00    D  =1179         LDA  FORCE    ; GET THE CUTTER MOTOR CURRENT
 05F8 FE1A         =1180         CPI  01AH     ; COMPARE WITH THE LIMIT FOR FAST MOVE MODE
 05FA DA0D06       =1181         JC   ANYMOD   ; PERMIT ALL IF LESS
 05FD 3A1700    D  =1182         LDA  MDSIGN   ; OTHERWISE, SET THE SLOW MOVE MODE
 0600 F603         =1183         ORI  003H
 0602 321700    D  =1184         STA  MDSIGN
 0605 3A1D00    D  =1185         LDA  FORCE    ; GET THE CUTTER MOTOR CURRENT
 0608 FEE5         =1186         CPI  0E5H     ; COMPARE WITH THE MAXIMUM PERMITTED CURRENT
 060A D23807       =1187         JNC  STOPMD   ; JUMP TO STOP THE BOOM IF THE FORCE IS GREATER
 060D 2A1A00    D  =1188 ANYMOD: LHLD SLDESR   ; LOAD THE REGISTER L WITH THE DESIRED SLEW VALUE
                   =1189                       ;  AND THE REGISTER H WITH THE DESIRED RAISE VALUE
 0610 EB           =1190         XCHG          ; EXCHANGE REGISTERS D,E WITH REGISTERS H,L
 0611 2A2300    D  =1191         LHLD SLCRRT   ; LOAD THE REGISTER L WITH THE CURRENT SLEW VALUE
                   =1192                       ;  AND THE REGISTER H WITH THE CURRENT RAISE VALUE
 0614 7D           =1193         MOV  A,L      ; LOAD THE ACCUMULATOR WITH THE CURRENT SLEW VALUE
 0615 93           =1194         SUB  E        ; SUBTRACT THE DESIRED SLEW VALUE FROM THIS CURRENT VALUE
 0616 17           =1195         RAL           ; ROTATE LEFT TO KEEP THE CARRY FLAG
 0617 5F           =1196         MOV  E,A      ; MOVE THIS RESULT TO E
 0618 1F           =1197         RAR           ; ROTATE RIGHT TO RECOVER THE ORIGINAL VALUE
 0619 D21E06       =1198         JNC  $+5
 061C 2F           =1199         CMA           ; FIND THE ABSOLUTE DISTANCE BETWEEN THE TWO SLEW VALUES
 061D 3C           =1200         INR  A
 061E FE02         =1201         CPI  02H
 0620 E5           =1202         PUSH H        ; SAVE H,L
 0621 210300    D  =1203         LXI  H,SIGNAL
 0624 DA3606       =1204         JC   CLOSL
 0627 7B           =1205         MOV  A,E      ; CHECK FOR THE DIRECTION OF SLEW REQUIRED
 0628 1F           =1206         RAR
 0629 DA3106       =1207         JC   RIGHT1
 062C 366F         =1208 LEFT1:  MVI  M,06FH   ; SET THE MOVE LEFT SIGNAL
 062E C33806       =1209         JMP  RSCMP
 0631 369F         =1210 RIGHT1: MVI  M,09FH   ; SET THE MOVE RIGHT SIGNAL
 0633 C33806       =1211         JMP  RSCMP
 0636 36FF         =1212 CLOSL:  MVI  M,0FFH   ; SET THE NO-MOVE SIGNAL
 0638 3A0200    D  =1213 RSCMP:  LDA  SORTFL   ; CHECK IF THE PROFILE DATA HAVE BEEN SORTED
 063B 17           =1214         RAL
 063C D25106       =1215         JNC  NTSORT   ; SKIP IF THE DATA HAVE NOT BEEN SORTED
 063F 2A0400    D  =1216         LHLD STARCV   ; GET THE LOWEST LIMIT FOR THE BOOM RAISE
 0642 7E           =1217         MOV  A,M
 0643 BA           =1218         CMP  D        ; COMPARE THIS VALUE WITH THE DESIRED VALUE
 0644 DA4806       =1219         JC   $+4
 0647 57           =1220         MOV  D,A      ; RESET THIS VALUE IF DESIRED < LOWEST BOUND
```

CONTROL WITH SIMPLE COMPARISION

```
LOC  OBJ          LINE        SOURCE STATEMENT

0648 2A0600   D =1221          LHLD LASTCV    ; GET THE UPPEST LIMIT FOR THE BOOM RAISE
064B 7E         =1222          MOV  A,M
064C BA         =1223          CMP  D         ; COMPARE THIS VALUE WITH THE DESIRED VALUE
064D D25106     =1224          JNC  $+4
0650 57         =1225          MOV  D,A       ; RESET THIS VALUE IF DESIRED > HIGHEST BOUND
0651 E1         =1226 NTSORT: POP  H           ; RECOVER THE H,L REGISTERS
0652 7C         =1227          MOV  A,H       ; CHECK FOR DIRECTION OF VERTICAL MOVEMENT REQUIRED
0653 92         =1228          SUB  D         ; BY COMPARING THE CURRENT AND DESIRED RAISE VALUES
0654 17         =1229          RAL            ; SAVE THE CARRY FLAG IN THE
0655 57         =1230          MOV  D,A       ; LSB OF THE REGISTER D
0656 1F         =1231          RAR
0657 D25C06     =1232          JNC  $+5
065A 2F         =1233          CMA
065B 3C         =1234          INR  A
065C FE02       =1235          CPI  02H
065E DA7E06     =1236          JC   CLOSR
0661 7A         =1237          MOV  A,D       ; SEE WHAT IS THE APPROPRIATE MOVEMENT --- UP OR DOWN
0662 1F         =1238          RAR
0663 110300   D =1239          LXI  D,SIGNAL  ; LOAD D,E WITH SIGNAL ADDRESS
0666 1A         =1240          LDAX D         ; GET THE SIGNAL
0667 DA7406     =1241          JC   UP1
066A E6F6       =1242 DOWN1:  ANI  0F6H      ; SET THE MOVE DOWN SIGNAL
066C 12         =1243          STAX D         ; STORE THIS NEW SIGNAL
066D 3A2400   D =1244          LDA  RSCRRT    ; GET THE CURRENT RAISE DATA
0670 3D         =1245          DCR  A
0671 C38106     =1246          JMP  CHKBND
0674 E6F9       =1247 UP1:    ANI  0F9H      ; SET THE MOVE UP SIGNAL
0676 12         =1248          STAX D         ; STORE THIS NEW SIGNAL
0677 3A2400   D =1249          LDA  RSCRRT    ; GET THE CURRENT RAISE DATA
067A 3C         =1250          INR  A
067B C38106     =1251          JMP  CHKBND
067E 3A2400   D =1252 CLOSR:  LDA  RSCRRT    ; GET THE CURRENT RAISE DATA
0681 F5         =1253 CHKBND: PUSH PSW         ; SAVE THE CURRENT PROCESSOR STATUS WORD
0682 3A0200   D =1254          LDA  SORTFL    ; CHECK IF THE PROFILE DATA HAVE BEEN SORTED
0685 17         =1255          RAL
0686 D22807     =1256          JNC  NTSRT1    ; SKIP IF THE PROFILE DATA HAVE NOT BEEN SORTED
0689 F1         =1257          POP  PSW       ; RECOVER THE PROCESSOR STATUS WORD
068A 6F         =1258          MOV  L,A       ; FIND THE ADDRESS IN THE SORTED TABLE
068B 2600       =1259          MVI  H,000H    ; FOR THIS 'RAISE' LEVEL
068D 29         =1260          DAD  H         ; THIS ADDRESS = STADD + 4*RAISE
068E 29         =1261          DAD  H
068F 0100BC     =1262          LXI  B,STADD
0692 09         =1263          DAD  B
0693 EB         =1264          XCHG           ; TEMPORARILY STORE IN D,E
0694 2A0600   D =1265          LHLD LASTCV    ; LIMIT THE MAXIMUM ADDRESS TO LASTCV
0697 7A         =1266          MOV  A,D
0698 BC         =1267          CMP  H
0699 DAA506     =1268          JC   CHKB1     ; NO NEW SETTING IF LESS THAN (<)
069C C2A406     =1269          JNZ  CHKB2     ; SET NEW ADDRESS IF GREATER  (>)
069F 7B         =1270          MOV  A,E       ; CHECK FOR THE LOW ORDER BYTE IF EQUAL (=)
06A0 BD         =1271          CMP  L
06A1 DAA506     =1272          JC   CHKB1     ; NO NEW SETTING IF LESS (<)
06A4 EB         =1273 CHKB2:  XCHG           ; OTHERWISE SET THIS MAXIMUM ADDRESS (>=)
06A5 2A0400   D =1274 CHKB1:  LHLD STARCV    ; LIMIT THE MINIMUM ADDRESS TO THE STARCV BLOCK
06A8 7A         =1275          MOV  A,D       ;
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

06A9 BC           =1276         CMP   H
06AA DAB506       =1277         JC    CHKB3     ; NEW SETTING IF CURRENT IS LESS    (<)
06AD C2B606       =1278         JNZ   CHKB4     ; NO SETTING IF CURRENT IS GREATER (>)
06B0 7B           =1279         MOV   A,E       ; CHECK FOR THE LOW ORDER BYTE      (=)
06B1 BD           =1280         CMP   L
06B2 D2B606       =1281         JNC   CHKB4     ; NO SETTING IF NOT LESS THAN       (>=)
06B5 EB           =1282 CHKB3:  XCHG            ; SET THE NEW ADDRESS VALUE
06B6 EB           =1283 CHKB4:  XCHG            ; RESTORE THIS ADDRESS IN H,L
06B7 23           =1284         INX   H         ; POINT TO THE MINIMUM SLEW BOUND
06B8 3A1700     D =1285         LDA   MDSIGN    ; GET THE SPEED CONTROL MODE SETTING
06BB 1F           =1286         RAR             ; CHECK IF THE FAST MODE IS SET FOR SLEWING
06BC D2C306       =1287         JNC   FASTSL
06BF 1F           =1288         RAR             ; CHECK IF THE FAST MODE IS SET FOR ELEVATING
06C0 DAEA06       =1289         JC    NTFAST    ; JUMP IF THE SLOW MODE IS SET FOR BOTH SLEWING AND LIFTING
06C3 C5           =1290 FASTSL: PUSH  B         ; SAVE THE REGISTERS B,C
06C4 7E           =1291         MOV   A,M       ; GET THE MINIMUM SLEW BOUND
06C5 C618         =1292         ADI   018H      ; INCREASE THIS BY 018H
06C7 D2CB06       =1293         JNC   $+4       ; LIMIT THE MAXIMUM TO 0FFH
06CA 9F           =1294         SBB   A         ; FORM 0FFH BY SUBTRACTING A AND BORROW FROM ACC
06CB 4F           =1295         MOV   C,A       ; SAVE THIS CONTENT IN C
06CC 23           =1296         INX   H         ; INCREMENT H,L TO POINT AT THE MAXIMUM SLEW BOUND
06CD 7E           =1297         MOV   A,M       ; GET THIS VALUE
06CE D617         =1298         SUI   017H      ; DECREASE THIS BY 017H (TOLERANCE FOR FAST MODE)
06D0 D2D406       =1299         JNC   $+4       ; LIMIT THE MINIMUM TO 00H
06D3 97           =1300         SUB   A         ; FORM 00H BY SUBTRACTING ACC FROM ACC
06D4 47           =1301         MOV   B,A       ; SAVE THIS VALUE TO B
06D5 2B           =1302         DCX   H         ; DECREMENT H,L TO POINT AT THE MINIMUM SLEW BOUND
06D6 3A2300     D =1303         LDA   SLCRRT    ; GET THE CURRENT SLEW DATA
06D9 B9           =1304         CMP   C         ; COMPARE WITH THE 'TOLERATED' MINIMUM BOUND FOR FAST MODE
06DA DAE106       =1305         JC    FASTM2    ; JUMP TO SET THE SLOW MODE IF WITHIN THE TOLERATED REGION
06DD B8           =1306         CMP   B         ; COMPARE WITH THE 'TOLERATED' MAXIMUM BOUND FOR FAST MODE
06DE DAE906       =1307         JC    FASTM3    ; JUMP IF OUTSIDE THIS 'TOLERATED' REGION
06E1 3A1700     D =1308 FASTM2: LDA   MDSIGN    ; GET THE CURRENT SPEED MODE SETTING
06E4 F603         =1309         ORI   03H       ; SET THE SLOW MODE
06E6 321700     D =1310         STA   MDSIGN    ; SEND THIS SIGNAL TO THE PORT
06E9 C1           =1311 FASTM3: POP   B         ; RECOVER THE REGISTERS B,C
06EA 3A2300     D =1312 NTFAST: LDA   SLCRRT    ; GET THE CURRENT SLEW DATA
06ED BE           =1313         CMP   M
06EE DAFC06       =1314         JC    NOVER1    ; IF CURRENT < MIN. BOUND  THEN ONLY RIGHT SLEW ALLOWED
06F1 C20407       =1315         JNZ   VERT2     ; IF CURRENT > MIN. BOUND  THEN MOVEMENT ALLOWED
06F4 3A0300     D =1316         LDA   SIGNAL    ; IF CURRENT = MIN. BOUND  THEN NO LEFT SLEW
06F7 F690         =1317         ORI   090H      ; ALLOW  RIGHT & VERTICAL MOVEMENT
                  =1318                         ; AFTER ORING THE PATTERN IS: 1XX1 XXXX
06F9 C30107       =1319         JMP   VERT3     ; JUMP TO SET THIS SIGNAL
06FC 3A0300     D =1320 NOVER1: LDA   SIGNAL    ; ALLOW RIGHT MOVE ONLY
06FF F69F         =1321         ORI   09FH      ; AFTER ORING THE PATTERN IS: 1XX1 1111
0701 320300     D =1322 VERT3:  STA   SIGNAL    ; STORE THIS SIGNAL
0704 23           =1323 VERT2:  INX   H
0705 3A2300     D =1324         LDA   SLCRRT
0708 BE           =1325         CMP   M
0709 DA2207       =1326         JC    VERT4     ; JUMP IF CURRENT SLEW < MAX. BOUND
070C CA1707       =1327         JZ    ONVER2    ; JUMP IF CURRENT SLEW = MAX. BOUND
070F 3A0300     D =1328         LDA   SIGNAL    ; CURRENT SLEW > MAX. BOUND
0712 F66F         =1329         ORI   06FH      ; ALLOW LEFT MOVE ONLY
                  =1330                         ; AFTER ORING THE PATTERN IS: X11X 1111
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0          CNTRL    PAGE   30
CONTROL WITH SIMPLE COMPARISION

 LOC  OBJ          LINE        SOURCE STATEMENT

 0714 C31C07       =1331               JMP   VERT5
 0717 3A0300     D =1332 ONVER2:  LDA   SIGNAL    ; CURRENT SLEW = MAX. BOUND
 071A F660         =1333               ORI   060H     ; ALLOW LEFT AND VERTICAL MOVE ONLY
                   =1334                              ; AFTER ORING THE PATTERN IS: X11X XXXX
 071C 320300     D =1335 VERT5:   STA   SIGNAL    ; STORE THIS SIGNAL
 071F C32C07       =1336               JMP   OUTRS    ;
 0722 3A0300     D =1337 VERT4:   LDA   SIGNAL    ; ALL MOVEMENT ALLOWED
 0725 C32C07       =1338               JMP   OUTRS
 0728 F1           =1339 NTSRT1:    POP   PSW
 0729 3A0300     D =1340               LDA   SIGNAL    ; GET THE SIGNAL FOR THE DIRECTIONAL VALVES
 072C D300         =1341 OUTRS:     OUT   00H      ; OUTPUT THE PATTERN TO PORT 0
 072E 3A1700     D =1342               LDA   MDSIGN
 0731 D301         =1343               OUT   01H      ; SEND THE MOVE-MODE OUT TO THE PORT
 0733 C1           =1344               POP   B        ; RECOVER THE REGISTERS BEFORE RETURNING TO THE CALLING PROGRAM
 0734 D1           =1345               POP   D
 0735 E1           =1346               POP   H
 0736 F1           =1347               POP   PSW
 0737 C9           =1348               RET
 0738 3EFF         =1349 STOPMD:    MVI   A,0FFH   ; STOP THE BOOM CUTTER BECAUSE OF HIGH MOTOR CURRENT
 073A 320300     D =1350               STA   SIGNAL
 073D C32C07       =1351               JMP   OUTRS
                   =1352 ;*****************************************************************
 0740 F5           =1353 SMOVE1:    PUSH  PSW      ; A ROUTINE TO MOVE THE CUTTER TO THE DESIRED POSITION SLOWLY
 0741 E5           =1354               PUSH  H        ; UNTIL IT HAS REACHED THAT POSITION (THE DESIRED POSITION HAS
 0742 F3           =1355 SMOVE2:    DI             ; BEEN STORED IN THE TWO LOCATIONS FROM SLDESR)
 0743 CD4000       =1356               CALL  SETLNK   ; SET THE DATA LINK BETWEEN THE TWO SYSTMES
 0746 3AA21B       =1357               LDA   CRRTSL   ; GET AND STORE THE CURRENT CUTTER POSITION
 0749 6F           =1358               MOV   L,A
 074A 3AAD1B       =1359               LDA   CRRTRS
 074D 67           =1360               MOV   H,A
 074E 222300     D =1361               SHLD  SLCRRT
 0751 3ADD1B       =1362               LDA   CUTFOR   ; GET THE CUTTER MOTOR CURRENT VALUE
 0754 321D00     D =1363               STA   FORCE    ; STORE THIS VALUE
 0757 CD5200       =1364               CALL  OFFLNK   ; RELEASE THE DATA LINK
 075A FB           =1365               EI
 075B CDEA05       =1366               CALL  SMOVE
 075E DB00         =1367               IN    00H      ; CHECK IF VALVES ARE CLOSED
 0760 2F           =1368               CMA
 0761 B7           =1369               ORA   A
 0762 C24207       =1370               JNZ   SMOVE2   ; CONTINUE THE MOVE IF NOT
 0765 CD6F07       =1371               CALL  CHKPOS   ; CHECK IF THE DESIRED POSITION HAS BEEN REACHED
 0768 17           =1372               RAL
 0769 D24207       =1373               JNC   SMOVE2   ; LOOP BACK IF NOT
 076C E1           =1374               POP   H        ; OTHERWISE, RECOVER THE REGISTERS AND RETURN
 076D F1           =1375               POP   PSW
 076E C9           =1376               RET
 076F 3A1D00     D =1377 CHKPOS:  LDA   FORCE    ; CHECK IF THE MOVEMENT OF THE BOOM CUTTER IS STOPPED
 0772 FEE5         =1378               CPI   0E5H     ; BECAUSE OF A HIGH CUTTING FORCE
 0774 9F           =1379               SBB   A        ; SET THE ACCUMULATOR TO INDICATE IN THE FOLLOWING WAY:
                   =1380                              ;     00  ---  IF IT IS STOPPED DUE TO HIGH CUTTING FORCE
                   =1381                              ;     FF  ---  IF IT IS STOPPED DUE TO POSITION ACHEIVED
 0775 C9           =1382               RET
                    1383 $EJECT
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   1384 $INCLUDE(:F1:FMOVE1.SRC)
                  =1385 ;*************************************************************
                  =1386 ;*      A ROUTINE TO FAST MOVE THE BOOM TO THE DESIRED POSITION     *
                  =1387 ;*************************************************************
                  =1388 PUBLIC  FMOVE
                  =1389 ;*************************************************************
                  =1390         ASEG            ; DECLARE ABSOLUTE PROGRAM CODE SEGMENT
80AE              =1391         ORG  080AEH     ; FMOVE STARTING ADDRESS: 80AE H
80AE F5           =1392 FMOVE:  PUSH PSW        ; SAVE THE REGISTERS
80AF E5           =1393         PUSH H
80B0 D5           =1394         PUSH D
80B1 C5           =1395         PUSH B
80B2 F3           =1396         DI              ; DISABLE THE INTERRUPT
80B3 211000       =1397         LXI  H,16D      ; SET THE BYTE COUNT FOR DATA TRANSFER
80B6 E5           =1398         PUSH H
80B7 CD4000       =1399         CALL SETLNK     ; SET UP THE DATA LINK BETWEEN THE TWO SYSTEMS
80BA 01E51B       =1400         LXI  B,SLVEL    ; SET THE STARTING ADDRESS FOR SOURCE DATA BLOCK
80BD 112500   D   =1401         LXI  D,VELSL    ; SET THE STARTING ADDRESS FOR DESTINATION DATA BLOCK
80C0 CD9700       =1402         CALL TRANSF     ; TRANSFER THE DATA
80C3 3AA21B       =1403         LDA  CRRTSL     ; GET AND STORE THE CURRENT BOOM POSITION
80C6 6F           =1404         MOV  L,A
80C7 3AAD1B       =1405         LDA  CRRTRS
80CA 67           =1406         MOV  H,A
80CB 222300   D   =1407         SHLD SLCRRT
80CE 3ADD1B       =1408         LDA  CUTFOR     ; GET THE CUTTER MOTOR CURRENT READING
80D1 321D00   D   =1409         STA  FORCE      ; STORE THIS VALUE
80D4 CD5200       =1410         CALL OFFLNK     ; RELEASE THE DATA LINK BETWEEN THE TWO SYSTEMS
80D7 E1           =1411         POP  H          ; POP THE STACK TO KEEP THE BALANCE
80D8 FB           =1412         EI
80D9 DB01         =1413         IN   01H        ; GET THE CURRENT VALVE STATUS
80DB E6FC         =1414         ANI  0FCH       ; SET THE SPEED CONTROL VALVE TO FAST MODE BY ANDING IT WITH 11111100B
80DD F5           =1415         PUSH PSW
80DE 2A1A00   D   =1416         LHLD SLDESR     ; LOAD THE REGISTER L WITH THE DESIRED SLEW VALUE
                  =1417                         ;  AND THE REGISTER H WITH THE DESIRED RAISE VALUE
80E1 EB           =1418         XCHG            ; EXCHANGE REGISTERS D,E WITH REGISTERS H,L
80E2 2A2300   D   =1419         LHLD SLCRRT     ; LOAD THE REGISTER L WITH THE CURRENT SLEW VALUE
                  =1420                         ;  AND THE REGISTER H WITH THE CURRENT RAISE VALUE
80E5 7D           =1421         MOV  A,L        ; LOAD THE ACCUMULATOR WITH THE CURRENT SLEW VALUE
80E6 93           =1422         SUB  E          ; SUBTRACT THE DESIRED SLEW VALUE FROM THIS CURRENT VALUE
80E7 D2EC80       =1423         JNC  $+5
80EA 2F           =1424         CMA
80EB 3C           =1425         INR  A
80EC FE18         =1426         CPI  018H
80EE D2F580       =1427         JNC  $+7
80F1 F1           =1428         POP  PSW
80F2 F601         =1429         ORI  01H        ; SET SLEW TO SLOW MODE
80F4 F5           =1430         PUSH PSW
80F5 7C           =1431         MOV  A,H        ; SUBTRACT THE DESIRED RAISE VALUE FROM THE CURRENT VALUE
80F6 92           =1432         SUB  D
80F7 D2FC80       =1433         JNC  $+5
80FA 2F           =1434         CMA
80FB 3C           =1435         INR  A
80FC FE18         =1436         CPI  018H
80FE D20581       =1437         JNC  $+7
8101 F1           =1438         POP  PSW
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

8102 F602        =1439          ORI  003H      ; SET THE RAISE TO SLOW MODE
8104 F5          =1440          PUSH PSW       ; PUT THIS SIGNAL ONTO THE STACK
8105 F1          =1441          POP  PSW       ; GET BACK THE MOVE-MODE SIGNAL
8106 C3F205      =1442          JMP  SMOVEF    ; JUMP TO CONTINUE THE OPERATION
                  1443 $INCLUDE(:F1:DATA.SRC)
                 =1444 ;*************************************************************
                 =1445 ;*      DATA SEGMENT FOR THE BOOM POSITIONAL CONTROL PROGRAM    *
                 =1446 ;*************************************************************
                 =1447 PUBLIC  CRRTSL,CRRTRS,MOVEST,DESRSL,DESRRS,PFLEND,PFLST,CUTTER
                 =1448 PUBLIC  SLVEL,SLACC,RSVEL,RSACC,STADD,LASTAD,DDR01,TEMP,TEMP1,TEMPR,TEMPR1
                 =1449 PUBLIC  SLCRRT,RSCRRT,SLDESR,RSDESR,VELSL,VELRS,ACCSL,ACCRS,PREVSL,PREVRS
                 =1450 PUBLIC  SIMUX,SIMUY,SIMUAX,SIMUAY,SIMUAZ
                 =1451 ASEG
1BA2             =1452 ORG     01BA2H
                 =1453 ;*************************************************************
                 =1454 ;*      SYMBOLS TO BE USED IN THE DATA CAPTURING SYSTEM         *
                 =1455 ;*************************************************************
                 =1456 ;SYMBOL                    DEFINITION
                 =1457 ;------                    ----------
1BA2             =1458 CRRTSL: DS    1          ; CURRENT SLEW POSITION        (1 BYTE)
1BAD             =1459 ORG     01BADH
1BAD             =1460 CRRTRS: DS    1          ; CURRENT RAISE POSITION       (1 BYTE)
1BDA             =1461 ORG     01BDAH
1BDA             =1462 DESRSL: DS    1          ; DESIRED SLEW POSITION        (1 BYTE)
1BDB             =1463 DESRRS: DS    1          ; DESIRED RAISE POSITION       (1 BYTE)
1BDC             =1464 MOVEST: DS    1          ; CURRENT MOVE LEVER STATUS    (1 BYTE)
1BDD             =1465 CUTFOR: DS    1          ; CUTTER MOTOR CURRENT         (1 BYTE)
1BE3             =1466 ORG     01BE3H
1BE3             =1467 PREVSL: DS    1          ; CURRENT SLEW POSITION        (1 BYTE)
1BE4             =1468 PREVRS: DS    1          ; CURRENT RAISE POSITION       (1 BYTE)
1BE5             =1469 SLVEL:  DS    4          ; CURRENT SLEW VELOCITY        (4 BYTES)
1BE9             =1470 SLACC:  DS    4          ; CURRENT SLEW ACCELERATION    (4 BYTES)
1BED             =1471 RSVEL:  DS    4          ; CURRENT RAISE VELOCITY       (4 BYTES)
1BF1             =1472 RSACC:  DS    4          ; CURRENT RAISE ACCELERATION   (4 BYTES)
1BFE             =1473 ORG     1BFEH
1BFE             =1474 PFLEND: DS    2          ; LAST ADDRESS FOR PROFILE DATA (2 BYTES)
1C00             =1475 PFLST:  DS 400H          ; STARTING ADDRESS FOR PROFILE DATA (MAX. 400H BYTES)
BBFE             =1476 ORG     0BBFEH
                 =1477 ;*************************************************************
                 =1478 ;*     SYMBOLS TO BE USED IN THE MAIN CONTROL SYSTEM PROGRAM    *
                 =1479 ;*************************************************************
                 =1480 ;SYMBOL                    DEFINITION
                 =1481 ;------                    ----------
BBFE             =1482 LASTAD: DS    2          ; LAST ADDRESS FOR THE PROFILE (2 BYTES)
BC00             =1483 STADD:  DS 400H          ; DATA STORAGE FOR THE PROFILE (MAX. 400H BYTES)
                 =1484 DSEG                     ; DEFINE DATA SEGMENT
0000             =1485 STKPTR:  DS    0         ; DEFINE THE HIGHEST STACK BYTE TO BE THE ONE BELOW
                 =1486                          ; THE LOWEST DATA MEMORY BYTE
0000             =1487 INTMSK:  DS    1         ; STORAGE LOCATION FOR THE INTERRUPT MASK
0001             =1488 CUTSTP:  DS    1         ; LOCATION FOR THE CUTTER STEP SIZE IN ADVANCING
0002             =1489 SORTFL:  DS    1         ; FLAG FOR SORTED PROFILE DATA
                 =1490                          ;   00  --- NOT YET SORTED
                 =1491                          ;   FF  --- SORTING COMPLETED
0003             =1492 SIGNAL:  DS    1         ; SIGNAL FOR THE DIRECTIONAL VALVES
0004             =1493 STARCV:  DS    2         ; DEFINE THE STORAGE FOR STARTING CAVITY 'RAISE' LEVEL
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0006             =1494 LASTCV:  DS    2        ; DEFINE THE STORAGE FOR THE LAST 'CAVITY' LEVEL
 0008             =1495 LASTSO:  DS    2        ; DEFINE THE STORAGE FOR THE LAST SORTED CAVITY ADDRESS
 000A             =1496 LEVEL:   DS    2        ; DEFINE THE STORAGE FOR THE ADDRESS FOR 'X'
 000C             =1497 DMIN1:   DS    2        ; DEFINE THE STORAGE FOR TEMPORARY MIN
 000E             =1498 DMAX1:   DS    2        ; DEFINE THE STORAGE FOR TEMPORARY MAX
 0010             =1499 FIRSL:   DS    2        ; TEMPORARY STORAGE FOR THE NEXT TWO DESIRED SLEW VALUES
 0012             =1500 FOURST:  DS    2        ; TEMPORARY STORAGE FOR 4*(CUTTER STEP SIZE)
 0014             =1501 LASTCT:  DS    2        ; STORAGE FOR THE ADDRESS OF THE LAST DATA BLOCK IN CAVITY CUTTING
 0016             =1502 HALFST:  DS    1        ; STORAGE FOR HALF CUTTER STEP SIZE
 0017             =1503 MDSIGN:  DS    1        ; STORAGE FOR THE BOOM MOVING SPEED MODE
 0018             =1504 DDR01:   DS    1        ; DATA DIRECTION REGISTER FOR PORT 1
 0019             =1505 CUTTER:  DS    1        ; CUTTER STATUS --- 00=STOPPED, 0FFH=STARTED
 001A             =1506 SLDESR:  DS    1        ; DESIRED SLEW POSITION        (1 BYTE)
 001B             =1507 RSDESR:  DS    1        ; DESIRED RAISE POSITION       (1 BYTE)
 001C             =1508 STMOVE:  DS    1        ; 'MOVE' CONTROL STATUS        (1 BYTE)
 001D             =1509 FORCE:   DS    1        ; CUTTER MOTOR CURRENT         (1 BYTE)
 001E             =1510 SIMUX:   DS    1        ; SIMULATED X DISPLACEMENT     (1 BYTE)
 001F             =1511 SIMUY:   DS    1        ; SIMULATED Y DISPLACEMENT     (1 BYTE)
 0020             =1512 SIMUAX:  DS    1        ; SIMULATED ALPHA X            (1 BYTE)
 0021             =1513 SIMUAY:  DS    1        ; SIMULATED ALPHA Y            (1 BYTE)
 0022             =1514 SIMUAZ:  DS    1        ; SIMULATED ALPHA Z            (1 BYTE)
 0023             =1515 SLCRRT:  DS    1        ; CURRENT SLEW POSITION        (1 BYTE)
 0024             =1516 RSCRRT:  DS    1        ; CURRENT RAISE POSITION       (1 BYTE)
 0025             =1517 VELSL:   DS    4        ; CURRENT SLEW VELOCITY        (4 BYTES)
 0029             =1518 ACCSL:   DS    4        ; CURRENT SLEW ACCELERATION    (4 BYTES)
 002D             =1519 VELRS:   DS    4        ; CURRENT RAISE VELOCITY       (4 BYTES)
 0031             =1520 ACCRS:   DS    4        ; CURRENT RAISE ACCELERATION   (4 BYTES)
 0035             =1521 TEMP:    DS    4        ; TEMPORARY STORAGE LOCATION   (4 BYTES)
 0039             =1522 TEMP1:   DS    4
 003D             =1523 TEMPR:   DS    4        ; TEMPORATY STORAGE FOR RESULT (4 BYTES)
 0041             =1524 TEMPR1:  DS    4
 0045             =1525 DIFF1:   DS    4        ; DIFFERENCE BETWEEN CURRENT AND DESIRED (4 BYTES)
 0049             =1526 SWFN:    DS    4        ; SWITCHING FUNCTION VALUE     (4 BYTES)
 004D             =1527 DUMMY1:  DS    4        ;
 0051             =1528 FNSIGN:  DS    1        ; EXPECTED SIGN FOR SWITCHING  (1 BYTE)
                  =1529 ;*************************************************************
                  =1530 ;*                                                           *
                  =1531 ;*************************************************************
                  =1532 $EJECT
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

                   =1533 END


PUBLIC SYMBOLS
ACCRS  D 0031    ACCSL  D 0029    CAVITY A 0218    CRRTRS A 1BAD    CRRTSL A 1BA2    CUTSTP D 0001    CUTTER D 0019
DDR01  D 0018    DESRRS A 1BDB    DESRSL A 1BDA    DMIN1  D 000C    EXPAND A 0509    FIRSL  D 0010    FMOVE  A 80AE
FOURST D 0012    INTMSK D 0000    LASTAD A BBFE    LASTCV D 0006    LASTSO D 0008    LEVEL  D 000A    MOVE   A 02DC
MOVEST A 1BDC    OFFLNK A 0052    PFLEND A 1BFE    PFLST  A 1C00    PREVRS A 1BE4    PREVSL A 1BE3    PROFIL A 00B5
RSACC  A 1BF1    RSCRRT D 0024    RSDESR D 001B    RST45  A 032B    RSVEL  A 1BED    SETLNK A 0040    SHORT  A 0578
SIGNAL D 0003    SIMUAX D 0020    SIMUAY D 0021    SIMUAZ D 0022    SIMUX  D 001E    SIMUY  D 001F    SLACC  A 1BE9
SLCRRT D 0023    SLDESR D 001A    SLVEL  A 1BE5    SMOVE  A 05EA    SMOVE1 A 0740    SMOVE2 A 0742    SMOVEF A 05F2
SORTFL D 0002    STADD  A BC00    STARCV D 0004    STKPTR D 0000    TEMP   D 0035    TEMP1  D 0039    TEMPR  D 003D
TEMPR1 D 0041    TRANSF A 0097    TRNPFL A 0075    VELRS  D 002D    VELSL  D 0025

EXTERNAL SYMBOLS


USER SYMBOLS
ACCRS  D 0031    ACCSL  D 0029    ANYMOD A 060D    CALSMV A 0266    CAVITY A 0218    CHKB1  A 06A5    CHKB2  A 06A4
CHKB3  A 06B5    CHKB4  A 06B6    CHKBND A 0681    CHKPOS A 076F    CLOSL  A 0636    CLOSR  A 067E    COLDST A 0008
COMPL  A 03C0    CONT11 A 041A    CONT12 A 0486    CONT21 A 0411    CONT22 A 0458    CONT31 A 0422    CONT32 A 045E
CONT41 A 03F0    CONT42 A 0449    CONT5  A 05AC    CONT52 A 049D    CONT6  A 05B0    CONT62 A 047B    CONTC0 A 0299
CONTC2 A 02AE    CONTC5 A 027F    CONTP1 A 01FB    CONTR1 A 0380    CONTR2 A 038C    CRRTRS A 1BAD    CRRTSL A 1BA2
CSTORE A 04B1    CTPRO1 A 0109    CTPRO2 A 0157    CUTFOR A 1BDD    CUTSTP D 0001    CUTTER D 0019    DDR01  D 0018
DESRRS A 1BDB    DESRSL A 1BDA    DIFF1  D 0045    DMAX1  D 000E    DMIN1  D 000C    DOWN1  A 066A    DUMMY1 D 004D
DUPLI  A 04C7    DUPLI1 A 04CE    DUPLI2 A 04E0    EXPA1  A 0525    EXPA2  A 053D    EXPA3  A 0541    EXPA4  A 054E
EXPAF  A 056E    EXPAND A 0509    FAST1  A 01BF    FASTM2 A 06E1    FASTM3 A 06E9    FASTS  A 00E6    FASTS1 A 00F7
FASTSL A 06C3    FIRSL  D 0010    FMOVE  A 80AE    FMOVES A 05EE    FNSIGN D 0051    FORCE  D 001D    FOURST D 0012
HALFST D 0016    INTMSK D 0000    LASTAD A BBFE    LASTCT D 0014    LASTCV D 0006    LASTSO D 0008    LEFT1  A 062C
LEVEL  D 000A    LOOP   A 01F0    LOOP1  A 007F    LOOP11 A 03EB    LOOP12 A 0441    LOOP3  A 00A5    MDSIGN D 0017
MOVE   A 02DC    MOVE1  A 0316    MOVE2  A 02E7    MOVEST A 1BDC    NODUP  A 04E6    NOMOVE A 031C    NORED  A 05E2
NOTLOW A 0170    NOTSRT A 01B0    NOTUP  A 0119    NOVER1 A 06FC    NTFAST A 06EA    NTSORT A 0651    NTSRT1 A 0728
OFFLNK A 0052    ONVER2 A 0717    OUTRS  A 072C    OVER1  A 0133    PFLEND A 1BFE    PFLST  A 1C00    PREVRS A 1BE4
PREVSL A 1BE3    PROFIL A 00B5    REDU1  A 05DD    REDU2  A 05E1    REDUCE A 05B6    REDULS A 024D    REDUST A 023D
RIGHT1 A 0631    RSACC  A 1BF1    RSCMP  A 0638    RSCRRT D 0024    RSDESR D 001B    RST38  A 0059    RST45  A 032B
RST55  A 002C    RST65  A 0034    RST75  A 003C    RSVEL  A 1BED    SETLNK A 0040    SHORT  A 0578    SIGNAL D 0003
SIMUAX D 0020    SIMUAY D 0021    SIMUAZ D 0022    SIMUX  D 001E    SIMUY  D 001F    SLACC  A 1BE9    SLCRRT D 0023
SLDESR D 001A    SLVEL  A 1BE5    SMOVE  A 05EA    SMOVE1 A 0740    SMOVE2 A 0742    SMOVEF A 05F2    SORT   A 03DD
SORTFL D 0002    STADD  A BC00    STARCV D 0004    START  A 0000    STKPTR D 0000    STMOVE D 001C    STOFP1 A 0209
STOP1  A 0434    STOP2  A 046F    STOPMD A 0738    SWFN   D 0049    TEMP   D 0035    TEMP1  D 0039    TEMPR  D 003D
TEMPR1 D 0041    TRANSF A 0097    TRAP   A 0024    TRNPFL A 0075    UNDER1 A 0182    UP1    A 0674    VELRS  D 002D
VELSL  D 0025    VERT2  A 0704    VERT3  A 0701    VERT4  A 0722    VERT5  A 071C

ASSEMBLY COMPLETE,  NO ERRORS
```

APPENDIX H

LISTING OF THE ASSEMBLY LANGUAGE PROGRAMS FOR THE 'FMOVE' ROUTINE WITH SWITCHING FUNCTION EVALUATION

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   1385 $INCLUDE(:F1:FMOVE.SRC)
                  =1386 ;*****************************************************************
                  =1387 ;*                                                               *
                  =1388 ;*****************************************************************
                  =1389            ASEG
8000              =1390            ORG     08000H
                  =1391 $INCLUDE(:F1:CONAPU.SRC)
                 1=1392 ;*****************************************************************
                 1=1393 ;*      ROUTINES TO SEND/READ BYTES TO/FROM APU STACK             *
                 1=1394 ;*           I.E.  MEMORY (----) APU STACK                       *
                 1=1395 ;*      ON ENTRY:  REGISTERS B,C CONTAIN STARTING MEMORY ADDRESS  *
                 1=1396 ;*                 WITH LEAST SIGNIFICANT BYTE FIRST             *
                 1=1397 ;*      THE APU 8231 IS MEMORY MAPPED WITH ADDRESSES FC00H - FDFFH *
                 1=1398 ;*                 DATA WITH ADDRESSES: 0FCXX H                  *
                 1=1399 ;*              COMMAND WITH ADDRESSES: 0FDXX H                   *
                 1=1400 ;*                                                               *
                 1=1401 ;*                 S/R NAME   FUNCTION                           *
                 1=1402 ;*                 --------   --------                           *
                 1=1403 ;*                 SEND2      SEND  2  BYTES                     *
                 1=1404 ;*                 SEND4      SEND  4  BYTES                     *
                 1=1405 ;*                 READ2      READ  2  BYTES                     *
                 1=1406 ;*                 READ4      READ  4  BYTES                     *
                 1=1407 ;*                 SEND30     SEND  4  BYTES (3 ZERO MSBs)       *
                 1=1408 ;*                 SEND10     SEND  2  BYTES (1 ZERO MSB)        *
                 1=1409 ;*                                                               *
                 1=1410 ;*****************************************************************
                 1=1411 ASEG
                 1=1412 PUBLIC  SEND2,SEND4,READ2,READ4,SEND30,SEND10,CHKBSY
8000 E5          1=1413 SEND4:  PUSH H        ; SAVE H,L-REGISTERS
8001 2104FC      1=1414         LXI  H,100H*APUD+04H  ; LOAD APU DATA STACK ADDRESS AND COUNT
                 1=1415                               ; VALUE FOR BYTE TRANSFER
8004 C5          1=1416 SENDL1: PUSH B        ; SAVE B,C-REGISTERS
8005 F5          1=1417         PUSH PSW      ; SAVE PROCESSOR STATUS WORD
8006 0B          1=1418         DCX  B        ; GET MEMORY ADDRESS PREPARED
8007 03          1=1419 SENDL2: INX  B
8008 0A          1=1420         LDAX B        ; GET BYTE FROM MEMORY AND
8009 77          1=1421         MOV  M,A      ; SEND BYTE TO APU DATA STACK
800A 2D          1=1422         DCR  L
800B C20780      1=1423         JNZ  SENDL2
800E F1          1=1424         POP  PSW      ; RECOVER REGISTERS BEFORE RETURN
800F C1          1=1425         POP  B
8010 E1          1=1426         POP  H
8011 C9          1=1427         RET           ; RETURN TO CALLING PROGRAMME
8012 E5          1=1428 SEND2:  PUSH H        ; SAME AS SEND4 BUT WITH A BYTE COUNT OF 2
8013 2102FC      1=1429         LXI  H,100H*APUD+02H  ; LOAD APU DATA STACK ADDRESS AND COUNT
                 1=1430                               ; VALUE FOR BYTE TRANSFER
8016 C30480      1=1431         JMP  SENDL1
                 1=1432 ;*****************************************************************
                 1=1433 ;*      ROUTINE TO READ AND STORE 4 BYTES FROM THE APU DATA STACK  *
                 1=1434 ;*****************************************************************
8019 E5          1=1435 READ4:  PUSH H
801A 210404      1=1436         LXI  H,0404H  ; LOAD COUNT VALUE FOR BYTE TRANSFER
801D F5          1=1437 READL3: PUSH PSW      ; SAVE PROCESSOR STATUS WORD
801E 03          1=1438 READL4: INX  B        ; LOCATE MEMORY ADDRESS TO THE MOST SIGNIFICANT
801F 25          1=1439         DCR  H        ; BYTE AND READY FOR DATA TRANSFER
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

8020 C21E80     1=1440          JNZ  READL4
8023 DBFC       1=1441 READL5:  IN   APUD      ; GET BYTE FROM APU DATA STACK (AS AN INPUT PORT)
8025 0B         1=1442          DCX  B         ; STORE DATA INTO MEMORY USING THE B,C-POINTER
8026 02         1=1443          STAX B
8027 2D         1=1444          DCR  L         ; DECREMENT BYTE COUNTER
8028 C22380     1=1445          JNZ  READL5    ; LOOP BACK IF NOT YET ZERO
802B F1         1=1446          POP  PSW       ; RECOVER REGISTERS BEFORE RETURNING TO CALLING
802C E1         1=1447          POP  H         ; PROGRAMME
802D C9         1=1448          RET            ; RETURN TO CALLING SEGMENT
802E E5         1=1449 READ2:   PUSH H         ; SAME AS READ4 BUT WITH 2 BYTES TO BE TRANSFERRED
802F 210202     1=1450          LXI  H,0202H
8032 C31D80     1=1451          JMP  READL3
                1=1452 ;*******************************************************************
                1=1453 ;*      ROUTINE TO SEND DATA BYTES WITH ZERO LSBs                  *
                1=1454 ;*******************************************************************
8035 F5         1=1455 SEND30:  PUSH PSW       ; A ROUTINE TO SEND 4 BYTES TO THE APU DATA STACK WITH 3 ZERO MSBs
8036 E5         1=1456          PUSH H         ; ADDRESS OF THE MSB SPECIFIED IN B,C
8037 26FC       1=1457          MVI  H,APUD    ; LOAD REGISTER H WITH THE APU DATA STACK ADDRESS
8039 0A         1=1458          LDAX B         ; GET THE LSB AND
803A 77         1=1459          MOV  M,A       ; SEND IT TO THE APU DATA STACK
803B 97         1=1460          SUB  A         ; CLEAR THE ACCUMULATOR AND
803C 77         1=1461          MOV  M,A       ; SEND THE THREE ZERO MSBs
803D 77         1=1462          MOV  M,A
803E 77         1=1463 SEND31:  MOV  M,A
803F E1         1=1464          POP  H         ; RECOVER THE REGISTERS BEFORE
8040 F1         1=1465          POP  PSW
8041 C9         1=1466          RET            ; RETURN TO THE CALLING PROGRAM
8042 F5         1=1467 SEND10:  PUSH PSW       ; A ROUTINE TO SEND 2 BYTES TO THE APU DATA STACK WITH 1 ZERO MSB
8043 E5         1=1468          PUSH H         ; THE ADDRESS OF THE MSB IS SPECIFIED IN REGISTERS B,C
8044 26FC       1=1469          MVI  H,APUD
8046 0A         1=1470          LDAX B         ; GET THE LSB
8047 77         1=1471          MOV  M,A       ; SEND IT TO THE APU DATA STACK
8048 97         1=1472          SUB  A
8049 C33E80     1=1473          JMP  SEND31    ; JUMP TO SEND THE DATA TO THE APU STACK
                1=1474 ;*******************************************************************
                1=1475 ;*     A ROUTINE TO CONVERT THE FLOATING POINT FORMAT RECOGNIZED    *
                1=1476 ;*     BY THE FLOATING POINT LIBRARY (FPAL) TO THAT RECOGNIZED     *
                1=1477 ;*     BY THE ARITHMETIC PROCESSING UNIT 8231A                     *
                1=1478 ;*     ON ENTRY: REGISTERS B,C --- ADDRESS OF THE LOWEST BYTE OF   *
                1=1479 ;*                                 THE 4-BYTE BFP NUMBER           *
                1=1480 ;*     ON EXIT:  REGISTERS B,C --- ADDRESS OF THE LOWEST BYTE OF   *
                1=1481 ;*                                 THE 4-BYTE BFP NUMBER           *
                1=1482 ;*******************************************************************
                1=1483 PUBLIC  CONVER      ; DEFINE NAME OF THIS ROUTINE AND OPEN TO OTHER SEGMENTS
                1=1484 ;*******************************************************************
804C F5         1=1485 CONVER:  PUSH PSW   ; SAVE THE REGISTERS ON ENTERING THIS ROUTINE
804D D5         1=1486          PUSH D
804E C5         1=1487          PUSH B
804F 03         1=1488          INX  B     ; GET THE LOWEST BIT OF THE EXPONENT FIELD AND
8050 03         1=1489          INX  B
8051 0A         1=1490          LDAX B
8052 17         1=1491          RAL        ; PUT IT INTO THE CARRY FLAG
8053 03         1=1492          INX  B     ; GET THE EXPONENT FIELD
8054 0A         1=1493          LDAX B
8055 17         1=1494          RAL
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

  8056 F600       1=1495          ORI   00H     ; CHECK IF THE EXPONENT IS ZERO
  8058 CA9480     1=1496          JZ    RET10   ; IF YES THEN NO CONVERSION IS NEEDED
  805B 1E7E       1=1497          MVI   E,07EH  ; TAKE AWAY THE OFFSET OF THE EXPONENT FIELD
  805D 93         1=1498          SUB   E       ; EXPONENT IS OFFSET BY 07EH
  805E 57         1=1499          MOV   D,A     ; TEMPORARILY STORE RESULT IN D
  805F DA7880     1=1500          JC    NEGAT   ; EXPONENT IS NEGATIVE
  8062 FE40       1=1501 POSIT:   CPI   040H    ; CHECK IF OVERFLOW THE EXPONENT BYTE
  8064 DA8880     1=1502          JC    CONT10  ; NO ADJUSTMENT IF WITHIN THE RANGE
  8067 163F       1=1503          MVI   D,03FH  ; ASSUME THE LARGEST NUMBER IF OVERFLOW
  8069 3EFF       1=1504          MVI   A,0FFH  ;      THE LARGEST NUMBER IS: 3F FF FF FF H
  806B 0B         1=1505          DCX   B
  806C 02         1=1506          STAX  B
  806D 0B         1=1507          DCX   B
  806E 02         1=1508          STAX  B
  806F 0B         1=1509          DCX   B
  8070 02         1=1510          STAX  B
  8071 03         1=1511          INX   B
  8072 03         1=1512          INX   B
  8073 03         1=1513          INX   B
  8074 1C         1=1514          INR   E
  8075 C38B80     1=1515          JMP   CONT20  ; JUMP TO CONTINUE
  8078 FEC0       1=1516 NEGAT:   CPI   0C0H    ; CHECK IF UNDERFLOW
  807A D28880     1=1517          JNC   CONT10  ; NO ADJUSTMENT IF WITHIN THE RANGE
  807D 97         1=1518          SUB   A       ; ASSUME ZERO IF UNDERFLOW
  807E 02         1=1519          STAX  B       ;    ZERO IS REPRESENTED BY: 00 00 00 00 H
  807F 0B         1=1520          DCX   B
  8080 02         1=1521          STAX  B
  8081 0B         1=1522          DCX   B
  8082 02         1=1523          STAX  B
  8083 0B         1=1524          DCX   B
  8084 02         1=1525          STAX  B
  8085 C39480     1=1526          JMP   RET10
  8088 1C         1=1527 CONT10:  INR   E       ; MASK OUT THE M.S.B.
  8089 A3         1=1528          ANA   E
  808A 57         1=1529          MOV   D,A     ; AND TEMPORARILY STORE THIS IN D
  808B 0A         1=1530 CONT20:  LDAX  B       ; EXTRACT THE SIGN BIT FROM THE LOWEST BYTE
  808C 1C         1=1531          INR   E       ; MASK OUT THE 7 L.S.B.
  808D A3         1=1532          ANA   E
  808E B2         1=1533          ORA   D       ; COUPLE THIS TO THE EXPONENT BYTE
  808F 02         1=1534          STAX  B       ; STORE THIS RESULTING SIGN-EXPONENT BYTE
  8090 0B         1=1535          DCX   B       ; GET THE FIRST FRACTION BYTE (SECOND BYTE OF THE 4)
  8091 0A         1=1536          LDAX  B
  8092 B3         1=1537          ORA   E       ; PUT A 'ONE' IN THE M.S.B.
  8093 02         1=1538          STAX  B       ; PUT THE RESULTING BYTE BACK TO THE MEMORY
  8094 C1         1=1539 RET10:   POP   B       ; RECOVER REGISTERS BEFORE RETURN
  8095 D1         1=1540          POP   D
  8096 F1         1=1541          POP   PSW
  8097 C9         1=1542          RET
                  1=1543 ;*************************************************************************
                  1=1544 ;*       RESET THE APU                                                   *
                  1=1545 ;*************************************************************************
                  1=1546 PUBLIC   RSTAPU
  8098 F5         1=1547 RSTAPU:  PUSH PSW
  8099 DB01       1=1548          IN    01H     ; GET THE PORT 01 SETTING
  809B F610       1=1549          ORI   010H    ; SET THE HIGH ACTIVE BIT 4 TO RESET THE APU
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

809D D301        1=1550          OUT   01H     ; SEND THIS RESET SIGNAL TO THE APU RESET PIN
809F 00          1=1551          NOP           ; KEEP THE RESET SIGNAL ACTIVE FOR THE TIME REQUIRED
80A0 00          1=1552          NOP
80A1 E6EF        1=1553          ANI   0EFH    ; REMOVE THE RESET SIGNAL FOR THE APU
80A3 D301        1=1554          OUT   01H
80A5 F1          1=1555          POP   PSW
80A6 C9          1=1556          RET
                 1=1557                        ;
80A7 DBFD        1=1558 CHKBSY:  IN    APUC    ; GET THE APU STATUS WORD
80A9 17          1=1559          RAL           ; CHECK IF THE APU IS BUSY
80AA DAA780      1=1560          JC    $-3     ; KEEP LOOPING IF IT IS BUSY
80AD C9          1=1561          RET           ; OTHERWISE RETURN TO THE CALLING SEGMENT
                 1=1562                        ;
                  =1563 $INCLUDE(:F1:APUCMD.SRC)
                 1=1564 $NOLIST
                 1=1615 $LIST
                  =1616 ;*****************************************************************
                  =1617 ;*        A ROUTINE FOR FAST MOVE OF BOOM CUTTER                 *
                  =1618 ;*        WITH THE SWITCHING FUNCTION EVALUATION                 *
                  =1619 ;*        VERTICAL MOVEMENTS HAVE PREFERENCE OVER                *
                  =1620 ;*        HORIZONTAL MOVEMENTS                                   *
                  =1621 ;*****************************************************************
                  =1622 PUBLIC    FMOVE
                  =1623           ASEG
80AE F5           =1624 FMOVE:    PUSH    PSW           ; SAVE THE REGISTERS ON ENTRY
80AF E5           =1625           PUSH    H
80B0 D5           =1626           PUSH    D
80B1 C5           =1627           PUSH    B
80B2 F3           =1628           DI                    ; DISABLE SYSTEM INTERRUPU
80B3 211000       =1629           LXI     H,16D         ; SET BYTE TRANSFER COUNTER
80B6 E5           =1630           PUSH    H
80B7 CD4000       =1631           CALL    SETLNK        ; SET THE LINK BETWEEN THE TWO SYSTEMS
80BA 01E51B       =1632           LXI     B,SLVEL       ; SET THE STARTING ADDRESS FOR SOURCE DATA
80BD 112500     D =1633           LXI     D,VELSL       ; SET THE STARTING ADDRESS FOR DESTINATION
80C0 CD9700       =1634           CALL    TRANSF        ; TRANSFER BYTE
80C3 3ADD1B       =1635           LDA     CUTFOR        ; GET THE CUTTER FORCE
80C6 321D00     D =1636           STA     FORCE         ; AND STORE IT
80C9 3AA21B       =1637           LDA     CRRTSL        ; GET THE CURRENT BOOM POSITION
80CC 6F           =1638           MOV     L,A
80CD 3AAD1B       =1639           LDA     CRRTRS
80D0 67           =1640           MOV     H,A
80D1 222300     D =1641           SHLD    SLCRRT        ; AND STORE THIS INFORMATION
80D4 CD5200       =1642           CALL    OFFLNK        ; RELEASE THE LINK
80D7 E1           =1643           POP     H             ; SCRATCH THE BYTE COUNTER FOR TRANSFER
80D8 FB           =1644           EI                    ; ENABLE INTERRUPTS
80D9 3A1D00     D =1645           LDA     FORCE         ; CHECK FOR THE FORCE SETTING
80DC FE18         =1646           CPI     018H          ;       LOWER BOUND = 018H
80DE D2EE05       =1647           JNC     FMOVES        ; JUMP TO SLOW MOVE ROUTINE IF LARGER
80E1 2A2300     D =1648           LHLD    SLCRRT        ; OTHERWISE, CHECK IF FAST MOVEMENT IS ALLOWED
80E4 EB           =1649           XCHG                  ; DUE TO THE POSSIBILITY OF OVERSHOOT
80E5 2A1A00     D =1650           LHLD    SLDESR
80E8 DB01         =1651           IN      01H
80EA E6FC         =1652           ANI     0FCH
80EC 321700     D =1653           STA     MDSIGN
80EF 7D           =1654           MOV     A,L           ; FIRSTLY, FOR SLEW
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

80F0 93           =1655         SUB     E
80F1 D2F680       =1656         JNC     $+5
80F4 2F           =1657         CMA
80F5 3C           =1658         INR     A
80F6 FE04         =1659         CPI     04H             ; TOLERANCE IS SET TO 4 BITS
80F8 D20381       =1660         JNC     FCONT1          ; CONTINUE IF DIFFERENCE IS LARGER
80FB 3A1700   D   =1661         LDA     MDSIGN
80FE F601         =1662         ORI     01H             ; SET SLOW SLEW
8100 321700   D   =1663         STA     MDSIGN
8103 7C           =1664 FCONT1: MOV     A,H             ; THEN CHECK FOR RAISE
8104 92           =1665         SUB     D
8105 D20A81       =1666         JNC     $+5
8108 2F           =1667         CMA
8109 3C           =1668         INR     A
810A FE04         =1669         CPI     04H
810C D21781       =1670         JNC     FCONT
810F 3A1700   D   =1671         LDA     MDSIGN
8112 F602         =1672         ORI     02H             ; LIMIT LIFTING TO SLOW MODE
8114 321700   D   =1673         STA     MDSIGN
8117 3A1700   D   =1674 FCONT:  LDA     MDSIGN          ; CHECK IF THE SLOW MODE HAS BEEN SET
811A F6FC         =1675         ORI     0FCH
811C FEFF         =1676         CPI     0FFH            ; JUMP TO SLOW MODE IF YES
811E CAEE05       =1677         JZ      FMOVES
8121 3A0200   D   =1678         LDA     SORTFL          ; CHECK IF THE PROFILE DATA HAS BEEN SORTED AND EXPANDED
8124 17           =1679         RAL
8125 D2B683       =1680         JNC     NOSORT          ; JUMP IF THE PROFILE DATA HAS NOT BEEN SORTED
8128 2A0600   D   =1681         LHLD    LASTCV          ; OTHERWISE, LIMIT THE DESIRED POSITION TO THE BOUNDS
812B EB           =1682         XCHG
812C 2A0400   D   =1683         LHLD    STARCV
812F 3A1B00   D   =1684         LDA     RSDESR          ; FIRST FOR THE RAISE DATA
8132 BE           =1685         CMP     M               ;       FOR LOWEST LIMIT
8133 D23781       =1686         JNC     $+4
8136 7E           =1687         MOV     A,M
8137 EB           =1688         XCHG
8138 BE           =1689         CMP     M               ;       FOR UPPER MOST LIMIT
8139 DA3D81       =1690         JC      $+4
813C 7E           =1691         MOV     A,M
813D 321B00   D   =1692         STA     RSDESR          ; STORE THIS ADJUSTED SETTING
8140 6F           =1693         MOV     L,A             ; NOW CHECK FOR THE SLEW BOUNDS
8141 2600         =1694         MVI     H,000H
8143 29           =1695         DAD     H
8144 29           =1696         DAD     H
8145 1100BC       =1697         LXI     D,STADD
8148 19           =1698         DAD     D
8149 23           =1699         INX     H
814A 3A1A00   D   =1700         LDA     SLDESR          ; POINT TO THE LEFTMOST SLEW BOUND
814D BE           =1701         CMP     M               ; GET THE DESIRED SLEW POSITION
814E D25281       =1702         JNC     $+4
8151 7E           =1703         MOV     A,M             ; SKIP IF GREATER
8152 23           =1704         INX     H
8153 BE           =1705         CMP     M               ; POINT TO THE RIGHTMOST SLEW BOUND
8154 DA5881       =1706         JC      $+4
8157 7E           =1707         MOV     A,M             ; SKIP IF SMALLER
8158 321A00   D   =1708         STA     SLDESR
815B 3A2400   D   =1709         LDA     RSCRRT          ; STORE THIS ADJUSTED SETTING
                                                        ; CHECK IF CURRENT BOOM POSITION IS IN THE NEIGHBOURHOOD
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

815E D610          =1710          SUI     010H          ; OF THE PROFILE BOUNDARY
8160 DA6A81        =1711          JC      $+10D
8163 2A0400    D   =1712          LHLD    STARCV        ; CHECK IF IT IS CLOSE TO THE LOWEST BOUND
8166 BE            =1713          CMP     M
8167 D27481        =1714          JNC     $+13D
816A F5            =1715          PUSH    PSW           ; IF YES, SET SLOW LIFT MODE
816B 3A1700    D   =1716          LDA     MDSIGN
816E F602          =1717          ORI     02H
8170 321700    D   =1718          STA     MDSIGN
8173 F1            =1719          POP     PSW
8174 C61F          =1720          ADI     01FH          ; OTHERWISE, CHECK FOR THE UPPER MOST BOUND
8176 DA8081        =1721          JC      $+10D
8179 2A0600    D   =1722          LHLD    LASTCV
817C BE            =1723          CMP     M
817D DA8881        =1724          JC      $+11D
8180 3A1700    D   =1725          LDA     MDSIGN        ; IF HIGHER, THEN SET SLOW LIFT MODE
8183 F602          =1726          ORI     02H
8185 321700    D   =1727          STA     MDSIGN
8188 3A2400    D   =1728          LDA     RSCRRT        ; RAISE IS AWAY FROM THE ALTITUDE LIMITS
818B 6F            =1729          MOV     L,A           ; CHECK FOR THE SLEW BOUNDS
818C 2600          =1730          MVI     H,00H
818E 29            =1731          DAD     H
818F 29            =1732          DAD     H
8190 1100BC        =1733          LXI     D,STADD
8193 19            =1734          DAD     D
8194 23            =1735          INX     H             ; POINT H,L TO THE LEFTMOST SLEW BOUND
8195 3A2300    D   =1736          LDA     SLCRRT        ; GET THE CURRENT SLEW DATA
8198 D610          =1737          SUI     010H
819A BE            =1738          CMP     M             ; COMPARE WITH THE LEFTMOST SLEW BOUND
819B DAEE05        =1739          JC      FMOVES        ; JUMP TO SLOW MOVE IF WITHIN
819E C61F          =1740          ADI     01FH          ; CHECK THE RIGHMOST BOUND
81A0 DAEE05        =1741          JC      FMOVES        ; JUMP TO SLOW MOVE IF WITHIN
81A3 23            =1742          INX     H
81A4 BE            =1743          CMP     M
81A5 D2EE05        =1744          JNC     FMOVES
81A8 3A1700    D   =1745          LDA     MDSIGN        ; NOW THE CURRENT BOOM POSITION IS WITHIN THE 'FREE' AREA
                   =1746                                ; GET THE CURRENT SPEED MODE SETTING
81AB D301          =1747 SPEED:    OUT     01H           ; SET THE SPEED MODE TO HIGH
81AD 211000        =1748 S6LOOP:   LXI     H,16D         ; LOAD THE COUNTER FOR BYTE TRANSFER
81B0 E5            =1749          PUSH    H
81B1 CD4000        =1750          CALL    SETLNK        ; SET UP THE INTERFACE LINK
81B4 01E51B        =1751          LXI     B,SLVEL       ; SET THE STARTING SOURCE ADDRESS
81B7 112500    D   =1752          LXI     D,VELSL       ; SET THE STARTING DESTINATION ADDRESS
81BA CD9700        =1753          CALL    TRANSF        ; CALL THE BYTE TRANSFER ROUTINE
81BD 3AA21B        =1754          LDA     CRRTSL
81C0 322300    D   =1755          STA     SLCRRT
81C3 3AAD1B        =1756          LDA     CRRTRS
81C6 322400    D   =1757          STA     RSCRRT
81C9 CD5200        =1758          CALL    OFFLNK        ; RELEASE THE LINK
81CC E1            =1759          POP     H
81CD CD9880        =1760          CALL    RSTAPU        ; RESET THE APU
81D0 011B00    D   =1761          LXI     B,RSDESR      ; GET THE DESIRED RAISE POSITION
81D3 CD4280        =1762          CALL    SEND10        ; SEND THIS TO THE APU DATA STACK
81D6 26FD          =1763          MVI     H,APUC        ; LOAD THE AUP COMMAND PORT ADDRESS
81D8 361D          =1764          MVI     M,FLTS        ; FLOAT THIS INTEGER
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

81DA 3A1B00    D =1765          LDA     RSDESR          ;  B <--- RSDESR
81DD 47          =1766          MOV     B,A
81DE 3A2400    D =1767          LDA     RSCRRT          ;  A <--- RSCRRT
81E1 90          =1768          SUB     B               ;  RSCRRT - RSDESR
81E2 9F          =1769          SBB     A               ;        A = 000H  IF RSCRRT >= RSDESR
                 =1770                                  ;        A = 0FFH  IF RSCRRT <  RSDESR
81E3 325100    D =1771          STA     FNSIGN          ; STORE THIS FUNCTION SIGN
81E6 014D00    D =1772          LXI     B,DUMMY1        ; LOAD ADDRESS FOR THE RESULT
81E9 CDA780      =1773          CALL    CHKBSY
81EC CD1980      =1774          CALL    READ4
81EF 012400    D =1775          LXI     B,RSCRRT        ; FORM THE DIFFERENCE BETWEEN THE CURRENT AND 040H
81F2 CD4280      =1776          CALL    SEND10
81F5 3611        =1777          MVI     M,FLTS
81F7 014D00    D =1778          LXI     B,DUMMY1
81FA CDA780      =1779          CALL    CHKBSY
81FD CD0080      =1780          CALL    SEND4
8200 3611        =1781          MVI     M,FSUB          ; PERFORM (CURRENT SLEW - DESIRED SLEW)
8202 26FD        =1782          MVI     H,APUC          ; LOAD THE APU COMMAND PORT ADDRESS
8204 012D00    D =1783          LXI     B,VELRS         ; CONVERT THE VELOCITY INTO APU BFP FORMAT
8207 CD4C80      =1784          CALL    CONVER
820A 014500    D =1785          LXI     B,DIFF1         ; STORE (CURRENT RAISE - DESIRED RAISE)
820D CDA780      =1786          CALL    CHKBSY
8210 CD1980      =1787          CALL    READ4
8213 012D00    D =1788          LXI     B,VELRS         ; SEND THE VELOCITY TO THE APU DATA STACK
8216 CD0080      =1789          CALL    SEND4
8219 01D583      =1790          LXI     B,TIMECV        ; SEND THE COEFFICIENT (TIMECV) TO APU STACK
821C CD0080      =1791          CALL    SEND4
821F 3612        =1792          MVI     M,FMUL          ; PERFORM (VELRS * TIMECO)
8221 013100    D =1793          LXI     B,ACCRS         ; CONVERT THE ACCELERATION INTO APU BFP
8224 CD4C80      =1794          CALL    CONVER
8227 014500    D =1795          LXI     B,DIFF1
822A CDA780      =1796          CALL    CHKBSY
822D CD0080      =1797          CALL    SEND4           ; SEND (CURRENT - DESIRED) TO APU
8230 3610        =1798          MVI     M,FADD          ; FIND (VELRS*TIMECV + (CURRENT - DESIRED))
8232 01D983      =1799          LXI     B,CONSTV
8235 CDA780      =1800          CALL    CHKBSY
8238 CD0080      =1801          CALL    SEND4
823B 3610        =1802          MVI     M,FADD          ; FIND (VELRS*TIMECV + (CURRENT - DESIRED) + CONSTV)
823D 014900    D =1803          LXI     B,SWFN
8240 CDA780      =1804          CALL    CHKBSY
8243 CD1980      =1805          CALL    READ4           ; READ AND STORE THIS
8246 3A4C00    D =1806          LDA     SWFN+3
8249 17          =1807          RAL
824A 9F          =1808          SBB     A               ;  A = 0FFH IF EVALUATED FUNCTION IS -VE
                 =1809                                  ;  A = 000H IF EVALUATED FUNCTION IS +VE
824B 215100    D =1810          LXI     H,FNSIGN        ; POINT H,L TO FNSIGN FOR COMPARISION
824E BE          =1811          CMP     M
824F C26082      =1812          JNZ     STOPRS          ; STOP IF NOT EQUAL
8252 B7          =1813          ORA     A               ; CHECK WHAT SIGNAL IS REQUIRED
8253 C25B82      =1814          JNZ     $+8
8256 3EF6        =1815          MVI     A,0F6H          ; SET THE LOWER DOWN SIGNAL
8258 C36282      =1816          JMP     $+10D
825B 3EF9        =1817          MVI     A,0F9H          ; SET THE RAISE UP SIGNAL
825D C36282      =1818          JMP     $+5
8260 3EFF        =1819 STOPRS:  MVI     A,0FFH          ; SET THE NO-LIFT SIGNAL
```

```
LOC  OBJ       LINE      SOURCE STATEMENT

8262 320300    D =1820       STA     SIGNAL      ; STORE THIS SIGNAL
8265 3A2400    D =1821       LDA     RSCRRT      ; GET THE CURRENT RAISE
8268 4F          =1822       MOV     C,A
8269 3A1B00    D =1823       LDA     RSDESR      ; GET THE DESIRED RAISE
826C 91          =1824       SUB     C           ; FIND THE DIFFERENCE
826D D27282      =1825       JNC     $+5
8270 2F          =1826       CMA                 ; FORM THE ABSOLUTE DIFFERENCE
8271 3C          =1827       INR     A
8272 FE02        =1828       CPI     02H
8274 D27B82      =1829       JNC     $+7
8277 9F          =1830       SBB     A           ; ENSURE NO-LIFT IF WITHIN THE TOLERANCE
8278 320300    D =1831       STA     SIGNAL
827B 011A00    D =1832       LXI     B,SLDESR    ; GET THE DESIRED SLEW POSITION
827E CD4280      =1833       CALL    SEND10      ; SEND THIS TO THE APU DATA STACK
8281 26FD        =1834       MVI     H,APUC      ; LOAD THE AUP COMMAND PORT ADDRESS
8283 361D        =1835       MVI     M,FLTS      ; FLOAT THIS INTEGER
8285 3A1A00    D =1836       LDA     SLDESR      ;  B <--- SLDESR
8288 47          =1837       MOV     B,A
8289 3A2300    D =1838       LDA     SLCRRT      ;  A <--- SLCRRT
828C 90          =1839       SUB     B           ;  SLCRRT - SLDESR
828D 9F          =1840       SBB     A           ;       A = 000H  IF SLCRRT >= SLDESR
                 =1841       ;       A = 0FFH  IF SLCRRT <  SLDESR
828E 325100    D =1842       STA     FNSIGN      ; STORE THIS FUNCTION SIGN
8291 014D00    D =1843       LXI     B,DUMMY1    ; LOAD ADDRESS FOR THE RESULT
8294 CDA780      =1844       CALL    CHKBSY
8297 CD1980      =1845       CALL    READ4
829A 012300    D =1846       LXI     B,SLCRRT    ; FORM THE DIFFERENCE BETWEEN THE CURRENT AND 040H
829D CD4280      =1847       CALL    SEND10
82A0 361D        =1848       MVI     M,FLTS
82A2 014D00    D =1849       LXI     B,DUMMY1
82A5 CDA780      =1850       CALL    CHKBSY
82A8 CD0080      =1851       CALL    SEND4
82AB 3611        =1852       MVI     M,FSUB      ; PERFORM (CURRENT SLEW - DESIRED SLEW)
82AD 012500    D =1853       LXI     B,VELSL     ; CONVERT THE VELOCITY INTO APU BFP FORMAT
82B0 CD4C80      =1854       CALL    CONVER
82B3 014500    D =1855       LXI     B,DIFF1     ; STORE (CURRENT SLEW - DESIRED SLEW)
82B6 CDA780      =1856       CALL    CHKBSY
82B9 CD1980      =1857       CALL    READ4
82BC 012500    D =1858       LXI     B,VELSL     ; SEND THE VELOCITY TO THE APU DATA STACK
82BF CD0080      =1859       CALL    SEND4
82C2 01CD83      =1860       LXI     B,TIMECH    ; SEND THE COEFFICIENT (TIMECH) TO APU STACK
82C5 CD0080      =1861       CALL    SEND4
82C8 3612        =1862       MVI     M,FMUL      ; PERFORM (VELSL * TIMECH)
82CA 01D183      =1863       LXI     B,CONSTH    ;
82CD CDA780      =1864       CALL    CHKBSY
82D0 CD0080      =1865       CALL    SEND4
82D3 3610        =1866       MVI     M,FADD
82D5 012900    D =1867       LXI     B,ACCSL     ; CONVERT THE ACCELERATION INTO APU BFP
82D8 CD4C80      =1868       CALL    CONVER
82DB 014500    D =1869       LXI     B,DIFF1
82DE CDA780      =1870       CALL    CHKBSY
82E1 CD0080      =1871       CALL    SEND4       ; SEND (CURRENT - DESIRED) TO APU
82E4 3610        =1872       MVI     M,FADD      ; FIND (VELSL*TIMECO + (CURRENT - DESIRED))
82E6 014900    D =1873       LXI     B,SWFN
82E9 CDA780      =1874       CALL    CHKBSY
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

82EC CD1980       =1875           CALL    READ4           ; READ AND STORE THIS
82EF 3A4C00   D   =1876           LDA     SWFN+3
82F2 17           =1877           RAL
82F3 9F           =1878           SBB     A               ;  A = 0FFH IF EVALUATED FUNCTION IS -VE
                  =1879                                   ;  A = 000H IF EVALUATED FUNCTION IS +VE
82F4 215100   D   =1880           LXI     H,FNSIGN        ; POINT H,L TO FNSIGN FOR COMPARISION
82F7 BE           =1881           CMP     M               ; COMPARE TO SEE IF SLEW MOVEMENT IS REQUIRED
82F8 210300   D   =1882           LXI     H,SIGNAL        ; POINT H,L TO THE PREVIOUS LIFTING SIGNAL SET
82FB C20C83       =1883           JNZ     STOPSL          ; STOP IF NOT EQUAL
82FE B7           =1884           ORA     A               ; CHECK WHAT SIGNAL IS REQUIRED
82FF C20783       =1885           JNZ     $+8D
8302 3E6F         =1886           MVI     A,06FH          ; SET THE SLEW LEFT SIGNAL
8304 C30E83       =1887           JMP     $+10D
8307 3E9F         =1888           MVI     A,09FH          ; SET THE SLEW RIGHT SIGNAL
8309 C30E83       =1889           JMP     $+5D
830C 3EFF         =1890 STOPSL:   MVI     A,0FFH          ; SET THE NO-SLEW SIGNAL
830E A6           =1891           ANA     M               ; FORM THE RESULTANT SIGNAL
830F 77           =1892           MOV     M,A
8310 3A2300   D   =1893           LDA     SLCRRT          ; GET THE CURRENT SLEW
8313 4F           =1894           MOV     C,A
8314 3A1A00   D   =1895           LDA     SLDESR          ; GET THE DESIRED SLEW
8317 91           =1896           SUB     C               ; FORM THE DIFFERENCE
8318 D21D83       =1897           JNC     $+5
831B 2F           =1898           CMA                     ; FORM THE ABSOLUTE DIFFERENCE
831C 3C           =1899           INR     A
831D FE02         =1900           CPI     02H             ; COMPARE WITH THE TOLERANCE
831F D22683       =1901           JNC     DEBUG2
8322 3EF0         =1902           MVI     A,0F0H          ; ENSURE NO-SLEW IF WITHIN THE BAND
8324 B6           =1903           ORA     M               ; FORM THE COMBINED SIGNAL 1111 XXXX
8325 77           =1904           MOV     M,A
8326 7E           =1905 DEBUG2:   MOV     A,M
8327 F6F0         =1906           ORI     0F0H            ; CHECK IF LIFT SIGNAL HAS BEEN SET
8329 3C           =1907           INR     A               ;     (I.E.  PATTERN NOT XXXX 1111)
832A CA3383       =1908           JZ      $+9D            ; SEND THE SIGNAL IF LIFT IS NOT REQUIRED
832D 3D           =1909           DCR     A               ; OTHERWISE, SEND LIFT SIGNAL ONLY
832E D300         =1910           OUT     00H
8330 C33B83       =1911           JMP     STOPF1
8333 7E           =1912           MOV     A,M
8334 D300         =1913           OUT     00H
8336 FEFF         =1914           CPI     0FFH
8338 CA4083       =1915           JZ      STOPMV          ; JUMP IF THE DIRECTIONAL VALVES ARE CLOSED
833B C1           =1916 STOPF1:   POP     B               ; OTHERWISE, RECOVER THE REGISTERS AND
833C D1           =1917           POP     D
833D E1           =1918           POP     H
833E F1           =1919           POP     PSW
833F C9           =1920           RET                     ; RETURN
8340 DB01         =1921 STOPMV:   IN      001H            ; SET THE SLOW SPEED MODE
8342 F603         =1922           ORI     003H
8344 321700   D   =1923           STA     MDSIGN          ; STORE THE SPEED MODE SETTING
8347 D301         =1924           OUT     01H             ; SEND THIS TO THE VALVES
8349 21FFFF       =1925           LXI     H,0FFFFH        ; INTRODUCE TIME DELAY FOR VALVE TO RESPOND
834C 97           =1926           SUB     A
834D 2B           =1927           DCX     H
834E B4           =1928           ORA     H
834F B5           =1929           ORA     L
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

8350 C24C83       =1930               JNZ     $-4
8353 CD4000       =1931 ADJUST:       CALL    SETLNK      ; ADJUST THE FINAL BOOM POSITION IN SLOW MODE
8356 3AA21B       =1932               LDA     CRRTSL      ; GET THE CURRENT BOOM POSITION
8359 6F           =1933               MOV     L,A
835A 3AAD1B       =1934               LDA     CRRTRS
835D 67           =1935               MOV     H,A
835E 222300     D =1936               SHLD    SLCRRT      ; STORE THIS IN MEMORY OF THIS SYSTEM
8361 3ADD1B       =1937               LDA     CUTFOR      ; GET THE CUTTER FORCE
8364 321D00     D =1938               STA     FORCE       ; AND STORE THIS INFORMATION
8367 CD5200       =1939               CALL    OFFLNK
836A 3A1D00     D =1940               LDA     FORCE       ; CHECK IF THE CUTTER FORCE EXCEEDS THE UPPER LIMIT
836D FEE5         =1941               CPI     0E5H
836F D2AF83       =1942               JNC     STOPF2      ; STOP IF THE FORCE IS TOO HIGH
8372 2A2300     D =1943               LHLD    SLCRRT      ; SET THE APPROPRIATE SIGNALS FOR THE DIRECTIONAL VALVES
8375 EB           =1944               XCHG                ; PUT THE CURRENT BOOM POSITION IN D,E
8376 2A1A00     D =1945               LHLD    SLDESR      ; PUT THE DESIRED BOOM POSITION IN H,L
8379 7B           =1946               MOV     A,E         ; DETERMINE THE SLEW SIGNAL
837A 95           =1947               SUB     L
837B DA8383       =1948               JC      $+8
837E 0E6F         =1949               MVI     C,06FH      ; SET THE SLEW LEFT SIGNAL
8380 C38783       =1950               JMP     $+7
8383 2F           =1951               CMA
8384 3C           =1952               INR     A           ; FIND THE ABSOLUTE DIFFERENCE
8385 0E9F         =1953               MVI     C,09FH      ; SET THE SLEW RIGHT SIGNAL
8387 FE02         =1954               CPI     02H         ; COMPARE WITH THE ADC TOLERANCE
8389 D28E83       =1955               JNC     $+5
838C 0EFF         =1956               MVI     C,0FFH      ; SET THE NO-SLEW SIGNAL IF WITHIN THIS RANGE
838E 7A           =1957               MOV     A,D         ; DETERMINE THE RAISE SIGNAL
838F 94           =1958               SUB     H
8390 DA9883       =1959               JC      $+8
8393 06F6         =1960               MVI     B,0F6H      ; SET THE LOWER DOWN SIGNAL
8395 C39C83       =1961               JMP     $+7
8398 2F           =1962               CMA                 ; FIND THE ABSOLUTE DISTANCE BETWEEN THE CURRENT
8399 3C           =1963               INR     A           ; AND THE DESIRED RAISE POSITIONS
839A 06F9         =1964               MVI     B,0F9H      ; SET THE RAISE UP SIGNAL
839C FE02         =1965               CPI     02H         ; COMPARE WITH THE ADC TOLERANCE
839E D2A983       =1966               JNC     $+5
83A1 06FF         =1967               MVI     B,0FFH      ; SET THE NO-RAISE SIGNAL IF WITHIN THE RANGE
83A3 78           =1968               MOV     A,B
83A4 A1           =1969               ANA     C           ; FORM THE COMBINED SIGNAL
83A5 D300         =1970               OUT     00H         ; SEND THIS SIGNAL TO OPERATE THE DIRECTIONAL VALVES
83A7 FEFF         =1971               CPI     0FFH        ; CHECK IF THE DESIRED POSITION HAS BEEN REACHED
83A9 C25383       =1972               JNZ     ADJUST      ; LOOP BACK IF NOT
83AC C33B83       =1973               JMP     STOPF1      ; JUMP TO RETURN IF YES
83AF 3EFF         =1974 STOPF2:       MVI     A,0FFH      ; SET NO-MOVE SIGNAL DUE TO HIGH CUTTING FORCE
83B1 D300         =1975               OUT     00H
83B3 C35383       =1976               JMP     ADJUST      ; LOOP BACK TO ADJUST THE FINAL POSITION
83B6 DB01         =1977 NOSORT:       IN      01H         ; THE PROFILE DATA HAS NOT BEEN SORTED
83B8 E6FC         =1978               ANI     0FCH        ; SET THE FAST SPEED MODE
83BA C3F205       =1979               JMP     SMOVEF      ; JUMP TO CONTINUE
83BD 3EFF         =1980 STOPMF:       MVI     A,0FFH      ; CLOSE ALL THE DIRECTIONAL VALVES
83BF D300         =1981               OUT     00H
83C1 DB01         =1982               IN      01H         ; SET SLOW SPEED MODE
83C3 F609         =1983               ORI     09H
83C5 321700     D =1984               STA     MDSIGN
```

```
LOC  OBJ          LINE         SOURCE STATEMENT

83C8 D301        =1985            OUT     01H
83CA C33B83      =1986            JMP     STOPF1
83CD 99          =1987 TIMECH:    DB      099H,0F9H,0B7H,00FH
83CE F9          =
83CF B7          =
83D0 0F          =
83D1 44          =1988 CONSTH:    DB      044H,01FH,0EBH,0FFH
83D2 1F          =
83D3 EB          =
83D4 FF          =
83D5 00          =1989 TIMECV:    DB      000H,0F6H,0E3H,00FH
83D6 F6          =
83D7 E3          =
83D8 0F          =
83D9 00          =1990 CONSTV:    DB      000H,000H,080H,080H
83DA 00          =
83DB 80          =
83DC 80          =
                  1991 $INCLUDE(:F1:DATA.SRC)
                 =1992 ;*****************************************************************
                 =1993 ;*      DATA SEGMENT FOR THE BOOM POSITIONAL CONTROL PROGRAM      *
                 =1994 ;*****************************************************************
                 =1995 PUBLIC  CRRTSL,CRRTRS,MOVEST,DESRSL,DESRRS,PFLEND,PFLST,CUTTER
                 =1996 PUBLIC  SLVEL,SLACC,RSVEL,RSACC,STADD,LASTAD,DDR01,TEMP,TEMP1,TEMPR,TEMPR1
                 =1997 PUBLIC  SLCRRT,RSCRRT,SLDESR,RSDESR,VELSL,VELRS,ACCSL,ACCRS,PREVSL,PREVRS
                 =1998 PUBLIC  SIMUX,SIMUY,SIMUAX,SIMUAY,SIMUAZ
                 =1999 ASEG
1BA2             =2000 ORG     01BA2H
                 =2001 ;*****************************************************************
                 =2002 ;*      SYMBOLS TO BE USED IN THE DATA CAPTURING SYSTEM           *
                 =2003 ;*****************************************************************
                 =2004 ;SYMBOL                    DEFINITION
                 =2005 ;------                    ----------
1BA2             =2006 CRRTSL: DS    1        ; CURRENT SLEW POSITION        (1 BYTE)
1BAD             =2007 ORG     01BADH
1BAD             =2008 CRRTRS: DS    1        ; CURRENT RAISE POSITION       (1 BYTE)
1BDA             =2009 ORG     01BDAH
1BDA             =2010 DESRSL: DS    1        ; DESIRED SLEW POSITION        (1 BYTE)
1BDB             =2011 DESRRS: DS    1        ; DESIRED RAISE POSITION       (1 BYTE)
1BDC             =2012 MOVEST: DS    1        ; CURRENT MOVE LEVER STATUS    (1 BYTE)
1BDD             =2013 CUTFOR: DS    1        ; CUTTER MOTOR CURRENT         (1 BYTE)
1BE3             =2014 ORG     01BE3H
1BE3             =2015 PREVSL: DS    1        ; CURRENT SLEW POSITION        (1 BYTE)
1BE4             =2016 PREVRS: DS    1        ; CURRENT RAISE POSITION       (1 BYTE)
1BE5             =2017 SLVEL:  DS    4        ; CURRENT SLEW VELOCITY        (4 BYTES)
1BE9             =2018 SLACC:  DS    4        ; CURRENT SLEW ACCELERATION    (4 BYTES)
1BED             =2019 RSVEL:  DS    4        ; CURRENT RAISE VELOCITY       (4 BYTES)
1BF1             =2020 RSACC:  DS    4        ; CURRENT RAISE ACCELERATION   (4 BYTES)
1BFE             =2021 ORG     1BFEH
1BFE             =2022 PFLEND: DS    2        ; LAST ADDRESS FOR PROFILE DATA (2 BYTES)
1C00             =2023 PFLST:  DS 400H        ; STARTING ADDRESS FOR PROFILE DATA (MAX. 400H BYTES)
BBFE             =2024 ORG     0BBFEH
                 =2025 ;*****************************************************************
                 =2026 ;*     SYMBOLS TO BE USED IN THE MAIN CONTROL SYSTEM PROGRAM      *
                 =2027 ;*****************************************************************
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                  =2028 ;SYMBOL                    DEFINITION
                  =2029 ;------                    ----------
BBFE              =2030 LASTAD: DS   2             ; LAST ADDRESS FOR THE PROFILE (2 BYTES)
BC00              =2031 STADD:  DS 400H            ; DATA STORAGE FOR THE PROFILE (MAX. 400H BYTES)
                  =2032 DSEG                       ; DEFINE DATA SEGMENT
0000              =2033 STKPTR: DS   0             ; DEFINE THE HIGHEST STACK BYTE TO BE THE ONE BELOW
                  =2034                            ; THE LOWEST DATA MEMORY BYTE
0000              =2035 INTMSK: DS   1             ; STORAGE LOCATION FOR THE INTERRUPT MASK
0001              =2036 CUTSTP: DS   1             ; LOCATION FOR THE CUTTER STEP SIZE IN ADVANCING
0002              =2037 SORTFL: DS   1             ; FLAG FOR SORTED PROFILE DATA
                  =2038                            ;   00  --- NOT YET SORTED
                  =2039                            ;   FF  --- SORTING COMPLETED
0003              =2040 SIGNAL: DS   1             ; SIGNAL FOR THE DIRECTIONAL VALVES
0004              =2041 STARCV: DS   2             ; DEFINE THE STORAGE FOR STARTING CAVITY 'RAISE' LEVEL
0006              =2042 LASTCV: DS   2             ; DEFINE THE STORAGE FOR THE LAST 'CAVITY' LEVEL
0008              =2043 LASTSO: DS   2             ; DEFINE THE STORAGE FOR THE LAST SORTED CAVITY ADDRESS
000A              =2044 LEVEL:  DS   2             ; DEFINE THE STORAGE FOR THE ADDRESS FOR 'X'
000C              =2045 DMIN1:  DS   2             ; DEFINE THE STORAGE FOR TEMPORARY MIN
000E              =2046 DMAX1:  DS   2             ; DEFINE THE STORAGE FOR TEMPORARY MAX
0010              =2047 FIRSL:  DS   2             ; TEMPORARY STORAGE FOR THE NEXT TWO DESIRED SLEW VALUES
0012              =2048 FOURST: DS   2             ; TEMPORARY STORAGE FOR 4*(CUTTER STEP SIZE)
0014              =2049 LASTCT: DS   2             ; STORAGE FOR THE ADDRESS OF THE LAST DATA BLOCK IN CAVITY CUTTING
0016              =2050 HALFST: DS   1             ; STORAGE FOR HALF CUTTER STEP SIZE
0017              =2051 MDSIGN: DS   1             ; STORAGE FOR THE BOOM MOVING SPEED MODE
0018              =2052 DDR01:  DS   1             ; DATA DIRECTION REGISTER FOR PORT 1
0019              =2053 CUTTER: DS   1             ; CUTTER STATUS --- 00=STOPPED, 0FFH=STARTED
001A              =2054 SLDESR: DS   1             ; DESIRED SLEW POSITION          (1 BYTE)
001B              =2055 RSDESR: DS   1             ; DESIRED RAISE POSITION         (1 BYTE)
001C              =2056 STMOVE: DS   1             ; 'MOVE' CONTROL STATUS          (1 BYTE)
001D              =2057 FORCE:  DS   1             ; CUTTER MOTOR CURRENT           (1 BYTE)
001E              =2058 SIMUX:  DS   1             ; SIMULATED X DISPLACEMENT       (1 BYTE)
001F              =2059 SIMUY:  DS   1             ; SIMULATED Y DISPLACEMENT       (1 BYTE)
0020              =2060 SIMUAX: DS   1             ; SIMULATED ALPHA X              (1 BYTE)
0021              =2061 SIMUAY: DS   1             ; SIMULATED ALPHA Y              (1 BYTE)
0022              =2062 SIMUAZ: DS   1             ; SIMULATED ALPHA Z              (1 BYTE)
0023              =2063 SLCRRT: DS   1             ; CURRENT SLEW POSITION          (1 BYTE)
0024              =2064 RSCRRT: DS   1             ; CURRENT RAISE POSITION         (1 BYTE)
0025              =2065 VELSL:  DS   4             ; CURRENT SLEW VELOCITY          (4 BYTES)
0029              =2066 ACCSL:  DS   4             ; CURRENT SLEW ACCELERATION      (4 BYTES)
002D              =2067 VELRS:  DS   4             ; CURRENT RAISE VELOCITY         (4 BYTES)
0031              =2068 ACCRS:  DS   4             ; CURRENT RAISE ACCELERATION     (4 BYTES)
0035              =2069 TEMP:   DS   4             ; TEMPORARY STORAGE LOCATION     (4 BYTES)
0039              =2070 TEMP1:  DS   4
003D              =2071 TEMPR:  DS   4             ; TEMPORATY STORAGE FOR RESULT (4 BYTES)
0041              =2072 TEMPR1: DS   4
0045              =2073 DIFF1:  DS   4             ; DIFFERENCE BETWEEN CURRENT AND DESIRED (4 BYTES)
0049              =2074 SWFN:   DS   4             ; SWITCHING FUNCTION VALUE       (4 BYTES)
004D              =2075 DUMMY1: DS   4             ;
0051              =2076 FNSIGN: DS   1             ; EXPECTED SIGN FOR SWITCHING  (1 BYTE)
                  =2077 ;*********************************************************
                  =2078 ;*                                                       *
                  =2079 ;*********************************************************
                  =2080 $EJECT
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

                   =2081 END


PUBLIC SYMBOLS
ACCRS  D 0031   ACCSL  D 0029   CAVITY A 0218   CHKBSY A 80A7   CONVER A 804C   CRRTRS A 1BAD   CRRTSL A 1BA2
CUTSTP D 0001   CUTTER D 0019   DDR01  D 0018   DESRRS A 1BDB   DESRSL A 1BDA   DMIN1  D 000C   EXPAND A 0509
FIRSL  D 0010   FMOVE  A 80AE   FOURST D 0012   INTMSK D 0000   LASTAD A BBFE   LASTCV D 0006   LASTSO D 0008
LEVEL  D 000A   MOVE   A 02DC   MOVEST A 1BDC   OFFLNK A 0052   PFLEND A 1BFE   PFLST  A 1C00   PREVRS A 1BE4
PREVSL A 1BE3   PROFIL A 00B5   READ2  A 802E   READ4  A 8019   RSACC  A 1BF1   RSCRRT D 0024   RSDESR D 001B
RST45  A 032B   RSTAPU A 8098   RSVEL  A 1BED   SEND10 A 8042   SEND2  A 8012   SEND30 A 8035   SEND4  A 8000
SETLNK A 0040   SHORT  A 0578   SIGNAL D 0003   SIMUAX D 0020   SIMUAY D 0021   SIMUAZ D 0022   SIMUX  D 001E
SIMUY  D 001F   SLACC  A 1BE9   SLCRRT D 0023   SLDESR D 001A   SLVEL  A 1BE5   SMOVE  A 05EA   SMOVE1 A 0740
SMOVEF A 05F2   SORT   A 03DD   SORTFL D 0002   STADD  A BC00   STARCV D 0004   STKPTR D 0000   TEMP   D 0035
TEMP1  D 0039   TEMPR  D 003D   TEMPR1 D 0041   TRANSF A 0097   TRNPFL A 0075   VELRS  D 002D   VELSL  D 0025

EXTERNAL SYMBOLS


USER SYMBOLS
ACCRS  D 0031   ACCSL  D 0029   ACOS   A 0006   ADJUST A 8353   ANYMOD A 060D   APUC   A 00FD   APUD   A 00FC
ASIN   A 0005   ATAN   A 0007   CALEXP A 03C5   CALSMV A 0266   CAVITY A 0218   CHECK2 A 0462   CHKB1  A 06A5
CHKB2  A 06A4   CHKB3  A 06B5   CHKB4  A 06B6   CHKBND A 0681   CHKBSY A 80A7   CHKPOS A 076F   CHSD   A 0034
CHSF   A 0015   CHSS   A 0074   CLOSL  A 0636   CLOSR  A 067E   COLDST A 0008   COMPL  A 03C0   CONSTH A 83D1
CONSTV A 83D9   CONT10 A 8088   CONT11 A 041A   CONT12 A 0486   CONT20 A 808B   CONT21 A 0411   CONT22 A 0458
CONT31 A 0422   CONT32 A 045E   CONT41 A 03F0   CONT42 A 0449   CONT5  A 05AC   CONT52 A 049D   CONT6  A 05B0
CONT62 A 047B   CONTC0 A 0299   CONTC2 A 02AE   CONTC5 A 027F   CONTP1 A 01FB   CONTR1 A 0380   CONTR2 A 038C
CONVER A 804C   COS    A 0003   CRRTRS A 1BAD   CRRTSL A 1BA2   CSTORE A 04B1   CTPRO1 A 0109   CTPRO2 A 0157
CUTFOK A 1BDD   CUTSTP D 0001   CUTTER D 0019   DADD   A 002C   DDIV   A 002F   DDR01  D 0018   DEBUG2 A 8326
DESRRS A 1BDB   DESRSL A 1BDA   DIFF1  D 0045   DMAX1  D 000E   DMIN1  D 000C   DMUL   A 002E   DMUU   A 0036
DOWN1  A 066A   DSUB   A 002D   DUMMY1 D 004D   DUPLI  A 04C7   DUPLI1 A 04CE   DUPLI2 A 04E0   EXP    A 000A
EXPA1  A 0525   EXPA2  A 053D   EXPA3  A 0541   EXPA4  A 054E   EXPAF  A 056E   EXPAND A 0509   FADD   A 0010
FAST1  A 01BF   FASTM2 A 06E1   FASTM3 A 06E9   FASTS  A 00E6   FASTS1 A 00F7   FASTSL A 06C3   FCONT  A 8117
FCONT1 A 8103   FDIV   A 0013   FIRSL  D 0010   FIXD   A 001E   FIXS   A 001F   FLTD   A 001C   FLTS   A 001D
FMOVE  A 80AE   FMOVES A 05EE   FMUL   A 0012   FNSIGN D 0051   FORCE  D 001D   FOURST D 0012   FSUB   A 0011
HALFST D 0016   INTMSK D 0000   LASTAD A BBFE   LASTCT D 0014   LASTCV D 0006   LASTSO D 0008   LEFT1  A 062C
LEVEL  D 000A   LN     A 0009   LOG    A 0008   LOOP   A 01F0   LOOP1  A 007F   LOOP11 A 03EB   LOOP12 A 0441
LOOP3  A 00A5   MDSIGN D 0017   MOVE   A 02DC   MOVE1  A 0316   MOVE2  A 02E7   MOVEST A 1BDC   NEGAT  A 8078
NODUP  A 04E6   NOMOVE A 031C   NORED  A 05E2   NOSORT A 83B6   NOTLOW A 0170   NOTSRT A 01B0   NOTUP  A 0119
NOVER1 A 06FC   NTFAST A 06EA   NTSORT A 0651   NTSRT1 A 0728   OFFLNK A 0052   ONVER2 A 0717   OUTRS  A 072C
OVER1  A 0133   PFLEND A 1BFE   PFLST  A 1C00   POPD   A 0038   POPF   A 0018   POPS   A 0078   POSIT  A 8062
PREVRS A 1BE4   PREVSL A 1BE3   PROFIL A 00B5   PTOD   A 0037   PTOF   A 0017   PTOS   A 0077   PUP1   A 001A
PWR    A 000B   READ2  A 802E   READ4  A 8019   READL3 A 801D   READL4 A 801E   READL5 A 8023   REDU1  A 05DD
REDU2  A 05E1   REDUCE A 05B6   REDULS A 024D   REDUST A 023D   RET10  A 8094   RIGHT1 A 0631   RSACC  A 1BF1
RSCMP  A 0638   RSCRRT D 0024   RSDESR D 001B   RST38  A 0059   RST45  A 032B   RST55  A 002C   RST65  A 0034
RST75  A 003C   RSTAPU A 8098   RSVEL  A 1BED   S6LOOP A 81AD   SADD   A 006C   SDIV   A 006F   SEND10 A 8042
SEND2  A 8012   SEND30 A 8035   SEND31 A 803E   SEND4  A 8000   SENDL1 A 8004   SENDL2 A 8007   SETLNK A 0040
SHORT  A 0578   SIGNAL D 0003   SIMUAX D 0020   SIMUAY D 0021   SIMUAZ D 0022   SIMUX  D 001E   SIMUY  D 001F
SIN    A 0002   SLACC  A 1BE9   SLCRRT D 0023   SLDESR D 001A   SLVEL  A 1BE5   SMOVE  A 05EA   SMOVE1 A 0740
SMOVE2 A 0742   SMOVEF A 05F2   SMUL   A 006E   SMUU   A 0076   SORT   A 03DD   SORTFL D 0002   SPEED  A 81AB
SQRT   A 0001   SSUB   A 006D   STADD  A BC00   STARCV D 0004   START  A 0000   STKPTR D 0000   STMOVE D 001C
STOFP1 A 0209   STOP1  A 0434   STOP2  A 046F   STOPF1 A 83BB   STOPF2 A 83AF   STOPMD A 0738   STOPMF A 83BD
STOMMV A 8340   STOPRS A 8260   STOPSL A 830C   SWFN   D 0049   TAN    A 0004   TEMP   D 0035   TEMP1  D 0039
TEMPR  D 003D   TEMPR1 D 0041   TIMECH A 83CD   TIMECV A 83D5   TRANSF A 0097   TRAP   A 0024   TRNPFL A 0075
UNDER1 A 0182   UP1    A 0674   VELRS  D 002D   VELSL  D 0025   VERT2  A 0704   VERT3  A 0701   VERT4  A 0722
VERT5  A 071C   XCHD   A 0039   XCHF   A 0019   XCHS   A 0079
```

APPENDIX I

Brief Description of the Hydraulic Circuit for the Boom Ripper under Test

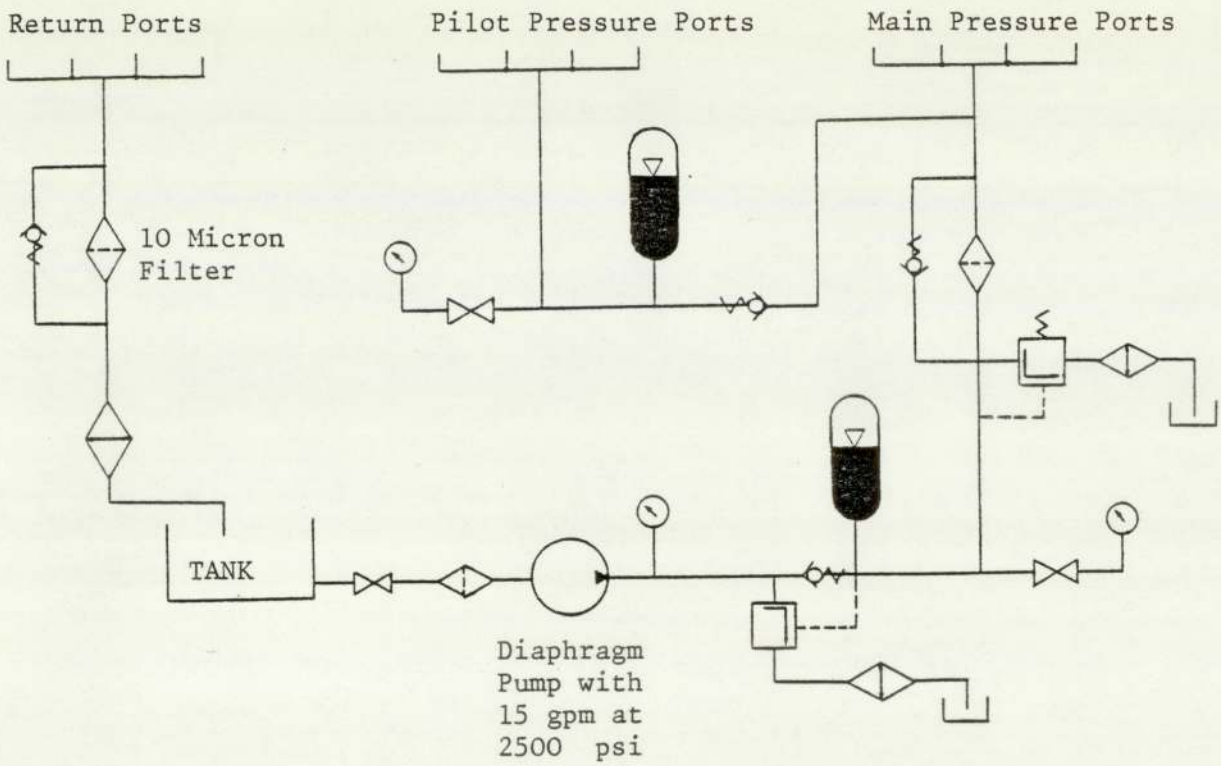


Figure I.1 Schematic Diagram for the Hydraulic Power Supply Circuit

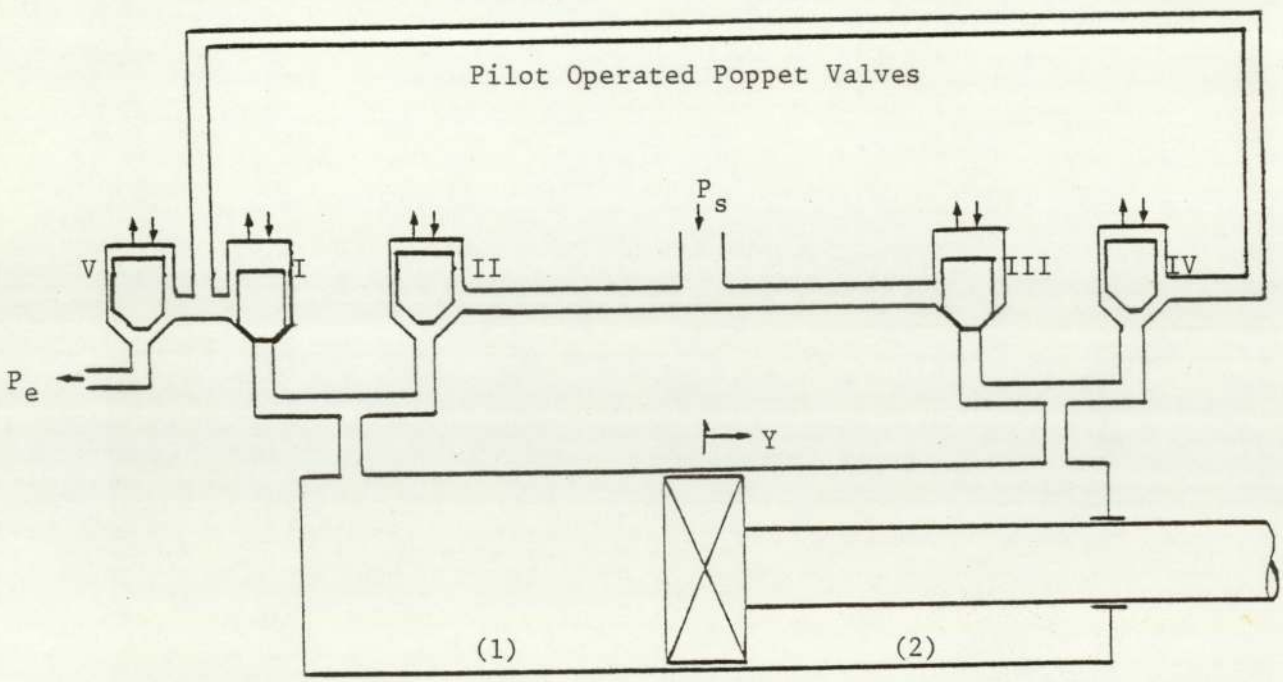


Figure I.2 Arrangement for One of the Hydraulic Actuators in the Boom Ripper under Test

The boom ripper under test is connected to a hydraulic power pack which delivers the 5/95 diluted emulsion at a flow rate of 1.135 litres/sec with a system supply pressure of 172.4 bars using a diaphragm pump. Fig I.1 shows the schematic diagram for this hydraulic power supply circuit.

Figure I.2 shows the arrangement of one of the hydraulic actuators used in the test rig. Only the main poppets are shown in this diagram, they are driven by their pilot poppet valves, which are operated on the amplified electrical signals from the microprocessor control system (please refer to Fig 6.2).

Four of these valves are the directional control valves: this actuator is moved towards right by opening the valve-pair (II) and (IV), while it will be forced to go left by opening the poppet-pair (I) and (III). The poppet (V) is the metering out control valve, which is also pilot-valve operated. When the directional control valves are opened, the poppet (V) can be operated in one of the following two modes: (i) it can be fully opened to provide the maximum flow area so that the actuator will move at the high speed, or (ii) this can be closed to leave a small opening to reduce the fluid flow rate by this metering out technique for moving the hydraulic actuator at a slow speed.

All the four hydraulic actuators are connected in the same way as described in the above paragraph. The paired actuators for the boom arcing or lifting share the same

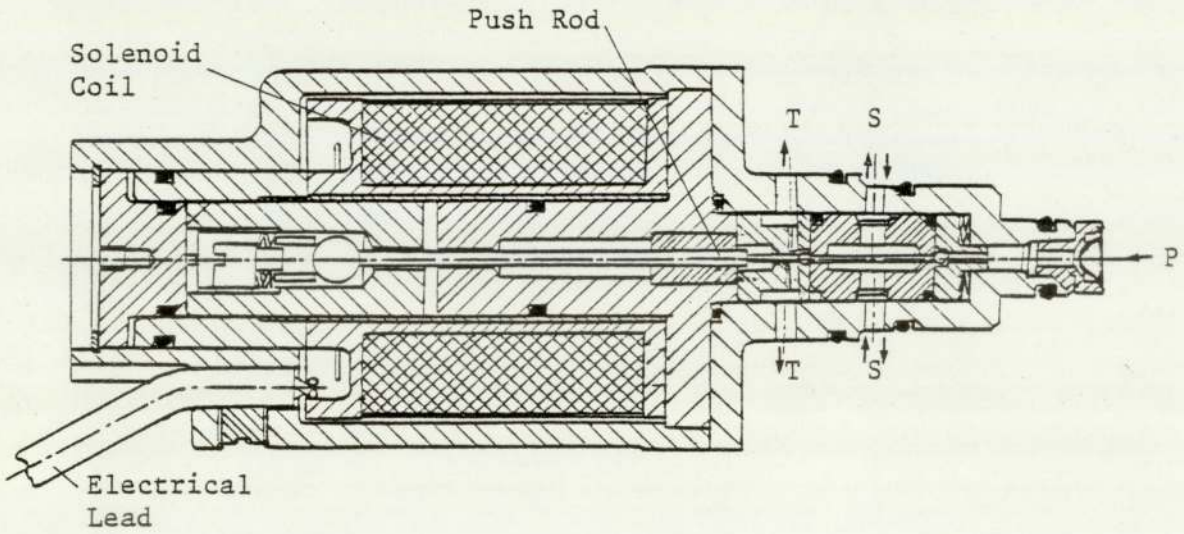


Figure I.3 Cross-section of the Solenoid Operated Pilot Valve used in the System under Test

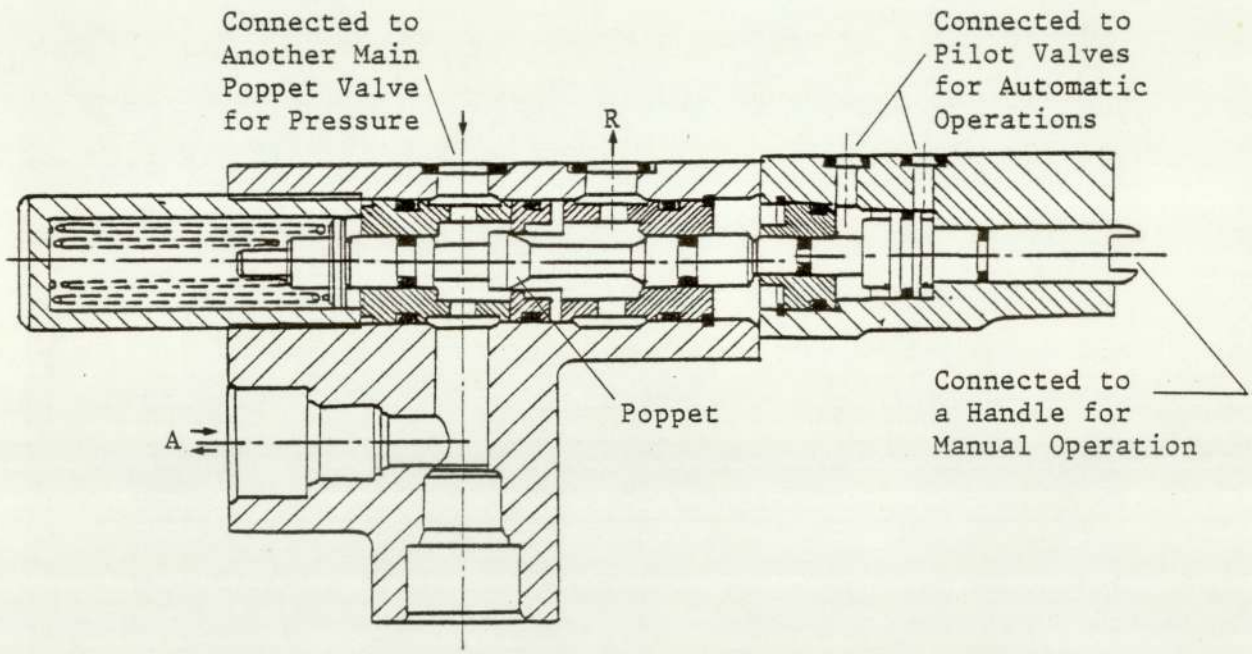


Figure I.4 Cross-section of the Main Poppet Valve used in the System under Test

set of directional control and metering out valves, so that only five pilots are operated to provide the desired boom movement in either slewing or lifting.

Figure I.3 shows the cross-section of one of the solenoid operated pilot valves used in the system under test, while Figure I.4 shows the cross-section of one of the main poppets used. When the solenoid of the pilot valve is not operated, the push rod is brought to its leftmost position by the spring, the ball valve between P and S is forced to rest on its seat by the supply pressure at P, but the ball valve between S and T is opened, therefore the fluid will flow from S to T. If this solenoid is energised, the push rod will be pulled to the right to reduce the gap by the electro-magnetic force generated. This action causes the ball valve between S and T to close with that between P and S to open. Therefore, the fluid will then flow from P to S. The S of this pilot is connected to operate the main poppet. If the pilot solenoid is de-energised, the main poppet will be forced to rest on its seat by its spring, therefore the fluid will flow between A and the paired poppet. When the pilot solenoid is energised, the poppet will be lifted off its seat by the pilot pressure and the fluid flows between A and R. The R in one of the paired poppets is connected to the system supply with R in the other one connected to the system exhaust.

The performance of these valves is being studied in a parallel project by J H Knight at Aston University. The Solenoid takes time to establish the magnetic flux when

energised and to remove the residual electro-magnetism when it is de-energised. Therefore there is a time delay in the pilot operation. This time delay can be shortened by reducing the mass of the moving parts in the solenoid. It is found that the time delay in the main poppet operation is mainly due to that in the pilot stage.