HIERARCHICAL COMPUTER CONTROL USING MULTI-MICROPROCESSOR SYSTEMS

BY

HEMANT ANANT DESHMUKH

--ooOoo--

THESIS SUBMITTED FOR THE AWARD OF THE DEGREE OF DOCTOR OF PHILOSOPHY

--ooOoo--

DEPARTMENT OF SYSTEMS SCIENCE
THE CITY UNIVERSITY
LONDON

OCTOBER 1981

TO MY LATE FATHER

CONTENTS

CONTENTS continued PAGE

CONTENTS continued PAGE

CONTENTS continued PAGE

TABLES

ILLUSTRATIONS

ILLUSTRATIONS continued PAGE

ILLUSTRATIONS continued PAGE

ILLUSTRATIONS continued PAGE

ILLUSTRATIONS continued PAGE

ACKNOWLEDGEMENTS

I would like to extend my thanks to my thesis supervisor, Professor P. D. Roberts, for his advice, guidance and interest throughout. I am also indebted to [redacted] for his valuable assistance during the development of the work. I would also like to thank [redacted] for their helpful suggestions.

I am grateful to [redacted] of the Department of Electrical Engineering and [redacted] of Computer Unit for their helpful suggestions and valuable assistance in the Microprocessor Laboratory.

Finally, I would like to thank my wife for her enduring patience and constant support. Thanks are also due to [redacted] for her proficient typing.

The work was supported by the Science Research Council in the form of a grant.

DECLARATION

The work described in this thesis was carried out in the Department of Systems Science, The City University, London, under the supervision of Professor P. D. Roberts. No part of this work has been submitted for any other degree. All sources of information have been duly referenced.

I grant powers of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to me. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

October 1981

PUBLICATIONS

Part of this work was used as a basis of the following papers:

1. A hierarchically structured multi-microprocessor system, presented at the Fifth EUROMICRO Symposium on Microprocessing and Microprogramming, held on August 28-30 1979 at Göteborg.

2. Model of a processor within a distributed computing system, presented at the Seventh EUROMICRO Symposium on Microprocessing and Microprogramming, held on September 8-10 1981 at Paris.

ABSTRACT

The recent advances in integrated circuits technology and the consequent emergence of microprocessors have increased interest in developing multi-microprocessor systems. Microprocessors and microcomputers are being coupled together in increasingly large numbers in a tightly or loosely coupled manner as distributed computing structures which include complex interconnection mechanisms and interfaces to link these to an application. Superimposed on this hardware structure, software is written to provide the communication protocols, synchronisation between sequential processes and application programs and so on. A microprocessor or a microcomputer, as a processing element, is a major programmable component in these distributed computing systems which share the primary advantages over conventional large computer systems of low cost, reliability and possibly speed of operation. The main task of implementing a distributed computing system interfaced to a real-time large-scale complex system is the partitioning of the main control problem into smaller subproblems and identifying the interactions between them, so that the subproblems and interactions can be programmed into the processing elements.

This thesis is aimed at the study of hierarchical computer control using multi-microprocessor systems. In particular, it is concerned with the design and practical application of microprocessors and a PDP-11/10 minicomputer

to on-line distributed and hierarchical control of a laboratory-based pilot scale Travelling Load Furnace (TLF). The basic processing module from which the system is configured is known as a Hierarchical Microprocessor System Unit and consists of a number of Fairchild/Mostek F8 microprocessor system chips, a common block of semiconductor memory and a bidirectional scratchpad memory interface. The configuration is designed so that a single HMSU can be used either independently or as a building block in an expandable hierarchical environment. The hierarchical control scheme involves the use of three processing units of the HMSU to implement three term control action on the eight zones of the TLF. The eight zones of the TLF are divided into 2, 3 and 3 heating zones designated as the preheat, heat and soak sections respectively. Any one section can be assigned to any one of the processing units (e.g. a Master processor or either of the two slave processors) of the HMSU. Operator communication and overall co-ordination of the system is performed by a host PDP-11/10 minicomputer.

The main outcome of the research reveals that it is feasible to implement multi-microprocessor systems such as the HMSU for real-time, on-line hierarchical computer control of industrial processes such as the TLF. However, in order to justify the cost-effectiveness of such systems, the need for proper development tools such as Microprocessor Development Systems (MDS) with in-circuit-emulation capabilities, testing and debugging tools such as

Logic Analysers etc. is paramount. The experience gained as a result of practical implementation of the HMSU for the control of the TLF has been invaluable so far as the insight into the problems of developing hardware, software and that of partitioning of a control problem into smaller subproblems and their interactions is concerned. The work reported in this thesis will provide a useful foundation for evaluating and extending further possibilities of developing multi-microprocessor systems.

CHAPTER 1 - INTRODUCTION

The impact of recent advances in Large Scale Integrated (LSI) circuit technology towards low-cost processors and memory modules has caused increased experimentation with multiple processors, multi-microprocessors and multi-microcomputer organisations. A variety of multi-processor and multi-microprocessor systems have been described which use similar hardware but which differ in the way in which the components are interconnected. The spectrum of these Distributed Computing Systems range from networks of conventional computers, systems containing sets of microprocessors and novel forms of highly parallel computer architectures with greater integration of processing and storage. The motivations and importance of research into these distributed computing systems are many and varied (SRC 1980). These include:

1. Performance: eventually it will be impossible to increase the speed of a single processor and retain commercial viability. Several processors, co-operating on a single task, will be the only way to greatly enhance performance.

2. Reliability: a fully distributed system should be able to tolerate faults caused by either software or hardware. Hardware faults might be tolerated by having more than one of each critical element. Software faults might be reduced by running different algorithms in parallel and checking the validity of results.

3. Clarity: many problems are naturally parallel. Some problems are inherently simpler if expressed as a set of interconnected and communicating processes. If a problem's solution is expressed in this way, it might be easier to verify the correctness for the whole solution by partitioning it into subproblem solutions of individual processes and their interactions. This approach inherently gives a better insight into a large-scale complex problem.

4. Distribution: in areas such as real-time control, it is often important that processor power is available where it is required in order to minimise the bandwidth requirements of data paths.

5. Cost: the low cost of microprocessors and memory systems will allow certain tasks to be performed more economically on sets of microprocessors than on a single mainframe processor.

In the Department of Systems Science at The City University, a research program in computer control of Travelling Load Furnaces (TLFs) and their application is being carried out, with the object of finding improved and more efficient control schemes to be applied in industry. To this end, the design and modelling of an experimental Travelling Load Furnace for computer control was undertaken by R. Caffin in 1972 and subsequently, further experimentation was performed by H. H. Sheena using a digital Ferranti ARGUS 500 computer in 1977. Based on this

research, a project entitled "Microprocessor control of a Travelling Load Oven" was successfully completed by the author in 1977 using the Fairchild F8 microprocessor evaluation kit. This work and the influence of the above motivations has directed this research with the following objectives:

1. To study parallel processing aspect of on-line computer control.

2. To design a multi-microprocessor system to the on-line distributed and hierarchical control of the laboratory-based pilot scale Travelling Load Furnace in the department.

The options available for designing a distributed computing system are enormous. A decision about the distribution of hardware and software to go along with it depends mainly on the application for which this distribution is sought in the first place. The distribution of hardware for information processing where it is needed may be limited by cost considerations whereas the distribution of software to perform the desired processing may be limited by storage capacity and software development costs. The optimum choice for both the hardware and software suggests a modular design approach for the distributed computing system. In this approach, a processor is made responsible for a particular task which is some fraction of the overall distribution of the main problem task. When a number of such processors, with their assigned tasks, are interconnected as required by the co-ordination of

individual tasks, the overall system then accounts for the distributed solution of the main problem task. Thus the main task of design and implementation of a distributed computing system is the partitioning of the main control problem into smaller subproblems and identifying the interactions between them, so that the subproblems and interactions can be programmed into the individual processors of the distributed computing system.

The modular design approach is used for the development of a multi-microprocessor system for on-line distributed and hierarchical control of the TLF. The basic processing module from which the system is configured is known as a "Hierarchical Microprocessor System Unit" (HMSU). The hardware configuration of the HMSU required to control the TLF consists of three F8 microprocessor systems, a common memory block, analogue input and digital-input-output interfaces and a bidirectional scratchpad memory interface. Each processor has its own private memory but the bulk of the memory is common to all processors. It is the task of one particular processor designated the Master processor to control access by any other processor (called a slave processor) to the common memory. Apart from this function, each individual processor acts independently, performing a dedicated control function (i.e. three term control action on different sections of the TLF) via its own Input/Output channels. The three processors operate asynchronously, all interprocessor communication being conducted through the common memory under control of the

Master processor. The Unit as a whole communicates with the outside environment, which may be another HMSU, a large host computer, or any other processing equipment. In this case, the HMSU unit is controlled by a PDP-11/10 minicomputer. The master-slave relationship of processors within the HMSU and on-line supervision of the HMSU by the PDP-11/10 minicomputer accounts for the hierarchical structure developed.

In the thesis, other structures using the HMSU as a building block are discussed in Chapter 5. Since the application undertaken is related to the control of industrial processes, Chapter 2 discusses a role of microprocessors in process control and its related instrumentation. A set of design guidelines for the use of microprocessors in process control environment are also given in this chapter. The applications which are based on a single microprocessor based system are enormous and it is impossible to enlist them. However, the applications covered by the use of multiple microprocessors in distributed computing systems are relatively few but the number of these applications have been increasing rapidly. The Science Research Council of the UK have co-ordinated a research programme in distributed computing system and its annual report outlines on current state of research on the subject. Chapter 3 reports on the study of multiple processor system, problems of designing with multi-microcomputer system and general aspects of system design with respect to distributed computing system.

In real-time large-scale complex system environment, the use of distributed computing system is highlighted by its interfacing issues. A new model of a processing element of a distributed computing system suitable for such interfacing is proposed in Chapter 4. The application of the model in two hypothetical applications is also considered. Chapter 6 describes the Travelling Load Furnace, the PID control algorithm and modifications required for the existing interfaces to the department's TLF. Chapters 7 and 8 describe the software development for the HMSU and the PDP-11/10 minicomputer and Chapter 9 discusses methods used for testing the HMSU hardware and its related software.

The full implementation of the complete HMSU system for on-line distributed and hierarchical control of the TLF was set back by the lack of proper development and debugging tools. Despite this fact, however, the research undertaken demonstrates practical problems of implementing a multi-microprocessor system such as the HMSU. As such, this thesis will provide a useful basis for evaluating and extending further research on multi-microprocessor systems and their applications.

CHAPTER 2 - MICROPROCESSORS IN PROCESS CONTROL

2.1 INTRODUCTION

The technology of applying digital computers to process control has developed rapidly since the late 1950s. A typical computer control system then comprised a centralised minicomputer with backing stores (disks) and about 8 k or 16 k of 16 bit words. Such a system would interface with the plant via 'backup' controllers which were essential safeguards against computer failures. These safeguards were needed because computer hardware was comparatively unreliable and catastrophic effects of the failures of a computer which controlled perhaps 100 to 200 loops were intolerable.

In the 1970s, this centralised configuration has given way to smaller computing units. These smaller units individually control small sections of the process and collectively form a plant-wide control system which is interconnected by a digital communication system (Brown, 1979). This modern configuration, termed as a distributed control system, has resulted directly due to the rise of microprocessors.

In this chapter, a review of the process control problem and control techniques such as supervisory control and direct digital control is made. The role of microprocessors in a distributed control system is investigated and some useful design guidelines as to the use of microprocessor-based control systems in a process control environment are also given.

2.2 *THE PROCESS CONTROL PROBLEM*

Many industrial processes have been reported to have used successful computer control systems. These include petroleum and petrochemical plants, blast furnaces, paper machines, textile mills and glass industries (Smith, 1972). Each has its unique problems but the common feature is that the energy is utilised to move and to convert raw materials into final products. Control over the final output product is achieved by computers which handle information aspects regarding the process. In all of these processes, process information is obtained or derived from process variables which are divided into four categories as illustrated in Figure 2.1.

1. Manipulated variables: These are variables such as input raw material flow rate, steam pressure in a vessel etc. whose values can be adjusted by the control system by either analogue (conventional) or digital methods.

2. Controlled variables: The measure of the performance of the plant is determined from these variables whose values are kept at some predetermined target values (set points) by the control system. Examples include production rate, product quality etc.

3. Disturbances: These are variables whose values affect the operation of the process but which are not subject to adjustment by the control system. Examples include composition of raw material, change in ambient temperature etc. Some disturbances can be measured while others cannot.

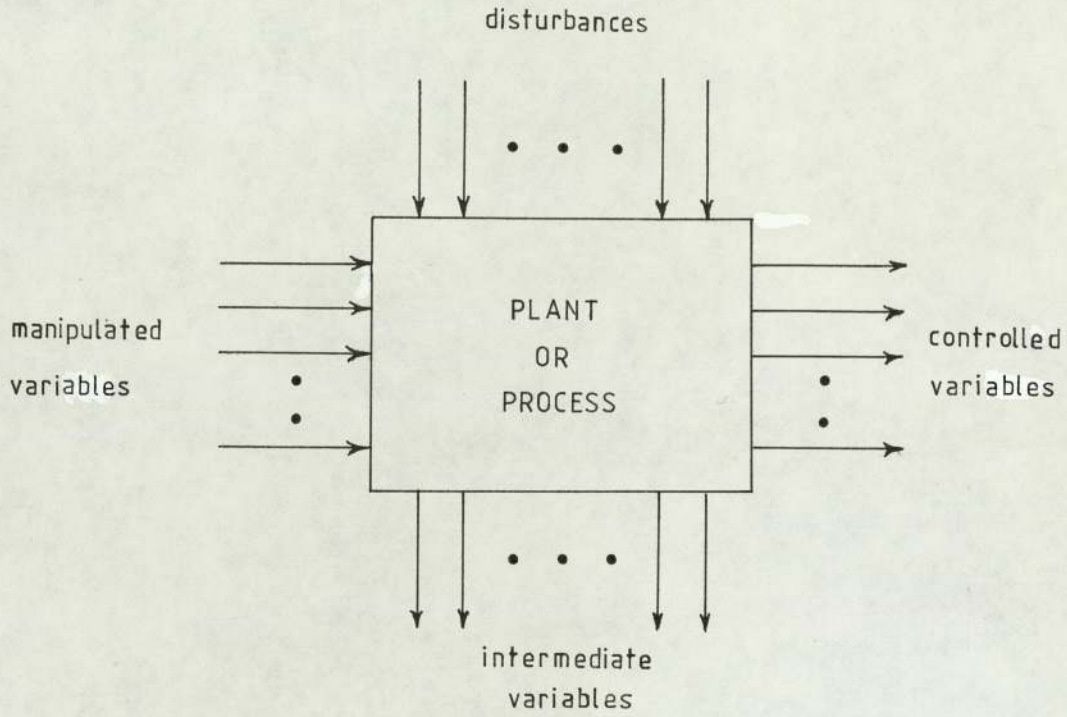


FIGURE 2.1 : General representation of process variables.

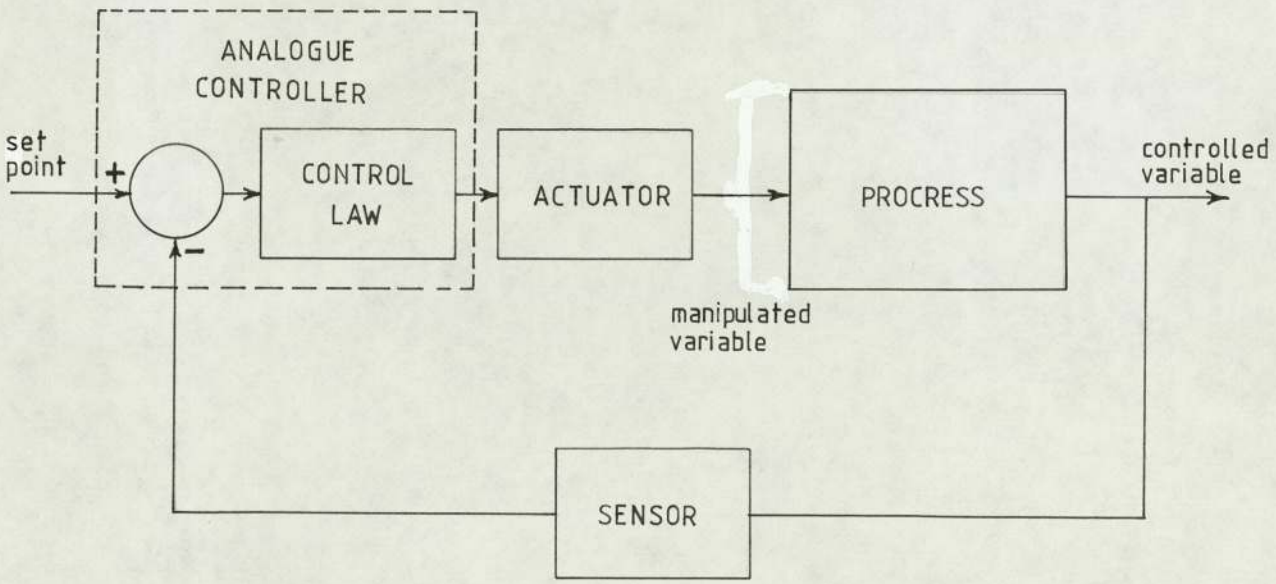


FIGURE 2.2 : Basic conventional feedback control loop

4. Intermediate variables: These appear at some intermediate point in the process. The control system can use these advantageously in determining appropriate control action. Examples include temperature of the mix, mix compositions etc.

The general control problem is to adjust the manipulated variables so as to maintain the controlled variables at their target set values in face of disturbances. The control of a typical process plant which has several variables in the above categories is no simple task. This task is further complicated if a mathematical model is required of the process characteristics. The process characteristics depend firstly on the level of plant operation (the plant is usually highly nonlinear) and, secondly, even at a constant operating level, a plant's characteristics often change with time (the plant is non-stationary).

Supplementary to the above process control problem, the most common question of primary concern is "How to use a computer to generate larger economic returns from the process?" The ability of the digital computer to acquire large quantities of data from the process, analyse it and make logical decisions based upon the results makes it most attractive for such an application.

2.3 COMPUTER CONTROL SYSTEM

The computer control of a process plant can be achieved in numerous ways. The various ways of control

depend upon the computer and the process plant configuration, control techniques and control schemes. These are summarised in Table 2.1.

In general, the control schemes are of a more theoretical nature, whereas control techniques are more practically oriented. However, the choice of control scheme depends upon the process to be controlled and this, in turn, determines the control technique to be adopted.

Before looking into digital control systems, the appreciation of the conventional approach to a process control problem is a helpful background. The basic control loop in a conventional (analog) system is the simple feedback loop illustrated in Figure 2.2. The control law generates a change in manipulated variable so as to drive the error between the set point and measured control variable to zero. This controller output is imposed upon the process by an actuator, which is an automatic positioning valve in many process control cases. The control law commonly used is the proportional-integral-derivative (PID) relationship or some simplification thereof.

In a typical plant, there may be anywhere from a few of these controllers to upwards of a hundred or more. Until the late 1950s, these controller devices were invariably pneumatic. Most of these controllers and later their counterparts, initially vacuum-tube and then solid state electronic controllers, basically suffered from inflexibility. This inflexibility imposed several burdens upon the control system designer:

COMPUTER AND PROCESS PLANT CONFIGURATIONS	CONTROL TECHNIQUES	CONTROL SCHEMES
1. Off line - manual data collection	1. Data logging	1. Sequence control
- automatic data collection		2. Regulatory control
2. In line (real time)	2. Supervisory control	- Feedback control
3. On line (real time)		- Feedforward control
- open loop mode	3. Direct digital control	- Ratio control
- closed loop mode		- Cascade control
4. Time sharing	4. Distributed control	3. Multivariable control
		4. Optimising control

TABLE 2.1 - PROCESS CONTROL TECHNIQUES AND SCHEMES

1. The control strategy must be such that it can be implemented with analog hardware.

2. Any subsequent modification to control strategy requires modifications of the analog hardware.

In the mid-1950s, the digital computers began to play a significant role in process control. This was due to the fact that any control strategy is programmable and most modifications in the strategy require simply program changes and not hardware changes.

It is not the subject matter of this Chapter to discuss the control schemes outlined in Table 2.1, because these are well documented elsewhere in textbooks (e.g. Lowe and Hidden, 1971; Smith, 1972; Savas, 1965). The following sections review some of the important features of control techniques currently practised in process control industries.

2.3.1 Data loggers

To record a large amount of process data manually is slow, tedious and inaccurate, and may involve considerable manpower expenditure. This suggests the value of automatic on-line data collection and computer control. However, as illustrated in Figure 2.3, the data logger is not directly active in the control or regulation of the process. It simply records the values of important process variables at regular intervals of time. During process modelling, carefully devised process tests generate a lot of necessary data for which a data logger is vital; however, data

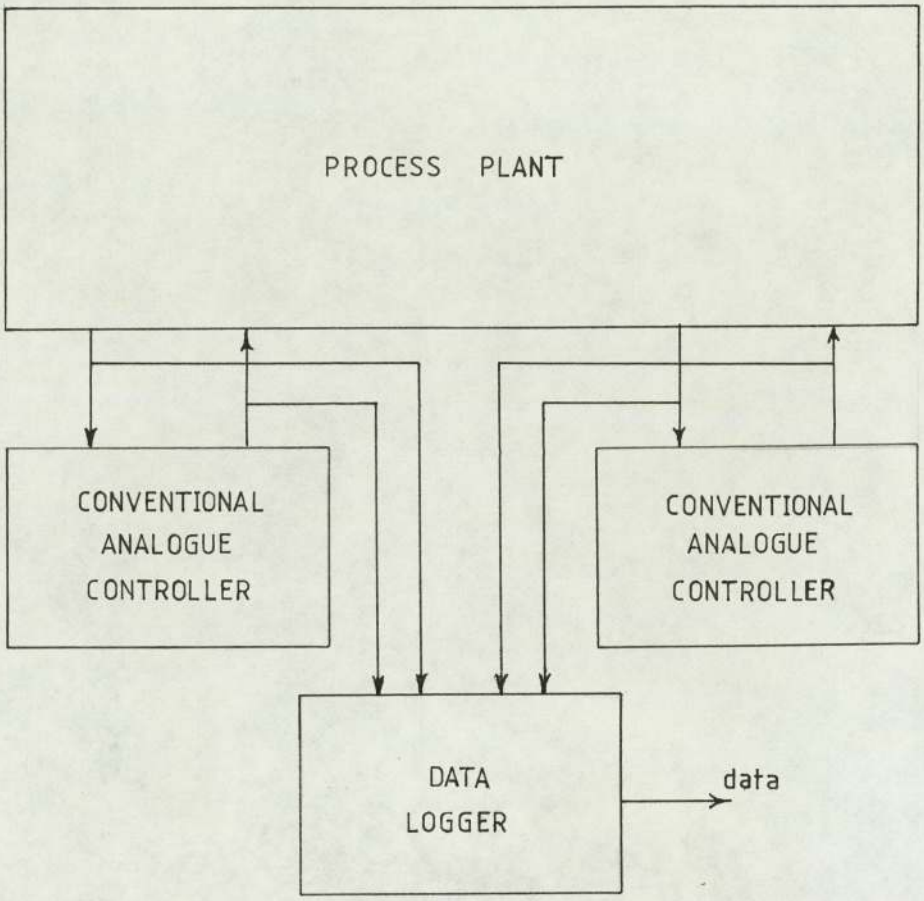


FIGURE 2.3 : Data logger

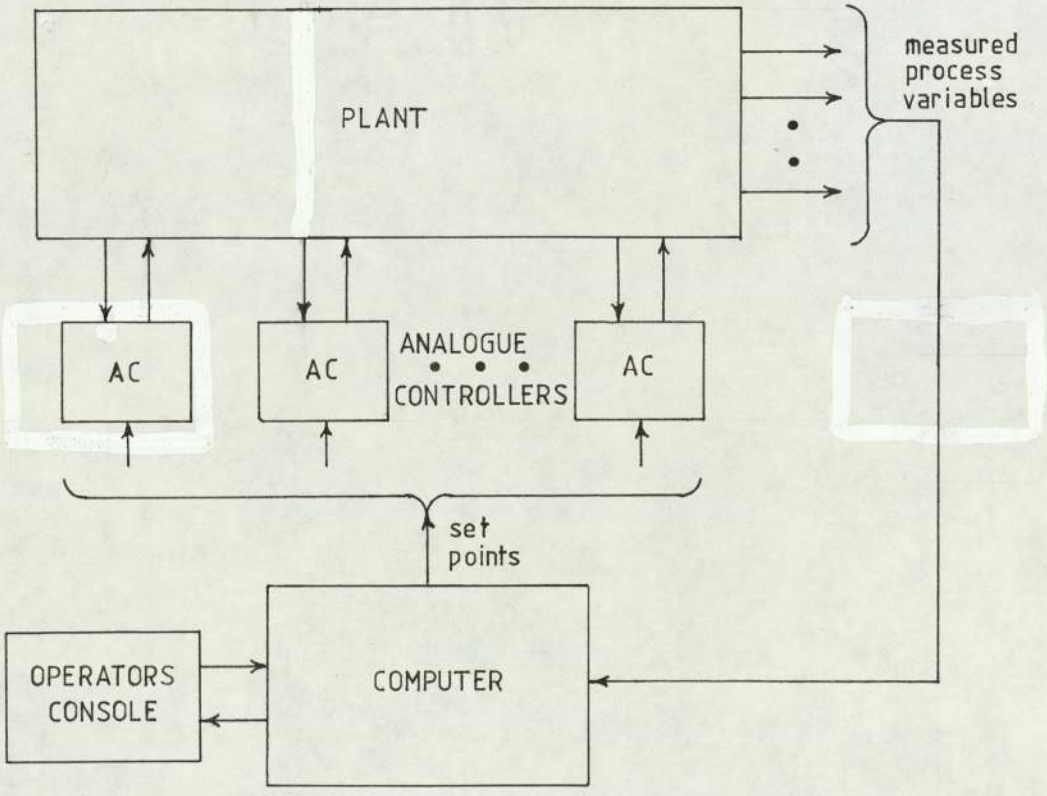


FIGURE 2.4 : Supervisory control system

logging in itself is not adequate. In a few exceptions, such as nuclear power plants, where the records must be maintained, and laboratory automation systems, data logging is of primary importance.

2.3.2 *Supervisory control*

Supervisory control systems are usually based on process models where the basic objective is to optimise the financial returns on investment. Typical input information needed for a process model might include:

1. Cost of raw materials and utilities
2. Value of products
3. Composition of raw materials and products
4. Current values of process variables
5. Constraints on the process operation (e.g. safety limitations, preventive maintenance etc.)
6. Specifications on products
7. Demands and market fluctuations for the products.

The operating strategies based upon these inputs and the process models which are generated by the computer are usually too complex to be handled by operating personnel. Thus, in many cases, the control computer simply provides the set points for the analog controllers, as illustrated in Figure 2.4. In this configuration, a single centralised computer is used which does not replace analog hardware. The backup problem is not as critical, for in case of computer failure the set points simply remain at their last setting or can be manually adjusted.

The problems of supervisory control fall mainly into a software category, and the main obstacle to the installation of supervisory system is that mathematical models of plants are seldom available beforehand. Thus, the economics of supervisory systems are based on the prospect of the system producing sufficient improvements in process operation to justify the financial investment in the computer control system.

2.3.3 *Direct Digital Control (DDC)*

The most basic form of Direct Digital Control (DDC) involves the replacement of individual hardware elements (analog controllers) wherever possible with the time shared components of a digital control computer. In the DDC technique the computer calculates the values of the manipulated variables directly from the values of the set-points, measured controlled variables and the control algorithm (e.g. discrete equivalent of conventional PID relationship). The decisions of the computer are applied directly to the process and hence the name DDC. The control arrangement is shown in Figure 2.5.

Direct digital control has been a fundamental and major step towards easy and economical application of modern control technology. It introduces the flexibility of a choice of specifying any control strategy that can be programmed in a control computer system. Addition of control loops to the existing ones, feedforward and combination systems can be used more widely when the only components which must be added to the system are transducers

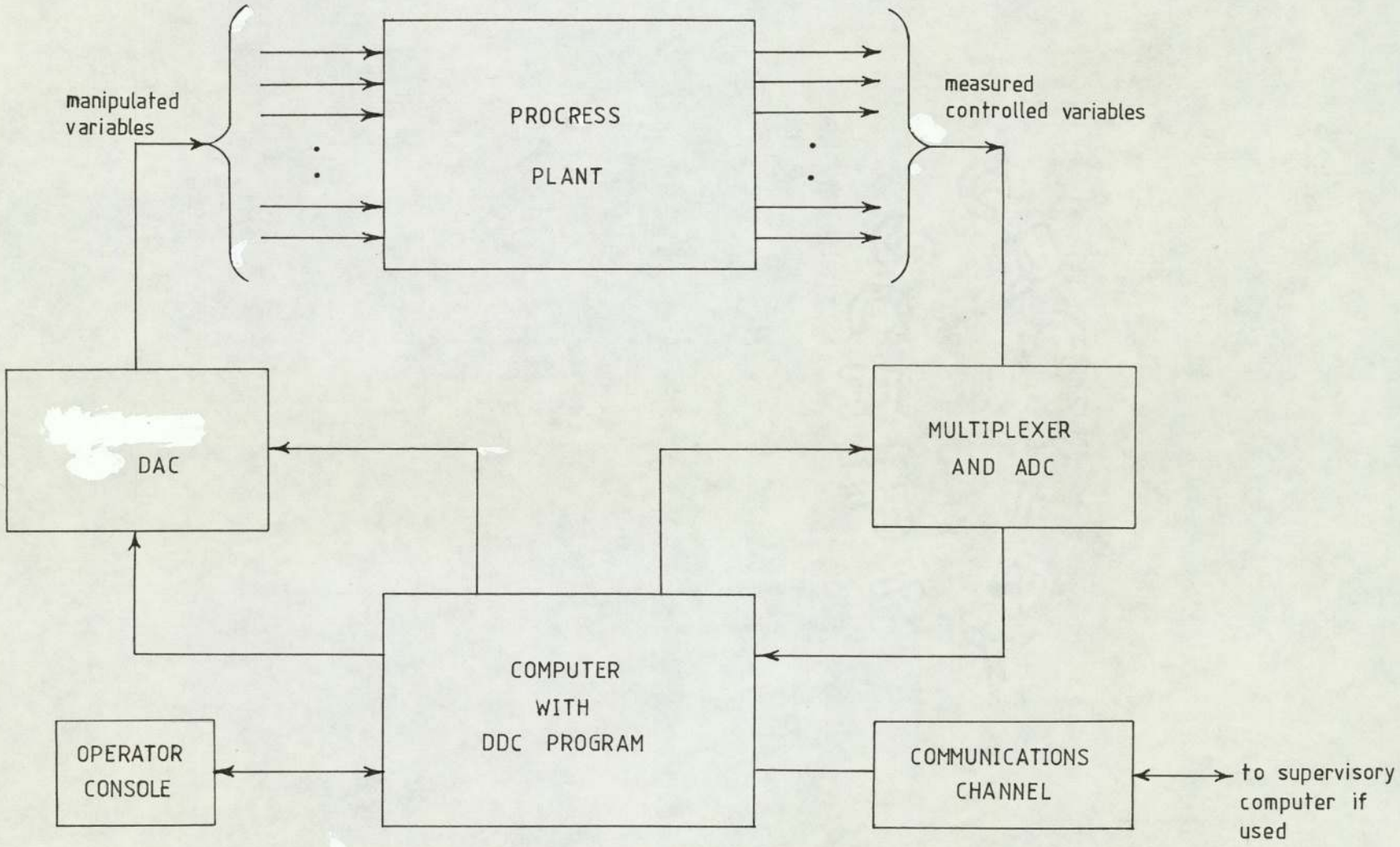


FIGURE 2.5 : Direct Digital Control

and mathematical expressions. The design of a complex process control system employing DDC allows the possibility for redesign and "customisation" after installation.

The economic justification of a process control system employing DDC technique depends upon efficient computer utilisation, computer down-time costs and the ingenuity of operating personnel to make desired program changes. In addition, if a supervisory computer is also used, then the problems associated with it would be encountered as much as with a DDC technique.

2.4 *DISTRIBUTED CONTROL SYSTEM*

A distributed control system is mainly a decentralised control system where the individual subsystem control units are distributed among the physical subsystems of the overall process. These systems have been developed, not specifically for process control application, but also for more commercial applications, such as banking, inter-company, data-base centralisation, airline reservation systems, military systems etc. In industrial computer process control, the digital process control function is distributed among the individual physical units, using microprocessors for example, which permit control tasks and physical location to be distributed in the plant; such a system has benefits of improved control, reliability, flexibility and reduced cabling costs (Roberts, 1979).

The distributed approach to control system designs can be developed to exploit modularity in both the process

control units and the structure of the communication network. These concepts can be pursued in both hardware and software, and are key features in the production of reliable and manageable systems (Holding and King, 1979). The flexibility of the resultant control system actually increases overall systems integrity. Equally important, it provides a system which can be easily implemented, adapted, extended, or replaced, either in part or as a whole. The characteristic features of such a distributed control system can be given in a tree diagram, shown in Figure 2.6. Although some of the features are categorised under software in the diagram, they do have a close relationship with some of the features of hardware. For example, the communication between the processors is very much dependent upon how the processors are structured.

2.4.1 The microprocessor role

Although not impossible, it may not be useful to develop a system that has all the features mentioned above. This is because the flexibility and low cost of the microprocessor allow it to be used in so many applications that it is difficult to put any bounds on the areas of application. Recent surveys of application to control illustrate the wide range (e.g. Aspinall, 1978; Spencer, 1976; Barker, 1978). In no way is a particular software or a hardware solution appropriate to all applications. That is why it is essential to see the role of a microprocessor with some distinctions in the type of application.

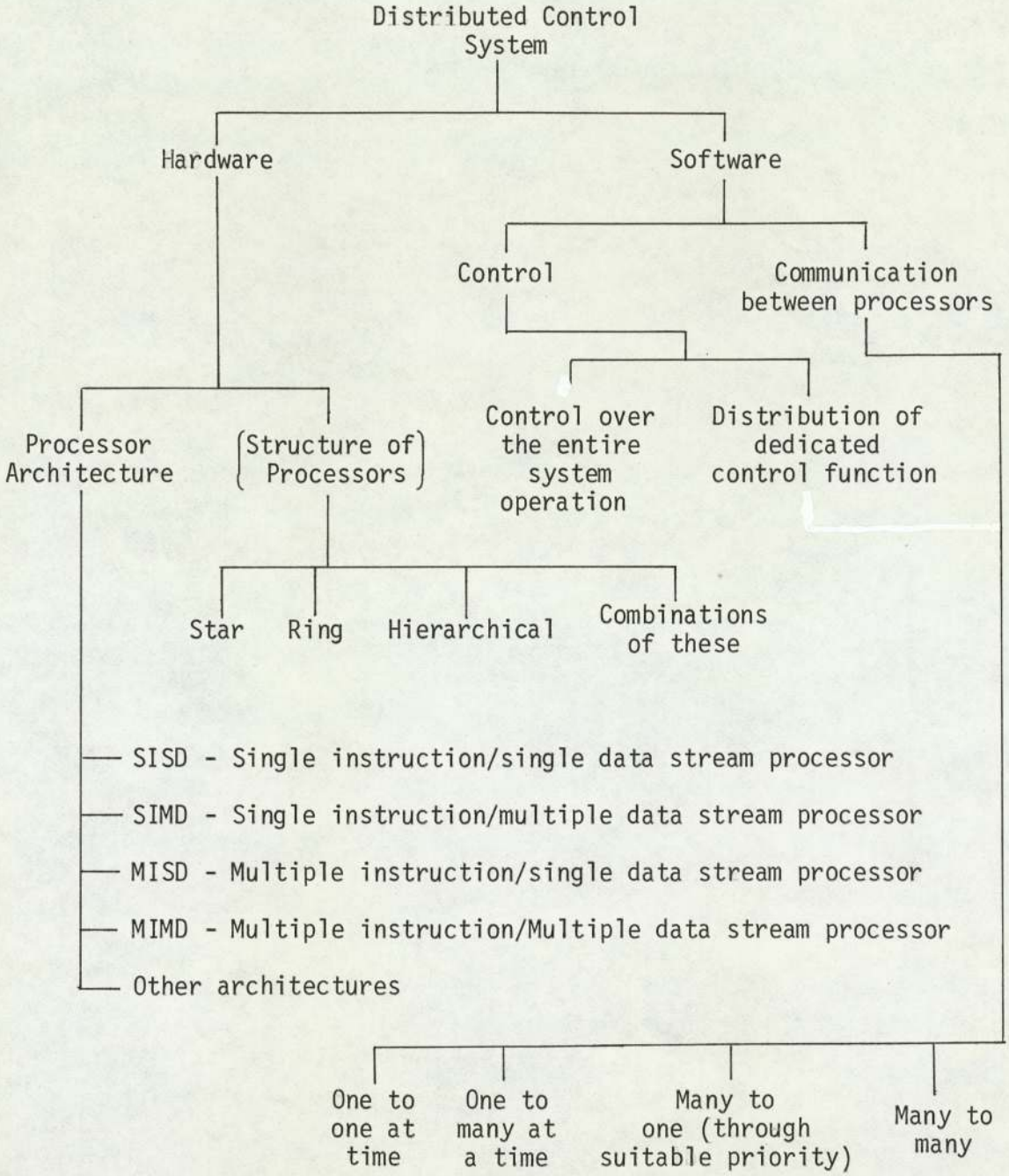


FIGURE 2.6: Distributed control system's characteristic features

The market for microprocessors in process control will be in applications with standard programs with a limited variation in functional response (Wilkie, 1979). In this type, firstly, there will be replacement for existing units, frequently with additional features. Some examples are:

1. Low-cost replacement for analog controllers
2. Intelligent alarm and acquisition systems
3. Intelligent instruments with communication capabilities.

These applications are essentially at a component level. The second type of more novel applications might be regarded at component level because they depend on the flexibility which surrounds the basic equipment, for example:

1. Sophisticated control strategies, such as self-tuning controllers
2. High reliability systems.

These applications are important to the process control designer and allow a variety of new features to be included in the system (e.g. displays). The third type of application, where microprocessors are of significant importance, is an area previously covered by minicomputers, although not always economically. These applications include:

1. Distributed control on a unit process basis
2. Sequence control
3. Mixed sequence and continuous control.

For pure sequence control, the existing dedicated PLCs (Programmable Logic Controllers) provide an economic

solution especially for very high-speed work. However, the inclusion of data logging, VDU display features or of continuous control may prove that a microprocessor-based system solution is more appropriate.

2.4.2 *The process control requirements*

Having considered the role of a microprocessor, it is worth looking into the operational requirements of process control within the background of distributed control systems. A "top down" design approach of a distributed system for overall plant control and optimisation can be considered to meet these requirements, which can be divided into a number of hierarchical levels. This is shown in Figure 2.7. The lowest level is usually concerned with the detailed control of process plant. The next level is associated with the co-ordination of plant controllers to produce a unified overall system. The highest level serves to provide plant optimisation and management information. This hierarchical operational organisation has to be implemented within the physical structure of the actual distributed system during the design process.

Very often, in process control, time critical real-time operations extend throughout all levels and their execution is essential to correct plant operation. The majority of real-time tasks, which are fundamental to the design of a distributed control system, are associated with detailed plant control. This may involve sequence or continuous control operations with auxiliary monitoring and alarm functions. The requirements are serviced in a secure

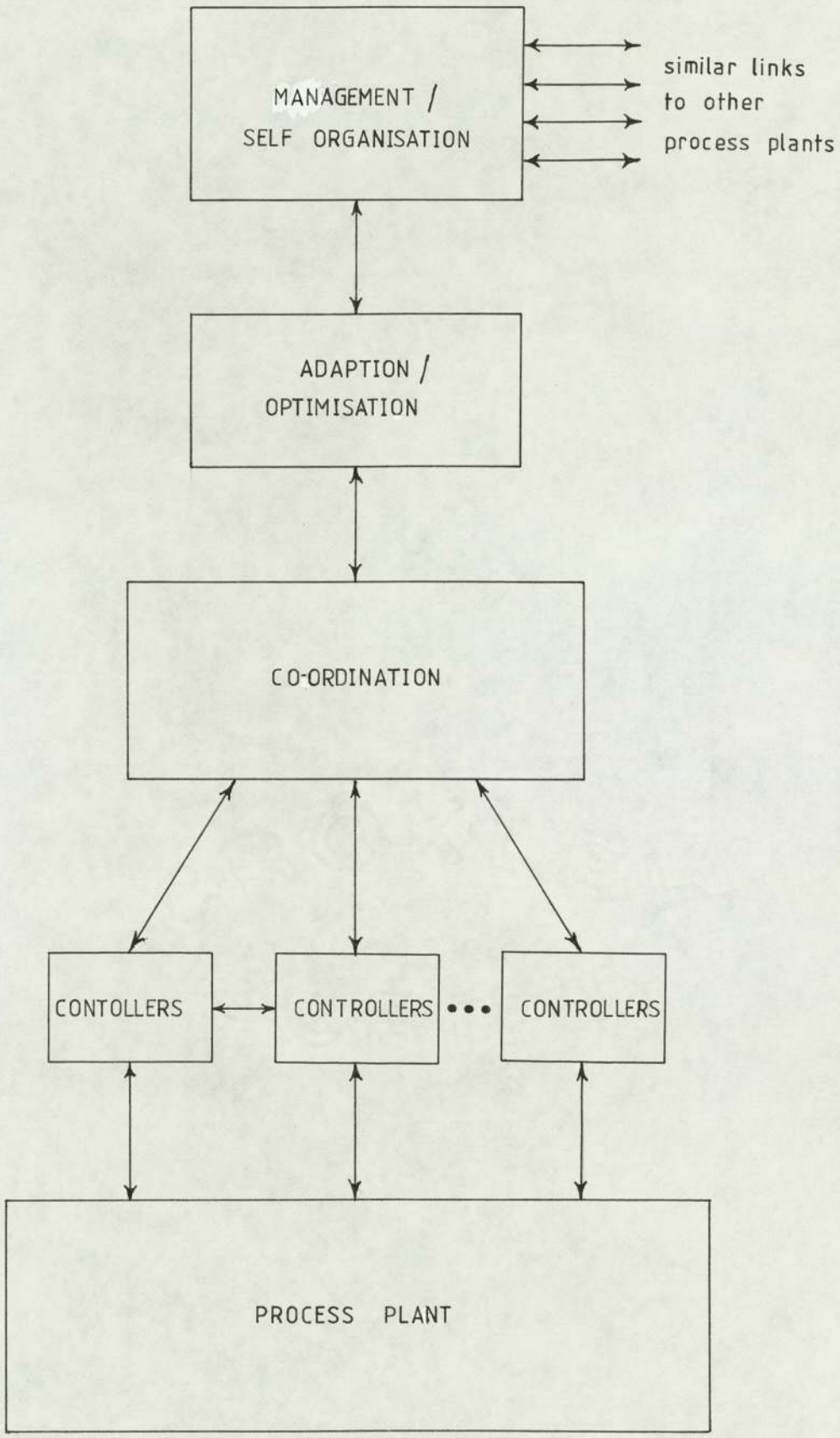


FIGURE 2.7 : Functional layers of Hierarchy

and reliable way. The co-ordination of first-line controllers also needs to be carried out in real-time and this imposes stringent requirements on communication handlers, network and the various protocols of communication. It should also be noted that the supervisory and management system may also be involved in real-time scheduling, logging and display functions, apart from their normal decision-making.

2.4.3 *Advantages of distributed control systems*

A distributed control system as described above is very similar to that of the team approach taken by co-operating humans to solve a problem too large for one individual (Bibbero, 1977). The advantages of such a distributed control system are many and summarised as follows:

1. It is more economical because of the low cost of microprocessors. This makes the first-line controllers relatively cheap so that it is economical to consider spare controllers for use in the event of failure. Should a failure occur, its effect will be limited to only a small part of the process and in many cases this part can be operated manually until the replacement controller is put into service. Equally, a failure of a higher level computer would not prevent plant operation but would merely reduce efficiency until the failure was corrected.

2. The distributed control system in its functional levels of hierarchical structure, which is very suitable

for process control application, has several advantages over the use of a large central computer. The process control can be built step by step and experiments in control in various parts of the process can be made at reasonable cost (Edgington, 1979). The advantages of step-by-step approach to building up the control hierarchy are:

A. Sophisticated control: Computer-based control leading to improved efficiency.

B. The implementation rate can be arranged to suit subprocess requirements.

C. Technological: A greater flexibility in developing technical ability of process operators because the system is implemented gradually.

D. Low risk: The effect of failure is localised to a small area.

E. Future: The system can be expanded and changed to meet changing requirements or increased understanding of the process to be controlled.

3. The distributed control system also provides a communications medium and processing facility which can be used to provide non-critical information processing, data-logging or display, using various peripherals distributed about the system. In particular, it can support facilities for the on-line editing of control programs for the various units in the system.

4. With the recent advances in the theory of hierarchical control on the one hand and multi-processor technology on the other, the optimal or near-optimal regulation of large processes in engineering, socio-economics etc. is rapidly becoming a real possibility (Billingsley and Singh, 1975). This is an enormous advantage, taking into consideration the characteristics that the distributed control system exhibits.

2.5 *DESIGN GUIDELINES FOR THE USE OF MICROPROCESSORS IN A PROCESS CONTROL ENVIRONMENT*

Microprocessors are relatively new devices; their potential needs to be well understood before being applied to any desired application. A variety of questions should be answered in the design process of a microprocessor-based system. For a process control application, the following set of design guidelines have been given for microprocessor based systems (Weissberger, 1975).

1. The nature of application: It may be

   (A) A programmable controller
   (B) A dedicated processor
   (C) An element in a distributed control system.

2. (A) What is the number of

      (a) functional tasks involved?
      (b) input/output points?
      (c) points to be controlled?
      (d) loops to be controlled?

(B) What is the processing load?

(C) Is real-time response required?

3. A decision as to the functional task subdivision and input/output signals assignments for processing elements is needed. The data load and throughput rate for processing element also needs to be determined.

4. Microprocessor selection: This can be very critical and depends on several factors. These are:

(a) Availability
(b) Supplier reputation
(c) Software support
(d) Instruction set, word length
(e) Speed of operation
(f) Architecture - interrupt capability, registers etc.
(g) Second source
(h) Memory capability
(i) Package count
(j) Number of power rails
(k) Power consumption
(l) Development system.

Also, in the selection processes the software design needs careful attention; for example, programming flexibility, word size (data/instruction), address capacity, addressing modes (indexed, indirect, relative, direct etc.), instruction set (repertoire and speed), register compliment

(arithmetic, index, status, accumulators, general purpose) etc.

5. Environmental considerations: These include

(a) Industrial noise, temperature, electrical noise

(b) Distance between process variables

(c) Power dissipation, consumption and cooling

(d) Input/output interfacing

(e) Future expansion, space etc.

6. Interfacing: This is a very important stage in the design process and this includes:

(a) Transducers

(b) Amplifiers

(c) A/D converters

(d) Multiplexers, demultiplexers

(e) D/A converters

(f) External event counters for real-time application

(g) DMA facilities

(h) Line drivers, line receivers, moderns, UARTs

(i) Cabling, twisted pairs, coaxials, ribbon, optic fibres etc.

(j) Displays

(k) Consoles, telephone links etc.

(l) Earth loops.

7. Distributing: As described earlier, distributing can produce a cost-effective solution. This may include:

(a) Distribution of microprocessor/controllers along the peripherals of the plant floor with a centralised minicomputer

(b) Distribution of individual power supply.

(c) Distribution of functional task by partitioning and software modularity.

A lot of cost savings can be made if the above guidelines are followed in the development of microprocessor-based systems for process control application.

2.6 CONCLUSIONS

In process control, the computer has become one of the primary instruments for control. The advent of large-scale integrated circuits and microprocessors has radically changed the capability and applicability of distributed computer control systems. These systems can be applied to a wide range of applications and trial installations have been established in a number of industries (IEE Conference publication, 1977). The modularity and flexibility of these systems make them more reliable and manageable than centralised systems. In many situations, they present a more attractive and economic solution to the control problem.

The review of the control techniques presented in this chapter suggests how the changes have taken place over the

last two decades. A lot of further research, however, is needed and the scope is enormous in areas such as distributed processing, architecture, operational attributes, resource management etc. (SRC Annual Report, 1977). It has been the experience of several years that the theory is always ahead of its practical implementation. This is also true in process control and the distributed control systems attempt to bridge such a gap.

Another area which is of interest is that of communication between processors and the issues of the development of a standard for communication between the intelligent subsystems of a process control system (Lee, 1976). The development of higher-level languages for distributed control systems and the development of different architectures for multiprocessors have been at the open end of the research activities in the universities and industrial research centres. The concept of a transputer (Aspinall, 1978), for example, falls into the category of such architectural developments. In general, the pressures for change in computer system architecture are: (1) language and programming based, (2) applications and systems based, (3) reliability and technology based, or combinations of these drives for change (Elliott, 1978).

CHAPTER 3 - SYSTEM DESIGN

3.1 INTRODUCTION

The process of system design is essentially a process of translating the problem specification in a high-level natural language into the problem solution in a lower-level language notation. The human brain is unable to deal completely with more than a certain amount of information at any one time (Miller, 1956). Therefore, the only natural way in which a large-scale task may be comprehended and solved is by splitting it up into a set of smaller, comprehensive subtasks in a logical manner. The translation of the problem specification, of a large-scale task into the problem solution is usually too complex to be performed in one stage (Dowsing, 1978). As such, it is normally broken down into a number of smaller translation steps, each step lowering the level of the language used for the specification and the complexity of the system needed to understand it.

It is true, in general, that design is an art and the object of art is no simple truth but complex beauty and so any design usually involves making personal choices and trade-offs depending upon cost constraints and time limitations. So far as designing with computers or microcomputers is concerned, the lower-level language notation typically ranges in complexity between a high-level programming language and a hardware logic design language which can readily be used by software and hardware

implementation systems respectively. Furthermore, the advent of microprocessors has opened up a new design era of multi-microprocessors or multi-microcomputers in which the designer can think in terms of parallelism or concurrent performing of smaller subtasks. A design solution resulting from the use of multi-microprocessors/microcomputers may perhaps surpass the human brain capability of dealing with only a limited amount of information at any one time!

In this chapter, the different phases of the system design process are examined and the problems of designing with microprocessors are outlined. An attempt to classify a multiple processor system is made and a review of such a system is also given. Finally, the design issues relevant to a multi-microcomputer or distributed system are discussed.

3.2 *GENERAL ASPECTS OF SYSTEM DESIGN*

The process of design in general starts with an effort to answer a simple question: "What is it that we want to achieve?" The answer usually attempts to establish the goals or objectives about a system to be designed. A defined set of goals or objectives results from a feasibility study of the intended system. When such a system is envisaged to be feasible under given cost constraints and time limitations, the process of system design continues with the following subtasks:

1. Problem specification: This first important step involves an unambiguous, rigorous and detailed specific-

ation of the problem. The specification must be detailed enough for a correct solution to be produced but not over-specified with irrelevant information.

2. Logical design of the problem solution: The next task is to decide on the method, the algorithm and alternatives for solving the problem. A designer has to harness his skills to discover which is the "best" solution for the particular problem in hand. The next logical task is to produce a formal definition of the chosen problem solution which may be implemented with the available implementation tools, either hardware or software or a mixture of the two. This task involves a decomposition of the high-level problem description into a lower-level description containing details which are more implementation dependent. This forms a basis for the implementation subtask.

3. Implementation: A task of the implementation phase is to map the logical design onto the implementation system. This phase is typically constrained heavily by costs, time and available resources. An experienced designer may not have to pay any penalty for the constraints heavily imposed on the implementation phase if these are well anticipated and estimated in the feasibility study of the system design.

4. Testing: The output of the implementation stage takes the shape of the intended system but the behaviour of such a system needs to be tested in this phase. This phase requires testing tools and skills. Any errors, which have

occurred in the previous phases of design are revealed in such a testing stage. Generally, it is best to mingle implementation and testing in order to detect these errors, because sooner the error is detected the easier it is to correct and less effort is extended.

5. Optimisation: Optimisation stage is not strictly a part of the design phase but is an important technique for modifying the design so that the resource requirements of the problem solution may be met. This phase also avoids the necessity for complete redesign of the system using a different approach or algorithm.

The design process is an iterative procedure based around the subtasks outlined above with the specification, testing and, if necessary, optimisation taking place at each stage of the problem solution. The complexity and likelihood of errors is reduced if the designer ensures to take smaller steps at any stage of the system design. Another important aspect of any system design is the quality of its documentation (Fitzgerald and Fitzgerald, 1973). A full documentation of a system design should provide the solution to the problem, the reasons why the particular design decisions were taken, the underlying strategies and their consequences on the rest of the design.

3.2.1 Designing with microprocessors

Although applicable for any systems, the above general aspects of system design can be followed for systems

incorporating microprocessors as well. However, there are some important issues, described here, which make designing with microprocessors a special case.

Microprocessors are a new technology and this technology is revolutionising the way in which new electronics based products are designed. It is creating a whole new set of problems for designers. Part of this new design philosophy results from the fact that in a microprocessor, system functions are stored in memory instead of wired into discrete logic devices, and the system designer has the possibility of making modifications simply by changing the program stored in memory instead of redesigning the hardware. Hence the software now becomes as important a part of the design process as the hardware. The basic design task in a microprocessor-based system can be broken down into three areas: software, hardware and software/hardware integration. This is shown in Figure 3.1.

3.2.1.1 Software: The first step is to design the program, a task which requires knowledge of the design objectives and the microprocessor characteristics. The design guidelines mentioned in Chapter 2 are very useful for this purpose. For many practical programs, the use of an assembler is necessary; this means coding the flowchart into a source program and from this assembling into the object code which will run on the actual microprocessor. There are a number of ways of achieving an object code from a source program. These are outlined in Figure 3.2.

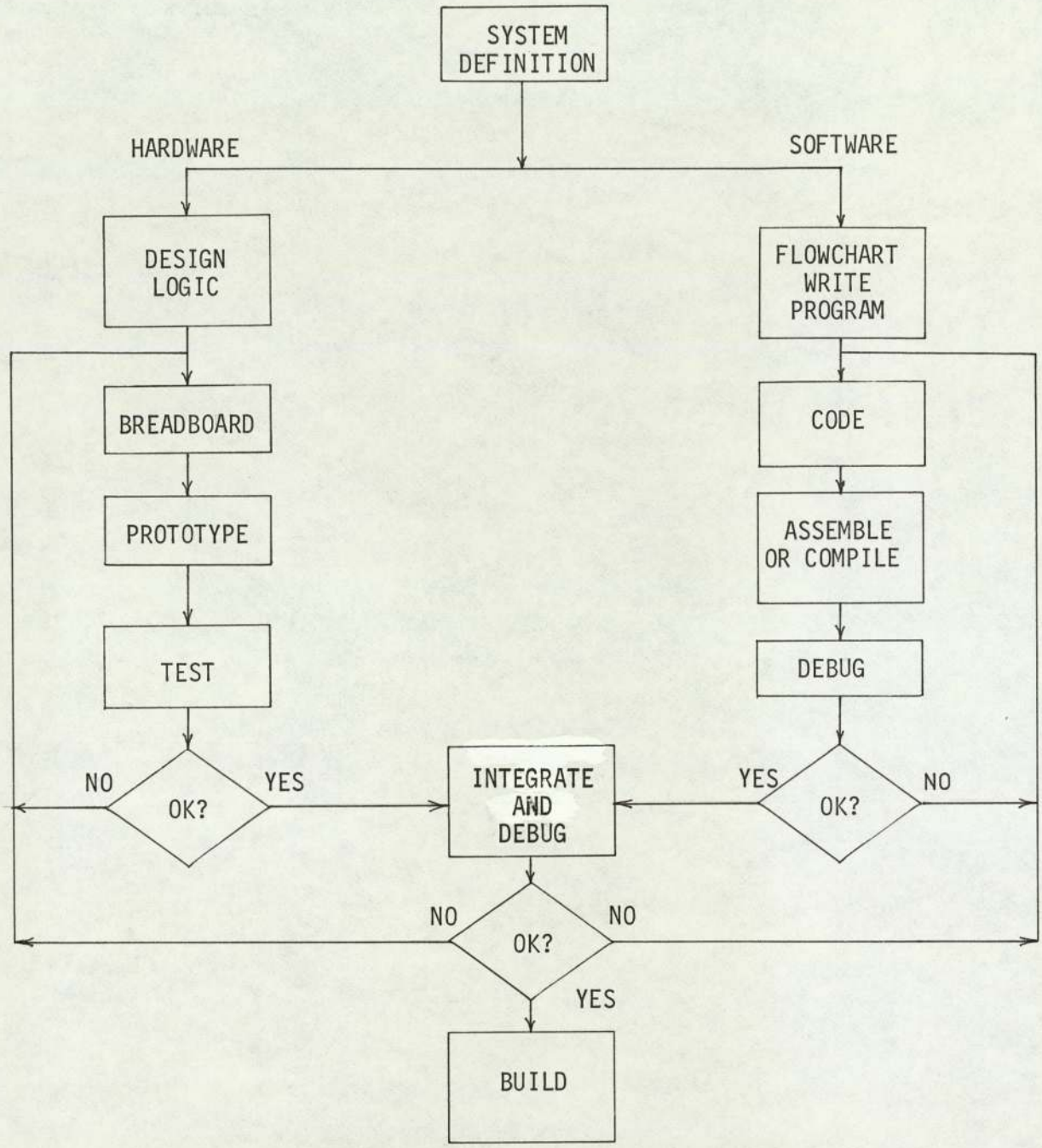


FIGURE 3.1: The basic design task in a microprocessor-based system

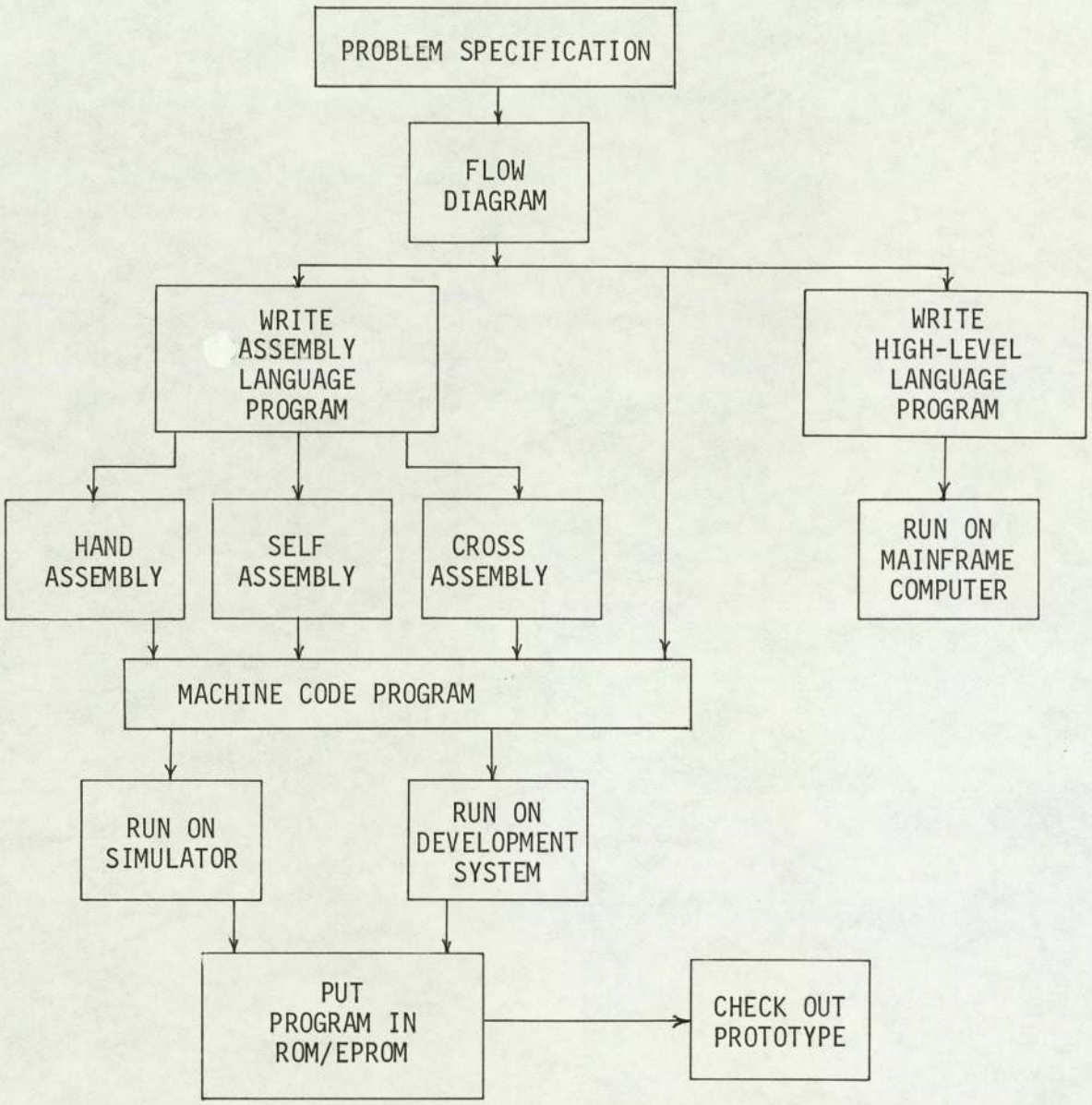


FIGURE 3.2: A general program development procedure

Coding a source program requires the use of a text editor with the ability to enter text, modify, insert and delete - plus a set of utilities for creating, loading and manipulating text files and outputting to a printer or terminal. For assembling into object code, the assembler needs to be speedy and it must produce relocatable code so that programs can be written in modules which are linked together after assembly. A high-level language can be used instead of an assembler, and the choice is very much dependent on the application. However, with many micro-processor systems, high-level language solution is not fully available. In general, the high-level language approach is best for quick design completion, low-volume products and where data manipulation is important. The assembler is better for high-volume products or real-time control applications where speed is important.

Debugging of software consists of removing all program errors. A certain amount of debugging can be done on an emulator, but since the final operation is dependent on the actual hardware, most debugging has to be done during the critical software/hardware integration phase.

3.2.1.2 Hardware: The first step is logic design which, like program design, can be done using information on devices (data sheets) and a knowledge of the design objectives. Breadboarding of the circuit modules is carried out to obtain a prototype. This is a very typical procedure followed by most electronic design engineers and the tools involved are typically an oscilloscope, digital

voltmeter and, more frequently nowadays, a logic analyser. Using such instruments, major hardware faults can be detected but thorough debugging and testing is possible only during software/hardware integration.

Another important point of consideration, while designing hardware, is that of deciding the level at which to start designing with microprocessors. There are three basic levels of supply of microprocessor hardware:

1. Chip level: Starting from chip level can be useful if large production is anticipated, where the design costs are spread over many units. However, it does require a large outlay in time and money to get started.

2. Board level: Standard functions available on ready-made boards is a very convenient way of implementing a system quickly and at reasonable cost, provided the restrictions and limitations of the particular board are understood and allowed for.

3. System level: Standard systems can be bought from a number of suppliers. These are self-contained units or microcomputers.

The choice of the level of hardware depends on the application and such factors as flexibility, expandability and maintainability.

3.2.1.3 Software/Hardware Integration: This is the critical stage in completing any successful working design. It is impossible to tell whether the software is working

correctly without using the hardware or vice versa. Therefore, the task of debugging the original design becomes a dynamic, interactive process; for example, one may overcome a hardware problem by modifying the software or vice versa. Tools such as in-circuit-emulators, logic analysers are very useful at this design phase.

3.3 THE IMPACT OF MICROPROCESSORS ON USERS

Having seen some of the implications of designing with microprocessors, it is worth noticeing the impact of microprocessors on users. Microprocessors, as with main-frame computers, have same attributes of association with peripheral devices, the development environment and the user's environment. Mainframe computers have been the case of bedrock investments for a long time and still will be for some time to come but now, it is the user of microprocessors who has to make such huge investments in his own environment. Furthermore, the user is allured by ever-increasing cheapness of available microprocessors and new announcements of more and more powerful microprocessor architectures and their potential. The peripheral devices for use with microprocessors and microcomputers are becoming a medium-life phenomenon whereas the development environment for microprocessors themselves is becoming a long-life one. This is all depicted in Table 3.1.

Carter (1978) has well reported a number of problems of using microprocessors in areas such as technical, manpower, commercial and sales and marketing. Although these are documented from the viewpoint of a company producing

USER'S ENVIRONMENT	TRANSIENT	PROCESSORS MICROPROCESSORS
DEVELOPMENT ENVIRONMENT	MEDIUM LIFE	PERIPHERALS
PERIPHERALS	LONG LIFE	DEVELOPMENT ENVIRONMENT
PROCESSORS MAINFRAME COMPUTERS	BEDROCK INVESTMENT	USER'S ENVIRONMENT

TABLE 3.1: Impact of microprocessors on user environment

its first microprocessor-based product, the technical and manpower areas of problems are similar for any microprocessor development project. Another problem area of important consideration is that of the cost and the benefits of a microprocessor-based project. Microprocessor technology is changing rapidly and costs are also changing quickly. It is important to repeat cost/benefit analyses at regular intervals, especially if the project is a longterm one. In the total costs of a microprocessor project, the basic cost of the microprocessor chip is indeed the tip of an iceberg as shown in Figure 3.3.

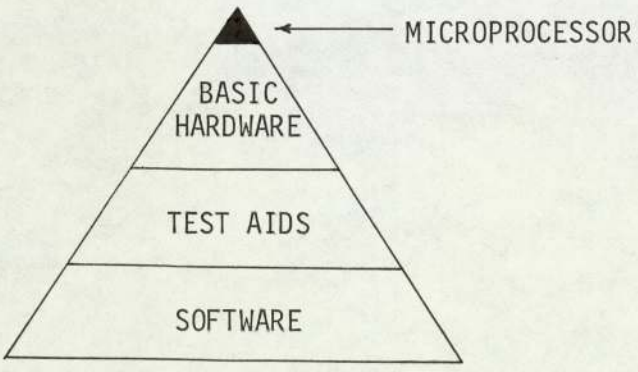


FIGURE 3.3: Total system costs

Working with microprocessors is initially expensive, although these costs are not repeated for successive projects unless the choice of processor is changed.

Different considerations outlined in this section are relevant to system design and should be considered as a part of the design process while designing with microprocessors.

3.4 *MULTIPLE PROCESSOR SYSTEM*

The concept of a multiple processor system is not new and has been used in very large EDP (Electronic Data Processing) systems for several years. But the use of microprocessors in such systems is rather recent. There are two basic reasons why a multiple processor system should be envisaged using microprocessors. Firstly, the microprocessors are very cheap and secondly, since they are constrained in computing power by the physical limitations of the chip capability, an extension of this power through the use of a multiple processor system makes it viable to produce large as well as small EDP systems.

3.4.1 *Review of multiple processor system*

A review of the literature (e.g. Searle and Freberg, 1975; Weissberger, 1977; Anderson and Jenson, 1975; Flynn, 1972; Thurber and Wald, 1975) reveals a considerable confusion in the classification of multiple processor (computer) system. The same name is given to different computer organisations and different names are assigned to

the same computer organisations. For example, Joseph (1976) has reported some twenty-four different ways of referring to distributed processing which emphasises a particular architectural difference. He also further admits considerable confusion that exists as to the meaning of the term "distributed processing". However, Flynn (1972) has suggested a basic classification scheme which describes the method of operation based on the number of instruction streams and data streams in the system. A brief mention of this was included in the characteristic features of distributed control system (Fig. 2.6, Chapter 2), but a more elaborate tree diagram, shown in Figure 3.4, outlines some other features associated with a multiple processor system.

Since different system terms are used today, it is important to give some definitions. A good review can be found in Searle and Ferberg (1975).

A multiple processor system contains more than one processor. Each processor may be a microprocessor or a microcomputer executing a specific task. A microcomputer is a microprocessor system with its own memory and peripherals. Software considerations allow one to discern two kinds of multiple processor system:

1. A distributed system, also called a multi-microcomputer system or distributed intelligence microcomputer system (DIMS) (Russo, 1977), in which each microcomputer performs a dedicated function as part of a single partitioned system. This static allocation of tasks allows

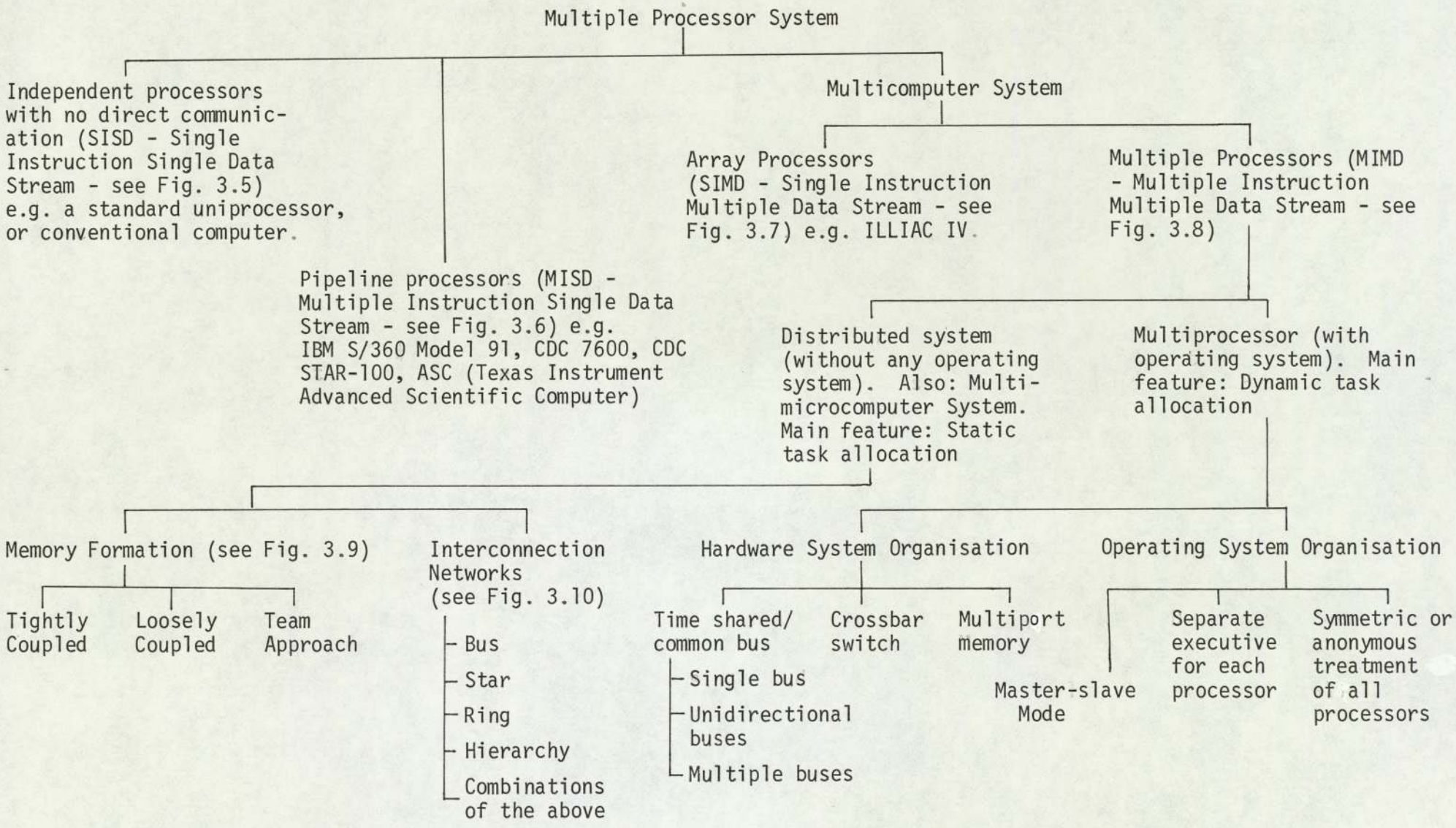


FIGURE 3.4: Features of Multiple Processor Systems

the partition of software and is an attractive solution for microprocessors. In such a multi-microcomputer system, there is no integrated operating system as such, but there exists some kind of communication protocol, either implemented in software or hardware or the combination of the two, in order to facilitate communication between a number of processors. In a distributed system, individual microcomputers may be locally distributed or there can be geographical distribution of microcomputers depending upon application.

2. A multiprocessor system implies a single integrated operating system which is capable of dynamic allocation of system tasks. Software is much more complex for such a system than for a distributed system, but allows balanced processing loads in real time and fail-soft capability.

The Figure 3.4 shows that a distributed system and a multiprocessor system are in the same group of multi-computer systems which are characterised by multiple instruction stream operating on multiple streams of data (e.g. see Fig. 3.8). Apart from this, two more categories of multiple processor system need defining. These are as follows:

1. An array processor is one in which multiple streams of data are treated simultaneously by processing elements in response to signals from a control unit, decoding a single instruction stream. The only

qualification that distinguishes an array processor from a multiprocessor is that the control of the number of processing elements is always associated with one control unit (e.g. see Fig. 3.7).

One example of an array processor is the ILLIAC IV system (Feierbach and Stevenson, 1979). The ILLIAC IV has a single control unit (CU) to direct the activities of 64 processing elements; these processing elements execute the same instruction in parallel but on different data fetched from their local memories. Information is exchanged among the processors through a routing network; processes are logically arranged in a ring but the implementation allows routes of a distance of eight processors to take the same time as routes of a distance of one processor. All processors are required for array operation; programs are written and compiled for execution on 64 processors. When a processor fails, the entire machine is unavailable until it is fixed. There is no run-time error detection; failures are detected by periodic confidence tests.

The ILLIAC IV architecture is also partially reconfigurable via software so that each 64-bit processing element could be partitioned into either two 32-bit or eight 8-bit processors. The major application areas for this type of array processors are the many large-scale scientific problems in mathematics, numerical analysis and engineering in which the nature of data to be processed is in matrix form.

2. A pipeline processor can be regarded as a form of functional partitioning of CPU microfunctions i.e. a multiple instruction stream operating on a single data stream fetched from memory (e.g. see Fig. 3.6).

The CDC STAR-100 system (named from the STring/ARray data it is designed to process) is one of the best known pipelined systems (Spencer, 1976). The CDC STAR has a computer network consisting of nine computers which execute the operating system, handle the files and deal with the input/output equipment, and the very large central computer which handles the processing on the string and array data.

Multiprocessor systems, array processor systems and pipeline processor systems have been well discussed in the literature (e.g. Searle and Ferberg, 1975; Thurber and Wald, 1975; Feierbach and Stevenson, 1979). Most of these systems have a clearly established modular nature in their architecture. A computer architecture based on LSI modules allows for a simple software controlled reconfiguration of interconnections among modules. For example, processor modules may be switched among several main memory modules, I/O modules etc. This concept of reconfiguration of architecture by software is not new; the LSI technology, however, has enhanced it. The ILLIAC IV (Feierbach and Stevenson, 1979), C.mmp (Wulf and Bell, 1972), Cm* (Swan, Fuller and Siewiorek, 1977) are some of the examples of multicomputer system with capabilities of reconfiguration of architecture.

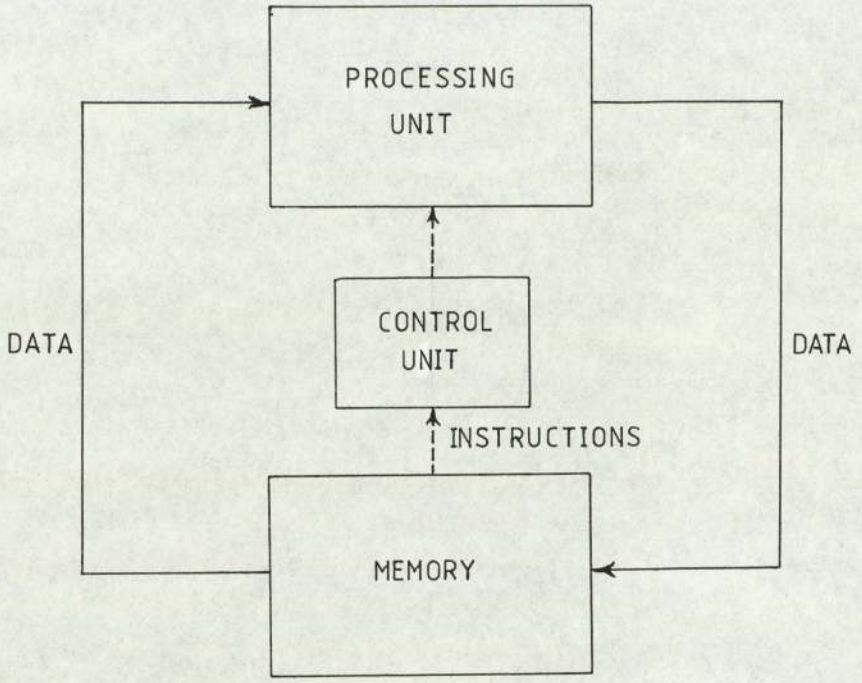


FIGURE 3.5 : SISD PROCESSOR

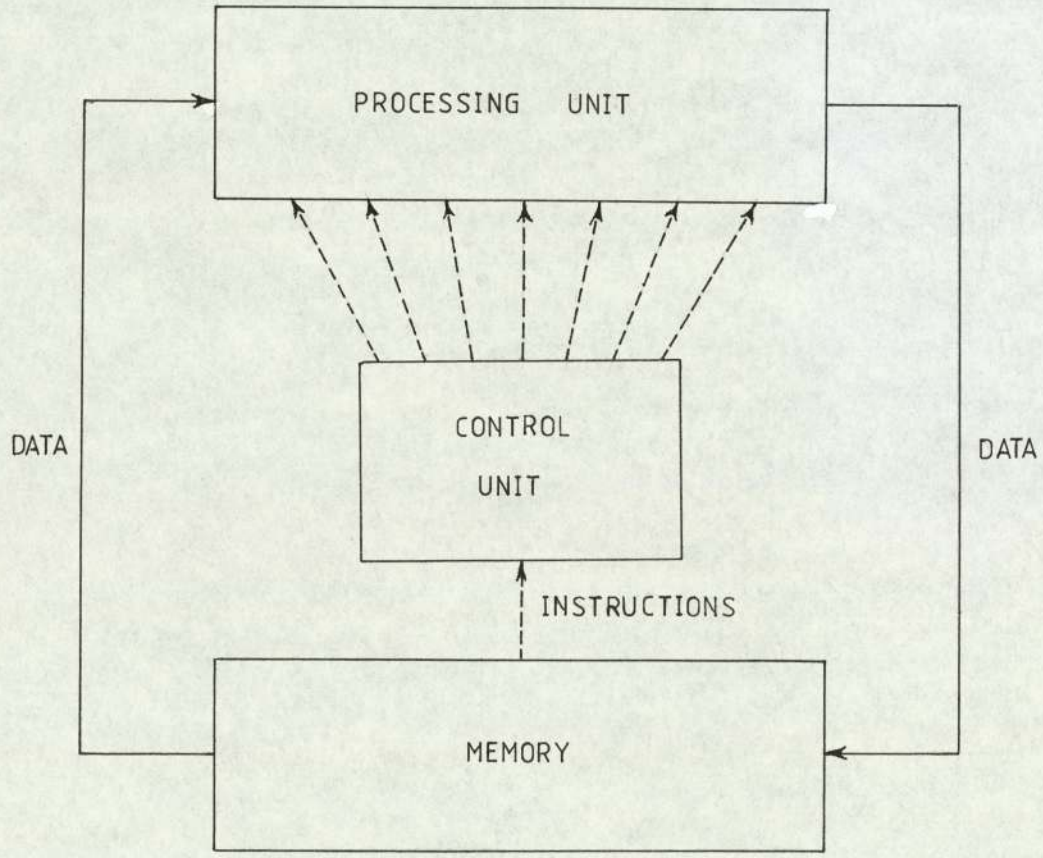


FIGURE 3.6 : MISD PROCESSOR

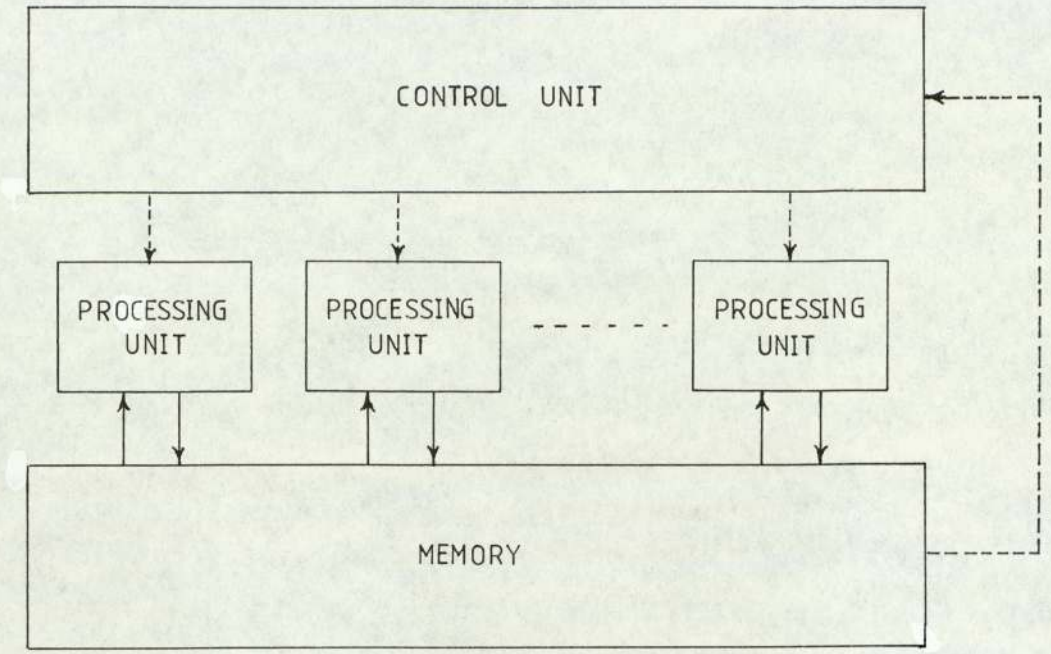


FIGURE 3.7 : SIMD PROCESSOR

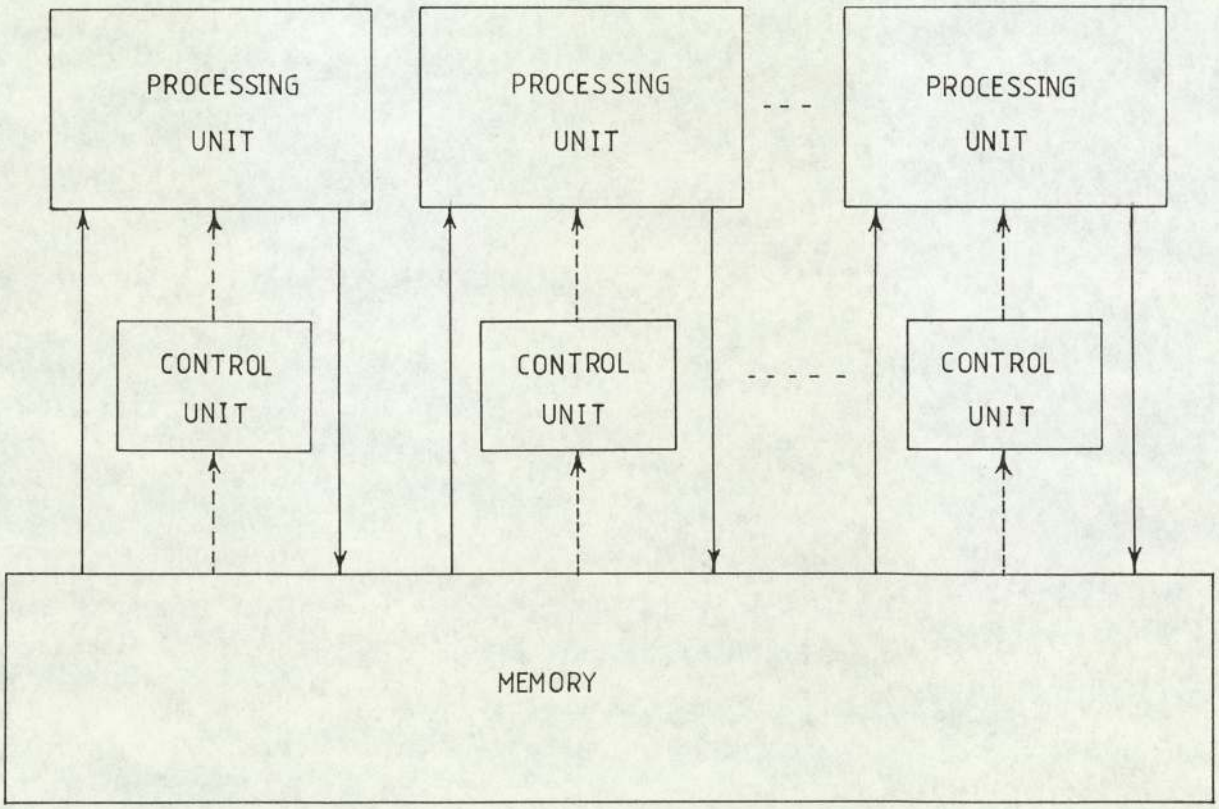
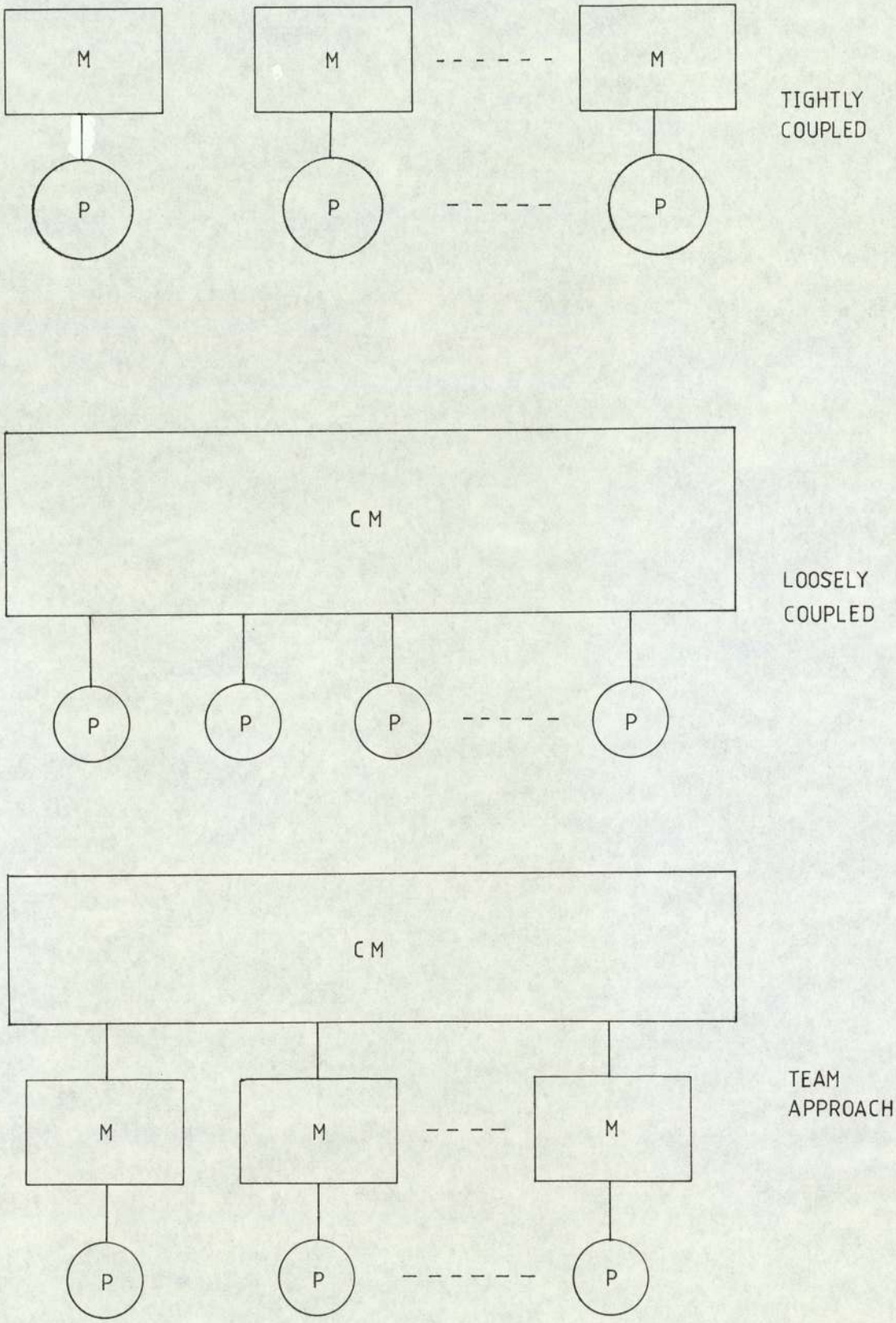


FIGURE 3.8 : MIMD PROCESSOR



NOTE:- CM=COMMON MEMORY , M=MOMERY , P=PROCESSOR.

FIGURE 3.9 : MEMORY FORMATIONS

A class of new multicomputer system which cannot be placed under the classification of Figure 3.4 has been envisaged by Kartashev and Kartashev (1978). This is a new LSI multicomputer system with dynamic architecture which allows one to reconfigure via software and in microseconds all available hardware resources (widths of processors, memories and I/O units), each time forming in the system new computers with different sizes. Based upon given cost criteria, this system with dynamic architecture has been comparatively evaluated for synchronous, asynchronous and modular control organisations.

It is not the object of this chapter to discuss the details of multiprocessor systems and their complex operating systems because these research subjects are well treated elsewhere in the literature (e.g. White, 1976) and basically there are many software problems associated with operating system design and high-level programming language design for such systems. As such, the following section concentrates on the design issues of multi-microprocessor/ microcomputer systems or distributed systems.

3.4.2 Problems of designing with multi-microcomputer system

As outlined in Figure 3.4, one of the main features of a multi-microcomputer system is a lack of an operating system and the static nature of allocation of tasks among a number of processors. This means that a system designer has to use low-cost microprocessors to design a multi-microprocessor system which is oriented towards an application such that the application problem is carefully subdivided

for parallel processing or concurrent execution. The main advantage of such a subdivided application problem is modular software development. However, the design of such a distributed system poses several interesting problems and these are discussed subsequently.

3.4.2.1 System architecture: The system architecture differs from a processor architecture and is usually influenced by application requirements. The following factors govern the system architecture:

1. Control and management of resources: The resources, whether hardware or software, which are distributed among various processing elements, should be efficiently used. If a resource is made common or is shared, then due consideration must be given to resolve conflicts for its use.

2. Load balancing and reliability: The nature of the application determines as to how the processing could be balanced among various processing elements. This requirement may arise due to failure of any processing element. The reliability specification determines whether the system component failure is tolerable and, if so, how it degrades the overall system performance.

3.4.2.2 Communication and control: This is an essential feature of a distributed system and the quality of performance of the entire system depends on communication and control of information and the complexity of protocol used for it. The factors to be considered are:

1. Interconnection of processing elements (e.g. see Fig. 3.10). The choice of interconnection depends upon the nature of application, flexibility, reliability, cost and complexity of control. A combination of various networks in Figure 3.10 is also possible.

2. Inter-process communication: The flow of control and data information between various processes processed in processing elements can be achieved by numerous communication protocols. These may be based on the following:

A. Serial communication using UARTs, Modems etc.

B. Parallel communication:

(a) port to port transfer using polling techniques

(b) port to port transfer using interrupt techniques

(c) DMA transfer

(d) transfer using buffer memory.

C. Synchronous/asynchronous communication.

3. Information transfer rates/capacity: The transfer rate and capacity of a channel determines the number of busses required and their bandwidths.

4. Message handling: If the communication is based on messages between various modules, then the following considerations are important:

A. Message format should include information about source, destination, priority and error checking information.

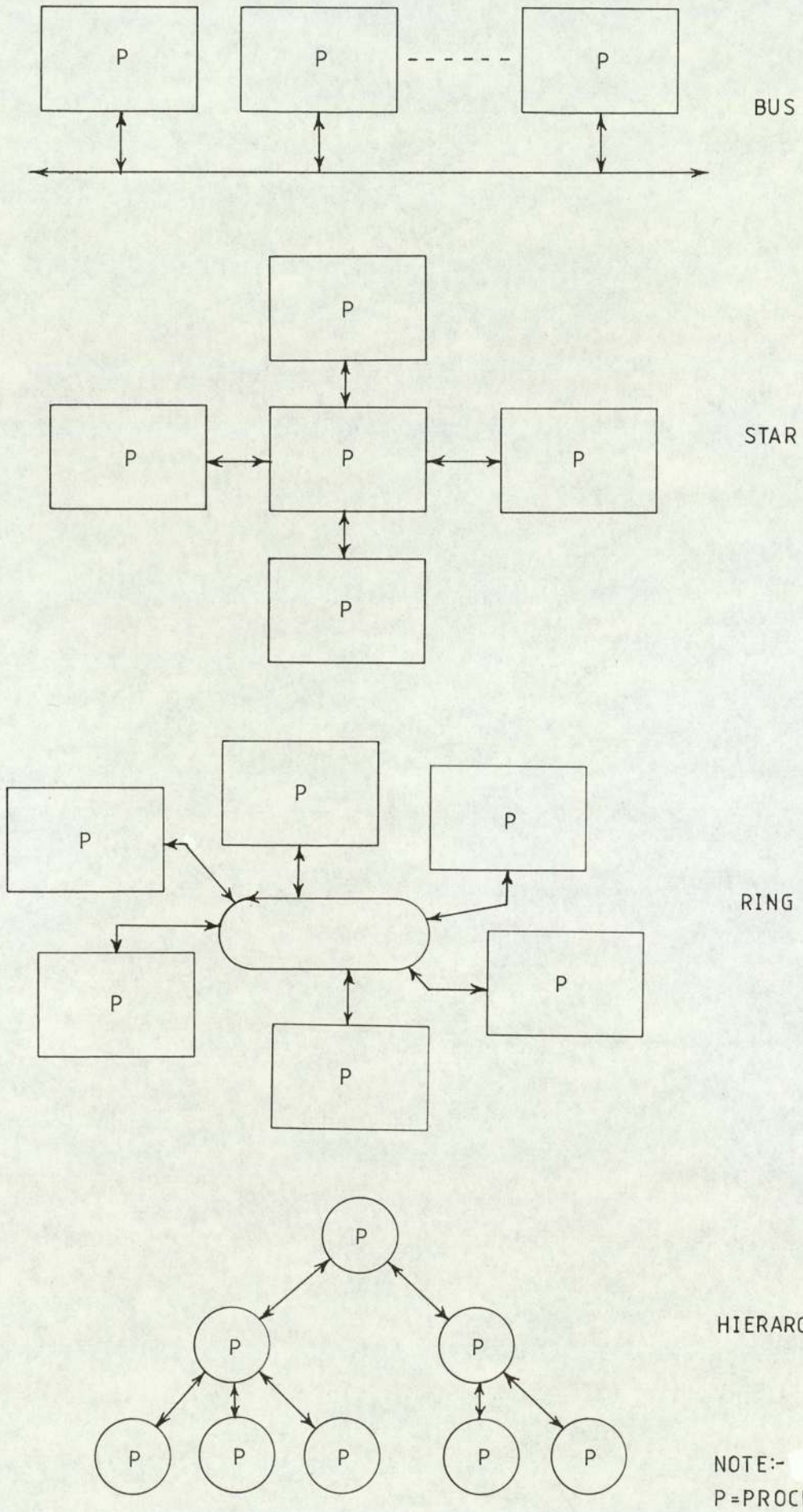


FIGURE 3.10 : INTERCONNECTION NETWORKS

B. Length of message: It could be short or long or of a fixed length.

C. Frequency of messages.

D. Error probability.

E. Acknowledgement delays.

F. Channel transmission rate.

5. System response time requirements: Normally several modules use the same data channel and hence sufficient data transfer rates should be maintained while meeting timing and bus utilisation constraints, and reduced queue lengths. Whenever the source generation rate is low, techniques of buffering and multiplexing may be used.

3.4.2.3 Distributed processing: Along with the static distribution of an application task implemented into various processors as individual subtasks, there exists interactions between them. Efficient handling of such interactions between modules depends on the design of the system architecture and the communication and control aspects of distributed processing. In addition, although the distribution of subtasks into processors is static, the actual processing and utilisation of these subtasks may be of a dynamic nature. For these reasons, the following elements of distributed processing need attention:

1. Task allocation: This consists of specifying explicitly the disjoint tasks and their interactions associated with a given problem. This may be possible only

for well-defined application areas where the requirements are known in advance. Another method is to determine the sets of individual tasks and allocate them in the local memories of the individual processors.

2. Fail-safe capability: One of the motivations for a distributed system is to provide a fail-safe system. Therefore, distributed processing needs to incorporate a detection mechanism for failures in the system and to isolate them so that the errors will not be propagated throughout the entire system. This feature is very useful for maintaining the system.

3. Data association and synchronisation: In many real-time, time-critical applications, the data to be processed in various processors needs to keep track of its source, when it was generated and how far and in which processor it has been processed so as to make further decisions for processing it or discarding it. This is clearly a data association problem which depends on synchronisation at various stages of distributed processing.

4. Resource allocation: Just as task allocation, software control of hardware resource (e.g. memory, data bus etc.) allocation and deallocation is another important task. Since there is no central scheduler in a distributed system, sufficient intelligence should be provided at various processors so that they can self-schedule and handle resource allocation. This is achieved by providing

updated strategic information about the system status, thereby allowing a particular processor to decide upon the allocation of a resource for a given request at any given time.

3.4.2.4 Distributed data base: It is a very common requirement for a distributed system to have data distributed among various processing units as well as a common data base which bears a functional relationship between various processes residing at various mouldes of the distributed system. This requirement is more prominent if the system as a whole is working on a single overall application problem. However, the structure of a data base, whether distributed or common, depends on the application at hand and the following points are important in this respect:

1. Memory partitioning: The size of memory for data and program should be determined carefully for each processing module, allowing for expansion if necessary. This basically depends on the application and the access time requirements.

2. Nature of data: The data base may be static or dynamic in nature. A static condition refers to data segments or files that are not modified, while data which gets modified and utilised either externally or internally during discrete processing steps can be considered as dynamic.

3. Data access time and throughput: For certain real time applications, the data access time may be very critical and hence a careful memory system design is needed to satisfy the system throughput requirements. Holland (1980) has described three ways of improving the system throughput by separating the data in, data out and memory address busses of the memory system. These are (a) address anticipation, (b) pipelining, and (c) cache memory.

4. Access conflicts and deadlocks: When a data base is shared between various processing elements, there may exist conflicts in accessing certain data items and simultaneous access may not be permitted. Such conflicts should be considered in conjunction with the allowable delays, priorities for access and the cost associated with the duplication of memory system hardware. Unresolved access conflicts and/or the unavailability of critical data items or control information can lead to a deadlock situation. Hence deadlock prevention is important and provision must be made to detect and backup in the case of a possible deadlock.

3.4.2.5 System reliability, availability and survivability: It is very difficult to discuss quantitatively concept of reliability, availability and survivability of distributed systems because basically these systems are application-oriented and faults leading to system breakdown are, in general, intermittent in nature. However, these issues are very important if reliable system performance is required, which is generally the case. For this

reason, it is desirable to implement error detection and recovery techniques within the system. If a detected error fails to recover, then the system survival depends on the operation of the critical processing modules and isolation of the failed module. In such cases, it is necessary to provide sufficient redundant information about the system to be able to recover even after the total failure of some subsystem. If the cost constraints allow, extra redundant hardware may also be used to backup the system to improve reliability.

3.4.2.6 System development and testing: When a distributed system has been carefully designed, based upon the considerations outlined above, the development and testing of such a system can be a major problem. The design issues mentioned earlier in Section 3.2.1, as applied to the development of a system incorporating a single microprocessor, are multiplied by the complexity of the number of such systems, their interrelationship and interconnections which make up a single multi-microprocessor/microcomputer system. The ease of development and testing of such a system depends upon the fine description of the details of the lowest level of language notation for system architecture, hardware and software, and their integration.

The development of a single processing element or a module which forms a subsystem of a distributed system can be performed partially using design techniques outlined in Section 3.2.1. However, the functional contribution of such a subsystem towards the entire system is very difficult to

test because it can only be tested if the rest of the system is present. That is why the integration of various subsystem modules, developed and partially tested individually needs carefully programmed test procedures. This need also arises due to the absence of a general purpose system which can simulate a multi-microprocessor system environment for real-time applications. The only way around this problem is to develop and build a desired multi-microprocessor system by a step-by-step approach. In this approach, partially tested developed subsystem modules are integrated one by one and testing is carried out with modular test programs or built-in test procedures together with externally generated signals which simulate the real-time application environment. When such a distributed system is developed and tested successfully, then the simulated environment can be replaced by the actual real-time application.

3.5 CONCLUSIONS

This chapter demonstrates that a trend towards systematic design of multiple processor systems is developing. The classification issues for such systems are vague because of their multi-dimensional attributes and complexity and lack of acceptable common terminology. However, an attempt to classify these systems, based upon easily identifiable characteristics, have been made.

The problems of designing with microprocessor and multi-microprocessor systems suggest that numerous design

issues need utmost attention even prior to undertaking a microprocessor-based project. In particular, integrating various modules of distributed systems can be a major problem. The distinction between a distributed system and a multiprocessor system, based upon the operating system, is very weak because it is perfectly feasible to intermix some powerful features of an operating system with the flexibility and variety of characteristics offered by distributed systems. For example, it may be possible to build modular systems which include such features as dynamic task allocation (i.e. reconfigurability) to suit a variety of applications. However, this in itself is a research area.

CHAPTER 4 - MODEL OF A PROCESSOR WITHIN A DISTRIBUTED COMPUTING SYSTEM

4.1 INTRODUCTION

Distributed computing systems are still at the forefront of their evolutionary process. This evolution is taking place at architectural design level, interprocess communication design level, intercomputer communication design level and application level. Consequently, there are many and varied definitions and taxonomies of distributed computing systems (Jensen, Thurber and Schneider, 1979). However, these systems in general refer to the use of multiple, quasi-independent processing modules whose actions are co-ordinated to accomplish a large task or to implement a large system. In general, a designer of these systems is concerned with the following agenda:

1. Distribution of computing power both in hardware and software.

2. Distribution of information processing in the form of top-down distribution of tasks and bottom-up co-ordination of tasks.

3. Distribution of data. This has two categories: (a) data generated as an output from a distributed task, and (b) data required as an input to a distributed task.

A meaningful implementation of the above is usually associated with a specific application that characterises a distributed computing system.

The emphasis of this chapter is on issues, regarding the interfacing of a distributed computing system to a large-scale real-time complex system. A dual port memory utilisation is reviewed on this background, and a realisation of a hypothetical application is considered.

4.2 *REAL-TIME DISTRIBUTED COMPUTING SYSTEM*

Figure 4.1 shows our description of a distributed computing system. The system is employed to serve the needs of a large-scale complex real-time system for its information processing. It is assumed that the large-scale system already exists. This assumption is reasonable with most practical systems. For example, one can think of a large chemical processing plant, the throughput and performance of which needs improvement. A distributed computing system can be a cost-effective solution for such a problem.

In the Figure, P1, P2, P3 etc. are microcomputers with normal attributes of a conventional computer system. This facilitates a desired task-oriented program development environment for any processor to be accomplished independently under Phase I. The processors' interconnection interface and their physical interconnection system bears a close relationship which is exclusive to the processors only. This relationship can be made adaptable for a variety of microcomputer networks and communication protocols that link the processors for the co-ordination of their individual functional tasks. A functional task may involve numerous interactions of a processor with the large-scale real-time system or it can be a task of micro-co-ordination

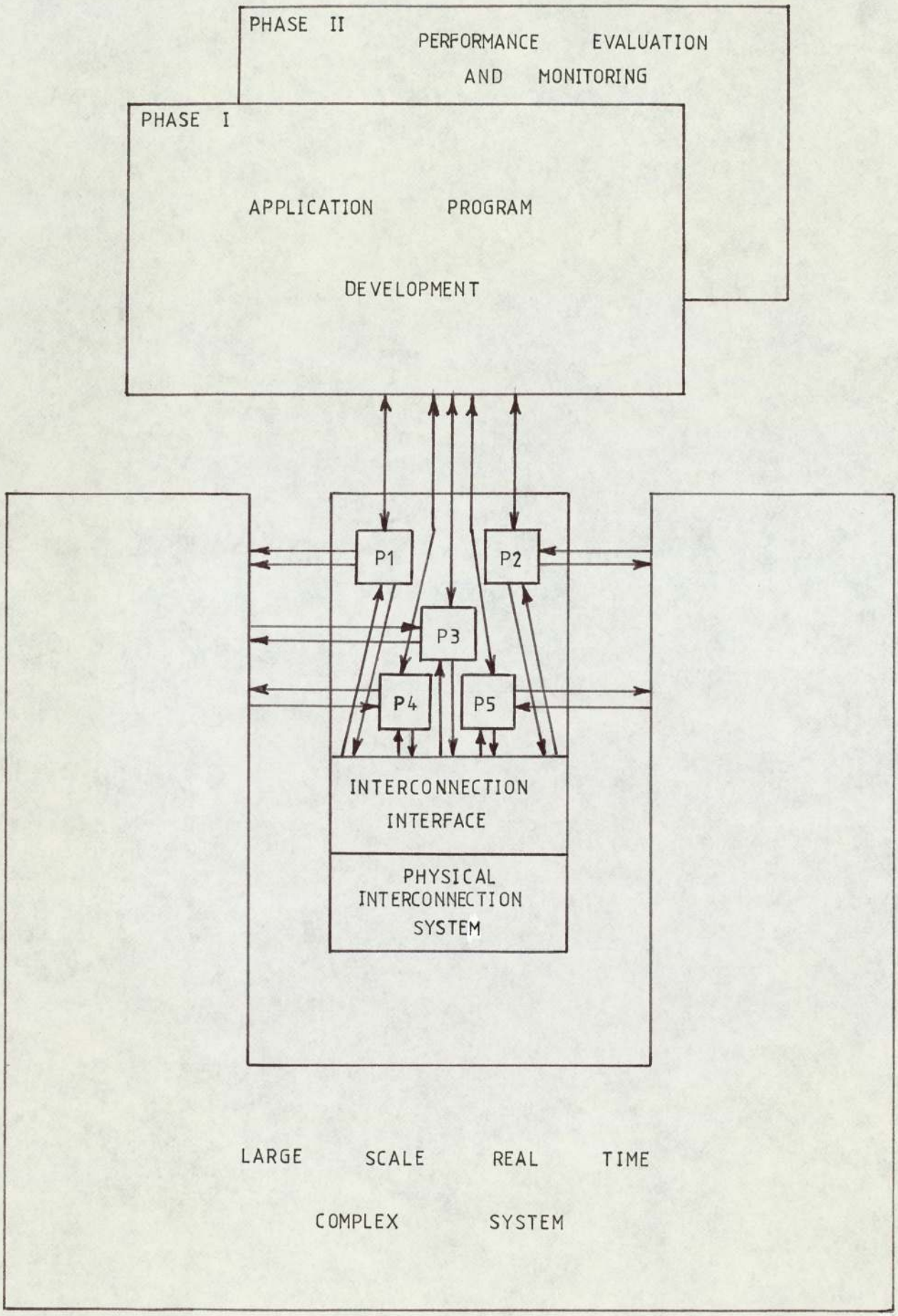


FIGURE 4.1: Description of a Distributed Computing System

of other neighbouring tasks. An overall collective co-ordination of tasks executed by the processors is thus dispersed amongst the processors, each one being a contributor to it to some extent.

A separation of Phase II and I is quite arbitrary in the system shown. The performance evaluation and monitoring of the processors' behaviour and the large-scale systems' behaviour in Phase II occurs as a result of successive developments in Phase I. A gradual hand-over from a then existing control scheme of the large-scale real-time system to a new implementation of a distributed computing system is thus possible with the following major advantages:

1. A modular development of the system, both at software and hardware level.

2. A better insight into the system down to a smallest subtask level.

3. A reduction in down time of the large-scale system.

4. Improved maintainability due to improved failure detection.

5. Improved performance, throughput and reliability.

4.3 MODEL OF A PROCESSING ELEMENT

A major element of a distributed computing system is a processor. A task programmed into the memory of this processor accounts for its information processing capabilities. A processing task is performed on the input

information to produce the resultant output information. A detailed description of a model of a processing element for a distributed computing system is shown in Figure 4.2.

One of the main features of this model is that a processor is assigned a task which is composed of a "process" and "output information data". The process may contain several subtasks. The task of a processor is activated by one or a number of sets of input information data which is contained in the "Information Accumulator Node" (IAN). The output information data from the processor is deposited in the "Information Distributer Node" (IDN). The characteristics of IAN and IDN are such that a processor avoids direct interference with another processor's task and vice versa. This facilitates the identification of a processor's communication requirements with another processor or processors.

Another interesting feature of the model is that a processor receives its input information data without any forced interruption of its task execution. Similarly, a processor generates its output information data which is made available to another processor to read it whenever it is free to do so. Thus input information data received by a processor is transformed into output information data by a task. The output information data generated in this way can flow through four different kinds of information links. These are:

1. Feedback Information Link: As the name suggests, the output data is fed back as input to the same task. For

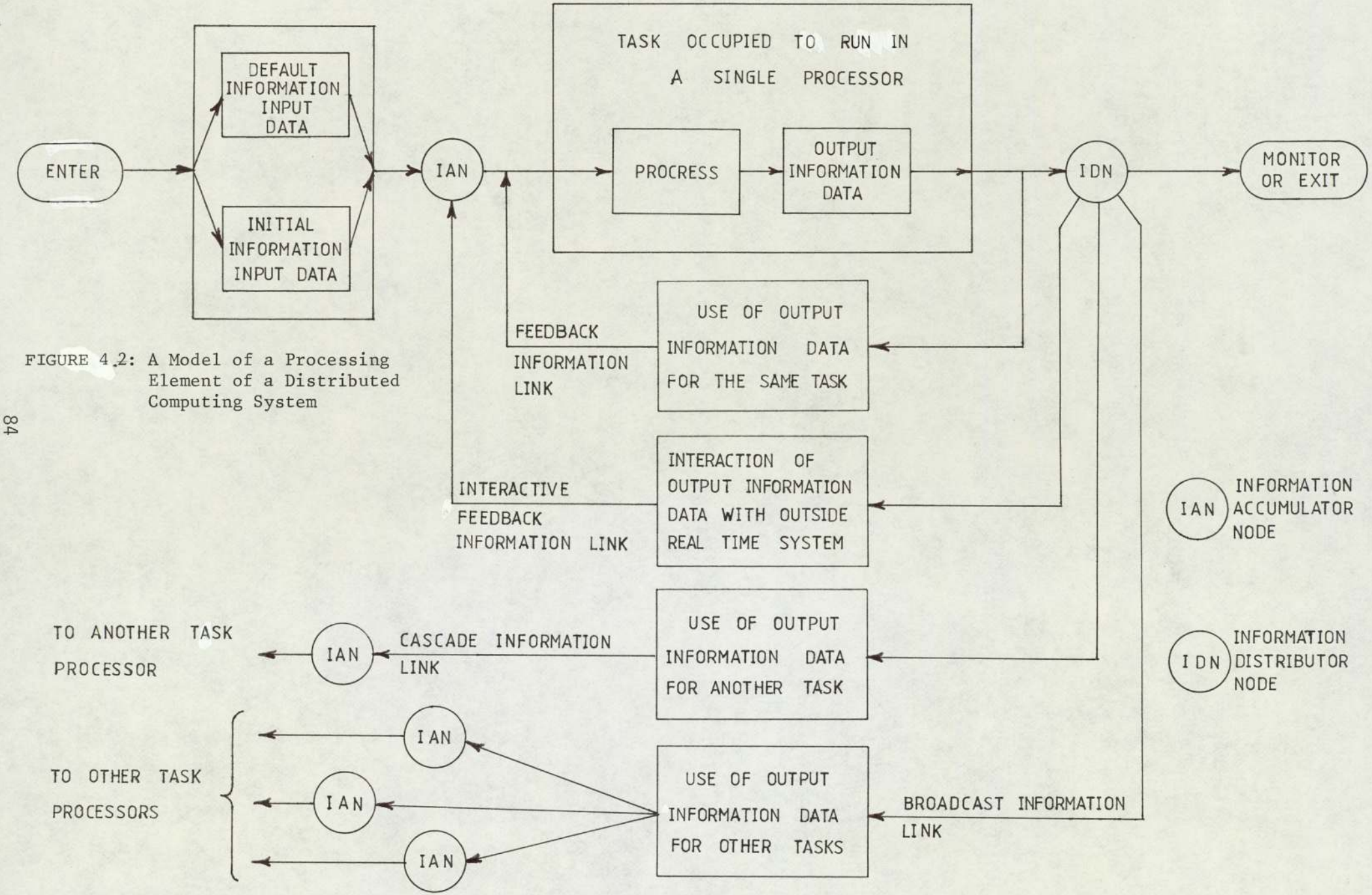
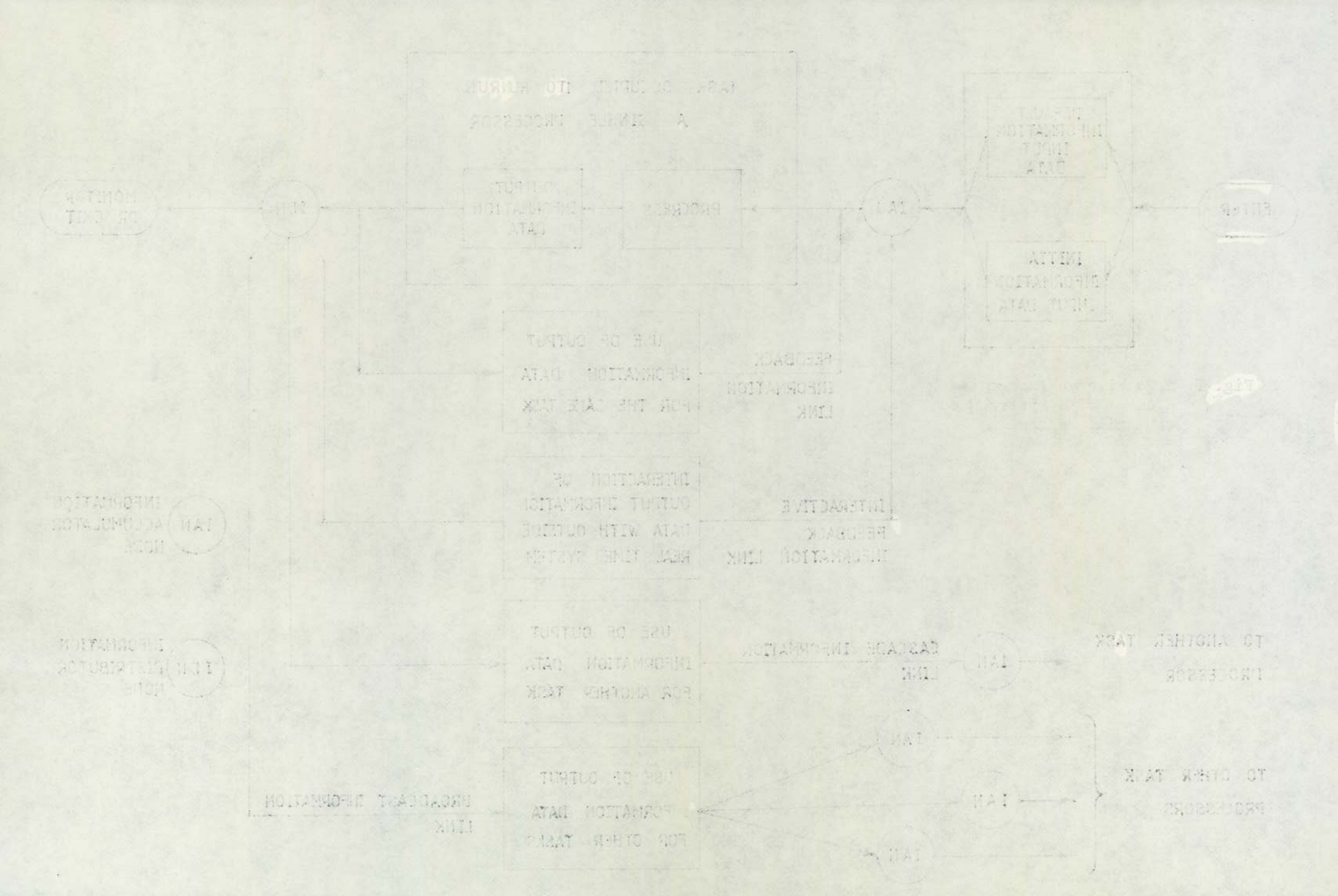


FIGURE 4.2: A Model of a Processing Element of a Distributed Computing System



example, recursive type algorithms run in a monoprocessor fall under this category.

2. Interactive Feedback Information Link: In this type of link, the output data interacts with its outside world which it is controlling, and the processor reacts to the data presented to it by its controlling environment. Direct Digital Control (DDC) of a process is a good example to illustrate this.

3. Cascade Information Link: Using this link, it may be possible to cascade a number of task processors. This situation may arise if a single processor fails to accommodate a single large task or, for example, it may be that a processor after completing its task wishes to trigger another task processor in cascade with it.

4. Broadcast Information Link: This link is basically an extension of the cascade link in which output information data is made available simultaneously to a number of other task processors. This link is very useful if a number of task processors execute identical tasks. This link may also be useful in synchronising different task processors.

A physical implementation of IDN and IAN is shown in Figure 4.3 and Figure 4.4 respectively. Each module of IDN or IAN is made from dual port scratchpad buffer memory. The roles of IDN and IAN are identical. In both of them, information data is stored from one end and it is made available at the other end. However, the way in which the modules are grouped and connected makes them either IAN or

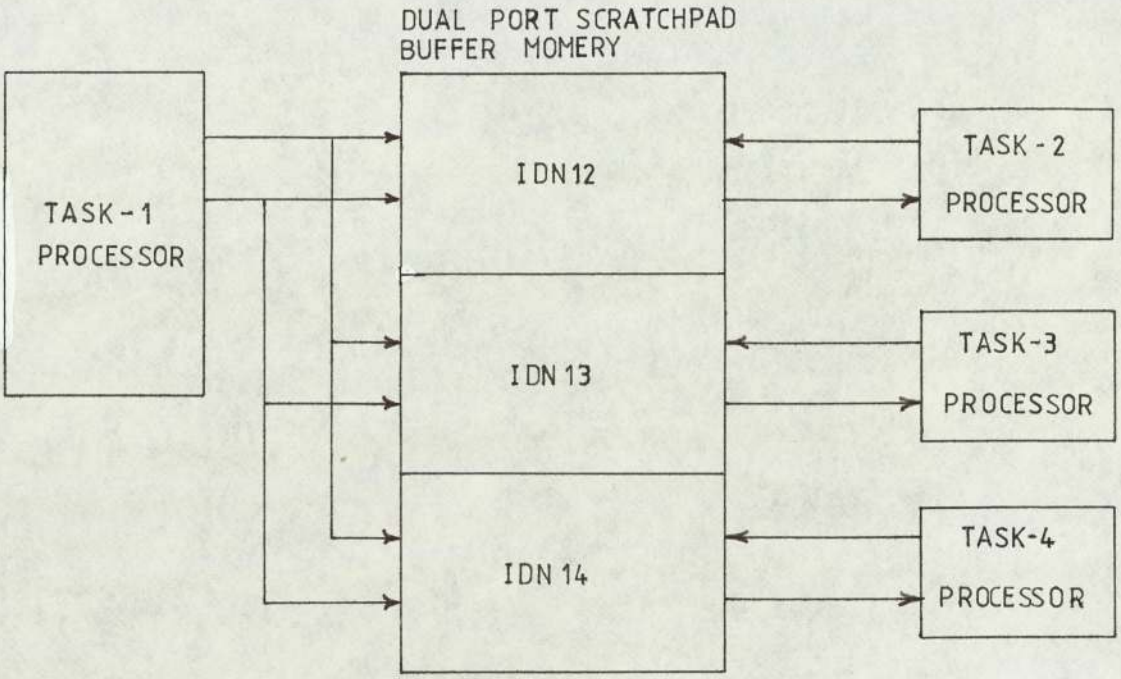


FIGURE 4.3: Physical Implementation of the Information Distribution Node

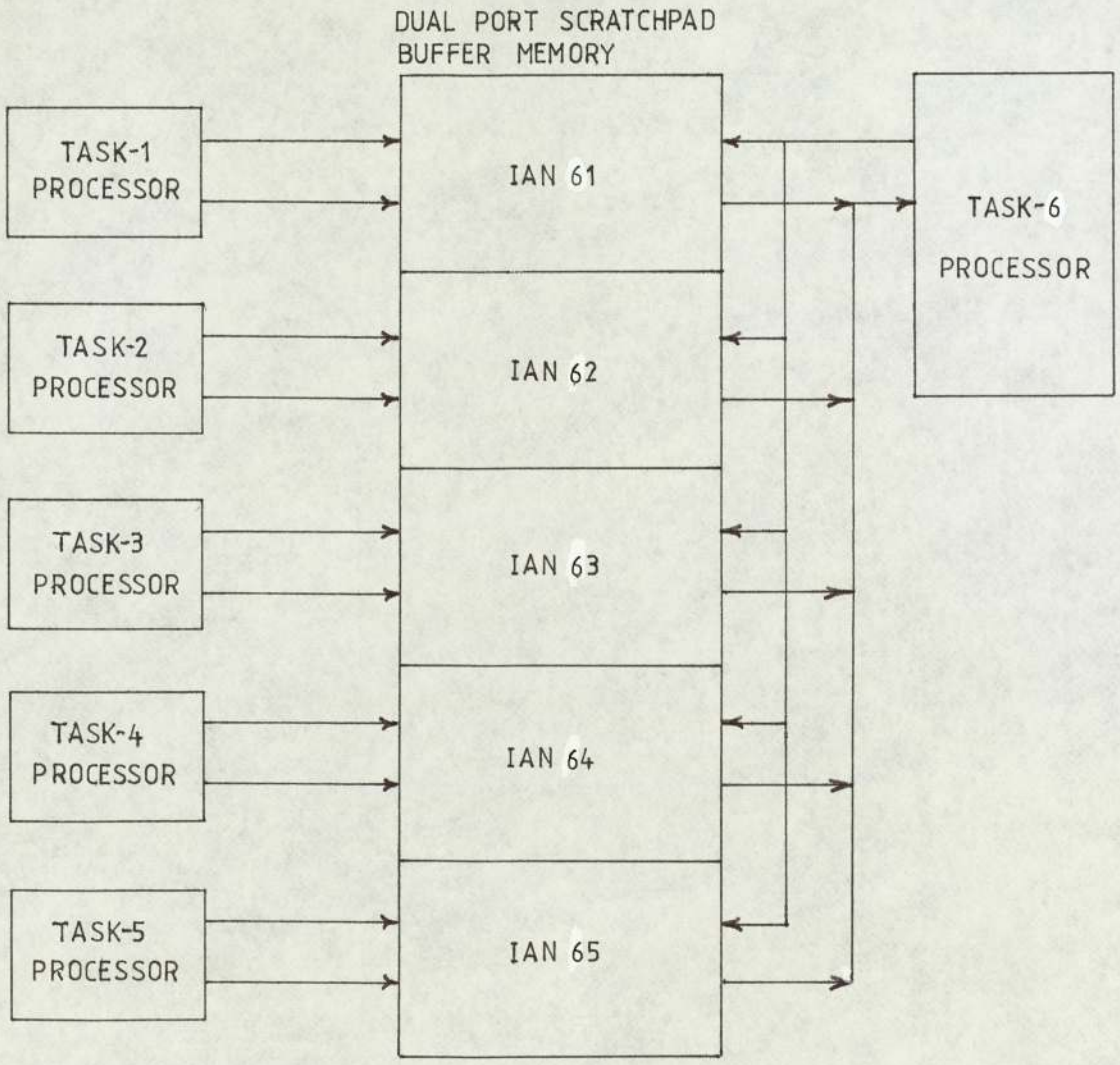


FIGURE 4.4: Physical Implementation of the Information Accumulator Node

IDN. For example, in Figure 4.3 IDN13 represents that the Task 1 processor distributes its output information data to the Task 3 processor. Similarly, in Figure 4.4 IAN74 represents that the Task 7 processor accumulates its input information data from the Task 4 processor and so on. Figure 4.5 shows an example of two cross-coupled processors P1 and P2.

4.4 INFORMATION AND TASK HIERARCHY

A task processor may contain one or more information links. The feedback and interactive feedback information links are mainly associated with a monoprocessor, while cascade and broadcast information links account strongly for a distributed computing system. However, the smaller the number of information links, then the more simple the task becomes.

In order to derive some criteria for quantifying a complex task, one can look at the information hierarchy used in information theory. This includes:

1. Symbolic Information: At this level, messages are transmitted (and data is stored) as a collection of symbols; these symbols form basic building blocks from which all higher forms of information hierarchy are developed.

2. Syntatic Information: This is contained in the rules limiting the way in which various symbols can be combined.

3. Semantic Information: This is contained in the meaning which the recipient can perceive in a message.

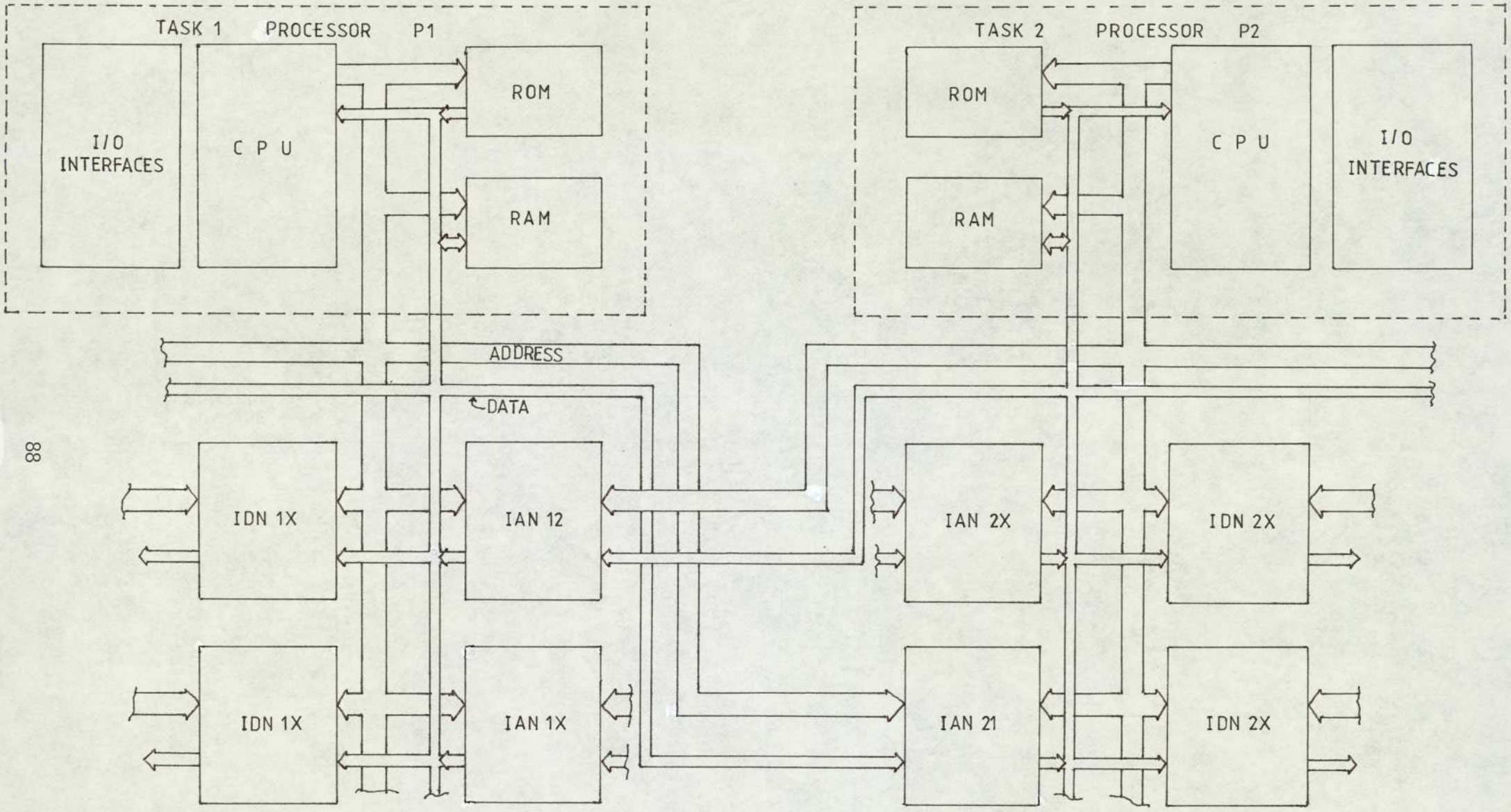


FIGURE 4.5: An Example of two cross coupled Task Processors P1 and P2

4. Pragmatic Information: This is concerned with the practical use to which the recipient may make of a message.

5. Aesthetic Information: This describes the ability of the message to affect the senses and decisions of the recipient.

A processing element of a distributed computing system bears an analogous relationship to the above information hierarchy. This is shown in Table 4.1. In a distributed computing environment, we are mainly concerned with the aesthetic level of task hierarchy where a task performed by one processing element affects the decision of another or several other processing elements. In other words, this is concerned with the micro-co-ordination function amongst the processing elements. This micro-co-ordination function is responsible for minor decision-making based upon the outcome from different neighbouring task processors and information filtering. Information filtering relates to the form in which a distributed computing system presents the net quantified information about its controlling environment to a human operator to perceive the performance of the controlled environment. This task of presenting information by a distributed computing system to human perception represents the highest level in the information hierarchy.

4.5 CONTROL SYSTEM PHILOSOPHY

The form in which a model of a processing element of a distributed computing system is presented also reminds us

INFORMATION HIERARCHY	A PROCESSING ELEMENT	
	SOFTWARE	HARDWARE
1. Symbolic	"0" and "1" Symbols	Logic gates, registers, counters, flipflops, decoders etc.
2. Syntactic	Hexadecimal, octal numbers	CPU, ROM, RAM etc.
3. Semantic	Instruction set as implied to be perceived by a computer	Computer architecture
4. Pragmatic	Programming of a task or an algorithm	Interfacing a computer to a real outside world
	Integrated System	
5. Aesthetic	Ability of one integrated system to interact with another integrated system	

TABLE 4.1: Analogy of information hierarchy to a processing element

of "signal flow diagrams" and "block diagrams" from classical control system theory. The terms "feedback" and "cascade" have been chosen deliberately to bring about a philosophical analogy of a distributed computing system to classical control systems theory.

In a distributed computing system we are concerned with "information flow diagrams" similar to "signal flow diagrams" in a classical control system. A time constant, for example, of a processor to run its task relates to a delay in time after which output information data appears when input information data is applied. This delay may not be of a fixed duration; usually this will have maximum and minimum limits on it depending on the volume, rate and nature of input information data. This naturally leads to stability consideration for task processors to be determined. For example, a stack overflow in task processor will be clearly an unstable situation. A deadlock situation between two task processors is another unstable condition and so on.

In order to identify such unstable conditions, information flow in a distributed computing system should be "observable" and consequently "controllable". This feature relates to the fact that each task processor should be examinable for its task-handling and information-handling attributes. Another possible outcome of this examination is the identification of a "critical path" along which critical tasks are processed. This is analogous to a PERT analysis in system design. This critical path can be very important with regard to the "micro-co-ordination" or "decision-making"ability of a task processor.

4.6 *DUAL PORT MEMORY UTILISATION*

A dual port scratchpad memory allows a data item to be stored (written) into a location from one port and allows it to be retrieved (read) from the location at another port. These read and write operations can be performed simultaneously. This type of memory acts as a buffer storage medium for the communicating task processors and serves to isolate the internal data, control and address bus systems of these processors. Input/output information data flow occurs through this medium denoted by IAN and IDN in the model.

There are two ways in which the dual port scratchpad memory may be connected to a task processor. If the volume of information data flow is small, the processor's parallel input/output ports may be used. However, this means that there will be a smaller number of I/O ports available for connecting other peripherals or interfacing circuits. Another way is to connect the processor's internal address and data bus to either the read or write end of the dual port memory. This allows data storage and retrieval operations to be the same as RAM. This type of connection is suitable for a large volume of data transfer.

The size of the dual port scratchpad buffer memory used in IAN and IDN is a function of the information transfer needed between task processors. There are three kinds of devices available to build IAN or IDN modules. These are:

1. SN74LS670 MSI 16 bit TTL register files. These files are organised as four words by four bit with on chip address decoding for separate write and read functions, thus permitting simultaneous reading from one location and writing to another. The device is 16 pin DIN packaged. It requires a combination of two such devices to implement only four word bytes of storage locations (Deshmukh, 1979).

2. AM29705 is a 16 words by 4 bit 2 port RAM. This device has two output ports each with separate output control and separate four-bit latches on each output port. The device is 28 pin DIN packaged (AMD Data Sheet).

3. MC10806 is a dual access stack with 32 x 9 memory, two address ports, two 9-bit data input/output ports, two 9-bit output registers, flipflops in a single MECL bipolar LSI circuit. The device is 48 pin QUIL packaged (Motorola, 1979).

The first two devices are simpler to interface with most microprocessors while the third has rather special characteristics. For the purpose of designing systems with these devices, data sheets are available and so no further discussion is given here.

The hardware utilisation of a dual port scratchpad memory module such as IAN or IDN requires additional software protocols for the purpose of information flow between task processors. An outline of simple protocol primitives that may be used is shown in Table 4.2. This set of primitives is derived for two cross-coupled task processors P1 and P2, as shown in Figure 4.5. Table 4.2 shows what

NO	PROTOCOL PRIMITIVE FOR IDN12 = IAN21 (TASK PROCESSOR 1)	PROTOCOL PRIMITIVE FOR IDN21 = IAN12 (TASK PROCESSOR 2)
1	Stop Read/Start Write	Stop Write/Start Read
2	Number of bytes of information	--
3	Block program Loading/Finish (a) Starting address (2 bytes) (b) Block number	)) Block Read)
4	Task number (a) Task trigger/End (b) Repetition number	)) Task Status
5	Read time information	--
6	Data constants	--
7	Information set number	Set accept information

TABLE 4.2: Information Protocol Primitives

information data Task Processor 1 intends to distribute to Task Processor 2. This information storage is done in IDN12 = IAN21. The Task Processor 2 on the other hand may acknowledge its input information data via IDN21 = IAN12. This coupling of the two processors for their information exchange is entirely programmable and task-oriented.

4.7 APPLICATIONS

The applications mentioned in this section are hypothetical and intended to show potential areas where our model of a distributed computing element can be employed. Two applications are considered:

1. Mathematical Modelling: Modern control philosophy suggests that a real-time large-scale complex system may be analysed and controlled by utilising its mathematical model which resides within a computer system. Any interactions between the subsystems of such a large-scale real-time system can be accounted for by implementing these subsystems within our model of a processing element of the distributed computing system. The mathematical model performance and the actual system's performance can then be compared at a subsystem level. Additionally, actual subsystem's parameters can be estimated and consequently its model parameters can be updated for that particular subsystem. Furthermore, different mathematical models can be analysed and tested.

2. Simultaneous Serviceing of Interrupts: In some situations with a real-time system, it may be difficult to

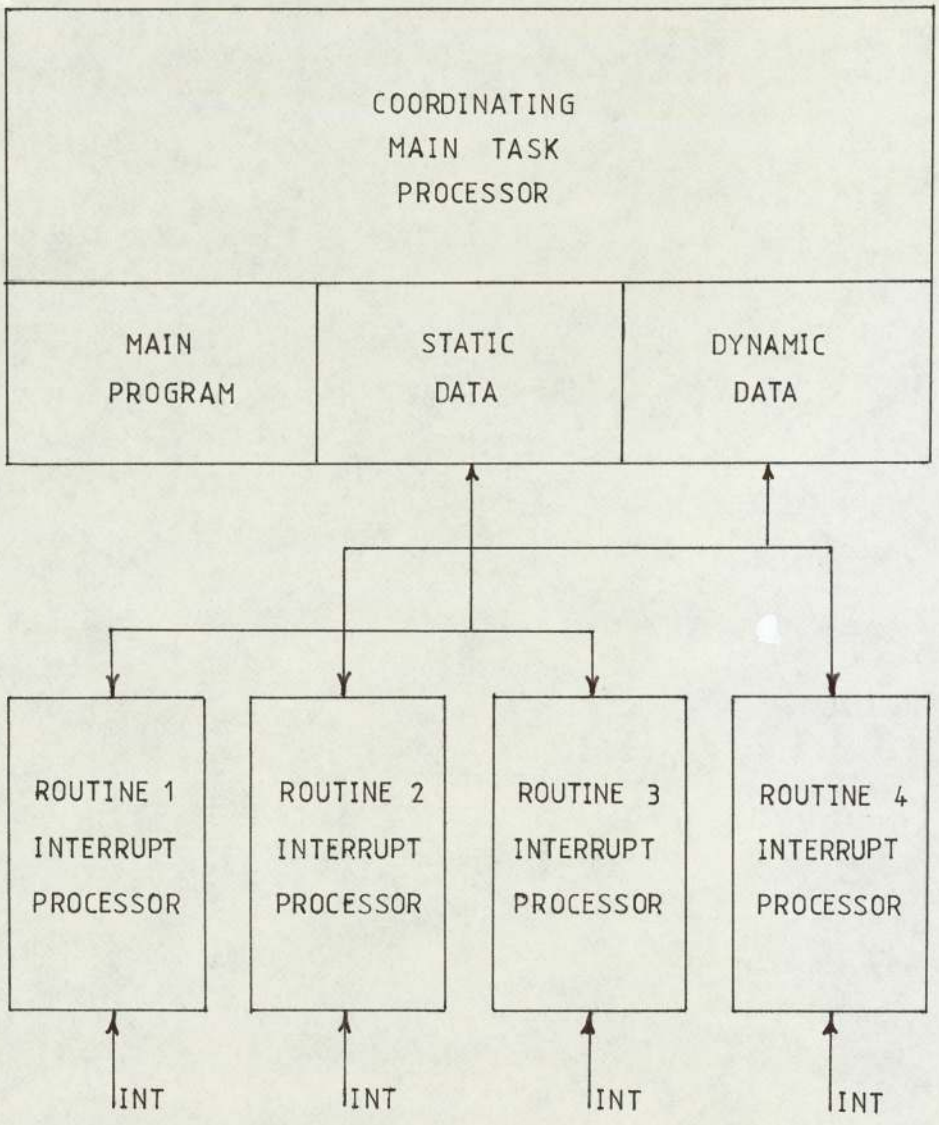


FIGURE 4.6: Simultaneous Servicing of Interrupts

determine the priority structure of a number of external as well as software interrupts. In these circumstances, different interrupt service routines can be implemented in a processing element of a distributed computing system and the overall co-ordination of these interrupt service processors can be performed by another processor whose task will be to run the main program and take care of any static and dynamic data movements to and from the interrupt service processors via IDNs and IANs. This is shown in Figure 4.6. In the Figure, routines 1 and 3 depend on static data from the main task processor whereas routines 2 and 4 depend on dynamic data. Hence, these two routines interact with each other as well as with the main program. The structure of the interrupt processors depends upon the actual application and how it relates to the main program in the co-ordinating processor.

4.8 CONCLUSIONS

A new model of a processing element within a distributed computing system is presented. The cost-effectiveness of this model needs to be evaluated. The model provides means by which new possibilities of communication protocols may be implemented which are task-oriented. The model also facilitates a clear partitioning of subtasks and a definition of their interactions.

CHAPTER 5 - A HIERARCHICALLY STRUCTURED MULTI-MICROPROCESSOR SYSTEM

5.1 INTRODUCTION

As computers and processors have become smaller, cheaper and more reliable, it is becoming more common to design systems with more than one actual processor. A large variety of computer interconnection structures has been proposed covering the range from tightly to loosely coupled networks and multiprocessors to array processors (Anderson and Jensen, 1975; Enslow, 1974). The concept of distributed processing had its origins in the data processing field before the start of the microprocessor revolution. Enslow (1978), in attempting to clarify the concept of distributed data processing, claims that at least four physical components of a system might be distributed: processing logic, data, the processing itself and the control of the operation (e.g. the operating system). Research in this area is continuing.

The advent of microprocessors has helped to enlarge the concept of distributed processing beyond the confines of data processing applications. Many types of system have been described, ranging from a series of unconnected computers each performing separate tasks through to a single computer system within which a number of computing elements are connected. Enslow (1976) has discussed systems classified as multiprocessors which contain two or more central processors of comparable capability. These processors

share access to a common memory, common input/output channels and common control devices; the entire system is controlled by a single integrated operating system.

Microprocessor technology, however, is often best employed in systems which are constructed of processing units each of which is independent in itself but which communicates with some or all of a number of other processing units in the overall system. Each processing unit may have a number of dedicated tasks in normal operation; there is, however, no integrated operating system, and the control of the system may be distributed among the individual units.

This chapter describes such a system in which the units are connected in a hierarchical structure. The basic processing module from which the system is configured is known as a "Hierarchical Microprocessor System Unit" (HMSU). The system is designed for the control of a pilot-scale 8-zone travelling-load furnace, but is sufficiently flexible to have a wide variety of process control applications.

The HMSU structure consists of a number of Fairchild/Mostek F8 family chips, a common block of semiconductor memory and a pair of Intermediate Scratchpad Memory Interface. The configuration is designed so that a single HMSU can be used either independently or as a building-block in an expandable hierarchical environment. In either case, it will normally run dedicated programs which will be held in ROMs.

Similar uses of microcomputers have been described by other workers. Harris and Smith (1977) have analysed a

number of multiprocessor architectures and have discussed a multi-microprocessor architecture having a hierarchical structure. Steinhoff (1976) concludes that the computational potential of minicomputers and a set of bipolar microprocessors can be harnessed for solving some large scientific problems that cannot otherwise be solved within normal economic and practical constraints. It is not necessary for all the processors within one system to be of the same type; for example, Pathak (1977) describes a configuration of one Intel 8080 and three SC/MP processors. Hughes (1976) incorporates TI 9900 series microprocessors and 990 computers for multiprocessor navigation systems. Tanaka (1976) introduces a new type of hierarchical multi-microprocessor system that includes nine microprocessors operating in a system under the overall control of a host ECLIPSE S/200 computer.

The objective in developing the HMSU is to make use of the numerous advantages offered by distributed processing in establishing a hardware basis for the implementation of optimal control schemes for large-scale system problems.

5.2 *HMSU PHILOSOPHY*

The hardware configuration of the HMSU is designed on the following basis. The unit consists essentially of a number of individual processors and memory blocks. Each processor has its own private memory, but the bulk of the memory is common to all processors. It is the task of one particular processor, designated the Master Processor, to

control access by any other processor to the common memory. Apart from this function, each individual processor acts independently, performing designated control functions via its own I/O channels. The processors operate asynchronously, all inter-processor communication being via the common memory under control of the Master Processor.

The unit as a whole communicates with the outside environment via a special buffer known as the Intermediate Scratchpad Memory Interface (ISMI). The outside environment may be either another HMSU or a larger host computer, or indeed any other processing equipment as required by a particular application.

5.3 INFORMATION FLOW

In designing multi-microprocessor systems such as HMSU, it is important to consider the basic principles of information flow. Microprocessors are intelligent devices capable of acting as a source or as a sink of information. Figure 5.1 shows an example of three units acting as sources and sinks of information and the arrows indicate all the possible ways of information flow that may occur. If these three units are to communicate sensibly with each other, then at any one time one unit must be transmitting information and the other two receiving it. It will greatly help the synchronisation problem if the data is transmitted by the source to a temporary intermediate store, from which it may be received by the sink or sinks when they are ready to do so. This avoids the difficulty that can occur with "handshake" systems when two processors may each be waiting for the other.

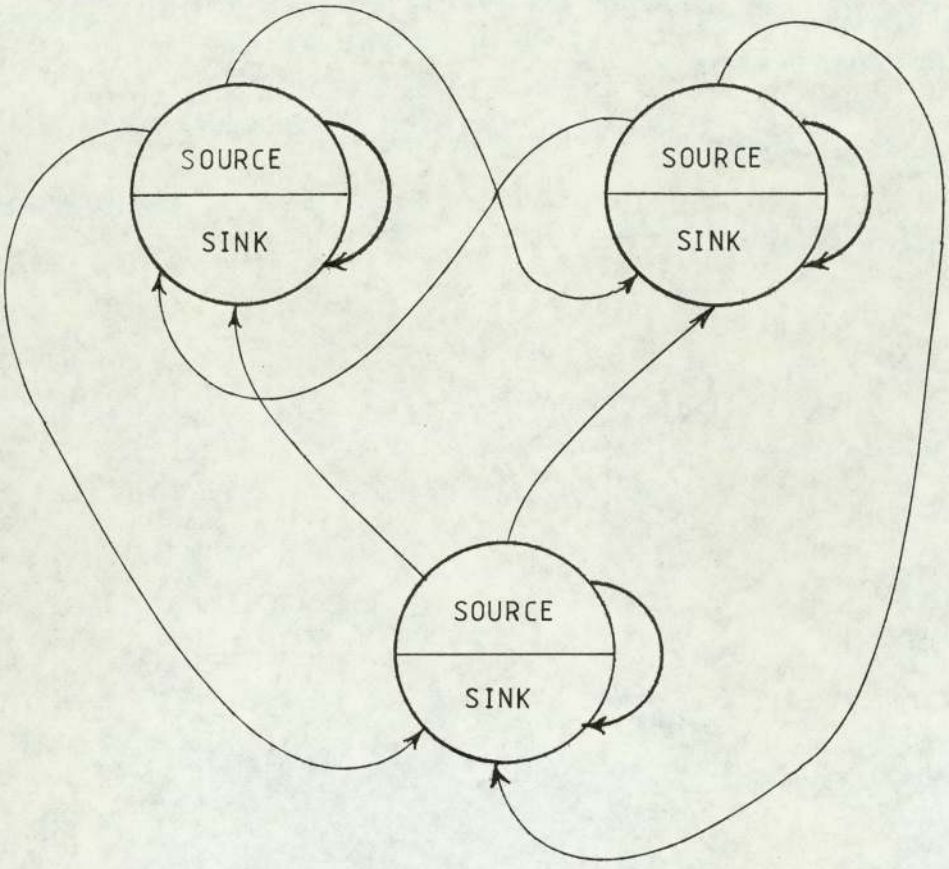


FIGURE 5.1 : Information Flow

The Intermediate Scratchpad Memory Interface (ISMI) that forms such a temporary store for the HMSU allows two processors to use it to deposit or retrieve data or control information. Asynchronous reading or writing of data can be performed by the two processors simultaneously. In the case of dedicated applications, the form of interprocessor information flow is completely known and some simple synchronisation schemes may be adequate. In our case, where a pair of ISMIs is employed as intermediate information storage media, simple software controlled synchronisation primitives for block data transfer can be utilised. For example, in Figure 5.2 two processors, P1 and P2, are

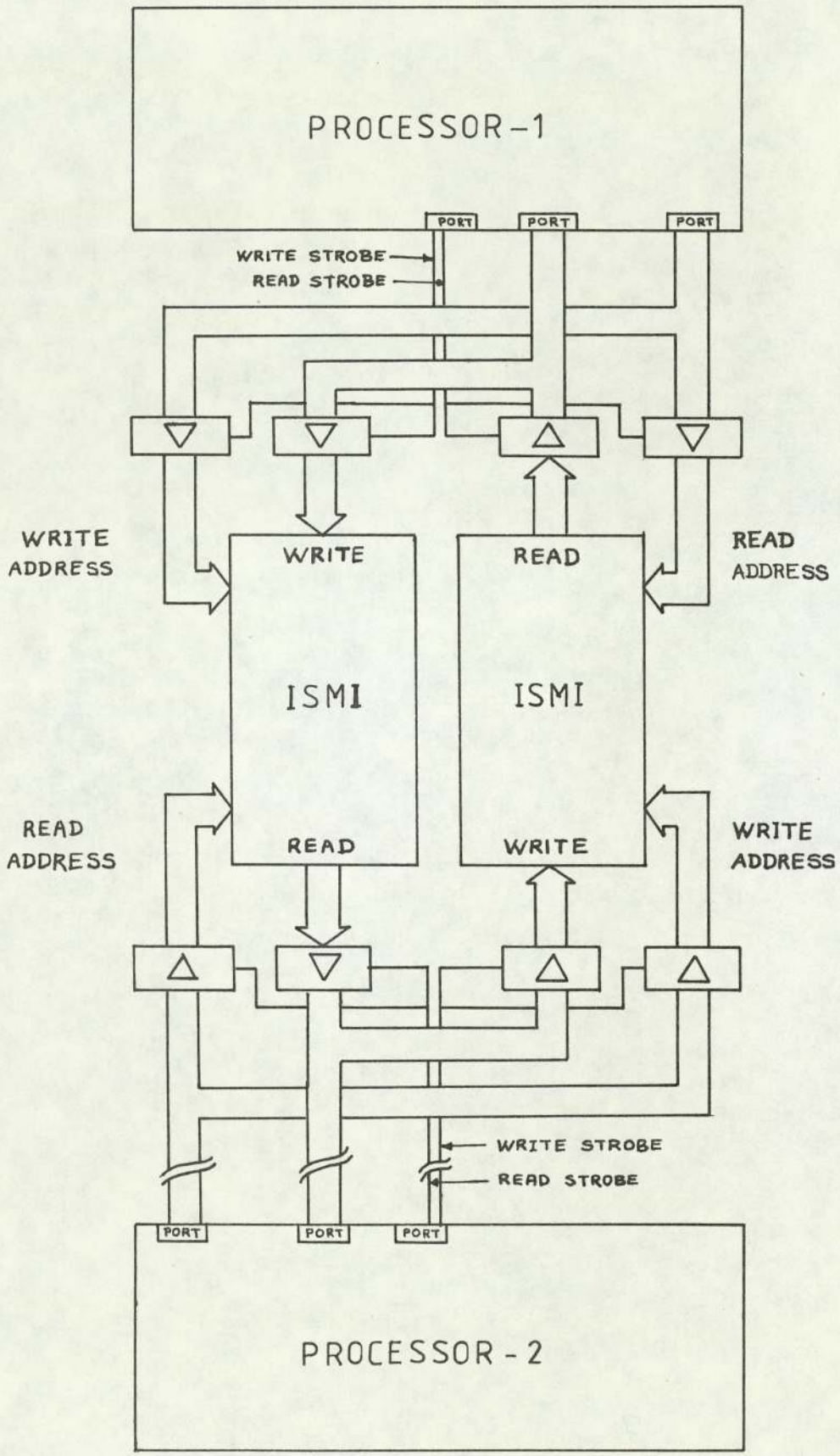


FIGURE 5.2 : Bidirectional Communication between processors via a pair of ISMIs

linked by a pair of ISMIs. A and A_1 are the locations in this pair which are periodically monitored by processors P2 and P1 respectively. These locations can be used as flags or codes for various sets of block data. The following Table 5.1 shows how A and A_1 are used as flags for a block data transfer between the two processors P1 and P2. The only constraint on the software programmer is in the assignment of individual ISMI locations to particular items of data. If the functioning of a task residing with P2 depends upon the data generated by P1, deadlock can occur. However, in a dedicated system such as a HMSU, where applications can be either homogeneous or heterogeneous (Siewiorek, 1975), deadlock problems are certainly anticipated by the very nature of hierarchy. The frequency of deadlocks in a multi-microprocessor structure is an important question open to experimentation.

The volume of data flow in the two directions need not, of course, be the same. This is a necessary feature for use in a hierarchical structure, where one processor at high level may be receiving a great deal of data from a lower-level processor but sending to it only a few command signals at frequent intervals.

5.4 STRUCTURE OF THE ISMI

The Intermediate Scratchpad Memory Interface is built from SN74LS670 MSI 16-bit TTL register files. These files are organised as four 4-bit words: on-chip address decoding is provided separately for reading and writing, thus permitting simultaneous reading from one location and

A MONITORED BY P2	A^1 MONITORED BY P1	COMMENTS
0	X	P1 starts writing for P2 ∴ P2 should not read
1	X	P1 completes writing ∴ P2 can read
1	0	P2 starts reading ∴ P1 should not write
1	1	P2 completes reading ∴ P1 can write
X	0	P2 starts writing for P1 ∴ P1 should not read
X	1	P2 completes writing ∴ P1 can read
0	1	P1 starts reading ∴ P2 should not write
1	1	P1 completes reading ∴ P2 can write
X signifies don't care condition		

TABLE 5.1 : Communication protocol for processors of FIGURE 5.2

writing to another. The SN74LS670 components can be organised into a memory of up to 512 words of any number of multiples of 4 bits. The fast access time (typically 20 ns) and tri-state output makes this type of component ideal for use in intermediate memories. The organisation of an ISMI of 256 x 8-bit words is shown in Figure 5.3. The scratchpad of this size requires 128 chips of SN74LS670, ten multiplexers and four driver chips. For communication in both directions between two processors, a pair of ISMIs is required.

Several reasons can be numerated for the choice of an ISMI to couple two processors. Although the use of DMA channels can be envisaged for communication between two processors, we find that DMA transfer requires extra complex interface circuitry with synchronisation logic. Compared to this, the use of ISMI avoids the need for such a complex interface and also has the advantage of asynchronous communication. The use of ISMI also frees the DMA channels of the processors to be connected to peripheral devices, for which they are more suitable. The use of ISMI gives much more flexibility especially when designing multimicroprocessor systems.

5.5 COMMON MEMORY

A Common Memory (CM) providing a data store which is accessible by several processors at the same hierarchical level is an important feature of the HMSU. Because the processors are operating asynchronously, it is necessary to ensure that only one processor attempts to access the

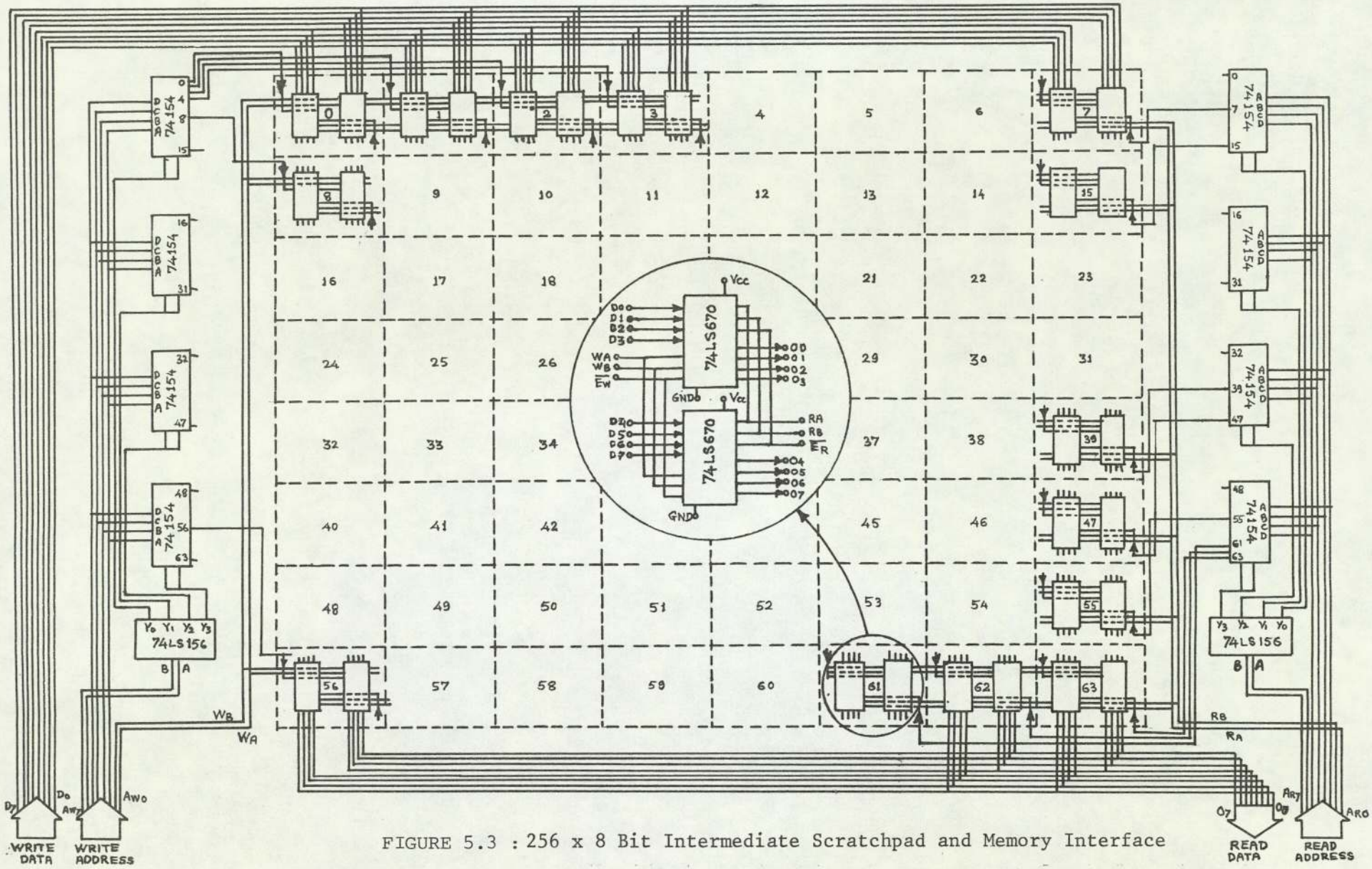


FIGURE 5.3 : 256 x 8 Bit Intermediate Scratchpad and Memory Interface

memory at a time. This is achieved by a master-slave organisation, the master processor having the task of allocating access to the CM to slave processors as required. This type of organisation is quite common; see, for example, Russo (1976) and Witten and Jenkins (1978), although other possible structures have been described (Enslow, 1976). The master-slave structure has been chosen for the HMSU because of the simplicity of both hardware and software required, and because of its applicability to asymmetric systems in which the workloads of master and slave processors are appreciably different.

Figure 5.4 illustrates a master-slave configuration in which eight processors are used. In this particular design, the processors are Fairchild/Mostek F8 chips, chosen largely because of locally available software. The configuration allows the master processor always to have access to the CM by setting up its own address code on its address port. When the master processor decides to grant access to any of the slave processors, it will output the address code for the particular slave processor on the address port. This address code will link the CPU READ and $\overline{R/W}$ lines of the slave to the CM via a multiplexer. At the same time, the demultiplexer unit opens the appropriate buffers to link the internal address and data buses of the slave to the external address and data buses of the CM. The demultiplexer unit also sets up an external interrupt for the slave which will activate its common memory access program.

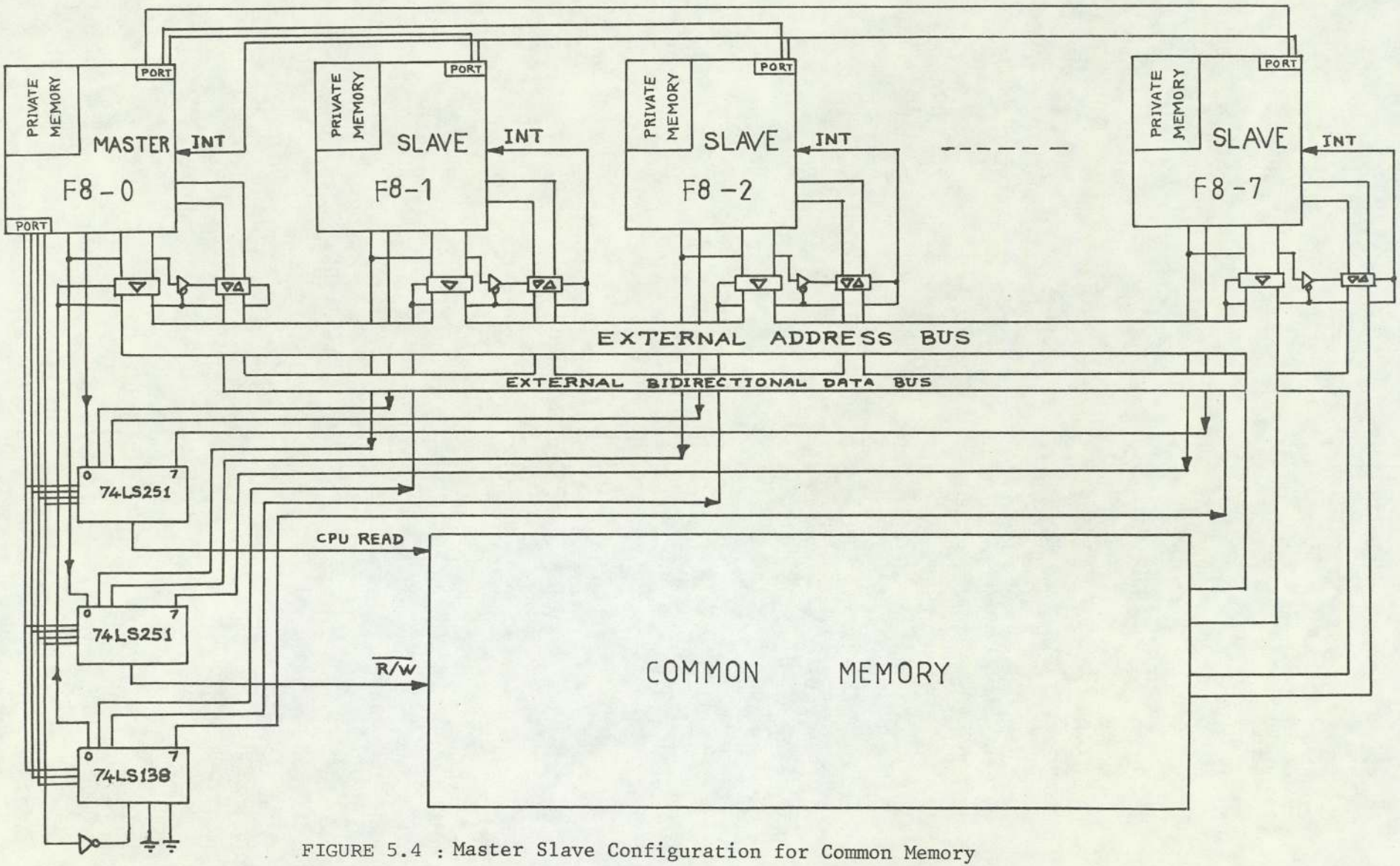


FIGURE 5.4 : Master Slave Configuration for Common Memory

At the end of the CM access routine, the slave processor will signal to the master via the master's external interrupt line. As an additional check, it is possible for the slave to send an identification code to one of the master's I/O ports. This will enable the master to recognise which slave has finished a CM data transfer, which may be useful if there is a queue of CM access requests. This additional check is not necessary to the system, however.

The HMSU architecture is arranged so that all CM access requests are generated by the master processor; the slaves do not need to generate such requests themselves. This avoids the problem of concatenation. When the master does not need to ask for any slave to transfer data to or from the CM, it can treat the CM as its own private memory, transferring data in and out at any such time.

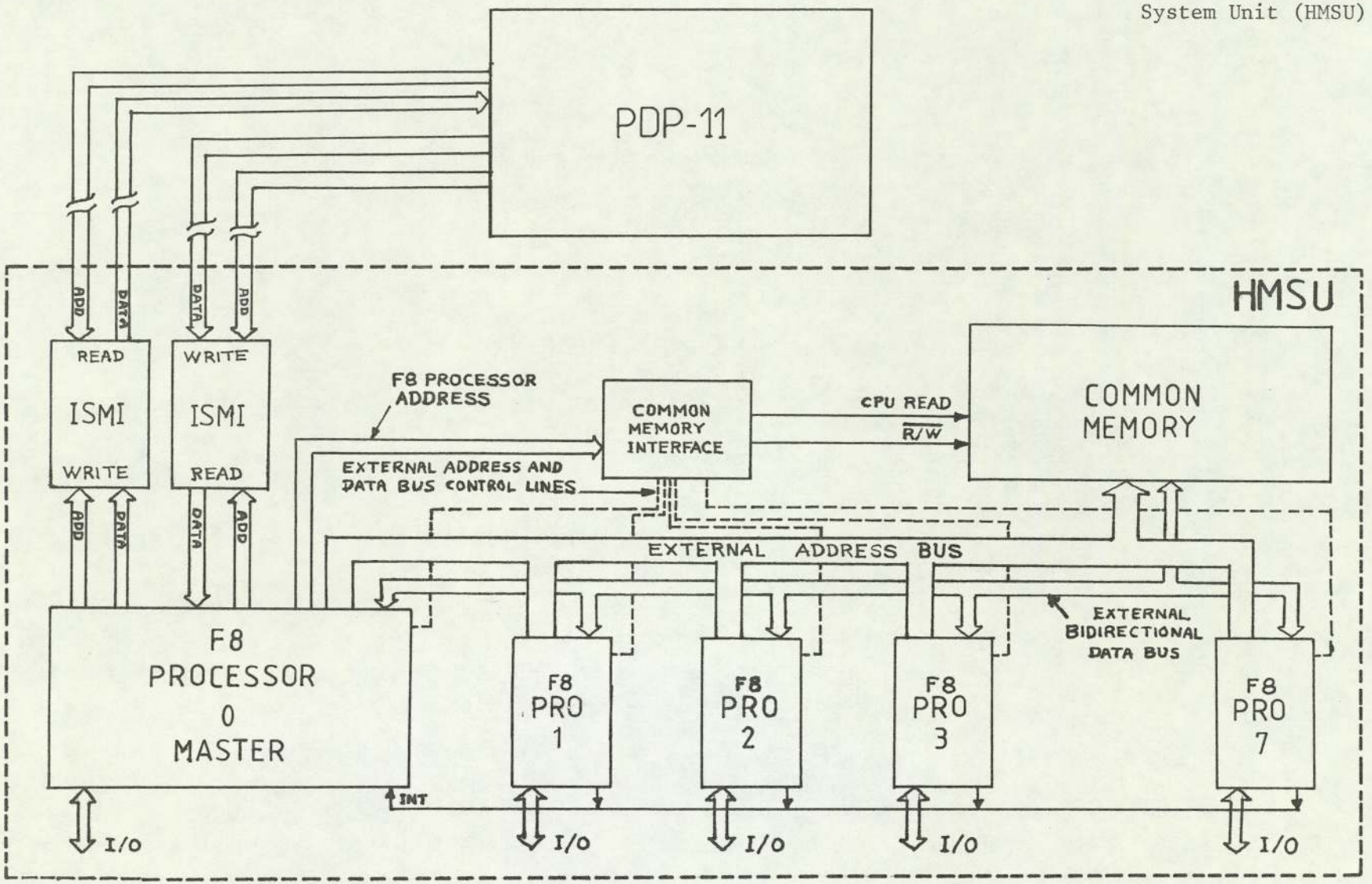
5.6 HMSU ARCHITECTURE

The hardware of the HMSU is very simple. It consists of a master processor, a pair of ISMIs serving as a bi-directional data transfer interface, a common memory as described above, and a number of slave processors. The organisation of the HMSU is shown in Figure 5.5.

The master processor performs the following sytem functions:

1. Receiving of data from the supremal level host computer via the ISMI and transfer of that data to the Common Memory.

FIGURE 5.5 : Hierarchical Microprocessor System Unit (HMSU)



2. Generating CM access requests to allow the slave processors both to receive data from the CM and to transmit data gathered or generated by each slave to the CM.

3. Transmitting slave-generated data from the CM back to the supremal computer, again via the ISMI.

In addition to these system tasks, the master processor may be assigned some user tasks if the processing load allows. It is desirable, however, to share the total processing load as equally as possible between all the processors, otherwise the overall system performance may become degraded.

The slave processors perform individual user tasks, gathering process data and implementing control functions via their own individual interfaces and I/O ports. The proposed structure of the HMSU incorporates up to seven slave processors. However, more than seven slave processors can be used if desired, or alternatively, more than one HMSU can be employed. This distributed processing structure makes the system very flexible and easily expandable.

A further possibility offered by the HMSU architecture is that the supremal computer may be used to change the allocation of tasks amongst the slave processors in the event of a hardware failure, a facility which makes the system of very high integrity.

5.7 HMSU STRUCTURE

A variety of structures can be developed using HMSUs. Since a pair of ISMIs may be used to link any two processors operating asynchronously, we may use this interface to link two HMSUs together via their master processors, or to

link a master processor to a particular slave processor either within the same HMSU or in a different one. This gives very great flexibility in the design of multi-processor structures.

Figures 5.6, 5.7 and 5.8 show three different systems constructed from HMSUs each of which has a master and three slave processors. Figure 5.6 shows a hierarchical structure using three HMSUs and a supremal host computer. Figure 5.7 has four HMSUs in a star formation around the host computer and Figure 5.8 shows five HMSUs and a host computer in a ring formation. There are endless permutations on these three structures: the decision as to what type of structure to use is dependent entirely on the application.

5.8 CONCLUSIONS

A hierarchical organisation of microprocessors, known as the HMSU, has been described. This sytem may be constructed at low cost from standard LSI components. The use of private memories for each processor combined with intermediate memory for interprocess communication avoids the synchronisation problems often associated with multi-processor systems and allows great flexibility in designing large-scale systems based on a number of basic HMSU blocks.

The main disadvantage of the system at present is a rather high chip count. This would be considerably reduced if the ISMI were implemented on a single LSI or VLSI chip, which is technologically, if not economically, possible.

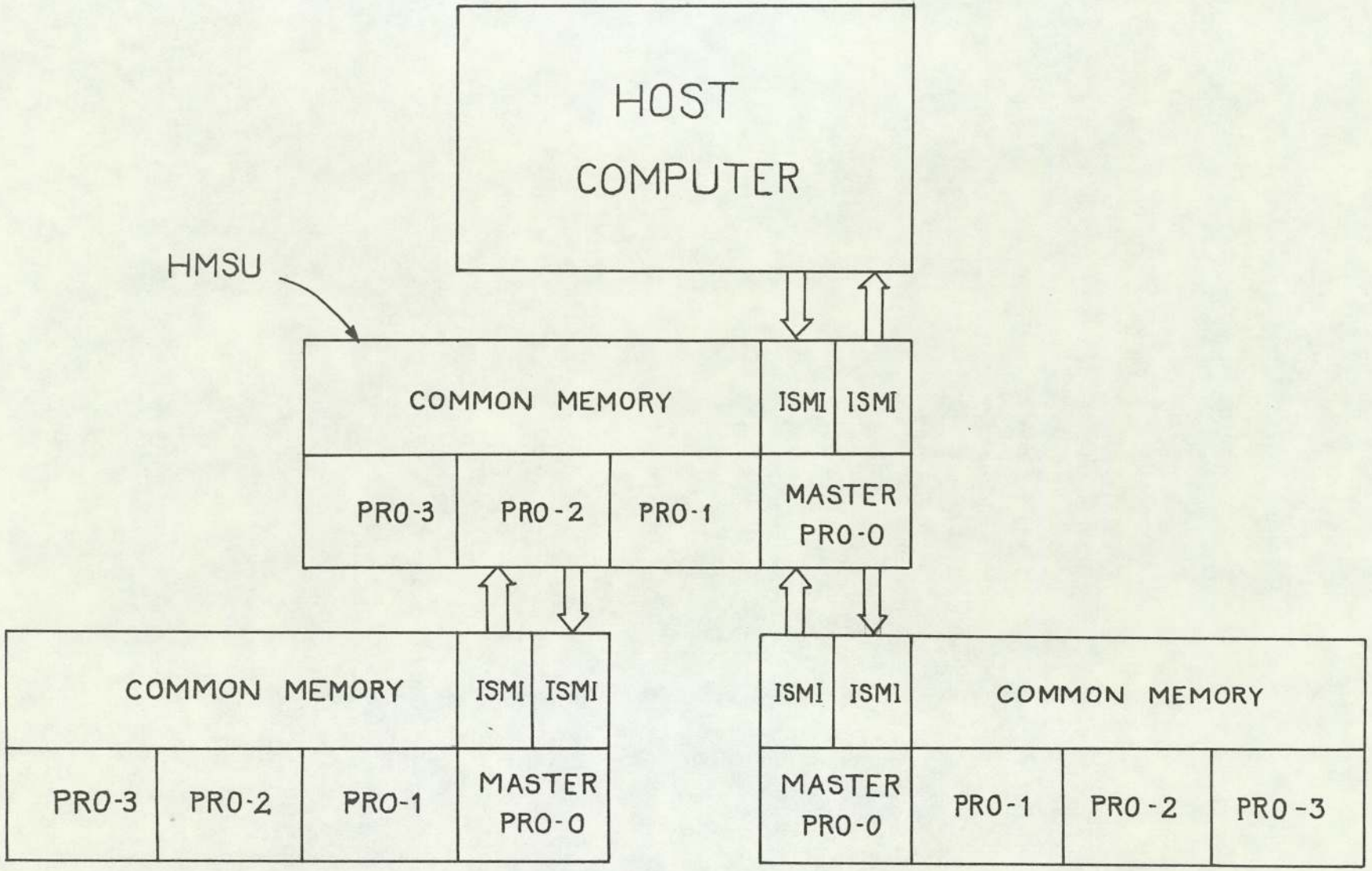


FIGURE 5.6 : HIERARCHICAL STRUCTURE

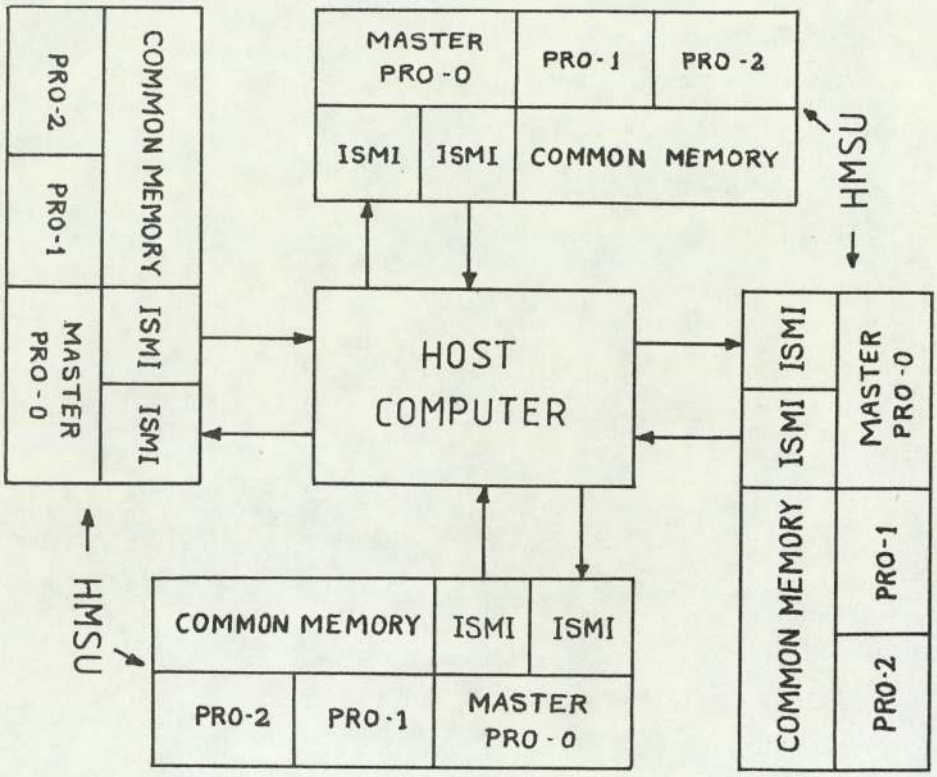


FIGURE 5.7 : STAR STRUCTURE

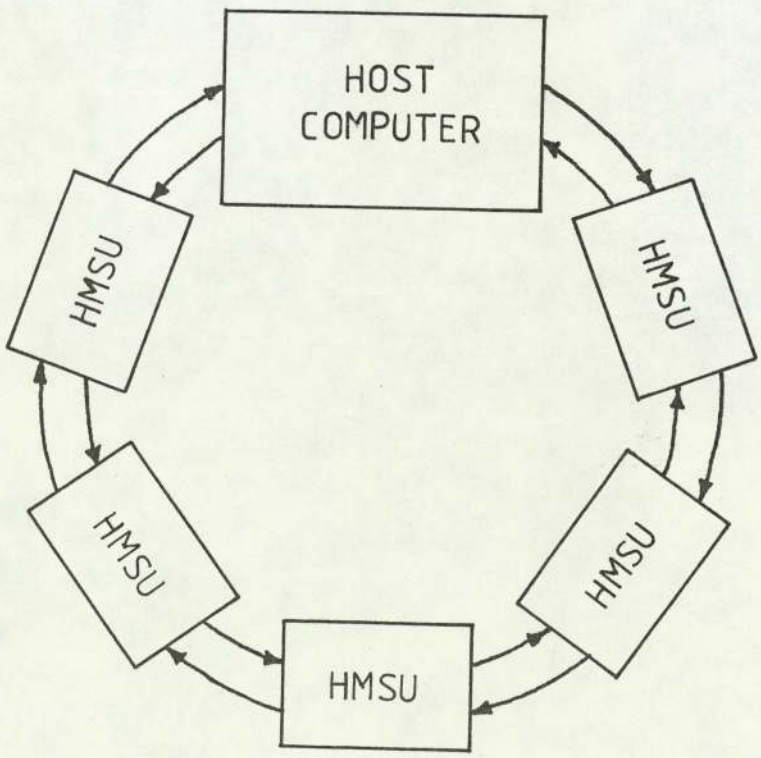


FIGURE 5.8 : RING STRUCTURE

With the present trends of new technologies, increasing reliability and falling costs, system integrity may be enhanced by the use of multiple hardware, and the HMSU is fully capable of providing a high integrity system once the necessary diagnostic software has been developed. A further advantage of the HMSU structure is that it allows modular development of software for each individual processor within the structure. The HMSU structure is specially useful for large-scale systems where a large system problem can be subdivided into smaller subsystem problems. Individual processors in the HMSU can be employed to these smaller subsystem problems and co-ordination for these problems can be achieved by a host computer. Other potential applications for HMSU can be homogeneous or heterogeneous.

There are several further research issues to be pursued such as employing a hardware arbiter for allocating common memory rather than total access control of common memory by a master processor, deadlock problems associated with the hierarchical systems and overall system performance.

CHAPTER 6 - CONTROL OF A TRAVELLING LOAD FURNACE

6.1 INTRODUCTION

The HMSU, as discussed in Chapter 5, was designed in the first instance as part of a programme of work on the control of an 8-zone travelling load electrical billet reheating furnace. A travelling load furnace (TLF) constructed in the Department of Systems Science of The City University is basically a laboratory version of industrial TLFs designed to carry out computer control experiments (Caffin, 1972; Sheena, 1977).

Various schemes exist which may be used to control the TLF. Some are simple to implement and require minimal amounts of information about the properties of the plant, while some are sophisticated and optimal in performance but require detailed knowledge of the process and the plant, its inputs and disturbances. A simplified empirical model was developed and tested for the heating of slabs of metal in a multizone TLF (Caffin, 1972), whereas Sheena (1977) implemented and tested the PID algorithm and the on-line least square identification and control schemes.

This chapter briefly describes the TLF and discusses an incremental form of PID control scheme with reference to the control requirements needed to interface the HMSU to the TLF. The design details of the electronic interface and modifications to the existing interface needed for this purpose are also given.

6.2 FURNACE DESCRIPTION

The TLF consists of a 2.7 metre long tunnel with a number of separately controlled electrical heaters distributed along its length and a conveyor carrying blocks of metal (loads) through it. The furnace is designed to heat loads to temperatures up to 500^{o}C. The conveyor is driven by a DC motor. Each of the eight heating zones is powered by two banks of three 1 kw electric fire elements, giving a total furnace power of 48 kw. The zone lining walls are made of aluminium reflectors which are air-cooled and the sections near the heaters (top and bottom surfaces) are water-cooled. Detailed specification of the furnace and the interface with the computer may be found in Caffin (1972).

A schematic diagram of the furnace interfaced with the HMSU and the PDP-11/10 minicomputer is shown in Figure 6.1. The hierarchical computer control strategy using HMSU and the PDP-11/10 minicomputer is explained later in the chapter. The normal operation of the furnace consists of the loads travelling through the heating zones and recycling them after suitable cooling. (A water shower is built outside the furnace if forced cooling is required.) The positions of the loads in the furnace are tracked using a set of six microswitches that are closed by appropriately placed bolts on the conveyor. The interrupt signal generated by the closure of microswitches is processed by the computer.

The electric power input to each of the heating elements in the eight zones or to the speed of the DC motor is adjustable in small discrete steps from zero power to

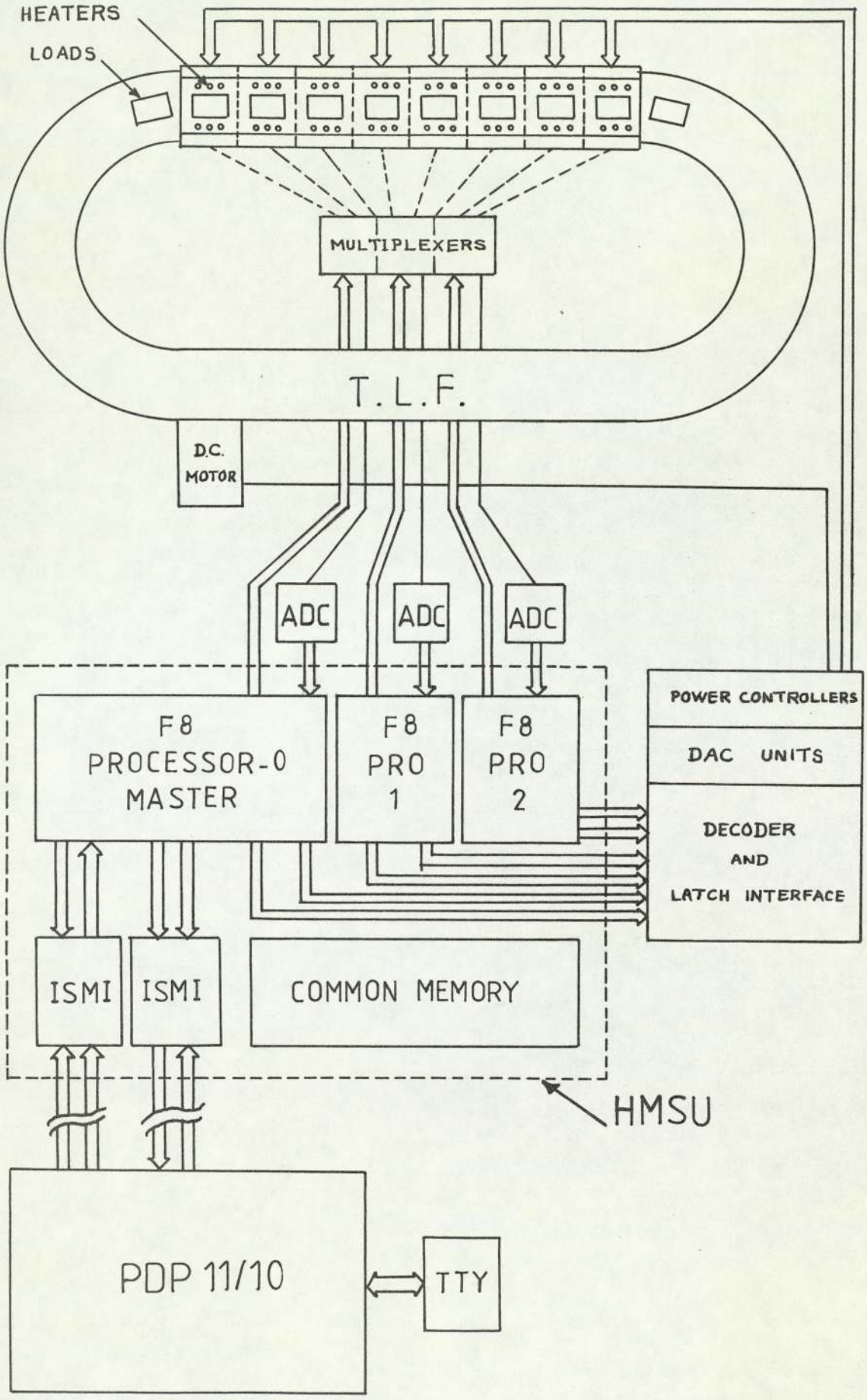


FIGURE 6.1 : The TLF interfaced with the HMSU and the PDP-11/10 minicomputer

full power or zero speed to maximum speed using silicon controlled rectifier power units. The control signals to these nine power units are generated by the computer and transmitted via DAC units.

The temperatures of the loads are measured using Chromal Alumel, stainless steel sheathed and magnesia insulated thermocouples inserted inside each individual load. However, this kind of arrangement for measuring temperatures is rather uncommon as compared with industrial practice. The analogue signals from the thermocouples are multiplexed, amplified and sent to the computer via an A to D converter.

6.3 PID CONTROL SCHEME

The most conventional form of controller used in the process industry is the three-term controller with the constituent terms: Proportional-Integral-Derivative action control (PID). Although PID is the most tried-out method of control, it was chosen for implementation within the processors of the HMSU because of its simplicity and as an experimental example.

The basic concept of PID feedback control is to use the error, the integral of the error and its rate of change between a measured variable and a set-point to generate a signal that actuates the control devices to influence the process so as to reduce this error to zero. The set-point may be truly constant or it may be a programmed profile generated by hardware or software. In the TLF, the zone

setpoints along the length of the furnace define the temperature profile required for the loads. A typical control loop for the measurement and control of temperature within a process plant (or equivalent representative of the TLF) is shown in Figure 6.2. The Figure shows that a set-point setting, sampling of error signal, its filtering and the PID algorithm implementation is performed in a digital computer.

The discrete PID algorithm is derived from the continuous form of the three-term control algorithm. For a continuously controlled process variable, the analogue control signal is given by:

$$p(t) = K\left(e(t) + \frac{1}{Ti}\int_{o}^{t} e(t)\,dt + Td\,\frac{de(t)}{dt}\right) \qquad 6.3.1$$

where $p(t)$ = Control signal at time t

$e(t)$ = Error between measured and set-point values

Ti = Integral time constant or reset time

Td = Derivative time constant or rate time

and K = Proportional gain.

Taking Laplace transform of the above equation becomes:

$$P(S) = K\left(E(S) + \frac{E(S)}{Ti\ S} + Td(S\ E(S) - e(0))\right) \qquad 6.3.2$$

Assuming at $t = 0$, $e(0) = 0$, equation 6.3.2 becomes:

$$P(S) = K\left(1 + \frac{1}{Ti\ S} + Td\ S\right)E(S) \qquad 6.3.3$$

If the error signal is filtered before the control signal is applied, the effects of noise originating from the process or instrumentation are reduced. However,

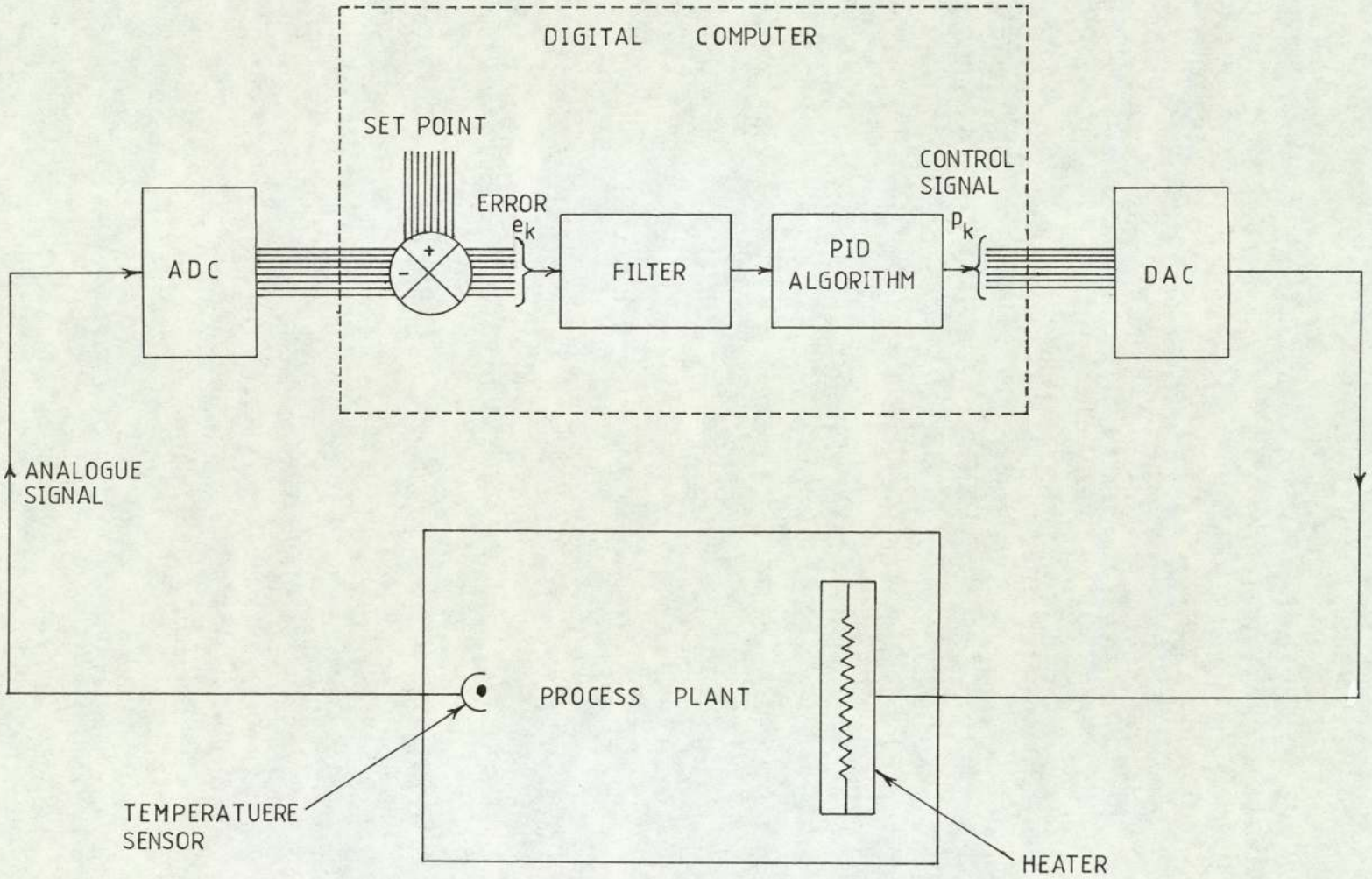


FIGURE 6.2 : A typical temperature control loop of a process plant

filtering will reduce to some extent the effectiveness of the derivative action. Using a simple first order filter with a transfer function as:

$$E(S) = \frac{\hat{E(S)}}{1 + Tf\ S} \qquad 6.3.4$$

where $\hat{E(S)}$ = Laplace transform of actual error signal

$E(S)$ = Laplace transform of filtered error signal

and Tf = Filter time constant.

Combining equations 6.3.3 and 6.3.4, we get

$$P(S) = \frac{K}{1 + Tf\ S}\left(1 + \frac{1}{Ti\ S} + Td\ S\right) E(S) \qquad 6.3.5$$

i.e.

$$P(S) + TfSP(S) = K\left(1 + \frac{1}{Ti\ S} + Td\ S\right)E(S) \qquad 6.3.6$$

The inverse Laplace transform of equation 6.3.6 gives:

$$p(t) + Tf\frac{d}{dt}p(t) = K\left(e(t) + \frac{1}{Ti}\int_0^t e(t)\ dt + Td\frac{d}{dt}e(t)\right) \qquad 6.3.7$$

For digital implementation, a discrete form of equation 6.3.7 is:

$$Pk + Tf\frac{(Pk - P_{k-1})}{\tau} = K\left(e_k + \frac{\tau}{Ti}\sum_{r=0}^{K} e_r + Td\frac{(e_k - e_{k-1})}{\tau}\right) \qquad 6.3.8$$

where K = Sampling instant

e_k = Error signal at the Kth sample interval

Pk = Control signal at the Kth sample interval

τ = Time interval between samples.

Equation 6.3.8 for the (K-1)th sample interval is:

$$P_{K-1} + Tf\frac{P_{k-1} - P_{k-2}}{\tau} = K\left(e_{k-1} + \frac{\tau}{Ti}\sum_{r=0}^{K-1} e_r + Td\frac{e_{k-1} - e_{k-2}}{\tau}\right) \qquad 6.3.9$$

Subtracting equation 6.3.9 from equation 6.3.8 and simplifying gives:

$$\Delta Pk = \frac{K\tau}{Tf+\tau}\left[(e_k-e_{k-1}) + \frac{\tau}{Ti}e_k + \frac{Td}{\tau}(e_k-2e_{k-1}+e_{k-2})\right] + \frac{Tf}{Tf+\tau}\Delta P_{k-1} \quad 6.3.10$$

where

$$\Delta P_k = P_k - P_{k-1} \quad 6.3.11$$

Equation 6.3.10 may be written as:

$$\Delta P_k = K_1 e_k + K_2 e_{k-1} + K_3 e_{k-2} + K_4 \Delta P_{k-1} \quad 6.3.12$$

where

$$\left.\begin{aligned} K_1 &= \frac{K\tau}{Tf+\tau}\left(1 + \frac{\tau}{Ti} + \frac{Td}{\tau}\right) \\ K_2 &= -\frac{K\tau}{Tf+\tau}\left(1 + \frac{2Td}{\tau}\right) \\ K_3 &= \frac{K\,Td}{Tf+\tau} \end{aligned}\right\} \quad 6.3.13$$

and

$$K_4 = \frac{Tf}{Tf+\tau}$$

Equation 6.3.12 is simpler in arithmetic form than equation 6.3.10 but a selection of the values of Ki(i = 1,2,3 and 4) which will suit the process plant for tuning of the PID algorithm is very difficult. However, since these values of Ki are determined by equations 6.3.13, the operator has a convenient choice for the values of K, τ, Ti, Td and Tf, with which he is much more familiar in terms of a feel for process control. Equations 6.3.11, 6.3.12 and 6.3.13 are used for implementation in the software. It is worth noting that all the control loops of the furnace share the same PID algorithm but each control loop has its own set of parameters, past errors and control signals.

6.3.1 *Control requirements for the HMSU*

One of the main objectives of implementing the HMSU to interface with the TLF is achieved by splitting up the control task into smaller tasks so as to allow parallel processing and distributed control. For this purpose, the furnace has been considered as being divided into three areas having 2, 3 and 3 heating zones respectively and referred to as the preheat, heat and soak sections. This is shown in Figure 6.3. Initially, the master processor of the HMSU is assigned to the preheat section which controls two control loops for the zones 7 and 6, whereas the slave I and slave II processors are assigned to the following heat and soak sections which control each of the three control loops for the zones 5, 4, 3 and 2, 1, 0 respectively. A temperature profile is defined by the set-point temperatures for each of the three sections.

The division of the control tasks for each of the processors in the HMSU and the PDP-11/10 minicomputer are set out as follows:

1. Equations 6.3.11 and 6.3.12 are used for implementation in each of the processors of the HMSU, so that each processor behaves as a PID controller for the TLF.

2. An individual processor of the HMSU is responsible for measuring the load temperature by sampling at regular intervals via its ADC interface channel and sending appropriate control signals to the zone heaters via its DAC interface channel.

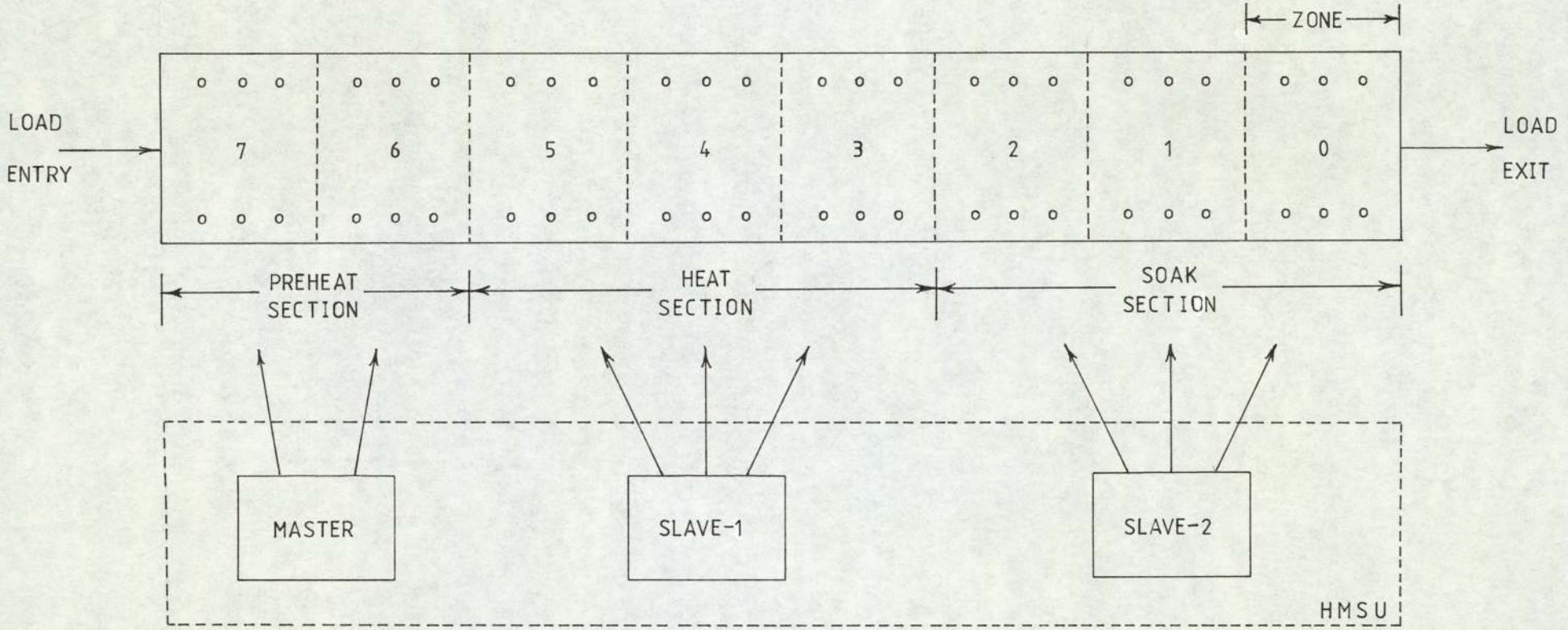


FIGURE 6.3 : Heating sections of the TLF

3. The master processor, additionally, is made responsible for data distribution to the slave processors via the common memory module, while the data collection (e.g. set-points, controller constants, sampling intervals, motor speed etc.) is performed from the PDP-11 computer via the ISMI module. The master processor also collects information data on the current measured temperatures of each billet and the power output to each zone of the furnace and returns it to the PDP-11 host computer again via the common memory and the ISMI. The master processor controls the speed of the conveyor which is maintained constant.

4. The PDP-11 computer implements equations 6.3.13 with a check on the suitability of the steady-state gain value which is derived from equation 6.3.12. Under the steady-state condition $\Delta P_k \simeq \Delta P_{k-1}$ and $e_k \simeq e_{k-1} \simeq e_{k-2}$. Therefore the equation 6.3.12 may be written as:

$$\Delta P_k = (K_1 + K_2 + K_3)\, e_k + K4 \Delta P_k$$

The steady-state gain is defined as $\Delta P_k / e_k$ as $t \to \infty$ i.e.

$$K_{SG} = \frac{\Delta P_k}{e_k} = \frac{K_1 + K_2 + K_3}{1 - K4}$$

From equations 6.3.13, it can be shown that $K_{SG} = K_T$. For steady-state value of the error to be zero, K_{SG} is required to be positive. This control requirement for the value of K_{SG} is verified by the operator before the controller constants K1, K2, K3 and K4 are passed onto the HMSU.

Another task of the PDP-11 computer is to communicate with the operator and manipulate the input information data in a suitable form and present it to the master processor of the HMSU for its distribution. The display of process generated data from the TLF is also performed by the PDP-11 computer via its GT42 display processor. More details of the PDP-11 tasks are covered in Chapter 8.

5. One important feature of a control requirement for the HMSU is the operation of the controllers by selection of a control mode from a set of three control modes. The three control modes are outlined in Table 6.1. The operator sets up a desired control mode which allocates specific groups of zones of the TLF to be under the control of specific processors of the HMSU. For example, under a control mode (v), the master processor controls zones 2,1,0; the slave I controls zones 7,6 and the slave II controls zones 5,4,3 and so on. Thus the three groups of zones of the TLF are transparent to control action from the processors. The importance of this feature is recognised when a switching of a control mode may be necessary in the event of a failure of a processor controlling a critical group of zones (e.g. a soak zone).

CONTROL MODE	MASTER	SLAVE I	SLAVE II
	ZONES	ZONES	ZONES
U	7,6	5,4,3	2,1,0
V	2,1,0	7,6	5,4,3
W	5,4,3	2,1,0	7,6

TABLE 6.1 : Control modes

6. As the loads travel through from one zone to another zone, the corresponding zone controllers need to update the load addresses. A load update signal provided by the closure of a microswitch is passed simultaneously onto each of the processors of the HMSU as an interrupt signal. A software routine implemented in each processor of the HMSU accounts for updating the load address simultaneously.

6.4 ELECTRONIC INTERFACE REQUIREMENTS

In order to interface the hardware of the HMSU to the TLF, a suitable electronic interface is required for each processor so that it can transfer data to and from the furnace. The data transfer is concerned with addressing zones, addressing thermocouples for temperature measurements and digital data representation of temperature signals. This is achieved by the input/output interface shown in Figure 6.4. It may be noted that since each processor of the HMSU behaves as a controller for the TLF, its input/output interface is identical to that shown in the Figure.

As mentioned earlier in Section 5.5 of Chapter 5, the processors of the HMSU are designed around the Fairchild/ Mostek F8 microprocessor chips set. The input/output interface in Figure 6.4 uses three eight-bit bidirectional ports (Ports 0, 1 and 8) of the F8 microprocessor. The port 0 is used to input an 8-bit equivalent of a temperature measurement, obtained via ADC82, unipolar analogue to digital converter. The same port is also used to output an 8-bit

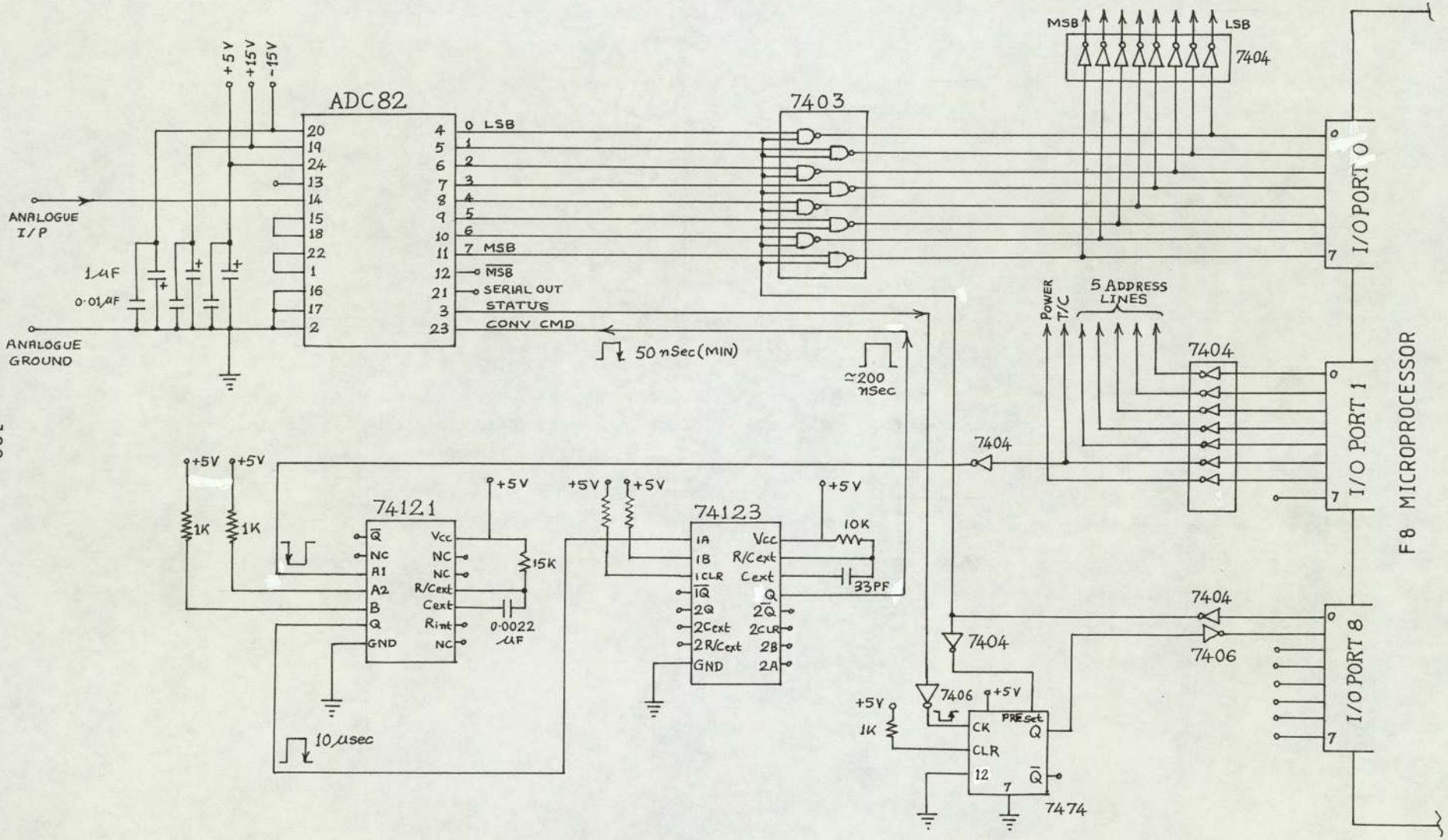


FIGURE 6.4 : Input/Output interface

equivalent of a control signal (P_k) to the TLF. A processor acquires a temperature measurement by setting up a thermocouple address at Port 1 (bits 0 to 5). This causes 74121 (monostable multivibrator with Schmitt-trigger input) and 74123 (retriggerable monostable multivibrator) circuits to generate a conversion command input for the ADC82. When the ADC82 completes the conversion of analogue signal, it generates a status signal which is input to 7474 (dual D-type positive-edge-triggered flip-flop with preset and clear inputs) via a 7406 (inverter). A change in the state at bit 1 of Port 8 caused by 7474 accounts for informing the processor to read the 8-bit equivalent of temperature via its Port 1 by opening the 7403 gates via its Port 8. A processor sends out 8-bit equivalent of a control signal (P_k) via its Port 0 prior to closing the 7403 gates and also after addressing the appropriate zone via its Port 1 (using bits 0 to 3 and bit 6).

6.5 MODIFICATION REQUIREMENTS TO EXISTING INTERFACE

The existing electronic interface allows for a single computer connection to be made to control the TLF, irrespective of any control scheme implementation. An ARGUS-500 process control computer was used by Caffin (1972) and Sheena (1977). This interface restricts the use of a multimicroprocessor system such as the HMSU to implement a distributed control scheme. Hence there is a need for interface modification. There are two main areas where these modifications are essential.

1. Firstly, since we want all the three microprocessors of the HMSU to make simultaneous temperature measurements of the load, a single ADC channel fails to satisfy this requirement. Hence, as pointed out in Section 6.4, three independent ADC channels are needed. In all 30 thermocouples, signals are multiplexed and the output analogue signal is passed onto the ADC unit (of the input/output interface) before its necessary amplification. Thus, for the three ADC channels, three analogue signals are required from three independent multiplexer units. This requirement is quite unique for this particular TLF because of the unconventional way in which the temperatures are measured.

2. Secondly, since the three microprocessors of the HMSU compute the actual power (control signal P_k) required for the zones they are assigned to control, no two microprocessors should be allowed to control a single zone via its DAC channel. However, although the assignment of which microprocessor will control which zone is done beforehand (by an operator's choice) a flexibility of a control of any of the eight zones by any of the three microprocessors is desirable. This requirement leads to a major modification in the existing interface.

Figure 6.5 shows some relevant details of the existing computer/furnace interface. Address lines 0 to 4 are used for either thermocouple addressing or zone addressing. When measuring a temperature, a thermocouple address and strobe (on address line 5) is latched by 9308 8-bit latch and is decoded by 9311 decoder (4 line to 16 line decoder/demulti-

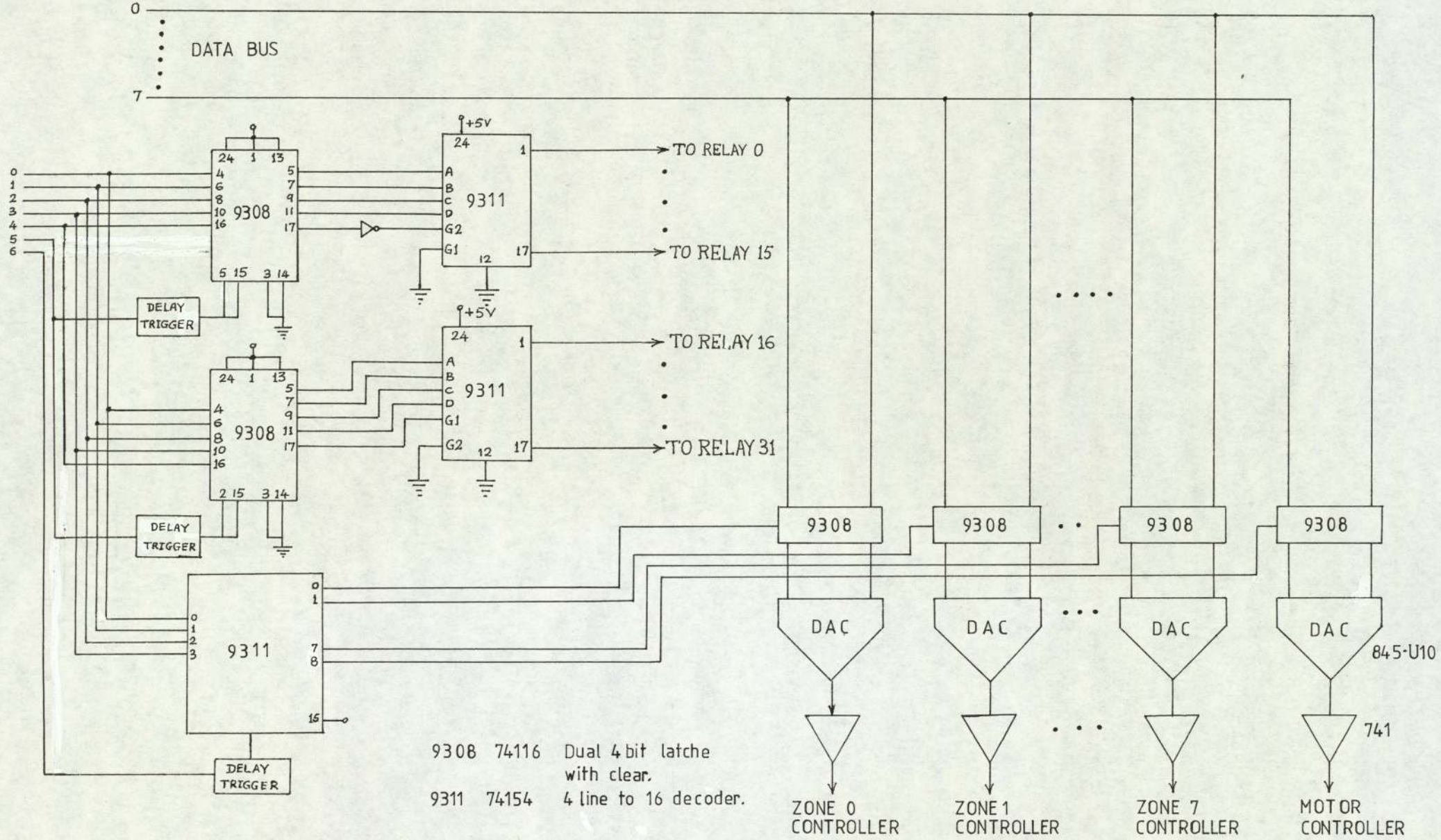


FIGURE 6.5 : Existing computer/furnace interface

plexer). The output lines of the decoder trip the appropriate reed relay for the temperature measurement. When an 8-bit equivalent of appropriate power (control signal P_k) is to be sent to a desired zone, that zone address and strobe (on address line 6) are decoded to enable the 9308 data latch of the DAC channel.

The reed relays used in the multiplexer unit are bulky devices and introduce noise in temperature measurement. The size of the multiplexer unit can be considerably reduced when a set of four, 4051 (single 8 channel) analogue multiplexers are used for multiplexing thermocouple signals. The proposed arrangement using these devices is shown in Figure 6.6. The thermocouple (T/C) address and the T/C strobe is latched by 74116 (Dual 4 bit) latch and is decoded further to activate the appropriate 4051 analogue multiplexer. The analogue signal is further amplified by a single stage 741 amplifier and passed onto the ADC. Three such analogue multiplexer units are needed for the three microprocessors.

Figure 6.7 shows a proposed (major) modification for the DAC channels of the zone controlling interface that would satisfy the second requirement. The Figure shows three data bus channels from the three microprocessors and the fourth data bus channel for the ARGUS 500 computer. All bus channels are buffered by 74LS241 octal tristate buffers. For every DAC channel there are four buffers, only one of which is enabled when a zone address appears on 74154 decoder (4 line to 16 line) from the corresponding processor. The output from the 7440 (NAND buffers) also enable the 9308

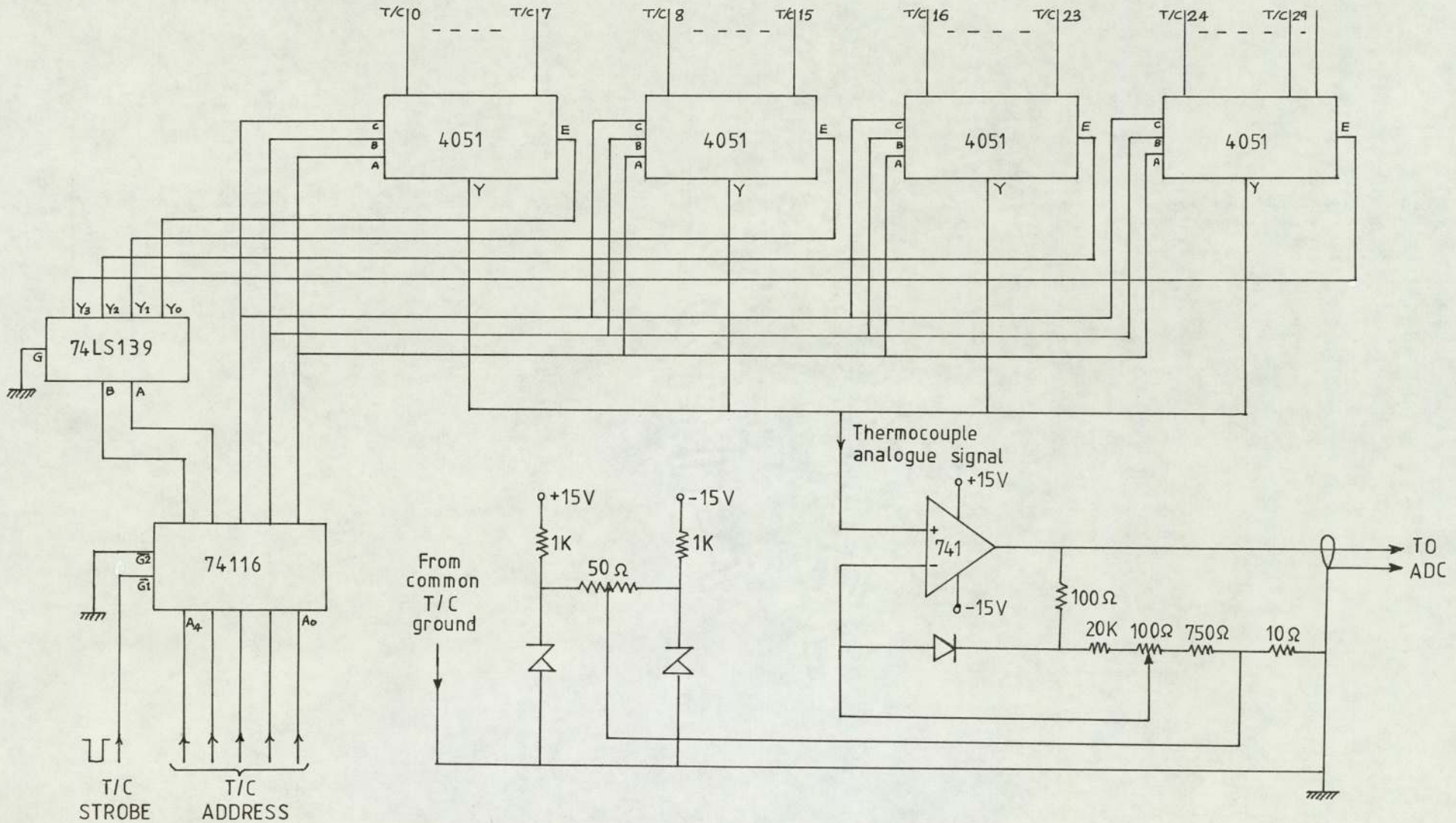


FIGURE 6.6 : Analogue multiplexer interface

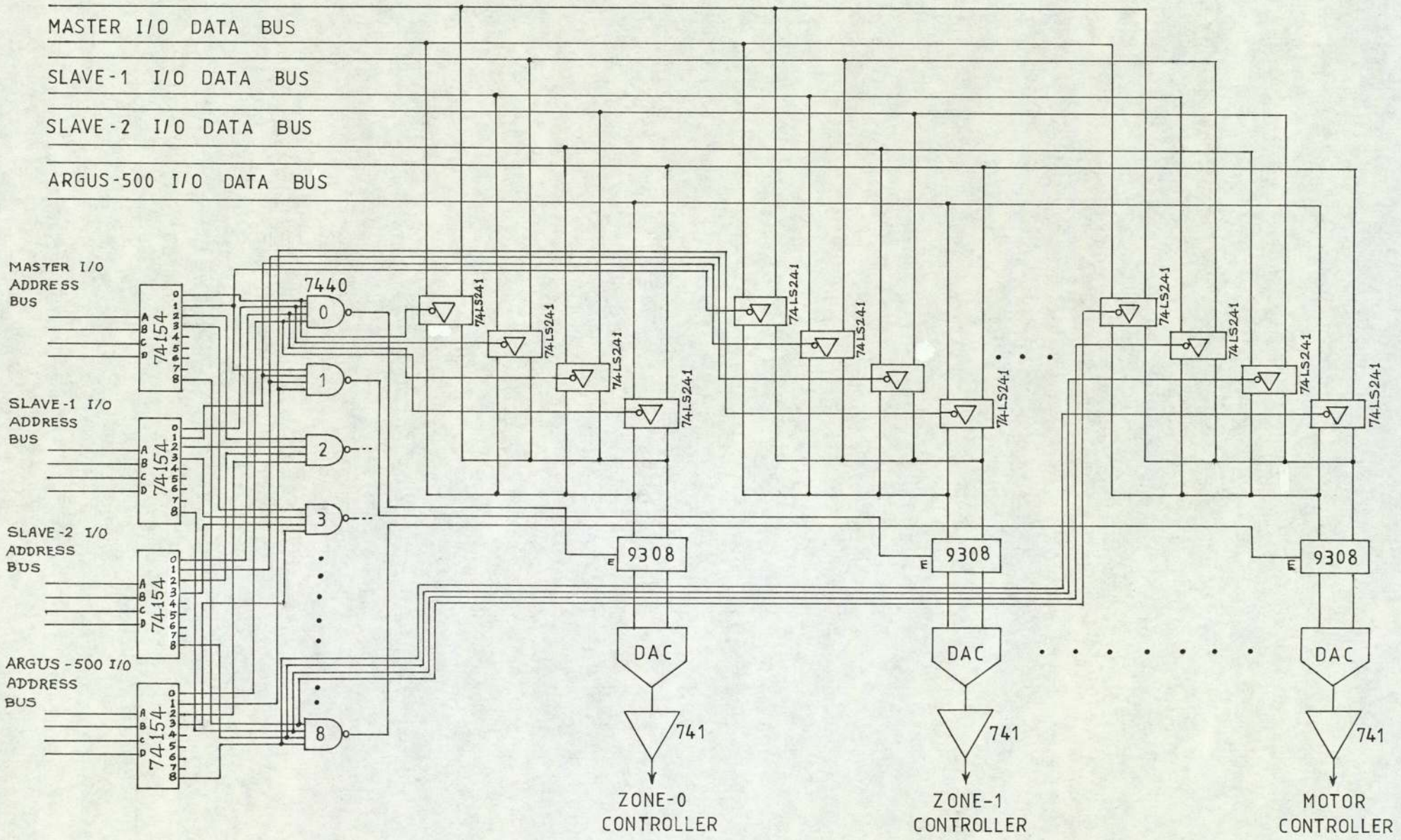


FIGURE 6.7 : Modification to DAC channels

8-bit latches at the same time so that the buffered data is latched for that particular DAC channel. Each DAC unit converts the latched digital data into analogue signal which is amplified by 741 operational amplifier and is responsible for controlling the level of heating inside the zone.

6.6 CONCLUSIONS

The TLF described in this chapter is a typical example of the kind of application selected for employing hierarchical computer control using a multi-microprocessor system such as the HMSU. The application clearly establishes the control requirements both at software level and hardware interface level. The implementation of the control requirements for the HMSU is a subject matter for Chapters 7 and 8. The overall control strategy used for the HMSU to control the TLF requires a major modification of the existing interface. This has remained in the proposal stage mainly because of the lack of suitable development and testing environment for the HMSU and hence is a topic of further investigation.

CHAPTER 7 - SOFTWARE DEVELOPMENT FOR THE HMSU SYSTEM

7.1 INTRODUCTION

The software for the HMSU mainly consists of independently stored programs, residing in PROMs of individual processors. For the purpose of this project, these programs are designed such that each processor within the HMSU behaves as a controller for the Travelling Load Furnace (TLF) described in the previous chapter. The processors of the HMSU execute their stored programs simultaneously. This accounts for various interactions between (1) the processors of the HMSU and the TLF, (2) the master and the slave processors and (3) the master and the host PDP-11/10 minicomputer. The software development for the HMSU to resolve these interactions is indeed a complex task. Other features of this software development task include:

1. the use of a low level programming language for the Fairchild F8 microprocessor.
2. programming for real-time operation
3. programming with due care for software dependency on hardware architecture
4. programming for a multi-level interrupt structure.

With reference to the above features, this chapter describes a program for the master processor of the HMSU. The program design is based upon the three major interactions outlined above. The entry and exit points for the flowcharts given in this chapter include, for ready reference,

the line numbers of the corresponding listing given in Appendix C. Furthermore, some details of programming features unique to the F8 microprocessor, and hardware details of the master processor, are covered in Appendices B and A respectively.

7.2 *SOFTWARE DEVELOPMENT AID*

In order to develop an object code program from a source code program, a need for a software development aid is of vital importance. A general program development procedure has been already outlined in Chapter 3. The program described in this chapter was initially developed on time-sharing, MAXI-MOP operating system for the ICL 1905E mainframe computer system. The F8 cross-assembler (Mk 3 version) made available by Davies (1977) was used to produce an object code program. The cross-assembler is a two-pass assembler and Figure 7.1 shows its general structure for producing TAPE and STOR subfiles from a source program subfile called PROG. The TAPE subfile may be used to produce a paper-tape version of the object code for loading into the target F8 microprocessor or for loading and testing it on a simulator. The STOR file contains the listing of the source subfile PROG and its corresponding object code.

During the course of development of software for the master processor of the HMSU system, the MAXI-MOP operating system and the ICL 1905E mainframe computer system were both withdrawn from service and, for this reason, a need arose to transfer and create new files onto another

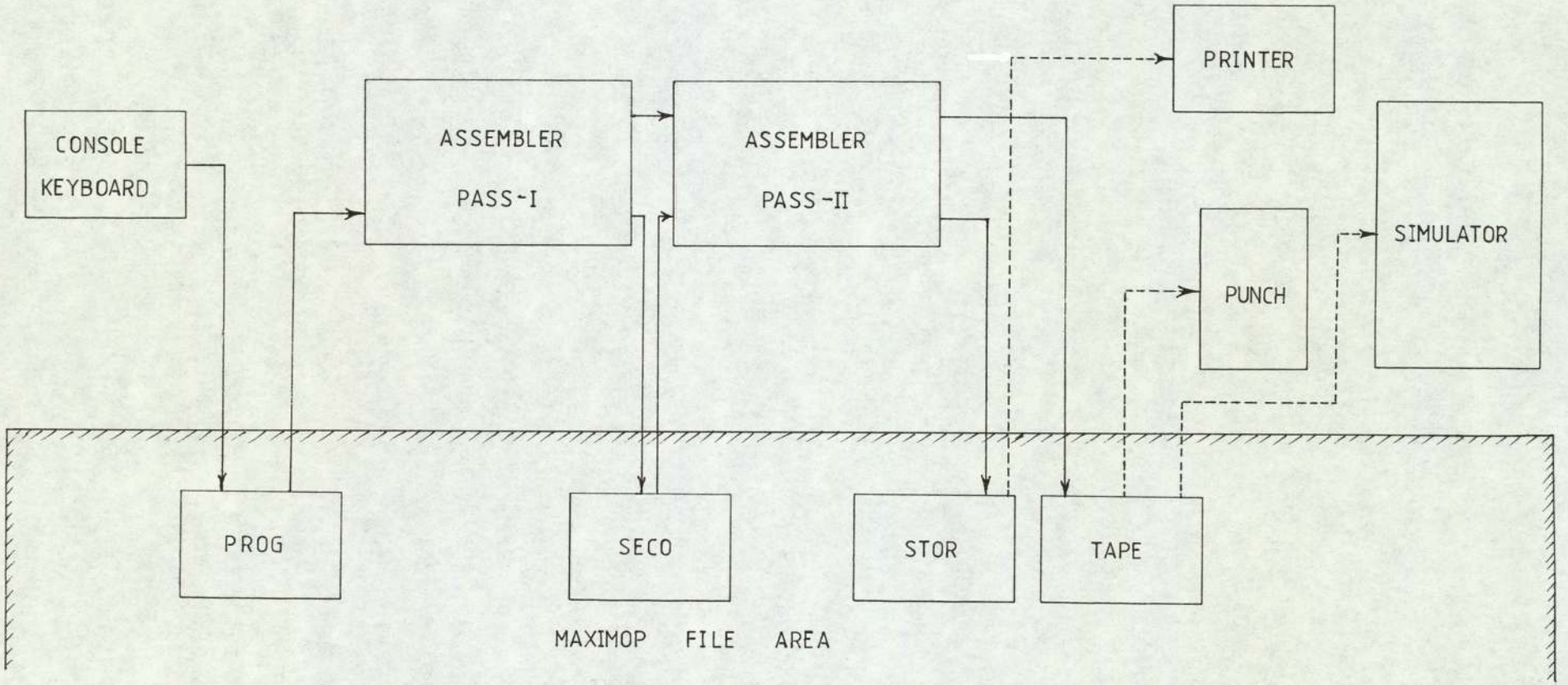


FIGURE 7.1 : The F8 cross assembler (Mk 3) structure

machine. The MOSTEK Z80 disk system (MDS Z80 system) was made available in the Electrical and Electronic Engineering Department and the F8 Assembler (F8XASM) available on this system was finally used for the program development described in this chapter. The program development was found to be more efficient than the MAXI-MOP system because the MDS Z80 system (1979) is a single-user system with facilities such as Editor, F8 assembler (F8XASM), linker etc. and a versatile operating system (OS). As the MDS Z80 system description and how to use it are given in the reference manual, no further discussion is made here. The object code of the master program called HMSU-SRC (Hierarchical Microprocessor System Unit-SouRCe) is produced in Intel format which is intended for loading into 2708 PROMs. The HMSU-LST contains the F8 assembly language source program and the corresponding object code generation, an alphabetical list of labels and their cross-references and the number of errors occurring during the assembly.

7.3 ASSUMPTIONS AND DEFINITIONS

Before continuing the discussion of the master program, it is worth mentioning various assumptions and definitions governing the program. The assumptions are as follows:

1. The hexadecimal number system is used to represent temperatures, digital control signal output, constants etc.

2. Arithmetic calculations are performed using two's complement so that H'00' to H'7F' represent the positive integers from 0 to 127 and H'FF' to H'80' represent the negative integers from -1 to -128.

3. Fixed point arithmetic is used.

4. The hexadecimal representation of temperatures is such that for every byte change, a change of $2^{o}C$ in temperature is obtained. Thus, for example, H'19' represents $50^{o}C$, H'32' represents $100^{o}C$ and so on.

The definitions used are as follows:

1. The PROM and RAM address ranges are defined as shown in Figure 7.2. The memory map for the slave processors is identical to that of the master while the memory map of the common memory is common to all three processors of the HMSU.

2. The 64 bytes of scratchpad registers available on the CPU (i.e. the F8 microprocessor chip), for each of the processors are defined to store transient data. The significance of this data carries specific interpretation and this is shown in Figure 7.3. For example, the control loop-1 in the Figure shows the use of eight registers (0'70' to 0'77') for storing temperature measurements, error signals, control signals etc. of the PID algorithm described in the previous chapter.

3. Each of the master and slave processors provides 14 ports, out of which six are input/output and the remainder are write only. The ports assignment is defined as shown in Table 7.1.

4. The zone addresses in term of hexadecimal numbers range from H'40' to H'47' for Zone '0' to Zone '7' respect-

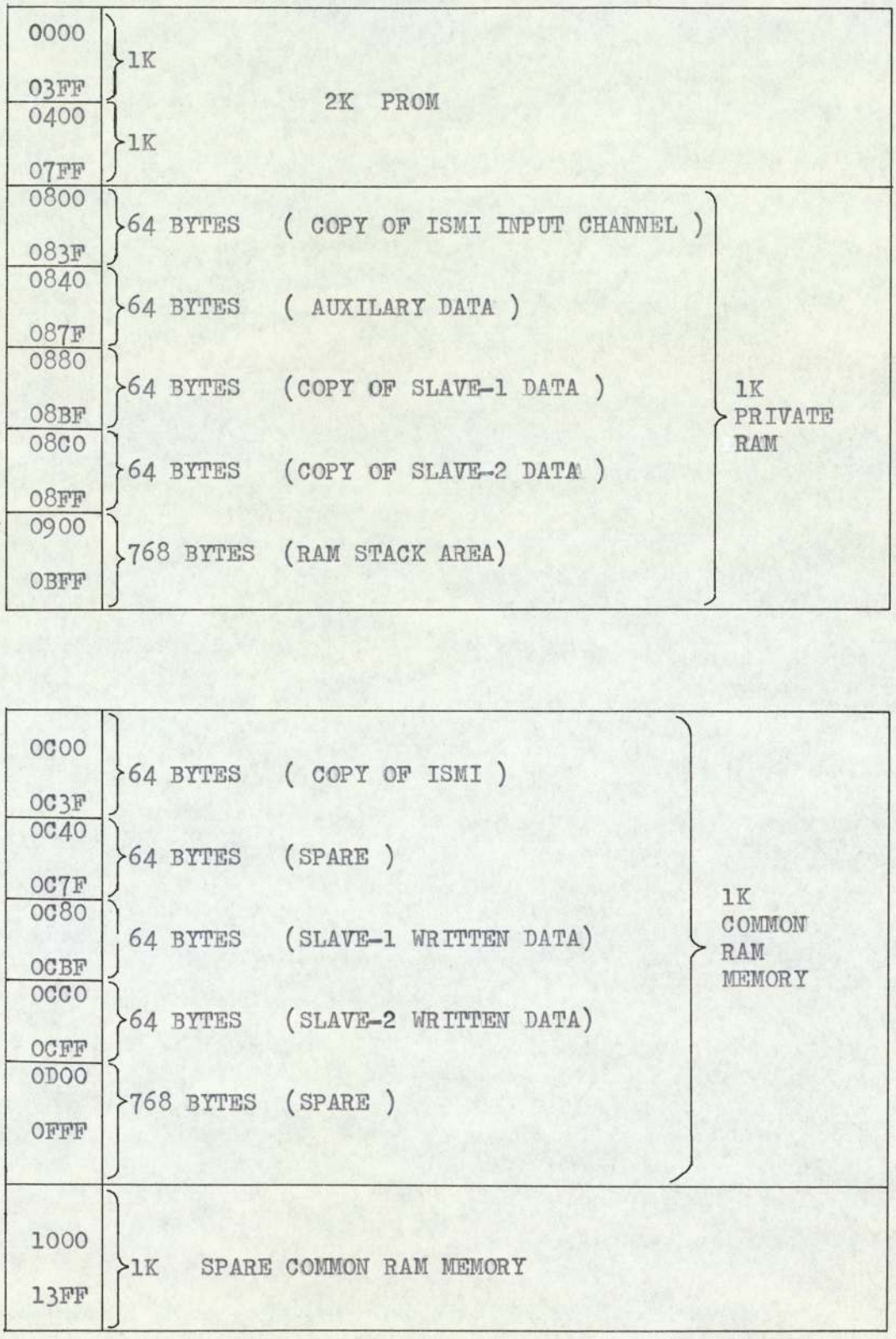


FIGURE 7.2 : Memory map of the master processor and common memory

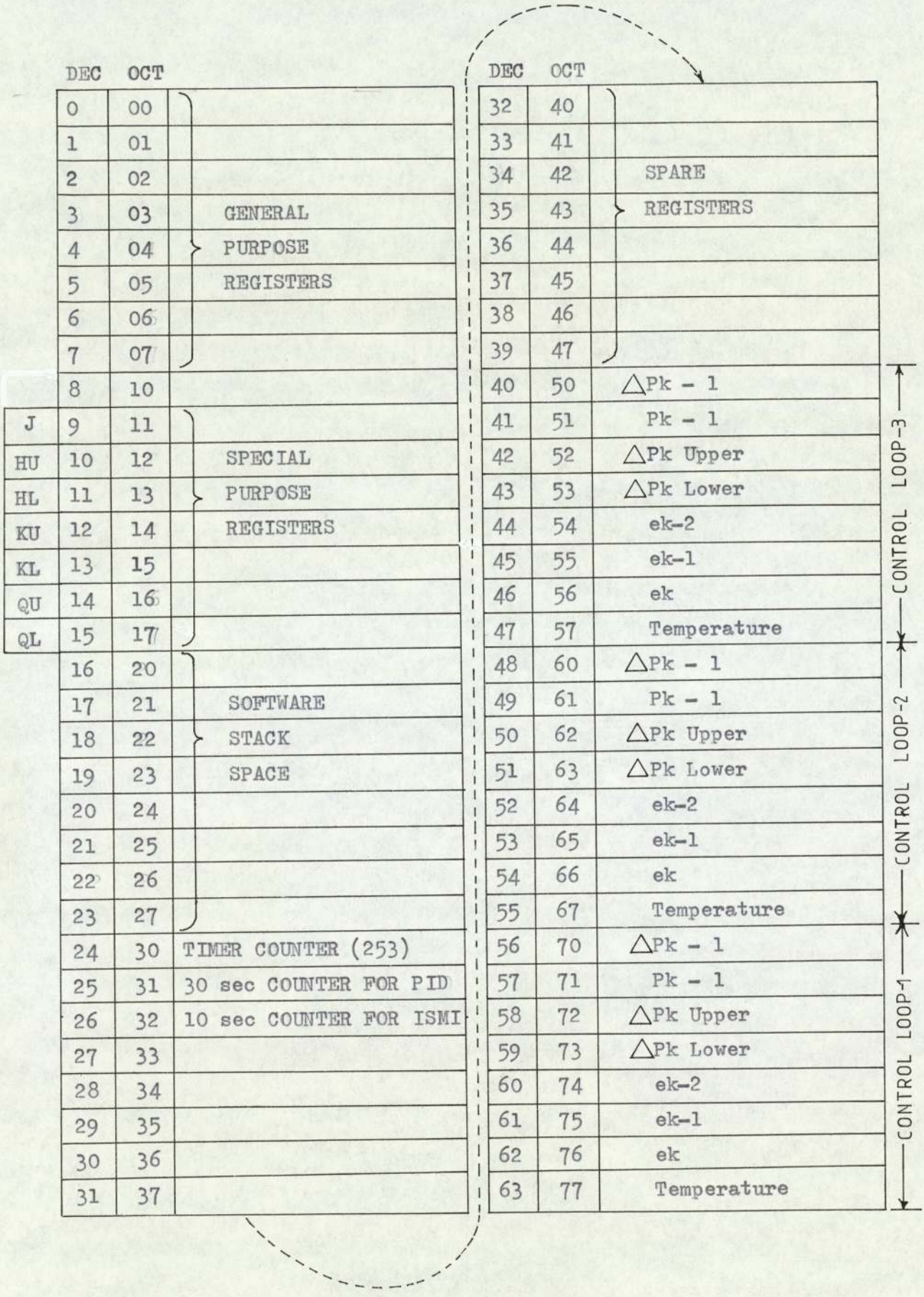


FIGURE 7.3 : Scratchpad memory map

CHIP TYPE	PORT NO	MASTER PROCESSOR	SLAVE I PROCESSOR	SLAVE II PROCESSOR
3850 CPU	0	Input from ADC and output data to the TLF (ref. Fig. 6.4)	Same as master	Same as master
	1	Zone and T/C addresses plus strobes (ref. Fig. 6.4)	-do-	-do-
3861 PIO (MK90002)	8	Bits 0 and 1 for I/O interface (ref. Fig. 6.4)	Same as master	Same as master
	9	Used for setting up slave addresses	Not used	Not used
	A	Interrupt control port (write only)	Same as master	Same as master
	B	Timer control port (write only)		
3861 PIO (MK90003)	20	ISMI interface	Not used	Not used
	21	ISMI interface		
	22	Interrupt control port (write only)	Same as master	Same as master
	23	Timer control port (write only)		
3853 SMI	C	Interrupt vector address upper byte (write only)	Same as master	Same as master
	D	Interrupt vector address lower byte (write only)	-do-	-do-
	E	Interrupt control port (write only)	-do-	-do-
	F	Timer control port (write only)	-do-	-do-

TABLE 7.1 : Ports assignment

ively. Similarly, the loads which travel through the TLF have address range from H'20' to H'3D' for Load 0 to Load 29. The motor which controls the speed of the conveyor has an address of H'50'.

5. The input and output channels of the HMSU which consist of ISMI memory modules account for data collection from and by the PDP-11 respectively. The 64 locations of each of the ISMI memory modules have write and read addresses. The read addresses range from H'40' to H'7F' and the write addresses range from H'80' to H'BF'. The memory locations of these modules are defined to store particular items of data. The labels of these data items and their storage locations are depicted in the matrix form shown in Figures 7.4 and 7.5. The significance of the labels used in these Figures and those used in the program, given in Appendix C, is given at the end of this chapter.

7.4 PROGRAM DESCRIPTION

The master program is only a one-third part of the overall software required for the three processors of the HMSU. However, its development is critical because the master processor behaves as a communicator with the slave processors, via common memory, and with the PDP-11/10 minicomputer, via ISMI memory modules. Thus, as far as the slave processors are concerned, their communication with the master via common memory is initiated by the master processor using external interrupts, whereas the master communicates with the PDP-11 at regular intervals using

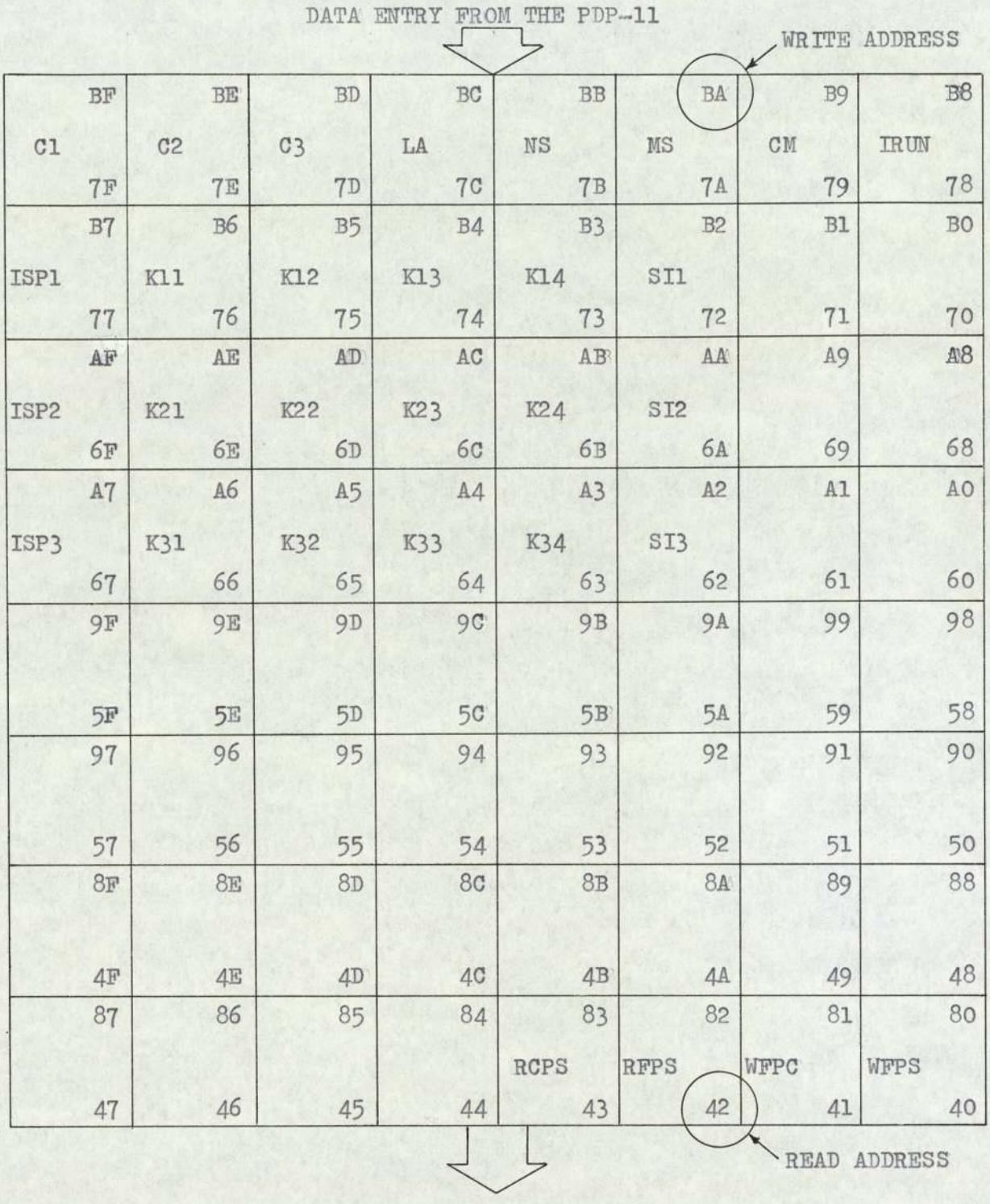


FIGURE 7.4 : ISMI memory map - input channel of the HMSU

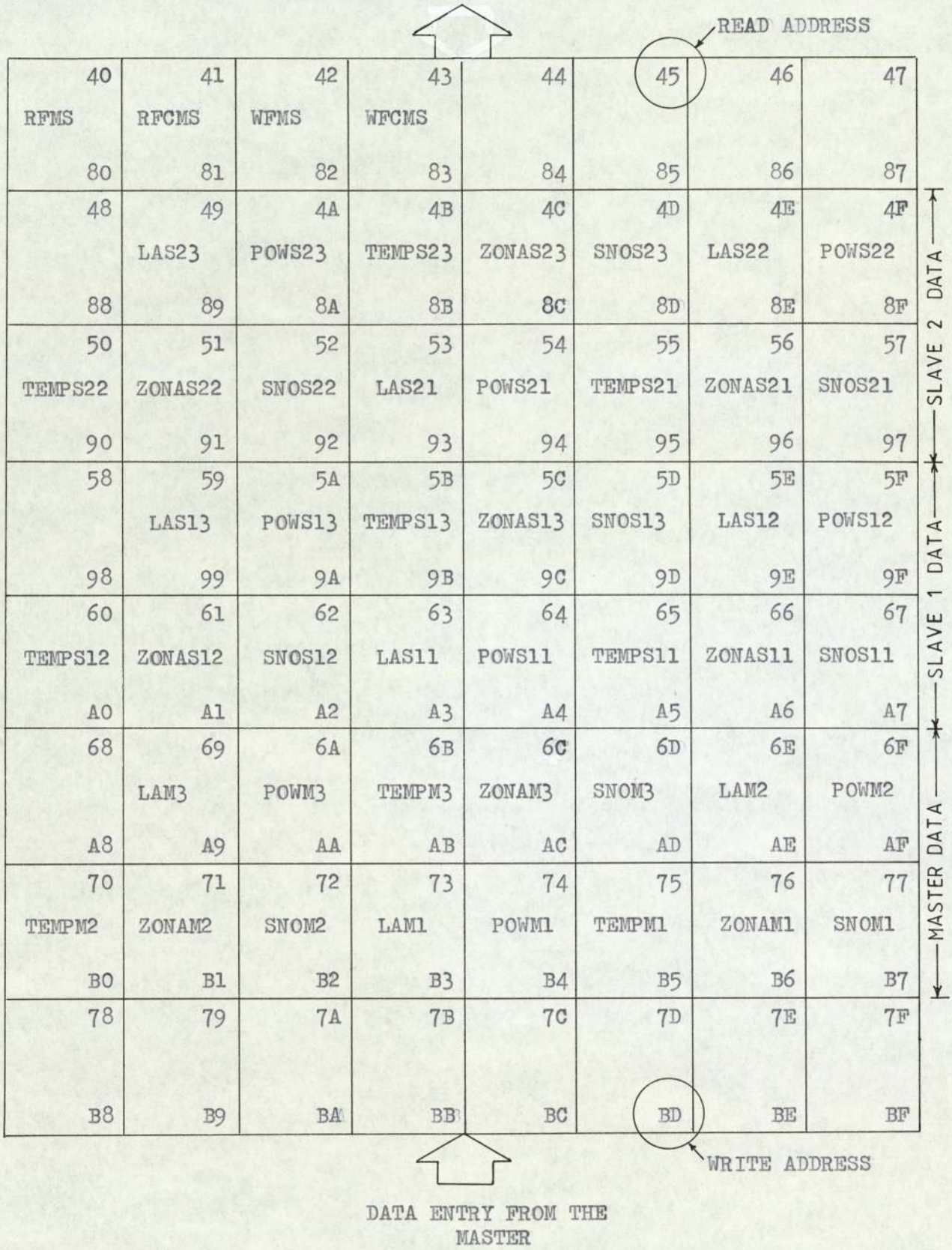


FIGURE 7.5 : ISMI memory map - output channel of the HMSU

real-time software interrupts. Thus the structure of the master program is based on the processing of various interrupts. Numerous routines are described in the following sections which handle interrupts, communication aspects of the master processor, its controller actions etc.

7.4.1 Interrupt Structure

The hardware architecture of the F8 processor provides interrupt handling capability using a serial priority network known as a "daisy-chain". The details of this are described in Appendix B. An interrupt structure with assigned priorities used in the master program is shown in Figure 7.6. The priorities are assigned as follows:

1. First priority: A timer available on the first PIO chip runs continuously and is used to count real time. The timer port of this chip is loaded with a maximum count of 253 so that this PIO chip pulls the $\overline{\text{INT REQ}}$ line low, every 3.953 milliseconds. Using this timer, a PID algorithm is entered after counting a time equal to the sampling interval, required for the measurement of temperature of the loads. The master processor also makes use of this timer to see if the PDP-11 has sent any new information for the controllers. This viewing process is performed at regular intervals, using the timer, say every 10 seconds for example.

2. Second priority: An external interrupt line available on the second PIO chip is used to inform the processor the load position within the zone which is under

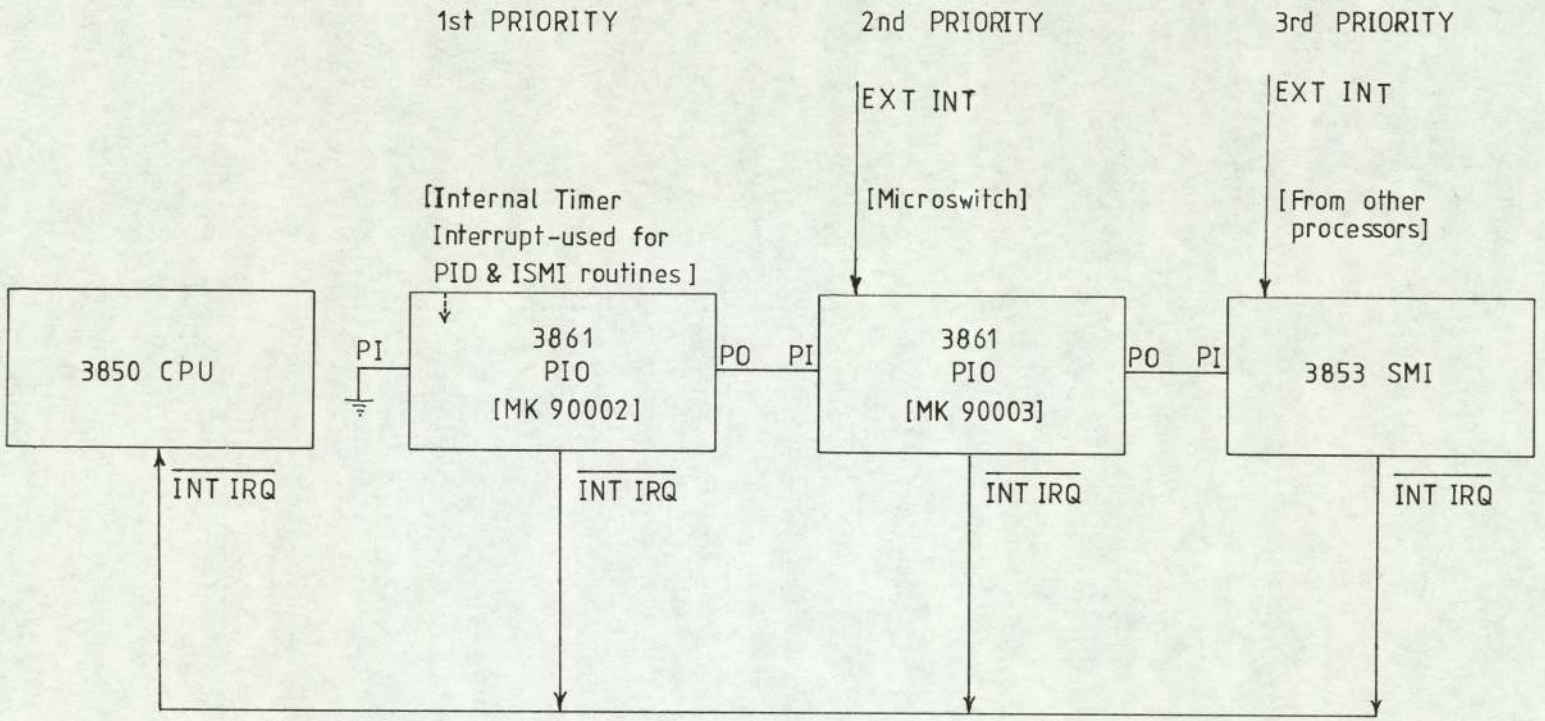


FIGURE 7.6 : Interrupt structure with priority

control. A signal coming from a microswitch, indicating the position of the load, pulls this line low to generate the external interrupt. A microswitch interrupt routine which is then entered allows the load addresses to be updated as the loads travel through from one zone of the TLF to the next.

3. Third priority: This priority level, which also uses the external interrupt line available on the SMI chip, is used for signalling the master processor that a particular slave processor has finished with the access of the common memory. The master processor, having received this interrupt, allows the next slave processor or itself to have access to common memory.

The interrupts generated by the PIO and SMI chips with the above-mentioned priorities need to be processed one at a time by the CPU. However, a possible occurrence of a higher priority interrupt causing a lower priority program execution to be interrupted needs careful handling. This issue is further complicated by the uncertainty with which these multi-level interrupts occur. Although serviceing a large number of interrupts with one CPU having a single hardware stack is inefficient (Fairchild, F8 users' guide, 1976), this problem can be overcome by using entry and exit protocols during interrupt serviceing.

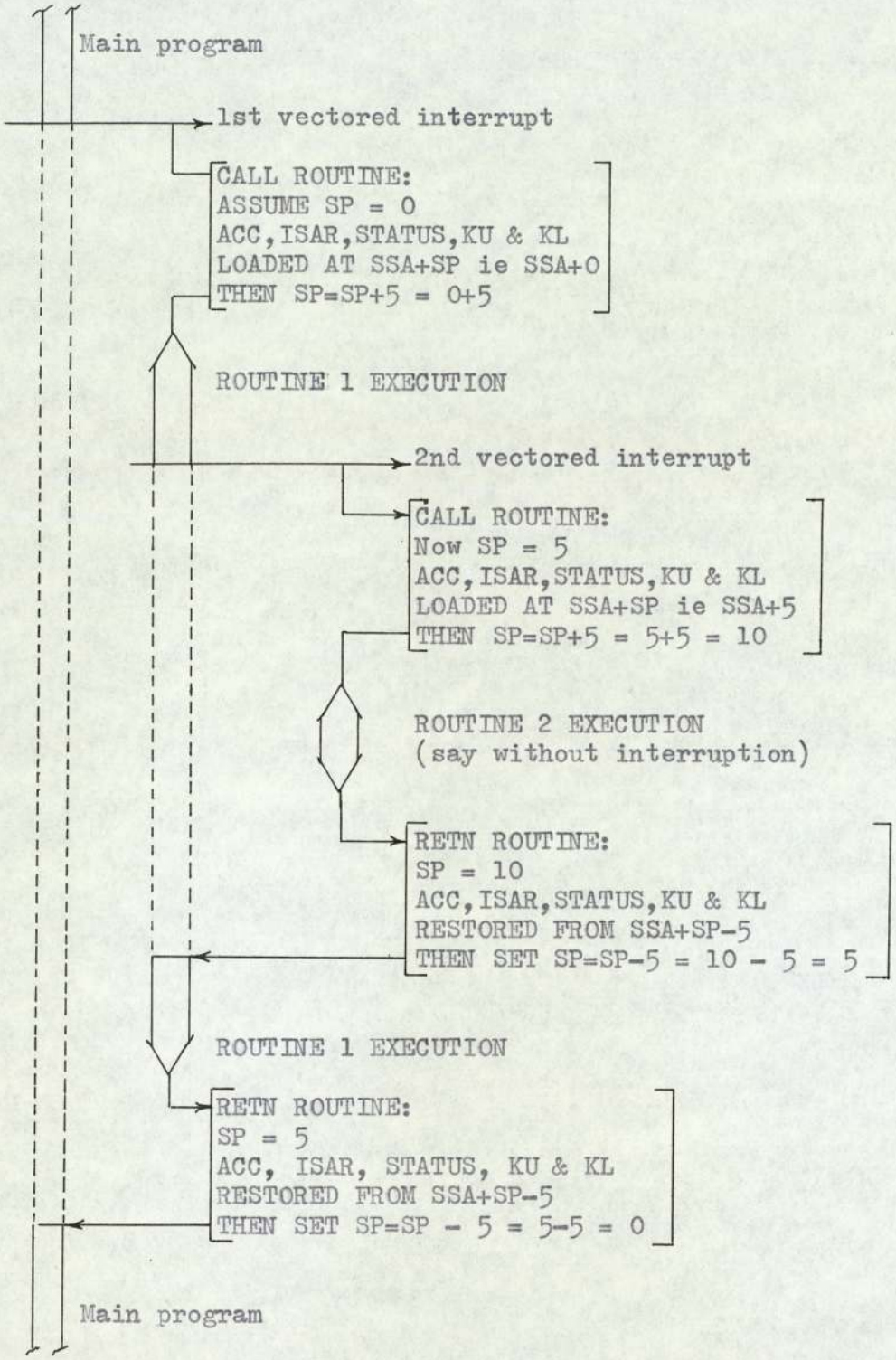
In order to handle these multi-level interrupts, two routines are implemented which use a RAM memory area. This area is used as a stack to store contents of the accumulator, ISAR (Indirect Scratchpad Address Register), status

register, and registers: DCO, DC1, KU and KL, when a current program execution is interrupted by a higher priority interrupt. This storing process is performed by a routine called CALL. When the execution of the higher priority interrupt routine is complete, the contents of the accumulator, ISAR, status register and registers: DCO, DC1, KU and KL are restored by using the RETN routine, so that the current program execution is resumed. A pointer is maintained in the scratchpad buffer area of the CPU, to point to the next empty stack area of the RAM memory. This pointer is incremented by nine locations at the end of the CALL routine and is decremented by nine locations at the end of the RETN routine. Furthermore, interrupts are enabled at the CPU after the CALL routine and disabled at the beginning of the RETN routine, and re-enabled according to the program execution which immediately follows. An example of a two-level priority interrupt structure is shown in Figure 7.7. The flow charts for the CALL and RETN routines are shown in Figure 7.19.

7.4.2 Initialisation

The master program is initialised at the beginning of the main program execution. The initialisation procedure is entered when the HMSU is switched on or by reset action. The following list shows the actions performed during the initialisation procedure:

1. Disable all interrupts
2. Clear all the input output ports



NOTE: 1) SSA = Starting Stack Address
ACC = Accumulator
ISAR = Indirect Scratchpad Address Register
KU = Upper byte of K Register
KL = Lower byte of K Register

2) ROUTINE 2 is of higher priority than ROUTINE 1

FIGURE 7.7 : Example of two-level priority interrupt structure

3. Clear the control loop buffers.

4. Switch off all the heaters of the TLF and switch off the motor of the conveyor by using the SHUT routine.

5. Clear 256 bytes of the private RAM where the input channel ISMI memory contents are to be copied.

6. Close the timer port and enable the external interrupt port of the second priority PIO chip.

7. Load port H'OB' of the first priority PIO with a 253 count and enable timer interrupts at this chip.

8. Load the SMI vector address ports H'OC' and H'OD' with H'O1' and H'FO' (i.e. vector address H'O1FO') and enable external interrupts at the SMI chip.

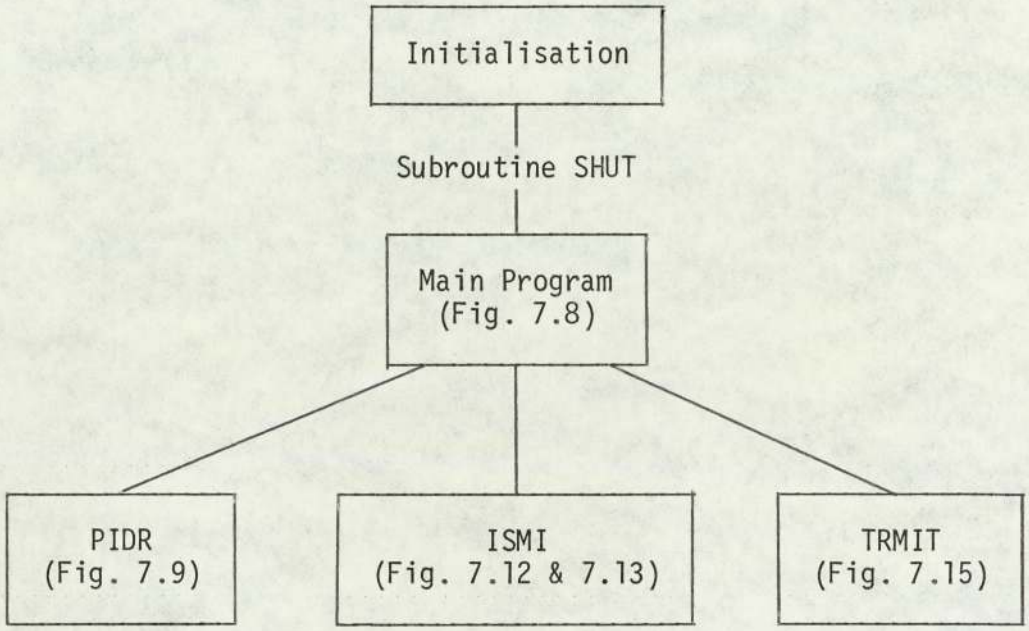
9. Load ISAR 30, 31 and 32 with counts 253, 30 and 10 respectively. These are the timer count, sampling interval count of 30 seconds and ISMI scan count of 10 seconds.

10. The PIDFLG (i.e. PID flag) and SNO (sample number) are cleared and ISMIFG (i.e. ISMI flag) is set.

7.4.3 The main program

The main program is basically a very short program in which the master processor loops around, checking if either the PIDFLG flag or the ISMIFG flag or the TRF flag is set. This program is executed at the lowest priority. The following tree structure shows how the main program is related to other subroutines implemented in Appendix C.

The flow charts of some specific routines such as INPU, OUTPU, COPY etc. are shown in the corresponding Figure numbers shown in brackets. In addition, the main program



Subroutines:

1. INPU (Fig. 7.10)
2. OUTPU (Fig. 7.11)
3. BSUBT - Binary subtraction
4. BMPY - Binary multiplication
5. TRAN - Register transfer
6. BADD - Binary addition
7. RECORD - Record control loop parameters
8. STOP - Stop or end of TLF run
9. MODLZA - Modify Load-zone addresses

Subroutines:

1. CLEAR - Clear Ports
2. COPY (Fig. 7.14)

Subroutines:

1. CLEAR - Clear Ports
2. WRITE - Write into output channel of ISMI

is mainly interrupted by the following;

1. Timer interrupt routine (Fig. 7.16): This routine as described earlier in the priority structure is responsible for counting real time and setting PIDFLG and ISMIFG

when the corresponding sampling intervals and ISMI scanning periods are completed.

2. External interrupt caused by the microswitch (Fig. 7.17): This routine is responsible for updating each load address as the loads travel through the TLF.

3. External interrupt caused by the other slave processors (Fig. 7.18): This routine, as mentioned earlier, facilitates access to the common memory by the processors of the HMSU under the supervision of the master processor.

In support of the above three interrupt routines, the interrupt structure demands the use of CALL and RETN subroutines, the flow-charts of which are shown in Figure 7.19.

7.5 CONCLUSIONS

The program described in this chapter applies to the master processor only. Similar program development is necessary for the Slave I and II processors except for the inclusion of the ISMI routines. Although the master program described here is produced with no assembly errors, the logical testing and debugging of the program on the actual hardware could not be performed due to lack of testing and debugging facilities. The cross-software development aid, such as using cross-assembler on the MAXIMOP system, for program development has its limitations and is inefficient for program development of multi-microprocessors. The scope for further development of the program is enormous; for example, performance evaluation, self-diagnosis of

hardware, failure detection and alarm condition signalling (i.e. fault-tolerance mechanisms) etc. requires further research.

LIST OF LABELS

FIGURE 7.4

C1, C2, C3 = Status of controllers 1, 2 & 3. It may be either ON or OFF.

LA = Load address.

NS = Number of samples.

MS = Motor speed of the conveyor.

CM = Control Mode. It may be UUU, VVV or WWW.

IRUN = Integer run number for the TLF.

ISP1, ISP2) ISP3) = Integer set point temperature for controllers 1, 2 & 3.

K11, K12, K13, K14 = Four controller constants for controller no. 1.

K21, K22, K23, K24 = Four " " " " no. 2.

K31, K32, K33, K34 = Four " " " " no. 3.

SI1, SI2, SI3 = Sampling intervals for controller no. 1, 2 & 3.

RCPS = Read count PDP set.

RFPS = Read flag PDP set.

WFCPS = Write flag count PDP set.

WFPS = Write flage PDP set.

FIGURE 7.5

RFMS = Read flage master set.

RFCMS = Read flage count master set.

WFMS = Write flage master set.

WFCMS = Write flage count master set.

SNOM1, SNOM2, SNOM3 = Sample number in control loop 1, 2 & 3 of the master.

SNOS11, SNOS12, SNOS13 = Sample number in control loop 1, 2 & 3 of the Slave I.

SNOS21, SNOS22, SNOS23 = Sample number in control loop 1, 2 & 3 of the Slave II.

ZONAM1, ZONAM2, ZONAM3 = Zone address in control loop 1, 2 & 3 of the master.

ZONAS11, ZONAS12, ZONAS13 = Zone address in control loop 1, 2 & 3 of the Slave I.

ZONAS21, ZONAS22, ZONAS23 = Zone address in control loop 1, 2 & 3 of the Slave II.

TEMPM1, TEMPM2, TEMPM3 = Temperature measurement in control loop 1, 2 & 3 for the master.

TEMPS11, TEMPS12, TEMPS13 = Temperature measurement in control loop 1, 2 & 3 for the Slave I.

TEMPS21, TEMPS22, TEMPS23 = Temperature measurement in control loop 1, 2 & 3 for the Slave II.

LAM1, LAM2, LAM3 = Load address in control loop 1, 2 & 3 for the master.

LAS11, LAS12, LAS13 = Load address in control loop 1, 2 & 3 for the Slave I.

LAS21, LAS22, LAS23 = Load address in control loop 1, 2 & 3 for the Slave II.

FIGURE 7.8

PIDFLG = PID control algorithm flag.

ISMIFG = ISMI memory (input channel) scan flag.

TRF = Transmit flage for data transmission to output channel of the ISMI.

PIDR = PID routine entry point.

ISMI = ISMI routine entry point.

TRMITT, TRMIT = Transmit routine entry point.

FIGURE 7.9

TSLA = Temporary starting load address.

SLA = Starting load address.

TSZONA = Temporary starting zone address.

SZONA = Starting zone address.

LZAC = Load-zone address counter.

L1, L2, L3 = Entry points for control loop 1, 2 & 3.

CALCU = Calculation of control signal using PID algorithm (entry point).

RECORD = Record routine that records the values of control loop calculations and measurements.

MODLZA = Modify load and zone addresses.

FIGURE 7.18

SPAO	= Store of port address for the master.
SPA1	= Store of port address for the Slave I.
SPA2	= Store of port address for the Slave II.
CMAR	= Entry point for the common memory access routine.
UUU	= Entry point for control mode UUU.
VVV	= Entry point for control mode VVV.
WWW	= Entry point for control mode WWW.

FIGURE 7.19

ISAR	= Indirect scratchpad address register.
CALL	= Entry point of CALL subroutine.
DCO	= 16 bit data counter register.
DC1	= 16 bit data counter stack register.
SP	= Stack pointer.
SSA	= Starting stack address in RAM memory area.
KU	= Upper byte of K register.
KL	= Lower byte of K register.

FIGURE 7.15

MTRF	= Master transmit flag.
S1TRF	= Slave I transmit flag.
S2TRF	= Slave II transmit flag.
TRFC	= Transmit flage count.

FIGURE 7.17

MSCNT	= Microswitch counter.

NOTE

It is important that the critical parts of the various programs (eg. Figure 7.12) are made interrupt proof. For instance, important flags should be tested before interrupts are enabled.

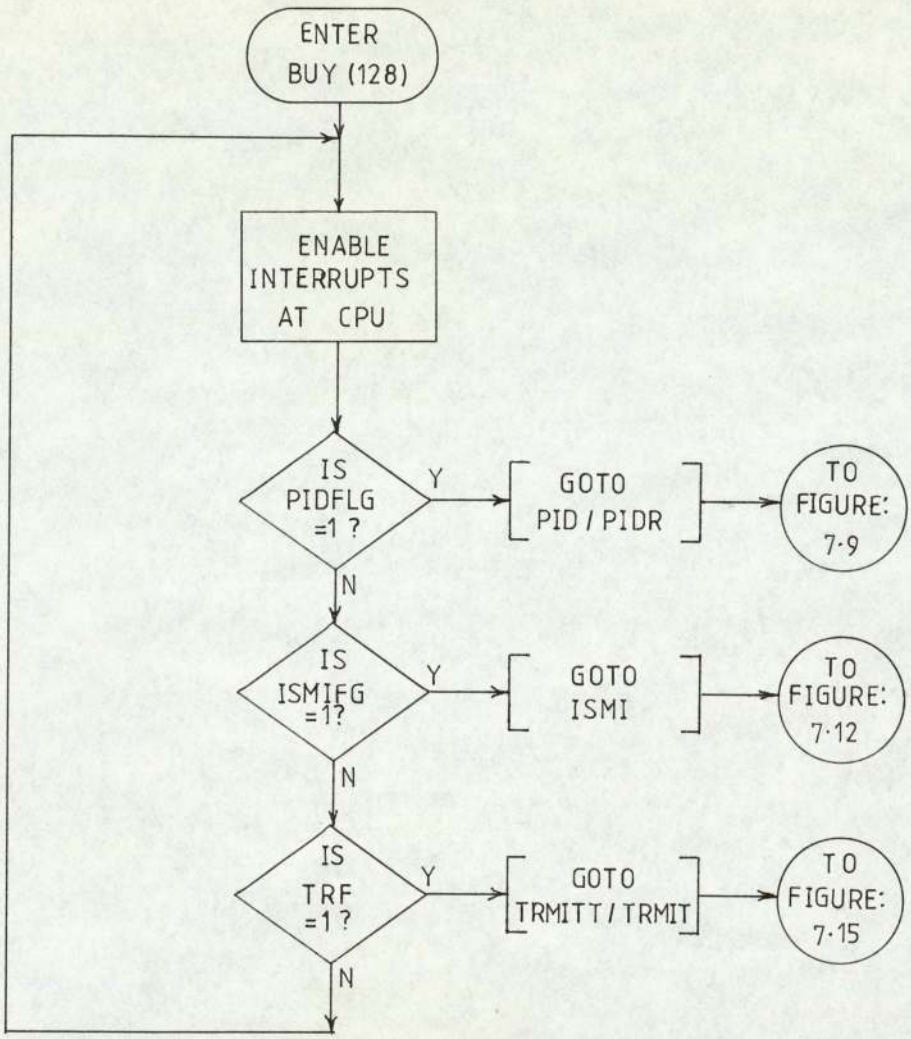


FIGURE 7.8 : Main program of the master processor

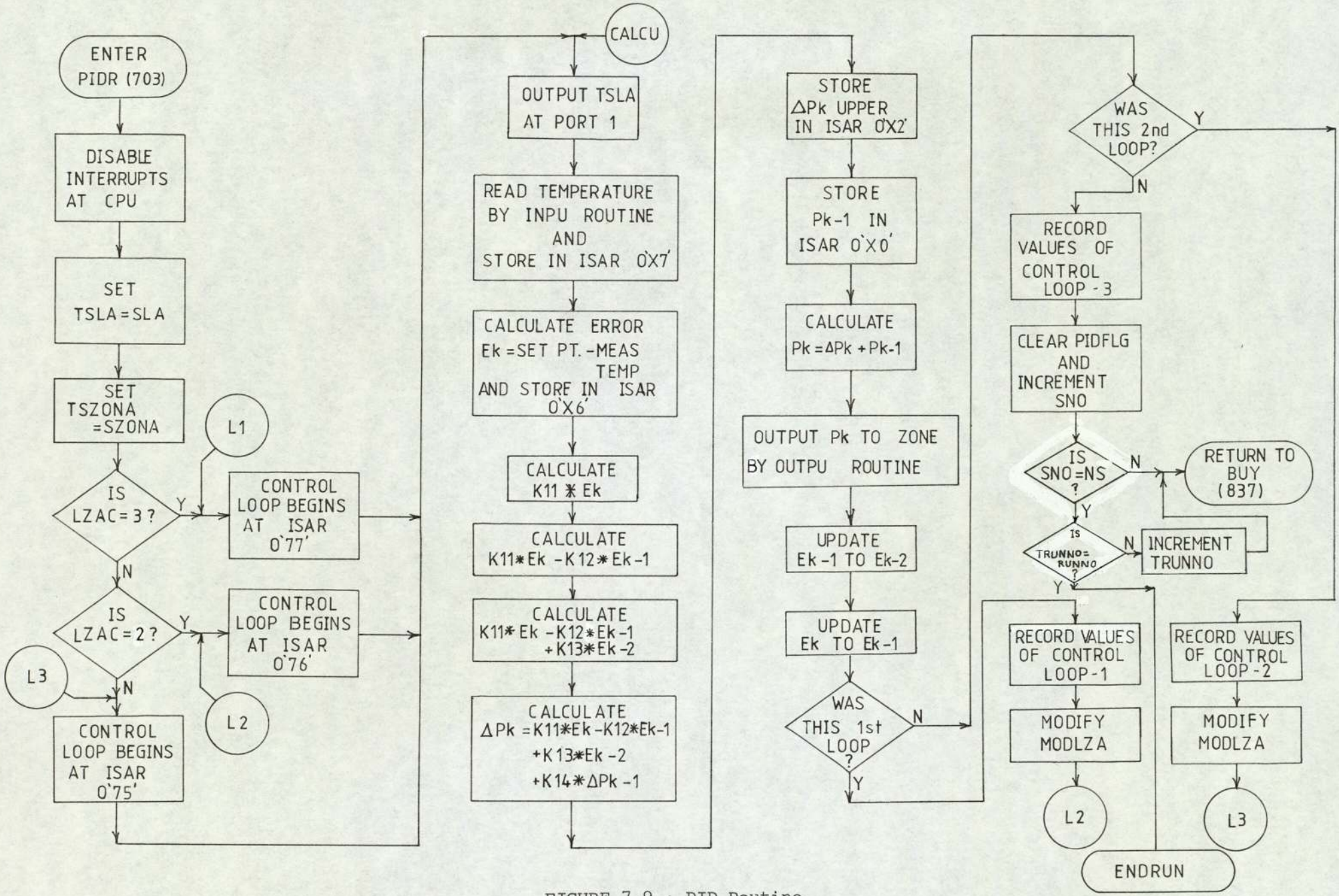


FIGURE 7.9 : PID Routine

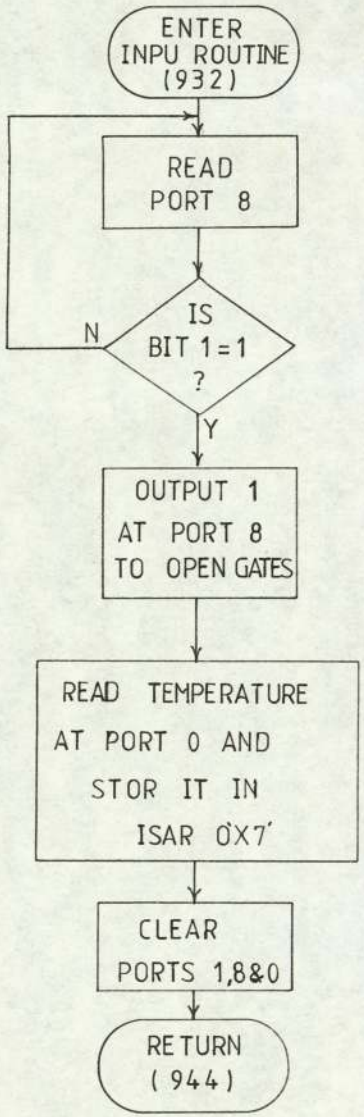


FIGURE 7.10 : INPU subroutine to read in temperature of a load

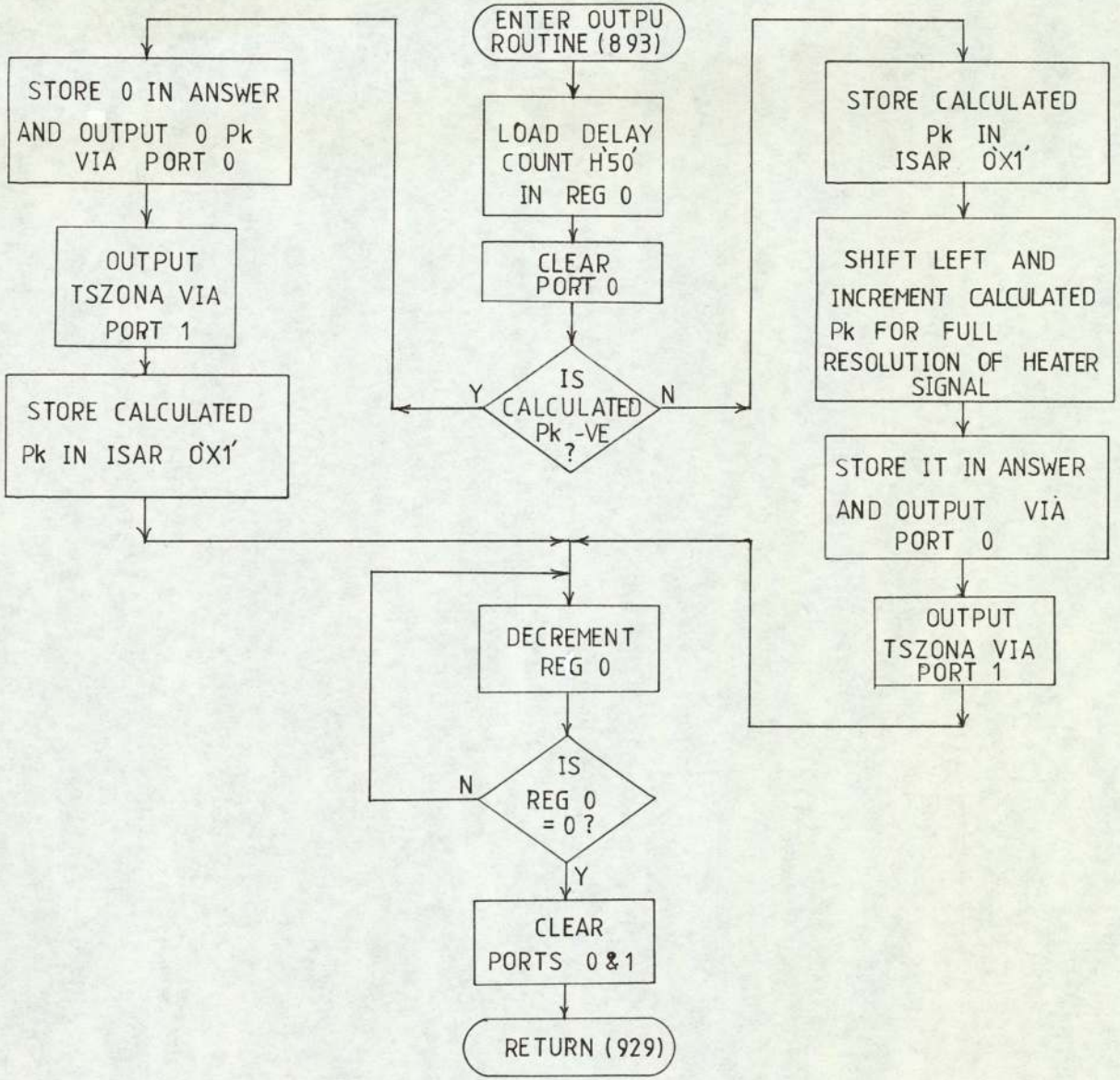


FIGURE 7.11 : OUTPU subroutine to output power to a zone

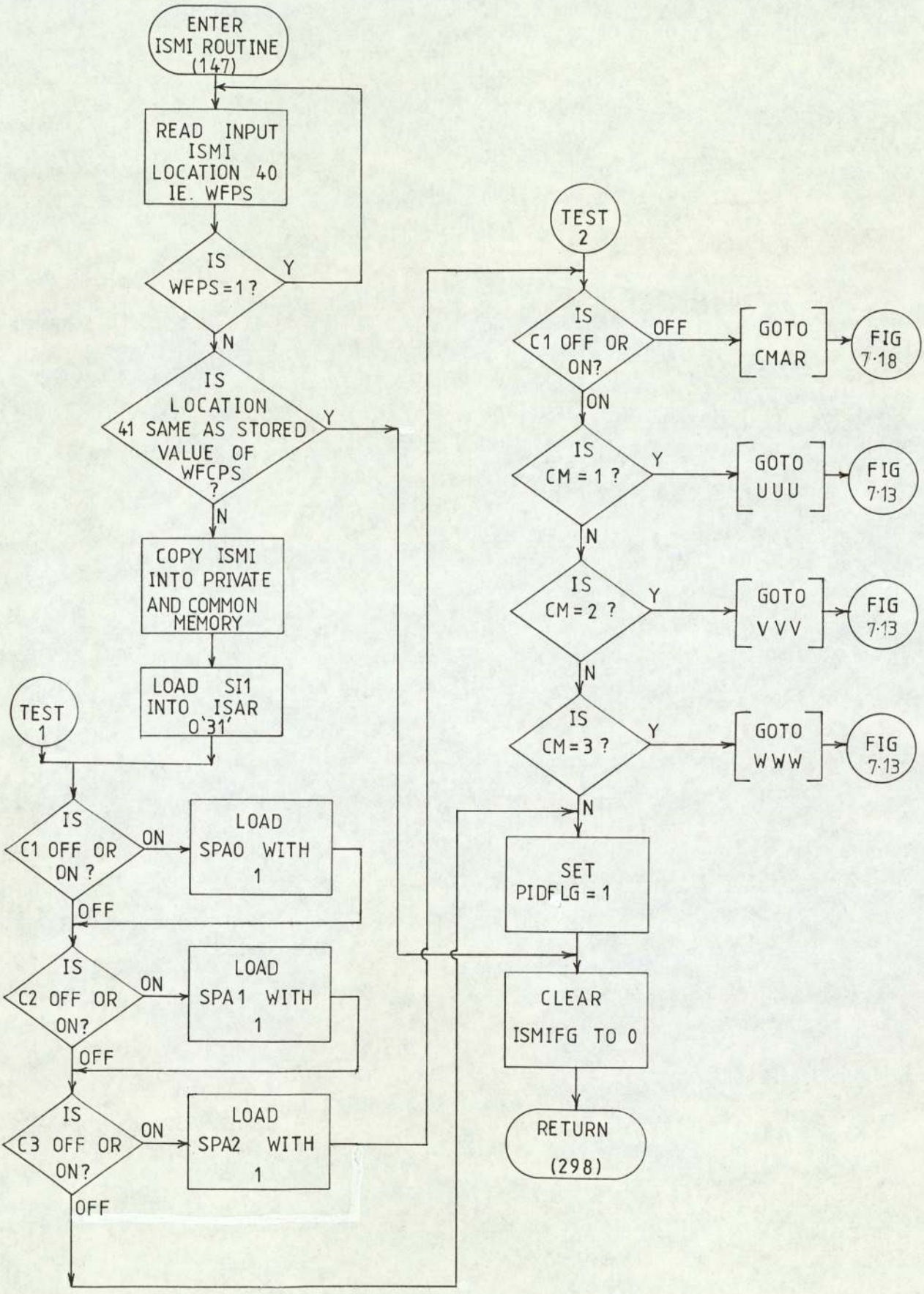


FIGURE 7.12 : ISMI Routine

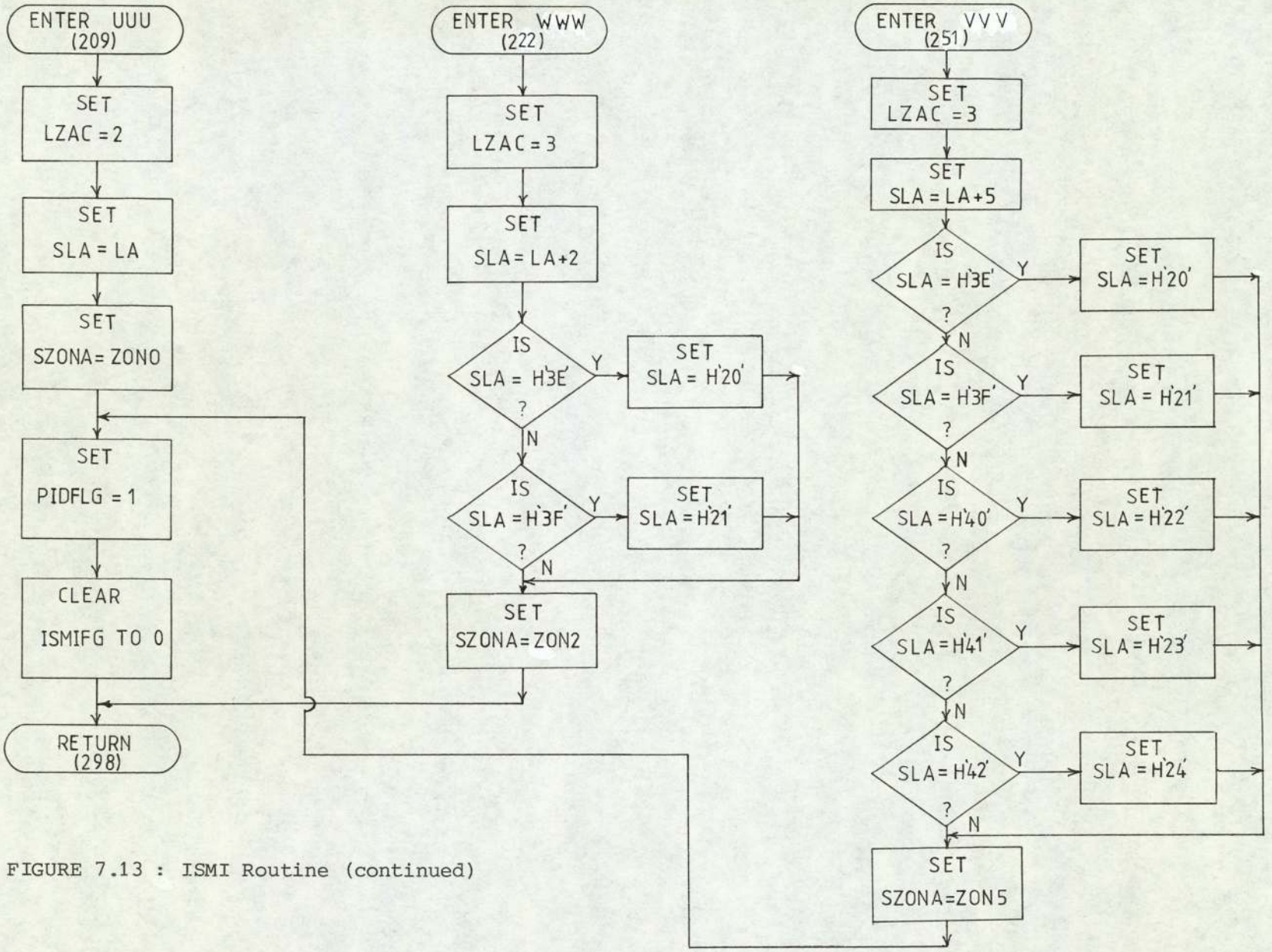


FIGURE 7.13 : ISMI Routine (continued)

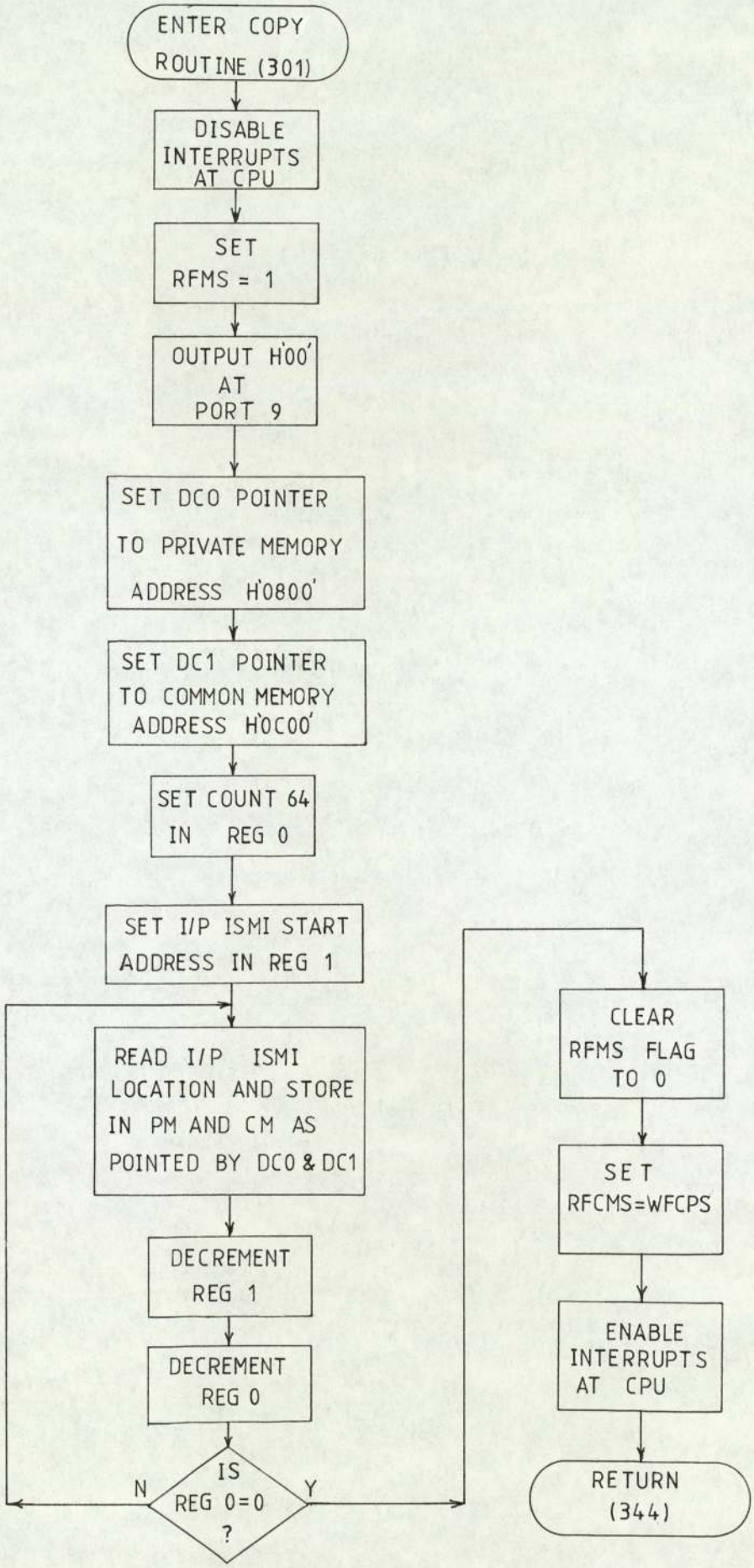


FIGURE 7.14 : Subroutine to copy ISMI data into PM and CM

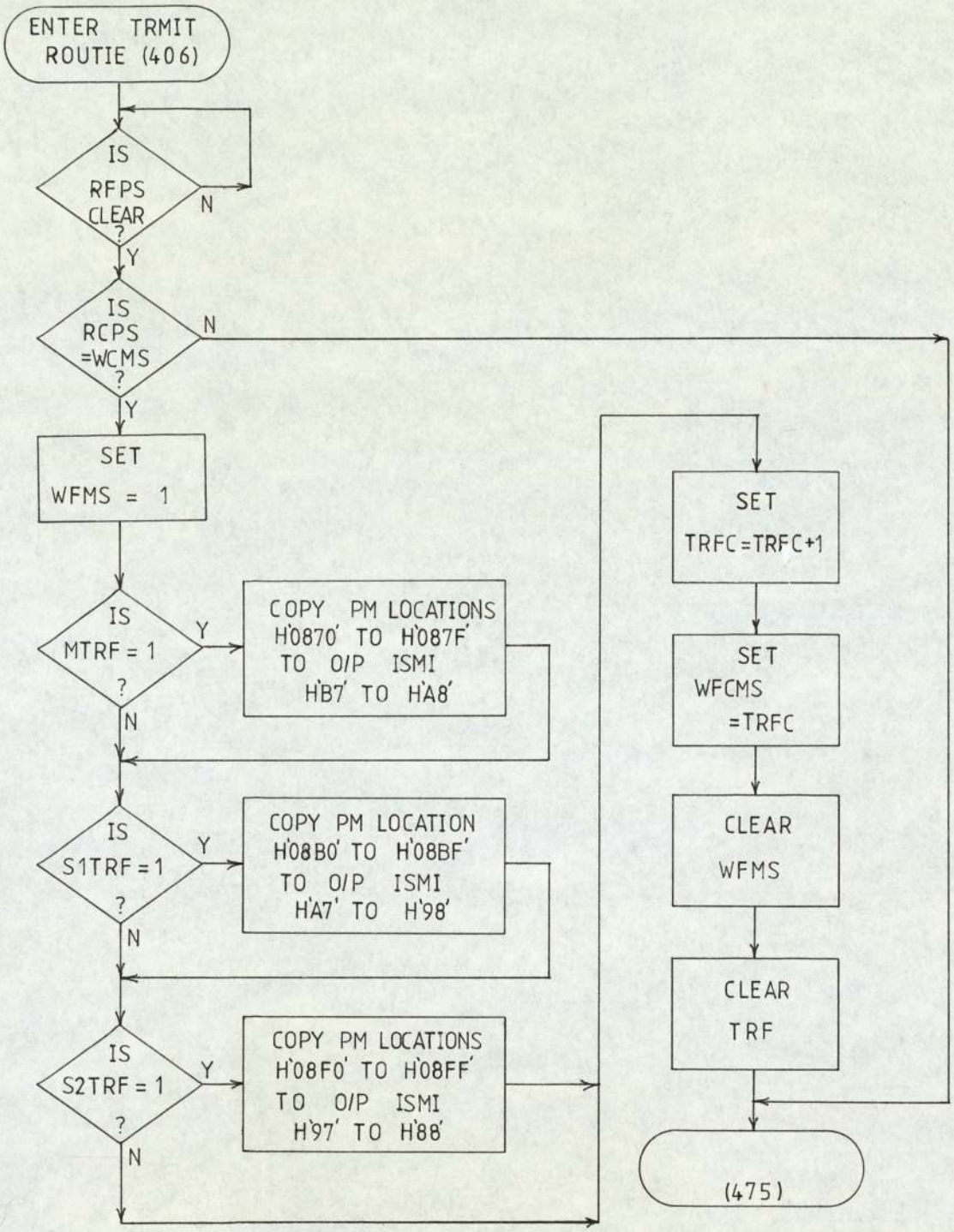


FIGURE 7.15 : TRMIT routine for data transfer from the master to o/p ISMI channel

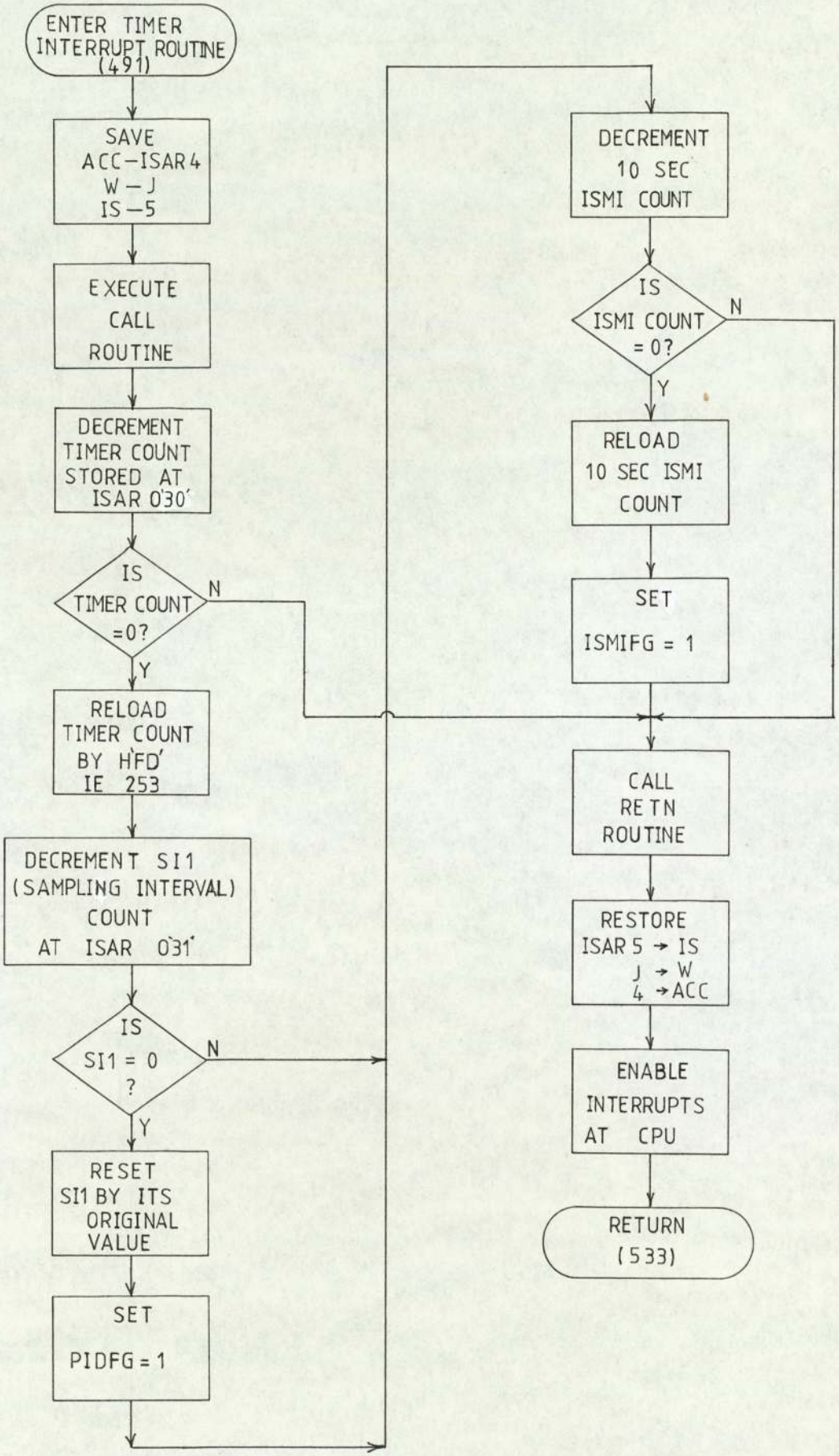


FIGURE 7.16 : TIMER Interrupt Routine

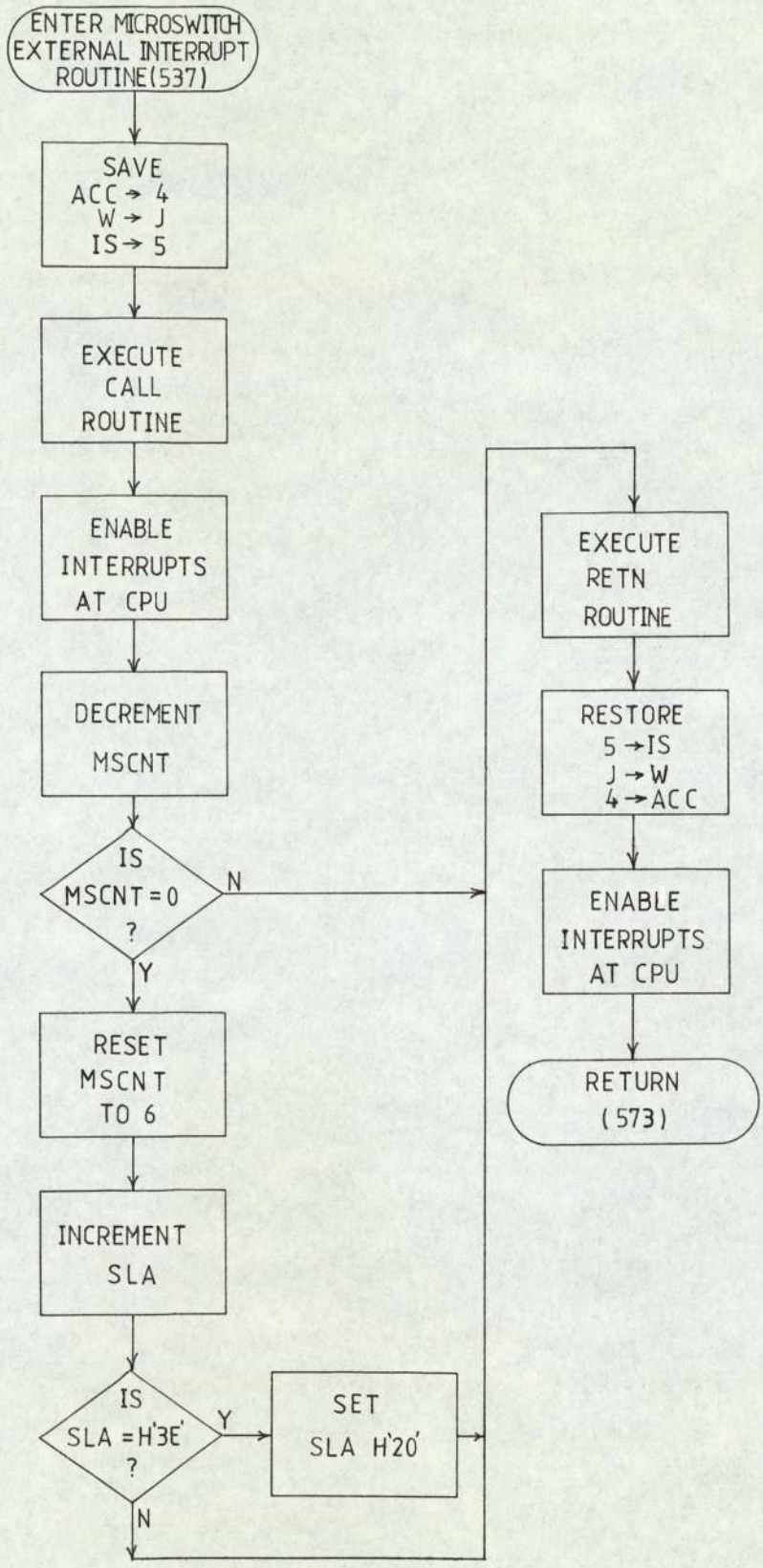


FIGURE 7.17 : External Interrupt routine for load address update

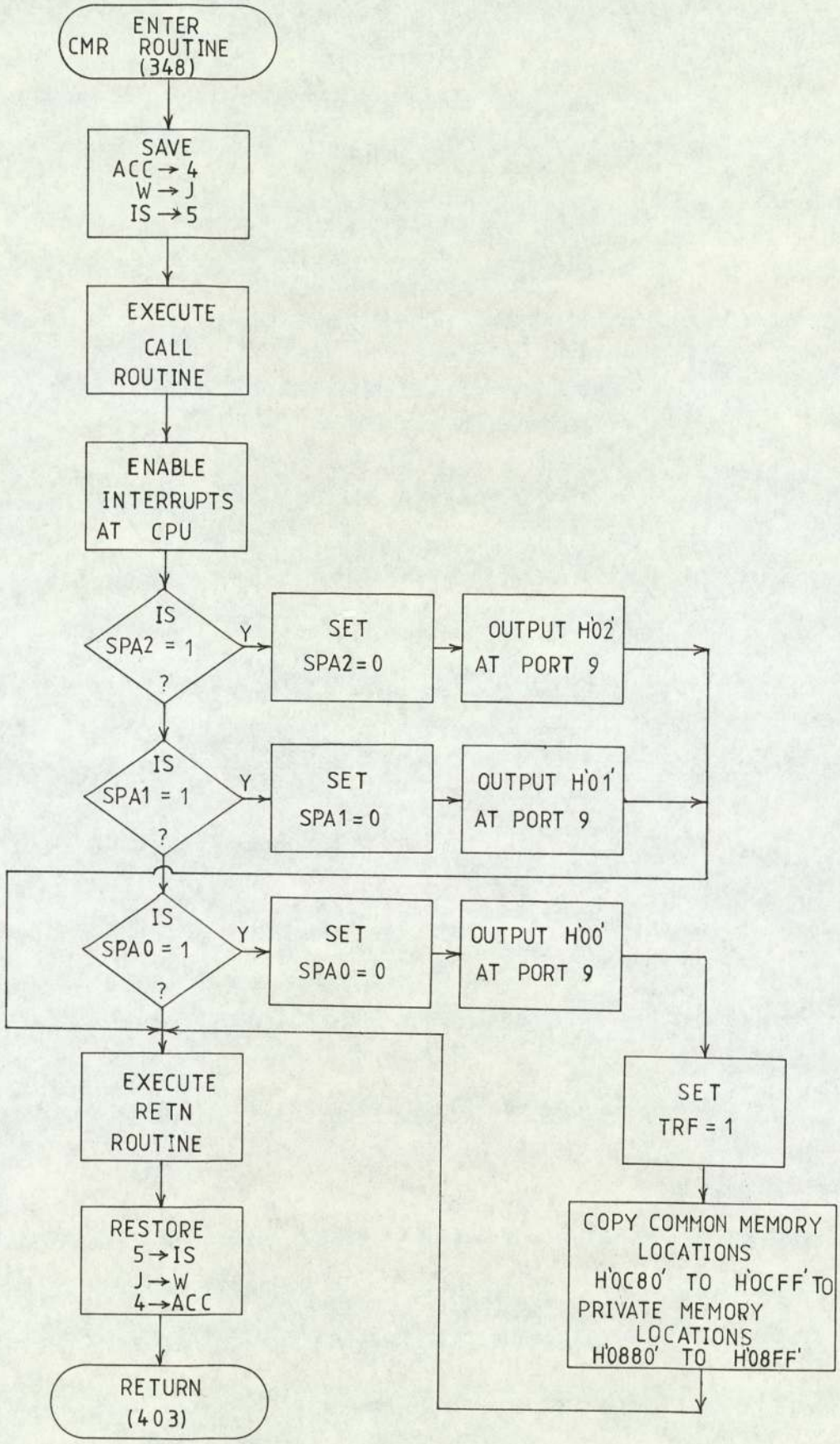


FIGURE 7.18 Common memory routine

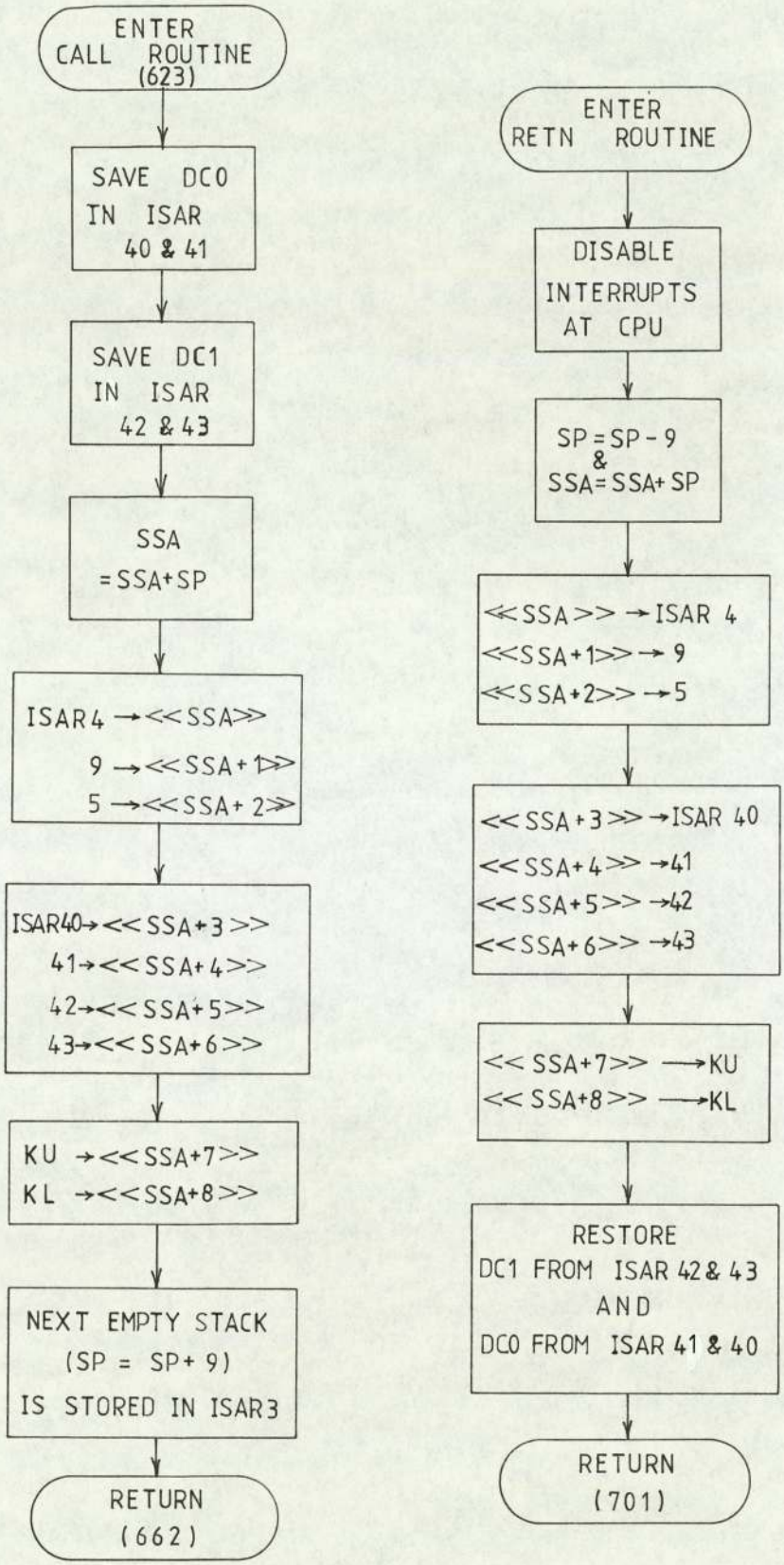


FIGURE 7.19 : CALL and RETN routine

CHAPTER 8 - SOFTWARE DEVELOPMENT FOR THE PDP-11/10 MINICOMPUTER

8.1 INTRODUCTION

This chapter covers the software development carried out on the PDP-11/10 minicomputer, which forms the supremal control level of the Hierarchical Microprocessor System Unit (HMSU), intended for controlling the Travelling Load Furnace (TLF). The processors of the HMSU are required to be activated by feeding them with the necessary input information data (e.g. controller constants, set points, control mode etc.) for controlling the TLF. This function is performed by the PDP-11/10 minicomputer in conjunction with the operator of the TLF. For this purpose, the PDP-11/10 is programmed to accomplish the following functional objectives:

1. To communicate with the operator in a suitable language with which he is familiar concerning the control process of the TLF.
2. To check on the validity of the operator set information data.
3. To allow the operator to change any data which is set either by default or by himself.
4. To display the operator set information data.
5. To convert the operator set information data in a suitable form in order to pass it onto the master processor of the HMSU.

6. To display the process variables and control signals in a graphical representation.

The implementation of the above objectives is based on the following software development features:

1. Use of the RT-11 operating system for the PDP-11 /10 minicomputer system.

2. Use of the high-level programming language, FORTRAN IV.

3. Use of a DR11-C input-output interface which provides 16 output lines and 16 input lines.

4. Use of the assembly language of the PDP-11/10, to write routines which handle data-flow through the DR11-C interface.

With reference to the above implementation, this chapter describes the program called "DCHMSU". The listing of this program is given in Appendix D.

8.2 SOFTWARE DEVELOPMENT AID

The software development aid provided under the PDP-11/10 minicomputer system consists of the RT-11 single-user programming and operating system with either single-job operation or powerful Foreground/Background (F/B) capabilities. The system also provides basic program development aids such as Editor, Assembler, Linker, Debugger, a librarian etc. A detailed description of these is well documented in the manuals. A general layout of the software development environment is shown in Figure 8.1.

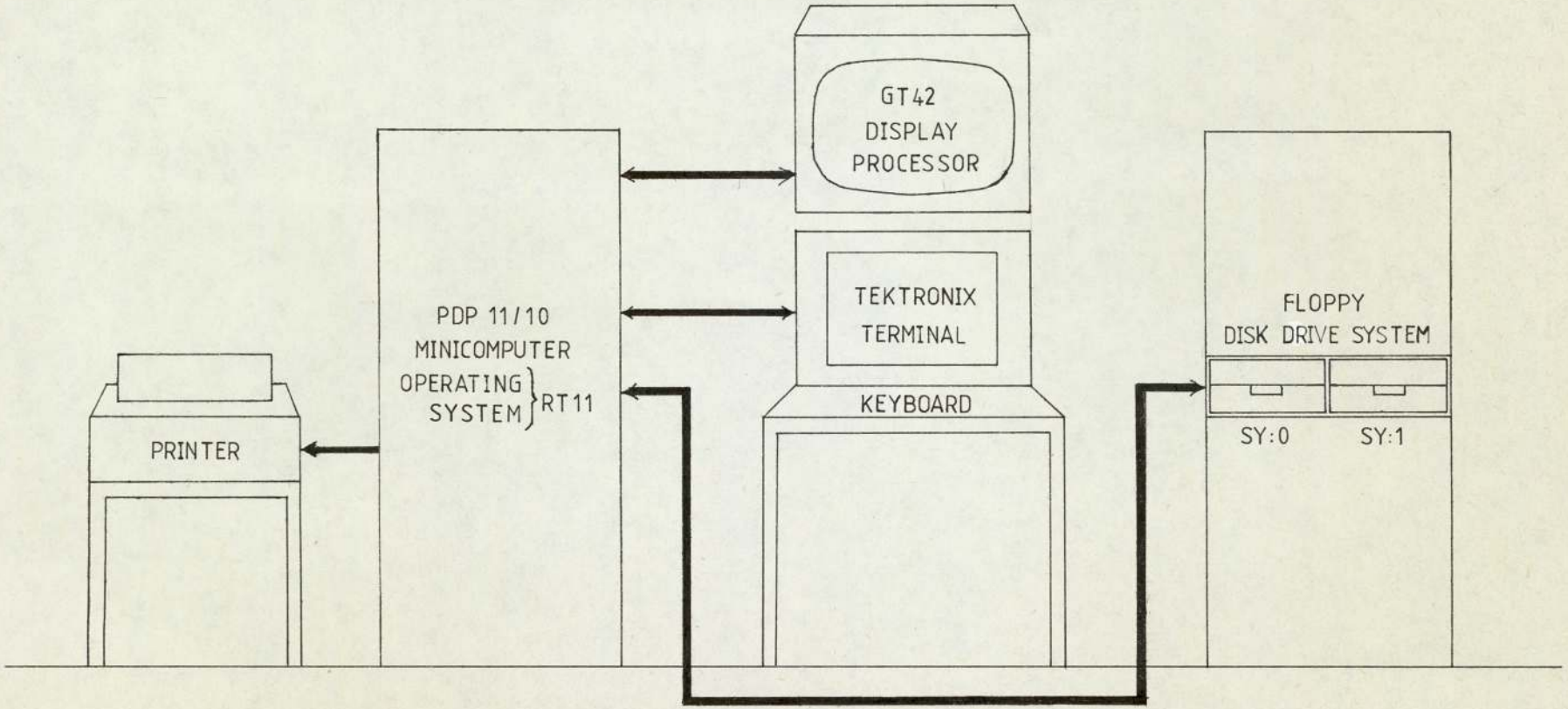


FIGURE 8.1 : Software development environment for the PDP 11/10 minicomputer

For the purpose of this project, single-job operation is chosen for simplicity. A general development procedure for generating an executable object code module from a FORTRAN source program is outlined in Figure 8.2. If a modification to the FORTRAN source program is required, it is made using the Editor and subsequent compilation and linking operations are performed on the modified version of the source program. The process of modification is repeated until the desired objectives are achieved when running the final version of the object module.

8.3 PROGRAM STRUCTURE

The structure of the "DCHMSU" program is modular. Each module is written in the form of a subroutine. These subroutines implement the functional objectives outlined in the introduction. The program execution guides the operator to set the following information as required by the processors of the HMSU:

1. The gain (k), sampling interval (τ), integral action time (T_i), derivative action time (Td) and the filter time constant (T_F), which are the main parameters which determine the values of controller constants K1, K2, K3 and K4 as given by the equations 6.3.13 of Chapter 6. The program allows the operator to set these parameters independently for each of the three controllers of the HMSU.

2. Additionally, the program asks the operator to set the control mode (refer to Section 6.3.1), the address of a load in 'zone 0', the conveyor speed, the status of a

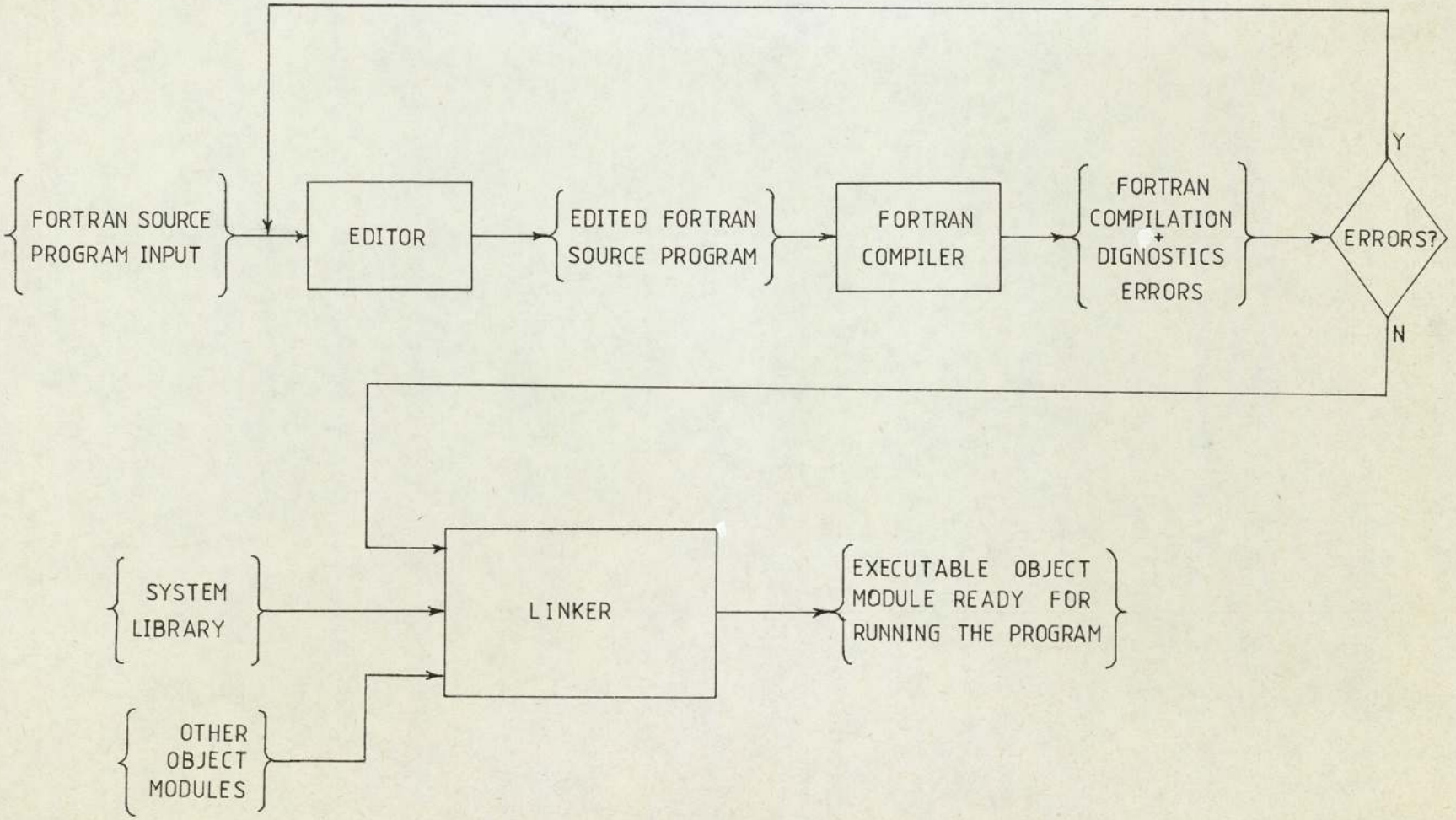


FIGURE 8.2 : A general FORTRAN source program development procedure

controller (i.e. either ON or OFF), the number of hours the TLF should run or the number of samples required to measure and control the temperature of the loads and the set point temperatures for the controllers.

Having set the above information, the program checks on the validity of the parameters. For example, it solves equations 6.3.13 and checks whether the steady state gain, computed from the values of K1, K2, K3 and K4 is positive. If it works out to be -ve, the program requests the operator to change the values of the appropriate parameters for that particular controller. The operator is also able to alter the value of any wrongly set parameter. If no information is set by the operator, the program assumes normal operating conditions for the controllers and sets the values of the various parameters by default determined at the initialisation of the DCHMSU program.

Finally, the program works out the values of the parameters in integers, each integer being one byte (8 bits) wide. This calculation is essential as these bytes, which represent the values of the parameters, are passed onto the master processors of the HMSU via the Intermediate Scratch-pad Memory Interface (ISMI). Noting that the storage locations in the ISMI can only store a byte per location, the program also generates the appropriate addresses of these locations where each parameter gets stored. In other words, the program prepares the data for transmission as requred by the ISMI memory map shown in Figure 7.4 of the previous chapter.

Since the "DCHMSU" program is written in a high-level language such as FORTRAN IV, it is easier to comprehend it from its listing given in Appendix D. However, instead of presenting flowcharts for the various subroutines, the next section demonstrates a session run which describes the actions taken by an operator and their countereffects as produced by the program execution. Table 8.1 shows the description of the various subroutines developed for the program as a whole.

8.3.1 Command Structure

Communication between the operator and the PDP-11/10 minicomputer, when the DCHMSU program is executed, is performed by different types of commands that are available to the operator. In all five commands have been developed. The information about these commands is depicted to the operator at the console when the DCHMSU program is run. This is shown in Figure 8.3. When the operator selects, say a "PAR" command, more information about the parameters of the controllers is printed out. Thus the effect of the first command is shown in Figure 8.7.

When a particular input command and its action is complete, the operator is required to press a "BREAK" key on the console. This brings him to the command mode and all the input commands available to him are displayed at the console. An object code module called IBREAK.OBJ is used during the linking procedure of all the object modules required by the DCHMSU program. This IBREAK.OBJ module accounts for the action of pressing a "BREAK" key.

NAME	*DESCRIPTION*
PROGRAM 'DCHMSU'	Under this title the following subroutines are compiled - 1. System library routine: "PRINT" - 2. Subroutine: "Q" - A question-answer subroutine that requires passing of two parameters: 1st parameter - A question in quotes 2nd parameter - An answer as integer - 3. Subroutine: "CONST" - Requires passing of five parameters: (a) Gain - GKX (b) Sampling interval - TX (c) Integral action time - TIX (d) Derivative acting time - TDX (e) Filter time constant - TFX (where x = either 1 or 2 or 3) - 4. Subroutine: "LIST" - This subroutine lists the descriptions of the parameters and requires no passing of any parameter.
SUBROUTINE OPINFO	This subroutine prints out the operator set information. Makes use of the following common blocks: - 1. Block 1 - C1, C2, C3, CM - 2. Block 3 - GK1, GK2, GK3, T1, T2, T3, TI1, TI2, TI3, TD1, TD2, TD3, TF1, TF2, TF3 - 3. Block 4 - LA, NS, MS, RH, ISP1, ISP2, ISP3 And it uses a system library routine: "CLOSE" for closing output buffer for the printer.
SUBROUTINE CHANGE	This subroutine allows the operator to change the value of any parameter described by the "LIST" subroutine. It uses the following common blocks: - 1. Block 1 as described above - 2. Block 2 - NOC1, NOC2, NOC3 - 3. Block 3 as described above - 4. Block 4 as described above Makes use of EQUIVALENCE statement.
SUBROUTINE CALCU	This subroutine calculates the values of controller constants given by equations 6.3.13. Requires input parameters: GK, T, TI, TD and TF and output parameters: XK1, XK2, XK3 and XK4. (Note: This subroutine is compiled along with the Program DCHMSU)
	CONTINUED/...

TABLE 8.1 : Subroutine modules for the DCHMSU program.

TABLE 8.1 (continued from previous page)

NAME	*DESCRIPTION*
SUBROUTINE SUBIR	1. This is a number crunching subroutine. It uses common block 5 - XK, IXK, makes use of equivalence statements. It converts fractional values of controller constants into binary (byte) fractions, their octal equivalents and integer equivalents and prints them out. This subroutine is exclusively used during development only. 2. This subroutine calls an assembly language subroutine called NUMB. This NUMB subroutine is mainly used for assembling the decimal equivalent of a binary fraction as '0's and '1's. 3. This subroutine also calls a system library routine called "CLOSE" to close the output buffer for the printer.
SUBROUTINE SEND	This subroutine assembles various values of parameters and their corresponding addresses (required by the ISMI) into two integer arrays of 64 dimension (note: the input channel of the HMSU i.e. ISMI has 64 memory locations). The routine also prints out these integer arrays. This feature is used only during the development phase. It uses the following common blocks: - 1. Block 2 as described previously - 2. Block 3 " " " - 3. Block 4 " " " - 4. Block 5 " " " - 5. Block 6 - KCM, IRCPS, IRFPS, IWCPS, IWFPS - 6. Block 7 - ISA, ISD, IRUN It also makes use of EQUIVALENCE statements.

The second command in Figure 8.3 displays the default values of the parameters as shown by its effect in Figure 8.7. These default values may not necessarily match with actual values required at run time. For example, the load address in zone 0 may not be '0'. Hence a "SET" command is required which enables the operator to set the different parameters. The effect of the "SET" command is demonstrated in Figures 8.3 and 8.4. To check and compare the new values of the parameters with their default values, the operator makes use of "DIS" (fourth command in the run sequence) comnand. Thus, changes made in the parameter values may be compared from Figure 8.7 (i.e. effect of second command) and Figure 8.8 (i.e. effect of fourth command). Note that the status of controller no. 2 has been changed to "OFF". However, its set point and default constants are not altered.

In order to change an undesired value of a parameter, the operator can make use of "CHA" command. This command allows the operator to directly specify a particular parameter. When such a parameter is specified by its name, its current value is displayed on the console and the operator is asked to specify its new value. The operator is also asked if he wants to change any more parameters; a "Yes" answer sets him in the "CHA" command loop and a "No" answer brings him back into the general input command mode. The effect of a "CHA" command (the fifth command in the run sequence) is shown in Figure 8.5.

The sixth command used by the operator in the run sequence is again a "DIS" command, the effect of which is

shown in Figure 8.9. This may again be compared with the effects of fourth and second "DIS" commands.

Finally, the seventh command in the run sequence is a "CON" command which continues the rest of the program. This command is mainly included for development purposes, in order to display the number-crunching process described in Table 8.1. The effect of this command is to point various computed values of controller constants (i.e. K1, K2, K3 and K4), the steady state gain of the controllers etc. This effect is shown in Figures 8.9 and 8.10.

8.4 CONCLUSIONS

This chapter illustrates the state of the program developed on the PDP-11/10 minicomputer. There is plenty of scope for further developments on this program. For example, the functional objective no. 6 mentioned in the introduction needs implementation, the "CON" command needs modification so that the communication between the HMSU and the PDP-11 is established via the ISMI Interfaces and the DR11-C interface. The program development is not complete for the reasons mentioned in the conclusion sections of the previous two chapters. It may be possible to use the Foreground/Background capabilities of the PDP-11 minicomputer so that the operator's communication program is run as a foreground job and the graphic display of process variables and their control as a background job, with facilities for displaying any desired zone-temperature profile in real time operation. However, this requires further work.

```
•RUN DCHMSU
 ************************************************
 THE FOLLOWING INPUT COMMANDS ARE AVAILABLE
 [1] "DIS"- PRINTS OUT OPERATOR SET INFORMATION
 [2] "SET"- OPERATOR CAN SET THE PARAMETERS
 [3] "CON"- PROGRAM CONTINUES
 [4] "CHA"- OPERATOR CAN CHANGE THE PARAMETERS
 [5] "PAR"- PRINTS OUT THE LIST OF PARAMETERS
 - PRESS RETURN KEY AFTER ANY INPUT COMMAND
 ************************************************
PAR
DIS
SET
SELECT THE CONTROLLER NO - (I.E. 1 OR 2 OR 3)
1

DO YOU WANT CONTROLLER-1 TO BE ON ?
Y
 SPECIFY CONTROLLER-1 SET POINT
100
 SPECIFY CONTROLLER-1 CONSTANTS
 GAIN
0.03
 SAMPLING INTERVAL
35.0
 INTEGRAL ACTION TIME
90.0
 DERIVATIVE ACTION TIME
30.0
 FILTER TIME CONSTANT
30.0

DO YOU WANT CONTROLLER-2 TO BE ON ?
N
SELECT THE CONTROLLER NO - (I.E. 1 OR 2 OR 3)
3

DO YOU WANT CONTROLLER-3 TO BE ON ?
Y
 SPECIFY CONTROLLER-3 SET POINT
200
```

1st Command
2nd Command
3rd Command
Effect of 3rd Command

FIGURE 8.3 : Session run of the DCHMSU program (Input/output appearing on the console)

```
 SPECIFY CONTROLLER-3 CONSTANTS
 GAIN
Ø.Ø75
 SAMPLING INTERVAL
32.Ø
 INTEGRAL ACTION TIME
9Ø.Ø
 DERIVATIVE ACTION TIME
3Ø.Ø
 FILTER TIME CONSTANT
3Ø.Ø
 WHAT IS THE CONTROL MODE FOR CONTROLLRRS?
V
 WHAT IS THE INITIAL LOAD ADDRESS IN ZONE-Ø?
2
 SPECIFY THE NUMBER OF SAMPLES
1ØØ
 SPECIFY THE MOTOR SPEED
Ø
 SPECIFY THE RUN TIME FOR THE FURNACE
 IN HOURS - (INTEGER VALUE)
5
 PRESS BREAK KEY NOW
 ************************************************
 THE FOLLOWING INPUT COMMANDS ARE AVAILABLE
 [1] "DIS"- PRINTS OUT OPERATOR SET INFORMATION
 [2] "SET"- OPERATOR CAN SET THE PARAMETERS
 [3] "CON"- PROGRAM CONTINUES
 [4] "CHA"- OPERATOR CAN CHANGE THE PARAMETERS
 [5] "PAR"- PRINTS OUT THE LIST OF PARAMETERS
 - PRESS RETURN KEY AFTER ANY INPUT COMMAND
 ************************************************
DIS
```

Effect of 3rd Command Continued

Effect of pressing Break Key

4th Command

FIGURE 8.4 : Session run of the DCHMSU program (continued)

```
CHA
 SPECIFY THE PARAMETER YOU WANT TO CHANGE
RH

THE CURRENT VALUE OF RH    =    5
SPECIFY THE NEW VALUE OF RH
4

 DO YOU WANT TO CHANGE ANY MORE PARAMETERS?
Y
 SPECIFY THE PARAMETER YOU WANT TO CHANGE
C2

THE CURRENT VALUE OF C2    =ON
SPECIFY THE NEW VALUE OF C2
OFF

 DO YOU WANT TO CHANGE ANY MORE PARAMETERS?
Y
 SPECIFY THE PARAMETER YOU WANT TO CHANGE
C2

THE CURRENT VALUE OF C2    =OFF
SPECIFY THE NEW VALUE OF C2
ON

 DO YOU WANT TO CHANGE ANY MORE PARAMETERS?
Y
 SPECIFY THE PARAMETER YOU WANT TO CHANGE
T22↑X

THE CURRENT VALUE OF T22 =ON
SPECIFY THE NEW VALUE OF T22
ON

 DO YOU WANT TO CHANGE ANY MORE PARAMETERS?
Y
 SPECIFY THE PARAMETER YOU WANT TO CHANGE
T2

THE CURRENT VALUE OF T2    = 30.000
SPECIFY THE NEW VALUE OF T2
35.00

 DO YOU WANT TO CHANGE ANY MORE PARAMETERS?
N
 PRESS BREAK KEY
 ************************************************
 THE FOLLOWING INPUT COMMANDS ARE AVAILABLE
 [1] "DIS"- PRINTS OUT OPERATOR SET INFORMATION
 [2] "SET"- OPERATOR CAN SET THE PARAMETERS
 [3] "CON"- PROGRAM CONTINUES
 [4] "CHA"- OPERATOR CAN CHANGE THE PARAMETERS
 [5] "PAR"- PRINTS OUT THE LIST OF PARAMETERS
 - PRESS RETURN KEY AFTER ANY INPUT COMMAND
 ************************************************
```

5th Command

Effect of 5th Command

Effect of pressing Break Key

FIGURE 8.5 : Session run of the DCHMSU program (continued)

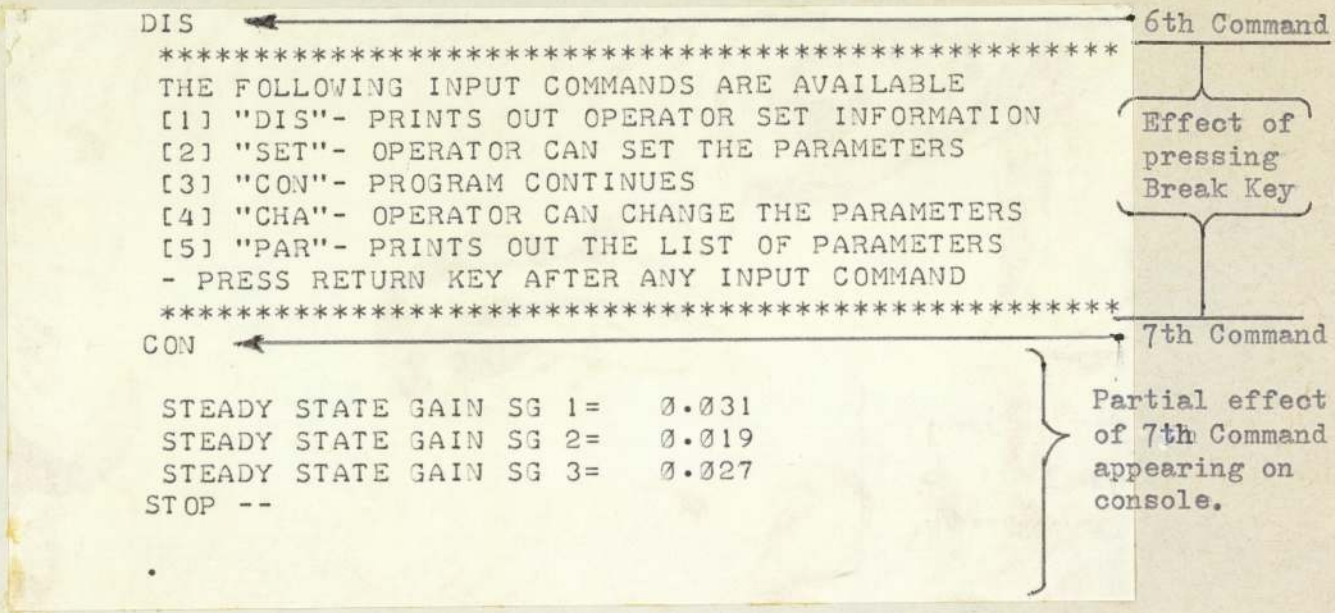


FIGURE 8.6 : Session run of the DCHMSU program (continued)

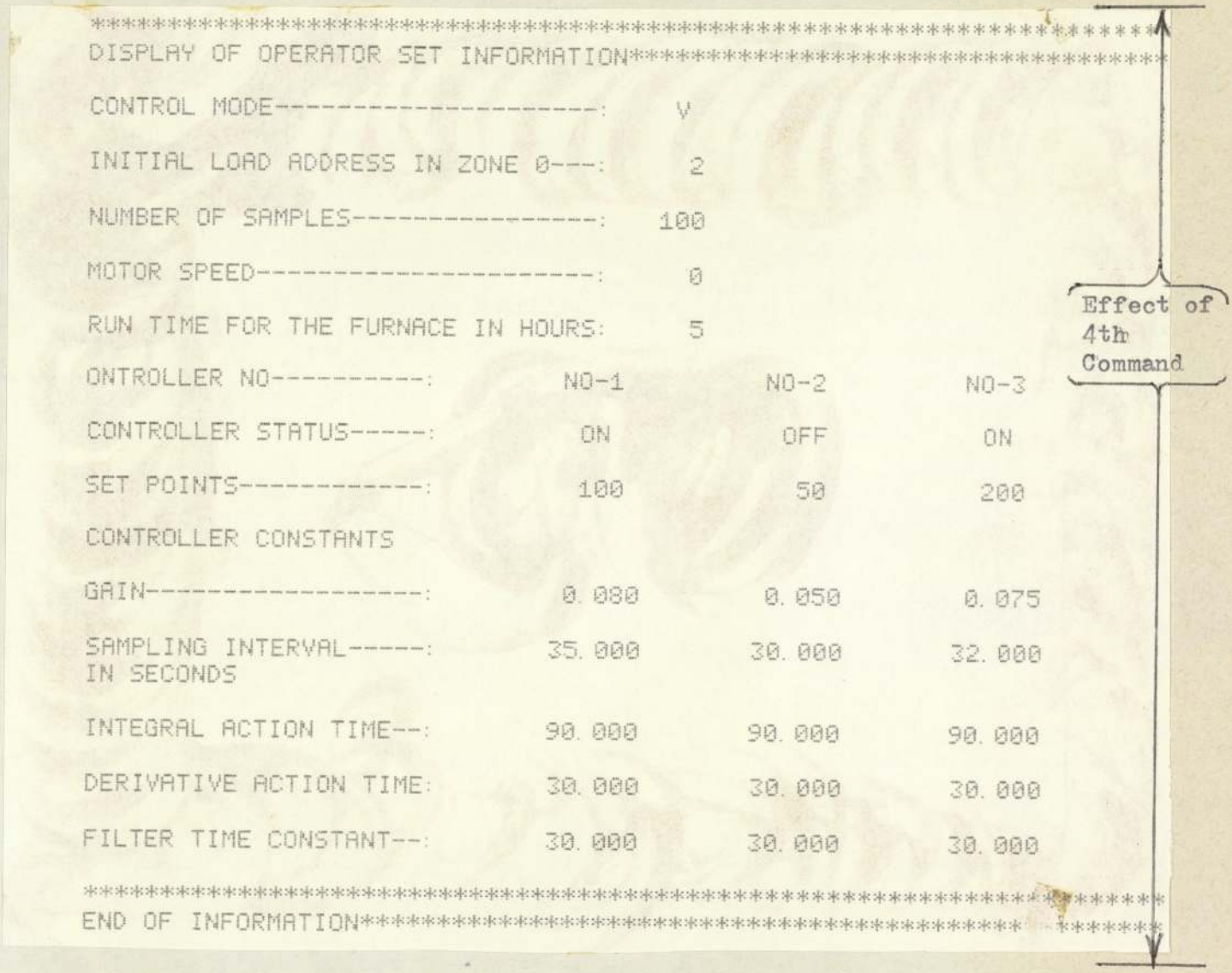
```
 THE SYMBOLIC PARAMETERS TO BE USED IN THE CHANGE SUBROUTINE ARE:
CM - CONTROL MODE
LA - INITIAL LOAD ADDRESS IN ZONE 0
NS - NUMBER OF SAMPLES
MS - CONVEYOR MOTOR SPEED
C1,C2,C3 - CONTROLLER STATUS FOR CONTROLLERS 1,2,3
ISP1,ISP2,ISP3 - SET POINTS FOR CONTROLLERS 1,2,3
GK1,GK2,GK3 - GAIN CINSTANTS FOR CONTROLLERS 1,2,3
T1,T2,T3 - SAMPLING INTERVALS FOR CONTROLLERS 1,2,3
TI1,TI2,TI3 - INTEGRAL ACTION TIMES.....
TD1,TD2,TD3 - DERIVATIVE ACTION TIMES...
TF1,TF2,TF3 - FILTER TIME CONSTANTS.....
```

Effect of 1st Command

```
**************************************************************************
DISPLAY OF OPERATOR SET INFORMATION*******************************************

CONTROL MODE---------------------:     U

INITIAL LOAD ADDRESS IN ZONE 0---:     0

NUMBER OF SAMPLES----------------:    100

MOTOR SPEED----------------------:     0

RUN TIME FOR THE FURNACE IN HOURS:     0

ONTROLLER NO----------:        NO-1          NO-2          NO-3

CONTROLLER STATUS-----:         ON            ON            ON

SET POINTS------------:          50            50            50

CONTROLLER CONSTANTS

GAIN------------------:       0.050         0.050         0.050

SAMPLING INTERVAL-----:      30.000        30.000        30.000
IN SECONDS

INTEGRAL ACTION TIME--:      90.000        90.000        90.000

DERIVATIVE ACTION TIME:      30.000        30.000        30.000

FILTER TIME CONSTANT--:      30.000        30.000        30.000

**************************************************************************
END OF INFORMATION***********************************************************
```

Effect of 2nd Command

FIGURE 8.7 : Print-out during the session run of the DCHMSU program



```
**********************************************************************
DISPLAY OF OPERATOR SET INFORMATION**********************************

CONTROL MODE---------------------:     V

INITIAL LOAD ADDRESS IN ZONE 0---:     2

NUMBER OF SAMPLES----------------:   100

MOTOR SPEED----------------------:     0

RUN TIME FOR THE FURNACE IN HOURS:     5

ONTROLLER NO----------:        NO-1          NO-2          NO-3

CONTROLLER STATUS-----:          ON           OFF            ON

SET POINTS------------:         100            50           200

CONTROLLER CONSTANTS

GAIN------------------:       0.080         0.050         0.075

SAMPLING INTERVAL-----:      35.000        30.000        32.000
IN SECONDS

INTEGRAL ACTION TIME--:      90.000        90.000        90.000

DERIVATIVE ACTION TIME:      30.000        30.000        30.000

FILTER TIME CONSTANT--:      30.000        30.000        30.000

**********************************************************************
END OF INFORMATION***************************************************
```

Effect of 4th Command

FIGURE 8.8 : Print-out during the session run of the DCHMSU program (continued)

```
**********************************************************************
DISPLAY OF OPERATOR SET INFORMATION**********************************

CONTROL MODE----------------------:     V

INITIAL LOAD ADDRESS IN ZONE 0---:      2

NUMBER OF SAMPLES-----------------:    100

MOTOR SPEED-----------------------:      0

RUN TIME FOR THE FURNACE IN HOURS:      4

ONTROLLER NU----------:         NO-1          NO-2          NO-3

CONTROLLER STATUS-----:          ON            OFF           ON

SET POINTS------------:          100            50           200

CONTROLLER CONSTANTS

GAIN------------------:         0.080         0.050         0.075

SAMPLING INTERVAL-----:        35.000        35.000        32.000
IN SECONDS

INTEGRAL ACTION TIME--:        90.000        90.000        90.000

DERIVATIVE ACTION TIME:        30.000        30.000        30.000

FILTER TIME CONSTANT--:        30.000        30.000        30.000

**********************************************************************
END OF INFORMATION***************************************************
```

Effect of 6th Command

```
   K1           K2           K3           K4

  0.097      -0.117       0.037       0.462

  0.060      -0.073       0.023       0.462

  0.089      -0.111       0.036       0.484
 STEADY STATE GAIN SG 1=    0.031
 STEADY STATE GAIN SG 2=    0.019
 STEADY STATE GAIN SG 3=    0.027
NUMBER OF SAMPLES= 240
RUN NUMBER-IRUN=    1
ACTUAL RUN TIME OF THE FURNACE WILL BE= 4HRS:39MINS.
    A(I)    FRACTION OF A(I)    BINARY FRACTION    OCTAL EQU    INTEGER EQU
    0.097        0.097          00011000            30           24
   -0.117        0.117          00011101            35           29
    0.037        0.037          00001001            11            9
    0.462        0.462          01110110           166          118
    0.060        0.060          00001111            17           15
   -0.073        0.073          00010010            22           18
    0.023        0.023          00000101             5            5
    0.462        0.462          01110110           166          118
    0.089        0.089          00010110            26           22
   -0.111        0.111          00011100            34           28
    0.036        0.036          00001001            11            9
    0.484        0.484          01111011           173          123
FINISH
```

Effect of 7th Command.

FIGURE 8.9 : Print-out during the session run of the DCHMSU program (continued)

Address	Parameter
191	255
190	0
189	256
188	2
187	240
186	0
185	1
184	1
183	100
182	24
181	29
180	9
179	118
178	35
177	0
176	0
175	50
174	15
173	18
172	5
171	118
170	35
169	0
168	0
167	200
166	22
165	28
164	9
163	123
162	32
161	0
160	0
159	0
158	0
157	0
156	0
155	0
154	0
153	0
152	0
151	0
150	0
149	0
148	0
147	0
146	0
145	0
144	0
143	0
142	0
141	0
140	0
139	0
138	0
137	0
136	0
135	0
134	0
133	0
132	0
131	10
130	11
129	12
128	13

FIGURE 8.10 : Print-out during the session run of the DCHMSU program (continued)

CHAPTER 9 - DISCUSSION

9.1 INTRODUCTION

In a modular multi-microprocessor system development, it is desirable to develop a piece of hardware and/or software, which forms a small subsystem of the whole and to test it for its behaviour and performance. The tests generally reveal the correctness of the design of such a small subsystem and any modifications necessary to improve its design and performance. These individually tested subsystem modules, when assembled to produce a complete system, tend to create less problems during their integration phase of the development.

This chapter is aimed specifically at this aspect of testing. In particular, it discusses the testing of hardware and software modules for the HMSU and the software modules for the PDP-11 minicomputer. Each module under test is described with the following common features:

1. The object of testing a particular module.
2. The experimental or test arrangement, circuit diagram, program listing etc.
3. The outcome of the test.

The following sections describe various modules under test with the above features.

9.2 SIMULATION OF MICROSWITCH INTERRUPTS

This simulation is carried out with the following objectives:

1. To test for a parallel data transfer between the two F8 processors via their bidirectional Input/Output ports.

2. To test a sequence of F8 assembly language instructions that handle external interrupts.

3. To test a sequence of F8 assembly language instructions that handle the updating of load addresses as the loads travel through the Travelling Load Furnace from one zone to the next. A load address is changed after six microswitch interrupts.

In order to achieve the above objectives, two identical "F8 Evaluation Kits" were employed and the simulation set-up using these kits is shown in Figure 9.1. A brief description of the F8 Evaluation Kit is given in Appendix B. The F8 cross-assembler available on the MAXIMOP system, mentioned in Chapter 7, is used to develop assembly language programs shown in Figures 9.2 and 9.3. The paper tape versions of the object code generated for these two programs is loaded into the RAM memory of each processor. That is, PROGRAM 1 is loaded in Processor 1 and PROGRAM 2 in Processor 2 respectively.

The "PROGRAM 1" shown in Figure 9.2 makes use of the external interrupt line available onthe SMI (Static Memory Interface) chip. The program initialises the interrupt control ports on this chip and loops into an idle loop, enabling interrupts at the CPU. When an external interrupt is received from Processor 2, the program reads in a

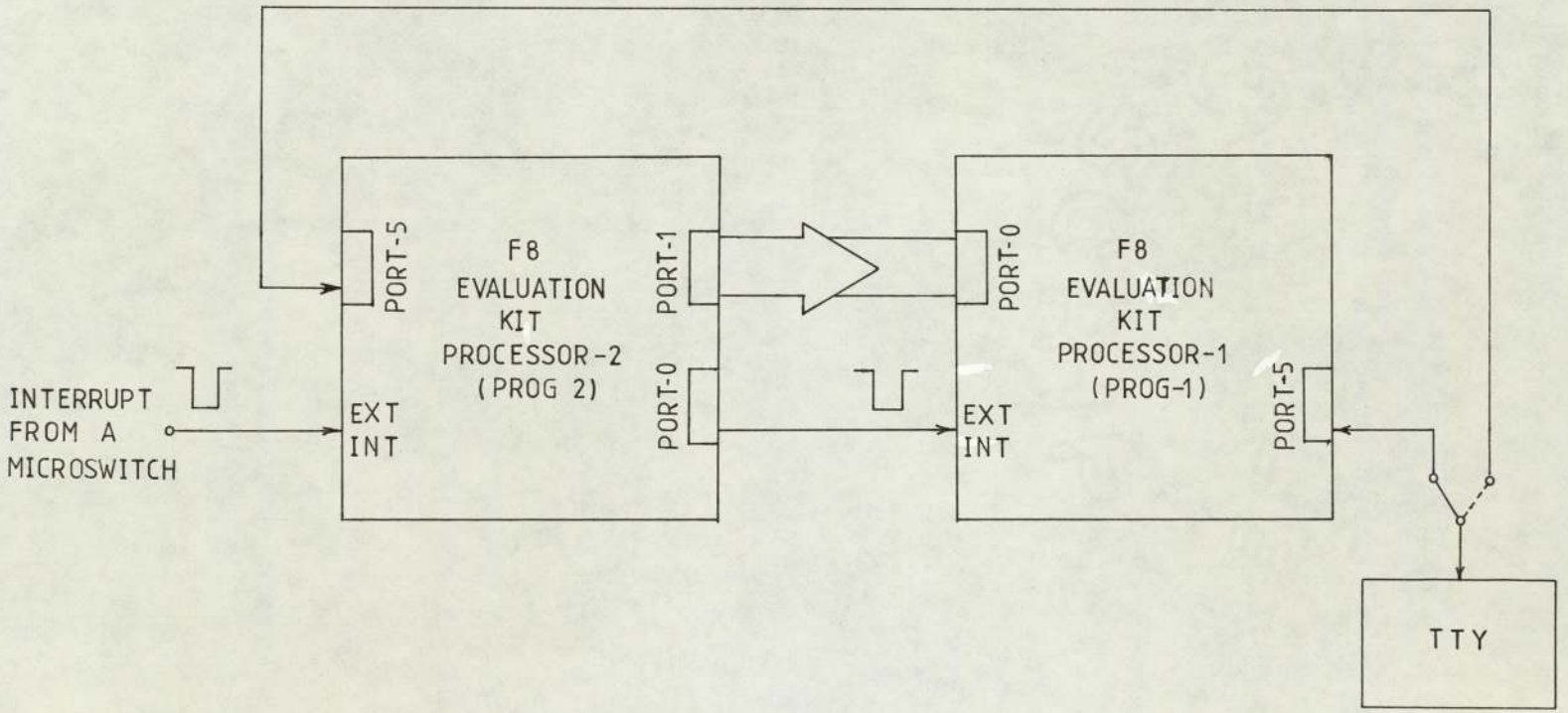


FIGURE 9.1 : Simulation set-up for microswitch interrupts using two identical F8 Evaluation Kit processors

```
←LIST STOR
FIELD TRIAL: PLEASE INFORM DR.A.C.DAVIES OF ANY ERRORS
   DATE  08/06/78   TIME  14.47.02
MOSTEK F8 CROSS ASSEMBLER  THE CITY UNIVERSITY
LONDON, EXPERIMENTAL VERSION MK3

                          ORG    H'0404'
                          PUNCH ON
                TTYOUT    EQU    H'035D'
**ADVICE: REPLACE LI BY LIS
0404 20 0F                LI     H'0F'
0406 53                   LR     3,A
0407 67                   LISU   7
0408 6F                   LISL   7
0409 54                   LR     4,A
040A 1F                   INC
040B 56                   LR     6,A
040C 75                   LIS    5
040D BC                   OUTS   H'0C'
040E 20 80                LI   H'80'
0410 BD                   OUTS   H'0D'
0411 71                   LIS    1
0412 BE                   OUTS   H'0E'
0413 1B           LOOP    EI
0414 90 FE                BR     LOOP
                          ORG    H'0580'
0580 70                   LIS    0
0581 BE                   OUTS   H'0E'
0582 73                   LIS    3
0583 B6                   OUTS   6
0584 A0                   INS    0
0585 50                   LR     0,A
0586 14                   SR     4
0587 24 30                AI     H'30'
0589 52                   LR     2,A
058A 28 04 40             PI     CHK
058D 42                   LR     A,2
058E 58                   LR 0   8,A
058F 40                   LR     A,0
0590 15                   SL     4
0591 14                   SR     4
0592 24 30                AI     H'30'
0594 52                   LR     2,A
0595 28 04 40             PI     CHK
0598 48                   LR     A,8
0599 5C                   LR     S,A
059A 1B                   EI
059B 28 03 5D             PI     TTYOUT
059E 42                   LR     A,2
059F 5C                   LR     S,A
05A0 1B                   EI
05A1 28 03 5D             PI     TTYOUT
05A4 20 20                LI     H'20'
05A6 5C                   LR     S,A
05A7 1B                   EI
05A8 28 03 5D             PI     TTYOUT
05AB 33                   DS     3
05AC 84 03                BZ     LOAD
05AE 90 12                BR     STOP
**ADVICE: REPLACE LI BY LIS
05B0 20 0A      LOAD      LI     H'0A'
05B2 5C                   LR     S,A
05B3 1B                   EI
05B4 28 03 5D             PI     TTYOUT
```

FIGURE 9.2 : PROGRAM-1 for the F8 Evaluation Kit processor 1 of Figure 9.1

```
05B7 20 0D              LI      H'0D'
05B9 5C                 LR      S,A
05BA 1B                 EI
05BB 28 03 5D           PI      TTYOUT
**ADVICE: REPLACE LI BY LIS
05BE 20 0F              LI      H'0F'
05C0 53                 LR      3,A
05C1 70         STOP    LIS     0
05C2 B5                 OUTS    5
05C3 B6           P     OUTS    6
05C4 B0                 OUTS    0
05C5 71                 LIS     1
05C6 BE                 OUTS    H'0E'
05C7 29 04 13           JMP     LOOP
                        ORG     H'0440'
                *CHK SUBROUTINE FOR CHECKING A TO F HEX NUMBERS
0440 08         CHK     LR      K,P
0441 42                 LR      A,2
0442 23 3A              XI      H'3A'
0444 84 1C              BZ      A
0446 42                 LR      A,2
0447 23 3B              XI      H'3B'
0449 84 1C              BZ      B
044B 42                 LR      A,2
044C 23 3C              XI      H'3C'
044E 84 1C              BZ      C
0450 42                 LR      A,2
0451 23 3D              XI      H'3D'
0453 84 1C              BZ      D
0455 42                 LR      A,2
0456 23 3E              XI      H'3E'
0458 84 1C              BZ      E
045A 42                 LR      A,2
045B 23 3F              XI      H'3F'
045D 84 1C              BZ      F
045F 90 1D              BR      COUT
0461 20 41      A       LI      H'41'
0463 52                 LR      2,A
0464 90 18              BR      COUT
0466 20 42      B       LI      H'42'
0468 52                 LR      2,A
0469 90 13              BR      COUT
046B 20 43      C       LI      H'43'
046D 52                 LR      2,A
046E 90 0E              BR      COUT
0470 20 44      D       LI      H'44'
0472 52                 LR      2,A
0473 90 09              BR      COUT
0475 20 45      E       LI      H'45'
0477 52                 LR      2,A
0478 90 04              BR      COUT
047A 20 46      F       LI      H'46'
047C 52                 LR      2,A
047D 0C         COUT    PK                      RETURN
        END OF ASSEMBLY
NUMBER OF ERRORS=  0

A       0461 B      0466 C      046B CHK    0440 COUT   047D D      0470
E       0475 F      047A LOAD   05B0 LOOP   0413 STOP   05C1 TTYOUT=035D

TIME ELAPSED   1.05 MINUTES
CHANNEL  2 NOW    40 BUCKETS
CHANNEL  7 NOW    10 BUCKETS
```

FIGURE 9.2 (continued)

```
13-23-52- LIST ST02
FIELD TRIAL: PLEASE INFORM DR.A.C.DAVIES OF ANY ERRORS
   DATE  08/06/78   TIME  13.22.14
MOSTEK F8 CROSS ASSEMBLER  THE CITY UNIVERSITY
LONDON, EXPERIMENTAL VERSION MK3

                          ORG    H'0500'
                          PUNCH ON
0500 2A 07 00             DCI    H'0700'
0503 76                   LIS    6
0504 17                   ST
0505 BC                   OUTS   H'0C'
0506 20 38                LI     H'38'
0508 17                   ST
0509 70                   LIS    0
050A B5                   OUTS   5
050B 20 80                LI     H'80'
050D BD                   OUTS   H'0D'
050E 71                   LIS    1
050F BE                   OUTS   H'0E'
0510 1B          LOOP     EI
0511 90 FE                BR     LOOP
                          ORG    H'0680'
0680 2A 07 00             DCI    H'0700'
0683 16                   LM
0684 50                   LR     0,A
0685 30                   DS     0
0686 84 08                BZ     LOAD
0688 40                   LR     A,0
0689 2A 07 00             DCI    H'0700'
068C 17                   ST
068D 90 20                BR     SEND
068F 76          LOAD     LIS    6
0690 2A 07 00             DCI    H'0700'
0693 17                   ST
0694 90 04                BR     MOD
0696 29 05 10 NOMOD       JMP    LOOP
0699 16          MOD      LM
069A 50                   LR     0,A
069B 30                   DS     0
069C 40                   LR     A,0
069D 25 21                CI     H'21'
069F 84 08                BZ     RECT
06A1 2A 07 01             DCI    H'0701'
06A4 40                   LR     A,0
06A5 17                   ST
06A6 90 07                BR     SEND
06A8 20 3F       RECT     LI     H'3F'
06AA 2A 07 01             DCI    H'0701'
06AD 17                   ST
06AE 70          SEND     LIS    0
06AF B1                   OUTS   1
06B0 2A 07 01             DCI    H'0701'
06B3 16                   LM
06B4 B1                   OUTS   1
06B5 71                   LIS    1
06B6 B0                   OUTS   0
06B7 2B                   NOP
06B8 2B                   NOP
06B9 70                   LIS    0
06BA B0                   OUTS   0
06BB 90 DA                BR     NOMOD
        END OF ASSEMBLY
NUMBER OF ERRORS=  0

LOAD   068F LOOP   0510 MOD    0699 NOMOD  0696 RECT   06A8 SEND   06AE

TIME ELAPSED   1.60 MINUTES
CHANNEL  2 NOW    40 BUCKETS
CHANNEL  7 NOW    10 BUCKETS
```

FIGURE 9.3 : PROGRAM-2 for the F8 Evaluation Kit Processor 2 of Figure 9.1

```
•E 404
LC38 38 38 37 37 37 37 37 37 36 36 36 36 36

35 35 35 35 35 35 34 34 34 34 34 34 33 33 33

33 33 33 32 32 32 32 32 32 31 31 31 31 31 31

30 30 30 30 30 2F 2F 2F 2F 2F 2F 2E 2E 2E 2E

2E 2D 2D 2D 2D 2D 2D 2C 2C 2C 2C 2C 2C 2B 2B

2B 2B 2B 2B 2A 2A 2A 2A 2A 2A 29 29 29 29 29

29 28 28 28 28 28 28 27 27 27 27 27 27 26 26

26 26 26 26 25 25 25 25 25 25 24 24 24 24 24

24 23 23 23 23 23 23 22 22 22 22 22 22 3F 3F

3F 3F 3F 3F 3E 3E 3E 3E 3E 3E 3D 3D 3D 3D 3D

3D 3C 3C 3C 3C 3C 3C 3B 3B 3B 3B 3B 3B 3A 3A

3A 3A 3A 3A 39 39 39 39 39 39 38 38 38
• 38 38

38 37 37 37 37 37 37 36 36 36 36 35 35 35 35

35 34 34 34 34 34 34 33 33 33 33 33 33 32 32

32 32 32 32 31 31 31 31 30 30 30 30 2F 2F 2F

2F 2F 2F 2E 2E 2E 2E 2E 2E 2D 2D 2D 2D 2D 2D

2C 2C 2C 2C 2C 2C 2B 2B 2B 2B 2B 2B 2A 2A 2A


•E 404

•2A 2A 2A 29 29 29 29 29 29 28 28 28 28 28 28

27 27 27 27 27 27 26 26 26 26 26 26 25 25 25

25 25 24 24 24 24 24 24 23 23 23 23 23 23 22

22 22 22 22 22 3F 3F 3F 3F 3F 3F 3E 3E 3E 3E

3E 3E 3D 3D 3D 3D 3D 3D 3C 3C 3C 3C 3C 3C 3B

3B 3B 3B 3B 3B 3A 3A 3A 3A 3A 3A 39 39 39 39

39 39
```

FIGURE 9.4 : Simulation output for the set-up shown in Figure 9.1

byte pattern on Port 0 and converts its lower and upper four bits into ASCII characters, corresponding to the hexadecimal numbers and prints them out onto the TTY using a TTYOUT routine available on the 3851 PSU (Program Storage Unit) chip of Processor 1. The program execution thus prints a hexadecimal number corresponding to each byte received on Port 0, per external interrupt received from Processor 2.

The PROGRAM 2, shown in Figure 9.3, also makes use of the external interrupt line available on the SMI chip of Processor 2. After initialising the interrupt control ports on this chip, the program loops into an idle loop, enabling interrupts at the CPU. When a manually generated external interrupt occurs, simulating an interrupt due to the closure of a microswitch, the program generates a load address and outputs it on Port 1 and also ouputs a sequence: H'01' followed by H'00' on Port 0. The output sequence on Port 0 causes an external interrupt generation which is linked to the external interrupt line of Processor 1. The output load address is changed only when Processor 2 receives six external interrupts. This is because a load is assumed to pass through a heating zone of the TLF with six discrete positions (Caffin, 1972).

During the testing procedure, PROGRAM 2 is initially loaded into Processor 2 and set into execution using the Execute command available on the F8 Evaluation Kit's DDT-1 (Designers Development Tool-1) program. Then the TTY is switched to Processor 1 and its PROGRAM 1 is loaded and set

into execution. Then a manually generated external interrupt at Processor 2 causes Processor 1 to print the load address. The resulting output of the simulation set-up is shown in Figure 9.4. Since there are only 30 loads, it may be noted that the load address changes from H'30' to H'2F' to account for the recirculation of the loads.

9.3 TESTING OF INTERMEDIATE SCRATCHPAD MEMORY INTERFACE

The Intermediate Scratchpad Memory Interfaces (ISMI) used as a buffered communication medium between the HMSU and the PDP-11/10 minicomputer are initially tested using two identical F8 Evaluation Kits. The objective was to test the hardware of the ISMI circuit boards. The test arrangement is shown in Figure 9.5. In the Figure, Processor 1 is used as a transmitter and Processor 2 is used as a receiver. A teletype (TTY) is used to load and execute the programs loaded into the RAM memory of each processor.

During the testing procedure, a program shown in Figure 9.8 is executed on Processor 2. This program clears 64 RAM locations (with address from H'0500' to H'0540') of Processor 2. This clearing operation is performed because the receiver program, shown in Figure 9.7, when executed uses these locations to store data it receives from Processor 1 via the ISMI interface. The TTY is then switched over to Processor 1 and the transmitter program shown in Figure 9.6 is executed. This program sends arbitrary data via Port 1 to the ISMI. The 64 locations of the ISMI, where this data is stored, are addressed via Port 0. The

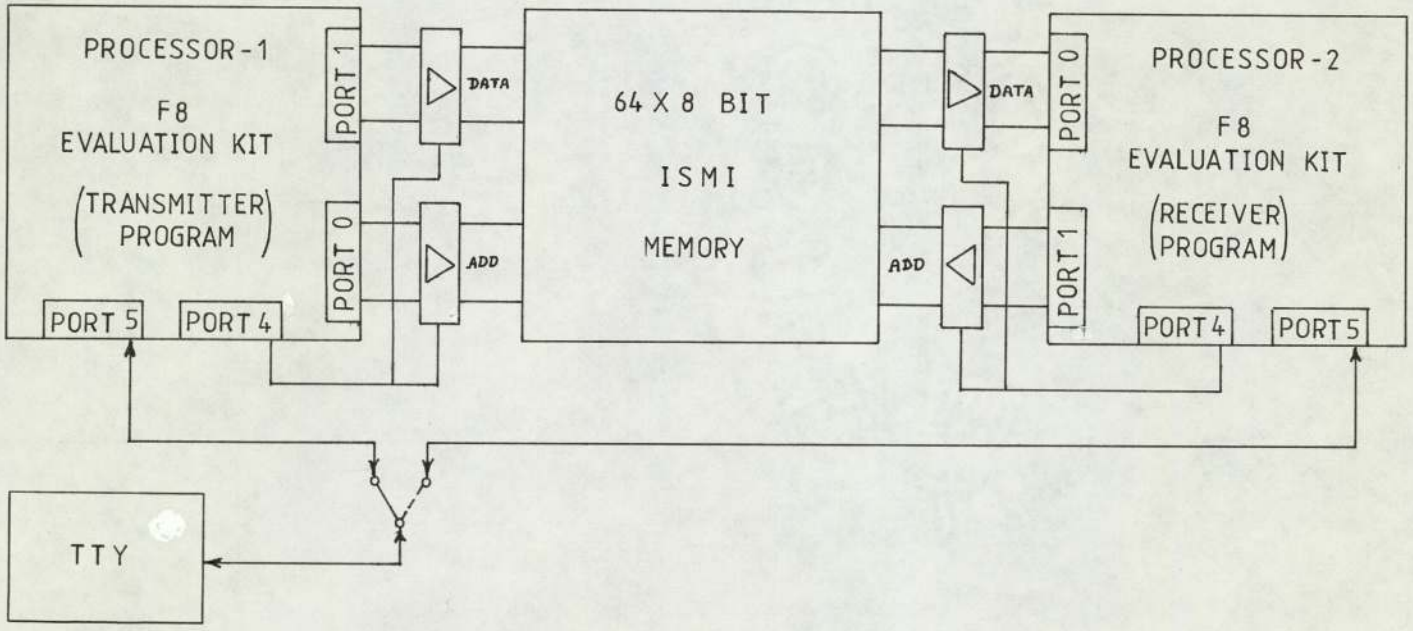


FIGURE 9.5 : Arrangement for testing ISMI using two F8 Evaluation Kits

```
400  70          CLR     CLEAR ACCUMULATOR.
401  20 40       LI  H'40'   64 COUNTS STORED IN
403  50          LR  0,A     REG 0.
404  20 FF       LI  H'FF'   ARBITRARY DATA STORED IN
406  51          LR  1,A     REG 1.
407  20 FF       LI  H'FF'   ISMI ADDRESS STORED IN
409  52          LR  2,A     REG 2.
40A  70    LOOP  LIS 0       CLEAR PORTS
40B  B0          OUTS 0      '0'
40C  B1          OUTS 1      '1'
40D  B4          OUTS 4      AND '4'.
40E  71          LIS 1       STROBE TO OPEN
40F  B4          OUTS 4      BUFFERS OF ISMI.
410  41          LR  A,1     MAKE DATA AVAILABLE
411  B1          OUTS 1      AT PORT '1'.
412  42          LR  A,2     MAKE ADDRESS AVAILABLE
413  B0          OUTS 0      AT PORT '0'.
414  31          DS  1       DECREMENT DATA.
415  32          DS  2       DECREMENT ADDRESS.
416  30          DS  0       DECREMENT COUNT.
417  94 F2       BNZ LOOP    IF NOT ZERO, RETURN TO LOOP.
419  70          LIS 0       CLEAR PORTS
41A  B0          OUTS 0      '0'
41B  B1          OUTS 1      '1'
41C  B4          OUTS 4      '4'
41D  B5          OUTS 5      '5'.
41E  29 00 00    JMP H'0000' RETURN TO DDT - 1.
```

FIGURE 9.6 : "Transmitter" program for Processor 1 of Figure 9.5

```
400  70              CLR     ;CLEAR ACCUMULATOR.
401  2A 05 00        DCI     H'0500'     ;LOAD DATA COUNTER WITH 0500.
404  20 40           LI      H'40'       ;64 COUNTS STORED IN
406  50              LR      0,A         ;REG 0.
407  20 FF           LI      H'FF'       ;ISMI ADDRESS STORED IN
409  51              LR      1,A         ;REG 1.
40A  70       LOOP   LIS     0           ;CLEAR PORTS
40B  B0              OUTS    0           ;'0'
40C  B1              OUTS    1           ;'1'
40D  B4              OUTS    4           ;AND '4'.
40E  71              LIS     1           ;STROBE TO OPEN
40F  B4              OUTS    4           ;BUFFERS OF ISMI.
410  41              LR      A,1         ;MAKE ADDRESS AVAILABLE
411  B1              OUTS    1           ;AT PORT '1'.
412  A0              INS     0           ;INPUT DATA AND
413  17              ST                  ;STORE AWAY.
414  31              DS      1           ;DECREMENT ADDRESS.
415  30              DS      0           ;DECREMENT COUNTER.
416  94 F3           BNZ     LOOP        ;IF NOT ZERO, RETURN TO LOOP.
418  70              LIS     0           ;CLEAR PORTS
419  B0              OUTS    0           ;'0'
41A  B1              OUTS    1           ;'1'
41B  B4              OUTS    4           ;'4'
41C  B5              OUTS    5           ;'5'.
41D  29 00 00        JMP     H'0000'     ;RETURN TO DDT-1.
```

FIGURE 9.7 : "Receiver" program for Processor 2 of Figure 9.5

```
600  20 40           LI      H'40'       ;64 COUNTS STORED IN
602  50              LR      0,A         ;REG 0.
603  2A 05 00        DCI     H'0500'     ;POINTER AT 0500 .
606  70              LIS     0           ;CLEAR ACCUMULATOR.
607  17       LOOP   ST                  ;H'00' STORED IN 1ST LOCATION
608  30              DS      0           ;DECREMENT COUNTER.
609  94 FD           BNZ     LOOP        ;IF NOT ZERO, RETURN TO LOOP.
60B  29 00 00        JMP     H'0000'     ;RETURN TO DDT-1.
```

FIGURE 9.8 : Program to clear 64 locations of RAM of Processor 2 of Figure 9.5

TTY is then switched over to Processor 2 and the receiver program is executed. This program reads the 64 locations of the ISMI and writes them into locations from H'0500' to H'0540'. Since the data generated by the transmitter program is known, the same data should be output if the contents of locations H'0500' to H'0540' are printed out using the DDT-1 program. All the programs (i.e. Figures 9.6, 9.7 and 9.8) were fairly short. They were hand-assembled and the paper tape versions of these were produced for testing another identical ISMI circuit board. The test was successful for both the ISMI circuit boards which proved the correctness of the same. Since the test was fairly simple, the results of the test are not included.

9.4 TESTING OF PRIVATE MEMORY AND COMMON MEMORY MODULES

Since the master and the slave processors of the HMSU are built using the F8 microprocessor chip set, any application software required for these processors is required to be embedded into PROMs. Indeed it is difficult to test the hardware of such a processor without any software. The availability of the DDT-1 program on 3851 PSU ROM chip allows some testing of the hardware of the processor. For example, the read and write capabilities of a RAM memory may be tested.

In case of the F8 microprocessor system, the CPU executes its first instruction which is stored at H'0000', after the reset action. Hence, any application program must begin at this address. The DDT-1 program on the 3851 PSU ROM chip does start at H'0000'. However, this means

that this chip cannot be used in the final system as the controller program stored in PROMs should also start at H'0000'. This requirement makes the testing of the hardware of the processor a complex task. However, this problem is overcome by using a PROM simulator. A test setup using a PROM simulator for the master processor and the DDT-1 program for one slave processor of the HMSU system is shown in Figure 9.9. The objectives of the test are as follows:

1. To test the chip select ($\overline{cs}$) logic for the PROMs, the Private RAM memory and the Common RAM memory.

2. To test the master I/O interface which controls the connection of external address and data buses of the common memory to a particular processor's internal address and data bus.

3. To test the read/write operation of the Private RAM memory and the Common RAM memory modules.

The chip select ($\overline{cs}$) logic that selects the PROMS for read operation and the Private Memory and the Common Memory for read and write operation is shown in Figure 9.10. This logic is built on each processor board of the HMSU. The Common Memory module, which contains its chip select decoding logic, requires address bus, data bus, $\overline{R/W}$ signal and CPUREAD signal of a particular processor that requires the access of it. In Figure 9.10 the address bus, $\overline{R/W}$ signal and the CPUREAD signal are the output signals of the F8 processor and hence are buffered using the 80C97 hex

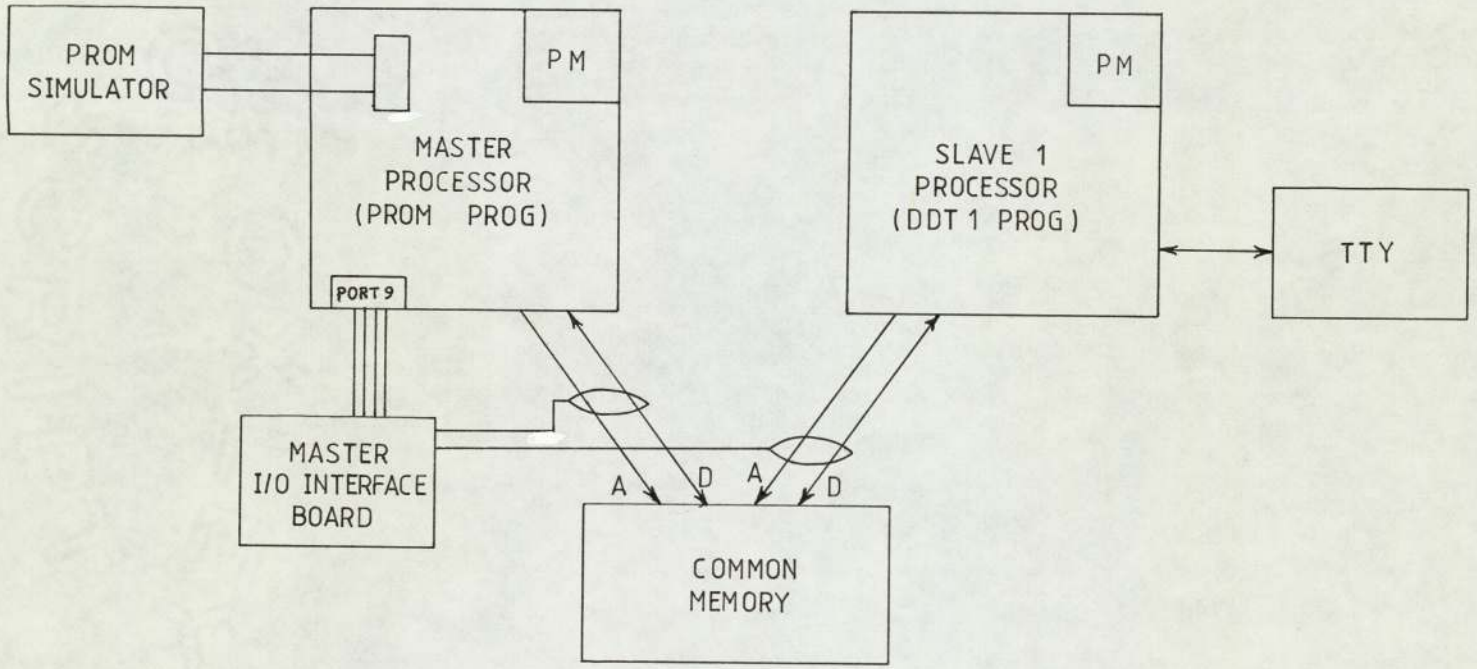


FIGURE 9.9 : Set-up using a part of the HMSU for testing PM and CM memory modules

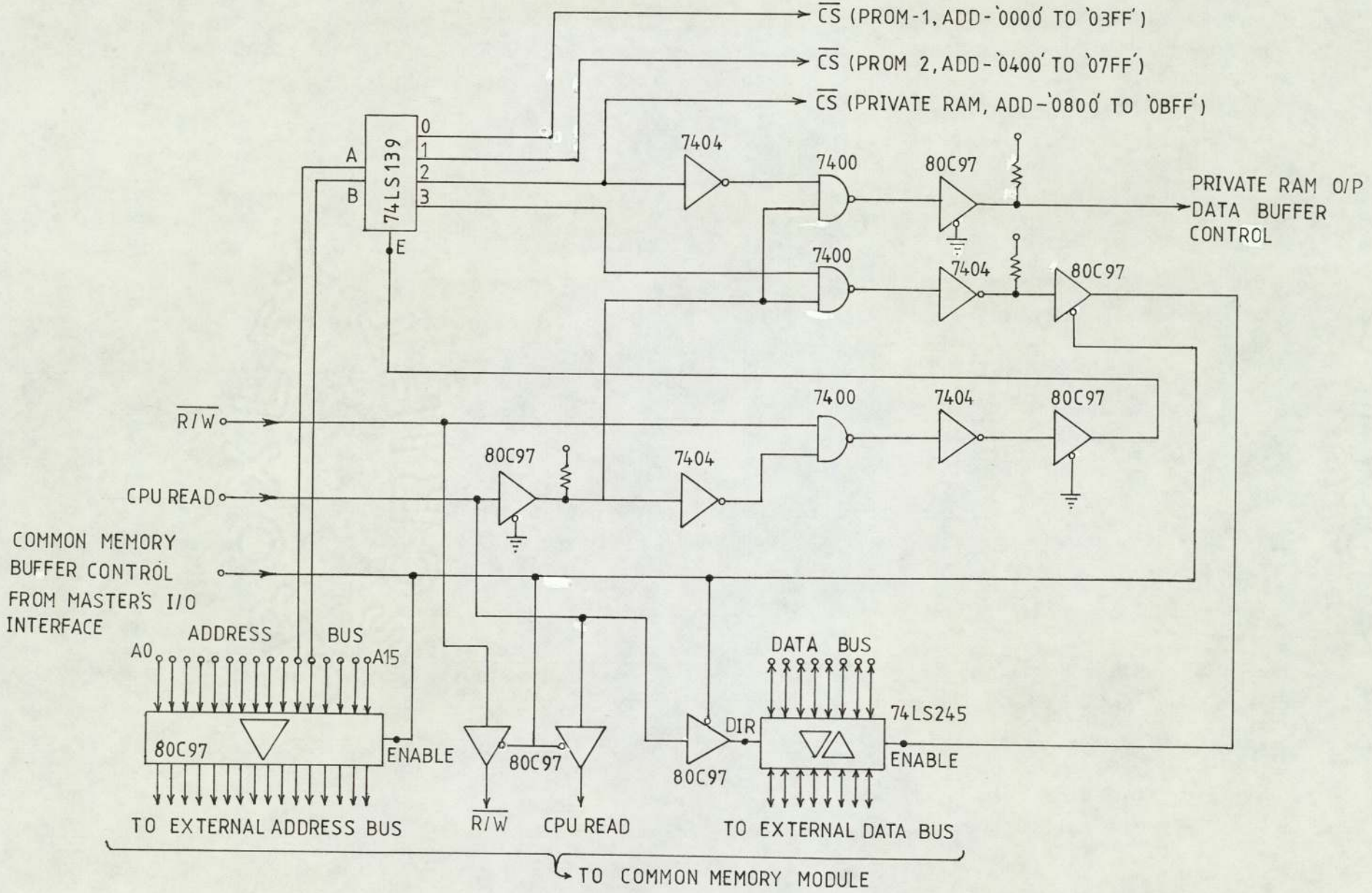


FIGURE 9.10 : Chip select logic diagram for the EPROM, PM and CM memory modules

```
0000 1A           DI
0001 70           CLR
0002 BA           OUTS   H'0A'
0003 BE           OUTS   H'0E'
0004 27 22        OUT    H'22'
0006 B9           OUTS   H'09'
0007 20 7F        LI     H'7F'
0009 51           LR     1,A
000A 2A 0C 00     DCI    H'0C00'
000D 2C           XDC
000E 2A 08 00     DCI    H'0800'
0011 41      LOOP LR     A,1
0012 17           ST
0013 2C           XDC
0014 17           ST
0015 2C           XDC
0016 31           DS     1
0017 94 F9        BNZ    LOOP
0019 20 7F        LI     H'7F'
001B 51           LR     1,A
001C 2A 0D 00     DCI    H'0D00'
001F 2C           XDC
0020 2A 08 00     DCI    H'0800'
0023 16        LP LM
0024 18           COM
0025 2C           XDC
0026 17           ST
0027 2C           XDC
0028 31           DS     1
0029 94 F9        BNZ    LP
002B 20 7F        LI     H'7F'
002D 51           LR     1,A
002E 2A 0E 00     DCI    H'0E00'
0031 2C           XDC
0032 2A 0D 00     DCI    H'0D00'
0035 16        LX LM
0036 18           COM
0037 2C           XDC
0038 17           ST
0039 2C           XDC
003A 31           DS     1
003B 94 F9        BNZ    LX
003D 71           LIS    1
003E B9           OUTS   H'09'
003F 70           CLR
0040 2B      RTN  NOP
0041 90 FE        BR     RTN
```

FIGURE 9.11 : Hand assembled program for the PROM simulator of Figure 9.9

```
.X
.
.
.T C00,C7F
0C00  7F 7E 7D 7C  7B 7A 79 78  77 76 75 74  73 72 71 70
0 C10  6F 6E 6D 6C  6B 6A 69 68  67 66 65 64  63 62 61 60
C20  5F 5E 5D 5C  5B 5A 59 58  57 56 55 54  53 52 51 50
0 C30  4F 4E 4D 4C  4B 4A 49 48  47 46 45 44  43 42 41 40
C40  3F 3E 3D 3C  3B 3A 39 38  37 36 35 34  33 32 31 30
C50  2F 2E 2D 2C  2B 2A 29 28  27 26 25 24  23 22 21 20
C60  1F 1E 1D 1C  1B 1A 19 18  17 16 15 14  13 12 11 10
C70  0F 0E 0D 0C  0B 0A 09 08  07 06 05 04  03 02 01 FF
T D00,D7F
0D00  80 81 82 83  84 85 86 87  88 89 8A 8B  8C 8D 8E 8F
D10  90 91 92 93  94 95 96 97  98 99 9A 9B  9C 9D 9E 9F
D20  A0 A1 A2 A3  A4 A5 A6 A7  A8 A9 AA AB  AC AD AE AF
D300  B0 B1 B2 B3  B4 B5 B6 B7  B8 B9 BA BB  BC BD BE BF
D40  C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
D50  D0 D1 D2 D3  D4 D5 D6 D7  D8 D9 DA DB  DC DD DE DF
D60  E0 E1 E2 E3  E4 E5 E6 E7  E8 E9 EA EB  EC ED EE EF
D70  F0 F1 F2 F3  F4 F5 F6 F7  F8 F9 FA FB  FC FD FE 00
T E00,E7F
0E00  7F 7E 7D 7C  7B 7A 79 78  77 76 75 74  73 72 71 70
E10  6F 6E 6D 6C  6B 6A 69 68  67 66 65 64  63 62 61 60
E20  5F 5E 5D 5C  5B 5A 59 58  57 56 55 54  53 52 51 50
E30  4F 4E 4D 4C  4B 4A 49 48  47 46 45 44  43 42 41 40
E40  3F 3E 3D 3C  3B 3A 39 38  37 36 35 34  33 32 31 30
E50  2F 2E 2D 2C  2B 2A 29 28  27 26 25 24  23 22 21 20
E60  1F 1E 1D 1C  1B 1A 19 18  17 16 15 14  13 12 11 10
E70  0F 0E 0D 0C  0B 0A 09 08  07 06 05 04  03 02 01 FF
.
```

FIGURE 9.12 : Slave processor's output for the test set-up of Figure 9.9

tri-state buffers. The data bus of the F8 processor is bidirectional and hence is buffered using the 74LS245 octal bus tranceivers. The direction-control signal for the tranceivers is derived from the CPUREAD signal and the 74LS139 decoder logic. The master processor controls the access of the Common Memory by a particular processor by lowering the enable signal to these buffers and tranceivers via its Input/Output interface.

In order to test the chip select logic of the processor, the following procedure is used. The PROM simulator's RAM memory is loaded with a small hand-assembled program shown in Figure 9.11. The PROM compatible plug at the end of a flat ribbon cable from the PROM simulator is placed in the socket of the PROM-1 position of the master processor. The processors of the HMSU, as shown in the arrangement of Figure 9.9, are powered up and manually reset. The master processor immediately executes its PROM simulator program. The test program performs the following operations in sequence:

1. It writes into 128 locations of the Private Memory RAM with starting address: H'0800' and Common Memory RAM with starting address: H'0C00'. The beginning pattern written is H'7F' which is decremented from one location to the next.

2. It reads from the written patterns (128 locations) of the Private Memory RAM (locations H'0800' to H'087F'), complements each pattern and writes into the Common Memory RAM with starting address: H'0D00'.

3. Then it reads 128 locations from the Common Memory RAM with starting address H'0D00' and writes into the Common Memory RAM with starting address H'0E00'.

4. Finally, it sends H'01' at its Port 9 and performs an idle loop. Sending H'01' at Port 9 causes the Slave 1 processor to have the access of the Common Memory.

When the fourth operation of the above program is complete, the DDT-1 program on the 3851 PSU chip of the slave processor can be used to type out the contents of the Common Memory. The teletype output of the above test is shown in Figure 9.12. As expected, the Common Memory contents of locations H'0C00' to H'0C7F' show the correct write operation to the Common Memory, the locations H'0D00' to H'0D7F' show the correct read operation and the contents' inversion from the Private Memory and hence the write operation performed in the first sequence for the Private Memory, and finally locations H'0E00' to H'0E7F' show the correct read operation from H'0D00' to H'0D7F', the inversion of the contents read and the correct write operation to the Common Memory. The test thus proves that the chip select logic shown in Figure 9.10 performs its required function correctly.

9.5 ADVANCED TEST FOR THE HMSU

Based on the success of the previous tests, it was decided that some means for testing the hardware of the HMSU as a whole was necessary. The design of this test is based on the same resources available as used for previous

tests. A schematic diagram with data and address paths between various modules of the HMSU is shown in the test setup of Figure 9.13. A sequence of steps in which various modules are involved in data transfer for this test are as follows:

Step 1

1. To begin with, a program execution in Slave 1 processor causes some arbitrary data to be written into ISMI (Module 1). Then this processor waits for an interrupt to come from the master processor.

2. When the interrupt comes from the master processor, the Slave 1 processor copies the Common Memory data into its private memory, inverts this data and writes back into the Common Memory in a different memory space and signals the master processor that it has finished with its access to the Common Memory.

3. The Slave 1 program ends its execution by returning its control to the DDT-1 program.

Step 2

1. While the above events are taking place in Slave 1 processor, a program in the PROM simulator for the master processor causes the master processor to wait until data has been written into ISMI (Module 1) by the Slave 1 processor.

2. When the data in ISMI (Module 1) is completely written, the master copies this data from the ISMI into its Private Memory and Common Memory and sends Slave 1 address

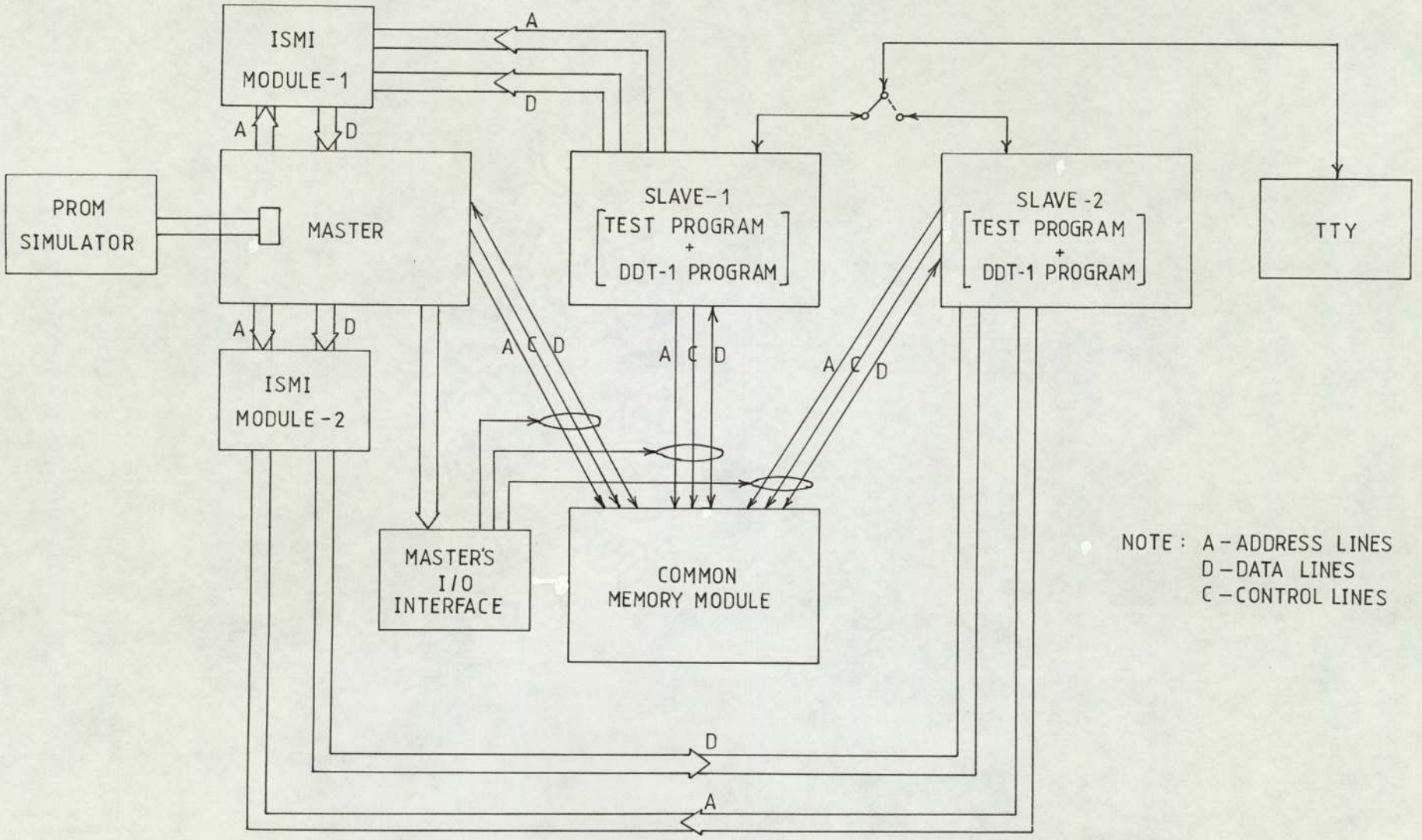


FIGURE 9.13 : Advanced test set-up for the HMSU

to its I/O interface so that Slave 1 processor can have access to Common Memory.

3. The master processor then waits for an interrupt to be received from Slave 1 processor.

4. When this interrupt is received, the master processor then sends the Slave 2 address to its I/O interface so that the Slave 2 processor can have access to Common Memory,

5. The master processor then waits for an interrupt to be received from the Slave 2 processor.

6. When this interrupt is received, the master processor sends the master address to its I/O interface so that it itself can have the access to the Common Memory.

7. The master processor then copies the Slave 1 written data from the Common Memory and writes it into the ISMI (Module 2).

8. The master processor then performs an idle loop.

Step 3

1. While the events in the first two steps are taking place in the master and Slave 1 processor, a program execution in the Slave 2 processor causes it to wait for an interrupt to come from the master processor.

2. When this interrupt is received, the Slave 2 processor copies the master written data from the Common Memory into its Private Memory and signals the master processor that it has finished with access to the Common Memory.

3. The Slave 2 processor then waits until data has been written into ISMI (Module 2) by the master processor.

4. When the data in ISMI (Module 2) is completely written, the Slave 2 processor then copies this data from the ISMI (Module 2) into its Private Memory.

5. The Slave 2 program ends its execution by returning its control to the DDT-1 program.

The above steps explicitly define the tasks required to be performed by each processor. The arbitrary data referred in Step 1 corresponds to 64 bytes as a block of data. Since all the block data movements are through ISMI modules, Common Memory module and the master processor, these are recorded by Slave 1 and Slave 2 processors indirectly in their respective Private Memories. The contents of these Private Memories can be output to a TTY using the DDT-1 program. The implementation of the tasks in the form of programs required for the three processors in this test are not given here. The reason for this was that another test of ISMI modules, not mentioned in this chapter, indicated a hardware fault on one of the ISMI modules. This particular ISMI module showed an error on the most significant bit of alternate locations of its 64 memory locations. The investigation of this fault with limited testing resources caused this test to be suspended. However, this test clearly shows the complex nature of hardware and software integration design phase as related to a multi-microprocessor system development.

9.6 ASSEMBLY LANGUAGE SUBROUTINE TESTS ON PDP-11 MINICOMPUTER

In this section, two assembly language subroutines, which are called by the high-level language program written in FORTRAN IV, are discussed. The assembly language subroutines are developed using the MACRO assembler of the PDP-11 minicomputer. The object modules produced as an output from the MACRO assembler are linked with the object modules of their main FORTRAN IV programs. The subroutines and their main programs are as follows:

9.6.1 Program IR and the NUMB macro subroutine

The program IR reads ten real numbers from the console and stores them in a real array A(I). The integer part of the real number is removed and the fractional part of the number is converted into a binary fraction, that is, using 2^{-n} where n = 1,8. Thus, for example, 0.04 is represented as 00000001 and 0.999 is represented approximately as 11111111. The binary point (equivalent to a decimal point) before the binary fraction is assumed. The NUMB subroutine converts the binary fraction into its corresponding integer value which is required to be sent to the HMSU via the ISMI memory modules. The objective of testing this IR program is thus twofold:

1. To test the calling of the assembly language program such as NUMB by correctly passing the required parameters from the high-level language program, such as program IR, and

2. To test the correctness of the NUMB macro subroutine which converts a string of '0's and '1's of eight bits width into the equivalent integer number.

The listing of the NUMB macro subroutine is shown in Figure 9.14. The register R5, as used in any autodecrement deferred mode, contains the address of an argument list having the format shown in Figure 9.15. The register R1 is used as a temporary register and after its initialisation, the argument contents are added to it and an arithmetic shift left operation is performed on it until all the arguments are added. Thus an integer is formed in R1 from a string of '0's and '1's of eight bits width. The IR program listing is shown in Figure 9.16 and the corresponding output result of the program execution is shown in Figure 9.17. It may be noted that the NUMB subroutine is used in the DCHMSU program described in the previous chapter.

9.6.2 Program TRIAL and the SUB2 macro subroutine

In the DCHMSU program, the operator set information in its final form is assembled by the SEND subroutine. Each element of the address array and the corresponding element of the data array needs packing into a 16 bit word which can be output to the Input ISMI channel of the HMSU, via the DR11-C interface. The necessary connection arrangement between the DR11-C interface and the ISMI modules is shown in Figure 9.18. The packing process of two independently stored bytes to form a 16 bit word is performed by the SUB2

```
NUMB    RT-11 MACRO VM02-12    00:10:26 PAGE 1


1                       .TITLE NUMB
2                       .GLOBL NUMB
3                       .MCALL ..V2..,.REGDEF
4 000000                .REGDEF
5 000000                ..V2..
6         000000        R0=%0
7 000000 012500 NUMB:   MOV     (R5)+,R0
8 000002 005001         CLR     R1
9 000004 105300         DECB    R0
10 00006 063501 1$:     ADD     @(R5)+,R1
11 00010 006301         ASL     R1
12 00012 105300         DECB    R0
13 00014 001374         BNE     1$
14 00016 006201         ASR     R1
15 00020 010135         MOV     R1,@(R5)+
16 00022 000207         RTS     PC
17        000000'       .END    NUMB
NUMB    RT-11 MACRO VM02-12    00:10:26 PAGE 1+
SYMBOL TABLE

NUMB    000000RG        PC    =%000007        R0    =%000000
R1    =%000001          R2    =%000002        R3    =%000003
R4    =%000004          R5    =%000005        SP    =%000006
..V2 = 000001
. ABS.   000000     000
         000024     001
ERRORS DETECTED: 0
FREE CORE: 12365. WORDS

NUMB OBJ,LP:=NUMB MAC
```

FIGURE 9.14 : MACRO Assembly of the NUMB subroutine

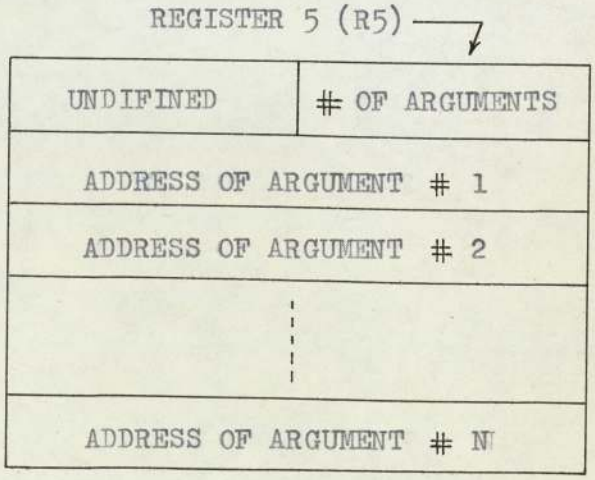


FIGURE 9.15 : Format of argument list used by Register 5 (R5) during FORTRAN subroutine linkage

```
FORTRAN IV      V01C-03A    TUE 29-APR-80 02:33:30                    PAGE 001

0001            PROGRAM IR
0002            DIMENSION A(10),AR(10),SF(8),NAA(10),IA(10,8)
0003            DATA SF/0.5,0.25,0.125,0.0625,0.03125,0.015625,
               1 0.0078125,0.00390625/
0004            CALL PRINT(' TYPE THE VALUES OF A')
0005            READ(5,111)(A(I),I=1,10)
0006  111       FORMAT(F7.3)
0007            DO 2 I=1,10
0008            P(I)=ABS(A(I))
0009            IP=IFIX(P(I))
0010            AR(I)=P(I)-FLOAT(IP)
0011            BA=AR(I)
0012            DO 3 N=1,8
0013            TA(N)=BA-SF(N)
0014            IF(TA(N))25,29,29
0015  25        IA(I,N)=0
0016            GOTO 3
0017  29        IA(I,N)=1
0018            BA=TA(N)
0019  3         CONTINUE
0020  2         CONTINUE
0021            DO 50 I=1,10
0022            CALL NUMB (IA(I,1),IA(I,2),IA(I,3),IA(I,4),IA(I,5),IA(I,6),
               1 IA(I,7),IA(I,8),NAA(I))
0023  50        CONTINUE
0024            WRITE(6,300)
0025  300       FORMAT(1H ,4X,'A(I)',4X,'FRACTION OF A(I)',4X,'BINARY FRACTIO
               1 4X,'OCTAL EQU',4X,'INTEGER EQU')
0026            DO 20 I=1,10
0027            WRITE(6,112)A(I),AR(I),(IA(I,N),N=1,8),NAA(I),NAA(I)
0028  112       FORMAT(1H ,2X,F7.3,6X,F6.3,12X,8(I1),8X,O3,8X,I3)
0029  20        CONTINUE
0030            WRITE(6,200)
0031  200       FORMAT(1H ,'FINISH')
0032            CALL CLOSE(6)
0033            STOP
0034            END
```

FIGURE 9.16 : FORTRAN IV program IR which calls the NUMB subroutine

```
   A(I)     FRACTION OF A(I)     BINARY FRACTION      OCTAL EQU     INTEGER EQU
 100.004          0.004          00000001                1              1
 999.999          0.999          11111111              377            255
   0.900          0.900          11100110              346            230
   0.500          0.500          10000000              200            128
   0.250          0.250          01000000              100             64
   0.125          0.125          00100000               40             32
 -56.219          0.219          00111000               70             56
 222.222          0.222          00111000               70             56
   0.015          0.015          00000011                3              3
   0.150          0.150          00100110               46             38
FINISH
```

FIGURE 9.17 : Output result of IR program of Figure 9.16

macro subroutine. The objective of a test program called TRIAL is to test the correctness of the SUB2 macro subroutine which performs the packing of two bytes into a 16 bit word. The TRIAL program, the SUB2 program and the output result of the TRIAL program is shown in Figure 9.19.

The TRIAL program reads two sets of four integers and stores them into arrays K and L. The corresponding elements of these arrays are packed side by side and the resulting integer is stored in Array N. Array K corresponds to the data byte and Array L corresponds to the address byte. Thus, when a packed element of Array N is sent out via the DROUT output register, the upper byte will contain the data and the lower byte will contain the address. The output result of the TRIAL program shows the correct packing process.

9.7 SIMULATION OF DISPLAY OF PROCESS VARIABLES ON GT42 DISPLAY PROCESSOR

The main objective of this simulation exercise is to indicate to the operator of the TLF, the process variables such as set point temperature, actual temperature profile, level of controlled power output to the heaters in a particular zone etc. in a graphical representation. The program called DISPLY which performs this simulation is shown in Figure 9.20. The program uses a variety of subroutines, described in the VT-11 Graphic Support manual, and the real-time TIMR subroutine. A file containing sample numbers, sampling times, measured temperatures and normalised power levels for the heaters is produced and called as

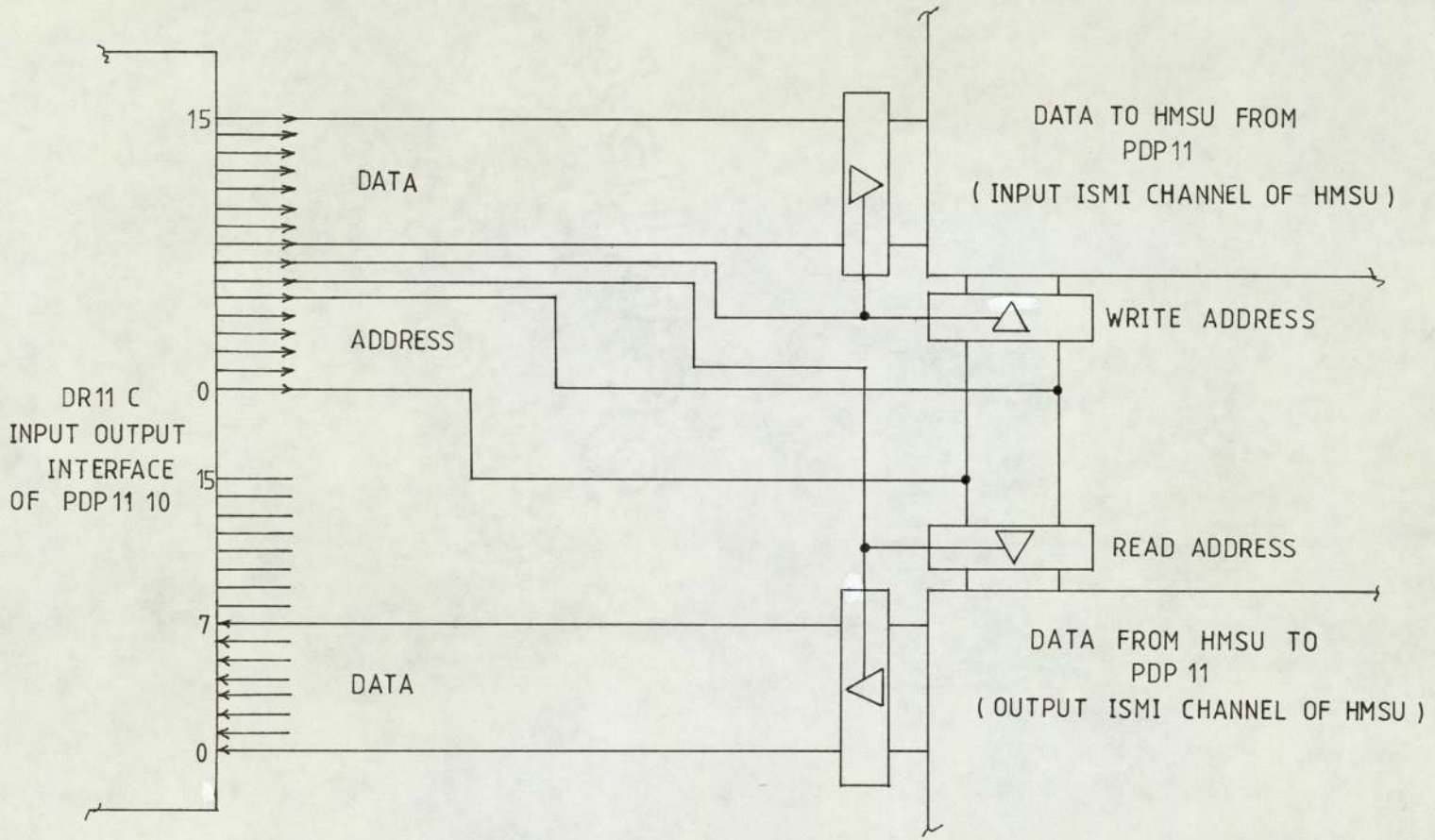


FIGURE 9.18 : Connection arrangement between DR11-C interface and ISMI modules

```
      PROGRAM TRIAL
C     THIS PROGRAM CALLS FOR A SUBROUTINE WRITTEN IN ASSEMBLY
C     LANGUAGE AND PRINTS THE RESULTS ON THE PRINTER.
      DIMENSION K(4),L(4),N(4)
      COMMON K,L,N
      CALL PRINT(' TYPE THE VALUES OF K')
      READ (5,100) (K(I),I=1,4)
      CALL PRINT(' TYPE THE VALUES OF L')
      READ (5,100) (L(I),I=1,4)
      DO 10 I=1,4
      CALL SUB2(K(I),L(I),N(I))
10    CONTINUE
      WRITE(6,200)(K(I),L(I),N(I),I=1,4)
200   FORMAT(1H ,3X,I4,3X,I4,3X,I4)
100   FORMAT(I4)
      STOP
      END
```

```
      .TITLE SUB2
      .GLOBL SUB2
      .MCALL ..V2..,.REGDEF
      .REGDEF
      DRCSR=164020
      DROUT=DRCSR+2
      DRIN=DROUT+2
SUB2: CLR     DRCSR              ;CLEAR DR11-C STATUS REGISTER
      ADD     #2,R5
      CLR     R1                 ;CLEAR REGISTER 1

1$:   MOV     @(R5)+,R1          ;LOAD R1 WITH DATA
      SWAB    R1                 ;SHIFT DATA TO HIGHER BYTE
      ADD     @(R5)+,R1          ;FILL THE LOWER BYTE OF R1 BY ADDRESS
      MOV     R1,DROUT           ;OUTPUT DATA & ADDRESS TO ISMI
      MOV     R1,@(R5)+
      RTS     PC                 ;RETURN
      .END
```

```
 0      0      0
 0      1      1
 0      2      2
 0      3      3

 1      0    256
 2      0    512
 3      0    768
 4      0   1024

 1      0    256
 1     10    266
 1     20    276
 1     30    286

 1      1    257
 2      2    514
 3      3    771
 4      4   1028
```

FIGURE 9.19 : Program TRIAL, MACRO subroutine SUB2 and output result of TRIAL program.

a DATA file. Each sample from the DATA file is fed as an input to the DISPLY program and the DISPLY program displays graphically the data contained in each sample in real time. Thus it simulates the real-time process variable changes influenced by the control algorithm. The results of the simulation output are shown in Figure 9.21 and simulated test samples of a set of data are shown in the DATA file of Figure 9.22. The dash-dotted line in Figure 9.21 shows a set point temperature of 200^{o}C, the bottom rectangular curve shows the level of power required and the smooth curve which meets the set-point line shows the variation of temperatures. It should be noted that the simulation program DISPLY is not implemented into the DCHMSU program.

9.8 CONCLUSION

This chapter indicates one of the transient states of a typical experimental environment under which the project was performed. This phase of experimentation was found to be very important in order to investigate capabilities of the hardware and software developed. The methods of testing and simulations outlined in this chapter point to areas where improvements and further testing is needed. For example, one critical area might be located in the third test (i.e. Section 9.4) where a failure of tristate buffers or 74LS245 tranceivers could create unpredictable problems such as a data bus contention during the memory access. The hardware fault found before an advanced test on the HMSU as a whole could be performed, needs further investigation. In such circumstances, what measures or diagnostic

```
FORTRAN IV      V01C-03A   TUE 04-DEC-79 00:09:46                PAGE 001

0001          PROGRAM DISPLY
0002          DIMENSION IBUF(800),T(60),TMP(60),P(60),K(60)
0003          CALL ASSIGN(10,'DATA',0)
0004          I=1
0005          K(0)=0
0006          T(0)=0.0
0007          P(0)=0.0
0008          TMP(0)=0.0
0009          CALL PRINT('WHAT IS THE SET POINT TEMP ?')
0010          READ(5,150) SP
0011  150     FORMAT(F9.3)
0012          CALL FREE
0013          CALL INIT(IBUF,800)
0014          CALL SCAL(-40.,-20.,340.,600.)
0015          CALL APNT(0.,0.,0,-5,0,1)
0016          CALL LVECT(340.,0.,0,5,0,1)
0017          CALL APNT(0.,0.,0,-5,0,1)
0018          CALL LVECT(0.,500.,0,5,0,1)
0019          CALL APNT(-10.,550.,0,-5,0,1)
0020          CALL TEXT('TEMP')
0021          CALL APNT(-15.,0.,0,-5,0,1)
0022          CALL LVECT(0.,500.,0,5,0,1)
0023          CALL APNT(-40.,525.,0,-5,0,1)
0024          CALL TEXT('POWER')
0025          CALL APNT(30.,-15.,0,-5,0,1)
0026          CALL TEXT('TIME IN MINS')
0027          CALL APNT(0.,SP,0,-5,0,1)
0028          CALL LVECT(340.,0.,0,5,0,4)
0029          CALL VECT(-100.,-20.,0,-5,0,1)
0030          CALL NMBR(1,SP,'F9.3')
0031          CALL APNT(0.,0.,0,-5,0,1)
0032          CALL PRINT('N      T        TMP        P')
0033  100     READ(5,200)N,T(I),TMP(I),P(I),JNG
0034  200     FORMAT(I3,3F9.3,I2)
0035          IF(JNG.EQ.0)GOTO 500
0037          TD=T(I)-T(I-1)
0038          TDN=-TD
0039          TMPD=TMP(I)-TMP(I-1)
0040          TMPN=-TMPD
0041          TMPP=TMP(I-1)
0042          TMPPN=-TMPP
0043          PP=P(I-1)
0044          PD=P(I)-P(I-1)
0045          PN=-P(I)
0046          CALL VECT(0.,TMPP,0,-5,0,1)
0047          CALL VECT(TD,TMPD,0,5,0,1)
0048          CALL VECT(TDN,TMPN,0,-5,0,1)
0049          CALL VECT(0.,TMPPN,0,-5,0,1)
0050          CALL VECT(0.,PP,0,-5,0,1)
0051          CALL VECT(TD,0.,0,5,0,1)
0052          CALL VECT(0.,PD,0,5,0,1)
0053          CALL VECT(0.,PN,0,-5,0,1)
0054          CALL TIME(15*60)
0055  300     CALL TIMR(IE)
```

FIGURE 9.20 : DISPLY program

```
FORTRAN IV      V01C-03A   TUE 04-DEC-79 00:09:46                    PAGE 002

0056            IF(IE.NE.0)GOTO 300
0058            WRITE(10,200)N,T(I),TMP(I),P(I),JNG
0059     *      K(I)=N
0060            I=I+1
0061            IF(I.GE.61)GOTO 500
0063            GO TO 100
0064  500       CALL TIME(0)
0065            CALL STOP
0066            WRITE(6,245)SP
0067  245       FORMAT(1H ,' SET POINT TEMP=',F9.3/)
0068            WRITE(6,250)
0069  250       FORMAT(1H0,'  N        T            TMP            P')
0070            WRITE(6,255)((K(I),T(I),TMP(I),P(I)),I=1,N)
0071  255       FORMAT(1H ,I3,2X,F9.3,2X,F9.3,2X,F9.3)
0072            CALL CLOSE(10)
0073            STOP
0074            END
FORTRAN IV      STORAGE MAP
```

FIGURE 9.20 : DISPLY program (continued)

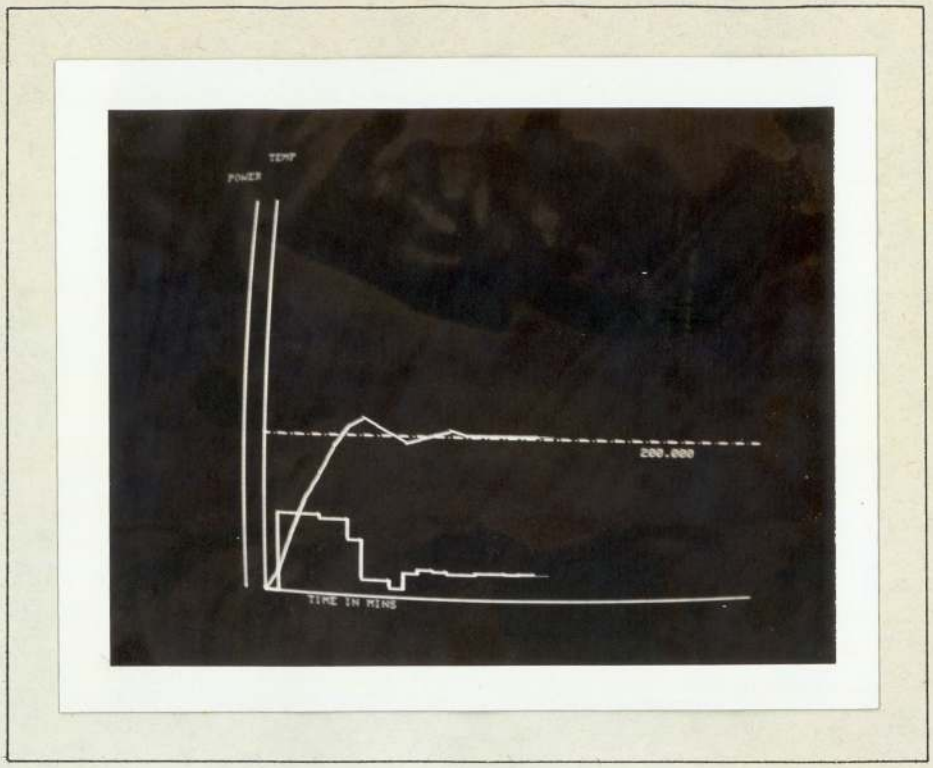


FIGURE 9.21 : Simulation output of DISPLY program on GT42 Display processor

SET POINT TEMP= 200.000

N	T	TMP	P
1	10.000	25.000	100.000
2	20.000	75.000	100.000
3	30.000	120.000	100.000
4	40.000	150.000	95.000
5	50.000	185.000	95.000
6	60.000	210.000	70.000
7	70.000	220.000	20.000
8	80.000	210.000	20.000
9	90.000	200.000	10.000
10	100.000	190.000	30.000
11	110.000	195.000	35.000
12	120.000	200.000	33.000
13	130.000	205.000	30.000
14	140.000	200.000	30.000
15	150.000	200.000	32.000
16	160.000	200.000	32.000
17	170.000	200.000	31.000
18	180.000	200.000	32.000
19	190.000	201.000	31.000
20	200.000	200.000	32.000
408	210.000	200.000	31.000

FIGURE 9.22 : Data file showing process variables.

procedures or failure detection methods or devices should be used must be carefully considered. Furthermore, the actual application program testing in the integration phase certainly needs sophisticated tools which are available on the Microprocessor Development Systems (MDS). The PROM simulator used for the tests, allows the simulation of a PROM for only one processor. The need of hardware and software testing tools required in a multi-microprocessor environment may surpass the cost-effectiveness hoped to be achieved by a multi-microprocessor system. These are just a few areas where further investigations are needed.

CHAPTER 10 - CONCLUSIONS

The research has shown that it is feasible to apply microprocessors for on-line parallel processing of information. Any application involved in using a multi-microprocessor system requires analysing the application so that the overall control problem is subdivided into smaller subproblems which are suitable for parallel execution on individual microprocessor-based systems, and any interactions between these subproblems are handled by communication links. The organisations of such systems range from locally distributed to geographically distributed microprocessor and microcomputer systems and a variety of applications range from homogenous to heterogenous applications. The communication links range from serial links to parallel links and man-machine to interprocess communications. It should be emphasised that a designer of such systems is required to balance, firstly, the distribution of hardware and software for the chosen application. Secondly, the application is required to be broken down into its information processing needs in the form of a top-down distribution of tasks and a bottom-up co-ordination of these tasks. Finally, since the hardware, software and tasks are distributed, the distribution of data and its flow to and from various tasks is of paramount importance.

A model of a processor within a distributed computing system which is proposed in this thesis specifically discusses its interfacing issues within a large-scale, real-

time complex system environment. It outlines the importance of application program development and its performance evaluation and monitoring. The four information links described in the model account for a variety of ways of data and control information distribution amongst the processors of the distributed computing system. The use of dual port memory modules as IANs and IDNs for data and control information distribution serve also as a buffered communication medium and provides new possibilities for communication protocols to be designed which are task-oriented.

The design of the Hierarchical Multi-microprocessor System Unit (HMSU) combines the IAN/IDN concept developed in the model and the resource sharing concept in the form of a master-slave relationship with respect to the access of common memory. A modular structure of the HMSU and its use as a building block allow other structures such as hierarchical, star, ring and combinations of these to be configured. The hardware design of the HMSU presented in the thesis is particularly organised using a Fairchild/ Mostek F8 microprocessor chips set mainly because of local software development facilities, such as a F8 cross-assembler on the MAXIMOP system, and F8 Evaluation Kits were available. However, since software development facilities are not included or superimposed on the processors of the HMSU, the task of application program development, its performance evaluation, monitoring and testing becomes particularly difficult. These problems are very vivid in the thesis when the HMSU is employed to implement

hierarchical control of the department's Travelling Load Furnace (TLF). If the luxury of providing a highly integrated and fault-tolerant system is to be envisaged, for example one processor taking control over the other in case of the failure of the second, the interfacing issues of the controlled process by the processors of a distributed processing system, such as the HMSU, requires special attention. The ability of a master processor or either of the slave processors to control any one section of the TLF not only requires modifications to the existing interfaces but also requires software diagnostic procedures or failure detection mechanisms to be implemented. A design proposal for modifying existing interfaces of the TLF and a simple mechanism of control mode selection procedure have been described for this purpose.

The research as a whole encompasses design of electronic circuits for input/output interfacing, design of F8 processors of the HMSU and the HMSU architecture, programming of control tasks for the processors of the HMSU in the F8 assembly language, programming of man-machine communication with respect to the control of the Travelling Load Furnace in a high-level language using the PDP-11/10 minicomputer and testing integration aspects of hardware and software developed. The last phase, namely the testing for integration of hardware and software closes a loop of the overall design cycles and the outcome begins to emerge in the form of problems encountered during practical implementation. These problems are highlighted and discussed in the thesis. In particular, the need for proper development

tools both at software and hardware level are vital to the development of the project. The suitability of the F8 processors for the HMSU, for example, can be questioned. The high chip count used in the design of ISMI could be minimised by the use of VLSI technology. Although the costs of CPUs and memory components are reducing the cost of putting these together in a multi-microprocessor system and the cost of writing software for such systems really brings up the cost-effectiveness issue, especially when the application involved is just one-off. These are some of the areas which may be in the realms of research for some time to come. As such, it is difficult to establish a direct relationship of the work undertaken to immediate industrial usage. However, this research will provide a useful benchmark for developing multi-microprocessor systems for hierarchical control of industrial processes.

FOOTNOTE: Further consideration is needed within the programs of the HMSU and the PDP11/10 minicomputer to ensure that critical parts are made interrupt proof, possibly through the implementation of Dijkstra's semaphore techniques (Dijkstra, 1968).

REFERENCES

AMD Data Sheet.

AMD Semiconductor Products, Data Sheet.

ANDERSON, G. A. and JENSON, E. D. (1975).

Computer Interconnection Structures: Taxonomy, Characteristics and Examples. ACM Computing Surveys, Vol. 7, no. 4, pp.197-213 (Dec. 1975).

ASPINALL, D. (1978) (Editor).

The microprocessor and its application. An advance course. Cambridge University Press, Cambridge.

BAILEY, W. N., GAYLER, J. R. and ROBERTS, P. D. (1979).

Introductory guide to using the department's PDP11/10 computer system, The City University, DSS/WNB-JRG-PDR/178 (Feb. 1979).

BARKER, H. A. (1978).

The microprocessors in control. IEE Control and Automation Division, Chairman's Address.

BIBBERO, R. J. (1977).

Microprocessors in Instruments and Control. John Willey and Sons, New York.

BILLINGSLEY, and SINGH, M. G. (1975).

On-line hierarchical control of large scale systems using multi-processors. IEE - 2nd Conference, Vol. 127, Part 21/25 (April 1975).

BROWN, T. J. (1979).

Elements of distributed control systems. Trends in on-line computer control system. IEE Conference, Vol. 172, Part 27-29 (March 1979).

CAFFIN, R. (1972).

The design and modelling of an experimental travelling load furnace. PhD Thesis: The City University, Department of Systems Science, London.

CARTER, J. W. (1978).

The problems of using microprocessors. Measurement and control. Vol.11, pp.81-97 (Feb. 1978).

DAVIES, A. C. (1977).

F8 Microprocessor User's Guide. The City University, Department of Electrical and Electronic Engineering, London.

DESHMUKH, H. A. (1977).

Microprocessor control of a Travelling Load Oven. MSc Thesis. The City University, Department of Systems Science, London.

DESHMUKH, H. A., SCOTT, R. G. F. and ROBERTS, P. D. (1979).
A hierarchically structured multi-microprocessor system. Microprocessors and their applications. Fifth EUROMICRO Symposium on Microprocessing and Microprogramming, Göteborg (Aug. 1979).

DATA SHEET for ADC82 (1979).
General Catalog, Burr-Brown.

DOWSING, R. D. (1978).
Introduction to system design. The microprocessor and its application. An advance course. Edited by ASPINALL, D., Cambridge University Press, pp.93-117.

EDGINGTON, J. H. (1979).
Control in the glass industry and future automation: PART 1. Measurement and Control, Vol. 12 (July 1979).

ELLIOTT, I. and ORGANICK (1978).
New directions in computer systems architecture. Large scale integration. EUROMICRO Symposium, Munich (Oct. 1978).

ENSLOW, P. H. ed. (1974).
Multiprocessors and parallel processing. John Wiley, New York.

ENSLOW, P. H. Jr. (1978).
What is a 'Distributed' Data Processing Systems? Computer pp.13-21 (Jan. 1978).

ENSLOW, P. H. Jr. (1978).
Multiprocessors and other parallel systems: An introduction and overview. Infotech State of the Art Report. Multiprocessor Systems, pp.219-262.

F8 USER'S GUIDE, Fairchild Microsystems (1976).

FEIERBACH, G. and STEVENSON, D. (1979).
The ILLIAC IV, Super Computers, Infotech State of the Art Report.

FITZGERALD, J. M. and FITZGERALD, A. F. (1973).
Fundamentals of Systems Analysis. John Wiley, New York.

FLYNN, M. J. (1972).
Some computer organizations and their effectiveness. IEEE Trans on Computers. Vol. C-21, no. 9, pp.948-960 (Sept. 1972).

HARRIS, J. A. and SMITH, D. R. (1977).
Hierarchical multiprocessor organisations. Fourth Symposium on Computer Architecture, University of Maryland.

HOLDING, D. J. and KING, P. J. (1979).
Experience with a distributed microprocessor control system in an industrial environment. Trends in on-line computer control system. IEE Conference. Vol. 172, Part 27-29 (March 1979).

HOLLAND, P. M. (1980).
Memory system design. IEE Colloquium on Hardware Design Techniques. Digest no. 1980/11, pp.4/1-4/5 (March 1980).

HUGHES, J. M. (1976).
Multiprocessor Navigation Systems. Digest of papers. Fall Compcon 76, pp.264-68.

IEE CONFERENCE PUBLICATION (1977).
Distributed computer control systems. IEE Conference Publication Vol. 153 (Sept. 1977).

JENSEN, E. D., THURBER, K. J. and SCHNEIDER (1979).
A review of systematic methods in distributed processor interconnection; Tutorial: Distributed Processor Communication Architecture; First International Conference on Distributed Computing System. Huntsville, Alabama (Oct. 1979).

JOSEPH, E. C. (1976).
Distributed processing architecture - past, present and future trends. Distributed Systems. Infotech State of the Art Report, pp.319-333.

KARTASHEV, S. I. and KARTASHEV, S. P. (1978).
Selection of the control organisation for a multicomputer system with dynamic architecture. Large Scale Integration EUROMICRO Symposium, pp.346-357 (Oct. 1978).

LEE, J. L. (1976).
Intersubsystem communications for process control. Instrumentation in the chemical and petroleum industries. Vol. 12. Proceedings of the 1976 Computer Interface Instrumentation System.

LOWE, E. I. and HIDDEN, A. E. (1971).
Computer Control in Process Industries. Peter Peregrinus Ltd.

MDS Z80 SYSTEM (1979).
MDS Z80 System reference manuals.

MILLER, G. A. (1956).
The magical number seven plus or minus two: Some limits on our capacity for processing information. Psychol. Rev. Vol. 63, pp.81-97.

MOSTEK, F8 Microprocessor hardware support.
Application Note, F8 Evaluation Kit.

MOTOROLA (1979).
Microcomputer Components, Motorola Semiconductors.

PATHAK, J. (1977).
Software setup eases traffic flow for multiprocessors. Electronics pp.108-112 (March 1977).

PDP-11/10 FORTRAN IV.
Language reference manual.

PDP-11/10 FORTRAN IV.
User's Guide.

PDP-11/10 System Reference Manual.

ROBERTS, P. D. (1979).
Introduction to large-scale control system. IEE Computing and Control Division. Specialist Seminar on Optimal Control of Large Scale Systems (Sept. 1979).

RUSSO, P. M. (1976).
An interface for multi-microprocessor systems. Digest of papers, Fall Comp. Con. pp.277-282.

RUSSO, P. M. (1977).
Interprocessor communication for multi-microcomputer systems. Computer Vol. 10, No. 4, pp.67-75 (April 1977).

SAVAS, E. S. (1965).
Computer Control of Industrial Processes. McGraw-Hill, New York.

SEARLE, B. C. and FREBERG, D. E. (1975).
Tutorial: Microprocessor application in multiple processor systems. Computer, pp 22-30 (Oct. 1975).

SHEENA, H. H. (1977).
Computer control of a travelling load furnace. PhD Thesis: The City University, Department of Systems Science, London.

SIEWIOREK, D. P. (1975).
Process co-ordination in multi-microprocessor systems. Euromicro Workshop, Nice (June 1975).

SMITH, C. L. (1972).
Digital Computer Process Control. Intext Educational Publishers.

SPENCER, J. P. (1976) (Editor).
Distributed Systems. Infotech State of the Art Report. Infotech International Ltd.

SRC (1977).
Distributed computing systems. Annual report. The Computing Science Committee of the Science Research Council (Sept. 1977-Sept. 1978).

SRC (1980).
Distributed computing systems. The Computing Science Committee of the Science Research Council (Sept. 1979-Sept. 1980).

STEINCHOFF, J. and McGILL, R. (1976).
An approach solving scientific problems using multiple-microprocessors. EUROMICRO Symposium, Venice, pp.285-293 (Oct. 1976).

SWAN, R. J., FULLER, S. H. and SIEWIOREK, D. P. (1977).
Cm* - A modular multimicroprocessor. AFIPS Conference Proceedings, AFIPS Press, Vol. 46, pp.637-643.

TANAKA, Y., MIYASHITA, K., KOYAMA, S., MIYAMOTO, E. and TSUDA, T. (1976).
HARPS - A new hierarchical array processor system. EUROMICRO Symposium, Venice, pp.91-98 (Oct. 1976).

THURBER, K. J. and WALD, L. D. (1975).
Associative and parallel processors. Computing Surveys Vol. 7, No. 4, pp.215-255 (Dec. 1975).

THE TTL DATA BOOK FOR DESIGN ENGINEERS,
2nd Edition.

VT-11 GRAPHIC SUPPORT.
PDP-11 Reference Manual.

WEISSBERGER, A. J. (1975).
Microprocessors simplify control system. Canadian Electronic Engineering (June 1975).

WEISSBERGER, A. J. (1977).
Analysis of multiple microprocessor system architectures. Computer Design, pp.151-163 (June 1977).

WHITE, C. H. (1976) (Editor).
Distributed Systems. Infotech State of the Art Report.

WILKIE, J. D. F. (1979).
A microprocessor philosophy for process control systems. Trend in on-line computer control system. IEE Conference Publication, vol. 172, pp.27-29 (March 1979).

WITTEN, J. H. and RICHARD, L. J. (1978).
Processor-processor dialogue through existing input-output channels. Computer and Digital Techniques, Vol. 1, No. 4 (Oct. 1978).

WULF, W. A. and BELL, C. G. (1972).
C.mmp - A multimini processor. Fall Joint Computer Conference, AFIPS Proc., Vol. 41, Montvale, N.J.: AFIPS Press.

DIJKSTRA, E. W. (1968).
Co-operating sequential programming. Programming Languages. Genuys, F. (Editor). Academic Press, London.

APPENDIX A - HARDWARE DETAILS OF THE F8 MICROPROCESSOR

The three F8 microprocessor boards built for the HMSU are identical. Each board consists of the following:

1. One - 3850 CPU (Central Processing Unit)
2. Two - 3861 PIO (Peripheral Input/Output) Chips (i.e. versions MK 90002 and MK 90003)
3. One - 3851 PSU (Program Storage Unit)
4. One - 3853 SMI (Static Memory Interface)
5. Two - 2708 EPROM chips (i.e. 2 kilobytes of PROM memory)
6. Eight - 2102 Static RAM chips (i.e. 1 kilobyte of static RAM memory).

A detailed circuit diagram for the F8 microprocessor board is shown in Figure A1. The inclusion of 3851 PSU in which the DDT-1 (Designer's Development Tool 1) program resides, allows the testing of the F8 microprocessor circuit board. However, this PSU chip cannot be used when EPROM chips containing the HMSU control program are used. The reason for this is that the DDT-1 program and the HMSU control program both start at H'0000' address. Thus, only one program can be run at a time. Additionally, when the HMSU control program is to be used, the PRIORITY OUT line from the 3861 PIO (MK 90003 version) chip is directly connected to the PRIORITY IN line of the 3853 SMI chip. The 3850 CPU chip is provided with manual reset (switch S1) and automatic "Power ON" reset inputs. These input lines are connected to EXT RESET input of the CPU through 7432 OR

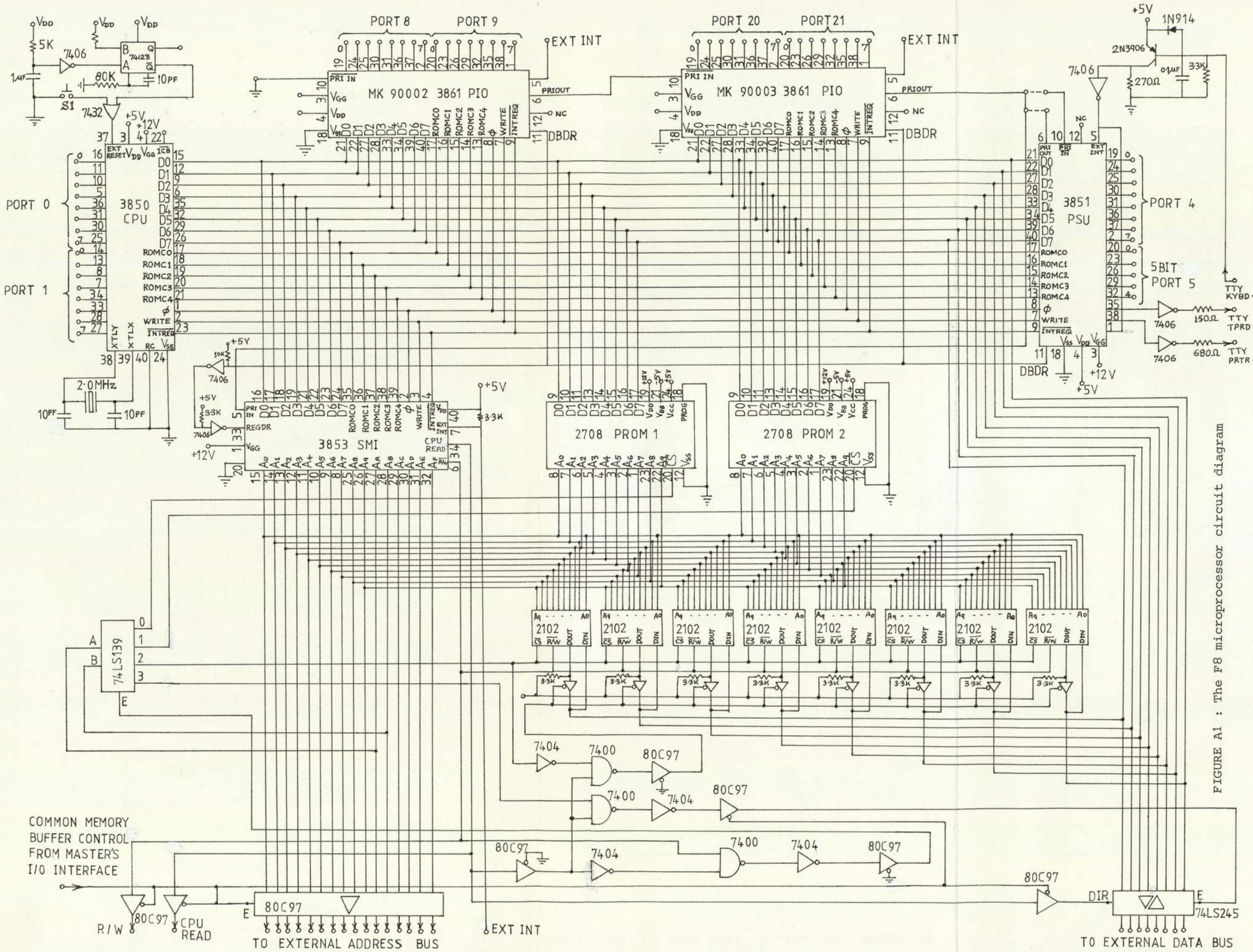


FIGURE A1 : The F8 microprocessor circuit diagram

gate. The F8 microprocessor board mainly provides six 8-bit Input/Output ports, three external interrupt lines, sixteen external address lines (i.e. address bus) and eight bidirectional external data lines (i.e. data bus). All the IC chips use wire-wrap sockets which are mounted on the DIP vero board (No. 10-0154L). One of the F8 microprocessor circuit boards is shown in Figure A2. Figure A3 illustrates the two sides of ISMI circuit board and Figure A4 shows the HMSU on the background of the Travelling Load Furnace (TLF).

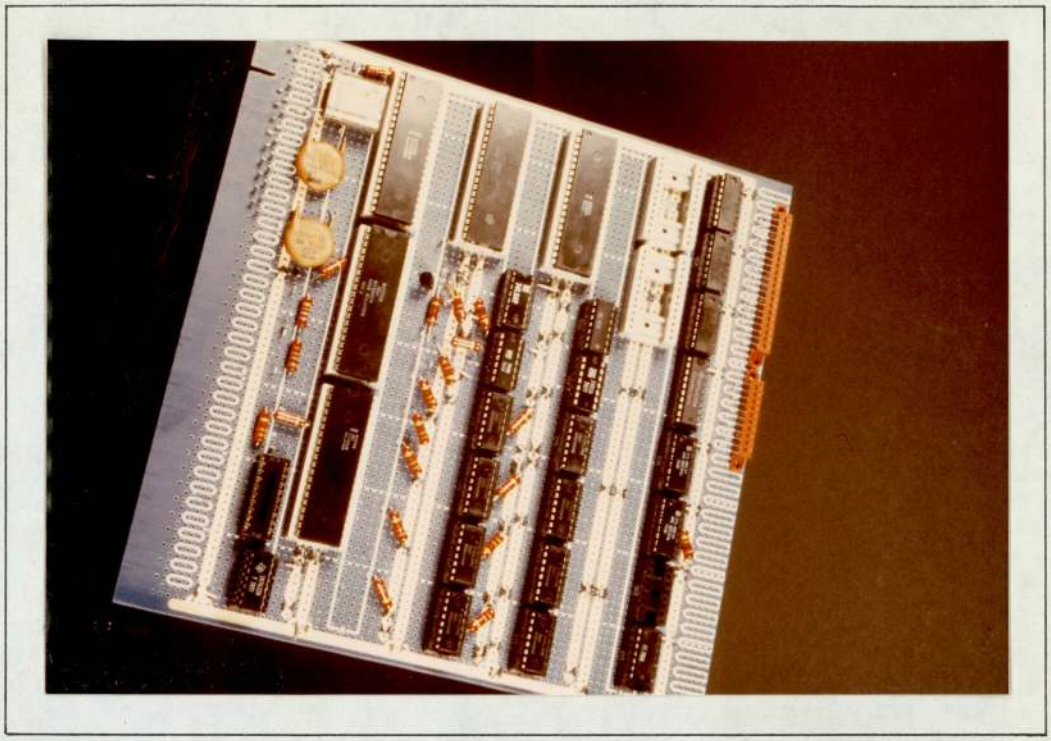


FIGURE A2 : The F8 microprocessor circuit board

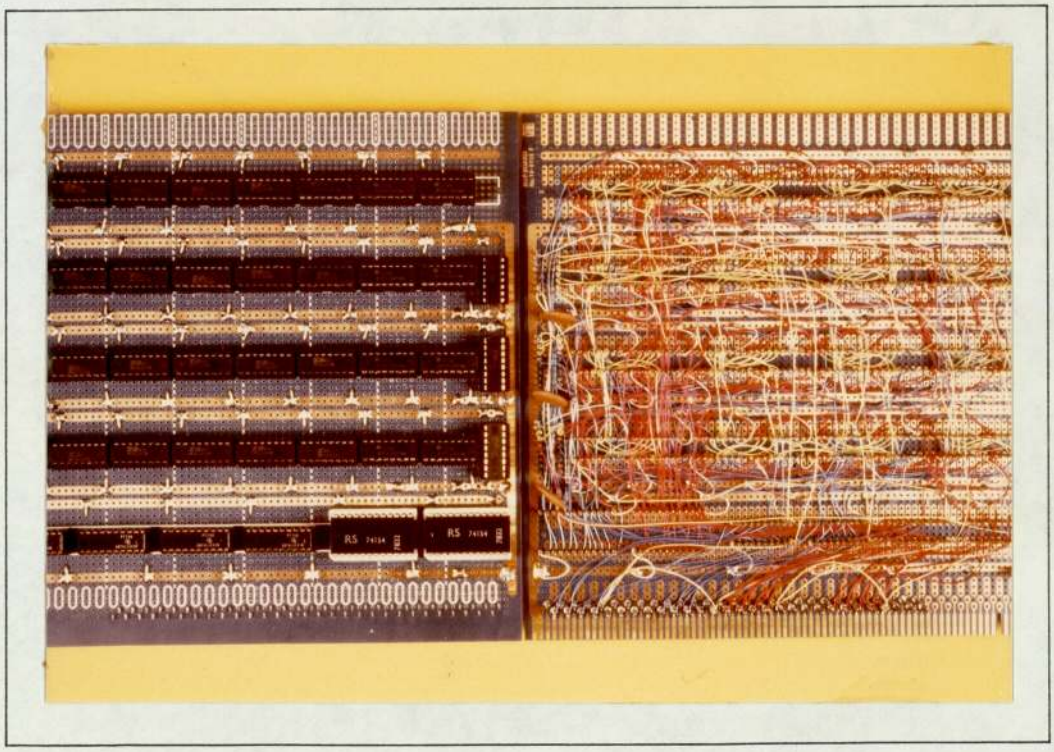


FIGURE A3 : The ISMI circuit boards

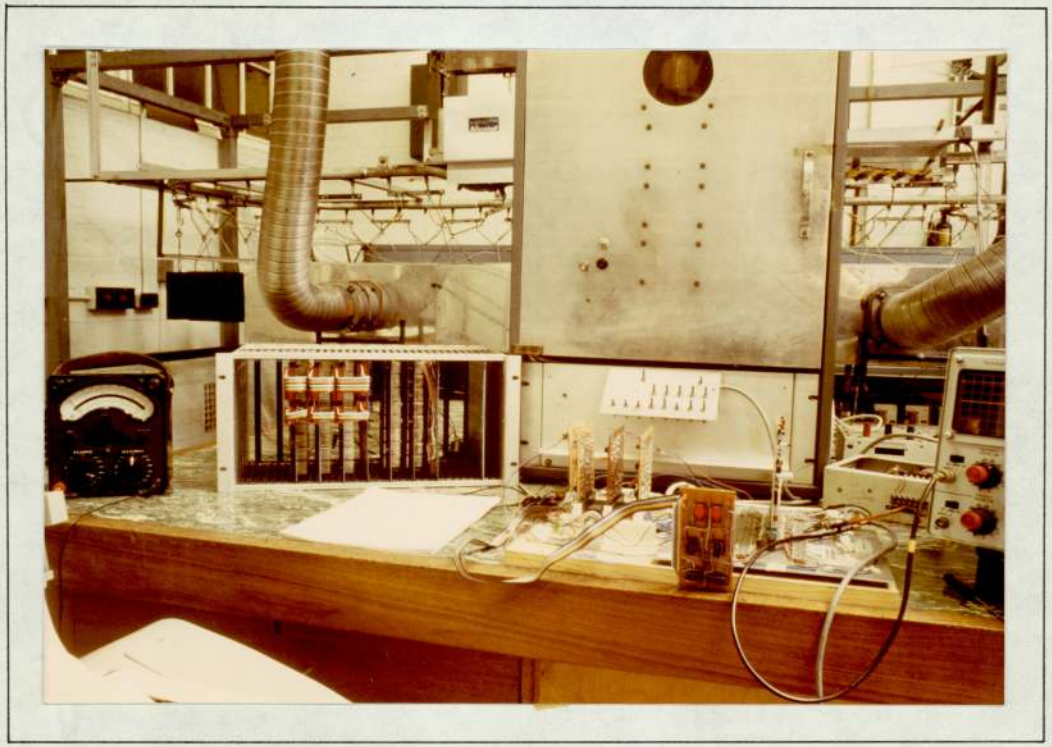


FIGURE A4 : The HMSU on the background of the TLF

APPENDIX B - THE F8 PROGRAMMING FEATURES

This appendix covers a brief description of the F8 Evaluation Kit, some important programming features unique to the F8 microprocessor and the F8 instruction set.

B1 THE F8 EVALUATION KIT

The F8 Evaluation Kit built by MOSTEK consists of minimum hardware system containing 3850 CPU, 3851 PSU, 3853 SMI and 1 kilobyte of static memory RAM and a Teletype interface (20 mA loop). The Designer's Development Tool 1 (DDT-1) program resides in the 3851 PSU which is located in the low order 1 kilobyte of memory space (i.e. H'0000' to H'03FF'). The RAM address space range from H'0400' to H'07FF'. All eight bits of Port 0, Port 1 and Port 4 are available to the user, providing 24 I/O lines. A selection 110 or 300 baud Teletype rate is available from Port 5.

The DDT-1 program serves a convenient means for evaluating the F8 and the debugging of application programs. A summary of the commands accepted by the DDT-1 is as follows:

1. B - Breakpoint (software) address.
   Format: B aaaa, where aaaa is a breakpoint address.
2. C - Copy memory arrays.
   Format: C ssss, ffff, dddd, where ssss = start address, ffff = finish address and dddd = destination address
3. D - Dump memory onto paper tape.
   Format: D ssss, ffff, where ssss = start address and ffff = finish address.

4. E - Execute at specific address.

   Format: E ssss, where ssss = start address.

5. H - Hexadecimal arithmetic operations.

   Format: H a + b = result or H aaaa + bbbb - cccc = result.

6. L - Load memory from paper tape.

7. M - Memory content display and modify.

   Format: M aaaa, where aaaa = address of memory location.

8. P - Port content display and modify.

   Format: P pp, where pp is the port address to be examined or modified.

9. T - Type memory content array.

   Format: T ssss, ffff, where ssss = start address and ffff = finish address of the memory block to be printed.

B2 IMPORTANT PROGRAMMING FEATURES

1. When power is turned on, all PCO (program counter registers) in the F8 microprocessor system are set to 0. Therefore, the first instruction executed is located at memory byte 0. Thus, the first program to be executed must be originated at H'0000'.

2. A subroutine linkage is associated with calling from and returning to the main program. There are two instructions used to call a subroutine into execution:
(a) Instruction PK saves the contents of the program counter (PCO) in the stack register (PC1), then loads the

subroutine starting address from the K register into the program counter.

(b) Instruction PI saves the contents of the program counter in the stack register. It then loads the subroutine starting address (which is in the two bytes of object program following the PI op code byte) into the program counter.

Similarly, there are two ways to return from subroutines:

(a) Instruction POP moves the contents of the stack register back to PCO.

(b) Instruction Pk may also be used to return from a subroutine by having the return address in the k registers.

If, for example, subroutines are nested two deep, the following steps show the call and return sequence:

Initially, outer routine start address is put in k: <k> = b

Outer Call	Pk	<PC0> → PC1	a → PC1
		<K> → PC0	b → PC0

Save PC1 in k in preparation for inner call:

	LR K,P	<PC1> → k	a → k
Inner Call	PI	<PC0> → PC1	c → PC1
		(.+1)(.+2) → PC0	e → PC0
Inner Return	POP	<PC1> → PC0	c → PC0
Outer Return	Pk	<PC0> → PC1	d → PC1
		<K> → PC0	a → PC0

where a, b, c, d and e are 16 bit addresses.

For nesting to greater depth, a stack for return addresses is required to be set up.

3. The basic interrupt handling capacity is a micro-programmed function of the 3850 CPU. The sequence of events surrounding an interrupt is as follows:

(a) For interrupts to be processed, interrupts must be enabled within the 3850 CPU and at the chip receiving the interrupt request signal (i.e. 3861 or 3851 or 3853 chips).

(b) When more than one device simultaneously request to interrupt the 3850 CPU, priorities are determined on the basis of 'daisy-chaining'. The daisy-chain sequence is a hardware feature of an F8-microprocessor system.

(c) When a valid interrupt request signal is detected by the 3850 CPU, it ceases current program execution at the conclusion of the instruction currently being executed. However, an interrupt will not be acknowledged at the conclusion of the following privileged instructions:

```
Pk
PI
POP
JMP
OUTS (except 0,1)
OUT
EI
LR  W,J
```

(d) The 3850 CPU SENDS out an interrupt acknowledge signal. It is the way in which this signal is trapped that implements interrupt priority, when more than one interrupt request line is true, as described in step (b).

(e) When the 3850 CPU sends out an interrupt acknowledge signal, it clears the interrupt enable status within the 3850 CPU thus disabling all subsequent interrupts.

(f) The chip that traps the interrupt acknowledge signal output in step (e) responds by transmitting the contents of its interrupt address register as the next contents of PC0 register. These interrupt addresses are as follows for different chips:

CHIP	INTERRUPT ADDRESSES	
	TIMER	EXTERNAL
3851 PSU	Non-programmable mask option	
3861 PIO (MK 90002)	H'0340'	H'03C0'
3861 PIO (MK 90003)	H'0320'	H'03A0'
3853 SMI	Programmable option	

(g) The PSU or SMI logic moves the contents of PC0 to PC1 and then loads the address from step (f) into PC0. Thus, a program dedicated to the acknowledged interrupt request line is executed.

B3 THE F8 INSTRUCTION SET

The following pages describe the F8 instruction set.

The F8 instruction set
(p.245-247)
has been removed
for copyright reasons

APPENDIX C - THE HMSU PROGRAM LISTING

The following HMSU program is for the master processor of the HMSU:

```
                                        MOSTEK 3870/F8 CROSS ASSEMBLER PAGE 0001
 ADDR  OBJECT FLAG ST # SOURCE STATEMENT       DATASET = DK0:HMSU  .SRC

                    0001 ;TITLE-HMSU-AN ASSEMBLY LANGUAGE PROGRAMME FOR MASTER PR
OCESSOR
 >0800              0002 C1      EQU     H'0800'
 >0801              0003 C2      EQU     H'0801'
 >0802              0004 C3      EQU     H'0802'
 >0803              0005 LA      EQU     H'0803'
 >0804              0006 NS      EQU     H'0804'
 >0805              0007 MS      EQU     H'0805'
 >0806              0008 CM      EQU     H'0806'
 >0807              0009 RUNNO   EQU     H'0807'
 >0808              0010 ISP1    EQU     H'0808'
 >0809              0011 K11     EQU     H'0809'
 >080A              0012 K12     EQU     H'080A'
 >080B              0013 K13     EQU     H'080B'
 >080C              0014 K14     EQU     H'080C'
 >080D              0015 SI1     EQU     H'080D'
 >0810              0016 ISP2    EQU     H'0810'
 >0811              0017 K21     EQU     H'0811'
 >0812              0018 K22     EQU     H'0812'
 >0813              0019 K23     EQU     H'0813'
 >0814              0020 K24     EQU     H'0814'
 >0815              0021 SI2     EQU     H'0815'
 >0818              0022 ISP3    EQU     H'0818'
 >0819              0023 K31     EQU     H'0819'
 >081A              0024 K32     EQU     H'081A'
 >081B              0025 K33     EQU     H'081B'
 >081C              0026 K34     EQU     H'081C'
 >081D              0027 SI3     EQU     H'081D'
 >083C              0028 RCPS    EQU     H'083C'        ;READ COUNTER PDP SET
 >083D              0029 RFPS    EQU     H'083D'        ;READ FLAG PDP SET
 >083E              0030 WFCPS   EQU     H'083E'        ;WRITE FLAG COUNTER PDP
SET
 >083F              0031 WFPS    EQU     H'083F'        ;WRITE FLAG PDP SET
 >0040              0032 ZON0    EQU     H'40'          ;HEATING ZONE ADDRESSES:
 >0041              0033 ZON1    EQU     H'41'
 >0042              0034 ZON2    EQU     H'42'
 >0043              0035 ZON3    EQU     H'43'
 >0044              0036 ZON4    EQU     H'44'
 >0045              0037 ZON5    EQU     H'45'
 >0046              0038 ZON6    EQU     H'46'
 >0047              0039 ZON7    EQU     H'47'
 >0050              0040 MOTA    EQU     H'50'          ;CONVEYOR MOTOR ADDRESS
 >0840              0041 LZAC    EQU     H'0840'        ;LOAD & ZONE ADDRESS COU
NTER
 >0841              0042 SZONA   EQU     H'0841'        ;STARTING ZONE ADDRESS
 >0842              0043 TSZONA  EQU     H'0842'        ;TEMP STARTING ZONE ADDR
ESS
 >0843              0044 RFCMS   EQU     H'0843'        ;READ FLAG COUNTER MASTE
R SET
 >0844              0045 PIDFLG  EQU     H'0844'        ;PID FLAG
 >0845              0046 ISMIFG  EQU     H'0845'        ;ISMI FLAG
 >0846              0047 MSFLG   EQU     H'0846'        ;MICRO SWITCH FLAG
 >0847              0048 CMFLG   EQU     H'0847'        ;COMMON MEMORY FLAG
 >0848              0049 SLA     EQU     H'0848'        ;STARTING LOAD ADDRESS
 >0849              0050 TSLA    EQU     H'0849'        ;TEMP STARTING LOAD ADDR
ESS
 >084A              0051 TLZAC   EQU     H'084A'        ;TEMP LOAD & ZONE ADD CO
UNTER
 >087F              0052 SPA0    EQU     H'087F'        ;STORED PORT ADD FOR MAS
TER
 >08BF              0053 SPA1    EQU     H'08BF'        ;STORED PORT ADD FOR SLA
VE1
 >08FF              0054 SPA2    EQU     H'08FF'        ;STORED PORT ADD FOR SLA
VE2
 >084C              0055 RFMS    EQU     H'084C'        ;READ FLAG MASTER SET
 >084F              0056 WCMS    EQU     H'084F'        ;WRITE COUNTER MASTER SE
T
 >0850              0057 WFMS    EQU     H'0850'        ;WRITE FLAG MASTER SET
 >087E              0058 MTRF    EQU     H'087E'        ;MASTER TRANSMITT FLAG
```

```
 ADDR  OBJECT FLAG ST # SOURCE STATEMENT       DATASET = DK0:HMSU  .SRC

 >08BE              0059 S1TRF   EQU     H'08BE'          ;SLAVE1 TRANSMITT FLAG
 >08FE              0060 S2TRF   EQU     H'08FE'          ;SLAVE2 TRANSMITT FLAG
 >0851              0061 TRF     EQU     H'0851'          ;TRANSMISSION FLAG
 >0852              0062 TRFC    EQU     H'0852'          ;TRANSMISSION FLAG COUNT
ER
 >0A00              0063 SSA     EQU     H'0A00'          ;STARTING STACK ADDRESS
 >0853              0064 MSCNT   EQU     H'0853'          ;MICRO SWITCH COUNTER
 >0854              0065 SNO     EQU     H'0854'          ;SAMPLE NUMBER STORE
 >0855              0066 ANSWER  EQU     H'0855'          ;TEMP STORE FOR CALCULAT
ED POWER
 >0856              0067 TRUNNO  EQU     H'0856'          ;TEMP STORE FOR RUNNO
 >0870              0068 RECO1   EQU     H'0870'          ;LOOP1 RECORD ADDRESS
 >0875              0069 RECO2   EQU     H'0875'          ;LOOP2 RECORD ADDRESS
 >087A              0070 RECO3   EQU     H'087A'          ;LOOP3 RECORD ADDRESS
                    0071 ;INITIALIZATION PROCEDURE STARTS HERE.................
 >0000              0072         ORG     H'0000'          ;PROGRAM-HMSU STARTS AT
0000
'0000  1A          0073         DI                       ;DISABLE ALL INTERRUPTS
AT CPU
'0001  70          0074         CLR                      ;CLEAR I/O PORTS
'0002  B0          0075         OUTS    0                ;NO-0
'0003  B1          0076         OUTS    1                ;NO-1
'0004  B8          0077         OUTS    8                ;NO-8
'0005  B9          0078         OUTS    9                ;NO-9
'0006  BA          0079         OUTS    H'0A'            ;NO-A
'0007  2720        0080         OUT     H'20'            ;NO-20
'0009  2721        0081         OUT     H'21'            ;NO-21
'000B  2722        0082         OUT     H'22'            ;NO-22
'000D  67          0083         LISU    7                ;CLEAR CONTROL LOOP BUFF
ER
'000E  280407'   A 0084         PI      ZERO             ;0'77'-0'70'
'0011  66          0085         LISU    6                ;0'67'-0'60'
'0012  280407'   A 0086         PI      ZERO
'0015  65          0087         LISU    5                ;0'57'-0'50'
'0016  280407'   A 0088         PI      ZERO
'0019  28040E'   A 0089         PI      SHUT             ;JUMP TO SHUTDOWN SUBROU
TINE
'001C  2A0800      0090         DCI     C1
'001F  20FF        0091         LI      H'FF'
'0021  50          0092         LR      0,A
'0022  70          0093         LIS     0
'0023  17          0094 AGA     ST
'0024  30          0095         DS      0
'0025  94FD        0096         BNZ     AGA
'0027  20FF        0097         LI      H'FF'            ;CLOSE TIMER AT PIO-2
'0029  2723        0098         OUT     H'23'
'002B  71          0099         LIS     1                ;EXT INT ENABLED AT PIO-
2
'002C  2722        0100         OUT     H'22'
'002E  20FD        0101         LI      H'FD'            ;253 TIMER MAX COUNT LOA
DED IN
'0030  BB          0102         OUTS    H'0B'            ;PORT-B OF PIO-1
'0031  73          0103         LIS     3                ;TIMER INT ENABLED AT PI
O-1
'0032  BA          0104         OUTS    H'0A'
'0033  71          0105         LIS     1                ;SMI VECTOR ADD PORTS H'
0C'-H'0D'
'0034  BC          0106         OUTS    H'0C'            ;ARE LOADED WITH H'0280'
'0035  20F0        0107         LI      H'F0'
'0037  BD          0108         OUTS    H'0D'
'0038  71          0109         LIS     1                ;EXT INT ENABLED AT SMI
'0039  BE          0110         OUTS    H'0E'
'003A  63          0111         LISU    3
'003B  68          0112         LISL    0
'003C  20FD        0113         LI      H'FD'
'003E  5D          0114         LR      I,A
'003F  201E        0115         LI      30
'0041  5D          0116         LR      I,A
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DKO:HMSU .SRC

```
'0042  200A          0117          LI      H'0A'
'0044  5C            0118          LR      S,A
'0045  70            0119          LIS     0
'0046  2A0844        0120          DCI     PIDFLG
'0049  17            0121          ST
'004A  2A0854        0122          DCI     SNO
'004D  17            0123          ST
'004E  2A0845        0124          DCI     ISMIFG
'0051  71            0125          LIS     1
'0052  17            0126          ST
                     0127 ;MAIN PROGRAM - LOWEST PRIORITY ROUTINE STARTS HERE.....
'0053  1B            0128 BUY      EI
'0054  2A0844        0129          DCI     PIDFLG
'0057  16            0130          LM
'0058  2501          0131          CI      1
'005A  8413          0132          BZ      PID
'005C  2A0845        0133          DCI     ISMIFG
'005F  16            0134          LM
'0060  2501          0135          CI      1
'0062  8415          0136          BZ      ISMI
'0064  2A0851        0137          DCI     TRF
'0067  16            0138          LM
'0068  2501          0139          CI      1
'006A  8408          0140          BZ      TRMITT
'006C  90E6          0141          BR      BUY
'006E  290489'  A    0142 PID      JMP     PIDR
'0071  90E1          0143          BR      BUY
'0073  290248'  A    0144 TRMITT   JMP     TRMIT
'0076  90DC          0145          BR      BUY
                     0146 ;ISMI - ROUTINE LOOKS FOR NEW DATA INPUT FROM PDP-11....
'0078  2040          0147 ISMI     LI      H'40'
'007A  2720          0148          OUT     H'20'
'007C  2621          0149 STAY     IN      H'21'
'007E  2501          0150          CI      1
'0080  84FB          0151          BZ      STAY
'0082  2A083E        0152 FULL     DCI     WFCPS
'0085  280400'  A    0153          PI      CLEAR
'0088  2041          0154          LI      H'41'
'008A  2720          0155          OUT     H'20'
'008C  2621          0156          IN      H'21'
'008E  8D            0157          CM
'008F  840D          0158          BZ      SAME
'0091  280191'  A    0159          PI      COPY
'0094  63            0160          LISU    3
'0095  69            0161          LISL    1
'0096  2A080D        0162          DCI     SI1
'0099  16            0163          LM
'009A  5C            0164          LR      S,A
'009B  9004          0165          BR      TEST1
'009D  290189'  A    0166 SAME     JMP     BUYOT
'00A0  2A0800        0167 TEST1    DCI     C1
'00A3  16            0168          LM
'00A4  25FF          0169          CI      H'FF'
'00A6  8403          0170          BZ      SET1
'00A8  9006          0171          BR      X1
'00AA  2A087F        0172 SET1     DCI     SPA0
'00AD  71            0173          LIS     1
'00AE  17            0174          ST
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'00AF  2A0801        0175 X1       DCI     C2
'00B2  16            0176          LM
'00B3  25FF          0177          CI      H'FF'
'00B5  8403          0178          BZ      SET2
'00B7  9006          0179          BR      X2
'00B9  2A08BF        0180 SET2     DCI     SPA1
'00BC  71            0181          LIS     1
'00BD  17            0182          ST
'00BE  2A0802        0183 X2       DCI     C3
'00C1  16            0184          LM
'00C2  25FF          0185          CI      H'FF'
'00C4  8404          0186          BZ      SET3
'00C6  290184'   A   0187          JMP     BUYOUT
'00C9  2A08FF        0188 SET3     DCI     SPA2
'00CC  71            0189          LIS     1
'00CD  17            0190          ST
'00CE  2A0800        0191 TEST2    DCI     C1
'00D1  16            0192          LM
'00D2  25FF          0193          CI      H'FF'
'00D4  8404          0194          BZ      ON
'00D6  2901F0'   A   0195          JMP     CMAR
'00D9  2A0806        0196 ON       DCI     CM
'00DC  16            0197          LM
'00DD  2501          0198          CI      H'01'
'00DF  8414          0199          BZ      UUU
'00E1  2A0806        0200          DCI     CM
'00E4  16            0201          LM
'00E5  2502          0202          CI      H'02'
'00E7  8455          0203          BZ      VVV
'00E9  2A0806        0204          DCI     CM
'00EC  16            0205          LM
'00ED  2503          0206          CI      H'03'
'00EF  841C          0207          BZ      WWW
'00F1  290184'   A   0208          JMP     BUYOUT
'00F4  2A0840        0209 UUU      DCI     LZAC
'00F7  72            0210          LIS     2
'00F8  17            0211          ST
'00F9  2A0848        0212          DCI     SLA
'00FC  2C            0213          XDC
'00FD  2A0803        0214          DCI     LA
'0100  16            0215          LM
'0101  2C            0216          XDC
'0102  17            0217          ST
'0103  2A0841        0218          DCI     SZONA
'0106  2040          0219          LI      ZONO
'0108  17            0220          ST
'0109  290184'   A   0221          JMP     BUYOUT
'010C  2A0840        0222 WWW      DCI     LZAC
'010F  73            0223          LIS     3
'0110  17            0224          ST
'0111  2A0848        0225          DCI     SLA
'0114  2C            0226          XDC
'0115  2A0803        0227          DCI     LA
'0118  16            0228          LM
'0119  2402          0229          AI      H'02'
'011B  50            0230          LR      0,A
'011C  253E          0231          CI      H'3E'
'011E  840B          0232          BZ      LAO
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'0120  40            0233          LR      A,0
'0121  253F          0234          CI      H'3F'
'0123  840C          0235          BZ      LAN
'0125  2C            0236          XDC
'0126  40            0237          LR      A,0
'0127  17            0238          ST
'0128  900B          0239          BR      BUY1
'012A  2020          0240 LA0      LI      H'20'
'012C  2C            0241          XDC
'012D  17            0242          ST
'012E  9005          0243          BR      BUY1
'0130  2021          0244 LAN      LI      H'21'
'0132  2C            0245          XDC
'0133  17            0246          ST
'0134  2A0841        0247 BUY1     DCI     SZONA
'0137  2042          0248          LI      ZON2
'0139  17            0249          ST
'013A  290053'   A   0250          JMP     BUY
'013D  2A0840        0251 VVV      DCI     LZAC
'0140  73            0252          LIS     3
'0141  17            0253          ST
'0142  2A0848        0254          DCI     SLA
'0145  2C            0255          XDC
'0146  2A0803        0256          DCI     LA
'0149  16            0257          LM
'014A  2405          0258          AI      H'05'
'014C  50            0259          LR      0,A
'014D  253E          0260          CI      H'3E'
'014F  841A          0261          BZ      LA0
'0151  40            0262          LR      A,0
'0152  253F          0263          CI      H'3F'
'0154  8419          0264          BZ      LA1
'0156  40            0265          LR      A,0
'0157  2540          0266          CI      H'40'
'0159  8418          0267          BZ      LA2
'015B  40            0268          LR      A,0
'015C  2541          0269          CI      H'41'
'015E  8417          0270          BZ      LA3
'0160  50            0271          LR      0,A
'0161  2542          0272          CI      H'42'
'0163  8416          0273          BZ      LA4
'0165  2C            0274          XDC
'0166  40            0275          LR      A,0
'0167  17            0276          ST
'0168  9015          0277          BR      BUY2
'016A  2020          0278 LA0      LI      H'20'
'016C  900F          0279          BR      BUY3
'016E  2021          0280 LA1      LI      H'21'
'0170  900B          0281          BR      BUY3
'0172  2022          0282 LA2      LI      H'22'
'0174  9007          0283          BR      BUY3
'0176  2023          0284 LA3      LI      H'23'
'0178  9003          0285          BR      BUY3
'017A  2024          0286 LA4      LI      H'24'
'017C  2C            0287 BUY3     XDC
'017D  17            0288          ST
'017E  2A0841        0289 BUY2     DCI     SZONA
'0181  2045          0290          LI      ZON5
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'0183  17          0291          ST
'0184  2A0844      0292 BUYOUT   DCI     PIDFLG
'0187  71          0293          LIS     1
'0188  17          0294          ST
'0189  2A0845      0295 BUYOT    DCI     ISMIFG
'018C  70          0296          LIS     0
'018D  17          0297          ST
'018E  290053'  A  0298          JMP     BUY
                   0299 ;COPY SUBROUTINE COPIES ISMI INTO MASTER'S PRIVATE MEMOR
Y
                   0300 ;AND COMMON MEMORY OF THE HMSU SYSTEM.
'0191  08          0301 COPY     LR      K,P
'0192  1A          0302          DI
'0193  71          0303          LIS     1
'0194  2721        0304          OUT     H'21'
'0196  2080        0305          LI      H'80'
'0198  2720        0306          OUT     H'20'
'019A  70          0307          LIS     0
'019B  B9          0308          OUTS    9
'019C  2720        0309          OUT     H'20'
'019E  2A0C00      0310          DCI     H'0C00'
'01A1  2C          0311          XDC
'01A2  2A0800      0312          DCI     H'0800'
'01A5  2040        0313          LI      H'40'           ;COUNT 64 LOADED IN REG
0
'01A7  50          0314          LR      0,A
'01A8  207F        0315          LI      H'7F'           ;READ ADD LOADEDIN REG 1
'01AA  41          0316 REPT     LR      A,1
'01AB  2720        0317          OUT     H'20'
'01AD  2621        0318          IN      H'21'
'01AF  17          0319          ST
'01B0  2C          0320          XDC
'01B1  17          0321          ST
'01B2  2C          0322          XDC
'01B3  70          0323          LIS     0
'01B4  2720        0324          OUT     H'20'
'01B6  2721        0325          OUT     H'21'
'01B8  31          0326          DS      1
'01B9  30          0327          DS      0
'01BA  94EF        0328          BNZ     REPT
'01BC  2A083E      0329          DCI     WFCPS
'01BF  70          0330          LIS     0
'01C0  2721        0331          OUT     H'21'
'01C2  2080        0332          LI      H'80'
'01C4  2720        0333          OUT     H'20'
'01C6  70          0334          LIS     0
'01C7  2720        0335          OUT     H'20'
'01C9  16          0336          LM
'01CA  2721        0337          OUT     H'21'
'01CC  2081        0338          LI      H'81'
'01CE  2720        0339          OUT     H'20'
'01D0  70          0340          LIS     0
'01D1  2720        0341          OUT     H'20'
'01D3  2721        0342          OUT     H'21'
'01D5  1B          0343          EI
'01D6  0C          0344          PK                      ;RETURN
                   0345 ;COMMON MEMORY ROUTINE MAKES COMMON DATA AVAILABLE TO OT
HER SLAVE
                   0346 ;PROCESSORS.
 >01F0             0347          ORG     H'01F0'
'01F0  08          0348 CMAR     LR      K,P
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'01F1  54            0349          LR      4,A
'01F2  1E            0350          LR      J,W
'01F3  0A            0351          LR      A,IS
'01F4  55            0352          LR      5,A
'01F5  280435'   A   0353          PI      CALL
'01F8  1B            0354          EI
'01F9  2A08FF        0355          DCI     SPA2
'01FC  16            0356          LM
'01FD  2501          0357          CI      1
'01FF  8413          0358          BZ      SLV2
'0201  2A08BF        0359          DCI     SPA1
'0204  16            0360          LM
'0205  2501          0361          CI      1
'0207  8414          0362          BZ      SLV1
'0209  2A087F        0363          DCI     SPA0
'020C  16            0364          LM
'020D  2501          0365          CI      1
'020F  8413          0366          BZ      SLV0
'0211  902D          0367          BR      OUTT
'0213  2A08FF        0368 SLV2     DCI     SPA2
'0216  70            0369          LIS     0
'0217  17            0370          ST
'0218  72            0371          LIS     2
'0219  B9            0372          OUTS    9
'021A  9024          0373          BR      OUTT
'021C  2A08BF        0374 SLV1     DCI     SPA1
'021F  70            0375          LIS     0
'0220  17            0376          ST
'0221  71            0377          LIS     1
'0222  B9            0378          OUTS    9
'0223  2A087F        0379 SLV0     DCI     SPA0
'0226  70            0380          LIS     0
'0227  17            0381          ST
'0228  B9            0382          OUTS    9
'0229  2A0851        0383          DCI     TRF
'022C  71            0384          LIS     1
'022D  17            0385          ST
'022E  2080          0386          LI      H'80'
'0230  50            0387          LR      0,A
'0231  2A0880        0388          DCI     H'0880'
'0234  2C            0389          XDC
'0235  2A0C80        0390          DCI     H'0C80'
'0238  16            0391 REP      LM
'0239  2C            0392          XDC
'023A  17            0393          ST
'023B  2C            0394          XDC
'023C  30            0395          DS      0
'023D  94FA          0396          BNZ     REP
'023F  280460'   A   0397 OUTT     PI      RETN
'0242  45            0398          LR      A,5
'0243  0B            0399          LR      IS,A
'0244  1D            0400          LR      W,J
'0245  44            0401          LR      A,4
'0246  1B            0402          EI
'0247  0C            0403          PK                      ;RETURN
                     0404 ;TRMIT-THIS SUBROUTINE TRANSMITS MASTER,SLAVE1 &
                     0405 ;SLAVE2 GENERATED DATA TO PDP-11 VIA ISMI
'0248  2042          0406 TRMIT    LI      H'42'
```

```
 ADDR  OBJECT FLAG ST # SOURCE STATEMENT
'024A  2720          0407          OUT     H'20'
'024C  2621          0408 TRY      IN      H'21'
'024E  2501          0409          CI      1
'0250  84FB          0410          BZ      TRY
'0252  2A084F        0411          DCI     WCMS
'0255  2043          0412          LI      H'43'
'0257  2720          0413          OUT     H'20'
'0259  2621          0414          IN      H'21'
'025B  8D            0415          CM
'025C  8404          0416          BZ      SEND
'025E  290053'  A    0417          JMP     BUY
'0261  280400'  A    0418 SEND     PI      CLEAR
'0264  71            0419          LIS     1
'0265  2721          0420          OUT     H'21'
'0267  2082          0421          LI      H'82'
'0269  2720          0422          OUT     H'20'
'026B  2A087E        0423 TR0      DCI     MTRF
'026E  16            0424          LM
'026F  2501          0425          CI      1
'0271  8413          0426          BZ      MTR
'0273  2A08BE        0427 TR1      DCI     S1TRF
'0276  16            0428          LM
'0277  2501          0429          CI      1
'0279  8419          0430          BZ      S1TR
'027B  2A08FE        0431 TR2      DCI     S2TRF
'027E  16            0432          LM
'027F  2501          0433          CI      1
'0281  841F          0434          BZ      S2TR
'0283  9029          0435          BR      NEW
'0285  2010          0436 MTR      LI      H'10'
'0287  50            0437          LR      0,A
'0288  20B7          0438          LI      H'B7'
'028A  51            0439          LR      1,A
'028B  2A087F        0440          DCI     H'087F'
'028E  2802D3'  A    0441          PI      WRITE
'0291  90E1          0442          BR      TR1
'0293  2010          0443 S1TR     LI      H'10'
'0295  50            0444          LR      0,A
'0296  20A7          0445          LI      H'A7'
'0298  51            0446          LR      1,A
'0299  2A08BF        0447          DCI     H'08BF'
'029C  2802D3'  A    0448          PI      WRITE
'029F  90DB          0449          BR      TR2
'02A1  2010          0450 S2TR     LI      H'10'
'02A3  50            0451          LR      0,A
'02A4  2097          0452          LI      H'97'
'02A6  51            0453          LR      1,A
'02A7  2A08FF        0454          DCI     H'08FF'
'02AA  2802D3'  A    0455          PI      WRITE
'02AD  280400'  A    0456 NEW      PI      CLEAR
'02B0  2A0852        0457          DCI     TRFC
'02B3  2C            0458          XDC
'02B4  2A0852        0459          DCI     TRFC
'02B7  16            0460          LM
'02B8  1F            0461          INC
'02B9  2C            0462          XDC
'02BA  17            0463          ST
'02BB  2721          0464          OUT     H'21'
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'02BD  2083          0465          LI      H'83'
'02BF  2720          0466          OUT     H'20'
'02C1  280400'   A   0467          PI      CLEAR
'02C4  70            0468          LIS     0
'02C5  2721          0469          OUT     H'21'
'02C7  2082          0470          LI      H'82'
'02C9  2720          0471          OUT     H'20'
'02CB  2A0851        0472          DCI     TRF
'02CE  70            0473          LIS     0
'02CF  17            0474          ST
'02D0  290053'   A   0475          JMP     BUY
                     0476 ;WRITE-SUBROUTINE COPIES PM DATA INTO
                     0477 ;ISMI WHICH IS USED BY PDP-11 FOR DISPLAY
'02D3  08            0478 WRITE    LR      K,P
'02D4  70            0479 W1       LIS     0
'02D5  2720          0480          OUT     H'20'
'02D7  2721          0481          OUT     H'21'
'02D9  16            0482          LM
'02DA  2721          0483          OUT     H'21'
'02DC  41            0484          LR      A,1
'02DD  2720          0485          OUT     H'20'
'02DF  30            0486          DS      0
'02E0  94F3          0487          BNZ     W1
'02E2  0C            0488          PK                    RETURN
                     0489 ;TIMER INTERRUPT ROUTINE STARTS HERE.................
                     0490 ;TOP PRIORITY ROUTINE - PIO-1-90002 - VECT ADD H'0340'
 >0340               0491          ORG     H'0340'
'0340  08            0492          LR      K,P
'0341  54            0493          LR      4,A
'0342  1E            0494          LR      J,W
'0343  0A            0495          LR      A,IS
'0344  55            0496          LR      5,A
'0345  280435'   A   0497          PI      CALL
'0348  63            0498          LISU    3
'0349  68            0499          LISL    0
'034A  4C            0500          LR      A,S
'034B  24FF          0501          AI      H'FF'
'034D  9427          0502          BNZ     EXIIT
'034F  20FD          0503          LI      H'FD'
'0351  5D            0504          LR      I,A
'0352  4C            0505          LR      A,S
'0353  24FF          0506          AI      H'FF'
'0355  8409          0507          BZ      SETP
'0357  6A            0508 XP       LISL    2
'0358  4C            0509          LR      A,S
'0359  24FF          0510          AI      H'FF'
'035B  8410          0511          BZ      SETI
'035D  9017          0512          BR      EXIIT
'035F  69            0513 SETP     LISL    1
'0360  2A080D        0514          DCI     SI1
'0363  16            0515          LM
'0364  5C            0516          LR      S,A
'0365  2A0844        0517          DCI     PIDFLG
'0368  71            0518          LIS     1
'0369  17            0519          ST
'036A  90EC          0520          BR      XP
'036C  6A            0521 SETI     LISL    2
'036D  200A          0522          LI      H'0A'
```

```
ADDR  OBJECT FLAG ST # SOURCE STATEMENT        DATASET = DK0:HMSU  .SRC

'036F  5C            0523          LR      S,A
'0370  2A0845        0524          DCI     ISMIFG
'0373  71            0525          LIS     1
'0374  17            0526          ST
'0375  280460'   A   0527 EXIIT    PI      RETN
'0378  45            0528          LR      A,5
'0379  0B            0529          LR      IS,A
'037A  1D            0530          LR      W,J
'037B  44            0531          LR      A,4
'037C  1B            0532          EI
'037D  0C            0533          PK                      ;RETURN
                     0534 ;EXTERNAL MICROSWITCH INTERRUPT ROUTINE STARTS HERE....
                     0535 ;PIO-2-VECT ADD H'03A0'
 >03A0               0536          ORG     H'03A0'
'03A0  08            0537          LR      K,P
'03A1  54            0538          LR      4,A
'03A2  1E            0539          LR      J,W
'03A3  0A            0540          LR      A,IS
'03A4  55            0541          LR      5,A
'03A5  280435'   A   0542          PI      CALL
'03A8  1B            0543          EI
'03A9  2A0853        0544          DCI     MSCNT
'03AC  2C            0545          XDC
'03AD  2A0853        0546          DCI     MSCNT
'03B0  16            0547          LM
'03B1  24FF          0548          AI      H'FF'
'03B3  2C            0549          XDC
'03B4  17            0550          ST
'03B5  8403          0551          BZ      SET6
'03B7  9022          0552          BR      QUIT
'03B9  2A0853        0553 SET6     DCI     MSCNT
'03BC  76            0554          LIS     6
'03BD  17            0555          ST
'03BE  2A0848        0556          DCI     SLA
'03C1  16            0557          LM
'03C2  1F            0558          INC
'03C3  2A0848        0559          DCI     SLA
'03C6  17            0560          ST
'03C7  253E          0561          CI      H'3E'
'03C9  8403          0562          BZ      NVAL
'03CB  900E          0563          BR      QUIT
'03CD  2020          0564 NVAL     LI      H'20'
'03CF  2A0848        0565          DCI     SLA
'03D2  17            0566          ST
'03D3  280460'   A   0567          PI      RETN
'03D6  45            0568          LR      A,5
'03D7  0B            0569          LR      IS,A
'03D8  1D            0570          LR      W,J
'03D9  44            0571          LR      A,4
'03DA  1B            0572 QUIT     EI
'03DB  0C            0573          PK                      ;RETURN
                     0574 ;CLEAR-SUBROUTINE CLEARS I/O PORTS 20 &21
 >0400               0575          ORG     H'0400'
'0400  08            0576 CLEAR    LR      K,P
'0401  70            0577          LIS     0
'0402  2720          0578          OUT     H'20'
'0404  2721          0579          OUT     H'21'
'0406  0C            0580          PK                      ;RETURN
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
                    0581 ;ZERO SUBROUTINE CLEARS THE CONTROL LOOP BUFFERS
'0407  08           0582 ZERO    LR      K,P
'0408  6F           0583         LISL    7
'0409  70           0584         LIS     0
'040A  5E           0585 LUP1    LR      D,A
'040B  8FFE         0586         BR7     LUP1
'040D  0C           0587         PK
                    0588 ;SHUT SUBROUTINE STOPS THE CONVEYOR AND
                    0589 ;SWITCHES OFF ALL THE HEATERS OF THE FURNACE
'040E  08           0590 SHUT    LR      K,P
'040F  2050         0591         LI      H'50'
'0411  50           0592         LR      0,A
'0412  70           0593         LIS     0
'0413  B0           0594         OUTS    0
'0414  40           0595         LR      A,0
'0415  B1           0596         OUTS    1
'0416  30           0597 WAIT1   DS      0
'0417  94FE         0598         BNZ     WAIT1
'0419  2050         0599         LI      H'50'
'041B  50           0600         LR      0,A
'041C  78           0601         LIS     8
'041D  51           0602         LR      1,A
'041E  40           0603         LR      A,0
'041F  B1           0604         OUTS    1
'0420  2047         0605         LI      H'47'
'0422  52           0606         LR      2,A
'0423  70           0607 CLOSE   LIS     0
'0424  B0           0608         OUTS    0
'0425  42           0609         LR      A,2
'0426  B1           0610         OUTS    1
'0427  30           0611 WAIT    DS      0
'0428  94FE         0612         BNZ     WAIT
'042A  70           0613         LIS     0
'042B  B1           0614         OUTS    1
'042C  B0           0615         OUTS    0
'042D  2050         0616         LI      H'50'
'042F  50           0617         LR      0,A
'0430  32           0618         DS      2
'0431  31           0619         DS      1
'0432  94F0         0620         BNZ     CLOSE
'0434  0C           0621         PK                  ;RETURN
                    0622 ;CALL SUBROUTINE STORES WORKING REGISTORS IN RAM STACK
'0435  64           0623 CALL    LISU    4
'0436  68           0624         LISL    0
'0437  0E           0625         LR      Q,DC
'0438  02           0626         LR      A,QU
'0439  5D           0627         LR      I,A
'043A  03           0628         LR      A,QL
'043B  5D           0629         LR      I,A
'043C  2C           0630         XDC
'043D  0E           0631         LR      Q,DC
'043E  02           0632         LR      A,QU
'043F  5D           0633         LR      I,A
'0440  03           0634         LR      A,QL
'0441  5C           0635         LR      S,A
'0442  2A0A00       0636         DCI     SSA
'0445  43           0637         LR      A,3
'0446  8E           0638         ADC
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'0447  44          0639          LR      A,4
'0448  17          0640          ST
'0449  49          0641          LR      A,9
'044A  17          0642          ST
'044B  45          0643          LR      A,5
'044C  17          0644          ST
'044D  64          0645          LISU    4
'044E  68          0646          LISL    0
'044F  4D          0647          LR      A,I
'0450  17          0648          ST
'0451  4D          0649          LR      A,I
'0452  17          0650          ST
'0453  4D          0651          LR      A,I
'0454  17          0652          ST
'0455  4C          0653          LR      A,S
'0456  17          0654          ST
'0457  00          0655          LR      A,KU
'0458  17          0656          ST
'0459  01          0657          LR      A,KL
'045A  17          0658          ST
'045B  43          0659          LR      A,3
'045C  2409        0660          AI      H'09'
'045E  53          0661          LR      3,A
'045F  1C          0662          POP
                   0663 ;RETN SUBROUTINE RETURNS THE WORKING REGISTERS FROM RAM
STACK
'0460  1A          0664 RETN     DI
'0461  2A0A00      0665          DCI     SSA
'0464  43          0666          LR      A,3
'0465  24F7        0667          AI      H'F7'
'0467  53          0668          LR      3,A
'0468  8E          0669          ADC
'0469  16          0670          LM
'046A  54          0671          LR      4,A
'046B  16          0672          LM
'046C  59          0673          LR      9,A
'046D  16          0674          LM
'046E  55          0675          LR      5,A
'046F  64          0676          LISU    4
'0470  68          0677          LISL    0
'0471  16          0678          LM
'0472  5D          0679          LR      I,A
'0473  16          0680          LM
'0474  5D          0681          LR      I,A
'0475  16          0682          LM
'0476  5D          0683          LR      I,A
'0477  16          0684          LM
'0478  5C          0685          LR      S,A
'0479  16          0686          LM
'047A  04          0687          LR      KU,A
'047B  16          0688          LM
'047C  05          0689          LR      KL,A
'047D  4E          0690          LR      A,D
'047E  07          0691          LR      QL,A
'047F  4E          0692          LR      A,D
'0480  06          0693          LR      QU,A
'0481  0F          0694          LR      DC,Q
'0482  2C          0695          XDC
'0483  4E          0696          LR      A,D
```

```
'0484  07          0697          LR      QL,A
'0485  4E          0698          LR      A,D
'0486  06          0699          LR      QU,A
'0487  0F          0700          LR      DC,Q
'0488  1C          0701          POP
                   0702 ;PID - ROUTINE STARTS HERE.........
'0489  1A          0703 PIDR     DI
'048A  2A0849      0704          DCI     TSLA    ;DETERMINES STARTING LOAD ADDRES
S
'048D  2C          0705          XDC
'048E  2A0848      0706          DCI     SLA
'0491  16          0707          LM
'0492  2C          0708          XDC
'0493  17          0709          ST
'0494  2A0842      0710          DCI     TSZONA  ;DETERMINES STARTING ZONE ADDRES
S
'0497  2C          0711          XDC
'0498  2A0841      0712          DCI     SZONA
'049B  16          0713          LM
'049C  2C          0714          XDC
'049D  17          0715          ST
'049E  2A0840      0716          DCI     LZAC
'04A1  16          0717          LM
'04A2  2503        0718          CI      3
'04A4  8409        0719          BZ      L1
'04A6  2A0840      0720          DCI     LZAC
'04A9  16          0721          LM
'04AA  2502        0722          CI      2
'04AC  8405        0723          BZ      L2
'04AE  67          0724 L1       LISU    7
'04AF  6F          0725          LISL    7
'04B0  9007        0726          BR      CALCU
'04B2  66          0727 L2       LISU    6
'04B3  6F          0728          LISL    7
'04B4  9003        0729          BR      CALCU
'04B6  65          0730 L3       LISU    5
'04B7  6F          0731          LISL    7
'04B8  2A0849      0732 CALCU    DCI     TSLA
'04BB  16          0733          LM
'04BC  B1          0734          OUTS    1
'04BD  2805EE'  A  0735          PI      INPU
'04C0  2A0808      0736          DCI     ISP1
'04C3  16          0737          LM
'04C4  5A          0738          LR      10,A
'04C5  70          0739          LIS     0
'04C6  5B          0740          LR      11,A    ;REG 11 CONTAINS ZERO
'04C7  51          0741          LR      1,A     ;REG 1 CONTAINS ZERO
'04C8  4C          0742          LR      A,S
'04C9  50          0743          LR      0,A     ;REG 0 CONTAINS MEASURED TEMPERA
TURE
'04CA  280619'  A  0744          PI      BSUBT
'04CD  6E          0745          LISL    6
'04CE  4A          0746          LR      A,10
'04CF  5C          0747          LR      S,A     ;STORE AWAYERROR EK
'04D0  52          0748          LR      2,A     ;LOAD EK IN REG 2
'04D1  2A0809      0749          DCI     K11     ;K11 IS LOADED IN REG 1
'04D4  16          0750          LM
'04D5  51          0751          LR      1,A
'04D6  28063A'  A  0752          PI      BMPY    ;K11*EK
'04D9  46          0753          LR      A,6
'04DA  5A          0754          LR      10,A
```

```
 ADDR  OBJECT FLAG ST # SOURCE STATEMENT

'04DB  70           0755          LIS     0
'04DC  5B           0756          LR      11,A
'04DD  6D           0757          LISL    5         ;POINTER AT EK-1
'04DE  4C           0758          LR      A,S
'04DF  52           0759          LR      2,A       ;EK-1 IN REG 2
'04E0  2A080A       0760          DCI     K12       ;K12 IS LOADED IN REG1
'04E3  16           0761          LM
'04E4  51           0762          LR      1,A
'04E5  28063A'   A  0763          PI      BMPY      ;K12*EK-1
'04E8  280695'   A  0764          PI      TRAN
'04EB  280619'   A  0765          PI      BSUBT     ;K11*EK-K12*EK-1 = RESULT IN HU.
HL
'04EE  6C           0766          LISL    4         ;POINTER AT EK-2
'04EF  4C           0767          LR      A,S
'04F0  52           0768          LR      2,A       ;EK-2 IN REG 2
'04F1  2A080B       0769          DCI     K13       ;K13 IN REG 1
'04F4  16           0770          LM
'04F5  51           0771          LR      1,A
'04F6  28063A'   A  0772          PI      BMPY      ;K13*EK-2
'04F9  280695'   A  0773          PI      TRAN
'04FC  2805FD'   A  0774          PI      BADD      ;K11*EK - K12*EK-1 + K13*EK-2 =
HU.HL
'04FF  68           0775          LISL    0
'0500  4C           0776          LR      A,S
'0501  52           0777          LR      2,A       ;DEK(PK-1)U IS MOVED TO REG 2
'0502  2A080C       0778          DCI     K14       ;K14 IN REG 1
'0505  16           0779          LM
'0506  51           0780          LR      1,A
'0507  28063A'   A  0781          PI      BMPY      ;K14*DEL(PK-1)U
'050A  280695'   A  0782          PI      TRAN
'050D  2805FD'   A  0783          PI      BADD      ;K11*EK - K12*EK-1 + K13*EK-2 +
K14*DEL(PK-1)
'0510  6A           0784          LISL    2         ; POINTER AT DEL(PK)U
'0511  4A           0785          LR      A,10
'0512  5C           0786          LR      S,A
'0513  68           0787          LISL    0         ;POINTER AT DEL(PK-1)
'0514  5C           0788          LR      S,A
'0515  6B           0789          LISL    3         ;POINTER AT DEL(PK)L
'0516  4B           0790          LR      A,11
'0517  5C           0791          LR      S,A
'0518  69           0792          LISL    1         ;POINTER AT PK-1
'0519  4C           0793          LR      A,S
'051A  50           0794          LR      0,A
'051B  70           0795          LIS     0
'051C  51           0796          LR      1,A
'051D  2805FD'   A  0797          PI      BADD      ;PK=DEL(PK) + PK-1 = HU.HL
'0520  2805BB'   A  0798          PI      OUTPU     ;POWER IS OUTPUT TO HEATERS
'0523  6D           0799          LISL    5
'0524  4E           0800          LR      A,D
'0525  5C           0801          LR      S,A       ;UPDATE EK-1 TO EK-2
'0526  6E           0802          LISL    6
'0527  4E           0803          LR      A,D
'0528  5C           0804          LR      S,A       ;UPDATE EK TO EK-1
'0529  0A           0805          LR      A,IS
'052A  18           0806          COM
'052B  2138         0807          NI      H'38'
'052D  843F         0808          BZ      CL2
'052F  0A           0809          LR      A,IS
'0530  18           0810          COM
'0531  2130         0811          NI      H'30'
'0533  8445         0812          BZ      CL3
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'0535  2A087A       0813          DCI     RECO3
'0538  280599'   A  0814          PI      RECORD
'053B  2A0844       0815          DCI     PIDFLG
'053E  70           0816          LIS     0
'053F  17           0817          ST
'0540  2A0854       0818          DCI     SNO
'0543  16           0819          LM
'0544  1F           0820          INC
'0545  2A0854       0821          DCI     SNO
'0548  17           0822          ST
'0549  2A0804       0823          DCI     NS
'054C  8D           0824          CM
'054D  8404         0825          BZ      TSTRUN
'054F  290053'   A  0826          JMP     BUY
'0552  2A0856       0827 TSTRUN   DCI     TRUNNO
'0555  16           0828          LM
'0556  2A0807       0829          DCI     RUNNO
'0559  8D           0830          CM
'055A  840D         0831          BZ      ENDRUN
'055C  2A0856       0832          DCI     TRUNNO
'055F  16           0833          LM
'0560  1F           0834          INC
'0561  2A0856       0835          DCI     TRUNNO
'0564  17           0836          ST
'0565  290053'   A  0837          JMP     BUY
'0568  28069B'   A  0838 ENDRUN   PI      STOP
'056B  90FC         0839          BR      ENDRUN
'056D  2A0870       0840 CL2      DCI     RECO1
'0570  280599'   A  0841          PI      RECORD
'0573  280585'   A  0842          PI      MODLZA
'0576  2904B2'   A  0843          JMP     L2
'0579  2A0875       0844 CL3      DCI     RECO2
'057C  280599'   A  0845          PI      RECORD
'057F  280585'   A  0846          PI      MODLZA
'0582  2904B6'   A  0847          JMP     L3
                    0848 ;MODLZA SUBROUTINE MODIFIES LOAD & ZONE ADDRESS FOR THE
                    0849 ;NEXT CONTROL LOOP OF THE PID ALGORITHM
'0585  08           0850 MODLZA   LR      K,P
'0586  2A0849       0851          DCI     TSLA
'0589  16           0852          LM
'058A  1F           0853          INC
'058B  2A0849       0854          DCI     TSLA
'058E  17           0855          ST
'058F  2A0842       0856          DCI     TSZONA
'0592  16           0857          LM
'0593  1F           0858          INC
'0594  2A0842       0859          DCI     TSZONA
'0597  17           0860          ST
'0598  0C           0861          PK              ;RETURN
                    0862 ;RECORD SUBROUTINE RECORDS SAMPLE NUMBER, ZONE ADDRESS,
                    0863 ;LOAD TEMPERATURE, POWER TO THE ZONE & LOAD ADDRESS
'0599  08           0864 RECORD   LR      K,P
'059A  2C           0865          XDC
'059B  2A0854       0866          DCI     SNO
'059E  16           0867          LM
'059F  2C           0868          XDC
'05A0  17           0869          ST
'05A1  2C           0870          XDC
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'05A2  2A0842        0871          DCI     TSZONA
'05A5  16            0872          LM
'05A6  2C            0873          XDC
'05A7  17            0874          ST
'05A8  6F            0875          LISL    7
'05A9  4C            0876          LR      A,S
'05AA  17            0877          ST
'05AB  2C            0878          XDC
'05AC  2A0855        0879          DCI     ANSWER
'05AF  16            0880          LM
'05B0  2C            0881          XDC
'05B1  17            0882          ST
'05B2  2C            0883          XDC
'05B3  2A0849        0884          DCI     TSLA
'05B6  16            0885          LM
'05B7  2C            0886          XDC
'05B8  17            0887          ST
'05B9  0C            0888          PK              ;RETURN
'05BA  2B            0889          NOP
                     0890 ;OUTPU SUBROUTINE OUTPUTS A BYTE OF ABSOLUTE POWER
                     0891 ;VIA PORT 0 FOLLOWED BY 150 MICRO SECONDS DELAY AND
                     0892 ;THEN THE ZONE ADDRESS IS SET UP ON PORT 1
'05BB  08            0893 OUTPU    LR      K,P
'05BC  2050          0894          LI      H'50'
'05BE  50            0895          LR      0,A
'05BF  70            0896          LIS     0
'05C0  B0            0897          OUTS    0
'05C1  4A            0898          LR      A,10
'05C2  2380          0899          XI      H'80'
'05C4  8103          0900          BP      NIL
'05C6  9011          0901          BR      CONT
'05C8  70            0902 NIL      LIS     0
'05C9  2A0855        0903          DCI     ANSWER
'05CC  17            0904          ST
'05CD  B0            0905          OUTS    0
'05CE  2A0842        0906          DCI     TSZONA
'05D1  16            0907          LM
'05D2  B1            0908          OUTS    1
'05D3  69            0909          LISL    1
'05D4  4A            0910          LR      A,10
'05D5  5C            0911          LR      S,A
'05D6  9010          0912          BR      TST
'05D8  69            0913 CONT     LISL    1
'05D9  4A            0914          LR      A,10
'05DA  5C            0915          LR      S,A
'05DB  13            0916          SL      1
'05DC  1F            0917          INC
'05DD  2A0855        0918          DCI     ANSWER
'05E0  17            0919          ST
'05E1  B0            0920          OUTS    0
'05E2  2A0842        0921          DCI     TSZONA
'05E5  16            0922          LM
'05E6  B1            0923          OUTS    1
'05E7  30            0924 TST      DS      0
'05E8  94FE          0925          BNZ     TST
'05EA  70            0926          LIS     0
'05EB  B0            0927          OUTS    0
'05EC  B1            0928          OUTS    1
```

```
ADDR  OBJECT FLAG ST # SOURCE STATEMENT       DATASET = DK0:HMSU  .SRC

'05ED  0C          0929          PK              ;RETURN
                   0930 ;INPU SUBROUTINE INPUTS A BYTE OF DATA VIA PORT 0
                   0931 ;POLLING ON STATUS BIT 1 OF PORT 8
'05EE  08          0932 INPU     LR      K,P
'05EF  A8          0933 LOOP2    INS     8
'05F0  2102        0934          NI      H'02'           ;MASK BIT 1
'05F2  84FC        0935          BZ      LOOP2
'05F4  71          0936          LIS     1               ;OPEN GATE
'05F5  B8          0937          OUTS    8               ;VIA PORT 8
'05F6  A0          0938          INS     0               ;READ DATA VIA PORT 0
'05F7  5C          0939          LR      S,A             ;AND STORE IN 0'77'
'05F8  70          0940          LIS     0
'05F9  B1          0941          OUTS    1
'05FA  B8          0942          OUTS    8               ;CLOSE GATE
'05FB  B0          0943          OUTS    0
'05FC  0C          0944          PK                      ;RETURN
                   0945 ;BADD SUBROUTINE-16 BIT BINARY ADDITION IN
                   0946 ;REG(HU).REG(HL)+REG(0).REG(1)=HU.HL
'05FD  08          0947 BADD     LR      K,P
'05FE  10          0948          LR      DC,H    ;MOVE AUGEND TO DCO
'05FF  41          0949          LR      A,1     ;LOAD LOWER ORDER BYTE OF ADDEND
'0600  8E          0950          ADC             ;ADD TO AUGEND
'0601  F1          0951          NS      1
'0602  11          0952          LR      H,DC    ;RESTORE PARTIAL SUM TO H
'0603  4A          0953          LR      A,10    ;LOAD HIGH ORDER BYTE OF PARTIAL
 ANS
'0604  8102        0954          BP      BAO
'0606  1F          0955          INC
'0607  C0          0956 BAO      AS      0       ;ADD HIGH ORDER BYTE OF ADDEND
'0608  5A          0957          LR      10,A    ;RESTORE TO GIVE COMPLETE ANS IN
 H
'0609  980E        0958          BNO     FINISH
'060B  40          0959          LR      A,0
'060C  2380        0960          XI      H'80'
'060E  8106        0961          BP      NEGT
'0610  207F        0962          LI      H'7F'
'0612  5A          0963          LR      10,A
'0613  9004        0964          BR      FINISH
'0615  2080        0965 NEGT     LI      H'80'
'0617  5A          0966          LR      10,A
'0618  0C          0967 FINISH   PK              ;RETURN
                   0968 ;BSUBT SUBROUTINE-16 BIT BINARY SUBTRACTION IN
                   0969 ;REG(HU).REG(HL)-REG(0).REG(1)=HU.HL
'0619  08          0970 BSUBT    LR      K,P
'061A  4A          0971          LR      A,10
'061B  58          0972          LR      8,A
'061C  10          0973          LR      DC,H    ;MOVE SUBTRAHEND TO DCO
'061D  41          0974          LR      A,1     ;LOAD LOW ORDER BYTE OF MINUEND
'061E  18          0975          COM             ;COMPLEMENT IT
'061F  1F          0976          INC             ;INCREMENT IT
'0620  8E          0977          ADC             ;ADD TO SUBTRAHEND
'0621  8102        0978          BP      BSO     ;IF BIT 7 = 1...
'0623  30          0979          DS      0       ;COMPENSATE BY DECREMENTING HIGH
 ORDER BYTE
'0624  11          0980 BSO      LR      H,DC    ;RESTORE PARTIAL SUM TO H
'0625  40          0981          LR      A,0     ;LOAD HIGH ORDER BYTE OF MINUEND
'0626  18          0982          COM             ;COMPLEMENT
'0627  1F          0983          INC             ;INCREMENT
'0628  CA          0984          AS      10
'0629  5A          0985          LR      10,A    ;STORE ANSWER IN REG(10)
'062A  980E        0986          BNO     DONE
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'062C  48          0987          LR      A,8
'062D  2380        0988          XI      H'80'
'062F  8106        0989          BP      NPOS
'0631  207F        0990          LI      H'7F'
'0633  5A          0991          LR      10,A
'0634  9004        0992          BR      DONE
'0636  2080        0993 NPOS     LI      H'80'
'0638  5A          0994          LR      10,A
'0639  0C          0995 DONE     PK                ;RETURN
                   0996 ;BMPY SUBROUTINE - 8 BIT(REG1:MULTIPLIER)*8 BIT(REG2:MUL
TIPLICAND)
                   0997 ;=16 BIT PRODUCT IN REG7(INT).REG6(FRA)
'063A  08          0998 BMPY     LR      K,P
'063B  41          0999          LR      A,1
'063C  E2          1000          XS      2
'063D  58          1001          LR      8,A       ;SET MARK IN REG 8 FOR -VE ANSWE
R
'063E  41          1002          LR      A,1
'063F  2380        1003          XI      H'80'     ;TEST FOR -VE NUMBER IN REG 1
'0641  8108        1004          BP      NEG1
'0643  42          1005 NEXT     LR      A,2       ;TEST FOR -VE NUMBER IN REG 2
'0644  2380        1006          XI      H'80'
'0646  8109        1007          BP      NEG2
'0648  900B        1008          BR      MULT      ;BRANCH TO MULTIPLICATION
'064A  41          1009 NEG1     LR      A,1
'064B  18          1010          COM
'064C  1F          1011          INC
'064D  51          1012          LR      1,A       ;REG1 NUMBER IS +VE NOW
'064E  90F4        1013          BR      NEXT
'0650  42          1014 NEG2     LR      A,2
'0651  18          1015          COM
'0652  1F          1016          INC
'0653  52          1017          LR      2,A       ;REG2 NUMBER IS +VE NOW
'0654  78          1018 MULT     LIS     8         ;INITIALISE COUNTER TO 8
'0655  55          1019          LR      5,A
'0656  70          1020          LIS     0         ;INITIALISE PARTIAL PRODUCT
'0657  56          1021          LR      6,A
'0658  57          1022          LR      7,A
'0659  70          1023 BMP1     LIS     0
'065A  C1          1024          AS      1         ;IS SIGN BIT OF MULTIPLIER SET ?
'065B  8107        1025          BP      BMP2      ;NO! THEN SHIFT PARTIAL PRODUCT
'065D  42          1026          LR      A,2
'065E  C6          1027          AS      6         ;YES! ADD MULTIPLICAND TO
'065F  56          1028          LR      6,A       ;PARTIAL PRODUCT
'0660  47          1029          LR      A,7
'0661  19          1030          LNK
'0662  57          1031          LR      7,A
'0663  41          1032 BMP2     LR      A,1
'0664  13          1033          SL      1         ;SHIFT THE MULTIPLIER LEFT 1
'0665  51          1034          LR      1,A
'0666  35          1035          DS      5         ;DECREMENT THE COUNTER
'0667  840C        1036          BZ      BMP3      ;EXIT IF DONE
'0669  46          1037          LR      A,6
'066A  C6          1038          AS      6
'066B  56          1039          LR      6,A
'066C  1E          1040          LR      J,W       ;SAVE STATUS FOR CARRY
'066D  47          1041          LR      A,7
'066E  13          1042          SL      1
'066F  1D          1043          LR      W,J
'0670  19          1044          LNK
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'0671  57          1045          LR      7,A
'0672  90E6        1046          BR      BMP1    ;GO BACK FOR NEXT
'0674  47          1047 BMP3     LR      A,7
'0675  2300        1048          XI      H'00'   ;TEST FOR ALL ZEROS IN REG 7
'0677  8406        1049          BZ      TEST
'0679  207F        1050          LI      H'7F'
'067B  56          1051          LR      6,A
'067C  900C        1052          BR      ANS
'067E  46          1053 TEST     LR      A,6
'067F  2380        1054          XI      H'80'
'0681  8103        1055          BP      TWOS
'0683  9005        1056          BR      ANS
'0685  46          1057 TWOS     LR      A,6
'0686  18          1058          COM
'0687  1F          1059          INC
'0688  56          1060          LR      6,A
'0689  48          1061 ANS      LR      A,8
'068A  2380        1062          XI      H'80'   ;CHECK FOR -VE ANSWER
'068C  8103        1063          BP      NEGA
'068E  9005        1064          BR      EXIT
'0690  46          1065 NEGA     LR      A,6
'0691  18          1066          COM
'0692  1F          1067          INC
'0693  56          1068          LR      6,A
'0694  0C          1069 EXIT     PK              ;RETURN
                   1070 ;TRAN SUBROUTINE TRANSFERS THE CONTENTS OF REG 6 TO 0 &
REG1=0
'0695  08          1071 TRAN     LR      K,P
'0696  46          1072          LR      A,6
'0697  50          1073          LR      0,A
'0698  70          1074          LIS     0
'0699  51          1075          LR      1,A
'069A  0C          1076          PK              ;RETURN
                   1077 ;STOP SUBROUTINE STOPS THE CONVEYOR AND CLOSES ALL THE H
EATING
                   1078 ;ZONES ASSOCIATED WITH CONTROLLER NO-1
'069B  08          1079 STOP     LR      K,P
'069C  2050        1080          LI      H'50'
'069E  50          1081          LR      0,A
'069F  70          1082          LIS     0
'06A0  B0          1083          OUTS    0
'06A1  40          1084          LR      A,0
'06A2  B1          1085          OUTS    1
'06A3  30          1086 TST2     DS      0
'06A4  94FE        1087          BNZ     TST2
'06A6  2050        1088          LI      H'50'
'06A8  50          1089          LR      0,A
'06A9  2A0840      1090          DCI     LZAC
'06AC  16          1091          LM
'06AD  51          1092          LR      1,A
'06AE  2A0841      1093          DCI     SZONA
'06B1  16          1094          LM
'06B2  52          1095          LR      2,A
'06B3  70          1096 TURN     LIS     0
'06B4  B0          1097          OUTS    0
'06B5  42          1098          LR      A,2
'06B6  B1          1099          OUTS    1
'06B7  30          1100 TST1     DS      0
'06B8  94FE        1101          BNZ     TST1
'06BA  2050        1102          LI      H'50'
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
'06BC  50          1103          LR      0,A
'06BD  32          1104          DS      2
'06BE  31          1105          DS      1
'06BF  94F3        1106          BNZ     TURN
'06C1  0C          1107          PK                ;RETURN
'>06C2             1108 END
```


ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

AGA	0023	ANS	0689	ANSWER	0855	BADD	05FD
BAO	0607	BMP1	0659	BMP2	0663	BMP3	0674
BMPY	063A	BSO	0624	BSUBT	0619	BUY	0053
BUY1	0134	BUY2	017E	BUY3	017C	BUYOT	0189
BUYOUT	0184	C1	0800	C2	0801	C3	0802
CALCU	04B8	CALL	0435	CL2	056D	CL3	0579
CLEAR	0400	CLOSE	0423	CM	0806	CMAR	01F0
CMFLG	0847	CONT	05D8	COPY	0191	DONE	0639
END	06C2	ENDRUN	0568	EXIIT	0375	EXIT	0694
FINISH	0618	FULL	0082	INPU	05EE	ISMI	0078
ISMIFG	0845	ISP1	0808	ISP2	0810	ISP3	0818
K11	0809	K12	080A	K13	080B	K14	080C
K21	0811	K22	0812	K23	0813	K24	0814
K31	0819	K32	081A	K33	081B	K34	081C
L1	04AE	L2	04B2	L3	04B6	LA	0803
LA0	016A	LA1	016E	LA2	0172	LA3	0176
LA4	017A	LAN	0130	LAO	012A	LOOP2	05EF
LUP1	040A	LZAC	0840	MODLZA	0585	MOTA	0050
MS	0805	MSCNT	0853	MSFLG	0846	MTR	0285
MTRF	087E	MULT	0654	NEG1	064A	NEG2	0650
NEGA	0690	NEGT	0615	NEW	02AD	NEXT	0643
NIL	05C8	NPOS	0636	NS	0804	NVAL	03CD
ON	00D9	OUTPU	05BB	OUTT	023F	PID	006E
PIDFLG	0844	PIDR	0489	QUIT	03DA	RCPS	083C
RECO1	0870	RECO2	0875	RECO3	087A	RECORD	0599
REP	0238	REPT	01AA	RETN	0460	RFCMS	0843
RFMS	084C	RFPS	083D	RUNNO	0807	S1TR	0293
S1TRF	08BE	S2TR	02A1	S2TRF	08FE	SAME	009D
SEND	0261	SET1	00AA	SET2	00B9	SET3	00C9
SET6	03B9	SETI	036C	SETP	035F	SHUT	040E
SI1	080D	SI2	0815	SI3	081D	SLA	0848
SLV0	0223	SLV1	021C	SLV2	0213	SNO	0854
SPA0	087F	SPA1	08BF	SPA2	08FF	SSA	0A00
STAY	007C	STOP	069B	SZONA	0841	TEST	067E
TEST1	00A0	TEST2	00CE	TLZAC	084A	TR0	026B
TR1	0273	TR2	027B	TRAN	0695	TRF	0851
TRFC	0852	TRMIT	0248	TRMITT	0073	TRUNNO	0856
TRY	024C	TSLA	0849	TST	05E7	TST1	06B7
TST2	06A3	TSTRUN	0552	TSZONA	0842	TURN	06B3
TWOS	0685	UUU	00F4	VVV	013D	W1	02D4
WAIT	0427	WAIT1	0416	WCMS	084F	WFCPS	083E
WFMS	0850	WFPS	083F	WRITE	02D3	WWW	010C
X1	00AF	X2	00BE	XP	0357	ZERO	0407
ZON0	0040	ZON1	0041	ZON2	0042	ZON3	0043
ZON4	0044	ZON5	0045	ZON6	0046	ZON7	0047

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

CROSS REFERENCE LISTING

SYMBOL	VALUE	TYPE	STMT	STATEMENT REFERENCES
AGA	0023		0094	0096
ANS	0689		1061	1056 1052
ANSWER	0855		0066	0918 0903 0879
BADD	05FD		0947	0797 0783 0774
BAD	0607		0956	0954
BMP1	0659		1023	1046
BMP2	0663		1032	1025
BMP3	0674		1047	1036
BMPY	063A		0998	0781 0772 0763 0752
BSO	0624		0980	0978
BSUBT	0619		0970	0765 0744
BUY	0053		0128	0837 0826 0475 0417 0298 0250 0145 0143 0141
BUY1	0134		0247	0243 0239
BUY2	017E		0289	0277
BUY3	017C		0287	0285 0283 0281 0279
BUYOT	0189		0295	0166
BUYOUT	0184		0292	0221 0208 0187
C1	0800		0002	0191 0167 0090
C2	0801		0003	0175
C3	0802		0004	0183
CALCU	04B8		0732	0729 0726
CALL	0435		0623	0542 0497 0353
CL2	056D		0840	0808
CL3	0579		0844	0812
CLEAR	0400		0576	0467 0456 0418 0153
CLOSE	0423		0607	0620
CM	0806		0008	0204 0200 0196
CMAR	01F0		0348	0195
CMFLG	0847		0048	
CONT	05D8		0913	0901
COPY	0191		0301	0159
DONE	0639		0995	0992 0986
END	06C2		1108	
ENDRUN	0568		0838	0839 0831
EXIIT	0375		0527	0512 0502
EXIT	0694		1069	1064
FINISH	0618		0967	0964 0958
FULL	0082		0152	
INPU	05EE		0932	0735
ISMI	0078		0147	0136
ISMIFG	0845		0046	0524 0295 0133 0124
ISP1	0808		0010	0736
ISP2	0810		0016	
ISP3	0818		0022	
K11	0809		0011	0749
K12	080A		0012	0760
K13	080B		0013	0769
K14	080C		0014	0778
K21	0811		0017	
K22	0812		0018	
K23	0813		0019	
K24	0814		0020	
K31	0819		0023	
K32	081A		0024	
K33	081B		0025	
K34	081C		0026	

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DK0:HMSU .SRC

```
L1      04AE            0724    0719
L2      04B2            0727    0843 0723
L3      04B6            0730    0847
LA      0803            0005    0256 0227 0214
LA0     016A            0278    0261
LA1     016E            0280    0264
LA2     0172            0282    0267
LA3     0176            0284    0270
LA4     017A            0286    0273
LAN     0130            0244    0235
LAO     012A            0240    0232
LOOP2   05EF            0933    0935
LUP1    040A            0585    0586
LZAC    0840            0041    1090 0720 0716 0251 0222 0209
MODLZA  0585            0850    0846 0842
MOTA    0050            0040
MS      0805            0007
MSCNT   0853            0064    0553 0546 0544
MSFLG   0846            0047
MTR     0285            0436    0426
MTRF    087E            0058    0423
MULT    0654            1018    1008
NEG1    064A            1009    1004
NEG2    0650            1014    1007
NEGA    0690            1065    1063
NEGT    0615            0965    0961
NEW     02AD            0456    0435
NEXT    0643            1005    1013
NIL     05C8            0902    0900
NPOS    0636            0993    0989
NS      0804            0006    0823
NVAL    03CD            0564    0562
ON      00D9            0196    0194
OUTPU   05BB            0893    0798
OUTT    023F            0397    0373 0367
PID     006E            0142    0132
PIDFLG  0844            0045    0815 0517 0292 0129 0120
PIDR    0489            0703    0142
QUIT    03DA            0572    0563 0552
RCPS    083C            0028
RECO1   0870            0068    0840
RECO2   0875            0069    0844
RECO3   087A            0070    0813
RECORD  0599            0864    0845 0841 0814
REP     0238            0391    0396
REPT    01AA            0316    0328
RETN    0460            0664    0567 0527 0397
RFCMS   0843            0044
RFMS    084C            0055
RFPS    083D            0029
RUNNO   0807            0009    0829
S1TR    0293            0443    0430
S1TRF   08BE            0059    0427
S2TR    02A1            0450    0434
S2TRF   08FE            0060    0431
SAME    009D            0166    0158
SEND    0261            0418    0416
SET1    00AA            0172    0170
```

```
SET2    00B9            0180    0178
SET3    00C9            0188    0186
SET6    03B9            0553    0551
SETI    036C            0521    0511
SETP    035F            0513    0507
SHUT    040E            0590    0089
SI1     080D            0015    0514 0162
SI2     0815            0021
SI3     081D            0027
SLA     0848            0049    0706 0565 0559 0556 0254 0225 0212
SLV0    0223            0379    0366
SLV1    021C            0374    0362
SLV2    0213            0368    0358
SNO     0854            0065    0866 0821 0818 0122
SPA0    087F            0052    0379 0363 0172
SPA1    08BF            0053    0374 0359 0180
SPA2    08FF            0054    0368 0355 0188
SSA     0A00            0063    0665 0636
STAY    007C            0149    0151
STOP    069B            1079    0838
SZONA   0841            0042    1093 0712 0289 0247 0218
TEST    067E            1053    1049
TEST1   00A0            0167    0165
TEST2   00CE            0191
TLZAC   084A            0051
TR0     026B            0423
TR1     0273            0427    0442
TR2     027B            0431    0449
TRAN    0695            1071    0782 0773 0764
TRF     0851            0061    0472 0383 0137
TRFC    0852            0062    0459 0457
TRMIT   0248            0406    0144
TRMITT  0073            0144    0140
TRUNNO  0856            0067    0835 0832 0827
TRY     024C            0408    0410
TSLA    0849            0050    0884 0854 0851 0732 0704
TST     05E7            0924    0925 0912
TST1    06B7            1100    1101
TST2    06A3            1086    1087
TSTRUN  0552            0827    0825
TSZONA  0842            0043    0921 0906 0871 0859 0856 0710
TURN    06B3            1096    1106
TWOS    0685            1057    1055
UUU     00F4            0209    0199
VVV     013D            0251    0203
W1      02D4            0479    0487
WAIT    0427            0611    0612
WAIT1   0416            0597    0598
WCMS    084F            0056    0411
WFCPS   083E            0030    0329 0152
WFMS    0850            0057
WFPS    083F            0031
WRITE   02D3            0478    0455 0448 0441
WWW     010C            0222    0207
X1      00AF            0175    0171
X2      00BE            0183    0179
XP      0357            0508    0520
ZERO    0407            0582    0088 0086 0084
```

ADDR OBJECT FLAG ST # SOURCE STATEMENT DATASET = DKO:HMSU .SRC

```
ZON0    0040            0032    0219
ZON1    0041            0033
ZON2    0042            0034    0248
ZON3    0043            0035
ZON4    0044            0036
ZON5    0045            0037    0290
ZON6    0046            0038
ZON7    0047            0039

ERRORS=0000
```

APPENDIX D - THE DCHMSU PROGRAM LISTING

The following DCHMSU program listing for the PDP-11/10 minicomputer system:

```
      PROGRAM DCHMSU
C
C THIS IS A COMMUNICATION PROGRAM BETWEEN A OPERATOR AND THE
C SUPREMAL LEVEL PDP-11/10 MINICOMPUTER. THE INFIMAL  LEVEL
C CONSISTS OF THREE CONTROLLERS USED FOR CONTROLLING THE
C TRAVELLING LOAD FURNACE IN THE DEPT OF SYSTEMS SCIENCE.
C THE THREE CONTROLLERS ARE IMPLEMENTED IN THE HIERARCHICAL
C MICROPROCESSOR SYSTEM UNIT (HMSU) USING THREE F8 MICROPROCESSORS.
C
      DIMENSION XK(4,3),A(12),AR(12),SF(8),NAA(12),IA(12,8),
      1 IXK(4,3),SG(3),ISA(64),ISD(64)
      INTEGER RH,IRUN
      DIMENSION S1(4),S2(15),S3(7),S4(2),S11(4),S22(15),IS3(7),IC(3)
      COMMON /BLK1/C1,C2,C3,CM
      COMMON /BLK2/NO1C,NO2C,NO3C
      COMMON /BLK3/GK1,GK2,GK3,T1,T2,T3,TI1,TI2,TI3,TD1,TD2,TD3,
      1 TF1,TF2,TF3
      COMMON /BLK4/LA,NS,MS,RH,ISP1,ISP2,ISP3
      COMMON /BLK5/XK,IXK
      COMMON /BLK6/KCM,IRCPS,IRFPS,IWCPS,IWFPS
      COMMON /BLK7/ISA,ISD,IRUN
      DATA ON/'ON'/,OFF/'OFF'/,DIS/'DIS'/,SET/'SET'/,CON/'CON'/
      DATA CHA/'CHA'/,PAR/'PAR'/
      DATA PX/0.004/,U/'U'/,V/'V'/,W/'W'/
C THE DEFAULT VALUES OF THE CONTROLLER PARAMETERS ARE :
      CM=U
      LA=0
      NS=100
      MS=0
      RH=0
      C1=ON
      C2=ON
      C3=ON
      ISP1=50.0
      ISP2=50.0
      ISP3=50.0
      GK1=0.05
      GK2=0.05
      GK3=0.05
      T1=30.0
      T2=30.0
      T3=30.0
      TI1=90.0
      TI2=90.0
      TI3=90.0
      TD1=30.0
      TD2=30.0
      TD3=30.0
      TF1=30.0
      TF2=30.0
      TF3=30.0
      NO1C=255
      NO2C=255
      NO3C=255
      GOTO 700
50    CALL PRINT('SELECT THE CONTROLLER NO - (I.E. 1 OR 2 OR 3)')
      READ(5,1001)ICNO
1001  FORMAT(I2)
      IF(ICNO.EQ.1)GOTO 60
      IF(ICNO.EQ.2)GOTO 2
      IF(ICNO.EQ.3)GOTO 5
```

```
51      GOTO 50
60      CALL Q('DO YOU WANT CONTROLLER-1 TO BE ON ?',INO1C)
        IF(INO1C)1,2,3
1       NO1C=0
        C1=OFF
        GOTO 51
3       NO1C=255
        C1=ON
        CALL PRINT(' SPECIFY CONTROLLER-1 SET POINT')
        READ(5,1111)ISP1
        CALL PRINT(' SPECIFY CONTROLLER-1 CONSTANTS')
        CALL CONST(GK1,T1,TI1,TD1,TF1)
2       CALL Q('DO YOU WANT CONTROLLER-2 TO BE ON ?',INO2C)
        IF(INO2C)4,5,6
4       NO2C=0
        C2=OFF
        GOTO 51
6       NO2C=255
        C2=ON
        CALL PRINT(' SPECIFY CONTROLLER-2 SET POINT')
        READ(5,1111)ISP2
        CALL PRINT(' SPECIFY CONTROLLER-2 CONSTANTS')
        CALL CONST(GK2,T2,TI2,TD2,TF2)
5       CALL Q('DO YOU WANT CONTROLLER-3 TO BE ON ?',INO3C)
        IF(INO3C)7,8,9
7       NO3C=0
        C3=OFF
        GOTO 51
9       NO3C=255
        C3=ON
        CALL PRINT(' SPECIFY CONTROLLER-3 SET POINT')
        READ(5,1111)ISP3
1111    FORMAT(I4)
        CALL PRINT(' SPECIFY CONTROLLER-3 CONSTANTS')
        CALL CONST(GK3,T3,TI3,TD3,TF3)
8       CALL PRINT(' WHAT IS THE CONTROL MODE FOR CONTROLLRRS?')
        READ(5,500)CM
500     FORMAT(A1)
        CALL PRINT(' WHAT IS THE INITIAL LOAD ADDRESS IN ZONE-0?')
        READ(5,501)LA
501     FORMAT(I2)
        CALL PRINT(' SPECIFY THE NUMBER OF SAMPLES')
        READ(5,502)NS
502     FORMAT(I4)
        CALL PRINT(' SPECIFY THE MOTOR SPEED')
        READ(5,503)MS
503     FORMAT(I4)
        CALL PRINT(' SPECIFY THE RUN TIME FOR THE FURNACE')
        CALL PRINT(' IN HOURS - (INTEGER VALUE)')
        READ(5,503)RH
        CALL PRINT(' PRESS BREAK KEY NOW')
25      IF(IBREAK().NE.0)GOTO 700
        GOTO 25
700     CALL PRINT(' ***************************************************')
        CALL PRINT(' THE FOLLOWING INPUT COMMANDS ARE AVAILABLE')
        CALL PRINT(' [1] "DIS"- PRINTS OUT OPERATOR SET INFORMATION')
        CALL PRINT(' [2] "SET"- OPERATOR CAN SET THE PARAMETERS')
        CALL PRINT(' [3] "CON"- PROGRAM CONTINUES')
        CALL PRINT(' [4] "CHA"- OPERATOR CAN CHANGE THE PARAMETERS')
        CALL PRINT(' [5] "PAR"- PRINTS OUT THE LIST OF PARAMETERS')
        CALL PRINT(' - PRESS RETURN KEY AFTER ANY INPUT COMMAND')
        CALL PRINT(' ***************************************************')
599     READ(5,600)COMD
600     FORMAT(A4)
        IF(COMD.EQ.DIS)GOTO 701
        IF(COMD.EQ.SET)GOTO 702
```

```
      IF(COMD.EQ.CON)GOTO 703
      IF(COMD.EQ.CHA)GOTO 704
      IF(COMD.EQ.PAR)GOTO 705
      CALL PRINT(' *ERROR*-INVALID COMMAND, TYPE COMMAND AGAIN')
      GOTO 599
705   CALL LIST
      GOTO 25
701   CALL OPINFO
      GOTO 25
702   GOTO 50
704   CALL CHANGE
      CONTINUE
      CALL Q(' DO YOU WANT TO CHANGE ANY MORE PARAMETERS?',IR)
      IF(IR)311,311,704
311   CALL PRINT(' PRESS BREAK KEY')
      GOTO 25
703   CONTINUE
      CALL CALCU(GK1,T1,TI1,TD1,TF1,XK(1,1),XK(2,1),XK(3,1),XK(4,1))
      CALL CALCU(GK2,T2,TI2,TD2,TF2,XK(1,2),XK(2,2),XK(3,2),XK(4,2))
      CALL CALCU(GK3,T3,TI3,TD3,TF3,XK(1,3),XK(2,3),XK(3,3),XK(4,3))
      DO 1010 I=1,3
      DO 1011 J=1,4
      IF(XK(J,I).GT.1.0.OR.XK(J,I).EQ.1.0)GOTO 1012
1011  CONTINUE
1010  CONTINUE
      GOTO 1015
1012  CALL PRINT('...INVALID CONTROLLER CONSTANTS.................')
      WRITE(5,1013)I
1013  FORMAT(1H ,'...SELECT NEW PARAMETERS FOR CONTROLLER NO',I1,'..'/)
      GOTO 1016
1015  WRITE(6,39)
39    FORMAT(1X/1H ,'  K1',8X,'  K2',8X,'  K3',8X,'  K4')
      DO 1005 I=1,3
      WRITE(6,40)XK(1,I),XK(2,I),XK(3,I),XK(4,I)
40    FORMAT(1X/1H ,F7.3,4X,F7.3,4X,F7.3,4X,F7.3)
1005  CONTINUE
      DO 1006 I=1,3
      Z=XK(1,I)+XK(2,I)+XK(3,I)
      Y=1.0-XK(4,I)
      SG(I)=Z/Y
      IF(SG(I))82,82,83
83    IF(SG(I).LT.PX)GOTO 82
      WRITE(5,84)I,SG(I)
      GOTO 1006
82    CALL PRINT('...UNSTABLE STEADY STATE GAIN.....................')
      CALL PRINT('...SELECT NEW VALUES FOR K,T,TI...................')
      WRITE(5,85)I
85    FORMAT(1H ,'...OF CONTROLLER NO ',I2)
1016  CALL PRINT('PRESS BREAK KEY AND USE *CHA* COMMAND')
      GOTO 25
84    FORMAT(1H ,' STEADY STATE GAIN SG',I2,'=',1X,F7.3)
1006  CONTINUE
      DO 1007 I=1,3
      WRITE(6,84)I,SG(I)
1007  CONTINUE
      IRUN=0
      BIG=AMAX1(T1,T2,T3)
      IF(RH.LE.0)GOTO 31
      TIM=FLOAT(RH)*3600.0
      GOTO 2001
31    IF(NS.EQ.0)GOTO 999
      TIM=FLOAT(NS)*BIG
2001  SAM=TIM/BIG
      IF(SAM.GE.240.0)GOTO 100
      NS=INT(SAM)
      IRUN=0
```

```
      GOTO 331
100   NS=240
      RUNNO=SAM/240.0
      IRUN=INT(RUNNO)
331   RNS=FLOAT(NS)
      TM=(RNS*BIG)/3600.0
      REP=FLOAT(IRUN)
      ATIM=TM*(1.0+REP)
      IHRS=INT(ATIM)
      FATIM=ATIM-FLOAT(IHRS)
      AMIN=FATIM*60.0
      IMIN=INT(AMIN)
      WRITE(6,101)NS
101   FORMAT(1H ,'NUMBER OF SAMPLES=',I4)
      WRITE(6,102)IRUN
102   FORMAT(1H,'RUN NUMBER-IRUN=',I4)
      WRITE(6,103)IHRS,IMIN
103   FORMAT(1H ,'ACTUAL RUN TIME OF THE FURNACE WILL BE=',I2,'HRS:',
     1 I2,'MINS.')
      GOTO 998
999   CALL PRINT(' ...ERROR- NS=0 IS INVALID...')
      CALL PRINT(' USE "CHA" COMMAND')
      GOTO 700
998   CALL SUBIR
      CALL SEND
      STOP
      END
      SUBROUTINE Q(A,I)
      REAL N,NC
      DATA Y/1HY/,N/1HN/,NC/2HNC/
      CALL PRINT(' ')
      CALL PRINT(A)
      READ(5,1000)ANS
1000  FORMAT(A4)
      I=0
      IF(ANS.EQ.NC)GOTO 92
      IF(ANS.EQ.Y)GOTO 90
      IF(ANS.EQ.N)GOTO 91
      GOTO 92
90    I=1
      GOTO 92
91    I=-1
92    CONTINUE
      RETURN
      END
      SUBROUTINE CONST(GK,T,TI,TD,TF)
      CALL PRINT(' GAIN')
      READ(5,2000)GK
      CALL PRINT(' SAMPLING INTERVAL')
      READ(5,2000)T
      CALL PRINT(' INTEGRAL ACTION TIME')
      READ(5,2000)TI
      CALL PRINT(' DERIVATIVE ACTION TIME')
      READ(5,2000)TD
      CALL PRINT(' FILTER TIME CONSTANT')
      READ(5,2000)TF
2000  FORMAT(F7.3)
      RETURN
      END
      SUBROUTINE CALCU(GK,T,TI,TD,TF,XK1,XK2,XK3,XK4)
      P1=GK*T
      P2=TF+T
      XK1=(P1/P2)*(1.0+(T/TI)+(TD/T))
      XK2=-((P1/P2)*(1.0+((2.0*TD)/T)))
      XK3=(GK*TD)/P2
      XK4=TF/P2
```

```
      RETURN
      END
      SUBROUTINE LIST
      WRITE(6,3000)
3000  FORMAT(1H ,'THE SYMBOLIC PARAMETERS TO BE USED IN THE CHANGE
     1 SUBROUTINE ARE:'/,
     1 'CM - CONTROL MODE'/,
     1 'LA - INITIAL LOAD ADDRESS IN ZONE 0'/,
     1 'NS - NUMBER OF SAMPLES'/,
     1 'MS - CONVEYOR MOTOR SPEED'/,
     1 'C1,C2,C3 - CONTROLLER STATUS FOR CONTROLLERS 1,2,3'/,
     1 'ISP1,ISP2,ISP3 - SET POINTS FOR CONTROLLERS 1,2,3'/,
     1 'GK1,GK2,GK3 - GAIN CINSTANTS FOR CONTROLLERS 1,2,3'/,
     1 'T1,T2,T3 - SAMPLING INTERVALS FOR CONTROLLERS 1,2,3'/,
     1 'TI1,TI2,TI3 - INTEGRAL ACTION TIMES.....'/,
     1 'TD1,TD2,TD3 - DERIVATIVE ACTION TIMES...'/,
     1 'TF1,TF2,TF3 - FILTER TIME CONSTANTS.....')
      CALL CLOSE(6)
      RETURN
      END
```

```
      SUBROUTINE CHANGE
      INTEGER RH
      DIMENSION S1(4),S2(15),S3(7),S4(2),S11(4),S22(15),IS3(7),IC(3)
      COMMON /BLK1/CM,C1,C2,C3
      COMMON /BLK2/NO1C,NO2C,NO3C
      COMMON /BLK3/GK1,GK2,GK3,T1,T2,T3,TI1,TI2,TI3,TD1,TD2,TD3,
     1 TF1,TF2,TF3
      COMMON /BLK4/LA,NS,MS,RH,ISP1,ISP2,ISP3
      EQUIVALENCE (CM,S11(1))
      EQUIVALENCE (IC(1),NO1C),(IC(2),NO2C),(IC(3),NO3C)
      EQUIVALENCE (GK1,S22(1))
      EQUIVALENCE (LA,IS3(1))
      DATA S1/'CM','C1','C2','C3'/
      DATA S2/'GK1','GK2','GK3','T1','T2','T3','TI1','TI2','TI3',
     1 'TD1','TD2','TD3','TF1','TF2','TF3'/
      DATA S3/'LA','NS','MS','RH','ISP1','ISP2','ISP3'/
      DATA S4/'ON','OFF'/
9     CALL PRINT(' SPECIFY THE PARAMETER YOU WANT TO CHANGE')
      READ(5,400)PARA
400   FORMAT(A4)
      IF(PARA.EQ.CM)GOTO 20
      DO 10 K=2,4
      IF(PARA.EQ.S1(K))GOTO 21
10    CONTINUE
      DO 11 I=1,15
      IF(PARA.EQ.S2(I))GOTO 22
11    CONTINUE
      DO 12 J=1,7
      IF(PARA.EQ.S3(J))GOTO 23
12    CONTINUE
20    WRITE(5,300)PARA,S11(1)
      WRITE(5,301)PARA
      READ(5,302)S11(1)
      GOTO 310
21    WRITE(5,300)PARA,S11(K)
300   FORMAT(' THE CURRENT VALUE OF ',A4,' =',A4)
      WRITE(5,301)PARA
301   FORMAT(' SPECIFY THE NEW VALUE OF ',A4)
      READ(5,302)S11(K)
302   FORMAT(A4)
      IF(S11(K).EQ.S4(1))GOTO 30
      IF(S11(K).EQ.S4(2))GOTO 31
      GOTO 310
30    IC(K)=256
      GOTO 310
31    IC(K)=0
      GOTO 310
22    WRITE(5,303)PARA,S22(I)
303   FORMAT(' THE CURRENT VALUE OF ',A4,' =',F7.3)
      WRITE(5,301)PARA
      READ(5,304)S22(I)
304   FORMAT(F7.3)
      GOTO 310
23    WRITE(5,305)PARA,IS3(J)
305   FORMAT(' THE CURRENT VALUE OF ',A4,' =',I4)
      WRITE(5,301)PARA
      READ(5,306)IS3(J)
306   FORMAT(I4)
310   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OPINFO
      INTEGER RH
      COMMON /BLK1/C1,C2,C3,CM
      COMMON /BLK3/GK1,GK2,GK3,T1,T2,T3,TI1,TI2,TI3,TD1,TD2,TD3,
     1 TF1,TF2,TF3
      COMMON /BLK4/LA,NS,MS,RH,ISP1,ISP2,ISP3
      WRITE(6,149)
  149 FORMAT(' **********************************************************
     1***************'/' DISPLAY OF OPERATOR SET INFORMATION*******
     2****************************')
      WRITE(6,150)CM
  150 FORMAT(1X/' CONTROL MODE--------------------:',4X,A1)
      WRITE(6,151)LA
  151 FORMAT(1X/' INITIAL LOAD ADDRESS IN ZONE 0---:',2X,I4)
      WRITE(6,152)NS
  152 FORMAT(1X/' NUMBER OF SAMPLES---------------:',2X,I4)
      WRITE(6,153)MS
  153 FORMAT(1X/' MOTOR SPEED---------------------:',2X,I4)
      WRITE(6,165)RH
  165 FORMAT(1X/' RUN TIME FOR THE FURNACE IN HOURS:',2X,I4)
      WRITE(6,154)
  154 FORMAT(1X/'CONTROLLER NO----------:',8X,'NO-1',9X,'NO-2',9X,
     1  'NO-3')
      WRITE(6,155)C1,C2,C3
  155 FORMAT(1X/' CONTROLLER STATUS-----:',9X,A3,10X,A3,10X,A3)
      WRITE(6,156)ISP1,ISP2,ISP3
  156 FORMAT(1X/' SET POINTS------------:',9X,I3,10X,I3,10X,I3)
      WRITE(6,157)
  157 FORMAT(1X/' CONTROLLER CONSTANTS')
      WRITE(6,158)GK1,GK2,GK3
  158 FORMAT(1X/' GAIN------------------:',6X,F7.3,6X,F7.3,6X,F7.3)
      WRITE(6,159)T1,T2,T3
  159 FORMAT(1X/' SAMPLING INTERVAL-----:',6X,F7.3,6X,F7.3,6X,F7.3)
      WRITE(6,160)
  160 FORMAT(' IN SECONDS')
      WRITE(6,161)TI1,TI2,TI3
  161 FORMAT(1X/' INTEGRAL ACTION TIME--:',6X,F7.3,6X,F7.3,6X,F7.3)
      WRITE(6,162)TD1,TD2,TD3
  162 FORMAT(1X/' DERIVATIVE ACTION TIME:',6X,F7.3,6X,F7.3,6X,F7.3)
      WRITE(6,163)TF1,TF2,TF3
  163 FORMAT(1X/' FILTER TIME CONSTANT--:',6X,F7.3,6X,F7.3,6X,F7.3)
      WRITE(6,164)
  164 FORMAT(1X/' **********************************************************
     1******************'/' END OF INFORMATION**********************
     2*****************************')
      CALL CLOSE(6)
      RETURN
      END
```

```
      SUBROUTINE SUBIR
      DIMENSION XK(4,3),A(12),AR(12),SF(8),NAA(12),IA(12,8),
     1 IXK(4,3)
      COMMON /BLK5/XK,IXK
      EQUIVALENCE (XK(1,1),A(1))
      EQUIVALENCE (IXK(1,1),NAA(1))
      DATA SF/0.5,0.25,0.125,0.0625,0.03125,0.015625,
     1 0.0078125,0.00390625/
      DO 2 I=1,12
      P(I)=ABS(A(I))
      IP=IFIX(P(I))
      AR(I)=P(I)-FLOAT(IP)
      BA=AR(I)
      DO 3 N=1,8
      TA(N)=BA-SF(N)
      IF(TA(N))25,29,29
25    IA(I,N)=0
      GOTO 3
29    IA(I,N)=1
      BA=TA(N)
3     CONTINUE
2     CONTINUE
      DO 50 I=1,12
      CALL NUMB (IA(I,1),IA(I,2),IA(I,3),IA(I,4),IA(I,5),IA(I,6),
     1 IA(I,7),IA(I,8),NAA(I))
50    CONTINUE
      WRITE(6,300)
300   FORMAT(1H ,4X,'A(I)',4X,'FRACTION OF A(I)',4X,'BINARY FRACTION',
     1 4X,'OCTAL EQU',4X,'INTEGER EQU')
      DO 20 I=1,12
      WRITE(6,112)A(I),AR(I),(IA(I,N),N=1,8),NAA(I),NAA(I)
112   FORMAT(1H ,2X,F7.3,6X,F6.3,12X,8(I1),8X,O3,8X,I3)
20    CONTINUE
      WRITE(6,200)
200   FORMAT(1H ,'FINISH')
      CALL CLOSE(6)
      RETURN
      END
```

```
      SUBROUTINE SEND
      INTEGER RH
      DIMENSION ISA(64),ISD(64),N1(4),N2(4),N3(4),Z(12)
      DIMENSION XK(4,3),IXK(4,3)
      COMMON /BLK2/NO1C,NO2C,NO3C
      COMMON /BLK3/GK1,GK2,GK3,T1,T2,T3,TI1,TI2,TI3,TD1,TD2,TD3,
     1 TF1,TF2,TF3
      COMMON /BLK4/LA,NS,MS,RH,ISP1,ISP2,ISP3
      COMMON /BLK5/XK,IXK
      COMMON /BLK6/KCM,IRCPS,IRFPS,IWCPS,IWFPS
      COMMON /BLK7/ISA,ISD,IRUN
      EQUIVALENCE (XK(1,1),Z(1))
      EQUIVALENCE (IXK(1,1),N1(1))
      EQUIVALENCE (IXK(1,2),N2(1))
      EQUIVALENCE (IXK(1,3),N3(1))
      ISA(64)=128
      KCM=1
      IRCPS=10
      IRFPS=11
      IWCPS=12
      IWFPS=13
      DO 1 N=1,63
      ISA(64-N)=ISA(64)+N
1     CONTINUE
      ISD(1)=NO1C
      ISD(2)=NO2C
      ISD(3)=NO3C
      ISD(4)=LA
      ISD(5)=NS
      ISD(6)=MS
      ISD(7)=KCM
      ISD(8)=IRUN
      ISD(9)=ISP1
      ISD(14)=INT(T1)
      ISD(17)=ISP2
      ISD(22)=INT(T2)
      ISD(25)=ISP3
      ISD(30)=INT(T3)
      DO 4 J=1,2
      ISD(14+J)=0
      ISD(22+J)=0
      ISD(30+J)=0
4     CONTINUE
      DO 5 K=1,28
      ISD(32+K)=0
5     CONTINUE
      DO 7 M=1,4
      ISD(9+M)=N1(M)
      ISD(17+M)=N2(M)
      ISD(25+M)=N3(M)
7     CONTINUE
      ISD(61)=IRCPS
      ISD(62)=IRFPS
      ISD(63)=IWCPS
      ISD(64)=IWFPS
      DO 3 N=1,64
      WRITE(6,2)ISA(N),ISD(N)
2     FORMAT(2(I4,3X))
3     CONTINUE
      RETURN
      END
```