THE APPLICATION OF MICROCOMPUTERS

IN

LARGE INDUSTRIAL INSTALLATIONS

(DISTRIBUTED MICROCOMPUTERS)

A Thesis submitted for Examination
for the Degree of Master of Engineering

by

Alan. W. Cotton.

Dip. E.E., Dip. Electronic Eng.,

B. Elec. Eng. (V.I.C.)

DEPARTMENT

OF

ELECTRICAL AND ELECTRONIC ENGINEERING,

FOOTSCRAY INSTITUTE OF TECHNOLOGY.

1984

SYNOPSIS

This thesis is concerned with the study of Microcomputers in Large Industrial Installations as a replacement for the traditional remote control and monitoring techniques.

A typical conveyor transport installation is described and was used as the basis for the system model developed by the author.

The thesis outlines the traditional techniques of control and data transmission in a widely distributed conveyor network and shows how they were used, or modified, for a microcomputer-based scheme.

In addition to the method of data transmission adopted by the author for the system model, the need for standard data transmission techniques (including error checking) is also described.

To be complete, research into distributed microcomputer systems must cover computer techniques (software and hardware), input/output requirements and linking techniques. The thesis describes each facet in detail, and includes the problems encountered during implementation.

The thesis also includes an outline of the Loy Yang project which is one of two systems for the State Electricity Commission of Victoria (SECV) where some of the concepts have been implemented since the research was completed.

ACKNOWLEDGEMENTS

Acknowledgement is gratefully extended to Messrs W A Evans and G J Lowe, to the Footscray Institute of Technology and to the State Electricity Commission of Victoria for providing the opportunity and financial assistance to carry out this research project.

The project was supervised by Messrs W A Evans and G J Lowe, for whose advice I am grateful.

There are many people I wish to thank, in particular the following:

. My wife - for her patience throughout the work.

. Messrs J M Alexander, H R Bunting and E J P Clayfield - for their guidance, continued support and encouragement.

. Messrs H W Bloxom and R M Glenn - for the original motivations behind the research.

. Messrs J Ikin, D Marshall and R Said - for their assistance in the compilation of this thesis.

. Messrs I Strachan, D Jagmin, L Omachen, G Atkinson, D Wilson and T Stork - for their assistance.

. Miss E C Watson, Mrs L Price and Mrs H Langley for the organising and the typing of the thesis.

STATEMENT OF ORIGINALITY

I certify that the work reported in this thesis is my own in the following respects:

(a) The writing of this thesis is my own work.

(b) The design and implementation of the distributed control system, writing and testing of software, and the development of the electronic circuitry are my own work.

No part of this work has been presented by the author for any other Degree. The author's publication relating to the topic is listed below and a reprint is included in Appendix E.

LOWE, G J, and COTTON, A W,
"Hierarchical Control Using Satellite Microprocessors" Conference on Microprocessor Systems, The Institution of Engineers, Australia, 15-16 November, 1977.

Alan W Cotton

TABLE OF CONTENTS

APPENDICES

PHOTOGRAPHS

CHAPTER 1 : INTRODUCTION

1.1 BACKGROUND

1.1.1 General

A prerequisite for the overall development, co-ordination and operation of large complex industrial installations is a centralised control system. One example is the system required by the State Electricity Commission of Victoria for the control and monitoring of the mining of brown coal in the Latrobe Valley. As the size of the power stations increase, so does the demand for fossil fuel and, to meet this demand, the open cut operations become more and more complex. To keep the bunkers of the power stations full, a network of dredgers and conveyors in an open cut is used to excavate and transport the coal.

In the early stages of development of the Latrobe Valley coalfields, the supply of coal was fairly easy to co-ordinate because the demand for fuel was low but, as the demand increased and the transport distances involved also increased, it became necessary to optimise the control and monitoring. In practice, a permanent centralised control centre is used and is located remote from the working faces of the open cut. One of the functions of the control centre is to co-ordinate the activities on a daily and long-term basis. Without the control centre, the conveyor network would be most inefficient in its operation. The amount of time lost starting and stopping the conveyors without overall co-ordination would be considerable. A further function of the control centre is to optimise the operations of the conveyor network.

Since the early 1960s, large industrial installations have combined both central minicomputer and telemetry techniques to control the activities of a complex process. In the case of an open cut conveyor transport network, the use of computers had been confined to the central control system with hardwired links (a single wire link for each signal monitored) to the plant.

1.1.2 The Process Under Investigation

Typically, an open cut conveyor transport network consists of manned dredgers and stackers and unmanned conveyors, power distribution centres and pumping stations. During start-up, normal operations and stopping the network must be controlled to avoid overfilling and spillage at the transfer points.

With items of plant being located over a wide area there are many remote control functions to be performed, such as the starting and stopping of multiple motor drives, belt slip detection and motor protection. Traditionally, these functions have been controlled mainly by relay based remote control sub-systems with some of the later conveyor control systems of the 1970's using discrete solid state logic.

Each of the traditional style control sub-systems also has an interface between the control sub-system and the telemetry sub-system which, because of the physical connections required, is expensive. The combination of the control and the telemetry sub-systems provides the remote control function from the control centre. For each output from the control centre there was a separate input for the conveyor control sub-system which was connected to the output of the control centre telemetry sub-system. The combined sub-systems form the "remote slaves"

in a large, widely distributed conveyor control network. The search for a simpler interface was one of the major objectives of this study.

In addition to the overall control functions, there are monitoring or data acquisition functions which have to be carried out using the same interface between the control and telemetry sub-systems. The data is used to determine the reason for plant stoppages and hence loss of production. In the case of the unattended plant (conveyor and pumping stations) the importance of the information to be transmitted to the control centre is much greater than with the manned plant. On the manned plant, the information can be supplemented by voice communications.

1.1.3 Developments in Open Cut Control

Between 1966 and 1970, a remote control and monitoring system was installed in the State Electricity Commission's Morwell Open Cut to improve the reporting of plant stoppages. It consisted of a single central mini-computer with 16k of memory and several crossbar scanners for scanning the field inputs. There was no preprocessing of information at the plant, except for starting and stopping sequences and overall control was maintained from the control centre via the central minicomputer.

During the early 1970s, it was realised that the concepts used in the Morwell system would have to be improved for the next system required for the redeveloped Yallourn Open Cut. The Morwell system took six seconds to scan all of the field inputs and it was therefore not always possible to determine the correct chronological sequence of events during the stoppage of a conveyor line because the field inputs could change several times during the scan period.

During the period from 1972 to 1974, the author was involved in investigations into ways of improving the recording of the chronological sequence of events at individual conveyors.

It was not until 1975, when the first microcomputer was installed in the Morwell Open Cut for controlling a pumping station, that an economic solution became available. It was decided to try a prototype microcomputer-based control system. If any major problems did develop, it was possible to man the pumping station on a limited basis without any risk to the coal supplies. Whereas, with a conveyor, if the prototype control system had failed, it would have been more difficult to man the conveyor because of the dusty environment; and further more any stoppages would have affected coal supplies. The Morwell pumping station prototype controller provided the basis for a much larger distributed system and enabled a study to be made of the effects of the industrial environment without affecting coal supplies.

The author was involved in further developments in 1976 when a second microcomputer-based pumping station was installed at the Yallourn Open Cut. The Yallourn pumping station unit included the facility for transmitting eight status signals to an open cut control centre display controlled by the pumping station microcomputer.

These two pumping stations formed the basis for the research into the large scale application of the microcomputer in control systems for open cut conveyor control. Before proceeding with an actual conveyor system, it was decided to investigate the suitability of a microcomputer-based distributed control system. Approval was obtained for the author to investigate the application of microcomputers in a distributed system

with independent control at the remote locations in an overall hierarchical master/slave configuration. This investigation was necessary before proceeding with a major change in philosophy that could affect the operation of the conveyors and hence the supply of coal to the power stations.

1.2 OBJECTIVES OF THE RESEARCH

The concepts forming the basis of the research were formulated while working with the Morwell Open Cut control system and the two microcomputer-based pumping stations.

The Morwell Open Cut control system configuration could not always provide chronological recording of events. When an event occurred at a conveyor, the result had to be detected by the central minicomputer immediately. Therefore, an objective of the research was to demonstrate, using a microcomputer-based conveyor control system, that it is possible to record the events at the remote conveyors in the correct sequence regardless of the task being performed by the central system.

The pumping station at Yallourn demonstrated that if several events were to be scanned then an error-free method of transmitting data would have to be adopted.

In addition to the study of the technique of transmitting data, the research also aimed to investigate the problem of the compatibility of transmitted data from different systems. In developing the conveyor transport network, different manufacturer's control equipment had been used on the individual conveyors and this continues to be the case.

A model with a minicomputer master and a distributed microcomputer-based system representing two conveyors linked through high speed serial data links to the control centre was used to achieve the objectives of the study. These objectives can be summarised as follows:

. To demonstrate that independent control and monitoring can be maintained by the remote microcomputer-based sub-systems in the event of the link to the central computer being lost;

. To investigate the possibility of improving the chronological recording of data and to provide secure data transmission from the remote sub-systems;

. To distribute the central system tasks, making it possible to improve the management reports at the control centre;

. To remove single points of failure in an overall control system by distributing the tasks;

. To determine the necessary facilities and design techniques required for a microcomputer-based system;

. To obtain an understanding of the problems and software requirements associated with using microcomputer-based distributed sub-systems;

. To investigate the possibility of removing the expensive interface required between the remote plant control sub-system and the data acquisition sub-system (refer to Section 1.1.2) by

integrating the task of data acquisition in the sub-system control tasks;

. To determine the requirements for linking different manufacturers' microcomputer-based sub-systems together;

. To study the different types of system configurations possible;

. To replace the traditional hardwired conveyor control systems (relay or solid state) and telemetry systems with a more flexible system utilising serial data transmission, better suited to the one-off nature associated with large conveyors;

. To demonstrate that it is possible to change a conveyor's control functions by down-line loading (transmitting) new control parameters in the conveyor sub-system.

CHAPTER 2 : LITERATURE REVIEW

2.1 TRADITIONAL TECHNIQUES OF OPEN CUT CONTROL

Initially during the 1920's horse-drawn carts were used to transport coal to the power stations then as the coal demand increased the carts were replaced by electric trains. The trains were controlled from a central communications centre with some control of the railway signals. During the 1950s, it was decided that it would be more economic to use conveyors to transport the coal to the power stations. As the conveyor equipment increased in size and number, it became necessary to centralise the control and monitoring of the conveyor transport network. The first major remote control and monitoring system for open cut control used by the SECV was the system installed in the Morwell Open Cut.

The evaluation of technical and economical aspects by the Control System Suppliers showed the expediency of employing both minicomputer and telemetry techniques (using cross-bar scanners), to control the activities of the Morwell Open Cut. From the experiences of Hailstone (5) and the practical experiences of the author, the combination of a minicomputer and a telemetry system, would have been a relatively new technique for the 1960s. It has now been the practice for the past 25 years to use some form of telemetry (frequency division multiplexing, (FDM) or time division multiplexing,(TDM)) or direct wiring to connect the remote plant control systems to a central control centre minicomputer system to provide the overall system for the control and monitoring of the plant.

Other observations made and supported by Jenkins (6) in the area of distributed computers and Prophet (8) in his article ("A new tool for

Production Control") indicated that it would be feasible to implement major control functions by using dedicated computers. The work undertaken independently by Jenkins, Prophet and the author during the early 1970s suggested that a microprocessor-based computer sub-system could be dedicated to performing control and monitoring tasks in a distributed computer system.

The trends in control system techniques during the 1970s is discussed by Sargent and Lundy (9). In their paper they outline the reasons for an increase in the use of solid state control systems leading to the programmable devices of the 1980s.

A problem with the early minicomputer control system used in the Morwell Open Cut was the high cost and complexity of the interconnecting cabling system.

A second problem area highlighted by the Morwell Open Cut system was the low speed of handling data. In order to increase the thoughput of a data acquisition system, the speed of data transmission had to be increased. For the FDM and TDM systems, the environmental conditions determine the speed of transmission and as a result transmission rates of all sub-systems in the system had to be adjusted to the same rate to be compatible with the the slowest link. Deshon (2) indicated that, in the future, it should be possible with the microcomputer sub-system to vary the transmission rate to suit the environment. This is an important aspect for an open cut system which could be subject to electrical interference. The microcomputer would provide the automatic re-transmission of the message at a different speed or bit rate.

In the commercial data gathering systems of the late 1970's, a defined protocol or message format was used to ensure the secure transmission of information, but this was not the case in the industrial systems.

There is a number of transmission and error detection codes available (refer to Appendix F for more details). Transmission codes "Bose Chadahuri", "Baudot", "ASCII" and "BCD" (30) and error codes "CRC-12", "CRC-16" and "CRC-CCITT" (30) are the most commonly used. The message protocol of a data transmission system is basically a set of rules for operating the communication system. McNamara (30) in his book ("Technical Aspects of Data Communications") indicates some of the reasons for these rules, i.e.:

. The determination of which part of the message constitutes the control characters or the data portion;

. To eliminate duplicate messages, to avoid the loss of messages, and to properly identify messages that are re-transmitted by the error control system;

. The determination of which station is going to transmit or receive;

. Solving the problem of which message to transmit when there is no data to send;

. Solving the problem of which steps should be carried out if message flow suddenly ceases;

. The process of initiating transmission in an idle or quiescent system is often complex.

It is unfortunate that computer and peripheral equipment suppliers have developed their own protocols. Nevertheless, certain protocols (refer to Appendix F (p.183)) are now becoming standards by design or through wider use in industry. Standards institutes, such as American National Standards Institute (NASI) and the International Standards Organisation (ISO), are about to adopt certain protocols as standard protocols (12). The Australian Standards Association has published a data transmission standard (25) in an attempt to standardise on data transmission methods. Because of the large number of control equipment suppliers involved in conveyor control systems, data protocol was an important area investigated during this research.

2.2 MICROCOMPUTER HISTORY

In 1971 the Intel Corporation (INTEL) (a large semi-conductor supplier) produced a programmable device as a solution to a request for a flexible control system. The result was the INTEL 4004, the world's first microprocessor (a four (4) bit device). It was very slow by modern computer standards and it could address only 4k bytes of memory, but it was programmable and relatively inexpensive. It was then followed by the 8008, an eight bit device. As stated by Tobias (11) in his paper, microprocessor designs have continued to advance with more powerful and faster devices being developed such as the 16 bit and the bit slice devices.

The microprocessor is very much a hardware oriented product and owes its development to two major factors associated with Large Scale Integration (LSI). One is the technological development of the semi-conductor industry in the LSI area. The other factor is the economics of LSI. Previously, a hardwired control system required a one-off design for each application. An alternative in the 1980s is the microcomputer which can be customised by a change of program. The microcomputer of the early 1980's consisted of one microprocessor chip with a dedicated or defined software program and the required input/output interfaces.

One problem is that most of the advances have been in hardware. There is still an almost universal under-estimation of software problems in industrial microcomputer systems similar to those experienced with the large computers and minicomputers ten years earlier.

2.3 MONITORING AND CONTROL DEVELOPMENTS

With the advent of solid state logic (discrete components, subsequently LSI) the control functions were achieved in a similar manner to the relay systems by using interconnected logical "AND" and "OR" discrete logic blocks (9).

The changeover to solid state control devices was considered a sufficient change in direction for control system design during the 1970s and this made the acceptance of the microcomputer more difficult.

The question was, "Should a company disregard many years of experience and use this new microcomputer technology, considering the problems encountered with minicomputers during the 1960s?". Obviously the microcomputer had to have definite advantages over the techniques in use at the time and, in addition, an understanding of the problems of implementing control requirements using microcomputers was essential if a company was to change its direction.

Falling microcomputer hardware costs and rising costs of one-off hardwired logic systems made it economically attractive to replace traditional methods of remote control and monitoring in large installations. It was feasible to combine remote control, real-time data acquisition and distributed computing at the remote plant locations using microcomputer-based sub-systems linked to a central master. Amendt (1) suggested that a design engineer must consider new approaches to control design and that learning the techniques associated with microcomputer-based systems would prove to be invaluable. At the time of this research, the microcomputer was not widely accepted as a possible method for industrial control and hence there was a need for a study of microcomputer techniques.

South (10) also suggested that the form of remote control sub-systems used at the time could be replaced by a microcomputer-based remote sub-system. There was no preprocessing of information by the earlier control sub-systems but it is now possible to distribute some of the central system's processing of the data to the microcomputer, thereby increasing the effectiveness of the overall control system. This was also supported by Deshon (2) and Dominquez and Tennant (3).

The basic aim of the research program was to investigate a method of improving the data throughput of a centralised conveyor control system. This was to be achieved by using microcomputers to control the transmission of the data from a simulated conveyor network without complicating the operating procedures of the system. The following Chapters outline how the microcomputer was adapted to enable the increase in the data throughput for an actual conveyor control system.

CHAPTER 3 : THE RESEARCH PROGRAM

3.1 DETERMINATION OF THE SUITABILITY OF MICROCOMPUTER SUB-SYSTEMS

3.1.1 General

In a large open cut industrial installation, the control system must handle multiple control tasks distributed over a wide area. The layout of the conveyor network as shown by Figure 3.1 represents at least 30 separate plant items. Each conveyor (L100, etc) and dredger (D14, etc) requires a dedicated control sub-system to carry out the control and monitoring functions at each remote location.

It was demonstrated during the research, using modelling techniques for a limited system, that it is feasible to use a microcomputer-based sub-system for a remote control and monitoring system configured as shown by Figure 3.2. The sub-system tasks were handled by dedicated slave microcomputers, one for each plant item. To distribute some of the central system tasks, the slave sub-systems were controlled by a "Sub-master" or "Front-end" which was in turn controlled by the master.

The natural division between the plant systems, as shown in the open cut layout of Figure 3.1, is the deciding factor in determining the number of slaves controlled by a front-end or sub-master. It is considered undesirable to have plant items on opposite sides of an interchange (or transfer) area or in different conveyor routes connected to the same front-end. To avoid any interaction, or the loss of more than one route (there are 24 possible routes in Figure 3.1), each front-end would have only one dredger group, bunker group or stacker group of plant items directly linked.

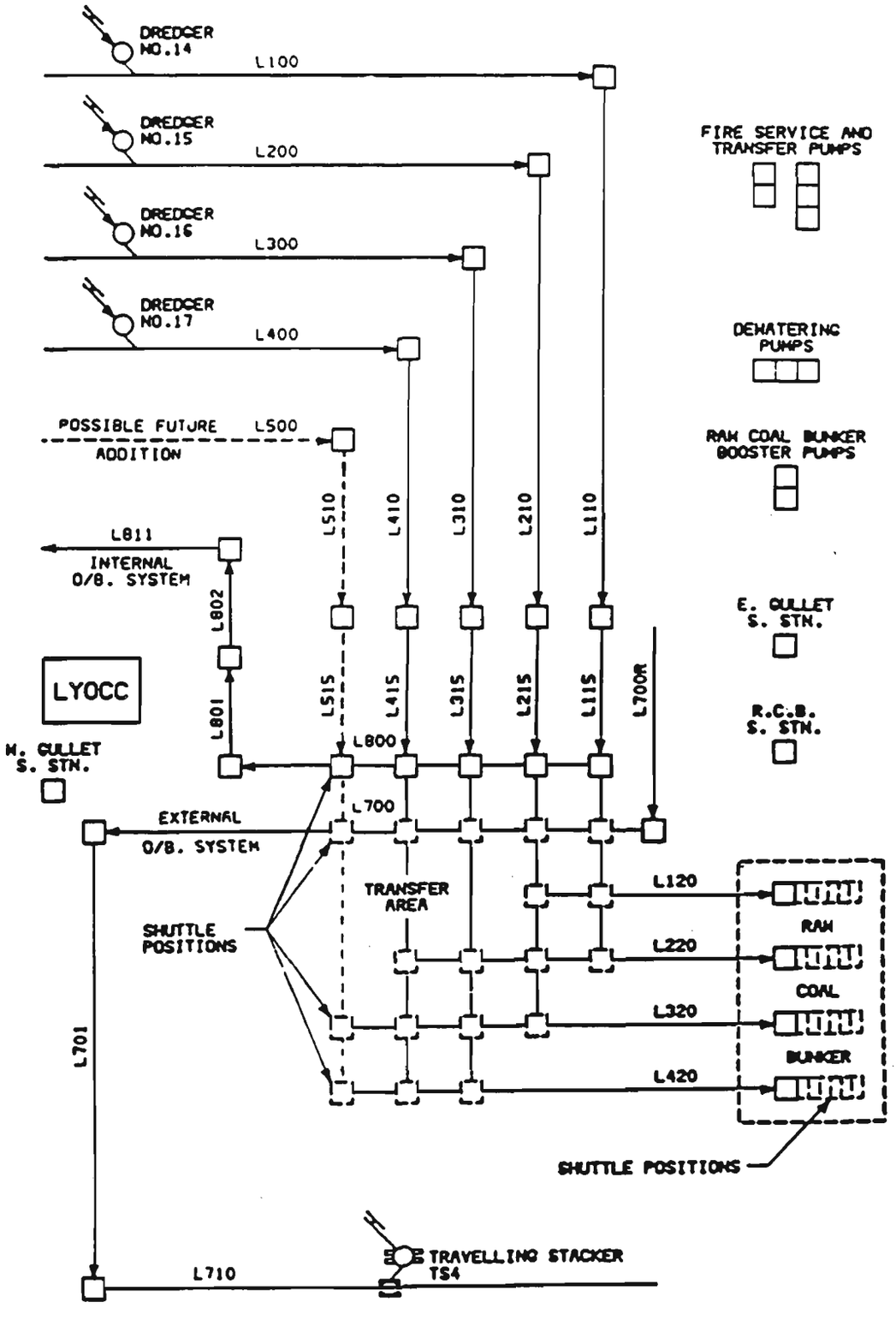


FIGURE 3.1 : LOY YANG OPEN CUT PLANT LAYOUT

A model consisting of one master, one front-end (sub-master) and two slave sub-systems was selected for the research program to check the concepts referred to in Section 1.2.

The model was configured in a similar manner to the full scale system of Figure 3.2. Each sub-system was connected by a high speed serial data link to the next level in the hierarchy. The master communicated with the front-end which in turn communicated with the slave sub-system using a defined message protocol.

The model represented the front-end (No 4) for the plant group associated with Dredger No 17 and the two slave sub-systems represented Conveyors L400 (Slave 1-S41) and L410 (Slave 2-S42) (see Figures 3.1 and 3.2). The research program was designed to provide a better understanding of the hardware and software requirements and the associated problems. Because a full scale system for the open cut represented many different facets (i.e. the interconnecting cables, the conveyor sub-systems and the overall control centre system) it was essential that the working model developed include all of these facets. The various levels in the hierarchy are discussed in detail in the following Sections.

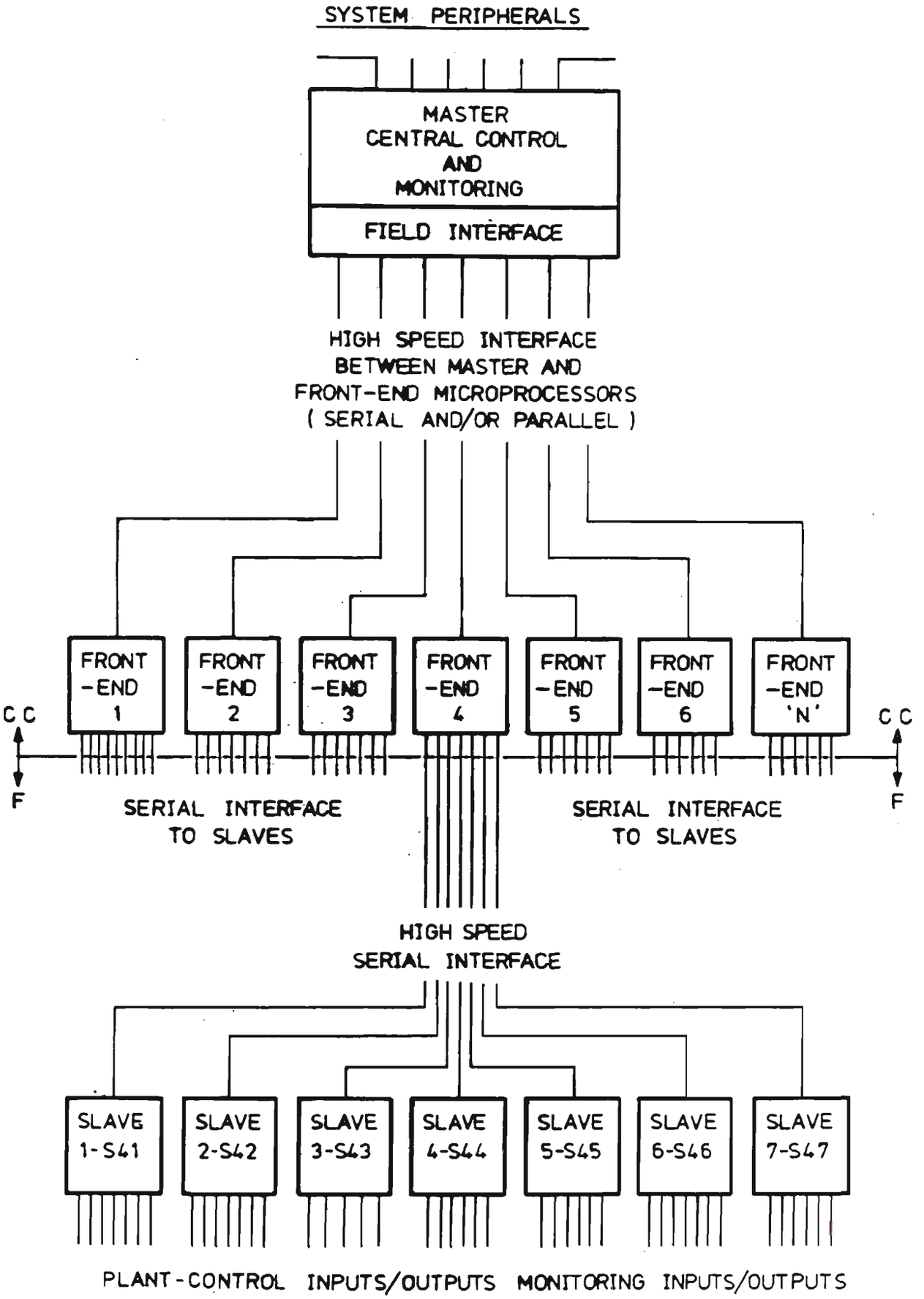


FIGURE 3.2 : SYSTEM CONFIGURATION

3.1.2 A Single/Multiple Microprocessor-based Sub-system

The sub-systems required to implement the system control strategy would be based on either a single microprocessor or a multiple microprocessor configuration.

3.1.2.1 A Single Microprocessor-based Microcomputer Sub-system

A single microprocessor chip microcomputer had to sequentially carry out the tasks previously performed by the traditional techniques and in addition demonstrate definite improvements.

The aim of replacing proven control and monitoring techniques by a single microprocessor-based sub-system, by the proper scheduling of multiple tasks, was to cover:

- the functions provided by a separate telemetry system;
- data acquisition via high-speed serial data links;
- preprocessing of data at remote plant locations;
- the problems of interconnecting different manufacturers' products using standard interfacing methods;
- the standardisation of sub-system hardware by having the differences in the local control algorithms in the individual software programs;

. a reduction in the central system functions, such as data storage and time 'tagging' (chronological recording) of events.

The microprocessor also enabled the development of an interactive diagnostic facility with VDU terminals which were used in conjunction with a limited English text (question and answer response) as described in 4.3.3.1.

3.1.2.2 A Multiple Microprocessor Microcomputer Sub-system

Instead of using a single microprocessor chip, it would have been possible to develop a sub-system that had multiple microprocessor chips controlling the various tasks. For example six microprocessors could have been dedicated to a sub-system with one for each specific task as follows:

. One for motor control, such as starting sequences or variable speed control, etc;

. One for protection monitoring (slip detection, faults, etc);

. One for data reduction on the information to be transmitted;

. One for data transmission, including control and message protocol emulation (standardisation possible by reprogramming for different formats);

. One for the different programming formats: such as relay ladder diagram or logic symbol programming for control;

. One for functional independence of tasks, such as having the local control functions separate from the data transmission function.

The multiple microprocessor-based sub-system would have simplified some of the complexity of the software required in the multiple task single processor sub-system. It is the author's view that each software task could have been written for the dedicated device, with less cost and complexity than for the single microprocessor software package. This multiple microprocessor-based sub-system model was not developed. At the time of this research a multiple microprocessor microcomputer was not available, refer to Chapter 7 regarding further work and recommendations.

3.1.3 Advantages and Disadvantages of Microcomputer-based Systems

An advantage of the microcomputer in some of the areas formerly dominated by the minicomputer is that it is an economical solution for single tasks. The conventional approach of using a minicomputer would be to use it for several parallel tasks. This is not ideal because a failure of a multiple-task minicomputer would affect the whole system. Using the model, it was demonstrated that by distributing the tasks over several microcomputers overall control was maintained during failures of the central minicomputer with higher reliability and at a lower cost.

Prior to this research it was anticipated that the use of microcomputer-based sub-systems for control would have the following advantages and disadvantages:

ADVANTAGES:

. Standardisation of hardware, with special software to suit the unique control requirements thus enabling the manufacturer to proceed with the hardware manufacture before completing the system design;

. Reduction in development and design of the hardware for each unique situation when compared to the one-off relay systems therefore each design only had to cover the different functional requirements by incorporating suitable software sub-routines;

. Reduction in central system dependence so that each sub-system can operate independently during failures of the central system;

. Reduction in central system tasks, especially in implementing the control requirements so that the remote sub-system need only transmit status changes as they occur;

. Distribution of the central system tasks to the microcomputer based sub-systems to provide increased reliability and flexibility;

. Reduction in the overall system hardware costs.

DISADVANTAGES:

. Changing to the new microcomputer-based sub-systems would require a change in existing fault finding techniques. In the past, fault-finding was achieved by visually checking through the relay

contacts for electrical continuity which is not possible with the microcomputer;

- In some cases, the introduction of a different style of technology (computer-based) would require the retraining of existing personnel;

- The cost of software could increase well above the original estimates if the software programmer does not fully understand the process to be controlled. This is a common problem with software based industrial systems.

These advantages and disadvantages were demonstrated and confirmed by the research work and have since the original study, been reinforced by the developments that have taken place at Loy Yang.

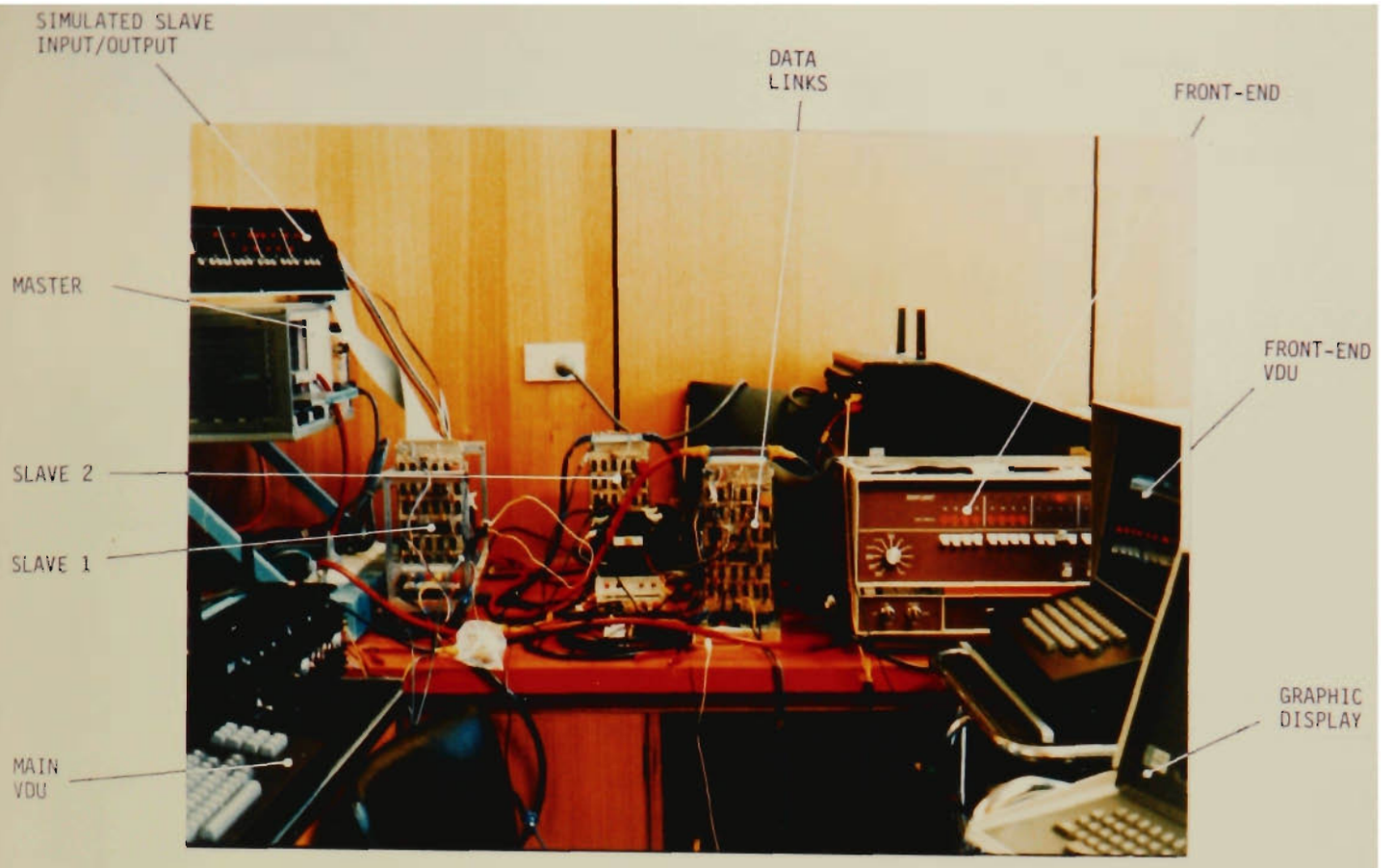
3.2 ESTABLISHMENT OF THE HIERARCHY OF THE PROTOTYPE SYSTEM

3.2.1 General

A hierarchical system consists of two or more levels of distributed sub-systems arranged in a pyramid or multi-level formation. At each level, a number of sub-systems (front-ends, slaves) operate in parallel. There is an iterative data transfer between the various levels with a preference for the data transfer down the pyramid to be treated as a command by the lower levels.

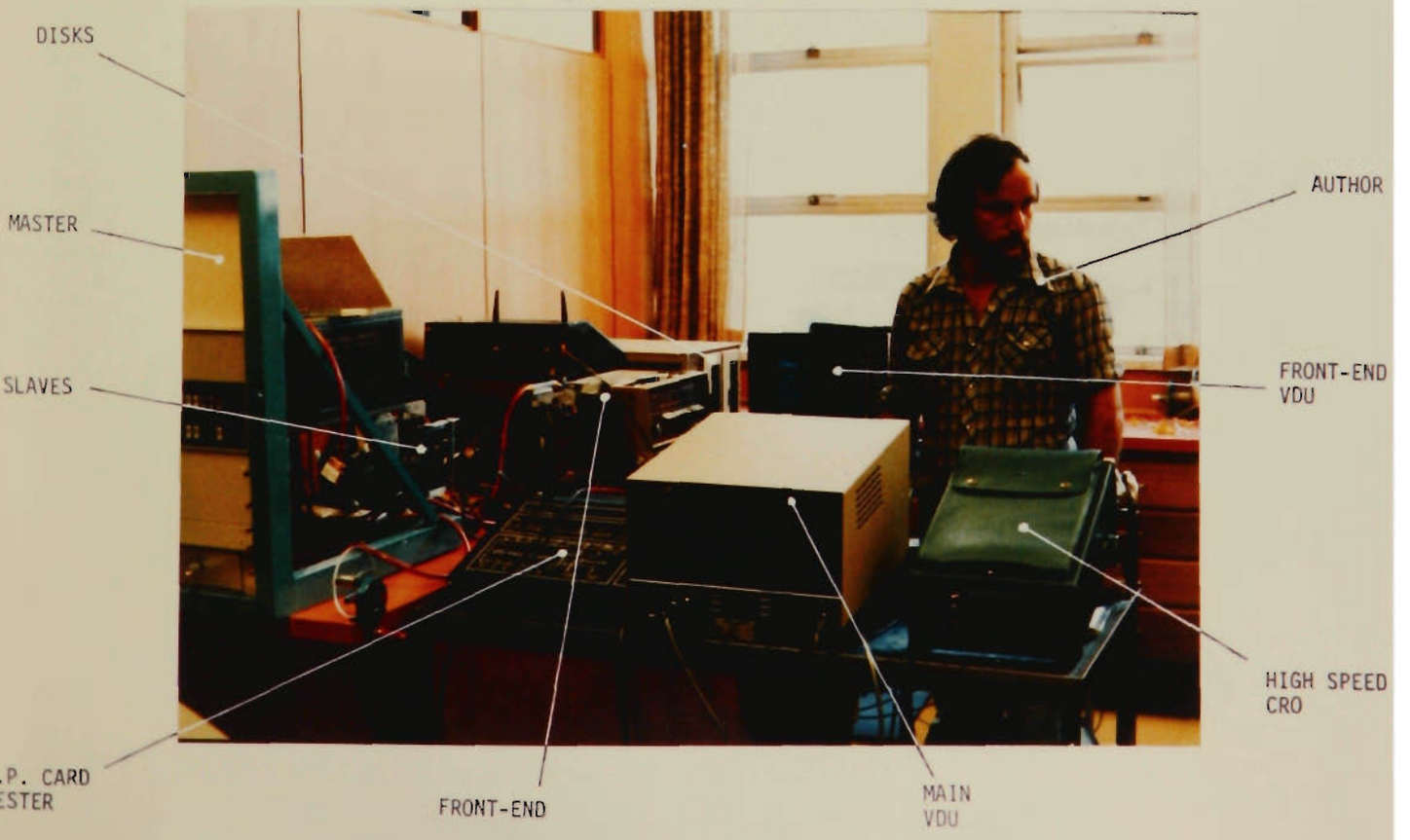
For the open cut plant situation considered in this project, the control centre master (a minicomputer) represented the highest level in the hierarchy with the front-ends (microcomputer-based) and the slaves (microcomputer-based) representing the second and third levels respectively. As outlined in Section 3.1.1, a front-end would control the slaves for the conveyors from the dredger to the transfer area or from the transfer area to the bunker as shown in Figure 3.1. The hierarchy of the control system was determined by the basic process to be controlled and the functional independence required between common or parallel tasks, such as a dredger to bunker route, taking into consideration any interchange area.

Accordingly, the hierarchy of the prototype system model was as shown in Figure 3.3 and Photograph No 2. This model enabled the requirements of an actual system to be investigated and demonstrated.



PHOTOGRAPH NO 1 - TOTAL SYSTEM INTERCONNECTION
(MASTER - FRONT-END - SLAVE 1, SLAVE 2)

RESEARCH LABORATORY



PHOTOGRAPH NO 2 - SYSTEM LAYOUT

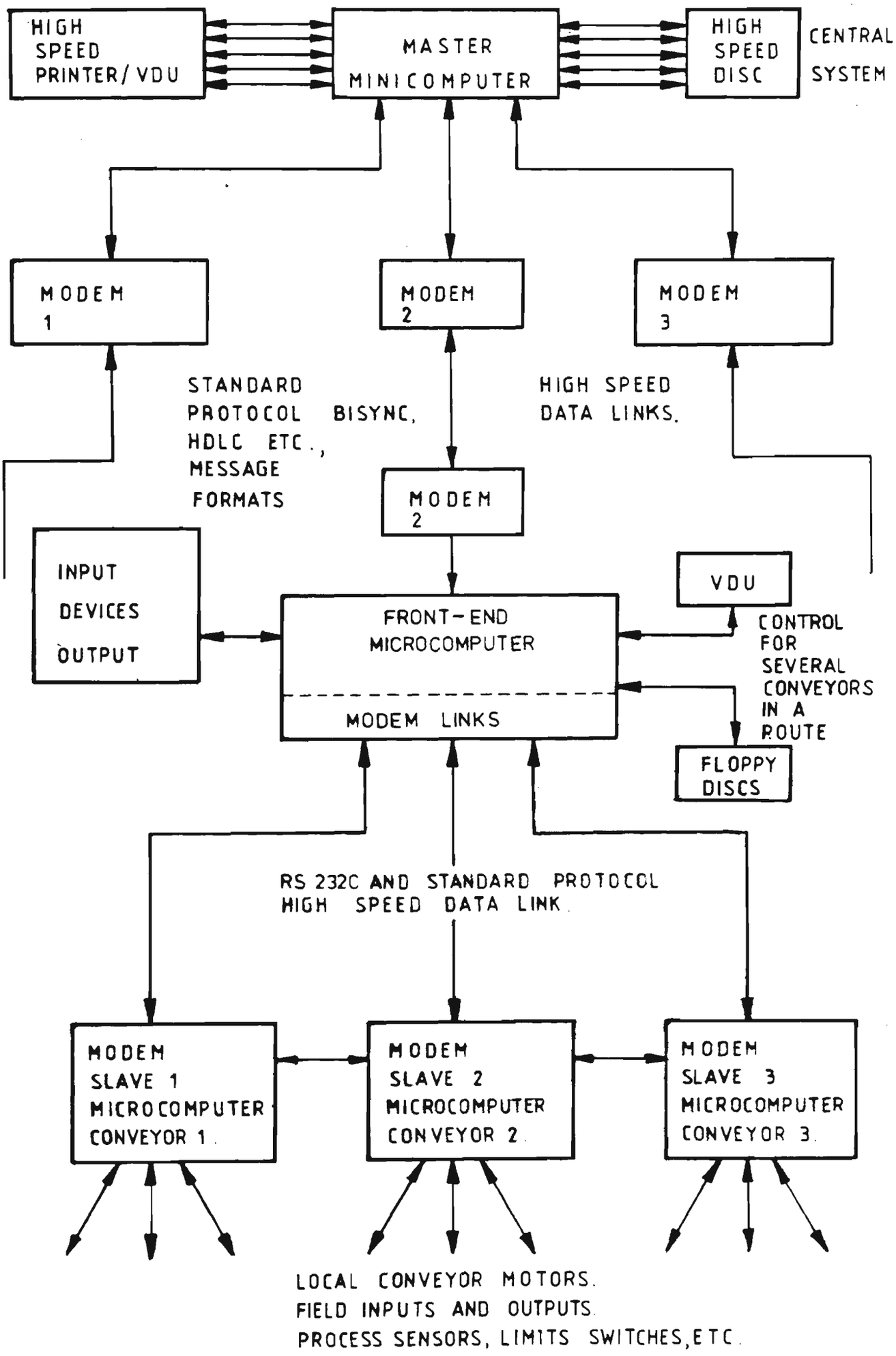


FIGURE 3.3 : A HIERARCHICAL SYSTEM : REPRESENTING A TYPICAL CONVEYOR NETWORK AS INVESTIGATED DURING THE RESEARCH

The anticipated configuration for an actual situation, as shown in Figure 3.1, (p.16) and Figure 3.3 consists of multiple microcomputers, each one with well defined tasks to perform. For a conveyor network, the overall control is maintained by the master (consisting of single or dual minicomputers). The next level in the system hierarchy is the front-end or sub-master level. Because of the number of front-ends, there must be a defined hierarchy or priority between the front-ends as determined by the conveyor route requirements.

From experience with earlier open cuts the most important group is the top conveyor line in the open cut followed by the lower groups. It is necessary to remove the top layers of overburden and coal in order to keep the alternative routes available. Therefore, the overburden group front-end is the highest priority front-end of the second level.

Similarly, the individual slaves also have a priority rating as determined by the position of the conveyor in the conveyor line. To avoid spillage, the last slave in the line is given the highest priority and to minimise the situation where a slave in the hierarchy was never serviced, a regular check must be initiated by the master.

The model as described in Chapters 3 and 4 is based on this configuration.

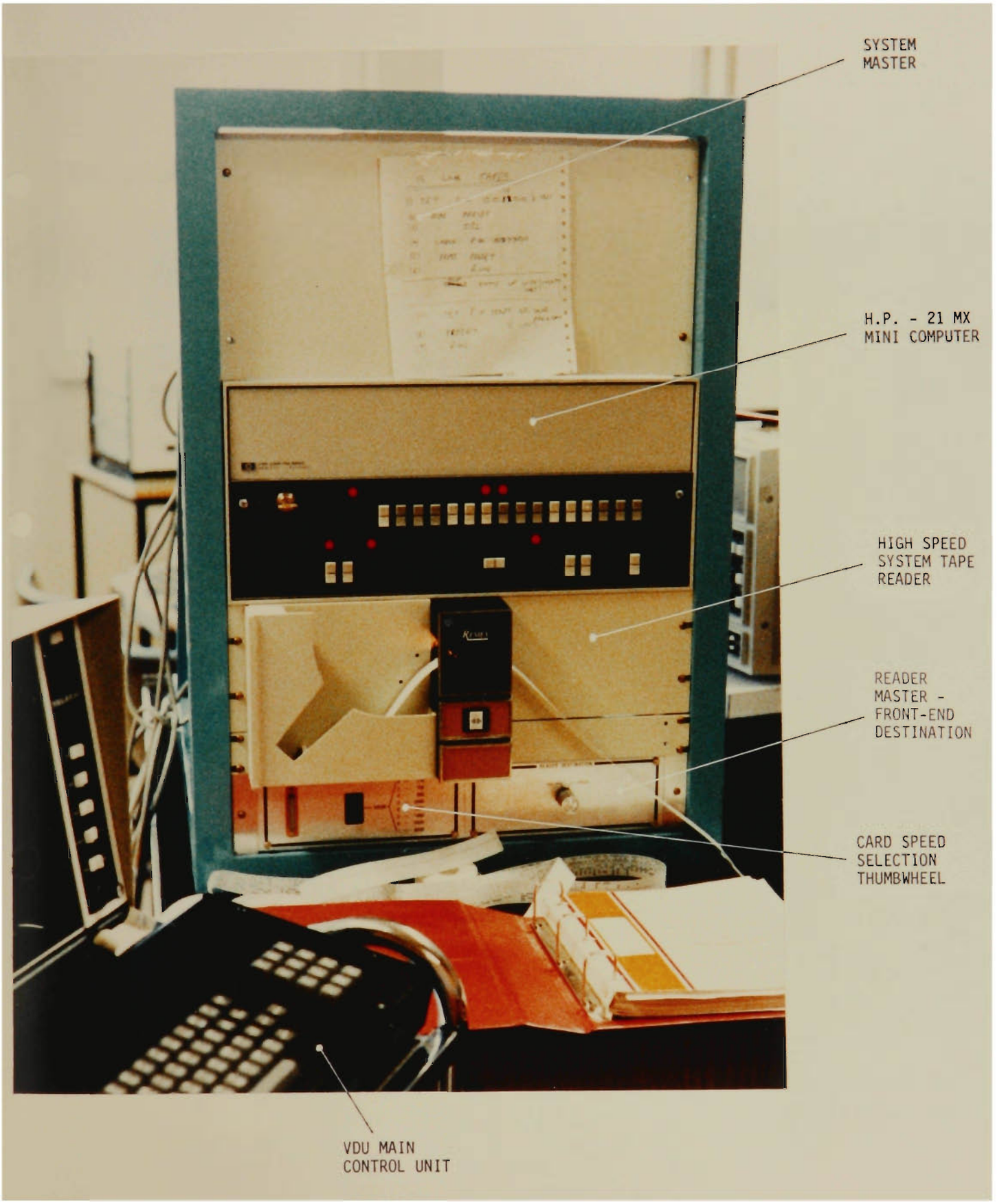
3.2.2 Master Sub-system

(Photographs Nos 4 and 5)

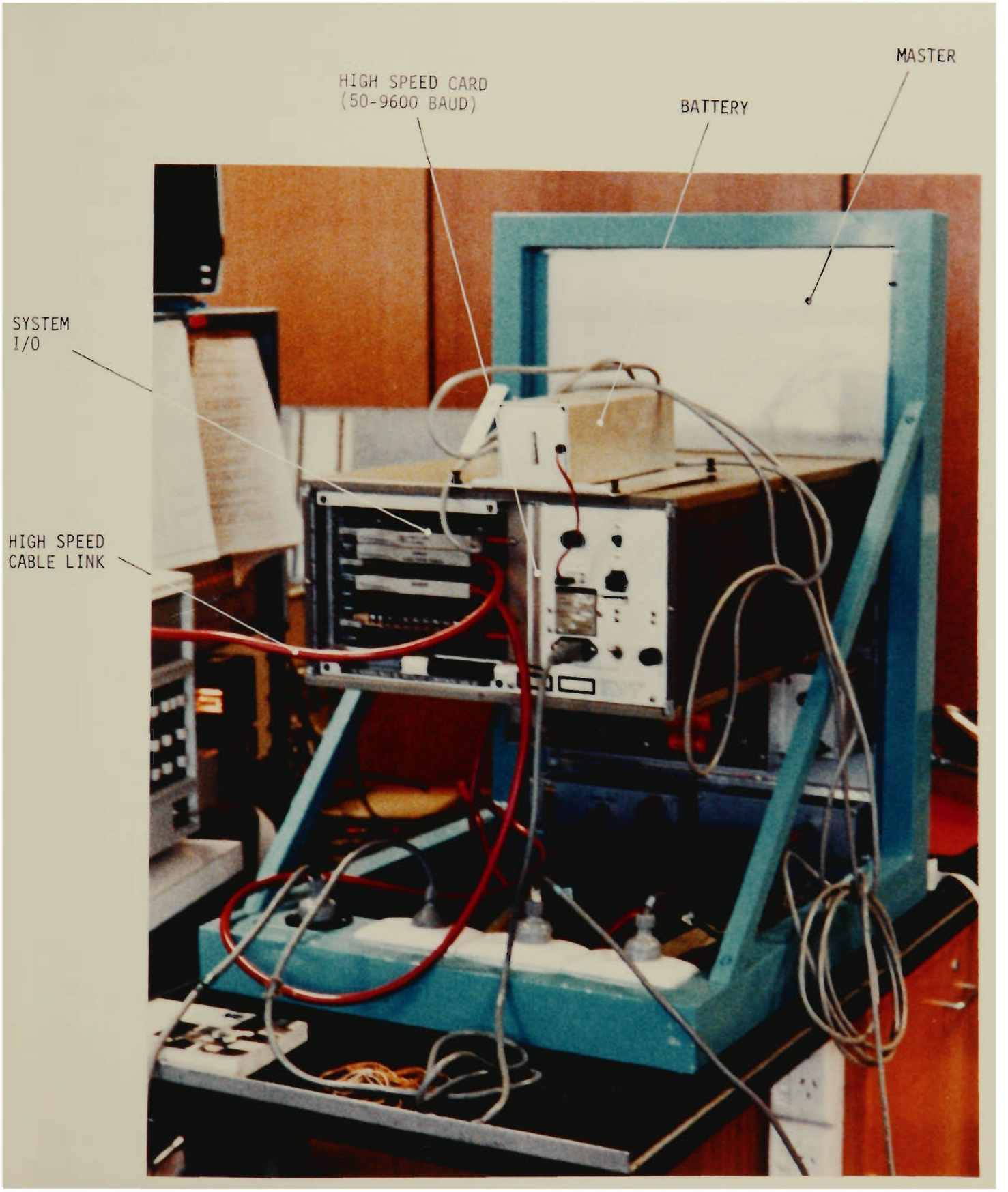
The 21MX Hewlett Packard minicomputer performed the task of system master for the model. It was controlled through the VDU keyboard which in turn controlled the graphic display of Figure 3.1 and the message transmission to the lower sub-systems (the front-end and the slaves).

Messages to the front-end and slaves (conveyors) were simulated in the master and transmitted via the high-speed serial data link developed for the 21MX minicomputer (refer to Photograph No 5). The message performed the same function as operating a start or stop button at the master. When the message was received by the front-end, it was decoded (as explained in Section 4.3.3.3 on (p.94)) and checked for errors before performing the task or command contained in the message. If the message contained a request for data from the front-end sub-system, the appropriate action of encoding the data was carried out. If the message was for a slave it was re-transmitted to the slaves by the front-end. The central master continued on with other tasks until it received an interrupt from the front-end.

In addition to checking for errors in messages from the master, the front-end also checked messages from the slaves before re-transmitting to the master. This reduced the load on the master because the front-end indicated to the slave that the data received had an error without interrupting the master. This error checking would be important in an actual conveyor network because the interconnecting control cables are located in the vicinity of high voltage equipment in the field and are therefore susceptible to electrical noise.



PHOTOGRAPH NO 4 - MASTER COMPUTER -
HEWLETT PACKARD
21 MX - FRONT VIEW



PHOTOGRAPH NO 5 - MASTER COMPUTER -
HEWLETT PACKARD
21 MX - REAR VIEW

3.2.3 Sub-master (Front-end) Sub-system

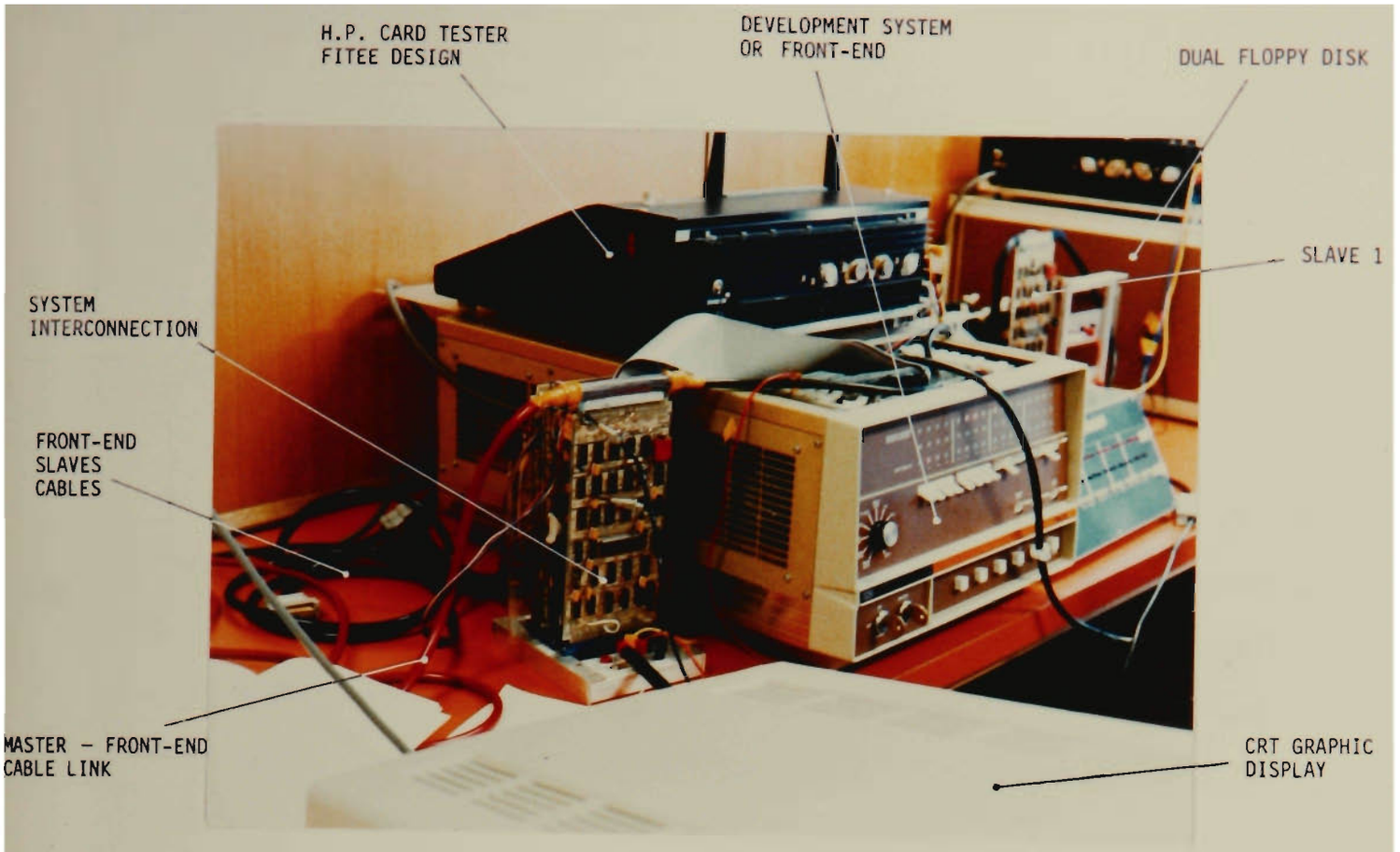
The second level of control in the hierarchy (Figure 3.3) was the front-end computer which was based on the National Semiconductor PACE microcomputer (refer Photograph No 6). The introduction of a front-end in the system strategy enabled the distribution of some of the functions normally performed by the master. In addition, the front-end sub-system also provided two secondary functions:

- The storage of programs for the model (the slave, front-end and master programs were stored on disc and subsequently down-line loaded on request);

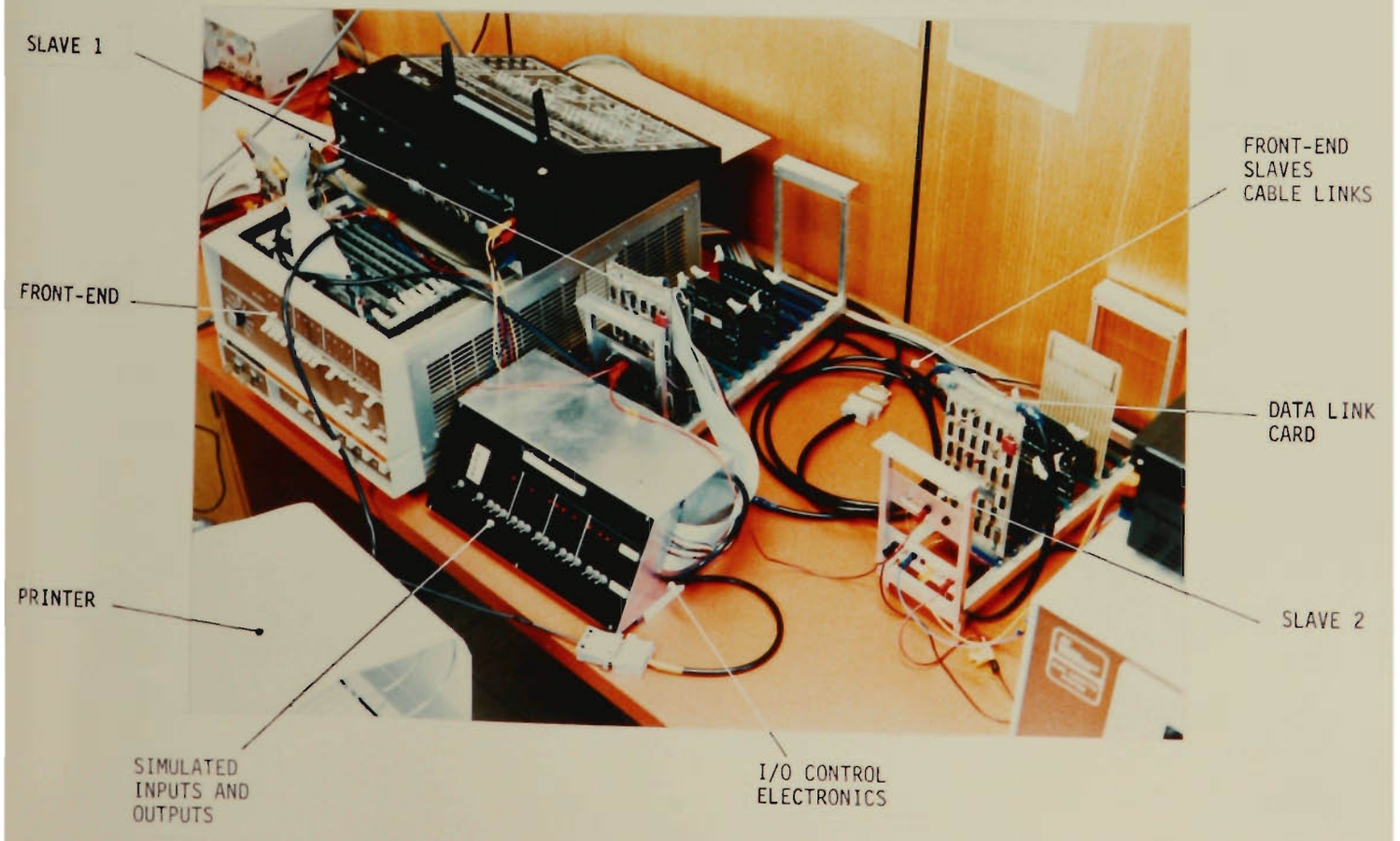
- Development facilities were provided for the microcomputer software and hardware via the National Semiconductor prototyping extender card.

The front-end serviced the messages from the master and re-transmitted to the slaves as outlined in Section 4.3.4. In the internal front-end hierarchy, the link to the master had the highest priority after all internal functions of the front-end were serviced.

The internal functions of the front-end were interrupts from the internal clock and the other software sub-routines (e.g. stack handling) during multiple sub-routine servicing. The stack function of the PACE Microprocessor was an internal logic function of the microprocessor chip that handled up to ten transfers of data words or sub-routine return addresses before requiring external memory for storage.



PHOTOGRAPH NO 6 - SYSTEM INTERCONNECTION AND FRONT-END ARRANGEMENT
(FRONT-END AND DEVELOPEMENT SYSTEM)



PHOTOGRAPH NO 7 - FRONT-END AND SLAVES 1 AND 2 LAYOUT

Distributed systems such as the model are subject to many coincident interrupt requests. In the model, in addition to random requests from the connected sub-systems, the front-end was required to service the front-end VDU (refer to Photograph No 2) and acknowledge requests such as time of day in hours, minutes and seconds.

The modular approach of the system model also allowed the removal of the front-end from the hierarchy. The software of the slave was designed to receive the protocol directly from any sub-system. Hence, for a small full scale system (e.g. during the initial development of an open cut with only one dredger and one conveyor line), it would be possible for the slave to be directly connected to the master.

3.2.4 Slave Sub-systems (Photograph No 7)

The third level in the hierarchy consisted of the microcomputer-based slave sub-systems and was also based on the PACE microcomputer. Each of the slaves had the same basic data transfer and internal management software packages as the front-end.

The individual slave sub-system which would normally be located at the remote plant sites (conveyors and dredgers) was configured as shown in Figure 3.4.

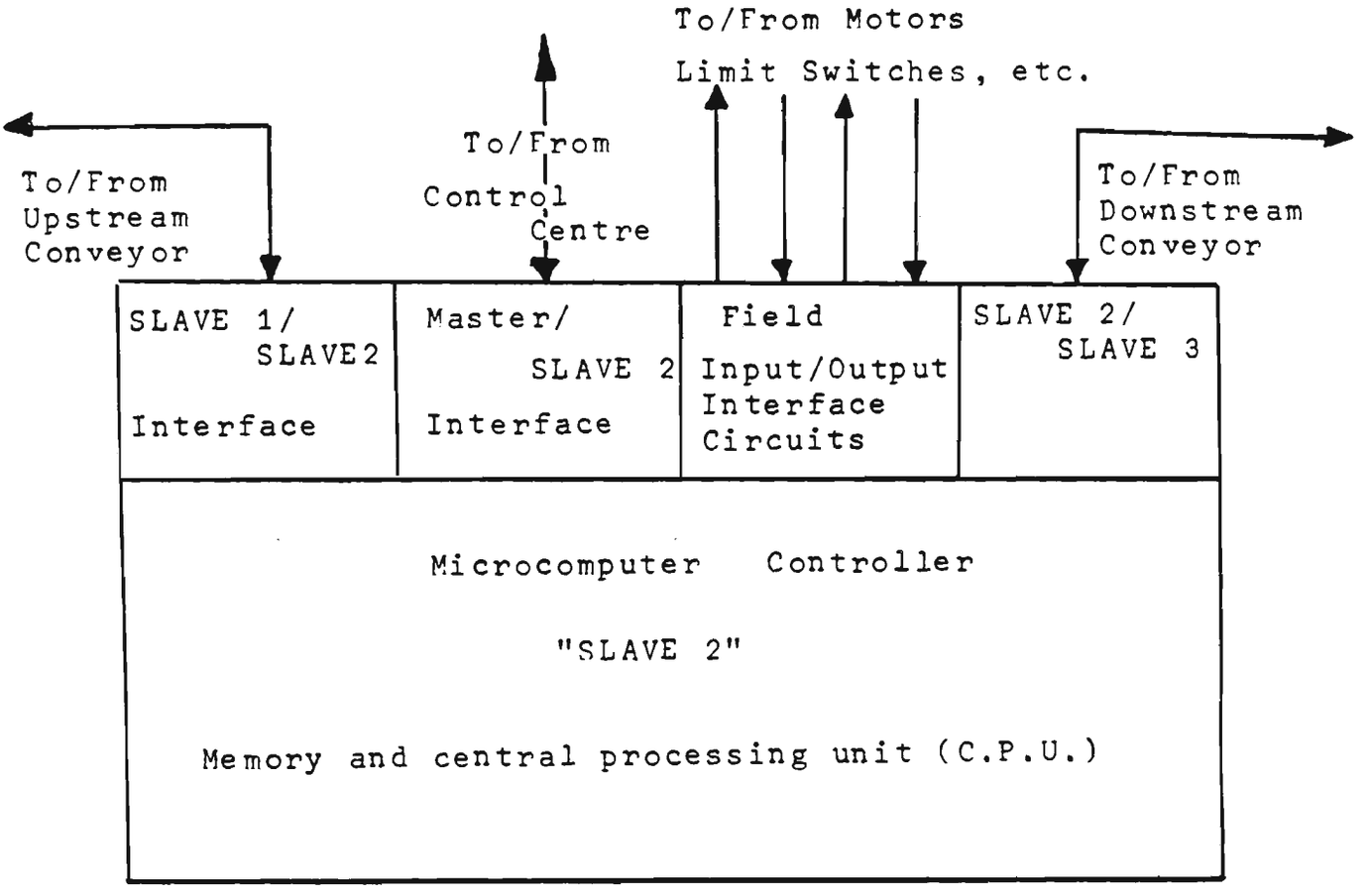


FIGURE 3.4 : INTERFACE REQUIREMENTS FOR A REMOTE SUB-SYSTEM

The slave consisted of a microcomputer-based sub-system with several high speed serial data links for interconnections with the front-end and adjacent slaves. The hierarchy within the sub-system was established with the direct link to the front-end as the highest priority and the alternative paths through the adjacent slave would be rated according to the direction of flow of the material on the conveyor. The priority of the different links was the same order for the slave as was required for the interconnecting links in the front-end (refer to Section 3.2.3). The slave was also rack mounted for ease of change or card replacement. The loss of small elements such as individual slaves in one conveyor route would be more acceptable than the major losses possible with more traditional techniques which could take out more than one conveyor route.

3.2.5 Methods of Communication

In the development of the hierarchical system, consideration was given to the way the sub-systems were connected together. At the hardware interface level, the following factors were taken into account:

- There was a variety of speeds used by the devices and sub-systems to be interfaced which was accommodated by the selectable logic on the interface card;

- The devices used were either current or voltage driven;

- The voltage requirements for voltage driven devices used the industry standards EIA (RS-232C) and CCITT (V24);

- The hardware interfaces had to be compatible with other serial devices and with existing software packages.

Once the hardware interfaces for the model had been designed, it was then necessary to decide if the configuration would be a party line or radial communications network.

3.2.5.1 Party-Line

In a party-line network, at the slave or front-end level, each sub-system would share a common data cable pair. All units would receive the same message but only the one addressed would respond.

3.2.5.2 Radial

In a radial network, each sub-system would have its own dedicated data cable pair radiating from the central sub-system. Sub-system identification is not necessary since the identification is inherent in the layout. The choice of the method to be used usually depends on the polling frequency, response time and the number of cable pairs required. Generally, in an open cut situation the smallest cable used (based on the consideration of its mechanical strength) would have 10 pairs, hence the one cable could support 10 sub-systems.

Following consideration of the cable pairs available and the speed required to identify the sub-system, a radial communication network was chosen for the prototype system.

3.3 INVESTIGATION OF THE DATA TRANSFER TECHNIQUES FOR THE MODEL

3.3.1 Modes of Operation

Most of the traditional modes of operation of communication systems are also applicable in hierarchical microcomputer systems, such as:

- polling (group, individual) of sub-systems by the master;
- interrupt driven communication with the sub-systems;
- continuous communication with the sub-systems;
- priority messages to and from the sub-systems.

Use of the microcomputer enabled a system model to be developed which used basically an interrupt driven and a time initiated polling structure.

There were two methods of operation which were fundamental to the overall system structure; interrogation on interrupt from the sub-systems and time controlled polling of the sub-systems. The polling mode was controlled by the clock (sub-routines were initiated on a time flag) while the interrupt mode was controlled by the hardware and the system hierarchical structure.

3.3.1.1 Interrupt Operation

In a steady state system under normal program execution, the sub-systems would be fairly inactive, stopping and starting motors, etc, but continually updating their record of the previous status of the plant. This mode would reduce the flow of data to and from the master computer by comparing the previous status (i.e. the status that was last transmitted to the master, which was stored in memory) to the current status. If a sub-system does not transmit, the master will assume the same condition exists; for example, if a slave has reported that a motor has stopped it will not report its status again unless it is restarted or the master requests a status check.

3.3.1.2 Polling (or Continuous) Operation

Polling, by definition follows a predetermined sequence in its acquisition of data. Instead of permitting any sub-system to report status changes as they occur, a polling system permits each sub-system to

report only when it is requested to do so. In this way, data transmission is completely controlled by the master.

Polling systems have no way of giving priority to important information. Each critical alarm must wait until it is next scanned before the master can act on the data. This is also one of the major problems of the traditional systems using relay or solid state logic. These earlier systems have no processing capability, hence there is no way of assigning priorities. A combination of the above modes can reduce the effect of these problems. The model allowed:

- faster scanning;
- checking for the presence of altered data;
- preprocessing of information;
- increased message efficiency;
- system response improvements.

3.3.2 The Message Protocol of the Model

Distributed microcomputer sub-systems require a structured message system similar to the earlier telemetry systems outlined in Section 2.1. In data transmission systems, data can be transmitted either synchronously or asynchronously. Synchronous transmission means that the data being transferred is in synchronisation with a timing signal or a strobe pulse. In synchronous systems, the strobe pulse is transmitted with the data. The strobe signal notifies the receiving device that valid data is available.

In serial asynchronous systems (such as the one under study), synchronisation is provided by the active data transmitted or received. The transmission of a particular data pattern consists of a start bit, eight data bits and one or two stop bits; testing for a particular data word "SYN" provides the synchronisation of the data blocks while the start and stop bits provide the synchronisation of each character received.

One of the advantages of a serial system is that it lends itself to transmission over telephone cables. The serial digital data (refer to Section 4.2.4) is converted to a frequency signal by a modem, placed onto a voice-grade cable and converted back to serial digital data by the receiving modem at the other end of the line.

In the model a defined protocol or message format was used to ensure the secure transmission of information.

The information outlined in the Literature Review of Chapter 2 and Appendix F regarding industry standards and the requirements of the Australian Standards Association was used as a guide for the protocol of Figure 3.5 (p.41) as adopted for the research model. The system protocol also included the following standard transmission control characters:

. "SYN" (Synchronous idle) - a signal from which synchronism may be achieved;

. "SOH" (start of heading) - first character of a heading;

. "ETX" (end of text) - terminates a text;

- "NAK" (negative acknowledge) - message not accepted;

- "ENQ" (enquiry) - request for a response;

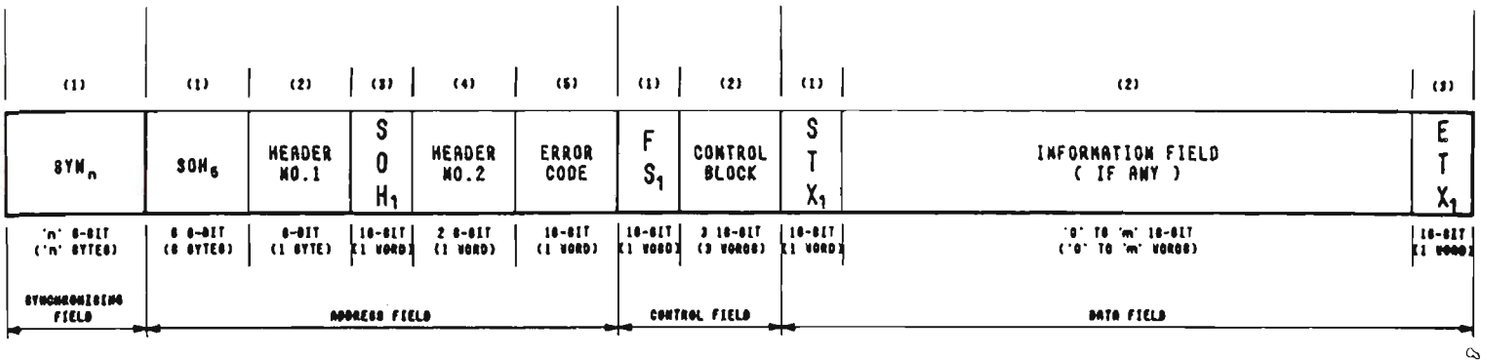
- "EOT" (end of transmission) - indicates the end of the message;

- "ACK" (acknowledge) - an affirmative response to the transmitting sub-system;

- "STX" (start of text) - used to precede text and terminate a heading;

- "FS" (File separator) - used to separate blocks of data.

Secure data transfer between the sub-systems was achieved using the protocol of Figure 3.5, as developed by the author, refer also to Appendix B, (p.153) for an example of the software sub-routine that used the developed message protocol to transfer data between the sub-systems.



MESSAGE FORMAT

NOTES:

SYNCHRONISING FIELD :

(1) SYNCHRONISING CHARACTERS (SYN) , MAXIMUM OF TEN ERROR BYTES BEFORE AN ENQUIRY IS SENT TO THE TRANSMITTING SUB-SYSTEM.

ADDRESS FIELD :

(1) START OF HEADING BLOCK NO.1 (SOH) , SIX CHARACTERS REQUIRED TO ESTABLISH A CORRECT SEQUENCE.

(2) "HEADER NO.1" , DATA LINK CHANNEL ADDRESS FOR THE NEXT SUB-SYSTEM LEVEL

(3) START OF HEADING BLOCK NO.2 (SOH) , A SINGLE CHARACTER IS USED TO SEPARATE THE TWO HEADER BLOCKS.

(4) "HEADER NO.2" , SECTOR NUMBER (MAXIMUM 256) IS THE MOST SIGNIFICANT BYTE. THE SUB-SYSTEM MEMORY IS SUBDIVIDED INTO 256 SECTORS.

WORD NUMBER (MAXIMUM 256) EACH SECTOR IS DIVIDED INTO 256 WORDS.

(5) ERROR CODE , A CHECKSUM CALCULATION OF THE INFORMATION FIELD FOR ERROR CHECKING.

CONTROL FIELD :

(1) FILE SEPARATOR (FS) , USED TO SEPARATE THE CONTROL FIELD FROM THE ADDRESS FIELD.

(2) THE "CONTROL BLOCK" CONSISTS OF THREE WORDS ,

- COMMAND WORD NO.1 FOR THE RECEIVING SUB-SYSTEM (16 BIT)
- COMMAND WORD NO.2 FOR THE NEXT LEVEL (THIRD LEVEL) AND THE FOLLOWING (THE FOURTH). ONE BYTE FOR EACH COMMAND.
- THE RELOCATABLE ADDRESS POINTER FOR THE INFORMATION BLOCK.

DATA FIELD :

(1) START OF TEXT (STX)

(2) "INFORMATION FIELD" , ALLOWS FOR A COMPLETE RELOAD OF THE RECEIVING SUB-SYSTEM MEMORY (UP TO 65,536 WORDS). VARIABLE LENGTH DATA.

(3) END OF TEXT (ETX)

FIGURE 3.5 : PROTOCOL FORMAT ADOPTED FOR THE RESEARCH

In addition to the standard control characters used in the system protocol of Figure 3.5, the standard control characters were also used to acknowledge messages or errors in messages.

The incoming message was loaded into a receive buffer and if at any stage during the transmission of data an error was detected, a standard error code, negative acknowledge "NAK" or enquiry "ENQ", was sent to the transmitting sub-system. It was possible to terminate the data transfer at any stage by transmitting an "EOT" - end of transmission character. If the data was received without an error being detected, the acknowledge signal "ACK" was transmitted by the receiving sub-system.

From Figure 3.5, it can be seen that the protocol developed followed the basic rules for information protocols as outlined in Appendix F. The protocol included a synchronising field, an address field, a control field and a data field. In addition, the system used the standard control characters such as "SYN" synchronise, "SOH" start of heading and "ETX" end of text.

Sections 4.3.3.3 (p.94) and 4.3.4 (p.99) explain how the system protocol and, in particular the control characters, were used to clarify two important areas of investigation:

. The data transfer techniques required for system modelling;

. The operating strategy of the model.

3.3.3 Error Detection in the Model

For error detection within the model, it was decided to develop the hardware error signals (refer to Section 4.2.4.4 on (p.73)) and a checksum sub-routine (refer to sub-routine "CHEKSM" as explained in Section 4.3.3.2, (p.89)) in order to investigate the requirements for data transfer and its security.

Each eight bits of the incoming data was subjected to the hardware error checks, followed by a test for particular 8-bit characters (i.e. "SYN", "SOH", "STX" and "EOT") as outlined in the above section. Once the data transfer was complete, it was then subjected to the checksum test. For an actual system, one of the cyclic redundancy checks (CRC) as outlined in Appendix F3 could be used. Using the more complex CRC code, instead of the checksum approach as used, would only have been an additional software coding exercise. This could be included if there was any further development of the model but was not implemented due to the time constraints.

CHAPTER 4 : THE EXPERIMENTAL MODEL

4.1 DEVELOPMENT OF THE MODEL

The model developed for the research represented the fundamental units required for the investigation of a microcomputer-based conveyor transport control and monitoring system. The configuration selected enabled the study of data transfer techniques within a hierarchical structure and comprised a hypothetical control centre master with several lower order sub-systems. In addition, the feasibility of interconnecting the lower order sub-systems via the alternate data path between the sub-systems was investigated.

In an open cut plant network, the operational and environmental constraints determine the basic parameters (system reliability, linking and construction) of the industrial control system. It was these parameters that were used as part of the criteria for the model. An understanding of the requirements of each one was gained by the author during many years of working with the application of control systems for open cut plant.

An overall plant control and monitoring system should not degrade the security of the coal and overburden removal networks. The configuration selected for the model meets the security requirements since any major failure of a system would be due to damage to the interconnecting cable, see Figure 4.1 (p.45). As can be seen from Figure 4.1 and the conveyor plant layout, Figure 3.1 (p.16) it is not possible to bypass a single plant item (dredger or conveyor) in the middle of a conveyor line. An outage of the conveyor route could be caused by the loss of one of the

individual slave (conveyor) sub-systems or by the loss of the cable along the conveyor structures which form the routes from the dredgers. For example, the L400 to L415 conveyor group (Figure 3.1, (p.16)) has only one cable with up to 24 pairs, damage to the cable or a failure of front-end No 4, as shown in Figures 3.2 and 4.1, would have the same effect.

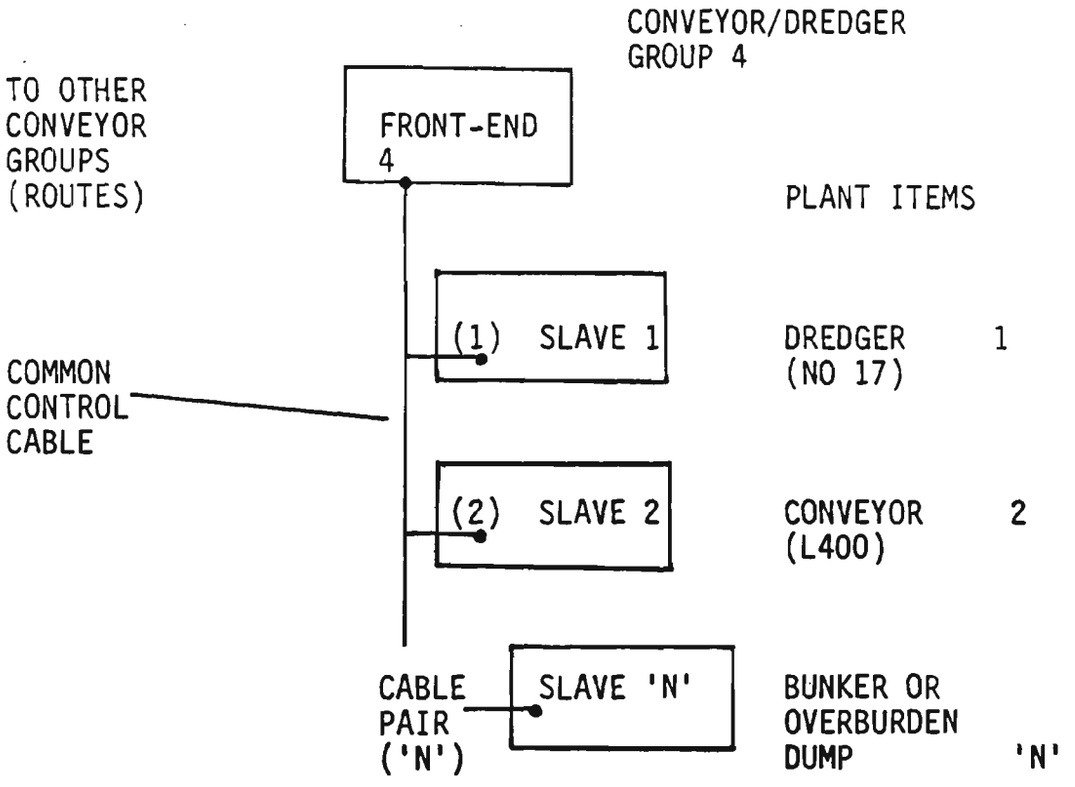


FIGURE 4.1 : COMMON CABLE FOR SUB-SYSTEMS OF THE SAME GROUP

Therefore, the use of a front-end in an open cut control system model, with a small reduction in the overall hardware reliability, was justified by the increased benefit of improved security of the overall control system. The front-end provided the opportunity for limited control at the control centre during any outage of the master, see Section 5.1.1.

Any other configuration would require control at the remote conveyor (slave) locations during failures of the master which would involve long delays while trying to co-ordinate the overall control. For comparison purposes, the hardware and software were also developed to allow for the case where the master controlled the slaves without the front-end. This was achieved by developing the software in the lower sub-system levels to a point where the front-end sub-systems became transparent by re-transmitting any messages received to the next level in the hierarchy (refer to Section 4.3.4) or by direct connections between the master and slaves.

Another configuration, using the same hardware and software in a multiple slave daisy chain arrangement is also possible, but due to the time constraints of the project this configuration was not investigated.

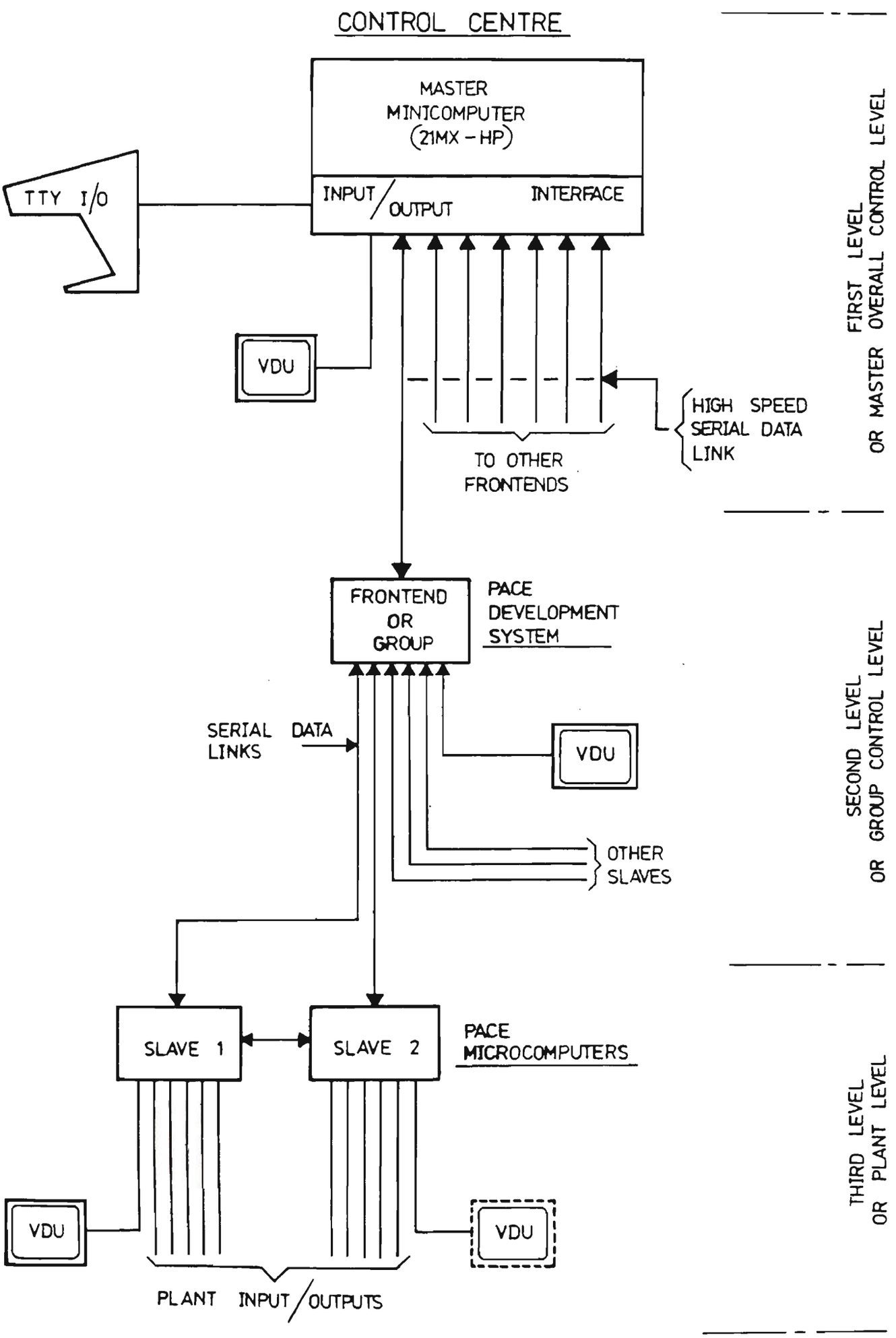


FIGURE 4.2 : SYSTEM CONFIGURATION OF THE EXPERIMENTAL MODEL

The model was therefore developed around the natural constraints of the open cut application and the sub-systems available. The completed model consisted of a single master minicomputer, one group front-end sub-system connected to two slave sub-systems representing two conveyors as shown in Figure 4.2.

4.2 THE HARDWARE REQUIREMENTS OF THE MODEL

4.2.1 Master Computer

The minicomputer selected for the master computer in this project was typical of the type of computer that has been used for this task since the early 1960s and as Hewlett Packard support facilities were already available at the Institute, the HP-21MX computer was selected as the master computer.

The master computer (21MX) was configured with a high speed paper tape reader, teletype, Real-time clock and a graphic display VDU. As part of the project, special high speed serial interfaces were designed for connection to the system VDU and to the other sub-systems (front-end and slaves) as shown in Figure 4.2.

With the 21MX computer, it is possible to service up to 56 distinct interrupts, each of which has a unique priority code associated with a corresponding interrupt location in memory and input/output interface channel. This is designed to suit the different peripherals and high speed data interfaces of the master. Within the hierarchy of the model the channels were selectively enabled or disabled under program control, thus switching the device connected into or out of the defined interrupt structure.

Each device (CRT, VDU, Real-time Clock, tape reader and front-end) was assigned a priority, based on the conveyor plant configuration of Figure 3.1 (p.16) and the expected master system functions. The selection of priorities enabled a systematic sequence for servicing interrupts from the different devices. It was also required that the high-speed devices should not have to wait for the low-speed device transfers.

The plant configuration of Figure 3.1 (p.16) consisting of five dredgers at different coal levels, set up the priorities for the model with the external overburden conveyor line as the highest priority followed by the next level down. For the model, consideration was only given to the priorities between the devices connected. Logically, the Real-time clock had the highest priority followed by the front-end interface, with the VDU unit for display purposes as the lowest priority.

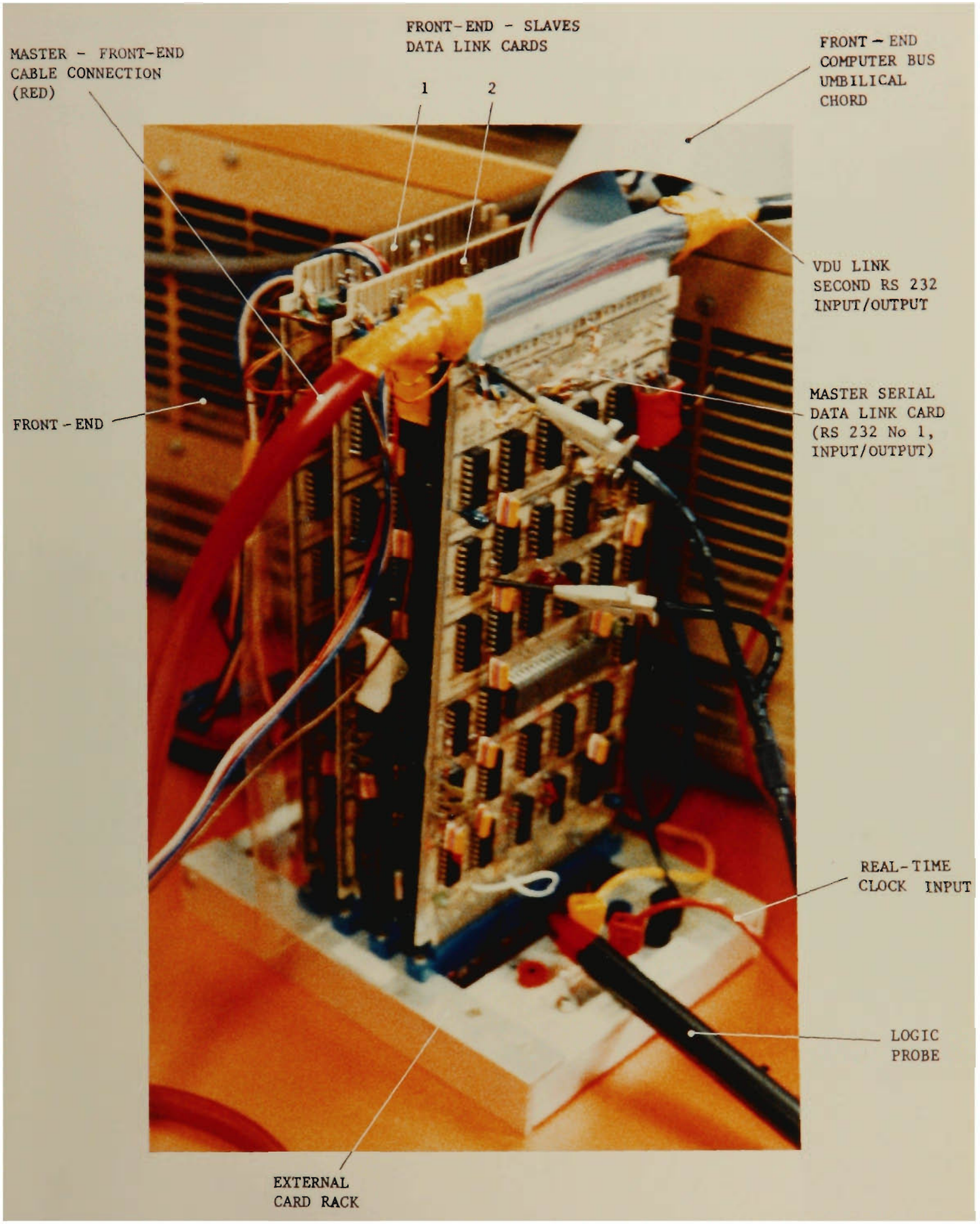
4.2.2 Front-end Computer

The front-end (Photograph No 6 (p.32)) of the model was a National Semiconductor Full Scale PACE microcomputer development system. The PACE development system consisted of 12k RAM memory, a high speed paper tape reader, a teletype, VDU, dual floppy discs. Several special high speed serial data links for communication with the slave and the master computers were developed as part of the project. The development system enabled each microcomputer-based sub-system (front-end and slaves) to be developed from the basic cards and components available.

The locally supported PACE microcomputer was chosen as the most suitable device for the task in order to avoid delays and problems with replacement hardware. In addition, the facilities available in the Electrical Engineering Department at the Footscray Institute of Technology provided supplementary support to a PACE microcomputer-based system study. All of the basic principles applied to the PACE microcomputer during the research can be applied to other microprocessors.

In addition to the standard cards, use was made of the prototyping facility of the development system (front-end) in order to develop the special high speed data links of the slaves and front-end and to enable the interconnection of the sub-systems to the front-end as shown by Photograph No 3.

Handling multiple tasks is an important function for computer systems and languages. The sub-systems of the model had to be able to handle several concurrent tasks in their combined local control and data transmission



PHOTOGRAPH NO 3 - SYSTEM INTERCONNECTION ARRANGEMENT

modes. To achieve the hierarchy for the front-end and the slaves, use was made of the 6-level priority interrupt structure of the PACE microprocessor shown in Figure 4.3. (p.53).

Each level was provided with a software driven enable "(IE2-IE5)" plus the overall enable "IEN". When an interrupt request (NIR1-NIR5) occurred, the associated request latch (IR1 to IR5) was set, if "IEN" was true then an interrupt was generated and recognised. The interrupt pointers or addresses of the various interrupt service sub-routines were stored in the memory (refer to Section 4.3.3.3 (p.91), regarding stack handling for more details).

Each of the interrupt levels was assigned a task in the system hierarchy. The highest priority interrupts were assigned housekeeping tasks such as stack or multiple sub-routine handling and Real-time clock servicing. The remaining interrupts were used to form the external hierarchical structure. See Section 4.3.3.3, on interrupt handling for further details. For systems that require expanded user interrupts, the technique outlined in the PACE user's manual (31) could be adopted, but for the model, an expanded interrupt capability was not required.

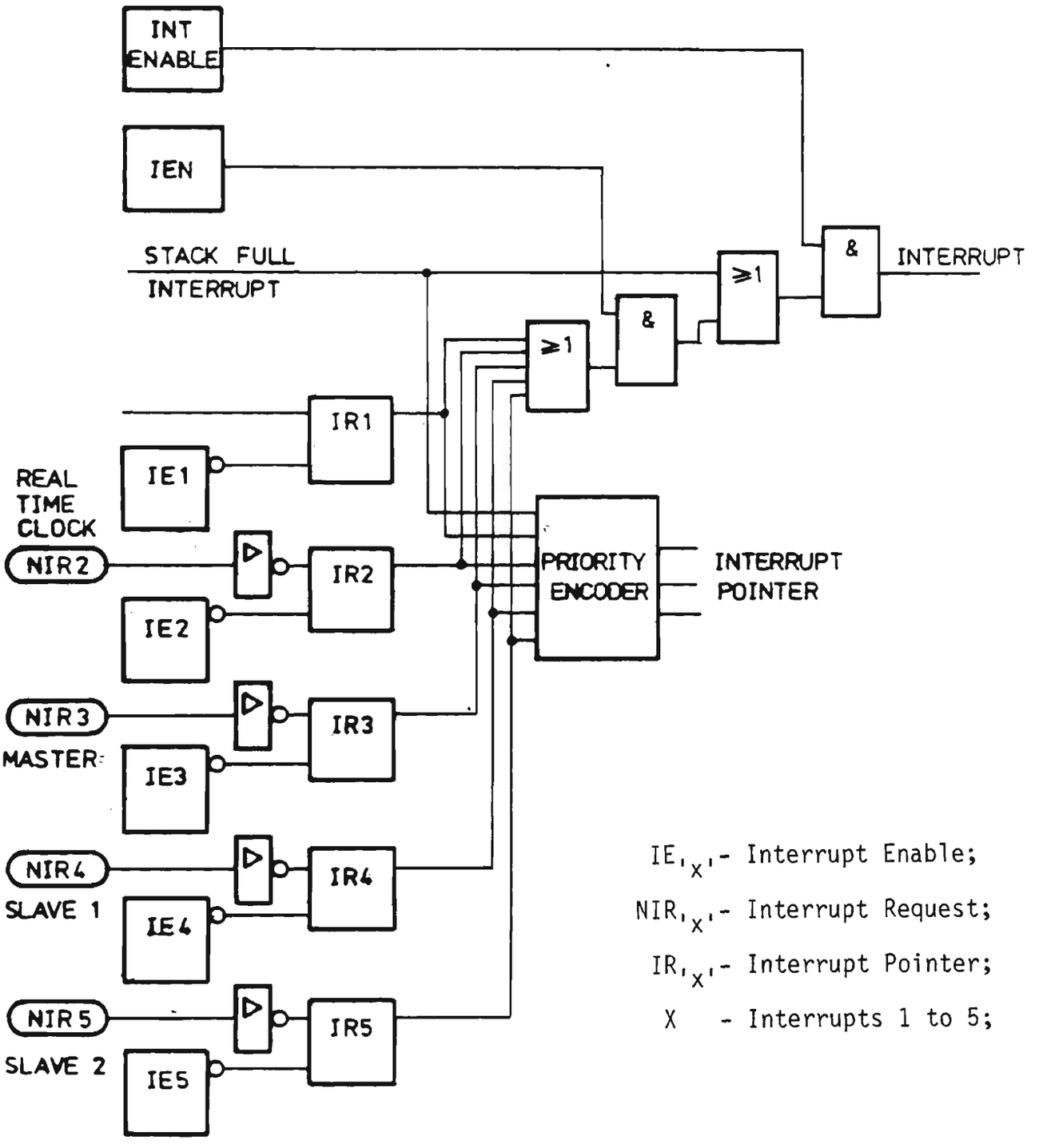


FIGURE 4.3 : PACE INTERRUPT SYSTEM AS IMPLEMENTED IN THE FRONT-END

4.2.3 Slave Computer

The hardware requirements of the slave computers were not as involved as those of either the master or the front-end. The slaves for the model consisted of the standard general purpose National Microcomputer cards (CPU, ROM and RAM). In addition a Real-time clock card, input/output panel and interface control card and a high speed serial link card was designed for the slave.

. The Central Processing Unit (CPU) card contained the microprocessor, a crystal controlled oscillator and data buffers. The flag and control sections provided the external signals for the connection to the other cards. Data transfers between the CPU and memory or peripheral devices were carried out over the 16 bit parallel input/output data bus.

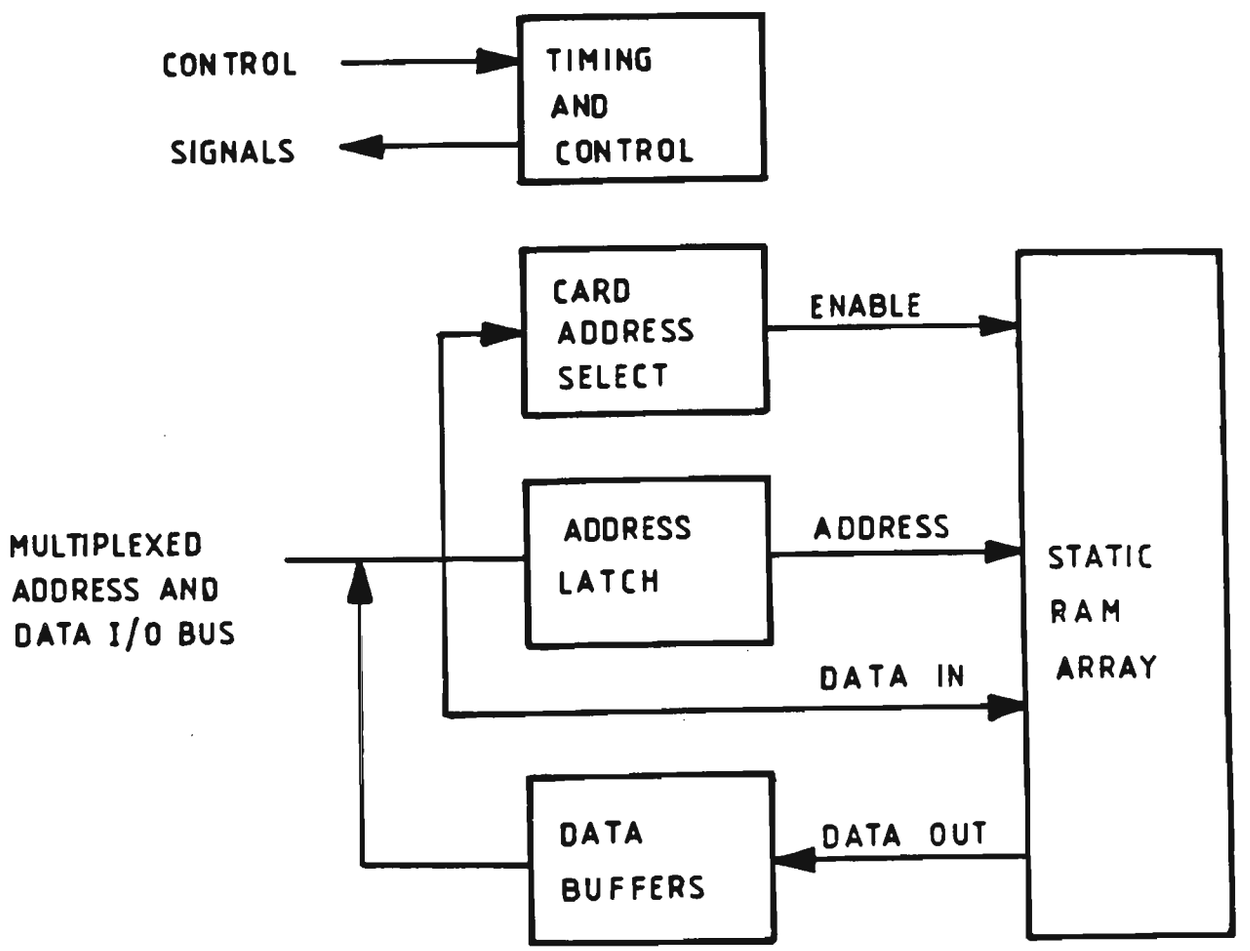


FIGURE 4.4 : RAM BLOCK DIAGRAM

. The National Semiconductor Memory card, as shown by the block diagram of Figure 4.4, consisted of the RAM or ROM memory chips - depending on the type of memory required. The address was latched to select the memory location for data transfer.

. The Real-time clock card was another important area of the real-time control situation under investigation. The Real-time clock card was specially customised to suit the timing requirements of the software programs and provided a predetermined pulse for the Real-time clock sub-routine. All events at the slave sub-systems were related to the time of occurrence and were therefore referenced to the actual time. In addition, the scheduling of the software sub-routines for execution was controlled by the Real-time clock pulses. The time period was fixed at the smallest value required (100 milliseconds), while larger periods were determined by the software sub-routines.

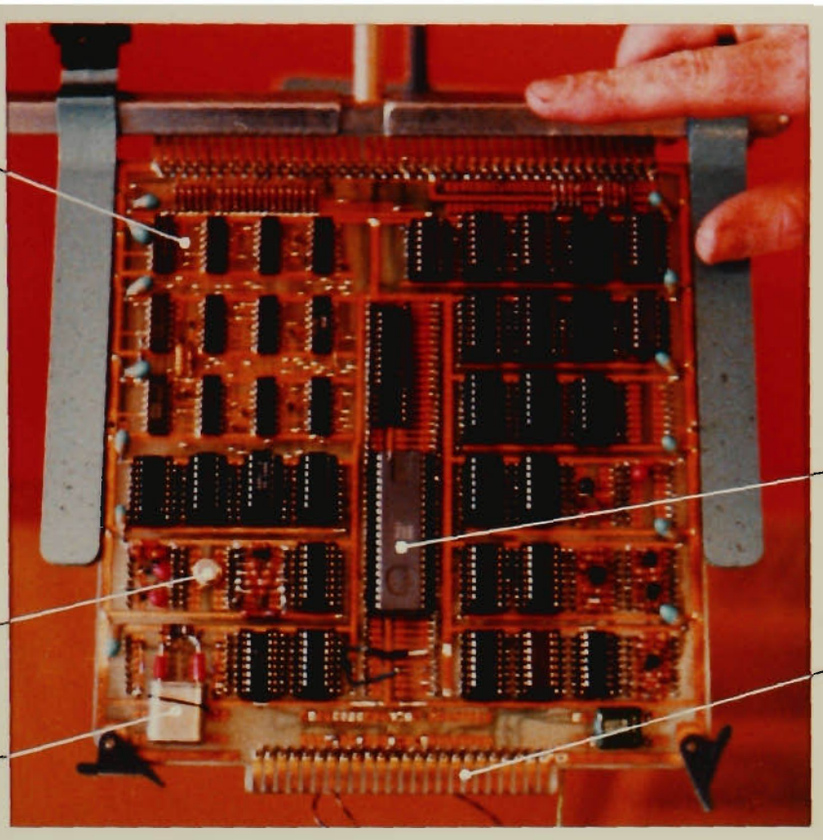
4.2.4 High Speed Data Interfaces

4.2.4.1 General

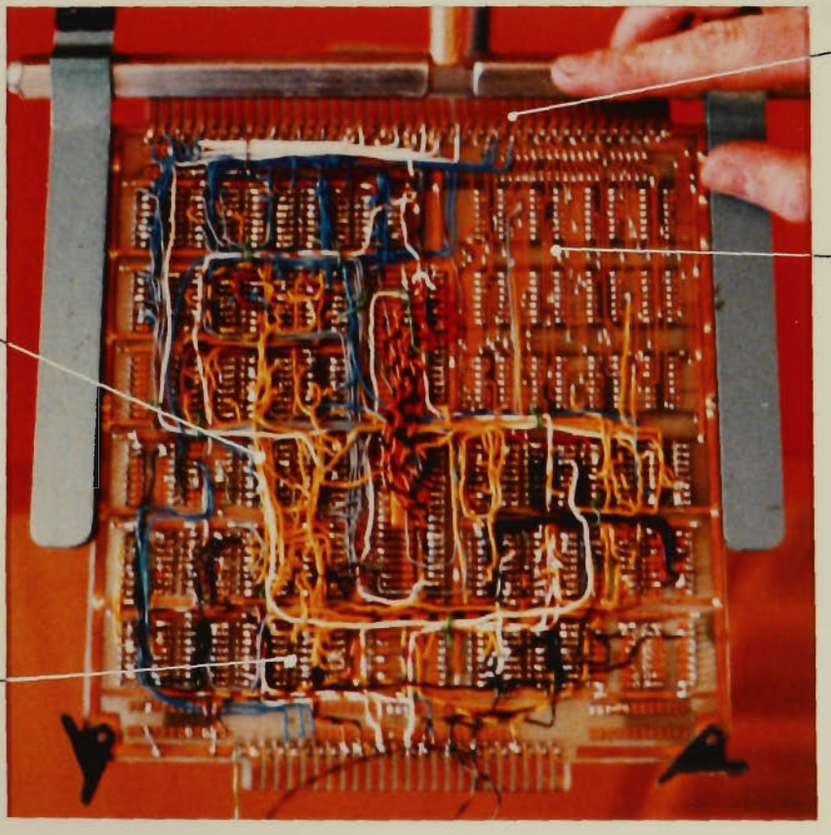
In addition to the standard computer cards and input/output devices used and adapted, there was a need for a high speed interface card, as shown in Photographs Nos 8 and 10, in order to link the sub-systems (master, front-end and slaves) of the model together and for interfacing the teletype and VDU to the computers. At the time of the research, the suppliers of the hardware had not developed a suitable application card which met all of the requirements of Section 3.2.1. The high speed data links were an essential part of the model, hence it was necessary to design and produce several cards for the model.

The hardware was developed using prototyping cards consisting of dual in-line integrated circuit sockets with wire wrap pins or soldered connections, as shown by the prototypes in Photographs Nos 8, 9 and 10. The appropriate cards were tested by using either the PACE development system or the Hewlett Packard card tester.

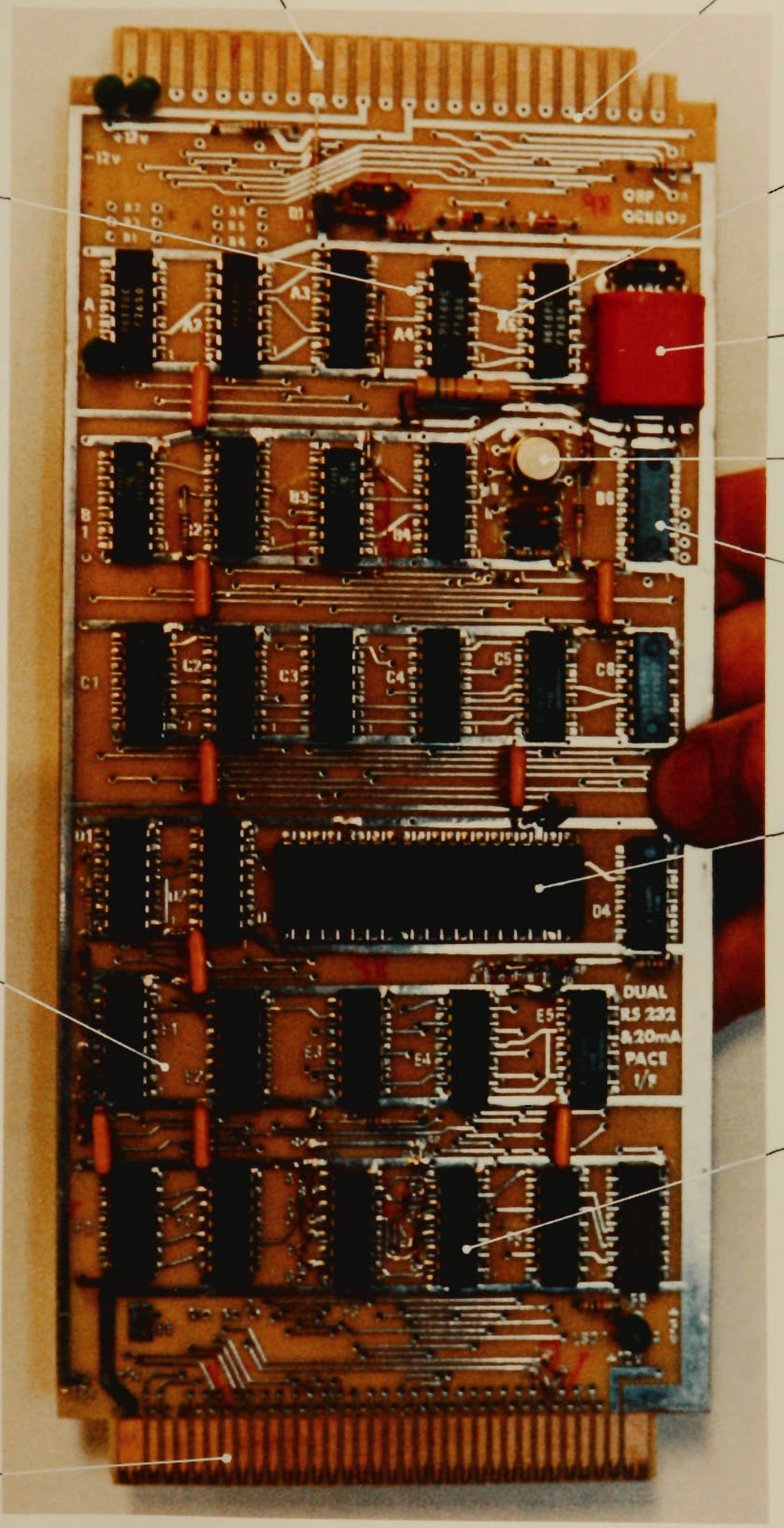
The following Sections (4.2.4.2 and 4.2.4.3) describe in block diagram form how the logic functions on the cards were implemented. As with most computers, the 21MX-HP minicomputer and the PACE microcomputer processor cards did not directly drive the peripheral devices connected, therefore it was necessary to analyse the backplane signals as detailed in the handbooks with the aid of a camera mounted on a high speed cathode ray oscilloscope in order to design the interface cards required.



PHOTOGRAPH NO 8 - MASTER COMPUTER
COMPONENT SIDE
HIGH SPEED DATA LINK INTERFACE



PHOTOGRAPH NO 9 - MASTER COMPUTER
SOLDER SIDE
HIGH SPEED DATA LINK INTERFACE



PHOTOGRAPH NO 10 - FRONTEND AND SLAVE
DUAL RS232 AND 20 MILLIAMP INTERFACE
- HIGH SPEED SERIAL DATA LINK

4.2.4.2 The Basic Building Blocks of the High Speed Interfaces

The basic function of the high speed interface cards was to convert the parallel data to serial data and vice versa at a predetermined and flexible speed. In order to meet the requirements of the model, two basic building blocks (a Universal Asychronous Receive and Transmit (UART) chip and a Programmable Baud Rate Chip) were used. These two devices were selected in preference to discrete shift registers in order to minimise the number of components and to standardise the cards as much as possible. The features of the two basic blocks were as follows:

. The UART: This device provided the central processing unit of each sub-system with more time to carry out other tasks while it transferred the data to the peripherals. The peripherals used were 8-bit devices and were within the speed range of the 8-bit UART. Use was made of the inbuilt flags of the UART to interrupt the computer after the data had been received or transmitted in order to transfer the data for processing.

  Although the 21MX minicomputer and the PACE microcomputers were able to process 16-bit data, the variable speed and asychronous features of the UART outweighed the disadvantage of processing 8-bit data. In both the 21MX minicomputer and PACE Microcomputer, the least significant 8 bits of the 16-bit data bus were used for the data transfers via the UART interface cards.

Although the design and development of the data link cards took time, it was highly desirable to use a standard interface device such as the UART because of the substantial time and effort that would be required for the development of a similar device using discrete components;

The Programmable Baud Rate Chip: Another important requirement of the high speed interface card was to be able to vary the speed at which data was transferred within the hierarchy of the model without major changes to the hardware. To achieve the variable output speed required the card edge connectors of the peripheral device were wired with the code that selected the speed of the programmable chip (as shown in Figure 4.5). This speed was then used to determine the baud (bits/s) rate at which the UART received or transmitted the data.

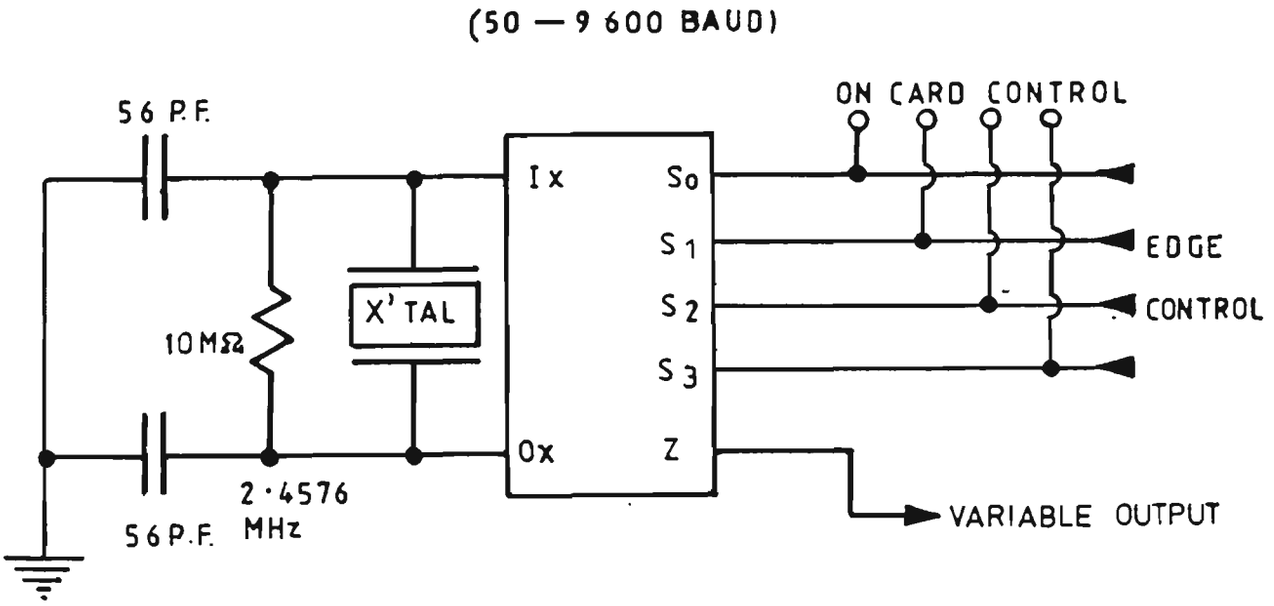
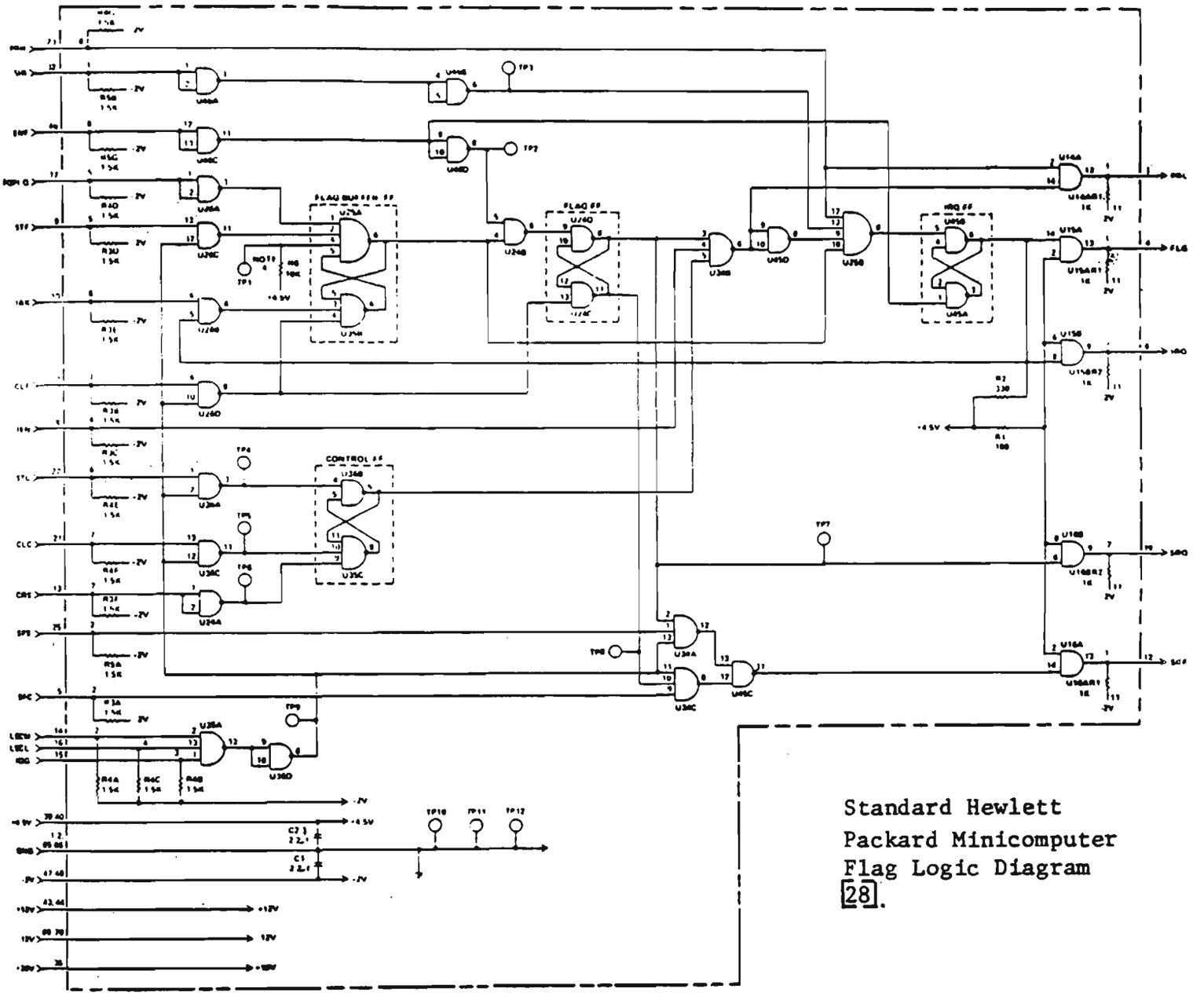


FIGURE 4.5 : PROGRAMMABLE BAUD RATE LOGIC

4.2.4.3 The High Speed Data Link of the Master

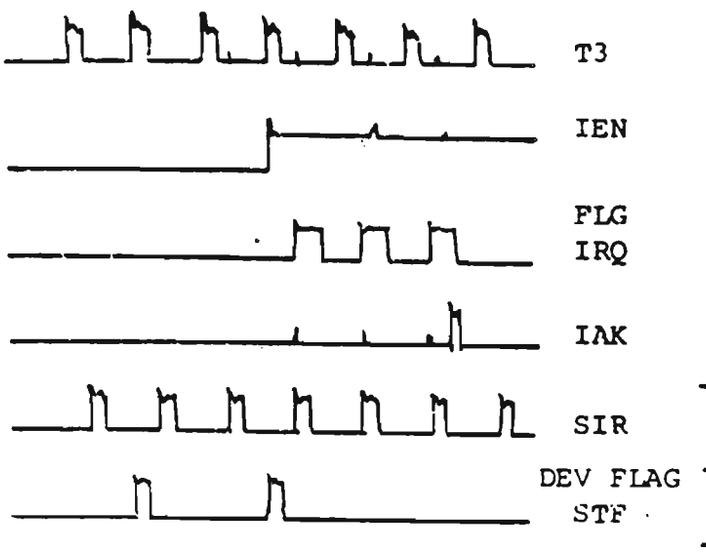
The card developed for the master was the one shown in Photographs Nos 8 and 9 and the circuit details in Appendix C. This card used the basic blocks, the UART and the programmable baud rate chip which enabled the data to be transferred at selectable baud rates of 50 to 9600 Baud (bits/s) depending on the device connected. As shown in Figure 4.6 and Photograph No 8, the prototype card for the 21MX had a standard flag logic section which provided standard signals on the card for data control.



Standard Hewlett Packard Minicomputer Flag Logic Diagram [28].

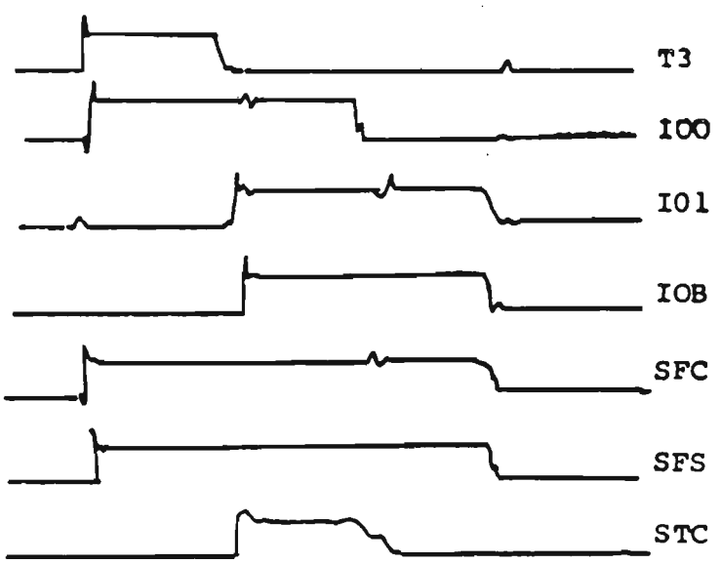
FIGURE 4.6 : FLAG BLOCK

It was necessary to include this standard flag logic as part of the logic for the 21MX interface card in order to make use of the computer backplane signals. During the execution of the different computer instructions, the following flag and input/output signals were observed and photographed.



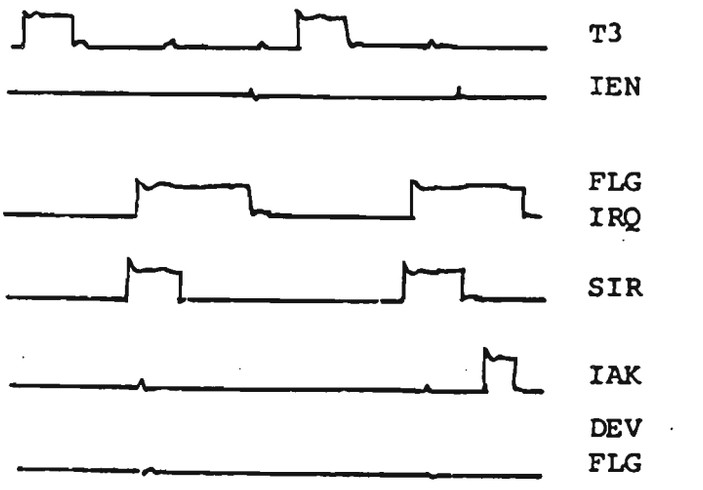
10 V/Div Vert. 2×10^{-6} Sec Hor/10mm

FIGURE 4.7 : INT. REQUEST



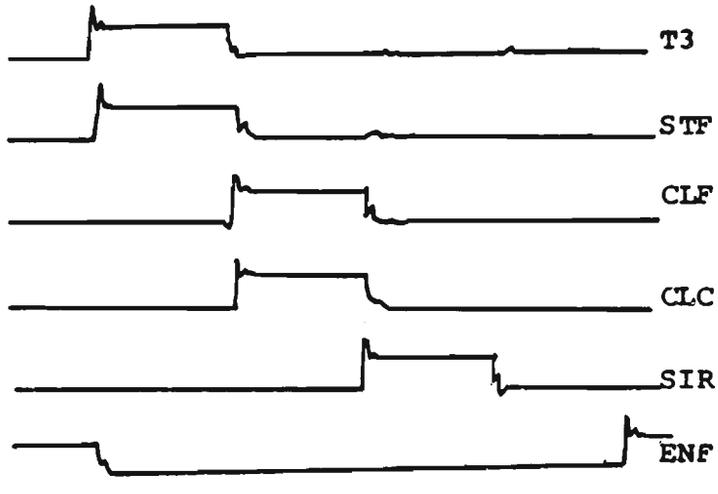
10 V/Div Vert. 0.2×10^{-6} sec Hor/10mm.

FIGURE 4.8 : INPUT/OUTPUT SIGNALS



10 V/Div Vert. 5×10^{-7} SEC. Hor/10mm

FIGURE 4.9 : INT. REQUEST



10 V/Div Vert. 2×10^{-6} SEC. Hor/10mm.

FIGURE 4.10 : SET AND CLEAR FLAGS

The following paragraphs show how the computer backplane signals were combined with the UART flags to implement the basic data transfer requirements for transmitting and receiving:

Transmit: This was achieved by loading the eight data bits (Bits 0-7) into a buffer on the card before being strobed into the UART chip. The UART then converted the data from parallel to serial data for transmission to the connected peripheral or sub-system. Depending on the type of device to be connected (RS232C industry standard voltage or 20 milliamp current, device) different pins on the output side of the card were connected. The "Transmit End of Character" (TEOC) flag of the UART was used to interrupt the computer when the card was ready for another eight bits.

Once the appropriate interrupt signal was given by the UART to the computer, the software sub-routine selected via the interrupt memory location determined the next step to be taken. This sub-routine, which is explained in the software Sections 4.3.2 and 4.3.3.3, used a combination of the "set device" control bit signal "STC" (see Figure 4.8) and the "output the register to device instruction" (OTA or OTB) which generated the data output signal "IOO" shown in Figure 4.8 on the computer backplane (refer also to the detailed circuit diagram in Appendix C and the block diagram of Figure 4.11). This signal was used to load the card control codes (for transmit), select codes (for different devices) or for loading data words into the data buffer. The "STC" signal was used to generate a transmit strobe pulse of a predetermined period to suit the UART timing specification. The time duration was determined by the logic within the control logic block shown in Figure 4.11.

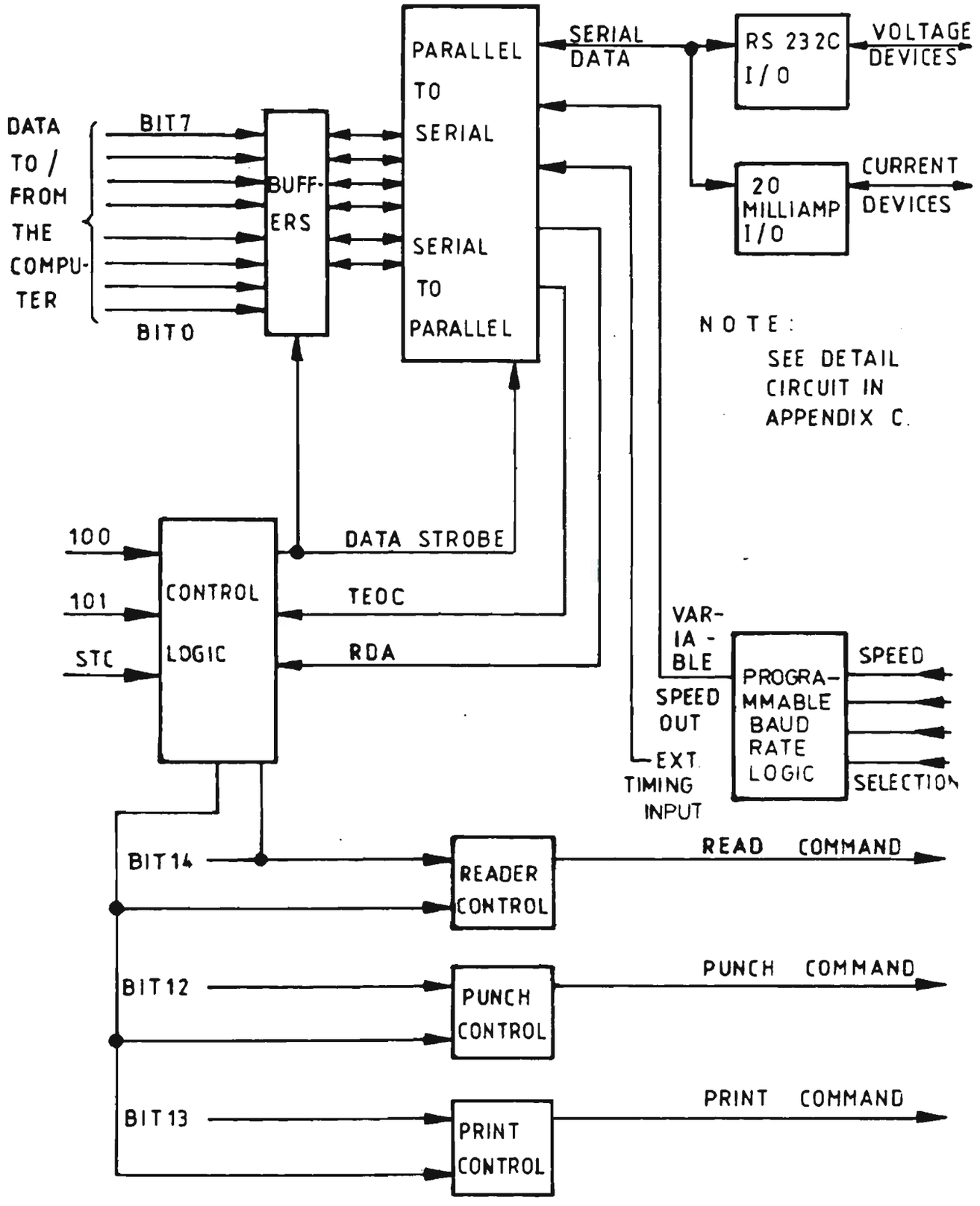


FIGURE 4.11 : SIMPLIFIED BLOCK DIAGRAM FOR THE SERIAL DATA INTERFACE CARD

The parallel data input to the UART was then converted to serial data and transmitted at a rate (50-9600 baud) determined by the programmable baud rate chip to the connected device (RS232C or 20 milliamp loop);

. Receive: The basic input concepts of the interface are as shown in Figure 4.12A and make use of another UART flag, "Received Data Available" (RDA), to signal that the data had been received.

The operational amplifier was used as a Schmitt trigger to remove any noise associated with the input. The Schmitt trigger arrangement, as shown in block form, functioned as a one-bit analogue to digital converter. The device was arranged so that it had two output states which were functions of the amplitude of the input excitation (Figure 4.12B). In addition to removing noise, the circuit converted 20 milliamp signals to TTL logic levels for the UART chip. The UART chip also had some noise immunity; it sampled the data every half data bit, testing for a low signal which signified the start of the data word.

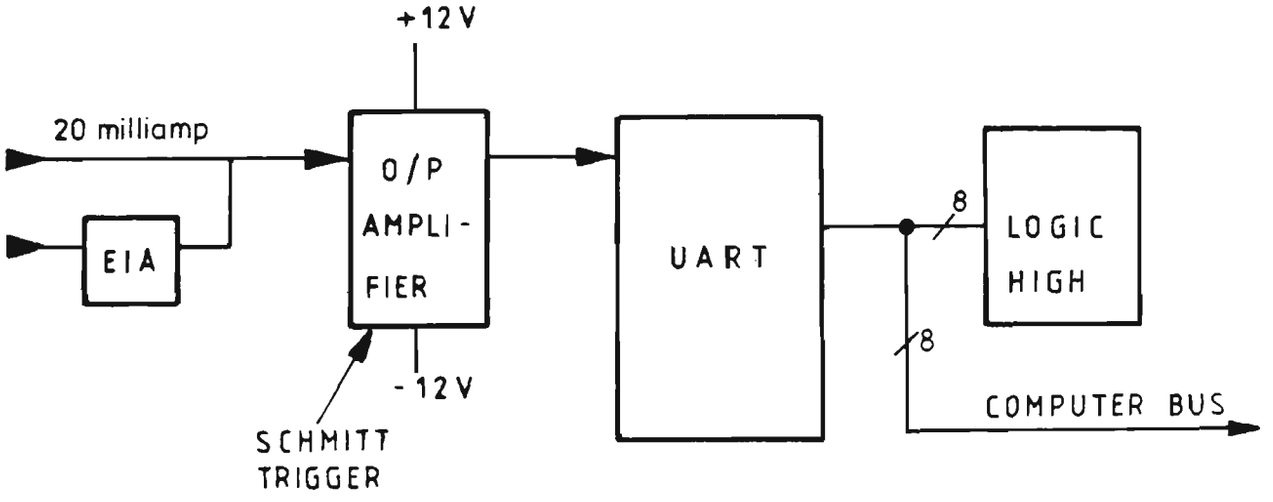


FIGURE 4.12A : BASIC INPUT BLOCK DIAGRAM

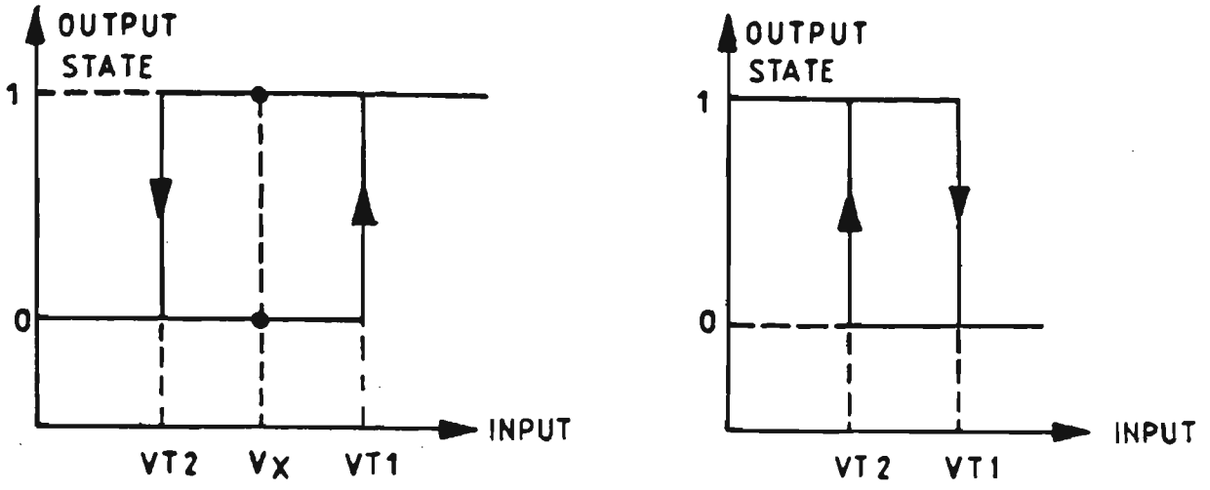


FIGURE 4.12B : THE INPUT/OUTPUT RELATIONSHIP OF THE SCHMITT TRIGGER

Switching between the two output states of the Schmitt trigger was very fast because it utilised the internal regenerative feedback set-up in the operational amplifier. The input/output relationship is illustrated in Figure 4.12B. When the input voltage exceeded the threshold voltage (V_{T1}), the output returned to the '0' state (29). As indicated, the device exhibits hysteresis in the region between V_{T1} and V_{T2}. Note that if the input rises to or falls to V_x, the output will remain in its last logic state of either '1' or '0'. Hence, the arrangement was used as a threshold filter and was used to remove the contact bounce noise such as occurred on the signal from a teletype. See Figure 4.13 for a diagrammatic representation of the observed signals.

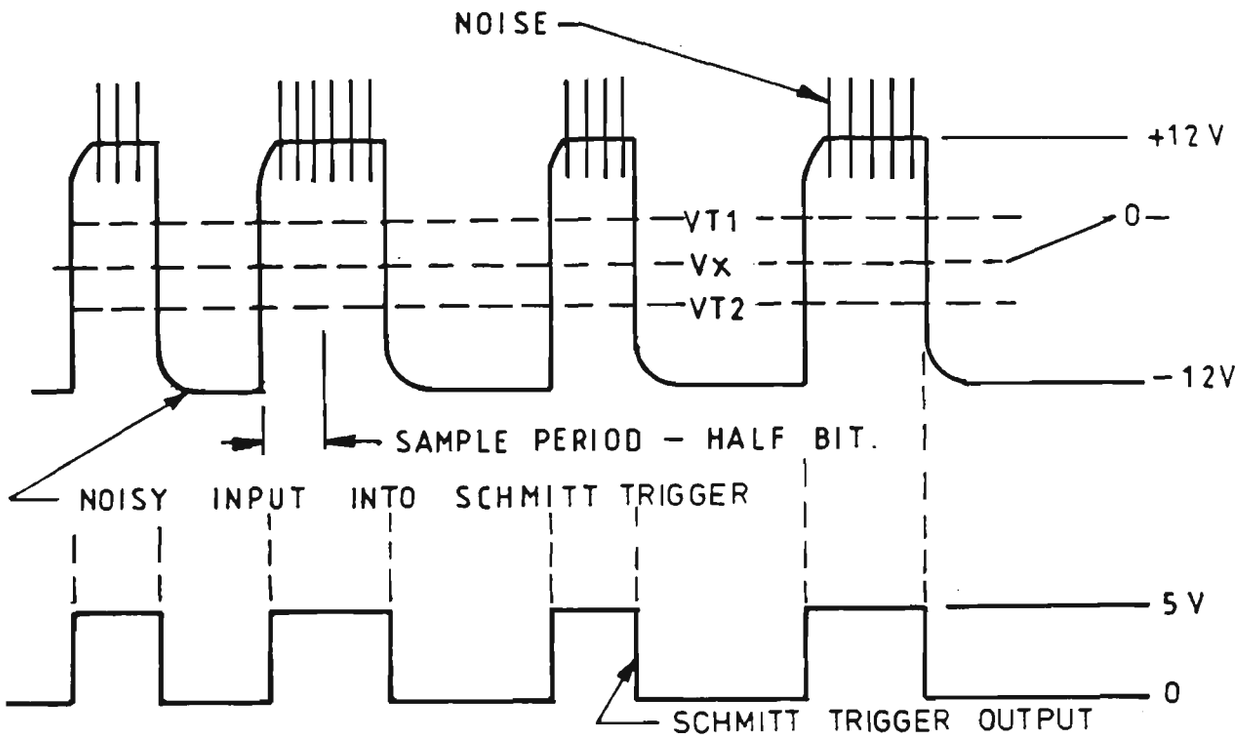


FIGURE 4.13 : SKETCH OF PHOTOGRAPHED SIGNALS

In addition to the above noise which can cause incorrect sampling by the UART chip due to spikes occurring a half-bit apart, it is also possible to cause a stalemate situation when reading from a teletype. This condition could occur if the interface card is given the command to read and the reader moves the tape one character space but the UART rejects the character as noise. If a noise spike occurs at the sample period as shown in Figure 4.13, the UART would wait for the next character and the computer would not give the next read command until the previous read command had been carried out.

To overcome this problem, a 16-bit counter was introduced into the logic (see Figure 4.14). (33) If a character was correctly received, the "Received Data Available" flag was reset and cleared the counter after 11 bits (eight bit data, one start and two stop bits) but if the UART rejected the start of the character the counter caused the reader to move on after a delay of five bits, avoiding the stalemate.

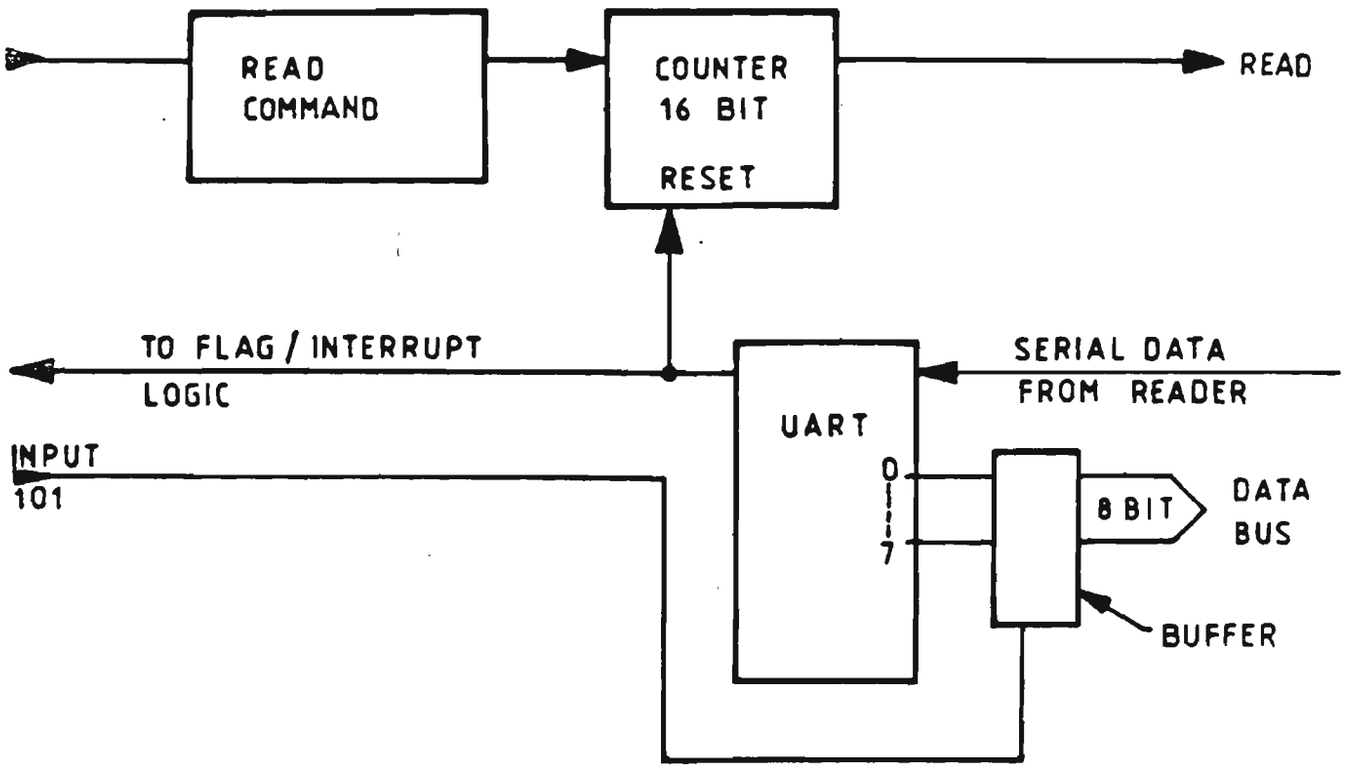


FIGURE 4.14 : BLOCK DIAGRAM OF READER CONTROL

The software can also use the inherent error checks of the UART because the framing error, receiver over-run and parity error flags, refer to Figure 4.17 (p.73), of the device, had been connected to bits 12 to 14 of the data bus. When the data is read into the computer by executing the "LIA" computer instruction, which generated the "IOI" signal (see Figure 4.8), the status of the UART error flags were also connected to the computer bus during the transfer of data to bits 0 to 7.

4.2.4.4 The High Speed Data Link of the Front-end and the Slaves (Photograph No 10)

The high speed data link card for the microcomputers of the model was developed as a dual RS232C (voltage) interface card.

The logic designs of the microcomputer cards were developed using the same basic building blocks (the UART and programmable baud rate chip) as the master computer.

The fundamental difference between the two computers (the minicomputer and the microcomputer) was that the PACE microcomputer did not have a select code or special input/output instruction set. The microcomputer peripheral interface cards were effectively a location in memory as far as its central processing unit (CPU) was concerned. Therefore, the interface card was developed using an address decoder. If the address matched the predetermined card address on the card (see circuit details in Appendix C and Photographs Nos 3 and 10) then the computer 16-bit address word was accepted by the card logic as a card select and function select code word. The first six bits provided the appropriate select code for the operation to be performed by the executed sub-routine. As shown in Table 4.1, the possible commands were determined by the bit pattern of the card address.

Bits	Function
5	RS232C 1 or 2 (Link (1) or (2))
4	Ready to transmit (RTS)
3	Ready to receive (RTR)
0-2	Data process code

TABLE 4.1 : DATA WORD FORMAT

The function select code for the 0-2 bits enabled eight different data process functions. These are listed in Table 4.2:

Binary	Function
000	print or transmit
001	tape read
010	power on
011	power off
100	echo
101	enable data
110	enable status
111	not used

TABLE 4.2 : FUNCTION SELECT CODES

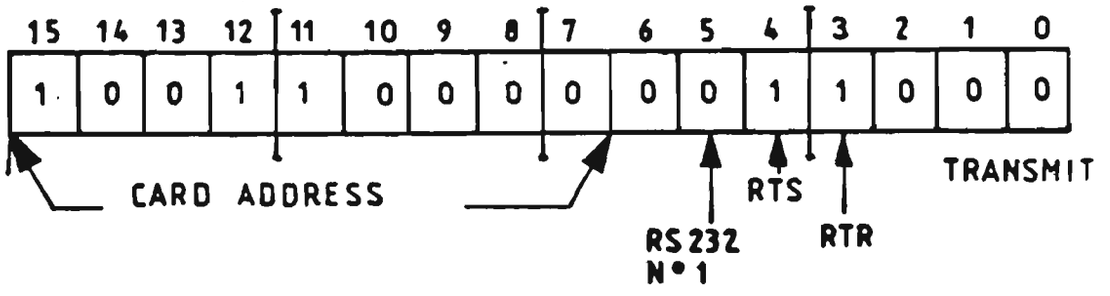


FIGURE 4.15 : CONTROL WORD : PACE

Figure 4.15 represents the control word necessary to select the serial link(RS232C No 1) for data transmission.

. Transmit Data: As previously discussed in Section 4.2.4.2, the front-end and slave serial data link interface cards used the UART chip. The UART converts the 8-bit parallel data to an acceptable serial form for transmission either as a 20 milliamp current signal or as an industry standard voltage signal (RS232C). The UART was used again to minimise the number of components on the cards and enabled the transmit function required.

To achieve the functions outlined in Table 4.2, the basic design adopted is as shown in Figure 4.16. In order to design the data link card logic, the microcomputer processor card signals were also studied and combined with the UART flags as outlined in Section 4.2.4.1 (p.56).

When the correct card address was received the output data strobe (ODS) signal of the CPU latched the data on the data bus into the UART.

The next data word was inhibited in the software either by waiting for a flag (JC13, etc) or an interrupt (NIR 2-5). Refer to Figure 4.3, (p.53), for the significance of the interrupt signals, i.e. NIR2 is an interrupt from the Real-time clock. Once the UART had completed the data transfer a signal (JC13 or NIR 2-5) was given to the central processing unit to continue with the next word to be transmitted.

The tape read function was also implemented in the same way as explained for the master interface (p.67) to avoid a halt in the program execution due to noise rejection.

. Receive Data: The data input circuit is also similar to the data input circuit on the master high speed interface card (p.65). For the dual RS232C input, an appropriate input code had been selected and defined in Table 4.1 to enable switching from one input to the other. In the case of the slaves the normal data transfer path is for data to be received from the front-end computer; it was also possible to switch to the alternative or indirect route via the second RS232C input/output chips (see Figure 4.16) which was connected to the adjacent slave sub-system, refer to Section 4.2.5 for more details. Data was received by the UART in a serial string of eight bits (0-7) which then generated a "Received Data Available" (RDA) flag, which in turn caused an interrupt

or flag on the computer backplane. An input sub-routine (MSTINP (p.162)) which included an instruction that provided the "Input Data Strobe" signal (IDS), was then executed. This enabled the UART high impedance outputs and the TTL high impedance bus drivers (UART outputs can drive one TTL gate only). Once enabled, the 8-bit data (incoming) was allowed to appear on the data bus and hence was read by the processor.

The UART flags were also cleared by the IDS signal ready for the next data word.

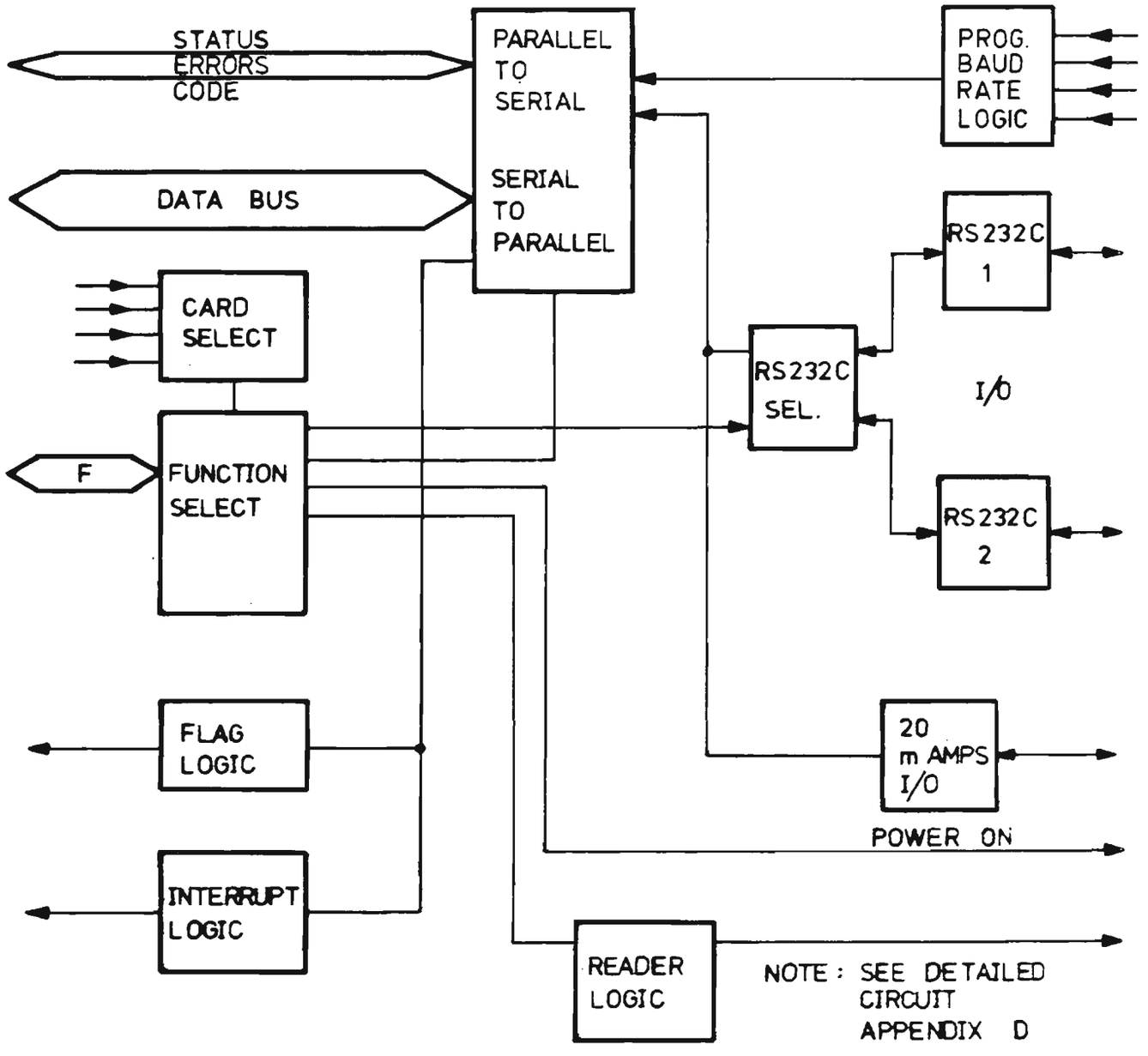


FIGURE 4.16 : A SIMPLIFIED BLOCK DIAGRAM FOR THE PACE DATA LINK CARD

Where high security techniques were required, it was possible to test for "Receiver Over-run" (ROR), "Received data Parity Errors" (RPE) and "Received data Framing Errors" (RFE) by using the UART hardware signals for ROR, RPE and RFE in a similar way as explained for the master interface card (see Appendix C for circuit details). The hardware signals were used in the error detection sub-routine "CHECKSM" in addition to the error code techniques as explained in Section 3.3.3. Before loading the next data word, a status request was made; the data on the computer bus was then tested for any ROR, RPE and RFE errors before continuing to load data. See Figure 4.17 below.

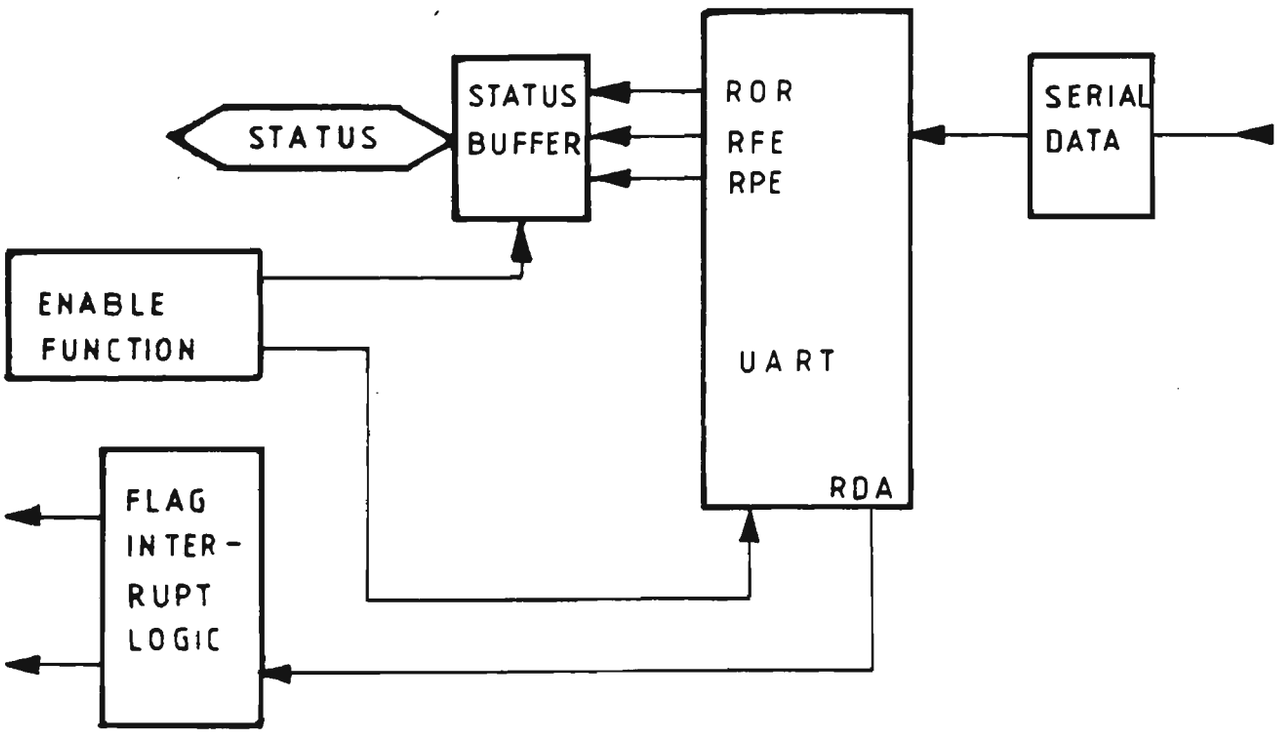


FIGURE 4.17 : ERROR STATUS BLOCK DIAGRAM

4.2.5 The Data Transfer Paths

In the distributed system model, as shown in Figure 4.18, the direct data path was to or from the slave through the front-end to the master.

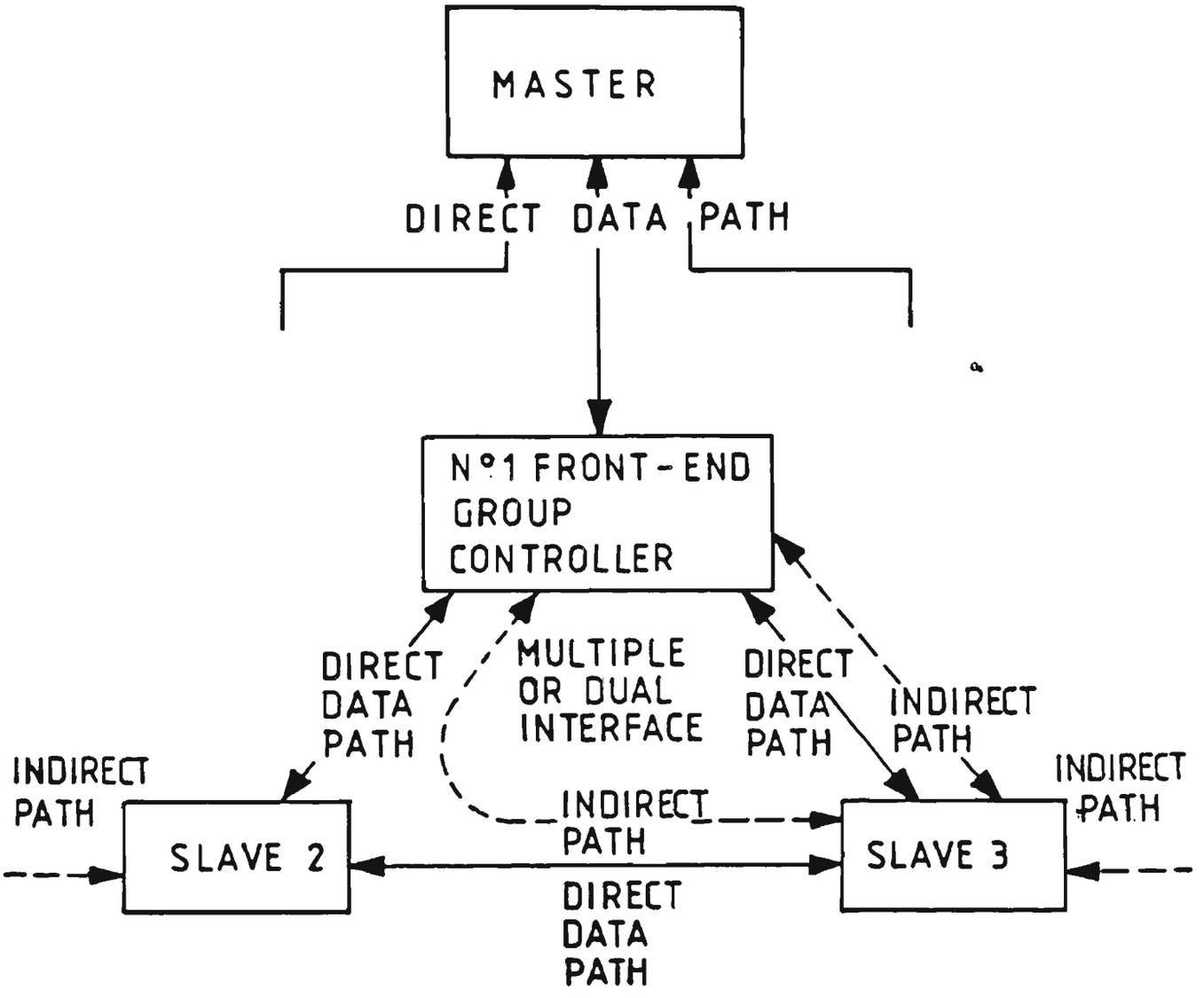


FIGURE 4.18 : INDIRECT/DIRECT DATA ROUTES OF THE MODEL

In order to provide a back-up to the direct link to the master the second or indirect link on the card was used. As shown in Figure 4.18, an indirect path was possible via the second set of RS232C inputs and outputs on the data link card. The computer used a second memory location in order to select the indirect data path. This route was also used for linking the sub-systems together; messages such as sequence start, which are required as a direct link between the conveyors, were passed via this link.

4.3 THE SYSTEM SOFTWARE FOR THE MODEL

4.3.1 General

In order to explain how the software was developed and the problems encountered in achieving the objectives of the research, a brief outline of the two computers is included.

. The 21MX Hewlett Packard Minicomputer: The computer has eight 16 bit working registers which can be selected for display and modification through the operator's panel or directly by the software sub-routines. The main registers are the A and B registers which can be directly addressed by the memory reference instructions. The memory is divided into fixed pages of 1024 words each. It is possible to directly address page zero (the current page) and the page in which the instruction is located. To address the other pages, indirect addressing is necessary and is realised by using a defined location in memory. A more detailed explanation of the 21MX computer can be found in the Hewlett Packard Reference Manual (28).

. The PACE Microcomputer: Four main functional aspects need to be considered during the development of the system software:

. the use of the registers and accumulators;
. the use of the status and control flags;
. the method of data transfer for the memory or peripherals;
. the stack servicing for multiple task handling.

. The Registers and Accumulators: The PACE microprocessor has seven 16 bit registers with four accumulators (AC0, AC1, AC2 and AC3) which are available to the programmer. Register AC0 serves as the principal working register while Registers AC2 and AC3 can be used as index registers.

. The Status and Control Flags: There are fourteen status and control flags which can be set or pulsed depending on program requirements for interrupt handling.

. The Data Transfer Techniques: The data transfer techniques for peripherals have already been briefly described in Section 4.2.4.2 (p.58). In contrast to the National approach other microprocessors (INTEL) and minicomputers (Hewlett Packard) have special input/output instructions. The PACE approach is more flexible by allowing the direct use of the full instruction set with the external devices by treating them as memory locations. In developing the software, consideration had to be given to the allocation of addresses to the memory and to the peripherals.

For the front-end sub-system of the model, consideration also had to be given to the allocation of memory which was used by National Semiconductor for the PACE Development System peripherals. It was suggested by National that memory locations 8000_{16} to $BFFF_{16}$ should be allocated to any new peripherals which would not conflict with any future peripherals that National Semiconductor produce for their development system. Any interfaces which were developed were able to be used with the standard peripherals thus avoiding conflicting signals. Another important consideration for the memory allocations was to decide if the technique of "Split Base Page" was going to be used for the

sub-systems (see Figure 4.19 below and (p.138) Appendix A) where it was possible to allow for a split base page. The technique was used to share the base page memory area addresses between memory and peripherals for high speed access, refer to reference (31) for a more detailed explanation.

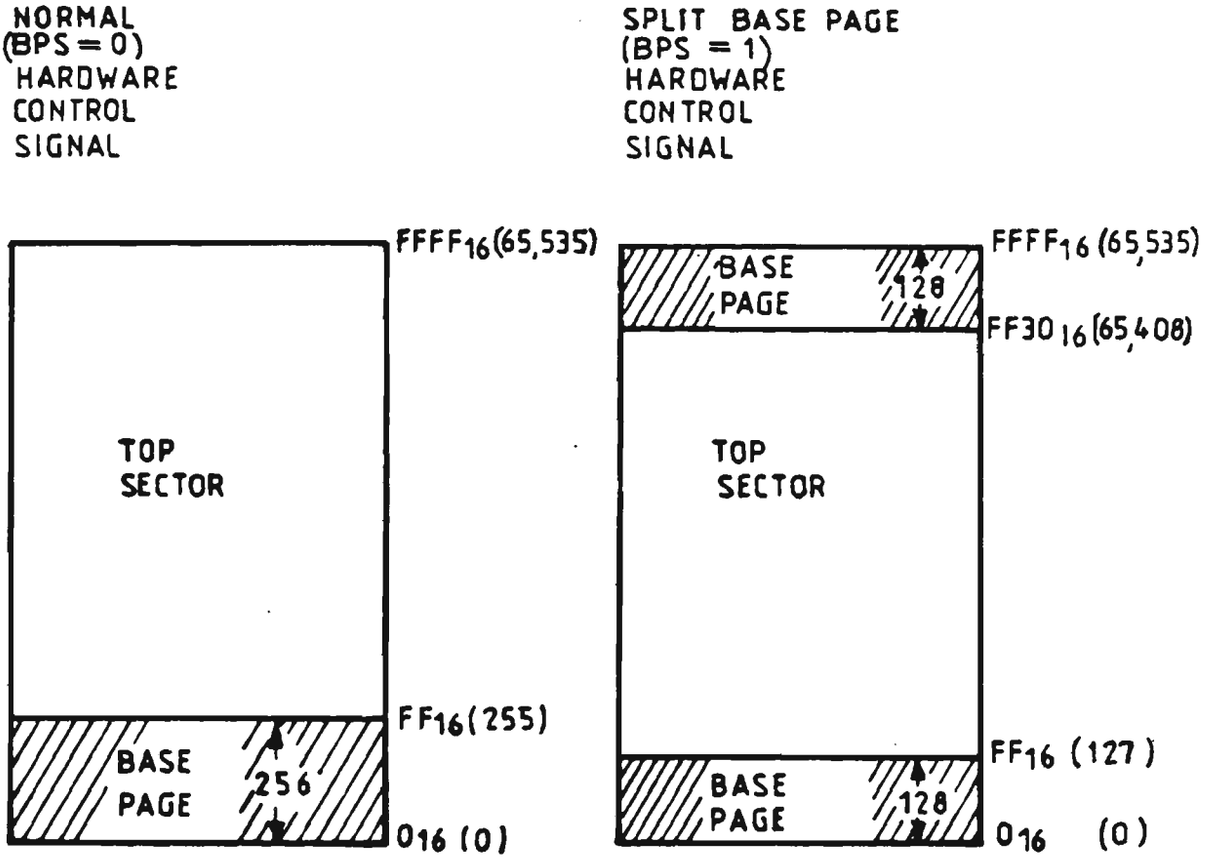


FIGURE 4.19 : BASE-PAGE MEMORY MAP

. The Stack Handling: An important functional aspect of the PACE microprocessor was the internal stack, which was used for servicing the sub-routines in the correct order by storing the return addresses of each sub-routine being serviced. The stack was also used for storing data

(accumulator content and flag status) by executing the "Push to Stack" instruction. The stack provided 10 words which were accessed in a sequential last-in, first-out (LIFO).

In simple applications this facility eliminates the need for an external stack but because of the size of the task and the different types of functions which were performed by the system model, it was necessary to develop a complex stack service sub-routine for each of the microcomputer sub-systems to provide increased stack space, refer to Section 4.3.3.3 for details.

4.3.2 System Software Elements

4.3.2.1 Standard Software

The main languages used were BASIC and the respective computer Assemblers (PACE and Hewlett Packard 21MX minicomputer). BASIC was used for driving the graphic representation of Figure 3.1 (p.16) in the Master, while the control and data transmission requirements were written in Assembler. The high level language "PASCAL" was not available at the time but from recent developments it appears that it could become a very useful language in the future (26) (15). In Section 5.2.2.1 the languages used in the Loy Yang Open Cut application are briefly covered.

The proven Assembler sub-routines (Load, Dump and Breakpoint) for loading and dumping programs and for breakpoint handling developed within the Electrical Engineering Department at Footscray Institute of Technology for the Hewlett Packard computer and the PACE microcomputer were integrated into the system software, refer to Section 4.3.3.2. Refer to

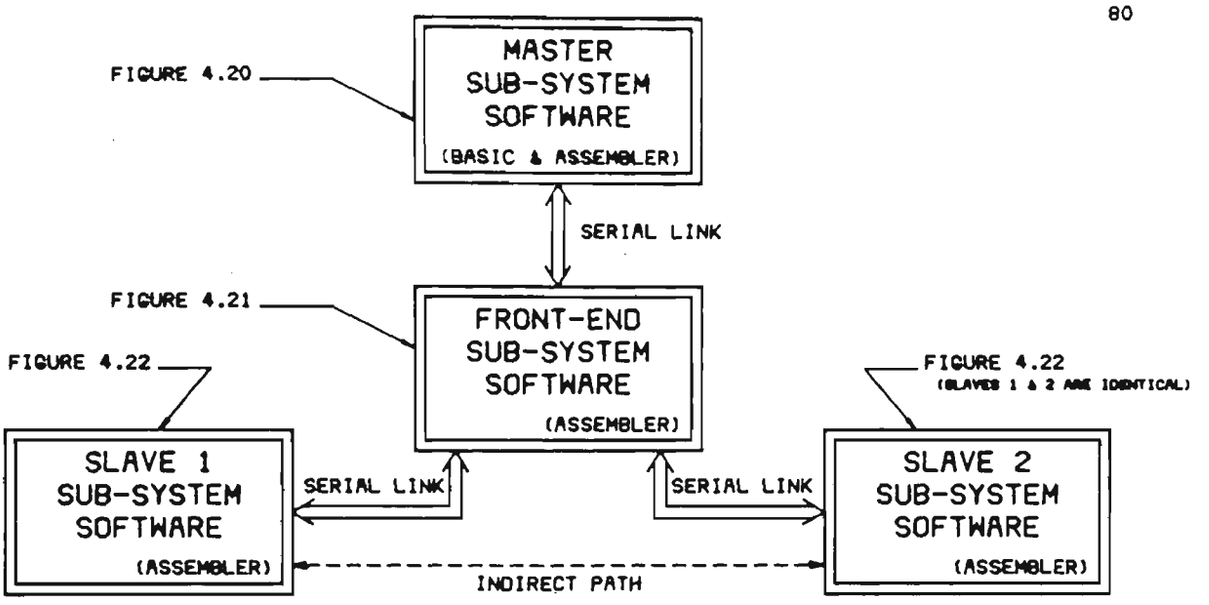
Figures 4.20 (p.80), 4.21 (p.82) and 4.22 (p.85) for a diagrammatic representation of the software developed by the author which is explained in the following Sections.

4.3.2.2 Developed Software for the Master

The role of the master in the hierachy of the model was fairly easy to demonstrate; hence, it did not require any complicated software (refer to Figure 4.20 for a diagrammatic representation of the master software).

To display the conveyor network of Figure 3.1 (p.16), use was made of the graphic software package for driving the graphic display VDU. The software for the master was written in BASIC in order to make use of the graphic display calls available for driving the VDU.

The master only had two devices to service, the front-end and the graphic (or master) VDU. An input from the VDU Keyboard was used to interrupt the display sub-routine (refer to Section 4.3.3.4 (p.98) for more details). To simplify the master sub-system software, the messages for communication with the lower levels (front-end and slaves) were predetermined and stored in memory in tabular form. A look-up table of VDU Keyboard characters was used to access the different messages for the demonstration of data transfer between the sub-systems as explained in Section 4.3.4. (p.99).



(INTERCONNECTION DIAGRAM FOR SYSTEM SOFTWARE , FIGURES 4.20,4.21 & 4.22)

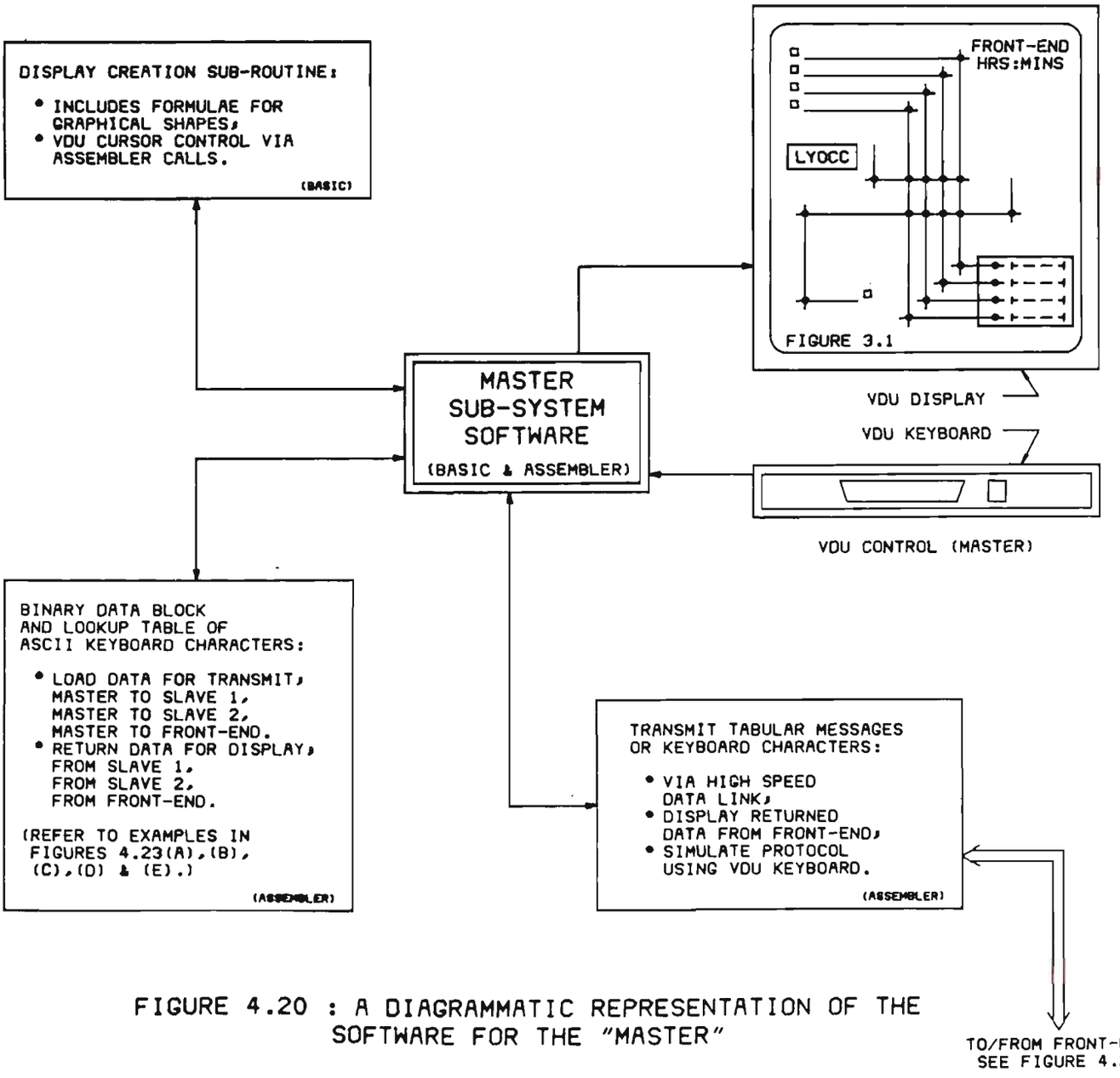


FIGURE 4.20 : A DIAGRAMMATIC REPRESENTATION OF THE SOFTWARE FOR THE "MASTER"

4.3.2.3 Developed Software for the Front-end

The most important sub-system in the model was the front-end, because it had to handle messages and interrupts from several sources, i.e. the master and slaves data links, its own internal functions (Real-time clock, Stack handling and Debug control) and the connected peripherals.

Basically the operation of the front-end depended on the interrupts from the various sub-routines (refer to Figure 4.21 for a diagrammatic representation of the software). In its steady-state mode, the front-end cycled through the debug and interactive character codes in a continuous loop. Any interrupts from the Stack servicing, Real-time clock control, Master to Front-end, Front-end to Slave 1 and Front-end to Slave 2 sub-routines interrupted the front-end in a defined priority order. Each interrupt controlled sub-routine could interrupt a lower priority sub-routine.

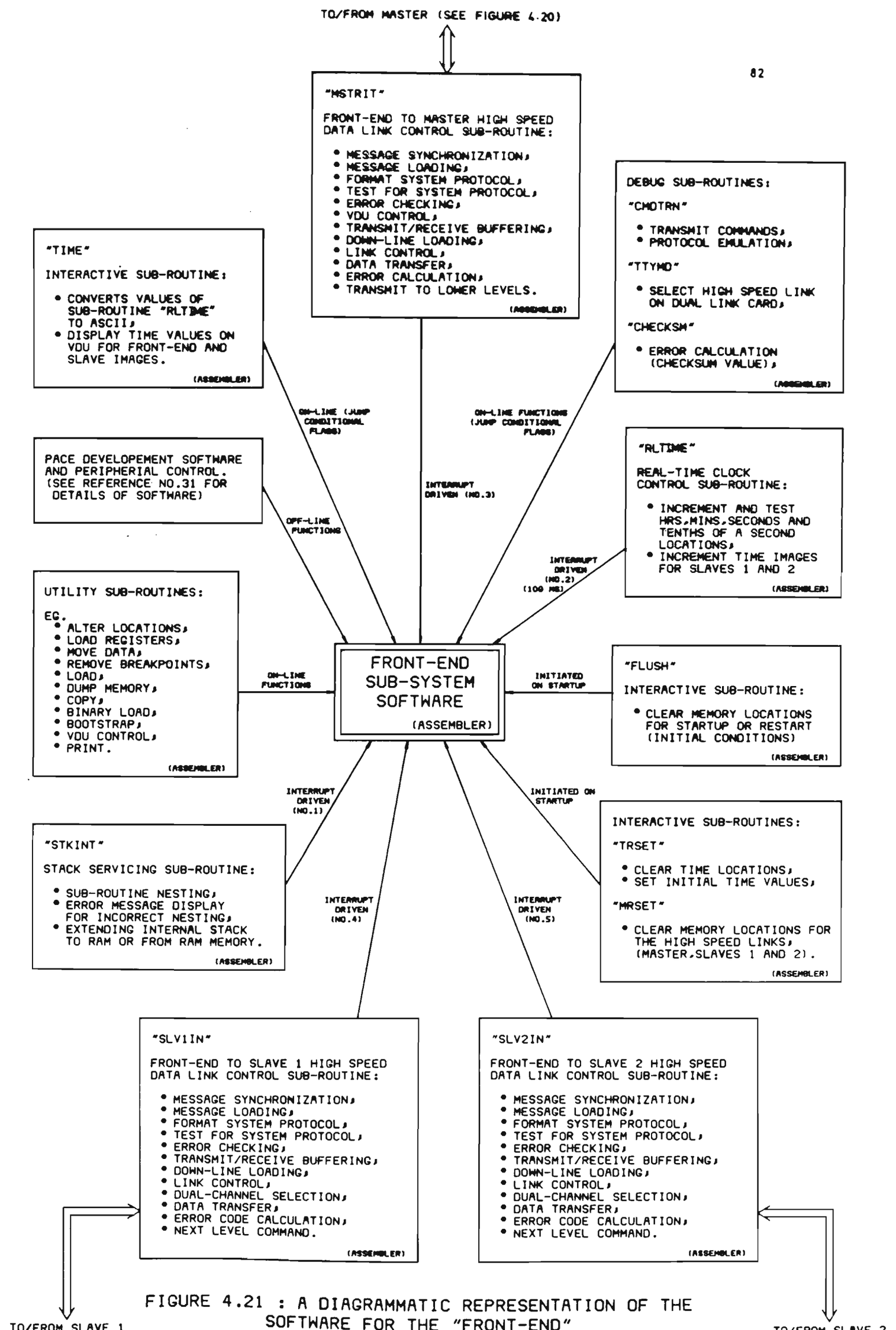


FIGURE 4.21 : A DIAGRAMMATIC REPRESENTATION OF THE SOFTWARE FOR THE "FRONT-END"

Sub-routine Priorities: The stack service sub-routine had the highest priority of all front-end sub-routines. As each sub-routine transferred the return address of the interrupted sub-routine and the contents of the registers to the internal stack, a check was made to determine if the number of words transferred to or from the internal stack had exceeded ten. This sub-routine could be initiated at any time but it did depend on the number of words on the stack during the transfer.

The second highest priority sub-routine was the Real-time clock sub-routine which was initiated every 100 milliseconds to update the clock values stores in memory.

In the front-end sub-system there were also the three major interrupt driven sub-routines, "MSTRIT", "SLV1IN" and "SLV2IN" which were used in the overall distributed system to communicate with the master (MSTRIT), Slave 1 (SLV1IN) and Slave 2 (SLV2IN). The sub-routine controlling the data link with the master (MSTRIT) had the highest priority of the three and the sub-routine SLV2IN had the lowest priority. The three sub-routines were similar except that the interrupt sub-routine "MSTRIT" also included a test for incoming VDU characters, refer to Appendix B (p.153) for the detailed flowchart. This test was used to determine whether a VDU had been connected in place of the master for control at the second level of the hierarchy.

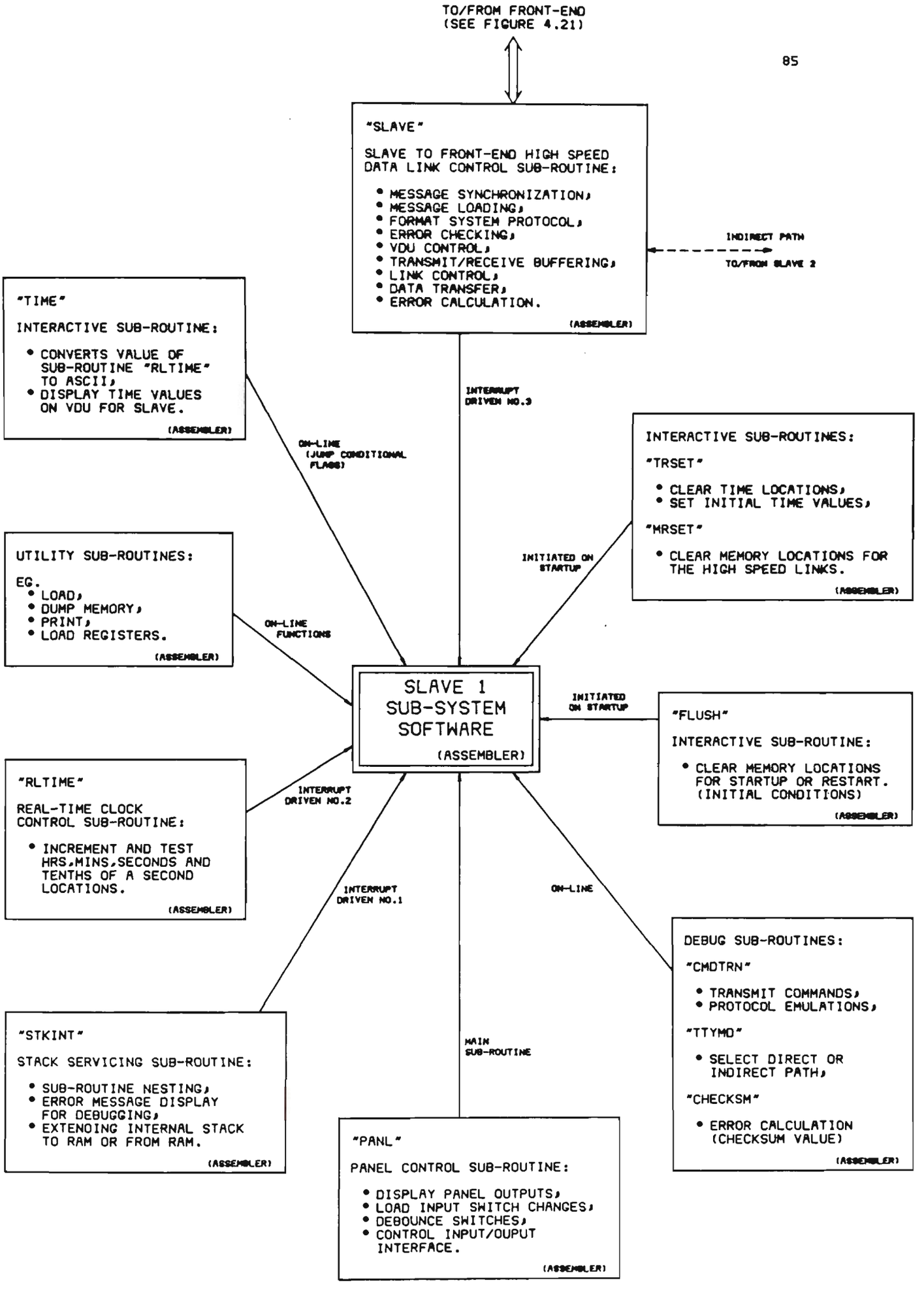
4.3.2.4 Developed Software for the Slaves

The software for the slaves used sub-routines with a similar structure to those used in the front-end, i.e. Stack servicing, Real-time clock, Data link and Debug sub-routines. In addition a minor sub-routine was

included for servicing the input and output panel as shown in Photograph No 7 (p.32). Refer to Figure 4.22 for a diagrammatic representation of the software for the slave.

The Panel was used to represent simple field inputs and outputs for local control at the conveyor plant. The control logic is far more complex and the number of inputs and outputs are much greater on actual conveyors. The panel was used to simulate the motors and protective devices connected in a real situation and the change of state of the panel was stored in memory with the time of the occurrence.

The slave sub-system software could be extended to simulate a small conveyor, including starting several motors, if required for future investigations.



NOTE:
SUB-ROUTINES STRUCTURED THE SAME AS THE FRONT-END.

FIGURE 4.22 : A DIAGRAMMATIC REPRESENTATION OF THE SOFTWARE FOR THE "SLAVES"

(SLAVES 1 & 2 ARE IDENTICAL)

4.3.3 The Structure of the Fundamental Software

4.3.3.1 Interactive Programs of the Front-end and Slaves

In industrial control systems interactive programs are performing an increasingly important role in the man to machine interface. Although it is not possible to quote current examples for conveyor systems (microcomputers have only been used since the research was completed for large conveyor control tasks) it was decided to incorporate some examples in the system model.

Typical examples of the interactive sub-routines developed and used in the model are :

- Clear memory locations "FLUSH";
- Clear current time values "TRSET";
- Clear interrupt pointers "MRSET";
- Display current time "TIME".

The 'clear' memory type sub-routines (FLUSH, TRSET and MRSET) were used during system startup, the sub-systems requested simple decisions in order to establish the initial conditions of the sub-system. Other sub-routines, such as "TIME", were used during normal operation, if for example the current time of the front-end was required, depressing the 'T' character of the connected VDU keyboard would initiate the sub-routine "TIME". This sub-routine converted the values stored by the interrupt driven sub-routine "RLTIME" into English text for display on the VDU.

The interactive sub-routines also enabled control of the hierarchial network from any level in the model. The control point could be shifted using a VDU and keyboard to initiate messages to the different sub-systems (refer to Section 5.1.5 (p.110)).

Other examples of the interactive sub-routines are shown in Appendix A (p.135) under the heading of 'FRNTND COMNDS'.

4.3.3.2 The Debug Programs of the Front-end and Slave

Use was made of the standard sub-routines, ('Load', 'Dump' and 'Breakpoint') developed at Footscray Institute of Technology for debugging the system.

These three sub-routines were combined with many others especially developed for the project by the author.

Typical examples of the special Debug sub-routines developed and used in the model were:

- Transmit a command "CMDTRN";
- High speed data link select "TTYMD";
- Error code calculate "CHEKSM";
- Print memory locations "PRINT".

Refer to Appendix A for the detailed flow charts and software listings. The debug sub-routines enabled the model to be gradually built up until all of the sub-systems were linked together and properly communicating using the defined system protocol for data transfer.

The initiation of the debug sub-routines was also achieved in the same way as for the interactive sub-routines. The debug sub-routines were the basic sub-routines to which all of the other sub-routines were added.

Each of the debug sub-routines had specific functions to perform for the model as follows:

. "CMDTRN" (p.147): This was an important debug sub-routine and was initiated by the control 'Z' character. The sub-routine enabled the eight bit data, or keyboard characters following the control 'Z' command to be transmitted via the high speed data link (Front-end to Master, Front-end to Slave 1 and Front-end to Slave 2) until an "EOT" character was processed. It was possible to transmit interactive replies and debug commands or to simulate a data transfer from one sub-system to the next (i.e. front-end to Slave 1).

  This sub-routine also enabled the transmission of simulated messages (using the system protocol) that were formulated using the VDU keyboard.

. "TTYMD": This sub-routine was used to define which channel of the dual RS232C data link was connected to the VDU. There were four possibilities of connecting the VDU to the front-end as can be seen from the number of cards in Photograph No 3 (p.51), i.e.:

- The Front-end to Master data link card;
- The Front-end to Slave 1 data link card;
- The Front-end to Slave 2 data link card;
- As a normal peripheral device.

- "CHEKSM" : This sub-routine was developed as part of the data link control sub-routines and as a debug program to provide the error code (a checksum value) for a block of data between any two memory locations "XX" and "YY" given by the command - "CS, XX, YY" (refer to Appendix A (p.135) for more details).

For the purpose of this project, the error code is defined as the calculated checksum of the information field (16 bit words), between the "STX" and "ETX" control characters of the system protocol. This value was calculated by the originating sub-system and stored in the address field of the transmitted message as shown by Figure 3.5 (p.41).

- "PRINT" (p.130) : This sub-routine was developed to provide a print-out of the contents of a block of memory between any two memory locations "XX" and "YY" (refer to Appendix A for more details).

4.3.3.3 The Interrupt Sub-routines of the Front-end and Slaves

As outlined in the preceding sections, the servicing of interrupts was a fundamental mode of operation of the system model.

The structure of the software and the data link card logic was such that the first eight bits received initiated an interrupt controlled

sub-routine. The selected sub-routine turned off the interrupt request logic of the lower priority interrupts until the incoming data block had been received or there was an enquiry on the data received. If the data was correct or there was an enquiry, the interrupt request logic was again enabled after the transfer was finished. The transmitting sub-system did not continue transmitting data if the previous data word was not received back as an echoed message from the receiving sub-system. Therefore, while a message was being received via one high speed interface, the other lower priority sub-system interface cards had their echo and interrupt functions turned off to avoid any loss of information. For example, if the front-end was processing a message from the master, the higher order interrupt would take precedence over a slave message transfer but it would not be able to stop the transmission from the slave if it did not also turn off the echoing of received data.

There were five major interrupt driven sub-routines involved:

. Stack servicing "STKINT";
. Real-time clock control "RLTIME";
. Master to Front-end control "MSTRIT";
. Front-end to Slave 1 control "SLV1IN";
. Front-end to Slave 2 control "SLV2IN".

The Stack servicing sub-routine had the highest priority with the Front-end to Slave 2 control as the lowest priority. As the title of each sub-routine suggests each sub-routine had a basic function to perform, i.e:

. Stack Servicing "STKINT" (refer to Appendix A (p.131) for the detailed flow chart and listing): An important aspect of a multiple task software system is its ability to be able to restore and continue executing any sub-routines that have been interrupted by a higher priority sub-routine. It became evident while trying to integrate all of the software tasks to establish the hierarchy of the model that it would not be possible to execute the software sub-routines in a sequential order and still maintain REAL-TIME control. Each interrupt would use five words of the stack for storing the return address and the contents of the four registers, therefore, three interrupts would require an extension to the stack. It was necessary to extend the ten word stack handling capabilities of the PACE microprocessor chip by developing a complex sub-routine (STKINT) which extended the stack into external memory. The stack-full or stack-empty signals of the PACE microprocessor were used to initiate the "STKINT" sub-routine via the interupt servicing function. The sub-routine transferred a full stack to memory or restored an empty stack from memory. It was not possible to control the return to each of the interrupted sub-routines by using only the "Push to Stack" instructions to control the sequence of return without extending the storage area.

During the development phase of the sub-system software, it was necessary to include a message to the VDU "stack-full" or "stack-empty" in order to monitor the co-ordination of the transfers to or from the stack. The sub-routine could be

initiated during the execution of any of the interrupt driven sub-routines, or by a "Push to/Pull from" the stack instruction, which causes the internal stack to overflow;

. Real-time clock control "RLTIME" (refer to Appendix A (p.126)) for the detailed flow chart and listing):

In order to provide the Real-time values for the interactive sub-routine "TIME" in each of the sub-systems, a Real-time interrupt driven sub-routine "RLTIME" (p.141) was introduced into each of the sub-systems. This sub-routine was initiated by the hardware clock card every 100 milliseconds. The sub-routine utilised the second highest priority interrupt and saved the contents of the registers and the return address of the active sub-routine before executing the sub-routine. The memory location for the tenths of a second portion of the stored current time value were incremented and tested to determine if the seconds, minutes or hours should also be incremented.

The sub-routine "TIME" for the front-end also updated the last image it had received from each of the slaves which was used to determine if the sub-system's clocks were synchronised. All of the values were stored in ASCII characters which provided the format for the display sub-routine "TIME";

The most important sub-routines developed for the model were the interrupt driven data link control sub-routines which were used to link the sub-systems together to achieve the system hierarchy.

Each sub-system used similar interrupt driven sub-routines to format the data (refer to Figure 3.5 (p.41) for the required format) for transfer via the high speed data links. The front-end had the three major interrupt driven sub-routines ("MSTRIT", "SLV1IN" and "SLV2IN") to control its data links with the other sub-systems.

The data link interrupts sub-routines provided many of the requirements for the model, i.e.:

. orderly servicing of the data transfers between the sub-systems;

. the method of data transfer;

. the prototype system hierarchy;

. the message protocol and error checks.

Since the structure of each sub-routine for controlling the high speed data links was very similar, only an explanation of the front-end sub-routine "MSTRIT" will be given here:

. Master to front-end control "MSTRIT" (refer to Appendix B (p.153) for the detailed flowchart and listing): When the UART on the interface card connected to the master received the first character (eight bits), it generated the interrupt (as explained in the hardware Section 4.2.2) which initiated the "MSTRIT" sub-routine.

The character received by the UART was then checked against a defined table (refer to Appendix B (p.161)) to determine if it was a VDU command or a synchronising character "SYN" indicating the start of a data transfer. Ten errors were allowed to enable the link to settle before an inquiry on the data received was transmitted back to the transmitting sub-system.

Once the link was established, a test was carried out for six start of heading characters "SOH" to identify the first header block as required for the system protocol. The next header block, which was separated from the first by a single "SOH", gave the next data link channel number for the front-end to use if the message was to be passed onto one of the slave sub-systems.

The address block enabled the variable length data of up to 256 words to be loaded commencing at one of 256 locations in the different sub-system memories depending on the address defined by the originating sub-system.

The memory of the sub-systems had been divided into 256 sectors of 256 words (256 x 256 = 65 536 maximum memory size). This provided the flexibility necessary in this type of distributed system for loading different memory arrangements depending on the individual slave configurations or control tasks. The transmit and receive buffers and the free memory space of each sub-system could be in different locations for each sub-system. Hence, the sector address and relocatable address pointer in the control block provided the required flexibility. The address field of the message (refer to Figure 3.5 (p.41)) also included the error

code for the data hence when the data had been received and loaded in the sub-system receive buffer, it was checked by the sub-routine. If a difference was detected between the calculated error code (a checksum value) and the received error code then the transmission control character "NACK" was transmitted back to the transmitting sub-system, refer to the general flowchart of Figure 4.23.

The next field of the message after the address field contained the control commands of the "Control Block" for determining the actions to be taken by the receiving sub-system. Refer to Section 4.3.4 for typical examples.

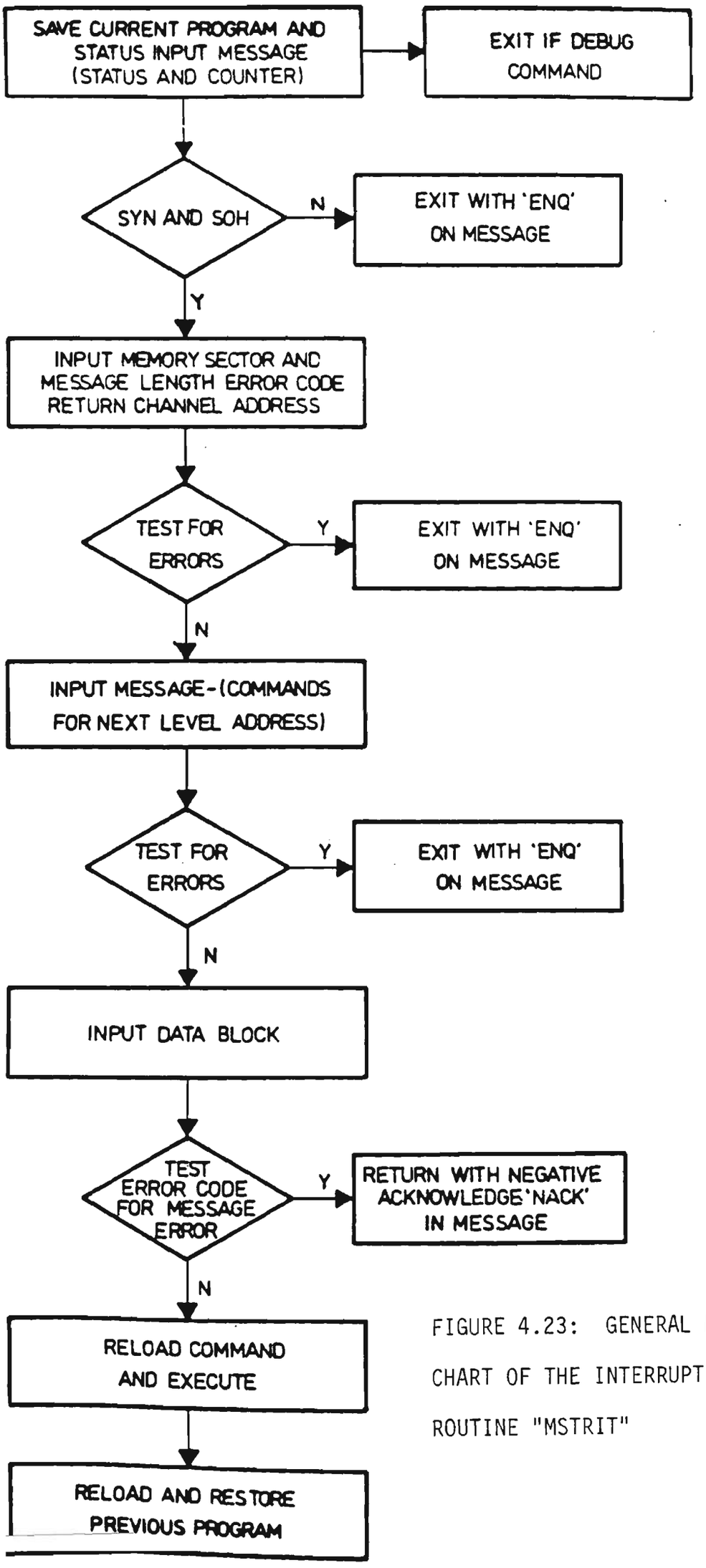


FIGURE 4.23: GENERAL FLOW-CHART OF THE INTERRUPT SUB-ROUTINE "MSTRIT"

The detailed flowcharts in Appendix B represent the sequence of action carried out by the high speed data link sub-routines, ("MSTRIT", "SLV1IN" and "SLV2IN") of the model.

In summary, the High Speed Data link interrupt service sub-routines of the sub-systems provided the basic software structure of the model. A message transmitted from the master contained the controlling commands (refer to Figures 4.24 (a), (b), (c), (d) and (e) for typical examples) which determined the actions to be carried out at the lower levels, similarly for messages from the front-end to the slaves. It was possible to control, alter and detect errors at all levels (master, front-end and slave) in the model.

4.3.3.4 BASIC Language Sub-routines in the Master

In order to draw a graphical representation of the open cut conveyor and dredger plant layout, a program was written in BASIC with Assembler sub-routine calls for the VDU (A Textronix CRT display unit with graphics). The symbols used in the display were calculated using mathematical formulae in BASIC with the resulting data transmitted to the display by way of an Assembler call. The unique symbols for representing the plant items were calculated and then used repeatedly depending on the number of conveyors or dredgers in a conveyor route.

Once the layout had been derived it was only a matter of requesting the raw data (consisting of CRT co-ordinates with the CRT dot on or off) for display.

In addition to the open cut plant display, a BASIC sub-routine with Assembler sub-routine calls was also used to set up the master to front-end data transfers. This enabled the BASIC sub-routine to control the display while transmitting data to the other sub-systems. A single keyboard character was used to initiate the Assembler call to transmit a defined data block.

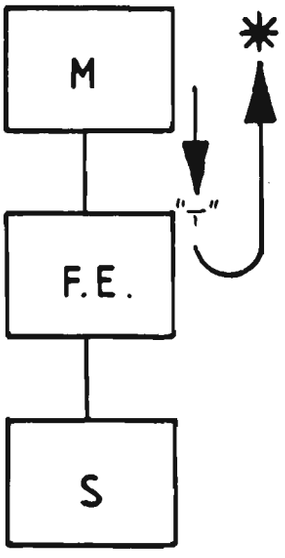
Once the data was available for driving the graphic CRT (the system VDU) from the system master it was also possible to use the data produced by the BASIC sub-routine to create a similar display with the Textronix unit connected at any sub-system. The sub-system's memory was down-line loaded with the raw display data from the central system. BASIC was confined to the system master and only the display data was used at the lower levels. BASIC was not used in the microcomputer sub-system software.

4.3.4 The Operating Strategy of the Model

An important feature of the message protocol in each of the high speed data link interrupt handling sub-routines (i.e. in "MSTRIT", "SLV1IN" and "SLV2IN") was the "Control Field" (refer to Figure 3.5 (p.41)). From the sub-routine, appendix B, (p.156) and the "Control Block" tables of Figures 4.24 (a), (b), (c), (d) and (e) it can be seen that the receiving sub-system substituted the first word of the "Control Block" with the second word and then reversed the two eight bit characters of the 16 bit word. This logic was used to enable the data to be transferred to the required destination or to be received from the originating sub-system computer regardless of the data transfer path which could be through several sub-systems and levels.

Each sub-system had the same basic protocol handling sub-routine; therefore, the treatment of the "Control Block" had to control the transfer of data through all levels. The words were reversed or substituted in order to set-up the correct transfer sequence as shown by the examples in Figures 4.24 (a), (b), (c), (d), and (e).

The least significant eight bits of the first word of the "Control Block" (which consisted of two ASCII characters) was used by the receiving sub-system, the second of the two ASCII characters was used by the next sub-system in the hierarchy. The second word of the "Control Block" was similarly used by any third and fourth level sub-systems as shown by Figures 4.24 (c) or (e).



MESSAGE "CONTROL BLOCK":	
Ø "T" / Φ Φ	1st Word) From ————) the 2nd Word) Master

Becomes:

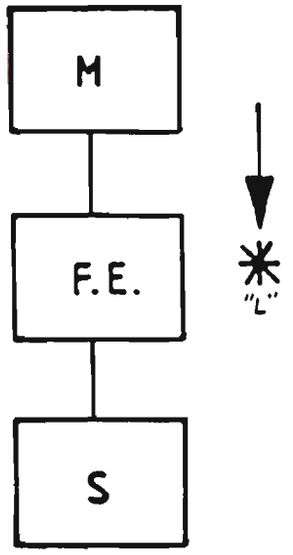
Φ "Φ" / Φ Φ	1st Word) Transposed ————) by the 2nd Word) Front-end

FIGURE 4.24 (a) MASTER TO FRONT-END : SEND A BLOCK OF DATA

Figure 4.24 (a) represents the case where, for example, there was a request for the current time of the front-end. The message block transmitted by the master contained the memory address of the front-end time value which had been stored by the sub-routine "RLTIME" as well as the "Control Block" as shown in the table above. The front-end computer responded with the two word data block (tenths of a second and seconds; minutes and hours) and included the error code for the data.

*	- Message destination
Φ,Φ,Ø	- Do not care characters (8 bits)
"T"	- Transmit command character (8 bits)
"L"	- Load command character (8 bits)
Ø"T"	- First word of command (Figure 4.24(a))
Φ Φ	- Second word of command (Figure 4.24(a)) (control commands)
(Possible "Control Block" characters)	

Each of the tables of Figures 4.24(a) to (e) indicates the required "Control Block" characters in the system message (refer to Figure 3.5 (p.41)). The original two words as determined by the master are shown. These two words are transposed by the receiving sub-system in order to determine the "Control Block" for the next level. The least significant character of the first word as shown by the quotation marks determines the action to be taken by the receiving sub-system.



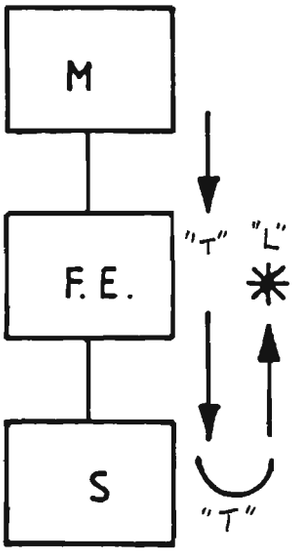
MESSAGE "CONTROL BLOCK":	
Ø "L" / Ø Ø	1st Word) From) the 2nd Word) Master

Becomes :

Ø "Ø" / Ø Ø	1st Word) Transposed) by the 2nd Word) Front-end

FIGURE 4.24 (b) MASTER TO FRONT-END : LOAD A BLOCK OF DATA

Figure 4.24 (b) is an example where the real-time clock value of the front-end value had to be changed. The new value was transmitted to the front-end, checked for errors and then loaded.



MESSAGE "CONTROL BLOCK":	
Master commands the slave to load a block of data into the front-end from the slave.	
Ø "T" / T L	1st Word) From) the 2nd Word) Master

Becomes in the front-end:

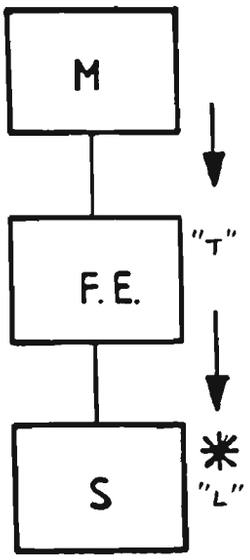
L "T" / L T	1st Word) Transposed) by the 2nd Word) Front-end

Then in the slave:

Φ "L" / Φ Φ	1st Word) Transposed) by the 2nd Word) Slave

FIGURE 4.21 (c) MASTER TO SLAVE : LOAD IN F.E.

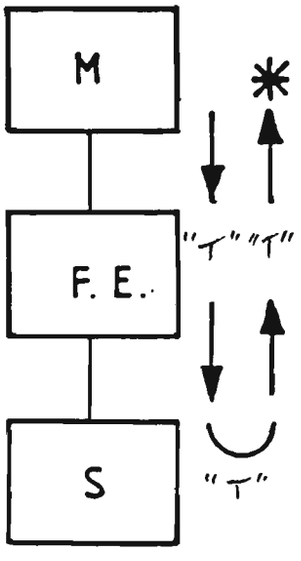
Another example of Figure 4.24 (c) would be the case of a direct link lost to a slave, the second RS232 link (or indirect path) would be used to communicate between the master and the slave via the adjacent slave.



MESSAGE "CONTROL BLOCK":	
Master commands the slave to load a block of data:	
ØT / LT	1st Word) From 2nd Word) the Master
Becomes:	
TL / ΦΩ	1st Word) Transposed 2nd Word) by the Front-end

FIGURE 4.24 (d) MASTER TO SLAVE : LOAD IN SLAVE

These two examples, 4.24 (d) and (e) are direct requests to a plant item such as a conveyor, Figure 4.24 (d) indicates the down-line loading of the starting parameters for the slave control sub-system and Figure 4.24 (e) is a request for data (a sequence of events) stored at the slave.



MESSAGE "CONTROL BLOCK":	
Master commands the slave to transmit a block of data back.	
ØT / TT	1st Word) From 2nd Word) the Master
Becomes in the front-end:	ΦT / TT
Then in the slave:	ΦT / TT)As Transposed

FIGURE 4.24 (e) MASTER TO SLAVE : SEND A BLOCK OF DATA

The examples in Figures 4.24 (a), (b), (c), (d) and (e) were used with the interrupts servicing sub-routines, the hardware, the system protocol and the different sub-systems (master, front-end and slaves) to

demonstrate the operating strategy of the Model. In order for the model to operate as a prototype system, each facet as explained in the preceding sections of Chapter 4 had to function correctly.

Depending on the function required, such as a request for the current time, a selection initiated via the system VDU would start a sub-routine which in turn generated a message using the system protocol for transmission to the lower levels.

A message from the master to the slave requesting data (i.e. from the control centre to the conveyors) would use all of the hardware facilities (interrupt flags, baud rate control and interconnected sub-systems) as well as the software facilities incorporated in the Model, such as the logic for processing the "Control Block" words.

A VDU and keyboard was used to control the hierarchical network from any level in the system while still maintaining the Master/Front-end/Slave configuration. Data was transferred through the system model using a message protocol that met the standards required (refer to AS 1484, Parts 1 to 5 (15) and Appendix F).

CHAPTER 5 : THE VALIDATION OF THE CONCEPTS INVESTIGATED

5.1 The Performance of the System Model

5.1.1 Independent Control

Using the model as outlined in Chapters 3 and 4, it was demonstrated that the concept of independent control can be maintained. This is especially important in open cut control of conveyors. Using the more traditional relay logic techniques of the past, the conveyor control systems had developed by the 1970's to a stage where the conveyor could continue to operate even with total loss of the control centre. It is essential for coal supplies to have no single point of failure in the overall control system. Therefore, one of the objectives of the research was to show that independent control at the plant would be maintained and that it could be improved by using microcomputer-based sub-systems. To demonstrate independent control using the microcomputer, the links (as shown in Figure 3.3 (p.26) between the slaves, the front-end and the master were disconnected. The slave continued to monitor the field simulation panel (refer Photograph No 7) and displayed outputs as a result of simulated inputs from the panel switches. The improvements in overall conveyor control and monitoring were demonstrated by using a VDU connected to the broken link at the slave. Control and monitoring was still possible using the VDU while separated from the central master and sub-master. It is therefore possible for mobile field personnel to control the conveyors via VDU units at the plant locations. The same exercise was repeated for the link between the master and the front-end. It also performed as an independent sub-system with control maintained

via the VDU connected to the disconnected interface. The software sub-routines of each sub-system, test for VDU control commands or the system protocol in any data received.

5.1.2 Results of Actual Data Transmission

Using the "Real-time Clock" card as outlined in Section 4.2.3, the changes in the field inputs were stored with the time of occurrence. After allowing for contact bounce, any change in the status of the input which had been previously transmitted to the master, initiated the transmission of a data block identifying the input and the time of occurrence. The slave sub-system was, therefore, able to record events chronologically with a resolution of one hundred milliseconds between occurrences which was approaching the speed of contact bounce. This improvement in recording is a significant advance over the Morwell Open Cut system's resolution of six seconds. In large conveyor networks, six seconds is not fast enough to record the order of occurrence of events over the whole open cut plant. The individual microcomputers have fewer inputs to monitor than a single overall computer and fewer functions to perform. As a result, the operations personnel would be able to identify the initiating fault.

In order to demonstrate the down-line loading of new sub-system parameters, a data block was transmitted to the slaves from the master in a similar manner as represented by Figure 4.24 (d) (p.102). The data contained new addresses for the direct and alternate links to the slave sub-system reversing their roles. As a result, the direct link changed over to the alternate link (refer to the indirect path of Figure 4.18 (p.74) demonstrating that any of the sub-system's parameters could be

changed to suit, for example, new digging patterns in a constantly changing open cut environment. This could avoid the need for major plant outages for changes to conveyor control functions.

In addition to providing a method of data transfer, the microcomputer based sub-system could in the future enable the removal of any fixed input to output interface between the conveyor control sub-system and the data acquisition sub-system. It was possible to demonstrate this by using the model because the slave sub-system developed, included both control and data acquisition functions within the one sub-system.

At the time of this study, each input monitored at the open cut control centres had a separate input in the data aquisition equipment connected to a separate output in the conveyor control systems. Using the serial transmission capabilities of the model, it was demonstrated that the status of the plant (approximately 6000 conveyor and dredger status signals in total) could be monitored using a transfer of data from the dedicated sub-systems. The number of inputs monitored by the master of the model were varied by the size and contents of the data block transferred.

It was realised during the development of the format of the data block and the message protocol for the model, that this aspect would represent the biggest problem for an open cut conveyor system. During the construction of the open cut, each of the conveyor sub-systems would be supplied by a different conveyor manufacturer, therefore, the plant specifications would have to detail the data and the expected message protocol formats. The interface, which would use a defined protocol instead of physical links, between the different manufacturer's

sub-systems, would still remain as the area to be carefully implemented. Serial data transfer does provide flexibility, but it would be necessary to specify the entire structure of the message and the data block. The execution of control commands ideally has to be error free.

5.1.3 Performance of the Hardware to the Manufacturer's Specifications

Two different manufacturer's Universal Asynchronous Receiver Transmitter (UART) chips were used for the high speed serial interfaces. The two devices were pin for pin compatible, with only small differences in their timing characteristics. The two devices performed identically at the lower speeds, but at 9600 baud the cheaper device did not always transmit the data loaded by the computer. This was evident when the data bit lengths were of the same time duration as the 'strobe' or 'transmit data' pulse which occurs at 9600 baud.

Another problem of the UART not stressed by the manufacturers, was its 'zero catching' feature. When standard Hewlett Packard software was used, the card control information was transferred to the interface card before the data. The problem with this, was that the control word transferred to the card contained zero's in the data section of the word. The zero's were retained by the UART blocking the data bits. Therefore, in order to use standard software sub-routines, a buffer had to be included on the interface card and the UART strobed every time it was accessed in order to clear any zeros.

5.1.4 Software Performance

Each of the sub-routines developed were used in the different sub-systems with only minor modifications. Another objective of the project was to show that a microcomputer-based sub-system would be better suited to the one-off nature of conveyor control than a hardwired sub-system. This was clearly demonstrated by being able to use the software developed for one sub-system in the other sub-systems.

A further objective was achieved by shifting some of the central system tasks to the sub-systems. Each of the sub-systems sorted its events into their chronological order and also tested each data transfer for errors before interrupting the master. This was a task previously carried out at the control centre.

5.1.5 Increased Reliability

As outlined in Section 4.2.5, the system model had an indirect link to the master via the alternate RS232C serial input and output on the high speed interface card. When the direct link was lost, it was demonstrated that the communication with the conveyor was maintained by using the alternative indirect data path as shown by Figure 4.18 in Section 4.2.5.

In the development of the system model, care was taken to ensure that, when the hierarchial strategy was established, this path had an order of priority lower than the direct path. To avoid a conflict or loss of message, it was necessary to test in this direction for messages on a time initiated basis and not have interrupts during the servicing of the direct link.

A second advantage of the dual link is that it provided an interface for the local command VDU, making it possible to take control at this level when required. Therefore, overall control can be maintained by using the VDU at the front-end level or conveyor level, refer to Figure 5.1. The alternate path provides a more reliable link between the sub-systems and can also increases the overall reliability of an open cut control system by providing more than one point of control and communication path.

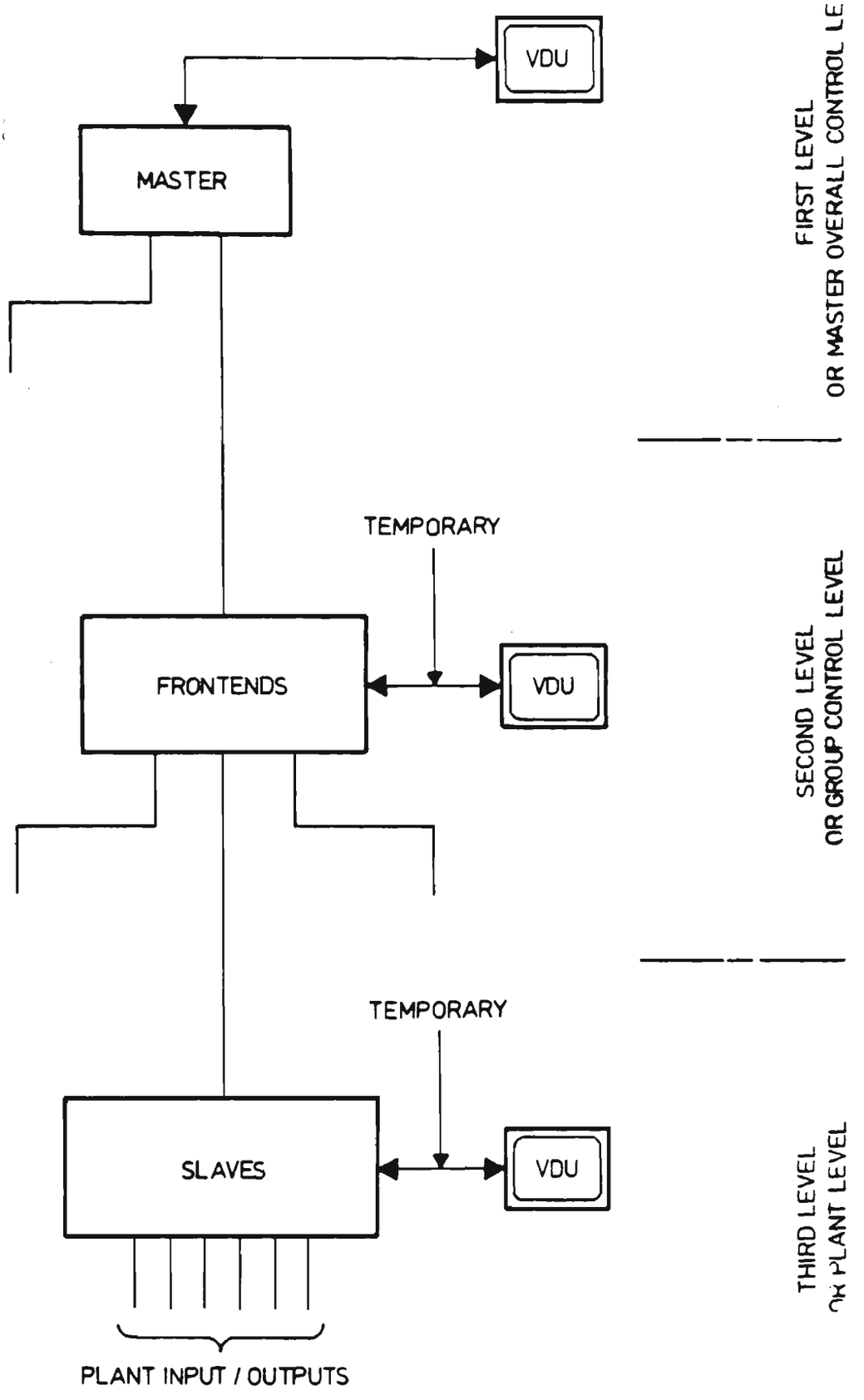


FIGURE 5.1 : VDU CONTROL LOCATIONS (LEVELS) IN THE MODEL

5.1.6 Interrupt Servicing for Multiple Tasks

Careful planning was required for the servicing of the interrupts of each sub-system task. Priorities were set for the different elements in the system model. Also, the multiple sub-routine handling techniques required modifications initially until the correct return addresses for the sub-routines were obtained. Light emitting diodes (leds) were used as a visual indication of the data transfers during the execution of each sub-routine and were connected to the transmit and receive lines. Error messages for incorrect sub-routine handling and the echoing of received data were used to establish the procedures for multiple tasks implementation.

5.2 THE SUCCESSFUL IMPLEMENTATION IN AN OPEN CUT ENVIRONMENT

5.2.1 General

After the research work had been completed and tenders had been called, orders were placed with Siemens for two large systems, one for the Yallourn Open Cut and another for the Loy Yang Open Cut. The Loy Yang Open Cut system is the most recent and was completed in November 1983, hence more of the objectives have been achieved at Loy Yang.

5.2.2 The Loy Yang System

Briefly the system for the Loy Yang Open Cut consists of three major parts, the Central Computer System (the master), the Remote Data Acquisition equipment (equivalent to the front-end and slaves) and the Microcomputer Conveyor Controllers.

5.2.2.1 The Central Computer System

The central system consists of two Siemens 300-R30 16 bit minicomputers in a master (active) and stand-by configuration. Either minicomputer can takeover the active role. A supervisory watchdog unit monitors the operation of the central system and in the event of a failure of the master or any peripheral, it automatically switches over the alternative device. Manual selection via a control panel of the back-up equipment is also possible. The central unit also contains a master clock for synchronising the overall system.

The languages used in the central system were 'Fortan' (for volume calculations), 'Pascal' (for displays), 'Assembler' (for system reports) and 'Simat' a Siemens control language.

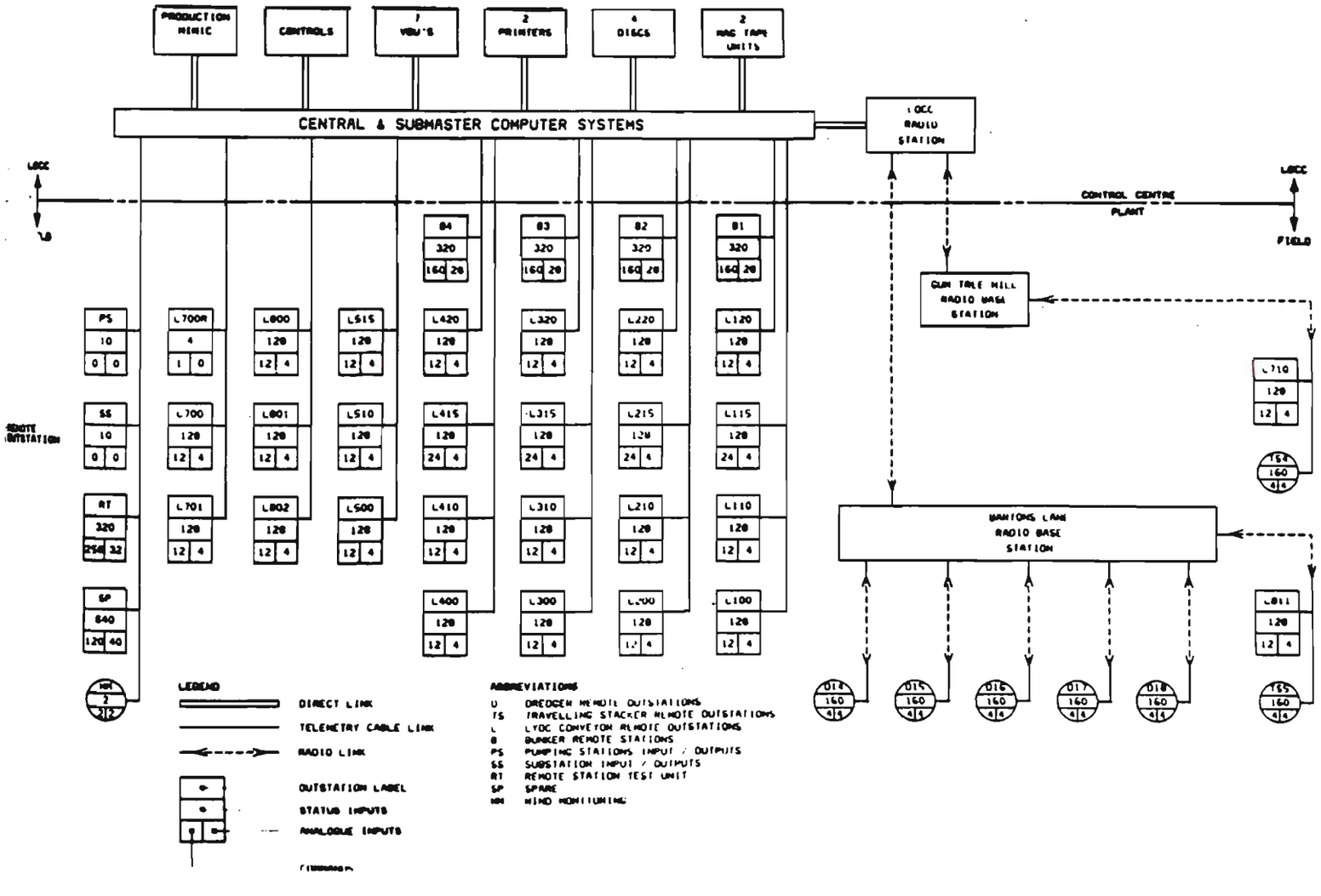


FIGURE 5.2 : LOY YANG SYSTEM CONFIGURATION

5.2.2.2 The Data Acquisition Equipment - The Central Remote Controllers and the Remote Outstations

The Central Remote Controllers are located in the control centre and comprise 11 microcomputer-based sub-masters which are linked via serial links to the central master. The sub-masters reduce the functions performed by the minicomputer by directly controlling the plant based slaves or remote outstations. All software is written in Assembler. The tasks performed by the central sub-master units are:

- message error checking;
- cyclic checking of remote outstations;
- maintaining an image of the plant status;
- control of the data transfer between the master and slaves.

The Central Remote Controllers are linked to the fixed plant (conveyors) over dedicated cable pairs and two wire modems, whereas the dredgers are linked via UHF radio links. The system of central remote controllers and data links are configured such that a single failure does not cause the loss of more than 10% of the system or one group of plant items.

The Remote Outstations (slaves) are designed to provide the necessary interface between the control centre and the plant items. The slave monitors and controls the status of the plant and transmits information back to the Control Centre. Each remote outstation has the same message checking and control capability as the central sub-masters.

A brief summary of the Remote Outstations functions are:

- the scanning of digital inputs;
- the scanning of pulse counters;
- analogue sampling of the raw coal bunker levels;
- storage of events during link loss;
- chronological recording of field events;
- the provision of control outputs.

5.2.2.3 The Microcomputer Conveyor Controllers

The microcomputer conveyor controller provides the same basic functions as the earlier relay or discrete logic systems. It controls the motors and provides outputs and inputs for the central system remote outstations. The original specification for the conveyor control sub-systems included the requirement for a serial link to the central system. But as a direct result of the research and a check on the developments in industry, it was decided to continue to use the traditional input to output interfacing method between the conveyor control and data aquisition systems as the major suppliers of control systems did not have a suitable serial link or message protocol available for industrial conveyor control systems at the time.

5.2.3 The Concepts Incorporated in the Loy Yang System

Many of the concepts investigated during the research program were incorporated in the Yallourn and Loy Yang Open Cuts, for example:

- independent control of plant was maintained by ensuring that the plant microcomputers do not rely on the central computer;

- chronological recording and secure data transmission techniques are used;

- some of the central system tasks have been distributed out to the data aquisition sub-systems;

- alternate paths are available via short sections of cables between the conveyor lines at the transfer and bunker areas;

- the traditional conveyor control sub-systems have been replaced by programmable microcomputer sub-systems.

The functions of the microcomputer conveyor controllers and the remote data acquisition equipment (remote outstation) were performed by the single slave sub-system in the system model as a single unit.

CHAPTER 6 : CONCLUSIONS

Following the development of the system model it has been demonstrated that a digital microcomputer-based system can successfully fulfil the basic requirements for the control and monitoring of an open cut coal and overburden transport scheme. All of the basic functions required for a full scale system were modelled, including data transmission between the different levels.

The timely installation of the distributed system in the State Electricity Commission's Loy Yang Open Cut with a minimum of delays has reinforced the original motivations behind the research. The Loy Yang system has incorporated many of the concepts investigated during the research program. It also has the capabilities of including more of the concepts in the future when further developments in the industrial control field make them standard features.

As a result of the research, the following conclusions were reached:

. A scheme using microcomputers for remote control of a conveyor transport network connected to a master while maintaining independent local control at the remote plant is feasible;

. System reliability and information gathering techniques can be improved by distributing some of the central system tasks between the plant sub-systems;

- An improvement in the detection of faults and the management of capital intensive large industrial plant is possible;

- Control at several levels in a large plant situation is possible by using the facilities provided by the microcomputer-based sub-systems;

- Many of the computer techniques, such as down-line loading, normally associated with large computer systems are also feasible with distributed microcomputers;

- The interfacing techniques used between sub-systems can be simple requiring only a minimum of hardware. Serial data links can also be used for interfacing control VDU's providing a solution to the loss of control from the higher level;

- One of the most important functions of remote control and monitoring of an open cut installation is to maintain a reliable coal supply. The distributed microcomputer, with its inherent reliability, will enable outages caused by the control systems to be reduced;

- Interactive software packages and other computer techniques can provide data for the control engineer to assist with the fault finding;

- In the future, a greater use will be made of the microcomputer to implement control functions hence reduced one-off design costs;

. Further advances in the emulation of message protocol by either hardware or software techniques will also benefit distributed microcomputer schemes by defining the standards to be used.

Distributed computer systems require a well defined message structure in order to communicate with other computers within the system. The successful use of the microcomputer will depend on a detailed functional description of the application and a thorough knowledge of digital techniques.

CHAPTER 7 : FURTHER WORK AND RECOMMENDATIONS

Now that the basic system model has been developed, further work could be carried out in the areas where additional developments are needed before any more advancements in industrial control can be made. These areas are:

- down-line loading of control parameters;
- linking of different manufacturer's microcomputer - based conveyor control systems directly to existing control centre masters;
- distributing more of the central system tasks to the remote sub-systems;
- increasing the amount of preprocessing of information at the plant;
- introducing more interactive diagnostic techniques;
- achieving limited control via VDU displays at the different levels in a system hierarchy.

This work could be carried out with the actual systems after further tests with the model.

In addition to using the model, it is recommended that an investigation into directly linking the plant controllers with the system master should include a study of the multiprocessor sub-systems due for release in the

near future. This device might be able to handle the tasks of control and data acquisition more effectively than a single microprocessor system.

The above tasks were implemented in the model, but since the model was a limited control system, they were not recommended for the Yallourn or Loy Yang projects until further tests are carried out. The linking of conveyor controllers direct to the central master and software controlled indirect paths had been included in the original designs for the conveyor plant but were removed as a result of the study. These options will be investigated in the future depending on developments in the Control Industry.

Other concepts which were included in the plant and control centre specifications at the time of the research were:

. replacement of traditional techniques with microcomputer-based control;

. fast scanning of plant signals and chronological recording of data to a resolution of 25 milliseconds;

. serial transmission of data to the control centre;

. distributed central system tasks.

As a result of the research, these concepts were continued and have since proven to be successful.

Many of the concepts of the study were incorporated in the SECV's plant specifications at the time of the research work. This was due to the long lead times required for design and installation and although some of the concepts have been removed, the basic objectives of the study have been achieved.

Successful completion of the initial stages of the Loy Yang Open Cut projects in 1984 has demonstrated that there is a future for the microcomputer in industrial control. In addition, it has demonstrated a role for the close co-operation between large industrial control system users and Engineering Institutes.

REFERENCES

HARDWARE DEVELOPMENTS

1 Amendt, A J, "MICROPROCESSORS - THE REVOLUTION IN CONTROL SYSTEM DESIGN". Control and Instrumentation (February 1976), pp 28-31.

2 Deshon, W E, "MICROPROCESSORS IMPROVE SERIAL COMMUNICATIONS", Control Engineering (March 1978), pp 127-129.

3 Dominguez, J and Tennant, J, "MICROCOMPUTERS UNLOAD DATA LINKS", Electrical World (15 July 1975).

4 Freeman, L L, and Fowler, A M, "DEVELOPMENT SYSTEMS - HARDWARE AND SOFTWARE REQUIREMENTS", The Institution of Engineers, Australia, Conference Papers No 78/13.

5 Hailstone, G, "SOME ASPECTS OF THE USE OF COMPUTERS IN SUPERVISORY CONTROL SYSTEMS", Electrical Engineer (October 1975), pp 13-18.

6 Jenkins, D W, "THE CHOICES BETWEEN DISTRIBUTED AND CENTRAL COMPUTING CONTROL", Control Engineering (June 1978), pp 61-64.

7 Knott, G J, "POTENTIAL USE OF MICROPROCESSORS IN THE DISTRIBUTION INDUSTRY", Electrical Engineer (May 1978), pp 67-71.

8 Prophet, G, "DISTRIBUTED PROCESSING POWER - A NEW TOOL FOR PRODUCTION CONTROL", Control and Instrumentation (June 1977), pp 41-43.

9 Sargent and Lundy, "ELECTRONIC VS CONVENTIONAL CONTROLS IN POWER PLANTS ADVANTAGES AND DISADVANTAGES", Combustion (January 1978), pp 20-22.

10 South, E, "A DISTRIBUTED PROCESSING SYSTEM FOR NETWORK CONTROL AND MONITORING".

11 Tobias, J, "DEVELOPMENTS AND TRENDS OF THE MICROPROCESSOR", Control Systems, Number 3, pp 17-31.

12 Washburn, J, "COMMUNICATIONS INTERFACE PRIMER", Instruments and Control Systems (March 1978), pp 43-48.

SOFTWARE

13 Bishop, P G, Parish, C C M and White, D J, "A MEDIUM LEVEL PROGRAMMING LANGUAGE FOR MICROPROCESSORS", Central Electricity Research Laboratories, Job No VL035 (March 1974).

14 Falk, H, "MICROCOMPUTER SOFTWARE MAKES ITS DEBUT", IEEE Spectrum (October 1974), pp 78-84.

15 Gibbons, J, "WHEN TO USE HIGHER LEVEL LANGUAGES IN MICROCOMPUTER-BASED SYSTEMS", Electronics (August 1975), pp 107-111.

16 Ogdin, C A, "THE HIGHS AND LOWS OF MICROCOMPUTER PROGRAMMING LANGUAGES", Instruments and Control Systems (June 1978), pp 81-84.

17 Steger, J P, "INTRODUCTION TO MICROPROCESSOR PROGRAMMING", Electronic Engineering (October 1975), pp 43-47.

18 Telecom Australia Research Laboratories, "MICROPROCESSOR NEWSLETTERS", (May 1976), J Hont.

ERROR TECHNIQUES

19 Burton, H O, "ERRORS AND ERROR CONTROL", Proceedings of The IEEE, Vol 60, No 11 (November 1972), pp 1293-1301.

20 CCITT, "INTERNATIONAL TELECOMMUNICATION RECOMMENDATIONS", Chapter 2.

21 Martin, J D, "USING POLYNOMIAL CODES", Electronic Engineering (July 1978), pp 46-49.

22 Owens, A and Harknett, M R, "BASIC AND EXTENDED CYCLIC HAMMING CODES", Electronic Engineering (December 1975), pp 34-37.

23 Philips Industries Ltd, "ENCODER AND DECODER FOR ERROR DETECTION", GZF1202 Product Information, pp 18-19.

24 Townsend, R L, and Watts, R N, "EFFECTIVENESS OF ERROR CONTROL IN DATA COMMUNICATION OVER THE SWITCHED TELEPHONE NETWORK", The Bell System Technical Journal, Volume XL111 (November 1964), No 6, pp 2611-2638.

GENERAL REFERENCES

25 Australian Standards Association, "DIGITAL DATA TRANSMISSION AS 1484, PARTS 1 TO 5".

26 Bowles, K L, "PROBLEM SOLVING USING PASCAL", Springer-Verlag.

27 Cotton, A W, and Lowe, G J, "HIERARCHICAL CONTROL USING SATELLITE MICROPROCESSORS", The Institution of Engineers, Australia, Conference papers on Microprocessor Systems, Publication No 77/11 (1977).

28 Hewlett Packard, "21MX COMPUTER SERIES - REFERENCE MANUAL", December 1974.

29 Kuo, F F, "COMPUTER APPLICATIONS IN ELECTRICAL ENGINEERING SERIES", Prentice-Hall Electrical Engineering Series. Digital Electronics with Engineering Applications.

30 McNamara, J E, "TECHNICAL ASPECTS OF DATA COMMUNICATIONS", Digital Equipment Corporation.

31 National Semiconductor, "PACE USERS MANUAL", (December 1974).

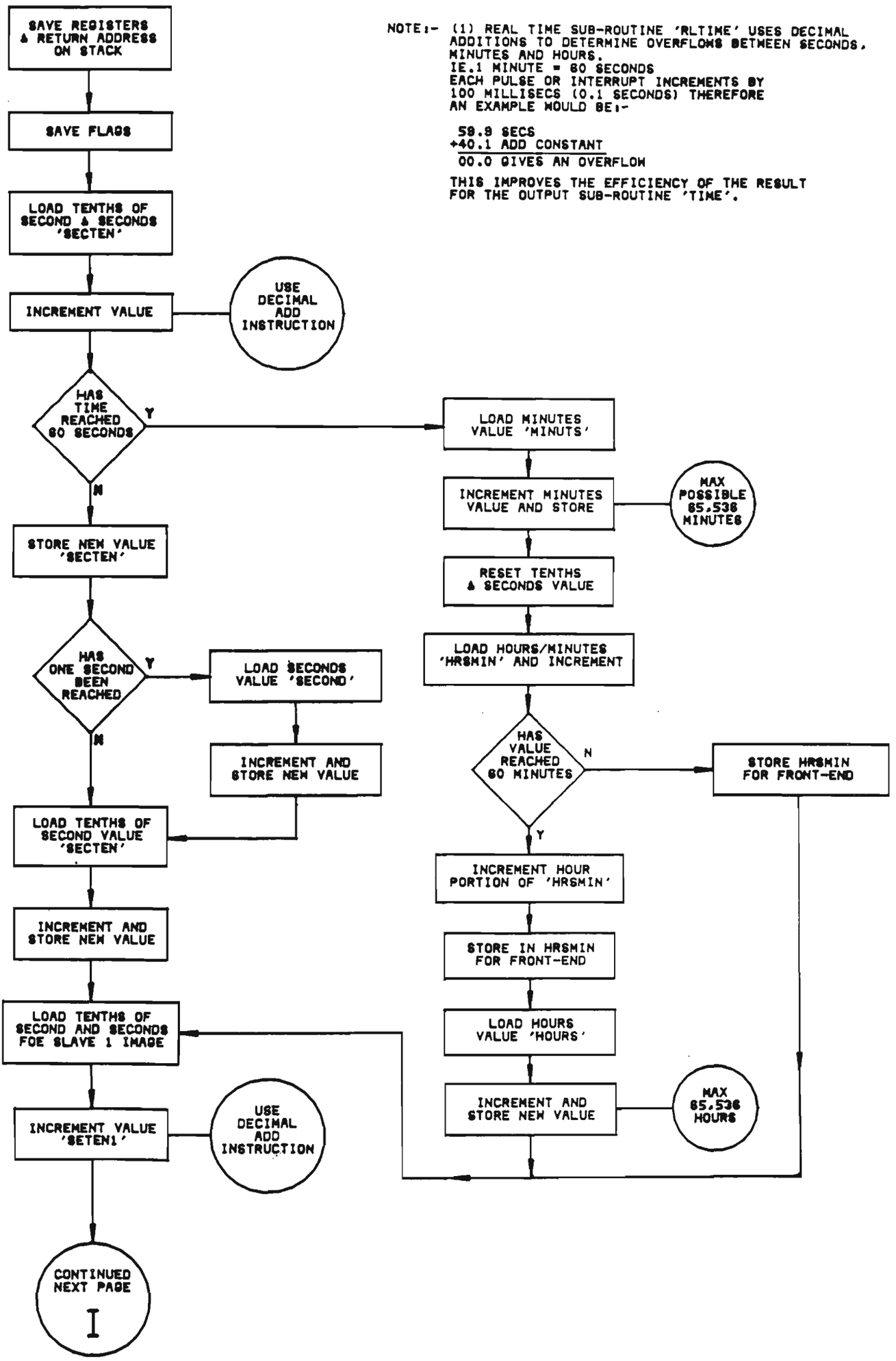
32 PACE, "MICROPROCESSOR ASSEMBLY LANGUAGE PROGRAMMING MANUAL", National Semiconductor (January 1977).

33 Wickes, W E, "LOGIC DESIGN WITH INTEGRATED CIRCUITS", Wiley, J, and Sons Inc.

APPENDIX A : SYSTEM SOFTWARE : PART A

INTERACTIVE AND DEBUG SUB-ROUTINES:

'RLTIME' SUB-ROUTINE - APPENDIX A

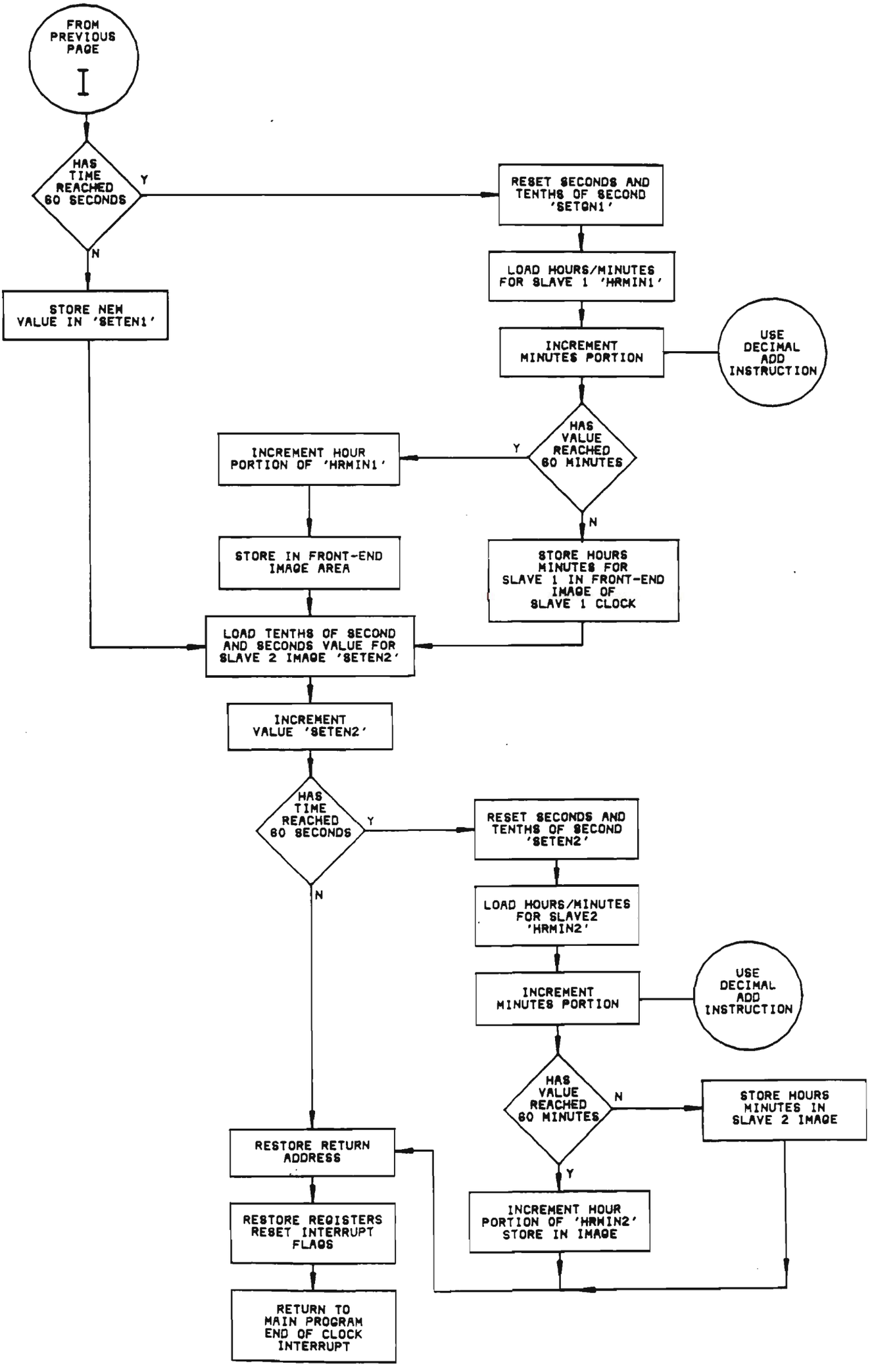


NOTE:- (1) REAL TIME SUB-ROUTINE 'RLTIME' USES DECIMAL ADDITIONS TO DETERMINE OVERFLOWS BETWEEN SECONDS, MINUTES AND HOURS.
IE.1 MINUTE = 60 SECONDS
EACH PULSE OR INTERRUPT INCREMENTS BY 100 MILLISECS (0.1 SECONDS) THEREFORE AN EXAMPLE WOULD BE:-

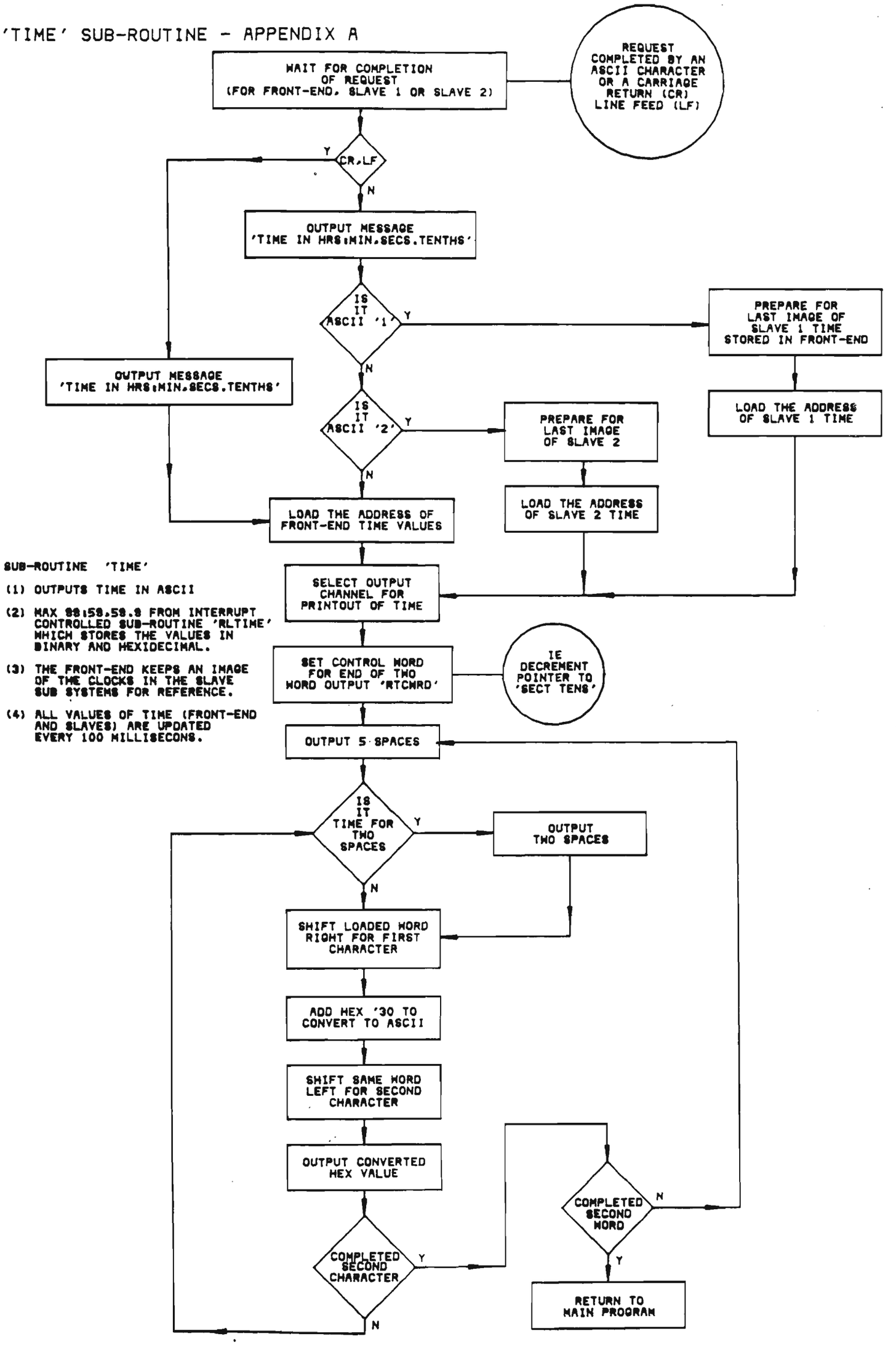
59.9 SECS
+40.1 ADD CONSTANT
00.0 GIVES AN OVERFLOW

THIS IMPROVES THE EFFICIENCY OF THE RESULT FOR THE OUTPUT SUB-ROUTINE 'TIME'.

'RLTIME' SUB-ROUTINE - APPENDIX A (PAGE 2 OF 2)



'TIME' SUB-ROUTINE - APPENDIX A



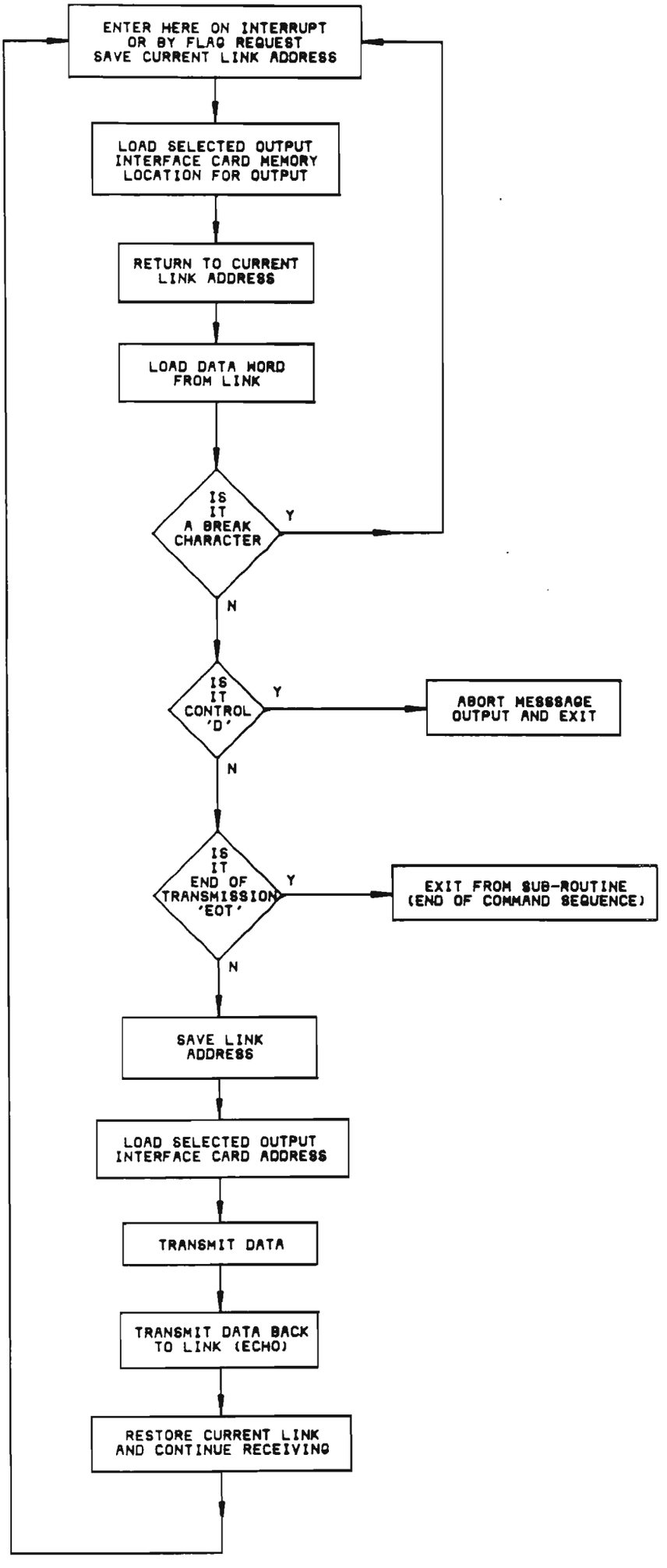
CONTROL 'Z' SUB-ROUTINE - APPENDIX A

NOTE:

(1) THIS SUB-ROUTINE ENABLES THE TRANSMISSION OF DATA WITHOUT THE SYSTEM PROTOCOL.

(2) THE LINK COULD BE CONNECTED TO ANOTHER LEVEL IN THE HIERARCHY TO RECEIVE THE UNFORMATED DATA OR A VDU UNIT.

(3) THE MESSAGE COULD BE USED FOR SIMULATING THE SYSTEM PROTOCOL.

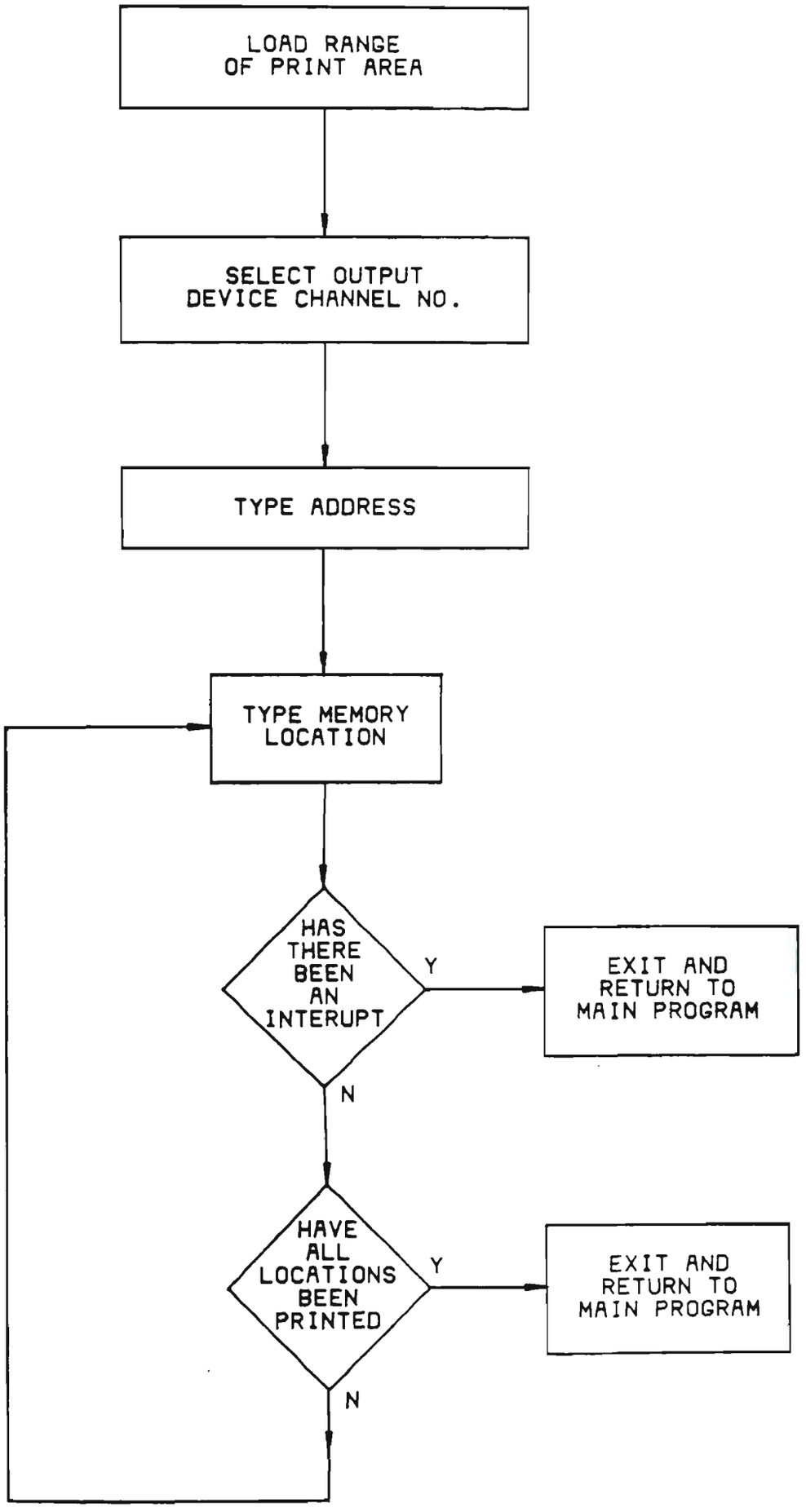


'PRINT' SUB-ROUTINE
- APPENDIX A

P'XX'
PRINT CONTENTS
OF 'XX'

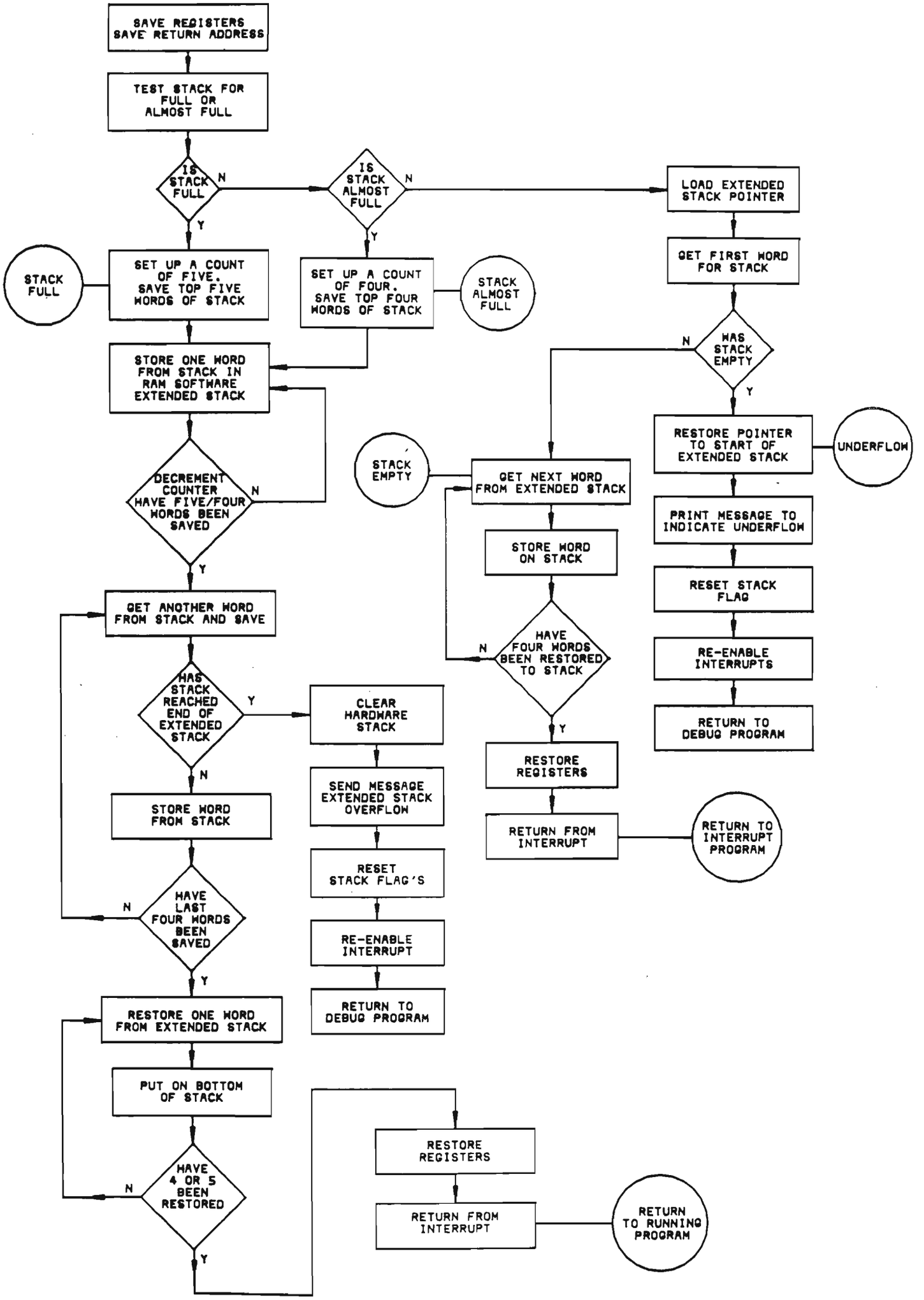
&

P'XX','YY'
PRINT CONTENTS
OF 'XX' TO 'YY'

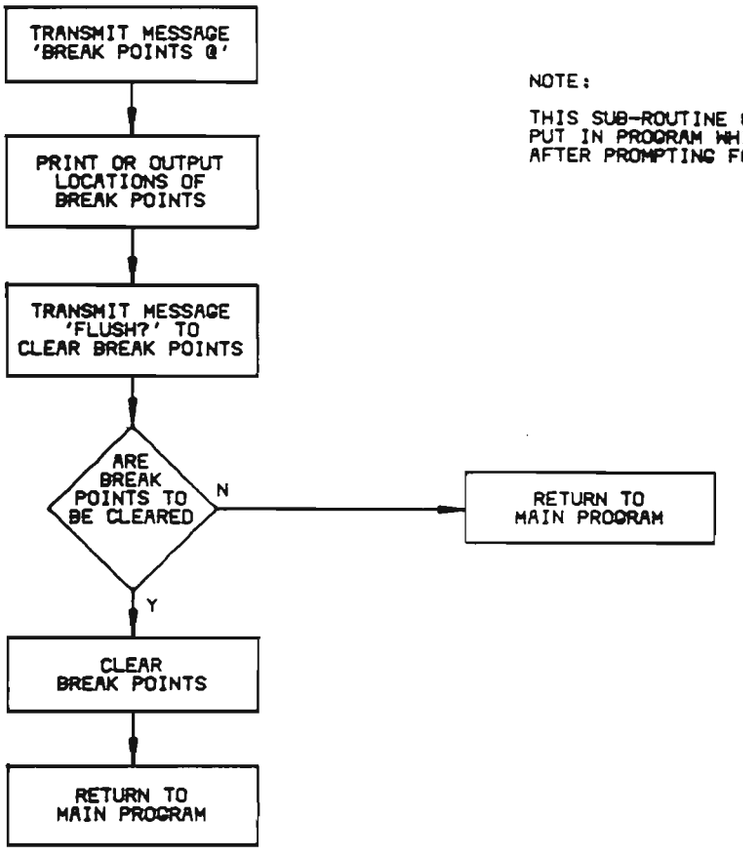


NOTE:- THIS SUB-ROUTINE PRINTS THE CONTENTS OF MEMORY FOR ONE LOCATION OR BETWEEN TWO LOCATIONS 'XX' TO 'YY'.

'STKFULL' SUB-ROUTINE' - APPENDIX A
'STKINT' (HIGHEST PRIORITY PROGRAM)



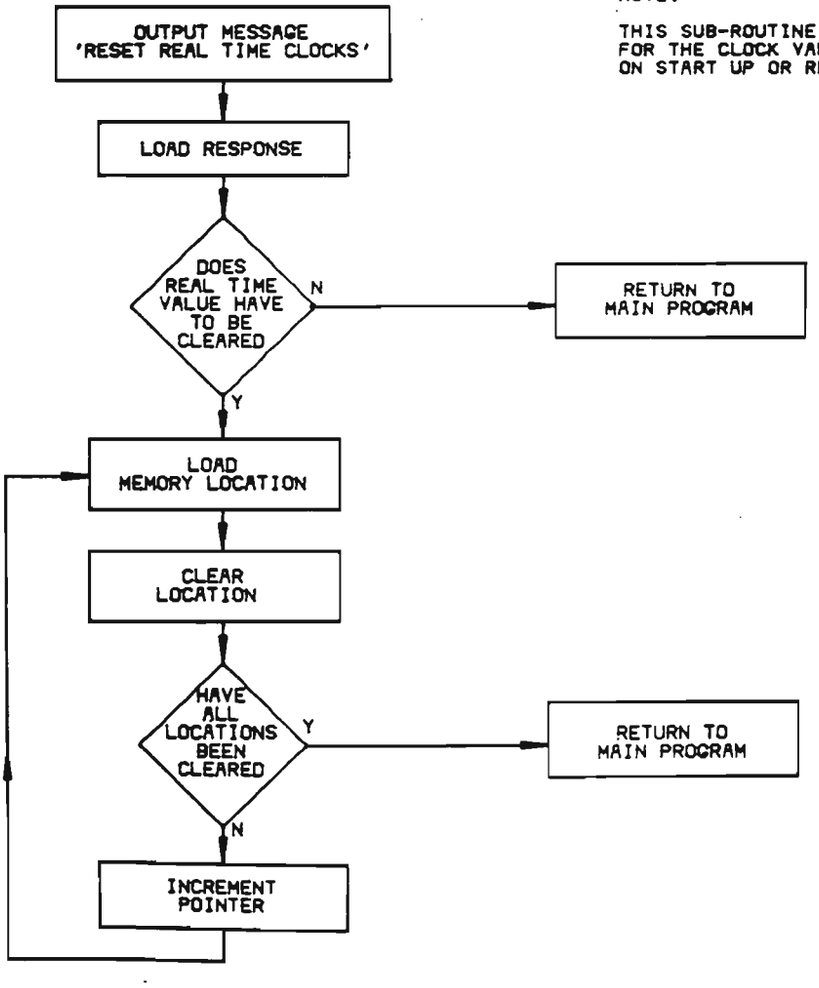
'FLUSH' SUB-ROUTINE - APPENDIX A
(AN INTERACTIVE SUB-ROUTINE)



NOTE:

THIS SUB-ROUTINE CLEARS ANY HALTS PUT IN PROGRAM WHILE DEBUGGING ERRORS, AFTER PROMPTING FOR AN ANSWER.

'TRSET' SUB-ROUTINE - APPENDIX A
(AN INTERACTIVE SUB-ROUTINE)



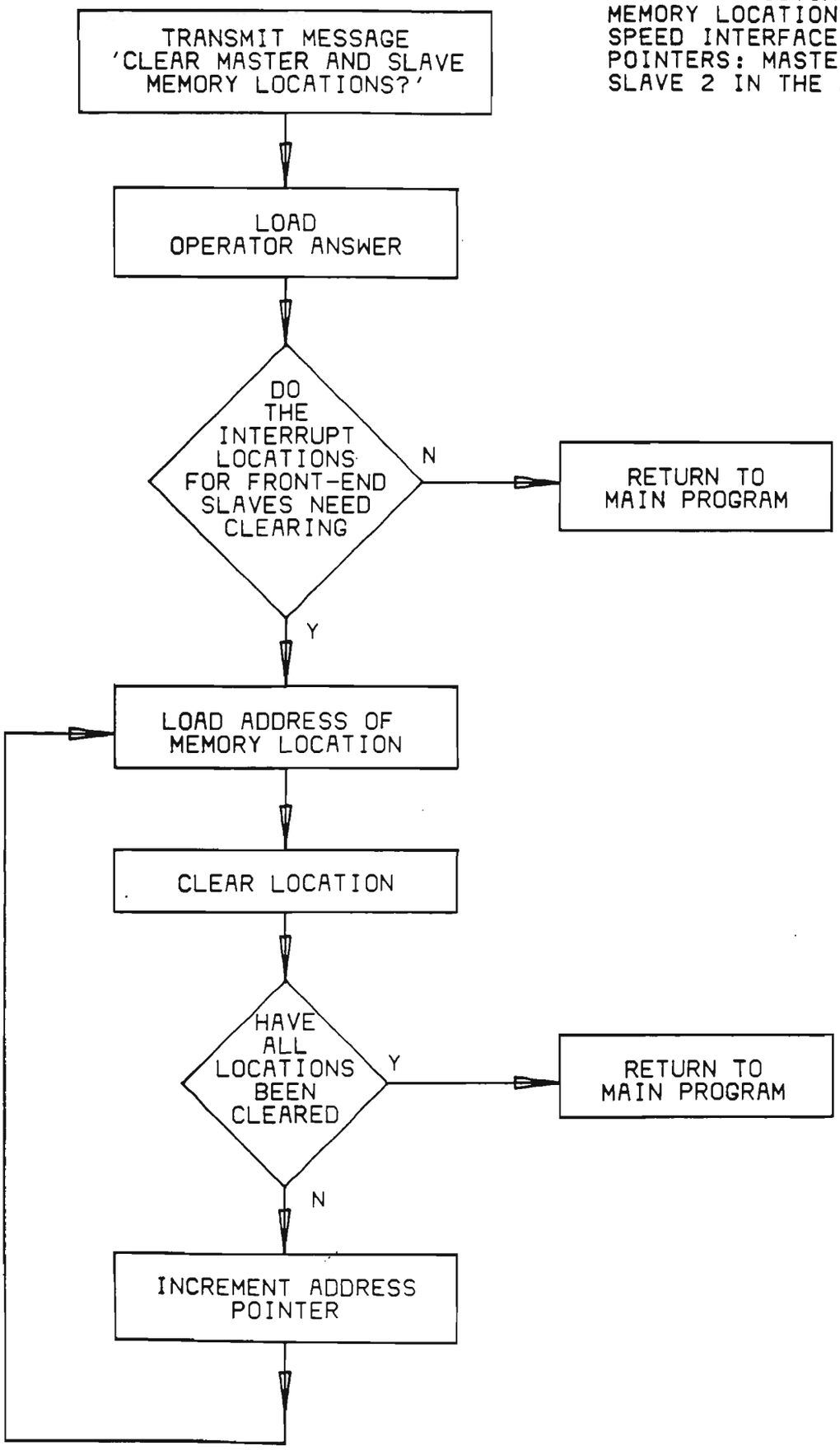
NOTE:

THIS SUB-ROUTINE CLEARS THE MEMORY LOCATION FOR THE CLOCK VALUES IN THE FRONT-END ON START UP OR REQUEST.

'MRSET' SUB-ROUTINE - APPENDIX A
(AN INTERACTIVE SUB-ROUTINE)

NOTE:

THIS SUB-ROUTINE CLEARS THE MEMORY LOCATIONS FOR THE HIGH SPEED INTERFACE INTERRUPT POINTERS: MASTER, SLAVE 1 AND SLAVE 2 IN THE FRONT-END.



```
 END PASS  1
PACE ASSEMBLER REV-A      20 DEC 76                PAGE    1
PACE  FTND CTRLPRGM 17/10/78


    1                ;
    2                ;
    3                        .TITLE PACE,'FTND CTRLPRGM 17/10/78'
    4 0000                   .ASECT
    5                        .NOBAS
    6                ;
    7                ;****************INITIAL COMNDS.*********************
    8                ;
    9                ;
   10                ;*********************************************************
   11                ;*                                                       *
   12                ;*         TYPE Y IF TIMES ARE TO BE CLEARED             *
   13                ;*              N "    "     " NOT TO BE   "           *
   14                ;*                                                       *
   15                ;*          "   Y "  MASTER $ SLAVES MEM. LOCS.  " .*
   16                ;*          "   N "  NOT   CLEARED                      *
   17                ;*                                                       *
   18                ;*                                                       *
   19                ;*    TYPE Y IN ANSWER TO FLUSH ! IF REQD. N IF NOT *
   20                ;*    Y LEAVES BREAK PTS.  N  CLEARS THEM            *
   21                ;*                                                       *
   22                ;*********************************************************
   23                ;
   24                ;
   25                ;
   26                ;*********************************************************
   27                ;*                  17/10/78  VERSION ************  *
   28                ;*   FRONTENDE DEV. CONTROL PROGRAM * "DEVDB".SRC   *
   29                ;* FRONTEND (DEBUG) CONTROL OF :-   *        .LM    *
   30                ;*      1 BINARY LOADER             *        .MP    *
   31                ;*      2 BINARY CORE COPIER          ************  *
   32                ;*      3 BOOTSTRAP LOADER                          *
   33                ;*      4 BOOTSTRAP CORE COPIER                     *
   34                ;*      5 ANY USER PROGRAMS                         *
   35                ;*      6 CONTROL OF SLAVE MICROS.                  *
   36                ;*      7 STKFUL INTERRUPTS.       ' IEN1 '         *
   37                ;*       OVERFLOW PRINTS  FS <PC>                   *
   38                ;*       UNDERFLOW  "     ES <PC>                   *
   39                ;*      8 REAL TIME INTERRUPT CLOCK 'IEN2'          *
   40                ;*      9 MASTER INTRPTS             'IEN3'         *
   41                ;*      10 OTHER SLAVES             'IEN4 OR 5'     *
   42                ;*                                                  *
   43                ;*    DESIGNED FOR PACE DEVELOPMENT SYSTEM          *
   44                ;*      AND IS CALLED FROM DISC.  IN THE NORMAL     *
   45                ;*      WAY    @DEV(DATE).MP ( IE. 6TH. USE 6 ).MP  *
   46                ;*      ASSEMBLY LISTING TIME APPROX. 4HRS - 35MINS.*
   47                ;*        BY TELETYPE. ASR ' 110 BAUD',             *
   48                ;*        OR BY TELETYPE 43 ' 300 BAUD '-2HR 35MINS *
   49                ;*                                                  *
   50                ;*                                                  *
   51                ;*       LIST SUPRSSD. AT END OF RAM LOCS APRX 250  *
   52                ;*         TO COMNDS SECTION APRX. 500              *
   53                ;*       &AGAIN AT END OF REAL TIME SUB TO STKFL SB.*
   54                ;*                                                  *
   55                ;*********************************************************
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE    3
PACE  FTND CTRLPRGM 17/10/78
FRNTND COMNDS


                .PAGE 'FRNTND COMNDS'
               ;*******************************************************
               ;*            PART B DEBDB.SRC                         *
               ;*            DEBUG FOR FRONTEND DEV.     SYSTEM.      *
               ;*                                                     *
               ;*                17/10/78                             *
               ;*       USES RAM FROM 0000 TO 0BFF  WITH SOME GAPS    *
               ;*                                                     *
               ;*                                                     *
               ;*             COMMANDS.....                           *
               ;*                                                     *
               ;*  A XX,DDC,YY,..] ALTER LOC XX TO DD                 *
               ;*                           (LOC XX+1 TO YY,ETC.)     *
               ;*                                                     *
               ;*  P XX          PRINT CONTENTS OF XX                 *
               ;*  P XX,YY       PRINT CONTENTS OF XX TO YY           *
               ;*                                                     *
               ;*  L N,XXC,YY,...]  LOAD REGISTER N WITH XX           *
               ;*                                   (N+1 WITH YY,ETC) *
               ;*                                                     *
               ;*      N=  0=> AC0          4=> FLAGS                 *
               ;*          1=> AC1          5=> TOP OF STACK          *
               ;*          2=> AC2          6=> PROGRAM COUNTER       *
               ;*          3=> AC3                                    *
               ;*                                                     *
               ;*  R     TYPE REGISTERS IN ABOVE ORDER                *
               ;*                                                     *
               ;*  I DD,XX,YY     MOVE DATA IN XX TO YY UP ONE        *
               ;*                  THEN INSERT DD IN LOCN. XX         *
               ;*                                                     *
               ;*  H            REMOVE ALL BREAKPOINTS                *
               ;*  H N          REMOVE BREAKPOINT N (N=1 TO 4)        *
               ;*  H N,XX       SET BREAKPOINT N AT LOCN. XX          *
               ;*                                                     *
               ;*  G            EXECUTE FROM PROGRAM COUNTER          *
               ;*  G XX         EXECUTE FROM LOCN. XX                 *
               ;*                                                     *
               ;*    " SUBSTITUTE","SUB",OR "CONTROL Z"               *
               ;*'CONTROL Z'TRANSMITS COMMANDS TO THE TRANS/REC.      *
               ;* CHANNEL AS SELECTED BY "CONTRL Q",  "DC1"          *
               ;*   TERMINATED BY " EOT ","CONTROL D"                 *
               ;* THE ABOVE IS ANOTHER METHOD OF COMMUNICATION        *
               ;*    WITH SYSTEM ELEMENTS.                            *
               ;*                                                     *
               ;*  D XX,YY      DUMP LOCS XX TO YY IN BINARY          *
               ;*                                                     *
               ;*    TEST SECOND CHAR. AFTER "C" FOR CHKSM REQST.     *
               ;*                                                     *
               ;*                                                     *
               ;*  C XX,YY      COPY LOCS XX TO YY IN ASCII(HEX)      *
               ;* CS XX,YY      IS A REQUEST FOR A CHECKSUM           *
               ;*                                                     *
               ;*  B            LOAD BINARY TAPE                      *
               ;*                                                     *
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE    4
PACE  FTND CTRLPRGM 17/10/78
FRNTND COMNDS


  110             ;*  S           LOAD BOOTSTRAP TAPE  [ASCII(HEX)]   *
  111             ;*                                                   *
  112             ;*  W           TYPE BREAKPOINT LOCATIONS            *
  113             ;*                                                   *
  114             ;*  X           CHANGE I/O MODE TO HEX               *
  115             ;*                                                   *
  116             ;*  O           CHANGE I/O MODE TO OCT               *
  117             ;*                                                   *
  118             ;*  T        REAL TIME CLOCK IN HRS:MINS,SEC.T'THS   *
  119             ;*                                                   *
  120             ;*      T=F.E. TIME,T1=SLAVE1 TIME,T2=SLAVE2 TIME    *
  121             ;*                                                   *
  122             ;*               100HOURS MAX.  IE. 99:59,59.9       *
  123             ;*                                                   *
  124             ;*              CHANGE THE I/O TRANS/REC. CHANNEL    *
  125             ;*       "CONTROL Q" N, "DC1" N                      *
  126             ;*             N =  CHANNEL NUMBER =1(MAIN CN'TL)    *
  127             ;*                                  "  =2(FRT'ED-MASTR) *
  128             ;*                                  "  =3(FRT'ED-SLVE2) *
  129             ;*                                  "  =4(FRT'ED-SLVE1) *
  130             ;*   NOTE:-                                          *
  131             ;*    'TRANS/REC.'CAN BE ANY DEVICE'110'TO'9600'BAUD*
  132             ;*      SUCH AS A VDU,TTY OR A HIGH SPEED SLAVE.     *
  133             ;*                                                   *
  134             ;*****************************************************
  135             ;
  136             ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE    5
PACE  FTND CTRLPRGM 17/10/78
POINTERS


  137                     .PAGE 'POINTERS'
  138           ;
  139           ;
  140      0003 A BIT0    = 3  ;  BOC AC0(BIT 0)
  141      0002 A POS     = 2  ; BOC IF AC0 POSITIVE
  142      000F A JC14    = 15
  143      000F A RFLG    = 15 ; TELETYPE INPUT FLAG
  144      000D A PFLG    = 13 ; TELETYPE OUTPUT FLAG
  145      000D A READR   = 13
  146      000F A SENDP   = 15
  147      0000 A AC0     = 0
  148      0000 A AC0     = 0
  149      0001 A AC1     = 1
  150      0002 A AC2     = 2
  151      0003 A AC3     = 3
  152      0001 A ZRO     = 1
  153      0005 A NZRO    = 5
  154      000B A NEG     = 11
  155      0001 A C1      = 1  ;  AC0  =  0
  156      0002 A C2      = 2 ; AC0 POS. (BIT 15 = 0)
  157      0000 A STKFUL = 0
  158      000B A NSIGN  = X'B
  159      0001 A IEN1    = 1
  160      0002 A IEN2    = 2
  161      0003 A IEN3    = 3
  162      0004 A IEN4    = 4
  163      0005 A IEN5    = 5
  164      0009 A IEN     = 9
  165      0020 A STSAV   =SAV0      ; START OF REGISTER FILE
  166      0026 A ENSAV   =PC        ; END OF REGISTER FILE
  167      002A A BKPNTS  =HLOC      ; START OF BREAKPOINT TABLE
  168      001F A 0000    =X'1F   ; RETURN JUMP FOR LOADERS ETC.
  169      0040 A TIMTS   =HRSMIN
  170      003C A TIMTS1  =HRMIN1
  171      003E A TIMTS2  =HRMIN2
  172      0000 A TTYOUT  = 0
  173      0001 A TTYTAP  =  1
  174      0002 A TTYON   = 2
  175      0005 A TTYGET  =  5
  176      0003 A TTYOFF  = 3
  177      9800 A ADDTTY = X'9800
  178           ;
  179           ;
  180           ;  BASE PAGE ADDRESSES
  181           ;
  182           ;  SHIFT INTERRUPT POINTERS
  183           ;     IF SPLIT BASE PAGE.
  184           ;
  185 0000              .=X'0
  186 0000 9801 A AAAA:   JMP @STARTI ; JUMP TO DEBUG INT'N. LOC'N. JMP.
  187 0001 00BD A STARTI: .WORD START
  188           ;
  189           ;  INTRPT SUB. LOCS.
  190           ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE    6
PACE  FTND CTRLPRGM 17/10/78
POINTERS


  191 0002 05B4 A          .WORD STKINT  ; STACK FULL
  192 0003 0170 A          .WORD RLTIME  ; REAL TIME CLOCK
  193 0004 06BB A          .WORD MSTRIT ;INTERRUPT #3 (MASTER INT. PTR.)
  194 0005 0871 A          .WORD SLV1IN ; INTERRUPT #4 (SLAVE1  "   "  )
  195 0006 09EF A          .WORD SLV2IN ; INTERRUPT #5 (SLAVE2  "    "  )
  196 0007 0026 A          .WORD PC     ; LEVEL ZERO PC LOCATION
  197 0008 1800 A          JMP 0        ; FAKE AN INITIALIZE
  198                ;
  199                ;   RS 232 ADDRESSES
  200                ;
  201 0009 9800 A TTYWD1: .WORD ADDTTY ;  SHIFT LOCATION TO
  202 000A 9820 A TTYWD2: .WORD ADDTTY+020 ; SUIT RELATIVE ADD. IN
  203                ;                                TTYFX SUBROUTINE
  204 000B 9900 A TTYWD3: .WORD ADDTTY+0100 ;     SLAVE2-"990X"
  205 000C 9880 A TTYWD4: .WORD ADDTTY+080  ;     SLAVE1-"988X"
  206 000D 0000 A SOFTDFT: .WORD 0000   ; * DO  NOT INSERT ANY PROG AT LOC.  "X
  207                ;                             DUE TO SYSTEM WILL NOT LOAD IT
  208                ;                                IT STAYS AS X'D0B2
  209                ;
  210                ;
  211 000E                 .= X'10
  212 0010 02D0 A XPUTW:  .WORD PUTW    ;****FOR CNTRL PNL IN POW. SYSTEM***
  213 0011 013F A XCRLF:  .WORD CRLF  ;****DO NOT MOVE*****
  214 0012 02B4 A XGECHO: .WORD GECHO ;***CALLED BY LOC'N. IN STKFL.**
  215                ;
  216                ;    SPARE SPACE HERE X'13 TO X'1F
  217                ;
  218                ;   RAM LOCATIONS ALTER FOR SPLIT TO X'FFBO APROX.
  219                ;
  220       0020 A SHIFT   = X'20
  221       0020 A SAV0    =0+SHIFT  ; REGISTER STORES
  222       0021 A SAV1    =1+SHIFT
  223       0022 A SAV2    =2+SHIFT
  224       0023 A SAV3    =3+SHIFT
  225       0024 A FLAGS   =4+SHIFT ; SAVE FLAGS
  226       0025 A STACK   =5+SHIFT ; TOP OF STACK
  227       0026 A PC      =6+SHIFT ; PROGRAM COUNTER
  228       0027 A POINTER =7+SHIFT ; POINTER TO BEGIN
  229       0028 A CWRD    =8+SHIFT ; INSERTION DATA
  230       0029 A DATA    =9+SHIFT ; INSERTION DATA
  231       002A A HLOC    =10+SHIFT  ; 4 BREAKPOINT LOCATIONS
  232       002E A HDATA   =14+SHIFT ; 4 BREAKPOINTED INSTRUCTIONS
  233       0033 A MODE    =19+SHIFT    ; CONTROLS I/O FORMAT
  234       0034 A PCG     =20+SHIFT    ; PC FOR GO SUBROUTINE
  235       0035 A TMP0    =21+SHIFT
  236       0036 A TMP1    =22+SHIFT
  237       0037 A TMP2    =23+SHIFT
  238       0038 A TMP3    =24+SHIFT
  239       0039 A TTYCHN  =25+SHIFT
  240       003A A TTYADD  =26+SHIFT
  241                ;
  242                ; REAL TIME CLOCK LOCS.
  243                ;
  244                ;   SLAVE 1 & 2 TIMES
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE    7
PACE  FTND CTRLPRGM 17/10/78
POINTERS


  245                ;
  246       003C A HRMIN1  =28+SHIFT  ; SLAVE1 TIME
  247       003B A SETEN1  =27+SHIFT
  248       003E A HRMIN2  =30+SHIFT  ; SLAVE2   "
  249       003D A SETEN2  =29+SHIFT
  250                ;
  251                ;
  252                ;   F.E.  TIMES
  253                ;
  254       0040 A HRSMIN  =32+SHIFT  ; HEX VALUE OF TIME
  255       003F A SECTEN  =31+SHIFT  ;  "    "    "   "
  256       0041 A TENTHS  =33+SHIFT  ; TENTHS OF SECS.
  257       0042 A SECOND  =34+SHIFT  ; SECONDS
  258       0043 A MINUTS  =35+SHIFT  ;  MINUTES
  259       0044 A HOURS   =36+SHIFT  ;  HOURS
  260       0045 A RETADT  =37+SHIFT  ;RETURN ADDS. FOR CLOCK
  261       0046 A RTCWRD  =38+SHIFT
  262       0047 A RTREG   =39+SHIFT ; TO 42+SHIFT
  263                ;
  264                ;   STKFUL INTERRUPT LOCATIONS
  265                ;
  266       004B A $SAV0   =43+SHIFT
  267       004C A $SAV1   =44+SHIFT
  268       004D A $SAV2   =45+SHIFT
  269       004E A $SAV3   =46+SHIFT
  270       004F A $RETA   =47+SHIFT
  271       0050 A $TEMP   =48+SHIFT
  272       0051 A SOFTST =49+SHIFT ; TO STKEND (INCREASE STK BY =X'1000)
  273       0060 A STKEND =64+SHIFT  ; ALLOWS 4 OVFLS.( & =X'10FF )
  274       0061 A $STAK   =65+SHIFT ; TO 69+SHIFT
  275       0066 A $SWPTR =70+SHIFT
  276       0067 A $SPTR   =71+SHIFT
  277                ;
  278                ;
  279                ; MASTER INTRPT. LOC'NS.
  280                ;
  281       0068 A MSTREG =72+SHIFT ; TO 75+SHIFT
  282       006C A RETMST =76+SHIFT ;RET. ADDS.
  283                ;
  284       2000 A MMSGST =X'2000   ; 9K IN F.E. MSG. STRE AREA F.E.
  285                ;
  286       006D A MSOHCT =77+SHIFT
  287       006E A MSNSYN =78+SHIFT ; INIT. INT CNT. NON-SYN CHS.
  288       006F A MSTCHL =79+SHIFT ; TX. CHANNEL STORE LOC.
  289       0070 A MSTCMD =80+SHIFT ; CMND. STORE LOC.
  290       0071 A MSTADD =81+SHIFT ; MESG LOAD ADDS IN MSTR.
  291       0072 A MSTERR =82+SHIFT ; ERROR CODE STR.
  292       0073 A MSERAD =83+SHIFT ;   "    "    ADDS. PTR.
  293       0074 A MSGEND =84+SHIFT ; TO 85+SHIFT SPARE LOCS FOR MASTER
  294                ;
  295                ;
  296                ;   SLAVE1 INTRPT LOC'NS.
  297                ;
  298                ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE    8
PACE  FTND CTRLPRGM 17/10/78
POINTERS


  299      0076 A SL1REG =86+SHIFT ; TO 89+SHIFT
  300      007A A RETSL1 =90+SHIFT ; RET. ADDRESS
  301           ;
  302      2800 A S1MGST =X'2800    ; 10K IN F.E. MSG. STRE AREA SL1.
  303           ;
  304      007B A S1SHCT =91+SHIFT
  305      007C A S1NSYN =92+SHIFT ; INIT. INPT. COUNT OF NON-SYN CHS.
  306      007D A SL1CHL =93+SHIFT ; TX. CHNL. STORE LOC.
  307      007E A SL1CMD =94+SHIFT ; SLV1. CMD. STORE LOC.
  308      007F A SL1ADD =95+SHIFT ; MESG LD ADDS IN F.E.
  309      0080 A SL1ERR =96+SHIFT  ;ERROR CODE STRE.
  310      0081 A S1ERAD =97+SHIFT  ;  "     "   ADS. PTR.
  311      0082 A S1MEND =98+SHIFT ; TO 99+SHIFT SPRE LOCS. FOR SLV1
  312           ;
  313           ;
  314           ;
  315           ;    SLAVE2  INTRPT. LOC'NS.
  316           ;
  317           ;
  318      0084 A SL2REG =100+SHIFT    ; TO 103+SHIFT
  319      0088 A RETSL2 =104+SHIFT ; RET. ADDRESS
  320           ;
  321      2C00 A S2MGST =X'2C00    ; 11K IN F.E. MSG. STRE AREA SL2.
  322           ;
  323      0089 A S2SHCT =105+SHIFT
  324      008A A S2NSYN =106+SHIFT ; INIT. INPT. COUNT OF NON-SYN CHS.
  325      008B A SL2CHL =107+SHIFT ; TX. CHNL. STORE LOC.
  326      008C A SL2CMD =108+SHIFT ; SLV2. CMD. STORE LOC.
  327      008D A SL2ADD =109+SHIFT ; MESG LD ADDS IN F.E.
  328      008E A SL2ERR =110+SHIFT  ;ERROR CODE STRE.
  329      008F A S2ERAD =111+SHIFT  ;  "     "   ADS. PTR.
  330      0090 A S2MEND =112+SHIFT ; TO 113+SHIFT SPRE LOCS. FOR SLV2
  331           ;
  332           ;
  333           ;
  334           ;
  335           ;
  336      0091 A LIMIT  =113+SHIFT    ; BASE PAGE STORAGE LIMIT
  337           ;
  338           ;
  339           ;
  340           ; SPACE HERE X'? TO X'FF
  341           ;
  342           ; MORE SPACE IS ACHIEVED BY
  343           ; SHIFTING START (BD) TO X'100
  344           ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   17
PACE  FTND CTRLPRGM 17/10/78
INTRPT. SUB. RLTIME-IEN2


  670                     .PAGE 'INTRPT. SUB. RLTIME-IEN2'
  671                ;************************************************
  672                ;*                                              *
  673                ;*    REAL TIME     SUBROUTINE                  *
  674                ;*    100HR. CLOCK BY INTERRUPTS OF 100MILLISEC*
  675                ;*         INTERRUPT 2 -IEN2 (NIR2)             *
  676                ;*                                              *
  677                ;************************************************
  678                ;
  679                ;
  680 0170 5C00 A RLTIME: RCPY 0,0   ; NOP
  681                ;
  682                ;
  683                ;
  684      0001 A         .LIST 01
  685                ;
  686                ;
  687                ;
  688                ;
  689 0171 D047 A         ST AC0,RTREG
  690 0172 D448 A         ST AC1,RTREG+1
  691 0173 D849 A         ST AC2,RTREG+2
  692 0174 DC4A A         ST AC3,RTREG+3
  693 0175 6400 A         PULL AC0
  694 0176 D045 A         ST AC0,RETADT
  695 0177 5C00 A         RCPY AC0,AC0  ; WAS PUSHF
  696 0178 0400 A         CFR AC0       ; MOVE FLAGS TO AC0
  697 0179 A944 A         AND AC0,TMASK ; MASK OLD INT. STATUS
  698 017A A544 A         OR  AC0,TINTST ; OR IN NEW "     "
  699 017B 0800 A         CRF AC0
  700 017C C148 A         LD 0,CONSTL
  701 017D 5F00 A         RCPY 0,3
  702 017E C03F A         LD 0,SECTEN     ; SECONDS/TENTHS OF SECS.
  703 017F 8B00 A         DECA 0,(3)      ; USE BCD INSTRUCTION 'DECA'
  704                ;                       FOR FORMATTING WORDS
  705 0180 9140 A         SUBB 0,CONST1     ;SUBTRACT CONST.
  706 0181 E144 A         ADD 0,ONET       ;  TWO'S COMPLEMENT
  707 0182 A947 A         AND 0,MSKT
  708 0183 4B0C A         BOC NSIGN,ICMIN1 ; 60 REACHED. RESET TO 0
  709 0184 D03F A         ST 0,SECTEN
  710 0185 A946 A         AND AC0,MSKT3
  711 0186 F13B A         SKNE AC0,ONESEC
  712 0187 1904 A         JMP INCSEC
  713 0188 C041 A INCTEN: LD AC0,TENTHS      ; INCREMENT
  714 0189 E13C A         ADD AC0,ONET       ; TENTHS OF A SECOND
  715 018A D041 A         ST AC0,TENTHS      ; COUNTER.
  716                                      ; MAX  65,536 OR 1.822HRS
  717 018B 194C A         JMP SL1TUD  ; JMP TO SL1 TIME UPDATE.
  718 018C C042 A INCSEC: LD AC0,SECOND
  719 018D E138 A         ADD AC0,ONET   ; INCREMENT SECOND
  720 018E D042 A         ST AC0,SECOND  ; COUNTER (MAX 65,536)
  721                                    ;         ( OR 18.22HRS.)
  722 018F 19F8 A         JMP INCTEN
  723 0190 C043 A ICMIN1: LD AC0,MINUTS ; INCREMENT MINUTE
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE   18
PACE  FTND CTRLPRGM 17/10/78
INTRPT. SUB. RLTIME-IEN2


  724 0191 E134 A          ADD AC0,ONET  ; COUNTER (MAX 65,536)
  725                                    ;        ( OR 48.27DAYS.)
  726 0192 D043 A          ST AC0,MINUTS
  727 0193 C135 A          LD 0,ZEROT  ;RESET SECOND/TENTHS
  728 0194 D03F A          ST 0,SECTEN   ; AND INCR. MINS & HRS.
  729 0195 C13A A          LD 0,CNST2L
  730 0196 5F00 A          RCPY 0,3
  731 0197 C046 A          LD 0,HRSMIN
  732 0198 8B00 A          DECA 0,(3)
  733 0199 9135 A          SUBB 0,CONST2
  734 019A E12B A          ADD 0,ONET
  735 019B 2010 A          ROL 0,8,0  ;TEST FOR SIXTY MINS.
  736 019C 4B03 A          BOC NSIGN,RESTT
  737 019D 2010 A          ROL 0,8,0  ; RESTORE BITS IN AC0
  738 019E D046 A STORHM:  ST 0,HRSMIN
  739 019F 1938 A          JMP SL1TUD; JMP TO SLAVES TIME UPDATE
  740 01A0 430D A RESTT:   BOC BIT0,REST1
  741 01A1 C125 A          LD 0,ONET2     ; RESET HRS-MINS LOCATION
  742 01A2 E046 A          ADD 0,HRSMIN
  743 01A3 A927 A          AND 0,MSKT2
  744 01A4 D046 A          ST AC0,HRSMIN
  745 01A5 C044 A          LD AC0,HOURS  ; INCREMENT HOUR COUNT.
  746 01A6 E11F A          ADD AC0,ONET  ; MAX 65,536 OR 7.5 YRS.
  747 01A7 D044 A          ST AC0,HOURS
  748 01A8 19F6 A          JMP STORHM+1
  749 01A9 5C40 A REST2:   RCPY 1,0
  750 01AA A921 A          AND 0,MSKT3
  751 01AB E11C A          ADD 0,ONET3
  752 01AC 2010 A          ROL 0,8,0
  753 01AD 19F0 A          JMP STORHM
  754 01AE 5D00 A REST1:   RCPY 0,1
  755 01AF A91D A          AND 0,MSKT4    ; WATCH ADDRESSING LATER
  756 01B0 F11C A          SKNE 0,MSKT4
  757 01B1 19F7 A          JMP REST2
  758 01B2 5C00 A          RCPY 0,0
  759 01B3 19ED A          JMP RESTT+1
  760 01B4 5C00 A RTINXT:  RCPY AC0,AC0  ; EXIT SUBROUTINE. WAS PULLF
  761 01B5 C045 A          LD AC0,RETADT ; RESTORE INTR RETURN ADD'S
  762 01B6 6000 A          PUSH AC0
  763 01B7 C047 A          LD AC0,RTREG
  764 01B8 C448 A          LD AC1,RTREG+1 ; RESTORE REGS.
  765 01B9 C849 A          LD AC2,RTREG+2
  766 01BA CC4A A          LD AC3,RTREG+3
  767 01BB 3200 A          PFLG IEN2     ; CLEAR INT. LATCH
  768 01BC 3280 A          SFLG IEN2
  769 01BD 7C00 A          RTI
  770              ;
  771              ;    ALLOW SPACE FOR TIMER INCR'TS.
  772              ;
  773 01BE 820B A TMASK:   .WORD X'820B ;STACK FULL CAN INT.
  774 01BF 8203 A TINTST:  .WORD X'8203 ; IEN1 = BIT "1"
  775 01C0 4010 A CONST:   .WORD X'4010
  776 01C1 4000 A CONST1:  .WORD X'4000
  777 01C2 0090 A ONESEC:  .WORD X'0090
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   19
PACE  FTND CTRLPRGM 17/10/78
INTRPT. SUB. RLTIME-IEN2


  778 01C3 98A2 A RSON:   .WORD X'98A2   ;KEYBOARD LOCATIONS
  779 01C4 98A5 A RSGET:  .WORD X'98A5
  780 01C5 01C0 A CONSTL: .WORD CONST
  781 01C6 0001 A ONET:   .WORD X'1
  782 01C7 0100 A ONET2:  .WORD X'0100
  783 01C8 0010 A ONET3:  .WORD X'0010
  784 01C9 0000 A ZEROT:  .WORD X'0
  785 01CA FFF0 A MSKT:   .WORD X'FFF0
  786 01CB FF00 A MSKT2:  .WORD X'FF00
  787 01CC 00F0 A MSKT3:  .WORD X'00F0
  788 01CD 000F A MSKT4:  .WORD X'000F
  789 01CE 0041 A CONSTM: .WORD X'0041
  790 01CF 0040 A CONST2: .WORD X'0040
  791 01D0 01CE A CNST2L: .WORD CONSTM  ; HRSMIN CONST. 9999-0040
  792 01D1 199D A ERROR9: JMP ERROR8
  793 01D2 1995 A SENDB:  JMP SENDB1
  794 01D3 1996 A PUTWI:  JMP PUTWI1
  795 01D4 1998 A JUMP1:  JMP JUMP11
  796 01D5 1998 A EXITP:  JMP EXITP1
  797             ;
  798             ;
  799             ;   UPDATE TIME SUB FOR CURRENT REC'D. TIMES
  800             ;     FOR SLAVES  SEE   FRONT PAGE DIRECT'NS.
  801             ;
  802 01D6 003B A SLTPT1: .WORD SETEN1
  803 01D7 003D A SLTPT2: .WORD SETEN2
  804             ;
  805 01D8 C9FD A SL1TUD: LD AC2,SLTPT1   ; SLAVE1 TIME UPDATE
  806 01D9 1503 A         JSR SLATUD
  807 01DA C9FC A SL2TUD: LD AC2,SLTPT2   ;      2   "    "
  808 01DB 1501 A         JSR SLATUD
  809 01DC 19D7 A         JMP RTINXT
  810             ;
  811 01DD C1E7 A SLATUD: LD AC0,CONSTL  ; SLAVES TIME UPD. SUB.
  812 01DE 5F00 A         RCPY AC0,AC3
  813 01DF C200 A         LD AC0,(AC2)
  814 01E0 8B00 A         DECA AC0,(AC3)
  815 01E1 91DF A         SUBB AC0,CONST1
  816 01E2 E1E3 A         ADD AC0,ONET
  817 01E3 A9E6 A         AND AC0,MSKT
  818 01E4 4B02 A         BOC NSIGN,SLVSEC
  819 01E5 D200 A         ST  AC0,(AC2)  ; TENTHS UPDATE
  820 01E6 8000 A         RTS 0
  821             ;
  822 01E7 C1E1 A SLVSEC: LD AC0,ZEROT
  823 01E8 D200 A         ST AC0,(AC2)    ; RESET SECTENTHS
  824 01E9 C1E6 A         LD AC0,CNST2L
  825 01EA 5F00 A         RCPY AC0,AC3
  826 01EB 7A01 A         AISZ AC2,1  ; RESET AC2 FOR HRSMIN1
  827 01EC C200 A         LD AC0,(AC2) ; LOAD   LOCN.
  828 01ED 8B00 A         DECA AC0,(AC3)
  829 01EE 91E0 A         SUBB AC0,CONST2
  830 01EF E1D6 A         ADD AC0,ONET
  831 01F0 2010 A         ROL 0,8,0
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   20
PACE  FTND CTRLPRGM 17/10/78
INTRPT. SUB. RLTIME-IEN2


  832 01F1 4B03 A          BOC NSIGN,SLHRIC  ; TEST FOR 60 MINS
  833 01F2 2010 A          ROL 0,8,0
  834 01F3 D200 A          ST AC0,(AC2)  ; STORE NEW MINS
  835 01F4 8000 A          RTS 0
  836                ;
  837 01F5 4305 A SLHRIC:  BOC BIT0,SLHRI1 ; SLAVE HR INCR.
  838 01F6 C1D0 A          LD AC0,ONET2
  839 01F7 E200 A          ADD AC0,(AC2)
  840 01F8 A9D2 A          AND AC0,MSKT2
  841 01F9 D200 A          ST AC0,(AC2)  ; STORE HR INCR'T.
  842 01FA 8000 A          RTS 0
  843                ;
  844 01FB 5D00 A SLHRI1:  RCPY 0,1
  845 01FC A9D0 A          AND 0,MSKT4
  846 01FD F1CF A          SKNE AC0,MSKT4
  847 01FE 1905 A          JMP SLHRST
  848 01FF C1C7 A          LD AC0,ONET2
  849 0200 E200 A          ADD AC0,(AC2)
  850 0201 A9C9 A          AND AC0,MSKT2
  851 0202 D200 A          ST AC0,(AC2)
  852 0203 8000 A          RTS 0
  853                ;
  854 0204 5C40 A SLHRST:  RCPY 1,0    ; SLAVE HR RESET
  855 0205 A9C6 A          AND AC0,MSKT3
  856 0206 E1C1 A          ADD AC0,ONET3
  857 0207 2010 A          ROL 0,8,0
  858 0208 D200 A          ST AC0,(AC2)   ; RESET HRS.
  859 0209 8000 A          RTS 0
  860                ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   24
PACE  FTND CTRLPRGM 17/10/78
INSTRUCTION EXECUTION


  952                       .PAGE
  953                ;**************************************************
  954                ;*                                                *
  955                ;*      T: PRINTS HRS:MINS,SEC.TENTHS OF SECS.    *
  956                ;*           MAX.  99:59  , 59.9   +.1SEC -REAL T.*
  957                ;*             INTERRUPT CONTROLLED CLOCK 'IEN2'  *
  958                ;*                                                *
  959                ;**************************************************
  960                ;
  961                ;
  962 023A 1579 A TIME:    JSR GECHO         ; WAIT FOR CR,LF
  963 023B 190B A          JMP FETIME
  964 023C F146 A          SKNE AC0,SLV1TM
  965 023D 190C A          JMP S1TIME
  966 023E F145 A          SKNE AC0,SLV2TM
  967 023F 190D A          JMP S2TIME
  968 0240 5340 A          LI AC3,TIMTS
  969 0241 1900 A FEPRTM:  JMP .+1
  970 0242 149A A          JSR TTYFX
  971 0243 5DC0 A          RCPY AC3,AC1
  972 0244 79FF A          AISZ AC1,-1
  973 0245 D446 A          ST AC1,RTCWRD
  974 0246 1909 A          JMP TIME2
  975 0247 152D A FETIME:  JSR RTMESG
  976 0248 5340 A          LI AC3,TIMTS
  977 0249 19F7 A          JMP FEPRTM
  978 024A 152A A S1TIME:  JSR RTMESG
  979 024B 533C A          LI AC3,TIMTS1
  980 024C 19F4 A          JMP FEPRTM
  981 024D 1527 A S2TIME:  JSR RTMESG
  982 024E 533E A          LI AC3,TIMTS2
  983 024F 19F1 A          JMP FEPRTM
  984 0250 5102 A TIME2:   LI AC1,2
  985 0251 6100 A          PUSH AC1
  986 0252 CB00 A          LD AC2,(AC3)
  987 0253 5020 A          LI AC0,X'20
  988 0254 5105 A          LI AC1,5
  989 0255 1537 A TSPACE:  JSR RTTS
  990 0256 79FF A          AISZ AC1,-1
  991 0257 19FD A          JMP TSPACE
  992 0258 5104 A          LI AC1,4
  993 0259 F534 A PIX1:    SKNE AC1,TWOSP
  994 025A 190B A          JMP SPTWO
  995 025B 5C80 A PIX2:    RCPY AC2,AC0
  996 025C 2A08 A          SHL AC2,4,0
  997 025D 2C18 A          SHR AC0,12,0
  998 025E 7830 A          AISZ  AC0,X'30
  999 025F 9D2C A          SKG AC0,RTNU9
 1000 0260 1901 A          JMP .+2
 1001 0261 7807 A          AISZ AC0,X'7
 1002 0262 152A A          JSR RTTS
 1003 0263 79FF A          AISZ AC1,-1
 1004 0264 19F4 A          JMP PIX1
 1005 0265 1903 A          JMP SPTWO1
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   25
PACE  FTND CTRLPRGM 17/10/78
INSTRUCTION EXECUTION


 1006 0266 5020 A SPTWO:  LI AC0,X'20
 1007 0267 1525 A         JSR RTTS
 1008 0268 19F2 A         JMP PIX2
 1009 0269 6500 A SPTWO1: PULL AC1
 1010 026A 79FF A         AISZ AC1,-1
 1011 026B 1902 A         JMP NEXTWD
 1012 026C 5100 A         LI AC1,0
 1013 026D 1906 A         JMP TIMEXT
 1014 026E 7BFF A NEXTWD: AISZ AC3,-1
 1015 026F 1900 A         JMP .+1
 1016 0270 5CC0 A         RCPY AC3,AC0
 1017 0271 F046 A         SKNE AC0,RTCWRD
 1018 0272 19DD A         JMP TIME2
 1019 0273 5100 A         LI AC1,0
 1020 0274 19C4 A TIMEXT: JMP EXITC
 1021 0275 6DC0 A RTMESG: RXCH AC3,AC1
 1022 0276 1500 A         JSR .+1          ; SET STACK WITH NXT. LOC.
 1023 0277 6700 A RTMSRF: PULL AC3         ; PUT INTO AC3
 1024 0278 ED16 A         ADD AC3,RTITLE   ; REAL MESSAGE REFERENCE
 1025 0279 C300 A RTMSCT: LD AC0,(AC3)     ;  "       "     COUNT
 1026 027A 4106 A         BOC 1,RETRNX
 1027 027B 2410 A         ROR 0,8,0
 1028 027C 1551 A         JSR SENDC
 1029 027D 2410 A         ROR 0,8,0
 1030 027E 154F A         JSR SENDC
 1031 027F 7B01 A         AISZ AC3,1
 1032 0280 19F8 A         JMP RTMSCT
 1033 0281 6F40 A RETRNX: RXCH AC1,AC3
 1034 0282 B000 A         RTS 0
 1035 0283 0031 A SLV1TM: .WORD X'31
 1036 0284 0032 A SLV2TM: .WORD X'32
 1037 0285 193F A SENDA:  JMP SEND
 1038 0286 1949 A PUTWA:  JMP PUTW
 1039 0287 196F A GETHXC: JMP GETHX
 1040 0288 192B A GECHOB: JMP GECHO
 1041 0289 1970 A GETHXB: JMP GETHXA
 1042 028A 19AD A ERRORB: JMP ERRORC
 1043 028B 0092 A TTYS1:  .WORD TTYS
 1044 028C 0039 A RTNU9:  .WORD X'39
 1045 028D 99FD A RTTS:   JMP @TTYS1
 1046 028E 0002 A TWOSP:  .WORD X'2
 1047 028F 0019 A RTITLE: .WORD RTMESS-RTMSRF
 1048 0290 2020 A RTMESS: .ASCII '  TIME IN HRS:MINS,SECS.TENTHS'
 1049 029F 0D0A A         .WORD X'0D0A
 1050 02A0 2020 A         .ASCII '             :    ,    .    '
 1051 02AF 0D0D A         .WORD X'0D0D
 1052 02B0 2020 A         .ASCII '     '
 1053 02B3 0000 A         .WORD 0
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   33
PACE  FTND CTRLPRGM 17/10/78
INSTRUCTION EXECUTION


 1321                       .PAGE
 1322                  ;*****************************************************
 1323                  ;*                                                   *
 1324                  ;*   CONTROL Z XXXXXX.../'EOT' TRANSMITS COMMANDS    *
 1325                  ;*  TO INTERFACE AS SELCTD. -"DC1 N"-TERM'TD BY EOT *
 1326                  ;*****************************************************
 1327                  ;
 1328 0379 1504 A CMDTRN: JSR GECMDI   ; ECHO ALL CHARACTERS
 1329 037A 19FB A         JMP EXIT1 ; (WITHOUT FURTHER PROCESSING)
 1330 037B 149A A         JSR TTYFX
 1331 037C 1516 A         JSR SENDR
 1332 037D 19FB A         JMP CMDTRN
 1333 037E 6300 A GECMDI: PUSH AC3
 1334 037F CC09 A         LD AC3,TTYWD1
 1335 0380 DC3A A         ST AC3,TTYADD
 1336 0381 6700 A         PULL AC3
 1337 0382 1511 A         JSR RECVN
 1338 0383 A913 A         AND AC0,H7F1
 1339 0384 41F9 A         BOC ZRO,GECMDI
 1340 0385 F111 A         SKNE AC0,H7F1
 1341 0386 19F7 A         JMP GECMDI
 1342 0387 F110 A         SKNE AC0,GALT1
 1343 0388 9910 A         JMP @ABORT1
 1344 0389 F110 A         SKNE AC0,EOT
 1345 038A 8000 A         RTS 0
 1346 038B 1501 A         JSR SENDQ
 1347 038C 8001 A         RTS 1
 1348 038D 6300 A SENDQ:  PUSH AC3
 1349 038E CC09 A         LD AC3,TTYWD1
 1350 038F DC3A A         ST AC3,TTYADD
 1351 0390 1502 A         JSR SENDR
 1352 0391 6700 A         PULL AC3
 1353 0392 8000 A         RTS 0
 1354 0393 9901 A SENDR: JMP @TMITS2
 1355 0394 9901 A RECVN: JMP @TMITR1
 1356 0395 0092 A TMITS2: .WORD TTYS
 1357 0396 0093 A TMITR1: .WORD TTYR
 1358 0397 007F A H7F1:   .WORD X'7F
 1359 0398 007D A GALT1:  .WORD X'7D
 1360 0399 032D A ABORT1: .WORD ABORT
 1361 039A 0004 A EOT:    .WORD X'4
 1362                  ;
 1363                  ;
 1364                  ;
 1365                  ;*****************************************************
 1366                  ;*                                                   *
 1367                  ;*    PRINT MEMORY CONTENTS OF XX TO YY              *
 1368                  ;*                                                   *
 1369                  ;*      P XX:YY  OR  P XX,YY  OR P XX                *
 1370                  ;*                                                   *
 1371                  ;*****************************************************
 1372                  ;
 1373 039B 15BF A PRINT:  JSR     RANGE          ; GET ADDRESS RANGE
 1374 039C 5100 A         LI AC1,0
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   34
PACE  FIND CTRLPRGM 17/10/78
INSTRUCTION EXECUTION


 1375 039D 6100 A          PUSH AC1
 1376 039E 149A A          JSR TTYFX
 1377 039F 15B3 A LINE:    JSR SCRLF ; NEW LINE : CR/LF FIRST
 1378 03A0 5EC0 A          RCPY    AC3,AC2
 1379 03A1 15B5 A          JSR PUTWY ;    TYPE ADDRESS
 1380 03A2 5020 A RTYP:    LI AC0,X'20           ; " "
 1381 03A3 15D3 A          JSR SENDI ;    SEND TWO BLANKS
 1382 03A4 CB00 A SWRD:    LD      AC2,(AC3)     ; TYPE OUT VALUE
 1383 03A5 15B1 A          JSR PUTWY
 1384 03A6 4F0B A          BOC RFLG,FIN  ; ATTEMPTED INPUT : STOP
 1385 03A7 FC28 A          SKNE    AC3,CWRD      ; CHECK IF DONE YET
 1386 03A8 1909 A          JMP     FIN           ; FINISHED
 1387 03A9 7B01 A          AISZ    AC3,1         ; INCREMENT ADDRESS
 1388 03AA 5CC0 A          RCPY AC3,AC0 ; CHECK FOR END OF LINE
 1389 03AB B90A A          SKAZ AC0,DO7
 1390 03AC 19F7 A          JMP     SWRD
 1391 03AD 6500 A          PULL AC1
 1392 03AE 6100 A          PUSH AC1
 1393 03AF 7900 A          AISZ  AC1,0  ; SKIP IF AC1=0 (REG.TYPE TEST)
 1394 03B0 19F3 A          JMP SWRD      ; CONTINUE LINE IF AC1=0
 1395 03B1 19ED A          JMP     LINE
 1396                ;
 1397 03B2 15A0 A FIN:     JSR SCRLF ; GIVE CR/LF WHEN FINISHED
 1398 03B3 6500 A          PULL AC1
 1399 03B4 5100 A          LI   AC1,0    ;  CLEAR STACK AND AC1
 1400 03B5 19C0 A          JMP     EXIT1         ; GO BACK TO PROMPT
 1401 03B6 0007 A DO7:     .WORD 7
 1402                ;
 1403                ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   52
PACE  FTND CTRLPRGM 17/10/78
STKFUL INTERRUPTS


 2030                       .PAGE 'STKFUL INTERRUPTS'
 2031              ;
 2032              ;*****************************************
 2033              ;*                                       *
 2034              ;*  STKFUL INTERRUPT SUBROUTINE          *
 2035              ;*                                       *
 2036              ;*****************************************
 2037              ;
 2038              ;
 2039              ; SAVE REGS. AND DETERMINE IF FULL/EMPTY.
 2040              ;
 2041 05B4 D04B A STKINT: ST ACO,$SAV0
 2042 05B5 D44C A         ST AC1,$SAV1
 2043 05B6 D84D A         ST AC2,$SAV2
 2044 05B7 DC4E A         ST AC3,$SAV3
 2045 05B8 6400 A         PULL ACO
 2046 05B9 D04F A         ST ACO,$RETA
 2047 05BA 401D A         BOC STKFUL,$FULL
 2048 05BB 6000 A         PUSH ACO
 2049 05BC 4018 A         BOC STKFUL,$AFULL   ; TEST IF ALMOST
 2050 05BD 6400 A           PULL ACO          ; FULL
 2051              ;
 2052              ;   STACK EMPTY. RESTORE FOUR WORDS.
 2053              ;
 2054 05BE 5104 A         LI AC1,4          ; STACK IS EMPTY
 2055 05BF AC67 A $EMP:   DSZ $SPTR  ; IF EXECUTED FIRST IT
 2056 05C0 A067 A         LD ACO,@$SPTR ; WILL STORE A $TEMP LOC'N.
 2057 05C1 6000 A         PUSH ACO
 2058 05C2 C067 A         LD ACO,$SPTR
 2059 05C3 F145 A         SKNE ACO,$FEMP  ; SKIP IF EXECUTED
 2060 05C4 1903 A         JMP RSTPTR      ; BEFORE A STKFUL.
 2061 05C5 79FF A         AISZ AC1,-1
 2062 05C6 19F8 A         JMP $EMP
 2063 05C7 1904 A         JMP $REST
 2064 05C8 C13F A RSTPTR: LD ACO,$SSBEG ; RESTORE PTR TO
 2065 05C9 D067 A         ST ACO,$SPTR  ; START OF SOFTWARE STK.
 2066 05CA 505C A         LI ACO,X'5C   ; LOAD UNFLW DEFAULT
 2067 05CB 192B A         JMP $UNFLW
 2068              ;
 2069              ; RESTORE REGISTERS AND RETURN FROM INTERRUPT
 2070              ;
 2071 05CC C04F A $REST: LD ACO,$RETA
 2072 05CD 6000 A         PUSH ACO
 2073 05CE C04B A         LD ACO,$SAV0
 2074 05CF C44C A         LD AC1,$SAV1
 2075 05D0 C84D A         LD AC2,$SAV2
 2076 05D1 CC4E A         LD AC3,$SAV3
 2077 05D2 3100 A         PFLG IEN1
 2078 05D3 3180 A         SFLG IEN1
 2079 05D4 7C00 A         RTI
 2080              ;
 2081              ; STACK IS ALMOST FULL
 2082              ;
 2083 05D5 6400 A $AFULL: PULL ACO
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE   53
PACE  FTND CTRLPRGM 17/10/78
STKFUL INTERRUPTS


 2084 05D6 5104 A          LI AC1,4    ; NO. OF WORDS TO SAVE
 2085 05D7 1901 A          JMP .+2
 2086               ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE   54
PACE  FTND CTRLPRGM 17/10/78
STKFUL INTRPT. SUB. -IEN1


 2087                      .PAGE "STKFUL INTRPT. SUB. -IEN1"
 2088               ;
 2089               ; STACK FULL. SAVE TOP FIVE ELEMENTS OF STACK
 2090               ;
 2091 05D8 5105 A $FULL:   LI AC1,5
 2092 05D9 D466 A          ST AC1,$SWPTR
 2093 05DA C92B A          LD AC2,$ADR
 2094                  ;
 2095 05DB 6400 A $LP1:    PULL AC0
 2096 05DC D200 A          ST AC0,(AC2)
 2097 05DD 7A01 A          AISZ AC2,1
 2098 05DE 79FF A          AISZ AC1,-1
 2099 05DF 19FB A          JMP $LP1
 2100               ;
 2101               ; NOW PUT BOTTOM FOUR WORDS ONTO SOFTWARE STACK
 2102               ;
 2103 05E0 5104 A          LI AC1,4
 2104 05E1 6400 A $LP2:    PULL AC0
 2105 05E2 B067 A          ST AC0,@$SPTR
 2106 05E3 C067 A          LD AC0,$SPTR ; CHECK FOR OVERFLOW.
 2107 05E4 F122 A          SKNE AC0,$SSEND
 2108 05E5 190B A          JMP $OVFL
 2109 05E6 7801 A          AISZ AC0,1
 2110 05E7 D067 A          ST AC0,$SPTR
 2111 05E8 79FF A          AISZ AC1,-1
 2112 05E9 19F7 A          JMP $LP2
 2113               ;
 2114               ; FINALLY RESTORE TOP 4/5 WORDS TO BOTTOM OF STACK
 2115               ;
 2116 05EA C466 A          LD AC1,$SWPTR ; LOAD TOP 4/5 " ".
 2117 05EB 7AFF A $LP3:    AISZ AC2,-1
 2118 05EC C200 A          LD AC0,(AC2)
 2119 05ED 6000 A          PUSH AC0
 2120 05EE 79FF A          AISZ AC1,-1
 2121 05EF 19FB A          JMP $LP3
 2122 05F0 19DB A          JMP $REST
 2123               ;
```

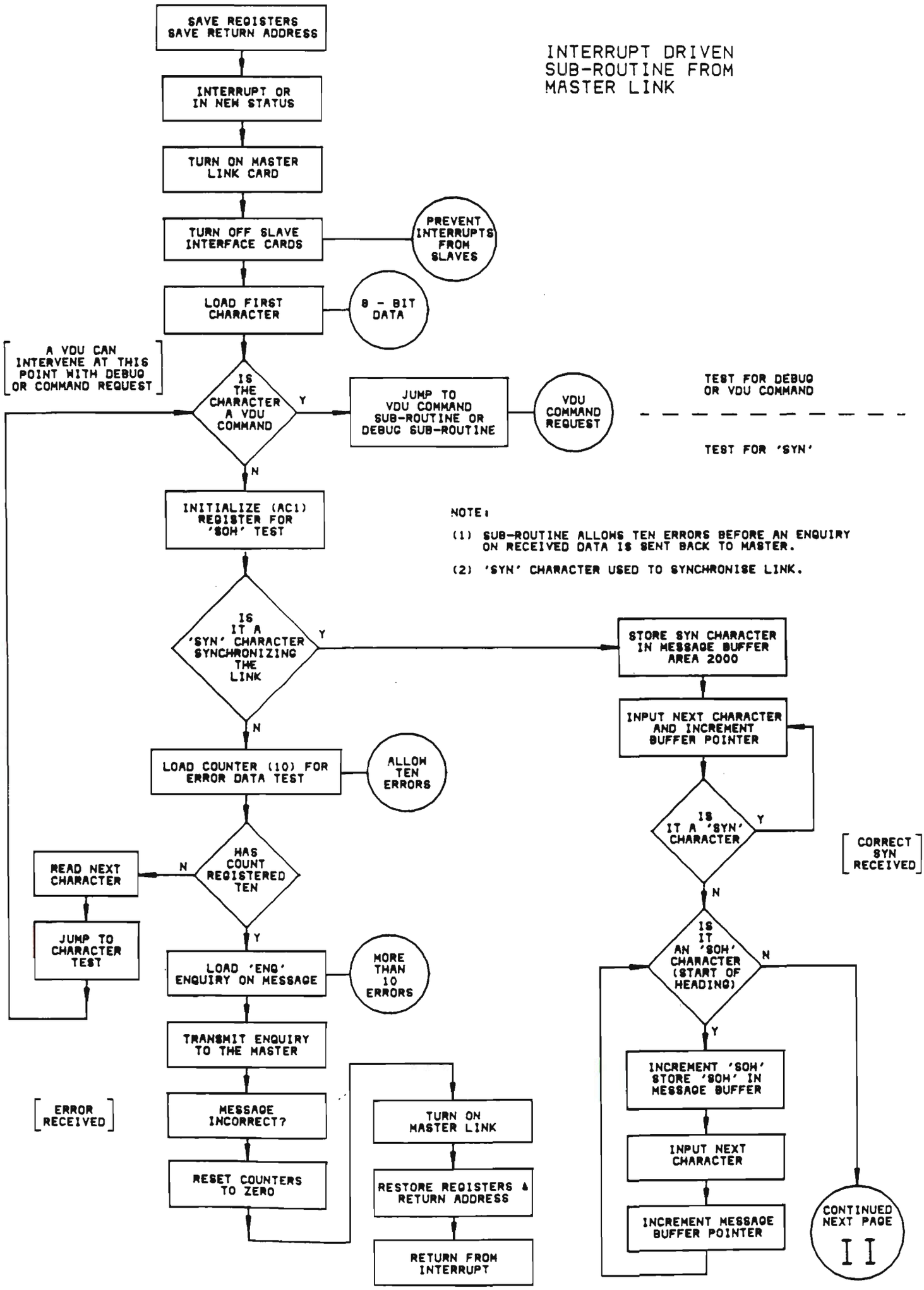
```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   55
PACE  FTND CTRLPRGM 17/10/78
STKFUL INTERRUPTS


 2124                        .PAGE 'STKFUL INTERRUPTS'
 2125                ;
 2126                ;  STACK OVERFLOW. /UNDERFLOWS.
 2127                ;
 2128 05F1 5109 A $OVFL:  LI AC1,9       ; STACK OVFLOW. HAS OCCURRED.
 2129 05F2 6400 A         PULL AC0
 2130 05F3 79FF A         AISZ AC1,-1
 2131 05F4 19FD A         JMP .-2
 2132 05F5 5046 A         LI   AC0,X'46 ;   TRANS. " F " TO INDICATE OVFLW.
 2133 05F6 1902 A         JMP $PCLOD
 2134 05F7 5045 A $UNFLW: LI   AC0,X'45 ; TRANS. " E" TO INDICATE UNFLW.
 2135 05F8 6000 A         PUSH AC0   ; PUSH STACK FOR UNFLW. RESET.
 2136 05F9 149A A $PCLOD: JSR TTYFX
 2137 05FA 1492 A         JSR TTYS
 2138 05FB 5053 A         LI AC0,X'53      ; SEND "S"
 2139 05FC 1492 A         JSR TTYS
 2140 05FD C84F A         LD AC2,$RETA       ; PC. AT OV/UNFLW.
 2141 05FE 9410 A         JSR @XPUTW       ; SHOW THIS !
 2142 05FF 9411 A         JSR @XCRLF       ; DO A CR-LF.
 2143 0600 C107 A         LD AC0,$SSBEG
 2144 0601 D067 A         ST AC0,$SPTR
 2145 0602 3100 A         PFLG 1
 2146 0603 3180 A         SFLG 1
 2147 0604 3980 A         SFLG 9  ;RE-ENABLE INTERRUPTS
 2148 0605 9904 A         JMP @PPRMT ; JUMP TO DEBUG
 2149                ;
 2150                ;    POINTERS.
 2151                ;
 2152 0606 0061 A $ADR:   .WORD $STAK
 2153 0607 0060 A $SSEND: .WORD STKEND  ;INCRES. TO 10FF FOR MORE
 2154 0608 0051 A $SSBEG: .WORD SOFTST ;   "      "  1000 "   "RM.
 2155 0609 0050 A $FEMP:  .WORD SOFTST-1 ; $SPTR DECRMTD. TOO EARLY
 2156 060A 00FB A PPRMT:  .WORD PROMPT ;POINTER TO DEBUG PROMPT ENTRY
 2157                ;
 2158                ;
 2159 060B 9411 A FLUSH:  JSR @XCRLF
 2160 060C 1512 A         JSR FMES
 2161 060D 062C A         .WORD FMES1
 2162 060E C82A A         LD AC2,HLOC
 2163 060F 9410 A         JSR @XPUTW
 2164 0610 C82B A           LD AC2,HLOC+1
 2165 0611 9410 A         JSR @XPUTW
 2166 0612 C82C A          LD AC2,HLOC+2
 2167 0613 9410 A         JSR @XPUTW
 2168 0614 C82D A         LD AC2,HLOC+3
 2169 0615 9410 A           JSR @XPUTW
 2170 0616 9411 A         JSR @XCRLF
 2171 0617 1507 A         JSR FMES
 2172 0618 0632 A           .WORD FMES2
 2173 0619 9412 A           JSR @XGECHO
 2174 061A 1902 A         JMP .+3   ;"CR" FLUSH !
 2175 061B F10F A         SKNE AC0,FASCN
 2176 061C 8008 A         RTS 8       ; "N" -- NO FLUSH
 2177 061D 5000 A       LI AC0,0    ;  FLUSH REQD.
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   56
PACE  FTND CTRLPRGM 17/10/78
STKFUL INTERRUPTS


2178 061E 8000 A          RTS 0
2179 061F 6600 A FMES:    PULL AC2
2180 0620 6200 A          PUSH AC2
2181 0621 CA00 A          LD AC2,0(2)
2182 0622 C200 A FMESA:   LD AC0,0(2)
2183 0623 4501 A          BOC NZRO,.+2
2184 0624 8001 A          RTS 1
2185 0625 2010 A          ROL 0,8,0
2186 0626 1492 A          JSR TTYS
2187 0627 2010 A          ROL 0,8,0
2188 0628 1492 A          JSR TTYS
2189 0629 7A01 A          AISZ 2,1
2190 062A 19F7 A          JMP FMESA
2191                ;
2192 062B 004E A FASCN:   .WORD X'4E
2193 062C 4252 A FMES1:    .ASCII 'BR/PTS @  '
2194 0631 0000 A          .WORD 0
2195 0632 464C A FMES2:    .ASCII 'FLUSH ? '
2196 0636 0000 A          .WORD 0
2197                ;
2198 0637 9411 A TRSET:   JSR @XCRLF
2199 0638 15E6 A          JSR FMES    ;  USE FLUSH PRT. SUB.
2200 0639 064B A          .WORD TMES1
2201 063A 9412 A          JSR @XGECHO
2202 063B 1902 A          JMP .+3     ; "CR" CLEAR
2203 063C F10C A          SKNE AC0,TASCN
2204 063D 8008 A          RTS 8       ; "N" NOT CLEARED
2205 063E 5000 A          LI AC0,0
2206 063F 8000 A          RTS 0       ;    CLEAR
2207                ;
2208                ;
2209 0640 9411 A MRSET:   JSR @XCRLF
2210 0641 15DD A          JSR FMES    ; USE FLUSH SUB.
2211 0642 0658 A          .WORD MMES1
2212 0643 9412 A          JSR @XGECHO
2213 0644 1902 A          JMP .+3     ; "CR" CLEAR
2214 0645 F104 A          SKNE AC0,MASCN
2215 0646 8008 A          RTS 8       ; "N" NOT CLEARED
2216 0647 5000 A          LI AC0,0
2217 0648 8000 A          RTS 0       ;    CLEAR
2218                ;
2219 0649 004E A TASCN:   .WORD X'4E
2220 064A 004E A MASCN:   .WORD X'4E
2221 064B 5245 A TMES1:   .ASCII 'RESET REAL TIME CLOCKS ?'
2222 0657 0000 A          .WORD 0
2223 0658 2020 A MMES1:   .ASCII '  "  MASTER & SLAVES MEM. LOCS.'
2224 0668 0000 A          .WORD 0
2225                ;
2226                ; CHECKSUM SUBROUTINE & PRINT RESULT
2227                ;
2228 0669 9900 A RANGE1:  JMP @RANGE2
2229 066A 035B A RANGE2:  .WORD RANGE ; GET MEM. RANGE
2230 066B 9900 A CRLFJ:   JMP @CRLFK
2231 066C 013F A CRLFK:   .WORD CRLF  ; CR/LF TX. SUB.
```

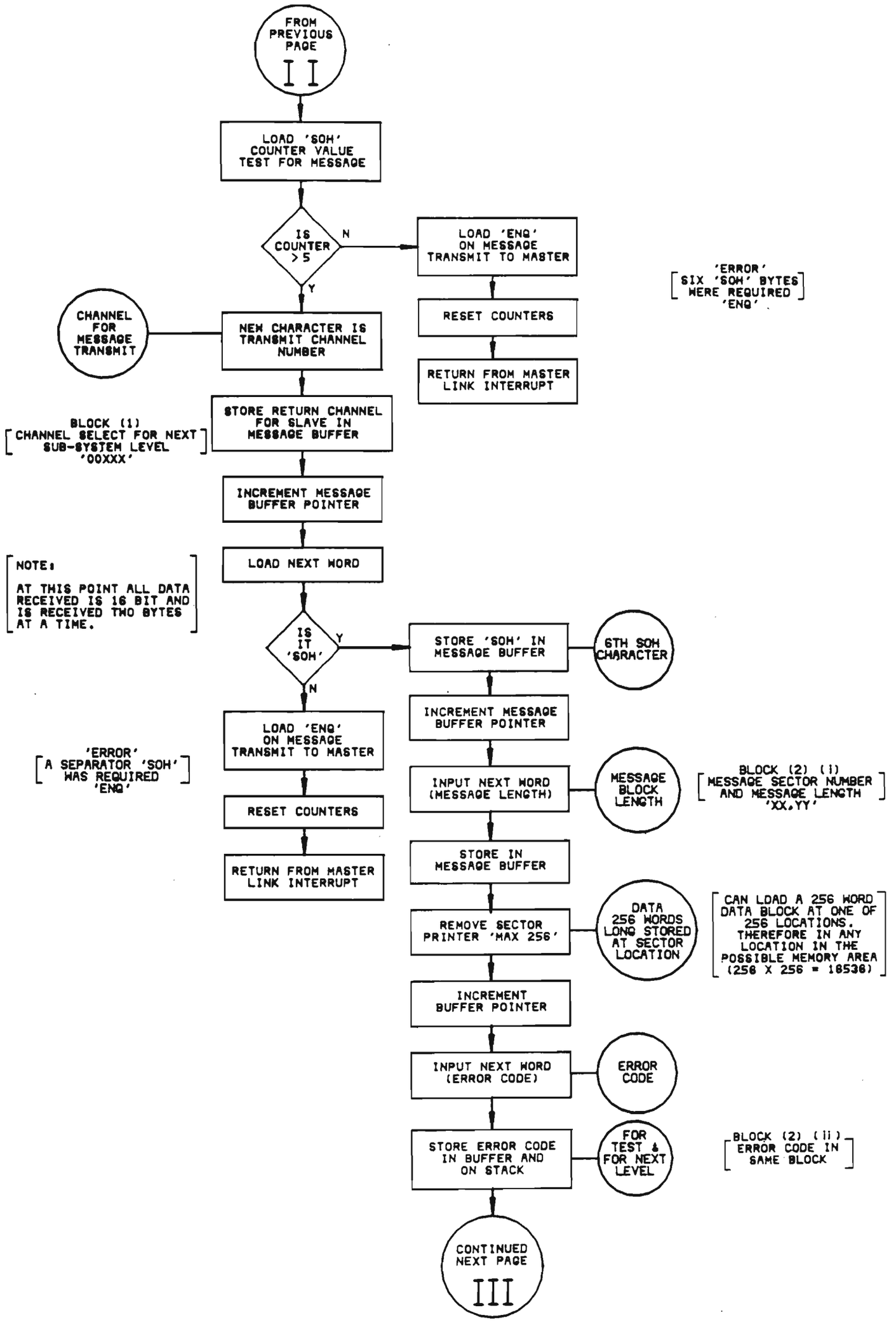
APPENDIX B : SYSTEM SOFTWARE : PART B

- FRONT-END TO MASTER INTERRUPT SUB-ROUTINE:

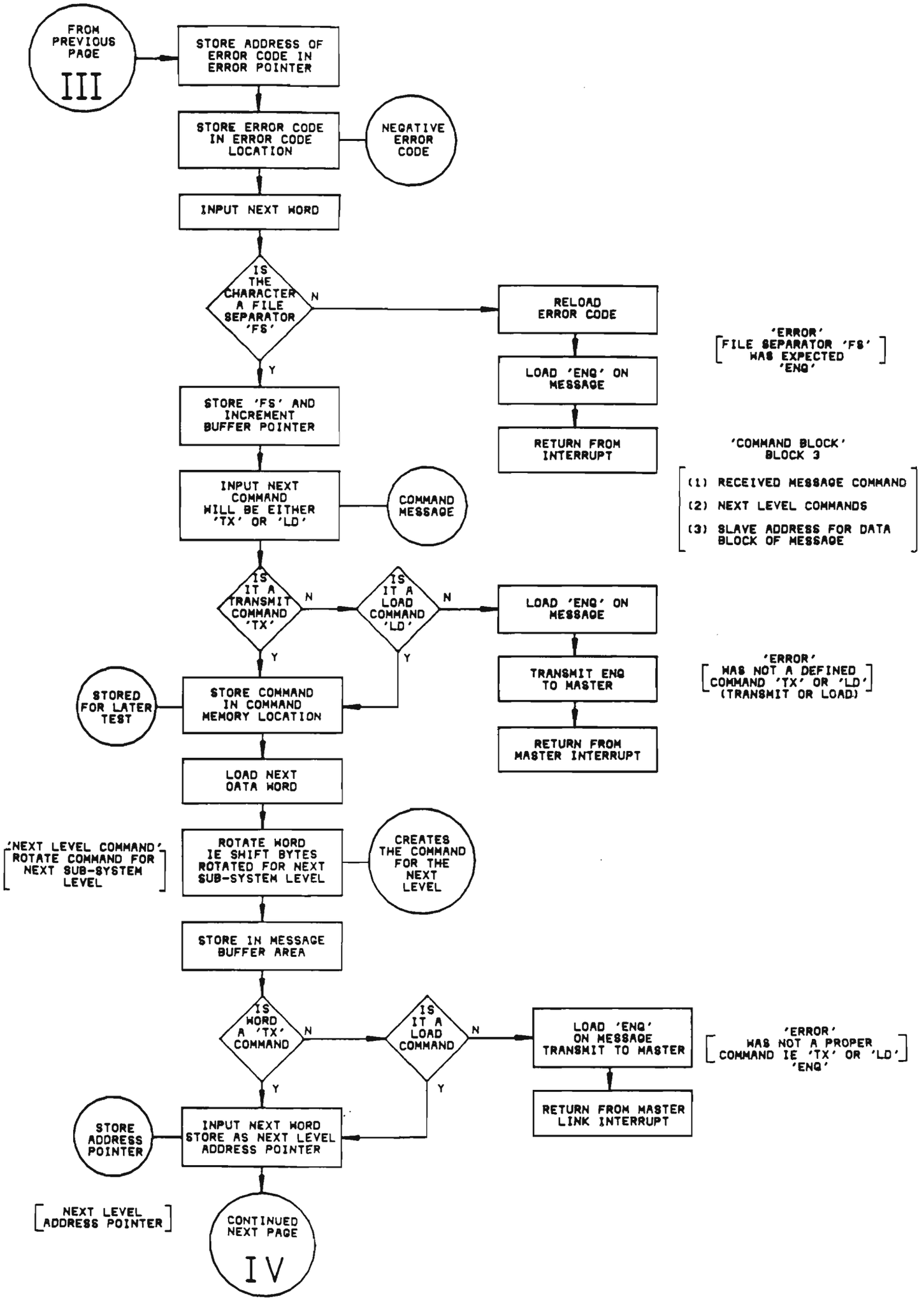
'MSTRIT' SUB-ROUTINE - APPENDIX B (PAGE 1 OF 6)
'INTERRUPT IEN'



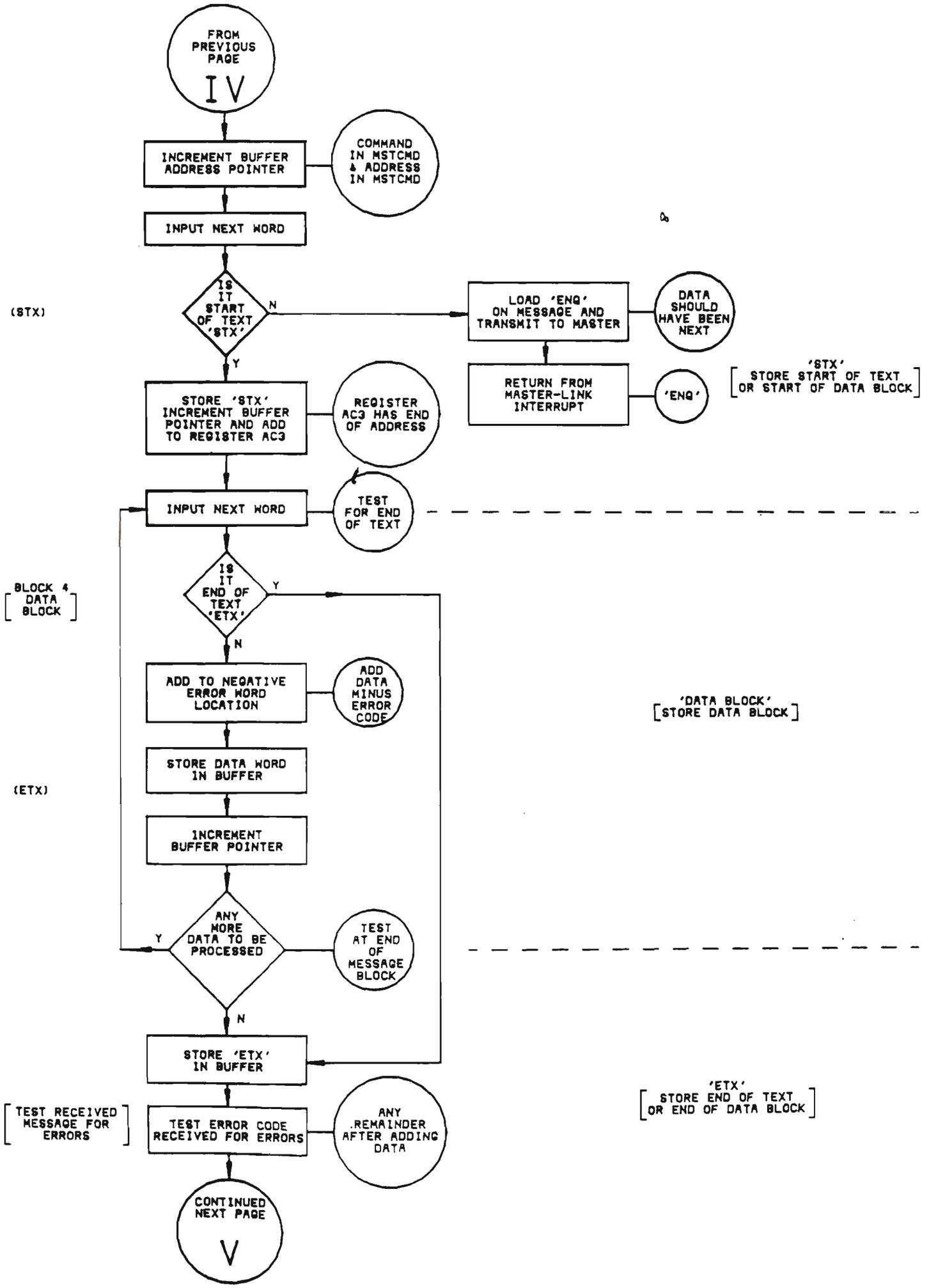
'MSTRIT' SUB-ROUTINE - APPENDIX B (PAGE 2 OF 6)



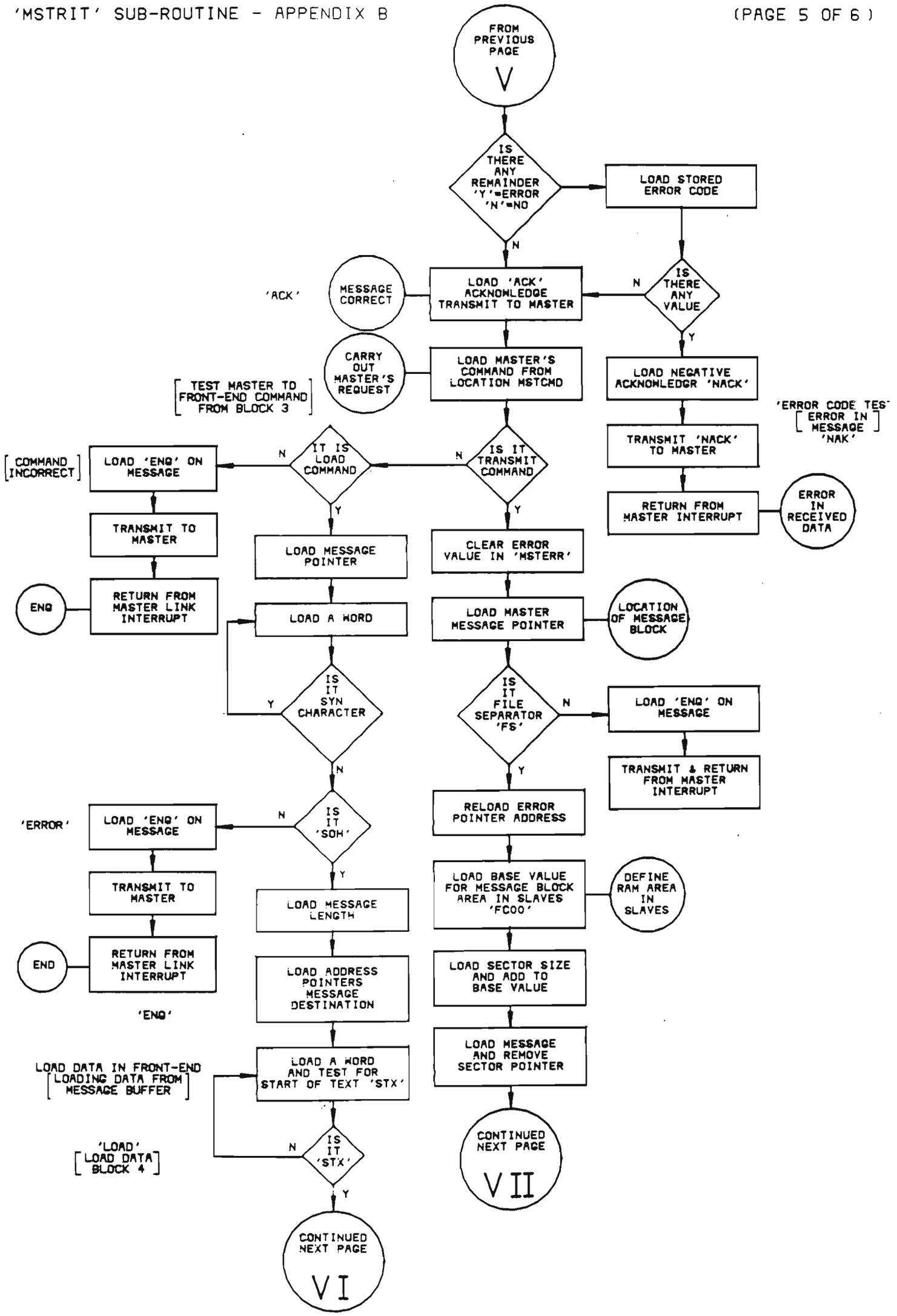
'MSTRIT' SUB-ROUTINE - APPENDIX B (PAGE 3 OF 6)



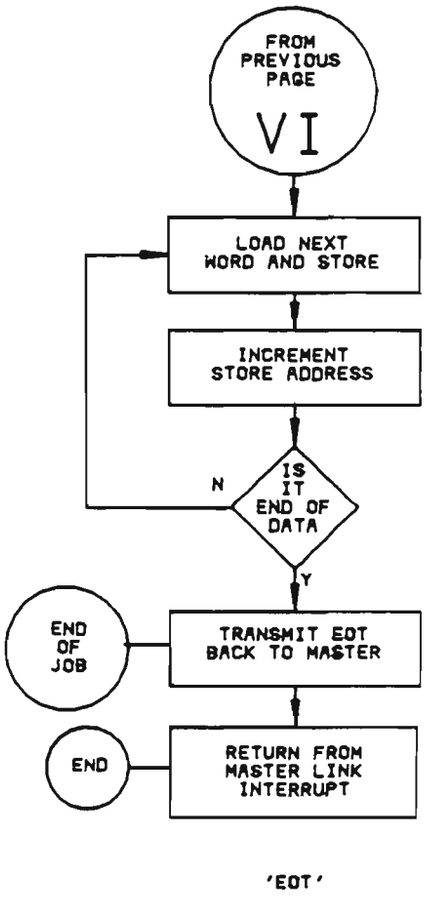
'MSTRIT' SUB-ROUTINE - APPENDIX B (PAGE 4 OF 6)



'MSTRIT' SUB-ROUTINE - APPENDIX B (PAGE 5 OF 6)

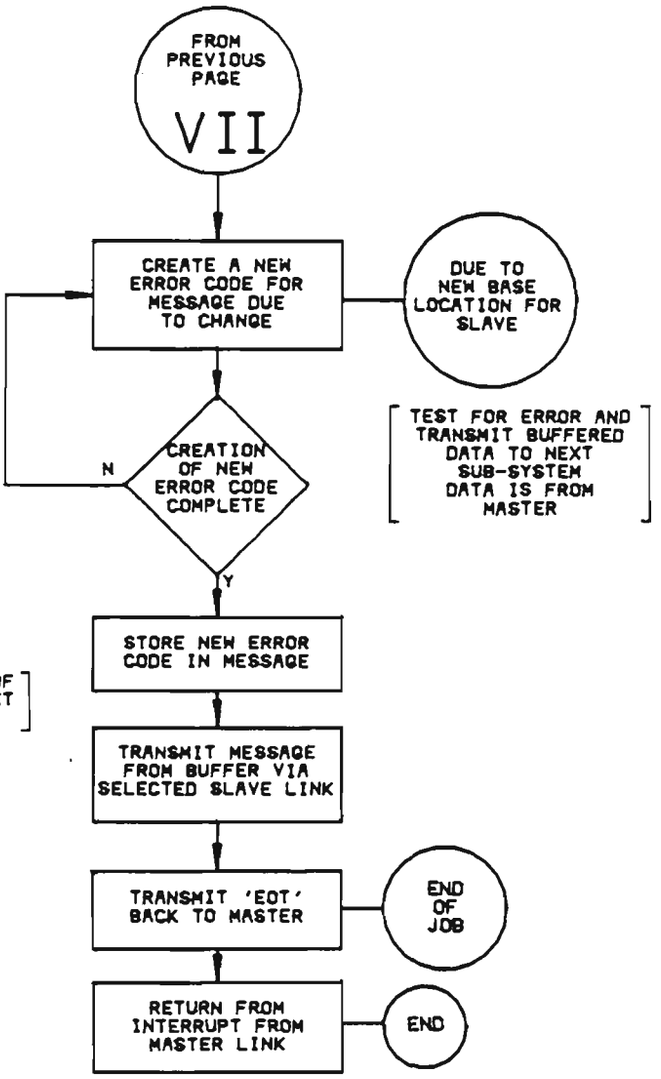


'MSTRIT' SUB-ROUTINE - APPENDIX B (PAGE 6 OF 6)



'EOT'

'TRANSMIT'
[TRANSMIT ALL OF MESSAGE TO NEXT SUB-SYSTEM]



'EOT'

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   10
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  391                       .PAGE 'MASTER INTRPT. SUB.-IEN3'
  392               ;
  393               ;*****************************************************
  394               ;*                                                   *
  395               ;*                                                   *
  396               ;*      MASTER INTERRUPT SUBROUTINE IEN3             *
  397               ;*    CMNDS.  FROM MASTER CAUSE INTRPTS.             *
  398               ;*    THIS ROUTINE WILL LOAD A TRANSMITTED CMND      *
  399               ;* FROM CHANNEL 1-TEST AND THEN TRANS. DEPENDING     *
  400               ;*  ON SELECTED CHNL. 'DC1 N' AS REC'D. IN CMND.     *
  401               ;*                  X'9800                           *
  402               ;*****************************************************
  403               ;
  404               ;
  405 06BB 5C00 A MSTRIT: RCPY AC0,AC0     ; MSTR INTRPT. SUB. START
  406 06BC D068 A         ST AC0,MSTREG    ; SAVE REGS.
  407 06BD D469 A         ST AC1,MSTREG+1
  408 06BE D86A A         ST AC2,MSTREG+2
  409 06BF DC6B A         ST AC3,MSTREG+3
  410 06C0 6400 A         PULL AC0
  411 06C1 D06C A         ST AC0,RETMST    ; RETURN ADDRESS
  412 06C2 0400 A         CFR AC0          ; MOVE " TO AC0
  413 06C3 A90C A         AND AC0,MMSK1    ;MASK OLD INT. STATUS
  414 06C4 A50C A         OR  AC0,MINTT1   ; OR IN NEW "    "
  415 06C5 0800 A         CRF AC0
  416               ;
  417               ;
  418 06C6 50FF A MINITN: LI AC0,X'FF ; LOAD DMMY WRD (INITIALIZE SUB.)
  419 06C7 CD0A A         LD AC3,MTRCD1      ; LOAD MASTER CARD ADDS.
  420 06C8 D302 A         ST AC0,TTYON(3)    ; TURN ON MASTER CARD
  421 06C9 CD09 A         LD AC3,SLVC11
  422 06CA D303 A         ST AC0,TTYOFF(3)   ; TURN OFF SLAVE1
  423 06CB CD08 A         LD AC3,SLVC21
  424 06CC D303 A         ST AC0,TTYOFF(3)    ; "    " SLAVE2
  425 06CD 155D A         JSR MSTINP
  426 06CE 5100 A         LI AC1,0         ; LOAD 'SOH' CNTR. W. 0
  427 06CF 1905 A         JMP CMDTST   ; TEST FIRST INPUT VDU/MSTR.
  428               ;
  429               ;
  430 06D0 8207 A MMSK1:  .WORD X'8207
  431 06D1 8207 A MINTT1: .WORD X'8207
  432 06D2 9800 A MTRCD1: .WORD X'9800
  433 06D3 9900 A SLVC11: .WORD X'9900
  434 06D4 9880 A SLVC21: .WORD X'9880
  435               ;
  436               ;
  437 06D5 5C00 A CMDTST: RCPY  AC0,AC0  ;NOP
  438 06D6 CD0E A         LD AC3,DGTBL
  439 06D7 F300 A NXMCMD: SKNE AC0,(AC3)  ;NXT MSTR CMD
  440 06D8 1905 A         JMP CMDGOT    ; EXIT IF EQ. TO CMD.
  441 06D9 C700 A         LD AC1,(AC3)
  442 06DA F52D A         SKNE AC1,TZRO
  443 06DB 192E A         JMP MSMSOH    ; ALTER HERE
  444 06DC 7B02 A         AISZ AC3,2     ;CHECK NXT CMD
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE   11
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


 445 06DD 19F9 A          JMP NXMCMD
 446 06DE CF01 A CMDGOT: LD AC3,1(AC3)   ; LOAD ADDS.
 447 06DF 5C00 A          RCPY AC0,AC0 ;WAS PULLF BUT CAUSED STK.
 448 06E0 C06C A          LD AC0,RETMST
 449 06E1 5C00 A          RCPY AC0,AC0  ;NOP
 450 06E2 6400 A          PULL AC0      ;  "    JSR GECHO  FRM STK.
 451 06E3 6400 A          PULL AC0      ;  "     "  AT  X'F2
 452               ;
 453 06E4 1B00 A          JMP (AC3)     ; GO TO PROG.
 454 06E5 06E6 A DGTBL:   .WORD DBGTBL
 455 06E6 0041 A DBGTBL: .WORD X'41     ; "A"
 456 06E7 0214 A          .WORD ALTER
 457 06E8 004C A          .WORD X'4C     ; "L"
 458 06E9 022A A          .WORD LDREG
 459 06EA 0050 A          .WORD X'50     ; "P"
 460 06EB 039B A          .WORD PRINT
 461 06EC 0043 A          .WORD X'43      ; "C" XX,YY COPY MEM. ONLY
 462 06ED 0433 A          .WORD COPPY  ;   "CS" XX,YY CHKSM REQST.
 463 06EE 0042 A          .WORD X'42     ;" B"
 464 06EF 0418 A          .WORD BINARY
 465 06F0 0044 A          .WORD X'44     ; "D"
 466 06F1 0429 A          .WORD DUMP
 467 06F2 0057 A          .WORD X'57       ;"W"
 468 06F3 049E A          .WORD WHERE
 469 06F4 0053 A          .WORD X'53     ; "S"
 470 06F5 041E A          .WORD STORE
 471 06F6 0052 A          .WORD X'52     ; "R"
 472 06F7 03B7 A          .WORD REGTYPE
 473 06F8 0049 A          .WORD X'49    ;"I"
 474 06F9 0368 A          .WORD INSERT
 475 06FA 0048 A          .WORD X'48     ;"H"
 476 06FB 03C6 A          .WORD HALT
 477 06FC 0047 A          .WORD X'47     ;"G"
 478 06FD 03F1 A          .WORD GO
 479 06FE 001A A          .WORD X'1A     ; CONTROL "Z","SUB"
 480 06FF 0379 A          .WORD CMDTRN
 481 0700 0058 A          .WORD X'58     ; "X"
 482 0701 020A A          .WORD MHEX
 483 0702 004F A          .WORD X'4F     ; "O"
 484 0703 020F A          .WORD MOCT
 485 0704 0011 A          .WORD X'11     ; "CONTRL Q" , " DC1"
 486 0705 00A5 A          .WORD TTYMD
 487 0706 0054 A          .WORD X'54     ; "T"
 488 0707 023A A          .WORD TIME
 489 0708 0000 A TZRO:    .WORD 000   ;END TABLE
 490               ;
 491               ;
 492 0709 076F A TBLIXT: .WORD MSTIXT  ; EXIT IF ZERO
 493               ;
 494               ;
 495 070A 5100 A MSMSOH:  LI AC1,0       ; LOAD 'SOH' CNTR. W. 0
 496 070B 193E A          JMP MSTSYN
 497               ;
 498               ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   12
PACE  FIND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  499                ;
  500 070C 1502 A MSTILD: JSR MSTRED      ; FRNT-END INPUT LOAD
  501 070D 6960 A         RADD AC0,AC1     ; FINAL 16 BITS
  502 070E 8000 A         RTS 0
  503                ;
  504 070F 6100 A MSTRED: PUSH AC1 ; FRNT-END READ 16 BITS
  505 0710 151A A         JSR MSTINP       ; INPUT LSB'S.
  506 0711 2810 A         SHL 0,8,0
  507 0712 6000 A         PUSH AC0
  508 0713 1517 A         JSR MSTINP       ; INPUT MSB'S.
  509 0714 6500 A         PULL AC1
  510 0715 5840 A         RXOR AC1,AC0
  511 0716 6500 A         PULL AC1
  512 0717 8000 A         RTS 0
  513                ;
  514 0718 6300 A MSTFET: PUSH AC3   ;FRNT-END.-MASTER TRANS'MT.
  515 0719 CD25 A         LD AC3,MSTCD1
  516 071A D303 A         ST AC0,TTYOFF(3)
  517 071B CD24 A         LD AC3,SL1CD1
  518 071C D303 A         ST AC0,TTYOFF(3)
  519 071D CD23 A         LD AC3,SL2CD1
  520 071E D303 A         ST AC0,TTYOFF(3)
  521 071F 6700 A         PULL AC3 ; AC3 CARD ADDS DETERMD. BY JSR CALL
  522 0720 D302 A         ST AC0,TTYON(3)
  523 0721 D300 A         ST AC0,TTYOUT(3)
  524 0722 4D01 A         BOC PFLG,.+2
  525 0723 19FE A         JMP .-1
  526 0724 4F01 A         BOC RFLG,.+2 ; WAIT FOR RTRN MSG
  527 0725 19FE A         JMP .-1
  528 0726 C305 A         LD AC0,TTYGET(3) ; CLEAR INCOM.
  529 0727 F11E A         SKNE AC0,MS2EOT ; SKIP IF NOT EOT
  530 0728 1910 A         JMP MSTTRM       ; UNCONDITIONAL TERMTE.
  531 0729 D303 A         ST AC0,TTYOFF(3)
  532 072A 8000 A         RTS 0
  533                ;
  534                ;
  535 072B 6300 A MSTINP: PUSH AC3        ; FRNT-END INPUT SUB.
  536 072C CD12 A         LD AC3,MSTCD1    ; LOAD CHNNL. 1 ADD'S.
  537 072D D302 A         ST AC0,TTYON(3)
  538 072E 4F01 A         BOC RFLG,.+2  ; SKIP IF INPUT FLAG SET
  539 072F 19FE A         JMP .-1
  540 0730 C305 A         LD AC0,TTYGET(3) ;  "  CHART'R. FROM  "
  541 0731 A911 A         AND AC0,MSRCMK  ;MSK UPPER BITS
  542 0732 F10B A         SKNE AC0,MS3EOT  ; SKIP IF NOT EOT
  543 0733 1905 A         JMP  MSTTRM     ; UNCONDITIONAL RETURN
  544 0734 D300 A MSTSTR: ST AC0,TTYOUT(3) ; PULSE LINE WITH CH.
  545 0735 4D01 A         BOC PFLG,.+2
  546 0736 19FE A         JMP .-1
  547 0737 6700 A         PULL AC3
  548 0738 8000 A         RTS 0
  549 0739 6700 A MSTTRM: PULL AC3   ; MASTER TERMNTE.
  550 073A 6400 A         PULL AC0  ; CLEAR STACK (JSR)
  551 073B 1933 A         JMP MSTIXT ;UNCONDITIONAL EXIT
  552                ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   13
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  553 073C 820F A MSTMSK: .WORD X'820F ; STKFL,CLK,MSTR
  554 073D 826F A MSTINT: .WORD X'820F ; CAN INTERPT.
  555 073E 0004 A MS3EOT: .WORD X'0004
  556                     ;
  557 073F 9800 A MSTCD1: .WORD X'9800
  558 0740 9900 A SL1CD1: .WORD X'9900
  559 0741 9880 A SL2CD1: .WORD X'9880
  560 0742 0001 A MSTONE: .WORD X'1
  561 0743 00FF A MSRCMK: .WORD X'00FF
  562 0744 000A A MSOCNT: .WORD X'A  ; DEC. 10 COUNT
  563 0745 0016 A MSTSCH: .WORD X'16 ;SYN CHARACTER
  564 0746 FF04 A MS2EOT: .WORD X'FF04 ; SEE MSTEOT
  565 0747 0001 A MS2SOH: .WORD X'1 ;  "   MSTSOH
  566 0748 0005 A MS2SHC: .WORD X'5 ; SEE SHMCNT
  567 0749 06BB A MSSRIT: .WORD MSTRIT
  568                     ;
  569                     ;
  570 074A F1FA A MSTSYN: SKNE AC0,MSTSCH  ; TEST FOR SYN CHTR.
  571 074B 1907 A         JMP MS0STR
  572 074C C46E A         LD AC1,MSNSYN  ; LD IN NU. SYN CNT.
  573 074D E5F4 A         ADD AC1,MSTONE ; INCRMT.
  574 074E D46E A         ST AC1,MSNSYN  ; RESET COUNT
  575 074F 6C40 A         RXCH AC1,AC0
  576 0750 9DF3 A         SKG AC0,MSOCNT ;IF >10 ENQ ON MESG.
  577 0751 99F7 A         JMP @MSSRIT    ;CONT'E. UNTL SOH OR XCESS
  578 0752 190D A         JMP MSEROR     ; OUTPUT ENQ ON MESG
  579 0753 C92E A MS0STR: LD AC2,MSTMPT ;LD MSTR MESG STR. PTR.
  580 0754 D200 A MSSSST: ST AC0,(AC2)  ; STR FIRST SYN
  581 0755 E9EC A         ADD AC2,MSTONE
  582 0756 15D4 A         JSR MSTINP  ; INPUT NXT CH.
  583 0757 F1ED A         SKNE AC0,MSTSCH ; TEST FOR NXT SYN CH.
  584 0758 19FB A         JMP MSSSST
  585 0759 F1ED A MSTRSH: SKNE AC0,MS2SOH   ; SKIP IF NOT START OF MESG.
  586 075A 190B A         JMP MSCTSH
  587 075B C46D A SHTST1: LD AC1,MSOHCT     ; START OF MESG. TEST
  588 075C 6C40 A         RXCH AC1,AC0
  589 075D 9DEA A         SKG AC0,MS2SHC ; SKIP IF >5 SOH CNT.
  590 075E 1901 A         JMP MSEROR
  591 075F 1936 A         JMP MSCNST   ; FTHR. PRCSG. INPT.  DATA ie. CHNNL. SL
  592 0760 5005 A MSEROR: LI AC0,X'5     ; ENQUIRY ON MESG.
  593 0761 CDDD A         LD AC3,MSTCD1
  594 0762 D300 A         ST AC0,TTYOUT(3)
  595 0763 4D01 A         BOC PFLG,.+2
  596 0764 19FE A         JMP .-1
  597 0765 1909 A         JMP MSTIXT
  598 0766 C46D A MSCTSH: LD AC1,MSOHCT
  599 0767 E5DA A         ADD AC1,MSTONE    ; MSTR. CNT. OF SOH'S
  600 0768 D46D A         ST AC1,MSOHCT
  601 0769 D200 A MSSTRE: ST AC0,(2)       ; STORE SOH DATA
  602 076A F1DB A         SKNE AC0,MS2EOT
  603 076B 1903 A         JMP MSTIXT   ; EXIT END OF CMND.
  604 076C E9D5 A         ADD AC2,MSTONE
  605 076D 15BD A         JSR MSTINP
  606 076E 19EA A         JMP MSTRSH   ; NXT. SOH
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   14
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  607                ;
  608 076F 5C00 A MSTIXT: RCPY AC0,AC0; EXIT ROUTINE
  609 0770 5000 A         LI AC0,0
  610 0771 D06D A         ST AC0,MSOHCT   ; RESET "SOH" COUNT
  611 0772 D06E A         ST AC0,MSNSYN ;  "   SYN     "
  612 0773 CDCB A         LD AC3,MSTCD1
  613 0774 D302 A         ST AC0,TTYON(3) ;TURN ON
  614 0775 CDCA A         LD AC3,SL1CD1
  615 0776 D303 A         ST AC0,TTYOFF(3) ;   "  OFF S1
  616 0777 CDC9 A         LD AC3,SL2CD1 ;    "  "  S2
  617 0778 D303 A         ST AC0,TTYOFF(3)
  618 0779 C06C A         LD AC0,RETMST
  619 077A 6000 A         PUSH AC0
  620 077B C068 A         LD AC0,MSTREG
  621 077C C469 A         LD AC1,MSTREG+1
  622 077D C86A A         LD AC2,MSTREG+2
  623 077E CC6B A         LD AC3,MSTREG+3
  624 077F 3300 A         PFLG IEN3
  625 0780 3380 A         SFLG IEN3
  626 0781 7C00 A         RTI
  627                ;
  628 0782 2000 A MSTMPT: .WORD MMSGST
  629 0783 C5FE A MSMTRN: LD AC1,MSTMPT   ;MESG. TRANSMIT SUB.
  630 0784 C873 A         LD AC2,MSERAD  ; LAST ADDS. DATA STORE
  631 0785 7A01 A         AISZ AC2,1      ;  INC TO INC. ETX CODE.
  632 0786 6D80 A         RXCH AC2,AC1 ; AC2 HAS END OF MESG. STORE.
  633 0787 C200 A MSMTX2: LD AC0,(AC2) ; TRNS BLOCK AS DETR AC2 TO AC3 VAL.
  634 0788 CC6F A         LD AC3,MSTCHL
  635 0789 6000 A         PUSH AC0
  636 078A 2410 A         ROR 0,8,0
  637 078B 158C A         JSR MSTFET
  638 078C 6400 A         PULL AC0
  639 078D 6000 A MSSITX: PUSH AC0 ; TRANS. A  CH. :NEED A RETURN CH.
  640 078E A96D A         AND AC0,MSRUD
  641 078F 1588 A         JSR MSTFET
  642 0790 6400 A         PULL AC0
  643 0791 E9B0 A         ADD AC2,MSTONE
  644 0792 D874 A         ST AC2,MSGEND
  645 0793 F474 A         SKNE AC1,MSGEND ; SKIP IF NOT EQ. END MES'G.
  646 0794 8000 A         RTS 0 ; RETURN TO EXIT ROUTINE.
  647 0795 19F1 A         JMP MSMTX2   ; CONTINUE UNTIL CMPLTD.
  648 0796 6D00 A MSCNST: RXCH AC0,AC1      ; MSTR. CHNNL. SELECT STORE
  649 0797 A969 A         AND AC0,MSTCHN  ;MSTR. CHNNL. SET
  650 0798 5F00 A         RCPY AC0,AC3
  651 0799 CF08 A         LD AC3,TTYUD1-1(3) ;GET TRN'S CHNNL
  652 079A DC6F A         ST AC3,MSTCHL
  653 079B 5001 A         LI AC0,1     ; RETURN ADDS. PTR.
  654 079C D200 A         ST AC0,(AC2) ; STORE RT'N. ADDS. FOR SLV.
  655 079D E9A4 A         ADD AC2,MSTONE
  656 079E 9554 A MSSCBL: JSR @MSINLD        ; INPT. STRT/NEXT BLOCK
  657 079F F15D A         SKNE AC0,MSTSOH    ;SKIP IF NOT   "
  658 07A0 1901 A         JMP MSMSTR
  659 07A1 19BE A         JMP MSEROR         ; OTPT. ENQ.    CHTR.
  660 07A2 D206 A MSMSTR: ST AC0,(AC2)       ; MESSAGE STORE (SOH)
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   15
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  661 07A3 E99E A          ADD AC2,MSTONE
  662 07A4 954E A          JSR @MSINLD          ; INPT MSSG. LNTH.
  663 07A5 D200 A          ST AC0,(AC2)    ; STORE MESG. LNTH.
  664 07A6 A91B A          AND AC0,MSTMLH  ; SLOUGH SECTOR PTR
  665 07A7 5F00 A          RCPY AC0,AC3
  666                ;
  667                ;    AC3 HAS MESSAGE LENGTH (MAX 256)
  668                ;
  669 07A8 E999 A          ADD AC2,MSTONE
  670 07A9 9549 A          JSR @MSINLD  ; INPT ERROR CODE
  671 07AA 5D00 A          RCPY AC0,AC1     ; LOAD " " IN AC1
  672                ;
  673                ;  AC1 & STK. HAS ER. CDE. =TOTL DATA WDS
  674                ;
  675 07AB D200 A          ST AC0,(AC2)  ; "  "  "
  676 07AC D873 A          ST AC2,MSERAD ; STRE. ERR. CDE MSG. ADDS
  677 07AD E994 A          ADD AC2,MSTONE
  678 07AE 6000 A          PUSH AC0  ;STORE  "  " " STK.
  679 07AF 7101 A          CAI  AC1,1        ; -(ERROR CODE)
  680 07B0 D472 A          ST AC1,MSTERR   ; STRE ERR CDE.
  681                ;
  682                ;   AC1 HAS -ERROR CODE
  683                ;
  684 07B1 9541 A          JSR @MSINLD       ; TEST FOR "FS"
  685 07B2 F150 A          SKNE AC0,MSTFSC    :FILE SEPARATOR CH.
  686 07B3 1902 A          JMP MSCMDL       ; JMP TO COMND. LD
  687 07B4 6400 A          PULL AC0
  688 07B5 19AA A          JMP MSEROR        ; OTPT. ENQ. CHTR.
  689 07B6 D200 A MSCMDL:  ST AC0,(AC2)   ; STORE FS.
  690 07B7 E98A A          ADD AC2,MSTONE
  691 07B8 953A A          JSR @MSINLD ; INPT. TX. OR LD CMND.
  692 07B9 A942 A          AND AC0,MSRWD
  693 07BA F104 A          SKNE AC0,MSTXCH
  694 07BB 1907 A          JMP MSVCMD  ; STORE TX CMND.
  695 07BC F103 A          SKNE AC0,MSLDCH
  696 07BD 1905 A          JMP MSVCMD  ; STORE LOAD CMND.
  697 07BE 19A1 A          JMP MSEROR  ; ENQ ON MSG
  698 07BF 0054 A MSTXCH:  .WORD X'54  ; TX CHARATR.  T
  699 07C0 004C A MSLDCH:  .WORD X'4C  ; LD   "      L
  700 07C1 0001 A MSTANE:  .WORD X'1
  701 07C2 00FF A MSTMLH:  .WORD X'00FF  ; MSTR. MSG. LNGTH. MASK
  702 07C3 D070 A MSVCMD:  ST AC0,MSTCMD  ;STORE COMAND
  703                ;
  704                ;
  705                ;   THIS ROUTINE ROTATES & CREATES
  706                ;   NEW CMNDS. FOR NXT LEVEL
  707                ;   DEPENDING ON SECOND CMND. GROUP
  708                ;
  709                ;
  710 07C4 952E A          JSR @MSINLD
  711 07C5 2410 A          ROR AC0,8,0  ; ROTATE NXT STAGE CMND
  712 07C6 D200 A          ST AC0,(AC2) ; STORE NEW COMMAND
  713 07C7 E9F9 A          ADD AC2,MSTANE
  714 07C8 D200 A          ST AC0,(AC2)  ; ROTATED FOR NEXT LEVEL
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   16
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  715 07C9 E9F7 A          ADD AC2,MSTANE ;   & STORED
  716 07CA 2410 A          ROR AC0,8,0   ; ROTATE BACK FOR TEST
  717 07CB A930 A          AND AC0,MSRWD
  718 07CC F1F2 A          SKNE AC0,MSTXCH
  719 07CD 1903 A          JMP MSTADP   ; JUMP TO MSTR ADDS PTR.
  720 07CE F1F1 A          SKNE AC0,MSLDCH
  721 07CF 1901 A          JMP MSTADP ;  "    "  "     "   "  .
  722 07D0 198F A          JMP MSEROR ;ENQ ON MSG
  723 07D1 9521 A MSTADP:  JSR @MSINLD
  724 07D2 D071 A          ST AC0,MSTADD  ; STORE ADDS PTR.
  725 07D3 D200 A          ST AC0,(2)
  726 07D4 E9EC A          ADD AC2,MSTANE
  727                ;
  728                ;   CMND. IN MSTCMD  ,  ADDS. LOC. IN MSTADD
  729                ;
  730 07D5 951D A          JSR @MSINLD     ; TEST FOR  STX
  731 07D6 F127 A          SKNE AC0,MSTSTX
  732 07D7 1901 A          JMP MSCDST
  733 07D8 1987 A          JMP MSEROR    ; TRNS. ENQ.
  734 07D9 D200 A MSCDST:  ST AC0,(AC2)    ; STORE  STX
  735 07DA E920 A          ADD AC2,MSTMNE
  736 07DB 6B80 A          RADD AC2,AC3     ; CMND. STORE AC3 = END TEXT
  737                ;
  738                ;   AC3  =  (END ADDS.-1)
  739                ;
  740 07DC 7301 A          CAI AC3,1   ; AC3= -(LAST LOC. -1)
  741 07DD 9515 A MSCDIP:  JSR @MSINLD       ; INPT. DATA
  742 07DE F123 A          SKNE AC0,MSTETX  ; TEST FOR END TEXT ETX.
  743 07DF 1909 A          JMP MSECDT        ; JMP TO ERROR TEST
  744 07E0 C472 A          LD AC1,MSTERR
  745 07E1 6900 A          RADD AC0,AC1
  746 07E2 D472 A          ST AC1,MSTERR ; DECREASE ERR CNT
  747 07E3 D200 A          ST AC0,(AC2)       ;STORE DATA
  748 07E4 E916 A          ADD AC2,MSTMNE   ; INCMT. ADDS.
  749 07E5 5C80 A          RCPY AC2,AC0
  750 07E6 68C0 A          RADD AC3,AC0   ; CURRENT ADDS-(LAST ADS.-1)
  751 07E7 4201 A          BOC  C2,MSECDT    ; BRNCH. TO ERROR CODE TST.
  752 07E8 19F4 A          JMP MSCDIP        ; CONTINUE INPT.
  753                ;
  754                ;  AC2 = END OF STORAGE
  755                ;
  756 07E9 5003 A MSECDT:  LI AC0,X'3  ; ETX
  757 07EA D200 A          ST AC0,(AC2)  ; STORE ETX
  758 07EB 6400 A          PULL AC0     ;   ERROR CODE TEST
  759 07EC 4117 A          BOC C1,CMMTST   ; JUMP TO CMND. EXECUTE
  760 07ED C472 A          LD AC1,MSTERR
  761 07EE 5C40 A          RCPY AC1,AC0
  762 07EF 4114 A          BOC C1,CMMTST   ;  "   "   "    "
  763 07F0 5015 A          LI AC0,X'15      ; LOAD NACK DUE TO ERR.
  764 07F1 9902 A          JMP @MSERR3
  765                ;
  766                ;
  767 07F2 072B A MS2INP:  .WORD MSTINP
  768 07F3 070C A MSINLD:  .WORD MSTILD
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   17
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  769 07F4 0761 A MSERR3: .WORD MSEROR+1
  770             ;
  771             ;
  772             ;
  773 07F5 0000 A MSNXSB: HALT      ; INSERT NEXT SUB HERE
  774 07F6 820F A MSITST: .WORD X'820F    ;NEW FLAGS
  775 07F7 820F A MSMSK:  .WORD X'820F    ; CLK CAN INTRPT.
  776 07F8 9800 A MMSCD1:  .WORD X'9800   ; MASTER CARD
  777 07F9 9800 A MSCD1A: .WORD X'9800 ;  FRTEND CARD ADDS
  778 07FA 9880 A S2CD2A: .WORD X'9880 ;  SLAVE2 CARD ADDS
  779 07FB 0001 A MSTMNE: .WORD X'1
  780 07FC 00FF A MSRWD:  .WORD X'FF  ; MASK OUT MSB'S.
  781 07FD 0001 A MSTSOH: .WORD X'1  ; "SOH" STRT. OF MSG.
  782 07FE 0002 A MSTSTX: .WORD X'2  ; STRT. OF TEXT "STX"
  783 07FF 0004 A MSTEOT: .WORD X'4  ; UNCONDITIONAL TERMINATE
  784 0800 0005 A SH1CNT: .WORD X'5   ; SOH COUNT
  785 0801 0007 A MSTCHN: .WORD X'7
  786 0802 0003 A MSTETX: .WORD X'3 ;END OF TEXT
  787 0803 001C A MSTFSC: .WORD X'1C   ; FILE SEPARATOR CHTR.
  788             ;
  789             ;
  790 0804 5006 A CMMTST: LI AC0,X'06  ; LOAD ACK -CORRECT MESG
  791 0805 CDF3 A         LD AC3,MSCD1A ; "  ADDS OF CARD
  792 0806 D302 A         ST AC0,TTYON(AC3)
  793 0807 D300 A         ST AC0,TTYOUT(AC3)
  794 0808 4D01 A         BOC PFLG,.+2
  795 0809 19FE A         JMP .-1
  796 080A C070 A         LD AC0,MSTCMD
  797 080B F1B3 A         SKNE AC0,MSTXCH
  798 080C 1904 A         JMP NECDLS       ; JMP TO NEW ERR CDE &DATA LD
  799 080D F1B2 A         SKNE AC0,MSLDCH
  800 080E 1943 A         JMP LDMXSB       ; "    "  LD  "      "
  801 080F 9931 A         JMP @MSERRR       ; CMND  ERROR
  802             ;
  803             ;
  804             ;   THIS SUB. CREATES
  805             ; 1 NEW DATA - DEPENDS ON REQ'T.
  806             ; 2  "  ERROR CODE FOR ABOVE
  807             ; 3  "  BASE ADDRESS FOR LOADING
  808             ;          IE. SEE   MSTSLA PTR.
  809             ;
  810             ;
  811 0810 2000 A MSTPT1: .WORD MMSGST
  812             ;
  813 0811 5200 A NECDLS: LI AC2,0
  814 0812 D872 A         ST AC2,MSTERR  ; CLEAR ERR CDE.
  815 0813 C5FC A         LD AC1,MSTPT1
  816 0814 6D80 A         RXCH AC2,AC1   ; AC1=0,AC2 =MSTMPT
  817 0815 C200 A MMSGLT: LD AC0,(AC2)  ; TEST FOR "FS"
  818 0816 F1EC A         SKNE AC0,MSTFSC
  819 0817 1903 A         JMP MFSFR      ; JUMP TO MSTR FS FOUND RTNE.
  820 0818 7A01 A         AISZ AC2,1
  821 0819 19FB A         JMP MMSGLT
  822 081A 9926 A         JMP @MSERRR
```

```
PACE ASSEMBLER REV-A      20 DEC 76              PAGE   18
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  823 081B 7A05 A MFSFR:  AISZ AC2,5  ; LOOK FOR MSG LNTH. & ER CD. ADDS
  824 081C D873 A         ST AC2,MSERAD  ; STRE VALUE OF AC2 IN LOC.
  825 081D 7AF9 A         AISZ AC2,-7
  826 081E CD20 A         LD AC3,MSTSLA ; LD BASE ADDS PTR.
  827 081F C200 A         LD AC0,(AC2)  ;  "  SECTOR PTR. (SEC. FOR DIF DEVICES
  828 0820 A91F A         AND AC0,MSTSEC ;
  829 0821 68C0 A         RADD AC3,AC0  ; ADD SECTOR TO BASE
  830 0822 CC71 A         LD AC3,MSTADD ; LOAD ADDS VALUE
  831 0823 68C0 A         RADD AC3,AC0  ; TOTAL = ADDS+SECTOR+BASE
  832 0824 7A05 A         AISZ AC2,5
  833 0825 D200 A         ST AC0,(AC2)   ; STRE. " & " PTR MSG ADDS.
  834 0826 7AFB A         AISZ AC2,-5   ; MSG. LNTH. LOC.
  835 0827 C200 A         LD AC0,(AC2) ; LD MSG LNTH IN AC0
  836 0828 A9D3 A         AND AC0,MSRWD  ; REMOVE SECTOR PTR.
  837 0829 C472 A         LD AC1,MSTERR
  838 082A 7A01 A         AISZ AC2,1
  839 082B D872 A         ST AC2,MSTERR ; STRE ADDS OF ERR IN ERR CDE LOC
  840 082C C871 A         LD AC2,MSTADD ; LD  ADDS  "   " AC2
  841 082D 5F80 A         RCPY AC2,AC3   ; STARTING  ADDS TX MSG
  842 082E 6B00 A         RADD AC0,AC3  ;   FINISHING "  "   " .
  843 082F 7301 A         CAI AC3,1     ; AC3 = -(LAST LOC. -1)
  844 0830 C200 A MFSFR1: LD AC0,(AC2) ; LD DATA
  845 0831 6900 A         RADD AC0,AC1 ; CREATE NEW ERR CDE.
  846 0832 7A01 A         AISZ AC2,1
  847 0833 D871 A         ST AC2,MSTADD ; STRE NXT LOC.
  848 0834 B073 A         ST AC0,@MSERAD ;  "  DATA IN F.E. STORE
  849 0835 5C80 A         RCPY AC2,AC0
  850 0836 68C0 A         RADD AC3,AC0  ; CURRENT ADDS -(LAST-1)
  851 0837 4204 A         BOC C2,MSNECS   ; BRANCH TO NEW ERR STRE
  852 0838 C073 A         LD AC0,MSERAD
  853 0839 7801 A         AISZ AC0,1
  854 083A D073 A         ST AC0,MSERAD
  855 083B 19F4 A         JMP MFSFR1    ; GET NEXT 16 BIT WD.
  856                 ;
  857 083C 6C40 A MSNECS: RXCH AC1,AC0
  858 083D B072 A         ST  AC0,@MSTERR ; STRE INDIRECT
  859                 ;
  860                 ;  NEW ERROR CODE
  861                 ;    "    DATA
  862                 ;   ALL LOADED IN MST MESG. LOC.
  863                 ;
  864                 ;
  865 083E 1907 A         JMP TXMXSB ; TRANS MSG.
  866 083F FC00 A MSTSLA: .WORD X'FC00 ; BASE SLAVE ADDS. PTR.
  867 0840 FF00 A MSTSEC: .WORD X'FF00 ; MSTR SECTOR ADDS MASK
  868                 ;
  869 0841 0760 A MSERRR: .WORD MSEROR
  870 0842 0718 A MSFETX: .WORD MSTFET
  871 0843 0783 A MSMTX1: .WORD MSMTRN
  872 0844 2000 A MSMPTR: .WORD MMSGST
  873 0845 0016 A MS2SCH: .WORD X'16 ;   SYN. CHARACTER
  874                 ;
  875                 ;  TRANSMIT STORED MESG-END WITH EOT.
  876                 ;
```

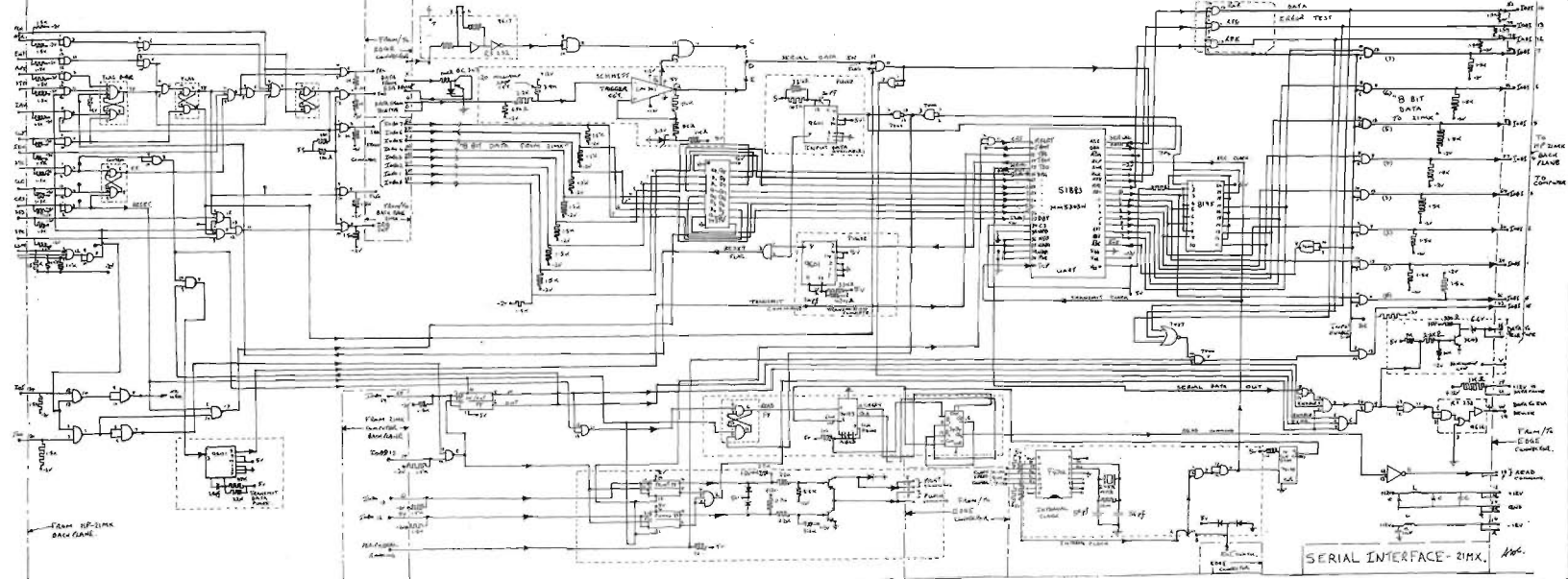
```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE   19
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  877 0846 CDB2 A TXMXSB: LD AC3,MSCD1A
  878 0847 D303 A         ST AC0,TTYOFF(AC3) ; TURN OFF CD1
  879 0848 CDB1 A         LD AC3,S2CD2A      ; CD2
  880 0849 D303 A         ST AC0,TTYOFF(AC3)
  881 084A CDAD A         LD AC3,MMSCD1
  882 084B D303 A         ST AC0,TTYOFF(AC3)
  883 084C 95F6 A         JSR @MSMTX1
  884 084D 5004 A TXMXAC: LI AC0,X'04         ; EOT
  885 084E 95F3 A         JSR @MSFETX
  886 084F 9900 A         JMP @MSTEXT
  887 0850 076F A MSTEXT: .WORD MSTIXT
  888 0851 00FF A MSTML1: .WORD X'00FF
  889                     ;
  890                     ;   LOAD MESG INTO REQ'D. MEM. AREA
  891                     ;
  892 0852 CDF1 A LDMXSB: LD AC3,MSMPTR ; SETUP LD MESG. PTR.
  893 0853 C300 A NXMSSH: LD AC0,(AC3)
  894 0854 EDA6 A         ADD AC3,MSTMNE
  895 0855 F1EF A         SKNE AC0,MS2SCH
  896 0856 19FC A         JMP  NXMSSH
  897 0857 F1A5 A         SKNE AC0,MSTSOH
  898 0858 1901 A         JMP NXLDLN
  899 0859 9916 A         JMP @MXLDEQ   ; MASTER ENQ ON MSG
  900                   ;
  901 085A C307 A NXLDLN: LD AC0,7(AC3)  ; LD MSG. LNGTH.
  902 085B A9F5 A         AND AC0,MSTML1
  903 085C 5F00 A         RCPY AC0,AC3
  904                   ;
  905                   ;    AC3= MESG, LENGTH (MAX 256)
  906                   ;
  907 085D C871 A         LD AC2,MSTADD
  908                   ;
  909                   ;    AC2= MESG DESTINATION
  910                   ;
  911 085E 6B80 A         RADD AC2,AC3
  912 085F DC71 A         ST AC3,MSTADD
  913                   ;
  914                   ;    MSTADD = LAST LOC.
  915                   ;     AC2   = FIRST LOC.
  916                   ;
  917 0860 CDE3 A         LD AC3,MSMPTR
  918 0861 C300 A MSDATS: LD AC0,(AC3)
  919 0862 ED98 A         ADD AC3,MSTMNE
  920 0863 F19A A         SKNE AC0,MSTSTX
  921 0864 1901 A         JMP MSSTDT
  922 0865 19FB A         JMP MSDATS      ; LOOK FOR STX
  923 0866 C300 A MSSTDT: LD AC0,(AC3)
  924 0867 D200 A         ST AC0,(AC2)    ; STORE DATA
  925 0868 ED92 A         ADD AC3,MSTMNE
  926 0869 E991 A         ADD AC2,MSTMNE
  927 086A F871 A         SKNE AC2,MSTADD
  928 086B 1901 A         JMP LDIXTS      ; JMP TO LD EXIT SUB.
  929 086C 19F9 A         JMP MSSTDT
  930 086D 5C00 A LDIXTS: RCPY AC0,AC0
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE   20
PACE  FTND CTRLPRGM 17/10/78
MASTER INTRPT. SUB.-IEN3


  931 086E 9900 A          JMP @TXMXC1
  932 086F 084D A TXMXC1:  .WORD TXMXAC
  933 0870 0760 A MXLDEQ:  .WORD MSEROR  ; ENQ ON LDNG
  934              ;
  935              ;
```

```
PACE ASSEMBLER REV-A      20 DEC 76                PAGE   39
PACE  FTND CTRLPRGM 17/10/78
SLV2IN INTRPT. SUB.-IEN5


1902              ;*                            *
1903              ;*****************************
1904              ;
1905              ;
1906              ;*****************************************
1907              ;*                                       *
1908              ;*                                       *
1909              ;*       MESG STORED.                    *
1910              ;*                                       *
1911              ;*        0000                           *
1912              ;*        0000                           *
1913              ;*         "                             *
1914              ;*         "       IGNORED NULLS         *
1915              ;*        0000                           *
1916              ;*        XXXX                           *
1917              ;*        XXXX    ALLOW TEN ERRORS IN MESG*
1918              ;*        XXXX                           *
1919              ;*        00SYN                          *
1920              ;*        00SYN   NO LIMIT ON NO.        *
1921              ;*        00SYN    ( CONTROL 'V')        *
1922              ;*        00SOH                          *
1923              ;*        00SOH    REQ. SIX  SOHS.       *
1924              ;*        00SOH                          *
1925              ;*        00XXX CHANNEL ADDS. TO NXT LEVEL*
1926              ;*        00SOH   SEPARATOR (CNTRL 'A')  *
1927              ;*  XX,YY  SECTOR ADDS.,MSG. LENGTH 16BIT*
1928              ;*        ERROR CODE.                    *
1929              ;*        00FS          (CNTRL 'SHIFT' L') *
1930              ;*        00TX OR 00LD TX='T'(X'54')/L('4C*
1931              ;*  TXTX; TXLD ; LDTX ;LD0   SLV CMNDS,  *
1932              ;*       SL1ADD PTR.  16 BIT CHNGE BY PRG *
1933              ;*        00STX   X'2 START OF TEXT (CL.B)*
1934              ;*        XXXX    DATA 16 BIT            *
1935              ;*        XXXX     "     DETERMINED BY LNG*
1936              ;*        00ETX   END OF TEXT            *
1937              ;*                                       *
1938              ;*   WILL EITHER TRANS ACK OR NACK AT END *
1939              ;*                                       *
1940              ;*    0 IS 8 BIT 0 IS 16 BIT WDS         *
1941              ;*    000 USE BRK KEY                    *
1942              ;*      EOT CNTRL 'D' & ETX CNTRL'C'     *
1943              ;*                                       *
1944              ;*****************************************
1945 0B6A                .=.+50
1946 0B9C 0000 A NXTSB2: .WORD 0
1947              ;
1948      00BD A          .END START
```

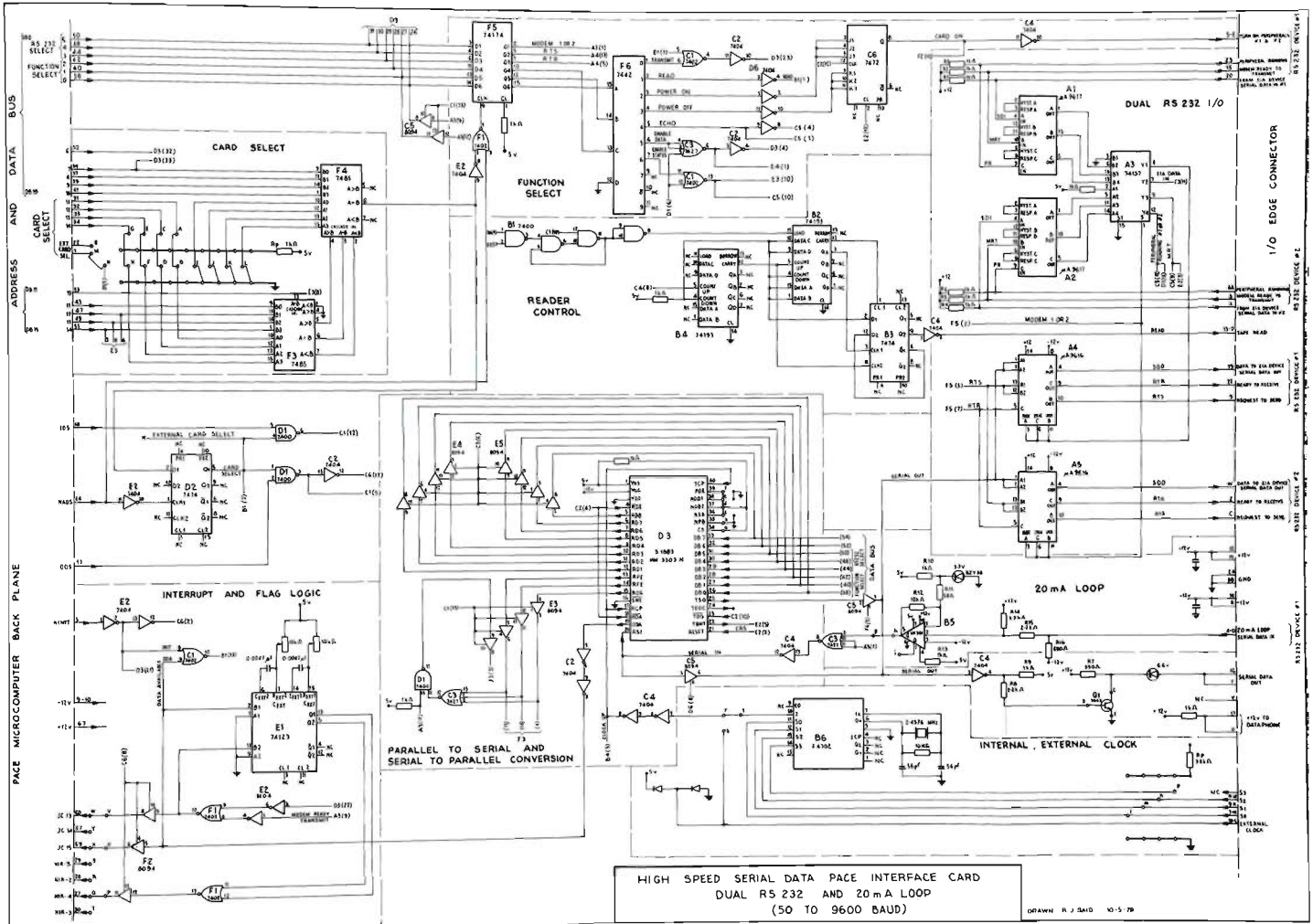
APPENDIX C : MASTER HIGH SPEED SERIAL DATA INTERFACE CARD FOR HP-21MX

- Circuit Diagram
- RS232C and 20 Milliamp Loop
- 50 to 9600 Baud


SERIAL INTERFACE - 21MX.

APPENDIX D : FRONTEND AND SLAVE HIGH SPEED SERIAL DATA INTERFACE CARD

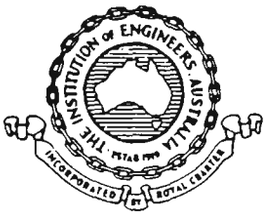
- Circuit Diagram
- Dual RS232C and 20 Milliamp Loop
- 50 to 9600 Baud


PACE MICROCOMPUTER BACK PLANE
ADDRESS AND DATA BUS
CARD SELECT
RS 232 SELECT
FUNCTION SELECT
CARD SELECT
FUNCTION SELECT
READER CONTROL
DUAL RS 232 I/O
I/O EDGE CONNECTOR
INTERRUPT AND FLAG LOGIC
PARALLEL TO SERIAL AND SERIAL TO PARALLEL CONVERSION
20 mA LOOP
INTERNAL , EXTERNAL CLOCK
HIGH SPEED SERIAL DATA PACE INTERFACE CARD
DUAL RS 232 AND 20 mA LOOP
(50 TO 9600 BAUD)
DRAWN R J BAID 10-5-79

APPENDIX E : PUBLISHED PAPER BY AUTHOR

"Hierarchical Control Using

Satellite Microprocessors"



Reprint

of a Paper Presented

at a Technical Conference

of

THE INSTITUTION OF ENGINEERS, AUSTRALIA

Hierarchical Control Using Satellite Microprocessors

A. W. COTTON

State Electricity Commission of Victoria

and

G. J. LOWE

Senior Lecturer, Footscray Institute of Technology, Victoria

SUMMARY The aim of this paper is to show some of the developments in hierarchial control and monitoring of large scale industrial installations. Past, present and future trends will be covered. Research work currently being undertaken at Footscray Institute of Technology, into the problems and benefits of a system using limited interconnection between satellite microcomputer controllers and a master minicomputer in a control environment will be used as the basis of the paper.

Advantages and disadvantages of present remote control and monitoring methods will be discussed as well as some of the pitfalls of implementing a microcomputer based scheme.

Both hardware and software requirements are presented and also general observations based on the authors' experiences. With falling microcomputer costs and increasing scales of operation in industrial installations serious consideration will have to be given to the methods of control and monitoring.

1 INTRODUCTION

With the development of large, complex industrial installations comes the need for a central control centre if overall co-ordination is to be maintained. One example is the development of power generation utilizing fossil fuels. As the size of the various power stations increased, so did the demand for fuel, hence the operations in the mining of the coal became more and more complex. In order to keep the bunkers of the power stations full, a network of conveyors is required to transport the coal from the dredgers (or excavators) in the bottom of the open-cut mine to the power station bunkers around the outside.

Initially the supply of coal was fairly easy to co-ordinate, but as the distances involved increased to kilometres and the transport equipment increased in size and numbers, it was found necessary to centralize the control and monitoring of the transport systems.

Generally, located remote from the working faces of the open cut mine, the control centre's function is to plan and control the activities on both a daily and long term basis. Without the control centre, this large and complicated transport system, would be most inefficient in its operation. The amount of time lost starting and stopping the long conveyor system is considerable, so the job of the control centre is to try and keep every item of plant in service and producing in the most efficient overall manner.

Briefly a remote control and monitoring scheme such as already mentioned, combines both computer and telemtry techniques in order to control the activities of a complex plant situation, consisting of manned dredgers and stackers and unmanned belt conveyors and pumping stations. This equipment must be controlled to avoid overfilling and spillage at the transfer points during start-up, normal operation and stopping.

In addition to this main control function there are monitoring functions which have to be carried out, in particular due to the unattended operation of the conveyors and pumping stations. In the case of the unattended plant the volume of information to be transmitted is much bigger than with the manned units.

The evaluation of technical and economical aspects shows the expediency of employing remote control and monitoring equipment. For the past 15-20 years the remote control and monitoring systems have generally used some form of multiplexing (frequency or time division, F.D.M., T.D.M.) or direct wiring to connect the remote outstations (conveyor drive units etc.) to the control centre which usually had a minicomputer to provide the overall strategy for control and monitoring the plant.

The control necessary for the individual machinery is placed on the unit itself in cubicles near the power circuits and equipment and normally operates at a lower voltage (240V a.c. or 110V d.c.) than the primary circuits. Initially the control logic was implemented using conventional relays, more recently (5-10 years) solid state logic has been used.

With falling hardware costs, especially in the field of small computers (microcomputers), and increasing plant size and complexity it is becoming feasible to implement the requirements of control and monitoring for large industrial installations by dedicating various functions to distributed computer at the remote outstations. The distributed computer would be linked together in a hierarchial control scheme with overall control still maintained at the control centre. Research is currently being carried out at Footscray Institute of Technology based on using the microcomputers as multiple slave computers in a distributed remote control and monitoring scheme. Initial results should be available late 1977 or early 1978.

A typical layout showing the remote outstations (at the conveyor drive ends, etc.) and the control centre is shown by Fig. 1. The diagram has been simplified by leaving out the interchange between the various levels within the open cut mine.

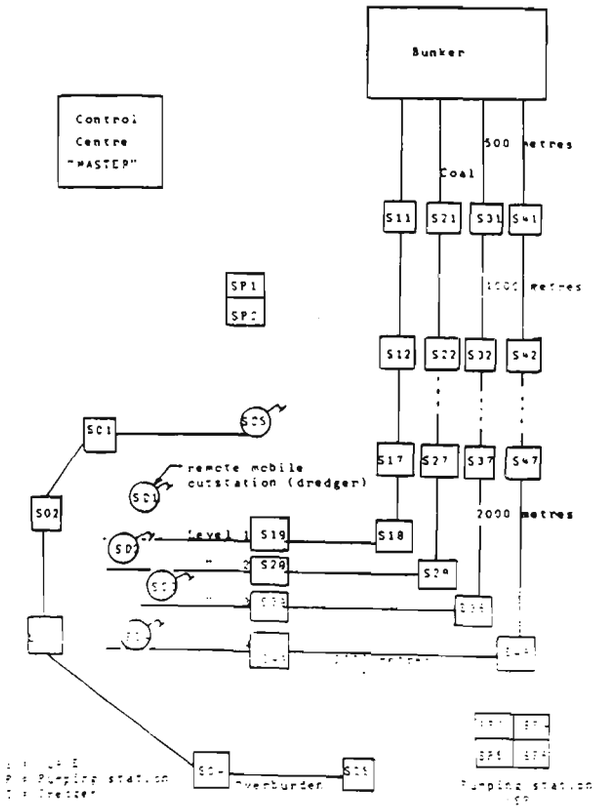


Figure 1 A typical "MASTER/SLAVE" layout

2 CURRENT METHODS OF CONTROL AND MONITORING

2.1 Control

The control requirements at each remote outstation are implemented by interlocking the various functions that have to be checked before a successful starting or stopping operation is carried out. Conventional relays were used to open or close contacts in a ladder diagram depending on the condition required. Extra contacts were provided on the same relay for monitoring purposes.

With the advent of solid state logic the control functions were then achieved in a similar manner with hardwired logical "and's" and "or's".

2.2 Monitoring

As mentioned in the introduction another important consideration is the method by which information will be sent to and from the control centre. There are in use today several ways of sending and receiving data.

2.2.1 Direct wiring

In direct-wired systems as the name suggests all of the signals are directly wired from the field to the control centre. Multi-cored control cables are generally used for connecting the terminal boxes on the plant to the main terminal strip in the control centre.

This method was one of the earliest used in transmitting the signals back to the control centre. It was quickly realised that, if thousands of signals have to be brought back many kilometres from remote locations, the cost and number of cables would become a very large proportion of the cost of an installation (generally in practice the life of a control cable is considered to be 15 years under good conditions) hence either a system is kept to a minimum or other methods should be assessed.

2.2.2 Multiplexing

Another technique for transmitting information is multiplexing (either frequency division or time division), the signals are sent over the same pair of wires but at different frequencies or different time intervals.

3 INTELLIGENT REMOTE OUTSTATIONS

Falling microcomputer hardware costs and rising hardwired logic system costs makes it economic to consider replacing the current method of remote control and monitoring in large installations. It is feasible to combine remote control, real-time data acquisition, distributed computing (hierarchical computer systems) low-cost communication links.

3.1 System Organisation

The individual remotes could be as shown in Figure 2.

3.2 Microprocessor Hardware

It is not possible to discuss in detail in this paper the hardware requirement for a control microprocessor and the following points simply indicate some of the major aspects.

3.2.1 Program storage-memory

Programs for the microprocessor are usually stored in ROM's as opposed to read/write store. The type of ROM used would be one of the following depending on the requirements of the system they are to be used in

(a) fuseable link (b) U-V erasable
(c) Mask programmed

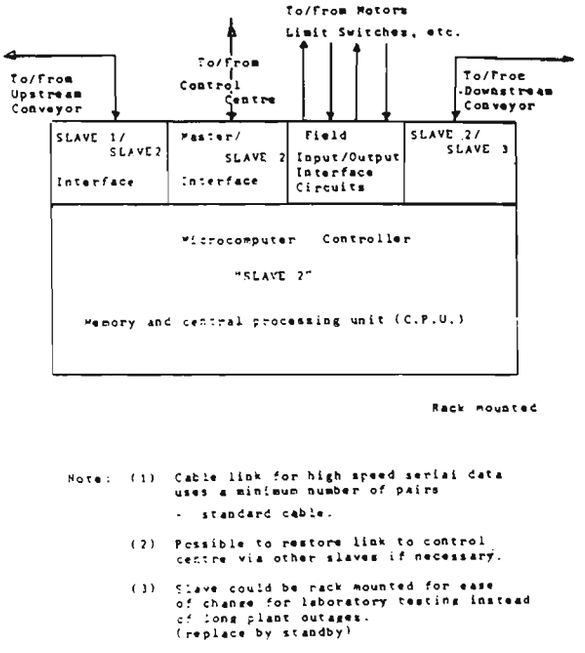


Figure 2 Interface requirements for a "Remote Outstation"

The dynamic memory of the outstation would use random access memory for storing counters, timers and data.

3.2.2 Interface requirements

In addition to the central processing unit (C.P.U.) and the memory a microcomputer controller will also require suitable interface circuits to provide the inputs and outputs during program execution. The interface decouples the primary circuits from the microprocessor by using either a relay interface or opto-couplers.

The main benefits of using microcomputers would be realized by connecting the remotes together by means of a simple serial interface as shown in Fig. 2.

The use of this serial interface simplifies the problem of compatibility of communications between computers and peripherals including different word lengths and bit rates.

The serial interface with which we are concerned in the research project is a self-synchronising type. It was developed so that it can be used with 20 milliamp current loop or the RS232 standards.

4 SYSTEM CONTROL STRATEGY

There are several system configurations currently being investigated as shown by Figures 3 and 4. Figure 3 shows the master computer maintaining overall control of the remote outstations via the front end microcomputers. For a system as shown in Figure 1 it is envisaged that each main route or flight of conveyor would have a front end microcomputer assigned to control the remote outstations associated with that line. The master would transmit or receive messages via the front end device. The second method under investigation as shown in Figure 4 would use the direct links between master and slave or possibly a secondary route through the upstream or downstream outstation.

4.1 The Communication System

Utilizing the serial interface as outlined, the communication system is designed so that the remote outstations will handshake with the master computer The master sends call messages of a particular station down the line and receives an acknowledgement. After this, instructions are to be sent to the remote computer requesting it to perform an operation or report the current status of the plant. The master station waits for a reply within a defined time, retransmission could then be carried out before the outstation is considered to be out of communication or another transmission route could be tried.

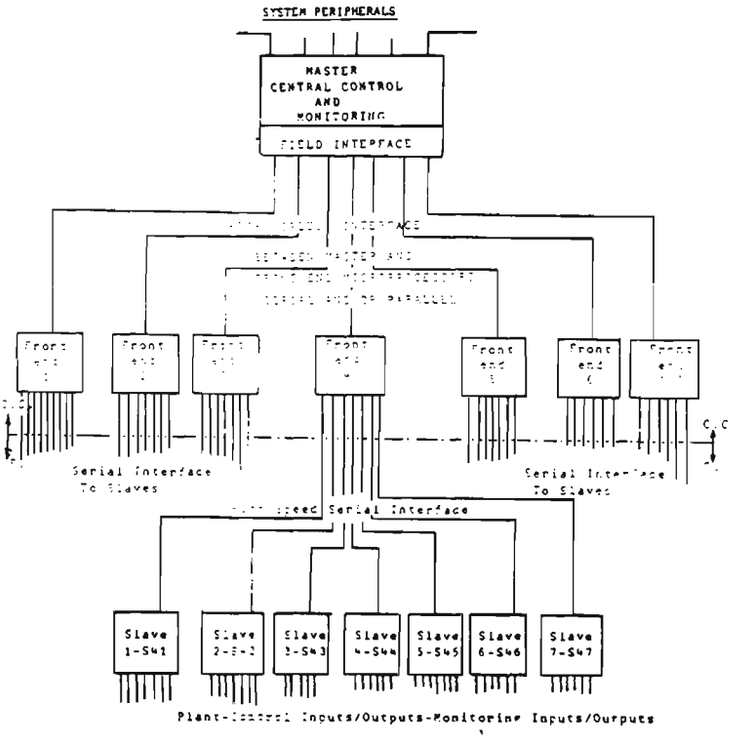


Figure 3 System Configuration - "FRONT END PROCESSORS"

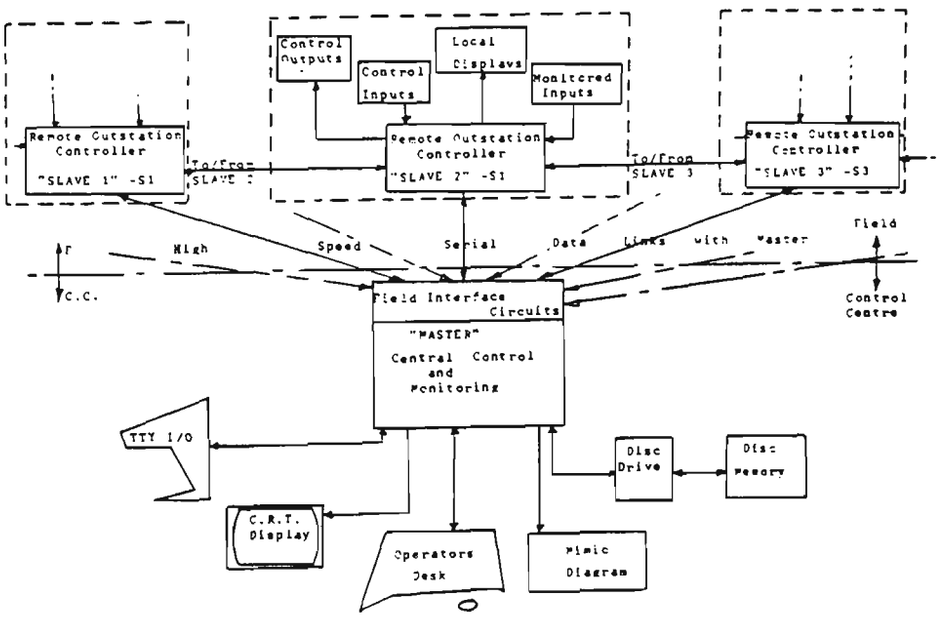


Figure 4 System Configuration - "DIRECT INTERFACING WITH SLA"

4.2 Mode of Operation

In systems as outlined in this study two basic modes of operation could be used, interrogation on interrupt (quiescent) or a continuous scan (or polling) of outstations.

In a quiescent system the remotes would be fairly inactive, stopping and starting motors, etc. under normal program execution but continually checking the previous status of the plant. This mode could be used to reduce the flow of data to the master computer. By comparing the status that it last transmitted to the master, which is stored in its own memory, the remote can determine if it is necessary to transmit to the master. The master will assume the same condition exists, if a remote has reported that a conveyor has stopped it will not report its status again, unless it tells the conveyor to start, or the master requests a complete status check.

The second mode to be considered is continuous scan, by definition, this method follows a predetermined routine in its acquisition of information Instead of permitting any station to report changes of status as they occur, a continuous scanning system permits each station to report only when it is "asked" to do so. In this way data transmission is completely controlled by the master station. However, continuously scanning systems have no way of giving priority to important information. Each critical alarm must wait until it is next scanned before the master station can act on that data. This is one of the major problems of the earlier systems using relay or solid state logic, direct connected or multiplexed. These earlier systems have no intelligence, hence there is no way of time sequencing events. A number of techniques have been employed to reduce the effect of this problem.

- faster scanning
- checking for the presence of altered data before scanning all data
- increased message efficiency
- system response improvements

4.3 Reliability and Security

The requirement to detect errors in data has been a concern of the industry since the first data acquisition and control systems were developed. With the countless error detection methods available what error-detection technique should be used. To determine the best method for a particular system the need should be clearly defined.

Basically the method employed should be efficient - the extra transmitted information required for error detection should be kept to a minimum. It should also be simple and economic - the error detection scheme should be relatively straightforward and the cost of all special equipment to be as low as possible.

Redundant data used for error detection purposes is one source of wasted time in large system. Any methods by which this wasted time can be reduced must be very carefully considered. It is proposed to investigate the method outlined in reference 1. In this method a simple error code detection on the master station to outstation is combined with a more powerful one using considerably more redundancy for the reverse direction - using a check-back before execute procedure, all control operations involving transmissions between master and slaves are subject also to the more powerful error detection code. Hence a reduced redundancy can be employed in one direction of transmission while still retaining a very low overall undetected error rate.

5 REQUIREMENTS FOR DEVELOPING A MICROCOMPUTER BASED SYSTEM

5.1 Hardware

During the development phase a convenient and economical means of expediting the development of software and hardware for the intended microprocessor system is required. Hence a prototyping or development system is needed which enables access to the microprocessor's registers, status flags and memory. There are several good development systems on the market which provided mass storage in the form of a floppy-disc operating system as well as high speed printouts and inputs.

For testing purposes a panel of switches and indicators or another microcomputer that simulates the process in which the microcomputer is to be used to control is essential. The use of a second microprocessor to simulate the plant providing repeatability is very useful during the debugging of the system.

5.2 Software

An often underestimated requirement is the software. The costliest part of a system using a microprocessor is the software. In choosing a microprocessor, it is essential to pay a lot of attention to the software. The following software is a prerequisite when considering available development kits.

5.2.1 Assembler

The assembler is used to convert mnemonic instructions into binary patterns. There are two general types of assembler packages available for microcomputers: (1) cross-assembler programs run on minicomputers (2) self assembler programs run on the microcomputer itself, usually in the form of a development system. Every microcomputer now produced has one or more accompanying cross-assemblers. With a teletype or CRT console, the user can type in his assembly language program. Self assemblers are written with a definite computer system in mind.

5.2.2 Editor

The editor is used to write, correct, and display a source program with a minimum amount of source handling. It also enables the generation of new source programs and the modification of existing sources programs in preparation for program assembly.

5.2.3 Loader

This software package is used to load the assembled programs into random access memory for test purposes before final programming of the ROM memory.

5.2.4 System software development techniques

It has been found from experience that the software requirements should be very well documented and also developed in small modules. Shortcuts and large programs leads to long debugging time and generally outweights the advantage of time saved (J. Hont 1976).

6 ADVANTAGES AND DISADVANTAGES

The microcomputer can be an ideal replacement for ordinary hardwired logic. Microcomputers offer both a new set of challenges and a new set of problems.

6.1 Flexibility

The microcomputer offers flexibility in system design because it enables the hardware designs to be carried out in parallel or earlier than the final software. Hence changes can be made to control strategy right up to the commissioning stage. The flexibility offered can also be a disadvantage since there are more factors to consider in the design of a scheme.

6.2 New techniques

With the introduction of intelligence at the remote outstations it could lead to a whole new set of techniques in plant control. Some of the possibilities are - control of maximum energy demand peaks- conveyor belt slip control - preprocessing of data, plus many others.

6.3 Speed

Though a microprocessor can execute any desired functions through a sequence of program steps, its limited instruction set makes that sequence long hence consideration must be given to execution time in real-time systems. Even when the same circuit family is involved, hardwired logic is faster.

7 CONCLUSIONS

The successful use of a microprocessor as a distributed computer controller in a large industrial installation will depend on careful assessment of the applications. In addition to the new benefits and problems associated with microprocessors there are many secondary ones that have not been considered in this paper (such as training of personnel) that must be included in the assessment.

The area of large scale hierarchical control and monitoring is one where the designer may find the microprocessor especially effective as an alternative to complex hardwired logic. Using a microprocessor can result in a standardisation of design approach, making it easier to get the design process under way. The microprocessor is typically capable of executing a large number of complex logic and arithmetic functions which would be costly to implement in hardware.

For anybody considering an industrial remote control and monitoring system in the future, a great deal of work will have to be put into the investigation stage of the project in order to determine what type of system will suit both the present and future requirements of the installation. Will it be the already proven hardwired - F.D.M., T.D.M. systems or will a system using distributed microcomputers be a better solution?

8 REFERENCES

GREENWAY, J.F., LAKHANI, A.H. and LOCKWOOD, D. (1971) Communication and error control aspects of a remote control system. Conference Publication No. 81, I.E.E. Sep.-Oct. 1971.

HONT, J. (1976), Review of Microprocessor Activities Microprocessor Newsletter, Telecom Australia Research Labs., Jan. 1976.

PROPHET, G. (1977), Distributed processing power- a new tool for production control. Control and Instrumentation, June 1977

SPEERS, G.S. (1973), Monitoring/Control by distributed computing. Datamation. Vol. 19, No. 8 August 1973, pp.47-49

TROXEL, D.E. (1975) Serial Interfaces for minicomputers. I.E.E.E. Transactions on Computers Vol. 24, No. 1, October, pp.1027-1028.

APPENDIX F : INDUSTRY STANDARDS FOR -

- MESSAGE PROTOCOLS
- ERROR DETECTION

F1 Brief Summary of Australian Standard Document (AS 1484)

. Part 1 - Basic mode control procedures. This part describes the implementation of the standard seven-bit character set for information interchange. It defines the formats for both the transmitted messages and the supervisory sequences which are part of the transmission control procedures. All control functions are to be performed by the use of ten specific transmission control characters.

. SOH (start of heading) - first character of a heading of an information message.
. STX (start of text) - used to precede text and terminate a heading.
. ETX (end of text) - terminates a text.
. EOT (end of transmission) - used to indicate the conclusion of transmission.
. ENQ (enquiry) - request for a response.
. ACK (acknowledge) - an affirmative response to the sender.
. DLE (date link escape) - used to change the meaning of a following character.
. NAK (negative acknowledge) - negative acknowledge.
. SYN (synchronous idle) - a signal from which synchronism may be achieved.
. ETB (end of transmission block) - for data block separation.
. BCC (block check character) - an error check character.

Examples of the format used with three types of messages:

a Format for the transmission of a message without identification -
STX --- Text of Message --- ETX (BCC)

b Format for the transmission of a message including an address block or heading for identification -

SOH --- Heading --- STX --- Text --- ETX (BCC)

c Format for the transmission of a heading or the identification of a remote outstation, etc -

SOH --- Heading --- ETB (BCC)

. Part 2 of the Standard covers the character structure for start/stop and synchronous transmission, defining the character structure for serial 'Asynchronous' and 'Synchronous' data transmission systems.

. Part 3 covers the requirements for error detection. This part defines one method of error detection and consists of an error check character (BCC) with the data in addition to the individual parity bits included with each data word.

. Part 4 defines the connector pin numbers for the interface between data terminal equipment and data communication equipment.

The fifth part of the Standard (AS 1484) covers the extensions to the control procedures in Part 1.

F2 INDUSTRY STANDARD PROTOCOLS

There are several protocols in use in industry today that meet the formats specified in the standards, such as AS 1484. Examples of these are 'BISYNC', 'DDCMP', 'SDLC' and the system protocol developed for the research project. A brief explanation of each protocol is included in the following chapters.

F2.1 Bisync - Character Oriented Protocol

One of the most widely used protocols in the industry is IBM's 'BISYNC'. It has been in use since 1968 for transmission between IBM computers and batch and video display terminals. 'BISYNC' is a character-oriented protocol and uses the special characters outlined in AS 1484 (STX, EOT, SYSN, etc) to separate the various fields of a message and to control the necessary protocol functions.

The overall format is as shown in Figure F1 below:

SYN SYN SOH --- HEADING --- STX --- TEXT ---- ETX ---- (BCC)

Figure F1 - 'BISYNC MESSAGE FORMAT'

The contents of the heading or identification block are defined by the user and the text portion of the message is variable in length, the length of which is also defined in the heading.

To detect and correct transmission errors, 'BISYNC' uses either vertical/longitudinal redundancy checks (VRC/LRC) or a cyclic redundance check (CRC) depending upon the information code being used. For 'ASCII', a VRC check is performed on each character and and (LRC) on the whole message.

If the code is 'EBCDIC', no VRC check is made, instead a CRC is calculated for the entire message, CRC-16 is used. See F 3.4 (ERROR CODE) for an explanation of CRC-16. If the block check character transmitted does not agree with the calculation by the receiver, several more attempts are made until the system determines that the transmission equipment is faulty, usually after a predetermined number of tries.

There are other protocols that have been developed that are similar to 'BISYNC' but they use different control characters.

F2.2 'DDCMP' - Byte Count Oriented Protocols

A study of 'BISYNC' shows that it is a fairly involved protocol with special procedures required to achieve transparent transmission and reception. 'DDCMP' protocol has been developed to solve some of the transparency problems. As shown in Figure F2, the format is similar to 'BISYNC' in that the message is broken into two parts; a header containing control information and a text body. Unlike, BISYNC, the header is necessary and forms the most important part of the message since it contains the message sequence numbering information and the character count. Each exchange of message starts with a message number from 0 to 255 depending on the number of the previous message. Whenever a station transmits a message, it assigns its next sequenced message number to that message. When an error occurs 'DDCMP' does not require an acknowledgement as the number assigned specified the sequence number of the last good message, (30).

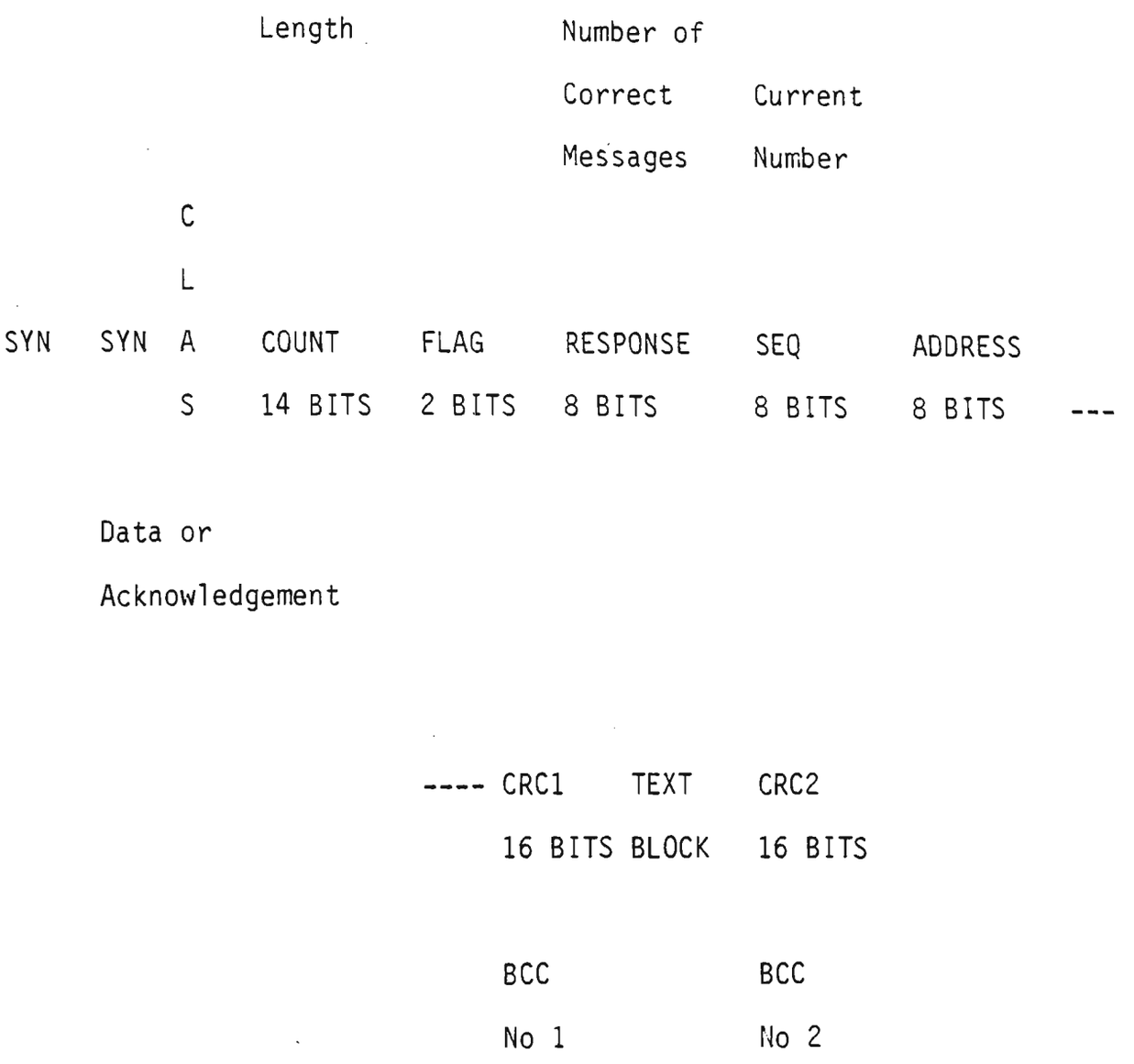


Figure F2 - 'DDCMP MESSAGE FORMAT'

'DDCMP' has two drawbacks; first, the header is relatively short, therefore, a system must have a buffer of the appropriate size ready on short notice. Secondly, the transmitting station must not include a 'sync' character in the middle of a message which would affect the character count, causing an error. The first problem is overcome by limiting the message length to a fixed length. The second is shared by other systems as well.

F2.3 'SDLC' - Bit Oriented Protocols

In 'SDLC' the information field is not restricted in format or content and can be of any length. The maximum length is determined by the length that can be expected to be received error free. The format for'SDLC' is as shown in Figure F3.

BEGINNING	ADDRESS	CONTROL	INFORMATION	FRAME	END
FLAG	8 BITS	8 BITS	ANY NUMBER	CHECK	FLAG

Figure F3 - 'SDLC FORMAT'

Two flags (two blocks of five '1's') one at either end of the message are used as reference points for the information contained within the message, address and control fields and are used to initiate the error checking. The end flag indicates that the last 16 bits is the error check.

The address field is eight bits long and indicates the destination for the message. The control field can have three formats such as information transfer format, supervisory format and nonsequenced format, (30).

The information format is used for data transmission and uses sequence numbering. The supervisory format is used in conjunction with the information format to initiate and control data transfer in the information format. The nonsequenced format is used for initialising stations.

'SDLC' is simpler than some of the other protocols but the error check calculations are more involved.

F3 : ERROR DETECTION

Data communication systems are subject to the introduction of errors by transient voltage interference from a variety of external sources, for example lightning, switching and crosstalk from other communications lines. With the numerous error detection methods available it is difficult to determine what error-detection technique should be used for a particular system.

Basically, the method employed should be efficient with the extra transmitted information required for error detection kept to a minimum. Redundant data used for error detection purposes is one source of wasted time in large systems. One method to reduce the data transmitted would be to have a simple error code detection on the master station to outstation message combined with a more powerful one for the reverse direction using a checkback on the received message before executing the procedure; hence a reduced redundancy can be employed in one direction of transmission while still retaining a very low overall undetected error rate, (19).

A common form of noise in transmission systems is burst noise. A noise burst of 100 milliseconds duration occurring during a 1200 bit per second transmission could corrupt 120 bits of data.

The periods of high error rate are generally separated by long intervals of low noise. Studies carried out by the authors of references (19) and (23) have indicated an error rate of between 10^{-4} and 4×10^{-5} for burst noise.

The following paragraphs explain some of the error detection techniques that are used in data transmission systems.

F3.1 PARITY TECHNIQUES

To determine if the bits of a character have been correctly received, it is quite simple to append an additional bit to each character. The parity technique sets the additional bit according to the rule that all characters shall have an odd or even number of ones (odd or even parity). In a parity system the device summates the ones and adds the appropriate bit during transmission. The receiver compares the parity bit expected with the received value and then decides if the data has been received in error. Parity is commonly included in LSI chips such as the UART. The limitation of parity is that it can only detect single (or three or five, etc) errors and this applies to odd or even parity.

F3.2 LONGITUDINAL REDUNDANCY CHECK (LRC)

A second form of error checking is parity on the columns known as Longitudinal Redundancy Check (LRC) which is also subject to incorrect checks because it is possible to have a double error in a column. There are numerous possibilities for double bit errors in characters to occur simultaneously with double bit errors in columns such that neither vertical redundancy check (VRC) nor (LRC) will indicate that the errors have occurred.

F3.4 CYCLIC REDUNDANCY CHECK (CRC)

The detection system most effective at detecting errors in communications systems with a minimal amount of hardware is the cyclic redundancy check (CRC).

CRC calculations are usually done in multiple exclusive OR's within hardware or software. The three most commonly used CRC codes are CRC-12, CRC-16 and CRC-CCITT. (20)

To evaluate the effectiveness of a cyclic coding error checking system, the data message must be considered in a mathematical form. The form most commonly used is a polynomial with a dummy variable term χ. The least significant bit (LSB) is χ° or 1, and the highest order term, representing the most significant bit (MSB) is χ^n. The coefficients of the polynomial indicate whether an individual bit is a '0' or '1'. Thus a data stream of 10 bits can be described as -

$$G(\chi) = 1110101101 \quad \text{(a random selection)}$$

$$= \chi^9 + \chi^8 + \chi^7 + \chi^5 + \chi^3 + \chi^2 + 1$$

The code polynomial $P(\chi)$ can be described in a similar way to $G(\chi)$, for example -

$$P(\chi) = \chi^5 + \chi^3 + \chi^2 + \chi + 1$$

$$= 101111$$

To generate the check bits for transmission the data polynomial $G(\chi)$ is divided by the code polynomial $P(\chi)$ to obtain a remainder. This remainder will be of a degree one less than that of the code polynomial, and will consist of the same number of bits as $P(\chi)$. The data block is followed by the remainder block in transmission. The message will be exactly divisible by the code polynomial at the receiver if no errors were introduced. (23)

The polynomial division of $G(\chi)$ by $P(\chi)$ is as follows:

$$G(\chi) = Q(\chi) \times P(\chi) + R(\chi)$$

Where $Q(\chi)$ is the quotient and $R(\chi)$ is the remainder. This gives -

$$G(\chi) - R(\chi) = Q(\chi) \times P(\chi)$$

Since $R(\chi) = -R(\chi)$ (modulo 2)

$$G(\chi) + R(\chi) = Q(\chi) \times P(\chi)$$

Before the division by $P(\chi)$ the data $G(\chi)$ is multiplied by χ^n (n is the degree of $P(\chi)$).

$$\chi^5 \; G(\chi) = \chi^{15} + \chi^{13} + \chi^{12} + \chi^{10} + \chi^8 + \chi^7 + \chi^5$$

This result is divided by $P(\chi)$ to give -

$$\chi^4 + \chi^2 + \chi + 1$$

Adding to $\chi^5 \; G(\chi)$ gives the message polynomial : $F(\chi)$

$$F(\chi) = \chi^{14} + \chi^{13} + \chi^{12} + \chi^{10} + \chi^8 + \chi^7 + \chi^5 + \chi^4 + \chi^2 + \chi + 1$$

If errors are introduced into the message resulting in an error polynomial $E(\chi)$ the received polynomial $H(\chi)$ becomes:

$$H(\chi) = F(\chi) + E(\chi)$$

The received polynomial $H(\chi)$ is divided by $P(\chi)$, the remainder will be $E(\chi)/P(\chi)$. Hence, a non-zero remainder indicates the presence of errors.

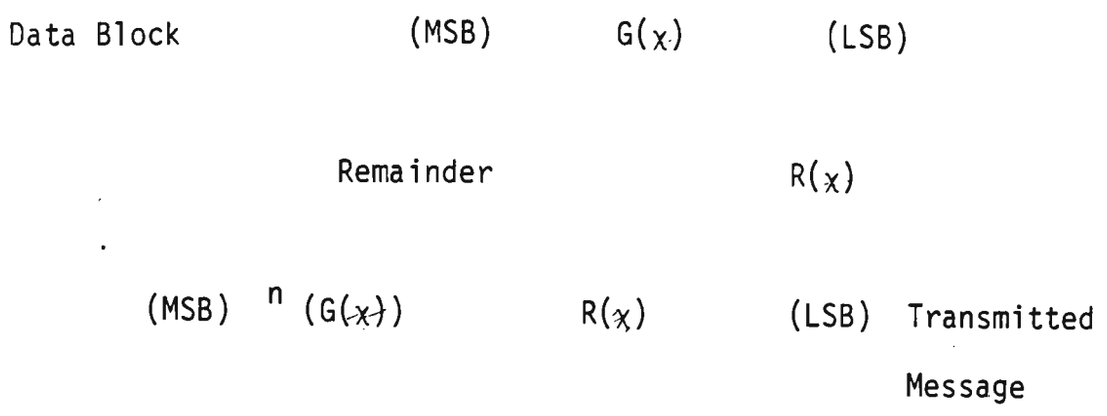


Figure F4 - "TRANSMISSION MESSAGE - ARRANGEMENT OF DATA STREAM"

There are many variations that can be chosen, generally the user selects a polynomial best suited to the kind of errors that occur in the system, (21).

CRC-CCITT and CRC-16 are used to detect bursts of 16 bits or longer to 99.998% efficiency (23) CRC-12 detects bursts longer than 12 bits with an efficiency of 99.975%, (23).

CRC-CCITT is used for European systems and when operating with eight-bit characters, the block check character (BCC) is 16 bits. The code polynomial 'P(χ)' for CRC-CCITT is $\chi^{16} + \chi^{12} + \chi^{5} + 1$.

CRC-16 is applied to synchronous systems that use eight-bit characters. The code polynomial P(χ) is $\chi^{16} + \chi^{15} + \chi^{2} + 1$.
CRC-12 is used with six-bit systems and uses $(\chi^{12} + \chi^{11} + \chi^{3} + \chi^{2} + \chi + 1)$ as the code polynomial P(χ).