

https://dspace.org/

The provision of microprocessor-based systems development facilities

Finucci, Agostino Piermaria Gabriele

2015-03-12
http://hdl.handle.net/10539/17261

Downloaded from a DSpace repository. https://dspace.org/

THE PROVISION OF MICROPROCESSOR-BASED [illegible] FACILITIES

Agostino Piermaria Gabriele Fin[illegible]

A Project Report submitted to the Faculty of Engineering
University of the Witwatersrand, Johannesburg
in partial fulfilment of the requirements for the award
of the Degree of Master of Science in Engineering

Johannesburg 1984

ABSTRACT

This project consisted of the planning and installation of facilities within the Department of Electrical Engineering at the University of the Witwatersrand for the development of microprocessor-based systems.

The project work covered the following areas: familiarisation with the principles of microprocessor-based systems and their development requirements; evaluation of the requirements to be met by development facilities; formulation of a policy for providing these development facilities; acquisition, configuration and integration of the development facilities; and final evaluation of the capabilities of the acquired development facilities.

The policy for the provision of development facilities which was employed implied a configuration which consisted of a central general-purpose computer system which would allow multi-user access to the development facilities. To provide functions which the central system could not perform a number of special-purpose stations would be connected to it. These stations would consist of commercially available development systems.

The project was undertaken in two one-year phases, each consisting of a complete cycle starting with the evaluation of requirements and ending with the evaluation of capabilities.

DECLARATION

I declare that this project report is my own unaided work. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

22 day of FEBRUARY, 1984.

DECLARATION

I declare that this project report is my own unaided work. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

22 day of FEBRUARY , 1984.

PREFACE

At the end of 1980 the university's Department of Electrical Engineering committed itself to upgrading its facilities for the development of microprocessor-based systems. The author was commissioned to plan and co-ordinate this as a basis for a Master of Science degree.

Acknowledgements

The author would like to acknowledge the considerable help he received from the following individuals:

Dr. I.M. MacLeod, Senior Lecturer, Department of Electrical Engineering, University of the Witwatersrand.

Prof. M.G. Rodd, Head of the Department of Electrical Engineering, University of the Witwatersrand.

He also received assistance from individuals representing the following companies:

AECI Ltd
Alpret (Pty) Ltd
Biowave Electronics (Pty) Ltd
CAP SA (Pty) Ltd
EBE (Pty) Ltd
Hewlett-Packard SA (Pty) Ltd
Protea PNI (Pty) Ltd
Ketecon Electronics (Pty) Ltd
SPL International (UK)
Systime SA (Pty) Ltd
Tek Logic (Pty) Ltd

CONTENTS

1.0 INTRODUCTION

1.1 Microprocessor-based System Development

1.2 Project Objective and Limitations

1.3 Organisation of Dissertation

2.0 DEVELOPMENT FACILITIES FOR MICROPROCESSOR-BASED SYSTEMS

2.1 The need for development facilities

2.2 Terminology of development facilities

2.3 Microprocessor-based systems development methods

2.3.1 Conception

2.3.2 Specification

2.3.3 Design

2.3.4 Coding

2.3.5 Translation

2.3.6 Integration

2.3.7 Validation

2.3.8 Operation

2.3.9 Revision

2.3.10 Discontinuation

2.4 Microprocessor-based systems development tools

2.4.1 Specification

2.4.2 Design

2.4.3 Coding

2.4.4 Translation

2.4.5 Integration

2.4.6 Validation

2.5 Microprocessor-based systems development environments

2.5.1 User interface

2.5.2 Host Operating System and Utilities

2.5.3 Hardware components

2.6 Conclusion

3.0 REQUIREMENTS FOR DEVELOPMENT FACILITIES

3.1 Development Model

3.2 User's Project Requirements

3.3 Characteristics of users

3.4 Target System

CONTENTS

3.4.1 Complexity
3.4.2 Architecture
3.5 User's Project Solution
3.5.1 Implementations employing High-Level Languages
3.5.2 Implementations employing Assembly Language
3.6 Problems encountered by the users
3.6.1 Programming Inability
3.6.2 Lack of programming concepts
3.6.3 Hardware/Software interaction problems
3.6.4 Hardware design faults
3.6.5 Complexity
3.6.6 Unavailability or unsuitability of Tools
3.6.7 Misuse of Tools
3.6.8 Access to development facilities
3.6.9 Time constraints
3.6.10 Overcommitment
3.7 Requirements for the development facility
3.8 Conclusion

4.0 POLICY FOR THE UPGRADING PROCESS
4.1 Current and Future Trends
4.1.1 Functionality
4.1.2 Configurations
4.2 Expansion Policy
4.2.1 Description
4.2.2 Review
4.3 Implementation difficulties
4.3.1 Debugging methods
4.3.2 Debugging Demands
4.4 Conclusion

5.0 THE UPGRADING PROCESS PHASE I
5.1 Initially available facilities
5.1.1 Intel facilities
5.1.2 Motorola facilities
5.1.3 General-purpose computing facilities
5.1.3.1 University Computer centre
5.1.3.2 In-house Minicomputer facilities
5.1.3.2.1 Data General Eclipse system

CONTENTS

5.1.3.2.2 Systime 5000 system
5.2 Statement of requirements
5.2.1 Programming environment for Intel 8080/5 and Zilog Z80
5.2.2 Chip-level debugging facilities for the Intel 8085
5.2.3 General-purpose device programming facilities
5.2.4 Intel 8086-based development support
5.3 Availability review
5.4 Acquisition of components
5.4.1 Programming environment for Intel 8080/5 and Zilog Z80
5.4.2 Chip-level debugging facilities for the Intel 8085
5.4.3 General-purpose device programming facilities
5.4.4 Intel 8086-based development support
5.5 Configuration of components
5.5.1 Physical configuration of components
5.5.2 Configuration of emulation stations
5.5.2.1 Hardware components
5.5.2.2 Software components
5.5.3 Configuration of the 8080/5 and Z80 programming environment
5.5.4 Configuration of device programming facilities
5.5.5 Configuration of Intel 8086 development environment
5.6 Evaluation of capabilities
5.6.1 Programming environment for 8080/5 and Z80
5.6.2 Chip-level debugging facilities for 8085
5.6.3 General-purpose device programming facilities
5.6.4 Intel 8086-based development support

6.0 THE UPGRADING PROCESS PHASE 2
6.1 Statement of requirements
6.1.1 Host storage capacity increase
6.1.2 High level language support for Intel 8085
6.1.3 Extension of chip-level design support
6.1.4 Increase in user friendliness
6.2 Availability review
6.3 Acquisition of components
6.4 Configuration of components
6.4.1 Physical configuration
6.4.2 Development host system

CONTENTS

6.4.3 HP64000 Development system
6.4.4 Integration of facilities
6.5 Evaluation of capabilities
6.5.1 Host storage capacity increase
6.5.2 High level language support for Intel 8085
6.5.3 Extension of chip-level design support
6.5.4 Increase in user friendlyness

7.0 EVALUATION, RECOMMENDATIONS AND CONCLUSIONS
7.1 Evaluation of results
7.1.1 User requirements
7.1.2 Target system requirements
7.1.3 Project S .ution requirements
7.1.4 Problems encountered by users
7.1.5 General requirements
7.2 Recommendations for future upgrading
7.2.1 Use of facilities
7.2.2 Integration of facilities
7.2.3 Upgrading of development methods
7.3 Conclusions

APPENDIX A Intermediate Report - Initial Assessment

APPENDIX B Manufacturers of Development Components

APPENDIX C Intellec Disk Labelling and Naming Conventions

APPENDIX D XFR - File Transfer Utility

CHAPTER 1
INTRODUCTION

The microprocessor has become one of the most widely used components in digital systems because of its ability to perform a wide range of complex functions. Also, user programmability provides extensive implementation flexibility.

This programmability costs little in terms of components but can be expensive in terms of the development of the programs needed to implement the required functions. In many instances the decrease in component cost of microprocessor-based systems, when compared with the previous generations of programmable systems, has been largely offset by an increase in software development costs.

This project addresses the development support problem. Much has been said about microprocessor-based system development facilities, primarily by the vendors of microprocessor development systems. But little or no attempt has been made either to provide a comprehensive solution to the problem or even to adopt a cohesive approach to its solution.

1.1 Microprocessor-based System Development

Microprocessor-based systems,are performing a much more varied range of functions than other programmable systems. This makes the task of developing programs much more difficult. This difficulty is compounded by the fact that the majority of microprocessor-based systems do not incorporate the development facilities found in typical mini- and mainframe computer systems. In the case of microprocessors, these facilities usually have to be provided by a separate system.

Some of the facilities provided for other programmable systems can be used; however these facilities will not be capable of solving the special problems which occur in microprocessor-based systems. It should be noted that many of the methods and approaches to systems development used for other programmable systems are not employed in the development of microprocessor-based systems and have been overlooked by many development system vendors.

At present the provision of facilities for the development of microprocessor-based systems is a new, poorly developed and fast changing field. Although it is extremely important to provide the most modern and capable facilities, this is not easily achievable. This is due largely to the fact that it is usually difficult to distinguish between useful capabilities and those included into development systems merely to sell the product. Many capabilities provided by vendors fulfill needs that are not, and may never be, felt.

The Department of Electrical Engineering at the University of the Witwatersrand committed itself at the end of 1980 to upgrading its existing facilities. The author was called upon to supervise this undertaking. This required the application of many different areas of knowledge within the field of Computer Engineering, ranging from fundamental concepts of software production to configuration of computer hardware.

1.2 Project Objective and Limitations

The objective was to configure and expand the department's facilities for the development of microprocessor-based systems.

In order to work towards this objective, familiarity with the field of microprocessor-based system development had to be gained. Thereafter the requirements of the department's development facilities had to be identified.

This project report documents this upgrading process by outlining the actions performed during the course of the project, and their motivation.

The material covered in this project report is limited to experiences within the department in the support of systems based on conventional microprocessors, and does not consider the support of microprogrammable devices; although some of the basic principles are applicable to this field as well, as shown by Wild (1983).

This implies that this project report is largely oriented towards the support of development of microprocessor-based systems in an educational environment, even though a number of industrial systems are developed within the department as well.

CHAPTER 2

DEVELOPMENT FACILITIES FOR MICROPROCESSOR-BASED SYSTEMS

Before describing the characteristics of development facilities for microprocessor-based systems, it is important to discuss why the need for these exists. This chapter looks at this aspect and proceeds to introduce the concepts and terminology used in the description of development facilities. Thereafter, the essence of the various development methods will be described, prior to outlining how the development facilities provide the means for using these methods. A good introductory coverage of this material in simpler and more specific terms is given by Tseng (1982a).

2.1 The need for development facilities

Microprocessor-based systems are usually used in applications where the cost of the hardware is very low. This is possible because the components that make up a microprocessor based system are relatively inexpensive. This means that the typical microprocessor-based system cannot have components within it which are used only during development, especially if the cost of these components is a significant portion of the total system cost.

In a microprocessor-based system the complex functions are performed by by the programs which are developed for the system (i.e. the software) and not directly performed by the hardware circuits and components. This results in a situation where the cost of hardware is very low because mass-produced components are used, whilst the cost of software is entirely dependant on the capabilities of the developers and the methods used for development because the software is redeveloped for each application. This means that in these systems software development is the major component of system development, and the development facilities are usually aimed primarily towards efficiency of software development.

The majority of programmable systems which are not microprocessor-based are more expensive and are used where ongoing user programmability is essential as is the case in most mini- and mainframe computer systems. These systems will typically have their development facilities totally integrated into the structure of the system. Most microprocessor-based systems are used in applications where the development is not ongoing, but is only performed during the initial stages of the system life-cycle or for occasional software revision. These dedicated applications require the addition of special components (used only for development and maintenance) for a relatively short portion of the system's life-cycle and only to a small number of the systems produced.

The need for development facilities

Development facilities for microprocessor-based systems are therefore not normally integrated into the systems themselves, and in this respect, they can be very different to the development facilities used in mini- and mainframe computers. The components which are needed to support the development of a system will only be connected to the system whilst it is being developed. The components which need to be connected to the system under development will allow the development facilities to perform their function.

2.2 Terminology of development facilities

The methods used for the development of the hardware in microprocessor-based systems are virtually identical to those employed in other user-programmable electronic systems. Software development forms the major part of the development effort because the cost of the software component in systems has become much larger than the hardware component and for this reason, the rest of this chapter will only consider the software development.

The microprocessor-based system under development is termed the "target", whilst the development facilities are provided by a number of computer systems which are called the "development hosts". In most cases, only one host is required. In other words, the development facilities are hosted on a number of computer systems and they are oriented towards supporting the development of a number of types of target systems.

A development facility consists of a number of very different components, some of which are hardware components, and others are software components. All these components together perform the functions of the development facilities. The software components consist of a set of programs which perform various functions which allow the development process to be partially automated. The software components rely in turn on the the hardware components to enable these programs to execute.

A development facility is usually based on a user-programmable computer system. Some development facilities are a based on a number of user programmable computer systems.

The computer systems upon which development facilities are variants of the typical micro-, mini- or mainframe computer systems and therefore are comprised largely of the same components.

2.2 Terminology of development facilities

The methods used for the development of the hardware in microprocessor-based systems are virtually identical to those employed in other user-programmable electronic systems. Software development forms the major part of the development effort because the cost of the software component in systems has become much larger than the hardware component and for this reason, the rest of this chapter will only consider the software development.

The microprocessor-based system under development is termed the "target", whilst the development facilities are provided by a number of computer systems which are called the "development hosts". In most cases, only one host is required. In other words, the development facilities are hosted on a number of computer systems and they are oriented towards supporting the development of a number of types of target systems.

A development facility consists of a number of very different components, some of which are hardware components, and others are software components. All these components together perform the functions of the development facilities. The software components consist of a set of programs which perform various functions which allow the development process to be partially automated. The software components rely in turn on the the hardware components to enable these programs to execute.

A development facility is usually based on a user-programmable computer system. Some development facilities are a based on a number of user programmable computer systems.

The computer systems upon which development facilities are variants of the typical micro-, mini- or mainframe computer systems and therefore are comprised largely of the same components.

The hardware components of these systems include a Central Processor Unit, a Primary Storage System (Memory), a Secondary Storage System (Disks), a Tertiary Storage System (Archives), a user interface terminal (usually with CRT and Keyboard), a hardcopy unit (Printer) and an interface to the system under development.

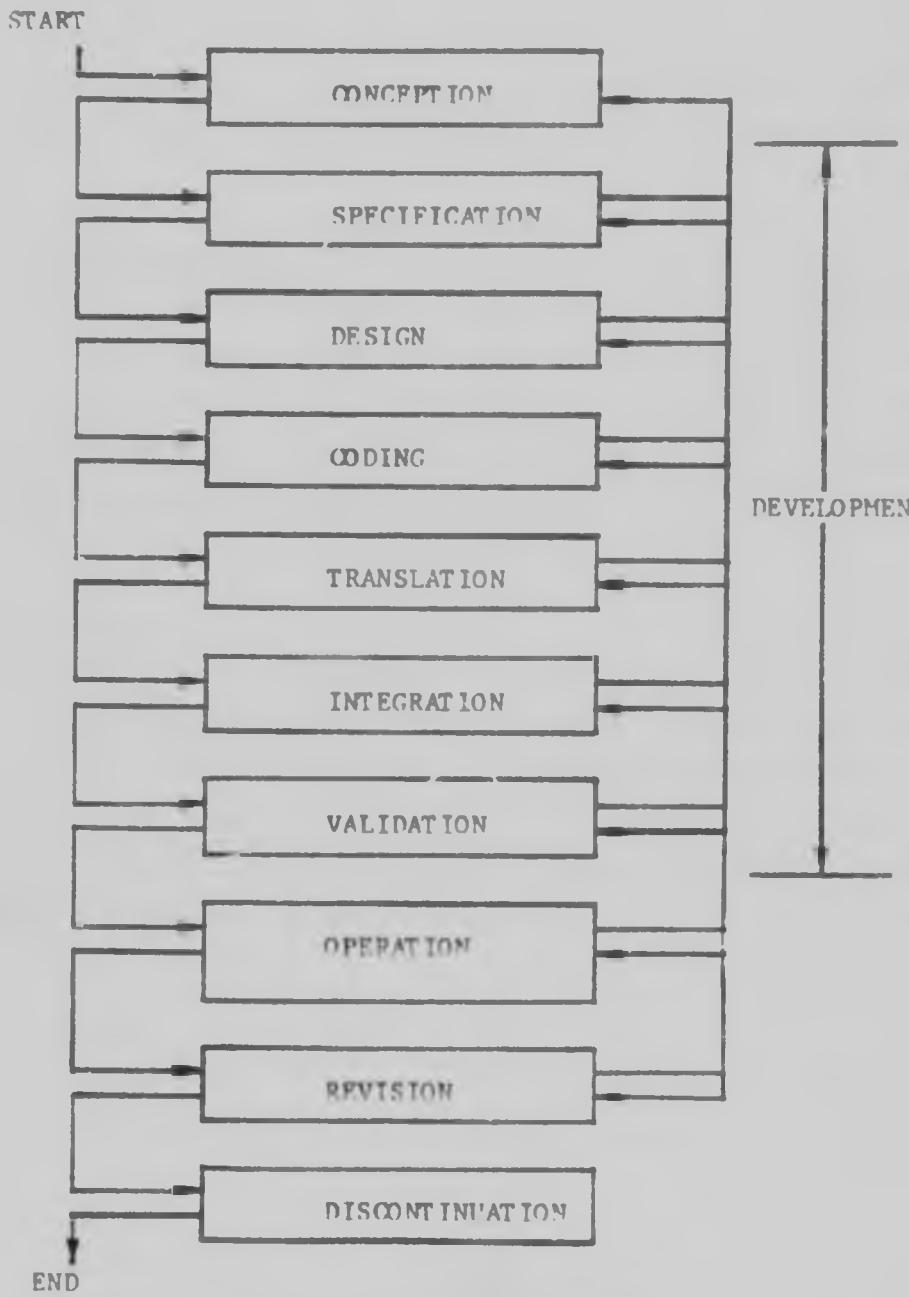
The use of the hardware components is managed and facilitated by the most important software component of all, the host operating system. The operating system controls the access by the users to the other software and hardware components. In systems where there are multiple users the operating system allows the use of the various components shared amongst them. The operating system itself is made up of a number of different programs and utilities, incliding programs which allows a user to interface directly to the operating system and request the use of the various facilities.

A "capability" is the property of a development facility which allows a certain development method to be used. Development facilities possess the capabilities for supporting a number of development methods. An example of a capability of a development facility is throughput. A development facility with a large throughput can perform an intricate set of operations in a short time. This means that this development facility can support the use of complicated and comprehensive methods which ideally, will make the development process easier for the user.

The components of a development facility that directly perform a function which is required by a particular development method are called the "development tools". Many functions are not directly required by a development method but are needed by the development tools or by the developer so as to facilitate the use of the development facilities. These functions are performed by components which are collectively termed the "development environment".

2.3 Microprocessor-based systems development methods

The majority of methods employed to develop software for microprocessor-based systems are characterised by the fact that they go through the following stages in the life-cycle of the system.



All systems will typically go through all of the above stages during their life cycles along the path shown on the left, but some stages or groups of stages will be repeated by tracing sections of the path along the right. The stages from Specification through

to Validation are called the development stages. A description of each stage in the life-cycle is given below:

2.3.1 Conception

This is the first stage in the life-cycle of a system and is where the system is first thought of in terms of satisfying an identified need. Typically a desirability study is done at this stage.

2.3.2 Specification

The functions to be performed by the system are established in this stage, ideally without implying any implementation method. In practice however, the system has to be specified in terms of some of the aspects of the implementation. A feasibility study is usually also performed at this stage.

2.3.3 Design

In the design stage, the implementation method is developed and the system specification used to derive a design specification which describes the solution to the implementation problem. This specification usually takes the form of a number of descriptions of program modules which implement all the identified, separable functions performed by the system.

2.3.4 Coding

In this stage, the programs for the microprocessor-based system are formally drawn up in a programming language.

2.3.5 Translation

The programs which have been coded are now translated into the code used by the microprocessor. This process is normally done automatically by a translator program, but sometimes it has to be done manually.

2.3.6 Integration

The various code modules are connected together to form the total software system during the integration stage.

2.3.7 Validation

The operations performed by the programs and thereby the functions performed by the system are analysed, verified and tested at this stage. For software, due to the fact that the solution cannot be formally developed or verified by means of established theories, this tends to be the most time and effort consuming stage in the development.

2.3.8 Operation

This stage will typically last the longest of all the stages, as the system is used and produces the required results.

2.3.9 Revision

In programmable systems it is quite common that a system's functions are altered after it has become operational, so as to improve its functionality. This is possible since the functions are largely performed by the software and therefore, should be relatively easy to change. Revision usually implies the undertaking of a limited development cycle.

2.3.10 Discontinuation

At this stage the use of the system is terminated and it ceases to exist as a system, even though some of its software and hardware components may be used in other systems.

2.4 Microprocessor-based systems development tools

Manufacturers of development facilities provide tools to support the development methods which they feel are most appropriate for the intended application. These methods usually differ only in various details, and the large majority of the methods go through the stages described in the previous section. The tools which are provided to aid the developer in performing the actions in the various stages of the development cycle are typically the same tools for all development methods. The differences in the development methods are directly attributable to the differences in the nature of the tools that are used in a certain stage of the development rather than what tools are provided.

The approaches to the provision of tools by various development facility manufacturers are outlined in the following publications. Conings (1982) describes Intel's approach which is characterised by the provision of integrated hardware and software development support tools. Motorola's policy for provision of tools, as described by Kister (1982) is pitched at a higher level of support oriented towards developers who are using board-level modules. Lee (1982a) puts forwards Hewlett-Packard's strategy which is strongly oriented towards providing a capable user interface for the development tools whilst also optimising development throughput.

The following subsections summarise the tools that are typically provided to aid the developer at each of the development stages in the life-cycle.

2.4.1 Specification

Specification tools are not being used by microprocessor-based systems developers to any significant extent, although it is the author's opinion that in the future, as the complexity of these systems increases, the use of specification tools borrowed from other development facilities will undoubtedly become common.

2.4 Microprocessor-based systems development tools

Manufacturers of development facilities provide tools to support the development methods which they feel are most appropriate for the intended application. These methods usually differ only in various details, and the large majority of the methods go through the stages described in the previous section. The tools which are provided to aid the developer in performing the actions in the various stages of the development cycle are typically the same tools for all development methods. The differences in the development methods are directly attributable to the differences in the nature of the tools that are used in a certain stage of the development rather than what tools are provided.

The approaches to the provision of tools by various development facility manufacturers are outlined in the following publications. Coutngs (1982) describes Intel's approach which is characterised by the provision of integrated hardware and software development support tools. Motorola's policy for provision of tools, as described by Kister (1982) is pitched at a higher level of support oriented towards developers who are using board-level modules. Lee (1982a) puts forwards Hewlett-Packard's strategy which is strongly oriented towards providing a capable user interface for the development tools whilst also optimising development throughput.

The following subsections summarise the tools that are typically provided to aid the developer at each of the development stages in the life-cycle.

2.4.1 Specification

Specification tools are not being used by microprocessor-based systems developers to any significant extent, although it is the author's opinion that in the future, as the complexity of these systems increases, the use of specification tools borrowed from other development facilities will undoubtedly become common.

2.4.2 Design

In the design phase, the solution is implemented by identifying the portions that have been previously designed and implemented in other previously developed systems. Effort is then concentrated on the portions which still have to be designed and on adapting the previously designed portions to the current implementation. Portions which are already designed usually perform commonly required functions and are also obtainable from outside sources. A good example of this is the use of a multi-tasking target executive, which allows a number of programs in the target to execute apparently concurrently. To use this executive in any design, it must be provided together with tools which will allow it to be configured.

2.4.3 Coding

The coding of programs is done in a language suitable for the application. This can either be a high level language or assembly language. The types of languages which are suitable for various applications are given in the next chapter.

2.4.4 Translation

To perform the translation of programs, a compiler for the language used in the coding or an assembler must be provided.

2.4.5 Integration

The various program modules are integrated either manually, or automatically by tools such as linkers. The capabilities of the linker will determine what structures can be used to support software modularity in the implementation of the system.

2.4.6 Validation

Validation in microprocessor-based systems involves the controlled and monitored execution of the developed programs. There are very few validation methods which do employ execution of the programs to be validated. Validation methods fall into two classes which

are distinguihed by the method used for the execution of the programs:

- Simulation: in which the instructions in the program under development are interpreted by a processor different to the target processor, with the aid of a simulation program. This implies that the validation tool can easily provide extensive control and monitoring capabilities because it executes in an artificially created and therefore more controllable and observable environment.

- Emulation: in which the instructions in the program under development are interpreted by the target processor and preferably in the target hardware. This implies that the execution environment of the program under development is virtually identical to the target environment. The execution is monitored and observed by the addition of software and hardware components to the target system or a target processor. These components are ideally included in a manner which makes their use virtually transparent to the program under execution.

Validation tools are thus either simulators or emulators. A more comprehensive description of validation methods (sometimes called debugging methods) and tools used in the process is given in the chapter which develops the upgrading policy.

2.5 Microprocessor-based systems development environments

The development environment is itself provided to support the use of the development tools at all of the stages in the life-cycle of the system. The emphasis of the evolution of development facilities in general revolves around the provision of a comprehensive development environment. Hewlett-Packard's Lee (1982) stresses that the development environment must provide effective facilities for performing the following functions:

1. The manipulation and storage of information

2. The support for modular programming

3. The coding of programs in languages appropiate to the application.

This trend is even more obvious in a description of development facilities provided by Texas Instruments as given by Bristow, Vinson and Woolen (1982). This description stresses the need for generality and flexibility which must inherent in the user interface, and the importance of support for developing applications which employ concurrent programming. The reason for this trend will be discussed after the requirements for development capabilities are outlined in the following chapter, when the expansion policy for the department's facilities is developed.

Microprocessor-based systems development environments

The development environment is provided by the components which support the use of the development tools i.e.

- the software components which interface with the user.

- the operating system and its utilities.

- the underlying hardware components which allow the execution of all the software components.

2.5.1 User interface

This is the most important component of the development environment because it is the only part with which the user comes directly into contact. The user interface allows the user to access the development facilities and thereby employ its capabilities. The user interface is therefore the most important factor affecting the productivity of the users. The design of a user interface is characterised by a three-way tradeoff between friendliness, functionality and simplicity of implementation. It is a general rule, that given a certain complexity of implementation, the more flexible and powerful the capabilities provided by a user interface, the more unfriendly it is.

Fuctions which are performed by the user interface include the interpretation of user commands, the guiding of the user in formulating these commands, the automatic execution of user definable sequences of commands and the provision of capabilities for developing user defined commands.

2.5.2 Host Operating System and Utilities

These components of the environment are the ones which lay the foundations for the functionality of the development facilities and therefore contribute to its usability and expandability.

The most important function performed by the Operating System is resource management and the most important resource in any system is secondary storage because this is where most of the information is kept. Therefore file management is the single most important function performed by the Operating system components. The capabilities provided by the file management system form the major touchstone for evaluating the capabilities of the host opearating system and its utilities. Functions which are performed by the file management system include file organisation and access control, file archival and backup, and file structure and content maintenance.

Other functions performed by the operating system include execution control, device handling, facility access control, resource sharing, device maintenance and the provision of high-level I/O operations.

2.5.3 Hardware components

As previously described, the hardware components of a development host form a typical user-programmable computer system with the added interfaces for connection to the target systems. The functionality of these components will determine important capabilities such as throughput, multi-user access and quantity of primary and secondary storage.

These components will allow the operating system, utilities and development tools to execute and therefore perform their function.

2.6 Conclusion

The majority of development facilities and the development methods that they support can be described in terms of a development process which is made up of a number of distinct stages. Development tools are provided to aid in performing the actions required by each of these stages. The development environment is provided to support the use of the development tools by allowing these to interact amongst themselves, and with the user.

CHAPTER 3

REQUIREMENTS FOR DEVELOPMENT FACILITIES

This chapter lays down the requirements for a development facility within the context of the Department of Electrical Engineering of the University of the Witwatersrand.

In order to make these requirements more understandable it is necessary to define a model for the development process. This will allow the requirements to be evaluated in separate classes before obtaining the overall requirements for the development facilities.

3.1 Development Model

A model for the development process can be conceived by first considering the basic components which interact within the development process. These are:

- the user.
- the development facility.
- the microprocessor-based target system which the user will employ to perform the required task.

Their interaction is as shown below:



The user is guided during the development by the feedback which he receives from the target, via the development facilities. By looking more closely at this feedback, one can introduce into the model the following conceptions:

- requirements of the project undertaken by the user.
- the solution which is attainable at any one time.
- the problems encountered by the user.

The latter can be seen as the difference between the requirements and the solution. This can be show as:



With this framework in mind the requirements can now be obtained by firstly examining those requirements which are determined outside the scope of the development facilities. These are:

- the user's project requirements.

- the characteristics of the users.

- the targets which they will employ and what forms their solutions will take.

With this information one can now consider the problems that the users will encounter and this will then allow one to conclude what the requirements for the development facilities are.

3.2 User's Project Requirements

These can best be reviewed by stating the classes of projects that users within the department generally undertake. These may be summarised as follows:

- Undergraduate course projects which typically entail taking general principles as covered in lectures and applying them to simple projects which will enforce the understanding of these principles. These projects are undertaken on a part-time basis during the term and they require the equivalent of one to two weeks of continuous work.

- Projects undertaken by final year students as their major project. The students acquire specific information about the field they are working in, and they apply this to a specific task so as to become proficient in the application of what they have learned. These projects can be of considerable complexity and they require the equivalent of six to nine weeks of continuous work.

- Postgraduate projects in fulfilment or partial fulfilment of higher degrees. The projects in this category range over a wide spectrum of complexity and applications. The manner in which these projects are tackled is probably more similar to the approach used in an industrial environment than the previously described projects. This is applicable even though some postgraduate projects tend to be less production-oriented than industrial projects. These projects are normally tackled on a full-time basis requiring four to twelve months of continuous work.

User's Project Requirements

It is evident that the users have widely different requirements in terms of the types of projects which they undertake. Most projects are characterised by having strict time deadlines which are satisfiable because the scope of the projects is limited by the supervisors.

3.3 Characteristics of users

In a way similar to the types of projects undertaken, users projects within the department fall into three categories:

- Undergraduate students - these students have no specific knowledge or experience that they can apply to the usage of the development facilities. They usually have only a knowledge of the principles behind its usage. They will normally be encountering the facilities for the first time and are looking for the shortest and simplest solution to their problems.

- Final year students - These students are usually no more advanced than the other undergraduate students, but when tackling their final year projects they will dedicate time to train themselves in the use of the facilities which their supervisors indicate should be used. They may have met the facilities beforehand and will normally make comprehensive use of them but will only employ them to solve the problems which they recognise.

- Postgraduate students - these people have widely differing previous experiences and will usually be able to dedicate enough time to learning the use and application of the available facilities to become experts. They will normally have encountered the facilities previously and will usually try to use them to the limits of their capabilities.

The variety of users who will require access to the facilities is too large to tailor the system to any single class. To accelerate the initial learning rate of the new users, friendliness and approachability are necessary requirements for the development facilities.

3.4 Target System

The characteristics of the target syster requirements will be examined under the headings of complexity and architecture.

3.4.1 Complexity

The complexities of the target systems can be separated into the following levels:

- Chip Level designs which are configured from the separate Integrated Circuit components for custom applications.

- Board Level designs which are configured lately from standard boards to implement a conventional system.

- Module Level designs which are composed of modules made up of standard boards. These are used to implement multiprocessor systems.

- System level designs which are built from standard computer systems and used to implement distibuted multicomputer systems.

It is evident that provision must be made for supporting the full range of complexity with facilities of the appropiate level. The facilities should support each level as completely and cost-effectively as possible.

3.4.2 Architecture

The processor architectures which should be supported will be classified as follows:

- Second generation or 8-bit microprocessors, specifically those to which the department decided to commit itself, being the Intel 8080 and 8085 and the Zilog Z80.

- Third generation or 16-bit processors, specifically the Intel 8086.

The choice of which processors to support was deliberately kept as small as possible so as to favour the quality rather than the variety of support. None the less, when considering the large range of complexities of target systems which must be supported, and also bearing in mind that room should be left open for easy future expansion of new architecture support, the requirement for flexibility and universality of the facilies is evident.

3.5 User's Project Solution

In this section an review is given into the form taken by the user's project solutions. They may be classified into those employing a high-level language for their implementation and those employing assembly language.

3.5.1 Implementations employing High-Level Languages

These implementations can be characterised by the level of abstraction and facilities provided by the high-level language, e.g. in complex systems a user may employ a high level language which, with the aid of an executive, will provide multi-tasking operation.

Another important characterisation of high-level language implementations is the programming orientation. Some systems are largely implemented using application oriented programming which does not require strong interaction between the program and the underlying hardware. Others, which tend to be the majority of microprocessor systems, need a strong interaction and are therefore programmed in systems programming languages and the target's assembly language. Other high level languages are usually unsuitable for use in the implementation of most microprocessor-based systems.

This means that compilers and their associated host and target support environments must be provided as part of the facilities. It is desirable to constrain the choice of language as little as possible, therefore as many compilers should be made available as possible.

3.5 User's Project Solution

In this section an review is given into the form taken by the user's project solutions. They may be classified into those employing a high-level language for their implementation and those employing assembly language.

3.5.1 Implementations employing High-Level Languages

These implementations can be characterised by the level of abstraction and facilities provided by the high-level language, e.g. in complex systems a user may employ a high level language which, with the aid of an executive, will provide multi-tasking operation.

Another important characterisation of high-level language implementations is the programming orientation. Some systems are largely implemented using application oriented programming which does not require strong interaction between the program and the underlying hardware. Others, which tend to be the majority of microprocessor systems, need a strong interaction and are therefore programmed in systems programming languages and the target's assembly language. Other high level languages are usually unsuitable for use in the implementation of most microprocessor-based systems.

This means that compilers and their associated host and target support environments must be provided as part of the facilities. It is desirable to constrain the choice of language as little as possible, therefore as many compilers should be made available as possible.

3.5.2 Implementations employing Assembly Language

Assembly Language will be employed for the implementation in certain special cases where a greater degree of implementation flexibility is required. These cases include:

- trivial implementations.

- educational exercises.

- implementations having critical time or space restrictions.

- implementations requiring special purpose structuring which cannot be supported by a High Level Language.

Assembly Language implementations will require powerful and mature Assemblers e.g. with macro processing and symbol scope control. A great strain is placed on the debugging tools used for such implementations as they will have to be powerful without being too difficult to use, and flexible to the extent of being user-tailorable e.g. multiprocessor symbolic simulators.

3.5.2 Implementations employing Assembly Language

Assembly Language will be employed for the implementation in certain special cases where a greater degree of implementation flexibility is required. These cases include:

- trivial implementations.

- educational exercises.

- implementations having critical time or space restrictions.

- implementations requiring special purpose structuring which cannot be supported by a High Level Language.

Assembly Language implementations will require powerful and mature Assemblers e.g. with macro processing and symbol scope control. A great strain is placed on the debugging tools used for such implementations as they will have to be powerful without being too difficult to use, and flexible to the extent of being user-tailorable e.g. multiprocessor symbolic simulators.

3.6 Problems encountered by the users

Now that the environment surrounding the development facilities has been defined, one can consider the problems which the users will encounter. The requirements for the capabilities of the development facility will be largely determined by the possible solutions to these problems.

3.6.1 Programming Inability

Typically undergraduate students have only had exposure to simple programming in FORTRAN. This means that they have considerable trouble in expressing themselves in low-level languages, and, when using assembly language they tend to make unwarranted assumptions. This implies that the availability of high level languages is a necessary requirement.

3.6.2 Lack of programming concepts

Users frequently find their problems insoluble because they lack the insight to either visualise their problem or the form of the solution. Users are usually unfamiliar with concepts of data and program structuring and information flow through their Software/Hardware complex. A development facility should provide capabilities for adopting these programming concepts in the development process in an obvious manner.

3.6.3 Hardware/Software interaction problems

Some users are still developing their understanding of the fundamental operation of their target systems. This means that they will have trouble with Hardware/Software interaction. A development facility should force an understanding of the fundamental operations before the user gets involved with more complicated problems.

3.6.4 Hardware design faults

Users are often confronted with emulators that do not behave predictably or with programs that are not executed logically. These are usually caused by marginal faults in the design of the target hardware. To rectify this, facilities such as logic analysers, should be available to debug hardware before it is used to execute and debug the software, even if an in-circuit emulator is used. Also, hardware debug programs for use with emulators (e.g. memory test programs and the like) should be available.

3.6.5 Complexity

Users are always struck by the complexity of an unknown facility, especially if they are educating themselves at the same time. It is obvious that the more capabilities a facility has, the more complex it will be, but the orderly and logical organisation of the capabilities will reduce the apparent complexity.

3.6.6 Unavailability or unsuitability of Tools

Postgraduate users very often can see the need for some very useful tools which can make the capabilities of the development facility more extensive. An ideal facility should provide all possible tools, but this is unrealistic as new tools are becoming available all the time. A more practical approach is for the development facility to provide a simple means for the users to add these tools at any time.

3.6.7 Misuse of Tools

Inexperienced users can often totally misinterpret the function of certain tools provided by a development facility, or even disregard the existence of tools. If a development facility has an obviously consistent structure with respect to the provision and fuctions of the tools it provides, users will naturally use tools correctly, and will come to expect tools which provide functions which are consistent with others within the total development framework.

3.6.8 Access to development facilities

Users will encounter problems when trying to access the development facilities as these are shared amongst a large number of different projects, and conflicts will occur. To minimise problems which arise from these access conflicts a development facility should cater for multiple users and should also provide maximal throughput for the users which are accessing simultaneously. A consequence of this is that they free the facility for use by others.

3.6.9 Time constraints

The major problem encountered by all users is the constrain placed on them in terms of the time they have to complete their work. As the scope of the projects is determined by the supervisors so as to make the deadlines realistic, the facility's throughput capabilities using various development methods must be known to the supervisors.

3.6.10 Overcommitment

This arises when the target system is targetted to an application which strains or exceeds its capabilities. It also arises when the development facilities are similarly employed. The limitations of the capabilities of the development facilities must be discernible to the supervisors as well as the users.

3.7 Requirements for the development facility

Up to now, many specific requirements have been stated. In this section, the general requirements can be summarised and the problem posed by this project can be stated as the provision of facilities to satisfy these requirements. The requirements may be listed as:

1. The development facilities must provide methods for multiple users to develop microprocessor-based systems concurrently. These methods must provide as much automation of the development process as possible so as to maximise throughput.

2. The capabilities of the development facilities must be as flexible as possible so as to support presently required, as well as future functionality. This should also be enhanced by the inclusion of capabilities for expanding the facilities.

3. The facilities must provide a comprehensive and friendly user interface.

4. The facilties must be implemented in as reliable a manner as possible. The department is understaffed, and has a large turnover of support staff and therefore cannot commit itself to extensive maintenance of the facilities. This applies especially to the Software components of the facilities.

5. The above requirements should be met with the maximum achievable cost effectiveness, as the department is financially constrained.

3.8 Conclusion

The requirements which are to be met by the development facilities which are to be provided arise because of the department's orientation towards educational project work. A broad and flexible base of support must be provided to meet the requirements.

These requirements will be used to formulate a policy for upgrading the department's development facilities in the following chapter.

CHAPTER 4

POLICY FOR THE UPGRADING PROCESS

The expansion and subsequent reconfiguration of the development facilities had to be guided by an integrated overall policy so that it could be effective. The application of this policy motivated all the actions which were performed during the course of the project.

The policy had to conform to current and future trends in the evolution of development facilities therefore the fundamentals of these trends had to be initially reviewed. Following this, the policy for the expansion and configuration of these facilities could be stated and motivated. The chapter is concluded by a consideration of the major implementation difficulties which can arise as a result of the adoption of this policy, and consideration is given to how these can be overcome.

4.1 Current and Futu e Trends

The current and future trends in the provision f microprocessor-bas d systems development facilities can be evaluated by first considering what the functionality increases in modern development systems are. When this has been done the configurations which support this functionality can be reviewed. A review of these trends is given by Tseng (1982b) and the following description is based on this material, together with an analysis of trends in modern offerings by development systems manufacturers.

4.1.1 Functionality

It has become evident to manufacturers of development systems that most of the general requirements as stated in the previous chapter can be met by providing a development facility which can be described as a "programmer's workbench". This is a programming environment which provides a large number of interrelated and integratable tools which the programmer employs to aid in the development of his software.

This approach was formalised and implemented by the designers of the UNIX operating system at Bell Laboratories and it is not surprising that the latest generation of development systems produced by Tektronix (1982) and Philips (1982) employ this operating system. It is similarly not surprising that new tools such as iMAKE which are being added to Intel development systems were derived from their UNIX equivalents. A programmers workbench is identical in concept to systems such as the Ada Programming Support Environment (APSE) described by Stenning et al (1981) and other similar software development support environments.

It is evident that any operating system which is tailored for software development provides the beginnings of such an environment. If such an operating system is mature and well developed it can provide more functionality as a development environment than

those on most current development systems. A typical example of this is any well established and well developed minicomputer operating system which supports software development (eg. Digital Equipment Corporation's RSX-11M, Data General Corporation's AOS and Hewlett Packard's RTE). The same observation can be made of timesharing operating systems available on many mainframe computer systems.

These trends also cater for many requirements which are not covered in the previous chapter because they are nor applicable to the department's needs at this point in time. These requirements include support for:

- Teamwork.

- Development of very large multimodule software complexes.

- Development of systems which require extensive. multidisciplinary interaction.

- Project management and documentation.

4.1.2 Configurations

All modern development facilities must provide the functionality previously described as well as a considerable increase in throughput compared to previous generations of development facilities. If the throughput is not increased users will not employ these new tools as their use will consume too much of the available time.

those on most current development systems. A typical example of this is any well established and well developed minicomputer operating system which supports software development (eg. Digital Equipment Corporation's RSX-11M, Data General Corporation's AOS and Hewlett Packard's RTE). The same observation can be made of timesharing operating systems available on many mainframe computer systems.

These trends also cater for many requirements which are not covered in the previous chapter because they are nor applicable to the department's needs at this point in time. These requirements include support for:

- Teamwork.

- Development of very large multimodule software complexes.

- Development of systems which require extensive. multidisciplinary interaction.

- Project management and documentation.

4.1.2 Configurations

All modern development facilities must provide the functionality previously described as well as a considerable increase in throughput compared to previous generations of development facilities. If the throughput is not increased users will not employ these new tools as their use will consume too much of the available time.

This increase in throughput, which is approximately of the order of 2 to 20 times that of the previous generation is accomplished by providing more powerful hardware components and more powerful operating systems to manage these hardware components. This is evidenced by the trends in current development systems to use powerful 16-bit CPUs, large main memories, and fast hard disks. These components are still not inexpensive and must be utilised fully to justify their cost. This explains the trend towards multi-user development systems which allows these costly resources to be dynamically shared.

Multi-user development systems fall into two categories: those that employ a single central processor and those that, at a small increase in cost provide a processor and memory for each user. The former configuration is more reminiscent of timesharing mainframe and minicomputer systems whilst the latter employs the more modern concepts of networking in order to share the expensive components which cannot be replicated for each user. An example of the former is the Tektronix (1982) 8560 MDL and of the latter the Hewlett-Packard (1982) 64000 system. The new Philips (1982) PM4422 is an example of a hybrid between the two configurations, although it is centrally controlled.

Generally speaking, the most extensive functionality is provided by configurations employing a central processor because these inherit their environments from mature minicomputer systems. On the other hand the development systems that employ networking are developed from scratch and have not yet attained the functionality of the others. A network-based development system promises the greatest potential in terms of throughput, but it will take considerable time before they develop the functionality and before all the components become cheap enough for this potential to be realised.

4.2 Expansion Policy

The proposed expansion policy will first be described, following which it will be reviewed in terms of the general requirements stated in the previous chapter, to see whether it precludes any of them.

4.2.1 Description

The priciple employed to implement this policy is that all the components of the development facilities will be employed to perform the functions which they can support best. Components of various types will be integrated into a complete development facilty so that they can be used together rather than separately.

The approach adopted is a very obvious one, best described and motivated for an academic environment by Glover and Bargainer (1981). The facility is centered around a single host minicomputer system which forms the connecting link between the various components and directly controls those facilities which must be dynamically shared. The facilities which are not dynamically shared, are connected to it and used independently, except that they require information which has been processed by the host.

Dynamically shareable facilities which the users will employ include editors, assemblers, compilers, linkers and simulators with their attendant file management and support utilities. Non-shareable facilities include emulators and logic analysers. This configuration will allow the bulk of users to perform development on the undedicated terminals of the host, and when required can use the specialised facilities without monopolising them unnecessarily.

Although the approach described above is relatively uncommon in industry, it is very extensively adopted in educational environments because it tends to be the only one which will give the re-

4.2 Expansion Policy

The proposed expansion policy will first be described, following which it will be reviewed in terms of the general requirements stated in the previous chapter, to see whether it precludes any of them.

4.2.1 Description

The priciple employed to implement this policy is that all the components of the development facilities will be employed to perform the functions which they can support best. Components of various types will be integrated into a complete development facilty so that they can be used together rather than separately.

The approach adopted is a very obvious one, best described and motivated for an academic environment by Glover and Bargainer (1981). The facility is centered around a single host minicomputer system which forms the connecting link between the various components and directly controls those facilities which must be dynamically shared. The facilities which are not dynamically shared, are connected to it and used independently, except that they require information which has been processed by the host.

Dynamically shareable facilities which the users will employ include editors, assemblers, compilers, linkers and simulators with their attendant file management and support utilities. Non-shareable facilities include emulators and logic analysers. This configuration will allow the bulk of users to perform development on the undedicated terminals of the host, and when required can use the specialised facilities without monopolising them unnecessarily.

Although the approach described above is relatively uncommon in industry, it is very extensively adopted in educational environments because it tends to be the only one which will give the re-

quired cost-effective throughput. Examples are given for university environments by Ahlgren (1981), Aylor and White (1981) and Zornig (1981), and a good description of their implementation, use and motivation is given by Alty (1982). Implementations are however, not only restricted to universities, since Saukkonen, Karppinen and Kemppainen (1982) describe a system which implements this policy in a very general manner in an industrially oriented environment. This implementation goes as far as having the host manage all accesses to the development systems connected to it. An example of a host-centered configuration as employed in an industrial educational environment is also given by Lumley (1981).

The policy proposed can be summarised by saying that it will implement a configuration which has centrally controlled sharing of those components which are easily shareable and which maintains communications with those that do not. The central controller will be based on a timesharing minicomputer system and it will host the development facilities. The facilities which will be shared are those that can be migrated to the host system. Others will remain in the attached development stations and can only be accessed in a local, non-shareable fashion.

4.2.2 Review

A system implemented using the described policy can provide a development environment for microprocessor-based systems. This is achieved by adding to these systems, the tools which are specific to microprocessor-based systems development such as assemblers. The other non-specific tools can be provided by the host operating system and its utilities. The host can be powerful enough to provide multi-user access with enough throughput. Functions which are not available on the host can can be provided by attached stations which will thereby extend the host's development environment.

The host development environment will co-ordinate the use of the development tools and can provide the required functionality, including the provision of a base for extending the capabilities of the development facility.

The user interface which will be inherited from the host minicomputer system can be comprehensive, but it may not be friendly enough. As the apparent friendliness of any computer system is dependant on whether the users have encountered it before, the lack of friendliness shown by the host may be tolerable if the users have used it previously, even for another different purpose.

Reliability of implementation will be met if the components employed are commercially supported and if local in-house software efforts will be limited to the development of interfaces between these components.

Cost-effectiveness is maintained because the major part of the investments which is the host system, has either been performed, or it may be employed for other uses rather than wasted on a specific application. Also the bulk of the components will be shareable.

In conclusion, the policy does not seem to preclude any of the requirements stated in the previous chapter.

4.3 Implementation difficulties

The major implementation difficulties center around the migration of components to perform various functions from the dedicated stations to the host, specifically functions such as debugging. If these functions are not migrated to the host, they cannot be dynamically shared amongst the users and therefore the policy does not provide sufficient cost-effectiveness. Cost-effectiveness can only be maintained if the functions performed by the dedicated stations are not in heavy and continuous demand. It is unrealistic to expect users not to be able to debug their programs whilst they are using the host directly, without access to the stations.

It is therefore appropriate to consider various debugging methods at this point, to see which ones can be migrated to the host system. If the debugging methods which can be migrated to the host can satisfy enough of the demand for debugging, the implementation of this policy is possible and desirable. Additionally, if the demand can also be met by employing a debugging method which calls for the replication of some inexpensive components, then the use of the policy is still justified.

4.3.1 Debugging methods

Debugging methods are varied, ranging from methods which require one to debug on the target system through to the use of simulators. Here are some examples of debugging methods:

1. In the target system with a monitor program.

2. In the target system with a state analyser with software debugging capabilities as shown by Dittman, Glasby and Benenati (1981).

3. In a development system which has all the hardware of the target as part of itself, as described by Harp (1981).

4. With an in-circuit emulator as described by Lejeune (1982) and Mihalik (1982).

5. With a remote emulator such as used in the MAGIC system reviewed by Finucci and MacLeod (1981).

6. Using a hardware/software simulator as described by Rodd and Gray (1979).

7. Employing a software simulator which simulates the target system.

The various methods are differentiated by different tradeoffs between how close the debugging environment is to the target and how much control the user has over the execution of his software. The different methods are chosen depending on which method can satisfy the particular debugging requirements.

Only the last three stated methods can be migrated to the host, but the hardware/software simulator requires in-house development as it cannot be commercially obtained.

The first method can be implemented in a cost effective manner even without migrating it to the hostby replicating a relatively inexpensive and standard target for each user.

4.3.2 Debugging Demands

The bulk of the demands for debugging can be met by the use of simulators and remote emulators, or at the worst, by many standard target systems.

4. With an in-circuit emulator as described by Lejeune (1982) and Mihalik (1982).

5. With a remote emulator such as used in the MAGIC system reviewed by Finucci and MacLeod (1981).

6. Using a hardware/software simulator as described by Rodd and Gray (1979).

7. Employing a software simulator which simulates the target system.

The various methods are differentiated by different tradeoffs between how close the debugging environment is to the target and how much control the user has over the execution of his software. The different methods are chosen depending on which method can satisfy the particular debugging requirements.

Only the last three stated methods can be migrated to the host, but the hardware/software simulator requires in-house development as it cannot be commercially obtained.

The first method can be implemented in a cost effective manner even without migrating it to the hostby replicating a relatively inexpensive and standard target for each user.

4.3.2 Debugging Demands

The bulk of the demands for debugging can be met by the use of simulators and remote emulators, or at the worst, by many standard target systems.

The simulators can cater for educational demands because these demands are relatively trivial in complexity, and the users are still becoming familiar with the target environment and they will find the simulated target environment more approachable because it is more predictable (if the simulation is sufficiently accurate). For complex applications, the simulator must be user-configurable, and in the hands of users which can predict, and cater for its limitations by employing the configurability: it can become a tool which can actually be more powerful than most in-circuit emulators. In a timesharing environment the simulator can be concurrently accessed by all the users.

A remote emulator can cater for debugging of systems at the board level of complexity, and will definitely be more cost effective for such systems than in-circuit emulators. It has the potential for being more cost effective than any other debugging facility when applied to levels more complex than board level system, if the functionality of remote emulators is extended so as to cater for the debugging of multi-tasking or even multi-processing systems. The remote emulator can be accessed by as many users as there are target systems connected to the host.

The rest of the demand which requires simple interaction interaction between the outside world and the target, can be met by a number of inexpensive monitor-equipped targets.

It is felt therefore that enough of the demand for debugging can be migrated to the central system or catered for by employing a number of standard targets for the policy to be viable.

4.4 Conclusion

This chapter has presented and motivated a policy for the provision of facilities for the development of microprocessor-based systems. This policy is simple yet generally applicable and it is independent of the current fashions in the field, thus being applicable in the future as well as the present.

CHAPTER 5
THE UPGRADING PROCESS - PHASE 1

The upgrading process was performed in two phases, one during 1981, and one during 1982. This was due to the fact that it is driven by finances which are available only as a lump sum at the beginning of the year.

The actions within the process will be documented in five chronologically distinct phases which cover:

1. Statement of requirements
2. Availability review
3. Acquisition of components
4. Configuration of components
5. Evaluation of capabilities

The final evaluation of capabilities would form the basis for the requirements for the upgrade process in the following year.

Before the requirements for phase 1 of the upgrading process can be stated it is necessary to document what facilities were initially available at the beginning of 1981 when the project started.

5.1 Initially available facilities

The initially available facilities will be described in two sections denoted by the manufacturers of the microprocessor-specific development facilities and a third section which outlines the general-purpose computing facilities within the department.

5.1.1 Intel facilities

The department's investment in Intel facilities consisted of one Intellec Series 800 Development System with ICE-80 in-circuit emulator. The hardware configuration comprised a full complement of memory, dual double density disk drives and dual single density disk drives. The ISIS-II operating system was employed, together with Intel standard utility software. This included PL/M-80 Compiler, ASM-80 Assembler and their supporting utilities. A PROMPT-80 unit was connected to the Intellec system for programming Intel 2708 and 8755 UV-EPROMS.

This basically constituted a single user development facility for the Intel 8080 processor, with software provisions for other 8-bit Intel processors.

5.1.2 Motorola facilities

The department owned a Motorola Exorciser I development system with a 6800 USE Module. This system, during the course of the this project had been dedicated to developing microprograms for a specialised application, therefore it was not considered as part of the generally available facilities.

It is possible that sometime in the future, this system may be integrated with the rest of the facilities, and at the very least, it is capable of supporting 6800-based systems development.

5.1.3 General-purpose computing facilities

The department has access to various general-purpose computers which can be employed to support various areas of microprocessor-based systems development. These fall into two categories which are described separately.

5.1.3.1 University Computer centre - This comprises a large mainframe system which can be used by many people. Most students are familiar with it and tools can be acquired for it which will enable it to aid in the process of development of microprocessor-based systems. Some of these were developed within the department by Jackson (1980) but these tend to be incomplete, unsupported and difficult to maintain.

This system is not an ideal microprocessor-based systems development host as it is overloaded, inflexible and it is difficult to obtain dedicated links to it, which is an important requirement if it has to communicate with the rest of the facilities.

5.1.3.2 In-house Minicomputer facilities - The department possesses two minicomputer systems which can be usefully employed to support microprocessor-based systems development. Both of these hosts can provide a general purpose software production environment and can be made to communicate efficiently with other components of the development facilities. The systems will be described separately in the following sections.

5.1.3.2.1 Data General Eclipse system - This constitutes an Eclipse S/140 computer system with 256KB of memory, a 25MB disk, printing facilities and 10 serial interface ports. It is controlled by the AOS operating system and supporting utilities which, as it stands can only contribute editing facilities towards the development of microprocessor-based systems.

5.1.3.2.2 Systime 5000 system - This machine is based on a Digital Equipment Corporation PDP-11/34 with 256KB of memory, two 4.8MB disks, printing facilities and 9 serial interface ports. It is managed by the RSX-11M operating system and its supporting utilities which can by itself only support the development of DEC LSI-11 microprocessors. At the start of the this project, it was only employed for control systems design and it was felt that this machine would be the first one to be used to host microprocessor-based systems development if this was required.

5.2 Statement of requirements

The requirements are derived so as to set specific goals for the capabilities which must be added to the development facilities as constrained by the finances available. The requirements would be derived by identifying the major areas of support which were not provided by the facilities at the time.

The available finances, from various sources were of the order of R25 000, and with this amount a number of capabilities were targetted for implementation during the year. The implementation would have to be completed before work on the final-year student projects was started in earnest (this would occur approximately on the 1 July 1981).

The various required capabilities will be described in the following sections.

5.2.1 Programming environment for Intel 8080/5 and Zilog Z80

The primary requirement was the establishment of a multi-user programming environment for supporting the development of systems based on the Intel 8080/5 and on the Zilog Z80 processors. The environment would be required primarily by the final-year students undertaking work on their final-year projects.

Due to the financial constraints, it was felt that the environment would be implemented primarily on the host computer, which would be the Systime 5000, by adding the following tools or software components:

1. Compilers and Assemblers for the 8080/5 and Z80.

2. Supporting Utilities for the above.

5.2 Statement of requirements

The requirements are derived so as to set specific goals for the capabilities which must be added to the development facilities as constrained by the finances available. The requirements would be derived by identifying the major areas of support which were not provided by the facilities at the time.

The available finances, from various sources were of the order of R25 000, and with this amount a number of capabilities were targetted for implementation during the year. The implementation would have to be completed before work on the final-year student projects was started in earnest (this would occur approximately on the 1 July 1981).

The various required capabilities will be described in the following sections.

5.2.1 Programming environment for Intel 8080/5 and Zilog Z80

The primary requirement was the establishment of a multi-user programming environment for supporting the development of systems based on the Intel 8080/5 and on the Zilog Z80 processors. The environment would be required primarily by the final-year students undertaking work on their final-year projects.

Due to the financial constraints, it was felt that the environment would be implemented primarily on the host computer, which would be the Systime 5000, by adding the following tools or software components:

1. Compilers and Assemblers for the 8080/5 and Z80.

2. Supporting Utilities for the above.

3. Simulators, or Utilities for downloading Object code to existing target systems for debugging.

These tools would be commercially obtained, or if they were unavailable and simple enough, would be developed in-house. It was decided that software components of high quality would be required, as the users would be educating themselves while they developed their software. This would mean that the software components would only be acquired after they had been evaluated on our system.

5.2.2 Chip-level debugging facilities for the Intel 8085

The Intel 8085 is the most widely used processor in the department and was being employed in at least two postgraduate projects which required chip level debugging. It was decided that this would be performed by an in-circuit emulator.

This implied that an 8085 emulation station had to be integrated into the facilities. It was decided that the existing Intellec Series 800 would not be converted into such a station as the 8080 emulation facilities would then be lost Therefore, a new independant emulation station would have to be acquired.

In order to spend as little money as possible, the emulation station would ideally only be capable of emulation, and software development would be performed elsewhere, preferably on the host. In such a case, utilities for downloading object code to the development station would have to be provided.

5.2.3 General-purpose device programming facilities

Initially, the department was only capable of supporting the programming of Intel 2708 and 8755 MOS EPROMs using the Intel PROMPT-80. The addition of expandable and truly general-purpose

programming facilities was considered to be an important requirement. It was felt that the initial implementation of these facilities would provide support for the programming of Intel 2716 and 2732 MOS EPROMs.

The requirement would be adequately met by acquiring a universal programming unit, which would be connected and controlled by the host computer, so that it could easily be shared by the majority of the users. If the device was also capable of programming TTL fusible link PROMs additional financing could be obtained for it, as a project which required the use of these PROMs for storing microprograms was in progress at the time.

5.2.4 Intel 8086-based development support

The remainder of the finances, if any, would be employed to add facilities for the development of Intel 8086 based systems, as projects using this processor were being considered for execution during the year. Most of these projects would employ board-level implementations.

programming facilities was considered to be an important requirement. It was felt that the initial implementation of these facilities would provide support for the programming of Intel 2716 and 2732 MOS EPROMs.

The requirement would be adequately met by acquiring a universal programming unit, which would be connected and controlled by the host computer, so that it could easily be shared by the majority of the users. If the device was also capable of programming TTL fusible link PROMs additional financing could be obtained for it, as a project which required the use of these PROMs for storing microprograms was in progress at the time.

5.2.4 Intel 8086-based development support

The remainder of the finances, if any, would be employed to add facilities for the development of Intel 8086 based systems, as projects using this processor were being considered for execution during the year. Most of these projects would employ board-level implementations.

5.3 Availability review

Prior to the acquisition of components, an availability review within the South-African context had to be performed. The results of this review were contained in an intermediate report which is reproduced in Appendix A. A list of development system component manufacturers considered is given in Appendix B.

This report detailed the available components which should be acquired to satisfy the requirements given above.

The American software house Boston Systems Office (BSO) was capable of supplying some of the required tools. These were CY8085 and CYZ80 assemblers and the SI8085 and SIZ80 simulators together with their supporting utilities. Their Pascal compilers were only available at exaggerated prices, possibly because these were still in development.

An in-circuit emulation station could be acquired from Tektronix (the TEK 8001) or Intel (Intellec Series II model 120). The Intellec was comparably priced, even though it included a floppy-disk drive.

Suitable universal PROM-Programmers could be acquired from Data I/O (System 17) or Kontron (MP80S).

Acquisition of the above components would exaust the finances available, but a requirement for the support of the Intel 8086 still needed to be met. Fortunately the department, as an educational institution would probably be able to acquire the MAGIC package from SPL International at no cost.

5.4 Acquisition of components

After reviewing what was available, a decision had to be reached which would determine which components would be acquired, and how the requirements would be met. The decisions taken will be documented in the following sections.

5.4.1 Programming environment for Intel 8080/5 and Zilog Z80

As the BSO components were the only ones available at the time, their quality and applicability to departimental use had to be determined. This was done by acquiring an evaluation copy of the tools for the Intel 8080/5, and trying them out.

All components conformed to the host operating system's command line and execution conventions, and users who were familiar with these would find them easy to use. The components were found to be relatively expensive, but their cost was justified because it was spread amongst the large number of users which could access them concurrently.

The assemblers were found to be fast and powerful. All the standard Intel instruction mnemonics were accepted by the assembler, but some of the assembly directives were different to their Intel equivalents. This was due to the fact that these assemblers implemented their directives in a consistent way which was independant of the processor supported, and in fact was reminiscent of the host assembler directives. It was felt that this was an advantage as the same powerful and manufacturer-independant directives would be provided for all processors supported.

The simulators exceeded all expectations in terms of quality and functionality. The commands were independant of the processor being simulated, and the symbolic debugging facilities provided not just user-defined symbols, but also full disassembly with the user symbols included, a very important feature still not found in

any debugging tools now available. The simulators are user-programmable in a language similar to BASIC, and this is used to add user-defined commands, change the presentation of information to the user and to transparently simulate components which are not supported directly by the simulator such as serial and parallel interface devices.

All supporting utilities were implemented in a processor independant manner and were found to be very functional and of consistently high quality, especially the very useful fully interactive linker. The utilities also included a symbol cross-reference generator, an object format converter supporting all known manufacturer's formats and a downloader for use with Tektronix systems such as the Tek 8001 and 8002.

The absence of high-level language compilers was unfortunate and could only be temporarily remedied by using the compilers provided by the existing Intellec station. Hopefully, it would be capable of meeting the expected demand, at least for the current year.

5.4.2 Chip-level debugging facilities for the Intel 8085

The decisions to be taken in this regard came down to making a choice between the Intel and the Tektronix stations. Both were comparably priced, and as the Tektronix station was of dated design and the Intel station also provided a disk, the Intellec was more attractive. It was therefore decided to acquire the Intellec station as this would also help in fulfilling the demand for compilation.

5.4.3 General-purpose device programming facilities

The DATA I/O System 17 programmer was very modern, and at the time, it was the first and only programmer to offer a universal programming module (the Unipak), albeit very costly. As both the Intellec station and the DATA I/O programmer was obtainable from

EBE and a special deal could be organised for its acquisition at a very favourable price, the DATA I/O programmer and the Unipak module became affordable and was therefore acquired.

5.4.4 Intel 8086-based development support

After discussions with representatives of SPL International UK and their local agents, the MAGIC development package for the Intel 8086 hosted under RSX-11M (the host operating system on the Systime 5000) was acquired.

This package is reviewed by Finucci and MacLeod (1981) and provides development facilities for systems starting from board-level complexity upwards. Program implementations using MAGIC are either in assembly language or in RTL/2 (Real Time Language/2; a standardised British language developed for process control) and if required, with the addition of a multi-tasking executive. Debugging is achieved by using a remote emulator. This package provides extensive development capabilities for projects requiring complex implementations.

5.5 Configuration of components

After the components were acquired and delivered, they had to be configured and integrated into the overall facility. Considerable familiarity has to be developed with the host so as to configure it to support microprocessor-based systems development concurrently with its other activities. This process will be documented by in the following sections.

5.5.1 Physical configuration of components

As the facility was centered around the Systime 5000 host system, other development system components had to be housed close to it. Appropiate rooms (which used to contain the old Illumination Laboratory) had to be cleared out and renovated.

The final layout of rooms included an airconditioned room which was normally closed off housing the host computer, a general-purpose terminal room with a printing terminal, three alphanumeric terminals and a graphics terminal with access by all users, and a microprocessor development room which housed the emulation stations and related components.

5.5.2 Configuration of emulation stations

These stations required the configuration of various hardware and software components.

5.5.2.1 Hardware components - Other than the installation and commissioning of the various emulation and memory modules in the stations, a decision had to be made as to the redistribution of the floppy-disk drives amongst the two stations. A dual single-density drive and a dual double-density drive could be placed in either of the stations independently. The Series II station also contained a single integral single-density drive.

In order to support PLM/80 compilation on both stations, at a dual drive had to be placed on each station. In order to have one station with both a single-density and a double-density drive, so as to be able to transfer files between the two, the double density drive was placed on the Series II station.

5.5.2.2 Software components - The configuration of software components was achieved by creating two system disks, one for each station, with the appropiate software. As the single-density drives did not have much capacity, two system disks had to be created for the 8080 support station, one containing the software components for emulation and editing, and one containing the software components for compilation.

The Intel screen-based editor CREDIT had to be reconfigured on the 8080 emualtion station for use with a ADDS Consul 580 terminal. Standard conventions were developed for disk label colours and disk names so as to simplify the management of disks and backups, as the disk density, owner and other characteristics were now easily identifiable. These are given in Appendix C.

The final aspect of software configuration involved developing a method for code to be downloaded from the host. With the aid of a program obtained from Insite (the Intel user group library) the Intellec Series II station could immediately be used as a terminal on the host, and with the aid of a few command files on the host, text could be downloaded reliably, with no handshake implemented, at 1200 baud. This implementation was considered sufficient, considering the small size of ASCII-HEX coded object files which were produced on the host. The BSO utility could produce standard Intel formatted object, including the symbols required for symbolic debugging by the Intel ICE-85b emulation control software. The Series 800 station was not connected to the host as it did not have an RS232 compatible interface, and considering that the demand for it to be connected was low, because it was used primarily

for compilation, no time was expended to connect it to the host. Files could still be moved to it from the host by using the other station and taking the disk over.

5.5.3 Configuration of the 8080/5 and Z80 programming environment

This was relatively trivial and no problems were encountered except that at first, no more than one user could access each utility at one time. This was solved by changing the method of naming and invocation of these utilities, so as to be in line with the other standard RSX-11M utilities.

At about this stage it was becoming evident that some kind of reorganisation of the files on the two disks of the host computer was required. The new organisation should allow easy backup, and had to conserve as much space as possible, as 2x4.8MB did not give much room to move in. Of the two disk units, one was fixed and the other was a removable cartridge, therefore the user files, which had to be backed up frequently, had to be constrained to the fixed cartridge. System software would be kept on the removable cartridge, and it had to be thoroughly examined and re-organised for it to fit into 4.8MB.

The disk space limitations, especially for the user files were beginning to constrain the usability of the system, and a very strict discipline had to be imposed on the users so as to keep the system usable.

5.5.4 Configuration of device programming facilities

The DATA I/O System 17 programmer and Unipak programming module were commissioned, and as it had a flexible RS232 compatible interface, it was connected at various times either to the host, or directly to the Intellec Series II development station. Although it was intended for computer remote control, it was found to be usable even when directly controlled by the user. This was the mode in which it was used, mainly because it was capable of satisfying the demand.

5.5.5 Configuration of Intel 8086 development environment

This basically entailed the installation and commissioning of the MAGIC package. This proved to be a non-trivial task due to the following problems:

1. The package occupied a large amount of online disk space if used in its released form.

2. Our installation was the first one to be performed in the country, and therefore the local agents had no previous experience either with this package, or software of similar nature.

3. The automatic delivery procedure provided with the software was found to be making invalid assumptions about the directory structures, and other such installation dependant details, and needed complete reworking.

4. Problems were experienced with establishing communication with the target single-board computer as there were some hardware configuration errors of the computer.

5. Errors existed in the interrupt handlers of the provided target multitasking executive. These were identified and solved by a member of the department's staff who had had previous experience with these executives.

The installation and commissioning of this package proved to be a very instructive task as much insight was gained into the workings of the host operating system and its application to the development of microprocessor systems.

5.6 Evaluation of capabilities

It was very important to evaluate the capabilities of the development facilities in the light of the upgrading that had occurred during the year, so that the requirements for the upgrading cycle during the next year could be formulated so as to provide the obviously missing capabilities.

The capabilities will be evaluated in separate sections which cover to what extent the requirements laid down at the beginning of the process were met.

5.6.1 Programming environment for 8080/5 and Z80

The capabilities of this environment were probed by two final-year projects, one postgraduate project and were considered by a further postgraduate project.

The final-vear students, found the facilities approachable and functional. Their main problems were due to the disk space limitations and they felt that they were not altogether happy with the results as they had not seen their programs running in 'real' hardware. They found their experiences with the use of an simulator as a debugging tool very valuable, and their unhappiness could have been cured by actually moving their code into a target system, which unfortunately did not exist.

The Z80 support facilities were extensively employed by a postgraduate group developing software for emulating a commercially available VDU on different hardware. This group uncovered the only outstanding bug found in any of the BSO utilities to date. This bug was trivial and easily recognisable and affected the interpretation of one of the instruction operands.

A postgraduate project considered the BSO 8085 simulator for debugging some specially structured assembly language coded programs running in coupled microprocessor systems. The simulator was attractive because it could provide automated program and data flow monitoring, and with some simple extensions could even simulate the interaction of the multiple processors.

BSO were approached so as to implement these extensions, but have been found more concerned with other problems. Never the less, the simulator will still be used in the debugging of this system, as it is the only tool which can provide the required execution monitoring facilities.

The major limitation (aside from the disk space problems) was the unavailability of high-level language compilers on the host, even though the Intellec stations managed to meet the demand for the year. Some users found the RSX-11 command language too terse, although when it became understood many users appreciated its flexibility (similar comments have been made in connection with the UNIX command language). The majority of users had not encountered the host system before and therefore some users found the task of learning the command language at the same time as learning the use of the development tools excessively difficult.

5.6.2 Chip-level debugging facilities for 8085

The Series II emulation station was extensively employed by two postgraduate projects to debug hardware that had been designed at the chip-level. In both cases it proved adequate for the task, and the method employed for the downloading of object code was adequate for this purpose. To reduce disk space limitations on the host, a number of postgraduate and staff users employed floppy disks created on the station for intermediate offline storage of their files from the host. This practice required more development of the download software as the method used was too slow and difficult to employ by new users.

A problem which manifested itself was the lack of space around the emulation stations which made it difficult for targets under development to be located near them. This problem had to be resolved in the next year, when the location of all the computer facilities in the department would be reorganised.

5.6.3 General-purpose device programming facilities

The programmer employed was found to be very capable and flexible, but it tended to be relatively difficult to use because it was intended for computer remote control and it was not used in this mode.

5.6.4 Intel 8086-based development support

The MAGIC package was employed by one final-year project and apart from some initial teething problems, which were due to the complexity of the package, it was found to be usable. It was concluded that it would be applicable to projects of large complexity due to its proven approach to the development of complex software. This approach was inherited from the minicomputer world.

CHAPTER 6

THE UPGRADING PROCESS - PHASE 2

The actions within the upgrading process will be documented in a manner similar to that used in the previous chapter.

The final evaluation of capabilities will form the basis for the conclusions reached in the next chapter.

6.1 Statement of requirements

The requirements were reached by considering the results of the evaluation of capabilities performed at the end of phase 1 of the upgrading process. After this the major areas of support which were not provided at the time could be identified, and requirements could be derived which would enable these areas to be covered.

The required capabilities would have to be implemented before work on the final-year projects was started in earnest.

The required capabilities will be described in the following sections:

6.1.1 Host storage capacity increase

The secondary (online disk) and tertiary (archival and backup) storage capacity of the host computer would definatively have to increase so as to remove many of the limitations felt by users in the previous year.

6.1.2 High level language support for Intel 8085

The majority of projects employed Intel 8085-based targets and could use a high-level language implementation so as to make the development easier for the majority of users.

This wou'd require the multi-access availability of suitable compilers so that a larger number of users could be catered for.

6.1.3 Extension of chip-level design support

Support for chip-level development for various microprocessors had to be extended. More facilities had to be provided for the Intel 8085 processor and the capability for supporting Intel 8086 and Zilog Z80 processors had to added.

6.1.4 Increase in user friendliness

All of the development facilities in the department had user interfaces which were not very friendly, although they met or exceeded the required functionality. It was felt that if friendliness could be improved the efficiency of the development facilities would benefit as the bulk of the user's time was spent in learning how to use the development facilities. The majority of undergraduate users did not encounter the facilities in any way before starting their projects, and this is what made the unfriendliness of the facilities intolerable.

6.2 Availability review

At the beginning of the year it became evident that a new multi-user development facility was going to be donated to the department. This development facility would attempt to satisfy some of the outstanding requirements.

In order to remain in line with the upgrading policy and so as to obtain the maximum benefit from this new facility it would have to be integrated with the other development facilities in the department.

The first requirement would be best met by acquiring a sealed media Winchester technology disk drive, and backups and archival would be performed by a 1/2-inch 9-track magnetic tape drive which had been donated to the department. The capacity of the disc drive required was estimated to be in the range of 50 to 100MB. Drives in this capacity range would have a greater degree of performance than the existing drive and therefore the throughput of the host system would be significantly increased, so that access could be provided to more users concurrently.

The availability of suitable disk drives and compatible controllers for the disk and tape drives had to be reviewed. The major constraint in the acquisition of these components would be that they could be covered by a maintenance contract, as the department did not have internal expertise which could be committed to this purpose.

6.3 Acquisition of components

The host computer system was upgraded by acquiring the following components:

1. Control Data Corporation 9730 Mini Module Drive (CDC 9730-160 MMD). This is a low cost sealed media Winchester technology disk drive with an unformatted capacity of 165MB. This drive was chosen in preference to a 80MB unit because the difference in cost was minimal.

2. Dataram S33/A1 disk drive controller. This controller is compatible with the above disk drive and would format it to give 2X67MB logical disk units. The controller emulates the Digital Equipment Corporation RM02/3 disk subsystem so that none of the software in the host computer needs to be altered.

3. Dataram T36 magnetic tape drive controller. This controller is compatible with the donated Wangco Model 10-45 PE tape drive and emulates a Digital Equipment Corporation TU10 tape subsystem with a density of 1600 bpi. and a speed of 45 ips.

4. Digital Equipment Corporation TU58 cartridge tape drive. This tape drive is a low cost device could be used to ease the transportation of software to or from other computers.

5. Digital Equipment Corporation DL-11W serial interface. This interface could be used to connect either the TU58 tape drive to the host or when it was not in use, any other device with an RS-232 compatible asynchronous serial interface.

6. Digital Equipment Corporation BA-11 expansion chassis. This expansion unit could accomodate the interfaces and controllers of the above drives and it included a power supply and cables to connect it to the host computer.

The Hewlett-Packard 64000 development system which was donated to the department included the following components:

1. Three HP64100 development stations. Each station contains a computer system and a VDU and keyboard to communicate with the users. One of the stations has an integral cartridge tape drive used for software transfer.

2. One HP7911P disk drive. This disk drive has a capacity of 27MB and includes a built in cartridge tape drive for backup purposes. It is shared amongst the three development stations.

3. One HP2631B printer. This device is shared amongst the three development stations.

4. The following components were also included. These could be distributed amongst the stations as required.

    1. One device programmer supporting Intel 2716 and 2732 family Eproms.

    2. Two Intel 8085 in-circuit emulators. The emulators and the following related components may be distributed amongst the development stations as required.

    3. One Intel 8086 in-circuit emulator.

    4. One Zilog Z80 in-circuit emulator.

6. Digital Equipment Corporation BA-11 expansion chassis. This expansion unit could accomodate the interfaces and controllers of the above drives and it included a power supply and cables to connect it to the host computer.

The Hewlett-Packard 64000 development system which was donated to the department included the following components:

1. Three HP64100 development stations. Each station contains a computer system and a VDU and keyboard to communicate with the users. One of the stations has an integral cartridge tape drive used for software transfer.

2. One HP7911P disk drive. This disk drive has a capacity of 27MB and includes a built in cartridge tape drive for backup purposes. It is shared amongst the three development stations.

3. One HP2631B printer. This device is shared amongst the three development stations.

4. The following components were also included. These could be distributed amongst the stations as required.

    1. One device programmer supporting Intel 2716 and 2732 family Eproms.

    2. Two Intel 8085 in-circuit emulators. The emulators and the following related components may be distributed amongst the development stations as required.

    3. One Intel 8086 in-circuit emulator.

    4. One Zilog Z80 in-circuit emulator.

5. Three emulation memory controllers and three emulation memory boards.

6. One internal emulation analyser.

5. An operating system and related utilities together with tools to use all the above components and allow programming of the various processors both in assembly language and in Pascal/64000.

6.4 Configuration of components

The acquired components had to be configured in such a way as to provide the required capabilities. This process is documented in four sections - the physical configuration of all the computing facilities, the configuration of the host system, the configuration of the HP64000 development system and the integration of the two systems.

6.4.1 Physical configuration

A complete reorganisation and relocation of all the computing facilities in the department was performed at the beginning of the year.

This was motivated because of the following reasons:

1. The space available near the emulation stations was insufficient to allow a number of targets to have access to them.

2. The new HP64000 system had to be integrated with the other development facilities and therefore had to be placed in the proximity of the host computer system. No room for expansion existed near the laboratory where the host system was currently hosted.

3. The increase in the number and usage of all the computing facilities in the department required that access had to be made easier by a larger number of students.

4. The new nature of the computing facilities required that the facilities had to be organised into a form which was similar to a computing centre, with low benches for the correct placement of VDUs. The existing organisation of the computing facilities was similar to that used in laboratories for electrical and electronic experimentation, with high benches and stools. This organisation was not

suited to computing work and also resulted in a large amount of floor area being wasted.

The new organisation of computer systems in the department was planned and executed using the following guidelines.

1. The central computer laboratory was reserved for equipment which had to have access only by users doing specific project work using some dedicated facilities.

2. The areas surrounding the central computer laboratory were employed for the placement of all the undedicated computer terminals (such as VDUs and printers), used for general-purpose access by users.

3. All other laboratories could house computer equipment which could be kept relatively independant of the central facilities.

The reorganisation resulted in the relocation of the host computer system in the central computer laboratory together with its development stations and the HP64000 development system. The general-purpose host terminals were placed in an area outside this laboratory, adjacent to the host. Work benches were modified to be of the correct height and were organised so that benches with an upper shelf were placed where targets would be connected to emulators, so that the upper shelf was used for the placement of the required instruments.

6.4.2 Development host system

The first stage in the configuration of the host system entailed the installation and commissioning of the components, following which, the operating system software had to be regenerated to make use of the new configuration.

The mechanical installation of the components was performed by the department and the electrical installation was performed by the sellers of the equipment. The cartridge tape drive was bought in kit form and had to be assembled, packaged, installed and commissioned within the department.

The operating system software was regenerated and reconfigured to suit the needs of the department. As the disc capacity was in excess of the requirement, only one logical unit was dedicated to online usage and the other was reserved for regeneration of the operating system.

A special technique for configuring the operating system software was employed which allowed the new serial interface to be used for the cartridge tape drive, for a target used as a remote emulator under the MAGIC system and for the universal device programmer. This allowed the number of terminals connected to the host to be increased.

6.4.3 HP64000 Development system

The acquired components were installed and were distributed amongst the stations so as to provide one Z80 emulation station and two 8085 emulation stations. This configuration would meet the need of the final-year users. The 8086 emulator was not immediately installed, and the internal analyser which it required was used in the other emulators as the need arose.

Two of the three stations were configured for interfacing to the host system and the other was configured for interfacing to a target system which was being developed by a postgraduate student.

6.4.4 Integration of facilities

The HP64000 development system had to be integrated with the rest of the facilities so that the development load could effectively be shared across the whole set of facilities.

The HP64000 stations which were acquired are not user-programmable and therefore one has to rely on the software present in them to allow them to communicate with other computer systems. The stations had a well designed 'terminal' facility which allowed the use of the station as a terminal of another computer as well as file transfer (both text files and ASCII-formatted binary files).

For the integration to be performed, a program which implemented the correct handshake protocols had to be written for the host computer. This task was tackled in two stages.

The first stage was undertaken before the HP64000 development system was delivered. In this stage the expertise of programming in a suitable language on the host was developed. RTL/2 was employed as a high level-language because it allowed flexible I/O through a serial interface to be performed. A program which allowed file transfers between Hewlett-Packard 2640 series terminal devices and the host computer at high speed with handshake was developed. This program which was called HPT, also implemented standard RSX-11 command line processing.

When the HP64000 development stations arrived, a new program for transferring files between them and the host was developed. This program was called XFR and a listing of it is given in appendix D.

Another aspect of the integration of the development facilities which was undertaken consisted in the transportation of the Intel

6.4.4 Integration of facilities

The HP64000 development system had to be integrated with the rest of the facilities so that the development load could effectively be shared across the whole set of facilities.

The HP64000 stations which were acquired are not user-programmable and therefore one has to rely on the software present in them to allow them to communicate with other computer systems. The stations had a well designed 'terminal' facility which allowed the use of the station as a terminal of another computer as well as file transfer (both text files and ASCII-formatted binary files).

For the integration to be performed, a program which implemented the correct handshake protocols had to be written for the host computer. This task was tackled in two stages.

The first stage was undertaken before the HP64000 development system was delivered. In this stage the expertise of programming in a suitable language on the host was developed. RTL/2 was employed as a high level-language because it allowed flexible I/O through a serial interface to be performed. A program which allowed file transfers between Hewlett-Packard 2640 series terminal devices and the host computer at high speed with handshake was developed. This program which was called HPT, also implemented standard RSX-11 command line processing.

When the HP64000 development stations arrived, a new program for transferring files between them and the host was developed. This program was called XFR and a listing of it is given in appendix D.

Another aspect of the integration of the development facilities which was undertaken consisted in the transportation of the Intel

Floating Point Arithmetic Library (FPAL) to the host system, and thereafter to the HP64000 system. This was only achievable if the source for FPAL was obtained. Intel do not provide the source, therefore all the separate modules of the library had to be disassembled and ported to the non-Intel systems. Thereafter it would be reassembled and reconverted into a relocatable object library.

6.5 Evaluation of capabilities

The capabilities resulting from the upgrading process are evaluated in the following sections.

6.5.1 Host storage capacity increase

The storage capacity and attendant capabilities as well as the throughput of the host computer system were effectively increased and met all the needs encountered during the year.

All components proved to be reliable and useful except that an intermittent fault was detected when the disk and tape drives are doing concurrent transfers. This fault has not been rectified yet.

6.5.2 High level language support for Intel 8085

The Pascal/64000 compilers provided with the HP64000 development system allowed a number of final year student groups to undertake the programming of their systems in a high-level language. The implementation of Pascal was found to be powerful and flexible but it had some drawbacks when compared to other implementations. The major drawbacks were the inability to declare string literals and the lack of real arithmetic, which made many of the programs written very incongruous.

The demand for access to these compilers was effectively met by the number of stations on the development system and by its throughput.

6.5.3 Extension of chip-level design support

This support was utilised by two final-year projects using Intel 8085 processors and one final-year project using the Zilog Z80 processor. The latter group were very successful in their use of the emulation capabilities. The Intel 8085 emulator users found many problems in the use of the emulator because of its erratic

behaviour. In one case, this was due to hardware faults in the design of the target system, and the emulator was in no position to be able to give indication of this fact.

The other erratic behaviour was attributable to two causes. The first was damage caused to the emulator pod buffers by voltage transients which occurred because the supply earths of the work benches had been accidentally disconnected. The second was a design fault on the part of Hewlett-Packard in the mechanical aspects of the connections at the tip of the emulation pod.

A postgraduate student completed his debugging successfully using the Intel 8085 emulator. Another student attempted to use this emulator for debugging a system with memory-mapped I/O devices and found it unusable because isolated memory locations could not be read from by the emulator under user control.

The debugging feature set of the emulators was found to be very extensive but surprisingly, few of the users employed anything like its full capabilities.

No demand for the 8086 emulator arose during the year.

6.5.4 Increase in user friendlyness

The bulk of microprocessor-based systems development by new users was performed on the HP64000 system and the key-directed command syntax provided by the system was greatly appreciated by the new users. This capability was so heavily relied on that few of the users consulted the manuals and they sometimes felt that these were unnecessary. As a result, many of the users wasted time with problems that need not have occurred if they had not been lulled into a sense of false security regarding their familiarity with the development system.

The documentation provided with the HP64000 system is of high standard, but its content does not cater properly for users who are unfamiliar with the principles of microprocessor-based systems development.

The problems encountered with the HP64000 system are largely attributable to the fact that it is still a relatively new and therefore poorly understood, immature and underdeveloped system because it has not been in use for a long enough period of time, both within the department and throughout the world. This means that it is still relatively inflexible in its approach to microprocessor-based systems development and the overall quality and operational consistencies of its capability set has not yet evolved. It does not effectively provide a development workbench as described in the chapter which outlines the upgrading policy.

Also, due to its novel architecture and recent introduction, it is not a very cost-effective solution to development problems, because the components which are duplicated for each user are very expensive and its large and ongoing development costs incurred by Hewlett-Packard have not yet been spread over enough users. It still appears to be one of the most capable development systems available at the present time.

The documentation provided with the HP64000 system is of high standard, but its content does not cater properly for users who are unfamiliar with the principles of microprocessor-based systems development.

The problems encountered with the HP64000 system are largely attributable to the fact that it is still a relatively new and therefore poorly understood, immature and underdeveloped system because it has not been in use for a long enough period of time, both within the department and throughout the world. This means that it is still relatively inflexible in its approach to microprocessor-based systems development and the overall quality and operational consistencies of its capability set has not yet evolved. It does not effectively provide a development workbench as described in the chapter which outlines the upgrading policy.

Also, due to its novel architecture and recent introduction, it is not a very cost-effective solution to development problems, because the components which are duplicated for each user are very expensive and its large and ongoing development costs incurred by Hewlett-Packard have not yet been spread over enough users. It still appears to be one of the most capable development systems available at the present time.

CHAPTER 7

EVALUATION, RECOMMENDATIONS AND CONCLUSIONS

At the end of the upgrading process it is necessary to evaluate the results achieved in terms of the previously stated requirements. From this evaluation recommendations for the future can be made and conclusions can be drawn.

7.1 Evaluation of results

The results achived will be evaluated by considering how well they meet the stated requirements. The organisation of this section follows that of Chapter 3.

7.1.1 User requirements

The facilities were not used for the support of undergraduate course projects, but only for final year and postgraduate projects. This is due to the following reasons:

- The throughput of the development host will not support more than 4 simultaneous users without untolerable response times.

- The proficiency of the students in the use of the facilities was limited.

- The level of the material presented to the students did not warrant developing this proficiency.

As foreseen, only the postgraduate students made use of the more sophisticated facilities. The majority of final year students used the HP64000 system because of its greater user friendlyness.

7.1.2 Target system requirements

The facilities provided a broad base of support which catered adequately for all required complexities of target systems.

All there required processor architectures were adequately supported, except for the unexpected need for support of the Motorola 68000.

7.1.3 Project Solution requirements

Most implementation methods were adequately supported, except that high-level language support was not available for the 8-bit processors on the development host.

7.1.4 Problems encountered by users

The development facilities provided a base of support which has the potential to resolve the majority of the problems encountered by the users. This potential is limited primarily by the discipline imposed by the development methods used and their application to the problem solution, rather than the inherent limitations of the development facilities.

7.1.5 General requirements

The development facilities provided in the course of the upgrading process completely overshadowed those that had been previously available both in terms of the number of users that could be catered for, and the development aids provided.

The generality and flexibility of the facilities is such that they can cater for present as well as future needs if the flexibility is correctly employed.

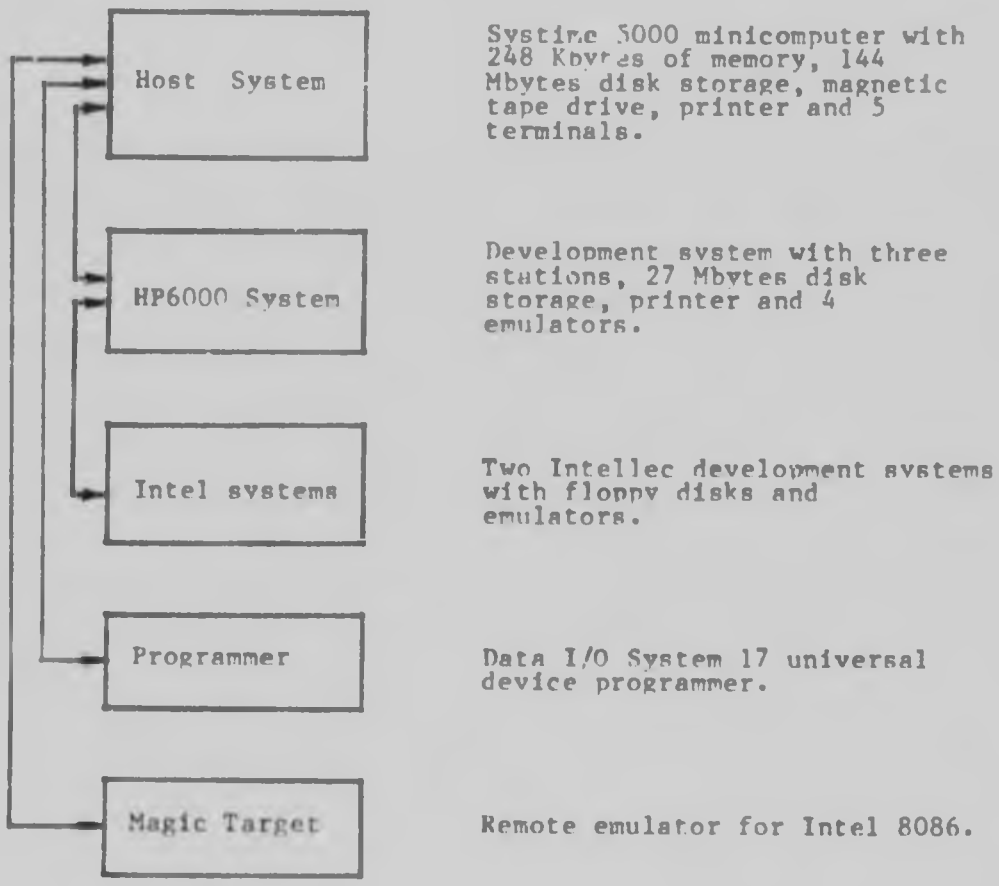
Both users and supervisors found the facilities approachable as they have been well received and accepted, especially if the extent to which the facilities were utilised is considered. The development host is not very user friendly although its facilities are approachable after an initial lerning period.

The facilities were implemented in a reliable manner without excessive reliance on departmental support staff.

Evaluation of results

The development facilities have the potential for supporting large numbers of users in an extremely cost effective fashion. This is not a characteristic of many commonly available development systems.

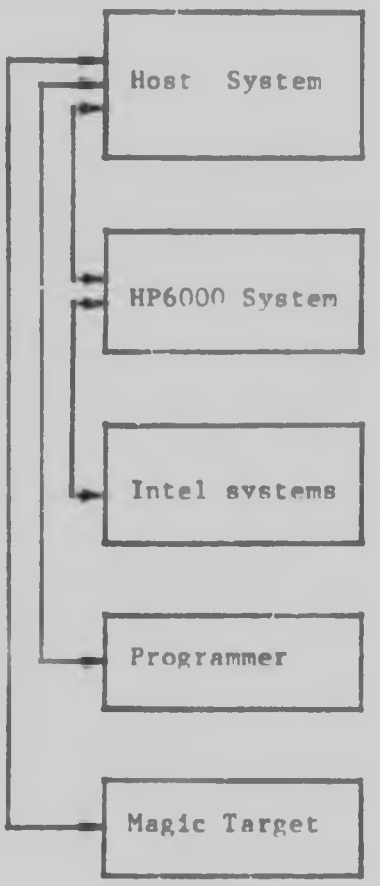
The final configuration of the development facilities is as given in the following figure:



Evaluation of results

The development facilities have the potential for supporting large numbers of users in an extremely cost effective fashion. This is not a characteristic of many commonly available development systems.

The final configuration of the development facilities is as given in the following figure:



Systime 5000 minicomputer with 248 Kbytes of memory, 144 Mbytes disk storage, magnetic tape drive, printer and 5 terminals.

Development system with three stations, 27 Mbytes disk storage, printer and 4 emulators.

Two Intellec development systems with floppy disks and emulators.

Data I/O System 17 universal device programmer.

Remote emulator for Intel 8086.

7.2 Recommendations for future upgrading

It is recommended that future upgrading should follow the principles outlined in this project report as they have been proved sound. In order to meet the departments requirements more effectively the following specific recommendations are made.

7.2.1 Use of facilities

More extensive use can be made of the development facilities so as to further realise their potential, especially for undergraduate course project work and laboratory sessions.

To do this the throughput of the development host may have to be increased. The friendliness of the development host must be increased to support these less experienced users. This can be achieved by installing a newer version of the RSX-11M operating system which includes a friend'ier command line interpreter.

For laboratory work, students can use the assemblers and simulators, and at a later stage, can control real hardware by interfacing target systems to the host. This can be done by connecting the department's newly acquired Mostek MDX Proto Kits to the host.

7.2.2 Integration of facilities

There is considerable scope for further integration of the facilities and unification of the development environment, especially with respect to the HP64000 system. It is feasible to perform development for the supported 16-bit processors on both the development host and the HP64000, eg. by writing programs in RTL/2 on the host and downloading object code and symbol tables to the emulator for debugging. Downloading does not present any problems, but appropriate format conversions must be performed. There are no difficulties foreseen in this regard.

7.2.3 Upgrading of development methods

There is a wide variety of development methods employed within the department and a number of useful procedures should be identified and documented to aid the users.

Many development packages are becoming available which can provide newer development methods but these tend to be hostable only on more modern software and hardware systems, such as the UNIX operating system and the Digital Equipment Corporation VAX-11 computer.

It is in the department's interest to expand in the direction of these systems especially when considering UNIX's increasing popularity and the VAX-11's compatibilty with the current host. To resolve conflicts between requirements and to make implementation of the newer development facilities possible it will probably become necessary to organise the facilities around a number of different development hosts. The HP64000 is already performing some of the functions of a host, and others can be added.

7.3 Conclusions

The wider application of microprocessor-based systems is primarily costrained by their development costs. A rational and comprehensive effort has been made in this project to identify the development facilities required by the department and to implement them. The knowledge gained thereby should aid in the understanding of the development process as applicable to microprocessor-based systems so that the development costs may be more effectively contained.

The formulated upgrading policy and the decisions taken during the upgrading process have made use of established principles rather than purely following current trends. This has resulted in the provision of facilities which are uncommonly organised around a central development host, but which are very capable.

This report stresses the application of established development principles to microprocessor-based systems without the aimless implementation of unusable development tools. The succesfull development strategies will always be based on previous experience in the field and the microprocessor-based system can and must rely on the experiences gained in mainframe and minicomputer systems. It is gratifying to see that many well-known development facility suppliers are starting to provide facilities which are based on well established principles and implemented with mature software and hardware components, eg. the Tektronix 8500 MDL series.

The Department of Electrical Engineering at the University of the Witwatersrand has acquired an extensive set of widely applicable development facilities.

REFERENCES

Ahlgren, David J. (1981). A Microcomputer Laboratory Work Station. IEEE Trans. on Education. VOL. E-24 NO. 1. February 1981. p. 59-62.

Alty, Jim. (1982). Development without Development Systems. Microprocessor Development and Development Systems. Ed. by Vincent Tseng. Great Britain: Granada, 1982. p. 171-193.

Aylor, James H. and White, Edward J. (1981). Creating a Microcomputer Facility. IEEE Trans. on Education. VOL. E-24 NO. 1. February 1981. p. 47-50.

Bristow, Geoff. Vinson, Peter. and Woolen, Dave. (1982). Software Facilities II. Microprocessor Development and Development Systems. Ed. by Vincent Tseng. Great Britain: Granada, 1982. p. 102-121.

Coninas, W. E. S. (1982). The Approaches to the Provision of Tools I. Microprocessor Development and Development Systems. Ed. by Vincent Tseng. Great Britain: Granada, 1982. p. 43-55.

Dittman, Paul. Glasby, Dennis and Benenati, Chris. (1981). Logic Analysers Symplify System Integration Tasks. Computer Design. March 1981. p.119-126.

Finucci, A. P. G. and MacLeod, I. M. (1981). MAGIC - A Modern Approach to Developing Microprocessor-based Systems. Pulse. November 1981. p. 39-42

Glover, John R. Jr. and Bargainer, James D. (1981). Integrating Hardware and Software in a Computer Engineering Laboratory. IEEE Trans. on Education. VOL. E-24 NO. 1. February 1981. p. 22-27.

Harp, Robert. (1981). Microcomputer Program Development Tools. Computer Design. December 1981. p. 147-150.

Hewlett Packard. (1982). Descriptive Literature on HP64000 Development Systems.

Jackson, S. H. (1980). Development of Microprocessor Crossassemblers and Simulators. MSc. Dissertation, University of the Witwatersrand, Johannesburg, 1980.

Kister, Jack. (1982). The Approaches to the Provision of Tools II. Microprocessor Development and Development Systems. Ed. by Vincent Tseng. Great Britain: Granada, 1982. p. 56-71.

Lee, Sam. The Approaches to the Provision of Tools III. (1982a). Microprocessor Development and Development Systems. Ed. by Vincent Tseng. Great Britain: Granada, 1982. p. 72-84.

Lee, Sam. Software Facilities I. (1982b). Microprocessor Development and Development Systems. Ed. by Vincent Tseng. Great Britain: Granada, 1982. p. 88-101.

Lejeune, Bernard. (1982). In-circuit Emulation I Microprocessor Development and Development Systems. Ed. by Vincent Tseng. Great Britain: Granada, 1982. p. 124-141.

Lumley, Robert M. (1981). An Industrial Microcomputer Education Program. IEEE Trans. on Education. VOL. E-24 NO. 1. May 1981. p. 136-141.

Mihalik, Mike. (1982). In-circuit Emulation II Microprocessor Development and Development Systems. Ed. by Vincent Tseng. Great Britain: Granada, 1982. p. 142-167.

Philips. (1982). Descriptive Literature on PMDS-II Development Systems.

Rodd, M. G. and Gray, G. T. (1979). An Alternative Approach to the Development of Microprocessor Software - A Hardware/Software Simulator. International Journal of Electrical Engineering Education. Vol. 16. p. 183-190.

Saukkonen, Samuli. Karppinen, Martti. and Kemppainen, Pekka. (1982). CAMP: An Environment for Efficient Microprocessor Software Production. Microprocessing and Microprogramming. 9(1982). p. 319-324.

Stenning, Vic et al. (1981). The Ada Environment, A Perspective. Computer. June 1981. p. 26-36.

Tektronix. (1982). Descriptive Literature on 8500 series MDL Development Systems.

Tseng, Vincent. ed. (1982a). Microprocessor Development and Development Systems. Great Britain: Granada, 1982. p. 1-40.

Tseng, Vincent. ed. (1982b). Microprocessor Development and Development Systems. Great Britain: Granada, 1982. p. 195-197.

Wil., B. G. (1983). A Development Facility for Microprogrammable Systems. MSc. Dissertation, University of the Witwatersrand, Johannesburg, 1983.

Zornig, John G. (1981). Buying Time with Talent: The Microprocessor in the Liberal Arts College. IEEE Trans. on Education. VOL. E-24 NO. 1. May 1981. p. 155-159.

APPENDIX A

INTERMEDIATE REPORT - INITIAL ASSESSMENT

WITS MICROPROCESSOR SYSTEMS DEVELOPMENT FACILITY

An Initial Assessment

By : - A.P.G.Finucci
On : - 21-APR-81

1.0 Introduction

1.1 Subject

This report is the initial assessment of the possible ways in which the needs of a Microprocessor Systems Development Facility in the Department of Electrical Engineering may be satisfied .

1.2 Object and Scope

The report has been written to enable a decision to be made regarding the purchase of hardware and software components which are to be integrated into the overall facility .

Only those components which are available at the time of writing are considered for purchase although some components which will become available later in the year are surveyed . As such , the recommendations are valid only in the first half of 1981 .

1.3 Report Organization

The relevant material is presented in three sections , the first one of which covers the requirements of the Development Facility . The second section goes on to survey the available components . The third Section gives a cost and performance breakdown of the components . This enables conclusions to be deduced and recommendations to be made regarding the purchase of the relevant components .

2.0 Development Facility Requirements

These will be discussed under the separate sections of implementation requirements (which relate to the function of the facility) and support requirements (which detail the capabilities) . A final section lists the components required in order to satisfy the requirements as given in the first two sections.

2.1 Implementation Requirements

Some general requirements arise from the fact that the facility is to be used in an academic environment . This implies that the facility must be cost effective . Also it must be generalised , to enable it to be tailored to any future and innovative use . If the facility is generalised , while at the same time maintaining a uniform structure enforced by universal standards , the generalisation can contribute to the cost effectiveness , especially in the long term .

Within the Department the PDP-11 based Systime computer is ideally suited to provide a multi-tasking environment for supporting the facility . It also enables programs which interact in real time with peripherals , to be easily written , thus making it a simpler task to interface specialised components (which may be required to implement the facilty) to it .

The use of this computer will also allow the sharing of such resources as the line printer and a universal programmer .

Another requirement of the system is that it must be reliable and it must come into operation this year in time for it to be used by the final year students for their projects . These requirements imply that it cannot be locally developed within the Department , but that it must be implemented by integrating commercially available software and hardware components . This integration must be achieved with a minimum of locally developed hardware or software .

2.2 Support Requirements

These will be considered under separate sections , the first describing the processors to be supported and the second describing the level at which they will be supported .

2.2.1 Processors to be supported -
The Department is presently committed to supporting the Intel 8080/8085 . Future 8 bit processors which are being considered for support include th Zylog Z80 , and the Intel range of single chip microcontrollers (eg 8748) . Support for the 16-bit processors is not strived for as preference is given to the new generation of 32-bit processors (eg Motorola M68000) . This does not preclude supporting processors such as the Intel 8086 if the components needed for such support are easily obtainable (ie are financed) .

2.2.2 Level of Support -
Level of support for processors falls under two categories , hardware and software . Also , both hardware and software are supported at a Debugging level .

2.2.2.1 Hardware Level -
Processor based hardware can be supported at the chip level or at a board level . Chip level support involves the use of such tools as in-circuit emulation and microprocessor-specific logic analysers . On the other hand , board level support requires a board level emulator which is normally made up of software packages residing in a board level computer and a host computer , these being coonected by a communication link (usually serial eg RS 232) .

The Department is committed to supporting the Intel 8085 at a chip level .

2.2.2.2 Software Level -
Processors can only be effectively supported at two levels of software , the Assembly Language level and the High Level Language level . The first requires an assembler , and the second requires a compiler . To maintain the multi-user quality of the facility these must be cross-assemblers/compilers running on the PDP-11 under its operating System RSX-11M .

The Department is committed to supporting the Intel 8085 at an Assembly Language and at a PLM/85 level .

2.2.2.3 Debugging Level -
Debugging is normally performed with emulation stations (for chip level support) or emulation programs (for board level support) . Because emulation stations are not multi user it is desirable to have some alternate system to provide a means of debugging chip level designs . This means can be given by a cross simulator program which runs on the host computer and simulates the hardware as seen from an emulation station .

2.3 Components Required

2.3.1 Hardware Components -
By considering the previously stated requirements it can be deduced that the following hardware components are required : -

1. An intelligent emulation station which can be easily attached to the computer .

2. A universal programmer which is easily interfaceable and controllable by the computer .

2.3.2 Software Components -
The following components are required : -

1. A Cross Assebler for each processor to be supported , with its relevant support software .

2. A Cross Compiler for each language to be supported on each processor .

3. A Cross Simulator for each processor .

4. Ancillary software to download programs to emulation stations and programmer and to control these peripherals (where required) .

3.0 Survey of Components

The components will be surveyed under headings given by the names of the suppliers of the components .

3.1 Boston Systems Office

This company is primarily a software house specialising in the production of cross software for microprocessor based systems development . They offer an extensive range of cross assemblers and cross simulators covering virtually all types of microprocessors . Their software is of a very high standard , but it is very costly .

BSO also offer Universal Microprocessor Development Systems (UMDS) but these are not supported locally .

3.2 Data I/O Corporation

The speciality of this company is Universal Programmer Systems and they produce a Universal Programmer (System 17) which is ideal for hooking up to a computer . They produce the usual range of personality modules for all PROMS , but additionally , they offer a universal module (Unipak) which , with no modification will allow the Programmer to program approximately 200 different PROMS . They also offer a MOS-PROM specific universal module (Mospak) as well as a low cost MOS-PROM specific Programmer (System 20) .

3.3 GenRad Futuredata

Futuredata universal microprocessor development systems are amongst the most well proven , They do not produce an emulation station , and their development stations are too costly to be used for such a task .

3.4 Hewlett-Packard

The HP64000 Development System is a state of the art product range which includes a Emulation Station (HP64005) but this will only be available later on in the year .

3.7 Intel Corporation

Intel is the only manufacturer reviewed which produces non- universal component , but they are included because of their emulation systems are very advanced and due to the Department's commitment to Intel Processors , they can still provide a cost effective solution to the problem . An Intel development system used as an emulation station is very cheap but it requires some software to be developped in order to hook it up to the host computer . The Department's existing development system (Intellec Series 800) can similarly be used in this fashion .

Intel also offer a cross software package for supporting the 8086/8088 Processors (MDS-311VX) but this will only run on a VAX11 Computer .

Intel is the only company which supports PLM , and this only on their developmxent system , therefore the Intellec station will have to be integrated into the facility .

3.8 [illegible]

[illegible] manufactures the cheapest Universal Programmers on the market , but still having the same [illegible] as any state of the art product , except [illegible] Universal Modules (such as the Data I/O [illegible]) .

3.9 Prolog Corporation

Another Universal Programmer manufacturer , Prolog offers a very expensive programmer system , which unfortunately has been discredited by some PROM manufacturers . Their products are thus not recommended .

3.10 SPL International

This Sofware House which is very active in the field of supporting RTL/2 offers a development facility for 16/32 bit processors (called Magic) on a board level from a RTL/2 Language level . This has been acquired by the Department , and it will be integrated into the facility . So far the facility only supports Intel 8086 , but it will also support the Motorola 68000/6809 and the DEC LSI-11 later in the year .

3.9 Tektronics Corporation

Tektronics markets an emulation station which is fully supported by hSO (the 8001) . It is of relatively old design , thus it will not have any more accessories developed for it . The newer Tektronics station (the 8540) will only be available later in the year . Tektronics also markets a single user Universal development system (the 8550) and a multi user system (the 8560) will be available later in the year.

3.10 Other Manufacturers

These have been superficially surveyed bu their products are not recommended because they are not universal . They include ; Advanced Micro Developments , Fairchild , Motorola , Mostek , RCA , Rockwell .

Emulogics Corporations manufacture universal emulation systems which emulate any processor by using a microprogram . Unfortunately these are not locally supported .

4.0 Conclusions and Recommendations

These will take the form of listing the recommended components to be included in the facility . These will be listed under the separate headings of hardware and software components . The prices of these components are given in the appendix . Not all recommended components are essential , and the list is very long , to enable a final decision to be taken which is dependant on the finances available only .

4.1 Hardware Components

4.1.1 Emulation Stations -

1. Tektronics 8001 with the following additional components : -
   1. Option 5 8085 Emulator
   2. Real Time Prototype Analiser
2. Intel DS-121/80-KIT with MDS-85P-ICE Emulator or MDS-49-ICE Emulator for single chip microcontrollers .
3. Either of the two previous Emulators to be incorporated into the existing Intellec Series 800 MDS

4.1.2 Programmers -

1. Data I/O 990-17XX Programmer with the following modules : -
   1. Unipak 950-0099
   2. Mospak 950-076
   3. The relevant Programming modules and socket adaptors for the required PROMS .
2. Kontron MP80S with RS232 Interface and the required Programming modules and socket adaptors .

4.2 Software Components

4.2.1 Cross Compilers -

As no cross compilers are available at this time , none are recommended . In order to support PLM . an Intellec MDS will have to be connected to the computer . This is an interim solution to the problem .

4.2.2 Cross Assemblers -

The BSO range of cross assemblers is the only one available . It has the following optional recommended support packages :-

1. Cross Linkage Editor
2. Cross Reference Generator

4.2.3 Cross Simulators -

The BSO range of cross simulators is recommended .

4.2.4 Ancillary Support Software -

Support for the Tektronics stations is included with the BSO assemblers . To enable communication with the Intellec station as well as controlling the Programmer some software will have to be written . This will constitute a relavtively simple and short task .

APPENDIX B

MANUFACTURERS OF DEVELOPMENT COMPONENTS

These are listed together with their local representatives:

Advanced Micro Developments (AMD) SCD (Pty) Ltd

AMI Radiokom (Pty) Ltd

Boston Systems Office (BSO) Systime SA (Pty) Ltd

CAP CPP CAP SA (Pty) Ltd

Data I/O EBE (Pty) Ltd

Emulogic Retecon Electronics (Pty) Ltd

Fairchild

Futuredata currently unrepresented but previously represented by Biowave Electronics (Pty) Ltd

Hewlett-Packard Hewlett-Packard SA (Pty) Ltd

Intel EBE (Pty) Ltd

Kontron Biowave Electronics (Pty) Ltd

Millenium Biowave Electronics (Pty) Ltd

Mostek

Motorola ASD (Pty) Ltd

Philips Philips SA (Pty) Ltd

Pro-Log

RCA

Rockwell International

SPL International (UK) Tek Logic (Pty) Ltd

Tektronix Protea PNI (Pty) Ltd

Zilog

APPENDIX C

INTELLEC DISK LABELLING AND NAMING CONVENTIONS

The disks on these stations (running under ISIS-II) are identified by the colour of their labels and their name.

The label colours give the density of the disks and their type of usage, as follows :-

Density	Usage	Label Colour
Single	General	Blue
Single	Owners only	Green
Double	General	Red
Double	Owners only	Yellow
All	Special	Grey

The disk name gives the type of disk and its use or owner as follows :-

The disk name format is :- nnnnnn.xyz
Where :-

nnnnnn - is the field indicating its use or owner
eg. ISIS41 a system disk V4.1

x - indicates the type of disk
eg. S a system disk
N a non-system disk

y - indicates the ownership class
eg. S belongs to the system
U belongs to a user

z - indicates the type of the disk
eg. M a master disk
B a backup disk

APPENDIX D

XFR - FILE TRANSFER UTILITY

The following material is reproduced:

1. Listing of RSX-11 HELP file for XFR.

2. Listing of XFR source in RTL/2.

3. Listing of source of I/O routines used by XFR - XPGD and APGD.

XFR File Transfer Utility

XFR is a utility which is used to transfer ASCII files between the host and its emulation stations.

Further help:

HISTORY COMMAND HP64000

2 HISTORY

XFR was developed at the University of the Witwatersrand in the department of Electrical Engineering. The specification for its protocols enabled it to communicate with the terminal utility provided by the HP64000, using the configuration file 'HOST:' on the 64000 (also HOST.HSR). This was done in July 1982. It was decided, that the same utility could be employed for all emulation stations, therefore software would have to be developed on and for the Intellec Stations to cater for this. This program would perform a function analogous to the terminal utility on the HP64000.

2 COMMAND

The command line conforms to RSX-11 conventions and has the form:

outspec=inspec

The file : transferred from 'inspec' to 'outspec'.

For downloading (ie from the host to the station) 'outspec' is either defaulted or it is 'TI:' whilst 'inspec' is the name of the file to be downloaded.

For uploading (ie from the station to the host) 'outspec' is the name of the of the file to be created on the host and 'inspec' is either defaulted or 'TI:'.

No command file processing has been provided as the protocols employed do not allow unattended operation because the user must synchronise the start of the tranfer. Further help is obtainable by typing:

HELP XFR HP64000

2 HP64000

To use XFR on the HP64000 the station must be plugged into the host and configured using the 'HOST:' command.

Example usage sessions are obtainable by typing:

HELP XFR HP64000 DOWNLOAD

or HELP XFR HP64000 UPLOAD

as applicable.

In these examples all user input is lower case, and all system response is in upper case. Sections preceded by semicolons are explanatory comments.

3 DOWNLOAD

```
>xfr                              ;'filename.abc' is the name of the file
XFR>=filename.abc                 ; on the host to be downloaded.
XFR -- WAITING TO DOWNLOAD FILE
                                  ; At this point the user hits the down-
```

```
                                        ;load key on the station and completes
                                        ; the command. The download will start
                                        ; when the user hits 'return' for the
                                        ; second time.
        XFR — DOWNLOAD COMPLETE         ; And the operation is finished.
        XFR>Z                           ; At this time the user may exit.
        >
        >xfr =filename.abc              ; Single line commands may be employed.
              .
              .

3 UPLOAD

        >xfr                            ;'filename.abc' is the name of the file
        XFR>filename.abc=               ; which will be created on the host.
        XFR — WAITING TO UPLOAD FILE
                                        ; At this point the user hits the up-
                                        ;load key on the station and completes
                                        ; the command. The upload will then
                                        ; start. On completion:
        XFR — UPLOAD COMPLETE           ; And the operation is finished.
        XFR>Z                           ; At this time the user may exit.
        >
        >xfr filename.abc=              ; Single line commands may be employed.
              .
              .
```

```
),XFR.LST=[103,10]XFR

TABLES=
CSNAME=
CNTROL=
RELOCATABLE=
OPTION=
TITLE=
IDENT=
PSECT=
ASECT=
CONCORDANCE=
      1 TITLE XFR -- FILE TRANSFER UTILITY EMPLOYING XON/XOFF PROTOCOL;
 XFR -- FILE TRANSFER UTILITY EMPLOYING XON/XOFF PROTOCOL
      2
      3 %------------------------------------------------------------------%
      4 %                                                                  %
      5 %                            X F R                                 %
      6 %                                                                  %
      7 % THIS PROGRAM TRANSFERS FILES EMPLOYING AN XON/XOFF PROTOCOL.     %
      8 % TRANSFERS CAN BE DONE IN EITHER THE DOWN DIRECTION (FROM THE     %
      9 % HOST TO A DEVICE) OR THE UP DIRECTION (FROM A DEVICE TO THE      %
     10 % HOST).                                                           %
     11 %                                                                  %
     12 % COMMANDLINE;                                                     %
     13 %                          OUTSPEC = INSPEC                        %
     14 %                                                                  %
     15 % THE DEFAULTED IOSPEC IS ASSUMED TO BE TI: WHICH IS CONSIDERED    %
     16 % TO BE THE DEVICE.                                                %
     17 %                                                                  %
     18 % REVISION INFORMATION:                                            %
     19 %                                                                  %
     20 % V1.0  A.P.G.FINUCCI   WITS ELEC ENG   4-OCT-82                   %
     21 % V1.1  A.P.G.FINUCCI   WITS ELEC ENG   26-NOV-82                  %
     22 %       CONVERTED TERMINATION CHARACTER TO SUB OR ^Z OR EOS.        %
     23 %                                                                  %
     24 % FOR THE DOWNLOAD THE XON/XOFF PROTOCOL IS PROVIDED BY THE TT:    %
     25 % DRIVER, WHILE FOR THE UPLOAD IT IS IMPLEMENTED BY THE PROC       %
     26 % XPUTGETDEV. THE COMMAND LINE IS EITHER OBTAINED FROM MCR OR      %
     27 % IT IS PROMPTED FOR EXPLICITLY. NO COMMAND FILE CAPABILITY IS     %
     28 % PROVIDED AS THE DOWNLOAD OR UPLOAD PROCESS REQUIRES MANUAL       %
     29 % INPUT FROM THE OPERATOR TO INITIATE IT ON THE DEVICE ANYWAY.     %
     30 % IT SHOULD BE NOTED THAT THE I/O SUPPORT PACKAGE AT PRESENT       %
     31 % IGNORES TERMINAL CHARACTERISTICS ON OUTPUT, THEREFORE TABS,      %
     32 % FORMFEEDS ETC. ARE PASSED THROUGH DIRECTLY.                      %
     33 %                                                                  %
     34 % PROCEDURES;                                                      %
     35 %                                                                  %
     36 %       RRJOB   GETS THE COMMAND LINE, CALLS RDCHK TO READ IT,     %
     37 %               CHECK IT AND DETERMINE THE DIRECTION OF THE        %
     38 %               TRANSFER, THEREAFTER CALLING EITHER DNLD OR        %
     39 %               UPLD TO TRANSFER THE FILE.                         %
     40 %                                                                  %
     41 %       RDCHK   STREAMS IN THE COMMAND LINE, PARSES IT AND         %
     42 %               CHECKS IT FOR VALIDITY. IT THEN DETERMINES THE     %
     43 %               DIRECTION OF THE TRANSFER.                         %
     44 %                                                                  %
     45 %       ISTI    DETERMINES WHETHER AN IOSPEC CONTAINS 'TI:'.       %
```

```
%
      INFILE    TRANSFERS A FILE FROM THE HOST TO THE DEVICE TI:%
%                                                              %
      TIFILE    TRANSFERS A FILE FROM THE DEVICE TI: TO THE HOST%
%                                                              %
      OPENIN    OPENS FILESPEC FOR STREAM INPUT.               %
%                                                              %
      OPENOUT   OPENS FILESPEC FOR STREAM OUTPUT.              %
%                                                              %
      TINOUT    ROUTES INPUT AND OUTPUT TO TERMINAL.           %
%                                                              %
      FIN       ROUTES INPUT TO FILE.                          %
%                                                              %
      FOUT      ROUTES OUTPUT TO FILE.                         %
%                                                              %
%--------------------------------------------------------------%

%--------------------------------------------------------------%
% STANDARD SUPPORT                                             %
%--------------------------------------------------------------%

% STREAM I/O %

SVC DATA RRSIO;  PROC () BYTE IN;  PROC (BYTE) OUT      ENDDATA;
SVC DATA RRSEI;  BYTE TERMCH,  IOFLAG                   ENDDATA;

EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT   TREAD;
EXT PROC (REF ARRAY BYTE)                       TWRT;

% ERROR RECOVERY %

SVC DATA RRERR; LABEL ERL; INT ERN; PROC(INT) ERP       ENDDATA;

EXT PROC (INT)                              RRGEL, RRERP;

%--------------------------------------------------------------%
% SYSTEM DEPENDANT SUPPORT -- DECI RSX-11 INTERFACE PACKAGE    %
%--------------------------------------------------------------%

% ABBREVIATIONS %

LET RAB = REF ARRAY BYTE;
LET RAI = REF ARRAY INT;

%-------------%
% INPUT/OUTPUT %
%-------------%

% I/O CHANNELS %

MODE IOCL       (RAB BFR, INT N, IN, PTR, MD, TRM);

SVC DATA RRCHAN; REF IOCL INCL,  OUTCL                  ENDDATA;

% PHYSICAL DEVICE I/O %
% ------------------- %

EXT PROC (INT, RAB)                         ASSIGN;
EXT PROC (INT)                              DELAY;
```

```
% DEVICE LOCKING %

EXT PROC ()                                 STARTIN, STARTOUT;
EXT PROC ()                                 ENDIN, ENDOUT;

% STREAM I/O PROCS %

EXT PROC () BYTE                            TTIN, HSIN;
EXT PROC (BYTE)                             TTOUT, HSOUT;

% ADVANCED FUNCTIONS %

EXT PROC ()                                 GETMCR;
EXT PROC (INT, INT) INT                     APUTGETIEV, XPUTGETIEV;

% FILE I/O %
% ---------- %

MODE FNBLK      (INT DEVN, REF BYTE DEVT,
                 INT DIRN, REF BYTE DIRT,
                 INT NAMN, REF BYTE NAMT);

EXT DATA FILE01;        RAI FIBTBL;     ENDDATA;

% GENERAL %

EXT PROC (RAI)                              FINIT;
EXT PROC (INT) INT                          FCLOSE, FFRINT;
EXT PROC (RAB, REF FNBLK)                   FNAME;

% STREAM ACCESS %

EXT PROC (REF FNBLK, INT, INT) INT      FSTRTIN, FSTRTOUT;
EXT PROC () BYTE                            INF;
EXT PROC (BYTE)                             OUTF;

% ASCII CONTROL CHARACTERS %
% ------------------------ %

LET ETX = 3;
LET SUB = 26;
LET NL  = 10;
LET XON = 17;
LET EOS = OCT 200;

%-------------------------------------------------------------------------%
% GLOBAL DATA DECLARATIONS                                                %
%-------------------------------------------------------------------------%

% I/O CHANNELS %
% ------------ %

DATA TIO;

        IOCL    TICL := (TIBF, 0, 1, 0, 0, 0),
                TOCL := (TOBF, 0, 1, 0, 0, 0);
        ARRAY (256) BYTE TIBF, TOBF;

ENDDATA;
```

```
166 DATA FIO;
167
168         IOCL    FCL := (FBF, 0, 2, 0, 0, 0);
169         ARRAY (256) BYTE FBF;
170         ENBLK   FNB := (0, DUMMY, 0, DUMMY, 0, DUMMY);
171         BYTE DUMMY;
172
173 ENDDATA;
174
175 % COMMAND LINE %
176 % ------------ %
177
178 DATA IOSPECS;
179
180         ARRAY (30) BYTE ISPEC, OSPEC;
181         INT ISLEN, OSLEN;
182
183 ENDDATA;
184
185 %-- PROCEDURE RRJOB -- ENTRY POINT ------------------------------%
186
187 ENT PROC RRJOB ();
188
189         INT     HASCML, % FLAG TO INDICATE WHETHER A COMMAND %
190                         % LINE WAS PRESENT. %
191                 RICSTS, % READ CHECK STATUS %
192                 AGAIN := 1,
193                 EOFLAG;
194
195         % INITIALISATIONS %
196
197         TINOUT (); OUT (NL);    % TERMINAL I/O %
198         FINIT (FIOTBL);         % FILE I/O %
199
200         GETMCR ();      % GET COMMAND LINE %
201         HASCML := INCL.N - INCL.PTR;
202
203         WHILE AGAIN = 1 DO
204                 IF HASCML <= 0 THEN
205                         TWRT('XFR)');
206                         APUTGETIEV (61, 0);
207                         EOFLAG := INCL.N;
208                         OUT (NL);
209                 END;
210
211                 IF INCL.PTR < INCL.N THEN
212                         RICSTS := RICHK (); % READ IT AND CHECK IT. %
213
214                         IF RICSTS < 0
215                                 THEN    TWRT ('XFR -- #
216                                 #ILLEGAL FUNCTION IN COMMAND#NL#');
217                         ELSEIF RICSTS = 1
218                                 THEN    INLD (ISPEC, ISLEN);
219                         ELSEIF RICSTS = 0
220                                 THEN    UPLD (OSPEC, OSLEN);
221                         END;
222                 END;
223
224                 IF EOFLAG = -1 OR HASCML > 0
225                         THEN    AGAIN := 0;
```

```
226                 END;
227         REP;
228 ENDPROC;
229
230 %-- PROC RICHK -- READ COMMAND AND CHECK IT ----------------------------%
231 %                                                                       %
232 % RETURNED STATUS:                                                     %
233 %                                                                       %
234 %        1      TRANSFER IS TO TI:                                     %
235 %        0      TRANSFER IS FROM TI:                                   %
236 %       -1      INVALID FUNCTION IN COMMAND                            %
237 %                                                                       %
238 %-----------------------------------------------------------------------%
239
240 PROC RICHK () INT;
241
242         OSLEN := TREAD (OSPEC, '=#NL,EOS#');
243
244         IF TERMCH # '=' THEN RETURN (-1); END;
245
246         ISLEN := TREAD (ISPEC, '#NL,EOS#');
247
248         IF              ISTI (OSPEC, OSLEN) = 1
249                 AND     ISTI (ISPEC, ISLEN) = 1
250                 THEN    RETURN (-1);
251         END;
252
253         IF ISTI (OSPEC, OSLEN) = 1 THEN RETURN (1); END;
254         IF ISTI (ISPEC, ISLEN) = 1 THEN RETURN (0); END;
255
256         RETURN (-1);
257 ENDPROC;
258
259 %-- PROC ISTI -- DETERMINES IF STRING IS NULL OR = 'TI:' ---------------%
260 %                                                                       %
261 % RETURNED VALUE:                                                     %
262 %                                                                       %
263 %       1       STRING IS NULL OR EQUAL TO 'TI:'                        %
264 %       0       STRING IS NOT NULL AND NOT EQUAL TO 'TI:'               %
265 %                                                                       %
266 %-----------------------------------------------------------------------%
267
268 PROC ISTI (REF STRING, INT STRLEN) INT;
269
270         IF STRLEN = 0 THEN RETURN (1); END;
271
272         IF              STRING(1) = 'T'
273                 AND     STRING(2) = 'I'
274                 AND     STRING(3) = ':'
275
276                 THEN    RETURN (1);
277         END;
278
279         RETURN (0);
280 ENDPROC;
281
282 %-- PROC DMLD -- TRANSFER FILE TO DEVICE -------------------------------%
283
284 PROC DMLD (REF FSPEC, INT FSLEN);
285
```

```
286         INT STS; BYTE CHAR;
287
288         STS := OPFIN (FSPEC, FSLEN);
289
290         IF STS < 0 THEN
291                 TWRT ("XFR -- #
292                 #UNABLE TO OPEN INPUT FILE#NL#");
293                 RETURN;
294         END;
295
296         TWRT ("XFR -- WAITING TO DOWNLOAD FILE");
297         STS := APUTGETIEV (1, 4);
298         IF STS < 0 THEN FCLOSE (-FCL.IV);
299                         TWRT (" -- RESPONSE TIMEDUT#NL#");
300                         RETURN ;
301         END;
302
303         STARTOUT ();
304         CHAR := IN ();
305         IF CHAR = EOS THEN RETURN; END;
306
307         FIN ();
308         CHAR := IN ();
309
310         WHILE CHAR # EOS DO
311
312                 OUT (CHAR);
313                 CHAR := IN ();
314
315         REP;
316
317         OUT (SUB); OUT (NL);
318         ENDOUT ();
319         TINOUT ();
320         DELAY (10);
321         TWRT ("XFR -- DOWNLOAD COMPLETED#NL#");
322
323 ENDPROC;
324
325 %-- PROC UPLD -- TRANSFER DEVICE TO FILE ----------%
326
327 PROC UPLD (REF FSPEC, INT FSLEN);
328
329         INT STS, AGAIN := 1; BYTE CHAR;
330
331         STS := OPFOUT (FSPEC, FSLEN);
332         IF STS < 0
333                 THEN    TWRT ("XFR -- #
334                         #UNABLE TO OPEN OUTPUT FILE#NL#");
335                         RETURN;
336         END;
337
338         STARTIN ();
339         TWRT ("XFR -- WAITING TO UPLOAD FILE"); OUT (NL);
340
341         WHILE AGAIN = 1 DO
342                 STS := XPUTGETIEV (255, 2);
343
344                 IF STS < 0 THEN
345                         TWRT ("#XON#XFR -- RESPONSE TIMEOUT#NL#");
```

```
346                         ENDIN ();
347                         FCLOSE (-FCL.IN);
348                         RETURN;
349                 END;
350
351                 CHAR := IN ();
352                 FOUT ();
353                 WHILE           CHAR # NL
354                         AND     CHAR # EOS
355                         AND     CHAR # ETX
356                 IO
357                         OUT (CHAR);
358                         CHAR := IN ();
359                 REP;
360                 IF CHAR = EOS OR CHAR = ETX
361                         THEN AGAIN := 0;
362                 ELSEIF CHAR = NL
363                         THEN OUT (NL);
364                 END;
365                 TINOUT ();
366         REP;
367
368         ENDIN ();
369         FCLOSE (-FCL.IN);
370         TWRT ('$XON#XFR -- UPLOAD COMPLETE!#NL$');
371
372 ENDPROC;
373
374 %-- PROC OPFIN -- OPEN FILE FOR STREAM INPUT ----------%
375 %                                                     %
376 % RETURN CODES ARE LIKE THOSE FROM FSTRTIN.           %
377 %                                                     %
378 %-----------------------------------------------------%
379
380 PROC OPFIN (RAB FSPEC, INT FSLEN) INT;
381
382         INT STS;
383
384         FSPEC(FSLEN + 1) := NL;
385         FNAME (FSPEC, FNB);
386         FIN ();
387
388         STS := FSTRTIN (FNB, 1, 2);
389
390         TINOUT ();
391
392         RETURN (STS);
393
394 ENDPROC;
395
396 %-- PROC OPFOUT -- OPEN FILE FOR STREAM OUTPUT --------%
397 %                                                     %
398 % RETURN CODES ARE AS PER FSTRTOUT.                   %
399 %                                                     %
400 %-----------------------------------------------------%
401
402 PROC OPFOUT (RAB FSPEC, INT FSLEN) INT;
403
404         INT STS;
405
```

```
346                         ENDIN ();
347                         FCLOSE (-FCL.IN);
348                         RETURN;
349                 END;
350
351                 CHAR := IN ();
352                 FOUT ();
353                 WHILE           CHAR # NL
354                         AND     CHAR # EOS
355                         AND     CHAR # ETX
356                 IO
357                         OUT (CHAR);
358                         CHAR := IN ();
359                 REP;
360                 IF CHAR = EOS OR CHAR = ETX
361                         THEN AGAIN := 0;
362                 ELSEIF CHAR = NL
363                         THEN OUT (NL);
364                 END;
365                 TINOUT ();
366         REP;
367
368         ENDIN ();
369         FCLOSE (-FCL.IN);
370         TWRT ("#XON#XFR -- UPLOAD COMPLETE!#NL#");
371
372 ENDPROC;
373
374 %-- PROC OPFIN -- OPEN FILE FOR STREAM INPUT ----------%
375 %                                                      %
376 % RETURN CODES ARE LIKE THOSE FROM FSTRTIN.            %
377 %                                                      %
378 %------------------------------------------------------%
379
380 PROC OPFIN (RAB FSPEC, INT FSLEN) INT;
381
382         INT STS;
383
384         FSPEC(FSLEN + 1) := NL;
385         FNAME (FSPEC, FNB);
386         FIN ();
387
388         STS := FSTRTIN (FNB, 1, 2);
389
390         TINOUT ();
391
392         RETURN (STS);
393
394 ENDPROC;
395
396 %-- PROC OPFOUT -- OPEN FILE FOR STREAM OUTPUT ---------%
397 %                                                      %
398 % RETURN CODES ARE AS PER FSTRTOUT.                    %
399 %                                                      %
400 %------------------------------------------------------%
401
402 PROC OPFOUT (RAB FSPEC, INT FSLEN) INT;
403
404         INT STS;
405
```

```
406         FSPEC(FSLEN + 1) := NL;
407         FNAME(FSPEC, FNB);
408         FOUT ();
409
410         STS := FSTRTOUT (FNB, 1, 2);
411
412         TINOUT ();
413
414         RETURN (STS);
415
416 ENDPROC;
417
418 %— PROC TINOUT — SWITCH INPUT & OUTPUT STREAMS TO TERMINAL ——%
419
420 PROC TINOUT ();
421
422         INCL := TICL;   OUTCL := TOCL;
423         IN   := TTIN;   OUT   := TTOUT;
424
425 ENDPROC;
426
427 %— PROC FIN — SWITCH INPUT STREAM TO FILE ————————————%
428
429 PROC FIN ();
430
431         INCL := FCL;
432         IN   := INF;
433
434 ENDPROC;
435
436 %— PROC FOUT — SWITCH OUTPUT STREAM TO FILE ——————————%
437
438 PROC FOUT ();
439
440         OUTCL := FCL;
441         OUT   := OUTF;
442
443 ENDPROC;
```

USE OF RESOURCES

RESOURCE	MAX	USED
IDENTIFIERS (I)	552	107
VARIABLES (I)	552	55
ATTRIB LVL (D)	345	52
ID NAMES (D)	345	96
NAME CHARS (N)	2078	448
GEN. LABELS (G)	345	31
CONSTANT POOL (C)	1380	10
STRING POOL (S)	4142	294
ARRAY BOUNDS (A)	69	2
MORE INFO. (G)	690	151

BLOCK SIZES (BYTES)

BLOCK	DECIMAL	OCTAL

'SPOOL'	380	454
TIO	548	1034
FIO	284	434
IOSPEC	68	104
RRJOB	204	314
RICH	168	258
ISTI	70	106
INLI	192	300
UPLI	244	364
UPFIN	78	116
OPFOUT	78	116
TINOUT	36	44
FIN	24	30
FOUT	24	30
'TOTAL'	2310	4406

COMPILATION OK

```
),XPGU,LST=[103,10]XPGU

TABLES=
CSNAME=
CNTRL=F
RELOCATABLE=
OPTION=
TITLE=
IDENT=
PSECT=
ASECT=
CONCORDANCE=
     1 TITLE XPUTGETDEV -- PROC TO SUPPORT I/O WITH XON/XOFF PROTOCOLS;
 XPUTGETDEV -- PROC TO SUPPORT I/O WITH XON/XOFF PROTOCOLS
     2
     3 %------------------------------------------------------------------%
     4 %                                                                  %
     5 %                        X P U T G E T D E V                       %
     6 %                                                                  %
     7 % THIS PROC ALLOWS FIXED LENGTH I/O TO BE PERFORMED WITH THE       %
     8 % STANDARD STREAM I/O MECHANISM.  IT PERFORMS RECORD I/O TO        %
     9 % A DEVICE USING THE SAME BUFFERS AND CHANNEL AS THE STANDARD      %
    10 % RECORD I/O PROCS, AND IT IS IDENTICAL TO PUTGETDEV EXCEPT        %
    11 % THAT IT SENDS AN XON AFTER THE PROMPT AND AN XOFF AFTER THE      %
    12 % INPUT IS COMPLETE.                                               %
    13 %                                                                  %
    14 % 1.0 21-JUN-82 A.P.G.FINUCCI   WITS ELEC ENG                      %
    15 %                                                                  %
    16 % THIS PROCEDURE DUMPS THE CURRENT OUTPUT BUFFER (AS FILLED BY     %
    17 % TTOUT OR HSOUT) FOLLOWED BY AN XON AND INDIVISIBLY, FILLS THE    %
    18 % INPUT BUFFER TO A SPECIFIED LENGTH (LEN) WITH A SPECIFIED        %
    19 % TIMEOUT VALUE (TMO).                                             %
    20 % THE VALUE -1 IS RETURNED IF TIMEOUT OCCURRED.                    %
    21 %                                                                  %
    22 %       LEN - THE NUMBER OF CHARACTERS REQUIRED FOR INPUT          %
    23 %       TMO - THE TIMEOUT VALUE IN TENS OF SECONDS                 %
    24 %                                                                  %
    25 % THE FOLLOWING CHANNEL RECORDS ARE AFFECTED:                      %
    26 %       OUTPUT  USES    OUCL.BFR( 1 TO OUTCL.PTR + 1)              %
    27 %       INPUT   USES    INCL.BFR( 1 TO LEN )                       %
    28 %       BOTH    USE     LU = INCL.LU                               %
    29 %       AFTERWARDS,     XXXCL.PTR := 0                             %
    30 %                       OUTCL.N   := LENGTH OUTCL.BFR              %
    31 %                        INCL.TRM := TERMINATING CHAR              %
    32 %                        INCL.N   := NO OF CHARACTERS              %
    33 %                                    ACTUALLY INPUT                %
    34 %                                                                  %
    35 % ERRORS:                                                          %
    36 %       ERRCL     (21)   OUTPUT DIRECTIVE ERROR                    %
    37 %       ERR      (121)   I/O ERROR, IOFLAG := 2                    %
    38 %                                                                  %
    39 %       ERR     (5000)   OUTPUT OVERFLOWS BUFFER DUE TO XON        %
    40 %       ERR     (5001)   LEN > LENGTH INCL.BFR                     %
    41 %                                                                  %
    42 %       UPON TIMEOUT, IOFLAG := 3                                  %
    43 %                                                                  %
    44 % THIS MODULE IS RE-ENTRANT.                                       %
    45 %                                                                  %
```

```
46 % THIS MODULE USES IMBEDDED CODE AND MUST BE COMPILED WITH A     %
47 % /ON:F SWITCH.                                                 %
48 %                                                               %
49 %---------------------------------------------------------------%
50
51 OPTION (1) EI ;
52
53 MODE IOCL  (REF ARRAY BYTE BFR, INT N, IN, PTR, MD, TRM);
54 MODE IOXD  (INT IOST1, IOST2, BITS, TMO);
55 MODE TFREC (INT TN, TEF, IOEF, MEF);
56
57 SVC DATA RRSED;  BYTE TERMCH, IOFLAG                    ENDDATA;
58 SVC DATA RRERR;  LABEL ERL; INT ERN; PROC (INT) ERP     ENDDATA;
59 SVC DATA RRCHAN; REF IOCL INCL, OUTCL                   ENDDATA;
60 SVC DATA RRERRX; INT LINENO; BYTE UEFLAG, ERRLUN; INT REXISW ENDDATA;
61 SVC DATA RRIOX;  REF IOXD INXD, OUTXD; REF TFREC TF     ENDDATA;
62
63 EXT PROC (INT)  RRGEL;
64
65 LET DC1 = 17;
66 LET XON = DC1;
67
68 %------------------------------------ PROC XPUTGETDEV ---%
69
70 ENT PROC XPUTGETDEV (INT LEN,TMO) INT:
71
72         INT             RES     :=0,
73                         LU      := INCL.IN,
74                         FLAG    := OUTCL.MD LOR INCL.MD,
75                         OLN     := OUTCL.PTR + 1,
76                         SEQ, SB;
77         REF BYTE        OBF     := OUTCL.BFR(1),
78                         IBF     := INCL.BFR(1);
79
80         OUTCL.PTR := 0;         % PURGE OUTPUT BUFFER %
81
82         IF OLN > LENGTH OUTCL.BFR
83                 THEN    ERP (5000);
84                         GOTO EXIT;
85         END;
86
87         OUTCL.BFR(OLN) := XON;
88
89         IF LEN > LENGTH INCL.BFR
90                 THEN    ERP (5001);
91                         GOTO EXIT;
92         END;
93
94 CODE 106,0;
95         .LIST   MEB
96         .MCALL  QIOSY$,TTSYM$,QIOW$S
97         QIOSY$
98         TTSYM$
99         MOV     #IO.RPR!TF,BIN!TF,TMO!TF,XOF,R3
100                *% READ AFTER BINARY PROMPT, WITH TIMEOUT AND XOFF %
101        BIT     #100,*FLAG(R5)          *% TEST NO-ECHO MODE          %
102        BEQ     1$                      *% R3 - FUNCTION CODE         %
103        BIS     #TF.INC,R3              *% SET NO-ECHO MODE FOR READ  %
104 1$:    TST     *LEN(R5)        *% CHECK THAT THERE ARE CHARACTERS    %
105        BLE     *OK             *% IN BUFFER, ELSE DO NOTHING.        %
```

```
106
107         MOV     R5,R1                       %% R1 - ADDRESS OF IOSB %
108         ADD     #%SB,R1
109 QIOW%S   R3,%LU(5),#1,,R1,,(%IEF(5),%LEN(5),%TMO(5),%DEF(5),%ULN(5))
110         MOV     $IOSW,%RSXIOSW/RRERRX(R0)
111         BCS     %IERR
112
113         CMPB    #IE.EOF,%SB(R5)
114         BEQ     %EOF
115
116         CMPB    #IS.TMO,%SB(R5)
117         BEQ     %TMIO
118
119         MOVB    %SB(R5),R1              %% WIDEN AND TEST SIGN  %
120         MOV     R1,%RSXIOSW/RRERRX(R0)
121         BGT     %OK                     %% I/O SUCCESSFUL       %
122 %RTL;
123
124 %-- I/O ERROR: TREAT IT AS RECOVERABLE, WITH STATUS IN RSXIOSW --%
125
126         IOFLAG := 2;
127         ERR (121);
128         GOTO EXIT;
129
130 %-- DIRECTIVE ERROR: TREAT AS FATAL, WITH IOSW IN RSXIOSW/RRERRX --%
131
132 IERR:   RRGEL(21);
133
134 %-- END OF FILE: FLAG IT FOR TTIN TO TELL USER LATER          --%
135
136 EOF:    INCL.N := -1;
137         GOTO ZERO;
138
139 %-- TIMEOUT OCCURRED: SET IOFLAG AND RETURN THE VALUE 1       --%
140
141 TMIO:   IOFLAG  := 3;
142         RES     := -1;
143
144 %-- ALL OK: SO RESET CHANNEL POINTERS AND RETURN TERMINATING  --%
145 %--         CHARACTER.                                        --%
146
147 OK:     INCL.N   := SIZ;        % CHARACTERS ACTUALLY INPUT     %
148         INCL.TRM := SB SRL 8;   % TERMINATING CHARACTER         %
149 ZERO:   INCL.PTR := 0;          % BUFFER POINTER                 %
150
151 EXIT:   RETURN (RES);
152
153 ENDPROC;
```

%%COMPILED AS SYSTEMS MODULE%%

%USE OF RESOURCES%

RESOURCE	MAX	USED
IDENTIFIERS (I)	552	56
VARIABLES (I)	552	44

GT PRIO. L.A.	345	12
ID NAMES (ID	345	5E
NAME CHARS (N	2076	216
GEN. LABELS (G	345	2
CONSTANT POOL (C	1380	13
STRING POOL (S	4142	2
ARRAY BOUNDS (A	69	0
MODE INFO. (O	696	41

PRIOR SIZES (BYTES)

PRIO	DECIMAL	OCTAL
'SPOOL'	0	0
XPUTGE	340	524
'TOTAL'	340	524

COMPILATION OK

```
>,APGD.LST=[103,10]APGD

TABLES=
CSNAME=
CNTROL=F
RELOCATABLE=
OPTION=
TITLE=
IDENT=
PSECT=
ASECT=
CONCORDANCE=
     1 TITLE APUTGETDEV — PROC TO SUPPORT INDIVISIBLE FIXED LENGTH I/O;
APUTGETDEV — PROC TO SUPPORT INDIVISIBLE FIXED LENGTH I/O
     2
     3 %------------------------------------------------------------------%
     4 %                                                                  %
     5 %                        A P U T G E T D E V                       %
     6 %                                                                  %
     7 % THIS PROC ALLOWS FIXED LENGTH I/O TO BE PERFORMED WITH THE       %
     8 % STANDARD STREAM I/O MECHANISM. IT PERFORMS RECORD I/O TO         %
     9 % A DEVICE USING THE SAME BUFFERS AND CHANNEL AS THE STANDARD      %
    10 % RECORD I/O PROCS, AND IT IS FUNCTIONALLY SIMILAR TO GETCOM'      %
    11 % IN THE DEVIO MODULE. IT IS IDENTICAL TO PUTGETDEV EXCEPT THAT    %
    12 % IT DOES NOT USE A BINARY PROMPT BUT AN ASCII PROMPT IN THE       %
    13 % QIO.                                                            %
    14 %                                                                 %
    15 % 1.0 13-JUL-82 A.P.G.FINUCCI -WITS ELEC ENG-                     %
    16 %                                                                 %
    17 % THIS PROCEDURE DUMPS THE CURRENT OUTPUT BUFFER (AS FILLED BY    %
    18 % TTOUT OR HSOUT) AND INDIVISIBLY, FILLS THE INPUT BUFFER TO A     %
    19 % SPECIFIED LENGTH (LEN) WITH A SPECIFIED TIMEOUT (TMO).          %
    20 % THE VALUE -1 IS RETURNED IF TIMEOUT OCCURRED.                   %
    21 %                                                                 %
    22 %       LEN - THE NUMBER OF CHARACTERS REQUIRED FOR INPUT         %
    23 %       TMO - THE TIMEOUT VALUE IN TENS OF SECONDS                %
    24 %                                                                 %
    25 % THE FOLLOWING CHANNEL RECORDS ARE AFFECTED:                     %
    26 %       OUTPUT  USES    OUCL.BFR( 1 TO OUTCL.PTR )                %
    27 %       INPUT   USES    INCL.BFR( 1 TO LEN )                      %
    28 %       BOTH    USE     LU = INCL.DV                              %
    29 %       AFTERWARDS,     XXXCL.PTR := 0                            %
    30 %                       OUTCL.N   := LENGTH OUTCL.BFR             %
    31 %                        INCL.TRM := TERMINATING CHAR             %
    32 %                        INCL.N   := NO OF CHARACTERS             %
    33 %                                    ACTUALLY INPUT               %
    34 %                                                                 %
    35 % ERRORS:                                                         %
    36 %       RNGEL    (21) = OUTPUT DIRECTIVE ERROR                    %
    37 %       ERR     (121) = I/O ERROR, IOFLAG := 2                    %
    38 %                                                                 %
    39 %       UPON TIMEOUT, IOFLAG := 3                                 %
    40 %                                                                 %
    41 % THIS MODULE IS RE-ENTRANT.                                      %
    42 %                                                                 %
    43 % THIS MODULE USES IMBEDDED CODE AND MUST BE COMPILED WITH A      %
    44 % /DN:F SWITCH.                                                   %
    45 %                                                                 %
```

```
46 %-------------------------------------------------------------------%
47
48 OPTION (1) EI ;
49
50 MODE IOCL  (REF ARRAY BYTE BFR, INT N, IV, PTR, MD, TRM);
51 MODE IOXD  (INT IOST1, IOST2, BITS, TMO);
52 MODE TFREC (INT TN, TEF, IOEF, MEF);
53
54 SVC DATA RRSED;  BYTE TERMCH, IOFLAG                        ENDDATA;
55 SVC DATA RRERR;  LABEL ERL; INT ERN; PROC (INT) ERP         ENDDATA;
56 SVC DATA RRCHAN; REF IOCL INCL, OUTCL                      ENDDATA;
57 SVC DATA RRERRX; INT LINENO; BYTE UEFLAG, ERRLUN; INT RSXISW ENDDATA;
58 SVC DATA RRIOX;  REF IOXD INXD, OUTXD; REF TFREC TF       ENDDATA;
59
60 EXT PROC (INT)  RRGEL;
61
62 %------------------------------------- PROC APUTGETIEV ---%
63
64 ENT PROC APUTGETIEV (INT LEN,TMO) INT;
65
66         INT             RES     :=0,
67                         LU      := INCL.IV,
68                         FLAG    := OUTCL.MD LOR INCL.MD,
69                         OLN     := OUTCL.PTR,
70                         SR2, SB;
71         REF BYTE        OBF     := OUTCL.BFR(1),
72                         IBF     := INCL.BFR(1);
73
74         OUTCL.PTR := 0;         % PURGE OUTPUT BUFFER %
75
76         IF LEN > LENGTH INCL.BFR
77                 THEN RRGEL (5000);
78         END;
79
80 CODE 106,0;
81         .LIST   MEB
82         .MCALL  QI      ,TTSYM$,QIOW$S
83         QIOSY$
84         TTSYM$
85         MOV     #IO.RTR!TF.TMO,R3       %% TIMED READ AFTER PROMPT   %
86         BIT     #100,%FLAG(R5)          %% TEST NO-ECHO MODE         %
87         BEQ     1$                      %% R3 - FUNCTION CODE        %
88         BIS     #TF.RNE,R3              %% SET NO-ECHO MODE FOR READ %
89 1$:     TST     %LEN(R5)        %% CHECK THAT THERE ARE CHARACTERS   %
90         BLE     %ON             %% IN BUFFER, ELSE DO NOTHING.       %
91
92         MOV     R5,R1                   %% R1 - ADDRESS OF IOSB %
93         ADD     #%SB,R1
94 QIOW$S  R3,%LU(5),#1,,R1,,(%IBF(5),%LEN(5),%TMO(5),%OBF(5),%OLN(5))
95         MOV     #IOSW,%RSXISW/RRERRX(R0)
96         BCS     %IOERR
97
98         CMPB    #IE.EOF,%SB(R5)
99         BEQ     %EOF
100
101        CMPB    #IS.TMO,%SB(R5)
102        BEQ     %TMOO
103
104        MOVB    %SB(R5),R1              %% WORD AND TEST SIGN  %
105        MOV     R1,%RSXISW/RRERRX(R0)
```

```
106         BGT     *OK                     %! I/O SUCCESSFUL       %
107 *RTL;
108
109 %-- I/O ERROR: TREAT IT AS RECOVERABLE, WITH STATUS IN RSXIOSW --%
110
111         IOFLAG := 2;
112         ERR (121);
113         GOTO EXIT;
114
115 %-- DIRECTIVE ERROR: TREAT AS FATAL, WITH IOSW IN RSXIOSW/RRERX --%
116
117 DERR:   RAISE(21);
118
119 %-- END OF FILE: FLAG IT FOR TTIN TO TELL USER LATER            --%
120
121 EOF:    INCL.N := -1;
122         GOTO ZERO;
123
124 %-- TIMEOUT OCCURRED: SET IOFLAG AND RETURN THE VALUE 1         --%
125
126 TMIO:   IOFLAG  := 3;
127         RES     := -1;
128
129 %-- ALL OK: SO RESET CHANNEL POINTERS AND RETURN TERMINATING    --%
130 %--         CHARACTER.                                          --%
131
132 OK:     INCL.N   := SE2;        % CHARACTERS ACTUALLY INPUT      %
133         INCL.TRM := SR SRL 8;   % TERMINATING CHARACTER          %
134 ZERO:   INCL.PTR := 0;          % BUFFER POINTER                 %
135
136 EXIT:   RETURN (RES);
137
138 ENDPROC;
```

COMPILED AS SYSTEMS MODULE

USE OF RESOURCES

RESOURCE	MAX	USED
IDENTIFIERS (I)	552	54
VARIABLES (I)	552	44
AT BRICK LVL (I)	345	10
ID NAMES (I)	345	53
NAME CHARS (N)	2078	210
GEN. LABELS (G)	345	1
CONSTANT POOL (C)	1380	10
STRING POOL (S)	4142	2
ARRAY BOUNDS (A)	69	0
MODE INFO. (G)	690	41

BRICK SIZES (BYTES)

BRICK	DECIMAL	OCTAL
'SPOOL'	0	0

```
APUTGE      298    452
'TOTAL'     298    452
COMPILATION OK
```

Author Finucci A P G
Name of thesis The Provision of Microprocessor-based systems development facilities 1984

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2013

LEGAL NOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed, or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page) for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.