SPECIAL REPORT

PERS 227 FEASIBILITY STUDY OF TIME-SHARING ON A MINI-COMPUTER

NATIONAL INSTITUTE FOR PERSONNEL RESEARCH
COUNCIL FOR SCIENTIFIC AND INDUSTRIAL RESEARCH

CSIR Special Report PERS 227 (pp. 1 - 91)
UDC 681.322.063.4:681.322.063.31/.32
Johannesburg, Republic of South Africa, July 1975

NATIONAL INSTITUTE FOR PERSONNEL RESEARCH
COUNCIL FOR SCIENTIFIC AND INDUSTRIAL RESEARCH

CSIR Special Report PERS 227 (pp. 1-91)
UDC 681.322.063.4:681.322.063.31/.32
Johannesburg, Republic of South Africa, July 1975

ISBN 0 7988 0740 7

CSIR Special Report PERS 227

Published by

National Institute for Personnel Research
Council for Scientific and Industrial Research
P.O. Box 10319
Johannesburg 2000
Republic of South Africa 1975

Printed in the Republic of South Africa by
National Institute for Personnel Research

S U M M A R Y

At the time when the purchase of a computer for data acquisition and control of experiments was under consideration, a time-sharing system seemed to be indicated to handle the heavy work load expected, but there was some doubt about the feasibility of operating such a system on a mini-computer in the NIPR environment. To avoid the possibility of purchasing equipment which might not prove adequate, the suppliers were asked to lend the NIPR those items which were specifically required for time-sharing so that a feasibility study could be carried out.

The feasibility of the system was determined by experience with it in real-time data collection in on-going projects, and by experiments in which the effects of time-sharing were measured during crucial operations.

The results of the study showed that the system performs adequately and that it satisfies the needs of researchers. On the Varian mini-computer the VORTEX operating system, despite its large core overheads, provides an effective time-sharing system that enables real-time programs to be run together with program development and testing. The terminals provided gave adequate control from remotely situated laboratories.

The study showed that time-sharing real-time data collection is feasible in the NIPR environment and therefore it is recommended that the loaned equipment be purchased.

A C K N O W L E D G E M E N T S

This project was directed by Mr D.J.M. Vorster, Director of the National Institute for Personnel Research and was carried out as part of the activities of the Automation and Computer Services Division. Special thanks are due to Mrs F. Mekhoe and Mrs N.J. Twala for typing the text and to Mr A.D. Molatedi for drawing the graphs.

Table of Contents

		Page
0	INTRODUCTION	1
	Structure of the Report	2
1	PURPOSES FOR WHICH THE NIPR REQUIRED A MINI-COMPUTER	3
	1.1 Neuropsychology Division	3
	1.2 Temperament and Personality Division	5
	1.3 Psychometrics Division	6
2	EXPECTED USAGE OF THE MINI-COMPUTER	7
3	PROS AND CONS OF PURCHASING A TIME-SHARING MINI-COMPUTER	9
4	PROPOSAL TO UNDERTAKE A FEASIBILITY STUDY	12
5	HARDWARE AND SOFTWARE USED FOR THE FEASIBILITY STUDY	13
	5.1 Configuration Chosen	13
	5.2 Computing Facilities	13
	5.3 Camac Capabilities	14
	5.4 Resource Allocation	14
6	DETAILED DESCRIPTION OF RELEVANT HARDWARE AND ITS OPERATION	16
	6.1 Varian 73 Interrupt and I/O Options	16
	6.2 Camac System	18
	6.2.1 Introduction of the Camac Concept	18
	The Camac Control Hierarchy	19
	The Camac LAM Concept	19
	6.2.2 The Camac System at the NIPR	21
	Introduction	21
	Block Transfer of Data	22
7	RELEVANT SOFTWARE	24
	7.1 The Vortex Operating System	24
	7.2 Camac Programming under Vortex	25
	7.3 Data Collection	26
8	VERIFICATION OF DATA-SPOOLING CAPACITY	27
	8.1 Data Transfer Rate	27
	Conclusions	28
	8.2 The Effects of Conflict Between Programs for Access to the Disk	29
	8.3 Effects of Conflicting Priorities	29
	8.4 Comparison of Actual with Theoretical Timing Calculations	30
	8.5 Summary	30

9 VERIFICATION OF THE TIME-SHARING CAPABILITIES OF THE SYSTEM 31
9.1 Introduction 31
9.2 Terminology 31
9.3 Experience with the System 31
9.4 Experiments for Timing Interrupt-Driven Transfers of Control 33
9.4.1 Introduction and Background to the Timing Experiments 33
9.4.2 The Data Acquisition Process 33
Block Transfer 34
Normal Mode 36
9.4.3 Comparison of Camac Timer and Varian Timer Rates 37
9.4.4 The First Series of Experiments : Normal Mode, Command Timing 38
Method and Measurements 38
Results 42
9.4.5 The Second Series of Experiments : Normal Mode, Continuous Timing 45
Method and Measurements 45
Constraints Necessary for Data Integrity 47
Results with One Real-Time Program 48
Results with Two Real-Time Programs 51
The First Experiment 51
Comparison with Theoretical Upper Bounds 51
The Second Experiment 52
Conclusions 53
Program Design Guidelines 54
9.4.6 Experiments with Block Transfer Using Continuous Timing 55
Data Conversion Times 56
9.4.7 Timing of R.T.E. Functions 58
9.4.8 Further Results of the Experiments 60
9.4.9 Conclusions and Summary 61

10 FURTHER ASPECTS OF THE FEASIBILITY STUDY 62
10.1 Simultaneous Use of Several Laboratories 62
10.2 Amount of Main Memory Used 63
10.3 Printing Requirements 66

11 FINAL CONCLUSIONS AND RECOMMENDATION 67

APPENDICES

A1 Computer Configuration 69

A2 Common Symbols and Macros 73

A3 Example of a Program Segment for Initializing the BIC and CAMAC in a Block Transfer Operation 75

A4 Interrupt Assignments 76

A5 Camac Modules and Interface 78

A6 NIPR Programming Conventions for the Camac Equipment 81

A7 Interrupt Handling 85

A8 Glossary of Data Processing Terminology Used in the Report 88

List of Tables

Table		Page
2.1	Estimates of Time to Complete a Project by Division	7
2.2	Estimates of Total Time Requirements	8
5.1	Resource Requirements	14
5.2	Experiment Combinations	15
8.1	Disk and Tape Rates	28
9.1	Comparison of Camac and Varian Timer Rates	37
9.2	Statistics from Experiments Using Normal Mode and Timing on Command	42
9.3	Statistics from Experiments Using Normal Mode and Continuous Timing	48
9.4	Statistics from Experiments Using Block Transfer and Continuous Timing	56
9.5	Time to RESUME a Program	59
10.1	Input/Output Buffering Memory Requirements	64
10.2	Memory Requirements for Program Development Facilities	65

List of Figures

Figure		Page
6.1	Camac Organisation	20
6.2	Timing Chart for Block Transfer of Data	23
7.1	Data Collection System	26
9.1	Data Sampling Timing Chart for Block Transfer	35
9.2	Timing Chart for Data Sampling under Program Control	36
9.3	Timing Chart for First Method Using Timing on Command	40
9.3a	Timing Chart Showing Entrainment of One Program by Another	41
9.4	Time-Sharing a Real-Time Program with Non-Real-Time Programs	48a
	Time-Sharing two Real-Time Programs : Continuous Timing, Normal Mode	
9.5a	EEG Program - 1 to 10 Channels	50a
9.5b	ECG Program - 1 Channel	50b
9.5c	Comparison with Theoretical Upper Bounds	50c
9.6a	EEG Program - 1 to 6 Channels; ECG Program - 5 Channels	50d
9.6b	Comparison with Theoretical Upper Bounds	50e
9.7a	Sampling at Different Rates	51a
9.7b	Comparison with Theoretical Upper Bounds	51b
9.8	Data Conversion Times with Block Transfer	56a

Chapter 0 Introduction

In the early 1970's it became apparent that several divisions of the NIPR required a mini-computer for their research. In analysing their needs it was found that the various functions the computer would be called upon to perform fell into one of three broad classifications : data collection, data analysis, and experimental control. The most urgent requirement and the most demanding in terms of computer capability was the monitoring and sampling of various physiological functions such as the electroencephalograph. The data analysis the mini-computer would be expected to perform would be simple and undemanding in terms of scheduling; most analyses would be performed on a large digital computer. Experimental control, offering the new possibility of presenting stimuli to subjects in varying combinations of type, duration, intensity, at regular or random intervals etc. in a rigidly controlled and repeatable manner was also foreseen as a requirement in the near future.

The request for the services of a computer was identified to come from three divisions. The Neuropsychology Division required a computer urgently and expected to use it continuously. The Personality and Temperament Division, and the Psychometric Division had less urgent needs and expected to make use of the computer intermittently. The choice was seen to lie between the purchase of one dedicated machine to be followed later by a second general purpose mini-computer or a single larger machine capable of multiprogramming. The first alternative would prove to be more expensive in the long run, and would not provide the other divisions with adequate opportunities for development in the immediate future. Another consideration was that the acquisition of funds for the purchase of the second machine a year or so after the first might prove difficult. On the other hand there was considerable controversy over the feasibility of multi-programming on mini-computers. It was decided, therefore, to purchase a single machine and reach agreement with the supplier whereby the additional equipment required for multi-programming would be loaned for a period during which a feasibility study would be carried out.

The Structure of the Report

This report describes the circumstances leading up to the feasibility study, the equipment and software used, the results of the experiments performed and conclusions that can be drawn. Details of the functions which the various projects of the NIPR require of the mini-computer are presented in chapter 1. Chapter 2 indicates the expected usage, time-wise, of the mini-computer by the various projects. Chapter 3 examines the pros and cons of time-sharing on a mini-computer and chapter 4 discusses the proposal to undertake a feasibility study to resolve the question. The basic configuration and the additional equipment required for the feasibility study is given in chapter 5, while chapters 6 and 7 give a detailed description of the relevant hardware and software. The verification of the feasibility study is discussed in terms of the data handling capacity of the system in chapter 8, the practicality of time-sharing real-time programs in chapter 9 and the results of experience with the system in chapter 10. The recommendations resulting from the study are presented in chapter 11.

Chapter 1 The Purposes For Which the NIPR Required a Mini-Computer

1.1 Requirements of the Neuropsychology Division

The principal work of the division involves monitoring neurophysiological functioning by amplifying and recording the electrical potentials generated by the nervous system. The electrical signals thus derived were (and to some extent, still are) evaluated by subjectively scanning the graphic output of the electroencephalograph or by making a multitude of measurements by hand. Some years ago a small special-purpose device, a Computer of Average Transients (CAT), was acquired by the Division. This device provided new analysis capabilities and relieved much of the manual work, but is limited in many respects when compared to a modern mini-computer. The increasing volume of work, the growing backlog of only partially analysed data and the need to apply sophisticated modern procedures created the need for computerised acquisition and conversion of data to a form suitable for further processing and analysis and off-line storage. The analyses required included digital filtering, period-amplitude and Fourier analyses of the EEG, and the detection and measurement of characteristics of the evoked response.

The following is a précis of the requirements of the Division as set down in a memorandum by Dr. R.D. Griesel dated 11th January 1972.

The techniques that a computer facility would extend or make possible included

(a) Evoked Response. Studies of brain potentials evoked in response to visual, auditory and/or somatic stimuli would particularly benefit from a computer controlled facility. Current experiments used manual methods of control, or special but limited hardware. The computer would also allow more accurate and detailed recording of the evoked response than the currently used special purpose Computer of Average Transients (CAT). Projects that would use this facility are 70/11, 64/42 and 63/3.

(b) Contingent Negative Variation. A facility which the computer makes possible is the presentation of conditional stimuli, according to a programmable variable reinforcement schedule, for studying the amount of uncertainty people will tolerate for allowing the development of the electroencephalographic contingent negative variation. Projects that would use this facility are 64/3 and 70/11 as well as military and road safety projects.

(c) Complex Stimulation. The computer would make feasible the presentation of complex visual stimuli on the visual display unit, or tachistoscopic presentation of visual stimuli, at automatically controlled programmable rates of information presentation. Projects that would use this facility are 70/11, 64/42, 63/3 and 64/3 as well as military and road safety projects.

(d) Controlled Stimulation. The computer would be used to present stimuli only when certain conditions exist in the central nervous system. This facility would be incorporated in the programs providing the above functions.

Projects in the Neuropsychology Division Requiring the Use of a Mini-Computer

Project 64/42 : Study of visual and auditory perception in different racial groups. The research aims at developing adequate tests of visual and auditory functioning to allow a comparative study of a White and Bantu groups.

Project 70/11 : EEG and intelligence. The research aims at investigating the validity of measures of central nervous activity (as measured by the electroencephalographic signals evoked by a stimulus) for predicting performance on intelligence tests. The greater objectivity of the psychophysiological measures and their relevance in certain theories regarding intellective functioning would make them the preferred method of assessing intelligence in a way unaffected by the cultural assumptions inherent in traditional intelligence tests.

Project 71/17 : EEG and psychomotor development. This project aims at documenting in terms of both electroencephalographic and psychomotor measures the rate of maturation of the normal Bantu child. The earlier studies of the NIPR concerning the effects of malnutritution on behaviour will be extended to assess the influence of infantile malnutrition on development as assessed in the above-mentionned way.

Project 63/3 : The EEG, behaviour and brain damage. Although one aspect of this study includes the provision of a clinical assessement of the degree of brain damage found in certain cases of behavioural aberration, the project allows the acquisition of normative data and the investigation of the relationship between these electrocerebral measures and behavioural and neurological assessments of the extent of brain damage.

Project 64/3 : Normal behaviour and brain function. On the basis of certain theoretical considerations it is expected that electrocerebral measures, including the EEG measures of speed of reaction to a stimulus, will be predictive of an individual's temperament as assessed psychometrically.

Other Projects : Some projects not forming part of the Division's general research programme but relevant here are the clinical electroencephalographic assessments performed from time to time and a fairly heavy commitment to outside bodies with regard both to similar clinical screening services and research into psychophysiological problems.

1.2 Requirements of the Temperament and Personality Division

The need of the Division for the use of a mini-computer was set out in the third report of the sub-committee for assessing future needs for D.P. equipment in 1971 and in a memorandum by Miss E. Spies dated 25th January 1973. The computer would be used for data recording

of physiological responses, test item presentation and on-line computation. Specifically, in project 64/19, Measurement of Effort, a computer was required to enable reliable measurement and recording of the heart-rate and several variables with which it is associated. The mini-computer would also make it possible to control accurately the stimulation producing various conditions of mental load (usually choice reaction in experimental studies). Preliminary analysis of the data would also be done on the mini-computer.

1.3 Requirements of the Psychometrics Division

The Psychometrics Division set out its proposed immediate and future real-time computing requirements in the third report of the sub-committee for assessing future needs of D.P. equipment at the NIPR in 1973 and in a memorandum by M.A. Coulter dated 25th January 1973. Project 64/7, Rate of Information Processing (RIP), required the computerised presentation of stimuli on a visual display unit to a subject at increasing rates and the computerised scoring of the testee's response. The test measures the speed at which a person can receive, evaluate and act on perceptual information. The test has been implemented and used as part of a selection battery for pupil pilots, but difficulties in administering and manually scoring the test have caused it to be dropped from use.

Future needs of the division for real-time computing facilities concerned with test presentation were envisioned as the following :

(a) the investigation of memory processes,
(b) the temporal integration of visual information,
(c) the attentional and selective aspects of information processing,
(d) testing tailored to individual requirements, and
(e) testing with feedback.

All these projects require a visual display terminal attached to a responsive computer with data storage and timing facilities.

Chapter 2 The Expected Usage of the Mini-Computer

A survey was carried out in early 1972 to assess the expected work load on the mini-computer. The estimates of the expected usage of the mini-computer were derived from a questionnaire that was completed by the divisions that expected to make use of the machine and other members of the NIPR with relevant experience. Although the estimates could only serve as rough guidelines, they were made by experienced people and definite conclusions could be drawn.

The course of an experiment requiring the use of a mini-computer was broken down into six stages and the six participants in the survey were asked for their estimates of the time needed to complete each stage. The results shown in Table 2.1 ranged from two to ten months. The total requirements for the research program of each division is given in Table 2.2. This estimate of two years to work through the current research program of the Institute made no allowance for the use of the machine for other purposes and assumed that there would be no delays in scheduling projects to use the computer. Two years was therefore a minimum estimate and the total time was thought to be more likely to be between three and four years.

Table 2.1 Estimates of Time to Complete a Project by Division

Experimental phase	No. of days for a project		
	EEG	ERGONOMICS	PSYCHOMETRICS
1. Setting up apparatus and getting it to work properly.	60	30	5
2. Compilation and testing programs for controlling the experiment and recording data.	43	5	2
3. Trial runs with live subjects and the complete setup.	4	5	3
4. Production - conducting the experiment	20	8	22
5. Compilation and testing programs for simple analyses.	23	5	1
6. Analyses.	100	8	1
Total number of days	250	61	34
Delays : 5% computer down, 5% apparatus down, 10% other.	50	12	7
TOTAL NUMBER OF DAYS	300	73	41
TOTAL NUMBER OF MONTHS	10	4	2

Table 2.2 Estimates of Total Divisional Dedicated Computer Time Requirements

Psychometrics	5 months
Temperament and Personality	4½ months
Neuropsychology	13 months
Total time for 7 projects	22½ months or approx. 2 years

Note : In estimating divisional dedicated requirements, it was assumed that constructing experimental apparatus and getting it to work would require only intermittent access to the computer and the main part of this would be overlapped with work on other projects and that this would also apply to a lesser extent to the compilation and testing of programs and trial runs with live subjects. Average times were obtained by weighting estimates according to the experience of the estimator.

Subsequent experience has shown that the Ergonomics and Psychometrics experiments required more than twice the time originally estimated. Unfortunately, no comparable information is available for EEG experiments as yet.

Admitting the roughness of the estimates, the conclusion was nevertheless inescapable that there was too much work for a single computer to handle in serial fashion. As the urgent need to process the large volumes of data generated by EEG projects would tend to lock out the development of programs needed for the experiments of the two other divisions, an acute conflict over computer resources could be anticipated. It was therefore necessary to consider the simultaneous multiple use of the computer, or the acquisition of two machines.

Chapter 3 Pros and Cons of Purchasing a Time-Sharing Mini-Computer

The question whether time-sharing on a mini-computer is practical is the subject of some controversy and cannot be answered in general. The answer depends on the desired application, on the available technology and on the current "state of the art". At the time the question was being considered in late 1972, Uttal, an eminent authority, was arguing against the use of small computers in a time-sharing fashion in the laboratory (1).**

The general arguments for time-sharing are that it allows more effective use to be made of the CPU and that it allows one computer to do the work of two or more. Arguments against time-sharing in the laboratory are the following :-

(i) The computer cannot service more than one user requiring rapid response as the first user to be served will lock out all other users.

(ii) Operating system overheads are relatively heavier on mini-computers because of limited hardware and instructions.

(iii) The cost of additional peripheral equipment to support time-sharing may be greater than the cost of purchasing additional computers.

(iv) A time-sharing system requires a large programming effort to implement.

At the time of the decision on a time-sharing system for data acquisition in the NIPR's laboratories there were several manufactures of mini-computers that offered time-sharing facilities as part of the standard software for their machines. Thus the fourth argument fell away.

** The controversy is still very much alive : In October 1973 an entire session of The Third National Conference on the Use of On-line Computers in Psychology held at St. Louis, Missouri was devoted to actual time-sharing and multiprogramming applications.(2) In May 1975, Datamation carried an article by B.K.P. Horn and P.H. Winston arguing that the end of time-sharing is in sight.(3)

Most of the mini-computers on the market had interrupt systems to allow rapid program switching with low software overheads. The actual data collecting routines were expected to be short and simple, so that the time taken to process an interrupt would be very short. The data rates required by the NIPR projects were not high in comparison to typical mini-computer processor speeds - the maximum being 16 channels of data to be sampled every millisecond - and it was estimated that the time to read the data at this rate and transfer it to off-line storage would require about half a millisecond in every millisecond of CPU time. Most projects required much lower sampling rates. Thus timing considerations were not expected to pose any problems and arguments (ii) and (iii) were not expected to apply in the NIPR environment.

The position was therefore that, if it could definitely be established that time-sharing was feasible in the NIPR situation, i.e. that the above arguments were valid, then the deciding factor would be the cost of a computer capable of time-sharing compared with that of two dedicated machines, or, more accurately, the cost/benefit ratio. It is not strictly possible to compare a time-shared system with a non-time-shared system with equivalent experimental control facilities since each system will have facilities that the other does not have. For example, the time-sharing machine will be the larger and will therefore allow program development to take place more rapidly and will allow larger analysis type programs to be run. Bearing this in mind, the costs of two configurations quoted to the NIPR in early 1972 were as follows. One system was a Varian, configured for time-sharing at a cost of R53,350 and the other was a non-time-sharing system costing R45,000, viz. a difference of R8,350. A second, smaller machine with 8k memory, a teletype, a clock, an analogue to digital converter and a casette tape for communication with the larger machine (which would also need a casette tape) would cost R14,700 which is about twice the additional cost of the time-sharing system. It was also known that the first extension to the system would be a display unit for portraying the parameters of EEG data on-line so that the recording apparatus can be kept in step with changing conditions in the experiment. To fit a display unit on the smaller machine would

cost R8700 for the screen and interface whereas these additions would only cost R4500 in the Varian machine. In this case, the difference in cost would be R10,500 in favour of the time-sharing system. Anticipated later needs for other peripherals such as printers, plotters, and additional memory capacity for both machines tips the balance still further in favour of the single machine. On this rough basis, the advantage in cost appears to lie with the time-sharing system.

---

(1) Uttal W.R. Misuse, abuse, overuse and unuse of on-line computer facilities by psychologists.
Behav. Res. Meth. and Instru;, 1972, Vol 4 (2) pp 55-60.

(2) Proceedings published in Behav. Res. Meth. and Instru; 1974, Vol 6 (2).

(3) Horn B.K.P. and Winston P.H. Personal Computers.
Datamation 1975, Vol. 21 (5).

Chapter 4 Proposal to Undertake a Feasibility Study

The proposal to use a mini-computer in time-sharing mode to sample data from two or more experiments simultaneously appeared feasible on theoretical grounds as worked out from program specifications, Camac speeds, magnetic tape and disk data rates and CPU processing times. But, because the performance of a system in practical situations often presents problems unforeseen in theoretical studies, it was decided to undertake a feasibility study before actually purchasing a full system.

Five propositions were put forward to be tested, i.e. that :-

(1) Computer participation in NIPR experiments could be adequately handled from terminals located in laboratories remote from the computer. In the case of EEG experiments, the terminal was to be equipped with a display unit a key board and graphic and hardcopy facilities.
In the case of Personality and Temperament experiments and Psychometrics experiments the terminals were to be equipped with a teletype and a visual display unit with a keyboard respectively.

(2) Two typical NIPR experiments could be run simultaneously.

(3) The minimum amount of core memory required would be 20K or 24K. The VORTEX operating system was expected to take 6K, and the Fortran compiler 8K so that to run a Fortran compilation and one foreground, data gathering program would require more than 16K.

(4) The disk and tape unit could handle the maximum data rates for storage and spooling.

(5) A teleprinter would suffice for the printing that was necessary.

Chapter 5 Hardware and Software used for the Feasibility Study.

5.1 Configuration chosen for the feasibility study.

The precise details of the equipment chosen for the feasibility study are given in appendix A1. The essential features of the system are as follows. The computer is a Varian V73 with hardware multiply/divide, 24K 16 bit words of memory, an operator's teletype, a disk, and a magnetic tape drive. The data logging facilities are catered for by a Borer Camac crate containing an analog to digital converter, a 16 channel multiplexer, two clock/timers, and two I/O registers. The remote terminals are equipped with : (1) a Tektronix display unit with graphic and hardcopy facilities, (2) a teletype, and (3) an Infoton teletype compatible display screen. Cables run to each terminal for connecting experimental apparatus directly to the Camac units. A small line printer was added to the system later.

5.2 Computing Facilities.

Typical instruction times in core memory range from 660 nanoseconds for loading a register to 6641 nanoseconds for a divide instruction. The computer and Camac have extensive interrupt facilities and the VORTEX disk-based operating system which is being used to run all programs on the machine, supports multi-programming. These features provide for rapid task switching when an interrupt is received. With this configuration it is possible to run programs for two experiments simultaneously (providing the programs have no conflicting peripheral, Camac or memory requirements).

Standard software which includes Fortran and an assembler with high level macro support for peripheral I/O and file handling facilities, allows the computer to support all proposed program development. However, the computer does not have floating point hardware, and fixed-point arithmetic is performed on 16 bit words, so only the simpler forms of data analysis are possible.

5.3 Camac Capabilities

The Camac system interfaces the computer with the experiment : the Camac crate contains modules for reading analog signals, reading and writing digital data, controlling sampling times and connecting outside (experimental) events to the computer's interrupt system.

5.4 Resource Allocation

The following tables show the hardware requirements of different classes of experiments (Table 5.1) and the consequent restraints this imposes on multiple operation (Table 5.2)

Table 5.1 Resource Requirements.

EXPERIMENT	RESOURCE									
	CPU USE DURING INTERRUPT	MAGNETIC DISK (D)	MAGNETIC TAPE (T)	MULTIPLEXER	ADC	I/O REGISTER	I/O REGISTER	CLOCK/TIMER	CLOCK/TIMER	VARIAN INTERVAL TIMER
	Milli-secs									
EEG(1) - DATA SAMPLING	0,5	or T	or D	*	*			*		
EEG(2) - EVOKED RESPONSE	0,5	or T	or D	*	*	*		*		
EEG(3) - CAT DATA TRANSFER	0,05	or T	or D			*				
ECG - DATA SAMPLING	0,3	or T	or D		*			*		*
RIP - DATA GATHERING	0,01	or T	or D				*		*	
TEST PRESENTATION	0,01	*								

Note

(1) An asterisk indicates the module or resource is required for exclusive use of the particular experiment.

(2) In principle, all experiments may log their data to disk or to tape.

(3) The CPU times listed include system overheads, and are estimates that were used in planning the study.

Table 5.2 EXPERIMENT COMBINATIONS.

EEG(1)	EEG(1)					
EEG(2)	C	EEG(2)				
EEG(3)	P	P	EEG(3)			
ECG	C	C	P	ECG		
RIP	P	P	P	P	RIP	
TEST	P	P	P	P	P	TEST

Note :

(1) C indicates a conflict in resource requirements.
P indicates it is possible to run the two experiments together.

(2) The table is derived from resource requirements given in Table 5.1. Some EEG type (1) experiments may require exclusive use of the disk and will therefore conflict with the Test Presentation experiments. Also it would be possible for an ECG experiment to share the ADC via the Multiplexer with EEG type (1) or (2) experiments provided they did not need more than 15 channels.

Chapter 6 Detailed Description of Relevant Hardware and Its Operation

This chapter describes those features of the system that are exploited by the experimental control programs. The instruction set of the Varian 73 is not discussed as it contains all the standard types of instructions (including hardware multiply and divide but no floating point instructions) and no unusual commands. The V73 interrupt system is described in the first part of this chapter, together with the BIC and PIM I/O options. The second part of the chapter describes the Camac system and the Camac modules that are being used by the NIPR.

6.1 Varian 73 Interrupt and I/O Options

The standard I/O instructions transfer one word of data to or from a peripheral device under CPU control, but there is the Buffer Interlace Controller (BIC) option which permits peripherals to transfer blocks of data directly to or from memory at rates up to 382,720 words per second by implementing a cycle-stealing direct memory access technique that allows the transfer of data to proceed in parallel with other processing. Cycle-stealing trap requests inhibit the processing of a stored program for only the memory cycle (i.e. 660 nanoseconds) required to transfer one word of data between memory and the peripheral.

Data transfer by the BIC is established under program control. The status of the BIC is tested and if it is not busy it is initialised. The selected peripheral devices are then initialised and the BIC is supplied with the initial and final addresses of a block in memory for the data. The BIC is then activated, the peripheral started and the BIC assumes control of the transfer of data, thus freeing the CPU for other processing. An example of the use of a BIC is given in the program segment shown in appendix A3.

The Varian 73 includes an interrupt capability by which certain I/O devices (and options such as memory protect), can, on a priority basis, request the computer to execute an instruction or series of instructions independently of the program in progress. Following an interrupt, the computer is directed to a memory address specified by the interrupting device where it fetches and executes the instruction (normally a jump-and-mark instruction) that results in the processing of a service routine. The computer returns to the original program though an appropriate jump instruction at the conclusion of the routine. This method of directly connecting an interrupt to the appropriate service routine eliminates the software overhead encountered in some other machines where the operating system has to periodically test whether a bit has been set in a special register. The requirements for implementing an interrupt handling routine under the VORTEX operating system are discussed in chapter 7.

Standard Varian peripheral controllers are normally not capable of generating an interrupt when they require attention because they cannot provide the necessary memory address. This capability is provided by the Priority Interrupt Module (PIM). A peripheral controller connected to a PIM directs an interrupt request to the PIM, which in turn implements the interrupt. Up to eight interrupt lines can be serviced by one PIM. Interrupt requests are stored and serviced in the order of their priority, which depends on how the connections to the PIM are made from the peripherals. The PIM has an 8-bit mask register which can be used to enable any or all selected eight interrupt lines. Appendix A4 gives the allocation of interrupts as the machine was configured in July 1975.

6.2 CAMAC System

6.2.1 Introduction to the CAMAC Concept

The Camac concept arose out of the need in nuclear physics research to interface many different kinds of transducers, and actuators to digital computers. A number of laboratories in the United States and Europe have established a standard interfacing system known as Camac, an acronym for Computer Aided Measurement and Control. It consists of a number of bins or crates, each of which will accept up to 24 modules containing instrumentation to be interfaced to the computer. These modules will each have one or more of the following functions:

Accepting logical commands from a computer,
Sending status information to a computer,
Accepting numerical data from a computer, and/or
Sending numerical data to a computer.

Camac replaces the great variety of input-output equipment found in various models of computers by a single nonproprietary design standardised both electrically and mechanically to ensure compatibility between modules supplied by different manufacturers.

At the rear of the crate, there is a dataway which provides power to the modules and also links them to the computer. It is designed to minimise the logic needed in a module and also to provide a large repertoire of operations for the module designer. The architecture of the dataway is not patterned on the input/output structure of any make of computer, and it is therefore necessary to interface it to the chosen computer, that is, to resolve the time and logic differences between the dataway and the host input/output system. Although this may seem to be merely a case of shifting the interfacing problem to a different part of the system, there are significant advantages. The interfacing task is performed just once for all the modules. The instruments and the computer are made independent and either may be changed or replaced with no effect on the other. Thus different combinations of modules, each combination representing a particular electronic interface between the computer and experimental apparatus, can be set up with no greater effort than placing the required modules in a crate and plugging them together and to the external

apparatus, and several groups of modules, each handling a different experiment, can operate in parallel. Formerly a different electronic interface would have had to be constructed for each combination of modules. The price which must be paid for this flexibility is the addition of another unit variously called the interface, the branch driver, the Camac controller or the Camac processor.

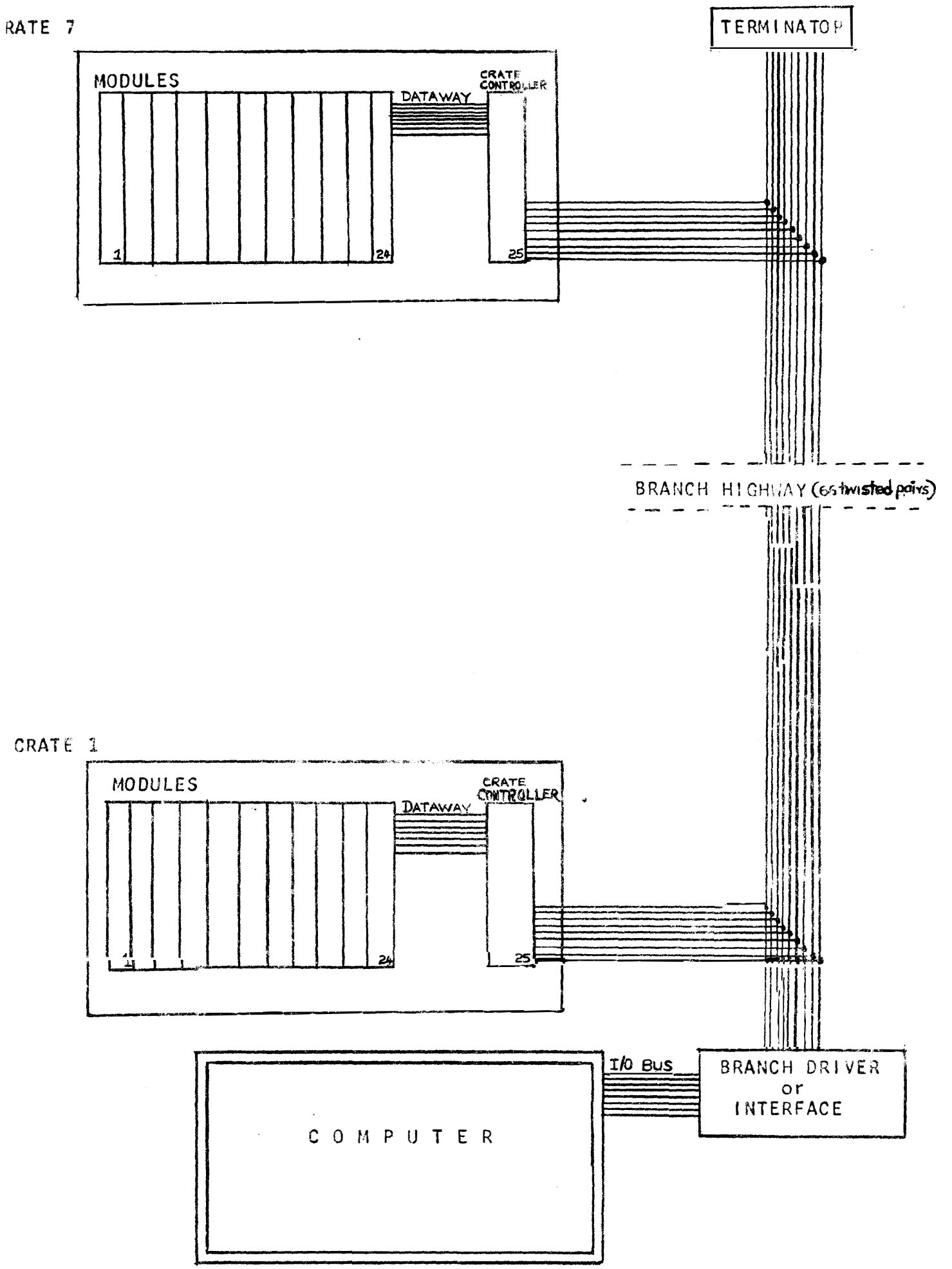
The CAMAC Control Hierachy

In summary then, the Camac system provides the following control hierachy. Each module finds its home in a crate that secures it mechanically and provides its interface to the power supplies and the dataway. The dataway provides the means of interconnection between the modules and the crate-controller within one crate. Multicrate Camac systems are organised as one or more larger structural units called branches in which a branch highway provides the means of interconnection between crate controllers, in up to seven crates, and the branch driver. The branch driver is the interface between the Camac system and the computer.

The CAMAC LAM Concept

When a condition occurs in a module that requires attention by the computer the module generates a signal on its LAM (Look At Me) line. Each station in a crate has a LAM line assigned to it individually. Some of these may be directly connected the computer's interrupt system via a PIM (see the tables in appendix A4) to enable service routines to be invoked quickly and with low software overheads. All of the LAM lines in a crate are connected to a status line associated with the crate, which appears as the BD (Branch Demand) bit in the status word that the computer can read from the Interface or branch driver. When the BD bit is set, the computer may read the status of all 24 LAM lines in a crate and thus determine which modules require attention. Some modules may have more than one condition that can cause a LAM to be set. In this case the condition must be determined by specific test commands.

Figure 6.1 CAMAC ORGANISATION



6.2.2 The CAMAC System at the NIPR

Introduction.

The Camac system purchased by the NIPR** includes modules to read analog signals from up to sixteen channels, to read digital data containing up to 36 bits, and to provide timing facilities to enable sampling of signals at programmable intervals.

The analog to digital converter (ADC) is used to sample an input voltage and convert it to a digital number which can be read by the computer.

The multiplexer is used in conjunction with the ADC to select one of sixteen input channels for conversion.

There are two clock/timers that can be used to initiate sampling operations or programs at programmable rates with a basic time unit of 0,1 milliseconds. Their LAM lines, which are set whenever a timer interval has expired, are connected to the computer's interrupt system thus enabling programs to be invoked at regular intervals.

There are two Input/Output registers which enable digital data to be read by the computer. Their LAM lines are also connected to the computer's interrupt system and are set by external equipment to indicate that there is new data available. This provides the facility for a program to lie dormant in the computer, thus freeing the processor for the use of other programs, until the experimental equipment requires attention.

The interface (branch driver) has an option crucial to the success of the feasibility study. This is the so-called Block Transfer Option which by, effecting series of CAMAC operations independently of the computer, allows data gathering to proceed in parallel with other processing.

** The components of the CAMAC purchased by the NIPR are listed in Table A1.

Descriptions of the facilities and functions of each module may be found in appendix A5, in the manufacturer's manuals, and in a report on Camac programming to be published by the NIPR.

Block Transfer of Data

The special features of the Varian BIC and PIM, the Camac ADC, multiplexer, and clock/timer modules are exploited by the Block Transfer Option of the Camac interface to allow data input to the computer with a minimum of CPU involvement in the data transfer process. The fundamantal function of the Block Transfer Option is to issue a specified Camac command a predetermined number of times. The way it is used in a typical multi-channel data sampling program is as follows :-

The program must initialize

(1) the BIC with the initial and final memory address of where it is to store the incoming data,

(2) the multiplexer with the first channel to be sampled,

(3) the clock/timer with the sample interval, and

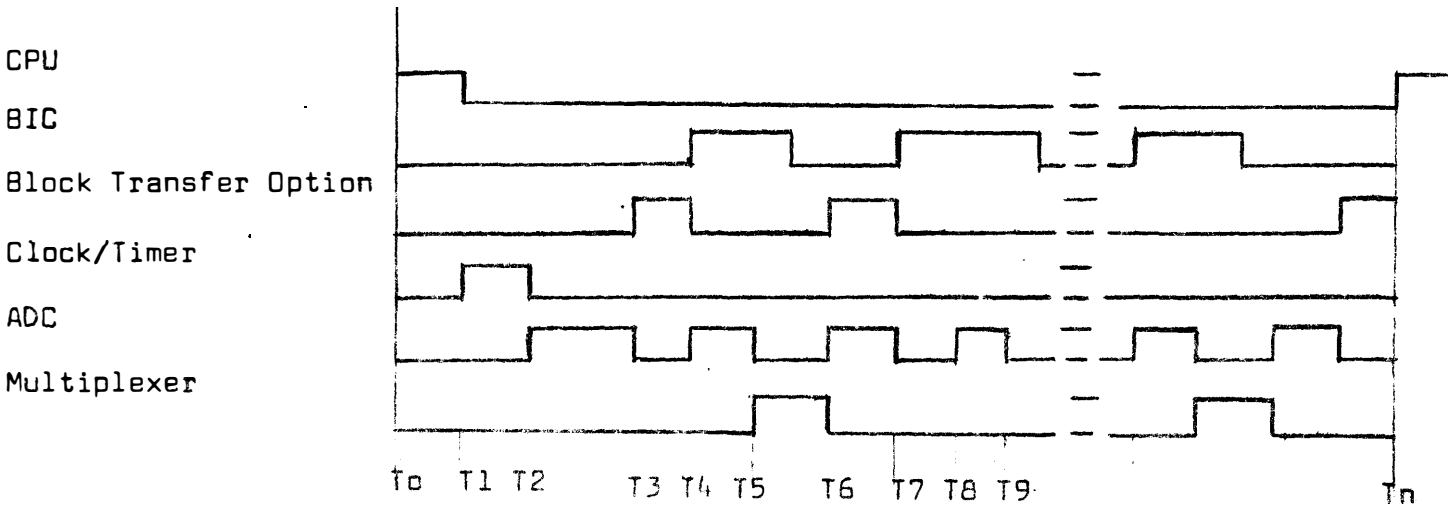
(4) the Block Transfer Option with the number of channels to be sampled and the Camac command to read data from the ADC.

The program then activates the block transfer and relinquishes control of the computer so that the CPU is free to be used by another program while the data transfer takes place in the following manner :

At the end of the sampling interval, the clock/timer generates a signal which actually begins the data transfer process. The ADC use this pulse to initiate the sampling of the signal on the first multiplexer channel, and its conversion to a digital value. At the end of the conversion, the ADC generates a pulse which signals the Block Transfer Option in the interface to issue a Camac read command to the ADC. The value converted by the ADC is passed to the BIC which stores it in the computer's memory without interrupting other processing. The read command also causes the ADC to pulse the multiplexer to cause it to switch to the next channel. When the switch has been completed the multiplexer pulses the ADC (as the timer did) causing it to begin a new sample and conversion. The cycle continues until the count which was set in the Block Transfer Option has been reached when an interrupt to the computer is generated. The interrupt processing routine initialises the BIC and Block Transfer for the next sample and processes the data in the required way.

The way this Block Transfer of data frees the CPU for processing other programs during the data acquisition cycle is illustrated in the timing chart in Figure 6.2.

Figure 6.2 Timing Chart for Block Transfer of Data



From T0 T1 the CPU is initialising the data transfer function.
From T1 to Tn, when the last data word has been read, the CPU is free.
From T1 to T2 the clock is timing the sample interval.
T2 is the end of the sample interval and the start of the data transfer.
From T2 to T3 the ADC is converting the signal on the first data channel.
From T3 to T4 the Block Transfer is issuing the first read command.
At T4 the ADC receives the read command, and passes the data to the BIC.
At T5 the ADC pulses the multiplexer.
From T5 to T6 the multiplexer is switching channels.
At T6 the multiplexer pulses the ADC and begins to second cycle.

At Tn the Block Transfer Function signals the CPU that the required number of data values have been read.

Chapter 7 Relevant Software

7.1 The VORTEX Operating System.

The VORTEX operating system of the Varian 73 computer provides for multiprogramming for several foreground programs (i.e. those which reside simultaneously in memory and are processed according to priority) and one background program which has the lowest priority and is liable to be moved from memory to disk if foreground programs require additional space. Jobs that are not time-dependant are run in the background. The system also provides device-independent I/O facilities. Real-time processing is implemented by hardware interrupt controls and software task scheduling, of foreground programs. Tasks may be scheduled for execution by operator requests, other tasks, device interrupts or the completion of time intervals. Background processing operations, such as Fortran compilations or DAS MR assemblies, are under control of the job control processor, itself a VORTEX background task, and which is run whenever there is sufficient memory and no foreground program is waiting to run. Transfer of control between programs is effected by interrupts or by scheduling.

All tasks are scheduled, activated, and executed by the real-time executive (RTE) component of the operating system on a priority basis. The services that the RTE provides include :

Scheduling a task.
Suspending a task.
Resuming execution of a suspended task.
Delaying the requesting task for a specified time.
Aborting a task.

The VORTEX Input/Output Control (IOC) component processes all requests for I/O to be performed on peripheral devices thereby providing a common, device-independent I/O system.

The VORTEX system is generated from a collection of manufacturer supplied programs on magnetic tape thus tailoring the operating system, to the specific requirements of each user installation. User written programs, such as interrupt handlers, can be incorporated into the operating system during the system generation process. Each time new facilities such as new peripherals or new interrupt processing routines are required the system must be regenerated.

7.2 CAMAC Programming under VORTEX.

In a multiprogramming environment two problems arise in real-time applications using a system such as the Camac.

Firstly, programs must ensure that they do not interfere with other programs by simultaneously using non-shareable resources. Since VORTEX does not provide the means of allocating resources to a particular task and allowing tasks to queue for allocation is only found in operating systems on larger machines, user programs must provide these functions for themselves. To provide these facilities on the NIPR machine, a set of conventions has been drawn up and is described in appendix A6 and in a report on Camac programming to be published by the NIPR. Secondly, the handling of interrupts must be transparent to all tasks running under VORTEX (i.e. the tasks must not have to take account of interrupts for other tasks) and it must be done in such a way as not to delay or interfere with time-critical tasks. VORTEX provides for user written interrupt handlers to be incorporated into the operating system at system generation time, but, although this eases the problem of ensuring transparancy, the software overheads in scheduling the task were estimated to be too great. However, writing a non-standard interrupt handling program turned out to be a relatively simple problem. The actual details and an example program are given in appendix A7.

In summing, the three possible ways of handling the Camac units under VORTEX are :

(1) Incorporating a driver into the operating system - This is too costly in overheads.

(2) Incorporating a user-written interrupt handler routine - This is also too costly in overheads.

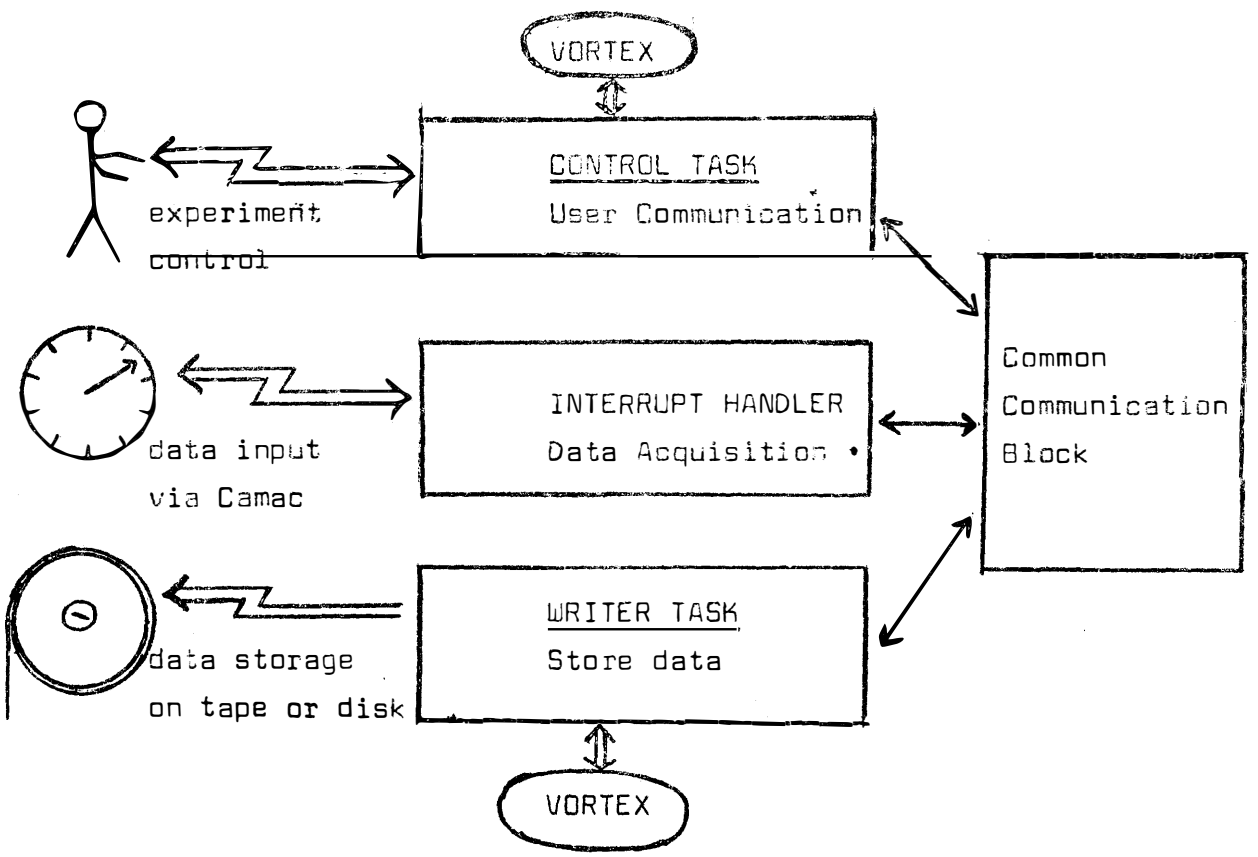
(3) Writing a non-standard interrupt handler program invisible to and outside the VORTEX system.

The third possibility was the one chosen for implementation at the NIPR.

7.3 Data Collection

The primary purpose of the Camac system is data acquisition and so the central function of the supporting software is that of data acquisition and storage. In each data acquisition program this is accomplished by 3 independent, cooperating processes. The first and main process, running as a VORTEX task, communicates with the user and controls the data acquisition process and the data storage process by providing blocks of storage in memory and by allowing the user to specify certain parameters. The data acquisition process is implemented as an interrupt routine that is invoked whenever the appropriate interrupt occurs; e.g. expiration of the timer interval or completion of a block transfer of data. This task runs invisibly to the VORTEX system and to all tasks running under VORTEX. The third process, called the writer, runs as a standard VORTEX task so that it can use the standard operating system I/O facilities for dumping data to disk or magnetic tape. This organisation is depicted in figure 7.1 below.

Figure 7.1 Data Collection System



Chapter 8 The Verification of Data Spooling Capacity

8.1 Data Transfer Rate

The rate at which data can be transferred to disk or magnetic tape is crucial to the success of the feasibility study. The first step, therefore, was to measure the actual maximum data rates that could be sustained by the disk and the tape as opposed to the theoretical values upon which the initial estimates of feasibility had been based. Account also had to be taken of the effect other programs could have on the data transfer rate.

To appreciate the results of the experiments which are summarised in table 8.1 below, some understanding of the way in which data transfer takes place is necessary. Data transfers to disk, to tape and from the Camac take place under control of the BIC option of the Varian 73. In chapter 6 it was explained that when the BIC transfers a word of data to or from memory it inhibits processing of the CPU for one memory cycle. In table 8.1 the maximum effective rate at which data can be written to the disk is shown to be 35,95 words per millisecond. At this rate an average of one in 42,146 memory cycles are used by the BIC to transfer a word out of memory. For the magnetic tape, the maximum rate of data transfer is 9,30 words per millisecond, which corresponds to one in 162,920 memory cycles. In the worst possible case, with data being written out both to disk and to tape at maximum rates and being read at 16 words per millisecond from the Camac system under Block Transfer, the total data rate of 61,25 words per millisecond is achieved which requires one in 24,73 memory cycles. Another program running simultaneously with the data transfers would therefore experience a drop in CPU speed of only 4,04%.

Conclusions : The maximum data rate that the disk will support, 35 words per millisecond, is well above the required maximum data sampling rate of 16 words per millisecond. The tape can support a maximum data sampling rate of 9 words per millisecond. At this rate, however, it requires all of available memory for buffers. With a reasonable buffer size of 1200 words the tape has a maximum data rate of 7 words per millisecond. This is well above the maximum rate expected from experiments other than EEG experiments. A buffer of 2000 would allow the data from an EEG experiment using 8 channels to be written to tape.

Table 8.1 Disk and Tape Data Rates

Rate at which data is written to : (in words per millisecond)	Buffer Size - Number of words			
	120	1200	2400	5760
DISK - ALONE	2,93	18,4	20,64	35,95
DISK - WITH COPY PROGRAM	1,18	9,31	15,60	25,65
TAPE - ALONE	2,13	7,32	8,20	9,30
TAPE - WITH COPY PROGRAM	2,13	7,32	8,20	9,29

Note : Data transfers to the disk in multiples of disk sectors which are blocks of 120 words so that, for efficient use of the disk, data should be in records that are multiples of 120 words. For ease in comparison between disk and tape timings, the buffer sizes for the tape were the same as for the disk.

The copy program simulates other user program activity requiring the disk.

8.2 The Effects of Conflict Between Programs for Access to the Disk

When data is transfered to disk, the disk read/write head is positioned to the correct track and it then waits until the correct sector appears beneath it before beginning the data transfer into sequential sectors. There are therefore two overheads and two potential bottlenecks :

(1) The seek to the correct track, and
(2) The search for the correct sector.

The second factor is inherent in rotating memories and is not affected the way the disk is used. However, the disk drive has only one head to serve both disks, thus, if more than one disk file is being accessed simultaneously, there will be competition for use of the head and in the worst case, the head will be moving continually from one file to the other. As a result, if, in a real-time experiment in which data is being logged to disk, other programs are also to access the disk, then the time for the read/write head to locate the data acquisition program's file and the data transfer time must be less than the rate at which data is being read by the Camac system.

The effect of competition for the disk head was measured by running a file copy program, which transferred a file from one part of the disk to another, simultaneously with a test program which wrote to the disk at maximum speed. The experiment was not the worst possible case, but represented a situation that would be commonly encountered in practice. The test program wrote across the entire disk, but the copying program used files located towards the middle of the disk surface rather than at either extreme.

8.3 Effects of Conflicting Priorities

All data transfer is initiated by software and it was here that another possible bottleneck was identified. The interrupts for the disk have higher priorities than those for the tape and it appeared likely that when disk interrupts were being generated at a maximum rate that the processing of these interrupts and the scheduling of new disk data transfers could possibly delay the scheduling of data transfers to the tape. This effect was measured by copying one file to itself in blocks of 120 words, or 1 disk sector, while writing to tape at the maximum rate with a program of higher priority.

Conclusions The results in table 8.1 above show that the effect of heavy disk usage on the tape data rate is negligible.

8.4 Comparison of Actual with Theoretical Timing Calculations

The actual times measured for writing to the disk (in stand alone mode) are a little less than those calculated by adding the maximum access time to the product of the data transfer rate and the buffer size. The tape writing times are a little longer than the calculated times, and the measured start-up time at 130 milliseconds is significantly longer than the theoretical 18 milliseconds.

8.5 Summary

To summarise, the data handling capacities of the disk and the magnetic tape units are more than adequate for the maximum requirements of the experiments presently planned at the NIPR providing adequate buffers are used.

Chapter 9 Verification of the Time-Sharing Capabilities of the System

9.1 Introduction

Experience with the system and experimental measurements have shown that time-sharing a mini-computer in the NIPR environment is feasible. Two real-time data acquisition programs have been written and used in live psychological experiments, simultaneously with program development at two other terminals with no interference of real-time programs by non-real-time programs. The only degradation experienced by the non-real-time programs was that imposed by the limitations of main memory space. Timing measurements were made with programs run both alone and in contrived bottlenecks and, with two caveats, have proved that the system provides adequate time-sharing facilities for the NIPR.

9.2 Terminology

Terms which may require some explanation and are used throughout the report are defined in the glossary in appendix A8. Specifically in this chapter from section 9.3 onwards the term PROJECT is used to refer to a "live", psychological experiment where the computer was used as a tool, and EXPERIMENT refers to the case where the computing system itself was the subject of an experiment in the feasibility study. Also, NORMAL MODE is used to describe the mode of data acquisition from the Camac unit using an explicit series of Camac I/O instructions in place of Block Transfer.

9.3 Experience with the System

Two programs were used in live projects for real-time data acquisition :

(a) A program for the Personality and Temperament Division sampled one channel of data from an electrocardiogram at 1 millisecond intervals. Its main memory requirements totalled 8k with 1k data beffers. This program was run continuously for about 10 weeks from April to June 1975 and thereafter once or twice weekly.

(b) A program, for the Neuropsychology Division, sampled one channel of electro-encephalographic data at 1 millisecond intervals and required 5k of main memory with buffers of 2k. This program was in continuous use over a period of two and a half weeks.

During the time that a real-time data acquisition program was running, normal program development, i.e. file editing, assemblies, testing in the background, etc., continued at one and sometimes two other terminals. There was no instance of a real-time program being affected by a non-real-time program during this period. Non-real-time programs experienced some degradation because of the small amount of main memory available for their use. This was particularly acute when the ECG program was running because there was then insufficient room for the DASMR assembler. Neither the EEG nor ECG programs left enough space for the Fortran compiler.

The reason the ECG program is 3k larger than the EEG program despite substantially smaller buffers is that it had one subroutine in Fortran which required system supporting subroutines occupying 3-4k. Further, these subroutines are not reentrant. This points to a limitation, namely, time-sharing programs should not be written in Fortran owing to its large demands on main memory or, alternatively, if the ease of programming provided by Fortran is important, then 24k of memory is inadequate and at least 32k is necessary.

The EEG and ECG programs could not be used simultaneously in a live experiment because both require the ADC. This could be remedied by the purchase of a second ADC. This illustrates an important advantage of the Camac system, i.e. additional modules for multiplexing data acquisition can be installed at little extra expense and with no additional interfacing or control requirements.

The most important conclusion from these operations is that the hardware and software are reliable and secure enough for development to proceed in parallell with real-time experiments. The provision of an interactive psychological test-presentation facility has involved a large amount of program development, all of which was carried out in parallel with either the real-time projects or other development. Development would have been considerably delayed if this work had had to wait its turn in the job queue.

9.4 Experiments for Timing Interrupt-Driven Transfers of Control

9.4.1 Introduction and Background to the Timing Experiments

Experiments were also carried out to extend the experience provided by actual processing. Measurements of timing parameters were made for both modes of real-time data acquisition through the Camac system, and also for some of the Real Time Executive services of the VORTEX operating system. The procedures followed and the measurements made are described in the following sections in enough detail to allow the experiments to be repeated. The non-technical reader may skip to a summary of the results in section 9.4.9.

At the beginning of the experiments several problems were encountered which, in the end, turned out to be more of theoretical than practical importance.

(a) Both the Varian and the Camac Timers have a resolution of 0,1 milliseconds and, as this is the order of the discrimination required in the timing measurement, rounding errors are large. However, definite conclusions could still be drawn from the results.

(b) Timing measurements were made by adding instructions for reading a clock to existing interrrupt processing routines. This modified the behaviour of these routines, but not so as to prevent useful results being obtained.

(c) The critical parameter was the delay in initiating an interrupt servicing routine following an interrupt. There is no way of measuring this time directly by software, but is possible to infer it reasonably accurately from other measurements.

9.4.2 The Data Acquisition Process

To understand the experiments it is necessary to describe the data acquisition process in more detail than in chapter 7.

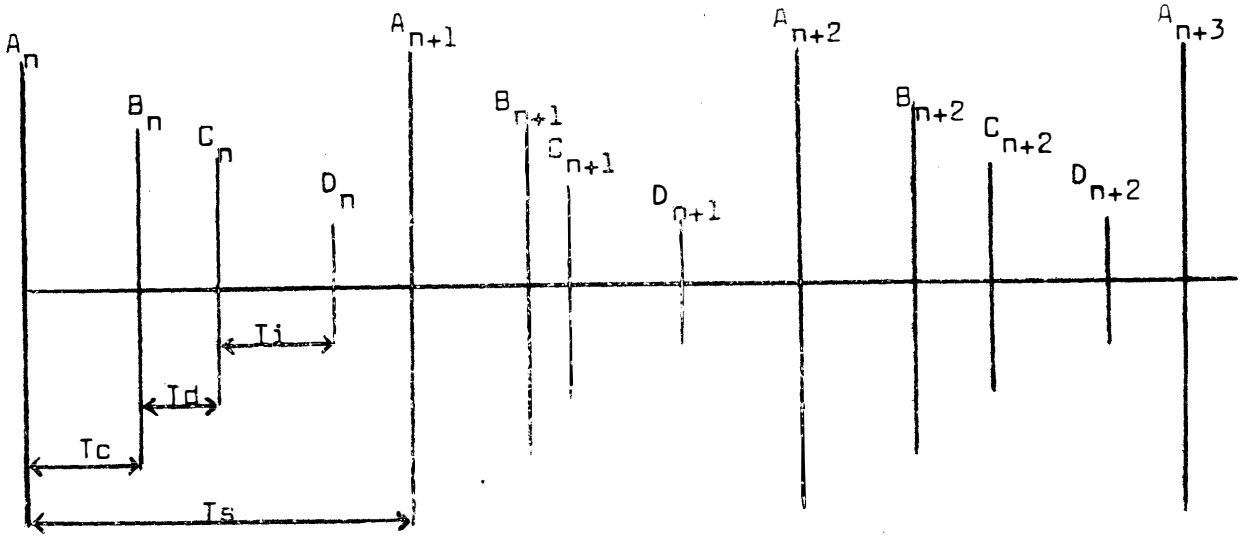
The Camac system is used for data acquisition in one of two modes. When Block transfer is used, data from Camac modules is transferred to the computer under control of a BIC. Otherwise, in Normal Mode, the data is read in with a series of Camac I/O commands issued under program control. A Camac timer times the sampling process in both cases but the timing signal is used differently.

Block Transfer

With the Block Transfer of data, the pulse from the clock starts a process consisting of a number of cycles of acquisition, conversion and transfer to the computer memory, independently of the CPU, and only when all the data has been transferred is an interrupt generated. The interrupt servicing routine reinitializes the BIC and Block Transfer and may do (a small amount of) processing on the data.

If data is not to be lost, it is necessary that the data conversion and transmission time, Tc, plus the delay in servicing the interrupt, Td, plus the interrupt service time, Ti, be less than the sample interval Ts. This is illustrated in figure 9.1 below. Because the digitising is started by the clock and carried out independently of the computer the variation in the sampling interval depends only on the accuracy of the clock. The advantage of Block Transfer is that the CPU has the interval Tp available for processing other programs.

Figure 9.1 Data Sampling Timing Chart for Block Transfer



Ts : Sample interval.

Tc : Data conversion and transmission time for n channels.

Td : Delay in initiating the interrupt servicing routine.

Ti : Interrupt servicing time.

A_n : Clock Pulse; commencement of data conversion process.

B_n : BIC complete interrupt; end of data transmission.

C_n : Control passed to interrupt servicing routine.

D_n : Control returned to interrupted program.

Notes : Ts is constant within the order of accuracy of the Camac Timer.

Tc does not vary much (see section 9.4.6) and is, for 16 channels, 0,55 milliseconds.

Td is unpredictable for it depends on the length of time other programs hold back interrupts while they carry out non-interruptible operations.

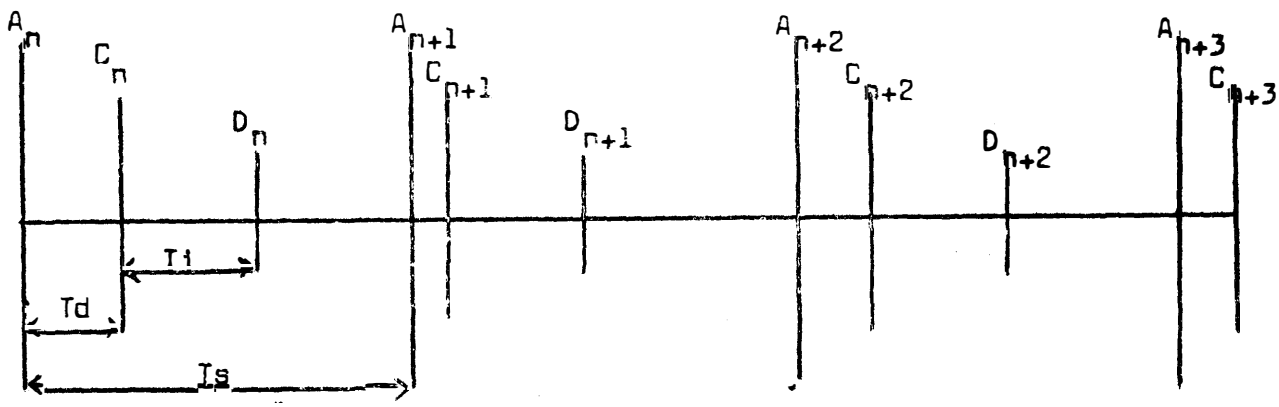
Ti depends on the lengths of the servicing routine and the operational cycle time of the CPU. The latter may vary according to the amount of cycle stealing used for data transfer. Ti represents the time the data acquisition program required exclusive use of the CPU.

Normal Mode

When the Normal Mode of operation is used instead of Block Transfer, the interrupt generated by the clock pulse transfers control to an interrupt servicing routine which initiates the digitizing and transfers data for n cycles to main memory by a sequence of Camac I/O commands. In each cycle, the routine tests whether the next item of data is ready before performing the transfer, and thus the active data transfer is accomplished under control of the CPU.

Figure 9.2 shows the timing chart, which is basically the same as figure 9.1 except that the data conversion time is incorporated in the interrupt servicing time. i.e. the data conversion commences at C_n and not at A_n. As the moment at which sampling commences depends on Td, a variation is introduced in sampling known as "jitter"** which, at high frequencies, corrupts the power spectral density and phase information of sampled waveforms.

Figure 9.2 Timing Chart for Data Sampling under Program Control



Ts : Sample Interval.

Ti : Interrupt servicing time.

Td : Delay in initiating the interrupt servicing routine.

A_n : Clock pulse; commencement of sample interval and interrupt generated.

C_n : Initiation of interrupt service routine, commencement of data sampling.

D_n : End of interrupt service routine; control returned to interrupted program.

** Otnes R.K. and Enochson L. Digital Time Series Analysis. John Wiley and Sons, New York, 1972.

9.4.3. Comparison of Camac Timer and Varian Timer Rates

Although the Camac Timer has a basic pulse rate of 0,1 milliseconds, its clock can be read with a resolution of only 1 millisecond. Thus the Varian Timer, which has a resolution of 0,1 milliseconds had to be used for measuring intervals of time.

As the Camac Timers were used for initiating the sampling processes it became necessary to determine the relative rates of the Camac and Varian Timers. It was found that, while the Camac Timers run at identical rates, they were 0,32 per cent faster than the Varian Timer as shown in table 9.1 below.

Table 9.1 Comparison of Camac and Varian Timer Rates

Camac Timer Interval	Measured by Varian Timer			Ratio of Camac Timer Rate to Varian Timer rates
	Maximum	Minimum	Mean	
Timer Units 4000	Timer Units 3988	Timer Units 3987	Timer Units 3987.3	1 : 0,9968

Notes : (i) 1 timer unit is 0,1 milliseconds

(ii) with a 12 bit scaler, the Camac Timers have a maximum interval of 409,5 milliseconds, hence the choice of 4000 timer units to minimise rounding errors.

** The accuracy of the Camac Timers, which are controlled by a 100k Hz quartz oscillator, was verified with an oscilloscope to be well within the accuracy required for physiological and psychological measurements.

9.4.4 The First Series of Experiments : Measurements of Interrupt Servicing Delays using Normal Mode and Command Timing

Method and Measurements

The first series of experiments were carried out as follows :-

(a) An experiment was initiated by setting the Camac Timer for a 1 millisecond interval and starting it simultaneously with the Varian Timer.

(b) At the beginning of each cycle the Camac Timer was started and the Varian Timer read.
(A_n in figure 9.3)

(c) After the elapse of 1 millisecond, the Camac Timer generated an interrupt.
(B_n in figure 9.3)

(d) When the interrupt servicing routine assumed control it :
(1) Read the Varian Timer.
(C_n in figure 9.3)
(2) Completed whatever processing was necessary.
(3) Restarted the Camac Timer and read the Varian Timer again.
(A_{n+1} in figure 9.3)
(4) Returned control to whatever program was interrupted.

Since the Camac Timer was used in the Command Mode (rather than the Continuous or automatic mode) the technique is referred to as "timing on command".

The recorded time gave :-

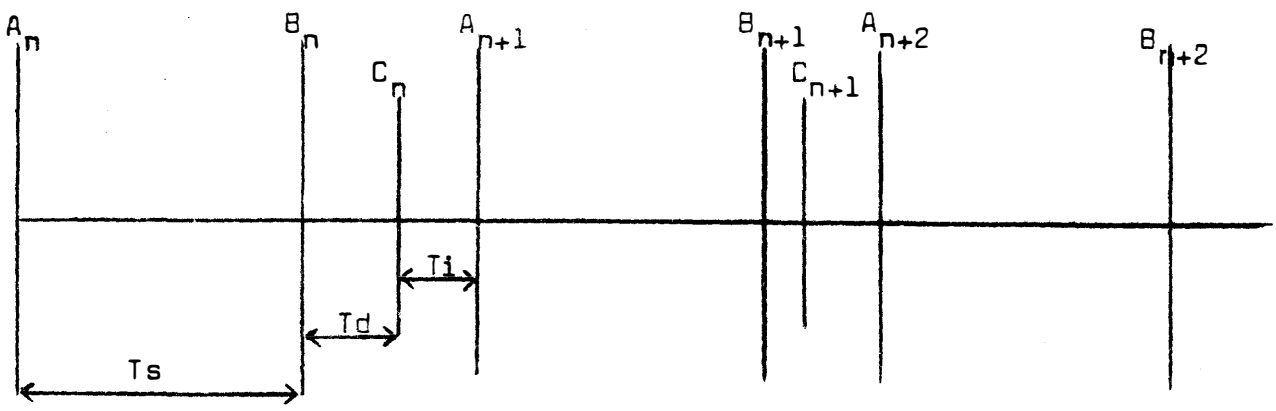
(1) The processing time $T_i = A_{n+1} - C_n$, and

(2) The timed interval, Ts plus the delay time, Td. i.e.
$Ts + Td = C_n - A_n$.

The times required for starting the Camac Timer and reading the Varian Timer are a few microseconds and are negligible in comparison with the rounding error in the Varian Timer measurements. The Varian Timer runs slightly slower than the Camac Timer and, although this factor can be neglected in calculating the maximum Td experienced because the rounding error is very much larger, it must be taken into account when calculating the mean delay time experienced, when rounding errors are eliminated by averaging.

Since the timed interval, Ts, was set at 1 millisecond, the maximum delay time, Td, is calculated by subtracting 1 from the largest $C_n - A_n$, and the mean delay is calculated by subtracting the Varian measure of the 1 millisecond timed by the Camac Timer i.e. 0,9968, from the mean value of Ts + Td.

Figure 9.3 Timing Chart for the First Method of Measuring Delay Times Using Timing on Command



Ts : Time interval of 1 millisecond.

Td : Delay in initiating interrupt servicing routine.

Ti : Processing time in interrupt servicing routine.

A_n : Start of timed interval, Camac Timer started and Varian Timer read.

B_n : End of timed interval; interrupt generated.

C_n : Interrupt servicing routine initiated; Varian Timer read.

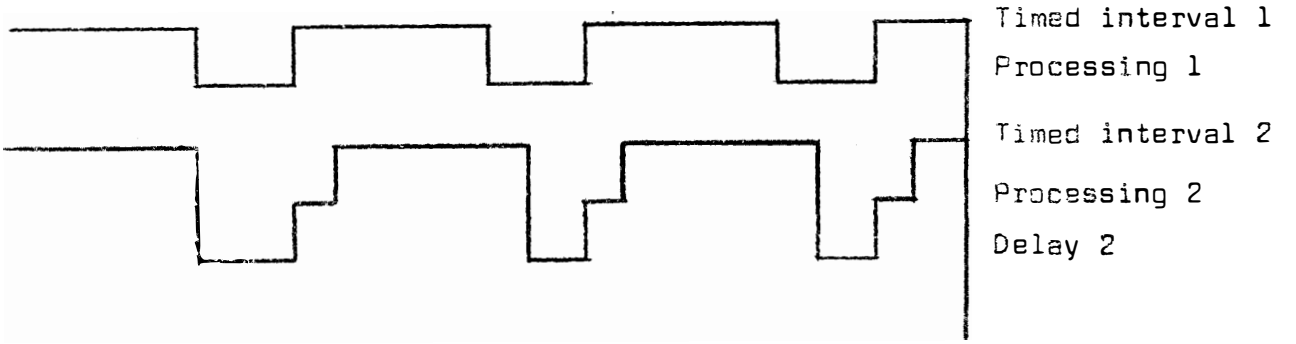
A_{n+1} : End of interrupt servicing routine; Camac Timer started and Varian Timer read. Start of a new cycle.

Note : The difference between using timing on command, illustrated in figure 9.3, and continuous timing, illustrated in figure 9.1 and 9.2, is that, in the former, the delay period and the interrupt servicing period are outside the timed interval while in the latter they are within the timed interval.

A set of typical results is given in table 9.2. The ECG and EEG programs were modified to write out the times Ts + Td and Ti to disk or to tape. The analysis program computed and printed out the maximum, minimum, mean and standard deviation of the times Ts + Td and Ti for successive samples of 1000 intervals. Runs usually about 60 to 90 seconds long, i.e. 60 000 to 90 000 intervals and the results shown are for samples with longest delays. One or two runs were made for each condition in table 9.2.

As both programs used the same time interval a constant delay less than the maximum might be expected due to one program becoming entrained on the other as illustrated in figure 9.3a below. On the other hand, if the interrupts for the ECG program occurred randomly during the processing of the EEG program the former should show an average delay of 0,050 milliseconds (the mean processing time for the EEG program reading one channel of data was approximately 0,100 milliseconds (lines 4,10 and 14 . table 9.2)).

Figure 9.3a Timing Chart Showing Entrainment of One Program by Another



Note how the timed interval of the second program has become displaced a constant time relative to the first.

Results

Table 9.2 Statistics of Selected Samples of 1000 Intervals from Experiments Using Normal Mode and Timing on Command

These samples have the longest delay times in batches of 60 to 90 samples where each sample represents 1000 Intervals (An interval is $A_{n+1} - A_n$ in figure 9.3).

			Td : milliseconds		Ti : milleseconds	
Line	Program	Other	Time to Interrupt		Processing Time	
No.	Timed	Programs	Maximum	Mean	Maximum	Mean
1	CD1	G	0,2	0,002	0,2	0,083
2	CD1	A	0,2	0,003	0,1	0,084
3	CD1	A	1,1	0,003	0,2	0,086
4	ET1	A	0,2	0,003	0,2	0,101
5	CT1	A	0,1	0,000	0,2	0,085
6	CD1	-	0,1	0,003	0,2	0,085
7	CD1	R	0,2	0,005	0,2	0,085
8	ET1	R	0,1	0,002	0,2	0,100
9	ET1	G	0,2	0,002	0,2	0,099
10	CD1	ET1	0,2	0,055	0,1	0,084
11	CD1	ET4	0,4	0,217	0,1	0,084
12	CD1	A,Z	0,2	0,001	0,1	0,086
13	CD1	A,Z	0,2	0,000	0,1	0,084
14	ET1	R,Z	0,2	0,000	0,2	0,102

Program Key :
- C - ECG program.
- 1...4 Number of channels read.
- D - Timing data written to disk.
- T - Timing data written to tape.
- G - General undefined activity in background and foreground.
- A - Assembler.
- R - Disk reading program (disk "exerciser").
- Z - Fix to prevent the disk driver from disabling Camac interrupts.

Line 1 of table 9.2 shows results for the ECG program run with non-real-time programs. The maximum measured delay was 0,2 milliseconds, so that allowing for the worst possible case of rounding errors, the largest possible delay is 0,3 milliseconds. The recorded mean delay time was 0,002 milliseconds.

Line 2 of table 9.2 shows the results for the ECG program run with an assembly with similar results to line 1. However, line 3 shows a delay of 1,1 milliseconds that occurred from 1 to 10 times during an assembly taking approximately 90 seconds. Lines 4 and 5 show when the data was written to tape these unusual delays did not occur. A trace program located the source of the delay in the VORTEX disk driver. A "fix" has been devised that prevents the driver from disabling Camac interrupts and since then no unusual delays have been detected. The fix does not affect the working of other programs. This is illustrated in lines 12 and 13 which show that the ECG program experienced no unusual delays when the fix had been applied and line 14 shows that it did not affect the EEG program.

Line 6 shows results when the ECG program was run alone and line 7 when run with a program that repeatedly read the same record from the disk, the "disk exerciser". The maximum Td is similar to the previous results, but the mean Td has increased to 0,005 milliseconds for the time-shared program. This was expected as the disk driver would at times inhibit interrupts. Lines 12, 13 and 14 show that, when the disk driver was prevented from inhibiting Camac interrupts, the mean delay was practically zero.

Lines 10 and 11 show the results when the ECG program was used with the EEG program, the latter first reading 1 channel of data and then 4 channels, both the maximum and the mean measured delays increased as expected. The maximum measured processing time, Ti of the EEG program was 0,2 milliseconds when reading 1 channel and 0,4 milliseconds reading 4 channels. These were the maximum times the EEG program could delay the ECG program and such was observed.

The actual average delay was 0,057 milliseconds. The additional delay of 0,007 milliseconds is similar to that observed in line 7 and could be due to the locking out of the ECG interrupt by :

(1) VORTEX Real Time Clock Processor.

(2) The disc or magnetic tape drivers.

(3) The locking out of EEG program itself by one of the above.

The variations in the processing times of the two programs (from 0,083 to 0,086 for the ECG program) are small and of no importance in this study, but for completeness the major sources are :-

(1) The effect of other programs. No explanation was found for this and the matter is discussed further in section 9.4.8.

(2) The rate at which the ADC converted the data. The dependence of the data conversion rate of the ADC is discussed in section 9.4.6 where it was important in determining the Block Transfer characteristics.

(3) Errors of measurement.

9.4.5 The Second Series of Experiments : Measuring Interrupt Servicing Delays using Normal Mode and Continuous Timing

Method and Measurements

A second series of experiments was conducted to determine the delay characteristics of two real-time programs run in parallel using the continuous timing mode illustrated in figure 9.2. This is the mode which data acquisition programs use in practice.

(a) An experiment was initiated by setting the Camac Timers for 1 millisecond intervals. The first program to be started would start its Camac Timer and the Varian Timer. The Camac Timer of the second program then was started independently.

For each program :

(b) At the end of a millisecond the Camac Timer generated an interrupt and automatically began timing the next interval. (A_n in figure 9.2)

(c) When the program gained control it :

1) Read the Varian Timer, (C_n in figure 9.2)

2) Completed whatever processing was necessary,

3) Read the Varian Timer again, (D_n in figure 9.2)

4) Returned control to whatever program had been interrupted.

Since the Camac Timers were used in Continuous Mode, the technique is referred to as "Continuous Timing".

The recorded times gave the processing time, $Ti = D_n - C_n$.

The Delay time $Td = C_n - A_n$ was calculated by subtracting the estimated Time of A_n from the measured time of C_n. Assuming that sampling began at A_o and that the sample interval is Ts, then the time A_n is $A_o + n.Ts$. Because Ts was defined by a Camac Timer but elapsed times were measured

with the Varian Timer, the measured value of A_n is given by

$$A_n = A_o + k.\ n.\ Ts \qquad (9.1)$$

where k is a scaling factor defined by

k = (Varian measure of 1 millisecond)/(Camac measure of 1 millisecond)

= 0.9968 (see section 9.4.3)

Thus the delay time, Td is given by

$$Td = C_n - A_n = C_n - A_o - k.n.Ts \qquad (9.2)$$

The floating point arithmetic implied by (9.2) was avoided by the following procedure which used only integer additions and subtractions.

Since Ts was 1 millisecond, i.e. 10 timer units, an accumulator for A_n was incremented by 10 every interrupt. After 32 interrupts, the discrepancy between the accumulator and the value the Varian Timer would have provided had it been able to measure A_n was, by (9.1),

$$32 \times 1{,}0 - 0{,}9968 \times 32 \times 1{,}0 = 0{,}1024 \text{ milliseconds.}$$

Therefore 1 timer unit was subtracted from the accumulator for A_n every 32 interrupts. Expressing this algebraically, the recursive formula used to calculate A_n was

$$A_n = A_{n-1} + Ts - ((n.\ Ts \text{ modulo } 32) + 1)/32) \qquad (9.3)$$

where is the "floor" function which extracts the integer part of its argument.

When Ts is 1 millisecond, the difference between A_n calculated from (9.3) and A_n calculated from (9.1) is

$$n(0{,}1024 - 0{,}1000)/32 = 0{,}000075\ n \text{ milliseconds} \qquad (9.4)$$

i.e. 1 timer unit in 1333 interrupts.

In the experiments using Continuous Timing, a delay of ~0,1 milliseconds was observed once every two or three seconds which implies that the error in using (9.3) instead of (9.1) to calculate A_n is in fact

$$\sim 0{,}00005\ n \text{ milliseconds.} \qquad (9.5)$$

The periodic correction (9.2) is overestimated through using only four decimal places for k. Working backwards from the error (9.5) gives 0,99688 as the value for k.

Constraints Necessary for Maintaining Data Integrity When Time-Sharing Real-Time Programmes

If two real-time progfams are to be run together, the following constraint is sufficient for maintaining the integrity of the data :

$$Td_k + Ti_k \leq Ts_k \qquad (9.6)$$

where k specifies the program,

Td_k is the maximum delay experienced after receiving an interrupt,

Ti_k is the maximum interrupt processing time, and

Ts_k is the sampling interval for the program.

If Td_s is the maximum delay due to factors other than the processing time of the other real-time program then further constraints for the case of two programs are :

$$Td_1 \leq Ti_2 + Td_s \qquad (9.7)$$

and $$Td_2 \leq Ti_1 + Td_s \qquad (9.8)$$

since Ti_2 is the maximum program 2 can delay program 1 and

Ti_1 is the maximum program 1 can delay program 2.

Hence $$Ti_1 \leq Ts_1 - Td_1 \leq Ts_1 - (Ti_2 + Td_s) \qquad (9.9)$$

and correspondingly for the other program. Combining the constraints we have

$$Ti_1 + Ti_2 \leq \tfrac{1}{2}(Ts_1 + Ts_2) - Td_s \qquad (9.10)$$

If Ts_2 and Ts_2 are both 1 millisecond then

$$Ti_1 + Ti_2 \leq 1 - Td_s \qquad (9.11)$$

and $$Ti_1 \leq 1 - Td_s - Ti_2 \qquad (9.12)$$

and $$Ti_2 \leq 1 - Td_s - Ti_1 \qquad (9.13)$$

Results with One Real-Time Program

Table 9.3 shows that this second method of estimating delays, Td, gave substantially the same results as the first. Comparison of line 1 of table 9.2 with line 1 of table 9.3 shows that, for the ECG program reading 1 channel, both methods detected the same maximum delay of 0,2 milliseconds. The average delay measured is 0,048 milliseconds which is merely the average error in estimating the time of an interrupt i.e.

$$(0,0 + 0,1/32 + 0,2/32 + \ldots 3,1/32)/32 = 0,48375 \text{ milliseconds} \quad (9.14)$$

Therefore the true mean delay is zero, which agrees with that measured previously.

The maximum processing time, Ti, is the same for both methods, but the mean processing time (0,021 milliseconds per interrupt) is larger for the second due to the additional time required to calculate the delay times.

Line 2 of table 9.3 shows that the method detected delays caused by the disk driver similar to those reported in line 3 of table 9.2.

Table 9.3 Statistics of Selected Samples of 1000 Intervals from Experiments Using Normal Mode and Continuous Timing

These are the samples with the longest delay times in batches of 80 to 90 samples where each sample represents 1000 Intervals.

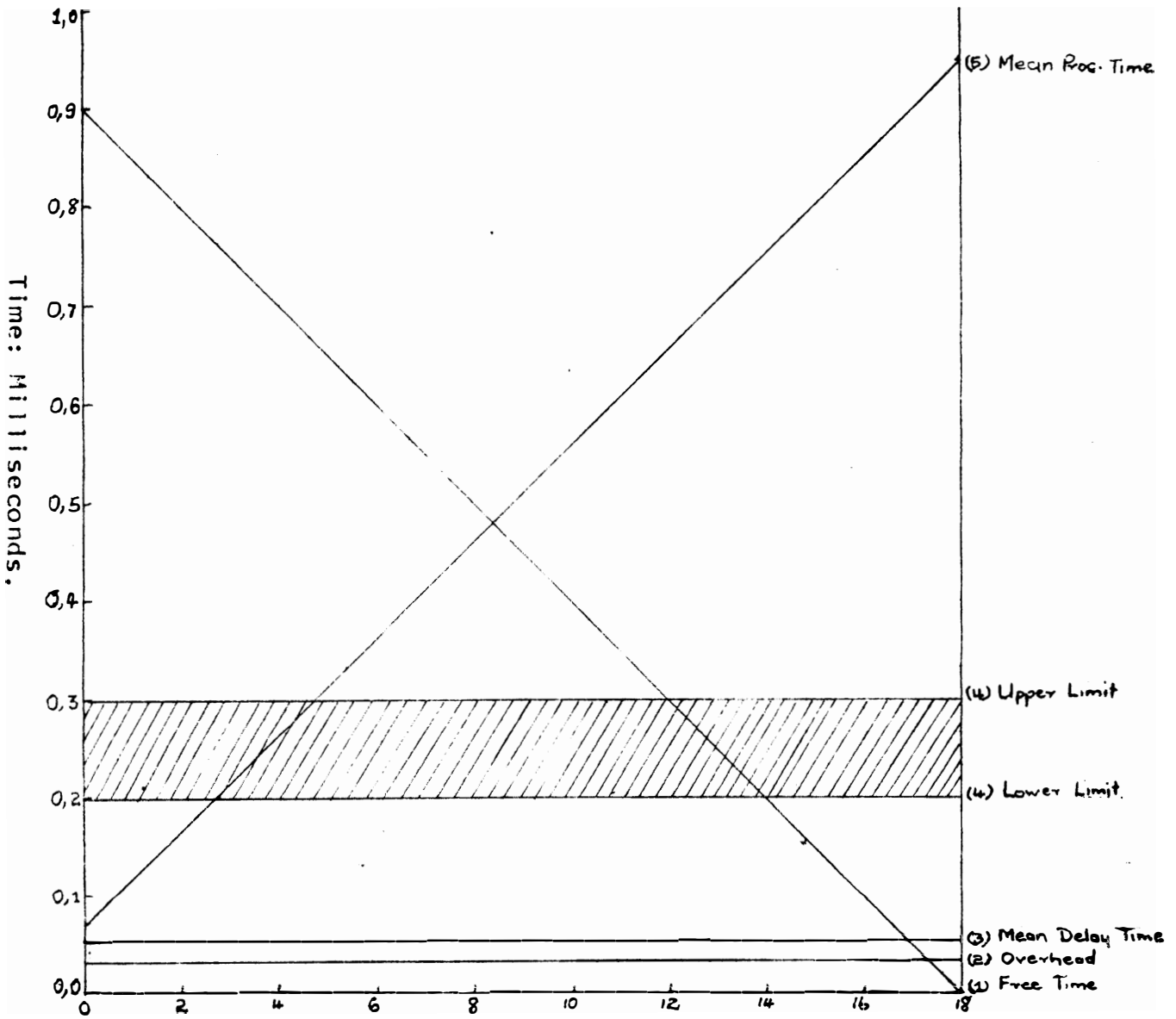
Td : milliseconds Ti : milliseconds

Line No.	Program Timed	Other Programs	Time to Interrupt		Processing Time	
			Maximum	Mean	Maximum	Mean
1	C D1	G	0,2	0,048	0,2	0,104
2	C D1	G	1,1	0,048	0,2	0,104

Program key : C - ECG program
D - Timing data written to disk
1 - 1 channel read
G - General, undefined activity in background and in foreground.

Figure 9.4 TIME-SHARING A REAL-TIME PROGRAM WITH NON-REAL-TIME PROGRAMS: CONTINUOUS TIMING IN NORMAL MODE.

The program timed is the EEG program (ETn) reading n channels of data and writing the timing data to tape.



n = Number of Channels.

NOTES:

(1) Free Time is CPU time available to non-real-time programs.
(2) Overhead is the CPU time used to spool data to disk or to tape plus VORTEX overheads.
(3) Mean Delay is the mean delay in initiating the interrupt servicing routine.
(4) The maximum delay in initiating the interrupt servicing routine. Upper and lower limits show the coarseness of the measurements.
(5) Mean Processing Time is the mean time to read from 0 to 18 channels of data.

Further results of experiments using the Continuous Timing mode are displayed graphically in figures 9.4 to 9.7. The measurements for a series of runs with the EEG program running alone and reading 0,1,2...18 channels are shown in figure 9.4. The mean value of Ti, the time spent in the interrupt handling routine, rose linearly with the number of channels read from an initial "housekeeping" overhead of 0,070 milliseconds. The maximum delay, Td, did not change with the number of channels read; it is shown as a shaded band in the figure to indicate the possible extent of rounding errors. The mean delay time was practically constant; varying from 0,048 milliseconds to 0,051 milliseconds over the entire experiment.

The time available to other, non-real-time programs (labelled Free Time in figure 9.4) shows a linear decrease with the number of channels read from 0,899 milliseconds to 0,005 milliseconds for 18 channels. The measure of free time was obtained from a program that ran at the lowest priority and continuously incremented a counter. At 1 second intervals a high priority program read the counter and reset it to zero again. The ratio of the value of the counter to the number of times it could have been incremented in 1 second of uninterrupted looping gave the fraction of CPU time available to the low priority program. The time not available to the low priority program was used by :

1) The Camac Interrupt handling routine in the EEG program which read n channels of data.

2) The data spooling program which wrote the timing data for the EEG program to tape.

3) VORTEX system programs such as drivers controlling peripherals, and cycle-stealing during the transfer of data.

4) VORTEX system routines such as the Real Time Clock processor which provides timing functions for the system, and the Dispatcher which passes control to the highest priority program that is ready to execute.

The line labelled Overhead shows an estimate of the time spent in VORTEX system programs and routines and in the data spooling program. It is calculated from

$$To = Ts - Ti - Tf \qquad (9.15)$$

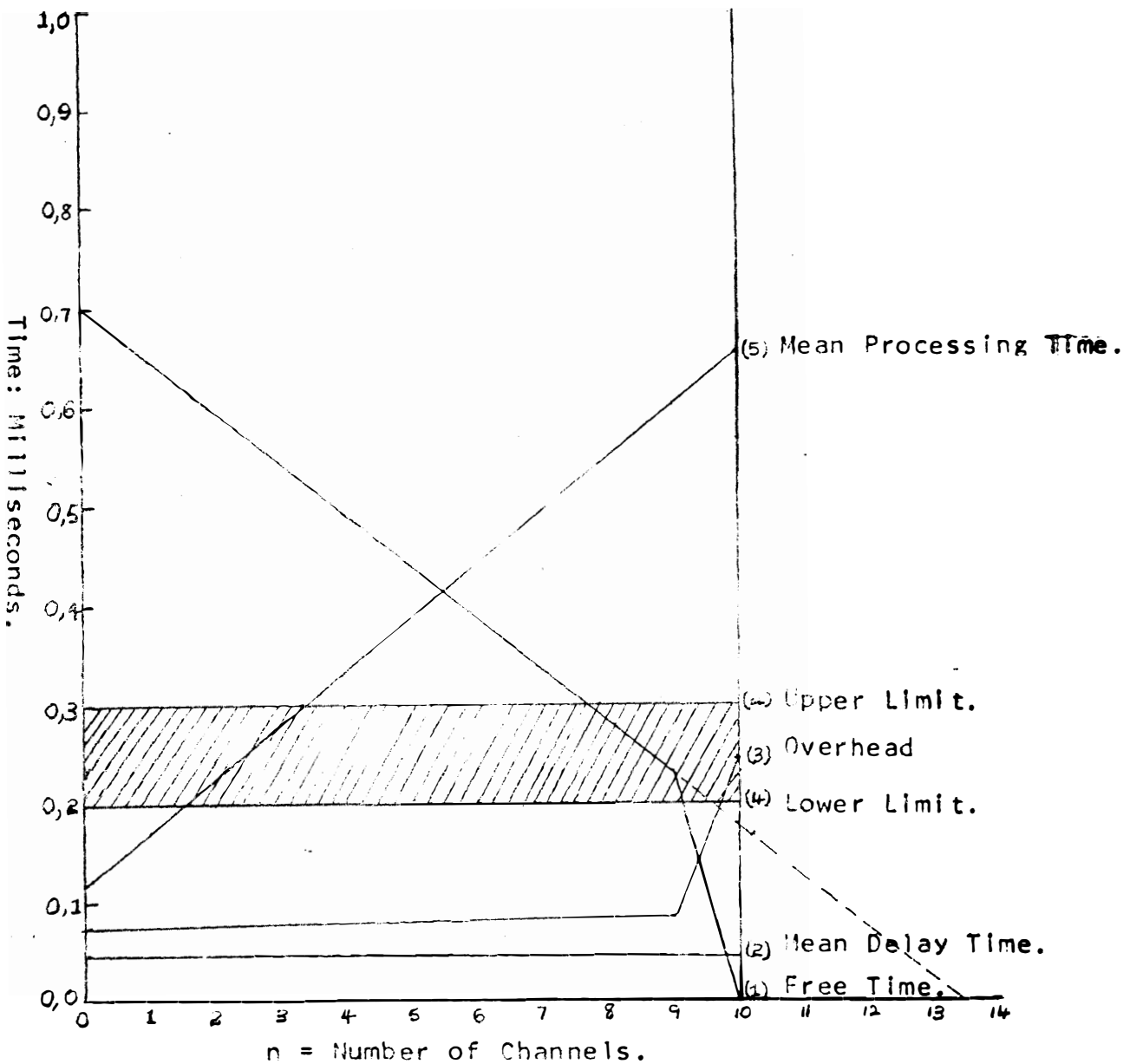
where To is the overhead time.
Ts is the sample interval.
Ti is the interrupt processing time.
Tf is the free time.

The graph of the free time shows that it was not possible to read more than 18 channels of data in one millisecond and perform other non-real-time functions.

Figure 9.5a TIME-SHARING TWO REAL-TIME PROGRAMS, CONTINUOUS TIMING AND NORMAL MODE.

EEG PROGRAM

The program timed is the EEG program (ETn) reading n channels of data and writing the data to tape.
It was time-shared with the ECG program (CD1) reading 1 channel and writing timing data to disk.



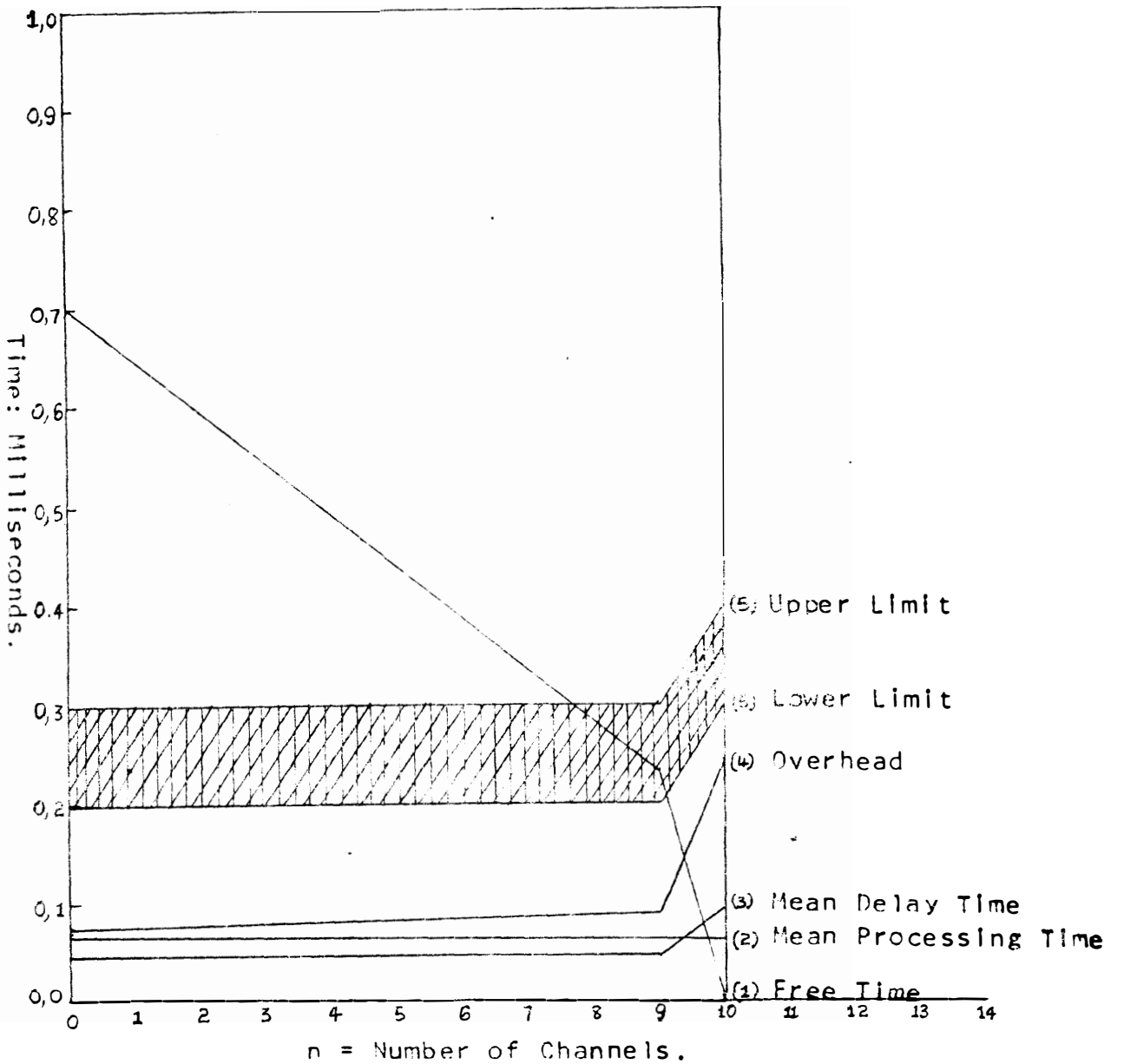
NOTES:

(1) Free Time is CPU time available to non-real-time programs.
(2) Mean Delay is the mean delay in initiating the interrupt servicing routine.
(3) Overhead is the CPU time used to spool data to disk or to tape plus VORTEX system overheads.
(4) The maximum delay in initiating the interrupt servicing routine. Upper and lower limits show the coarseness of the measurements.
(5) Mean Processing Time is the mean time to read from 0 to 10 channels of data.

Figure 9.5b TIME-SHARING TWO REAL-TIME PROGRAMS.
CONTINUOUS TIMING IN NORMAL MODE.

ECG PROGRAM

The program timed is the ECG program (CD1) reading 1 channel of data and writing timing data to disk.
It was time-shared with the EEG program (ETn) reading n channels and writing timing data to tape.



NOTES:

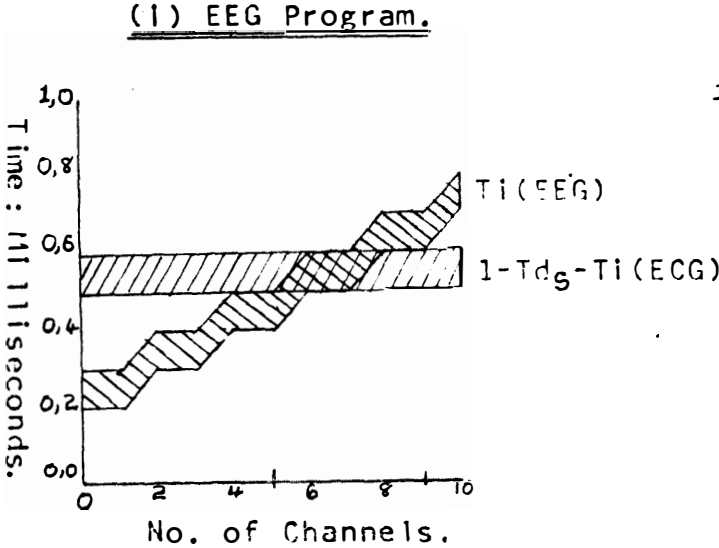
(1) Free Time is CPU time available to non-real-time programs.
(2) Mean Processing Time is the mean time to read 1 channel of data.
(3) Mean Delay is the mean delay in initiating the interrupt servicing routine.
(4) Overhead is the CPU time used to spool data to disk or to tape plus VORTEX system overheads.
(5) The maximum delay in initiating the interrupt servicing routine. Upper and lower limits show the coarseness of the measurements.

Figure 9.5c TIME SHARING TWO REAL-TIME PROGRAMS.
COMPARISON WITH THEORETICAL UPPER BOUNDS.

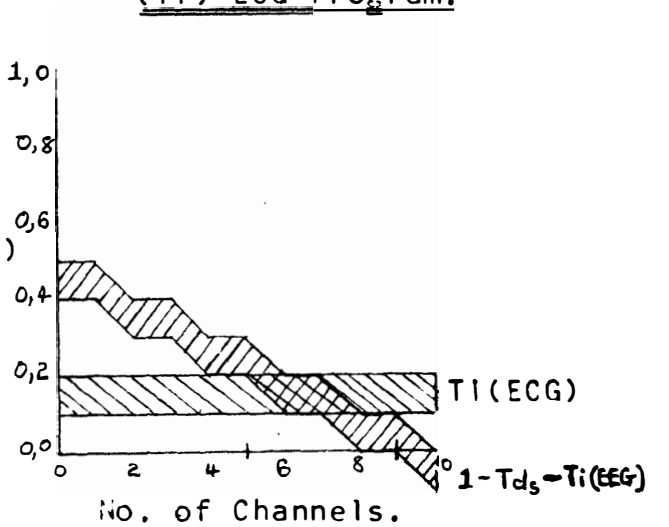
ECG program reading 1 channel every millisecond.
EEG program reading from 0 to 10 channels every millisecond.

Comparison of measured processing time with theoretical maximum permitted processing time.

(i) EEG Program.

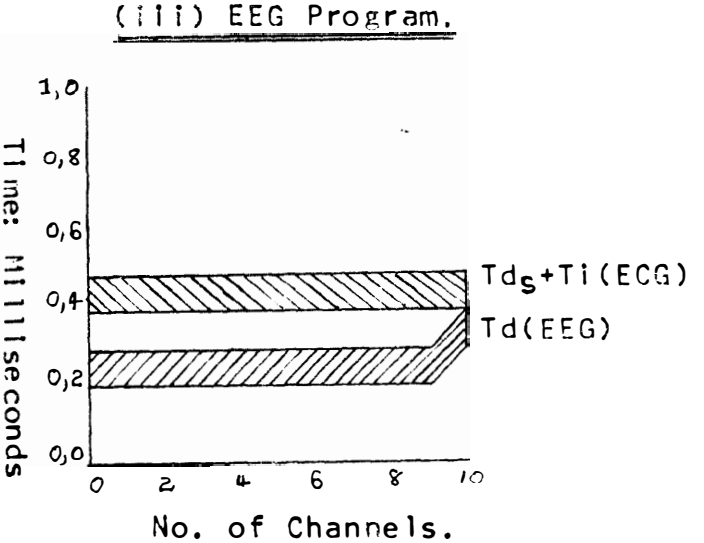


(ii) ECG Program.

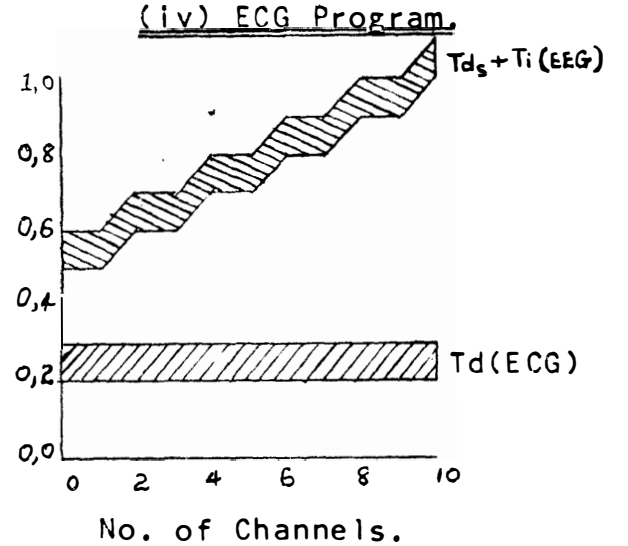


Comparison of measured delay with theoretical maximum delay.

(iii) EEG Program.



(iv) ECG Program.



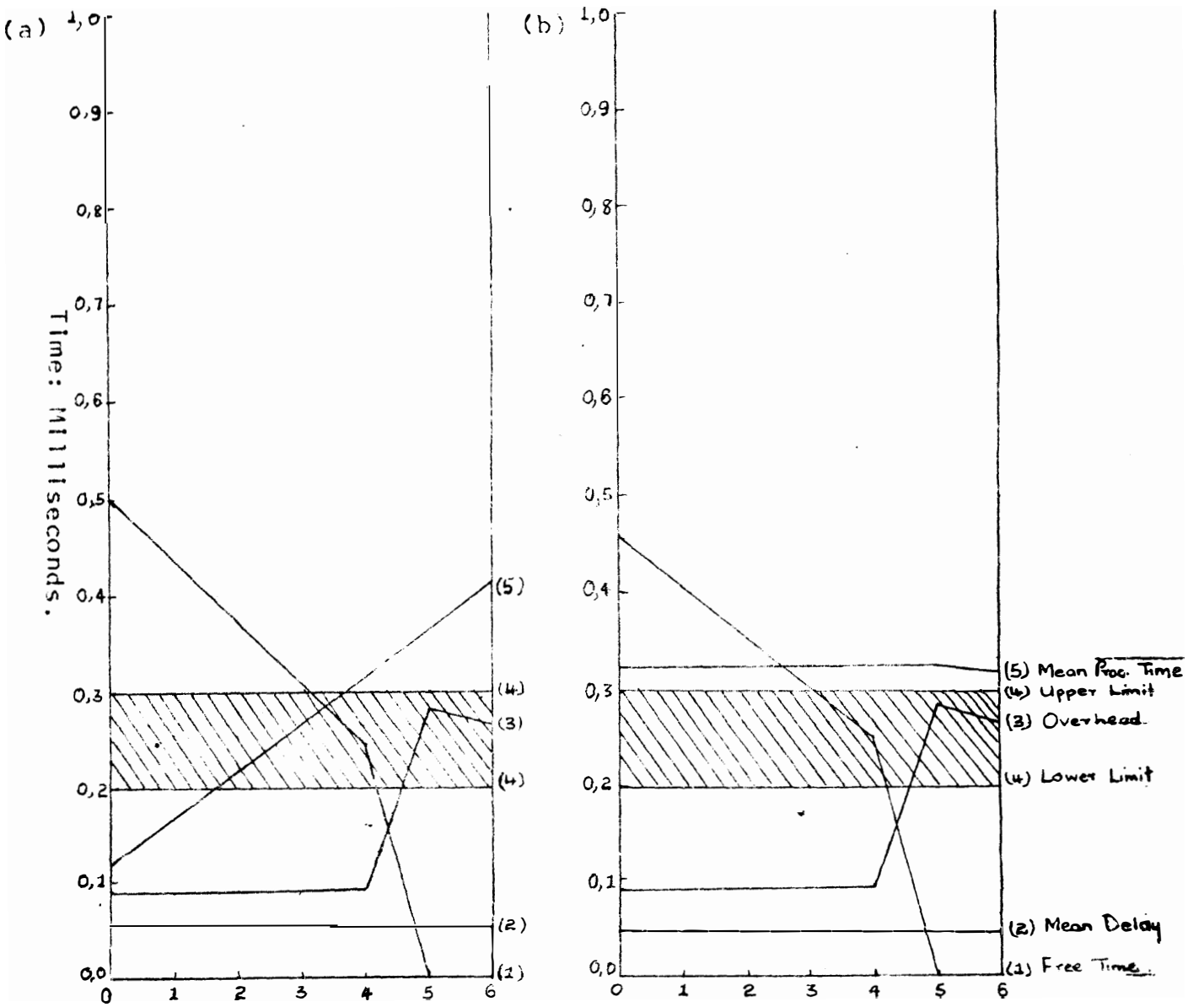
NOTE:

(1) Td = 0,3 milliseconds. The bands show the possible extent of rounding errors in Ti and Td.
(2) For no loss of data, (9.7) shows that:
$Ti_1 \leqslant 1 - TD_s - Ti_2$.
(3) The upper bound for the delay suffered by an interrupt routine is given by (9.2):
$Td_1 \leqslant Td_s + Ti_2$.

Figure 9.6a TIME-SHARING TWO REAL-TIME PROGRAMS. CONTINUOUS TIMING IN NORMAL MODE.

The two programs measured and time-shared are:-

In figure (a) The EEG program (ETn) reading n channels and
In figure (b) The EEG program (C5D) reading 5 channels.



NOTES:

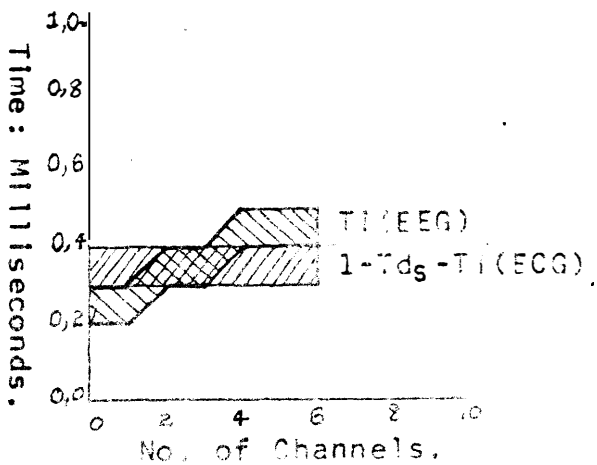
(1) Free Time: CPU time available to non-real-time programs.
(2) Mean Delay is the mean delay in initiating the interrupt servicing routine.
(3) Overhead is the CPU time used to spool data to tape or to disk plus VORTEX overheads.
(4) The maximum delay in initiating the interrupt servicing routine. Upper and lower limits show the coarseness of the measurements.
(5) Mean Processing Time is the mean time to read n channels of data for the EEG program, and 5 channels for the ECG programs.

Figure 9.6b TIME SHARING TWO REAL-TIME PROGRAMS.
COMPARISON WITH THEORETICAL UPPER BOUNDS.

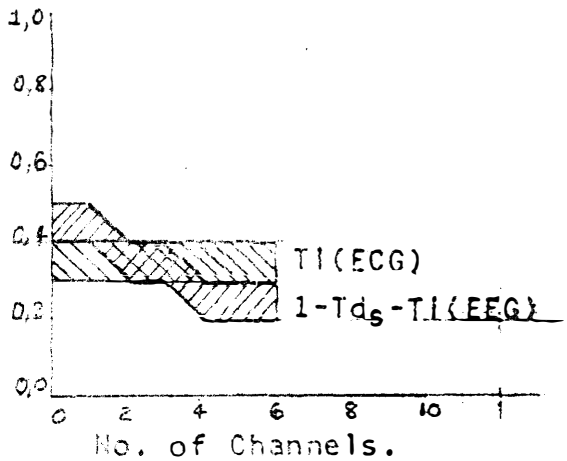
ECG program reading 1 channel every millisecond.
EEG program reading from 0 to 6 channels every millisecond.

Comparison of measured processing time with theoretical maximum permitted processing time.

(i) EEG Program.

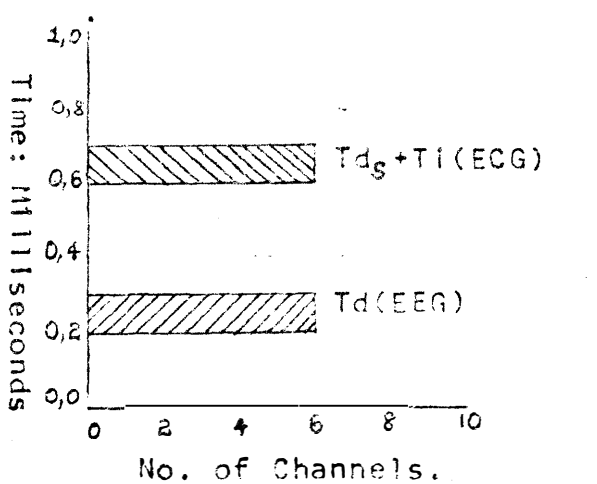


(ii) ECG Program.

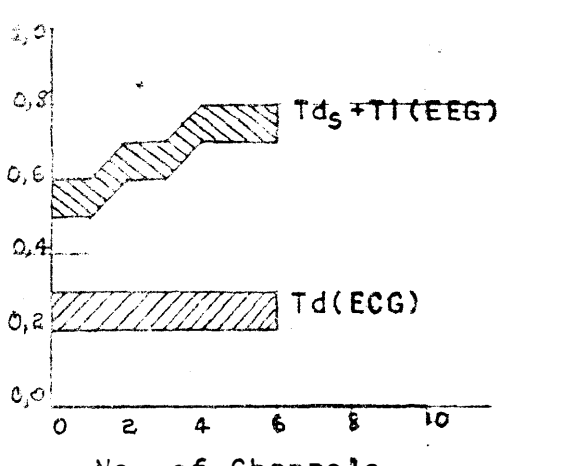


Comparison of measured delay time with theoretical maximum delay.

(iii) EEG Program.



(iv) ECG Program.



NOTE:

(1) $Td_S = 0,3$ milliseconds. The bands show the possible extent of rounding errors in TI and Td.

(2) For no loss of data, (9.7) shows that:
$$TI_1 \leqslant 1 - Td_S - TI_2 .$$

(3) The upper bound for the delay suffered by an interrupt routine is given by (9.2):
$$Td_1 \leqslant Td_S + TI_2$$

Results with Two Real-Time Programs

The First Experiment

Figures 9.5 and 9.6 show the effects of time-sharing two real-time programs. With both programs sampling at 1 millisecond intervals, the EEG program successfully read 9 channels while the ECG program was reading 1 channel; and with the ECG program reading 5 channels the EEG program read 5 channels without loss of data. In both cases when the EEG program was set to read one more channel there was insufficient CPU time for the spooling programs to write the data to off-line storage and records were lost.

However these results are not conclusive. Since the activation of the second program was delayed until the processing of the first was complete and they used the same sampling intervals, despite random starting points the programs would tend to synchronise, particularly when large numbers of channels were used. Thus in a run of one minute succesive intervals would have similar values instead of the 60,000 different values when Command Timing is used.

To obtain greater certainty, the theoretical limits to the number of channels that could be handled were considered and then a second experiment with unequal sampling intervals was run.

Comparison with The Theoretical Upper Bounds for Ensuring Data Intergrity

a) Maximum Permissible Processing Time

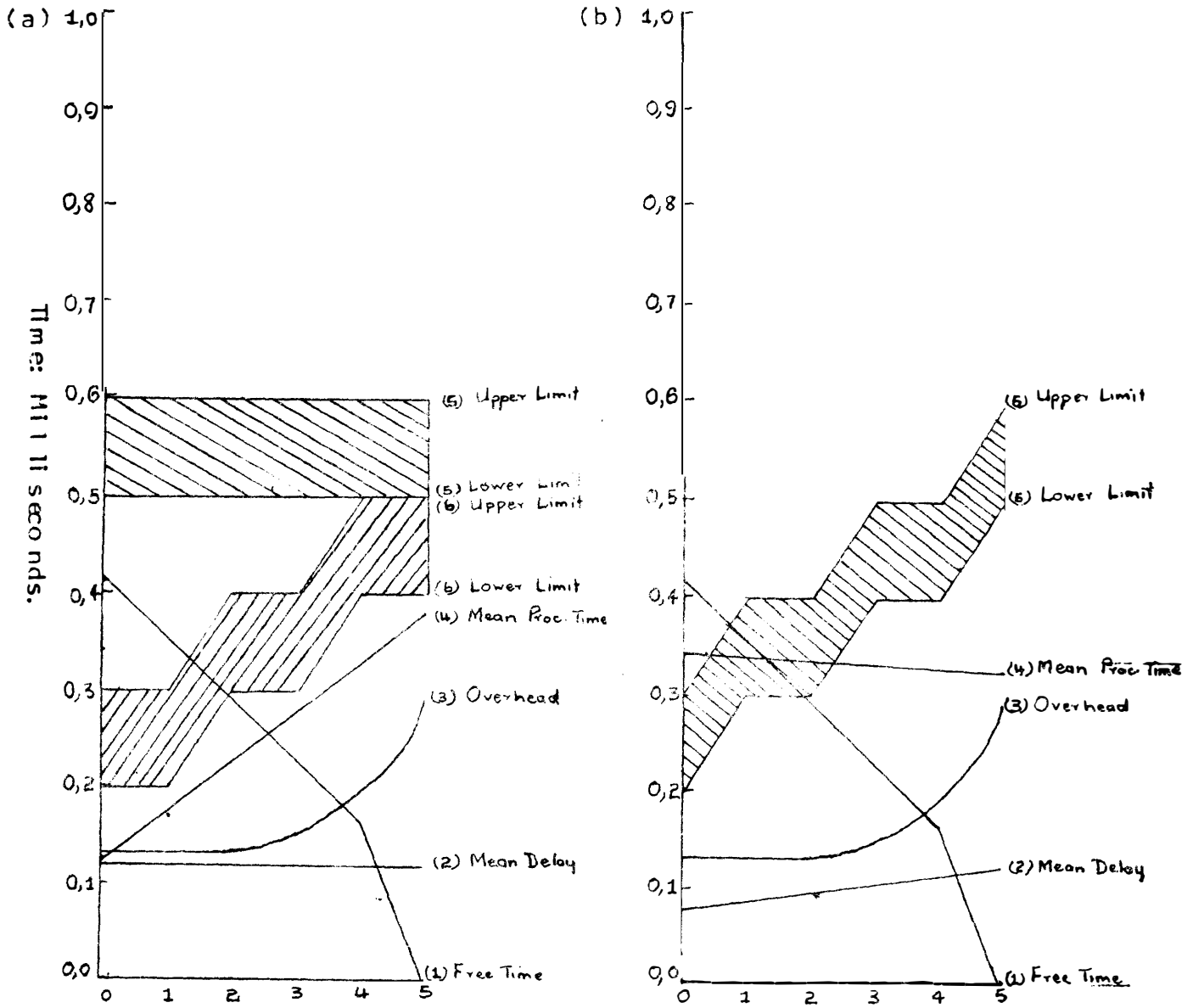
When two real-time programs sampling at 1 kilohertz are time-shared, (9.12) gives an upper bound for the processing times of the programs that will ensure no data is lost :

$$Ti_1 \leq 1 - Td_s - Ti_2$$

Td_s the maximum delay due to non-real-time programs was estimated from figure 9.4 to be 0.3 milliseconds. Figures 9.5a(i) and (ii) show that (9.12) is certainly satisfied for the ECG program reading 1 channel and the EEG program reading 5 channels, and is possibly satisfied for 9 channels if rounding errors in the measurements are ignored. With the

Figure 9.7a TIME-SHARING TWO REAL-TIME PROGRAMS SAMPLING AT DIFFERENT RATES. CONTINUOUS TIMING IN NORMAL MODE.

The two programs mesured are in :-
Figure (a). the EEG program ETn reading n channels of data at 1,0 millisecond intervals and
Figure (b). the ECG program reading 5 channels at 0,9 millisecond intervals.



NOTES:

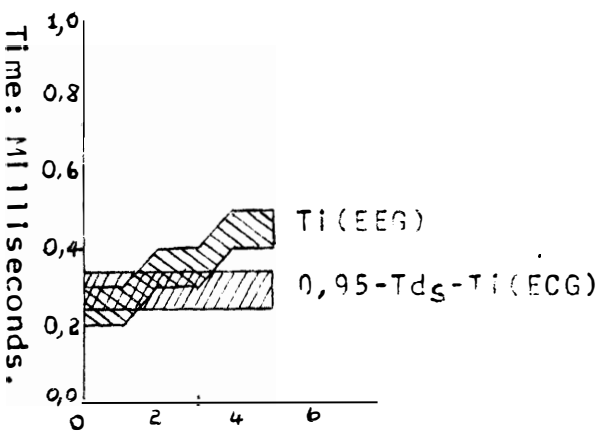
(1) Free Time: CPU time available for non-real-time programs.
(2) Mean Delay: The mean delay in initiating the interrupt servicing routine.
(3) Overhead is the CPU time used to spool data to disk or to tape plus VORTEX overheads.
(4) Mean Proc. Time is the mean time to read n channels for the EEG program, and 5 channels for the ECG program.
(5) The maximum delay in initiating the interrupt servicing routine. Upper and lower limits show the coarseness of the measurements.
(6) Maximum Processing Time, ie, the maximum time to read n channels of data. Upper and lower limits show the coarseness of the measurements.

Figure 9.7b TIME SHARING TWO REAL-TIME PROGRAMS.
COMPARISON WITH THEORETICAL UPPER BOUNDS.

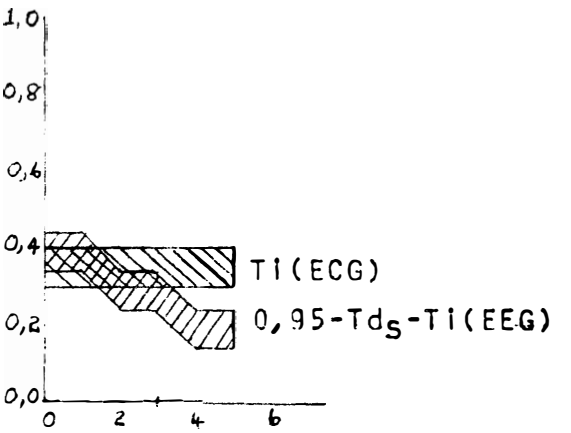
ECG programs reading 5 channels every 0,9 milliseconds.
EEG programs reading from 0 to 6 channels every millisecond.

Comparison of measured processing time with theoretical maximum permitted processing time.

(i) EEG Program

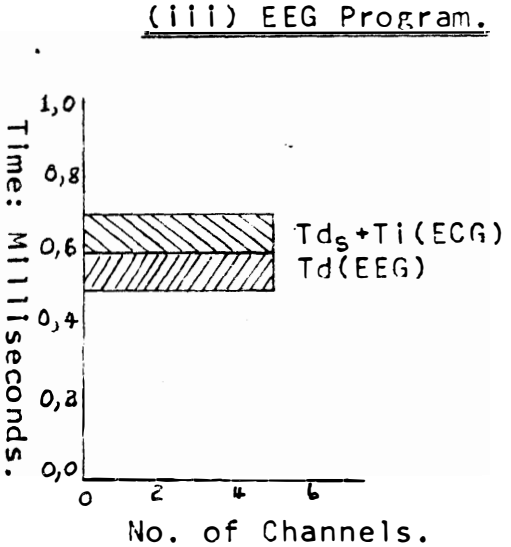


(ii) ECG Program

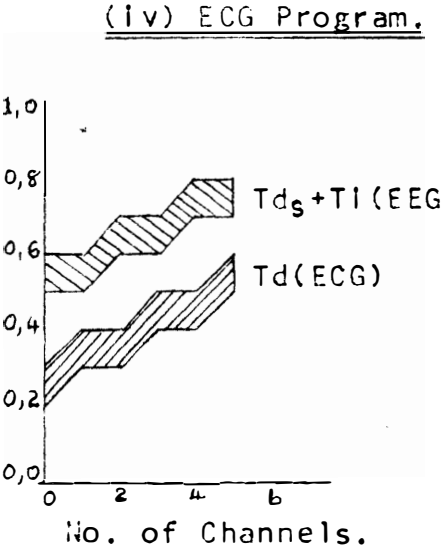


Comparison of measured delay with theoretical maximum delay.

(iii) EEG Program.



(iv) ECG Program.



NOTE:

(1) Td_S = 0,3 milliseconds. The bands show the possible extent of rounding errors in Ti and Td.

(2) For no loss of data, (9.16) shows that:
$Ti_1 \leqslant \frac{1}{2}(Ts_1 + Ts_2) - Td_S - Ti_2$.

(3) The upper bound for the delay suffered by an interrupt routine is given by (9.2):
$Td_1 \leqslant Td_S + Ti_2$.

ECG program reading 5 channels, figures 9.6c(i) and (ii) show that (9.12) is satisfied for the EEG program reading 1 channel, and if rounding errors in the times can be ignored, possibly 6 channels.

b) Maximum Predicted Delay

In (9.7) it is shown that the maximum a program will be delayed is

$$Td_1 \leq Td_s + Ti_2$$

Figures 9,5c(iii) and (iv) and 9.6c(iii) and (iv) show that in the experiments the maximum Td measured was constant (except for the case of 10 channels in figure 9.5c(iii)) and was well below the predicted delay. This suggests that within the limits of 9 and 5 channels respectively, neither program delayed the other.

The Second Experiment

As mentioned earlier, determining the distribution of delays for each condition (i.e. number of channels) would have required a large number of runs at each condition. This was not feasible in the time available, but the same information could be obtained by rerunning the experiment with different sampling intervals for each program to eliminate synchronization. Therefore a further experiment was performed with the ECG program reading 5 channels and sampling at 0,9 milliseconds and the EEG program reading 0 to 5 channels and sampling at 1 millisecond intervals. The results are shown in figure 9.7.

The constraint for maximum processing time, (9.10), in this case becomes :

$$Ti_1 \leq \tfrac{1}{2} (Ts_1 + Ts_2) - Td_s - Ti_2 \qquad (9.16)$$

and since $Ts_1 = 0,9$ and $Ts_2 =$ 1 millisecond :

$$Ti_1 \leq 0,95 - Td_s - Ti_2 \qquad (9.17)$$

The results from this experiment corroborate the previous results. Figures 9.7c(i) and (ii) show that the constraint is satisfied for possibly 3 channels. No data was lost however, when 5 channels were read, thus showing that the constraint is conservative.

The upper bound on the delays likely to be experienced is possibly reached by the EEG program - figure 9.7d(iii) - and is closely followed by the ECG program - figure 9.7d(iv). This is expected, for with unsynchronized sampling intervals, the interrupt for each program will occur progressively in steps of $\frac{1}{9}$ or $\frac{1}{10}$ across the entire sampling interval of the other program and thus the condition of maximum interference will arise.

Conclusions

The upper bound on the processing times of real-time programs given by (9.10), i.e.

$$Ti_1 + Ti_2 \leq \tfrac{1}{2}(Ts_1 + Ts_2) - Td_s$$

is conservative due to the relatively large rounding errors in measuring the parameters. In all three experiments this constraint was exceeded without loss of data.

The upper bound on the delays experienced by one program when time-shared with another real-time program is given by (9.7) i.e.

$$Td_1 \leq Td_s + Ti_2$$

and its limits were approached when the two programs were unsynchronized. When a real-time program was run alone, the mean free time curve was linear and could be used to predict the maximum number of channels that would be read - figure 9.4. However, when two real-time programs were time-shared the curve could not be used as the extrapolated maximum number of channels exceeded the number that could be read. This occurred when the two programs were sampling at the same rate or at slightly different rates. No explanation has yet been found for this anomaly.

The system is able to handle the stipulated maximum load for an EEG project, namely sampling 16 channels a millisecond.

When an EEG project uses 8 or fewer channels (the more likely case) the system is able to time-share a second real-time program using Camac modules.

Program Design Guidelines

The experiments resulted in the following rules which may be used as rough guides in designing a real-time data acquisition system.

1) A real-time program run alone may use up to 95% of the CPU capacity.

2) Two real-time programs shared, may together use up to 65% of the CPU capacity.

3) A conservative estimate of the "housekeeping" necessary to service an interrupt is 0,06 milliseconds.

4) The time to read 1 channel is 0,049 milliseconds.

5) The fraction of CPU time used by a real-time program may be estimated by running it with a program measuring unused time and allowing for a 5% overhead.
i.e. $R + S + 0{,}05 = 1$

where R is the fraction of CPU time used by the real-time program, and S is the fraction of CPU time used by the measuring program. This method does not require modification of the real-time program.

9.4.6 Experiments with Block Transfer Using Continuous Timing

Further experiments measured the delays experienced by a program using Block Transfer. The procedure used for the second series of measurements was followed except that the interrupt servicing routine was initiated by the interrupt generated by the BIC when it had transferred the last word of a block of data. Results are shown in table 9.4 below.

Line 1 of table 9.4 shows very similar results to line 1 of table 9.3,i.e. the maximum delay time was 0,2 milliseconds, the mean delay time, measured at 0,048 milliseconds, is essentially zero, and the average processing time was 0,105 milliseconds. While the program was reading 17 channels ** at 1 millisecond intervals, 85% of the CPU time was available for non-real-time programs. This is because with Block Transfer the data conversion (Tc in figure 9.1) is carried out in parallel with CPU operations. The same operation in Normal Mode leaves only 10% of the CPU time for other programs.

Line 2 of table 9.4 shows the delay experienced when Block Transfer was time-shared with a program continuously accessing the disk. These delays occurred several times a second, and appear to be related to the seek time of the disk because the shortest delays (about 15 milliseconds) are experienced when the same record is read repeatedly, and the longest delays (about 78 milliseconds) occur when records from widely separated files are read. The reason this occurs despite the high priority of the interrupt is being sought but until it is found, time-sharing a real-time program using Block Transfer is ruled out.

** When more than one channel is read using Block Transfer, the first channel is read twice. Thus reading 17 channels provides 16 channels of data and hence the comparison with Normal Mode reading 16 channels.

Table 9.4 Statistics of Selected Samples of 1000 Intervals from Experiments Using Block Transfer and Continuous Timing

These samples have the longest delay in batches of 60 to 90 samples where each sample represents 1000 intervals. 17 channels were converted each cycle and transferred to memory.

			Td : milliseconds		Ti : Milliseconds	
Line No.	Program Timed	Other Programs	Time to interrupt		Processing Time	
			Maximum	Mean	Maximum	Mean
1	B	-	0,2	0,048	0,2	0,105
2	B	R	43,3	0,115	0,4	0,105

Program key : B - EEG program reading 16 channels using Block Transfer.

R - Disk exerciser.

Data Conversion Times

The timing chart for Block Transfer data sampling (figure 9.1) shows that for no loss of data

$$Tc + Td + Ti \leq Ts \qquad (9.18)$$

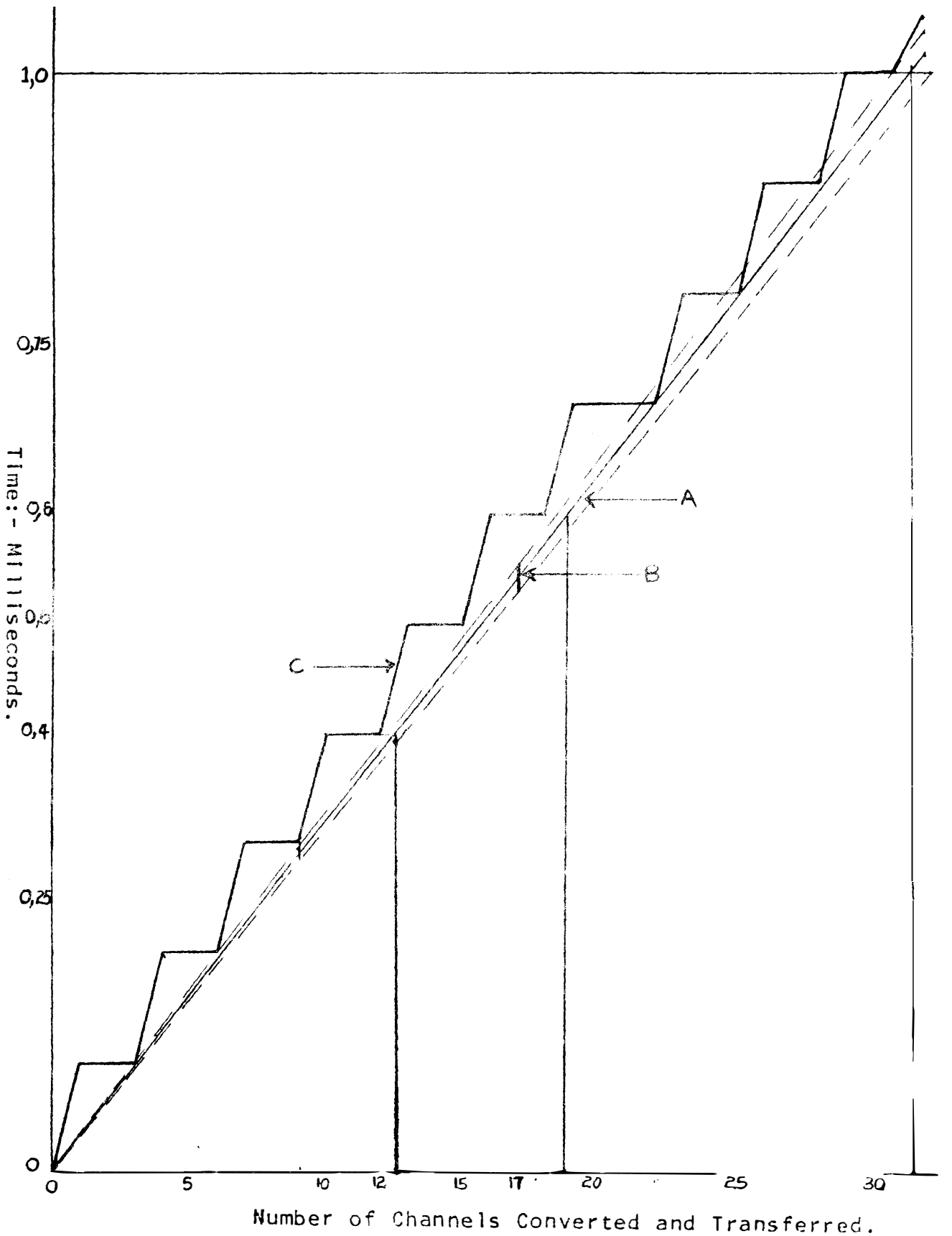
where Tc is the data conversion and transfer time,
Td is the maximum delay experienced,
Ti is the interrupt processing time, and
Ts is the sample interval.

Ts was set by the sampling program, Ti and Td were measured as described above and Tc was measured in a further series of experiments using the following procedure :-

a) All interrupts were disabled.
b) The BIC and Block Transfer were initialized.
c) The Block Transfer was started and the Varian Timer read.

Figure 9.8 DATA CONVERSION TIMES - MEANS OF 1000 BLOCK TRANSFERS.

NOTE: The full line A, represents the case when the ADC input was open. The range of different voltages for 17 channels is given by vertical bar B, and the broken lines indicate extrapolations from this range. The stepped line C, is the maximum conversion time measured.



d) The program looped on a "sense BIC ready" instruction until the Block Transfer had completed.

e) The Varian Timer was read.

f) All interrupts were enabled.

The difference in the times gave the duration of the specified number of data conversion, data transfer, and channel switch cycles under control of the Block Transfer Option equivalent to Tc in figure 9.1. The times required to read the Varian Timer and to execute several instructions before reading the Timer the second time amount to a few micro-seconds and can be ignored. Results are shown in figure 9.8 below. For these experiments the input to the ADC was open. The ADC uses a successive approximation method and the conversion in time depends on the voltage converted. For example, figure 9.8 shows that with the ADC input open the mean time for reading 17 channels was 0,541 milliseconds. With varying voltages mean conversion times ranged from 0,476 to 0,549 milliseconds.

Taking Td and Ti from table 9.4 and substituting into (9.18) with Ts = 1 millisecond gives :

$$Tc \leq 1 - 0,2 - 0,2 = \quad 0,6 \text{ milliseconds.} \qquad (9.19)$$

For reading 17 channels the maximum measured value of Tc was 0,6 milliseconds, satisfying (9.18). Tc, Ti and Td are maxima of 100 measurements, each subject to a rounding error of 0,1 milliseconds. Allowing for the worst case, (9.18) becomes

$$Tc \leq 1 - (Ti + 0,1) - (Td + 0,1) \text{ milliseconds}$$

or

$$Tc \leq 0,8 - Ti - Td. \qquad (9.20)$$

Using the same substitution as above gives

$$Tc \leq 0,4 \text{ milliseconds}$$

which allows up to 12 channels to be converted and read. This calculation is necessarily conservative given the coarseness of the measurements, and in practice 17 channels have been read at 1 millisecond intervals without loss of data, and possibly up to 25 channels could be safely read.

9.4.7 Timing of R.T.E. Functions

An additional experiment was run to determine VORTEX overheads in switching tasks using functions provided by the Real Time Executive (R.T.E.) component of the operating system.

The interrupt handling facilities provided by VORTEX allow for routines to be Directly Connected or Indirectly Connected to interrupts. Indirectly Connected interrupt handling routines are scheduled by the Common Interrupt Handler whenever their interrupts occur, and they run as normal VORTEX programs and may use all the facilities provided by the system. Directly Connected interrupt routines have control passed directly to them when the CPU recognizes their interrupts. Thus they are not subject to software priorities or to software overheads in receiving control as Indirectly Connected routines are. However they may not use VORTEX facilities such as input/output.

When an Indirectly Connected interrupt handling routine is scheduled it is subject to two sources of delay after the occurrence of an interrupt:

1) The delay in initiating the Common Interrupt Handler. This is common to all interrupt processing routines and the experiments described above in sections 9.4.4, 9.4.5 and 9.4.6 measured this type of delay.

2) The delay between the acceptance of an interrupt by the Common Interrupt Handler and the transfer of control to the appropriate routine. This function is similar to that performed by the macro, RESUME and so it was used to provide information about this second source of delay.

The results of the experiment are shown in table 9.5. Two programs measured the time between the execution of a RESUME macro in one and the dispatching of the second program by the VORTEX dispatcher.

Results showed that the average time for control to be passed to the second program in the experiment was less than 0,125 milliseconds, but delays of up to 0,5 milliseconds occurred when the programs were run alone and 0,7 milliseconds when time-shared with a program reading the disk. When the two programs were run with a lower priority than the operating

system programs (such as the disk driver and operator communications task) but at a higher priority than other time-shared programs, eg the disk exerciser, the mean delay was 0,255 milliseconds and the maximum delay 1,3 milliseconds. Further, operator commands could cause delays of up to 8,0 milliseconds. Clearly, the relative software priorities of Indirectly Connected interrupt handling programs are crucial to quick response times.

Calculations show that, even if Indirectly Connected interrupt routines are subject only to the second source of delay, the response times are too slow and uncertain to allow useful time-sharing with real-time programs sampling data at 1 millisecond intervals. A safe estimate of Td_s, the delay due to non-real-time programs when control is passed from the Common Interrupt Handler to an interrupt program, is 0,7 milliseconds. This is the delay a program using the disk may cause. For two real-time programs constraint (9.11) applies :

$$Ti_1 + Ti_2 \leq 1 - Td_s = 0{,}3 \text{ milliseconds.}$$

Since the EEG and the ECG program each had a maximum interrupt processing time when reading 1 channel of 0,2 milliseconds, $Ti_1 + Ti_2$ = 0,2 + 0,2 = 0,4 milliseconds and they could not be time-shared using Indirectly Connected interrupt routines. For one real-time program (i.e. Ti_2 = 0) reading not more than 3 channels time-sharing with a non-real-time program is possible. (For 3 channels Ti = 0,3 milliseconds and for 4 channels Ti = 0,4 milliseconds.)

The experiment thus showed that the magnitude of the delay in transferring control between programs using the the Common Interrupt Handler necessitates the use of Directly Connected interrupt routines in the NIPR environment.

Table 9.5 Time to RESUME a Program

Time-Sharing Condition		Delay : milliseconds	
Environment	Priority	Maximum	Mean
Alone	High	0,5	0,123
Disk exerciser	High	0,7	0,123
Alone	Low	0,7	0,153
Disk exerciser	Low	0,3	0,257

9.4.8 Further Results of the Experiments

The experiments raised additional questions that were not part of the feasibility study but will require further investigation when the load on the computer increases.

1) The sudden increase in mutual interference and overhead when two time-shared real-time programs are using the CPU almost to capacity described in section 9.4.5.

2) The effect of other programs on the processing time of the interrupt handling routines of real-time programs as shown in figure 9.7a where the mean processing time of the ECG program decreased slightly as the number of channels read by the EEG program increased. Figures 9.5a and 9.5b show the EEG program reading up to 10 channels while time-shared with the ECG program reading 1 channel. However, when the fix to the disk driver was used, the processing times (not the delay times) for both programs increased slightly and the EEG program was only able to read 8 channels.

3) The factors that contribute to the overhead (i.e. the difference between the time not available to non-real-time programs and the time actually used by a real-time interrupt handling routine). Two real-time programs time-shared have almost double the overhead of a single real-time program - compare figures 9.4 and 9.5a. Most of the overhead time is required by the data spooling routines and very little by VORTEX executive routines such as the dispatcher and real time clock processor. Each task requires several system control blocks which require periodic scanning by the dispatcher and other system routines but as increasing the number of real-time programs increases the number of tasks from 10 to 12 the number of system control blocks increases in the same ratio which is considerably less than two.

9.4.9 Conclusions and Summary

Measurements were made of timing characteristics of real-time data acquisition programs running in Normal mode and Block Transfer mode.

In Normal mode it was found that, sampling at 1 millisecond intervals one program can read up to 18 channels when time-shared with non-real-time programs, while two programs can together read up to 9 or 10 channels. Thus the system is able to handle the most severe sampling requirements that can be imposed by EEG projects and with the more usual EEG project the system is able to time-share ECG and psychometric projects.

The VORTEX disk driver can, under certain circumstances, generate delays of up to 1,1 milliseconds in sampling data, but the problem can be circumvented by preventing it from disabling Camac interrupts.

With Block Transfer, 16 channels per millisecond can be read using only 15% of the available CPU time, but it requires exclusive use of the Camac and thus Camac commands for other real-time programs must be synchronized and issued before the system is initiated for Block Transfer. This can be handled by making all time-shared real-time programs use a common routine for issuing Camac commands. A more serious problem is that, when Block Transfer is time-shared with programs using the disk, delays of up to 70 milliseconds occur. The cause has not yet been isolated, and until the problem is solved programs using Block Transfer must be run alone and the data must be written to tape.

Experiments using the RTE function RESUME established that system overheads when using the VORTEX interrupt handling procedures are too great for sampling at 1 kilohertz. Thus it is necessary to use the alternative method provided for the VORTEX system, namely directly connected interrupt handling. Further,in order to time-share Camac operations it is also necessary to bypass the normal procedure provided in the Camac system of testing for an interrupt and then locating its source and to have the interrupt generators of certain of the Camac modules hardwired directly to PIM lines.

Chapter 10 Further Aspects of the Feasibility Study

10.1 The Simultaneous use of Several Laboratories

The second object of the feasibility study was to study the handling of psychological experiments using a remotely situated computer. This is a matter of some importance as it would allow equipment to remain set up in laboratories for the lengthy periods necessary for obtaining sufficient numbers of subjects for psychological experiments.

Three laboratories were used, each communicating with the computer by means of a terminal and a 25-core screened post-office cable for connecting electronic apparatus. Three kinds of terminal were used. The EEG laboratory was equipped with a 4010 Tektronix display unit with graphic facilities, a hardcopy unit for displaying statistical information and a keyboard for controlling experiments; the Personality and Temperament laboratory was given an ASR-33 teletype for controlling and providing a permanent record of the progress of experiments; the Psychometrics Division had an Infoton display for presenting stimuli and a keyboard for subjects' responses. All the terminals functioned satisfactorily and research staff reported that their work was greatly facilitated by being able to use their own laboratories. The only adverse report was that the noise of the ASR-33 teletype was distracting to subjects and a silent display with hardcopy facilities should be used instead.

10.2 The Amount of Main Memory Used

Another aim of the feasibility study was to determine the amount of main memory required.

The following were established :-

(a) The VORTEX operating system requires $10\frac{1}{2}$k.

(b) Each program written in Fortran or using Fortran subroutines requires an additional $3\frac{1}{2}$k for Fortran system subroutines which are not reentrant.

(c) Input/output to disk or to tape requires buffers whose sizes are specified in table 10.1.

(d) Program development requires facilities whose memory requirements are listed in table 10.2.

Table 10.1 shows that two real-time programs, each of 2k in length, reading 1 and 8 channels and writing the data to disk and tape respectively, would together with the VORTEX system, require 19k ($(2+\frac{1}{4}+\frac{1}{4})$k for the first program, (2+2+2)k for the second program and $10\frac{1}{2}$k for VORTEX). Thus the minimum viable system for NIPR usage is 24k. Even then this leaves only 5k for background, which limits the facilities for program development to file editing and operator's and background commands. This explains the degradation mentioned earlier which is experienced during program development while real-time programs were running. If one of the real-time programs also uses Fortran then background work is almost entirely excluded. Up to the present, the two real-time programs described in chapter 9, were simple data gathering programs with modest requirements and program development has been feasible while they were running. The psychological testing programs, the EEG averaging programs and the programs for controlling experiments that are envisaged in future work will make much heavier demands for which 24k will certainly be inadequate and it will then be necessary to increase the memory to 32k.

Table 10.1 Input/Output Buffering Memory Requirements

Storage Medium	Data Rate - Words/Millisecond	Minimum Buffer Size-Words
Tape	2	120
	3	240
	4	360
	5	480
	6	720
	7	1200
	8	2000
Disk	1	120
	2	240
	3	360
	4	480
	5	600
	6	720
	7	840
	8	1080
	16	2520

Notes :

1) A real-time program requires two buffers of the size specified in the table,

2) For ease of comparison with disk data rates, the tape buffers are given in multiples of sector sizes. The reason that the tape buffers are smaller than the disk buffers for data rates up to 6 words per millisecond is that disk operations have a negligible effect on the rate at which data can be written to tape whereas other disk operations (by "stealing" the single read/write head) have a large effect on the rate at which data can be spooled to disk. This does not vary with the number of disk operations because the data spooling program should have the highest priority in the machine and can only be delayed by operations that have already been initiated.

Table 10.2 Memory Requirements for Program Development Facilities

Facility	Memory Requirements
Operator Commands	1k
File editing	2k
Background commands	3k
Load module generation	7k
File maintenance	7½k
Assembly	8k
Fortran compilation (trivial)	9½k
Fortran compilation (useful)	10½k

Note : The file editing facility is provided by a program written by the NIPR and usually run in the foreground.

10.3 Printing Requirements

The initial installation used the second ASR-33 teletype and a Tektronix hardcopy unit for permanent copies of programs, data and/or operational steps. Experience soon showed that the printing load was too heavy for the teletype, and the Tektronix hardcopy unit, though essential for permanent records of statistical information on experimental data was too expensive and too slow for general use. A Centronics line printer with a speed of 165 characters per second was therefore added and has proved adequate.

Chapter II Final Conclusions and Recommendation

The conclusions regarding the feasibility of the five propositions identified in chapter 4 are :

(1) Computer participation in NIPR experiments can be adequately handled from terminals located in laboratories remote from the computer. Chapter 10 showed that in psychological experiments this is a significant advantage because of the length of time apparatus may have to be left set up.

(2) Two typical NIPR projects can be run simultaneously. Experience with the system in real-time data acquisition and the measurements and tests described in chapter 9 prove that real-time programs can be time-shared and that the data rates that can be sustained are adequate for NIPR requirements. One unresolved problem, namely interference between the VORTEX disk driver and Block Transfer, is being investigated by the Varian agents.

(3) The minimum amount of core required at present is 24k and this may have to be increased to 32k when experiments reading 8 or more channels of data are planned.

(4) The disk and tape can handle the maximum data rates required at present for storage and spooling of data. It was shown in chapter 8 that the disk can handle a data rate of 16000 words per second (which is equivalent to reading 16 channels every millisecond), while the tape has a maximum capacity of 8000 words per second.

(5) The second teleprinter proved inadequate for the printing requirements of the system and it was necessary to install a printer.

Other conclusions are :-

(6) The VORTEX operating system, despite its large memory requirements and its overheads is extremely effective. Program development and testing are immeasurably facilitated both by the system utilities that are provided and by the fact that work can proceed in parallel on the machine from two or more terminals. Without these facilities, progress would have been very much slower.

(7) The Personality and Temperament project demonstrated the feasibility of using an ASR-33 teletype for controlling the type of experiments to be performed in that Division. The teletype, however, is noisy and may disturb subjects and should be replaced with a display unit. This would not entail additional expense.

(8) The 4010 Tektronix display unit with keyboard and hardcopy unit proved adequate for the requirements of the Neuropsychology Division, providing both remote control of the computer and graphic facilities for displaying statistical information relating to the data.

Recommendation

The study has shown that all of the equipment on loan is needed for meeting the data acquisition requirements of NIPR projects. Therefor it is recommended that the equipment be purchased.

APPENDIX A1

TABLE A1.1 COMPUTER CONFIGURATION

Equipment Originally Bought and Installed

QTY	MODEL	DESCRIPTION
1	7000	Varian Model 700 computer including : V73 Central Processor Unit. Hardware Multiply/Divide. Power Failure/Restart. Teletype Controller. Real Time Clock. Dual Memory Bus. 620 Compatible I/O Bus. Direct Memory Access. 7" Chassis with 5 additional module slots. Power Supply. Programmer's Console.
1	7037	8192 words (16 bits) Semi Conductor Memory.
1	7121	Memory Protect.
1	7967	Peripheral Back Plane Wiring Panel. LH
1	7966	Peripheral Back Plane Wiring Panel. RH
1	7962	I/O Expansion Chassis.
1	620-06C	Teletype ASR-33.
3	620-20	Buffer Interlace Controller.
3	7160	Priority Interrupt Module.
1	7911	Dual Controller Adapter.
1	7955	I/O Expansion Power Supply.
1	7920	I/O Cable with Padel Board Connectors.
1	620-36	Disk Memory and Controller.
1	620-30	Magnetic Tape Unit and Controller.
1	72-150-6	I/O Party Line Expander.

TABLE A1.2 COMPUTER CONFIGURATION

Equipment Loaned by the Supplier for the Duration of the Feasibility Study

QTY	MODEL	DESCRIPTION
2	7024	8192 words (16 bits) Core Memory.
1	620-06C	Teletype ASR-33.
1	620-82A	Universal Asynchronous Controller.
1	620-82B	Universal Asynchronous Controller.
1	40/10	Tektronix Display Terminal.

TABLE A1.3 COMPUTER CONFIGURATION

Equipment Added Subsequent to the Original Installation

QTY	MODEL	DESCRIPTION
1	-	Infoton Display Unit.
1	101	Centronics 165 cps Line Printer.
1		Hard copy unit for the Tektronix Display Terminal.

TABLE A1.4 CAMAC CONFIGURATION

QTY	MODEL	DESCRIPTION
1	1902	Borer Basic Crate for plug in power supply, including fan, crowbar and alarm module 1930.
1	2204	Borer Interface Camac branch highway - Varian 620 Computer basic version for program controlled data transfer including : Automatic CNA Scanner with DMA channel only. Block Transfer + LAM Synchronisation. Cable Varian Computer - Interface 3 meters long.
1	1912	Borer Power Pack including + 200V regulated supply.
5	1922	Voltage Regulators. **
1	1502	Borer Crate Controller.
1	1591	Borer Termination Unit for Branch Highway.
1	1704	Borer 16 Channel FET Multiplexer.
1	1243	Borer ADC Successive Approximation with Sample & Hold.
2	1411	Borer Clock/Preset Timer.
2	1033	Borer Input/Output Register.
1	1801	Borer Dataway Display.

** The system was originally supplied with 2 Voltage Regulators which were found to be insufficient and 3 more were supplied in January 1975.

TABLE A1.5 VORTEX OPERATING SYSTEM FEATURES AND FACILITIES

Real-time I/O processing.
Interrupt processing.
Multiprogramming of real-time and background tasks.
Priority task scheduling.
Automatic Load and go.
Device independent I/O system.
Operator communications.
Batch-processing job-control language.
Program overlays.
Fortran compiler.
DASMR assembler.
Load module generator.
Debugging aid.
Source editor.
Disk file management.
System generator.

APPENDIX A2

Figure A2.1 Common Symbols and Macros used in the Examples in the Appendices

```
*  CAMAC MACRO
*
*  USAGE
*                CAMAC  C,N,A,F
*  WHERE
*                C IS THE CRATE REGISTER NUMBER
*                N IS THE CRATE STATION (MODULE) NUMBER
*                A IS THE SUB-ADDRESS
*                F IS THE FUNCTION CODE
*
CAMAC            MAC
C                FORM   2,5,5,4
                 C      P(1),P(4),P(2),P(3)
                 EMAC
*  INDEX REGISTERS
X                EQU    1
B                EQU    2
*
*  BIT TEST DESIGNATIONS
RA0              EQU    040             BT JUMPS FOR A BIT = 0
RA1              EQU    0               BT JUMPS FOR A BIT = 1
*
*  CAMAC MODULE STATION NUMBERS
DISP             EQU    1               DATAWAY DISPLAY
MX               EQU    2               MULTIPLEXER
ADC              EQU    5               ANALOG TO DIGITAL CONVERTER
CLK              EQU    6               CLOCK/TIMER 1
IOREG            EQU    7               I/O REGISTER 1
```

Figure A2.1 Continued

```
*
*    VORTEX  SYSTEM   LOW   CORE   AREAS
LC              EQU    0300              LOW CORE
VØCTL           EQU    LC                CURRENT TASK TIDB LOCATION
VØCPL           EQU    LC+1              CURRENT PRIORITY LEVEL
VØCRS           EQU    LC+2              CURRENT REENTRANT STACK
VØLUP           EQU    LC+14             1st UNPROTECTED WORD
VØLLUP          EQU    LC+15             LAST UNPROTECTED WORD
*
*    TIDB    OFFSETS
TBTRD           EQU    0                 TASK THREAD
TBST            EQU    1                 TASK STATUS
TBPL            EQU    2                 STATUS CONT : PRIORITY
TBEVNT          EQU    3                 INTERRUPT EVENT
TBRSA           EQU    4                 A          REENT & SUSPND.
TBRSB           EQU    5                 B            "   "     "
TBRSX           EQU    6                 X            "   "     "
TBRSP           EQU    7                 OF/P         "   "     "
TBRSTS          EQU    8                 TEMP STRG    "   "     "
TBENTY          EQU    9                 TASK ENTRY LOCATION
TBTMS           EQU    10                TIME COUNTER 1
TBTMIN          EQU    11                TIME COUNTER 2
TBISA           EQU    12                A        INTERRUPT
TBISB           EQU    13                B            "
TBISX           EQU    14                X            "
TBISP           EQU    15                OF/P         "
TBISRS          EQU    16                REENT STACK  "
```

APPENDIX A3

Figure A 3.1 Example of a Program Segment for Initializing the BIC and Camac in a Block Transfer Operation.

```
*   INITIALIZE BIC FOR BLOCK TRANSFER
        EXC     023             INIT BIC
        SEN     022,*+3         CHECK BIC NOT BUSY

        HLT     0777            BOOBOO IF BIC NOT READY NOW
        LDAI    (REC)           ADDRESS OF 1ST DATA WORD

        OAR     022             SET BIC INITIAL ADDRESS REGISTER
        ADDI    15              ADDRESS OF LAST DATA WORD

        OAR     023             SET BIC FINAL ADDRESS REGISTER
        SEN     022,*+3         CHECK BIC NOT BUSY

        HLT     0222            BOOBOO IF BIC BUSY
        EXC     022             ENABLE (ACTIVATE) BIC
*
*   INITIALIZE CAMAC FOR BLOCK TRANSFER
        LDA     CRBLTR          WORD COUNTER & SYNCH FLAG
        OAR     050             SET INTERFACE
        LDA     ADRDBL          LOAD CNAF
        OAR     050             GIVE CNAF TO INTERFACE
*
*   ATTACH INTERFACE TO BIC FOR READING
        EXC     051             ATTACH INTERFACE
          .       .
          .       .
          .       .
CRBLTR  DATA    011756          SET WORD COUNTER + EXT SYNC
ADRDBL  CAMAC   1,ADC,0         READ FROM ADC
REC     BSS     16              DATA BUFFER FOR 16 WORDS
```

A P P E N D I X A4

Table A4.1 Interrupt Assignments

Standard Varian Interrupt addresses.

MEMORY ADDRESS	DEVICE AND FUNCTION
020,021	Memory Protect halt error
022,023	MP I/O error
024,025	MP write error
026,027	MP jump error
030,031	MP overflow error
032,033	MP I/O and overflow error
034,035	MP write and overflow error
036,037	MP jump and overflow error
040,041	Power failure
042,043	Power restart
044,045	Real Time Clock interval
046,047	RTC overflow
100-117	PIM 1 interrupts
120-137	PIM 2 interrupts
140-157	PIM 3 interrupts
160-167	PIM 4 interrupts

Table A4.2 Interrupt Assignments (Continued)

PIM interrupt line assignments

	LINE	ADDRESS	DEVICE AND FUNCTION
PIM 1	0	100,101	BIC 1 complete (End of disk transfer)
	1	102,103	-
	2	104,105	BIC 3 complete (End of magnetic tape transfer)
	3	106,107	SEEK 1 complete
	4	110,111	SEEK 2 complete
	5	112,113	-
	6	114,115	Infoton Read
	7	116,117	Infoton Write
PIM 2	0	120,121	BIC 2 complete (Camac Block Transfer)
	1	122,123	Camac Error and Block End
	2	124,125	-
	3	126,127	First I/O LAM (Crate station 7)
	4	130,131	First Clock LAM (Crate station 6)
	5	132,133	Second I/O LAM (Crate station 10)
	6	134,135	Second Clock LAM (Crate station 9)
	7	136,137	-
PIM 3	0	140,141	Centronics 101 printer
	1	142,143	Magnetic tape - Motion Complete
	2	144,145	T2 Read (second teletype)
	3	146,147	T2 Write (second teletype)
	4	150,151	Tektronix Read
	5	152,153	Tektronix Write
	6	154,155	OC Read (Operator's teletype)
	7	156,157	OC Write (Operator's teletype)

Note :

A LAM is a flag set by a Camac module.

APPENDIX A5

A5.1 Analog to Digital Converter - Borer Type 1243

The Borer Type 1243 ADC offers a 10 bit resolution on a range of -5 Volts to +5 Volts with a conversion time of 15 microseconds. It has a sample and hold feature that permits very narrow time slot measurements to be made. Front panel facilities include a "Start" input which allows an external signal to be used to initiate the digitising process and a "Scan + 1" output which provides a pulse when a digitised value has been read by the computer. This "Scan + 1" pulse may be used to signal the Borer Type 1704 multiplexer to switch to the next input channel.

A5.2 Multiplexer - Borer Type 1704

The Borer Type 1704 FET multiplexer is a 16 channel switching device designed to allow a large number of independent analog signal lines to be connected as required under program control to a single ADC. It may be operated in two basic modes. In the random access mode, any particular channel may be selected for connection to the ADC analog input. In the scanning mode, the "Scan+1" pulse from the Type 1243 ADC is used to automatically switch to the next channel after a digitised voltage has been read. When the switching is completed (in less than 2 microseconds) the multiplexer produces a pulse on its "Wait/Ready" output line which may be used to initiate a further digitising cycle in the ADC. The multiplexer may also be linked to other Type 1704 multiplexers in a daisy chain fashion to provide access to more than 16 channels.

A5.3 Clock/Timer - Borer Type 1411

The Borer 1411 clock/timer provides the exact time of day and highly accurate timing facilities under software control. Once the time of day has been programmatically set it is available in hours, minutes and seconds with a resolution of 1 millisecond.

For timer applications the module may be used in either a self-repeating mode to inititate a sampling process, at regular intervals or in the command mode under software control. At the end of the timer period both a front panel signal and a LAM are generated. For both clock/timers of the NIPR, the LAM is connected to the computer's

interrupt system via a PIM. The front panel signal may be used to initiate digitising in the Borer Type 1243 ADC via its "Start" input. The basic timer pulse rate is 0,1 milliseconds.

A5.4 Input/Output Register - Borer Type 1031

The Borer Type 1031 Input/Output Register is designed to permit non-Camac instruments to give digital data to a Camac system and be themselves influenced by the system. The facilities provided include a 36 bit input register, a 12 bit output register, an input strobe line and a "Give Data" signal for external instruments. A signal on the input strobe line causes data to be read into the input register and a LAM to be generated. For both of the I/O registers of the NIPR the LAM is connected to the computer's interrupt system. This enables outside instruments to directly signal the computer for attention.

A5.5 Dataway Display Borer Type 1801

The Borer Type 1801 Dataway Display module provides testing and monitoring facilities for a Camac system. It memorizes and displays the latest pattern of dataway signals with LEDs (light emitting diodes) on its front panel. Two operating modes are front-panel switch selectable : Display Mode and On-Line Mode. The display mode is used for monitoring signals on the dataway. In the on-line mode the module only responds when directly addressed. This mode is used for testing the crate stations, as it makes it possible to see exactly what data is being placed on the dataway in both read and write operations.

A5.7 Interface Camac to Varian 73 - Borer Type 2204

The Borer Type 2204 Interface is responsible for providing the function of the branch driver and for interfacing the camac system to the Varian. The Camac command word has 21 bits and the Camac data word has 24 bits, whereas the Varian 73 word length is 16 bits. The interface is responsible for matching the Varian capabilities to those of the Camac system.

A Camac command has the form CNAF where :-

C specifies a crate-register in the interface containing the crate number (or numbers) to be addressed,

N specifies the station number in the crate(s) where the required module is located;

A specifies a sub-address for the function, and

F specifies the function code.

The interface may be used in two data modes. When set by a "Single Word Data" control instruction subsequent data operations will transfer 16 bits of data between the Varian and the Camac. In read operations the Interface truncates the high order 8 bits of the Camac data word and in write operations it pads the high order 8 bits of the Camac data word with zeros. When the interface is set by a "Double Word Data" control instruction; subsequent data operations will transfer 24 bits of data between the Camac and Varian with two computer I/O instructions. The first instruction transfers the 8 high order bits of the Camac word to or from the 8 low order bits of the Varian word. The second instruction transfers the 16 low order bits of the Camac word to or from a full computer word. In write operations the Interface accepts the two portions of the data from the computer and then places the composite data word on the branch highway. In reading it similarly buffers the operation in reverse.

The interface also provides status and interrupt facilities to the computer. There are computer instructions for reading and testing the status of the interface and the crate (or crates in a multicrate system). Interrupts to the computer can be generated by conditions occuring in a crate, in the interface or in external equipment via the "External Interrupt" input on the front panel.

There are two optional facilities which the interface provides. The Auto CNA option enables the same Camac command to be presented to a sequence of Camac modules (in successive crates if necessary) with the highest possible speed and with minimal software requirements. The Block Transfer option (which has already been discussed in chapter 6), enables a number of data transfers to or from a module to take place by repeating a Camac command. The option contains a counter which is set to determine the number of times the command is to be repeated, and when this count is reached an interrupt to the computer is generated.

APPENDIX A6

NIPR Programming Conventions for the CAMAC Equipment

Allocation of CAMAC Facilities

To prevent several programs from trying to use non-shareable Camac modules at the same time, a set of flags is kept in a standard location in memory to indicate which resources are in use and which are available. The VORTEX operating system has allocated locations 2 - 017 in main memory for the use of application programs. In the NIPR conventions words 7, 010, and 011 are used to provide the flags which indicate which facilities have been allocated and which have not. Word 6 is used by all programs to set the interrupt mask in PIM2. This convention is summarised in the following table

Table A6.1 FIXED LOW CORE LOCATIONS.

Label	Address	Function
CØPMSK	6	PIM-2 interrupt mask
CØPIM	7	PIM-2 line allocation mask
CØCAM1	010	Camac modules 17 - 23 + Varian Free Running Counter.
CØCAM2	011	Camac modules 1 - 16 allocation mask.

The meaning of each bit in the words is given in tables A6.2 and A6.3

CØPMSK. This word is used to set the interrupt mask register on PIM-2 by all programs that change the mask. A bit present inhibits the interrupt from the corresponding line; a bit absent, allows interrupts from that line to be presented. A program will change only the bits corresponding to the lines used by it.

CØPIM. This word is used to show which lines on PIM-2 are being used by programs. A bit present implies that the corresponding line is being used. On entry, a program will use this word to check that there is no conflict with any other program's requirements, and will set the bits corresponding to the interrupt lines it uses. On exit it will remove these bits to show that the lines are free.

C∅CAM1 and C∅CAM2

These words are used to indicate which Camac modules are being used, and whether the Varian Free Running Counter is being used. On entry, a program will check that its requirements can be met without conflict, and it will set the bits corresponding to the modules that it uses. On exit it will remove these bits so that the modules may be used by other programs.

CAMAC Initialization.

When the VORTEX system is initialized the allocation mask words 6-011 are zeroed, and this fact is used to determine when it is safe to initialize the Camac system. It is a general requirement that between and including the times of testing a flag (i.e. a bit set in some word) and setting it or performing some action depending on the value of the flag, a program must inhibit all interrupts that may allow control to be passed to another task which could possibly change the status of the system.

The general procedure that must be followed by a program in initializing the Camac system, and/or changing the allocation masks is as follows :

(1) All interrupts must be disabled (i.e. from the PIM's and Clock), and bit 15 of allocation word C∅CAM1 must be checked to make certain that initialization by another program is not in progress.

(2) If an initialization is in progress, interrupts must be enabled, and the program should wait until the busy bit has been reset indicating initialization by the other program has been completed.

(3) If initialization is not blocked, the busy bit must be set, interrupts enabled and the initialization and allocation may continue. Interrupts should be disabled only when necessary, using the considerations in the section "Camac Programming" below.

The purpose of this procedure is twofold :-

(1) It should allow minimal interference with other concurrently running programs.

(2) It allows the use of standard VORTEX I/O facilities for operator communication during the initialization process.

A system subroutine has been provided for initializing the Camac system using the above procedure.

CAMAC Programming

Any Camac instructions that change the system in such a way that might affect other programs or allow other programs to interfere with the current task must be executed with all interrupts masked off. This is because any interrupt may cause another higher priority program to be scheduled.
The sequences of instructions that should not be interrupted include :

FROM	UNTIL
Set Double Word Data	Set Single Word Data
Set Double Word LAM	Set Single Word LAM
Data Transfer Command	Data Read or Written
Test	Sense or Read Status

For example, the following program segment should not be interrupted between lines 2 and 4.

```
1       LDAI    CNAF    CNAF FOR READ
2       OAR     050
3       SEN     050,*   WAIT TILL READY
4       CIA     051     051 INPUT DATA
```

If an interrupt occurred after the OAR instruction in line 2 and another program was dispatched and it issued Camac instructions. the results of the CIA 051 would be unpredictable.

Table A6.2 PIM-2 INTERRUPT ASSIGNMENTS

WORD	BIT	MASK	LINE	LOCATION	DESCRIPTION
CØPMSK	0	1	010	0120	BIC-2 complete
&	1	2	011	0122	CAMAC error, Block End
CØPIM	2	4	012	0124	-
	3	010	013	0126	First I/O LAM
	4	020	014	0130	First Clock LAM
	5	040	015	0132	Second I/O LAM
	6	0100	016	0134	Second Clock LAM
	7	0200	017	0136	-

Table A6.3 CAMAC ALLOCATION MASKS

WORD	BIT	BIT-NAME	BIT-VALUE	MODULE	DESCRIPTION
CØCAM1	15	CØIBSY	0100000		Initialization in progress
	14	CØVTIM	040000		Varian Free Running Counter
	13		020000		Spare
	12	CØEXBE	010000		External Block End
	11	CØEXST	04000		External Start
	10	CØEXSY	02000		External Synch
	9	CØEXIT	01000		External Interrupt
	8	CØEXIN	0400		Inhibit
	7-0			17-24	Not Used
CØCAM2	15-10			16-11	Not Used
	9	CØIOR2	01000	10	I/O Register 2
	8	CØCLK2	0400	9	Clock/Timer 2
	7	-	-	8	Not Used
	6	CØIOR1	0100	7	I/O Register 1
	5	CØCLK1	040	6	Clock/Timer 1
	4	CØADC	020	5	ADC
	3	-	-	4	(Blocked by ADC)
	2	CØMX	04	3	Multiplexer
	1	-	-	2	Not Used
	0	CØDISP	01	1	Dataway Display

APPENDIX A7

Interrupt Handling

The interrupt handler is connected by overlaying the interrupt location in memory with a JMPM (jump and mark) instruction, and enabling the appropriate PIM interrupt line. On entry to the routine, all interrupts (PIMS and the Real Time Clock) are disabled and the A, B, and X registers as well as the overflow status are saved. On exit from the interrupt handling routine, the A, B, and X registers and the overflow status must be restored and the appropriate interrupts enabled. If a background program was interrupted, then PIM, Clock and Memory Protect interrupts must be enabled. If a foreground program or the dispatcher was interrupted, then PIM and Clock interrupts should be enabled. Finally, if the real time clock (RTC) processor was interrupted, then only PIM interrupts should be enabled. When the interrupts have been enabled, control may be returned to the interrupted program at the address which was stored by the JMPM instruction. A program illustrating this is shown below in figure A7.1.

Figure A7.1

```
*  RETURN CONTROL TO INTERRUPTED PROGRAM
*  SET OR NOP ENABLE MEMORY PROTECT AND CLOCK INTERRUPTS
*  AS REQUIRED.

        LDB         NOP             LOAD A NOP INSTRUCTION
        STB         MEMPR           NOP MEM PROTECTION ENABLE
        STB         CLKEN           NOP CLOCK INTERRUPT ENABLE
        LDA         VØCTL
        SUBI        037
        JAZ         RESTRG          SKIP IF RTC PROCESSOR WAS INTERRUPTED
        LDA         ENCLK           LOAD AN "EXC 0147" INSTRUCTION
        STA         CLKEN           SET ENABLE CLOCK INTERRUPT
        LDA         VØCTL
        JAN         RESTRG          SKIP IF DISPATCHER WAS INTERRUPTED
        LDX         VØCTL
        LDA         TBST,X          PICK UP TASK STATUS
        BT          RA1+8,RESTRG    SKIP IF NOT PROTECTED
        LDA         VØLLUP          LOAD LOW BOUND OF UNPROTECTED MEMORY
        SUBE        INTADD
        JAN         RESTRG          SKIP IF RETURN TO UNPROTECTED CODE
        LDAE        INTADD
        SUB         VØLUP           UPPER BOUND OF UNPROTECTED MEMORY
        JAN         RESTRG          SKIP IF RETURN TO UNPROTECTED CODE
        LDA         MEMEN           LOAD "EXC 0645" INSTRUCTION
        STA         MEMPR           ENABLE MEMORY PROTECT INSTRUCTION
```

Figure A7.1 Continued

```
*  RESTORE REGISTERS, OVERFLOW STATUS, ENABLE INTERRUPTS & RETURN
RESTRG      BSS     0
            ROF
            LDA     SOF
            JAZ     *+3
            SOF
            LDA     SA
            IDB     SB
            LDX     SX
MEMPR       BSS     1       ENABLE MEM PROTECT OR NOP
CLKEN       BSS     1       ENABLE CLOCK INTERRUPTS OR NOP
            EXC     0244    ENABLE PIMS
            JMP*    INTADD  RETURN TO INTERRUPTED PROGRAM
            .       .
            .       .
            .       .
MEMEN       EXC     0645    ENABLE MEMORY PROTECT
ENCLK       EXC     0147    ENABLE CLOCK INTERRUPTS
*  INTERRUPT HANDLER
INTADD      ENTR            ENTRY POINT
            EXC     0747    DISABLE CLOCK INTERRUPTS
            EXC     0444    DISABLE PIM INTERRUPTS
            STA     SA      SAVE REGS
            STB     SB
            STX     SX
            TZA             SAVE OVERFLOW STATUS
            JOFN    *+3
            LDAI    0100000
            STA     SOF
*  READ AND/OR PROCESS DATA
            .       .
            .       .
            .       .
```

A P P E N D I X A8

Glossary of Data Processing Terminology Used in this Report

Some of the terms that appear in this glossary are terms that, either have a wide range of meaning in the data processing literature and are used with restricted meaning in this report, or they have been specially adopted for use here to avoid more cumbersome expressions and repetitious qualifications. Wherever possible, the definitions made here closely follow those in the "Glossary of Computing Terminology" by C.L. Meek, published by CCM Information Corporation, New York, 1972.

ADC. (See appendix A5, section A5.1) Analog to Digital Converter; it accepts analog voltages as input and produces equivalent digital values in a form acceptable to digital computers.

ANALOG DATA. Data represented in a continuous form as contrasted with digital data represented in a discrete form. Analog data usually represents physical variables as voltages.

ALONE. The term is used in this report to refer to a program that is run under the VORTEX operating system with no competing programs. For example, the psychological experimenter is said to run his time-sharing program alone if there are no other users of the machine even though he uses the VORTEX operating system to control peripheral devices, provide operator facilities and/or load programs and overlay subroutines. The VORTEX programs that provide these facilities are regarded as cooperating programs.

BIC. (See section 6.1) The Buffer Interlace Controller is an option of the Varian 73 computer which allows blocks of data to be transferred between the computer memory and a peripheral by means of a cycle stealing technique without involving the use of the CPU.

CAMAC TIMER/VARIAN TIMER. The timing facility provided by the Borer type 1411 Clock/Timer (see appendix 5 Section A5.3) is referred to as the Camac Timer while the Free Running Counter of the Varian Real Time Clock is referred to as the Varian Timer.

CPU. Central Processing Unit is the portion of the computer that consists of the arithmetic unit and the control unit that decodes and executes instructions.

DRIVER. A program of the operating system used to control peripheral devices such as disks, tapes etc.

ECG. Electrocardiogram; a record of the electrical potentials generated by the heart and obtained from electrodes on the body surface.

EEG. Electroencephalogram; a record of the electrical activity of the brain and obtained from electrodes placed on the scalp.

FOREGROUND/BACKGROUND. These two terms normally contrast time-sharing programs with batch programs, but under the VORTEX operating system they have a slightly different meaning. Foreground and background programs are all multiprogrammed, but foreground programs have absolute priority over background programs. Foreground programs are protected from background programs by the memory protection option and background programs cannot issue I/O instructions which may hold up foreground programs, nor can they execute a HALT instruction. A background program may be checkpointed and copied to disk by the VORTEX operating system if a foreground program requires the main memory it is occupying and, finally, only one background program is permitted while multiple foreground programs are allowed. Thus background is provided specifically for program development and system housekeeping such as file maintenance. The VORTEX supplied compilers, assembler and file management facilities all run in the background, but the file editor written by the NIPR can be, and usually is run in the foreground.

INTERRUPT. (See section 6.1). A break in the normal processing of a program occurring in such a way that the flow can be resumed from that point at a later time. Interrupts can be initiated by signals originating within the computer system to synchronize the operations of various components, or they can be initiated by signals exterior to the computer to synchronize the operation of the computer system with the outside world.

MULTIPLEX. The process of transferring data from several devices operating at relatively low transfer rates to a device operating at a high transfer rate by interleaving the data in such a manner that the high speed device is not obliged to wait for a low speed device.

MULTIPLEXER. (See appendix A5, section A5.2) A computer controlled analog switch that can connect selected analog inputs to a single point, thus making it possible for multiple analog inputs to share a single ADC.

MULTIPROGRAMMING. The procedure by which two or more independent programs can share use of the central processer. It is the concept of multiplexing the use of the computer applied at the program level in contrast to time-sharing which is multiplexing the use of the computer at the user's level. The programmer is able to write his programs as if they will run alone in the machine. However, it was shown in chapter 7 that some care has to be taken to avoid conflicts between programs using peripherals.

NON-REAL-TIME. Is used to refer to those programs (both foreground and background) that do not have critical response times. It is thus a generic term for the assemblies, compilations, program development and testing activities, and data analyses that use stored or recoverable data as opposed to the live and non-recoverable data used by real-time programs.

OPERATING SYSTEM. The monitor executive system which :

(1) controls the order in which programs run and provides for transition between one program and another, (2) provides high level, device independent, input/output facilities and (3) controls one or more language processors, with file maintenance and other facilities.

PIM. (See section 6.1) The Priority Interrupt Module is an option of the Varian 73 which allows peripheral devices or external equipment to signal their state by generating an interrupt to the computer.

REAL-TIME. This is the characteristic of a data processing operation which is run in synchronization with an external physical process with special emphasis on the lack of control of input rates and the requirement that the program must receive, transmit and/or process all the real-time data before the next batch is ready. At the NIPR this requirement is receiving, processing and storing a batch of data and re-initializing within 1 millisecond.

TIME-SHARING. Is the facility whereby two or more people can simultaneously use the computing system independently of each other. In the NIPR context this means that there may be up to four users at as many terminals and each user is not affected by the other users and does not affect them (except in terms of perceived speed and memory capacity). Multiprogramming is the means by which time-sharing is achieved on the Varian 73 computer.