

https://dspace.org/

The implementation of a data acquisition and dissemination system using a micro-computer as a intelligent front-end processor to a multi-user mini computer

Chapman, Mark Jonathon

2014-12-15
http://hdl.handle.net/10539/15983

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

THE IMPLEMENTATION OF A DATA ACQUISITION AND DISSEMINATION SYSTEM USING A MICROCOMPUTER AS AN INTELLIGENT FRONT-END PROCESSOR TO A MULTI-USER MINICOMPUTER.

Mark Jonathon Chapman

A dissertation submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg as a partial requirement for the Degree of Master of Science in Engineering.

Johannesburg, 1984.

ABSTRACT

This dissertation describes the design, development and implementation of the software for a real-time data acquisition system. A microcomputer was employed as an intelligent front-end processor to a multiuser minicomputer system for the real-time input and output of analog signals.

The data acquisition system, BADEDAS (Basic Analog and Digital Experimental Data Acquisition System), provides the users of an Eclipse minicomputer system with an efficient means of acquiring digital samples of analog signals and disseminating digital samples as analog signals.

BADEDAS exploits the concept of device independent I/O to achieve an othogonal and flexible architecture.

BADEDAS provides the choice of two user interfaces to the system and the ability to include simple user-written processing routines in the front-end processor's control software.

The use of a microcomputer as a front-end processor removes the real-time I/O requirements from the minicomputer and also reduces the risk electrical damage to it.

DECLARATIOI

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

mJ Chgan

27th day of JANUARY, 1984

DEDICATION

This dissertation is dedicated to Hildegard and Carl.

CONTENTS

CHAPTER 1 INTRODUCTION

1.1 SCOPE 1
1.2 TERMINOLOGY 2
1.3 REASON FOR UNDERTAKING THE PROJECT 2
1.4 BACKGROUND TO THE PROJECT 2
1.5 AIM OF THE PROJECT 4
1.6 PREVIOUS WORK IN THE FIELD 5
1.7 SURVEY OF RELATED LITERATURE: 6

CHAPTER 2 BADEDAS ENVIRONMENT

2.1 HARDWARE ENVIRONMENT 7
2.2 SOFTWARE ENVIRONMENT 9
2.3 SOFTWARE ORGANISATION 9

CHAPTER 3 DESCRIPTION OF SOFTWARE

3.1 DESIGN OVERVIEW 13
3.2 BADEDAS MNEMONICS AND INTERNAL CODES 16
3.3 BADEDAS-E 19
3.4 USER INTERFACES 19
3.4.1 DOWN-LINE LOADING 23
3.5 BADEDAS-M 24
3.5.1 KERNEL 24
3.5.1.1 Interrupts: 25
3.5.1.2 Memory Management: 25
3.5.1.3 Command Interpreter: 28
3.5.1.4 User Routine Scheduling: 29
3.5.1.5 Message Logging: 29
3.5.1.6 Timeouts: 29
3.5.1.7 Wake-ups: 30
3.5.1.8 Traps: 30
3.5.2 I/O DEVICE DRIVERS 31
3.5.2.1 A/D And D/A Device Drivers: 31
3.5.2.2 Host Communication Driver: 32
3.5.2.3 Adding Extra Device Drivers: 33
3.5.3 USER ROUTINES 33
3.6 COMMUNICATION PROTOCOL: 34

CHAPTER 4 PROJECT PROCEDURE

4.1 OVERALL APPROACH 37
4.2 SYSTEM DEFINITION 37
4.3 SYSTEM DESIGN 38
4.4 PROGRAMMING 41
4.4.1 GENERAL 41
4.4.2 MICRONOVA ASSEMBLER 42
4.5 TESTING AND INTEGRATION 43

4.6 ACCEPTANCE TESTING 43
4.6.1 EXPERIMENT 1 45
4.6.2 EXPERIMENT 2 46
4.6.3 EXPERIMENT C 46
4.6.4 EXPERIMENT D 47
4.6.5 OTHER EXPERIMENTS 48

CHAPTER 5 CONCLUSIONS

5.1 LIMITATIONS 49
5.2 PROBLEMS 50
5.3 CONCLUSION 51

APPENDIX A RTL/2

A.1 SCOPE 54
A.2 BACKGROUND 54
A.3 THE SUITABILITY OF RTL/2 FOR THE IMPLEMENTATION OF BADEDAS 55
A.4 THE ATTEMPT TO CONVERT RDOS RTL/2 TO AOS RTL/2 . . 56

APPENDIX B PROTOCOL

B.1 MULTIBYTE PACKETS 59
B.2 SINGLE BYTE PACKETS 60
B.2.1 ACK0, ACK1 And NAK 61
B.2.2 SYNC And SACK 61
B.2.3 ENQ 62
B.2.4 XON And XOFF 62
B.3 TIMEOUTS 62

APPENDIX C EXAMPLE OF BADEDAS USER ROUTINE

APPENDIX D BADEDAS FORTRAN EXAMPLE

LIST OF TABLES

TABLE 1: BADEDAS mnemonic and internal code table 18
TABLE 2: BADEDAS command summary 19
TABLE 3: BADEDAS Fortran 5 call summary 21

LIST OF FIGURES

FIG 1: Hardware Configuration 8
FIG 2A: Software Configuration - Command Interface 11
FIG 2B: Software Configuration - Fortran 5 12
FIG 3: Conceptual Model 15
FIG 4: Pipe Conceptual Model 17
FIG 5A: BADEDAS-E Fortran-5 Interface 22
FIG 5B: BADEDAS-E Command Interface 22
FIG 6: BADEDAS-E Functional Breakdown 39
FIG 7: Memory Management System Functional Breakdown 40
FIG 8: Set-up For Acceptance Testing 44

CHAPTER 1

INTRODUCTION

1.1 SCOPE

This document aims to provide an overall description of the BADEDAS system and the procedure followed in creating it.

The document consists of five chapters:

* Chapter 1, The Introduction, serves to familiarise the reader with the reason for doing the project, the aim of the project and the background to the project.

* Chapter 2, BADEDAS Environment, describes the hardware and software environment within which BADEDAS must function.

* Chapter 3, Description of Software, provides a functional description of BADEDAS.

* Chapter 4, Project Procedure, outlines the procedure followed for the project.

* chapter 5, Conclusions, concludes the document by discussing the limitations of BADEDAS, the problems experienced during the project and summarising the results obtained.

Other BADEDAS documents are:

* The BADEDAS User's Manual - the instructions for using BADEDAS.

* The BADEDAS Technical Manual - a detailed technical description of the BADEDAS software.

* The BADEDAS Software Listings - the source code of all the BADEDAS software.

1.2 TERMINOLOGY

BADEDAS is a system for the acquisition and dissemination of data. Data acquisition, meaning the gathering of data from various places, is a familiar term in the computer world. Data dissemination is a less familiar term and its intended meaning here is the distribution or dispersion of data to various places. In other words, data dissemination can be thought of as the opposite process to data acquisition.

1.3 REASON FOR UNDERTAKING THE PROJECT

The Department of Electrical Engineering offers a two year full-time MSc(Eng.) degree programme in Computer Engineering. This programme is aimed specifically at the Computer Science graduate who wishes to expand his knowledge of the electrical engineering aspects of computers. The curriculum includes course work as well as a minor project in the first year of study and a major project in the second year. This dissertation describes a project undertaken in the latter category.

1.4 BACKGROUND TO THE PROJECT

The Electrical Engineering Department has two multiuser computer systems in its computer laboratory. One of these is a Data General Eclipse S/140 which runs under the control of AOS (Advanced Operating System).

The Eclipse is a powerful 16 bit minicomputer with a half a megabyte of memory, 25 megabytes of online disk storage and eight user consoles, while AOS is a multiprogramming operating system which is most suitable for software development and interactive scientific problem solving.

Although a large library of digital signal processing software is available to users of the Eclipse system, there has, in the past, been no convenient means for the real-time sampling of analog signals as input data for this software or for the real-time output of processed analog signals.

AOS, due to its multiprogramming nature cannot guarantee a short response time to all interrupts and is therefore not suitable for conducting real-time data acquisition and dissemination for users. In addition to this, the value and importance of the Eclipse to the department make it undesirable to permit the

relatively uncontrolled input and output of electrical signals to and from the Eclipse's chassis. Thus, it was decided to use a microcomputer as a dedicated front-end processor to the Eclipse for handling the input and output of analog signals.

The instruction set of the Micronova is (with a few exceptions) a subset of that of the Eclipse, thus making it possible for software to be developed on the Eclipse for execution on the Micronova. As there was a requirement that the front-end processor's software be configured on the Eclipse, the Micronova was an obvious choice for the front-end processor.

1.5 AIM OF THE PROJECT

In general, the aim of this project was to develope a data acquisition and dissemination system, which allows the user to benefit from the responsiveness of a dedicated Micronova front-end processor as well as the processing power, data management and data storage capabilities of the Eclipse.

In order to achieve this aim it was necessary to develope a small, efficient and dedicated software package on the Eclipse which could be down-line loaded into the Micronova. The package has the ability to conduct the input and output of analog signals in real-time, using the Micronova's high speed analog to digital and digital to analog converters. Digital samples of analog signals can be routed to the Eclipse for storage or processing and processed digital samples can be routed from the Eclipse to the front-end processor for output as analog signals. Provision has also been made to enable simple user-written routines to be incorporated into this software package for performing a limited level of processing of digital samples within the front-end.

In order to make the package easily accessable to the users of the Eclipse, a software interface to the front-end processor, which runs on the Eclipse, had to be developed. In addition to this it was necessary to develop an efficient protocol for communication between the Eclipse and the Micronova.

1.6 PREVIOUS WORK IN THE FIELD

Perseus Computing and Automation (Pty) Ltd have developed a software package for communication in a multiprocess, multiprocessor and multitasking environment. Called ETETCCP (End-To-End Task Communication Package), it was designed specifically for use in real-time process control applications involving an Eclipse, as host, front-ended by a multitude of Micronovas (Ying and Currer, 1978).

An attempt to run this package on the Department's Eclipse computer, front-ended by a single Micronova, met with limited success (Thompson 1981). Data transferred from the Micronova to the Eclipse was frequently lost and the console response time for other Eclipse users was noticably degraded whenever communication between the Micronova and Eclipse took place. To overcome these problems the baud rate of the link between the two processors had to be made as low as 300 baud.

A case study described by Ying and Currer (1978) involved the use of ETETCCP on an Eclipse S230 front-ended via a data control unit (DCU) by 30 Micronovas. The use of a DCU quite drastically reduces communication overheads for the CPU. Without a DCU, the CPU is interrupted for each byte of a message sent or received. With a DCU the I/O shifts from character to message level. The overheads of sending or receiving the characters of the message are off-loaded onto the DCU. The absence of a DCU in the department's Eclipse system could have been to some degree responsible for the low performance achieved.

1.7 SURVEY OF RELATED LITERATURE:

Prucnal (1982) outlines the application of a system similar to BADEDAS. This paper provides a concise description of a first course in digital signal processing offered to students at Columbia University, USA. The course includes a number of laboratory experiments using a PDP 11. Real-time wave sampling and wave output are achieved using A/D and D/A converters installed in the PDP 11's mainframe. Sampling rates used are low, but the author mentions that the system is soon to be expanded using a dedicated microprocessor interface (ie. a front-end processor) to enhance performance.

Cocanet (Rowe and Birman, 1982), a local network developed at the University of California, Berkley, extends some of the features of the Unix operating system for inter-computer communication. The design of BADEDAS was influenced by Unix pipe and filter concepts described in this paper.

Taemond and De Bruyn (1982) describe a system in which a microcomputer is used as a front-end processor for automatic measurement in an LSI design laboratory. Although the application is different to that proposed for BADEDAS, the system is of a similar configuration. This brief paper provided a good insight to such a system.

1.7 SURVEY OF RELATED LITERATURE:

Prucnal (1982) outlines the application of a system similar to BADEDAS. This paper provides a concise description of a first course in digital signal processing offered to students at Columbia University, USA. The course includes a number of laboratory experiments using a PDP 11. Real-time wave sampling and wave output are achieved using A/D and D/A converters installed in the PDP 11's mainframe. Sampling rates used are low, but the author mentions that the system is soon to be expanded using a dedicated microprocessor interface (ie. a front-end processor) to enhance performance.

Cocanet (Rowe and Birman, 1982), a local network developed at the University of California, Berkley, extends some of the features of the Unix operating system for inter-computer communication. The design of BADEDAS was influenced by Unix pipe and filter concepts described in this paper.

Taemond and De Bruyn (1982) describe a system in which a microcomputer is used as a front-end processor for automatic measurement in an LSI design laboratory. Although the application is different to that proposed for BADEDAS, the system is of a similar configuration. This brief paper provided a good insight to such a system.

CHAPTER 2

BADEDAS ENVIRONMENT

2.1 HARDWARE ENVIRONMENT

The BADEDAS hardware configuration was fixed on conception of the project by the availability of suitable hardware within the department and the software compatibility of that hardware.

The hardware necessary to support BADEDAS consists of an Eclipse minicomputer, with at least one user console, front-ended by a Micronova microcomputer. Figure 1 shows the BADEDAS hardware configuration.

BADEDAS makes use of two of the Eclipse's console ports, one for the link to the Micronova and the other for the user console. The Eclipse user console is the only point from which the user interacts with BADEDAS and, apart from the set up procedure of connecting the external clocks and the analog input and output leads, as required by the application, and switching the Micronova on, there is no further user action required for the operation of BADEDAS.

The link between the Eclipse and the Micronova is a full-duplex asynchronous serial link which may run at up to 9600 baud. At the Micronova, this link must be connected to the first teletype port TT0.

BADEDAS occupies just over 3 kilobytes of memory so at least 4 kilowords of memory are required in the 16 bit Micronova in order for it to run. The Micronova second teletype port, TT1, is used for the logging of error and status messages. This port, which may be attached to any display or hard copy terminal with an RS232 or 20 milliamp current loop interface, is used purely as an output port and any input to it is ignored. One reason for having a message logging device separate from the

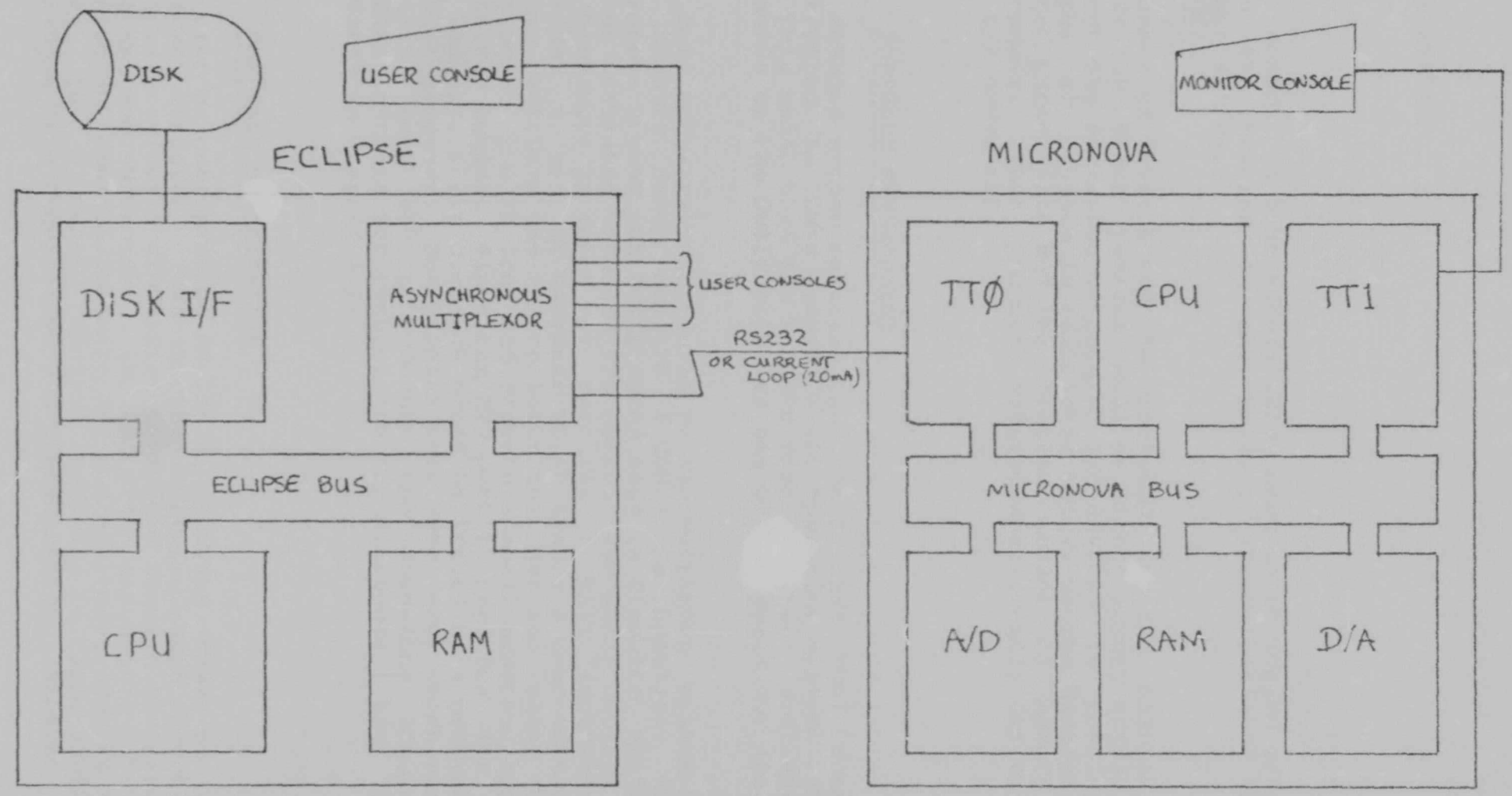


FIGURE 1 : HARDWARE CONFIGURATION

user console is that events which need to be logged often occur when the program on the Eclipse which initiated them is no longer active.

A number of devices may be included in the BADEDAS system. Typically these devices will be direct memory access devices, though the inclusion of program I/O devices is possible. The number of devices is restricted mainly by the desired level of system performance and the limited amount of memory in the Micronova. This project catered specifically for one A/D and one D/A converter.

2.2 SOFTWARE ENVIRONMENT

The BADEDAS system requires that the Eclipse host computer be controlled by Data General's AOS operating system. The reason for this being that the BADEDAS user interface software, which executes on the Eclipse, makes use of AOS calls for its I/O.

The only software requirement on the Micronova system is that the console debug monitor (CDM) is installed. CDM is the Micronova system monitor program held in firmware which is an option available with the TTO board. CDM gains control whenever the Micronova is reset or executes a halt instruction. It provides a set of commands which enable a user to examine and change registers and memory locations, set and clear a single breakpoint and to load and execute a small program, typically a bootstrap loader. Since the TTO port is used for the link to the Eclipse, these commands could be issued by a program running on that computer. The BADEDAS user interface, which runs on the Eclipse, uses CDM to conduct the down-line loading of the BADEDAS software for controlling the Micronova (see "Down-line Loading" in Chapter 3).

2.3 SOFTWARE ORGANISATION

The user or user program employing BADEDAS does so from the Eclipse computer. The BADEDAS software which executes on the Eclipse provides a simplified interface between the user or user program and Micronova, it is also responsible for down-line loading the control software for the Micronova from an AOS disk file. The Micronova control software is responsible for all real-time data acquisition and dissemination. BADEDAS software

can therefore be classified according to the CPU which executes it. BADEDAS software which executes on the Eclipse is called BADEDAS-E and that which executes on the Micronova is called BADEDAS-M (see figures 2A and 2B).

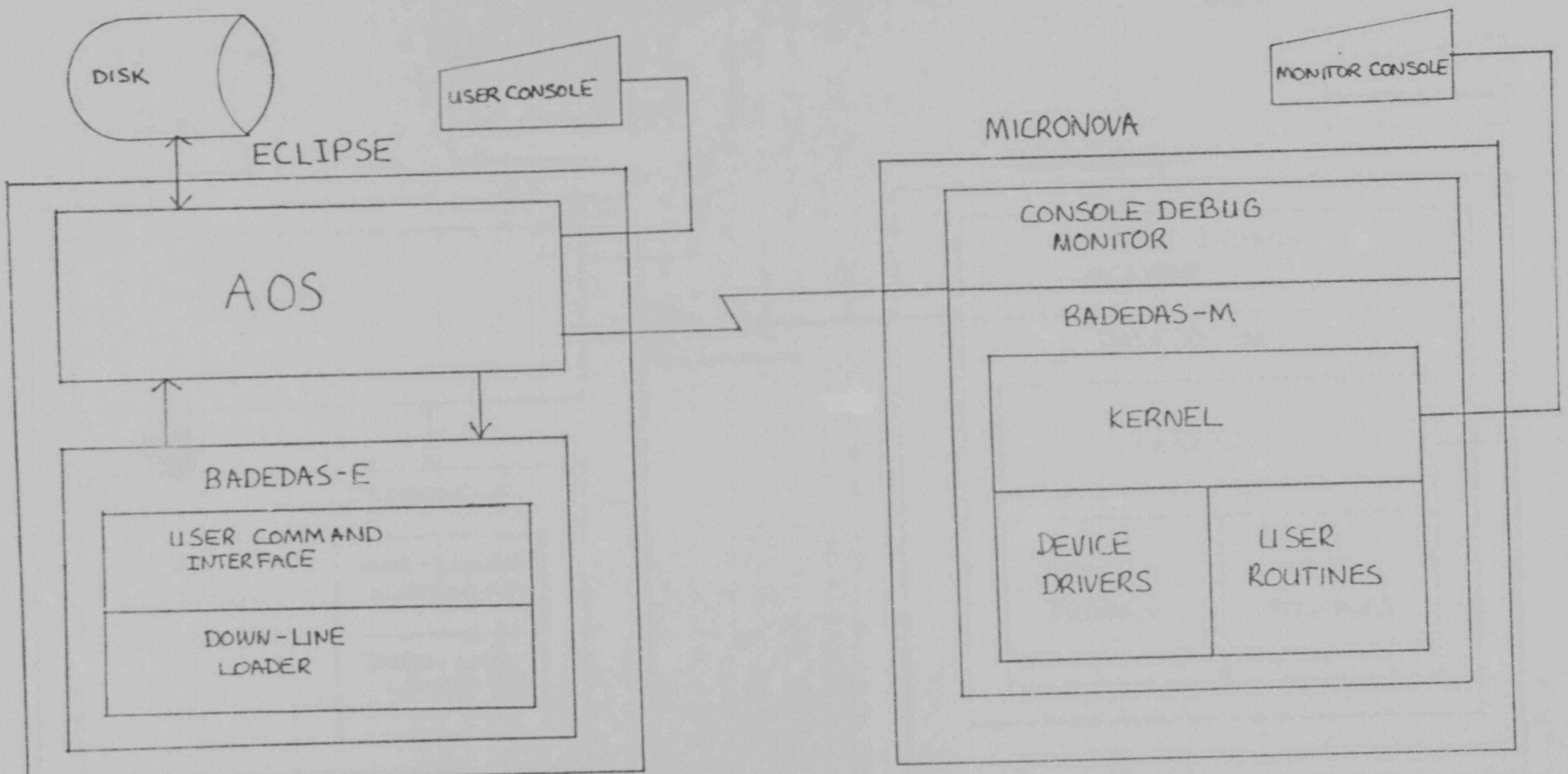


FIGURE 2A: SOFTWARE CONFIGURATION – COMMAND INTERFACE

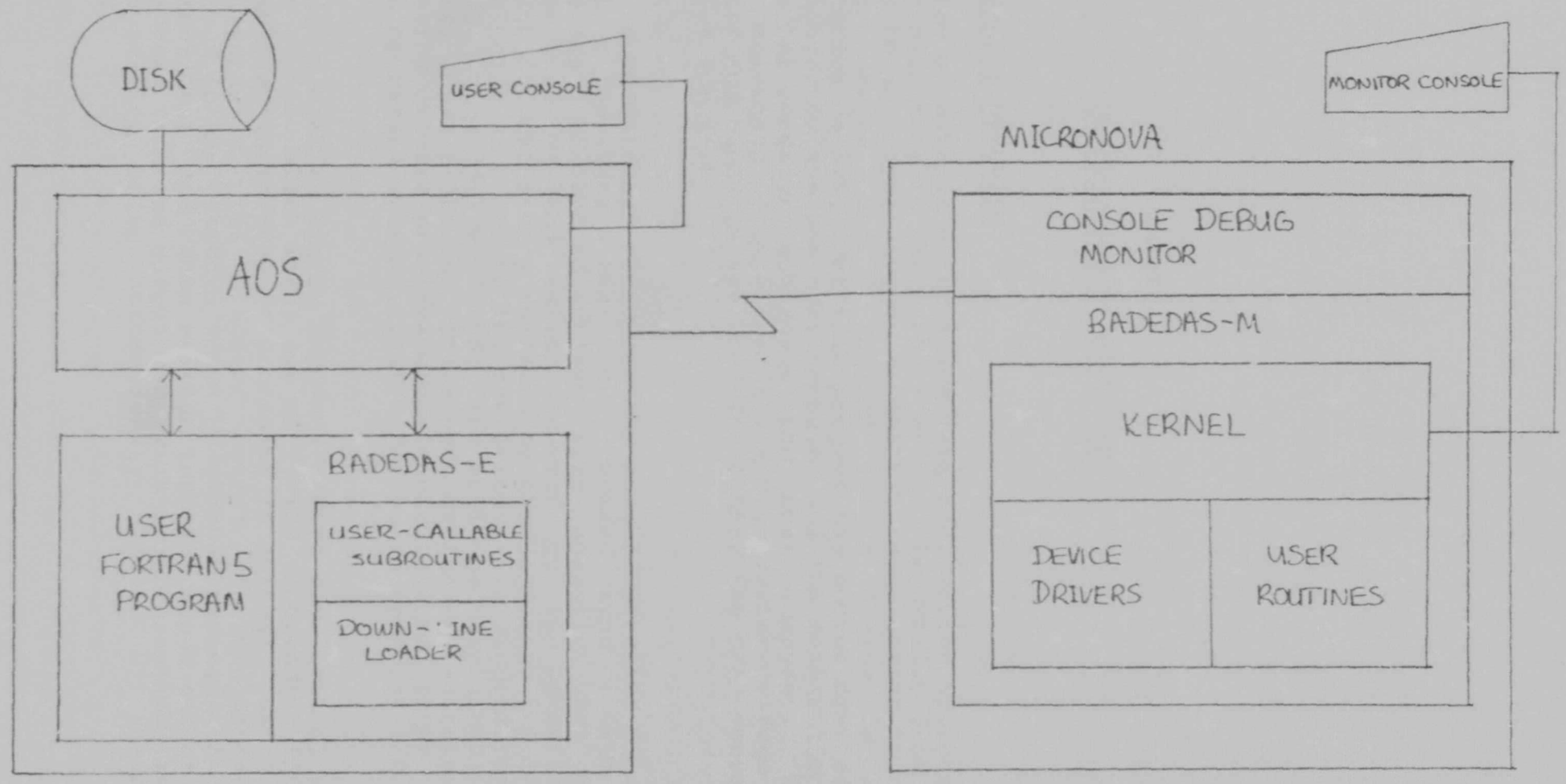


FIGURE 2B: SOFTWARE CONFIGURATION - FORTRAN 5 INTERFACE

CHAPTER 3

DESCRIPTION OF SOFTWARE

3.1 DESIGN OVERVIEW

The flow of data in a front-end data acquisition processor, such as the Micronova in the BADEDAS system, is usually from an input device to an output device. A BADEDAS input device is called a source device and a BADEDAS output device is called a destination device. For this project the source devices were an A/D converter and the host computer and the destination devices were a D/A converter and again the host computer. So, data could originate from either the A/D converter or from the host computer and could be destined for either the D/A converter or the host computer.

In order to maintain orthogonality in the operation of a system, it is important that one command type (with varying parameters) be used to achieve each specific user operation irrespective of the units affected by the operation. This approach is most simply followed by standardising the interfaces to the units affected by the operations. In BADEDAS these units are I/O device drivers and, as will be explained in what follows, user routines. The standardising of the interfaces to the device drivers provides a desirable feature, found in most modern operating systems, called device independent I/O.

The Unix operating system supports a data flow control mechanism called a pipe, which makes use of device independent I/O and the standardised interfaces of processes in general. Using the Unix pipe mechanism, the standard output of one process may be connected to the standard input of another process. A pipe is unidirectional and data written into the input end of the pipe may be read from the output end, with sychronisation, scheduling and buffering being automatically handled by the operating system.

In the above description of a Unix pipe, the processes referred to can be device drivers, so data can be routed from an input device to an output device.

The clear approach to the use of standardised interfaces for the flow of data between processes described in the Unix pipe definition, together with the descriptive nature of its name, prompted the naming of the similar, albeit simpler, mechanism in BADEDAS as a BADEDAS pipe. Using a BADEDAS pipe command, the standard output of a source device driver may be connected to the standard input of a destination device driver (see BADEDAS Conceptual Model, Figure 3).

The I/O devices used for data acquisition and data dissemination, the A/D and the D/A converters, are most efficiently operated in data channel mode (using direct memory access). In this mode of operation, these devices input or output complete data blocks without needing program intervention. In order to exploit this powerful mode of operation, it was decided to make the primary unit of data in BADEDAS the data block.

A data block is simply a collection of data which occupies a contiguous set of memory locations called a memory block. A memory block is preceded by a header and followed by a trailer. The header and trailer contain information regarding the size of the block and whether it is in use, if in use they also contain information about the amount of data in the block and the device that put it there. A detailed description of the memory block is given in the description of the memory management system below. The format of data within a data block is not restricted by BADEDAS. If the data format of an input device does not suit that of an output device, the user has the facility to transform it to a suitable format (see "User Routines" in this chapter). The data formats of the devices used in this project are compatible.

A memory block is of the size required by the source device driver which inputs data to it. In BADEDAS, the data occupying a memory block is never moved in memory, and tasks which need to access that data are passed the memory block address. When the data occupying a memory block has been output, the memory block is put back into a memory pool for re-use.

An alternative to this approach would have been for each device driver to be allocated a buffer of fixed size and location. Using this approach, in order for data, input by one device, to

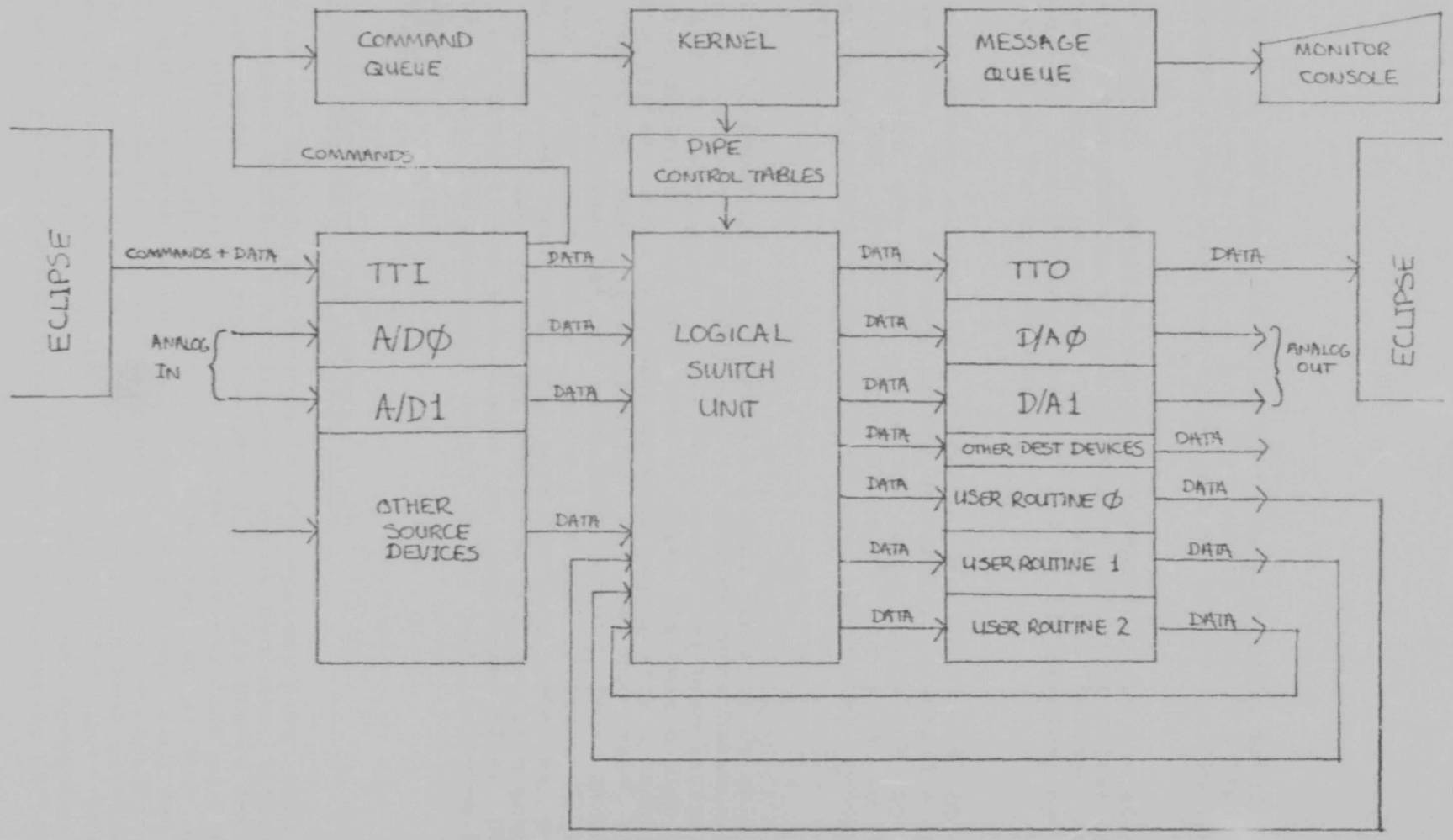


FIGURE 3: BADEDAS CONCEPTUAL MODEL

be output by another, the data has to be physically moved from the source device's buffer to the destination device's buffer. This approach makes inefficient use of both memory and CPU time.

Another possible alternative would have been to allow the source and destination devices to share a common ring-type buffer each with its own buffer pointer. This approach is, however, not suitable for data channel operation and could therefore not be as efficient as the approach taken.

All device drivers either input or output data in blocks of of variable size. When a source device driver has assembled a data block it places the block into the pipe to which it is connected. The block "flows through" the pipe to the destination device driver at the other end, which then proceeds to output it.

A functional requirement of BADEDAS was that the user be able to include a routine for processing of data en-route between its source and its destination. This requirement has been satisfied by further exploiting the Unix pipe concept. In Unix a pipe may include a filter, which is a process which reads from a standard input, performs some transformation on the data read, then writes the transformed data to a standard output. In a BADEDAS pipe the counterpart of a filter is a user routine. A user routine, which has been named in a pipe command, receives each block of data when it becomes available from the pipe source device, processes it and then places the transformed block back into the pipe for subsequent output by the destination device (see figure 4).

3.2 BADEDAS MNEMONICS AND INTERNAL CODES

All source and destination devices in the system have both a three character BADEDAS mnemonic and a single character internal code (see table 1). A pipe is identified by its source device and all blocks passing through a pipe bear the code of their source device as identification. A pipe may only have one source and one destination device and, in general, each device may only be designated to one pipe at a time. An exception to this is the Eclipse host computer, which has been allocated three source device codes (see table 1 "E", "F" and "G"). BADEDAS has been designed to allow the concurrent operation of pipes, so in a system containing additional destination devices,

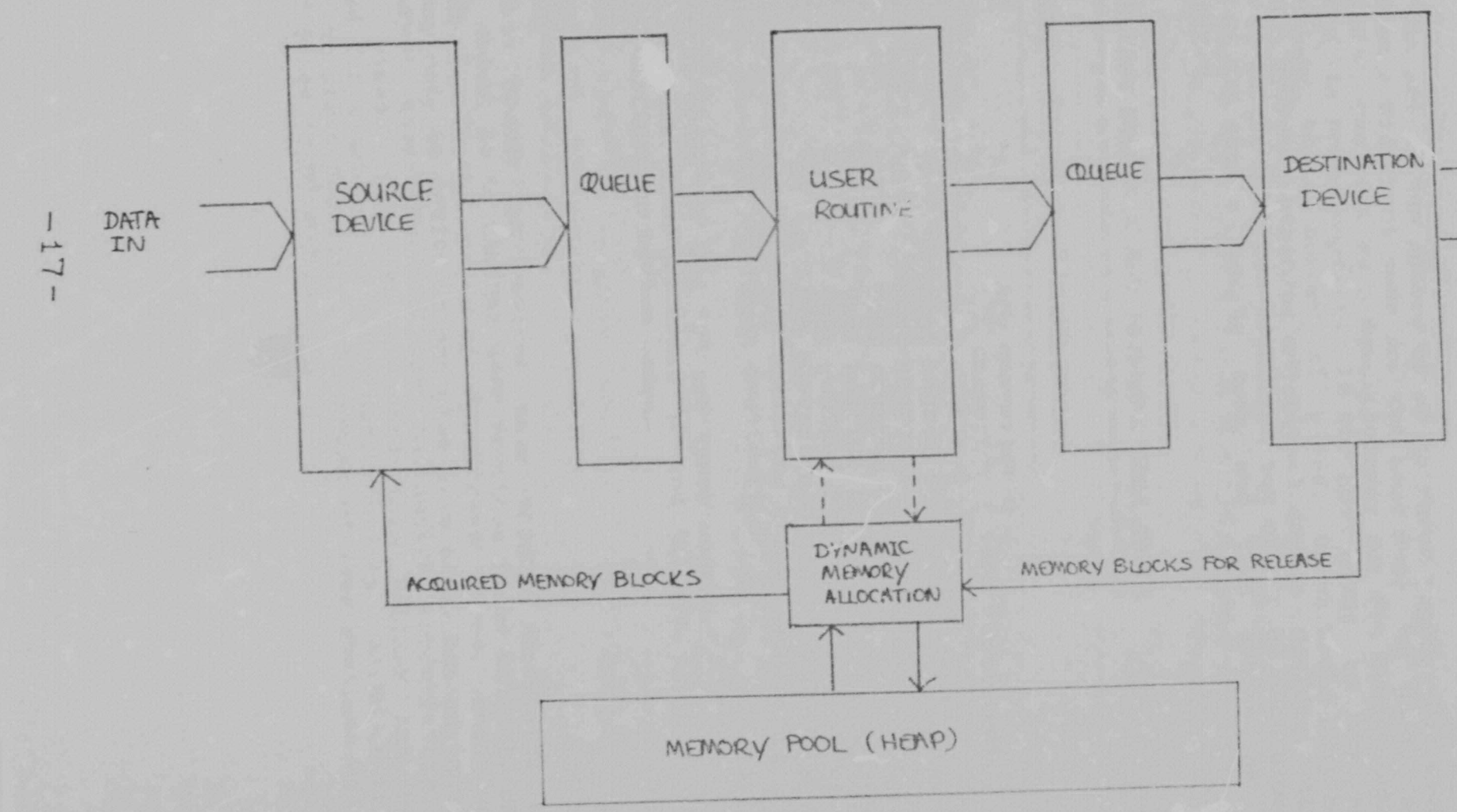


FIGURE 4: PADEDAS PIPE CONCEPTUAL MODEL

the host can act as the source of up to three pipes. For each of these pipes a different code for the host must be used, so that BADEDAS-M can act as a demultiplexer for the data blocks received from the host computer. If the host needs to act as the destination to a number of pipes, which would only be possible in a system containing additional source devices, the user program in the host can establish the origin of each block by looking at its source device code and then acting as a demultiplexer.

TABLE 1: BADEDAS MNEMONIC AND INTERNAL CODE TABLE

MNEMONIC	INT CODE	DESCRIPTION
AD0	"A"	A/D converter 0
AD1	"B"	A/D converter 1 (not installed)
HS0	"E"	Host source 0
HS1	"F"	Host source 1
HS2	"G"	Host source 2
UR0	"H"	User routine 0
UR1	"I"	User routine 1
UR2	"J"	User routine 2
DA0	"K"	D/A converter 0
DA1	"L"	D/A converter 1 (not installed)
HD0	"O"	Host destination

NOTE: Internal codes "C" and "D" are spare source device codes. Internal codes "M" and "N" are spare destination device codes.

In the same manner as devices have BADEDAS mnemonics and internal codes, so do BADEDAS user routines (see table 1, UR0, UR1 and UR2). In the case of concurrent pipes, each user routine may only be active in one pipe at a time, but any number of different user routines may be active concurrently in separate pipes. In its standard form BADEDAS has been configured for a maximum of three user routines. Although it is unlikely to prove necessary, the number of user routines can be increased (see "USER ROUTINES").

3.3 BADEDAS-E

BADEDAS-E is the set of all BADEDAS software, which executes on the Eclipse. This software conducts the down-line loading of the front-end processor and provides the user interface to it.

3.4 USER INTERFACES

The user controls BADEDAS using a simple set of commands entered from an Eclipse user console or by calling BADEDAS subroutines from a Fortran 5 program.

The four BADEDAS commands are summarised in table 2. Devices and user routines are always referenced in these commands by way of their three character BADEDAS mnemonic (see table 1).

TABLE 2: BADEDAS COMMAND SUMMARY

COMMAND	DESCRIPTION
X BAD/I	Initialise a BADEDAS device or user routine eg. X BAD/I AD0,4096
X BAD/P	Set up and start a BADEDAS pipe eg. X BAD/P AD0,UR0,FILE.DAT,3
X BAD/T	Terminate a BADEDAS pipe eg. X BAD/T AD0
X BAD/R	Reset BADEDAS eg. X BAD/R

The initialise command is used to set up the device drivers and user routines. Examples of this would be to set the size of blocks to be input by a source device or to set the values of the arguments for a transformation to be performed by a user routine. The example shown for the initialise command in table 2 sets the input blocksize of the A/D converter to 4096 words.

The pipe command is used to establish a BADEDAS pipe between a source and a destination device and to start the flow of data blocks along that pipe. The pipe command may include the mnemonic of a user routine which is to perform some sort of processing on the data blocks passing through the pipe.

Author Chapman M J
Name of thesis The implementation of a data acquisition and dissemination system using a microcomputer as an intelligent front-end processor to a multi-user minicomputer 1984

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2013

LEGAL NOTICES:

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

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

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