MINICOMPUTER UTILIZATION FOR DATA ACQUISITION AND PROCESSING

Philip Neal Hatfield

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MINICOMPUTER UTILIZATION FOR DATA ACQUISITION AND PROCESSING

by

Philip Neal Hatfield

Thesis Advisor: L. V. Schmidt

September 1972

Minicomputer Utilization for Data

Acquisition and Processing

by

Philip Neal Hatfield
Lieutenant Commander, United States Navy
B.S., Purdue University, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1972

ABSTRACT

An investigation is conducted of minicomputer utilization for data acquisition and/or processing functions. The study is directed primarily toward the suitability of the on-board Data 620 minicomputer system for performance of these functions. System configuration and a listing of available software are included in addition to applicable system operating procedures. Finally, illustrative I/0 software routines required to service a proposed data processing task are presented.

TABLE OF CONTENTS

I. INTRODUCTION

The purpose of this study is to investigate the feasibility of minicomputer utilization for data collection and/or processing functions applicable to the Aeronautics Department at the Naval Postgraduate School. Since the Data 620 minicomputer system is on-board and not being utilized, the study is directed primarily toward the suitability of this system for performance of the above functions.

The Data 620 system was received in 1967 and has been inactive for a considerable period of time (cf Section II). As a result, numerous uncertainties existed relating to actual system configuration and available software compatibility. Initial efforts were therefore directed toward the resolution of these uncertainties.

These efforts revealed that in many cases the software routines were either inoperative or the documented operating procedures were incorrect. Communication with Varian Data Machines representatives resulted in the acquisition of a new set of system software. However, initial system operations still required extensive trial-and-error procedures and editing of similar system documentation. To preclude repetition of these time consuming efforts, detailed operating procedures applicable to the 620 system were documented and are contained in Section III of this study.

Finally, to investigate programming efforts, requirements for a representative data processing task were specified and input/output software routines were prepared to comply with these specifications.

II. DATA 620 SYSTEM

The Data 620 was built by Data Machines, Inc. (now Varian Data Machines) for the Fleet Numerical Weather Center (FNWC). Available records indicate that negotiations for purchase of the system commenced in 1965 with delivery to the FNWC occurring in 1967. In 1970, the system was turned over to the Naval Postgraduate School and is currently located in the Electrical Engineering Computer Laboratory.

There is little evidence to indicate that the system had been operated extensively prior to this study. Furthermore, there has been no formal preventive or corrective maintenance performed on the system since its receipt by the school. It is interesting to note that no computer malfunctions were detected during approximately 125 hours of operation. On the other hand, numerous failures were encountered with the ASR-33 teletype unit. In-house attempts to rectify the problems were only moderately successful. At any rate, this is obviously an intolerable situation if the system is to be utilized effectively for any purpose.

A. CONFIGURATION

The Data 620 system consists of the basic 620 minicomputer interfaced to an ASR-33 teletype which serves as the sole I/0 device. The unit includes a paper tape reader and punch. Although numerous options were available when the machine was purchased, there are no

records of the purchase or installation of any of these options. The Data 620 is a general purpose, parallel, binary computer characterized by:

Construction	Solid state, discrete components
Speed	1.8 microsecond memory cycle
Memory	4096 words
Word length	18 bits
Accessible registers	A (Accumulation)
	B (Low order accumulation)
	I (Instruction)
	P (Instruction Counter)
	X (Index)
Arithmetic functions	Single precision, fixed point add/subtract
Software	Complete package (Appendix C)

Although the 620 is physically large when compared to present state-of-the art machines, it appears to be well built and is obviously extremely reliable.

In addition, it has a memory protection feature which allows complete shutdown of the system for considerable periods of time without destruction of memory storage. This feature also proved to be very reliable.

B. LIMITATIONS

It is anticipated that a minicomputer system would be used for numerous relatively simple, repetitive data collection and/or processing functions. Furthermore, the system would be utilized primarily by personnel with limited experience in computer operation and programming at the assembler level. As a result, system flexibility and ease of operation are considered to be primary factors in the determination of system suitability. From this viewpoint, limitations of the Data 620 in its present configuration are related to its:

Memory Capacity

I/0 Device

Software Preparation

1. Memory Capacity

Initially, the 4096 memory capacity was considered to be adequate for performance of the desired functions. Further investigation proved this assumption to be erroneous. One difficulty arises from the requirement of software routines to perform all arithmetic operations with the exception of fixed point add/subtract. Although the routines are available (Appendix C), their use requires excessive memory allocation.

An illustrative example is that to perform the function of raising a floating point number to a fixed point power requires approximately 1200 words of memory. This is in excess of 25% of the total memory capacity.

A second difficulty arises when using the DAS 4A assembler program. The function of this program is to convert a source program tape into a loadable object program tape which allows preparation of software routines using assembly language. When loaded, the assembler program occupies in excess of 80% of the total memory capacity. The design and operation of the assembler is such that this does not preclude the assembly of large programs. However, if the object program requires more than 20% of the memory capacity, loading of the program will destroy a portion of the assembler program. Thus, if a given program requires reassembly due to improper operation or additional requirements, the DAS 4A assembler program must first be reloaded. Furthermore, the DAS 4A assembler is a two pass assembler which requires that the source program tape be read two times to complete assembly.

2. I/0 Device

As previously indicated, the ASR-33 teletype is the sole I/0 device. Although the print function of the unit is adequate for the system's intended use, the paper tape reader severely limits effective utilization of the Data 620 system. The rate of input is 10 characters per second (cps) which is extremely slow when compared to the computer capabilities. This results in a 15 minute loading time for the DAS 4A assembler. Program assembly is even more time consuming because of the "two pass" requirement.

3. Software Preparation

The DAS 4A assembler program allows programming of the system in assembler language. Although programming at this level is not difficult, it is tedious and time consuming particularly when compared to the compiler (FORTRAN) level. Once the program has been written and punched on paper tape, it still must be assembled prior to actual use.

III. DATA 620 SYSTEM OPERATING PROCEDURES

This section is intended to provide the relatively inexperienced operator with sufficient information to operate the system using standard system software. The section is divided into the following areas:

Bootstrap Loader
Binary Load/Dump Program
DAS 4A Assembler
Type B Teletype Controller

Procedures presented in this section have been obtained through trial-and-error methods in conjunction with applicable information contained in Refs. 1 and 2. Information relating to software capabilities has been edited to comply with the system configuration.

A. BOOTSTRAP LOADER

The bootstrap loader provides the means by which the binary load/dump program is loaded into memory. It must be loaded manually and is intended to remain in memory during system operation. As a result, the program need be loaded only when a new system is being initialized or the contents of memory are unknown.

To load the bootstrap loader:

1. Turn power on, press computer RESET and clear all registers by depressing register RESET for each register.

2. Set computer OPERATION to STEP mode and computer CONTROL to REPEAT mode.

3. Load instruction 054000 in the I (instruction) register. This instruction stores the contents of the A register relative to the P (Instruction counter) register.

4. Load the starting memory address (007756) of the bootstrap loader into the P register.

5. Load the following instruction codes in the A register, pressing step after loading each instruction. The computer loads the contents of the A register into the address specified by the P register which is incremented by one after each instruction is loaded.

ADDRESS	INSTRUCTION	MNEMONIC
007756	102601	CIB
007757	004013	ASLB
007760	004041	LRLB
007761	004446	LLRL
007762	001020	JBZ
007763	007772	MEMORY ADDRESS
007764	055000	STA
007765	001010	JAZ
007766	007600	MEMORY ADDRESS
007767	005144	IXR
007770	005101	INCR
007771	102601	CIB
007772	101201	SEN
007773	007756	MEMORY ADDRESS
007774	001000	JMP
007775	007772	MEMORY ADDRESS

To determine that the bootstrap loader has been properly loaded, perform the following steps.

1. Initialize CPU by pressing computer RESET.
2. Clear all registers.
3. Load instruction 014000 (load A relative to P) in the I register.

4. Load the starting memory address (007756) in the P register.
5. Set computer OPERATION to STEP mode and computer CONTROL to REPEAT mode.
6. Press STEP. The contents of each memory address is sequentially displayed in A register each time STEP is pressed.
7. If an error is found, reload erroneous instruction codes into memory. Note that the P register contains the error address plus one.

The above procedures can be used to manually load and check any program by loading the appropriate starting memory address in the P register.

B. BINARY LOAD/DUMP PROGRAM (BLD II)

The binary load/dump object program, which is loaded by the bootstrap loader, allows the user to load object programs from the teletype paper tape reader. An object program is produced by the DAS 4A assembler. BLD II also allows the user to punch the specified contents of memory on paper tape in a reloadable format. Once loaded, BLD II occupies addresses 07400 to 07755. It is recommended that the program remain resident in core and ; therefore, need be loaded only when initializing a new system or if the program has been inadvertently destroyed. The procedures that follow assume that the computer has been energized and the bootstrap loader is resident in core.

1. <u>Loading the Binary Load/Dump Program</u>

a. Place teletype in Off-Line mode and press:

(1) CNTRL and 'D' (Print suppress)

(2) CNTRL and 'T' (Punch off)

(3) CNTRL and 'Q' (Reader on)

b. Set teletype On-line.

c. Set the teletype reader control lever in the LOAD position and insert the BLD II program tape in the reader with the first binary frame in the read position.

d. Set Computer controls as follows:

(1) Sense switch 1, 2 and 3 to OFF.

(2) REPEAT to OFF.

(3) STEP/RUN to STEP.

(4) Clean all registers.

(5) Manually enter 007770 in P register.

(6) Manually enter 007600 in X register.

(7) Press SYSTEM RESET and RUN.

e. To initiate loading, set teletype reader control lever to RUN position.

f. A successful load of the BLD II program is indicated by:

(1) Computer in STEP mode.

(2) Teletype reader halted.

(3) P register = 007600.

(4) B register = 000000.

2. <u>Procedure to load program tapes</u>

Once the BLD II program is resident in core, the system is ready to load standard or locally assembled object program tapes. The

procedures that follow assume the computer has been energized:

a. Place teletype in Off-line mode and press

(1) CNTRL and 'D' (Print Suppress)

(2) CNTRL and 'T' (Punch OFF)

(3) CNTRL and 'Q' (Reader On)

b. Set teletype on-line.

c. Place the program tape in the reader with first binary frame in the read position and set reader control lever to RUN.

d. Set computer controls:

(1) Sense switch 1, 2 and 3 to OFF.

(2) REPEAT to OFF.

(3) STEP/RUN to STEP.

(4) Clear all registers.

(5) Manually enter 007600 in the P register.

e. Manually set the A register to desired load mode

(1) $A < 0$ to verify program tape which performs only check-sum error-checking to ensure that an object tape contains no errors before loading into core memory.

(2) $A=0$ to load program tape and halt.

(3) $A > 0$ to load program tape and execute the program.

f. Press SYSTEM RESET and RUN

g. A successful load is indicated by:

(1) Computer in STEP mode.

(2) Reader halted.

(3) P register = 007600

(4) A register = load mode

(5) X register = execution address.

h. A checksum on format error is indicated by

(1) Computer in STEP mode.

(2) Reader halted.

(3) P register = 007600.

(4) B register = 777777.

(5) X register = load address of last record read.

i. To restart, position the program tape at the previous record mark and press RUN. The standard object program contains numerous record marks which consist of a series of 8 level punches.

3. Procedure to Punch Program Tapes

The BLD II program allows areas of memory to be punched on paper tape in an object-tape-loadable format. That is, a selected program or data can be punched from memory which may be reloaded by use of BLD II. The procedure that follows assumes the computer has been energized.

a. Place teletype in off-line mode and press:

(1) CNTRL and '0' (Print suppress).

(2) CNTRL and 'R' (Punch on).

b. Set teletype on-line

c. Set computer controls

(1) Sense switch 1, 2 and 3 to off.

(2) REPEAT to off.

(3) STEP/RUN to STEP.

(4) Clear all registers.

(5) Set A register = first address of area to be punched.

(6) Set B register = last address of area to be punched.

(7) Set X register = address of first instructions to be executed (at load time) or to 777777 if noncontiguous memory areas are to be punched.

(8) Set P register = 007404.

d. Press SYSTEM RESET and RUN

e. Tape will be punched and the computer will go to STEP mode with registers unaltered. If noncontiguous areas are to be punched, perform steps c through d above, entering the new areas to be punched in the A and B registers. Prior to punching the last area, set the X register to the address of the first instruction to be executed (at load time).

C. DAS 4A ASSEMBLER

The DAS 4A assembler language (DAS) allows the replacement of numerical codes with instruction mnemonics. This provides for memory addresses to be referenced symbolically and for constants to be used with automatic conversion to binary values. Additionally, comments can be added between symbolic statements or appended to the statements to facilitate program checkout and documentation. The

DAS mnemonics and corresponding functions are described in references 3 and 4.

1. Purpose and Description

The function of the DAS 4A program is to translate symbolically coded instructions and data (Source program) into binary instructions and data (Object program). The object program is output on punched paper tape in a format which can be loaded by use of BLD II. Also, the source and object program can be listed side by side on the teletype printer. An example of such a listing is contained in Appendix B.

The DAS 4A program tape is an object program and is loaded using the procedure in Section III B 2. It is recommended that the "load and execute" mode be used when loading the assembler. Actually, the assembler program tape is comprised of two sections; the I/0 section and the Assembler section. During loading of the program, the teletype will make the following three requests for definitions of I/0 devices:

ENTER DEVICE NAME FOR SI (source input)

ENTER DEVICE NAME FOR BO (binary output)

ENTER DEVICE NAME FOR LO (list output)

Respond to each request in turn with a carriage return which is a default assignment specifying the teletype as the I/0 device. After response to the third request, loading will continue. Upon completion of a successful load the system is ready for program assembly.

2. <u>Operation</u>

It is important to note that DAS 4A is a two pass assembly system, which means that the source program must be read two times for complete assembly. During the first pass, values are assigned to all labels appearing in the location field and placed in the label table. During the second pass, the appropriate values for the instruction field and the variable field are assembled into the object instruction. Output occurs during the second pass and is controlled by the operator as described in the procedures that follow.

a. Pass 1

(1) Place teletype off line and press.

CNTRL and 'D' (Print suppress)

CNTRL and 'T' (Punch off)

CNTRL and 'Q' (Reader on)

(2) Set teletype on-line and place source program in reader.

(3) Set computer controls:

Sense switch 1 to On

Sense switches 2, 3 to Off

REPEAT to OFF

STEP/RUN to STEP

Clear all registers

(4) Press RESET, STEP and RUN

b. Pass 2

(1) Reposition source tape to beginning in reader.

(2) Set sense switch 1 to OFF.

(3) Set sense switch 2:

ON-for program listing.

OFF-to suppress program listing.

(4) Set sense switch 3:

ON-to output punched object program tape.

OFF-to suppress punched output.

(5) Clear all registers.

(6) Press SYSTEM RESET, STEP and RUN.

(7) To obtain extra copies of the program, repeat pass 2 as desired.

An END statement in the source program terminates both passes 1 and 2. A MORE directive in the source program causes the computer to stop and wait until the inputs are prepared and RUN is pressed.

D. TYPE B TELETYPE CONTROLLER

The applicable I/0 instructions for the system in its current configuration are presented to assist the operator in program preparations. The instructions are presented in DAS 4A assembler language followed by the corresponding six digit machine language code.

1. <u>Transfer</u>

,INA	,01	102101	Read read reg to A
,CIA	,01	102501	Read read reg to cleared A
,INB	,01	102201	Read read reg to B
,CIB	,01	102601	Read read reg to cleared B
,IME	,01	102001	Read read reg to Memory
,0AR	,01	103101	Load write reg from A
,0BR	,01	103201	Load write reg from B
,OME	,01	103001	Load write reg from Memory

2. <u>Sense</u>

,SEN	,101	101101	Write reg ready
,SEN	,201	101201	Read reg ready

It is important to note that the Read and Write buffer registers are 8 bits in length. They transfer to and from the lowest order 8 bits in the A and B registers on the specified memory locations.

IV. USING INPUT/OUTPUT SUBROUTINES

This section contains the necessary information for use of the I/0 subroutines prepared by the author. The routines were prepared to determine and illustrate the software requirements to I/0 a block of data in accordance with Appendix C. They are written in DAS 4A language and contained in Appendix D. Once assembled, the routines are intended to remain resident in core (439 words) and used in conjunction with locally prepared data processing routines for specific problems.

A. GENERAL INFORMATION

To input a block of data, the data must be preceded by a line of data identifying storage location and size of the data block. The initial line of data must also contain data block processing parameters for the given problems.

1. Terminology

Terms used in the presentation of material in this section are defined as follows:

a. Valid Data

(1) Decimal Digits (0 through 9).

(2) Sign (+, -).

(3) Space (terminates data word).

(4) Carriage return (terminates data block and data word).

b. Data Word

Sign followed by 1 to 4 decimal digits.

c. Data Line

A set of data words terminated by a carriage return.

d. Data Block

Block of data to be processed. It is comprised of a set of data lines but does not include initial data line described above.

2. Memory Allocations

Computer locations 500 through 515 octal are reserved for storage of the initial data line required prior to input of the data block. The information will be stored as follows:

Location	Content
0501	Desired storage address of first data word in data block.
0502	Number of data lines in data block.
0503-0514	Data processing parameters stored sequentially in order of input.

After input of data block, location 0500 will contain address of last data word in the data block.

Locations 0515 through 1024 octal are reserved for storage of the data block.

3. Data Format

a. Input

The input routine is designed to allow loading of an initial data line, followed by a data block utilizing the teletype keyboard or

paper tape reader. The format of the data must be in accordance with the following:

(1) All non-valid data will be ignored.

(2) A data word containing more than 4 digits and/or non-valid data will be ignored.

(3) Absence of sign indicates positive data word.

(4) Data words, with exception of last data word in a data line, must be separated by a minimum of one space.

(5) Data words are right adjusted, e.g., an input of one digit, X, will be processed as 000X.

b. Output

The output routine is designed to output a data block after it has been processed. The data will be printed on a new page with 10 data words; each separated by a single space, per line. Each line will be single spaced.

B. SUBROUTINE DESCRIPTION

The following information is provided to assist in the use of the input/output subroutines. The supporting subroutines in addition to the main routines are included to allow individual use if applicable.

1. <u>Data Block Input and Print Routine (DATB)</u>

Subroutine DATB is designed to input a block of data to be processed. The input device is the teletype keyboard or paper tape reader. Actuation of the computer sense switch one will allow input

via the keyboard; otherwise, input is from the paper tape reader. Use of subroutine DATB requires an initial data line containing the data block storage, size and processing parameters. Computer storage is allocated for a maximum of 10 data processing parameters and a maximum of 200 data words to be processed.

The subroutine will accept and store one data line in BCD, sign magnitude format. Input is then suspended while the data line is echo printed. Upon completion of print each data word is converted to binary and restored in its initial location. This sequence is repeated until the specified number of data lines has been input.

The call sequence and storage requirements are as follows:

CALL SEQUENCE	,JMPM	,01025
DATB		34 WORDS
SUPPORTING SUBROUTINES		255 WORDS
TOTAL STORAGE		289 WORDS

2. Data Line Input and Print Routine (DATL)

Subroutine DATL is the main supporting routine for subroutine DATB. It will accept and store one data line in BCD, sign magnitude format. Input is then suspended while the data line is echo printed. Upon completion of print each data word is converted to binary and restored in its initial location. The input device is the teletype keyboard or paper tape reader. Actuation of the computer sense switch one will allow input via the keyboard; otherwise, input is from the paper tape reader.

Individual use of the routine requires entry with the desired storage address, of the first data word in the data line, in the X register. The routine exits with the X register increased by the number of data words in the data line.

The call sequence and storage requirements are as follows:

CALL SEQUENCE	,JMPM	,DATL
DATL		42 WORDS
SUPPORTING SUBROUTINES		213 WORDS
TOTAL STORAGE		255 WORDS

3. Data Line Input and Store Routine (DASS)

Subroutine DASS is the basic building block for the input routines DATB and DATL. It is designed to accept and store one data line with the format criterion previously described. It will activate the input device, teletype keyboard or paper tape reader, in accordance with the condition of computer sense switch one. Actuation of sense switch one energizes the keyboard; otherwise, the paper tape reader is energized. Upon receipt of a carriage return which signals termination of a data line, the input device is deenergized. The data words are stored relative to the X register in BCD, sign magnitude format. Individual use of the routine requires entry with desired storage origin of first data word in the X register. The routine exits, upon receipt of carriage return, with the X register increased by the number of data words in the data line.

The call sequence and storage requirements are as follows:

CALL SEQUENCE	, JMPM , DASS
DASS	100 WORDS
SUPPORTING ROUTINES	8 WORDS
TOTAL STORAGE	108 WORDS

4. ODE - BCD Output

Subroutine ODE will output one data word via the teletype printer. The data word must be in the A register in BCD, sign magnitude format. The output format is one space followed by the sign (+, -) and then 4 decimal digits. The X register is unchanged. The call sequence and storage requirements are as follows:

CALL SEQUENCE	, JMPM , ODE
ODE	34 WORDS
SUPPORTING SUBROUTINES	8 WORDS
TOTAL STORAGE	42 WORDS

5. Output Single Character (BO)

Subroutine BO will output an 8 bit binary word from the B register via the teletype printer. The word must be right adjusted. The call sequence and storage requirements are as follows:

CALL SEQUENCE	, JMPM , BO
BO	8 WORDS
SUPPORTING SUBROUTINES	0 WORDS
TOTAL STORAGE	8 WORDS

6. Data Block Output (DABO)

Subroutine DABO is used to output a block of data after it has been processed. The output is via the teletype printer in a format previously described. Use of the routine requires entry with the address of the first data word in the data block to be stored in 0501 octal and the address of the last data word in the data block to be stored

in 0500 octal. The routine exits with the X register unchanged. The call sequence and storage requirements are as follows:

CALL SEQUENCE	, JMPM	, DABO
DABO		41 WORDS
SUPPORTING SUBROUTINES		109 WORDS
TOTAL STORAGE		150 WORDS

7. Standard Software Routines

Subroutines DATB, DATL and DABO utilize four standard software supporting routines. The use and description of these routines are contained in reference 5. Storage requirements are indicated below:

a. XMUL Standard Software Multiply

XMUL	40 WORDS
SUPPORTING ROUTINES	0 WORDS
TOTAL STORAGE	40 WORDS

b. XDTB Fixed Point Integer Decimal to Binary Conversion

XDTB	31 WORDS
SUPPORTING ROUTINES	40 WORDS
	71 WORDS

c. XDIV Standard Software Divide

XDIV	77 WORDS
SUPPORTING ROUTINES	0
TOTAL	77 WORDS

d. XBTD Fixed Point Integer BIN to DEC Conversion

XBTD	34 WORDS
SUPPORTING ROUTINES	77
TOTAL	101 WORDS

V. CONCLUSIONS

A. RESULTS

The results of this study raise a question in the author's mind as to the definition of a minicomputer. Use of memory storage capacity alone to classify modern computer systems would be extremely misleading due to the development of microprogramming techniques. For example, the 1200-word memory requirement for the Data 620 to raise a floating point number to a fixed point power causes one to ponder the equivalent memory capacities of some of the small electronic desk calculators. The microprogramming of arithmetic functions obviously greatly reduces the memory storage capacity requirements of a computer system.

Since the Data 620 requires that all arithmetic functions, with the exception of fixed point add/subtract, be performed by software, its present memory capacity is considered inadequate to perform the functions described in Section II B. Furthermore, the time and effort involved in software design and assembly is considered to be excessive as evidenced by the information presented in Section IV and Appendix D of this study.

Based on the material presented in this study and experience in operating the Data 620 system, it is concluded that:

1. The Data 620, in its present configuration, is not suitable to perform the desired data collection and/or processing functions.

2. The minimum requirements to upgrade the system would include the addition of 4096 words of memory, a magnetic tape deck and a high speed paper tape reader which would involve a significant expenditure.

3. The Data 620 system is suitable for performing control and/or monitoring functions in situations where software requirements remain essentially constant for a considerable period of time. It is also very well suited for utilization in a basic computer course such as EE 2810.

4. A suitable system for use by the Aeronautics Department for data collection and/or processing should possess the following attributes:

a. Easily and conveniently programmable, preferably from a keyboard incorporating the common arithmetic functions.

b. I/0 including cassette magnetic tape, paper tape and teletype. The format of the I/0 should also be easily and conveniently programmable.

c. Facility for conveniently interfacing with devices supplying ASCII coded data.

B. RECOMMENDATIONS

Based on the preceding material and with consideration of the age and corresponding obsolescence of the Data 620 system, the following recommendations are offered:

1. There should be no expenditures toward upgrading the Data 620 system.

2. Provisions for adequate maintenance of the Data 620 system should be established and efforts to utilize the system in courses of instruction similar to EE 2810 should be initiated.

3. Prior to the purchase of any future systems, a thorough investigation should be conducted to insure the system in question will effectively perform the desired functions without excessive expenditures for "accessories".

APPENDIX A

SYSTEM SOFTWARE

A complete directory of all standard software program tapes currently on-hand is contained herein. The directory is divided into the following areas:

MAINTENANCE/TEST
OPERATIONAL
MATH

Program format (Source or Object) is also indicated for each program. This is important to note since a source program must be assembled into an object program prior to use. The DAS 4A assembler performs this function. The BLD II program contains the load routine for all object programs.

A. MAINTENANCE/TEST

1. Maintain II Test Executive (Object)

   92U0107-001C

   06-02-71

2. 620 Instruction Test Part I (Object)

   92U0107-002C

   12-19-70

3. 620 Instruction Test Part II (Object)

   92U0107-003C

   3-12-71

4. <u>620 Memory Test Part I (Object)</u>

   92U0107-0200

   07-06-71

5. <u>620 Memory Test Part II (Object)</u>

   92U0107-0ZIA

   05-25-71

6. <u>620 Teletype Test (Object)</u>

   92U0107-0050

   02-18-71

B. OPERATIONAL

1. <u>DAS 4A Assembler and I/0 (Object)</u>

   92U0304-098D

   11-08-71

2. <u>Debug Utility (Aid II) (Object)</u>

   92U0207-001A

   11-15-71

3. <u>Source Tape Correction (Cor) (Source)</u>

   Version B

4. <u>Source Tape Correction (Cor) (Object)</u>

   Version B

5. <u>Binary Load/Dump (BLD II) (Object)</u>

   Version 4.1

C. MATH

The available mathematical functions are contained on two program source tapes. Each of the tapes, FIXED POINT MATH and FLOATING POINT MATH, contain a series of subroutines separated by a blank section. The subroutines are terminated by a MORE, END combination to allow assembly of only the required routines for problem solution. The subroutines are listed below in the order they appear in the tapes.

1. Fixed Point Math (Source)

XCOS - Single Precision Cosine

XEXN - Single Precision Negative Exponential

XATN - Single Precision Arctangent

XLOG - Single Precision Logarithm

POLY - Single Precision Polynomial Evaluation

XSIN - Single Precision Sine

XEXP - Single Precision Positive Exponential

XSQT - Single Precision Square Root

XMUL - Standard Software Multiply

XBTD - Integer Binary to Decimal Conversion

XDTB - Integer Decimal to Binary Conversion

XDIV - Standard Software Divide

XDAD - Fixed Point Routine

XDCO - Double Precision Two's Complement

XDMU - Double Precision Multiply

XDSU - Double Precision Subtract

XDDI - Double Precision Divide

2. Floating Point Math

$HE - I**J (Fixed Point Numbers)

$PE - A**I (A floating point, I fixed point)

$QE - A**B (Floating point numbers)

ALOG - Natural Log

EXP - Computes E**X, ARG, X is floating point

COS - Cosine

SIN - Sine

ATAN - Arctangent

SQRT - Square Root

$QM - Multiply

$QN - Divide

$QK - Add

$QL - Subtract

$FAS - Add/Subtract

$FSM - Separate mantissa

$NML - Normalize routine

$IS - Fixed point integer to floating point

$PS - Floating point to integer

IABS - Absolute value of fixed point number

ABS - Absolute value of floating point number

ISIG - Sets sign of fixed point number

SIGN - Sets sign of input parameter

$HM - Integer Multiply

$HN - Integer Divide

$SE - Subprogram entry control

$ER - Error Subroutine

APPENDIX B

DAS 4A LISTING

```
                   *       FIXED POINT MATH  18 BIT,  NO MUL/DIV
                   *
                   *  XMUL          STANDARD SOFTWARE MULTIPLY
                   *
                   *              A, B>[B*PAR]<A
                   *              X IS UNCHANGED  40  WORDS
007100                      , ORG    , 07100
        000022    NBIT      , EQU    , 18
007100  124025    ADD       , DATA   , 0124025    ADD MCND
007101  134023    ERA       , DATA   , 0134023    ERA SIGN
007102  124013    SOF       , DATA   , 0124013    ADD SIGN
007103  144007    SUB       , DATA   , 0144007    SUB CND
007104  007400    BGN       , ROF    ,            RESET OF
007105  074037              , STX    , XMXR       SAVE XR
007106  034033              , LDX    , XMUL       GET ADDRESS OF CALL SEQ
007107  035000              , LDX    , 0, 1       GET ADDR OF MCND
007110  035000              , LDX    , 0, 1       GET MCND
007111  074034              , STX    , MCND       AND SAVE
007112  034035              , LDX    , K15        SET BIT COUNT
007113  004462    RPT       , LLRL   , 18
007114  004461              , LLRL   , 17         A SIGN > LSB OF MPLR
007115  124031              , ADD    , XSIG       SET OF IF LSB>1
007116  004441              , LLRL   , 1          ALIGN PARTIAL PRODUCT
007117  003001              , XOF    , ADD        ADD MCND IF LSB>1
007120  007100
007121  004501              , LASR   , 1          AND SHIFT RIGHT
007122  003001              , XOF    , ERA        INVERT SIGN IF OF
007123  007101
007124  005344              , DXR    ,            COUNT BITS DEVELOPED
007125  001040              , JXZ    , *+4        JMP IF DONE
007126  007131
007127  001000              , JMP    , RPT        ELSE REPEAT
007130  007113
007131  004462              , LLRL   , NBIT       A SIGN>MPLR SIGN
007132  003004              , XAN    , SOF        SET OF IF NEG MPLR
007133  007102
007134  004462              , LLRL   , NBIT       RESET PRODUCT
007135  003001              , XOF    , SUB        SUB MCND IF NEG MPLR
007136  007103
```

```
007137  044002          , INR    , XMUL   SET RETURN
007140  034004          , LDX    , XMXR   RESTORE XR
007141  001000          , JMP*   , XMUL   A, B  B*M  A
007142  107142
007142          XMUL  , BES    , 0      ENTRY
007143  001000          , JMP    , BGN
007144  007104
007145          XMXR  , BSS    , 1      TEMPORARY STORAGE
@@%@9%J UK SJ KL G@ @G C[C@C(@G C ]@$2@$1J YES !@XA@9@@@$$
007147  400000  XSIG  , DATA   , 0400000
007150  000021  K15   , DATA   , 17
        000000          , END    ,
LITERALS
@@B@9' @@@@Q  94POINTERS
@@@@@@@@@SYMBOLS
007100  ADD    007104 BGN   007101  ERA  007150 K15 007146 MCND
000022 NBIT
00/113  RPT    007102 SOF   00 103  SUB  007142 XMUL 007145
XMXR 007147 XSIG
 0 ERRORS
```

APPENDIX C

PROPOSED DATA PROCESSING REQUIREMENT

A. PROBLEM DEFINITION

A requirement exists to process a block of data contained on paper tape in ASCII code. It is desired to prepare software routines to allow utilization of the Data 620 system to perform the following functions.

1. Input and echo print the block of data.

2. Convert the data to binary format for processing and store in memory.

3. After processing, print the results.

4. Provide for input of parameters, required to process the data, from either paper tape or the teletype keyboard.

B. FORMAT DEFINITION

The software I/0 routines must be compatible with the format specifications listed below:

1. A line of data is terminated by a Carriage Return, X-OFF and Line Feed in that order.

2. A line of data may contain from 1 to 10 data words.

3. A data word consists of a Space, Sign (+, -) and 1 to 4 decimal digits in that order.

4. Storage must be allocated for a minimum of 50 data words and 5 data processing parameters.

APPENDIX D

INPUT/OUTPUT DIRECTORY

The directory lists subroutines required to input/output a block of data. The main input (DATB)/output (DATO) routines contain numerous supporting subroutines which may be used individually if applicable to a given problem. Each listing delineates the required supporting subroutine calls in order that all programs may be assembled.

All programs are terminated with a MORE, END, combination to facilitate preparation and allow assembly of only those desired programs.

Only the following subroutines have been tested:

ODE
BO
XMUL
XDIV
XDTB
XBTB

1. <u>Data Block Input and Print Routine (DATB)</u>

Supporting subroutines: DATL
DASS
ODE
BO
XMUL
XDTB

```
         , ORG          , 01025
DATB     , ENTR         ,
         , LDB          , NP
         , JMPM         , BO
```

```
         , LDB         , CR
         , JMPM        , B0
         , LDB         , CR
         , JMPM        , B0
         , LDX         , =0501      Storage origin
                                    parameters
         , JMPM        , DATL       Get parameters
         , LDB         , LF
         , JMPM        , B0

         , LDX         , 0501       Get storage origin
                                    for data block
         , STX         , TEP
         , LDA         , 0502       Get # data lines
         , STA         , TEP+1
NLIN     , JMPM        , DATL       Read, store, print
                                    convert to binary
                                    and restore one
                                    data line
         , LDA         , TEP+1
         , DAR         ,
         , JAZ         , FIN        Check for end of
                                    data, YES, exit
                                    NO, get next line
         , STA         , TEP+1
         , JMP         , NLIN
FIN      , STX         , 0500       Stores location of
                                    final data word in
                                    0500
         , JMP*        , DATB
TEP      , DATA        , 0, 0
NP       , DATA        , 0214
LF       , DATA        , 0212
CR       , DATA        , 0215
         , MORE        ,
         , END         ,
```

2. <u>Data Line Input and Print Routine (DATL)</u>

Supporting Subroutines: DASS
ODE
BO
XMUL
XDTB

```
DATL    , ENTR     ,
        , STX      , DL0        Save data origin,
                                location
        , JMPM     , DASS       Get one line data
        , STX      , NWD
        , LDX      , DL0
NXWD    , LDA      , 0, 1
        , JMPM     , ODE        Output one word
        , LDA      , 0, 1
        , JAP      , DB         BCD--BINARY
        , LRLA     , 1
        , LSRA     , 1          Remove sign bit
        , JMPM     , XDTB
        , CPA      ,
        , IAR      ,            Two's compliment
        , JMP      , DB+1       Store neg number
DB      , JMPM     , XDTB
        , STB      , 0, 1       Store binary
                                number
        , IXR      ,
        , TXA      ,
        , SUB      , NWD        Check end data
                                line
        , JAZ      , DED        Yes, exit
        , JMP      , NXWD       No, get next word
DED     , LDB      , CR
        , JMPM     , B0
        , LDB      , LF
        , JMPM     , B0
        , JMP*     , DATL       Return
NWD     , DATA     , 0
DL0     , DATA     , 0
CR      , DATA     , 0215
LF      , DATA     , 0212
        , MORE     ,
        , END      ,
```

3. <u>Data Line Input and Store Routine (DASS)</u>

Supporting subroutines: B0

```
DASS    , ENTR     ,
        , ROF      ,            Reset ov'flow
        , STX      , TEM+2      Save X
        , JSSI     , KIN        Jump, keyboard
                                input
```

```
         , LDB      , =0204        No, tape input,
                                   print off, read on
         , JMPM     , B0
         , LDB      , =0221
         , JMPM     , B0
         , JMP      , RCH
KIN      , LDB      , =0223        Keyboard input
         , JMPM     , B0           Reader off
         , LDB      , =0215        CR
         , JMPM     , B0
         , LDB      , =0212        LF
         , JMPM     , B0
         , LDB      , =0201        Printer on
         , JMPM     , B0
RCH      , LDX      , =4           Set max 4 digits
         , TZB      ,
         , SEN      , 0201, *4     Input character
                                   A reg
         , JMP      , *-2
         , CIA      , 1
         , STA      , TEM          Save character
         , SUB      , =0255        Test neg
         , JAZ      , SAV          Save sign
         , JMP      , SAV+1        Pos Num
SAV      , LDB      , =0400000
         , STB      , TEM+1        Store sign, + or -
VCH      , LDA      , TEM
         , SUB      , =0260        Lower limit, valid
                                   data
         , JAN      , LND          Test end of data
                                   line
         , LDA      , TEM
         , SUB      , =0372        Upper limit, valid
                                   data
         , JAN      , NCH          Count digits, max
                                   of 4
         , JMP      , RCH          Invalid data,
                                   clear out
LND      , LDA      , TEM
         , SUB      , =0215        Test end data line
         , XAZ      , SOF          Set flag
         , JAZ      , GSN          Get sign
         , JMP      , RCH          Invalid data,
                                   clear
```

```
NCH     , DXR     ,           Reduce digit count
        , LDA     , TEM
        , LRLA    , 14        Adjust digit, BCD,
                              B reg
        , LLRL    , 4
        , SEN     , 0201.A2
        , JMP     , *-2       Get digit
AZ      , CIA     , 1
        , STA     , TEM       Save
        , SUB     , =0240     Test end of word
        , JAZ     , GSN       Yes, get sign; no,
                              test valid input
        , JMP     , VCH
GSN     , TXA     ,           Test for min of 1,
                              max of 4
        , SUB     , DMAX      Digit/WRD
        , JAZ     , RCH       No, start over
        , JAN     , RCH
        , TBA     ,           Yes, get word
        , ADD     , TEM+1     Add sign
        , LDX     , TEM+2     Set storage loc
        , STA     , 0, 1
        , JOF     , EDL       End of data line
        , IXR     ,           Not end data line,
        , STX     , TEM+2     Set next storage
                              loc
        , JMP     , RCH       Get next data wrd
EDL     , LDB     , =0223     Turn reader off
        , JMPM    , B0
        , ROF     ,           Reset ov'flow
        , IXR     ,
        , JMP*    , DASS      Return
TEM     , DATA    , 0, 0, 0   Temp storage
DMAX    , DATA    , 4
        , MORE    ,
        , END     ,
```

4. <u>ODE-BCD Output</u>

Supporting subroutines: BO

```
ODE      , ENTR     ,
         , LDB      , KI                   Output space
         , JMPM     , B0
         , LDB      , KI+3                 Print enable
         , JMPM     , B0
         , TZB      ,
         , LLSR     , 7
         , ADD      , KI+1                 Output sign:
                                           +, -
         , JMPM     , B0
         , STX      , T0                   Save X
         , LDXI     , 4                    4 digits
LPI      , TZB      ,
         , LLSR     , 4                    Assemble digit
                                           for output
         , ORA      , KI+2
         , LLRL     , 8
         , JMPM     , B0                   Output 1 digit
         , DXR      ,
         , JXZ      , RESX                 4 digits, exit
         , JMP      , LPI                  No, get next
                                           digit
RESX     , LDX      , T0                   Restore X reg
         , JMP*     , ODE                  Return
TO       , DATA     , 0
KI       , DATA     , 0255, 0253000,
                      054000, 0201
         , MORE     ,
         , END      ,
```

5. Output Single Character B0)

Supporting subroutines: None

```
B0       , ENTR     ,
         , SEN      , 0101, B01
         , JMP      , B0+1
B01      , OBR      , 01
         , JMP*     , B0
         , MORE     ,
         , END      ,
```

6. Data Block Output (DABO)

Supporting subroutines: ODE
B0
XDIV
XBTB

```
DAB0     ,ENTR     ,
         ,STX      ,TEM       Save X
         ,LDX      ,0501      Data start loc
         ,LDB      ,=0214     New page
         ,JMPM     ,B0
NL       ,LDB      ,=0215     CR
         ,JMPM     ,B0
         ,LDB      ,=0212     LF
         ,JMPM     ,B0
NWRD     ,LDA      ,0,1
         ,JAN      ,SG
         ,JMPM     ,XBTD      BIN to BCD +
                              num
         ,JMP      ,0P
SG       ,JMPM     ,XBTD      BIN TO BCD. -
                              num
         ,ORA      ,NEG
0P       ,JMPM     ,0DE       Output 1 word
         ,IXR      ,
         ,TXA      ,
         ,SUB      ,0500      Chk end data
                              block
         ,JAZ      ,EX        Yes, exit
         ,TXA      ,
         ,SUB      ,WPL       No, chk end data
                              line
         ,JAZ      ,NL        Yes, start new
                              line
         ,JMP      ,NWRD      No, get next
                              word
EX       ,LDX      ,TEM       Restore X
         ,JMP*     ,DAB0
TEM      ,DATA     ,0
NEG      ,DATA     ,0400000
WPL      ,DATA     ,10
         ,MORE     ,
         ,END
```

7. Standard Software Multiply (XMUL)

Supporting subroutines: None

```
NBIT    , EQU     , 18
ADD     , DATA    , 0124025    Add mcnd
ERA     , DATA    , 0134023    Era sign
SOF     , DATA    , 0124013    Add sign
SUB     , DATA    , 0144007    SUB CND
BGN     , ROF     .            Reset OF
        , STX     , XMXR       Save XR
        , LDX     , XMUL       Get address of
                               call seq
        , LDX     , 0, 1       Get addr of MCND
        , LDX     , 0,1        Get MCND
        , STX     , MCND       And save
        , LDX     , K15        Set bit count
RPT     , LLRL    , 18
        , LLRL    , 17         A sign   LSB of
                               MPLR
        , ADD     , XSIG       Set of IF LSB   1
        , LLRL    , 1          Align partial
                               product
        , XOF     , ADD        Add MCND if
                               LSB   1
        , LASR    , 1          And shift right
        , XOF     , ERA        Invert sign if of
        , DXR     ,            Count bits developed
        , JXZ     , *+4        JMP if done
        , JMP     , RPT        Else repeat
        , LLRL    , NBIT       A sign MPLR sign
        , XAN     , SOF        Set OF if NEG MPLR
        , LLRL    , NBIT       Reset product
        , XOF     , SUB        Sub MCND if neg
                               MPLR
        , INR     , XMUL       Set return
        , LDX     , XMXR       Restore XR
        , JMP*    , XMUL       A, B   B*M   A
XMUL    , BES     , 0          Entry
        , JMP     , BGN
XMXR    , BSS     , 1          Temporary storage
MCND    , BSS     , 1
XSIG    , DATA    , 0400000
K15     , DATA    , 17
        , MORE    ,
        , END     ,
```

8. <u>Fixed Point Integer Decimal to Binary Conversion (XDTB)</u>

Supporting subroutines: XMUL

```
XDTB    ,ENTR    ,
        ,STA     ,AB
        ,STX     ,AB+1
        ,TAB     ,
        ,LDXI    ,3          Initial IZE count
        ,TZA     ,
        ,STA     ,AB+2
        ,LLRL    ,2          Shift BR, 18 bit
                             word only
        ,LLRL    ,4          Get next digit
        ,STB     ,AB+3
        ,LDB     ,AB+2
        ,CALL    ,XMUL, BA
        ,JXZ     ,*+7        Jump if complete
        ,DXR     ,           Else count digits
        ,STB     ,*+11       Save partial
                             product
        ,LDB     ,*+11       Get remaining
                             digits
        ,JMP     ,*-11       Go get next
                             product
        ,LDA     ,*+5        Restore AR
        ,LDX     ,*+5        Restore XR
        ,JMP*    ,XDTB       Return
BA      ,DATA    ,10
AB      ,DATA    ,0,0,0,0    Temp storage
        ,MORE    ,
        ,END     ,
```

9. <u>Standard Software Divide (XDIV)</u>

Supporting subroutines: None

```
TOP     ,STX     ,XR         Save XR
        ,DECR    ,4          Set sign indicator
        ,JAP     ,POSU       Set dividend pos
        ,CPX     ,           Set DSIN  neg
        ,CPB     ,           L0 order two, S
                             compl
        ,IBR     ,
        ,LRLB    ,1          Sign   0
        ,LSRB    ,1
        ,CPA     ,           Hi order two, S
                             compl
```

```
          ,JBZ        ,*+4
          ,JMP        ,*+3
          ,IAR        ,
POSU      ,STX        ,DSIN       Save DIVDN sign
          ,STA        ,DVDN       Save DIVDN
          ,LDX        ,XDIV       Get addr of call
                                  seq
          ,LDX        ,0,1        Get addr of PARAM
          ,LDA        ,0,1        Get divisor
          ,LDX        ,DSIN       Get DIVDN sign
          ,JAP        ,*+5        Set divisor POS
          ,CPX        ,           Set quotient sign
          ,CPA        ,           Two, S compl
          ,IAR        ,
          ,STA        ,DVSR       Save divisor
          ,LDA        ,DVDN       Get DIVDN
          ,STX        ,QSIN       Save QUOT sign
          ,LDX        ,K14        Set cycle count
          ,LRLB       ,1          Adjust LO order
                                  (delete sign)
          ,SUB        ,DVSR       SUB divsor
          ,SOF        ,
          ,JAP        ,XERR       JMP if overflow
                                  error
          ,ROF
NEGU      ,LLRL       ,1          Develop 14
                                  quotient bits
          ,ADD        ,DVSR       (Non restoring
                                  algorithm)
TEST      ,JXZ        ,ADJ        JMP if complete
          ,DXR        ,           Count bits
          ,JAN        ,NEGU       Jump if neg
                                  remainder
          ,LLRL       ,1          Shift quotorem
          ,SUB        ,DVSR       Subtract DIVSR
          ,JMP        ,TEST       Go test
ADJ       ,LRLB       ,1          Get last quotient
                                  bit
          ,JAP        ,*+4        JMP if OR
          ,IBR        ,           Else set LOB
          ,ADD        ,DVSR       Restore remainder
          ,LDX        ,QSIN       Get true quotient
          ,JXZ        ,*+4        JMP if negative
                                  quot
```

```
        , CPB      ,             Else set positive
        , DBR      ,
        , IBR      ,
        , LDX      , DSIN        GET TRUE REMAINDER
        , JXZ      , *+4         JMP if remainder
                                 neg
        , JMP      , *+4         Else leave pos
        , CPA      ,
        , IAR      ,
XERR    , INR      , XDIV        Set return
        , LDX      , XR
        , JMP*     , XDIV        A, BOM B   QUOT
                                 A  REM
XDIV    , BES      , 0           Entry
        , JMP      , TOP
K14     , DATA     , 16
XR      , BSS      , 1           Temp storage
DVSR    , BSS      , 1
DVDN    , BSS      , 1
DSIN    , BSS      , 1
QSIN    , BSS      , 1
        , MORE     ,
        , END      ,
```

10. <u>Fixed Point Integer BIN to DEC Conversion (XBTD)</u>

Supporting subroutines: XDIV

```
XBTD    , ENTR     ,
        , STA      , AC
        , STX      , AC+1
        , JAP      , *+4         Jump if positive
        , CPA      ,             Else complement
        , IAR      ,             And add one
        , TAB      ,
        , LDXI     , 3           Initialize count
        , TZA      ,
        , CALL     , XDIV, BC
        , STB      , *+17        Save BIN VAL
        , LDB      , *+17        Get previous
                                 digits
        , LLSR     , 4           Attach digit to
                                 result
        , JXZ      , *+7         Jump if complete
        , DXR      ,             Else count digits
        , STB      , *+12        Save digits assembled
        , LDB      , *+10        Get binary VAL
```

```
          , JMP      , *-11        Go get next digit
          , LSRB     , 2           Position to low
                                   order 18 bit only
          , LDA      , *+4         Restore AR
          , LDX      , *+4         Restore XR
          , JMP*     , XBTD        Return
AC        , DATA     , 0, 0, 0, 0  Temp storage
BC        , DATA     , 10          Constant
          , MORE     ,
          , END      ,
```

BIBLIOGRAPHY

1. Varian 620/f Computer Handbook, Varian Data Machines, 1970.

2. Varian 620/L Computer Handbook, Varian Data Machines, 1971.

3. DATA 620 SYSTEM REFERENCE MANUAL, Data Machines, Inc., Pub. No. S-2002-0866, August 1966.

4. DATA 620 PROGRAMMER REFERENCE MANUAL, Data Machines, Inc., Pub. No. S-2000-0866, August, 1966.

5. DATA 620 SUBROUTINE MANUAL, Data Machines, Inc., Pub. No. F-2002-366, October 1966.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Chairman, Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
4.	Professor L. V. Schmidt, Code 57Sx Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
5.	Asst Professor V. M. Powers, Code 52Pw Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
6.	LCDR Philip N. Hatfield, USN 626 S. Second Street West Memphis, Arkansas 72301	1

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

Minicomputer Utilization for Data Acquisition and Processing

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; September 1972

5. AUTHOR(S) (First name, middle initial, last name)

Philip N. Hatfield

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1972	56	5

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

An investigation is conducted of minicomputer utilization for data acquisition and/or processing functions. The study is directed primarily toward the suitability of the on-board Data 620 minicomputer system for performance of these functions. System configuration and a listing of available software are included in addition to applicable system operating procedures. Finally, illustrative I/0 software routines required to service a proposed data processing task are presented.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Unclassified

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Minicomputer						
Data 620						

DD FORM 1 NOV 65 1473 (BACK)
S/N 0101-807-6821

A-31409

Thesis 137118
H318 Hatfield
c.1 Minicomputer utilization for data acquisition and processing.