NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A USER'S GUIDE FOR THE DATA GENERAL NOVA® 800 MINICOMPUTER

by

Grant Douglas Ralph

December 1976

Thesis Advisor: Donald E. Kirk

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A USER'S GUIDE FOR THE DATA GENERAL NOVA® 800 MINICOMPUTER

by

Grant Douglas Ralph

December 1976

Thesis Advisor: Donald E. Kirk

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A User's Guide for the Data General Nova® 800 Minicomputer		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis December 1976
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Grant Douglas Ralph		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, Ca 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1976
		13. NUMBER OF PAGES 215
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

NOVA®
Data General Corporation
Minicomputer

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis is a comprehensive summary of the Data General NOVA® 800 minicomputer system used in the Electrical Engineering laboratory at the United States Naval Postgraduate School. The system hardware is discussed briefly. The major emphasis is placed on programming concepts which are presented in a modular form to encourage employment as a user's guide and instructional aid. Programming exercises are designed to consolidate the concepts introduced and demonstrate the advancement in sophistication each

new technique provides. Minimal discussion of the basic required instructions precedes each exercise to allow early and frequent personal operating experience on the equipment.

A USER'S GUIDE FOR THE DATA GENERAL NOVA® 800 MINICOMPUTER

by

Grant Douglas Ralph
Major, Canadian Forces
Bachelor of Engineering, Carleton University, Ottawa
Ontario, 1969

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

December 1976

Thesis
R17
c.2

ABSTRACT

This thesis is a comprehensive summary of the Data General NOVA® 800 minicomputer system used in the Electrical Engineering laboratory at the United States Naval Postgraduate School. The system hardware is discussed briefly. The major emphasis is placed on programming concepts which are presented in a modular form to encourage employment as a user's guide and instructional aid. Programming exercises are designed to consolidate the concepts introduced and demonstrate the advancement in sophistication each new technique provides. Minimal discussion of the basic required instructions precedes each exercise to allow early and frequent personal operating experience on the equipment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AB	- Absolute locatable Binary coding
AC	- Accumulators, as in AC0, AC1, AC2, AC3
A/D	- Analog to Digital Converter
ALC	- Arithmetic or Logical instruction
ASCII	-American Standard code for Information Interchange
ASM	- Extended Assembler program
ASR	- Automatic Send and Receive
BAUD	- Standard bit rate unit of teletype communication
BIT	- Binary Digit
BYTE	- Eight bits
CIL/W	-Core Image Loader/Writer program
CLI	- Command Line Interpreter program
CP	- Character Pointer in the EDITOR
CPU	- Central Processing Unit
CR	- Carriage Return on teletype
CRT	- The TEKTRONIX® TEK 31/10 Cathde Ray Tube
CRY	- Carry flag
CT	- Cassette Transport, as in CT0 and CT1
CTRL	- Control Key on teletype
D/A	- Digital to Analog Converter
DGC	- Data General Corporation
DMA	- Direct Memory Access
EDIT	- Symbolic Text Editor program
ESC	- Escape Key on teletype
FF	- Form Feed Key on teletype
IBM	- International Buisness Machines Incorporated
I/O	- Input/Output communication with peripherals
ION	- Interrupt On indicator light
LF	- Line Feed Key on teletype

LFE -	Library File Editor program
LSI -	Large Scale Integrated circuit
K -	Thousands, as in 1K, 8K, ...
MRI -	Memory Reference Instruction
MSB -	Most Significant Bit
PC -	Program Location Counter
PTP -	High Speed Paper Tape Punch
PTR -	High Speed Paper Tape Reader
RB -	Relocatable Binary coding
RLDR-	Extended Relocatable Loader program
ROM -	Read-Only Memory
RTC -	Real Time Clock
SR -	Source Language Routine (Assembly)
SOS -	Stand-Alone Operating System
SYSG-	System Generation program
TTI -	Teletype Input from the keyboard
TTO -	Teletype Output
TTP -	Teletype paper tape Punch
TTR -	Teletype paper tape Reader
TTY -	Teletype in general
VAC -	Voltage in Alternating Current

ACKNOWLEDGEMENTS

I would like to thank Data General Corporation of Southboro, Massachusetts for its kind permission to use the invaluable information from the many manuals available.

I am indepted to the Data General personnel in Palo Alto, California for their friendly and prompt attention whenever I needed information or reference manuals.

Special thanks to Professor Donald Kirk for finding time in his busy schedule as Chairman of the Department of Electrical Engineering at the Naval Postgraduate School, to take me 'under his wing'.

I. INTRODUCTION

A. BACKGROUND

A Data General Corporation (DGC) NOVA® 800 minicomputer has been available to the Electrical Engineering Department of the Naval Postgraduate School since 1974. During that period a limited effort has been directed towards utilizing this basic digital machine as an instructional aid in laboratory course work.

A 1975 thesis [Ref. 1] resulted in an analog to digital (A/D) and digital to analog (D/A) interface and a brief summary of the system's feedback control capability.

B. PURPOSE

A review of the system after completion of the first stage of hardware development identified the difficulty of a first time user in trying to comprehend what the system was or how it could be used. The large number of technical manuals which referred to various modifications and option characteristics amplified the confusion of the novice. Unless this problem was solved system development and expansion by thesis projects would be severely restricted. This thesis serves as a complete user's guide for the beginner. It is intended to summarize the information in all the various technical manuals and to explain the essential details. The assumption is made that the reader is familiar with the general features of a digital computer. It is hoped to stimulate the user to read further by injecting simple short exercises yielding the satisfaction of causing 'the beast' to respond in the desired manner.

Chapter II gives a description of the hardware configuration of the DGC NOVA® 800 microcomputer system available at the Naval Postgraduate School. The Operator Console switches are described with examples of their use. The chapter is intended to familiarize the user with the system and the manual techniques that may be used to initialize it.

In Chapter III the software operating system is introduced and the user is tutored through the four basic steps of program creation by the use of an example program.

In Chapter IV the programmer is taught the details of Assembly language source code. The techniques of direct program controlled communication, interrupts and cassette tape read or write are introduced.

Chapter V is the conclusion; it describes present system problems and recommendations for further system developement.

II. HARDWARE

The capability of any computer system is dictated by the basic model of processor (CPU), the options installed, the peripherals available, and the memory capacity (Fig. 1).

A. CENTRAL PROCESSING UNIT (CPU)

The present CPU is a NOVA® JUMBO 800 minicomputer with 8,192 (8K) words of 16-bit ferrite core memory [Ref. 2]. The highest direct address for the 8K memory is 17,777. Maximum core expansion (8K) words of 16-bit ferrite core memory. [Ref. 2] Maximum core expansion is to 32,767 (32K) words. This highest possible address of 77,777 (octal) requires only a 15-bit program location counter (PC). The sixteenth bit is used for indirect addressing. The highest indirect address for the 8K memory is 117,777. There are four 16-bit accumulators (AC0, AC1, AC2, AC3) and a carry flag (CRY) of one bit. Several memory locations have special significance. Absolute locations 0 and 1 are used during interrupt processing, and locations 20 thru 37 are automatically modified when indirectly addressed.[Ref. 3]

B. REAL TIME CLOCK (RTC)

A real time clock is available and its use is described further in the instruction set and programming sections. It may be used to pause within a program that is executing or to trigger interrupts for servicing routines on a real time basis. [Ref. 4]

C. TELETYPE (TTY)

The ASR-33 model teletype is available for keyboard input (TTI), or printout (TTO) and paper tape reading (TTR) and punching (TTP). This paper tape capability should not be confused with the high speed paper tape capability (PTR and PTP), which is not available at the Postgraduate School. The specific model of teletype has direct implications on the way communications can be maintained with it. [Ref. 5]

D. CASSETTE DRIVERS (CT0 AND CT1)

A maximum of eight cassettes can be configured with the system. The two units presently installed are designated as CT0 and CT1 by setting the appropriate thumb wheel on the driver chassis. Cassettes may be controlled by programmed routines or by using routines provided under the STAND-ALONE OPERATING SYSTEM (SOS). When the power cord is connected into the CPU rear outlet and the toggle switch is in the REMOTE position, the CPU master key controls the turn on and off of both units.[Ref. 5]

E. DIGITAL TO ANALOG CONVERTER (D/A)

The D/A provides simultaneous output of two bipolar 10 volt analog signals and one timing signal from a 12-bit code [Ref. 8]. The A/D and D/A were incorporated in the system in a previous thesis project completed in 1975. [Ref. 1]

F. ANALOG TO DIGITAL CONVERTER (A/D)

The A/D converter provides high speed translation of bipolar 10 volt differential analog signals to a 12-bit binary code. The most significant bit (MSB) is then extended to complete the normal 16-bit word length. The A/D will multiplex any one of eight addressable inputs. [Ref. 7]

G. TEKTRONIX® DISPLAY (CRT)

An adapter kit has been purchased to allow the CRT display to supplement the teletype unit. The necessary hardware connections have not been made. The CRT could use the second TTY connections and device codes (TTI1 and TTO1).[Ref. 6]

H. OPERATOR CONSOLE

The operation of the computer and the contents of specified memory locations can be observed or altered by using the operator console (Fig. 2). The lights in the upper right-hand portion of the console display control conditions, the rows of lights in the upper center portion display the processor registers. If a light is lit, it means the corresponding bit is 1. If the light is not lit, the corresponding bit is 0.

Below the lights is a bank of toggle switches through which the operator can supply addresses and data to the processor. When these switches are in the up position, they represent a 1; when down, they represent a 0. Only switches 1 thru 15 are used for entering addresses. The data register can be used in conjunction with some of the operating switches, located at the bottom of the panel. Each switch lever is actually two momentary-contact logical switches with a common off position in the center. Lifting the lever up turns on the switch whose name is printed above it; pressing it down turns on the switch whose name is written below it. When released, these switches automatically return to off.

At the upper left is a 3-position key-operated rotary switch that controls power and locks the console. Turning it to ON simply turns on power. This also turns on the rear power outlet. Turning to LOCK keeps power on and disables the operating switches so no one can interfere with the operation of the processor. The operator can still use the data switches to supply information to the program. If the CPU stops, the function switches are enabled.

1. Indicator Lights

A few indicator lights display useful information while the processor is running, but most change too frequently and are therefore discussed in terms of the information they display when the processor is stopped. The address lights display the contents of the program location counter (PC). The numbered data lights display the data written in the last memory reference. FETCH, DEFER, and EXECUTE, are the state indicators. They specify the type of cycle (state) the processor will enter if operations are continued by pressing the CONTINUE or MEMORY STEP switch. The indicator meaning is true when the light is lit.

a. RUN

The processor is in normal operation. The CPU is executing instructions or data is being transfered via the data channel. When the computer stops the light goes out. In RUN only switches STOP and RESET are enabled.

b. ION

The program interrupt capability is enabled (The Interrupt-On flag is 1).

c. FETCH

The next CPU cycle will be used to obtain an instruction from memory.

d. DEFER

The next processor cycle will be used to fetch an address word in an indirectly addressed memory instruction.

e. EXECUTE

The next CPU cycle will be used to perform an instruction. This next cycle will be used to reference memory for an operand in a move data or modify memory instruction.

2. Operating Switches

All of the switches in the bottom row except STOP and RESET are interlocked so they have no effect if RUN is lit. The four pairs of switches at the left are for depositing data in the accumulators and examining their contents. Lifting a switch up loads the contents of the data switches into the specified accumulator; pressing it down displays the contents of the accumulator in the data lights.

a. ACCUMULATOR DEPOSIT

The left-hand four switches reference the four CPU accumulators and are numbered 0-3 from left to right. Each switch affects only its corresponding accumulator (AC). When one of these switches is pushed up, the current setting of the data switches is deposited into the appropriate accumulator. The data lights display the new contents of that AC.

b. ACCUMULATOR EXAMINE

When one of these switches is depressed, the contents of the corresponding accumulator are displayed in the data lights.

Example

If the operator wishes to load AC0 with 126440 and AC1 with 063610; the procedure is:

-Turn the Power switch to ON. The FETCH light will turn on.

-Set the data switches to 126440.

-Press AC0 DEPOSIT. The data lights will read 126440. The carry and address lights can be ignored.

-Set the data switches to 063610.

-Press AC1 DEPOSIT. The data lights will read 063610. The carry and address lights can be ignored.

-The contents of AC0 are checked to ensure the data was entered correctly by pressing AC0 EXAMINE. The data lights will read 126440.

-Similarly the contents of AC1 are checked by pressing AC1 EXAMINE. The data lights will read 063610.

c. START

When this switch is pushed up, the START function is performed. The address indicated by data switches 1-15 is placed in PC and sequential operation of the CPU begins there. The FETCH and RUN indicator lights are turned on.

d. CONTINUE

When this switch is depressed, the CONTINUE function is performed. Sequential operation of the processor continues from the current state of the computer.

e. RESET

When this switch is pushed up, the RESET function is performed. The CPU is stopped after completing the current processor cycle. The flags in all Input/Output (I/O) devices are cleared, the 16-bit priority mask, the Interrupt-On flag, and all Busy and Done flags are set to 0 and the RTC is set to line frequency. Information deposited in an accumulator from the console is displayed in the lights but is not actually entered into the accumulator until the CPU performs some other operation. Therefore pressing RESET after an ACCUMULATOR DEPOSIT prevents the data from actually reaching the AC.

f. STOP

When this switch is pushed down, the STOP function is performed. The CPU is stopped after completing the current instruction and before executing the next instruction. If an I/O device requests an interrupt during the execution of the current instruction, it is serviced before the CPU is stopped. All outstanding data channel requests are honoured before the CPU is stopped. After the processor stops, the address lights display the address of the next instruction to be executed and the data lights display the current contents of the memory bus. If the current instruction contains an infinitely long indirect addressing chain or there are continuous data channel requests, pressing STOP will not stop the computer. A RESET will be required.

g. DEPOSIT

When this switch is pushed up, the DEPOSIT function is performed. The current setting of the data switches is placed into the location addressed by the current value of the program counter. The updated value of the altered word is displayed in the data lights.

h. DEPOSIT NEXT

When this switch is depressed, the DEPOSIT NEXT function is performed. The program counter is incremented by one and the current setting of the data switches is placed into the word addressed by the updated value of PC. The updated value of PC is displayed in the address lights and the new contents of the altered location are displayed in the data lights.

i. EXAMINE

The address contained in data switches 1-15 is loaded into PC and displayed in the address lights. The contents of the word addressed by PC are then read and displayed in the data lights.

j. EXAMINE NEXT

The current value of PC is incremented by one and the new value is displayed in the address lights. The contents of the work addressed by the updated PC are then read and displayed in the data lights.

(© Data General Corporation 1976. Reproduced from HOW TO USE THE NOVA® COMPUTERS by permission of Data General Corporation, Southboro, MA)

Example

If the operator wishes to load the Table 1 data starting at absolute locations 17757; the procedure is:

-Turn the Power switch to ON. The FETCH light will turn on.

-Set the data switches to 017757.

-Press EXAMINE. The address lights will read 017757. Ignore the carry and data lights for now.

-Set the data switches to 126440.

-press DEPOSIT. The data lights will read 126440. The address lights will read 017757 and carry can be ignored.

-Set the data switches to 063610.

-Press DEPOSIT NEXT. Note that the address lights have been incremented to 017760. The data lights will read 063610 and carry can be ignored.

-Set the data switches to 000777.

-Press DEPOSIT NEXT. The address lights will read 017761. The data lights will read 000777 and carry can be ignored.

-To verify that the data was entered properly, set the data switches to 017757.

-Press EXAMINE. The address lights will read 017757. The data lights will read 126440 which confirms what was intended got entered, and carry can be ignored.

-Press EXAMINE NEXT. The address lights will read 017760. The data lights will read 063610 and carry can be ignored.

-Press EXAMINE NEXT. The address lights will read 017761. The data lights will read 000777 and carry can be ignored.

If a mistake is made entering the contents of a location, the procedure is:

-Set the data switches to the address to be corrected.

-Press EXAMINE, this sets the PC.

-Set the data switches to the correct contents of the desired address.

-Press DEPOSIT. The address lights and data lights will indicate the location and its new contents.

Table 1 EXAMPLE DATA

LOCATION	DATA
17757	126440
17760	063610
17761	000777

k. INSTRUCTION STEP

When this switch is pushed down, the INSTRUCTION STEP function is performed. The instruction contained in the word addressed by the current value of the program counter is executed and then the CPU is stopped. The address lights display the updated value of PC and the data lights display the contents of the memory bus. The meaning of the data displayed depends on the instruction as follows:

LDA, STA, ISZ, and DSZ display the operand.

JMP and JSR for direct mode display the instruction, for indirect mode display the effective address.

Arithmetic and logical instructions display the instruction.

Input/Output instructions display the data.

The mnemonics LDA, STA, ISZ, DSZ, JMP and JSR are Assembly language instructions that are explained in more detail in Chapter IV.

l. MEMORY STEP

When this switch is pushed up, the MEMORY STEP function is performed. The CPU performs a single processor cycle and then stops. At completion the lights indicate the next state to be executed. The address lights display PC and the data lights display the data for the last memory step. Changing the contents of an AC between memory steps may destroy information necessary for the execution of the remainder of the instruction.

m. PROGRAM LOAD

When this switch is pushed up, the PROGRAM LOAD function is performed. The contents of the read-only memory (ROM) bootstrap are placed in memory locations 0-37 (octal), then the RUN light is turned on and normal operation is begun at location 0.

3. Exercise 1

At this point it would be wise to become familiar with the computer and console operations by completing the following exercise.

This exercise is designed to familiarize the user with the operator's console and to introduce two techniques for loading the BOOTSTRAP loader program which is part of system initialization (Appendix D). Either of these two techniques can be used in place of the normal initialization procedure described in Appendix A , if the cassette transports are not available. Before proceeding the reader should become familiar with Appendix B which describes console procedures. Before starting to enter the Manual BOOTSTRAP below, complete the machine code program in Section A of Appendix D by filling in the appropriate XX and dd values.

Manual Bootstrap

This is the most basic technique an operator can use for initialization. It requires only the basic computer without the PROGRAM LOAD switch and the ASR 33 teletype to operate:

1.Turn on main power.

2.Enter the manual BOOTSTRAP starting at location 017757. Section A of Appendix D explains the BOOTSTRAP.

3.Turn the TTY power switch to LINE.

4.Mount BINARY LOADER paper tape 091-000004-04 in the TTR.

When loading paper tape, place the leading end in the read station and set the remainder on the floor immediatly below there, clear of obstacles. The TTR switch should be at FREE while loading. Feed the blank leader past the read station by hand and stop with one or more blank frames before the data. Check that the data is program and not just an identification code. This is done before loading by inspecting the tape for a hole pattern that can be read as the tape identification number. Set the TTR switch to START. Section C1 of Appendix D explains the BINARY LOADER. Section B of Appendix C explains the TTR.

5.Execute the BOOTSTRAP program by setting the data switches to 017770 and pressing RESET and START. The BINARY LOADER will be read into core. The address lights will read 017776. The data lights will read 063077. At this point the system is initialized and the operator can use the BINARY LOADER to load any absolute binary paper tape appropriate for what he intends to do. In subsequent sections of this thesis the operator will learn what programs might be appropriate.

6.In order to demonstrate several other console switch functions, let's restart the manual BOOTSTRAP in a slightly different way. Since it has already been entered in memory, set the data switches to the start address 17770 and press RESET.

7.Mount the BINARY LOADER paper tape in the TTR. Ensure that the first data frame is not past the read station.

8.Press EXAMINE to set the PC to the start address (017770).

9.By repetitively pressing MEMORY STEP trace the progress of the BOOTSTRAP execution until address 017766 is about to enter the FETCH cycle. Two frames of the BINARY LOADER paper tape will be read.

10. Now repetitively press INSTRUCTION STEP and trace the progress of the BOOTSTRAP execution until the address 017766 returns. Another frame of the BINARY LOADER paper tape will be read.

11. Pressing CONTINUE will read in the remainder of the BINARY LOADER tape. The system has been re-initialized.

Automatic Bootstrap

This is a slightly more sophisticated initialization technique that requires the basic computer with the PROGRAM LOAD switch and the ASR 33 teletype to operate:

1. Set the data switches to 000010; this specifies TTR input.

2. Mount the BINARY LOADER paper tape 091-000036 in the TTR. The different identification number from the manual procedure indicates that this is a different version of the program written specifically for the Automatic BOOTSTRAP. The SELFLOADING BOOTSTRAP AND BINARY LOADER program is explained in Section C2 of Appendix D.

3. Press RESET and PROGRAM LOAD. The paper tape will be read into memory. The address lights will read 000121 and the data lights will read 063077. At this point the system is initialized and the operator can use the BINARY LOADER to load any absolute binary paper tape appropriate for what he intends to do.

4. The automatic BOOTSTRAP can be loaded with the power switch in LOCK and the key removed.

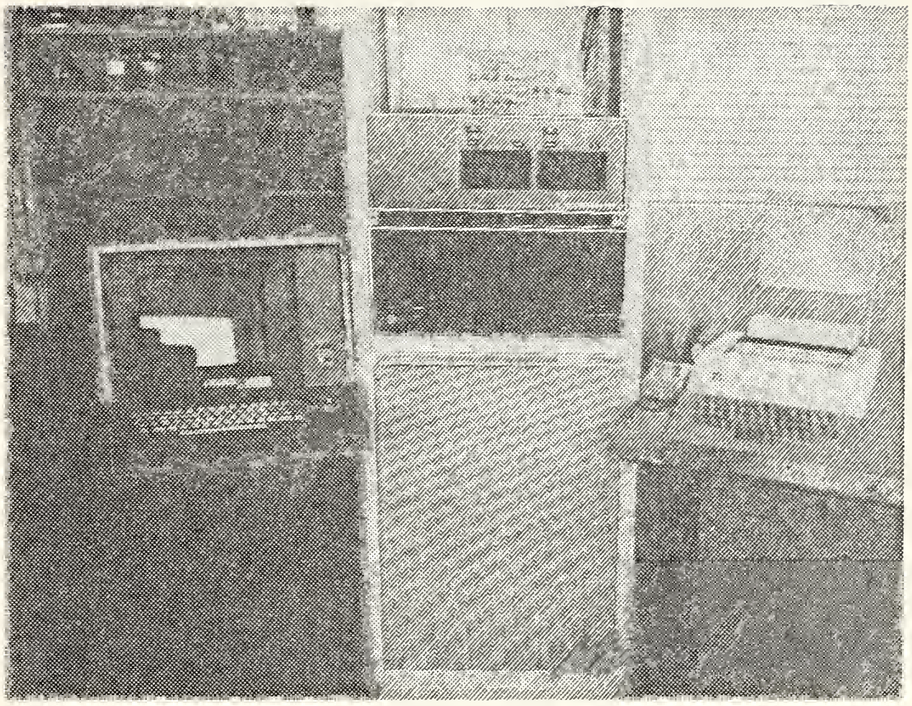


Figure 1 - NOVA® 800 SYSTEM CONFIGURATION

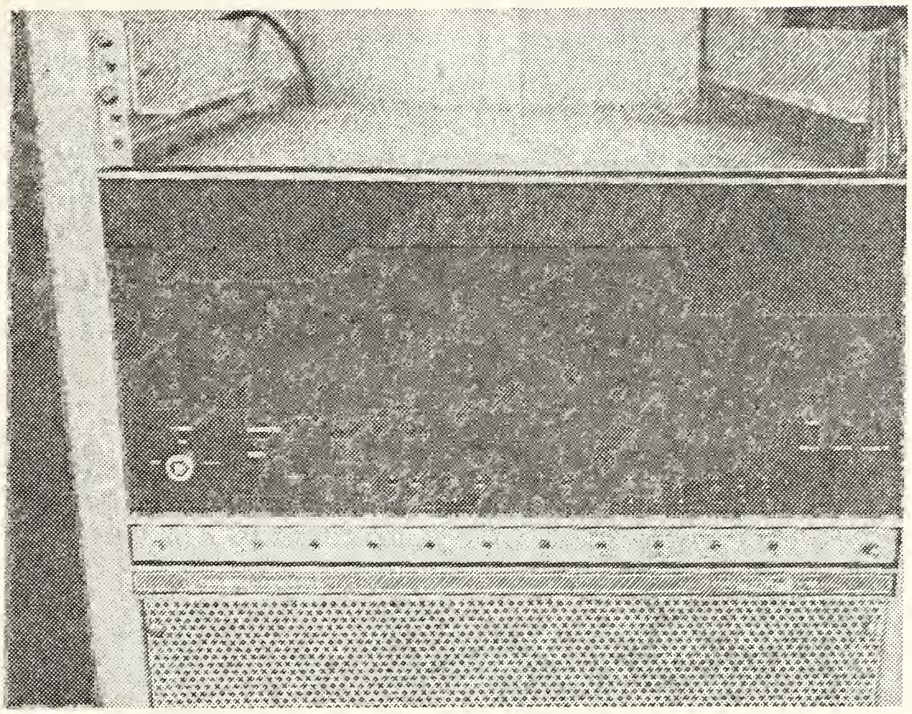


Figure 2 - NOVA® 800 OPERATOR'S CONSOLE

III. SOFTWARE

Due to the present limited memory capacity (8K) the convenience of higher level languages like Algol, Basic and Fortran is not available. The present working code is primarily Assembly language with some knowledge of machine language being of benefit.

Several Data General Corporation (DGC) programs are available on paper tapes. An index of their identification numbers is included as Appendices V-Z. Further documentation appears in the list of manuals in Appendix U.

A. THE STAND-ALONE OPERATING SYSTEM

The NOVA® 800 is programmed within a software environment called the STAND-ALONE OPERATING SYSTEM (SOS). By using certain programs within a particular SOS it is possible to:

-initialize the computer

-allow a desired program on a specified peripheral device to be read into or written from memory

-create a new program by inputting Assembly language code from the teletype

-correct mistakes or change existing programs

-translate the Assembly language source code into a relocatable binary (RB) machine language code

-translate the RB code into absolute locatable binary (AB) code in memory that is a suitable form for understanding and executing by the CPU as a program.

The particular programs and functions available in any SOS are decided at the time of its creation by operator selection of appropriate utility programs which when combined will fulfill the requirements of the specified hardware configuration in which it will be used. If a cassette driver is available, the selected SOS utilities may be stored on a master tape which can be called the SOS master cassette.

B. STEP 1 IN PROGRAM CREATION

The SOS utility programs are what the programmer must use to create a program. To produce a file of source program code he must know and be able to use the following utility programs; the CORE IMAGE LOADER/WRITER, the COMMAND LINE INTERPRETER, and the SYMBOLIC TEXT EDITOR. The programmer uses the first two programs to load in the EDITOR so that programs can be created and saved on a cassette.

Due to its limited size, the PROGRAM LOAD hardware BOOTSTRAP is used to load another loading routine. For the cassette system this other loading routine is called the CORE IMAGE LOADER/WRITER and must be on file 0 of a cassette mounted on unit 0 (Section C3 of Appendix D). However, it fulfills the same function as a BINARY LOADER in the paper tape environment.

The distinction of paper tape from cassette environment is purely arbitrary to the CPU since the Large Scale Integration (LSI) hardware BOOTSTRAP uses the data switches to determine the device code.

On the SOS master cassette, programs are loaded into sequential files starting at 0. Therefore, for SOS, file 0 contains the CORE IMAGE LOADER/WRITER. [Ref. 12]

1. Core Image Loader/Writer

The CORE IMAGE LOADER/WRITER (CIL/W) program on the SOS master cassette is identical to paper tape 091-000067-02. It performs two utility functions: it loads core image files from cassette tape into core and produces core image files on cassette tape [Refs. 12 and 13]. The CORE IMAGE LOADER/WRITER program works only with cassettes.

The CORE IMAGE LOADER/WRITER can be bootstrapped from file 0 of the SOS master cassette on unit 0. When first loaded, the tape must be rewound manually. The normal loading procedure is described in Appendix A.

The Loader/Writer is read into page zero (0-377) initially and then relocates itself to the last 400 (octal) locations in core. After relocation a prompt # on the teletype indicates that the CORE IMAGE LOADER/WRITER is ready. Once it is in core the Loader may be restarted by setting the data switches to the last memory address, pressing RESET, and then START (For 8K set 017777).

The # symbol indicates the Loader is waiting for the operator to respond with a cassette unit number (0-7) and a file number (0-99) separated by a colon. Specifying unit 0 is optional. The indicated cassette file is loaded into memory upon command termination by a teletype CARRIAGE RETURN. If data switch 0 on the console is 1, the program will halt on completion of the load. If the switch is 0, control is passed to the loaded program linked through location 405.

If the Loader encounters a non-recoverable error while trying to load a file, it will type *ERR and halt with a code in AC0. The error codes are explained in Section A of Appendix Q. If rewinding and substituting a different cassette tape does not clear the error condition, a hardware fault is indicated.

The CORE IMAGE WRITER operates in a manner similar to that of the Loader. When the Writer is started it outputs a # prompt and waits for specification of a device number and a file number separated by a colon. After typing the unit and file numbers followed by a CARRIAGE RETURN, the operator receives NMAX as a prompt. The operator responds to the prompt message NMAX by typing the highest core address (octal) whose contents he wants written into the cassette file he specified initially. The program always starts at absolute address zero and after completing a successful write, the message OK is typed and the routine HALTS. Non-recoverable errors are handled the same as with the Loader. [Ref. 12]

Example

After loading the CIL/W the user receives the prompt #. When creating programs he selects the next SOS utility program that is appropriate for his stage in the program creation. He may choose to load the SOS utility programs by the COMMAND LINE INTERPRETER (CLI) mnemonic load commands. To do this, the CLI must be loaded. To load the CLI, which is on file 1 of the SOS master cassette, the operator types 0:1 and CARRIAGE RETURN after the prompt #. The CLI prompt R indicates that it is ready for a command. The command line at this point will look like:

```
# 0:1 (CARRIAGE RETURN)
R
```

2. Command Line Interpreter

The COMMAND LINE INTERPRETER (CLI) is a utility program which performs certain file maintenance chores for the user and implements mnemonic loading of other utility programs from a Master tape. The CLI accepts commands typed by the operator on the teletype. When it is ready to receive a command a teletype prompt of R and CARRIAGE RETURN is sent.

In order to use the CLI, the CORE IMAGE LOADER/WRITER must be in core, and the Master cassette must be on CT0.

The CLI can be loaded using the CORE IMAGE LOADER/WRITER. Many CLI commands cause it to be overwritten in core and a reload is required to return to the CLI operation. CLI commands are explained in Appendix F. [Ref. 12]

Example

After the CLI is loaded the programmer is ready to load the EDITOR program. The user types EDIT and CARRIAGE RETURN after the CLI prompt R. When the EDITOR is ready to accept commands the symbol * is displayed on the teletype. The command line at this point will look like:

```
R EDIT (CARRIAGE RETURN)
*
```

3. Symbolic Text Editor

The TEXT EDITOR is used to create or modify ASCII files. The prompt * is given when the program is ready to accept editing commands. The EDIT instructions are explained in Appendix G.

Once loaded the TEXT EDITOR is self-starting and provides over 6,000 characters or six pages of normal symbolic source text (for 8K).

The NOVA® editing commands are divided into groups, those that input and output the contents to and from the edit buffer and those that modify the contents contained in the buffer. Input commands read a program (or part of a program) into the buffer for later editing. The edit commands are used to modify the contents of the buffer. After updating the buffer, the corrected program may be placed onto a file by the output commands. Several commands can be specified at one time by separating them with the symbol $ which is caused by striking the escape (ESC) key once. A command or string of commands is executed by striking the ESC key twice ($$).

The command structure is versatile enough to allow changes at the character level as well as the line level. Line numbering is continually updated as lines are inserted and deleted. String searches provide a convenient method of locating characters. [Refs. 14 and 15]

Example

Now that the CLI has loaded the EDITOR a program can be created. However, remember it is not always necessary to load SOS utility programs using the CLI. If the procedure in Appendix A has been followed, the CORE IMAGE LOADER/WRITER will do the same thing by loading for example, file 2 (the EDITOR), as in the following command line:

```
# 0:2 (CARRIAGE RETURN)
*
```

Most utility programs will reinitialize the CORE IMAGE LOADER/WRITER by the CTRL C command. The EDITOR uses the H command (Appendix E). In those cases where the SOS has been halted (by some catostrophic error) the standard data switch setting of 017777 forces the system to reinitialize the CIL/W when the operator presses the console switches RESET and START. Since the system automatically restarts by loading (executing) the CORE IMAGE LOADER/WRITER, the above technique is often more convenient for loading the SOS utility programs.

After the EDITOR prompt *, the programmer must ensure that a scratch tape is mounted on unit 1. The steps in creation are to open a write file on the first available file (file 0 on a scratch tape), insert the necessary source code into the edit buffer, terminate the insert command by striking the ESC key twice, type the buffer contents to verify they are correct, save the program on the output file, close the edit buffer, open the saved file for reading, yank the file into the input edit buffer and type the buffer contents to confirm the correct program.

<u>Some confusion can develope over the symbol $. The ESC key prints the $ when struck and there is also an independent character $. The $ in an edit command string always signifies the ESC key. Any other occurence means the $ key on the TTY.</u>

Example

To produce a source tape file by the creation steps listed above, the following command lines are typed on the teletype:

```
*GWCT1:0$$                        (open CT1:0 for writing)
*I (CARRIAGE RETURN)              (begin inserting source code)
program ‹carriage return›         (inserted by operator)
$$                                (terminate insert command)
*T$$                              (type the buffer contents)
PROGRAM                           (contents of the buffer)
*B$P$GC$$                         (record and close the buffer)
*GRCT1:0$$                        (open CT1:0 for reading)
*Y$T$$                            (input and type buffer)
PROGRAM                           (program listing)
*                                 (ready to continue)
```

When attempting to open a file for reading (GR) or writing (GW) an error will be indicated by I/O ERROR followed by the two digit system error number. The command CTRL A will reinitialize the SYMBOLIC TEXT EDITOR without destroying the contents of the edit buffer. These errors are often caused by the operator not rewinding the cassette when it is first mounted to check that it is seated properly.

4. Exercise 2

Since at this point it is assumed that the reader is learning the system, follow the steps in the preceeding paragraphs to create a source file containing the Teletype Output Example Program provided in Section A of Appendix R. This program will be used as an example throughout the sections on assembling, loading and executing procedures which follow. When you execute this Assembly language program later, it will print the following message on the teletype:

```
 CONGRATULATIONS!

   YOU HAVE COMPLETED YOUR FIRST PROGRAM CREATION.
```

C. STEP 2 IN PROGRAM CREATION

After the programmer has written an assemble language source file it must be translated into a binary code that the CPU can understand. This involves two procedures. The first procedure is a translation into relocatable binary (RB) code that does not have all of its addresses resolved and therefore cannot be executed by the processor. This translation into addresses relative to the first line of programming is done by the EXTENDED ASSEMBLER.

1. Extended Assembler

The EXTENDED ASSEMBLER, like the basic ASSEMBLER, converts symbolic source statements into machine language code. In addition to basic ASSEMBLER features the extended version provides relocation, interprogram communication, conditional assembly and more powerful number definition facilities. [Ref. 12]

The EXTENDED ASSEMBLER will assemble one or more ASCII source files to a relocatable binary file with an optional listing file. Input files are assembled in the order they were specified in the command line. A cassette tape unit may not be used for both input and output, nor may it be used for more than one output file. More than one input file is allowed from the same unit.

The teletype prompt ASM indicates the EXTENDED ASSEMBLER is ready to accept commands. The operator must not insert a space before the first entry following ASM because it is provided by the ASSEMBLER program and command format errors cause unpredictable results. The ASSEMBLER does not use the ESC key so that all $ symbols are understood to be the corresponding $ key on the TTY. These commands are explained in Appendix H. [Ref. 13]

Example

So far in the example program, only an Assembly language source file has been created. This next step will create another file that must be on a different cassette. Now there is a problem. We have two cassette units, CT0 has the SOS master tape, CT1 has the new source tape and a new cassette is required. Since the SOS master cassette is only used at the time a utility program as loaded into memory it is the only one available for the new file. The following procedure is to be used with caution:

-Mount the Assembly language source tape on CT1 and press REWIND.

-Mount the SOS master tape on CT0 and press REWIND.

-Initialize the system by the procedure in Appendix A.

-Load the EXTENDED ASSEMBLER. The command line will look like:

```
    # 0:3 (CARRIAGE RETURN)
    ASM
```

-Mount the new scratch tape on cassette unit 0 (CT0) and press REWIND.

-If the assembly source tape is file 0 on CT1 and the RB file is to be saved on file 0 of CT0 and a teletype listing is desired, the command line for a normal two pass assembler will be:

```
ASM 1 CT1:0  CT0:0/B $TTO/L (CARRIAGE RETURN)
          (this command is explained above)
LOCATION (MACHINE CODE) (SOURCE CODE)
          (THESE COLUMNS ARE ASSEMBLER OUTPUT)
LABEL DIRECTORY     (this list is explained below)
ASM       (ready to continue)
```

Remember that the ASM automatically supplies the first entry space; violating the given command format spacing may cause errors.

During an ASSEMBLER listing several symbols are inserted to inform the programmer what kind of addressing has been generated. Table 2 summarizes the symbol flags and their meanings.

Table 2 ASSEMBLER FLAGS

ADDRESS FLAG	MEANING
blank	Address word is absolute
-	Address word is page zero relocatable
'	Address word is normally relocatable

CONTENTS FLAG	MEANING
blank	Contents of word are absolute
-	Contents of word are page zero relocatable
=	Contents of word are page 0 byte relocatable
'	Contents of word are normally relocatable
$	Storage word reference a byte disp. external

The LABEL DIRECTORY is an alphabetical list of the LABELS that have been created and their relative addresses. This can be used for debugging program errors by adding the relative address to the entry address given at load time to obtain the absolute location. Section B of Appendix R is the teletype listing of the example program's assembly.

D. STEP 3 IN PROGRAM CREATION

The final step in program creation is the second procedure mentioned for translating the source code. This process takes the RB file from the ASSEMBLER output and replaces all the relative addresses with absolute memory locations. The resulting new absolute locatable binary file (AB) is in a core-image form that is executable by the processor. The translating routine is called the EXTENDED RELOCATABLE LOADER.

1. Extended Relocatable Loader

The RELOCATABLE LOADER produces an absolute binary core-image (or save) file from relocatable binary files. The loader accepts any number of relocatable binary files as input, resolves external displacements and normal externals, and maintains an entry symbol table that can be printed on demand. Extensive error detection logic is provided to prevent various fatal and non-fatal errors. A successful load is indicated by the prompt OK. The Loader enters ZREL user programs beginning at absolute address 50 (octal), and NREL user programs starting at location 440 (Fig. 3).

The mnemonics ZREL and NREL are Assembler language pseudo-operations which indicate the memory area the programmer wants the routine loaded into. Assembler addressing is explained in Appendix L. The first 377 (octal) locations in core are called page zero addresses because they can be addressed directly (mode 0). This allows any locations defined in this area to be accessed more easily than any others because no type of indirect indexing techniques are necessary. These other addresses are located in the NREL area of memory and must be accessed by indexing a location that holds a pointer address that is within 200 (octal) locations of the location desired. The teletype prompt RLDR indicates the RELOCATABLE LOADER is ready to accept commands. Commands are explained in Appendix I. [Ref. 13]

Example

The same requirement for the new file has again created a prcblem. The procedure for managing the cassettes in completion of the assemble is as follows:

-Move the RB tape from CT0 to CT1 and press REWIND.

-Replace the SOS master tape on CT0 and press REWIND.

-Load the RELOCATABLE LOADER.

The command line will be:

```
    # 0:4 (CARRIAGE RETURN)
    RLDR
```

-Mount the new scratch tape (AB)) on CT0 and press REWIND.

-If the RB source tape is file 0 on CT1 and the AB file is to be saved on file 0 of CT0 and a teletype listing is desired, the command line will be:

```
RLDR CT1:0 CT0:0/S $TTO/L (CARRIAGE RETURN)
          (this command is explained above)
LIST OF INPUT PROGRAMS
          (this list is explained below)
   NMAX ------ (next NREL address available)
   ZMAX ------ (next ZREL address available)
    EST        (not used)
    SST        (not used)

  LIST OF ENTRY POINT ADDRESSES
          (this list is explained below)


OK        (relocatable loading completed)
```

<u>Remember that the RLDR automatically supplies the first entry space; violating the given command format spacing may cause errors.</u>

The LIST OF INPUT PROGRAMS contains the titles of the referenced file programs in the order they were loaded. NMAX is the first available normal relocatable address and ZMAX is the first available page zero address. This gives an indication of how much memory has been used. EST and SST are parameters used in a disc operating system and are not used in SOS. ENTRY POINTS are the first locations for executable code for each program in the order in which they were loaded. The RLDR teletype output for the example program is Section C of Appendix R.

E. STEP 4 IN PROGRAM CREATION

If the loaded program was coded with an end pseudo-operation that has the program title, the RELOCATABLE LOADER will generate coding that forces the system execution to continue at the entry point for that routine once the load is complete. Therefore the execution of any program can be achieved by simply causing it to be loaded into memory. However if the control is not coded to be passed to the program, the operator must know the entry address of the program and set the PC via the data switches. If the normal routine is followed the operator executes the CORE IMAGE LOADER program and in response to the prompt # he inserts the unit and file number of the program he wants executed. For a program on file 6 of cassette unit 0 the command line will be:

```
#0:0 (CARRIAGE RETURN)
```

Section D cf Appendix R shows the execution of the TTO example program.

1. Exercise 3

The SOS provides very convenient access to the EDITOR and other functions. This exercise is designed to demonstrate the facility with which SOS can be used. Remember SOS is just a convenient software arrangement on magnetic tape, made up from paper tape programs that can also be brought into memory individually by the procedures demonstrated in Exercise 1.

1.Follow the system initialization procedure in Appendix A.

2.Use the CORE IMAGE LOADER to verify the contents of the SOS are as indicated by receiving the correct Prompt message. The procedure is indicated in the example in section B1 of Chapter III.

Table 3 SOS PROMPT MESSAGES

FILE	PROMPT	PROGRAM (CALL)
0	#	Core Image Loader/Writer
1	R	Command Line Interpreter (CLI)
2	*	Symbolic Text Editor (EDIT)
3	ASM	Extended Assembler (ASM)
4	RLDR	Extended Relocatable Loader (RLDR)
5	LFE	Library File Editor (LFE)
6	SYSG	SYSGEN (SYSG)

3.Reload the utilities using the CLI.

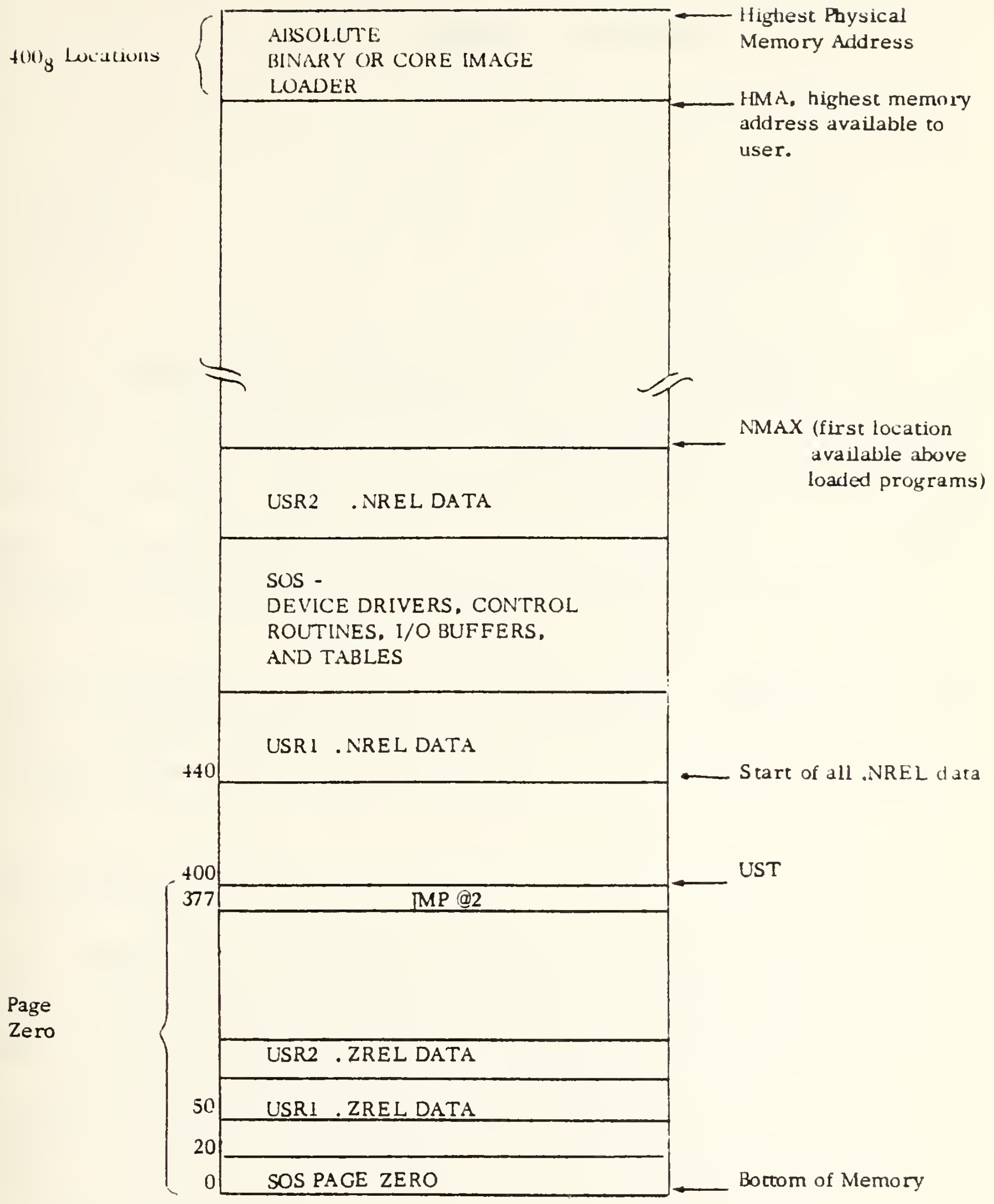


Figure 3 - MEMORY SPACE ALLOCATIONS

IV. ASSEMBLY LANGUAGE

A. FORMAT

The ASSEMBLER program allows programmers to write programs in a symbolic mnemonic language instead of direct numeric machine code. The NOVA® ASSEMBLY language is free format. Within broad limits, the programmer is free to determine the format of the listing of his program.

The ASSEMBLER program automatically segments the TTY listing into 11 inch pages with pagination and the title in the upper left corner as follows:

```
---
 0001 TITLE
```

A new page can be forced at any point in the listing by the FF key. The source program is divided into character strings called lines by the requirement that every statement must be terminated with a carriage return (CR). The ASSEMBLER program provides a predetermined set of tabulation points at columns 1, 9, 17, 25 etc. Striking CTRL I on the TTY keyboard advances the spacing to the next tab setting that ensures one space separation from the last entry. All redundant spaces, tabs, and CARRIAGE RETURNS are interpreted only for listing format.

This allows the programmer to adopt a convenient general instruction format which separates a line into four possible fields:

LABEL: OPCODE OPERAND ;COMMENT

The ASSEMBLER recognises all ASCII characters except NULL, LF, RUB OUT and FF. The FF does not generate computer instructions, but it can be used to affect the source listing format. The characters . (when used alone), @, ", and # have special significance.

. indicates the current location or contents of PC.
@ places a 1 in the indirect bit of instruction (bit 5) and address words (bit 0).
" replaces the next character by its ASCII code.
(except RUB OUT, LF, FF, or NULL)
# places a 1 in the NO LOAD bit (bit 12) of an arithmetic or logical command.

A LABEL is a name symbol of one or more alphanumeric characters that represents the location at which it is defined. The symbol . is also legal in a LABEL if it does not occur by itself. The first character must be a . or a letter and all LABELS are terminated by a colon (:). The first five characters of any LABEL are all that are used by the ASSEMBLER and must be distinct from all other LABELS. LABELS are optional.

The OPCODE is separated from the LABEL by the colon, so spaces are not necessary except for readability. The particular OPCODE is what decides whether the location is intended as data or an instruction. However the real distinction between data and instructions is whether the binary code can be interpreted by the CPU. Appendix M summarizes the Assembly language instruction mnemonics. They can be separated into three general classes (Fig. 4).

<u>Memory Reference Instruction Class (MRI):</u> This class contains instructions which move data between the accumulators and memory, instructions which modify memory, and jump instructions which alter the program flow of execution. Appendix L summarizes the machine code and Assembly language formats.

<u>Arithmetic and Logical Class (ALC):</u> This class contains instructions which manipulate the contents of accumulators and the Carry flag and instructions which perform all the arithmetic and logical functions between accumulators. Appendix L summarizes the machine code and Assembly language formats.

<u>Input/Output Instruction Class (I/O):</u> This class contains instructions which move data between the accumulators and the I/O peripheral device and instructions which only control those devices. Appendix L summarizes the machine code and Assembly language formats.

An instruction OPCODE is separated from the OPERAND by at least one space, comma or TAB. A space is recommended for better field distinction. There can be up to three OPERANDS, each separated in a similar manner. Because spaces are transparent (undetected by the ASSEMBLER) a zero OPERAND must be explicitly defined when it precedes a non-zero OPERAND. Unspecified OPERANDS are assumed zero. It is recommended that commas be used for OPERAND separators.

The optional COMMENT is the last thing on any line before the CR. It must be started with a semi-colon (;) which will separate it from the OPERAND. A complete line of COMMENT or a continued COMMENT must still start with the semi-colon. Although the full 72 characters on the teletype line can be used for COMMENT, it should always be remembered that the ASSEMBLER program lists the source code shifted over to the right to allow for the machine code. This limits the useful line length to 56 characters. [Ref. 2]

B. INPUT/OUTPUT

Input/Output (I/O) is the process of moving information in a computer system between the central processing unit and peripherals such as the teletype,A/D converter, D/A converter and cassette transports. Peripherals can serve two main purposes, they provide the computer with a means of communicating with its surroundings (TTY, A/D and D/A) and they can supplement main memory with a secondary storage capability (CT0 and CT1).

The direction of all information transfers on the I/O bus is defined relative to the computer. Output always refers to moving information from the computer to a peripheral; input always refers to moving information from a peripheral to the computer.

The information transferred between a computer and a peripheral can be classified as status, control and data. Status information indicates the peripherals state; busy or ready, or operatingimproperly. Control information is used to tell the peripheral what to do. Data is the information exchanged during reading, writing, storing or processing.

The amount of information transfered, one bit, eight bits (byte), sixteen bits (2 bytes or 1 word), or a group of words (block) depends on the peripheral device.

Information is transferred in one of three ways, under direct program control (TTY, A/D and D/A), under single word Interrupt control (TTY, CT0 and CT1) or under data channel Direct Memory Access (CT0 and CT1), depending on the peripheral and the I/O instruction used.

During input the peripheral's controller places the data in one of three possible holding registers (A, B, C) depending on the device, signals the CPU the data is ready and the processor brings the data into the computer. During output the CPU sends data to an output holding register in the device and the device signals when it is ready for the next data output. For the teletype, only one holding register (A) is involved, the device code is 10 (Appendix O) and two flip-flops (Done and Busy), associated with that device, achieve the controlling functions. The three commands NIO, DOA, DIA can be used with the standard I/O Skip instructions of Table 19, to achieve communication with the teletype (Section A of Appendix C).

The NIO instruction may sometimes be used to set the device in some desired state by appending the appropriate control designator (Table 18).

Normal input is achieved with a DIAS AC,TTO command. The input data is placed in AC. Notice the mnemonic TTO or TTI is recognised by the Assembler program as meaning device code 10. Usually the second Assembly language argument is the number for the device code. A word of caution at this point, the DIAS instruction will input whatever data is in the input holding register of the TTY before it enables the device so that the user can strike a character key. The programmer must also realize that the TTY does not automatically print the characters struck by the operator. This requires that the programmer output the input character to make it appear that the struck character was printed. This technique is called echo printing.

Normal output is achieved with a DOAS AC,TTO command. The data in AC is placed in the output holding register of the TTY and the appended S enables the TTY to print it. The data is preserved in the accumulator. This allows the echo print routine to consist of a DIAS AC,TTI for input, then a DOAS AC,TTO for echo print and the program can still operate on the input character that remains in the AC.

The A/D and D/A were incorporated in the system in a previous thesis. Since this construction was an individual effort,, the only source of hardware wiring documentation is Reference 1. The A/D operates on device code 21 and uses the associated Done and Busy flip-flops in the normal manner (Appendix L). However the following use of I/O instructions is peculiar to this device interface.

First, the programmer loads the number of the input channel for the A/D into a selected accumulator.

Second , the programmer instructs the A/D to start a conversion cycle by issuing a DOCS AC,21 command. The appended letter S on the DOC command sets the Busy flip-flop and clears the Done flip-flop.

On completion of the conversion, approximately 20 microseconds later, the A/D will set the Done and clear the Busy flip-flops. At that time the programmer may issue a DIC AC,21 command to retrieve the converted data in an accumulator of his choice.

The D/A operates on device code 23 and does not require the use of Done or Busy flip-flops. It settles to 0.01 percent of final value within three microseconds. The present configuration is only connected to allow X or Y output selection by entering a 0 for channel X or a 1 for channel Y in the desired accumulator and executing a

DOB AC,23 command. The computer output data is transfered from the selected accumulator into the previously designated D/A output channel's holding register by a DOA AC,23 command. The D/A continuously outputs values corresponding to the register contents and therefore needs no direct start of conversion instruction.

1. Exercise 4

This exercise is designed to start the user learning the first essential step in computer communications. If programs can be written to allow some sort of output message at critical points in their execution then the user has some indication that they are executing correctly.

Using the TTO Example program in Appendix R, modify the buffer contents to output a message that contains the following information:

```
NAME,     RANK
STREET ADDRESS
CITY,    STATE
ZIP       CODE
```

Ensure that the edges are parallel and that the left margin is in column 9. The pseudo-operation Assembly language instruction .TXT is explained in Section C of this chapter. Section F of Appendix S contains the Assembler listing of a solution program for exercise 4.

C. PSEUDO-OPERATIONS

A special set of instructions called pseudo-operations (PSEUDO-OPS) are essential when creating a program. Although they generate no program instruction code they communicate important information to the ASSEMBLER and RELOCATABLE LOADER programs. These commands all begin with the symbol period (.). The PSEUDO-OPS are explained in the order they would occur in a program like the TTO EXAMPLE PROGRAM in Appendix R.

.TITL title

This command designates the five character title as the identifier for the program being created. The title will be repeated in the ASSEMBLER List of Input Programs Listing Pagination and Label Directory and in the RELOCATABLE LOADER List of Input Programs and List of Entry Point Addresses (Sections C and D of chapter III). If .TITL is omitted the utilities will substitute the title MAIN.

.ENT label list

This command resolves the addressing between programs. The programmer lists all of the labels that he wants to call that are in programs outside his own. To reduce confusion it is recommended that the title, first ENT label and first instruction to be executed in the program be identical. Separate subroutines must define their names as entry symbols so that outside calls can link addresses.

.EXTN

This is the command that relates internal program references to the .ENT location that they are addressing. A program calling a separate subroutine must state that its name is an external symbol.

.NREL or .ZREL

These commands instruct the RLDR where to start loading the program code when converting to absolute locations. The first zero relocatable (ZREL) program starts at location 50 and subsequent programs loaded at the same time start where the last program stopped until the ZREL area is full. Overflowing the ZREL area causes an error message. Normal relocatable code loads in a similar manner starting at location 440 (Figure 3). This is the first location after the ZREL area. Program types can be mixed.

.LOC address

This command allows the programmer to force the RLDR to start placing code at a specified location. This is the command to enter an Interrupt routine address into location 2 or a specific count into the AutoIncrement and AutoDecrement locations. The RLDR carries on loading from that address until told otherwise by a ZREL, a NREL or another LOC command.

.BLK count

This command tells the ASSEMBLER program to leave blank the number of words specified by count. This instruction is used to define I/O buffers as follows:

```
  BUFER:  .BLK  count
```

.TXT 'message'

This command stores the text message defined within any set of user designated symbols (quotation marks are suggested) in a block of words. Characters are stored in pairs with the left ASCII character code in bits 8-15 and the right character code in bits 0-7 as follows:

```
  NOTE:  .TXT  'ABC'
```

will give this ASCII code buffer:

```
    BA
 nullC
```

The coding of an actual buffer can be seen in the Assembler listing for the TTO EXAMPLE program which is in Section B of Appendix R.

.END start address

This is the last command in a program creation. It instructs the ASSEMBLER program to write a command at the end of the program that will cause the CORE IMAGE LOADER (actually the Binary Loader portion of it) to start executing at the start address location specified, after the load is completed. If the start address is omitted (.END) the loader will HALT on load completion. The unspecified start address is the type of .END used in subroutine programs.

D. PROGRAMMING SUMMARY

The preceeding discussion on I/O and PSEUDO-OPS and frequent reference to Appendix L on Assembly language formats and Appendix M on Assembly language codes, should allow the reader to understand the TTO EXAMPLE program of Appendix R.

The first section of the program, delineated by the full line of asterisks, consists of general comments to identify the program and aid the user/programmer to see what the routine does.

The .TITL, .ENT and .NREL pseudo-ops designate the title and only externally accessible label as TTOEX. The program is normally relocatable; i.e. the loading starts at location 440. That is why the entry point TTOEX is listed as 440 in the relocatable load. This procedure is recommended so that the limited page zero locations can be used by programs that may require them. Another alternative is to define all tables and data as page zero (using a .LOC) and place the program for NREL so that the data can be addressed in the direct mode. However short independent programs in page zero eliminate addressing mode difficulties.

The first LDA instruction is used to save the address of the output buffer in a register so that it can be manipulated by an index to step through the elements of the table. This common technique of using a pointer to an address is achieved by the definition just before the program ends:

```
 PBUF:  BUFER
```

The next LDA instruction is part of an incrementing loop that increases the buffer pointer count and steps through the text defined by .TXT while outputting the message to the TTO.

A common technique for terminating a program that transfers data, is to keep checking for a special code that will only occur once the program is to HALT.

The MOV# instruction is designed to do nothing (#) but it does skip the HALT instruction if non-zero data is found in AC0.

The SKPBZ instruction checks to see if the TTO is occupied with output. If the Busy flip-flop is set the

program executes the JMP .-1 instruction. Otherwise it skips and continues.

The JMP instruction has employed the special symbol . which indicates the present location. Decrementing the present address by 1 causes the JMP to return to the previous SKPBZ instruction to continue. This causes a tight loop to occur while the program waits for the teletype to be done so it can continue with the output.

The DOAS instruction causes the character in bits 8-15 to be printed on the TTY. Since the next character is in bits 0-7 it is swapped into position for output while the other character is actually being typed.

The SKPBZ and JMP instructions are another pause while the program waits for the TTY to complete typing the first output character.

The second character is output by the second DOAS instruction. Again since there is some time delay in the mechanical motion of the teletype several instructions can be executed to reduce the waiting time.

The pointer is incremented to select the next buffer word and the program returns to the loop beginning by the last JMP instruction. Notice that because in this short program you can be certain the address of LOOP is within 377 locations of the JMP instruction, the actual location label can be used in the direct mode (ommitting the mode defaults to or 1).

The .END TTOEX pseudo-op designates the end of the program that is to be executed from location TTOEX on completion of loading.

E. INTERRUPTS

It should have been obvious in the TTO EXAMPLE program that all that looping and waiting was wasteful. The Interrupt facility provides a way of allowing the program to continue processing while a peripheral, which is far slower than the CPU, finishes its task.

When the peripheral finishes its task and sets the Done flip-flop this generates an Interrupt Request (if the device is wired for Interrupts). If the Interrupt On facility is enabled and if the Interrupt Disable mask bit for that device is 0 then the request is recognised. The CPU will service this interrupt when it completes the next instruction, if all DMA requests have been answered and if all higher priority peripherals (determined by who is physically closest) are answered.

Two locations in memory are automatically used during an Interrupt. The location where the program should return to continue after the Interrupt is saved in address 0 and the processor tries to execute an Interrupt processing routine whose start address is pointed to by the contents of address one. The processor routine must protect all accumulator contents and the carry so they can be restored prior to returning to the main program. It is the programmer's responsibility to clear the Done flip-flop when he wants to continue communication with that peripheral. When a device causes an Interrupt the Interrupt On flip-flop is disabled, so the programmer must reset Interrupt On if he desires that facility.

Example

The technique for programming an Interrupt is as follows:

-Place the address of the service routine in location 1

-Create a service routine that:

- -saves the accumulators and the carry
- -processes interrupts
- -clears the Done flip-flop
- -restarts the device if desired
- -restores the accumulators and the carry
- -enables Interrupt On
- -returns to the address contained in location 0

-Create a main program that:

- -initially enables the interrupt
- -clears device's Interrupt Disable mask bit
- -starts the device
- -continues processing

1. Exercise 5

Create a program that uses the Real Time Clock on an Interrupt basis to output a repeating count from 0 thru 9 at precisely 1 second intervals. Since no large amount of processing will be required in the main program a simple loop that does nothing will be sufficient. Check the timing by counting the period of several count cycles.

An example of this sort of technique without looping is included as Section G of Appendix S. The program INIT starts the clock the first time. INTRUP processes the interrupt and protects the accumulators and carry. SUPR is a general subroutine that allows a table of job routines that may be serviced by one real time clock. EXEC2 is a subroutine that types the count 0-9 on a one second basis.

F. PROGRAMMING THE CASSETTE UNITS

Programming the cassette units would be a lengthy and complicated task if carried out with the basic I/O instruction set that has been presented so far. Fortunately the STAND-ALONE OPERATING SYSTEM provides a set of I/O utility programs for communication with any peripheral in the system. For cassette programming the SOS commands are most convenient because they provide a functional read and write capability. All SOS commands have the following format:

```
.SYSTM
command
error return
continue return
```

The available commands are:

Table 4 SOS COMMANDS

COMMAND	MEANING
.SYSI	Initialize SOS devices
.OPEN	Open a file before writing or reading
.CLOSE	Close a file after writing or reading
.RESET	Close all open files
.GTATR	Get file status
.RDS	Read sequential characters
.RDL	Read sequential lines
.WRS	Write sequential characters
.WRL	Write sequential lines
.GCHAR	Read a TTI character
.PCHAR	Write a TTO character
.MEM	Determine available memory
.MEMI	Allocate a memory increment

If there is an error the system returns to the location following the SOS command with a system error code in AC2 (Section H of Appendix Q). Normally the SOS command performs its function and the system returns to the second location following the SOS command and continues. Further detailed explanation of the SOS commands can be found in References 12 and 13.

To program the cassette transport the procedure is as follows:

```
.EXTN .SOS, .CTU1  ;necessary for SOS commands
.
.
.
.SYSTM
.SYSI  ;initiates SOS devices
JMP error
.SYSTM
LDA 0,file ;AC0 contains the file number for Open
.SYSTM
.OPEN  31 ;Open CT1, device code 31
JMP error
LDA 0, buffer byte address ;byte address=2xaddress
LDA 1,buffer byte count ;number of characters
.SYSTM
.WRS  31 ;Record the output buffer
JMP error
LDA 0,buffer byte address ;SOS destroys all ACs
LDA 1,buffer byte count
.SYSTM
.RDS  31 ;Load recording into the input buffer
JMP error
.SYSTM
.CLOSE 31 ;close CT1
.
.
.
Type the buffer ;use TTO Example program
.
.
.
```

When the program is ready for the RLDR routine the operator must first load the Stand-Alone Operating System Library from paper tape Library program 099-000010-08 or from a cassette file it has previously been recorded on. Second, the operator must load the Stand-Alone System Cassette Driver from paper tape Library program 099-000041-02 or from a cassette file it has previously been recorded on. After these two programs are loaded in the order specified, the user program can be loaded and the external symbol references to the system labels .SOS and .CTU1 will be resolved. References 12 and 13 describe a separate trigger program, created by the user, to resolve these external references but it is not necessary. The external references will be resolved if the SOS I/O Driver utility program is loaded before the user program in the relocatable load. The user should not be alarmed at the page and one half length of the List of Entry Point Addresses, ncr the unidentified symbol errors that appear beside half of them. The .SOS and .CTU1 routines were written for a general system with all of the availabe options and peripherals. The undefined symbols are not used in our limited system.

1. <u>Exercise 6</u>

Using the TTY input and output routines you have written, create a program that takes a message typed in from the teletype. saves the message in an input buffer, records that buffer on a cassette file, loads the cassette file into an output buffer and outputs that buffer on teletype. The following message from Exercise 4 would be appropriate:

```
NAME,     RANK
STREET ADDRESS
CITY,    STATE
ZIP       CODE
```

Ensure that the edges are parallel and that the left margin is in column 9.

Section H of Appendix S is the Assembler listing for a system that will perform cassette communication. CASET uses the subroutines BIOA and TYPE that are in Sections B and C of Appendix S. TYPIO is a subroutine for entering characters from the teletype and packing them into a buffer area. It is included with the Assembler listing of CASET.

G. REVIEW OF PROGRAM CREATION

A brief summary of what has been covered in the creation process may help to tie it all together.

1.The user loads the CIL/W using the procedure discussed in Appendix A. Remember that if the console is still set up from a normal SOS user and if the CIL/W is still in core with the data switches set to 017777; initialization is achieved by pressing RESET and START.

2.Use the CIL/W to load the EDITOR and insert the Assembly language source code program. Remember to save the source on a scratch tape before closing the buffer.

3.Return to the CIL/W to load the ASSEMBLER. Replace the SOS master cassette on unit 0 with a new scratch tape (don't forget to REWIND) and execute the ASM command desired.

4.Move the ASSEMBLER relocatable binary tape file output from unit 0 to unit 1. Remount the SOS master cassette and press REWIND for both units.

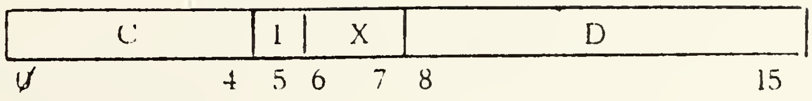
5.Return to the CIL/W to load the RELOCATABLE LOADER. Replace the SOS master cassette on unit 0 with another new scratch tape and execute the RLDR desired.

6.Return to the CIL/W and load the new absolute binary program that you just created on unit 0.

The SOS cassette system does not protect any files coming after the file being written into. The user must save these files on a separate scratch tape.

MRI FORMAT

Machine Code:



Assembly Code:

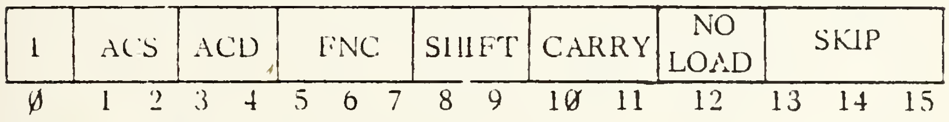
Label: OPCODE   AC, D, X   ; Comment

or

Label: OPCODE   D, X   ; Comment

ALC FORMAT

Machine Code:

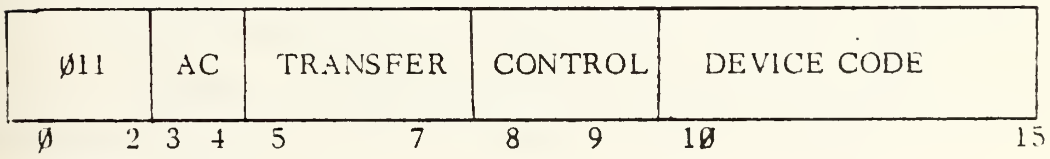


Assembly Code:

Label: FNC   ACS, ACD, SKIP   ; Comment

I/O FORMAT

Machine Code:



Assembly Code:

Label: Transfer AC, Device Code   ; Comment

Figure 4 - INSTRUCTION FORMATS

V. CONCLUSION

A. HARDWARE PROBLEMS

During the process of installing the A/D and D/A connections for sign bit extension, external potentiometers and MSB/LSB connectors a wiring problem has developed. The A/D does not operate correctly. The original thesis project resulted in a properly working model [Ref. 1]. The A/D has been factory checked and calibrated and is working properly. Although the wiring connections have been rechecked, no difference can be found from the pinning list of Reference 1. The manufacturer has offered to check the wiring diagram.

B. RECOMMENDATIONS

If A/D noise problems develope it is recommended that the flat cable connector from the patch board to the A/D inputs be changed to twisted pairs. This is a similar style of connecting cable that is commercially available.

It is recommended that the remaining D/A functions be connected so the full potential of the device can be used. Programming in Assembly language is a tedious and complicated process. Because the programmer must indicate the addressing modes and other details, errors are frequent in program creation. The next logical step in developing

the system is to use the Fortran, Basic or Algol programs that are already available. However this will require an expansion of the memory capacity to 24K words. This expansion would facilitate the interfacing of the NOVA® with the IBM 360 system.

It is unknown what affect the cut DMA lines are having on the interrupt capability, however this facility should be connected. This would allow an interrupt routine to sequence through the input channels on a timed basis and permit general feedback control applications. In this regard the present Real Time Clock is not very useful. The slowest clock frequency is 10Hz. This frequent interrupt rate requires some counting technique to permit, for example a one second sampling interval. That implies that at least ten interrupts must occur before the real job can be executed. Each interrupt requires a time delay to be serviced and the counting routine requires additional time to calculate the number of interrupts to have occurred since the last job was serviced. These delays can be estimated from the instruction execution times in Appendix N, and the interrupt count adjusted to allow for that processing time. However in a larger system when there are several other jobs to be serviced the delay time will vary due to the job load to an unacceptable degree for the accuracy required for proper feedback control. The effect of this random error could be reduced by applying the standard stochastic feedback control techniques, however more controlled laboratory situations would result if another presettable counter were installed in the second Real Time Clock RTC1 (Appendix O).

Finally, it is recommended that the TEKTRONIX display be connected at the second teletype device position TTO1 and TTI1. The present teletype is too slow and noisy to be really convenient. It could be kept for the paper tape and print functions.

As comprehensive as this thesis is, it can not be expected to provide the amount of detail that is to be found in the available documentation. The List of Manuals in Appendix U should provide any additional information required. Frequent references throughout the text guide the user to the correct publications to answer most questions.

APPENDIX A

NOVA® 800 SYSTEM INITIALIZATION

A. PRELIMINARY CONNECTIONS

Verify the following connections:

1.Connect CPU power cords to 115 VAC.

2.Connect ASR 33 teletype power cord to 115 VAC.
Connect TTY data cable to CPU rear I/O socket P2.

3.Connect cassette driver power cord to CPU rear outlet.
Connect cassette data cable to CPU rear I/O socket P5.

B. SWITCHES

4.Set cassette switch to REMOTE.
Set right-hand thumb wheel switch to 0.
Set left-hand thumb wheel switch to 1.

5.Set operator's console to ON.

6.Set TTY to LINE.

C. BOOTSTRAP

7.Mount SOS cassette on CT0 and press REWIND.

8.Set data switches to 100034 for cassette load.

9.Press PROGRAM LOAD. The teletype prompt # indicates correct initialization of the CORE IMAGE LOADER/WRITER.

10.Set the data switches to 017777.

APPENDIX B

CONSOLE OPERATIONS

A. TO SET PC AND CHECK THE CONTENTS OF A LOCATION

1.Set the data switches to the desired address.

2.Press EXAMINE. For AC use ACCUMULATOR EXAMINE.

B. TO ENTER OR MODIFY BINARY CODE

1.Set PC to the address.

2.Set the desired binary code in the data switches.

3.Press DEPOSIT. For AC use ACCUMULATOR DEPOSIT.

C. TO MANUALLY ENTER MACHINE CODE PROGRAMS

1.Set PC to the first program location.

2.Enter the binary contents for the PC address using the procedure described in B.

3.Set the data switches to the contents of the next program address.

4.Press DEPOSIT NEXT.

5.Repeat 3 and 4 until the entire program is entered.

D. TO VERIFY PROGRAM ENTRY

1.Set PC to the first program location.

2.Press EXAMINE.

3.Press EXAMINE NEXT.

4.Repeat 3 for each program address.

E. TO EXECUTE A PROGRAM

1.Enter the program using the procedure described in C.

2.Verify the program has been entered using the procedure described in D.

3.Set the data switches to the program start address.

4.Press RESET.

5.Press START.

APPENDIX C

OPERATING PROCEDURES

A. ASR 33 TELETYPE

The ASR 33 is an automatic Send and Receive terminal comprising a keyboard (TTI), printer (TTO), paper tape reader (TTR) and paper tape punch (TTP). It operates at a transmission rate of 10 characters per second (110 BAUD) and prints up to 72 characters per line at six lines to the inch. The model 33 has eight and one half inch width paper and will print only upper case ASCII code. Lower case codes are printed as upper case. Maintenance information is contained in Appendix T.

The teletype has separate input and output functions and therefore can be treated as two distinct devices. Each has its own device code, Busy, Done and Interrupt Disable flags, a separate buffer, and its own interrupt priority mask assignment (Appendix O). Striking a key places that character code in the A input buffer awaiting program retrieval. Input characters must be re-sent as output if the operator wishes the key that is struck to be printed (echo print). Model 33 printers ignore the even parity bit (MSB) in the 8-bit ASCII code listed in Appendix P.

There are three groups of switches on the terminal (Fig. 5). The right-hand switch has three positions for controlling all terminal functions as follows:

OFF-Power to the terminal is disabled.

LOCAL-Enables the terminal to operate independent of the computer.

LINE-Enables bi-directional communication with the CPU. This allows the teletype to be used as a separate typewriter, paper tape punch or paper tape listing device.

The left-hand set of four switches, which control paper tape operations, are selected by depressing the button for the desired function. When the button is pushed in the following operations are enabled:

ON-The punch will make paper tape for the operator if the control switch is at LOCAL or will list computer output on paper tape if the control switch is at LINE.

REL-New paper tape may be loaded in the punch.

B.SP-If an error occures when the operator is punching paper tape with the control switch in LOCAL, this switch moves the tape back one frame to allow deletion of the mistake by striking the RUB OUT key.

The paper tape reader is controlled by the left-hand three-position switch as follows:

STOP-The reader is disabled with the sprocket engaged.

START-The reader is enabled. In LINE the TTR responds to CPU commands. In LOCAL it will start to read a loaded paper tape.

FREE-The reader is disabled with the sprocket released so that paper tapes can be positioned for reading.
[Ref. 5]

Example

when attempting to read a paper tape the following procedure should be carried out:

-release the retaining clip that holds the paper tape on the read station.

-set the TTR switch to FREE.

-place the top end of the paper tape in the reader guides and the remainder of the tape on the floor below there, clear of obstacles.

-manually lead the paper tape through the guides towards the operator and stop with one blank frame before the data reaches the read station. Occassionally paper tapes have a hole pattern that can be read as their identification number, placed before the actual program data. Ensure that the tape is manually fed past that point. Ensure that the paper tape is mounted the correct side up, so that the sprocket holes are engaged.

-close the retaining clip, it snaps into place.

-set the TTR switch to START, the paper tape is now mounted and ready for reading.

B. CASSETTE TRANSPORT

The DGC cassette transport allows a more rapid and convenient means of program creation under the SOS. The character transfer rate is 1600 bytes per second at an average tape speed of 30 inches per second. Each 200 foot cassette requires 85 seconds for total rewind. The average storage capacity is 100,000 bytes or 800,000 bits. Each cassette is designated a logical unit number by positioning its thumbwheel switch. (Fig. -) Only one transport can be reading or writing at any time, and it must be moving in a forward direction. Each cassette can record files numbered 0 thru 99. The system can accomodate up to eight units (0-7) but it is presently configured in the SOS software for only 0 and 1. Cassette files are generally specified by CTunit:file, however for the CIL/W the CT is omitted (unit:file). The automatic BOOTSTRAP requires the SOS master tape to be on unit 0 for system initialization.

Power can be supplied independently to allow normal control by the ON/OFF switch. However if power is connected from the CPU rear outlet and the cassette switch is at REMOTE, the cassette unit will turn on a short time delay after the computer power is enabled.

CAUTION:

The possibility of noise spikes destroying tape data dictates the precaution of always mounting cassettes after power on and removing them prior to power off.

Once mounted, tapes must be positioned to the beginning of file 0 (BOT) by pressing REWIND. The user must be certain the tape is properly seated with the right-hand Cassette-in-Position and left-hand Write Enable switches engaged. These switches work with the small red tabs that are positioned over the holes in the upper edge of each cassette tape. The upper right tab is usually not moved from the position where the hole is uncovered.The actual Cassette-in-Position switch is not located over the hole so it has no effect. The upper left tab does control the write capability. User's must ensure the left tab is in the correct position whenever a protected tape is mounted on the transport. The SOS master cassette is a write protected tape. A thin piece of sticking tape ensures that a tape remains write protected. During the program creation procedure the operator is changing cassettes often, if it is remembered to REWIND the cassette before continuing most seating problems will be corrected before they cause a problem with reading or writing of files [Ref. 5]. Maintenance information is contained in Appendix T.

Programmers must be aware that there is no automatic protection of program files that occur after the file that is being modified. Increased length of the modified program causes the first locations in the next file (the start block etc.) to be overwritten. This destroys important control information so that subsequent files to the one modified cannot be accessed.

C. DATAX CONVERTERS

The A/D and D/A converters are extremely simple to use (Fig. 7). A connector board provides a convenient central location for user selection of input channels 0-7 and output signals X, Y, and Z. Each analog signal must be connected as an input and return pair designated by the appropriate labels. The Z output is a special timing signal for CRT applications. All signals must be adjusted for a ±10.00 volt swing and employ a 12-bit code that has the most significant bit extended for 16-bit input to the CPU. The following table is the basic 12-bit code:

Table 5 ANALOG CONVERSION CODE

VOLTS	12-BIT CODE
+ 9.9951	011111111111
0.0000	000000000000
-10.0000	100000000000

[Ref. 7]

The A/D has not been connected for Interrupt or Direct Memory Access (DMA) communication. The necessary circuits are included on the interface board but they have not been completed. In addition two DMA lines on the underside of the printed circuit board have been broken to allow the present installation to operate correctly. The 12-bit digital output of the A/D has been converted to 16-bits by connecting the MSB of the A/D code to the higher order bits of the computer data lines to achieve sign bit extension of the code.
Maintenance information is contained in Appendix T.

D. CALIBRATION

Three calibration routines; ADCOD, CADO, and DAC, are provided in Appendix S. They are designed for user convenience and involve minimal hardware connections. Appropriate instructions for the user are printed on the TTY. To eliminate the need for making connections on the printed circuit interface board and the requirement to place the interface on an extender board, the LSB, MSB and digital ground lines for the A/D are presented at labeled sockets on the side of the interface board. Both the A/D and the D/A are connected with the following labeled external potentiometers positioned so they are accessable at the side of the interface board; A/D RANGE, A/D OFFSET, D/A X and Y GAIN and D/A X and Y ZERO. The general user should not require to adjust the calibration. The laboratory supervisor can calibrate the converters by sliding the computer chassis forward on the rack until it hits the stops, then loading and executing one of the following routines.

Since the D/A converter cannot achieve the ±10.24 volt swing that the A/D can be set for, a compromise of ±10.00 volts was chosen to ensure compatability when they are used together. Program comment references to 10.24 volts should read 10.00 volts as indicated in the following sections.

ADCOD

The program A/D CODE TEST (ADCOD) is a procedure that converts an analog signal connected to channel 0 of the input board and prints the resulting 16-bit data word on the teletype. By presenting a known voltage source at channel 0 the operator can determine if the correct code is being produced. If the known source is +10.00 volts (or -10.00 volts) the RANGE potentiometer can be adjusted for the correct full scale reading (Table 5). The calibration is an iterative technique in two respects; the operator must depress the console switch CONTINUE to cause the A/D to reconvert the presented signal to see what effect the potentiometer is having and, the RANGE adjustment is not independent of the OFFSET adjustment. Once the RANGE is adjusted the channel 0 voltage is reset to 0 and the OFFSET potentiometer is adjusted. Remember to repeat this process until there is no more adjustment required.

Section A of Appendix S is a copy of the ADCOD Assembler listing. The subroutine BIOA is called to convert a binary word (the A/D code) into an ASCII character string for TTY output by the subroutine TYPE.
The Assembler listing for BIOA and TYPE are included as Sections B and C of Appendix S.

CADO

The program CALIBRATION of the A/D on the OSCILLOSCOPE (CADO) is a procedure for more accurate calibration of the A/D. A precisely measured, stable voltage source is connected to channel 0 on the input board. A teletype message prompts the operator to make the proper connection and voltage settings. By monitoring the MSB at -0.0025 volts, the oscilloscope must detect a 50 percent duty cycle because this is its transition point from negative to positive voltage codes. The duty cycle of MSB is adjusted by the OFFSET potentiometer. By monitoring the LSB at -9.9976 volts the oscilloscope must detect a 50 percent duty cycle due to this being the transition point for full scale negative readings. This duty cycle is adjusted with the RANGE potentiometer. As indicated above, the process must be repeated until the adjustments have stopped. Since the routine continuously tells the A/D to convert the present channel 0 voltage, an almost continuous reading is obtained and the CONTINUE switch on the console is not used. The detection of the 50 percent duty cycles on the oscillcscope has never been achieved satisfactorily. Section D of Appendix S is the Assembler listing for this program.

DAC

The program D/A CALIBRATION (DAC) is a procedure to convert program designated codes for 0.0000 and +10.0000 volts to the X and Y channel outputs, which can be monitored by a voltmeter. A teletype message reminds the operator of the correct procedure. Initially the routine places a zero voltage code into the X and Y holding registers of the D/A. This code is continuously converted to an output voltage until it is overwritten. Therefore the ZERO potentiometer can be adjusted for a minimum while continuously monitoring the voltage output that results. Pressing the console switch CONTINUE releases the program to start the GAIN calibration. The GAIN potentiometers are adjusted for a full scale reading of +10.0000 volts. Pressing CONTINUE again restarts the program at the ZERO adjust routine to allow an iterative technique in calibrating the D/A. Section E of Appendix S is the Assembler listing for this program.

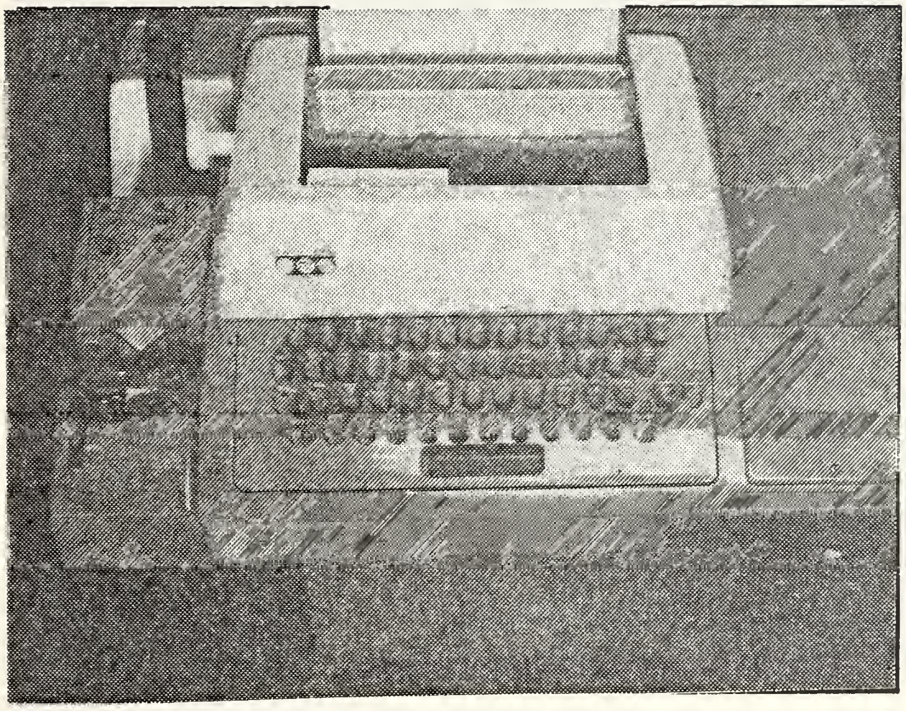


Figure 5 - ASR 33 TERMINAL

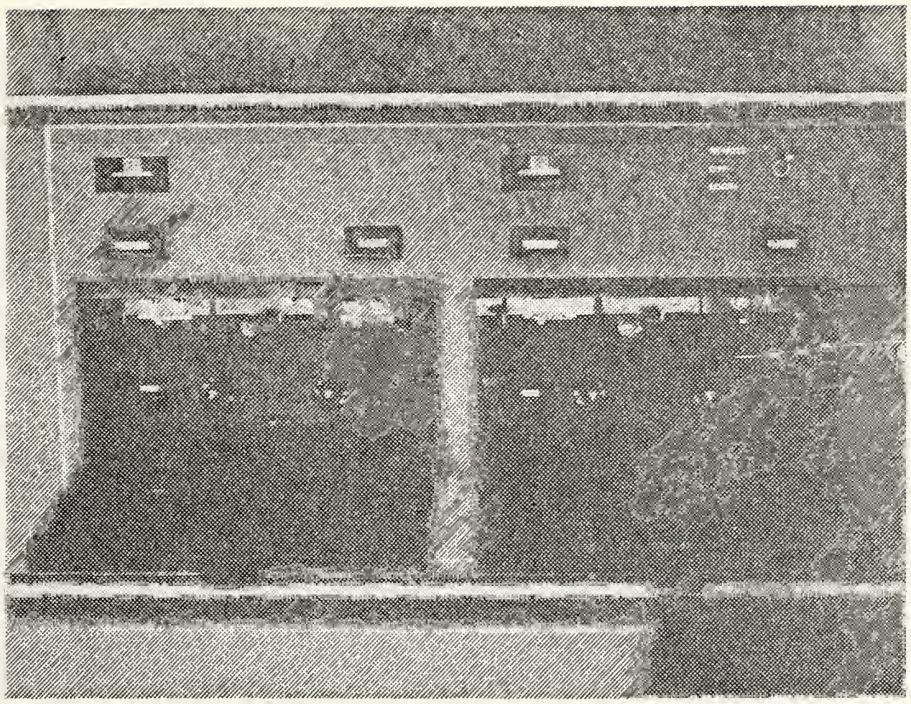


Figure 6 - DGC CASSETTE TRANSPORT

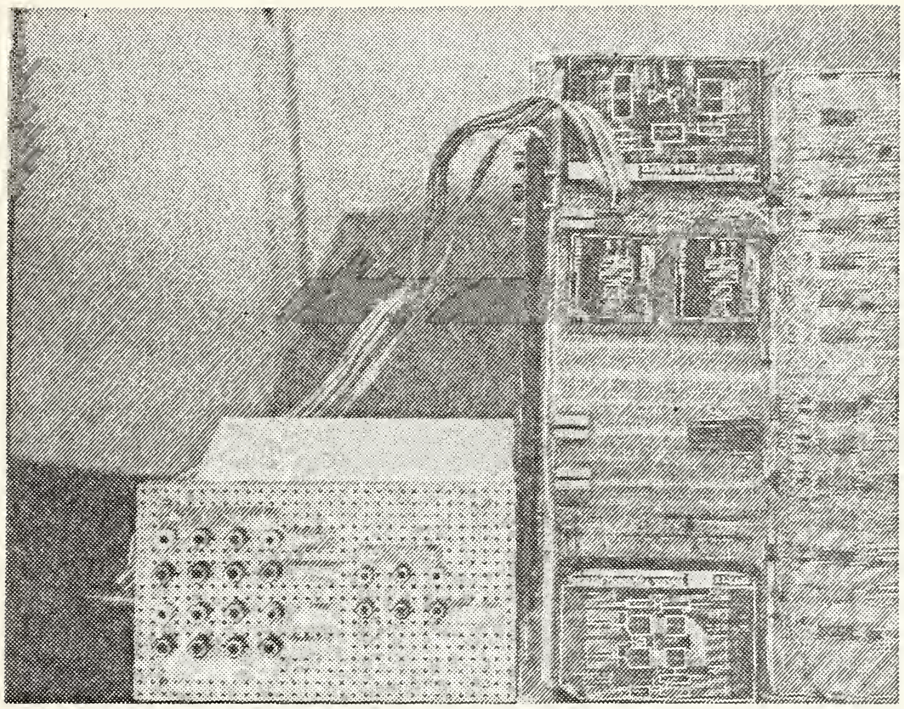


Figure 7 - A/D AND D/A INTERFACE

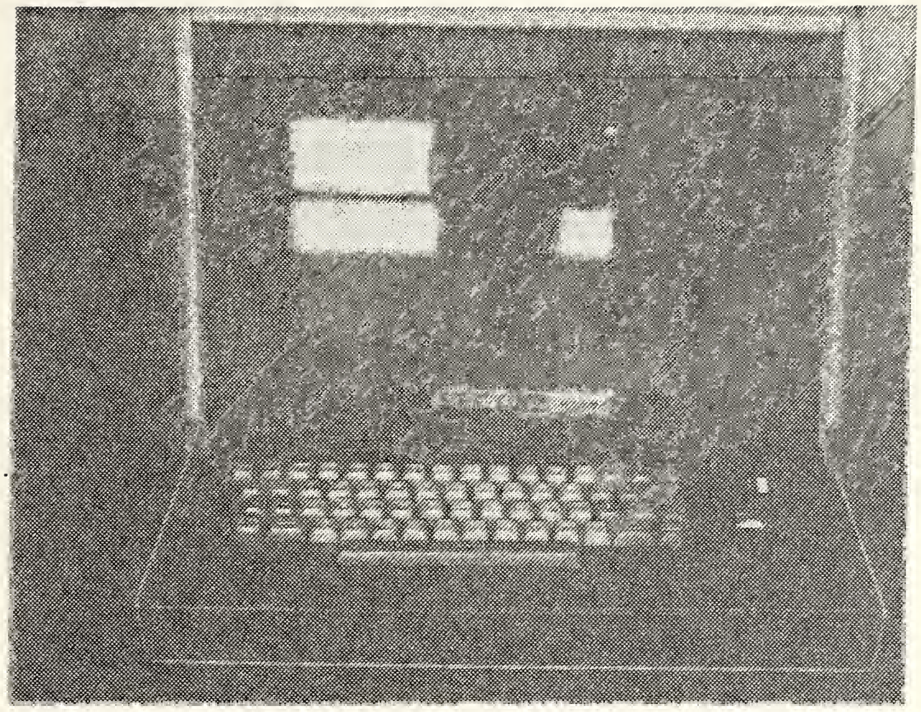


Figure 8 - TEKTRONIX® TEK 31/10 CATHODE RAY TUBE

APPENDIX D

LOADING PROGRAMS

Before a program can be executed, it must be brought into memory. This requires that a loading program already reside in memory. In the event that there is no loading program in memory, a small, specialized loading program is normally placed in memory and used to read in the loading program. This small loading program is called a BOOTSTRAP LOADER. The function of the bootstrap loader is to read in a more general-purpose loading program which can be used to load the user's programs. Two methods are available for entering a BOOTSTRAP LOADER into memory. The operator can either enter it via the data switches and the deposit switch or he can use the PROGRAM LOAD option. [Ref. 4]

A. MANUAL LOADING

Without the PROGRAM LOAD option, a BOOTSTRAP LOADER must be entered into memory manually using the switches on the console. The following loader is the BOOTSTRAP LOADER designed for use with BINARY LOADER 091-000004. It reads in a specially formatted tape from either the paper tape reader (PTR) or the teletype reader (TTR).

```
LOCATION  CONTENTS  TAG   ASSEMBLER         COMMENTS

0XX757    126440    GET:  SUBO       1,1    ;Clear AC1 and
                                            ; carry
0XX760    0636dd          SKPDN      dd     ;Device busy?
0XX761    000777          JMP        .-1    ;Yes
0XX762    0605dd          DIAS       0,dd   ;Read frame
                                            ; from device
0XX763    127100          ADDL       1,1    ;Shift AC1 left
                                            ; 2 bits
0XX764    127100          ADDL       1,1    ;Shift AC1 left
                                            ; 2 bits
0XX765    107003          ADD        0,1,SNC ;Add in new
                                            ; frame
0XX766    000772          JMP        GET+1 ;Get new frame
0XX767    001400          JMP        0,3    ;Full word, return
0XX770    0601dd  BSTRP:  NIOS dd    ;Prime the device
0XX771    004766          JSR        GET    ;Get a word
0XX772    044402          STA        .+2    ;Store it
0XX773    004764          JSR        GET    ;Get another word
0XX774   (see comment)   (STA)      (1,.+1;These instructions
0XX775   (see comment)   (JMP)      (.-4)  ;are loaded by the
0XX776   (see comment)   (HALT)            ;binary loader
                                            ;stop address
```

The BOOTSTRAP should be placed in memory starting at the location which is 20 (octal) less than the highest available memory address (for 8K start at 17757). For the XX in the Location column, substitute the most significant 2 digits of the highest available memory address as described in the following table:

Table 6 HIGHEST MEMORY ADDRESSES

MEMORY	ADDRESS	XX
2,000	003777	03
4,000	007777	07
6,000	013777	13
8,000	017777	17
10,000	023777	23
.	.	.
.	.	.
.	.	.

For dd in the Contents column, substitute 10 (octal) if the TTR is being used, or 12 if the PTR is being used. After the BOOTSTRAP is entered, start it at location XX770 (17770 for 8K). Execution terminates when the BINARY LOADER is completly loaded, at address XX776, with the data lights reading 063077.

B. AUTOMATIC PROGRAM LOAD

The automatic program load is designed for use with the SELFLOADING BOOTSTRAP AND BINARY LOADER paper tape 091-000036 or the STAND-ALONE OPERATING SYSTEM CASSETTE LOADER/WRITER 091-000067, which should be on File 0 of cassette unit 0 [Ref. 12]. The BOOTSTRAP reads the data switches, sets up its own I/O instructions with the specified device code in switches 10-15, and then continues in accordance with the value of data switch 0. [Ref. 3]

If switch 0 is 0, the BOOTSTRAP reads low-speed input like the TTR. If the device is not low-speed the program halts. The device must supply 8-bit data bytes, and each pair of bytes is stored as a single word in memory wherein the first and second bytes read become the left and right halves of the word. The program ignores tape leader and does not begin storing any words until it reads a nonzero synchronization byte. The first word following that byte must be the negative of the total number of words to be read (including the first word), for a maximum of 192 (decimal) words. The program stores the words beginning at location 100. After reading all the data, it jumps to the last word stored.

If the switch is 1, the BOOTSTRAP starts the high-speed device (such as the cassette drivers) for data channel storage beginning at absolute location 0, and then loops at location 377 until a loaded data word causes it to do something else. Addressing a low-speed device stops the program before input occures.

C. BINARY LOADER PROGRAMS

The BINARY LOADER program loads absolute object tapes into memory and resides in absolute locations 0XX646-0XX777 in core. It is common practice to write programs which do not alter these locations, thus eliminating the need to reload the loaders. In all but very rare instances, DGC standard software is written so as not to destroy the BINARY LOADER or BOOTSTRAP LOADER programs. In no case will any of this software destroy the BOOTSTRAP LOADER program.

If the End Block on the object tape specifies a starting address of the program, the BINARY LOADER will transfer control to that location once tape is loaded. Otherwise, load the starting address of the program into the data switches, press RESET then START.

There are two BINARY LOADER programs available, the manual BINARY LOADER and the SELFLOADING BOOTSTRAP AND BINARY LOADER. [Ref. 2]

1. Manual Bootstrap Binary Loader

The paper tape 091-000004 is the BINARY LOADER program to be used with the manual BOOTSTRAP. The input to the Loader is an absolute binary tape. The tape is punched in blocks separated by null (all zero) characters. Two tape characters form a 16-bit word; the first character forms bits 8-15 of the data word and the second tape character forms bits 0-7. [Ref. 10]

The BINARY LOADER routine is executed by mounting the desired absolute binary paper tape program on the TTR, entering SXX777 in the data switches and pressing RESET and START. The S represents data switch 0 and should be 1 if input is PTR and 0 for TTR. The XX represents the most significant 2 digits of the highest available memory address. The result of executing the BINARY LOADER routine is a loaded program ready for execution. A HALT may be interpreted by its location displayed in the address lights, as follows:

0XX741 means loaded program did not specify a start
 address. The user must set the data switches and
 press RESET then START.

0XX727 has two possible causes.
 1.The user's program attempted to overwrite
 the loader, or
 2.The last block read has a checksum error.
 Tapes produced under SOS must be reread from
 the first block. Repeated checksum errors
 indicate a bad tape.

2. The Selfloading Bootstrap and Binary Loader

The SELFLOADING BOOTSTRAP AND BINARY LOADER paper tape 091-000036 is used in conjunction with the PROGRAM LOAD feature. Once the BOOTSTRAP is complete it sizes memory, interprets the device code, and reads in the BINARY LOADER. Determination of the highest location is accomplished by writing and reading locations at 1K increments until the information read back is the same as that written. The BINARY LOADER image is placed in the highest locations of alterable memory. When the tape has been read in, the processor will HALT at location 00121. An object tape can then be read on the same device simply by depressing CONTINUE. For subsequent object program loads the proceedure is:

1.Put the object tape in the reader.
2.Set the data switches to 0XX777.
3.Set data switch 0. PTR=1, TTR=0.
4.Press START.

This BINARY LOADER is similar to the manual version except for the HALT addresses.

0XX740	means no start address.
0XX726	means checksum failure. If repositioning the tape to the beginning of the last block read and continuing has no effect, then the tape is in error.

3. The Core Image Loader/Writer

The CORE IMAGE LOADER/WRITER program on the SOS master cassette is identical to paper tape 091-000067-02. It performs two utility functions: it loads core image files from cassette tape into core and produces core image files on cassette tape [Refs. 12 and 13]. The CORE IMAGE LOADER/WRITER program works only with cassettes.

The CORE IMAGE LOADER/WRITER can be bootstrapped from file 0 of the SOS master cassette on unit 0. The tape must be rewound manually. The normal loading procedure is described in Appendix A.

The Loader/Writer is read into page zero (0-377) initially and then relocates itself to the last 400 (octal) locations in core. After relocation a prompt # on the teletype indicates that the CORE IMAGE LOADER/WRITER is ready. Once it is in core the Loader may be restarted by setting the data switches to the last memory address, pressing RESET, and then START. (For 8K set 017777)

The # symbol indicates the loader is waiting for the operator to respond with a cassette unit number (0-7) and a file number (0-99) separated by a colon. Specifying unit 0 is optional. The indicated cassette file is loaded into memory upon command termination by a teletype RETURN. If data switch 0 on the console is 1, the program will halt on completion of the load. If the switch is 0, control is passed to the loaded program linked through location 405.

If the loader encounters a non-recoverable error while trying to load a file, it will type *ERR and halt with a code in AC0. The error codes are explained in Section A of Appendix Q. If rewinding and substituting a different cassette tape does not clear the error condition, a hardware fault is indicated.

APPENDIX E

SOS

When using device mnemonics within the SOS environment, the user must add the prefix $. The teletype codes are $TTO, $TTI, $TTR, and $TTP.

Since the EDIT commands delineator (ESC) prints as a $, considerable care must be taken to ensure that the ESC or $ keys are used properly.

The SOS master cassette has a standard file format:

Table 7 SOS MASTER TAPE

FILE	PROMPT	PROGRAM (CALL)
0	#	Core Image Loader/Writer
1	R	Command Line Interpreter (CLI)
2	*	Symbolic Text Editor (EDIT)
3	ASM	Extended Assembler (ASM)
4	RLDR	Extended Relocatable Loader (RLDR)
5	LFE	Library File Editor (LFE)
6	SYSG	SYSGEN (SYSG)

By setting data switch 0 to 0 prior to loading a SOS utility, the user can permit the automatic typing of the appropriate prompt message to signal correct initialization. Setting data switch 0 to 1 forces the Loader to halt before control is passed to the loaded routine.

There are two possible ways of interrupting and terminating a currently executing utility program from the teletype.

1.Pressing CTRL and A on the keyboard causes all utilities to stop, initialize, and re-issue the prompt message. The EDITOR will only respond to CTRL A during a T, Y, N, E or P edit command and the input buffer will remain intact. This is the only release from a GR or a GW command error.

2.Pressing CTRL and C will cause all utilities except the TEXT EDITOR to return to the CORE IMAGE LOADER. The EDITOR ignores CTRL C and uses the edit command H to return to the Loader. [Ref. 13]

The two SOS master cassette utility programs LIBRARY FILE EDITOR and SYSGEN are not often used. A brief description is included here for completeness.

A. LIBRARY FILE EDITOR

The LIBRARY FILE EDITOR (LFE) provides a means of updating and interpreting a set of relocatable binary files that are gathered together into one special file called a Library.

The LFE allows the user to:

-analyse the contents of a library file

-list titles in a library file

-merge libraries

-update libraries

-extract logical records from a library file

-create his own library files.

The LFE is self-starting and prompts the operator with LFE when it is ready to accept a command string. Commands are explained in Appendix J. [Ref. 16]

B. SYSGEN

SYSGEN generates special programs called triggers which may be used to link user programs to SOS utility programs via their entry symbols. For each entry symbol included in the command line, an external normal reference to the program is included in the trigger. The trigger is entirely made up of these external normal references. When the utility is ready to accept a command line, the prompt SYSG is typed. Ccmmands are explained in Appendix K.
[Ref. 13]

APPENDIX F

CLI COMMANDS

CLI functions are executed by pressing RETURN after the command.

ASM
This command causes file 3 on CT0 to be loaded. If the master cassette is mounted, the EXTENDED ASSEMBLER overwrites the CLI.

RLDR $TTR
This command will load an absolute binary tape with the CLI binary block loader. The input device can be either $TTR or $PTR. Both the CORE IMAGE LOADER/WRITER and the CLI are overwritten.

CTx:yy Core image file yy on cassette unit x overwrites the CLI. Incorrect unit or file numbers cause the error message FILE NON-EXISTENT on the TTO.

EDIT
File 2 on CT0 is loaded. The CLI is overwritten by the SYMBOLIC TEXT EDITOR.

INIT CTx or RELEASE CTx
The specified cassette unit is rewound. Incorrect unit x causes the error message ILLEGAL FILE NAME.

LFE
File 5 on CT0 overwrites the CLI. If CT0 is the master cassette, the LIBRARY FILE EDITOR is loaded.

MKSAVE infile outfile
The input file (AB) is converted to a core image output file. Possible error messages are:

```
NOT ENOUGH ARGUMENTS
ILLEGAL FILE NAME
ILLEGAL COMMAND FOR DEVICE
DEVICE IS READ PROTECTED
FILE NON-EXISTENT
CHECKSUM ERROR
PHASE ERROR
```

RLDR
File 4 on CT0 overwrites the CLI. If CT0 is the master cassette, the EXTENDED RELOCATABLE LOADER is loaded.

SYSG
File 6 on CT0 overwrites the CLI. This is the SYSGEN routine on the master tape.

XFER source destination
This command transfers the source file to the destination file. Appending /A means the source is even parity ASCII. [Ref. 12]

APPENDIX G

EDIT COMMANDS

For a discussion on how to use the TEXT EDITOR refer to Section B3 of Chapter III.

ESC

Striking the escape (ESC) key on the TTY causes a $ to be printed. The escape key ($) is used once to delimit edit commands. If the command has no argument the $ is optional. Two successive codes ($$) execute the command string.

RUBOUT

This key deletes the last typed character. Repeated rubouts delete successive characters in that line from right to left. The character being deleted is echoed on the TTY.

TAB

The EDITOR has predefined tab positions at columns 1, 9, 17, 25,... which are used with CTRL I. The tabs may be turned off by CTRL P and back on by repeating it.

GR input

Before beginning modifications to an existing routine it must be brought into the edit buffer. This command enables the input file specified for reading. A cassette file is specified by; CTunit:file. The same cassette unit cannot be simultaneously write enabled (GW). No actual read occures. To clear a GR buffer lock-up use CTRL A.

GW output

Immediately after read enabling, the write file should be assigned. A cassette file is specified by; CTunit:file. The same cassette unit cannot be simultaneously read enabled (GR). No actual write occures. To clear a GW buffer lock-up use CTRL A.

GC

All output files must be closed with this command. No actual write occures. Multiple files may be appended by successive GR commands before a GC.

H

The EDIT is terminated and control returns to the CORE IMAGE LOADER/WRITER.

Y

The first page of symbolic text is read into the edit buffer. A page is a character string terminated by a form feed. An input device must have been previously enabled. The character pointer (CP) is positioned at the start of the buffer.

A

This command appends a page of input to the present contents of the edit buffer. CP points to the first character appended.

nT

The number of lines n is typed. Omitting n causes the entire buffer to print.

B

CP is moved to the beginning of the buffer.

nJ

CP is placed at the beginning of line n.

L

CP advances to the beginning of the n'th line from the present position. Any value of n is accepted, however too large a value acts like B or Z commands. Omitting n moves CP to the beginning of the present line.

nM

CP moves by the character count n.

Z

CP is positioned at the end of the buffer.

Coldnew
This command searches from the present position to the end of the buffer and replaces the first character group 'old' with 'new'. CP points to the first character after 'new'. If unsuccessful STR NOT FOUND is typed and CP points to the beginning of the buffer. Omitting 'new' deletes 'old'.

Iinput$
This is the command for creating a program. Existing programs insert 'input' before the position CP and adjust the CP count to point to the end of 'input'.

nI
The octal number n is masked to 7-bits and inserted at CP.

nD
This command deletes n characters relative to CP.

nK
This command deletes n lines from the CP position. CP movement is like the nL command but all characters passed over are erased.

Sstring$
This command searches foreward from the CP for the character group 'string'. CP moves to the last character of the first group found. Unsuccessful search leaves CP at the beginning of the buffer.

Nstring$

The EDITOR executes P and Y commands until the string is found or the input file completed.

Qstring$

This is a search like the Nstring$ command without the P.

XMcode$code$...$$

One macro-command can be defined as the specified command string.

nX

The previously defined XM is executed n times.

XD

The macro XM is deleted.

nF

This command outputs n inches of leader. Greater than 100 inches is ignored. Omitting n causes a form feed.

nP

This command outputs n lines from CP with a form feed. Too small a buffer causes a halt at the buffer's end. Omitting n outputs all the contents after CP.

nPW

This is the same as the nP command but without the form feed.

E

This command outputs the edit buffer and the remainder of the input file.

nR

This command outputs a page and inputs a page, repeated n times.

:

This command prints the number of lines in the edit buffer.

.

This command prints the line number of CP.

=

This command prints the number of characters in the edit buffer.

CTRL A

This command re-initializes the EDITOR with the buffer unchanged. This control is only acknowledged during T, Y, N, E, or P.

CTRL C

This command cancels the present line. If a command string is executing it will halt. CP repositions to the beginning of the buffer.

CTRL I

This command inserts tabulation.

CTRL T

This command resets for a new tape. The input device stops and the buffer is cleared.
[Refs. 14 and 15]

APPENDIX H

ASM COMMANDS

The Assembler takes two passes to translate an ASCII source file to a relocatable binary program. The method of translation and the files involved are designated by the user typing a command line after the ASM prompt. The general command format is:

```
 ASM M/m File/u File/u
```

Where M is the mandatory Assembly mode, which must be first, and /m is the optional mode modifier. An unlimited number of participating Files are then listed with their optional use designators /u. Omitting the space between fields causes errors that may not be detected.

ASM functions are executed by pressing RETURN after the command. An assembly can be carried out on an ASCII source file in any one of the three following modes:

0-Perform pass one on the specified input source file(s). Halt with the highest symbol table address in AC0.

1-Perform passes one and two on the specified input files, producing binary and listing files as specified. At the completion of pass two, the assembler prompts with ASM.

2-Perform pass two only on the specified input files, producing the specified binary and listing files. The symbol table used is that produced by the most recent pass one assembly. The prompt ASM signifies completion.

Any Assembler mode can be modified by appending the following optional codes:

Table 8 ASSEMBLER MODE DESIGNATORS

DESIGNATOR	MEANING
/E	Suppress assembly error messages to the TTO.
/T	Suppress the symbol table listing
/U	Include local (user) symbols in the binary output file.

After the basic assembly mode has been indicated, the files are listed with optional appended codes that indicate specific uses as follows:

Table 9 ASSEMBLER FILE DESIGNATORS

DESIGNATOR	MEANING
/B	Relocatable binary file to be output on this device.
/L	Output device for the listing.
/N	Any input file not to be listed on pass two.
/P	Pause before accepting this file. The message PAUSE - NEXT FILE, devicename is output. The assembly continues when any key is struck on the teletype.
/S	Skip this file during pass two.
/n	Repeat this file n times. (n from 2 thru 9)

A typical command to Assemble file 4 of CT1 and a paper tape to file 6 on CT0 with a teletype listing would look like:

```
 ASM 1 CT1:4 $TTR CT0:6/B $TTO/L
```

[Ref. 12]

APPENDIX I

RLDR COMMANDS

The RELOCATABLE LOADER translates the relocatable addressing of the Assembler's RB output into absolute locations in memory and resolves the displacements among any routines that have been combined at load time. A successful load is indicated by the message OK. The command line is typed by the operator after the prompt RLDR. The general format is:

```
 RLDR File/S File/u
```

Where a /S is mandatory and any number of additional participating files are listed with their optional use designators /u.

The RLDR automatically spaces before the first entry and must not have a space inserted there by the operator. Omitting the space between fields causes errors that may not be detected.

RLDR functions are executed by pressing RETURN after the command string. Files are listed with optional appended codes that indicate specific uses as follows:

Table 10 LOADER FILE DESIGNATORS

DESIGNATOR	MEANING
/L	Causes a listing of the symbol table on the output file or device whose name precedes the use code. Symbols in the table are ordered numerically by symbol value.
/L/A	Changes the /L to an alphabetical listing.
/N	Set the starting load address (NMAX) for the file that follows, to this absolute address.
/P	Pause befcre opening this file.
/S	This is the mandatory save file.
/U	Load user symbols appearing within this file.
/n	Load this file n times (n from 2 thru 9).

[Ref. 12]

A typical command to load files 4 and 5 of CT0 and a paper tape into file 3 on CT1 with a teletype listing would look like:

```
 RLDR CT0:4 CT0:5 $TTR CT1:3/S $TTO/L
```

APPENDIX J

LFE COMMANDS

The LFE is a specialized utility program for maintaining Library files. Since the expected use of this program is small, only a brief overview of the complex command structure is given here. Further details may be found in Reference 16.

The command string is typed by the operator after the prompt LFE. The general format is:

```
 LFE Key File/u File/u
```

Where Key is a letter indicating the function desired. The participating Files are then listed with their optional use designators /u. A File may be a Binary, which is an RB file not in a Library, or a Logical Record, which is an RB file in a Library. A Binary being placed in a Logical Record of a Library is called an Update.

<u>The LFE automatically spaces before the first entry and must not have a space inserted there by the operator. Omitting the space between fields causes errors that may not be detected.</u>

LFE functions are executed by pressing RETURN after the command. The following Key letters are available:

Table 11 LFE KEY DESIGNATORS

<u>DESIGNATOR</u> <u>MEANING</u>

A-Itemize the global declarations of the file. A global declaration is an Assembler language pseudo-operation explained in Section C of Chapter IV.
D-Delete Logical Record.
I-Insert Binary into a Library (Update).
M-Combine Libraries and Binaries in a new Library.
R-Replace Logical Records with new Binaries.
T-List titles in a set of Libraries or Binaries.
X-Extract specific Logical Records from a Library.

After the basic LFE operation has been indicated, the files are listed with optional appended codes that indicate specific uses as follows:

Table 12 LFE FILE DESIGNATORS

<u>DESIGNATOR</u> <u>MEANING</u>

/A-Make insertions after this Logical Record.
/B-Make insertions before this Logical Record.
 -Or, this is a Binary file.
/I-This is the input file.
/O-This is the output file.
 -Or, this is the new Library name.
/R-Itemize the global declarations in this file.
/#-For # substitute the number of $TTR files to read.

An entire command string can be deleted by typing SHIFT L. Single characters are deleted with RUBOUT, and a back arrow echoes each erasure. Multiple erasures move from right to left deleting characters on the same line.

If an error condition is detected, a message will be output. Improper command strings result in no output. An execution error attempts to identify the file responsible and closes all Library file outputs. Section F of Appendix Q summarizes the error messages.

The following operator prompt messages are possible:

LOAD device, STRIKE ANY KEY.
This message may be preceded by INPUT or UPDATE to help identify which device is waiting.

REMOVE INPUT MASTER AND LOAD U.F
This message prompts the operator when an Update file is to be read in the same device that inputs the Library file.

REMOVE U.F AND LOAD BACK INPUT MASTER
After the Update file has been read in, the Library file must again be read.
[Ref. 16]

APPENDIX K

SYSG COMMANDS

SYSGEN generates triggers for use in configuring SOS utility programs. A trigger is a program that resolves external references to entry symbols in SOS Libraries. The prompt SYSG is followed by a list of entry symbols for each desired utility, a file designated for output and an optional trigger name. The general format is:

```
 SYSG driver driver outputfile/O trigger/T
```

Where the driver is a desired entry symbol, /O specifies the output file and /T specifies the trigger name. Omitting the trigger name results in the default title SGTRG. The command string is executed by pressing RETURN. Section G of Appendix Q summarizes the error messages.
[Refs. 12 and 13]

APPENDIX L

MACHINE CODE AND ASSEMBLER LANGUAGE FIELDS

Bit positions in all 16-bit words are numbered 0-15 from left to right.

The MRI instruction word is divided into four fields:

-The command field C (bits 0-4) designates the type of instruction (OPCODE) and sometimes the accumulator (A/C) involved.

-The addressing mode field I (bit 5) designates indirect addressing. If I is 1 the effective address points to a new effective address.

-The index field X (bits 6 and 7) indicates the addressing mode of the instruction.

-The displacement field D (bits 8-15) contain an integer that may be used to obtain the effective address.

The information to insert in all these fields must be communicated to the ASSEMBLER program.

The Move Data MRI asembly language format is:

```
LABEL: OPCODE AC,D,X ;COMMENT
```

The Modify Memory and Jump MRI format does not have the AC field. Move Data OPCODES are LDA and STA. Modify Memory OPCODES are ISZ and DSZ. Jump OPCODES are JMP and JSR.

The I field is designated by using the symbol @ anywhere in the assembly language instruction. It is suggested that prefixing the displacement (@D) would be a logical choice.

The effective address E is formed by the X and D fields. It is the location that is to be referenced. Using Table 13 the effective address may be calculated by the following equation:

$E = (X) + D$

Where (X) means the contents of X.

Table 13 EFFECTIVE ADDRESS DETERMINATION

X	(X)	EFFECTIVE ADDRESS
00	0	Page zero addressing $0 \leq E \leq 377$ (octal)

In the following modes if bit 8 is 0, D is positive; if bit 8 is 1 D is two's complement.

X	(X)	EFFECTIVE ADDRESS
01	(PC)	Relative addressing $(. - D) \leq E \leq (. + D)$
10	(AC2)	Base register addressing $(AC-D) \leq E \leq (AC + D)$
11	(AC3)	Base register addressing $(AC - D) \leq E \leq (AC + D)$

When programming in assembly language the X value will determine how the ASSEMBLER program will handle D. If X is 0 or blank and $D \leq 377$, the mode X is set to 00 and D is unchanged. If $D > 377$ the present location L is checked to see if it is within 200 locations of D. If $L-200 \leq D \leq L+177$ the mode X is set to 01 and D is replaced by L - D. Any other X value forces the mode indicated. However, if $-200 \leq D \leq 177$ an address error is flagged by the symbol A.

The ALC instruction word is divided into eight fields:

-Bit 0 is always at 1.

-The source accumulator field (ACS) designates where the data to be operated on is taken from. (bits 1 and 2) It is usually left unchanged.

-The destination accumulator field (ACD) designates where the result of the operation is to be stored. (bits 3 and 4) Occassionally the original ACD data is used to calculate the result.

-The function field (FNC) designates the command. (bits 5-7)

-The carry field (CARRY) designates the value of carry in the function generator _prior_ to performing the operation. (bits 10 and 11) This base value is affected by the function results. Toc large a result to store in 16-bits results in the base carry value being complemented due to overflow.

-The shift field (SHIFT) designates whether the result of a function is rotated left or right before loading into ACD. (bits 8 and 9)

-The skip field (SKIP) designates a test condition for the shifted result, to determine if the next sequential location is to be skipped. (bits 13-15)

-The no-load field (NO LOAD) designates if the shifted result that has been tested for any skip conditions, will in fact be loaded into ACD.

The ALC assembly language format provides the information for all of the instruction word fields as follows:

```
LABEL: FNC ACS,ACD,SKIP ;COMMENT
```

The basic ALC function codes are COM, NEG, AND, INC, ADD, SUB, ADC, and MOV. These codes may be modified by appending a letter for the carry bit as follows:

Table 14 CARRY DESIGNATORS

DESIGNATOR	MEANING
blank	carry based on current carry state
Z	set carry base to 0
O	set carry to 1
C	carry based on current carry state complemented

The function code can be further modified by appending the following shift letters:

Table 15 SHIFT DESIGNATORS

DESIGNATOR	MEANING
blank	no shift
L	rotate left 1 bit, CRY to bit 15, bit 0 to CRY
R	rotate right 1 bit, CRY to bit 0, bit 15 to CRY
S	exchange bits 0-7 with bits 8-15, CRY unchanged

The last modification to the function code could be to append a # symbol which indicates that the result is not to be loaded into ACD.
The SKIP mnemonics are as follows:

Table 16 SKIP DESIGNATORS

DESIGNATOR	MEANING
blank	never skip
SKP	always skip
SZC	skip on zero carry
SNC	skip on zero result (bits 0-15)
SNR	skip on non-zero result (bits 0-15)
SEZ	skip on zero (result + carry)
SBN	skip on non-zero (result + carry)

The I/O instruction word is divided into five fields:

-Bits 0-2 are always 011.

-The accumulator field (AC) designates where in the processor the data is to be output from or input to. (bits 3 and 4)

-The transfer field (TRANSFER) designates which of up to three possible device buffers (A, B, C) will be used and whether this is input to the computer or output from the computer. (bits 5-7)

-The control field (CONTROL) designates device control instructions that manipulate the Busy and Done flip-flops of the specified peripheral. A set Busy flip-flop indicates the device has been assigned an I/O task. When a device has completed its task and is ready to process a new request, it clears the Busy and sets the Done flip-flops. If both flip-flops are 0 the device is idle. (bits 8 and 9)

-The device code field (DEVICE CODE) specifies the peripheral involved in the I/O function. DEVICE CODE 00 is not used and 77 denotes special CPU functions. (bits 10-15)

The I/O assembly language format provides the information for all of the instruction word fields as follows:

LABEL: TRANSFER AC,DEVICE CODE

The basic I/O transfer codes are NIO, DIA, DOA, DIB, DOB, DIC, and DOC. However, when the CPU commands are desired, several special mnemonics will generate their I/O equivalent as follows:

Table 17 SPECIAL CPU MNEMONICS

MNEMONIC	EQUIVALENT	MEANING
READS	DIA -,CPU	Read data switches
IORST	DICC 0,CPU	I/O reset
HALT	DOC 0,CPU	Stop processing
INTEN	NIOS CPU	Interrupt enable
INTDS	NIOC CPU	Interrupt disable
INTA	DIB -,CPU	Interrupt acknowledge
MSKO	DOB -,CPU	Mask the interrupt disables

The basic TRANSFER code can be modified by appending a letter for the CONTROL field as follows:

Table 18 CONTROL DESIGNATORS

DESIGNATOR	MEANING
blank	no control
C	clear Busy and Done, idles device
S	set Busy, clear Done, starts device
P	special device pulse, flip-flops anaffected

One other set of TRANSFER codes qualifies as a special I/O instruction since in assembly language the programmer specifies only an OPCODE and a DEVICE CODE.

Table 19 I/O SKIP INSTRUCTIONS

DESIGNATOR	MEANING
SKPBN	skip if Busy is one
SKPBZ	skip if Busy is zero
SKPDN	skip if Done is one
SKPDZ	skip if Done is zero

APPENDIX M

ASSEMBLY LANGUAGE INSTRUCTIONS

ASSEMBLY MNEMONIC	MACHINE CODE	COMMENTS
ADC	102000	Add the complement of ACS to ACD; use Carry as base for carry bit.
ADCC	102060	ADC but complement carry is base
ADCCL	102160	ADCC with rotate left
ADCCR	102260	ADCC with rotate right
ADCCS	102360	ADCC with swap halves of result
ADCL	102100	ADC with rotate left
ADCO	102040	ADC but 1 is base for carry bit
ADCOL	102140	ADCO with rotate left
ADCOR	102240	ADCO with rotate right
ADCOS	102340	ADCO with swap halves of result
ADCR	102200	ADC with rotate right
ADCS	102300	ADC with swap halves of result
ADCZ	102020	ADC but 0 is base for carry bit
ADCZL	102120	ADCZ with rotate left
ADCZR	102220	ADCZ with rotate right
ADCZS	102320	ADCZ with swap halves of result
ADD	103000	Add ACS to ACD; carry bit based on CRY
ADDC	103060	ADD but complement carry is base
ADDCL	103160	ADDC with rotate left

ASSEMBLY MNEMONIC	MACHINE CODE	COMMENTS
ADDCR	103260	ADDC with rotate right
ADDCS	103360	ADDC with swap halves of result
ADDL	103100	ADD with rotate left
ADDO	103040	ADD but 1 is base for carry bit
ADDOL	103140	ADDO with rotate left
ADDOR	103240	ADDO with rotate right
ADDOS	103340	ADDO with swap halves of result
ADDR	103200	ADD with rotate right
ADDS	103300	ADD with swap halves of result
ADDZ	103020	ADD but 0 is base for carry bit
ADDZL	103120	ADDZ with rotate left
ADDZR	103220	ADDZ with rotate right
ADDZS	103320	ADDZ with swap halves of result
AND	103400	Logically And ACS with ACD; CRY is carry bit
ANDC	103460	AND but complement Carry is carry bit
ANDCL	103560	ANDC with rotate left
ANDCR	103660	ANDC with rotate right
ANDCS	103760	ANDC with swap halves of result
ANDL	103500	AND with rotate left
ANDO	103440	AND but carry bit is 1
ANDOL	103540	ANDO with rotate left
ANDOR	103640	ANDO with rotate right
ANDOS	103740	ANDO with swap halves of result
ANDR	103600	AND with rotate right
ANDS	103700	AND with swap halves of result
ANDZ	103420	AND but carry bit is 0
ANDZL	103520	ANDZ with rotate left

ASSEMBLY MNEMONIC	MACINE CODE	COMMENTS
ANDZR	103620	ANDZ with rotate right
ANDZS	103720	ANDZ with swap halves of result
COM	100000	Complement ACS into ACD; CRY is carry bit
COMC	100060	COM but complement CRY is carry bit
COMCL	100160	COMC with rotate left
COMCR	100260	COMC with rotate right
COMCS	100360	COMC with swap halves of result
COML	100100	COM with rotate left
COMO	100040	COM but carry bit is 1
COMOL	100140	CCMO with rotate left
COMOR	100240	COMO with rotate right
COMOS	100340	COMO with swap halves of result
COMR	100200	COM with rotate right
COMS	100300	COM with swap halves of result
COMZ	100020	COM but carry bit is 0
COMZL	100120	COMZ with rotate left
COMZR	100220	COMZ with rotate right
COMZS	100320	COMZ with swap halves of result
DIA	060400	Input, A buffer data to AC
DIAC	060600	DIA and clear device
DIAP	060700	DIA and send pulse to device
DIAS	060500	DIA and start device
DIB	061400	Input, B buffer data to AC
DIBC	061600	DIB and clear device
DIBP	061700	DIB and send pulse to device
DIBS	061500	DIB and start device
DIC	062400	Input, C buffer data to AC
DICC	062600	DIC and clear device

ASSEMBLY MNEMONIC	MACHINE CODE	COMMENTS
DICP	062700	DIC and send pulse to device
DICS	062500	DIC and start device
DIV	073101	AC0 and AC1 divided by AC2. Overflow sets Carry. Quotient in AC1, remainder in AC0.
DOA	061000	Output AC data to buffer A
DOAC	061200	DOA and clear device
DOAP	061300	DOA and send pulse to device
DOAS	061100	DOA and start device
DOB	062000	Output AC data to buffer B
DOBC	062200	DOB and clear device
DOBP	062300	DOB and send pulse to device
DOBS	062100	DOB and start device
DOC	063000	Output AC data to buffer C
DOCC	063200	DOC and clear device
DOCP	063300	DOC and send pulse to device
DOCS	063100	DOC and start device
DSZ	014000	Subtract 1 from the contents of E, skip if result is zero.
HALT	063077	Halt the processor
INC	101400	Place ACS+1 in ACD, CRY is carry bit base
INCC	101460	INC but complement CRY is base
INCCL	101560	INCC with rotate left
INCCR	101660	INCC with rotate right
INCCS	101760	INCC with swap halves of result
INCL	101500	INC with rotate left
INCO	101440	INC but 1 is base for carry bit
INCOL	101540	INCO with rotate left
INCOR	101640	INCO with rotate right

ASSEMBLY MNEMONIC	MACHINE CODE	COMMENTS
INCOS	101740	INCO with swap halves of result
INCR	101600	INC with rotate right
INCS	101700	INC with swap halves of result
INCZ	101420	INC but 0 is base for carry bit
INCZL	101520	INCZ with rotate left
INCZR	101620	INCZ with rotate right
INCZS	101720	INCZ with swap halves of result
INTA	061477	Acknowledge interrupt by loading code of nearest device requesting an interrupt into bits 10-15 of AC
INTDS	060277	Disable interrupts, clear Interrupt On flag
INTEN	060177	Enable interrupts, set Interrupt On flag
IORST	062677	Clear I/O devices and Interrupt On flag, set RTC to line frequency
ISZ	010000	Add 1 to contents of E, skip if zero result
JMP	000000	Jump to location E
JSR	004000	Save PC+1 in AC3 and jump to location E
LDA	020000	Load contents of E into AC
MOV	101000	Load ACS into ACD, carry bit is CRY
MOVC	101060	MOV but carry bit is CRY complement
MOVCL	101160	MOVC with rotate left
MOVCR	101260	MOVC with rotate right
MOVCS	101360	MOVC with swap halves of result
MOVL	101100	MOV with rotate left
MOVO	101040	MOV but carry bit is 1
MOVOL	101140	MOVO with rotate left
MOVOR	101240	MOVO with rotate right

ASSEMBLY MACHINE

MNEMONIC	CODE	COMMENTS
MOVOS	101340	MOVO with swap halves of result
MOVR	101200	MOV with rotate right
MOVS	101300	MOV with swap halves of result
MOVZ	101020	MOV but carry bit is 0
MOVZL	101120	MOVZ with rotate left
MOVZR	101220	MOVZ with rotate right
MOVZS	101320	MOVZ with swap halves of result
MSKO	062077	Set Interrupt Disable flags to AC mask
MUL	073301	Multiply AC1 by AC2, add AC0, result in AC0 and AC1
NEG	100400	Place negative ACS in ACD, CRY is carry bit base
NEGC	100460	NEG but complement CRY is base
NEGCL	100560	NEGC with rotate left
NEGCR	100660	NEGC with rotate right
NEGCS	100760	NEGC with swap halves of result
NEGL	100500	NEG with rotate left
NEGO	100440	NEG but 1 is base for carry bit
NEGOL	100540	NEGO with rotate left
NEGOR	100640	NEGO with rotate right
NEGOS	100740	NEGO with swap halves of result
NEGR	100600	NEG with rotate right
NEGS	100700	NEG with swap halves of result
NEGZ	100420	NEG but 0 is base for carry bit
NEGZL	100520	NEGZ with rotate left
NEGZR	100620	NEGZ with rotate right
NEGZS	100720	NEGZ with swap halves of result
NIO	060000	No operation
NIOC	060200	Clear device

ASSEMBLY MNEMONIC	MACHINE CODE	COMMENTS
NIOP	060300	Send pulse to device
NIOS	060100	Start device
READS	060477	Read console data switches into AC
SBN	000007	Skip if carry and result are zero appended to arithmetic and logical instructions
SEZ	000006	Skip if carry or result are zero appended to arithmetic and logical instructions
SKP	000001	Skip, add 1 to PC appended to arithmetic and logical instructions
SKPBN	063400	Skip if Busy is 1
SKPBZ	063500	Skip if Busy is 0
SKPDN	063600	Skip if Done is 1
SKPDZ	063700	Skip if Done is 0
SNC	000003	Skip if carry bit is 1 appended to arithmetic and logical instructions
SNR	000005	Skip if result is nonzero appended to arithmetic and logical instructions
STA	040000	Store AC in location E
SUB	102400	Subtract ACS from ACD, result in ACD carry bit based on CRY

ASSEMBLY MNEMONIC	MACHINE CODE	COMMENTS
SUBC	102460	SUB but complement CRY is base
SUBCL	102560	SUBC with rotate left
SUBCR	102660	SUBC with rotate right
SUBCS	102760	SUBC with swap halves of result
SUBL	102500	SUB with rotate left
SUBO	102440	SUB but 1 is base for carry bit
SUBOL	102540	SUBO with rotate left
SUBOR	102640	SUBO with rotate right
SUBOS	102740	SUBO with swap halves of result
SUBR	102600	SUB with rotate right
SUBS	102700	SUB with swap,halves of result
SUBZ	102420	SUB but 0 is base for carry bit
SUBZL	102520	SUBZ with rotate left
SUBZR	102620	SUBZ with rotate right
SUBZS	102720	SUBZ with swap halves of result
SZC	000002	Skip if carry is 0 appended to arithmetic and logical instructions
SZR	000004	Skip if result is 0 appended to arithmetic and logical instructions
@	002000	Indirect addressing
#	000010	Inhibit carry and result loading

[Ref. 2]

APPENDIX N

INSTRUCTION EXECUTION TIMES

When twc numbers are given, the one at the left of the slash is the time for an isolated transfer, the one at the right is the minimum time between consecutive transfers. Times are in microseconds.

INSTRUCTION	TIME
LDA, STA	1.6
ISZ, DSZ	1.8
JMP, JSR	0.8
Indirect addressing add	0.8
Autoindexing add	0.2
COM, NEG, INC	0.8*
ADC, SUB, ADD, AND	0.8*
*If skip occures add	0.2
I/O input (except INTA)	2.2#
NIO, I/O output	2.2*
#S, C, or P add	0.6
I/O skips	1.4*
INTA	2.2
MUL, DIV	8.8
Unsuccessful	1.6
Interrupt with multiply/divide	10.6
without multiply/divide	4.6

INSTRUCTION	TIME
Data Channel	
Input, Output	2.0
Increment	2.2
Latency	3.6
High speed channel	
Input	0.8
Output	0.8/1.0
Increment	1.0/1.2
Latency	
With I/O	3.6
Without I/O	2.0

[Ref. 2]

APPENDIX O

I/O DEVICE CODES AND MNEMONICS

DEVICE CODE	MNEMONIC	PRIORITY MASK	DEVICE
00	--	--	Power Fail
01	MDV	--	Multiply/Divide
02	MMPU	--	Memory Management and Protection Unit
02	MAP0	--	Memory Allocation
03	MAP1	--	and
04	MAP2	--	Protection
05			
06	MCAT	12	Multiprocessor adapter transmitter
07	MCAR	12	Multiprocessor adapter receiver
10	TTI	14	Teletype input
11	TTO	15	Teletype output
12	PTR	11	Paper tape reader
13	PTP	13	Paper tape punch
14	RTC	13	Real time clock option
15	PLT	12	Incremental plotter
16	CDR	10	Card reader
17	LPT	12	Line printer

DEVICE CODE	MNEMONIC	PRIORITY MASK	DEVICE
20	DSK	09	Fixed head disk
21	ADCV	08	A/D converter
22	MTA	10	Magnetic tape
23	DACV	--	D/A converter
24	DCM	00	Data communications mux.
25			
26			
27			
30	QTY	14	Asynchronous hardware mux.
31	IBM1	13	IBM 360/370 interface
32	IBM2	13	IBM 360/370 interface
33	DKP	07	Moving head disc
34	CAS	10	Cassette tape
34	MX1	10	Multiline asynchronous
35	MX2	11	controller
36	IPB	06	Interprocessor bus
37	IVT	06	IPB watchdog timer
40	DPI	08	IPB full-duplex input
41	DPO	08	IPB full-duplex output
42	DIO	07	Digital I/O
43	DIOT	06	Digital I/O timer
44	MXM	12	MX1/2 modem control
45			
46	MCAT1	12	Second MCAT
47	MCAR1	12	Second MCAR

(© Data General Corporation 1975. Reproduced from NOVA® AND ECLIPSE LINE COMPUTERS by permission of Data General Corporation, Southboro, MA)

DEVICE CODE	MNEMONIC	PRIORITY MASK	DEVICE
50	TTI1	14	Second TTI
51	TTO1	15	Second TTO
52	PTR1	11	Second PTR
53	PTP1	13	Second PTP
54	RTC1	13	Second RTC
55	PLT1	12	Second PLT
56	CDR1	10	Second CDR
57	LPT1	12	Second LPT
60	DSK1	09	Second DSK
61	ADCV1	08	Second ADCV
62	MTA1	10	Second MTA
63	DACV1	--	Second DACV
64	FPU1	05	Alternate location
65	FPU2	05	for
66	FPU4	05	floating point
67			
70	QTY1	14	Second QTY
71		13	Second IBM1
72		13	Second IBM2
73	DKP1	07	Second DKP
74		11	Second MX1
75		11	Second MX2
74	FPU1	05	or
75	FPU2	05	Floating
76	FPU	05	Point
77	CPU	--	Central processor and console functions

[Ref. 6]

APPENDIX P

ASCII CODE

EVEN PAR. BIT	7-BIT OCTAL CODE	CHARACTER	COMMENTS
0	000	NUL	Null, tape feed, CTRL shift P
1	001	SOH	Start heading or message, CTRL A
1	002	STX	Start text or end of address, CTRL B
0	003	ETX	End text or message, CTRL C
1	004	EOT	End transmission, CTRL D
0	005	ENQ	Enquire identification, CTRL E
0	006	ACK	Acknowledge, RU, CTRL F
1	007	BEL	Ring bell, CTRL G
1	010	BS	Backspace, CTRL H
0	011	HT	Horizontal tab,CTRL I
0	012	LF	Line feed, CTRL J
1	013	VT	Vertical tab, CTRL K
0	014	FF	Form feed, new page, CTRL L
1	015	CR	Carriage return, CTRL M
1	016	SO	Shift ribbon to red, CTRL N
0	017	SI	Shift ribbon to black, CTRL O
1	020	DLE	CTRL P
0	021	DC1	CTRL Q
0	022	DC2	CTRL R

EVEN PAR. BIT	7-BIT OCTAL CODE	CHARACTER	COMMENTS
1	023	DC3	CTRL S
0	024	DC4	CTRL T
1	025	NAK	Error, CTRL U
1	026	SYN	CTRL V
0	027	ETB	End of block, CTRL W
0	030	CAN	Cancel, CTRL X
1	031	EM	CTRL Y
1	032	SUB	CTRL Z
0	033	ESC	Escape, CTRL shift K
1	034	FS	File separator, CTRL shift L
0	035	GS	Group separator, CTRL shift M
0	036	RS	Record separator, CTRL shift N
1	037	US	CTRL shift O
1	040	SP	Space
0	041	!	
0	042	"	
1	043	#	
0	044	$	
1	045	%	
1	046	&	Ampersand
0	047	'	Apostrophe, accent acute
0	050	(	
1	051	)	
1	052	*	
0	053	+	
1	054	,	
0	055	-	

EVEN PAR. BIT	7-BIT OCTAL CODE	CHARACTER	COMMENTS
0	056	.	
1	057	/	
0	060	0	
1	061	1	
1	062	2	
0	063	3	
1	064	4	
0	065	5	
0	066	6	
1	067	7	
1	070	8	
0	071	9	
0	072	:	
1	073	;	
0	074	<	
1	075	=	
1	076	>	
0	077	?	
1	100	@	
0	101	A	
0	102	B	
1	103	C	
0	104	D	
1	105	E	
1	106	F	
0	107	G	
0	110	H	

EVEN PAR. BIT	7-BIT OCTAL CODE	CHARACTER	COMMENTS
1	111	I	
1	112	J	
0	113	K	
1	114	L	
0	115	M	
0	116	N	
1	117	O	
0	120	P	
1	121	Q	
1	122	R	
0	123	S	
1	124	T	
0	125	U	
0	126	V	
1	127	W	
1	130	X	
0	131	Y	
0	132	Z	
1	133	[	Shift K
0	134	|	Shift L
1	135	]	Shift M
1	136	|	Up arrow
0	137	-	Back arrow
0	140	'	Accent grave
1	141	a	
1	142	b	
0	143	c	

EVEN PAR. BIT	7-BIT OCTAL CODE	CHARACTER	COMMENTS
1	144	d	
0	145	e	
0	146	f	
1	147	g	
1	150	h	
0	151	i	
0	152	j	
1	153	k	
0	154	l	
1	155	m	
1	156	n	
0	157	o	
1	160	p	
0	161	q	
0	162	r	
1	163	s	
0	164	t	
1	165	u	
1	166	v	
0	167	w	
0	170	x	
1	171	y	
1	172	z	
0	173	{	
1	174	|	
0	175	}	
0	176	-	Special symbol

EVEN PAR. BIT	7-BIT OCTAL CODE	CHARACTER	COMMENTS
1	177	DEL	Delete, rub out
-	---	REPT	Repeats any other key while held
-	---	LOC LF	Local line feed
-	---	LOC CR	Local carriage return
-	---	BREAK	Continuous null string
-	---	HERE IS	Null string

[Ref. 2]

APPENDIX Q

ERROR CODES

A. CORE IMAGE LOADER/WRITER ERRORS

The loader/writer will type *ERR with a code in AC0. The following list describes the error condition indicated by a one in the status word bit position. [Refs. 12 and 13]

BIT	MEANING
1	Data late
3	Illegal command
5	Lateral parity error in a word
6	Addressed tape is beyond the EOT marker
8	Addressed tape is at load point
10	Bad tape
13	Unit is write locked
14	Odd number of bytes detected

B. CLI ERRORS

ERROR MESSAGE	MEANING
FILE NON-EXISTENT	Attempt to load or save an illegal cassette file.
ILLEGAL FILE NAME	Attempt to rewind or make a save file on a non-existent unit
PHASE ERROR CHECKSUM ERROR	Errors in saving a file.
NOT ENOUGH ARGUMENTS	
ILLEGAL COMMAND FOR DEVICE	
DEVICE IS READ PROTECTED	

[Ref. 12]

C. EDIT ERRORS

ERROR MESSAGE	MEANING
BUFFER CAPACITY EXCEEDED DURING COMMAND INPUT. COMMAND IS TERMINATED AND BEING EXECUTED.	Command string exceeds capacity of edit buffer.
BUFFER IS FULL - CANNOT DO A	Attempting to append a page when the buffer is full.
BUFFER IS FULL - Y OR A INPUT TERMINATED.	During a read, buffer capacity has been exceeded. A partial page has been read in.
FILE CAN'T BE USED FOR INPUT	Attempt to read a read-protected file.
FILE CAN'T BE USED FOR OUTPUT	Attempt to write a write-protected file.
ILLEGAL FILE NAME	File name does not conform to a legal file name.
MACRO ERROR	Undefined or reursive macro.

ERROR MESSAGE	MEANING
NO OUTPUT FILE	Attempt to issue output command without first opening an output file.
NO SUCH FILE	Attempt to specify an input file which doesn't exist.
OUTPUT ALREADY ACTIVE	Attempt to get for writing an output file which has not been closed, and is still active.
PARITY ERROR IN LINE n	Read parity error in line n. Bad character replaced by |.
STR NOT FOUND	Unsuccessful string search.
??command string	Illegal edit command.

[Refs. 14 and 15]

D. ASM ERRORS

ERROR MESSAGE	MEANING
NO.END	No .END statement
I/O ERROR nn	nn is the error code
1	Illegal file name
7	Read-protected file
10	Write-protected file
12	Non-existent file

[Ref. 12]

E. RLDR ERRORS

ERROR MESSAGE	MEANING
NO INPUT FILE SPECIFIED.	
NO SAVE FILE SPECIFIED.	No core image output device has been specified with /S.
SAVE FILE IS READ/WRITE PROTECTED.	The save file must permit both reading and writing. (cassette only)
I/O ERROR nn	See ASM errors.

[Ref. 12]

F. LFE ERRORS

ERROR MESSAGE	MEANING
M	The same first 5 characters in two or more entry symbols.
U	An undefined external entry.
P	An external entry defined before its reference.
ILLEGAL KEY: key	Indicated letter is not legal.
SWITCH ERROR: u	Indicated file use is not permitted with this operation.
TOO MANY ARGUMENTS IN COMMAND LINE	The 200 character command line buffer is exceeded.
NO INPUT FILE?	This operation requires an input file.
NO OUTPUT FILE?	This operation requires an output file.
ERROR CONDITION IN INPUT FILE: inputfile	Incorrect device mnemonic.
ERROR CONDITION IN OUTPUT FILE: outputfile	Incorrect device mnemonic.
ERROR CONDITION IN UPDATE FILE: updatefile	Incorrect device mnemonic.
ERROR CONDITION IN LISTING FILE: listingfile	Incorrect device mnemonic.

```
CHECKSUM ERROR IN LOGICAL RECORD: recordfile
               Bad paper tape.
CHECKSUM ERROR IN UPDATE FILE: updatefile
               Bad paper tape.
BLOCK ERROR IN UPDATE FILE: updatefile
               Improper input block format.
BLOCK ERROR IN LOGICAL RECORD: inputfile
               Improper Logical Record format.
LOGICAL RECORD NOT RECOVERABLE: recordfile
               Different file types cannot be
                input from the same device.
UPDATE FILE NOT FOUND FOR L.R: logicalfile
               R command requires both an
                Update and a Logical Record
                file to be specified.
SYMBOL TABLE OVERFLOW
               I command has insufficient
                memory space available.
UNEXPECTED ERROR FROM SYSTEM
               Hardware malfunction.
LOGICAL RECORD NOT FOUND: recordfile
NO LISTING FILE: DEFAULT LISTING ON TTO
```

[Ref. 16]

G. SYSG ERRORS

ERROR MESSAGE	MEANING
NOT ENOUGH ARGUMENTS	
OUTPUT FILE WRITE PROTECTED, FILE: filename	
NO OUTPUTFILE SPECIFIED	
ILLEGAL SYMBOL NAME: symbol	Invalid character in command line.
FILE DOES NOT EXIST, FILE: filename	
UNEXPECTED SYSTEM ERROR	Computer halts with system error code in AC2.

[Refs. 12 and 13]

H. SYSTEM ERRORS

A system error results in a computer halt with a code in AC2 that is interpreted as follows:

CODE	MEANING
0	Illegal channel number
1	Illegal file name
2	Illegal system command
3	Illegal command for device
4	Not a saved file
5	Attempted to write an existent file
6	End of file
7	Read protected file
10	Write protected file
11	Attempt to create an existent file
12	Non-existent file
13	Attempt to alter a permanent file
14	Attributes protected
15	File nct opened
21	Attempt to use a UFT already in use
22	Line limit exceeded
23	Attempt to restore a non-existent image
24	Parity error on read line
25	Trying to push too many levels
26	Not enough memory available
27	Out of file space

CODE	MEANING
30	File read error
31	Unit not prperly selected
32	Illegal starting address
33	Attempt to read into system area
35	Files specified on different directories
36	Illegal device name
37	Illegal overlay number
40	Illegal overlay file attribute
41	User set time error
42	Out of TCB's
43	Signal to busy address
44	Squash file error
45	Device already in system
46	Insufficient contiguous blocks
47	Quantity error
50	Error in user task queue table
100	Not enough arguments
101	Illegal attribute
102	No debug address
103	No continuation address
104	No starting address
105	Checksum error
106	No source file specified
107	Not a command
110	Illegal block type
111	No files match specifier
112	Phase error
113	Too many arguments

[Refs. 12 and 13]

APPENDIX R

TELETYPE OUTPUT EXAMPLE PROGRAM CREATION

A. TTO EXAMPLE PROGRAM SOURCE LISTING

```
#0:2
*GRCT1:0$$
*I;*********************************************
;
; TTO EXAMPLE PROGRAM
;
;PROGRAM FOR TTY OUTPUT OF A PACKED BUFFER,
;TERMINATED BY A ZERO WORD.
;       EXAMPLE:        WORD    CHARACTER
;       BUFER:          1       2,1
;                       2       4,3
;       TERM:           3       0,0
;
;*********************************************
         .TITL   TTOEX
         .ENT    TTOEX
         .NREL
TTOEX:   LDA     3,PBUF  ;TABLE POINTER
LOOP:    LDA     0,0,3   ;PASS PARAMETER TO ACO
         MOV#    0,0,SNR ;CHECK FOR TERMINATION CODE
         HALT            ;WAIT UNTIL AVAILABLE
         JMP     .-1
         DOAS    0,TTO   ;OUTPUT RIGHT-MOST CHARACTER
```

```
        MOVS    0,0      ;SWAP CHARACTERS
        SKPBZ   TTO      ;WAIT UNTIL AVAILABLE
        JMP     .-1
        DOAS    0,TTO    ;2ND CHARACTER
        INC     3,3      ;NEXT BUFFER ADDRESS
        JMP     LOOP     ;NEXT WORD
BUFER:  .TXT    '<015><012>CONGRATULATIONS!
                <015><012><040><040>YOU<040>
                HAVE<040>COMPLETED<040>
                YOUR<040>FIRST<040>PROGRAM
                <040>CREATION.<000><000>'
PBUF:   BUFER
        .END    TTOEX
$$
*GWCT1:0$$
*B$P$GC$$
*H$$

#
```

B. TTO EXAMPLE PROGRAM ASSEMBLER OUTPUT

```
#0:3
ASM 1 CT1:0 CT0:0/B $TTO/L

PROGRAM IS RELOCATABLE


---
  0001  TTOEX
                    ;*******************************************
                    ;
                    ; TTO EXAMPLE PROGRAM
                    ;
                    ;PROGRAM FOR TTY OUTPUT OF A PACKED BUFFER,
                    ;TERMINATED BY A ZERO WORD.
                    ;       EXAMPLE:        WORD     CHARACTER
                    ;       BUFER:          1        2,1
                    ;                       2        4,3
                    ;       TERM:           3        0,0
                    ;
                    ;*******************************************
                            .TITL   TTOEX
                            .ENT    TTOEX
                            .NREL
00000'034465 TTOEX:         LDA     3,PBUF  ;TABLE POINTER
00001'021400 LOOP:          LDA     080,3   ;PASS PARAMETER TO
                                            ;AC0
00002'101015                MOV#    0,0,SNR ;CHECK FOR
                                            ;TERMINATION CODE
00003'063077                HALT            ;TERMINATE
00004'063511                SKPBZ   TTO     ;WAIT UNTIL AVAILABLE
00005'000777                JMP     .-1
00006'061111                DOAS    0,TTO   ;OUTPUT RIGHT-MOST
```

```
                                                CHARACTER
00007'101300            MOVS    0,0             ;SWAP CHARACTERS
00010'063511            SKPBZ   TTO             ;WAIT UNTIL AVAILABLE
00011'000777            JMP     .-1
00012'061111            DOAS    0,TTO           ;2ND CHARACTER
00013'175400            INC     3,3             ;NEXT BUFFER ADDRESS
00014'000765            JMP     LOOP            ;NEXT WORD
                BUFER:  .TXT    '<015><012>CONGRATULATIONS!
00015'005015
00016'047503
00017'043516
00020'040522
00021'052524
00022'040514
00023'044524
00024'047117
00025'020523
00026'004411                    <015><012><040><040>YOU<040>
00027'005015
00030'020040
00031'047531
00032'020125
00033'004411                    HAVE<040>COMPLETED<040>
00034'040510
00035'042526
00036'041440
00037'046517
00040'046120
00041'052105
00042'042105
00043'004440                    YOUR<040>FIRST<040>PROGRAM
00044'054411
00045'052517
00046'020122
00047'044506
00050'051522
```

```
00051'020124
00052'051120


---
  0002   TTOEX
00053'043517
00054'040522
00055'004515                         <040>CREATION.<000><000>'
00056'020011
00057'051103
00060'040505
00061'044524
00062'047117
00063'000056
00064'000000
00065'000015'  PBUF:    BUFER
      000000'           .END     TTOEX


---
  0003   TTOEX
BUFER 000015'
LOOP  000001'
PBUF  000065'
TTOEX 000000'


ASM
#
```

C. TTO EXAMPLE PROGRAM RLDR OUTPUT

```
#0:4
RLDR CT1:0 CT0:0/S $TTO/L
 TTOEX
    NMAX 000526
    ZMAX 000050
    CSZE
     EST
     SST

   TTOEX 000440


OK

#
```

D. TTO EXAMPLE PROGRAM EXECUTION

```
# 0:0
CONGRATULATIONS!
  YOU HAVE COMPLETED YOUR FIRST PROGRAM CREATION.
```

APPENDIX S

ASSORTED PROGRAMS

A. ADCOD

```
                  ;*********************************************
                  ;
                  ; A/D CODE TEST
                  ;
                  ; PROCEDURE: CONNECT ANALOG SIGNAL ON CH Ø
                  ;            SIGNAL IS CONVERTED AND TYPED
                  ;            BY ACTUATING THE CPU SWITCH
                  ;           'CONTINUE'
                  ; LIMITS: +/- 1Ø.24 VOLTS
                  ;
                  ; 11 OCTOBER 1976,GDR
                  ;
                  ;************************************************
                          .TITL   ADCOD
                          .ENT    ADCOD
                          .EXTN   BIOA,TYPE
                          .NREL
ØØØØØ'ØØ6433 ADCOD:       JSR     @LTYPE
ØØØØ1'ØØØØ35'             PROMT
ØØØØ2'Ø2Ø534 ADCO1:       LDA     Ø,ZERO ;CH Ø IN AC Ø
ØØØØ3'Ø63121              DOCS    Ø,21   ;START A/D SAMPLE
ØØØØ4'Ø63621              SKPDN   21     ;WAIT FOR COMPLETION
ØØØØ5'ØØØ777              JMP     .-1
ØØØØ6'Ø66421              DIC     1,21   ;LOAD DIGITAL DATA
ØØØØ7'Ø3Ø421              LDA     2,CR ;USE NEW LINE EACH SAMPLE
ØØØ1Ø'Ø71111              DOAS    2,TTO
ØØØ11'Ø63611              SKPDN   TTO
ØØØ12'ØØØ777              JMP     .-1
ØØØ13'Ø3Ø416              LDA     2,LF
ØØØ14'Ø71111              DOAS    2,TTO
ØØØ15'Ø63611              SKPDN   TTO
ØØØ16'ØØØ777              JMP     .-1
ØØØ17'Ø2Ø415              LDA     Ø,LSTUF ;PARAMETER FOR BIOA
ØØØ2Ø'ØØ6412              JSR     @LBIOA
ØØØ21'Ø63Ø77              HALT           ;RESTART BY CONSOLE
ØØØ22'ØØØ76Ø              JMP     ADCO1  ;SWITCH 'CONTINUE'
ØØØ23'Ø54514 STUFF:       STA    .3,RSTUF ;SAVE RETURN
ØØØ24'Ø61111              DOAS    Ø,TTO  ;OUTPUT DATA
ØØØ25'Ø63611              SKPDN   TTO
ØØØ26'ØØØ777              JMP     .-1
ØØØ27'ØØ251Ø              JMP     @RSTUF  ;RETURN
ØØØ3Ø'ØØØØ15 CR:          Ø15
ØØØ31'ØØØØ12 LF:          Ø12
ØØØ32'177777 LBIOA:       BIOA
ØØØ33'177777 LTYPE:       TYPE
ØØØ34'ØØØØ23'LSTUF:       STUFF
             PROMT:       .TXT '<Ø15><Ø12>A/D<Ø4Ø>CODE<Ø4Ø>[illegible]
```

```
00035'005015
00036'027501
00037'020104
00040'047503
00041'042504
00042'042440
00043'044103
00044'004517                    <015><012>PROCEDURE:<015><012>
00045'006411
00046'050012
00047'047522
00050'042503
00051'052504
00052'042522
00053'006472
00054'004412         CONNECT<040>ANALOG<040>SOURCE<040>
00055'041411
00056'047117
00057'042516
00060'052103
00061'040440
00062'040516
00063'047514
00064'020107
00065'047523
00066'051125
00067'042503
00070'004440         TO<040>CH0<015><012>LIMITS:<040>
00071'052011
00072'020117
00073'044103
00074'006460
00075'046012
00076'046511
00077'052111
00100'035123
00101'004440         +/-10.24V<015><012>NEW<040>SAMPLE
00102'025411
00103'026457
00104'030061
00105'031056
00106'053064
00107'005015
00110'042516
00111'020127
00112'040523
00113'050115
00114'042514
00115'004411         <040>BY<040>CONSOLE<040>SWITCH<040>
00116'041040
00117'020131
00120'047503
00121'051516
```

```
00122'046117
00123'020105
00124'053523
00125'052111
00126'044103          CONTINUE<015><012>'
00127'004440
00130'041411
00131'047117
00132'044524
00133'052516
00134'006505
00135'000012
00136'000000 ZERO:    0
00137'000000 RSTUF:   0
      000000'         .END ADCOD
```

B.   BIOA

```
                ;*************************************************
                ;
                ; SUBROUTINE BIOA
                ;
                ; DATA GENERAL  BINO.SR  090-000032-01
                ; MODIFIED FOR NPS USE
                ;
                ; BINARY TO OCTAL ASCII CONVERT
                ; CONVERTS A 16-BIT BINARY WORD TO AN OCTAL
                ; CHARACTER STRING
                ;        INPUT:   USER ROUTINE IN AC0
                ;                 BINARY NUMBER IN AC1
                ; OUTPUT:ASCII CHARACTER STRING, TERMINATED BY
                ;         NULL CHARACTER
                ;         CHARACTERS PASSED RIGHT ADJUSTED
                ;         TO THE USER ROUTINE WHOSE ADDRESS
                ;         MUST BE STORED IN AC0
                ;         STRING OF FORM:
                ;         OOOOOO(NULL)
                ;         WHERE "O'S" REPRESENT OCTAL DIGITS
                ; CALLING SEQUENCE:
                ;        .EXTN    BIOA
                ;        -----
                ;        JSR      @LBIOA
                ;        RETURN
                ;        -----
                ;LBIOA: BIOA
                ;
                ;************************************
                       .TITL    BIOA
                       .ENT     BIOA
                       .NREL
00000'040425 BIOA:     STA      0,.EF40 ;LINK USER ROUTINE
00001'054421           STA      3,.EF03 ;SAVE RETURN
00002'152621           SUBZR 2,2,SKP    ; 100000 TO AC2
00003'146401 .EF99:    SUB 2,1,SKP      ; DECREASE CURRENT DIGIT
00004'020420 .EF98:    LDA 0,.EF20      ; GET OCTAL 57
00005'101400           INC 0,0         ; FORM ASCII OUTPUT DIGIT
00006'146533           SUBZL# 2,1,SNC;  - IMPLIES DIGIT COMPLETE
00007'000774           JMP .EF99    ; NOT DONE, SUBTRACT 1 FROM
                                    ; CURRENT DIGIT
00010'050413           STA 2,.EF10; SAVE SUBTRACT CONSTANT
00011'006414           JSR @.EF40 ; PUT OUT A DIGIT
00012'030411           LDA 2,.EF10; RESTORE SUBTRACT CONSTANT
00013'151220           MOVZR 2,2    ; POSITION "1" FOR NEXT OCTAL
                                    ; DIGIT
00014'151220           MOVZR 2,2
00015'151224           MOVZR 2,2,SZR
00016'000766           JMP .EF98        ; NOT DONE
```

```
00017'141000           MOV 2,0
00020'006405           JSR @.EF40; PUT OUT NULL CHARACTER
00021'002401           JMP @.EF03; RETURN
00022'000000 .EF03:    0          ; SAVE RETURN
00023'000000 .EF10:    0          ; SAVE LOCATION FOR SUBTRACT
                                  ; CONSTANT
00024'000057 .EF20:    57         ; ASCII CONSTANT
00025'000000 .EF40:    0          ;USER ROUTINE LINKAGE
                       .END       ;INSERTED 17 SEPT 76
```

C. TYPE

```
             ;*******************************************:
             ;
             ; SUBROUTINE  TYPE
             ;
             ;GLOBAL SUBROUTINE FOR TTY OUTPUT OF A PACKED
             ;BUFFER,TERMINATED BY A ZERO WORD.BUFFER
             ;START ADDRESS PASSED AS ARGUMENT.
             ;         EXAMPLE:          WORD     CHARACTER
             ;         BUFFER:           1        2,1
             ;                           2        4,3
             ;         TERM:             3        0,0
             ;CALL SEQUENCE
             ;         .EXTN    TYPE
             ;         ...
             ;         JSR      @LTYPE
             ;         BUFFER
             ;         ...
             ;LTYPE: TYPE
             ;
             ;********************************************
                       .TITL    TYPE
                       .ENT     TYPE
                       .NREL
00000'054421 TYPE:     STA      3,LINK    ;SAVE RETURN
00001'035400           LDA      3,0,3     ;TABLE POINTER
00002'021400 LOOP:     LDA      0,0,3     ;PASS PARAMETER TO A0
00003'101015           MOV#     0,0,SNR   ;CHECK FOR TERMINATION
00004'000412           JMP      TYPEX     ;TERMINATE
00005'063511           SKPBZ    TTO       ;WAIT UNTIL AVAILABLE
00006'000777           JMP      .-1
00007'061111           DOAS     0,TTO     ;OUTPUT RIGHT-MOST CHAR
00010'101300           MOVS     0,0       ;SWAP CHARACTERS
00011'063511           SKPBZ    TTO       ;WAIT UNTIL AVAILABLE
00012'000777           JMP      .-1
00013'061111           DOAS     0,TTO     ;2ND CHARACTER
00014'175400           INC      3,3       ;NEXT BUFFER ADDRESS
00015'000765           JMP      LOOP      ;NEXT WORD
             ;
00016'034403 TYPEX:    LDA      3,LINK
00017'175400           INC      3,3
00020'001400           JMP      0,3
00021'000000 LINK:     0
                       .END
```

D. CADO

```
              ;*******************************************
              ;
              ; ROUTINE CADO
              ;
              ; FOR CALIBRATION OF THE A/D CONVERTER
              ; BY ANALOG INPUT TO CHANNEL ZERO AND
              ; MONITORING SPECIFIC BIT OUTPUTS ON AN
              ; OSCILLOSCOPE
              ;
              ; LOGIC LEVELS: 1= 2.7V (>2.2V)
              ;               0= 0.0V (<0.2V)
              ;
              ; A TELETYPE MESSAGE REMINDS THE OPERATOR
              ; OF THE CORRECT PROCEDURE.
              ;
              ; 11 OCTOBER 1976,GDR
              ;
              ;**********************************************
                      .TITL   CADO
                      .ENT    CADO
                      .EXTN   TYPE
                      .ZREL
00000-006007-CADO:    JSR     @LTYPE
00001-000010-         PROMT
00002-020261-CADO1:   LDA     0,ZERO,0        ;CH 0 IN AC 0
00003-063121          DOCS    0,21     ;START A/D CONVERSION
00004-063621          SKPDN   21       ;WAIT FOR COMPLETION
00005-000004-         JMP     .-1
00006-000002-         JMP     CADO1,0  ;RELOOP
00007-177777 LTYPE:   TYPE
             PROMT:   .TXT    '<015><012>A/D<040>CALIBRATION
00010-005015
00011-027501
00012-020104
00013-040503
00014-044514
00015-051102
00016-052101
00017-047511
00020-004516                   <015><012>PROCEDURE:<015><012>
00021-006411
00022-050012
00023-047522
00024-042503
00025-052504
00026-042522
00027-006472
00030-004412                   CONNECT<040>ANALOG<040>VOLTAGE
```

```
00031-041411
00032-047117
00033-042516
00034-052103
00035-040440
00036-040516
00037-047514
00040-020107
00041-047526
00042-052114
00043-043501
00044-004505
00045-020011     <040>TO<040>CHANNEL<040>0
00046-047524
00047-041440
00050-040510
00051-047116
00052-046105
00053-030040
00054-004411
00055-005015     <015><012>SET<040>VOLTAGE<040>AT
00056-042523
00057-020124
00060-047526
00061-052114
00062-043501
00063-020105
00064-052101
00065-004411
00066-026440     <040>-0.0025V<015><012>MONITOR<040>
00067-027060
00070-030060
00071-032462
00072-006526
00073-046412
00074-047117
00075-052111
00076-051117
00077-004440
00100-046411     MSB<040>BIT<040>PIN<040>28L<040>
00101-041123
00102-041040
00103-052111
00104-050040
00105-047111
00106-031040
00107-046070
00110-004440
00111-047411     ON<040>OSCILLOSCOPE<015><012>ADJUST
00112-020116
00113-051517
00114-044503
00115-046114
```

```
00116-051517
00117-047503
00120-042520
00121-005015
00122-042101
00123-052512
00124-052123
00125-004411                <040>OFFSET<040>FOR<040>50<040>
00126-047440
00127-043106
00130-042523
00131-020124
00132-047506
00133-020122
00134-030065
00135-004440                PERCENT<040>DUTY<040>CYCLE
00136-050011
00137-051105
00140-042503
00141-052116
00142-042040
00143-052125
00144-020131
00145-054503
00146-046103
00147-004505                <015><012>RESET<040>VOLTAGE<040>
00150-006411
00151-051012
00152-051505
00153-052105
00154-053040
00155-046117
00156-040524
00157-042507
00160-004440                TO<040>-10.2375V<015><012>MONITOR
00161-052011
00162-020117
00163-030455
00164-027060
00165-031462
00166-032467
00167-006526
00170-046412
00171-047117
00172-052111
00173-051117
00174-004411                <040>LSB<040>BIT<040>PIN<040>34L
00175-046040
00176-041123
00177-041040
00200-052111
00201-050040
00202-047111
00203-031440
```

```
00204-046064
00205-004411                  <015><012>ADJUST<040>RANGE<040>
00206-005015
00207-042101
00210-052512
00211-052123
00212-051040
00213-047101
00214-042507
00215-004440                  FOR<040>50<040>PERCENT<040>DUTY
00216-043011
00217-051117
00220-032440
00221-020060
00222-042520
00223-041522
00224-047105
00225-020124
00226-052504
00227-054524
00230-004411                  <040>CYCLE<015><012><012>LOGIC
00231-041440
00232-041531
00233-042514
00234-005015
00235-046012
00236-043517
00237-041511
00240-004411                  <040>LEVELS:<040>1=2.7V<040>
00241-046040
00242-053105
00243-046105
00244-035123
00245-030440
00246-031075
00247-033456
00250-020126
00251-004411                  AND<040>0=0.0V<015><012>'
00252-047101
00253-020104
00254-036460
00255-027060
00256-053060
00257-005015
00260-000000
00261-000000 ZERO:    0
     000000-          .END    CADO
```

E. DAC

```
           ;*********************************************************
           ;
           ; ROUTINE DAC
           ;
           ; FOR CALIBRATION OF THE D/A CONVERTER
           ; BY MONITORING THE APPROPRIATE CHANNEL(X OR Y)
           ; WITH A VOLTMETER AND ADJUSTING THE
           ; ZERO POTENTIOMETER FOR THE MINIMUM VALUE AND
           ; THE GAIN POTENTIOMETER FOR THE DESIRED
           ; RANGE AT MAXIMUM VALUE.(10.000 VOLTS)
           ;
           ; A TELETYPE MESSAGE REMINDS THE OPERATOR OF
           ; THE CORRECT PROCEDURE.
           ;
           ; 21 OCTOBER 1976,GDR
           ;
           ;*********************************************************
                   .TITL   DAC
                   .ENT    DAC
                   .EXTN   TYPE
                   .NREL
00000'006424 DAC:  JSR     @LTYPE  ;INTRODUCTION
00001'000027'      INTRO
00002'006422 MIN:  JSR     @LTYPE
00003'000237'      ZEROV
00004'030422       LDA     2,ZERO,0        ;DATA 0 IN AC 2
00005'004410       JSR     DAC1
00006'063077       HALT            ;CONTINUE FOR FULL
00007'006415 MAX:  JSR     @LTYPE  ;SCALE CALIBRATION
00010'000275'      TENV
00011'030414       LDA     2,TEN,0 ;DATA 10.000V IN AC 2
00012'004403       JSR     DAC1
00013'063077       HALT            ;CONTINUE FOR ZERO
00014'000766       JMP     MIN     ;CALIBRATION
00015'020411 DAC1: LDA     0,ZERO,0        ;CH X(=0) IN AC
00016'105400       INC     0,1     ;CH Y(=1) IN AC 1
00017'062023       DOB     0,23    ;SELECT X CHANNEL
00020'071023       DOA     2,23    ;OUTPUT X
00021'066023       DOB     1,23    ;SELECT Y CHANNEL
00022'071023       DOA     2,23    ;OUTPUT Y
00023'001400       JMP     0,3     ;RETURN
00024'177777 LTYPE: TYPE
00025'003777 TEN:  03777
00026'000000 ZERO: 0
             INTRO: .TXT   '<015><012>D/A<040>CALIBRATION
00027'005015
00030'027504
00031'020101
00032'040503
00033'044514
```

```
034'051102
035'052101
036'047511
037'004516          <015><012>PROCEDURE:<015><012>
040'006411
041'050012
042'047522
043'042503
044'052504
045'042522
046'006472
047'004412          CONNECT<040>DVM<040>TO<040>CHX
050'041411
051'047117
052'042516
053'052103
054'042040
055'046526
056'052040
057'020117
060'044103
061'004530          <040>(PINS8U+9L)<015><012>ADJUST
062'020011
063'050050
064'047111
065'034123
066'025525
067'046071
070'006451
071'040412
072'045104
073'051525
074'004524          <040>CHX<040>ZERO<040>POTENTIOMETER
075'020011
076'044103
077'020130
00'042532
01'047522
02'050040
03'052117
04'047105
05'044524
06'046517
07'052105
10'051105
11'004411          <015><012>REPEAT<040>PROCEDURE<040>
12'005015
13'042522
14'042520
15'052101
16'050040
17'047522
20'042503
21'052504
```

```
00122'042522
00123'004440                              FOR<040>CHY<040>(PINS6U+6L)
00124'043011
00125'051117
00126'041440
00127'054510
00130'024040
00131'044520
00132'051516
00133'052466
00134'033053
00135'024514
00136'004411                              <015><012>FULL<040>SCALE<040>
00137'005015
00140'052506
00141'046114
00142'051440
00143'040503
00144'042514
00145'004440                       CALIBRATION<040>STARTS<040>BY<040>
00146'041411
00147'046101
00150'041111
00151'040522
00152'044524
00153'047117
00154'051440
00155'040524
00156'052122
00157'020123
00160'054502
00161'004440                       DEPRESSING<040>CONTINUE<040>ON<040>
00162'042011
00163'050105
00164'042522
00165'051523
00166'047111
00167'020107
00170'047503
00171'052116
00172'047111
00173'042525
00174'047440
00175'020116
00176'004411                       THE<040>CONSOLE<015><012>ADJUST<040>
00177'044124
00200'020105
00201'047503
00202'051516
00203'046117
00204'006505
00205'040412
00206'045104
00207'051525
```

```
00210'020124
00211'004411
00212'040507                        GAIN<040>POTENTIOMETERS<040>AS<040>
00213'047111
00214'050040
00215'052117
00216'047105
00217'044524
00220'046517
00221'052105
00222'051105
00223'020123
00224'051501
00225'004440
00226'040411                        APPROPRIATE.<015><012><000><000>'
00227'050120
00230'047522
00231'051120
00232'040511
00233'042524
00234'006456
00235'000012
00236'000000
                  ZEROV:   .TXT      '<015><012>ADJUST<040>ZERO
00237'005015
00240'042101
00241'052512
00242'052123
00243'055040
00244'051105
00245'004517                         <040>AND<040>CONTINUE<040>FOR
00246'020011
00247'047101
00250'020104
00251'047503
00252'052116
00253'047111
00254'042525
00255'043040
00256'051117
00257'004411                         <040>FULL<040>SCALE<040>ADJUST
00260'043040
00261'046125
00262'020114
00263'041523
00264'046101
00265'020105
00266'042101
00267'052512
00270'052123
00271'004411                         <015><012><000><000>'
00272'005015
00273'000000
00274'000000
```

```
                  TENV:      .TXT      '<015><012>TEN<040>VOLT<040>
00275'005015
00276'042524
00277'020116
00300'047526
00301'052114
00302'004440                           ADJUST<040>AND<040>CONTINUE
00303'040411
00304'045104
00305'051525
00306'020124
00307'047101
00310'020104
00311'047503
00312'052116
00313'047111
00314'042525
00315'004411                           <040>FOR<040>ZERO<040>ADJUST
00316'043040
00317'051117
00320'055040
00321'051105
00322'020117
00323'042101
00324'052512
00325'052123
00326'004411                           <015><012><000><000>'
00327'005015
00330'000000
00331'000000
      000000'                .END      DAC
```

F. EXERCISE 4 SOLUTION

```
                 ;GLOBAL SUBROUTINE FOR TTY OUTPUT OF A PACKED
                 ;BUFFER,TERMINATED BY A ZERO WORD.BUFFER
                 ;START ADDRESS PASSED AS ARGUMENT.
                 ;        EXAMPLE:        21
                 ;                        43
                 ;                        00
                 ;CALL SEQUENCE
                 ;        .EXTN   TYPE
                 ;        ...
                 ;        JSR     @LTYPE
                 ;        BUFER
                 ;        ...
                 ;LTYPE:  TYPE
                 ;
                          .TITL   TYPE
                          .ENT    TYPE
                          .NREL
00000'054421 TYPE:        STA     3,LINK    ;SAVE RETURN
00001'035400              LDA     3,0,3     ;TABLE POINTER
00002'021400 LOOP:        LDA     0,0,3     ;PASS PARAMETER TO A0
00003'101015              MOV#    0,0,SNR   ;CHECK FOR TERMINATION
00004'000412              JMP     TYPEX     ;TERMINATE
00005'063511              SKPBZ   TTO       ;WAIT UNTIL AVAILABLE
00006'000777              JMP     .-1
00007'061111              DOAS    0,TTO     ;OUTPUT RIGHT-MOST CHAR
00010'101300              MOVS    0,0       ;SWAP CHARACTERS
00011'063511              SKPBZ   TTO       ;WAIT UNTIL AVAILABLE
00012'000777              JMP     .-1
00013'061111              DOAS    0,TTO     ;2ND CHARACTER
00014'175400              INC     3,3       ;NEXT BUFFER ADDRESS
00015'000765              JMP     LOOP      ;NEXT WORD
             ;
00016'034403 TYPEX:       LDA     3,LINK
00017'175400              INC     3,3
00020'001400              JMP     0,3
00021'000000 LINK:        0
             ;
             ;TEST ROUTINE
             ;
00022'006463 TESTR:       JSR     @LTYPE
00023'000025'             BUFER
00024'063077              HALT
             BUFER:.TXT"<015><012>GRANT<040>DOUGLAS<040>RALPH
00025'005015
00026'051107
00027'047101
00030'020124
00031'047504
00032'043525
```

```
00033'040514
00034'020123
00035'040522
00036'050114
00037'004510
00040'006411
00041'031012
00042'020064                    <015><012>24<040><040><040>RALSTON<040>
00043'020040
00044'040522
00045'051514
00046'047524
00047'020116
00050'004411
00051'042040
00052'044522                      <040>DRIVE<015><012>MONTEREY<040>
00053'042526
00054'005015
00055'047515
00056'052116
00057'051105
00060'054505
00061'004440
00062'041411
00063'046101                     CALIFORNIA<015><012>U.S.A.<040><040>
00064'043111
00065'051117
00066'044516
00067'006501
00070'052412
00071'051456
00072'040456
00073'020056
00074'004440
00075'020011
00076'020040                     <040><040><040><040><040><040>93940'
00077'020040
00100'034440
00101'034463
00102'030064
00103'000000
00104'000000 TERM:   0
00105'000000'LTYPE:  TYPE
      000022'        .END TESTR
```

G. EXERCISE 5 SOLUTION

```
                 ;*********************************************
                 ; INTERRUPT INITIALIZATION
                 ;*********************************************
                         .TITL    INIT
                         .ENT     INIT
                         .EXTN    INTRUP
      000001             .LOC     1
00001 177777             INTRUP
                         .NREL
00000'020410 INIT:       LDA      0,MASK     ;SET PRIORITY MASK
00001'062077             MSKO     0
00002'020405             LDA      0,HZ       ;SET CLOCK FREQUENCY
00003'061114             DOAS     0,RTC
00004'060177             INTEN
00005'000400             JMP      .
00006'000777             JMP      .-1
00007'000001 HZ:         1                   ;BASIC CLOCK OF 10HZ
00010'177773 MASK:       177773              ;ENABLE RTC=BIT 13
      000000'            .END INIT
                 ;*********************************************
                 ; MOD FOR RTC ONLY
                 ; ROUTINE TO SERVICE I/O INTERRUPTS BY POLLING
                 ;*********************************************
                         .TITL    INTRUP
                         .ENT     INTRUP
                         .EXTN    SUPR
                         .NREL
00000'060277 INTRUP:     INTDS
00001'040421             STA      0,SAV0     ;SAVE THE STATE
00002'044421             STA      1,SAV1
00003'050421             STA      2,SAV2
00004'054421             STA      3,SAV3
00005'101100             MOVL     0,0        ;SAVE THE CARRY
00006'040420             STA      0,SAVK
00007'063714             SKPDZ    RTC
00010'006411             JSR      @LSUPR     ;CLOCK REQUEST
00011'020415             LDA      0,SAVK     ;REFRESH CARRY
00012'101200             MOVR     0,0
00013'020407             LDA      0,SAV0     ;REFRESH STATE
00014'024407             LDA      1,SAV1
00015'030407             LDA      2,SAV2
00016'034407             LDA      3,SAV3
00017'060177             INTEN
00020'002000             JMP      @0
00021'177777 LSUPR:      SUPR
00022'000000 SAV0:       0
00023'000000 SAV1:       0
00024'000000 SAV2:       0
00025'000000 SAV3:       0
00026'000000 SAVK:       0
                         .END
```

```
                     ;*****************************************
                     ;
                     ; MOD FOR RTC ONLY
                     ;
                     ; SUPERVISOR BY RTC INTERRUPT
                     ;
                     ;*****************************************
                             .TITL SUPR
                             .ENT  SUPR
                             .EXTN EXEC1
                             .NREL
00000'054452 SUPR:   STA    3,RSUPR  ;SAVE RETURN
00001'152440         SUBO   2,2      ;CLEAR AC2 FOR JOB COUNT
00002'034446 NUJOB:  LDA    3,PJOBK  ;GET THE JOB COUNT
00003'157000         ADD    2,3
00004'021400         LDA    0,0,3
00005'105404         INC    0,1,SZR  ;TASK ASSIGNED?
00006'000424         JMP    NUTSK    ;NO
00007'040444 TASK:   STA    0,SAVE0  ;SAVE STATE
00010'044444         STA    1,SAVE1
00011'050444         STA    2,SAVE2
00012'054444         STA    3,SAVE3
00013'125100         MOVL   1,1      ;SAVE THE CARRY
00014'044443         STA    1,SAVEK
00015'034434         LDA    3,PNUK   ;GET REFRESH COUNT
00016'157000         ADD    2,3
00017'025400         LDA    1,0,3
00020'044434         STA    1,SAVE1  ;REFRESH SAVED JOB COUNT
00021'034426         LDA    3,PJOB   ;SERVICE JOB REQUEST
00022'157000         ADD    2,3
00023'007400         JSR    @0,3
00024'024433         LDA    1,SAVEK  ;REFRESH CARRY
00025'125200         MOVR   1,1
00026'020425         LDA    0,SAVE0  ;REFRESH STATE
00027'024425         LDA    1,SAVE1
00030'030425         LDA    2,SAVE2
00031'034425         LDA    3,SAVE3
00032'045400 NUTSK:  STA    1,0,3    ;UPDATE JOB COUNT
00033'151400         INC    2,2      ;ON TO NEXT JOB
00034'034411         LDA    3,NJOB   ;NUMBER OF JOBS
00035'156414         SUB#   2,3,SZR  ;LAST JOB CHECKED?
00036'000744         JMP    NUJOB    ;NO
00037'030403         LDA    2,HZZ    ;SET CLOCK FREQUENCY
00040'071114         DOAS   2,RTC
00041'002411         JMP    @RSUPR   ;TERMINATE
00042'000001 HZZ:    1               ;BASIC CLOCK OF 10 HZ
00043'177777 JOB:    EXEC1           ;JOB 1
00044'177766 JOBK:   -12             ;NEGATIVE COUNT
```

```
00045'000001 NJOB:    1
00046'177766 NUK:     -12
00047'000043'PJOB:    JOB
00050'000044'PJOBK:   JOBK
00051'000046'PNUK:    NUK
00052'000000 RSUPR:   0
00053'000000 SAVE0:   0
00054'000000 SAVE1:   0
00055'000000 SAVE2:   0
00056'000000 SAVE3:   0
00057'000000 SAVEK:   0
                      .END
```

```
                ;*********************************************
                ;
                ; EXEC1 = EXEC2
                ;
                ;GLOBAL ROUTINE TO TEST RTC SYSTEM BY COUNT 0-9
                ;
                ;*********************************************
                        .TITL   EXEC2   ;EXEC1 IS THE CALL
                        .ENT    EXEC1
                        .EXTN   TYPE
                        .NREL
00000'054420 EXEC1:     STA     3,REXEC1        ;SAVE RETURN
00001'006414            JSR     @LTYPE  ;PRINT NUMBER COUNT
00002'000012'           CRLF
00003'020410            LDA     0,COUNT
00004'101400            INC     0,0     ;INCREASE THE COUNT AND
00005'024411            LDA     1,NINE  ;CHECK FOR LARGEST DIG
00006'106472            SUBC#   0,1,SZC
00007'020410            LDA     0,ZERO
00010'040403            STA     0,COUNT
00011'002407            JMP     @REXEC1
00012'005215 CRLF:      005215          ;<012><015>
             COUNT:     .TXT    @<060><000>@
00013'000060
00014'000000
00015'177777 LTYPE:     TYPE
00016'000071 NINE:      71
00017'000060 ZERO:      60
00020'000000 REXEC1:    0
                        .END
```

H. EXERCISE 6 SOLUTION

```
          ;*******************************************************
          ;
          ; CASET
          ;
          ;ROUTINE TO DEMONSTRATE CASSETTE I/O VIA SOS TO A
          ;SCRATCH TAPE ON UNIT 1
          ;
          ;
          ; 27 OCTOBER 1976,GDR
          ;
          ;*******************************************************
                        .TITL    CASET
                        .ENT     CASET
                        .EXTN    .SOS,.CTU1,TYPE,TYPIØ,BIOA
                        .NREL
       ØØØ2Ø2 BUFF:     .BLK     2Ø2
ØØ2Ø2'ØØØØØØ FILE:      Ø
ØØ2Ø3'ØØ6474 CASET:     JSR      @LTYPE   ;PROMPT THE USER
ØØ2Ø4'ØØØ32Ø'           PROMPT
ØØ2Ø5'ØØ6473            JSR      @LTYPI   ;INPUT ENABLED
ØØ2Ø6'ØØØ352'           BUFFER
ØØ2Ø7'ØØ6Ø17            .SYSTM            ;BEGIN SOS CALLS
ØØ21Ø'Ø21Ø22            .SYSI             ;INITIALIZE SOS
ØØ211'ØØØ436            JMP      ERROR
ØØ212'Ø2Ø77Ø            LDA      Ø,FILE   ;FILE=Ø
ØØ213'ØØ6Ø17            .SYSTM
ØØ214'Ø14Ø31            .OPEN    31       ;CT1=31
ØØ215'ØØØ432            JMP      ERROR
ØØ216'Ø2Ø456            LDA      Ø,BYPTØ
ØØ217'Ø24457            LDA      1,BYTCT
ØØ22Ø'ØØ6Ø17            .SYSTM
ØØ221'Ø16431            .WRS     31       ;RECORD THE BUFFER
ØØ222'ØØØ425            JMP      ERROR
ØØ223'ØØ6Ø17            .SYSTM
ØØ224'Ø14431            .CLOSE   31
ØØ225'ØØØ422            JMP      ERROR
ØØ226'Ø2Ø754            LDA      Ø,FILE
ØØ227'ØØ6Ø17            .SYSTM
ØØ23Ø'Ø14Ø31            .OPEN    31
ØØ231'ØØØ416            JMP      ERROR
ØØ232'Ø2Ø443            LDA      Ø,BYPT1
ØØ233'Ø24443            LDA      1,BYTCT
ØØ234'ØØ6Ø17            .SYSTM
ØØ235'Ø15Ø31            .RDS     31       ;DUMP THE RECORDING
ØØ236'ØØØ411            JMP      ERROR
ØØ237'ØØ6441            JSR      @LTYPE   ;PROMPT USER
```

```
ØØ24Ø'ØØØ343'           VERIFY
ØØ241'ØØ6436            JSR      @LTYPE;PRINT CONTENTS OF BUFFER
ØØ242'ØØØØØ1'           BUFF
ØØ243'ØØ6434            JSR      @LTYPE
ØØ244'ØØØ3Ø2'           NOTE
ØØ245'Ø62677            IORST
ØØ246'ØØ2433            JMP      @SYS
ØØ247'1454ØØ ERROR:     MOV      2,1     ;ERROR CODE TO AC 1
ØØ25Ø'Ø3Ø42Ø            LDA      2,CR
ØØ251'Ø71111            DOAS     2,TTO
ØØ252'Ø63611            SKPDN    TTO
ØØ253'ØØØ777            JMP      .-1
ØØ254'Ø3Ø415            LDA      2,LF
ØØ255'Ø71111            DOAS     2,TTO
ØØ256'Ø63611            SKPDN    TTO
ØØ257'ØØØ777            JMP      .-1
ØØ26Ø'Ø2Ø4Ø3            LDA      Ø,STUFF ;BIOA USER LINKAGE
ØØ261'ØØ6411            JSR      @LBIOA
ØØ262'Ø63Ø77            HALT
ØØ263'Ø5441Ø STUFF:     STA      3,RSTUF ;SAVE RETURN
ØØ264'Ø61111            DOAS     Ø,TTO   ;OUTPUT DATA
ØØ265'Ø63611            SKPDN    TTO
ØØ266'ØØØ777            JMP      .-1
ØØ267'ØØ24Ø4            JMP      @RSTUF
ØØ27Ø'ØØØØ15 CR:        Ø15
ØØ271'ØØØØ12 LF:        Ø12
ØØ272'177777 LBIOA:     BIOA
ØØ273'ØØØØØØ RSTUF:     Ø
ØØ274'ØØØ724"BYPTØ:     2*BUFFER
ØØ275'ØØØØØØ"BYPT1:     2*BUFF
ØØ276'ØØØ2Ø2 BYTCT:     2Ø2         ;TOTAL BUFFER = 13Ø CHARACTERS
ØØ277'177777 LTYPE:     TYPE
ØØ3ØØ'177777 LTYPI:     TYPIØ
ØØ3Ø1'Ø17777 SYS:       17777
             NOTE:      .TXT     '<Ø15><Ø12>NORMAL<Ø4Ø>
ØØ3Ø2'ØØ5Ø15
ØØ3Ø3'Ø47516
ØØ3Ø4'Ø46522
ØØ3Ø5'Ø461Ø1
ØØ3Ø6'ØØ444Ø                     TERMINATION<Ø15><Ø12><Ø><Ø>'
ØØ3Ø7'Ø52Ø11
ØØ31Ø'Ø511Ø5
ØØ311'Ø44515
ØØ312'Ø4Ø516
ØØ313'Ø44524
ØØ314'Ø47117
ØØ315'ØØ5Ø15
ØØ316'ØØØØØØ
ØØ317'ØØØØØØ
             PROMPT:    .TXT     '<Ø15><Ø12>INPUT,TERMINATE BY
```

```
00320'005015
00321'047111
00322'052520
00323'026124
00324'042524
00325'046522
00326'047111
00327'052101
00330'020105
00331'054502
00332'004411                [HERE<040>[S]<015><012><0><0><0>'
00333'044133
00334'051105
00335'020105
00336'051511
00337'006535
00340'000012
00341'000000
00342'000000
                  VERIFY:  .TXT '<015><012>BUFFER<015><012><0><0>'
00343'005015
00344'052502
00345'043106
00346'051105
00347'005015
00350'000000
00351'000000
       000202 BUFFER: .BLK 202       ;LIMIT TO 130 CHARACTERS
00554'000000 ZERO:    0    ;BUFFER PROTECTION FOR CASSETTES
       000203'         .END CASET
```

```
                  ;***********************************************
                  ; SUBROUTINE TYPIØ
                  ;GLOBAL SUBROUTINE FOR TTY INPUT TO A PACKED
                  ;BUFFER,TERMINATED BY 'HERE IS'.BUFFER
                  ;START ADDRESS PASSED AS ARGUMENT.
                  ;BYTE COUNT RETURNED IN AC 2.
                  ;AUTOMATICALLY TERMINATES AFTER 13Ø(DECIMAL)
                  ;        EXAMPLE:          WORD     CHARACTER
                  ;        BUFFER:           1        2,1
                  ;                          2        4,3
                  ;        TERM:             3        Ø,Ø
                  ;CALL SEQUENCE
                  ;        .EXTN    TYPIØ
                  ;        ...
                  ;        JSR      @LTYPI
                  ;        BUFFER
                  ;        ...
                  ;LTYPI: TYPIØ

                  ;***********************************************
                          .TITL TYPIØ
                          .ENT  TYPIØ
                          .NREL
ØØØØØ'1714ØØ TYPIØ:   INC    3,2        ;STEP PAST ARGUMENT
ØØØØ1'Ø5Ø431          STA    2,RTYPI    ;SAVE RETURN
ØØØØ2'Ø3Ø431          LDA    2,ZERO     ;ZERO CHARACTER COUNT
ØØØØ3'Ø354ØØ          LDA    3,Ø,3      ;TABLE POINTER
ØØØØ4'Ø6Ø11Ø          NIOS   TTI        ;ENABLE INPUT
ØØØØ5'12644Ø NUWRD:   SUBO   1,1        ;CLEAR HOLDING BUFFER
ØØØØ6'Ø6361Ø CHAR2:   SKPDN  TTI        ;WAIT FOR INPUT
ØØØØ7'ØØØ777          JMP    .-1
ØØØ1Ø'Ø6Ø51Ø          DIAS   Ø,TTI      ;SAVE IN AØ
ØØØ11'Ø63511          SKPBZ  TTO        ;WAIT FOR ECHO
ØØØ12'ØØØ777          JMP    .-1
ØØØ13'Ø61111          DOAS   Ø,TTO      ;ECHO PRINT
ØØØ14'1Ø7365          ADDCS  Ø,1,SNR    ;1ST WORD=C1,Ø,CRY SET
                                        ;2ND WORD=C2,C1,CRY CLEAR
ØØØ15'ØØØ413          JMP    TYPIX      ;EXIT ON 1ST OR 2ND NULL
ØØØ16'125ØØ2          MOV    1,1,SZC    ;CRY SET, FIRST CHARACTER
ØØØ17'ØØØ767          JMP    CHAR2
ØØØ2Ø'Ø454ØØ          STA    1,Ø,3      ;STORE 2 CHAR/WORD
ØØØ21'1514ØØ          INC    2,2        ;MAINTAIN CHARACTER COUNT
ØØØ22'1514ØØ          INC    2,2
ØØØ23'Ø244Ø6          LDA    1,LIMIT
ØØØ24'132415          SUB#   1,2,SNR    ;LIMIT OF 13Ø CHARACTERS
ØØØ25'ØØØ4Ø3          JMP    TYPIX      ;TERMINATE ON FULL BUFFER
ØØØ26'1754ØØ          INC    3,3        ;NEXT BUFFER OPENING
ØØØ27'ØØØ756          JMP    NUWRD      ;CONTINUE
             ;
ØØØ3Ø'ØØ24Ø2 TYPIX:   JMP    @RTYPI
ØØØ31'ØØØ2Ø2 LIMIT:   2Ø2
ØØØ32'ØØØØØØ RTYPI:   Ø
ØØØ33'ØØØØØØ ZERO:    Ø
                      .END
```

APPENDIX T

MAINTENANCE

This Appendix summarizes a few random notes on maintaining the equipment.

CPU and CASSETTE TRANSPORT

The nearest source of manufacturer assistance is:

Data General Corporation,
Field Service Office,
1054 Elwell Court,
Palo Alto, CA 94306

General office: phone 965-9100
Local sales: Tom Larson phone 965-1010
Software support: Jim Isaak phone 965-1010
Hardware support: Denis Hutchinson / Steve Parell

Naval Postgraduate School:
Software support: Dave Norman X2641
Supply and Repair: Al Gilkes X2422
Comptroller: Donna McNicol X2770

The CPU and cassette units need no maintenance by the normal user. Periodically the cooling fans should be lubricated. It is recommended that during frequent use periods the cassette heads be cleaned once a week. A solvent such as ETHYL alcohcl is suggested.

ASR 33 TELETYPE

For normal maintenance routine advice see the micro-computer laboratory technicians Walt and Don on the fifth floor of Spanagel Hall.

The striker arm for the TTY type-drum has a rubber cap to protect the raised character outlines. Loss of this cap rapidly destroys the TTY print capability. The fault may be detected by the louder metallic sounding striking of the type-arm. Continuous user inspection is recommended.

DATAX CONVERTERS

The nearest source of manufacturer assistance is:

Repair Services,
Data Translation Corporation,
23 Strathmore Road,
Natck, MA 01760

General assistance: Mr. Fishman phone 617-655-5300

INITIAL DISTRIBUTION LIST

1.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
2.	Department Chairman, Code 62 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	6
3.	Data General Corporation Route 9 Southboro, Massachusetts 01772	1
4.	Major Grant D. Ralph, CF DMCS 2 National Defense Headquarters Ottawa, Ontario, K1A OK2 CANADA	2

APPENDIX U

LIST OF MANUALS

014-000001-02
4025 IBM System 360/370 Interface

014-000003-01
Summary of Terminal and Data Set Interfaces

014-000005-01
Type 4015 Synchronous communications Controller

014-000008-00
How to Order Cables for the NOVA® Computers

014-000011
How to Wire the TTY 4009 Modification Kit

014-000013-04
How to Install and Use the NOVA® Cassette System

014-000014-00
Communications Cabling

014-000015-02
Programmable Synchronous Line Adapters

015-000009-00

How to Use the NOVA® Computers

015-000015

NOVA® Cassette Preliminary Technical Manual

015-000021-02

Peripherals Programmers Reference Manual

015-000023-03

Programers Reference Manual

015-000031-02

Interface Designers Reference Manual

015-000043-00

NOVA® 800/820 and Jumbo 800 Computers Technical Manual

017-000001-01

Synchronous Communications Package

017-000004-01

Remote Synchronous Terminal Control Program

093-000002-01

Bootstrap Loader User's Manual

093-000003-06

Binary Loader User's Manual

093-000017-02

Assembler

093-000018-05

NOVA® Text Editor

093-000018-06

NOVA® Text Editor

093-000020-02

Debug II User's Manual

093-000038-01

Debug I User's Manual

093-000039-00

Relocatable Loader

093-000040-00

Extended Assembler

093-000041-03

Relocatable Math Library File

093-000042-01

Single User Basic

093-000044-02

Debug III User's Manual

093-000052-02

Extended Algol User's Manual

093-000053-05

Fortran IV User's Manual

093-000053-07

Fortran IV User's Manual

093-000054-00

NOVA® Assembler for the IBM 360

093-000055-03

Selfloading Bootstrap and Binary Loader

093-000062-03

The Stand-Alone Operating System User's Manual

093-000062-04

The Stand-Alone Operating System User's Manual

093-000065-02

Extended Basic User's Manual

093-000067-01

Introduction to Programming the NOVA® Computers

093-000074-01

Library File Editor

093-000081-02

Macro-Assembler User's Manual

093-000083-04

Introduction to Real Time Disk Operating System

093-000084-00

Octal Editor

093-000090-01

Fundamentals of Small Computer Programming

MISCELLANEOUS

Authorized ADP Schedule Price List for 1976

Supplemental Price List for 1976

DG/DAC Sensor I/O Subsystem 012-244

NOVA® Cassette System Information Package

Datax User Instruction Guide number 1600-674-01

Point Plotter Dual D/A Converter Model DT212

Datax High Speed Data Acquisition System Modules

APPENDIX V

RELOCATABLE BINARY UTILITY PROGRAMS

089-000031-03
Relocatable Debug II

089-000046-05
Relocatable FPI

089-000073-02
Extended Debug III

089-000080-02
Relocatable Binary Punch

089-000081-04
Stand-Alone Library File Editor

089-000104-02
Stand-Alone Operating System Text Editor

089-000106-02
Stand-Alone Operating System Extended Assembler

089-000120-02
SOS Cassette/Magnetic Tape Relocatable Loader

089-000121-02
Stand-Alone Operating System Command Line Interpreter

089-000122-02

Stand-Alone Operating System Generation Program

089-000137-02

SOS Extended Basic MP

089-000138-02

SOS Extended Basic RP

089-000139-02

SOS Extended Basic PUR

089-000156-02

Extended Basic Software Multiply/Divide

089-000159-01

SOS Extended Basic Dummy Print

089-000160-01

SOS Extended Basic Dummy Matrix

APPENDIX W

ASSEMBLER SOURCE SUBROUTINES

090-000010-02
System Subroutine Core Compare

090-000012-01
System Subroutine Single Precision Absolute Value

090-000013-01
System Subroutine Single Precision Signed Multiply

090-000014-01
System Subroutine Single Precision Signed Divide

090-000015-01
System Subroutine Double Precision Absolute Value

090-000016-01
System Subroutine Double Precision Signed Multiply/Divide

090-000017-02
System Subroutine Double Precision Addition

090-000018-01
System Subroutine Double Precision Subtraction

090-000019-01
System Subroutine Double Precision Negate

090-000020-01
System Subroutine Unsigned Multiply

090-000021-01
System Subroutine Unsigned Divide

090-000025-01
System Subroutine Logical Exclusive OR

090-000026-02
System Subroutine Logical Inclusive OR

090-000027-01
System Subroutine Single Precision Binary Coded Decimal To Binary

090-000028-01
System Subroutine Single Precision Binary To Binary Coded Decimal

090-000029-01
System Subroutine Single Precision Decimal To Binary

090-000030-01
System Subroutine Single Precision Binary To Decimal

090-000031-01
System Subroutine Single Precision Octal To Binary

090-000032-01
System Subroutine Single Precision Binary To Octal

090-000033-01
System Subroutine Double Precision Binary Coded Decimal To Binary

090-000034-01

System Subroutine Double Precision Binary To Binary Coded Decimal

090-000035-01

System Subroutine Double Precision Decimal To Binary

090-000036-02

System Subroutine Double Precision Binary To Decimal

090-000037-01

System Subroutine Parity Generator

090-000038-01

System Subroutine Binary To Gray Code

090-000039-01

System Subroutine Gray Code To Binary

090-000040-01

System Subroutine Random Number Generator

090-000043-01

System Subroutine Debug I

090-000257-05

System Subroutine Fortran Runtime Parameters

090-000498-04

System Subroutine Stand-Alone Parameters

090-000520

System Subroutine Real Time Operating System Parameters

090-000883-02

System Subroutine Real Time Disk Operating System Parameters

090-000889-01

System Subroutine Stand-Alone Operating System Parameters

090-001482-00

System Subroutine Instruction Definitions NOVA® Basic

090-001483-00

System Subroutine Instruction Definitions Floating Point Interpreter

090-001484-00

System Subroutine Definitions For The Operating System

APPENDIX X

ABSOLUTE BINARY UTILITY PROGRAMS

091-000001-07
Paper Tape Editor

091-000002-08
Paper Tape Assembler

091-000003-03
Debug II For 4K, 06200-07577

091-000004-04
Binary Loader For The Manually Loaded Bootstrap

091-000005-02
Binary Punch (High Core)

091-000006-01
Binary Punch (Low Core)

091-000007-02
Core Compare

091-000008-04
Tape Duplicator

091-000010-03
Debug II For 4K, 00400-01777

091-000012-08
Basic Floating Point For 4K, 05600-07577

091-000013-08
Extended Floating Point For 4K, 04100-07577

091-000014-01
Floating Point To Octal Converter

091-000016-04
Relocatable Loader

091-000017-07
Extended Assembler

091-000018-07
Single User Basic

091-000036
Selfloading Bootstrap And Binary Loader

091-000038-07
Extended Relocatable Loader

091-000052-03
Fortran Compiler For 8K

091-000057-04
Stand-Alone Library File Editor

091-000067-02
Stand-Alone Operating System Cassette Loader/Writer

091-000069-03
Stand-Alone Operating System Extended Assembler Without Mass Storage

091-000070-03
Stand-Alone Operating System Generation Program Without Mass Storage

091-000071-03
Stand-Alone Operating System Generation Program With Cassette

091-000072-03
Stand-Alone Operating System Command Line Interpreter With Cassette

091-000073-03
Stand-Alone Operating System Relocatable Loader With Cassette

091-000C77-02
Stand-Alone Operating System Suoroutine Extended Basic

091-000081-01
Real Time Operating System Generation Program

APPENDIX Y

HARDWARE TEST PROGRAMS

095-000002
Memory Address Test

095-000007
Checkerboard II

095-000011
Real Time Clock Test

095-000012
Exerciser

095-000016-05
Binary 4015 Communications Controller

095-000028-02
Binary 4026 DCM Multiplexor Diagnostic

095-000031
Checkerboard III

095-000035-03
Binary 4029 Communications Controller

095-000044-04
800/1200 Power Shut Down Test

095-000045-04

NOVA® 800 Logic Test

095-000048-04

NOVA® 800 Teletype Test

095-000073-02

Binary 4060 Quad TTY Multiplexor

095-000099-02

Supernova, NOVA® 800/1200 Multiply/Divide Test

APPENDIX Z

LIBRARY PROGRAMS

099-000001-02

Relocatable Math Library File

099-000005-07

Fortran Runtime Library I

099-000006-04

Fortran Runtime Lirary II

099-000007-05

Fortran Runtime Lirary III

099-000008-02

Runtime Library Software Multiply/Divide

099-000009-02

Supernova, NOVA® 800/1200 Runtime Library Hardware Multiply/Divide

099-000010-08

Stand-Alone Operating System Library

099-000011-02

Runtime Library NOVA® Hardware Multiply/Divide

099-000012-05
Algol Runtime Library I

099-000013-05
Algol Runtime Library II

099-000014-04
Algol Runtime Library III

099-000018-03
Fortran Runtime Library 0

099-000020-02
Dummy Stand-Alone Operating System Library

099-000021-03
Fortran Data Plot Library

099-000041-02
Stand-Alone Operating System Cassette Driver

099-000060-00
Real Time Operating System Task Monitor Library

099-000061-00
Real Time Operating System Handler Library

099-000062-00
Real Time Operating System Cassette Handler Library

099-000077-02
Stand-Alone Operating System Single User Extended Basic

LIST OF REFERENCES

1. Pounds, J. W., The Data General NOVA® 800 Minicomputer as a Digital Controller Master of Science Thesis, Naval Postgraduate School, Monterey, 1975.

2. Data General Corporation, Reference Manual 093-000067-01, Introduction to Programming the NOVA® Computers, 1972.

3. Data General Corporation, Reference Manual 015-000009-00, How to Use the NOVA® Computers, April 1971.

4. Data General Corporation, Reference Manual 015-000023-03, Programmer's Reference Manual NOVA® LINE COMPUTERS, January 1976.

5. Data General Corporation, Reference Manual 015-000021-02, Programmer's Reference Manual PERIPHERALS, July 1975.

6. Data General Corporation, Reference Manual 015-000031-02, Interface Designer's Reference Manual NOVA^{R} AND ECLIPSE LINE COMPUTERS, August 1975.

7. Data Translation Corporation, Engineering Specification 1600-674 Rev. 1, DATAX User Instruction Guide, 1974.

8. Data Translation Corporation, Data Sheet DT212, Point Plotter Dual D/A Converter,October 1974.

9. Data General Corporation, Reference Manual 093-000002-01, _User's Manual BOOTSTRAP LOADER_, August 1970.

10. Data General Corporation, Reference Manual 093-000003-06, _User's Manual BINARY LOADER PROGRAM_, January 1973.

11. Data General Corporation, Reference Manual 093-000055-03, _SELFLOADING BOOTSTRAP AND BINARY LOADER_, February 1973.

12. Data General Corporation, reference Manual 093-000062-03, _User's Manual THE STAND-ALONE OPERATING SYSTEM_, June 1973.

13. Data General Corporation, Reference Manual 093-000062-04, _User's Manual STAND-ALONE OPERATING SYSTEM_, July 1974.

14. Data General Corporation, Reference Manual 093-000018-05, _NOVA® TEXT EDITOR_, November 1972.

15. Data General Corporation, Reference Manual 093-000018-06, _NOVA® TEXT EDITOR_, June 1973.

16. Data General Corporation, Reference Manual 093-000074-01, _LIBRARY FILE EDITOR_, December 1972.